TESIS DOCTORAL

IMPLEMENTACIÓN EN HARDWARE DE SISTEMAS DE ALTA FIABILIDAD BASADOS EN METODOLOGÍAS ESTOCÁSTICAS

PROGRAMA DE DOCTORAT EN ENGINYERIA ELECTRÒNICA UPC-UIB

AUTOR

Sr. VICENTE JOSÉ CANALS GUINAND

DIRECTOR DE LA TESIS

Dr. JOSEP L. ROSSELLÓ SANZ

Universitat de les Illes Balears

DEPARTAMENT DE FÍSICA

UNIVERSITAT DE LES ILLES BALEARS

Palma de Mallorca, Mayo de 2012

AGRADECIMIENTOS

Deseo expresar mi más sincero agradecimiento a todas las personas que me han ayudado y apoyado a lo largo de la realización de esta tesis. En primer lugar, deseo agradecer a mi director de tesis José Luís Rosselló Sanz, que me ha guiado, apoyado y aconsejado a lo largo del desarrollo de este trabajo de investigación. Conjuntamente con mi director es de especial mención el apoyo y los consejos recibidos por Antoni Morro Gomila conjuntamente con el cual hemos llevado y llevamos a cabo nuestras investigaciones.

Debo reseñar mi profundo reconocimiento y agradecimiento a los profesores del área de conocimiento de Tecnología Electrónica Miquel Roca Adrover, Eugeni Isern Riutort y Eugeni García Moreno a su vez como responsable del área de conocimiento, por su apoyo en los momentos cruciales que han posibilitado que llegara a buen puerto el presente trabajo de investigación. A la vez hago participes de mi agradecimiento al resto de miembros del área de Tecnología Electrónica que me han apoyado a lo largo de estos años.

A la vez me gustaría agradecer particularmente el apoyo y la complicidad que he recibido a lo largo de estos años por los compañeros del despacho (F-014) de Tecnología Electrónica como son: Marcos Rosales, Kay Suenaga, Bartomeu Alorda, Salvador Barceló, André Sousa, Gabriel Torrens, Xavier Gili y en especial a Jaume Verd Martorell un referente científico y moral para todos nosotros. También es de agradecer el apoyo recibido por mis compañeros de departamento Rubén Santamarta y Ramón Pujol a lo largo de estos años. Así como el apoyo recibido por el maestro de laboratorio de la Escuela Politécnica Superior Mateu Fernández. A su vez debo agradecer especialmente el apoyo técnico y humano recibido por los *Servicios Científico Técnicos de la UIB*, encarnados en la figura de Raúl Sánchez Torres.

No podría concluir, sin reconocer el apoyo y la financiación recibida a lo largo de estos años por la "*Cátedra SAMPOL de Eficiencia y Gestión Energética*" la cual me ha permitido

compaginar mis tareas de investigación a la vez que se me ha ofrecido la oportunidad de aplicar mis conocimientos en el mundo real. Por ello me gustaría agradecer el apoyo recibido por Andréu Moià Pol y a Víctor Martínez Moll como co-directores cátedra en la UIB, a la vez que a los miembros de esa gran familia como es *SAMPOL Ingeniería y Obras S.A* encarnada en María Calafat (Responsable de I+D+I), Juan Antonio Vicens (Jefe de la división de energía) y D. Gabriel Sampol (Presidente de *SAMPOL Ingeniería y Obras S.A*).

En el marco personal debo agradecer a Raúl García y en especial a José Antonio Gobea por el apoyo recibido a largo de todos estos años.

Y por último, aunque no menos importante, deseo recordar y agradecer el apoyo recibido de mis padres.

RESUMEN

La sociedad actual demanda cada vez más aplicaciones computacionalmente exigentes y que se implementen de forma energéticamente eficiente. Esto obliga a la industria del semiconductor a mantener una continua progresión de la tecnología CMOS. No obstante, los expertos vaticinan que el fin de la era de la progresión de la tecnología CMOS se acerca, puesto que se prevé que alrededor del 2020 la tecnología CMOS llegue a su límite. Cuando ésta llegue al punto conocido como "Red Brick Wall", las limitaciones físicas, tecnológicas y económicas no harán viable el proseguir por esta senda. Todo ello ha motivado que a lo largo de la última década tanto instituciones públicas como privadas apostasen por el desarrollo de soluciones tecnológicas alternativas como es el caso de la nanotecnología (nanotubos, nanohilos, tecnologías basadas en el grafeno, etc.). En esta tesis planteamos una solución alternativa para poder afrontar algunos de los problemas computacionalmente exigentes. Esta solución hace uso de la tecnología CMOS actual sustituyendo la forma de computación clásica desarrollada por Von Neumann por formas de computación no convencionales. Éste es el caso de las computaciones basadas en lógicas pulsantes y en especial la conocida como computación estocástica, la cual proporciona un aumento de la fiabilidad y del paralelismo en los sistemas digitales.

En esta tesis se presenta el desarrollo y evaluación de todo un conjunto de bloques computacionales estocásticos implementados mediante elementos digitales clásicos. A partir de estos bloques se proponen diversas metodologías computacionalmente eficientes que mediante su uso permiten afrontar algunos problemas de computación masiva de forma mucho más eficiente. En especial se ha centrado el estudio en los problemas relacionados con el campo del reconocimiento de patrones.

Palabras Claves: Computación Estocástica, Reconocimiento de Patrones, Inteligencia Computacional, Redes Neuronales, Redes Neuronales Pulsantes, Caos, Generación de números aleatorios, VLSI, Dispositivos de lógica programable

ABSTRACT

Today's society demands the use of applications with a high computational complexity that must be executed in an energy-efficient way. Therefore the semiconductor industry is forced to maintain the CMOS technology progression. However, experts predict that the end of the age of CMOS technology progression is approaching. It is expected that at 2020 CMOS technology would reach the point known as "Red Brick Wall" at which the physical, technological and economic limitations of CMOS technology will be unavoidable. All of this has caused that over the last decade public and private institutions has bet by the development of alternative technological solutions as is the case of nanotechnology (nanotubes, nanowires, graphene, etc.). In this thesis we propose an alternative solution to address some of the computationally exigent problems by using the current CMOS technology but replacing the classical computing way developed by Von Neumann by other forms of unconventional computing. This is the case of computing based on pulsed logic and especially the stochastic computing that provide a significant increase of the parallelism and the reliability of the systems. This thesis presents the development and evaluation of different stochastic computing methodologies implemented by digital gates. The different methods proposed are able to face some massive computing problems more efficiently than classical digital electronics. This is the case of those fields related to pattern recognition, which is the field we have focused the main part of the research work developed in this thesis.

Keywords: Stochastic Computing, Pattern Recognition, Machine Learning, Neural Networks, Spiking Neural Networks, Chaos, Random Number Generation, FPGA, VLSI

ÍNDICE GENERAL

Pág.

ÍNDICE DE TABLAS

Pág.

- **CAPÍTULO 2**

- **CAPÍTULO 3**

- **CAPÍTULO 4**

ÍNDICE DE FIGURAS

Pág.

- **CAPÍTULO 2**

- **CAPÍTULO 3**

- **CAPÍTULO 4**

- **ANEXO A**

1. INTRODUCCIÓN Y OBJETIVOS DE LA TESIS

El presente capítulo tiene como objetivos el planteamiento del problema a resolver y establecer los objetivos del trabajo de investigación a desarrollar.

1.1 MOTIVACIÓN DE LA TESIS

En la situación actual la Ley de Moore [1-1], que representa la hoja de ruta de la industria del semiconductor y en esencia de toda la industria tecnológica de los últimos 50 años, se haya amenazada ante la escalada tecnológica que nos ha llevado al uso de dispositivos de dimensiones nanométricas. Esto es debido a que para tecnologías de fabricación sub-45nm [1-2] las corrientes de fuga, los errores de proceso, el ruido y las variaciones debidas al proceso de fabricación; así como variaciones de la tensión y temperatura hacen muy difícil obtener los beneficios que uno cabría esperar de la reducción del tamaño del circuito, debido principalmente a los problemas de fiabilidad. Esto se une a una tendencia constante a aumentar la complejidad funcional y reducir el consumo de las aplicaciones, lo cual se traduce en un problema de consumo y fiabilidad en los sistemas en chip (SOC) "*System-On-Chip*". Por lo tanto, los problemas de consumo y fiabilidad no son problemas independientes sino que es hallan intrínsicamente relacionados para la industria del semiconductor, ya que son los principales causantes de la actual inhibición de la Ley de Moore. Por ello no es sorprendente que ya desde el año 2001 la "*International Technology Roadmap for Semiconductors*" (ITRS) [1-3] haya considerado la fiabilidad y la eficiencia energética como dos de los desafíos más importantes a los que se enfrenta la industria del semiconductor, y en consecuencia toda la industria tecnológica mundial. Cabe remarcar que este no es problema nuevo, ya que John Von Neumann [1-4] allá por el año 1956 fue el primero en abordar cómo conseguir una computación fiable a partir de componentes que no lo son..

En la actualidad han surgido toda una nueva generación de aplicaciones encuadradas en alguna de las siguientes tres categorías: reconocimiento de patrones "*Pattern Recognittion*", minería de datos "*Data-Mining*" y síntesis digital (imágenes y/o video). Éstas necesitan procesar cantidades ingentes de información para posteriormente poder ser exportadas y operadas en modelos o aplicaciones basadas en el conocimiento. De hecho, éste es el conjunto de aplicaciones que demanda la sociedad moderna actual, ya sea en el campo de la seguridad, la salud o la energía [1-5].

Este tipo de aplicaciones, especialmente las que se caracterizan por tener que procesar grandes cantidades de información, presentan indicadores de rendimiento estadísticos. Por ejemplo, en la compresión de video hallamos la relación señal ruido (*SNR*), en las redes de comunicaciones existe el "*Bit-Error-Rate*" (*BER*), la probabilidad de detección de un objetivo en un sistema de reconocimiento de patrones, así como muchos otros más. Todo ello nos conduce a que la computación estocástica [1-4, 1-9] (la cual describiremos en el capítulo segundo) desechada a finales de la década de 1970 frente a la incipiente computación clásica (basada en el uso del microprocesador), se presente hoy en día como un enfoque elegante para el diseño de sistemas robustos y energéticamente eficientes (ya que su capacidad de paralelismo les permite operar a menores frecuencias de reloj) [1-6]. La computación estocástica es capaz de explotar de forma inherente su naturaleza estadística para resolver de forma muy simple la mayoría de aplicaciones emergentes (anteriormente descritas). No obstante, paralelamente a la computación estocástica existen actualmente otras vías con una base probabilística que permiten la implementación de forma fiable de los sistemas de procesamiento demandados por la próxima generación de aplicaciones, a la vez que permiten afrontar los problemas tecnológicos actuales. Estos sistemas pueden ser por ejemplo las redes neuronales tanto en la versión clásica como en su versión bioinspirada o pulsante "*Spiking Neurons*". Estos sistemas son conocidos por su capacidad de implementación de sistemas de muy alta fiabilidad [1-7, 1-8].

1.2 OBJETIVOS DE LA TESIS

El campo de acción en el cual se enmarca el presente trabajo de investigación es el del procesamiento masivo de información para aplicaciones orientadas al reconocimiento de

patrones. El objetivo principal del presente trabajo de investigación es la "*Implementación en Hardware de Sistemas de alta fiabilidad basados en Metodologías Estocásticas*", a fin de desarrollar y evaluar los elementos computacionales estocásticos básicos orientados a la resolución de problemas de procesamiento masivo. Para ello, el objetivo principal de la tesis se desarrollado alrededor de los siguientes cuatro objetivos secundarios:

- Desarrollo y evaluación de los fundamentos básicos de la computación basada en metodologías pulsantes/estocásticas.
- Desarrollo de los elementos y metodologías básicas para la aplicación de la computación estocástica al campo del reconocimiento de patrones.
- Desarrollo de nuevas implementaciones de redes neuronales basadas en la computación estocástica, para su aplicación al campo del reconocimiento de patrones.
- Desarrollo de nuevas estructuras digitales para la implementación de redes neuronales pulsantes, así como el desarrollo de nuevas metodologías de configuración de estas redes orientadas al reconocimiento de patrones.

Una vez enumerados los diferentes objetivos procederemos a describir de forma detallada cada uno de ellos:

El primero de ellos establece el desarrollo y evaluación de los elementos básicos de una metodología general que permita la utilización de una lógica basada en pulsos mediante el uso de circuitos digitales. Este primer objetivo es de gran importancia puesto que los elementos desarrollados serán la base sobre la que sustentarán los diferentes desarrollos y aplicaciones presentados.

El segundo objetivo pretende abordar el uso de las metodologías de computación basadas en pulsos (principalmente la computación estocástica) para su aplicación al procesamiento masivo de datos, en el campo del reconocimiento de patrones "*Pattern Recognition*" y en su aproximación estadística. Dichos elementos de procesamiento se diseñarán

completamente orientados a la implementación de estructuras de procesamiento en paralelo.

El tercer objetivo pretende abordar el problema del procesamiento masivo de datos orientado al reconocimiento de patrones mediante el uso de la computación estocástica y de redes neuronales. Para ello se pretende desarrollar una metodología de implementación de redes basadas en elementos de computación estocástica que permitan una fácil implementación de éstas además de una fácil configuración a partir de los posibles procesos de entreno usando herramientas o aplicaciones matemáticas estándar.

El cuarto y último objetivo se enmarca igual que el tercero en el campo de las redes neuronales. En este caso se pretende abordar el problema de la computación masiva orientada al reconocimiento de patrones mediante elementos de procesamiento simple conocidos como neuronas pulsantes "*Spiking Neurons*" basados en una codificación de la información pulsante (extremadamente insensible al ruido), pero esta vez dejando de lado la computación estocástica puramente dicha. Por lo tanto, se pretenden abordar otras formas de procesamiento de la información bioinspiradas. Para ello abordaremos el desarrollo de circuitos digitales que emulen estructuras neuronales de la forma más fidedigna posible, a la vez que se abordará el desarrollo de metodologías de configuración de estas redes orientadas al procesamiento masivo de la información en el campo del reconocimiento de patrones. Dichas metodologías serán implementadas directamente en Hardware.

1.3 ESTRUCTURA DE LA TESIS

En el presente subapartado se describe de forma resumida la estructura y el contenido de cada una de las partes del presente trabajo de investigación. Éste se estructura alrededor de cinco capítulos y dos anexos, hallándose contenido el grueso del trabajo de investigación realizado alrededor de tres capítulos como son: el capítulo segundo, el tercero y cuarto. A fin de guiar al lector a través de los diferentes contenidos abordados en la presente tesis, hemos procedido a realizar un pequeño resumen de cada uno de ellos:

- Capítulo 1: En este capítulo se presenta una introducción general al problema a resolver, a la vez que se establecen los objetivos del trabajo de investigación.

- Capítulo 2. Detalla en primera instancia los orígenes y las virtudes de la computación estocástica. Posteriormente se introducen los mecanismos de conversión y reconversión de las magnitudes binarias en señales estocásticas. También se aborda el problema de la necesidad de la gran cantidad de generadores de números aleatorios que se requiere para esta tarea, presentándose una solución propia para mitigar este problema. A continuación se detallan las diferentes codificaciones estocásticas de la información clásicas (unipolar y bipolar), para las cuales se presenta todo un conjunto de bloques propios desarrollados para servir de base a las diferentes aplicaciones desarrolladas en los capítulos tercero y cuarto. Finalmente se presentan dos nuevas codificaciones estocásticas a las que se ha denominado *codificación estocástica extendida sin signo* y *codificación estocástica extendida con signo*, para solventar las limitaciones de las codificaciones estocásticas clásicas. Ambas codificaciones se basan en la codificación de las magnitudes como la razón de dos señales estocásticas. Para ambas codificaciones se presenta todo un conjunto de bloques estocásticos propios, los cuales serán la base de las aplicaciones presentadas en los capítulos tercero y cuarto.

- Capítulo 3. En este capítulo se presenta una introducción a todo un conjunto de conocimientos relativos al reconocimiento de patrones. Se presenta la implementación estocástica de funciones densidad de probabilidad (f.d.p) puramente normales, a fin de servir como base de la evaluación de un bloque estocástico capaz de evaluar la probabilidad a posteriori de una distribución de probabilidad, así como de una metodología estocástica de clasificación computacionalmente eficiente denominada *Clasificación Estocástico Multidimensional* (MSC). Finalmente se presenta una implementación estocástica de la metodología de las ventanas *Parzen* para la estimación de las distribuciones de probabilidad en el supuesto del reconocimiento de patrones no paramétrico (en el

cual se desconoce la distribución de probabilidad condicional de una determinada distribución de datos). De esta forma se establece un mecanismo completo que permite la clasificación estocástica de cualquier conjunto de datos, se conozca o no a priori su distribución de probabilidad condicional.

- Capítulo 4. Se inicia con una introducción desde el punto de vista histórico y conceptual al campo de las redes neuronales clásicas (primera y segunda generación de redes neuronales), haciendo hincapié en su aplicación al campo del reconocimiento de patrones. A continuación se introducen las redes neuronales pulsantes (redes de tercera generación). Seguidamente se establece el marco teórico para el desarrollo de un sistema de aprendizaje hardware para redes neuronales pulsantes especialmente orientadas al reconocimiento de patrones temporales. A la vez se presenta el diseño y la evaluación de una neurona pulsante mixta analógico digital a fin de emular el comportamiento y la fiabilidad de una neurona biológica, para seguidamente abordar la implementación estocástica de redes neuronales de primera y segunda generación. En esta última parte se detalla la implementación y caracterización de las neuronas más comunes (Perceptrón, neurona lineal y neurona Tangente-Sigmoidal) mediante una nueva codificación estocástica (codificación estocástica extendida con signo) especialmente desarrollada para la implementación de redes neuronales. Finalmente se presenta la implementación de un clasificador mediante una red neuronal estocástica basada en las neuronas anteriormente descritas.

- Capítulo 5. En este capítulo se recogen las principales conclusiones obtenidas a lo largo de los anteriores capítulos, a la vez que se plantean las líneas futuras de trabajo derivadas del presente trabajo de investigación.

- Anexo A. Describe el diseño y caracterización experimental de uno de los circuitos caóticos CMOS más pequeños (en número de transistores requeridos) que podemos hallar en la literatura. Éste se fundamenta sobre una pequeña red neuronal analógica

recurrente. Finalmente se evalúan las propiedades de aleatoriedad de dicho circuito para su aplicación al campo de la generación de números aleatorios.

- Anexo B. Presenta los códigos fuente (ANSI-C) de las diferentes aplicaciones informáticas desarrolladas como soporte para la resolución de los diferentes ejemplos de aplicación presentados a lo largo del trabajo de investigación.

2. LA COMPUTACIÓN ESTOCÁSTICA

El objetivo de este capítulo no es otro que el presentar las bases de la computación estocástica, sobre la que se cimienta una gran parte de las aportaciones que se presentará a lo largo de los siguientes capítulos.

El presente capítulo se ha estructurado en dos partes: una introducción en la cual se abordan los orígenes y la evaluación histórica de la computación estocástica, para luego en la segunda parte pasar a describir la metodología para la obtención de señales estocásticas y su posterior reconversión. Adicionalmente se presentan las diferentes codificaciones de la información estocásticas evaluadas en esta tesis, así como la implementación digital de las diferentes operaciones y funciones para cada una de ellas.

2.1 INTRODUCCIÓN

Desde los orígenes de la computación moderna en la década de 1940 ha existido un interés latente en la comunidad científica en el estudio de arquitecturas de computación radicalmente distintas a las tradicionalmente conocidas como de Von Newman [2-25]. La principal característica de estos sistemas es su alto grado de paralelismo, la existencia de multitud de elementos de computación de baja complejidad, comportamiento no-lineal y una gran interconexión entre ellos. El interés hacia estos sistemas radica que a priori ofrecen unas ventajas inmejorables para implementar de forma rápida aplicaciones de inteligencia computacional como puede ser el reconocimiento de patrones o cualquier sistema que necesite de capacidades de adaptación o aprendizaje. En la presente tesis doctoral utilizaremos conceptos de computación estocástica [2-1] para el desarrollo de sistemas de procesamiento basados en la interacción de múltiples componentes de baja complejidad (como pueden ser las redes neuronales). Las técnicas de computación que se presentan son las más óptimas para la construcción de dichos elementos computacionales, que si bien presentan un comportamiento básico no demasiado complejo, su naturaleza no-

lineal los hace extremadamente complicados de implementar de forma tradicional mediante sistemas digitales.

Paradójicamente los conceptos básicos de la computación probabilística o mejor dicho la computación basada en el uso de variables aleatorias, fueron introducidos por el mismo John Von Neumann en el año 1953 [2-26], aunque esta forma de computación no se desarrolló teóricamente ni fue posible ninguna implementación hasta la década de 1960 gracias a los avances tecnológicos que se produjeron en el campo de la electrónica y la computación. Estos avances condujeron en el año 1965 a dos grupos independientes de investigadores que desarrollaron el concepto de la computación estocástica basada en el uso de la probabilidad de conmutación de señales binarias. Dicha probabilidad se utiliza para codificar el valor de una magnitud analógica, pudiendo éstas ser operadas mediante puertas digitales estándares. El primero de estos grupos ubicado en el "*Standard Telecommunications Laboratories*" en Inglaterra y compuesto por B.R Gaines y J.H Andreae trabajaba en el campo de la optimización de los sistemas de control especialmente orientado al aprendizaje de máquinas. Dicho grupo realizó una publicación interna al respecto titulada "*Stochastic Analog Computer*" [2-27] donde se presentaba la forma práctica de implementar dicha computación. Un segundo grupo ubicado en la universidad de Illinois en Estados Unidos y compuesto por W.J. Poppelbaum y C. Afuso que se hallaban trabajando en sistemas de procesamiento de imágenes, realizaron una publicación relacionada con esta temática titulada "Noise Computer" [2-29]. A ambos grupos de investigación se les atribuye el ser los padres de la computación estocástica.

Por lo tanto, la computación estocástica es el resultado de aplicar las leyes probabilísticas a los bloques lógicos digitales [2-30], [2-28], [2-1], [2-12], pasando éstos a realizar operaciones pseudo-analógicas mediante el uso de tramas de pulsos estocásticos (señales estocásticas) que representan las magnitudes analógicas a operar. Dichas señales binarias (Figura 2-1) representarán diferentes magnitudes en función de la probabilidad de conmutación asignada en un intervalo dado de tiempo (fijo), o lo que es lo mismo la información se codifica como el valor medio de conmutación de una secuencia de pulsos aleatorios e independientes entre

sí. Esto posibilita el poder tratar con magnitudes analógicas mediante sistemas puramente digitales.

La computación estocástica presenta diversas ventajas frente a las técnicas clásicas de procesado digital como pueden ser su tolerancia a errores (derivada de su naturaleza probabilística) además de permitir implementar complejas operaciones aritméticas mediante unas pocas señales binarias (bits) que operan sobre puertas lógicas elementales. Existe también una gran facilidad para la implementación de sistema de procesado en paralelo e implementaciones hardware [2-2]. Si nos remitimos al trabajo realizado por Naes et al. [2-3] predice que la computación estocástica puede incrementar el paralelismo y la redundancia de los sistemas de forma considerable a la vez que se incrementa el rendimiento de la circuitería digital de procesamiento.

La implementación de sistemas de computación en paralelo es uno de los campos donde la computación estocástica puede ofrecer mayores ventajas [2-4], [2-10], en vista que las arquitecturas de procesamiento en paralelo tradicionales tienen el inconveniente de requerir de una gran cantidad de bloques hardware (ya sea en implementaciones sobre ASIC's o en FPGA's) para realizar dichas funciones. Por lo tanto, el número de tareas que pueden ejecutarse en paralelo en dichos sistemas es pequeño, siendo éste el paradigma de la computación clásica o tradicional. Por otra parte, la computación estocástica se ha presentado como una metodología muy útil para la implementación de sistemas complejos de filtrado digital [2-10], [2-33] debido en parte a su capacidad de paralelismo.

En relación a las tareas relacionadas con el reconocimiento de patrones, la lógica estocástica presenta a priori unas capacidades potenciales de computación muy superiores a las que pueden ofrecer las técnicas de computación en paralelo clásicas [2-3], [2-41]. Durante las últimas décadas se ha observado un creciente interés en el desarrollo de metodologías que permitan la implementación de redes neuronales mediante lógica estocástica [2-5], [2-6], [2-7], [2-8], [2-32], [2-36]. El objetivo final de todos estos trabajos es el posibilitar la implementación de redes neuronales extensas con un alto grado de interconexión entre neuronas en un solo chip (ASIC's) o en algún elemento de lógica programable de tamaño medio (FPGA o CPLD).

No obstante, a su vez la lógica estocástica tiene una serie de inconvenientes como es la necesidad de un gran número de generadores de señales aleatorias descorrelacionadas a fin de poder convertir las magnitudes binarias en pulsantes y posibilitar así la correcta operación de los diferentes bloques funcionales. Estos elementos consumen gran cantidad de elementos lógicos o área de integración. Otro inconveniente proviene de la naturaleza aleatoria de la computación estocástica, en la cual toda señal estocástica tiene asociada una determinada varianza, la cual nos limitará la precisión con la que podremos reconvertir dicha señal en su magnitud binaria equivalente. Este es el principal motivo por el cual la computación estocástica históricamente ha fracasado frente a la computación clásica como sistema de procesamiento de datos. Debido a ello se ha visto relegada su aplicación a la solución de cierto tipo de problemas en los cuales su naturaleza probabilística le hace tener grandes ventajas frente a las implementaciones computacionales clásicas.

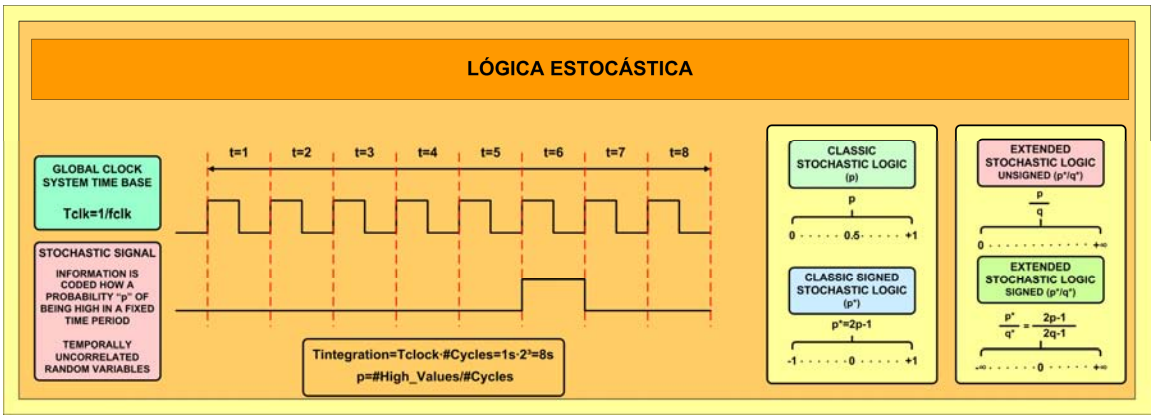

Figura 2-1: Conceptos básicos de la computación estocástica (Figura N)

A lo largo del presente capítulo se presentan las diferentes versiones de la lógica estocástica (Figura 2-1) evaluadas, alguna de ellas ya bien conocidas a través de la literatura [2-1], [2-30], [2-35] como pueden ser la lógica estocástica clásica *sin signo* (definida entre 0 y 1) o la lógica estocástica clásica con signo (definida entre -1 y 1) descritas por R.G Gaines como *representaciones bipolares* [2-34], [2-35]. También se presentan otras variantes desarrolladas por nuestra parte a fin de intentar sortear algunas de las limitaciones halladas en las descripciones clásicas a la hora de realizar las diversas implementaciones de sistemas de computación estocástica que se presentan en lo largo de la presente tesis. Las variantes

desarrolladas son la *lógica estocástica extendida sin signo* y *estocástica extendida con signo*. La primera de ellas representa un primer paso (incompleto) que se llevó a cabo para permitir la representación y procesado de señales estocásticas de magnitudes descritas en el intervalor [0…+∞), y la *lógica estocástica con signo* que viene a ser la evolución de la anterior permitiendo representar y procesar señales definidas en el rango de (-∞….+∞).
En los siguientes subapartados se presentan para cada una de las variantes de la lógica, las implementaciones de las diferentes operaciones aritméticas y toda una serie de funciones matemáticas necesarias para poder realizar con las aplicaciones que se presentan a lo largo de los posteriores capítulos.

2.2 LAS SEÑALES ESTOCÁSTICAS

Para poder hacer uso de cualquiera de las variantes de la lógica estocástica será imprescindible siempre convertir las magnitudes binarias "X" (las magnitudes binarias las denotaremos siempre en mayúsculas) a sus correspondientes señales estocásticas "x" (las señales estocásticas las denotaremos siempre en minúsculas) y viceversa. Los procedimientos y/o fundamentos para la realización de dicha conversión y reconversión, están bien detallados en la literatura [2-1], [2-4], [2-10], [2-11], [2-12]. Dichos procedimientos son válidos para cualquiera de las variantes o codificaciones de la lógica estocástica, y no requieren de ninguna adaptación para su correcto funcionamiento.
Antes de abordar la descripción de los elementos encargados de dichas conversiones se hace necesario describir cómo se debe proceder a obtener las magnitudes binarias "X" (de N-bits) que posteriormente deben ser convertidas en señales estocásticas. Las magnitudes binarias "X" son el resultado de un proceso de normalización previa de los valores reales $\Re$ a representar (normalizados en función del valor mayor), a fin de asegurar que las primeras se hallen correctamente definidas en el espacio de representación admitido por la variante de la lógica estocástica (clásica con signo, extendida sin signo y extendida con signo), siendo la resolución de esta conversión directamente proporcional al número de bits "N" usados para su conversión. Por lo tanto, en este proceso de conversión se debe estimar que el error que será equivalente a $\pm\left(1bit/2^{N}\right)\cdot Espacio\_respresentación\_lógica$. Por lo tanto las magnitudes binarias (de N-bits) a convertir representan en realidad una fracción del espacio

de representación (o una probabilidad), y no realmente el valor binario asignado. A fin de facilitar la compresión del proceso anteriormente descrito, se presentan dos ejemplos:

El siguiente ejemplo presenta la conversión del valor real “0,5” a su magnitud binaria equivalente, con una resolución de 16-bits. Como puede apreciarse en la Ecuación [2-1] tendrá como magnitud binaria equivalente X=32767.

$$\left.\begin{array}{l} X^* \in |0,1| \in \Re^+ \\ Espacio\_binario \rightarrow (0,..,2^{16}-1) \in \aleph \end{array}\right\} \rightarrow \begin{cases} X^* = 0,5 \rightarrow X = 0,5 \cdot \left(2^{16}-1\right) = \\ = 0,5 \cdot 65535 \approx 32767 \\ X^* = 0 \rightarrow X = 0 \\ X^* = 1 \rightarrow X = 65535 \end{cases}$$

Ecuación [2-1]

El segundo presenta la conversión de dos valores reales “1,52 y “0,5” a sus magnitudes estocásticas equivalentes “A” y “B”, para su posterior conversión y operación. Como puede apreciarse en la Ecuación [2-2] tendrá como factor de normalización “K=0,67” y magnitudes binarias equivalentes “A=65535”.y “B=21845”.

$$\left.\begin{array}{l} X^* \in |0,1| \in \Re^+ \\ Espacio\_binario \rightarrow (0,..,2^{16}-1) \in \aleph \end{array}\right\} \rightarrow \begin{cases} A^* = 1,5 \rightarrow A = \dfrac{1}{1,5} \cdot 1,5 \cdot \left(2^{16}-1\right) = \\ = 1 \cdot 65535 = 65535 \\ B^* = 0,5 \rightarrow B = \dfrac{1}{1,5} \cdot 0,5 \cdot \left(2^{16}-1\right) = \\ = 21845 \\ K_{normalización} = \dfrac{1}{1,5} \end{cases}$$

Ecuación [2-2]

A continuación se describe la implemantación digital para los dos bloques requeridos para realizar dichas operaciones, uno para la conversión y otro para reconversión de las señales estocásticas, los cuales se describen en detalle a continuación.

El primero de ellos es el bloque convertidor "*Binary-To-Pulse"* (B2P(N)) para palabras de N-bits, que se encargará de traducir las magnitudes binarias "X" de N-bits (el número de bits se fija para un determinado proceso) en su correspondiente señal estocástica "x".

Por otro lado, las señales estocásticas "x" pueden volver a ser convertidas en magnitudes binarias "X" mediante un módulo convertidor "*Pulse-To-Binary*" de módulo N (P2B(N)); que se encargará de integrar los pulsos a nivel alto recibidos a lo largo de N períodos de reloj, que se corresponden con el período de evaluación "T_{EVAL}".

2.2.1 CONVERSIÓN DE UNA SEÑAL PULSANTE A UNA MAGNITUD BINARIA

Para proceder con la conversión de una señal pulsante a una magnitud binaria se requerirá de un bloque convertidor al cual se le ha denominado Pulse-To-Binary de orden N (P2B(N)) (Figura.2-2); que consiste en un circuito digital que se encarga de integrar el número de pulsos a nivel alto provistos por una señal estocástica "x" durante N-ciclos de reloj (que se corresponde con un tiempo de evaluación $T_{EVAL} = N \cdot T$, donde "T" es el período de señal de reloj global del sistema). La salida de este bloque es una magnitud binaria de n-bits que se mantiene fija a la salida, hasta que se refresca al cabo de N ciclos de reloj (que se corresponde con el período de evaluación del sistema "T_{EVAL}", anteriormente descrito) con el nuevo valor de la integración. Dicho bloque se compone internamente de un contador módulo N (el valor N está directamente relacionado con el número de bits usado para la conversión de la magnitud binaria en pulsante, mediante el bloque B2P(N)) y de un elemento de memoria de N-bits a la salida que se encarga de almacenar en dicho registro el valor del conteo (hace las funciones de un circuito de *Sample & Hold*, en la electrónica analógica).

Cabe remarcar que existirá siempre un error presente en dicha conversión, el cual puede ser evaluado de forma probabilística. Si se define $P_N(x)$, como la probabilidad que la salida del bloque P2B(N) sea igual a una determinada magnitud binaria "X", partiendo del hecho

que dicha probabilidad viene regida por una distribución binomial, tendremos que (Ecuación [2-3]):

$$P_N(X) = \binom{N}{X} \cdot p^X \cdot (1-p)^{N-X} \qquad \text{Ecuación [2-3]}$$

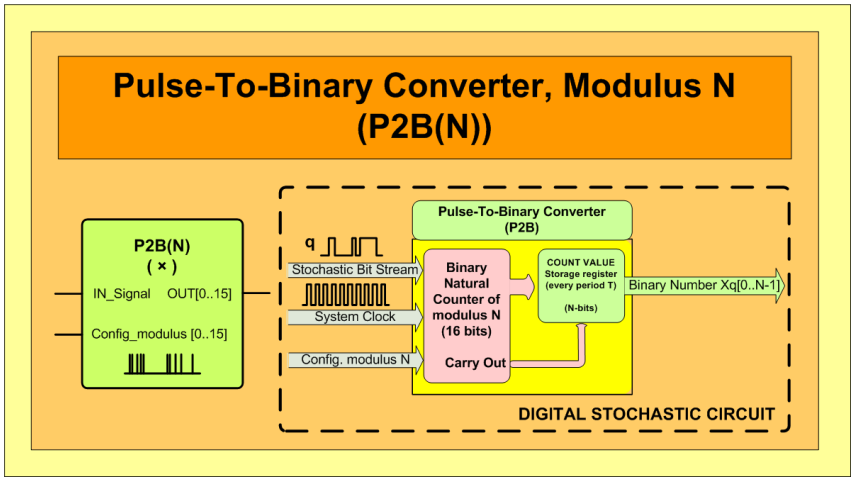

Figura 2-2: Bloque Pulse-To-Binary converter (P2B(N))

Donde el parámetro “p” representa la probabilidad de conmutación de la señal de entrada, es decir, la probabilidad asociada a la señal estocástica “x”.

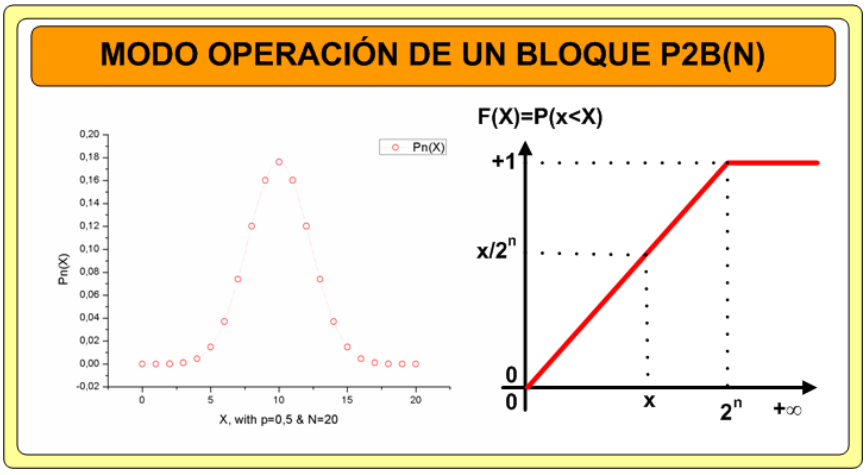

Figura 2-3: Modo de operación del bloque P2B(N)

Por lo tanto el valor medio obtenido a la salida (Figura 2-3) de un bloque P2B(N), equivale al valor esperado de la magnitud binaria X asociada a la señal estocástica con probabilidad

de activación “p”, de tal forma que $\langle X \rangle = N \cdot P$. Por otra parte, la desviación estándar de dicha distribución vendrá dada por la expresión: $\sigma^2 = N \cdot p \cdot (1 - p)$. El valor máximo de la desviación estándar “$\sigma_{\max}$”estará relacionado con el error relativo de cada una de las N conversiones, de tal forma que se evalúa como $Error = 2 \cdot \sigma_{\max} / N$. El valor máximo de dicha dispersión se obtiene para un valor de “p=1/2”, por lo tanto el error relativo para cada una de las conversiones será del orden de $Error \approx 1 / N^{\frac{1}{2}}$.

De la expresión obtenida para el error relativo se puede obtener una relación entre el error relativo de la conversión y el tiempo de conversión/evaluación (T_{EVAL}) asociado al sistema, que vendrá dado por la siguiente expresión (Ecuación [2-4]):

$$T_{EVAL} = N.T = \left(\frac{1}{Error} \right)^2 \cdot T \qquad \text{Ecuación [2-4]}$$

Por lo tanto, de la Ecuación [2-4] se puede concluir que existe una relación de compromiso entre el tiempo de concesión/evaluación y el error relativo cometido en cada una de las conversiones. La consecución de errores relativos pequeños requerirá de períodos de integración muy largos y períodos de integración/evaluación muy cortos equivaldrán a errores relativos muy grandes. Como ejemplo mencionar que para obtener una precisión de 8-bits se requerirá de conversores con una de resolución de 16-bits.

Las diferentes aplicaciones que hacen uso de las diversas codificaciones de la lógica estocástica que se describen a lo largo de los siguientes capítulos, se han medido e implementado usando bloques “*P2B(N)*” de 16-bits (precisión de 8-bits), y una señal de reloj de 50MHz, con la cual se han obtenido unos resultados razonables en cuestión de precisión y tiempo de evaluación/integración del sistema.

2.2.2 CONVERSIÓN DE UNA MAGNITUD BINARIA A UNA SEÑAL PULSANTE

La conversión de una magnitud binaria "X" de N-bits a su señal estocástica "x" equivalente (1-bit) se realizará mediante un bloque convertidor, al cual se le ha denominado *Binary-To-Pulse de módulo N* "*B2P(N)*", cuya implementación digital se presenta en la Figura 2-4.

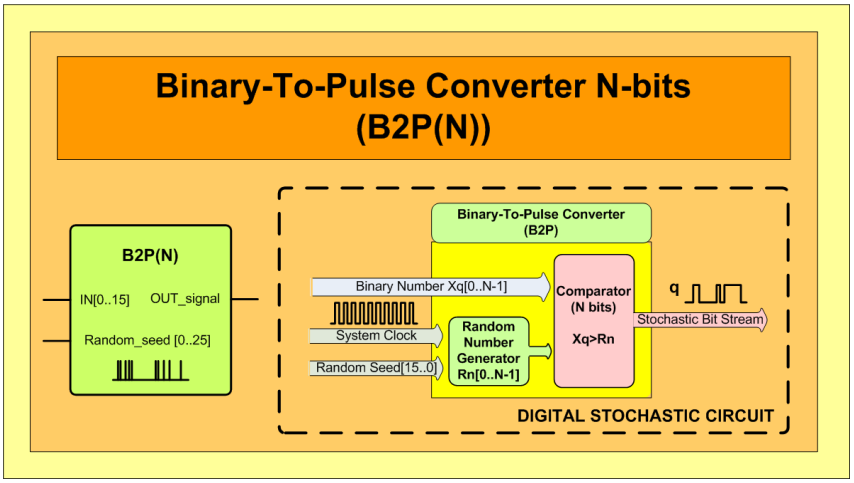

Figura 2-4: Bloque Binary-To-Pulse converter (B2P(N))

La forma genérica de implementar esta conversión ha sido mediante un comparador de N-bits y un circuito generador de número aleatorios de $N_{Aleatorio}$-bits (donde las palabras aleatorias usadas para la conversión deben cumplir con la condición $N_{bits\_aleatorioz} \geq N_{bits\_magnitud\_binaria}$), tal y como se detalla en la literatura [2-1, 2-11]. El principio de funcionamiento del circuito *B2P(N)* se basa en disponer de un generador de números aleatorios o pseudo-aleatorio de N-bits (con un período de repetición mayor al período de evaluación "T_{EVAL}"), además de un comparador digital de N-bits encargado de comparar si el valor de la magnitud binaria "X" es mayor al número obtenido por el generador de números aleatorios. En tal caso durante dicho ciclo de reloj la salida del comparador se mantendrá a nivel alto "1", mientras que si es menor se mantendrá a nivel bajo "0"

Por lo tanto, en cada período evaluación T_{EVAL} del sistema, se generarán N valores aleatorios (tantos como pulsos de reloj). Al ser aleatorios, podrán ser considerados como equiprobables, por lo tanto en un determinado intervalo de tiempo los valores generados habrán cubierto todo el espacio de representación [0..$\left(2^n - 1\right)$] de forma uniforme.

Por ejemplo, en el caso que se pretenda obtener la conversión del valor real de "0,5" mediante un bloque *B2P(N)* de 16-bits, se obtendrá una magnitud binaria equivalente igual a "X=32767", que dicho bloque se encargará en convertir en una señal pulsante que permanecerá el 50% del período de evaluación a nivel alto '1'. No obstante las salidas de este bloque no seguirá ningún patrón concreto, ya que la salida se obtiene a partir de un generador de números aleatorios o pseudo aleatorios que no sigue ningún patrón concreto de repetición en dicho período de evaluación.

Desde el punto de vista estadístico, las señales estocásticas obtenidas de la comparación de las magnitudes binarias "X", con una secuencia aleatoria de números de "Rn", al suponerse esta última uniformemente distribuida en el tiempo vendrán representadas por las siguientes funciones: la función densidad de probabilidad (Ecuación [2-5]) que presenta un valor constante $1/2^n$ en todo el rango de representación y un valor nulo en el resto, por otra parte tenemos la función distribución de probabilidad (Ecuación [2-6]) que se encarga de determinar la probabilidad de conmutación de una determinada magnitud binaria y finalmente el valor medio (Ecuación [2-7]) de la secuencia estocástica generada que será proporcional a la magnitud binaria codificada.

$$f_X(x) = \begin{cases} \dfrac{1}{2^n} \rightarrow 0 \le X < 2^n \\ 0 \rightarrow X \ge 2^n \end{cases} \qquad \text{Ecuación [2-5]}$$

$$F_X(x) = P(x < b) = \begin{cases} \dfrac{b}{2^n} \rightarrow 0 \le b < 2^n \\ 1 \rightarrow b \ge 2^n \end{cases} \qquad \text{Ecuación [2-6]}$$

$$E\left[S_X(x)\right] = 1 \cdot P(x < X) + 0 \cdot P(x \geq X) = \frac{X}{2^n} \qquad \text{Ecuación [2-7]}$$

2.2.2.1 Generación On-Chip de números pseudo-aleatorios

Tal y como, se ha comentado a lo largo del apartado anterior, uno de los bloques más importantes usados en computación estocástica son los convertidores de código B2P(N). Esto es debido a que las aplicaciones en las que suele ser interesante el uso de la computación estocástica, suelen haber muchas magnitudes/datos de entrada y muy pocos datos de salida (por Ej. en el reconocimiento de patrones, redes neuronales…). El uso de múltiples bloques P2B implica la necesidad de disponer de la misma cantidad de generadores de números aleatorios o pseudo-aleatorios independientes entre ellos, a fin de poder generar señales estocásticas temporalmente descorreladas entre sí.

En la literatura se hallan multitud de estrategias para la generación de secuencias aleatorias y pseudo aleatorias de bits, entre ellas algunas de las de mayor relevancia se presentan a continuación:

- La técnica de amplificación del ruido térmico en amplificadores de alta ganancia, es una de las más extendidas en el diseño de generadores de números aleatorios. Se basa en el uso de un amplificador de alta ganancia con una respuesta plana (constante) en frecuencia en el rango de operación. A partir de él se obtendrá un generador de ruido blanco, el cual si se filtra a través de un limitador duro de amplitud (como en la desmodulación FM), se obtiene una señal pulsante con unas muy buenas propiedades de aleatoriedad. Estos circuitos se caracterizan por ocupar gran cantidad de área en los circuitos integrados al estar basados sobre un amplificador analógico, además hay que tener presente los problemas que pueden surgir de autoacoplamiento entre ellos si se integra más de uno de estos en el mismo integrado, debido a la alta ganancia de dichos amplificadores. Integraciones de este tipo de generadores de números aleatorios son fáciles de hallar en la literatura como por ejemplo en [2-13].
- Otra técnica para la generación de secuencias aleatorias es la basada en el uso de circuitos caóticos integrados en ASIC’s. Integraciones de este tipo de circuitos generadores de números aleatorios son descritas en la literatura en [2-14, 2-15].

- Para la generación de secuencias aleatorias se puede hacer uso de circuitos biestables especialmente diseñados para mantener comportamientos metaestables, asegurando por diseño la misma probabilidad de generación de señales a nivel alto y bajo. Al estar constituidos estos circuitos por muy pocos componentes, son muy eficientes en términos de área de integración. Pero en contrapartida son circuitos muy sensibles a los parámetros eléctricos (cargas a las salidas de los biestables), así como requerir de un forzado inicial para llevar al sistema al estado metaestable. En la literatura hallamos multitud de diseños basados en este principio, como pueden ser [2-16, 2-17].
- Finalmente hallamos los procedimientos de generación de secuencias de números pseudo-aleatorios basados en algoritmos computacionales, como pueden ser los sistemas basados sobre registros de desplazamiento con realimentación lineal, más conocidos en la literatura por sus siglas en ingles "*Linear Feedback Shift Register*" (*LFSR*). Estos generadores son muy usados en los sistemas digitales (primordialmente en los basados en lógica programable *FPGA* o *CPLD*) debido a su compatibilidad directa, y su fácil implementación mediante bloques digitales genéricos (biestables y puertas lógicas). En la literatura se hallan gran cantidad de trabajos que hacen uso de esta metodología para la generación de números aleatorios, como por ejemplo en [2-18].

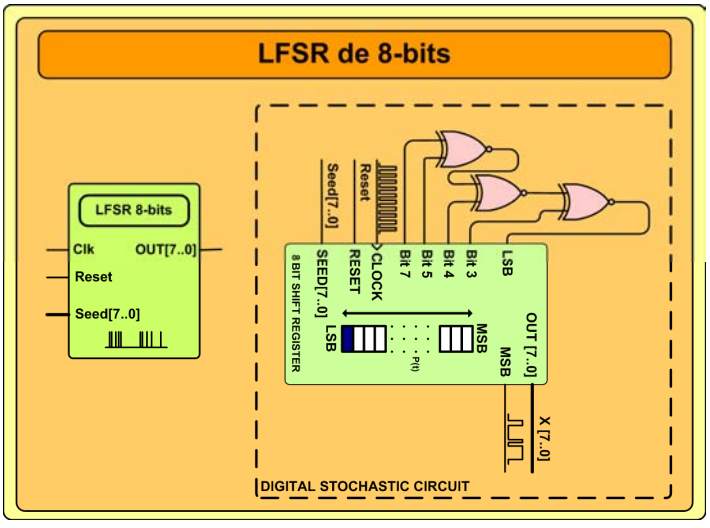

Figura 2-5: Estructura genérica de un bloque LFSR de 8-bits

Para la generación *on-chip* de números aleatorios se ha optado por la última técnica basada sobre registros de desplazamiento realimentados linealmente o también llamados *Linear Feedback Shift Register (LFSR) de N-bits,* debido a su facilidad de implementación en los sistemas implementados sobre lógica programable.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.std_logic_arith.all;
--USE ieee.std_logic_unsigned.all;

ENTITY lfsr_26_conf IS
PORT (clk,reset,enable: IN BIT; seed: IN STD_LOGIC_VECTOR (0 TO 25);
        number:OUT STD_LOGIC_VECTOR(0 TO 25)
);
END lfsr_26_conf;

ARCHITECTURE arch OF lfsr_26_conf IS
SIGNAL tmp: STD_LOGIC_VECTOR(0 TO 25);
BEGIN
PROCESS (clk,reset)
        BEGIN
        IF reset ='1' THEN
                tmp<=seed;
        ELSIF (enable='1') THEN
        IF (clk'EVENT AND clk='1') THEN
                for i in 0 to 24 loop
        tmp(i+1) <= tmp(i);
      end loop;
      tmp(0) <= tmp(25) XNOR tmp(5) XNOR tmp(1) XNOR tmp(0);
        END IF;
        END IF;
END PROCESS;
number <=tmp;
END arch;
```

Figura 2-6: Implementación de un bloque LFSR de 26-bits (usado en los bloques B2P(N)) en VHDL

En realidad, un bloque LFSR [2-9] (Figura 2-5) no es más que una estructura digital síncrona que incorpora en su interior una cadena de biestables interconectados en el cual la entrada del primer biestable de la cadena se corresponde con la señal de realimentación obtenida a partir de una función de transformación lineal de los estados anteriores (implementadas mediante puertas XOR o XNOR). El valor inicial de esta estructura (el valor inicial que toman los diferentes biestables de la cadena) se denomina semilla, y al ser la forma de operar de dicho circuito determinista la secuencia de valores producidos está completamente determinada por el estado actual o el estado anterior. Por lo tanto, dicha estructura reproduce una secuencia de $\left(2^m - 1\right)$ estados (siendo "m" el número de registros

internos de la *LFSR*) que pueden ser considerados como aleatorios entre ellos. Con un período de repetición de estos bloques igual a $\left(2^m - 1\right)$ ciclos de reloj.

No obstante, mediante la modificación del valor de la semilla inicial de una *LFSR* podremos obtener distintas señales aleatorias independientes. Esta afirmación es válida siempre que el período de evaluación del sistema (T_{EVAL}), sea igual o mayor al período máximo de repetición de la LFSR.

La secuencia de realimentación de una LFSR se puede representar como un polinomio de base dos (lo cual indica que los valores que podrán tomar los coeficientes del polinómio serán '0' o '1'), en el cual aparecen las posiciones de los biestables que intervienen en la generación de la señal de realimentación. Este polinómio recibe el nombre de polinómio de realimentación o función generadora. Para que el período de repetición sea máximo, el polinómio de realimentación deberá ser primitivo forzosamente. Cabe remarcar que solamente en el caso que halla un número par de bits que intervengan en la realimentación se obtendrá un polinómio primitivo, y éstos han de ser primos entre sí. Además puede haber más de una secuencia de bits de realimentación que proporcionen un período de repetición máximo.

A continuación, y a modo de ejemplo consideraremos el caso de la LFSR de 26-bits, usada como generador de números aleatorio en los bloques B2P(N) genérico (de los 26 bits de la LFSR sólo se han interconectado los 16-bits más significativos con el comparador). Después de muchas pruebas/medidas experimentales se ha concluido que 26-bits es el tamaño óptimo de la LFSR para los sistemas de computación estocástica implementados hasta el momento. Esta LFSR tiene un período de repetición de 67.108.863 ciclos de reloj.

La función generadora del polinómio primitivo de la *LFSR* de 26 Bits, viene descrito por la Ecuación [2-8]:

$$f(x) = x^{25} + x^5 + x^1 + 1 \qquad \text{Ecuación [2-8]}$$

La implementación del bloque LFSR en los bloques P2B(N) se ha realizado en lenguaje de descripción de hardware (*VHDL*) como se muestra en la Figura 2-6 a fin de facilitar su sintetización en dispositivos de lógica programable (*FPGA's o CPLD's*).

Tal y como se ha descrito hasta el momento, con este circuito se puede implementar de forma simple y rápida un generador de números aleatorios. No obstante este tipo de circuitos tiene el inconveniente de ocupar gran cantidad de celdas lógicas o área de semiconductor: sobre todo en el caso de ser implementadas sobre sistemas de lógica programables (FPGA o CPLD), ya que requieren de multitud de bloques lógicos (Biestables) para su operación (para una *LFSR* de 26 bits requerirá de al menos 26 elementos lógicos de una FPGA). Por lo tanto, se puede afirmar que los bloques generadores de números aleatorios integrados en los bloque convertidores *B2P(N)* y en consecuencia estos últimos, son los elementos que más recursos lógicos consumen en un sistema de computación estocástico.

2.2.2.2 Optimización en la generación de números pseudo-aleatorios para su aplicación a grandes sistemas de computación estocástica

Además de la generación de números pseudo-aleatorios existen otras metodologías para la generación de señales aleatorias como puede ser el uso de sistemas caóticos. El concepto caos está relacionado con la imposibilidad de realizar predicciones precisas a largo plazo del comportamiento de determinados sistemas no lineales, siendo esta propiedad lo que los hace adecuados para la obtención de números aleatorios, necesarios para la generación de señales estocásticas reales. Tal y como se describe en el **Anexo A**, en los últimos años se han desarrollado diferentes diseños de circuitos caóticos capaces de generar señales aleatorias con muy pocos transistores, por tanto actualmente ya se dispone de generadores de números aleatorios que son fácilmente integrables y que ocupan muy poca área. En todo caso, para el uso de estos circuitos conjuntamente con dispositivos de lógica programable (FPGA o CPLD) será necesario interconectarlos externamente (off-chip).

Todo lo expuesto hasta el momento pone de manifiesto dos grandes dificultades. Por una parte, el uso de multitud de bloques *B2P(N)* cada cual con su respectiva semilla, lo que implica un gran gasto en términos de recursos lógicos de la FPGA. Por la otra parte, el uso de circuitos caóticos está limitado puesto que al tener que interconectarse externamente a

las FPGA se dará un desequilibrio en términos de frecuencia de operación entre ambos sistemas (del orden de kHz en el caso de los circuitos caóticos y del orden de decenas o centenares de MHz en el caso de la FPGA). Como respuesta a dicha situación se ha desarrollado una metodología que obtiene una solución de compromiso, que consiste en realizar mutaciones de la semilla de las LFSR mediante el uso de solamente un circuito caótico externo. A grandes rasgos la metodología consiste en ir mutando periódicamente la semilla de una *LFSR* relativamente pequeña (en este caso 8-bits, frente a los 26-bits de las usadas hasta el momento), mediante el uso de un único circuito caótico generador de pulsos aleatorios ubicado fuera del dispositivo de lógica programable (FPGA). De esta forma se compensan parte de las desventajas descritas anteriormente.

Por lo tanto, esta metodología no requiere que el circuito caótico (generador de pulsos aleatorios) esté integrado en el elemento de lógica programable (FPGA) y permite a su vez poder operar con circuitos caóticos externos, que operen a frecuencias mucho menores que la frecuencia interna de operación del sistema de computación estocástico "$f_{Clock} = 1/T_{Clock}$". No obstante, para que el sistema opere correctamente se requiere que el período de mutación de la LFSR sea mucho menor que el período de repetición (Ecuación [2-9]), a fin de asegurar que las señales estocásticas generadas estén descorrelacionadas entre ellas.

$T_{Mutación} << \left(2^n - 1\right) T_{LFSR\_CLK}$, siendo "n" el número de bits Ecuación [2-9]

Para implementar esta metodología se ha fabricado un (*ASIC*) en cuyo interior se ubica un circuito caótico [2-14] (que se presenta en el Anexo A) desarrollado como un oscilador caótico (síncrono, gobernado por una señal de reloj externa) basado en el uso de una red neuronal analógica. El circuito es capaz de operar a una frecuencia de 330Khz y su salida es la encargada de mutar periódicamente de valor del registro interno de la LFSR, eliminado así el problema derivado de tener un período de repetición inferior al período de evaluación del sistema.

En el caso evaluado se parte de una frecuencia de 50MHz, siendo la *LFSR* mínima a usar de 8-bits, ya que su período de repetición será de 195kHz justo al límite de la frecuencia de operación del generador de pulsos aleatorios desarrollado.

Los elementos que componen la estructura de la metodología desarrollada se presentan en la Figura 2-7. El sistema se compone de un oscilador externo de alta calidad que se encarga de proporcionar la señal de reloj al elemento de lógica programable (FPGA), en el que se implementa todo el sistema de computación estocástico.

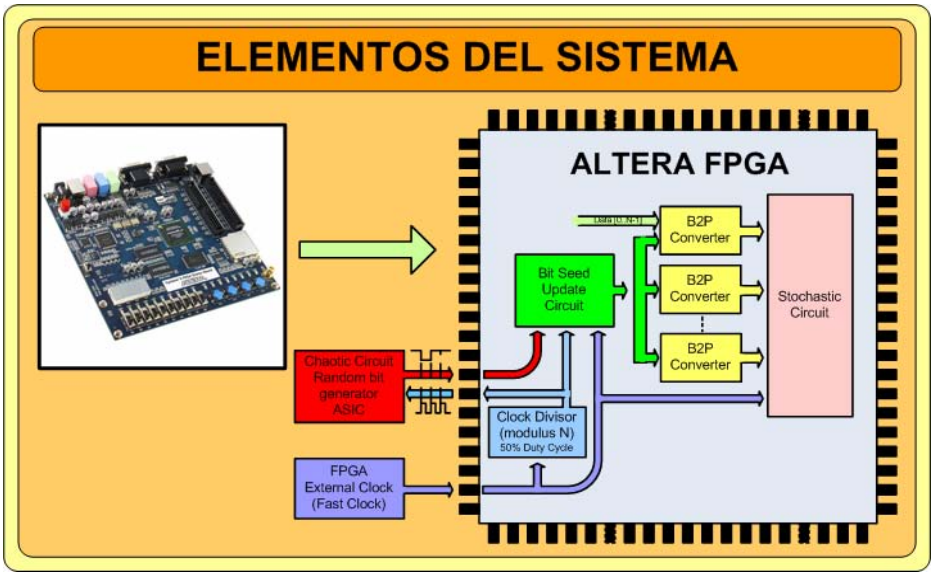

Figura 2-7: Elementos hardware involucrados en la metodología propuesta

El circuito caótico se ubica fuera de la FPGA, hallándose interconectado a ésta mediante dos señales digitales; una señal de reloj encargada de permitir la operación/evaluación del circuito caótico y otra que se corresponde con el nivel lógico resultado de la evaluación del circuito caótico. Por lo tanto, esta metodología tan sólo consumirá dos E/S del elemento de lógica programable.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--USE ieee.std_logic_arith.all;
--USE ieee.std_logic_unsigned.all;

ENTITY lfsr_8_seed IS
PORT (clk,reset,enable: IN BIT; seed: IN STD_LOGIC_VECTOR(0 TO 7);update_bit: IN BIT;random_bit:
IN BIT;
        number:OUT STD_LOGIC_VECTOR(0 TO 7)
);
END lfsr_8_seed;

ARCHITECTURE arch OF lfsr_8_seed IS
SIGNAL tmp: STD_LOGIC_VECTOR(0 TO 7);
BEGIN
PROCESS (clk,reset)
        BEGIN
        IF reset ='1' THEN
                tmp<=seed;
        ELSIF (enable='1') THEN
        IF (clk'EVENT AND clk='1') THEN

                IF (update_bit = '1') THEN
                        IF (random_bit='1')THEN
                                for i in 0 to 7 loop
                                        tmp (i) <= tmp (i) XOR seed(i) XOR '1';
                                end loop;
                        ELSE
                                for i in 0 to 7 loop
                                        tmp (i) <= tmp (i) XOR seed(i) XOR '0';
                                end loop;
                   END IF;
                ELSE
                        for i in 0 to 6 loop
                                tmp(i+1) <= tmp(i);
                        end loop;
                        tmp(0) <= tmp(7) XNOR tmp(5) XNOR tmp(4) XNOR tmp(3);
                END IF;

        END IF;
        END IF;
END PROCESS;
number <=tmp;
END arch;
```

Figura 2-8: Codificación VHDL del bloque LFSR modificado

La operación del sistema será la siguiente: la señal de reloj generada por el oscilador externo (50MHz) se conecta con una de las entradas de distribución de reloj de la FPGA. La frecuencia de señal de reloj puede ser fácilmente modificada internamente mediante un bloque PLL digital "*Phase-Locked-Loop*", para aumentar o reducir dicha frecuencia, según lo requiera la aplicación. Acto seguido la señal de reloj pasa por un circuito divisor de frecuencia a fin de proporcionar una señal de reloj de 195kHz (que se corresponde con la frecuencia de repetición de la *LFSR* de 8bits) al circuito caótico externo, a la vez que al bloque interno encargado de gestionar el proceso de actualización de las semillas de las *LFSR* del sistema.

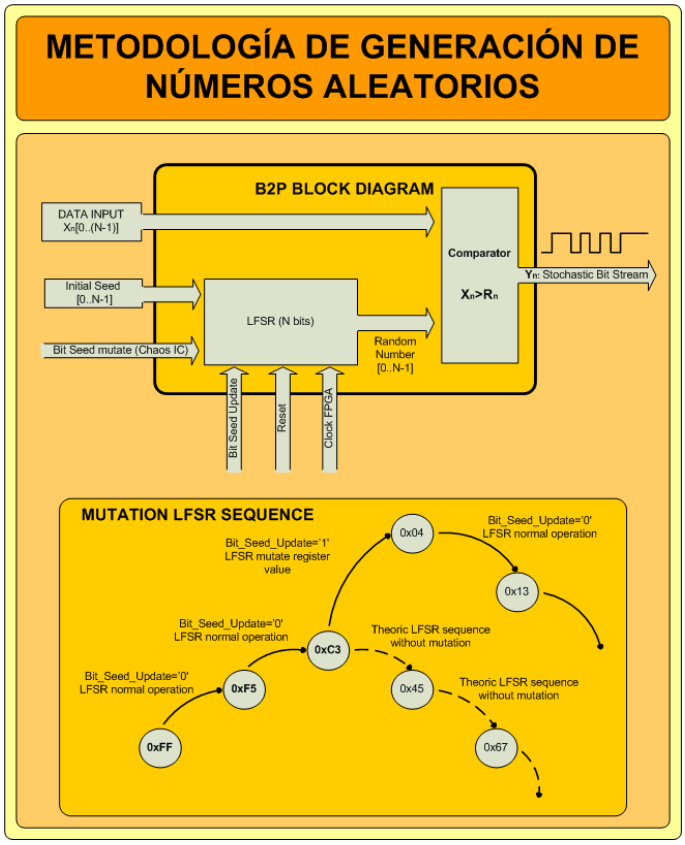

Figura 2-9: Bloque B2P(N) modificado y descripción de la secuencia de generación de números aleatorios (incorporando la mutación)

La señal digital evaluada por el circuito caótico es la encargada de la mutación de las semillas de las *LFSR*. El bloque de actualización de las semillas, a partir de la señal de reloj y el bit aleatorio obtenido, genera una señal global encargada de indicar a todos los bloques *LFSR* que deben actualizar sus semillas "*Bit_Update_Seed*". Una segunda señal de "Bit_Seed_Mutate" (filtrada del circuito caótico) será distribuida a todos los módulos *LFSR* integrados en todos los bloques *B2P(N)* del sistema.

Los bloques LFSR usados para esta metodología han requerido de una cierta modificación de su código (VHDL) (Figura 2-8) con respecto a los boques LFSR descritos hasta el momento, ya que en el instante en que se activa (a nivel alto) la señal "*Bit_Seed_Update*" procederá a mutar los 7 bits más significativos contenidos en los biestables de los LFSR. El

procedimiento de operación del sistema se presenta en la siguiente expresión (Ecuación [2-10]).

$$\begin{cases} \dfrac{(\textit{Modo de operación normal})}{\textit{Bit\_Seed\_Update} = '0' \,\&\, \mathrm{Clk} = '1'} \Rightarrow \begin{cases} q[i]' = q[i\text{-}1],\ i = 1..7 \\ \textit{Feedback} \rightarrow q[0]' = \overline{q[7] \oplus q[5] \oplus q[4] \oplus q[3]} \end{cases} \\ \dfrac{(\textit{Modo de operación mutación})}{\textit{Bit\_Seed\_Update} = '1' \,\&\, \mathrm{Clk} = '1'} \Rightarrow \begin{cases} q[i]' = q[i] \oplus seed[i] \oplus \textit{Bit\_Seed\_Mutate}, \mathrm{i} = 1..7 \\ \textit{Feedback} \rightarrow q[0]' = \overline{q[7] \oplus q[5] \oplus q[4] \oplus q[3]} \end{cases} \\ \dfrac{(\textit{Modo de operación reset})}{\mathrm{Re}\,set = '1' \,\&\, \mathrm{Clk} = '1'} \Rightarrow \{q[i]' = seed[i], \mathrm{i} = 0..7 \end{cases}$$

Ecuación [2-10]

El diagrama de los B2P(N) que integran los LFSR modificados, así como un esquema descriptivo del proceso de generación de números aleatorios incluyendo el proceso de mutación se presentan en la Figura 2-9. Cabe remarca que cada LFSR del sistema será configurada con una semilla diferente a la de los demás módulos LFSR.

2.2.2.2.1 Resultados experimentales

Para comprobar la viabilidad y eficiencia de la metodología desarrollada se ha implementado experimentalmente el sistema sobre una tarjeta comercial "*Cyclone II Starter Board*" que incorpora una (FPGA) *Cyclone II EP2C20F484C7* del fabricante norteamericano ALTERA Corp. a la cual se le ha conectado el circuito caótico generador de señales aleatorias (Anexo A). Una fotografía del montaje experimental se presenta en la Figura 2-10.

El sistema de computación estocástica usado para evaluar el rendimiento del sistema ha consistido en un bloque divisor estocástico, en concreto se ha realizado la operación "*y=0,027/x*". Se ha elegido esta función debido a la complejidad interna del bloque estocástico en cuestión, siendo el sistema de computación estocástico implementado el formado por dos bloques B2P(N), un bloque divisor estocástico "*y=x/y=0,0027/y*" y un bloque *P2B(N)* de salida. La implementación realizada y los resultados experimentales obtenidos de la metodología propuesta dotada de las *LFSR* modificadas se presentan en la Figura 2-11.

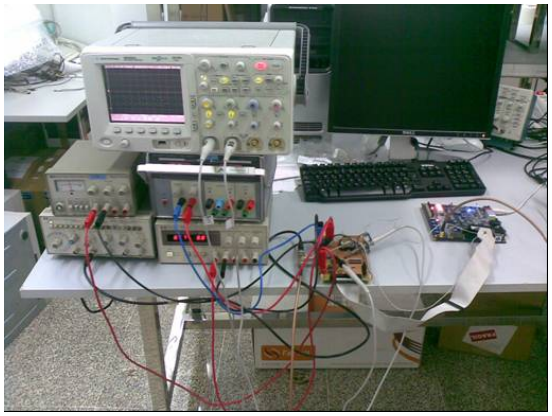

Figura 2-10: Montaje experimental del sistema

En la parte derecha de la Figura 2-11 se presentan los resultados obtenidos mediante la computación estocástica (símbolos) y los predichos por la teoría (línea continua). Como puede observarse, existe una gran correlación entre la teoría y las medidas experimentales (que se corresponde a un coeficiente de correlación de *producto-momento de Pearson* igual a "*r=0,9995"*).

Los resultados obtenidos son semejantes o incluso mejores a los obtenidos mediante el uso de las LFSR de 26-bits (metodología On-Chip). Con esto se demuestra que la metodología es apta para la generación de números aleatorios orientados a la computación estocástica.

Tabla 2-1: Evaluación de la metodología propuesta de generación de números aleatorios

# de circuitos de test	Recursos lógicos de la FPGA	Mediante el uso de B2P(N) con LFSR de 26bits	Mediante el uso de la metodología propuesta	Reducción de los recursos lógicos de la FPGA [%]
5	Logic Elements (LE)	1.141	1.098	**3,77**
	Combinational Functions	745	721	3,22
	Dedicated Logic Registers	860	830	3,49
15	Logic Elements (LE)	1.601	1.477	**7,75**
	Combinational Functions	1.205	1.101	8,63
	Dedicated Logic Registers	1.020	990	2,94
40	Logic Elements (LE)	2.752	2.424	**11,92**
	Combinational Functions	2.355	2.051	12,91
	Dedicated Logic Registers	1.420	1.390	2,11
120	Logic Elements (LE)	6.429	5.467	**14,96**
	Combinational Functions	6.035	5.091	15,64
	Dedicated Logic Registers	2.700	2.671	1,07
200	Logic Elements (LE)	10.109	8.502	**15,90**
	Combinational Functions	9.715	8.131	16,30
	Dedicated Logic Registers	3.980	3.950	0,75

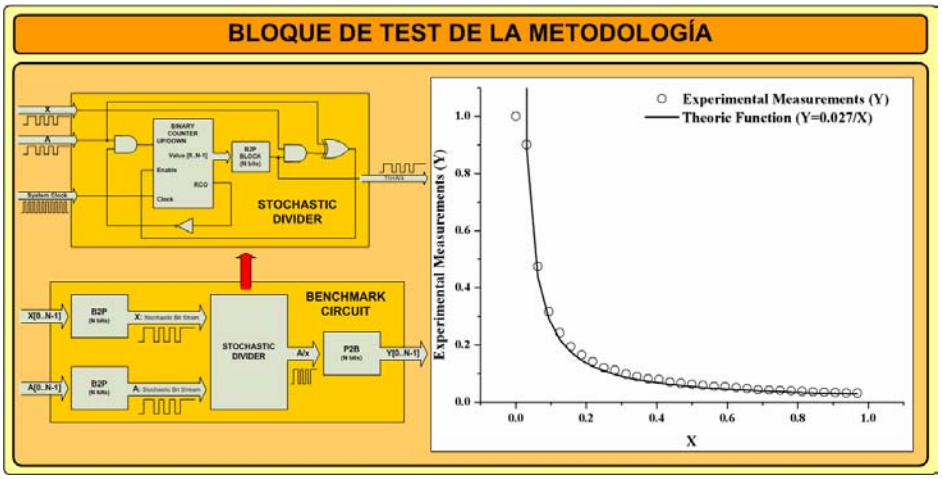

Figura 2-11: Circuito de test y resultados experimentales obtenidos mediante la metodología propuesta

Una vez comprobada la operatividad de la metodología se ha procedido a evaluar el rendimiento en términos de elementos lógicos (FPGA) consumidos, ya que la reducción del consumo de recursos lógicos en las FPGA es el objetivo fundamental de dicha metodología. Para ello se ha procedido a replicar en diferentes cantidades el sistema estocástico de test (Figura 2-11), a fin de poder evaluar el efecto de la circuitería adicional incorporada sobre el rendimiento de la metodología. Se ha implementado el sistema anteriormente descrito (*B2P(N)* incorporando la *LFSRs* de 26-bits), para determinar el consumo estándar de los recursos lógicos de la FPGA. Acto seguido se ha procedido a repetir el proceso con la metodología propuesta.

En la Tabla 2-1 se presentan los resultados experimentales obtenidos. Estos resultados muestran cómo mediante la metodología propuesta el consumo de elementos lógicos disminuye a medida que se van aumentando las estructuras de test, llegando a una reducción máxima de recursos lógicos cercana al 16% para estructuras de computación estocásticas densas (con 200 circuitos de test operando en paralelo).

2.2.2.3 Correlaciones entre señales estocásticas

Los sistemas estocásticos se implementan como sistemas síncronos (controlados por una señal global de reloj, que estipula los períodos de evaluación y los de mantenimiento). A cada evento de reloj las señales estocásticas mantienen fijo su valor aleatorio. Por otra parte las señales se relacionan entre ellas mediante puertas o bloques lógicos que se rigen por las leyes probabilísticas. Así por ejemplo mediante una puerta **AND** obtenemos a la salida de la puerta el producto de las dos señales de entrada (que se corresponde con la probabilidad de colisión entre estas dos señales). Cabe remarcar que uno de los requerimientos básicos de la computación estocástica consiste en la necesaria descorrelación entre las distintas señales a operar. En donde entenderemos como la descorrelación de señales a la independencia entre ellas (Figura 2-12).

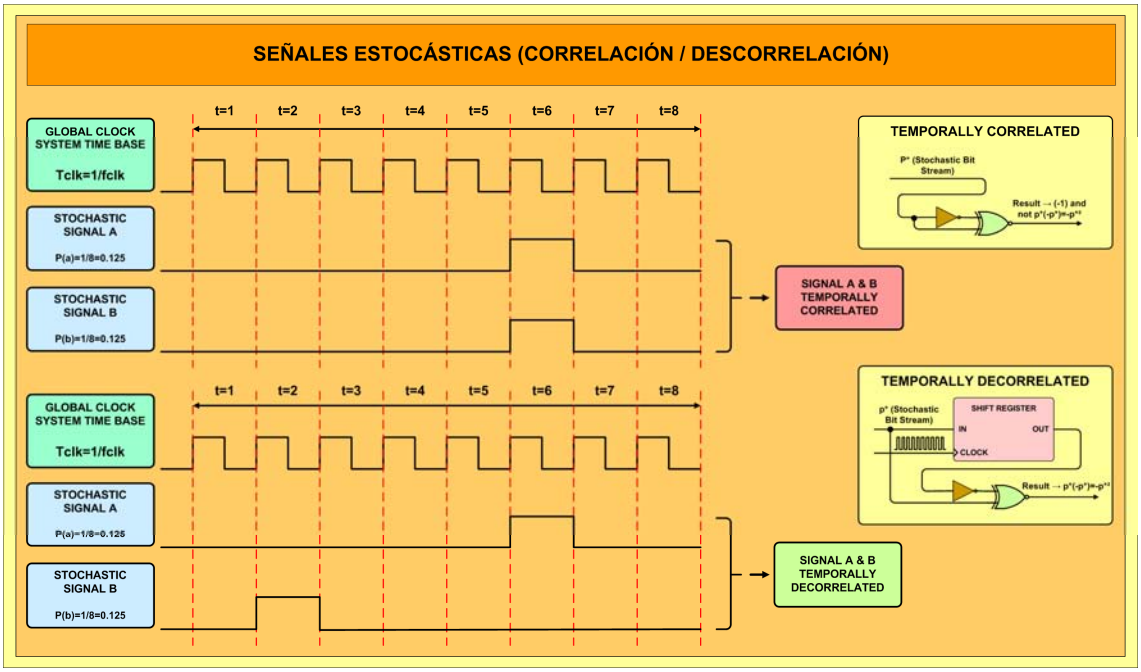

Figura 2-12: Impacto de la correlación sobre la computación estocástica

En la Figura 2-12 se presenta de forma esquemática la importancia de la descorrelación de señales cuando se concatenan funciones o bloques aritméticos a fin de obtener un determinado resultado. A fin de presentar este fenómeno de una forma simple, se ha optado por la implementación de la función " $f(p) = p \cdot (1-p) = p - p^2$ ". Como se puede apreciar

en la primara implementación (parte superior derecha de la Figura 2-12), al existir correlación temporal entre ambas señales “p” y “1-p” (ya que provienen de la misma señal estocástica) la salida del circuito será siempre un nivel bajo ‘0’. En el instante en que la señal p es ‘1’y la señal 1-p es ‘0’ y viceversa, la salida siempre se hallará fija a nivel bajo ‘0’. Con objeto de eliminar la correlación y permitir un correcto funcionamiento del circuito estocástico, será necesario descorrelacionar temporalmente ambas señales, para lo cual deberemos insertar elementos de memoria: biestables D para obtener un retardo de un ciclo de reloj entre señales, o registros de desplazamiento si se pretende descorrelacionar las señales estocásticas más de un ciclo de reloj. Generalmente haremos uso de registros de desplazamiento de 8-bits para obtener una descorrelación efectiva.

Si nos fijamos ahora en la parte inferior de la Figura 2-12 podremos apreciar cómo con la inserción de un registro de desplazamiento se ha eliminado la correlación. De esta forma se obtiene el resultado correcto del producto entre el valor de la señal de “p” y el valor de la señal “1-p”.

2.2.3 LA CODIFICACIÓN ESTOCÁSTICA CLÁSICA (UCSL)

La lógica estocástica en su codificación clásica ha sido abordada en la literatura desde mediados de la década de 1960 [2-1, 2-29, 2-35], a dicha codificación la denominaremos "*Unsigned Classic Stochastic Logic*" (*UCSL*). Esta lógica a diferencia de la lógica digital clásica (electrónica digital) que se basa en señales deterministas, se basa en el uso de señales digitales aleatorias. Estas tramas digitales consisten en secuencias temporales de ceros y unos que son procesadas mediante puertas lógicas elementales (iguales que las usadas en la electrónica digital clásica), tales como pueden ser las puertas **AND**, **OR**, etc. En el caso de la codificación estocástica clásica sin signo los valores de dicha distribución estadística representarán valores reales en el intervalo [0, +1], que corresponden a la probabilidad de ocurrencia de los '1' lógicos frente a la de los '0' lógicos en un determinado intervalo de observación. Mediante esta codificación las señales estocásticas "x" únicamente podrán ser definidas para un espacio $\Re^+$ acotado $x \in \left|0,1\right|$, lo que implica que no se podrán representar magnitudes negativas ni tampoco mayores a +1. Esta es una de las limitaciones básicas de la codificación estocástica clásica. Cabe remarcar, que esta lógica tiene la ventaja de permitir realizar las diversas funciones aritméticas básicas con puertas/bloques lógicas básicas [2-11, 2-19], aunque tiene la desventaja de llevar asociado un error de estimación inherente relacionado con la varianza de las señales. De todas formas, esta lógica será óptima para determinadas aplicaciones de inteligencia computacional, donde la precisión de las operaciones no es un requerimiento crítico (como el reconocimiento de patrones, las redes neuronales artificiales y el procesado de imágenes/señales).

Las principales funciones matemáticas desarrolladas para la codificación estocástica clásica sin signo se presentan en lo largo de los siguientes subapartados con sus correspondientes implementaciones digitales y comprobaciones experimentales. La implementación de funciones matemáticas tales como pueden ser la suma, la multiplicación, la división y algunas otras ya han sido desarrolladas y presentadas en publicaciones científicas por otros investigadores [2-10, 2-11, 2-19, 2-30]. Cabe remarcar que las implementaciones

realizadas, y que a continuación se presentan son completamente originales defiriendo en muchos casos notablemente de las conocidas hasta el momento.

2.2.3.1 La operación complementaria

La lógica estocástica en su codificación clásica (*UCSL*) no permite la presentación de magnitudes con signo al estar la información definida en el intervalo [0, +1], por ello este bloque determina la señal estocástica complementaria asociada a una magnitud binaria "P" definida en [0, +1]. La función complementaria "1-p" (Ecuación [2-11]) se implementa mediante una puerta **NOT** con su entrada conectada a la señal estocástica "p" (generada a partir de la magnitud "P" a través de un módulo *B2P(N)*.

$$P_{Magnitud\_binaria} \rightarrow p_{señal\_estocástica\_1bit} \xrightarrow[Complementario]{} OUT = P^* = (1-P) \rightarrow out = NOT(p) = (1-p)$$

Ecuación [2-11]

En la figura 2-13, se presenta la implementación digital del bloque *señal complementaria* mediante la codificación estocástica clásica (sin signo).

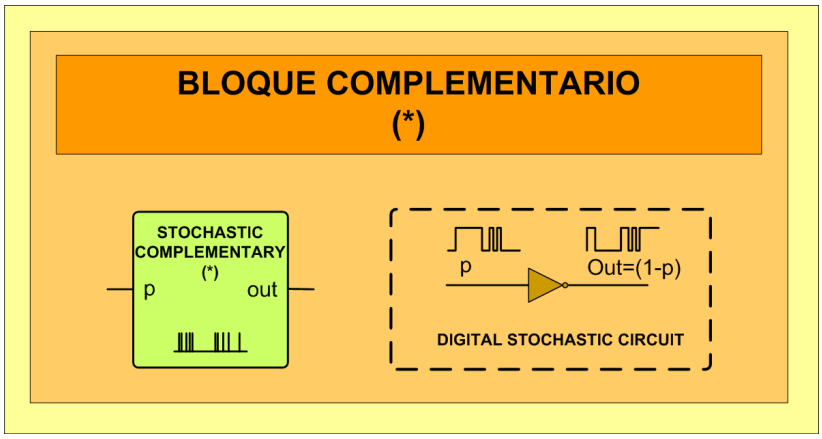

Figura 2-13: Bloque de *señal complementaria* (UCSL)

Para comprobar el correcto funcionamiento del *bloque señal complementaria* se ha procedido a realizar una serie de medidas experimentales (Tabla 2-2). Se ha variado el valor de "P", y se ha medido el valor a la salida del bloque a fin de comprobar si el funcionamiento de la implementación realizada se corresponde con el funcionamiento predicho para dicho bloque.

Tabla 2-2: Medidas del bloque operación complementaria (UCSL)			
DATOS DE ENTRADA		EXPERIMENTAL	
Valor P	P [16bits]	OUT [16bits]	Valor OUT
0,0000	0	65535	1,0000
0,0313	2048	63539	0,9695
0,0625	4096	61363	0,9363
0,0938	6144	59490	0,9078
0,1250	8192	57346	0,8750
0,1563	10240	55311	0,8440
0,1875	12288	53328	0,8137
0,2188	14336	51227	0,7817
0,2500	16384	49181	0,7505
0,2813	18432	47144	0,7194
0,3125	20480	45045	0,6873
0,3438	22528	43084	0,6574
0,3750	24576	40873	0,6237
0,4063	26624	39038	0,5957
0,4375	28672	36774	0,5611
0,4688	30720	34868	0,5321
0,5000	32768	32821	0,5008
0,5313	34816	30590	0,4668
0,5625	36864	28668	0,4374
0,5938	38912	26503	0,4044
0,6250	40960	24328	0,3712
0,6563	43008	22586	0,3446
0,6875	45056	20517	0,3131
0,7188	47104	18337	0,2798
0,7500	49152	16406	0,2503
0,7813	51200	14412	0,2199
0,8125	53248	12262	0,1871
0,8438	55296	10171	0,1552
0,8750	57344	8239	0,1257
0,9063	59392	6141	0,0937
0,9375	61440	4088	0,0624
0,9688	63488	2056	0,0314

Si representamos gráficamente (Figura 2-14) los resultados experimentales obtenidos realizando un ajuste por mínimos cuadrados de los datos representados, obtenemos la función de transferencia del bloque.

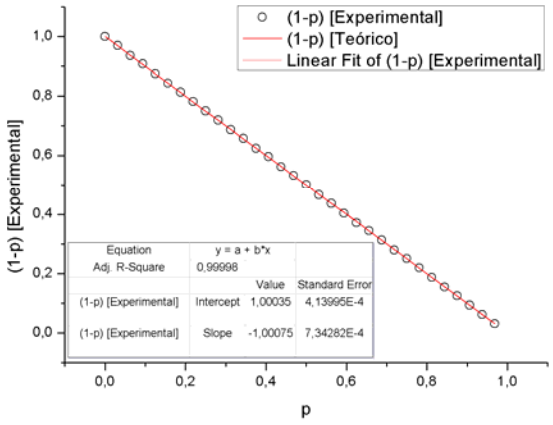

Figura 2-14: Representación gráfica de las medidas experimentales del bloque señal complementaria (UCSL)

Si comparamos la función de transferencia obtenida (Figura 2-14) con la descrita por la teoría (Ecuación [2-12]) para dicho bloque podemos apreciar cómo ambas son idénticas.

$$\begin{aligned} &\text{Teórica} \quad \rightarrow OUT_{TEO} = (1-P) \\ &\text{Experimental} \quad \rightarrow OUT_{EXP} = (1-1{,}0002 \cdot P), \text{ con una } R^2 = 1 \end{aligned}$$

Ecuación [2-12]

2.2.3.2 La operación suma en valor medio

En la codificación estocástica clásica sin signo la implementación de la suma entre dos magnitudes binarias "P" y "Q" se viene históricamente realizando mediante una puerta **OR** de dos entradas [2-1].

$$\left.\begin{array}{ll} P_{Magnitud\_binaria} & \rightarrow p_{señal\_estocástica\_1bit} \\ Q_{Magnitud\_binaria} & \rightarrow q_{señal\_estocástica\_1bit} \\ OUT_{Magnitud\_binaria} & \leftarrow out_{señal\_estocástica\_1bit} \end{array}\right\} \xrightarrow{Suma} \begin{cases} OUT = (P+Q) \rightarrow E(out(p,q)) = \\ = E(p \text{ OR } q)P = P(p) + P(q) - P(p) \cdot P(q) = \\ = \dfrac{P}{2^n} + \dfrac{Q}{2^n} - \dfrac{P \cdot Q}{2^{2n}} = (\max 1,\ p + q - p \cdot q) \end{cases}$$

Ecuación [2-13]

A la salida de la puerta hallaremos una señal que estará relacionada con la suma de ambas probabilidades menos un factor proporcional a la probabilidad que ambas entradas estén

activas a la vez (probabilidad de colisión), cuya media se determina mediante la Ecuación [2-13].

Dicha implementación como bien se describe en la literatura [2-11] minusvalora el valor de la suma para valores de "P" y/o "Q" medios y grandes, puesto que dicha implementación solamente es válida para señales estocásticas de entrada con un valor medio asociado muy pequeño que asegure la existencia de muy poco solapamiento en ambas señales.

A fin de comprobar las discrepancias descritas por la literatura entre la función que se pretende implementar (Suma) y la que realmente implementa mediante una puerta **OR** se ha procedido a realizar una serie de medidas experimentales (Tabla 2-3), fijando "Q=0,5", mientras que "P" se ha variado en el intervalo [0, +1].

Tabla 2-3: Medidas de la suma subestimada (mediante puerta OR) (UCLS)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
Valor P	P [16bits]	Valor Q	Q [16bits]	OUT [16bits]	Valor OUT	OUT=P+Q	ERROR
0,0000	0	0,0000	0	32859	0,5014	0,5000	0,0014
0,0313	2048	0,5000	32768	33636	0,5133	0,5313	0,0180
0,0625	4096	0,5000	32768	34867	0,5320	0,5625	0,0305
0,0938	6144	0,5000	32768	35674	0,5444	0,5938	0,0494
0,1250	8192	0,5000	32768	37052	0,5654	0,6250	0,0596
0,1563	10240	0,5000	32768	38184	0,5827	0,6563	0,0736
0,1875	12288	0,5000	32768	38755	0,5914	0,6875	0,0961
0,2188	14336	0,5000	32768	39940	0,6094	0,7188	0,1093
0,2500	16384	0,5000	32768	40837	0,6231	0,7500	0,1269
0,2813	18432	0,5000	32768	41805	0,6379	0,7813	0,1434
0,3125	20480	0,5000	32768	42580	0,6497	0,8125	0,1628
0,3438	22528	0,5000	32768	44149	0,6737	0,8438	0,1701
0,3750	24576	0,5000	32768	45212	0,6899	0,8750	0,1851
0,4063	26624	0,5000	32768	46067	0,7029	0,9063	0,2033
0,4375	28672	0,5000	32768	46904	0,7157	0,9375	0,2218
0,4688	30720	0,5000	32768	48129	0,7344	0,9688	0,2344
0,5000	32768	0,5000	32768	49093	0,7491	1,0000	0,2509
0,5313	34816	0,5000	32768	49961	0,7624	1,0313	0,2689
0,5625	36864	0,5000	32768	51427	0,7847	1,0625	0,2778
0,5938	38912	0,5000	32768	52199	0,7965	1,0938	0,2973
0,6250	40960	0,5000	32768	52987	0,8085	1,1250	0,3165
0,6563	43008	0,5000	32768	54274	0,8282	1,1563	0,3281
0,6875	45056	0,5000	32768	55120	0,8411	1,1875	0,3464
0,7188	47104	0,5000	32768	56287	0,8589	1,2188	0,3599
0,7500	49152	0,5000	32768	57445	0,8766	1,2500	0,3735
0,7813	51200	0,5000	32768	58403	0,8912	1,2813	0,3901
0,8125	53248	0,5000	32768	59377	0,9060	1,3125	0,4065
0,8438	55296	0,5000	32768	60358	0,9210	1,3438	0,4228
0,8750	57344	0,5000	32768	61561	0,9394	1,3750	0,4357
0,9063	59392	0,5000	32768	63404	0,9675	1,4063	0,4388
0,9375	61440	0,5000	32768	64540	0,9848	1,4375	0,4527
0,9688	63488	0,5000	32768	64472	0,9838	1,4688	0,4850

Como puede apreciarse (Tabla 2-3) para valores cercanos a “1” de la magnitud binaria “P”, la puerta lógica **OR** minusvalora hasta en un 50% el resultado de la suma. Además el bloque suma satura a “+1” para valores superiores a dicho valor.

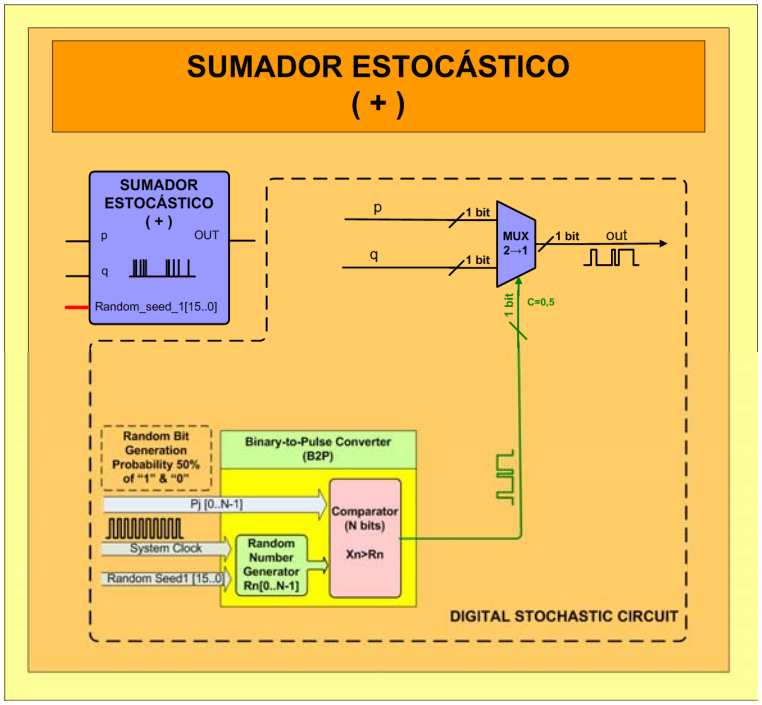

Figura 2-15: Bloque sumador valor medio (UCLS)

Para dar solución a este problema se propone una nueva aproximación para la realización de la suma estocástica basada en el uso de un multiplexor digital. En este caso el multiplexor dispondrá de dos entradas “p” y “q” (a través de las cuales se introducirán las señales a operar) y de una señal adicional de selección “c”, obteniendo a la salida del bloque una nueva trama con distribución de probabilidad de activación proporcional a una combinación lineal de las distribuciones de probabilidad de las señales operadas. Dicha combinación lineal está forzosamente relacionada con la señal de control de la activación, que deberá hallarse descorrelada con respecto a las dos señales a operar. El valor esperado o valor medio de la distribución resultante se describe mediante la Ecuación [2-14].

$$\left.\begin{array}{ll} P_{Magnitud\_binaria} & \rightarrow \mathrm{p}_{\text{señal\_estocástica\_1bit}} \\ Q_{Magnitud\_binaria} & \rightarrow \mathrm{q}_{\text{señal\_estocástica\_1bit}} \\ \mathrm{OUT}_{\text{Magnitud\_binaria}} & \leftarrow \mathrm{out}_{\text{señal\_estocástica\_1bit}} \end{array}\right\} \xrightarrow{Suma} \begin{cases} OUT = (P+Q) \rightarrow E(out(p,q,c)) = \\ = E(c\cdot p + (1-c).q) = P(c)\cdot P(p) + P(1-c)\cdot P(q) = \\ = \dfrac{C\cdot P}{2^{2n}} + \dfrac{(1-C)\cdot Q}{2^{2n}} \end{cases}$$

Para asegurar que la suma entre "p" y "q" esté ponderada se deberá elegir un valor de la señal de control de activación "c", tal que ambas señales dispongan del mismo tiempo de evaluación en un periodo de integración, se requerirá que :

$c = 1 - c \rightarrow 2\cdot c = 1 \rightarrow c = 0{,}5 \rightarrow E(out) = 0{,}5\cdot(P(p) + P(q))$

Para el valor máximo de "p" y "q" se cumplirá con el límite superior "1" $\rightarrow 0{,}5\cdot(1+1) = 1$

Para el valor mínimo de "p" y "q" se cumplirá con el límite inferior "0" $\rightarrow 0{,}5\cdot(0+0) = 0$

Ecuación [2-14]

De la Ecuación [2-14] se deduce que la aproximación propuesta implementa la suma ponderada de la distribución de probabilidad de las señales pulsantes. Por lo tanto el valor medio "$\overline{out}$" que hallaremos contenidos en el acumulador de nuestro bloque P2B(N) después de "$N_{ciclos\_evaluación}$" será una variable aleatoria de valor medio (Ecuación [2-15]):

$$\overline{out} = N_{ciclos\_evaluación} \cdot \frac{0{,}5\cdot(P+Q)}{2^{2n}}$$

Ecuación [2-15]

La desviación típica del valor acumulado en el bloque P2B(N) se describe mediante la Ecuación [2-16]:

$$\sigma(\overline{out}) = \sqrt{N_{ciclos\_evaluación} \cdot \frac{0{,}5\cdot(P+Q)}{2^{2n}} \cdot \left(1 - \frac{0{,}5\cdot(P+Q)}{2^{2n}}\right)}$$

Ecuación [2-16]

La implementación digital de la operación suma se presenta en la Figura 2-15. Para probar el correcto funcionamiento del *bloque sumador en valor medio* se ha procedido a realizar una serie de medidas experimentales (Tabla 2-4). En éstas se ha fijado el valor de "Q=0,5" y se ha variado el valor de "P" entre 0 y +1 a fin de comprobar si el funcionamiento de la implementación realizada se corresponde con el funcionamiento predicho teóricamente.

Tabla 2-4: Medidas bloque sumador en valor medio (UCSL)

DATOS DE ENTRADA				EXPERIMENTAL	
Valor P	P [16bits]	Q [16bits]	Valor Q	OUT [16bits]	Valor OUT
0,0000	0	32768	0,500	16364	0,2497
0,0313	2048	32768	0,500	17367	0,2650
0,0625	4096	32768	0,500	18541	0,2829

0,0938	6144	32768	0,500	19459	0,2969
0,1250	8192	32768	0,500	20274	0,3094
0,1563	10240	32768	0,500	21415	0,3268
0,1875	12288	32768	0,500	22600	0,3449
0,2188	14336	32768	0,500	23511	0,3588
0,2500	16384	32768	0,500	24305	0,3709
0,2813	18432	32768	0,500	25658	0,3915
0,3125	20480	32768	0,500	26487	0,4042
0,3438	22528	32768	0,500	27574	0,4208
0,3750	24576	32768	0,500	28576	0,4360
0,4063	26624	32768	0,500	29892	0,4561
0,4375	28672	32768	0,500	30624	0,4673
0,4688	30720	32768	0,500	31632	0,4827
0,5000	32768	32768	0,500	32781	0,5002
0,5313	34816	32768	0,500	33873	0,5169
0,5625	36864	32768	0,500	34864	0,5320
0,5938	38912	32768	0,500	35716	0,5450
0,6250	40960	32768	0,500	36659	0,5594
0,6563	43008	32768	0,500	37559	0,5731
0,6875	45056	32768	0,500	38675	0,5901
0,7188	47104	32768	0,500	39892	0,6087
0,7500	49152	32768	0,500	41084	0,6269
0,7813	51200	32768	0,500	41772	0,6374
0,8125	53248	32768	0,500	43047	0,6569
0,8438	55296	32768	0,500	43842	0,6690
0,8750	57344	32768	0,500	45344	0,6919
0,9063	59392	32768	0,500	45960	0,7013
0,9375	61440	32768	0,500	47045	0,7179
0,9688	63488	32768	0,500	48207	0,7356

Si se representan gráficamente los resultados experimentales (Figura 2-16), y realizamos un ajuste lineal a estos datos obtendremos la función de transferencia del bloque.

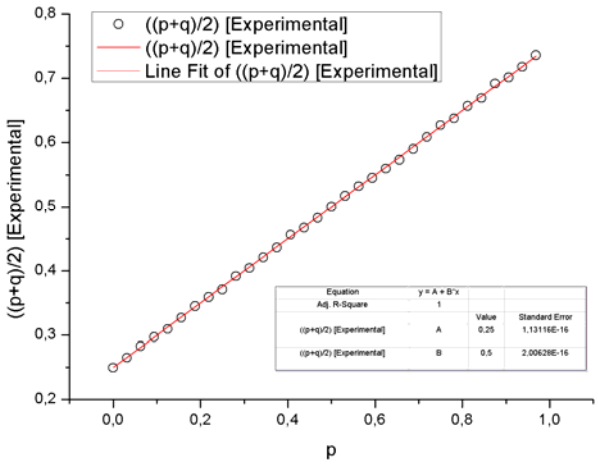

Figura 2-16: Representación gráfica de las medidas experimentales del bloque sumador en valor medio (UCSL)

A partir de la función de transferencia obtenida (Ecuación [2-17]) se puede demostrar cómo el bloque implementado realiza la suma en valor medio de las distribuciones de probabilidad asociadas a las dos señales estocásticas de entrada.

$$\begin{aligned} &\text{Teórica} \quad \rightarrow OUT_{TEO} = 0{,}5000\cdot(P+Q) \\ &\text{Experimental} \quad \rightarrow OUT_{EXP} = 0{,}4988\cdot(P+Q), \text{ con una } R^2 = 0{,}9998 \end{aligned} \qquad \text{Ecuación [2-17]}$$

Éste es uno de los más graves inconvenientes que presenta la codificación estocástica clásica que debe tenerse muy presente; ya que se refiere a la imposibilidad de implementar la función suma de forma exacta.

2.2.3.3 La operación resta en valor absoluto

Otro elemento imprescindible de los que se debe disponer es el bloque restador. La implementación de dicho bloque en nuestra aproximación es completamente diferente a la de los anteriores bloques descritos hasta el momento. En las operaciones descritas con anterioridad era condición necesaria que la señales estuvieran descorrelacionadas entre ellas. En oposición a este tipo de operaciones, para la realización de la resta estocástica es necesario que exista una correlación máxima entre señales. De esta forma la resta se puede implementar mediante una puerta **XOR**, ya que como se muestra en la tabla de verdad (Tabla 2-5) esta puerta realiza la resta (elimina cualquier coincidencia entre señales) en valor absoluto (debido a la imposibilidad de representar valores negativos en la codificación estocástica clásica) de dos señales estocásticas siempre y cuando las dos señales estén correlacionadas.

Tabla 2-5: Tabla de verdad XOR

P	Q	OUT
0	0	0
1	0	1
1	1	0
0	1	1

En consecuencia, la distribución estadística de la trama de salida del bloque tendrá asociada un valor medio que vendrá regido por la Ecuación [2-18].

$$\left.\begin{array}{l} p \rightarrow \text{Señal estocástica descorrelacionada} \\ q \rightarrow \text{Señal estocástica descorrelacionada} \\ p' \rightarrow \text{Señal estocástica correlacionada} \\ q' \rightarrow \text{Señal estocástica correlacionada} \end{array}\right\} \xrightarrow{\text{Resta}} \begin{cases} E\left(out\left(p,q\right)\right) = p' \oplus q' = \left|p-q\right| = \\ = \left|\dfrac{P}{2^n} - \dfrac{Q}{2^n}\right| \end{cases}$$

Ecuación [2-18]

Es de reseñar que la resta en valor absoluto está completamente definida en el espacio de representación [0, +1]. El principal inconveniente de esta aproximación es el hecho que dicho bloque es lento desde un punto de vista computacional, puesto que requiere de al menos 2^{17} ciclos de reloj para convertir una señal estocástica descorrelacionada en otra idéntica pero correlacionada. Este hecho se ha de unir al retardo mínimo que se dará en la evaluación de la puerta **XOR**.

En la Figura 2-17 se presenta el diagrama de bloques del circuito *Restador en valor absoluto* que se ha desarrollado. En él se puede apreciar como sendos bloques *P2B(N)* se encargan de evaluar el valor medio de la distribución de probabilidades de las señales pulsantes de entrada “p” y “q”, para luego seguidamente proceder a reconvertir sendos valores en sus equivalente pulsantes correlacionados mediante bloques *B2P(N)*, que hacen uso del mismo generador de números aleatorios. para finalmente dichas señales se operan mediante una puerta **XOR**.

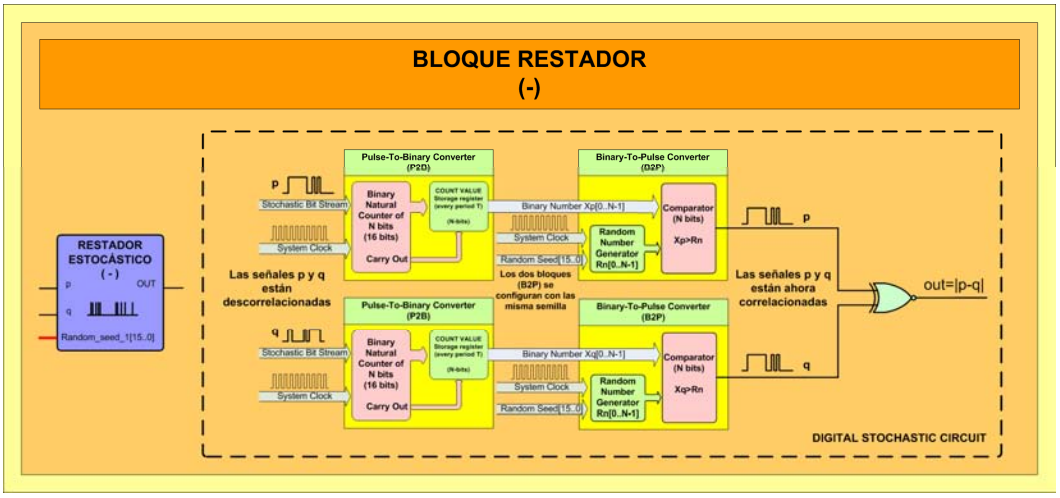

Figura 2-17: Bloque Restador en valor absoluto (UCLS)

Para comprobar el correcto funcionamiento del *bloque Restador en valor absoluto* desarrollado se han procedido a realizar una serie de medidas experimentales (Tabla 2-6); en las cuales se ha fijado el valor de "Q=0,5" y se ha variado el valor de "P" entre [0, +1].

Tabla 2-6: Medidas bloque restador en valor absoluto (UCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	Q [16bits]	Valor Q	OUT [16bits]	Valor OUT	OUT=|P-Q|	ERROR
0	0,0000	32768	0,5000	32760	0,4999	0,5000	0,0001
4096	0,0625	32768	0,5000	28665	0,4374	0,4375	0,0001
8192	0,1250	32768	0,5000	24625	0,3758	0,3750	0,0007
12288	0,1875	32768	0,5000	20441	0,3119	0,3125	0,0006
16384	0,2500	32768	0,5000	16534	0,2523	0,2500	0,0023
20480	0,3125	32768	0,5000	12440	0,1898	0,1875	0,0023
24576	0,3750	32768	0,5000	8205	0,1252	0,1250	0,0002
28672	0,4375	32768	0,5000	4019	0,0613	0,0625	0,0012
32768	0,5000	32768	0,5000	0	0,0000	0,0000	0,0000
36864	0,5625	32768	0,5000	4079	0,0622	0,0625	0,0003
40960	0,6250	32768	0,5000	8207	0,1252	0,1250	0,0002
45056	0,6875	32768	0,5000	12241	0,1868	0,1875	0,0007
49152	0,7500	32768	0,5000	16316	0,2490	0,2500	0,0010
53248	0,8125	32768	0,5000	20399	0,3113	0,3125	0,0012
57344	0,8750	32768	0,5000	24503	0,3739	0,3750	0,0011
61440	0,9375	32768	0,5000	28587	0,4362	0,4375	0,0013

Si representamos gráficamente los datos de la Tabla 2-6 obtenemos la Figura 2-18.

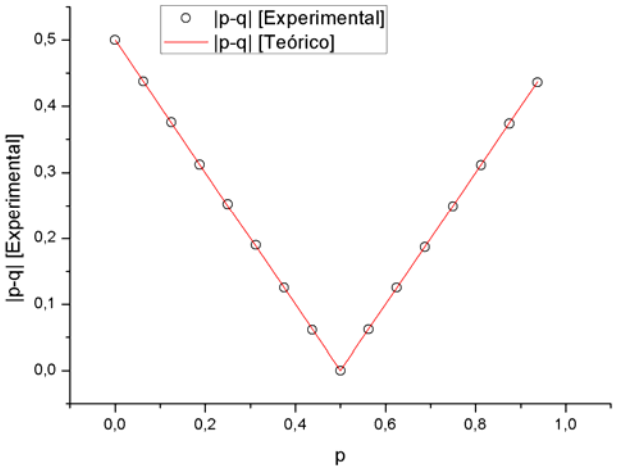

Figura 2-18: Resultados experimentales del bloque Restador en valor absoluto (UCLS)

A fin de comprobar el óptimo funcionamiento del módulo en todo el espacio 2D de representación de las señales “p” y “q” se ha realizado un barrido de 1024 medidas de la salida del bloque variando a la vez “P” y “Q” en [0, +1], para comprobar si la función se comportaba correctamente en todo el espacio de representación. En la Figura 2-19 se presentan las medidas realizadas.

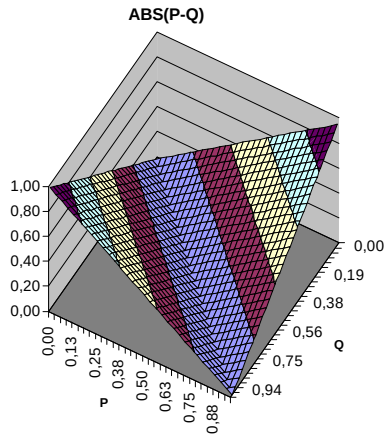

Figura 2-19: Barrido 2D del bloque Restador en valor absoluto (UCLS)

Como puede apreciarse en la Figura 2-18, los valores experimentales son prácticamente idénticos a los predichos por la teoría en la Ecuación [2-13], con lo cual se demuestra que la implementación realizada es correcta.

2.2.3.4 La operación multiplicación

En la codificación estocástica clásica sin signo, la implementación de la multiplicación la (Ecuación [2-19]) entre dos magnitudes binarias “P” y “Q” se ha realizado mediante [2-1] una puerta **AND**.

$$\left.\begin{array}{l} P_{Magnitud\_binaria} \rightarrow \mathrm{p}_{\text{señal\_estocástica\_1bit}} \\ Q_{Magnitud\_binaria} \rightarrow \mathrm{q}_{\text{señal\_estocástica\_1bit}} \\ \mathrm{OUT}_{\text{Magnitud\_binaria}} \leftarrow \mathrm{out}_{\text{señal\_estocástica\_1bit}} \end{array}\right\} \xrightarrow{Multiplicación} \left\{\begin{array}{l} OUT = (P.Q) \rightarrow E(out(p,q)) = \\ = E(p \text{ AND } q) = P(p)\cdot P(q) = \\ = \dfrac{P\cdot Q}{2^{2n}} \end{array}\right.$$

Ecuación [2-19]

Cabe destacar, que el valor medio de la distribución probabilidad asociadas a las señales estocásticas “p” y “q” será un valor comprendido en el intervalo [0, +1], lo que implicará que la multiplicación entre ambas distribuciones de probabilidad nunca proporcionará una distribución con una media mayor a +1, siendo completamente definida en el espacio de representación de la lógica estocástica sin signo.

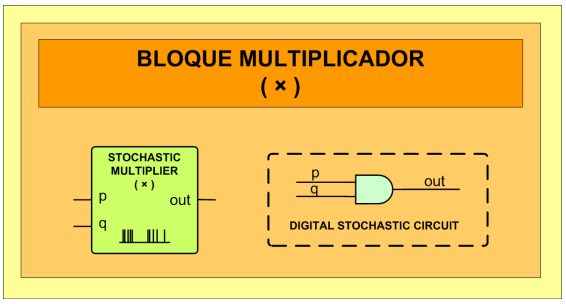

Figura 2-20: Bloque multiplicador estocástico (UCLS)

La implementación digital del bloque multiplicador de dos señales estocásticas se presenta en la Figura 2-20. Por lo tanto la salida “$\overline{out}$”del bloque *P2B(N)* conectado a la salida “out” del bloque multiplicador, después de “$N_{ciclos\_evaluación}$” será una variable aleatoria de valor (Ecuación [2-20]):

$$\overline{out} = N_{ciclos\_evaluación} \cdot \frac{P \cdot Q}{2^{2n}} \qquad \text{Ecuación [2-20]}$$

La desviación típica del valor acumulado en el bloque *P2B(N)* se describe mediante la Ecuación [2-21]:

$$\sigma\left(\overline{out}\right) = \sqrt{N_{ciclos\_evaluación} \cdot \frac{P \cdot Q}{2^{2n}} \cdot \left(1 - \frac{P \cdot Q}{2^{2n}}\right)} \qquad \text{Ecuación [2-21]}$$

Para comprobar el correcto funcionamiento del *bloque multiplicador* se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-7. Éstashan consistido en fijar constante el valor de “Q=0,0630” e ir variando el valor de “P” entre [0, +0.9688].

Tabla 2-7: Medidas del bloque Multiplicación (UCSL)	
DATOS DE ENTRADA	**EXPERIMENTAL**

Valor P	P [16bits]	Valor Q	Q [16bits]	OUT [16bits	Valor OUT
0,0000	0	0,000	0	0	0,0000
0,0313	2048	0,063	4096	132	0,0020
0,0625	4096	0,063	4096	237	0,0036
0,0938	6144	0,063	4096	395	0,0060
0,1250	8192	0,063	4096	510	0,0078
0,1563	10240	0,063	4096	653	0,0100
0,1875	12288	0,063	4096	758	0,0116
0,2188	14336	0,063	4096	901	0,0137
0,2500	16384	0,063	4096	1077	0,0164
0,2813	18432	0,063	4096	1186	0,0181
0,3125	20480	0,063	4096	1292	0,0197
0,3438	22528	0,063	4096	1346	0,0205
0,3750	24576	0,063	4096	1566	0,0239
0,4063	26624	0,063	4096	1628	0,0248
0,4375	28672	0,063	4096	1803	0,0275
0,4688	30720	0,063	4096	1883	0,0287
0,5000	32768	0,063	4096	2051	0,0313
0,5313	34816	0,063	4096	2207	0,0337
0,5625	36864	0,063	4096	2303	0,0351
0,5938	38912	0,063	4096	2449	0,0374
0,6250	40960	0,063	4096	2576	0,0393
0,6563	43008	0,063	4096	2704	0,0413
0,6875	45056	0,063	4096	2802	0,0428
0,7188	47104	0,063	4096	2932	0,0447
0,7500	49152	0,063	4096	3093	0,0472
0,7813	51200	0,063	4096	3218	0,0491
0,8125	53248	0,063	4096	3349	0,0511
0,8438	55296	0,063	4096	3488	0,0532
0,8750	57344	0,063	4096	3568	0,0544
0,9063	59392	0,063	4096	3689	0,0563
0,9375	61440	0,063	4096	3854	0,0588
0,9688	63488	0,063	4096	3976	0,0607

Si representamos gráficamente los datos obtenidos en la Tabla 2-7 y realizamos un ajuste por mínimos cuadrados de los datos representados, obtendremos la función de transferencia que se presenta en la Figura 2-21.

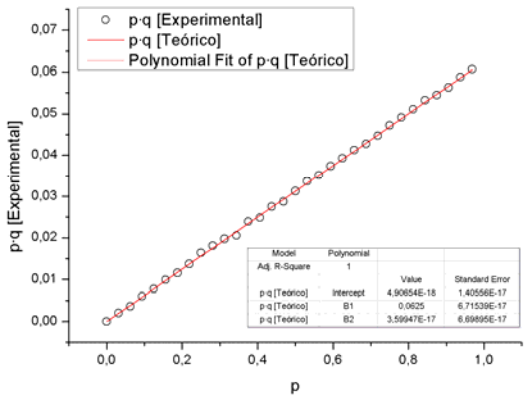

Figura 2-21: Resultados experimentales del bloque Multiplicador (UCLS)

Si finalmente se compara la función de transferencia obtenida (Ecuación [2-22]) con la descrita por la teoría (Ecuación [2-19]) para dicho bloque, podremos apreciar como ambas son prácticamente idénticas.

$$\text{Experimental} \quad \rightarrow OUT_{EXP} = 0{,}0626 \cdot P \text{, con una } R^2 = 0{,}9996$$
$$\text{Teórico} \quad \rightarrow OUT_{TEO} = 0{,}0630 \cdot P$$

Ecuación [2-22]

La pequeña discrepancia entre ambas proviene principalmente de la dispersión asociada al proceso de conversión de las magnitudes binarias a pulsantes y de la falta de resolución en el proceso de conversión de las señales pulsantes a magnitudes binarias (debido al uso de un número de ciclos de integración no lo suficientemente grande).

2.2.3.5 La operación potenciación (P^N)

Para implementar la operación potencia N-ésima de una señal pulsante "p" es necesario multiplicarla por ella misma un número determinado de veces (N), intentando en todo momento mantener la descorrelación entre las señales que se multipliquen.

Como ejemplo de potenciación se presenta la operación cuadrado (potencia 2) para una señal estocástica "p" de entrada. Esta función se ha desarrollado debido a que es necesaria para la implementación de la función Gaussiana que se describe en los siguientes subapartados. La implementación de la función cuadrado consistirá en un bloque multiplicador (puerta AND) y de un elemento encargado de descorrelacionar (registro de desplazamiento) las señales a multiplicar, que en este caso son la misma. Después de numerosas pruebas, se ha determinado que la profundidad óptima del registro de desplazamiento a usar para la incorrelación entre señales debe ser de 8-bits.

La implementación digital del *bloque cuadrado* (potencia 2) la que se presenta en la Figura 2-22

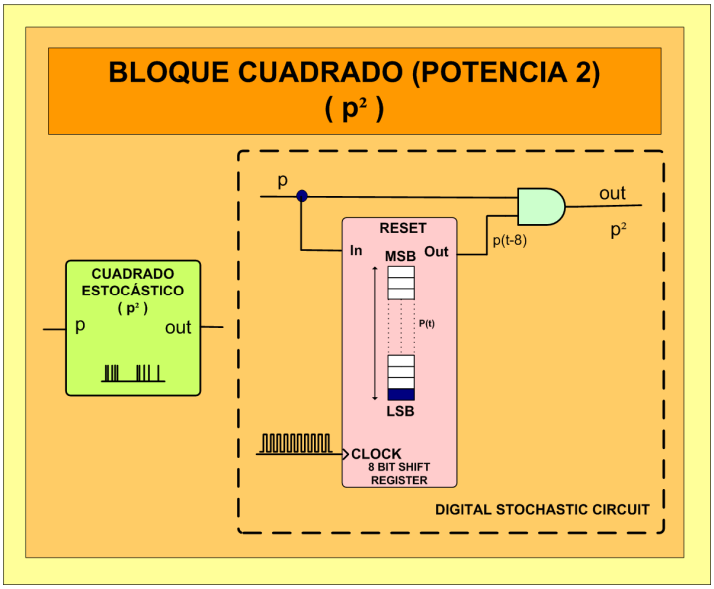

Figura 2-22: Bloque potencia 2 (UCLS)

Al igual que sucede con la multiplicación entre dos señales pulsantes diferentes, en el caso de la potenciación el resultado de multiplicar una distribución de probabilidad definida en el espacio de representación [0, +1] por si misma da lugar a una nueva distribución de probabilidad cuyo valor medio será proporcional al cuadrado del valor medio de la distribución de entrada, el valor del cual se hallará completamente definido en el rango [0, +1] de la lógica estocástica sin signo. Dicha operación se regirá por la Ecuación [2-23]:

$$\left.\begin{array}{l} P_{Magnitud\_binaria} \rightarrow \mathrm{p}_{\text{señal\_estocástica\_1bit}} \\ \mathrm{OUT}_{\text{Magnitud\_binaria}} \leftarrow \mathrm{out}_{\text{señal\_estocástica\_1bit}} \end{array}\right\} \xrightarrow{Cuadrado} \begin{cases} OUT = (P \cdot Q) \rightarrow E(out(p(t), p(t-8))) = \\ = E(p(t) \text{ AND } p(t-8)) = P(p) \cdot P(p) = \\ = \dfrac{P^2}{2^{2n}} \end{cases}$$

Por lo tanto, si $\mathrm{P} \in [0,1] \rightarrow \mathrm{P}^2 \in [0,1]$

Ecuación [2-23]

Para comprobar el correcto funcionamiento del *bloque cuadrado (potencia 2)* se ha procedido a realizar unas medidas experimentales (Tabla 2-8) variando el valor medio de la distribución de probabilidades asociada a la señal pulsante "p" en el intervalo [0, +1].

Tabla 2-8: Medidas del bloque potencia 2 (UCSL)

DATOS DE ENTRADA		EXPERIMENTAL	
P	Valor P (16bits)	OUT(16bits)	OUT
0,0000	0	71	0,0011
0,0313	2048	240	0,0037
0,0625	4096	545	0,0083
0,0938	6144	1002	0,0153
0,1250	8192	1060	0,0162
0,1563	10240	1694	0,0258
0,1875	12288	2285	0,0349
0,2188	14336	3139	0,0479
0,2500	16384	4036	0,0616
0,2813	18432	5083	0,0776
0,3125	20480	6518	0,0995
0,3438	22528	7650	0,1167
0,3750	24576	9269	0,1414
0,4063	26624	10757	0,1641
0,4375	28672	12619	0,1926
0,4688	30720	14198	0,2166
0,5000	32768	16372	0,2498
0,5313	34816	18386	0,2806
0,5625	36864	20984	0,3202
0,5938	38912	23024	0,3513
0,6250	40960	25969	0,3963
0,6563	43008	28270	0,4314
0,6875	45056	30982	0,4728
0,7188	47104	33562	0,5121
0,7500	49152	36845	0,5622
0,7813	51200	39958	0,6097
0,8125	53248	43352	0,6615
0,8438	55296	46292	0,7064
0,8750	57344	49896	0,7614
0,9063	59392	53627	0,8183
0,9375	61440	57698	0,8804
0,9688	63488	61370	0,9364

Si representamos gráficamente los datos obtenidos en la Tabla 2-8 y realizamos un ajuste por mínimos cuadrados de los datos representados, obtendremos una función de transferencia que se presenta en la Figura 2-23.

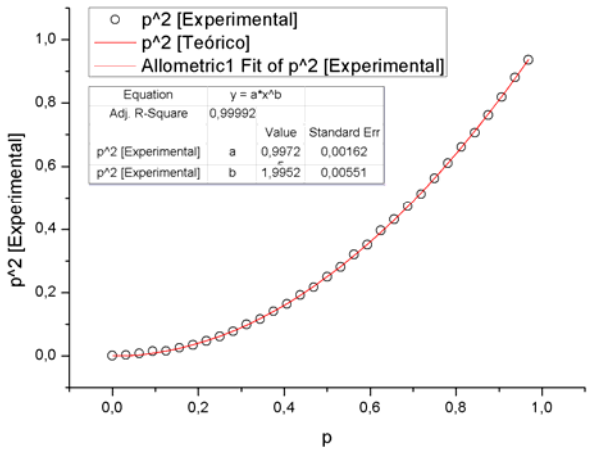

Figura 2-23: Resultados experimentales del bloque cuadrado (UCSL)

Si ahora comparamos la función de transferencia obtenida de la Figura 2-23 (Ecuación [2-24]), con la predicha por la teoría (Ecuación [2-23]) para la presente implementación del bloque podremos apreciar cómo ambas son prácticamente idénticas.

$$\begin{aligned} &\text{Teórico} && \rightarrow OUT_{TEO} = p^2 \\ &\text{Experimental} && \rightarrow OUT_{EXP} = 0{,}9938 \cdot p^2 \text{, con una } R^2 = 0{,}9938 \end{aligned} \qquad \text{Ecuación [2-24]}$$

2.2.3.6 Funciones de recurrencia

La implmentación de las funciones presentadas hasta el momento ha sido prácticamente directa mediante el uso de circuitos puramente combinacionales (sin ningún tipo de realimentación). En el presente apartado se aborda el uso de la realimentación para presentar toda una nueva familia de funciones no-lineales, fácilmente sintetizables y que consumen muy pocos recursos lógicos en su implementación mediante hardware programable (*FPGA o CPLD*).

A continuación se presentan los bloques/funciones más interesantes que se han desarrollado para dar soporte a las diversas aplicaciones que se presentan en capítulos posteriores en las

que se ha hecho uso de la lógica estocástica, como puede ser en el campo de la implementación de redes neuronales y el reconocimiento de patrones.

2.2.3.6.1 La función $(1/(1+p))$

Para el diseño de la función (1/(1+p)) se ha requerido del uso de la teoría de control (más concretamente de la relacionada con los sistemas realimentados) a fin de determinar los bloques y elementos digitales necesarios para la implementación de esta función de recurrencia.

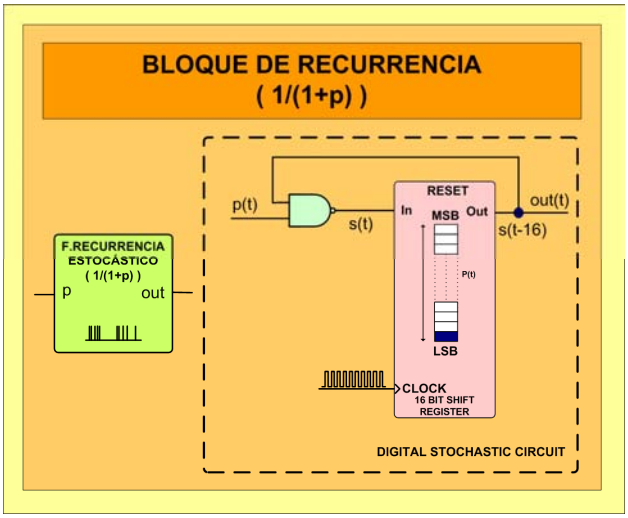

Figura 2-24: Bloque función (1/(1+p)) (UCLS)

La implementación digital de la función (1/(1+p)) se presenta en la Figura 2-24, y se basa en el uso de una puerta **NAND** de 2 entradas. En una de ellas se inyecta la señal pulsante "p" y en la otra la realimentación del circuito (que se corresponde con la señal de salida del circuito "out"). La salida de esta puerta lógica está conectada a un elemento de memoria síncrono (biestable o un registro de desplazamiento) encargado de almacenar el valor de salida de la puerta una serie de ciclos de retardado para descorrelacionar la señal de salida del circuito con respecto a las señales de entrada de la puerta **NAND.** La presente función se implementó inicialmente con sólo un biestable tipo D (tal y como se deriva del estudio teórico realizado), pero en vista de los resultados no totalmente satisfactorios obtenidos en la fase de pruebas derivados de la persistencia de correlación entre la señal de

realimentación y la señal de entrada “p” se terminó sustituyendo dicho biestable por un registro de desplazamientos de 16-bits, que se corresponde con el mismo número de bits que los bloques (B2P de 16-bits) usados para la generación de la señal estocásticas “p” de test.

El comportamiento de este bloque de recurrencia se regirá mediante la Ecuación [2-25] matemática, que se presenta a continuación:

$$\left.\begin{array}{l} s(t) = (1 - p(t) \cdot out(t)) \\ \text{out}(t) = s(t - 16) \end{array}\right\} \rightarrow \text{out} = (1 - p \cdot \text{out})$$

$$Suponemos\_que: s(t-16) \approx s(t) \qquad \text{Ecuación [2-25]}$$

$$\rightarrow \text{out} \cdot (1 + p) = 1 \rightarrow E(out) = \frac{1}{1 + P(p)} = \frac{2^n}{2^n + \frac{P}{2^n}} = \frac{2^{2n}}{2^{2n} + P}$$

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-9. Estas medidas han consistido en ir variando el valor medio de la distribución asociada a la señal pulsante “p” en el intervalo [0, +1].

Tabla 2-9: Medidas de la función (1/(1+p)) (UCSL)

DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	OUT [16bits]	Valor OUT	OUT=(1/(1+P))	ERROR
0	0,0000	65535	1,0000	1,0000	0,0000
4096	0,0625	61647	0,9407	0,9412	0,0005
8192	0,1250	58496	0,8926	0,8889	0,0037
12288	0,1875	55251	0,8431	0,8421	0,0010
16384	0,2500	52320	0,7984	0,8000	0,0016
20480	0,3125	49948	0,7622	0,7619	0,0003
24576	0,3750	47690	0,7277	0,7273	0,0004
28672	0,4375	45701	0,6974	0,6956	0,0017
32768	0,5000	43697	0,6668	0,6667	0,0001
36864	0,5625	41900	0,6394	0,6400	0,0006
40960	0,6250	40232	0,6139	0,6154	0,0015
45056	0,6875	38788	0,5919	0,5926	0,0007
49152	0,7500	37487	0,5720	0,5714	0,0006
53248	0,8125	36115	0,5511	0,5517	0,0006
57344	0,8750	34921	0,5329	0,5333	0,0005
61440	0,9375	33845	0,5164	0,5161	0,0003

Seguidamente procedemos a representar gráficamente (Figura 2-25) por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación realizada.

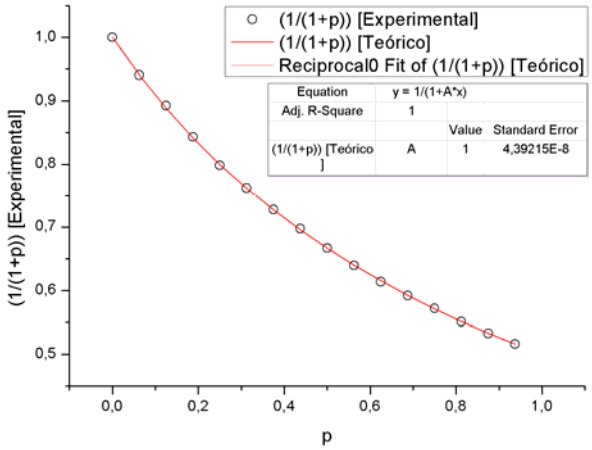

Figura 2-25: Resultados experimentales de la función (1/(1+p)) (UCLS)

Como se puede apreciar la función de transferencia experimental (Figura 2-25) del bloque implementado es idéntica a la descrita por la teoría (Ecuación [2-25]), tal y como se muestra en la Ecuación [2-26]. El valor absoluto del error (Tabla 2-9) entre la señal medida y el valor predicho por la teoría es ínfimo para cada una de las medidas realizadas.

$$\begin{aligned} &\text{Experimental} \quad \rightarrow OUT_{EXP} = 1/(1+1\cdot P), \text{ con una } R^2 = 1 \\ &\text{Teórico} \quad \rightarrow OUT_{TEO} = 1/(1+P) \end{aligned} \qquad \text{Ecuación [2-26]}$$

2.2.3.6.2 La función $(p/(1+p))$

Para la implementación de la función estocástica (p/(1+p)) se procederá de forma similar al modo en el que se ha obtenido la anterior función de recurrencia (1/(1+p)). Se partirá de la teoría de realimentación para la evaluación de la función de recurrencia que proporcione la expresión no-lineal requerida.

La implementación digital para obtener la función (p/(1+p)) se presenta en la Figura 2-26. Ésta consiste en una puerta **AND** de dos entradas, en una de estas se inyecta la señal estocástica "p" y en la otra la realimentación negada (mediante una puerta **NOT**) de la salida del circuito. La salida del circuito es la salida descorrelacionada de la puerta **AND** mediante un elemento de memoria síncrono, más concretamente con un registro de

desplazamiento de 16-bits. El comportamiento que predice la teoría para este bloque recurrente se presenta en la Ecuación [2-27].

$$\left.\begin{array}{l} s(t) = \left(p(t)\cdot(1-out(t))\right) \\ \text{out}(t) = s(t-16) \end{array}\right\} \rightarrow \text{out} = p\cdot(1-\text{out})$$

$$\text{Supondremos que}: s(t-16) \approx s(t) \qquad \text{Ecuación [2-27]}$$

$$\rightarrow \text{out}\cdot(1+p) = p \rightarrow E(out) = \frac{P(p)}{1+P(p)} = \frac{\frac{P}{2^n}}{2^n + \frac{P}{2^n}} = \frac{P}{2^{2n}+P}$$

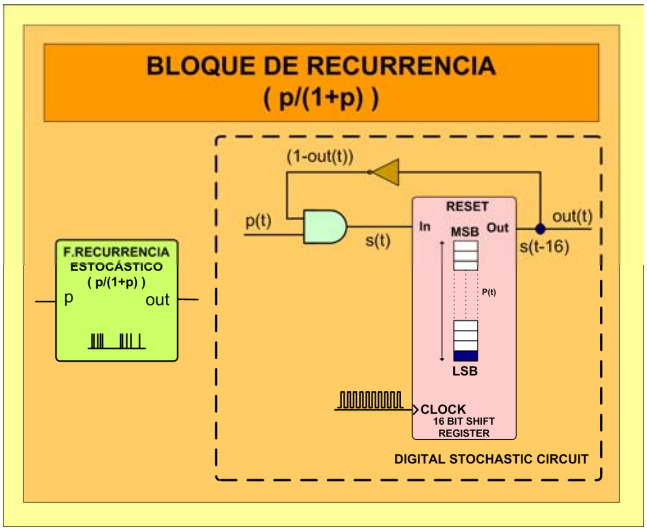

Figura 2-26: Bloque función (p/(1+p)) (UCLS)

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-10. Estas medidas han consistido en ir variando el valor medio de la distribución asociada a la señal pulsante "p" en el intervalo [0, +1].

Tabla 2-10: Medidas de la función (p/(1+p)) (UCLS)

DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	OUT [16bits]	Valor OUT	OUT=(1/(1+P))	ERROR
0	0,0000	0	0,0000	0,0000	0,0000
4096	0,0625	3843	0,0586	0,0588	0,0002
8192	0,1250	7200	0,1099	0,1111	0,0012
12288	0,1875	10255	0,1565	0,1579	0,0014
16384	0,2500	13294	0,2029	0,2000	0,0029
20480	0,3125	15714	0,2398	0,2381	0,0017

24576	0,3750	17930	0,2736	0,2727	0,0009
28672	0,4375	19940	0,3043	0,3044	0,0001
32768	0,5000	21720	0,3314	0,3333	0,0019
36864	0,5625	23624	0,3605	0,3600	0,0005
40960	0,6250	25240	0,3851	0,3846	0,0005
45056	0,6875	26707	0,4075	0,4074	0,0001
49152	0,7500	28193	0,4302	0,4286	0,0016
53248	0,8125	29393	0,4485	0,4483	0,0002
57344	0,8750	30534	0,4659	0,4667	0,0008
61440	0,9375	31714	0,4839	0,4839	0,0000

Seguidamente procedemos a representar gráficamente (Figura 2-26) por una parte los resultados experimentales obtenidos (símbolos), y por otra los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación realizada.

