UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INGENIERÍA

Departamento de Electrotecnia

REDES NEURONALES ANALÓGICAS PARA LA COMPUTACIÓN PARALELA MASIVA DE SEÑALES EN TIEMPO REAL

Gustavo Indalecio Eugenio CANCELO

Tesis presentada para obtener el grado de
DOCTOR EN INGENIERÍA

Director: Carlos Frede Christiansen

Co-director: Miguel Angel Mayosky

La Plata, octubre de 1996

A mis padres

A Silvia, Juliana y Martina

AGRADECIMIENTOS

Deseo agradecer profundamente a mis directores Carlos Christiansen y Miguel Mayosky por su colaboración, paciencia y sabios consejos que hicieron posible la elaboración de esta Tesis

Desde el año 1985, en el cual comence mi primera Beca, mis pasos fueron guiados por el inolvidable profesor y amigo Jose Maria Catalfo, a quien no solo debo gran parte de mi formación profesional sino tambien humana, dentro del ámbito universitario.

Quiero, tambien, expresar mi agradecimiento a todos mis compañeros del Leici, que conforman un grupo donde se mezcla amistad, buen compañerismo y apoyo profesional desinteresado, en el cual se disfruta el trabajo diario.

Deseo agradecer a mi esposa Silvia por su constante apoyo y cariño, los cuales han sido fundamentales para que este trabajo llegara a buen fin.

Agradezco a las instituciones que apoyaron mi trabajo de investigación: La Facultad de Ingeniería de la Universidad Nacional de La Plata y el Consejo Nacional de Investigaciones Científicas y Tecnicas (CONICET).

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN

1.1 INTRODUCCIÓN AL TEMA Y MOTIVACIÓN 1
2.2 ORGANIZACIÓN DE LA TESIS 5

CAPÍTULO 2: REDES NEURONALES ARTIFICIALES

2.1 REDES NEURONALES ARTIFICIALES Y COMPLEJIDAD 7
2.2 ORGANIZACIÓN Y PROPÓSITO DEL CAPITULO 10
2.3 CLASES DIFERENTES DE MODELOS NEURONALES 10
2.4 MODELOS NEURONALES ESTÁTICOS 13
2.5 EL PERCEPTRÓN SIMPLE 14
2.5.1 ALGORITMOS DE APRENDIZAJE EN PERCEPTRON SIMPLES 16
2.5.1.1 ALGORITMOS LINEALES 17
2.5.1.2 ALGORITMOS NO LINEALES 18
2.5.1.3 ALGORITMOS DE GRADIENTE
2.5.1.3.1 CONVERGENCIA DEL ALGORITMO DE GRADIENTE 20
2.5.1.4 FUNCIONES DE ACTIVACIÓN NO LINEALES 22
2.6 PERCEPTRONES MULTICAPA 24
2.6.1 APRENDIZAJE EN REDES MULTICAPA 26
2.6.2 PROBLEMAS Y LIMITACIONES DE LOS MLP 30
2.6.3 GENERALIZACIÓN 31
2.6.3.1 GENERALIZACIÓN PROMEDIO 32
2.6.3.2 GENERALIZACIÓN DEL CASO MAS DESFAVORABLE 35
2.6.4 LIMITE EN EL NUMERO DE UNIDADES OCULTAS EN MLPs 35
2.7 APROXIMACIÓN Y APRENDIZAJE 39

2.8 REDES NEURONALES BASADAS EN FUNCIONES DE BASE RADIAL GAUSSIANAS 44
2.8.1 DIMENSIONAMIENTO Y ELECCIÓN DE LOS PARÁMETROS DE LA RED 45

CAPÍTULO 3: NEUROCOMPUTACIÓN

3.1 INTRODUCCIÓN 53
3.2 DISPOSITIVOS Y SISTEMAS NEURONALES 55
3.3 ANALÓGICO vs. DIGITAL 56
3.3.1 NEURONAS ANALÓGICAS 57
3.3.1.1. ARQUITECTURA DE NEURONAS ANALÓGICAS 58
3.3.2 NEURONAS DIGITALES 60
3.3.3 NEURONAS MIXTAS ANALÓGICO-DIGITALES 62
3.3.3.1 PESOS ANALÓGICOS EN EL SISTEMA DE CODIFICACIÓN DE PULSOS 64
3.3.4 COMPARACIÓN ENTRE NEURONAS ANALÓGICAS, DIGITALES E HÍBRIDAS: ANÁLISIS Y DISCUSIÓN 65
3.3.4.1 FLEXIBILIDAD: DISPOSITIVOS DE PROPÓSITOS GENERALES Y DISPOSITIVOS ORIENTADOS A APLICACIONES ESPECIFICAS 68
3.3.4.2 EL PROBLEMA DEL ALMACENAMIENTO DE LOS COEFICIENTES DE PESO 70
3.4 DISPOSITIVOS NEURONALES: ESTADO DEL ARTE 72
3.4.1 ANNA (Analog Neural Network Arithmetic and logic unit) 73
3.4.2 CNAPS (Connected Network of adaptive Processors) 73
3.4.3 ETANN (Electrically Trainable Analog Neural Network) 74
3.5 NEUROCOMPUTADORAS 76
3.5.1 GENES: Arreglo sistólico bidimensional 78
3.5.2 PROCESADOR NEURONAL PROGRAMABLE 80
3.5.3 RETINA ARTIFICIAL 81
3.6 DISCUSIÓN 82

CAPÍTULO 4: ELECTRÓNICA ANALÓGICA PARA REDES NEURONALES ARTIFICIALES

4.1 INTRODUCCIÓN 85
4.2 MODELO DE UNA NEURONA 85
4.3 ELECTRÓNICA ANALÓGICA PARA REDES NEURONALES ARTIFICIALES. 88
4.4 VLSI EN REDES NEURONALES 90
4.4.1 IMPLEMENTACIÓN EN REDES ANALÓGICAS 90
4.4.2 LIMITES DE OPERACIÓN 95
4.4.3 EL PROBLEMA DE LA TENSIÓN DE SALIDA 98
4.4.4 AMPLIFICADORES Y MULTIPLICADORES DE AMPLIO RANGO 99
4.4.5 FUNCIONES NO LINEALES 101
4.4.5.1 SIGMOIDE DE SALIDA CON GANANCIA VARIABLE 102
4.5 IMPLEMENTACIÓN DE UNA NEURONA RBF GAUSSIANA 103
4.6 CONSIDERACIONES SOBRE EL DISEÑO DE LA CELDA NEURONAL RBF GAUSSIANA 110

CAPÍTULO 5: SISTEMA NEURONAL ANALÓGICO NETMAP

5.1 INTRODUCCIÓN 113
5.2 DISEÑO DE UNA NEUROCOMPUTADORA 116
5.3 LA RED NEURONAL ANALÓGICA 117
5.4 ANCHO DE BANDA DE PROCESAMIENTO EN EL SISTEMA NETMAP 119
5.5 APRENDIZAJE EN EL SISTEMA NETMAP 121
5.6 OPTIMIZACIÓN DE LA ARQUITECTURA DEL SISTEMA NETMAP 122
5.6.1 RELACIÓN ENTRE LAS FUNCIONES DE UNA NEUROCOMPUTADORA Y EL SISTEMA NETMAP 124
5.7 LECTURA DE LOS PESOS SINÁPTICOS 125
5.8 MODIFICACIÓN DE PESOS EN LA RED NEURONAL 128

5.9 ORGANIZACIÓN DEL HARDWARE DE LA NEUROCOMPUTADORA 130
5.9.1 CONVERSIÓN DE PATRONES DE ENTRADA Y DE SALIDA DE LA RED NEURONAL. 133
5.10 DISCUSIÓN 135

CAPÍTULO 6: FUNCIONES DE PROGRAMACIÓN DEL SISTEMA NEURONAL NETMAP

6.1 INTRODUCCIÓN 136
6.2 ENTRENAMIENTO DE LA RED NEURONAL EN EL SISTEMA NETMAP 137
6.2.1 RED NEURONAL DENTRO DEL LAZO DE APRENDIZAJE 138
6.3 SIMULACIÓN DEL MODELO DE LA MEMORIA ANALÓGICA 140
6.3.1 GENERACIÓN DE LAS TABLAS DE VPP, TP Y B. 142
6.4 SIMULADOR DE REDES NEURONALES 144
6.5 FUNCIONES DE MEDICIÓN Y PROGRAMACIÓN EN BAJO NIVEL 145
6.6 PROGRAMACIÓN DEL PROCESADOR DIGITAL DE SEÑALES 147
6.6.1 CANAL DE PATRONES DE ENTRADA 148
6.6.2 COMUNICACIÓN ENTRE PROCESADORES 149
6.7 INTERFAZ ENTRE EL SISTEMA NETMAP Y EL USUARIO 150
6.7.1 INTERFAZ CON EL USUARIO 151
6.7.1.1 MÓDULO MANIPULADOR DE ARCHIVOS 153
6.7.1.1.1 FORMATO DEL ARCHIVO DE ESPECIFICACION DE LA RED NEURONAL 154
6.7.1.1.2 FUNCIONES DEL MODULO DOR DE ARCHIVOS 156
6.8 DISCUSIÓN 157

CAPÍTULO 7: CONCLUSIONES Y DESARROLLOS FUTUROS EN TEMAS RELACIONADOS CON LA TESIS

7.1 CONCLUSIONES 158

APÉNDICE A: EFECTO TÚNEL FOWLER-NORDHEIM

A.1. EFECTO TÚNEL EN DISPOSITIVOS DE COMPUERTA FLOTANTE A.1
A.2 MÉTODOS QUE USAN AISLANTES DE OSi GRUESOS A.2
A.3 MODELOS QUE USAN AISLANTES DE OSi DELGADOS A.3
A.3.1 MODELO DE UNA CELDA MOS CON COMPUERTA FLOTANTE A.7
A.3.2 CALCULO DE V_{tun} A.7
A.3.3 CALCULO DE LA TENSIÓN UMBRAL Vt A.9
A.3.4 DINÁMICA DE Vt DURANTE LA PROGRAMACIÓN/BORRADO A.10

APÉNDICE B: MODELOS DEL TRANSISTOR MOS

B.1 MODELOS DEL TRANSISTOR MOS B.1
B.2 BALANCE DE POTENCIALES Y CARGAS B.2
B.3 OPERACIÓN EN LA ZONA DE DEPLESIÓN B.6
B.4 FLUJO DE CORRIENTE POR DIFUSIÓN B.7
B.5 CONDUCTANCIA DE DRENADOR Y EFECTO EARLY B.8
B.6 EFECTO DEL SUSTRATO B.9

APÉNDICE C: DISPOSITIVO NEURONAL ETANN

C.1 DISPOSITIVO NEURONAL PARA EL PROCESAMIENTO DE PATRONES C.1
C.2 CONFIGURACIÓN INTERNA Y TEORÍA DE FUNCIONAMIENTO DEL ETANN C.2
C.3 ARQUITECTURA DE UNA SINAPSIS C.3
C.4 RESPUESTA SIGMOIDEA C.7

C.5 OPERACIÓN DEL ETANN EN MODO DE PROCESAMIENTO PARALELO C.9

APÉNDICE D CIRCUITOS DEL SISTEMA NETMAP

D.1 PLACA DSPCARD D.1
D.2 MAPAS DE MEMORIA D.4
D.3 MEMORIA COMPARTIDA D.6
D.4 MAPA DE DISPOSITIVOS DE ENTRADA/SALIDA D.7
D.5 PLACA PBOX D.8
D.5.1 DOBLADOR DE TENSIÓN D.9
D.5.2 CIRCUITO DE PROGRAMACIÓN DE COEFICIENTES DE PESO D.10
D.5.3 CIRCUITO DE CONVERSIÓN DE PATRONES DE ENTRADA D.11
D.5.4 CONVERSIÓN DE PATRONES DE SALIDA Y MEMORIA DE ALMACENAMIENTO TEMPORARIO D.12
D.5.5 CIRCUITO DE MEDICIÓN PARA LA LECTURA DE PESOS D.14

REFERENCIAS R.1

ÍNDICE DE FIGURAS

Figura 2.1	a) RNA estáticas b) RNA recurrentes	11
Figura 2.2.	a) Aprendizaje supervisado b) aprendizaje no supervisado	12
Figura 2.3	Modelo Neuronal *feedforward* de n-capas	14
Figura 2.4	Perceptrón simple	15
Figura 2.5	Funciones de activación	15
Figura 2.6	Problema del XOR a) unidad lineal b) preprocesamiento cuadratico	16
Figura 2.7	Distancia al plano $w^T . x = 0$	19
Figura 2.8	Perceptrón multicapa	25
Figura 2.9	Generación de los δ en el algoritmo de retropropagación	29
Figura 2.10	Red neuronal con una capa oculta	35
Figura 2.1	a) celdas de una RNA de dos capas. b) disposición de las celdas en un hipercubo.	36
Figura 2.12	Medida de la capacidad en función de la relación m/2n+1	38
Figura 2.13	Red Neuronal Gaussiana de dos capas	44
Figura 2.14	Mapa bidimensional con obstáculos, grilla de tamaño fijo	48
Figura 2.15	Esqueleto de distancias máximas a los obstáculos	49
Figura 2.16	Trayectoria óptima	51
Figura 3.1	Matriz sináptica	59
Figura 3.2	Multiplicador de Bult-Wallinga	59
Figura 3.3	Unidad de procesamiento neuronal digital	61
Figura 3.4	Sistema neuronal digital	62
Figura 3.5	Codificación de pulsos: a) Neurona, b) Sinapsis	62
Figura 3.6	Sumador	63
Figura 3.7	Modulador de pesos sinápticos	63
Figura 3.8	Memoria analógica en el sistema de codificación de pulsos	64
Figura 3.9	Generación del pulso en función del valor almacenado en memoria	64
Figura 3.10	Variación de Vt en función de la carga en la compuerta flotante	71
Figura 3.11	Diagrama en bloques	75

Figura 3.12	Diagrama de cantidad de nodos en función de la complejidad	77
Figura 3.13	Arreglo sistólico a) arquitectura bidimensional b) esquema interno	79
Figura 3.14	Algoritmo de ruta para la actualización de los pesos	80
Figura 3.15	Sistema de resistencias conmutadas a) Esquema básico b) Dispositivo de 8 canales	81
Figura 3.16	Retina artificial	82
Figura 4.1	Modelo eléctrico equivalente del funcionamiento de la membrana	87
Figura 4.2	Curva temporal de depolarización de la membrana	88
Figura 4.3	Amplificador de transconductancia	91
Figura 4.4	Curvas de corriente en el amplificador de transconductancia	92
Figura 4.5	Multiplicador analógico de cuatro cuadrantes	93
Figura 4.6	Curvas de corriente del multiplicador de cuatro cuadrantes	95
Figura 4.7	Corriente de salida vs. tensión de salida	99
Figura 4.8	Amplificador de amplio rango de tensión de salida	99
Figura 4.9	Multiplicador de cuatro cuadrantes y amplio rango de V_{out}	100
Figura 4.10	Diagrama en bloques	104
Figura 4.11	Función módulo de una tensión diferencial de entrada	105
Figura 4.12	I_{out} del bloque que calcula la función módulo de V_d	106
Figura 4.13	Función cuadrática en base a logaritmo y exponencial	106
Figura 4.14	Corriente de salida del bloque cuadrático	107
Figura 4.15	Bloque exponencial	108
Figura 4.16	Salida gaussiana de corriente usando I_b como parámetro	109
Figura 4.17	Circuito completo de una neurona gaussiana de una entrada unidimensional	109
Figura 4.18	Circuito completo de una neurona gaussiana con entrada n_dimensional	110
Figura 4.19	Variación de la corriente de salida en función de I	112

Figura 5.1	Sistemas neuronales: a) sistema adaptivo para procesamiento de patrones. b) control por modelo de referencia.	113
Figura 5.2	Sistema neuronal funcionando como acelerador de una computadora secuencial	114
Figura 5.3	Organización del sistema Netmap	115
Figura 5.4	Arreglos paralelos: a) en varias capas b) bus común	116
Figura 5.5	Estructura interna de la unidad de procesamiento	118
Figura 5.6	Esquema de la sinapsis	118
Figura 5.7	Canal de procesamiento de patrones	120
Figura 5.8	Rendimiento unitario en función de la cantidad de procesadores paralelos	120
Figura 5.9	Estructura jeráquica en el sistema Netmap	122
Figura 5.10	Sistema para medida de VTOwt/ref	127
Figura 5.11	Modelo de la compuerta flotante en programación	128
Figura 5.12	Tiempos de programación de un coeficiente sináptico	129
Figura 5.13	Diagrama en bloques de la placa DSPcard	130
Figura 5.14	Diagrama en bloques PBox	133
Figura 5.15	Diagrama de tiempos del microcontrolador, A/D y FIFO	134
Figura 6.1	Algoritmo de programación de pesos	139
Figura 6.2	Algoritmo de programación	141
Figura 6.3	Curvas de V_{PP2} vs. ΔV_{TOwg}	143
Figura 6.4	Convergencia del algoritmo de programación de pesos	144
Figura 6.5	Paralelización en la transferencia de patrones	149
Figura 6.6	Ejemplo interfaz usuario	151
Tabla 6.1	Area de comandos	150
Tabla 6.2	Funciones del módulo de interfaz con el usuario	153
Tabla 6.3	Formato de archivo de especificación de una red neuronal	155
Tabla 6.4	Funciones del manipulador de archivos	156

CAPITULO 1

INTRODUCCIÓN

Durante más de 40 años las últimas generaciones humanas han visto como la computadora, en sus más variadas formas, ha ido ocupando espacios de la vida cotidiana. Muchas aplicaciones son visibles en forma directa, tales como la calculadora o computadoras personales, los cajeros automáticos, etc. Tantas otras están más ocultas aunque no por eso son menos importantes e incluyen aplicaciones a instrumental médico, supervisión y seguridad de automóviles, comunicación digital, etc. Además, el uso masivo de las computadoras, controladores inteligentes y robots ha permitido que la producción industrial bajara sus costos, haciendo que productos tradicionalmente caros sean ahora de consumo masivo.

La intención aquí no es hacer un análisis sociológico sobre la mejora de la calidad de vida con la computadora, sino, someramente, justificar el gran interés de la comunidad científica del área por hacer computadoras más potentes, versátiles y económicas.

El desarrollo de las ciencias de la computación ha seguido la linea arquitectónica del matemático John von Neumann que en 1952 diseñó una estructura basada en un procesamiento secuencial de datos e instrucciones. Esta estructura sigue rigurosamente un programa secuencial almacenado en memoria. La arquitectura de von Neumann se basa en la lógica de procedimientos que comúnmente utilizamos, hallando soluciones parciales a un problema, y que luego son utilizadas para lograr la solución final.

Contrariamente, una visión del cerebro a distancias microscópicas no se condice con estructuras tipo von Neumann ni con programas almacenados. En cambio, existe una computación masiva de unidades concurrentes y redundancia de tareas y conexiones que proporcionan robustez. El cerebro está especialmente orientado al procesamiento de información sensorial compleja y ruidosa. El ejemplo clásico y más ilustrativo es el de la visión. En dicho

proceso intervienen más de 10 mil millones de neuronas. Ninguna computadora secuencial es capaz de alcanzar esta potencia de cómputo. Además, desde el punto de vista algoritmico, no existe ningún sistema artificial que pueda emular esta función con resultados similares. En otras palabras, el cerebro posee cualidades indiscutibles para la realización de tareas cognitivas, algoritmicamente complejas y con señalización ruidosa.

Si bien el futuro de las computadoras tradicionales parece cada vez más promisorio y complementario de las habilidades humanas, los sistemas neuronales han atraído la atención de un gran número de científicos, entre ellos biólogos e ingenieros, por diferentes motivos. La ciencia neurobiológica se halla interesada en entender el funcionamiento del cerebro y las estructuras y procedimientos involucrados. En cambio, la ingeniería eléctrica y ciencias de computación están interesadas en obtener paradigmas computacionales con raíces en la neurobiología, para su aplicación en problemas tradicionales tales como reconocimiento temporal y espacial de patrones, visión, voz, control, comunicaciones, etc.

El espectro de investigaciones es muy amplio. Si bien en muchos casos, biólogos, físicos, matemáticos e ingenieros comparten un mismo proyecto, en otros, y en particular en nuestra área de interés, se utilizan modelos neuronales artificiales que no respetan al pie de la letra los postulados biológicos, de los cuales solo toman algunas pocas características beneficiosas. A este respecto podemos decir, que aun teniendo en cuenta la gran diversidad de modelos neuronales artificiales y clasificaciones existentes, existen dos características fundamentales y unificadoras: paralelismo masivo y aprendizaje. Estos temas serán discutidos en profundidad en varios puntos del presente trabajo.

Si miramos a los modelos neuronales artificiales como un conjunto de herramientas para solucionar problemas de ingeniería, como los que ya han sido mencionados, deben considerarse tres aspectos fundamentales:

- representación
- aprendizaje

- implementación

El problema de la representación está íntimamente ligado al desarrollo de la teoría y de los modelos neuronales. Se refiere a las clases de funciones que pueden representarse con una red neuronal. En general, el estudio de una red neuronal puede plantearse como un problema de aproximación, asociación, clasificación, predicción o control. En la teoría de aproximación de funciones, las redes neuronales estáticas pueden verse como interpoladores nolineales que representan la mejor aproximación de una hipersuperficie sobre un dominio dado de muestras, según una norma de medida determinada. Las memorias asociativas y los clasificadores también dependen de la estructura del modelo neuronal para calcular la capacidad o el rango de representación. En el caso particular del control, es fundamental el estudio de la estabilidad no solo de la red neuronal misma, en el caso que ésta tenga dinámica, sino del sistema controlado.

El problema del aprendizaje consiste en encontrar un algoritmo que genere un conjunto óptimo de parámetros de una función neuronal. Estos parámetros pueden ser hallados a través del cálculo de una ecuación cerrada, o utilizando un algoritmo iterativo que modifica sucesivamente los valores de éste vector de parámetros hasta llegar a un punto de equilibrio. El punto de equilibrio existe siempre y cuando el algoritmo de aprendizaje sea estable y converja a un vector de parámetros en tiempo finito o en forma asintótica. En cualquier caso, se considera que el sistema ha aprendido cuando la medida de un criterio de error se halle por debajo del valor mínimo aceptado. Algunos métodos utilizados son análogos a los usados para calcular filtros adaptivos o controladores adaptables.

La implementación de las redes neuronales es el tema principal de esta tesis. Si bien las redes neuronales son interesantes desde un punto de vista teórico, y ya han producido resultados relevantes en el área de reconocimiento de patrones (Haykin, 1994) y el control (Partasarathy, 1992), el fin último de las mismas es plasmar esa enorme potencia de cómputo en implementaciones

paralelas. Las redes neuronales son particularmente atractivas cuando se aplican a problemas complejos, para los cuales las soluciones de los métodos tradicionales son insatisfactorias. Las características adaptivas de una red neuronal juegan un papel preponderante. Pero justamente, estos problemas complejos resueltos con métodos neuronales necesitan una gran potencia de cómputo. Una de las críticas importantes de la que las redes neuronales han sido objeto es, justamente, el largo tiempo que lleva simular y entrenar sus modelos en computadoras digitales que trabajan en forma secuencial. Una computadora orientada a la computación neuronal deberá poseer múltiples unidades computacionales independientes, con un funcionamiento local y alto nivel de interconectividad.

El desarrollo de nuevas técnicas en neurocomputación obliga a estudiar los sistemas neuronales tanto desde el punto de vista de la unidad elemental de cómputo como de la integración que estas unidades deben tener para conformar una estructura eficiente de computación paralela. La interrelación entre la unidad de cómputo y el sistema es escencial dado que cada unidad encierra todo lo necesario para realizar la función de procesamiento elemental. No existen módulos funcionales especializados como en las computadoras convencionales, tales como memoria, unidades aritméticas, interfases, etc. En consecuencia, en esta tesis se realiza un estudio sobre las posibilidades de la electrónica para modelar unidades y sistemas neuronales. En particular, se proponen implementaciones de redes neuronales analógicas. La primera a nivel microelectrónico, comprende el diseño de una neurona computacional con función de activación gaussiana. Primeramente, se demostrará que las unidades computacionales con función de activación gaussiana forman una base para aproximadores universales de funciones continuas y discontinuas medibles. La segunda abarca la realización de una neurocomputadora basada en un dispositivo neuronal analógico. Este sistema aprovecha toda la potencia de cómputo de un procesador neuronal capaz de procesar 300 mil patrones analógicos por segundo. El ambiente desarrollado permite que dicho procesador se convierta en el corazón de una computadora neuronal, dentro de una computadora personal tradicional. Para ello, se desarrolla el hardware de interfaz y procesamiento necesario y el software para comunicación entre

procesadores, conversión análogo-digital de patrones y de manejo del sistema computador.

1.1 ORGANIZACIÓN DE LA TESIS

El capítulo 2 en su primera parte introduce el tema general de las redes neuronales artificiales y centra su atención en un tipo particular, las redes estáticas *feedforward*. En este capítulo se analizan los temas centrales de este tipo de redes, aprendizaje, capacidad y generalización. Se analizan los teoremas fundamentales que aseguran la suficiencia de dos capas en una red neuronal como mejor aproximador de funciones continuas. Como modelos neuronales se utilizan dos tipos diferentes, los perceptrones multicapa y las redes de base radial (RBF) con función de activación gaussiana. Por último se propone el diseño de una red neuronal en un caso típico de aproximación de una curva en un mapa bidimensional.

El capítulo 3 plantea el problema de la generación de modelos neuronales en sistemas electrónicos. Estos sistemas dan lugar a elementos y sistemas de cómputo neuronales. Se detallan las características de técnicas analógicas, digitales e híbridas. Se realiza un análisis comparativo, destacando las ventajas y desventajas en cada caso. Este análisis es utilizado para tomar decisiones en virtud del desarrollo de modelos para neuronas y computadoras neuronales.

El capítulo 4 plantea el uso de dispositivos microelectrónicos en la generación de funciones representativas de modelos neuronales. Primeramente se analizan los bloques constitutivos básicos y algunos problemas y limitaciones de los transistores. A continuación se propone un modelo para una neurona RBF gaussiana y se presentan los resultados obtenidos.

En el capítulo 5 se propone una neurocomputadora a la cual se la ha denominado Netmap. Este sistema permite construir redes neuronales estáticas de una o dos capas de procesamiento. Se describe el sistema

neuronal, la arquitectura multiprocesador utilizada, y los métodos para la optimización temporal en el procesamiento de patrones y modificación de pesos durante el aprendizaje.

El capítulo 6 está dedicado a la programación de tareas dentro de la neurocomputadora. Se analiza la funcionalidad del código dentro del sistema y la comunicación con otros módulos. Se presentan, también, las simulaciones relizadas en función de solucionar el problema de la programación de los pesos de la red neuronal. Se propone un algoritmo de modificación de pesos que incluye al dispositivo neuronal como parte del lazo de aprendizaje.

En el capitulo 7 se hace una discusión sobre los resultados alcanzados y futuras realizaciones en los temas de esta tesis.

CAPITULO 2

REDES NEURONALES ARTIFICIALES

2.1 REDES NEURONALES ARTIFICIALES Y COMPLEJIDAD

A nadie escapa que el avance de las diferentes disciplinas científicas hace al investigador mirar hacia problemas que crecen día a día en complejidad. Intuitivamente podríamos definir la complejidad de un problema en función del esfuerzo y el tiempo de trabajo que debe dedicarse para la obtención de una solución satisfactoria Formalmente, la complejidad puede medirse utilizando diversos indicadores que varían según la disciplina e inclusive dentro de una misma disciplina, respecto de que propiedad del sistema quiera ser conocida. Los métodos científicos utilizados en la mayoría de las disciplinas, incluyendo, por caso, física, matemática, biología, e ingeniería, generan modelos de los sistemas que se desean analizar y/o controlar. En muchos de estos casos, y particularmente, en los que conciernen a la ingeniería, la matemática es la herramienta básica y fundamental para el modelado de la física de un sistema. Esto permite hacer un análisis cualitativo y cuantitativo sobre el comportamiento del sistema y, si es deseable, actuar sobre el mismo para modificar este comportamiento a voluntad.

Si bien no existen dudas que la matemática provee una forma elegante para unificar el discurso sobre especificación de modelos físicos, nuevas ideas para métodos de análisis y control surgen frecuentemente. Entre ellas, las redes neuronales, son de las que mayor crecimiento han tenido en la última década. Los métodos que utilizan redes neuronales artificiales no están contrapuestos con los métodos tradicionales de identificación, interpolación, control, etc. Es más, todas las técnicas conocidas son utilizadas en combinación para atacar un problema complejo de optimización, control, reconocimiento de patrones y muchos otros.

El nombre Redes Neuronales Artificiales (RNA), es por demás ambiguo y la aparición, día a día, de nuevos modelos acentúa aun más esta ambigüedad. En realidad sólo dos premisas son necesarias para pertenecer a la categoría RNA: paralelismo de múltiples unidades de cómputo y aprendizaje o adaptividad de los parámetros en dichas unidades computacionales.

Es difícil unificar definiciones sobre RNA debido a que el espectro de intereses de quienes se han volcado a este tema es muy diverso (Arbib,1989) (Amari,1972,1983) (Lipmann,1987) (Michel,1990) (Tank,1986). Quienes están vinculados a la neurobiología (McCulloch,1943) (Kats,1966) (Hodgkin,1952a,b) (Hinton,1983) tratan de entender y modelar el comportamiento de neuronas y sistemas neuronales biológicos. En el extremo opuesto, ingenieros y científicos de la computación ven a las redes neuronales como unidades de procesamiento que encierran paradigmas computacionales muy potentes (Anderson,1972) (Baum,1989) (Kohonen,1990) (Miller,1990). Existe una continua simbiosis entre estas disciplinas que nos permitiría hacer una vasta clasificación de áreas de interés y aplicaciones (Mead,1987a,b) (Lipmann,1989b).

Una de las razones principales que ha hecho a los ingenieros mirar hacia los sistemas neuronales es el hecho que las computadoras solo son más eficientes que el cerebro humano (y que el de muchos otros animales inferiores) para un conjunto, si bien importante, pequeño de tareas. Básicamente, las computadoras han sido diseñadas para realizar tareas que involucran un procedimiento secuencial, repetitivo, tales como rutinas matemáticas y de acceso a banco de datos. Estas tareas pueden clasificarse como algoritmicamente simples. En cambio, el cerebro aventaja a las computadoras en problemas complejos, especialmente los relacionados con las funciones sensoriales, visión, reconocimiento de voz, etc (Lipmann,1989a) (Bertero,1988) (Narendra,1991) (Nagata,1990). El procesamiento de imágenes realizado por el cerebro es muy superior al alcanzable con cualquier computadora. Las imágenes procesadas son ruidosas, con variación de distancia, posición, e iluminación respecto a un objeto, otras veces con objetos en movimiento o parcialmente ocultos. Los algoritmos que cumplimentan este tipo de tareas en computadoras convencionales son excesivamente complejos y poco tolerantes a fallas. La ventaja más importante de las computadoras tradicionales no es

tanto la función que hacen sino la sorprendente velocidad con que pueden desarrollarla. En otras palabras los tiempos de procesamiento de unidades computacionales electrónicas son 5 o 6 órdenes de magnitud más veloces y continúan mejorando día a día. De todas maneras, estas velocidades no alcanzan a competir en rendimiento y calidad con el cerebro cuando se trata de realizar funciones como las antes mencionadas. Imaginemos, por un momento, la potencia de un cerebro humano cuyas neuronas tuviesen tiempos de cómputo del orden de los nanosegundos.

De todas maneras la intención de la ingeniería no es construir réplicas electrónicas de cerebros humanos sino utilizar ciertas características propias de los sistemas neuronales para realizar nuevas arquitecturas computacionales, diferentes a las arquitecturas secuenciales tipo von Neumann(1946,1958,1986). Se dijo que las características más salientes que permiten darle categoría de red neuronal a un modelo son paralelismo y aprendizaje (Lipman,1987) (Haykin,1994). Paralelismo se refiere a que la arquitectura de una red neuronal cuenta con un número importante de unidades simples que procesan información local, es decir todas al mismo tiempo. El aprendizaje se relaciona con la capacidad que la red posee para modificar su comportamiento en función de un objetivo deseado. En otras palabras, una red neuronal debe tener parámetros adaptables, susceptibles de ser modificados durante dicha etapa de aprendizaje.

En los orígenes de la computación neuronal, si bien existen hitos previos de sistemas adaptativos y de sistemas paralelos, la gran mayoría de los autores señala como trabajos pioneros los de McCulloch-Pitts(1943) y Hebb(1949). Actualmente existen decenas de modelos neuronales que si bien mantienen las dos características fundamentales antes mencionadas, difieren enormemente en su concepción y funcionamiento. Algunos de ellos, como los modelos de Grosberg (1988), poseen mayores similitudes con las redes biológicas que otros tales como los modelos de Hopfield (Tank,1987), Kohonen(1990), o CMAC(Miller,1990).

2.2 ORGANIZACION Y PROPOSITO DEL CAPITULO

El propósito de este capítulo no es hacer un análisis extensivo sobre redes neuronales artificiales sino una introducción sobre un tipo particular, las redes estáticas *feedforward* (Rosemblatt,1958) (Rumelhart,1986a) (Widrow,1990). En la actualidad existe una amplia bibliografía sobre el tema, en muchos casos tratado con una profundidad mayor que la que se presentará en las secciones subsiguientes. El interés aquí es hacer una introducción sobre los modelos neuronales *feedforward*, su potencialidad y sus inconvenientes.

La sección 2.3 analiza algunas categorizaciones posibles y áreas de acción de las RNA. La sección 2.4 define a las redes neuronales estáticas *feed_forward* que serán el tema central de este trabajo. La sección 2.5 analiza la red neuronal más simple, el Perceptrón, los algoritmos de aprendizaje asociados y problemas de convergencia. Las secciones 2.6, 2.7 y 2.8 están dedicadas a las redes neuronales multicapa, en particular al Perceptrón multicapa (MLP) y a las funciones de base radial con salida de activación gaussiana (RBF). Se analiza la potencialidad de estas redes como clasificadores e interpoladores, sus problemas constructivos y capacidad. La sección 2.8 se basa en la teoría de regularización para la aproximación de funciones multivariables de entrada y salida. Finalmente, en la sección 2.9 se utilizan las RBF gaussianas para la aproximación de una función en un mapa bi-dimensional.

2.3 CLASES DIFERENTES DE MODELOS NEURONALES

Sin el propósito de hacer una exhaustiva clasificación de los modelos neuronales, podemos resaltar algunas propiedades intrínsecas que sirven para definir su utilidad, rango de acción y propiedades emergentes. Para una clasificación detallada puede verse Lippmann(1987) o Simpson(1990).

Un primer intento de clasificación puede separar a los modelos neuronales en *estáticos* y *dinámicos* o *recurrentes* (fig.2.1) (Tank,1987) (Kohonen,1990) (Haykin,1995) (Rumelhart,1986a) Los modelos estáticos realizan un mapeo

entre entrada y salida. Despreciando el tiempo de procesamiento interno, la salida se obtiene en forma inmediata en función de la entrada, no existe memoria ni dinámica de estados en el sistema neuronal. Por el contrario los sistemas recurrentes si la poseen, son sistemas realimentados que ante un estímulo de entrada evolucionan hasta converger a una salida estable. Casos típicos de ambos sistemas son el Perceptrón (Rosemblatt,1960a) (de una o múltiples capas) y la memoria asociativa de Hopfield, respectivamente (Tank,1987). Si bien ambos pueden ser utilizados con propósitos similares, por ejemplo reconocimiento de patrones, las redes de Hopfield son ampliamente conocidas como memorias asociativas o direccionables por su contenido, los Perceptrones en cambio se utilizan como clasificadores o para el cómputo de funciones analógicas, o en aproximación de hipersuperficies (Cybenko,1988) (Arai,1989) (Amari,1990) (Poggio,1990a,b) (Huang,1990,1991).

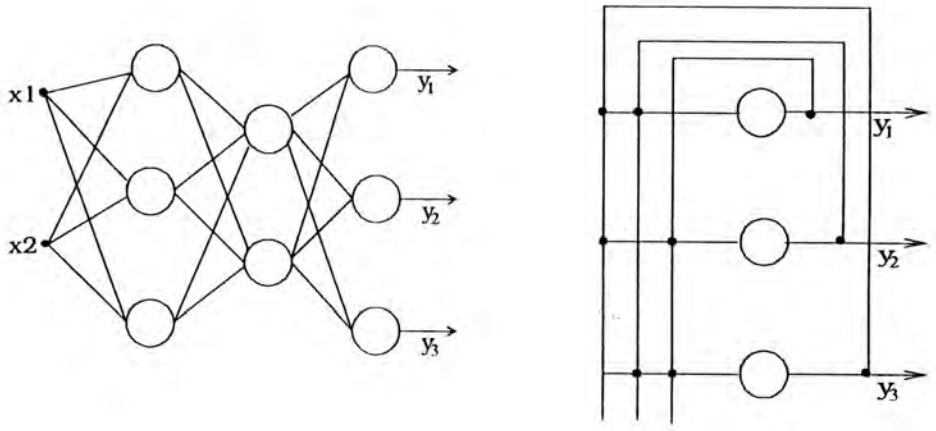

Figura 2.1.a RNA estáticas 2.1.b RNA recurrentes

Otra característica saliente en la clasificación es la referente a la forma en que se realiza el aprendizaje Lippmann(1987) (Haykin,1995) (Rumelhart,1986a). Se dice que el sistema hace un aprendizaje supervisado si existe un tutor que utiliza patrones de entrenamiento para medir el error de la salida de la red respecto a los objetivos deseados y un algoritmo de modificación de los parámetros de la red en función de este error (fig.2.2a) (Hush,1992). Un ejemplo típico es el algoritmo de retropropagación (*backpropagation*) aplicado a Perceptrones multicapa. Cuando no existen patrones de entrenamiento el aprendizaje se dice no-supervisado (fig.2.2b). En este caso los parámetros

adaptativos de la red se modifican en función de otras reglas como por ejemplo momentos estadísticos, correlación entre las entradas, etc. Este tipo de redes forman grupos o clasificaciones basados en cierta redundancia de los patrones de entrenamiento. Como ejemplo típico podemos mencionar las redes de aprendizaje competitivo y las de Hebb (Hebb,1949) (Kohonen,1990) (Haykin,1995).

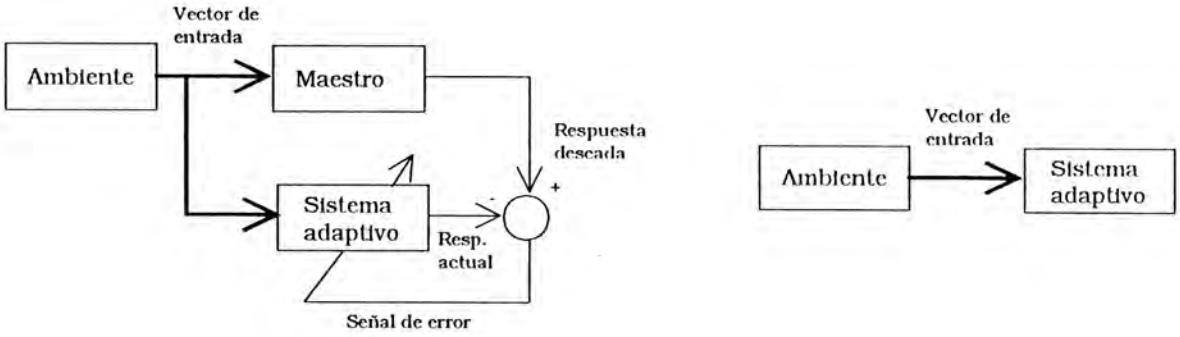

Figura 2.2.a aprendizaje supervisado 2.2.b aprendizaje no supervisado

Otras clasificaciones pueden hacerse respecto a la no linealidad de la función de activación de la neurona: función discontinua (e.g. función signo) (Rosenblatt,1962), o funciones suaves (sigmoide, funciones de base radial tipo gaussiana) (Werbos,1989,1990) (Liu,1993); también respecto de la forma de computar la función de activación (estocástica, p.ej: modelo de Boltzman (Haykin,1995) o determinística p.ej: Perceptrón, Hopfield) y de la forma de seleccionar el coeficiente adaptable que debe actualizarse (en forma sincrónica, asincrónica o aleatoria) (Haykin,1995).

Los diferentes modelos neuronales están dirigidos a áreas de aplicación distintas. Las redes estáticas generan un mapa o función no lineal entre los patrones de entrada y la salida deseada. Filtros y controladores continuos o discretos pueden ser implementados de esta manera. En el caso de salida discreta pueden hacerse clasificadores y decodificadores (Widrow,1985) (Werbos,1992) (Poggio,1990a).

Las redes con dinámica, se valen de una superficie de energía para generar mínimos (atractores) que constituyen memorias de patrones. Estas memorias

tienen el poder de responder a patrones de entrada generando el patrón almacenado más cercano según alguna definición de distancia propicia. Es decir, funcionan como memorias asociativas (Hertz,1991) (Tank,1987).

Las redes que utilizan entrenamiento no supervisado son de utilidad en los casos que se desea clasificar datos utilizando alguna propiedad de los mismos pero que no se conoce a priori. Se utilizan medios estadísticos para separar los datos y generar prototipos de clases. Estos prototipos pueden ser modificados si se modifica la estadística de entrada (Amari,1972) (Kohonen,1990) (Rumelhart,1986a).

En general, las redes neuronales, a pesar de las grandes diferencias entre modelos, poseen similitudes en sus potencialidades y sus inconvenientes. Estas similitudes parten de las dos condiciones básicas para pertenecer a la categoría de red neuronal. El paralelismo y redundancia de conexiones genera robustez. Los algoritmos son distribuidos en el sentido que una falla degrada porcentualmente el rendimiento computacional de la red. Tienen, además, la capacidad de *generalizar* (sec. 2.6.2) (Baum,1989) (Gallant,1990) frente a entradas desconocidas. Por el contrario comparten el problema de los sistemas adaptativos en los cuales se hace difícil (a veces imposible) asegurar la convergencia del algoritmo de entrenamiento.

2.4 MODELOS NEURONALES ESTÁTICOS

Como se dijo, este capítulo solo considera las redes neuronales artificiales estáticas cuyo modelo es de tipo *feedforward*. La estructura es como indica la figura 2.3. La red puede tener una o más capas de unidades de cómputo con una función de activación lineal o no lineal. Las entradas sólo están conectadas a una capa de procesamiento y en los sistemas multicapa, las salidas de una capa son entradas para la capa siguiente (estructura tipo *top-down*) (Rosenblatt,1962) (Rumelhart, 1986a) (Widrow,1988,1990).

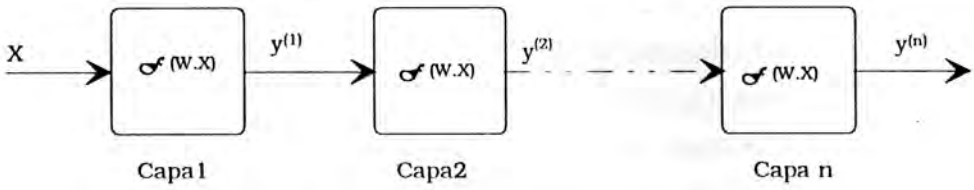

Figura 2.3 Modelo Neuronal *feedforward* de n-capas

2.5 EL PERCEPTRON SIMPLE

La estructura neuronal más sencilla es el Perceptrón simple. Tiene su origen en los trabajos hechos por Rosenblatt en la década del 60 (Rosenblatt, 1960a,b, 1962). Este nombre se ha extendido a las redes neuronales *feedforward* multicapa sin realimentación ni inhibiciones laterales que poseen función de activación sigmoidea. El Perceptrón utiliza entrenamiento supervisado, es decir, los parámetros de la red son adaptados con la ayuda de patrones en los cuales para una entrada dada se conoce la salida que se desea que la red posea. El Preceptrón de Rosemblatt solo posee una capa de unidades de procesamiento.

La figura 2.4 muestra la estructura del Perceptrón simple. Cada neurona recibe el estímulo de una entrada $X \in \Re^n$. Supongamos que estos eventos son discretos, es decir, existe un índice temporal $\mathbf{k} \in \aleph$ tal que el patrón de entrada en un tiempo **k** esté dado por $X_k = [X_0, X_{1k}, \ldots, X_{nk}]$. X_0 es una entrada fija igual a 1 afectada por un coeficiente de peso llamado umbral. El valor escalar de la salida en dicho instante será $Y_k = f(X_k^T . W_k)$. La función **f** actúa sobre el producto interno de la entrada y un vector de coeficientes o pesos $W_k = [W_0, W_{1k}, \ldots, W_{nk}]$, $W \in \Re^{n+1}$. Esta función, denominada función de activación, puede ser lineal, no lineal suave (p.e. sigmoidea) o con una discontinuidad dura como en el caso de la función signo (fig. 2.5). La función de activación es primordial en la elección de una arquitectura de red debido a que tiene influencia en el tipo de problemas que pueden resolverse con dicha red y también en la convergencia de los algoritmos de aprendizaje.

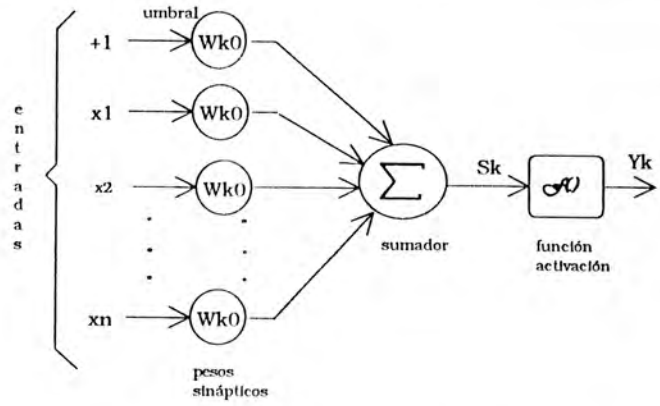

Figura 2.4 Perceptrón simple

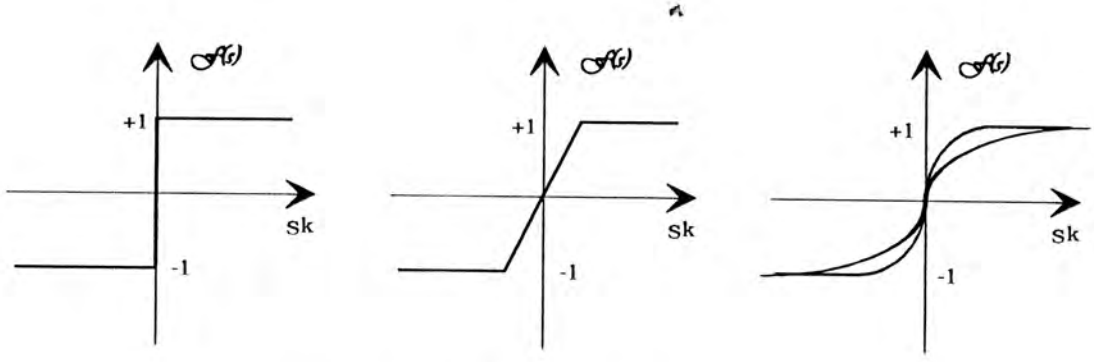

Figura 2.5 Funciones de activación

Es fácil demostrar que el Perceptrón simple sólo puede aprender a clasificar patrones que son linealmente separables (Rosenblatt, 1960a). Es decir aquellos X, $X \in \Re^n$, en los cuales puede encontrarse un hiperplano capaz de dividirlos en dos clases. En $\Re^2$ el plano divisor sería una recta de ecuación

$$X_2 = -\frac{W_1}{W_2} \cdot X_1 - \frac{W_0}{W_2}$$

El contra ejemplo clásico en la literatura es la función binaria XOR, la cual no puede ser clasificada correctamente con un Perceptrón simple capa (fig. 2.6.a). Este tipo de problemas puede ser solucionado potenciando la unidad de cómputo o utilizando una red multicapa (Widrow,1990).

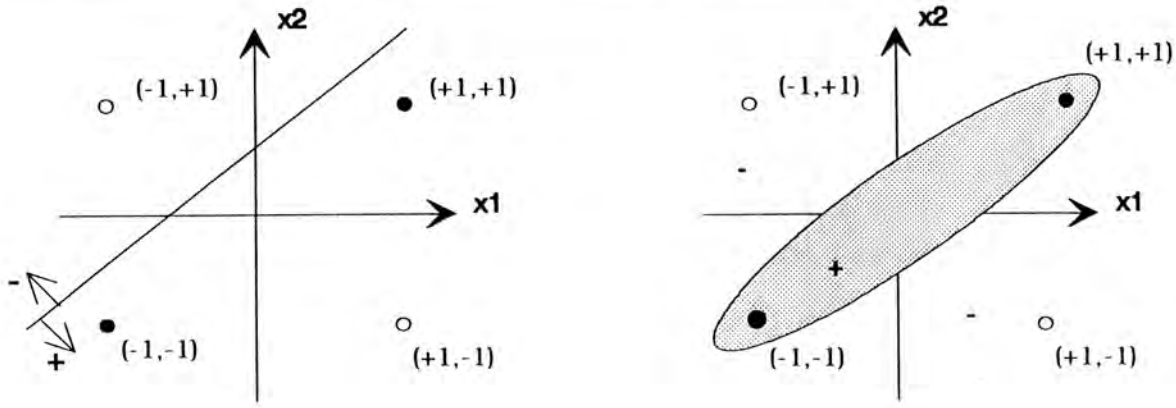

Figura 2.6 problema del XOR a) unidad lineal b) preprocesamiento cuadratico

Los clasificadores no lineales corresponden al primer tipo mencionado en el párrafo anterior. Si se hace un preprocesamiento del vector de entrada utilizando una función polinómica, trigonométrica, de base radial, o similar, pueden clasificarse patrones aunque no sean linealmente separables. La figura 2.6.b muestra un ejemplo de función cuadrática discriminante de entrada que permite implementar correctamente una XOR de dos entradas. El estudio de las funciones preprocesadoras es importante en problemas de interpolación y aproximación de funciones. La habilidad que la red tenga para generalizar sobre patrones nuevos depende en gran medida de estas funciones. El módulo preprocesador puede también ser considerado como una primer capa de unidades de procesamiento, convirtiendo, así, a este esquema en una arquitectura multicapa. En la sección 2.6 se analizan redes neuronales multicapa.

2.5.1 ALGORITMOS DE APRENDIZAJE EN PERCEPTRONES SIMPLES

Si bien es cierto que el Perceptrón solo puede representar una clase limitada de funciones, es lo suficientemente simple y existe un análisis formal que describe su comportamiento. Dado que una de las arquitecturas de redes neuronales más utilizadas se compone de varias capas de unidades tipo Perceptrón, los

análisis pueden hacerse extensivos, algunas veces, a redes multicapa o al menos pueden dar alguna pista sobre los problemas de redes más complejas.

El Perceptrón posee un vector **W** de elementos adaptables. La adaptación se hace utilizando un algoritmo cuya finalidad es la de minimizar algún funcional de la medida del error de salida ε_k. El error se genera como diferencia entre una señal tutora y una señal de salida, lineal o no-lineal, de la neurona (fig. 2.4). Un algoritmo de adaptación lineal utiliza el error $\varepsilon_k = d_k - s_k$, donde d_k es la salida deseada y s_k una función lineal de $\mathbf{x_k}$ y $\mathbf{w_k}$, que representa la salida del bloque lineal. en cambio un algoritmo no-lineal define el error en función de la salida Y_k. Es decir, $\varepsilon_k = d_k - Y_k$

2.5.1.1 ALGORITMOS LINEALES

El algoritmo lineal más utilizado en Perceptrones simples es el α-LMS de Widrow-Hoff (Nguyen,1989a) (Hinton,1990) (Widrow,1990) (Haykin,1995). Este algoritmo es lineal respecto de los parámetros adaptables **W** de la red. Se define al error por:

$$\varepsilon_k = d_k - s_k = d_k - W_k^T \cdot X_k \qquad (2.1)$$

La adaptación se efectúa en cada presentación de patrones siguiendo el principio de disturbancia mínima para mantener los patrones ya aprendidos. En el instante **k**, el vector de pesos es calculado como

$$W_{k+1} = W_k + \alpha \cdot \frac{\varepsilon_k X_k}{|X_k|^2} \qquad (2.2)$$

El nuevo vector de pesos es generado en base al valor del instante **k** más una componente que tiene la misma dirección del vector del patrón de entrada X_k. Puede verse que el ΔW_k es proporcional al error ε_k y a un coeficiente α que controla la velocidad de convergencia del algoritmo.

Reemplazando (2.1) en (2.2)

$$\Delta \varepsilon_k = \Delta(d_k - W_k^T \cdot X_k) = - X_k^T \cdot \Delta W_k$$

$$\Delta \varepsilon_k = -\alpha \varepsilon_k$$

Se observa que una modificación ΔW_k en el vector de pesos produce una variación $\Delta \varepsilon_k$ proporcional al error de salida que genera el patrón de entrenamiento. Si hacemos el coeficiente α igual a 1 el error para el patrón **k** se reduce a cero en una sola iteración. Infortunadamente si los patrones de entrada no son ortogonales, que es en realidad el caso habitual, el próximo patrón presentado a la red puede desandar el camino hecho por un patrón anterior, haciendo que el algoritmo diverja o tenga un ciclo límite con un error inaceptable. De todas maneras el estudio de convergencia del algoritmo α-LMS es sencillo debido a que el sistema es lineal y tiene un solo mínimo. Valores prácticos de α son entre 0.1 y 0.5.

2.5.1.2 ALGORITMOS NO LINEALES

El algoritmo original de Rosenblatt mide el error entre la señal tutora y la salida del cuantificador. La función es por lo tanto no lineal (Rosenblatt, 1960b)

$$\varepsilon_k = d_k - y_k = d_k - \mathrm{sgn}\left(\sum_j W_{ij}^T \cdot X_{jk} \right) \qquad (2.3)$$

En este caso la adaptación se hace sumando o restando el vector de entrada X_k al vector de pesos W_k dependiendo de que el error sea positivo o negativo respectivamente.

$$W_{k+1} = W_k + \alpha . \frac{\varepsilon_k X_k}{2} \qquad (2.4)$$

Debido a que **d** e **y** son binarias ±1, el error $\varepsilon_k \in \{-2,0,2\}$. El algoritmo usa generalmente valores de **α=1**. El teorema de convergencia del Perceptrón demuestra que si existe una solución al problema, es decir si los patrones son

linealmente separables, el algoritmo converge a la solución en un tiempo finito **M**. La demostración de este teorema está basada en el cómputo de límites en el vector |W| y en la superposición (*overlap*) **W.W***, siendo **W*** el valor solución. El teorema demuestra que existe un número finito de pasos **M** tal que

$$M \leq \frac{N}{D_{max}^2}$$

donde **D** representa la distancia del patrón de entrada más cercano al hiperplano que separa las clases (figura 2.7). **Dmax** es esta misma distancia cuando el vector de pesos es óptimo. La figura 2.7 muestra la evolución de la distancia **D(w)** durante el aprendizaje. Es obvio que **M** depende del número de entradas **N** e indirectamente y en forma inversamente proporcional, del número de patrones que deben separarse. En realidad puede decirse que estadísticamente **Dmax** disminuye a medida que aumentamos el número de patrones **p**. Una demostración completa del teorema de convergencia está disponible en (Rosenblatt,1960b) (Haykin,1995) (Hertz,1991).

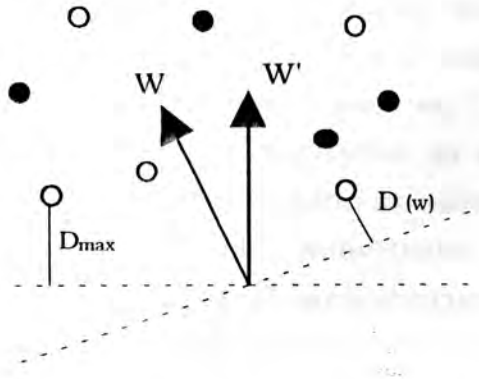

Figura 2.7 distancia al plano $w^T.x = 0$

El mayor problema del algoritmo original del Perceptrón simple es que si los patrones a clasificar no son linealmente independientes, el algoritmo no termina nunca y el hiperplano de separación oscila permanentemente. Además, se observa que el algoritmo α-LMS lo aventaja en el hecho de que se acerca mejor a una solución sub-óptima cuando los patrones no son linealmente separables, infortunadamente debido a que el objetivo del α-LMS es reducir el

error lineal, no siempre es capaz de separar cualquier conjunto linealmente separable aunque se aproxime a ello.

Pueden encontrarse numerosas variantes dentro de la bibliografía, tal vez la más utilizada es la que define una zona muerta alrededor del valor cero de $\mathbf{S_k}$. El algoritmo de adaptación fuera de la zona muerta es igual al anterior, pero si $\mathbf{S_k}$ está dentro de la zona, **W** es adaptado como si la salida fuese incorrecta. Este algoritmo tiene el mismo rendimiento que el anterior en patrones linealmente separables y funciona mejor que el primero en caso que no lo sean.

2.5.1.3 ALGORITMOS DE GRADIENTE

Los algoritmos anteriores tienen como objetivo reducir el error instantáneo en cada uno de los patrones presentados a la red. Los algoritmos de gradiente, en cambio, realizan la adaptación de los parámetros minimizando el error promedio en todo el conjunto de patrones de prueba. El funcional más utilizado es, sin duda, el que minimiza el error medio cuadrático (MSE). La convergencia global por este método está garantizada para Perceptrones simples con función de activación lineal o no lineal suave, debido a que la superficie de error posee un único mínimo.
Otros métodos más sofisticados, como gradiente generalizado y cuasi-Newton ofrecen mejores características de convergencia pero exigen cumplimentar condiciones adicionales las cuales no siempre pueden ser garantizadas.

El algoritmo MSE aplicado al vector de pesos $\mathbf{W_k}$ es:

$$W_{k+1} = W_k + \mu(-\nabla_k) \qquad (2.5)$$

donde μ el la constante de aprendizaje y ∇_k es el gradiente en un punto de la superficie de error MSE en función de $\mathbf{W_k}$.

El algoritmo de gradiente aplicado al Perceptrón simple es lineal si la adaptación del vector de pesos es proporcional al error lineal.

Sea $X \in \Re^n$ con $X=[X_1, X_2, \ldots, X_n]$ el patrón de entrada de una red que posee, luego de la iteración **k** un vector de pesos $W_k=[W_{1k}, W_{2k}, \ldots, W_{nk}]$, con $W \in \Re^n$. El error cuadrático se define como

$$\varepsilon_k^2 = (d_k - X_k^T . W_k)^2$$
$$\varepsilon_k^2 = d_k^2 - 2 d_k X_k^T W_k + W_k^T X_k X_k^T . W_k$$

Tomando el promedio sobre el conjunto total de patrones de entrenamiento,

$$E[\varepsilon_k^2]_{W_k} = E[d_k^2] - 2E[d_k X_k^T] W_k + W_k^T E[X_k X_k^T] W_k \qquad (2.6)$$

se observa que esta es una función cuadrática en el vector de pesos. Las esperanzas pueden definirse como correlaciones entre entrada y salida y matriz de autocorrelación respectivamente. Por lo tanto el error tiene la forma de un hiperparaboloide en $\Re^n$ con un único mínimo. En dicho mínimo el gradiente es cero, por lo tanto podemos hallar el vector de pesos óptimo usando el gradiente

$$\nabla_k = -2E[d_k^2 X_k^T] + 2E[X_k X_k^T] W_k \qquad (2.7)$$

El óptimo se encuentra en el valle donde $\nabla_k=0$.

$$W^* = E[X_k X_k^T]^{-1} . E[d_k^2 X_k^T] \qquad (2.8)$$

esta es la ecuación de Wiener. La matriz **W*** que la representa se denomina matriz de Wiener-Hopf (Wiener, 1948).

Widrow y Lehr (Widrow, 1990) proponen el algoritmo μ-LMS utilizando el gradiente instantáneo en lugar del verdadero gradiente para acelerar el algoritmo. Puede demostrarse que si μ es pequeño, el gradiente instantáneo se aproxima al gradiente verdadero. El valor de μ está acotado por la inversa de la traza de la matriz de autocorrelación de entrada. Una prueba de esto puede encontrarse en (Widrow, 1985)

$$0 < \mu < \frac{1}{\text{traza} E[X_k X_k^T]}$$

2.5.1.3.1 Convergencia del algoritmo de gradiente

Puede hacerse un análisis más riguroso sobre la convergencia y la velocidad de convergencia del algoritmo del gradiente haciendo una transformación lineal que diagonalice la forma cuadrática (2.7)

$$E(W) = (\omega_\lambda - \omega_\lambda^*)^T . a_\lambda . (\omega_\lambda - \omega_\lambda^*) \qquad (2.9)$$

Los ϖ_λ son combinaciones lineales de los pesos $\mathbf{W_j}$. Los a_λ son los autovalores del sistema y son mayores o iguales que cero, ya que la forma cuadrática es positiva semi-definida. ω_λ^* representa el valor óptimo de pesos en la dirección del autovector λ. Note que si algún $a_\lambda=0$, E es constante en la dirección del autovector correspondiente. Dado que la transformación es lineal, minimizar E(W) en (2.7) y (2.9) es equivalente. En consecuencia la variación en el vector de pesos es

$$\Delta W_\lambda = -2\mu a_\lambda (\omega_\lambda - \omega_\lambda^0) \qquad (2.10)$$

definiendo $\delta\omega_\lambda = \omega_\lambda - \omega_\lambda^0$ como la distancia al óptimo en la dirección λ, su variación es

$$\delta\omega_\lambda^{new} = \delta\omega_\lambda^{old} + \Delta\omega_\lambda = (1 - 2\mu a_\lambda)\delta\omega_\lambda^{old} \qquad (2.11)$$

vemos que el algoritmo converge para $|1 - 2\mu a_\lambda| < 1$. Es decir que el valor de μ está condicionado por el máximo autovalor del sistema según

$$\mu < \frac{1}{a_\lambda^{max}}$$

2.5.1.4 FUNCIONES DE ACTIVACIÓN NO LINEALES

Las funciones no lineales son computacionalmente más potentes que las lineales (Rumelhart,1986a) (Widrow,1990) (Hertz,1991) (Haykin,1995) (Funahashi,1989) (Tank,1987). Ya hemos visto que los Perceptrones simples

solo pueden clasificar conjuntos linealmente separables. Las limitaciones para aproximar funciones con sólo unidades lineales son obvias. Además, existen muchas funciones que no pueden ser resueltas con redes de una única capa y sí con arquitecturas multicapa. En este caso, vale aclarar que un arreglo multicapa de unidades lineales no mejora la capacidad de representación de una sola capa dado que cada capa es una transformación lineal de la entrada, y una composición de múltiples transformaciones lineales puede ser reemplazada por una única transformación lineal, es decir, por una sola capa. En otras palabras, toda composición de transformaciones lineales posee una transformación equivalente representada por una única transformación que es una matriz igual al producto de las matrices de las transformaciones individuales.

En general las unidades computacionales con función de activación no lineal obligan a definir una función error no lineal. Si la función de activación es derivable, podemos utilizar el algoritmo de gradiente anteriormente descripto aplicado a la función del error. La función sigmoidea de la figura 2.5 $(f:\Re^n \to \Re)$, es infinitamente derivable, C^∞, y monótonamente creciente. Los ejemplos más utilizados son

$$y_k = tgh(S_k)$$

$$y_k = \frac{1}{1+e^{-S_k}}$$

Entonces, el error medio cuadrático se calcula como

$$E(\varepsilon^2) = \frac{1}{2}\sum_k \left(d_k - tgh(W^T.X_k)\right)^2 \qquad (2.12)$$

el algoritmo utilizado es el mismo de la ecuación (2.5). Diferenciando (2.12) para calcular el gradiente

$$\nabla_k = \frac{\partial E(\varepsilon_k)^2}{\partial W_k} = 2\sum_k (X_k . e_k . tgh'(S_k))$$

$$W_{k+1} = W_k + 2\mu\,\varepsilon_k\, tgh'(S_k)\, X_k$$

Las funciones utilizadas para la sigmoide facilitan el cálculo de la derivada. En nuestro caso tgh'(.) = 1-tgh(.)^2. Entonces,

$$W_{k+1} = W_k + 2\mu\,\varepsilon_k\left(1 - tgh^2(S_k)\right) X_k \qquad (2.13)$$

la derivada se calcula utilizando los valores ya obtenidos de **tgh(.)**. Si se define $\delta_1^k = \varepsilon_k . tgh'(S_k)$ la cual nos da información de cuan sensible es el error cuadrático a una variación de la salida lineal **Sk**. Podemos escribir

$$W_{k+1} = W_k + 2\mu\,\delta_k\, X_k \qquad (2.14)$$

Podemos concluir diciendo que el algoritmo de gradiente aplicado a una función cuadrática del error es sencillo y tiene asegurada la convergencia. Sin embargo las redes neuronales de mayor utilidad se componen de una o mas capas de neuronas con funciones de activación no lineal. En estos casos, el algoritmo de gradiente debe recorrer una superficie de error que no tiene un único mínimo, con lo cual puede quedar atrapado en mínimos locales. En una red monocapa, las no linealidades suaves de la sigmoide no ocasionan problemas de convergencia, salvo en la velocidad con que el algoritmo converge para **|Sk|** grandes, ya que **ΔWk** se hace pequeño. Cuando la no linealidad utilizada es una función discontinua del tipo de la función signo, la superficie de error se torna mucho más compleja, con mínimos relativos numerosos e importantes.

2.6 PERCEPTRONES MULTICAPA

Las limitaciones funcionales de las redes *feedforward* monocapa pueden ser salvadas con estructuras multicapa (Rumelhart,1986a) (Widrow,1990) (Hertz,1991) (Haykin,1995) (Funahashi,1989). La potencialidad de los Perceptrones multicapa (MLP) puede ser analizada según su naturaleza discreta o continua por la habilidad de representar funciones Booleanas de n-entradas, funcionando como un clasificador de patrones; y por su capacidad de aproximar funciones para realizar transformaciones no lineales.

Funahashi (1989), Poggio(1990) y Cotter(1990) entre otros autores demuestran que la no linealidad de la función de activación no es crucial en la potencialidad de la red neuronal para computar un dado mapa de entrada/salida. Se ha demostrado, también, que una red neuronal con una capa oculta con unidades de cómputo no lineal y una capa de salida lineal es capaz de aproximar cualquier función con un error arbitrariamente pequeño. Sin embargo funciones poco suaves pueden hacer que el número de unidades computacionales crezca enormemente. Existe también evidencia de que aumentar el número de capas en una red neuronal puede ayudar a reducir el número de unidades de cómputo. Infortunadamente, en general, no existe un procedimiento constructivo para la determinación de variables importantes de la red, tales como número de capas, número de unidades por capa, cantidad de muestras necesarias para el entrenamiento, etc.

Las siguientes secciones del capítulo están dedicadas al análisis de dos tipos diferentes de redes multicapa, los llamados Perceptrones multicapa y las redes que utilizan funciones de activación gaussiana.

Las principales lineas de investigación en el tema se han ocupado de interrogantes como:

- Qué funciones pueden ser representadas por redes neuronales *feedforward*?
- Cuál es el número óptimo de capas para una dada red ?
- Cuál es el número óptimo de unidades por capa de la red ?
- Cómo debe entrenarse la red para lograr convergencia sobre la superficie de error ?. Y cuál es la forma más conveniente de definir el error ?

Los aportes sobre capacidad, generalización y convergencia de los algoritmos de aprendizaje han hecho aportes significativos al respecto, aún cuando siguen siendo materia de corriente investigación (Baum,1989) (Bertero,1988) (Vapnik,1971) (Huang,1990,1991).

Los Perceptrones multicapa pueden ser vistos como una extensión del Perceptrón simple, al que se le ha agregado una o más etapas similares en cascada (fig. 2.8). La función que realiza una estructura de este tipo es

$$F(x,w) = \sigma\left(\sum_n w_n . \sigma\left(\sum_j v_j . (\ldots)\right)\right) \qquad (2.15)$$

donde σ es la función de activación de tipo sigmoidea o signo, y W=(w_n, v_j, ...) el vector de parámetros.

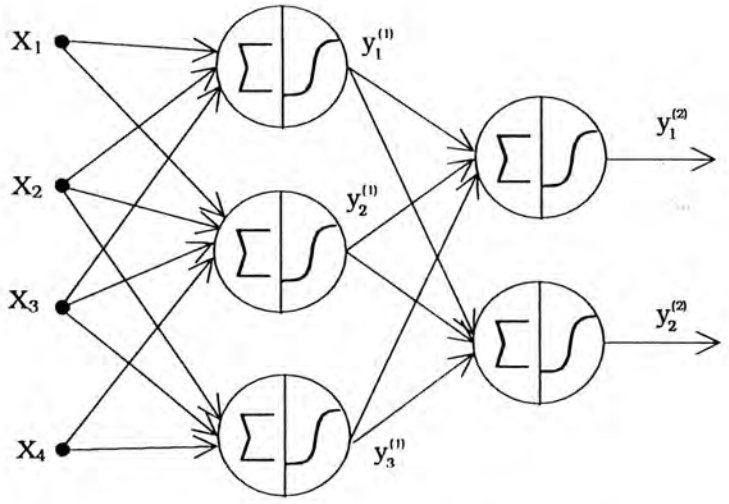

Figura 2.8 Perceptrón multicapa

2.6.1 APRENDIZAJE EN REDES MULTICAPA

La adaptación de los coeficientes de peso de cada una de las capas de una red tipo MLP no es tan simple como en un Perceptrón monocapa. Primeramente, existe el problema de relacionar las variaciones de estos coeficientes con una efectiva reducción del funcional del error. Luego, el problema de implementar un algoritmo que sea capaz de recorrer en forma efectiva el espacio de estados de la superficie del error hasta encontrar un vector de pesos óptimo en cada capa de la red (Rumelhart,1986a).

La solución más utilizada en los últimos años es el algoritmo de retropropagación. Si bien este algoritmo fue inventado en forma independiente por Werbos (1974), Parker (1987), y LeCun (1988), se debe finalmente a Rumelhart (et al) (1986a) la gran difusión y aceptación que ha logrado. Estos trabajos junto a los de Hopfield (1985,1986) han producido un enorme crecimiento en el estudio de las redes neuronales.

Los conceptos básicos del algoritmo de retropropagación son similares a los usados en la adaptación de los parámetros del Perceptrón simple. La actualización de los pesos se hace utilizando la misma ecuación (2.14)

$$W_{k+1} = W_k + 2\mu\,\delta_k\,X_k \qquad (2.16)$$

simplemente se modifica la definición de los δ_k.

La figura 2.8 muestra una red *feedforward* de dos capas. El vector de entrada tiene conexiones sólo con las unidades de la capa 1, las salidas de esta capa sirven como entradas a la capa siguiente. Definiendo las relaciones de entrada/salida de la misma forma que en la sección 2.5.1, y agregando un supra-índice que indica el número de capa referida, valen las siguientes ecuaciones. En todas las ecuaciones los índices j e i refieren la entrada y salida de una conexión neuronal, respectivamente; y el índice k, en todos los casos, representa al k-esimo patrón presentado a la red.

$$y_i^{(2)} = sgm\left(\sum_{j=1}^{N2} W_{ij}^{(2)}.sgm\left(\sum_{j=1}^{N1} W_{ij}^{(1)}.X_{j(k)}\right)\right) \qquad (2.17)$$

$$E(\varepsilon^2) = \frac{1}{2}\sum_{\forall i,k}\left(d_k - y_k^{(2)}\right)^2 \qquad (2.18)$$

$$\Delta W_{ij}^{(l)} = -\mu.\frac{\partial E(\varepsilon_k)^2}{\partial W_{ij}^{(l)}}X_{j(k)} \qquad (2.19)$$

donde el supra-índice l indica una de las capas de la red neuronal en cuestión. Para derivar la ecuación (2.17) respecto de un peso de la capa de salida utilizamos la regla de la cadena,

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = \frac{\partial E}{\partial S_j^{(2)}} \cdot \frac{\partial S_j^{(2)}}{\partial W_{ij}^{(2)}} = -2 \sum_{\forall i,k} (d_{ik} - y_{ik}).sgm'(S_j^{(2)}) y_{i(k)}^{(1)} \cdot \frac{\partial S_i^{(2)}}{\partial S_j^{(2)}}$$

$$\frac{\partial E}{\partial W_{ij}^{(2)}} = -2 \sum_{k} \delta_{i(k)} \cdot y_{i(k)}^{(1)}$$

note que la sumatoria en **k,i** pierde el indice **i** debido a que todas menos una de las salidas **Si** son independientes de **Wij**. El **δi** de la capa exterior es definido por,

$$\delta_{i(k)}^{(2)} = (d_{ik} - y_{ik}).sgm'(S_{i(k)}^{(2)})$$

Si se desea modificar un peso de la capa 1, debe calcularse el gradiente

$$\nabla_k = \frac{\partial E}{\partial W_{ij}^{(1)}} = \sum_{i=1}^{N2} \frac{\partial E}{\partial S_i^{(2)}} \cdot \frac{\partial S_i^{(2)}}{\partial S_j^{(1)}} \cdot \frac{\partial S_j^{(1)}}{\partial W_{ij}^{(1)}}$$

$$\frac{\partial S_j^{(1)}}{\partial W_{ij}^{(1)}} = X_{j(k)}$$

$$\frac{\partial S_i^{(2)}}{\partial S_j^{(1)}} = \frac{\partial}{\partial S_j^{(1)}} \left[\sum_{\forall j} W_{ij}^{(2)} y_j^{(1)} \right] = \frac{\partial}{\partial S_j^{(1)}} \left[\sum_{\forall j} W_{ij}^{(2)} .sigm(S_j^{(1)}) \right]$$

$$\frac{\partial S_i^{(2)}}{\partial S_j^{(1)}} = \sum_{\forall j} W_{ij}^{(2)} .sigm'(S_j^{(1)}) = 2\delta_i^{(1)}$$

puede verse que la derivada de **E** respecto de un **Si** de la capa (1) involucra a todos los **Si** de la capa (2) ya que todos poseen una conexión a través de un $W_{ij}^{(2)}$. Pero cada neurona de la capa (2) posee una y sólo una conexión a un dado **Si** de la capa (1), en consecuencia, sólo un término de cada sumatoria es afectado por estas variaciones. Para el ejemplo de la figura (2.8)

$$\frac{\partial E}{\partial S_i^{(1)}} = -2(d_{1k} - y_{1k}).sgm'(S_1^{(2)}).W_{1i}^{(2)}.sgm'(S_1^{(1)}) - 2(d_{2k} - y_{2k}).sgm'(S_2^{(2)}).W_{2i}^{(2)}.sgm'(S_2^{(1)})$$

$$\frac{\partial E}{\partial S_i^{(1)}} = -2 \sum_{\forall j} (d_{jk} - y_{jk}).sgm'(S_j^{(2)}).W_{ji}^{(2)}.sgm'(S_j^{(1)})$$

si definimos $\delta_{i(k)}^{(1)} = \frac{\partial E}{\partial S_{i(k)}^{(1)}}$ para un dado patrón k, podemos escribir

$$\delta_{i(k)}^{(1)} = sgm'\left(S_{i(k)}^{(2)}\right)\sum_{\forall j} W_{ij}^{(2)} \cdot \delta_{i(k)}^{(2)}$$

Las variaciones de los pesos durante el aprendizaje por retropropagación pueden ser vistas invirtiendo el sentido de flujo de señal en la red neuronal; y utilizando los δ como señales de entrada. Puede verse que los δ (fig. 2.9) se propagan desde la salida hacia la entrada (retropropagación) multiplicados por el coeficiente de peso que vincula a la conexión entre dicho δ y un nodo de la capa inmediata inferior, multiplicado por un factor que depende de la derivada de la función sigmoidea en el nodo de destino. En otras palabras los δ en cada capa pueden obtenerse pensando que en la red se han invertido el rol de la salida y la entrada, y que cada nodo de las nuevas entradas es excitado por una señal igual al δ calculado para la capa de salida. En consecuencia, las modificaciones de los pesos se realizan de acuerdo a

$$\Delta W_{ij}^{(1)} = -\mu \cdot \sum_{\forall k} \delta_{ik}^{(1)} \cdot X_{jk}^{(1)} \qquad (2.20)$$

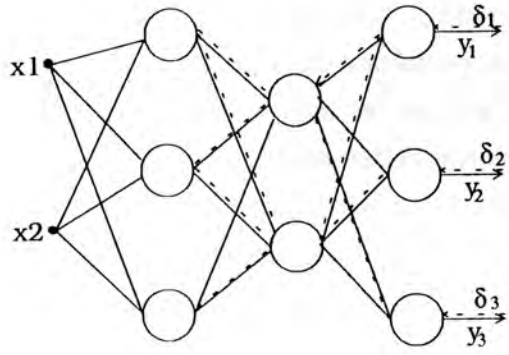

Figura 2.9 Generación de los δ en el algoritmo de retropropagación

La modificación de los pesos en el algoritmo de retropropagación es proporcional a la derivada de la función sigmoidea en el punto de operación. Debido a la forma de la sigmoide esta derivada puede ser muy pequeña cuando los errores medidos son grandes, originando superficies de error planas y extensas que hacen excesivamente lenta la convergencia del algoritmo. Mejoras

respecto del algoritmo original pueden encontrarse en Guo (1991), Kruschke (1991), Rumelhart (1986b), Solla (1988), Nguyen (1989a), Lipmann (1987), Hush (1992).

2.6.2 PROBLEMAS Y LIMITACIONES DE LAS REDES ESTÁTICAS MULTICAPA

Los problemas fundamentales de los MLP y otros modelos estáticos multicapa se condicionan tanto a la especificación como al aprendizaje. Es decir, es fundamental determinar qué tipo de problemas pueden ser solucionados usando un dado modelo neuronal. Formalmente, qué clases de funciones pueden ser representadas o, al menos, aproximadas. Luego, si un modelo neuronal es considerado apto para cierta aplicación, debe determinarse el tamaño de la red neuronal; dado tanto por el número de capas como por el número de unidades computacionales por capa. Finalmente, debe atacarse el problema del aprendizaje, dado que el buen funcionamiento de los algoritmos es dependiente de los datos de entrada.

Uno de los cuestionamientos más importantes que se le hacen a las redes neuronales está referido a que no existe un procedimiento constructivo en la determinación de las variables de diseño para un dado problema. Funahashi (1989), Poggio (1990) y otros han hecho aportes sobre la representación de mapas continuos usando redes multicapa. Funahashi demuestra que cualquier mapa continuo puede ser aproximado con precisión infinita con una red MLP que posea al menos una capa de unidades ocultas. De todas maneras, sus teoremas no son constructivos y el número de neuronas en la capa oculta puede ser excesivamente grande. Lo mismo sucede con las redes de base radial gaussianas, que son muy potentes para detectar características locales en un espacio n-dimensional de entrada, pero cuyo tamaño aumenta exponencialmente con la dimensión del espacio.

Huang (Huang, 1991) demuestra que en el caso de MLP funcionando como clasificadores, una estructura multicapa soluciona el problema de

separabilidad lineal, simplemente haciendo que la segunda capa implemente funciones del tipo OR, AND o MAJ de las salidas de la primer capa.

Es difícil poder conocer a priori el número óptimo de capas y neuronas para una dada aplicación, pero al menos pueden hallarse ciertas cotas que permiten comenzar con una red de tamaño adecuado. Luego de la etapa de adaptación de los parámetros y verificación de resultados, pueden aplicarse técnicas de reducción (*pruning*) que modifiquen el tamaño de la red siguiendo un criterio de optimalidad (Hush, 1993).

Las cotas para el diseño de una red neuronal se obtienen haciendo estudios de generalización y capacidad en un dado modelo neuronal. Las dos secciones (y subsecciones) siguientes presentan una introducción al estudio de dichos problemas los cuales son lineas fuertes de investigación.

2.6.3 GENERALIZACIÓN

La generalización representa una medida del comportamiento de una red neuronal entrenada frente a estímulos de entrada que no han sido presentados durante la faz de entrenamiento. Es decir, entradas que no han influido sobre las sucesivas modificaciones que el vector de pesos ha tenido durante el aprendizaje (Baum, 1989) (Vapnik,1971,1982) (Gallant,1990) (Muroga,1966).

La generalización depende fundamentalmente de tres factores:

- el número de patrones de entrenamiento
- la relación de éstos con la complejidad del problema
- el tamaño de la red

Intuitivamente, puede decirse que para un dado problema, cuantos más datos tengamos en mano, mayor definición tendrá el problema subyacente. Por otro lado, cuanto mayor sea nuestra red, la probabilidad, que pueda solucionar un

dado problema se acrecentará. En consecuencia, podemos tratar el problema de generalización con dos hipótesis diferentes:

- manteniendo una estructura de red fija, conforme a las dimensiones del problema, y evaluando la generalización en función del número de patrones de entrada.

- con un dado conjunto de patrones de entrada que se encuentran en el espacio $\Re^n$, y determinando el tamaño óptimo de la red para minimizar un funcional medido sobre ese conjunto de datos.

El primero de estos problemas es analizado a través de la generalización promedio en función de los datos de entrada. El segundo establece un límite para la generalización en el peor caso.

2.6.3.1 GENERALIZACIÓN PROMEDIO

Si bien estos conceptos pueden ser extendidos al caso continuo, serán analizados para el caso de vectores en el espacio $\{0,1\}^N$

Definición 2.1: Sea **F** el conjunto de todas las funciones lógicas **f : x->y** con $\mathbf{x} \in \Re^n$ e $\mathbf{y} \in \Re$ implementables por una red neuronal; y sea $\mathbf{f} \in \mathbf{F}$ una de esas funciones. Se define una medida de la generalización **g(f)** como la fracción del dominio correctamente representado por **f**.

Definición 2.2: Sea $\mathbf{F_P} \subset \mathbf{F}$ el conjunto de funciones en **F** consistente con un conjunto de patrones de entrada **P**. Entonces, se define la generalización promedio como el promedio de los valores de generalización de todas las funciones $\mathbf{F_P}$.

Los trabajos sobre generalización promedio, iniciados por Schuartz (1990), sorprenden por el hecho de que, luego de la faz de aprendizaje, para una dada red y conjunto de patrones de entrenamiento, la generalización promedio puede

ser calculada a priori, antes de la etapa de validación, si se conoce la distribución de la generalización en el instante inicial.

Dada una red **M(x,w)** de arquitectura fija **R**. Puede definirse el volumen del espacio que ocupan todos los posibles vectores **W** como

$$V_0 = \int \rho(W).dW \qquad (2.21)$$

en el cual **ρ(w)** representa la distribución de pesos **w** en el espacio R-dimensional. Esta integral está definida en un compacto. Por ejemplo, cualquier algoritmo de aprendizaje que normalice el vector **w** cumple con las condiciones para que la integral no sea divergente. Luego, cada función **f** en **F** puede ser representada por un conjunto infinito de valores de **w** que ocupan un volumen **V_0(f)** incluido en **V_0**. Es decir,

$$V_0 = \int \rho(W).\Theta(W).dW \qquad (2.22)$$

donde **Θ(w) = 1** cuando **f(w) = f** para cada patrón de entrada, y cero en caso contrario.

Entonces $R_o(f) = \frac{V_0(f)}{V_0}$ representa la fracción del volumen de pesos en las cuales se representa **f** en forma correcta. Por otra parte, dado un conjunto de patrones de entrenamiento **P** definidos como pares **(x_k,y_k)**, pertenecientes al dominio de la función **f**, y asumiendo que esos patrones pueden ser aprendidos por una red **M(x,w)**, definimos,

$$V_p = \int \rho(W).\prod_{k=1}^{p} I(f_w, x^k).dW$$

$$I(f_w, x^k) = \begin{cases} 1 & si \quad f_w(x^k) = f(x^k) \\ 0 & en \quad caso \quad contrario \end{cases}$$

como el volumen del espacio de pesos compatible con el conjunto de patrones de entrada **P**. Si el conjunto de patrones de entrada es pequeño puede haber más de una función $f_w \in F$ compatible con dicho conjunto. A medida que

aumentamos la cantidad de patrones de entrada disminuye el número de posibles funciones y por lo tanto el valor de $\mathbf{V_P}$ decrece. Entonces se define,

$$R_p(f) = \frac{V_p(f)}{V_p} \qquad (2.23)$$

como la fracción del espacio de pesos consistente con la función $\mathbf{f} \in \mathbf{F}$ y luego de haber aprendido **p** patrones de entrenamiento. En este caso

$$V_p(f) = \int \rho(W).\Theta_f(w).\prod_{k=1}^{p} I(f_w, x^k).dW$$

$$V_p(f) = V_0(f).\prod_{k=1}^{p} I(f_w, x^k)$$

Ahora bien, si consideramos el promedio de $\mathbf{V_P(f)}$ para todo el conjunto de patrones,

$$\langle V_p(f)\rangle = V_0(f).\left\langle \prod_{k=1}^{p} I(f_w, x^k)\right\rangle = V_0(f).g(f)^p$$
$$g(f) = \langle I(f, x^k)\rangle$$

donde **g(f)** representa la generalización de la red sobre un conjunto de patrones **P**. En otras palabras, para un dado **P**, **g(f)** da una idea de cuan bien $\mathbf{f_w}$ se aproxima a **f**. Si vemos a los patrones de entrenamiento $\mathbf{X_k}$ como una variable aleatoria **g(f)** representa el promedio de las probabilidades de que $\mathbf{f_w}$ sea igual a **f** para cada $\mathbf{X_k}$.

Si definimos $\rho_P(g)$ como una función distribución que mide la habilidad de generalizar para cada $\mathbf{f} \in \mathbf{F}$, puede deducirse que $\rho_P(g)$ es proporcional a,

$$\rho_p(g) \propto g(f)^p.\rho_0(g) \qquad (2.24)$$

este resultado es importante ya que si se conoce la distribución de **g** antes de comenzar el entrenamiento de la red, puede deducirse la cantidad de patrones de entrenamiento necesarios. Esto es, suponiendo que todos estos patrones

puedan ser aprendidos por la red independientemente del método utilizado para el entrenamiento.

2.6.3.2 GENERALIZACIÓN DEL CASO MAS DESFAVORABLE

La generalización promedio presenta algunos problemas para su aplicación. Se necesita conocer a priori la distribución $\rho_0(\mathbf{g})$ y considerar los promedios en función del espacio de pesos pero sin tener en cuenta cómo ese espacio es recorrido en virtud del algoritmo de aprendizaje. La generalización a través del caso más desfavorable computa límites en el conjunto de patrones de entrenamiento y los del problema actual. Un resultado importante en esta teoría se debe a Vapnik y Chervonenkis (1971,1982), y se refiere a la dimensión que debe tener un sistema (**VCdim**) para poder dicotomizar un conjunto $\mathbf{S} \in \{0,1\}^N$. Es fácil demostrar que para un perceptrón simple con una entrada $\mathbf{x} \in \Re^n$ el coeficiente **VCdim = n+1**.

2.6.4 LIMITES EN EL NUMERO DE UNIDADES OCULTAS EN MLP

El problema de generalización está íntimamente relacionado con el de capacidad y con el número de unidades ocultas en una red *feedforward* multicapa (Makhoul,1989) (Huang,1990,1991) (Hush,1992) (Baum,1989). La implementación de una función arbitraria que pertenece a una clase de funciones $F = \{f : x -> y / x \in \{0,1\}^n, y \in \{0,1\}^p\}$ por una red neuronal con una única capa de unidades ocultas, puede ser vista de la siguiente forma (fig. 2.10):

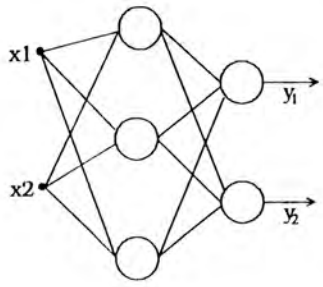

Figura 2.10 Red neuronal con una capa oculta

Sea **n** la dimensión del espacio de entrada y **m** el número de unidades de la capa oculta. Dado que la salida de cada unidad oculta es binaria, existe un hiperplano de dimensión **n-1** que separa el espacio n-dimensional de entrada en dos clases, 0 y 1. La intersección de **m** hiperplanos en el espacio dará lugar a celdas cerradas y a celdas abiertas como se indica en la figura 2.11

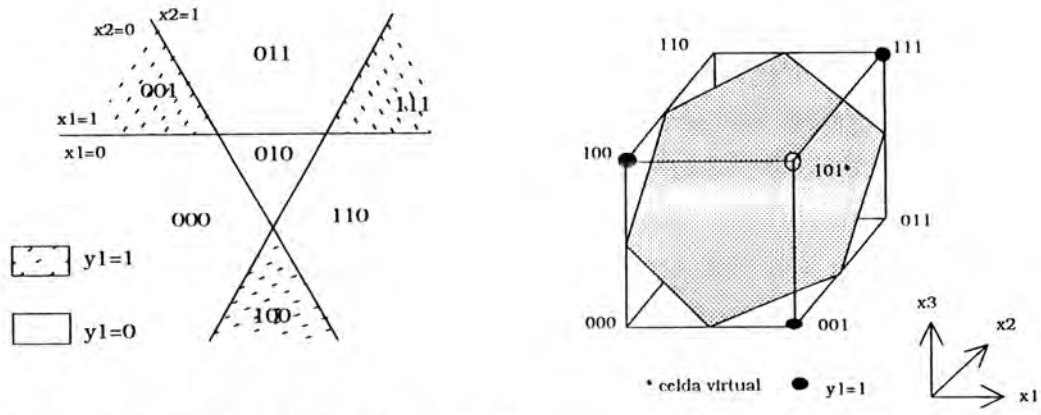

Figura 2.11a celdas de una RNA de dos capas. 2.11.b disposición de las celdas en un hipercubo.

Si cada elemento **x** de entrada es un vértice de un hipercubo n-dimensional, el problema puede ser reformulado a: cuántas unidades son necesarias para dicotomizar los **x** patrones de entrada?. La red neuronal de la figura 2.10 posee tres neuronas ocultas que dividen el espacio de entrada $\Re^2$ en celdas, la neurona de salida clasifica cada una de las áreas surgidas de la dicotomización en clases **0** y **1**.

Cada celda generada podría ser clasificada en una de dos clases por la neurona de salida. Es decir, la cantidad de celdas generadas dará una medida de la capacidad de la red neuronal. De todas formas, aún cuando cada elemento de entrada sea confinado individualmente a una celda, no está garantizado que exista un vector de pesos en la neurona de salida que pueda cumplir esta función.

La única condición que se impone para que las celdas no sean conjuntos vacios es que los hiperplanos se encuentren en *posición general.*

Definición 2.3: Un conjunto $\mathbf{S} \in \{0,1\}^N$ de **k** elementos está en posición general cuando no más de **j+1** de esos elementos se encuentran sobre un plano **j-1** dimensional $\forall \mathbf{j} \in \{\mathbf{2},\mathbf{n}\}$.

Debemos hallar el número de celdas C(m,n) formado por **m** hiperplanos en un espacio n-dimensional. Puede verse fácilmente que si el número de planos **m** es menor que el orden del espacio, cada nuevo plano cortará a todos los anteriores dividiendo a cada una de las celdas en dos. Es decir,

$$C(m,n) = 2^m \qquad m \leq n$$

Para **m > n**, cada nuevo hiperplano cortará solo **M** celdas. El número de nuevas celdas creadas por el m-ésimo hiperplano es igual al número de celdas que los otros **m-1** planos forman en su intersección con el m-ésimo. Debido a que los hiperplanos son espacios **n-1** dimensionales, los **m-1** planos anteriores forman una cantidad de celdas dada por el número combinatorio **C(m-1,n-1)**. Es decir que el m-ésimo plano agrega esta cantidad a los **C(m-1,n)** ya existentes. Entonces, el número de celdas obedece a la ecuación recursiva

$$C(m,n) = C(m-1, n) + C(m-1, n-1) \qquad (2.25)$$

Puede demostrarse por el principio de inducción que,

$$C(m,n) = \sum_{i=0}^{\min(m,n)} \binom{m}{i} \qquad (2.26)$$

Claramente, en un espacio unidimensional, cada punto divide al espacio en una celda más, **C(m,n) = C(m-1,n) + 1**. Si aumentamos el orden del espacio n en una dimensión, el índice superior de la sumatoria aumenta en uno, por lo tanto

$$C(m,n+1) = C(m,n) + \binom{m}{n+1} \qquad (2.27)$$

Como ya se dijo, aumentar en uno la cantidad de hiperplanos genera **M=C(m-1,n-1)** nuevas celdas. Si reescribimos (2.27) para **m-1** y **n**, y luego reemplazamos en (2.25)

$$C(m,n) = 2.C(m-1,n) - \binom{m-1}{n} \qquad (2.28)$$

muestra que a **C(m,n)** le falta $\binom{m-1}{n}$ para duplicar el número de nodos preexistentes. También se observa que esta cantidad es igual al número de vértices que se forman de la intersección de los planos. Justamente, la razón de que un nuevo plano no corte a todos los anteriores es la creación de vértices en el proceso. El caso más interesante se presenta para **m>>n**, ya que en la ecuación **C(m,n)** el último término de la sumatoria es dominante

$$C(m,n) \cong \frac{m^n}{n!} \qquad m >> n \qquad (2.29)$$

De la ecuación 2.26 vemos que, para el caso más interesante, el de **m>n**, $C(m,n) < 2^m$. Esto quiere decir que no todas las combinaciones binarias son posibles a la salida de la capa oculta.

La unidad de la capa de salida puede dicotomizar conjuntos que se encuentran en celdas conectadas unas a otras. Es decir, si las salidas de la capa oculta son los vértices de un hipercubo m-dimensional, pueden dicotomizarse aquellos conjuntos de celdas que pueden conectarse a lo largo de los lados del hipercubo. En este proceso las combinaciones que no pueden darse constituyen una especie de celdas virtuales (*dont care*) que ayudan a la conectividad. La figura 2.12 representa la capacidad relativa a 2^m en función de $m/2n+1$

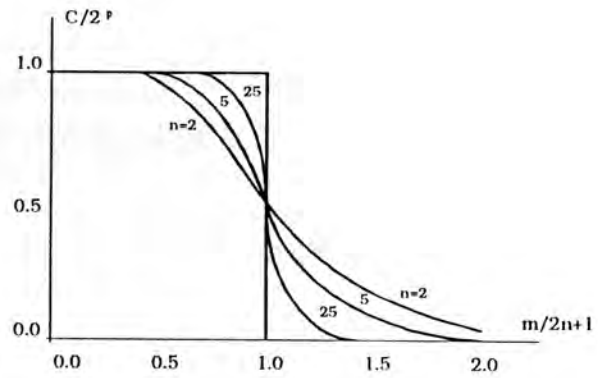

Figura 2.12 Medida de la capacidad en función de la relación m/2n+1

2.7 APROXIMACIÓN Y APRENDIZAJE

El problema de aproximar o interpolar una función continua **f(x)** a través de una función aproximante **F(x,w)**, se reduce a encontrar un número fijo de parámetros **w** que pertenecen al espacio de parámetros **P**. Donde **x** y **w** son vectores reales de dimensión **n** y **m** respectivamente. Dado **F**, se debe encontrar el conjunto **w** que aproxima de la mejor forma a **f** sobre un conjunto de muestras o ejemplos de dicha función (Cybenko,1988) (Cotter,1990) (Geva,1992) (Funahashi,1989) (Poggio,1990) (Liu,1993) (Sanner,1992) (Stinchcombe,1989). En realidad, en este problema pueden distinguirse tres aspectos fundamentales definidos por,

- qué funciones **f** pueden ser implementadas y por cuáles **F**
- qué algoritmo usar para hallar los parámetros **w** óptimos
- cómo implementar la función aproximante en hardware paralelo (analógico, preferentemente).

Podemos definir, entonces, el problema de la aproximación a través de una función distancia en una norma inducida L^p dada por (Cotter,1990),

$$d(f(x),F(x,w)) < \varepsilon \qquad \|f(x)-F(x,w)\|_{L^p} < \varepsilon \qquad (2.30)$$

donde ε es un valor pequeño arbitrario, y $F(x,w):\Re^n \rightarrow \Re$ depende continuamente de **w**.

Los teoremas de Stone-Weierstrass y de Lusin (Cotter, 1990) definen qué clases de funciones **f** pueden ser aproximadas y qué requisitos deben cumplir las funciones aproximantes **F**. Si el dominio **D** es un compacto en $\Re^n$, y $\Im$ el conjunto de funciones continuas sobre D que satisfacen que,

i) la función constante **f(x)=1** pertenece a $\Im$
ii) para dos puntos $\mathbf{x_1} \neq \mathbf{x_2}$ en **D**, existe una **f** en $\Im$ tal que $\mathbf{f(x_1)} \neq \mathbf{f(x_2)}$
iii) la suma **f+g** y producto **f.g** de funciones es cerrada en $\Im$

entonces $\Im$ es denso en **C(D)**, el conjunto de funciones continuas en **D**; y existe **F** que aproxima a **f** según (1). Puede decirse, también, que existe una sucesión convergente tal que

$$\lim_{n\to\infty} \left\| f(x) - F_n(x,w) \right\|_{L^p} = 0$$

El teorema de Lusin incluye cierto tipo de funciones no continuas que quedan excluidas del teorema de Stone-Weierstrass. Esto significa que si **f(x)** es una función medible, $\int_{\Re^n} |f(x)|^p \, dx < \infty$, y acotada en **D**, salvo en un número contable de puntos, entonces, dado un $\delta > 0$ existe una función **F** sobre **D** tal que la medida donde $\mathbf{F} \neq \mathbf{f} < \delta$. En otras palabras, el volumen total de las esferas en $\Re^n$, necesario para incluir los puntos donde $\mathbf{F} \neq \mathbf{f}$ es menor que δ.

No solo es interesante aproximar **f(x)** por una función **F(x,w)** sino, además, encontrar la mejor aproximación posible dentro del conjunto de funciones $\Im$. La mejor aproximante **F*** corresponderá a un conjunto óptimo de parámetros **W*** de manera tal que,

$$d\left(f(x), F(x,w^*)\right) \leq d\left(f(x), F(x,w)\right) \qquad (2.31)$$

para todo w en el espacio de parámetros (Liu 1993).

Los modelos neuronales típicos, previamente analizados en este capítulo, pueden definirse como ejemplos particulares que siguen las definiciones (2.30) y (2.31)

- F(x,w) = X.W

representa el caso lineal correspondiente a una red sin capa oculta.

- $F(x,w) = \sum_{i=1}^{m} w_i \phi_i(x)$

corresponde a una red con una capa de unidades ocultas. En este caso la función **F** es lineal en las bases **φi**. La interpolación Spline y las expansiones de series de polinomios ortogonales, son de la misma forma.

- $F(x,w) = \sigma\left(\sum_n w_n . \sigma\left(\sum_j v_j . (\ldots)\right)\right)$

este caso corresponde al modelo típico del Perceptrón multicapa, con función de activación σ y vector de parámetros **W=(wn, vj, ...)**.

En general, cada esquema de aproximación tiene un algoritmo específico para encontrar el vector de parámetros óptimos. Una de las soluciones más utilizadas consiste en definir un funcional del error de la salida en función del vector de parámetros, y luego navegar en esa superficie hacia el punto de mínimo error tomando la dirección del gradiente o gradiente conjugado. Este algoritmo puede ser muchas veces ineficiente, pero tiene aplicabilidad a un amplio rango de problemas.

De todas maneras, se quiere puntualizar que los modelos neuronales ejemplificados pueden ser considerados como métodos específicos de aproximación de funciones. Es decir, el problema de aprendizaje en una red neuronal de este tipo puede ser tratado desde el punto de vista de la teoría de aproximaciones clásica.

Poggio (Poggio et al, 1990) plantea el problema de aproximación como un problema de generación de un mapa de entrada/salida a través de muestras (ejemplos). Este caso es propicio para la aplicación de la teoría de regularización a la reconstrucción de superficies. Hay una conexión directa entre esta teoría y el método de interpolación con Funciones de Base Radial (RBF), que puede ser implementado por medio de redes neuronales multicapa.

En este contexto, el aprendizaje se traduce como el algoritmo que permite encontrar un conjunto de parámetros tal que permita a nuestra función aproximar una función dada por sus muestras y generalizar en puntos

intermedios. Ya hemos definido en la sección 2.6.3 la generalización como la estimación de la salida en los puntos de entrada donde no se poseen muestras.

En general, el problema del aprendizaje para aproximación de una superficie es *ill-posed* (i.e. no biunívocamente determinado) debido a que no es posible reconstruir un mapa en zonas donde no existen datos. Además los datos poseen, en general, ruido. Para mejorar el enfoque del problema deben hacerse ciertas suposiciones. Algunas de ellas serán basadas en el conocimiento que tengamos sobre el problema, tal como mapeo lineal, rango positivo, cierto dominio finito en el espacio o la frecuencia, etc. Pero, además, pueden hacerse otras suposiciones basadas en condiciones más generales acerca de las funciones que modelan un mundo real, tales como redundancia de los datos, y suavidad de las mismas.

Las técnicas de regularización explotan estas características de las funciones (Tikonov, 1977) (Poggio,1985). La teoria estandar de regularización aplica el cálculo variacional a la minimización de un funcional de dos términos,

$$H(F) = \sum_i \left(f(x_i) - F(x_i, w) \right)^2 + \lambda \| PF \|^2 \qquad (2.32)$$

el primer término considera la distancia entre los datos y la función aproximante, el segundo representa la medida del costo de un funcional asociado al conocimiento a priori que se posee de la función. Si utilizamos, para P, el operador diferencial, la minimización del funcional H nos lleva a la ecuación de Euler-Lagrange (Shilov,1965),

$$\hat{P}\, PF(x,w) = \frac{1}{\lambda} \sum_{i=1}^{N} \left(f(x_i) - F(x_i, w) \right) \delta(x - x_i) \qquad (2.33)$$

en el cual $\hat{P}$ es el adjunto del operador diferencial P. La ecuación (2.33) es una ecuación diferencial parcial cuya solución puede ser escrita por la función de Green del operador diferencial $\hat{P}\,P$, es decir la ecuación diferencial queda,

$$\hat{P}\, PG(x, x_i) = \delta(x - x_i)$$

las δ convierten la transformación integral en una sumatoria de N términos

$$f(x) = \frac{1}{\lambda}\sum_{i=1}^{N}(f(x_i) - F(x_i, w))G(x, x_i) \qquad (2.34)$$

La ecuación (2.34) dice que la solución se encuentra en un espacio N-dimensional de funciones. Una base de este espacio la constituyen las funciones G(x,xi), donde las **xi** son las muestras. Los coeficientes $c_i = (f(x_i) - F(x_i, w))/\lambda$ pueden obtenerse evaluando el sistema lineal $c = (G+\lambda I)^{-1} f$. Entonces,

$$f(x) = \sum_{i=1}^{N} c_i G(x, x_i) \qquad (2.35)$$

Como el operador $\hat{P}$ P es autoadjunto, su función de Green es simétrica y con autovalores reales. G es traslacionalmente invariante, y en el caso particular de que sea, además, rotacionalmente invariante, será una función radial

$$f(x) = \sum_{i=1}^{N} c_i G(\|x - x_i\|) \qquad (2.36)$$

Diferentes operadores P dan lugar a conocidas funciones de Green para aproximación e interpolación, entre ellas las Splines multidimensionales, las *thin-plate* Splines, etc.

Pero la función base en la que concentraremos la mayor atención es la solución de la ecuación diferencial $\sum_{n=0}^{\infty}(-1)^n \frac{\sigma^{2m}}{m!2^m}\nabla^{2n} G(x) = \delta(x)$, que utilizando técnicas de transformada de Fourier nos da la ecuación de Green

$$G(x) = c\, e^{-(x-x_i)^2/\sigma^2} \qquad (2.37)$$

2.8 REDES NEURONALES BASADAS EN FUNCIONES DE BASE RADIAL GAUSSIANAS

La estructura de una red neuronal RBF gaussiana (fig. 2.13) consta de dos capas. La capa oculta computa la función base de la ecuación (2.37), la capa de salida es una transformación lineal de $\Re^n \rightarrow \Re$ del espacio de salida de la capa oculta en el campo escalar (Chen,1991) (Liu,1993) (Mees,1992) (Poggio,1990a,b) (Sanner,1992) (Stinchcombe,1989). Es decir,

$$G(x) = \sum_{j=1}^{n} c_j e^{-(x-x_j)^2/\sigma j^2} \qquad (2.38)$$

en la cual **x** y **xj** son vectores que pertenecen al espacio m-dimensional de entrada, **xj** representa la media, y σj la desviación estandard de cada gaussiana. La extensión de este modelo a salidas r-dimensionales es trivial.

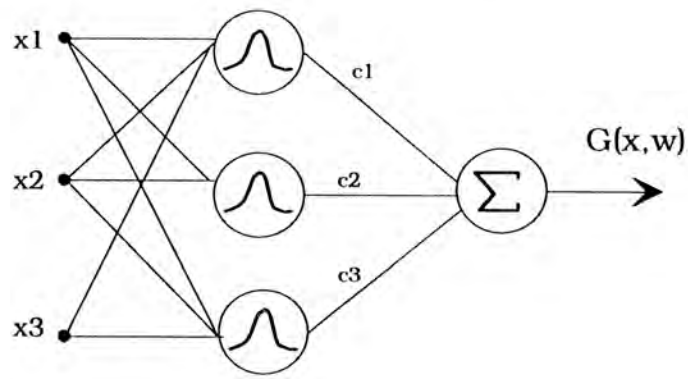

Figura 2.13 Red Neuronal Gaussiana de dos capas

Desde el punto de vista de la teoría de aproximación, esta red tiene tres propiedades interesantes,

i) Puede aproximar en forma arbitraria cualquier función continua multi-variable sobre un dominio compacto, si se utiliza una cantidad suficiente de unidades.

ii) La función de salida **G** es lineal en el vector de parámetros **c**, en consecuencia, tiene la propiedad de *mejor aproximador*. Los parámetros **c** pueden ser hallados fácilmente usando métodos de regresión.

iii) La solución es óptima en el sentido que minimiza el funcional H que incluye una medida de las "oscilaciones" donde no existen datos. Esto elimina soluciones que interpolan perfectamente las muestras pero oscilan en los lugares donde éstas faltan.

Como fue dicho en sec. 2.7 Funahashi (Funahashi, 1989) y Cotter (Cotter, 1990) extienden la clase de funciones que pueden ser aproximadas a funciones no-continuas medibles.

2.8.1 DIMENSIONAMIENTO Y ELECCIÓN DE LOS PARÁMETROS DE LA RED

Si bien el modelo neuronal RBF gausiano descripto puede representar una amplia clase de funciones multidimensionales, en las aplicaciones pueden existir problemas que hagan esta realización impráctica.

- la complejidad de la red (número de centros) es del orden del número de datos de entrada.
- la probabilidad que el conjunto de datos sea *ill-conditioned* aumenta con el número de los mismos.
- el ruido en los datos fuerza oscilaciones indeseadas.

Estos problemas pueden ser superados haciendo ciertas extensiones a la teoría de regularización estandar (Tikonov,1977) (Poggio,1985,1990a,b). La función **f(x)** puede ser aproximada por la expansión de una base finita

$$\tilde{f}(x) = \sum_{i=1}^{n} c_i g_i(x) \qquad (2.39)$$

donde las **gi(x)** forman un conjunto linealmente independiente. Los coeficientes **ci** son elegidos de manera tal que (2.39) se aproxime a la solución verdadera. Si se hace **n=N**, y **gi(x)** igual a la función de Green de la ec. (2.36), la función coincide con la solución exacta.

El problema del mal condicionamiento de los datos de entrada puede ser mejorado aplicando sobre ellos una transformación lineal. Esta transformación podría, en el mejor de los casos, lograr un conjunto ortogonal. En realidad, puede utilizarse un subconjunto (ortogonal) que genere el mismo espacio que generan los datos. La norma euclídea considerada en (2.36) se transforma en una norma *pesada* debido a la transformación lineal,

$$\|x-x_i\|_W^2 = (x-x_i)^T W^T W(x-x_i)$$

En el caso particular de que **W** sea diagonal, las componentes de **x** y **xi** pueden ser factorizadas y las entradas de **W** actúan como diferentes **σi** en cada componente. En este caso, el funcional **Hw** es,

$$H_W(F) = \sum_i (f(x_i)-F(x_i,w))^2 + \lambda \|PF\|_W^2$$

Esto significa que la regularización se realiza en el espacio transformado **Wx**. La función de Green asociada será $G\left(\|x\|_W^2\right)$ y la función que aproxima a **f** es,

$$\tilde{f}(x) = \sum_{k=1}^{n} c_k G_k\left(\|x-x_k\|_W^2\right) \qquad (2.40)$$

Elección de los centros y la matriz **W:** (Poggio, 1990a)

En el caso más general, las incógnitas son: **n** coeficientes **ck**, **mxn** centros **xi** y **mxm** coeficientes en la matriz **W**. Si se impone la condición de que la elección de estos parámetros debe ser óptima en el sentido de minimizar $H\left[\tilde{f}\right]$.

$$\frac{\partial H\left[\tilde{f}\right]}{\partial c_k} = 0 \qquad \frac{\partial H\left[\tilde{f}\right]}{\partial x_k} = 0 \qquad \frac{\partial H\left[\tilde{f}\right]}{\partial W} = 0$$

Para hallar estos parámetros pueden aplicarse métodos iterativos, tales como gradiente descendente, gradiente conjugado, *simulated annealing*, etc. La

ecuación de obtención de los **ck** es cuadrática y tiene la convergencia asegurada por cualquiera de estos métodos.

$$\frac{\partial H\left[\tilde{f}\right]}{\partial c_k} = -2\sum_{k=1}^{n}\Delta_k G\left(\|x_l - x_k\|_W^2\right) \qquad (2.41)$$

$$\frac{\partial H\left[\tilde{f}\right]}{\partial x_k} = 4\,c_k\sum_{k=1}^{n}\Delta_k G'\left(\|x_l - x_k\|_W^2\right) W^T W(x_l - x_k) \qquad (2.42)$$

$$\frac{\partial H\left[\tilde{f}\right]}{\partial W} = -4\,W\sum_{k=1}^{n} c_k \sum_{l=1}^{N}\Delta_l G'\left(\|x_l - x_k\|_W^2\right) Q_{l,k} \qquad (2.43)$$

donde se ha definido $\Delta_l = f(x_l) - F(x_l, w)$, y $Q_{l,k} = (x_l - x_k)(x_l - x_k)^T$

Las interpretaciones que se desprenden de estas últimas ecuaciones son las siguientes:

- la ecuación (2.41) corrige sobre la suma total de ejemplos proporcionalmente al producto entre el error **Δi** y la actividad de la unidad, medida por la función de Green para ese ejemplo.

- la ecuación (2.42) representa un algoritmo de agrupamiento (*clustering*). Si se aplica gradiente descendente a esta ecuación, los centros **xk** se mueven hacia la región central de la mayoría de los datos agrupados en esa zona.

- en la ecuación (2.43) **Qi,k** representa la correlación de los ejemplos relativos a un dado centro **xk**. Bajo ciertas condiciones de simplificación (Poggio, 1990) en la matriz **W**, la ecuación converge a una matriz cuyas filas son los autovectores de **Q** asociados a los autovalores mas pequeños. Es decir, esta matriz genera el espacio ortogonal al generado por el análisis de componente principal del conjunto de datos (Chen,1991). Es decir que esta ecuación permite lograr una reducción de la dimensionalidad del espacio de entrada.

2.9 APLICACIÓN DE RBF GAUSSIANAS A LA APROXIMACIÓN DE UNA CURVA EN UN PLANO.

La teoría de aproximación de funciones a través de redes neuronales con unidades de cómputo cuya función de activación es gaussiana, se utilizó para la interpolación de una curva en un plano bidimensional.

La aplicación planteada, según se muestra en la figura 2.14, considera el mapa de un espacio bidimensional en el cual los sectores oscuros representan obstáculos. El gráfico podría representar la imagen obtenida por una cámara de visión fija de un ambiente en el cual se desea planificar una trayectoria. En consecuencia se marcan **O** y **D** como puntos de origen y destino, respectivamente, de dicha trayectoria (Foux,1993) (Gouzenes,1984) (Lozano-Perez,1979) (Zelinsky,1992).

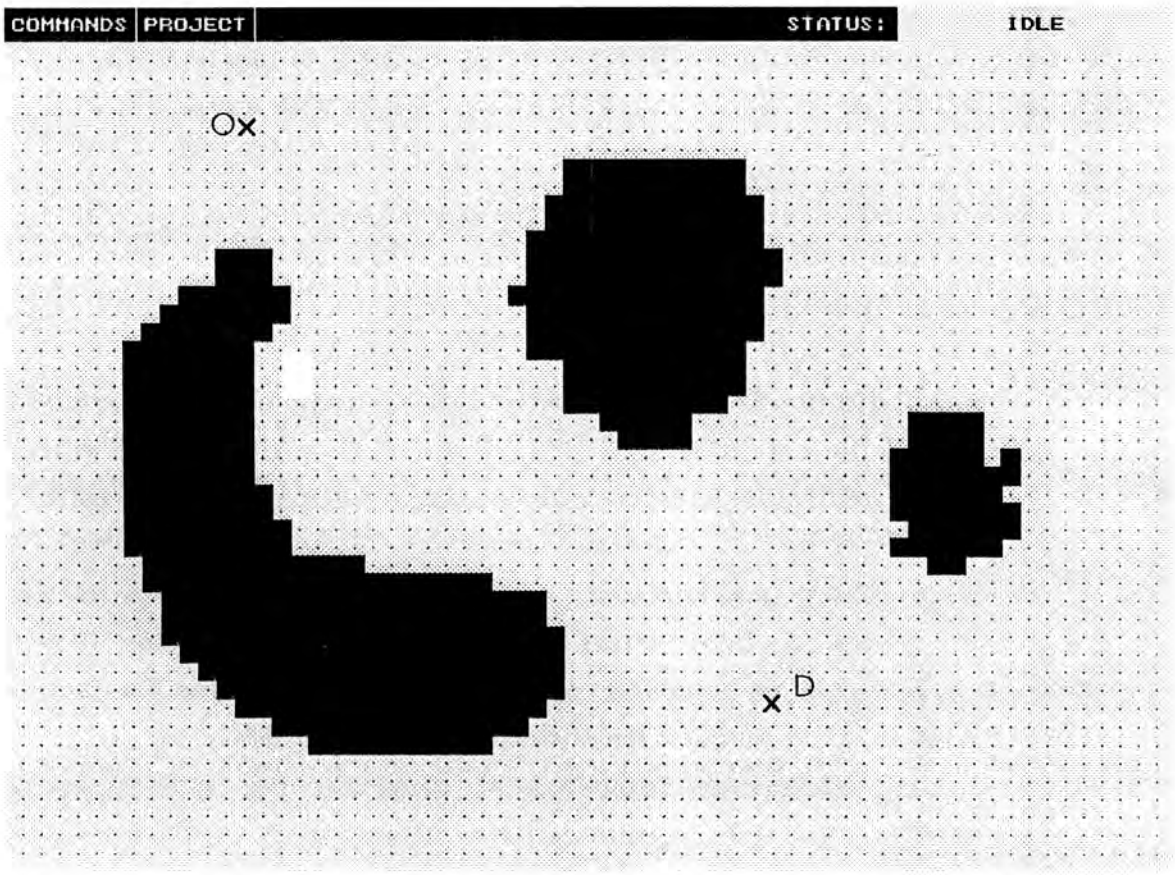

Figura 2.14 Mapa bidimensional con obstáculos, grilla de tamaño fijo

La trayectoria la navegación segura entre O y D. Para ello se define una superficie de gradiente de potencial (Barraquand,1992) (Slack,1993) (Cancelo,Mayosky,1995). Una superficie de este tipo se construye generando una grilla discreta del espacio, como muestra la figura 2.14, y asignando un valor de potencial a cada punto del espacio según un criterio de optimización dado. En este caso, se ha elegido un funcional combinado de distancia en el cual se desea minimizar el recorrido entre **O** y **D** pero manteniéndose alejado de los obstáculos. El punto móvil **M** debe partir desde **O** y llegar a **D** recorriendo la superficie de potencial, siguiendo el camino de gradiente máximo (negativo).

Primeramente, se genera una grilla de paso fijo como se indica en la figura 2.14. A partir de ella se construye un árbol o esqueleto de distancias a los objetos. Se considera que los objetos (y sus interiores) están a distancia cero, los puntos de la grilla más cercanos a ellos, a distancia uno y así siguiendo. Esto crea un esqueleto de distancias como se ve en la figura 2.15 (Cancelo,Mayosky,1995).