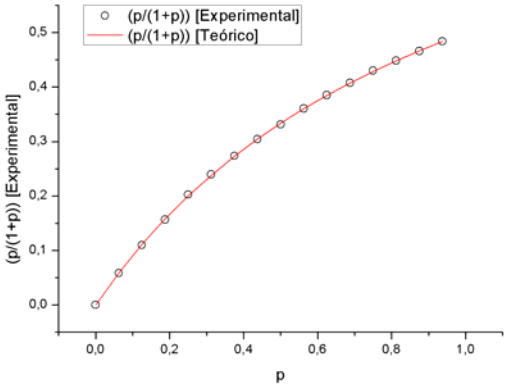

Figura 2-27: Resultados experimentales de la función (p/(1+p)) (UCLS)

Como se puede apreciar en la Figura 2-27 la función de recurrencia implementada sigue el mismo comportamiento que el descrito por la teoría. Además, si se revisa la última columna de la Tabla 2-10 (ERROR) se apreciará cómo el valor absoluto del error entre la señal medida y el valor predicho por la teoría es ínfimo para cada una de las medidas realizadas.

2.2.3.6.3 La función $(p/(1+p\cdot q))$

Para la implementación de la función estocástica no-lineal (p/(1+p·q)) se procederá de forma similar a las funciones anteriores. La implementación digital realizada a partir de una función de recurrencia se presenta en la Figura 2-28.

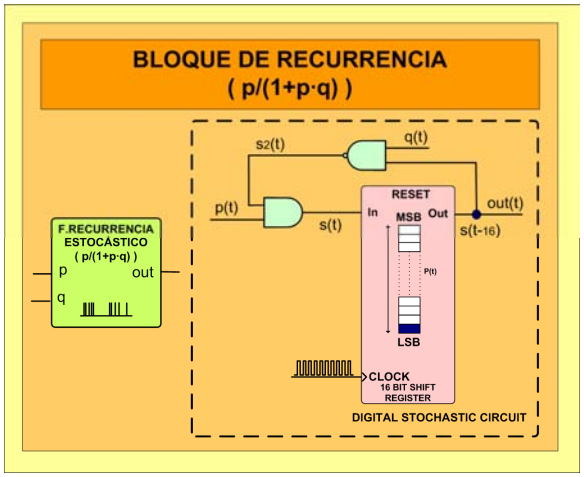

Figura 2-28: Bloque función (p/(1+p·q)) (UCLS)

El bloque (Figura 2-28) se compone de una puerta **AND** de dos entradas, en una de ellas se inyecta la señal pulsante "p" y en la otra la realimentación del bloque (señal "s2"). Que se obtiene mediante una puerta **NAND** que opera la salida del bloque con la segunda señal estocástica "q". La salida del bloque es la señal descorrelacionada "s(t-16)" resultante de la anteriormente descrita puerta **AND** "s(t)", después de discurrir a través de un elemento de memoria síncrono, más concretamente con un registro de desplazamiento de 16-bits. El comportamiento predicho por la teoría para este bloque es el descrito en la Ecuación [2-28].

$$\left.\begin{array}{l} s(t) = p(t)\cdot s2(t) \\ s2(t) = (1 - out(t)\cdot q(t)) \\ \text{out}(t) = s(t-16) \end{array}\right\} \rightarrow \text{out} = p\cdot(1-\text{out}(t)\cdot q(t))$$

$$\text{Supondremos que}: s(t-16) \approx s(t)$$

$$\rightarrow \text{out}\cdot(1+\text{p}\cdot\text{q}) = p \rightarrow E(out) = \frac{P(p)}{1+P(p)\cdot P(q)} = \frac{\frac{P}{2^n}}{2^n + \frac{P\cdot Q}{2^{2n}}} = \frac{2^n \cdot P}{2^{3n} + P\cdot Q}$$

Ecuación [2-28]

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-11. Estas medidas han consistido en fijar el valor medio de la distribución de la señal pulsante "q=0,5" e ir variando el valor medio de la distribución asociada a la señal pulsante "p" en [0,+1].

Tabla 2-11: Medidas de la función (p/(1+p·q)) (UCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	Q [16bits]	Valor Q	OUT [16bits]	Valor OUT	OUT=P/(1+P·Q)	ERROR
0	0,0000	32768	0,5000	0	0,0000	0,0000	0,0000
4096	0,0625	32768	0,5000	3819	0,0583	0,0606	0,0023
8192	0,1250	32768	0,5000	7766	0,1185	0,1176	0,0009
12288	0,1875	32768	0,5000	11122	0,1697	0,1714	0,0017
16384	0,2500	32768	0,5000	14608	0,2229	0,2222	0,0007
20480	0,3125	32768	0,5000	17757	0,2710	0,2703	0,0007
24576	0,3750	32768	0,5000	20955	0,3198	0,3158	0,0040
28672	0,4375	32768	0,5000	23634	0,3606	0,3590	0,0017
32768	0,5000	32768	0,5000	26227	0,4002	0,4000	0,0002
36864	0,5625	32768	0,5000	28717	0,4382	0,4390	0,0008
40960	0,6250	32768	0,5000	31357	0,4785	0,4762	0,0023
45056	0,6875	32768	0,5000	33571	0,5123	0,5116	0,0006
49152	0,7500	32768	0,5000	35759	0,5456	0,5455	0,0002
53248	0,8125	32768	0,5000	37883	0,5781	0,5778	0,0003
57344	0,8750	32768	0,5000	39916	0,6091	0,6087	0,0004
61440	0,9375	32768	0,5000	41895	0,6393	0,6383	0,0010

Seguidamente procedemos a representar gráficamente (Figura 2-29) por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación realizada.

Como se puede apreciar en la Figura 2-29 la función de recurrencia implementada sigue el mismo comportamiento que el descrito por la teoría. Si se revisa la última columna de la Tabla 2-11 se comprobará cómo el valor absoluto del error entre la señal medida y el valor predicho por la teoría es ínfimo para cada una de las medidas realizadas.

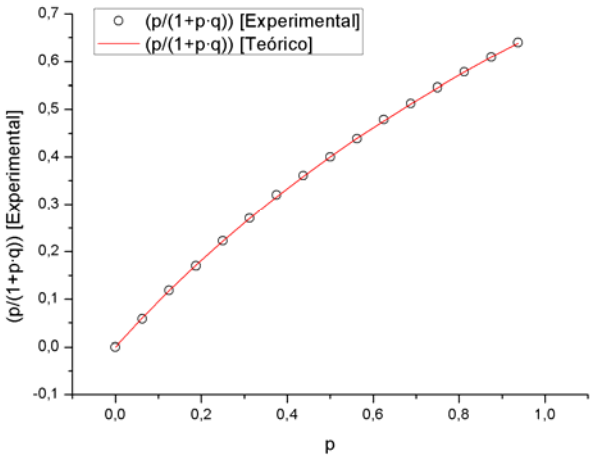

Figura 2-29: Resultados experimentales de la función (p/(1+p·q)) (UCSL)

2.2.3.6.4 La función $(1/(1+p{\cdot}q))$

De forma similar a las anteriores funciones de recurrencia implementadas se ha procedido a la implementación de la función (1/(1+p·q)). En la Figura 2-30 se presenta su implementación digital.

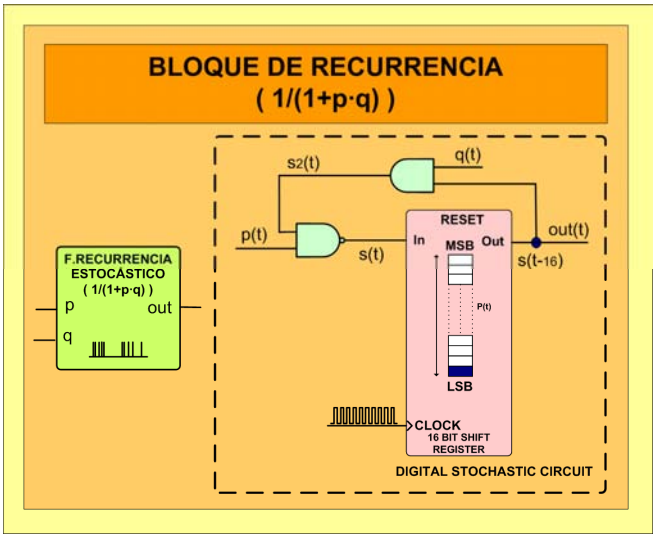

Figura 2-30: Bloque función (1/(1+p·q)) (UCLS)

El bloque desarrollado se compone de una puerta N**AND** de dos entradas, en una de ellas se inyecta la señal pulsante "p" y en la otra la señal de realimentación (señal "s2(t)"); que proviene de operar la salida del bloque mediante una puerta **AND** con una segunda señal estocástica "q". La salida del bloque es la salida descorrelacionada "s(t-16)" de la puerta N**AND** "s(t)" mediante un elemento de memoria síncrono, más concretamente con un registro de desplazamiento de 16-bits.

El comportamiento predicho por la teoría para este bloque es el descrito por la Ecuación [2-29].

$$\left.\begin{array}{l} s(t) = (1 - p(t) \cdot s2(t)) \\ s2(t) = q(t) \cdot out(t) \\ \text{out}(t) = s(t-1) \end{array}\right\} \rightarrow \text{out} = (1 - p \cdot q \cdot out)$$

$$\text{Supondremos que}: s(t-16) \approx s(t)$$

$$\rightarrow \text{out} \cdot (1 + p \cdot q) = 1 \rightarrow E(out) = \frac{1}{1 + P(p) \cdot P(q)} = \frac{2^n}{2^n + \frac{P \cdot Q}{2^{2n}}} = \frac{2^{3n}}{2^{3n} + P \cdot Q}$$

Ecuación [2-29]

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-12, que han consistido en fijar el valor medio de la distribución de la señal pulsante "q=0,5" y variar el valor medio de la distribución asociada a la señal pulsante "p"en el intervalo [0, +0.9375].

Tabla 2-12: Medidas de la función (1/(1+p·q)) (UCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	Q [16bits]	Valor Q	OUT [16bits]	Valor OUT	OUT=1/(1+P·Q)	ERROR
0	0,0000	32768	0,5000	65535	1,0000	1,0000	0,0000
4096	0,0625	32768	0,5000	63616	0,9707	0,9697	0,0010
8192	0,1250	32768	0,5000	61825	0,9434	0,9412	0,0022
12288	0,1875	32768	0,5000	59821	0,9128	0,9143	0,0015
16384	0,2500	32768	0,5000	58202	0,8881	0,8889	0,0008
20480	0,3125	32768	0,5000	56595	0,8636	0,8649	0,0013
24576	0,3750	32768	0,5000	55189	0,8421	0,8421	0,0000
28672	0,4375	32768	0,5000	53784	0,8207	0,8205	0,0002
32768	0,5000	32768	0,5000	52500	0,8011	0,8000	0,0011
36864	0,5625	32768	0,5000	51051	0,7790	0,7805	0,0015
40960	0,6250	32768	0,5000	50020	0,7633	0,7619	0,0014
45056	0,6875	32768	0,5000	48723	0,7435	0,7442	0,0007
49152	0,7500	32768	0,5000	47754	0,7287	0,7273	0,0014
53248	0,8125	32768	0,5000	46564	0,7105	0,7111	0,0006

57344	0,8750	32768	0,5000	45649	0,6966	0,6956	0,0009
61440	0,9375	32768	0,5000	44612	0,6807	0,6808	0,0001

Seguidamente procederemos a representar gráficamente (Figura 2-31) por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación digital realizada.

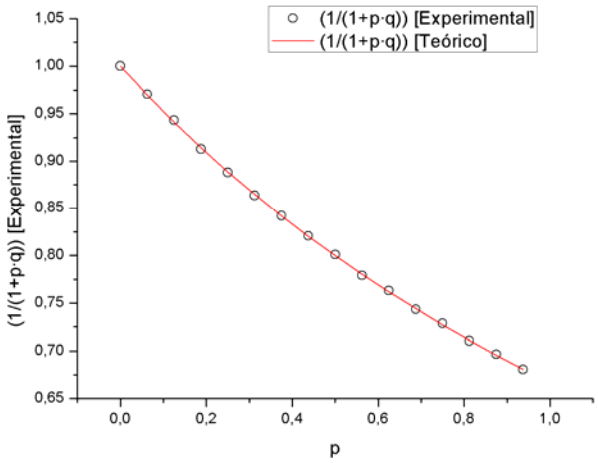

Figura 2-31: Resultados experimentales de la función (1/(1+p·q)) (UCSL)

Como bien puede apreciarse en la Figura 2-31 la función de recurrencia implementada sigue el mismo comportamiento que el descrito por la teoría. Si revisamos la última columna de la Tabla 2-12 se aprecia cómo el valor absoluto del error entre la señal medida y el valor predicho por la teoría para cada una de las medidas realizadas es ínfimo.

2.2.3.7 La operación inversa ($f^{-1}(p)$)

El bloque que a continuación se describe constituye un instrumento genérico para la obtención de la operación inversa $f^{-1}(p)$ a partir de una función o bloque estocástico conocido $f(p)$. Mediante la estructura que se describe a continuación, es posible determinar la inversa de cualquier función que tenga su representación en el espacio definido para la codificación clasica (UCSL) [0, +1].

La implementación digital del bloque *operación inversa* $f^{-1}(p)$ se presenta en la Figura 2-32. El funcionamiento de este bloque se basa en un circuito de realimentación que evalúa el error entre la señal esperada "p" y la señal "p evaluada" obtenida de inyectar la señal obtenida de la función inversa $f^{-1}(p)$ a través del bloque $f(p)$. El circuito se compone de un contador Up/Down conectado a un bloque (*P2B*) encargado de aproximar de forma iterativa la señal pulsada $h = f^{-1}(p)$. La salida del bloque *B2P(N)* se conecta a la entrada del bloque de función $f(p)$ que se desee invertir, para posteriormente comparar mediante una puerta **XOR** dicha salida con la entrada de la señal estocástica "p" al bloque. La salida de la puerta XOR proporciona una señal pulsante indicativa del error producido entre la señal obtenida y la deseada que se realimenta al contador binario Up/Down. El sistema converge en un máximo de 2^N ciclos de reloj (donde N se corresponde con número de bits del contador) con la señal esperada "p".

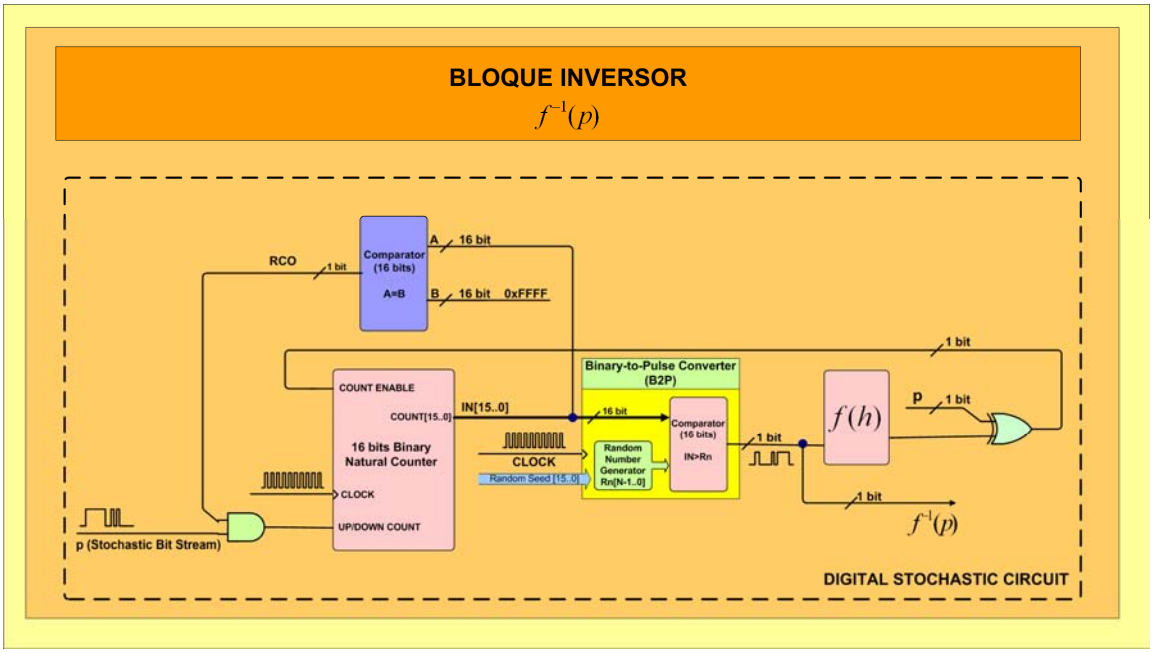

Figura 2-32: Bloque operación inversa ($f^{-1}(p)$) (UCSL)

Más detalladamente el funcionamiento de los diferentes elementos es el siguiente; la puerta **AND** a la entrada del contador se corresponde con un mecanismo de protección que conjuntamente con la puerta **NOT** conectada a la salida del comparador se encargan de

determinar si hemos llegado al último valor del contador. Esta salida hace las funciones de señal de *Ripple-Carry-Out (RCO)*, para impedir que el contador de módulo N se desborde. Cuando las señal de *Ripple-Carry-Out (RCO)* se activa (a nivel alto) el inversor la convierte a nivel bajo, forzando así que independientemente de lo que valga la señal "p" en ese instante de tiempo, el contador decremente su valor. Así se evitan fluctuaciones bruscas en el valor del contador que provocarían un comportamiento inestable del bloque.

El bloque contador Up/Down sólo modificará su valor en ±1, si la señal (*EN=activación*) que proviene de la salida de la puerta lógica **XOR** es 1 (que determina el error entre la señal "p" estimada por el bloque de función $f(h) = p$ y la inyectada desde el exterior "p").

Hay que tener muy presente que para que este circuito opere se requiere de la incorporación de una función $f(h)$, que debe ser invertible. Este circuito ha sido utilizado de forma satisfactoria en la implementación de la función división de dos señales pulsantes y para la obtención de la función raíz cuadrada. No obstante con este circuito nos ha sido imposible obtener la función logaritmo neperiano *Ln(p)* a partir de la función exponencial estocástica descrita más adelante en el presente capítulo.

A continuación se describen en detalle las diversas implementaciones realizadas a partir de este bloque así como los resultados experimentales obtenidos para cada una de ellas.

2.2.3.7.1 La operación división

La implementación de la operación división entre dos señales estocásticas "p" y "q" se basa en el uso del bloque operación inversa anteriormente descrito, al cual se le incorpora como función $f(h)$ un bloque multiplicador de dos señales estocásticas "p" y "q" con la finalidad de invertirlo y así obtener una señal pulsante con un valor medio proporcional a "p/q".

El circuito digital desarrollado para implementar dicha función el que se presenta a continuación en la Figura 2-33.

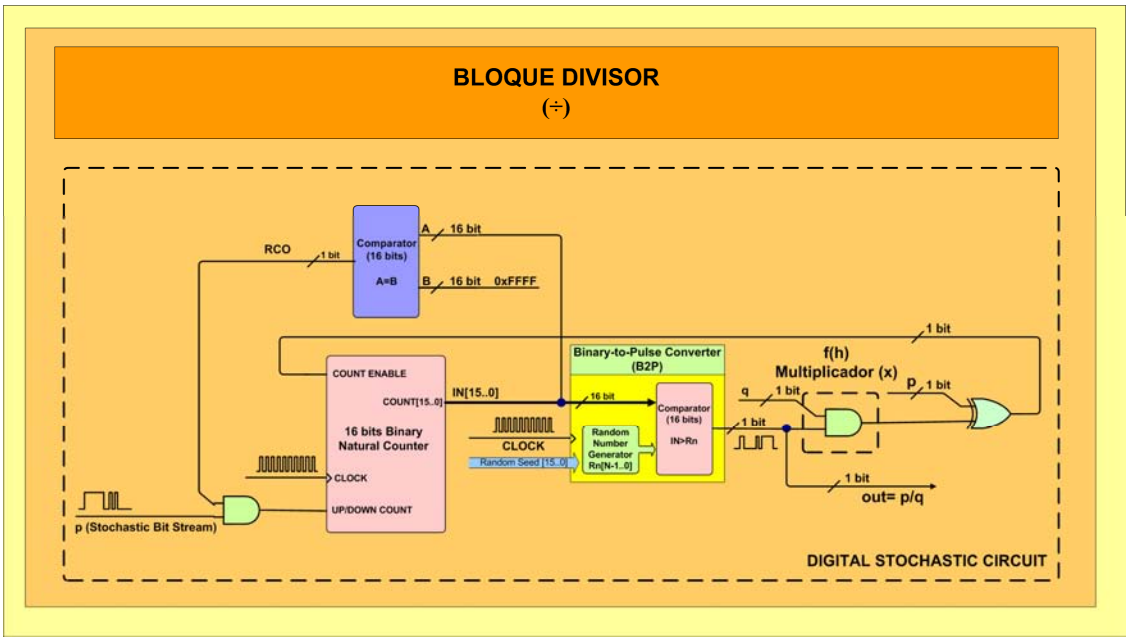

Figura 2-33: Bloque divisor (p/q) (UCSL)

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-13. Éstas han consistido en fijar el valor medio de la distribución de la señal pulsante “p=0,5” a la vez que se ha variado el valor medio de la distribución asociada a la señal pulsante “q” en [0,+1].

Tabla 2-13: Medidas de la función división (UCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	Q [16bits]	Valor Q	OUT [16bits]	Valor OUT	OUT=P/Q	ERROR
32768	0,5000	0	0,0000	65535	1,0000	+∞	+∞
32768	0,5000	4096	0,0625	65535	1,0000	8,0000	7,0000
32768	0,5000	8192	0,1250	65535	1,0000	4,0000	3,0000
32768	0,5000	12288	0,1875	65535	1,0000	2,6667	1,6667
32768	0,5000	16384	0,2500	65535	1,0000	2,0000	1,0000
32768	0,5000	20480	0,3125	65535	1,0000	1,6000	0,6000
32768	0,5000	24576	0,3750	65532	1,0000	1,3333	0,3334
32768	0,5000	28672	0,4375	65533	1,0000	1,1429	0,1429
32768	0,5000	32768	0,5000	65159	0,9943	1,0000	0,0057
32768	0,5000	36864	0,5625	58467	0,8921	0,8889	0,0033
32768	0,5000	40960	0,6250	52452	0,8004	0,8000	0,0004
32768	0,5000	45056	0,6875	47906	0,7310	0,7273	0,0037
32768	0,5000	49152	0,7500	43380	0,6619	0,6667	0,0047

32768	0,5000	53248	0,8125	40488	0,6178	0,6154	0,0024
32768	0,5000	57344	0,8750	37142	0,5668	0,5714	0,0047
32768	0,5000	61440	0,9375	34948	0,5333	0,5333	0,0001

Procederemos a representar gráficamente en la Figura 2-34 por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación digital realizada.

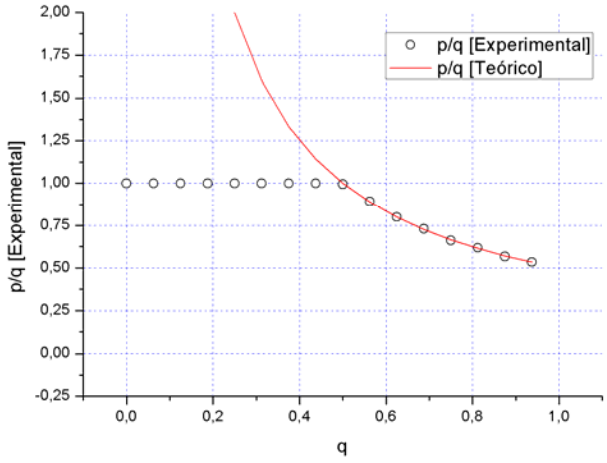

Figura 2-34: Resultados experimentales de la operación división (UCSL)

Como puede apreciarse en la Figura 2-34 la función división estocástica implementada sigue el mismo comportamiento que el descrito por la teoría, en el rango de representación de la lógica estocástica. Si se revisa la última columna de la (Tabla 2-13), se podrá apreciar cómo el valor absoluto del error entre la señal medida y el valor predicho por la teoría es ínfimo para las diversas medidas realizadas. También se puede observar unos resultados análogos en la Figura 2-11.

2.2.3.7.2 La operación raíz cuadrada

Al igual que para la operación anterior, la implementación del bloque raíz cuadrada de una señal estocástica "p" de entrada se basa en el uso del bloque función inversa anteriormente descrito, al cual se le incorpora como función $f(h)$ un bloque potencia 2 con objeto de invertirlo y así obtener una señal pulsante con un valor medio proporcional a "$\sqrt{p}$". El circuito digital desarrollado que implementa dicha función se presenta en la Figura 2-35.

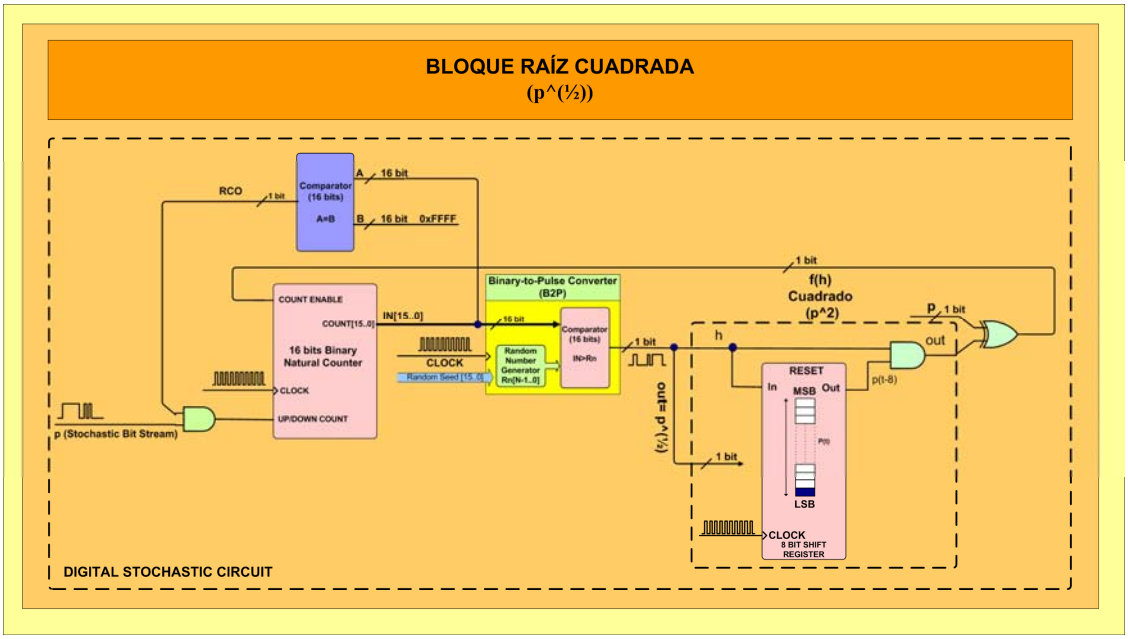

Figura 2-35: Bloque raíz cuadrada (UCSL)

Para comprobar el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-14. Éstas han consistido en ir variando el valor medio de la distribución asociada a la señal pulsante "p" en el intervalo [0, +1].

Tabla 2-14: Medidas de la operación raíz cuadrada (UCSL)

DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	OUT [16bits]	Valor OUT	OUT=(P^(1/2))	ERROR
0	0,0000	7	0,0001	0,0000	0,0001
4096	0,0625	16332	0,2492	0,2500	0,0008
8192	0,1250	22047	0,3364	0,3536	0,0171
12288	0,1875	28035	0,4278	0,4330	0,0052
16384	0,2500	32829	0,5009	0,5000	0,0009
20480	0,3125	36679	0,5597	0,5590	0,0007
24576	0,3750	39365	0,6007	0,6124	0,0117
28672	0,4375	42809	0,6532	0,6614	0,0082
32768	0,5000	45967	0,7014	0,7071	0,0057
36864	0,5625	49184	0,7505	0,7500	0,0005
40960	0,6250	51247	0,7820	0,7906	0,0086
45056	0,6875	54332	0,8291	0,8292	0,0001
49152	0,7500	56506	0,8622	0,8660	0,0038
53248	0,8125	59054	0,9011	0,9014	0,0003
57344	0,8750	61134	0,9328	0,9354	0,0026
61440	0,9375	63422	0,9678	0,9683	0,0005

Procederemos a representar gráficamente (Figura 2-36) por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a fin de poder evaluar la bondad de la implementación realizada.

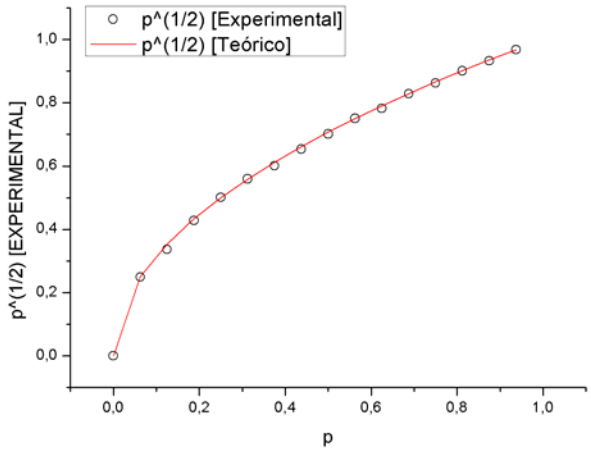

Figura 2-36: Resultados experimentales la operación raíz cuadrada (UCSL)

Como puede apreciarse en la Figura 2-36 la función raíz cuadrada implementada sigue el mismo comportamiento que el descrito por la teoría, en vistas que dicha función se encuentra completamente definida en el rango de representación de la lógica estocástica. Si revisamos la última columna de la Tabla 2-14, se puede apreciar cómo el valor absoluto del error entre la señal medida y el valor predicho por la teoría es muy pequeño para la mayoría de las medidas realizadas.

2.2.3.8 La función exponencial negativa ($e^{-k \cdot p}$)

Antes de iniciar la descripción del fundamento teórico sobre el cual se sustenta la implementación estocástica realizada de la función exponencial negativa $e^{-k \cdot p}$, se hace necesario describir alguna de las propiedades matemáticas más significativas de la función exponencial de las cuales nos serviremos a lo largo de la fundamentación teórica de la implementación realizada.

La función exponencial en matemáticas se corresponde con la función e^{x}, siendo “e” el número de Euler o constante de Napier, que se corresponde a un número irracional de valor

aproximado 2.71828...; siendo su dominio de definición el conjunto de los números reales $\Re \in (-\infty,+\infty)$, con la particularidad que su derivada es la misma función. El número "e" es la base de los logaritmos naturales y corresponde a la función inversa del logaritmo natural "*ln(x)*".

La función exponencial "e^x" puede ser descrita de diferentes formas equivalentes entre si. Puede ser una serie infinita de potencias (Ecuación [2-30]) o un límite de una sucesión de potencias (Ecuación [2-31]).

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... \qquad \text{Ecuación [2-30]}$$

$$e^x = \lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n . \qquad \text{Ecuación [2-31]}$$

Las propiedades fundamentales que satisface la función exponencial (para cualquier base independientemente que sea "e") se describen en la Ecuación [2-32]:

$$\begin{cases} e^{x+y} = e^x \cdot e^y . \\ e^{x-y} = e^x \big/ e^y \\ e^{-x} = 1 \big/ e^x \\ e^0 = 1 \\ \dfrac{d}{dx} e^x = e^x \\ \lim\limits_{x\to-\infty} e^x = 0 \\ \lim\limits_{x\to+\infty} e^x = \infty \end{cases} \qquad \text{Ecuación [2-32]}$$

En el marco de la presente tesis la importancia de la función exponencial radica en el hecho que es el elemento esencial para poder implementar la función Gaussiana; de la cual nos serviremos para definir funciones distribución de probabilidad de una categoría a partir de las "n" muestras de referencia mediante una distribución de **Parzen** a fin de poder implementar metodologías de reconocimiento de patrones basadas en lógica estocástica.

Antes de proseguir con la descripción de la fundamentación teórica sobre la que se sustenta la implementación estocástica de la función exponencial negativa, es conveniente poner de manifiesto que en la literatura existen otras implementaciones estocásticas de dicha función como puede ser la de *C.L Janer et al.* [2-10] en la cual se aproximaba esta función $e^{-1.040 \cdot p}$ mediante un circuito estocástico que implementaba el desarrollo de Taylor (Ecuación [2-24]) de tercer orden de dicha función; para luego hacer uso de dicha función en la implementación de filtros digitales de respuesta finita al impulso (FIR). El error obtenido en su estimación de la función exponencial negativa es inferior al 7% [2-20]. La limitación principal de esta implementación radica en el hecho que para función exponencial diferente que se quiera implementar $e^{-k \cdot p}$ se requerirá de la implementación de un nuevo circuito estocástico. Esta implementación estocástica a su vez no es apta para la obtención de una función exponencial que requiera de un parámetro "k>1".

Para ello en nuestros diseños hemos buscado una fundamentación teórica que nos permitiera implementar la función exponencial negativa real y no una aproximación de esta como hallamos en la literatura. Para ello partimos del hecho que disponemos de un bloque o función cuya salida se rige por una distribución binomial (Ecuación [2-33])), que se caracteriza por ser una distribución de probabilidad discreta [2-22]. El número de éxitos de una secuencia de "n" ensayos de Bernoulli, independientes entre sí y con una probabilidad fija "p" de ocurrencia del éxito entre los diversos ensayos y viene descrito por:

$$P(X = k) = f(k) = \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} \quad \forall \mathrm{k} = 0,1,...,\mathrm{n} \qquad \text{Ecuación [2-33]}$$

Caracterizándose cada ensayo de Bernoulli por ser dicotómico, es decir que únicamente puede tomar dos posibles valores. Al primer estado se le llama *éxito* con una probabilidad de ocurrencia "p" y al otro *fracaso* con una probabilidad de ocurrencia (1-p). Una distribución binomial es el resultado de repetir "n" veces el experimento de Bernoulli de forma independiente con objeto de evaluar la probabilidad de obtener un determinado número de éxitos "x". El esperado o valor medio vendrá descrito por la Ecuación [2-34]:

$$E(X) = n \cdot p \qquad \text{Ecuación [2-34]}$$

La distribución binomial en el caso límite [2-22, 2-23] en el que la probabilidad de "p" sea muy pequeña (prácticamente cero) y el número de ensayos "n" realizado sea relativamente grande (a la práctica $n \geq 30$), tenderá a comportarse como una distribución de Poisson con un valor medio $\lambda = n \cdot p$, tal y como se demuestra en la ecuación [2-35]:

$$\lim_{\substack{n\to\infty \\ p\to 0}} P(X=k) = \lim_{\substack{n\to\infty \\ p\to 0}} f(k) = \lim_{\substack{n\to\infty \\ p\to 0}} \binom{n}{k} \cdot p^k \cdot (1-p)^{n-k} = \begin{cases} p = \lambda/n \\ \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} \binom{n}{k} \cdot (\lambda/n)^k \cdot (1-\lambda/n)^{n-k} = a \end{cases}$$

$$a = \lim_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n!}{k!\cdot(n-k)!} \cdot (\lambda/n)^k \cdot (1-\lambda/n)^{n-k} = \frac{\lambda^k}{k!} \lim_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n!}{n^k \cdot (n-k)! \cdot (1-\lambda/n)^k} \cdot (1-\lambda/n)^n = b$$

$$b = \frac{\lambda^k}{k!} \lim_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n!}{(n-k)! \cdot (1-\lambda)^k} \cdot (1-\lambda/n)^n = \frac{\lambda^k}{k!} \lim_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n!}{(n-k)! \cdot (1-\lambda)^k} \cdot \lim_{\substack{n\to\infty \\ \lambda/n\to 0}} (1-\lambda/n)^n = c$$

$$= \begin{cases} e^x = \lim\limits_{x\to\infty} (1+1/x)^x \\ \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} (1-\lambda/n)^n = \lim\limits_{x\to\infty} (1+1/x)^{x(-\lambda)} = e^{-\lambda} \\ \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{(n-\lambda)^k} = \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n^k}{n^k} = 1 \\ c = \frac{\lambda^k}{k!} \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} \frac{n\cdot(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{(n-\lambda)^k} \cdot \lim\limits_{\substack{n\to\infty \\ \lambda/n\to 0}} (1-\lambda/n)^n = \frac{\lambda^k}{k!} \cdot 1 \cdot e^{-\lambda} = \frac{e^{-\lambda}\cdot\lambda^k}{k!} \end{cases}$$

Ecuación [2-35]

Siendo la distribución de Poisson una distribución de probabilidad discreta que expresa, a partir de una frecuencia de ocurrencia media "λ", la probabilidad que se den un determinado número entero de eventos "k" en un cierto intervalo de tiempo. La función densidad de probabilidad característica de la distribución de Poisson se presenta en la Ecuación [2-36]

$$P(X=k) = f(k;\lambda) = \frac{e^{-\lambda}\cdot\lambda^k}{k!}, \text{k} = 0,1,2,3,\ldots., \lambda > 0 \qquad \text{Ecuación [2-36]}$$

Donde la media de la distribución vendrá dada por la Ecuación [2-37]:

$$E[X]=\sum_{i=0}^{\infty}\frac{e^{-\lambda}\cdot\lambda^{i}}{i!}\cdot i=\lambda\cdot e^{-\lambda}\cdot\sum_{i=1}^{\infty}\frac{\lambda^{i-1}}{(i-1)!}=\begin{cases} j=i-1 \\ e^{\lambda}=\sum_{j=0}^{\infty}\frac{\lambda^{i}}{j!} \\ =\lambda\cdot e^{-\lambda}\cdot\sum_{j=0}^{\infty}\frac{\lambda^{i}}{j!}=\lambda\cdot e^{-\lambda}\cdot e^{+\lambda}=\lambda \end{cases}$$

Ecuación [2-37]

Siendo la expresión obtenida para la varianza de la distribución la que se presenta en la Ecuación [2-38]:

$$Var[X]=E[X^{2}]-(E[X])^{2}=(\lambda^{2}+\lambda)-(\lambda)^{2}=\lambda$$

$$E[X^{2}]=\sum_{i=0}^{\infty}\frac{e^{-\lambda}\cdot\lambda^{i}}{i!}\cdot i^{2}=\lambda\cdot e^{-\lambda}\cdot\sum_{i=1}^{\infty}\frac{i\cdot\lambda^{i-1}}{(i-1)!}=\begin{cases} j=i-1 \\ e^{\lambda}=\sum_{j=0}^{\infty}\frac{\lambda^{i}}{j!} \\ =\lambda\cdot e^{-\lambda}\cdot\sum_{j=0}^{\infty}\frac{(j+1)\cdot\lambda^{i}}{j!}=a \end{cases}$$

$$a=\lambda\cdot\left(e^{-\lambda}\cdot\sum_{j=0}^{\infty}\frac{j\cdot\lambda^{i}}{j!}+e^{-\lambda}\cdot\sum_{j=0}^{\infty}\frac{\lambda^{i}}{j!}\right)=\lambda\cdot(\lambda+1)=\lambda^{2}+\lambda$$

$$E[X]=\lambda$$

Ecuación [2-38]

Para la implementación digital del bloque exponencial negativa e^{-x} (Figura 2-37) se hará uso de las propiedades estadísticas anteriormente descritas. El bloque fundamental sobre el que se sustenta la implementación es un bloque *P2B(N)* de 16-bits cuya salida se rige por una distribución puramente binomial, como ya se ha descrito en el apartado 2.1.1.1. La salida del bloque *P2B(N)* se convierte en una distribución de Poisson (Ecuación [2-35]) si el valor medio de la distribución de la señal pulsante de entrada "p" es cercano a cero en todo momento y el número de muestras "$n \geq 30$" es suficientemente grande. La primera

condición se consigue multiplicando mediante una puerta **AND** la señal de entrada "p" por una constante de ajuste "*Constant_ajust[15...0]*" que se encarga de asegurar que para todo el espectro de valores que pueda tomar "p" [0, +1] sea siempre cercana a cero. Además para que la aproximación sea válida el número de ensayos "n" debe ser lo suficientemente grande lo que en la práctica significará que $n \geq 30$ [2-24]. El número de ensayos "n" nunca debe ser mayor al número de ciclos de reloj 2^n del período de evaluación genérico de los bloques estocásticos ($T_{EVAL} = 2^n \cdot T_{Clock} = 2^{16} \cdot T_{Clock}$), lo que se consigue fijando el período de evaluación del bloque P2B en cuestión de forma que se rija por la Ecuación [2-39]:

$$\begin{cases} T_{B2P(N)} = Evaluation\_time \cdot T_{Clock} \\ T_{Eval} = 2^{16} \cdot T_{Clock} \\ n \geq 30 \end{cases} \rightarrow \begin{cases} n = Evaluation\_time \geq 30 \\ num\_pul\_\text{int}\_sist = \dfrac{T_{Eval}}{T_{B2P(N)}} = \dfrac{2^{16} \cdot T_{Clock}}{n \cdot T_{Clock}} \geq 1 \end{cases}$$

Ecuación [2-39]

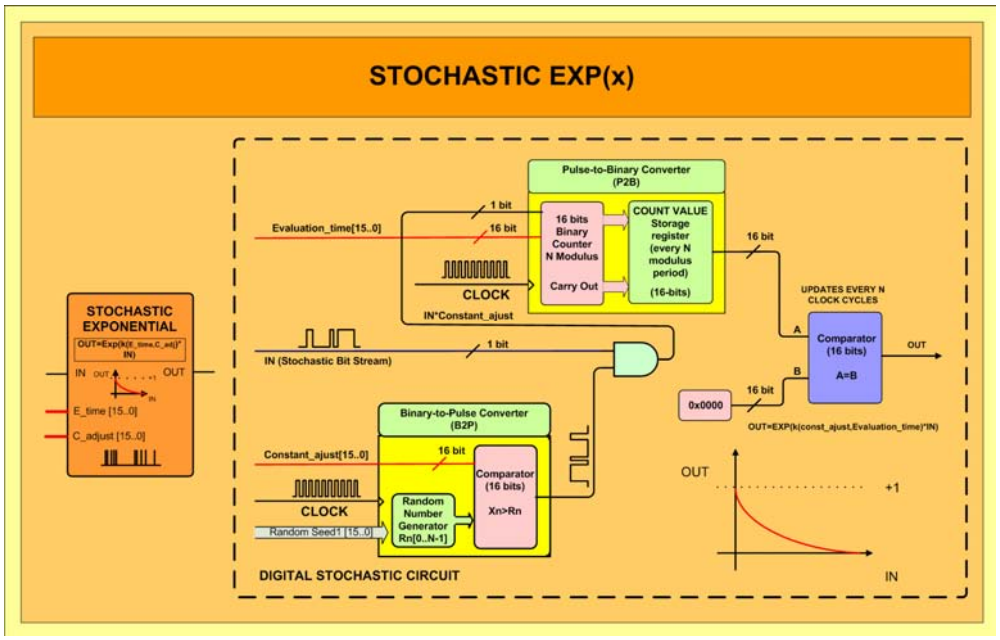

Figura 2-37: Bloque exponencial negativa (UCSL)

Una vez fijadas las anteriores condiciones la salida del bloque P2B(N) pasará a representar la probabilidad de obtener "j" pulsos a la salida del contador durante un intervalo de tiempo ($T_{B2P(N)} = Evaluation\_time \cdot T_{Clock}$), dada una probabilidad media de conmutación de la señal

proporcional a ($\lambda = \text{Constant\_ajust}\cdot p\cdot n$), que se ajusta a la distribución de Poisson descrita en la expresión (Ecuación [2-40]):

$$\begin{cases} \lambda = \text{Constant\_ajust}\cdot p\cdot n = \text{Constant\_ajust}\cdot p\cdot \text{Evaluation\_time} \\ n = \text{Evaluation\_time} \\ P(X=j) = f(j;\lambda) = \dfrac{e^{-\lambda}\cdot\lambda^{j}}{j!} = \dfrac{e^{-(\text{Constant\_ajust}\cdot p\cdot n)}\cdot(\text{Constant\_ajust}\cdot p\cdot n)^{j}}{j!},\ j = 0,1,2,3,....,\ \lambda > 0 \end{cases}$$

Ecuación [2-40]

No obstante, el objetivo es obtener a la salida una función exponencial negativa no una función proporcional a ésta. De esta forma tendremos el caso particular en el que "j=0" (probabilidad de no contabilizar ningún pulso durante "n" ciclos de reloj) con lo que la Ecuación [2-40] se reduce a la Ecuación [2-41].

$$\begin{cases} P(X=j) = f(j;\lambda) = \dfrac{e^{-\lambda}\cdot\lambda^{0}}{0!} = e^{-(Cons\tan t\_ajust\cdot p\cdot n)} = e^{-(Cons\tan t\_ajust\cdot Evaluation\_time\cdot p)},\ j = 0,1,2,..,\ \lambda > 0 \\ \text{Constant\_ajust} = \dfrac{Dato\_n-bits}{2^{n}} \end{cases}$$

Ecuación [2-41]

Esto se consigue comparando la palabra de salida (16-bits) del bloque *B2P(N)* con el "0" en la codificación estocástica clásica"0x0000". De esta forma se consigue una salida igual a '1' si son iguales y '0' en el resto de casos. Dicha salida se mantendrá fija durante varios ciclos de reloj "*Evaluation_time*". Dicho comparador se encarga de convertir la magnitud binaria de salida (16-bits) del bloque P2B(N) nuevamente en una nueva señal pulsante con una probabilidad de activación igual a "P(X=j)". Ya que la magnitud de salida "X=P2B(N)" se compara con una constante de 16-bits fijada a una valor "j".

A fin de comprobar el correcto funcionamiento de la función "e^{-x}" implementada se ha procedido a realizar una serie de medidas experimentales. Las medidas realizadas (Tabla 2-15) ha consistido en ir variando el valor medio de la distribución asociada a la señal pulsante "p" entre [0, +1] mientras los parámetros del bloque han sido prefijados a "*Evaluation_time=600*" y "*Constant_adjust*$= 655/2^{16} = 0{,}01$".

Tabla 2-15: Medidas de la función $e^{-k \cdot x}$ (UCSL)					
DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	OUT [18bits]	Valor OUT	OUT=	ERROR
0	0,0000	262144	1,0000	1,0000	0,0000
4096	0,0625	205489	0,7839	0,6874	0,0965
8192	0,1250	157224	0,5998	0,4726	0,1272
12288	0,1875	123115	0,4696	0,3248	0,1448
16384	0,2500	92554	0,3531	0,2233	0,1298
20480	0,3125	73923	0,2820	0,1535	0,1285
24576	0,3750	54090	0,2063	0,1055	0,1008
28672	0,4375	47015	0,1793	0,0725	0,1068
32768	0,5000	34858	0,1330	0,0499	0,0831
36864	0,5625	30585	0,1167	0,0343	0,0824
40960	0,6250	21636	0,0825	0,0236	0,0590
45056	0,6875	21035	0,0802	0,0162	0,0640
49152	0,7500	19833	0,0757	0,0111	0,0645
53248	0,8125	18631	0,0711	0,0077	0,0634
57344	0,8750	13823	0,0527	0,0053	0,0475
61440	0,9375	13360	0,0510	0,0036	0,0473

Los resultados experimentales (símbolos) de la Tabla 2-15 se han representado gráficamente en la Figura 2-38.

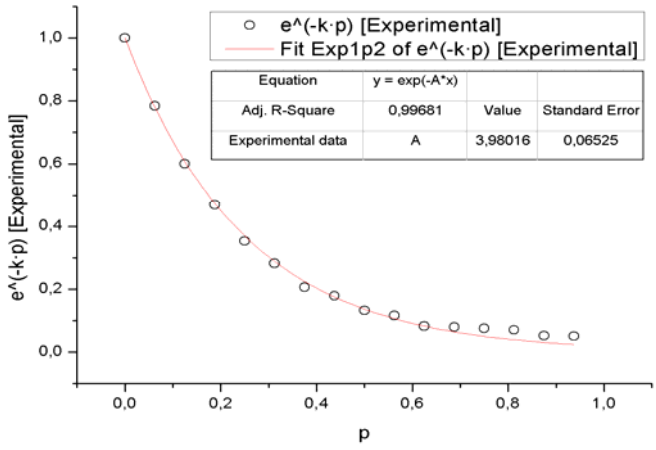

Figura 2-38: Resultados experimentales de la función e^{-x} (UCSL)

De la función matemática ajustada a los datos experimentales en la Figura 2-38 se deriva primeramente que la salida del bloque implementado no se rige por la función matemática evaluada teóricamente $e^{-(\text{Constant\_ajust}\cdot\text{Evaluation\_time}\cdot p)}$, sino que existe un factor adicional presente. A fin de evaluar el valor de este factor y si este se mantiene constante para un rango amplio de variación del parámetro "*Evaluation_time*" fijamos el valor de

"*Constant_adjust*$= 655/2^{16} = 0{,}01$" y luego procederemos a variar el parámetro "Evaluation_time" entre [100, 2000]. Para facilitar la inteligibilidad de los resultados obtenidos tan sólo se representarán en la Figura 2-39 algunas de las medidas realizadas.

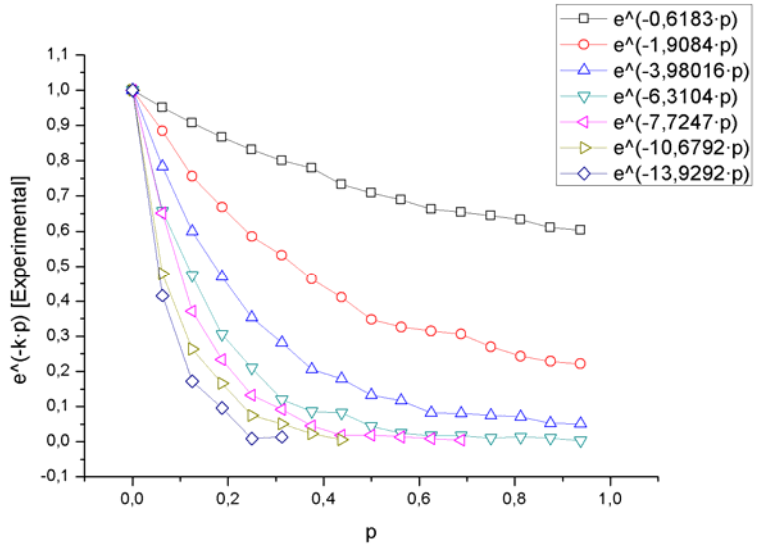

Figura 2-39: Resultados experimentales Pseudo Tangente hiperbólica para diferentes valores de la constante 'Evaluation_Time' (UCSL)

Una vez realizadas las diversas medidas experimentales (Figura 2-39) se ha procedido a evaluar la función exponencial que mejor se les ajusta mediante una serie de ajustes por mínimos cuadrados. Los resultados obtenidos se presentan en la Tabla 2-16 para poder así determinar el factor de ajuste existente entre la "$k_{Teórica}$" predicha por la teoría y la "$k_{Experimental}$" obtenida experimentalmente.

Tabla 2-16: Ajuste de la función $e^{-k \cdot x}$ (UCSL)

Teórico			Experimental	Ajuste
Constant_adjust	Evaluation_time	k=Constant_adjust·Evaluation_time	k(experimental)	Factor ajuste
655/2^16=0,01	100	0,9995	0,6163	0,6166
655/2^16=0,01	200	1,9989	1,2454	0,6231
655/2^16=0,01	300	2,9984	1,9084	0,6365
655/2^16=0,01	400	3,9978	2,5869	0,6471
655/2^16=0,01	500	4,9973	3,3289	0,6661
655/2^16=0,01	600	5,9967	3,9802	0,6637
655/2^16=0,01	700	6,9962	4,5691	0,6531
655/2^16=0,01	800	7,9956	5,4465	0,6812
655/2^16=0,01	900	8,9951	6,3104	0,7015
655/2^16=0,01	1000	9,9945	6,9935	0,6997

655/2^16=0,01	1100	10,9940	7,5104	0,6831
655/2^16=0,01	1200	11,9934	7,7247	0,6441
655/2^16=0,01	1300	12,9929	9,8996	0,7619
655/2^16=0,01	1400	13,9923	9,6363	0,6887
655/2^16=0,01	1500	14,9918	10,6792	0,7123
655/2^16=0,01	1600	15,9912	11,3239	0,7081
655/2^16=0,01	1700	16,9907	12,1399	0,7145
655/2^16=0,01	1800	17,9901	13,9292	0,7743
655/2^16=0,01	1900	18,9896	13,6619	0,7194
655/2^16=0,01	2000	19,9890	15,1599	0,7584

A partir de los datos presentes en la Tabla 2-16 se ha procedido a ajustar una función lineal que relacione la "$k_{Teórica}$" con la "$k_{Experimental}$", dicha relación se presenta a continuación mediante la Ecuación [2-42]:

$$k_{Experimental} = 0{,}7189 \cdot \mathrm{k}_{\mathrm{Teórico}} = 0{,}7189 \cdot Constant\_ajust \cdot Evaluation\_time \text{ , con una } \mathrm{R}^2 = 0{,}9899$$

Ecuación [2-42]

Finalmente y a fin de comprobar que el anterior factor de ajuste fuese independiente del parámetro "Constant_ajust" hemos procedido a realizar toda una serie de medidas fijando ahora el valor del parámetro "Evaluation_Time". Los resultados obtenidos nos permiten afirmar que esta constante de ajuste es independiente de los valores que tomen cualquiera de los dos parámetros.

Por otra parte la presencia de este factor es una incógnita ya que la teoría no lo predice, aunque hemos pensado que su origen se deba al uso de generadores de números pseudo aleatorios (en nuestro caso LFSR de 26 bits) en lugar del uso de generadores de números aleatorios puros.

De la forma descrita se puede implementar cualquier función exponencial $e^{-(0{,}7189 \cdot Constant\_ajust \cdot Evaluation\_time \cdot p)}$ ajustando preferentemente el valor del parámetro "Evaluation_time" y manteniendo el valor de "Constant_ajust" cercano a cero, a fin de cumplir con las condiciones fijadas por la aproximación aplicada.

2.2.3.9 La función Gaussiana ($a \cdot e^{-\left(1/2 \cdot c^2\right) \cdot (p-b)^2}$)

La función Gaussiana [2-21] es una función matemática simétrica con forma de campana, definida mediante la ecuación [2-43]:

$$f(x) = a \cdot e^{-\frac{(x-b)^2}{2 \cdot c^2}}$$ Ecuación [2-43]

Donde los parámetros a, b y c son constantes reales, siendo el parámetro "a" la altura de la campana de Gauss, centrada en el punto definido por el parámetro "b" y con una anchura de la misma (una varianza σ) fijada mediante el parámetro "c".

El valor exacto de la integral impropia sobre todo el rango real puede definirse a partir del valor obtenido de la integral de Gauss (Ecuación [2-44]).

$$\int_{-\infty}^{+\infty} a \cdot e^{-\frac{(x-b)^2}{2 \cdot c^2}} \cdot dx = a \cdot |c| \cdot \sqrt{2 \cdot \pi}$$ Ecuación [2-44]

La importancia de esta función matemática radica en el hecho que aparece en numerosos campos de las ciencias experimentales, ciencias naturales, ciencias sociales, matemáticas y sobre todo en la ingeniería. Particularmente en el marco de la presente tesis nos centraremos en la aplicación de esta función en el campo de la estadística y la teoría de probabilidades, en el cual dicha función aparece como la función densidad de probabilidad de una *distribución normal* que nos permite modelar fenómenos cuyos mecanismos subyacentes son muchas veces desconocidos o son el resultado de la suma de multitud de variables. Mediante el *Teorema del límite central* [2-22] se puede justificar el uso de la *distribución normal*, asumiendo que cada observación se obtiene como la suma de una serie de eventos independientes entre sí.

Las funciones gaussianas se usan para describir la función densidad de probabilidad de una variable aleatoria continua "X" que se rige por una *distribución normal* "$X \sim N(\mu, \sigma)$" con unos parámetros "μ" (que se corresponde con la media de la distribución) y "σ" (que se corresponde con la desviación típica) mientras que "σ^2" se corresponderá con la varianza de la distribución. La función densidad de probabilidad vendrá dada por la Ecuación [2-45].

$$f(x)=a\cdot e^{-\frac{(x-b)^2}{2\cdot c^2}} \rightarrow \begin{cases} a=\dfrac{1}{c\cdot\sqrt{2\cdot\pi}}=\dfrac{1}{\sigma\cdot\sqrt{2\cdot\pi}} \\ \mu=b \\ \sigma^2=c^2 \rightarrow \sigma>0 \\ x\in\Re \end{cases}$$ Ecuación [2-45]

$$\rightarrow f(x)=\frac{1}{\sigma\cdot\sqrt{2\cdot\pi}}\cdot e^{-\frac{(x-\mu)^2}{2\cdot\sigma^2}}, \forall x\in\Re$$

Donde una *distribución normal* [2-24] es aquella en la cual el área encerrada por la función densidad de probabilidad (Ecuación [2-44]) y el eje de abscisas es igual a la unidad "1". Al ser simétrica respecto al eje que pasa por x = μ, deja un área igual a "0,5" a la izquierda y otra igual a "0,5" a la derecha.

Adicionalmente existe lo que se conoce como *distribución normal estándar* [2-24] (o de referencia) que se corresponde con una distribución de media cero (μ =0) y desviación típica la unidad (σ =1). En esta distribución, la probabilidad de ocurrencia de una variable aleatoria "X" continua vendrá descrita por la función distribución de probabilidad $P(X\le x)=\phi_{\mu,\sigma^2}(x)$ (Ecuación [2-46]), los valores de la cual se hallan tabuladas en la literatura [2-23].

$$\phi_{\mu,\sigma^2}(x)=\int_{-\infty}^{x}\varphi_{\mu,\sigma^2}(u)\cdot du=\frac{1}{\sigma\cdot\sqrt{2\cdot\pi}}\cdot\int_{-\infty}^{x}e^{-\frac{(u-\mu)^2}{2\cdot\sigma^2}}\cdot du \rightarrow \phi_{0,1}(x)=\frac{1}{\sqrt{2\cdot\pi}}\cdot\int_{-\infty}^{x}e^{-\frac{u^2}{2}}\cdot du, \forall x\in\Re$$

Ecuación [2-46]

Algunas de las propiedades más importantes de dichas distribuciones son:

- Las funciones normales son simétricas respecto de su media μ
- La moda y la mediana son iguales a la media μ
- Los dos puntos de inflexión de la campana de Gauss cruzan por $x=\mu-\sigma$ y $x=\mu+\sigma$
- La función distribución de probabilidad alrededor de la media μ tiene una serie de propiedades my interesantes para el establecimiento de intervalos de confianza, como son:

- En el intervalo $[\mu+\sigma, \mu-\sigma]$ se halla comprendida aproximadamente el 68,26% del área de la distribución
- En el intervalo $[\mu+2\cdot\sigma, \mu-2\cdot\sigma]$ se halla comprendida aproximadamente el 95,44% del área de la distribución
- En el intervalo $[\mu+3\cdot\sigma, \mu-3\cdot\sigma]$ se halla comprendida aproximadamente el 99,74% del área de la distribución

- Si disponemos una variable aleatoria $X \sim N(\mu, \sigma^2)$, donde "a" y "b" son números reales entonces $(a\cdot X + b) \sim N(a\cdot\mu + b, a\cdot\sigma^2)$
- Si disponemos dos variables normales aleatorias independientes $X \sim N(\mu_x, \sigma_x^2)$ y $Y \sim N(\mu_y, \sigma_y^2)$, de:
  - La suma estará normalmente distribuida con: $U = X + Y \sim N(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$
  - La diferencia estará normalmente distribuida con: $U = X - Y \sim N(\mu_x - \mu_y, \sigma_x^2 + \sigma_y^2)$
- Si disponemos de "n" variables normales estándares independientes la media muestral y la varianza muestral son independientes (Ecuación [2-47]).

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$
$$S^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})}{n-1}$$

Ecuación [2-47]

Entonces, como consecuencia de la simetría respecto de su media μ, es posible relacionar todas las variables aleatorias normales con la función distribución normal estándar. Por lo tanto, si disponemos de una variable normal aleatoria $X \sim N(\mu, \sigma^2)$ será posible definir una segunda variable normal aleatoria $Z = (X-\mu)/\sigma$, tal que $Z \sim N(0,1)$. El proceso de transformación de la distribución asociada a una variable $X \sim N(\mu, \sigma\ )$ en una $N(0,1)$ se denomina *normalización* o *estandarización* de la variable X. Una de las consecuencias más

importantes de esta relación estriba en el hecho que la función distribución de probabilidad [2-24] pasará a poder escribirse como en la Ecuación [2-48]:

$$P(X \leq x) = \Phi\left(\frac{x-\mu}{\sigma}\right) \qquad \text{Ecuación [2-48]}$$

Debido a que la variable Z es una distribución normal estándar $N(0,1)$, la variable aleatoria normal "X" estará caracterizada por una media "μ" y una varianza "σ^2", de tal forma que "$X = \sigma \cdot Z + \mu$".

Para finalizar con la descripción de las propiedades de la distribución normal presentamos el *Teorema del Límite Central* [2-24] el cual establece que bajo ciertas circunstancias (como pueden ser distribuciones independientes e idénticamente distribuidas con varianza finita) la suma de un gran número de variables aleatorias se distribuyen aproximadamente como una distribución normal. Esto implica en la práctica que la distribución normal puede usarse como aproximación a otras distribuciones de probabilidad como es el caso de la distribución binomial y la distribución de Poisson (descritas a lo largo del presente capítulo). En el caso de la distribución binomial (descrita por "n" y "p") se puede aproximar su resultado por una distribución normal, para valores de "n" grandes y valores de "p" no cercanos a 0 y +1. De esta forma la distribución normal aproximada tendrá parámetros "$\mu = n \cdot p$" y "$\sigma^2 = n \cdot p \cdot (1-p)$".
La distribución de Poisson (descrita por "μ") se podrá aproximar por una distribución binomial para valores de "μ" grandes, con lo que la distribución normal aproximada tendrá parámetros "$\mu = \sigma^2 = \lambda$". Se debe tener muy presente que estas aproximaciones son mucho menos precisas en las colas de la distribución que para valores cercanos a la media.

Una vez finalizada la descripción de las propiedades de las matemáticas de las funciones normales pasaremos a describir los pormenores de la implementación estocástica realizada del bloque densidad de probabilidad normal (Figura 2-40).

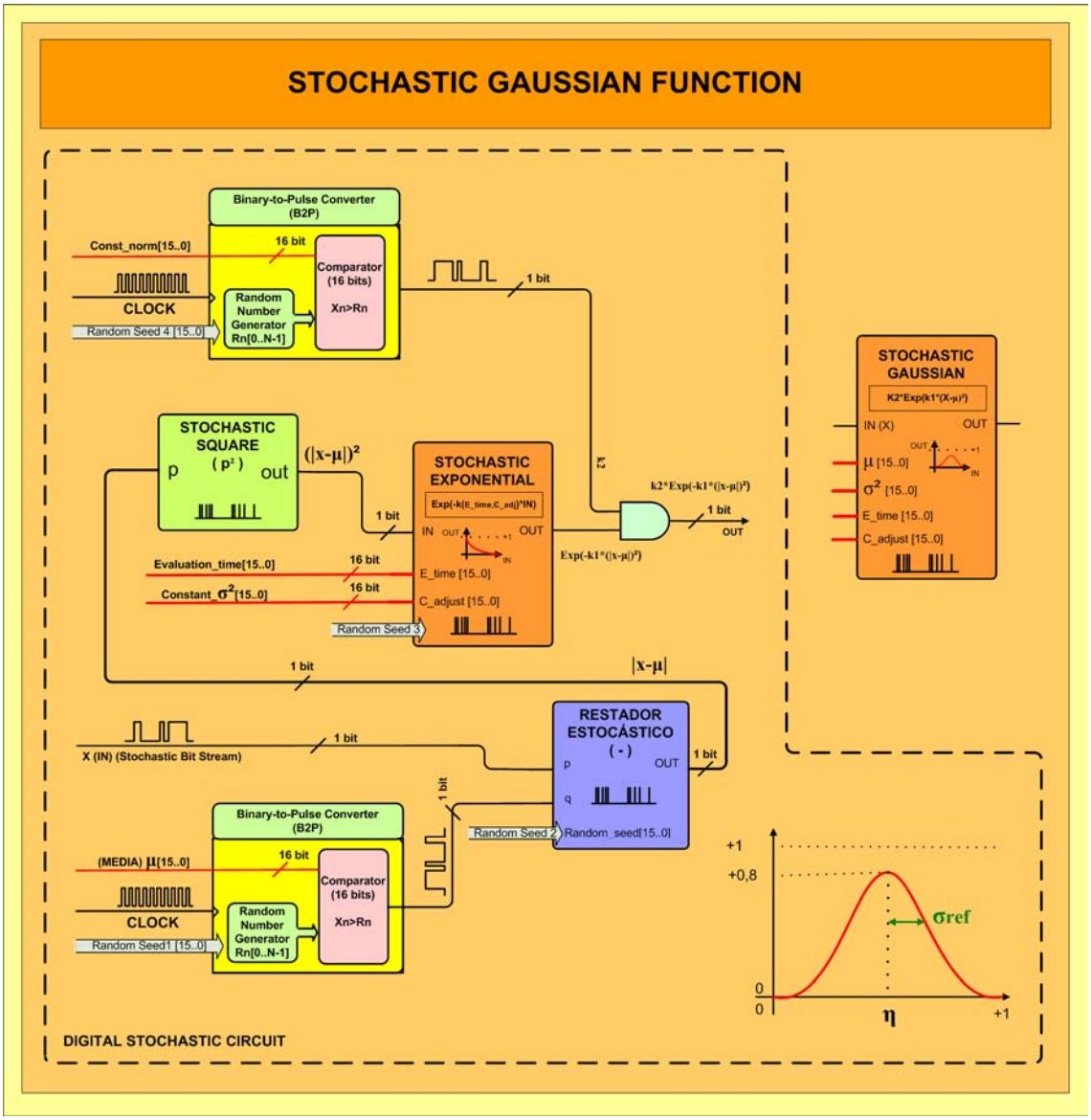

Figura 2-40: Bloque función Gaussiana (UCSL)

El elemento fundamental sobre la que se sustenta la implementación del presente bloque es la función exponencial descrita en el apartado anterior, ya que es imprescindible si se pretende reproducir una función Gaussiana mediante la ecuación [2-36]. Por lo tanto, deberemos operar las diversas distribuciones de probabilidad asociadas a las señales pulsantes de entrada para obtener la señal idónea de entrada al bloque exponencial. La primera operación a realizar es la resta entre la señal de entrada "IN" y la señal asociada a

la media de la distribución "μ", la cual realizaremos con el bloque de resta en valor absoluto (descrito en el subapartado 2.1.2.3). El hecho de realizar la resta en valor absoluto en vez de la resta natural, no tiene mucha importancia ya que la salida del bloque se elevará al cuadrado mediante un bloque potencia 2 (descrito en el subapartado 2.1.2.5), obteniéndose a la salida una señal cuyo valor medio se rige por la Ecuación [2-49].

$$\left.\begin{array}{l} IN \rightarrow \text{Señal estocastica descorrelacionada} \\ \mu \rightarrow \text{Señal estocastica descorrelacionada} \\ IN' \rightarrow \text{Señal estocastica correlacionada} \\ \mu' \rightarrow \text{Señal estocastica correlacionada} \end{array}\right\} \longrightarrow \begin{cases} E\left(out\left(IN,\mu\right)\right) = \left(IN' \oplus \mu'\right)(t) \cdot \left(IN' \oplus \mu'\right)(t-8) \\ = \left(\left|IN-\mu\right|\right)^2 = \left(IN-\mu\right)^2 = \dfrac{\left(\left|IN-\mu\right|\right)^2}{2^{2 \cdot n}} \end{cases}$$

Ecuación [2-49]

Por lo tanto, a la entrada del bloque exponencial por cuestiones de velocidad de operación (evitando así el uso de un bloque divisor) se ha optado por usar la señal $\left(\left|IN-\mu\right|\right)^2$ en vez de $Z^2 = \left(\left(X-\mu\right)/\sigma\right)^2$ como se ha descrito en la teoría (Ecuación 2-47). De esta forma el factor "$1/\sigma^2$"se introducirá mediante la relación entre el factor de ajuste "*Constant_adjunt*" y el parámetro "*Evaluation_time*" que mantendremos constante del bloque *estocá*stico *Exponencial negativa* con la cual ajustaremos el valor de la desviación estándar σ de la Gaussiana. No obstante para que el bloque exponencial opere correctamente se requerirá que el valor de esta constante sea cercano a cero. El parámetro "Evaluation_time" se fijará a un valor constante que equivaldrá a un valor de "$n \rightarrow \infty$", a fin que se asegure el supuesto de la aproximación de la salida del bloque a una distribución de Poisson. Todo ello limita el rango de valores que podrá tomar la varianza σ^2. Con lo cual la salida del bloque exponencial se regirá por la Ecuación [2-50]:

$$f(0;\lambda) = f\left(0;\left(|IN-\mu|\right)^2 \cdot n\right) = e^{-\left(\frac{1}{2\cdot\sigma^2}\cdot\left(|IN-\mu|\right)^2\right)} = e^{-\left(0{,}7189\cdot Cons\tan t\_ajust\cdot Evaluation\_time\cdot\left(|IN-\mu|\right)^2\right)}$$

$$\sigma^2 = \frac{1}{2\cdot 0{,}7189\cdot Cons\tan t\_ajust\cdot Evaluation\_time} \rightarrow \sigma = \pm\sqrt{\frac{1}{2\cdot 0{,}7189\cdot Cons\tan t\_ajust\cdot Evaluation\_time}} \rightarrow$$

$$\rightarrow \begin{cases} Evaluation\_time = \dfrac{1}{2\cdot 0{,}7189\cdot Cons\tan t\_ajust\cdot\sigma^2} \\ Cons\tan t\_ajust = \dfrac{\#Const\_ajust(16bits)}{2^{16}} \end{cases}$$

Ecuación [2-50]

Entonces a fin de asegurar la normalización de la distribución Gaussiana implementada para que se ajuste a una distribución normal se ha incorporado una puerta **AND** para multiplicar la salida del bloque *Exponencial negativa* por el parámetro "Constant_normalization" (Ecuación [2-38]), de tal forma que la salida del bloque se regirá por la Ecuación [2-51]:

$$out = Const\_norm\cdot f(0;\lambda) = (Const\_norm)\cdot e^{-\left(\frac{1}{2\cdot\sigma^2}\cdot\left(|IN-\mu|\right)^2\right)}$$

$$Const\_norm = \frac{1}{\sigma\cdot\sqrt{2\cdot\pi}} = \sqrt{\frac{2\cdot 0{,}7189\cdot Cons\tan t\_ajust\cdot Evaluation\_time}{2\cdot\pi}} \rightarrow$$

$$\rightarrow \lim_{\substack{Cons\tan t\_ajust\rightarrow 0 \\ Evaluation\_time\rightarrow 65535}} \sqrt{\frac{0{,}7189\cdot Cons\tan t\_ajust\cdot Evaluation\_time}{\pi}} >> 1$$

Ecuación [2-51]

De la Ecuación [2-51] se deduce que el valor de la constante de normalización que se debe usar para normalizar la salida del bloque exponencial negativa excede el rango de la representación de la lógica estocástica clásica, por lo cual hablaremos desde este momento de función pseudo-normales estocásticas.

Para comprobar el correcto funcionamiento de la implementación de la función Gaussiana pseudo-normal desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-17. Éstas han consistido en ir variando el valor medio de la distribución asociada a la señal pulsante "IN" entre [0, +1], con los parámetros del bloque prefijados a "*Media μ=32768=0,5*", "*Evaluation_time=1600*" y "*Constant_adjust=655*" que prefijarían una inversa de la varianza teórica igual a "$1/(2\cdot\sigma^2)$*=0,0870*", y finalmente una constante de normalización "*Const_norm=1*"

Tabla 2-17: Medidas de la función gaussiana pseudo-normal (UCSL)					
DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	OUT [16bits]	OUT	OUT= $e^{-\left(0,0711\cdot\left(\left\|IN-\mu\right\|\right)^2\right)}$	ERROR
0	0,0000	1601	0,0244	0,0297	0,0053
4096	0,0625	3202	0,0489	0,0677	0,0189
8192	0,1250	9572	0,1461	0,1384	0,0077
12288	0,1875	16010	0,2443	0,2532	0,0089
16384	0,2500	27217	0,4153	0,4152	0,0001
20480	0,3125	35723	0,5451	0,6099	0,0648
24576	0,3750	48862	0,7456	0,8027	0,0571
28672	0,4375	60732	0,9267	0,9466	0,0198
32768	0,5000	65535	1,0000	1,0000	0,0000
36864	0,5625	62333	0,9511	0,9465	0,0046
40960	0,6250	52727	0,8046	0,8027	0,0019
45056	0,6875	41156	0,6280	0,6099	0,0181
49152	0,7500	28818	0,4397	0,4151	0,0246
53248	0,8125	16463	0,2512	0,2532	0,0020
57344	0,8750	9816	0,1498	0,1383	0,0114
61440	0,9375	3202	0,0489	0,0677	0,0189

En la práctica la inversa de la varianza obtenida experimentalmente ($1/(2\cdot\sigma^2)=0,0711$) dista un cierto factor de la predicha teóricamente. Debido a este hecho, hemos decidido ajustar experimentalmente una función matemática que nos prediga el valor que tomará la varianza en función de los parámetros de ajuste del circuito. No obstante, antes de proceder con ello presentamos en la Figura 2-41 los resultados experimentales obtenidos (símbolos) a la vez que los predichos por la teoría (línea continua).

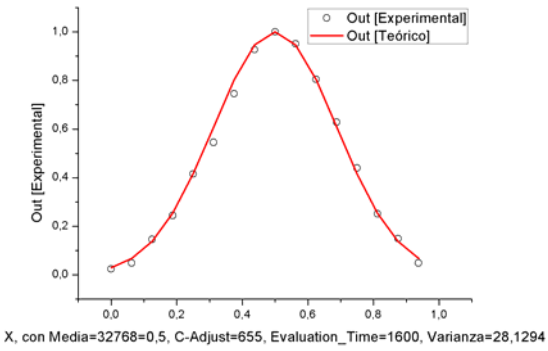

Figura 2-41: Resultados experimentales de la función Gaussiana pseudo-normal (UCSL)

Como bien puede apreciarse tanto en la Figura 2-41 como en la última columna de la Tabla 2-17, la aproximación realizada se ajusta bien a la función Gaussiana real.

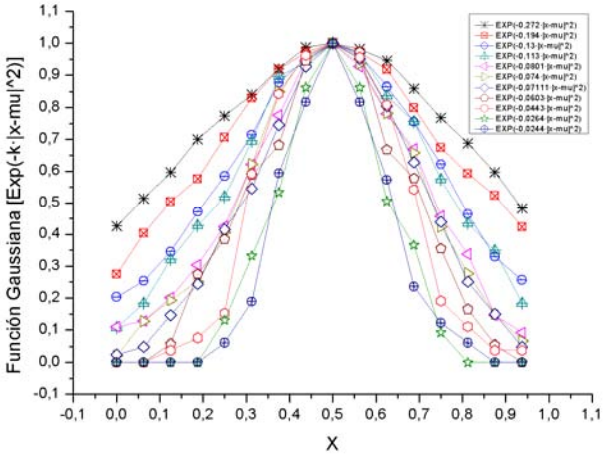

Figura 2-42: Resultados experimentales de la función Gaussiana fijada en un valor medio μ=0,5 y a la cual hemos procedido a modificar la varianza de la distribución (UCSL).

A la vista que la posición de la media de la distribución experimental "μ" se halla en la posición esperada, pero que en cambio los parámetros de la varianza experimental no se ajustan a lo predicho por la teoría, hemos procedido a realizar un conjunto de medidas experimentales (Figura 2-42) que se presentan en la Tabla [2-18]. En éstas se de termina la función matemática que ajusta la varianza experimental o la inversa de esta ($1/\sigma^2$) en función de los parámetros de configuración del bloque en función, como son "*Evaluation_time*" y "*Constant_adjust*". Para ello hemos fijamos el valor de "*Constant_adjust*$= 655/2^{16} = 0{,}01$" para luego proceder a variar el parámetro "Evaluation_time" entre [400, 4000].

Tabla 2-18: Ajuste de la varianza experimental de la función gaussiana (UCSL)

Teórico			Experimental	Teórica
Constant_adjust	**Evaluation_time**	**Factor=C_adjust·E_time**	σ^2	σ^2
655/2^16=0,01	400	3,9978	3,6765	2,8740
655/2^16=0,01	600	5,9967	5,1546	4,3110
655/2^16=0,01	800	7,9956	7,6923	5,7480
655/2^16=0,01	1000	9,9945	8,8496	7,1851
655/2^16=0,01	1200	11,9934	12,4844	8,6221
655/2^16=0,01	1400	13,9923	13,5135	10,0591
655/2^16=0,01	1600	15,9912	14,0647	11,4961
655/2^16=0,01	1800	17,9901	16,5837	12,9331
655/2^16=0,01	2000	19,9890	20,7900	14,3701
655/2^16=0,01	2500	24,9866	23,0947	17,9626
655/2^16=0,01	3000	29,9835	37,8788	21,5552
655/2^16=0,01	4000	39,9780	40,9836	28,7402

A partir de los datos presentes en la Tabla [2-18] se ha procedido a ajustar una función lineal entre el producto de los parámetros "C-Adjust·Evaluation_time" y la varianza de la distribución ajustada experimentalmente. La relación obtenida entre ambas se presenta a continuación mediante la Ecuación [2-52]:

$$\sigma^2 = \frac{0{,}55405}{Constant\_ajust \cdot Evaluation\_time}, \qquad R^2 = 0{,}9917 \qquad \text{Ecuación [2-52]}$$

Finalmente y a fin de comprobar que el anterior factor de ajuste fuese independiente del parámetro "Constant_ajust" hemos procedido a realizar toda una serie de medidas fijando ahora el valor del parámetro "Evaluation_Time". Los resultados obtenidos nos permiten afirmar que esta constante de ajuste es independiente de los valores que tomen cualquiera de los dos parámetros de ajuste de la función exponencial.

De la Ecuación [2-52] se deduce que la varianza de la distribución se corresponderá al producto de los parámetros de ajuste, como establece la teoría. Por otra parte la presencia del factor "k1=0,5541" es una incógnita por nuestra parte; ya que la teoría no lo predice. Además en caso de que se diera debería corresponderse al factor "k1=1/(2·0,7189)=0,6955" evaluado en el ajuste de la ganancia de la función exponencial negativa.

2.2.4 LA CODIFICACIÓN ESTOCÁSTICA CLÁSICA CON SIGNO (SCSL)

La codificación estocástica clásica sólo permite codificar las distribuciones de probabilidad de conmutación asociadas a las señales pulsantes un valor medio entre 0 y +1, en consecuencia no puede operar con valores negativos. No obstante dicha representación numérica se puede extender a valores negativos mediante la realización de un cambio de variable (Ecuación [2-53]) donde "p" representa la probabilidad de conmutación de la señal pulsante.

$$p \in |0,+1| \rightarrow p^* = 2 \cdot p - 1 \rightarrow p^* \in |-1,+1| \qquad \text{Ecuación [2-53]}$$

De esta forma una señal estocástica en vez de representar señales con valores medios de conmutación comprendidos entre 0 y +1, se representan señales con valores medios de conmutación entre [-1, +1]. Cabe destacar que en esta nueva representación el cero "0" se corresponde con un valor medio de conmutación del 50%→p=0,5 de la señal pulsante.

Dicha notación ya ha sido tratada previamente en la literatura; como B.R Gaines en el año 1969 en un artículo titulado "*Stochatic Computing Systems*" [2-12] en el cual se describían los "*formatos bipolares*", que vienen a ser lo mismo que lo que se define en esta tesis como lógica estocástica clásica con signo. Trabajos más recientes como el de D.Brown Bradley y C. Card Howard (2001) titulado "*Stochastic Neural Computation I: Computational Elements"* [2-19] han vuelto a abordar la implementación de múltiples funciones aritméticas basadas en las codificaciones bipolares de B.R. Gaines, para la implementación de redes neuronales pulsantes.

Con esta nueva notación las operaciones aritméticas descritas en el apartado anterior dejan de poder realizarse mediante los mismos bloques lógicos. Por ejemplo en esta codificación la multiplicación ya no puede realizarse mediante una puerta **AND**, sino que debe realizarse mediante una puerta **XNOR.** Aunque este cambio soluciona el problema de la representación de los números negativos que existe en la lógica estocástica clásica, sigue

prevaleciendo el problema de sólo poder representar valores een el intervalo [-1, +1]. Como una solución parcial a esta limitación existe la posibilidad de normalizar los diversos factores y magnitudes, a fin que todos los valores estén acotados en [-1, +1]. El inconveniente de esta solución es que requiere de un conocimiento previo muy detallado de los rangos de valores de las magnitudes a operar, a fin de poder realizar la normalización más idónea, y maximizar así la resolución del sistema. Esta notación no permitirá representar en ningún momento un resultado intermedio de una operación cuyo resultado sea mayor a "+1" o menor a "-1", saturando a estos valores límite.

Desde el punto de vista estadístico una de las consecuencias de este cambio de variables es que el error de la conversión se duplica (Ecuación [2-54]) al haberse duplicado el espacio de representación inicial de [0, +1] a [-1, +1].

$$\pm\frac{1bit}{2^n}\rightarrow[0,+1]\Rightarrow[-1,+1]\rightarrow\pm\frac{1bit}{2^n}2=\pm\frac{1bit}{2^{n-1}} \qquad \text{Ecuación [2-54]}$$

Para esta codificación estocástica clásica con signo "*Signed Classic Stochastic Logic*" (*SCSL*) se han desarrollado un gran número de operaciones y funciones estocásticas a fin de poder implementar cualquier sistema. Cabe remarcar que muchas de ellas serán la base de la implementación de bloque para otras codificaciones que describiremos más adelante en este capítulo.

Las funciones aritméticas/matemáticas desarrolladas se presentan a lo largo de los siguientes subapartados con sus correspondientes implementaciones digitales así como las medidas experimentales realizadas a fin de comprobar su correcto funcionamiento.

2.2.4.1 La operación complementaria

La operación complementaria de una señal estocástica dada "p*" se realiza mediante una puerta **NOT** al igual que sucede en la lógica estocástica clásica, siendo uno de los pocos bloques de función que se implementan de igual forma para ambas codificaciones.

La demostración teórica de la equivalencia entre la operación lógica **NOT** y la función complementaria se presenta en la Ecuación [2-52] para una señal pulsante "p*" con un

valor medio de la distribución codificado mediante el cambio de variables (Ecuación [2-55]) anteriormente descrito.

$$\left\{\overline{p} = (1-p) \Rightarrow \text{clasicamente} \Rightarrow out = NOT(p) = \overline{p} = (1-p)\right.$$

Si los datos han sido codificados mediante el cambio de variables de lógica estocástica con signo :

$$\begin{cases} p^* = 2p - 1 \\ out^* = 2out - 1 \end{cases} \rightarrow \begin{cases} p = \dfrac{p^*+1}{2} \\ out = \dfrac{out^*+1}{2} \end{cases} \Rightarrow out = \frac{out^*+1}{2} = 1 - \frac{p^*+1}{2} \Rightarrow out^* = 2 - p^* - 2 = -p^*$$

$$E[out^*] = P(-p^*) = -\frac{(2 \cdot P - 1)}{2^n} = \frac{(1 - 2 \cdot P)}{2^n}$$

Ecuación [2-55]

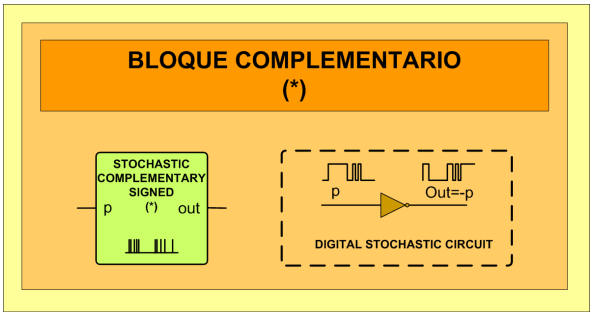

Figura 2-43: Bloque operación complementaria (SCSL)

El valor medio "$\overline{out}$" a la salida del bloque P2B(N) después de "$N_{ciclos\_evaluación}$" será una variable aleatoria de valor medio (Ecuación [2-56]):

$$\overline{out} = N_{ciclos\_evaluación} \cdot \frac{(1 - 2 \cdot P)}{2^n} \qquad \text{Ecuación [2-56]}$$

La desviación típica del valor acumulado en el bloque P2B(N) vendrá descrita por la Ecuación [2-57]:

$$\sigma(\overline{out}) = \sqrt{N_{ciclos\_evaluación} \cdot \frac{(1 - 2 \cdot P)}{2^n} \cdot \left(1 - \frac{(1 - 2 \cdot P)}{2^n}\right)} \qquad \text{Ecuación [2-57]}$$

En la Figura 2-43 se presenta la implementación digital realizada para el bloque función complementaria.

A continuación procederemos a presentar en la Tabla 2-19, las medidas realizadas para este bloque, variando el valor medio de la distribución asociada a la señal pulsante “p*” en [-1, +1].

Tabla 2-19: Medidas de la función complementaria (SCSL)

DATOS DE ENTRADA		EXPERIMENTAL	
P [16bits]	Valor P	OUT [16bits]	Valor OUT
0	-1,0000	65535	1,0000
4096	-0,8750	61363	0,8727
8192	-0,7500	57346	0,7501
12288	-0,6250	53328	0,6275
16384	-0,5000	49181	0,5009
20480	-0,3750	45045	0,3747
24576	-0,2500	40873	0,2474
28672	-0,1250	36774	0,1223
32768	0,0000	32821	0,0016
36864	0,1250	28668	-0,1251
40960	0,2500	24328	-0,2576
45056	0,3750	20517	-0,3739
49152	0,5000	20517	-0,3739
53248	0,6250	12262	-0,6258
57344	0,7500	8239	-0,7486
61440	0,8750	4000	-0,8780

Seguidamente representamos gráficamente (Figura 2-44) los datos experimentales de la Tabla 2-19; por una parte los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), y a continuación realizamos un ajuste por mínimos cuadrados para evaluar la bondad de la implementación digital.

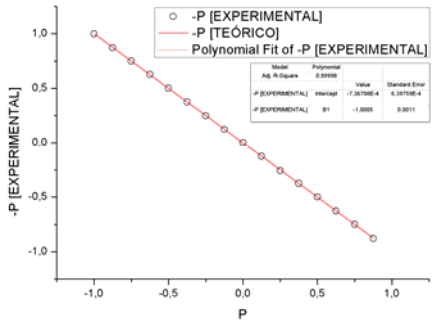

Figura 2-44: Resultados experimentales de la función complementaria (SCSL)

Si comparamos la función de transferencia obtenida en la Figura 2-44 a partir del ajuste por mínimos cuadrados con la descrita por la teoría (Ecuación [2-55]) se puede apreciar en la Ecuación [2-58] como ambos resultados son idénticos.

Teórico $\rightarrow OUT_{TEO} = (-1 \cdot P)$

Experimental $\rightarrow OUT_{EXP} = (-1{,}0005 \cdot P)$, con una $R^2 = 1$

Ecuación [2-58]

2.2.4.2 La operación suma/resta en valor medio

La operación suma en valor medio al igual que la operación complementaria se implementa de forma idéntica que en la codificación estocástica clásica sin signo (mediante un multiplexor digital que dispone de dos entradas para interconectar con las señales estocásticas "p*" y "q*" a sumar y una señal adicional de selección "c"). A la salida del multiplexor se obtiene una nueva trama con distribución de probabilidad que es una combinación lineal de las distribuciones de probabilidad de las señales operadas. Esta combinación lineal está forzosamente relacionada con la señal de control de la activación, que deberá hallarse descorrelacionada con respecto a las dos señales a operar.

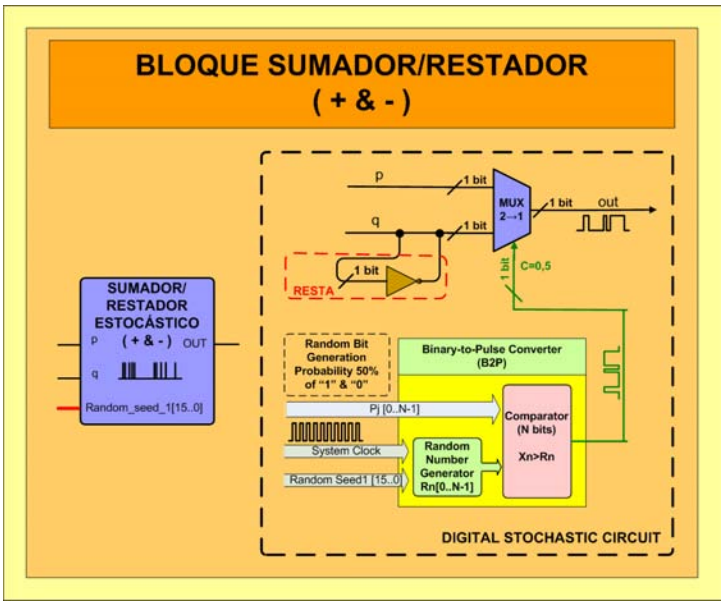

Figura 2-45: Bloque sumador/restador (SCSL)

Una de las ventajas de la nueva codificación estriba en el hecho que la función resta se implementa de forma idéntica que la suma, ya que el bloque suma puede operar con magnitudes de distinto signo. Para restar dos señales estocásticas deberemos obtener la señal complementaria de la señal a restar (mediante una puerta NOT) y proceder a sumar la señal complementaria resultante con la otra señal pulsante a operar.

La demostración teórica de la equivalencia entre el uso de un bloque multiplexor digital y la función suma en valor medio se presenta en la Ecuación [2-59] para dos señales pulsantes “p*” y “q*” con un valor medio de la distribución codificado mediante el cambio de variables (Ecuación [2-53]) anteriormente descrito, y una señal de control de conmutación ”c*” fijada a un 50%.

$$\begin{cases} \bar{p} = (1-p) \\ \bar{q} = (1-q) \\ \bar{c} = (1-c) \end{cases} \Rightarrow \text{clasicamente} \Rightarrow out = p\cdot c + q\cdot\bar{c} = p\cdot c + q\cdot(1-c) = p\cdot c - q\cdot c + q$$

Si los datos han sido codificados mediante el cambio de variables de la lógica estocástica con signo se tiene:

$$\begin{cases} p^* = 2p - 1 \\ q^* = 2\cdot q - 1 \\ c^* = 2\cdot c - 1 \\ out^* = 2out - 1 \end{cases} \rightarrow \begin{cases} p = \dfrac{p^*+1}{2} \\ q = \dfrac{q^*+1}{2} \\ c = \dfrac{c^*+1}{2} \\ out = \dfrac{out^*+1}{2} \end{cases} \Rightarrow out = \frac{out^*+1}{2} = \frac{(p^*+1)}{2}\cdot\frac{(c^*+1)}{2} - \frac{(q^*+1)}{2}\cdot\frac{(c^*+1)}{2} + \frac{q^*+1}{2}$$

$$\Rightarrow 2\cdot out^* = p^*\cdot c^* + p^* + c^* + 1 - q^*\cdot c^* - q^* - c^* - 1 + 2\cdot q^* + 2 - 2 = p^*\cdot c^* - q^*\cdot c^* + p^* + q^* \Rightarrow$$

$$\Rightarrow out^* = \frac{p^*+q^*+p^*\cdot c^*-q^*\cdot c^*}{2} \rightarrow \text{En el supuesto que fijemos } c^* = 0 \rightarrow out^* = \frac{p^*+q^*}{2}$$

$$E(out^*) = \frac{1}{2}\cdot\left(\frac{(2P-1)}{2^n} + \frac{(2Q-1)}{2^n}\right) = \frac{(P+Q-1)}{2^n}$$

Ecuación [2-59]

La implementación digital del bloque sumador se presenta en la Figura 2-45, mientras que para implementar el restador se requiere de la incorporación de una puerta **NOT** (como se indica en la Figura 2-45).

A continuación presentamos en la Tabla 2-20 las medidas realizadas para corroborar el correcto funcionamiento de la implementación digital realizada. Las cuales han consistido en fijar el valor medio de la distribución de la señal pulsante “q*=-0,25” y variar el valor medio de la distribución asociada a la señal pulsante “p*” entre [-1, +1].

Tabla 2-20: Medidas de la operación suma/resta (SCSL)							
DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
Valor P	P [16bits]	Valor Q	Q [16bits]	OUT [16bits]	Valor OUT	OUT=P+Q	ERROR
-1,0000	0	-1,0000	0	0	-1,0000	-1,0000	0,0000
-1,0000	0	0,2500	24576	-0,2500	-0,6235	-0,6250	0,0016
-0,8750	4096	0,2500	24576	-0,2500	-0,5542	-0,5625	0,0083
-0,7500	8192	0,2500	24576	-0,2500	-0,4942	-0,5000	0,0058
-0,6250	12288	0,2500	24576	-0,2500	-0,4341	-0,4375	0,0034
-0,5000	16384	0,2500	24576	-0,2500	-0,3728	-0,3750	0,0022
-0,3750	20480	0,2500	24576	-0,2500	-0,3156	-0,3125	0,0031
-0,2500	24576	0,2500	24576	-0,2500	-0,2558	-0,2500	0,0058
-0,1250	28672	0,2500	24576	-0,2500	-0,1875	-0,1875	0,0000
0,0000	32768	0,2500	24576	-0,2500	-0,1189	-0,1250	0,0061
0,1250	36864	0,2500	24576	-0,2500	-0,0668	-0,0625	0,0043
0,2500	40960	0,2500	24576	-0,2500	0,0080	0,0000	0,0080
0,3750	45056	0,2500	24576	-0,2500	0,0639	0,0625	0,0014
0,5000	49152	0,2500	24576	-0,2500	0,1321	0,1250	0,0070
0,6250	53248	0,2500	24576	-0,2500	0,1814	0,1875	0,0061
0,7500	57344	0,2500	24576	-0,2500	0,2474	0,2500	0,0027
0,8750	61440	0,2500	24576	-0,2500	0,3110	0,3125	0,0015

Si representamos gráficamente (Figura 2-46) los datos experimentales de la Tabla 2-20 vemos por una parte los resultados experimentales obtenidos (símbolos) y por la otra los predichos por la teoría (línea continua). Finalmente también hemos realizado un ajuste por mínimos cuadrados de los primeros para evaluar la bondad de la implementación digital realizada.

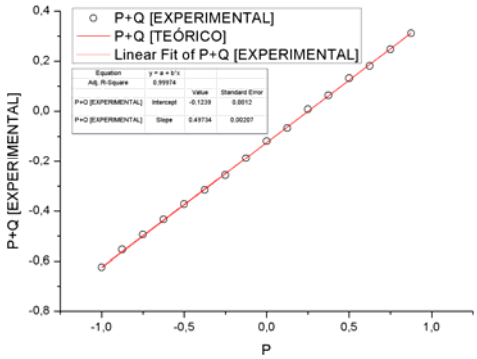

Figura 2-46: Resultados experimentales de la operación suma/resta (SCSL)

Si comparamos la función de transferencia obtenida de la Figura 2-46 mediante el ajuste por mínimos cuadrados, con la descrita por la teoría (Ecuación [2-59]) se puede apreciar en la Ecuación [2-60] cómo ambos resultados son prácticamente idénticos.

$$\text{Teórica} \rightarrow OUT_{TEO} = 0{,}5\cdot(P+Q) = 0{,}5\cdot(P-0{,}25) = 0{,}5\cdot P - 0{,}125$$
$$\text{Experimental} \rightarrow OUT_{EXP} = 0{,}4973\cdot(P-0.1239), \text{ con una } R^2 = 0{,}9997$$

Ecuación [2-60]

2.2.4.3 La operación multiplicación

Como ya se ha comentado la implementación de la multiplicación entre dos señales estocásticas "p*" y "q*" se realizará mediante una puerta **XNOR** de dos entradas.

El uso de una puerta **XNOR** en vez de una puerta **AND** como sucede en la lógica estocástica clásica se debe al cambio de variables realizado (Ecuación [2-53]). Mediante la Ecuación [2-61] se presenta la equivalencia matemática entre la operación aritmética multiplicación y la función **XNOR** sobre dos señales pulsantes con un valor medio codificado mediante el anterior cambio de variables.

$$\begin{cases} \bar{p} = (1-p) \\ q = (1-p) \end{cases} \Rightarrow \text{clasicamente} \Rightarrow out = (p\ XNOR\ q) = \overline{p\cdot\bar{q} + \bar{p}\cdot q} = (\bar{p} + q)\cdot(p + \bar{q}) =$$

$$= \bar{p}\cdot\bar{q} + p\cdot q = (1-p)\cdot(1-q) + p\cdot q = 1 - q - p + p\cdot q + p\cdot q = 1 - q - p + 2\cdot p\cdot q$$

Si los datos han sido codificados mediante el cambio de variables de la lógica estocástica con signo:

$$\begin{cases} p^* = 2p - 1 \\ q^* = 2q - 1 \\ out^* = 2out - 1 \end{cases} \rightarrow \begin{cases} p = \dfrac{p^*+1}{2} \\ q = \dfrac{q^*+1}{2} \\ out = \dfrac{out^*+1}{2} \end{cases} \Rightarrow out = \frac{out^*+1}{2} = 1 - \frac{q^*+1}{2} - \frac{p^*+1}{2} + 2\cdot\frac{p^*+1}{2}\cdot\frac{q^*+1}{2} \Rightarrow$$

$$\Rightarrow out^* = 2 - (q^*+1) - (p^*+1) + (p^*+1)\cdot(q^*+1) - 1 = p^*\cdot q^* + p^* + q^* - p^* - q^* = p^*\cdot q^*$$

$$E(out^*) = P(p^*)\cdot P(q^*) = \frac{(2\cdot P - 1)}{2^n}\cdot\frac{(2\cdot Q - 1)}{2^n} = \frac{(2\cdot P - 1)\cdot(2\cdot Q - 1)}{2^{2n}}$$

Ecuación [2-61]

Donde las señales etiquetadas como "p*" y "q*" son señales pulsantes genéricas cuyo valor medio asignado ha sido codificado mediante el cambio de variables presentado en la Ecuación [2-53]. Por lo tanto, con la Ecuación [2-61] se demuestra teóricamente que la puerta **XNOR** implementa la operación multiplicación, en la presente codificación.

La implementación digital del bloque multiplicación se presenta a continuación en la Figura 2-47.

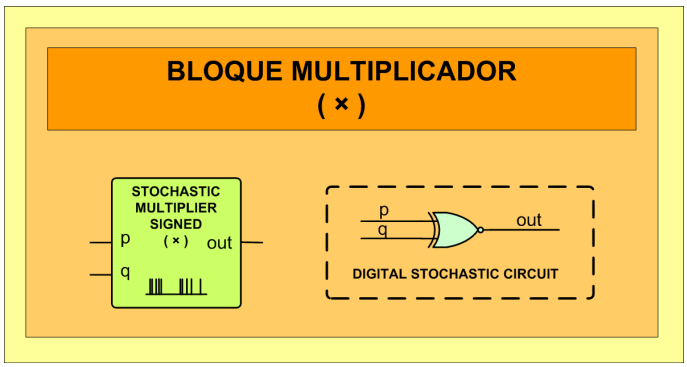

Figura 2-47: Bloque multiplicación (SCSL)

Para comprobar el correcto funcionamiento de la implementación digital realizada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-21. Éstas han consistido en fijar el valor medio de la distribución de la señal pulsante "q*=-0,25" y variar el valor medio de la distribución asociada a la señal pulsante "p*" entre [-1, +1].

Tabla 2-21: Medidas de la operación multiplicación (SCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
Valor P	P [16bits]	Valor Q	Q [16bits]	OUT [16bits]	Valor OUT	OUT=P·Q	ERROR
-1,0000	0	-1,0000	0	65533	0,9999	1,0001	0,0001
-1,0000	0	-0,2500	24576	40953	0,2498	0,2500	0,0002
-0,8750	4096	-0,2500	24576	40016	0,2212	0,2188	0,0024
-0,7500	8192	-0,2500	24576	39054	0,1918	0,1875	0,0043
-0,6250	12288	-0,2500	24576	38048	0,1611	0,1563	0,0049
-0,5000	16384	-0,2500	24576	36840	0,1243	0,1250	0,0007
-0,3750	20480	-0,2500	24576	35951	0,0971	0,0938	0,0034
-0,2500	24576	-0,2500	24576	34974	0,0673	0,0625	0,0048
-0,1250	28672	-0,2500	24576	33892	0,0343	0,0313	0,0031
0,0000	32768	-0,2500	24576	32653	-0,0035	0,0000	0,0035
0,1250	36864	-0,2500	24576	31588	-0,0360	-0,0313	0,0048
0,2500	40960	-0,2500	24576	30830	-0,0591	-0,0625	0,0034
0,3750	45056	-0,2500	24576	29929	-0,0866	-0,0938	0,0071
0,5000	49152	-0,2500	24576	28796	-0,1212	-0,1250	0,0038
0,6250	53248	-0,2500	24576	27577	-0,1584	-0,1563	0,0022
0,7500	57344	-0,2500	24576	26793	-0,1823	-0,1875	0,0052
0,8750	61440	-0,2500	24576	25481	-0,2224	-0,2188	0,0036

En la Figura 2-48 mostramos la comparación de los resultados experimentales obtenidos (símbolos) con los predichos por la teoría (línea continua), a la vez que realizamos un ajuste

de mínimos cuadrados sobre las medidas experimentales a fin de poder evaluar la bondad de la implementación digital realizada.

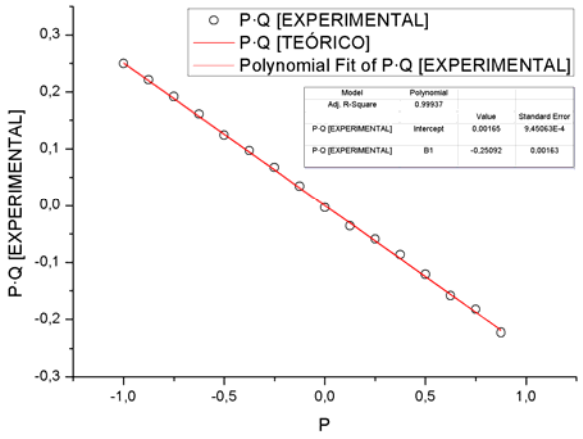

Figura 2-48: Resultados experimentales de la operación multiplicación (SCSL)

Comparando la función de transferencia obtenida de la Figura 2-48 mediante el ajuste por mínimos cuadrados con la descrita por la teoría (Ecuación [2-61]) para los valores iniciales dados, se puede apreciar en la Ecuación [2-62] como ambos resultados son prácticamente idénticos. El error relativo máximo alcanzado (Tabla 2-21) en todas las medidas experimentales realizadas es del 0,26%.

$$\text{Experimental} \quad \rightarrow OUT_{EXP} = -0.2509 \cdot P \text{, con una } R^2 = 0{,}9994$$
$$\text{Teórico} \quad \rightarrow OUT_{TEO} = -0{,}25 \cdot P$$

Ecuación [2-62]

2.2.4.4 La operación potenciación (P^N)

La operación potenciación se implementa de forma idéntica que en la codificación estocástica clásica sin signo, con la salvedad que para obtener la potencia N de una señal pulsante “p*” será necesario multiplicarla N veces por ellas misma mediante una puerta **XNOR** en vez de una puerta **AND** como sucedía con la codificación estocástica clásica.

Como ejemplo de potenciación se presenta la operación cuadrado (potencia 2) para una señal pulsante "p*" de entrada. Al igual que en el caso descrito para la codificación estocástica clásica (con signo) la descorrelación temporal entre las señales estocásticas a operar se ha obtenido mediante el uso de un registro de desplazamientos de 8-bits. Dicha operación vendrá descrita teóricamente mediante la Ecuación [2-63]:

$$\left.\begin{array}{l} P_{Magnitud\_binaria} \rightarrow \mathrm{p}*_{\text{señal\_estocástica\_1bit}} \\ \mathrm{OUT}_{\text{Magnitud\_binaria}} \leftarrow \mathrm{out}*_{\text{señal\_estocástica\_1bit}} \end{array}\right\} \xrightarrow{Cuadrado} \begin{cases} OUT* = (P* \cdot P*) \rightarrow E(out(p(t), p(t-8))) = \\ = E(p*(t)\ \mathrm{XNOR}\ p*(t-8)) = P(p*)\cdot P(p*) = \\ = \dfrac{(2P-1)^2}{2^{2n}} \end{cases}$$

Ecuación [2-63]

Cabe remarcar que el resultado de la potencia N de una señal pulsante con un valor medio "p*" definido entre [-1,+1] será una nueva distribución con un valor medio asociado descrito por la Ecuación [2-64]:

$$p* \in [-1,+1] \rightarrow (p*)^N = \begin{cases} p* > 0 \rightarrow \begin{cases} N\ \text{par} \rightarrow (p*)^N \in [0,+1] \\ \text{N impar} \rightarrow (p*)^N \in [0,+1] \end{cases} \\ \text{p*} < 0 \rightarrow \begin{cases} N\ \text{par} \rightarrow (p*)^N \in [0,+1] \\ \text{N impar} \rightarrow -1\cdot(p*)^N \in [-1,0] \end{cases} \\ p* = 0 \rightarrow (p*)^N = 0 \end{cases}$$

Ecuación [2-64]

La implementación digital del bloque cuadrado se presenta a continuación en la Figura 2-49.

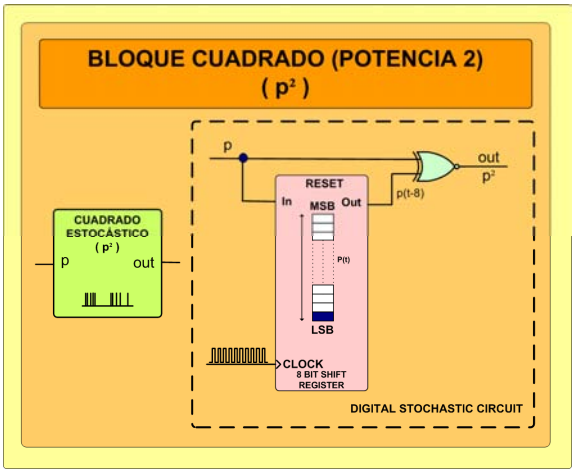

Figura 2-49: Bloque cuadrado (SCSL)

Para la comprobación del correcto funcionamiento del bloque propuesto se han realizado una serie de medidas experimentales variando el valor medio de la distribución asociada a la señal pulsante "p*" entre [-1, +1], los resultados obtenidos se presentan en la Tabla 2-22

Tabla 2-22: Medidas de la función potencia 2 (SCSL)

DATOS DE ENTRADA		EXPERIMENTAL	
P [16bits]	Valor P	OUT [16bits]	Valor OUT
0	-1,0000	65535	1,0000
4096	-0,8750	57761	0,7627
8192	-0,7500	51191	0,5622
12288	-0,6250	45909	0,4010
16384	-0,5000	40615	0,2395
20480	-0,3750	37402	0,1414
24576	-0,2500	34891	0,0648
28672	-0,1250	33227	0,0140
32768	0,0000	33021	0,0077
36864	0,1250	33334	0,0173
40960	0,2500	34854	0,0637
45056	0,3750	37167	0,1342
49152	0,5000	41164	0,2562
53248	0,6250	45627	0,3924
57344	0,7500	51415	0,5691
61440	0,8750	57744	0,7622

A continuación procedemos a representar (Figura 2-50) por una parte los resultados experimentales obtenidos (símbolos) y por otra los predichos por la teoría (línea continua), para finalmente realizar un ajuste por mínimos cuadrados de las medidas experimentales obteniendo así la función de transferencia experimental del bloque.

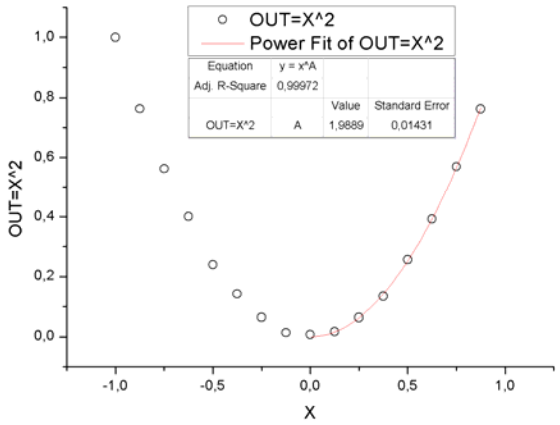

Figura 2-50: Resultados experimentales de la operación potencia 2 (SCSL)

Si comparamos la función de transferencia obtenida de la Figura 2-50 mediante el ajuste por mínimos cuadrados, con la descrita por la teoría, se puede apreciar como (Ecuación [2-65]) ambos resultados son prácticamente idénticos.

$$\text{Experimental} \quad \rightarrow OUT_{EXP} = P^{1,9889}, \text{con una } R^2 = 0,9997$$
$$\text{Teórico} \quad \rightarrow OUT_{TEO} = P^2$$

Ecuación [2-65]

2.2.4.5 La operación división

Para la implementación de la división se ha usado el bloque de *operación inversa* $f^{-1}(p)$ descrito anteriormente en el subapartado de la codificación estocástica clásica sin signo al cual se le ha modificado la etapa de entrada para admitir señales pulsantes con signo. La función $f(h)$ a invertir será la función multiplicación (puerta **XNOR**). Como resultado obtendremos la división entre las señales "p*/q*". Este bloque tan sólo representará correctamente los valores acotados en el rango de representación de la presente codificación [-1, +1]. En la Figura 2-51 se presenta la implementación digital realizada por el bloque divisor con signo.

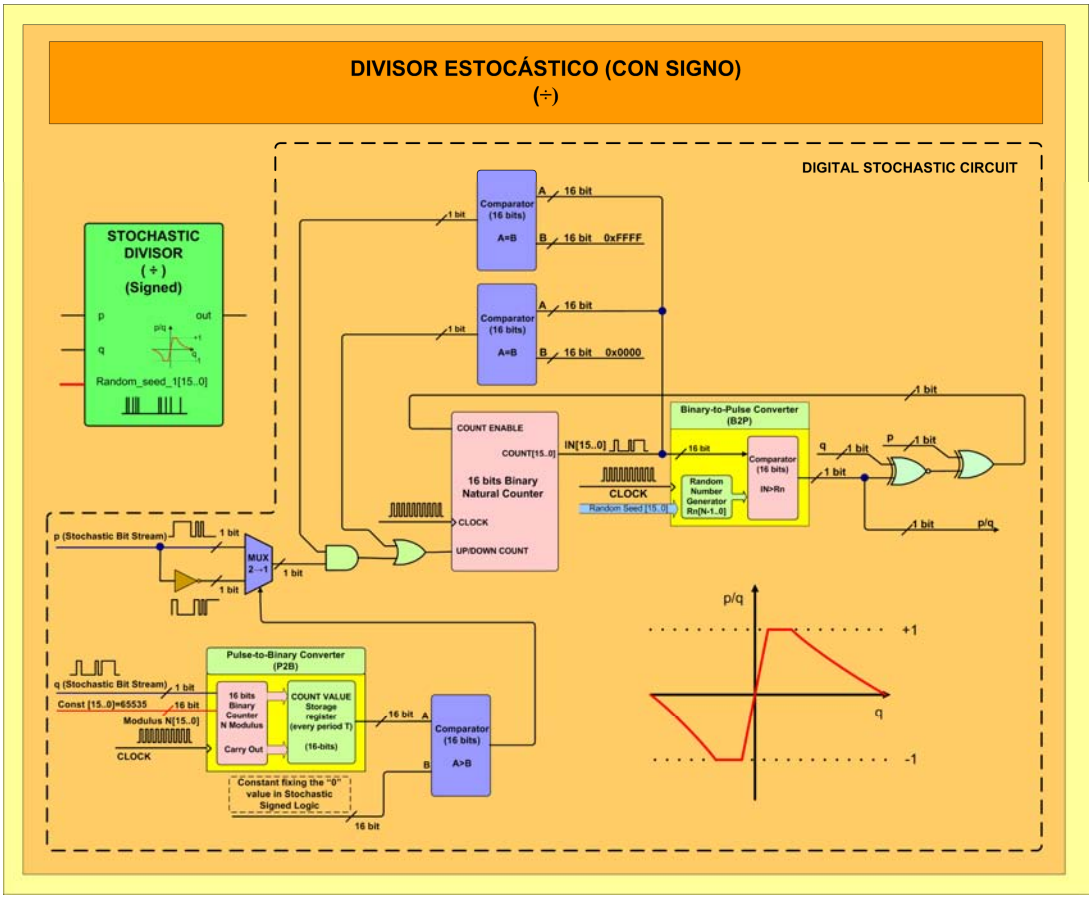

Figura 2-51: Bloque divisor (SCSL)

La operación del bloque divisor estocástico con signo (p*/q*) está basado en el uso de una red de realimentación (que determina el error entre la señal esperada y la señal de entrada) y un contador Up/Down conectado a un bloque *P2B(N)*. La salida de dicho bloque es la entrada del bloque a invertir, en el caso de la división este bloque será una puerta **XNOR** de dos entradas. El proceso de evaluación de la solución es iterativo y tiene un período máximo de evaluación de 2^N ciclos de reloj (donde N se corresponde al número de bits del contador).

No obstante, el bloque divisor necesario para poder operar correctamente en los cuatro cuadrantes tiene que corregir el signo de la señal de entrada “p*” en función del signo de la señal “q*”. Esto implica la necesidad de un bloque *P2B(N)* que se encargue de determinar

la magnitud binaria asociada a la señal “q*”, para posteriormente compararla con el valor del “0” en dicha representación lógica (que seria “0x0FFF” para señales de 16-bits) a fin determinar si el valor medio de la señal “q*” es positivo o negativo. Por lo tanto, la salida de dicho comparador está conectada a un multiplexor (2 entradas a 1 salida) a fin de dejar pasar la señal “p*” sin modificar (si la señal “q*” representa una magnitud positiva) o cambiarla de signo en caso contrario.

Al bloque divisor se le ha incorporado una protección adicional que impide que el contador desborde por el lado negativo “0x0000” y así evitar oscilaciones en la operación de éste. Esto se ha conseguido insertando un comparador a la salida del contador para determinar en qué instante el valor es igual a “0x0000”. La salida del anterior esquema se ha conectado a una puerta **OR**, instalada después de la puerta **AND** de la protección anti-rebaso del valor máximo de conteo “0xFFFF”, descrita anteriormente para el bloque función inversa $f^{-1}(p*)$.

Los resultados obtenidos en el proceso de comprobación del presente bloque de función, se presentan en la Tabla 2-23. Los cuales se han hallado fijando el valor medio de la señal “p*=0,125” y variando el valor medio de la distribución asociada a la señal “q*” en el intervalo [-1, +1].

Tabla 2-23: Medidas de la operación división (SCSL)

DATOS DE ENTRADA				EXPERIMENTAL		TEÓRICO	
P [16bits]	Valor P	Q [16bits]	Valor Q	OUT=P/Q [16bits]	Valor OUT	OUT=P/Q	ERROR
36864	0,1250	0	-1,0000	28726	-0,1233	-0,1250	0,0017
36864	0,1250	2048	-0,9375	28334	-0,1353	-0,1334	0,0019
36864	0,1250	4096	-0,8750	28084	-0,1429	-0,1429	0,0000
36864	0,1250	6144	-0,8125	27652	-0,1561	-0,1539	0,0022
36864	0,1250	8192	-0,7500	27203	-0,1698	-0,1667	0,0031
36864	0,1250	10240	-0,6875	26650	-0,1867	-0,1819	0,0048
36864	0,1250	12288	-0,6250	26033	-0,2055	-0,2001	0,0055
36864	0,1250	14336	-0,5625	25410	-0,2245	-0,2223	0,0022
36864	0,1250	16384	-0,5000	24558	-0,2505	-0,2501	0,0005
36864	0,1250	18432	-0,4375	22929	-0,3002	-0,2858	0,0144
36864	0,1250	20480	-0,3750	21305	-0,3498	-0,3334	0,0164
36864	0,1250	22528	-0,3125	20209	-0,3833	-0,4001	0,0169
36864	0,1250	24576	-0,2500	16684	-0,4908	-0,5002	0,0094
36864	0,1250	26624	-0,1875	10959	-0,6655	-0,6669	0,0014
36864	0,1250	28672	-0,1250	1197	-0,9635	-1,0005	0,0370
36864	0,1250	30720	-0,0625	10	-0,9997	-2,0015	1,0018
36864	0,1250	32768	0,0000	45057	0,3751	97,0000	96,6249
36864	0,1250	34816	0,0625	65523	0,9997	1,9995	0,9998

36864	0,1250	36864	0,1250	65509	0,9992	1,0000	0,0008
36864	0,1250	38912	0,1875	54708	0,6696	0,6667	0,0029
36864	0,1250	40960	0,2500	48956	0,4941	0,5001	0,0060
36864	0,1250	43008	0,3125	45742	0,3960	0,4001	0,0041
36864	0,1250	45056	0,3750	44198	0,3489	0,3334	0,0155
36864	0,1250	47104	0,4375	42410	0,2943	0,2858	0,0085
36864	0,1250	49152	0,5000	41172	0,2565	0,2500	0,0065
36864	0,1250	51200	0,5625	40244	0,2282	0,2223	0,0059
36864	0,1250	53248	0,6250	39159	0,1951	0,2000	0,0050
36864	0,1250	55296	0,6876	38926	0,1880	0,1819	0,0061
36864	0,1250	57344	0,7501	38378	0,1712	0,1667	0,0045
36864	0,1250	59392	0,8126	37958	0,1584	0,1539	0,0045
36864	0,1250	61440	0,8751	37490	0,1441	0,1429	0,0013
36864	0,1250	63488	0,9376	37096	0,1321	0,1334	0,0012

Las medidas de la Tabla 2-23 se representan gráficamente en la Figura 2-52. Los resultados experimentales obtenidos se representan con símbolos, mientras que los predichos por la teoría se representan mediante una línea continua.

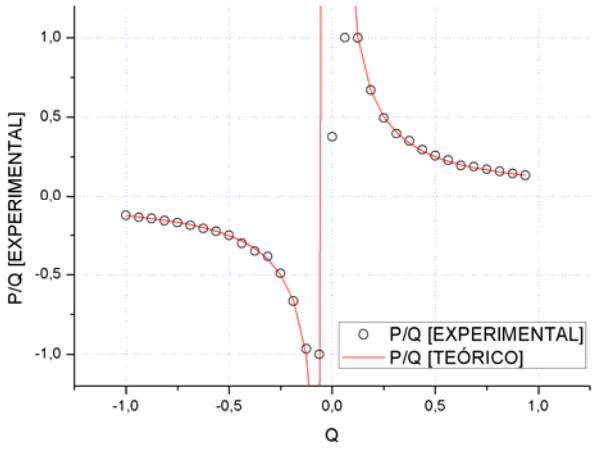

Figura 2-52: Resultados experimentales de la operación división (SCSL)

Como bien se puede apreciar en la Figura 2-52, la función división implementada se comporta idénticamente a lo que predice la teoría. Cabe remarcar que en las zonas fuera del rango de representación de la codificación clásica con signo el bloque satura a sus valores extremos -1 o +1 según el caso. Este bloque se ha implementado fundamentalmente para la implementación de redes neuronales artificiales (*ANN*).

2.2.4.6 La función pseudo tangente hiperbólica

Matemáticamente las funciones hiperbólicas dan la posición de un punto cualquiera "a" de la rama positiva de la hipérbola de ecuación $x^2 - y^2 = 1$ (en un sistema de coordenadas ortonormal), ya que dicho punto "a" pertenece a dicha hipérbola si sus coordenadas verificaran que $\cosh(a)^2 - senh(a)^2 = 1$.

La función tangente hiperbólica de un punto $x \in \Re$ se designa como "*Tanh(x)*" definiéndose como el cociente entre el seno hiperbólico y el coseno hiperbólico de un número real x, tal y como se muestra en la Ecuación [2-66]:

$$\tanh(x) = \frac{senh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} \qquad \text{Ecuación [2-66]}$$

En el marco de la presente tesis la importancia de la función tangente hiperbólica radica en el hecho que es una de las funciones de activación más habituales de las redes neuronales artificiales (*ANN*). La salida de la función de halla acotada asimptóticamente entre [+1, -1]. La función tangente hiperbólica es muy útil para la implementación de metodologías de entrenamiento de redes neuronales artificiales basadas en potenciales decrecientes, ya que la función es diferenciable.

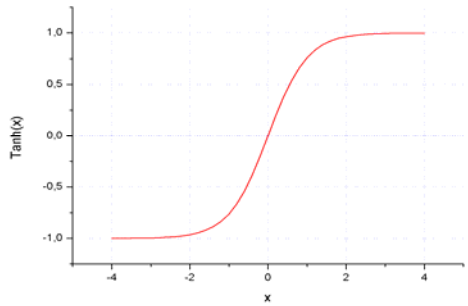

Figura 2-53: Función Tanh(x)

Su popularidad en el mundo de las redes neuronales se debe fundamentalmente a que es usada muy a menudo como función de transferencia (Figura 2-53), ya que dispone de una región lineal cerca del cero y satura asimptóticamente a +1 o -1 en el resto del espacio. Esta

función es muy similar a la función de activación Sigmoidal que se halla acotada asimptóticamente entre 0 y +1.

La importancia de esta función para la implementación de neuronas artificiales nos ha llevado a desarrollar un bloque de función estocástica cuya salida se ajusta bastante bien a la función de transferencia de la tangente hiperbólica real. Hemos llamado al bloque desarrollado pseudo-tangente hiperbólica. La función pseudo tangente hiperbólica "*pseudo-Tanh(p*)*" de una señal pulsante "p*" se ha aproximado mediante el diseño de un bloque cuya salida pulsante se rige por una distribución binomial, que al ser integrada durante un intervalo de tiempo (por ejemplo un período de evaluación (T_{EVAL})) mediante un bloque P2B(N) se obtiene a su salida una función distribución de probabilidad semejante a la función de transferencia de la tangente hiperbólica (Figura 2-54).

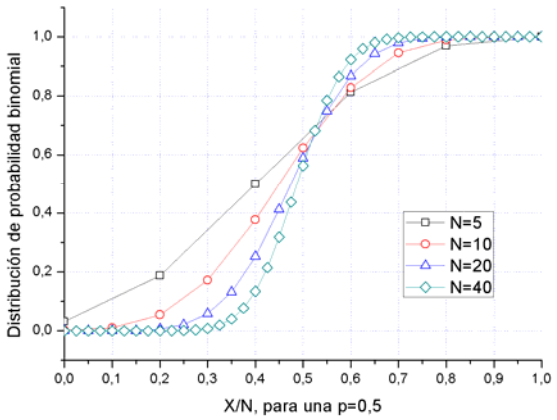

Figura 2-54: Función distribución de probabilidad binomial

Antes de proseguir es conveniente recordar qué se entiende como una distribución de probabilidades binomial en estadística. Ésta es una distribución discreta que mide el número de éxitos de una secuencia de "n" ensayos de Bernoulli, independientes entre sí y con una probabilidad fija "p" de ocurrencia entre los diversos ensayos. Donde un experimento de Bernoulli aquel que se caracteriza por ser dicotómico, es decir que únicamente puede tomar dos posibles valores. Al primer estado se le llama *éxito* con una probabilidad de ocurrencia "p" y al otro *fracaso* con una probabilidad de ocurrencia (1-p). Una distribución binomial es el resultado de repetir "n" veces el experimento de Bernoulli

de forma independiente con objeto de evaluar la probabilidad (Ecuación [2-67]) de obtener un determinado número de éxitos “x”.

$$P(X=k)=f(k)=\binom{n}{k}\cdot p^{k}\cdot(1-p)^{n-k}\quad \forall \mathrm{k}=0,1,...,\mathrm{n} \qquad \text{Ecuación [2-67]}$$

Siendo la función distribución de la probabilidad descrita mediante la Ecuación [2-68].

$$F(x)=\begin{cases}0 & \text{si } x<0\\ \sum_{k=0}^{x}\binom{n}{k}\cdot p^{k}\cdot(1-p)^{n-k} & \text{si } 0\le \mathrm{x}\le \mathrm{n}\\ 1 & \text{si } x\ge \mathrm{n}\end{cases} \qquad \text{Ecuación [2-68]}$$

Por lo tanto, el bloque idóneo para implementar esta función es el *P2B(N)* ya que su salida se rige por una distribución de Bernoulli. Al tratarse de un bloque estocástico éste operará sobre señales pulsantes compuestas por ‘0’ y ‘1’ lógicos, siendo válido afirmar que opera sobre un fenómeno dicotómico. Además puesto que las señales estocásticas han sido generadas por comparación con una secuencia de valores aleatorios independientes entre sí (mediante un bloque *B2P(N)*) podemos afirmar que las medidas son independientes entre sí. Para poder afirmar que la probabilidad “p” de conmutación asociada al valor medio de la distribución pulsante (generada a partir de un bloque *P2B(N)*) a lo largo de los “n” ensayos se mantiene constante, se debe proceder a fijar un período para la realización de dichos “n” ensayos ($n\cdot T_{ensayo}=n\cdot(Gain\cdot T_{Clock})$) mucho menor al período de evaluación (T_{EVAL}) del sistema estocástico. En consecuencia deberá cumplirse la siguiente expresión (Ecuación [2-69]):

$$n\cdot Gain\cdot T_{clock} << T_{EVAL} \qquad \text{Ecuación [2-69]}$$

La salida del bloque *P2B(N)* es una palabra de N-bits que se rige por una distribución binomial, la cual deberá reconvertirse nuevamente en una señal pulsante dicotómica (entre los dos valores de nuestra representación -1 y +1) mediante un bloque comparador de “N” bits. La palabra de salida del bloque se compara con el valor de referencia del cero en la codificación estocástica elegida. Dicha salida pulsante debe ser finalmente integrada mediante un bloque P2B(N) durante al menos un periodo de evaluación (T_{EVAL}), a fin de

obtener la función distribución de probabilidad del sistema. Mediante la Ecuación [2-70] se presenta la función matemática que describe el modo de operación de aproximación propuesta.

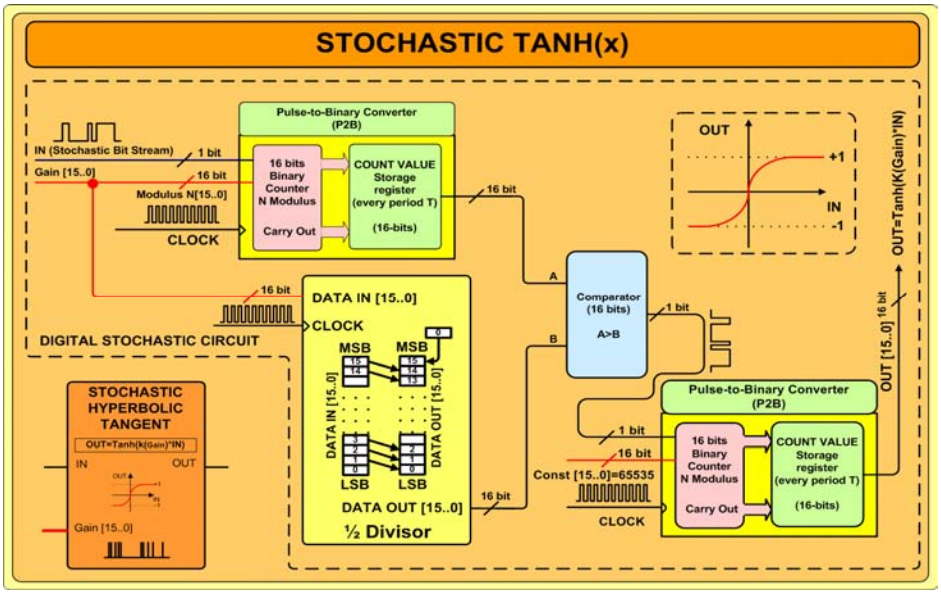

Figura 2-55: Bloque pseudo-tangente hiperbólica (SCSL)

La entrada al bloque es una señal pulsante con un valor medio "k" asociado a la distribución de probabilidad, que implicará una salida del bloque P2B(N) dada por :

$$\begin{cases} T_{muestra} = Gain \cdot T_{Clock} \\ n = 16 bits \\ p = \dfrac{(2P-1)}{2^n} = \dfrac{(2P-1)}{2^{16}} \end{cases}$$

$$\begin{cases} \text{Salida del bloque P2B(N)} \rightarrow f(k) = \binom{Gain}{k} \cdot p^k \cdot (1-p)^{Gain-k} = \\ = \dfrac{Gain!}{k! \cdot (Gain-k)!} \cdot \left(\dfrac{(2P-1)}{2^{16}} \right)^k \cdot \left(1 - \dfrac{(2P-1)}{2^{16}} \right)^{Gain-k} \quad \forall k = 0,1,...,Gain \end{cases}$$

Entonces cada $T_{muestra} = Gain \cdot T_{Clock}$ se evaluará la salida del bloque P2B(N) :

La salida seguirá una distribución binomial $\begin{cases} f(k) \le \dfrac{Gain}{2} \rightarrow out = 0 \rightarrow out^* = -1 \\ f(k) > \dfrac{Gain}{2} \rightarrow out = 1 \rightarrow out^* = +1 \end{cases}$

Finalmente la salida del último bloque P2B(N) será :

$$F(x) = \sum_{u=0}^{x} \binom{n}{u} \cdot f(x)^u \cdot (1 - f(k))^{n-u} \quad si\ 0 \le x \le n$$

Ecuación [2-70]

El diseño digital de la función pseudo tangente hiperbólica "*Tanh(k·p*)*" se presenta en la Figura 2-55; el cual se compone internamente de un bloque convertidor *P2B(N)* de 16-bits, un circuito divisor módulo 2 de 16-bits, un comparador (A>B) de 16-bits y un segundo bloque *B2P(N)* para integrar la salida del comparador.

La forma de operación del circuito se basa en un bloque P2B(N) que cuenta el número de pulsos estocásticos a nivel alto '1' recibidos en la entrada **IN** durante un determinado intervalo de ciclos de reloj igual al valor binario asignado al parámetro **Gain** (palabra de 16-bits). Finalmente se compara si la palabra de 16-bits de la salida del bloque *P2B(N)* "**A**" es mayor que la mitad del valor del parámetro **Gain**, es decir **Gain/2** que se corresponde con el punto donde se ubica el "0" en la codificación estocástica con signo. Para una correcta operación del bloque, el valor asignado al parámetro **Gain** debe ser siempre impar (a fin de asegurar que se encuentren el mismo número de estados por encima y por debajo del estado equivalente al cero, asegurando así el paso por "0" de la salida de la función cuando "IN*=0"). Su valor debe ser relativamente pequeño menor a "**0x00FF"**, a fin de conseguir que la salida del bloque se ajuste a una distribución de Bernoulli sobre la que se basa esta implementación.