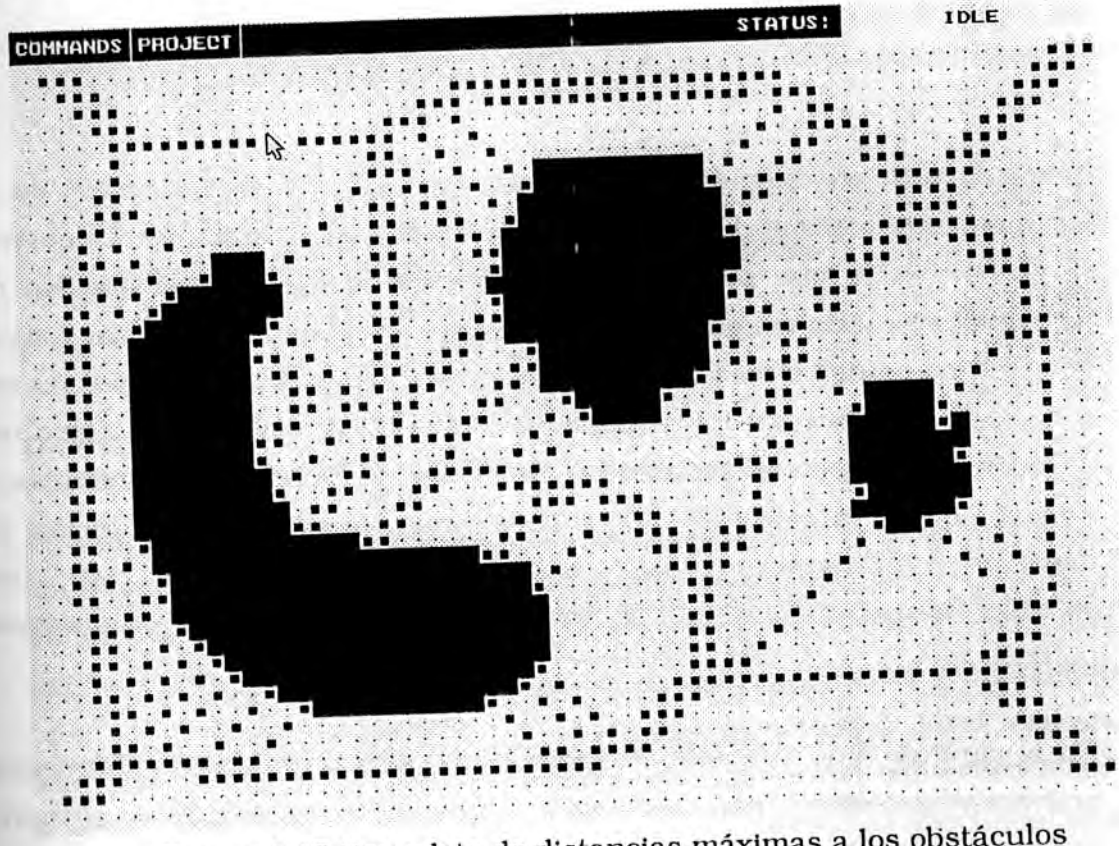

Figura 2.15 Esqueleto de distancias máximas a los obstáculos

La superficie de potencial se crea utilizando el algoritmo de Dijtra (Dijkstra, 1959) o transformada de distancia (Zelinsky, 1992). El nodo de destino se pone a potencial cero y luego se incrementa el potencial según un costo fijo a los vecinos. En nuestro caso se aplica esta función, primeramente desde el destino, a todos los puntos del esqueleto de distancias. Esto asegura que cualquier punto del esqueleto posea un potencial menor que el de los vecinos que no están en él. A continuación, se calcula el potencial en el resto de los puntos de la grilla.

La trayectoria se genera siguiendo el camino de gradiente máximo (negativo) partiendo desde **O**. Debido a la construcción realizada, el camino alcanza rápidamente el esqueleto de distancia y luego sigue el mismo hasta el punto **D**. Debido al la naturaleza discreta de la grilla, se obtienen solo muestras del camino de **O** a **D**. Para interpolar, se utilizan unidades gaussianas con funciones de activación descriptas por la ecuación (2.38). Las gaussianas fueron centradas en puntos de la grilla sobre las muestras obtenidas. El ancho de las mismas se obtuvo utilizando la ecuación

$$\sigma_i^2 = \frac{1}{N_i} \sum_{x \in \theta_i} (x - x_i)^T (x - x_i)$$

donde θ_i y N_i representan los elementos y el número de elementos de cada agrupamiento (*cluster*) de entrada a cada una de las gaussianas. La curva que se obtiene es la que se indica en la figura 2.16. Se observa que las redes neuronales gaussianas pueden ser útiles para aproximar o interpolar funciones de este tipo en dos o más dimensiones. En el caso de navegación en un medio con obstáculos pueden introducirse variantes interesantes, tales como radios de curvatura máximos permitidos, funciones de potencial combinadas, etc. Además puede construirse una grilla de tamaño variable en función de la densidad de obstáculos del ambiente (fig. 2.16), lo cual reduce considerablemente el número de puntos y acelera los cálculos de distancias y potenciales.

En el capítulo 4 se propone el diseño de redes neuronales con funciones de activación gaussianas para circuitos VLSI analógicos aptos para el cómputo de funciones como las analizadas en esta aplicación. Estos circuitos, aun cuando

pueden ser utilizados para propósitos generales, son particularmente interesantes en tareas de visión robótica debido a la complejidad de la misma y el alto grado de paralelismo involucrado.

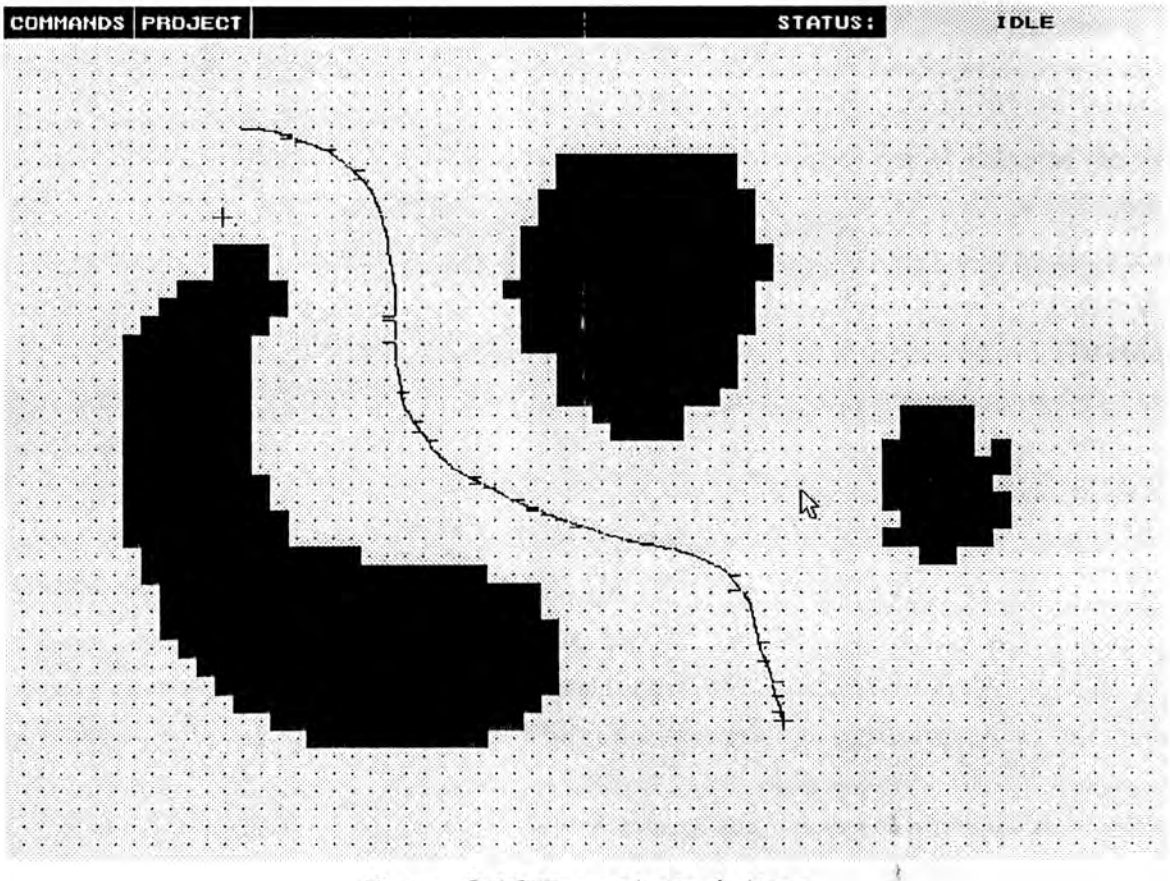

Figura 2.16 Trayectoria óptima

DISCUSION:

En este capítulo se han planteado los tres problemas claves de una estructura neuronal: representación, aprendizaje e implementación. Además se ha hecho un análisis teórico sobre los los dos primeros, enfocado a los modelos estáticos multicapa. Este enfoque pone de manifiesto las propiedades y limitaciones de estos modelos, tanto a nivel neuronal interno como a nivel del sistema neuronal. Los estudios sobre generalización y capacidad son escenciales para la generación de un sistema neuronal capáz de representar adecuadamente un

problema dado. La implementación de un sistema neuronal debe comprender primeramente la generación de una neurona como la unidad constitutiva de dicho sistema. Debido a ello, en el próximo se estudian las posibles soluciones al problema neuronal tanto desde el punto de vista de la unidad como del sistema.

CAPITULO 3

NEUROCOMPUTACIÓN

3.1 INTRODUCCIÓN

Los modelos neuronales artificiales están basados tanto en el estudio de las redes neuronales biológicas como en consideraciones teórico-prácticas convenientes para la representación del mundo físico. Si bien el uso de estos modelos se ha extendido a muy diversas ramas de la ciencia, el estudio del presente capítulo y sus conclusiones, se refieren a la aplicación de sistemas neuronales en ingeniería; aún cuando estas propiedades pueden ser extensibles a otras áreas.

Los avances teóricos en redes neuronales han sido significativos, particularmente en los últimos diez años. Si bien, la investigación en el tema ha sido continua desde las épocas de McCulloch-Pitts y Hebb, no cabe duda que ha recibido un notable impulso luego de la aparición del modelo de Hopfield y el algoritmo de aprendizaje basado en retropropagación del error, para redes neuronales estáticas multicapa. Las aplicaciones de la teoría en ingeniería comprenden tareas de clasificación, procesamiento de señal, modelado de procesos complejos, optimización, control, visión, etc. Aquí, la teoría de redes neuronales ha dado lugar a un nuevo enfoque o punto de vista sobre la manera de resolver estos problemas. Cabe aclarar, que no es obvio que los modelos neuronales sean más eficientes que otras teorías en la resolución de un problema en particular. Son pocos los casos en que se puede demostrar formalmente que un modelo neuronal es mejor para la resolución de una clase específica de problemas. Muchas veces debemos conformarnos con acumular evidencia comparativa entre diversas técnicas aplicadas a casos particulares de problemas de un mismo tipo o clase. De todas maneras esta situación es común a otras áreas de la teoría en problemas de ingeniería. Sin embargo, existe una propiedad de los modelos neuronales comparativamente superior, la capacidad de computo paralelo que redunda en un elevado rendimiento.

Ya fue dicho durante los capítulos 1 y 2 que amen de la diversidad de modelos neuronales existentes, existen dos propiedades unificantes: el paralelismo masivo de unidades computacionales simples y la adaptatividad de los parámetros del modelo en función de las características de los datos de entrada. Una dada arquitectura neuronal, se caracteriza por: la cantidad de funciones que puede representar, su algoritmo de aprendizaje y la forma en que esta red neuronal ha sido construida. En el capítulo 2 se analizó, la capacidad representativa de una clase de modelos neuronales y sus algoritmos de adaptatividad. En este capítulo se realiza un análisis sobre la implementación paralela de redes neuronales artificiales.

Este es, sin duda, un tema fundamental en la investigación sobre modelos neuronales, ya que la construcción de los mismos en hardware posibilita el pleno aprovechamiento de la capacidad de cómputo paralela de la red. Actualmente, más del 90% de las redes neuronales existentes son simuladas en computadoras convencionales. Estas simulaciones han servido para desarrollar la teoría de los diferentes modelos, descubrir nuevos enfoques y algoritmos para la resolución de un problema o una clase de problemas y dar solución a aplicaciones específicas de la ingeniería. Sin embargo, las redes neuronales son particularmente atractivas cuando se aplican a problemas complejos, para los cuales los métodos tradicionales sólo dan soluciones pobres, y en los que se puede hacer valer las características adaptivas de dicha red. Pero justamente, estos problemas complejos resueltos con métodos neuronales necesitan una gran potencia de cómputo. Simular los modelos de la red neuronal en una computadora convencional sólo tiene el beneficio de la flexibilidad del sistema de cómputo pero se torna rápidamente ineficiente en función de la complejidad del problema. En las secciones subsiguientes se analizan diferentes formas de generar dispositivos y sistemas de cómputo neuronal.

3.2 DISPOSITIVOS Y SISTEMAS NEURONALES

Implementar un sistema neuronal que sea capaz de representar un dado modelo neuronal, implica generar una arquitectura paralela de las unidades computacionales de dicha red. Cada unidad computacional se convierte en un procesador neuronal. En la simulación de modelos neuronales en computadoras los procesadores neuronales son simulados por uno o más microprocesadores convencionales. En una computadora neuronal debe existir un arreglo paralelo de procesadores especialmente orientados para la computación de un dado algoritmo. Aquí surgen varios interrogantes en el diseño de los procesadores neuronales en función de: la tecnología digital o analógica a utilizar, la flexibilidad que debe tener el sistema para adaptarse a distintos modelos neuronales y a distintos tamaños de una red; la integración que debe tener el sistema con señales sensoriales externas o con usuarios.

Las distintas elecciones hechas en función de estos interrogantes afectan en forma directa propiedades del sistema tales como:

- precisión de la función de cómputo,
- ancho de banda de procesamiento,
- forma de almacenamiento de los parámetros adaptativos,
- interconexión y comunicación entre unidades neuronales,
- consumo de potencia,
- tipo de arquitectura,
- capacidad,
- costo.

Básicamente, puede pensarse que un procesador neuronal elemental es una unidad RISC (Reduced Instruccion Set Controller) digital o analógica. Si no se requiere mayor flexibilidad, cada neurona computa un algoritmo fijo, consistente en multiplicaciones, sumas y la aplicación de una función no-lineal. Sin lugar a dudas, la implementación de redes neuronales no sería posible si el gran avance que ha tenido la tecnología de integración en gran escala (VLSI). Actualmente, los procesos tecnológicos más avanzados permiten integrar algunas decenas de millones de transistores en un solo dispositivo. Se

espera que con los materiales y procesos actuales el límite de la tecnología pueda ser mejorado en al menos dos órdenes de magnitud en la próxima década. Lo cual constituye un pronóstico promisorio para alcanzar un número considerable de neuronas por cada dispositivo. Esto es fundamentalmente importante debido a que, en general la interconexión entre neuronas es mucho mayor que con las señales sensoriales externas.

Los dispositivos VLSI permiten explotar el paralelismo intrínseco existente en una red neuronal generando múltiples unidades idénticas, con una topología regular y simplificando el diseño del mismo. Además, proveen un mecanismo eficiente y robusto para la implementación de resultados teóricos sobre modelos neuronales e investigación sobre alta densidad de componentes y transistores de canal corto.

3.3 ANALÓGICO vs. DIGITAL

Un punto fundamental en la generación de sistemas neuronales es la unidad de procesamiento neuronal. Estas unidades tienen una importancia mucho mayor que en una computadora secuencial, debido a que las neurocomputadoras aumentan su potencialidad aumentando el número de unidades. Además, la mayoría de la unidades neuronales incluyen la memoria de la neurocomputadora. Como se dijo, una unidad de computo neuronal es un procesador digital o analógico que realiza un conjunto reducido de operaciones simples. La física de dispositivos electrónicos permite representar excelentemente los modelos matemáticos neuronales.

Dentro de las tecnologías electrónicas disponibles, sin dudas, la CMOS (*Complementary Metal Oxide Semiconductor*) se ha adueñado del tema en forma casi excluyente. Los motivos fundamentales de esta preferencia son, entre otros: la gran versatilidad que posee la tecnología CMOS, posibilitando el diseño de múltiples funciones; conversión corriente a tensión (y viceversa) con un solo transistor; bajo consumo de potencia; dos regiones de funcionamiento en modo activo, como fuente de corriente en la zona de saturación y como

resistencia variable en la región de funcionamiento ohmico o lineal. Conmutación de alta velocidad que permite trabajar en circuitos digitales con frecuencias de reloj superior a 100 MHz.

El apéndice B describe al transistor MOS en tres regiones de funcionamiento: deplexión, inversión débil e inversión fuerte. La curva de corriente de drenador Id en función de la tensión compuerta-substrato Vgs sigue inicialmente una ley exponencial. Este modo de funcionamiento se denomina sub-umbral y se extiende hasta valores Vgs ~ 0.7v. Para un rango de valores de Vgs más elevados Id disminuye su respuesta exponencial frente a la entrada Vgs hasta que para Vgs > 2.Vt ley que las relaciona es aproximadamente cuadrática.

Una unidad de procesamiento neuronal puede ser realizada utilizando los modelos de la electrónica analógica, digital o una combinación de ambas. Cada una de ellas tiene ventajas y desventajas comparativas que se detallarán a continuación.

3.3.1 NEURONAS ANALOGICAS

Los modelos analógicos son esencialmente compactos cuando representan las funciones aritméticas neuronales tales como: multiplicación, suma, y ciertas funciones no lineales (3.1) y (3.2).

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad (3.1)$$

$$y_k = \varphi(u_k - \theta_k) \qquad (3.2)$$

En el procesamiento de señales analógicas pueden usarse tensiones o corrientes. Si bien, salvo en el caso de conductancias o resistencias de valor infinito, toda tensión lleva una corriente asociada, la señalización puede ser más fácilmente manipulada en función de una de ellas. Como se verá expresamente en el capítulo 4, la operación suma puede ser representada,

siguiendo la ley de Kirchoff, simplemente por un nodo al cual llegan fuentes de corriente.

Un producto de términos puede ser modelado por medio de un amplificador de transconductancia con ganancia variable. En este caso existe una conversión de la señal de una tensión de entrada a una corriente de salida.

El modelo electrónico analógico de la función no-lineal φ depende, obviamente, del tipo de función requerida. Si la función es de tipo sigmoidea o umbral con discontinuidad (función signo), puede ser representada por un amplificador diferencial. La pendiente de la función sigmoidea puede ser ajustada variando la ganancia de tensión de dicho amplificador. En el caso límite de la función umbral esta ganancia puede tomar valores muy grandes usando técnicas de conmutación como en las compuertas lógicas.

La electrónica analógica es muy compacta comparada con la digital, las funciones mencionadas pueden ser realizadas con un número mínimo de transistores en un rango de 1 a pocas decenas. Tienen un ancho de banda de procesamiento mucho mayor que el de la electrónica digital, especialmente cuando se los utiliza en el modo de corriente.

3.3.1.1. ARQUITECTURA DE NEURONAS ANALOGICAS

Según la ecuación (.1) cada salida y es función del producto interno de los vectores **x** y **w**. Si extendemos el espacio de salida a un vector m-dimensional, el vector **y** es el resultado del producto de una matriz de pesos w por un vector de entradas **x** (3.3). Esta ecuación puede ser representada por una matriz de sinapses como indica la figura 3.1.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} w_{11} \cdots\cdots w_{1n} \\ \vdots \quad \cdots \quad \vdots \\ w_{m1} \cdots\cdots w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad (3.3)$$

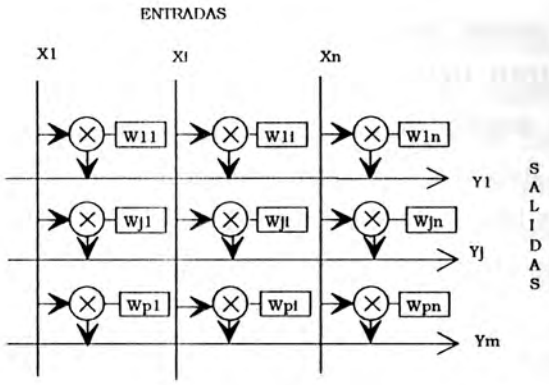

Figura 3.1 Matriz sináptica

Una red neuronal monocapa cuya dimensionalidad de entrada es **n** y de salida **m**, debe realizar n.m multiplicaciones (y sumas) sinápticas, y sólo m transformaciones no-lineales. Es decir, es crítico el tamaño y complejidad del circuito multiplicador, ya que consume un área importante del VLSI. La figura 3.2 representa un multiplicador propuesto por Bult y Wallinga (Burt, 1987). Este multiplicador provee una corriente de salida:

$$I_{SAL} = \frac{2 I_a I_b}{\left[k(V_2 - 2V_t)^2\right]} = \frac{I_a I_b}{2 I_0} \qquad (3.4)$$

donde Ia e Ib representan fuentes externas de corriente en forma de modo común y modo diferencial. En este circuito los transistores trabajan en zona de saturación post-umbral (ley cuadrática), en la ecuación (3.4) k representa la transconductancia y Vt la tensión de umbral del MOS.

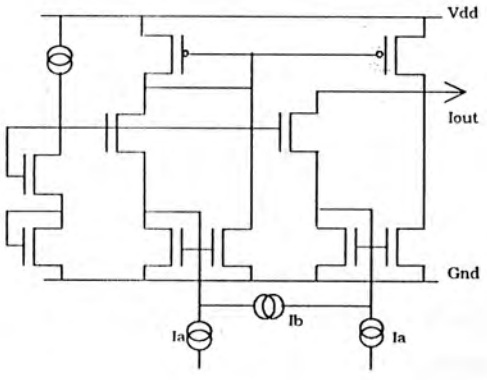

Figura 3.2 multiplicador de Bult-Wallinga

Este circuito posee buen rango dinámico lineal y gran ancho de banda. Los autores aseguran tiempos de subida menores a 50 nseg, *slew-rate* de .28μ A/nseg y un ancho de banda superior a 5 MHz. Sin embargo el circuito necesita dos fuentes externas, idénticas, de corriente y una referencia I0 común a todas las celdas multiplicadoras. En el próximo capítulo se analizaran en detalle otros circuitos multiplicadores analógicos.

Otro punto clave en el diseño de una red neuronal en VLSI es el conexionado entre neuronas. Existen tantas conexiones como productos en el circuito, es decir m.n para una red de una capa plenamente conectada. Este conexionado metálico consume gran cantidad de espacio del VLSI y complica en gran medida el diseño, ya que las conexiónes se entrecruzan entre todas las entradas de una capa con todas las neuronas de dicha capa. La redes neuronales digitales pueden solucionar en parte este problema generando buses compartidos en forma de redes de datos y utilizando algún mecanismo de multiplexado.

3.3.2 NEURONAS DIGITALES

La tecnología CMOS digital aporta una mayor flexibilidad para el diseño de unidades de procesamiento neuronal. Estas unidades pueden diseñarse, verdaderamente, como un procesador RISC elemental, el cual posea una unidad aritmética y un multiplicador. Las funciones no-lineales pueden ser implementadas dentro del dispositivo o con la ayuda de tablas en memoria.

La figura 3.3 representa un esquema simplificado de la arquitectura de una unidad de procesamiento neuronal digital. Cada unidad de procesamiento es conectada a dos buses para entrada y salida de datos. Estos datos deben ser almacenados en registros internos. Además la lógica de comunicaciones debe poseer un par de registros para la identificación de la neurona (número de capa y posición dentro de ella). Dicha lógica debe encargarse de recibir señales de entrada y enviar los resultados del procesamiento a las neuronas conectadas.

La unidad de procesamiento debe contener al menos una unidad aritmético-lógica (ALU), un registro acumulador (Ax) y un registro auxiliar para la multiplicación (Mx). En este caso la multiplicación se hace por algoritmo de suma y corrimiento (*add-shift*). Implementar un multiplicador por hardware representa un costo adicional de área de integración, especialmente, sabiendo que, cada unidad de procesamiento necesita su propio multiplicador. Las operaciones son controladas por el procesador consistente en un puntero de instrucciones, un registro de datos (MDR), un registro de instrucciones (ICR) y la memoria de datos/programa.

Un procesador neuronal de este tipo puede incluir mas de una unidad de procesamiento en un mismo dispositivo VLSI. Inclusive, manteniendo una estructura de comunicaciones fija entre las unidades de un mismo dispositivo (Fig. 3.4). De esta manera no se aumenta la cantidad de pines. La figura 3.4 muestra un arreglo bidimensional de procesadores neuronales comunicados por conexiones punto a punto en una de las direcciones del arreglo (horizontal) y por medio de un nodo especial encargado de hacer un algoritmo de enrutamiento para la otra dirección. Este nodo debe posibilitar que los mensajes lleguen a todas las unidades neuronales que poseen conexión sináptica con cada una de ellas.

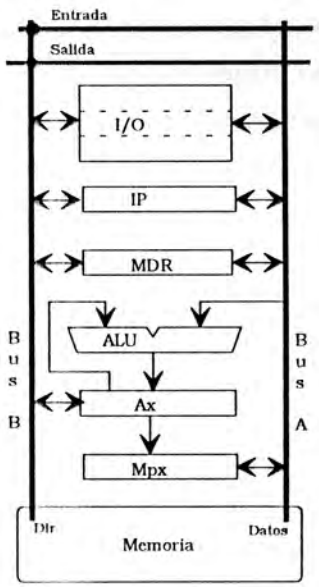

Figura 3.3 Unidad de procesamiento neuronal digital

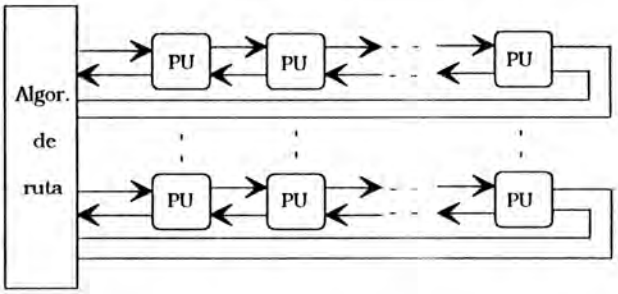

Figura 3.4 Sistema neuronal digital

3.3.3 NEURONAS MIXTAS ANALOGICO-DIGITALES

De las técnicas híbridas analógico-digiteles, sin duda, la que mayor interés ha suscitado es la aritmética de pulsos. Probablemente, el interés se deba a la similitud que puede encontrarse con las neuronas biológicas en lo que se refiere al pulso como mecanismo neurotransmisor. Esta técnica ha sido desarrollada por Murray (Murray, 1987a, 1987b, 1989) y trata de tomar ciertos beneficios de ambas técnicas. Básicamente, existe una codificación de la señal en forma digital y analógica conviviendo en distintas partes del circuito neuronal. La señal analógica se representa por una tensión $\mathbf{v_i}$ como una función continua del tiempo. La señal digital es codificada en pulsos digitales a una frecuencia de portadora mayor que el ancho de banda de la señal continua. La figura 3.5a muestra el esquema de una neurona la cual recibe una señal de entrada analógica $\mathbf{v_i}$ y proporciona una salida $\mathbf{y_j}$ codificada en pulsos, cuya frecuencia es proporcional a la amplitud de la entrada. Este comportamiento puede ser generado por medio de un oscilador controlado por tensión (VCO). La función sináptica en este modelo es la de regular (modular) la cantidad de pulsos que conforman la señal neurotransmisora (Fig. 3.5b). Las salidas neuronales son sumadas en tensión o en corriente utilizando funciones de tipo OR.

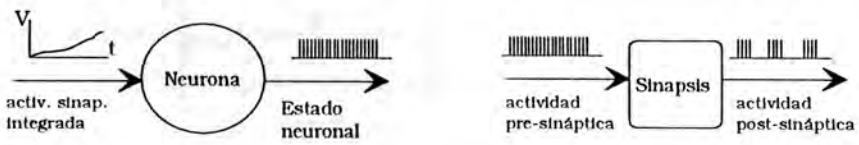

Figura 3.5 Codificación de pulsos: a) Neurona, b) Sinapsis

La figura 3.6 muestra un esquema en el cual se suman los aportes sinápticos excitatorios e inhibitorios de los pulsos y su conversión a una tensión en un capacitor. Esta tensión resultante controla un VCO entre límites de frecuencia ϕ_{min} y ϕ_{max}.

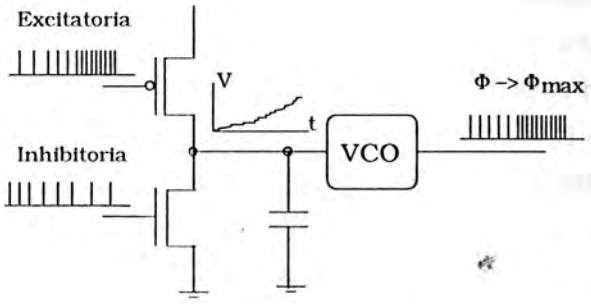

Figura 3.6 Sumador

La función sináptica se realiza utilizando pulsos cuya relación de trabajo decrece desde 1/2 hasta un cierto número $1/2^N$. Este número N indica el número de bits de precisión del circuito. El coeficiente de peso sináptico es almacenado en memoria en forma digital y modula la habilitación de los pulsos mencionados anteriormente. Es decir cada bit **i** de memoria en 1 habilita el paso de pulsos de la portadora hacia la salida por un intervalo $1/2^i$. Los bits de la palabra de peso en 0 inhabilitan que dichos pulsos pasen a la salida sináptica. La función sinaptica puede ser representada por la ecuación:

$$o_j = S_j(W_{j1}c_1 + W_{j2}c_2 + \ldots + W_{jN}c_N) \qquad (3.5)$$

La figura 3.7 muestra la manera en que estos pulsos son combinados mediante una función OR e inyectados a la columna exitatoria o inhibitoria que conducen al cuerpo de la neurona.

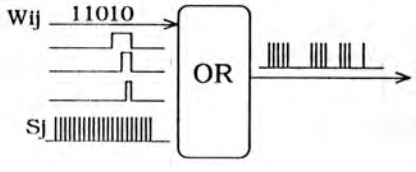

Figura 3.7 Modulador de pesos sinápticos

3.3.3.1 PESOS ANALOGICOS EN EL SISTEMA DE CODIFICACION DE PULSOS

La memoria digital de almacenamiento del peso en la función sináptica puede ser reemplazada por una versión plenamente analógica (Murray,1989). En este caso el par inversor T1-T2 de la figura 3.8 está asociado a la función sináptica. La misma muestra que el coeficiente de peso es almacenado dinámicamente como una tensión sobre el capacitor cm. El pulso de portadora S_j maneja el par inversor. Cuando S_j=0 el capacitor de salida se carga a la tensión almacenada en la memoria dinámica. Cuando S_j=1 se habilita la descarga del capacitor a través de T2 con una constante de tiempo mucho mayor que la de carga (Fig. 3.9). Como puede verse en la figura 3.9 la memoria dinámica controla el valor de continua de salida, y por lo tanto el ancho del pulso que se conforma a la salida del segundo inversor.

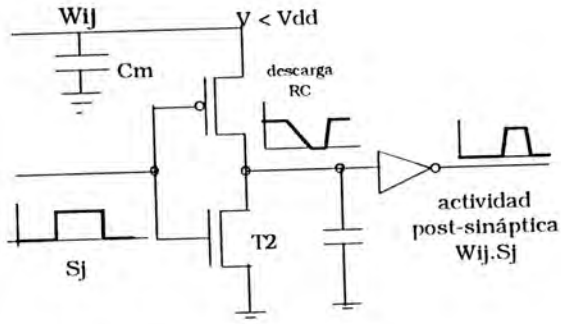

Figura 3.8 Memoria analógica en el sistema de codificación de pulsos

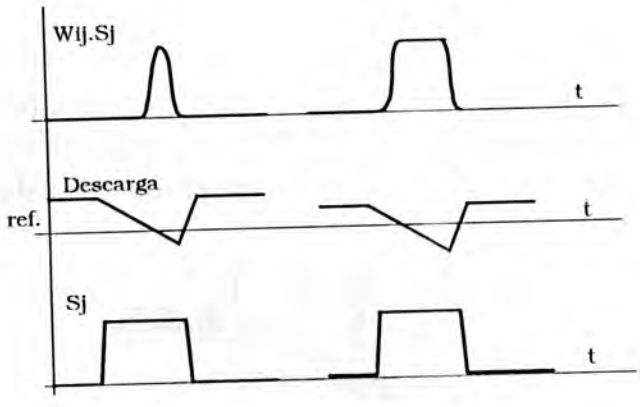

Figura 3.9 Generación del pulso en función del valor almacenado en memoria

Este método de almacenamiento analógico reduce considerablemente el área destinada a la matriz de coeficientes de pesos respecto del modelo digital. Además, el uso de pseudo-relojes del modelo digital en complicado de implementar y produce ruido y picos de consumo de corriente dentro del VLSI.

3.3.4 COMPARACION ENTRE NEURONAS ANALOGICAS, DIGITALES E HIBRIDAS: ANALISIS Y DISCUSION

La importancia de llevar los modelos neuronales teóricos al campo de las aplicaciones es fundamental desde el punto de vista del rendimiento de la red neuronal en la solución de sistemas complejos. El elemento constitutivo básico de las neurocomputadoras es la unidad de procesamiento. Estas unidades computacionales deben ser dispuestas de manera de generar una arquitectura de cómputo en paralelo en una organización regular de una o más capas.

Los dispositivos neuronales en VLSI pueden concentrar un número importante de unidades de cómputo, dependiendo del límite de la tecnología empleada y de la complejidad requerida por la unidad neuronal. A este respecto puede decirse que los modelos analógicos poseen una amplia ventaja sobre los digitales e híbridos. Un multiplicador sináptico puede ser desarrollado con una decena de transistores, al igual que los módulos que implementan las funciones no-lineales de salida. En particular, si se utiliza la tecnología MOS trabajando en la zona previa al umbral (*subthreshold*) la característica de transferencia entre corriente y tensión sigue una ley exponencial, lo cual permite implementar, con pocos transistores la respuesta sigmoidea típica en los modelos tipo Perceptrón. En consecuencia, podemos hacer un cálculo aproximado de la cantidad de transistores necesaria en una neurona analógica a través de la ecuación:

$$N_T = \sum_{j=1}^{n} \left(n_{mul}^{j} \cdot conex^{j-1} + n_f^j \right) conex^{j+1} \quad (3.6)$$

en la cual N_T es el número total de transistores utilizados, n_{mul} y n_f, el número de transistores del multiplicador y de la función no lineal respectivamente; y **conex** representa el número de conexiones o *fan-in* de cada neurona. En todos los casos el supra-índice representa el número de capa de la red neuronal.

Las neuronas digitales, por el contrario no son compactas. La forma más económica de hacer un multiplicador de punto fijo es utilizando una unidad aritmético-lógica (ALU) y un registro de desplazamiento. En este caso la multiplicación se realiza por sumas y desplazamientos hacia la izquierda del multiplicador. Aún cuando los transistores de la tecnología digital son más pequeños que los de la analógica, tanto la ALU como un registro de desplazamiento, consumen un área mucho mayor que en el caso anterior. Esta área aumenta con el número de bits de precisión del multiplicador. Además, aumenta el número de iteraciones en la cuenta y en consecuencia, se reduce el ancho de banda de la red neuronal. Si bién existen multiplicadores que realizan la operación en un solo paso, el área que consumen es prohibitiva si el circuito debe repetirse para cada unidad de procesamiento.

En la tecnología híbrida pueden darse ambas situaciones, pero si se opta por la memoria analógica dinámica, el área consumida es pequeña y comparable a la del multiplicador analógico.

Otro punto importante es el referente a la potencia consumida por cada neurona. Si se desea alcanzar un nivel de integración elevado, el consumo de potencia por neurona debe ser extremadamente bajo. Por ejemplo, una neurona biológica disipa del orden de 10^{-12}watts, un transistor trabajando en conmutación, alrededor de 1 millon de veces más. Aún cuando parece imposible mejorar el consumo en esta cifra, pueden realizarse algunas mejoras respecto de los circuitos tradicionales usados tanto en tecnología analógica como digital. Una es bajar el nivel de tensión de la fuente a valores cercanos a 1 o 2 volts. La otra, es utilizar al MOS en la zona *subthreshold*, donde la corriente de drenador es del orden de los nanoamperes o inclusive picoamperes. Esta última facilita el hecho de poder disminuir la fuente ya que la zona *subthreshold*, se encuentra para valores de tensión compuerta-fuente menores a 0,7 volts.

Podemos definir al ancho de banda de un circuito neuronal como el número de patrones procesados por segundo. Aquí las ventajas no son tan claras. El tiempo de procesamiento típico medido por el tiempo de respuesta (rise-time) en una neurona analógica es del orden 100 nseg. Una neurona digital con un reloj elevado necesita del orden de 200 a 300 nseg. De todas maneras estas diferencias no son grandes y las ventajas podrían revertirse si se utiliza tecnologías submicrón con relojes mayores de 100 MHz.

Los diseños digitales aventajan a los analógicos, fundamentalmente, en dos aspectos, exactitud y precisión de los cómputos. La exactitud está relacionada a la mejor relación señal-ruido de las señales digitales frente a las analógicas. Este aspecto puede ser importante, sobre todo, cuando la red neuronal tiene varias capas de procesamiento o cuando aumenta la distancia de las conexiones sinápticas. En contrapartida, aunque las neuronas analógicas sean más suceptibles al ruido, es lógico que en un VLSI digital haya mas ruido generado por la misma electrónica digital. Si, además, esta electrónica es asíncrona, las conmutaciones de los relojes asincrónicos pueden ocasionar cambios de estado en líneas vecinas (*cross-talk*). Es importante notar que, aún cuando los ruidos pueden ser muy perjudiciales, en una red neuronal la gran cantidad de conexiones sinápticas y neuronas en paralelo hace que un error simple tenga una influencia moderada en el cómputo final. Además, como se vio en el capítulo 2, las funciones no-lineales de salida saturan para un nivel elevado de la entrada, lo cual favorece la afirmación anterior.

Respecto de la precisión, no cabe duda que los sistemas digitales son ideales para fijar con exactitud la precisión de un circuito, medida en bits. Por la última afirmación del párrafo anterior, las redes neuronales no requieren una alta precisión. Las redes de Hopfield y todas aquellas que tienen una función no-lineal tipo umbral sólo necesitan dos estados de salida, es decir, un bit de precisión. En general, para las neuronas con salida continua tampoco se necesitan muchos bits de precisión, alcanza con 3 a 8 bits, dependiendo de la aplicación.

Los circuitos analógicos son en general compactos, por ende no se utilizan técnicas de compartición o multiplexado de los elementos de cómputo o canales de comunicación. En cambio, estas técnicas pueden ser muy útiles en los sistemas digitales. Como se desprende de lo analizado, los circuitos neuronales digitales son más costosos en área de VLSI, por consiguiente, puede ser ventajoso diseñar circuitos con tecnologías avanzadas y veloces, con alta capacidad de procesamiento. Estas unidades de procesamiento real (UP) pueden emular múltiples neuronas virtuales (EP). Esto reduce el número de UP que deben integrarse en un mismo dispositivo y pueden compartirse los canales internos de comunicación de datos entre neuronas. Por otro lado, se complica la arquitectura de diseño debido a que debe implementarse una manera de distinguir las diferentes EP virtuales dentro de una misma UP. Este es el caso de una arquitectura como la descripta en el ejemplo del procesador RISC de la sección 3.3.2. La figura 3.3 muestra como múltiples unidades UP pueden ser conectadas a buses comunes de datos y direcciones.

En las neuronas digitales los buses compartidos son esenciales debido a que es necesario el acceso a zonas de memoria global. Un caso típico es el de la generación de la función no-lineal de salida. Las funciones exponenciales, trigonométricas, hiperbólicas o gaussianas deben ser almacenadas en forma de una tabla en una zona de memoria de acceso global. El problema es justamente la saturación del canal de comunicaciones entre las neuronas y la zona de memoria global. Además, las funciones llevadas a tablas consumen mucha memoria, aún cuando la precisión sea baja.

3.3.4.1 FLEXIBILIDAD: DISPOSITIVOS DE PROPOSITOS GENERALES Y DISPOSITIVOS ORIENTADOS A APLICACIONES ESPECIFICAS

Sin duda, uno de los aspectos más importantes en el diseño de un VLSI neuronal está referido a su flexibilidad para implementar diferentes modelos. En otras palabras, si el diseño del circuito neuronal está orientado a un único modelo neuronal o si la estructura interna permite ejecutar otros algoritmos. Al

respecto, podemos hacer una categorización simple dividiendo los mismos en circuitos de:

- propósito general
- propósito especial u orientados hacia una sola aplicación

Esta clasificación está bastante correlacionada con la posibilidad de utilizar una tecnología analógica o digital. La flexibilidad de los diseños analógicos es mucho menor. Cambiar un modelo neuronal significa cambiar una parte sustancial de la función sináptica o neuronal, lo cual en un circuito analógico se traduce en una organización totalmente diferente de los transistores que lo integran. Además, dos funciones de complejidad matemática similar pueden traducirse en complejidades de diseño electrónico muy distintas. Por ejemplo: función tgh(.) y función gaussiana. En consecuencia, los circuitos neuronales analógicos deben ser orientados hacia un modelo en particular. Más aún, muchas veces, hacia una aplicación en particular.

Hoy en día, esta falta de flexibilidad se compensa, parcialmente, por el gran avance de los procesos de integración y fabricación; comúnmente llamados *custom*. Es muy probable que se incrementen mucho más las facilidades para la fabricación "a medida" de circuitos electrónicos orientados a una aplicación determinada.

De todas maneras, la tecnología digital proporciona una flexibilidad mayor para el desarrollo de modelos neuronales, especialmente cuando se trata de prototipos de laboratorio. Los procesadores neuronales, como el descripto en la sección 3.3.2, pueden ejecutar varios algoritmos de cómputo correspondiente a distintos modelos o diferentes arquitecturas de interconexión y número de capas dentro de un mismo modelo. Si las unidades de computo digitales ejecutan un programa en memoria, cambiar de modelo sólo significa modificar el programa. Sin embargo, esta flexibilidad no es totalmente gratis, ya que tener una memoria de programa, buses compartidos y otras características propias de la arquitectura lleva a un costo extra de área del VLSI que disminuye enormemente el número de unidades de procesamiento que pueden integrarse en un mismo dispositivo.

3.3.4.2 EL PROBLEMA DEL ALMACENAMIENTO DE LOS COEFICIENTES DE PESO

El problema de almacenar eficientemente los parámetros de la red neuronal es central en el diseño de la misma. Los parámetros de una red neuronal son: los pesos, Wij, los umbrales θj, y algunos coeficientes propios del modelo, tal como la temperatura de la sigmoide en los Perceptrones y redes de Boltzman.

Estos coeficientes son modificados durante la ejecución del algoritmo de aprendizaje y se mantienen constantes en la faz de procesamiento paralelo. En su mayoría, los algoritmos de entrenamiento son iterativos, por lo cual cada parámetro es modificado un gran número de veces, generalmente, en pequeñas cantidades. Esto impone la necesidad de tener una memoria de lectura escritura, ya sea digital o analógica.

El almacenamiento en estas memorias puede ser temporario en celdas dinámicas o estáticas, o semi-permanente en memorias eléctricamente reprogramables. Las memorias digitales pueden tener cualquiera de estos tres esquemas. Las analógicas pueden usar celdas dinámicas o eléctricamente reprogramables. El almacenamiento en memoria de los pesos en forma digital es idéntico al utilizado en las memorias digitales comunes. En cambio las memorias analógicas constituyen un fenómeno novedoso en el cual aún resta mucho por investigar. Un ejemplo de memorias analógicas es el utilizado en el modelo analógico-digital de la sección 3.3.3.1.

Las memorias analógicas pueden ser dinámicas o semi-permanentes. La celda de la memoria analógica dinámica es similar a su correspondiente digital. El valor del dato almacenado es función de la tensión en el capacitor de entrada de un transistor MOS. El mecanismo de control de esta tensión puede hacerse usando una llave electrónica como en los circuitos de capacitores conmutados. Las llaves son MOS con una corriente de fuga muy pequeña, no obstante, estas memorias requieren un sistema de refresco. Vale aclarar que el refresco aquí es mucho más crítico que en el caso de las memorias digitales, debido a que éstas sólo tienen dos estados, en cambio las memorias analógicas deben mantener la precisión de bit seleccionada en el diseño. Una ventaja de las memorias

dinámicas es que son compactas y el ancho de banda de programación es elevado. Sin embargo, tienen el problema que necesitan sistemas de refresco asincrónicos y complicados, debido a que los pesos de las sinapses son modificados, también, en forma asincrona.

Una técnica que ha despertado mucho interés es la de escritura semi-permanente o borrable eléctricamente EEPROM. Estas memorias se diseñan utilizando un transistor MOS con dos compuertas. Una de ellas es flotante, es decir, sin acceso desde el exterior. El propósito de esta compuerta es la de controlar la característica tensión-corriente del MOS a través de la carga almacenada (Fig. 3.10). La tensión umbral, Vt, se modifica en función de la carga en la compuerta flotante.

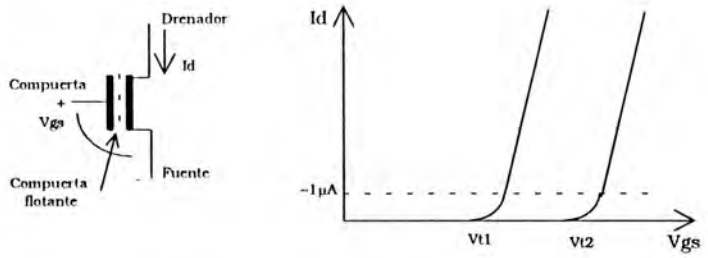

Figura 3.10 variación de Vt en función de la carga en la compuerta flotante

Dado que esta compuerta no está accesible desde el exterior (Fig. 3.10), se utiliza el fenómeno túnel o Fowler-Norldheim para llevar cargas, que quedan almacenadas en forma estática. El apéndice A está dedicado al análisis del efecto Fowler-Norldheim en memorias de compuerta flotante. La modificación de carga en la compuerta requiere de potenciales de programación elevados, entre 14 y 24 volts; que generan un campo eléctrico suficientemente alto como para producir el efecto túnel. Este proceso puede modelarse a través de la ecuación:

$$\Delta V_t = V_{TOt} + V_{PP2} - V_{TOwt} - 1,375.10^{-6} B / \ln\left[e^{(B/E_0)} + 32420.B.T_P\right] \qquad (3.7)$$

donde VPP2 y TP representan la tensión y el ancho del pulso de programación y ΔVt la diferencia entre el valor de tensión de compuerta actual y el deseado. B, Eo y VTO son constantes propias del MOS.

La gran ventaja de estas memorias analógicas es el poder retener la carga almacenada sin necesidad de una fuente de alimentación. Para evacuar la carga de la compuerta flotante se requiere un campo eléctrico similar aplicado en sentido inverso. Si bien existen pequeñas fugas de carga, puede asegurarse una variación menor al 1% en varios meses.

En las celdas estandar se necesitan pulsos de programación elevados, por lo que existe el riesgo de alcanzar tensiones de ruptura del transistor. Para evitar ello, se utilizan capas de OSi delgadas entre las compuertas, o superficies texturadas que aumentan el campo eléctrico a bajas tensiones de programación (Apéndice A)(Ellis, 1982)(Carley, 1989).

3.4 DISPOSITIVOS NEURONALES: ESTADO DEL ARTE

En los últimos cinco años han surgido algunas decenas de proyectos orientados a la generación de dispositivos de computación neuronal. Los mismos varían en un amplio rango de complejidad, versatilidad y área de aplicaciones. La revista IEEE Transaction on Neural Networks ha dedicado tres números completos a la generación de redes neuronales en hardware (IEEE, 1991, 1992, 1993). Otras actividades sobresalientes son las realizadas por el grupo del Dr. Carver Mead del Instituto Tecnológico de California, Caltech, quien desarrolló sistemas neuronales especiales orientados a substituir los órganos que realizan la primera etapa de la función sensorial humana; tales como una Retina y una Coclea. Además ha implementado un sensor de movimiento óptico y un dispositivo llamado *seehear* destinado a traducir imágenes ópticas en señales auditivas capaces de ser percibidas por personas ciegas o disminuidos visuales. Los trabajos de Mead utilizan tecnología CMOS en la zona de funcionamiento anterior al umbral (*subthreshold*). Las propiedades y resultados han sido volcados en el libro *Analog VLSI and Neural Sistems*, (Mead, 1989).

El propósito de esta sección es presentar brevemente algunas de las realizaciones principales con el motivo de hacer un análisis comparativo del

estado del arte de los dispositivos de computación neuronal. Este análisis ha servido para elaborar un criterio sobre la conveniencia de utilizar una determinada tecnología, arquitectura, y otros propósitos, de diseño, ya enumerados, en las realizaciones hechas para esta tesis y que son volcadas en los capítulos siguientes.

3.4.1 ANNA (Analog Neural Network Arithmetic and logic unit)

Este dispositivo es híbrido y está orientado a la arquitectura de Perceptrón multicapa. Utiliza un esquema combinado analógico-digital, por el cual las primeras capas de procesamiento son analógicas, compactas y de baja precisión, la última capa es digital y aporta mayor precisión. Este dispositivo permite construir Perceptrones con un número variable de capas. La comunicación de entrada y salida es digital. El vector de entrada es cargado en un registro de desplazamiento especial de 64 bits (*barrel shifter*) y afecta a la unidad de computación neuronal, donde se realizan las operaciones sinápticas. Este dispositivo posee una memoria interna de 4096 pesos analógicos que pueden ser flexiblemente agrupados según las necesidades de cada capa.

Las características principales del ANNA son: la flexibilidad de su arquitectura, el elevado número (teórico) de computaciones sinápticas, 10.000 Mc/s, y su sistema de precisión controlada. Por el contrario, posee un método de control de la memoria complejo, con funciones de almacenamiento dinámico y refresco; y necesita de un procesador adicional (DSP) que genere una secuencia de palabras de microcódigo correspondientes a la topología implementada.

3.4.2 CNAPS (Connected Network of adaptive Processors)

Este dispositivo es un procesador digital SIMD, es decir, trabaja por una secuencia de una instrucción y múltiples datos. Su arquitectura interna es un arreglo lineal de unidades de procesamiento conectadas a un único módulo

encargado de generar las secuencias de instrucciones. Este último está encargado también de manejar la entrada y salida de datos.

La representación de los datos es en punto fijo. Cada procesador posee un multiplicador de 9 x 16 bits, un sumador de 32 bits y 32 registros de almacenamiento local. CNAPS posee, internamente, 2 M de pesos de 1 bit y 256 K pesos de 8 bits. Con un reloj de 25 MHz es capas de realizar 1.600 Mc/s

Este dispositivo, en configuración de Perceptrón multicapa, ha sido utilizado eficazmente para implementar una versión del sistema NETtalk (Senowski, 1987) y demostró una rendimiento equivalente a 2000 veces el de una estación SUN-Sparc.

3.4.3 ETANN (Electrically Trainable Analog Neural Network)

El dispositivo neuronal Etann es, hasta la fecha, la implementación más avanzada realizada en VLSI orientada a redes neuronales. Este dispositivo, fabricado con tecnología CMOS de 1µ, integra 128 neuronas analógicas en dos capas de procesamiento en cascada. En el modo de procesamiento paralelo (PDP) cada neurona en cada capa computa en forma analógica la función

$$O_i = sigmoide\left(\sum_{j=1}^{64}(W_{i,j}.I_j)+\theta_i\right) \qquad (3.8)$$

donde $\mathbf{O_i}$ es una de las salidas analógicas de la red, $\mathbf{I_j}$ son las entradas analógicas, $\mathbf{W_{ij}}$ son los coeficientes de peso internos, y $\boldsymbol{\theta_i}$ los valores de umbral. La dimensión máxima del vector de entrada es 128 y la del vector de salida 64. Los 8192 coeficientes $\mathbf{W_{ij}}$, distribuidos en dos matrices de 64x64, y los 2048 umbrales $\boldsymbol{\theta_i}$ son reprogramables eléctricamente (Apéndice A).

La capacidad máxima de procesamiento del Etann, para un vector de entrada de dimensión 128 y de salida 64 es de unos 8K multiplicaciones y sumas en 1µ seg; es decir, más de 8 exp(9) multiplicaciones y sumas por segundo, con una precisión equivalente a 7 bits. La capacidad de procesamiento del Etann es

equivalente a unos 800 procesadores digitales trabajando a 10 MIPS, considerando que estos procesadores pueden ser acoplados en forma perfecta.

El Etann posee una función de activación continua de tipo sigmoidea que puede ser modificada a una de tipo umbral usando un modo de alta ganancia en el amplificador de salida. Este modo permite construir redes con salida binaria, por ejemplo, tipo Hopfield, con la ayuda de capacitores externos.

La figura 3.11 representa el diagrama en bloques del Etann. El dispositivo es fundamentalmente analógico, pero posee electrónica de control digital para la selección individual de sinapses, umbrales, y neuronas; y para el control de los diferentes modos de funcionamiento. Las señales lógicas externas para decodificación son compatibles TTL

En la figura se destacan dos arreglos matriciales de 64x64 que representan los coeficientes reprogramables **Wij** de cada una de las dos capas que son configurables en cascada para construir una red con capa oculta; y otros dos bloques correspondientes a los umbrales **θi** de 16x64 asociados a las matrices antes mencionadas. En la figura 3.1 puede verse en detalle la disposición del arreglo matricial respecto de las entradas. Las salidas de los multiplicadores son sumadas y procesadas para conformar el nodo de salida. Las señales de entrada, **Io-I63**, admiten un rango dinámico de 0 a 5volts. El rango dinámico de las salidas, **Oo-O63**, es de 0volt al doble del valor de una referencia externa.

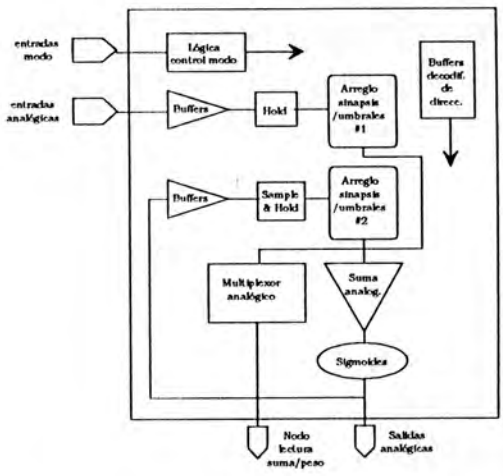

Figura 3.11 Diagrama en bloques

Puede verse en la figura 3.11 que existe una realimentación de las salidas hacia el segundo bloque matricula de pesos. Esto permite implementar una segunda capa de procesamiento dentro del mismo dispositivo. Todas las señales de entrada y realimentación pasan a través de *buffers* separadores y son almacenadas temporariamente (*holds*).

3.5 NEUROCOMPUTADORAS

Una computadora neuronal es, esencialmente, un arreglo paralelo de procesadores interconectados que operan en forma concurrente. Los procesadores neuronales son, en general, de la misma naturaleza, (analógicos, digitales o híbridos) y poseen la misma arquitectura interna. Las neurocomputadoras pueden estar destinadas a un propósito general u orientadas a una aplicación especifica. Las primeras permiten la computación de diversos algoritmos neuronales, redes de distinta configuración y número de capas; y de distintos algoritmos de entrenamiento. Las segundas son mucho menos flexibles, poseen, en general, una arquitectura fija o con pocas variantes, representan un solo modelo neuronal y poseen una interfaz sensorial adaptada para el problema especifico.

La complejidad de las neurocomputadoras es más variable aún que la de los dispositivos neuronales. El mínimo del rango de complejidad lo constituyen las computadoras secuenciales convencionales, simulando algoritmos de modelos neuronales. El máximo esta dado por supercomputadoras que disponen distintas formas de arreglos paralelos con una arquitectura orientada a la computación neuronal. La figura 3.12 representa el número de nodos de una neurocomputadora en función de la complejidad de los mismos, para el espectro existente.

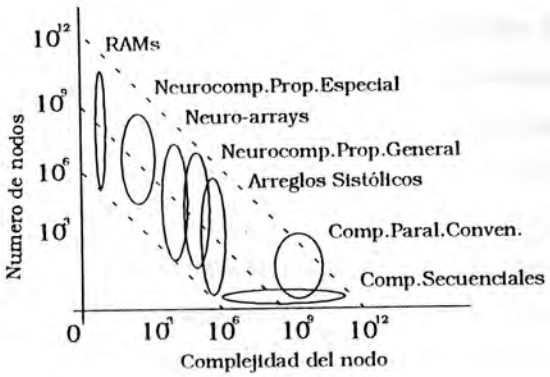

Figura 3.12 Diagrama de cantidad de nodos en función de la complejidad

Las computadoras neuronales son afectadas en forma directa por las propiedades de los modelos analógicos, digitales o híbridos descriptos anteriormente. En los últimos años ciertas lineas de investigación sobre computación paralela se han volcado al estudio de neurocomputadoras. Ambas ciencias comparten reglas comunes tales como múltiples unidades de cómputo, arquitectura de interconexión de estas unidades, compartición de la memoria, estructura de comunicación entre unidades de cómputo, sincronización, etc.

En especial, las neurocomputadoras de propósito general son bastante similares a las computadoras paralelas o supercomputadoras. Inclusive, se han desarrollado algunas neurocomputadoras utilizando procesadores convencionales tipo DSP (Morgan, 1992) o RISC (Treleaven, 1989). En general, cualquiera de estos procesadores posee una complejidad mayor que la requerida para ejecutar algoritmos neuronales. Una posibilidad es utilizar un subconjunto de la arquitectura de estos procesadores para disminuir la complejidad del sistema asociado, interconexión, compartición de recursos, etc.

Otra opción es la generación de arreglos sistólicos especialmente diseñados para modelos neuronales (Lehmann, 1993). Ambos casos aceptan distintos modelos neuronales y algoritmos de entrenamiento.

La neurocomputación de propósitos generales con procesadores digitales comparte también muchos de los inconvenientes de las computadoras paralelas, fundamentalmente, debido a que los procesadores no son lo suficientemente simples y compactos como para utilizar uno por sinapsis o por

neurona del modelo elegido. Cada procesador simula la operación de más de una unidad de procesamiento virtual, con lo cual, surgen problemas como el de asignación de tareas y cargas a los procesadores, granularización de los programas, etc.

También los procesadores analógicos pueden ser utilizados para el diseño de computadoras neuronales de propósitos generales. Si bien, se dijo, estas son menos flexibles en función de los modelos que pueden representar y del número de capas, neuronas y sinapsis que conforman una arquitectura dada. Si bien, aún la mayoría de la investigación en redes analógicas se encuentra a nivel del dispositivo, se consignan algunos desarrollos de neurocomputadora analógicas utilizando circuitos convencionales y/o *custom* (Fisher, 1991).

Aún no han aparecido en la literatura científica computadoras que utilizan dispositivos híbridos con la técnica de codificación de pulsos. Murray y Del Corso, se hallan abocados a un desarrollo de este tipo.

Dentro de los sistemas neuronales de propósito especial se destacan claramente los de Carver Mead del Instituto tecnológico de California (Caltech). Este grupo ha realizado varios desarrollos en el área de la bioingeniería, tales como una retina electrónica, una coclea, un sistema VLSI binaural, y un sensor de movimiento óptico (Mead, 1989).

En la sección siguiente, se presentan algunos ejemplos que reflejan el estado del arte de las neurocomputadoras, tanto de propósito general como aquellas orientadas a una aplicación en particular.

3.5.1 GENES: Arreglo sistólico bidimensional

El proyecto GENES utiliza la tecnología CMOS digital para generar un arreglo sistólico bidimensional orientado a la computación de varios algoritmos neuronales. Cada dispositivo que integra el sistema de cómputo, implementa un arreglo bidimensional de 4x4 celdas como indica la figura 3.13a. La

arquitectura de la celda está orientada al cómputo de la función sináptica y también a la ecuación que determina la actualización del coeficiente de peso del algoritmo de aprendizaje. Esta computación, salvo para las redes que utilizan el algoritmo de retropropagación, es completamente local. La arquitectura de la celda se detalla en la figura 3.13b. Debido al mayor número de sinapses respecto del de neuronas, las celdas han sido orientadas al cómputo sináptico, dejando la función neuronal para ser almacenadas en tablas.

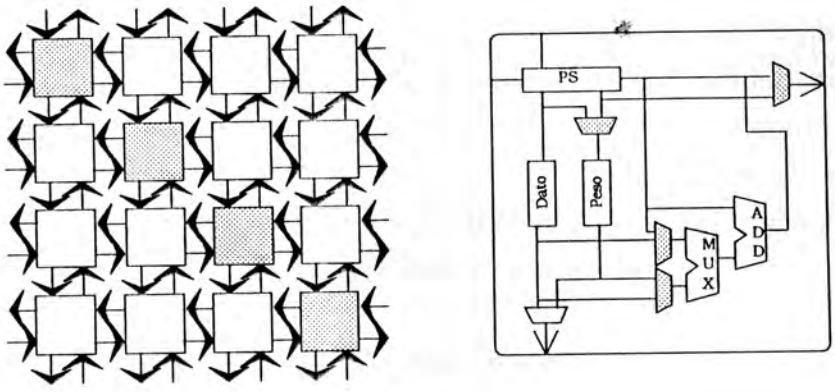

Figura 3.13 Arreglo sistólico a) arquitectura bidimensional b) esquema interno

La ecuación (3.9) representa el algoritmo de aprendizaje de Hebb como ejemplo de funcionamiento del arreglo sistólico. La figura 3.14a muestra como se introducen los vectores X e Y en el arreglo y circulan por el mismo en direcciones de norte a sur y de este a oeste, respectivamente. En régimen permanente pueden haber más de un patrón circulando por el arreglo a un dado tiempo, lo cual permite explotar toda la capacidad paralela del mismo. El algoritmo de ruta y cómputo de los patrones puede ser modificado para mejorar su rendimiento (Fig. 3.14b). El extremo derecho del arreglo realiza la función de ingreso del dato y de la computación de la función no lineal $\phi(.)$.

$$w_{ij}(n+1) = w_{ij}(n) + \eta x_j y_i \qquad \forall i, j \qquad (3.9)$$

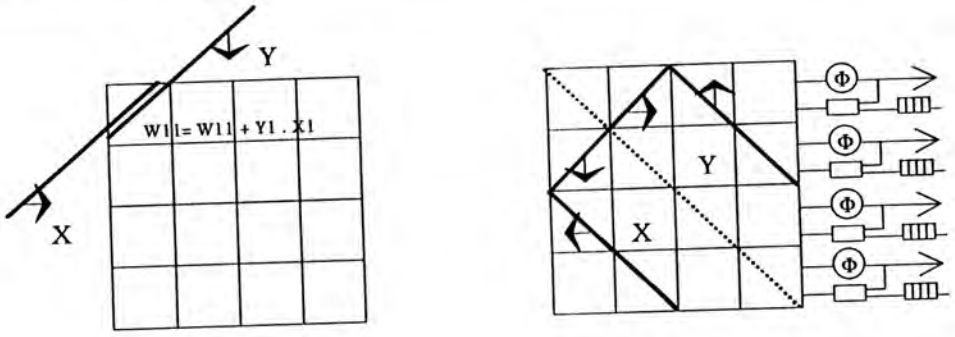

Figura 3.14 Algoritmo de ruta para la actualización de los pesos

El dato computado debe ser transmitido a los cuatro vecinos de cada celda. Estos datos deben ser almacenados dentro de cada celda, por lo cual se disponen cinco registros dedicados especialmente a cada uno de los parámetros almacenados. El corazón de la celda lo constituye un multiplicador-acumulador tipo MAC. La precisión de las operaciones es de 8 bits en punto fijo, para disminuir el consumo de área.

El rendimiento de este sistema ha sido probado para 16 neuronas, con una frecuencia de reloj de 10 MHz, obteniendo un promedio de 50 Mc/s.

3.5.2 PROCESADOR NEURONAL PROGRAMABLE:

El grupo de fisica aplicada de los laboratorios de investigaciones de la compañía Lockheed ha desarrollado un procesador neuronal analógico basado en el diseño *custom* de un resistor variable. Esta neurocomputadora posee 256 neuronas y 2048 sinapsis sobre 16 placas VME. El procesador permite computar algoritmos de modelos neuronales *feed-fordward* y Hopfield.

Cada neurona posee un amplificador operacional para la función no lineal y un dispositivo especial consistente en un conjunto de resistencias y llaves fabricado con tecnologia *gate-array* analógico. La figura 3.15a muestra el esquema de las resistencias conmutadas por llaves bipolares programables. Ocho de estos grupos se integran en cada dispositivo custom. La figura 3.15b

esquematiza una neurona analógica que incluye uno de estos dispositivos junto al amplificador operacional.

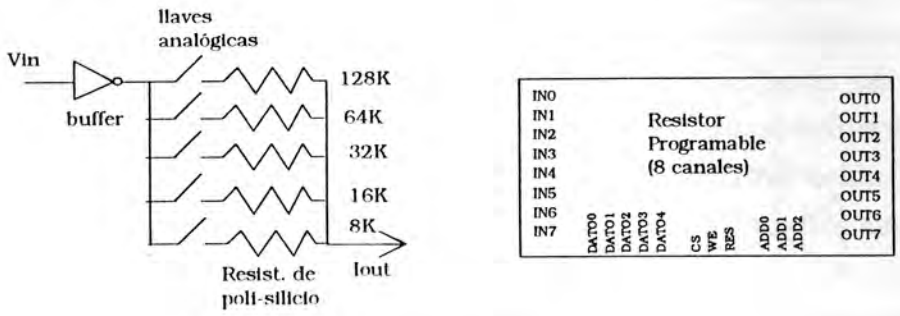

Figura 3.15 Sistema de resistencias conmutadas a) Esquema básico b) Dispositivo de 8 canales

El ancho de banda de esta neurocomputadora es de 90 KHz, y la resolución de 5 bits. Los pesos del sistema son controlados externamente por un procesador digital asociado. Si bien las entradas del sistema son puramente analógicas, se adicionó un conjunto de conversores digital a analógico con el fin de poder procesar patrones almacenados en la computadora.

Este procesador fue utilizado eficazmente en el control de lazo cerrado de un sistema de posicionamiento de espejos (Fisher, 1991).

3.5.3 RETINA ARTIFICIAL

La retina proyecta una imagen fisica sobre un arreglo de fotorreceptores generando la primera *imagen neural*. Esta imagen es posteriormente transmitida hacia el nervio óptico. C. Mead (1989), ha desarrollado una retina artificial. Este es un ejemplo de un sistema de neurocómputo especialmente dedicado a una aplicación. La retina electrónica consiste de: fotorreceptores, que computan el logaritmo de la señal de intensidad lumínica; celdas horizontales, conformadas por un arreglo bidimensional de resistores en formando hexágonos y que se encargan de distribuir y promediar la señal de entrada; las celdas bipolares, que producen una salida proporcional a la

diferencia entre la señal del fotorreceptor y la de las celdas horizontales. La figura 3.16 representa la disposición de las celdas horizontales. Se detalla además, el esquema de una celda bipolar con su fotorreceptor asociado.

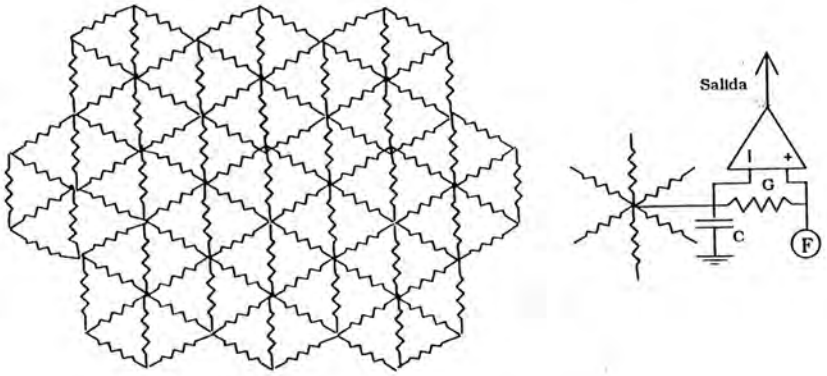

Figura 3.16 Retina artificial

Las señales de salida obtenidas por la retina artificial son muy similares a las reales, a velocidades electrónicas y con una alta tolerancia a imperfecciones del dispositivo, las cuales son características de los sistemas analógicos paralelos y masivos.

3.6 DISCUSION

En este capítulo se pone en evidencia las propiedades más importantes de los modelos analógicos, digitales e híbridos para la implementación de modelos neuronales, tanto a nivel de la celda neuronal mínima como a nivel del sistema neurocomputador. Se destaca en cada caso la influencia de los modelos analógicos y digitales en la precisión, ancho de banda, tipo de arquitectura, flexibilidad, almacenamiento de los parámetros, consumo de potencia, capacidad, comunicación e interconexión de las neuronas y el costo asociado al área del VLSI o a la integración del sistema.

A nivel del dispositivo, se han puesto de manifiesto los parámetros que redundan en las principales consideraciones que deben ser tenidas en cuenta en la implementación de modelos neuronales, tanto para ser utilizados como

parte constitutiva de neurocomputadoras de propósito general, como aquellas orientadas a una aplicación específica.

El análisis es fundamental para el desarrollo de esta tesis. Es obvio que no existe una única opción sobresaliente tanto para la generación de un VLSI neuronal, como para la de un sistema neuronal. De todas maneras, se ha optado por el modelo analógico tanto para el diseño de una celda neuronal como para el de una neurocomputadora. Las razones para esta decisión son:

Los dispositivos analógicos pueden integrar un número de neuronas mucho mayor en un mismo dispositivo.

Las celdas analógicas son compactas y tienen una relación uno a uno con las sinapsis y neuronas que computan.

Los modelos analógicos tienen una mayor proximidad con los modelos neuronales biológicos, en el modo en que realizan las funciones de procesamiento, en aspectos relacionados con el crecimiento potencial del sistema, consumo de potencia por neurona, crecimiento del interconexionado sináptico, etc.

Las celdas analógicas poseen un mayor ancho de banda, especialmente utilizando circuitos en el modo de corriente.

Aún cuando los modelos digitales poseen mayor resolución e inmunidad al ruido, estos aspectos quedan minimizados por el paralelismo masivo de unidades de cómputo de la red neuronal. Inclusive, un pequeño porcentaje de las neuronas y/o pesos sinápticos pueden estar fuera del rango de especificación, contaminados por ruido o poseer algún mal funcionamiento severo, sin afectar de forma determinante el funcionamiento global de la red. En este contexto, las neuronas analógicas saturan en forma natural a los valores de tensión de alimentación, mientras que en los dispositivos digitales debe controlarse estos limites así como el fenómeno de *overflow*.

Las neurocomputadoras analógicas pueden conectarse diréctamente a señales sensoriales y actuadoras externas.

Las lineas de investigación en computación neuronal analógica están menos exploradas y presentan un futuro más atractivo, especialmente para la aplicación de redes neuronales a problemas de tiempo real, tales como control, visión, reconocimiento de voz, y muchos otros.

En función de lo antedicho, este trabajo doctoral ha generado un modelo analógico a nivel neuronal. Este modelo será detalladamente descripto en el capítulo 4 de esta tesis y representa una neurona estática cuya función de activación es gaussiana. Se ha generado también una neurocomputadora analógica basada en el modelo perceptrón multicapa. Si bien, está orientada a una neurocomputadora de propósitos generales, es fácilmente adaptable a una aplicación específica y mantiene toda la potencialidad de cómputo intrínseca de dicha arquitectura. Los capítulos 5 y 6 de esta tesis reflejan lo hecho en este tema.

CAPITULO 4

REDES NEURONALES ANALOGICAS

4.1 INTRODUCCIÓN

Ya han sido puntualizadas, a muy grandes rasgos, unas pocas diferencias y similitudes entre el cerebro humano y las computadoras. También las ventajas de los sistemas neuronales biológicos respecto de las computadoras tradicionales para realizar un sinnúmero de tareas *cognitivas*. Solamente, en aquellas tareas basadas en computaciones fuertemente aritméticas, puede una computadora tradicional, con arquitectura tipo von Neumann o Hardvar, mejorar el rendimiento del cerebro humano. La ventaja más importante de las computadoras tradicionales radica en el hecho de que el tiempo de procesamiento de una unidad en silicio es tres o cuatro órdenes de magnitud menor que la de una unidad biológica.

Los modelos simplificados tomados de una neurona biológica pueden permitirnos nuevas arquitecturas computacionales orientadas a afrontar dichos problemas con soluciones más convenientes y veloces.

4.2 MODELO DE UNA NEURONA

La neurona es la unidad constitutiva básica del sistema nervioso. Cuenta con la maquinaria metabólica para producir la energía necesaria para el procesamiento y transmisión de información.

Un árbol de dendritas, llamadas *sinapsis*, realiza un procesamiento primario, recogiendo información de otras neuronas y realizando funciones tales como producto, suma, integración, etc. La resistencia eléctrica de las dendritas

atenúa las señales y disminuye el acoplamiento eléctrico con otras dendritas del árbol. El axón, que muchas neuronas poseen, es utilizado para *cuantificar* la información que debe ser transmitida a largas distancias (Hodgkin, 1952a). Las entradas son integradas por la capacidad de la célula hasta lograr una despolarización del citoplasma por encima de una tensión umbral, superado el cual, se genera una salida llamada potencial de acción (Hodgkin, 1952b).

La resistencia del citoplasma es suficientemente alta como para que la información transmitida sea poco menos que totalmente atenuada en distancias menores a 1 mm. Por esta razón el axón está equipado con regeneradores de señal ubicados en los nodos de Ranvier, que actúan como repetidores (DuBois, 1983).

El citoplasma de una neurona se encuentra polarizado a unos -80mV con respecto al fluido extracelular. Esta diferencia de potencial es soportada a través de la membrana que recubre el citoplasma. Si suficiente corriente es inyectada dentro del citoplasma en dirección de despolarizarlo, superado un umbral de -40mV, se genera un pulso nervioso.

La membrana nerviosa, de unos 50Å es responsable del intercambio de actividad eléctrica en la célula. Soporta un campo eléctrico mayor a 10^7 V / m, curiosamente, un valor similar al campo eléctrico en el óxido de silicio de un transistor MOS. Esto forma una barrera de potencial para los iones de sodio (Na), potasio (K) y cloro (Cl), responsables de las corrientes que atraviesan la membrana y determinan la actividad de la neurona. Dicha barrera de energía es de unos 2,4 eV, por lo cual, a temperatura ambiente la probabilidad de que los iones la atraviesen es excesivamente baja y constituye un aislante perfecto.

Las fuentes de poder que se encuentran en la membrana nerviosa de la célula despiden iones de Na hacia el exterior e importan iones de K hacia el citoplasma. Habrá, entonces, una concentración interna de iones K y una externa de iones Na, los cuales por un proceso de difusión generan una carga neta a ambos extremos de la membrana. Esta carga neta se almacena en la capacidad de la membrana de la neurona, causando un potencial negativo

respecto del fluido extracelular. La ecuación que gobierna este proceso es similar a la ecuación de difusión en una juntura p-n (Katz, 1966)

$$V_r = -\frac{kT}{q}.\ln\frac{N_{in}}{N_{ex}} \qquad (4.1)$$

donde Nin y Nex son las densidades de iones a ambos lados de la membrana. La figura 4.1 representa un circuito eléctrico equivalente

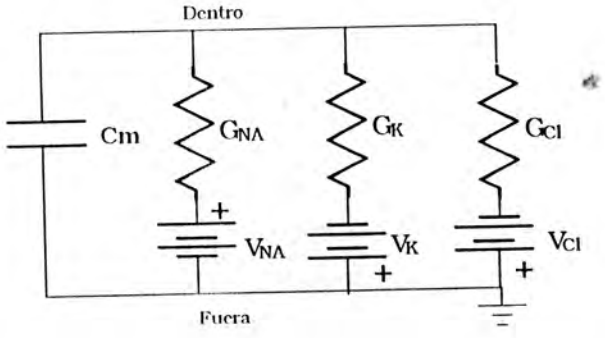

Figura 4.1 modelo eléctrico equivalente del funcionamiento de la membrana

La corriente que carga a el capacitor de la membrana es

$$I = (V_k - V).G_k + (V_{Na} - V).G_{Na} + (V_{Cl} - V).G_{Cl} \qquad (4.2)$$

con lo cual la tensión de reposo, para I=0 y despreciando la corriente de los iones de cloro es

$$V_0 = \frac{V_k G_k + V_{Na} G_{Na}}{G_k + G_{Na}} \qquad (4.3)$$

Una señal excitadora despolariza la membrana y una señal inhibitoria la hiperpolariza. Actuando sobre las tensiones o las conductancias de (4.3) se logra modificar la actividad eléctrica de una neurona.

Inyecciones de corriente dentro del citoplasma provenientes de las sinapsis, pueden producir efectos exitatorios o inhibitorios (Hodgkin, 1952b). Los pulsos exitatorios que logran una despolarización mayor que la tensión de umbral,

generan un pulso eléctrico de salida que crece exponencialmente hasta saturar. Como puede verse en la fig. 4.2, la respuesta del sistema, por debajo del nivel de disparo, satura en unos pocos milisegundos. Dicho funcionamiento puede ser modelado a través de una red R-C en la analogía de componentes eléctricos.

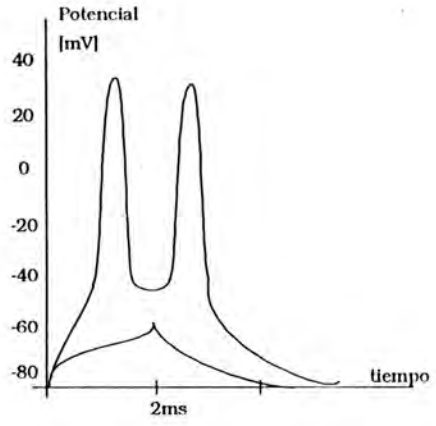

Figura 4.2 curva temporal de depolarización de la membrana

4.3 ELECTRÓNICA ANALÓGICA PARA REDES NEURONALES ARTIFICIALES.

No todos los modelos neuronales artificiales se ajustan a modelos neuronales biológicos. Sin embargo, el conocimiento de los modelos biológicos ha servido de base para crear nuevas arquitecturas computacionales. En esta sección y subsiguientes se analizan algunos circuitos electrónicos analógicos básicos para la construcción de redes neuronales artificiales. Si bien todos ellos han sido utilizados largamente en diseños de sistemas electrónicos analógicos, son tratados, aquí, como bloques constitutivos de redes neuronales artificiales.

La aritmética de funciones utilizadas en modelos de redes neuronales artificiales es variada y dependiente de la arquitectura usada para modelar una neurona. Sin embargo, operaciones tales como suma, resta y multiplicación se encuentran presentes en todas ellas. Otras funciones corrientes son integración, derivación y funciones no lineales con curvas de transferencia continuas o discontinuas. Dentro de las primeras, las funciones más utilizadas

son las de forma sigmoidea y gaussiana, las discontinuas emplean la función signo o una función similar cuyas salidas posibles son +1 y -1 lógicos (o también +1 y 0).

Cuando estas operaciones se implementan directamente sobre señales analógicas, generan múltiples inconvenientes. La mayoría de ellos debido a la diferencia entre los modelos matemáticos que se desean implementar y los elementos físicos utilizados en la implementación (Muroga, 1966). Otros problemas son generados por las limitaciones de la tecnología utilizada, tales como, capacidad de integración en una superficie de silicio, disipación máxima de potencia, velocidad de procesamiento, etc.

En implementaciones en silicio, existe cierta dispersión en los parámetros de los dispositivos utilizados, aun cuando estos estén ubicados cerca para mantener condiciones de temperatura uniforme. La convergencia (fan-in) y divergencia de señales (fan-out), es limitada, y el copiado múltiple de señales, imperfecto. Este problema se acentúa si se desea emular el número de ramificaciones que sufren las señales en una red neuronal (una neurona biológica tiene un árbol dendrítico de 1000 a 10.000 sinapsis).

Las redes neuronales artificiales poseen menos entradas que las redes biológicas, en un factor de varios órdenes de magnitud, debido a limitaciones actuales de la tecnología electrónica. Sin embargo, el aumento de la cantidad de entradas y la conectividad entre unidades computacionales es deseable. Abu-Mostafa(1988a,b) define dos factores importantes, el factor analógico y la entropía, que refuerzan esta aseveración.

La potencia de cómputo, en modo analógico, es proporcional al cuadrado del número de entradas (K); mientras que el tamaño de la unidad de cómputo es lineal respecto a K.

El factor de entropía se basa en un medida del desorden en que se encuentra el ambiente del cual las entradas se nutren de información. Si bien las neuronas reciben información *local*, el número y la zona del mapa de configuración que

las entradas deben cubrir debe ser mayor que la entropía del ambiente (Abu-Mostafa, 1988b).

A pesar de los inconvenientes mencionados, la computación analógica de señales y su aplicación a la implementación de modelos neuronales es suficientemente atractiva para compensar estos problemas. Básicamente, debido a la enorme potencia de cómputo que puede alcanzarse con la implementación analógica. Además, es razonable pensar que los problemas tecnológicos enumerados van a ser superados o al menos reducidos con el avance de la tecnología en los años venideros. Por el momento, la tecnología más utilizada para la construcción de dispositivos VLSI orientados a redes neuronales es la CMOS. Las ventajas más importantes que posee esta tecnología son:

- el proceso de fabricación es simple y está bien desarrollado
- el nivel de integración es muy alto
- los transistores MOS tienen una impedancia de entrada muy alta, bajo consumo y potencia y bajo nivel de ruido
- pueden integrarse capacitores y resistencias con buena precisión y bajo costo
- pueden implementarse llaves (switches) casi ideales
- se puede combinar electrónica digital y analógica en un mismo dispositivo

4.4 VLSI EN REDES NEURONALES

4.4.1 IMPLEMENTACION EN REDES ANALOGICAS

Todos los modelos neuronales requieren la multiplicación de señales analógicas (Mead,1989,1990). La mayoría de ellas efectúa el producto de una señal de entrada a la neurona por un coeficiente estático. Este coeficiente permanece inalterado durante la faz de cómputo o procesamiento de la red neuronal, pero debe ser modificable durante la faz de entrenamiento. En Perceptrones, MLP o RBF, representan los coeficientes con que se pesan las

entradas en cada capa (oculta o de salida). En redes tipo Hopfield (Tank,1987) son los coeficientes de la matriz de correlación que permite definir los atractores en la superficie de energía (Bruck,1988).

Para lograr este efecto multiplicativo puede utilizarse un multiplicador analógico basado en un amplificador de transconductancia diferencial. El circuito de la figura 4.3 muestra un amplificador diferencial cuyas corrientes de emisor están fijadas por una fuente de corriente Q_o. Asimismo, el valor de esta corriente de polarización puede verse como un coeficiente que multiplica a una variable de entrada.

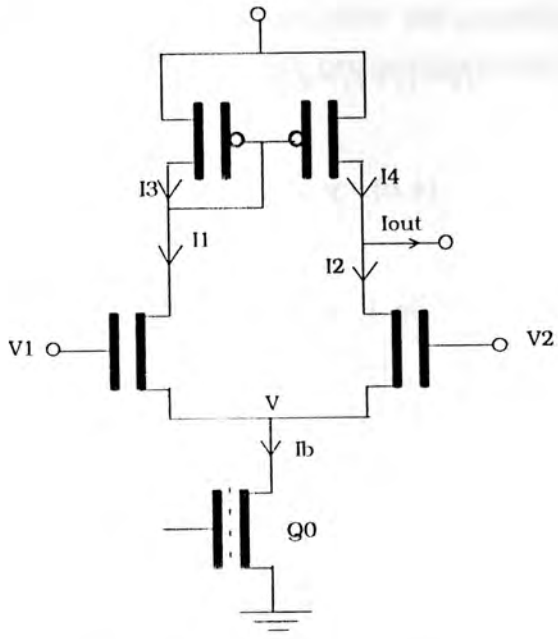

Figura 4.3 amplificador de transconductancia

En el transistor MOS la conducción de corriente es debida a dos efectos característicos: difusión y desplazamiento. La compuerta del MOS actúa de manera de generar un efecto capacitivo entre dicho terminal y el sustrato. Dependiendo de la tensión de compuerta se generan distintos procesos físicos, analizados en el Apéndice B, que se denominan *empobrecimiento* e *inversión*. El comportamiento de la corriente de drenador es diferente para ambas zonas e inclusive pueden determinarse sub-regiones.

En los diseños realizados y en los circuitos analizados en este capítulo utilizaremos tecnología CMOS con transistores trabajando en la zona de *inversión débil* (también llamada zona *subthreshold*) (Apéndice B) (Mead, 1989) (Tsividis, 1977.) (Jacoboni, et al, 1977) (Hoeneisen, 1972) (Massobrio, 1993).

Para esta zona la ecuación de corriente de saturación en un MOS es exponencial con V_{GS}. Si no consideramos el efecto de sustrato y el efecto Early (Early, 1952),

$$I_1 = I_0 e^{(V_1 - V)/V_t} \qquad (4.4) \quad \text{y} \qquad I_2 = I_0 e^{(V_2 - V)/V_t} \qquad (4.5)$$

además

$$I_b = I_1 + I_2 = I_0 e^{-V/V_t}\left(e^{V_1/V_t} + e^{V_2/V_t}\right)$$

$$I_1 = I_b \frac{e^{V_1/V_t}}{e^{V_1/V_t} + e^{V_2/V_t}} \qquad (4.6) \quad \text{y} \qquad I_2 = I_b \frac{e^{V_2/V_t}}{e^{V_1/V_t} + e^{V_2/V_t}} \qquad (4.7)$$

la fig. 4.4 indica la forma de I_1 e I_2 en función de la tensión diferencial de entrada

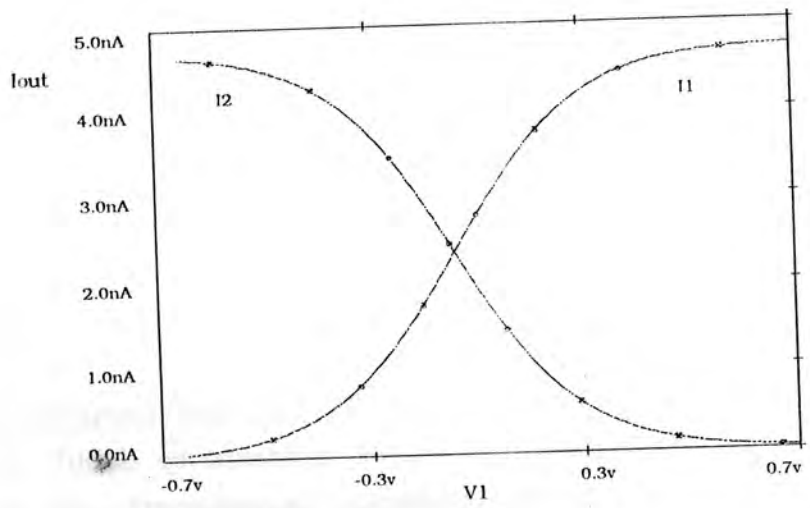

Figura 4.4 curvas de corriente en el amplificador de transconductancia

Restando las ecuaciones (4.6) y (4.7) obtenemos que la corriente diferencial de salida I_{1-2} es proporcional al producto de Ib y de la tensión diferencial de entrada,

$$I_1 - I_2 = I_b . \tanh\left(\frac{V_1 - V_2}{2V_t}\right) \qquad (4.8)$$

Para valores pequeños de entrada (V1-V2) la ecuación (4.8) puede aproximarse por la ecuación lineal

$$I_1 - I_2 = \frac{q I_b}{2kT} . (V_1 - V_2) \qquad (4.9)$$

vemos que la corriente diferencial de salida es proporcional al producto de dos términos, la tensión diferencial de entrada y la corriente de polarización del diferencial o también, la transconductancia del amplificador. Utilizando como base este circuito, podemos construir un multiplicador de cuatro cuadrantes (fig. 4.5).

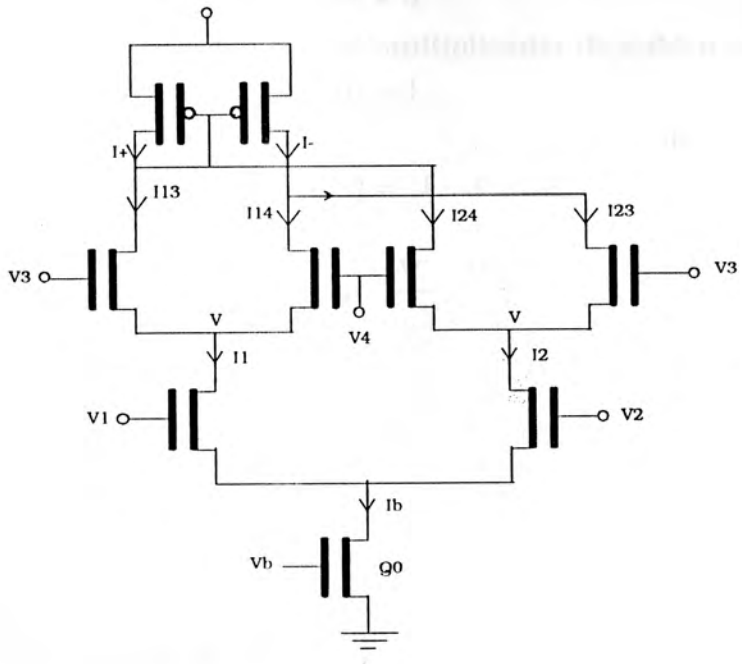

Figura 4.5 multiplicador analógico de cuatro cuadrantes

En este caso las corrientes en las ramas I_1 e I_2 son:

$$I_1 = I_b \frac{e^{V_1/V_t}}{e^{V_1/V_t} + e^{V_2/V_t}} = \frac{I_b}{2}\left(1 + tgh\frac{V_1 - V_2}{2}\right)$$

$$I_2 = I_b \frac{e^{V_2/V_t}}{e^{V_1/V_t} + e^{V_2/V_t}} = \frac{I_b}{2}\left(1 - tgh\frac{V_1 - V_2}{2}\right)$$

Las corrientes de salida I+ e I- se forman sumando y restando las corrientes de las ramas I_{13}, I_{14}, I_{23} e I_{24}:

$$I_{13} = \frac{I_1}{2}\left(1 + tgh\frac{V_3 - V_4}{2V_t}\right) \qquad I_{14} = \frac{I_1}{2}\left(1 - tgh\frac{V_3 - V_4}{2V_t}\right)$$

$$I_{23} = \frac{I_2}{2}\left(1 + tgh\frac{V_3 - V_4}{2V_t}\right) \qquad I_{24} = \frac{I_2}{2}\left(1 - tgh\frac{V_3 - V_4}{2V_t}\right)$$

Luego

$$I_+ = I_{13} + I_{24}$$

$$I_- = I_{14} + I_{23}$$

$$I_{out} = I_+ - I_- = I_{13} + I_{24} - I_{14} + I_{23}$$

$$I_{out} = I_b \cdot \text{tgh}\frac{V_1 - V_2}{2V_t} \cdot \text{tgh}\frac{V_3 - V_4}{2V_t} \qquad (4.10)$$

Las curvas de la figura 4.6 fueron obtenidas por simulación, utilizando la tensión diferencial V_1-V_2 como parámetro. Puede observarse que la corriente de salida mantiene una relación lineal respecto de la variable independiente V_3-V_4 en un rango dinámico reducido (aproximadamente 2Vt) en el cual la función **tgh(x)** puede substituirse por **x.** De todas maneras esta falta de linealidad puede actuar en forma favorable limitando el aporte de corriente que el multiplicador aporta a la próxima etapa. En general, una neurona efectúa una suma algebraica de múltiples salidas de los multiplicadores, en consecuencia, la saturación en la característica de salida del multiplicador evita que alguna entrada fuera de rango (p.ej. con ruido) tenga una influencia decisiva, con lo cual se aumenta la robustez del sistema. En promedio cada neurona aporta una corriente igual a 1/2N del total, siendo N el número de sinapsis del sistema.

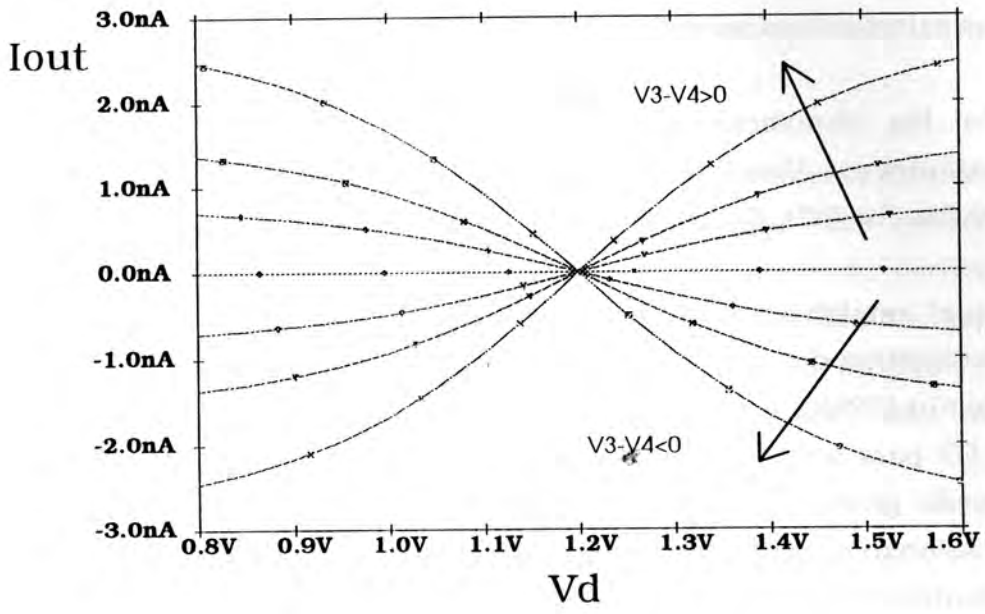

Figura 4.6 curvas de corriente del multiplicador de cuatro cuadrantes

4.4.2 LIMITES DE OPERACION

Si bien la computación de señales analógicas es muy atractiva en la implementación de redes neuronales artificiales, deben solucionarse problemas típicos del diseño de circuitos VLSI. Básicamente, debido a la disparidad de las características de los transistores y al apartamiento de un transistor MOS respecto del modelo ideal como fuente de corriente. Esta última circunstancia, genera una pendiente no nula en la característica de salida y ocasiona limitaciones en el rango de tensiones, ya que los transistores pueden salir de la zona de saturación (Mead, 1989) (Tsividis, 1977.).

La tecnología CMOS de 1µ posee una disparidad típica en los valores de I_o de los transistores, teóricamente idénticos, de un 20%, aunque valores de hasta un 100% son posibles inclusive en transistores ubicados físicamente cerca en la pastilla. Esto equivale a unos ±10mV en tensión de compuerta del MOS. En consecuencia, se favorece el diseño de circuitos autocompensados por técnicas

de realimentación o aquellas configuraciones circuitales donde se promedian los valores dispares en los parámetros.

En las simulaciones realizadas en Spice se utilizaron los siguientes parámetros: Vto=0.8 para NMOS y -0.8 para PMOS, γ=0.68, λ=7exp(-6), ϕ =0.56, T_{ox}=200, K_p=30exp(-6).

En el amplificador de transconductancia de la figura 4.3 la disparidad en los parámetros del amplificador diferencial disminuye el factor de rechazo de modo común (FRMC) generando una diferencia de corrientes en los drenadores de Q1 y Q2 para señales de entrada de modo común puro. Además, la carga activa puede generar problemas debido a que las diferencias en los parámetros ocasionan una pérdida de precisión en la reflectividad del espejo de corriente (Roubik, 1986).

Un amplificador de transconductancia diseñado únicamente con transistores MOS de canal N posee una ganancia en modo diferencial aproximada por:

$$A_{dd} = -g_m/g_l \qquad \text{y} \qquad g_l = g_{ml} + g_{dl} + g_{mbl} + g_d$$

donde g_m es la transconductancia directa del diferencial y g_l es la conductancia de carga, que engloba la conductancia de salida del transistor del par diferencial g_d, y las conductancias del transistor del espejo de corriente que actúa de carga. Estas son: la transconductancia directa g_{ml}, la conductancia de salida, g_{dl}, y la conductancia del sustrato, g_{mbl}. El término dominante en g_l es g_{ml}, en consecuencia, la ganancia diferencial puede aproximarse como un cociente de dos transconductancias o, también, por

$$A_{dd} = -\sqrt{\frac{(W/L)_i}{(W/L)_l}}$$

Puede verse, entonces, que el aumento de ganancia se logra a costo de aumentar el área del transistor del diferencial. Es decir, esta configuración no permitirá ganancias mayores que 10 o 12. Si se utilizan NMOS de empobrecimiento como carga activa, pueden unirse los terminales de

compuerta y fuente ($V_{GS}=0$) en lugar de compuerta y drenador ($V_{GD}=0$) consiguiendo la eliminación de g_{ml} de la expresión de g_t.. Esto aumenta la ganancia diferencial, al menos, en un factor diez.

El FRMC del amplificador de transconductancia es directamente proporcional a la transconductancia directa g_m de los transistores del diferencial e inversamente proporcional a la conductancia de la fuente de corriente que provee la polarización al par diferencial. Debido a que éste es uno de los factores de mérito más importantes en el diseño de amplificadores y multiplicadores utilizados en redes neuronales, puede ser necesario implementar algunas mejoras.

$$FRMC \propto 2\frac{g_{ml}}{g_o}$$

El empleo de transistores NMOS de deplexión no mejora el FRMC debido a que tanto A_{dd} como A_{cc} dependen de g_t.. En cambio, si disponemos de tecnología CMOS, pueden utilizarse transistores PMOS para el espejo de corriente. En dicho caso, los substratos separados para los NMOS y PMOS se conectan a V_{ss} y V_{dd}, respectivamente, eliminando las tensiones V_{BS} y el efecto del substrato (*body effect*). El FRMC aumenta en un factor proporcional al cociente entre la transconductancia del transistor de carga y la conductancia de salida del transistor del diferencial g_{ml}/g_{dt}. Este factor es, normalmente, mucho mayor que uno.

$$FRMC \propto 2\frac{g_{ml}\cdot g_{ml}}{g_o\cdot g_{dt}}$$

Otra posibilidad es utilizar un circuito adicional que actúe como regulador de la tensión de modo común en la salida del diferencial. Este circuito mide el modo común a la salida y actúa sobre la fuente de corriente disminuyendo dichas variaciones.

4.4.3 EL PROBLEMA DE LA TENSION DE SALIDA

Cuando se implementan redes neuronales con circuitos electrónicos analógicos deben acoplarse etapas de procesamiento. En nuestro caso, la salida del amplificador de transconductancia o multiplicador analógico alimenta alguna otra etapa, tal como, un circuito sumador o un integrador, etc. La tensión de salida del amplificador diferencial, V_{out} (fig. 4.3), está determinada en parte por la entrada de la etapa siguiente, por lo cual es deseable que el amplificador funcione para un amplio rango de valores de V_{out}.

La pendiente de la característica I_{out}-V_{out} es la conductancia de salida del amplificador diferencial g_{out}, que posee un valor finito cuasi constante para un amplio rango de V_{out} (fig. 4.7). Si suponemos que la tensión V_{out} es controlada externamente encontramos un límite de variación superior e inferior. Aumentando V_{out}, la corriente de salida disminuye lentamente hasta que V_{out} supera el valor de V_{DD}, en ese caso el transistor Q4 invierte su funcionamiento, se intercambian los terminales de fuente y drenador y el transistor invierte el sentido de la corriente, tomando corriente del nodo I_{out}, es decir, la corriente de salida decrece exponencialmente.

El límite inferior pone una condición más rigurosa. La tensión del nodo V, en funcionamiento normal, se encuentra entre V_1 y V_b; llamando V_1 al mínimo entre V_1 y V_2. Supongamos que $V_1 >> V_2$, En este caso $V=V_1-V_b$, ya que Q_1 se lleva toda la corriente de Q_b. Si ahora variamos externamente el valor de V_{out} hasta un valor inferior al potencial de V, se intercambian los terminales de fuente y drenador de Q_2 y ambos Q_4 y Q_2 aportan corriente en un mismo sentido al nodo I_{out}. Esto ocasiona un incremento exponencial de I_{out} luego que V_{out} haya descendido por debajo de V_2-V_b, momento en el cual el monto de I_2 es comparable al de I_1. Idéntico resultado se obtiene en el caso en que $V_2 >> V_1$. En consecuencia, la mínima tensión V_{out} a la que el amplificador funciona correctamente es:

$$(V_{out})_{min} = \min(V_1,V_2)-V_b \qquad (4.11)$$

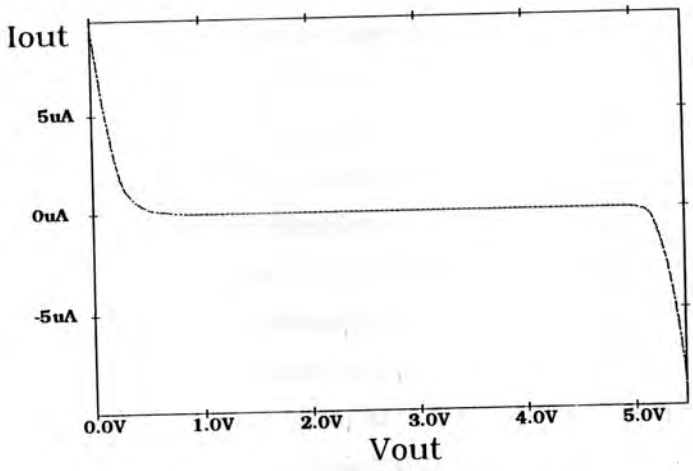

Figura 4.7 corriente de salida vs. tensión de salida

4.4.4 AMPLIFICADORES Y MULTIPLICADORES DE AMPLIO RANGO

Para aumentar el rango dinámico de entrada debe eliminarse el problema de la tensión de salida en el amplificador de trasconductancia. Esto puede lograrse utilizando espejos de corriente que trasladan el nodo de salida V_{out}. Puede verse en la fig. 4.8 que la corriente del drenador de Q_1 es reflejada a través del espejo formado por Q_3 y Q_4; y la corriente del drenador de Q_2 es reflejada dos veces primero por Q_5-Q_6 y luego por Q_7-Q_8. Los drenadores de Q_4 y Q_8 forman el nodo de salida.

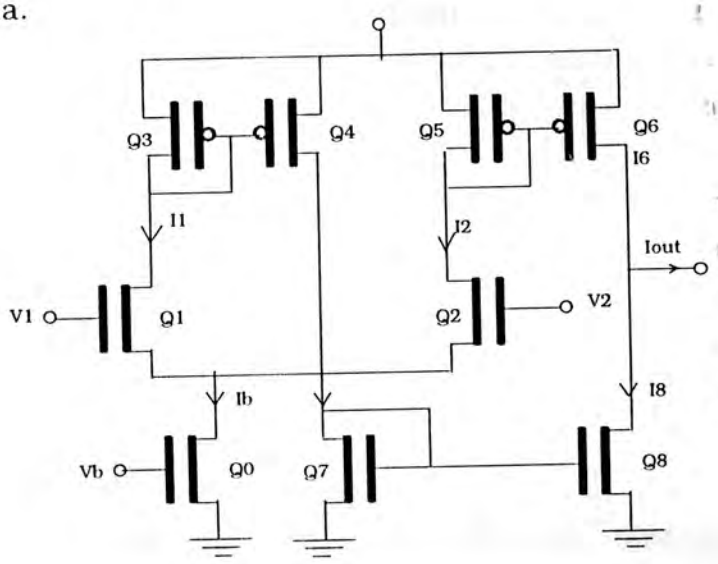

Figura 4.8 amplificador de amplio rango de tensión de salida

El rango de tensiones diferenciales de entrada V_1-V_2 es ahora independiente de V_{out}. Si bien Q_1 y Q_2 siguen siendo afectados por los potenciales de sus drenadores respecto de sus tensiones de compuerta, se mantienen a una tensión casi constante y cercana a V_{DD} debido a que ambos tienen una conductancia de carga pequeña y de valor constante. En otras palabras, Q2 ya no tiene el problema asociado a la conductancia de carga en su drenador y ahora mantiene una tensión entre fuente y drenador similar a la de Q1. En este circuito, solo Q4 y Q8 trabajan sobre un rango amplio de tensiones en drenador, y puede aumentarse la longitud de sus canales, L, lo suficiente como para mantener una baja conductancia de drenador.

En el caso del multiplicador el problema es aún mayor. Si el nodo de salida es cargado por una etapa que influye fuertemente en el valor de V_{out}, este estará condicionado por las entradas V_3 y V_4 de manera que $(V_{out})_{min} = \min(V_3,V_4)-V_b$. Además, las tensiones de los drenadores V_L y V_R, del par diferencial inferior deben satisfacer las condiciones de V_{min} respecto de sus entradas, V_1 y V_2,

$$(V_L,V_R)_{min} = \min(V_1,V_2)-V_b.$$

V_L y V_R dependen de V_3 y V_4, dado que V_L y V_R toman el valor del mayor entre V_3 y V_4. Es decir, $(V_L,V_R) = \max(V_3,V_4)-V_b$. Igualando ambas ecuaciones,

$$\max(V_3,V_4) > \min(V_1,V_2) \qquad (4.12)$$

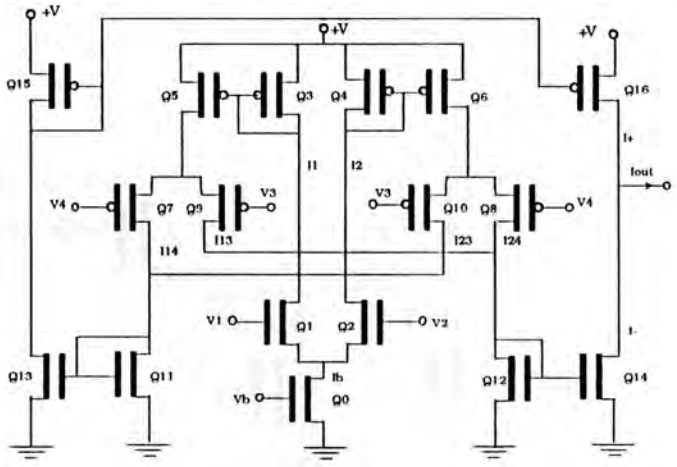

Figura 4.9 multiplicador de cuatro cuadrantes y amplio rango de V_{out}

Si pensamos en valores arbitrarios de entradas para V_1, V_2, V_3 y V_4 el multiplicador pierde linealidad cuando la ecuación (4.11) no se cumple.

La solución a este problema es similar a la utilizada para el amplificador de transconductancia, es decir, independizar los potenciales de V_{out}, V_L, y V_R de las entradas V_1, V_2, V_3 y V_4 utilizando espejos de corriente. El circuito de la figura 4.9 logra ese objetivo con una cantidad de transistores que no alcanza al doble de los utilizados en el circuito original.

4.4.5 FUNCIONES NO LINEALES

Si bien los circuitos ya analizados poseen un comportamiento no lineal en todo su rango de aplicación, son utilizados, preferentemente, en la porción considerada lineal. Es decir, aquella en la cual pueden despreciarse las no linealidades introducidas debido a su influencia casi nula en el cómputo de la señal de salida. De todas maneras, las no linealidades ocasionadas por transistores trabajando en la zona de corte o saturación, pueden resultar altamente beneficiosas, aportando una cuota de tolerancia a fallos o robustez, en sistemas con un gran número de conexiones (sinapsis); moderando el efecto de entradas ruidosas que se hallen fuera del rango normal de trabajo.

Sin embargo, no es este el principal motivo para la utilización de funciones no lineales en redes neuronales. Dichas funciones forman parte explícita del modelo neuronal. Como ha sido analizado en el capítulo 2, las redes neuronales proporcionan una transformación entrada/salida de $\Re^n$ en $\Re^m$. Las no linealidades del modelo neuronal son necesarias en la aproximación de funciones continuas o mapas de entrada/salida. Si bien puede utilizarse una función de salida lineal, la capacidad de cómputo de este tipo de modelos neuronales está limitado a la clasificación de patrones de entrada linealmente separables.

Las funciones de salida no lineales más utilizadas difieren notablemente. Algunas son discontinuas, otras continuas pero no derivables y un tercer grupo

de funciones *suaves* (continuas y derivables). Las funciones más representativas de cada grupo son, la función signo, la función lineal a tramos y la función tipo sigmoide, respectivamente. Estas son las que mayor atención han concentrado hasta el presente, fundamentalmente, debido a que el algoritmo de retropropagación utiliza una función de esta tipo. Sin embargo existen muchas funciones base, de utilidad en la teoría de aproximación de funciones y reconocimiento de patrones. Cada función base tiene ciertas propiedades que la hacen ápta para un dado dominio de aplicaciones. Últimamente, la función del tipo gaussiana ha sido favorecida por resultados alentadores en aproximación de funciones (Poggio, 1990) (Hush, 1993). Las funciones gaussianas son particularmente atractivas en casos donde las neuronas pueden actuar sobre una campo visual reducido del espacio total de entrada. Es decir cada patrón de entrada es local en el sentido que afecta una cantidad reducida del total de neuronas de la red. Un ejemplo típico de aplicación de este tipo de neuronas es el de visión artificial, donde cada neurona recibe información de un campo visual reducido y redundante respecto de las neuronas vecinas.

Debido a esto se considera que la generación de neuronas con funciones de activación gaussiana son escenciales en el desarrollo de la computación paralela masiva. El problema de visión artificial sólo puede ser resuelto en tiempo real por la computación masiva de unidades paralelas. En lo que resta de este capítulo se proponen circuitos para la implementación de funciones de base radial con activación gaussiana.

4.4.5.1 SIGMOIDE DE SALIDA CON GANANCIA VARIABLE

La función de salida tipo sigmoide es continua y crece en forma monótona desde valores cercanos a cero para entradas altamente negativas asintóticamente hacia uno para entradas grandes y positivas. Matemáticamente puede ser representada por funciones del tipo

$$f(x) = \frac{1}{1+e^{-\beta x}} \qquad \text{o bien} \qquad f(x) = tgh(\beta x)$$

Circuitalmente, puede ser representado con un amplificador de transconductancia como el analizado en 4.4.1.

La tensión de salida es proporcional a la transconductancia del amplificador, a través de la corriente de polarización entregada por la fuente de corriente, y a la tensión diferencial de entrada por medio de la ecuación

$$V_{OUT} = I_b / g_l . tgh\left(\frac{V_d}{V_T}\right)$$

donde Ib es la corriente de polarización y gl la conductancia de carga del amplificador.

La transconductancia diferencial g_m puede obtenerse derivando I_{out} respecto de V_d.

$$g_m = \frac{d\, I_{OUT}}{d\, V_d} = 4\frac{I_b}{V_T} . \sec h\left(\frac{V_d}{V_t}\right)$$

en condiciones de polarización (Vd=0), g_{mo} es directamente proporcional a la corriente de polarización I_b. En consecuencia I_b funciona como una ganancia variable en la transferencia de la sigmoide, también llamado "temperatura" de la sigmoide en analogía con el modelo

4.5 IMPLEMENTACION DE UNA NEURONA RBF CON FUNCION DE ACTIVACION GAUSSIANA

En la sec.2.8 se definió la estructura de una neurona gaussiana como una combinación lineal de funciones base

$$G(x,w) = \sum_i c_i g_i(x,w) \qquad (4.13).$$

Cada función base es de la forma

$$g_i(x,w) = e^{-(x-w)^2/\sigma_i^2} \qquad (4.14)$$

donde **x** y **w** son vectores n-dimensionales, c_i es un coeficiente que pesa las funciones base en la salida, y σ_i es un coeficiente que modula el ancho de cada función base. Puede verse que la función toma un valor máximo, e igual a **1**, para entradas **x** que coinciden con el valor central **w** y luego decae rápidamente en función de la distancia entre **x** y **w**. El parámetro σ_i actúa de modo de controlar la incidencia de esta distancia en el exponente de la función exponencial negativa. Es decir, es la desviación estándar de la función gaussiana que, geométricamente hablando, modula el ancho de la función.

El circuito diseñado (fig. 4.10) para implementar la función de base gaussiana responde al siguiente diagrama en bloques:

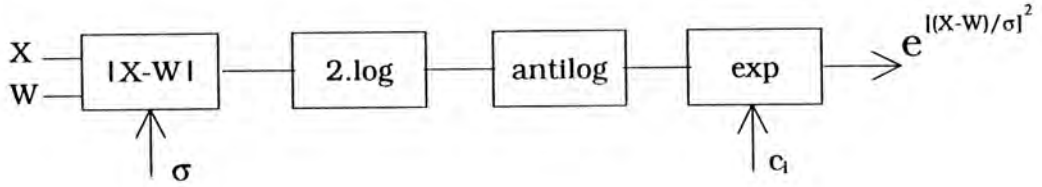

Figura 4.10 diagrama en bloques

Como se dijo, todos los transistores MOS utilizados trabajan en la zona de inversión débil (apéndice B), en la cual la corriente de drenador es función exponencial de la tensión de control V_{GS}.

El primer bloque calcula la función valor absoluto de la distancia (norma euclidea) entre los vectores **x** y **w**. Se genera una corriente diferencial de salida, I_{sali}, en función del módulo de la diferencia de tensiones de entrada. Es importante que la salida de corriente de este bloque sea siempre en el mismo sentido (signo positivo) para facilitar el diseño de los módulos que siguen a continuación.

La figura 4.11 muestra dos amplificadores diferenciales cuyas transconductancias pueden ser modificadas variando la corriente de polarización que generan Q_{b1} y Q_{b2}. Los amplificadores A_1 y A_2 son idénticos a los descriptos en 4.4.1. Las entradas son x_i y w_i, componentes de los vectores **x** y **w**; y están invertidas en A2 respecto de A1. La salida I_{sali} se obtiene

pasando por dos espejos de corriente de canal-p. Si V1 > V2, A1 genera una corriente negativa (hacia afuera) que es reflejada por el espejo formado por Q5 y Q6, en cambio A2 intenta generar una corriente positiva, pero la tensión de salida crece hacia VDD cortando al espejo Q11-Q12. Es decir, cada rama funciona como un rectificador de media onda con entrada diferencial y el conjunto como un rectificador onda completa. Entonces, la corriente de salida Isal1 es:

$$I_{sal1} = I_b . th\left(\frac{\kappa|x_t - w_t|}{2}\right) \qquad (4.15)$$

Esta función puede aproximarse, en un rango de tensiones de entrada de ± 300mV, por la función,

$$I_{sal1} = \frac{\kappa I_b}{2} . |x_t - w_t| \qquad (4.16)$$

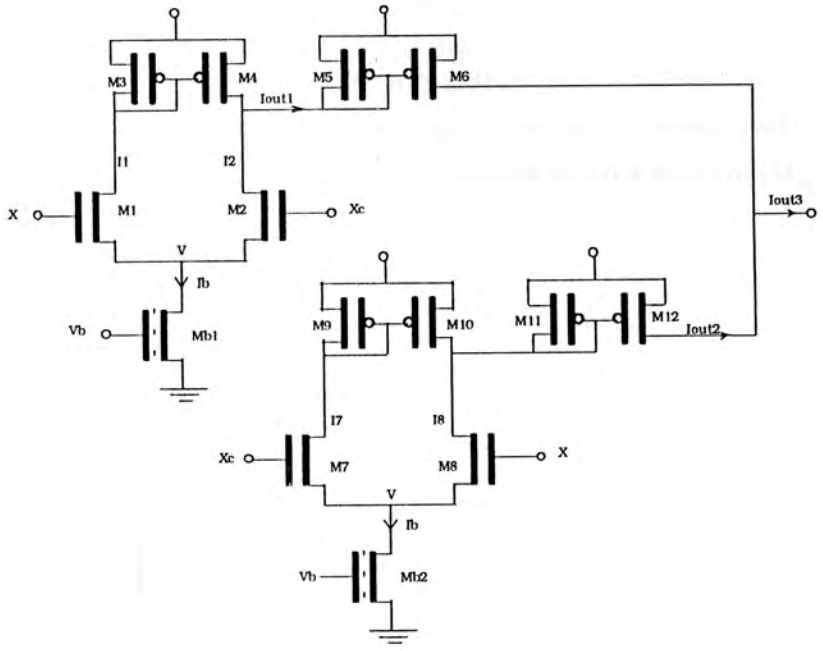

Figura 4.11 función módulo de una tensión diferencial de entrada

La figura 4.12 muestra la salida de simulación de este bloque circuital para distintos valores de I_b de polarización de los amplificadores diferenciales. Puede utilizarse dicha corriente para controlar el parámetro σ de la expresión (4.14). Es decir, σ proporcional a la inversa de I_b.