Para comprobar el correcto funcionamiento de la implementación realizada para dicho bloque hemos procedido a realizar una serie de medidas experimentales (Tabla 2-24) que

han consistido en fijar el parámetro "**Gain=5"** y variar el valor medio de la señal pulsante de entrada "IN*" entre [-1, +1].

Tabla 2-24: Medidas de la función pseudo tangente hiperbólica (SCSL)

DATOS DE ENTRADA		EXPERIMENTAL		TEÓRICO	
IN [16bits]	Valor IN	OUT [16bits]	Valor OUT	OUT=tanh(k·x) k=1,6590	ERROR
0	-1,0000	0	-1,0000	-0,9301	0,0699
4096	-0,8750	2040	-0,9377	-0,8960	0,0417
8192	-0,7500	4032	-0,8770	-0,8467	0,0303
12288	-0,6250	7212	-0,7799	-0,7767	0,0033
16384	-0,5000	10484	-0,6801	-0,6802	0,0002
20480	-0,3750	16086	-0,5091	-0,5526	0,0435
24576	-0,2500	21470	-0,3448	-0,3925	0,0477
28672	-0,1250	28668	-0,1251	-0,2045	0,0793
32768	0,0000	32973	0,0063	0,0000	0,0063
36864	0,1250	36921	0,1267	0,2045	0,0777
40960	0,2500	43615	0,3310	0,3925	0,0615
45056	0,3750	49389	0,5072	0,5526	0,0454
49152	0,5000	55105	0,6817	0,6802	0,0015
53248	0,6250	57459	0,7535	0,7767	0,0231
57344	0,7500	61275	0,8700	0,8467	0,0233
61440	0,8750	63476	0,9371	0,8960	0,0411

Los resultados experimantales presentados en la Tabla 2-24 se representan gráficamente en la Figura 2-56, a fin de mostrar la semejanza entre la salida del bloque desarrollado (símbolos) y la función teórica ajustada (línea continua).

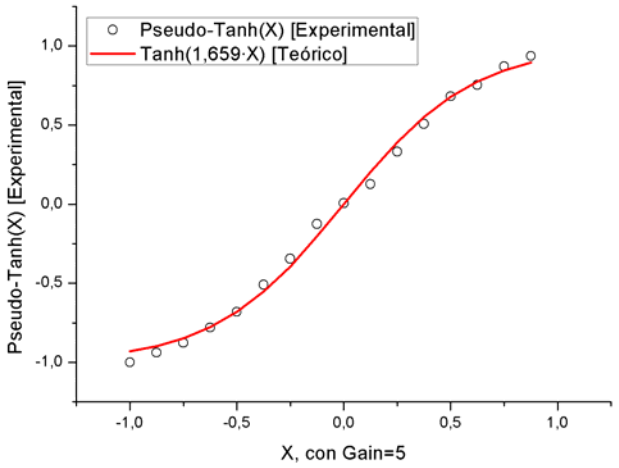

Figura 2-56: Resultados experimentales de la función *pseudo tangente hiperbólica* (SCSL)

Como bien puede apreciarse tanto en la Figura 2-56, así como en la última columna de la Tabla [2-17], la aproximación realizada ajusta bastante bien la tendencia de la función *Tanh(x)* aunque en determinados puntos se minusvalora y en otros se sobrevalora el valor teórico de ésta.

Con objeto de obtener una relación matemática que nos relacione el valor del parámetro de configuración del bloque "Gain" con el valor de la ganancia de salida del bloque *pseudo Tanh(x)*, hemos procedido a realizar un conjunto de medidas adicionales a fin de determinar el valor experimental de la ganancia "k" a la cual se corresponden. Los resultados obtenidos se presentan en la Tabla [2-25]:

Tabla 2-25: Ganancia de la función pseudo tangente hiperbólica (SCSL)

Parámetro Gain [16bits]	Ganancia del la función Tanh(k·x)
3	1,4400
5	1,6590
7	1,8900
9	2,1190
11	2,2300
21	3,0000
31	3,6000
41	3,9800
61	5,1400

A partir de los valores de "k" obtenidos para los diferentes valores de "Gain" representados en la Tabla [2-25] hemos procedido a realizar un ajuste lineal a fin de evaluar la relación existente entre ambas. Dicha relación se presenta mediante la Ecuación [2-71]:

$$k = 0{,}0624 \cdot \text{Gain} + 1{,}4748\,,\ \text{Gain} \in \mathbb{N},\ \text{con una } R^2 = 0{,}9851 \qquad \text{Ecuación [2-71]}$$

2.2.5 La codificación estocástica extendida (UESL)

En este apartado se presenta una nueva metodología que hemos desarrollado y que se basa en el uso de la razón entre dos señales estocásticas clásicas. A esta metodología la hemos denominado “codificación estocástica extendida”. Esta nueva variante de codificación, al igual que la codificación estocástica clásica, puede adaptarse para representar magnitudes con signo y sin signo con la salvedad que ahora en esta codificación se podrá representar cualquier valor real.

En el caso de la codificación estocástica extendida sin signo que se aborda a continuación, el espacio de representación son los números reales positivos $\Re^+$. En esta variante de la lógica las magnitudes se representan mediante dos señales pulsantes. Cada magnitud “X” vendrá representada por dichas señales como la razón entre los dos valores medios de activación de las señales “p” y “q” que se hallan definidas en el intervalo [0, +1]. De esta forma se podrán representar valores entre [0, +∞]. Las magnitudes así codificadas mediante la razón “p/q” serán proporcionales a la razón de los valores binarios “P” y ”Q” a partir de los cuales se han obtenido las anteriores señales pulsantes. Este hecho se presenta en la Ecuación [2-72].

$$\begin{cases} p \in |0,+1| \\ q \in |0,+1| \end{cases} \rightarrow E(X) = E\left(\frac{p}{q}\right) = \frac{P/2^n}{Q/2^n} = \frac{P}{Q}, X \in [0,+\infty]$$

$$\lim_{p\to\infty} E\left(\frac{p}{q}\right) \approx \infty, \lim_{p\to 0} E\left(\frac{p}{q}\right) \approx 0, \lim_{q\to\infty} E\left(\frac{p}{q}\right) \approx 0, \lim_{q\to 0} E\left(\frac{p}{q}\right) \approx \infty \qquad \text{Ecuación [2-72]}$$

$$\lim_{\substack{q\to\infty \\ q\to\infty}} E\left(\frac{p}{q}\right) \approx Ind, \lim_{\substack{q\to 0 \\ q\to 0}} E\left(\frac{p}{q}\right) \approx Ind$$

Al igual que en la lógica estocástica clásica (en cualquiera de sus variantes) la conversión de dichas magnitudes binarias en pulsantes se realizará mediante una pareja de bloques conversores *Binary-to-Pulse Converter B2P(N)* de 16-bits (que se compone internamente de un bloque generador de números aleatorios y un comparador) idénticos a los usados hasta el momento.

En esta representación se debe procurar codificar las magnitudes mediante los valores medios de conmutación que han de ser lo más grandes posibles para que las señales “p” y “q” no puedan proporcionar resultados con un margen de error demasiado alto.

Un ejemplo de ello es la codificación de la magnitud X=5 mediante dos magnitudes binarias “P=5” Y “Q=1” (Ecuación [2-73]), de tal forma que X=P/Q=5 pero con unos ratios de conmutación de cinco pulsos a nivel alto en 2^n ciclos de reloj y un pulso a nivel alto cada “2^n” ciclos de reloj. Para este caso es muy probable que la mayor parte de bloques estocásticos no operen correctamente.

$$\begin{cases} P = \dfrac{5}{2^n} \\ Q = \dfrac{1}{2^n} \end{cases} \rightarrow X = \frac{P}{Q} = \frac{5 \diagup 2^n}{1 \diagup 2^n} = \frac{5}{1} = 5 \qquad \text{Ecuación [2-73]}$$

La forma óptima de codificar la magnitud X=5 mediante dos magnitudes binarias “P=50000” y “Q=10000” (Ecuación [2-74]) (para una codificación de la información mediante *B2P(N)* de 16-bits) consiste en maximizar los valores medios de conmutación.

$$\begin{cases} P = \dfrac{50000}{2^{16}} \\ Q = \dfrac{10000}{2^{16}} \end{cases} \rightarrow X = \frac{P}{Q} = \frac{50000 \diagup 2^{16}}{10000 \diagup 2^{16}} = \frac{50000}{10000} = 5 \qquad \text{Ecuación [2-74]}$$

Por otra parte, para sistemas de computación extensos habrá que ir regenerando las señales de forma periódica ya que los ratios de conmutación para las señales “p” y “q” resultantes se degradan en función de qué bloque de computación se utilice.

Por lo tanto, con esta nueva notación las operaciones aritméticas se realizarán de forma similar a como se realizaban en la codificación estocástica clásica sin signo, con la salvedad que ahora deberemos operar con el numerador y el denominador. Aunque con esta codificación podremos representar valores entre [0,+∞] sigue prevaleciendo el problema de sólo poder representar valores positivos. Por otro lado las operaciones aritméticas se realizarán de forma similar a como se realizaban en la codificación estocástica clásica sin

signo, donde por ejemplo la multiplicación entre dos señales pulsantes se realiza mediante puertas **AND** de dos entradas.

Desde el punto de vista estadístico una consecuencia de esta nueva representación es que el error de la conversión vendrá dado por la Ecuación [2-75]. En ésta podemos apreciar cómo el error pasa a depender del valor de las magnitudes binarias P y Q con que se ha codificado X, y no es independiente de ellas como sucedía hasta el momento.

$$\begin{cases} Error(P) = \pm\dfrac{1bit}{2^n} \rightarrow [0,+1] \\ Error(Q) = \pm\dfrac{1bit}{2^n} \rightarrow [0,+1] \\ Q = \dfrac{Dato1}{2^n} \\ P = \dfrac{Dato2}{2^n} \end{cases} \Rightarrow [0,+\infty] \rightarrow X = \frac{P}{Q} \rightarrow Error(X) = \left|\frac{1}{Q}\right| \cdot Error(P) + \left|\frac{P}{Q^2}\right| \cdot Error(Q)$$

Ecuación [2-75]

A esta nueva codificación estocástica la hemos denominado *codificación estocástica extendida sin signo* o "*Unsigned Extended Stochastic Logic*" (UESL). Ésta ha sido desarrollada y evaluada como un paso intermedio hacia otras representaciones mucho más interesantes como pueden ser la lógica estocástica extendida con signo, que se presenta en el siguiente apartado. Cabe remarcar que la codificación extendida sin signo funcionará correctamente cualquier circuito estocástico que haya sido diseñado para operar con la lógica estocástica clásica, no obstante su rango de operación estará limitado entre [0, +1], ya que tan sólo podrán operar sobre una de las dos señales presentes en dicha codificación. Esto será muy útil en algunos casos debido a la imposibilidad de implementar la función deseada como la razón entre dos señales pulsantes, por lo tanto tan sólo se dispondrá de la función estocástica clásica sin signo.

El principal uso que le hemos dado a la codificación extendida sin signo ha sido la implementación de sistemas de reconocimiento de patrones, basados en la aproximación estadística.

Las funciones aritméticas/matemáticas desarrolladas se presentan a lo largo de los siguientes subapartados con sus correspondientes implementaciones digitales y medidas experimentales para cada una de ellas.

2.2.5.1 La operación suma

La implementación de la operación suma entre dos magnitudes X e Y, codificadas mediante la razón entre dos magnitudes binarias "X=P/Q" y "Y=R/S", son a su vez representadas mediante la razón entre los valores medios de conmutación de las señales pulsantes equivalentes "x=p/q" y "y=r/s". En este caso en particular la implementación de la suma será algo más compleja que en el caso de la lógica estocástica clásica, ya que se debe operar las señales para obtener las resultantes del numerador y del denominador (Ecuación [2-76]):

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow Z = X + Y = \frac{P}{Q} + \frac{R}{S} = \frac{(P \cdot S + R \cdot Q)}{Q \cdot S} \qquad \text{Ecuación [2-76]}$$

La operación suma realizada mediante esta codificación tiene la ventaja de permitirnos implementar la suma natural de dos magnitudes puesto que hasta el momento sólo era posible evaluar la suma en valor medio. En este caso, el factor de ponderación presente en la codificación estocástica clásica (en el numerador) podrá ser contrarestado en el denominador. De esta forma, la suma de los productos cruzados "p·s+r.q" en el numerador la realizaremos mediante un multiplexor (2→1) gobernado por una señal de control de la activación "c" que forzosamente deberá hallarse descorrelacionada con respecto de las otras cuatros señales operadas; mientras que los productos cruzados entre las señales pulsantes se realizarán mediante sendas puertas **AND** de dos entradas. En el denominador tendremos el producto de los denominadores de las razones sumadas, multiplicadas a su vez por la señal de control de la activación "c" del numerador (c·p·s) a fin de compensar dicho factor en la suma del numerador. Por lo tanto mediante esta estrategia se consigue que la salida del bloque sea la suma natural de las magnitudes "X" e "Y", tal y como se demuestra en la Ecuación [2-77]:

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow OUT = (X+Y) = \left(\frac{P}{Q} + \frac{R}{S}\right) = \left(\frac{P \cdot S + R \cdot Q}{Q \cdot S}\right) \rightarrow E(out(p,q,r,s,c)) =$$

$$= E\left(\frac{c \cdot (p \cdot s) + (1-c) \cdot (r \cdot q)}{c \cdot q \cdot s}\right) = \frac{P(c) \cdot P(p) \cdot P(s) + P(1-c) \cdot P(r) \cdot P(q)}{P(c) \cdot P(q) \cdot P(s)} = \frac{C \cdot P \cdot S / 2^{3 \cdot n} + (1-C) \cdot R \cdot Q / 2^{3 \cdot n}}{C \cdot Q \cdot S / 2^{3 \cdot n}} =$$

$$= \frac{C \cdot P \cdot S + (1-C) \cdot R \cdot Q}{C \cdot Q \cdot S} \underset{C=0,5}{=} \frac{0,5 \cdot P \cdot S + 0,5 \cdot R \cdot Q}{0,5 \cdot Q \cdot S} = \frac{P \cdot S + R \cdot Q}{Q \cdot S}$$

Ecuación [2-77]

Para asegurar que la suma del numerador entre "p·s" y "r·q" esté ponderada se procederá a elegir un valor de la señal de control de activación "c" tal que asegure que ambas señales dispongan del mismo tiempo de evaluación en un determinado período de integración. Por lo tanto se requerirá que "c=0,5". De esta forma se asegura que el valor máximo de "p·s" y "q·r" cumpla con el límite superior de "+1" →0,5·(1·1+1·1)=1. Mientras que para el valor mínimo de "p·s" y "q·r" cumplirá con el límite inferior "0"→0,5·(0·0+0·0)=0.

Los valores medios que hallaremos contenidos en los acumuladores los dos bloques *P2B(N)* (uno para el numerador y otro para el denominador) después de "$N_{ciclos\_evaluación}$" serán sendas variables aleatorias con valores medios descritos por la Ecuación [2-78]:

$$\begin{cases} \overline{out}_{\text{numerador}} = N_{\text{ciclos\_evaluación}} \cdot \dfrac{0,5 \cdot (P \cdot S + R \cdot Q)}{2^{3 \cdot n}} \\ \overline{out}_{\text{denominador}} = N_{\text{ciclos\_evaluación}} \cdot \dfrac{0,5 \cdot (Q \cdot S)}{2^{3 \cdot n}} \end{cases}$$

Ecuación [2-78]

La implementación digital encargada de realizar esta operación se presenta en la Figura 2-57. De la implementación presentada se desprende que las señales resultantes del numerador y del denominador están correlacionadas al haberse usado la misma señal "c" para operar con sendas señales. Por lo tanto su salida deberá descorrelacionarse mediante elementos de memoria (registro de desplazamiento).

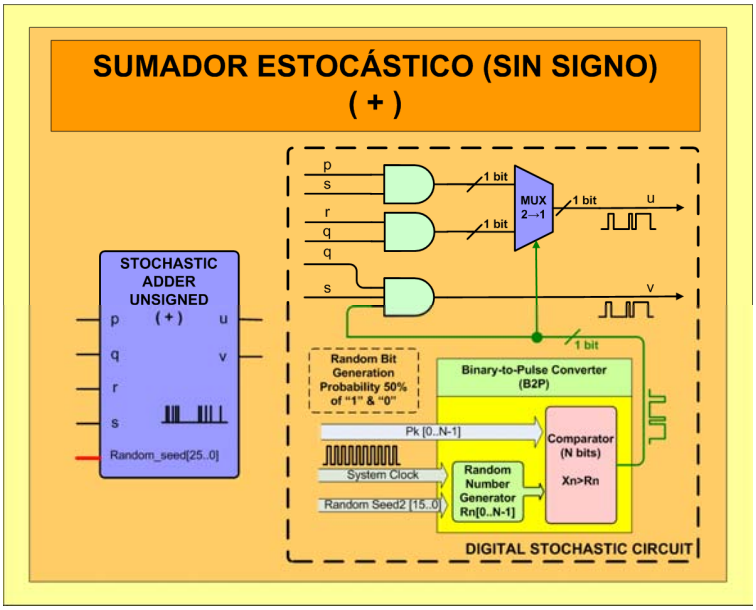

Figura 2-57: Bloque suma natural (UESL)

A fin de comprobar el correcto funcionamiento de la implementación realizada del circuito sumador natural, se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-26. Éstas se han hallado fijando el valor medio de la señal y "y=1,25" y variando el valor medio de la señal asociada a la razón "x" entre [0,+2].

Tabla 2-26: Medidas de la operación suma natural (UESL)

DATOS DE ENTRADA						EXPERIMENTAL			TEÓRICO	
P [16bits]	Q [16bits]	Valor X=P/Q	R [16bits]	S [16bits]	Valor Y=R/S	U [16bits]	V [16bits]	Valor OUT (X+Y)	X+Y	ERROR
0	0	Ind	0	0	Ind	0	0	Ind	Ind	Ind
0	8192	0	0	8192	0	0	513	0	0	0
4096	8192	0,5	10240	8192	1,25	876	510	1,7176	1,75	0,0324
6144	8192	0,75	10240	8192	1,25	1026	515	1,9922	2	0,0078
8192	8192	1	10240	8192	1,25	1167	494	2,3623	2,25	0,1123
10240	8192	1,25	10240	8192	1,25	1259	511	2,4638	2,5	0,0362
12288	8192	1,5	10240	8192	1,25	1408	500	2,8160	2,75	0,0660
14336	8192	1,75	10240	8192	1,25	1607	542	2,9649	3	0,0351
16384	8192	2	10240	8192	1,25	1641	494	3,3219	3,25	0,0719

Procedemos a representar gráficamente (Figura 2-58) los datos obtenidos en la Tabla 2-26, donde los resultados experimentales obtenidos se representan con símbolos y los valores predichos por la teoría se representan mediante una línea continua.

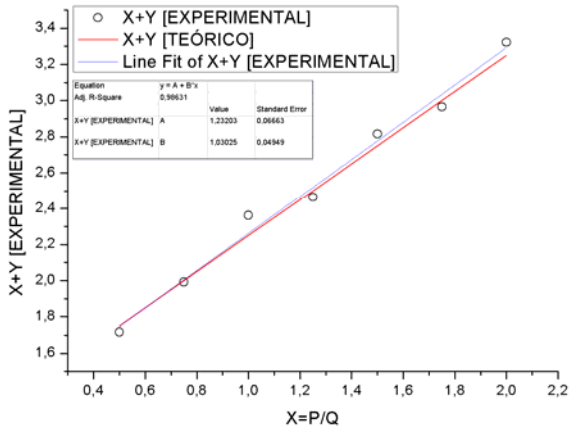

Figura 2-58: Resultados experimentales de la operación suma natural (UESL)

Si a continuación procedemos a comparar la función de transferencia obtenida de la Figura 2-58 mediante un ajuste por mínimos cuadrados con la descrita por la teoría para la presente configuración del circuito estocástico, podremos apreciar (Ecuación [2-79]) cómo ambas son muy semejantes.

$$\begin{aligned} &\text{Experimental} && \rightarrow OUT_{EXP} = (1{,}0303 \cdot X + 1{,}2320), \text{ con una } R^2 = 0{,}9863 \\ &\text{Teórico} && \rightarrow OUT_{TEO} = (1 \cdot X + 1{,}25) \end{aligned}$$

Ecuación [2-79]

2.2.5.2 La operación resta en valor absoluto

La operación aritmética resta en valor absoluto entre dos magnitudes binarias “X=P/Q” e “Y=R/S” será similar a la forma con la que se realiza la suma. De todas formas se requerirá operar las diferentes señales presentes para obtener el numerador y el denominador de la magnitud resultante “Z” de la operación resta (Ecuación [2-80]).

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow Z = X - Y = \frac{P}{Q} - \frac{R}{S} = \frac{(P \cdot S - R \cdot Q)}{Q \cdot S} \qquad \text{Ecuación [2-80]}$$

Al igual que sucedía en la codificación estocástica clásica sin signo, tan sólo podremos implementar la resta en valor absoluto y no la resta natural. Ya que en esta codificación no se pueden representar magnitudes negativas.

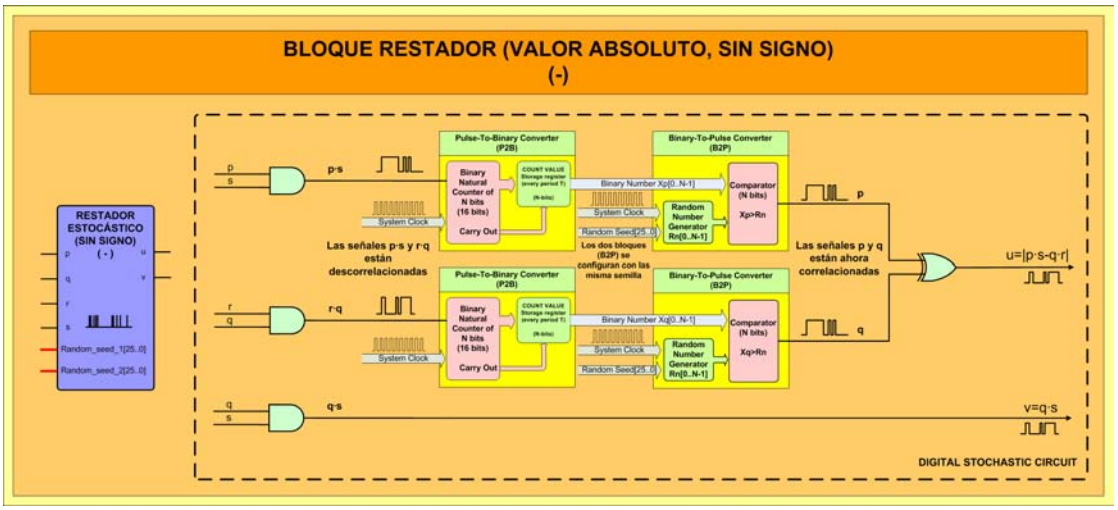

Figura 2-59: Bloque restador en valor absoluto (UESL)

La implementación realizada del bloque restador se muestra en la Figura 2-59, en ella apreciamos cómo los productos entre los numeradores y los denominadores de las señales (p·s) y (r·q) se han implementado mediante sendas puertas **AND** de dos entradas; cuyas señales de salida a su vez están conectadas a dos bloques *P2B(N)* que se encargan de obtener el valor medio de conmutación de la señal inyectada. Posteriormente se vuelven a convertir en dos señales pulsantes mediante una pareja de bloque *B2P(N)* que usan la misma semilla. El objetivo es que ambas señales estén correlacionadas entre sí y así al operarlas a través de una puerta **XOR** obtengamos la resta en valor absoluto entre las dos señales. La señal del denominador se obtendrá mediante el producto (puerta **AND**) de los denominadores de las magnitudes codificadas. Si procedemos a evaluar matemáticamente mediante la Ecuación [2-81] podemos demostrar que teóricamente la implementación realizada implementa la función resta en valor absoluto de las magnitudes “X” y “Y”.

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow OUT = (X - Y) = \left(\frac{P}{Q} + \frac{R}{S}\right) = \left(\frac{P \cdot S - R \cdot Q}{Q \cdot S}\right) \rightarrow E(out(p,q,r,s)) =$$

$$= E\left(\frac{(p \cdot s)' \oplus (r \cdot q)'}{q \cdot s}\right) = \frac{(P(p) \cdot P(s))' \oplus (P(r) \cdot P(q))'}{P(q) \cdot P(s)} = \frac{P(p) \cdot P(s) - P(r) \cdot P(q)}{P(q) \cdot P(s)} =$$

$$= \frac{P \cdot S/2^{2 \cdot n} + R \cdot Q/2^{2 \cdot n}}{Q \cdot S/2^{2 \cdot n}} = \frac{P \cdot S - R \cdot Q}{Q \cdot S}$$

$( )' \rightarrow$ Señales correlacionadas temporalmente entre si

Ecuación [2-81]

Para comprobar experimentalmente el correcto funcionamiento de la implementación desarrollada se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-27. En éstas se ha fijado la magnitud "Y=2" y se ha procedido a variar el valor de la magnitud "X" en el intervalo [0, +7].

Tabla 2-27: Medidas de la operación resta en valor absoluto (UESL)

DATOS DE ENTRADA						EXPERIMENTAL			TEÓRICO	
P [16bits]	Q [16bits]	Valor X=P/Q	R [16bits]	S [16bits]	Valor Y=R/S	U [16bits]	V [16bits]	OUT (X+Y)	X+Y	ERROR
0	0	Ind	0	0	Ind	0	0	Ind	Ind	Ind
0	8192	0	0	8192	0	2	65535	0,0000	0	0,0000
8192	8192	1	8192	8192	1	196	65351	0,0030	0	0,0030
16384	8192	2	16384	8192	2	129	16576	0,0078	0	0,0078
24576	8192	3	16384	8192	2	10772	10871	0,9909	1	0,0091
32768	8192	4	16384	8192	2	16691	8272	2,0178	2	0,0178
40960	8192	5	16384	8192	2	20013	6753	2,9636	3	0,0364
49152	8192	6	16384	8192	2	21716	5541	3,9191	4	0,0809
57344	8192	7	16384	8192	2	23099	4672	4,9441	5	0,0559

Si procedemos a graficar (Figura 2-60) los resultados experimentales (símbolos) obtenidos a la vez que los descritos por la teoría (línea continua) (Tabla 2-27), podremos comprobar la idoneidad de la implementación realizada.

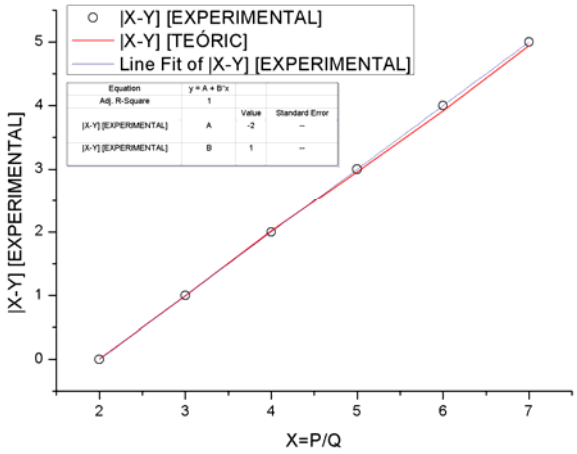

Figura 2-60: Resultados experimentales de la operación resta en valor absoluto (UESL)

Si finalmente comparamos la función de transferencia obtenida con el ajuste por mínimos cuadrados de los datos experimentales, con la función teóricamante predicha comprobaremos cómo ambas son idénticas (Ecuación [2-82]).

$$\begin{aligned} &\text{Experimental} \quad \rightarrow OUT_{EXP} = (1 \cdot X - 2), \text{ con una } R^2 = 1 \\ &\text{Teórico} \quad \rightarrow OUT_{TEO} = |1 \cdot X - 2| \end{aligned} \qquad \text{Ecuación [2-82]}$$

2.2.5.3 La operación multiplicación

La multiplicación en la presente codificación es idéntica a la usada en la codificación estocástica clásica con la salvedad que ahora se deben operar numerador y denominador por separado, por todo ello la operación multiplicación vendrá regida por la Ecuación [2-83]:

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow Z = X \cdot Y = \frac{P}{Q} \cdot \frac{R}{S} = \frac{P \cdot R}{Q \cdot S} \qquad \text{Ecuación [2-83]}$$

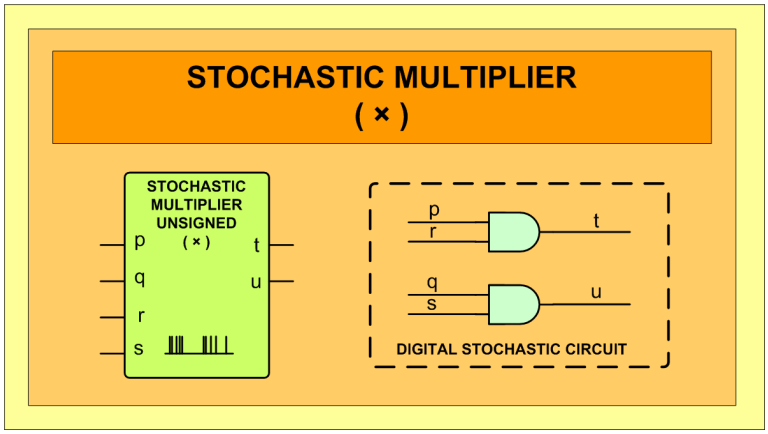

Figura 2-61: Bloque multiplicador (UESL)

La implementación de este bloque (Figura 2-61) se basa en el uso de dos puertas **AND** encargadas de operar los valores medios de conmutación de las distribuciones de probabilidad de las señales (p·r) y (q·s), de tal forma que la salida se regirá por la ecuación [2-84]:

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow OUT = X \cdot Y = \left(\frac{P}{Q} \cdot \frac{R}{S} \right) \rightarrow E\left(out\left(p,q,r,s\right)\right) = E\left(\frac{p \cdot r}{q \cdot s} \right) = \frac{P\left(p\right) \cdot P\left(r\right)}{P\left(q\right) \cdot P\left(s\right)} = \frac{P \cdot R / 2^{2 \cdot n}}{Q \cdot S / 2^{2 \cdot n}} = \frac{P \cdot R}{Q \cdot S}$$

Ecuación [2-84]

Los resultados de las medidas experimentales realizadas sobre el circuito multiplicador estocástico se presentan en la Tabla 2-28. En ellas se ha fijado el valor de la magnitud "Y=0,5" y se ha procedido a variar el valor de "X" en el intervalo [0, +7,5].

Tabla 2-28: Medidas de la operación multiplicación (UESL)

DATOS DE ENTRADA						EXPERIMENTAL			TEÓRICO	
P [16bits]	Q [16bits]	Valor X=P/Q	R [16bits]	S [16bits]	Valor Y=R/S	U [16bits]	V [16bits]	OUT (U/V)	X·Y (Teo)	ERRO R
0	8192	0,0000	8192	16384	0,5	0	2003	0,0000	0,0000	0,0000
4092	8192	0,4995	8192	16384	0,5	499	2052	0,2432	0,2498	0,0066
8184	8192	0,9990	8192	16384	0,5	1008	2017	0,4998	0,4995	0,0002
12276	8192	1,4985	8192	16384	0,5	1548	2028	0,7633	0,7493	0,0140
16368	8192	1,9980	8192	16384	0,5	2100	2042	1,0284	0,9990	0,0294

20460	8192	2,4976	8192	16384	0,5	2583	2096	1,2323	1,2488	0,0164
24552	8192	2,9971	8192	16384	0,5	3061	2080	1,4716	1,4985	0,0269
28644	8192	3,4966	8192	16384	0,5	3653	2090	1,7478	1,7483	0,0004
32736	8192	3,9961	8192	16384	0,5	4096	2131	1,9221	1,9980	0,0759
36828	8192	4,4956	8192	16384	0,5	4494	1991	2,2572	2,2478	0,0094
40920	8192	4,9951	8192	16384	0,5	5242	2115	2,4785	2,4976	0,0191
45012	8192	5,4946	8192	16384	0,5	5557	2091	2,6576	2,7473	0,0897
49104	8192	5,9941	8192	16384	0,5	5960	1929	3,0897	2,9971	0,0926
53196	8192	6,4937	8192	16384	0,5	6676	1998	3,3413	3,2468	0,0945
57288	8192	6,9932	8192	16384	0,5	7181	2005	3,5815	3,4966	0,0850
61380	8192	7,4927	8192	16384	0,5	7807	2066	3,7788	3,7463	0,0325

A continuación representamos (Figura 2-62) los resultados experimentales (símbolos) obtenidos junto los descritos de los descritos por la teoría (línea continua) (Tabla 2-28). A su vez evaluamos la función de transferencia experimental mediante un ajuste lineal sobre los datos medidos.

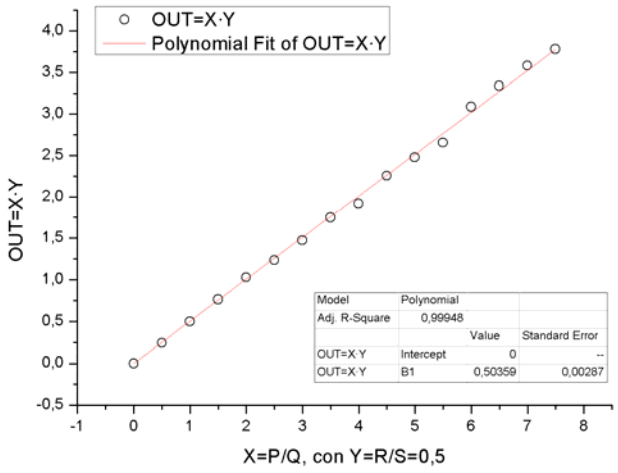

Figura 2-62: Resultados experimentales de la función multiplicación (UESL)

Comparando la función de transferencia obtenida a partir de las medidas experimentales (Figura 2-62) con la función teóricamente predicha para dicho bloque podemos afirmar que ambas funciones (Ecuación [2-85]) son prácticamente idénticas.

$$\begin{aligned} &\text{Experimental} \quad \rightarrow OUT_{EXP} = 0{,}5036 \cdot X, \text{ con una } R^2 = 0{,}9995 \\ &\text{Teórico} \quad \rightarrow OUT_{TEO} = 0{,}5000 \cdot X \end{aligned} \qquad \text{Ecuación [2-85]}$$

2.2.5.4 La operación potenciación (P^N)

La potenciación, al igual que la multiplicación, se realiza de forma idéntica que para la lógica estocástica sin signo (mediante puertas **AND** y registros de memoria para descorrelacionar temporalmente las señales de sí mismas). La diferencia es que ahora deben usarse dos bloques de potenciación: una para el numerador y otro para el denominador. Como ejemplo de la funcionalidad de esta función en lógica estocástica extendida se ha procedido a implementar un bloque (Figura 2-63) que evalúe el cuadrado de una magnitud "X".

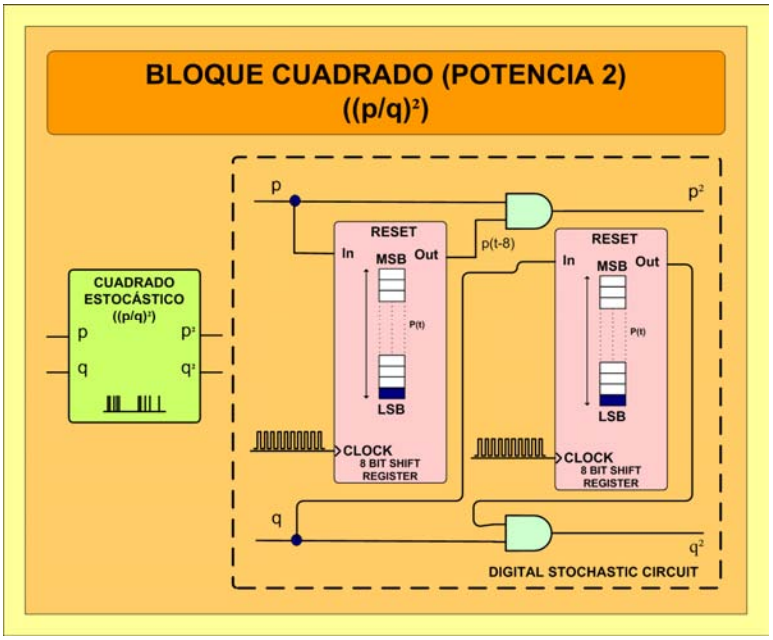

Figura 2-63: Bloque cuadrado (UESL)

Como comprobación del circuito implementado se ha procedido a realizar una serie de medidas experimentales que se muestran en la Tabla 2-29, que han consistido en ir variando la magnitud "X" en el intervalo [0, +7].

Tabla 2-29: Medidas de la operación cuadrado (UESL)

DATOS DE ENTRADA			EXPERIMENTAL			TEÓRICO	
P [16bits]	Q [16bits]	Valor X=P/Q	U [16bits]	V [16bits]	OUT (U/V)	X^2 (Teórica)	ERROR
0	0	Ind	0	0	Ind	Ind	Ind

0	8192	0,0000	0	1049	0,0000	0	0,0000
8192	8192	1,0000	1029	1010	1,0188	1	0,0188
16384	8192	2,0000	4159	1035	4,0184	4	0,0184
24576	8192	3,0000	9362	1060	8,8321	9	0,1679
32768	8192	4,0000	16479	1020	16,1559	16	0,1559
40960	8192	5,0000	25751	1019	25,2709	25	0,2709
49152	8192	6,0000	36985	1031	35,8729	36	0,1271
57344	8192	7,0000	49884	1015	49,1468	49	0,1468

Si representamos (Figura 2-64) los datos experimentales obtenidos en la Tabla 2-29 (símbolos) y los valores teóricos (línea continua) podremos apreciar la tendencia de ambos.

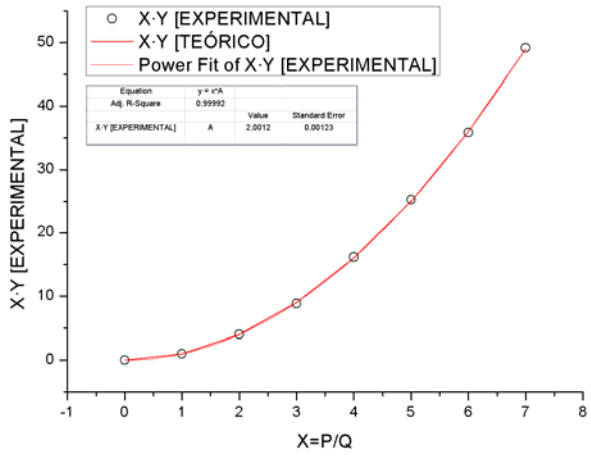

Figura 2-64: Resultados experimentales de la operación cuadrado (UESL)

Como puede apreciarse en la Figura 2-64 la función de transferencia obtenida es idéntica a la predicha por la teoría, tal y como se muestra en la Ecuación [2-86].

$$\text{Experimental} \quad \rightarrow OUT_{EXP} = X^{2,0012}, \text{ con una } R^2 = 0,9999$$
$$\text{Teórico} \quad \rightarrow OUT_{TEO} = X^2$$

Ecuación [2-86]

2.2.5.5 La operación división

En la implementación de la operación división es donde se vislumbra una de las grandes ventajas de la lógica estocástica extendida, ya que permite implementar la división entre dos magnitudes “X=P/Q” y “Y=R/S” mediante una pareja de puertas **AND** que realizan el

producto cruzado entre las señales del numerador y del denominador, tal y como se muestra en la Ecuación [2-87]:

$$\begin{cases} X = P/Q \\ Y = R/S \end{cases} \rightarrow OUT = X/Y = \left(\frac{P}{Q}/\frac{R}{S}\right) = \frac{P \cdot S}{R \cdot Q} \rightarrow E(out(p,q,r,s)) = E\left(\frac{p/q}{r/s}\right) = \frac{P(p) \cdot P(s)}{P(r) \cdot P(q)} =$$

$$= \frac{P \cdot S / 2^{2 \cdot n}}{R \cdot Q / 2^{2 \cdot n}} = \frac{P \cdot S}{R \cdot Q}$$

Ecuación [2-87]

Como puede apreciarse en la Figura 2-65, la implementación de la división es extremadamente simple y consume muy pocos recursos lógicos en comparación a las implementaciones realizadas para la codificación estocástica clásica con y sin signo.

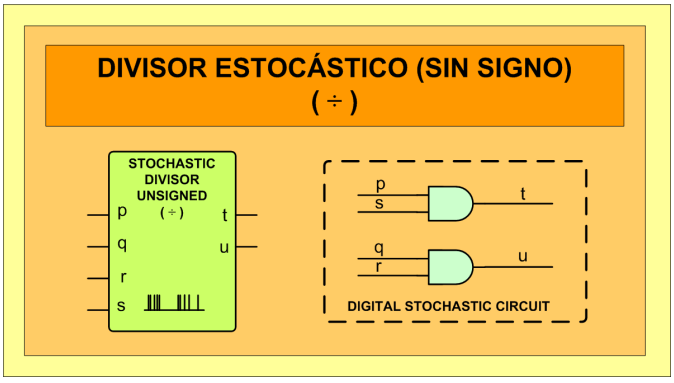

Figura 2-65: Bloque divisor (UESL)

A fin de corroborar que la implementación realizada se comporta de la forma predicha, se ha procedido a realizar una serie de medidas experimentales que se presentan en la Tabla 2-30: las cuales han consistido en fijar el valor de la magnitud "Y=1,25" mientras se varía el valor de la magnitud "X" en el intervalo [0, +7,25].

Tabla 2-30: Medidas de la división (UESL)

DATOS DE ENTRADA						EXPERIMENTAL			TEÓRICO	
P [16bits]	Q [16bits]	Valor X=P/Q	R [16bits]	S [16bits]	Valor Y=R/S	U [16bits]	V [16bits]	Valor OUT (X/Y)	X·Y	ERROR
0	8192	0	0	8192	0	0	0	Ind	Ind	Ind
10240	8192	1,25	10240	8192	1,25	1260	1291	0,9760	1,0000	0,0240

18432	8192	2,25	10240	8192	1,25	2268	1275	1,7788	1,8000	0,0212
26624	8192	3,25	10240	8192	1,25	3283	1265	2,5953	2,6000	0,0047
34816	8192	4,25	10240	8192	1,25	4368	1276	3,4232	3,4000	0,0232
43008	8192	5,25	10240	8192	1,25	5358	1308	4,0963	4,2000	0,1037
51200	8192	6,25	10240	8192	1,25	6201	1276	4,8597	5,0000	0,1403
59392	8192	7,25	10240	8192	1,25	7595	1292	5,8785	5,8000	0,0785

Seguidamente procedemos a representar en la Figura 2-66 los resultados experimentales (símbolos) obtenidos en la Tabla 2-30 a la vez que los descritos por la teoría (línea continua).

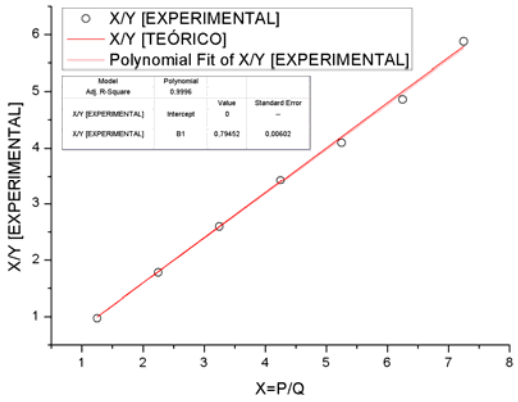

Figura 2-66: Resultados experimentales de la función división (UESL)

Finalmente procedemos a comparar la función de transferencia experimental obtenida en la Figura 2-66 mediante un ajuste por mínimos cuadrados con la descrita por la teoría para dicho bloque estocástico. Como podemos apreciar en la Ecuación [2-88] ambas son muy parecidas, siendo la principal discrepancia existente entre ellas derivado del error de cuantificación existente en la lógica estocástica sin signo.

$$\text{Experimental} \rightarrow OUT_{EXP} = 0{,}7945 \cdot X \text{ , con una } R^2 = 0{,}9996$$

$$\text{Teórico} \rightarrow OUT_{TEO} = \frac{X}{1{,}25} = 0{,}8000 \cdot X \qquad \text{Ecuación [2-88]}$$

2.2.6 LA CODIFICACIÓN EXTENDIDA CON SIGNO (SESL)

Las diversas variantes de la codificación estocástica presentadas hasta el momento presentan una serie de inconvenientes. El principal inconveniente se deriva de la presencia de una serie de errores estadísticos en el proceso de conversión de la señal estocástica a binaria. Como se ha visto en el apartado de lógica estocástica con signo esta representación puede ser modificada para representar magnitudes en el intervalo [-1, +1], mediante un simple cambio de variables "p*=2·p+1" (donde "p" representa el valor medio de conmutación asociado a la señal pulsante). Posteriormente y a fin de expandir las posibilidades de la codificación estocástica clásica se ha desarrollado la codificación estocástica extendida sin signo que ha permitido representar magnitudes binarias entre [0,+∞) mediante la razón entre una pareja de señales estocásticas, pero a diferencia de las anteriores codificaciones el error de conversión de las magnitudes dependerá directamente de los valores medios codificados. Finalmente y con objeto de solventar las deficiencias de la anterior representación se ha procedido a desarrollar la lógica estocástica extendida con signo, que haciendo uso del mismo cambio de variables se ha conseguido extender el rango de representación a todo el eje real (-∞, +∞).

$$\begin{cases} p \in [0,+1] \\ q \in [0,+1] \end{cases} \rightarrow \begin{cases} p^* \in [-1,+1] \\ q^* \in [-1,+1] \end{cases}$$

$$\begin{cases} p^* = 2 \cdot p - 1 \\ q^* = 2 \cdot q - 1 \end{cases} \Rightarrow x^* = \frac{p^*}{q^*} \rightarrow E(x) = E\left(\frac{p^*}{q^*}\right) = \frac{(2 \cdot P - 1)/2^2}{(2 \cdot Q - 1)/2^2} = \frac{2 \cdot P - 1}{2 \cdot Q - 1}, \forall x^* \in [-\infty,+\infty]$$

Ecuación [2-89]

Al igual que en la anterior codificación, las magnitudes "X" se representarán mediante la razón asociada a los valores medios de conmutación de dos señales pulsantes "p*" y "q*" definidos entre [-1, +1], de tal forma que la razón entre ambos permita representar valores entre (-∞, +∞). Por lo tanto, mediante la razón de dos señales pulsantes "p" y "q" inicialmente definidas entre [0, +1] pero aplicándoles el cambio de variables "x*=2y-1" demostramos (Ecuación [2-89]) cómo la razón resultante es proporcional a las magnitudes binarias "P" y "Q".

La nueva codificación, al igual que las anteriores, hará uso de bloques B2P(N) idénticos a los anteriores, con la única salvedad que necesitaremos dos (uno para el numerador y otro para el denominador). Para la conversión de las señales pulsantes a sus respectivas magnitudes binarias se usarán bloques P2B(N) idénticos a los usados hasta el momento.

Del mismo modo que ocurría en la lógica estocástica extendida sin signo es muy conveniente para el buen funcionamiento de los diferentes bloques estocásticos que los valores usados "P" y "Q" para codificar las diversas razones "X=P/Q" se correspondan a señales con valores medios de conmutación relativamente grandes. Para sistemas de computación extensos se requerirá de una regeneración de señales "p" y "q" que formen la razón "p/q" ya que los ratios de conmutación de la pareja de señales van decreciendo a medida que atraviesan los diversos bloques, hasta el punto de no permitir operar correctamente los diversos bloques estocásticos.

Cabe remarcar que la codificación estocástica extendida con signo "*Signed Extended Stochastic Logic*" (*SESL*) funcionará correctamente con cualquier circuito que haya sido diseñado para operar con la codificación estocástica clásica con signo, no obstante su rango de operación en algunos casos estará limitado a [-1,+1].

$$\begin{cases} Error(p^*) = \pm 2 \cdot \dfrac{1bit}{2^n} \rightarrow [-1,+1] \\ Error(q^*) = \pm 2 \cdot \dfrac{1bit}{2^n} \rightarrow [-1,+1] \\ Q^* = \dfrac{Dato1}{2^n} \\ P^* = \dfrac{Dato2}{2^n} \end{cases} \Rightarrow (-\infty,+\infty)$$

Ecuación [2-90]

$$X^* = \frac{P^*}{Q^*} \rightarrow Error(X^*) = \left|\frac{1}{Q^*}\right| \cdot Error(P^*) + \left|\frac{P^*}{Q^{*2}}\right| \cdot Error(Q^*)$$

En esta nueva codificación estocástica, el error de conversión asociado a una magnitud “X” que vendrá dado por la Ecuación [2-90]. Este error se ha duplicado con respecto de la lógica estocástica extendida sin signo, dependiendo éste a su vez de los valores que tomen las magnitudes “P” y “Q” usadas en la codificación.

La nueva codificación hará factible la implementación de redes neuronales y de sistemas de reconocimiento de patrones de forma sistemática. En el caso de las redes neuronales los rangos de valor de los pesos de conexión entre neuronas obtenidos mediante aplicaciones/algoritmos estándares suelen exceder los rangos de representación de las anteriores codificaciones, y en consecuencia para operar con ellos es necesaria su normalización en función del peso mayor presente en la red. Otra de las grandes ventajas de esta codificación es que permite operar directamente con bases de datos estándares (moléculares, huellas dactilares, etc.) sin tener que proceder a complejas y costosas (en tiempo de proceso) normalizaciones de los datos al rango de operación de la lógica para poder implementar sistemas de reconocimiento de patrones.

2.2.6.1 La operación suma/resta

Mediante esta nueva codificación que permite operar con magnitudes positivas y negativas, las operaciones aritméticas suma y resta son a la práctica equivalentes, con la salvedad que en la resta deberá procederse a modificar el signo (mediante una puerta **NOT**, de forma idéntica a como se procedía en la codificación estocástica clásica con signo). La operación suma entre dos magnitudes “X” e “Y” codificadas cada una de ellas mediante la razón de los valores medios de conmutación de dos señales pulsantes “p/q” y “”r/s” respectivamente, se regirá por la Ecuación [2-91]. Mientras que la resta lo hará por la Ecuación [2-92].

$$\begin{cases} X^* = P^* \big/ Q^* \\ Y^* = R^* \big/ S^* \end{cases} \rightarrow Z^* = X^* + Y^* = \frac{P^*}{Q^*} + \frac{R^*}{S^*} = \frac{\left(P^* \cdot S^* + R^* \cdot Q^*\right)}{Q^* \cdot S^*} \quad \text{Ecuación [2-91]}$$

$$\begin{cases} X^* = P^* \big/ Q^* \\ Y^* = R^* \big/ S^* \end{cases} \rightarrow Z^* = X^* - Y^* = \frac{P^*}{Q^*} - \frac{R^*}{S^*} = \frac{\left(P^* \cdot S^* - R^* \cdot Q^*\right)}{Q^* \cdot S^*} \quad \text{Ecuación [2-92]}$$

Tal y como se demuestra a continuación esta nueva codificación tiene la ventaja de permitir implementar la suma y la resta natural de dos magnitudes “X” e “Y”.

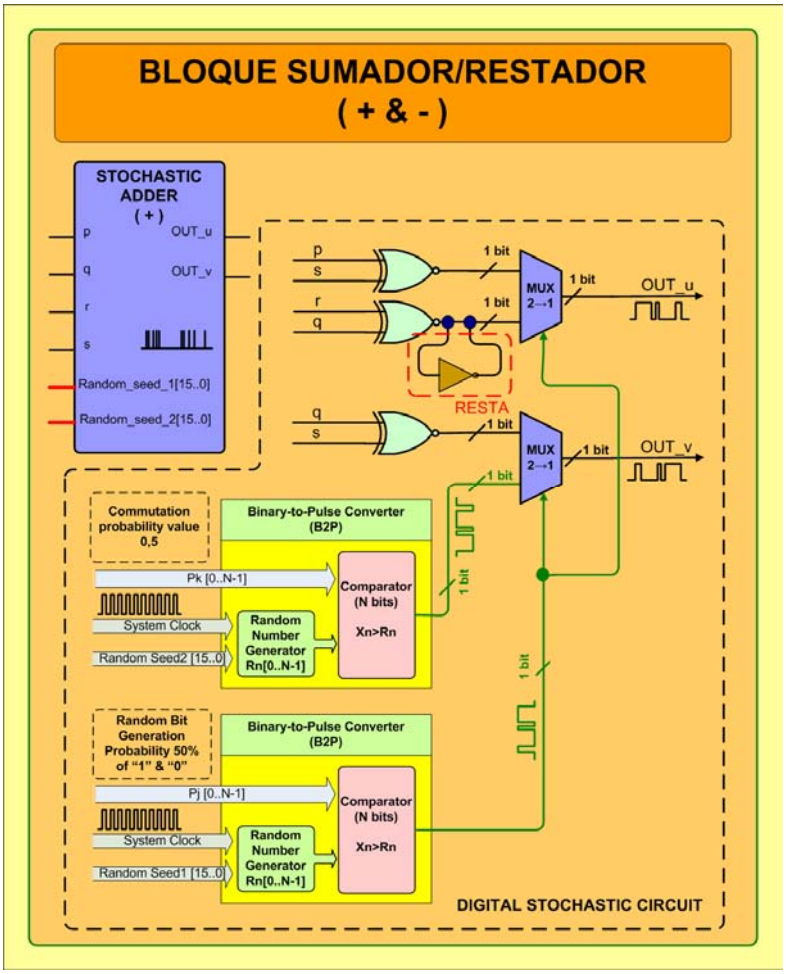

Figura 2-67: Bloque sumador/restador (SESL)

La implementación del bloque sumador/restador (Figura 2-67) se compondrá de dos partes: una para el numerador y otra para el denominador.

El circuito del numerador se compone de un multiplexor (2→1) gobernado por una señal de control de la activación “c” descorrelacionada. Evaluando así el valor medio del producto cruzado entre las señales del numerador y del denominador (p*·s* y r*·q*) implementado mediante una pareja de puertas **XNOR** de dos entradas. En el caso del restador en el numerador se insertará una puerta **NOT** entre la salida de la puerta XNOR que implementa

el producto (r·q) y la entrada el multiplexor (2→1) a fin de modificar el signo de la señal codificada, obteniendo así la resta en valor medio.

A su vez el circuito que evalúa el denominador en esta codificación es algo más complejo que en los casos anteriores, ya que se ha de realizar el producto de los denominadores en valor medio y compensar así el factor (1/2) del numerador. Será necesario realizar el producto entre las señales de los numeradores mediante una puerta **XNOR** de dos entradas para luego mediante una multiplexor (2→1) proceder a sumar en valor medio esta señal, con una señal constante con un valor medio de conmutación igual a "0,5".

La demostración matemática de la función implementada mediante el circuito propuesto en la Figura 2-67, se presenta mediante la Ecuación [2-93]:

$$\begin{cases} x = p/q \\ y = r/s \\ j = \overline{p \oplus s} \\ k = \overline{r \oplus q} \\ l = \overline{q \oplus s} \end{cases} \Rightarrow \text{clasicamente} \Rightarrow out = \frac{out\_u}{out\_v} = \frac{p}{q} + \frac{r}{s} = \frac{\left(\overline{p \oplus s}\right)c + \left(\overline{r \oplus q}\right)\bar{c}}{\left(\overline{q \oplus s}\right)c + 0{,}5\cdot\bar{c}} = \frac{j\cdot c + k\cdot\bar{c}}{l\cdot c + 0{,}5\cdot\bar{c}} = a$$

$$\begin{cases} \bar{c} = (1-c) \\ \bar{l} = (1-l) \end{cases} \Rightarrow a = \frac{j\cdot c + k\cdot(1-c)}{l\cdot c + 0{,}5\cdot(1-c)} = \frac{j\cdot c + k - k\cdot c}{l\cdot c + 0{,}5 1 - 0{,}5\cdot c} = b$$

Ecuación [2-93]

Si de la expresión resultante de la Ecuación [2-93] suponemos que los datos han sido codificados mediante el cambio de variables de la lógica estocástica extendida con signo obtendremos la siguiente expresión (Ecuación [2-94]):

$$\begin{cases} j^* = 2\cdot j - 1 \\ k^* = 2\cdot k - 1 \\ l^* = 2\cdot l - 1 \\ c^* = 2\cdot c - 1 \\ out\_u^* = 2\cdot out\_u - 1 \\ out\_v^* = 2\cdot out\_v - 1 \end{cases} \rightarrow \begin{cases} j = \dfrac{j^*+1}{2} \\ k = \dfrac{k^*+1}{2} \\ l = \dfrac{l^*+1}{2} \\ c = \dfrac{c^*+1}{2} \\ out\_u = \dfrac{out\_u^*+1}{2} \\ out\_v = \dfrac{out\_v^*+1}{2} \end{cases} \qquad \Rightarrow b = out = \frac{u}{v} = \frac{j\cdot c + k - k\cdot c}{l\cdot c + 0{,}5\cdot 1 - 0{,}5\cdot c} \Rightarrow c$$

$$c = \frac{\dfrac{out\_u^*+1}{2}}{\dfrac{out\_v^*+1}{2}} = \frac{\dfrac{j^*+1}{2}\cdot\dfrac{c^*+1}{2} + \dfrac{k^*+1}{2} - \dfrac{k^*+1}{2}\cdot\dfrac{c^*+1}{2}}{\dfrac{l^*+1}{2}\cdot\dfrac{c^*+1}{2} + \dfrac{1}{2}\cdot 1 - \dfrac{1}{2}\cdot\dfrac{c^*+1}{2}} = \frac{\frac{1}{4}\cdot\left(\left(j^*+1\right)\cdot\left(c^*+1\right) + 2\cdot\left(k^*+1\right) - \left(k^*+1\right)\cdot\left(c^*+1\right)\right)}{\frac{1}{4}\cdot\left(\left(l^*+1\right)\cdot\left(c^*+1\right) + 2 - \left(c^*+1\right)\right)} = d$$

$$\Rightarrow \frac{2\cdot\left(\dfrac{out\_u^*+1}{2}\right)}{2\cdot\left(\dfrac{out\_u^*+1}{2}\right)} = \frac{2\cdot\frac{1}{4}\cdot\left(\left(j^*+1\right)\cdot\left(c^*+1\right) + 2\cdot\left(k^*+1\right) - \left(k^*+1\right)\cdot\left(c^*+1\right)\right)}{2\cdot\frac{1}{4}\cdot\left(\left(l^*+1\right)\cdot\left(c^*+1\right) + 2 - \left(c^*+1\right)\right)} = e$$

$$\Rightarrow \frac{out\_u^*+1}{out\_v^*+1} = \frac{\frac{1}{2}\cdot\left(\left(j^*+1\right)\cdot\left(c^*+1\right) + 2\cdot\left(k^*+1\right) - \left(k^*+1\right)\cdot\left(c^*+1\right)\right)}{\frac{1}{2}\cdot\left(\left(l^*+1\right)\cdot\left(c^*+1\right) + 2 - \left(c^*+1\right)\right)} =$$

$$= \frac{\frac{1}{2}\cdot\left(j^*\cdot c^* + j^* + c^* + 1 + 2\cdot k^* + 2 - k^*\cdot c^* - k^* - c^* - 1\right)}{\frac{1}{2}\cdot\left(l^*\cdot c^* + c^* + l^* + 1 + 2 - c^* - 1\right)} = \frac{\frac{1}{2}\cdot\left(c^*\cdot\left(j^* + 1 - k^* - 1\right) + j^* + 1 + 2\cdot k^* + 2 - k^* - 1\right)}{\frac{1}{2}\cdot\left(c^*\cdot\left(l^* + 1 - 1\right) + l^* + 2\right)} =$$

$$= \frac{\frac{1}{2}\cdot\left(c^*\cdot\left(j^* - k^*\right) + j^* + k^* + 2\right)}{\frac{1}{2}\cdot\left(c^*\cdot\left(l^*\right) + 2 + l^*\right)} = g$$

En el supuesto que fijemos $c^* = 0 \rightarrow g = \dfrac{0{,}5\cdot j^* + 0{,}5\cdot k^* + 1}{+1 + 0{,}5\cdot l^*} \Rightarrow \dfrac{out\_u^*+1}{out\_v^*+1} = \dfrac{0{,}5\cdot j^* + 0{,}5\cdot k^* + 1}{+1 + 0{,}5\cdot l^*} \Rightarrow$

$$\text{Si procedemos ahora a sustituir las expresiones} \rightarrow \begin{cases} j^* = \overline{p \oplus s} = p^* \cdot s^* \\ k^* = \overline{r \oplus q} = r^* \cdot q^* \\ l^* = \overline{q \oplus s} = q^* \cdot s^* \end{cases}$$

$$\begin{cases} out\_u^* = 0{,}5 \cdot j^* + 0{,}5 \cdot k^* + 1 - 1 = 0{,}5 \cdot j^* + 0{,}5 \cdot k^* \\ out\_v^* = +1 + 0{,}5 \cdot l^* - 1 = 0{,}5 \cdot l^* \end{cases} \rightarrow out^* = \frac{out\_u^*}{out\_v^*} = \frac{0{,}5 \cdot j^* + 0{,}5 \cdot k^*}{0{,}5 \cdot l^*} = \frac{j^* + k^*}{l^*} = \frac{p^* \cdot s^* + r^* \cdot q^*}{q^* \cdot s^*}$$

$$E(out^*) = \frac{\left(\frac{(2 \cdot P - 1)}{2^n} \cdot \frac{(2 \cdot S - 1)}{2^n} + \frac{(2 \cdot R - 1)}{2^n} \cdot \frac{(2 \cdot Q - 1)}{2^n} \right)}{\left(\frac{(2 \cdot Q - 1)}{2^n} \cdot \frac{(2 \cdot S - 1)}{2^n} \right)} = \frac{(2 \cdot P - 1) \cdot (2 \cdot S - 1) + (2 \cdot R - 1) \cdot (2 \cdot Q - 1)}{(2 \cdot Q - 1) \cdot (2 \cdot S - 1)}$$

Ecuación [2-94]

Para comprobar la implementación realizada del circuito sumador se ha procedido a realizar una serie de medidas experimentales las cuales se presentan en la Tabla 2-31. Las medidas han consistido en fijar el valor de "Y=R/S=0,214" e ir variando el valor de "X"=P/Q" en el intervalo [-1, +1,143].

Tabla 2-31: Medidas de la operación suma/resta (SESL)

Datos de entrada (valores equivalentes)		Experimental (salida) [16bits]	Experimental (salida) [16bits]	Experimental (valor equivalente)
Valor X=p*/q*	**Valor Y=r*/s***	**t*=p*s*+q*r***	**u*=q*s***	**Z=t*/u***
1.143	0.214	49630	45037	1.374
1.000	0.214	47487	45057	1.198
0.857	0.214	45342	45083	1.021
0.714	0.214	44321	44898	0.952
0.571	0.214	42437	45071	0.786
0.429	0.214	40722	45051	0.648
0.286	0.214	39679	45074	0.562
0.143	0.214	37462	44945	0.385
0.000	0.214	35255	44860	0.206
-0.143	0.214	33394	45151	0.051
-0.286	0.214	31291	45177	-0.119
-0.429	0.214	30282	45199	-0.200
-0.571	0.214	28099	45012	-0.381
-0.714	0.214	26271	45159	-0.524
-0.857	0.214	25254	44839	-0.622
-1.000	0.214	23317	45075	-0.768

A continuación procedemos a representar gráficamente (Figura 2-68) los datos experimentales obtenidos (símbolos) en la Tabla 2-31, y sobre éstos se realiza un ajuste por mínimos cuadrados para determinar la función de transferencia.

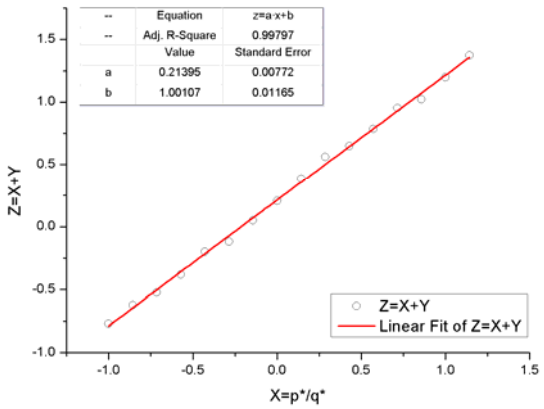

Figura 2-68: Resultados experimentales de la operación suma (SESL)

Si finalmente procedemos a comparar (Ecuación 2-95) la función de transferencia experimental obtenida en la Figura 2-68 con la descrita por la teoría, podremos apreciar cómo ambas funciones son idénticas.

$$\text{Teórica} \quad \rightarrow Z = 1 \cdot X + 0.2140$$
$$\text{Experimental} \quad \rightarrow Z = 1.0011 \cdot X + 0.2140 \text{, con una } R^2 = 0{,}9980$$

Ecuación [2-95]

2.2.6.2 La operación multiplicación/división

En la codificación estocástica extendida con signo la multiplicación y la división natural se realizarán mediante una pareja de puertas **XNOR** de dos entradas, que en el caso de la multiplicación se realizará mediante el producto directo entre las señales de los numeradores y los denominadores (Ecuación [2-96]). Para la división se procederá a realizar el producto cruzado entre las señales de los numeradores con los denominadores (Ecuación [2-97]).

$$\left.\begin{array}{l} X = \dfrac{p*}{q*} \\ Y = \dfrac{r*}{s*} \end{array}\right\} \xrightarrow[Multiplicación]{} OUT = X \cdot Y = \frac{`p* \cdot r*}{q* \; s*} = \frac{p* \cdot r*}{q* \cdot s*} \qquad \text{Ecuación [2-96]}$$

$$\left.\begin{array}{l} X = \dfrac{p*}{q*} \\ Y = \dfrac{r*}{s*} \end{array}\right\} \xrightarrow[División]{} OUT = \frac{X}{Y} = \frac{p* \diagup q*}{r* \diagup s*} = \frac{p* \cdot s*}{q* \cdot r*} \qquad \text{Ecuación [2-97]}$$

La implementación del circuito multiplicador y divisor se presenta en la Figura 2-69, como puede apreciarse ambos circuitos son idénticos y sólo las señales de entrada de las puertas **XNOR** se hallan intercambiadas.

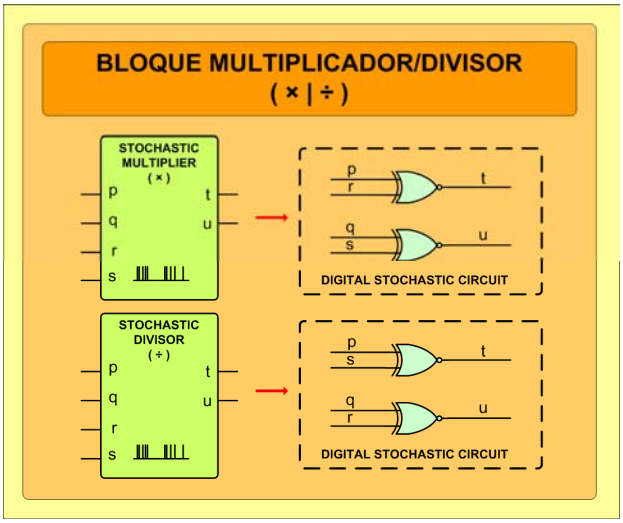

Figura 2-69: Bloque multiplicador/divisor (SESL)

Antes de proceder a las comprobaciones experimentales se ha comprobado teóricamente en la Ecuación [2-98] la equivalencia matemática entre la operación aritmética multiplicación con la implementación digital presentada en la Figura 2-69.

$$\begin{cases} x = p/q \\ y = r/s \end{cases} \Rightarrow \text{clasicamente } \Rightarrow out = \frac{out\_u}{out\_v} = x \cdot y = \frac{p}{q} \cdot \frac{r}{s} = \frac{\left(\overline{p \oplus r}\right)}{\left(\overline{q \oplus s}\right)} = \frac{\bar{p} \cdot \bar{r} + p \cdot r}{\bar{q} \cdot \bar{s} + q \cdot s} = a$$

$$\begin{cases} \bar{p} = (1-p) \\ \bar{r} = (1-r) \\ \bar{q} = (1-q) \\ \bar{s} = (1-s) \end{cases} \Rightarrow a = \frac{(1-p)\cdot(1-r) + p \cdot r}{(1-q)\cdot(1-s) + q \cdot s} = \frac{1-p-r+p \cdot r+p \cdot r}{1-q-s+q \cdot s+q \cdot s} = \frac{1-p-r+2 \cdot p \cdot r}{1-q-s+2 \cdot q \cdot s} = b$$

Si los datos han sido codificados mediante el cambio de variables de logica estocástica con signo :

$$\begin{cases} p^* = 2 \cdot p - 1 \\ q^* = 2 \cdot q - 1 \\ r^* = 2 \cdot r - 1 \\ s^* = 2 \cdot s - 1 \\ out\_u^* = 2 \cdot out\_u - 1 \\ out\_v^* = 2 \cdot out\_v - 1 \end{cases} \rightarrow \begin{cases} p = \frac{p^*+1}{2} \\ q = \frac{q^*+1}{2} \\ r = \frac{r^*+1}{2} \\ s = \frac{s^*+1}{2} \\ out\_u = \frac{out\_u^*+1}{2} \\ out\_v = \frac{out\_v^*+1}{2} \end{cases} \Rightarrow b = out = \frac{u}{v} = \frac{1-p-r+2 \cdot p \cdot r}{1-q-s+2 \cdot q \cdot s} \Rightarrow c$$

$$c \Rightarrow out^* = \frac{\frac{out\_u^*+1}{2}}{\frac{out\_v^*+1}{2}} = \frac{1 - \frac{p^*+1}{2} - \frac{r^*+1}{2} + 2 \cdot \frac{p^*+1}{2} \cdot \frac{r^*+1}{2}}{1 - \frac{q^*+1}{2} - \frac{s^*+1}{2} + 2 \cdot \frac{q^*+1}{2} \cdot \frac{s^*+1}{2}} = d$$

$$d = \frac{\frac{1}{2} \cdot \left(2 - p^* - 1 - r^* - 1 + p^* \cdot r^* + p^* + r^* + 1\right)}{\frac{1}{2} \cdot \left(2 - q^* - 1 - s^* - 1 + q^* \cdot s^* + q^* + s^* + 1\right)} = \frac{\frac{1}{2} \cdot \left(1 + p^* \cdot r^*\right)}{\frac{1}{2} \cdot \left(1 + q^* \cdot s^*\right)} = \frac{1 + p^* \cdot r^*}{1 + q^* \cdot s^*}$$

$$out^* = \frac{out\_u^*+1}{out\_v^*+1} = \frac{1 + p^* \cdot r^*}{1 + q^* \cdot s^*} \rightarrow out^* = \frac{out\_u^*}{out\_v^*} = \frac{p^* \cdot r^*}{q^* \cdot s^*}$$

$$E(out^*) = \frac{\frac{(2 \cdot P - 1)}{2^n} \cdot \frac{(2 \cdot R - 1)}{2^n}}{\frac{(2 \cdot Q - 1)}{2^n} \cdot \frac{(2 \cdot S - 1)}{2^n}} = \frac{(2 \cdot P - 1)\cdot(2 \cdot R - 1)}{(2 \cdot Q - 1)\cdot(2 \cdot S - 1)}$$

Ecuación [2-98]

Entonces, una vez demostrada teóricamente la implementación realizada pasaremos a presentar en la Tabla 2-32 las medidas experimentales realizadas sobre el circuito multiplicador, que han consistido en fijar el valor de (Y=R*/S*=0,9333) e ir variando el valor de (X=P/Q) en el intervalo [-1, +1]

Tabla 2-32: Medidas de la operación multiplicación (SESL)

Datos de entrada (Codificados)		Experimental [16bits]	Experimental [16bits]	Experimental [16bits]	Teórico	Error Abs(Th-Exp)
X=p*/q*	Y=r*/s*	t*=p*·*r*	u*=q*·s*	Z=t*/u*	Z	Z
1,0000	0,9333	59510	61478	0,9315	0,9333	0,0019
0,9333	0,9333	57793	61495	0,8711	0,8711	0,0000
0,8667	0,9333	55657	61411	0,7991	0,8089	0,0098
0,8000	0,9333	54398	61661	0,7486	0,7467	0,0020
0,7333	0,9333	52556	61521	0,6882	0,6844	0,0038
0,6667	0,9333	50852	61497	0,6295	0,6222	0,0072
0,6000	0,9333	48594	61461	0,5516	0,5600	0,0084
0,5333	0,9333	47368	61609	0,5062	0,4978	0,0084
0,4667	0,9333	45104	61242	0,4332	0,4356	0,0023
0,4000	0,9333	43443	61483	0,3718	0,3733	0,0016
0,3333	0,9333	41810	61441	0,3153	0,3111	0,0042
0,2667	0,9333	40082	61602	0,2537	0,2489	0,0048
0,2000	0,9333	38143	61487	0,1872	0,1867	0,0005
0,1333	0,9333	36290	61627	0,1220	0,1244	0,0024
0,0667	0,9333	34459	61475	0,0589	0,0622	0,0033
0,0000	0,9333	32786	61586	0,0006	0,0000	0,0006
-0,0667	0,9333	31117	61810	-0,0568	-0,0622	0,0054
-0,1333	0,9333	29237	61495	-0,1229	-0,1244	0,0015
-0,2000	0,9333	27456	61576	-0,1844	-0,1867	0,0023
-0,2667	0,9333	25413	61689	-0,2543	-0,2489	0,0054
-0,3333	0,9333	24002	61617	-0,3039	-0,3111	0,0073
-0,4000	0,9333	22326	61484	-0,3636	-0,3733	0,0097
-0,4667	0,9333	20300	61523	-0,4336	-0,4356	0,0020
-0,5333	0,9333	18134	61603	-0,5075	-0,4978	0,0097
-0,6000	0,9333	16482	61662	-0,5636	-0,5600	0,0036
-0,6667	0,9333	14947	61644	-0,6172	-0,6222	0,0051
-0,7333	0,9333	12960	61546	-0,6883	-0,6844	0,0039
-0,8000	0,9333	11485	61583	-0,7386	-0,7467	0,0081
-0,8667	0,9333	9578	61539	-0,8060	-0,8089	0,0029
-0,9333	0,9333	7967	61444	-0,8649	-0,8711	0,0062
-1,0000	0,9333	6092	61512	-0,9281	-0,9333	0,0053

Acto seguido procedemos a representar gráficamente (Figura 2-70) las medidas experimentales (símbolos) presentadas en la Tabla 2-32, para luego proceder a ajustar una función de transferencia a partir de los datos experimentales, el ajuste lineal se ha representado mediante una línea continua.

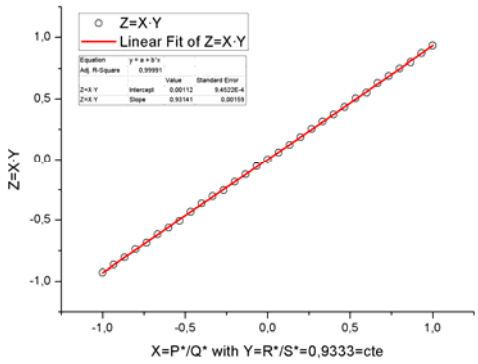

Figura 2-70: Resultados experimentales de la multiplicación (SESL)

Finalmente procedemos a comparar en la Ecuación [2-99] las funciones de transferencia experimental obtenida en la Figura 2-70 con la esperada por la teoría. En ella podemos apreciar como ambas funciones de transferencia son prácticamente idénticas.

$$\begin{aligned} &\text{Teórica} && \rightarrow Z = 0.9333 \cdot X \\ &\text{Experimental} && \rightarrow Z = 0.9314 \cdot X \text{ , con una } R^2 = 0{,}9999 \end{aligned} \qquad \text{Ecuación [2-99]}$$

2.2.6.3 Bloques Mixtos

Para codificación estocástica extendida con signo (al igual que sucede con la sin signo) existen toda una serie de funciones que no pueden implementarse como fracciones simples P/Q o simplemente que aun no hemos hallado la forma de implementarlas mediante la razón entre dos señales pulsantes “p” y “q”. Éste es el caso de algunas funciones como por ejemplo la tangente hiperbólica

Para ello y como solución de compromiso para poder implementar dichas funciones en la lógica estocástica extendída con signo hemos desarrollado un bloque al cual lo hemos definido como *función mixta* (Figura 2-71), que permite mediante la metodología que presentamos a continuación operar bloques estocásticos clásicos (con signo) con señales extendidas (con signo). Las codificaciones son idénticas con la salvedad que en la segunda la información se halla codificada como la razón entre dos señales pulsantes.

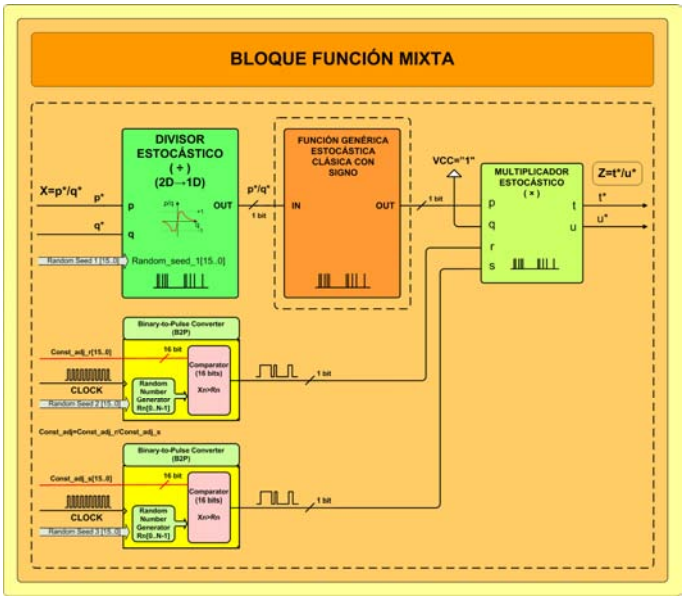

Figura 2-71: Bloque función mixta (SESL)

La metodología aquí propuesta se basa en el uso de un bloque divisor (descrito en el subapartado 2.2.4.5) que evalúa a su salida una señal pulsante equivalente a la razón entre las dos señales pulsantes “p*” y “q*” de entrada que codifican la magnitud “X=P/Q”. Como bien puede apreciarse el primer inconveniente radica en el hecho que si la razón entre ambas señales es mayor o menor al rango de representación de la señal estocástica de salida [-1,+1] ésta saturará a sus valores límite. Seguidamente la señal resultante del bloque divisor (equivalente a la razón (p*/q*)) estará conectado a la entrada del bloque estocástico clásico con signo cuya función de transferencia deseamos disponer. Finalmente la salida de dicho bloque se reconvertirá a una magnitud extendida mediante un bloque multiplicador extendido con signo. La salida del bloque anterior se conectará con la señal del numerador “p*” de la magnitud extendida a codificar mientras que el numerador “q*” de dicha señal se le prefija un valor de “+1” conectando la señal a uno “1→2·1-1=+1” lógico fijo. Las señales “r*” y “s*” encargadas de codificar mediante su razón la magnitud “Y=r*/s*”, servirán para multiplicar la salida del bloque por una constante de ajuste en el caso que esta se requiera. Así pues, la función de salida se regirá mediante la Ecuación [2-100]:

$$\left\{ X^* = \frac{P^*}{Q^*} \rightarrow x^* = \frac{p^*}{q^*} \in (-\infty,+\infty) \rightarrow out^*_{Divisor} = \frac{p^*}{q^*} \in [-1,+1] \rightarrow \right.$$

$$out^*_{f(x)} = f\left(out^*_{Divisor}\right) \in [-1,+1] \rightarrow out^*_{circuit} = \frac{f\left(\frac{p^*}{q^*}\right)}{+1} \cdot \frac{Cons\tan t\_adjust\_r}{Cons\tan t\_adjust\_s} \in (-\infty,+\infty)$$

Ecuación [2-100]

En el supuesto que no se requiera del ajuste de la magnitud de salida se puede prescindir del bloque multiplicador. Conectando la salida del bloque clásico con signo en el numerador de la señal pulsante de salida y como numerador una señal fija a un '1' lógico, se obtiene una magnitud estocástica extendida "Z" de salida (directamente proporcional a la razón entre ambas señales).

A continuación se presenta la función mixta que implementa la función *pseudo tangente hiperbólica*.

2.2.6.3.1 La función pseudo tangente hiperbólica

La implementación de la función *pseudo tangente hiperbólica* en es fundamental para la implementación de funciones de activación no lineales en el campo de las redes neuronales. Las funciones de activación son principalmente las Sigmoidales y Tangente-Sigmoidales, siendo estas extremadamente versátiles a la hora de implementar una red neuronal. Por lo tanto para reproducir la función de activación tangente Sigmoidal será necesario disponer de un bloque estocástico extendido que implemente la función tangente hiperbólica de una magnitud extendida "X" codificada mediante la razón de dos señales pulsantes "p" y "q", cada una de ellas con un rango de representación de [-1, +1].

Ante la imposibilidad de hallar una función extendida nativa que implemente satisfactoriamente dicha función como la fracción de dos señales pulsantes, nos hemos visto forzados a implementar esta función mediante la metodología anteriormente descrita como una *función mixta* (como puede apreciarse en la Figura 2-72).

La implementación del bloque ha consistido en usar un divisor clásico con signo a fin de evaluar la razón entra las señales en que se ha codificado la magnitud extendida

“X*=p*/q*” de entrada, obtenido a la salida una nueva señal estocástica ahora acotada en el intervalo [-1, +1]. A priori este hecho supone un serio inconveniente a la hora de implementar la función tangente hiperbólica “*tanh(x)*” natural, ya que para valores de entrada entre [-1, +1] está aún no ha saturado a sus valores límite de salida [-1, +1]. Por lo tanto, para solventar este problema usaremos valores de ganancia “k” de la función “*tanh(x)*”, para los cuales la salida de la función se halle totalmente (o casi totalmente) definida en el intervalo de representación [-1, +1]. Es decir que para valores de entrada cercanos a -1 o +1 su salida ya haya saturado a sus valores límite. De esta forma se posibilita que no se pierda información en el proceso de evaluación de la razón entre las señales de entrada “p” y “q”.

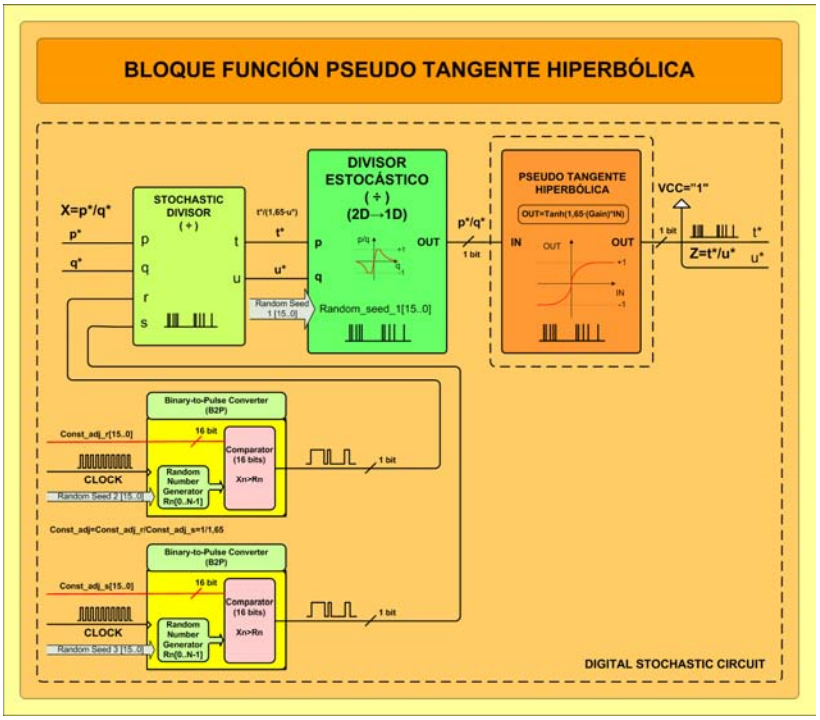

Figura 2-72: Bloque función pseudo tangente hiperbólica (SESL)

El objetivo es obtener a la salida del bloque la función “*Tanh(x)*” y no “*Tanh(k·x)*”, por lo tanto deberemos proceder a dividir la razón de entrada por una factor “*1/k*” mediante un bloque divisor extendido, a fin de normalizar la función implementada por el bloque *función mixta*.

Al no requerir la salida del bloque de ningún reescalado, se ha prescindido del bloque multiplicador a la salida. La señal de salida del denominador del bloque de la función pseudo tangente hiperbólica se ha fijado a nivel alto '1' (lo que equivale a fijarla a un valor "+1"). Por lo tanto, la salida del bloque vendrá regida por la Ecuación [2-101]:

$$out^* = \frac{u}{v} = \frac{Tanh\left(k \cdot \left(\frac{p^*}{q^*} \cdot \frac{1}{k}\right)\right)}{1} = \frac{Tanh\left(p^*/q^*\right)}{1} \qquad \text{Ecuación [2-101]}$$

Una vez descrita la metodología para la implementación de la función pseudo tangente hiperbólica en la codificación (SESL) hemos procedido a evaluar el correcto funcionamiento de la implementación realizada. Para ello hemos fijando el valor del parámetro de ajuste de entrada del circuito "Y=R*/S*=1", el valor del parámetro "Gain=7" del bloque *pseudo-tangente hiperbólica* y finalmente hemos variado el valor de la razón de entrada "X=P*/Q*" en el intervalo [-1,25, +1,33]. Los resultados experimentales obtenidos los que se presentan en la Tabla [2-33]:

Tabla 2-33: Medidas de la función pseudo tangente hiperbólica (SESL)

Datos de entrada (Codificados)		Experimental [16bits]	Experimental [16bits]	Teórico	Teórico [16bits]	Error Abs(Th -Exp)
X=p*/q*	Y=r*/s*	Numerador OUT (t*)	Valor OUT t*/65535	Valor OUT=Tanh(1,86·t*/u*)	Valor OUT=Tanh(1,86·t*/u*)	Z
1,3333	1,0000	65535	1,0000	0,9861	65080	0,0140
1,1667	1,0000	65423	0,9966	0,9743	64693	0,0223
1,0000	1,0000	65215	0,9903	0,9527	63985	0,0376
0,8333	1,0000	63567	0,9400	0,9138	62711	0,0262
0,6667	1,0000	60935	0,8596	0,8454	60472	0,0142
0,5000	1,0000	56191	0,7149	0,7306	56708	0,0157
0,3333	1,0000	49063	0,4973	0,5511	50826	0,0538
0,1666	1,0000	40479	0,2354	0,3004	42611	0,0650
0,0000	1,0000	32344	-0,0129	-0,0001	32766	0,0128
-0,1667	1,0000	24783	-0,2437	-0,3005	22921	0,0569
-0,3334	1,0000	16639	-0,4922	-0,5512	14706	0,0590
-0,5001	1,0000	8792	-0,7317	-0,7306	8826	0,0010
-0,6667	1,0000	4959	-0,8487	-0,8455	5063	0,0032
-0,8334	1,0000	2152	-0,9343	-0,9138	2824	0,0205
-1,0001	1,0000	536	-0,9836	-0,9527	1550	0,0309
-1,1668	1,0000	208	-0,9937	-0,9743	843	0,0194
-1,2501	1,0000	144	-0,9956	-0,9811	620	0,0145

Como bien puede apreciarse en la Tabla [2-33] el parámetro de ajuste "Gain=7" del bloque *pseudo tanh(x)* se corresponde experimentalmente con una ganancia de "k=1,86" de la

función *tanh(k·x)* teórica. Finalmente hemos procedido a representar gráficamente (Figura 2-73) las medidas experimentales (símbolos) presentadas en la Tabla [2-33], a fin de presentar la semejanza entre la función implementada con la predicha por la teoría.

A continuación hemos evaluado, nuevamente la función matemática que relaciona la ganancia de salida del bloque *pseudo tangente hiperbólica* con el parámetro de ajuste "Gain", a fin de tener en cuenta en este caso la presencia del bloque divisor. Los resultados obtenidos para los diferentes valores "Gain" se presentan a continuación en la Tabla [2-34]:

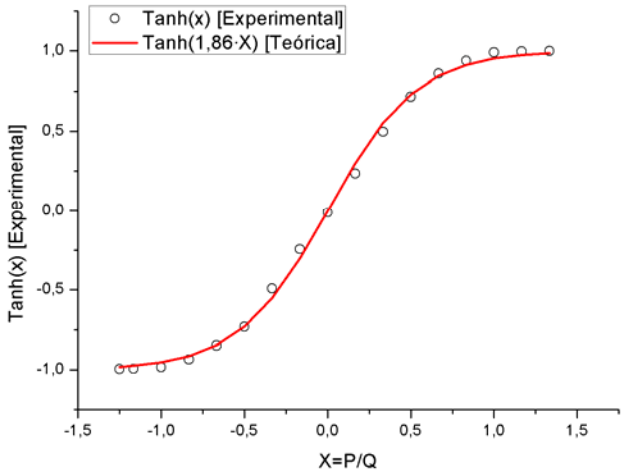

Figura 2-73: Resultados experimentales de la función pseudo tangente hiperbólica (SESL)

Tabla 2-34: Ganancia de la función pseudo tangente hiperbólica (SESL)

Parámetro Gain [16bits]	Ganancia del la función Tanh(k·x)
3	1,4200
5	1,6500
7	1,8600
9	2,0300
11	2,1600
21	2,8400
41	4,0687

A partir de los valores de ganancia "k" obtenidos para los diferentes valores del parámetro "Gain" presentados en la Tabla [2-34], hemos procedido a realizar un ajuste lineal sobre

ellos, para determinar la relación entre ambas variables. La relación obtenida se presenta mediante la Ecuación [2-102]:

$$k = 0.0659 \cdot \text{Gain} + 1.3668 \text{, con una } R^2 = 0{,}9997 \qquad \text{Ecuación [2-102]}$$

3. RECONOCIMIENTO DE PATRONES

En el presente capítulo se aborda la implementación estocástica de sistemas de reconocimiento de patrones en el marco de la aproximación estadística. A fin de establecer el marco teórico sobre el cual se cimientan las diferentes aportaciones realizadas se ha procedido a la realización de una introducción general al campo del reconocimiento de patrones, y en especial a los sistemas basados en la aproximación estadística. Se presentan diversas aportaciones realizadas en el campo del reconocimiento de patrones estocástico en el supuesto de conocer las funciones distribución de probabilidad de las diversas categorías. A su vez se abordarán las metodologías de clasificación estocásticas desarrolladas para el campo del reconocimiento de patrones (válidas para cualquier supuesto), así como diversas implementaciones estocásticas de las *funciones distribución de probabilidad (f.d.p)* de las categorías conocidas.

Finalmente se aborda el reconocimiento de patrones estocástico en el supuesto de densidades de probabilidad condicionales desconocidas, debido a que en el mundo real generalmente se desconoce a priori la *(f.d.p)* característica de una población. Para ello se presentan las aportaciones realizadas fundamentadas en la metodología de las *ventanas de Parzen*.

3.1 INTRODUCCIÓN GENERAL

El reconocimiento de patrones [3-1] es una parte del conjunto de conocimientos multidisciplinares encuadrados en el campo de la ciencia conocido como Inteligencia Computacional o *Machine Learning*, el cual permite a partir de una serie de datos (directamente escogidos del entorno en observación) realizar una determinada acción basada en la “categoría” de un patrón de referencia de la serie. Como ejemplos de formas de reconocimiento debemos mencionar el facial, el auditivo, el de caracteres e infinidad de muchos otros que son realizados por los seres vivos de forma inconsciente. Estas funciones

cognitivas superiores, aunque parezcan triviales para el ser humano, son extremadamente complejas y laboriosas de implementar en los sistemas automáticos.

El objetivo básico del reconocimiento de patrones es asociar estímulos de entrada con la "categoría" a la cual pertenecen. Un "patrón" será la formación que caracteriza y diferencia el elemento a reconocer. Como es obvio los patrones deben extraerse siempre de las mismas fuentes, es decir, no se pueden comparar patrones obtenidos para el reconocimiento de caracteres con los obtenidos para el reconocimiento del habla; no obstante, sí que podemos fusionarlos para generar un nuevo patrón capaz de reconocer el habla y la escritura de un grupo de personas. En el presente capítulo nos referiremos a los patrones como muestras, funciones, características o vectores. El concepto de "clase" se refiere a un conjunto o grupo de patrones con características similares entre si y diferentes a los patrones de otras clases. Cabe remarcar que en el presente texto hablaremos indistintamente de clases o categorías.

Un sistema de reconocimiento de patrones se puede descomponer genéricamente en cinco etapas diferentes [3-1, 3-3], cada una de ellas con una función bien definida en el proceso; tal y como se muestra en la Figura 3-1.

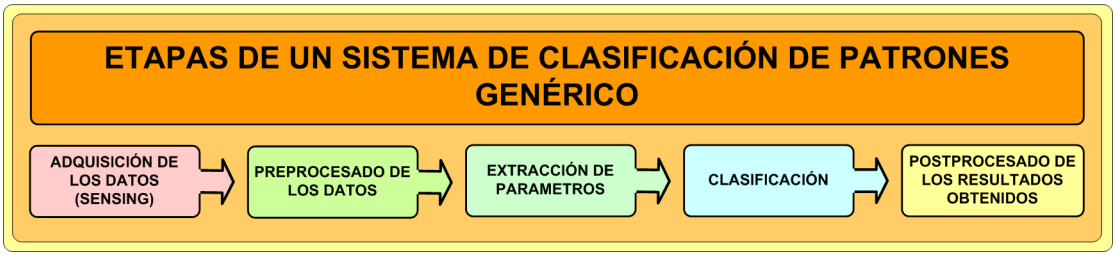

Figura 3-1: Partes funcionales de un sistema genérico de clasificación

A continuación describimos la función de cada una de las etapas del proceso de reconocimiento de patrones detallado en la Figura 3-1:

- **Adquisición de datos:** esta etapa se encarga de la adquisición del patrón mediante el uso de transductores (sensores ambientales, cámaras, escáneres láser, micrófonos, etc.), a fin de disponer de una representación digital de los patrones, posibilitando

así el procesado de la información. Se puede prescindir de esta etapa en el caso que se trabaje con bases de datos (moléculas químicas activas, censos de población, etc.).

- **Preprocesado de los datos:** en esta etapa se pueden llevar a cabo diferentes tareas en función del tipo de datos del que se disponga. Por ejemplo en el caso de señales provenientes de la medida de magnitudes analógicas se debe preceder a un prefiltrado de la información a fin de eliminar el ruido y la señal parásita de la línea eléctrica (50Hz). En el caso de trabajar con imágenes digitales, se debe proceder a la segmentación del objeto de estudio del fondo de la imagen, a la eliminación del ruido de ésta, su normalización, etc.

- **Extracción/Selección de parámetros:** esta etapa se encarga de reducir los datos de entrada a una serie de características cualitativas o cuantitativas que los caractericen. Aunque en la mayoría de casos la función de selección y de extracción de las características se consideren equivalentes. La “extracción de parámetros” se refiere a la creación de nuevas características usando combinaciones de otras ya existentes, o transformaciones del conjunto de características ya existente. Por otro lado, la “selección” se refiere a la elección de un subgrupo de las características que existen sin ningún tipo de transformación.

- **Clasificación:** es la etapa del reconocimiento de patrones en la cual se procede al análisis de los parámetros adquiridos para la determinación de la clase o categoría a la cual pertenecen dichos parámetros. Las características de las señales permiten discriminar entre las diferentes clases existentes y reconocer las que son similares a una determinada clase o no.

- **Post procesado:** esta parte de la cadena del reconocimiento de patrones se encarga de evaluar el coste de la decisión del clasificador intentando minimizar el riesgo de

error de las diferentes herramientas, principalmente desde la perspectiva de las observaciones realizadas.

No obstante, las diversas etapas del reconocimiento de patrones en los sistemas reales muchas veces són difíciles de diferenciar, puesto que se encuentran combinadas entre si. El modo de operación de las diferentes partes del sistema no son unidireccionales, puesto que etapas posteriores del sistema pueden generar resultados que sirven para adaptar el comportamiento de etapas previas del mismo sistema (realimentaciones).

Una vez descritos los elementos genéricos que integran un sistema de reconocimiento de patrones pasaremos a describir los tipos de reconocimiento en función del tipo de aprendizaje usado para la evaluación de las categorías del sistema. En la práctica podemos afirmar que el reconocimiento de patrones se divide a su vez en dos grandes familias: el basado en el aprendizaje no supervisado y el supervisado.

En el caso del aprendizaje no supervisado las señales de entrada no se hallan etiquetadas (identificadas) y por lo tanto el sistema intentará establecer la organización de las clases basadas en las propiedades de los datos adquiridos. Por otro lado está el aprendizaje supervisado en el cual los datos de entrada tienen preasignadas unas etiquetas que nos servirán de conjunto de entrenamiento. A la vez debe tenerse presente que en función de las características del sistema a tratar, existen al menos dos formas estándares de afrontar el problema de la clasificación: la clasificación sintáctica (o estructural) y la estadística. La clasificación sintáctica (o estructural) se basa en la caracterización de la estructura inherente de las características cualitativas de los datos. Por esta razón, los patrones complejos se pueden descomponer mediante una estructura jerárquica en sub-patrones más simples, ya que cada patrón puede representarse mediante interrelaciones de los sub-patrones más simples, llamados “primitivas”. Esta aproximación puede explicarse con una analogía a la sintaxis de una lengua [3-3], donde las frases se puede considerar como un patrón complejo, siendo las primitivas del anterior patrón el alfabeto y las relaciones entre las primitivas vendría a ser la gramática. De esta forma cada uno de los patrones complejos

puede ser descrito con un número limitado de primitivas y de relaciones entre éstas. En cambio en el reconocimiento de patrones basado en la aproximación estadística los patrones se caracterizan mediante la estadística de las características cuantitativas de los datos de entrada. Esta última opción ha sido la adoptada por nuestra parte en las diferentes aportaciones realizadas en este campo, las cuales presentaremos en los siguientes subapartados. Por lo tanto, en el presente capítulo nos centraremos exclusivamente en la descripción de la aproximación estadística para el reconocimiento de patrones.

3.1.1 RECONOCIMIENTO DE PATRONES BASADO EN EL SUPUESTO DE LA APROXIMACIÓN ESTADÍSTICA

En el caso del reconocimiento de patrones basado en la aproximación estadística [3-1, 3-4] cada patrón se representa mediante un conjunto de características organizadas en un vector d-dimensional $x=(x_1, x2, x3,\ldots, xd)^T$. Dicho espacio se conoce con el nombre de espacio de configuraciones en el cual cada dimensión se corresponde con un descriptor. Es lógico esperar que los vectores de una misma clase formen un grupo relativamente compacto y lo suficientemente separado de las otras clases. Por lo tanto cada una de estas áreas presentes en el espacio de configuraciones puede describirse mediante una *función densidad de probabilidad* (o un peso en el caso de una distribución de probabilidad discreta) para cada clase, que se obtendrá a partir de un conjunto de datos de entrenamiento. Entonces, cada vector "**x**" perteneciente a la clase w_i (una de las **"n"** clases presentes en el sistema $\{w_1, w2, w3,\ldots, wn\}$) se puede interpretar como una medida realizada aleatoriamente de la probabilidad condicional de la función densidad de probabilidad "$p(x \mid w_i)$". Esto implica que en función de la información que se disponga a priori de las funciones de distribución de probabilidad se deberá proceder de una forma u otra a la hora de diseñar el clasificador. Las diferentes opciones de clasificación disponibles en la aproximación estadística para el reconocimiento de patrones se presentan de forma esquemática en la Figura 3-2 [3-3]. En ella se detallan las posibles soluciones en función del nivel de conocimiento de la forma de la distribución de probabilidad a priori.

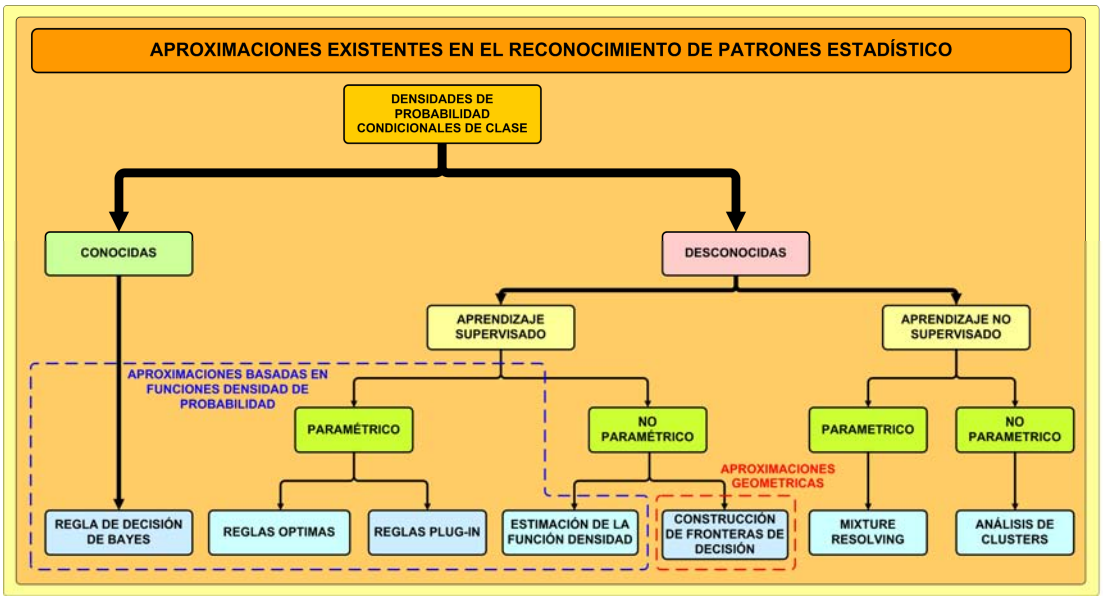

Figura 3-2: Aproximaciones en el reconocimiento de patrones estadístico

En el caso que todas las funciones densidad de probabilidad condicional para cada una de las clases sean conocidas, entonces la regla de Bayes puede ser usada para el diseño e implementación del clasificador. No obstante, generalmente no se dispondrá de estas distribuciones a priori, y por lo tanto deberán ser estimadas a partir de la información o datos existentes. Si la forma de la distribución es conocida (por Ej.: el nivel de ruido en telecomunicaciones es sabido que viene regido por una distribución normal) pero algunos parámetros de ésta són desconocidos, implicará la necesidad de hacer uso de una solución *paramétrica;* a fin de determinar estos parámetros desconocidos. Por el contrario si la forma de la distribución es desconocida, se deberá hacer uso de una técnica *no paramétrica*; a fin de evaluar la distribución de nuestro problema. Esta solución es la aproximación más general de proceder en la resolución de un problema basado en aprendizaje supervisado. En este caso se puede proceder a estimar las funciones densidad de probabilidad (por ejemplo, haciendo uso de la técnica de las ventanas de *Parzen*) o directamente intentar construir las fronteras de decisión entre las diferentes clases (por ejemplo, haciendo uso de la técnica de los k-próximos vecinos "**k-NN**"). Las soluciones anteriormente propuestas representan a su vez un claro ejemplo de otra división existente entre las aproximaciones basadas en estimar las funciones de densidad de probabilidad para

luego poder evaluar las fronteras de decisión mediante alguna función discriminante (consiste en una función encargada de determinar si una muestra pertenece a una clase u otra); y las basadas en aproximaciones geométricas que pretenden evaluar las fronteras de decisión entre clases a partir de los datos de entrenamiento. De todas formas, hay que tener muy presente que en algunas circunstancias las dos aproximaciones són equivalentes.

A lo largo de los siguientes subapartados, se presentan de forma resumida las principales técnicas de extracción de parámetros y clasificación, ya que conforman el núcleo del problema asociado al reconocimiento de patrones estadístico.

3.1.1.1 La extracción de parámetros

La extracción de parámetros es una de las partes más complejas y difíciles de implementar en un sistema estadístico para la clasificación de patrones. Su función primordial es determinar el conjunto de características numéricas que mejor representan los datos de entrada al sistema, ya que si las características son representativas de la clase, a la vez que discriminantes entre clases, la tarea del clasificador será mucho más fácil. A la vez es deseable que estas características sean invariantes ante rotaciones, translaciones y tamaño de los objetos a clasificar. En el caso que éstas no lo sean, los datos de entrada pueden ser tratados en una etapa previa de preprocesado para hacerlos invariantes. Esta es la razón que hace que la selección de las características sea el elemento más crítico del sistema, en vista que depende totalmente de las características del patrón a reconocer, mientras que las herramientas de clasificación són elementos más bien ya estandarizados.

Otro de los mecanismos de la extracción de parámetros es la reducción de la dimensionalidad del problema. Mediante esta reducción se pretende simplificar la representación de los datos y en consecuencia hacer que el trabajo del clasificador sea más liviano. A la vez, hay que tener mucho cuidado en no reducir en demasía la capacidad de discriminación del sistema, puesto que una reducción demasiado importante de los datos puede producir una pérdida de información.

Por lo tanto, la extracción de parámetros intenta transformar linealmente o no-linealmente los vectores "**x**" del espacio d-dimensional original $\left(x \in \Re^d\right)$ en un sub-espacio m-

dimensional, en el cual $m \leq d$. Las principales técnicas lineales conocidas y usadas són [3-1, 3-5, 3-6, 3-7] las siguientes: el Análisis del Componente Principal (PCA), el análisis del discriminante lineal (LDA) y el análisis de Componente Independiente (ICA). La técnica (PCA) "*Principal Component Analisys*" también conocida como la transformación Karhunen-Loêve (KTL), es una transformación lineal que permite representar un patrón como la combinación de un conjunto significativo de vectores propios (los que se corresponden con los valores propios más grandes) obtenidos de la matriz de covarianza del conjunto de patrones. Por otra parte, la técnica (ICA) "*Independent Component Analisys*" es similar a la (PCA) excepto que ha sido diseñada para permitir representar distribuciones de componentes no-gaussianos o no-normales. Finalmente la técnica (LDA) "*Linear Discriminant Analisys*" también conocida como Discriminante Lineal de Fisher (FLD), hace uso de las clases de información de las muestras para intentar maximizar la separación entre clases, mientras que minimiza la dispersión dentro de la propia clase.

Con respecto a las transformaciones no-lineales, es muy común el uso del llamado "*kernel trick*". Esta técnica se fundamenta en mapear las observaciones no-lineales obtenidas (originales) en un espacio de mayor dimensión que el original, para poder hacer uso de una transformación lineal. La técnica más conocida y usada es el *kernel PCA* (KPCA) [3-8, 3-9], aunque existen otras versiones no lineales. Además de éstas existen otras técnicas o herramientas como son: las (MDS) "*MultiDimensional Scaling*" [3-11], las redes neuronales, Mapa de Auto-Organización (SOM) o *mapas de Kohonen* [3-10], así como algunas otras técnicas más minoritarias que no hemos abordado en este apartado.

3.1.1.2 La clasificación

Una vez extraídos los parámetros la clasificación es la siguiente etapa en el diseño de un sistema de reconocimiento de patrones. Como ya hemos comentado anteriormente la elección de un clasificador no es tan dependiente de la aplicación como la extracción de parámetros, y en la mayor parte de los casos se realizará en función de la disponibilidad del algoritmo para su implementación en un sistema dado. En la práctica existen tres grandes aproximaciones para el diseño de clasificadores en función del concepto sobre el cual se basan: la similaridad, la geometría y la probabilidad (estas técnicas se describen en detalle

en las siguientes publicaciones [3-1, 3-5, 3-6, 3-7]). Cabe remarcar que existen además de las descritas otras aproximaciones que no se detallan en el presente capítulo al ser su relevancia mucho menor.

El primer tipo de clasificador que abordaremos es el basado en la similaridad, el cual se cimienta en la suposición que patrones similares deben corresponderse con una misma clase. Una de las técnicas basadas en este concepto es el "*template matching*" [3-12], que consiste en comparar el patrón a reconocer con una serie de perfiles almacenados a fin de conseguir una estimación de la similaridad de la medida como si se tratase de una correlación. Otra opción es el uso de clasificador de *la mínima distancia*, en el cual se elige una métrica (por ejemplo: la distancia euclidiana) y un prototipo para la clase (como puede ser la media de las medidas de cada una de las clases). Por lo tanto la clase cuya distancia sea mínima entre la muestra y la muestra prototipo será la elegida.

El segundo tipo de clasificador es el basado en la geometría entre clases. Estos clasificadores intentan estimar las fronteras de decisión, que se corresponden con los hiperplanos que separan las diversas clases obtenidos mediante la optimización de algunos criterios de error de la distribución. Los ejemplos más representativos de este tipo de clasificadores son el "*Support Vector Machine*" (SVM) [3-13] o las redes neuronales basadas en redes de *perceptrón* de una sóla capa.

El tercer y último tipo de clasificador es el basado en la aproximación estadística, el cual dependiendo de la información suministrada por las variables tomará una forma u otra (existen diferentes soluciones como se muestra en la Figura 3-2). En el caso de no disponer de ninguna información adicional, sólo se podrá generar un mecanismo de decisión entre clases basado en la probabilidad de la clase a priori $p(w_i)$ (obtenida a partir de un número suficientemente grande de muestras aleatorias) y siendo el mecanismo de decisión elegido, la clase con la mayor probabilidad de todas:

$$p(w_i) > p(w_j),\ 1 \le \mathrm{i, j} \le \mathrm{c(\#clases)},\ \mathrm{i} \ne \mathrm{j} \qquad \text{Ecuación [3-1]}$$

No obstante el objetivo primordial de toda técnica estadística es la obtención de la densidad de probabilidad condicional a posteriori $p(w_i \mid x)$, que se corresponde con la probabilidad que una muestra pertenezca a una clase w_i dado el hecho de haberse medido la variable *x*. La forma de obtener estas densidades es lo que diferencian las diferentes técnicas existentes, pero una vez obtenidas estas distribuciones la regla de Bayes se aplicará para obtener el mínimo error en la estimación, tal y como se muestra en la siguiente expresión (Ecuación [3-2]):

$$p(w_i \mid x) > p(w_j \mid x),\ 1 \le i, j \le c(\#\text{clases}),\ i \ne j,\ x \in w_i \qquad \text{Ecuación [3-2]}$$

Hay que tener presente que en la mayor parte de los casos sólo seremos capaces de estimar las densidades de probabilidad de la clase condicional $p(x \mid w_i)$ (también conocida como similaridad), y mediante la regla de Bayes obtendremos las probabilidades a posteriori (Ecuación [3-3]):

$$p(w_i \mid x) = \frac{p(x \mid w_i) \cdot p(w_i)}{p(x)} \qquad \text{Ecuación [3-3]}$$

En esta expresión puede apreciarse cómo el denominador no depende de la clase, por lo tanto podemos simplificar la expresión para la obtención del clasificador de mínimo error que se regirá por la Ecuación [3-4].

$$p(x \mid w_i) \cdot p(w_i) > p(x \mid w_j) \cdot p(w_j),\ 1 \le i, j \le c(\#\text{clases}),\ i \ne j,\ x \in w_i \qquad \text{Ecuación [3-4]}$$

De aquí se deduce que el clasificador óptimo puede diseñarse si se conocen las probabilidades de suceso a priori $p(w_i)$ y las densidades de probabilidad condicionales a priori de cada una de las clases $p(x \mid w_i)$. Sin embargo, hay que tener muy presente que en una aplicación real estas probabilidades no suelen estar disponibles y la clasificación tan sólo puede realizarse a partir de algunos conocimientos previos del problema y de las muestras de entrenamiento. En el caso de conocer la forma paramétrica de las densidades de probabilidad de las clases condicionales, la tarea se reduce a encontrar los parámetros que caracterizan la distribución (por ejemplo: en el caso de una distribución Gaussiana ésta vendrá caracterizada por la media y la varianza). Ambos métodos son comúnmente usados [3-1], mientras que el método de la máxima verosimilitud busca los parámetros que mejor

se ajustan a los datos de entrenamiento de cada una de las clases. La estimación Bayesiana trata los parámetros como variables aleatorias con una densidad de probabilidad a priori que mediante los datos de entrenamiento se convierten en la función densidad de probabilidad condicional a posteriori.
Las distribuciones Gaussianas multivariables son usadas comúnmente en las estimaciones paramétricas. Dependiendo de las suposiciones realizadas sobre las matrices de covarianza, se pueden hallar diferentes soluciones. Por ejemplo en el caso de considerarse que todas las matrices sean iguales para todas las clases la regla de Bayes nos proporciona un clasificador lineal. En el caso que las matrices para cada clase sean diferentes se obtendrá un clasificador cuadrático.

En el caso de las soluciones no paramétricas existen dos aproximaciones: las basadas sobre el clasificador clásico llamado *k-próximos vecinos* (k-NN) y el clasificador basado en la técnica de las *ventanas de Parzen*. En el primer caso el clasificador (k-NN) "*k-Nearest Neighbours*" se basa en hallar la distancia del vector a clasificar con los vectores de los k-próximos vecinos. Una vez calculada esta distancia, se procede a evaluar cuál ha sido la clase de los k próximos vecinos que más ha sido identificada, y se procede a asignar esta clase a los datos testeados.

El clasificador basado en la técnica de las *ventanas de Parzen* impone una función ventana sobre cada muestra de entrenamiento para conseguir así una estimación de la función densidad de cada una de las clases (éste es el motivo por el cuál se conoce este método como *Ventana de Parzen* o estimación de la densidad del kernel). Generalmente se suelen usar funciones gaussianas para la generación de la función densidad aunque también pueden usarse otros tipos de funciones como son las uniformes o las triangulares. Entonces el clasificador asigna a una determinada muestra la clase que obtiene la mayor probabilidad basándose sobre las funciones densidad de todas las clases procesadas en el entrenamiento. Una diferencia sustancial entre ambos estimadores versa en el hecho que el método basado en las *ventanas de Parzen* evalúa la probabilidad condicional de la distribución " $p(x \mid w_i)$"

para una determinada clase, mientras que el método “k-NN” evalúa directamente el valor de la probabilidad a posteriori “ $p(w_i \mid x)$” de una determinada clase.

Además de los tres tipos de clasificadores hasta ahora descritos, existen otros clasificadores como los árboles de decisión y las soluciones basadas en redes neuronales, que no abordaremos en el presente capítulo.

Para concluir con este subapartado es conveniente indicar que los clasificadores pueden ser vistos desde otro punto de vista, como pueden ser las funciones discriminantes. Un clasificador puede determinarse mediante un conjunto “**c”** (uno para cada clase) de funciones discriminantes $\left(g_i(x): \Re^n \rightarrow \Re, 1 \leq \mathrm{i} \leq \mathrm{c}\right)$ que proporcionan a su vez un conjunto de resultados numérico o marcadores para cada una de las clases de una determinada muestra “**x”**; a la cual se le suele asignar como clase la que haya obtenido el mayor marcador de todas las funciones discriminantes. Por lo tanto, lo que hacen las funciones discriminantes es dividir el espacio $\Re^n$, en “c” regiones de decisión (zonas de clasificación) las cuales son delimitadas por las fronteras de clasificación.
Con esta aproximación se pueden usar las estimaciones de las densidades de probabilidad a posteriori como funciones discriminantes:

$$g_i(x) = p(x \mid w_i) \cdot p(w_i) \qquad \text{Ecuación [3-5]}$$

Por lo tanto, se usarán funciones lineales, cuadráticas o de kernel para el clasificador basado en las ventanas de *Parzen*, y funciones basadas en distancias (Euclidiana, Manhattan,..) para el clasificador “**k-NN**”.

3.2 EL RECONOCIMIENTO DE PATRONES ESTOCÁSTICO EN EL SUPUESTO DE DENSIDADES DE PROBABILIDAD CONDICIONALES CONOCIDAS

En este subapartado se aborda la implementación estocástica de los diferentes mecanismos de clasificación desarrollados para el reconocimiento de patrones estocástico, evaluados en el supuesto de densidades de probabilidad condicionales conocidas. Adicionalmente en este subapartado se abordan también diversas implementaciones estocásticas para la generación de las densidades de probabilidad condicionales a priori (f.d.p) de las diferentes clases $p(x \mid w_i)$ de un sistema.

3.2.1 INTRODUCCIÓN

La naturaleza probabilística de la lógica estocástica (descrita a lo largo del segundo capítulo de esta tesis) aporta una ventaja inherente a la hora de implementar metodologías probabilísticas de reconocimiento de patrones. Por lo tanto antes de proceder a la descripción de las diversas implementaciones desarrolladas, realizaremos una breve introducción a las variables aleatorias así como a los mecanismos bayesianos de clasificación de éstas.

Los procesos de toma de decisiones bajo incertidumbre (la gran mayoría de los procesos con los que trataremos) son aquellos en los que los parámetros varían con el tiempo y obedecen a procesos estocásticos. Los mecanismos de toma de decisiones utilizan algunos conceptos de la estadística, por ello es necesario que recordemos algunos de estos conceptos. Por ejemplo, sea un experimento dado "E" el cual se repite "n" veces, donde el conjunto de patrones "Ω" esta formado por dos clases distintas "$\{w_1, w_2\}$" dispondremos de la siguiente información (Ecuación [3-6]):

$$\begin{cases} P(A) = {}^{n_A}\!/_{n} \rightarrow \text{Probabilidad de ocurrencia de A} \\ P(B) = {}^{n_B}\!/_{n} \rightarrow \text{Probabilidad de ocurrencia de B} \\ P(A \cap B) = {}^{n_{A \cap B}}\!/_{n} \rightarrow \text{Probabilidad de ocurrencia de A y B} \end{cases} \rightarrow \begin{cases} \underset{n \to \infty}{Lim} P(B \mid A) = \dfrac{P(A \cap B)}{P(A)} = \\ = \dfrac{{}^{n_{A \cap B}}\!/_{n}}{{}^{n_A}\!/_{n}} = \dfrac{n_{A \cap B}}{n_A} \end{cases}$$

Ecuación [3-6]

Donde "$P(B \mid A)$" se corresponde a la probabilidad condicional que se de "B" dado que ya se dio "A". Con todo ello, la teoría de decisión de Bayes se basa en el hecho que el conjunto de todos los patrones "Ω" a reconocer son elementos potenciales de las "J" clases distintas "$w_j, j = 1,2,....,J$", de tal forma que las propiedades del *conjunto de las clases informacionales* vienen descritos mediante la Ecuación [3-7]:

$$\Omega = \{w_1, w_2, ..., w_J\}$$

Ecuación [3-7]

$$\text{Propiedades} \rightarrow \begin{cases} w_i \cap w_j = \phi \\ \bigcup_{i=1}^{J} w_i = \Omega \end{cases}$$

Donde la probabilidad a priori se corresponde con el conocimiento a priori acerca de una clase antes de realizar experimento alguno. A esta probabilidad la denotaremos como "$P(w_i)$" y se regirá por las siguientes propiedades estadísticas (Ecuación [3-8]):

$$P(w_i) = \pi_i, \text{i} = 1,2,...,\text{J}$$

$$\text{Propiedades} \rightarrow \begin{cases} P(w_i) = \pi_i \geq 0, \text{ j} = 1,2,...,\text{J} \\ \sum_{i=1}^{J} P(w_i) = \sum_{i=1}^{J} \pi_j = 1 \end{cases}$$

Ecuación [3-8]

Siendo la regla de decisión trivial para un conjunto de patrones formado por dos clases "$\Omega = \{w_1, w_2\}$", la que consiste en determinar cuál de las dos tiene asociada una mayor probabilidad a priori de darse; tal y como se describe a continuación (Ecuación [3-9]):

$$\begin{cases} \text{clase } w_1 \text{ si: } \pi_1 > \pi_2 \\ \text{clase } w_2 \text{ si: } \pi_1 < \pi_2 \end{cases}$$

Ecuación [3-9]

No obstante, la probabilidad de error en un sistema constituido por dos clases se corresponderá con el valor de la probabilidad menor de las dos categorías comparadas (π_1

o π_2). En vistas de esto, se hace necesario incorporar *evidencias* a la toma de decisiones. Por tanto es necesario disponer de una experiencia (*prototipos*) y de un patrón (*evidencia*) para poder clasificar. Por ello, deberemos tener siempre una serie de consideraciones muy presentes, como son: que los valores de los patrones asociados a una determinada clase deben ser sustancialmente diferentes a los de otra clase. Los patrones asociados a una misma clase tienen asociada una determinada variabilidad, por ello no són exactamente iguales. En consecuencia, la variabilidad en las observaciones se expresa en términos probabilísticos como que "x" es una variable aleatoria cuya distribución depende de la clase "w_i". La función densidad de probabilidad de "x" en el supuesto que su clase sea "w_i", la denotaremos como "$P(x \mid w_i)$" siendo la función densidad de probabilidad condicionada de "x". Entonces para cada clase "w_i" deberá verificarse que su función densidad de probabilidad se halle normalizada a la unidad (aunque podría ser a un determinado valor diferente a "1"), tal y como se indica en la Ecuación [3-10].

$$\int_x P(x \mid w_i) = 1 \qquad \text{Ecuación [3-10]}$$

Otra regla de decisión trivial para un conjunto de patrones formado por dos clases "$\Omega = \{w_1, w_2\}$" se corresponde en determinar para una observación "x" cuál de las dos funciones densidad de probabilidad condicionada de "x" es mayor.

$$\begin{cases} \text{clase } w_1 \text{ si: } P(x \mid w_1) > P(x \mid w_2) \\ \text{clase } w_2 \text{ si: } P(x \mid w_1) < P(x \mid w_2) \end{cases} \qquad \text{Ecuación [3-11]}$$

Esta forma de decisión es mucho mejor que la anterior aunque no se corresponde con la solución que minimiza el error de decisión.

Podremos determinar cómo influye este conocimiento sobre la decisión acerca de la clase esperada correspondiente a una determinada medida "x" mediante lo que se conoce como probabilidad a posteriori ($P(w_i \mid x)$). La cual debe interpretarse como la probabilidad que la clase cierta sea "w_i" dado que el valor observado/medido es "x".

Por lo tanto, la Regla de Bayes (Ecuación [3-12]) [3-17] muestra cómo el valor observado/medido "x" modifica la decisión basada en "$P(w_i)=\pi_i$", a través de la probabilidad condicional "$P(x \mid w_i)$" introduciendo la *probabilidad a posteriori* "$P(w_i \mid x)$"

$$\begin{cases} P(w_i \mid x)=\dfrac{P(w_i)\cdot P(x \mid w_i)\cdot}{P(x)} \\ P(x)=\sum_{i=1}^{J} P(w_i)P(x \mid w_i) \end{cases} \rightarrow P(w_i \mid x)=\frac{P(w_i)\cdot P(x \mid w_i)}{\sum_{j=1}^{J} P(w_i)\cdot P(x \mid w_i)} \qquad \text{Ecuación [3-12]}$$

En cualquier caso deberán cumplirse siempre las siguientes propiedades (Ecuación [3-13]):

$$\begin{cases} \text{Si } P(x \mid w_i)=0, \mathrm{P}(w_i \mid x)=0 \text{ y } \sum_{\mathrm{j}=1}^{\mathrm{J}} P(w_j \mid x)=1 \ \forall\, i \neq j \\ \text{Si } P(x \mid w_1)=\cdots\cdots=P(x \mid w_J) \rightarrow \mathrm{P}(w_1 \mid x)=\cdots\cdots=\mathrm{P}(w_J \mid x)=\dfrac{1}{\mathrm{J}} \\ \forall x, \sum_{\mathrm{i}=1}^{\mathrm{J}} \mathrm{P}(w_i \mid x)=1 \end{cases} \qquad \text{Ecuación [3-13]}$$

La regla de clasificación de Bayes sigue la inclinación natural de asignar a la muestra "x" la clase para la cual la probabilidad a posteriori sea mayor "$P(x \mid w_i)$", de tal forma que se cumpla la Ecuación [3-14]. No obstante esta regla minimiza la probabilidad media de error.

$$P(w_k \mid x) > P(w_i \mid x), \forall\, \mathrm{i} \neq \mathrm{k} \qquad \text{Ecuación [3-14]}$$

Siendo una equivalente a la anterior regla descrita por la Ecuación [3-15]:

$$P(x \mid w_k)\cdot P(w_k) > P(x \mid w_i)\cdot P(w_i), \forall\, \mathrm{i} \neq \mathrm{k} \qquad \text{Ecuación [3-15]}$$

En el hipotético caso en el que "$P(w_k)=P(w_i), \forall\, \mathrm{i} \neq \mathrm{k}$" a la ecuación [3-15] se la conoce como *la regla de máxima verosimilitud*.

Siendo uno de los grandes problemas de la regla de clasificación a posteriori el hecho de considerar las probabilidades a priori en su evaluación, ya que las clases muy infrecuentes resultan seriamente castigadas. En el caso que no se conozcan las densidades de probabilidad de los atributos de cada clase pero sí las a priori, no podrá usarse la regla de

Bayes. En este caso deberá hacerse uso de técnicas geométricas, como puede ser la distancia euclidiana a la media de cada clase.

Si estimamos el error en la clasificación, por ejemplo en el caso de un conjunto de dos patrones "$\Omega = \{w_1, w_2\}$" unidimensionales, hallaremos que las causas que pueden producir un error de clasificación son basicamante que una determinada medida se halle en una región con máxima probabilidad a posteriori de la clase contraria a la cual pertenece dicha medida. Donde estos hechos son mutuamente exclusivos y exhaustivos. Por lo tanto el error de clasificación vendrá dado por la siguiente expresión (Ecuación [3-16]):

$$P(Error) = P(x \in R_2, w_1) + P(x \in R_1, w_2) = P(x \in R_2 \mid w_1) \cdot P(w_1) + P(x \in R_1 \mid w_2) \cdot P(w_2) =$$
$$= \int_{R_2} P(x \mid w_1) \cdot P(w_1) \cdot dx + \int_{R_1} P(x \mid w_2) \cdot P(w_2) \cdot dx$$

Ecuación [3-16]

La evaluación de estas integrales es desafortunadamente casi siempre imposible de realizar. Esto es debido a que sabemos que el clasificador de Bayes es óptimo pero no podemos determinar con exactitud cuál es su bondad. Por lo tanto, la única solución para la determinación del error será el uso de *estimadores*, siendo las bases de la estimación la existencia de un conjunto de datos de entrenamiento "T" compuesto por "N" prototipos de las "J" clases. Dicho conjunto de entrenamiento "T" se usará para construir y evaluar el clasificador. La estimación del error se calculará a partir de la proporción de prototipos "(X_i, c_i)" incorrectamente etiquetados "$\Delta(X_i, c_i)$" por el clasificador "d":

$$\Delta(X_i, c_i) = \begin{cases} 1 \text{ si } d(x_i) \neq c_i \text{ (error)} \\ 0 \text{ si } d(x_i) = c_i \text{ (acierto)} \end{cases} \quad \text{Ecuación [3-17]}$$

Siendo las tres técnicas más comunes de estimación las siguientes: estimación por re-sustitución, estimación mediante un conjunto de prueba y estimación por validación cruzada.

3.2.2 IMPLEMENTACIÓN ESTOCÁSTICA DE LAS FUNCIONES DENSIDAD DE PROBABILIDAD CONDICIONAL

El primer paso para poder representar el comportamiento de una población estadística de observaciones/datos/medidas es disponer de suficientes valores a partir de los cuales poder representar su distribución de probabilidad [3-5]. Existen multitud de funciones de distribución teóricas, cada una de ellas basadas en un modelo de compartimiento del proceso que generó el universo de observaciones.

De ello se deriva que la aplicación de una de estas distribuciones teóricas a una determinada población estará justificada si las hipótesis (o suposiciones) del modelo de comportamiento del proceso que generó la población se cumplen. Por lo tanto, si conocemos el fenómeno que dio lugar a una determinada población de mediciones u observaciones entonces podremos afirmar que la distribución de probabilidades de dicha población se ajusta a un determinado modelo teórico (binomial, exponencial, Weisbull, Gauss, etc.).

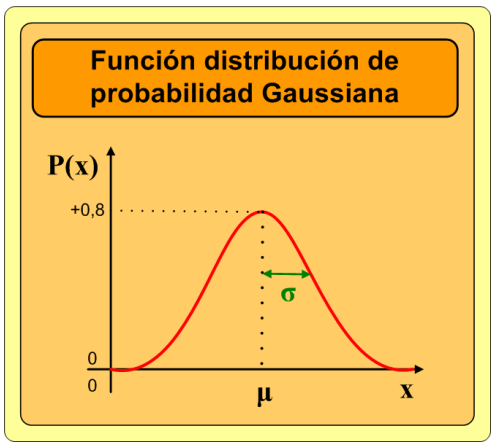

Figura 3-3: Función distribución de probabilidad Gaussiana (Definida mediante una media (μ) y una desviación estándar (σ))

Por ejemplo, en la práctica se sabe que el número de partículas alfa emitidas por un material radioactivo se rige por una distribución de Poisson. Al igual que por ejemplo: el nivel de ruido en las telecomunicaciones, el efecto de un fármaco después de su ingestión,

la distribución de las características morfológicas de los individuos, así como muchos otros fenómenos se rigen por una distribución normal o también conocida como distribución Gaussiana. Una distribución de probabilidad normal [3-14] o de Gauss (Figura 3-3), se caracteriza por una curva en forma de campana con un eje de simetría en el punto correspondiente al promedio del universo (μ), así como por una distancia entre el eje de simetría de la campana al punto de inflexión de la curva (σ) que se corresponde con la desviación estándar de la distribución.

La expresión matemática que describe dicha función densidad de probabilidad (**f.d.p**) se presenta en la Ecuación [3-18]. Dicha distribución tiene la peculiaridad que el área debajo de la curva se halla normalizada a "+1". A la vez el área encerrada por la curva comprendida entre (μ-σ) y (μ+σ) es aproximadamente igual a "68%" del área total, mientras que el área comprendida entre (μ-2σ) y (μ+2σ) es aproximadamente el "95%" del área total.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in \Re \qquad \text{Ecuación [3-18]}$$

El parámetro "**μ**" se corresponde con el promedio o la media de todas las observaciones (N observaciones), la cual se determina mediante la Ecuación [3-19]. Para caracterizar la dispersión de la distribución se usa la varianza "σ²" (Ecuación [3-20]) y la desviación estándar "σ" (Ecuación [3-21]).

La varianza "σ²" es una magnitud que nos permite comparar poblaciones, mientras que la desviación típica "σ" tiene las mismas unidades que las medidas con las que estamos evaluando la dispersión. Ambas permiten comparar el grado de dispersión de las distintas poblaciones.

$$Media = \mu = \frac{\sum_{i=1}^{N}(x_i)}{N} \qquad \text{Ecuación [3-19]}$$

$$Varianza = \sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \mu)^2}{N} \qquad \text{Ecuación [3-20]}$$

$$Deviación\_estándar = \sqrt{Varianza} = \sigma = \sqrt{\frac{\sum_{i=1}^{N}\left(x_i - \mu\right)^2}{N}} \qquad \text{Ecuación [3-21]}$$

Para concluir con esta introducción, debe remarcarse que todas las metodologías de reconocimiento de patrones basadas en el supuesto de distribuciones densidad de probabilidad conocidas que se presentan a lo largo de los siguientes subapartados se basan en el supuesto que la población de medidas u observaciones de una determinada clase o población viene regida mediante una *distribución Normal o de Gauss.*

El objetivo del siguiente sub-subapartado es presentar los diferentes desarrollos realizados de funciones densidad de probabilidad estocástica. Inicialmente describiremos la implementación de la función *f.d.p estocástica pseudo normal* que fue la primera aproximación estocástica que desarrollamos y usamos en un sistema práctico de reconocimiento de patrones. Seguidamente pasaremos a describir la *f.d.p estocástica Gaussiana* desarrollada posteriormente para subsanar algunas de las deficiencias presentes en la primera implementación realizada (cuya implementación ya ha sido descrita mayormente en el capítulo segundo de la presente tesis).