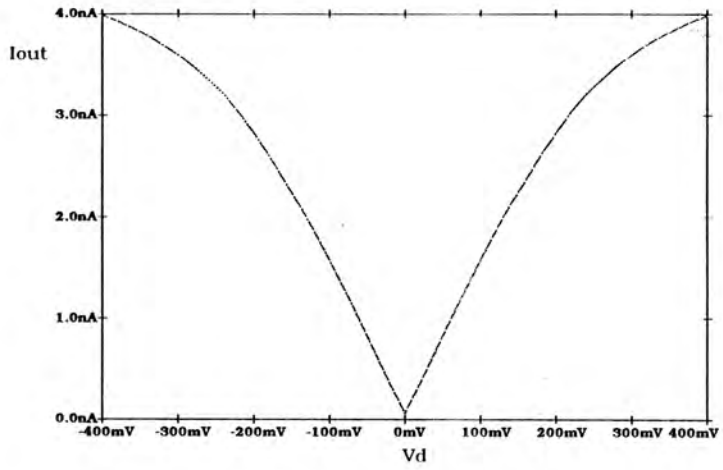

Figura 4.12 I_{out} del bloque que calcula la función módulo de V_d

La función cuadrática se realiza por logaritmación y suma. Los transistores Q_{14} y Q_{15} están conectados en serie, la tensión V_{GS} de cada uno de ellos es una función logaritmica de la corriente de drenador (fig.4.13),

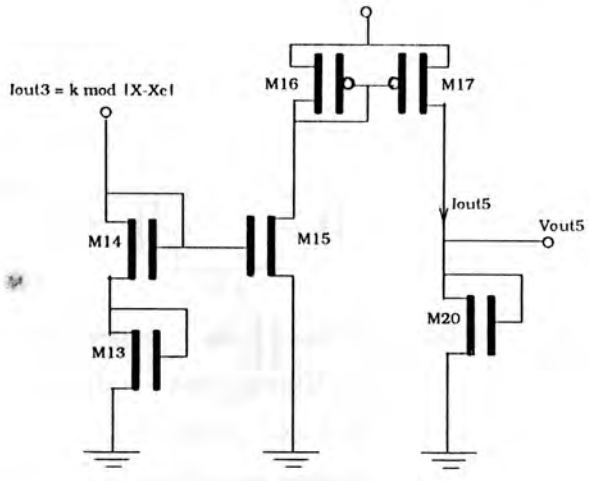

Figura 4.13 función cuadrática en base a logaritmo y exponencial

$$V_{GS} = \frac{kT}{\kappa q}.\ln\left(\frac{I_{sal1}}{I_0}\right) \qquad (4.17)$$

si $I_{01} = I_{02}$ entonces $V_{GS1} = V_{GS2}$, y $V_2 = 2.V_{GS}$. Por lo tanto obtenemos un doblador de tensión; y dado que dicha tensión es el logaritmo de la corriente, usando la aproximación (4.16),

$$V_{sal2} = 2\frac{kT}{\kappa q}.\ln\left(\frac{\kappa I_b}{2I_0}|x_t - w_t|\right) \qquad (4.18)$$

luego, Q_{15} computa el antilogaritmo de V_{sal2} generando I_{sal3} que responde al cuadrado de la señal de entrada diferencial.

$$I_{sal3} = I_0.\exp(V_{sal2}) = I_0.\exp\left[2\frac{kT}{\kappa q}.\ln\left(\frac{\kappa I_b}{2I_0}|x_t - w_t|\right)\right] \qquad (4.19)$$

$$I_{sal3} = \alpha.I_b.(x_t - w_t)^2 \qquad (4.20)$$

en el cual α representa una constante que engloba a I_0 y κ. La figura 4.14 representa la salida del bloque cuadrático obtenida por simulación

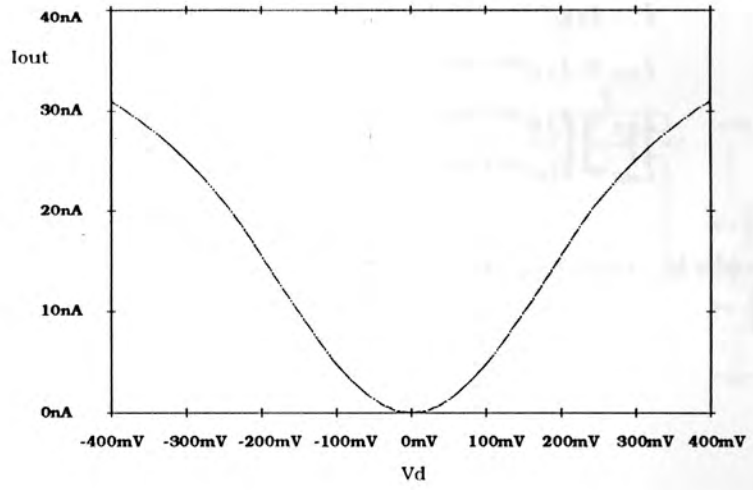

Figura 4.14 corriente de salida del bloque cuadrático

El espejo de corriente formado por los transistores Q_{16} y Q_{17} reflejan la corriente de salida I_{sal3} y la convierten en una tensión sobre la resistencia R_1, generando V_{sal4}. Esta tensión es la entrada para el módulo exponencial formado por los transistores Q_{18} y Q_{19} y la fuente de corriente I (Fig.4.15). El funcionamiento del circuito exponencial es el siguiente: Q_{18} y Q_{19} forman un espejo de corriente en el cual la corriente de salida Iout es proporcional a la corriente de la fuente externa I y a la exponencial negativa de la señal de entrada,

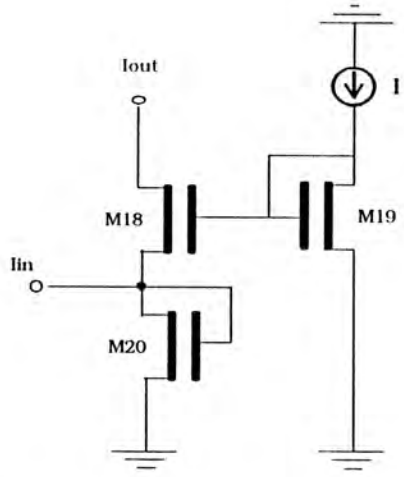

Figura 4.15 bloque exponencial

$$
\begin{aligned}
I &= I_0 e^{q\kappa V_{GS19}/kT} \\
I_{out} &= I_0 e^{q\kappa V_{GS18}/kT} = I_0 e^{q\kappa (V_{GS19}-V_{sal4})/kT} \\
I_{out} &= I_0 e^{q\kappa V_{GS19}/kT} . e^{-q\kappa V_{sal4}/kT} \\
I_{out} &= I . e^{-q\kappa V_{sal4}/kT}
\end{aligned}
\qquad (4.21)
$$

Reemplazando la ecuación (4.20) en (4.21)

$$I_{out} = I . e^{-q\alpha R_1 I_b^2 (x_i - w_i)^2/kT}$$

$$I_{out} = I . e^{-\beta (x_i - w_i)^2/\sigma^2} \qquad (4.22)$$

En ésta última ecuación $\beta = q\alpha R_1/kT$ y σ es la inversa de I_b. La figura 4.16 muestra los resultados obtenidos de la simulación para distintos valores de I_b.

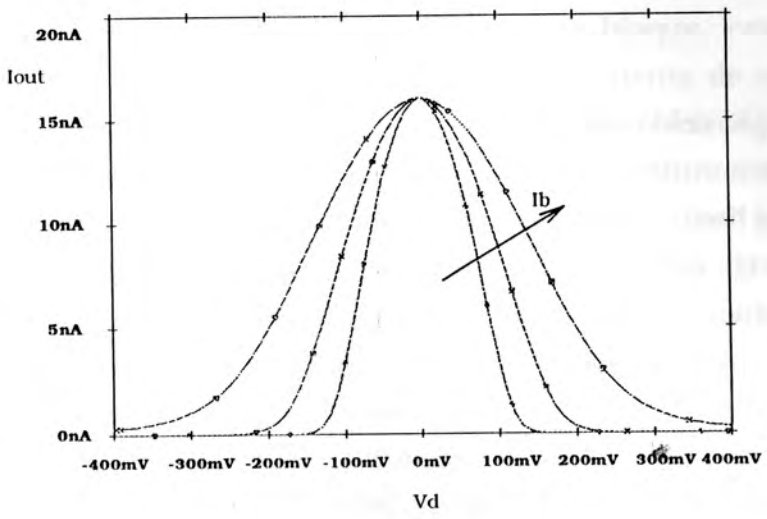

Figura 4.16 salida gaussiana de corriente usando I_b como parámetro

La figura 4.17 representa el esquema circuital completo

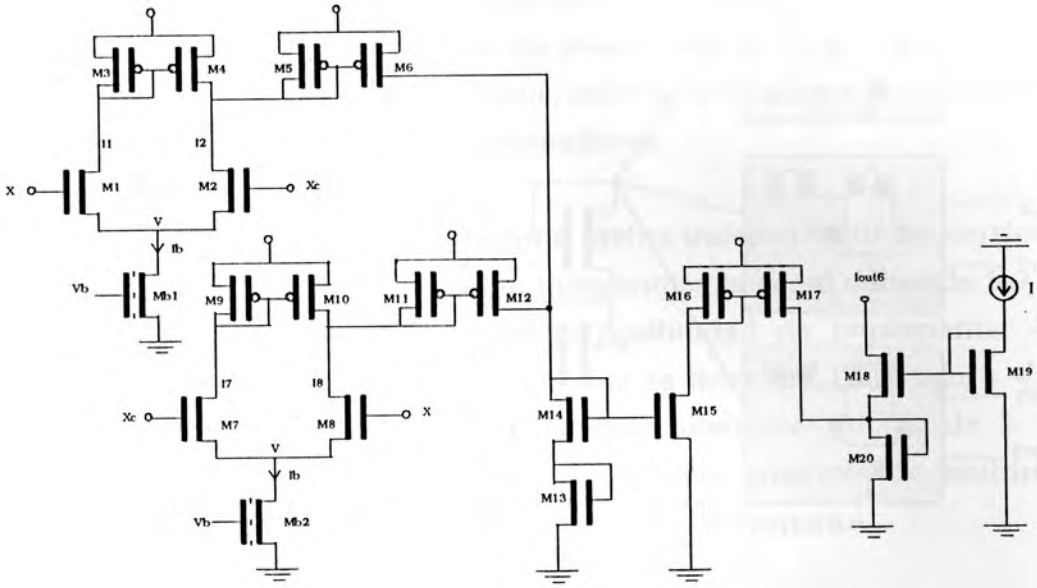

Figura 4.17 circuito completo de una neurona gaussiana de una entrada unidimensional

4.6 CONSIDERACIONES SOBRE EL DISEÑO DE LA CELDA NEURONAL GAUSSIANA

El diseño de la celda neuronal fue hecho para un único par de entradas, pero la extensión a múltiples entradas es inmediata mientras nos mantengamos dentro de los límites del *fan-in* de los transistores que realizan la logaritmación. La figura 4.18 muestra cómo se pueden agregar pares de entradas **xi-wi** a través de bloques de amplificadores diferenciales y espejos de corriente. Obviamente, el agregado de entradas pone una exigencia adicional sobre el bloque logaritmador, ya que este bloque deberá aumentar su rango dinámico de corriente en proporción directa al número de corrientes de entrada de cada bloque $|x_i-w_i|$. Afortunadamente, el bloque de logaritmación propuesto puede computar corrientes en un rango dinámico de 4 órdenes de magnitud; abarcando un intervalo aproximado de [10exp(-11) ; 10exp(-7)] Amp. Limitando la corriente de cada bloque, puede lograrse un número significativo de entradas **x-w**.

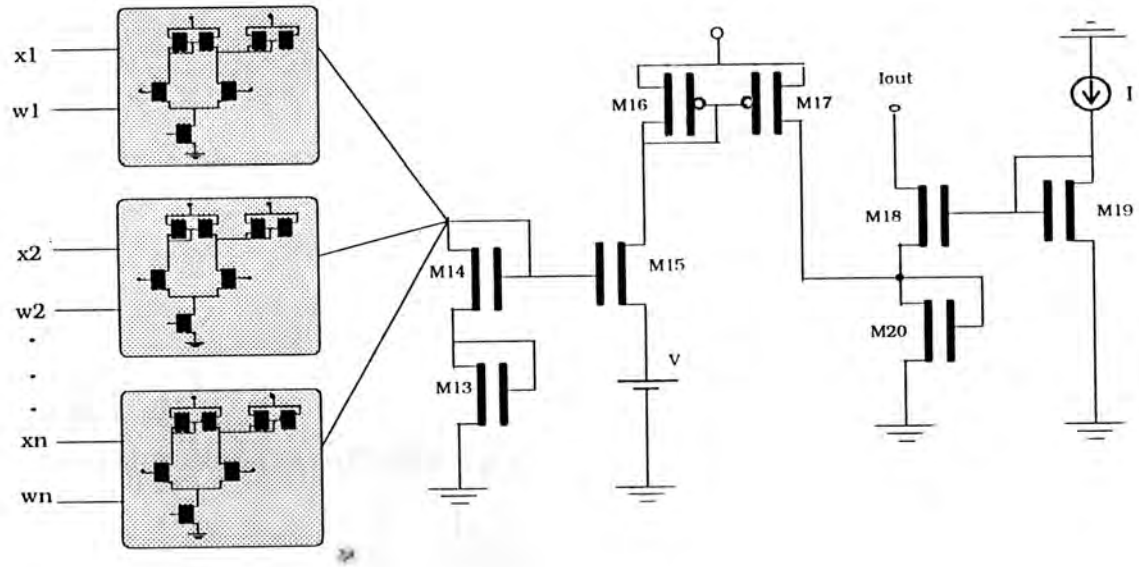

Figura 4.18 circuito completo de una neurona gaussiana con entrada n_dimensional

Otro punto crítico son los posibles errores introducidos por disparidad en los transistores Q_{13} y Q_{14} del bloque logaritmador. Este bloque, como se dijo, genera una tensión (V_{GS}) que es el logaritmo de la corriente de entrada (I_d). Dado que los transistores están en serie, sus corrientes serán iguales, y en caso de que exista un desbalance entre las corrientes de saturación de ambos transistores, el mismo se traducirá como una tensión de error (offset) entre V_{GS1} y V_{GS2}. Una posible solución a este problema consiste en agregar, dentro del bloque de logaritmación, un regulador simple que compense el *offset* por realimentación.

De todas maneras vale la pena aclarar que los errores en una celda neuronal, si no son excesivos, no tienen un rol perjudicial que invalide el diseño realizado. Esto es debido a que las redes neuronales en general y los algoritmos de entrenamiento de estas redes, son adaptivos siguiendo un funcional del error global del mapa que se desea representar. Quiere decir, que los errores pueden ser compensados por el masivo paralelismo de las unidades de proceso durante el entrenamiento de la red. Existe una evidencia formal (Funahashi 1989; otros) de que la no linealidad de la neurona no es fundamental en el problema de representación de un mapa de entrada-salida. En nuestro caso, la celda RBF posee excelentes condiciones de procesamiento local del espacio n-dimensional de entrada. En otras palabras, solo las entradas $\mathbf{x}$ próximas a $\mathbf{w}$ tienen influencia sobre la salida $\mathbf{g_i}$ correspondiente.

En el diseño de la celda RBF se incluye una fuente independiente de corriente **I**. Esta fuente cumple un doble propósito: independiza la señal de salida I_{out} de la corriente de saturación de Q_{18} y da la posibilidad de implementar los coeficientes $\mathbf{c_i}$ de la ecuación (4.13); ya que I_{out} es lineal en **I**. La figura 4.19 representa la salida de simulación de I_{out} para distintos valores de **I**. La representación de la ecuación global (4.13) requiere entonces de múltiples celdas con valores diferentes Ij sumadas en un nodo de corriente.

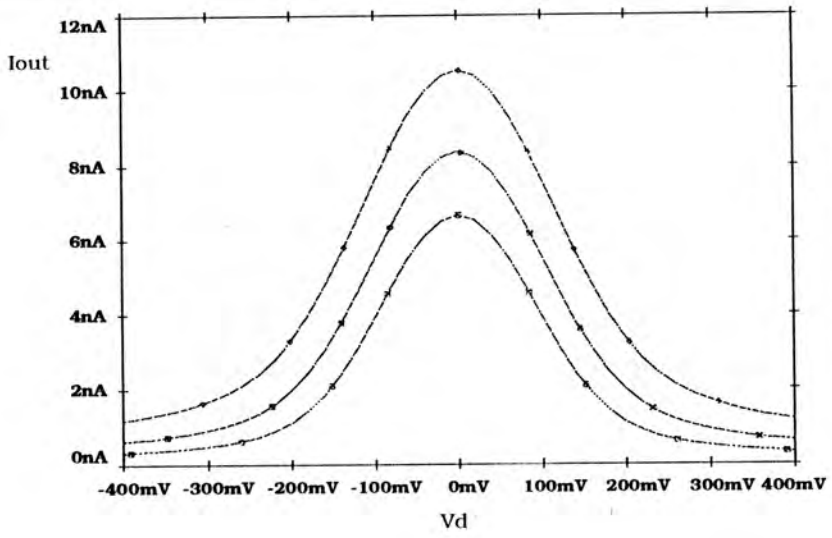

Figura 4.19 variación de la corriente de salida en función del generador I

Esta neurona constituye la unidad de proceso central de una red neuronal basada en funciones de base radial gaussianas. La tecnología CMOS de 1µ (o menor) permite integrar un elevado numero de estas unidades en un mismo dispositivo. Las entradas **x-w** pueden ser compartidas por varias neuronas o agrupadas en la forma de campo visual. Dado que la cantidad de transistores por celda es pequeña, puede pensarse en una gran cantidad de neuronas en un mismo dispositivo. En este caso, la mayor limitación estará en el número de entradas y salidas compatibles con un determinado encapsulado. Además, si se agregan llaves analógicas en la etapa de entrada, puede dársele flexibilización a la arquitectura de la red neuronal.

CAPITULO 5

NETMAP: COMPUTADORA NEURONAL ANALOGICA

5.1 INTRODUCCION

El procesamiento analógico de señales por un conjunto masivo de unidades computacionales paralelas permite resolver eficientemente problemas ingenieriles en tiempo real. En el capítulo 3 de esta tesis se expuso extensamente las ventajas y problemas en la realización de celdas neuronales y neurocomputadoras asociadas a cada una de las técnicas disponibles. Este análisis concluyó con la conveniencia de realizar una computadora neuronal analógica. Entre otras razones han prevalecido: la velocidad de procesamiento de las unidades analógicas, su tamaño reducido en área de silicio, la posibilidad de conectarse a señales sensoriales y actuadoras analógicas directamente, y su mayor proximidad con los modelos biológicos. La figura 5.1 muestra dos ejemplos de como se conecta la computadora analógica en procesamiento de señal y control.

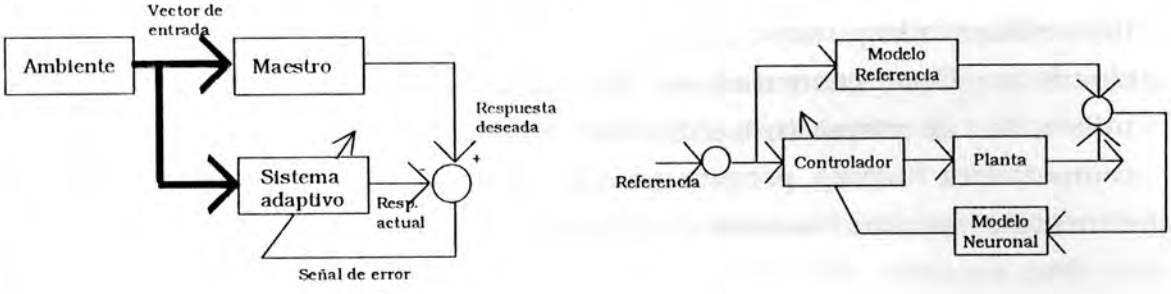

Figura 5.1 Sistemas neuronales: a) sistema adaptivo para procesamiento de patrones. b) control por modelo de referencia.

En ciertosos casos la computadora neuronal procesa directamente los patrones analógicos provenientes de los sensores. En otros casos, especialmente si los patrones de entrada son digitales por naturaleza o surgen del muestreo y

conversión de una señal de entrada, la computadora puede utilizarse como acelerador de una computadora digital. Fig 5.2.

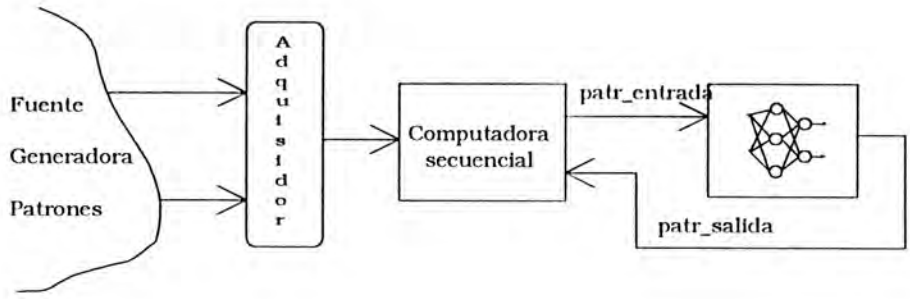

Figura 5.2 sistema neuronal funcionando como acelerador de una computadora secuencial

Las redes neuronales generadas en la computadora neuronal deben ser entrenadas, es decir, debe existir un mecanismo de adaptación de los coeficientes internos de la red neuronal.

En capítulos anteriores se analizó la importancia de generar un sistema neuronal para el procesamiento de patrones a alta velocidad. Este sistema debe poseer unidades de procesamiento tal como la nerona gaussiana realizada en el capítulo 4. El propósito principal de este trabajo radica en crear una neurocomputadora capaz de resolver problemas complejos en tiempo real, no simulado. Esta computadora, denominada Netmap, ha sido organizada utilizando al dispositivo neuronal Etann como base (apéndice C). La computadora Netmap, permite generar redes neuronales estáticas multicapa de dimensión variable, hasta con un máximo de 10 mil sinapsis y 128 neuronas.

Debido a que el sistema Netmap, es un proyecto experimental, se ha puesto énfasis en la flexibilidad que debe poseer. En consecuencia la neurocomputadora ha sido organizada de manera de poder procesar tanto patrones analógicos como digitales. Los patrones digitales deben ser convertidos a analógicos para su procesamiento, debido a la naturaleza analógica del dispositivo de procesamiento neuronal. Dada la velocidad de procesamiento paralelo de la red neuronal, la conversión de dichos patrones digitales impone

un serio requerimiento de velocidad para no degradar el rendimiento de la neurocomputadora.

Los coeficientes de peso de la red neuronal deben ser adaptados durante el algoritmo de aprendizaje. En consecuencia el sistema Netmap ha sido organizado como indica la figura 5.3.

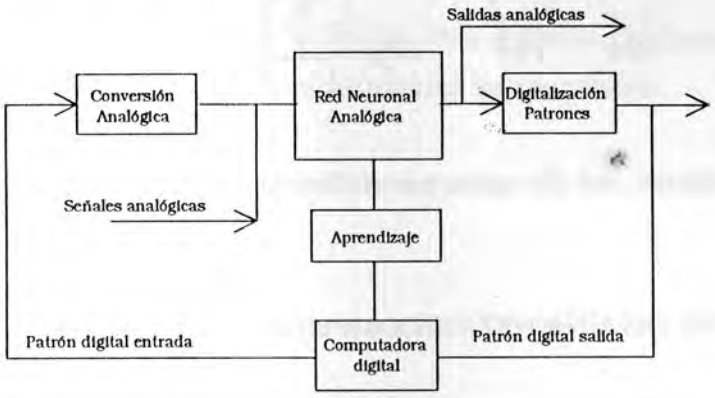

Figura 5.3 Organización del sistema Netmap

La elección del Etann como dispositivo de base para la neurocomputadora se debió a su elevada capacidad de cómputo. En el capítulo 3 se describió, brevemente, la arquitectura interna del Etann. En el apéndice C se realiza una descripción mucho más detallada del mismo. El Etann está orientado a dos modelos neuronales diferentes: el modelo estático perceptrón multicapa y el modelo de Hopfield. Este último es un modelo recurrente y necesita el auxilio de capacitores externos conectados al vector de salida de la red.

El sistema Netmap sólo toma la configuración estática multicapa. Cada dispositivo puede generar una red de dos capas de 64 neuronas cada una, plenamente interconectadas. La potencia de cómputo del sistema, puede aumentarse expandiendo la red neuronal generando una estructura de multiples dispositivos. La figura 5.4a muestra una expansión en dos dimensiones sin necesidad de electrónica de acoplamiento. En este caso varía el esquema de interconexión de las señales de entrada y salida. Asimismo, puede hacerse una expansión manteniendo fija la estructura de entrada y

salida utilizando un canal común, como indica la figura 5.4b. En este caso debe multiplexarse las señales analógicas que van a cada dispositivo.

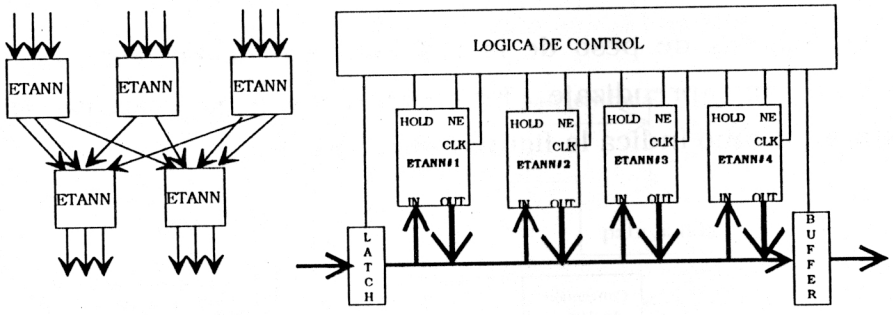

Figura 5.4 Arreglos paralelos: a) en varias capas b) bus comun

5.2 DISEÑO DE UNA NEUROCOMPUTADORA

El diseño de un neurocomputador analógico fue realizado siguiendo varias lineas fundamentales basadas en la flexibilidad y velocidad de procesamiento de patrones. El sistema realizado es una computadora de propósitos generales, es decir posee la flexibilidad de modificar su arquitectura dentro de los límites máximos de su capacidad. Como se dijo en la introducción de este capítulo una computadora analógica puede ser utilizada para procesar señales de la misma naturaleza dentro de un sistema de medición, control, etc. Pero además puede ser utilizada para análisis de datos en tiempo diferido y otras aplicaciones que incluyen a un usuario en el ciclo de procesamiento. Este punto obliga a generar una estructura de soporte que permita que la neurocomputadora esté inserta dentro de un sistema estandar, dado que, de otro modo, debería extenderse el diseño más alla de la unidad de procesamiento, a todas las funciones de la computadora que tienen su interfaz con el usuario (p.e. video, teclado). En este caso la unidad de procesamiento neuronal se convierte en un sistema de cómputo acelerador asociado. La figura 5.3 muestra el diagrama utilizado para el diseño del sistema Netmap. La computadora neuronal posee dos estados fundamentales:

- Aprendizaje: durante el cual los parámetros adaptivos de la red son modificados siguiendo un proceso de entrenamiento supervisado (capítulo

2). En este proceso se minimiza un funcional del error en la red frente a entradas y salidas conocidas.

- Validación: durante el cual la red neuronal, con parámetros fijos y optimizados, se utiliza para procesar patrones. Es el estado normal de funcionamiento de la red neuronal como tal.

Basado en el esquema de la figura 5.3 y los dos estados de funcionamiento, se distinguen en el sistema Netmap tres funciones primordiales:

- un sistema de conversión y acondicionamiento de las señales y control de procesamiento.
- un sistema de medición y programación de los parámetros de la red neuronal
- un sistema de control del algoritmo de aprendizaje

5.3 LA RED NEURONAL ANALOGICA

Los condicionamientos sobre las entradas y salidas analógicas están impuestos por el dispositivo de cómputo neuronal. La red neuronal se organiza como un doble arreglo matricial en cascada de 64x64 sinapsis cada una. Cada sinapsis tiene asociado un coeficiente que funciona como el peso de la red neuronal. Estos coeficientes se programan eléctricamente a través de un pulso de carga. Cada matriz sináptica posee, además, 1024 umbrales, lo cual totalizan 10240 pesos programables. La figura 5.5 muestra sólo una de las capas de procesamiento. La función computada por la red neuronal para una y dos capas, respectivamente, es:

$$O_i = sigm\left(\sum_{j=1}^{64}(W_{i,j}\cdot I_j)+\theta_i\right) \qquad (5.1)$$

$$O_k = sigm\left(\sum_{i=1}^{64} W_{i,k}\cdot sigm\left(\sum_{j=1}^{64}(W_{i,j}\cdot I_j)+\theta_i\right)+\theta_k\right) \qquad (5.2)$$

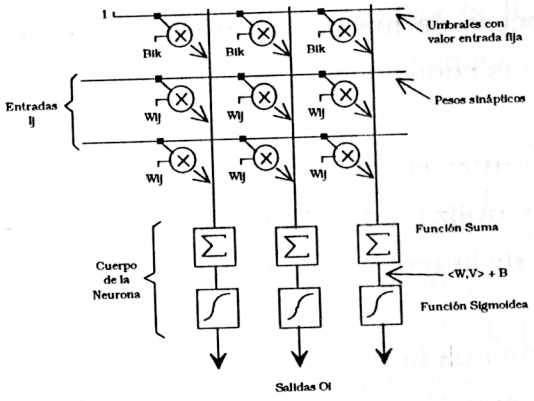

Figura 5.5 Estructura interna de la unidad de procesamiento

Para cada sinapsis, se utiliza un multiplicador analógico como el descripto en el capítulo 4 (Fig. 5.6). El mismo está compuesto por un arreglo de dos amplificadores de transconductancia cuyas entradas y salidas se encuentran en anti-paralelo, formando un multiplicador de cuatro cuadrantes. La corriente diferencial de salida **ΔIout** es proporcional al producto entre **ΔVin** y **ΔVwt**. Luego la suma de todas las corrientes diferenciales correspondientes a cada entrada **Ij** y a cada peso **Wij** se realiza en el nodo sumador.

$$\Delta I_{out} = I_b . tgh \frac{\Delta V_{in}}{2 V_t} . tgh \frac{\Delta V_{WT}}{2 V_t} \qquad (5.3)$$

La corriente **ΔIout** se obtiene como una suma algebraica de las corrientes de rama,

$$\Delta I_{out} = I^+ - I^- = I_{13} + I_{24} - I_{14} - I_{23} \qquad (5.4)$$

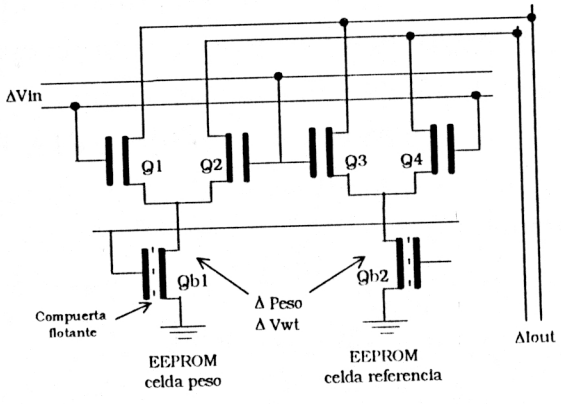

Figura 5.6 Esquema de la sinapsis

Los transistores Qb1 y Qb2 de las fuentes de corriente de los amplificadores diferenciales, son dispositivos que poseen una compuerta flotante. Dicha compuerta de silicio poli-saturado forma parte del transistor MOS y es inaccesible desde sus contactos externos. Si bien la compuerta accesible de estos transistores está fija a un potencial interno, la corriente de drenaje (Id_{Qb1} e Id_{Qb2}) de las fuentes de corriente, y por lo tanto la transconductancia de los amplificadores diferenciales, puede ser modificada por medio de la compuerta flotante. El apéndice A provee una explicación detallada del proceso tunel Fowler-Nordheim y del funcionamiento de la compuerta flotante en transistores MOS. El efecto de una inyección de carga a la compuerta flotante es macroscópicamente observado como un corrimiento de la característica de transferencia del MOS. La curva se mueve de manera que se modifica el valor de tensión umbral **VTO**. Debido a que el circuito de la sinapsis es doble en las fuentes de corriente, se consiguen corrimientos diferenciales en **VTO** y por ende, la multiplicación en los cuatro cuadrantes.

Las 128 corrientes diferenciales a lo largo de cada columna de ambas matrices de procesamiento son sumadas para conformar la salida de la neurona. Un amplificador de transimpedancia convierte esta suma de corrientes diferenciales en un valor de tensión. La función de transferencia de este amplificador posee la no linealidad de tipo sigmoidea analizada en la sección C.4 del apéndice C. El amplificador posee un control de ganancia que permite modificar la pendiente de esta característica. Este control sobre la realimentación del amplificador funciona a modo de control del parámetro de *temperatura* de la función sigmoide (Rumelhart, 1986a).

5.4 ANCHO DE BANDA DE PROCESAMIENTO EN EL SISTEMA NETMAP

En muchos puntos del diseño de la neurocomputadora las condiciones de flexibilidad y de velocidad de procesamiento están en relación de compromiso. El ancho de banda de procesamiento de patrones en la red neuronal es de 300.000 patr/seg. Esto impone el requerimiento de un ancho de banda similar en los canales de digitalización de entrada y salida que se encuentran en serie

con la red neuronal (Fig. 5.7). El sistema Netmap disminuye este ancho de banda en aproximadamente 40% de este valor. De todas maneras, la potencia de cómputo es equivalente a 120 procesadores RISC capaces de realizar una multiplicación y suma cada 33 nseg. Esta última cuenta es optimista a favor del procesador RISC dado que se lo ha ubicado en condiciones ideales de procesamiento de cómputo sin tener en cuenta los lazos de control y la manipulación de datos desde y hacia la memoria que realiza este procesador.

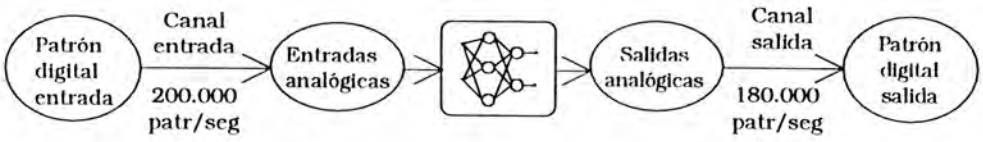

Figura 5.7 Canal de procesamiento de patrones

Debe tenerse en cuenta, también, que la efectividad de la capacidad de cómputo de procesadores digitales interconectados en arreglos paralelos dista mucho de ser ideal, es decir puede calcularse la potencia de cómputo de n procesadores interconectados como $P(n) = n.\eta(n).P_u$, donde la potencia total de cómputo, P(n), es proporcional a la potencia unitaria, Pu, y a η(n), que es una función que depende del número de procesadores. Esta función es siempre menor que uno y decrece rápidamente cuando aumenta n (fig 5.8).

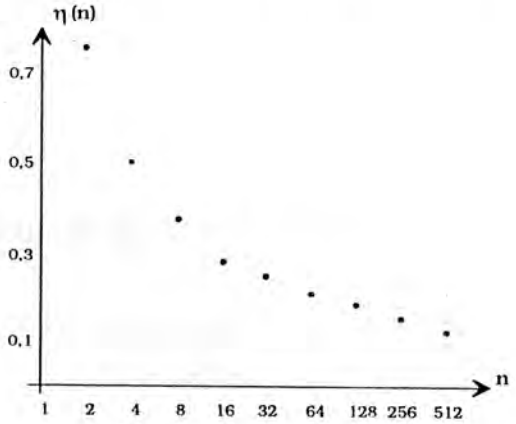

Figura 5.8 rendimiento unitario en función de la cantidad de procesadores paralelos

5.5 APRENDIZAJE EN EL SISTEMA NETMAP

El algoritmo de aprendizaje en el sistema Netmap es externo a la red neuronal. Tanto el cálculo de los coeficientes de peso como el control de programación de los mismos debe realizarse externamente a la red de cómputo analógica. Aquí se presentan varios problemas en lo referente al entrenamiento de la red. El proceso de entrenamiento es supervisado, siguiendo algoritmos tales como retropropagación (capítulo 2), Madalaine III (Widrow, 1990) o *simulated annealing* (Haykin, 1994). El algoritmo de aprendizaje calcula los pesos de la red neuronal en función de los valores actuales más la estadística del error de las muestras de entrada y salida consideradas. Una vez obtenidos estos coeficientes deben ser volcados a la red neuronal real de la neurocomputadora.

Para el proceso de cálculo de los nuevos pesos se utiliza el modelo matemático de la función de cómputo de las sinapsis y neuronas de las ecuaciones 5.1 y 5.2. La implementación electrónica posee discrepancias con estos modelos matemáticos. Estas diferencias surgen de las diferencias de valores entre los transistores que conforman los circuitos multiplicadores y amplificadores. Las diferencias existen no sólo respecto del modelo matemático sino entre sinapsis, y entre neuronas de la red. Por este motivo se propone un nuevo algoritmo de entrenamiento que incluye la red neuronal real dentro del lazo de aprendizaje.

En el aprendizaje, claramente, se destacan dos tareas:

- el cálculo de los coeficientes de peso por medio de un algoritmo de aprendizaje.

- la modificación de esos coeficientes dentro de la red neuronal analógica.

La neurocomputadora Netmap realiza ambas tareas en paralelo

5.6 OPTIMIZACION DE LA ARQUITECTURA DEL SISTEMA NETMAP

La neurocomputadora Netmap posee dos modos básicos de funcionamiento:

- procesamiento paralelo de patrones.
- aprendizaje o modificación de los parámetros internos.

Optimizar la arquitectura del sistema significa distribuir las unidades de procesamiento de manera tal de lograr un ancho de banda uniforme en la transferencia de patrones. Los algoritmos de aprendizaje son iterativos y, en consecuencia, bastante lentos. Por ende el cálculo de los pesos y la programación de los mismos debe realizarse en forma paralela para evitar mayores demoras. En ambos modos de funcionamiento, existen tareas de distinto nivel jerárquico. Las funciones de menor jerarquía o funciones de bajo nivel, son aquellas que involucran acción sobre el hardware de la neurocomputadora. Las funciones de mayor jerarquía o de alto nivel son aquellas que ven al hardware en forma lógica por medio de un programa tipo *driver*. La figura 5.9 representa el esquema jerárquico a nivel de hardware y de software del sistema Netmap.

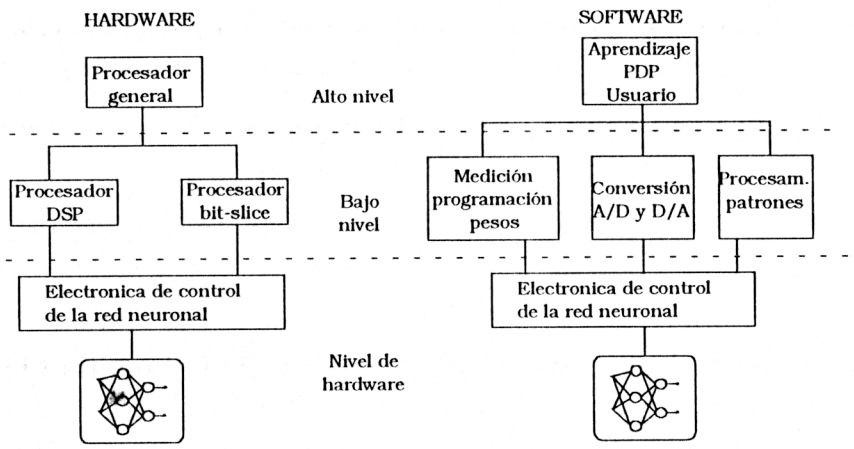

Figura 5.9 Estructura jeráquica en el sistema Netmap

Vale la pena destacar que denominación de alto y bajo nivel refiere a una visión conceptual del sistema. No hay relación con la importancia o rapidez con

que una función debe desarrollarse. Aunque, en general, las funciones de bajo nivel son conceptualmente más simples y de rápida ejecución.

Al respecto, se ha dispuesto dos procesadores para realizar las funciones de bajo nivel. El primero de estos procesadores es un procesador digital de señales (DSP) y se encarga de las funciones que involucran una intensiva computación aritmética y manejo de patrones de entrada y salida hacia la red neuronal. Todos los modos de funcionamiento del Etann y la conversión analógico-digital de patrones es realizada por el DSP. El segundo procesador utilizado es un microsecuenciador y se encarga del segundo canal de comunicaciones y conversión de datos. Es decir, del mantenimiento del canal de salida de patrones.

En un nivel jerárquico superior, se ubica al procesador de la computadora personal. Este procesador se encarga de las funciones de alto nivel del procesamiento paralelo, del aprendizaje, y de la interfaz entre la computadora neuronal y el usuario. Estas funciones comprenden el algoritmo de entrenamiento con la red en el lazo de aprendizaje, las funciones de especificación de una red neuronal, el manejo de los archivos de especificación, etc.

Aunque, los cuatro procesadores del sistema **Netmap** poseen un acoplamiento débil, son de naturaleza muy diferente y están dedicados a tareas de distintos niveles jerárquicos, han sido ubicados de manera de optimizar la tarea de aprendizaje y procesamiento de la red neuronal.

Fisicamente, el sistema **Netmap** ha sido dividido en dos placas. La primer placa, llamada **DSPcard**, concentra la parte de procesamiento puramente digital de bajo nivel de las funciones del Etann. La segunda placa, denominada **PBox**, concentra toda la electrónica analógica, de conversión y de control asociada a la red neuronal.

5.6.1 RELACION ENTRE LAS FUNCIONES DE UNA NEUROCOMPUTADORA Y EL SISTEMA NETMAP

El procesamiento de patrones y el entrenamiento de la red neuronal en el sistema Netmap involucran las funciones de:

- Transferencia y conversión de patrones entre la computadora digital y la neurocomputadora.

- Lectura de los valores analógicos correspondientes a los pesos sinápticos, umbrales y actividad neuronal.

- Modificación de los coeficientes de peso

Cada una de estas funciones involucra procesamiento en todos los niveles jerárquicos. Por lo cual los cuatro procesadores del sistema Netmap intervienen en forma concurrente en cada función.

El procesamiento de patrones tiene lugar tanto durante el entrenamiento de una red neuronal como en la faz de validación de la misma. Dicho procesamiento requiere de la transferencia de patrones hacia el procesador neuronal, e implica la conversión digital a analógica de señales y la transferencia desde la red neuronal a la computadora previa digitalización de las señales de salida. El procesador de la computadora personal controla el algoritmo de entrenamiento o validación de la red neuronal, el DSP maneja el flujo de patrones hacia la red neuronal y la conversión analógica de los mismos; el microprocesador bit-slice controla la conversión a digital y las memorias de almacenamiento temporario.

Para la lectura de las sinapsis y los umbrales se ha diseñado un método indirecto de medición debido a que la magnitud a medir no se encuentra disponible. El procedimiento empleado se describe en la sección 5.7, a continuación. La lectura de los nodos suma, previo a la aplicación de la función sigmoide, se realiza midiendo corriente diferencial. La lectura de una neurona puede hacerse tanto leyendo diréctamente la salida como a través de

una salida particular, **NO** (*neuron output*), conectada internamente por medio de llaves analógicas (apéndice C).

Modificar un valor sináptico o umbral es un procedimiento complejo, debido a que debe modificarse la carga almacenada en una compuerta flotante de un transistor MOS por efecto túnel. Esta función se realiza generando pulsos de carga de un valor de tensión y tiempo dados hacia la celda en cuestión. Los valores de tensión (**VPP**) y tiempo (**TP**) son función del error producido por la diferencia entre el valor medido y deseado del coeficiente de peso. La ecuación que los relaciona es trascentente en las incógnitas, con lo cual no pueden escribirse en forma explicita. Esto obliga a utilizar algoritmos iterativos para la obtención de **VPP** y **TP**. Dada la lentitud del proceso de modificación de pesos frente a la alta velocidad de procesamiento, se han hecho consideraciones especiales en el diseño del sistema **Netmap** las cuales serán desarrolladas en la sección 5.9.

5.7 LECTURA DE LOS PESOS SINAPTICOS

El almacenamiento de los coeficientes sinápticos W_{ij} y θ_i es estático, y funcionan como una memoria analógica de almacenamiento permanente. Estos pesos dependen de la carga acumulada en una compuerta flotante de un transistor MOS y pueden ser reprogramados eléctricamente. Tan importante como la programación es la lectura de estos coeficientes. Fundamentalmente porque la ecuación que vincula el pulso de carga de programación es función del valor almacenado.

El problema surge debido a que la medida de estos pesos no puede hacerse en forma directa ya que la variable no está accesible para lectura. No puede accederse a la medida de la carga almacenada en la compuerta flotante sin perturbarla. En consecuencia se optó por una medición indirecta basada en la relación entre la carga almacenada y el corrimiento de la tensión umbral de la característica de entrada del MOS de compuerta flotante. El apéndice C da un

análisis más detallado de esta función. De esta manera se hace una relación lineal entre el coeficiente de peso sináptico y la tensión umbral del MOS.

Debido a que cada peso es un valor diferencial formado por las tensiones de compuerta de las fuentes de corriente del multiplicador analógico (fig.5.10). Toda lectura o modificación en dicho peso, debe hacerse en función de ambos transistores. La tensión de compuerta sigue la ecuación

$$V_{TOwt} = W_{ij}/2 + 0.5 \qquad (5.5.a)$$

$$V_{TOref} = W_{ij}/2 - 0.5 \qquad (5.5.b)$$

Donde Wij es el peso sináptico y VTOwt y VTOref coresponden a los valores umbrales de las características de los dos transistores que forman la celda y los 0.5v se deben al valor intrínceco de tensión de la juntura.

Durante la medición el terminal de fuente del MOS de la fuente de corriente es polarizado a 2 volts, y el drenador de uno de los transistores del par diferencial se polariza con 3 volts. El drenador del otro transistor del diferencial es conectado a una rampa ascendente entre 0 y 4 volts, mientras se mide la corriente de drenador del primer transistor. Los circuitos de decodificación de direcciones permiten individualizar cada una de estas celdas.

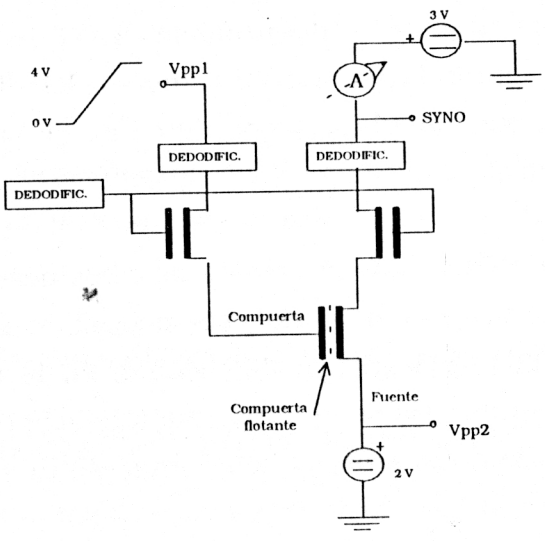

Figura 5.10 Sistema para medida de VTOwt/ref

De la figura se observa que la compuerta de la fuente de corriente sigue la tensión de la rampa en **Vrampa** hasta que sea superada la tensión umbral **VTO**. Hasta ese momento la corriente de drenador del MOS del diferencial es nula. El amperímetro cumple con la misión de detectar el momento en que la corriente crece exponencialmente (detecta valores de corrientes > 1μA) debido a que la rampa supera el valor umbral. Entonces, el valor de corrimiento del umbral, respecto al valor intrínseco se calcula como,

$$VTO_{WT} = V_{rampa} - V_{PP2} - VTO_t$$
$$VTO_t = .5v \qquad V_{PP2} = 2v$$

El circuito realizado para cumplir esta función consta de un generador en forma de rampa, y un detector de umbral. El generador es un conversor digital-analógico de alta velocidad controlable por programa desde el DSP. El detector de umbral es un comparador analógico rápido cuya salida queda registrada para su lectura desde el DSP. El circuito correspondiente se encuentra en la sección D.5.5 del apéndice D.

5.8 MODIFICACION DE PESOS EN LA RED NEURONAL

La modificación de un coeficiente de peso **Wij** (o **θi**) se hace reprogramando la tensión de umbral de ambos transistores de las fuentes de corriente del multiplicador analógico de la sinapsis. Para ello debe modificarse la carga que se encuentra almacenada en la compuerta flotante de dichos dispositivos. Esta carga pasa por efecto túnel desde el drenador hacia la compuerta flotante (inaccesible desde el exterior) atravesando una fina capa de OSi de unos 100 amstrongs. (Apéndice A).

El campo eléctrico a través del óxido de silicio es uniforme, por lo que la densidad de corriente y el campo eléctrico debido al efecto túnel pueden calcularse como (Kolodny, 1986)

$$J_{tun} = \alpha E_{tun}^2 . \exp(-\beta / E_{tun}) \qquad (5.6)$$

$$E_{tun} = |V_{tun}| / X_{tun}$$

donde Etun, Vtun y Xtun son el campo eléctrico, el potencial y el espesor entre el drenador y la compuerta flotante, y α y β son constantes. La variación de tensión en la compuerta flotante puede modelarse a través de un circuito simplificado como el de la figura 5.11

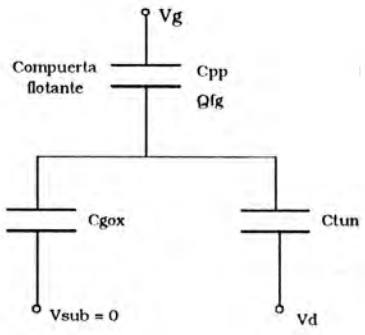

Figura 5.11 Modelo de la compuerta flotante en programación

como es obvio, la dinámica de variación de tensión de umbral responde a la ley

$$\Delta V_t = -\frac{Q_{fg}}{C_{pp}} \qquad (5.7)$$

la variación de la tensión de compuerta flotante en función de los valores medidos y los pulsos de programación es (Apéndice A)

$$\Delta V_t = V_{TOi} + V_{PP} - V_{TOwt} - 1{,}375.10^{-6} B / \ln\left[e^{(B/E_0)} + 32420.B.T_P\right] \qquad (5.8)$$

$$E_0 = 7{,}273.10^5 . (V_{PP} + V_{TOwt} - V_{TOi})$$

donde VTOi = 0.5volts es el valor intrínseco del umbral; **VPP** y **TP** son respectivamente, los valores de tensión y de tiempo del pulso de programación; **VTOwt** es el valor de umbral medido; y **B** es un coeficiente que depende de la física del dispositivo, y cuyo valor se encuentra entre 200 y 350.

Como puede verse, la ecuación 5.8 es trascendente en **VPP** y en **TP** por lo que debe utilizarse algún método iterativo para la obtención de los valores de programación.

El circuito de programación de los pesos de la red neuronal actúa enviando carga por efecto túnel a la compuerta flotante de dos transistores MOS de las fuente de corriente del multiplicador analógico de la sinapsis. Esta carga se especifica por pulsos de altura **VPP** y ancho **TP**. Para que exista efecto túnel (Apéndice A) los pulsos deben tener al menos 14 volts y no deben superar la tensión de ruptura de la juntura del transistor, unos 24 volts. El Etann requiere dos señales de alta tensión en forma pulsante **VPP1** y **VPP2**. El circuito de programaciónde la neurocomputadora genera estas señales utilizando dos conversores digital-analógico y amplificadores operacionales. La figura 5.12 muestra el diagrama de tiempos de un ciclo de programación para incrementos y decrementos tanto de la celda de peso como de la celda de referencia del multiplicador sináptico. La señal **VPP1** se mantiene constante en 18 volts, la señal **VPP2** varía según el valor calculado en la ecuación (5.8). El circuito de programación se encuentra en la sección D.5.2 del apéndice D.

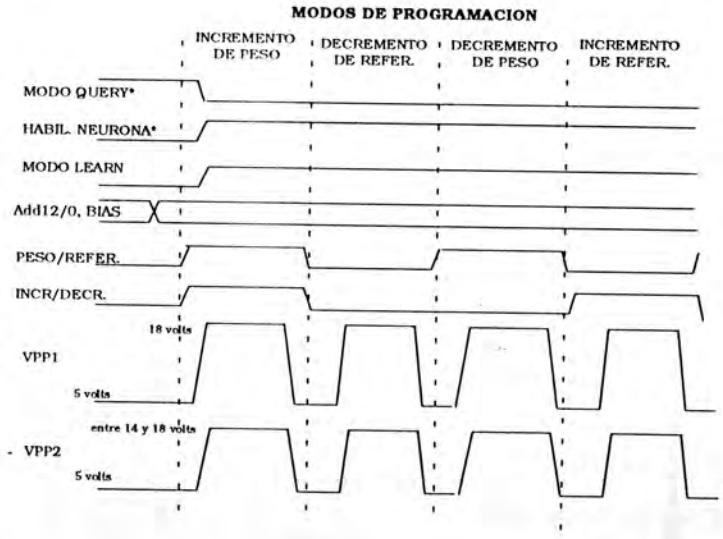

Figura 5.12 Tiempos de programación de un coeficiente sináptico

La ecuación 5.8 rige el comportamiento estático del efecto tunel. El rango dinámico de tensiones de programación es entre 14 y 18 volts, y el

correspondiente al ancho del pulso de programación entre 1µseg y 1mseg. El algoritmo completo de programación es complejo e involucra circuitos como los descriptos en esta sección. La solución adoptada para la modificación de los coeficientes de peso se detalla en el capítulo 6.

5.9 ORGANIZACION DEL HARDWARE DE LA NEUROCOMPUTADORA

Fisicamente, el sistema **Netmap** ha sido dividido en dos placas. La primer placa, llamada **DSPcard**, concentra la parte de procesamiento puramente digital de bajo nivel de las funciones del Etann. Para ello cuenta con un procesador digital de señales. Esta placa ha sido desarrollada para trabajar dentro de una computadora personal como una neurocomputadora aceleradora. La segunda placa, denominada **PBox**, concentra toda la electrónica analógica, de conversión y de control asociada al Etann. Es una tarjeta quasi-esclava y se comunica con la **DSPcard** por medio de una interfaz paralelo de alta velocidad (6 Mbyte/sec). La figura 5.13 representa el diagrama en bloques de la placa **DSPcard**.

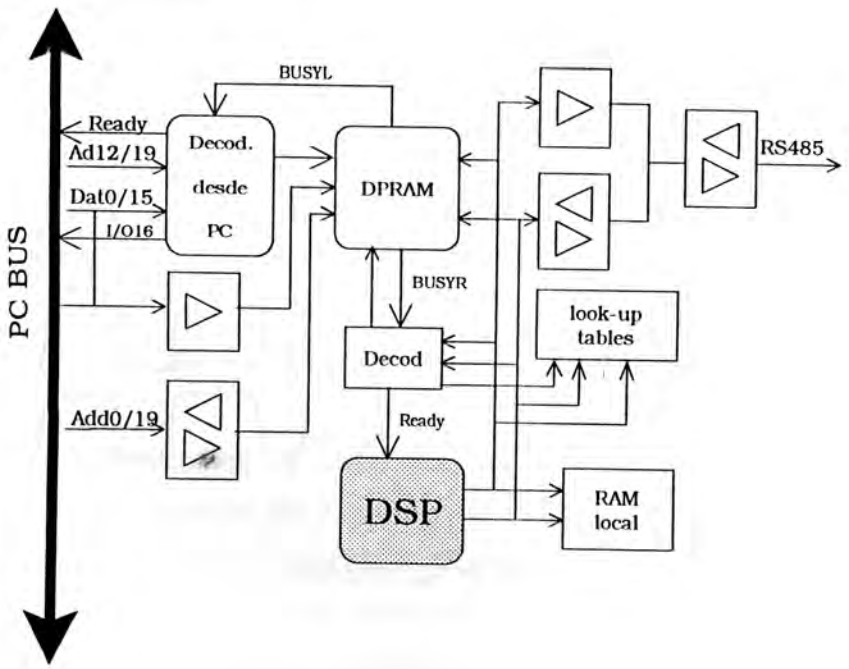

Figura 5.13 Diagrama en bloques de la placa DSPcard

El procesador digital de señales es un TMS320C25 (Texas Instr., 1990) de Texas Instrument, trabajando a 10 MIPS. Este procesador se comunica con el *bus* de la computadora personal a través de una memoria de doble puerta de 8Kbytes. El DSP actúa como maestro de la placa. Internamente, posee una arquitectura Harvard de 16 bits, 8 Kbytes de memoria Eprom, y 1 Kbyte de memoria RAM, ambas organizadas en palabras (16 bits). La unidad aritmético-lógica permite realizar una multiplicación y una acumulación en un ciclo de reloj (100 nseg.), además de una operación de registro en paralelo. Para evitar problemas de inicialización del sistema, los programas del DSP fueron optimizados en tamaño para que pudieran ser incluidos completamente dentro de los 8K de Eprom.

La placa **DSPcard** posee una interfaz de 16 bits con la computadora personal (Eggbrecht, 1991). La decodificación de la placa, vista desde la computadora personal, es un mapa de 8Kbytes (*0 wait state*) reubicables en cualquier posición de la zona alta de la memoria (HMA). La lógica de control de decodificación local de las direcciones de la computadora personal maneja los accesos simultáneos a la misma posición en la memoria compartida. En caso que haya un acceso de escritura simultaneo a la misma posición de la memoria compartida, la memoria genera una señal de retardo que es interceptada por la lógica de control de la placa, que la sincroniza con los tiempos de ciclo de la computadora personal (fig. 5.13). En este ceso pueden generarse ciclos de espera para ambos procesadores hasta que el procesador que posee la memoria, la libere. Los accesos desde la computadora personal pueden ser efectuados en 8 y en 16 bits, la lógica de la **DSPcard** maneja el control de tamaño de acceso a los datos (fig. 5.13).

En la placa **DSPcard**, el DSP tiene un *bus* local con recursos privados. La memoria RAM local del sistema (32 KBytes) se utiliza como memoria de almacenamiento temporaria de los patrones de entrada/salida y como memoria de almacenamiento local de las variables del sistema. La memoria Eprom externa de 64 Kbytes es usada para almacenar las tablas de las curvas de **Vpp** y **Tp** utilizadas en el cálculo de los pulsos de programación de los pesos. Estas tablas son generadas *off-line* en los programas de simulación que se

describirán en el capitulo 6, y aceleran el cómputo de los pulsos de programación.

El DSP tiene su mapa de memorias dividido en tres secciones: **PS**, memoria de programa, **DS**, memoria de datos, e **IS**, memoria de entrada/salida. Cada sección direcciona 64Kwords. En el apéndice D se describe la disposición de recursos locales y compartidos del DSP en los mapas de memoria respectivos.

Vale destacar que las exigencias en el ancho de banda de transferencia de datos (Fig. 5.3) imponen accesos sin ciclos de espera a los recursos de memoria local y decodificaciones rápidas del orden de 5 a 10 nseg.

La conexión de la placa **DSPcard** con la **PBox** se efectuó por medio de una interfaz que usa la codificación eléctrica RS485 con direcciones, datos y lineas de control en paralelo. Se eligió la señalización RS485 por sus buenas características de velocidad de transmisión e inmunidad al ruido (National, 1992). Las direcciones y datos del DSP son multiplexados antes de ser convertidas al protocolo RS485 para disminuir el número de lineas a casi la mitad. Sin embargo el rendimiento de esta interfaz no disminuye con el multiplexado debido a que existe un tiempo muerto entre la decodificación de direcciones y la utilización de los datos. En el apéndice D se muesrtran los circuitos correspondientes y los diagramas de temporización de la comunicación entre ambas placas.

La placa **PBox** realiza las funciones de conversión analógico-digital y control de las señales que llegan a la red neuronal. Posee, además, una memoria de almacenamiento temporario para los patrones procesados y un microcontrolador que se encarga de la conversión analógico-digital. La figura 5.14 representa un esquema de bloques de las funciones de la placa **PBox**, que incluye:

- Interfaz RS485
- Convertidores digital-analógico para pulsos de alta tensión
- Circuito de conversión de los patrones de entrada
- Control de ganancia de la función sigmoide

- Circuito de lectura de coeficientes de peso
- Conversión analógico-digital y memoria de almacenamiento temporario
- Circuito de lectura de nodo suma o salida de neurona

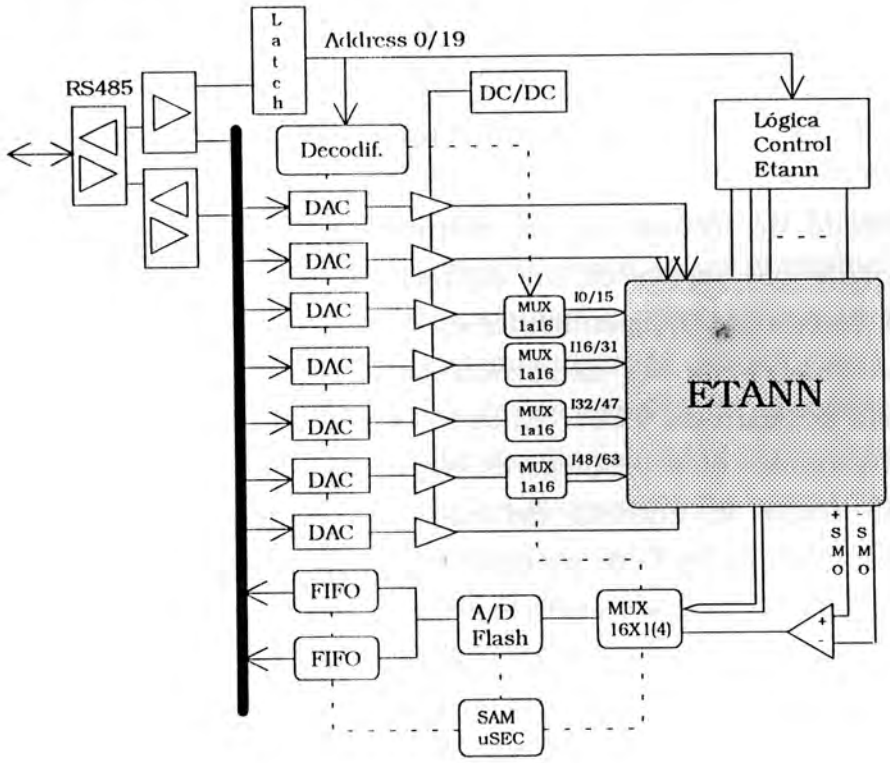

Figura 5.14 diagrama en bloques PBox

El circuito correspondiente a cada uno de estos bloques está detallado en el apéndice D. Esta placa concentra la electrónica de bajo nivel de los circuitos de lectura y modificación de los coeficientes sinápticos tratados anteriormente; asi como para el ajuste de la ganancia de la función sigmoidea de las neuronas.

5.9.1 CONVERSION DE PATRONES DE ENTRADA Y DE SALIDA DE LA RED NEURONAL.

Los patrones generados en la computadora pasan al sistema Netmap a través de la memoria compartida (Fig. 5.3). Estos patrones digitales son convertidos a analógicos para ser procesados por la red neuronal. Para ello se dispone un

conjunto de conversores digital-analógico y de multiplexores analógicos de alta velocidad. Los conversores están dispuestos en pares. De esta forma se optimiza la escritura en los DACs, un par de DACs se escribe durante el tiempo de establecimiento del otro par. Los multiplexores, manejados por la lógica de decodificación, conducen la señal de los conversores a cada una de las 64 entradas analógicas del Etann. Cada salida de multiplexores junto con un capacitor de .22nf actúa como circuito de muestra y retención (*sample&hold*).

El circuito de lectura de los patrones procesados por la red neuronal es controlado por medio de un microcontrolador. Este controlador trabaja en forma autónoma, independientemente del DSP, quien sólo controla los modos de funcionamiento del microcontrolador durante el modo de procesamiento paralelo de patrones (PDP). El circuito que lee las 64 salidas del Etann consta de un conjunto de multiplexores organizados en dos capas, que conducen las salidas hacia la entrada del conversor analógico-digital. La función del microcontrolador es la de manejar las lineas de selección de los multiplexores, las señales de temporización del conversor A/D y el almacenamiento de la salida digital del conversor en un par de memorias tipo FIFO (First in first out) de 512 palabras cada una. El conversor analógico-digital, de 8 bits, es de tipo *flash* y trabaja a 10 millones de conversiones por segundo. El microcontrolador puede ser programado para leer secuencialmente las 64 lineas de salida del Etann o selectivamente un grupo de ellas. La figura 5.15 muestra un diagrama de tiempos del ciclo del microcontrolador.

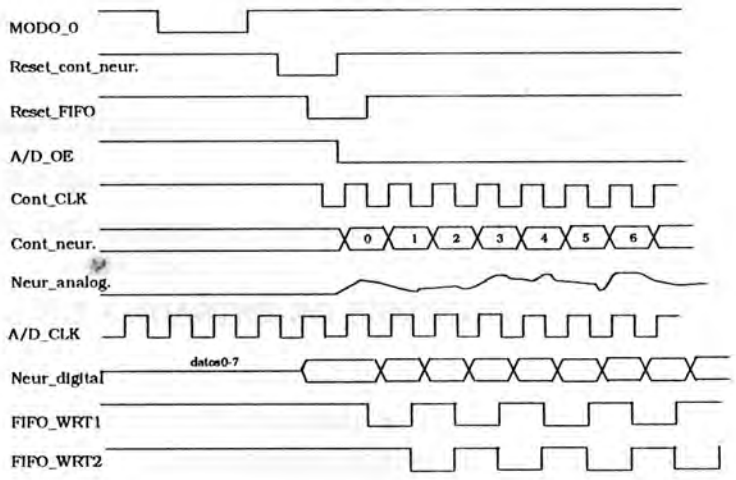

Figura 5.15 Diagrama de tiempos del microcontrolador, A/D y FIFO

5.10 DISCUSION

Basado en los modelos neuronales estáticos y las técnicas de implementación electrónica, se ha generado una computadora neuronal analógica, de propósitos generales, capaz de procesar un flujo de 180 mil patrones por segundo. El sistema ha sido concebido con la suficiente flexibilidad para poder ser utilizado en sistemas de procesamiento en tiempo real con patrones analógicos o digitales. La neurocomputadora se ha integrado a un sistema de computo personal que permite una completa interacción con el usuario. La optimización del diseño de la computadora neuronal es un problema complejo debido a la naturaleza diferente de los patrones y procesadores analógicos y digitales. Esto redunda en características muy diversas que obligan a extremar medidas en cuanto al ancho de banda de los canales de comunicación. La electrónica involucrada es en su totalidad de alta velocidad, compatible con las exigencias de la red neuronal analógica. Los mayores esfuerzos en este sentido han estado orientados a crear una arquitectura tal que pudiera mantener el orden de magnitud de la velocidad de procesamiento de la red neuronal analógica para procesar patrones digitales, con la mínima cantidad de elementos externos posibles.

En este capítulo se han descripto las consideraciones básicas y el diseño de la electrónica de la neurocomputadora. Se ha puesto énfasis en los procedimientos de programación de los coeficientes de peso de la red neuronal, esenciales para cualquier algoritmo de aprendizaje. Se ha definido también el nivel jerárquico que ocupan los distintos procesadores utilizados que le dan al diseño de la neurocomputadora una característica extra de computadora paralela. El trabajo de diseño específico de cada circuito de funciones propias del hardware del sistema está detallado en el apéndice C.

La resolución de las funciones de la computadora neuronal en hardware es solo parcial. Otros problemas intrínsecos de la arquitectura y funciones neuronales deben resolverse con técnicas de software. Afortunadamente, puede mantenerse la jerarquía de tareas creada para el hardware. Estos problemas son resueltos en el capítulo 6.

CAPITULO 6

FUNCIONES DE PROGRAMACIÓN DEL SISTEMA NEURONAL NETMAP

6.1 INTRODUCCION

Ha quedado claramente expresado, que el sistema Netmap es una neurocomputadora de propósitos generales para el procesamiento de patrones analógicos o digitales en tiempo real. En el capítulo anterior se definió la arquitectura óptima para este sistema basado en la acción paralela de cuatro procesadores. Se puso especial énfasis en la flexibilidad del sistema para definir distintas arquitecturas de modelos neuronales estáticos multicapa y en el ancho de banda para procesar tanto patrones analógicos como digitales. En base a estas consideraciones, se ha creado una estructura jerarquizada de funciones que deben ser ejecutadas tanto por el hardware dispuesto como por el software de base de la neurocomputadora. En la sección 5.6 se definieron dos modos básicos de funcionamiento del sistema Netmap:

- procesamiento paralelo de patrones.
- aprendizaje o modificación de los parámetros internos.

Además, en la figura 5.9 se estableció el paralelismo entre los elementos de procesamiento y las funciones de software asociadas. Dado que la estructura multiprocesador del sistema Netmap es jerárquica, las funciones que cada procesador desarrolla es de naturaleza diferente. El acople entre los procesadores es débil. Cada uno posee autonomía para la concreción de funciones unitarias específicas y se comunica con los otros a través de funciones primitivas. En consecuencia, las funciones de programación se han realizado de manera tal que la acción de un procesador sea lo más transparente posible para el resto de los procesadores. Es decir, la

concurrencia de tareas acelera la acción de cada procesador si se optimiza individualmente la respuesta de los procesadores conectados al mismo.
Dentro de la estructura jerárquica de funciones, las de mayor nivel son aquellas conectadas con el usuario del sistema Netmap. Debe existir una interfaz que permita especificar una dada arquitectura de red neuronal. Dicha especificación se realiza a través de un archivo que considera el número de capas, neuronas y conexiones de la red neuronal, así como todos los valores deseados.

La comunicación con el usuario debe hacerse de modo que el mismo perciba la evolución que tiene la red tanto durante la faz de entrenamiento como durante el procesamiento de los patrones. Para este seguimiento se ha creado una interfaz gráfica y un simulador neuronal.

Durante las funciones de entrenaniento y procesamiento de la red neuronal, el procesador de señales ubicado en la placa DSPcard ejecuta las funciones de medición y modificación de los parámetros de la red, conversión digital-analógica y control del hardware del sistema. El microsecuenciador, a su vez, está dedicado a controlar el canal de salida de la red neuronal y su conversión a valores digitales.

En este capítulo se describe la implementación de las funciones antedichas en los programas realizados para cada uno de los procesadores y la solución de la ecuación de programación de los pesos de la red neuronal a través de un conjunto de tablas.

6.2 ENTRENAMIENTO DE LA RED NEURONAL EN EL SISTEMA NETMAP

El entrenamiento de la red neuronal llevado al sistema Netmap es un proceso complejo. Durante el entrenamiento de la red, un algoritmo de aprendizaje supervisado, tal como el de retropropagación, genera los coeficientes de peso que deben grabarse dentro de la red neuronal. Como se vió en el capítulo 2, este algoritmo es iterativo, y se basa en minimizar un funcional del error

generado por los patrones de prueba que son presentados a la red neuronal. La aplicación iterada de estos vectores de prueba modifica paulatinamente los pesos de la red hasta que se logra la convergencia en el mínimo de la superficie de error del sistema.

En el sistema Netmap, durante cada iteración del algoritmo de aprendizaje, llamada época, los pesos resultantes son efectivamente grabados dentro del dispositivo neuronal analógico. En el capítulo 4 se analizó el almacenamiento en estas memorias analógicas, utilizando pulsos de carga, que por efecto tunel, logran polarizar la característica de transferencia de un multiplicador sináptico. Sin embargo deben aún resolverse problemas asociados con la función de programación de los pesos. Las soluciones propuestas se realizaron por software, utilizando el hardware anteriormente descripto.

6.2.1 RED NEURONAL DENTRO DEL LAZO DE APRENDIZAJE

La figura 6.1 muestra el algoritmo de entrenamiento realizado, el cual incluye a la red neuronal analógica dentro del lazo de entrenamiento. Se observa, que cada iteración del algoritmo de aprendizaje desencadena un proceso de modificación de los coeficientes internos de la red neuronal. El bloque programación de pesos puede, entonces, ser expandido en el conjunto de funciones detalladas en la parte derecha de la figura.

El algoritmo *chip-in-the-loop* desencadena una secuencia de funciones de medición de los parámetros almacenados, y programación por pulsos de carga hasta lograr el valor deseado. Este proceso en si, es también, un proceso iterativo debido a que el modelo de la memoria analógica no es exacto y posee dispersión en sus parámetros. Sin embargo posee la ventaja de modificar los coeficientes de peso de la red neuronal, siguiendo, no un modelo teórico como el de las ecuaciónes 5.1 o 5.2, sino utilizando neuronas y sinapsis reales; eliminando así los desbalances de la dispersión de los parámetros internos.

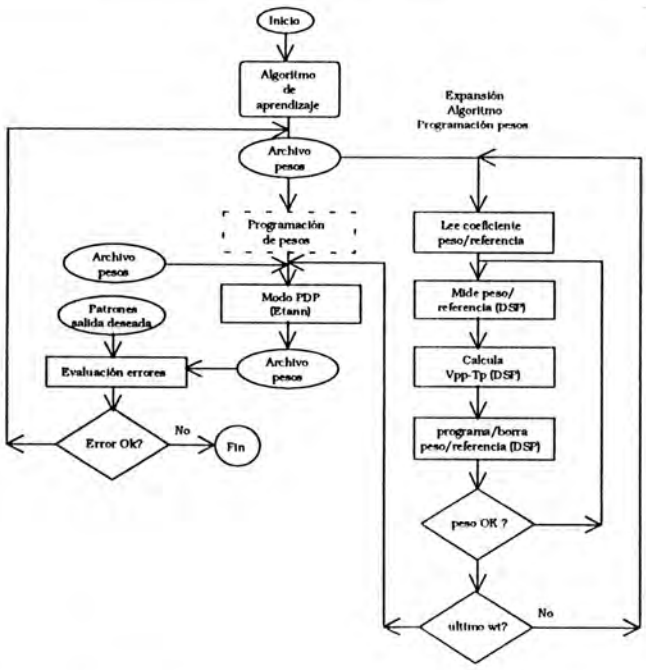

Figura 6.1 Algoritmo de programación de pesos

El precio que debe pagarse para lograr el entrenamiento con la red neuronal dentro del lazo de aprendizaje es el tiempo involucrado en este proceso. El sistema de programación, es en sí, un proceso lento. Esto se debe, fundamentalmente a tres motivos:

- el pulso de carga de programación de los pesos de la red neuronal posee un rango dinámico que va del microsegundo hasta 100 µseg.

- la medición de la carga almacenada en la compuerta flotante lleva unos 5µ seg.

- el cálculo de los valores de altura y ancho del pulso de carga (**VPP** y **TP**) se obtiene resolviendo una ecuación tracendente en las variables incógnitas.

La primera condición es intrinceca del proceso de programación de los coeficientes de peso, en consecuencia inmejorable. La única optimización al respecto, en la neurocomputadora, se ha hecho a nivel del sistema, distribuyendo las cargas en los procesadores de manera que se encuentren

activos realizando tareas concurrentes durante la faz de programación de la celda neuronal.

La segunda de estas afirmaciones ya ha sido optimizada utilizando el método de aproximaciones sucesivas en la medición indirecta de la carga almacenada en la compuerta flotante. En el capítulo 5 se vió que utilizando una rampa el tiempo de medición era dependiente del valor de la carga a medir. Con un dominio entre 0,9 y 77 µseg. La mejora introducida por el método de aproximaciones sucesivas reduce este tiempo a 1/8 de la media para una distribución uniforme de los valores medidos.

La última de estas condiciones ha sido optimizada realizando una simulación del modelo de la memoria analógica y generando un conjunto discreto de valores de programación. La proxima sección analiza en detalle lo realizado al respecto.

6.3 SIMULACION DEL MODELO DE LA MEMORIA ANALOGICA

El algoritmo de aprendizaje genera en cada iteración, un conjunto de pesos que deben ser programados en la red neuronal. La ecuación 6.1 representa el modelo de variación de tensión umbral de la característica de transferencia del MOS con compuerta flotante que se utiliza en la confección del peso sináptico de la neurona. El apéndice A realiza un estudio detallado del efecto tunel al cual responde este modelo.

$$\Delta V_t = V_{TOt} + V_{PP2} - V_{TOwt} - 1{,}375.10^{-6}\, B / \ln\left[e^{(B/E_0)} + 32420.B.T_P\right] \qquad (6.1)$$

$$E_0 = 7{,}273.10^5.(V_{PP2} + V_{TOwt} - V_{TOt})$$

En esta ecuación ΔVt representa la diferencia entre el valor de tensión medido en la compuerta y el deseado (del algoritmo de aprendizaje). VTOwt es el valor de tensión medido, y, VPP y TP, la altura y el ancho del pulso de programación. Además, el coeficiente B de esta ecuación es un valor que depende de constantes físicas del proceso (Apéndice A). El valor de B oscila entre 200exp(6)

y 350exp(6). Debido a que esta ecuación es trascendente en las variables que se desean obtener, se utiliza un metodo numérico basado en un procedimiento iterativo para hallar los valores de VPP y TP. En este proceso se identifica, además, el valor del coeficiente B para el dispositivo neuronal. Se repite el cálculo que indica la ecuación 6.2 hasta que el error 6.3 sea menor que un valor prefijado

$$V_{PP2}^{n+1} = V_{PP2}^{n} - \frac{F_1(V_{PP2})}{F_2(V_{PP2})} \qquad (6.2)$$

$$F_1(V_{PP2}) = V_{TOwt} + V_{PP2} - \Delta V_{TO} - 1.375.10^{-5}.B/\ln[e^{(B/E_0)} + 32420.B.T_P]$$

$$F_2(V_{PP2}) = 1 - [B^2.e^{(B/E_0)}/\ln^2[e^{(B/E_0)} + 32420.B.T_P].E_0^2.[e^{(B/E_0)} + 32420.B.T_P]]$$

$$\varepsilon = V_{PP2}^{n+1} - V_{PP2}^{n} \rightarrow 0 \qquad (6.3)$$

En las ecuaciones F1 y F2 todos los valores utilizados corresponden a la iteración n. El coeficiente B se inicializa con un valor medio (p.ej. 300). El algoritmo de programación de un coeficiente de peso se representa en la figura 6.2 e incluye: medir el nivel de umbral del MOS, encontrar los valores de VPP2 y TP, enviar el pulso de programación y luego en función del error cometido, estimar el coeficiente B y repetir el proceso.

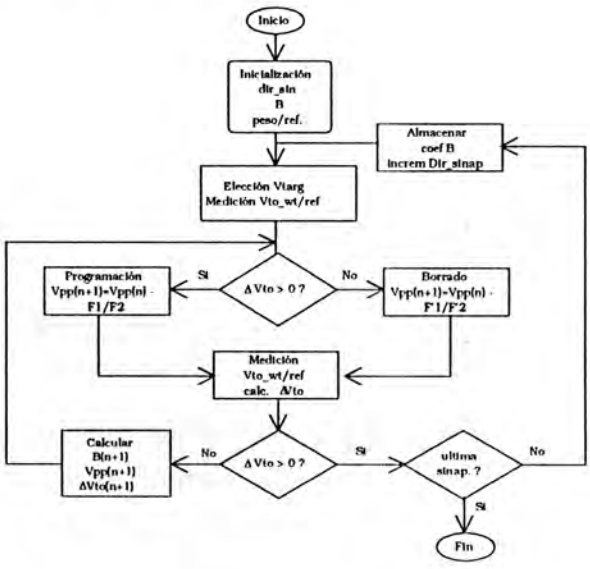

Figura 6.2 Algoritmo de programación

El coeficiente B se obtiene de igual manera, usando el proceso iterativo de la ecuación 6.4. Las funciones G1 y G2 se obtienen de la ecuación general 6.1

$$B_{n+1} = B_n - \frac{G_1(B)}{G_2(B)} \qquad (6.4)$$

$$G_1(B) = 7.27.10^5 (V_{TOwt} + V_{PP2} - \Delta V_{TO} - .5).\ln[e^{(B/E_0)} + 32420.B.T_P] - B$$

$$G_2(B) = 7.27.10^5 (V_{TOwt} + V_{PP2} - \Delta V_{TO} - .5)[e^{(B/E_0)}/32420.E_0.T_P]/[e^{(B/E_0)} + 32420.B.T_P] - 1$$

$$\varepsilon = G_{n+1} - G_n \rightarrow 0 \qquad (6.5)$$

Puede verse que en el algoritmo de la figura 6.2 existen 3 procesos iterativos: uno para hallar **VPP2** y **TP**, otro para hallar **B** y el proceso de programar la celda neuronal hasta que **VTOwg** converja al valor deseado Vtarg. La razón por la cual no podemos hacer que **VTOwg = Vtarg** con un solo pulso de programación se debe a que el modelo matemático de la ecuación 6.1 no refleja en forma exacta el proceso físico que es llevado a cabo. Además existen discrepancias entre las constantes definidad de celda a celda, y por último, el proceso es iniciado con valores supuestos como en el caso de la constante **B**.

Por otro lado, estos tres procesos iterativos convierten al algoritmo de programación en un proceso extremadamente lento para cualquier microprocesador. En consecuencia la solución adoptada consiste en realizar todos los cálculos de **VPP2, TP** y **B** en tiempo diferido e incluirlos en el sistema en forma de tablas. El proceso de generación de estas tablas se describe a continuación

6.3.1 GENERACION DE LAS TABLAS DE VPP, TP Y B.

Si volvemos a observar el algoritmo de programación (fig. 6.3) podemos ver que si los valores de **VPP2, TP** y **B** son tabulados, el algoritmo solamente posee un lazo iterativo debido a la efectividad que posea el pulso de programación para reducir el error en **VTOwg - Vtarg.**

Las simulaciones en base a modelos y las estadísticas trabajando con las celdas verdaderas demuestran que no se necesitan más de 10 iteraciones y en promedio 4 para programar un peso.

Las tablas de **VPP2**, **TP** y **B** fueron obtenidas por simulación en computadora. La figura 6.3 muestra la salida de simulación de las curvas de la ecuación 6.1. Estas curvas dan **VPP2** en función del error ΔV_{TOwg} usando **VTOwg** como parámetro.

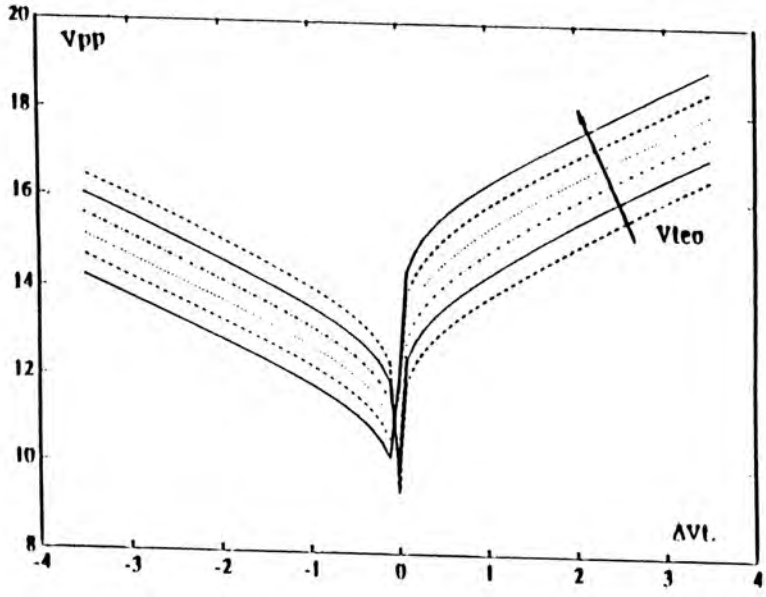

Figura 6.3 curvas de **VPP2** vs. ΔV_{TOwg}

El procedimiento que se siguió para la confección de las tablas fue el siguiente: se utilizó un rango de valores mínimo y máximo para **VPP2** de 14 a 20 volts que es el rango donde existe mayor efectividad del efecto tunel sin poner en riesgo la integridad del transistor. Asimismo se eligió que **TP** funcionara como un factor de escala tomando valores discretos de 1 a 100 µseg. Es decir, el algoritmo de cálculo de **VPP2** utiliza un valor fijo de **TP**, si el valor hallado se

encuentra fuera del rango dinámico, repite el proceso cambiando el valor de **Tp**. De esta manera se obtiene una tabla de pares de valores **Vpp2-Tp**. Si bién el proceso de cálculo es lento, se hace una única vez. La tabla de **B** se hace de manera similar.

En las simulaciones de la programación de una celda con compuerta flotante en la cual se debe identificar el coeficiente **B**, se agregó un porcentaje sumado como valor aleatorio a la medida **Vtowg**. La figura 6.4 muestra algunos casos típicos en donde el valor **Vtowg** se aproxima al valor **Vtarg**, y la reducción del error, respectivamente.

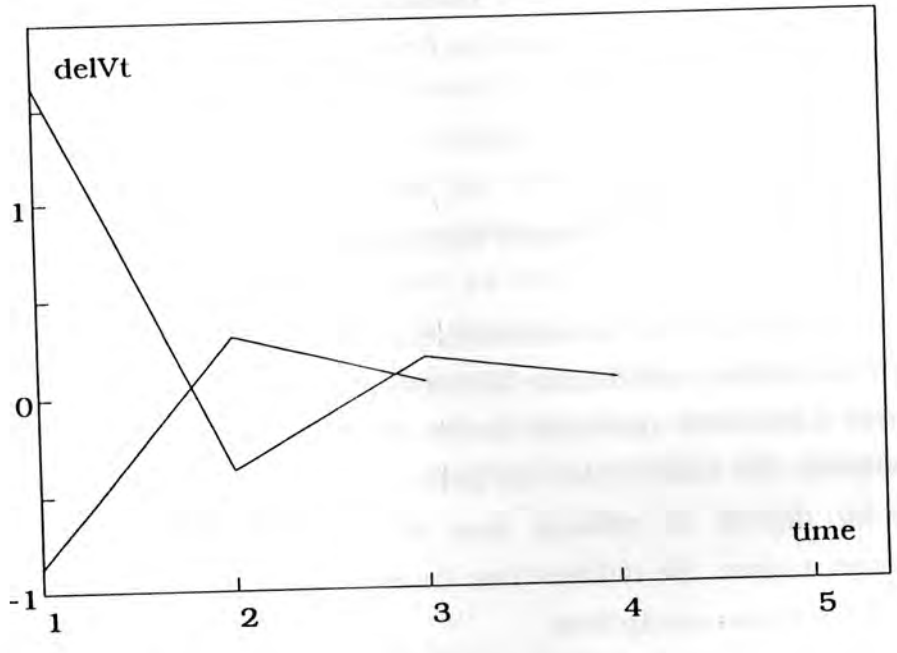

Figura 6.4 convergencia del algoritmo de programación de pesos

6.4 SIMULADOR DE REDES NEURONALES

La tabulación de valores permite acelerar enormemente el cálculo de los valores del pulso de programación manteniendo al dispositivo neuronal dentro del lazo de aprendizaje de la neurocomputadora. De todas maneras, se ha

creado un simulador neuronal que permite prescindir del dispositivo neuronal, al menos durante la primera parte del entrenamiento.

Existen dos opciones dentro del modelo utilizado por el simulador. La primera, simplemente, utiliza las ecuaciones del algoritmo de retropropagación de la sección 2.6.1. La segunda se vale de una función realizada por el procesador digital de señales con la cual se realiza un modelo numérico de cada una de las 128 neuronas de la red neuronal. En este caso se ponen de manifiesto las discrepancias propias de la electrónica de las mismas. En ambos casos el sistema de entrenamiento puede evolucionar por simulación hasta la convergencia de los coeficientes de peso, programar estos valores y luego continuar con el entrenamiento usando la red neuronal dentro del lazo de aprendizaje para lograr una sintonía fina de los mismos.

6.5 FUNCIONES DE MEDICION Y PROGRAMACION EN BAJO NIEVEL

Quedó de manifiesto en las secciones anteriores, que en el entrenamiento de la red neuronal deben modificarse las memorias sinápticas analógicas, cuando se incluye al dispositivo neuronal dentro del lazo de aprendizaje. El procesador que computa los coeficientes de peso solicita la ejecución de ésta tarea al procesador digital de señales, que es quien maneja la electrónica de la neurocomputadora. Se definen tres funciones básicas para el manejo de los coeficientes de peso sinápticos:

- medición de la tensión umbral
- pulso de programación controlado
- pulso de borrado controlado

La función VT_MEASSURE mide el valor de tensión umbral VTo en el transistor de una de las dos fuentes de corriente del multiplicador analógico que se utilizan en la implementación de la sinapsis. Como fue dicho en la sección 5.3

cada sinapsis posee un coeficiente de peso diferencial, el cual puede tomar valores positivos y negativos. El valor diferencial se obtiene usando dos celdas una de las cuales funciona como referencia. La tensión VTo depende de la carga almacenada en la compuerta flotante del MOS y se mide en forma indirecta como fue explicado en la sección 5.7.

La función VT_MEASSURE configura a la red neuronal con niveles fijos de tensión y genera la rampa de 0 a 4 volts en VPP. El programa se encarga, además, de configurar a la red neuronal en modo de lectura, inicializar un registro de direcciones que selecciona la sinapsis que se quiere leer y de controlar el registro de estados en el cual se indica el momento de detener la rampa de medida y obtener el valor de VTo final.

Existe una segunda versión de esta función, que realiza exactamente la misma tarea pero en lugar de generar una rampa, calcula VTo por aproximaciones sucesivas. La ventaja de esta rutina es, obviamente, ahorro en tiempo de procesamiento. La rutina anterior no tiene una duración fija, sino que es dependiente del valor del VTo a medir. La rampa se genera incrementando la salida de un conversor DAC. Si VTo es alta se necesitan más escrituras y conversiones del DAC. La versión mejorada, en cambio, utiliza un número fijo de 8 conversiones, una por bit de resolución del DAC siguiendo el procedimiento de aproximaciones sucesivas. En promedio esta última rutina es 8 veces más veloz que la primera.

La función de programación VPPROGRAM genera un pulso de altura VPP y duración TP según los valores calculados por el algoritmo de entrenamiento. Este pulso tiene el propósito de modificar el valor de tensión umbral VTo variando la carga de la compuerta flotante. La función configura a la red neuronal en modo programación, especifica la dirección de la sinapsis o umbral a modificar, teniendo en cuenta si la modificación se hará sobre la celda de referencia o la de peso del coeficiente diferencial. La temporización de las señales (fig. 5.12) se hace por programa. (Cancelo, 1993).

La función de borrado es muy similar a la de programación, salvo que debe extraerse carga por efecto túnel que ha sido previamente almacenada en la compuerta flotante. En este caso el pulso de alta tensión VPP debe aplicarse al terminal de drenador en lugar de la compuerta. La función de borrado, VPPERASE, controla ciertas llaves analógicas que identifican la sinapsis y drenador en cuestión. El diagrama de tiempos de esta función es similar al de la función de programación.

6.6 PROGRAMACION DEL PROCESADOR DIGITAL DE SEÑALES

El procesador digital de señales, ubicado en la placa DSPcard de la computadora neuronal, es veloz en la realización de funciones de bajo nivel de programación. Es decir controla la electrónica asociada a la medicion y programación de los pesos de la red neuronal, el canal de patrones de entrada a la red y las comunicaciones con el resto de los procesadores involucrados. Además, controla los modos de funcionamiento de los procesadores que estan jerárquicamente más abajo (Fig. 5.9), es decir, al microsecuenciador y al procesador neuronal.

Para ello se ha creado un pequeño kernel en C que realiza las funciones de interfaz de comunicaciones con el procesador de la PC e intérprete de comandos. Este intérprete se maneja a través de primitivas editadas desde el procesador de alto nivel y que desencadenan la ejecución de una o más funciones sobre la neurocomputadora. Las primitivas pueden encadenarse y son ejecutadas por prioridad de entrada en la tabla de comandos

Todas las funciones restantes, vinculadas al control del hardware, medición y programación fueron realizadas en assembler para acelerar su ejecución y reducir el tamaño del código. De esta manera, todo el código generado para el DSP puede ser almacenado dentro de la memoria interna del procesador, maximizando la velocidad de ejecución y minimizando los accesos externos.

6.6.1 CANAL DE PATRONES DE ENTRADA

Tanto durante el entrenamiento de la red como durante la faz de procesamiento, los patrones digitales son procesados en forma analógica por la red neuronal. El procesador digital de señales mantiene el canal de entrada de procesamiento (Fig. 5.7). Se han generado dos funciones principales vinculadas al manejo de dicho vínculo:

- la función de control del modo de procesamiento
- la función de procesamiento paralelo de patrones

la primera función, PDP_MODE, configura la red neuronal en condiciones iniciales para procesamiento paralelo. Esta función fija el parámetro *temperatura* de las sigmoides de las neuronas modificando la ganancia de los amplificadores de salida. Se inicializan, además, las memorias de almacenamiento de los datos de salida (FIFO's) y el modo de funcionamiento del microcontrolador.

La función de procesamiento de patrones, SRPATT, maneja la entrada y salida de patrones que son procesados por la red neuronal. Este módulo realiza las siguientes funciones

- maneja la memoria circular de lectura en el área de patrones de entrada de la memoria de almacenamiento compartido.
- genera las condiciones iniciales del controlador de datos de salida.
- transfiere los datos de la memoria compartida a los conversores digital-analogicos.
- maneja la selección de multiplexores que conducen los datos convertidos a las entradas de la red neuronal.
- maneja el ingreso de datos dentro de la red.
- lee los patrones de salida de las memorias de almacenamiento temporario.
- maneja la memoria circular de lectura en el área de patrones de salida de la memoria compartida.

Los algoritmos de manejo del canal de procesamiento de patrones han sido concebidos de manera tal de optimizar el flujo de transferencia de datos desde y hacia la red neuronal. Para ello se utilizaron técnicas de solapamiento (*pipelining*) utilizando todo el ancho de banda posible del procesador digital de señales. La figura 6.5 muestra la secuencia de tiempos correspondiente. El microsecuenciador maneja la lectura de los datos analógicos de salida y su almacenamiento temporario en memoria. Antes de que el microsecuenciador comience a almacenar el patrón N en memoria, el DSP ha comenzado la conversión del patrón N+1 en los DACs. Este procedimiento acelera la transferencia de patrones aumentando el ancho de banda en aproximadamente un 45%.

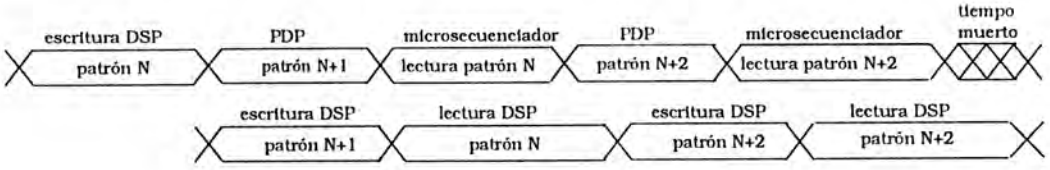

Figura 6.5 paralelización en la transferencia de patrones

6.6.2 COMUNICACION ENTRE PROCESADORES

El intérprete de comandos del DSP (Cancelo, 1994) sirve de interfaz de comunicaciones con el procesador de la computadora personal. Este programa mantiene una estructura como muestra la tabla 6.1, la cual se ubica en una zona de memoria compartida entre ambos procesadores. El DSP es manejado a través de primitivas editadas por el procesador de la computadora personal en la zona de comandos. Cada primitiva genera una o más funciones en assembler relacionadas con tareas sobre el procesador de la red neuronal. Las primitivas pueden encadenarse y son ejecutadas por prioridad de entrada a la tabla de comandos. Existe además una zona reservada a los parámetros de cada primitiva. Durante la decodificación que hace el DSP se obtiene el número de parámetros que es fijo para cada comando.

Cada vez que finaliza la ejecución de una primitiva el DSP devuelve a la tabla un estado que indica el éxito o fracaso del comando y los resultados obtenidos, en el área destinada a tal efecto.

Las áreas de patrones de entrada y salida están destinadas al almacenamiento de patrones para procesamiento en la red neuronal. El área destinada para cada una de ellas es de aproximadamente 8Kby. Es decir pueden almacenarse hasta 128 patrones de 64 bytes cada uno.

DPRAM	Tamaño en bytes
COMANDOS	16
ESTADOS	16
AREA DE PARÁMETROS	64
AREA DE RESULTADOS	64
AREA DE PATRONES DE ENTRADA	8K
AREA DE PATRONES DE SALIDA	8K

Tabla 6.1 Area de comandos

6.7 INTERFAZ ENTRE EL SISTEMA NETMAP Y EL USUARIO

El sistema Netmap es una computadora neuronal de propósitos generales en la cual un usuario puede construir redes neuronales de diversos tamaños y configuraciones. La interfaz entre el usuario y el sistema es interactiva, a través de una interfaz gráfica en la que se puede seguir la evolución de la red neuronal durante las fases de aprendizaje y procesamiento paralelo de patrones. La configuración de la red neuronal se hace a través de un archivo de formato específico en el cual se incluyen los datos de entrada. Los resultados son tambien almacenados en archivos para su posterior análisis.

Para estas tareas se han definido dos módulos: el que realiza la interfaz con el usuario y el manipulador de archivos de especificación de la red.

6.7.1 INTERFAZ CON EL USUARIO

Este módulo provee al usuario una interfaz de comunicaciones para el manejo de todas las funciones definidas en el sistema Netmap. Esta interfaz utiliza menúes basados en ventanas que se abren en la pantalla de acuerdo a la función seleccionada. Si bién las funciones de este módulo han sido escritas en lenguaje C, se utilizó un conjunto de herramientas orientado especialmente a la generación de ventanas en pantalla. La figura 6.6 muestra una de las pantallas de la interfaz usuario. La interfaz usuario habilita el uso del *mouse* y las flechas del teclado. Cada opción del menú puede ser activada utilizando una letra de la palabra clave que la defina, la cual aparece resaltada en color diferente. Las opciones más utilizadas han sido conectadas a ciertas teclas de función del teclado.

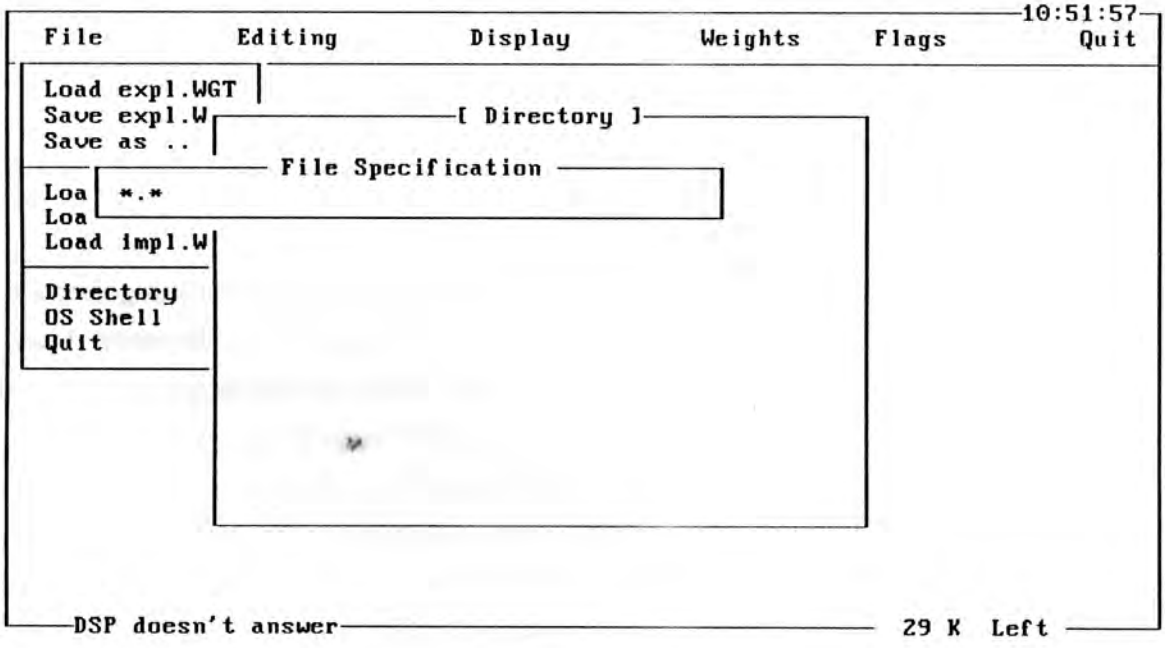

Figura 6.6 ejemplo interfaz usuario

Puede verse en la figura que la interfaz posee un menú en forma de barra, en el cual se hallan las opciones: *FILE, EDIT, WEIGHTS, BIAS,* y *QUIT.* Cada una de estas opciones posee un sub-menú acorde a las funciones que pueden implementarse.

El menú FILE permite leer (LOAD) y guardar (SAVE) archivos de pesos (*.WGT) y de direcciones (*.ADD) utilizando las funciones del módulo manipulador de archivos. Se implementó, además, una función que permite acceder a los directorios, permitiendo moverse por todo el árbol y seleccionando directamente un archivo deseado.

El menú EDIT permite editar un archivo nuevo o abrir para modificar un archivo existente de pesos, direcciones, patrones o de especificación de la red neuronal. Existen dos funciones principales que se encargan de la edición de un archivo *runedit* y *editnew.*

La opción DISPLAY muestra en pantalla el contenido de un archivo de pesos o direcciones, o el contenido de un conjunto de coeficientes internos del Etann. En el primer caso se muestran en pantalla los valores del último archivo cargado en memoria, en el caso de los coeficientes internos, se abre una ventana de selección del grupo de coeficientes de interés.

El menú WEIGHT provee opciones para programar leer o poner a cero una sinapsis en particular o un conjunto de ellas indicadas por medio de un archivo. En el caso de programar se usa un archivo *.WGT y en los otros casos uno *.ADD.

La opción BIAS repite las funciones especificadas en el párrafo anterior pero para coeficientes umbrales en lugar de pesos.

Por último el menú QUIT permite salir del programa o acceder al sistema operativo pudiendo retornar con el comando Exit.

La interfaz usuario del sistema Netmap utiliza los servicios de los otros módulos tales como el manipulador de archivos, el de control de las funciones

de la red neuronal y el simulador neuronal. Provee, además, el acceso hacia la interfaz gráfica de visualización de los pesos de la red neuronal.

Las funciones principales en este módulo son listadas en la tabla 6.2

FUNCION	**Descripción**
veryf_handsk	verifica funcionamiento de la placa DSP
loadwgt_file	carga archivo de pesos
loadadd_file	carga archivo de direcciones
loadwgt_net	carga archivo de descripción de la red neuronal
savewgt_file	guarda archivo de pesos
disply_txt	muestra un texto en cualquier ventana abierta
progr_from_file	programa pesos según lo especificado en file
read_from_file	lee pesos según lo especificado en file
zero_from_file	pone a cero pesos según lo especificado en file
progr_single	programa un único peso
read_single	lee un único peso
zero_single	pone a cero un único peso
progr_bias	programa un único umbral
read_bias	lee un único umbral
zero_bias	pone a cero un único umbral
disply_histogram	muestra un histograma en pantalla
dump_neuron	lee los coeficientes del Etann
menu	genera los menúes en pantalla
edit_new	abre un archivo para edición
dos_shell	función de escapa al sistema operativo

Tabla 6.2 Funciones del módulo de interfaz con el usuario

6.7.1.1 MÓDULO MANIPULADOR DE ARCHIVOS

En este módulo se generan todas las funciones necesarias para la manipulación de archivos de especificación de una red neuronal. Las funciones que pueden ejecutarse sobre el sistema **Netmap** son de lectura de los

coeficientes internos, de modificación de esos coeficientes y de procesamiento paralelo de una red específica. En consecuencia se definen 4 tipos de archivos que pueden ser manipulados por estas funciones.

- Archivo de solo direcciones de sinapsiss y/o umbrales.
- Archivo de pares de direcciones y pesos de sinapsis y/o umbrales.
- Archivo de especificación de una estructura de red con una o dos capas.
- Archivo de patrones de entrada

El primero solo contiene direcciones en decimal (*default*) o hexadecimal y esta destinado a comandos de lectura de los coeficientes ya cargados en el Etann. Estos archivos utilizan la extensión .ADR.

El segundo especifica el coeficiente de peso asociado a una sinapsis en particular y es el formato de archivo utilizado en los comandos de modificación de pesos. Estos comandos se encargan de ejecutar la función de programación sobre las direcciones especificadas en el archivo y con los valores de peso que ahi se encuentran. Las direcciones pueden indicarse por su representación decimal o hexadecimal y los pesos en punto flotante. Estos archivos usan la extensión .WGT

El archivo de patrones contiene uno o más patrones que se desea sean procesados por la red neuronal en modo PDP. Debido a que todos los patrones excitan las entradas de una estructura de red previamente definida, sólo se necesita un encabezamiento que indique la longitud del patrón.

6.7.1.1.1 Formato del archivo de especificación de la red neuronal

El archivo de especificación de una red neuronal define la estrucrura de una dada red con una o dos capas de procesamiento. El formato de este archivo contiene las palabras clave (*keywords*): **LAYER**, **NEURON**, **WEIGHT**, **BIAS**. La sintaxis del archivo debe ser tal que se especifique claramente cada peso a que sinapsis-neurona-capa de la red corresponde. Se crea una jerarquía de niveles en la cuál **LAYER** es el nivel más alto, seguido por **NEURON** y luego **WEIGHT** y

BIAS. En consecuencia la palabra **LAYERx**: determina que todo lo que sigue dentro del archivo pertenece a neuronas que están en la capa **x**. Asimismo, la palabra **NEURONy**: indica lo mismo dentro del dominio de la neurona **y**. La tabla 6.3, a continuación, es un ejemplo de especificación de una red neuronal.

LAYER0	número de capa 0
Neuron2	número de neurona bajo capa 0
weight	valores pesos en orden direc. ascendentes
0:18 5.0	
2.5 2.4 -0.3	
39:63 1.0	
bias	valores umbrales en orden direc. ascend.
-1.2	
1:7 4.0	
Neuron3	número de neurona bajo capa 0
bias	
-1.1 2.3	
LAYER1	número de capa 1
Neuron5	
weight	
0.02 0.05 1.2 1.8 -1.2	
10:15 1.0	
bias	
0:7 -.5	
end	

Tabla 6.3 Formato de archivo de especificación de una red neuronal

La tabla muestra que pueden especificarse conjuntos de coeficientes que poseen un mismo valor utilizando la notación <desde**XX**:hasta**YY** valor> en el cual **XX** e **YY** representan la primera y última sinapsis del conjunto que quiere configurarse con un mismo valor.

Los valores de pesos pueden ser generados con distintos simuladores neuronales, los cuales no poseen, en general, una normalización de los

Los simuladores neuronales, en general, no poseen programas que realicen una normalización de los coeficientes de peso. En consecuencia, el máximo valor de peso generado por un algoritmo de aprendizaje depende del tipo de aplicación y de la estructura de la red (número de capas, neuronas y pesos). Como el Etann posee un rango dinámico fijo, entre -2.5 y 2.5, este módulo se encarga de hacer una normalización respecto del máximo valor absoluto de los pesos contenidos en el archivo.

Se asume que todos los coeficientes que no figuran en el archivo de especificación de la red neuronal son iguales a cero.

6.7.1.1.2 Funciones del módulo manipulador de archivos

Este módulo utiliza nueve funciones principales en el manejo de los archivos anteriormente descriptos. Estas funciones son las indicadas en la tabla 6.4

load_wgt_file
store_wgt_file
parse_wgt_fle
load_add_file
store_add_file
extract_token
parse_net_descr
allowed_action
get_patt

Tabla 6.4 Funciones del manipulador de archivos

Las funciones de lectura (*load*) y escritura (*save*) se aplican a los archivos de direcciones (.ADD) y de pesos (.WGT). La función de lectura maneja la asignación de memoria para los datos leídos desde el archivo. Controla el formato del archivo y provee errores de diversos tipos tales como: error de

lectura, error de ubicación de memoria, error de formato, etc. La función de escritura permite guardar en un archivo los datos leídos o procesados por el Etann.

Las funciones *parse_wt*, *parse_net*, *allowed_action* y *extract_token* están destinadas a verificar el formato de los archivos de entrada. Estas funciones generan una salida de error indicando el tipo de error y el lugar donde este se ha producido. En particular la función *allowed_action* genera una lista de las funciones permitidas que pueden efectuarse sobre un archivo particular. La función *get_patt* lee un patrón de un archivo de patrones de entrada

6.8 DISCUSION

En el presente capítulo se ha dado una visión de la neurocomputadora Netmap desde el punto de vista de la programación y optimización de las funciones de entrenamiento y procesamiento de patrones. Se analizó la inclusión de la red neuronal dentro del lazo de aprendizaje y la posibilidad de incluir un simulador con características personalizadas respecto de cada neurona de la red. En función de optimizar el tiempo de procesamiento y programación, se propusieron técnicas de medición de los coeficientes de peso, simulaciones del modelo de la memoria analógica para generar tablas de valores de los pulsos de programación y solapamiento en la transferencia de patrones para aumentar el ancho de banda del mismo.

En este capítulo se detalla la interfaz entre la neurocomputadora y el usuario y la forma en que se especifican les redes neuronales, permitiendo una acción interactiva entre ambos.

CAPITULO 7

CONCLUSIONES Y DESARROLLOS FUTUROS EN TEMAS VINCULADOS A ESTA TESIS

7.1 CONCLUSIONES

En los capítulos precedentes de esta tesis se ha puesto de manifiesto la importancia que posee crear modelos electrónicos de computación paralela para redes neuronales. Si bien la teoría de redes neuronales ha desarrollado métodos novedosos para la resolución de problemas de ingeniería, la implementación paralela de estos algoritmos es esencial. Mas aún, la continua evolución de los modelos neuronales y su creciente complejidad hacen que las computadoras convencionales sean excesivamente lentas trabajando en simulación. La evolución de las computadoras secuenciales es rápida y compensa en ciertos aspectos este déficit, sin embargo muchos problemas y aplicaciones están aún, a órdenes de magnitud de esas potencias de cómputo. Como fue dicho en el capítulo 3, la eficiencia de procesadores convencionales en paralelo disminuye rápidamente con el numero de ellos. El problema de transmisión de información entre procesadores es una limitación seria en la eficiencia del sistema.

En general, los modelos neuronales son muy diferentes entre si y poseen distintas áreas de aplicación. Por ello la discusión se centró en una porción del dominio de aplicación de redes neuronales, los modelos estáticos multicapa. En el capítulo 2 se desarrolló la teoría básica de estos modelos neuronales aptas para la aproximación de funciones continuas y booleanas y para el mapeo de patrones de señales. En este capítulo se analizó como una red neuronal de dos capas es capaz de aproximar una curva en un espacio tridimensional dado por una función de potencial $V=f(x_1,x_2)$. La aproximación de funciones por métodos adaptivos es particularmente interesante para resolver problemas de la ingeniería asociados al procesamiento de señales sensoriales, utilizando redes neuronales estáticas.

En el caso particular estudiado en este capítulo, se planteó la navegación segura de un móvil en un ambiente con obstáculos. Este problema fue resuelto con una red neuronal gaussiana de dos capas. La conveniencia de utilizar este modelo frente al MLP radica, fundamentalmente, en la característica local de la señal de información. En un ambiente proveniente de una señal de video, cada punto tiene una alta correlación con sus vecinos cercanos (obstáculos o espacio libre) pero no con puntos más lejanos del miesmo ambiente. Las neuronas gaussianas son mejores para hacer aproximaciones localizadas, mientras que los MLP hacen una aproximación global del mapa de entrada/salida.

Este problema marca una de las lineas futuras de investigación a posteriori de esta tesis. El problema de navegación es más complejo aún si se permite que los objetos del ambiente se muevan con trayectorias no conocidas. Para este caso, si bien pueden usarse algunas técnicas conocidas, tales como funciones de potencial y transformadas de distancia, deben computarse con la velocidad de cambio del ambiente. En consecuencia, la aplicación de redes neuronales, y mas aún, la implementación de modelos neuronales en electrónica paralela, es escencial. Al respecto se planean trabajos de modelado de unidades y sistemas neuronales específicos a la función de cómputo del flujo óptico para la navegación autónoma en un medio con obstáculos móviles. Ya se han obtenido algunos modelos analógicos e ideas para generar electrónica paralela de computación del flujo óptico que permiten generar trayectorias óptimas en tiempo real.

En el capítulo 3 se pusieron en evidencia las propiedades más importantes de los modelos analógicos, digitales e híbridos para la implementación de modelos neuronales, tanto a nivel de la celda neuronal mínima como a nivel del sistema neurocomputador. Se destacó en cada caso la influencia de los modelos analógicos y digitales que caracterizan al sistema. Es obvio que no existe una única opción sobresaliente tanto para la generación de un VLSI neuronal, como para la de un sistema neuronal. De todas maneras, se ha optado por el modelo analógico tanto para el diseño de una celda neuronal como para el de una neurocomputadora. Las razones para esta decisión fueron basadas, fundamentalmente, en la gran cantidad de neuronas que pueden integrarse en un mismo dispositivo, el elevado ancho de banda de procesamiento, que

permite realizar tareas complejas en tiempo real, y la mejor adaptación de los sistemas analógicos, a los problemas de interés pudiendo conectarse directamente a señales sensoriales y actuadoras externas. La computación neuronal analógica es un campo poco explorado y enormemente atractivo para el procesamiento de señales en tiempo real, tales como las provenientes de una cámara de visión en un problema de navegación robótica.

En el capítulo 4, en concordancia con lo expresado en los capítulos 2 y 3, se diseñó un modelo de neurona cuya función de activación es gaussiana. Se presentó un esquema circuital compacto, el cual permite que un número elevado de estas unidades sean integradas en un mismo dispositivo. Una de las principales propiedades de los circuitos propuestos es la posibilidad de controlar ciertos parámetros vinculados directamente con las variables adaptivas en una red neuronal gaussiana. Es decir, existe una función lineal entre los pesos de una red neuronal y el control de dichos parámetros del modelo de la neurona gaussiana. Este modelo ha sido pensado para crear un sistema neuronal de propósitos generales con entradas y salidas analógicas. Aunque puede ser fácilmente adaptado a un sistema neuronal orientado a una aplicación en particular, como la mencionada en el párrafo anterior.

El sistema de computación neuronal desarrollado en los capítulos 5 y 6 pone de manifiesto la gran capacidad y flexibilidad de la implementación de redes neuronales analógicas. El sistema Netmap procesa 180.000 patrones/seg generando una potencia de cómputo equivalente a 120 veces la de un procesador RISC capaz de hacer una multiplicación y una suma en 33 nseg.

El sistema Netmap se desarrolló de manera que patrones con señalización analógica puedan ser conectados en forma directa, generando, también, una salida analógica. Sin embargo, se ha potenciado al procesador neuronal para funcionar como un sistema acelerador de una computadora secuencial. De esta manera, pueden construirse modelos neuronales multicapa de configuración variable, de igual modo que en simulación, pero con una red neuronal analógica real. El ambiente creado es autónomo, permite, a través de una interfaz gráfica, especificar y entrenar redes neuronales con el dispositivo analógico dentro del lazo de aprendizaje. Se ha generado, también, un simulador neuronal, con modelos matemáticos y numéricos, para acelerar el

aprendizaje de la red. En el camino, se dio solución a ciertos problemas derivados de la tecnología de memorias analógicas con compuerta flotante, en la medición y programación de los coeficientes de peso de la red neuronal. La realización de la neurocomputadora demandó un cuidadoso diseño y optimización de los canales de comunicaciones en el cual intervienen cerca de 200 circuitos integrados, divididos en dos placas. El gran ancho de banda de procesamiento de la red neuronal impone una exigencia de temporización de los circuitos involucrados en el orden del nanosegundo.

Desarrollos futuros, permitirán aumentar la potencia de cómputo de este sistema generando una estructura de múltiples dispositivos. Una organización multicapa, como la mencionada en el capítulo 5.1, no solo potenciaría la unidad de cómputo, sino que permitiría aumentar el número de capas de la red neuronal.

Otra línea futura de trabajo, consistiría en utilizar el modelo neuronal gaussiano en reemplazo del procesador neuronal del sistema Netmap. Si bien, como se dijo, ambos modelos neuronales tienen dominios de aplicaciones comunes, el rendimiento de una red neuronal con uno u otro modelo es función de la naturaleza de los patrones a procesar. El modelo del sistema Netmap se adapta a estos posibles cambios futuros.

En síntesis, los rápidos avances en la teoría de redes neuronales y en las tecnologías de integración electrónica permiten predecir una mayor integración de los modelos neuronales en sistemas de cómputo orientados a tal propósito. El trabajo realizado ha pretendido avanzar en esa dirección integrando funciones neuronales a nivel unitario y de sistema. Esta línea novedosa es aún incipiente y tomará sin duda mucho mayor vuelo en los próximos años. En los cuales se verán no solo neurocomputadoras de propósitos generales sino, además, una gran cantidad de funciones orgánicas, sensoriales y actuadoras controladas por medio de electrónica neuronal especializada.

Apéndice A

Efecto Tunel Fowler-Nordheim

A.1. EFECTO TUNEL EN DISPOSITIVOS DE COMPUERTA FLOTANTE

Entre las tecnologías emergentes para el diseño de memorias analógicas y digitales reprogramables eléctricamente, se destaca aquella que utiliza almacenamiento de electrones, que por efecto túnel pasan de una superficie de silicio (Si) a un medio aislante tal como el dióxido de silicio (SiO_2) (fig. A.1) (Nordheim, 1928). Circuitalmente, un transistor MOS con compuerta flotante puede variar la tensión de umbral V_{TO} en función de la carga almacenada.

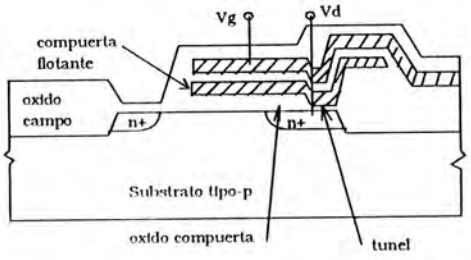

FIGURA A.1 MOS de compuerta flotante

La energía entre las bandas de conducción y de valencia es de aproximadamente 1,1 eV en el Si y de 9 eV en el SiO_2. En una interfaz Si-SiO_2 la barrera de potencial es de 3,25 eV, en consecuencia, a temperatura ambiente, la energía de los electrones es baja y la probabilidad que pasen la juntura es casi nula. Fowler y Nordheim demostraron que en presencia de un campo eléctrico elevado, las bandas de energía se distorsionan aumentando la probabilidad que electrones pasen al SiO_2. Esta corriente sigue la ley (Ellis, 1982)

$$J_{FN} = \frac{q^3 \quad E^2}{8 \quad \pi \quad h \quad \Phi_B} . \exp\left(\frac{-4 \quad \sqrt{2m} \quad \Phi_B^{3/2}}{3hqE} \right) \qquad \text{(A.1)}$$

Hay dos estructuras típicas que alcanzan campos eléctricos suficientemente altos para producir efecto túnel. Una de ellas utiliza una capa de oxido muy fina (~100 Å) entre superficies de Si planas. La segunda usa superficies de Si no planas (texturadas) y una capa de óxido más gruesa (hasta 800 Å).

La primera de estas estructuras sufre el inconveniente que el campo eléctrico necesario para producir efecto túnel es elevado, entre 20 y 25 volts, con tensiones cercanas a la de ruptura de la juntura.

La segunda estructura se ve favorecida por el hecho que la textura de la superficie de Si, aumenta el área de emisión de electrones. Las protuberancias en la superficie del Si son de 200 a 300 Å de alto y 1000 a 1500 Å en la base. Tensiones mayores de 14v producen densidades de corriente apreciables ($>10\mu A/cm^2$). Como contrapartida, la textura de la superficie emisora dificulta el cálculo de E en la ecuación A.1.

A.2 MÉTODOS QUE USAN AISLANTES DE OSi GRUESOS

El cálculo de E ha sido durante mucho tiempo un problema difícil, debido a que las lineas de campo no siguen trayectorias paralelas en este tipo de superficies. Las soluciones propuestas, basadas en aproximaciones y simplificaciones de la superficie emisora, no concuerdan con los datos obtenidos experimentalmente en un rango amplio de tensiones. Los parámetros son en general elegidos arbitrariamente y deben ser modificados para cubrir distintos rangos de tensiones.