3.2.2.1 Implementación estocástica de una f.d.p pseudo-normal

Para poder evaluar el buen funcionamiento de una determinada metodología de reconocimiento de patrones lo más sencillo es proceder a testearla sobre densidades de probabilidad conocidas. Éste es el motivo que nos ha llevado a desarrollar un circuito estocástico (lo más simple posible) capaz de emular a su salida una función densidad de probabilidad normal. Está deberá dotarse de una serie de parámetros de ajuste como son: una media "η" y una dispersión "σ_{ref}", como bien se ha presentado en la Figura 3-3. Dicho circuito se ha implementado mediante la lógica estocástica clásica sin signo (codificación unipolar) (UCLS).

El circuito desarrollado que se presenta en la Figura 3-5 se corresponde al primer desarrollo [3-15, 3-16] que realizamos antes de conseguir un bloque estocástico que implementase la

función Gaussiana (puramente normal), la implementación de la cual se ha presentado en el capítulo segundo de la presente tesis (apartado 2.2.3.9).

El principio de operación del bloque *función densidad de probabilidad pseudo-normal* desarrollado requiere de una señal pulsante "x" de entrada (señal estocástica), la cual es integrada temporalmente con un bloque P2B(N). Mediante el ajuste del número de ciclos de integración "N" del bloque P2B(N) se podrá fijar la varianza de la distribución "σ", siendo la salida del bloque un valor digital de n-bits (en nuestra implementación han sido 16-bits) que representa la probabilidad que se de un determinado suceso en el intervalo [0, +1].

$$OUT\left[P2B(N)\right] = P_N(x) = \binom{N}{x} \cdot p^N \cdot (1-p)^{N-x} \qquad \text{Ecuación [3-22]}$$

Como bien ya hemos descrito en el capítulo segundo de la presente tesis, la salida de los bloques P2B(N) viene regida por una distribución binomial (Ecuación [3-22]). A su vez en la Figura 3-4 se presenta la distribución de probabilidad de la salida de un bloque B2P(N) con un valor de "N=20" al cual se le ha fijado la probabilidad de la señal de entrada "p=0.5" y se ha procedido a variar la probabilidad de ocurrencia de "X" entre 0 y 20. Como bien puede apreciarse el área encerrada por esta distribución igual a "+1".

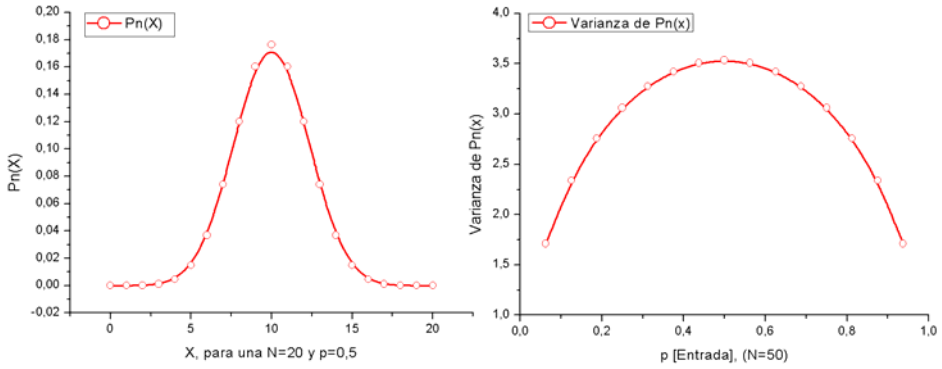

Figura 3-4: (Izquierda) Distribución binomial Pn(X) para (p=0,5 fija, una N=20 y variando X entre [0..20]), (Derecha) varianza de la distribución en función de la probabilidad de entrada

La salida (entrada A del comparador) de n-bits del bloque P2B(N) se comparará con un parámetro de n-bits que representa la media "μ" de nuestra distribución (entrada B del comparador) acotada en el intervalo [0, +1] (correspondiéndose este espacio de representación a un valor digital comprendido entre 0 y N), obteniéndose así una estimación de la probabilidad de coincidencia entre la media de la distribución "μ" y el valor binario equivalente de la señal pulsante de la entrada (obtenida después de "N" ciclos de integración). Todo ello realizado con un simple comparador digital de (n-bits), cuya salida se hallará a nivel alto '1' si A=B y en el resto de casos a nivel bajo '0'.

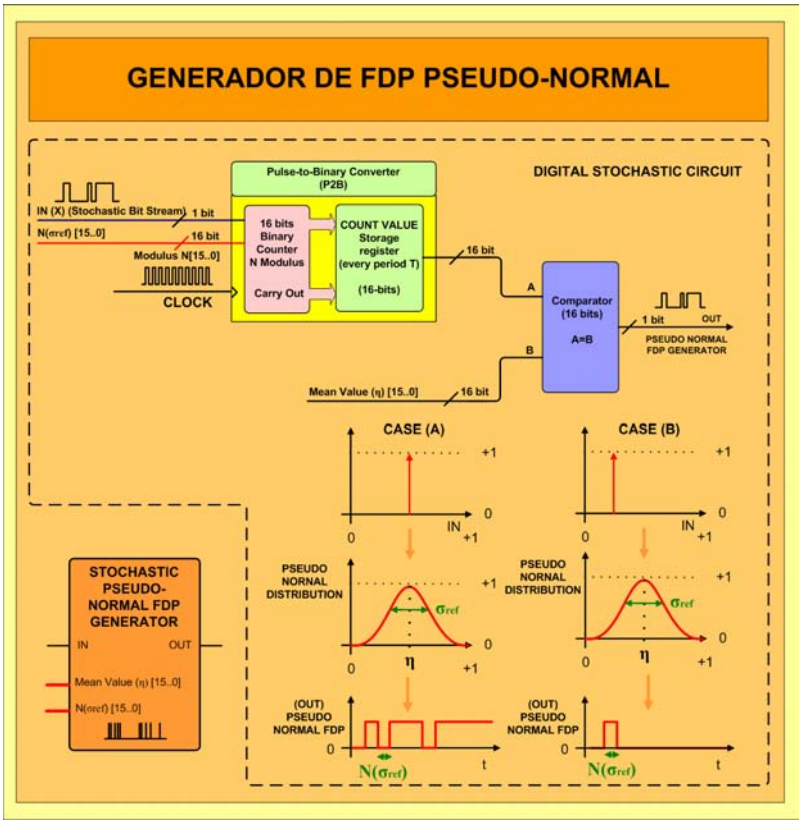

Figura 3-5: Circuito digital que implementa la *fdp* pseudo-normal estocástica

La salida del comparador (1-bit) a su vez se regirá por una distribución binomial (Ecuación [3-23]) la cual se aproximará bastante bien a una distribución normal si la desviación estándar (σ_{ref}) asociada a la distribución del P2B(N) original es lo suficientemente grande.

La desviación estándar de la distribución "$\sigma_{\mathrm{Re}f}$" aumenta a medida que "N" se hace pequeño.

$$out\_comparador = P_N(\mu) = \binom{N}{\mu} \cdot p^N \cdot (1-p)^{N-\mu}, \quad \forall\ 0 \le \mu \le N \quad \text{Ecuación [3-23]}$$

La cual a su vez será evaluada "$2^{16}/N$" veces en un período de valuación del sistema que consistirá en "2^{16}" ciclos de reloj, si se realiza mediante un módulo P2B(N) de 16-bits. Esto conlleva que el valor esperado del bloque vendrá dado por la siguiente expresión (Ecuación [3-24]):

$$E[out\_bloque] = P_N(\mu) \cdot \frac{2^{16}}{N} = \left(\binom{N}{\mu} \cdot p^N \cdot (1-p)^{N-\mu} \right) \cdot \frac{2^{16}}{N}, \quad \forall\ 0 \le \mu \le N$$

Ecuación [3-24]

Siendo la señal pulsante de la salida del comparador (Figura 3-5) proporcional a la probabilidad que la función (configuración) se encuentre en la categoría definida por la distribución binomial centrada en la señal de referencia (la media "μ"). Entonces a partir de la teoría fundamental de probabilidades, la dispersión relativa a la distribución $(\sigma_{\mathrm{Re}f} / N)$ debe ser inversamente proporcional al valor de "N" elegido:

$$\frac{\sigma_{\mathrm{Re}f}}{N} \approx \frac{\sqrt{p \cdot (1-p)}}{\sqrt{N}} \qquad \text{Ecuación [3-25]}$$

Por lo tanto, valores de N grandes/pequeños implicarán una baja/alta dispersión de la distribución.

Una vez descrito el principio de operación del *bloque f.d.p pseudo normal* pasaremos a presentar la evaluación experimental del circuito digital propuesto (Figura 3-5). Para ello empezaremos evaluando la correcta operación del bloque, fijando los parámetros "N=100" y "$\mu = 50 \rightarrow \mu = 50/N = 0{,}5$". Seguidamente variaremos la probabilidad de la señal estocástica "IN" de entrada al circuito en el intervalo [0, +0,9375]. Los resultados experimentales obtenidos se presentan en la Tabla 3-1:

Tabla 3-1: Medidas experimentales del bloque f.d.p pseudo normal							
Experimental [Datos de entrada]					Experimental [datos de salida]		
IN [16bits]	Valor IN	N	μ [16bits]	Valor μ	OUT [16bits]	Valor OUT	Área
0	0,0000	100	50	0,5	0	0,0000	0,0000
4096	0,0625	100	50	0,5	0	0,0000	0,0000
8192	0,1250	100	50	0,5	0	0,0000	0,0000
12288	0,1875	100	50	0,5	0	0,0000	0,0000
16384	0,2500	100	50	0,5	0	0,0000	0,0000
20480	0,3125	100	50	0,5	101	0,0015	0,0031
24576	0,3750	100	50	0,5	909	0,0139	0,0277
28672	0,4375	100	50	0,5	2727	0,0416	0,0832
32768	0,5000	100	50	0,5	4949	0,0755	0,1510
36864	0,5625	100	50	0,5	2110	0,0322	0,0644
40960	0,6250	100	50	0,5	808	0,0123	0,0247
45056	0,6875	100	50	0,5	217	0,0033	0,0066
49152	0,7500	100	50	0,5	0	0,0000	0,0000
53248	0,8125	100	50	0,5	0	0,0000	0,0000
57344	0,8750	100	50	0,5	0	0,0000	0,0000
61440	0,9375	100	50	0,5	0	0,0000	0,0000
						Área total:	**0,01127**

En la última columna de la Tabla [3-1] se presenta el valor de la contribución de cada medida al área encerrada por la función de salida, evaluada mediante el método numérico de integración de los trapecios (configurado con un parámetro h=0.0625). A continuación procederemos a representar gráficamente (Figura 3-6) las medidas experimentales presentadas en la Tabla [3-1], a fin de visualizar la forma de la distribución de salida del bloque f.d.p pseudo normal, así como evaluar el valor de la varianza y media experimental de la distribución.

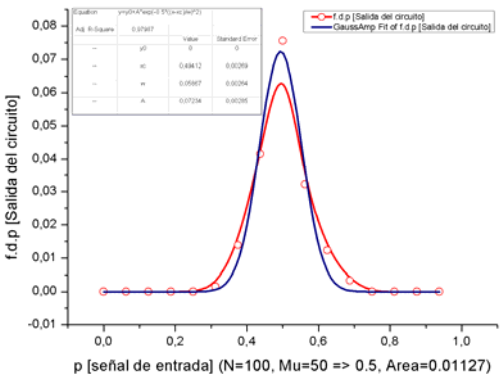

Figura 3-6: Distribución de salida del bloque f.d.p pseudo-normal para ‘N=20’ y una media ‘μ=0,5’

Como puede apreciarse en la Figura 3-6, la salida del bloque se asemeja con gran verosimilitud a la salida de una función normal. Los parámetros de la distribución normal a la cual se asemeja mejor son los siguientes: una media "$\mu = 0{,}49412 \approx 0{,}5$" prácticamente idéntica a la asignada al bloque y una desviación estándar "$\sigma = 0{,}05867$".

Una vez demostrada la correcta operación del bloque procederemos a determinar si la variación en la desviación estándar "σ"se rige realmente por la expresión predicha por la teoría (Ecuación [3-25]). Para ello hemos fijado el valor medio de la distribución "$\mu = 0{,}5$" para diferentes valores de "$N = \{13, 25, 50, 100, 200\}$" (que se hallan directamente relacionados con la desviación estándar de la distribución "$\sigma_{\mathrm{Re}\,f}$" obtenida a la salida) procediendo para cada uno de ellos a medir la salida variando la probabilidad de la señal estocástica de entrada en el intervalo [0, +1].

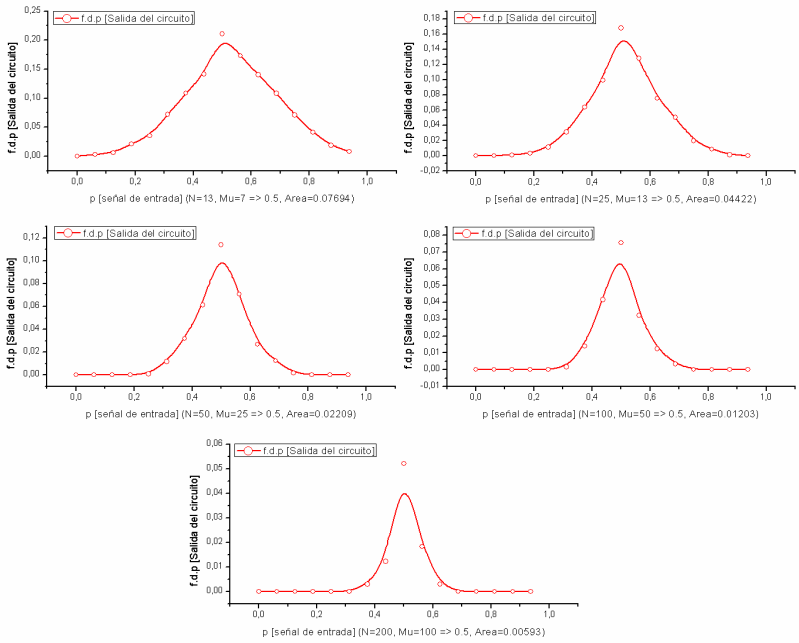

Figura 3-7: *f.d.p pseudo normal* variación de σ en función del parámetro N del circuito

A partir de las medidas experimentales realizadas (Figura 3-7) hemos procedido a ajustar una función Gaussiana para cada una de ellas, determinando así la desviación estándar de la

distribución experimental. Los valores ajustados obtenidos se presentan a continuación en la Tabla [3-2].

Tabla 3-2: Desviación estándar en función del parámetro N	
Valor N (Bloque P2B(N) del circuito)	Desviación estándar " σ "
13	0,1553
25	0,1082
50	0,0753
100	0,0587
200	0,0400

A partir de los datos de la Tabla [3-2], hemos procedido a realizar un ajuste potencial de la *desviación estándar* "σ" de la distribución en función del parámetro "N":

$$\sigma = \frac{0{,}55324}{\sqrt{N}}, \text{con una } R^2 = 0{,}9957 \qquad \text{Ecuación [3-26]}$$

Como bien puede apreciarse en la Ecuación [3-26] la relación experimental obtenida concuerda totalmente con la predicha por la teoría (Ecuación [3-25]), por lo tanto podemos afirmar que la implementación es completamente operativa.

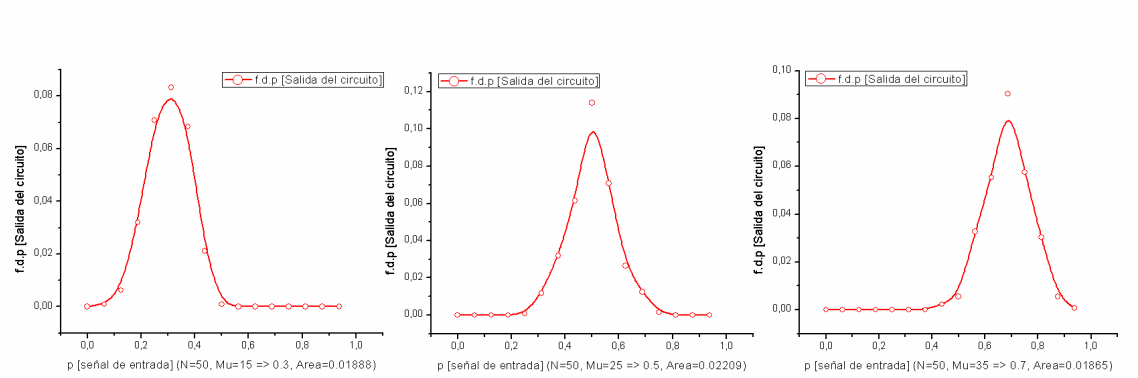

Figura 3-8: Área de la *f.d.p pseudo normal* en función de la posición de la media de la distribución

Finalmente, hemos procedido a evaluar si el área encerrada por la función distribución de probabilidad de salida de nuestro bloque varía en función del punto donde se halle situado el valor medio de la distribución. Para ello, hemos procedido a fijar el valor del parámetro "N=50" del bloque a la vez que se ha variado la probabilidad de la señal estocástica de

entrada en el intervalo [0, +1] para diferentes valores de media de la distribución "$\mu = \{0.3, 0.5, 0.7\}$". Los resultados obtenidos se presentan en la Tabla [3-3].

Tabla 3-3: Área de la distribución en función del valor de la media N

Media de la distribución "μ"	Área de la distribución de salida de la f.d.p
0,3	0,01888
0,5	0,02209
0,7	0,01865

Como bien puede apreciarse en la Tabla [3-3] y en la Figura 3-8, el área de la distribución varía en función del valor medio asignado a la distribución. El máximo del área se corresponde con el valor de la media "$\mu = 0.5$" que es el punto en el que la varianza de la distribución de salida del bloque P2B(N) se hace máxima. Por lo tanto, el valor máximo del área será para "$\mu = 0.5$" y decrecerá de forma cuadrática alrededor de dicho punto. A partir de las diferentes medidas experimentales realizadas podemos concluir que el bloque implementado permite la obtención de múltiples distribuciones de probabilidad pseudo normales (mediante el ajuste de dos parámetros del bloque "μ" y "$\sigma = f(N)$"). No obstante tiene el inconveniente que la distribución de probabilidad de salida de este bloque no encierra en ningún caso un área constante e igual a "+1" como se requiere para que una distribución de probabilidad sea considerada como normal. Para ello, en la codificación estocástica clásica (UCSL) definida entre [0, +1] deberíamos proceder a su normalización, no obstante puede darse el caso que el factor de normalización se mayor a +1. Para este caso definiremos el área de la distribución más pequeña presente en las diferentes distribuciones a clasificar como el área unitaria de nuestro sistema, procediendo a normalizar el resto de distribuciones en función de este valor. A su vez deberemos proceder a la normalización de la distribución para un determinado valor medio (ya que el área encerrada por esta distribución es dependiente de la posición del valor medio de la distribución). Todos estos inconvenientes son los que nos han forzado a desarrollar una f.d.p estocástica puramente normal. La cual se presenta en el siguiente subapartado.

3.2.2.2 Implementación estocástica de una f.d.p normal

Para poder evaluar el buen funcionamiento de una determinada metodología de reconocimiento de patrones, así como para poder estimar la función distribución de probabilidad de una categoría dada (evaluada mediante métodos no-paramétricos) se necesita disponer de funciones densidad de probabilidad normales. La forma de dichas distribuciones ha de ajustarse mediante dos parámetros como son: la media "μ" y la desviación estándar "σ". Por lo tanto, en vista de los diferentes inconvenientes operacionales descritos anteriormente para la *función distribución de probabilidad pseudo-normal* desarrollada hemos procedido a desarrollar una función distribución de probabilidad realmente normal (mediante la codificación UCLS), basada en el uso del bloque función Gaussiana descrita en el apartado 2.2.3.9 del capítulo segundo de la presente tesis. En este capítulo se describe el principio de operación detallado de la función exponencial (Parte izquierda Figura 3-9), así como la relación matemática entre los valores de los parámetros de ajuste y la función matemática a la cual representan.

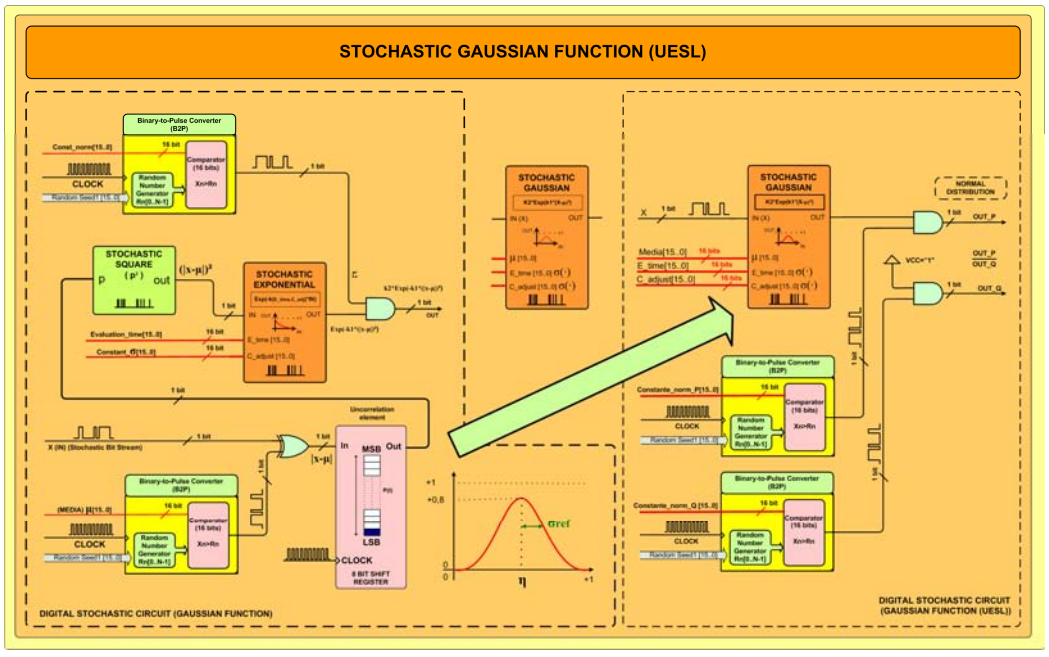

Figura 3-9: (Izquierda) Bloque función Guassiana (UCSL), (Derecha) Bloque estocástico función Gaussiana (UESL) normalizada

Hemos procedido a implementar en un dispositivo de lógica programable (FPGA) un bloque *función gaussiana*, que representa una distribución de probabilidad con una media "$\mu = 0.5$" y una desviación estándar "$\sigma = 0.1177$", que se corresponden a los siguientes parámetros de configuración del bloque (Ecuación [3-27]) en su implementación a 16-bits:

$$\begin{cases} \mu = 0.5 \rightarrow media = \mu \cdot 2^{16} = 0.5 \cdot 65535 = 32768 \\ \sigma = \dfrac{0.55405}{Eval\_time \cdot \left(\dfrac{C\_adjust}{2^{16}} \right)} \rightarrow \begin{cases} \sigma = 0.1177 \\ C\_adjust = 655 \end{cases} \rightarrow Eval\_time = 4000 \end{cases}$$

Ecuación [3-27]

Por lo tanto, configuraremos el bloque con los siguientes parámetros "media=32768", "C_adjust=655" y "Evaluation_time=4000", a la vez que procedemos a variar la entrada del bloque "X(IN)" en el intervalo [0, +1]. Los resultados obtenidos se presentan en la tercera y cuarta columna de la Tabla [3-4]:

Tabla 3-4: Medidas experimentales del bloque Gaussiana (UESL)

Experimental [Datos de entrada]		Gaussiana (UCSL) No Normalizada		Gaussiana (UESL) Normalizada			Teórico
IN [16bits]	Valor IN	OUT [16bits]	Valor OUT	OUT_P [16bits]	OUT_Q [16bits]	Valor OUT (P/Q)	OUT [Teórico]
0	0,0000	0	0,0000	0	18804	0,0000	0,0002
2048	0,0313	0	0,0000	1	18789	0,0001	0,0007
4096	0,0625	0	0,0000	0	18365	0,0000	0,0021
6144	0,0938	0	0,0000	2	18855	0,0001	0,0059
8192	0,1250	0	0,0000	0	18591	0,0000	0,0151
10240	0,1563	0	0,0000	4	18609	0,0002	0,0360
12288	0,1875	0	0,0000	0	18916	0,0000	0,0798
14336	0,2188	4001	0,0611	4021	18671	0,2154	0,1637
16384	0,2500	8036	0,1226	7957	18658	0,4265	0,3116
18432	0,2813	12627	0,1927	11787	18239	0,6463	0,5497
20480	0,3125	17934	0,2737	20316	18692	1,0869	0,8992
22528	0,3438	24006	0,3663	28216	18431	1,5309	1,3636
24576	0,3750	32008	0,4884	38233	18547	2,0614	1,9170
26624	0,4063	37528	0,5726	45684	18783	2,4322	2,4985
28672	0,4375	47633	0,7268	57571	19030	3,0253	3,0191
30720	0,4688	57533	0,8779	61530	18783	3,2758	3,3821
32768	0,5000	65535	1,0000	65535	18565	3,5300	3,5125
34816	0,5313	57533	0,8779	61613	18455	3,3386	3,3820
36864	0,5625	53022	0,8091	57707	18345	3,1457	3,0189
38912	0,5938	45530	0,6947	49509	19024	2,6024	2,4983
40960	0,6250	35546	0,5424	35371	18628	1,8988	1,9167
43008	0,6563	25525	0,3895	23722	18574	1,2772	1,3633
45056	0,6875	18457	0,2816	15668	18825	0,8323	0,8990
47104	0,7188	12003	0,1832	7780	18613	0,4180	0,5496
49152	0,7500	8002	0,1221	4115	18743	0,2195	0,3115
51200	0,7813	4001	0,0611	0	18635	0,0000	0,1637
53248	0,8125	0	0,0000	0	18743	0,0000	0,0797

55296	0,8438	0	0,0000	7	18612	0,0004	0,0360
57344	0,8750	0	0,0000	1	18904	0,0001	0,0151
59392	0,9063	0	0,0000	0	18685	0,0000	0,0059
61440	0,9375	0	0,0000	6	18630	0,0003	0,0021
63488	0,9688	0	0,0000	1	18510	0,0001	0,0007
		Área total:	**0,2701**		**Área total:**	**0,9989**	

Si procedemos a graficar (parte superior en la Figura 3-10) el valor de la función de salida en función de la probabilidad de entrada "X", podremos apreciar cómo la salida del bloque se ajusta perfectamente a una función Gaussiana. No obstante si ahora evaluamos el área encerrada debajo de dicha función (mediante la técnica de integración por trapecios) es de "Área=0.2701", por lo tanto su área no es "+1" como se requiere para que una función de distribución de probabilidad sea normal (Ecuación [3-18]).

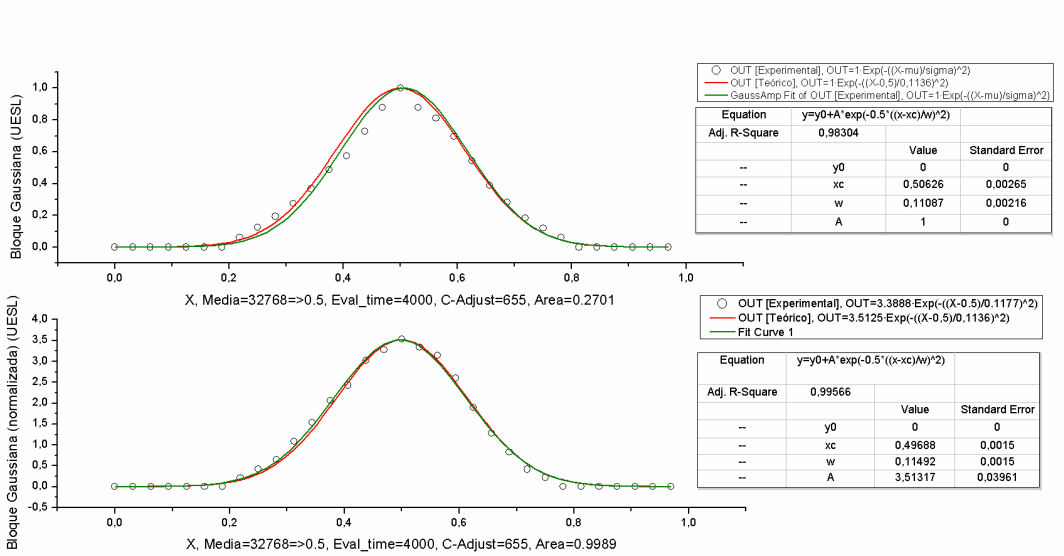

Figura 3-10: (Arriba) Bloque Gaussiana sin normalizar, (Abajo) Bloque Gaussiana normalizada

Para poder resolver este inconveniente se requiere que normalicemos dicha función mediante la multiplicación a la salida del bloque del siguiente factor de normalización (Ecuación [3-28]):

$$\{\sigma = 0.1177 \rightarrow \frac{1}{\sigma\sqrt{2\pi}} = 3.3888 \qquad \text{Ecuación [3-28]}$$

Siendo el factor normalización hallado "factor_norm=3.3888", el cual excede de mucho el rango de representación de codificación estocástica clásica. Por lo tanto, es inviable

mediante esta codificación poder normalizar dicha función desde un punto de vista rigurosamente matemático.

Debido a esta circunstancia nos hemos visto forzados a desarrollar un nueva codificación estocástica específica para la implementación de este tipo de sistemas a la cual hemos llamado codificación estocástica extendida sin signo (UESL) descrita en apartado “2.2.5” del capítulo segundo de la presente tesis. Mediante este método podremos escalar valores en el intervalo [0, +∞) mediante la evaluación de la razón entre dos señales estocásticas clásicas que representan una magnitud en [0, +1]. Por lo tanto, el factor de normalización anteriormente evaluado ahora lo podremos codificar mediante la razón de dos señales, tal y como se muestra en la siguiente expresión (Ecuación [3-29]):

$$\sigma = 0.1177 \rightarrow \frac{1}{\sigma\sqrt{2\pi}} = 3.3888 \rightarrow 3.3888 = \frac{1}{x} \rightarrow x = 0.29508 \rightarrow$$
$$\rightarrow \begin{cases} 1.00000 \cdot 2^{16} \rightarrow 65535 \\ 0.29508 \cdot 2^{16} \rightarrow 19339 \end{cases} \rightarrow factor\_norm = \frac{P}{Q} = \frac{65535}{19339}$$

Ecuación [3-29]

No obstante, existe otro inconveniente relacionado con el hecho que no disponemos actualmente de ningún bloque (UESL) nativo que implemente la función Gaussiana (es una de las tareas futuras a llevar a cabo). Para poder sortear este problema hemos procedido a usar un bloque *función Gaussiana* implementada mediante la codificación (UCSL) cuyo rango de representación de la salida se halla definida en [0, +1]. Entonces para obtener una función normal (UESL) (parte derecha de la Figura 3-9) usaremos la salida de este bloque conectada al numerador (mientras que en el denominador fijaremos una señal a +1) de un bloque multiplicador (UESL) (formado internamente por dos puertas **AND**), multiplicando así dicha salida por el factor normalización anteriormente evaluado (Ecuación [3-29]). La expresión matemática (Ecuación [3-30]) que describirá la salida de nuestro bloque será la siguiente:

$$P(x) = factor\_norm \cdot \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{1} = \frac{65535}{19339} \cdot \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{65535}, \quad x \in [0,+1] \qquad \text{Ecuación [3-30]}$$

Seguidamente hemos procedido a repetir las medias experimentales mediante el circuito digital presente en la parte izquierda de la Figura [3-9]. Los resultados obtenidos se presentan en las columnas cuatro, cinco y seis de la Tabla [3-4], que los representamos gráficamente (parte inferior de la Figura 3-10). Si ahora procedemos a evaluar el área encerrada por dicha función "Área=0.9989" mediante la técnica de los trapecios, podremos apreciar como ahora sí disponemos de una distribución normal real, apta para la implementación de sistemas de reconocimiento de patrones estadísticos.

3.2.3 LA CLASIFICACIÓN BAYESIANA ESTOCÁSTICA

Una vez evaluadas las diferentes implementaciones de las f.d.p, ya disponemos del elemento básico para la evaluación de la probabilidad a posteriori de una determinada categoría. Ya que la evaluación de esta probabilidad es el mecanismo que nos asegurará que la probabilidad de error en la estimación sea mínima. Entonces a partir de las f.d.p estocásticas (obtenidas mediante los bloques estocásticos anteriormente descritos) podremos implementar la probabilidad condicional de cada una de las diversas clases o categorías presentes en nuestra región de muestreo. De esta forma podremos determinar la función de probabilidad a posteriori $P(C_i|x)$. Para evaluar esta probabilidad se requerirá de un circuito estocástico que implemente la formula de Bayes $P(C_i|x)$ (Ecuación [3-31]) para cada una de las N clases o categorías existentes en el sistema.

$$P(C_i \mid x) = \frac{P(C_i) \cdot P(x \mid C_i)}{\sum_{j=0}^{N} P(C_j) P(x \mid C_j)} \qquad \text{Ecuación [3-31]}$$

Donde la expresión $P(C_i|x)$ representa la probabilidad a posteriori, que no es nada más que la probabilidad de encontrarse en la categoría "C_i" dada una medida "x". Siendo $P(C_i)$ la probabilidad a priori de la clase o categoría "C_i", es decir la probabilidad de ocurrencia de dicha categoría. Por otro lado $P(x|C_i)$ es la función de probabilidad condicional de "C_i" con respecto de "x", que se corresponde con la probabilidad que una medida o dato de entrada "x" pertenezca a una clase o categoría "C_i".

A lo largo de los siguientes subapartados se presenta la implementación estocástica de la regla de Bayes (para la evaluación de la probabilidad a posteriori) mediante la codificación estocástica (UCSL) (que se corresponden a los trabajos previos) y mediante la codificación (UESL) (que se corresponde a los últimos trabajos desarrollados, que son totalmente acordes con las propiedades y requerimientos de una descripción estadística purista).

3.2.3.1 Implementación estocástica (UCSL) de la regla de Bayes

El circuito estocástico a continuación presentado se corresponde con los primeros desarrollos que realizamos [3-16] en el campo del reconocimiento de patrones estadístico. Éste es capaz de determinar la probabilidad a posteriori de una determinada distribución de probabilidad mediante el uso de un sistema estocástico que implementa la regla de Bayes.

Para ofrecer más claridad al lector en la descripción de la implementación realizada, presentaremos un ejemplo simple como es la evaluación de la probabilidad a posteriori de una determinada clase "A", de un conjunto de patrones "Ω" formado por sólo dos clases (A y B). Para ello, según la regla de Bayes (Ecuación [3-31]) deberemos evaluar en el numerador el producto de la probabilidad condicional de la clase "A" (*P(x|A)* obtenida mediante uno de los bloques f.d.p anteriormente descritos) por la probabilidad a priori de la clase "A", de tal forma que en el numerador deberemos evaluar la siguiente expresión (Ecuación [3-32]):

$$P(A)\cdot P(x \mid A) \qquad \text{Ecuación [3-32]}$$

Por otro lado en el denominador deberemos proceder a la suma de los productos de las probabilidades condicionales "$P(x|C_i)$" de cada categoría por su probabilidad a priori "$P(C_i)$", obteniéndose así un término "*P(x)*" que tiene la función de normalizar la función de distribución de probabilidad a posteriori. En nuestro caso la expresión a evaluar es la siguiente (Ecuación [3-33]):

$$P(x) = P(A)\cdot P(x \mid A) + P(B)\cdot P(x \mid B) \qquad \text{Ecuación [3-33]}$$

Una vez descrita la función matemática a implementar pasaremos a describir la implantación digital del sistema estocástico, la cual se presenta en la Figura 3-11.

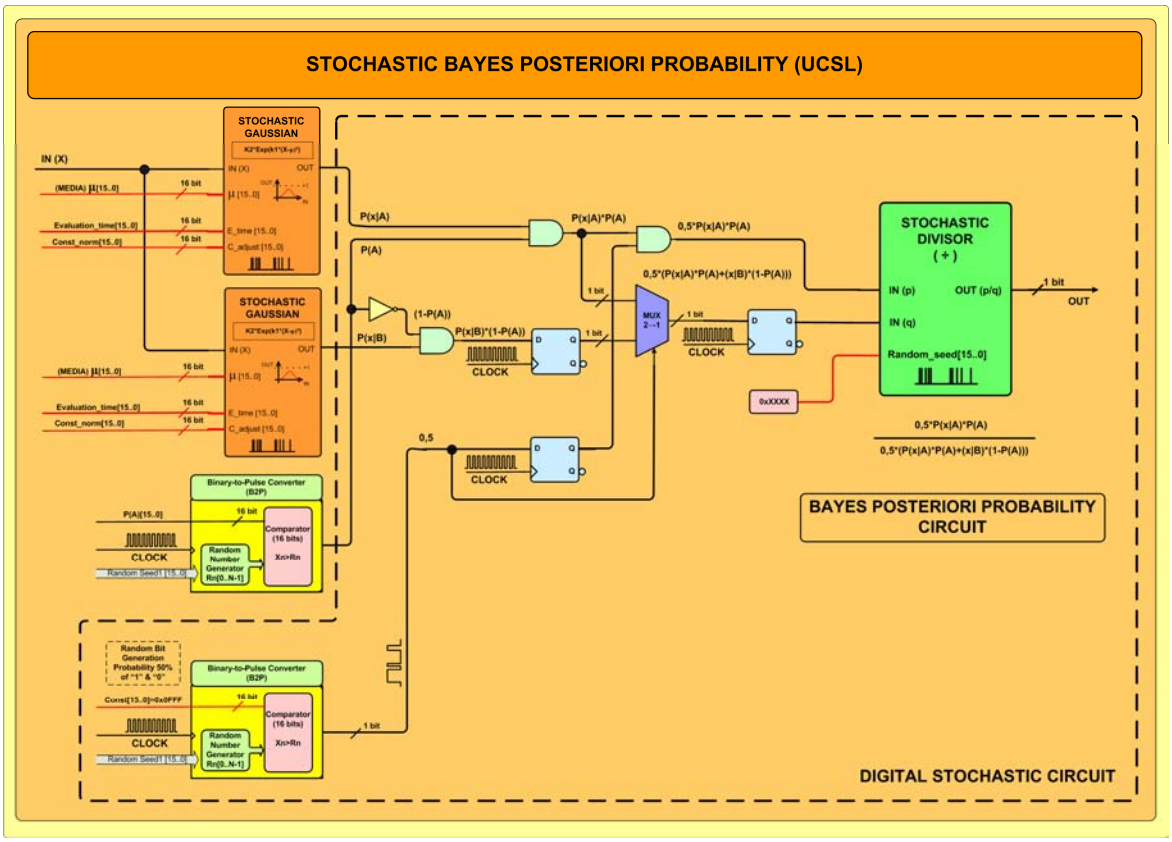

Figura 3-11: Circuito genérico para la evaluación de la probabilidad a posteriori (UCSL)

Empezaremos describiendo la implementación estocástica de la función del denominador (Ecuación [3-33]). Ésta tiene influencia en algunas correcciones que deberemos realizar en la implementación de la función del numerador con tal de asegurar la corrección matemática de la implementación realizada. Por lo tanto, inicialmente procederemos a realizar el producto de la probabilidad condicional por la probabilidad a priori para cada una de las clases mediante una pareja de puertas **AND**. Al tratarse de un sistema con dos clases, la probabilidad a priori de la segunda categoría (B) una vez conocida la probabilidad de la primera (A) se evaluará mediante una puerta **NOT** que implementa la función (*1-P(A)*). Entonces dicha pareja de productos se suman mediante un bloque sumador en valor medio, que como bien hemos visto evalúa la suma de las diferentes señales ponderada por el número de términos (en este caso dos), siendo la salida (Ecuación [3-33]) de este bloque:

$$\frac{1}{2}\cdot\left(P(A)\cdot P(x\mid A)+P(B)\cdot P(x\mid B)\right) \qquad \text{Ecuación [3-33]}$$

Entonces para compensar este factor nos veremos forzados a multiplicar la señal del numerador (que se corresponde al producto de la probabilidad condicional de la clase A por la probabilidad a priori de la clase A) por un factor "0.5". De esta forma se obtiene una relación (Ecuación [3-34]) entre el numerador y el denominador totalmente equivalente a la descrita por la teoría (Ecuación [3-31]).

$$\frac{\frac{1}{2}\cdot\left(P(A)\cdot P(x\mid A)\right)}{\frac{1}{2}\cdot\left(P(A)\cdot P(x\mid A)+P(B)\cdot P(x\mid B)\right)}=\frac{P(A)\cdot P(x\mid A)}{P(A)\cdot P(x\mid A)+P(B)\cdot P(x\mid B)}$$

Ecuación [3-34]

Finalmente para evaluar la razón entre el numerador y el denominador usaremos un bloque divisor (UCSL) (descrito en el capítulo segundo apartado 2.2.3.7.1) para obtener así una señal estocástica de salida definida en el intervalo [0, +1]. En este caso no hay problema de infravaloración de la salida de la razón entre las señales, ya que esta está acotada teóricamente entre 0 y +1. Como bien puede apreciarse, en el circuito se hallan repartidos un conjunto de biestables tipo D, la función de los cuales es descorrelacionar las señales estocásticas entre sí para evitar la correlación temporal entre señales evaluadas a partir de los mismos datos de origen.

Para evaluar la bondad de la implementación realizada, en el siguiente subapartado presentaremos un conjunto de medidas experimentales mediante las cuales se demuestra la correcta operación de la implementación realizada.

3.2.3.1.1 Evaluación de la implementación estocástica (UCLS) en el supuesto de un sistema de dos clases (1D)

Hemos procedido a la integración de un sistema formado por dos clases para la evaluación [3-16] de la operatividad del bloque estocástico en la codificación (UCSL), que determinará la distribución de probabilidad a posteriori de una determinada señal. Para ello se ha usado una pareja de bloques *f.d.p pseudo normal* (implementados mediante bloques de 16-bits)

para implementar las probabilidades condicionales de cada una de las categorías. El área encerrada por estos bloques se ha normalizado al valor del área menor de ambas (ya que se ha multiplicado la distribución de probabilidad condicional mayor por un factor inferior a uno, por lo tanto representable en la codificación UCSL). De esta forma se ha conseguido igualar las áreas de las distribuciones de probabilidad condicionales. Este hecho no debería modificar la salida del bloque ya que mediante el término "P(x)" se procederá de forma inherente a la normalización de la distribución de salida.

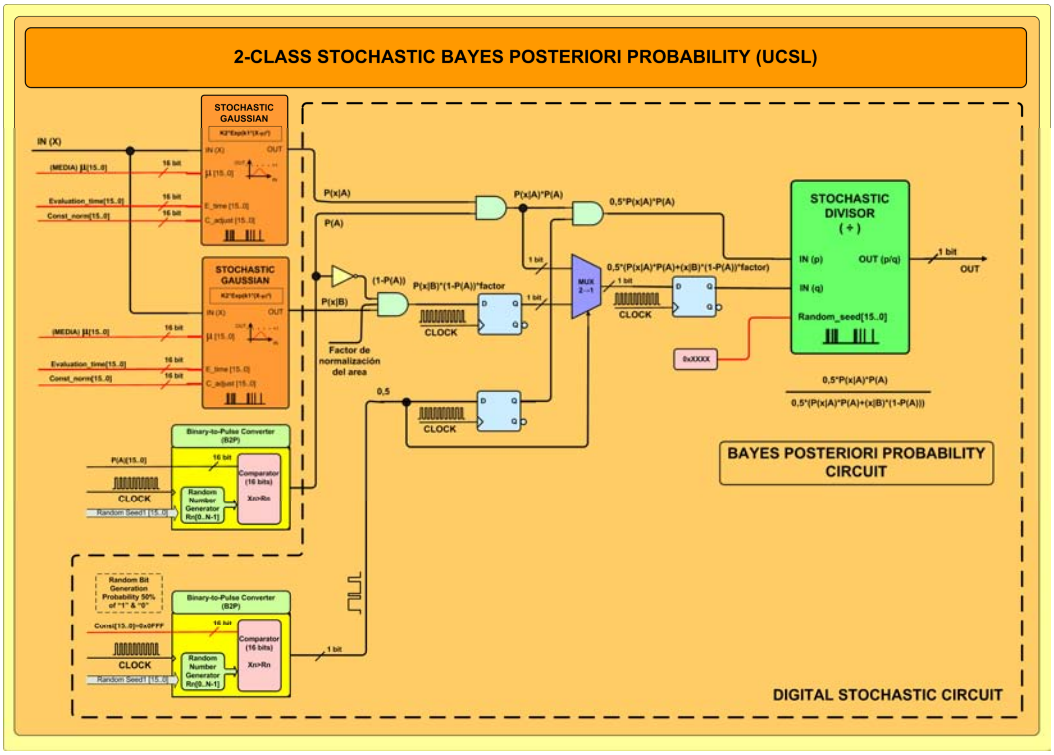

Figura 3-12: Bloque de evaluación de la probabilidad a posteriori para dos clases (UCSL)

Esta forma de proceder es una solución *ad-hoc* al problema, siendo difícil de sistematizar para la implementación de sistemas de clasificación compuestos por muchas categorías. Cabe remarcar que el funcionamiento interno del circuito (Figura 3-12) evaluador de la distribución a posteriori no lo describiremos nuevamente puesto que lo hemos descrito en el apartado anterior.

Para la implementación de estas dos clases hemos usado una pareja de bloques *f.d.p pseudo normal* (en su implementación de 8bits) para modelar la función distribución de probabilidad condicional de cada una de ellas "*P(x|A)*" y "*P(x|B)*". El área ha sido normalizada al valor del área menor de ambas, lo que ha implicado multiplicar la salida de la *f.d.p pseudo normal* de la clase B (el que tiene mayor área=0,0206) por un factor de normalización "0,8989" a fin que su área sea igual al de la clase A (Área=0,0185). Los parámetros de configuración para cada una de ellas los presentados en la Tabla [3-5], mientras que la probabilidad a priori de ambas distribuciones la hemos fijado a "*P(A)=P(B)=0,5*".

Tabla 3-5: Parámetros de configuración de las categorías A y B

Clase A					
Período de integración N [8bits]	σ equivalente	μ [16bits]	Valor μ	P(A) [8bits]	Valor P(A)
50	0,08	8	15/50=0,30	127	127/255=0,5
Clase B					
Período de integración N [8bits]	σ equivalente	μ [8bits]	Valor μ	P(B) [8bits]	Valor P(B)
50	0,08	25	25/50=0,50	127	127/255=0,5

Una vez configuradas las probabilidades condicionales y las probabilidades a priori de cada categoría con los parámetros de la Tabla [3-5], hemos procedido a variar la señal estocástica de entrada "X" en el intervalo [0, +1]. Los resultados obtenidos se presentan en la Tabla [3-6]:

Tabla 3-6: Evaluación experimental del bloque potabilidad a posteriori en un sistema de dos clases (UCSL)

Datos de entrada		Valores experimentales				Valores teóricos	
X [16bits]	Valor X	Valor probabilidad condicional clase A P(x|A)	Valor probabilidad condicional clase B P(x|B)	OUT Probabilidad a posteriori P(A|x) [16bits]	Valor Probabilidad a posteriori P(A|x)	Valor P(A|x) Teórico	Error=ABS(Teo-Exp)
0	0,0000	0,0000	0,0000	3	0,0000	Ind	Ind
2048	0,0313	0,0000	0,0000	65517	0,9997	Ind	Ind
4096	0,0625	0,0008	0,0000	65426	0,9983	1,0000	0,0017
6144	0,0938	0,0031	0,0000	65484	0,9992	1,0000	0,0008
8192	0,1250	0,0070	0,0000	65456	0,9988	1,0000	0,0012
10240	0,1563	0,0202	0,0000	65378	0,9976	1,0000	0,0024
12288	0,1875	0,0335	0,0000	65155	0,9942	1,0000	0,0058
14336	0,2188	0,0607	0,0000	64706	0,9874	1,0000	0,0126
16384	0,2500	0,0848	0,0015	64360	0,9821	0,9827	0,0006
18432	0,2813	0,0942	0,0055	61368	0,9364	0,9452	0,0088
20480	0,3125	0,0820	0,0212	57275	0,8740	0,7944	0,0796
22528	0,3438	0,0615	0,0227	51209	0,7814	0,7306	0,0508
24576	0,3750	0,0521	0,0292	45051	0,6874	0,6408	0,0466

26624	0,4063	0,0498	0,0278	32878	0,5017	0,6414	0,1397
28672	0,4375	0,0389	0,0612	20045	0,3059	0,3886	0,0827
30720	0,4688	0,0117	0,0673	9593	0,1464	0,1478	0,0014
32768	0,5000	0,0023	0,1006	1246	0,0190	0,0227	0,0037
34816	0,5313	0,0023	0,0812	1078	0,0164	0,0279	0,0115
36864	0,5625	0,0008	0,0638	1140	0,0174	0,0120	0,053
38912	0,5938	0,0008	0,0415	786	0,0120	0,0184	0,0064
40960	0,6250	0,0000	0,0232	492	0,0075	0,0000	0,0075
43008	0,6563	0,0000	0,0181	492	0,0075	0,0000	0,0075
45056	0,6875	0,0000	0,0134	263	0,0040	0,0000	0,0040
47104	0,7188	0,0000	0,0037	304	0,0046	0,0000	0,0046
49152	0,7500	0,0000	0,0021	46	0,0007	0,0000	0,0007
51200	0,7813	0,0000	0,0007	45	0,0007	0,0000	0,0007
53248	0,8125	0,0000	0,0006	16	0,0002	0,0000	0,0002
55296	0,8438	0,0000	0,0000	16	0,0002	Ind	Ind
57344	0,8750	0,0000	0,0000	12	0,0002	Ind	Ind
59392	0,9063	0,0000	0,0000	37	0,0006	Ind	Ind
61440	0,9375	0,0000	0,0000	21	0,0003	Ind	Ind
63488	0,9688	0,0000	0,0000	32	0,0005	Ind	Ind
Área evaluada:		0,01895	0,01829			**Error Total:**	Ind

Como podemos apreciar en la Tabla [3-6] los valores del área evaluada para las clases A y B son prácticamente idénticas. El área la hemos evaluado mediante la regla de integración numérica de los trapecios [3-24]. Los valores obtenidos experimentalmente son prácticamente idénticos a los predichos por la teoría, los cuales han sido evaluados a partir de los valores experimentales obtenidos a partir de las distribuciones de probabilidad condicionales de cada categoría. Si procedemos a representar gráficamente (Figura 3-13) la distribución de probabilidad condicional de la clase A (línea negra con círculos) y de la clase B (línea roja con cuadrados), a la vez que la distribución de probabilidad a posteriori de la clase A $P(A|x)$ (línea azul), se deduce claramente que la implementación (UCSL) evalúa correctamente las distribuciones de probabilidad a posteriori. La transición entre las dos categorías (punto de intersección en 0.5) se corresponde experimentalmente con el punto donde las probabilidades condicionales de cada categoría se hacen idénticas $P(x|A)=P(x|B)$.

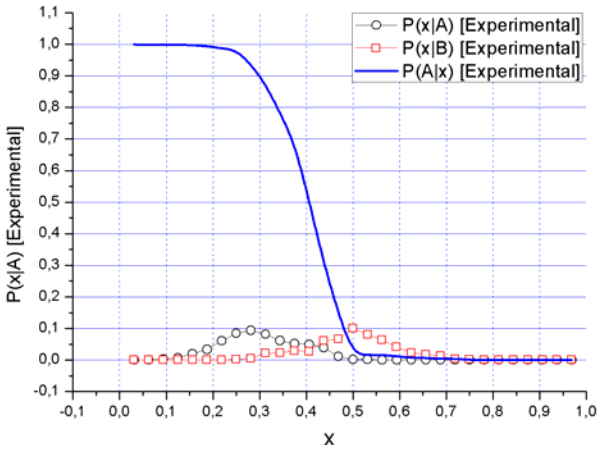

Figura 3-13: Medidas experimentales de la evaluación de la probabilidad a posteriori de un sistema de dos clases (UCSL)

3.2.3.2 Implementación estocástica (UESL) de la determinación de la probabilidad a posteriori

También hemos desarrollado un bloque (UESL) capaz de evaluar la probabilidad a posteriori de una función de distribución de probabilidad condicional de una determinada clase $P(x|C_i)$ implementada mediante la codificación (UESL). Dicha implementación ha sido necesaria a la vista de los inconvenientes prácticos hallados en las implementaciones estocásticas basadas en la codificación clásica (UCSL).

Para ello procederemos igual que en el caso (UCSL) al implementar la regla de Bayes (Ecuación [3-31]) para un sistema de dos categorías (A y B), presentando así la metodología básica mediante la cual podríamos expandir el sistema para el número "n" de clases contenidas en un conjunto dado de patrones "Ω".

Igual que antes, en el numerador (Ecuación [3-31]) hallaremos el producto de la probabilidad condicional $P(x|C_i)$ por la probabilidad a priori $P(C_i)$ de cada categoría. En el denominador hallaremos el factor de normalización $P(x)$ formado por la suma de todas las contribuciones de las diferentes categorías "$\sum_{i=1}^{n} P(x \mid C_i) \cdot P(C_i)$". No obstante en este caso tanto el numerador como el denominador se hallarán codificados mediante la razón de dos

señales estocásticas (codificación UESL), con un rango de representación de valores en el intervalo [0, +∞).

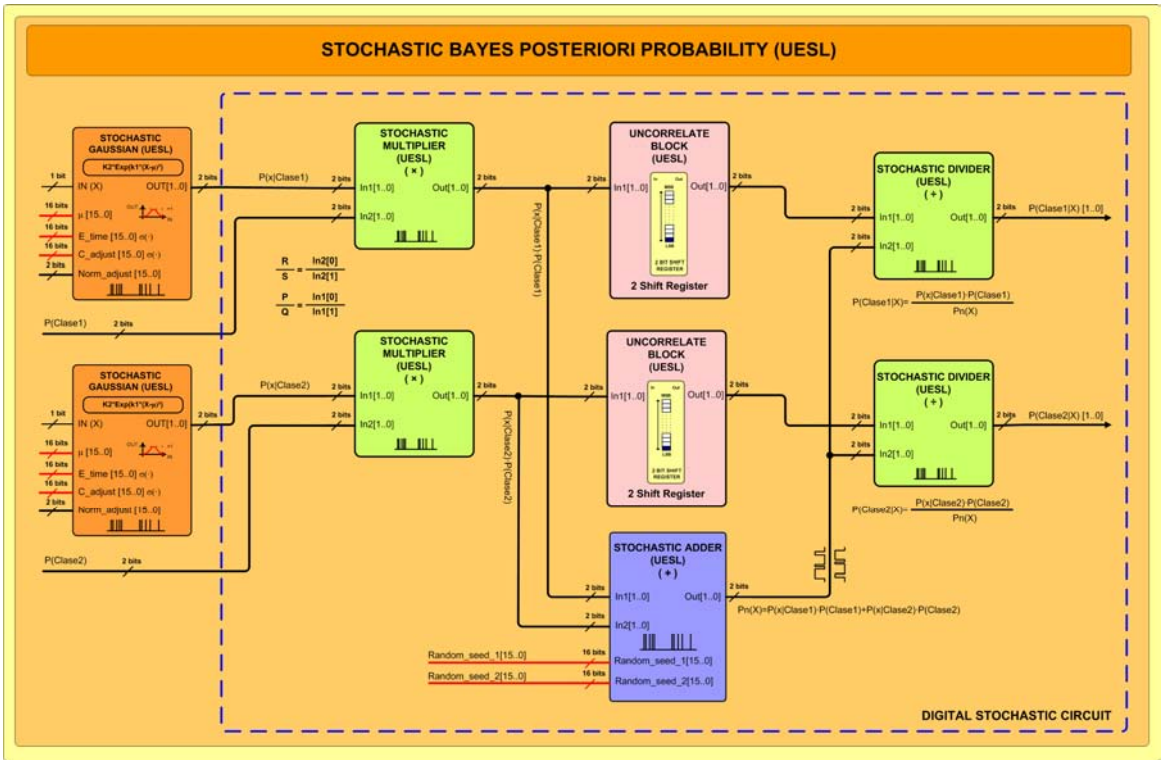

Figura 3-14: Bloque de evaluación de la probabilidad a posteriori (UESL)

La implementación UESL de la regla de Bayes se presenta en la Figura 3-14. Ésta consta de una pareja de bloques multiplicadores (UESL) (constituidos internamente por dos puertas **AND** (apartado 2.2.5.3)) que avalúan los productos $P(x|C_i)\cdot P(C_i)$ para cada categoría presente en el sistema (en nuestro caso son dos (A y B)). Posteriormente la salida de cada uno de estos multiplicadores se halla conectada a las entradas de un bloque sumador de dos entradas (UESL) (descrito en el apartado 2.2.5.1), el cual evaluará la suma natural de las diferentes contribuciones y no la suma ponderada como sucedía en la implementación UCSL. Finalmente, para la evaluación del factor presente en el numerador de la Ecuación [3-31] descorrelacionaremos la contribución $P(x|C_i)\cdot P(C_i)$ del factor $P(x)$ para cada categoría mediante un registro de desplazamiento de dos bits, para evitar así correlaciones no deseadas en la evaluación de la división (UESL) (formado internamente por una pareja de puertas **AND** (apartado 2.2.5.5)) evualando así las señales de salida para cada categoría.

Como bien puede apreciarse la implementación digital de la probabilidad a priori en esta codificación es mucho más simple que en el caso de la codificación UCSL que requería de un bloque divisor mucho más complejo.

A fin de probar la correcta operación de la implementación estocástica presentada en la Figura 3-14, procederemos en el siguiente subapartado a realizar un conjunto de medidas experimentales.

3.2.3.2.1 Evaluación de la implementación estocástica (UESL) en el supuesto de un sistema de dos clases (1D)

A fin de evaluar la correcta operación del bloque probabilidad a posteriori se ha planteado un problema formado por dos clases (A y B), del cual conocemos la probabilidad condicional de cada clase así como su probabilidad a priori.

La función probabilidad condicional de la clase $P(x|C_i)$ se ha implementado mediante sendos bloques *f.d.p normal* (UESL) (cuya área ha sido normalizada a +1), mientras que la probabilidad a priori $P(C_i)$ de ambas categorías se ha fijado en 0.5, es decir las hemos definido como equiprobables. Los parámetros de configuración para cada uno de los bloques *f.d.p normal* de cada categoría los presentamos en la Tabla [3-7]:

Tabla 3-7: Parámetros de configuración de las diferentes categorías

Clase 1							
μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
16384	0,25	6000	655	0,0092	4,1504	65535	15790
P(Clase1)	P(Clase1) [16bits]						
0,5	32768						
Clase 2							
μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
45056	0,6875	6000	655	0,0092	4,1504	65535	15790
P(Clase2)	P(Clase2) [16bits]						
0,5	32768						

Una vez configuradas las probabilidades condicionales y las probabilidades a priori de cada categoría con los parámetros de la Tabla [3-6], hemos procedido a variar la señal estocástica de entrada "X" en el intervalo [0, +1]. Siendo los resultados experimentales obtenidos los que se presentan en la Tabla [3-8]:

Tabla 3-8: Medidas experimentales del bloque probabilidad a posteriori (Regla de Bayes) para dos clases (UESL)

Datos de entrada		Clase 1 [Experimental]			Clase 2 [Experimental]		
IN [16bits]	Valor IN	OUT P(Clase1|X)_P	OUT P(Clase1|X)_Q	OUT (P/Q) P(Clase1|X)	OUT P(Clase2|X)_P	OUT P(Clase2|X)_Q	OUT (P/Q) P(Clase2|X)
0	0,0000	73	77	0,9481	0	79	0,0000
2048	0,0313	82	85	0,9647	0	95	0,0000
4096	0,0625	170	174	0,9770	0	187	0,0000
6144	0,0938	251	248	1,0121	0	248	0,0000
8192	0,1250	356	353	1,0085	0	361	0,0000
10240	0,1563	697	695	1,0029	0	726	0,0000
12288	0,1875	747	756	0,9881	0	831	0,0000
14336	0,2188	941	955	0,9853	0	969	0,0000
16384	0,2500	946	953	0,9927	0	971	0,0000
18432	0,2813	903	926	0,9752	0	962	0,0000
20480	0,3125	600	599	1,0017	0	625	0,0000
22528	0,3438	264	264	1,0000	0	286	0,0000
24576	0,3750	497	509	0,9764	0	557	0,0000
26624	0,4063	253	253	1,0000	0	280	0,0000
28672	0,4375	81	136	0,5956	44	133	0,3308
30720	0,4688	55	148	0,3716	83	170	0,4882
32768	0,5000	72	353	0,2040	278	363	0,7658
34816	0,5313	0	215	0,0000	189	201	0,9403
36864	0,5625	0	167	0,0000	163	161	1,0124
38912	0,5938	0	861	0,0000	842	842	1,0000
40960	0,6250	0	614	0,0000	599	606	0,9884
43008	0,6563	0	1007	0,0000	960	964	0,9959
45056	0,6875	0	1023	0,0000	949	947	1,0021
47104	0,7188	0	1012	0,0000	968	976	0,9918
49152	0,7500	0	595	0,0000	588	589	0,9983
51200	0,7813	0	374	0,0000	340	342	0,9942
53248	0,8125	0	598	0,0000	539	546	0,9872
55296	0,8438	0	188	0,0000	183	183	1,0000
57344	0,8750	0	82	0,0000	82	84	0,9762
59392	0,9063	0	177	0,0000	177	180	0,9833
61440	0,9375	0	97	0,0000	61	64	0,9531
63488	0,9688	0	82	0,0000	76	80	0,9500

Si representamos gráficamente la distribución de probabilidad a posteriori para la clase A (línea azul con el símbolo x) y la clase B (línea naranja con símbolo estrella), a la vez que las distribuciones de probabilidad condicionales de cada clase (clase A = línea rosa con círculos, y clase B = línea verde con cuadrados) obtenemos la siguiente Figura 3-15.

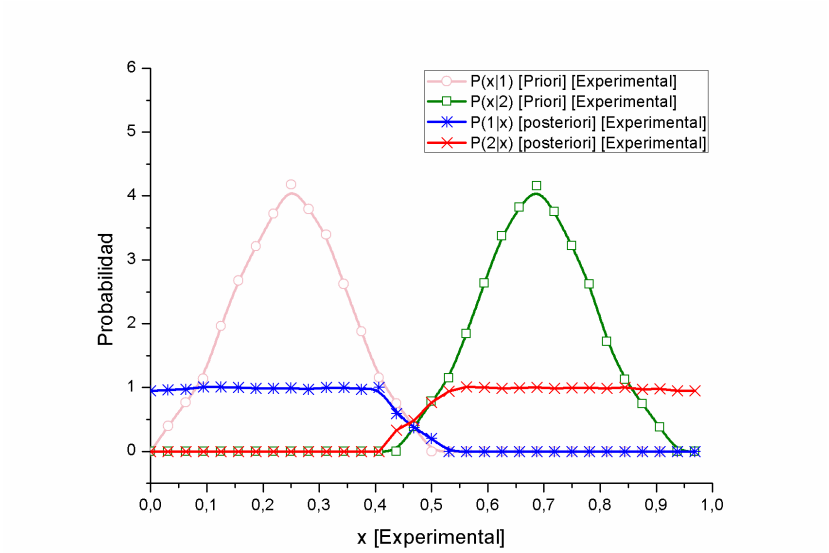

Figura 3-15: Medidas experimentales del bloque probabilidad a posteriori (Bayes) para un sistema de dos clases (UESL)

Como bien puede apreciarse en la Figura 3-15, el bloque probabilidad a posteriori (UESL) evalúa de forma completamente correcta la probabilidad a posteriori para ambas distribuciones, puesto que el valor de las probabilidades a posteriori de ambas clases se hacen iguales en el punto en el que los valores de las probabilidades condicionales se igualan. En vistas que las probabilidades a priori son iguales.

3.2.4 CLASIFICACIÓN MEDIANTE LA TÉCNICA DE COMPARACIÓN MULTIDIMENSIONAL ESTOCÁSTICA

A lo largo de los anteriores apartados hemos presentado los mecanismos básicos para la implementación y evaluación de las diferentes distribuciones de probabilidad descritas en el campo de la estadística (como pueden ser las: a priori, condicionales y a posteriori) de forma estocástica. A continuación mostraremos cómo determinar de forma sistemática y sobre todo eficiente (desde el punto de vista de tiempo de computación), cuál es la clase más probable dados una serie de datos de entrada, en el supuesto que el conjunto de patrones "Ω" esté formado por un número lo suficientemente grande de clases. Por lo tanto, en respuesta a esta necesidad hemos desarrollado una técnica estocástica de clasificación a

la cual hemos denominado *comparador multidimensional estocástico* [3-15, 3-16]. Donde su implementación digital para las diferentes codificaciones estocásticas se aborda de forma detallada a lo largo de los siguientes subapartados.

La técnica desarrollada permite realizar de forma muy rápida una comparación entre un conjunto de datos de entrada (distribuciones de probabilidad a posteriori o condicionales, distancias euclidianas o Manhattan,…) del sistema con respecto a unos valores de referencia del espacio de configuraciones. Las salidas de los diferentes elementos comparadores se hallan entrelazadas, a fin de determinar cuál es la categoría más probable del sistema (conjunto de comparaciones realizadas). Consequentemente el bloque comparador estocástico desarrollado es el elemento central de esta metodología, al ser el elemento de decisión del sistema. El principio de operación se basa en aplicar la metodología de computación bio-inspirada conocida como "*Winner-Take-All*" usada primordialmente en el campo de las redes neuronales recurrentes como mecanismo de configuración de la red, implementando el conocido como *aprendizaje competitivo*. Mediante este método las salidas de las neuronas de una red se inhiben entre sí, a la vez que también pueden activarse a través de conexiones reflexivas. No obstante transcurrido un cierto intervalo de tiempo, tan sólo un nodo de la capa de salida de la red neuronal se mantendrá activo (la señal de entrada más probable). Por lo tanto, la red neuronal utilizará dicha inhibición no lineal para escoger la salida correspondiente a la señal de entrada más probable.

La primitiva de computación "*Winner-Take-All*" se ha implementado para diferentes tipos de redes neuronales, como son las redes neuronales clásicas (segunda generación) [3-18] o más recientemente para las redes neuronales pulsantes (tercera generación) [3-19]. Cabe remarcar que las redes neuronales basadas en el principio computacional "*Winner-Take-All*" son usadas comúnmente en los modelos computacionales que intentan emular el comportamiento del cerebro humano, en particular las usadas para emular el mecanismo de toma de decisiones distribuida que se da en la corteza cerebral. Por lo tanto podemos afirmar que se trata de un principio computacional bio-inspirado. Como ejemplos más significativos de su utilidad encontramos en la literatura: modelos jerárquicos de visión [3-

20] y modelos de atención selectiva y reconocimiento [3-18, 3-21]. Cabe destacar que formalmente se ha demostrado [3-22] que el principio de decisión "*Winner-Take-All*" es mucho más poderoso computacionalmente que otras operaciones no-lineales de decisión, como puede ser el "*Thresholding*".
Adicionalmente es importante resaltar que el principio computacional "*Winner-Take All*" es ampliamente usado en el diseño de sistemas CMOS analógicos para la implementación de mecanismos de decisión VLSI [3-23].

A lo largo de los siguientes subapartados se presentan las diferentes implementaciones de la metodología de comparación multidimensional (MSC) para las diferentes codificaciones estocásticas, así como sus respectivas evaluaciones experimentales.

3.2.4.1 Implementación estocástica (UCSL) de la metodología de comparación multidimensional

En el presente sub-apartado se aborda la implementación digital de un bloque estocástico en la codificación estocástica clásica (unipolar) (UCSL) capaz de comparar datos provenientes de diferentes fuentes mediante el uso del principio computacional "*Winner-Take-All*". Este método parece ser la opción de clasificación más prometedora para trabajar con señales pulsantes.

Es de reseñar que el uso primordial que se dará a este bloque digital es el reconocimiento/clasificación de patrones en el supuesto de la aproximación estadística. Por lo tanto, la metodología que se propone permite una fácil generalización al reconocimiento de M-categorías, donde cada una de estas categorías pueden ser a su vez n-dimensionales, siempre y cuando cada una de las contribuciones unidimensionales sean independientes entre si. En el peor de los casos procederemos a clasificar mediante la probabilidad condicional (que en el caso multidimensional obtendremos a partir de las probabilidades condicionales individuales "$P(x,y|C_i)=P(x|C_i)\cdot P(y|C_i)$"), y en el supuesto de conocer las probabilidades a priori para las diferentes clases "$P(C_i)$" presentes en nuestro sistema podremos clasificar sobre la probabilidad a posteriori, ya que es la que minimiza el error de

decisión. En resumen los mecanismos de decisión admisibles mediante esta técnica se presentan en la Ecuación [3-35]:

$$\begin{cases} Clasificación\_prob\_condicional \rightarrow P(x \mid C_i) > P(x \mid C_j), \forall i \neq j \\ Clasificación\_prob\_posteriori \rightarrow P(x \mid C_i) \cdot P(C_i) > P(x \mid C_j) \cdot P(C_i), \forall i \neq j \end{cases}$$

Ecuación [3-35]

Como bien hemos indicado al principio del capítulo nos centraremos básicamente en el reconocimiento de patrones estadístico. Por ello, y a fin de ser lo más claros con el lector a la hora de describir su funcionamiento nos restringiremos a distribuciones de probabilidad condicional conocidas (implementadas mediante los bloques *f.d.p* anteriormente descritos) para la evaluación de la capacidad computacional del bloque propuesto.

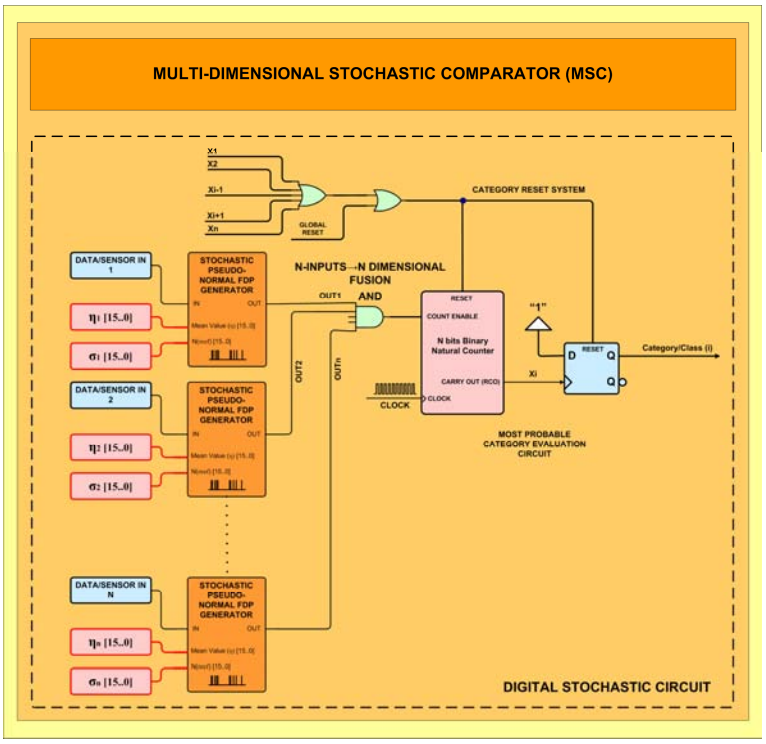

Figura 3-16: Diseño del bloque de comparación multidimensional estocástica (MSC). Donde la señal de salida del circuito 'category (i)' valdrá '1', en el caso que la señal/es de entrada se corresponden a la categoria i-ésima

En la Figura 3-16 se presenta el circuito digital que conforma el bloque estocástico encargado de implementar dicha metodología de clasificación n-dimensional. En dicha

figura podemos apreciar cómo una vez conocidas las funciones de distribución condicional (implementadas mediante bloques *f.d.p* estocásticos) pueden ser fusionadas mediante una puerta **AND** de tantas entradas como dimensiones tenga la categoría. Esta puerta se encarga de evaluar la densidad de probabilidad condicional de la categoría $P(x,y|C_i)$, obtenida como el producto estocástico de las diferentes contribuciones unidimensionales "$P(x,y|C_i)=P(x|C_i)\cdot P(y|C_i)$". Una vez obtenidas estas distribuciones de probabilidad globales para una determinada categoría C_i podremos proceder a evaluar cuál de las M categorías presentes en el sistema es la más probable para un período dado, siguiendo con la filosofía del principio computacional "*Winner-take-All*".

En el supuesto del reconocimiento de patrones estadístico los datos de entrada al sistema serán evaluados a través de una función distribución de probabilidad condicional característica de una determinada categoría. Esto implicará que la señal de entrada al elemento clasificador desarrollado será siempre una señal pulsante proporcional a la probabilidad de coincidencia de los datos de entrada en relación a la pertenencia a dicha clase o categoría. Con ello la salida de la puerta **AND** anteriormente descrita se conectará a la señal de activación del conteo "*Enable*" de un bloque contador de "N" bits. El número de bits de este contador deberá fijarse a un valor de compromiso que permita obtener una correcta evaluación de las diferentes categorías, a la vez que se minimiza el período de evaluación del sistema. Como regla general tomaremos como valor de "N" como la mitad de los bits usados en el bloque P2B(N) para la conversión de los datos binarios en señales estocásticas "$2^m \rightarrow N = m/2$".

En el instante en que uno de los elementos contadores (correspondientes a una determinada categoría "C_i") se desborde, se activará la señal de salida (RCO) (señal "x_i" fijada a nivel alto '1') del contador, la cual a su vez estará conectada a la entrada de reloj de un Flip-Flop tipo D. La salida de dicho bloque se pondrá a nivel alto '1' de forma estable (hasta que se produzca la próxima evaluación del sistema), indicando al sistema que dichos datos de entrada se corresponden a la categoría i-esima "*category(i) o Ci*". Por lo tanto la activación a nivel alto '1' de alguna de estas salidas, indicará al sistema la pertenencia de los datos a esa categoría, mientras que si se halla a nivel bajo '0' indicará la no pertenencia.

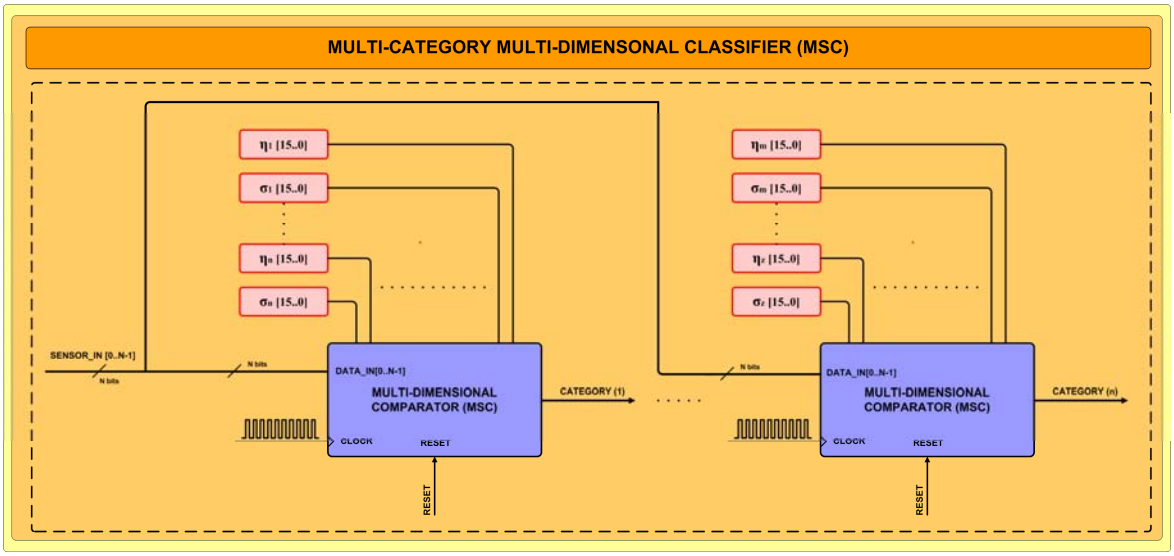

Figura 3-17: Arquitectura de un clasificador multi-categoría (M categorías diferentes)

Cuando alguno de los contadores incorporados en los diversos bloques (MSC) alcance el valor de desbordamiento (la señal RCO='1' → X_i='1') provocará que los contadores y los biestables D del resto de módulos (*MSC*) se reseteen automáticamente. A su vez, la señal combinada del resto de "M-1" clases se obtiene mediante una puerta OR de "M-1" entradas Por lo tanto, tan sólo la salida de uno de los bloques clasificadores (*MSC*) podrá permanecer activa a la vez, que se corresponderá con la categoría más probable de todas las evaluadas.

Para reconocer más de una categoría a la vez a partir de los diferentes datos de entrada se requiere de un clasificador multi-categoría (Figura 3-17). Este clasificador se compondrá de varios bloques de *comparación multidimensional estocástica (MSC)* conectados en paralelo, donde cada uno de ellos se hallará asociado a una distribución de probabilidad característica de una clase C_i (la cual podrá ser unidimensional o multidimensional).

Una vez descrito el funcionamiento de este bloque procederemos en los siguientes apartados a evaluar su capacidad computacional mediante la presentación de diferentes ejemplos prácticos.

3.2.4.1.1 Evaluación de la metodología en el supuesto de un sistema de dos clases (1D)

El objetivo principal de este subapartado es presentar una evaluación del rendimiento computacional [3-16] del bloque estocástico (*MSC*) anteriormente propuesto. Para ello nos hemos fijado como objetivo el resolver el problema de clasificación más simple que uno podría proponer en el campo del reconocimiento de patrones como es la clasificación basada en la probabilidad a posteriori de un sistema constituido por dos clases (A y B), ambas definidas en el mismo espacio unidimensional.

Para la implementación de estas dos clases hemos usado una pareja de bloques *f.d.p pseudo normales* (en su versión de 8bits) para modelar la función distribución de probabilidad condicional de cada una de ellas "*P(x|A) y P(x|B)*", siendo los parámetros de configuración para cada una los presentados en la Tabla [3-9], mientras que la probabilidad a priori para ambas distribuciones la hemos fijado a "*P(A)=P(B)=0,5*".

Tabla 3-9: Parámetros de configuración de las categorías A y B

Clase A					
Período de integración N [8bits]	σ equivalente	μ [8bits]	Valor μ	P(A) [8bits]	Valor P(A)
50	0,08	15	15/50=0,30	127	127/255=0,5
Clase B					
Período de integración N [16bits]	σ equivalente	μ [8bits]	Valor μ	P(B) [8bits]	Valor P(B)
50	0,08	25	25/50=0,50	127	127/255=0,5

No obstante, al ser idénticas las probabilidades a priori para ambas clases recuperaremos la regla de máxima verosimilitud, la cual nos indica que el error de clasificación será mínimo clasificando a partir de la probabilidad condicional (Ecuación [3-36]), que es lo que hemos realizado.

$$\begin{cases} P(A)\cdot P(x \mid A) > P(B)\cdot P(x \mid B) \\ P(A) = P(B) \end{cases} \rightarrow P(x \mid A) > P(x \mid B) \qquad \text{Ecuación [3-36]}$$

Entonces a partir de los elementos anteriormente descritos hemos implementado el circuito digital de clasificación multidimensional estocástica (*MSC*) que se muestra en la Figura 3-18.

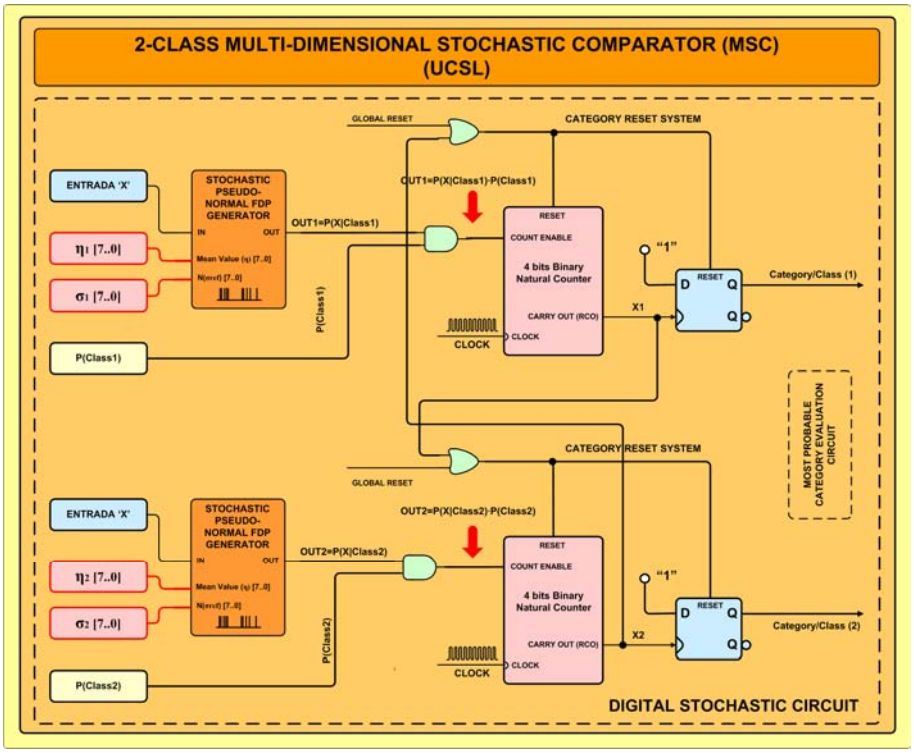

Figura 3-18: Diagrama del circuito MSC usado para la clasificación de dos clases (UCSL)

Los diferentes elementos del bloque *MSC* ya han sido descritos en detalle en el aparatado anterior, por lo tanto no volveremos a describir la función de cada uno de ellos. Debemos resaltar que en esta implementación hemos usado contadores de 4-bits para optimizar el período de evaluación del sistema. La probabilidad a posteriori que sea más probable para una señal de entrada "X" se hallará en promedio más tiempo a nivel alto, por ello será la que con más probabilidad hará rebosar el contador de dicha clase, fijando a nivel alto la salida de esta.

A continuación se presentan los resultados experimentales obtenidos de eficiencia computacional evaluada en un período de computación de un segundo para cada una de las tres metodologías abordadas (Tabla [3-10]).

Tabla 3-10: Comparación entre las diferentes metodologías de reconocimiento de patrones evaluadas

	Metodología estocástica	Metodología clásica	Mediante Software

	propuesta (FPGA)	electrónica digital (FPGA)	ejecutado sobre un Microcroprocesador
Soporte físico (IC)	(Altera Corp.) Stratix II EP2S60	(Altera Corp.) Stratix II EP2S60	Intel Q9400
Technologic de semiconductor (nm)	90	90	45
Frecuencia de operación del circuito (GHz)	0,45	0,45	2,66
Precio del elementote proceso (USD)	1.325,00	1.325,00	260,00
Número de estructuras de proceso, en el mismo dispositivo (FPGA Chip)	650	1	1
Tiempo de proceso por comparación (μs)	7,000	0,113	30,550
Rendimiento del sistema (millones de comparaciones por segundo)	92,86	8,85	0,03

En la primera columna de la Tabla [3-10] hallamos los resultados obtenidos mediante la metodología estocástica de comparación anteriormente presentada, que implementa la comparación estocástica entre las dos clases (A y B) mediante un bloque MSC unidimensional (con solamente una característica, es decir una única f.d.p). Gracias al bajo coste en términos de elementos lógicos (LE) de la implementación desarrollada, se ha conseguido integrar hasta un máximo de 650 bloques operativos en una sola FPGA. La FPGA usada para la implementación del sistema es una STRATIX II modelo EP2S60 del fabricante norteamericano ALTERA Corp.

En la segunda columna de la Tabla [3-10] se presentan los resultados obtenidos mediante un circuito puramente digital que evalúa la probabilidad a posteriori de ambas clases. En este caso se han usado bloques digitales de computación clásica implementados en una FPGA ALTERA de la familia STRATIX II, modelo EP2S60 (la misma que en el caso anterior). La implementación es mucho más compleja que en el caso anterior al requerir de diferentes bloques multiplicadores, divisores, sumadores y comparadores, todo ello unido a una memoria encastrada para implementar una Lookup-Table de N-bits donde se ha almacenado la función exponencial (a fin de implementar digitalmente las funciones densidad de probabilidad). Todo ello ha conducido a que tan sólo hallan recursos lógicos en la FPGA para implementar solamente uno de estos circuitos. A pesar que esta implementación es mucho más rápida (unas 70 veces) que la primera (la cual hace uso de la lógica estocástica) consume unas 600 veces más recursos lógicos. Este hecho resalta la dificultad ya conocida de implementación de estructuras de procesamiento en paralelo mediante la lógica digital clásica.

Finalmente en la última columna hallamos los resultados obtenidos mediante la implementación de un clasificador de dos clases a posteriori puramente software. Dicha aplicación se ha realizado en ANSI C (el código fuente del cual se adjunta en el Anexo B de la presente tesis) para su ejecución monoprocesador. El procesador del ordenador usado para la evaluación del sistema ha sido un quad-core Q9400 de Intel Corp operando a frecuencia de 2,66GHz. Dicha aplicación ha resultado ser computacionalmente unas 3000 veces más lenta que la implementación mediante la lógica estocástica, esto es debido a que la ejecución de la aplicación software es puramente secuencial, frente al paralelismo de la implementación estocástica.

3.2.4.1.2 Evaluación de la metodología para la generación de rutas en un entorno bidimensional virtual

En este subapartado se presenta un ejemplo adicional que realizamos para comprobar el correcto funcionamiento de la metodología en un sistema bidimensional, algo más complejo que el anteriormente descrito. En concreto el ejemplo que se presenta ha consistido en implementar un clasificador multidimensional aplicado a la solución de problemas de navegación en entornos bidimensionales (2D) conocidos, a partir del reconocimiento del entorno en el que uno se halla.

La estructura del sistema implementado se presenta en la (Figura. 3-19). En éste hallamos inicialmente un bloque de memoria (de un 1KBit de capacidad) que contiene un entorno virtual 2D, en el cual se pretende que nuestro sistema genere las rutas de desplazamiento necesarias para que un objeto pueda desplazarse entre las diferentes zonas de dicho entorno. En la Figura 3-20 se presenta el contenido de la memoria en la cual hemos almacenado el entorno virtual. Dicha memoria se ha estructurado en una matriz de 32x32 registros de memoria de 1bit, siendo su contenido ‘0’ si la posición está libre (color blanco) y ‘1’ si esta posición está ocupada/impenetrable (color negro).

Figura 3-19: Diagrama de bloques del sistema de generación de rutas

El funcionamiento del sistema implementado es el siguiente: para cada ciclo de reloj, el sistema se ubica en una nueva posición libre de su entorno (se desplaza de posición de memoria), donde su posición relativa respecto al entorno se determinará mediante el bloque denominado "*Position Block*". Este bloque se encarga de determinar la distancia a las zonas impenetrables/muros cercanos, obteniendo así un conjunto de cuatro parámetros o distancias del entorno (parámetros n, s, e y w de la Figura 4.10) mediante los cuales se procede al reconocimiento de la zona en la cual nos hallamos. Por lo tanto, dichas señales son las entradas de un clasificador 4-D en el cual todas las distancias son independientes entre si, tal y como se muestra en la Figura 3-29. Cada una de las cuatro señales de entrada se corresponde con una de las dimensiones del sistema de reconocimiento.

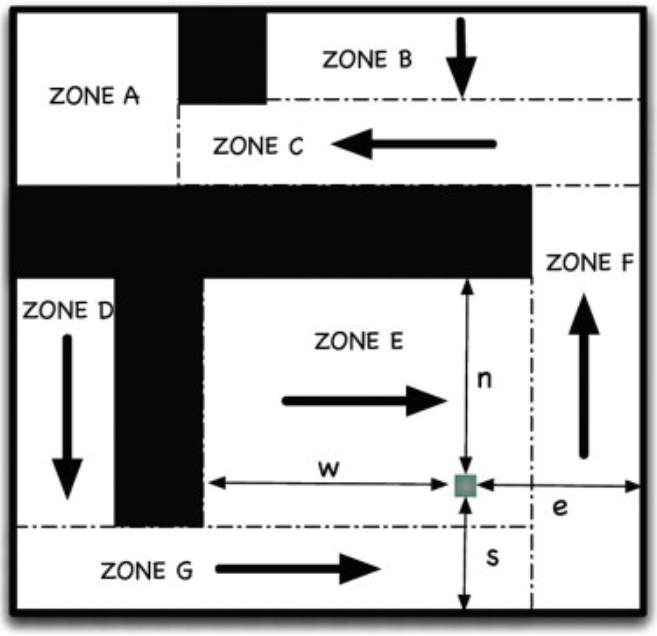

Figura 3-20: Entorno virtual creado para testar la metodología propuesta. En él se especifican las zonas en que se divide el espacio y la dirección de movimiento predefinida para cada zona.

Finalmente el bloque clasificador (cuya implementación interna se presenta en la Figura 3-21) toma los datos de la posición relativa (n, s, w, e) evaluados mediante el bloque "*Position Block*" como datos de entrada de las funciones distribución de probabilidad condicional "P(Ci|n,s,e,w)" obtenidas mediante MATLAB para cada una de las siete categorías o zonas (Zonas A, B, C, D, E, F, G presentes en la Figura 3-20) del entorno bidimensional. Cada una de estas regiones vendrá descrita mediante una distribución de probabilidad condicional $P(Ci \mid n,s,e,w) = P(Ci \mid n)\cdot P(Ci \mid s)\cdot P(Ci \mid e)\cdot P(Ci \mid w)$ cuatri-dimensional. Al ser independientes entre sí cada una de las contribuciones dimensionales, la combinación de todas se consigue mediante el producto conjunto de todas ellas. Cada una de las cuatro contribuciones unidimensionales que conforman una determinada categoría (tenemos un total de 7 categorías) se ha implementado mediante bloques *f.d.p pseudo normales*. Estos bloques los hemos configurado mediante una determinada media (μ) y una desviación estándar (σ) características. Los parámetros de estas distribuciones para cada una de las siete clases (un total de 28 parámetros) los hemos evaluado mediante una aplicación MATLAB específicamente desarrollada para ello. No obstante al ser la

probabilidad a priori de todas la categorías la misma “1/7” hemos procedido a clasificar directamente sobre la probabilidad condicional.

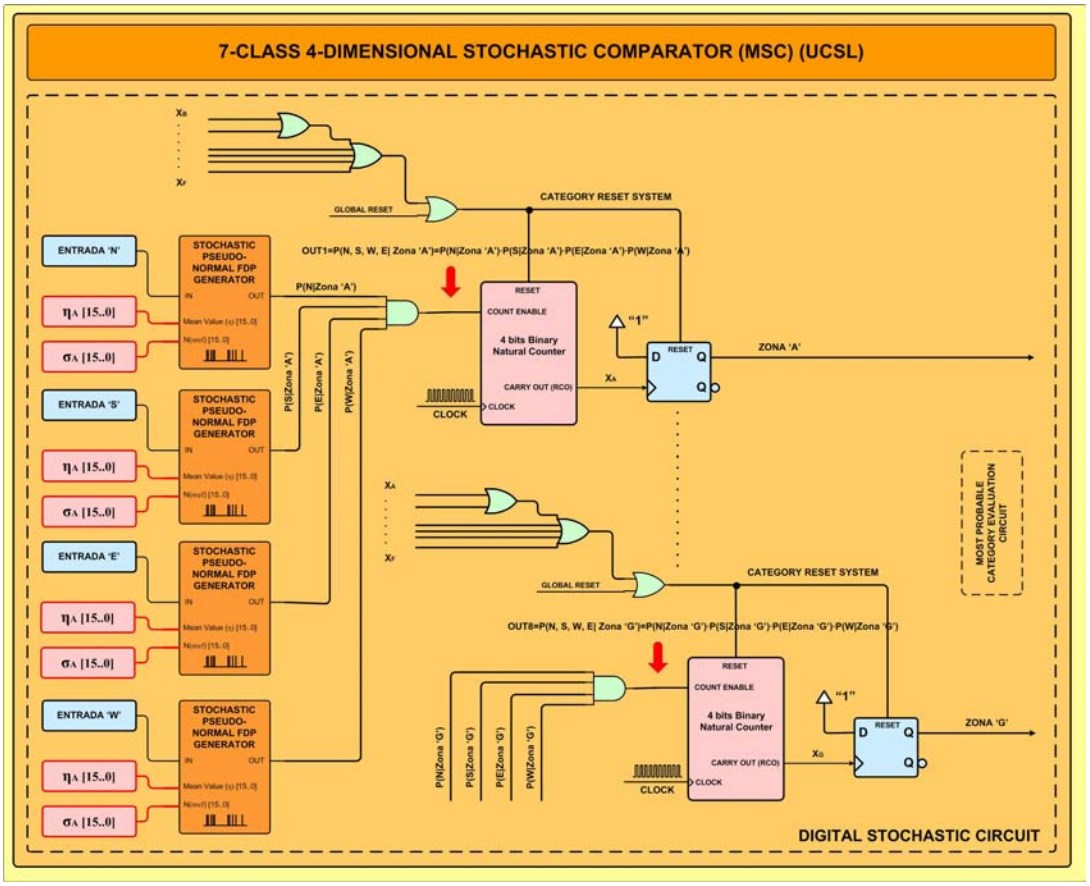

Figura 3-21: Bloque (MSC) para siete clases cuatridimensionales

Una vez identificada la zona en la que nos hallamos, el bloque “*Movement Decision Block*” asignará una dirección predefinida de desplazamiento. Esta dirección está definida previamente mediante una “*Look-up-Table*” para cada zona. Entonces dicha dirección de movimiento es enviada al bloque “Position Block”, el cual se encargará de ejecutar el desplazamiento desde la posición actual hasta la nueva posición, a la vez que determinará nuevamente las posiciones relativas a los diferentes obstáculos (n, s, e, w), repitiéndose así nuevamente el proceso anteriormente descrito hasta llegar a la zona designada como objetivo. El bloque “*VGA output block*” (Figura 3-19) es el encargado de generar la señal

de video (VGA) para visualizar sobre un monitor el proceso de generación de los vectores de desplazamiento de una zona a otra del entorno virtual.

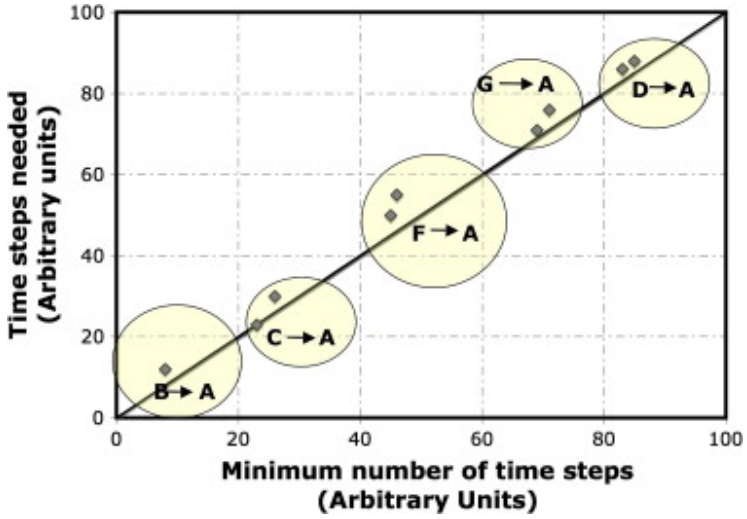

Figura 3-22: Ciclos de proceso necesarios para generar una trayectoria que conduzca desde cualquier punto del entorno virtual hasta la zona A.

Para la evaluación del sistema hemos procedido a su implementación sobre una placa de entrenamiento educacional modelo "*ALTERA UP2 Unversity Program*", que incorporará una *FPGA* modelo EPF10K70RC240-4. Para evaluar el bloque hemos configurado el bloque "*Movement Decision Block*" con las direcciones específicas (predefinidas) para permitir que independientemente del punto de inicio en el entorno virtual (2D), el sistema se dirija hacia la "zona A". A continuación hemos procedido a realizar diferentes experimentos, situando el punto de origen del sistema en diferentes lugares del entorno 2D virtual para evaluar el número de ciclos de proceso que se necesitan para poder alcanzar la ruta que lleve desde el punto de origen a la zona A.

Cabe destacar que en todos los casos evaluados el sistema ha conseguido llegar a la zona A sin grandes dificultades. Los valores obtenidos para el período de convergencia a la zona A para los diferentes itinerarios desde las diversas zonas de origen, se presentan en la Figura 3-22 (los resultados experimentales se han representado mediante los rombos negros). A la vez en la misma figura se comparan los ciclos/período de proceso requeridos experimentalmente para alcanzar la zona A (eje Y de la Figura 3-22), frente al número

óptimo de ciclos de proceso requeridos para ir de una determinada zona hasta la zona A (eje X de la Figura 3-22), donde dicha relación se representa mediante la línea negra.

De los resultados obtenidos se desprende como cabría esperar, que las rutas que parten de orígenes en zonas cercanas a la *zona A* se correspondan a valores bajos del tiempo de proceso, mientras que en el caso de partir de zonas alejadas a la *zona A*, este tiempo se hace mucho mayor.
Si nos fijamos detalladamente en los resultados experimentales podremos apreciar como los ciclos de proceso obtenidos son extremadamente similares a los valores óptimos de ciclos de proceso, que se corresponden con la trayectoria óptima, para cualquiera de las zonas. Esto no hace más que demostrar la eficiencia computacional de la metodología de clasificación basada en el uso de bloques *MSC*.
Cabe reseñar a su vez la fiabilidad del sistema implementado, puesto que en el momento en el cual se pudieran confundir dos entornos similares correspondientes a dos órdenes de movimientos distintos, el sistema duda pero no se bloquea (problema del *local minima*). Éste es un fenómeno interesante puesto que el sistema, al dudar, prueba distintas opciones y no se queda estancado en una (la más probable) si esta fuese errónea. De esta forma, la incertidumbre inherente es la base de la solución a los problemas que plantea el entorno.

3.2.4.2 Implementación estocástica (UESL) de la metodología de comparación multidimensional

Como hemos descrito en la implementación (UCSL) de la *metodología de comparación multidimensional*, nuestro principal interés versa en el desarrollo de una metodología estocástica computacionalmente eficiente en el campo del reconocimiento de patrones estadístico. En el desarrollo de las f.d.p hallamos el problema que, para que éstas pudieran ser distribuciones normales (área igual a +1), se requería del uso de la multiplicación a la salida del bloque *gaussiana (UCSL)* por un factor mayor a +1 para que realmente el área encerrada debajo de dicha función fuera la unidad. Todo ello nos condujo a implementar un bloque *f.d.p normal* mediante la codificación (UESL). Éste es el motivo que nos ha llevado

a implementar el bloque clasificador *MSC* capaz de clasificar categorías descritas mediante la codificación estocástica (UESL).

A continuación se presenta una solución de comparador basada en dicha codificación.

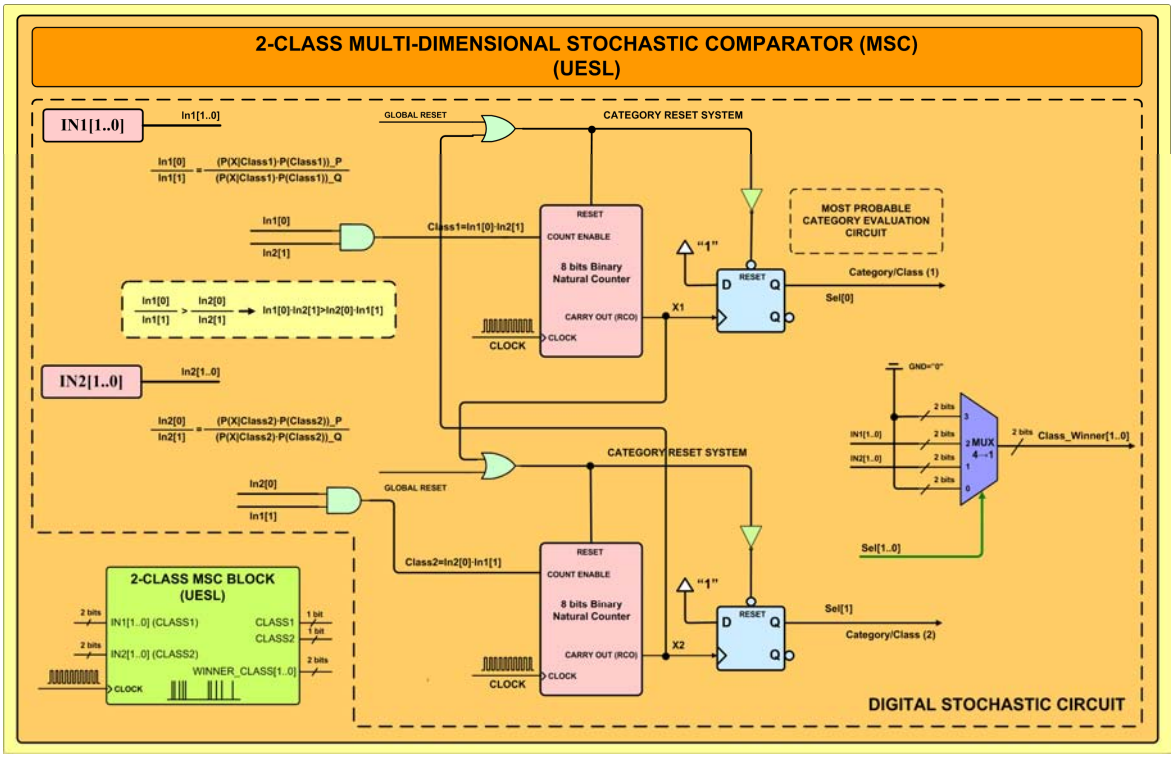

Figura 3-23: Bloque MSC clasificador de dos clases (UESL)

La implementación digital del bloque MSC en la codificación extendida (UESL) es prácticamente idéntica a la descrita para la codificación (UCSL), salvo la limitación que ahora sólo podremos proceder a comparar las clases por parejas. Esto es debido a que ahora debemos proceder a comparar razones de señales estocásticas, y no dos señales estocásticas independientes como anteriormente sucedía.

Si nos fijamos en la Ecuación [3-37] podemos apreciar cómo la comparación entre dos valores (A y B) en la codificación (UESL) es equivalente a la comparación de sus productos cruzados. De esta forma solamente se pueden comparar parejas de señales en lugar de poderlo hacer en conjunto.

$$\begin{cases} A = \dfrac{P}{Q} \\ B = \dfrac{R}{S} \end{cases}, \forall P,Q,R,S \in [0,+1] \rightarrow A > B \rightarrow \frac{P}{Q} > \frac{R}{S} \rightarrow P \cdot S > R \cdot Q$$

Ecuación [3-37]

Como veremos a continuación esto no es un problema insalvable, puesto que hemos conseguido sistematizar la implementación de estos bloques para permitir la clasificación de N clases (UESL) a la vez.

El bloque clasificador *MSC* (UESL) (Figura 3-23) para dos clases (elemento base de esta metodología) se compone internamente de los siguientes elementos. El primer elemento que hallamos a la entrada de este bloque es una pareja de puertas **AND** que se encargan de realizar el producto cruzado estocástico entre las dos razones de entrada al bloque. La pareja de señales resultantes nos permitirá evaluar si la magnitud "A" es mayor que la "B" o viceversa, haciendo uso de un circuito muy parecido al que desarrollamos para la codificación clásica (UCSL). Su funcionamiento interno es el siguiente: las salidas de las puertas **AND** se conectan a las respectivas señales de activación "*Enable*" de conteo de una pareja de contadores de "N" bits (en nuestro caso de 8bits, que se corresponde a la mitad de bits del bloque B2P(N) de 16-bits usados genéricamente). De esta forma en el instante en que uno de los elementos contadores (correspondientes a la categoría '0' o '1') se desborde, se activará la señal de salida (*RCO*) (señal "X1" o "X2" fijada a nivel alto '1', según el caso) del contador en cuestión, la cual a su vez fijará la salida del respectivo biestable D a nivel alto '1'. A la vez mediante la señal "Xi" de la categoría más probable procederemos a resetear el contador y el biestable de salida de la categoría menos probable.

La limitación de sólo poder comparar dos magnitudes estocásticas a la vez mediante un mismo bloque (*MSC*), nos obligará a concatenar bloques (*MSC*) en forma piramidal (tal y como posteriormente describiremos. Para ello en cada bloque MSC (UESL) hemos incorporado un Multiplexor de 4 entradas de 2-bits (a fin de fijar a la salida, las dos señales estocásticas que conforman la razón de la clase más probable evaluada por el bloque). En la

entrada uno ‘1’ del multiplexor asignaremos las señales de entrada de la clase 0, mientras que en la entrada ‘2’ asignaremos las señales de entrada de la case ‘1’. Finalmente las entradas ‘0’ y ‘3’ las fijaremos a valor bajo ya que se corresponden a estados en los cuales nuestro circuito no puede ni debe encontrarse. De esta forma se evitaran posibles clasificaciones erróneas al iniciar el bloque.

Como señales de control de este multiplexor usaremos las salidas de cada uno de los biestables de cada categoría, asignando como bit de menor peso “sel[0]” la salida de la clase ’0’, y como bit de mayor peso “sel[1]” la salida de la clase ‘1’. Por lo tanto, la tabla de verdad que regirá este multiplexor es la siguiente (Tabla [3-11]):

Tabla 3-11: Tabla de verdad para la selección de la categoría de salida (MSC) (UESL)

Entrada de selección del multiplexor (4→1) (sel[1..0])	Razón de salida del bloque
0→b00→clase0=’0’ y clase1=’0’ (no admitida)	Salida fija a nivel bajo ‘0’
1→b01→clase0=’1’ y clase1=’0’	Clase 0
2→b10→clase0=’0’ y clase1=’1’	Clase 1
3→b11→clase0=’1’ y clase1=’1’ (no admitida)	Salida fija a nivel bajo ‘0’

Cabe remarcar que a la práctica, para saber qué valor de entrada es mayor nos limitaremos a monitorizar el valor de salida de la primera clase. Si la salida se halla a nivel bajo nos indicará que la clase 0 es la más probable, y si ésta se halla a nivel alto la más probable será la clase 1.

Una vez descrito el funcionamiento del bloque clasificador (MSC) básico en la codificación (UESL) pasaremos a describir la forma de combinar estos bloques para poder clasificar más de dos clases. Para ello deberemos proceder a concatenar etapas de bloques *MSC* (que etapa tras etapa compararan de dos en dos las clases más probables evaluadas enteriormente) hasta que hallemos la clase más probable de todas.

Para evaluar “k” clases, necesitaremos “m” etapas de comparadores de dos clases *MSC*, siendo la relación matemática entre ambas magnitudes la siguiente (Ecuación [3-38]):

$$\begin{cases} \mathrm{k}: \text{ clases} \\ \mathrm{m}: \text{ etapas} \end{cases} \rightarrow \begin{cases} k \leq 2^{m} \\ k > 2^{m-1} \end{cases}, \forall k, m \in \mathrm{N} \qquad \text{Ecuación [3-38]}$$

Donde para cada etapa requeriremos de “n” bloques comparadores de dos clases, siendo éste valor evaluado mediante la siguiente expresión (Ecuación [3-39]):

$$\begin{cases} \text{m}: \text{número de etapas} \\ \text{k}: \text{número de clases a comparar} \\ \text{n}: \text{número de bloques MSC presentes por etapa} \\ \text{i}: \text{índice de la etapa} \end{cases} \rightarrow n \leq \frac{k}{2^i}, \begin{cases} i \in 1,...,m \\ n \in \text{N} \end{cases}$$

Ecuación [3-39]

Mediante esta metodología el período mínimo de evaluación del sistema se verá incrementado con el número de etapas de bloques *MSC* (*UESL*) necesarios para clasificar todas las clases presentes en el sistema. El período mínimo de evaluación del bloque *MSC* (UESL) en función del número de etapas viene descrito por la Ecuación [3-40]:

$$\begin{cases} \text{m}: \text{número de etapas} \\ \text{z}: \text{número de bits del bloque contador usado} \rightarrow T_{\min} = m \cdot 2^z \\ \text{T}_{\min}: \text{periodo mínimo de evaluación} \end{cases} \qquad \text{Ecuación [3-40]}$$

A continuación presentaremos un ejemplo práctico (Figura 3-24) donde describiremos la implementación de un clasificador para 8 clases de entrada.

Si partimos del hecho de que el sistema se compone de “k=8” clases distintas, se requerirá de un mínimo de “m=3” etapas para evaluar la clase mayor del sistema. En la primera etapa necesitaremos “n=4” bloques *MSC* de dos entradas, en la segunda etapa “n=2” bloques *MSC* y en la tercera y última etapa “n=1” un solo bloque *MSC*.

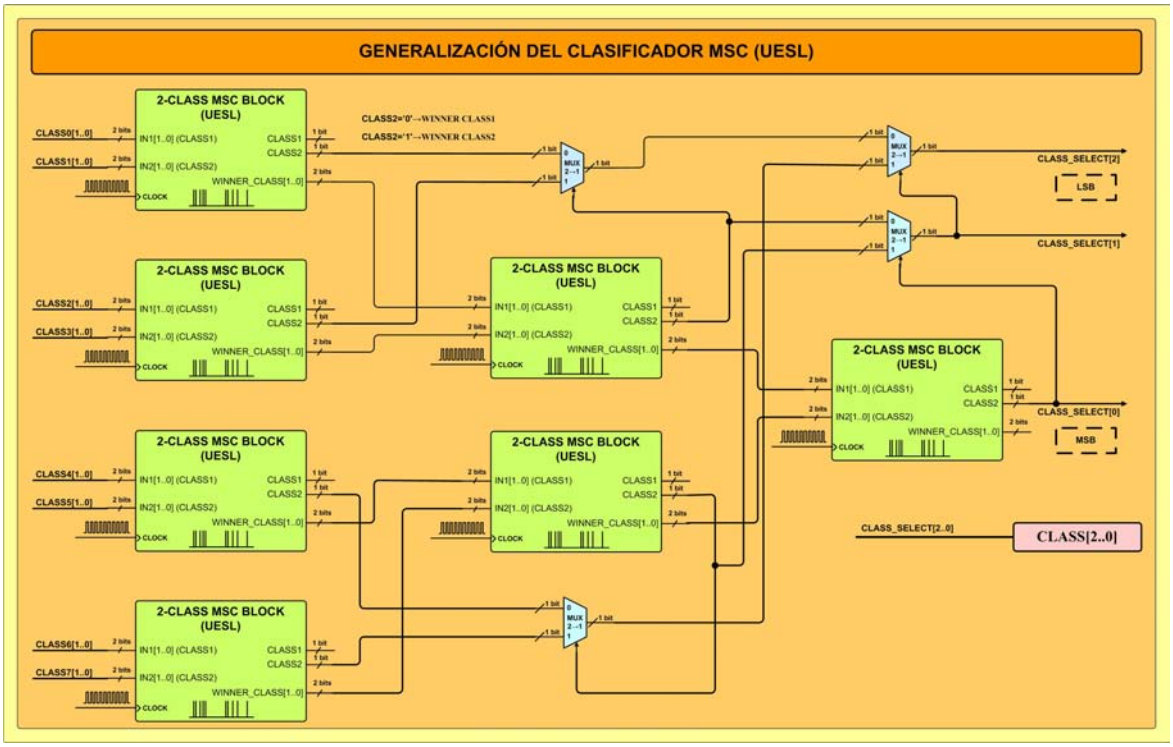

Figura 3-24: Generalización a N-clases del clasificador MSC (8 clases) (UESL)

En la Figura 3-24 se presenta la evaluación digital del código de la categoría “Category[m-1,..,0]” más probable del sistema, la cual se ha implementado de la siguiente forma: inicialmente se ha asignando el bit de mayor peso “Category[m-1]=Category[2]” a la salida de la clase 2 del bloque *MSC* de la última etapa. A la vez esta misma salida se ha conectado a la señal de selección de un multiplexor cuyas entradas son las salidas de los bloques *MSC* de la segunda etapa. Mediante este multiplexor evaluaremos el segundo bit más significativo “Category[1]”. Finalmente este bit (al igual que sucediera anteriormente) se hallará conectado a la entrada de selección de un tercer multiplexor a cuyas entradas tenemos conectadas la salida de una pareja de multiplexores previos encargados de determinar cuál era la categoría más probable de las dos ramas de decisión existentes. Todo esto matemáticamente se describe de la siguiente forma (Ecuación [3-40]):

$$
\begin{aligned}
\text{MSB} \rightarrow\ & \text{Category}[2] = \text{clase2}[\text{i} = 3, \text{n} = 1] \\
& \text{Category}[1] = \text{clase2}[\text{i} = 2, \text{n} = 1]\cdot\overline{\text{Category}[2]} + \text{clase2}[\text{i} = 2, \text{n} = 2]\cdot\text{Category}[2] \\
\text{LSB} \rightarrow\ & \text{Category}[0] = \left(\text{clase2}[\text{i} = 1, \text{n} = 1]\cdot\overline{\text{clase2}[\text{i} = 2, \text{n} = 1]} + \text{clase2}[\text{i} = 1, \text{n} = 2]\cdot\text{clase2}[\text{i} = 2, \text{n} = 1]\right)\overline{\text{Category}[1]} + \\
& + \left(\text{clase2}[\text{i} = 1, \text{n} = 3]\cdot\overline{\text{clase2}[\text{i} = 2, \text{n} = 2]} + \text{clase2}[\text{i} = 1, \text{n} = 4]\cdot\text{clase2}[\text{i} = 2, \text{n} = 2]\right)\text{Category}[1]
\end{aligned}
$$

Ecuación [3-40]

Donde el índice "i" se refiere a la capa donde se halla el bloque concatenado, mientras que "n" se refiere al identificador de nivel del bloque *MSC* en una determinada capa de concatenación.

Una vez descrito el funcionamiento del bloque *MSC* (*UESL*) pasaremos a evaluar en el siguiente sub-apartado si éste es realmente operativo en el marco del reconocimiento de patrones estadístico.

3.2.4.2.1 Evaluación de la metodología en el supuesto de sistema de cuatro clases (1D)

Hemos procedido a implementar un circuito calsificador (Figura 3-25) basado en la probabilidad a posteriori, para un conjunto de patrones "Ω" constituido por cuatro categorías restringidas a un espacio unidimensional (1D). El sistema se ha implementado experimentalmente en una FPGA de la familia Cyclone II.

En la figura 3-25 hallamos en primera instancia los cuatro bloques usados para implementar la función de probabilidad condicional de las cuatro clases "$P(x|C_i)$" distintas, para los cuales hemos usado sendos bloques *f.d.p normales* (UESL) (cuya área ha sido normalizada a +1). Posteriormente vemos cómo la salida de estos bloques acceden a un bloque multiplicador (UESL) encargado de multiplicar dicha distribución por la probabilidad a priori "$P(C_i)$" de cada una de las clases evaluadas. De esta forma se obtiene un factor proporcional a la probabilidad a posteriori para cada categoría y por lo tanto nuestro clasificador realizará la siguiente operación "$P(x|C_i)\cdot P(C_i) > P(x|C_j)\cdot P(C_j)$", donde $i \neq j$. Las probabilidades a priori $P(C_i)$ para las cuatro categorías se han fijado a un valor $P(C_i)=0.25$, es decir, las hemos definido como equiprobables, siendo los parámetros de configuración

para cada uno de los cuatro bloques *f.d.p normales* de cada categoría, los que se presentan en la Tabla [3-12]:

Tabla 3-12: Parámetros de configuración de las diferentes categorías

Clase 1

μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
16384	0,25	6000	655	0,0092	4,1504	65535	15790

P(Clase1)	P(Clase1) [16bits]
0,25	16384

Clase 2

μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
32768	0,5000	6000	655	0,0092	4,1504	65535	15790

P(Clase2)	P(Clase2) [16bits]
0,25	16384

Clase 3

μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
45056	0,6875	6000	655	0,0092	4,1504	65535	15790

P(Clase2)	P(Clase2) [16bits]
0,25	16384

Clase 4

μ [16bits]	Valor μ	EvaluationTime	C_adjust	σ (modelo)	Factor de normalización (P/Q)	Factor_P [16bits]	Factor_Q [16bits]
59392	0,9063	6000	655	0,0092	4,1504	65535	15790

P(Clase2)	P(Clase2) [16bits]
0,25	16384

Una vez evaluadas las razones proporcionales a la probabilidad a posteriori para cada una de las categorías hemos implementado el circuito clasificador estocástico mediante la concatenación de bloques *MSC* (UESL) de dos clases, usando la metodología descrita en el subapartado anterior.

Si partimos del hecho de que nuestro sistema de clasificación se compone de “k=4” clases distintas, se requerirá de un mínimo de “m=2” capas para evaluar la clase mayor del sistema. En la primera etapa necesitaremos “n=2” bloques *MSC* de dos entradas, en la segunda etapa necesitaremos “n=1” un sólo bloque *MSC*.

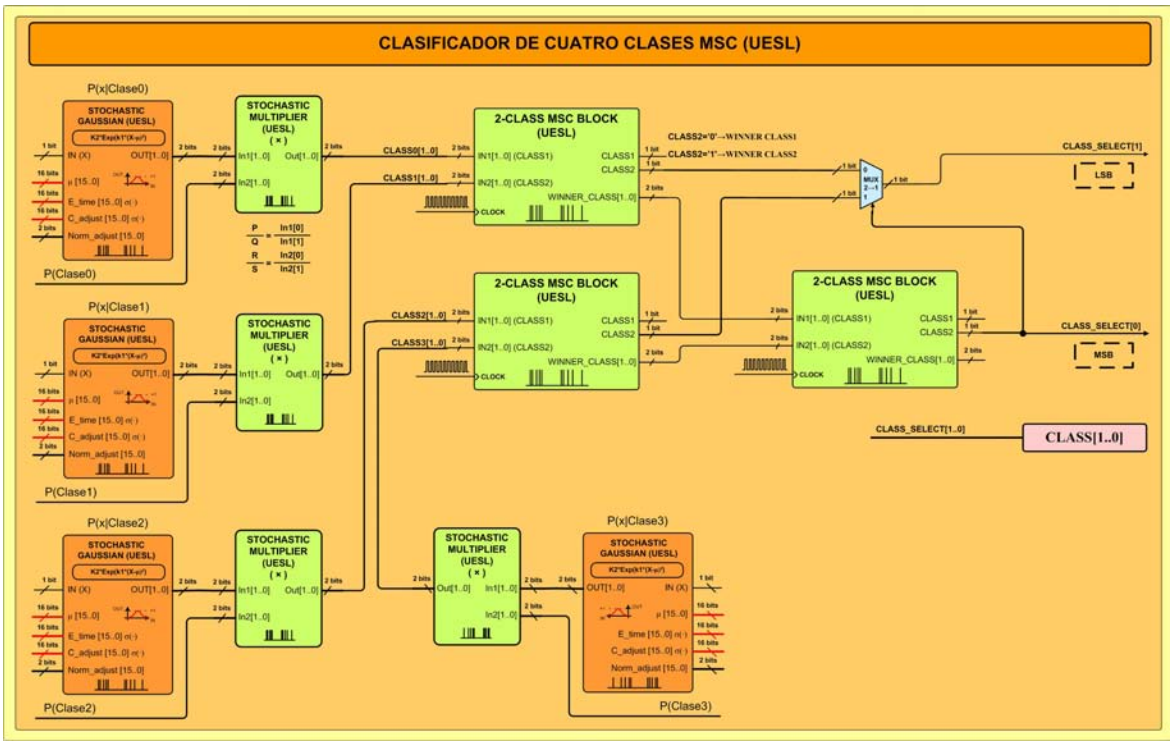

Figura 3-25: Circuito clasificador MSC para 4 clases (UESL)

En la figura 3-25 se presenta la evaluación digital del código de la categoría “Category[m-1,..,0]” más probable. El bit de mayor peso de la señal de evaluación de categoría a la salida de la clase 2 del bloque *MSC* de la segunda capa está conectado a la señal de selección de un multiplexor cuyas entradas son las salidas de la clase 2 de los bloques *MSC* de la primera etapa. De esta forma evaluaremos el bit menos significativo de la clasificación “Category[0]”. El mecanismo de evaluación de las diferentes clases vendrá descrito por la Ecuación [3-41]:

$$\text{MSB} \rightarrow \text{Category}[1] = \text{clase2}[i = 2, n = 1]$$
$$\text{LSB} \rightarrow \text{Category}[0] = \text{clase2}[i = 1, n = 1]\cdot\overline{\text{Category}[2]} + \text{clase2}[i = 1, n = 2]\cdot\text{Category}[2]$$

Ecuación [3-41]

Donde el índice “i” se refiere a la capa donde se halla el bloque MSC concatenado, mientras que “n” se refiere al identificador de nivel del bloque *MSC* en una determinada capa de concatenación.

Una vez configuradas las probabilidades condicionales y las probabilidades a priori para cada categoría con los parámetros de la Tabla [3-12] hemos procedido a variar la señal estocástica de entrada "IN(X)" en el intervalo [0, +1]. Los resultados experimentales que se han obtenido se presentan en la Tabla [3-13]:

Tabla 3-13: Medidas experimentales del circuito MSC para la clasificación de 4 clases (UESL)

Datos de entrada [Experimental]						Experimental	Teórico
IN [16bits]	Valor IN	Clase 1 Distribución Gaussiana	Clase 2 Distribución Gaussiana	Clase 3 Distribución Gaussiana	Clase 4 Distribución Gaussiana	MSC Clase Medida	Clase de salida
0	0,0000	0,0005	0,0000	0,0000	0,0000	0	0
2048	0,0313	0,3971	0,0000	0,0000	0,0000	0	0
4096	0,0625	0,7663	0,0000	0,0000	0,0000	0	0
6144	0,0938	1,1381	0,0000	0,0000	0,0000	0	0
8192	0,1250	1,9621	0,0000	0,0000	0,0000	0	0
10240	0,1563	2,6746	0,0000	0,0000	0,0000	0	0
12288	0,1875	3,2127	0,0000	0,0000	0,0000	0	0
14336	0,2188	3,7223	0,0000	0,0000	0,0000	0	0
16384	0,2500	4,1777	0,0716	0,0000	0,0000	0	0
18432	0,2813	3,7943	0,3813	0,0000	0,0000	0	0
20480	0,3125	3,3946	0,7877	0,0000	0,0000	0	0
22528	0,3438	2,6212	1,1497	0,0005	0,0000	0	0
24576	0,3750	1,8773	1,8837	0,0001	0,0000	1	1
26624	0,4063	1,1552	2,6502	0,0004	0,0000	1	1
28672	0,4375	0,7540	3,3668	0,0001	0,0000	1	1
30720	0,4688	0,3881	3,8447	0,3852	0,0000	1	1
32768	0,5000	0,0004	4,1326	0,7796	0,0000	1	1
34816	0,5313	0,0001	3,8340	1,1449	0,0002	1	1
36864	0,5625	0,0000	3,4522	1,8442	0,0003	1	1
38912	0,5938	0,0000	2,5836	2,6357	0,0000	2	2
40960	0,6250	0,0000	1,8750	3,3737	0,0001	2	2
43008	0,6563	0,0000	1,1448	3,8245	0,0001	2	2
45056	0,6875	0,0000	0,7663	4,1565	0,3901	2	2
47104	0,7188	0,0000	0,3765	3,7516	0,7580	2	2
49152	0,7500	0,0000	0,0001	3,2205	1,2311	2	2
51200	0,7813	0,0000	0,0001	2,6188	1,8447	2	2
53248	0,8125	0,0000	0,0000	1,7180	2,4959	3	3
55296	0,8438	0,0000	0,0000	1,1276	3,4044	3	3
57344	0,8750	0,0000	0,0000	0,7459	3,7032	3	3
59392	0,9063	0,0000	0,0000	0,3797	4,1575	3	3
61440	0,9375	0,0000	0,0000	0,0001	3,7214	3	3
63488	0,9688	0,0000	0,0000	0,0000	3,3639	3	3
Área experimental:		**1,0012**	**1,0094**	**0,9909**	**0,7309**		

En la Figura 3-26 representamos las diferentes distribuciones de probabilidad condicional $P(IN(X)|C_i)$ obtenidas (i=1...4) para cada una de las cuatro clases existentes (Clase 1 (línea roja círculos), Clase 2 (línea verde triángulos), Clase 3 (línea azul cuadrados) y la clase 4 (línea roja estrellas)). También hemos representado el valor de la magnitud digital

“Category[1..0]” qué nos indicará la categoría evaluada mediante nuestro sistema (línea negra).

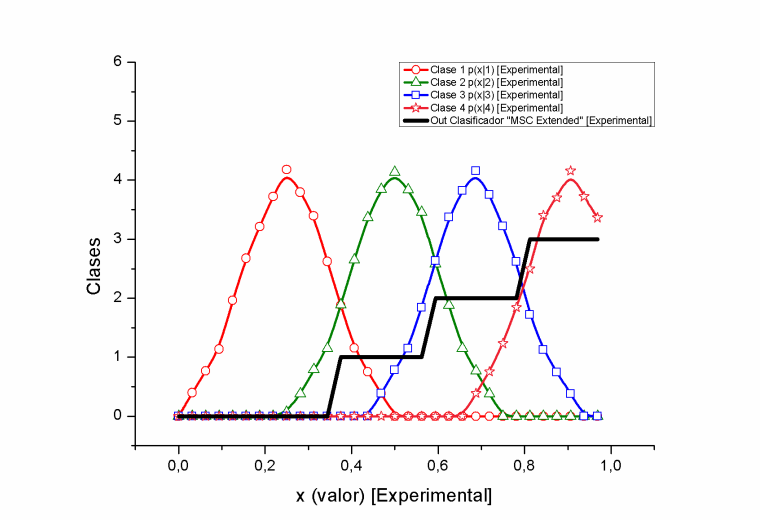

Figura 3-26: Medidas experimentales del clasificador MSC (UESL) para un sistema 1D de 4 clases

De la figura 3-26 y de las dos últimas columnas de la Tabla [3-13] (valores teóricos y experimentales de la clase más probable) se deduce claramente que la implementación (UESL) clasifica correctamente bajo cualquier circunstancia la categoría más probable.

3.3 EL RECONOCIMIENTO DE PATRONES ESTOCÁSTICO EN EL SUPUESTO DE DENSIDADES DE PROBABILIDAD CONDICIONALES DESCONOCIDAS

A lo largo del presente capítulo abordamos el diseño e implementación de los bloques estocásticos necesarios (en sus diferentes codificaciones) para poder obtener la probabilidad a posteriori de una distribución de probabilidad dada; así como un mecanismo de clasificación eficiente computacionalmente como es el *MSC* (basado en el principio computacional del “*Winner-Take-All*”) para la clasificación sistemática de sistemas constituidos por N clases. Todos ellos han sido evaluados mediante funciones densidad de

probabilidad conocidas, minimizando así la complejidad de evaluación de éstas. Con lo cual, a la hora de afrontar un problema real (bases de datos de moléculas bio-activas, históricos de consumo energético, reconocimiento de objetos y entornos, etc...) en el marco del reconocimiento de patrones estadístico no dispondremos nunca de ninguna función distribución de probabilidad condicional para las diferentes clases existentes en el sistema. Todo ello se suma al hecho que generalmente en los problemas reales las distribuciones de probabilidad condicional de una clase no podrán ser modeladas mediante distribuciones de probabilidad puramente normales (por lo tanto, los métodos paramétricos para la evaluación de estas distribuciones de probabilidad no serán operativos), ya que su forma real dista mucho de ésta, pudiendo ser tanto continua como discontinua a lo largo del espacio de evaluación del sistema. Por lo tanto en la literatura hallamos [3-1] que para la evaluación estadística de estas distribuciones deberemos hacer uso de métodos no-paramétricos como pueden ser el de los *k próximos vecinos* "k-NN" o el de las *ventanas de Parzen*. Todo esto lo describiremos a lo largo de la introducción, para posteriormente pasar a describir la implementación estocástica de la técnica de las *ventanas de Parzen* para la evaluación de densidades de probabilidad condicionales de un sistema a partir de unos datos de entrenamiento (evaluados experimentalmente). Finalmente procederemos a evaluar la operatividad de estas distribuciones para la resolución de un problema de reconocimiento de patrones simple.

3.3.1 INTRODUCCIÓN

En la práctica existen dos tipos de métodos no-paramétricos [3-1, 3-6] para la estimación de las funciones densidad de probabilidad condicional $p(x \mid w_i)$ correspondiente a una determinada clase. El primero de ellos intenta estimar la función de densidad de probabilidad a priori $p(x \mid w_i)$ a partir de las muestras o patrones de entrenamiento como sucede con el método de las *ventanas de Parzen* [3-25, 3-26], mientras que el segundo intenta estimar la función de densidad de probabilidad a posteriori (para así obtener las funciones de decisión o las funciones de frontera entre clases) como es el método de los *k-próximos vecinos* k-NN [3-27]. Estas dos metodologías constituyen dos formas de afrontar un mismo problema. El primer caso (ventanas de *Parzen*) es una aproximación estadística

al problema, mientras que el segundo (*k-próximos vecinos*) puede interpretarse como una aproximación geométrica al problema. Aunque, como veremos estas dos técnicas son mucho más semejantes en su base de lo que uno a priori cabría esperar.

Cabe remarcar que la mayoría de las técnicas no paramétricas intentan estimar la función densidad de probabilidad condicional $p(x \mid w_i)$ desconocida, basándose en el hecho que la probabilidad "*P*" que un vector "*x*" se encuentre en una región $R \subset \Re^d$, viene dada por la Ecuación [3-42]:

$$P = \int_R p(u) \cdot du \qquad \text{Ecuación [3-42]}$$

Como se puede apreciar la probabilidad "P" evaluada por la Ecuación [3-42], no es más que una versión suavizada o promediada de la función densidad de probabilidad $p(x)$, lo que implicará que podremos estimar el valor de la función densidad a partir de la probabilidad "P". Asumiendo que se dispone de "N" muestras $\{x_1, x_2, x_3, x_4, ..., x_N\}$ de una determinada distribución, las cuales son independientes entre sí e independientemente distribuidas, entonces la probabilidad de que "k" de las "N" muestras se hallen en dicha región "R", vendrá dada por una distribución binomial (Ecuación [3-43]).

$$P_k = \binom{N}{k} p^k (1-P)^{N-k} \qquad \text{Ecuación [3-43]}$$

Pudiéndose estimar el valor de P, como la fracción de las N nuestras que se hallan encerradas en dicha región "R" mediante la siguiente expresión (Ecuación [3-44]):

$$P = \int_R p(u) \cdot du \approx p(x) \int_R du = p(x) \cdot V_R \qquad \text{Ecuación [3-44]}$$

Donde V_R es el volumen encerrado por la región "R" del espació completo $\Re^d$. Entonces si combinamos las ecuaciones (Ecuación [3-43] y Ecuación [3-44]), se obtiene una expresión (Ecuación [3-46]) que estima el valor de la densidad de probabilidad en dicha región "R".

$$\frac{k}{N} \approx p(x)V_R \rightarrow p(x) \approx \frac{k}{NV_R} \qquad \text{Ecuación [3-45]}$$

La Ecuación [3-45] se obtendrá para el caso de un número ilimitado de muestras y en caso que el volumen de la región "R" se haga cada vez más y más pequeño. En la práctica el número de muestras será siempre limitado, y el volumen V_R no puede hacerse arbitrariamente pequeño, ya que podría llegar el punto en que no contuviera ninguna de las muestras $(p(x) \approx 0)$ o un número no significativo de éstas. Todo ello implica que en la práctica la función densidad de probabilidad estimada $p(x)$ se hallará en una cierta fracción promediada implícitamente, y deberá asumirse una cierta varianza en dicha probabilidad, es decir en la relación (k/N).