Ellis (Ellis,1982) resolvió el problema por medio de la geometria diferencial, transformando el espacio Euclideano en un espacio en el cual las lineas de E son paralelas. La densidad de corriente por efecto túnel puede expresarse:

$$J_{colectado} = \frac{\left(\int_S J_{FN} . dA \right)_{emitido}}{\left(\int_S d\, A_{colectado} \right)} \qquad \text{(A.2)}$$

donde J_{FN} responde a la ecuación en (A.1). La solución propuesta está basada en la invariancia del gradiente en superficies equipotenciales. El gradiente de V es ortogonal al plano tangente a cualquier superficie equipotencial que contiene un punto P entre las superficies emisora (B1) y colectora (B2). Sean $\xi_1 y \xi_2$ vectores que forman una base del plano tangente a la superficie equipotencial que contiene a P. Las ecuaciones del potencial en este sistema son:

$$\frac{d^2 V}{d\xi_3^2} = 0 \qquad \xi_3 = \xi_1 \wedge \xi_2 \qquad \text{(A.3)}$$

Las condiciones de contorno en estas superficies son

$$V(B_1) = V_\Lambda \qquad V(B_2) = 0 \qquad \text{(A.4)}$$

Haciendo una transformación a un sistema de coordenadas esféricas, el campo eléctrico puede definirse como:

$$E = -\nabla V \qquad E = \left(-dV/d\xi_3\right)\left|d\xi_3/d\theta\right|\left[\left|d\theta/dr\right| + \frac{1}{|r|} e_\theta\right] \qquad \text{(A.5)}$$

Luego el potencial:

$$V(r) = V_\Lambda \frac{|\xi_3(r)| - |\xi_3(B_1)|}{|\xi_3(B_2)| - |\xi_3(B_1)|} \qquad \text{(A.6)}$$

La expresión original (A.1) fue derivada para un campo eléctrico uniforme, la expresión en (A.5) describe también un campo eléctrico uniforme pero en un sistema de coordenadas no-lineal. Esta expresión tiene varias propiedades interesantes. El campo eléctrico es inversamente proporcional a la diferencia de las curvaturas de las superficies emisora y colectora. Además, depende de la derivada de la curvatura de la superficie emisora. La figura A.2 muestra coincidencia entre los datos calculados y los resultados obtenidos experimentalmente en un rango de tensiones entre 12 y 20 volts.

El mejoramiento en el efecto túnel es aproximadamente 4 veces respecto del que usa campo eléctrico uniforme.

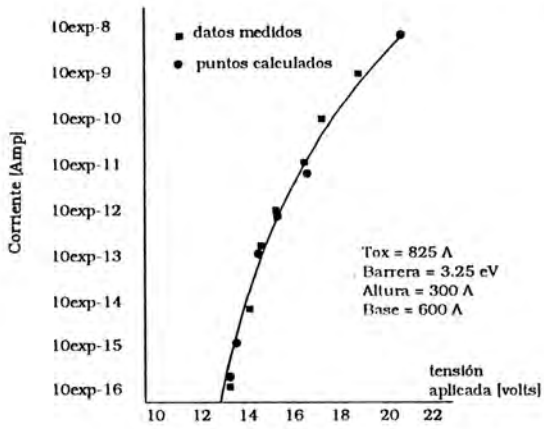

FIGURA A.2 Corriente tunel en superficies texturadas

Una alternativa a los dispositivos con superficies texturadas es la planteada por Carley (Carley, 1989) (Kolodny, 1986). En este caso se propone una arquitectura especial para el diseño del transistor Las corrientes electrónicas por efecto túnel hacia y desde la compuerta flotante son menores que en el caso anterior, pero el método tiene la ventaja de poder ser utilizado en procesos de fabricación CMOS estandar. Esto hace disponible el uso de dispositivos con compuerta flotante a un número mayor de usuarios que no tienen acceso a los procesos de superficies texturadas. En este caso, la mejora del campo eléctrico necesario para producir corrientes apreciables por efecto túnel se consigue modificando la forma del inyector. El rectángulo de polisilicon (fig. A.3) concentra líneas de campo en las esquinas. Si bien las expresiones analíticas son muy complejas y por lo tanto no están disponibles; experimentalmente pueden obtenerse curvas como las de la figura A.4a y b.

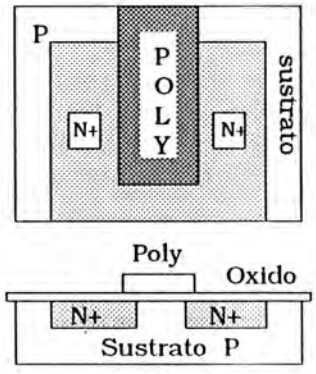

FIGURA A.3 Vista superior y corte transversal del MOS

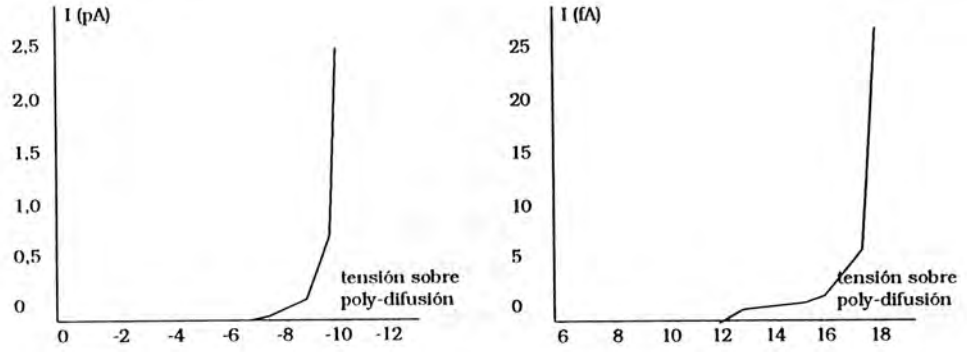

FIGURA 4 Corriente tunel en MOS de película delgada

Otra característica interesante de esta estructura es que el inyector de corriente se comporta como una fuente de corriente constante para rangos de tensión de unos 3 volts (entre 12,5v y 15,5v).

Cualquiera de las alternativas consideradas posee una buena estabilidad del potencial adquirido por la compuerta flotante debido a que las tensiones de trabajo del drenador y de la compuerta accesible del MOS están muy por debajo de aquellas necesarias para producir efecto túnel. La figura A.5 muestra la variación de la tensión umbral Vt en función del tiempo para distintos valores de tensión en la compuerta accesible Vg. La variación de Vt para tensiones normales de operación en la compuerta (hasta 5v), es prácticamente nula inclusive en un periodo de tiempo de varios años.

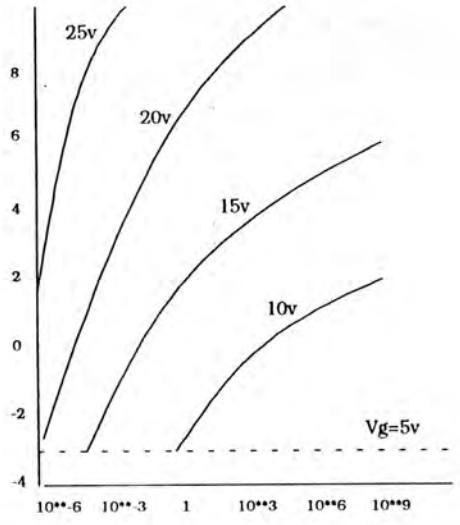

FIGURA A.5 variación de la tensión umbral en función del tiempo

A.3 MODELOS QUE USAN AISLANTES DE OSi DELGADOS

Durante los últimos años se ha producido un importante desarrollo de memorias EEPROM (Electrically Erasable Programable Read Only Memory) (Kolodny, 1986) (Euzent, 1981) (Wang, 1980). Estas memorias digitales se basan en el almacenamiento de un bit de información (0 o 1) el cual puede ser dinámicamente modificado aplicando un pulso de tensión suficientemente elevado (entre 14 y 25v) a la compuerta de un transistor MOS que posee, además, una segunda compuerta flotante. Al aplicar un pulso de tensión elevado, se genera una corriente de electrones que, por efecto túnel, pasa a la compuerta flotante y permanecen almacenados hasta que sean removidos por un pulso similar en el drenador del dispositvo. Estas operaciones se denominan, respectivamente, de *escritura* y de *borrado* de la memoria. En las memorias digitales este proceso es simple ya que no se necesita precisión en el cálculo de la carga que debe ser enviada por efecto túnel a la compuerta flotante para almacenar un 0 o un 1 lógico. Si se genera un pulso de programación excesivamente largo, la corriente de carga hacia la celda satura.

En la actualidad, las técnicas de almacenamiento en MOS de compuerta flotante están siendo depuradas para lograr memorias analógicas. Estas memorias permiten el almacenamiento de una carga que puede tomar cualquier valor dentro de un intervalo continuo.

Las áreas de la ingeniería electrónica tales como redes neuronales, computación paralela de señales analógicas, procesamiento de señales (video, voz, etc.), análisis de patrones de información, etc., realizan procesamiento sobre señales analógicas. Las técnicas de procesamiento digital por computadora que se han utilizado durante las últimas décadas, generan un elevado volumen de datos y requieren una gran potencia de cómputo de las unidades de proceso. La aparición de circuitos analógicos adaptivos o memorias analógicas puede ayudar en buena forma, acelerando el cómputo de diversos tipos de funciones tales como multiplicaciones, sumas, procesamiento no lineal, etc.

Las memorias analógicas basadas en celdas MOS con compuertas flotantes varían la tensión de umbral, Vt, de un transistor MOS. A diferencia de las memorias digitales, en las cuales solo interesa conocer cual es el cuanto mínimo de carga necesario para pasar de un valor 0 almacenado a un valor 1 (o viceversa), en las memorias analógicas se necesita tener un modelo adecuado del control de la carga que se almacena por efecto túnel. La exactitud requerida en el valor de carga almacenado depende de la aplicación en la que se utiliza la memoria, pero valores de 1% son típicamente necesarios.

A.3.1 MODELO DE UNA CELDA MOS CON COMPUERTA FLOTANTE (Kolodny, 1986)

Se considerará dispositivos MOS de compuerta flotante que utilizan un aislador fino de óxido de silicio, cuya estructura se indica en la figura A.1. El campo eléctrico a través del oxido de silicio es uniforme, por lo que la densidad de corriente debida al efecto túnel puede ser descrita por la ecuación:

$$J_{tun} = \alpha E_{tun}^2 . \exp(-\beta/E_{tun}) \qquad (A.7)$$

donde E_{tun} es el campo eléctrico en el óxido y α y β son constantes. Además E_{tun} está dado por:

$$E_{tun} = |V_{tun}| \ / \ X_{tun} \qquad (A.8)$$

donde V_{tun} es la caida de tensión en el óxido y X_{tun} su espesor.

A.3.2 CALCULO DE V_{tun}

Un circuito equivalente simplificado del proceso túnel se muestra en la figura A.6. C_{pp} es la capacidad inter-poly, C_{tun} es la capacidad del oxido de silicio en la zona de túnel y C_{gox} es la capacidad del óxido entre la compuerta flotante y el

substrato. Sea Q_{fg} la cantidad de carga almacenada en la compuerta flotante. V_{tun} puede expresarse en términos de acoplamiento:

$$|V_{tun}|_{write} = V_g.K_w \qquad (A.9)$$

donde,

$$K_w = \frac{C_{pp}}{C_{pp} + C_{gox} + C_{tun}} \qquad (A.10)$$

y

$$|V_{tun}|_{erase} = V_d.K_e \qquad (A.11)$$

donde

$$K_e = 1 - \frac{C_{tun}}{C_{pp} + C_{gox} + C_{tun}} \qquad (A.12)$$

Los acoplamientos K_w y K_e denotan que porcentaje de la tensión de programación (borrado) aparece a través del óxido de silicio. Las ecuaciones (A.9) y (A.11) son aplicables solamente cuando la carga almacenada Q_{fg} es nula. Si existe carga negativa almacenada, durante la operación de escritura, la tensión de túnel se reduce a

$$|Vtun|_{write} = V_g \cdot K_w + \frac{Q_{fg}}{C_{pp} + C_{gox} + C_{tun}} \qquad (A.13)$$

En la operación de borrado, la carga negativa inicial, incrementa la tensión en el óxido de silicio a

$$|V_{tun}|_{erase} = V_d \cdot K_e - \frac{Q_{fg}}{C_{pp} + C_{gox} + C_{tun}} \qquad (A.14)$$

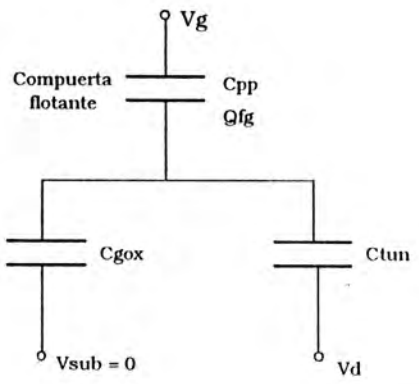

FIGURA A.6 Modelo simplificado de la celda EEPROM

A.3.3 CALCULO DE LA TENSIÓN UMBRAL Vt

La tensión inicial de umbral con $Q_{fg}=0$ es V_{ti}. Los pulsos de escritura/borrado alteran la tensión umbral de acuerdo a la relación

$$\Delta Vt = -\frac{Q_{fg}}{C_{pp}} \qquad (A.15)$$

Usando las ecuaciones (A.13) y (A.14) pueden obtenerse las tensiones de umbral luego de un pulso de programación

$$V_{tw} = V_{ti} - \frac{Q_{fg}}{C_{pp}} = V_{ti} + V_g\left[1 - \frac{V_{tun}}{K_w \cdot V_g}\right] \qquad (A.16)$$

$$V_{te} = V_{ti} - \frac{Q_{fg}}{C_{pp}} = V_{ti} - V_d\left[\frac{K_e}{K_w} - \frac{V_{tun}}{K_w \cdot V_d}\right] \qquad (A.17)$$

V_{tw} es la tensión umbral de la memoria luego del pulso de escritura y V_{te} la tensión umbral luego de un pulso de borrado. Este proceso tiene una dinámica asociada que depende de la carga Q_{fg} que pasa por efecto túnel. Durante el proceso de escritura (borrado) la variación de carga, modifica la tensión V_{tun} y, por consiguiente, el campo eléctrico aplicado al óxido de silicio. Durante el proceso de escritura, la carga negativa que se almacena tiende a disminuir el campo eléctrico hasta que el efecto túnel se detiene. Esto se produce para valores de unos 10^7volt/cm.

Para optimizar la transferencia de carga, las constantes de acoplamiento K_w y K_e deben ser lo mas próximo posible a uno. En la práctica alcanzan valores cercanos a 0,7.

A.3.4 DINÁMICA DE Vt DURANTE LA PROGRAMACIÓN/BORRADO

La variación de la carga que pasa por efecto túnel a la compuerta flotante es

$$\frac{dQ_{fg}}{dt} = A_{tun} \cdot J_{tun} \qquad (A.18)$$

Resolviendo esta ecuación diferencial para

$$|Vtun|_{write} = V_g \cdot K_w + \frac{Q_{fg}}{C_{pp} + C_{gox} + C_{tun}}$$

y

$$J_{tun} = \frac{\alpha}{X_{tun}^2} V_{tun}^2 \cdot \exp(-\beta X_{tun}/V_{tun})$$

obtenemos

$$V_{tw}(t) = V_{ti} + V_g - \frac{1}{K_w} \cdot \frac{B}{\ln(A.B.t + E_1)} \qquad (A.19)$$

De igual forma, utilizando (A.7), (A.8), (A.14) y (A.18) obtenemos la expresión de Vt para un pulso de borrado

$$V_{te}(t) = V_{ti} + V_d \cdot \frac{K_e}{K_w} + \frac{1}{K_w} \cdot \frac{B}{\ln(A.B.t + E_2)} \qquad (A.20)$$

donde

$$A = \frac{A_{tun} \cdot \alpha}{X_{tun} \cdot (C_{pp} + C_{gox} + C_{tun})}$$

$$B = \beta . X_{tun}$$

$$E_1 = \exp\left[\frac{B}{K_w \cdot (V_g + V_{ti} + V_t(0))}\right]$$

$$E_2 = \exp\left[\frac{B}{V_d \cdot K_e + K_w \cdot V_t(0) + K_w \cdot V_{ti}}\right]$$

Vt(0) es la tensión umbral antes del pulso de programación. La constante de tiempo del sistema puede ser definida como

$$\tau = \frac{1}{A.B}\exp\left[\frac{B}{K_w(V_g + V_{ti} - V_t(0))}\right]$$

Como puede verse en la ecuación (A.19) para valores grandes de tiempo t la función aproxima

$$V_{tw}(t) = V_{ti} + V_g - \frac{1}{K_w}\cdot\frac{B}{\ln(A.B.t)}$$

Similarmente, se encuentra una ecuación para $V_{te}(t)$. Estas expresiones permiten evaluar compromisos entre los distintos factores intervinientes en la operación de programación (borrado), tales como tiempo de programación, retención y tensiones aplicadas V_g y V_d. Los parámetros α, β, V_{ti}, K_w y K_e deben ser calculados o medidos experimentalmente.

La medición de los parámetros tales como espesores de la capa de óxido de silicio y polisilicones se hace por técnicas C-V. En cambio, la medición de los parámetros mencionados en el párrafo anterior es dificil y deben utilizarse técnicas especiales para la tecnología de compuerta flotante. El factor de acoplamiento Kw se mide comparando tensiones de umbral o transconductacias entre un MOS común y uno de compuerta flotante. Ambos en el mismo punto de operación. El factor de acoplamiento K_e puede hallarse de forma similar comparando las pendientes de la característica V_t-V_d entre un MOS común y uno con compuerta flotante.

Los parámetros α y β se extraen de las características I-V del efecto tunel. Existen ciertas controversias en la forma en que estos parámetros son medidos debido a efectos de trmpas de carga en el óxido de silicio. Por simplicidad, α y β pueden medirse luego que haya circulado una carga superior a 0.1 Q/cm^2, luego que el efecto de trampa de carga ha saturado. Basado en estas consideraciones, $\alpha = 1{,}88\times10^{-6}\,A/V^2$ y $\beta = 2{,}55\times10^{8}\,V/cm$ son valores típicos para espesores de óxidos de 100 a 200 Å.

Apéndice B

EL TRANSISTOR MOS

B.1 MODELOS DEL TRANSISTOR MOS

En este apéndice el desarrollo de los modelos para las distintas zonas de funcionamiento del MOS se hace en base a potenciales de contacto, en lugar de utilizar la teoría de bandas de energía (Massobrio, 1993).

Un potencial de contacto se genera en una juntura de dos materiales, semiconductores o metales. Si los portadores se encuentran con diferentes niveles de concentración a ambos lados de la juntura, la cruzan y se difunden, dejando del otro lado una carga de polaridad opuesta. Estas cargas crean un campo eléctrico que se va oponiendo al movimiento de cargas. En el punto de equilibrio el potencial electrostático será ϕ_{MS} (Tsividis, 1987).

En primera aproximación puede considerarse la estructura del MOS que se forma apilando un aislador (SiO_2) y una compuerta (*gate*) metálica o polisílica, sobre un substrato semiconductor de bajo dopaje.

La figura B.1a muestra la distribución de cargas debido al potencial de contacto en una juntura MOS con substrato tipo p. La neutralidad de carga en los materiales de la juntura puede reestablecerse usando una fuente externa de valor ϕ_{MS} como indica la figura B.1b.

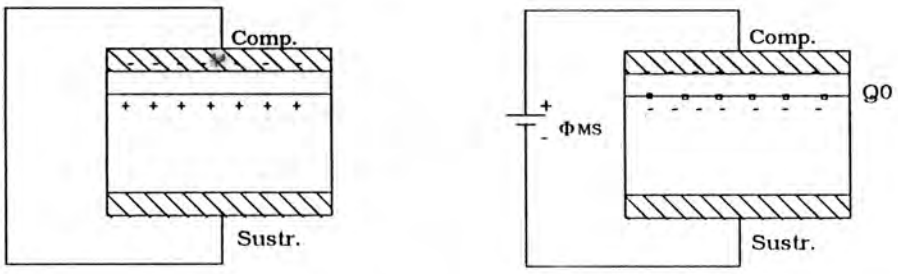

FIGURA B.1 a) Distribución de cargas en la juntura b) Neutralización

Otro fenómeno que genera cargas en el proceso de fabricación MOS se debe a la forma en que se crea la juntura abrupta y cómo se deposita el aislante. En el proceso se crean iones y cargas netas positivas de valor Q_0. Esta carga induce una carga negativa $-Q_0$ en la compuerta, que puede ser compensada externamente con una batería de valor $-Q_0/C_{ox}$, donde C_{ox} es la capacidad del aislante. Los valores de carga y capacidad se expresan por unidad de area.

Puede definirse, entonces, el potencial de "banda plana" (*flat band*) como el total externo necesario para mantener la estructura en forma neutra.

$$V_{FB} = \phi_{MS} - \frac{Q_0}{C_{OX}} \qquad \text{(B.1)}$$

B.2 BALANCE DE POTENCIALES Y CARGAS (Massobrio, 1993)

Si se aplica una fuente externa V_{GB} de valor diferente a la tensión de banda plana, dicha tensión igualará la suma: $V_{GB} = \phi_{OX} + \phi_S + \phi_{MS}$. Donde ϕ_{ox} es la caída en el aislante y ϕ_s es el potencial de superficie.

De la misma manera, utilizando el principio de neutralidad de la carga, $Q_G + Q_0 + Q_{SC} = 0$, en la cual, Q_G es la carga neta en la compuerta y Q_{SC} la carga en la superficie del semiconductor.

Además de la condición de banda plana podemos tener, en una primera aproximación, tres estados diferentes llamados acumulación, deplesión e inversión.

B.2.1: Acumulación: Cuando $V_{GB} < V_{FB}$ se acumulan cargas negativas en la compuerta; éstas deben ser balanceadas por una cantidad equivalente de cargas positivas en la superficie del semiconductor; formándose un capacitor de placas planas y ancho igual al del aislante. La disminución de tensión en V_{GB} es absorbida por ϕ_{ox} y ϕ_s (Tsividis, 1987).

$$V_{GB} < V_{FB} \qquad Q_{SC} > 0 \qquad \phi_S < 0 \qquad \phi_{OX} < 0$$

$$C_{OX} = \frac{\varepsilon_{OX}}{t_{OX}} \qquad t_{OX} = k_{OX} \cdot \varepsilon_0$$

B.2.2: Deplesión: Si $V_{GB} > V_{FB}$ la carga de compuerta es positiva respecto del valor de banda plana. Estas cargas repelen los huecos del sustrato p, formandosé una zona de deplesión (Tsividis, 1987). Cuanto mayor sea V_{GB} , mayor será el ancho de la zona de deplesión. En consecuencia el valor de C_{ox} disminuye como se ve en la figura B.2.

$$V_{GB} > V_{FB} \qquad Q_{SC} < 0 \qquad \phi_S > 0 \qquad \phi_{OX} > 0$$

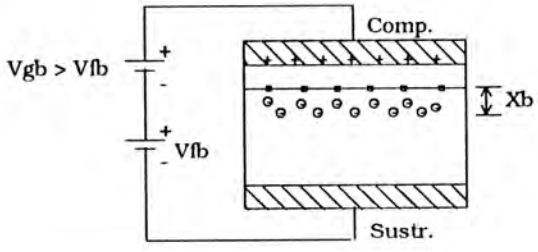

FIGURA B.2 MOS en deplesión

B.2.3: Inversión: (Massobrio, 1993) Si V_{GB} aumenta aun más, otro fenómeno ocurre, ϕ_s comienza a ser lo suficientemente elevado como para atraer electrones libres a la superficie. Cuando la densidad de electrones excede a la de huecos en la superficie, en el material tipo p, se considera que se llega al límite inferior de una zona llamada inversión. En este caso la capacidad vuelve a crecer producto de la reducción de la separación entre placas, que otra vez es t_{ox}.

Puede considerarse que la carga Q_{SC} en inversión está formada por dos tipos de carga, Q_B debida a las cargas por deplesión y Q_I. Para valores de V_{GB} suficientemente elevados puede considerarse que el efecto de inversión es dominante y Q_B despreciable.

Carga debida a electrones libres Q_I: La concentración electrónica en la superficie n_s está dada por la ley de Boltzmann en función de la concentración en el substrato

$$n_s = n_b \cdot e^{q\phi_s/kT}$$

Donde ϕ_s se mide respecto del interior del substrato donde el potencial es cero. Utilizando las expresiones $p_b \cong N_a \quad n_b \cong \frac{n_i^2}{N_a}$, el potencial de superficie puede escribirse como

$$n_s = N_a \cdot e^{q(\phi_s - 2\phi_p)/kT}$$

La concentración electrónica en cualquier otro punto diferente a la superficie, $n(y)$, puede hallarse sustituyendo ϕ_s por el potencial $\phi(y)$ correspondiente.

Cargas debidas a huecos en deplesión Q_B: Para hallar esta carga, utilizando leyes electrostáticas en la juntura, se hacen dos suposiciones: que el ancho de la zona debida a cargas en inversión es despreciable respecto al ancho de deplesión; y que la caída de potencial en esta zona es también despreciable. Estas suposiciones son validas especialmente en la zona en que Q_{SC} es debida mayormente a Q_B.

En dichas condiciones $N_D >> N_A$ y el ancho de la zona de deplesión es

$$x_B \cong \sqrt{\frac{2\varepsilon_s \phi_s}{q N_A}} \qquad \text{(B.3)}$$

entonces, la carga por unidad de volumen puede expresarse como

$$Q_B = -q N_A x_B = -\sqrt{2q\varepsilon_s N_A \phi_s} \qquad \text{(B.4)}$$

Utilizando las mismas leyes electrostáticas la carga Q_{SC} puede ser evaluada directamente (Sze, 1969) (Nicollian, 1982) (Massobrio, 1993)

$$Q_{SC} = -\sqrt{2q\varepsilon_s N_A \phi_s}\cdot\sqrt{\phi_s + \frac{kT}{q}\cdot e^{q(\phi_s - 2\phi_p)/kT}} \qquad \text{(B.5)}$$

La carga en inversión puede ser obtenida como diferencia de las ecuaciones (B.5) y (B.4)

$$Q_I = -\sqrt{2q\varepsilon_s N_A \phi_s}\cdot\left[\sqrt{\phi_s + \frac{kT}{q}\cdot e^{q(\phi_s - 2\phi_p)/kT}} - \sqrt{\phi_s}\right] \qquad \text{(B.6)}$$

La figura B.3 muestra las curvas de Q_B, Q_I y Q_{SC} en función del potencial de superficie ϕ_s. Es conveniente dividir el gráfico en cuatro zonas, una en la cual dominan claramente las cargas por deplesión, hasta ϕ_p, y otras tres denominadas de inversión débil, moderada y fuerte. ϕ_{MO} y ϕ_{HO} delimitan dichas zonas. ϕ_{MO} es ligeramente superior a $2\phi_p$, a partir del cual Q_I crece exponencialmente con ϕ_s.

La relación entre V_{GB} y ϕ_s puede deducirse de las ecuaciones (B.7) y (B.8)

$$V_{GB} = \phi_{OX} + \phi_S + \phi_{MS} \qquad \text{(B.7)} \qquad\qquad Q_G + Q_0 + Q_{SC} = 0 \qquad \text{(B.8)}$$

reemplazando, además, $Q_G = C_{ox}\cdot\phi_{ox}$

$$V_{GB} = V_{FB} + \phi_s + \gamma\sqrt{\phi_s + \frac{kT}{q}\cdot e^{q(\phi_s - 2\phi_p)/kT}} \qquad \text{(B.9)}$$

donde

$$\gamma = \frac{\sqrt{2q\varepsilon_s N_A}}{C_{ox}}$$

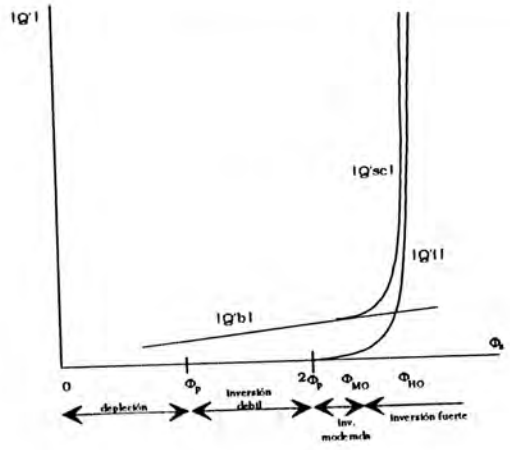

FIGURA B.3 Curvas de carga en función del potencial de superficie

B.3 OPERACION EN LA ZONA DE DEPLESION (Nicollian, 1982) (Massobrio, 1993) (Mead, 1989):

El problema general del MOS es complejo debido a que ϕ_s depende de la electrostática de cargas móviles en el canal, las cargas móviles dependen de ϕ_s a través de las ecuaciones de corriente, y las condiciones de contorno de la densidad de cargas móviles dependen de ϕ_s a través de la distribución de Fermi. Afortunadamente, este ciclo de dependencias puede eliminarse en la zona anterior al umbral V_{TO}.

Como fue dicho en la sección B.2, las variaciones de potencial de compuerta se distribuyen en variaciones de potencial en el óxido ϕ_{ox} y en la superficie ϕ_s. En consecuencia, podemos ver este efecto como una falta de eficiencia de la compuerta en el control de ϕ_s. Es decir, definimos

$$\kappa = \frac{\partial \phi_s}{\partial V_g} \qquad \text{(B.10)}$$

En deplesión, tenemos una zona de densidad de carga constante de ancho x_o. Suponiendo que el potencial es cero en un lugar del substrato más allá de la zona de deplesión, y considerando un campo eléctrico que se incrementa

linealmente debido a la densidad de carga constante, el potencial de superficie y la carga total en la zona de deplesión son

$$\phi_s = -\frac{\rho x_0^2}{2\varepsilon_s} \qquad Q_B = \rho x_0 = \sqrt{-2\rho\varepsilon_s\phi_s}$$

Entonces, podemos definir la capacidad de la zona de deplesión como

$$C_B = -\frac{\partial Q_B}{\partial \phi_s} = \sqrt{-\frac{\rho\varepsilon_s}{2\phi_s}}$$

Aplicando una tensión V_G a la compuerta, y considerando la tensión de banda plana,

$$V_G = \phi_s + \phi_{FB} - \frac{t_{ox}}{\varepsilon_{ox}}.Q_B = \phi_s + \phi_{FB} + \frac{\sqrt{-2\rho\varepsilon_s\phi_s}}{C_{ox}} \qquad \text{(B.11)}$$

De esta expresión es fácil ver que la efectividad de la tensión de compuerta es (Mead, 1989)

$$\frac{1}{\kappa} = \frac{\partial V_G}{\partial \phi_s} = 1 + \frac{1}{C_{ox}}.\sqrt{-\frac{\rho\varepsilon_s}{2\phi_s}} \qquad \text{(B.12)}$$

es decir la razón de incrementos de tensiones de compuerta y superficie es mayor que uno.

B.4 FLUJO DE CORRIENTE POR DIFUSION (Shichman, 1968) (Tsividis, 1987)

El transistor MOS se construye agregando a la estructura analizada en las secciones anteriores, dos regiones fuertemente dopadas de silicio de tipo contrario al del substrato llamadas fuente y drenador (figura B.5). Al aplicar una tensión entre fuente y drenador se genera una corriente de difusión debido a la diferente concentración de portadores a lo largo del canal. La concentración de partículas sigue la distribución de Boltzmann $N = N_0.e^{-qV/kT}$.

Las concentraciones en los terminales de fuente y drenador son, respectivamente, (Antognetti, 1982)

$$N_s = N_0 \cdot e^{-\Psi_s/kT} \qquad N_d = N_0 \cdot e^{-\Psi_d/kT}$$

donde

$$\Psi_s = \Psi_0 + q(\kappa V_G - V_s) \qquad \Psi_d = \Psi_0 + q(\kappa V_G - V_d)$$

donde ψ_0 es el potencial entre las regiones y el canal. En consecuencia

$$N_s = N_1 \cdot e^{-q\phi_s/kT} \cdot e^{qV_s/kT} \qquad N_d = N_1 \cdot e^{-\phi_s/kT} \cdot e^{qV_d/kT}$$

definiendo $N_1 = N_0 \cdot e^{-\Psi_0/kT}$

Conociendo que la corriente por difusión puede escribirse como

$$I = -wqD\frac{\partial N}{\partial x} = -wqD\frac{N_d - N_s}{L} \qquad \text{(B.13)}$$

reemplazando los valores de N_d y N_s y agrupando las constantes en I_0

$$I = I_0 e^{-q\kappa V_{gs}/kT} \cdot \left(1 - e^{qV_{ds}/kT}\right) \qquad \text{(B.14)}$$

B.5 CONDUCTANCIA DE DRENADOR Y EFECTO EARLY (Early, 1952) (Tsividis, 1987) (Mead, 1989)

En el cálculo de la corriente (B.13) se utilizó una longitud de canal constante L. En realidad la longitud del canal depende de las zonas de deplesión alrededor de la fuente y el drenador. En un MOS-p, el potencial de drenador es mayor que el de fuente, en consecuencia, la zona de deplesión es también mayor cercana al drenador. Además las condiciones de carga en el drenador no son suficientes para el mantenimiento del canal en las vecindades de esta juntura. El canal decrece en longitud, incrementando el gradiente de concentración de

portadores y también la corriente. Este efecto, llamado Early, puede ser visto como una conductancia g_d distinta de cero en la zona de saturación del transistor.

$$g_d = \frac{\partial I}{\partial V_d} = \frac{\partial I}{\partial x}. = -\frac{I}{x}.\frac{\partial x}{\partial V_d} \equiv \frac{I}{V_0} \qquad \text{(B.15)}$$

A la expresión de la corriente dada en (B.14) puede agregarse el efecto de la conductancia considerando la aproximación de (B.15)

$$I' = I + g_d . V_{ds} \qquad \text{(B.16)}$$

B.6 EFECTO DEL SUSTRATO ("Body effect") (Jacoboni, 1977) (Tsividis, 1987) (Mead, 1989)

El efecto del substrato se manifiesta por la diferencia de eficiencia de la compuerta en el control de la carga en el canal. Supongamos que se desea mantener constante el flujo de corriente. Si se incrementa el potencial de fuente V_S respecto del substrato V_B, el potencial de compuerta V_G debe aumentar para aumentar la cantidad de carga Q_B en la zona de depleción. La relación entre ambos potenciales se desprende de (B.x)

$$V_G = V_s + \Delta\phi + V_{FB} + \gamma\sqrt{V_s + \Delta\phi} \qquad \gamma = \sqrt{-2\rho\,\varepsilon_s}/C_{ox} \qquad \text{(B.17)}$$

γ es función de la densidad de dopaje.

APENDICE C

DISPOSITIVO NEURONAL ETANN

C.1 DISPOSITIVO NEURONAL PARA EL PROCESAMIENTO DE PATRONES

El dispositivo neuronal Etann (Electrically Trainable Analog Neural Network), fabricado por la división de investigaciones de la corporación Intel, es, hasta la fecha, la implementación más avanzada realizada en VLSI orientada a redes neuronales. Este dispositivo, fabricado con tecnología CMOS de 1µ, integra 128 neuronas analógicas en dos capas de procesamiento en cascada. En el modo de procesamiento paralelo (PDP) cada neurona en cada capa computa en forma analógica la función

$$O_i = sigmoide\left(\sum_{j=1}^{64}(W_{i,j}.I_j)+\theta_i\right) \qquad (C.1)$$

donde **Oi** es una de las salidas analógicas de la red, **Ij** son las entradas analógicas, **Wij** son los coeficientes de peso internos, y **θi** los valores de umbral. La dimensión máxima del vector de entrada es 128 y la del vector de salida 64. Los 8192 coeficientes **Wij**, distribuidos en dos matrices de 64x64, y los 2048 umbrales **θi** son reprogramables eléctricamente (Apéndice A).

La capacidad máxima de procesamiento del Etann, para un vector de entrada de dimensión 128 y de salida 64 es de unos 8K multiplicaciones y sumas en 1µ seg; es decir, más de 8 exp(9) multiplicaciones y sumas por segundo, con una precisión equivalente a 7 bits. Si bien la precisión es baja, es suficiente para la mayoría de las aplicaciones de redes neuronales, donde lo que cuenta, es la computación masiva, tolerante a fallos, en los cuales, inclusive, algún porcentaje de neuronas fuera de sus valores de tolerancia son aceptables debido a la redundancia de la red. Muchos modelos neuronales utilizan, incluso, salidas binarizadas. La capacidad de procesamiento del Etann es

equivalente a unos 800 procesadores digitales trabajando a 10 MIPS, considerando que estos procesadores pueden ser acoplados en forma perfecta.

El Etann posee una función de activación continua de tipo sigmoidea que puede ser modificada a una función casi binaria usando un modo de alta ganancia en el amplificador de salida. Este modo permite construir redes con salida binaria, por ejemplo, tipo Hopfield, con la ayuda de capacitores externos.

C.2 CONFIGURACION INTERNA Y TEORÍA DE FUNCIONAMIENTO DEL ETANN

La figura C.1 representa el diagrama en bloques del Etann. El dispositivo es fundamentalmente analógico, pero posee electrónica de control digital para la selección individual de sinapses, umbrales, y neuronas; y para el control de los diferentes modos de funcionamiento. Las señales lógicas externas para decodificación son compatibles TTL

En la figura se destacan dos arreglos matriciales de 64x64 que representan los coeficientes reprogramables **Wij** de cada una de las dos capas que son configurables en cascada para construir una red con capa oculta; y otros dos bloques correspondientes a los umbrales **θi** de 16x64 asociados a las matrices antes mencionadas. En la figura C.2 puede verse en detalle la disposición del arreglo matricial respecto de las entradas. Las salidas de los multiplicadores son sumadas y procesadas para conformar el nodo de salida. Las señales de entrada, **Io-I63**, admiten un rango dinámico de 0 a 5volts. Internamente, las entradas serán positivas si superan la señal de referencia **VREFI**, y negativas en caso contrario. El valor de esta referencia es asignado externamente. El rango dinámico de las salidas, **Oo-O63**, es de 0volt al doble del valor de la referencia de salidas, **VREFO**.

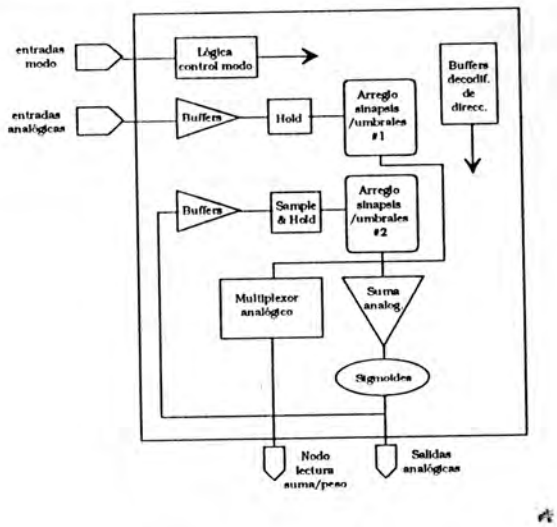

FIGURA C.1 Diagrama en bloques

Puede verse en la figura C.1 que existe una realimentación de las salidas hacia el segundo bloque matricial de pesos. Esto permite implementar una segunda capa de procesamiento dentro del mismo dispositivo. Todas las señales de entrada y realimentación pasan a través de *buffers* separadores y son almacenadas temporariamente (*holds*).

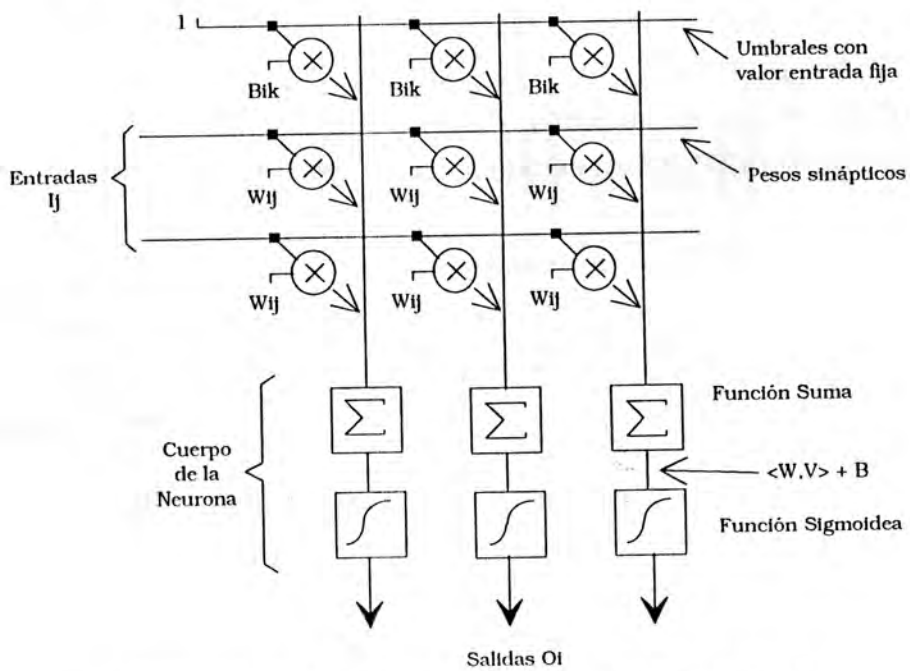

FIGURA C.2 Estructura funcional de una capa

Cada uno de los coeficientes internos del Etann (**W_{ij}** y **θ_i**) puede ser individualmente seleccionado por medio de trece lineas de direccionamiento, **A_0-A_{12}**. La linea **A_{12}** selecciona entre ambos bloques matriciales y de coeficientes umbral. Las lineas **A_0-A_5** seleccionan la columna de la matriz correspondiente, y las lineas **A_6-A_{11}**, cada una de las filas. La señal de control **BIAS**, indica que la selección apunta hacia uno de los bloques de coeficientes umbral.

El control de muestreo y almacenamiento (*sample and hold*) de las señales de entrada y de las que provienen de la realimentación, se realiza a través de las señales de control **HOLD** y **CLK**, respectivamente. Las señales **$RESET_f$** y **$RESET_i$** conectan (o desconectan) fisicamente las salidas de los muestreadores a los bloques de procesamiento, evitando que ambos sean conectados al mismo tiempo.

Cada uno de los coeficientes de peso **W_{ij}** es una tensión diferencial. Cada una de sus partes, denominadas *peso* y *referencia*, puede ser individualmente seleccionada y leída en forma indirecta utilizando la salida **SYNO** (*synapse output*). La señal de control **WT** selecciona entre ambas partes. La señal analógica de salida, suma de los productos entre los **W_{ij}** e **I_j**, puede ser leida antes de ser procesada por la función sigmoidea, en forma de una corriente diferencial, en los nodos **SMO+** y **SMO-**. Las señales de control que habilitan esta lectura son **QUERY** en estado activo y **LRN** (*learn*) en estado inactivo.

Como ya se mencionó anteriormente, los coeficientes **W_{ij}** y **θ_i** son re-programables eléctricamente. La modificación de estos coeficientes se hace de la misma forma que en las celdas de memoria EEPROM, utilizando el efecto Fowler-Norldheim (Apéndice A). Las entradas de señal **V_{PP1}** y **V_{PP2}** reciben los pulsos de alta tensión para tal efecto.

C.3. ARQUITECTURA DE UNA SINAPSIS

Como se dijo, las sinapses son circuitos analógicos que permiten la multiplicación de una señal de entrada por un coeficiente previamente

almacenado. En el modelo de la ecuación 4.1, representan a cada uno de los productos dentro de la sumatoria. La figura 4.3 representa el esquema de la sinapse. Puede verse que el mismo está compuesto por un arreglo de dos amplificadores de transconductancia cuyas entradas y salidas se encuentran en anti-paralelo, formando un multiplicador de cuatro cuadrantes como el analizado en la sección 3.4.1 del capítulo 3 (Mead, 1989). La corriente diferencial de salida **ΔIo** es proporcional al producto entre **ΔVIN** y **ΔVWT**. Luego la suma de todas las corrientes diferenciales correspondientes a cada entrada **Ij** y a cada peso **Wij** se realiza por Kirchoff en el nodo sumador.

$$\Delta I_{out} = I_b . tgh \frac{\Delta V_{in}}{2V_t} . tgh \frac{\Delta V_{WT}}{2V_t} \qquad (C.2)$$

La corriente **ΔIo** se obtiene como una suma algebraica de las corrientes de rama,

$$\Delta I_{out} = I^+ - I^- = I_{13} + I_{24} - I_{14} - I_{23} \qquad (4.3)$$

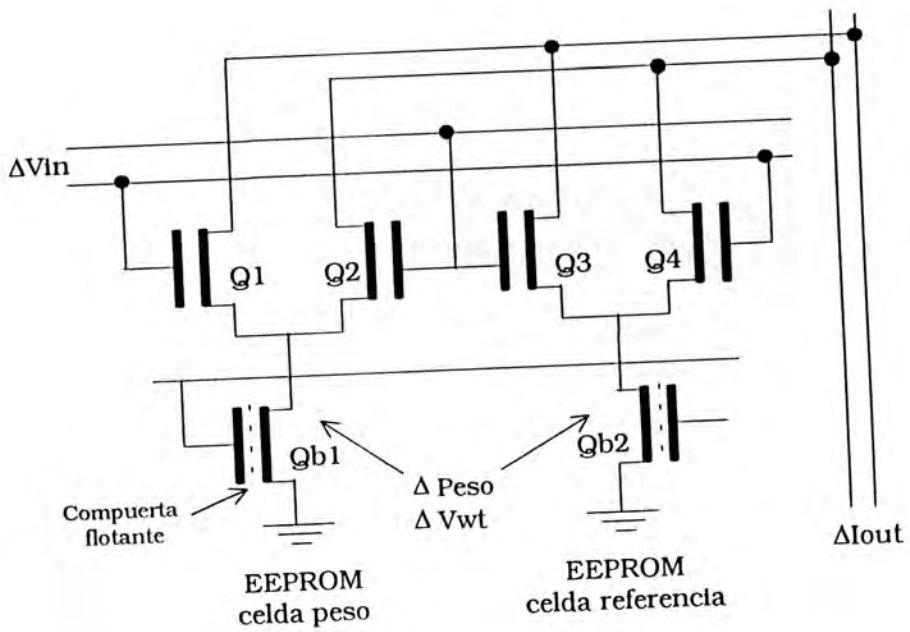

FIGURA 4.3 ESQUEMA DE LA SINAPSE

Los transistores Qb1 y Qb2 de las fuentes de corriente de los amplificadores diferenciales, son dispositivos que poseen una compuerta flotante. Dicha compuerta de silicio poli-saturado forma parte del transistor MOS y es inaccesible desde sus contactos externos. Si bien la compuerta accesible de estos transistores está fija a un potencial interno, la corriente de drenaje (Id_{Qb1} e Id_{Qb2}) de las fuentes de corriente, y por lo tanto la transconductancia de los amplificadores diferenciales, puede ser modificada por medio de la compuerta flotante. El apéndice A provee una explicación detallada del proceso túnel Fowler-Nordheim y del funcionamiento de la compuerta flotante en transistores MOS. El efecto de una inyección de carga a la compuerta flotante es macroscópicamente observado en un corrimiento de la característica de entrada del MOS (fig. 4.4). La curva se mueve de manera que se modifica el valor de tensión umbral **VTO**. En la sección 4.6.3 se trata la modificación de . Debido a que el circuito de la sinapse es doble en las fuentes de corriente, se consigue corrimientos diferenciales en **VTO** y por ende, la multiplicación en los cuatro cuadrantes.

La tensión umbral intrínseca es de 0.5volt. En consecuencia para aprovechar el mayor rango de linealidad del circuito, un coeficiente de peso **Wij** debe ser almacenado modificando simétricamente **VTOwt** y **VTOref** alrededor de 0.5v.

$$\mathbf{V_{TOwt} = W_{ij}/2 + 0.5}$$
$$\mathbf{V_{TOref} = W_{ij}/2 - 0.5} \qquad (4.4)$$

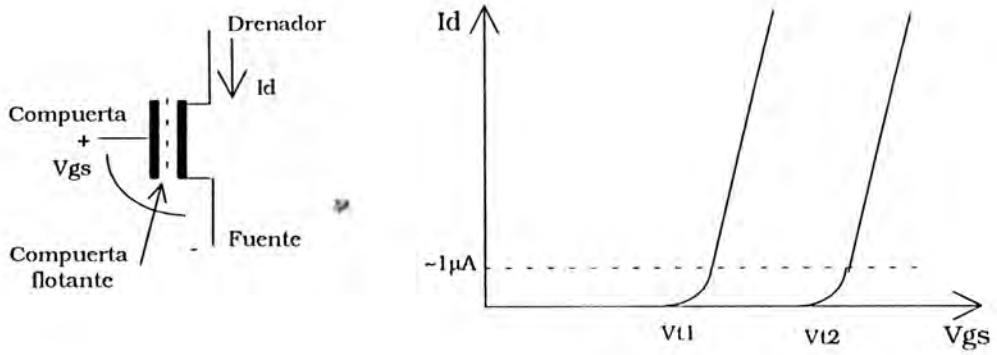

FIGURA 4.4 Diagrama de la Compuerta flotante y caract. Id-Vgs

La figura 4.5 muestra las características **ΔIo-ΔVIN** del multiplicador para valores diferentes de **ΔVWT**. Como puede verse en la figura 4.1, cada entrada está conectada a 64 sinapses que llegan a 64 neuronas diferentes. Si nuestra arquitectura es tal que no todas las entradas utilizadas deben conectarse a todas las neuronas de nuestra red, los coeficientes de las sinapses de dichas entradas deben ser puestas a cero. Debido a que el método de programación de los coeficientes es un proceso complejo, la forma más eficiente es saturar los valores de **VTOwt** y **VTOref** a sus valores máximos logrando un **ΔVWT = 0**. Como se verá en la sec. 4.6.3 y capítulos subsiguientes, esta forma asegura que se hace en el mínimo tiempo y que se reduce el consumo de potencia del Etann debido a que los transistores de la sinapse quedan polarizados en la zona de corte.

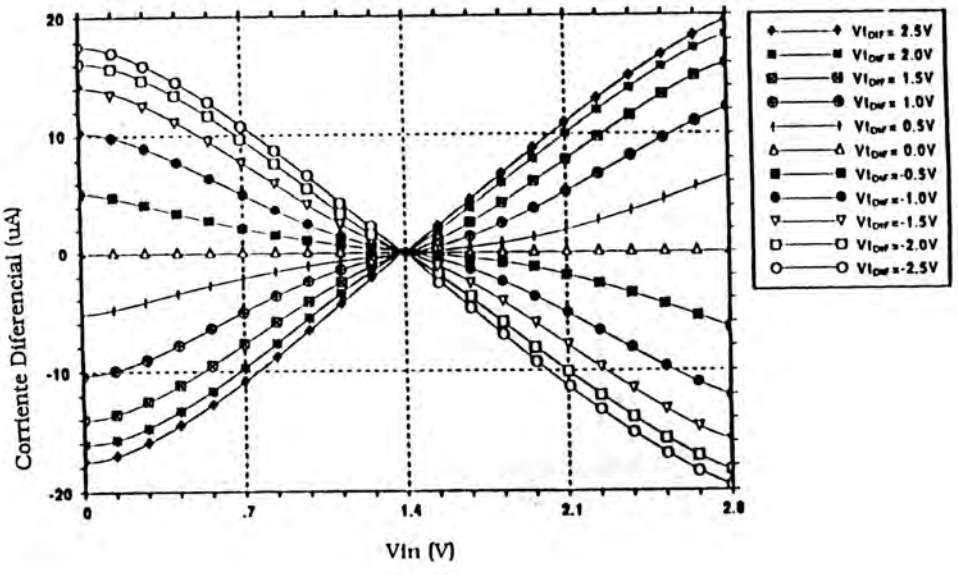

FIGURA 4.5 Características del multiplicador

C.4 RESPUESTA SIGMOIDEA

Las 160 corrientes diferenciales a lo largo de cada columna de ambas matrices de procesamiento son sumadas para conformar la salida de la neurona. Un

amplificador de transimpedancia convierte esta suma de corrientes diferenciales en un valor de tensión (no diferencial). La función de transferencia de este amplificador posee la no linealidad de tipo sigmoidea analizada en la sec. 3.4.5.1. El amplificador posee un control de ganancia (VGAIN) que permite modificar la pendiente de esta característica. Este control sobre la realimentación del amplificador funciona a modo de control del parámetro de *temperatura* (Rumelhart,1986) de la función sigmoide representada por la ecuación (3.12). La figura 4.6 muestra las características de transferencia del nodo sumador para distintos valores de **VGAIN**. Este control es único para los 64 nodos de salida, lo cual es razonable para la mayoría de las aplicaciones en las cuales todas las neuronas tienen la misma función de activación. Sin embargo, en una implementación VLSI de este tipo, las discrepancias entre los transistores de los amplificadores sumadores de cada neurona generan tensiones de offset de entrada. Estas diferencias de tensiones de offset individuales pueden ser corregidas usando uno de los coeficientes de umbral θ_i, durante la inicialización del sistema. Utilizando un umbral por neurona, se modifica su coeficiente de peso hasta que la salida O_i sea cero para tensiones cero de entrada I_j.

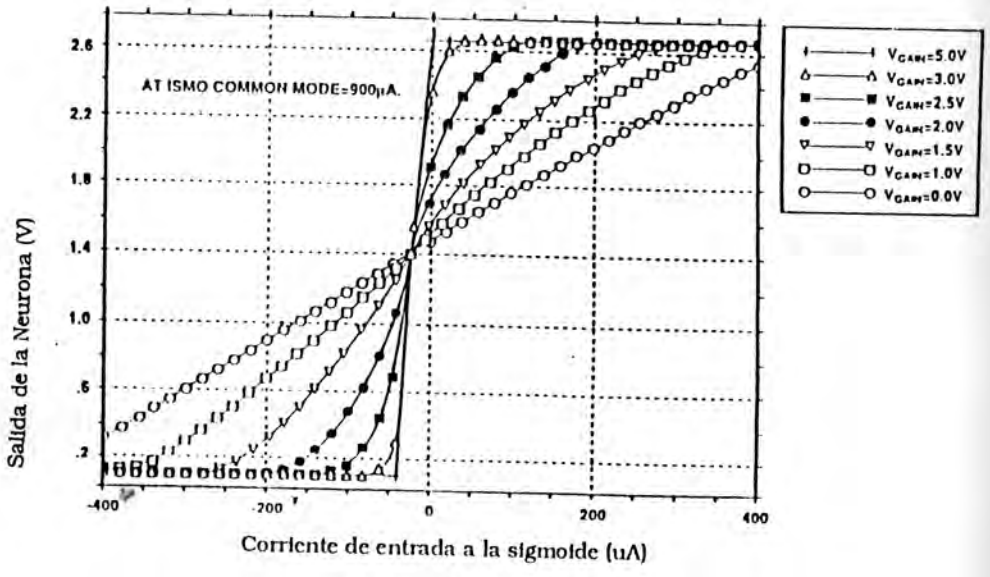

FIGURA 4.6 Características de la sigmoide

El rango dinámico de la salida de la neurona puede ser modificado, a través de la referencia **VREFo**, entre 1 y 4 volts. La tensión de referencia indica el punto medio de este rango dinámico. La figura 4.7 muestra la tensión de salida de la sigmoide en función de la corriente de entrada, tomando **VREFo** ,como parámetro.

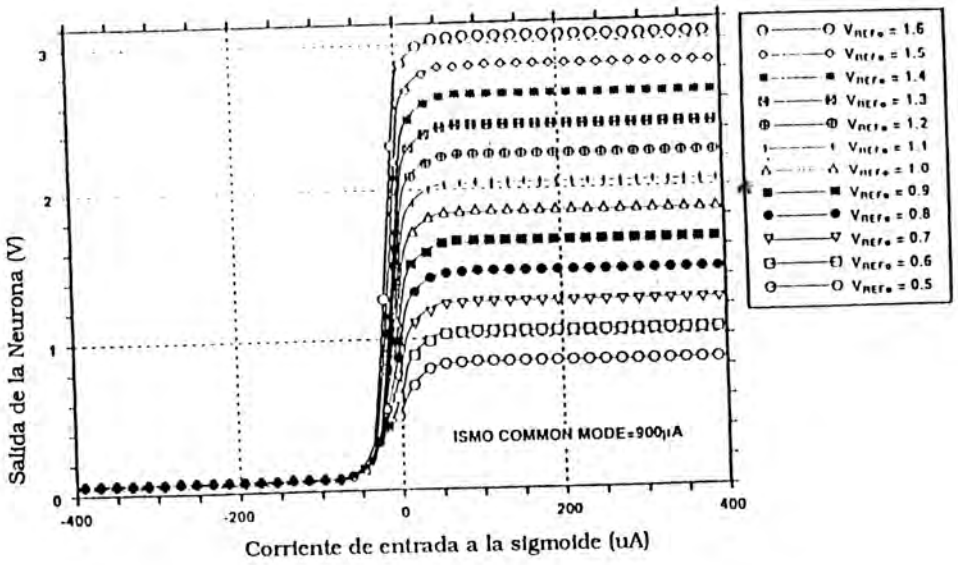

FIGURA 4.7 Características de la sigmoide

C.5 OPERACIÓN DEL ETANN EN MODO DE PROCESAMIENTO PARALELO

El modo principal de funcionamiento del Etann es el modo de procesamiento paralelo (PDP) en el cual el dispositivo computa los patrones de entrada según la función (4.1). Si se utiliza la realimentación interna de las salidas (fig. 4.1), puede implementarse una red neuronal tipo Perceptrón multicapa con una capa oculta usando un solo dispositivo. El tiempo de procesamiento es de 3µ seg. por capa. La figura 4.8 muestra el ciclo de procesamiento de patrones dentro del Etann.

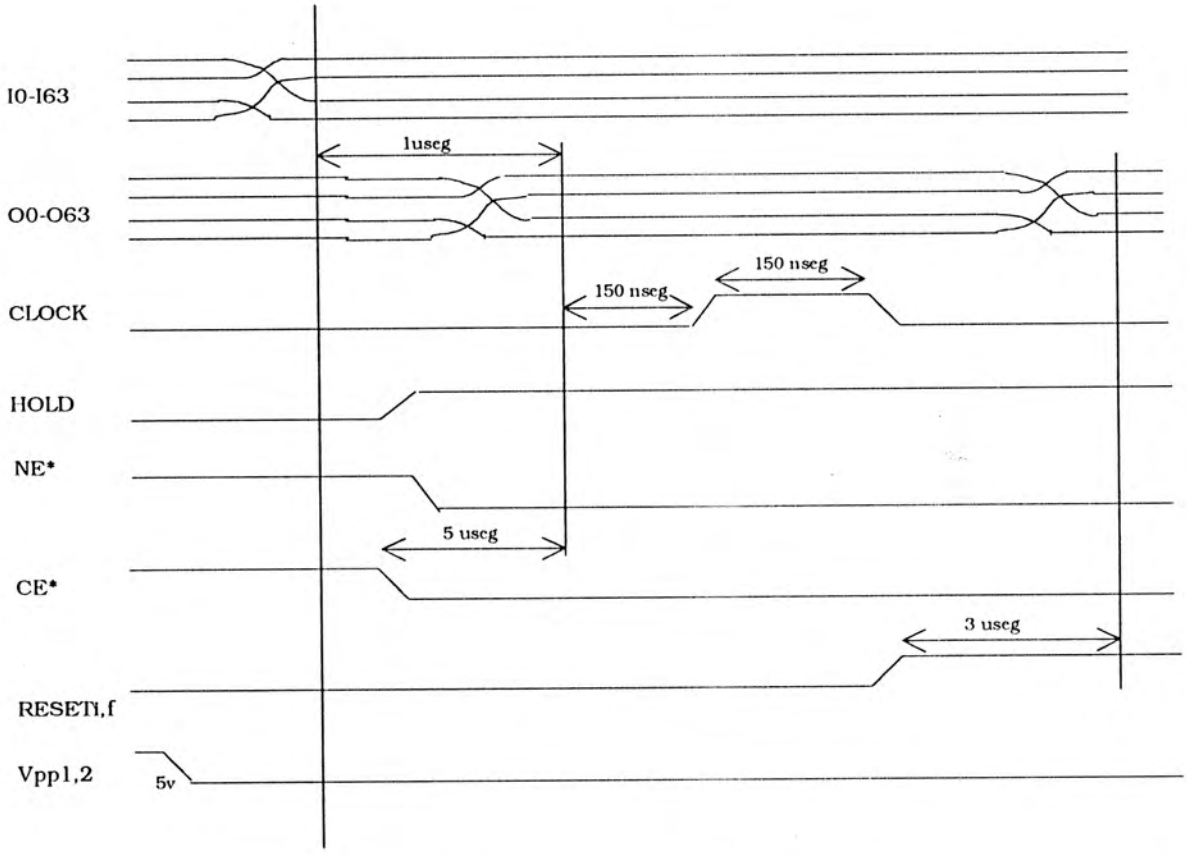

FIGURA 4.8 ciclo de temporización en modo PDP

APENDICE D

CIRCUITOS DEL SISTEMA NETMAP

D.1 PLACA DSPCARD

La placa **DSPcard** actúa como una placa aceleradora que concentra las funciones de procesamiento relacionadas con el Etann descritas en sección 5.2 (Cancelo, 1995). La figura 5.1 representa el diagrama en bloques de la placa.

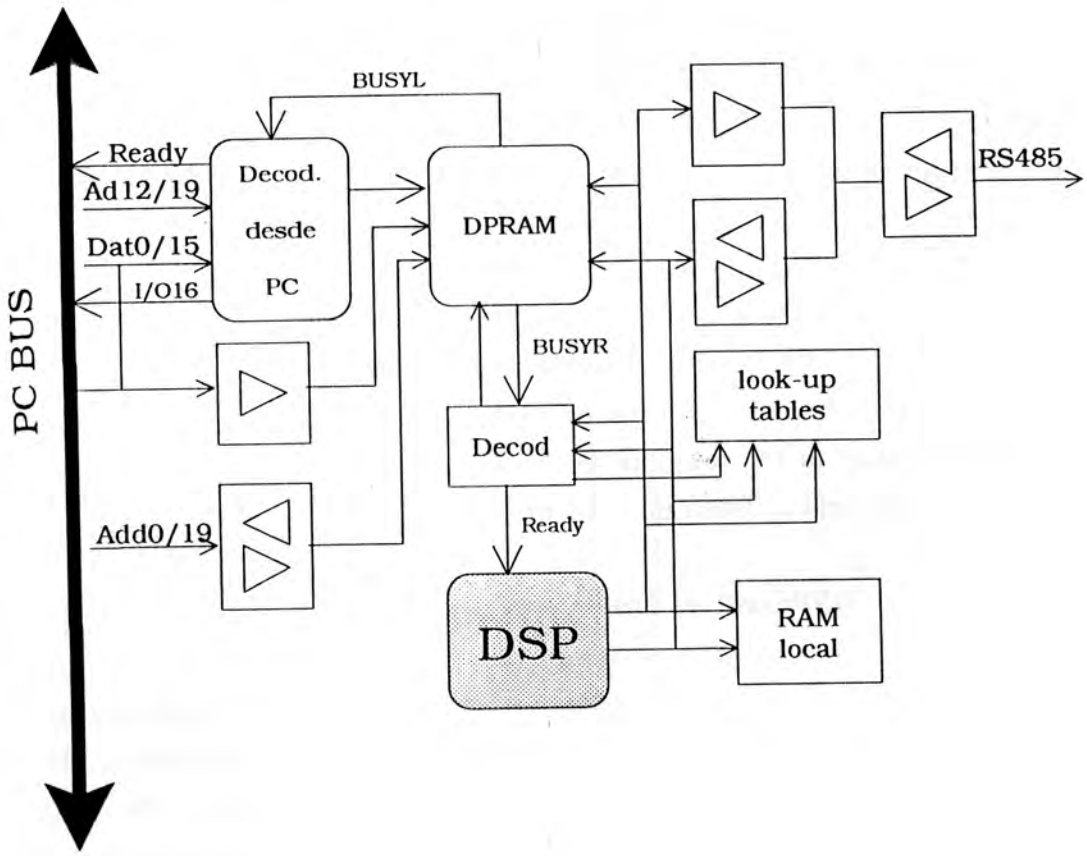

FIGURA D.1 Diagrama en bloques de la placa DSPcard

El procesador digital de señales es un TMS320C25 (Texas Instr., 1990) de Texas Instrument, trabajando a 10 MIPS. Este procesador se comunica con el

bus de la computadora personal a través de una memoria de doble puerta de 8Kbytes. El DSP actúa como maestro de la placa. Internamente, posee una arquitectura Harvard de 16 bits, 8 Kbytes de memoria Eprom, y 1 Kbyte de memoria RAM, ambas organizadas en palabras (16 bits). La unidad aritmético-lógica permite realizar una multiplicación y una acumulación en un ciclo de reloj (100 nseg.), además de una operación de registro en paralelo. Para evitar problemas de inicialización del sistema, los programas del DSP fueron optimizados en tamaño para que pudieran ser incluidos completamente dentro de los 8K de Eprom.

La placa **DSPcard** posee una interfaz de 16 bits con la computadora personal. La decodificación de la placa, vista desde la PC, es un mapa de 8Kbytes (*0 wait state*) re-ubicables, por medio de llaves, en cualquier posición de la zona alta de la memoria (HMA). La lógica de control de decodificación de las direcciones de la PC maneja la linea de READY (Eggbrecht, 1991) cuando hay accesos simultáneos a la misma posición en la memoria compartida. En caso que haya un acceso de escritura simultaneo a la misma posición de la memoria compartida, la memoria genera una linea BUSY que es interceptada por la lógica de control de la placa, que la sincroniza con los tiempos de ciclo de la PC (fig. 5.1). En este ceso pueden generarse ciclos de espera para ambos procesadores hasta que el procesador que posee la memoria, la libere. Los accesos desde la PC pueden ser efectuados en 8 y en 16 bits, la lógica de la **DSPcard** maneja la linea de control I/O CS16 (fig. 5.1).

En la placa **DSPcard**, el DSP tiene un *bus* local con recursos privados. La memoria RAM local del sistema (32 KBytes) se utiliza como memoria de almacenamiento temporaria de los patrones de entrada/salida y como memoria borrador de las variables del sistema. La memoria Eprom externa de 64 Kbytes es usada para almacenar las tablas de las curvas de V_{PP2} y T_P utilizadas en el cálculo de los pulsos de programación de los pesos. Estas tablas son generadas *off-line* en los programas de simulación que se describirán en el capitulo 6, y aceleran el cómputo de los pulsos de programación.

El DSP tiene su mapa de memorias dividido en tres secciones: **PS**, memoria de programa, **DS**, memoria de datos, e **IS**, memoria de entrada/salida. Cada

sección direcciona 64Kwords. La figura D.4a, D.4b, y D.4c muestran la disposición de los recursos locales y compartidos del DSP en los mapas de memoria respectivos.

Para la decodificación de estos mapas de memoria (Cancelo, 1994) se utilizaron dispositivos lógicos programables (GAL22V10) de 10nseg. La decodificación de las señales provenientes de la PC se hacen con una EP910 de 35nseg, dado que se necesita mayor cantidad de lineas de entrada y no es tan exigente el requerimiento de tiempo.

La conexión de la placa **DSPcard** con la **PBox** se efectuó por medio de una interfaz que usa la codificación eléctrica RS485 con direcciones, datos y lineas de control en paralelo. Se eligió la señalización RS485 por sus buenas características de velocidad de transmisión e inmunidad al ruido (National, 1992). Las direcciones y datos del DSP son multiplexados antes de ser convertidas al protocolo RS485 para disminuir el número de lineas a casi la mitad. La figura D.2 muestra los terminales de transmisión y recepción de ambas placas que realizan la función de multiplexado a la interfaz.

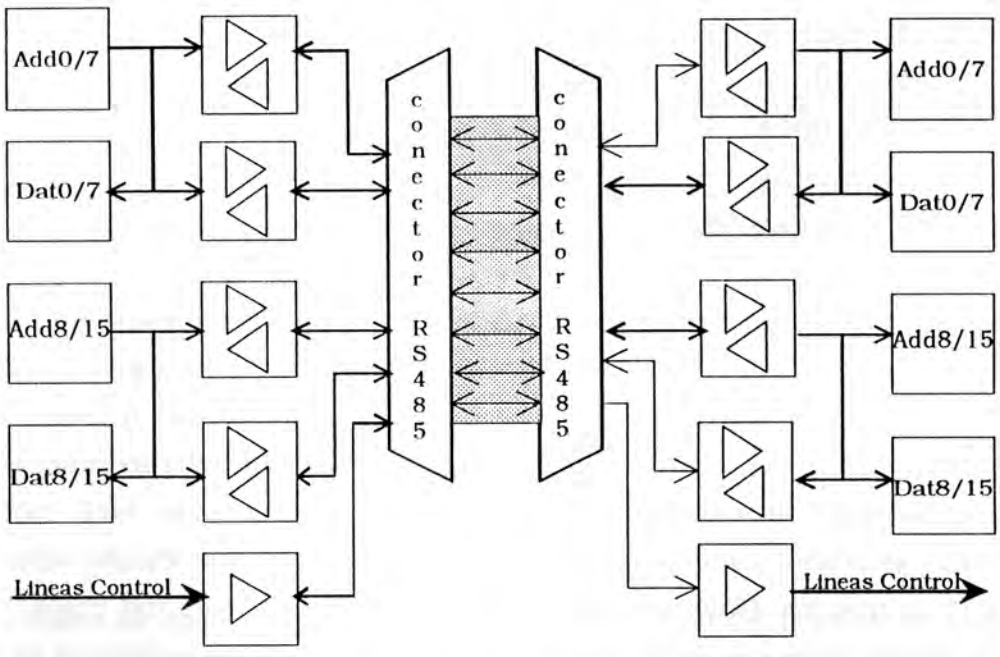

FIGURA D.2 Interfaz RS485

La figura 5.3 muestra el ciclo de temporización de lectura/escritura a través de la interfaz RS-485.

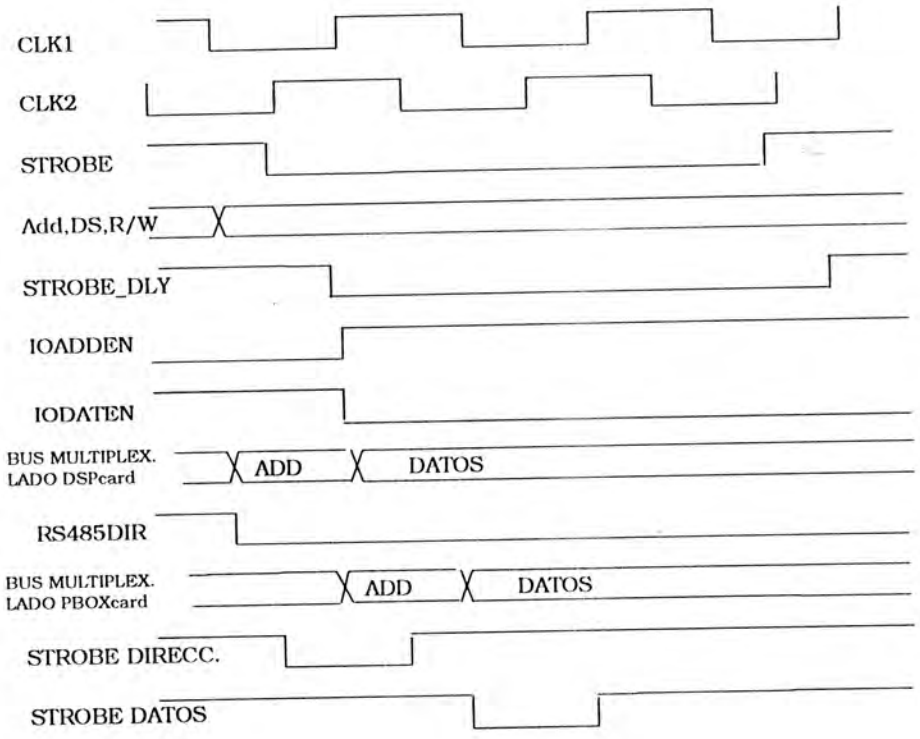

FIGURA D.3 Diagrama de tiempos de la interfaz RS-485 en la placa DSPcard

D.2 MAPAS DE MEMORIA

Los tres mapas de memoria del DSP están destinados a: memoria de programa PS, memoria de datos, DS, y memoria de entrada/salida, IS. Cada uno de ellos permite direccionar 64 Kword. El DSP puede ser configurado a través de una linea de control, en modo microprocesador o modo microcomputador. El sistema Netmap lo configura como microcomputador, lo cual ubica las memorias internas como parte de los respectivos mapas. Puede verse en la figura D.4a que los 4Kword de Eprom ocupan la parte baja del mapa PS. Los 32Kword mas altos son ocupados por las tablas de valores asociados al cálculo de **VPP2** y **TP**.

	Tipo	Dirección
EPROM local	interna	0000-07FF
tablas look-up	externa	8000-FFFF

FIGURAS D.4a Mapa de memoria PS

	Tipo	Dirección
RAM_DSP	interna	0000-0FFF
RAM baja	privada	2000-3FFF
RAM alta	privada	4000-5FFF
DPRAM baja	compartida	6000-7FFF
DPRAM alta	compartida	8000-9FFF

FIGURAS D.4b Mapa de memoria DS

	Tipo	Dirección
UARTS	selección canal RS232	0000-001F
DAC1S	conversor D/A	A000-A0FF
DAC2S	conversor D/A	A100-A1FF
VPPS	conversor D/A	A200-A2FF
VGAINS	conversor D/A	A300-A3FF
ETANNADDS	latch direcciones	A500-A5FF
STATS	registro estados	B400-B4FF
PLOADS	contador carga paralelo	B000-B0FF
FIFORS	memoria Fifo	B100-B3FF

FIGURAS D.4c Mapa de memoria IS

La memoria de datos (fig. D.4b) ubica la RAM interna del DSP en la parte baja, la memoria externa local en 2000H y la memoria compartida en 8000H.

La placa **PBox** esta ubicada completamente en la zona de entrada/salida, IS. La figura D.4c enumera los dispositivos que son decodificados en esta área de memoria. Además, se ha utilizado la zona de memoria IS entre A900H y AFFFH para un conjunto de comandos de la lógica asociada a la placa **PBox**. Estos comandos se encargan del manejo de los modos del Etann, así como de la inicialización del microcontrolador *bit-slice* (Cancelo, 1993, manual).

D.3 MEMORIA COMPARTIDA

La zona de memoria compartida (fig. D.4b) está subdividida en forma lógica en las zonas indicadas en la figura D.5. Esta memoria tiene el propósito de comunicar al DSP y al procesador de la computadora personal. La comunicación se realiza a través de comandos desde la PC al DSP (*command driven*). Los comandos editados por el procesador PC son contestados al final de su realización por el DSP en área de estados. Cada comando puede pasar parámetros en el área destinada a tal efecto y recibir resultados por parte del DSP. La zona llamada de *archivo de entrada* corresponde a los patrones desde la PC hacia el DSP para su procesamiento por el Etann y la llamada *archivo de salida*, a los patrones de salida del Etann que deben ser leidos por la computadora. El tamaño de estas zonas permite almacenar simultaneamente 64 patrones de tamaño máximo en cada una.

DPRAM	Direcciones
COMMANDOS	6000-6001
ESTADOS	6002-6003
AREA DE PARAMETROS	6004-600F
AREA DE RESULTADOS	6010-601F
AREA DE ARCHIVO DE ENTRADA	6020-7FFF
AREA DE ARCHIVO DE SALIDA	8000-9FFF

FIGURAS D.5 Mapa de memoria DPRAM

D.4 MAPA DE DISPOSITIVOS DE ENTRADA/SALIDA

La figura D.4c indica los dispositivos de la placa **PBox** que pueden ser direccionados para lectura o escritura desde el procesador DSP. DAC1S y DAC2S corresponden a las lineas de selección de los conversores digital a analógico. Estos conversores de 8 bits están dispuestos en pares para utilizar los 16 bits del *bus* del DSP. En ellos son almacenados los datos de los patrones de entrada al Etann.

VPPS es la linea de selección de los conversores digital a analógico de los generadores de pulsos de alta tensión de programación durante la modificación de los pesos en el algoritmo de entrenamiento de la red neuronal.

VGAINS selecciona el conversor analógico-digital que controla la ganancia de la sigmoide de salida de la red neuronal. El rango dinámico de la salida del conversor es de 0 a 5 volts, permitiendo hacer funcionar la red en modo de alta ganancia y obtener salidas lógicas booleanas para redes que así lo requieran.

ETNADDS selecciona un registro en el cual se guarda la dirección de la sinapse, nodo suma o neurona sobre la que se realizará una operación de lectura o escritura.

STATS permite leer un registro en el cual se refleja el estado de distintos componentes de la placa **PBox**.

PLOADS selecciona un registro en el cual se indica cual es la primer neurona del Etann que debe leerse a la salida de procesamiento de patrones.

FIFORDS selecciona la memoria de almacenamiento temporario de los patrones procesados por el Etann.

Las figuras D.6 y D.7 muestran el esquema completo y la foto de la placa **DSPcard**.

D.5 PLACA PBOX

La placa **PBox** posee unos 60 circuitos integrados que están asociados a las funciones de conversión analógico-digital y control de las señales que llegan a la red neuronal. Posee, además, una memoria de almacenamiento temporario de patrones procesados y un microcontrolador que se encarga de la conversión analógico-digital. Los bloques funcionales de esta placa son

- Interfaz RS485
- Lógica de decodificación
- Convertidor DC/DC
- Convertidores digital-analógico para pulsos de alta tensión
- Circuito de conversión de los patrones de entrada
- Control de ganancia de la función sigmoide
- Circuito de lectura de coeficientes de peso
- Conversión analógico-digital y memoria de almacenamiento temporario
- Circuito de lectura de nodo suma o salida de neurona

La figura D.8 representa un esquema de bloques de las funciones de la placa **PBox**.

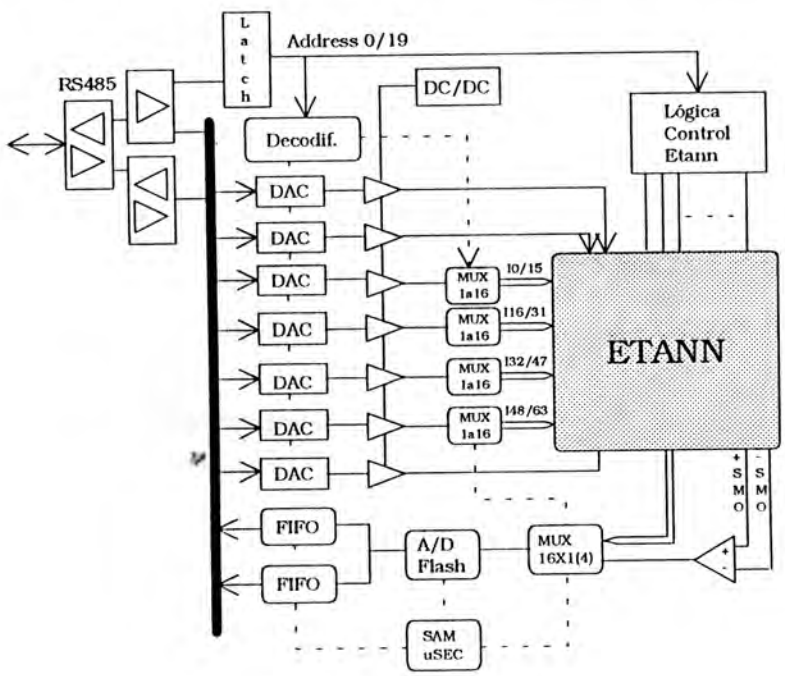

FIGURA D.8 diagrama en bloques PBox

La interfaz RS485 permite que la placa **PBox** sea conectada a la **DSPcard** a través de una conexión de alta velocidad. Esta interfaz multiplexa las lineas de datos y direcciones del microprocesador y recibe, además, señalización de control de la decodificación de la placa **DSPcard**.

La lógica de decodificación sobre la placa **PBox** está destinada a la selección de los conversores digital-analógico **DAC1S**, **DAC2S**, **VGAIN**, **VPPS**, a la lectura de la memoria de almacenamiento temporario **FIFORS**, al registro de selección de direcciones del Etann **ETNADDS**, al registro de estados **STATS** y genera la temporización de las señales necesarias para los ciclos de lectura/escritura. Estos dispositivos se encuentran ubicados en el mapa de entrada/salida **IS**.

D.5.1 DOBLADOR DE TENSIÓN

El módulo convertidor DC/DC esta basado en el regulador por modulación de ancho de pulso LM3524 (National, 1982). Este circuito se lo utiliza como doblador de tensión para obtener +24 volts de una fuente de +12 volts, para ser utilizados en los amplificadores del circuito de programación de pesos de la red neuronal. La figura D.9 muestra en detalle el circuito utilizado

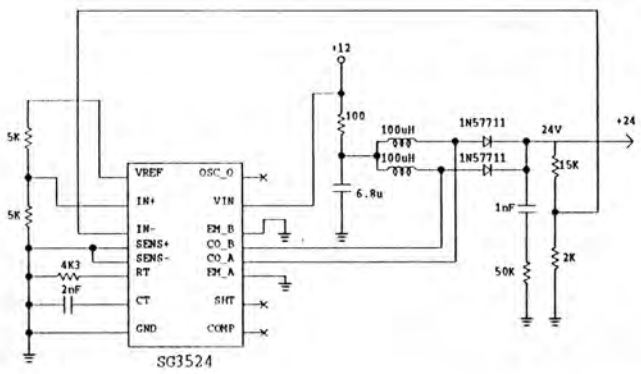

FIGURA D.9 Circuito DC/DC

D.5.2 CIRCUITO DE PROGRAMACIÓN DE COEFICIENTES DE PESO

El circuito de programación de los pesos de la red neuronal (fig. D.10) actúa enviando carga por efecto túnel a la compuerta flotante de dos transistores MOS de las fuente de corriente del multiplicador analógico de la sinapse. Esta carga se especifica por pulsos de altura **VPP2** y ancho **TP**. Para que exista efecto túnel (Apéndice A) los pulsos deben tener al menos 14 volts y no deben superar la tensión de ruptura de la juntura del transistor, unos 28 volts. El Etann requiera dos señales de alta tensión en forma pulsante **VPP1** y **VPP2**. El circuito de la figura D.10 genera estas señales utilizando dos conversores digital-analógico y un par de amplificadores operacionales. El rango dinámico de salida del conjunto conversor-amplificador es de 12 volts, así que se utiliza un segundo amplificador de ganancia 2 para elevar el rango a 24 volts. La figura D.11 muestra el diagrama de tiempos de un ciclo de programación. La señal **VPP1** se mantiene constante en 18 volts, la señal **VPP2** varía según el valor calculado en la ecuación (4.8). Las señales **VPP1** y **VPP2** son utilizadas, también, durante otros modos de funcionamiento del Etann, en procesamiento paralelo son mantenidas fijas a +5 volts, y durante la lectura de pesos **VPP1** genera la rampa, mientras que **VPP2** permanece constante a +2 volts.

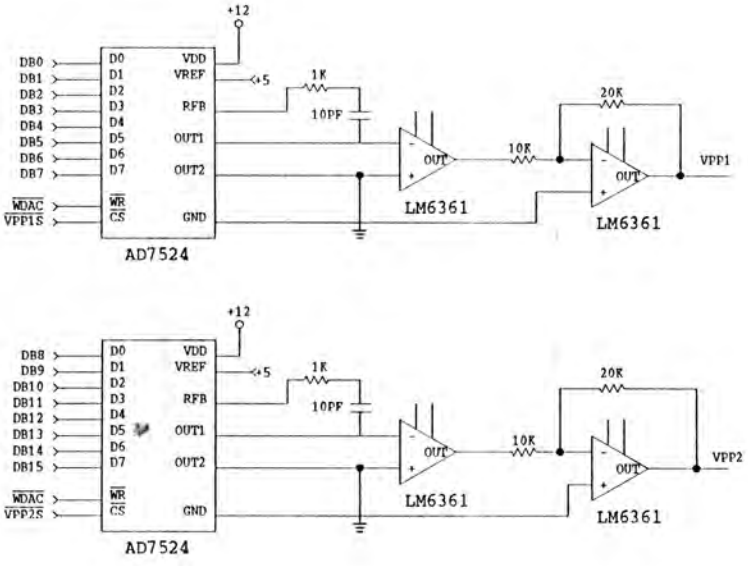

FIGURA D.10 circuito Vpp

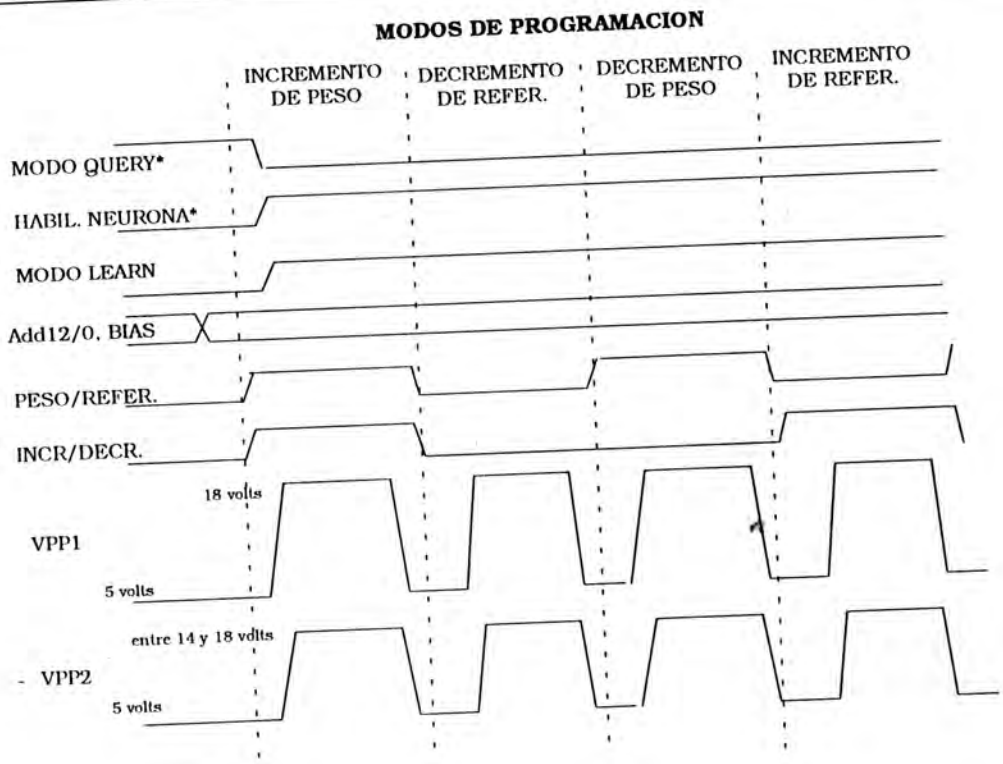

FIGURA D.11 Temporización del ciclo de programación de pesos

D.5.3 CIRCUITO DE CONVERSIÓN DE PATRONES DE ENTRADA

Los patrones generados en la computadora pasan al sistema Netmap a través de la memoria compartida. Los patrones digitales son convertidos a analógicos previamente a ser procesados por el Etann. Para ello se dispone un conjunto de cuatro conversores digital-analógico y cuatro multiplexores analógicos. Los conversores están dispuestos en dos pares; cada par ocupa la zona baja y alta del bus de datos del DSP. De esta forma se optimiza la escritura en los DACs, un par de DACs se escribe durante el tiempo de establecimiento del otro par (fig. D.12). Los multiplexers, manejados por la lógica de decodificación, conducen la señal de los conversores a cada una de las 64 entradas analógicas del Etann. Cada salida de multiplexer junto con un capacitor de .22nf actúa como circuito de muestra y retención (*sample&hold*).

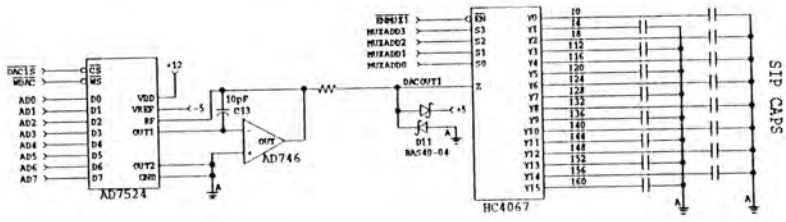

FIGURA D.12 Circ DACs y multiplexers.

D.5.4 CONVERSIÓN DE PATRONES DE SALIDA Y MEMORIA DE ALMACENAMIENTO TEMPORARIO

El circuito de lectura de los patrones procesados por el Etann es controlado por un microcontrolador *bit-slice*. Este controlador trabaja en forma autónoma, independientemente del DSP. De todas maneras, el DSP controla los modos de funcionamiento del sistema en bajo nivel, inicializando el microcontrolador durante el modo de procesamiento paralelo de patrones (PDP). El circuito que lee las 64 salidas del Etann (fig. D.13) consta de un conjunto de multiplexers organizados en dos capas. La primera tiene cuatro que reducen el número de canales de 64 a 4. Luego otro multiplexer selecciona uno de estos cuatro canales hacia la entrada del conversor analógico-digital. La función del microcontrolador es la de manejar las lineas de selección de los multiplexers, las señales de temporización del conversor A/D y el almacenamiento de la salida digital del conversor en un par de memorias tipo FIFO (First in first out) de 512 words cada una. El conversor analógico-digital MP7684 es un conversor de tipo *flash* de 8 bits, trabajando a 10 millones de conversiones por segundo.

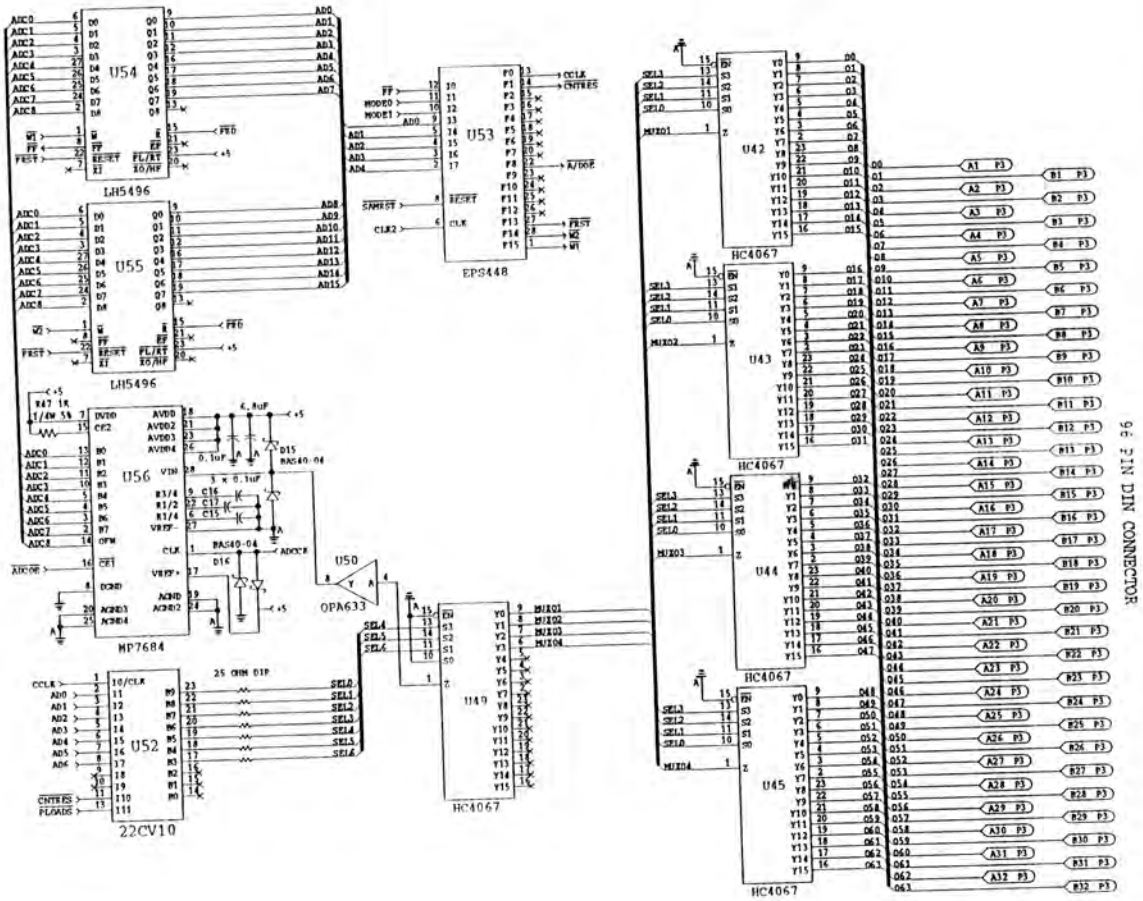

FIGURA D.13 Esquema circuito conv. A/D y microsec.

El microcontrolador posee dos modos de funcionamiento el primer modo activado por un pulso en la linea MODE0 lo programa para leer las 64 salidas del Etann y almacenar los datos en las memorias FIFO. El segundo modo se activa con un pulso en la linea MODE1 y programa al microcontrolador para leer solo un número indicado de lineas de salida, comenzando desde la linea indicada por un registro interno que posee la lógica de selección de multiplexers. El modo 1 evita que redes neuronales con pocas salidas deban leerse las 64 salidas que posee el Etann. La figura 5.14 muestra un diagrama de tiempos del ciclo del microcontrolador.

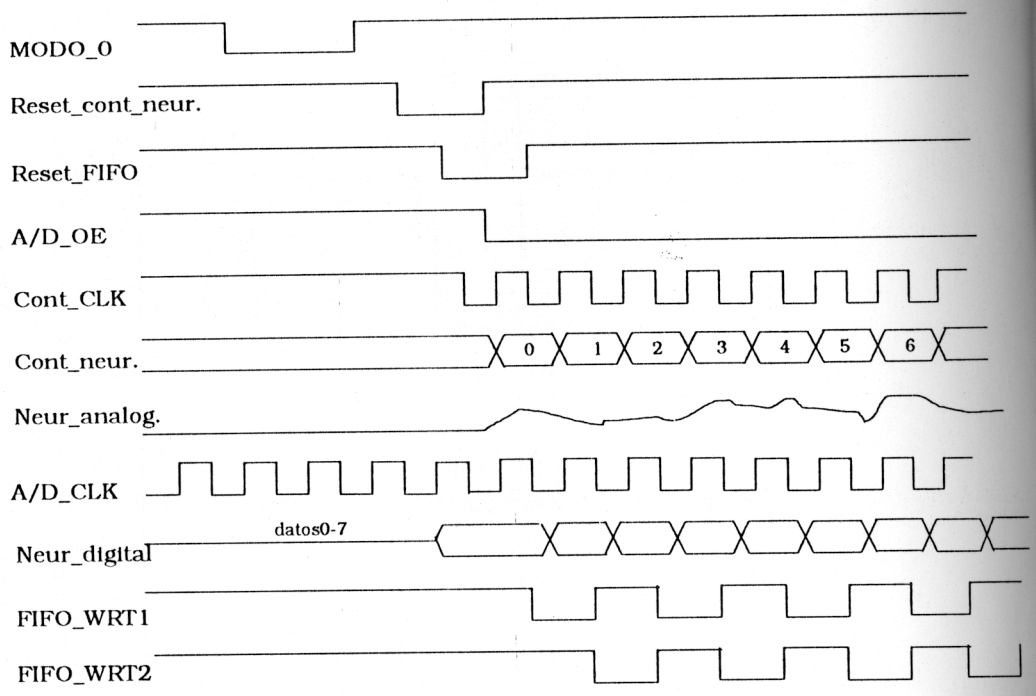

FIGURA D.14 Timing del microcontrolador, A/D y fifos

D.5.5 CIRCUITO DE MEDICIÓN PARA LA LECTURA DE PESOS

Se describió en la sección 5.3 que toda lectura en un peso, debe hacerse en función de ambos las medidas en dos transistores. La tensión de compuerta sigue la ecuación

$$V_{TOwt} = W_{ij}/2 + 0.5$$

$$V_{TOref} = W_{ij}/2 - 0.5$$

Los circuitos de decodificación de direcciones y las señales de control WT y BIAS permiten individualizar cada una de estas celdas. Debido a que el valor del coeficiente de peso depende de la compuerta flotante del MOS de la fuente de corriente, debe hacerse una medición indirecta utilizando el circuito de la figura 5.10 de sección 5.7. El terminal de fuente del MOS de la fuente de corriente es polarizado a 2 volts, y el drenador de uno de los transistores del par diferencial se polariza con 3 volts. El drenador del otro transistor del diferencial es conectado a una rampa ascendente entre 0 y 4 volts, mientras se

mide la corriente de drenador del primer transistor. El circuito implementado (fig. D.15) utiliza un conversor digital-analógico para generar la rampa de tensión en **VPP1** y un circuito de medida formado por un amplificador operacional que convierte la señal de corriente a tensión y un comparador que discrimina el umbral donde la corriente ha comenzado su crecimiento exponencial. El disparo del conversor se refleja en el registro de estados cuyo valor puede leerse desde el DSP para detener la rampa y finalizar la medida.

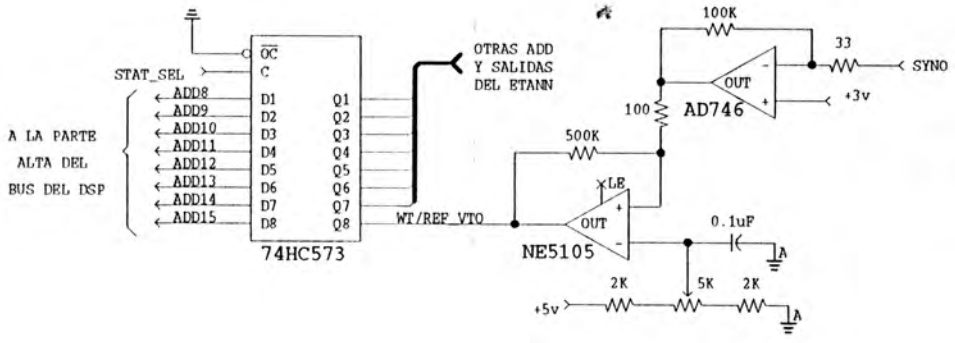

FIGURA D.15 Sistema para medida de VTOwt/ref

Abu-Mostafa, Y. 1988a. *Connectivity versus entropy.* In Anderson, D, Neural Information Processing Systems. New York: American Institute of Physics, p.1.

Abu-Mostafa, Y. 1988b. *Lower bound for connectivity in local-learning neural networks.* Journal of Complexity, 4:246.

Antognetti, P., Caviglia, D., Profumo, E., 1982. *CAD Model for threshold and subthreshold conduction in MOSFETs.* IEEE J. on Solid Stete Circuits, 17, 1982.

Amari, S., 1983, *Field theory of self-organizing neural nets,* IEEE Transactions on Systems, Man, and Cybernetics, SMC-13, pp 741-748.

Amari, S., 1972, *Characteristics of random nets of analog neuron-like elements,* IEEE Transactions on Systems, Man, and Cybernetics, SMC-2, pp 643-657.

Amari, S., 1990, *Mathematical Foundations of Neurocomputing,* Proceedings of the IEEE, vol. 78, No 9, sep. 1990.

Anderson, J., 1972, *A Simple Neural Network Generating an Interactive Memory,* Mathematical Biosciences 14, pp 197-220.

Arai, M., 1989, *Mapping Abilities of Three-Layer Neural Networks,* Proceedings of the Int. Joint Conf. on Neural Networks, Whashington D.C. I-419-423.

Arbib, M., 1989, *The Metaphorical Brain,* 2nd edition, New York: Wiley.

Banks, S., Harrison, R., 1991, *A Neural Network Approach to Lyapunov Function Design,* Mathematical and Intelligent models in system simulation, IMACS, pp. 345-348.

Barraquand B., Langlois, B., Latombe, J., 1992, *Numerical potential field techniques for robot path planning,* IEEE Trans. Syst. Man, and Cyber. vol. 22, No.2, pp 224-241

Baum, E., Haussler, D., 1989, *What Size Net Gives Valid Generalization?* Neural Computation 1, 151-160, MIT.

Bertero, M., Poggio, T., 1988, *Ill Posed Problems in Early Vision,* Proceedings of the IEEE, vol. 76, No. 8, pp 869-888.

Bruck, J., Goodman, J., 1988, *A Generalized Convergence Theorem for Neural Networks,* IEEE Transactions on Information Theory, vol. 34, No. 5, pp 1089-1092.

Cancelo, G. Hansen, S., 1992, *A programming and training environment for ETANN,* Fermilab Report July 1992.

Cancelo, G., Hansen, S., 1993, *ETANN Trainer and Programmer (T&P) targets fast Neural Network weight setting to allow "chip-in-the-loop" training,* V Reunión de Trabajo en Procesamiento de Información y Control realizada del 1 al 4 Diciembre 1993 en Tucumán, Argentina.

Cancelo, G., Hansen, S., 1994, *Analog Neural Network Development System with fast on line training capabilities* Twentieth Annual Conference of the IEEE Industrial Electronic Society. IECON 1994. realizado en Bologna Italia del 5 al 9 Septiembre de 1994.

Cancelo, G., Hansen, S., 1995, *NETMAP, an Integrated System for Building Neural Networks* aceptado para su publicación en el Journal of Sistem Architecture.

Cancelo, G., Mayosky, M., 1995, *Gaussian Networks for Autonomous Vehicle Path Planning,* IEEE Int. Symp. on Artificial Neural Networks, Taipei, Taiwan, B1-01.

G. Cancelo, M. A. Mayosky, Roberto Vignoni, *Neural Networks for vehicle path planning and control.* IFAC: LCA '95, 4th symposium on low cost Automation, Sept. 13-15, 1995.

Cancelo, G., Mayosky, M., *A Gaussian Neuron Circuit for RBF-function approximation* Enviado al IEEE Transactions on Circuits and Systems.

Chen, S., Cowan, F., Grant, P., 1991, *Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks,* IEEE Transaction on Neural Networks, vol. 2, No. 2, pp 302-309.

Cotter, N., 1990, *The Stone-Weierstrass theorem and its applications to neural networks,* IEEE Transactions on Neural Networks, vol 1, No 4.

Cybenko, g., 1988, Approximation by superpositions of a sigmoidal function, Urbana: University of Illinois.

DuBois, J.M., Schneider, M.F. and Khodorov, B.I. 1983. *Voltage dependence of intermembrane charge movement and conductance activation of batachotoxin-modified sodium channels in frog node of Ranvier.* Journal of General Physiology, 81:829.

Duda, R., Hart, P., 1973, *Pattern Classification and Scene Analysis,* New York: Wiley.

Early, J.M. 1952. *Efects of space-charge layer widening in junction transistors.* Proceeding of the IRE, 40 pp.1401.

Eggbrecht, L.C., 1991. *Interfacing to the IBM Personal Computer.* SAMS, In: Macmillan Computer Publishing.

Euzent, B., Boruta, N., Lee, J., Jenq, C., 1981, *Reliability aspects of a floating gate EEPROM*. Proceedings Int. Reliability Physics Symp.

Fisher, W., Fujimoto, R., Smithson, R., 1991, A Programmable Analog Neural Processor, IEEE Transactions on Neural Networks, March, vol.2, No2, pp.222-229.

Funahashi, K., 1989, *On the approximate Realization of Continuous Mappings by Neural Networks*, Neural Networks, vol 2, pp183-192.

Foux, G., Heymann, M., Bruckstein, A., 1993, *Two dimentional robot navigation among unknown stationary poligonal obstacles*, IEEE Trans. Rob. Autom. vol. 9, No.1, pp 96-102.

Gallant, S., 1990, *A Connectionist Learning Algorithm with Provable Generalization and Scaling Bounds*, Neural Networks, vol. 3, pp 191-201.

Geva, S., Sitte, J., 1992, *A Constructive Method for Multivariate Function Approximation by Multilayer Perceptrons*, IEEE Transaction on Neural networks, vol. 3, No. 4.

Grasberg, S., 1988, *Competitive learning: From interactive activation to adaptive resonance*, In Neural Networks and natural Intelligence, pp213-250, Cambridge MA:MIT Press.

Gouzenes, L., 1984, *Strategies for solving collision-free trajectory problems for movile and manipulator robots*, Int. J. Robotics Res., vol. 3 No. 4, pp 51-65.

Guo, H., Gelfand, S., 1991, *Analysis of Gradient Descent Learning Algorithms for Multilayer Feedforward Neural Networks*, IEEE Transactions on Circuits and Systems, vol. 38, No. 8, pp 883-893.

Haykin, S., 1994, *Neural Networks, A Comprehensive Foundation*, Macmillan, New York.

Hebb, D., 1949, *Organization of Behaviour: A Neuropsychological Theory*. New York: Wiley.

Hertz, J., Krogh, A., Palmer, R., 1991, *Introduction to the Theory of Neural Computation*, Addison Wesley.

Hinton, G., Nowlan, S., 1990, *The boostrap Widrow-Hoff rule as a cluster formation algorithm*, Neural Computation 2, pp 355-362.

Hinton, G., Sejnowsky, t., 1983, *Optimal perceptual inference*, Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp 448-453. Washington DC.

Hodgkin, A.L., Huxley, A.F., 1952a, *Measurements of current-voltage relations in the membrane of the giant axon of Loligo.* Journal of Physiology, 116:424.

Hodgkin, A.L., Huxley, A.F., 1952b, *Current carried by sodium and potasium ions through the membrane of the giant axon of Loligo.* Journal of Physiology, 116:449.

Hoeneisen, B. and Mead, C.A. 1972. *Current-voltage characteristics of small size MOS transistors.* IEEE Transactions on Electron Devices, 19. pp.382.

Hopfield, J., Tank, T., *Neural Computation of decisions in optimization problems,* Biological Cybernetics, 52, pp 141-152.

Hopfield, J., Tank, T., 1986, *Computing with neural circuits: A model,* Science 233, pp 625-633.

Huang, S., Huang, Y., 1990. *Learning Algorithms for Perceptrons Using Back-propagation with Selective Updates,* IEEE Control Systems Magazine, Abril 1990, pp 56-61.

Huang, S., Huang, Y., 1991. *Bounds on the number of hidden neurons in multilayer Perceptrons.* IEEE Transaction on Neural Networks, vol.2 No 1. Jan. 1991

Hush, D., Horne, B., 1992, *Error Surfaces for Multilayer Perceptrons,* IEEE Transactions on System, man, and Cybernetics, vol. 22, No. 5, p. 1152-1161.

Hush, D., Horne, B., 1993, *Progress in Supervised Neural Networks,* IEEE Signal Processing Magazine, vol. 10, No. 1.

Jacoboni, C., Canali, C., Ottaviani, G. and Quaranta, A., 1977. *A review of some charge transport properties of silicon.* Solid State Electronics, 20 pp.77.

Katz, B. 1966. *Nerve, Muscle and Synapse.* New York: McGraw-Hill.

Kohonen, T., *The Self-Organizing Map,* Proceedings of the IEEE, vol. 78, No 9, sep. 1990.

Kolodny, A., Nieh, S., Eitan, B., Shappir, J., *Analysis and Modeling of Foating-Gate EEPROM Cells.* Transactions on Electron Devices, vol. ED-33, No 6, Junio 1986.

Kruschke, J., Movellan, J., 1991, *Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Back-propagation Networks,* IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, No. 1, pp 273-280.

LeCun, Y., 1988, *A theoretical framework for back-propagation,* Proc. of the 1988 Connectionist Models Summer School (Touretzky, Hinton, Sejnowsky, ed.) pp 21-28. San Mateo, CA:Morgan Kaufmann.

Lehmann, C., Viredaz, M., Blayo, F., 1993, *A Generic Systolic Array Building Block for Neural Networks with On-Chip Learning*, IEEE Transaction on Neural Networks, vol4 No3, May 1993, pp.400-407.

Lippmann, R., 1987, *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, vol 4, No 2, pp2-22.

Lippmann, R., 1989a, *A review of neural networks for speech recognition*, Neural Computation 1, pp 1-38.

Lippmann, R., 1989b, *Pattern classification using neural networks*, IEEE Communications magazine, 27, pp 47-64.

Liu, B., Si, J., 1993, *The Best Approximation Properties and Error Bound of Gaussian Networks*, Proceedings of the 32nd Conference on Decision and Control, San Antonio, Texas, Dec. 1993, pp 2798-2803.

Lozano-Perez, T., Wesley, M., 1979, *An algorithm for planning collision free paths among polyedral obstacles*, Comm. of the ACM, vol. ACM-22, No.10, pp 224-241.

Makhoul, J., El-Jaroudi, A., Schwartz, R., *Formation of Disconnected Decision Regions with a Single Hidden Layer.* Proceedings of IEEE International Joint Conference on Neural Networks, Washington D.C., Junio 18-22, 1989, I-455-I-460.

Massobrio, G., Antognetti, P., 1993. *Semiconductor device modeling with Spice.* New York: McGraw-Hill.

McCulloch, W., Pitts, W., 1943, *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, 5, pp 115-133.

McEliece, R., et al, 1987, *The Capacity of the Hopfield Associative Memory*, IEEE Transactions on Information Theory, vol. IT-33, No. 4, pp 461-482.

Mead, C., Conway, L., 1980, *Introduction to VLSI systems*, Reading, MA: Addison-Wesley.

Mead, C., 1987a, *Silicon model of neural computation*, 1st Int. Conf. on Neural Networks, vol. 1, pp 93-106, San Diego, CA.

Mead, C., 1987b, *Neural hardware for vision*, Engineering and Science, 2-7.

Mead, C. 1989. *Analog VLSI and Neural Systems.* Reading, MA: Addison-Wesley.

Mead, C., 1990, *Neuromorphic Electronic Systems*, Proceedings of the IEEE, vol. 78, No 10, oct. 1990.

Mees, A., Jackson, M., Chua, L., 1992, *Device Modeling by Radial Basis Functions*, IEEE Transactions on Circuits and Systems, vol. 39, No. 1, pp 19-27.

Michel, A., Farrell, J., 1990, *Associative Memories via Artificial Neural Networks*, IEEE Control Systems Magazine, April 1990, pp 6-17.

Miller III, T., Glanz, F., Kraft III, G., 1990, *CMAC: An Associative Neural Network Alternative to Backpropagation*, Proceedings of the IEEE, vol. 78, No 10.

Minsky, M., Papert, S., 1988, *Perceptrons*, Expanded edition, Cambridge, MA: MIT Press.

Minsky, M., Selfridge, O., 1961, *Learning in random nets*, Information Theory Fourth London Symposium. London: Butterworth.

Morgan, N., Beck, J., Kohn, P., Bilmes, J., 1992, *Neurocomputing on the RAP*, Digital Parallel Implementations of Neural Networks, K.W. Przytula and V.K.Prasanna, Eds. Engelwood Cliffs, NJ: Prentice Hall.

Muroga, S. and Toda, I. 1966. *Lower bound on the number of threshold functions.* IEEE Transactions on Electronic Computers, EC-15 pp.805.

Nagata, S., Sekiguchi, M., Asakawa, K., 1990, *Movil robot control by a structured hierarchical neural network*, IEEE Contr. Sist. Magazine, vol. 10, No.3, pp 69-76.

Narendra, K., Parthasarathy, K., 1991, *Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks*, IEEE Transactions on Neural Networks, vol. 2, No. 2, pp 252-262.

National Semiconductor, 1992. *Interface: Line drivers and receivers databook.*

Nguyen, D., Lee, J., 1989a, *A New LMS-based Algorithm for Rapid Adaptive Classification in Dynamic Environments*, Neural Networks, vol. 2, pp 215-228.

Nguyen, D., Widrow, B., 1989b, *The truck backer-upper: An example of self-learning in neural networks*, Int. Joint Conf. on Neural Networks, vol. 2, pp 357-363, Washington DC.

Nicollian, E., Brews, J., *MOS physics and technology*, Wiley, New York.

Parker, D., 1987, *Optimal algorithms for adaptive networks: Second order back-propagation, second order direct propagation, and second order Hebbian learning*, IEEE 1st Int. Conf. on Neural Networks, vol. 2, pp 593-600. San Diego, CA.

Poggio, T., Girosi, F., 1990a, *Networks for Approximation and Learning*, Proceedings of the IEEE, vol. 78, No 9, sep. 1990.

Poggio, T., Girosi, F., 1990b, *Regularization algorithms for learning that are equivalent to multilayer networks*, Science 247, pp 978-982.

Poggio, T., Torre, V., Koch, C., 1985, *Computational vision and regularization theory*, Nature (London) 317, pp 314-319.

Rosenblatt, F., 1958. *The Perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review 65, 386-408.

Rosenblatt, F., 1960a. *Perceptron simulation experiments.* Proceedings of the Institute of Radio Engineers. 48, 301-309.

Rosenblatt, F., 1960b. *On the convergence of reinforcement procedures in simple Perceptrons.* Report VG-1196-G-4. Cornell Aeronautical Laboratory, Buffalo, NY.

Rosenblatt, F., 1962. *Principles of Neurodynamics.* Washington, DC: Spartan Books.

Roubik, G., Temes, G., 1986. *Analog MOS integrated circuits for signal processing.* New York: John Wiley.

Rumelhart, D., McClelland, J., 1986a, *Parallel Distributed Processing, Explorations in the Microestructure of Cognition, vol. 1: Foundations*, MIT Press.

Rumelhart, D., Hinton, G., Williams, R., 1986b, *Learning internal representation by back-propagating errors*, Nature (London), 323, pp 533-536.

Sanner, R., Slotine, J., 1992, *Gaussian Networks for Direct Adaptive Control*, IEEE Transactions on Neural Networks, vol. 3, No. 6, pp 837-863.

Schwartz, E., *Computational Neuroscience, Cambridge*, MA: MIT Press

Shichman, H., Hodges, D., *Modeling and simulation of insulated-gate field-effect transistor switching circuits*, IEEE J. on Solid Stete Circuits, SC-3.

Shilov, G., *Mathematical Analysis, A special course*, Pergamon Press, NY, 1965.

Simpson, P., 1990, *Artificial Neural Networks*, Pergamon Press.

Slack, M., 1993, *Navigation Templates: Mediating qualitative guidance and qualitative control in movil robots*, IEEE Transactions on Syst., Man, and Cybernetics, vol. 23, No.2, pp 452-466.

Stinchcombe, M., White, H., 1989, *Universal Approximation Using Feedforward Networks with non-Sigmoid Hidden Layer Activation Functions*, Proceedings of the Int. Joint Conf. on Neural Networks, Whashington D.C. I-613-617.

Solla, S., Levine, E., Fleisher, M., 1988, *Accelerated learning in layered neural networks*, Complex Systems 2, pp 625-640.

Sze, S., *Physics of semiconductor devices*. Wiley, New York.

Tank, D., Hopfield, J., 1986, *Simple neural Optimization Networks: An A/D convertor, signal decision circuit and a linear programming circuit*, IEEE Transactions on Circuit and Systems CAS-33, pp 533-541.

Tank, D., Hopfield, J., 1987, *Neural Computation by Concentrating Information in Time*, Proc. of the Nat. Acad. Sciences of the USA, 84, 1896-1900.

Texas Instruments, 1988. *TMS320C25 Assembly Languaje Tools*.

Texas Instruments, 1991. *TMS320C2x/5x Optimizing C Compiler*.

Tikhonov, A., Arsenin, 1977, *Solutions to ill-posed problems*, Whashington DC: Whinston.

Treleaven, P., Pacheco, M., Vellasco, M., 1989, VLSI Architectures for Neural Networks, IEEE Micro, Dic. 1989, pp.8-26.

Tsividis, Y., 1987. *Operation and Modelling of MOS Transistor*. New York: McGraw-Hill.

Vapnik, V., 1982, *Estimation of Dependences Based on Empirical Data*, New York: Springer-Verlag.

Vapnik, V., Chervonenkis, A., 1971, *On the uniform convergence of relative frequencies of events to their probabilities*, Theoretical Probabilities and its Applications, 17, pp 264-280.

von Neumann J., Burks, A., Goldstine, H., 1946, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*, U.S.Army Ordnance Department Report.

von Neumann J., 1958, *The Computer and the Brain*, New Haven, CT: Yale University Press.

von Neumann J., 1986, *Papers of Jonh von Neumann on Computing and Computer Theory*, Cambridge MA: MIT Press.

Wang, S., 1980 *Charge retention of floating gate transistors under applied bias condition*. IEEE Transaction on Electron Devices, vol ED-27, p.297.

Werbos, P., 1974, *Beyond regresion: New tools for prediction and analysis in the behavioral sciences*, PhD. Thesis, Harvard University, Cambridge, MA.

Werbos, P., 1989, *Backpropagation and Neurocontrol: A review and Prospectus*, Int. Joint Conf. on Neural Networks, vol. 1, pp 209-216, Washington, D.C.

Werbos, P., 1990, *Backpropagation through time: What it does and how to do it.* Proceeding of the IEEE. vol 78, No9, pp.1550-1560.

Werbos, P., 1992, *Neural Networks and the Human Mind: New Mathematics Fits Humanistic Insight*, IEEE Int. Conf. on Systems, man, and Cybernetics, vol. 1, pp 78-83, Chicago, IL.

Widrow, B., and Stearns, S. 1985. *Adaptive Signal Processing.* Englewood Ciffs, Nj: Prentice-Hall.

Widrow, B., and Winter, R., 1988, Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition, IEEE Computer Magazine, vol. 21, No. 3.

Widrow, B., and Lehr, M., 1990. 30 years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation. Proceeding of the IEEE. vol 78, No9, pp.1415-1440.

Wiener, N., 1948, *Cybernetics: Or Control and Communication in the Animal and the Machine*, New York, Wiley.

Zelinsky, A., 1992, *A movil robot exploration algoritm*, IEEE Trans. Rob. Autom. vol. 8, No.6, pp 707-717.