Es interesante presentar desde el punto de vista teórico cómo estas limitaciones pueden afrontarse. Para estimar la función densidad de probabilidad en el punto "x", crearemos un conjunto de regiones $\{R_1, R_2, R_3, R_4, ..., R_N\}$ que contengan dicho punto "x"; donde la primera región contiene una muestra, la segunda región dos muestras (la anterior y una más), y así sucesivamente. V_N es el volumen de la región R_N, k_N es el número de muestras que se hallan encerradas en dicha región R_N, y $p_N(x)$ es la N-ésima estimación de la densidad de probabilidad $p(x)$ en "x", la cual vendrá regida por la siguiente expresión (Ecuación [3-46]):

$$p_N(x) = \frac{k_N}{NV_N} \qquad \text{Ecuación [3-46]}$$

Por lo tanto, de la Ecuación [3-46] se deduce que para que $p_N(x)$ pueda converger a la densidad de probabilidad $p(x)$, deberán cumplirse las siguientes condiciones (Ecuación [3-47]):

$$\begin{pmatrix} \lim_{N\to\infty} V_N = 0 \\ \lim_{N\to\infty} k_N = \infty \\ \lim_{N\to\infty} \frac{k_N}{N} = 0 \end{pmatrix} \qquad \text{Ecuación [3-47]}$$

La primera de las condiciones trata de asegurar que el espacio de probabilidades promedio P/V converja a $p(x)$ puesto que las diversas regiones se van encogiendo uniformemente unas dentro de las otras también intenta que la densidad de probabilidad $p(\cdot)$ sea continua en la posición "x". La segunda de las condiciones sólo tiene sentido si se asegura que el ratio de frecuencia converja (en probabilidad) a la probabilidad "P", mientras que la tercera condición es imprescindible si se pretende evaluar $p_N(x)$ mediante la Ecuación [3-46], y se quiere que ésta converja completamente. En resumen, con esta última condición se impone que aunque un gran número de muestras se encuentren encerradas en la región R_N, la fracción de éstas será insignificante frente al número total de muestras.

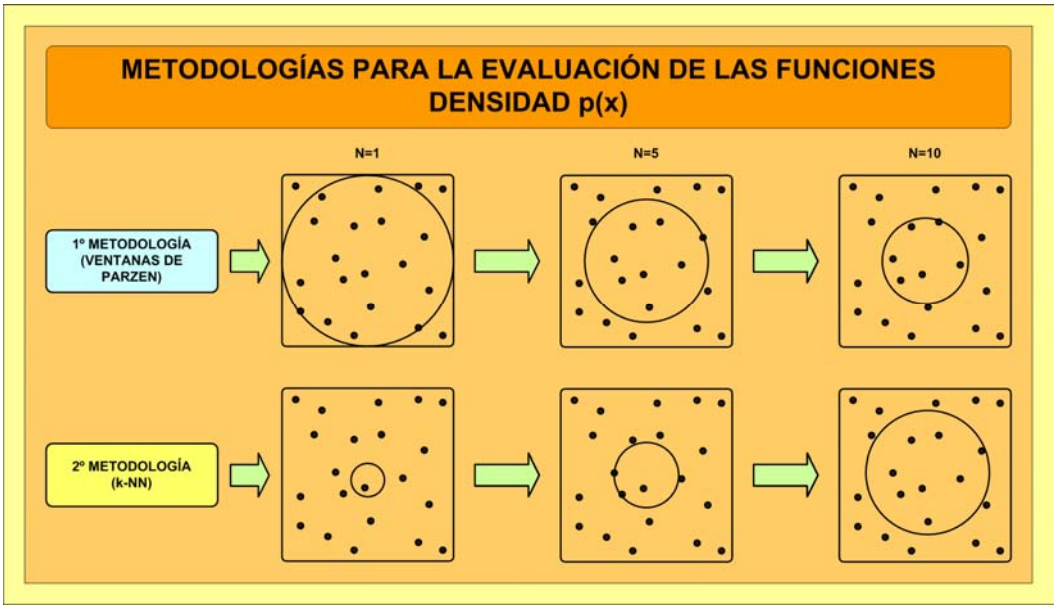

Figura 3-27: Principios de operación de los métodos k-NN y Parzen.

En la pràctica sólo existen dos caminos o métodos (Figura 3-27) para conseguir secuencias de regiones que cumplan con las condiciones anteriormente descritas (Ecuación [3-47]). El primer camino consiste en prefijar que el volumen de la región V_N sea decreciente en función de "N" escogiendo así la región inicial. La Ecuación [3-48] es una expresión válida para V_N ya que cumple con estas premisas.

$$V_N(x) = \frac{1}{\sqrt{N}}$$ Ecuación [3-48]

Mediante la ecuación [3-48] es fácil comprobar cómo el número de muestras aleatorias k_N confinadas en la región R_N, y la relación k_N / N se comporta correctamente, o al menos hasta el punto "*x*". De esta forma se consigue que la densidad de probabilidad estimada $p_N(x)$ converja hacia el valor de la densidad de probabilidad real $p(x)$. Esta primera aproximación a la resolución del problema es la base del método de las ventanas de *Parzen*, que se describe con detalle en el siguiente subapartado.

La segunda forma de asegurar la convergencia de dichas estimaciones se basa en prefijar k_N (número de muestras que se hallan encerradas en la región R_N) en función del número total de muestras del sistema "N", siendo una expresión para k_N que cumpla con estas premisas la que se presenta mediante la Ecuación [3-49]:

$$k_N(x) = \sqrt{N}$$ Ecuación [3-49]

En este caso el volumen V_N encerrado por la región R_N decrecerá hasta que únicamente incluya los k_N próximos vecinos de "x". Esta aproximación es la base del método llamado de los *k-próximos vecinos* (k-NN). Por lo tanto los dos caminos o métodos propuestos se aseguran que el sistema converja a la densidad de probabilidad real $p(x)$.

3.3.2 RECONOCIMIENTO DE PATRONES ESTADÍSTICO DE DISTRIBUCIONES NO-PARAMÉTRICAS MEDIANTE LA METODOLOGÍA DE VENTANAS DE PARZEN

A lo largo del siguiente apartado describiremos detalladamente la metodología de las *ventanas de Parzen* para la estimación de las funciones densidad de probabilidad condicional de una determinada clase.

Para introducir esta metodología debemos visualizar una región R_N que delimita un hipercubo de d-dimensiones, con una longitud h_N para cada uno de sus lados. El volumen vendrá dado por la Ecuación [3-50]:

$$V_N = h_N^d \qquad \text{Ecuación [3-50]}$$

Para obtener una expresión analítica para k_N, que se corresponda con el número de muestras que se encuentran encerradas en dicho volumen V_N, se requiere de la definición de una función de ventana $\varphi(u)$.

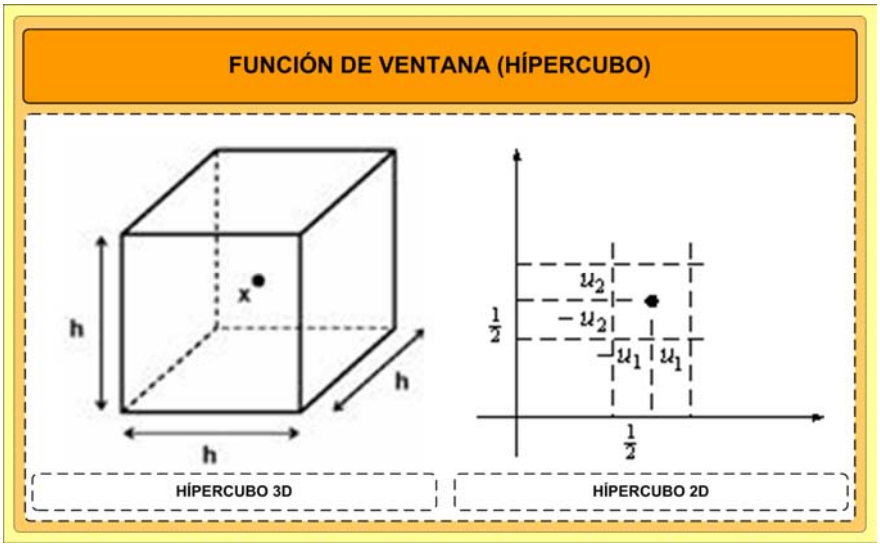

Figura 3-28: Hipercubo unitario

Siendo la función ventana más simple la que viene dada por la siguiente Ecuación [3-51]:

$$\varphi(u) = \begin{cases} 1 \rightarrow |u_j| \leq 0.5,\ j = 1,...,d \\ 0 \rightarrow \text{Resto de casos} \end{cases} \qquad \text{Ecuación [3-51]}$$

Donde $\varphi(u)$ define un hipercubo unitario centrado en el origen. En el caso que la función no se halle centrada en el origen de coordenadas sino e un punto “x”, la expresión para la función $\varphi(u)$ se regirá por la siguiente Ecuación [3-52]:

$$\varphi(u)=\varphi\left(\frac{x-x_i}{h_N}\right)=\begin{cases}1\rightarrow\left|\frac{x-x_i}{h_N}\right|\leq 0.5,\ j=1,...,d\\ 0\rightarrow \text{Re}sto\_de\_casos\end{cases}$$ Ecuación [3-52]

Por ello, $\varphi(u)$ valdrá "1" en el caso que la posición x_i de una muestra se halle confinada en el volumen V_N centrado en el punto "x". La función $\varphi(u)$ valdrá "0" en el resto de casos. Por lo tanto, el subconjunto de las N muestras que se hayan confinadas en dicho hipercubo vendrá dado por la Ecuación [3-53]:

$$k_N=\sum_{i=1}^{N}\varphi\left(\frac{x-x_i}{h_N}\right)$$ Ecuación [3-53]

Si ahora sustituimos la Ecuación [3-53] en la Ecuación [3-46] (obtenida en el apartado anterior para la estimación de la densidad de probabilidad) obtendremos una nueva expresión genérica (Ecuación [3-54]) para la estimación de la función densidad de probabilidad.

$$p_n(x)=\frac{1}{N}\sum_{i=1}^{N}\frac{\varphi\left(\frac{x-x_i}{h_N}\right)}{V_N}$$ Ecuación [3-54]

De la ecuación [3-54] se deduce que la estimación obtenida para $p_N(x)$ es básicamente un promedio de funciones que dependen de la posición de origen "x" y de la posición de la muestra x_i. En esencia la función de ventana $\varphi(u)$ se usa para interpolar; ya que cada muestra contribuye a la estimación de la función densidad de probabilidad en función de su distancia a la posición de origen "x".

A partir de la Ecuación [3-54] se obtiene una forma mucho más genérica para la estimación de las funciones densidad de probabilidad de lo que cabría esperar a priori. Por lo tanto si pretendemos obtener una estimación legítima, o lo que es lo mismo, bien comportada (sin discontinuidades) de la función densidad de probabilidad será necesario que ésta cumpla al menos con dos premisas. Su valor nunca será negativo y la integral de la función sobre todo el espacio valdrá "1", es decir se hallará normalizada a la unidad. Esto podrá asegurarse si

la función de ventana usada para su evaluación cumple con las anteriores premisas. Más concretamente, ésta deberá cumplir con las siguientes condiciones (Ecuación [3-55]):

$$\begin{cases} \varphi(x) \geq 0 \\ \int \varphi(u) du = 1 \end{cases} \qquad \text{Ecuación [3-55]}$$

Existe una tercera condición relacionada con el volumen de la región V_N, el cual deberá ser decreciente con el número N. Por lo tanto, en el caso del volumen del hipercubo que se rige por la relación $V_N = h_N^d$, podremos asegurar que la estimación $p_N(x)$ de la función densidad cumple con estas condiciones.

Antes de proseguir es necesario examinar cuál es el efecto de la variación en la anchura de los lados del hipercubo sobre la estimación de la función densidad de probabilidad $p_N(x)$. Para ello definiremos una función $\delta_N(x)$ que vendrá dada por la Ecuación [3-56].

$$\delta_N(x) = \frac{\varphi\left(\frac{x}{h_N}\right)}{V_N} =\rightarrow Hipercubo \rightarrow = \frac{\varphi\left(\frac{x}{h_N}\right)}{h_N^d} \qquad \text{Ecuación [3-56]}$$

La estimación de la función densidad de probabilidad $p_N(x)$ viene a ser como una media aritmética de las N funciones $\delta_N(x)$ del sistema que contribuyen a ésta (una para cada muestra), tal y como se describe en la Ecuación [3-57].

$$p_N(x) = \frac{\sum_{i=1}^{N} \delta_N(x - x_i)}{N} \qquad \text{Ecuación [3-57]}$$

En el caso del hipercubo cuyo volumen viene regido por la relación $V_N = h_N^d$, una modificación de la distancia de los lados h_N afectará a la vez a la amplitud y la anchura de la función $\delta_N(x)$.

En el caso que h_N sea muy grande (un hipercubo grande), el valor de la función $\delta_N(x)$ se irá haciendo cada vez más pequeña, pero a la vez contribuirán a dicha función muestras cuya posición x_i se hallará cada vez más alejada del origen del sistema "x", con lo cual el

rango de variaciones variará desde $\delta_N(0)$ hasta $\delta_N(x-x_i)$, siendo en este caso el rango de variación de la amplitud muy amplio y ésta muy suave. Por lo tanto, la estimación de la función densidad de probabilidad $p_N(x)$ consistirá en la superposición de N funciones $\delta_N(x-x_i)$ generales, caracterizadas por una variación muy lenta y suave; que desembocará en una estimación de la función densidad de probabilidad $p(x)$ prácticamente desenfocada. En el caso que h_N sea muy pequeño, la amplitud de la función $\delta_N(x-x_i)$ será muy grande cerca del punto $x=x_i$. Por lo tanto, la estimación de la función densidad de probabilidad $p(x)$ se obtendrá a partir de la superposición de N funciones muy abruptas centradas prácticamente en cada una de las muestras, lo cual contribuirá a una estimación errática o ruidosa.

$$\int \delta_N(x-x_i)dx = \int \frac{1}{V_N}\delta_N\left(\frac{x-x_i}{h_N}\right)dx = \begin{cases}\int \delta_N(u)du = 1 \\ u = x - x_i\end{cases}$$

$$(\text{Hípercubo}) \rightarrow \{u = x - x_i\} \rightarrow = \frac{1}{h_N^d}\int_{-h_N/2,\ldots,-h_N/2}^{+h_N/2,\ldots,+h_N/2} du = \qquad \text{Ecuación [3-58]}$$

$$= \frac{1}{h_N^d}h_N^d = 1$$

Para cualquier valor de h_N la distribución de probabilidad deberá hallarse normalizada a la unidad, por lo tanto deberá cumplir con la condición descrita por la Ecuación [3-58].

Así pues, a medida que h_N se vaya haciendo más pequeño (acercándose a cero) $\delta_N(x-x_i)$ se aproximará a una función delta de Dirac centrada en la posición de la muestra x_i, y por lo tanto $p_N(x)$ se aproximará a una superposición de deltas de Dirac centradas en las muestras. Esto muestra con claridad la importancia de la elección de una distancia de arista h_N o en su caso de un volumen V_N adecuado, ya que su valor afectará directamente a la estimación de la función densidad de probabilidad $p(x)$. En el caso en que V_N sea demasiado grande la estimación de la función densidad de probabilidad $p(x)$ tendrá muy

poca resolución. Por el contrario, si el volumen V_N es demasiado pequeño la estimación de la función densidad de probabilidad $p(x)$ tendrá una gran dispersión estadística.

De esta forma, con un número finito de muestras (caso general), lo idóneo es buscar una solución de compromiso para el valor del volumen V_N. En el caso en que se disponga de un número ilimitado de muestras será viable obtener un volumen V_N que se acerque lentamente a cero. A medida que el número de muestras "N" va creciendo, la función $p_N(x)$ va convergiendo hacia el valor exacto de la función densidad de probabilidad real. La convergencia anteriormente descrita se refiere concretamente a la convergencia de una secuencia de variables aleatorias. Para cualquier posición de origen "x", el valor de $p_N(x)$ dependerá de las diferentes posiciones de las muestras aleatorias $\{x_1, x_2, ..., x_N\}$. Por lo tanto, la función $p_N(x)$ vendrá descrita por una media $\overline{p_N(x)}$ y una varianza $\sigma_N^2(x)$. De esta forma se podrá afirma que $p_N(x)$ converge a la función densidad de probabilidad real $p(x)$ siempre que se cumplan las siguientes condiciones (Ecuación [3-59]):

$$\begin{cases} \lim_{N\to\infty} \overline{p_N}(x) = p(x) \\ \lim_{N\to\infty} \sigma_N^2(x) = 0 \end{cases} \qquad \text{Ecuación [3-59]}$$

A fin de probar que dichas condiciones aseguran la convergencia de la estimación de la función densidad de probabilidad, éstas deberán aplicarse sobre una distribución desconocida creadas a partir de una función de ventana $\varphi(u)$ con una longitud de arista h_N. Obviamente es preciso que la función $p(\cdot)$ en la posición de origen "x" sea continua, y que la función de ventana cumpla que $\varphi(x) > 0$ y $\int \varphi(u)du = 1$.

Si buscamos evaluar funciones continuas $p(\cdot)$ mediante el uso de funciones de ventana discontinuas $\varphi(\cdot)$ (por ejemplo funciones escalón), la estimación resultante presentará discontinuidades. Por lo tanto, es recomendable hacer uso de funciones de ventana o kernels $\varphi(\cdot)$ continuos (kernels Gaussianos), a fin de evitar las discontinuidades en las estimaciones.

A partir de las anteriores premisas se deducen las siguientes condiciones adicionales (Ecuación [3-60]) necesarias para asegurar la convergencia de la estimación de la función densidad de probabilidad $p_N(x)$.

$$\begin{cases} \underset{u}{Sup}\ \varphi(u) < \infty \\ \lim_{\|u\|\to\infty} \varphi(u)\prod_{i=1}^{d} u_i = 0 \\ \lim_{u\to\infty} V_N = 0 \\ \lim_{u\to\infty} NV_N = \infty \end{cases} \qquad \text{Ecuación [3-60]}$$

De la Ecuación [3-60] se desprende que las dos primeras expresiones se encargan de asegurar un buen comportamiento de la función de ventana $\varphi(\cdot)$, la cual satisfacen la mayoría de las funciones que uno podría usar (gaussianas, triangulares, escalones,…). Las dos últimas expresiones se encargan de asegurar que el volumen V_N tienda a cero, con un decaimiento sostenido más lento que "1/n". Estas condiciones son indispensables a fin de asegurar la convergencia de la media y la varianza de la distribución de probabilidad, tal y como mostramos a continuación.

Para demostrar la convergencia de la media, partiremos de la suposición que $\overline{p_N(x)}$ se corresponde con la de $p_N(x)$, y que las diferentes posiciones de x_i de las "N" muestras aleatorias son independientes entre si e idénticamente distribuidas. De acuerdo con la suposición realizada obtendremos la siguiente expresión matemática (Ecuación [3-61]) para la función densidad de probabilidad $p(x)$ desconocida:

$$\overline{p_N(x)} = \varepsilon[p_N(x)] = \frac{1}{N}\sum_{i=1}^{N}\varepsilon\left[\frac{\varphi\left(\frac{x-x_i}{h_N}\right)}{V_N}\right] = \qquad \text{Ecuación [3-61]}$$

$$= \int \frac{1}{V_N}\varphi\left(\frac{x-v}{h_N}\right)p(v)dv = \int \frac{1}{V_N}\delta_N(x-v)p(v)dv$$

La Ecuación [3-61] se deduce que el valor esperado de la estimación de la densidad de probabilidad $p_N(x)$ se corresponde con el valor promedio de la función densidad desconocida. También puede interpretarse como una convolución entre la función densidad desconocida $p(\cdot)$ y la función de ventana aplicada $\varphi(\cdot)$. Así pues, $\overline{p_N(x)}$ es la versión difuminada/borrosa de $p(x)$ vista a través de la ventana de promediado. El volumen V_N tenderá a cero, y la función $\delta_N(x-v)$ se aproxima a una función delta de Dirac centrada en "x". Así pues, si la función $p(\cdot)$ debe ser continua en "x", entonces la condición de $\lim_{u\to\infty} V_N = 0$ asegurará que $\overline{p_N(x)}$ se aproximará cada vez más a $p_N(x)$, a medida que el número de muestras N vaya creciendo.

$$\sigma_N^2(x) = \sum_{i=1}^{N} \varepsilon\left[\left(\frac{\varphi\left(\frac{x-x_i}{h_N} - \overline{p_N(x)}\right)}{NV_N}\right)^2\right] = N\varepsilon\left[\left(\frac{\varphi\left(\frac{x-x_i}{h_N}\right)}{NV_N}\right)^2\right] - \frac{\overline{p_N(x)}^2}{N} =$$

$$= \frac{1}{NV_N}\int \frac{1}{V_N}\varphi^2\left(\frac{x-v}{h_N}\right)p(v)dv - \frac{p_N^2(x)}{N}$$

Ecuación [3-62]

La Ecuación [3-62] muestra cómo no es imprescindible un número infinito de muestras para hacer que la función $\overline{p_N(x)}$ se aproxime a la densidad de probabilidad real $p(x)$. Para conseguir esto para cualquier número "N" de muestras se fuerza que el volumen V_N tienda a cero. Por lo tanto, para un conjunto de "N" muestras, una estimación abrupta de la función densidad de probabilidad resulta inútil. Este hecho resalta la necesidad de tener muy presente el valor de la varianza para la estimación de la densidad de probabilidad. Por lo tanto $p_N(x)$ se corresponderá con la suma de las funciones estadísticamente independientes de la "N" muestras aleatorias, con lo cual la varianza será la suma de las varianzas de cada uno de los términos, donde la varianza de la estimación de la probabilidad vendrá dada por la Ecuación [3-62]. Si en dicha ecuación despreciamos el segundo término, delimitando así el valor máximo que puede tomar la función de ventana

$\varphi(\cdot)$ e imponiendo la condición que $\lim_{u\to\infty} V_N = 0$, obtendremos la siguiente expresión para la varianza:

$$\sigma_N^2(x) \leq \frac{\sup(\varphi(\cdot))\overline{p_N}(x)}{NV_N} \qquad \text{Ecuación [3-63]}$$

De la Ecuación [3-63] se deduce que para obtener varianzas pequeñas se necesitarán valores de V_N grandes (y no pequeños como sucede con la media); ya que grandes volúmenes de V_N suavizan las variaciones locales de la función densidad de probabilidad. Si el numerador de la expresión se mantiene finito y el número de muestras N ubicado en el denominador tiende a infinito se puede asumir que el volumen V_N tienda a cero, a fin de minimizar la varianza de la distribución en vistas que el término NV_N tiende a infinito. Con ello se podrá tomar cualquier volumen V_N que venga regido por una expresión matemática que cumpla con las condiciones $\lim_{u\to\infty} NV_N = \infty$ y $\lim_{u\to\infty} V_N = 0$.

Este es el resultado teórico más importante que se puede obtener para la metodología de las *ventanas de Parzen*. Desafortunadamente no hay información acerca de cómo se debe elegir la función de ventana $\varphi(\cdot)$ ni el volumen del espacio V_N, a fin de obtener unos resultados óptimos (para un número finito de muestras). Esto es debido a que en realidad, a no ser que se disponga de información adicional acerca de la distribución de probabilidad real $p(x)$, más allá del hecho que ésta sea continua, no dispondremos de ninguna forma directa para optimizar los resultados a partir de un número finito de muestras.

3.3.2.1 Metodología de las ventanas de Parzen en el supuesto de un kernel Gaussiano

A continuación procederemos a describir de forma detallada el estimador de *Parzen* que usaremos en los siguientes subapartados, en los cuales describiremos de forma detallada la implementación estocástica de esta metodología.

La metodología de las *ventanas de Parzen* para la estimación de funciones densidad de probabilidad no es más que una técnica de interpolación de la información [3-1, 3-30, 3-31], la cual a partir de una muestra aleatoria de datos "z" pertenecientes a una misma categoría, permite estimar la f.d.p "P(z)" de cuya muestra deriva.

Supongamos que queremos estimar el valor de la *f.d.p* de una determinada clase "C_i" en un punto "*x*". Entonces procederemos a situar una función de ventana "$\varphi(\cdot)$" centrada en "*x*" y determinar cuantas observaciones (etiquetadas mediante su posición "x_i") se hallan delimitadas por dicha función de ventana, o bien cuál es la contribución de cada una de ellas "x_i" a la función de ventana. Por lo tanto, el valor de la *f.d.p* en el punto "*x*" vendrá dado por la suma de las contribuciones a ella que se hallan en una muestra aleatoria formada por "n" datos dados. Por lo tanto, la estimación de la *f.d.p* basada en las ventanas de *Parzen* vendrá dada genéricamente por la expresión:

$$P(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n^d} \cdot \varphi\left(\frac{x - x_i}{h_n}\right) \qquad \text{Ecuación [3-64]}$$

Donde "d" se corresponde con la dimensión de la función de ventana de Parzen usada, la cual a su vez deberá ser bien comportada. Esto quiere decir que cumple con las condiciones descritas en las ecuaciones (Ecuación [3-50], Ecuación [3-60]). Además el parámetro de ancho de banda o anchura de la ventana "h_n" deberá ser positiva, correspondiéndose genéricamente a la anchura de la función de ventana. Típicamente el parámetro de ancho de banda "h_n" se elegirá basándose en el número de observaciones disponibles para una muestra dada. Además la función de ventana o de Kernel "$\varphi(\cdot)$" será típicamente unimodal, siendo a su vez en sí misma una f.d.p. Este hecho facilitará mucho que la función f.d.p a estimar satisfaga las condiciones de una f.d.p bien comportada.

Una vez en este punto particularizaremos el resultado para el caso del kernel Gaussiano (f.d.p Gaussiana o distribución normal) (Ecuación [3-65]) que se corresponde con una de las funciones de ventana más usadas para la estimación de la función densidad de

probabilidad de una determinada categoría, ya que es infinitamente diferenciable, a la vez que prestará todas su propiedades a la función que ayudará a estimar.

$$\varphi(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}},x\in\Re \qquad \text{Ecuación [3-65]}$$

A la vez debemos tener muy presente que nos en el marco del reconocimiento de patrones estadísticos, a la vez que disponemos de un bloque estocastico funcional que implmenta la función Gaussiana (descrita en el capítulo segundo de la presente tesis). Este hecho nos permitirá estimar la f.d.p. directamente en Hardware (sintetizando el bloque en una FPGA) a partir de un conjunto de datos perteneciente a una determinada categoría. Para posteriormente proceder a evaluar si los datos de otra muestra aleatoria pertenecen o no a dicha categoría.

Por lo tanto, la f.d.p de un conjunto de "*n*" datos pertenecientes a una determinada categoría "C_i", en el supuesto que usemos un kernel Gaussiano unidimensional "*d=1*", vendrá dada por la combinación de las ecuaciones (Ecuación [3-65] y Ecuación [3-65]) obtenido así la siguiente expresión:

$$P(x\mid C_i)=P(x)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_x\sqrt{2\pi}}e^{-\frac{(x-x_i)^2}{2h_x^2}},x\in\Re \qquad \text{Ecuación [3-66]}$$

Dicha Ecuación [3-66] se corresponde al promedio de las "n" funciones Gaussianas centradas en un punto "$\mu_i=x_i$", debiéndose prefijar para cada muestra nuestra la desviación estándar "$h_x=\sigma_x$" asociada.

En el caso que pretendamos estimar la f.d.p de una distribución multidimensional, como podría ser una distribución tridimensional evaluada mediante un kernel Gaussiano (siempre suponiendo que cada una de las contribuciones dimensionales son independientes entre si) ésta se evaluará mediante la Ecuación [3-67]:

$$P(x,y,z\mid C_i)=P(x,y,z)=\frac{1}{n}\sum_{i=1}^{n}\left(\left(\frac{1}{h_x\sqrt{2\pi}}e^{-\frac{(x-x_i)^2}{2h_x^2}}\right)\cdot\left(\frac{1}{h_y\sqrt{2\pi}}e^{-\frac{(y-y_i)^2}{2h_y^2}}\right)\cdot\left(\frac{1}{h_z\sqrt{2\pi}}e^{-\frac{(z-z_i)^2}{2h_z^2}}\right)\right),x,y,z\in\Re$$

Ecuación [3-67]

Para el caso tridimensional, cada una de las "n" contribuciónes a la estimación de la f.d.p vendrá dada por un vector de posición "$\mu_i = \{x_i, y_i, z_i\}$". Al Igual que antes deberemos fijar un vector de desviaciones típicas o parámetros de ancho de banda "$h = \{h_x, h_y, h_z\}$" de todas las funciones de ventana. Normalmente igualaremos los valores del parámetro de ancho de banda para todas las contribuciones dimensionales de la distribución a fin de simplificar la estimación de la f.d.p.

Para finalizar con este apartado, debemos remarcar que prefijar el valor de la desviación estándar "$h_x = \sigma_x$" a un valor incorrecto o poco óptimo provocará que la f.d.p estimada se parezca más o menos a la real. Generalmente tomaremos un valor para el parámetro de ancho de banda del orden "$h \approx n^{-1/5}$" con una convergencia más lenta que en el caso de los métodos paramétricos tal y como veremos a lo largo del siguiente sub-apartado. A fin de presentar la importancia de una correcta elección del parámetro de ancho de banda hemos procedido a evaluar la estimación de una f.d.p en el intervalo "$x = [-1,+8]$" obtenida a partir de cinco contribuciones "n=5", las cuales se hallan localizadas en "$x_i = \{1,2,2.5,3,6\}$". Dicho proceso lo hemos repetido cuatro veces para diferentes valores del parámetro de ancho de banda "$h = \{0.25,0.5,1,2\}$". Los resultados obtenidos se presentan en la Figura 3-29. La f.d.p estimada para los diferentes valores del ancho de banda se corresponde en todas las gráficas con la línea verde con los círculos.

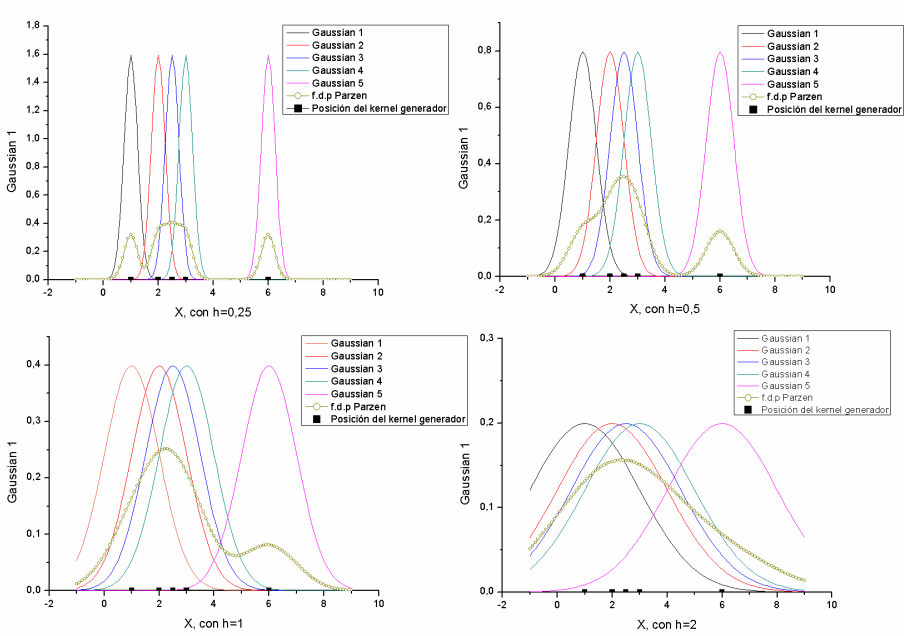

Figura 3-29: Efectos del valor del parámetro de ancho de banda en la estimación de la f.d.p

3.3.2.2 Elección del parámetro de ancho de banda óptimo

Como podemos apreciar en la Figura 3-29, la elección del parámetro de ancho de banda "h" es crucial para una correcta estimación de la f.d.p. " $\hat{f}_h(x)$ " a partir de una serie de datos experimentales. La elección de un parámetro "h" u otro [3-33] provocará que la f.d.p resultante sea una función suave o abrupta.

Por lo tanto, en el supuesto de una distribución de probabilidad unidimensional el parámetro de ancho de banda "h" óptimo será aquel que minimice el error de la f.d.p estimada " $\hat{p}_h(x)$ " y la f.d.p " $p(x)$ ". La forma genérica de evaluar esto será mediante el error cuadrático medio "*Mean Square Error*" (MSE) en un determinado punto "x", el cual vendrá dado por la Ecuación [3-68]:

$$MSE(x) = E\left[\left(\hat{p}_h(x) - p(x)\right)^2\right] = \underbrace{E\left[\hat{p}_h(x) - p(x)\right]^2}_{bias=sesgo} + \underbrace{\text{var}\left[\hat{p}_h(x)\right]}_{Varianza}$$

$$\hat{p}_h(x) = \sum_{i=1}^{n} \frac{1}{n \cdot h} \cdot \varphi\left(\frac{x - x_i}{h}\right)$$

Ecuación [3-68]

Donde la Ecuación [3-68] nos presenta un claro ejemplo de la relación de compromiso existente entre el valor de la varianza y del valor de Bias (sesgo estadístico). El valor de Bias de la estimación (Ecuación [3-69]) nos indicará cual ha sido el error sistemático incurrido por la estimación.

$$\begin{cases} \text{Bias} \rightarrow Bias\left(\hat{p}_h(x)\right) = \dfrac{h^2}{2} \cdot p''(x) \cdot \mu_2(x) + o\left(h^2\right), \quad h \rightarrow 0 \\ \mu_2(x) = \int s^2 \cdot \varphi(s) \cdot ds \end{cases}$$

Ecuación [3-69]

Por lo tanto, el valor de Bias será proporcional a "h^2", lo que provocará que un valor pequeño del parámetro "h" disminuya el valor de Bias. De todas formas el valor del sesgo estadístico (Bias) dependerá de la curvatura de la función estimada.

Por otra parte, la varianza (Ecuación [3-70]) nos indicará el error aleatorio incurrido en dicha estimación.

$$\begin{cases} \text{Varianza} \rightarrow Var\left(\hat{p}_h(x)\right) = \dfrac{1}{n \cdot h} \cdot \|\varphi\|_2^2 \cdot p(x) + o\left(\dfrac{1}{n \cdot h}\right), \quad \text{n} \cdot h \rightarrow \infty \\ \|\varphi\|_2^2 = \int \varphi^2(s) \cdot ds, \text{ que se corresponde con la norma cuadrada } \text{L}_2 \text{ de } \varphi(\ ) \end{cases}$$

Ecuación [3-70]

De lo que se deduce que la varianza será proporcional a "$(n \cdot h)^{-1}$", lo que provocará que valores grandes de "h" o "n" reduzcan la varianza de la distribución. A su vez, funciones de ventana planas reducirán la varianza ya que ésta es proporcional a "$\|\varphi\|_2^2$".

Por lo tanto, un incremento del valor del parámetro de ancho de banda "h" provocará una disminución de la varianza, a la vez que aumentará el valor de Bias, y viceversa.

Todo ello nos conduce a que para obtener el parámetro de ancho de banda óptimo lo más fiable será comparar nuestra f.d.p estimada con una distribución de probabilidad estándar de referencia. Por lo tanto, asumiendo una f.d.p estándar deberemos hallar el parámetro del

ancho de banda que minimice la integral del error cuadrático medio "*Mean Integrate Square Error*" (MISE) (es decir el total del error a lo largo de toda la distribución, y no el error puntual como sucedía en el caso MSE) el cual vendrá dado por la Ecuación [3-71]::

$$\begin{cases} h_{MISE} = \arg\_\min\left\{E\left[\int \left(\hat{p}_h(x) - p(x)\right)^2 \cdot dx\right]\right\} \\ MISE\left(\hat{p}_h(x)\right) = \dfrac{1}{n \cdot h} \cdot \|\varphi\|_2^2 + \dfrac{h^4}{4} \cdot \mu_2(\varphi)^2 \cdot \left\|p^{''}\right\|_2^2 + o\left(\dfrac{1}{n \cdot h}\right) + o\left(h^4\right), \text{ cuando h} \to 0, \text{ n·h} \to \infty \\ \begin{cases} \mu_2(x) = \int s^2 \cdot \varphi(s) \cdot ds \\ \|\varphi\|_2^2 = \int \varphi^2(s) \cdot ds \end{cases} \end{cases}$$

Ecuación [3-71]

La Ecuación [3-71] puede simplificarse ignorándose los términos de orden superior, obteniendo así una aproximación a la integral del error cuadrático medio (AMISE), la cual viene descrita mediante la Ecuación [3-72]:

$$AMISE\left(\hat{p}_h(x)\right) = \frac{1}{n \cdot h} \cdot \|\varphi\|_2^2 + \frac{h^4}{4} \cdot \mu_2(\varphi)^2 \cdot \left\|p^{''}\right\|_2^2 \qquad \text{Ecuación [3-72]}$$

Basándonos en la aproximación AMISE se ha demostrado [3-32] que el parámetro de ancho de banda óptimo (Ecuación [3-73]) vendrá dado:

$$h_{optimo} = \left(\frac{\|\varphi\|_2^2}{\left\|p^{''}\right\|_2^2 \cdot \mu_2(\varphi)^2 \cdot n} \right)^{\frac{1}{5}} \approx n^{-\frac{1}{5}} \qquad \text{Ecuación [3-73]}$$

Donde el valor del parámetro óptimo será del orden de "$n^{-1/5}$" donde "n" se corresponde con el número de muestras con las cuales hemos estimado la distribución. La Ecuación [3-72] depende a su vez de "$\left\|p^{''}\right\|_2^2$" la cual es generalmente desconocida, siendo la convergencia media de AMISE para el parámetro óptimo de ancho de banda "h_{optimo}" la que se presenta en la Ecuación [3-74]:

$$AMISE\left(\hat{p}_{h_{optimo}}(x)\right) = \frac{5}{4} \cdot \left(\|\varphi\|_2^2\right)^{\frac{4}{5}} \cdot \left(\mu_2(\varphi)^2 \cdot \left\|p^{''}\right\|_2\right)^{\frac{2}{5}} \cdot n^{-\frac{4}{5}} \approx n^{-\frac{4}{5}} \qquad \text{Ecuación [3-74]}$$

No obstante, en el supuesto que conozcamos a priori la forma de la distribución a estimar, y que ésta sea una distribución normal "$p\left(\mu,\sigma^2\right)$" seremos capaces de evaluar la contribución "$\left\|p^{"}\right\|_2^2$", la cual vendrá dada por la siguiente expresión (ecuación [3-75]):

$$\left\|p^{"}\right\|_2^2 = \frac{3}{8\cdot\sqrt{\pi}}\sigma^{-5} \approx 0{,}212\cdot\sigma^{-5} \qquad \text{Ecuación [3-75]}$$

En consecuencia el valor del parámetro de ancho de banda óptimo (Ecuación [3-76]) será:

$$\begin{cases} h_{optimo} = \left(\dfrac{\|\varphi\|_2^2}{\left\|p^{"}\right\|_2^2\cdot\mu_2(\varphi)^2\cdot n}\right)^{\frac{1}{5}} \approx \left(\dfrac{4\cdot\hat{\sigma}^5}{3\cdot n}\right)^{\frac{1}{5}} \approx 1{,}06\cdot\hat{\sigma}\cdot n^{-\frac{1}{5}} \\ \hat{\sigma} = \sigma \end{cases} \qquad \text{Ecuación [3-76]}$$

Donde "$\hat{\sigma}$" se corresponde con la desviación estándar de las muestras usadas para la realización de dicha estimación. A esta aproximación del parámetro "h" se la conoce como aproximación a la distribución normal, o regla del pulgar de Silverman.

Para el caso de estimaciones de f.d.p multidimensionales [3-32, 3-33] el parámetro de ancho de banda óptimo "h_{optimo}" avaluado a partir de la aproximación a la MISE (Ecuación [3-77]) será la siguiente:

$$h_{optimo} \approx n^{-\frac{1}{(4+d)}} \qquad \text{Ecuación [3-77]}$$

Donde "*n*" representa el número de muestras usadas para la estimación de la distribución, mientras que "*d*" se refiere a la dimensión de la distribución a evaluar. Al igual que en el caso unidimensional, si conocemos a priori la forma de la distribución a estimar, y esta se rige experimentalmente por una distribución normal multidimensional los parámetros de ancho de banda vendrán dados por la Ecuación [3-78]:

$$\tilde{h}_j = \left(\frac{4}{d+2}\right)^{\frac{1}{(d+4)}}\cdot n^{\frac{1}{(d+4)}}\cdot\tilde{\sigma}_j \qquad \text{Ecuación [3-78]}$$

Donde "$\tilde{\sigma}_j$"se corresponde con la desviación estándar de las muestras usadas para la estimación de dicha dimensión "j".

Una vez descrita la metodología de las ventanas de *Parzen* basada en el uso de funciones de ventana normales, así como la forma de elección del parámetro de ancho de banda óptimo, pasaremos a presentar en los siguientes apartados la implementación estocástica de dicha técnica, orientada a su aplicación en sistemas de clasificación y/o reconocimiento de patrones.

3.3.3 IMPLEMENTACIÓN ESTOCÁSTICA DE LA METODOLOGÍA DE VENTANAS DE PARZEN MEDIANTE EL USO DEL KERNEL GAUSSIANO

A lo largo del presente subapartado presentaremos la implementación estocástica de la técnica de las ventanas de *Parzen* mediante el uso de una función de ventana Gaussiana (implementada mediante un bloque digital) para la estimación de la función probabilidad condicional caracter de una determinada categoría. El objetivo es su aplicación al campo de la clasificación y/o reconocimiento de patrones estadístico.

La función matemática genérica (Ecuación [3-79]) que deberá implementar cualquier bloque estocástico que pretenda estimar la función densidad de probabilidad condicional de una muestra "m" dimensional constituida por "*n*" medidas, de las cuales sólo "n_i" pertenecen a la clase "C_j" vendrá dada por la siguiente expresión:

$$P(1,..,m \mid C_j) = P(1,...,m) = \frac{1}{n_i}\sum_{i=1}^{n_i}\left(\left(\frac{1}{h_1\sqrt{2\pi}}e^{-\frac{(x-1_i)^2}{2h_1^2}}\right)\cdot(\ )\cdots\cdots\cdots(\ )\cdot\left(\frac{1}{h_m\sqrt{2\pi}}e^{-\frac{(z-m_i)^2}{2h_m^2}}\right)\right),1,....m \in \Re$$

Ecuación [3-79]

El circuito estocástico desarrollado para implementar la función matemática (Ecuación [3-79]) que estima una f.d.p condicional de una determinada clase, mediante la técnica de las ventanas de *Parzen* sobre un kernel Gaussiano, se presenta en la Figura 3-30.

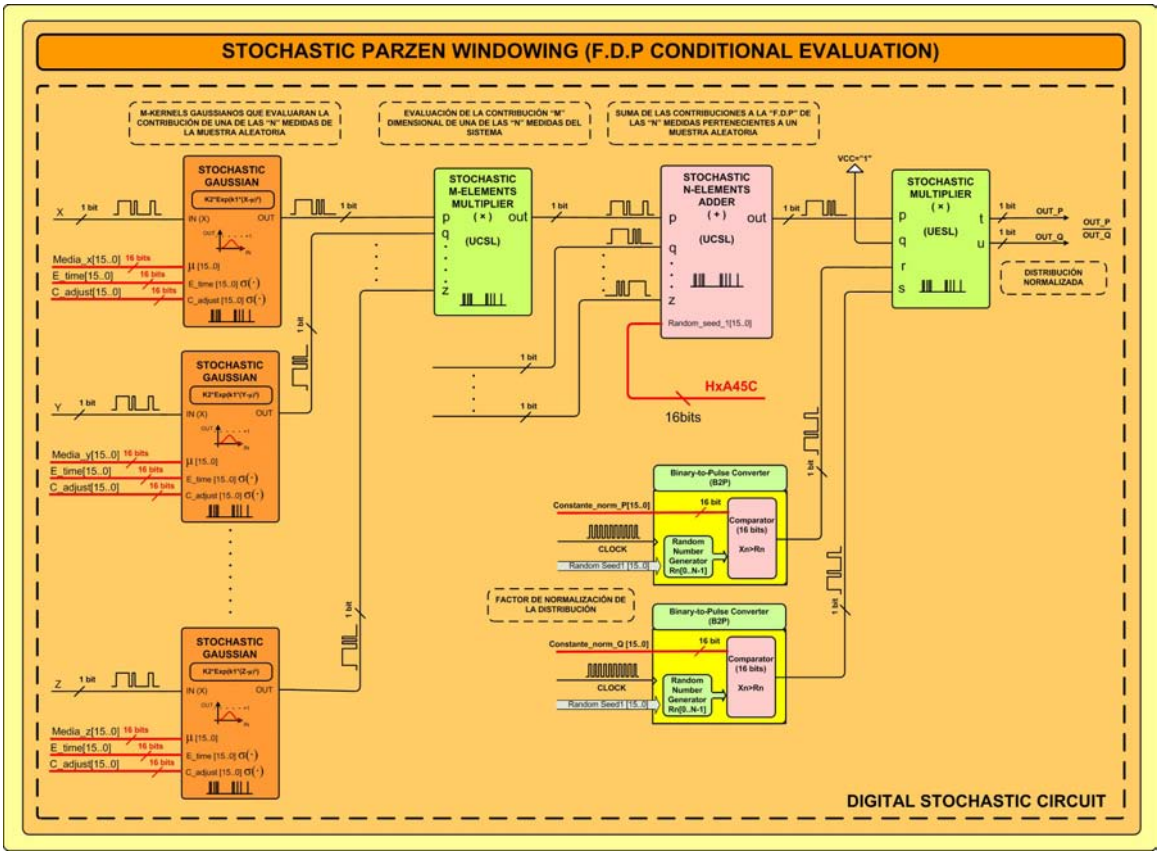

Figura 3-30: Bloque genérico para la evaluación de una f.d.p de una clase m-dimensional, obtenida a partir de n-contribuciones

Si nos fijamos detalladamente en la parte izquierda de la Figura 3-30, hallaremos un conjunto de bloques estocásticos "*f.d.p normal*" (UCSL) que implementan una función Gaussiana (bloque descrito en el apartado 2.2.3.9) no normalizada a la unidad, parametrizada mediante un valor medio "μ" y una desviación estándar "σ" (prefijada mediante una relación entre los parámetros "E_time" y "C_adjust") para cada uno de ellos. Mediante estos bloques implementaremos las funciones de ventana unidimensionales "$\varphi(\cdot)$" que nos servirán para obtener la contribución "*m*" dimensional de una determinada medida perteneciente a una muestra aleatoria. El valor medio "μ" de cada gaussiana se corresponderá con el valor de una de las coordenadas de las "*m*" dimensiones de una determinada medida, que contribuirá a una determinada clase "C_j". La desviación estándar "σ" la deberemos fijar en función del valor predicho por la metodología de las ventanas de

Parzen, por lo tanto la contribución obtenida mediante cada uno de estos bloques será la siguiente (Ecuación [3-80]):

$$\varphi(x) = e^{-\frac{(x-x_i)^2}{2h_x^2}} \qquad \text{Ecuación [3-80]}$$

Entonces para obtener la contribución de una medida "*m*" dimensional que contribuya a la estimación de la f.d.p de una determinada clase "C_j" deberemos proceder a multiplicar mediante un bloque "*multiplicador (UCSL)*" de "m" entradas (implementado internamente mediante una puerta **AND** de "m" entradas) las contribuciones unidimensionales de cada una de las "m" funciones de ventana "$\varphi(\cdot)$", al suponerse éstas independientes entre sí. La contribución "m" dimensional que obtendremos para cada medida será la siguiente (Ecuación [3-81]):

$$\varphi(1,..,m) = \varphi(1)\cdot\varphi(2)\cdot\varphi(3)\cdot\cdots\cdots\cdot\varphi(m) = e^{-\frac{(x-1_i)^2}{2h_1^2}} \cdot e^{-\frac{(x-2_i)^2}{2h_2^2}} \cdot e^{-\frac{(x-3_i)^2}{2h_3^2}} \cdots\cdots\cdot e^{-\frac{(x-m_i)^2}{2h_m^2}}$$

Ecuación [3-81]

Cabe remarcar que generalmente en nuestras aplicaciones asignaremos el mismo valor al parámetro de ancho de banda para todas las dimensiones. Los valores del parámetro de ancho de banda "*h*" se variarán sólo en caso que se disponga de información adicional que nos indicase dicha necesidad. Por lo tanto este hecho nos permitirá operar con funciones Gaussianas en la codificación (UCSL) no normalizadas. En consecuencia la Ecuación [3-81] se convertirá en la Ecuación [3-82]::

$$\varphi(1,..,m) = \varphi(1)\cdot\varphi(2)\cdot\varphi(3)\cdot\cdots\cdots\cdot\varphi(m) = e^{-\frac{(x-1_i)^2}{2h^2}} \cdot e^{-\frac{(x-2_i)^2}{2h^2}} \cdot e^{-\frac{(x-3_i)^2}{2h^2}} \cdots\cdots\cdot e^{-\frac{(x-m_i)^2}{2h^2}}$$

Ecuación [3-82]

Al ser el parámetro de ancho de banda "*h*" igual para cada una de las "*m*" funciones de ventana usadas para evaluar la contribución de una determinada medida, el factor de normalización para cada contribucion vendrá dado por la Ecuación [3-83]:

$$Factor\_normalización = \frac{1}{h^m \cdot (2\cdot\pi)^{\frac{m}{2}}} \qquad \text{Ecuación [3-83]}$$

Por lo tanto, la Ecuación [3-79] se podra reescribir de la siguiente forma (Ecuación [3-84]):

$$P(1,..,m \mid C_j) = P(1,...,m) = \left(\frac{1}{h^m \cdot (2\cdot\pi)^{\frac{m}{2}}} \right) \cdot \frac{1}{n_i} \sum_{i=1}^{n_i} \left(e^{-\frac{(x-1_i)^2}{2h^2}} \cdot e^{-\frac{(x-2_i)^2}{2h^2}} \cdot e^{-\frac{(x-3_i)^2}{2h^2}} \cdots\cdots e^{-\frac{(x-m_i)^2}{2h^2}} \right), 1,....m \in \Re$$

Ecuación [3-84]

La principal ventaja obtenida del uso de esta propiedad matemática es el poder aplicar directamente los bloques "*f.d.p normal*" en su codificación natural (UCSL), posibilitando a su vez el uso de un bloque *suma* (UCSL) (descrito en el apartado 2.2.3.2) para sumar de forma ponderada las contribuciones de las "n_i" medidas pertenecientes a una categoría "C_j". De esta forma obtendremos directamente la suma ponderada (Ecuación [3-85]) sin requerir de ningún circuito divisor (UESL) que hubiésemos necesitado en el caso de operar con bloques gaussianos normalizados (UESL). De esta manera se consumen así muchos menos recursos hardware (una cantidad menor de puertas lógicas) para la implementación de la multiplicación de las diferentes funciones de ventana, así como la posterior suma de las "n" contribuciones a dicha distribución.

$$out\_suma = \frac{1}{n_i} \sum_{i=1}^{n_i} \left(e^{-\frac{(x-1_i)^2}{2h^2}} \cdot e^{-\frac{(x-2_i)^2}{2h^2}} \cdot e^{-\frac{(x-3_i)^2}{2h^2}} \cdots\cdots e^{-\frac{(x-m_i)^2}{2h^2}} \right), 1,....m \in \Re \quad \text{Ecuación [3-85]}$$

Una vez sumadas las "n_i" contribuciones "m" dimensionales a una categoría dada procederemos a la normalización de la estimación de la f.d.p condicional. Para ello requeriremos de un bloque multiplicador (UESL), en el cual introduciremos en el numerador del primer dato a multiplicar la salida del bloque sumador ponderado, mientras la señal del numerador la fijaremos a nivel alto. A la vez, a la entrada del otro dato procederemos a fijar una razón entre dos señales estocásticas equivalentes al factor de normalización descrito en la Ecuación [3-83]. De esta forma obtendremos la estimación de la distribución condicional "$P(1,2,\ldots,m|C_j)$" de una clase "C_j" dada, normalizada a la unidad. Donde el resultado final obtenido se presenta mediante la siguiente expresión (Ecuación [3-86]):

$$OUT = \frac{P}{Q} = \frac{\frac{1}{n_i}\sum_{i=1}^{n_i}\left(e^{-\frac{(x-1_i)^2}{2h^2}} \cdot e^{-\frac{(x-2_i)^2}{2h^2}} \cdot e^{-\frac{(x-3_i)^2}{2h^2}} \cdots\cdots e^{-\frac{(x-m_i)^2}{2h^2}} \right)}{1} \cdot \left(\frac{factor\_normalización\_p}{factor\_normalización\_q} \right)$$

Ecuación [3-86]

Con lo cual la salida de este bloque siempre será una razón entre dos señales estocásticas codificadas en la forma (UESL), lo que nos obligará a usar siempre bloques de evaluación de la probabilidad a posteriori y/o de clasificación MSC en su codificación (UESL) (que son algo más complejos que los basados en la codificación UCSL).

A fin de evaluar la operatividad del circuito estocástico aquí propuesto procederemos a lo largo de los siguientes subapartados a estimar la f.d.p de una clase unidimensional y de otra bidimensional.

3.3.3.1 Estimación estocástica de una f.d.p unidimensional (1D) mediante la técnica de las ventanas de Parzen

En el presente subapartado procederemos a la estimación de la función densidad de probabilidad condicional para una clase "C_j" unidimensional (1D), a partir de cinco medidas experimentales pertenecientes a una misma clase obtenidas a partir de una única muestra aleatoria (que harán las funciones de conjunto de entrenamiento de nuestro sistema). De esta forma se verificará la correcta estimación de dicha f.d.p mediante el uso del bloque estocástico que se presenta en la Figura 3-31, el cual implementa la metodología de las ventanas de Parzen anteriormente descrita.

Para implementar este estimador de la f.d.p condicional para la clase "C_j" (Figura 3-31) usaremos cinco bloques "*f.d.p normal*" para implementar las funciones de ventana "$\varphi(\cdot)$" centradas en las posiciones "$\mu_i = \{x_1, x_2, x_3, x_4, x_5\}$". Éstas se corresponden con las pociciones de las cinco medidas unidimensionales obtenidas experimentalmente, mientras que el parámetro de ancho banda "h" deberíamos fijarlo a partir de la Ecuación [3-73]. Esta expresión establece una relación entre el número de medidas "n=5" que contribuyen a la

estimación de la f.d.p y dicho parámetro "$h \approx n^{-1/5} = 5^{-1/5} = 0{,}7248$" de ancho de banda. Para este caso en particular el valor óptimo del parámetro de ancho de banda se corresponde con una distribución muy poco vistosa (ya que la hemos evaluado y se corresponde prácticamente con una distribución puramente normal). A fin de presentar al lector un ejemplo mucho más vistoso hemos optado por la elección de un parámetro de ancho de banda diferente "$h = 0{,}05$". Este parámetro lo hemos fijado como constante para todas las funciones de ventana.

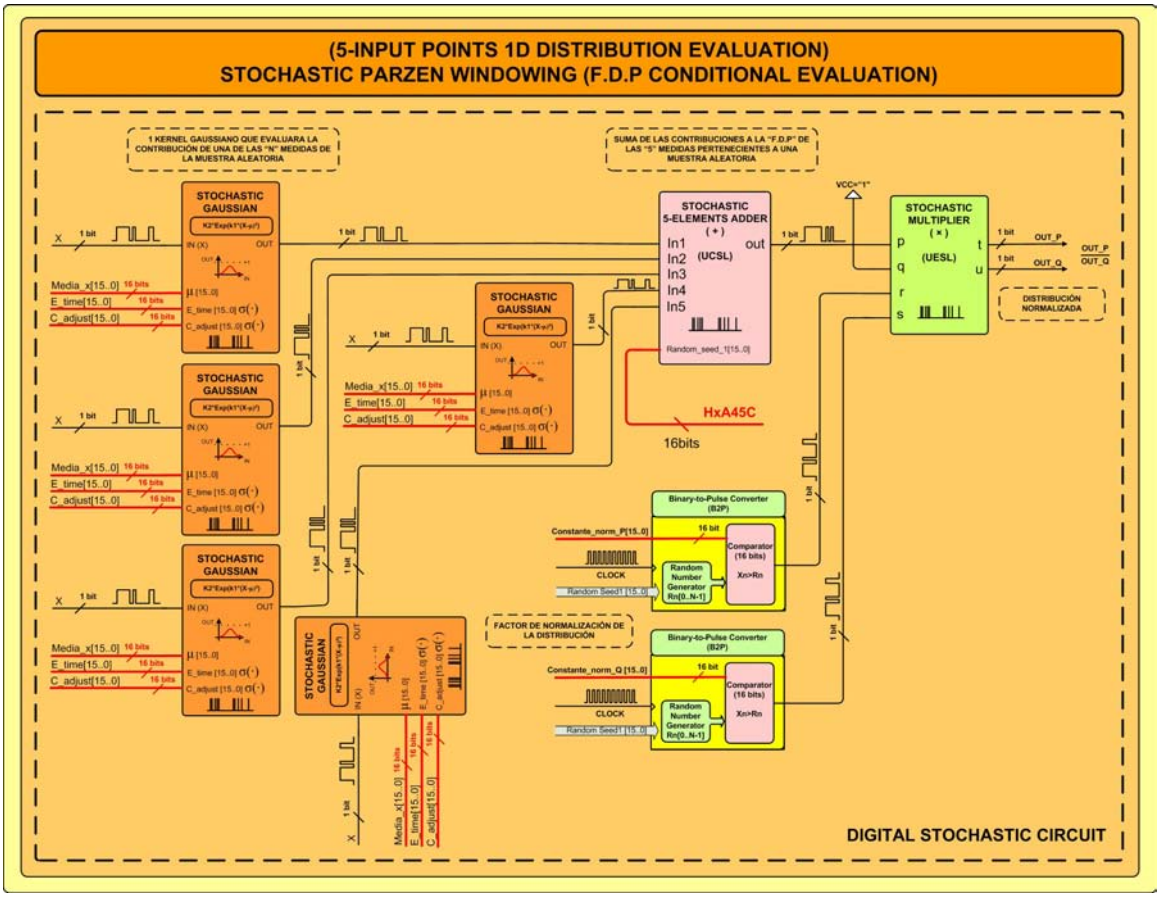

Figura 3-31: Bloque estimador de la f.d.p condicional de una clase C_j (1D) a partir de cinco datos de entrada

Siendo la parametrización de las diferentes funciones de ventana presentes en el sistema la presentada en la Tabla [3-14]:

Tabla 3-14: Parámetros de configuración de las diferentes funciones de ventana (caso 1D)				
Función de ventana (Medida 1)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho de banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
1638	0,025	0,0500	1101	655
Función de ventana (Medida 2)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho de banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
6552	0,1	0,0500	1101	655
Función de ventana (Medida 3)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho de banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
8190	0,1250	0,0500	1101	655
Función de ventana (Medida 4)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho de banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
9828	0,1500	0,0500	1101	655
Función de ventana (Medida 5)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho de banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
14742	0,2249	0,0500	1101	655

Con las diferentes funciones de ventana unidimensionales ya implementadas, procederemos a su suma directa mediante un bloque sumador (UCSL) de cinco entradas que evaluará de forma inherente la suma ponderada de las cinco señales. Al tratarse de un caso unidimensional no hemos requerido de ningún bloque que fusione las diferentes contribuciones unidimensionales de una determinada medida como sucederia en los casos multidimensionales.

Una vez sumadas las diferentes contribuciones de la estimación de la distribución de probabilidad condicional de la clase "C_j", nos restará proceder a normalizar la distribución multiplicándola (mediante un bloque multiplicador UESL) por un factor de normalización global de la distribución que obtendremos a partir de la Ecuación [3-83], fijando "d=1". Los valores de la razón de normalización a usar se detallan en la Tabla [3-15]:

Tabla 3-15: Parámetros de normalización de la estimación de la distribución (1D)		
Factor de normalización evaluado (P/Q)	Factor de normalización P (Numerador de la razón de normalización)	Factor de normalización Q (Denominador de la razón de normalización)
7,9789	65535	8214

Con todo el sistema dispuesto hemos procedido a evaluar la salida de dicho estimador estocástico, variando el valor de la señal estocástica de entrada "*x*" en el intervalo [0, +1], siendo los resultados obtenidos los que se presentan en la Tabla [3-16]:

Tabla 3-16: Medidas experimentales del estimador de la f.d.p condicional obtenidas a partir de 5 datos

Datos de entrada [Experimental]					Calores Teóricos	
IN [16bits]	Valor IN	Salida del bloque ventanas de Parzen P(x|Cj)_P	Salida del bloque ventanas de Parzen P(x|Cj)_Q	Estimación P(x|Cj) Razón (P/Q)	Estimación Teórica de P(x|Cj)	ERROR=ABS(TEO-EXP) Clase de salida
0	0,0000	17132	8225	2,0829	1,7125	0,3704
2048	0,0313	21692	8217	2,6399	2,5753	0,0645
4096	0,0625	30627	8228	3,7223	3,4942	0,2281
6144	0,0938	35470	8208	4,3214	4,4157	0,0943
8192	0,1250	39122	8207	4,7669	4,8448	0,0779
10240	0,1563	33488	8201	4,0834	4,4142	0,3308
12288	0,1875	30012	8225	3,6489	3,4923	0,1565
14336	0,2188	21856	8212	2,6615	2,5736	0,0878
16384	0,2500	13043	8187	1,5931	1,7109	0,1177
18432	0,2813	8580	8231	1,0424	0,9115	0,1309
20480	0,3125	2408	8199	0,2937	0,3541	0,0604
22528	0,3438	878	8218	0,1068	0,0958	0,0110
24576	0,3750	0	8214	0,0000	0,0177	0,0177
26624	0,4063	0	8214	0,0000	0,0022	0,0022
28672	0,4375	0	8214	0,0000	0,0002	0,0002
30720	0,4688	0	8214	0,0000	0,0000	0,0000
32768	0,5000	0	8214	0,0000	0,0000	0,0000
34816	0,5313	0	8214	0,0000	0,0000	0,0000
36864	0,5625	0	8214	0,0000	0,0000	0,0000
38912	0,5938	0	8214	0,0000	0,0000	0,0000
40960	0,6250	0	8214	0,0000	0,0000	0,0000
43008	0,6563	0	8214	0,0000	0,0000	0,0000
45056	0,6875	0	8214	0,0000	0,0000	0,0000
47104	0,7188	0	8214	0,0000	0,0000	0,0000
49152	0,7500	0	8214	0,0000	0,0000	0,0000
51200	0,7813	0	8214	0,0000	0,0000	0,0000
53248	0,8125	0	8214	0,0000	0,0000	0,0000
55296	0,8438	0	8214	0,0000	0,0000	0,0000
57344	0,8750	0	8214	0,0000	0,0000	0,0000
59392	0,9063	0	8214	0,0000	0,0000	0,0000
61440	0,9375	0	8214	0,0000	0,0000	0,0000
63488	0,9688	0	8214	0,0000	0,0000	0,0000
					Error Total:	**1,7506**

Si nos fijamos en la cuarta y quinta columnas de la Tabla [3-16] veremos como el valor de la estimación experimental y teórica (mediante MATLAB) son prácticamente idénticas, donde las pequeñas discrepancias existentes entre ambas (última columna de la Tabla [3-16] error) son básicamente debidas a la naturaleza aleatoria de la implementación. Por lo tanto, podemos concluir afirmando que el bloque estocástico desarrollado es completamente operativo.

A continuación procederemos a representar gráficamente (Figura 3-32) las diferentes contribuciones (correspondientes a cada una de las cinco funciones de ventana (líneas de colores sin símbolos)) a la estimación de la función distribución de probabilidad condicional teórica (línea verde). A la vez que representamos la distribución de probabilidad condicional obtenida experimentalmente para dicha clase (círculos rojos).

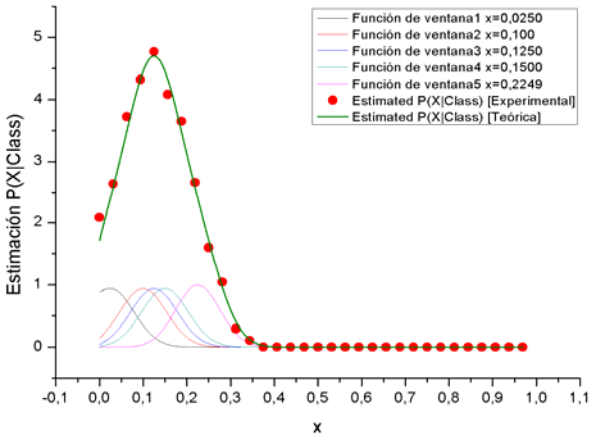

Figura 3-32: f.d.p condicional estimada experimentalmente vs f.d.p condicional estimada teóricamente a partir de 5 medidas 1D

Por lo tanto, como bien podemos apreciar en la Figura 3-32 la estimación experimental de la f.d.p condicional concuerda perfectamente con la predicha por la teoría.

3.3.3.2 Estimación estocástica de una f.d.p bidimensional (2D) mediante la técnica de las ventanas de Parzen

Una vez demostrada la operatividad del bloque desarrollado para la estimación de una función densidad de probabilidad condicional unidimensional (1D) procederemos en este subapartado a evaluar si dicho bloque calcula correctamente la estimación de una distribución de probabilidad bidimensional, pudiéndose extrapolar dichos resultados a un caso “m” dimensional.

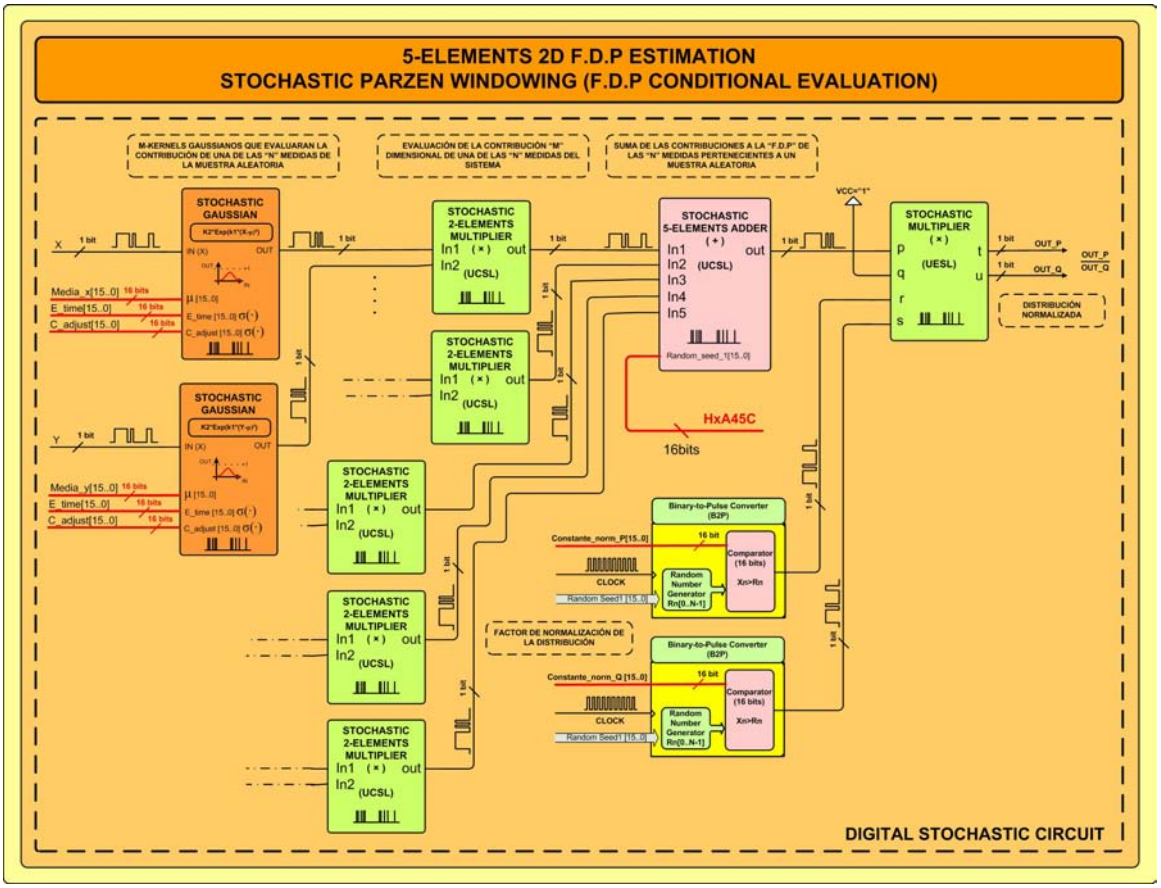

Figura 3-33: Bloque estimador de la f.d.p condicional de una clase C_j (2D) a partir de cinco datos de entrada

El circuito de la figura 3-33 muestra cómo se estima una función densidad de probabilidad condicional para una clase "C_j" bidimensional (2D) a partir de cinco medidas experimentales pertenecientes a dicha clase obtenidas de una única muestra aleatoria (que harán las funciones de conjunto de entrenamiento de nuestro sistema). Éste es idéntico al presentado para el caso unidimensional (figura 3-31) con la salvedad que ahora para la evaluación de la contribución de cada una de las medidas deberemos proceder primeramente a la multiplicación de las funciones de ventana unidimensionales "$\varphi(\cdot)$" centradas en las posiciónes "$\mu_i = \{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}, \{x_5, y_5\}\}$" correspondientes a las contribuciones de cada una de las dos dimensiones "{x, y}" para cada medida, tal y como se presenta en la Ecuación [3-87]:

$$\varphi(x,y)=\varphi(x)\cdot\varphi(y) \qquad \text{Ecuación [3-87]}$$

Por lo tanto, mediante la Ecuación [3-87] evaluaremos la aportación de cada una de las cinco medidas a la estimación de la f.d.p condicional de la clase "C_j". El parámetro de ancho banda "h" lo fijaremos arbitrariamente en este caso, haciéndolo constante para todas las funciones de ventana independientemente de la dimensión que representen. La parametrización de las diferentes funciones de ventana se presenta de forma detallada en la Tabla [3-17]:

Tabla 3-17: Parámetros de configuración de las diferentes funciones de ventana (caso 2D)

Función de ventana 2D (Medida 1)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
19656	0,2999	0,0028	2000	6554
μ=y1 [16bits]	Valor μ=y1			
1638	0,025			
Función de ventana 2D (Medida 2)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
22932	0,3499	0,0028	2000	6554
μ=y1 [16bits]	Valor μ=y1			
6552	0,1			
Función de ventana 2D (Medida 3)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
24570	0,3749	0,0028	2000	6554
μ=y1 [16bits]	Valor μ=y1			
8190	0,1250			
Función de ventana 2D (Medida 4)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
26208	0,3999	0,0028	2000	6554
μ=y1 [16bits]	Valor μ=y1			
9828	0,1500			
Función de ventana 2D (Medida 5)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
32768	0,5000	0,0028	2000	6554
μ=y1 [16bits]	Valor μ=y1			
14742	0,2249			

Finalmente para normalizar la distribución deberemos proceder a multiplicar la salida del bloque suma ponderada (UCSL) de las cinco contribuciones bidimensionales (correspondientes a cada una de las cinco muestras) por un factor de normalización global de la distribución que obtendremos a partir de la Ecuación [3-77] (fijando "d=2"), mediante

el uso de un bloque multiplicador UESL. Los valores de la razón de normalización usada son los presentados en la Tabla [3-18]:

Tabla 3-18: Parámetros de normalización de la distribución (2D)

Factor de normalización evaluado (P/Q)	Factor de normalización P (Numerador de la razón de normalización)	Factor de normalización Q (Denominador de la razón de normalización)
2030,0380	65535	323

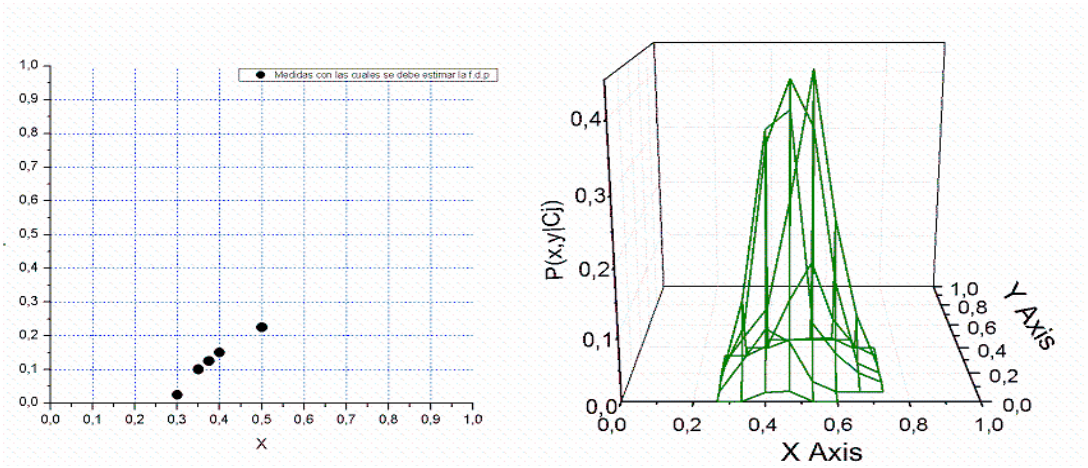

Figura 3-34: (Izquierda) coordenadas de los cinco puntos usados para la estimación de la función distribución de probabilidad condicional (2D), (Derecha) f.d.p 2D experimental estimada a partir de 5 medidas experimentales

Una vez dispuesto todo el sistema hemos procedido a evaluar la salida de dicho estimador estocástico variando los valores de las entradas estocásticas del sistema "{x, y]" ambas en el intervalo [0, +1], cubriendo así todo el espacio de representación del sistema. No obstante los resultados obtenidos para dicha distribución conforman una tabla de 16 filas por 16 columnas la cual se hace muy dificil de presentar de forma compacta en el presente documento. Por lo tanto, procederemos a presentar la distribución de probabilidad condicional obtenida en este caso en la Figura 3-34. Como podemos apreciar en dicha figura, mediante el bloque estocástico desarrollado hemos conseguido estimar la f.d.p condicional "P(x, y|C_j)"de una distribución 2D, demostrando así la capacidad de esta implementación estocástica para la estimación de f.d.p. de "m" dimensiones.

3.3.3.3 Clasificación estocástica mediante la técnica de comparación multidimensional MSC para la clasificación de dos distribuciones de probabilidad 1D evaluadas mediante la técnica de las ventanas de Parzen

Para concluir con este capítulo hemos procedido a evaluar la funcionalidad de la implementación estocástica de la técnica de las ventanas de *Parzen* en el campo de la clasificación y/o reconocimiento de patrones. Para ello en el presente subapartado, procederemos a evaluar la funcionalidad práctica de la implementación estocástica de la técnica de las ventanas de *Parzen* a la hora de obtener las funciones densidad de probabilidad condicional (a partir de una serie de datos de entrenamiento) necesarias para poder clasificar en base a la probabilidad a posteriori (mediante un bloque MSC) de todo un conjunto de datos de entrada al sistema.

Para ello inicialmente hemos procedido a tomar de forma aleatoria una muestra experimental de un sistema "X" unidimensional (1D), del cual conocemos a priori la existencia de dos clases (A y B) bien diferenciadas. El objetivo de dicha muestra (constituida por "n=10" medidas experimentales) es el servir como conjunto de entrenamiento del sistema para la evaluación de las probablidades condicionales de cada una de las clases "P(x|A) y P(x|B)". En dicha muestra hemos identificado cinco muestras pertenecientes a la clase A "$n_A = 5$" a la vez que cinco muestras pertenecientes a la clase B "$n_B = 5$". A partir de dicha información hemos evaluado las probabilidades a priori para las diferentes clases existentes en la muestra, mediante la Ecuación [3-88]:

$$\begin{cases} P(A) = \dfrac{n_A}{n} = \dfrac{5}{10} = 0{,}5 \\ P(B) = \dfrac{n_B}{n} = \dfrac{5}{10} = 0{,}5 \end{cases} \qquad \text{Ecuación [3-88]}$$

El conocimiento de los valores de estas probabilidades a priori es fundamental si se pretende implementar un sistema de clasificación basado en la probabilidad a posteriori de las clases "A y B". Una vez evaluadas las probabilidades a priori de las diferentes clases

debemos proceder a la estimación de las probabilidades condicionales para cada una de las dos distribuciones "*P(x|A)* y *P(x|B)*" a partir de los datos experimentales, estimando dichas *f.d.p* condicionales para cada una de las clases mediante una pareja de bloques estocásticos que implementan la metodologia de las ventanas de *Parzen*. Las funciones matemáticas a implementar por cada uno de los bloques estimadores se describen en la Ecuación [3-89]:

$$\begin{cases} \text{Clase A} \rightarrow P(x \mid A) = \left(\dfrac{1}{h \cdot (2 \cdot \pi)^{\frac{1}{2}}} \right) \cdot \dfrac{1}{n_A} \displaystyle\sum_{i=1}^{n_A} e^{-\frac{(x-\mu_A)^2}{2h^2}} = \left(\dfrac{1}{h \cdot (2 \cdot \pi)^{\frac{1}{2}}} \right) \cdot \dfrac{1}{5} \displaystyle\sum_{i=1}^{5} e^{-\frac{(x-\mu_A)^2}{2h^2}} \\ \text{Clase B} \rightarrow P(x \mid B) = \left(\dfrac{1}{h \cdot (2 \cdot \pi)^{\frac{1}{2}}} \right) \cdot \dfrac{1}{n_B} \displaystyle\sum_{i=1}^{n_B} e^{-\frac{(x-\mu_B)^2}{2h^2}} = \left(\dfrac{1}{h \cdot (2 \cdot \pi)^{\frac{1}{2}}} \right) \cdot \dfrac{1}{5} \displaystyle\sum_{i=1}^{5} e^{-\frac{(x-\mu_B)^2}{2h^2}} \end{cases}$$

Ecuación [3-89]

Una vez estimadas las distribuciones de probabilidades condicionales y evaluadas las probabilidades a priori de cada una de las clases, nos hallaremos en disposición de implementar el mecanismo de clasificación basado en la probabilidad a posteriori de ambas clases, el cual se implementará la comparación que se presenta en la Ecuación [3-90] mediante un bloque multidimensional MSC (UESL).

$$\begin{cases} \text{Clase A} = 1 \rightarrow \text{P}(\text{x} \mid \text{A}) \cdot P(A) > \text{P}(\text{x} \mid \text{B}) \cdot P(B) \\ \text{Clase B} = 1 \rightarrow \text{P}(\text{x} \mid \text{B}) \cdot P(B) > \text{P}(\text{x} \mid \text{A}) \cdot P(A) \end{cases}$$

Ecuación [3-90]

Una vez propuesto y descrito el sistema de clasificación, pasaremos a describir en detalle la implementación digital (Figura 3-35) del circuito estocástico desarrollado para tal fin.

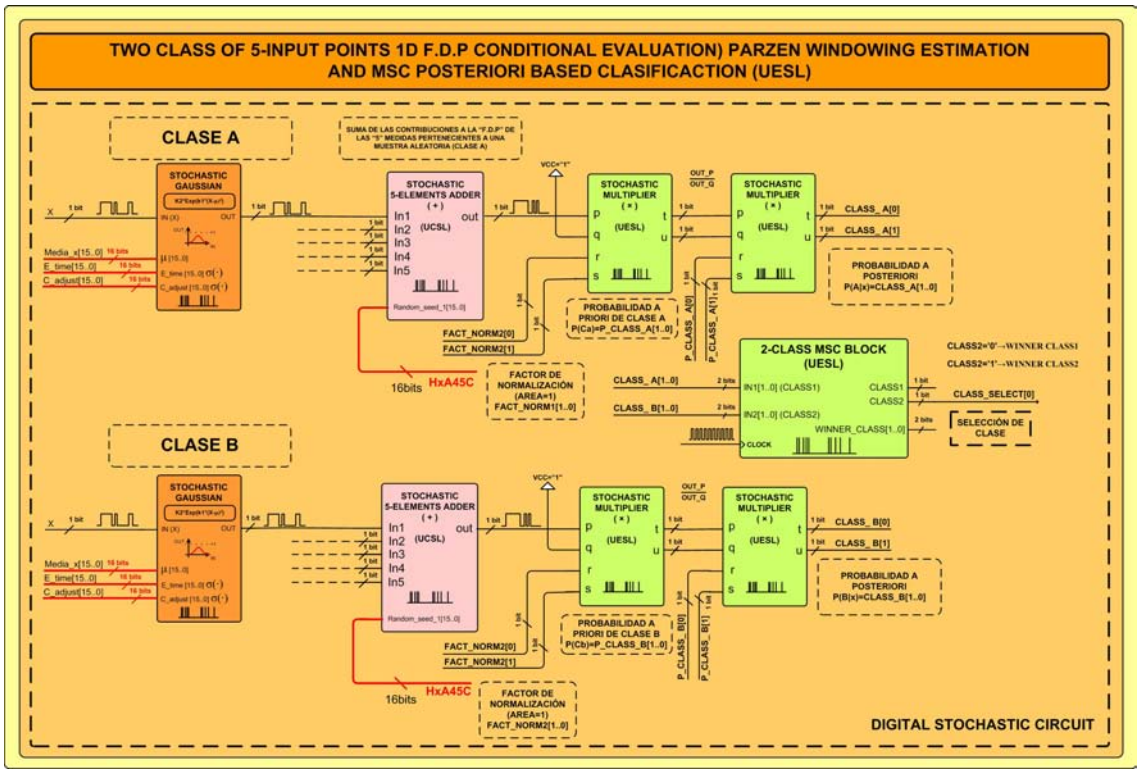

Figura 3-35: Bloque clasificador MSC (UESL) para un sistema 1D de 2 clases estimadas mediante la técnica de las ventanas de Parzen

En la parte izquierda de la Figura 3-35 hallamos dos conjuntos de cinco bloques "*f.d.p normal*", que se usan como funciones de ventana "$\varphi(\cdot)$" unidimensionales con las cuales evaluaremos la función distribución de probabilidad condicional de cada una de las categorías (A y B). Los valores del parámetro de ancho de banda "h" se han tomado iguales para todas las funciones de ventana del sistema. Donde el valor del parámetro de ancho banda lo hemos fijado en "h=0,05" con la finalidad de obtener unas estimaciones de f.d.p vistosas para el lector, cosa que no sucede si usamos el valor óptimo predicho por la teoría (Ecuación [3-73]) "$h \approx n^{-1/5} = 5^{-1/5} = 0{,}7248$". La parametrización de las diferentes funciones de ventana para cada clase se presenta de forma detallada en la Tabla [3-19]:

Tabla 3-19: Parámetros de configuración de las diferentes funciones de ventana de cada una de las dos clases presentes en la muestra experimental
CLASE A

Función de ventana (Medida 1)				
μ=x1 [16bits]	Valor μ=x1	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
1638	0,025	0,0500	1101	655
Función de ventana (Medida 2)				
μ=x2 [16bits]	Valor μ=x2	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
6552	0,1	0,0500	1101	655
Función de ventana (Medida 3)				
μ=x3 [16bits]	Valor μ=x3	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
8190	0,1250	0,0500	1101	655
Función de ventana (Medida 4)				
μ=x4 [16bits]	Valor μ=x4	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
9828	0,1500	0,0500	1101	655
Función de ventana (Medida 5)				
μ=x5 [16bits]	Valor μ=x5	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
14742	0,2249	0,0500	1101	655
CLASE B				
Función de ventana (Medida 1)				
μ=x6 [16bits]	Valor μ=x6	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
19656	0,2999	0,0500	1101	655
Función de ventana (Medida 2)				
μ=x7 [16bits]	Valor μ=x7	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
22932	0,3499	0,0500	1101	655
Función de ventana (Medida 3)				
μ=x8 [16bits]	Valor μ=x8	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
24570	0,3749	0,0500	1101	655
Función de ventana (Medida 4)				
μ=x9 [16bits]	Valor μ=x9	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
26208	0,3999	0,0500	1101	655
Función de ventana (Medida 5)				
μ=x10 [16bits]	Valor μ=x10	Parámetro de ancho banda "h" elegido σ (de la distribución):	[EvaluationTime] correspondiente a dicho parámetro de ancho de banda	[C_adjust] correspondiente a dicho parámetro de ancho de banda
32768	0,5000	0,0500	1101	655

Una vez parametrizadas las funciones de ventana para cada clase, procedemos mediante dos bloques sumadores (UCSL), a evaluar el valor medio de las cinco contribuciones para cada una de las distribuciones de probabilidad condicional "P(x|A) y P(x|B)". A continuación mediante sendos bloques multiplicadores (UESL) procederemos a normalizar las estimaciones de las f.d.p para cada categoria con los factores obtenidos de la Ecuación

[3-88]. Los valores de las razones de normalización obtenidos para ambas clases son las que se presentan en la Tabla [3-20]:

Tabla 3-20: Parámetros de normalización de la estimación de las distribuciones y valores de la probabilidad a priori para cada clase

CLASE A		
Factor de normalización evaluado (P/Q)	Factor de normalización P (Numerador de la razón de normalización)	Factor de normalización Q (Denominador de la razón de normalización)
7,9789	65535	8214
Probabilidad a posteriori de la clase A P(x|A) (P/Q)	Probabilidad a posteriori de la clase A P(x|A) (P) (Numerador de la razón de normalización)	Probabilidad a posteriori de la clase A P(x|A) (Q) (Denominador de la razón de normalización)
0,5	32768	65535
CLASE B		
Factor de normalización evaluado (P/Q)	Factor de normalización P (Numerador de la razón de normalización)	Factor de normalización Q (Denominador de la razón de normalización)
7,9789	65535	8214
Probabilidad a posteriori de la clase B P(x|B) (P/Q)	Probabilidad a posteriori de la clase A P(x|B) (P) (Numerador de la razón de normalización)	Probabilidad a posteriori de la clase A P(x|B) (Q) (Denominador de la razón de normalización)
0,5	32768	65535

Una vez normalizadas las distribuciones de probabilidad condicionales tan sólo restará multiplicarlas por el valor de la probabilidad a priori de cada clase. Finalmente la salida de estos bloques (el valor del producto de las probabilidades condicionales y a priori "$P(x|C_j)\cdot P(C_j)$" de cada clase) servirán de señales de entrada a un bloque MSC (UESL) con el cual discriminaremos cuál de las dos clases es la más probable en función de la posición del espacio unidimensional en la cual nos hallemos.

La salida digital de la clase 2 del bloque MSC nos indicará cuál de las dos clases es la más probable en cada posición "x" de nuestro espacio unidimensional (Clase2=1→Clase B, Clase2=0→Clase A).

Una vez finalizado el diseño y la parametrización de todo el sistema, hemos procedido a su implementación en una FPGA Cyclone II modelo EP2C20F484C7N del fabricante Altera Corp. Sobre dicha implementación hemos procedido a evaluar la salida del sistema de clasificador estocástico variando los valores de la señal estocástica de entrada "x" en el intervalo [0, +1], para cubrir así el espacio de representación del sistema. Los resultados experimentales obtenidos se presentan en la Tabla [3-21]:

Tabla 3-21: Medidas experimentales del clasificador MSC (UESL) para un sistema 1D de 2 clases estimadas mediante la técnica de las ventanas de Parzen

Datos de entrada [Experimental]		[Experimental]			Teórico
X [16bits]	Valor X	Distribución condicional de la clase A P(x|A)	Distribución condicional de la clase B P(x|B)	Salida del clasificador MSC (Clase)	P(A|X) Probabilidad a posteriori de la Clase A
0	0,0000	2,0829	0,0000	0 = Clase A	0,9982
2048	0,0313	2,6399	0,0000	0 = Clase A	0,9982
4096	0,0625	3,7223	0,0000	0 = Clase A	0,9995
6144	0,0938	4,3214	0,0000	0 = Clase A	0,9988
8192	0,1250	4,7669	0,0000	0 = Clase A	0,9961
10240	0,1563	4,0834	0,0421	0 = Clase A	0,9919
12288	0,1875	3,6489	0,1418	0 = Clase A	0,9656
14336	0,2188	2,6615	0,3895	0 = Clase A	0,8374
16384	0,2500	1,5931	1,0237	0 = Clase A	0,6001
18432	0,2813	1,0424	2,9437	1 = Clase B	0,265
20480	0,3125	0,2937	4,3124	1 = Clase B	0,0739
22528	0,3438	0,1068	5,1377	1 = Clase B	0,0207
24576	0,3750	0,0000	4,738	1 = Clase B	0,0019
26624	0,4063	0,0000	4,4688	1 = Clase B	0,0014
28672	0,4375	0,0000	2,6648	1 = Clase B	0,0001
30720	0,4688	0,0000	2,221	1 = Clase B	0,0001
32768	0,5000	0,0000	1,886	1 = Clase B	0,0001
34816	0,5313	0,0000	1,4784	1 = Clase B	0,0000
36864	0,5625	0,0000	0,9288	1 = Clase B	0,0000
38912	0,5938	0,0000	0,1956	1 = Clase B	0,0000
40960	0,6250	0,0000	0,1007	1 = Clase B	0,0000
43008	0,6563	0,0000	0,0000	1 = Clase B	0,0000
45056	0,6875	0,0000	0,0000	1 = Clase B	0,0000
47104	0,7188	0,0000	0,0000	1 = Clase B	0,0000
49152	0,7500	0,0000	0,0000	1 = Clase B	0,0000
51200	0,7813	0,0000	0,0000	1 = Clase B	0,0000
53248	0,8125	0,0000	0,0000	1 = Clase B	0,0000
55296	0,8438	0,0000	0,0000	1 = Clase B	0,0000
57344	0,8750	0,0000	0,0000	1 = Clase B	0,0000
59392	0,9063	0,0000	0,0000	1 = Clase B	0,0000
61440	0,9375	0,0000	0,0000	1 = Clase B	0,0000
63488	0,9688	0,0000	0,0000	1 = Clase B	0,0000

Si nos fijamos en las dos últimas columnas de la Tabla [3-21] podremos apreciar cómo los resultados de la clasificación concuerdan perfectamente con los predichos por la teoría (evaluados mediante la regla de Bayes en MATLAB).

Finalmente procederemos a representar gráficamente (Figura 3-36) las estimaciones de las funciones densidad de probabilidad condicional estimadas para cada una de las dos clases (Clase A línea negra, Clase B línea roja), a la vez que la salida digital de nuestro clasificador (línea azul con círculos, 0→Clase A, 1→Clase B) y la probabilidad teórica a posteriori de la clase A (línea verde con cuadrados).

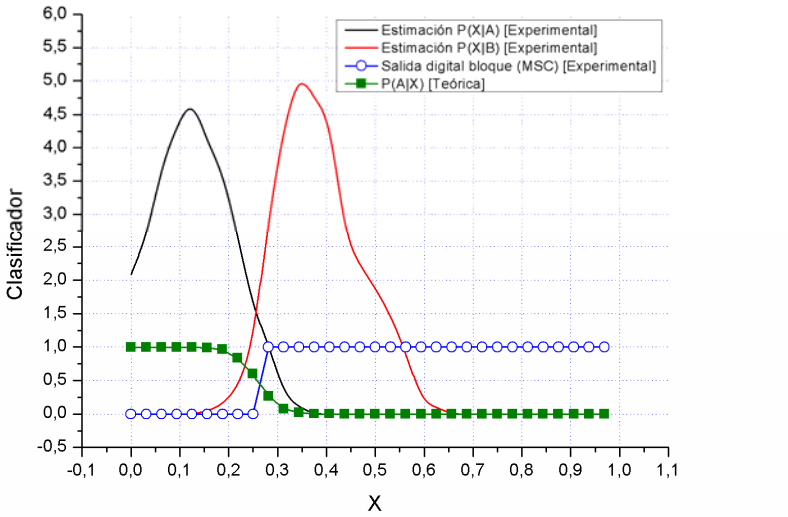

Figura 3-36: Resultados experimentales del clasificador MSC (UESL) para un sistema 1D de 2 clases estimadas mediante las ventanas de Parzen

Como bien puede apreciarse en la Figura 3-36, el clasificador desarrollado evalúa la clase más probable de forma totalmente correcta puesto que la salida del clasificador cambia de valor “0→1” en el punto en que la clase B se hace más probable que la clase A. Este punto coincide con el valor teórico para el cual la probabilidad a posteriori de la clase A “P(A|X)” toma un valor menor a 0,5.

Con este ejemplo finalizamos el capítulo referido al reconocimiento de patrones demostrando la operatividad de los circuitos estocásticos propuestos para la estimación de las f.d.p operando conjuntamente con los circuitos de clasificación estocástica (MSC) descritos al principio de este capítulo.

4. REDES NEURONALES

En este capítulo se aborda el desarrollo de redes neuronales de segunda generación mediante técnicas estocásticas, así como la implementación y entrenamiento de redes neuronales de tercera generación también conocidas como redes neuronales pulsantes. Para establecer el marco teórico sobre el cual se cimientan las diferentes aportaciones científicas realizadas se ha procedido a la redacción de una introducción general al campo de las redes neuronales. A su vez este capítulo se divide junto con la introducción en dos subapartados. En el primer subapartado se introducen las redes neuronales pulsantes así como las diversas aportaciones realizadas. Finalmente en el segundo subapartado se abordará una introducción a la implementación de redes neuronales de segunda generación mediante las diferentes codificaciones de lógica estocástica, para proseguir con la presentación de las aportaciones propias realizadas en dicho campo.

4.1 INTRODUCCIÓN GENERAL

Es objeto del presente subapartado realizar una introducción genérica al campo de las redes neuronales artificiales, incluyendo asimismo una descripción de la evolución histórica y de sus aplicaciones prácticas, a fin de establecer el marco sobre el cual poder presentar las diversas contribuciones realizadas en este campo.

4.1.1 FUNDAMENTOS BÁSICOS DE REDES NEURONALES ARTIFICIALES

El cerebro es el órgano donde se ubica el sistema nervioso central, el cual se compone de una intrincada red con miles de millones de células interconectadas llamadas “neuronas”. El estudio de las Redes Neuronales (NN) tiene por objeto entender cómo una gran colección de elementos “neuronas” interconectadas pueden dar lugar a metodologías de computación útiles.

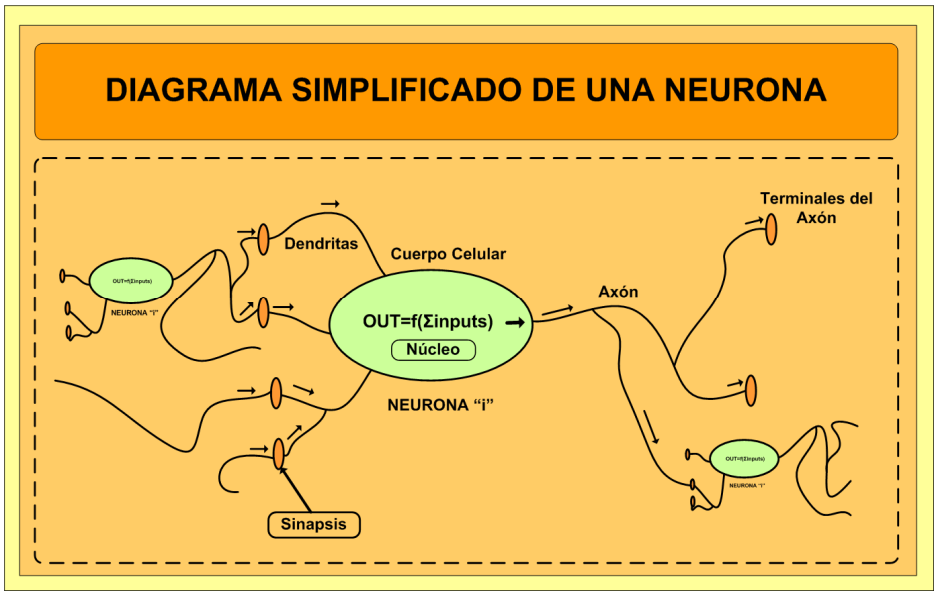

Figura 4-1: Redes neuronales

Una neurona real recibe multitud de señales en forma de pulsos eléctricos de otras neuronas. Estos pulsos son procesados de tal forma que una acumulación de ellos generará en la neurona recptora un nuevo pulso eléctrico que será transmitido a las demás neuronas interconectadas a ella, tal y como se muestra en la parte izquierda de la Figura 4-1. Así pues, cada neurona "procesa" los pulsos de entrada para generar pulsos de salida.

El campo de la ciencia que estudia las Redes Neuronales Artificiales (ANN) intenta capturar la esencia del procesamiento de la información en las neuronas reales, como se muestra en la parte superior derecha de la Figura 4-2. En ella se aprecia cómo la ratio de disparo de pulsos de salida de una determinada neurona viene regida por una magnitud escalar o "valor de actividad". Las conexiones direccionales entre neuronas determinan qué neuronas son las entradas de otras neuronas. Cada conexión tiene asociada un peso o configurador sináptico (pudiendo ser éstas excitatorias o inhibitorias), con lo cual la salida de una determinada neurona es una función de la suma ponderada (por los pesos) de las salidas de las neuronas que sirven de entrada a ésta. Dicha función recibe el nombre de función de transferencia de la neurona $f(\sum)$. En base a la función de transferencia podremos clasificar las neuronas de primera y segunda generación [4-37] en: lineales o no-lineales.

Atendiendo a la función de transferencia la salida en el caso de las neuronas *binarias* podrá ser solo “1” o “0”, en función de si el valor de la suma ponderada de las entradas sea mayor a un determinado nivel de disparo (neuronas de primera generación). Por otra parte, las neuronas no lineales dispondrán de una función de activación no lineal para limitar la amplitud de la neurona. Ejemplos de este tipo de neuronas pueden ser las neuronas sigmoidales, gaussianas, lineales a tramos, tangente hiperbólicas, y un largo etcétera según si su función de transferencia viene regida por alguna de las anteriores funciones. La salida de estas neuronas será siempre una magnitud real.

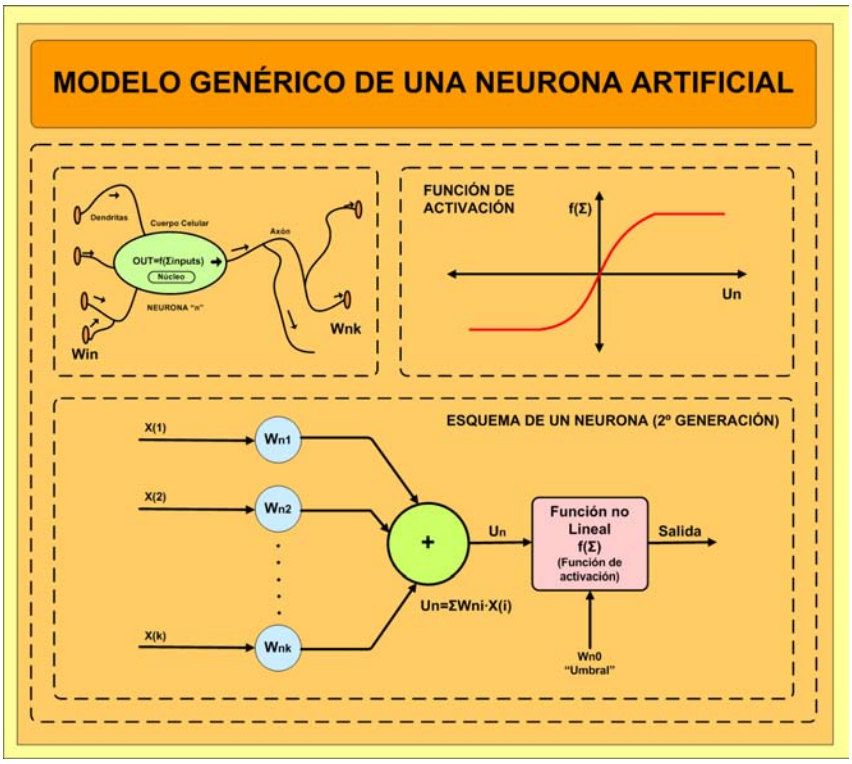

Figura 4-2: Modelo de una neurona artificial

En la Figura 4-2 se representa de forma esquemática una neurona artificial, cuyo modelo matemático básico se describe mediante la Ecuación [4-1]. En dicho modelo el valor del umbral de salida de la *función no lineal* o *función de activación* se ha codificado mediante el peso “ w_{n0} ” asociado a la señal de entrada “ $x(0)=1$ ”, a fin de simplificar la formulación matemática del modelo de neurona.

La neurona "n" $\rightarrow U_n = \sum_{j=1}^{k} w_{nj} \cdot x(j) \rightarrow salida = f(U_n - umbral)$, donde f(x) es una función de activación de la neurona.

Para simplificar la ecuación fijaremos : $\begin{cases} umbral = -w_{n0} \\ x(0) = 1 \end{cases} \rightarrow U_n = \sum_{j=0}^{k} w_{nj} \cdot x(j) \rightarrow salida = f(U_n)$

Ecuación [4-1]

Existen diversas funciones de activación aunque la mayoría de ellas como veremos más adelante son bioinspiradas, siendo las más representativas las que se presentan a continuación:

- La *función Signo o Umbral* (Ecuación [4-2]) permite la implementación de una neurona de primera generación o de MacCulloch-Pitts [4-1].

$$salida = f(U_n) = \begin{cases} 1 \rightarrow U_n > 0 \\ 0 \rightarrow U_n \leq 0 \end{cases} \qquad \text{Ecuación [4-2]}$$

- La *función Sigmoide* (Ecuación [4-3]) permite acotar el rango de salida de una neurona entre +b/2 y +b, a su vez fijaremos el valor de la pendiente de la función en el origen mediante el parámetro “a”, pudiendo así ajustar la función. En el caso de que el parámetro “a” sea lo suficientemente muy grande recuperaremos la función signo.

$$salida = f(U_n) = \frac{b}{1 + e^{-(a \cdot U_n)}} \qquad \text{Ecuación [4-3]}$$

- La función *Tangente hiperbólica* o *Tangente Sigmoidal* (Ecuación [4-4]), que se corresponde con una función equivalente a la anterior pero para sistemas bivaluados [-1,+1]

$$salida = f(U_n) = b \cdot \frac{1 + e^{+(a \cdot U_n)}}{1 + e^{-(a \cdot U_n)}} \qquad \text{Ecuación [4-4]}$$

- La *función Gaussiana* (Ecuación [4-5]) permite fundamentalmente implementar neuronas de base radial (RBF). Dichas neuronas se caracterizan por sólo contribuir con su salida al sistema para una determinada región del espacio de entradas. El parámetro

"k_1" se corresponde con la amplitud de la señal de salida, "k_2" es la media de la distribución y "k_3" es la desviación estándar de la distribución.

$$salida = f(U_n) = k_1 \cdot e^{\left(\frac{U_n - k_2}{k_3}\right)^2} \qquad \text{Ecuación [4-5]}$$

Donde en la práctica una red neuronal (NN) es un conjunto de neuronas artificiales interconectadas entre ellas mediante pesos de conexión "w_{ij}" obtenidos mediante reglas de aprendizaje que adaptan dicho valor "w_{ij}" a fin de optimizar el rendimiento de la red para la realización de una determinada tarea.

4.1.1.1 Arquitecturas de interconexión de neuronas artificiales

La interconexión de diferentes neuronas artificiales [4-38] da lugar a estructuras con capacidad para la implementación de funciones mucho más complejas e interesantes que no las que se pueden obtener con neuronas aisladas.

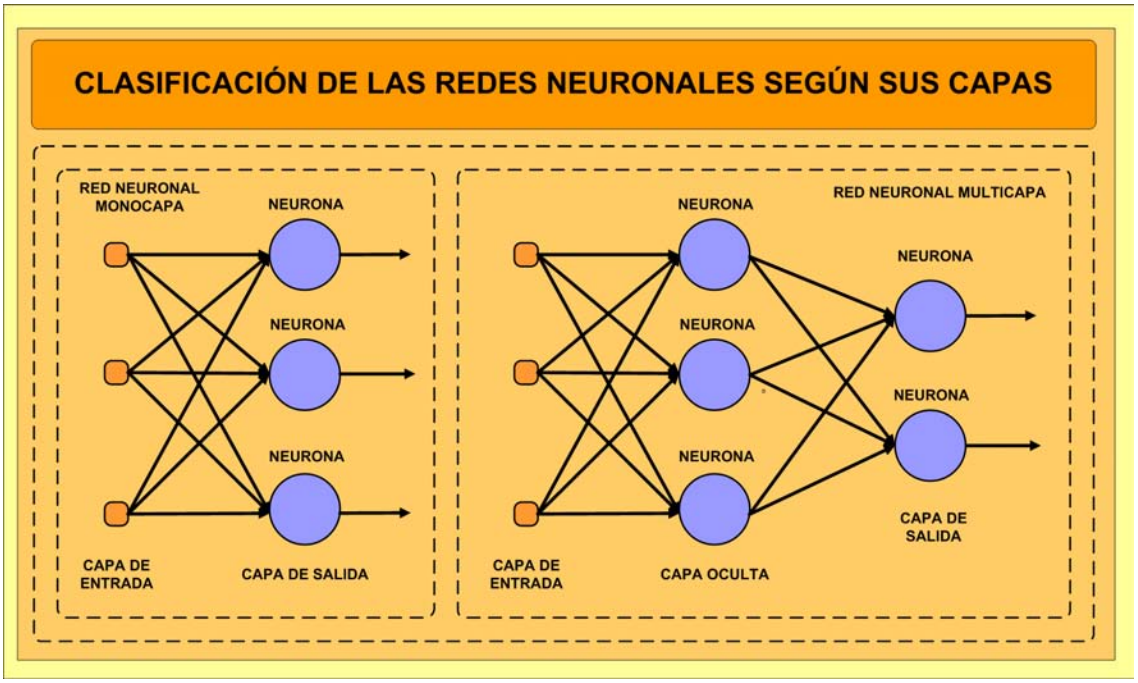

Figura 4-3: Arquitectura de capas de las redes neuronales

La distribución de neuronas dentro de una red se estructura mediante niveles o capas, con un número determinado de neuronas por capa. A su vez las redes neuronales con un número variable de capas se clasificarán como monocapa o multicapa (Figura 4-3). Las redes *monocapa* son las más sencillas que podemos hallar; formadas por una capa de entrada (que no realiza ninguna función o cálculo y que no se tiene en cuenta a la hora de conmutar como capa) y de una capa de salida. Su aplicación típica es la implementación de memorias asociativas. Por otro lado, las redes *multicapa* son, en esencia, una generalización de las anteriores, pero con un conjunto de capas intermedias entre la entrada y la salida, a las cuales se denomina *capas ocultas*.

En función del tipo de interconexión global, las redes podrán ser clasificadas como no recurrentes "*feed-forward*" o recurrentes "*feed-back*" (Figura 4-4). Las no recurrentes "*feed-forward*" son aquellas en las que la propagación de las señales sólo se produce en un sentido, no existiendo en ellas ninguna realimentación. Por lo tanto, este tipo de conexión no permitirá implementar sistemas con capacidad de memoria.

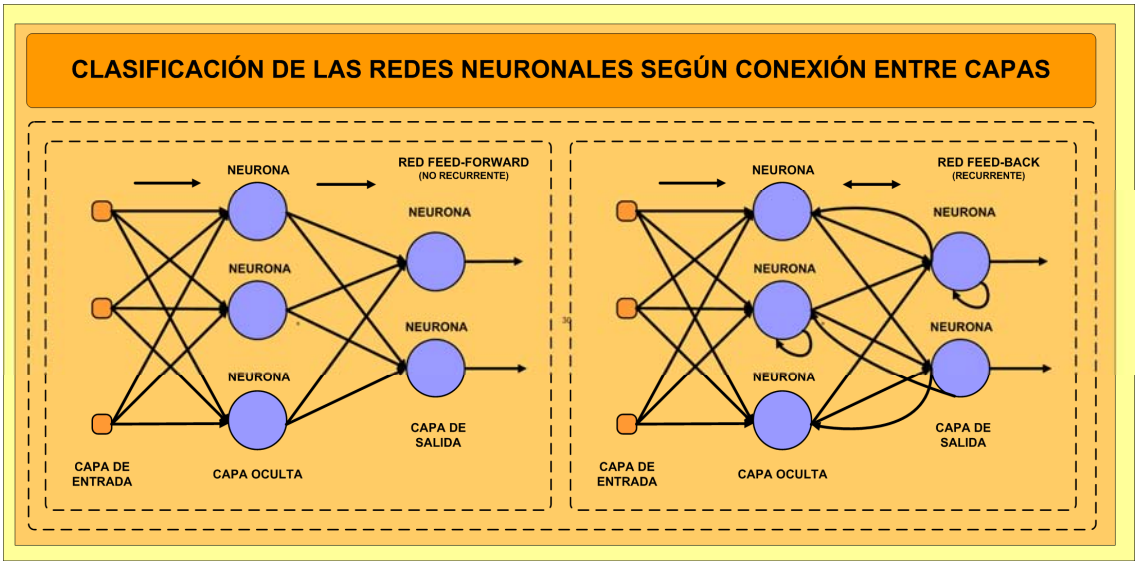

Figura 4-4: Conexiones entre capas en las redes neuronales

Las redes neuronales recurrentes se caracterizan por la existencia de lazos de realimentación, pudiendo ser éstos sobre una misma neurona, entre neuronas de una misma

capa o entre neuronas de diferentes capas. Este tipo de conexión permite la implementación de sistemas complejos no lineales [4-39].

El grado de conexión de una determinada neurona con respecto a la siguiente capa nos llevará a hablar de redes neuronales totalmente conectadas o de redes parcialmente conectadas. Las parcialmente conectadas, son aquellas en las cuales no se da la conexión total entre neuronas de diferentes capas al contrario de las totalmente conectadas.

4.1.2 EVOLUCIÓN HISTÓRICA DE LAS REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales (ANN) pueden parecer a disciplina de reciente aparición, aunque en realidad sus bases se instauraron con anterioridad al advenimiento de la era de los computadores. El primer trabajo realizado al respecto "*A Logical Calculus of Ideas Immanent in Nervous Activity*" data del año 1943 en el cual Warren McCulloch y Walter Pits presentaron el primer modelo simplificado de neurona biológica [4-1], que denominaron *neurona binaria* (Figura 4-5).

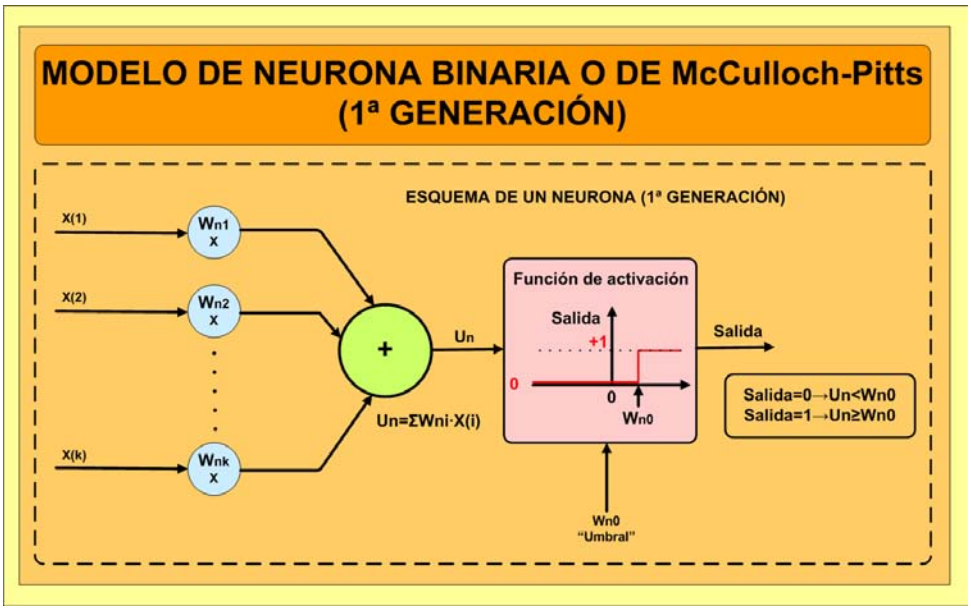

Figura 4-5: Neurona artificial binaria

Dicho modelo (Figura 4-5) dispone de una función de nivel como función de transferencia, donde el estado de salida será "activo" o "inactivo" en función de si la suma ponderada de

los estados de las neuronas interconectadas a ella es mayor o menor que un determinado nivel de disparo. Así pues, las interconexiones entre neuronas tienen un determinado sentido de conexión(de la neurona "i" a la neurona "j"), que tendrá asociado un peso específico w_{ij}. Por lo tanto, si la suma ponderada de los estados de conexión de las neuronas "i" respecto de una determinada neurona "j" excede un determinado nivel de disparo "T", el estado de salida de la neurona "j" se fijará como "activa" y en caso contrario como "inactiva". Mediante estos elementos se pueden implementar funciones binarias básicas como son: una puerta **AND** o una puerta **OR** de dos entradas.

En el año 1949 el psicólogo Donald Hebb (Canadá), padre de la biopsicología, publicó un libro titulado "The Organization of Behavior" [4-2] en el cual presentaban las reglas del aprendizaje Hebbiano. Más concretamente en el capítulo cuarto de dicho libro se proponia una regla de aprendizaje fisiológica, basada en la modificación de la sinapsis que se da en las neuronas reales. Además propuso que la conectividad del cerebro cambia conforme un organismo va adquiriendo nuevos conocimientos, ya que estos cambios van asociados a la creación de asociaciones neuronales. Mediante sus postulados sobre el aprendizaje, Hebb sugirió que la efectividad de una sinapsis variable entre dos neuronas se ve incrementada por la repetida activación de una neurona sobre la otra. Por lo tanto, desde el punto de vista neurofisiológico la regla planteada por Hebb, se corresponde con una regla variante temporal consistente en una interacción que aumenta la eficacia sináptica en función de la actividad pre y post sináptica entre dos neuronas. La regla de Hebb puede interpretarse como un tipo de aprendizaje no supervisado, en el que las conexiones entre dos neuronas se incrementan si ambas se activan al unísono.

A lo largo de la siguiente década, las redes neuronales (NN) dejaron de ser un campo únicamente influenciado por la neurociencia para dejar paso a aportaciones de campos tan dispares como la psicología y la ingeniería. En esta coyuntura, en el año 1956 los investigadores Rochester, Holland, Habait y Duda que trabajaban para IBM realizaron un trabajo científico [4-3] en el cual por primera vez se verificaba mediante simulaciones numéricas una teoría neuronal basada en el postulado de Hebb. Para poder implementar el sistema en las computadoras de esa época fue necesario aplicar una serie de limitaciones al

modelo de Hebb, como por ejemplo acotar el potencial sináptico, ya que en el modelo original propuesto por Hebb éste podía crecer sin límite.

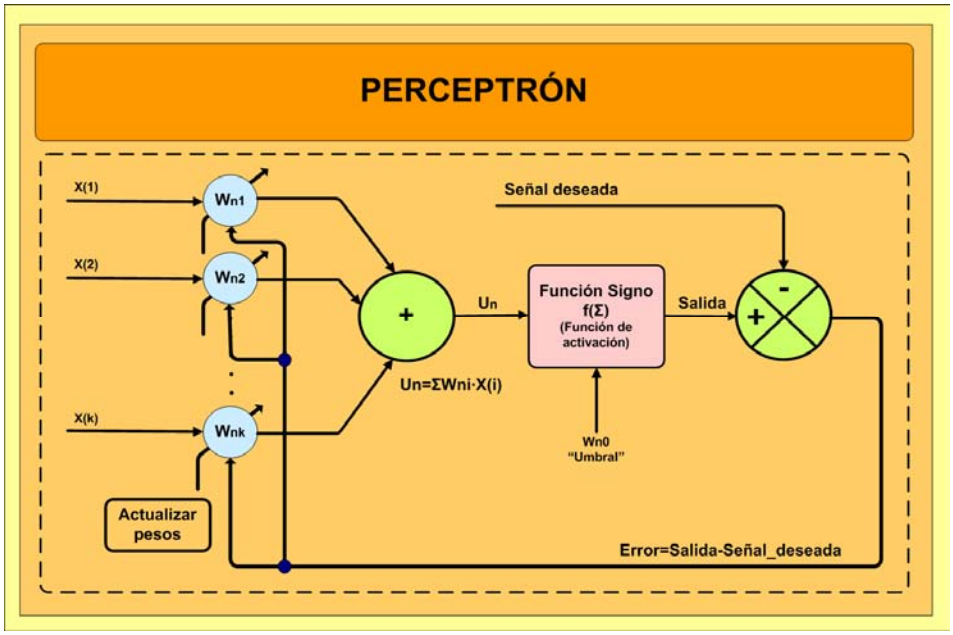

Figura 4-6: El Perceptrón

Al año siguiente Frank Rosenblatt (1957), uno de los padres de la computación, desarrolló y simuló el primer modelo de red neuronal "que emulaba un hipotético sistema nervioso" sobre un computador IBM704, el cual implementaba un sistema de aprendizaje basado en el ensayo y el error. A la vez, este modelo de neurona al que denominó "*Perceptron*" (Figura 4-6) representa una de las primeras implementaciones de un sistema conexionista. Este trabajo se llevó a cabo en el laboratorio aeronáutico de la Universidad de Cornell en Itaka (U.S.A). A partir del estudio de redes neuronales como el perceptrón, Rosenblatt realizó una publicación científica al año siguiente (1958) [4-4], en la cual se presentaban las leyes fundamentales de la organización para cualquier sistema de tratamiento de la información, tanto humano como artificial.

Dos años después (1960), los investigadores Bernard Widrow y Ted Hoff de la Universidad de Stanford (USA), desarrollaron un dispositivo electrónico llamado ADALINE "*ADAptative Linear Element*" [4-5] que usaba como mecanismo de aprendizaje el conocido como LMS "Least Mean Square" que trataba continuamente de minimizar el error de la

salida. Dicho sistema era básicamente una red neuronal monocapa que integraba un sistema adaptativo que podía aprender de forma más precisa y rápida que el resto de los perceptrones existentes en ese momento.

El año 1969 marcó el inició de un período de declive en el estudio de las redes neuronales artificiales, coincidiendo con la publicación del libro "*Perceptrons, An Introduction to Computational Geometry*" [4-6] de los investigadores Marvin Minsky y Seymour Papert del Instituto Tecnológico de Massachusetts (MIT) (U.S.A). En dicha obra evidenciaron las limitaciones del perceptrón de una capa para la resolución de todo un conjunto de problemas simples, concluyendo que un perceptrón solo permite resolver problemas linealmente separables, que a la práctica son una fracción muy pequeña de los problemas existentes. Su principal resultado fue demostrar matemáticamente la imposibilidad de implementar la función lógica **XOR** de dos entradas mediante un perceptrón. A su vez en dicho libro extrapolaron las limitaciones de los perceptrones de una capa a los multicapa, que por ese entonces no eran implementables en los sistemas de computación existente. Dicha publicación desalentó el interés de la comunidad científica por las redes neuronales artificiales (ANN) a lo largo de la década de 1970. Aun así, en esta década un grupo reducido de investigadores realizaron importantes avances en el campo de las redes neuronales artificiales. Este es el caso del neurofisiólogo James Anderson [4-7] de la Universidad de Brown (U.S.A) y el ingeniero Teuveo Kohonen [4-8] de la Universidad de Helsinki (Finlandia) que presentaron en el año 1972 de forma independiente un modelo muy similar de memoria asociativa, en el cual las neuronas se comportan como un sistema lineal que hace uso de la regla de aprendizaje de Hebb modificada formando en conjunto un asociador lineal.

Dos años después (1974) el investigador Paul Werbos de la Universidad de Harvard (U.S.A), en su tesis doctoral [4-9] presentó el método de "*Back-propagation*" para el aprendizaje de redes neuronales artificiales (ANN). Al cabo de los años el método de "*Back-propagation*" se ha convertido en el método más conocido y usado para el entrenamiento de redes neuronales. En esencia el método consiste en un algoritmo de aprendizaje supervisado que se encarga de minimizar un error cuadrático (comúnmente cuadrático) por medio de un gradiente descendiente, siendo la parte esencial del algoritmo

el cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

Al año siguiente (1975) Kunihiko Fukushima (Japón) presentó la primera red neuronal multicapa de la historia a la cual denominó *"Cognitron"* [4-10], utilizando como base una extensión del concepto de los perceptrones. El "*Cognitron"*, a diferencia del perceptrón, era capaz de aprender patrones sin ningún mecanismo externo que le indicase si el patrón reconocido era correcto o no. La aplicación que le dió K. Fukushima a su *Cognitron* fue el reconocimiento de caracteres escritos a mano "HandWritting".

A partir de la década de los 80 se produjo un resurgir del interés por parte de la comunidad científica en relación a las redes neuronales, ya que en ese período se desarrollaron toda una serie de trabajos científicos que han posibilitado la posterior evolución de las mismas. El primero de estos trabajos fue el del ingeniero biomédico Stephen Grossberg de la Universidad de Boston (1980), que realizó una publicación [4-11] en la que desarrollaba una teoría cognitiva y neuronal de cómo el cerebro puede aprender de forma estable y rápida, permitiendo así reconocer y recordar tanto objetos como eventos en un mundo cambiante. A esta teoría se la conoce como ART "*Adaptative Resonance Theory*". ART propone una solución al dilema de estabilidad-plasticidad, es decir, explica cómo un cerebro o máquina computacional puede aprender rápidamente acerca de nuevos objetos y eventos sin verse forzado a olvidar lo aprendido previamente y que es todavía "útil". Este trabajo ha sido muy significativo a la postre en el campo del reconocimiento de patrones.

Poco después Cristopher Von der Malsburg en el Instituto Max-Plank (1981) realizó un trabajo [4-12] en el cual se describía un nuevo tipo de red neuronal, basado en un modelo de neurona más realista, al cual llamó "*Spiking Neurons*" o neuronas pulsantes. Este nuevo modelo de neurona pretendía ayudar a resolver el problema de las interconexiones entre neuronas. En especial en dicho trabajo se indicaba que la señalización de los enlaces entre neuronas que codifican características de un mismo objeto, se correspondía con la sincronización del momento en el que las diferentes neuronas activan su salida "*La hipótesis de la sincronización*". Por lo tanto, las neuronas que codifican las características

pertenecientes a un determinado objeto se disparan en fase, permitiendo así la representación de múltiples objetos mediante la misma red.

Al año siguiente (1982) el físico John Hopfield del Instituto Tecnológico de California (U.S.A) publicó un trabajo clave [4-13] para el resurgimiento de las redes neuronales en el cual se desarrollaba la idea del uso de una función de energía para comprender la dinámica de una red neuronal recurrente con uniones sinápticas simétricas. En este trabajo Hopfield sólo permitía estados de salida bipolares a las neuronas (0 ó 1). En un trabajo posterior [4-14] amplió la función de energía planteada para otros sistemas, permitiendo así una salida real y continua para las neuronas. El principal uso de este tipo de redes ha sido como memorias y como instrumento para resolver problemas de optimización como el del viajante.

El mismo año (1982) Teuveo Kohonen publicó un trabajo pionero en el cual se presentaban los mapas auto-organizativos (SOM) "*Self Ornizative Maps*" [4-15], que se correspondían con un nuevo tipo de red neuronal entrenada mediante técnicas de aprendizaje no-supervisado que permitía obtener a su salida una representación discreta del espacio de muestras de entrenamiento de entrada, de muy baja dimensionalidad (generalmente 2D), al cual se denominó "*mapa*". Los (SOM) representan un tipo de red neuronal diferente ya que usan una función de vecindad para conservar las propiedades topológicas del espacio de entrada. Esto las hace útiles para visualizar en bajas dimensionalidades datos de muy alta dimensionalidad.

En un número especial de la revista del IEEE *Transactions on Systems, Man and Cybernetics* de (1983) dedicado a modelos de redes neuronales, se publicaron dos trabajos de gran importancia. El primero de K. Fukushima, S. Miyake y T. Ito (Japón) [4-16] presentaba una red neuronal llamada "Neocognitron", en cuyo diseño se combinaban ideas del campo de la filosofía, ingeniería y de la teoría de redes neuronales para implementar un sistema capaz de realizar tareas de reconocimiento de patrones. Este trabajo era la evolución del "*Cognitron"* descrito en un artículo anterior [4-10] por los mismos autores en el año 1975. El segundo trabajo realizado por A.G. Barto, R.S. Sutton y C.A. Anderson [4-17] (1983) abordaba el aprendizaje reforzado y sus aplicaciones en los sistemas de control.

Más concretamente en este trabajo se planteaba un nuevo tipo de aprendizaje supervisado que no requeriría del conocimiento del error total cometido por la red para poder aprender, a diferencia de lo que sucedía con el resto de sistemas de aprendizaje supervisado conocidos hasta el momento, no obstante, éste requería conocer el signo del error.

Tres años más tarde (1986) los investigadores G.E. Hinton y T.J. Sejnowski de la Universidad de Carnegie-Mellon (USA) presentaron un trabajo titulado "*Learning and relearning in Boltzmann machines*" [4-42] en el cual se describía una máquina de Boltzmann como mecanismo para obtener un sistema que permitiera simular la dinámica de un cerebro biológico. Ésta fue la primera red neuronal estocástica desarrollada. En esencia las máquinas de Boltzmann no son más que una red de Hopfield recurrente y discreta, con un mecanismo estocástico de actualización de las unidades ocultas. Las redes neuronales estocásticas serán abordadas en profundidad en siguientes apartados de esta tesis debido a su especial relevancia en relación a una serie de trabajos propios desarrollados.

El mismo año (1986) apareció un trabajo que junto al de Hopfield son los responsables de haber resucitado el interés científico por las redes neuronales. Dicho trabajo fue realizado por David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams [3-18], en el cual se desarrollaba el algoritmo de aprendizaje de retro propagación "*back-propagation*" para redes neuronales multicapa, presentando una serie de ejemplos prácticos donde se demostraba el potencial del método. A partir de este momento se produjo un aumento exponencial del número de trabajos científicos relacionados con las redes neuronales. Existe un gran número de aportaciones en lo referente a métodos de aprendizaje, arquitecturas de neuronas y redes, y sobre todo, se han realizado multitud de aplicaciones prácticas de las redes neuronales en todos los ámbitos de la ciencia e industria. No obstante existe toda una serie de trabajos posteriores que por su relevancia es necesario presentar.

Un trabajo a recalcar fue el realizado por los investigadores D.S. Broomhead y D. Lowe presentado en el año 1988 [4-28] en el que se introducía un método alternativo al perceptrón multicapa (MLP) para el ajuste de funciones no lineales. Mediante el diseño de una red neuronal específica constituida por tres capas se usan unas neuronas con un elemento no lineal consistente en una función de base radial (RBF) en la capa oculta.

Tres años más tarde (1991), W. Maass, G. Schnitger y E.D. Sontag, trabajando en la Universidad Técnica de Graz (Austria), presentaron un trabajo [4-29] (posteriormente presentaron una extensión de éste [4-30]) en el cual se planteaban las limitaciones de las neuronas binarias para implementar computaciones tanto con magnitudes analógicas como digitales, a la vez que se presentaba una solución basada en la adición de una función de activación no lineal. Pudiendo así implementar funciones digitales con muchas menos neuronas que no mediante el uso de neuronas binarias [4-1] (neuronas de primera generación), ya que estas nuevas se parecían mucho más a las biológicas que no las primeras [4-31]. Esto dio lugar a lo que posteriormente se conoció como las neuronas artificiales de segunda generación.

En esta época, se inició un ferviente interés en la comunidad científica por el estudio de las *máquinas de soporte vectorial* (SVM) "*Support Vector Machines*", las cuales, se basan en un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik [4-32] y su equipo en los laboratorios AT&T, siendo la utilidad básica de estos sistemas la resolución de problemas de clasificación y regresión.

A su vez, debemos destacar los trabajos referentes al desarrollo de las neuronas pulsantes "*Spiking Neurons*" [4-36, 4-38], que son un tipo de neurona con un mecanismo de operación muy parecido al de las biológicas, y que actualmente se conocen como la tercera generación de redes neuronales. Este tipo de neuronas se describen con mayor lujo de detalles a lo largo de los siguientes subapartados, al ser uno de los campos en los que hemos realizado algunas contribuciones científicas.

Son de reseñar los trabajos realizados por R.J. Williams y D. Zisper [4-39] en la consecución de algoritmos de aprendizaje que permitieran la implementación de redes neuronales recurrentes "*Feedback Networks*", especialmente útiles para estudiar la dinámica de sistemas no lineales.

Para concluir con esta introducción histórica, debe destacarse el desarrollo de las redes neuronales de retardo temporal (TDNN) "*Time Delay Neural Network*" por parte de K. Lang y G.E. Hinton (1988) [4-40] y de Alexander Waibel (1989) [4-41]. Estas redes no son más que un tipo de red multicapa realimentada cuyas neuronas ocultas y de salida son

replicadas a lo largo del tiempo, es decir, las salidas de una capa se almacenan en varios instantes de tiempo tras los cuales sirven para alimentar la siguiente capa.

4.1.3 FORTALEZAS Y APLICACIONES DE LAS REDES NEURONALES ARTIFICIALES

Como bien hemos podido apreciar en el anterior apartado, el campo de la ciencia que estudia las redes neuronales artificiales (ANN) es completamente multidisciplinar y en el cual se han realizado aportes significativos por parte de físicos, ingenieros, psicólogos, médicos y matemáticos. El interés por las redes neuronales radica en el potencial de computación que ofrecen, al tratarse de estructuras reales de cálculo distribuido en paralelo. Mediante estas estructuras de procesamiento se puede afrontar la resolución de problemas no lineales muy complejos de resolver de forma computacionalmente óptima, frente a la resolución basada en la computación secuencial clásica.

Debemos entender las redes neuronales como sistemas no lineales, en los cuales la asociación de un conjunto de estas neuronas (no-lineales) generará a su vez un sistema no-lineal. Esta propiedad es la que permite que se puedan emular sistemas reales no modelables mediante sistemas lineales. Otra característica reseñable de las redes neuronales es su tolerancia a fallos, ya que al tratarse de sistemas distribuidos, permiten que en el caso que algunos de los elementos (neuronas) fallen, el resultado de las salidas del sistema no se vea significativamente alterado. Este hecho las hace especialmente atractivas para la implementación de sistemas críticos de control/gestión, frente a las soluciones computacionales clásicas que ante un error del sistema dejan de operar.

Una las cualidades de las redes neuronales es su gran adaptabilidad, puesto que tiene la capacidad de modificar los parámetros (pesos) en función de los cambios que se produzcan en el entorno del sistema (ruido en las señales, cambio en las entradas, etc.…). No obstante la capacidad de adaptación del sistema es limitada, ya que una gran adaptabilidad implicaría en la práctica una inestabilidad de ésta ante pequeñas variaciones de las entradas. Éste se conoce como el dilema *de la estabilidad y plasticidad* del aprendizaje descrito en un trabajo [4-11] de Stephen Grossberg en el año (1980).

Otra de las ventajas de las redes neuronales es su capacidad de relacionar conjuntos de datos, permitiendo así establecer relaciones no lineales entre ellos. Si las comparamos con los métodos estadísticos clásicos que realizan la misma función, éstos no tienen porque cumplir condiciones de linealidad, estacionalidad [4-19] que sí requieren los primeros.

Las redes neuronales son elementos fácilmente integrables en silicio mediante técnicas VLSI [4-20] o en su defecto mediante dispositivos de lógica programable (FPGA o CPLD) [4-21], aunque mediante estas últimas sólo se podrán realizar implementaciones digitales.

Este conjunto de cualidades las hacen especialmente adecuadas para la resolución de determinados tipos de problemas como pueden ser los relacionados con la clasificación y el modelado de sistemas. Más concretamente en la resolución de problemas donde se dispone de un número elevado de datos de entrenamiento y en los cuales se puede asumir una cierta tolerancia a la imprecisión, pero que a su vez no es viable aplicar sobre ellos reglas rápidas y robustas de clasificación. Cabe remarcar que una red neuronal simple constituida sólo por una capa de entrada, una oculta y una de salida permite implementar consistentes estimadores estadísticos para la regresión de funciones desconocidas, o la implementación de clasificadores binarios óptimos.

En el campo del procesado de señal la aplicación de los sistemas basados en redes neuronales es muy amplia. Este es el caso del reconocimiento de patrones en imágenes [4-22] y voz [4-23]. En el campo del modelado hallamos aplicaciones de control donde se modela la respuesta del sistema para poder implementar un control predictivo [4-24], o en la implementación de sistemas de cancelación del ruido ambiente, instalado por ejemplo en los teléfonos móviles. A priori todas estas tareas son muy sencillas para los humanos pero extremadamente complejas para un sistema informático.

En el campo de la medicina se han desarrollado sistemas de clasificación destinados a resolver problemas de diagnóstico, como son algunos casos de cardiopatías que pueden ser diagnosticadas a partir de los pulsos cardíacos EVG [4-25], para la detección de tumores cancerígenos mediante imágenes médicas [4-26] entre otras muchas aplicaciones. Otro uso de las redes neuronales en medicina es el modelado de determinadas señales biomédicas (ECG, EMG, EEG) para la modelización o caracterización de diversas patologías [4-27]. A

su vez las redes neuronales se han aplicado con éxito en campos tan diversos como la economía, la industria farmacéutica y los videojuegos.

4.1.4 LAS REDES NEURONALES ARTIFICIALES EN EL CAMPO DEL RECONOCIMIENTO DE PATRONES

Las redes neuronales se pueden visualizar como sistemas masivos de computación en paralelo constituidos por un conjunto extremadamente grande de procesadores de información muy simples y con un gran número de interconexiones entre ellos.

Las redes neuronales más comúnmente usadas para las tareas de clasificación de patrones [4-47] son las redes feed-forward (sin capacidad de memoria, al no disponer de realimentación entre neuronas), entre las que se incluyen las redes perceptrón multicapa y las redes basadas en funciones de base radial (RBF). Estas redes se estructuran en capas y disponen de conexiones unidimensionales entre ellas. Otro tipo de red neuronal muy usada son los mapas auto organizativos (SOM), también conocidos como redes de Kohonen [4-48]. Éstas son principalmente utilizadas en procesos de agrupación de datos "*data-clustering*" y mapeado de características "*feature mapping*". El proceso de aprendizaje implicará la actualización de la arquitectura de la red así como de los pesos de las conexiones entre capas, de esta forma una red puede proceder a realizar de forma eficiente una determinada tarea de clasificación o el mapeado de categorías.

En los últimos tiempos han ganado mucha popularidad los métodos basados en redes neuronales para resolver problemas de clasificación. No obstante, hay que tener muy presente que aunque los principios subyacentes sobre los que se basan las redes neuronales son totalmente diferentes a las de los sistemas estadísticos clásicos de reconocimiento de patrones (presentados en el capítulo anterior), la mayoría de ellos son implícitamente equivalentes o muy similares a los métodos estadísticos clásicos de reconocimiento de patrones. En la Tabla 4-1 se presenta de forma resumida la tabla de equivalencias entre ambas aproximaciones.

Tabla 4-1: Relación de semejanzas entre métodos estadísticos y basados en redes neuronales

Reconocimiento de patrones estadístico	Redes Neuronales Artificiales
Función Lineal Discriminante	Perceptrón de una sola capa
Análisis del Componente Principal (PCA)	Redes auto asociativas, y varias redes PCA
Estimación de la probabilidad a posteriori	Perceptrón multicapa (MLP)
Análisis de discriminante no lineal	Perceptrón multicapa (MLP)
Clasificador basado en la estimación de la densidad de probabilidad mediante la ventana de Parzen	Redes neuronales de base radial (RBF)
Regla editada de k-NN	Mapas de Kohonen (LVQ)

4.2 LAS REDES NEURONALES PULSANTES

A lo largo de este apartado se presentan los fundamentos biológicos que conforman la base teórica que sustenta la implementación de las neuronas pulsantes "*spiking neurons*", elementos básicos de la tercera generación de redes neuronales [4-37], sobre los que procederemos a presentar las diversas aportaciones propias realizadas en este campo.

4.2.1 INTRODUCCIÓN

Las neuronas pulsantes o "*spiking neurons*" se basan en un mecanismo de generación de pulsos de salida a partir de un conjunto de pulsos de entrada. Cuando un pulso presináptico llega a una determinada conexión (sinapsis), ésta a su vez libera un neurotransmisor que modifica el estado interno de la neurona. Dicho estado interno viene descrito por el potencial de membrana de la neurona, de tal manera que en el momento en el que el valor del estado supera un cierto nivel umbral (θ) se generará un pulso de salida, después del cual el estado pasa a un valor fundamental. Este valor del potencial de reposo vendrá precedido por un estado llamado refractario en el cual la neurona será prácticamente insensible a estímulos. Por ello, el valor de la contribución de un pulso presináptico vendrá determinado por el tipo y la eficacia de la sinapsis, todo modelado mediante un peso (Figura. 4-7).

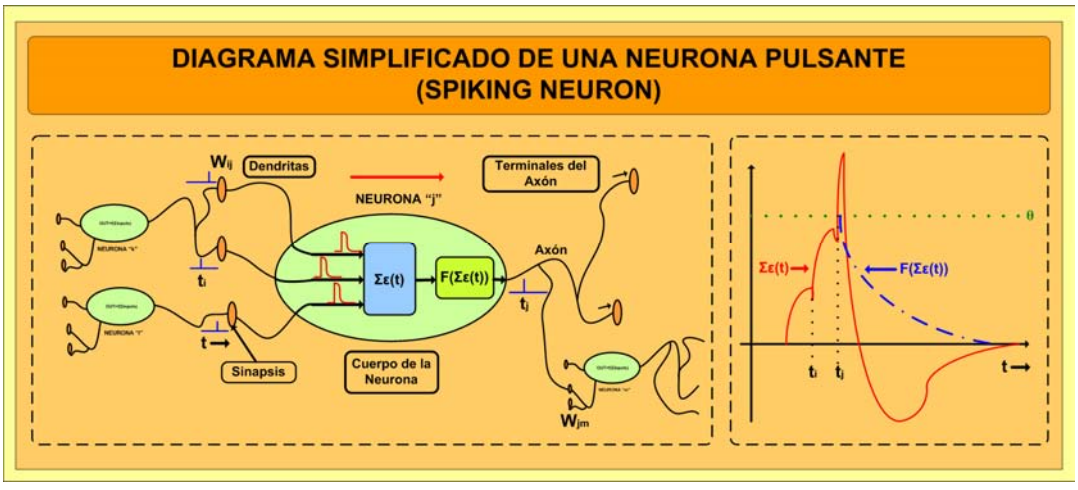

Figura 4-7: Neurona pulsante

Una red neuronal pulsante asíncrona es un elemento que modela esta cadena de eventos, en función de un conjunto de pulsos temporales en las señales de entrada (descritos matemáticamente como $\{t_i^1, t_j^1, ..., t_k^1\}$, donde "$t_i^n$" hace referencia al instante en que el pulso n-enésimo ha llegado a la neurona i-ésima) obteniendo así una respuesta temporal determinada por la conectividad de la red. La conectividad y el valor de los pesos de las conexiones determinan la respuesta temporal del pulso de salida para un patrón temporal de entradas, posibilitando así realizar determinados tipos de computaciones [4-35, 4-36]. Al contrario de lo que ocurre con las neuronas tradicionales, en las pulsantes no todos los pulsos o señales de entrada son iguales, puesto que su importancia está relacionada con el instante temporal en el que ésta llega a la red. Este conjunto de propiedades de las neuronas pulsantes han demostrado una capacidad de computación mucho más refinada o avanzada que la que pueden ofrecer las redes neuronales clásicas de primera o segunda generación [4-37]. Otra de las propiedades importantes de las redes neuronales pulsantes estriba en el hecho que el patrón de entrada se define mediante los instantes en que se reciben pulsos, con lo cual, la computación en una determinada neurona debe entenderse como invariante en la escala temporal. Dicho de otra forma, la codificación de las señales de entrada y salida está en relación a los intervalos entre pulsos, más que en los valores absolutos de los tiempos en los cuales se generan estas señales.

4.2.2 DESCRIPCIÓN DE LAS NEURONAS BIOLÓGICAS

El sistema nervioso de todos los seres vivos, y en especial el cerebro humano, está constituido por un vasto número de elementos básicos de procesamiento llamados *neuronas* que se hallan altamente interconectados entre ellos formando una intrincada malla en donde conviven conjuntamente con las células de la glía (encargadas de dar soporte energético a las neuronas). Cada neurona está constituida a su vez por tres partes bien diferenciadas llamadas dendritas, soma (cuerpo celular) y axón (figura 4-8). Las dendritas hacen la función de "dispositivos de entrada", recoger las señales de las diferentes neuronas y enviarlas al soma de la célula. A su vez, el soma o cuerpo celular hace las funciones de "*unidad central de proceso*" implementando en su interior una función no lineal de respuesta. Generando una señal de salida en el caso que el potencial de membrana sea lo

suficientemente grande. El axón hace las funciones de dispositivo de salida, que conduce las señales que genera a todas las neuronas a las que está conectado. La unión entre dos neuronas (una dendrita con un axón) recibe el nombre de sinapsis. En la práctica, los axones de una neurona pueden llegar a medir varios centímetros interconectando diversas partes del cerebro.

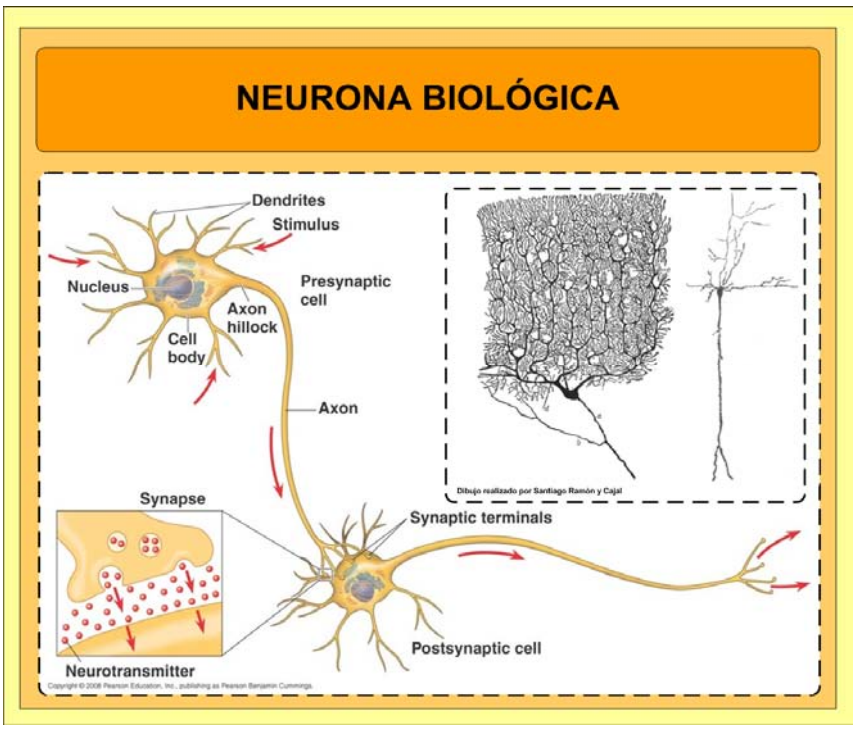

Figura 4-8: Neurona biológica

Se estima que hay alrededor de 10^{11} neuronas en el cerebro humano [4-33], las cuales se hallan repartidas en una serie de estructuras anatómicas bien diferenciadas, como son: el tronco cerebral, el cerebelo y la corteza cerebral. En cada una de estas estructuras existen diferentes tipos de neuronas, caracterizadas por un patrón de conexión específico y una respuesta a estímulos diferente, proporcionando un pulso de salida al conjunto de dendritas de las neuronas a las que se halla interconectada (Figura 4-8). La conectividad de cada neurona cortical es muy grande, habiéndose estimado que una neurona de este tipo se encuentra típicamente conectada a entre 1000 y 10000 neuronas [4-46].

Las señales neuronales consisten en una serie de pulsos eléctricos de corta duración, que reciben el nombre de potenciales de acción (AP) con una amplitud típica de unos 100mV y

una duración de entre 1ms y 2ms. Cabe remarcar que la señal de salida de una neurona (AP) no se degrada [4-33] en su trayecto a través del axón. A su vez, una cadena de potenciales de acción (AP) emitidos por una sola neurona se denomina tren de pulsos ”*Spike Train*”, ya sea una secuencia de eventos que se den de forma regular o irregular. Puesto que todos los pulsos que se transmiten entre neuronas son muy parecidos, se ha deducido que la forma del potencial de acción no tiene asociada ninguna información, sino que esta información se halla codificada en el número y en la temporalización de los mismos.

4.2.2.1 Propiedades de la membrana celular nerviosa

Las neuronas usan diversos tipos de señales eléctricas para codificar y transmitir la información. Dichas señales son causadas por las respuestas a los estímulos, siendo generadas por una serie de mecanismos biológicos bastante elaborados como los flujos iónicos a través de las membranas plasmáticas de las células nerviosas [4-33].

El potencial eléctrico de la membrana celular se denomina *potencial de membrana*, el cual es generado a partir de las diferencias en las concentraciones entre los iones específicos que se ubican dentro y fuera de las células nerviosas. Cada membrana nerviosa tiene asociado un potencial eléctrico residual que varía típicamente entre los -40mV y -90mV [4-49]. El gradiente de la concentración iónica y con ello el potencial de membrana, se ve regulado mediante los mecanismos de transporte a través de los canales iónicos activos en cada momento. Los mecanismos de transporte, también conocidos como bombas iónicas, son una serie de proteínas que se desplazan iones dentro y fuera de las células en contra del gradiente de la concentración iónica. Por otro lado, los canales iónicos son proteínas que permiten que sólo ciertos tipos de iones atraviesen la membrana celular, siempre en la dirección del gradiente de concentración iónica. Éstos se caracterizan por su selectividad en la permeabilidad de la membrana a un tipo dado de ión. Ambos mecanismos trabajan básicamente uno en contra del otro, siendo los causantes de la existencia de un potencial residual de la membrana celular, así como de la creación del potencial de acción y de los potenciales sinápticos.

4.2.2.2 La sinapsis

El lugar donde el axón de una neurona hace contacto con las dendritas de la neurona a la que está conectada recibe el nombre de sinapsis. Siendo la sinapsis química el tipo de sinapsis más común en los vertebrados [4-49, 4-33].

En la sinapsis química (Figura 4-9), el axón se acerca extremadamente a las dendritas de la neurona post-sináptica, dejando un espacio extremadamente pequeño entre ambas membranas celulares, dicho espacio recibe el nombre de hendidura sináptica. Este tipo de sinapsis actúa mediante procesos bioquímicos, donde neurotransmisores y neuroreceptores se encargan de cargar el potencial de membrana de la neurona post-sináptica. Es de reseñar que en el caso de la sinapsis química un potencial de acción no puede atravesar la hendidura sináptica existente entre dos neuronas de forma directa. En su lugar, dicho pulso nervioso es transportado químicamente entre dos neuronas mediante los llamados neurotransmisores. Éstos son segregados por la célula nerviosa que emite el pulso (célula presináptica), hallándose almacenados en las vesículas sinápticas ubicadas en el extremo del axón; la célula nerviosa que debe recibir el pulso nervioso (neurona post-sináptica) dispone en la superficie de su membrana de una serie de canales iónicos cuya apertura/cierre está controlada químicamente por un determinado neurotransmisor. La sinapsis química (Figura 4-9) se compone esencialmente de los siguientes elementos:

- Los versículos sinápticos: ubicados en el extremo del axón donde se almacenan los neurotransmisores. Dichos neurotransmisores se sintetizan en las mitocondrias y en otros orgánulos de la célula nerviosa.
- Los receptores sinápticos: se ubican en los extremos de la membrana celular de las dendritas, donde se hallan los neuroreceptores encargados de responder a un determinado neurotransmisor.
- La hendidura sináptica: es el espacio que existe entre la neurona presináptica y la postsináptica, siendo el lugar donde se produce la sinapsis propiamente dicha.

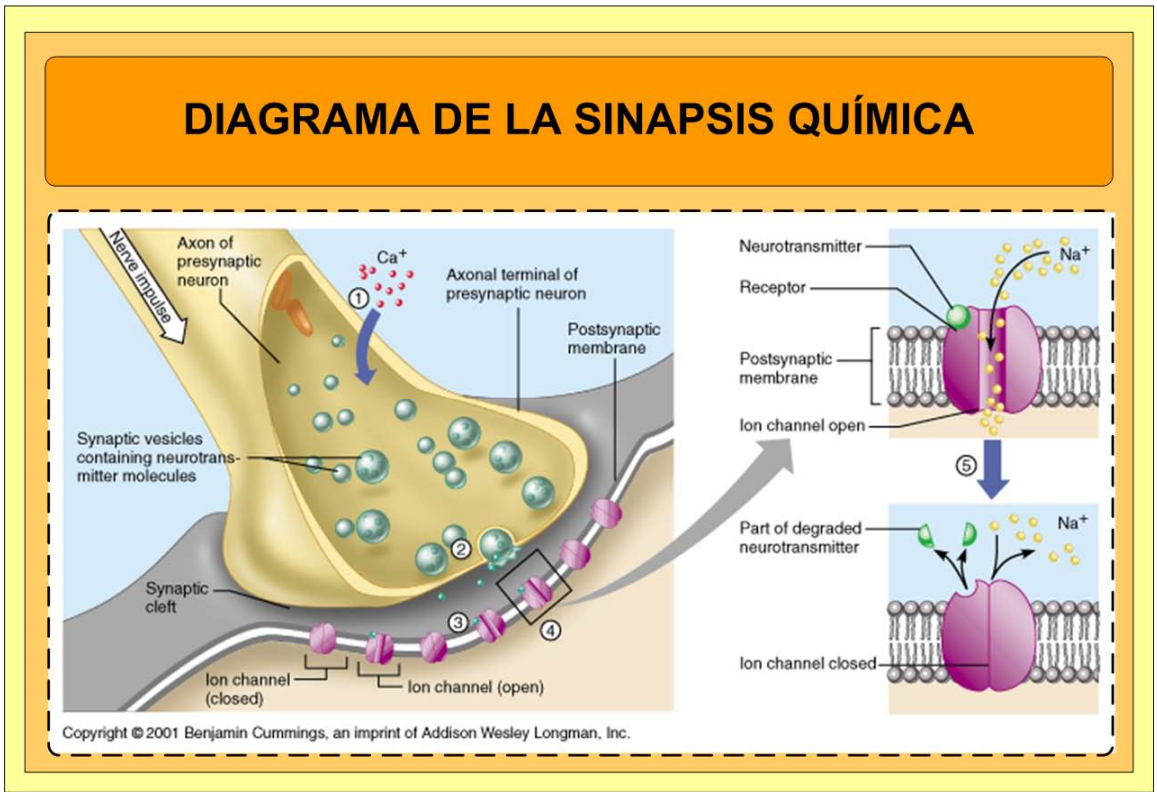

Figura 4-9: Diagrama de la sinapsis química

El proceso de creación de un potencial de acción (AP) [4-33] es el siguiente: inicialmente un neurotransmisor es sintetizado y almacenado en los versículos sinápticos, al llegar un potencial de acción a la zona terminal de las dendritas se despolarizan los canales de calcio haciendo fluir los iones hacia el interior de la célula. Este hecho provoca que los versículos sinápticos se fundan con la membrana liberando en la hendidura sináptica los neurotransmisores que contienen. A este proceso se le conoce con el nombre de exocitosis o segregación celular. Una vez los neurotransmisores han sido segregados, estos se difunden a través del canal sináptico hasta que interactúan con la membrana celular de la neurona post sináptica, donde se ubican una serie de neuroreceptores. En el caso que determinados neurotransmisores sean compatibles con algún neuroreceptor, éstos se unirán entre ellos causando así que se abra o cierre un determinado canal iónico de la membrana post sináptica. Esto, a su vez, provocará una despolarización (EPSP) o una hiperpolarización (IPSP) de la membrana celular postsináptica, que puede conducir a la

emisión de un potencial de acción postsináptico (AP) en el caso en que el potencial de membrana supere un determinado nivel de disparo (Figura 4-11).
Paralelamente, desde el instante en que se produce la liberación de los neurotransmisores, las vesículas sinápticas de la neurona presináptica inician un proceso de retorno a su posición desde la membrana plasmática, disminuyéndose paulatinamente la liberación de neurotransmisores. También existen toda una serie de encimas en el canal sináptico que se encargan de eliminar y rebajar la concentración de neurotransmisores. Los restos de los neurotransmisores eliminados por las encimas son absorbidos por la neurona presináptica mediante un proceso de endocitosis y usados para resintetizar más neurotransmisores mediante las mitocondrias. Este proceso biológico asegura que una neurona no esté activa permanentemente.

Además de la sinapsis química, las neuronas presentes en los sistemas nerviosos de los mamíferos pueden interactuar/acoplarse entre ellas mediante otros mecanismos sinápticos. En el caso del sistema nervioso del ser humano, éste hace uso de gran número de neurotransmisores y neuroreceptores, donde ninguno de ellos funciona de igual forma. En la práctica, los tipos de sinapsis se pueden agrupar en 5 grandes tipos [4-33, 4-49], que son:

- **Sinapsis excitatorias de canales iónicos:** este tipo de sinapsis tiene como neuroreceptores los canales del sodio. Cuando estos canales se hallan abiertos, los iones positivos fluyen hacia dentro de la célula, causando una despolarización local y aumentando la probabilidad de generación de un potencial de acción (AP). Este es el tipo de sinapsis que se ha descrito anteriormente. Los neurotransmisores típicos son la acetilcolina, el glutamato o el aspartato. Por otra parte, cuando los canales "Na^+" se hallan abiertos se produce una despolarización local, y si el nivel de disparo es alcanzado, se iniciará la emisión de un potencial de acción (AP).
- **Sinapsis inhibitorias de canales iónicos:** Este tipo de sinapsis usa como neuroreceptores los canales cloruro, que cuando se hallan abiertos dejan fluir dentro de la célula los iones negativos "Cl^-", causando una hiperpolarización de la membrana

provocando que la emisión de un potencial de acción (AP) sea mucho menos probable. Esta sinapsis permite inhibir la generación de pulsos por parte de la neurona. En este caso los neurotransmisores típicos son la glicina o el ácido aminobutírico.

- **Sinapsis sin canal:** Este tipo de sinapsis no hace uso de ningún canal iónico como elemento neuroreceptor, en su lugar usa las encimas que mantiene la membrana unida. Cuando éstas se activan mediante la emisión de un neurotransmisor catalizan un "*mensajero químico*" en el interior de la célula, el cual a su vez permite alterar toda una serie de aspectos del metabolismo de la célula. Particularmente, estos mensajeros químicos pueden alterar el número y la sensibilidad de los canales iónicos receptores presentes en dicha neurona. Este tipo de sinapsis está relacionado con la respuesta a corto y largo plazo, como es el caso del aprendizaje y la memoria.
- **Uniones Neuromusculares:** Este tipo de sinapsis se da entre una neurona motora y una célula muscular. El neurotransmisor usado es la acetilcolina, que se halla asociado a procesos excitatorios. Como neuroreceptor se utilizan las encimas que mantienen unida la membrana celular. Por lo tanto, cuando se activan los neuroreceptores éstos actúan modificando la sensibilidad de los canales iónicos receptores entre las neuronas motoras y las musculares. Adicionalmente, las neuronas motoras también forman sinapsis especializadas con las células secretoras.
- **Sinapsis eléctricas:** En este tipo de sinapsis, las membranas celulares de las dos neuronas deben tocarse a fin de poder compartir diferentes proteinas. Este hecho permite que el potencial de acción (AP) pase directamente de una membrana a la otra, siendo esta sinapsis extremadamente rápida pero a su vez bastante rara, ya que solamente se ha podido probar su existencia en el corazón y en los ojos. Concretamente de este tipo de sinapsis no se conocen demasiado bien sus aspectos funcionales, pero se cree que está relacionada con la sincronización neuronal.

4.2.2.3 El potencial post-sináptico (PSP)

La diferencia de potencial entre el interior del cuerpo celular (soma) y sus alrededores recibe el nombre de potencial de membrana. Este potencial se ve alterado por los llamados potenciales post sinápticos (PSPs) que se generan a partir de los pulsos recibidos desde las

neuronas presinápticas. En el caso que el potencial de membrana supere un determinado nivel de disparo, un potencial de acción (AP) es activado y enviado a través del axón y todas sus ramificaciones hasta llegar a las neuronas post sinápticas. En el supuesto que el potencial postsináptico sea positivo se lo denominará "*Potencial Post Sináptico excitatorio*" (EPSP), y en el caso que éste sea negativo se lo denominará "*Potencial Post Sináptico Inhibitorio*" (IPSP). Un ejemplo de cada uno de ellos se presenta en la Figura 4-10. Cabe remarcar que el proceso de emisión de un pulso de transmisión a través del axón tiene asociado un retardo a la transmisión, llamado retardo axonal [4-49, 4-33].

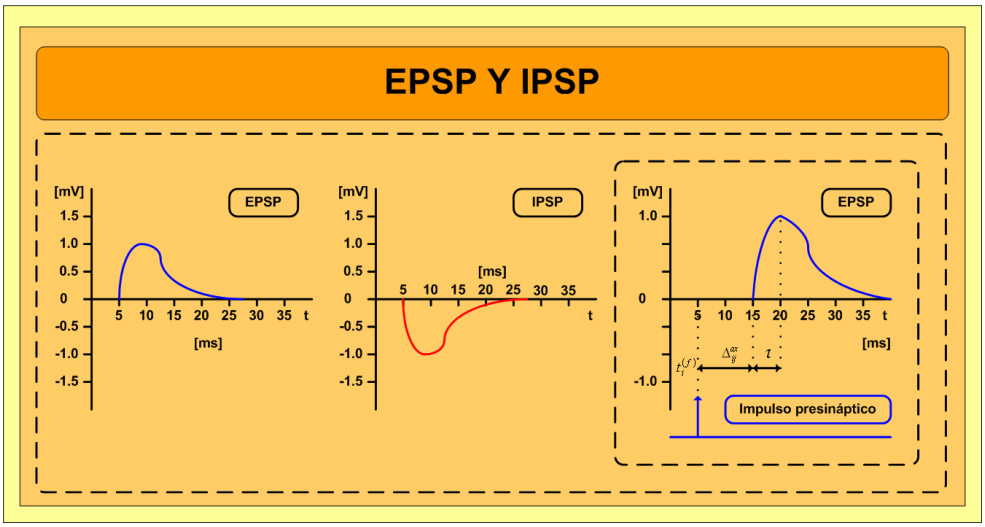

Figura 4-10: Potenciales (EPSP) e (IPSP)

En reposo, la membrana celular está polarizada negativamente con un valor del orden de los -65mV [4-76]. En el caso de una entrada excitatoria, dicha sinapsis reduce la polarización negativa de la membrana celular, este proceso recibe el nombre de *despolarización*. Las amplitudes típicas de los EPSP son del orden de 1mV. Por otro lado, una sinapsis inhibitoria provoca un aumento de la polarización negativa de la membrana, dicho proceso recibe el nombre de *hiperpolarización*. Generalmente, el nivel de disparo de un Potencial de Acción (AP) se haya en el rango de los 20-30mV [4-34].

A continuación, describiremos la descripción matemática del proceso de generación de un *Potencial Post Sináptico* (PSP). La neurona presináptica la denotaremos como "i" y la post

sináptica como “j”. El conjunto de neuronas presinápticas que están conectadas a una determinada neurona post sináptica “j” será descrito como “$\Gamma_j = \{i \mid i \text{ presinapticas a j}\}$”. El instante en que una neurona “j” emite un potencial de acción (AP) se conoce con el nombre de *instante de disparo* y se denota como “$t_j^{(f)}$”, donde el superíndice “f” puede variar entre 1 y n (identificador del disparo). De esta forma, un tren de pulsos de la neurona “j” se puede caracterizar mediante el conjunto de instantes de disparo dados por la Ecuación [4-6], donde el instante de disparo “$t_j^{(n)}$” se corresponde con el instante de emisión del pulso más reciente.

$$F_j = \left\{t_j^{(1)}, t_j^{(2)}, t_j^{(3)}, ..., t_j^{(n)}\right\} \qquad \text{Ecuación [4-6]}$$

El Potencial Post Sináptico (PSP) es la respuesta generada por la neurona post sináptica “j” y dicha función tiene la forma que se muestra en la figura 4-11. Debe tenerse en cuenta que la distancias intraneuronales y los tiempos de transmisión axonales finitos provocan que exista cierto retardo entre el instante de emisión de un pulso y el inicio de su correspondiente PSP. La expresión matemática que describe la formación de un PSP en una neurona “j” a partir de los pulsos emitidos por una neurona “i”, se presenta en la Ecuación [4-7] según lo indicado en referencia [4-34]:

$$\varepsilon_{ij}(t) = u_i(t) - u_{rest}(t=0) = \left[Exp\left(-\frac{\left(t - t_i^{(f)} - \Delta_{ij}^{ax}\right)}{\tau_{delay}} \right) - Exp\left(-\frac{\left(t - t_i^{(f)} - \Delta_{ij}^{ax}\right)}{\tau_{rise}} \right) \right] \cdot \Theta\left(t - t_i^{(f)} - \Delta_{ij}^{ax}\right)$$

Ecuación [4-7]

En la literatura hallamos todo un espectro de simplificaciones de la anterior expresión. Donde “τ_{delay}” y “τ_{rise}” son las constantes de tiempo de la respuesta del sistema en el primer caso, “Δ_{ij}^{ax}” es el retardo a la transmisión axonal, y “$\Theta(x)$” se corresponde con la función escalón de *Heaviside.* A pesar de todo, la ecuación [4-7] es ampliamente usada para la simulación de potenciales post sinápticos (PSP). No obstante, en ámbitos más formales/realistas (desde el punto de vista biológico) la función de kernel “$\varepsilon_{ij}(t)$” debe ser también función del último instante en que la neurona “j” se activó, ya que el PSP depende directamente del estado del potencial de membrana, el cual a su vez depende directamente

de "$\hat{t}_j$". En este caso la función de kernel "$\varepsilon_{ij}\left(t-\hat{t}_j, t-t_i^{(f)}-\Delta_{ij}^{ax}\right)$" puede interpretarse como el tiempo que debe transcurrir para la generación de un potencial post sináptico desde el instante "$t_i^{(f)}$" en que se dispara una neurona presináptica "i".

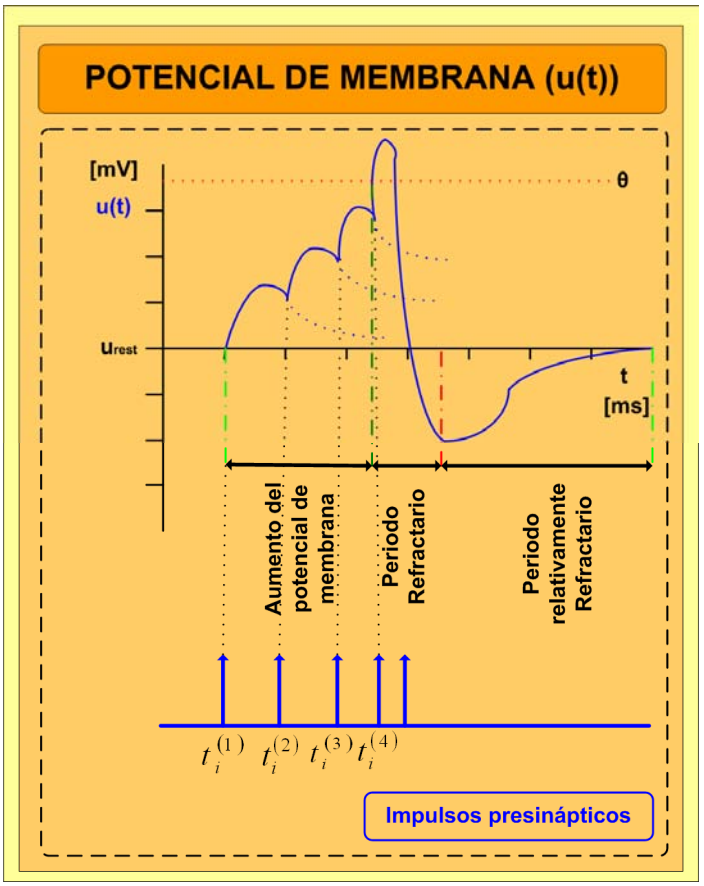

Figura 4-11: Potencial de membrana

Justo después de la emisión de un pulso, el potencial de membrana no retorna directamente al valor del potencial residual "u_{rest}", sino que pasa a través de una fase de híper-polarización [4-34] que hace que el nivel de polarización de la membrana esté muy por debajo del valor del potencial residual "u_{rest}". Con todo ello, mientras el potencial de acción (AP) sigue creciendo rápidamente o bajando abruptamente por encima del nivel de disparo, es imposible la emisión de un nuevo pulso, llamándose a este efecto "refractariedad absoluta". La hiperpolarización de la membrana después de la emisión de

un pulso hace que el potencial de membrana sea muy negativo justo después de la emisión de dicho pulso. Este potencial negativo va disminuyendo a lo largo de un período de tiempo hasta situarse en el valor del potencial residual de membrana "u_{rest}". Este intervalo de tiempo recibe el nombre de período de refractariedad relativa, dicho fenómeno se ilustra en la figura 4-11.

4.2.2.4 El potencial de equilibrio

La membrana neuronal se comporta como un material dieléctrico casi perfecto, a la vez que separa el medio intracelular del medio celular circundante. Como se ha descrito con anterioridad, en dicha membrana se hallan incrustadas todo un conjunto de proteínas que son las encargadas del transporte y canalización de iones específicos, los cuales se encargan del transporte activo de iones entre ambos medios dando lugar a diferentes concentraciones a cada lado de la membrana. La diferencia de concentraciones iónicas genera a su vez una diferencia de potencial entre ambos lados de la membrana. En el equilibrio, la diferencia de potencial generada por dicha diferencia de densidades vendrá descrita por la ecuación de Nerst [4-76] (Ecuación [4-8]):

$$E = \frac{k \cdot T}{q} \cdot Ln\left(\frac{C_{in}}{C_{out}}\right) \qquad \text{Ecuación [4-8]}$$

La ecuación de *Nerst* permite hallar el potencial de reducción de un electrodo cuando las condiciones no son las estándar (concentración 1 Mol, presión de 1atm, temperatura de 298 K). El parámetro "E" es el *potencial de equilibrio* o *potencial de Nerst*, "$k = 1.4 \cdot 10^{23} J / K$" es la constante de *Blotzman*, "T" es la temperatura del medio medido en Kelvin, "q" es la carga eléctrica de los iones involucrados, y "C_{in}" y "C_{out}" son las concentraciones iónicas dentro y fuera de la membrana celular. El *potencial de equilibrio* es conocido también como *potencial de reversión*, ya que dicho valor representa el punto en el que los iones invierten la dirección del flujo iónico para mantener el equilibrio. El potencial de equilibrio para los iones sodio (Na^+) es de $E_{Na} \approx -77mV$ y para los iones potasio (K^+) es de $E_K \approx +55mV$ [4-34]. En la Tabla 4-2 se presentan las concentraciones iónicas más habituales en los mamíferos.

Tabla 4-2: Concentraciones iónicas extracelulares y intracelulares (mamíferos)		
Ión	**Concentración intracelular [mM]**	**Concentración extracelular [mM]**
Potasio (K^+)	140	5
Sodio (Na^+)	5-15	145
Cloro (Cl^-)	4-30	110
Calcio (Ca^{++})	0.0001	1-2

Como hemos visto, en los procesos neuronales reales existen varios tipos de iones presentes simultáneamente y que contribuyen a la creación del potencial de membrana. El potencial de reposo "u_{rest}" viene determinado por el equilibrio dinámico entre los flujos iónicos que circulan a través de los diversos canales (que controlan la permeabilidad de la membrana) y el transporte activo realizado por las bombas iónicas. En el caso que exista más de un ión permeante en el sistema, la ecuación de *Nerst* deberá ser sustituida por la ecuación [4-9] que tiene en cuenta la permeabilidad de la membrana de cada ión involucrado en el proceso [4-49, 4-33]

$$E = 58 \cdot Log_{10}\left(\frac{P_K[K]_{out} + P_{Na}[Na]_{out} + P_{Cl}[Cl]_{in}}{P_K[K]_{in} + P_{Na}[Na]_{in} + P_{Cl}[Cl]_{out}}\right) \qquad \text{Ecuación [4-9]}$$

Donde "P_x" es la permeabilidad de la membrana, mientras que los subíndices "in" y "out" indican si la concentración de iones es la de dentro o fuera de la membrana. En el caso del catión "Cl^-" los índices han sido invertidos ya que en esta ecuación no se dispone de ningún factor que incorpore el signo de la carga iónica, mientras que el factor "58" se refiere a la temperatura de la estancia.

4.2.3 MODELOS FORMALES DE NEURONAS PULSANTES

Desde un punto de vista funcional, la actividad de las redes neuronales puede describirse en diversos niveles de abstracción y la elección de un modelo u otro dependerá del nivel demandado. Si deseamos describir a nivel microscópico el funcionamiento de una neurona, resulta que un gran número de canales iónicos y poros de las membranas celulares nerviosas se abren o cierran en función del voltaje al cual éstas se hallan o si están presentes/ausentes determinadas moléculas transmisoras. Para describir estos procesos, utilizaremos los modelos comportamentales que describen cada pequeño segmento de una

neurona como un conjunto de ecuaciones iónicas. Finalmente, si estamos interesados en un nivel alto de abstracción no requeriremos de ninguna información acerca de la distribución espacial de la neurona y mucho menos de los mecanismos iónicos involucrados en su operación. Se tomará la neurona como una unidad simple y homogénea, que generará pulsos de salida en el caso que la excitación recibida sea lo suficientemente grande. Los modelos que se ajustan a esta descripción son los modelos de ratio "*Rate models*". Claramente, la elección de un modelo u otro debe hacerse desde el punto de vista del tipo de fenómeno o comportamiento neuronal que se pretenda simular, o que se ajuste más a las necesidades de la aplicación que se pretende realizar.

Los modelos de neuronas pulsantes "*Spiking Neurons*" pueden dividirse en tres grandes grupos: los modelos basados en el nivel de disparo "*Threshold-Fire*", los modelos basados en conductancia "*Conductance* " y los comportamentales [4-34]. A lo largo de los siguientes subapartados se tratará cada uno de estos modelos a fin de presentar una visión general de cada uno de ellos, y situar así al lector en el marco en el cual se han desarrollado los diversos trabajos de investigación que describiremos posteriormente. Existe un cuarto grupo de modelos, llamados modelos de tasa "*Rate models*" [4-76, 4-34] que presentaremos aunque en la práctica no puedan considerarse como verdaderos modelos de neurona pulsante desde el punto de vista de una descripción biológica purista.

4.2.3.1 Modelos de *Threshold-Fire*

Los modelos "*Threshold-Fire*" se basan en realizar una suma temporal de todas las contribuciones al potencial de membrana $u(t)$ recibidas desde todas las neuronas presinápticas. Si dicha suma excede el nivel de disparo "θ" de la neurona post-sináptica ésta disparará. En el caso más simple uno puede asumir consiste en que el potencial post sináptico (PSP) crezca linealmente en el soma. Presentaremos dos de los modelos más representativos de esta categoría, como son el "*Integrate-and-fire*" (I&F) [4-36] y el modelo "Spike Response" (SRM) [4-34].

4.2.3.1.1 Modelo de Integrate-and-Fire (I&F)

El modelo de "*Integrate-and-fire*" (I&F) [4-36] es uno de los más conocidos y usados de neurona pulsante. También se lo conoce como modelo de "*Leaky-Integrate-and-Fire*" (LIF) [4-34], ya que se asume que la membrana se hiperpolariza de forma natural a través de los canales iónicos formados después de un potencial post sináptico (PSP).

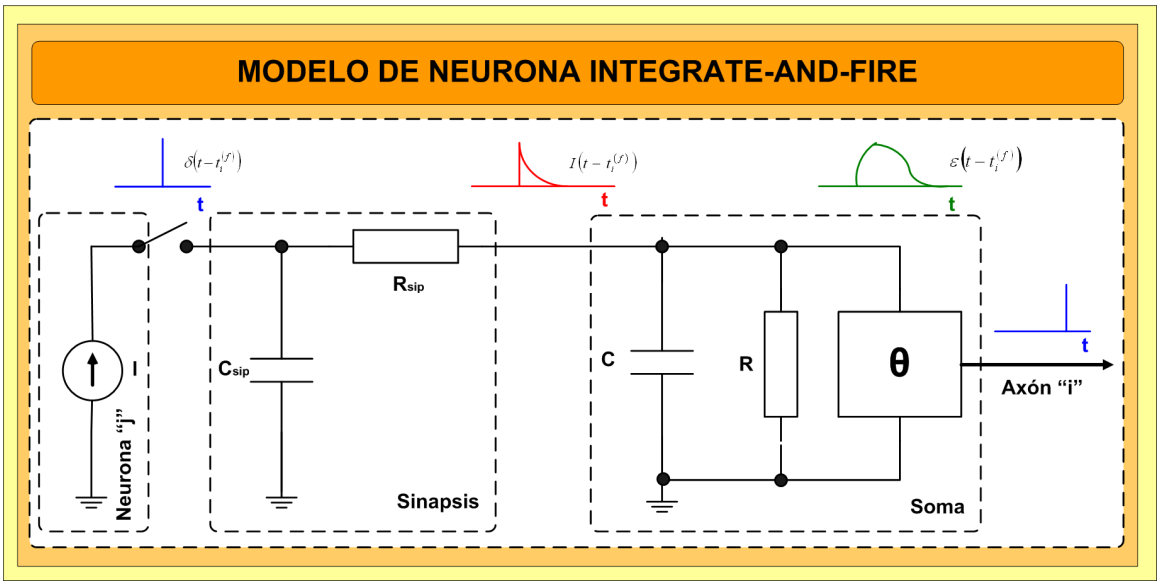

Figura 4-12: Modelo de neurona Integrate-and-Fire

Dicho modelo eléctrico (Figura 4-12), se basa en un circuito RC en paralelo (marcado por la línea discontinua) que hace la función de la soma (sumar temporalmente los pulsos recibidos). Dicho circuito se carga mediante pulsos de corriente $I(t)$. En el caso que la diferencia de potencial sobre el condensador "C" supere el valor de disparo "ϑ", el circuito se cortocircuita y genera un pulso de salida hacia las otras neuronas. El pulso emitido por la neurona presináptica se ha modelado como un filtro pasa bajas pasivo de primer orden, que se encarga de convertir el pulso de tensión $\delta\left(t-t_i^{(f)}\right)$ en un pulso de corriente $I\left(t-t_i^{(f)}\right)$. Los pulsos de corriente cargan el condensador C del circuito RC en paralelo, que generará el potencial postsináptico $\varepsilon\left(t-t_i^{(f)}\right)$. El condensador posee una resistencia en paralelo que lo descargará permanentemente. Siendo la Ecuación [4-10] del sistema la siguiente:

$$I(t) = \frac{u(t)}{R} + C \cdot \frac{du(t)}{dt} \rightarrow \tau_m \cdot \frac{du(t)}{dt} = -u(t) + R \cdot I(t) \qquad \text{Ecuación [4-10]}$$

La constante de tiempo $\tau_m = R \cdot C$, proporcionará el valor del tiempo de respuesta de la membrana, mientras que la tensión $u(t)$ se corresponderá con el potencial de membrana.

Para poder describir el comportamiento de la neurona pulsante, se requiere además de una condición de disparo. La neurona emitirá un pulso en el instante $t = t^{(f)}$ cuando el nivel de disparo "ϑ" es alcanzado, o lo que es lo mismo $u\left(t^{(f)}\right) = \vartheta$. Como puede apreciarse, en este modelo la forma del pulso no tiene relevancia y lo que realmente importará será el instante de emisión $t^{(f)}$. Inmediatamente después de dicha emisión el potencial de la membrana es fijado a "u_{rest}", valor a partir del cual se volverá a iniciar la integración. La condición adicional se regirá por la Ecuación [4-11]:

$$\lim_{t \rightarrow t^{(f)}, t < t^{(f)}} u(t) = u_{rest} \qquad \text{Ecuación [4-11]}$$

Una solución analítica para este modelo puede obtenerse considerando una corriente de entrada constante "I_0" y un potencial de reset "$u_{rest} = 0$". La condición inicial del sistema será "$u(t) = u_{rest} = 0$" lo que equivale a la siguiente expresión del potencial de membrana (Ecuación [4-12]):

$$u(t) = R \cdot I_0 \cdot \left[1 - e^{-\left(\frac{\left(1 - t^f\right)}{\tau_m} \right)} \right] \qquad \text{Ecuación [4-12]}$$

Si suponemos que un pulso es emitido en "$t = t^{(1)}$" el siguiente pulso de salida no se dará hasta que "$u(t) > \vartheta$". Para que se genere un pulso en el instante $t = t^{(2)}$ deberá cumplirse la condición "$u\left(t^{(2)}\right) = \vartheta$". A partir de dicha condición se podrá evaluar el tiempo medio "T" entre disparos (Ecuación [4-13]):

$$\vartheta = R \cdot I_0 \cdot \left[1 - e^{-\left(\frac{\left(t^{(2)} - t^{(1)} \right)}{\tau_m} \right)} \right] \rightarrow \begin{cases} T = \left(t^{(2)} - t^{(1)} \right) \\ \vartheta = R \cdot I_0 \cdot \left[1 - e^{-\left(\frac{T}{\tau_m} \right)} \right] \end{cases} \Rightarrow T = \tau_m \cdot \ln\left(\frac{R \cdot I_0}{R \cdot I_0 - \vartheta} \right)$$

Ecuación [4-13]

De la anterior Ecuación [4-13] se deduce que el período entre disparos "T" depende directamente de la corriente "I_0". Sí pretendemos tener en cuenta el período absoluto de refractariedad "Δ^{abs}", deberá sumarse al período entre disparos "T", con lo cual la frecuencia de disparo de la neurona "υ" vendrá dada por la Ecuación [4-14]:

$$\nu = \frac{1}{\left(\Delta^{abs} + \tau_m \cdot \ln\left(\frac{R \cdot I_0}{R \cdot I_0 - \vartheta} \right) \right)}$$

Ecuación [4-14]

En la literatura, hallamos diversas variaciones del modelo lineal de "*Integrate-and-fire"*. Éstas incorporan una segunda ecuación diferencial lineal para describir la dinámica de la corriente iónica del potasio "K^+" (la bomba de sodio/potasio) en la fijación de un alto nivel de disparo, fijando así un mecanismo que permite la variación en la frecuencia de disparo. Otras variantes son el modelo "*Integrate-and-Fire*" o "*Burst Neuron*" [4-77, 4-78] usado para modelar las neuronas corticales, y el modelo de "Resonate-and-Fire" [4-79].

En la Figura 4-13 se presenta un ejemplo de potencial de membrana [4-34] obtenido mediante un modelo lineal de Integrate-and-Fire, correspondiente a una corriente de entrada variable con el tiempo, con los siguientes parámetros: $\tau = 10ms$, $u_{rest} = 0$, R=1Ω y $I_0 = 1.5mA$.

No obstante, existe una versión más genérica de este modelo, que es el conocido como el modelo no lineal de "*Integrate-and-Fire*", que se rige por la siguiente Ecuación [4-15]:

$$\tau \cdot \frac{du(t)}{dt} = F(u(t)) + G(u(t)) \cdot I(t) \rightarrow \tau \cdot \frac{du(t)}{dt} = a_0 \cdot (u - u_m) \cdot (u - u_c) + R \cdot I$$

Ecuación [4-15]

Donde "$F(u(t))$" es una función de decaimiento constante dependiente del potencial de membrana y "$G(u(t))$" se corresponde con el valor de la resistencia dependiente del potencial de membrana. Modelos basados sobre esta forma más general son el "*Quadratic Integrate-and-Fire*" [4-34, 4-77]. En dicho modelo el valor de a_0 se toma positivo y constante mientras que el valor del voltaje crítico se toma con un valor mayor que el potencial de reset "$u_c > u_{rest}$" , por encima del cual su derivada temporal será positiva. Este modelo se conoce en la literatura como el *modelo canónico de Ermentrout-Kopell* [4-77] o neurona Θ, el cual incorpora latencias entre los pulsos, un nivel de disparo dependiente de la actividad y una biestabilidad de los modos de pulso (descanso/tonificación) [4-78].

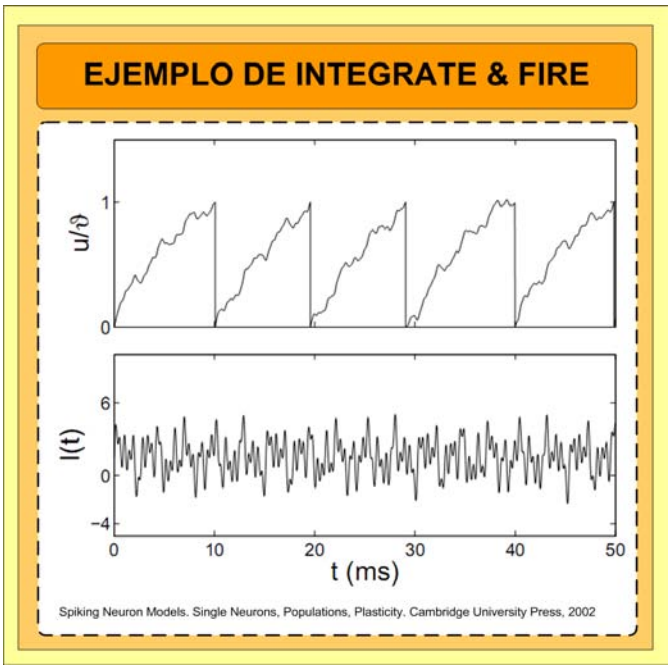

Figura 4-13: Ejemplo de u(t) obtenida mediante el modelo canónico de Ermentrout-Kopell

4.2.3.1.2 Modelo de Spike Response (SRM)

En este modelo el estado de la neurona "j" se describe mediante una sola variable de estado "$u_j(t)$". En caso que el valor de $u_j(t)$ se halle por encima el nivel de disparo "ϑ" en el instante "$t_j^{(f)}$", entonces se generará a la salida de la neurona (axón) un pulso mediante el

cual, formalmente denotaremos todo el conjunto de instantes/tiempos de disparo de la neurona "j" como (Ecuación 4-16):

$$F_j = \left\{ t_j^{(f)}; 1 \le f \le n \right\} = \left\{ t \mid u_j(t) = \vartheta \right\} \qquad \text{Ecuación [4-16]}$$

En la Figura [4-14], podemos apreciar como existen dos fenómenos diferentes que influyen en el potencial de membrana de la neurona "$u_j(t)$". El primero se corresponde a la contribución de todas las neuronas presinápticas "$i \in \Gamma_j$" que modela la respuesta de la neurona a los pulsos presinápticos (Ecuación [4-17]):

$$h(t) = \sum_{i \in \Gamma_j} \left(\sum_{t_i^{(f)} \in \Gamma_j} w_{ij} \cdot \varepsilon_{ij} \left(t - \hat{t}_j, t - t_i^{(f)} - \Delta_{ij}^{ax} \right) + I_j^{ext} \right) \qquad \text{Ecuación [4-17]}$$

Siendo "w_{ij}" el término que representa la eficacia de la sinapsis, mientras que "I_j^{ext}" se corresponde con la excitación externa y "ε_{ij}" es el potencial post sináptico de la neurona (PSP) (el cual puede ser excitatorio o inhibitorio), este último se evaluará mediante la Ecuación [4-7] anteriormente descrita. Cabe remarcar que el retardo axonal (Δ^{ax}) es despreciado y no aparecerá en las ecuaciones. Para evitar confusiones se realiza el siguiente cambio de variables $s = t - t_i^{(f)} - \Delta_{ij}^{ax}$.

La segunda contribución que hallamos en la respuesta del potencial de membrana $u_j(t)$ es el de reset, que se encarga de disminuir el potencial de membrana hasta el valor de "u_{rest}", incluyendo el *overshoot* negativo que sigue a la emisión del pulso de salida de la neurona (a fin de implementar el período refractario). Este nuevo término describe la respuesta de la neurona "j" a la emisión de su pulso de salida, y se modela mediante un kernel de respuesta "$\eta_j\left(t - \hat{t}_j \right)$" el cual dependerá del instante en que se haya disparado la neurona "$j\left(\hat{t}_j \right)$". Formalmente, los otros instantes de disparo de la neurona "j" se tienen en cuenta en el término de la suma "$\sum \eta_j\left(t - \hat{t}_j^{(f)} \right)$", con lo cual, la aproximación realizada no altera los resultados del modelo [4-36].

Para modelar la respuesta lineal del potencial de membrana a una corriente externa "I^{ext}" dada, se modificará el último término de la ecuación [4-17] para incorporar esta corrección en la expresión del potencial post-sináptico (Ecuación [4-18]).

$$h(t)=\sum_{i\in\Gamma_j}\left(\sum_{t_i^{(f)}\in\Gamma_j} w_{ij}\cdot\varepsilon_{ij}\left(t-\hat{t}_j,t-t_i^{(f)}-\Delta_{ij}^{ax}\right)+\int_0^{\infty}k\left(t-\hat{t}_j,s\right)\cdot I^{ext}(t-s)\cdot ds\right)$$

Ecuación [4-18]

La dependencia con el tiempo transcurrido desde la emisión del último pulso a "$\hat{t}_j$" se explica por el hecho que inmediatamente después de la emisión de éste la mayoría de canales iónicos se hallan abiertos, lo que implica una reducción significativa de la resistencia de la membrana [4-34].

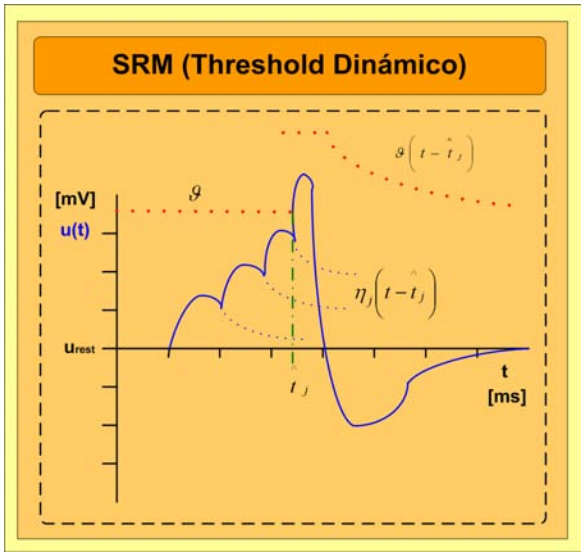

Figura 4-14: Potencial de membrana (efecto Dynamic Threshold)

En este modelo, el estado "$u_j(t)$" de la neurona "j" en un instante dado "t" vendrá dado por una superposición lineal de todas las contribuciones anteriormente descritas (Ecuación [4-19]).

$$u_j(t)=\eta_j\left(t-\hat{t}_j\right)+h\left(t\mid\hat{t}_j\right)$$ Ecuación [4-19]

La refractariedad, además de hallarse incorporada en el término del kernel "η", puede modelarse adicionalmente mediante un aumento del nivel de disparo después de la emisión del pulso "$\vartheta \to \vartheta(t - \hat{t}_j)$" (en el supuesto que se pretenda prevenir un disparo de la neurona en este período). Esto es equivalente a una híper-polarización del potencial de membrana como puede apreciarse en la (Figura 4-15). Esta técnica se conoce con el nombre de "Dynamic Threshold" [4-34, 4-36].

No obstante, existe un modelo de respuesta de pulso más simplificado "SRM_0" [4-36], que se basa en una serie de modificaciones del modelo completo ya descrito. El potencial de membrana que implementa se presenta en la Figura 4-15.

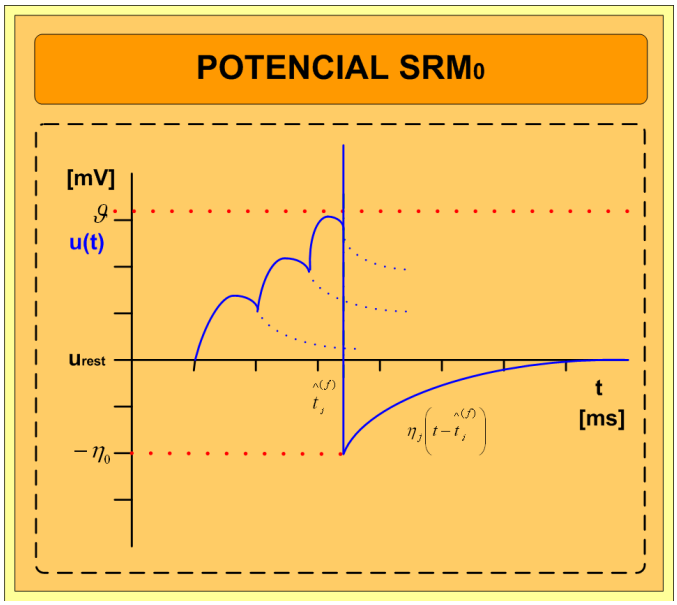

Figura 4-15: Modelo de potencial de membrana SMR0

El modelo simplificado supone que el valor de "ε_{ij}" se mantiene constante en todas las neuronas presinápticas, a la vez que es independiente del argumento "$t - \hat{t}_j$". También supone que la función kernel "η" es independiente del argumento "$t - \hat{t}_j$", con lo cual, la expresión del modelo simplificado pasará a ser (Ecuación [4-20]):

$$u_j(t) = \eta_j\left(t - \hat{t}_j\right) + \sum_i w_{ij} \cdot \sum_{t_j^{(f)}} \varepsilon_0(s) + \int_0^{\infty} k_0(s) \cdot I^{ext}(t-s) \cdot ds \qquad \text{Ecuación [4-20]}$$

Si al modelo de respuesta al pulso "*Spike Response Model*" se le incorpora un kernel adecuado, se obtiene el modelo de "*Integrate-and-Fire*". Esto implica que mediante la elección de un kernel apropiado, este modelo puede aproximar las ecuaciones del modelo de "*Hodgkin-Huxley*" con una entrada independiente del tiempo [4-34, 4-36].

4.2.3.2 Modelos basados en conductancia

Éste es el conjunto de modelos de neurona más complejo, ya que se basan en emular el intrincado funcionamiento de los canales iónicos del proceso sináptico. Como bien hemos visto, con la apertura y el cierre de los canales iónicos la conductancia de la membrana variará proporcionalmente al fenómeno anterior, este proceso puede ser descrito mediante un conjunto de ecuaciones diferenciales. Las diferencias entre los distintos modelos basados en la conductancia se derivan principalmente de la elección de los canales y de los parámetros usados en el conjunto de ecuaciones diferenciales resultantes.

4.2.3.2.1 Modelo de Hodgkin-Huxley

El modelo de Hodgkin-Huxley es comúnmente aceptado por la comunidad científica como la descripción original del modelo de conductancia de la membrana neuronal. Este modelo fue presentado por los fisiólogos Alan Lloyd Hodgkin y Andrew Huxley en el año 1952 [4-80] como resultado de sus experimentos sobre el axón gigante del calamar atlántico, lo que les valió a ambos el premio Nobel de Fisiología/Medicina en el año 1963.

Los experimentos realizados por Hodgkin y Huxley permitieron descubrir la existencia de tres tipos de corriente iónica en la membrana de una neurona, como son: la del sodio (Na), la del potasio (K) y la corriente de fuga. Las dos primeras corrientes son controladas mediante unas tensiones específicas que gestionan la apertura y el cierre de sus respectivos canales iónicos. Mientras que la tercera tiene en cuenta el resto de canales iónicos (que son primordialmente los canales de los iones "Cl^-").

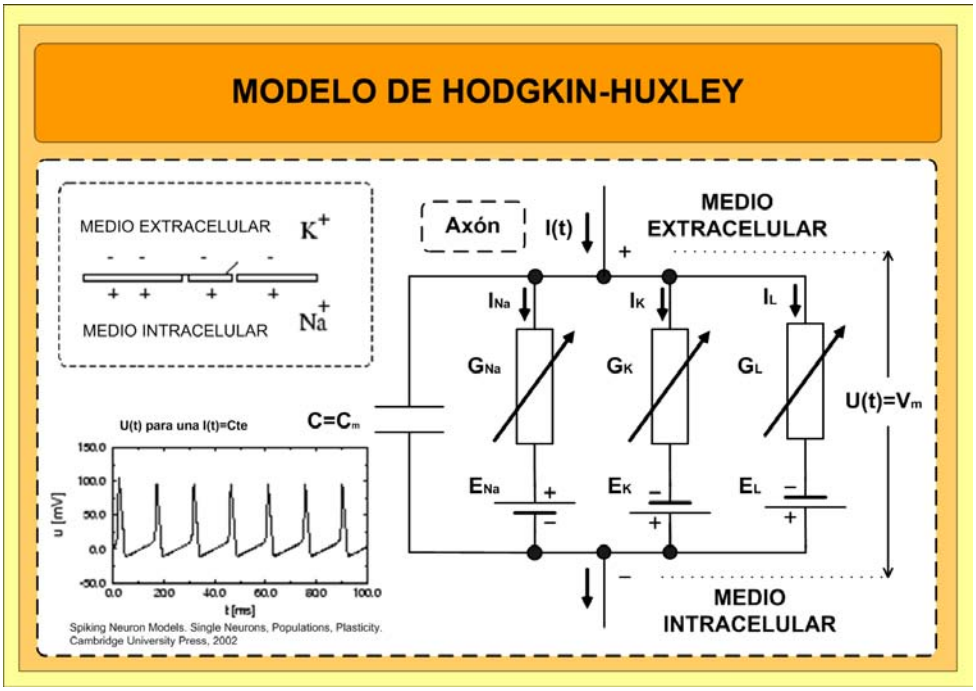

Figura 4-16: Esquema del modelo de Hodgkin-Huxley

Como bien puede apreciarse en la Figura 4-16, el modelo desarrollado por Hodgkin y Huxley se describe mediante un circuito RC, a través del cual se hace pasar una corriente "I". Este modelo se basa en modelar la membrana, que es un buen aislante, como un condensador. Además de este condensador hallamos tres resistencias en paralelo, una para cada tipo de canal iónico. Al ser los canales de sodio y del potasio dependientes de la tensión de la membrana, éstos se modelan como resistencias variables en función de dicha tensión. Al existir una diferencia entre las concentraciones iónicas dentro y fuera de la membrana, aparecerán sobre la célula una serie de potenciales de Nerst que se modelan mediante una serie de fuentes de alimentación DC. Siendo la suma de las corrientes que circula por cada una de las ramas igual a la corriente de entrada "$I(t)$", con lo cual tendremos (Ecuación [4-21]):

$$\begin{cases} I(t) = I_c(t) + \sum_{ch} I_{ch}(t) \\ I_c(t) = C \cdot \dfrac{du(t)}{dt} \end{cases} \rightarrow C \cdot \frac{du(t)}{dt} = -\sum_{ch} I_{ch}(t) + I(t) \qquad \text{Ecuación [4-21]}$$

Donde el potencial de membrana vendrá dado por "$u(t)$" mientras que "$\sum_{ch} I_{ch}(t)$" representa la suma de las corrientes iónicas que circulan a través de la membrana celular.

Para poder concretar su modelo, Hodgkin y Huxley caracterizaron experimentalmente la neurona gigante del calamar atlántico, de la cual obtuvieron la conductancia de los canales iónicos del sodio "g_{Na}", del potasio "g_k", el de pérdidas "g_L", los potenciales de equilibrio de los canales iónicos "E_{Na}, E_K, E_L" y la capacitancia de la membrana "C". Las dos primeras son dependientes de la tensión de membrana y del tiempo, mientras que la última es constante. Los parámetros que obtuvieron se presentan a continuación en la Tabla 4-3.

Tabla 4-3: Parámetros de la ecuación de Hodgkin-Huxley

Canal iónico	$E_x[mV]$	$g_x[mS/cm^2]$
Na	50.0	120.0
K	-77.0	36.0
L	-54.4	0.3
Cm [µF]	1µF/cm²	

Además de considerar los anteriores parámetros eléctricos, Hodgkin y Huxley incorporaron en su modelo tres variables más "m, n, h" dependientes de la tensión de membrana "$U(t)$" para modelar el estado de apertura y/o cierre de los canales iónicos. El parámetro "m" es el estado de apertura del canal de sodio, "n" es el estado de apertura del canal de potasio y "h" es el estado de cierre del canal del sodio. Con todo esto la suma de las corrientes iónicas pasa a escribirse como (Ecuación [4-22]):

$$\begin{cases} I_{Na}(t) = g_{Na} \cdot m^3 \cdot h \cdot (u(t) - E_{Na}) \\ I_K(t) = g_K \cdot n^4 (u(t) - E_K) \\ I_L(t) = g_L \cdot (u(t) - E_L) \end{cases} \rightarrow \begin{cases} \sum_{ch} I_{ch}(t) = g_{Na} \cdot m^3 \cdot h \cdot (u(t) - E_{Na}) + g_K \cdot n^4 (u(t) - E_K) + \\ + g_L \cdot (u(t) - E_L) \end{cases}$$

Ecuación [4-22]

De la ecuación [4-22] se deduce que los canales iónicos que estén siempre abiertos, su contribución será "$I_x = g_x \cdot u(t)$" [4-81]. Todo ello conduce a que la ecuación [4-23]

incorpore tres variables más dependientes de "$U(t)$", con lo cual tenemos una ecuación diferencial de variables dependientes.

$$C\cdot\frac{du(t)}{dt}=-\left(g_{Na}\cdot m^3\cdot h\cdot\left(u(t)-E_{Na}\right)+g_K\cdot n^4\left(u(t)-E_K\right)+g_L\cdot\left(u(t)-E_L\right)\right)+I(t)$$

Ecuación [4-23]

A su vez, Hodgkin y Huxley modelaron la dinámica de las variables (n, h, m) mediante las ecuaciones (Ecuación [4-24]):

$$\begin{cases}\dfrac{dm}{dt}=\alpha_m(u(t))\cdot(1-m(t))-\beta_m(u(t))\cdot m(t)\\ \dfrac{dn}{dt}=\alpha_n(u(t))\cdot(1-n(t))-\beta_n(u(t))\cdot n(t)\\ \dfrac{dh}{dt}=\alpha_h(u(t))\cdot(1-h(t))-\beta_h(u(t))\cdot h(t)\end{cases}$$

Ecuación [4-24]

Donde las funciones "$\alpha_x(u(t))$ y $\beta_x(u(t))$" son en realidad ecuaciones empíricas evaluadas para ajustar los resultados experimentales medidos por Hodgkin y Huxley. Dichas ecuaciones dependen del potencial de membrana y son diferentes para cada una de las variables (m, n y h), las cuales se presentan en la Tabla 4-4.

Tabla 4-4: Funciones $\alpha_x(u(t))$ y $\beta_x(u(t))$

Variable	$\alpha_x(u(t))[u/mV]$	$\beta_x(u(t))[u/mV]$
m	(2.5-0.1·u(t))/(Exp(2.5-0.1·u(t))	4·Exp(-u/18)
n	0.07·Exp(-u/20)	1/(Exp(3-0.1·u(t))+1)
h	(0.1-0.01·u(t))/(Exp(1-0.1·u(t))-1)	0.125·Exp(-u(t)/80)

El conjunto de las ecuaciones ([4-23] y [4-24]) y sus parámetros (Tabla [4-3] y Tabla [4-4]) describen el modelo de conductancia neuronal de Hodgkin-Huxley [4-82] el cual matemáticamente está constituido por un sistema de ecuaciones diferenciales en derivadas parciales, acopladas, no lineales, dependientes del espacio y del tiempo.

A fin de presentar la respuesta dinámica del sistema, se puede proceder a reescribir las ecuaciones (Ecuación [4-24]) en función de las constantes de equilibrio "m_0, n_0, h_0" y las

constantes de tiempo "τ_m, τ_n, τ_h" de las variables "m, n y h" del modelo. Con lo cual las anteriores expresiones se rescribirán como (Ecuación [4-25]):

$$\begin{cases} \dfrac{dm}{dt} = \dfrac{m_0 - m}{\tau_m} \\ \dfrac{dn}{dt} = \dfrac{n_0 - n}{\tau_n} \\ \dfrac{dh}{dt} = \dfrac{h_0 - h}{\tau_h} \end{cases} \leftarrow \begin{cases} \begin{cases} m_0 = \alpha_m(u(t)) \Big/ (\alpha_m(u(t)) + \beta_m(u(t))) \\ \tau_m = 1 \Big/ (\alpha_m(u(t)) + \beta_m(u(t))) \end{cases} \\ \begin{cases} n_0 = \alpha_n(u(t)) \Big/ (\alpha_n(u(t)) + \beta_n(u(t))) \\ \tau_n = 1 \Big/ (\alpha_n(u(t)) + \beta_n(u(t))) \end{cases} \\ \begin{cases} h_0 = \alpha_m(u(t)) \Big/ (\alpha_m(u(t)) + \beta_m(u(t))) \\ \tau_h = 1 \Big/ (\alpha_h(u(t)) + \beta_h(u(t))) \end{cases} \end{cases}$$

Ecuación [4-25]

Este modelo logra reproducir las principales características observadas para el potencial de acción (PSP). A modo de ejemplo cabe indicar que la velocidad de propagación del pulso nervioso hallada mediante este modelo es aproximadamente de "$\approx 18.8 ms^{-1}$", mientras que en el axón real es de "$\approx 21 ms^{-1}$". Otras características de las neuronas son imposibles de predecir con este modelo. Para corregir estas deficiencias, a lo largo de los años han ido surgiendo diferentes modificaciones a las ecuaciones del modelo de Hodgkin-Huxley para incorporarle otras situaciones experimentales, dando lugar a toda una serie de nuevos modelos. Estos nuevos modelos difieren de éste básicamente en la incorporación o la exclusión de canales iónicos adiciones, especialmente aquellos que tienen en cuenta el canal "Ca^{2+}" y los componentes lentos de la corriente del potasio. Como regla general, a cada canal "i" se le asignará una corriente iónica $I_i(t) = g_i \cdot x_i^{ni} \cdot (u(t) - E_i)$ [4-36, 4-76, 4-81], donde "x" es la variable que describirá la dinámica, mientras que "n_i" se corresponde con un exponente apropiado y "E_i" es su correspondiente potencial de equilibrio.

4.2.3.2.2 Modelo de FitzHugh-Nagumo

El modelo de *FitzHugh-Naguno* fue propuesto inicialmente por el fisiólogo Richard FitzHugh en el año 1961 [4-83] que lo llamó modelo "Bonhoeffer-van der Pol" ya que se basaba en una solución particular de la ecuación del oscilador de Van der Pol. Al año

siguiente (1962) dicho modelo fue implementado mediante un circuito electrónico (Figura 4-17) por J. Naguno et al. [4-84] para la descripción de un sistema excitable (una neurona).

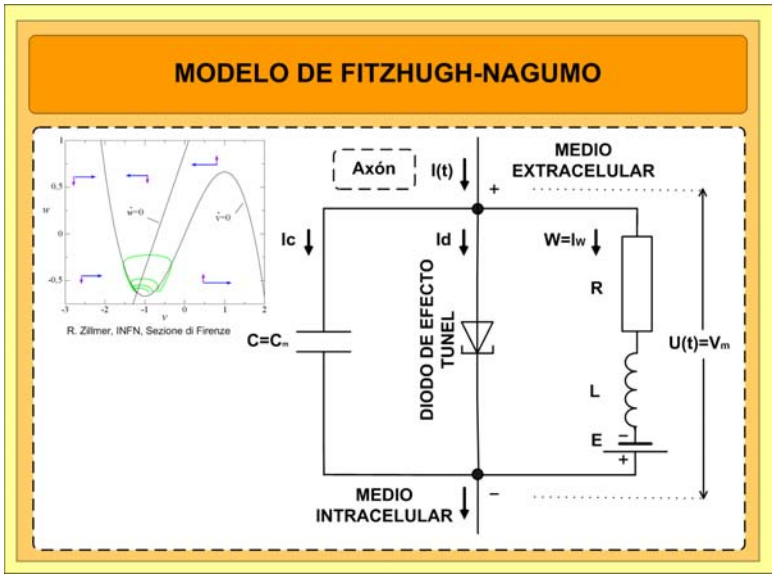

Figura 4-17: Esquema del circuito de Naguno et al.

Conceptualmente, el modelo de Fitzhugh-Naguno aisla matemáticamente las propiedades de excitación y propagación de las propiedades electroquímicas del flujo iónico del sodio y el potasio. El modelo (Ecuación [4-26]) consiste en una variable rápida que modela la tensión de membrana "$u(t)$" mediante un polinómio de tercer orden "$f(u(t))$" que permite una auto-excitación regenerativa mediante una realimentación positiva, unido esto a una variable lenta o de recuperación "w" que provee al sistema de un mecanismo de realimentación negativa.

$$\begin{cases} \dfrac{du(t)}{dt} = f(u(t)) - w(t) - I(t) \\ \dfrac{dw(t)}{dt} = a\cdot(b\cdot u(t) + d - c\cdot w(t)) \end{cases} \rightarrow \begin{cases} \dfrac{du(t)}{dt} = u(t) - u(t)^3/3 - w(t) - I(t) \\ \dfrac{dw(t)}{dt} = 0.08\cdot(u(t) + 0.7 - 0.8\cdot w(t)) \end{cases}$$

Ecuación [4-26]

Como podemos apreciar en la Ecuación [4-26], el modelo propuesto viene a ser una simplificación del propuesto por Hodgkin-Huxley (HH) que describe la dinámica de activación y desactivación de una neurona pulsante. En éste se han eliminado las variables

“m” y ”n” que modelan la apertura de los canales iónicos del sodio y potasio, introduciendo una nueva variable lenta “v” o variable de recuperación que tiene un función similar a la variable “h” del modelo (HH). De esta forma, un sistema de 4 dimensiones se simplifica en uno de dos, lo que permite presentar una explicación geométrica del fenómeno biológico relacionado con la excitación neuronal y el mecanismo de generación de pulsos nerviosos, por ello, es uno los modelos más comúnmente usados. De esta forma se reproducen la mayoría de características cualitativas de los pulsos eléctricos a través de los nervios y las fibras cardiacas, así como la existencia de niveles de disparo de la excitación, los períodos refractarios, tanto absolutos como relativos, y la generación de trenes de pulsos bajo la inyección de una corriente externa “I”.

4.2.3.2.3 Modelo de Morris-Lecar

El *modelo de Morris-Lecar* es un modelo de neurona biológica desarrollado por Cathy Morris y Harold Lecar en el año 1981 [4-83] para reproducir la variedad de comportamientos oscilantes relacionados con la conductancia de los cationes “Ca^{++}” (que provocan una corriente despolarizadora en la membrana) y “K^{+}” (que representa una corriente hiperpolarizadora de la membrana) en las fibras musculares del percebe gigante.
Dicho modelo consiste en un sistema bidimensional de ecuaciones diferenciales no lineales, que se considera como una simplificación del modelo cautridimensional de Hodgkin-Huxley. Este modelo es uno de los modelos más extendidos y usados por la neurociencia computacional.
Cualitativamente el modelo *Morris-Lecar* describe la relación entre el potencial de membrana “$u(t)$” y la activación de los canales iónicos dentro de la membrana celular. Con lo cual, este modelo implementa un mecanismo de respuesta para la excitación basado en una dependencia de la conductancia del canal “Ca^{++}” de la tensión de membrana “$u(t)$”, a la vez que incorpora un mecanismo lento de recuperación basado en la variación de la conductancia del canal “K^{+}”. Las expresiones (Ecuación [4-27]) que describen el modelo son las siguientes:

$$\begin{cases} C \cdot \dfrac{du(t)}{dt} = -g_L \cdot (u(t) - E_L) - g_{Ca} \cdot M_{ss} \cdot (u(t) - E_{Ca}) - g_K \cdot w(t) \cdot (u(t) - E_K) + I(t) \\ \dfrac{dw(t)}{dt} = \dfrac{(w_{ss}(u(t)) - w(t))}{\tau_w(u(t))} \end{cases}$$

Ecuación [4-27]

Donde "$u(t)$" se corresponde con el potencial de membrana, "$w(t)$" es la variable de recuperación, la cual es casi invariante a la conductancia normalizada del canal "K^+", mientras que "$I(t)$" se corresponde con la corriente de estimulación externa recibida. La variable relacionada con la conductancia normalizada "$w(t)$" es igual al valor instantáneo de la probabilidad a que el canal iónico "K^+" se halle abierto (en conducción), la segunda ecuación describe el proceso de relajación mediante el cual los canales proteicos experimentan transiciones conformacionales entre el estado iónico conductor y el no conductor. La clave de la excitabilidad eléctrica se halla en el hecho que las energías y las tasas de transición para el control de la apertura de los canales son fuertemente dependientes de la tensión de membrana.

Por otro lado, "$M_{ss}(u(t))$" y "$W_{ss}(u(t))$" son las funciones de probabilidad del estado de apertura determinadas a partir de la suposición que en el equilibrio los estados de apertura/cierre de los diversos canales se reparten de acuerdo a una distribución de Blotzman, mientras que "$\tau_w(u(t))$" representa la constante de tiempo de relajación del canal "K^+" en función de los cambios de tensión. Las expresiones que definen estas magnitudes se presentan a continuación (Ecuación [4-28]):

$$\begin{cases} M_{ss}(u(t)) = \dfrac{\left(1+Tanh\left(\left(u(t)-V_1\right)/V_2\right)\right)}{2} \\ W_{ss}(u(t)) = \dfrac{\left(1+Tanh\left(\left(u(t)-V_3\right)/V_4\right)\right)}{2} \\ \tau_w(u(t)) = \tau_0 \cdot Sech\left(\left(u(t)-V_3\right)/2 \cdot V_4\right) \end{cases} \rightarrow \begin{cases} \text{Siendo } V_1, V_2, V_3, V_4 \text{ y } \tau_0 \text{ los parámetros de ajuste} \\ \text{del estado estacionario y de la constante de tiempo} \\ \text{del sistema.} \end{cases}$$

Ecuación [4-28]

El modelo de Morris-Lecar es especialmente interesante para el modelado de neuronas pulsantes rápidas como pueden ser las neuronas piramidales del neocórtex.

4.2.3.2.4 Modelo de Hindmarsh-Rose

El modelo de Hindmarsh-Rose [4-86, 4-87] es en esencia un modelo de neurona que emula el comportamiento biológico de las neuronas del tálamo. Este modelo emula el rápido aumento y posterior descenso del potencial de membrana observado en los experimentos realizados sobre neuronas aisladas. En este caso, las variables relevantes son el potencial de membrana "$u(t)$", y dos variables más "$v(u(t))$ y $w(u(t))$" que vienen a representar el transporte iónico a través de la membrana para los diferentes canales iónicos. El transporte de iones de sodio y potasio se realiza a través de canales iónicos rápidos, la velocidad de los cuales se modela mediante la variable "$v(u(t))$" que se conoce como la variable de pulso, mientras que el transporte de iones a través de canales más lentos se modela a través de la variable "$w(u(t))$", que se conoce con el nombre de variable de estallido. Por lo tanto, el *modelo de Hindmarsh-Rose* se compone de tres ecuaciones diferenciales ordinarias no lineales adimensionales como se muestra en la Ecuación [4-29]:

$$\left(\begin{cases}\dfrac{du(t)}{dt} = v + F(u) - w + I(t)\\ \dfrac{dv(t)}{dt} = G(u) - v\\ \dfrac{dw(t)}{dt} = r\cdot(H(u) - w)\end{cases}\right. \rightarrow \begin{cases}\dfrac{du(t)}{dt} = v + 3\cdot u^2 - u^3 - w + I(t)\\ \dfrac{dv(t)}{dt} = 1 - 5\cdot u^2 - v\\ \dfrac{dw(t)}{dt} = 10^{-3}\cdot\left(4\cdot\left(u - {}^{8}\!/\!_{5}\right) - w\right)\end{cases}$$
$$\left.\begin{cases}F(u) = a\cdot u^2 - b\cdot u^3\\ G(u) = c - d\cdot u^2\\ H(u) = s\cdot(u - u_0)\end{cases}\right)$$

Ecuación [4-29]

La ecuación diferencial de la variable de estallido permite implementar una gran variedad de comportamientos dinámicos de la membrana de la neurona, incluyendo toda una serie de comportamientos impredecibles como pueden ser su dinámica caótica. De esta forma, el modelo de Hindmarsh-Rose es muy útil al ser un modelo relativamente simple que obtiene unos resultados cualitativamente bastante buenos si los comparamos con los valores empíricos.

4.2.3.2.5 Modelo de Izhikevich

El modelo de neurona de Izhikevich [4-88] se presentó en el año 2003 para ser una alternativa real a los modelos basados en "*integrate-and-fire*". El modelo se compone de dos variables, una que representa el potencial de la membrana y la otra el mecanismo de recuperación de la membrana (mediante la activación de las corrientes de potasio y la desactivación de las corrientes de sodio).

En este tipo de neurona pulsante, cuando la tensión de la membrana supera un determinado nivel de disparo, se emite un pulso, lo que provoca que las variables de tensión y de recuperación se relajen a un valor predeterminado. Las expresiones usadas (Ecuación [4-30]) para describir este modelo son las siguientes:

$$\begin{cases}\dfrac{du(t)}{dt} = a\cdot(b\cdot v - u) = 0.02\cdot(0.2\cdot v - u)\\ \dfrac{dv(t)}{dt} = 0.04\cdot v^2 + 5\cdot v + 140 - u + W\end{cases}$$

Ecuación [4-30]

Donde “a” y “b” son dos parámetros abstractos del modelo, “W” representa las entradas ponderadas de la neurona, “v(t)” representa la activación de la neurona y “u(t)” es la variable de recuperación. La tensión de membrana se obtiene integrando estas dos ecuaciones diferenciales (Ecuación [4-30]). En el caso que la tensión de membrana supere el nivel de disparo (prefijado a 30mV) las variables “u” y “v” se relajarán a los siguientes valores (Ecuación [4-31]):

$$v \geq 30mV \rightarrow \begin{cases} v \rightarrow c = -65 \\ u \rightarrow u + d = w + 8 \end{cases} \rightarrow \begin{cases} \text{Donde "c" y "d" son los parámetros del} \\ \text{modelo o sistema.} \end{cases}$$

Ecuación [4-31]

La principal ventaja de este modelo de neurona, es su capacidad de exhibir cualquier patrón de disparo característico de cualquiera de los tipos de neuronas corticales conocidas, eso si, mediante la correcta elección de los parámetros (a, b, c, d) [4-88].

4.2.3.3 Modelos compartimentales

Desde la primera descripción de una neurona realizada por Ramón y Cajal en el año 1903 [4-89], el conocimiento sobre la complejidad de las estructuras nerviosas se ha ampliado enormemente. A fin de capturar la compliejidad inherente de los sistemas nerviosos se han desarrollado los modelos compartimentales, los cuales toman en consideración la estructura espacial del árbol de dendritas y modelan el proceso de transmisión sináptica con gran detalle. Mediante estos métodos es posible considerar otras corrientes iónicas, además de las corrientes del sodio “ Na^+ ” y el potasio “ K^+ ” incorporadas en el modelo de Hodgkin-Huxley, como pueden ser las del Ca^{++} [4-36]. Cabe remarcar que existen gran cantidad de publicaciones dedicadas a este punto, pero al no ser nuestro objetivo el presentar una descripción completa de los diferentes modelos de neurona existentes, tan sólo introduciremos una serie de conceptos básicos a cerca de estos modelos.

Como ya hemos descrito en anteriores apartados, las dendritas reciben las señales sinápticas de las otras neuronas. Estas señales de entrada activan los canales iónicos que crean a su vez un potencial post-sináptico que es propagado hasta el cuerpo celular de la neurona o soma, donde los canales iónicos activados por dicha tensión crean los potenciales de acción. Generalmente, éstos se hallan ubicados en la región llamada axón de Hillock

ubicado justo en la base del soma [4-33, 4-49]. El axón es el encargado de propagar los potenciales de acción a todas sus ramas terminales donde se forma la conexión sináptica con otras neuronas.

La idea básica sobre la que se sustentan los modelos compartimentales es la de dividir todos estos componentes en pequeños compartimentos modelados mediante las ecuaciones que describen su circuito eléctrico equivalente. El uso de las ecuaciones diferenciales apropiadas para cada uno de estos compartimentos permitirá su simulación así como la descripción de las interacciones con los otros compartimentos. El concepto de un circuito eléctrico equivalente para cada pequeña parte de la membrana celular de la neurona es la base de los modelos compartiméntales [4-90].

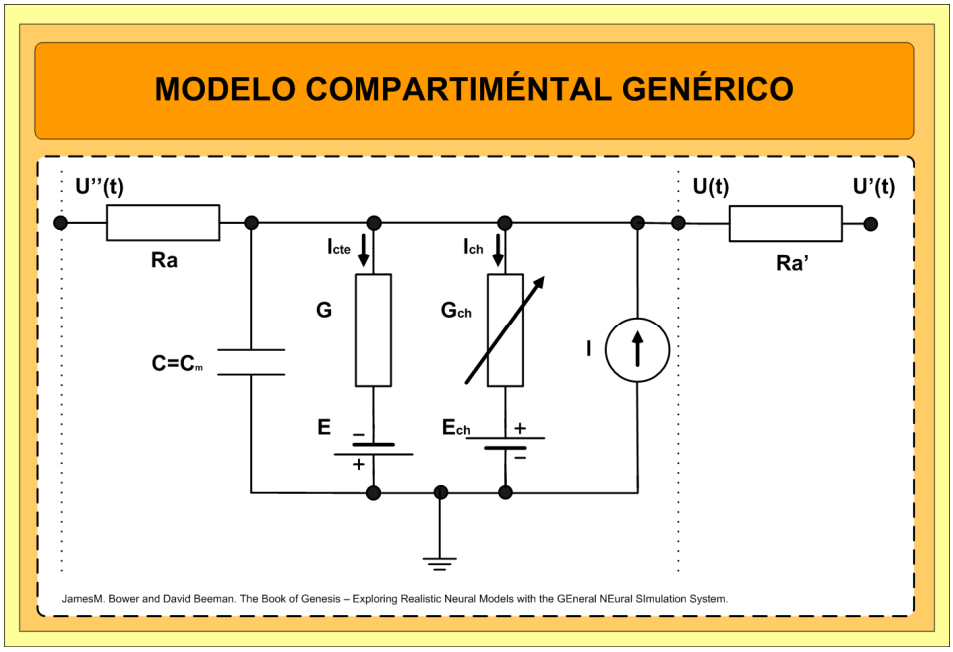

Figura 4-18: Circuito equivalente genérico de un compartimento neuronal

Uno de estos circuitos equivalentes se presenta en la Figura 4-18, donde “u” representa el potencial de membrana que aparece sobre la membrana de capacidad “C”, que a su vez causa la aparición de un flujo de corriente a través de la resistencia axial “ R_a ” hacia el interior del compartimento cuando existe una diferencia de potencial entre compartimentos

dada por $\left(u-u^{''}\right)$. "G_{ch}" es una resistencia variable cuyo valor viene gobernado por la conductancia de un canal iónico específico. A su vez, el correspondiente potencial de equilibrio del canal iónico se halla representado por una fuente DC en serie con la conductancia. Las otras resistencias representan la conductancia de pérdidas "G" y su potencial de equilibrio asociado "E" que toma un valor generalmente cercano a "$u_{\mathrm{Re}\,st}$". Mientras que "I" representa la señal externa de entrada a la neurona. Con todo ello dicho circuito se regirá por la ecuación [4-32]:

$$C\cdot\frac{\mathrm{du(t)}}{\mathrm{dt}}=(\mathrm{E}-\mathrm{u})\cdot G+\sum_{ch}\left(E_{ch}-u\right)\cdot G_{ch}+\frac{\left(u^{'}-u\right)}{R_a^{'}}+\frac{\left(u^{''}-u\right)}{R_a^{'}}+I(t)\quad \text{Ecuación [4-32]}$$

Dicha ecuación describe un modelo compartimental lineal. Cabe remarcar que existen otros modelos compartimentales mucho más complejos que permiten obtener conductancias no lineales para las diferentes corrientes iónicas, así como modelar las corrientes sinápticas con mucho mayor nivel de detalle.

4.2.4 CODIFICACIÓN DE LA INFORMACIÓN EN LAS NEURONAS PULSANTES

Una vez descritos los modelos formales más relevantes en el campo de las redes neuronales pulsantes queda aun por dilucidar la cuestión referente a cómo las neuronas codifican la información en los trenes de pulsos que emiten, y sobre todo cómo esta información es decodificada por el resto de neuronas. Ésta es una de las cuestiones fundamentales de la neurofisiología que aun permanecen abiertas [4-90] siendo esta cuestión abordada en multitud de publicaciones [4-91, 4-92].

Una secuencia o tren de pulsos puede contener información basada en diferentes esquemas de codificación. Por ejemplo, en las neuronas motoras la intensidad con la que se flexiona un músculo inervado depende únicamente de la tasa de disparo, o lo que es lo mismo el número medio de disparos por unidad de tiempo (un código basado en una tasa). En el otro extremo hallamos los complejos "códigos temporales" basados en el instante preciso en el

que se dan los pulsos temporales, los cuales están fijados a un determinado suceso externo, como sucede en el sistema auditivo [4-34].

Delimitar claramente los diferentes esquemas de codificación de la información no es nada sencillo [4-73], en la práctica hay establecidas al menos tres metodologías de codificación de la información:

- Las basadas en la codificación por tasa, donde la información se codifica mediante la tasa de disparo de las neuronas [4-34, 4-76].
- Las que se sirven de la codificación temporal, siendo la información codificada mediante el instante de emisión de los pulsos [4-34, 4-93]
- Las basadas en la codificación poblacional donde la información se codifica mediante la actividad de las diferentes poblaciones de neuronas, pudiendo una neurona participar a la vez de diferentes poblaciones [4-34, 4-75].

El uso de una u otra codificación por las neuronas reales es actualmente un tema de intenso debate en el campo de las neurociencias, por lo tanto, a continuación pasaremos a describir en detalle cada una de estas codificaciones.

4.2.4.1 Codificación por tasa de disparo

La *codificación por tasa de disparo* es el esquema tradicional por antonomasia de la neurociencia, el cual, se basa en la suposición que la mayoría de la información sobre un determinado estímulo se halla contenida en la tasa de disparo.

El concepto de la *codificación por la tasas de disparo* se remonta al trabajo de E.D. Adrian e Y. Zotterman realizado en el año 1926 [4-94] donde se presentaba como a medida que la tasa de disparo aumentaba en las neuronas receptoras ubicadas en los músculos, la fuerza aplicada por éstos aumentaba. Todo ello permitía tratar las respuestas neuronales de forma estadística.

En las décadas posteriores, la medición de las tasas de disparo se convirtió en la herramienta estándar para describir las propiedades de todos los tipos de neuronas sensoriales o corticales, en parte debido a la relativa facilidad de medición experimental de dichas tasas. Sin embargo, este enfoque elimina toda la información relativa al instante

exacto en el cual se ha disparado un determinado pulso aunque es extremadamente inmune al ruido. Debido a esto, a lo largo de los últimos años han empezado a surgir más y más evidencias experimentales que sugieren que el concepto de tasa de disparo, basada en un promedio temporal, puede ser demasiado simplista para describir la actividad cerebral [4-95].

Bajo el *concepto de tasa de disparo* se han desarrollado diferentes metodologías, cada una de ellas basada en una forma específica de promediado, tal y como se presenta en los siguientes subapartados. Cabe destacar que en la *codificación por tasa de disparo* los diferentes mecanismos de aprendizaje se basan en la modificación de los pesos que dependen de la actividad sináptica.

4.2.4.1.1 Tasa del conteo de pulsos

La tasa de conteo de pulsos "*Spike-Count Rate*" o promedio temporal "υ" (Figura 4.19), se obtiene contando el número de pulsos monitorizados durante un ensayo "$\mathrm{n}_{\mathrm{sp}}(T)$" y dividiéndolo por la duración de dicho ensayo "T" (Ecuación [4-33]).

$$\upsilon = \frac{\mathrm{n}_{\mathrm{sp}}(T)}{T} \ [\mathrm{Hz}] \qquad \text{Ecuación [4-33]}$$

La selección de la longitud temporal de la ventana "T" dependerá del tipo de neurona a examinar y del tipo de estímulo usado, los valores típicos usados son T = 100ms o T = 500 ms, pero dicha duración puede ser modificada. La tasa de conteo de pulsos puede determinarse a partir de un único ensayo, pero a costa de perder toda resolución temporal sobre las variaciones en la respuesta neural a lo largo de dicho período.
Esta metodología de promediado temporal funciona correctamente en los casos en que el estímulo aplicado sea constante o varíe lentamente, y se requiere de la reacción del sistema que no sea rápida. No obstante, en el mundo real las entradas no se suelen mantener estacionarias, ya que éstas suelen variar a menudo muy rápidamente.

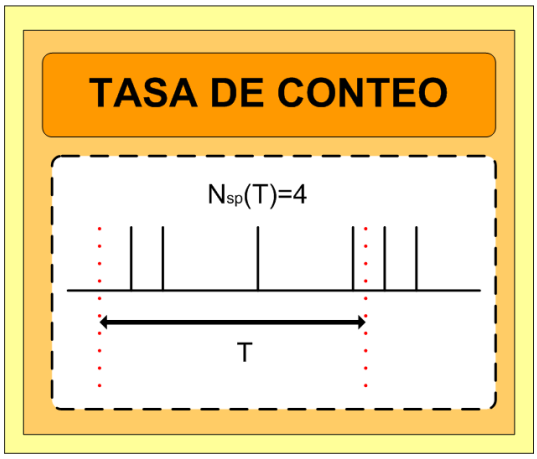

Figura 4-19: Tasa de conteo

A pesar de sus deficiencias, el concepto de tasa de conteo de pulsos es ampliamente usado no sólo en los experimentos sino también en los modelos de redes neuronales.

4.2.4.1.2 Tasa de disparo dependiente del tiempo

La *tasa de disparo dependiente del tiempo* "υ" se define como el promedio de pulsos (promediado sobre un número "n" de ensayos lo suficientemente grande) que se obtiene durante un corto intervalo de tiempo delimitado entre "t" y "$t+\Delta t$" dividiéndolo por dicho intervalo de tiempo (Ecuación [4-34]), obteniendo así una densidad de pulsos "$\rho_{n_{sp}}(t)$", que si la integramos a su vez entre dicho intervalo se obtiene la tasa de disparos promedio:

$$\upsilon = \int_{t}^{t+\Delta t} \rho_{n_{sp}}(t)\cdot dt = \frac{1}{\Delta t}\cdot \int_{t}^{t+\Delta t} \frac{\sum_{i=1}^{n} n_{sp}(t)}{n}\cdot dt \qquad [\text{Hz}] \qquad \text{Ecuación [4-34]}$$

Esta tasa es efectiva para los procesos basados en estímulos estacionarios, así como para los basados en estímulos que fluctúan con el tiempo. El intervalo "Δt" debe ser lo suficientemente grande (generalmente tiene una duración de varios milisegundos) para que se den un número lo suficientemente grande de pulsos, a fin que la estimación de la media sea fiable.

El principal inconveniente de este método radica en que no puede ser el esquema de codificación utilizado por las neuronas biológicas puesto que éstas no esperan a que se hayan dado “n” secuencias de sucesos semejantes para presentar su salida.
Sin embargo, la medida experimental de la tasa de disparos en función del tiempo puede tener sentido si nos hallamos en presencia de grandes poblaciones de neuronas independientes que reciben el mismo estímulo. En este caso, la solución experimental más simple consistirá en monitorizar una única neurona representativa de la población durante “n” ensayos. Por lo tanto, la codificación de la tasa de disparo en función del tiempo se basa en la suposición implícita que se está trabajando siempre con poblaciones de neuronas [4-96].

4.2.4.2 Codificación temporal

En el supuesto que la transferencia de una determinada información neuronal requiera del conocimiento preciso del instante en el que se produce el disparo, a dicho código se le identificará como un código temporal [4-97]. Si unimos este hecho a que un gran número de estudios científicos han fijado la resolución temporal de un código neuronal en alrededor de la milésima de segundo, podemos afirmar que el instante de disparo de un pulso es un elemento relevante en la codificación neuronal de la información [4-98, 4-99].

Los códigos temporales usan las características de la actividad pulsante que no puede ser descrita mediante la tasa de disparo. En el sistema nervioso no existe ninguna referencia de tiempo absoluta y por tanto la información es transmitida mediante la temporalización relativa entre los pulsos en una población de neuronas o con respecto a la oscilación característica del sistema nervioso central [4-98, 4-95]. A diferencia de la codificación por tasa, el modelo de codificación temporal intenta dar cuenta de los estímulos que producen un pequeño número de pulsos y son usados para la toma de decisiones rápidas, en las que las células de respuesta no tienen suficiente tiempo para integrar dicha tasa. Por ejemplo, si una neurona es capaz de disparar a una velocidad máxima de 100 pulsos por segundo, un estímulo de duración inferior a 10 ms probablemente tan sólo provocaría un pulso. Este

hecho puede explicar fenómenos neurológicos como la localización del origen de un sonido que se produce en el cerebro en un tiempo del orden de milisegundos.

La estructura temporal de un tren de pulsos evocado por un estímulo se determina mediante la dinámica del estímulo y la naturaleza del proceso de codificación neural. Cuando los estímulos cambian rápidamente tienden a generar pulsos en un instante de tiempo muy preciso, y a su vez producen un cambio rápido en las tasas de disparo independientemente de la estrategia de codificación usada.
Para la codificación temporal se requiere de una precisión temporal en la respuesta la cual no se deriva exclusivamente de la dinámica de los estímulos, pero que, sin embargo se refiere a las propiedades de éstos. La interacción entre el estímulo y la dinámica de codificación dificulta la identificación de un código temporal.

La codificación temporal es distinta e independiente para cada pulso. Por lo tanto, si cada pulso es independiente de los demás pulsos del tren, la característica temporal de la codificación neuronal vendrá determinada por el comportamiento de la frecuencia de disparo a su vez dependiente del tiempo "r(t)". Si el código "r(t)" fluctúa muy lentamente, a dicha codificación se la conoce como *código de tasa*, mientras que si fluctúa muy rápidamente se conoce como *código temporal*.

Los códigos de disparo de fase son un tipo de codificación que asigna una etiqueta de tiempo para cada pulso de acuerdo a una referencia de tiempo basada en la fase de oscilaciones locales en curso de baja frecuencia [4-100] o de alta frecuencia [4-101]. Una de las características de esta codificación es que las neuronas se adhieren a un orden de preferencia de emisión, lo que resulta en la aparición de una secuencia de disparo [4-102].

En el caso de la codificación temporal, el aprendizaje puede explicarse mediante la modificación de la actividad dependiente del retardo de la sinapsis [4-103]. Dichas modificaciones no sólo dependen de la tasa de pulsos sino también de lo patrones temporales de pulso, es decir, puede definirse como un caso de temporalización de pulsos

dependiente de la plasticidad donde la eficiencia sináptica es modulada por el instante preciso de los pulsos. Ésto puede considerarse como un indicio que los códigos temporales son usados para la transmisión de información cortical.

4.2.4.3 Codificación poblacional

La codificación poblacional representa los estímulos mediante las actividades conjuntas de un determinado número de neuronas. En esta codificación cada neurona tiene asociada una distribución de respuestas sobre un conjunto de entradas, donde las respuestas de muchas neuronas pueden combinarse para determinar un valor sobre las entradas.

La codificación poblacional capta los rasgos esenciales de codificación neural, pero a su vez es lo suficientemente simple para permitir su análisis teórico [4-104]. Diversos experimentos han puesto de manifiesto que el paradigma sobre el que se sustenta esta codificación es usada ampliamente en las áreas sensoriales y motoras del cerebro. Por ejemplo, en el área de la visión del lóbulo temporal las neuronas se hallan sintonizadas en una determinada dirección del movimiento [4-105]. Por lo tanto, si un objeto se mueve en una determinada dirección, esta dirección se determinará mediante el patrón de actividad de la población de neuronas.

Si idealizamos la situación, y estudiamos una población de neuronas con propiedades idéntica, en la cual todas las neuronas de la población tienen asociado el mismo patrón de conexiones de entrada y de salida; Los pulsos de las neuronas de dicha población “m” son enviados a otra población de neuronas “n”. Con ello, en esta imagen idealizada, cada neurona de la población “n” recibe la contribución de todas las neuronas de la población “m”, donde la fracción de neuronas activas presinápticas de la población “m” es la información relevante desde el punto de vista de la neurona receptora. Definiremos la actividad de la población “ $A(t)$ ” mediante la Ecuación [4-35]:

$$A(t) = \frac{1}{\Delta t} \cdot \frac{n_{act}(t; t+\Delta t)}{N} = \frac{1}{\Delta t} \cdot \frac{\int_{t}^{t+\Delta t} \sum_{j} \sum_{f} \delta\left(t - t_j^{(f)}\right) dt}{N}, \text{ [Hz]} \qquad \text{Ecuación [4-35]}$$

Siendo “N” el tamaño de la población y “$n_{act}(t;t+\Delta t)$” el número de pulsos sumados sobre todas las neuronas de la población que se han dado entre “$\{t,t+\Delta t\}$”. La codificación poblacional tiene toda una serie de ventajas como pueden ser la reducción de la incertidumbre debido a la variabilidad neuronal y la capacidad para representar diferentes atributos de los estímulos a la vez. Esta codificación es también mucho más rápida que la codificación por tasa de disparo y permite reflejar los cambios en las condiciones de estímulo casi instantáneamente [4-96,4-106].

4.2.4.3.1 Codificación posicional

Una codificación posicional típica involucra el uso de neuronas con una función de ajuste gaussiana cuya media varía linealmente en función de la intensidad del estímulo. Esto conlleva que dicha neurona responderá con una tasa de pulsos mayor en función de lo cerca que ésta se halle de la media de la distribución gaussiana. Sin embargo, el ruido inherente a las respuestas neuronales aconsejan la estimación de una función de máxima verosimilitud.

Esta codificación se aplica a variables continuas, como pueden ser: las encargadas del posicionamiento de las articulaciones, posicionamiento del ojo, la determinación de colores o la determinación de frecuencia acústica. Cualquier neurona individual es demasiado ruidosa para codificar fielmente dichas variables mediante una codificación de tasas de disparo, por lo tanto, el uso combinado de toda una población de neuronas permite asegurar una mayor precisión en dichas estimaciones.

4.2.5 EVOLUCIÓN HISTÓRICA DE LAS REDES NEURONALES PULSANTES

El estudio de las redes neuronales pulsantes se inició el año 1981 cuando Cristopher Von der Malsburg, en el Instituto Max-Plank realizó un trabajo [4-12] en el cual se describía un nuevo tipo de red neuronal, basada en un nuevo modelo de neurona que se parecía mucho más a las neuronas biológicas. A este tipo de neuronas se las denominó “*Spiking Neurons*” o neuronas pulsantes. Este nuevo modelo de neurona tenía el potencial de ayudar a resolver el problema de la interconexión neuronal. Concretamente, en este primer trabajo se

indicaba que a fin de señalar los enlaces entre neuronas que codifican las características de un determinado objeto, dichas neuronas deberían sincronizar el momento en que emiten la señal pulsante de salida "La hipótesis de la sincronización". Por lo tanto, las neuronas que codifican las características pertenecientes a un determinado objeto se disparan en fase, permitiendo así la representación de múltiples objetos mediante una misma red.

A su vez, en el año 1989 los investigadores C.M. Gray, P. Konig, A.K. Engel y W. Singer en el *Max-Planck-Institute for Brain Research* de Frankfurt (Alemania) realizaron un estudio [4-43] con gatos, en el cual se descubrieron aparentemente un ensamblado de correlaciones entre neuronas dependientes. Este trabajo fue interpretado por la comunidad científica como una constatación que los trabajos de Von der Malsburg estaban bien encaminados, lo que se tradujo en un aumento del interés por parte de la comunidad científica en el estudio de las neuronas pulsantes "*spiking neurons*". Especialmente estudiada ha sido la *hipótesis de la sincronización* por los investigadores Michael Shadlen y J. Movshon de la Universidad de New York (USA) [4-44], que en 1999 pusieron de manifiesto que la precisión con que los picos individuales son emitidos por las neuronas biológicas pueden ser muy altas. De esta forma demostraron la plausibilidad que mediante los instantes de emisión de los pulsos individuales se pudiera transmitir información entre diferentes neuronas.

Todas estas evidencias condujeron a la formulación de modelos mucho más refinados de computación neuronal pulsante que se concretaron en el año 1996, con el trabajo científico de Wolfgang Maass de la Universidad de Gratz (Austria) [4-45], en el cual se describía en detalle una red neuronal asíncrona pulsante "*Asynchronous Spiking Neural Network*" (ASNN's). Mediante este nuevo tipo de red, se modelaba en detalle el proceso preciso de creación del potencial de acción (AP) "*Action Potential*" de una neurona pulsante. Al año siguiente (1997) Wolfgang Maass mostró en otro trabajo [4-38] que las entradas para dichas neuronas pulsantes consisten en un conjunto de pulsos asincrónicamente temporizados, donde la precisa evaluación del instante de emisión del pulso de salida se puede interpretar como un proceso de computación de las entradas. Teóricamente las neuronas pulsantes tienen un gran potencial para la realización de procesos de computación complejos. Cabe remarcar que se han desarrollado trabajos científicos [4-38] y [4-34] en los

cuales se demuestra que su capacidad de computación es muy superior a las de las neuronas sigmoidales (basadas en funciones de activación no lineales) de segunda generación.

4.2.6 DESARROLLOS PROPIOS DE REDES NEURONALES PULSANTES

Como ya hemos descrito en capítulos anteriores, el desarrollo de soluciones hardware eficientes en el campo de la implementación de sistemas neuronales es uno de los grandes retos actuales de la ciencia y la tecnología. Las redes neuronales poseen inherentemente una capacidad intrínseca de computación en paralelo, la cual solamente se puede explotar en su totalidad mediante implementaciones hardware (VLSI y/o FPGA).

Este es el motivo por el cual la gran parte de sistemas de evaluación de problemas complejos (como los sistemas de reconocimiento de patrones en tiempo real [4-108]), son intratables por los sistemas digitales tradicionales (procesadores) en un tiempo razonable [4-107] a menos que se implementen mediante soluciones hardware (VLSI y/o FPGA).

La forma de procesamiento que implementan las redes neuronales es totalmente distribuida, lo que las hace tolerantes a fallos e inmunes al ruido [4-109] si las comparamos con los sistemas digitales secuenciales tradicionales.

A pesar de todas estas ventajas, a la hora de tratar con los sistemas neuronales existe un gran escollo que es el de conseguir los parámetros óptimos de configuración de la red; esto se debe a que la complejidad de ésta no crece linealmente con el número de conexiones. Por lo tanto, el desarrollo de estrategias de aprendizaje que permitan obtener de forma rápida soluciones óptimas para el espacio de configuración en redes neuronales extensas es uno de los ámbitos de mayor interés en este campo [4-17, 4-18, 4-32, 4-69].

Particularmente, en la última década se han realizado grandes esfuerzos en el desarrollo de las redes neuronales pulsantes "*Spiking Neural Networks*" (SNN)) [4-34] que intentan emular el funcionamiento real de las redes neuronales biológicas (tal y como se ha descrito en los anteriores subapartados). En dichas redes la información se codifica en forma de pulsos de tensión que reciben el nombre de *potenciales de acción* "*Action Potentials*" (AP).

Cada neurona pulsante se encarga de integrar los diferentes potenciales de acción (AP) en una variable llamada Potencial Post Sináptico "*Post Synaptic Potential*" (PSP) que hace las

funciones del potencial de membrana de las neuronas biológicas. Como bien se ha visto, este Potencial Post Sináptico depende del tiempo, ya que decae mientras no existan potenciales de acción a integrar. La principal característica de las neuronas pulsantes es su capacidad de proporcionar una salida (emisión de un potencial de acción (AP)). En el momento en el que el potencial post-sináptico sobrepasa un determinado valor umbral "*Threshold*", el potencial de acción (AP) de salida se transmite al resto de neuronas a las cuales se haya conectada.

A lo largo de los siguientes apartados se presentan diferentes implementaciones prácticas de redes neuronales pulsantes (SNN) que hemos desarrollado. Inicialmente se presenta un modelo de red neuronal pulsante digital, basada en un modelo de neurona *Threshold-and-Fire*. Sobre ésta, hemos desarrollado un algoritmo de aprendizaje basado en algoritmos genéticos [4-110]. A la vez, también se presenta el diseño e implementación hardware de una neurona pulsante mixta (analógico-digital), a fin de disponer de una neurona pulsante con las mejores características de ambos mundos [4-111].

4.2.6.1 Desarrollo de una arquitectura digital de SNN

En este sub-subapartado se presenta una aportación propia en el campo de las redes neuronales pulsantes. Ésta ha consistido en la implementación digital de una neurona pulsante basada en el modelo "Integrate & Fire". Posteriormente, presentamos una metodología de auto-aprendizaje especialmente diseñada para la implementación de sistemas de reconocimiento de patrones temporales.

4.2.6.1.1 Neurona Digital Pulsante

A continuación (Figura 4-20), presentamos el diseño de un modelo simplificado de neurona digital [4-110].

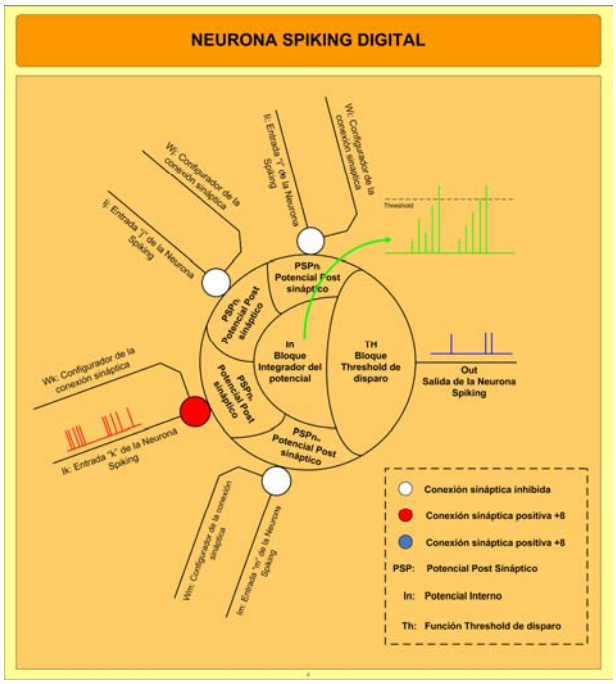

Figura 4-20: Descripción de una neurona pulsante digital

La neurona digital realizada se enmarca en un modelo "*Threshold-And-Fire*" [4-34], cuyo máximo exponente es el modelo "*Integrate & Fire*". Se ha elegido como esquema de codificación de la información el basado en la tasa de disparo, al ser el más usado y genérico de todos los esquemas de codificación. A continuación, se presenta un modelo simplificado de neurona pulsante digital (Figura 4-20) desarrollado para emular la dinámica del potencial de membrana y la generación de los potenciales de acción de una neurona biológica.

En el modelo desarrollado, la sinapsis entre neuronas puede configurarse de tres formas (inhibitoria, deshabilitada y excitatoria) siendo los factores de peso asociados a la sinapsis prefijados para cada una de las formas. Esta simplificación se ha realizado a fin de acotar las posibles configuraciones de la neurona. Además, el potencial de membrana "u(t)", generado a partir de los diferentes potenciales postsinápticos recibidos se ha modelado para que decaiga de forma lineal y no de forma exponencial como sucede en las neuronas biológicas [4-34]. Esta aproximación simplifica el hardware requerido, pero no implica una variación significativa del comportamiento cualitativo del modelo con respecto al de las

neuronas reales. Cuando el potencial de membrana supera un determinado valor digital de disparo (el cual es fijo) la neurona emite un potencial de acción de salida (AP). Otra de las simplificaciones realizadas es la eliminación del período refractario presente después de cada emisión de un pulso de salida o potencial de acción (AP). En los demás aspectos, el modelo neurona pulsante digital propuesto se ajusta al comportamiento descrito por las neuronas biológicas.

Debe remarcarse que el modelo de neurona pulsante que se presenta en la Figura. 4-20, no se ha desarrollado con el objetivo de emular el comportamiento real de una neurona biológica, sino con el de emular la respuesta de una neurona pulsante para servir de banco de pruebas para la metodología de auto-aprendizaje desarrollada.

La evolución temporal del potencial de membrana "u(t)" se ha codificado mediante un registro de memoria, cuyo contenido se actualiza tal y como se describe en la siguiente Ecuación [4-36]:

$$u(t+1)=\begin{cases} u(t)>0 \rightarrow u(t+1)=u(t)+\left(\sum_j I_j \cdot w_j\right)-1 \\ u(t)\leq 0 \rightarrow u(t+1)=0 \end{cases} \qquad \text{Ecuación [4-36]}$$

Donde "I_j" modela la recepción de un potencial de acción de una neurona "x" en una determinada sinapsis "j" de nuestra neurona, "w_j" es el peso de la sinapsis "j" encargado de evaluar el potencial post sináptico que contribuirá al valor del potencial de membrana "u(t)" de nuestra neurona, y finalmente "t" se corresponde con un valor discreto de tiempo gobernado por la señal global de reloj del sistema. A cada flanco de subida del reloj se evaluará el nuevo estado de cada una de las neuronas.

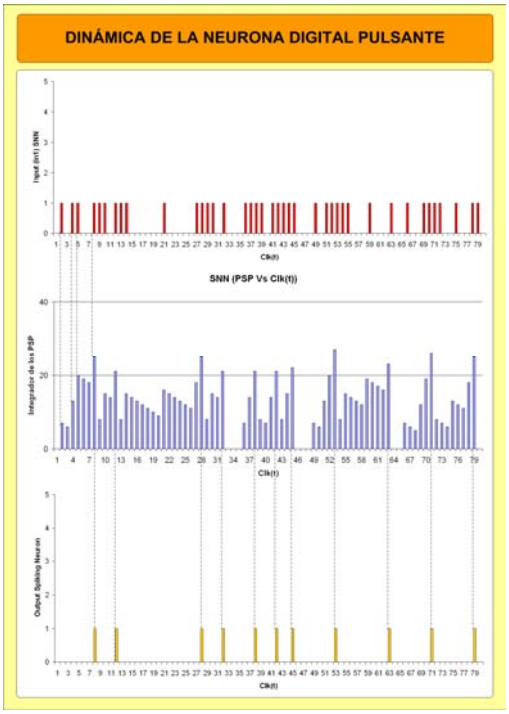

Figura 4-21: Dinámica de la neurona digital desarrollada

De esta forma, el potencial post sináptico generado (PSP) se incrementa o decrementa con cada pulso presináptico recibido en un factor proporcional al peso "w_j" asignado a dicha conexión. Con ello, un incremento o decremento del potencial de membrana "u(t)" estará directamente asociado con una conexión excitatoria/inhibitoria de dicha neurona. Así se produce un potencial de acción (AP) o pulso digital de salida en caso de que el valor del potencial de membrana "u(t)" supere un determinado nivel de disparo (en nuestro caso este nivel ha sido fijado a 20). En la Figura 4-21 se presenta la dinámica de la respuesta del modelo de neurona digital propuesto. Dichos resultados se han obtenido mediante la simulación numérica del modelo digital propuesto.

Si nos fijamos en la Figura 4-21, podemos apreciar cómo un incremento o un decremento estará siempre directamente asociado con una conexión excitatoria o inhibitoria de la neurona, la cual generará un pulso digital de salida (Figura 4.21) en el caso que el potencial de membrana llegue a superar el valor digital del nivel de disparo prefijado (en nuestro caso

este nivel ha sido fijado a 20). La forma de operación del bloque de disparo de la neurona viene descrita por la Ecuación [4-37]:

$$Th(u(t+1)) = \begin{cases} u(t+1) > Threshold = 20 \rightarrow Th(u(t+1)) = 1 \Rightarrow u(t+1) = 0 \\ 20 = Threshold \leq u(t+1) \leq 0 \rightarrow Th(u(t+1)) = 0 \Rightarrow u(t+1) = u(t+1) \end{cases}$$

Ecuación [4-37]

La implementación del modelo de neurona digital pulsante propuesto se ha realizado íntegramente en lenguaje de descripción hardware VHDL y se ha implementado en una FPGA. En dicho diseño, el peso sináptico "w_j" asignado a cada conexión se controla mediante una palabra de 2-bits. Por lo tanto, el sistema diseñado nos permite asignar digitalmente tres valores posibles al peso sináptico, que se corresponden con las siguientes fracciones del valor de la señal de disparo ($+2/5V_{Th}, 0, -2/5V_{Th}$) del potencial de membrana "u(t)". La correspondencia entre la codificación digital y los valores asignados para dichos pesos se presenta en la Tabla 4-5:

Tabla 4-5: Valores de los parámetros del modelo de neurona digital pulsante

Valor de disparo del potencial de membrana	20	
Codificación digital del peso sináptico "W_j" [2bits]	Fracción del valor de disparo asignado al peso sináptico "W_j"	Valor digital asignado a un peso sináptico "W_j"
b00 y b11	0	0
b10	$+\frac{2}{5}*V_{Th}$	+8
b01	$-\frac{2}{5}*V_{Th}$	-8

La codificación del modelo de la neurona digital pulsante se ha realizado en VHDL a fin de permitir una fácil integración del modelo de neurona en el sistema de autoaprendizaje presentado en el siguiente apartado. La codificación VHDL del modelo de la neurona digital se presenta en la Tabla 4-6.

Tabla 4-6: Codificación VHDL del modelo de neurona digital pulsante

```
------------------------------------------------------------
-- VHDL model for a digital Spiking Neuron --
------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY neurona IS

  PORT -- each input is codified with two bits
  (
    clk    : IN  STD_LOGIC;
```

```
    in1    : IN  STD_LOGIC;
    in1x  : IN  STD_LOGIC_VECTOR(0 to 1);

    in2    : IN  STD_LOGIC;
    in2x  : IN  STD_LOGIC_VECTOR(0 to 1);

    in3    : IN  STD_LOGIC;
    in3x  : IN  STD_LOGIC_VECTOR(0 to 1);

    in4    : IN  STD_LOGIC;
    in4x  : IN  STD_LOGIC_VECTOR(0 to 1);

    outx    : out  STD_LOGIC
  );

END neurona;

ARCHITECTURE neuron OF neurona IS

BEGIN
canviestat:
PROCESS (clk)
  VARIABLE state: INTEGER RANGE 0 TO 31;
BEGIN
  IF (clk'EVENT) and (clk='1') THEN
    outx<='0';
    IF (in1='1' and in1x="01") THEN
      state:=state+8;
    END IF;
    IF (in1='1' and in1x="10") THEN
      state:=state-8;
    END IF;
    IF (in2='1' and in2x="01") THEN
      state:=state+8;
    END IF;
    IF (in2='1' and in2x="10") THEN
      state:=state-8;
    END IF;
    IF (in3='1' and in3x="01") THEN
      state:=state+8;
    END IF;
    IF (in3='1' and in3x="10") THEN
      state:=state-8;
    END IF;
    IF (in4='1' and in4x="01") THEN
      state:=state+8;
    END IF;
    IF (in4='1' and in4x="10") THEN
      state:=state-8;
    END IF;
    state:=state-1;
    IF (state>20) THEN
      outx<='1';
      state:=1;
    END IF;
    IF (STATE=0) then
      outx<='0';
    END IF;
  END IF;
END PROCESS canviestat;
END neuron;
```

4.2.6.1.2 Metodología de aprendizaje basada en algoritmos genéticos

La arquitectura de sistema de auto aprendizaje desarrollado [4-110] para las redes neuronales digitales pulsantes se presenta en la Figura. 4-22. Ésta se compone de dos bloques básicos: uno que implementa un algoritmo genético "*Genetic Algorithm Circutry*" (GAC) y el otro que implementa una función de ajuste "*Fitness Circutry*" (FC).

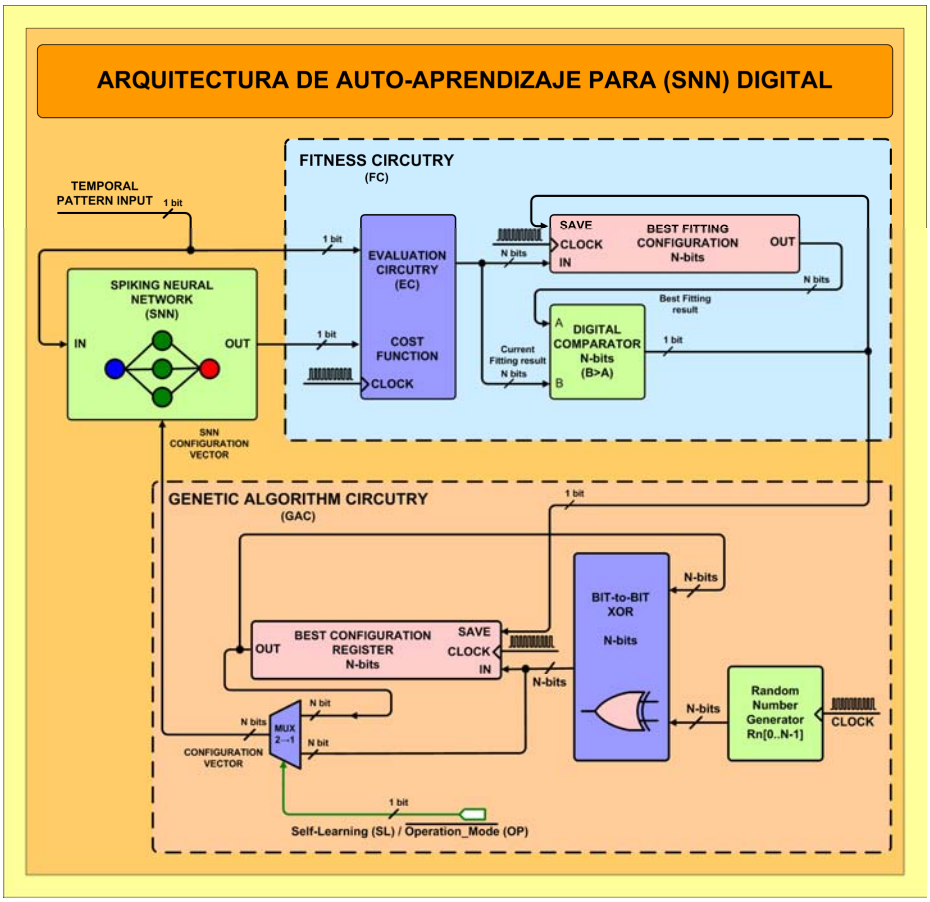

Figura 4-22: Diagrama de la metodología de auto aprendizaje

El bloque GAC es el encargado de generar las nuevas configuraciones de pesos de la red a partir de la mejor configuración obtenida hasta el momento, la cual se almacena en el registro de configuración "*Best Configuration Register*". En dicho bloque también se haya un circuito generador de números aleatorios "*Random Number Generator*", que se encarga de generar el vector de mutación para cada ciclo completo de evaluación de la red neuronal, el cual, a su vez, mediante una puerta lógica XOR, es operado con la mejor configuración obtenida hasta el momento "*Best Configuration Register*" obteniendo la nueva configuración de pesos que será testeada sobre la red neuronal. Esta nueva configuración de pesos (que se obtiene de la salida binaria del bloque XOR) será idéntica a la anterior, excepto en aquellos casos en los que el bit del vector aleatorio sea un "1", en tal caso se asignará a dicho bit del nuevo vector su valor complementario.

El nuevo vector de configuración será aplicado a la red pulsante, siempre y cuando la señal de control del multiplexor de N bits (tantos bits como se requieran para la configuración de los pesos de la red neuronal) del circuito de GAC se encuentre a nivel alto “SL=1” que se corresponde con el modo de operación de *autoaprendizaje* de la red neuronal. En cambio, si dicha señal se encuentra a nivel bajo “SL=0”, equivaldrá a que el circuito se haya en el *modo de funcionamiento*, siendo la configuración usada la evaluada como mejor hasta dicho instante.

El bloque “*Fitness Circuitry*” (FC) se encarga de evaluar la bondad de la nueva configuración para una determinada función asignada a la red neuronal (SNN). Durante el modo de entrenamiento de la red, el bloque de “*Evaluation circuitry*” se encarga de comparar el comportamiento de la SNN con el que debería presentar dicha red, a fin de evaluar una función de coste “*Configuration Fitness*”.
El valor de la función de coste obtenida en este proceso es comparada entonces con el valor de la función de coste de la mejor configuración obtenida hasta el momento, que se haya almacenada en el registro “*Best Fitting Configuration*”. Si el valor obtenido por la función de coste al final del período de evaluación es mayor que el correspondiente a la mejor configuración obtenida hasta el momento, entonces la salida del comparador digital se situará a nivel alto “1” provocando que tanto la función el registro de ajuste “*Best Fitting Configuration*” como el registro de configuración “*Best Configuration Register*” se actualicen con los nuevos parámetros obtenidos.

No obstante, cuando la red neural se encuentra en modo operación funcionamiento (SL=”0”), la configuración de la red neuronal pulsante (SNN) se encuentra fijada a la mejor configuración obtenida hasta dicho momento. A su vez, la señal “*Reset*” es usada para situar la red neuronal en una configuración conocida.

4.2.6.1.2.1 Generación de vectores aleatorios

Un elemento imprescindible para el correcto funcionamiento de la metodología descrita es la calidad del generador de números aleatorios. Es primordial que la probabilidad de

generación de todos los posibles vectores sea la misma para todos ellos y que todos los rangos de mutación sean contemplados por el sistema, ya sea un 0% de mutación (que se corresponde con valor aleatorio de $R_n = 0x000....00$) o un 100% de mutación (que se corresponde con valor aleatorio de $R_n = 0x111....11$).

Mediante esta estrategia aseguramos la capacidad del sistema para la obtención de un mínimo absoluto de la configuración, evitando así que el sistema se vea atrapado en una configuración correspondiente a un mínimo local. El sistema se ha implementado directamente sobre hardware programable (FPGA) obteniendo así la capacidad de usar millones de vectores de configuración por segundo, a fin de conseguir la configuración del sistema que minimice el error de la solución en un tiempo razonable.

Para la generación del vector de mutación debe usarse un generador de números aleatorios o generador de números pseudo-aleatorio. En esta implementación hemos usado un generador de números pseudo-aleatorios mediante un circuito "*Linear Feedback Shift Register*" (LFSR). Si hubiéramos optado por una implementación puramente VLSI se habría podido hacer uso de un circuito generador de números aleatorios puro como el que se presenta en el Anexo A de la presente tesis.

4.2.6.1.3 Evaluación de la metodología para el reconocimiento de patrones temporales

A fin de evaluar la metodología de auto aprendizaje de redes neuronales pulsantes anteriormente propuesta, se ha optado por aplicarla a la resolución de problemas de reconocimiento de patrones temporales directamente relacionados con la capacidad de "memorización" del sistema neuronal.

Durante la fase de entrenamiento una secuencia finita de vectores es aplicada repetidamente a la entrada de dicha red (secuencias de bits para el entrenamiento de la red). La tarea a realizar por la red es el reconocimiento de secuencias de bits de entrada. Es decir, para cada paso de tiempo, la red neuronal debe ser capaz de predecir el valor del próximo bit de la secuencia.

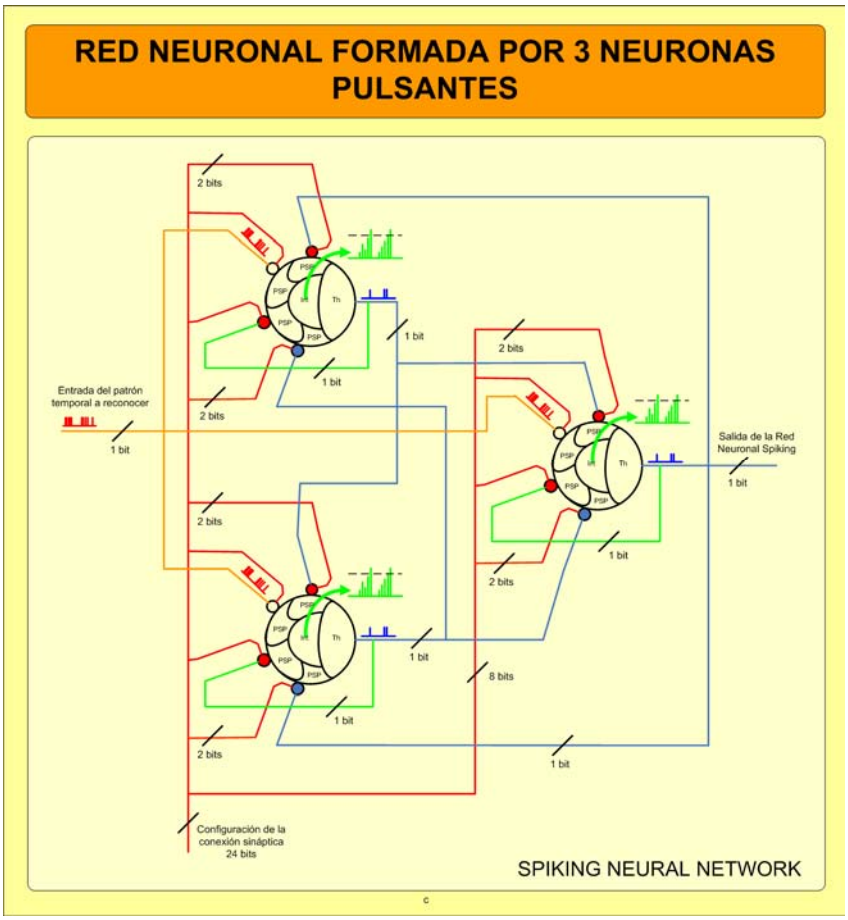

Figura 4-23: Red Neuronal pulsante 3-SNN

La eficiencia de la red se ha evaluado estimando la probabilidad de acierto de la predicción frente a la trama real de entrada de la secuencia temporal. A cada paso de evaluación el bloque GAC proporciona un nuevo vector de configuración mutado, que se usa para configurar los pesos de la red neuronal pulsante (SNN); siendo el bloque (FC) el encargado de evaluar la probabilidad de acierto de la predicción de la red neuronal pulsante (SNN), almacenando la configuración de la red en el registro "*Best Configuration Register*" en el caso que ésta sea mejor que la almacenada anteriormente, o descartándola si es igual o peor a la anteriormente almacenada.

Para evaluar el rendimiento de la metodología propuesta, se han implementado tres redes neuronales pulsantes diferentes (dotadas con topología completa), constituidas por 3, 5 y 8 neuronas pulsantes (3-SNN, 5-SNN y 8-SNN). En la Figura. 4-23, se presenta el diagrama

de la red de la SNN constituida por 3 neuronas pulsantes. Para el entrenamiento de la red se han usado las secuencias de bits lo más complejas posibles a fin de maximizar la dificultad del problema de reconocimiento de patrones. Por lo tanto, para el entrenamiento y test de la red neuronal hemos usando tramas pseudo-aleatorias, generadas mediante un bloque LFSR. Remarcar que las tramas de bits pseudo aleatorias se caracterizan por tener las mismas propiedades estadísticas que las tramas de bits puramente aleatorias, a excepción del hecho que las primeras tienen periodicidad, como bien ya hemos descrito en el capítulo segundo de la presente tesis.

Para evaluar la metodología propuesta, se han usado secuencias de bits pseudo-aleatorias con distinta periodicidad de repetición (7, 15, 31, 63, 127, 255, 511 y 1023 ciclos de proceso que se corresponden con el uso para el entrenamiento del sistema de circuitos LFSR de 3 a 10 bits). Con el uso de estas secuencias pseudo-aleatorias, la dificultad de la tarea de memorización se ha visto maximizada. Cabe remarcar que hemos aplicado cada una de estas tramas a las diferentes redes propuestas. En cada caso, anteriormente se ha configurado cada una de las redes mediante la metodología de auto aprendizaje propuesta.

El procedimiento experimental seguido para la evaluación de la metodología ha consistido en esperar a que la configuración de la red se estabilizase en un valor óptimo, a fin de evaluar la probabilidad de acierto asociada a dicha configuración. Posteriormente, se han reproducido a la entrada de las mismas tramas de bits pseudo-aleatorios con el fin de evaluar su tasa de acierto.
En la Figura 4-24 se presenta la probabilidad de acierto de las diferentes redes neuronales evaluadas en función del número de bits del que consta el circuito generador de bits pseudo aleatorios (usado para testear el circuito).

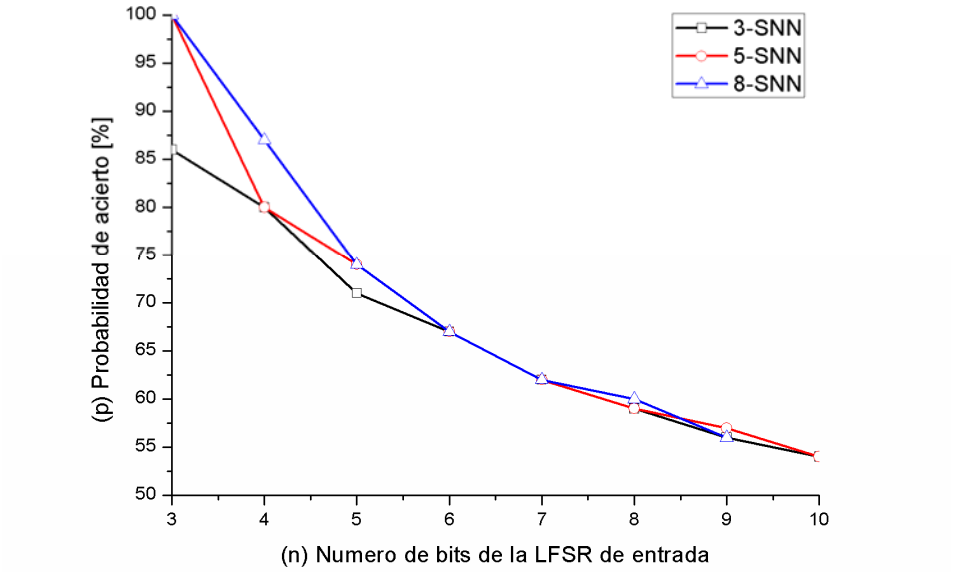

Figura 4-24: Probabilidad de acierto de la metodología de auto-aprendizaje

Como puede observarse en la Figura 4-24, a medida que el número de bits de la secuencia a evaluar crece y/o el número de neuronas de la red disminuye, la probabilidad de acierto/predicción de la red disminuye proporcionalmente.

A partir de las medidas experimentales obtenidas, hemos evaluado una regla matemática que relaciona la probabilidad de acierto de las diferentes redes evaluadas en función de la longitud de la sequencia de datos temporales a reconocer (Ecuación [4-38]):

$$p = \frac{1}{2} + \frac{2}{5}\sqrt{\frac{M}{N}} \qquad \text{Ecuación [4-38]}$$

Esta regla predice de forma aproximada la probabilidad de acierto “p” a partir de la longitud de la secuencia de bits a reconocer (N bits), y la topología de la red neuronal usada “M” (número de neuronas, conexiones de la red,…). Los valores del parámetro “M” obtenidos para la red 3-SNN es de “M=8,5”, para la red de 5-SNN tenemos que “M=10.5” y “M=12.5” para la red 8-SNN. Por lo tanto, lo que indica la Ecuación [4-38], es que a medida que la longitud de la secuencia temporal a reconocer va aumentando, la probabilidad de acierto de la red neuronal va decayendo hasta un valor límite de 0.5 (que se

corresponde con el 50% de probabilidad de acertar que el bit a reconocer sea un “0” o un “1”).

En esencia, el parámetro “M” mide la capacidad del sistema para reconocer las secuencias de bits pseudo-aleatorios, pudiendo ser usado para evaluar la capacidad de proceso/predicción de un sistema neuronal. El índice “M” de una SNN lo definiremos como la secuencia pseudo-aleatoria de mayor longitud de la cual es capaz de predecir con al menos un 90% de probabilidad de éxito. Este parámetro puede considerarse como un buen indicador de la capacidad de predicción de la red, independientemente del tipo y topología implementada experimentalmente.

Tabla 4-7: Comparación entre los valores reales de probabilidad de acierto experimentales y los previstos pro el modelo analítico

Longitud de la trama temporal a reconocer (N bits)	3-SNN		5-SNN		8-SNN	
	Prob. Experimental	Prob. Modelo	Prob. Experimental	Prob. Modelo	Prob. Experimental	Prob. Modelo
7	86%	94%	100%	99%	100%	100%
15	80%	80%	80%	83%	87%	87%
31	71%	71%	74%	73%	74%	75%
63	67%	65%	67%	66%	67%	68%
127	63%	60%	62%	62%	62%	63%
255	59%	57%	59%	58%	60%	59%
511	56%	55%	57%	56%	56%	56%
1023	54%	54%	54%	54%	54%	54%

Finalmente en la Tabla 4-7 se pueden comparar las predicciones de acierto obtenidas teóricamente mediante la Ecuación [4-38], con los valores de acierto obtenidos experimentalmente. En los resultados existe una muy buena relación entre las medidas experimentales y los datos estimados mediante el modelo desarrollado. No obstante, las mayores discrepancias entre ambos se dan para la red 3-SNN cuando N=7, la cual se debe primordialmente al hecho que el modelo presentado es continuo mientras que la probabilidad experimental es discreta; esto equivale, a que la predicción de acertar para N=7 sea siempre un número racional con el denominador igual a 7. Por lo tanto, la predicción de acierto obtenida con el modelo continuo del 94%, nunca podrá llegar a ser alcanzada por el sistema real (ya que sólo puede tomar los valores 6/7 (85%) o 7/7 (100%)).

4.2.6.2 Desarrollo de una neurona pulsante mixta

A lo largo de las últimas décadas se han dedicado muchos esfuerzos al desarrollo de redes neuronales pulsantes (SNN) mediante todo tipo de soluciones: ya sea mediante sistemas puramente digitales, totalmente analógicos o bien mixtos (formados por una parte analógica y otra digital).

En el caso de las implementaciones analógicas [4-114, 4-115], éstas tienen la ventaja de ocupar mucha menos área que las otras implementaciones VLSI, además suelen diseñarse para reproducir fielmente el comportamiento y la dinámica de las neuronas biológicas. Por lo tanto, éstas son conocidas por reproducir fielmente el comportamiento no lineal de las neuronas, mucho mejor que cualquiera de las demás implementaciones existentes. Por otra parte, tienen la gran desventaja de presentar una muy baja capacidad de reconfiguración de la topología de red, así como un dificil ajuste dinámico de los pesos sinápticos para cada una de las neuronas.

En el otro extremo hallamos las implementaciones digitales de neuronas pulsantes [4-116, 4-117] que podemos hallar implementadas directamente en FPGA’s o en VLSI (mediante tecnologías/celdas estándares CMOS digitales). De esta forma se obtienen unos costes de fabricación menores, así como una capacidad de configuración muy superior a la de los sistemas analógicos. Otra de las ventajas de las implementaciones digitales es su fiabilidad de operación con respecto a las implementaciones analógicas sensibles al ruido exterior. A su vez, el ciclo de diseño e implementación de las soluciones digitales es mucho menor que en el caso de las soluciones puramente analógicas, al poder ser las primeras fácilmente re-configurables en el caso en que se hayan implementado sobre dispositivos de lógica programable (FPGA).

Para aprovechar las ventajas de cada una de las tecnologías, recientemente se han empezado a desarrollar sistemas neuronales mixtos (analógico/digitales) [4-118, 4-119]. En estas implementaciones la parte digital del sistema neuronal se encargará de gestionar la conectividad de la red, mientras que la parte analógica será la encargada de reproducir la dinámica interna de la neurona. Con ello, las soluciones mixtas (analógico/digitales)

representan un avance en el campo de las redes al permitir la implementación de sistemas realistas, a la vez que masivos al poder ser integrados conjuntamente en un sólo chip.

A lo largo de los siguientes subapartados se presenta una arquitectura simple de neurona pulsante mixta (analógica/digital), basada en el funcionamiento de un circuito integrado caótico [4-120]. El funcionamiento detallado se describe en el *Anexo A* de la presente tesis. La neurona mixta desarrollada dispone de una metodología fiable de comunicación interneuronal, así como de una metodología lo suficientemente simple para el ajuste de los pesos sinápticos. A su vez, la neurona desarrollada integra en su interior un núcleo analógico para la reproducción del comportamiento no-lineal [4-116]. Con todo ello, la solución desarrollada requiere de muy poca área para su integración VLSI, al no requerir de complicados circuitos digitales que reproduzcan el comportamiento de la neurona [4-116].

4.2.6.2.1 Arquitectura de la neurona pulsante mixta

La arquitectura de la neurona pulsante mixta propuesta (Figura 4-25) se compone de una parte analógica y de otra digital. La parte digital es la encargada de permitir el establecimiento de una comunicación fiable y sencilla entre las diversas neuronas, mientras que el circuito analógico es el encargado de reproducir el proceso de generación del potencial de membrana “u(t)” y de los potenciales de acción (AP) de salida de la neurona.
A la parte digital de la neurona (Figura 4-25) se la ha denominado *Bloque digital sináptico* (DSB) y será la encargada de realizar las siguientes funciones: la gestión de los pesos (los cuales podrán ser excitatorios o inhibitorios). Los potenciales de acción post sinápticos (analógicos) han de ser filtrados (los denominaremos *Potenciales de Acción Filtrados* (FAP)) para que sean interpretados debidamente por el núcleo analógico, obteniendo así una comunicación fiable y segura.

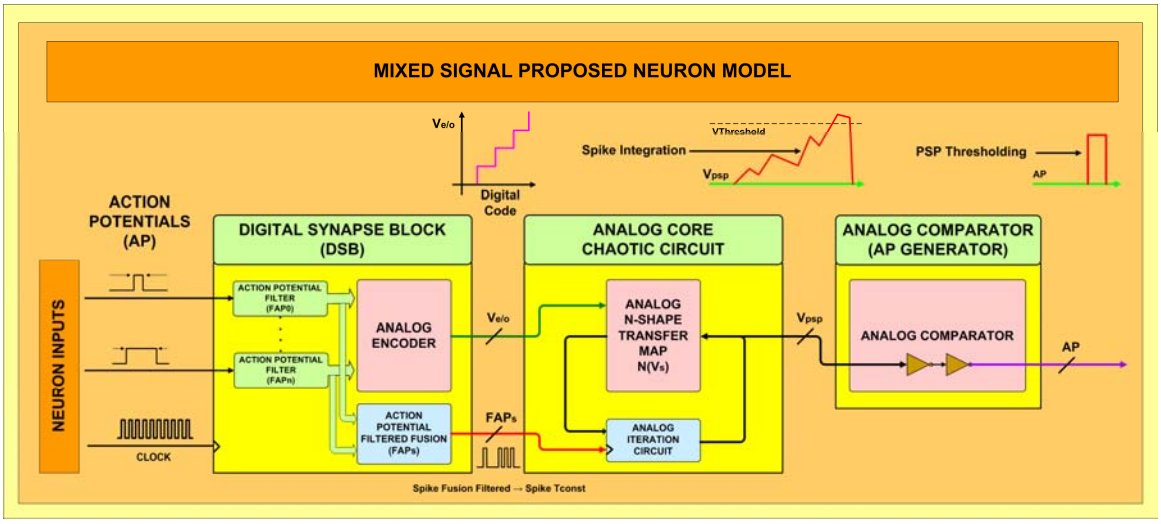

Figura 4-25: Arquitectura de la neurona pulsante mixta

Una de las ventajas más importantes de la arquitectura neuronal propuesta radica en el hecho que es capaz de filtrar las señales digitales espurias (glitches). Dichos glitches se generan en los circuitos digitales diseñados mediante tecnologías CMOS nanométricas, debido a la gran densidad de líneas de interconexión existentes. Esto implica un aumento del efecto de la diafonía (Crosstalk) entre pistas de interconexión adyacentes [4-121, 4-122]. Debido a esto, las señales espurias generadas (debidas al acoplamiento entre interconexiones independientes pero cercanas) pueden provocar disparos no deseados, y por lo tanto, un mal funcionamiento general de la red. Este problema es inherente a cualquier implementación asíncrona de redes neuronales pulsantes (como es el caso de las implementaciones analógicas), viéndose muy minimizado en las soluciones basadas en sistemas síncronos, como la solución desarrollada.

El núcleo analógico desarrollado para la neurona (Figura 4-25) realiza las siguientes funcionalidades: es el encargado de la evaluación del potencial de membrana “u(t)” a partir de la integración de todos los potenciales post-sinápticos recibidos (es decir, mediante la integración de los potenciales de acción filtrados (FAP)) que son generados por el bloque sináptico digital (DSG). Debido al carácter caótico del núcleo, cada neurona tendrá su comportamiento particular independientemente del de las demás, lo cual deshace la

correlación temporal entre señales generadas por las diversas neuronas de la red, maximizando así la asíncronicidad entre señales independientes. La descorrelación entre señales es primordial para que la red pueda operar correctamente. Afirmaremos que dos neuronas se hallan descorrelacionadas cuando son capaces de proporcionar diferentes salidas temporales para una misma entrada aunque ambas estén idénticamente configuradas (pesos de las neuronas de la red) y sean estimuladas de forma idéntica. Complementariamente, si dos neuronas son configuradas idénticamente y tienen los mismos estímulos de entrada presentan la misma salida temporal y se podrá afirmar que ambas neuronas se encuentran correlacionadas.

Dos neuronas descorrelacionadas con la misma configuración y los mismos estímulos de entrada deben generar una señal de salida cualitativamente similar, pero con distribución temporal significativamente diferente. La necesidad fundamental de disponer de neuronas descorrelacionadas radica en intentar minimizar las colisiones entre las señales presinápticas de entrada (en la parte digital del sistema (DSB)).

A su vez, la neurona desarrollada emula la asíncronicidad intrínseca de las neuronas biológicas mediante el uso de diferentes frecuencias de operación en los bloques digitales y los analógicos que integran la neurona. La parte digital de la neurona opera con una señal de reloj de alta frecuencia, mientras que el bloque analógico opera con una frecuencia mucho menor (del orden de 100 veces menor). Por lo tanto, si el núcleo analógico de la neurona se desarrolla correctamente, se podrá asumir que no existe correlación entre las neuronas de la red, lo que implicará que la probabilidad de colisión entre dos señales es relativamente pequeña y que en promedio el comportamiento del sistema puede considerarse como asíncrono.

4.2.6.2.1.1 Bloque analógico de la neurona mixta

El bloque analógico es el encargado de integrar los pulsos presinápticos de entrada filtrados por el bloque (DSB). Para descorrelacionar las señales neuronales hemos usado como elemento integrador (analógico) de la neurona un oscilador caótico. Cuando nos referimos a que el núcleo analógico se comporta caóticamante, ponemos de manifiesto la imposibilidad de hacer predicciones precisas sobre su comportamiento a largo plazo, ya que los sistemas

caóticos se caracterizan por ser extremadamente sensibles a las condiciones iniciales. De este modo se consigue una descorrelación efectiva entre neuronas.

Para reproducir el comportamiento caótico deseado, se ha diseñado un circuito VLSI analógico mediante una tecnología de 0.35μm, de *Austria Micro Systems* (AMS), con una función de transferencia en forma de letra N mayúscula "N(x)" [4-120] la cual es capaz de reproducir un comportamiento caótico [4-20]. En consecuencia, mediante la función de transferencia "N(x)" (Figura 4-26) se integrarán los pulsos presinápticos de entrada previamente filtrados (FAP) provenientes del bloque digital (DSB).

Como bien puede apreciarse en la Figura 4-26, el estado del núcleo analógico (PSPi) (que se corresponde con el valor del potencial de membrana de la neurona) es actualizado en cada iteración a través de la función "N(x)" de la siguiente forma (Ecuación [4-39]):

$$PSP_{i+1} = N\left(PSP_i\right) \qquad \text{Ecuación [4-39]}$$

Dependiendo del punto de operación en el que nos hallemos, en el mapa de la función "N(x)" fijaremos si el pulso post sináptico (FAP) recibido se corresponde a una conexión excitatoria o inhibitoria. La dinámica obtenida será a su vez extremadamente dependiente de las condiciones iniciales (como es de esperar al hacer uso de un circuito caótico) a fin de asegurar la asíncronicidad entre neuronas idénticas.

En la Figura 4-26 se presenta la relación entre dos iteraciones consecutivas de la función que integra el PSP (PSP_{i+1} y PSP_i) (que evalúa el potencial post-sináptico de membrana). La característica dinámica del sistema se describe mediante dos líneas: la curva de transferencia estática (la función con forma de letra N mayúscula) y la línea de tendencias (que se corresponde con la función identidad). Cuando no se recibe ningún pulso de entrada (FAP), es decir, la línea digital se haya fija a un nivel bajo "0", el potencial post sináptico (PSP) o potencial de membrana pasa a tener un valor definido por la función identidad (gracias a la descarga del condensador de puerta), emulando así los mecanismos biológicos de disminución de liberación y de eliminación de neurotransmisores en el canal sináptico que conduce paulatinamente a una disminución del potencial de membrana "u(t)".

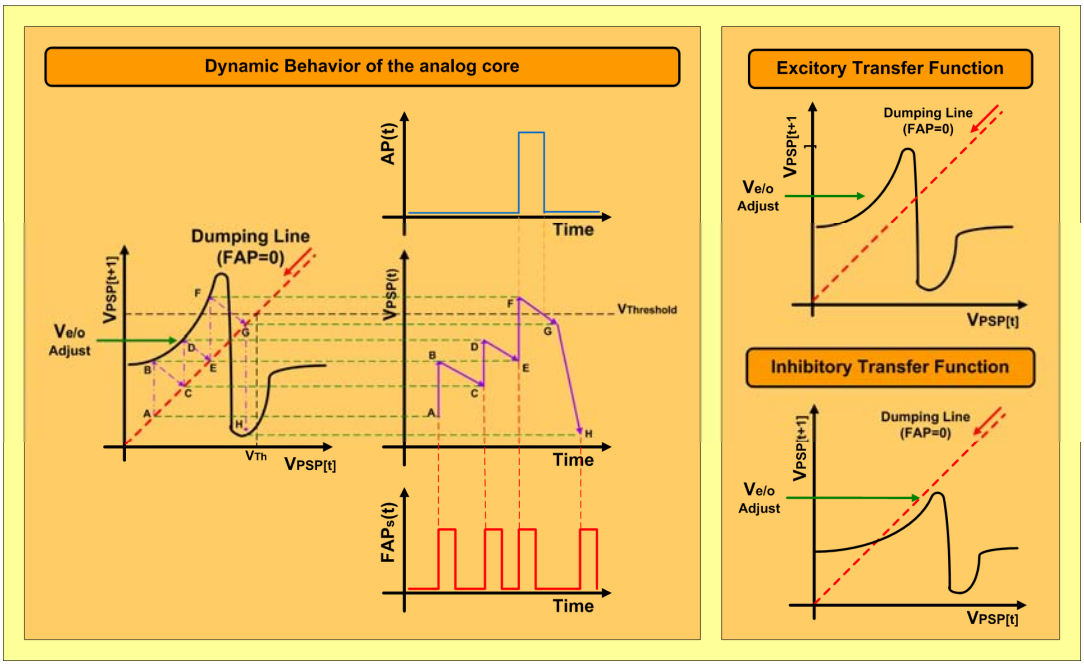

Figura 4-26: Dinámica de la función de transferencia del núcleo analógico

Si por el contrario se recibe un pulso post sináptico de entrada (FAP) a nivel lógico alto "1", el potencial post sináptico (PSP) o potencial de membrana "u(t)" pasa a tener un valor definido por una función de transferencia (función en forma de la letra N mayúscula). La funcionalidad de este mecanismo es el de realizar la integración de los potenciales de acción post sinápticos recibidos (FAP) como se muestra presenta en la parte izquierda de la Figura [4-26]. Entonces, para los casos excitatorios/inhibitorios la función de transferencia es la encargada de integrar positivamente/negativamente los pulsos post sinápticos (FAP) recibidos. Dicha función se ajusta mediante la tensión "$V_{e/o}$" generada por la parte del bloque digital (DSB), tal y como se muestra en la parte derecha de la Figura [4-26]. Finalmente, el núcleo analógico es el encargado de generar un potencial de acción (AP) si la tensión de membrana ("u(t)" definido mediante la variable PSP) supera la tensión prefijada para el umbral de disparo V_{TH}, al igual que sucede en las neuronas biológicas. Como elemento comparador se ha utilizado un inversor CMOS, el cual hemos diseñado para que conmute a una determinada tensión umbral V_{TH} a fin de simplificar el circuito y consumir la mínima área posible.

El circuito analógico que implementa la dinámica descrita hasta el momento se presenta en la Figura 4-27. Dicho circuito se basa íntegramente en el circuito caótico CMOS que hemos desarrollado para diversas aplicaciones y que se describe en detalle en el *Anexo A*.

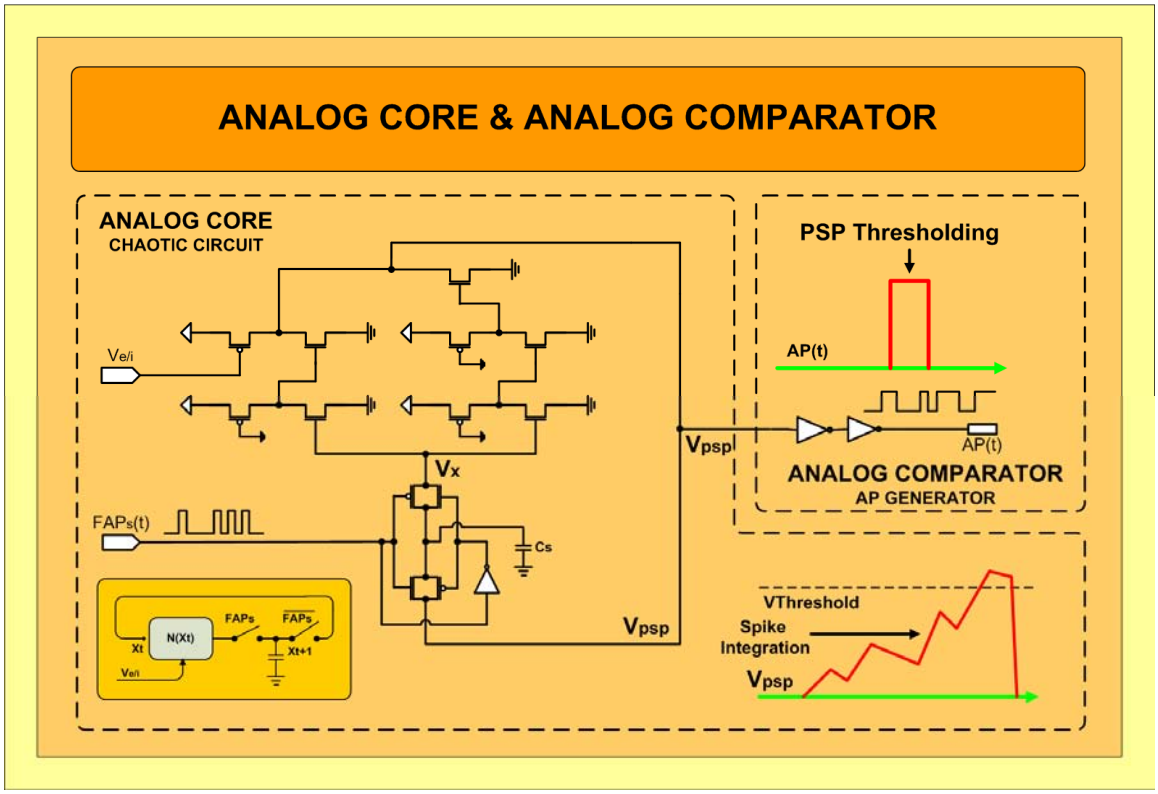

Figura 4-27: Diseño VLSI del núcleo analógico (Función 'N(x)' iterada) y del generador de (AP) de la neurona

La generación de las diferentes tensiones "$V_{e/o} = V_{i/o}$" se realiza directamente en el bloque digital (DSB), el cual incorpora 4 puertas de transmisión conectadas a cuatro referencias de tensión discretas, las cuales son las encargadas de proporcionar la tensión de polarización del transistor PMOS.

Una mejora del circuito consistiría en la sustitución del transistor PMOS por "m" transistores PMOS (cuyas dimensiones deberían ajustarse individualmente) conectados en paralelo, los cuales podrían ser controlados directamente mediante una palabra digital de m-bits, consiguiéndose así que el mapa de la función de transferencia dependiera de un

número digital. En el diseño fabricado y medido en laboratorio se integró sólo un transistor PMOS.

Si observamos la Figura 4-28 hallaremos las medidas experimentales realizadas sobre el presente circuito analógico, para una "$V_{e/o}$" (fijada a 0.75V) que se corresponde a una configuración excitatoria, y para otra "$V_{e/o}$" (fijada a 1.09V) que se corresponde a una configuración inhibitoria.

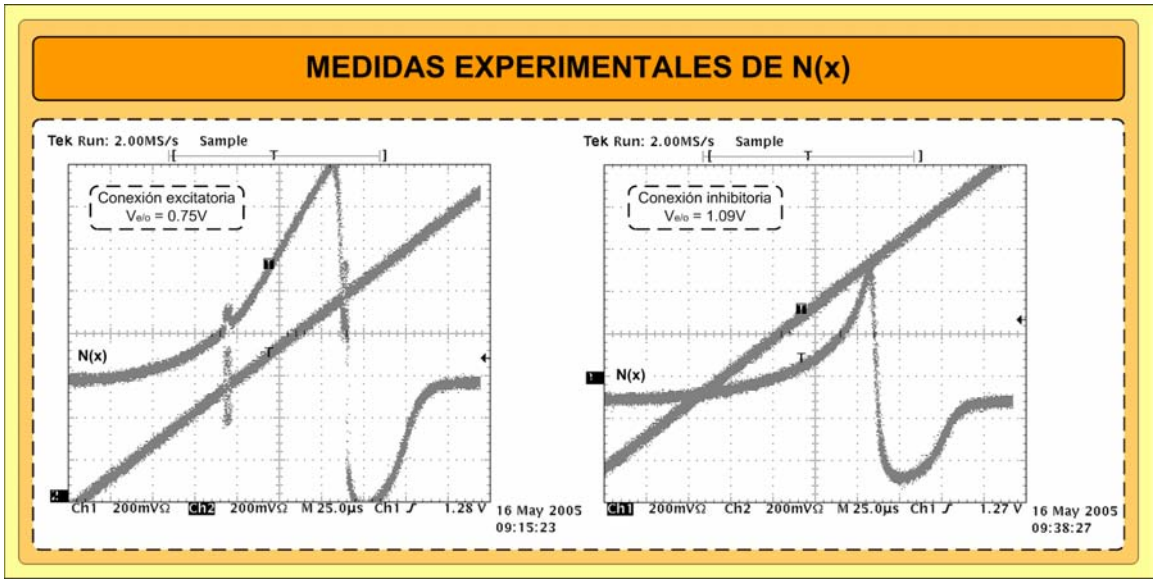

Figura 4-28: Medidas de la función de transferencia N(x) vs Ve/o

Para la generación de los potenciales de acción (AP), el nodo donde se ubica la tensión V_{PSP} (que se corresponde con el potencial de membrana) se haya interconectada con un comparador de tensión, encargado de generar las señales de los potenciales de acción (AP). Este comparador se ha implementado mediante dos inversores CMOS, a los cuales se les ha ajustado la tensión de disparo de los inversores V_{TH}. Por lo tanto, el potencial de acción se dispara sólo en el supuesto que $V_{PSP} > V_{TH.}$.

4.2.6.2.1.2 Bloque digital de la neurona mixta

El potencial de membrana "u(t)" o (PSP) resultado de la integración de los potenciales de acción post sinápticos se usa para generar los potenciales de acción (AP), que son a su vez transmitidos a las otras neuronas (como pulsos presinápticos). Por lo tanto, la circuitería

digital desarrollada trabaja sincronizada con una señal de reloj interna a la FPGA de alta velocidad, con un período de oscilación T_{clk}. El período de la señal de reloj se ha elegido de tal forma que sea mucho menor que el período típico de activación de las señales de los potenciales de acción (AP).

El *Bloque Sináptico Digital* (DSB) se compone de dos biestables tipo D, encargados de filtrar las señales que llegan de los potenciales de acción (AP) presinápticos y cuya finalidad es convertir estos pulsos en potenciales de acción post-sinápticos filtrados (FAP). Estas señales tendrán siempre un período fijo equivalente "T_{clk}" asignado. Todas las señales de (FAP) recibidas son fusionadas en una sola señal de FAP global mediante una puerta OR (Figura 4-29) de tantas entradas como entradas presinápticas disponga la neurona.

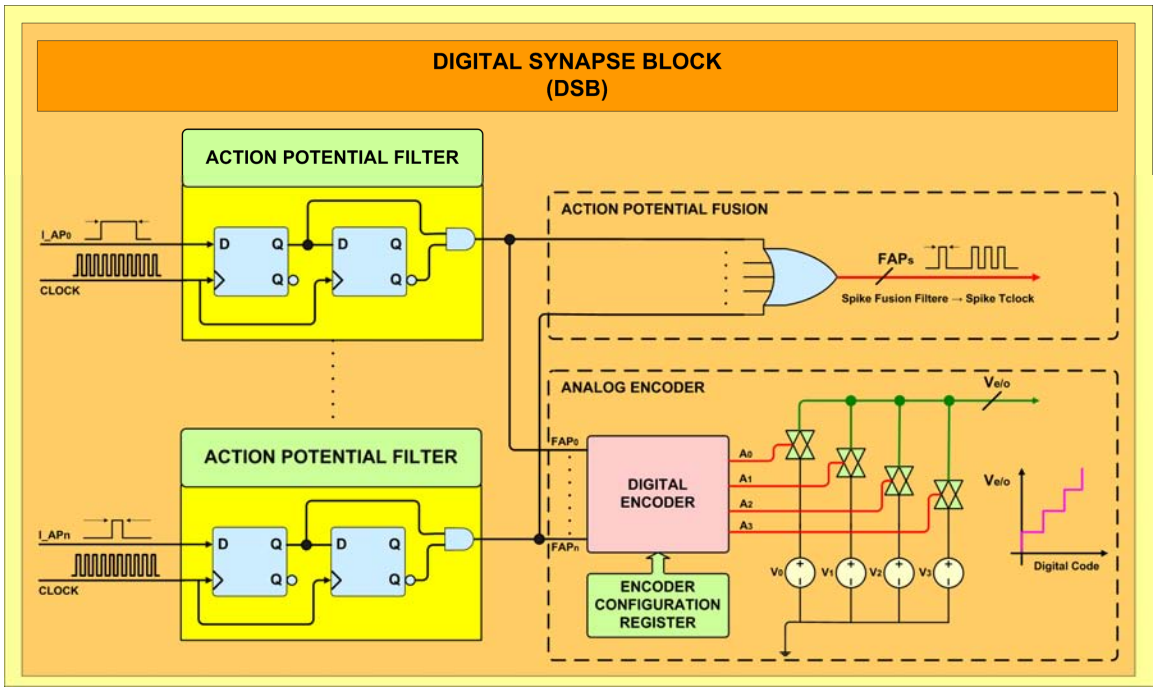

Figura 4-29: Diagrama del bloque sináptico digital (DSB)

La colisión entre señales (FAP) se ve minimizada si la frecuencia de la señal del reloj es lo suficientemente alta, unido al hecho que los núcleos analógicos de todas las neuronas se hallan descorrelacionados. Cualquier señal (AP) pre-sináptica con una duración inferior al

período de la señal de reloj usada por el sistema T_{clk} será considerada como una señal espuria, y no será transmitida al núcleo analógico.

El sub-bloque denominado codificador analógico "*Analog Encoder*" de la Figura 4-29 representa una solución particular para el caso de 4 tensiones de salida ajustables representadas por $(V_3,..,V_0)$. Éstas son seleccionadas mediante la activación de las diversas puertas de transmisión, que hacen las funciones de multiplexor analógico. Las referencia de tensión (LM285M) y las puertas de transmisión (SN74HC4066) se han implementado sobre el mismo circuito impreso en el que se ha ubicado el ASIC del núcleo analógico (Figura 4-30)), controladas a su vez mediante sus correspondientes señales digitales ($A_3,..,A_0$). El codificador digital controlado por las diversas señales FAP se ha diseñado para sólo permitir la activación de una sola de las señales digitales a la vez (se ha establecido una jerarquía de señales prioritarias). Estas reglas de selección/prioridad son configuradas mediante el contenido del *registro de configuración* del codificador digital (Figura 4-29), siendo éste el encargado de establecer la relación entre los diversos FAPs y la señales digitales ($A_3,..,A_0$), encargadas a su vez de la activación de las puertas de transmisión.

Todo ello implica que una vez activada una determinada señal "FPA_i" el codificador analógico asociado a cada una de ellas seleccionará la tensión de excitación/inhibición prefijada a la tensión $V_{e/o}$, que depende directamente del valor digital almacenado en el registro de configuración (parte inferior derecha de la Figura. 4-29) para el caso particular del circuito con 4 voltajes de configuración.

Por lo tanto, la curva estática de transferencia presentada en la Figura. 4-26 se hallará contínuamente variando en función de la señal de FAP que se encuentre activa en cada instante de tiempo.

4.2.6.2.2 Resultados experimentales de la neurona mixta

El prototipo de neurona mixta (analógico-digital) se ha desarrollado en tres partes bien diferenciadas: la primera puramente digital que se ha implementado en una FPGA, la segunda parte constituida por un conjunto de dispositivos discretos montados sobre un circuito impreso (en el cual hallamos un codificador analógico interconectado con un

conjunto de referencias de tensión) y la tercera parte del sistema es el ASIC del núcleo analógico (oscilador caótico) encapsulado en un PLCC68, ubicado en el mismo circuito impreso donde se ubica la segunda parte del sistema. El núcleo analógico de la neurona consiste en un oscilador caótico (Anexo A), realizado con una tecnología CMOS 0.35µm (parte izquierda de la Figura 4-30) de *Austria Micro Systems* (AMS). Dicho ASIC ocupa un área de silicio de 47µm x 57µms, y se ha implementado íntegramente mediante transistores MOS, el cual se ha encapsulado en un PLCC68 para facilitar su manipulación.

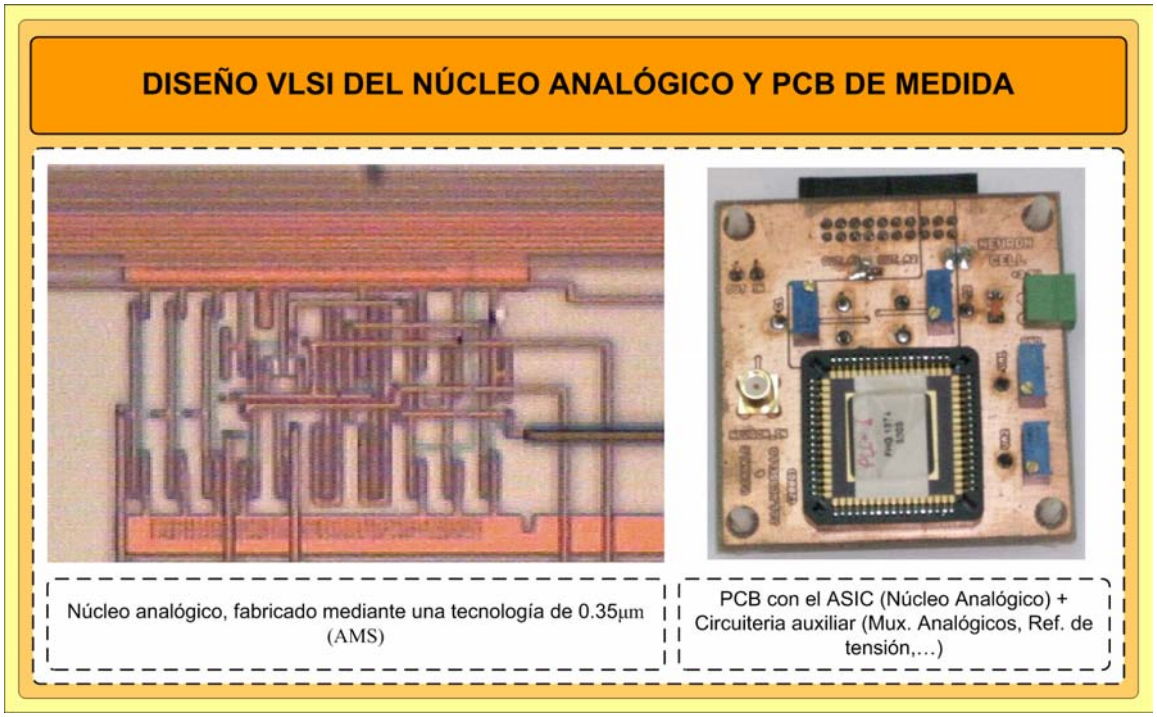

Figura 4-30: Fotografía del núcleo analógico fabricado y de la PCB con el ASIC + los elementos auxiliares

En la Figura 4-31 se presentan los voltajes experimentales medidos del potencial de membrana "u(t)" (tensión del potencial post sináptico V_{PSP}) que son el resultado de la integración de los diferentes potenciales de acción post sinápticos filtrados (FAP) recibidos. Tal y como puede apreciarse, cuando V_{PSP} es mayor a la tensión de umbral de disparo V_{TH} del mapa de iteración, se realiza la emisión del potencial de acción (AP). También en la misma Figura. 4-31 se puede apreciar cómo un total de 13 pulsos son necesarios para obtener la emisión de un (AP) de la neurona mixta desarrollada. La duración de los pulsos

post sinápticos (FAP) depende directamente del período del reloj usado en el bloque digital de la neurona, que en las medidas se ha fijado a $T_{Clock} = 250\,ns$. Con ello, cualquier señal presináptica con un período inferior a éste será filtrada al considerarse una señal espuria.

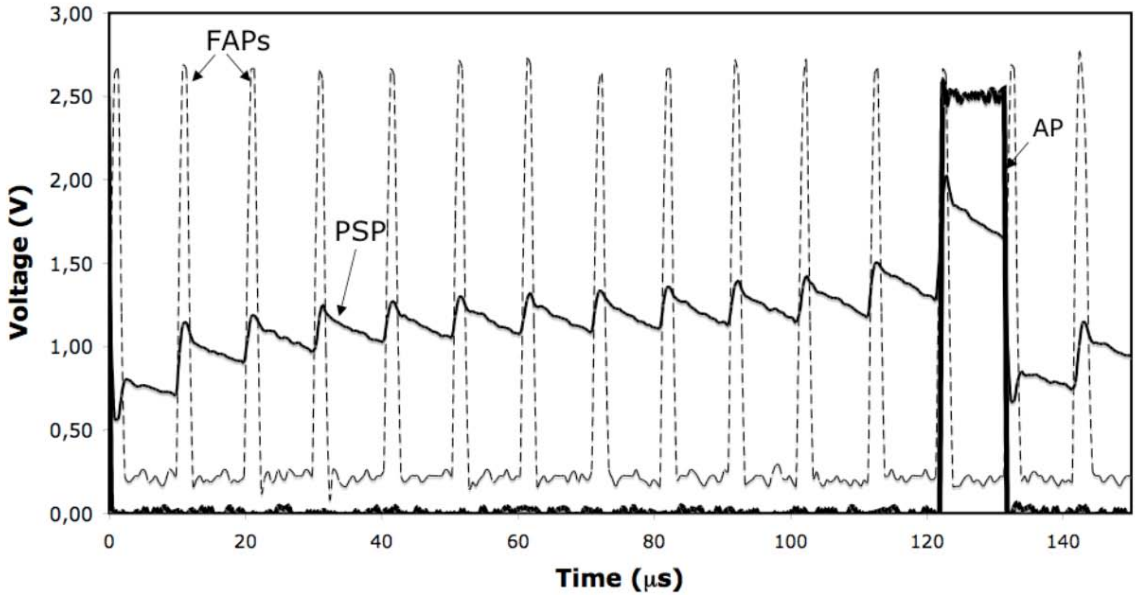

Figura 4-31: Medida experimental del potencial de membrana (PSP) para una conexión excitatoria

Si nos fijamos en la Figura 4-32 podemos apreciar la irregularidad en el comportamiento del sistema. Para la realización de estas medidas se ha excitado la neurona en una de sus entradas con un potencial de acción (AP) pre-sináptico, que ha consistido en una señal periódica cuadrada (la señal cuadrada se presenta en la parte inferior de la Figura 4-32). A fin de evaluar numéricamente la sensibilidad del sistema se han calculado los exponentes de Lyapunov de esta serie temporal, donde un exponente positivo/negativo indica un comportamiento del sistema caótico/periódico. Concretamente, los exponentes de Lyapunov de la serie temporal que se presenta en la Figura 4-32 se han evaluado mediante la técnica descrita por M.T. Rosenstein et al.[4-123], siendo el valor obtenido para el exponente de λ=+0.05 (que se corresponde con un comportamiento caótico de la señal). Por lo tanto, se puede afirmar que al existir comportamiento caótico en cada uno en los núcleos analógicos, los diferentes potenciales de acción (AP) generados en las diversas neuronas se

hallarán decorrelacionados, lo cual minimiza la probabilidad de colisión de dos señales de FAP en una misma neurona (debido a la metodología de fusión usada).

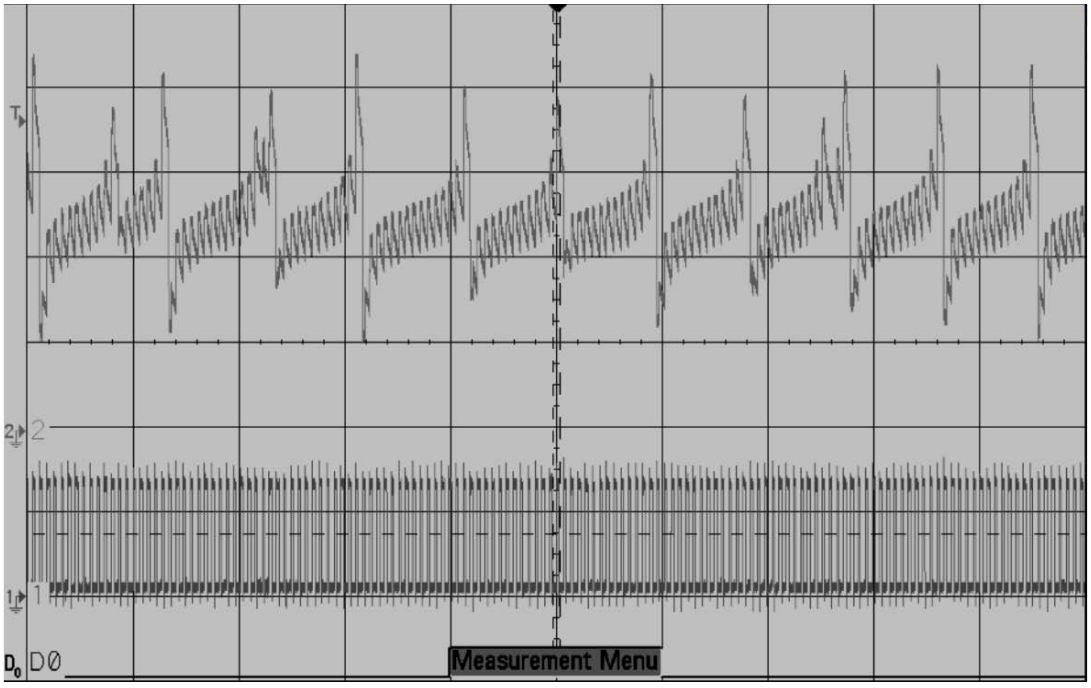

Figura 4-32: Medida experimental de una serie temporal del potencial de membrana (PSP)

Finalmente, en la Figura 4-33 se presenta el diagrama de bifurcaciones del potencial de membrana o potencial post sináptico V_{PSP} en relación a la tensión de configuración del enlace $V_{e/i}$, donde valores grandes/pequeños de esta última tensión están relacionados con configuraciones inhibitorias/excitatorias. Para la obtención de este diagrama se ha excitado la neurona con una señal cuadrada periódica (emulando un tren de potenciales de acción presinápticos) de frecuencia 2MHz. Para este caso se puede apreciar (Figura 4-33) la existencia de dos regiones con un comportamiento bien diferenciado, como es la zona de comportamiento periódico y la de comportamiento caótico. También se observa una zona de intermitencia periódica entre zonas caóticas.

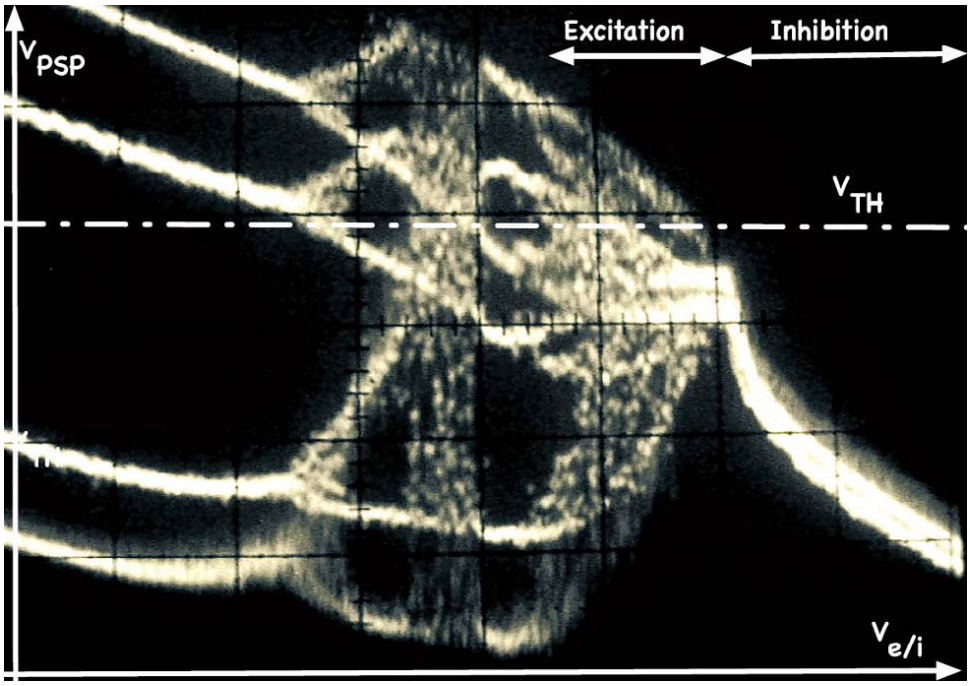

Figura 4-33: Mapa de bifurcaciones experimental del núcleo analógico

A partir del conjunto de medidas experimentales presentadas se concluye que la arquitectura de red basada en la neurona mixta pulsante propuesta es completamente funcional, habiéndose fusionado en la neurona propuesta las ventajas de las implementaciones de las neuronas digitales y analógicas.

4.3 LAS REDES NEURONALES ESTOCÁSTICAS

A continuación se presenta una introducción a las motivaciones y los trabajos científicos referentes a la implementación de redes neuronales mediante el uso de la computación estocástica, a fin de establecer el marco básico sobre el cual presentar las diferentes aportaciones realizadas en este campo. Concretamente, describiremos el desarrollo de las neuronas de segunda generación [4-38] implementadas mediante elementos estocásticos. Finalmente, presentaremos un ejemplo de implementación de una red neuronal basada en el uso de neuronas estocásticas basadas en la codificación extendida con signo (SESL).

4.3.1 INTRODUCCIÓN

Entre las aplicaciones más significativas de las redes neuronales se hallan el reconocimiento de patrones y la clasificación. Para estas aplicaciones no es fácil establecer un tamaño y una estructura estándar de red, ya que el tamaño está íntimamente relacionado con el problema a tratar. Aunque en la literatura [4-60] hallamos estimaciones que postulan que para la resolución de este tipo de problemas, en sistemas reales se requieren alrededor de unas 1000 neuronas, este número de neuronas, incluso con la actual tecnología de fabricación de circuitos integrados "*Very Large Scale Integration*" (VLSI), no es sencilla de integrar. Esto se une al hecho que al aumentar el nivel de integración, a fin de reducir el área de integración, aparecen nuevos fenómenos físicos y eléctricos que obligan a un nuevo rediseño del sistema para adaptarlo a las peculiaridades de dicha tecnología.
Por otro lado, si intentamos implementar estas redes neuronales mediante dispositivos de lógica programable (FPGA), la ingente cantidad de recursos lógicos que se requiere establece un factor limitante a estas implementaciones, lo que implica que la integración de determinados tamaños de red es inviable en dispositivos comerciales.

Cabe recordar que en la práctica las redes neuronales hardware (no implementadas en software) pueden implementarse en dispositivos de lógica programable (FPGA) y/o mediante técnicas VLSI. Según la forma de implementación de las redes se pueden clasificar principalmente en dos tipos: las implementadas mediante circuitos analógicos y

las basadas en sistemas digitales. Cabe remarcar que existe otro tipo de redes mixtas analógico/digitales (tratadas en el apartado anterior). Las redes neuronales artificiales (ANN) implementadas mediante circuitos analógicos tienen la ventaja de requerir un área mucho menor que sus equivalentes digitales, aunque su inmunidad al ruido y fiabilidad son más bien escasos. Asimismo, estos sistemas son difícilmente re-programables o re-configurables para realizar otra tarea que no sea para la cual han sido diseñados. Éste es el motivo por el cual las implementaciones analógicas se han centrado primordialmente en intentar emular las características y comportamientos de las neuronas biológicas [4-36, 4-65], o al menos algunas de sus características y/o formas de operación [4-36, 4-66].

En el otro lado hallamos las implementaciones de redes artificiales puramente digitales, que presentan un alto grado de fiabilidad, y son mucho más flexibles y fáciles de interconectar con los sistemas de computación. El número de transistores (VLSI) o en su caso los recursos lógicos (FPGA's) consumidos son muchísimo mayores que en las implementaciones analógicas. Esto se debe a que requieren de al menos un circuito multiplicador en punto flotante para evaluar la actividad presináptica de cada neurona, por no hablar de la implementación de la función de activación mediante un bloque "*look-at-table"*, o incluso mediante circuitos analógicos [4-64]. En el caso de las implementaciones digitales en dispositivos de lógica programable (FPGAs) tenemos que éstas están dotadas de capacidad de reprogramación, lo que unido a un ciclo de desarrollo extremadamente reducido y un precio razonable hacen que esta tecnología presente toda una serie de ventajas frente a las implementaciones digitales VLSI. Pero incluso con todas estas ventajas, las implementaciones de redes neuronales en (FPGA) totalmente paralelas requieren de una cantidad ingente de circuitos multiplicadores. Ya que cada conexión sináptica requiere de su multiplicador y el número de conexiones sinápticas aumenta con el cuadrado del número de neuronas. Por ejemplo, una capa oculta formada por 10 neuronas requiere de un total de 100 circuitos multiplicadores, mientras que si aumentamos a 1.000 se requiere de 1.000.000 de circuitos multiplicadores y así sucesivamente.

Para permitir implementaciones de redes neuronales hardware lo suficientemente grandes como para poder afrontar problemas reales únicamente será posible si el tamaño de la circuitería de multiplicación se ve drásticamente reducida. Esto se puede obtener de dos formas: reduciendo el número de multiplicadores (mediante sistemas *pipe-line*) y/o reduciendo la complejidad de los multiplicadores. Durante los últimos años se han propuesto toda una serie de soluciones basadas en el uso de codificaciones de la información en forma de trenes de pulsos [4-61, 4-62], a fin de consumir menos recursos hardware. A la vez, la comunidad científica [4-36] se ha focalizado en el estudio y la implementación de sistemas basados en neuronas pulsantes "*Spiking Neurons*" (descritas en el subapartado anterior) mucho más cercanas a las neuronas reales y que hacen uso de pulsos para la transmisión de la información.

Para implementar redes neuronales digitales clásicas normalmente se hace uso de un sólo circuito multiplicador serie por neurona, el cual mediante una técnica de "*time-division multiplexing*" (TDM) es capaz de procesar todas las señales provenientes de las entradas. De esta forma se consigue un aumento en la eficiencia de proceso por área de silicio (VLSI) o una reducción de los recursos lógicos consumidos (FPGA's) a cambio de perder velocidad de proceso.

Para la reducción del tamaño de los circuitos multiplicadores, D.E. Van Den Bout et al. [4-59] presentó en el año 1989 y posteriormente secundado por Kondo et al. [4-55] en el año 1992, propusieron una solución alternativa basada en el uso de la computación estocástica.

Posteriormente se han seguido realizando trabajos en esta línea [4-50, 4-53, 4-54, 4-55, 4-58], de los cuales destacamos los realizados por C.L. Janer et al. [4-51, 4-52] que fueron inmediatamente posteriores a los estudios iniciales. Como hemos visto en el segundo capítulo de la presente tesis, mediante la lógica estocástica (en cualquiera de sus codificaciones) podemos codificar magnitudes analógicas mediante trenes de pulsos, y se pueden realizar operaciones muy complejas mediante bloques digitales relativamente simples constituidos por muy pocos elementos lógicos (LE) [4-50, 4-51, 4-53, 4-63]. Por ejemplo, la multiplicación de dos señales estocásticas (en la codificación clásica sin signo) se implementa simplemente mediante una puerta **AND** de dos entradas. Este hecho hace que sea factible implementar redes neuronales densas, totalmente conectadas y que se

ejecuten en paralelo mediante bloques digitales básicos (puertas simples), obteniendo así un gran rendimiento del sistema.
A partir de la introducción del uso de la lógica estocástica en las redes neuronales, se han multiplicado el número de métodos que hacen uso de ella, como pueden ser: las memorias aleatorias probabilísticas "*Probabilistic RAM*" [4-67, 4-68], los métodos de aprendizaje de redes neuronales basadas en el uso de elementos aritméticos estocásticos [4-69, 4-70], y en el uso de autómatas celulares "*Cellular Automata*" [4-71] para la generación paralela de números pseudo aleatorios necesarios en las redes neuronales estocásticas. También es de remarcar el potencial de las redes neuronales estocásticas en el campo de las máquinas de Boltzman y Helmholtz [4-72, 4-73].

Los bloques aritméticos estocásticos se caracterizan por su tolerancia a las fluctuaciones de les entradas a la vez que su implementación es soportada y totalmente compatible con las modernas tecnologías de diseño y fabricación VLSI, así como por las diversas plataformas existentes de lógica programable (ALTERA, XILINX, ACTEL, etc.). A su vez, la lógica estocástica aporta toda una serie de beneficios frente a otras técnicas de computación, como pueden ser: la capacidad de realizar operaciones complejas mediante el uso de muy poca área o elementos lógicos, la posibilidad de lograr una gran conectividad, el uso de velocidades de reloj muy altas, así como la capacidad de modificar el tiempo de proceso modificando o la precisión de los circuitos sin ninguna modificación hardware. No obstante, existen dos inconvenientes fundamentales en la lógica estocástica que siempre debemos tener presentes, como son: una varianza inherente en las estimaciones de las señales estocásticas [4-74] y un retraso en la evaluación de la salida de los diferentes bloques. Ambas circunstancias están relacionadas entre si. Otro inconveniente, aunque menor es el rango de representación de magnitudes de las diferentes codificaciones [0, +1] ó [-1,+1]. Esto implicará que los pesos de entrenamiento de la red deberán reescalarse al valor del peso mayor a fin de poder representarse completamente.

No obstante el paralelismo que ofrece la lógica estocástica mediante la implementación de sistemas complejos en muy poca área o mediante muy pocos elementos lógicos (LE), hace

interesante afrontar y asumir todas sus desventajas. Debe remarcarse que las señales estocásticas y su aritmética son extremadamente inmunes al ruido o a la presencia de señales espurias, cosa que no sucede en los sistemas digitales clásicos. Todo ello unido a que en la lógica estocástica la precisión de las computaciones se controla mediante la dimensión temporal y no con el número de bits como sucede en las implementaciones clásicas.

Todos estos motivos son los que nos han empujado a desarrollar y evaluar nuestras propias implementaciones neuronales basadas en sistemas de computación estocásticos.

4.3.2 DESARROLLO DE NEURONAS ESTOCÁSTICAS

A partir de los fundamentos de la computación estocástica y del conjunto de bloques desarrollados en el segundo capítulo de la presente tesis, se han establecido las bases para la implementación de los diferentes modelos de neurona de segunda generación más usados. Estas neuronas se han desarrollado en hardware de forma simple y consumiendo muy pocos recursos hardware.

Los modelos de neurona de segunda generación tienen en cuenta que la salida de cada neurona (y_i) está relacionada con cada una de las entradas (x_{ij}), a través de la Ecuación [4-40]:

$$y_i = f\left(\sum_j \left(w_{ij} x_{ij} \right) + b_i \right) \qquad \text{Ecuación [4-40]}$$

Donde los coeficientes "w_{ij}" son los encargados de modelar el enlace sináptico de cada entrada, evaluando así la contribución postsináptica de cada una de las entradas presinápticas. El parámetro "b_i" representa el valor de Bias de la neurona y finalmente "$f(x)$" representa la función de activación. Esta función puede ser lineal o no en función del tratamiento de la información que se requiera en cada capa de la red. Cabe remarcar, que la función de activación o función de transferencia de una neurona *f(x)* es el único elemento que varía entre un tipo u otro de neurona artificial. Por ejemplo, si se pretende implementar una neurona lineal, la función de transferencia deberá ser la función identidad

(lo cual equivale a $y = f(x) = x$). En el caso que queramos implementar un perceptrón, la función de activación tendrá que ser una función escalón ($y = \Theta(x)$). Las funciones de activación no lineales permitirán implementar funcionalidades más complejas. Dichas funciones de transferencia pueden ser: las neuronas sigmoidales (función sigmoide), las Tangente sigmoidales (función tangente hiperbólica) y las RBF (función gaussiana). En la literatura podemos hallar algunos tipos adicionales de funciones de activación que las descritas anteriormente.

En los siguientes subapartados se presenta inicialmente la implementación de las partes genéricas a todos los modelos de neurona estocástica, para luego pasar a describir la combinación de estas partes con las funciones específicas para implementar alguno de los tipos de neuronas clásicas (segunda generación) más habituales como son: las neuronas lineales, Perceptrón y las Tangente-Sigmoidales.

4.3.2.1 Evaluación del potencial de membrana equivalente

Antes de pasar a describir la implementación particular de la función de activación para cada una de las neuronas anteriormente citadas, describiremos la implementación de la función suma (Ecuación [4-41]) de las contribuciones presinápticas que es común a todos los modelos de neuronas clásicas de segunda generación.

$$U_i = \sum_{j=1}^{k} w_{ij} \cdot x_{ij} + b_i = \begin{cases} \sum_{j=0}^{k} w_{ij} \cdot x_{ij} \\ w_{i0} = -b_i \end{cases} \qquad \text{Ecuación [4-41]}$$

Como bien sabemos por lo descrito a lo largo del presente capítulo, la implementación de redes neuronales requiere de la capacidad de representación de magnitudes mayores y menores que cero para poder codificar las contribuciones excitatorias e inhibitorias, las cuales integraremos mediante una función matemática que emula el potencial de membrana "U_i" (Ecuación [4-41]). Todo ello nos indica que para la implementación de neuronas y redes neuronales mediante la lógica estocástica requeriremos de una codificación que nos permita representar magnitudes positivas y negativas. Este hecho nos restringe al uso de la

codificación estocástica clásica con signo y a la lógica estocástica extendida con signo. Esta última se ha desarrollado especialmente para permitir solventar alguna de las limitaciones de la codificación estocástica clásica con signo en el campo de las implementaciones de redes neuronales clásicas.

A continuación, se presentan las implementaciones de los bloques digitales para la evaluación del potencial de membrana U_i resultante de la integración de todas las entradas presinápticas mediante el uso de las codificaciones estocásticas: clásica con signo y extendida con signo.

4.3.2.1.1 Evaluación del potencial de membrana mediante la codificación (SCSL)

Sí partimos de las ecuaciones [4-40 y 4-41] anteriormente descritas, podemos apreciar como la parte genérica de toda neurona "i" se compone realmente de dos elementos lineales: un primero formado por "j" multiplicadores estocásticos (tantos como conexiones sinápticas existentes) encargados de realizar el ajuste de cada una de las conexiones sinápticas mediante los pesos "w_{ij}", y de un elemento sumador global para realizar la función suma de las contribuciones post-sinápticas y del valor del nivel de *Bias* o de Offset de paso por cero.

A fin de ilustrar de forma clara y concisa el procedimiento de implementación de este bloque mediante la codificación estocástica clásica con signo (SCSL) se ha optado por la presentación de un ejemplo ilustrativo. Este ejemplo consistirá en presentar el bloque lineal (encargado de la evaluación del potencial de membrana U_i) de una neurona dotada de tres entradas presinápticas además de la señal de *Bias*.

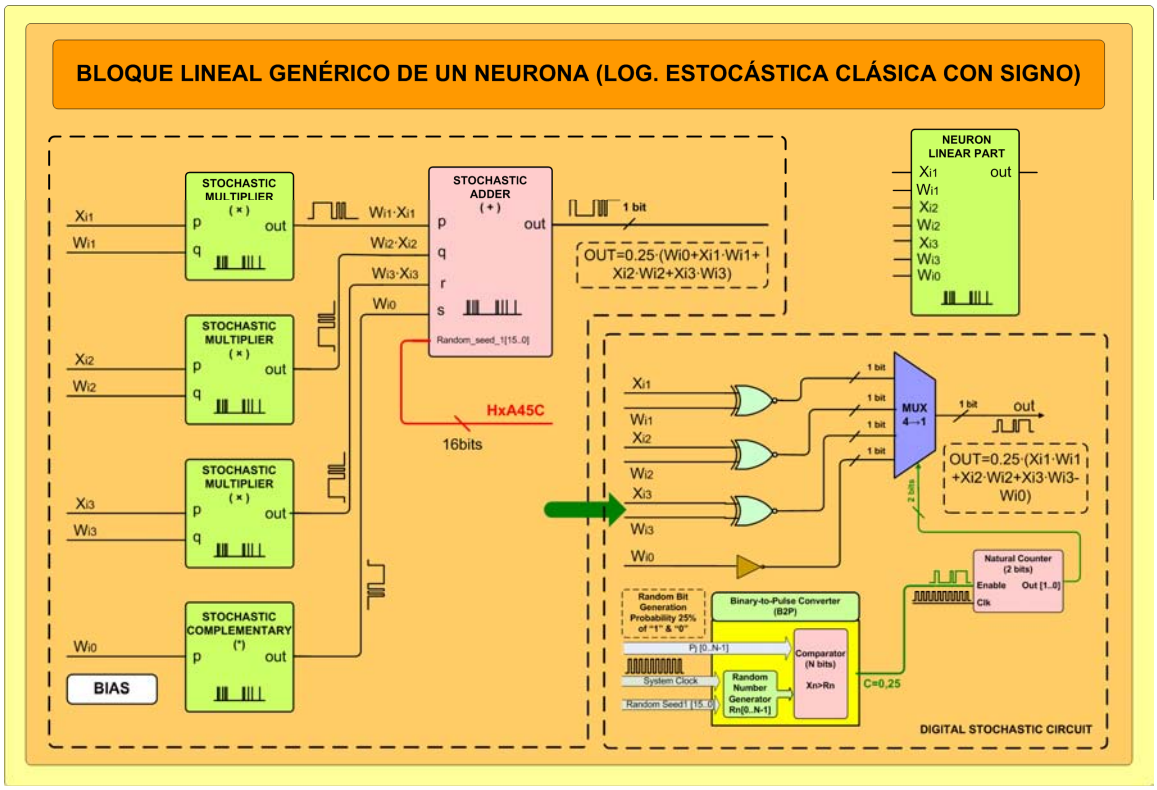

Figura 4-34: Esquemático del bloque lineal genérico de una neurona

La implementación estocástica del bloque lineal de una neurona de tres entradas se presenta en la Figura 4-34. En la parte izquierda de la figura se muestran los bloques que la componen. Inicialmente hallamos tres bloques multiplicadores encargados de evaluar la contribución postsináptica de cada una de las tres entradas (multiplicando la señal estocástica de entrada por su peso "$w_{ij} \cdot x_{ij}$"), así como un bloque inversor encargado de cambiar el signo de la contribución de la señal de *Bias*. La implementación digital de los anteriores bloques se muestra en la parte derecha de la Figura 4-34, donde se puede apreciar cómo la multiplicación de las señales de entrada se realiza con una simple puerta **XNOR** de dos entradas, mientras que la inversión de signo se realiza mediante una puerta **NOT**.

Una vez evaluadas todas las señales de salida de los anteriores bloques, éstas servirán de entrada a un bloque sumador estocástico (SCSL) de cuatro entradas que realizará la suma de las contribuciones de las entradas pulsantes. A la salida del bloque se obtiene la suma

ponderada de todas ellas (Ecuación [4-42]), dicha ponderación responde a un factor inversamente proporcional al número de sumandos "$1/n^{o}{}_{sumandos}$".

$$OUT = U_i = \left(\frac{1}{4}\right)\cdot\left(w_{i1}\cdot x_{i1} + w_{i2}\cdot x_{i2} + w_{i3}\cdot x_{i3} - w_{i0}\right) \qquad \text{Ecuación [4-42]}$$

La existencia de este factor de ponderación en el resultado (descrito en el capítulo 2, apartado 2.2.4.2) de la suma es un inconveniente, aunque este hecho se solventa mediante el uso de una función de activación con una ganancia interna prefijada para compensar dicho factor de compensación. En este caso la ganancia de la función de activación a usar debe ser de 4. La implementación digital del sumador estocástico dista un poco de la descrita en el segundo capítulo, en este caso, se debe proceder a la suma de 4 señales y no sólo 2. Para realizar esta suma se ha usado un multiplexor de cuatro entradas a una salida, donde la selección del canal a sumar, en vez de estar gobernada directamente por la señal estocástica, está controlada por un contador binario de 2-bits (módulo 3). La habilitación de conteo del contador, está gobernada por el generador de señales estocásticas (B2P) de 16-bits (con objeto de mantener la aleatoriedad en la suma de los componentes evitando así la aparición de coherencia entre las señales) configurado con una magnitud de valor $P_j = 16384$ que se corresponde con una señal activa el 25% del tiempo para un conversor pulsante de 16-bits.

En la Tabla 4-8 se presentan las medidas realizadas sobre el circuito estocástico (Figura 4-34) que implementa la parte lineal genérica a todas las neuronas de segunda generación. Para evaluar la correcta operación de dicho bloque se ha fijado el valor de las señales de entrada "$x_{i2} = x_{i3} = -0{,}125$", *Bias* "$w_{i0} = -0{,}50$", y todos los pesos de entrada de la red a "$w_{i1} = -0{,}875, w_{i2} = w_{i3} = -0.25$" y se ha procedido a variar el valor de la señal de entrada "x_{i1}" en el intervalo [-1, +0.875].

Tabla 4-8: Medidas experimentales del bloque lineal de un neurona de 4 entradas (SCSL)

X_{i0} [16bits] (Experimental)	X_{i0} Probabilidad equivalente	OUT=U_i [16bits] (Experimental)	OUT=U_i Probabilidad equivalente	OUT=U_i (Teórica)	Error ABS(Th-Exp)
0	-1,0000	36329	0,1087	0,1094	0,0007

4096	-0,8750	35482	0,0828	0,0820	0,0008
8192	-0,7500	34674	0,0582	0,0547	0,0035
12288	-0,6250	33530	0,0233	0,0273	0,0041
16384	-0,5000	32723	-0,0014	0,0000	0,0014
20480	-0,3750	31749	-0,0311	-0,0273	0,0038
24576	-0,2500	30891	-0,0573	-0,0547	0,0026
28672	-0,1250	30061	-0,0826	-0,0820	0,0006
32768	0,0000	29074	-0,1127	-0,1094	0,0034
36864	0,1250	28431	-0,1324	-0,1367	0,0044
40960	0,2500	27337	-0,1657	-0,1641	0,0017
45056	0,3750	26268	-0,1984	-0,1914	0,0070
49152	0,5000	25387	-0,2253	-0,2188	0,0065
53248	0,6250	24354	-0,2568	-0,2461	0,0107
57344	0,7500	23643	-0,2785	-0,2734	0,0050
61440	0,8750	22789	-0,3045	-0,3008	0,0038

En la Figura 4-35 representamos los resultados experimentales (símbolos) presentados en la Tabla 4-8, así como los predichos por la teoría (línea roja continua). A su vez, hemos realizado un ajuste por mínimos cuadrados (línea verde continua) de los datos experimentales.

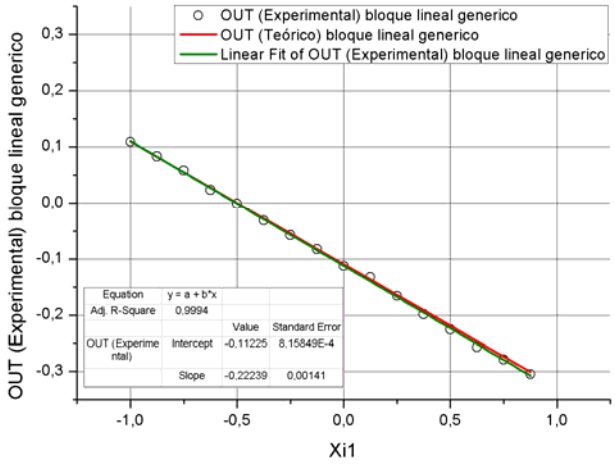

Figura 4-35: Resultados experimentales del bloque lineal de una neurona (SCSL)

Si comparamos la función de transferencia obtenida en la Figura 4-35 mediante el ajuste por mínimos cuadrados con la predicha por la teoría (Ecuación [4-42]), podemos afirmar que el sistema se comporta como cabe esperar, ya que ambas expresiones son prácticamente idénticas (Ecuación [4-43]).

$$Teórica \rightarrow OUT = U_i = \left(\frac{1}{4}\right)\cdot\left(w_{i1}\cdot x_{i1} + w_{i2}\cdot x_{i2} + w_{i3}\cdot x_{i3} - w_{i0}\right) = -0{,}2188\cdot x_{i1} - 0{,}1094$$

$$Experimental \rightarrow OUT = U_i = -0{,}2224\cdot x_{i1} - 0{,}1123 \rightarrow \Re^2 = 0{,}9994$$

Ecuación [4-43]

Como bien podemos apreciar en la Ecuación [4-43], la neurona lineal implementada mediante la codificación clásica con signo (SCSL) subestima el valor teórico de dicha neurona en un factor "*1/#sumandos*", lo que se traduce en que deberíamos reescalar los pesos de entrada un factor "k=nº de sumandos". En el caso de la codificación estocástica clásica con signo no se pueden representar magnitudes fuera del intervalo [-1,+1]. Si los pesos obtenidos originalmente mediante herramientas matemáticas estandarizadas (MATLAB, DTREG) para el entrenamiento de estas redes son mayores que el rango de representación de la lógica, tendremos un serio problema, ya que deberemos reescalar los pesos del sistema así como los valores de entrada. Esto implica que la implementación de neuronas estocásticas mediante la codificación clásica con signo es dificilmente generalizable y sistematizable.

4.3.2.1.2 Evaluación del potencial de membrana mediante la codificación (SESL)

Se ha implementado el bloque de evaluación del potencial de membrana mediante la codificación estocástica extendida con signo (SESL) que hemos desarrollado específicamente a fin de solventar los inconvenientes inherentes a la codificación estocástica clásica, como son la imposibilidad de representar magnitudes fuera del rango [-1,+1]. Este hecho implica no poder disponer de neuronas estocásticas lineales que puedan ser entrenadas mediante técnicas estándares, sino que requieren de técnicas ad-hoc que tengan en cuenta los factores de ponderado en la arquitectura de red.

Codificando las magnitudes binarias mediante la razón entre dos señales estocásticas (P/Q), podremos representar valores en el rango (-∞,+∞). En contrapartida, la desventaja inherente de esta metodología ésta relacionada con el aumento del error de representación (a medida que la razón (P/Q) crece).

Si partimos de las Ecuaciones [4-40 y 4-41], la parte genérica de toda neurona "i" se compone de dos elementos lineales: el primero formado por "j" multiplicadores estocásticos encargados de realizar el ajuste de cada una de las conexiones sinápticas mediante los pesos w_{ij_p}/w_{ij_q} y de un elemento sumador global de "j" sumandos o uno constituido por "j-1" sumadores simples encadenados para realizar la función suma de las contribuciones post-sinápticas y del valor del nivel de *Bias* o de *Offset*. Las ventajas e inconvenientes para realizar la suma de una forma u otra, se aborda más adelante puesto que, matemáticamente, ambas formas obtienen el mismo resultado, pero la degeneración de la señal estocástica es muy diferente.

Como ejemplo mostramos un bloque lineal encargado de la evaluación del potencial de membrana "U_i" de una neurona dotada de dos entradas presinápticas además de la señal de *Bias*.

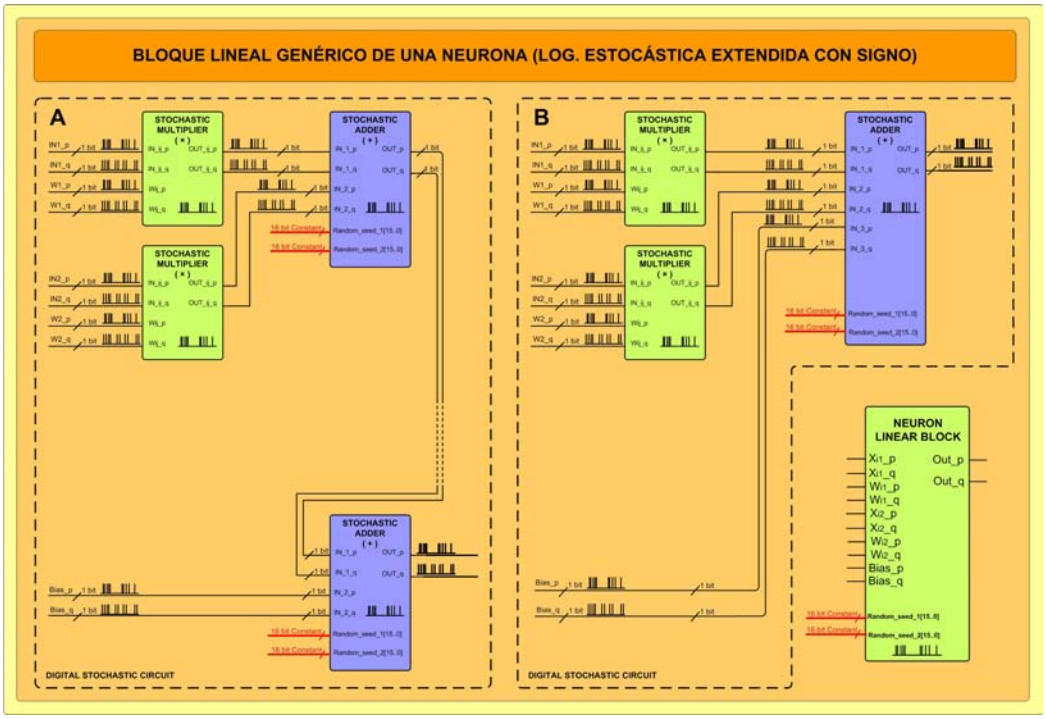

Figura 4-36: Esquema genérico del bloque lineal de una neurona de 3 entradas (SESL)

Las dos posibles implementaciones estocásticas del núcleo lineal de una neurona (A y B) se presentan en la Figura 4-36. En ella podemos apreciar que tanto en la implementación A como en la B aparecen un conjunto de tres bloques multiplicadores extendidos encargados

de evaluar la contribución post sináptica de cada una de las tres entradas (multiplicando la señal estocástica de entrada por su peso ($w_{ij\_p}.x_{ij\_q}$)/($w_{ij\_q}.x_{ij\_q}$), así como un bloque inversor extendido para cambiar el signo de la contribución de la señal de *Bias* ($bias_p$/ $bias_q$). La implementación digital de los bloques anteriormente descritos se presenta en la Figura 4-38, donde se puede apreciar cómo la multiplicación del numerador y del denominador de las señales de entrada se realiza mediante sendas puertas **XNOR** de dos entradas, mientras que la inversión de signo se realiza a su vez mediante una pareja de puertas **NOT**.

No obstante, las dos implementaciones propuestas (A y B) se diferencian a la hora de realizar la suma de las contribuciones post sinápticas individuales. En la opción A, para realizar la suma de "n" sumandos se requiere de "m=n-1" bloques sumadores individuales de dos entradas, con la ventaja de permitir una fácil automatización de la generación del bloque sumador de "n" sumandos a partir de "m" sumadores simples de dos entradas. Las señales de salida para el numerador y el denominador de la suma para el caso de la neurona de tres entradas anteriormente descrita vendrá regida por la siguiente Ecuación [4-44]:

$$\begin{cases} \begin{cases} w_{i0} = \dfrac{w_{i0\_p}}{w_{i0\_q}} \\ w_{i1} = \dfrac{w_{i1\_p}}{w_{i1\_q}} \\ w_{i2} = \dfrac{w_{i2\_p}}{w_{i2\_q}} \end{cases} \rightarrow \begin{cases} x_{i1} = \dfrac{x_{i1\_p}}{x_{i1\_q}} \\ x_{i2} = \dfrac{x_{i2\_p}}{x_{i2\_q}} \end{cases} \rightarrow \begin{cases} a = \left(w_{i1\_q} \cdot x_{i1\_q}\right) \cdot \left(w_{i2\_q} \cdot x_{i2\_q}\right) \\ b = \left(w_{i0\_q}\right) \cdot \left(w_{i2\_q} \cdot x_{i2\_q}\right) \\ c = \left(w_{i0\_q}\right) \cdot \left(w_{i1\_q} \cdot x_{i1\_q}\right) \end{cases} \rightarrow \\ \begin{cases} OUT = u_i = \displaystyle\sum_{j=0}^{n-1} w_{ij} \cdot x_{ij} = \dfrac{u_{i\_p}}{u_{i\_q}} = \dfrac{\left(1/2\right)^{n-1} \cdot \left(w_{i0\_p} \cdot a + w_{i1\_p} \cdot b + w_{i2\_p} \cdot c\right)}{\left(1/2\right)^{n-1} \cdot \left(\left(w_{i0\_q}\right) \cdot \left(w_{i1\_q} \cdot x_{i1\_q}\right) \cdot \left(w_{i2\_q} \cdot x_{i2\_q}\right)\right)} \\ n = 3 \end{cases} \end{cases}$$

Ecuación [4-44]

De la Ecuación [4-44] se desprende que la función suma obtiene la expresión correcta desde el punto de vista matemático, pero aparece tanto en el numerador y en el denominador un factor de proporcionalidad (0,5^n).

Mediante el circuito "B" de la Figura 4-36, para realizar la suma de "n" sumandos se ha implementado un circuito sumador específico (Figura 4-37), el cual es algo más complejo que el anteriormente descrito. Donde las señales de salida para el numerador y

denominador de la suma para el caso de la neurona de tres entradas se regirán por la siguiente Ecuación [4-45]:

$$\begin{cases} OUT = U_i = \sum_{j=0}^{n-1} w_{ij} \cdot x_{ij} = \dfrac{U_{i\_p}}{U_{i\_q}} = \dfrac{\left(1/n\right)\cdot\left(w_{i0\_p}\cdot a + w_{i1\_p}\cdot b + w_{i2\_p}\cdot c\right)}{\left(1/n\right)\cdot\left(\left(w_{i0\_q}\right)\cdot\left(w_{i1\_q}\cdot x_{i1\_q}\right)\cdot\left(w_{i2\_q}\cdot x_{i2\_q}\right)\right)} \\ n = 3 \end{cases}$$

Ecuación [4-45]

De la Ecuación [4-45] se desprende que la función suma implementada mediante un sumador estocástico específico se corresponde con la expresión correcta para dicha suma, a la vez que aparece un factor de proporcionalidad (1/n) tanto en el numerador como en el denominador.

Tabla 4-9: Factores de ponderación en función del número de sumandos (A y B) (SESL)

Número de sumandos	Factor de ponderación (implementación B)	Factor de ponderación (implementación A)
2	0,5000	0,5000
5	0,2000	0,0625
10	0,1000	0,0020
20	0,0500	1,9 E-6
30	0,0333	1,9·E-9

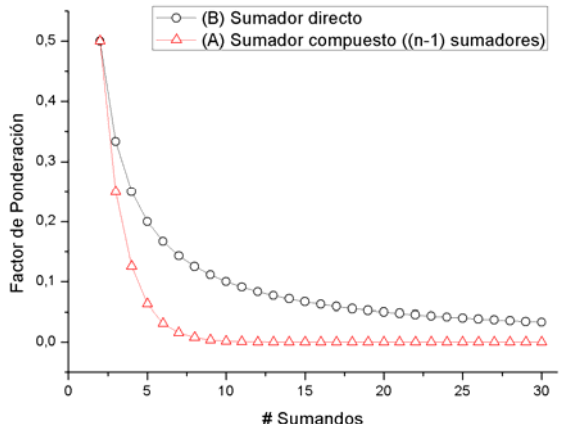

Figura 4-37: Factor de ponderación en función del número de sumandos (Implementaciones del sumador A y B)

Por lo tanto, en función de los resultados obtenidos en la Tabla 4-9 (Figura 4-37) para las implementaciones “A” y “B” se puede afirmar que el factor de ponderación se hará

prácticamente cero a medida que se incrementa el número de sumandos, produciendose así una degradación del valor de las señales estocásticas. En consecuencia, la forma más óptima de realizar la suma es mediante la implementación “B”.

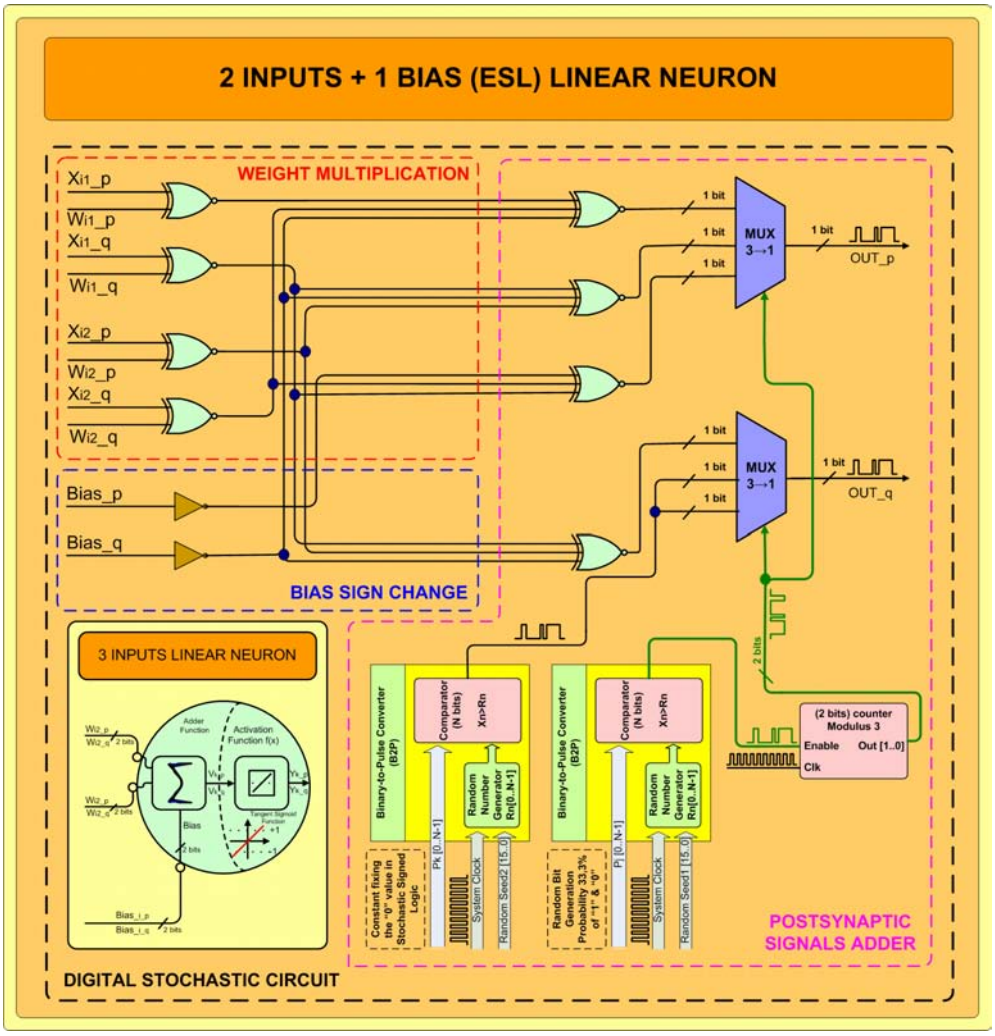

Figura 4-38: Implementación estocástica del bloque lineal extendido de una neurona de 3 entradas

En el caso “B”, al tener asignado la señal del numerador y la del denominador el mismo factor de ponderación, éstos se compensan obteniendo a la salida el valor exacto de la suma. Por lo tanto, no requeriremos a priori de ningún factor de ponderación de la función de activación presente en la neurona. Para la realización de esta suma se han usado dos multiplexores ($3IN \rightarrow 1OUT$), uno para el numerador y el otro para el denominador. La

selección del canal a sumar, en vez de estar gobernada directamente por la señal estocástica, está controlada por un contador binario módulo 3 de 2-bits cuya señal de habilitación de conteo está controlada por el generador de señales estocásticas (B2P) (con objeto de mantener la aleatoriedad en la suma de los componentes evitando así la aparición de coherencia entre las señales). Éste está configurado con un valor de ((2^16)/3), que se corresponde con una señal activa el 33,3% del tiempo.

En la Tabla 4-10 se presentan las medidas realizadas sobre el circuito estocástico (Figura 4-38) que implementa la parte genérica lineal presente en todas las neuronas de segunda generación. Para su realización hemos fijamos el valor de la señal de entrada x_{i1_p}/x_{i1_q}*=1*, *Bias* w_{i0_p}/w_{i0_q}*=-0,081067 y* todos los pesos de entrada de la red a los siguientes valores w_{i1_p}/w_{i1_q}*=-0,58052* y w_{i2_p}/w_{i2_q}*=-1,1766.* Finalmente procederemos a variar el valor de la señal de entrada x_{i1_p}/x_{i2_q} en el intervalo [-1.117, +1.333].

Tabla 4-10: Medidas experimentales del bloque lineal de una neurona de 3 entradas (SESL)

x_{i2_p} [16bits]	x_{i2_q} [16bits]	x_{i2_p}/x_{i2_q} Probabilidad equivalente	U_{i_p} [16bits]	U_{i_q} [16bits]	U_{i_p}/U_{i_q} Probabilidad equivalente	U_i Teórica	Error (Th-Exp)
0	8192	1,3333	20936	37728	-2,3855	-2,2304	0,1551
4096	8192	1,1667	22115	38100	-1,9979	-1,9532	0,0447
8192	8192	1,0000	23067	38034	-1,8422	-1,7571	0,0851
12288	8192	0,8333	24164	38121	-1,6073	-1,5610	0,0463
16384	8192	0,6667	25166	37869	-1,4903	-1,3649	0,1254
20480	8192	0,5000	26175	38115	-1,2330	-1,1688	0,0642
24576	8192	0,3333	27130	38207	-1,0366	-0,9727	0,0638
28672	8192	0,1667	28337	37985	-0,8493	-0,7766	0,0727
32768	8192	0,0000	29402	37985	-0,6452	-0,5806	0,0646
36864	8192	-0,1667	30419	37999	-0,4491	-0,3845	0,0646
40960	8192	-0,3333	31240	38075	-0,2879	-0,1884	0,0996
45056	8192	-0,5000	32217	37918	-0,1070	0,0077	0,1147
49152	8192	-0,6667	33212	37894	0,0866	0,2038	0,1172
53248	8192	-0,8333	34449	37910	0,3269	0,3999	0,0730
57344	8192	-1,0000	35267	37913	0,4857	0,5960	0,1103
61440	8192	-1,1667	36457	37870	0,7230	0,7921	0,0690

Posteriormente, hemos procedido a representar gráficamente (Figura 4-39) los resultados experimentales (símbolos) presentados en la Tabla 4-10. Así como los predichos por la teoría (línea continua roja), a su vez hemos realizado un ajuste por mínimos cuadrados de

los datos experimentales (línea verde continua) para determinar la función de transferencia de la implementación digital realizada.

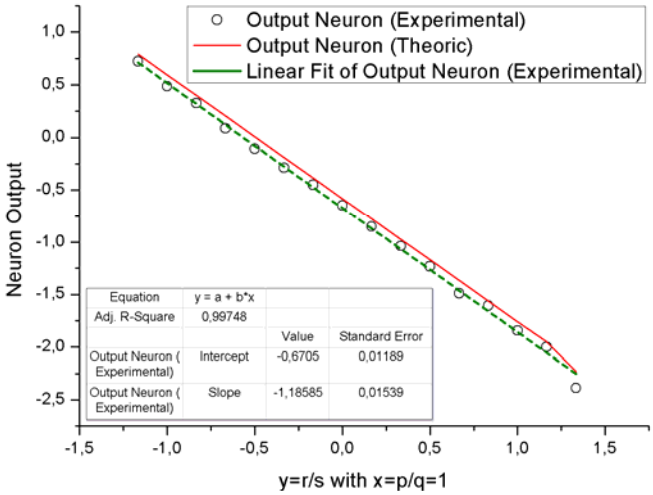

Figura 4-39: Resultados experimentales del bloque lineal extendido de una neurona de 3 entradas

Si ahora comparamos la función de transferencia obtenida de la Figura 4-39 mediante un ajuste por mínimos cuadrados con la predicha por la teoría (Ecuación [4-45]), podemos afirmar que ambos se comportan de forma muy semejante, aunque entre ambas expresiones (Ecuación [4-46]) existe un cierto Offset derivado del error de cuantización inherente a esta codificación (descrito en el capítulo segundo de la presente tesis).

$$Teórica \rightarrow OUT = U_i = -1{,}1873 \cdot x_{i1} - 0{,}5847$$
$$Experimental \rightarrow OUT = U_i = -1{,}1859 \cdot x_{i1} - 0{,}6705 \rightarrow \Re^2 = 0{,}9975$$

Ecuación [4-46]

4.3.2.2 La neurona perceptrón

En 1958, el psicólogo Frank Ronsenblant [4-4] desarrolló un modelo simple de neurona basado en el modelo de McCulloch y Pitts [4-1], a dicho modelo se le llamó *perceptrón*. Una de las características que presentaba este modelo fue su capacidad de aprendizaje para reconocer patrones. Esto es debido a que la neurona *Perceptrón* internamente se compone

de un bloque lineal, y una función de activación de salida se encarga de clasificar entre dos clases en función de su salida.

La principal característica de este modelo simple de neurona es su capacidad de realizar tareas de clasificación de forma automática, a partir de la aplicación de un patrón formado por un conjunto de muestras $x=\{x_1,x_2,x_3,\ldots,x_n\}$, cada de ellas emparejada con su salida deseada ($z=1$ ó $z=-1$). La salida de la neurona $z=f(x)$ vendrá regida por la función signo o sesgo operada sobre la suma ponderada de las entradas, de esta forma, la función de transferencia de una neurona *Perceptrón* se regirá por la ecuación [4-47]:

$$\begin{cases} z_i = f(x_1, x_2, ..., x_n) = f(x) = \text{sgn}(u_i(x) - \theta) = \begin{cases} +1 \rightarrow u_i = \sum_{j=0}^{n} w_{ij} \cdot x_{ij} \geq 0 \\ -1 \rightarrow u_i = \sum_{j=0}^{n} w_{ij} \cdot x_{ij} < 0 \end{cases} \\ \begin{cases} \theta = -w_{i0} \\ x_{i0} = +1 \end{cases} \end{cases}$$

Ecuación [4-47]

Donde $u_i(x)$ se corresponde con el potencial sináptico o de membrana, "θ" es el umbral de disparo de la neurona y "$\text{sgn}(x)$" es la función signo.

De la ecuación [4-47] se desprende que el *Perceptrón* simple define un híperplano de dimensión "n-1" capaz de separar dos clases (cada una de ellas representada por un valor de la salida [-1,+1] de la neurona). La ecuación del híperplano para una neurona de dos entradas viene definida por la siguiente Ecuación [4-48]:

$$\begin{cases} w_{i1} \cdot x_{i1} + w_{i2} \cdot x_{i2} - w_{i0} \cdot 1 = 0 \\ \theta = -w_{i0} \\ x_{i0} = 1 \end{cases} \rightarrow x_{i3} = -\frac{w_{i1}}{w_{i3}} \cdot x_{i1} + \frac{w_{i0}}{w_{i3}} \cdot 1 \qquad \text{Ecuación [4-48]}$$

Donde la parte que multiplica "x_{i1}" se corresponde con la pendiente de la recta mientras que la contribución adicional se corresponde con el punto de corte con el eje de coordenadas.

El modelo de neurona *Perceptrón* simple tiene algunas limitaciones como son la imposibilidad de resolver problemas que no sean linealmente separables. Un buen ejemplo

de ello, es su incapacidad de aprender la función lógica XOR. A razón de este hecho, esta neurona pasó a un segundo plano hasta la aparición de la regla de aprendizaje de retro-propagación del error "*Back-Propagation*" [4-9, 4-18].

Una vez descrita la función de transferencia y las características más significativas de este tipo de neuronas, pasaremos a describir su implementación mediante la codificación estocástica clásica con signo y la extendida con signo.

4.3.2.2.1 Implementación de la neurona perceptrón mediante la codificación (SCSL)

La implementación digital de la neurona Perceptrón mediante la codificación estocástica clásica con signo (SCSL) se presenta en la Figura 4-40.

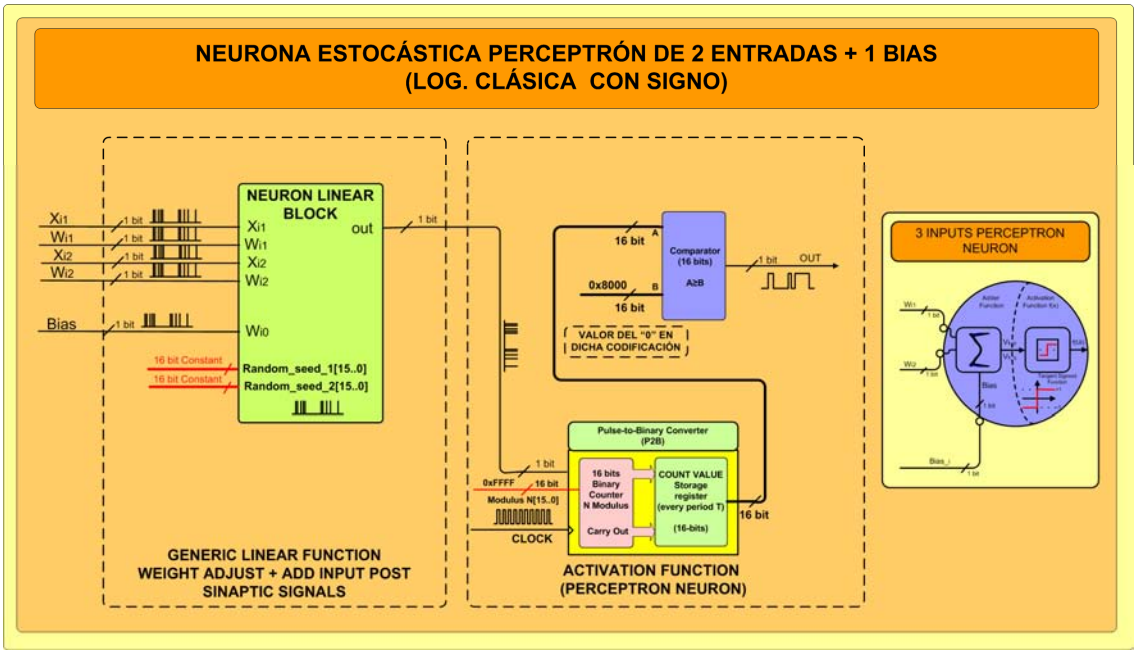

Figura 4-40: Implementación del Perceptrón de 2 entradas + 1 Bias (SCSL)

Como puede apreciarse en la figura anterior, la neurona perceptrón se compone de un bloque lineal genérico encargado de realizar la suma ponderada de las entradas (descrito anteriormente en el apartado 4.3.2.1.1), la salida del cual se ha interconectado con un

bloque estocástico que implementa la función signo (*Sgn(x)*) como función de activación de la neurona Perceptrón.

Para implementar dicha función se ha usado un bloque P2B de 16-bits a fin de integrar a lo largo de un período de evaluación "$T_{eval} = 2^{16} \cdot T_{clock}$" la salida del bloque lineal. A continuación, se compara el valor medio de la distribución medida con el valor del cero en esta codificación estocástica (en este caso se corresponde con "Zero_value=(2^16)/2=32768"). El problema de la subestimación de la suma en el bloque lineal (descrita en el apartado 4.3.2.1.1) no es ningún problema, ya que lo único importante desde el punto de vista operacional es dónde se haya el cero, y éste, pese a la subestimación, se sigue hallando en su posición correcta, con lo cual este hecho no afecta en nada la correcta operación de este tipo de neurona. En la práctica se ha usado un comparador de 16-bits, que fijará su salida a valor "1" si el valor integrado del bloque lineal se corresponde a un valor del potencial de membrana $u_i(t) \geq 0$, mientras que fijará su salida a "0" (que equivale a -1 en esta codificación) en el caso contrario.

Para probar el correcto funcionamiento del diseño propuesto de neurona perceptrón, hemos procedido a implementar en una FPGA de Altera Corp. una neurona Perceptrón de dos entradas con una señal adicional de *Bias*. La configuración de la neurona implementada ha consistido en fijar el valor de las señales de entrada "$x_{i2} = -0{,}125$", *Bias* "$w_{i0} = -0{,}50$", a la vez que se han fijado todos los pesos de entrada de la red a "$w_{i1} = -0{,}875$" y "$w_{i2} = -0.25$", y finalmente hemos procedido a variar el valor de la señal de entrada "x_{i1}" en el intervalo [-1,`+0.875]. Los resultados obtenidos se presentan en la Tabla 4-11.

Tabla 4-11: Medidas experimentales de la neurona Perceptrón (SCSL)

X_{i0} [16bits] (Experimental)	X_{i0} Probabilidad equivalente	OUT=U_i (lineal) [16bits] (Experimental)	OUT=U_i (lineal) Probabilidad equivalente (Experimental)	OUT=U_i (lineal) (Teórica)	OUT Perceptrón (Experimental)
0	-1,0000	37046	0,1306	0,1354	1
4096	-0,8750	36207	0,1049	0,0990	1
8192	-0,7500	34430	0,0507	0,0625	1
12288	-0,6250	33610	0,0257	0,0260	1
14336	-0,5625	33146	0,0115	0,0078	1
16384	-0,5000	32456	-0,0095	-0,0104	-1
18432	-0,4375	32042	-0,0222	-0,0286	-1

20480	-0,3750	31205	-0,0477	-0,0469	-1
24576	-0,2500	30180	-0,0790	-0,0833	-1
28672	-0,1250	28826	-0,1203	-0,1198	-1
32768	0,0000	27882	-0,1491	-0,1563	-1
36864	0,1250	26236	-0,1993	-0,1927	-1
40960	0,2500	25329	-0,2270	-0,2292	-1
45056	0,3750	24177	-0,2622	-0,2656	-1
49152	0,5000	22633	-0,3093	-0,3021	-1
53248	0,6250	21659	-0,3390	-0,3385	-1
57344	0,7500	20605	-0,3712	-0,3750	-1
61440	0,8750	19472	-0,4058	-0,4115	-1

Si comparamos los resultados obtenidos mediante la implementación estocástica realizada y los predichos por la teoría, podemos concluir que ambos son idénticos. No obstante, debe tenerse en cuenta que la señal del bloque lineal es integrada por un bloque *P2B(N)*, el cual tiene asociada a su salida una distribución binomial con una dispersión asociada "$\sigma = \sqrt{N \cdot p \cdot (1-p)}$". La señal de salida del bloque *P2B(N)* se conecta a la entrada (A) del comparador, que la compara con el valor del '0' (en dicha codificación). La salida de este bloque tendrá asociada una cierta dispersión a la hora de cortar el eje de abscisas o de ordenadas. Para finalizar con este subapartado hemos procedido a representar gráficamente (Figura 4-41) la salida de la neurona Perceptrón en función de la probabilidad de la señal "X_{i1}" de entrada (Tabla 4-10).

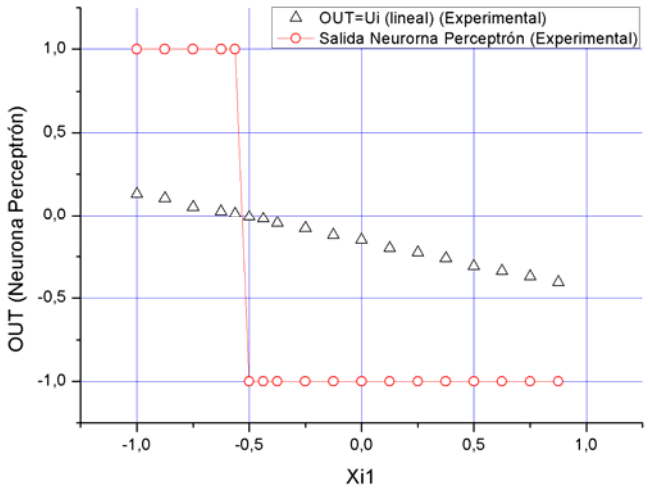

Figura 4-41: Resultados experimentales de la neurona Perceptrón (SCSL)

4.3.2.2.2 Implementación de la neurona perceptrón mediante la codificación (SESL)

La implementación digital de la neurona perceptrón mediante la lógica estocástica extendida con signo (SESL) es bastante parecida a la detallada anteriormente para la codificación estocástica clásica con signo (SCSL), como puede apreciarse en la Figura 4-42.

Esta neurona se compone de un bloque lineal genérico encargado de realizar la suma ponderada de las entradas (descrito anteriormente en el apartado 4.3.2.1.2) y de un bloque estocástico (ESL) que implementa la función signo (*Sgn(x)*) como función de activación de la neurona.

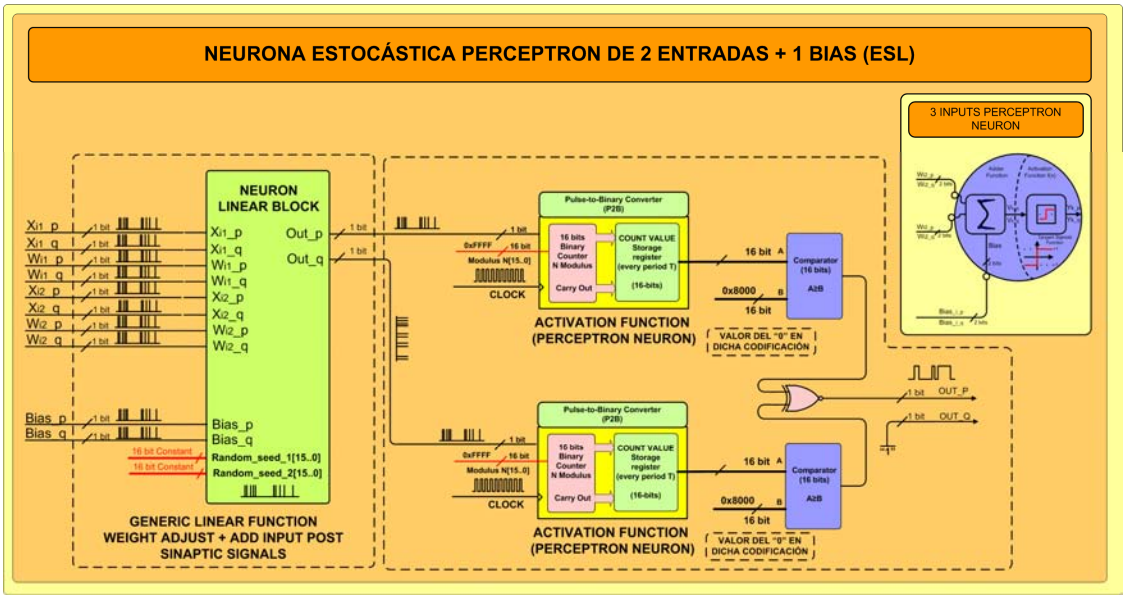

Figura 4-42: Implementación de la neurona Perceptrón de dos entradas + 1 Bias (SESL)

Como puede apreciarse en la Figura 4-42, para implementar la función signo en (SESL) se han usado dos bloques P2B de 16-bits (independientes) para integrar a lo largo de un período de evaluación "$T_{eval}=2^{16}T_{clock}$" las salidas pulsantes del numerador y del denominador. De esta forma se comparará el valor medio de la distribución medida con el valor del cero en esta codificación estocástica (en este caso se corresponde también a

"Zero_value=(2^16)/2=32768"). Finalmente, cada salida de los bloques P2B(N) (16-bits) se conectarán a la entrada (A) de su correspondiente bloque comparador que comparará dicho valor con el del cero (B). Los comparadores usados fijan su salida a "1" si el valor integrado del bloque lineal se corresponde a un valor del numerador/denominador del potencial de membrana "$u_i(t) \geq 0$", mientras que fijarán su salida a "0" (que equivale a -1 en esta codificación) en el resto de casos. En consecuencia, en esta implementación dispondremos de un comparador para el numerador y otro para el denominador, cuyas salidas se fusionarán mediante una puerta **XNOR** obteniendo así la señal del numerador de salida del bloque, mientras que el denominador se fija directamente a nivel alto "+1".
La función de la puerta **XNOR** es asegurar que la señal del numerador se fije a "+1" si el valor de numerador son ambos a la vez mayores a cero o menores a cero, y fijar la salida a "-1" para el resto de los casos.

Para comprobar la funcionalidad de la implementación propuesta se ha procedido a implementar una neurona *Perceptrón* de tres entradas con una señal adicional de *Bias en una FPGA* de Altera Corp. Para la realización de las medidas se ha partido del ejemplo ya descrito en el apartado (4.3.2.1.2) al cual se le han incorporado la función de activación signo *(Sgn(x))* desarrollada. Para su evaluación se ha fijado el valor de la señal de entrada "$x_{i1\_p}/x_{i1\_q} = +1$", *Bias* "$w_{i0\_p}/w_{i0\_q} = -0{,}081067$", y todos los pesos de entrada de la red a "$w_{i1\_p}/w_{i1\_q} = -0.58052$" y "$w_{i2\_p}/w_{i2\_q} = -1.1766$". A su vez, se ha procedido a variar el valor de la señal de entrada "x_{i2_p}/x_{i2_q}" en el intervalo [-1.117, +1.333]. Los resultados obtenidos son los que se presentan en la Tabla 4-12.

Tabla 4-12: Medidas experimentales de la neurona Perceptrón 3 entradas (SESL)

x_{i2_p} [16bits]	x_{i2_q} [16bits]	x_{i2_p}/x_{i2_q} Probabilidad equivalente	U_{i_p}/U_{i_q} (Experimental) (Bloque lineal)	U_i Teórica (Bloque lineal)	Error (Bloque lineal) (Th-Exp)	OUT Perceptrón (Experimental)	OUT Perceptrón (Teórica)
0	8192	1,3333	-2,3855	-2,2304	0,1551	-1	-1
4096	8192	1,1667	-1,9979	-1,9532	0,0447	-1	-1
8192	8192	1,0000	-1,8422	-1,7571	0,0851	-1	-1
12288	8192	0,8333	-1,6073	-1,5610	0,0463	-1	-1
16384	8192	0,6667	-1,4903	-1,3649	0,1254	-1	-1
20480	8192	0,5000	-1,2330	-1,1688	0,0642	-1	-1

24576	8192	0,3333	-1,0366	-0,9727	0,0638	-1	-1
28672	8192	0,1667	-0,8493	-0,7766	0,0727	-1	-1
32768	8192	0,0000	-0,6452	-0,5806	0,0646	-1	-1
36864	8192	-0,1667	-0,4491	-0,3845	0,0646	-1	-1
40960	8192	-0,3333	-0,2879	-0,1884	0,0996	-1	-1
45056	8192	-0,5000	-0,1070	0,0077	0,1147	-1	+1
49152	8192	-0,6667	0,0866	0,2038	0,1172	+1	+1
53248	8192	-0,8333	0,3269	0,3999	0,0730	+1	+1
57344	8192	-1,0000	0,4857	0,5960	0,1103	+1	+1
61440	8192	-1,1667	0,7230	0,7921	0,0690	+1	+1

Si comparamos los resultados obtenidos mediante la implementación estocástica realizada (SESL) y los predichos por la teoría, podemos concluir que ambos son prácticamente idénticos. Debemos tener muy presente que la pareja de bloques *P2B(N)* usados tienen asociada a su salida una distribución binomial con una dispersión asociada "$\sigma = \sqrt{N \cdot p \cdot (1-p)}$". Para finalizar con este subapartado hemos procedido a representar gráficamente la salida de la neurona Perceptrón (Figura 4-43) en función de la probabilidad de la señal "X_{i1}" de entrada (Tabla 4-12).

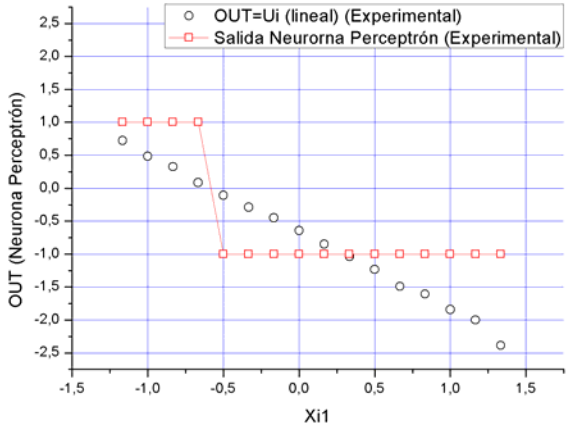

Figura 4-43: Resultados experimentales de la neurona Perceptrón de 3 entradas (SESL)

4.3.2.3 La neurona lineal

Una neurona lineal se define como aquella que su salida es linealmente dependiente a sus entradas. Con todo ello, la función de transferencia de este tipo de neuronas se describe mediante la Ecuación [4-49]:

$$\begin{cases} umbral/bias = -w_{n0} \\ x_{n0} = 1 \end{cases} \rightarrow salida_n = U_n = \sum_{j=0}^{k} w_{nj} \cdot x_{nj} \quad \text{Ecuación [4-49]}$$

Una neurona lineal (Figura 4-44) permite que su salida pueda tomar un valor continuo en lugar de uno bivalente como sucede en el caso de una neurona *Perceptrón*. Este tipo de neuronas fueron inicialmente desarrolladas por Widrow-Hoff en 1959 [4-5] en su red neuronal de una capa llamada Adeline, para la cual desarrollaron una regla de aprendizaje llamada de Widrow-Hoff o (LMS) (regla de los mínimos cuadrados o regla delta) que actualmente se sigue usando para evaluar el valor de los pesos óptimos proporcionalmente al valor del error de una red constituida por neuronas lineales.

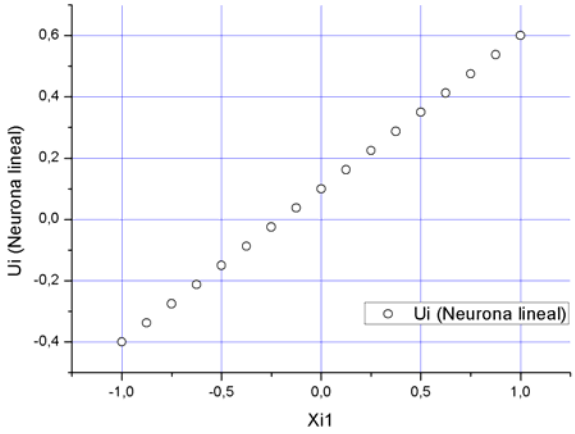

Figura 4-44: Función de transferencia de una neurona lineal de una entrada '$x_{i1} \epsilon [-1,+1]$' con un peso '$W_{i1}=0,5$' y una Bias '$W_{i0}=0,1$'

La función de transferencia lineal solo permite resolver problemas linealmente separables, aunque este tipo de neuronas inherentemente tienen asociados ciertos problemas como son una falta de persistencia en las respuestas (de modo que cambios muy pequeños en las entradas pueden producir fluctuaciones grandes en sus respuestas). No obstante, este tipo de neuronas se pueden combinar con otros tipos para formar una red neuronal mayor que explote alguna de sus ventajas.

En la práctica, las neuronas lineales se caracterizan por disponer como función de activación la identidad, lo que equivale a que se hallan constituidas únicamente por el

bloque lineal que evalua el potencial de membrana a partir de las contribuciones pre-sinápticas, siendo su salida la suma de éstas. Con todo, ello la implementación digital de las neuronas lineales ya ha sido abordada para la codificación estocástica clásica con signo (SCSL) en el apartado 4.3.2.1.1, y para la codificación estocástica extendida con signo (SESL) en el apartado 4.3.2.1.2.

Tan sólo recordar que las neuronas lineales implementadas mediante la codificación (SCSL) tienen el gran inconveniente de que su salida subestima el valor de la suma de las contribuciones presinápticas un factor (1/n), siendo "n" el número de sumandos. La única solución para ello la multiplicación de los pesos por un factor "n", lo que implicará que en algunos casos el valor del peso a codificar excederá el rango de representación de la codificación (SCSL), lo que a su vez provocará que para poder codificar dicho valor, deberemos proceder a reescalar todos los pesos así como todos los valores de entrada a la red. Esto resulta ser bastante engorroso a la vez que difícilmente sistematizable.

Por otra parte, las neuronas lineales implementadas mediante la codificación (SESL) evalúan correctamente el valor de la suma de las contribuciones presinápticas, no requiriendo éstas de ningún tipo de reescalado. La ventaja adicional es que mediante la codificación (SESL), es posible la representación de cualquier magnitud de entrada al sistema, ya que su rango de representación es de $(-\infty,+\infty)$.

4.3.2.4 La neurona tangente-sigmoidal

Una neurona Tangente-Sigmoidal se define como aquella cuya función de transferencia es no-lineal y bivaluada en el intervalo [-1, +1], que hace uso de la función tangente hiperbólica como la función de activación. El valor de entrada a dicha función de transferencia es la suma ponderada de las entradas pre-sinápticas a dicha neurona. Con todo ello, la función de transferencia de este tipo de neuronas viene descrita mediante la ecuación [4-50]:

$$\begin{cases} \begin{cases} umbral/bias = -w_{n0} \\ x_{n0} = 1 \end{cases} \\ f(x) = \tanh(x) = b \cdot \dfrac{1+e^{+(a \cdot U_n)}}{1+e^{-(a \cdot U_n)}} \end{cases} \rightarrow salida_n = f(U_n) = f\left(\sum_{j=0}^{k} w_{nj} \cdot x_{nj}\right) = \begin{cases} b \cdot \dfrac{1+e^{+(a \cdot U_n)}}{1+e^{-(a \cdot U_n)}} \\ a = b = 1 \end{cases}$$

Ecuación [4-50]

Las neuronas Tangente-Sigmoidales son usadas generalmente para resolver problemas de clasificación o reconocimiento de patrones, generalmente las hallaremos en redes neuronales combinadas con neuronas lineales.

Este tipo de neurona se desarrolló para permitir la resolución de problemas no linealmente separables, para lo cual se requiere de neuronas dotadas de funciones de transferencia no lineales como son la Logística Sigmoidal (*logSig(x)*) (acotada entre 0 y +1) y la Tangente-Sigmoidal (*TanSig(x)*).(acotada entre -1 y +1). Este tipo de neuronas usan como regla de aprendizaje genérica la regla de retro-propagación del error "*Back-Propagation*" [4-9, 4-18].

Una vez descritas la función de transferencia y las características más significativas de este tipo de neuronas, pasaremos a describir su implementación mediante la implementación estocástica clásica con signo y la extendida con signo.

4.3.2.4.1 Implementación de la neurona tangente-sigmoidal mediante la codificación (SCSL)

El circuito digital desarrollado para implementar la neurona Tangente-Sigmoidal mediante la codificación estocástica clásica con signo (SCSL) se presenta en la Figura 4-45. Ésta se compone de un bloque lineal genérico (encargado de realizar la suma ponderada de las entradas, descrito anteriormente en el apartado 4.3.2.1.1), la salida del cual se ha conectado a un bloque estocástico que implementa la función *pseudo tangente-hiperbólica* (*Tanh(x)*) como función de activación de la neurona Tangente-Sigmoidal. El funcionamiento interno de la función de activación (bloque estocástico *pseudo Tangente Hiperbólica*) se haya descrito en detalle en el apartado 2.2.4.6 del capítulo segundo de esta tesis. La salida de este bloque es a su vez una magnitud binaria de 16-bits, la cual debe ser convertida nuevamente

en una señal estocástica, para lo cual deberemos usar un bloque B2P(N) de 16-bits. En la Figura 4-45 se puede apreciar cómo el bloque *pseudo Tanh(x)* se haya conectado a un bloque *P2B(N)*, consiguiendo así que la salida de la neurona sea nuevamente una señal pulsante.

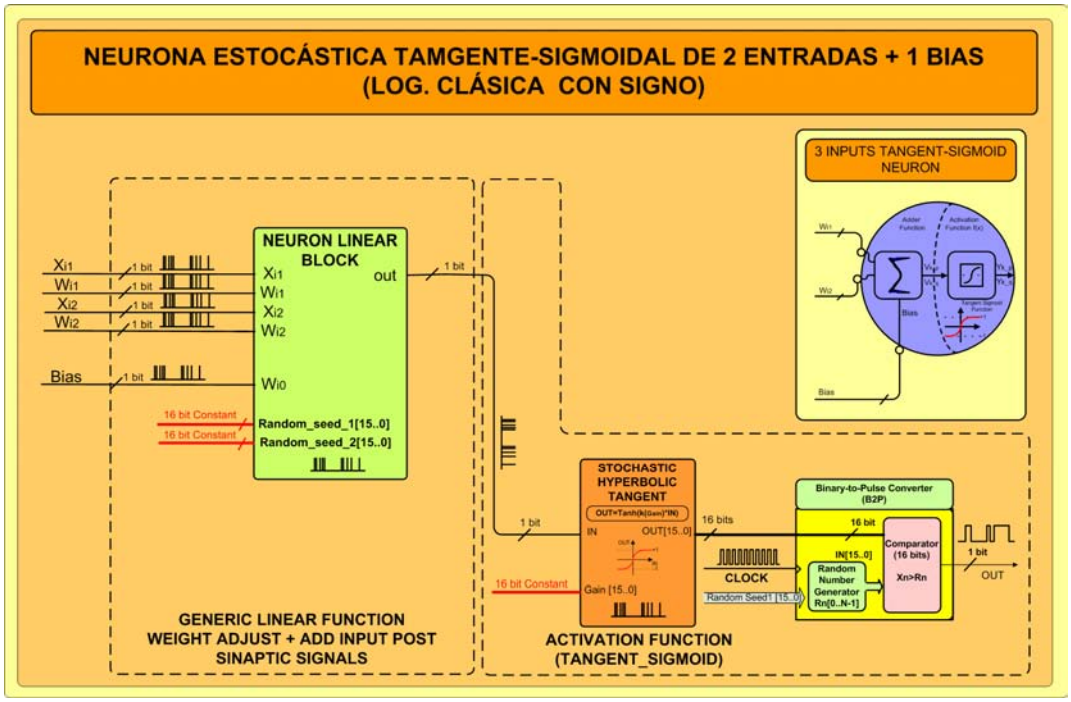

Figura 4-45: Implementación de una neurona Tangente-Sigmoidal de 2 entradas + 1 Bias (SCSL)

Si nos remitimos a la descripción de la función matemática implementada por el bloque estocástico "*pseudo tangente-hiperbólica"* en el apartado 2.2.4.6, constataremos cómo este bloque es incapaz de implementar la función tangente hiperbólica unitaria, o lo que es lo mismo, una función tangente hiperbólica con ganancia unidad "*Tanh(1,0·x)*". Esto es debido a que el valor mínimo de ganancia con el cual podemos parametrizar nuestro bloque pseudo tangente-hiperbólica es de "*k=1,3668*", siendo esta ganancia proporcional al parámetro "**Gain**" (palabra de 16-bits) con el cual se haya parametrizado el bloque. La función de activación ha de encontrarse completamante representada en el rango de representación de la señal de entrada (que en el caso extremo será de [-1, +1]). Para ello deberemos elegir correctamente el factor de ganancia de nuestra función de activación; no

obstante, para valores de “k=2” esta función ya se halla prácticamente representada tomando valores para “x=-1→out=-0,9640” y para “x=+1→out=+0,9640”. Por lo tanto, para poder compensar el efecto de este factor de ganancia para dicha función deberemos proceder a ajustar los valores de los pesos y el valor de Bias en un factor “1/k”, obteniéndose así el resultado que cabría esperar para una neurona con esos pesos iniciales. No obstante, si recordamos que el bloque lineal de una neurona estocastica en la codificación (SCSL) (descrito en el apartado 4.3.2.1.1) pondera la suma de las contribuciones presinápticas por un factor (1/n), donde “n” es el número de sumandos, se tiene que corregir el problema de la subestimación fijando la ganancia de la función de activación igual al número de sumandos “k=n”. La demostración matemática de ello se presenta en la Ecuación [4-51]:

$$\begin{cases} \begin{cases} bias = -w_{n0} \\ x_{n0} = 1 \end{cases} \\ w_{nj}, \forall j = 1,..,l \\ f(x) = \tanh(x) = \dfrac{1+e^{+(k\cdot U_n)}}{1+e^{-(k\cdot U_n)}} \\ k = n \end{cases} \rightarrow \begin{cases} salida_n = f(U_n) = f\left(\dfrac{1}{n}\sum_{j=0}^{l} w_{nj}\cdot x_{nj}\right) = \\ = \dfrac{1+e^{+\left(k\cdot\frac{U_n}{n}\right)}}{1+e^{-\left(k\cdot\frac{U_n}{n}\right)}} = Tanh\left(k\cdot\dfrac{U_n}{n}\right)_{k=n} = Tanh(U_n) \end{cases}$$

Ecuación [4-51]

Por lo tanto, mediante el bloque pseudo tangente hiperbólica ajustable hemos conseguido implementar una neurona tangente sigmoidal que no requiere a priori de ningún reescalado de los pesos de entrada, ni de los valores de Bias para su correcta operación.

Una vez descrito el mecanismo de operación de la neurona Tangente-Sigmoidal, procederemos a comprobar su correcto funcionamiento mediante la implementación de una neurona de dos entradas con una entrada adicional de Bias en una FPGA Cyclone II de la empresa Altera Corp. La configuración de la neurona implementada ha consistido en fijar el valor de las señales de entrada “$x_{i2} = +0{,}9$”, *Bias* “$w_{i0} = 0$”, a la vez que se han fijado todos los pesos de entrada de la red a “$w_{i1} = +0{,}9$” y “$w_{i2} = +0{,}2$”. En la Tabla 4-13 se presentan de forma resumida los parámetros de configuración de la neurona.

Tabla 4-13: Parámetros de la neurona Tangente-Sigmoidal (SCSL)			
Parámetros de la Neurona Tangente-Sigmoidal			
W_{i0} (Bias)	W_{i1}	W_{i2}	X_{i2}
0	+0,8999	+0,8999	0,2000
Parámetros de la función de activación Tangente-Sigmoidal			
n (número de sumando)	Gain (parámetro de configuración)	k (ganancia de la función de activación)	
3	21	3,000	

Una vez configurada la neurona y la función de activación con los parámetros descritos en la Tabla 4-13, hemos procedido a variar el valor medio de la distribución de entrada "x_{i1}" en el intervalo [-1, +0,875]. Los resultados obtenidos son los que se presentan en la Tabla 4-14.

Tabla 4-14: Medidas experimentales de la neurona Tangente-Sigmoidal (SCSL)					
Xi1 [16bits]	Xi1 Probabilidad equivalente	OUT [16bits] (Experimental)	OUT Probabilidad equivalente (Experimental)	OUT (calculada con los parámetros de la neurona sin reescalar) (Teórica)	Error ABS(Teo-Exp)
0	-1,0000	13106	-0,6000	-0,6169	0,0169
4096	-0,8750	15388	-0,5304	-0,5424	0,0120
8192	-0,7500	18466	-0,4365	-0,4582	0,0217
12288	-0,6250	21198	-0,3531	-0,3649	0,0118
14336	-0,5625	23436	-0,2848	-0,2636	0,0211
16384	-0,5000	28582	-0,1277	-0,1562	0,0285
18432	-0,4375	29061	-0,1131	-0,0450	0,0681
20480	-0,3750	35157	0,0729	0,0674	0,0055
24576	-0,2500	39112	0,1936	0,1781	0,0155
28672	-0,1250	41849	0,2771	0,2844	0,0073
32768	0,0000	47036	0,4354	0,3842	0,0512
36864	0,1250	49692	0,5165	0,4758	0,0407
40960	0,2500	52756	0,6100	0,5580	0,0519
45056	0,3750	54682	0,6688	0,6306	0,0381
49152	0,5000	57039	0,7407	0,6937	0,0470
53248	0,6250	57668	0,7599	0,7476	0,0123
57344	0,7500	13106	-0,6000	-0,6169	0,0169
61440	0,8750	15388	-0,5304	-0,5424	0,0120

Si a continuación procedemos a comparar los resultados experimentales obtenidos mediante la implementación estocástica realizada y los predichos por la teoría, podemos concluir que la implementación realizada obtiene el resultado esperado. Además, en esta implementación hemos conseguido corregir el problema del factor de subestimación descrito anteriormente. Finalmente procedemos a representar gráficamente (Figura 4-46) la salida de la neurona tangente-sigmoidal en función de la probabilidad de la señal "Xi1" de

entrada (Tabla 4-14) a fin de visualizar la función de transferencia obtenida experimentalmente.

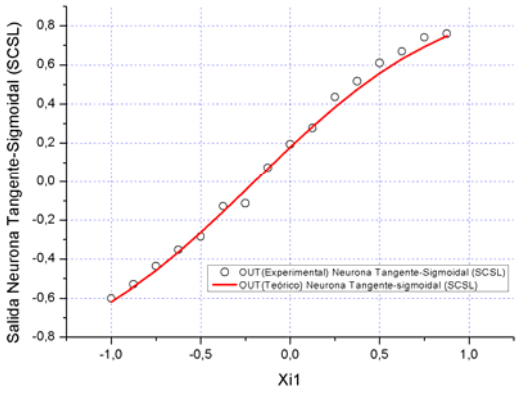

Figura 4-46: Resultados experimentales de la neurona tangente-sigmoidal (SCSL)

4.3.2.4.2 Implementación de la neurona tangente-sigmoidal mediante la codificación (SESL)

El circuito digital desarrollado para implementar la neurona Tangente-Sigmoidal mediante la codificación estocástica extendida con signo (SESL) se presenta en la Figura 4-47.

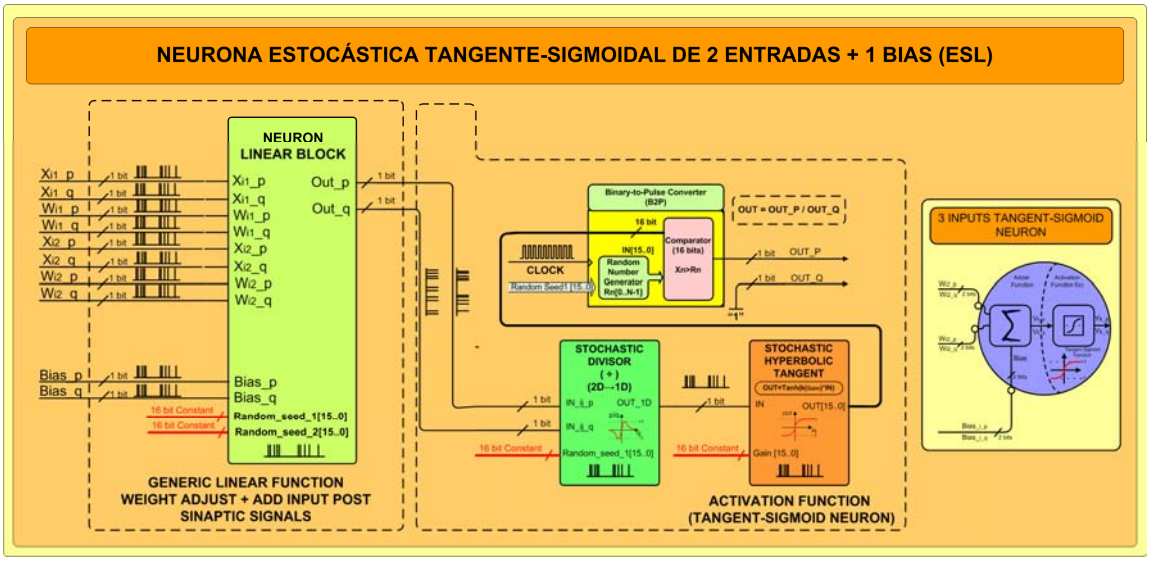

Figura 4-47: Implementación de una neurona tangente-sigmoidal de 2 entradas + 1 Bias (SESL)

Dicha implementación es muy semejante a la realizada para la codificación estocástica clásica con signo (SCSL). No obstante, en este caso hemos requerido de un bloque divisor (descrito en el subapartado 2.2.4.5) para evaluar la razón entre el numerador y el denominador de la salida del bloque lineal de la neurona. De esta forma se obtiene una sola señal estocástica de salida equivalente a la razón del numerador y denominador en formato Bipolar (codificación clásica con signo (SCSL)) acotada en el intervalo [-1, +1]. Este proceso de conversión es necesario ya que no disponemos de un bloque en la coficiación (SESL) capaz de implementar la función de activación "*Tanh(k·x)*", por lo tanto, nos hemos visto obligados a usar el bloque "*Tanh(k·x)*" diponible para la codificación (SCSL). Lo que implica que si la razón representada es mayor (menor) a +1 (-1) la salida del divisor saturará a +1 (-1). Este hecho a priori representaría un escollo insalvable para esta implementación, pero en el caso de la función *Pseudo Tangente Hiperbólica* (al cual se halla conectado) ésta tiene restringida su salida en el mismo intervalo. A todo ello, debe tenerse en cuenta que ésta debe hallarse completamente definida en este intervalo, para lo cual hemos configurado el bloque *Pseudo Tangente Hiperbólica* con un parámetro "**Gain**=41" que se corresponde con una ganancia "**k**=4,0687" de dicha función "*Tanh(k·x)*". En la Figura 4-48 se puede apreciar cómo la función de transferencia de la función de activación en el rango [-1, +1] se haya completamante definida.

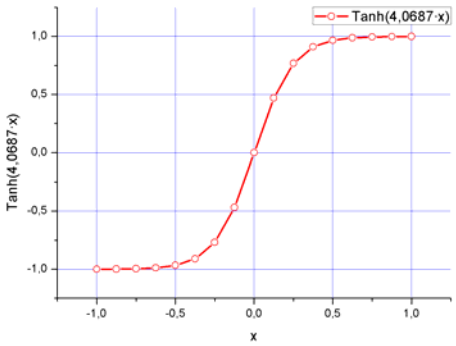

Figura 4-48: Representación gráfica de la función Tanh(4,0687·x) en el rango x∈[-1,+1]

Al igual que en la implementación anterior, la salida de este bloque *Pseudo Tangente Hiperbólica* es un dato de 16-bits, el cual debe ser convertido nuevamente a una señal estocástica mediante un bloque B2P(N) de 16-bits. En la Figura 4-46 se puede apreciar

cómo el bloque *pseudo Tanh(x)* se haya conectado con un bloque *P2B(N)*, siendo así la salida de la neurona nuevamente una señal pulsante. Cabe remarcar que la presencia del bloque divisor obedece al hecho que aun no hemos conseguido implementar un bloque en la codificación (SESL) que sea capaz de realizar de forma directa esta función matemática.

En este caso, el bloque lineal de la neurona en la codificación (SESL) evalúa correctamente la suma de las contribuciones pre-sinápticas. Para compensar el efecto del factor de ganancia "k" de la función de activación "*Tanh(x)*" deberemos proceder a ajustar los valores de los pesos y el valor de Bias en un factor "1/k". No obstante, no será necesario reescalar las entradas del sistema sino solo los pesos. De esta forma, se obtiene el resultado que uno cabría esperar para dicha neurona con los pesos iniciales. La demostración matemática de ello se presenta en la Ecuación [4-52]:

$$\begin{cases} \begin{cases} bias = -w_{n0}/k \\ x_{n0} = 1 \end{cases} \\ w_{nj}/k\,,\forall j = 1,..,l \\ f(x) = \tanh(x) = \dfrac{1+e^{+(k\cdot U_n)}}{1+e^{-(k\cdot U_n)}} \\ k = 4{,}0687 \end{cases} \rightarrow \begin{cases} salida_n = f(U_n) = f\left(\dfrac{1}{n}\sum_{j=0}^{l} w_{nj}\cdot x_{nj}\right) = \\ = \dfrac{1+e^{+\left(k\cdot\frac{U_n}{k}\right)}}{1+e^{-\left(k\cdot\frac{U_n}{k}\right)}} = Tanh\left(k\cdot\dfrac{U_n}{k}\right) = Tanh(U_n) \end{cases}$$

Ecuación [4-52]

Mediante la codificación (SESL) podemos implementar neuronas lineales que evalúan correctamante su salida (sin el problema de la subestimación de la suma), a la vez que podremos implementar neuronas tangente sigmoidales, tan sólo reescalando el valor de los pesos y de Bias de dichas neuronas. Esto implica que ya no tendremos ningún problema para sistematizar la implemantación de grandes redes neuronales estocásticas.

Una vez descritos los mecanismos de operación de la neurona Tangente-Sigmoidal en la codificación (SESL), hemos procedido a comprobar su correcta operación, implemantando una neurona de dos entradas con una entrada adicional de Bias en una FPGA Cyclone II (EP2C20F484C7N) de la empresa Altera Corp. Para su evaluación se ha fijado el valor de la señal de entrada "$x_{i1\_p}/x_{i1\_q} = +1$", *Bias* "$w_{i0\_p}/w_{i0\_q} = -0{,}081067$", todos los pesos de

entrada de la red a "$w_{i1\_p}/w_{i1\_q} = -0.58052$" y "$w_{i2\_p}/w_{i2\_q} = -1.1766$". En este caso deberemos proceder a reescalar los pesos y el Bias un factor "*1/k=1/4,0687=0,2458*". Los parámetros de configuración de la neurona son los que se presentan en la Tabla 4-15.

Tabla 4-15: Parámetros de la neurona Tangente-Sigmoidal (SESL)

Parámetros originales de la Neurona Tangente-Sigmoidal			
w_{i0_p}/w_{i0_q} (Bias)	w_{i1_p}/w_{i1_q}	w_{i2_p}/w_{i2_q}	x_{i2_p}/x_{i2_q}
-0,0811	-0,5805	-1,1766	+1
Parámetros reajustados para la neurona Tangente-Sigmoidal (Que obtendrán la salida esperada para dicha neurona)			
w_{i0_p}/w_{i0_q} (Bias)	w_{i1_p}/w_{i1_q}	w_{i2_p}/w_{i2_q}	x_{i2_p}/x_{i2_q}
-0,0199	-0,1427	-0,2892	+1

Una vez configurada la neurona con los parámetros descritos en la Tabla 4-15, hemos procedido a variar el valor medio de la distribución de entrada x_{i2} en el intervalo [-1.4, 1.6] para evaluar la correcta operación de la neurona desarrollada, siendo los resultados obtenidos para tal caso los que se presentan en la Tabla 4-16.

Tabla 4-16: Medidas experimentales de la neurona Tangente-Sigmoidal de 3 entradas (SESL)

x_{i2_p} [16bits]	x_{i2_q} [16bits]	x_{i2_p}/x_{i2_q} Probabilidad equivalente	OUT_{i_p} [16bits] (experimental)	OUT_{i_q} [16bits] (Experimental)	OUT (Experimental) (TanSig(x))	OUT Teórica (TanSig(x))	Error ABS (Teo-Exp)
0	12288	1,6000	0	65535	-1,0000	-0,9877	0,0123
4096	12288	1,4000	168	65535	-0,9949	-0,9804	0,0145
8192	12288	1,2000	252	65535	-0,9923	-0,9689	0,0235
12288	12288	1,0000	1134	65535	-0,9654	-0,9506	0,0148
16384	12288	0,8000	3022	65535	-0,9078	-0,9221	0,0143
20480	12288	0,6000	5712	65535	-0,8257	-0,8781	0,0524
24576	12288	0,4000	6846	65535	-0,7911	-0,8118	0,0207
28672	12288	0,2000	11298	65535	-0,6552	-0,7148	0,0596
32768	12288	0,0000	12899	65535	-0,6064	-0,5794	0,0270
36864	12288	-0,2000	21294	65535	-0,3502	-0,4022	0,0520
40960	12288	-0,4000	27567	65535	-0,1587	-0,1887	0,0299
45056	12288	-0,6000	33279	65535	0,0156	0,0443	0,0287
49152	12288	-0,8000	42987	65535	0,3119	0,2726	0,0392
53248	12288	-1,0000	48273	65535	0,4732	0,4738	0,0007
57344	12288	-1,2000	53775	65535	0,6411	0,6353	0,0058
61440	12288	-1,4000	56085	65535	0,7116	0,7555	0,0439

Si ahora comparamos los resultados experimentales con los predichos por la teoría, podremos apreciar como son prácticamente idénticos, siendo el mínimo error obtenido entre ambas implementaciones debido a la dispersión inherente a la codificación

estocástica. A su vez, mediante la codificación (SESL) ha desaparecido el problema de la subestimación inherente de la suma que existe en la implementación Bipolar (codificación clásica con signo), deberemos corregir previamante la ganancia de la función de activación (reescalando los pesos y valores de Bias). Para finalizar con este subapartado hemos procedido a representar gráficamente (Figura 4-49) la salida de la neurona Tangente-Sigmoidal en función de la probabilidad de la señal "X_{i2}" de entrada (Tabla 4-16).

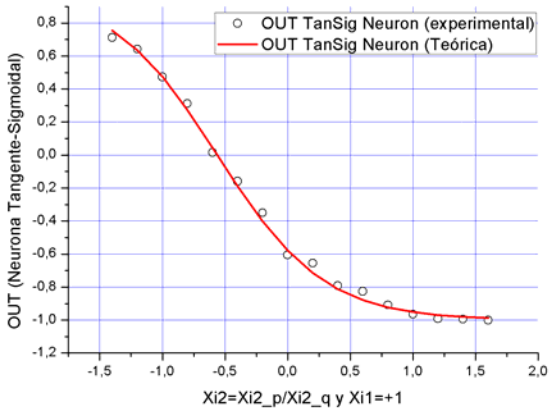

Figura 4-49: Resultados experimentales de la neurona Tangente-Sigmoidal (SESL)

4.3.3 IMPLEMENTACIÓN ESTOCÁSTICA DE UNA RED NEURONAL FEED-FORWARD

En este apartado presentamos una aplicación simple para demostrar la viabilidad del uso de la codificación estocástica extendida con signo (SESL) para la implementación de redes neuronales complejas [4-124]. Para ello, hemos desarrollado un clasificador simple mediante una red neuronal estocástica, para discriminar entre dos categorías diferentes de datos [A, B]; basado sobre una red neuronal Feed-Forward totalmente interconectada de tres capas (Figura 4-50), la capa de entrada se compone de dos señales estocásticas que representan las coordenadas "X" e "Y" del plano donde se ubican los elementos de las diferentes categorías [A, B], mientras que la capa oculta se compone de cinco neuronas lineales. La capa de salida se constituye por una neurona Tangente-Sigmoidal (no-lineal) encargada de realizar la tarea de clasificación entre ambas categorías.

Las entradas de la red (X, Y) se corresponden con las coordenadas espaciales de entrada a reconocer mientras que la neurona de salida nos indicará con un “+1” la pertenencia a la clase **A** y con una salida a “-1” la pertenencia a la clase **B**.

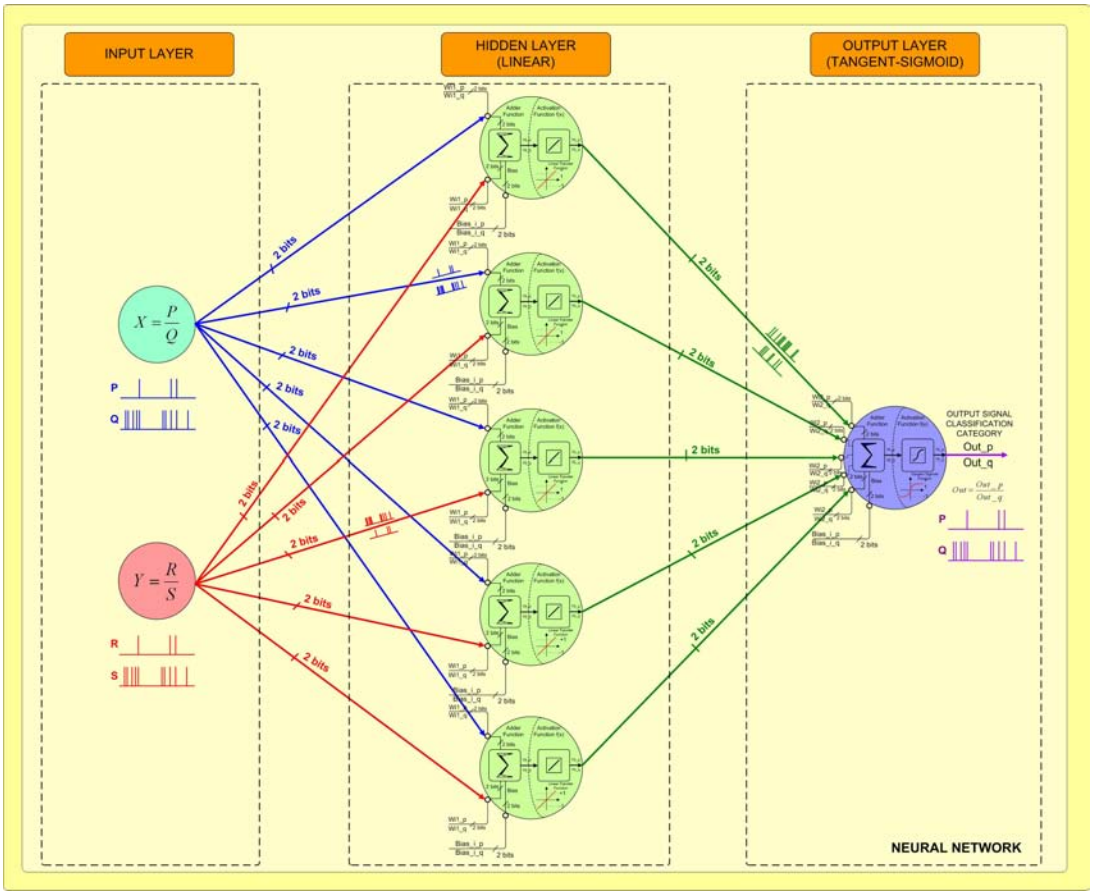

Figura 4-50: Diagrama de la red neuronal estocástica *Feed-Forward* 2-5-1

Inicialmente hemos implementado un programa en ANSI C para generar dos poblaciones (o clases) Bayesianas (Figura 4-51) de datos en un espacio bidimensional mediante una técnica basada en los algoritmos de Montecarlo (dicha aplicación se adjunta en el Anexo B de la presente tesis). Mediante esta aplicación también se han generado los ficheros de datos que requiere MATLAB para su ToolBox de entrenamiento de redes neuronales.

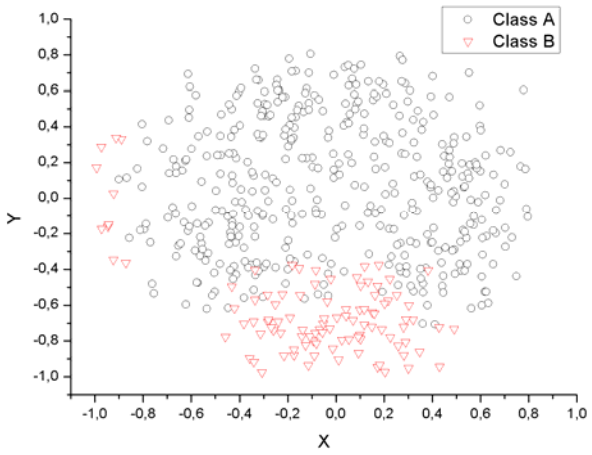

Figura 4-51: Clases [A, B] muestra de entrenamiento de la red neuronal

Una vez generados los ficheros de clase y de salida de la red neuronal mediante la anterior aplicación ANSI C, se ha procedido a importar dichos ficheros como matrices de datos en MATLAB. Una vez en MATLAB, hemos ejecutado el ToolBox de redes neuronales "**nntool**" para proceder al entrenamiento de la red neuronal propuesta, usando la técnica "*Feed-Forward Back-Propagation*". El proceso seguido es el que se muestra en la Figura 4-52.

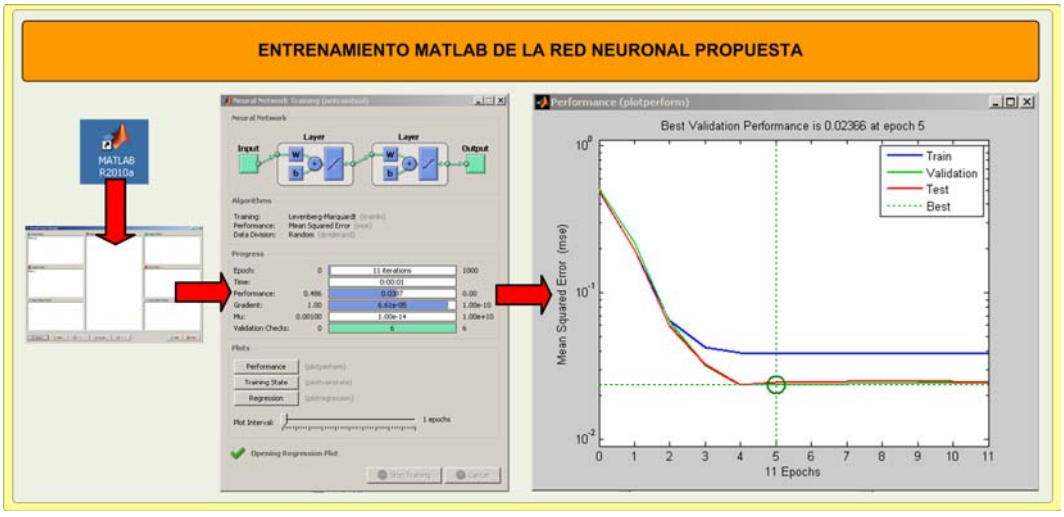

Figura 4-52: Resultados MATLAB del entrenamiento de la red neuronal propuesta

De esta forma se han obtenido los diferentes pesos y valores de Bias para cada una de las neuronas que componen la red neuronal 2-5-1. Dichos valores pueden implementarse directamente en hardware (FPGA), solo reajustando los valores de los pesos y del Bias en el caso de la neurona tangente-sigmoidal presente en la red propuesta. Para el reajuste de los pesos y la obtención de la razón de los valores equivalentes a los diferentes pesos y valores de Bias (necesario para su implementación mediante la codificación SESL), se ha desarrollado una aplicación propia en ANSI C (esta apliación se adjunta en el anexo B de la presente tesis). Los parámetros de configuración de la red obtenidos una vez reajustados y convertidos a valores de 16-bits se presentan en la Tabla 4-17.

Tabla 4-17: Parámetros de configuración de la red 2-5-1

Neurona lineal 1, capa oculta						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
P/Q	-0.0811	-0.5805	-1.1766	--	--	--
P	31459	13745	65535	--	--	--
Q	65535	65535	4918	--	--	--
Neurona lineal 2, capa oculta						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
P/Q	-0.3163	0.3422	1.2502	--	--	--
P	27663	43979	65535	--	--	--
Q	65535	65535	58978	--	--	--
Neurona lineal 3, capa oculta						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
P/Q	0.4107	0.4213	0.5180	--	--	--
P	39397	46572	49743	--	--	--
Q	65535	65535	65535	--	--	--
Neurona lineal 4, capa oculta						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
P/Q	0.9073	-0.5690	1.2361	--	--	--
P	47414	23583	52720	--	--	--
Q	65535	65535	65535	--	--	--
Neurona lineal 5, capa oculta						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
P/Q	0.9917	-0.6934	-0.1094	--	--	--
P	48776	21574	31002	--	--	--
Q	65535	65535	65535	--	--	--
Neurona Tangente-Sigmoidal, capa de salida						
	w_{i0} (Bias)	w_{i1}	w_{i2}	w_{i3}	w_{i4}	w_{i5}
	-0.6107	1.2996	-0.8807	-0.8298	-1.5308	-0.3880
Valores reescalados	-0.1255	0.3194	-0.2165	-0.2040	-0.3763	-0.0954
P	27764	43414	25553	25970	20227	29589
Q	65535	65535	65535	65535	65535	65535

La codificación hardware de esta red neuronal se ha realizado mediante la herramienta de sintetización y programación de dispositivos de lógica programable "**QUARTUS II"** (Altera Corp.) combinando bloques de código VHDL y esquemáticos. Toda la red se ha embebido sobre una FPGA de bajo coste Cyclone II modelo "*EP2C20F484C7N*" de la empresa norteamericana Altera Corp.

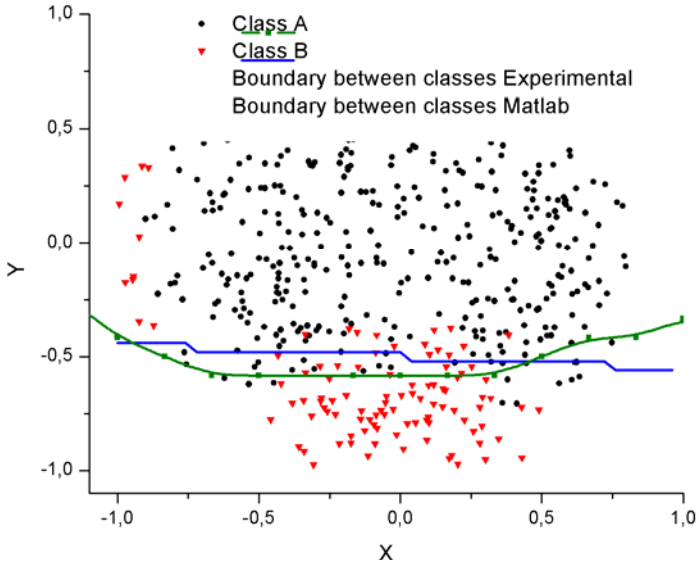

Figura 4-53: Resultados experimentales de clasificación obtenidos mediante la red neuronal propuesta

En la Figura 4-53 se presentan los resultados experimentales obtenidos evaluando todo el espacio de configuraciones de las entradas en los intervalos {X=-1, Y=-1} y {X=+1, Y=+1}. A fin de presentar una mayor legibilidad de los resultados de cara al lector, hemos evaluado a partir de los datos experimentales una línea de frontera entre clases que hemos representado mediante una línea continua de color verde. A su vez (mediante una línea azul continua), se representa la frontera de decisión teórica de la red. Dicha frontera teórica la hemos obtenido mediante una aplicación en ANSI C que simula la respuesta de la red para los parámetros obtenidos de entrenamiento (dicha aplicación se adjunta en el Anexo B de la presente tesis). Finalmente, las distribuciones de entrenamiento de la red se presentan mediante los símbolos negros (clase A) y los símbolos rojos (clase B).

Si nos fijamos en la Figura 4-53, podremos apreciar cómo la frontera de decisión experimental (línea verde) es muy parecida a la teórica (línea azul). El pequeño desajuste entre las dos soluciones se debe a nuestro parecer a dos fenómenos: el primero relacionado con la dispersión inherente de la codificación estocástica, y el segundo relacionado con el error de codificación que se hace extremadamente grande para valores cercanos a cero en la codificación estocástica (SESL). No obstante, estos resultados muestran que la metodología propuesta es apta para la implementación de redes neuronales estocásticas mediante el uso directo de los valores de entrenamiento obtenidos mediante aplicaciones matemáticas de cálculo simbólico estándares, que hacen uso de algoritmos de entrenamiento. De esta forma no se requiere de un reescalado global de los valores de entrada de la red al rango [-1, +1] como sí sucede en el caso de la codificación estocástica clásica con signo (SCSL).

5. CONCLUSIONES Y TRABAJO FUTURO

Es objeto del presente capítulo la exposición de las principales conclusiones y aportaciones del trabajo efectuado, así como la propuesta de las posibles líneas futuras de trabajo que se derivan de la presente tesis.

5.1 CONCLUSIONES

Del presente trabajo de investigación se desprenden todo un conjunto de conclusiones, las cuales procederemos a presentar ordenadas en función del objetivo inicial en el cual se hallan encuadradas:

El primer objetivo del trabajo de investigación se engloba en el "*desarrollo y evaluación de los fundamentos básicos de la computación basada en metodologías pulsantes/estocásticas*", las metas o conclusiones alcanzadas son las siguientes:

- Desarrollo de un circuito CMOS extremadamente simple [5-1, 5-5] orientado a la generación de números aleatorios para la generación de señales aleatorias.
- Desarrollo y evaluación de una técnica de optimización para la generación de números pseudo aleatorios en FPGA's.
- Desarrollo y evaluación de todo un conjunto propio de bloques digitales en el marco de la codificación estocástica clásica unipolar (UCSL) [5-4, 5-6, 5-9]. Es de destacar que se han implementado todas las funciones aritméticas básicas, así como un conjunto de bloques capaces de implementar algunas de la funciones matemáticas más habituales (por Ej. la función exponencial). De entre todos ellos, cabe destacar la implementación de un bloque digital que realiza la función Gaussiana, siendo ésta totalmente configurable.

- Desarrollo y evaluación de todo un conjunto propio de bloques digitales en el marco de la codificación estocástica clásica bipolar (SCSL). Para la cual se han implementado todas las funciones aritméticas básicas, así como un bloque que aproxima la función tangente hiperbólica.
- Finalmente se ha procedido al modelado matemático, desarrollo y evaluación de dos nuevas codificaciones estocásticas [5-10]. Éstas se basan en el uso de la razón entre dos señales estocásticas para la codificación de una magnitud binaria, permitiendo la extensión del rango de representación natural de la lógica estocástica a [0, +∞) en su versión sin signo (UECL), y a (-∞, +∞) en su versión con signo (SESL). La versión sin signo UESL ha sido desarrollada específicamente para su aplicación en el campo de reconocimiento de patrones estadístico, mientras que la versión con signo SESL se ha desarrollado para la implementación de redes neuronales estocásticas. Para ambas codificaciones se han desarrollado los bloques digitales capaces de implementar cualquier operación aritmética básica.

El segundo objetivo establecido en esta tesis se engloba en el "*desarrollo de los elementos y metodologías básicas para la aplicación de la computación estocástica al campo del reconocimiento de patrones*", los objetivos alcanzados son los siguientes:

- Se ha desarrollado y evaluado un bloque digital capaz de implementar una f.d.p pseudo-normal [5-4, 5-6].
- Se ha adaptado y evaluado el bloque estocástico Gaussiana (UCSL) a fin que éste implemente una f.d.p normal.
- Se han desarrollado y evaluado una pareja de bloques estocásticos implementados uno mediante la codificación (UCSL) y el otro en la codificación (UESL), capaces de evaluar la probabilidad a posteriori (mediante la regla de Bayes) [5-4] de una distribución de probabilidad dada.
- El desarrollo y evaluación básica de un mecanismo de clasificación estocástica computacionalmente eficiente basado en el paradigma computacional "*Winner-*

Take-All". Este mecanismo se ha implementado en forma de un bloque digital llamado "*Multidimensional Stochastic Classifier*" (MSC) [5-4, 5-6], el cual se ha implementado para las codificaciones (UCSL) y (UESL).

- Se ha evaluado la capacidad computacional de un bloque "*Multidimensional Stochastic Classifier*" (MSC) en relación a una implantación del mismo clasificador puramente software [5-4], resultando ser la implementación estocástica basada en un bloque (MSC) 3000 veces más rápida que la aplicación software equivalente.
- Se ha evaluado la capacidad de clasificación multidimensional del bloque estocástico MSC [5-6] mediante un sistema ejemplo en el cual a partir del reconocimiento de un entorno estructurado virtual se desplaza un objeto.
- Se ha desarrollado y evaluado un bloque digital capaz de implementar la metodología de las ventanas de *Parzen* estocásticamante, pudiendo así estimar las f.d.p condicionales a partir de los datos de una muestra estadística. De esta manera se pueden afrontar problemas de reconocimiento de patrones estadístico en el supuesto no-paramétrico.

En el marco del tercer objetivo del trabajo de investigación, que engloba el "*desarrollo de nuevas implementaciones de redes neuronales basadas en la computación estocástica, para su aplicación al campo del reconocimiento de patrones*", las metas alcanzadas son las siguientes:

- Se ha desarrollado y evaluado una neurona Perceptrón mediante bloques estocásticos en la codificación estocástica (SESL).
- Se ha desarrollado y evaluado una neurona lineal [5-10] mediante bloques estocásticos en la codificación estocástica (SESL).
- Se ha desarrollado y evaluado una neurona Tangente-Sigmoidal [5-10] mediante bloques estocásticos en la codificación estocástica (SESL).
- Se ha implementado y evaluado una red neuronal estocástica en la codificación (SESL) para la resolución de un problema de reconocimiento de patrones [5-10].

Finalmente el cuarto y último objetivo de la tesis engloba el "*desarrollo de nuevas estructuras digitales para la implementación de redes neuronales pulsantes, así como el desarrollo de nuevas metodologías de configuración de estas redes orientadas al reconocimiento de patrones*". Los objetivos alcanzados son los siguientes:

- Se ha desarrollado e implementado una neurona pulsante propia enmarcada en la familia de neuronas basadas en modelos "Integrate & Fire" [5-2, 5-7].
- Se ha implementado y evaluado un mecanismo de configuración de redes neuronales pulsantes digitales, orientado al reconocimiento de patrones temporales [5-2, 5-8].
- Se ha implementado y evaluado la operatividad de una neurona pulsante mixta (analógico/digital) como elemento básico de computación [5-3], basada en un núcleo analógico que emula el funcionamiento de una neurona biológica, mientras que la gestión de las interconexiones se ha implementado digitalmente en una FPGA.

Como conclusión final, podemos afirmar que se han desarrollado y evaluado los elementos básicos para poder afrontar con garantías la resolución de problemas de computación masiva, de forma fiable y computacionalmente eficiente mediante el uso de la computación estocástica.

5.2 DISEMINACIÓN DE LOS RESULTADOS DE LA TESIS

En el presente subapartado se detallan las diversas contribuciones al estado del arte derivadas del trabajo presentado en esta tesis, las cuales hemos organizado en dos grupos: las aportaciones en revistas internacionales indexadas y las aportaciones a conferencias nacionales/internacionales. A su vez, hemos ordenado cronológicamente las diferentes aportaciones realizadas.

- Las contribuciones en revistas internacionales indexadas directamente relacionadas con el trabajo presentado, son las siguientes:

[5-1] J.L. Rosselló, **V. Canals**, V., I. de Paúl, S. Bota, A. Morro, "A Simple CMOS Chaotic Integrated Circuit", IEICE Electronics Express, Volume 5, pp. 1042–1048, 2008

[5-2] J.L. Rosselló, I. De Paúl, **V. Canals**, "Self-configuring Spiking Neural Networks", IEICE Electronics Express, Volume 5, Issue 22, pp. 921-926, 2008

[5-3] J.L. Rosselló, **V. Canals**, A. Morro, J. Verd, "Chaos-based mixed signal implementation of spiking neurons", International Journal of Neural Systems, Volume 19, Issue 6, pp. 465-471, 2009

[5-4] **V. Canals**, A. Morro, J.L. Rosselló, "Stochastic-based pattern-recognition analysis", Pattern Recognition Letters, Volume 31, Issue 15, pp. 2353-2356, 2010

Adicionalmente a estas cuatro publicaciones en revistas internacionales indexadas, nos hallamos a la espera de una respuesta oficial al respecto de la posible publicación de dos trabajos adicionales relacionados directamente con esta tesis. El primero de ellos se corresponde con la metodología de optimización de la generación de números aleatorios presentada en el apartado 2.2.2.2, siendo el título del trabajo y la publicación a la cual se ha remitido los siguientes:

V. Canals, A. Morro, J.L. Rosselló, "Chaos-Based Stochastic Computation", Journal of Circuits, Systems and Computers (Enviado para su posible publicación en febrero de 2011)

El segundo versa sobre la metodología de implementación de redes neuronales estocásticas basadas en la codificación (SESL) presentada en el apartado 4.3 de la presente tesis. Siendo el título del trabajo y la publicación a la cual se ha remitido los siguientes:

V. Canals, A. Morro, J.L. Rosselló, “Extended stochastic logic and its application to neural network implementation”, International Journal of Neural Systems (Enviado para su posible publicación en julio de 2011)

- Las contribuciones en conferencias internacionales relacionadas con el trabajo presentado, son las siguientes:

[5-5] J.L. Rosselló, S. Bota, **V. Canals**, I. De Paul, J. Segura, “A fully CMOS low-cost chaotic neural network”, Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN 06), Article number 1716157, pp. 659-663, Vancouver (Canada), July 2006

[5-6] J.L. Rossello, **V. Canals**, I. de Paul, J. Segura, “Using Stochastic Logic for Efficient Pattern Recognition Analysis”, IEEE International Joint Conference on Neural Networks (IJCNN 2008) (IEEE World Congress on Computational Intelligence), pp. 1057-1061, Hong Kong (China), June 2008

[5-7] J.L. Rosselló, **V. Canals**, A. Morro, I. De Paúl, “Practical hardware implementation of self-configuring neural networks”, Proceedings of 6th International Symposium on Neural Networks, ISNN 2009; paper code 77071, Wuhan (China); May 2009, this paper is accessible at “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)”, Volume 5553 LNCS, Issue PART 3, pp. 1154-1159, 2009

[5-8] J.L. Rosselló, I. De Paúl, **V. Canals**, A. Morro, “Spiking neural network self-configuration for temporal pattern recognition analysis”, 19th International Conference on Artificial Neural Networks, paper code 77563, ICANN 2009; Limassol (Cyprus); September 2009, this paper is accessible at “Lecture Notes in Computer Science (including

subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)", Volume 5768 LNCS, Issue PART 1, pp. 421-428, 2009

[5-9] J.L. Rosselló, **V. Canals**, A. Morro, "Hardware implementation of stochastic-based Neural Networks", Proceedings of International Joint Conference on Neural Networks 2010 (IJCNN 2010), Article number 5596805, Barcelona (Spain); July 2010

[5-10] J.L. Rosselló, **V. Canals**, A. Morro, "Probabilistic-based Neural Network Implementation", IEEE International Joint Conference on Neural Networks (IJCNN 2012) (IEEE World Congress on Computational Intelligence), Brisbane (Australia), 10-15 June 2012

5.3 FUTURAS LÍNEAS DE TRABAJO

A lo largo del desarrollo de este trabajo de investigación, se han ido identificando nuevas líneas donde poder aplicar los conocimientos adquiridos y los resultados obtenidos, así como algunas propuestas encaminadas a mejorar y profundizar en el trabajo desarrollado. Por ello, a continuación presentamos un resumen de las posibles futuras líneas de trabajo.

- La primera y principal se enmarca en el plan nacional "*Desarrollo e implementación de sistemas de computación de muy alta velocidad mediante redes pulsantes y su aplicación a la búsqueda de nuevos fármacos*" (*TEC2011-23113*) del cual el autor es investigador y en donde se pretenden aplicar las diferentes metodologías de reconocimiento de patrones estocástico desarrolladas en la tesis a la resolución de problemas reales de computación masiva dentro del campo de la búsqueda de nuevos fármacos.
- Se pretende abordar la implementación de la función exponencial en la codificación estocástica clásica en su formato bipolar (SCSL) a fin de poder afrontar la implantación de redes neuronales de base radial (RBF). El objetivo es su aplicación en el campo del reconocimiento de patrones.

- Se pretende abordar el desarrollo y la evaluación de los beneficios computacionales de las implementación de redes neuronales recurrentes monocapa para su aplicación en el campo del procesamiento de señales, más concretamente en el campo del procesado de imágenes.
- Abordar la viabilidad de la aplicación de la computación estocástica en el campo del control predictivo adaptativo.
- Por último y en el marco de la relación Universidad-Empresa se pretende abordar la viabilidad del desarrollo de un sistema computacionalmente eficiente capaz de proporcionar en tiempo real una predicción del consumo térmico y eléctrico en el marco de una red de distrito aislada.

BIBLIOGRAFÍA

A continuación se presenta la relación entre los diferentes códigos bibliográficos y sus correspondientes referencias bibliográficas, las cuales hemos ordenado por capítulos para facilitar al lector su identificación.

REFERENCIAS CAPÍTULO 1

[1-1] G.E. Moore, "Cramming more components on to integrated circuits", Electronics Magazine, Volume 38, Issue 8, pp. 114-117, 1965

[1-2] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarazi, V. De, "Parameter variations and impact on circuits and microarchitecture," in Proceedings of DAC., pp. 338–342, 2003

[1-3] "International Technology Roadmap for Semiconductors 2011 update", ITRS, documento de dominio público, http://www.itrs.net/Links/2011ITRS/Home2011.htm

[1-4] J. Von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components", Automata Studies, pp. 43–98, 1956.

[1-5] J. Rabaey, D. Burke, K. Lutz, J. Wawrzynek, "Workloads of the future," IEEE Design and Test of Computers, Volume 25, Issue 4, pp. 358–365, 2008.

[1-6] H. Chen, J. Han, "Stochastic computational models for accurate reliability evaluation of logic circuits", Proceedings of the 20th IEEE/ACM Great Lakes Symposium on VLSI (GLSVLSI'10), Providence (Rhode Island, USA), pp. 61–66, 2010

[1-7] F. Soares, J. Burken, T. Marwala, “Neural network applications in advanced aircraft flight control system, a hybrid system, a flight test demonstration”, Proceedings of the 13th international conference on Neural information processing (ICONIP ’06), pp. 684-691, 2006

[1-8] A.K. Jain, R.P.W. Duin, Mao Jianchang, “Statistical pattern recognition: a review”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22, Issue 1, pp. 4-37, 2000

[1-9] B.R. Gaines, “Stochastic computing systems”, Advances in Information Systems Science, Volume. 2, pp.37-172, 1969

REFERENCIAS CAPÍTULO 2

[2-1] B.R. Gaines, “R68-18 Random pulse machines”, IEEE Transactions on Computers, pp. 410-410, Apr. 1968

[2-2] Bradley D. Brown and Howard C. Card, “Stochastic Neural Computation I: Computational Elements”, IEEE Transactions on Computers, Vol. 50, No. 9, pp.891-905, Sep 2001

[2-3] Sigbjorn Naess, Tor Sverre Lande, and Yngvar Berg, “Performance Enhancement in Stochastic Pulse Code Systems Using Parallelism and Redundancy”, Proceedings of European Solid State Circuits Conference (ESSCIRC 97), pp. 400-403, 1997

[2-4] Vincent Canals, Antoni Morro, Josep L. Rosselló, “Stochastic-based pattern-recognition analysis”, Pattern Recognition Letters, Ed. El Sevier, Volume 31, Issue 15, pp. 2353-2356, 1 November 2010

[2-5] Y. Maeda and Y. Fukuda, “FPGA Implementation of Pulse Density Hopfield Neural Network”, In Proc. Int. Joint Conf. on Neural Networks, Florida, USA, 2007

[2-6] Kondo Y, Sawada, Y, “Functional Abilities of a Stochastic Logic Neural Network”, IEEE Trans. on Neural Networks, 3 (3), pp. 434-443, 1992

[2-7] S. Sato, K. Nemoto, S. Akimoto, M Kinjo and K. Nakajima, “Implementation of a New Neurochip Using Stochastic Logic”, IEEE Trans. on Neural Networks, 14 (5), pp. 1122-1127, 2003

[2-8] S. L. Bade, B. Hutchings, “FPGA-Based Stochastic Neural Networks Implementation”, in Proc. IEEE Workshop on FPGAs for Custom Computing Machines, Napa Valley (CA) USA, pp. 189-198, 1994

[2-9] Peter Alfke, “Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators”, XILINX Corp. Application Note 052, USA, 7 July 1996

[2-10] C.L. Janer, J.M. Quero, J.G. Ortega and L.G. Franquelo, “Fully parallel Stochastic computation architecture”, IEEE Transactions on signal processing, Volume 44, Issue 8, pp. 2110–2117, August 1996

[2-11] S.L. Toral, J.M Quero and L.G. Franquelo, “Stochastic pulse coded arithmetic”, IEEE International Symposium on Circuits and Systems (ISCAS 2000), Geneva, Switzerland, 28-31 May 2000

[2-12] B.R. Gaines, “Stochastic computing systems”, Advances in Information Systems Science, Volume. 2, pp.37-172, 1969

[2-13] Huang Zhun, Chen Hongyi, “A truly random number generator based on thermal noise”, ASIC, 2001. Proceedings. 4th International Conference on ASIC (ICASIC 2001), pp. 862-864, Shanghai, China, 2001

[2-14] Rosselló, J.L., Canals, V., de Paúl, I., Bota, S., Morro, A. “A Simple CMOS Chaotic Integrated Circuit”, IEICE Electronics Express, Vol. 5, 1042–1048, 2008

[2-15] M. Delgado-Restituto, A. Rodriguez-Vazquez, “Integrated chaos generators”, Proceedings of the IEEE, vol. 90, pp. 747-767, May 2002.

[2-16] Michael Epstein, Laszlo Hars, Raymond Krasinski, Martin Rosner and Hao Zheng, “Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts”, Lecture Notes in Computer Science (Springer-Verlag), Volume 2779/2003, pp. 152-165, 2003

[2-17] Bellido, M.J.; Acosta, A.J.; Valencia, M.; Barriga, A.; Huertas, J.L.,” Simple binary random number generator”, IEEE Electronics Letters, Volume 28, Issue 7, pp. 617-618, 26 March 1992

[2-18] Kitsos, P. ; Sklavos, N. ; Zervas, N. ; Koufopavlou, O, “A reconfigurable linear feedback shift register (LFSR) for the Bluetooth system”,. The 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2001), Volume 2, pp.991-994, September 2001

[2-19] B. Brown and H. Card, “Stochastic Neural Computation I: Computational Elements”, IEEE Transactions on Computers, Volume. 50, Issue. 9, pp. 891–905, 2001.

[2-20] C.L. Janer, J.M. Quero and L.G. Franquelo, “Fully parallel summation in a new stochastic neural network architecture”, in Proc. IEEE international Conference of Neural Networks, San Francisco (CA) U.S.A, pp. 1498-1503, 1993

[2-21] K. F. Riley, M. P. Hobson, “Essential Mathematical Methods for the Physical Sciences“, Cambridge University Press, 1st edition, 2011

[2-22] Norman Lloyd Johnson, Adrienne W. Kemp, Samuel Kotz, “Univariate discrete distributions“, John Wiley and Sons, 2nd Edition, 565 pp, 1992,

[2-23] Milton Abramowitz, Irene A. Stegun, “Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables”, Courier Dover Publications, New York, 9th reprint, 1972.

[2-24] M.R. Spiegel, J. Siller, R.A. Srinivasan, “Probabilidad y estadística”, Ed. McGraw-Hill, Mexico, 2001

[2-25] J. von Neumann, "First Draft of a Report on the EDVAC", reprinted in Annals of the History of Computing, vol. 15, no. 4., pp. 27-75, 1993.

[2-26] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components", in Automata Studies, C.E. Shannon, Ed. Princeton University Press, p. 43, 1953

[2-27] B.R. Gaines, “A stochastic analog computer”, Standard Telecomunication Laboratories Internal Memorandum, 1-10, December 1965

[2-28] B.R. Gaines, “Stochastic Computing”, in AFIPS Spring Joint Conference, nº 30, Ed. Thompson, pp.149-156, Atlantic City U.S.A, 1967

[2-29] W.J. Poppelbaum, C. Afuso, “Noise Computer”, Quaterly Technical Progress Reports. Submitted April 1965, published Jaunary 1966, University of Illinois: department of Computer Science

[2-30] W.J. Poppelbaum, C. Afuso and J. Esch, “Stochastic computing elements and systems”, in AFIPS FJCC, nº31, 1967

[2-31] L.R. Goke, K.L. Doty, “Design of a Random-Pulse Computer for Classifying Binary Patterns”, IEEE Transactions on Computers, Volume C-2, Issue 12, pp. 1347 – 1354, December 1972

[2-32] J.M. Quero, J.G. Ortega, C.L. Janer, L.G. Franquelo, “VLSI implementation of a fully parallel stochastic neural network”, IEEE international conference on Neural Networks, volume 4, pp. 2040 - 2045, 1994

[2-33] J.M. Quero, S.L. Toral, J.G. Ortega, L.G. Franquelo, “Continuous time filter design using stochastic logic”, 42nd Midwest Symposium on Circuits and Systems, volume 1, pp. 113-116, Las Cruces (NM) U.S.A, 1999

[2-34] B.R. Gaines and J.G. Cleary, "Stochastic Computing in Neural Networks", Research Report, 87/276/24, Dept. Computer Science, University of Calgary, August 1987

[2-35] S.T. Ribeiro, “Random-Pulse machines”, IEEE Transactions on Electronic Computers, volume EC-16, number 3, June 1967

[2-36] J.L. Rosselló, V. Canals, A. Morro, “Hardware implementation of stochastic-based Neural Networks”, Proceedings of International Joint Conference on Neural Networks, IJCNN 2010, Paper number: 5596805, Barcelona Spain, 2010

[2-37] J.L. Rosselló, V. Canals, I. De Paul, J. Segura, “Using stochastic logic for efficient pattern recognition analysis”, Proceedings of International Joint Conference on Neural Networks 2008, IJCNN 2008; Article number: 4633929, pp. 1057-1061, Hong Kong, June 2008

REFERENCIAS CAPÍTULO 3

[3-1] Richard O. Duda, Peter E. Hart, David G. Stork, “Pattern classification”, 2nd Edition, John Wiley & Sons, New York (U.S.A), 2001

[3-2] Ludmila I. Kuncheva, “Combining Pattern Classifiers: Methods and Algorithms”, 1st edition, John Wiley & Sons, Hoboken (U.S.A), 2004

[3-3] A.K. Jain, R.P.W. Duin, Jianchang Mao, “Statistical pattern recognition: a review”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 22, Issue 1, pp. 4-37, 2000

[3-4] Luc Devroye, Laszlo Györfi, Gabor Lugosi, “A Probabilistic Theory of Pattern Recognition”, Springer, Berlin (Germany), 1996

[3-5] Christopher M. Bishop, “Pattern Recognition and Machine Learning”, Springer Science+Business Media. LLC, 2006

[3-6] Sergios Theodoridis, Konstantinos Koutroumbas, “Pattern Recognition”, 2nd Edition, Elsevier Academic Press, San Diego (U.S.A), 2003

[3-7] R.D. Ripley, “Pattern Recognition and Neural Networks”, Cambridge University Press, Cambridge (UK), 1996

[3-8] S. Haykin, “Neural Networks: A Comprehensive Foundation”, 2nd Edition, Prentice Hall, Englewood Cliffs (U.S.A), 1999

[3-9] Geoffrey J. McLachlan, “Discriminant Analysis and Statistical Pattern Recognition”, John Wiley & Sons, Hoboken (U.S.A), 2004

[3-10] T. Kohonen, “Self-Organizing Maps Self-Organizing Maps”, 3rd Edition, Springer Series in Information Sciences, Volume 30, Springer, Berlin (Germany), 2001

[3-11] I. Borg, P. Groenen, “Modern Multidimensional Scaling: theory and applications”, 2nd Edition, Springer-Verlag, New York (U.S.A), 2005

[3-12] R. Brunelli, “Template Matching Techniques in Computer Vision: Theory and Practice”, John Wiley and Sons, 2009

[3-13] Corinna Cortes, Vladimir Vapnik, “Support-Vector Networks”, Machine Learning, Volume 20, Issue 3, pp. 273-297, 1995

[3-14] Rryc Wlodzimierz, “The normal distribution: characterizations with applications“, Springer-Verlag, Ney York (U.S.A), 1995

[3-15] J.L. Rossello, V. Canals, I. de Paul, J. Segura, “Using Stochastic Logic for Efficient Pattern Recognition Analysis”, IEEE International Joint Conference on Neural Networks (IJCNN 2008) (IEEE World Congress on Computational Intelligence), pp. 1057-1061, Hong Kong (China), June 2008

[3-16] V. Canals, A. Morro, J.L. Rosselló, “Stochastic-based pattern-recognition analysis”, Pattern Recognition Letters, Volume 31, Issue 15, pp. 2353-2356, 2010

[3-17] T. Bayes, “An Essay towards solving a Problem in the Doctrine of Chances”, Philosophical Transactions of the Royal Society of London, Volume 53, pp. 370–418, 1763

[3-18] G.A. Carpenter, S. Grossberg, “A massively parallel architecture for a self-organizing neural pattern recognition machine”, Computer Vision, Graphics, and Image Processing, Volume 37, pp. 54-115, 1987

[3-19] M. Oster, R. Douglas, S.C. Liu, “Computation with spikes in a winner-take-all network”, Neural Computation, Volume 21, Issue 9, pp. 2437-2465, 2009

[3-20] M. Riesenhuber, T. Poggio, “Hierarchical models of object recognition in cortex”, Nature Neuroscience, Volume 2, Issue 11, pp. 1019-1025, 1999

[3-21] L. Itti, C. Koch, E. Niebur, “A model of saliency-based visual attention for rapid scene analysis”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 20, Issue 11, pp. 1254-1259, 1998

[3-22] W. Maass, “On the computational power of winner-take-all”, Neural Computation, Volume 12, Issue 11, pp. 2519-2535, 2000

[3-23] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, C. A. Mead, “Winner-take-all networks of O(N) complexity”, Advances in Neural Information Processing Systems 1, Morgan Kaufmann Publishers, San Francisco (CA) (USA), 1989

[3-24] Luis Vazquez, “Métodos Numéricos para la Física y la Ingeniería”, McGraw-Hill Iberoamericana, España, 2008

[3-25] E. Parzen, “On the estimation of a probability density function and mode”, Annals of Mathematical Statistics, Volume 33, pp. 1065-1076, 1962

[3-26] M, Rosenblatt, "Remarks on some nonparametric estimates of a density function", Annals of Mathematical Statistics, Volume 27, pp. 832–837, 1956

[3-27] D.O. Loftsgaarden, C.P. Quesenberry, “A Nonparametric Estimate of a Multivariate Density Function”, The Annals of Mathematical Statistics, Volume 36, Issue 3, pp. 1049-1051, 1965

[3-28] S.P. Awate, R.T. Whitaker, “Unsupervised, Information-Theoretic, Adaptive Image Filtering for Image Restoration”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 28, Issue 3, pp. 364-376, 2006

[3-29] D. Comaniciu, P. Meer, “Mean shift: A robust approach toward feature space analysis”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 24, Issue 5, pp.603-619, 2002

[3-30] M. Wand, M. Jones, “Kernel Smoothing”, Chapman and Hall (CRC Monographs on Statistics & Applied Probability), London (UK), 1995.

[3-31] J.S. Simonoff, “Smoothing Methods in Statistics”, Springer, U.S.A, 1996

[3-32] B.W. Silverman, “Density Estimation for Statistics and Data Analysis”. Chapman and Hall (CRC Monographs on Statistics & Applied Probability), London (UK), 1998

[3-33] J. Hwang, S. Lay, A. Lippman, “Nonparametric Multivariate Density Estimation: A Comparative Study”, IEEE Transactions on Signal Processing, Volume 42, Issue 10, pp. 2795-2810, 1994

REFERENCIAS CAPÍTULO 4

[4-1] W. McCulloch, W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, Bulletin of Mathematical Biophysics, vol. 5, pp. 115-133, 1943

[4-2] Donald Olding Hebb, “The Organization of Behavior”, Wiley & Sons, New York (U.S.A),1949

[4-3] N. Rochester, J.H. Holland, L.H. Haibt, W.L. Duda, “Tests on a cell assembly theory of the action of the brain, using a large digital computer”, IRE Transactions On Information Theory, IT-2, pp. 80-93, 1956

[4-4] Frank Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain”, Psychological Review, Vol. 65, No. 6, pp. 386-408, November 1958.

[4-5] B. Widrow, M.E. Hoff, “Adaptive Switching Circuits”, IRE WESCON Convention Record, 4: pp. 96-104, August 1960

[4-6] M. Minsky, S. Papert, “Perceptrons, An Introduction to Computational Geometry”, MIT Press, Cambridge (MA) (U.S.A), 1969

[4-7] J.A. Anderson, “A simple neural model generating an interactive memory”, Mathematical Biosciences, Volume 14, pp. 197-220, 1972

[4-8] Teuvo Kohonen. “Correlation matrix memories”, IEEE Transactions on Computers, Volume 21, pp. 353-359, 1972

[4-9] Paul J. Werbos. “Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences”, PhD thesis, Harvard University, 1974

[4-10] K. Fukushima, "Cognitron: A self-organizing multilayered neural network", Biological Cybernetics, vol. 20, part [3/4], pp. 121-136, November 1975.

[4-11] S. Grossberg, “How does a brain build a cognitive code?”, Psychological Review, Volume 87, pp. 1–51, April 1980

[4-12] Christoph von der Malsburg, “The correlation theory of brain function”, Internal Report 81-2, Max-Plank –Institute for Biophysical Chemistry, Göttingen (Germany), 1981

[4-13] John J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities”, Proceedings of the national academy of sciences of the United States of America, Volume 79, Issue 8, pp. 2554-2558, 1982

[4-14] John J. Hopfield, D.W. Tank, “Computing with neural circuits: a model”, Science, Volume 233, number 4764, pp. 625-633, 1994

[4-15] Teuvo Kohonen, “Self-organized formation of topologically correct feature maps”, Biological Cybernetics, Volume 43, pp. 59-69, 1982

[4-16] K. Fukushima, S. Miyake, T. Ito. “Neocognitron: a neural network model for a mechanism of visual pattern recognition”, IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-13 (Nb. 3), pp. 826-834, September/October 1983.

[4-17] A.G. Barto, R.S. Sutton, C.A. Anderson, “Neuronlike Adaptative Elements that can Solve Difficult Learning Control Problems” IEEE Transactions on Systems, Man, and Cybernetics, Volume SMC-13 (Nb. 3), pp. 834-846, September/October 1983.

[4-18] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, “Learning representations by back-propagating errors”, Nature, Volume 323, pp. 533-536, October 1986

[4-19] John G. Proakis, Dimitris G. Manolakis, “Digital signal processing”, Pearson Prentice Hall, 4th Edition, 2007

[4-20] C.C. Hsu, D. Gobovic, M.E. Zaghloul, H.H. Szu, “Chaotic neuron models and their VLSI circuit implementations”, IEEE Transactions on Neural Networks, Volume 7, Issue 6, pp. 1339-1350, November 1996

[4-21] M.J. Pearson, A.G. Pipe, B. Mitchinson, K.N. Gurney, C. Melhuish, I. Gilhespy, and M. Nibouche, "Implementing Spiking Neural Networks for Real-Time Signal-Processing and Control Applications: A Model-Validated FPGA Approach", IEEE Transactions on Neural Networks, pp.1472-1487, 2007

[4-22] H.A. Rowley, S. Baluja, T. Kanade, “Neural network-based face detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 20, Issue 1, pp. 23-38, January 1998

[4-23] S. Tamura, M. Nakamura, “Improvements to the noise reduction neural network”, International Conference on Acoustics, Speech, and Signal Processing, ICASSP-90, Volume 2, pp. 825–828, April 1990

[4-24] Huang Jin-Quan, F.L. Lewis, “Neural-network predictive control for nonlinear dynamic systems with time-delay”, IEEE Transactions on Neural Networks, Volume 14, Issue 2, pp. 377-389, 2003

[4-25] K. Minami, H. Nakajima, T. Toyoshima, “Real-time discrimination of ventricular tachyarrhythmia with Fourier-transform neural network”, IEEE Transactions on Biomedical Engineering, Volume 46, Issue 2, pp. 179-185, Febrary 1999

[4-26] M.F. Othman, M.A.M. Basri, “Probabilistic Neural Network for Brain Tumor Classification”, Second International Conference on Intelligent Systems, Modelling and Simulation (ISMS), pp. 136-138, January 2011

[4-27] G.T. Anderson, Zheng Ju, M.R. Clark, R.P. Wyeth, "A neural network model for prediction of progression of left anterior descending coronary artery stenosis in hyperlipidemic patients after a first myocardial infarction", . Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 257-258, 1993

[4-28] D.S. Broomhead, D. Lowe, "Multi-variable functional interpolation and adaptive networks". Complex Systems, Volume 2, pp. 321-355, 1988

[4-29] W. Maass, G. Schnitger, E.D. Sontag, "On the computational power of sigmoid versus Boolean threshold circuits" (extended abstract), Proceedings of the 32nd annual IEEE symposium on Foundations of computer science, Los Alamitos (CA) (USA), pp. 767-776, 1991.

[4-30] W. Maass, G. Schnitger, E.D. Sontag, "A comparison of the computational power of sigmoid and Boolean threshold circuits", Theoretical Advances in Neural Computation and Learning, pp. 127-151, 1994

[4-31] Bhaskar DasGupta, Georg Schnitger, "Efficient Approximation with Neural Networks: A Comparison of Gate Functions", Technical Report, Pennsylvania State University, June 1992

[4-32] Vladimir N. Vapnik, "Statistical Learning Theory", Springer Verlag, 1996

[4-33] Eric R. Kandel, James Harris Schwartz, Thomas M. Jessell, "Principles of neural science", 4th edition, McGraw-Hill (Health Professions Division), 2000 (first edition 1991)

[4-34] Wulfram Gerstner, Werner M. Kistler, "Spiking Neuron Models", Cambridge University Press, 2002

[4-35] S.J. Thorpe, J. Gautrais, “Rapid Visual Processing using Spike Asynchrony Neural Information Processing Systems”, pp. 901-907, M.I.T. Press, Cambridge (U.S.A), 1997

[4-36] Wolfgang Maass; Christopher M Bishop, “Pulsed neural networks”, M.I.T. Press, Cambridge (U.S.A), 1999

[4-37] J. Vreeken, “Spiking neural networks, an introduction”, Technical Report UU-CS-2003-008, Institute for Information and Computing Sciences, Universiteit Utrecht, 2002.

[4-38] Wofgang Maas, “Networks of spiking neurons: the third generation of neural network models” Transactions of the Society for Computer Simulation International, Special issue: simulation methodology in transportation systems, Volume 14, Issue 4, December. 1997

[4-39] R.J. Williams, D. Zisper, "Gradient Based Learning Algorithms for Recurrent Connectionist Networks", Technical Report, (College of Computer Science) Northeastern University, NU-CCS-90-9, pp.1-45, 1990

[4-40] K. Lang, G.E. Hinton, “The Development of TDNN Architecture for Speech Recognition”, Technical Report CMU-CS-88-152, Carnegie-Mellon University, 1988

[4-41] Alexander Waibel, “Modular construction of time delay neural networks for speech recognition”, Neural Computation, Volume 1, pp. 39-46, 1989

[4-42] G.E. Hinton, T.J. Sejnowski, "Learning and Relearning in Boltzmann Machines". Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations, MIT Press, Cambridge (U.S.A), pp. 282–317, 1986

[4-43] C.M. Gray, P. Konig, A.K. Engel, W. Singer, “Oscillatory responses in cat visual cortex exhibit intercolumnar synchronization which reflects global stimulus properties”, Nature, Volume 338, pp. 334–337, 1989

[4-44] M.N. Shadlen, J.A. Movshon, "Synchrony unbound: a critical evaluation of the temporal binding hypothesis", Neuron, volume 24, pp. 67-77, 1989

[4-45] W. Maass, “Lower Bounds for the Computational Power of Networks of Spiking Neurons”, Neural Computation, volume 8, issue 1. pp. 7-42, 1996

[4-46] R. Douglas, K. Martin, “Opening the grey box”. Trends in Neurosciences, volume 14, issue 7,pp. 286-293, 1991

[4-47] A.K. Jain, Mao Jianchang; K.M. Mohiuddin, “Artificial neural networks: a tutorial”, Computer, Volume 29, Issue 3, pp. 31-44, 1996

[4-48] T. Kohonen, “Self-Organizing Maps”, Springer Series in Information Sciences, volume 30, 1995.

[4-49] Dale Purves, George J. Augustine, David Fitzpatrick, William C. Hall, Anthony-Samuel LaMantia, James O. McNamara, and Leonard E. White, “NeuroScience”, 4th Edition, Sinauer Associates, Inc., 2012

[4-50] B. Brown and H. Card, “Stochastic Neural Computation I: Computational Elements”, IEEE Transactions on Computers, Volume. 50, Issue. 9, pp. 891–905, 2001.

[2-51] C.L. Janer, J.M. Quero and L.G. Franquelo, “Fully parallel summation in a new stochastic neural network architecture”, in Proc. IEEE international Conference of Neural Networks, San Francisco (CA) U.S.A, pp. 1498-1503, 1993

[4-52] C.L. Janer, J.M. Quero, J.G. Ortega and L.G. Franquelo, “Fully parallel Stochastic computation architecture”, IEEE Transactions on signal processing, Volume 44, Issue 8, pp. 2110–2117, August 1996

[4-53] A. Torralba, F. Colodro, E. Ibanez, L.G. Franquelo, “Two digital circuits for a fully parallel stochastic neural network”, IEEE Transactions on Neural Networks, volume 6, issue 5, pp. 1264-1268, 1995

[4-54] S. Sato, K. Nemoto, S. Akimoto, M. Kinjo, K. Nakajima, “Implementation of a new neurochip using stochastic logic”, IEEE Transactions on Neural Networks, volume 14, issue 5, pp. 1122-1127, 2003

[4-55] Y. Kondo, Y. Sawada, “Functional abilities of a stochastic logic neural network”, IEEE Transactions on Neural Networks. volume 3, issue 3, pp. 434-443, 1992

[4-56] Y. Maeda, Y. Fukuda, “FPGA Implementation of Pulse Density Hopfield Neural Network”, International Joint Conference on Neural Networks (IJCNN 2007), pp. 700-704, Orlando (FL) (U.S.A), August 2007

[4-57] Wolfgang Banzhaf, “On a Simple Stochastic Neuron-Like Unit”, Biological Cybernetics, volume 60, pp. 153- 160, 1988

[4-58] S.L. Bade, B.L. Hutchings, “FPGA-based stochastic neural networks-implementation”, On proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, Napa Valley (CA) (U.S.A), pp. 189-198, april 1994

[4-59] D.E. Van Den Bout, T.K. Miller, “A digital architecture employing stochasticism for the simulation of Hopfield neural nets”, IEEE Transactions on Circuits and Systems, volume 36, issue 5, pp. 732-738, 1989

[4-60] Judith E. Dayhoff, “Neural network architectures: An introduction”, Wiley & Sons, 1990

[4-61] G. Moon, M.E. Zaghloul, R.W. Newcomb, “VLSI implementation of synaptic weighting and summing in pulse coded neural-type cells”, IEEE Transactions on Neural Networks, volume 3, issue 3, pp. 394-403, 1992

[4-62] Sato Shigeo, Yumine Manabu, Yama Takayuki, Murota Junichi, Nakajima Koji, Sawada Yasuji, “LSI Neural Chip of Pulse-Output Network with Programmable Synapse”, IEICE TRANSACTIONS on Electronics, volume E78-C(1),pp. 94-100, 1995

[4-63] J.L. Rosselló, V. Canals, A. Morro, “Hardware implementation of stochastic-based Neural Networks”, Proceedings of International Joint Conference on Neural Networks 2010, IJCNN 2010; Article number 5596805, Barcelona (Spain); July 2010

[4-64] M.S. Tomlinson., D.J. Walker, M.A. Sivilotti, “A digital neural network architecture for VLSI”, On proceedings International Joint Conference on of Neural Networks (IJCNN 1990), volume 2, pp. 545-550, june 1990

[4-65] T. McKenna, J. Davis, S.F. Zornetzer, “Single Neuron Computation”, Academic Press, (Chapter 15), 1991

[4-66] Wofgang Maas, “Fast sigmoidal networks via spiking neurons”, Neural computation, volume 9, pp. 279-304, 1997

[4-67] T.G. Clarkson, Y. Guan, J.G. Taylor, D. Gorse, ”Generalization in probabilistic RAM nets”, IEEE Transactions on Neural Networks, volume 4, issue 2, pp. 360-363, 1993

[4-68] T.G. Clarkson, J.G. Taylor, "Hardware-rcalisable models of neural processing", in Proceedings of Conference of ArtificiaI Neural Networks conference,pp. 242-246, London (UK), 1989

[4-69] Jieyu Zhao, John Shawe-Taylor, Max van Daalen, “Learning in Stochastic Bit Stream Neural Networks”, Neural Networks, volume 9, issue 6, pp. 991–998, 1996

[4-70] Krzysztof Patan, Thomas Parisini “Stochastic learning methods for dynamic neural networks:simulated and real-data comparisons”, in Proceedings of the American Control Conference, pp., 2577-2582, Anchorage (AK) (U.S.A), may 2002

[4-71] P.D. Hortensius, R.D. McLeod, H.C. Card, “Parallel random number generation for VLSI systems using cellular automata”, IEEE Transactions on Computers, volume 38, issue 10, pp. 1466-1473, 1989

[4-72] G. E. Hinton, T. J. Sejnowski, “Learning and relearning in Blotzman machines”, in Parallel distributed processing: explorations in the microstructure of cognition, volume 1, chapter 7, MIT press, Cambridge (MA) (U.S.A), 1986

[4-73] G.E. Hinton, P. Dayan, B.J. Frey, R.M. Neal, “The "wake-sleep" algorithm for unsupervised neural networks”, Science, volume 268, number 5214, pp. 1158-1161, may 1995

[4-74] Howard C. Card, “Doubly Stochastic Poisson Processes in Artificial Neural Learning”, IEEE transactions on Neural Networks, volume 9. pp. 229-231, 1998

[4-75] Wulfram gerstner, “Rapid signal transmission by populations of spiking neurons”, In Proceedings of International Conference on Artificial Neural Networks (ICANN 99)., volume 1, pp. 7-12, Edinburgh (UK), 1999

[4-76] H.R. Wilson, “Spikes Decisions and Actions – Dynamical Foundations of Neuro-Science”, 1th Edition, The Oxford University Press, Oxford (UK), 1999

[4-77] E.M. Izhikevich, “Which model to use for cortical spiking neurons”, IEEE Transactions on Neural Networks, Volume 15, Issue 5, pp.:1063–1070, 2004

[4-78] N.F. Rulkov, “Modeling of spiking-bursting neural behavior using two- dimensional map”, Physical Review E, Volume 65, Issue 41922–1–9, 2002.

[4-79] E.M. Izhkevich, “Resonate-and-fire neurons”, Neural Networks, Volume 14, pp. 883–894, 2001

[4-80] A.L. Hodgkin, A.F. Huxley, “A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes”, Journal of Physiology, Volume 117, pp. 500–544, 1952

[4-81] C. Koch, “Biophysics of Computation: Information Processing in Single Neurons”, 1st ecition, Oxford University Press, New York (U.S.A), 1999.

[4-82] H. Hasegawa, “Responses of a Hodgkin-Huxley neuron to various types ofspike-train inputs”, Physical Review E, Volume 61, Issue 1, pp.718–726, 2000

[4-83] R. FitzHugh, “Impulses and physiological states in theoretical models of nerve membrane”, Biophysical Journal, Volume 1, pp.445–466, 1961

[4-84] J. Nagumo, S. Arimoto, S. Yoshizawa, “An active pulse transmission line simulating nerve axon”, Proceedings of Institute of Radio Engineers (IRE), Volume 50, pp.2061–2070, 1962

[4-85] C. Morris, H. Lecar, “Voltage oscillations in the barnacle giant muscle fiber”, Biophysical Journal, Volume 35, pp.193–213, 1981

[4-86] J.L. Hindmarsh, R.M. Rose, “A model of the nerve impulse using two first-order differential equations”, Nature, Volume 296, pp.162–164, 1982

[4-87] J.L. Hindmarsh, R.M. Rose, “A model of neuronal bursting using three coupled first order differential equations”, Proceedings of Royal Society B: Biological Sciences, London (UK), Volume 221, pp.87–102, 1984

[4-88] E.M. Izhikevich, “Simple model of spiking neurons”, IEEE Transactions on Neural Networks, Volume 14, Issue 6, pp.1569–1572, 2003

[4-89] Santiago Ramon y Cajal, “Texture of the Nervous System of Man and the Vertebrates”, Springer Verlag, NewYork (U.S.A), 2003

[4-90] J.M. Bower, D. Beeman, “The Book of Genesis – Exploring Realistic Neural Models with the GEneral NEural SImulation System”. Internet Version, 2nd edition, 2003

[4-91] K. Aihara, I. Tokuda, “Possible neural coding with interevent intervals of synchronous firing”, Physical Review E, Volume 66, Issue 26212–1–5, 2002

[4-92] F. Rieke, D. Warland, R. de Ruyter van Steveninck, W. Bialek, “Spikes – Exploring the Neural Code”, 1st Edition, The MIT Press, Cambridge (U.S.A), 1997

[4-93] S.M. Bohte, “The evidence of information processing with precise spike-times: A survey”, Natural Computing, Volume 3, pp. 195–206, 2004

[4-94] E.D. Adrian, Y. Zotterman, “The impulses produced by sensory nerve endings: Part II: The response of a single end organ”, Journal of Physiology, Volume 61, pp. 151-171, 1926

[4-95] R. Stein, E. Gossen, K. Jones, “Neuronal variability: noise or part of the signal?”, Nature Reviews Neuroscience, Volume 6, pp. 389–397, 2005

[4-96] W. Gerstner, “Neural codes: Firing rates and beyond”, Proceedings of the National Academy of Sciences of the United States of America, Volume 94, pp. 12740-12741, 1997

[4-97] P. Dayan, L.F. Abbott, “Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems”, The MIT Press, Cambridge (U.S.A), 2001

[4-98] S.J. Thorpe, “Spike arrival times: A highly efficient coding scheme for neural networks”, in book “Parallel processing in neural systems and computers”, Editors R. Eckmiller, G. Hartmann, G. Hauske, North-Holland Elsevier, 1990

[4-99] D.A. Butts, C. Weng, J. Jin, Chun-I Yeh, N.A. Lesica1, Jose-Manuel Alonso, G.B. Stanley, “Temporal precision in the neural code and the timescales of natural vision”, Nature, Volume 449,pp. 92-95, 2007

[4-100] M.A. Montemurro, M.J. Rasch, Y. Murayama, N.K. Logothetis, S. Panzeri, “Phase-of-Firing Coding of Natural Visual Stimuli in Primary Visual Cortex”, Current Biology, Volume 18, Issue 5, 2008

[4-101] P. Fries, D. Nikolić, W. Singer, “The gamma cycle”, Trends in Neurosciences, Volume 30, Issue 7, pp.309-316, 2007

[4-102] M.N. Havenith, S. Yu, J. Biederlack, N.H. Chen, W. Singer, D. Nikolić, “Synchrony makes neurons fire in sequence – and stimulus properties determine who is ahead”, Journal of Neuroscience, Volume 31, Issue 23, pp. 8570-8584, 2011

[4-103] E. Geoffrois, J.M. Edeline, J.F. Vibert, 1994, “Learning by Delay Modifications”, pp. 133-138, in “Computation in neurons and neural systems”, Editor Frank H. Eeckman, Kluwer Academic Publishers, 1994

[4-104] S. Wu, S. Amari, H. Nakahara, “Population Coding and Decoding in a Neural Field: A Computational Study”, Neural Computation, Volume 14, pp. 999-1026, 2002

[4-105] J.H.R. Maunsell, D.C. Van Essen, “Functional properties of neurons in middle temporal visual area of the Macaque monkey. I. Selectivity for stimulus direction, speed, and orientation”, Journal of Neurophysiology, Volume 49, pp. 1127–1147, 1983

[4-106] D.H. Hubel, T.N. Wiesel, “Receptive fields of single neurons in the cat's striate cortex”, Journal of Physiology, Volume148, pp. 574-591, 1959

[4-107] R. Malaka, S. Buck, “Solving nonlinear optimization problems using networks of spiking neurons”, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000), Volume 6, pp. 486-491, Como (Italy), 2000

[4-108] D.M. Sala, K.J. Cios, “Solving graph algorithms with networks of spiking neurons”, IEEE Transactions on Neural Networks, Volume 10, Issue 4. pp. 953-957, 1999

[4-109] T. Oya, T. Asai, Y. Amemiya, A. Schmid, Y. Leblebici, “Single-electron circuit for inhibitory spiking neural network with fault-tolerant architecture”, IEEE International Symposium on Circuits and Systems (ISCAS 2005), Volume 3, pp. 2535-2538, 2005

[4-110] J.L.Rosselló, I. De Paúl, V. Canals, “Self-configuring Spiking Neural Networks”, IEICE Electronics Express, Volume 5, Issue 22, pp. 921-926, 2008

[4-111] J.L. Rossello, V. Canals, A. Morro, J. Verd, “Chaos-based mixed signal implementation of spiking neurons”, International Journal of Neural Systems, Volume 19, Issue 6, pp. 465-471, 2009

[4-112] J.L. Rosselló, V. Canals, A. Morro, I. De Paúl, “Practical hardware implementation of self-configuring neural networks”, Proceedings of 6th International Symposium on Neural Networks, ISNN 2009; paper code 77071, Wuhan (China); May 2009, this paper is accessible at “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)”, Volume 5553 LNCS, Issue PART 3, pp. 1154-1159, 2009

[4-113] J.L. Rosselló, I. De Paúl, V. Canals, A. Morro, “Spiking neural network self-configuration for temporal pattern recognition analysis”, 19th International Conference on Artificial Neural Networks, paper code 77563, ICANN 2009; Limassol (Cyprus); September 2009, this paper is accessible at “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)”, Volume 5768 LNCS, Issue PART 1, pp. 421-428, 2009

[4-114] N. Langlois, P. Miche, A. Bensrhair, “Analogue circuits of a learning spiking neuron model”, In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2000), Volume 4, pp. 485-489, Como (Italy), 2000

[4-115] T. Dowrick, S. Hall, L. McDaid, O. Buiu, P. Kelly, “A Biologically Plausible Neuron Circuit”, International Joint Conference on Neural Networks (IJCNN 2007), pp. 715-719, Orlando (U.S.A), 2007

[4-116] M.J. Pearson, C. Melhuish, A.G. Pipe, M. Nibouche, L. Gilhesphy, K. Gurney, B. Mitchinson, “Design and FPGA implementation of an embedded real-time biologically plausible spiking neural network processor”, International Conference on Field Programmable Logic and Applications, pp. 582-585, 2005

[4-117] H. Torikai, A. Funew, T. Saito, “Approximation of Spike-trains by Digital Spiking Neuron”, International Joint Conference on Neural Networks (IJCNN 2007), pp. 2677-2682, Orlando (U.S.A), 2007

[4-118] S. Renaud, J. Tomas, Y. Bornat, A. Daouzli, S. Saighi, “Neuromimetic ICs with analog cores: an alternative for simulating spiking neural networks”, IEEE International Symposium on Circuits and Systems (ISCAS 2007), pp. 3355 - 3358, New Orleans (U.S.A), 2007

[4-119] G. Indiveri, E. Chicca, R. Douglas “A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike Timing Dependent Plasticity”, IEEE Transactions on Neural Networks, Volume 17, Issue 1, pp. 211-221, 2006

[4-120] J.L. Rosselló, V. Canals, V., I. de Paúl, S. Bota, A. Morro, “A Simple CMOS Chaotic Integrated Circuit”, IEICE Electronics Express, Volume 5, pp. 1042–1048, 2008

[4-121] F. Caignet, S. Delmas-Bendhia, E. Sicard, “The challenge of signal integrity in deep-submicrometer CMOS technology”, Proceedings of the IEEE, Volume 89, Issue 4, pp. 556-573, 2001

[4-122] S. Delmas-Bendhia, F. Caignet, E. Sicard, M. Roca, “On-chip sampling in CMOS integrated circuits”, IEEE Transactions on Electromagnetic Compatibility, Volume 41, Issue 4, pp. 403-406, 1999

[4-123] M.T. Rosenstein, J.J. Collins, C.J. De Luca, “A practical method for calculating largest Lyapunov exponents from small data sets”, Physica D, Volume 65, pp. 117-134, 1993

[4-124]] J.L. Rosselló, V. Canals, A. Morro, “Probabilistic-based Neural Network Implementation”, IEEE International Joint Conference on Neural Networks (IJCNN 2012) (IEEE World Congress on Computational Intelligence), Aceptado para su presentación oral, Junio 2012

REFERENCIAS CAPÍTULO 5

[5-1] J.L. Rosselló, V. Canals, V., I. de Paúl, S. Bota, A. Morro, “A Simple CMOS Chaotic Integrated Circuit”, IEICE Electronics Express, Volume 5, pp. 1042–1048, 2008

[5-2] J.L.Rosselló, I. De Paúl, V. Canals, “Self-configuring Spiking Neural Networks”, IEICE Electronics Express, Volume 5, Issue 22, pp. 921-926, 2008

[5-3] J.L. Rossello, V. Canals, A. Morro, J. Verd, “Chaos-based mixed signal implementation of spiking neurons”, International Journal of Neural Systems, Volume 19, Issue 6, pp. 465-471, 2009

[5-4] V. Canals, A. Morro, J.L. Rosselló, “Stochastic-based pattern-recognition analysis”, Pattern Recognition Letters, Volume 31, Issue 15, pp. 2353-2356, 2010

[5-5] J.L. Rosselló, S. Bota, V. Canals, I. De Paul, J. Segura, “A fully CMOS low-cost chaotic neural network”, Proceedings of IEEE International Conference on Neural Networks (JCNN 06), Article number 1716157, pp. 659-663, 2006

[5-6] J.L. Rosselló, V. Canals, I. de Paul, J. Segura, “Using Stochastic Logic for Efficient Pattern Recognition Analysis”, IEEE International Joint Conference on Neural Networks

(IJCNN 2008) (IEEE World Congress on Computational Intelligence), pp. 1057-1061, Hong Kong (China), June 2008

[5-7] J.L. Rosselló, V. Canals, A. Morro, I. De Paúl, “Practical hardware implementation of self-configuring neural networks”, Proceedings of 6th International Symposium on Neural Networks, ISNN 2009; paper code 77071, Wuhan (China); May 2009, this paper is accessible at “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)”, Volume 5553 LNCS, Issue PART 3, pp. 1154-1159, 2009

[5-8] J.L. Rosselló, I. De Paúl, V. Canals, A. Morro, “Spiking neural network self-configuration for temporal pattern recognition analysis”, 19th International Conference on Artificial Neural Networks, paper code 77563, ICANN 2009; Limassol (Cyprus); September 2009, this paper is accessible at “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)”, Volume 5768 LNCS, Issue PART 1, pp. 421-428, 2009

[5-9] J.L. Rosselló, V. Canals, A. Morro, “Hardware implementation of stochastic-based Neural Networks”, Proceedings of International Joint Conference on Neural Networks 2010, IJCNN 2010; Article number 5596805, Barcelona (Spain); July 2010

[5-10] J.L. Rosselló, V. Canals, A. Morro, “Probabilistic-based Neural Network Implementation”, IEEE International Joint Conference on Neural Networks (IJCNN 2012) (IEEE World Congress on Computational Intelligence), (Aceptado para su presentación oral), June 2012

REFERENCIAS ANEXO A

[A-1] Edward Lorenz, “The Essence of Chaos”, University of Washington Press, Seattle (U.S.A), 1993

[A-2] Jianjun Hou, Yuhui Zhan, Tao Wang, “Secure communication via synchronized chaotic circuit”, 6th International Conference on Signal Processing, Volume 2, pp. 1552-1555, Beijing (China), 2002

[A-3] A.J.K Klomkarn, P. Sooraksa, “Further investigation on trajectory of chaotic guiding signals for robotic systems”, IEEE International Symposium on Communications and Information Technology, (ISCIT 2004), Volume 2, pp. 1166-1170, 2004

[A-4] M. Delgado-Restituto, A. Rodriguez-Vazquez, “Integrated chaos generators”, Proceedings of the IEEE, Volume 90, Issue 5, 2002

[A-5] Hua-Chin Wang, Yih-Long Tseng, Hon-Chen Cheng, Ron Hu, “Switched-current 3-bit CMOS wideband random signal generator”, Southwest Symposium on Mixed-Signal Design, pp. 186–189, 2003

[A-6] M.J. Bellido, A.J. Acosta, M. Valencia, A. Barriga, J.L. Huertas, “Simple binary random number generator”, Electronics Letters, Volume 28, Issue 7, pp. 617-618, 1992

[A-7] T. Tanaka, E. Hiura, “Dynamical behavior of a chaotic neural and its application to optimization problems”, Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '05), Volume 2, pp. 753-757, 2005

[A-8] T. Stojanovski, L. Kocarev, “Chaos-based random number generators-part I: analysis [cryptography]”, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Volume 48, Issue 3, pp. 281-288, 2001

[A-9] L.O. Chua, “The Genesis of Chua’s Circuit”, Archiv für Elektronik und Ubertragung-stechnik, Volume 46, pp. 250-257, 1992

[A-10] T. Matsumoto, “A chaotic attractor from Chua's circuit”, IEEE Transactions on Circuits and Systems, Volume 31, Issue 12, 1984

[A-11] J.M. Cruz, L.O. Chua, “An IC chip of Chua's circuit”, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, Volume 40, Issue 10, pp. 614-625, 1993

[A-12] Y. Hosokawa, Y. Nishio, A. Ushida, “A design method of chaotic circuits using an oscillator and a resonator”, The IEEE International Symposium on Circuits and Systems (ISCAS 2001), Volume 2, pp. 373-376, 2001

[A-13] H. Nakano, K. Miyachi, T. Saito, “A simple nonautonomous chaotic circuit with a periodic pulse-train input”, Proceedings of the International Symposium on Circuits and Systems (ISCAS 2003), Volume 3, pp. 108-111, 2003

[A-14] A.S. Elwakil, “Nonautonomous pulse-driven chaotic oscillator based on Chua's circuit”, Proceedings of the International Symposium on Circuits and Systems (ISCAS 2003), Volume 3, pp. 136-139, 2003

[A-15] A.G. Radwan, A.M. Soliman, A.L. El-Sedeek, “MOS realization of the double-scroll-like chaotic equation”, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, Volume 50, Issue 2, pp. 285-288, 2003

[A-16] E. Lindberg, K. Murali, A. Tamasevicius, “The smallest transistor-based nonautonomous chaotic circuit”, IEEE Transactions on Circuits and Systems II: Express Briefs, Volume 52, Issue 10, pp. 661-664, 2005

[A-17] C.C. Hsu, D. Gobovic, M.E. Zaghloul, H.H. Szu, “Chaotic neuron models and their VLSI circuit implementations”, IEEE Transactions on Neural Networks, Volume 7, Issue 6, pp. 1339-1350, 1996

[A-18] B.R. Gaines, “R68-18 Random Pulse Machines”, IEEE Transactions on Computers, Volume 17, Issue 4, pp. 410-410, 1968

[A-19] J.L. Rosselló, V. Canals, I. de Paúl, S. Bota, A. Morro, “A simple CMOS chaotic integrated circuit”, IEICE Electronics Express, Volume 5, Issue 24, pp. 1042-1048, 2008

[A-20] M.T. Rosenstein, J.J. Collins, C.J. De Luca, “A practical method for calculating largest Lyapunov exponents from small data sets”, Physica D, Volume 65, pp. 117-134, 1993

[A-21] NIST (National Institute of Standards and Technology), “A statistical Test Suite for Random and Pseudorandom number generators for cryptographic applications” (NIST SP 800-22)

[A-22] NIST (National Institute of Standards and Technology), Security requirements for Cryptographic Modules (FIPS PUB 140-2, 2001)

[A-23] J.L. Rosselló, S. Bota, V. Canals, I. De Paul, J. Segura, “A fully CMOS low-cost chaotic neural network”, Proceedings of IEEE International Conference on Neural Networks (JCNN 06), Article number 1716157, pp. 659-663, 2006

A N E X O S

ANEXO A: CIRCUITO CAÓTICO

En este Anexo se presenta el diseño y la caracterización experimental de un oscilador caótico fabricado mediante una tecnología CMOS de 0.35μm. El principio de operación se basa en iterar una función de transferencia en forma de letra N generada mediante una pequeña red neuronal analógica. El circuito desarrollado se compone de 13 transistores MOS, siendo por ello su área de integración muy reducida. A su vez, dicho circuito dispone de una salida analógica y otra digital que hemos usado para generar secuencias de bits puramente aleatorias.

INTRODUCCIÓN

El término caos [A-1] se refiere a aquel comportamiento aparentemente errático o impredecible que presentan algunos sistemas dinámicos aunque su formulación matemática sea en principio determinista. En el ámbito científico a lo largo de las últimas décadas se ha despertado un gran interés en el diseño y análisis de sistemas caóticos, dado su paralelismo con fenómenos observables en la naturaleza. En la práctica podemos hallar gran cantidad de diseños e implementaciones de circuitos caóticos en campos tan dispersos de la ciencia y la tecnología, como pueden ser: las comunicaciones seguras [A-2], los sistemas de control robótico [A-3], las fuentes de ruido [A-4] (generalmente usadas en los sistemas de reconocimiento del habla). No obstante, los circuitos caóticos también son usados para la encriptación de redes de datos (Ethernet) fijas e inalámbricas [A-5], y para test analógico y digital de circuitos integrados [A-6].
En particular las redes neuronales con un comportamiento caótico pueden ser usadas para la resolución de problemas de optimización [A-7] y para la generación de números aleatorios [A-8].

A lo largo de las tres últimas décadas se han desarrollado la mayoría de los diseños e implementaciones de osciladores caóticos. El circuito de *Chua* [A-9, A-10] es uno de los circuitos caóticos más conocidos y estudiados (Figura A-1), que se compone de tan solo

cuatro dispositivos pasivos (dos condensadores, una resistencia y un inductor) y de un elemento resistivo no lineal llamado diodo *Chua*. Éste se construye mediante amplificadores operacionales y se comporta como una resistencia de valor negativo. Una integración en silicio del circuito de *Chua* se presentó en [A-11] usando una tecnología CMOS de 2µm, y ocupando un área de silicio de 2.5 x 2.8mm². Diversos investigadores han presentado otros osciladores caóticos [A-12 - A-14] basados en la combinación de elementos pasivos (resistencias, condensadores, inductores) y elementos no lineales (semejantes al diodo *Chua*) implementados mediante diversos amplificadores operacionales, lo que implica que para su integración se requiera de gran cantidad de área de silicio. Para solucionar este problema los investigadores a lo largo de los años han propuesto diseños más simples para minimizar el área de integración, por ejemplo el circuito propuesto por *Radwan et al.* [A-15] que se compone de 29 transistores *CMOS*. Dicho circuito implementa una ecuación caótica de doble desplazamiento. Más recientemente se ha presentado un circuito caótico con tan sólo 20 transistores *MOS* [A-16], el cual se compone de un transistor, dos condensadores y dos resistencias, pero requiere de una señal sinusoidal externa para su correcto funcionamiento.

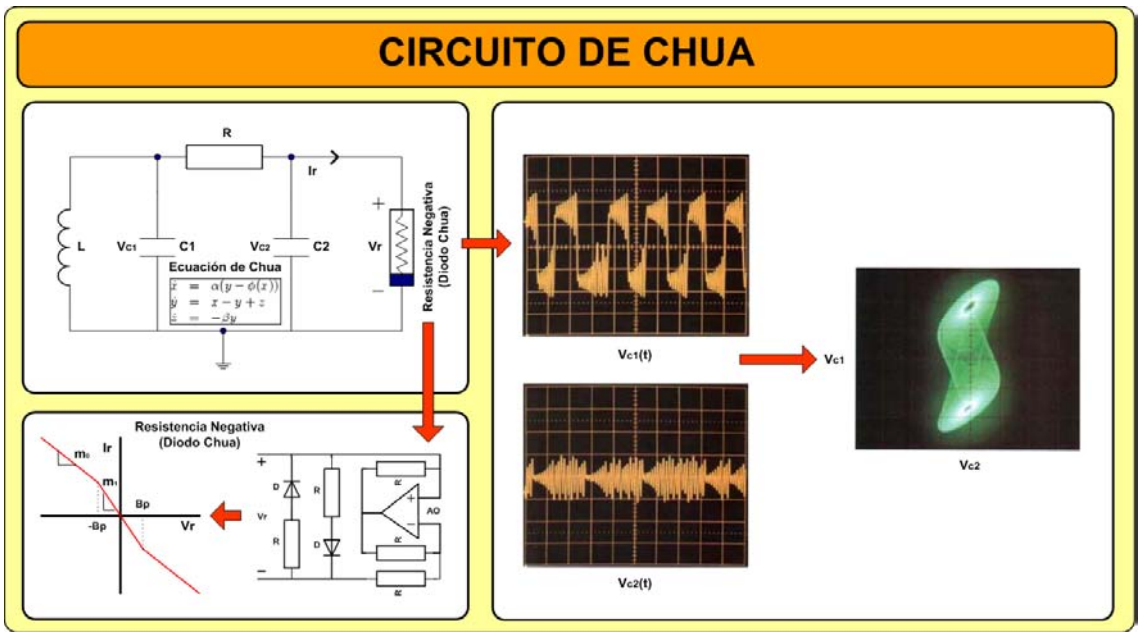

Figura A-1: Circuito caótico de *Chua*

Una vez introducido el marco en el cual se ha desarrollado nuestra aportación, pasaremos a describir el diseño y caracterización del circuito caótico desarrollado, que se compone de 21 transistores *MOS* fabricados con una tecnología *CMOS* de 0.35µm, de la empresa austriaca "*Austria Micro Systems*" (AMS). El área de silicio ocupada por el circuito es de 47 x 57 µm². Dicho circuito dispone de dos salidas caóticas una analógica y otra digital.

EL CIRCUITO DESARROLLADO

El esquema del circuito caótico que se ha diseñado y fabricado [A-19, A-23] usando una tecnología CMOS de 0.35µm de (AMS) se presenta en la Figura A-2.

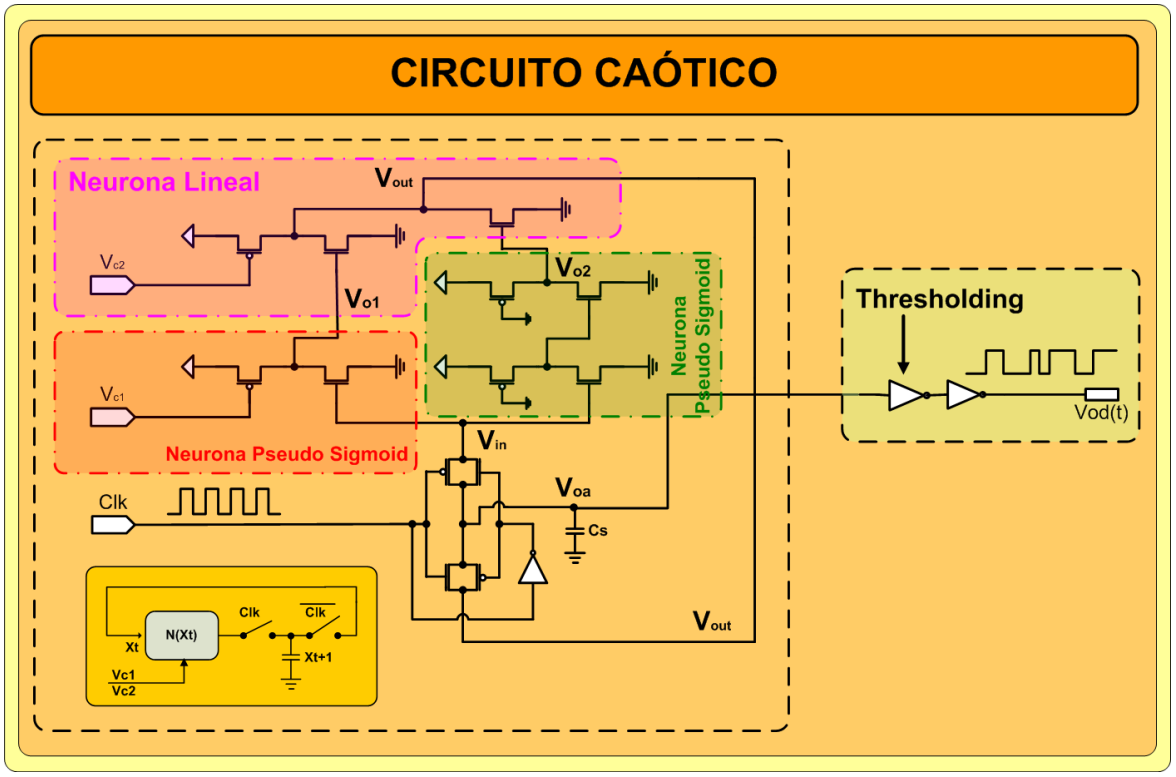

Figura A-2: Esquema del circuito caótico basado sobre una red neuronal analógica

Para obtener un comportamiento caótico a la salida de nuestro circuito, hemos diseñado una pequeña red neuronal analógica compuesta por tres pseudo-neuronas que implementan una función de transferencia "*N(x)*" en forma de la letra "N" mayúscula. Dicha red neuronal analógica se compone de dos neuronas en la capa oculta y una neurona en la capa de salida. Las dos neuronas de la capa oculta disponen de un elemento no lineal cuya función de

transferencia (función de activación) se asemeja a una función pseudo-sigmoidal "*f(x)*", mientras que la neurona de la capa de salida tiene un comportamiento semejante a una neurona lineal, realizando la suma ponderada por los pesos "W_{ij}" de cada una de las entradas provenientes de las neuronas de la capa oculta "I_j". En el caso de las neuronas de la capa oculta su salida "y_i" se regirá aproximadamente por la Ecuación [A-1] (a):

$$\begin{cases} y_i = f\left(\sum_j w_{ij} * I_j\right) \approx \dfrac{1}{1+e^{-\left(\sum_j W_{ij} * I_j\right)}} & \text{(a)} \\ y_i = f\left(\sum_j w_{ij} * I_j\right) \approx w_{i1} \cdot x_1 + w_{i2} \cdot x_2 & \text{(b)} \end{cases} \qquad \text{Ecuación [A-1]}$$

Mientras que la neurona de la capa de salida se regirá aproximadamente por la Ecuación [A-1] (b). No obstante, para que esta red neuronal (compuesta por tan sólo 9 transistores MOS) opere de la forma deseada presentando un comportamiento caótico a su salida es necesario situar el circuito en un determinado punto de operación, ajustando externamente la red "$N(x)$" mediante dos tensiones analógicas como son "V_{c1}" y "V_{c2}". Éstas han sido experimentalmente ajustadas a "$V_{c1} = 48.5mV$" y "$V_{c2} = 570.0mV$" para obtener el comportamiento caótico deseado para la red. Estas dos tensiones (V_{c1} y V_{c2}) son las encargadas de modificar la resistencia de conducción de los transistores *P-MOS* de la neurona de capa de salida y de una de las neuronas de la capa oculta, consecuentemente modificando la función de transferencia de toda la red neuronal. Una función de activación similar a una Sigmoidal se obtiene a la salida de cada neurona de la capa oculta, debido a la característica no lineal de los transistores *MOS*. Los pesos de la red neuronal se han prefijado para cada neurona ajustando la anchura del canal de los transistores *N-MOS*.

Finalmente para que opere la red neuronal desarrollada se requiere que el estado actual del sistema "x_n" se realimente en el sistema mediante la función de transferencia "$N(x_n)$" para posteriormente actualizarse en cada iteración, tal y como se describe en la Ecuación [A-2].

$$x_{n+1} = N(x_n)$$ Ecuación [A-2]

La actualización del valor de salida de la red se realiza mediante dos puertas de trasmisión presentes en el circuito (cuya activación se haya invertida una respecto de la otra, mediante un inversor *CMOS*) y a su vez controladas por una señal de reloj externa, permitiendo así el almacenamiento de la tensión actual de salida de la red y su posterior iteración. En el caso que la señal de reloj "*Clk*" se encuentre fija a nivel bajo '0' la tensión analógica de salida de la red "V_{OA}" es evaluada a partir del valor de salida anterior "$N(V_{OA})$", mientras que cuando la señal de reloj "*Clk*" se haya a nivel alto '1' el valor iterado reemplaza "$V_{OA}(t)$" al valor previo de la tensión de salida analógica "$V_{OA}(t-1)$". Finalmente la salida digital del circuito se obtiene interconectando una cadena de tres inversores a la señal analógica de salida, habiéndose ajustado en la fase de diseño la tensión de disparo de los inversores a "*1,59V*" para obtener así una distribución binomial en la cual haya el mismo número de señales a nivel alto y a nivel bajo en un determinado intervalo de tiempo.

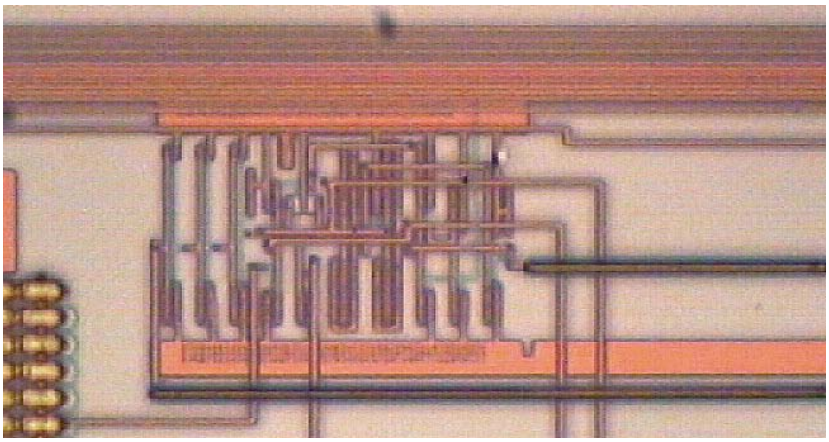

Figura A-3: Fotografía del Layout del prototipo de circuito caótico fabricado

Es de reseñar que el circuito desarrollado se compone en conjunto de 21 transistores *MOS* (aquí se incluyen todos los elementos del circuito), siendo éste capaz de generar tanto señales analógicas como digitales caóticas, que pueden ser usadas para la generación de números aleatorios.

El circuito desarrollado se ha fabricado usando una tecnología comercial de 0.35µm *CMOS*, ocupando todo el circuito un área de 2679µm². La tensión nominal de alimentación del

circuito es “3,3V”. Una fotografía del Layout del prototipo fabricado se presenta en la Figura A-3.

MEDIDAS EXPERIMENTALES

En la Figura A-4 se presenta la función de transferencia obtenida experimentalmente del circuito caótico, una vez ajustadas las tensiones de control de la red neuronal a unos valores específicos. A la vez, en la misma figura se presenta la función identidad ($y(x)=x$) del sistema. Por lo tanto, la salida de la red neuronal implementada será caótica cuando se produzca una intersección entre la función identidad y la función de transferencia de la red $N(x)$, que sucederá cuando $N(x)$ sea decreciente [A-17]. Los valores usados para configurar la red con las tensiones V_{c1} y V_{c2} han sido respectivamente 0V y 0,74V.

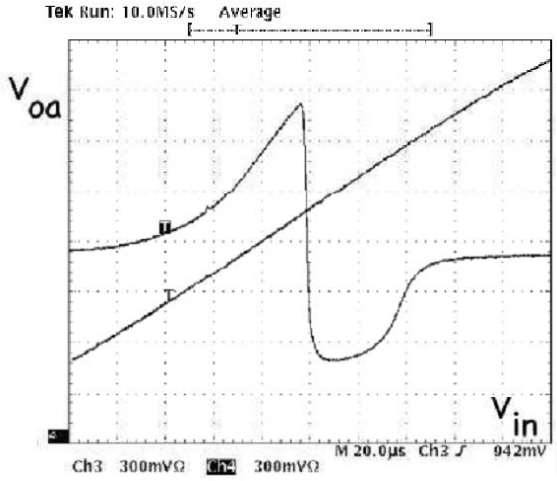

Figura A-4: Función de transferencia del circuito (Voa Vs Vin)

A su vez en la figura A-5 podemos apreciar la dinámica de la red neuronal al aplicarle una señal de reloj externa (cuadrada) de frecuencia 180kHz (Clk), obteniéndose a la salida de la neurona dos señales no periódicas (una digital y la otra analógica). La salida analógica del circuito se corresponde con la señal inferior de la figura A-5 (V_{oa}), mientras que la señal de en medio se corresponde con la salida digital del circuito (V_{od}). En dicha figura también se puede apreciar el nivel de disparo de los inversores (línea negra a trazos) usados para la conversión de la señal de salida analógica en una digital.

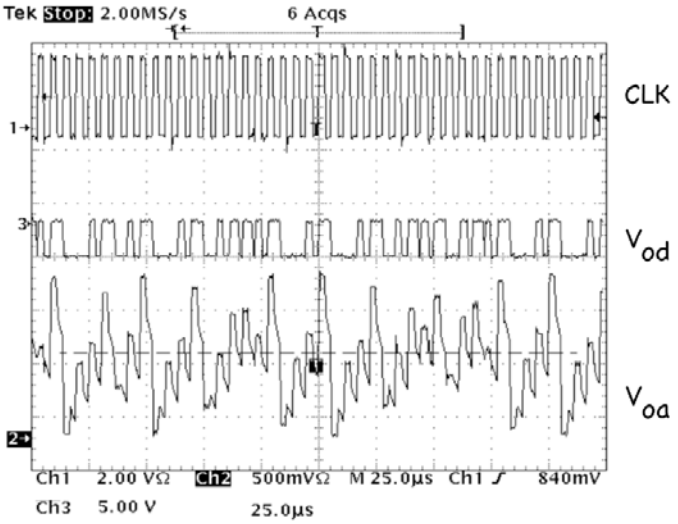

Figura A-5: Respuesta dinámica del circuito para los siguientes parámetros (Vc1=0V y Vc2=0,68V)

A continuación en la figura A-6 se presenta el fenómeno de la bifurcación en cascada por desdoblamiento de período. Dicho diagrama de bifurcación se ha obtenido variando la tensión de control V_{c1} entre 0 y 1.6V y monitorizando la tensión de salida analógica (V_{oa}) resultante de cada iteración de la red neuronal (Ecuación [A-2]). La medida se ha realizado mediante un osciloscopio analógico a fin de mejorar la visualización de la medición, donde cada división del eje de abscisas y ordenadas se corresponde a *200mV*.

Adicionalmente se han computado los valores del exponente de *Lyapunov* (figura A-7) para caracterizar la dinámica del sistema mediante la variación de la tensión de control V_{c2}, puesto que el valor del exponente de *Lyapunov* representa una métrica usada habitualmente para la estimación de cómo un sistema es sensible a las condiciones iniciales. Por lo tanto la existencia de un comportamiento caótico en un sistema está directamente relacionado con la presencia de exponentes positivos, tal y como sucede en nuestro circuito.

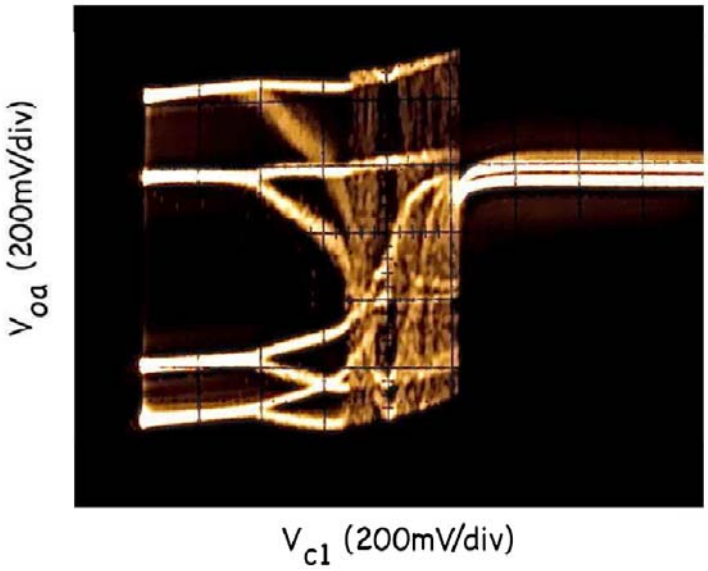

Figura A-6: Diagrama de Bifurcación obtenido variando la tensión de control Vc1 (Voa Vs Vc1)

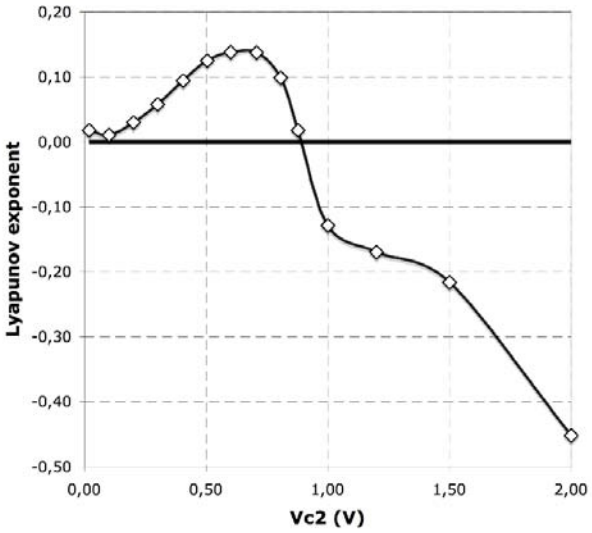

Figura A-7: Valor del exponente de *Lyapunov* en función de la tensión de control Vc2

Por lo tanto, dos trayectorias con condiciones iniciales muy parecidas divergen en promedio a un ritmo exponencial caracterizado por el exponente de *Lyapunov*. Es decir, la separación entre dos trayectorias "δx" presenta una evolución gobernada por la Ecuación [A-3].

$$\left|\delta x_k\right| = \left|\delta x_0\right| e^{\lambda k}$$ Ecuación [A-3]

Siendo "δx_k" la separación entre dos trayectorias en la iteración k-ésima, y "δx_0" la separación inicial. Fijando "$V_{c1} = 0V$" y tomando medidas para diferentes valores de V_{c2}, se ha podido calcular usando la técnica descrita en [A-20] los valores del exponente de *Lyapunov*. Como puede apreciarse en la figura A-7 el valor de "λ" es positivo y por lo tanto el comportamiento del sistema será caótico siempre y cuando se cumpla que la tensión de control sea "$V_{c2} < 0.9V$".

GENERACIÓN DE NÚMEROS ALEATORIOS

La principal aplicación que se le ha dado al circuito caótico descrito en el presente anexo A, ha sido la de proporcionar a los diversos circuitos descritos a lo largo de la presente tesis de un mecanismo real de generación de números aleatorios, a partir de las tramas de bits aleatorios generados en la salida digital de dicho circuito caótico.

Para evaluar la aleatoriedad de las tramas de bits generadas mediante el circuito caótico desarrollado, se han adquirido digitalmente una serie de muestras de 20,000 bits de profundidad mediante una tarjeta de adquisición de datos de *National Instruments* para posteriormente procesarlas mediante unos paquetes de software de test de la aleatoriedad de los datos (específico para aplicaciones criptográficas) [A-21, A-22] que incorpora los tests "*NIST SP 800-22*" y "*FIPS 140-2*". Los resultados de estos tests se presentan en la Tabla. A-1, y como puede apreciarse en dichos resultados el circuito desarrollado genera tramas de bits de muy buena calidad desde el punto de vista de la aleatoriedad.

Tabla A-1: Resultados de los paquetes de test estadístico (FIPS-140-2) y (NIST SP 800-22)

NIST SP 800-22			FIPS 140-2		
Test	**Condición de superación**	**Resultado obtenido @ f=330kHz**	**Test**	**Condición de superación**	**Resultado obtenido @ f=330kHz**
Frequency	>0.01	0.617075	Monobit	9725-10,275	10,113
Block Frequency	>0.01	1.000000	Poker Test	2.16-46.17	24.992000
Cumulative Sums Forward	>0.01	0.857965	Longest runs	1-26	10
Cumulative Sums Backward	>0.01	0.857965	Runs of length 1	2315-2685	2436
Runs	>0.01	0.112444	Runs of length 2	1114-1386	1225

Longest Runs	>0.01	0.111126	Runs of length 3	527-723	640
Rank	>0.01	x	Runs of length 4	240-384	312
Discrete FFT	>0.01	0.516412	Runs of length 5	103-209	158
Overlapping Templates	>0.01	0.516412			
Universal	>0.01	x			
Approximate Entropy	>0.01	x			
Serial	>0.01	0.143192			
Linear Complexity	>0.01	0.361781			
Frequency	>0.01	0.617075			
Block Frequency	>0.01	1.000000			
x: Test no evaluado por el paquete de test estadístico					

ANEXO B: CÓDIGOS ANSI-C

En el presente Anexo se muestran los diferentes códigos ANSI-C desarrollados para las diferentes aplicaciones presentadas a lo largo de la tesis.

EVALUACIÓN SOFTWARE DE LA PROBABILIDAD A POSTERIORI PARA DOS CLASES

El código C aquí presentado se corresponde a la aplicación desarrollada para la evaluación del tiempo de proceso (de una CPU) para la evaluación de la probabilidad a posteriori de una clase, en un sistema de dos clases (aparatado 3.2.4.1.1).

```
////////////////////////////////////////////////////////////////////////////////
// Programa realizado para convertir las contantes MATLAB, a constantes
// inteligibles para las redes neuronales estocásticas, integradas en FPGAs
//
// Progama realizado por: V. Canals, J.LL. Rosselló Y A. Morro
// V1.01
//
////////////////////////////////////////////////////////////////////////////////

#include <string.h>
#include <ctype.h>
#include <stdio.h>     // libreria para el acceso a ficheros
#include <stdlib.h>    // librerai necesaria para poder generar números aleatorios
#include <time.h>      // libreria necesaria para poder usar la función tiempo en la generación de números aleatorios
#include <math.h>      // libreria matemaica del sistema



//
// Rutina simple de determinación de la constantes de un red neuronal
//

void neuron_value(int *p, int *q, double dato)
{
double data=0.0;
double data1, data2=0.0;

//data1=dato/2.03;
//data1=dato/1.86;
```

```
//data1=dato/1.65;
data1=dato/4.0687;

if ((data1>1.0)||(data1<-1.0))
{
  *p=65535;
  data2=(32768.0/data1)+32768.0;
  *q=(int)data2;
}else{
  *q=65535;
  data2=data1*32768.0+32768.0;
  *p=(int)data2;
}
}

//
// El programa principal hace un escombrado de los puntos, para la represntacion de la función de transferencia
//

int main()
{
FILE *fich;                // fichero donde se gruarda la represntación del espacio

int i,j;
int p1, q1=0;
int num_datos=6;
// la sequencia de datdes es W10, W11, b1, W20, b2
//double datos[21]={1.2996,-0.88066,-0.82976,-1.5308,-0.38803};
double datos[21]={1.2996,-0.88066,-0.82976,-1.5308,-0.38803,-0.61074};
double data=0.0;




printf  ("------------------------------------------------------------\n\r");
printf  ("|calculo de los coeficientes de la FPGA              |\n\r");
printf  ("|V1.01                                               |\n\r");
printf  ("------------------------------------------------------------\n\r");
printf  ("\n\r");
printf  ("Factor de conversion: %f\n\r",4.0687);
printf  ("\n\r");
printf  ("la sequencia de datdes es W10, W11, b1, W20, b2 \n\r");
printf  ("\n\r");

for (i=0;i<num_datos;i++)
```

```
{
   p1=0;
   q1=0;
   neuron_value(&p1, &q1, datos[i]);
   printf(" num: %f, p: %d, q: %d\n\r", datos[i], p1, q1);
}
printf("Proceso finalizado.....\n\r");
return(0);
}
// Final del programa principal
```

SIMULACIÓN DE LA RESPUESTA DE UNA NEURONA PULSANTE DIGITAL

El código C aquí presentado se corresponde con la aplicación que desarrollamos para simular el modelo de la neurona pulsante propuesto en el capítulo 4 (aparatado 4.2.7.1) de la presente tesis, antes de proceder a su posterior implementación en una FPGA.

```
////////////////////////////////////////////////////////////////////////////////
// Programa realizado para la simulación de una neurona SNN
// Evaluación de un red neuronal para la realización de un gráfica de la evolución
//
// Programa realizado por: V. Canals, J.LL. Rosselló Y A. Morro
// V1.00
//
////////////////////////////////////////////////////////////////////////////////

#include <string.h>
#include <ctype.h>      // Tipo de ad
#include <stdio.h>     // libreria para el acceso a ficheros
#include <stdlib.h>    // librerai necesaria para poder generar números aleatorios
#include <time.h>      // libreria necesaria para poder usar la función tiempo en la generación de números aleatorios
#include <math.h>      // libreria matemaica del sistema
```

```
int neurona_snn_4_entradas (int in1, int in2, int in3, int in4, int w1, int w2, int w3, int w4, int *state, int *maxstate)
{
int salida=0;
   if (in1==1)
```

```
{
  switch (w1)
  {
   case 1:
        *state=*state+8;
        break;
   case 2:
        *state=*state-8;
        break;
   default:
        *state=*state;
        break;
   }
}

if (in2==1)
{
  switch (w2)
  {
   case 1:
        *state=*state+8;
        break;
   case 2:
        *state=*state-8;
        break;
   default:
        *state=*state;
        break;
   }
}

if (in3==1)
{
  switch (w3)
  {
   case 1:
        *state=*state+8;
        break;
   case 2:
        *state=*state-8;
        break;
   default:
        *state=*state;
        break;
   }
```

```
  }

  if (in4==1)
  {
    switch (w4)
    {
     case 1:
          *state=*state+8;
          break;
     case 2:
          *state=*state-8;
          break;
     default:
          *state=*state;
          break;
     }
  }

    if (*state>0)
    {
       *state=*state-1;
    }else{
        *state=0;
    }

    *maxstate=*state;
    if (*state>20)
    {
      *state=1;
      salida=1;
    }

    if (*state==0)
    {
      salida=0;
    }
    return (salida);
}

//
// Programa principal
//

int main()
{
```

```
int ent1,ent2,ent3,ent4=0;
int número=0;
int out_snn=0;
int i;
FILE *fich;                // fichero de salida de la simulación de la snn
int iteracions=80;
int valor_max=0;


fich = fopen("snn_test.txt", "w");
printf("\n\r");
printf("=============================================================\n\r");
printf("Programa para la simulación de un neurona spiking\n\r");
printf("V.canals, J.LL.Rosello, A. Morro\n\r");
printf("=============================================================\n\r");

// aqui es simila la neurona estocastica

fprintf(fich,"Input, PSP, Output, PSP_MAX\n");
for (i=0;i<iteracions;i++)
{
if (rand()>16384)
{
   ent1=1;
}else{
   ent1=0;
}

out_snn=neurona_snn_4_entradas (ent1, ent2, ent3, ent4, 1, 0, 0, 0, &número, &valor_max);
printf("Input: %d, PSP: %d, Sortida: %d, PSP_max: %d\n\r", ent1, número, out_snn, valor_max);
fprintf(fich,"%d, %d, %d, %d\n", ent1, número, out_snn, valor_max);
}

fclose(fich);
printf("\n\r");
printf("=============================================================\n\r");
printf("Total de puntos calculados por muestra: %f \n\r");
printf("Promedio de %d muestras: %f [ms]\n\r");
printf("=============================================================\n\r");
printf("Proceso finalitzado.....\n\r");
return(0);
}
```

```
// Final programa principal
```

GENERACIÓN DE DOS CLASES BAYESIANAS

El código C que se presenta a continuación se encarga de generar dos poblaciones aleatorias Bayesianas basadas en un método pseudo Montecarlo (Apartado 4.3.3).

```
////////////////////////////////////////////////////////////////////////////////////////
// Programa realizado para generar dos distribuciones de clases, para entrenar una red neuronal
// Se basa en la generación de dos distribuciones bayesianas en el espacio x, y
//
// Los limites de este espacio serán para X=0...65535 y Y=0...65536
// Modificación del programa para matlab
//
// Modificación del programa que permite una compresión del espacio a voluntad
//
//
// Progama realizado por: V. Canals, J.LL. Rosselló Y A. Morro
//
////////////////////////////////////////////////////////////////////////////////////////

#include <string.h>
#include <ctype.h>
#include <stdio.h>     // libreria para el acceso a ficheros
#include <stdlib.h>    // librerai necesaria para poder generar números aleatorios
#include <time.h>      // libreria necesaria para poder usar la función tiempo en la generación de números aleatorios
#include <math.h>      // libreria matemaica del sistema



//#define RAND_MAX 64536

//
// Calcul de la funcio gaussiana bisdimensional
//

float gaussiana(float sigma_x, float sigma_y, float mu_x, float mu_y, int x, int y)
{
float dada=0;
float dato1=0;
float dato2=0;
float dato3, dato4=0;
```

```
    dato1=pow(((float)x-mu_x),2.0);
    dato2=2.0*pow(sigma_x,2);
    dato3=pow(((float)y-mu_y),2.0);
    dato4=2.0*pow(sigma_y,2);
    dada=exp(-1.0*(dato1/dato2))*exp(-1.0*(dato3/dato4));
    return dada;
}


//
// Calcul de la f0
//

float func_f0 (float dat1, float dat2)
{
 float data;
 data=1.0/(dat1*dat2);
 return (data);
}

//
// Calcul de la p0
//

float func_p0 (float p1, float p2)
{
 float data;
 data=1.0-p1-p2;
 return (data);
}

//
// Calculo de la probabilidad a posteriori de una clase,en un sistema con tres clases
//
// Con el indice de la clase se indica de que calse de la tres se realiza la posteriori
//

float func_post (int clase, float f0, float p0, float f1, float p1, float f2, float p2)
{
float data=0;

switch (clase)
{
     case 1:
```

```
            data=(f1*p1)/(f0*p0+f1*p1+f2*p2);
            break;
      case 2:
            data=(f2*p2)/(f0*p0+f1*p1+f2*p2);
            break;
      default:
            data=0;
            break;
}
return (data);
}


//
// Programa principal
//

int main(){

// definicio de les variables
float gaussian1[64][64];
float gaussian2[64][64];
float threshold1=1000;
float threshold2=1000;
int clase1[64][64];
int clase2[64][64];
FILE *fich;                // fichero de salida de la clase 1
FILE *fich1;               // fichero de salida de la clase 2
FILE *fich2;               // fichero de salida de combinado con las 2 clases
FILE *fich3;               // fichero de salida de la distribución GAUSSIANA 1
FILE *fich4;               // fichero de salida de la distribución GAUSSIANA 2
FILE *fich5;               // fichero entrenamiento matlab x (input)
FILE *fich6;               // fichero entrenamiento matlab y (input)
FILE *fich7;               // fichero entrenamiento matlab z (output)


// constants de les distribucions gaussianes

// distribucio 1
float mu_x_1=46000.0;
float mu_y_1=45000.0;
float sigma_x_1=10000.0;
float sigma_y_1=7000.0;

// distribucio 2
```

```
float mu_x_2=20000.0;
float mu_y_2=25000.0;
float sigma_x_2=15000.0;
float sigma_y_2=7500.0;



//
// Parametres de al funcio de distribucio bayesiana
//
//

float p_c1=0.1;  // Probabilidad de la clase 1
float p_c2=0.1;  // Probabilidad de la clase 2
float p_c0=0;    // Probabilidad de la clase 0
float f_c1=0;    // Probabilidad a priori P(x/c1)
float f_c2=0;    // Probabilidad a priori P(x/c2)
float f_c0=0;    // Probabilidad a priori P(x/c0)

float post_c1=0;
float post_c2=0;

//
// Variables d'us general
//

int i,j,k=0;
int x,y=0;
float z;

//
// Variables de reescalat del espai de les distribucions
//

float xmin, xmax, ymin, ymax=0.0;
float xx, yy=0.0;
float deltax, deltay=0.0;



//
// Parametres funcionals
//

int num_tirades=1000;    // Nombre de números que es tiren en el sistema a fi de determinar si son de una clase o altre
```

```
//
// Inici del programa principal
//

printf  ("-------------------------------------------------------------\n\r");
printf  ("|Inicio del proceso de geneación de las distribuciones   |\n\r");
printf  ("|para las cales de las redes neuronales                  |\n\r");
printf  ("| Distribuciones bayesisanas V2.0                        |\n\r");
printf  ("-------------------------------------------------------------\n\r");

fich = fopen("clase1_bayes.txt", "w");
fich1 = fopen("clase2_bayes.txt", "w");
fich2 = fopen("combinada_bayes.txt", "w");
fich5= fopen("clase_p.mat", "w");
fich6= fopen("clase_q.mat", "w");
fich7= fopen("clase_z.mat", "w");

//fprintf(fich5,"P = [");
//fprintf(fich6,"Q = [");
//fprintf(fich7,"Z = [");

//
// calculo de los reescalados del espacio
//
xmin=0.0;
xmax=10.0;
ymin=0.0;
ymax=10.0;
deltax=xmax-xmin;
deltay=ymax-ymin;


srand(time(0));     // Cabiamos la semilla
for (i=0;i<num_tirades;i++)
{
   // Determinam un x, y aleatories al nostre espai
   x=2*rand();         // generamos el número aleatorio
   //srand(time(0));
   y=2*rand();
   p_c0=func_p0(p_c1,p_c2);
   f_c0=func_f0(0.02,0.01);         // amb aquesta probabilitat es control a el gap entre distribucions
   f_c1=gaussiana(sigma_x_1, sigma_y_1, mu_x_1, mu_y_1, x, y)*65536;
   f_c2=gaussiana(sigma_x_2, sigma_y_2, mu_x_2, mu_y_2, x, y)*65536;
   post_c1=func_post(1, f_c0, p_c0, f_c1, p_c1, f_c2, p_c2)*65536;
   post_c2=func_post(2, f_c0, p_c0, f_c1, p_c1, f_c2, p_c2)*65536;
```

```
//post_c0=65536-post_c1-post_c2;
// Ara obtenim el númeroa aleatori z
//fprintf(fich," %d, %d, %f, %f, %f, %f \n", x,y,z,f_c0,post_c1,post_c2);


xx=((float)x/65536)*deltax+xmin;
yy=((float)y/65536)*deltay+ymin;
z=2.0*(float)rand();

if ((z<post_c1))
{

      fprintf(fich," %d, %d\n", x,y);
      fprintf(fich2," %d, %d\n", x,y);
      fprintf(fich5,"%f %f;\n", xx, yy);
      fprintf(fich6,"%d ", y);
      fprintf(fich7,"%d ", 0);
         if (i<num_tirades-1)
         {
               //fprintf(fich5,";\n");
               //fprintf(fich6,";\n");
               //fprintf(fich7,";\n");
          }else{
               //fprintf(fich5,"]\n");
               //fprintf(fich6,"]\n");
               //fprintf(fich7,"]\n");
          }
}
if ((z>post_c1)&&(z<post_c1+post_c2))
{
      fprintf(fich1," %d, %d\n", x,y);
      fprintf(fich2," %d, %d\n", x,y);
      fprintf(fich5,"%f %f;\n", xx, yy);
      fprintf(fich6,"%d ", y);
      fprintf(fich7,"%d ", 1);
         if (i<num_tirades-1)
         {
               //fprintf(fich5,";\n");
               //fprintf(fich6,";\n");
               //fprintf(fich7,";\n");
          }else{
               //fprintf(fich5,"]\n");
               //fprintf(fich6,"]\n");
               //fprintf(fich7,"]\n");
          }
```

```
   }

   if (i==(num_tirades-1))
   {

//      fprintf(fich5,"]\n");
//      fprintf(fich6,"]\n");
//      fprintf(fich7,"]\n");
    }else{

    }

}
fclose(fich);
fclose(fich1);
fclose(fich2);
fclose(fich5);
fclose(fich6);
fclose(fich7);
printf("\n\r");
printf("Proceso finalitzado.....\n\r");
return(0);
}

// Final programa principal
```

SIMULACIÓN DE UNA RED NEURONAL 2-5-1

El código C aquí presentado se corresponde con la aplicación desarrollada para comprobar la correcta operación de la red neuronal a implementar en la FPGA a partir de los parámetros de red obtenidos mediante MATLAB (Apartado 4.3.3).

```
////////////////////////////////////////////////////////////////////////////////////////
// Programa realizado para la simulacion de un red neuronal
// Evaluación de un red neuronal y determinación de los contornos entre clases
//
// Progama realizado por: V. Canals, J.LL. Rosselló Y A. Morro
// V3.00
//
////////////////////////////////////////////////////////////////////////////////////////
```

```
#include <string.h>
#include <ctype.h>
#include <stdio.h>     // libreria para el acceso a ficheros
#include <stdlib.h>    // librerai necesaria para poder generar números aleatorios
#include <time.h>      // libreria necesaria para poder usar la función tiempo en la generación de números aleatorios
#include <math.h>      // libreria matemaica del sistema


//
// Rutina basica de la red neuronal, 2 entradas
//

double neuron_2_inputs_lineal(double x, double y, double w10, double w11, double b1)
{
  double dato=0;
  double data=0;
  dato=w10*x+w11*y+b1;
  data=dato;
  return (data);
}

//
// Rutina basica de la red neuronal, 5 entradas y una salida
//

double neuron_5_inputs(double x1, double x2, double x3, double x4, double x5, double w10, double w11, double w12, double w13,
double w14, double b1)
{
  double dato=0;
  double data=0;
  dato=w10*x1+w11*x2+w12*x3+w13*x4+w14*x5+b1;
  data=tanh(dato);
  return (data);
}


//
// El programa principal hace un escombrado de los puntos, para la represntacion de la función de transferencia
//

int main()
{

// definicion de las variables del proceso
```

```
FILE *fich, *fich1;                 // fichero donde se gruarda la represntación del espacio

// hay que definir

int num_neurons_in_layer=5;  // 5 neuronas a la entrada de la red neuronal
int num_inputs_neurons_in_layer=2;  // las neuronas tiene dos entradas
int num_neurons_out_layer=1;        // 1 neurona de salida del sistema
int num_inputs_neurons_out_layer=5; // número de entradas de la neurona de salidad del sistema

int n_mesures_x=50;
int n_mesures_y=50;
int t_mesures=0;                    // número total de mesures a fer

/*
double out_layer1[5]={0.0};
double w10_layer1[5]={-139.2683,-1.1643,-3.3161,5.9889,-18.132};
double w11_layer1[5]={-52.3172,2.088,20.1259,-11.0988,-0.68521};
double b1[5]={33.5689, -0.57637, 14.6269, 9.362,-16.2913};
double w20[5]={48.7604, -9.2694, -50.4493, -6.8926, 14.5399};
double b2=14.8196;
*/
double out_layer1[5]={0.0};
double w10_layer1[5]={-0.58052,0.34215,0.42128,-0.56901,-0.69342};
double w11_layer1[5]={-1.1766,1.2502,0.51804,1.2361,-0.10935};
double b1[5]={-0.081067,-0.31625,0.41073,0.90734,0.99171};
double w20[5]={1.2996,-0.88066,-0.82976,-1.5308,-0.38803};
double b2=-0.61074;

double out_net[50][50]={0,0};      // array doinde se guardaran los datos de la matriz de resultados
double x_equiv, y_equiv=0.0;       // posiciones equivalentes del sistema
double x_factor, y_factor=0.0;     // factor medida esuiespaciada en el espacio
double anterior=0;



// Variables de operacion

signed int i,j,k;                   //indices generales del sistema

// variables para determinar el tiempo de proceso del sistema

clock_t comienzo;
clock_t t_total;

printf  ("------------------------------------------------------------\n\r");
printf  ("| Simulacion de la red neuronal, paper ANN stoch Signed  |\n\r");
```

```
printf  ("| Xarxa 5 Neurones Lineals - 1 tangent sigmoidal (OUT)   |\n\r");
printf  ("| V3.00                                                 |\n\r");
printf  ("| V.Canals, J.LL Rossello, A. Morro                     |\n\r");
printf  ("--------------------------------------------------------\n\r");
printf  ("\n\r");


comienzo=clock();                   // tiempo del procesador al inicio del progama
x_factor=65535.0/(double)n_mesures_x;
y_factor=65535.0/(double)n_mesures_y;


for (i=0;i<n_mesures_x;i++)
{
   x_equiv=(((double)i*x_factor)-32768.0)/32768;
   for (j=n_mesures_y-1;j>=0;j--)
   {
      y_equiv=(((double)j*y_factor)-32768.0)/32768;
      for (k=0;k<num_neurons_in_layer;k++)
      {
          out_layer1[k]= neuron_2_inputs_lineal(x_equiv, y_equiv, w10_layer1[k], w11_layer1[k], b1[k]);
      }
      out_net[i][j]=neuron_5_inputs(out_layer1[0], out_layer1[1], out_layer1[2], out_layer1[3], out_layer1[4], w20[0], w20[1], w20[2],
w20[3], w20[4], b2);
   }
}

//t_total=(clock()-comienzo);         // Tiempo e proceso del sistema
//printf("Tiempo de proceso del sistema: %d [ms], Iteraciones: %u \n\r", t_total,n_mesures_x*n_mesures_y);

// proceso de escritura de las medidas en un fichero de texto
fich = fopen("ann_clases_th_v3.txt", "w");
for (i=0;i<n_mesures_x;i++)
{
   for (j=0;j<n_mesures_y;j++)
   {
      fprintf(fich,"%f,",out_net[i][j]);
   }
   fprintf(fich,"\n");
}

//
// Fichero, generador de la curva de deteccion de zonas
//
```

```
fich1 = fopen("curva_clases_v3.txt", "w");
anterior=0;
for (i=0;i<n_mesures_x;i++)
{
   x_equiv=(((double)i*x_factor)-32768.0)/32768;
   anterior=out_net[i][0];
   for (j=0;j<n_mesures_y;j++)
   {
      y_equiv=(((double)j*y_factor)-32768.0)/32768;
      if ( ((out_net[i][j]>=0)&&(anterior<0))||((out_net[i][j]<0)&&(anterior>=0)) )
      {
        fprintf(fich1,"%f, %f\n",x_equiv,y_equiv);
      }
      anterior=out_net[i][j];
   }
}

t_total=(clock()-comienzo);        // Tiempo e proceso del sistema
printf("Tiempo de proceso del sistema: %d [ms], Iteraciones: %u \n\r", t_total,n_mesures_x*n_mesures_y);

fclose(fich);
fclose(fich1);
printf("\n\r");
printf("Proceso finalitzado.....\n\r");
return(0);
}

// Final programa principal
```

APLICACIÓN PARA EL REAJUSTE DE LOS PARÁMETROS DE UNA RED NEURONAL EN LA CODIFICACIÓN (ESL)

En este subapartado se presenta el código C desarrollado para la conversión y reajuste de los pesos obtenidos mediante MATLAB para su directa implementación en una FPGA dotada de conversores P2B(N) de 16-bits.

```
////////////////////////////////////////////////////////////////////////////////////////
// Programa realizado para convertir las contantes MATLAB, a constantes
// Inteligibles para las redes neuronales estocásticas, integradas en FPGAs
```

```
//
// Progama realizado por: V. Canals, J.LL. Rosselló Y A. Morro
// V1.01
//
////////////////////////////////////////////////////////////////////////////////

#include <string.h>
#include <ctype.h>
#include <stdio.h>     // libreria para el acceso a ficheros
#include <stdlib.h>    // librerai necesaria para poder generar números aleatorios
#include <time.h>      // libreria necesaria para poder usar la función tiempo en la generación de números aleatorios
#include <math.h>      // libreria matemaica del sistema


//
// Rutina simple de determinación de la constantes de un red neuronal
//

void neuron_value(int *p, int *q, double dato)
{
double data=0.0;
double data1, data2=0.0;

//data1=dato/2.03;
//data1=dato/1.86;
//data1=dato/1.65;
data1=dato/4.0687;

if ((data1>1.0)||(data1<-1.0))
{
  *p=65535;
  data2=(32768.0/data1)+32768.0;
  *q=(int)data2;
}else{
  *q=65535;
  data2=data1*32768.0+32768.0;
  *p=(int)data2;
}
}

//
// El programa principal hace un escombrado de los puntos, para la represntacion de la función de transferencia
//

int main()
```

```
{
FILE *fich;                 // fichero donde se gruarda la represntación del espacio

int i,j;
int p1, q1=0;
int num_datos=6;
// la sequencia de datdes es W10, W11, b1, W20, b2
//double datos[21]={1.2996,-0.88066,-0.82976,-1.5308,-0.38803};
double datos[21]={1.2996,-0.88066,-0.82976,-1.5308,-0.38803,-0.61074};
double data=0.0;




printf  ("------------------------------------------------------------\n\r");
printf  ("|calculo de los coeficientes de la FPGA                |\n\r");
printf  ("|V1.01                                                 |\n\r");
printf  ("------------------------------------------------------------\n\r");
printf  ("\n\r");
printf  ("Factor de conversion: %f\n\r",4.0687);
printf  ("\n\r");
printf  ("la sequencia de datdes es W10, W11, b1, W20, b2 \n\r");
printf  ("\n\r");

for (i=0;i<num_datos;i++)
{
   p1=0;
   q1=0;
   neuron_value(&p1, &q1, datos[i]);
   printf(" num: %f, p: %d, q: %d\n\r", datos[i], p1, q1);
}
printf("Proceso finalizado.....\n\r");
return(0);
}
// Final del programa principal
```