



INAOE

Implementación Electrónica de una Red Neuronal Híbrida para el Reconocimiento del llanto de bebés.

Por

Israel Alejandro Suaste Rivas

Tesis sometida como requisito parcial para obtener el grado de **Maestro en Ciencias** en la especialidad de **Electrónica** en el Instituto Nacional de Astrofísica, Óptica y Electrónica.

Supervisada por:

Dr. José Alejandro Díaz Méndez

Dr. Carlos Alberto Reyes García

© INAOE 2004

Derechos reservados

El autor otorga al INAOE el permiso de reproducir y distribuir copias de esta tesis en su totalidad o en partes.

Tonantzintla, Puebla.

Diciembre 2004



AGRADECIMIENTOS

Agradezco con la misma importancia a:

Al Dr. Alejandro Díaz Méndez, por su invaluable colaboración durante el desarrollo de esta tesis.

Al Dr. Carlos Alberto Reyes-García, por su interés en la investigación sobre el reconocimiento del llanto de los bebés, así como por sus palabras alentadoras en cuanto a superación personal.

A toda mi familia por su apoyo incondicional.

A *todos* mis amigos y compañeros, por haber compartido tantas experiencias juntos.

A los Drs. Miguel Arias y René Cumplido, por haberme brindado las herramientas necesarias para el desarrollo de este trabajo.

Al Dr. Sergio Cano por las atenciones brindadas durante mi estancia en Cuba para la investigación relacionada con las Unidades de Llanto en los bebés.

Este trabajo es parte de un proyecto financiado por CONACYT (número 37914-A). Agradeciendo el apoyo brindado durante mis estudios.

RESUMEN

El llanto es el único medio que un bebé tiene para comunicarse con el exterior. De acuerdo a los especialistas, en el llanto se refleja su estado físico, patológico y/o anímico. Debido a esto, se han realizado varios esfuerzos por extraer características acústicas del llanto y relacionarlas con ciertas patologías.

En esta tesis se presenta el desarrollo de una arquitectura (Hardware) para el reconocimiento del llanto de bebé, con el objetivo de clasificar los tipos de llantos: normal y patológico, de bebés con problemas de audición (sordera) y asfixia.

Las características acústicas son obtenidas usando el Modelo de Predicción Lineal y para su clasificación, se utiliza una Red Neuronal Híbrida. Los resultados son muy alentadores, ya que se ha logrado un 94.6% de clasificación correcta en 2268 muestras de un segundo cada una. El resultado obtenido en simulaciones realizadas con Software fue el mismo que en la implementación electrónica.

Este trabajo presenta la primera etapa de un sistema dedicado al reconocimiento automático del llanto, el cual permitiría proporcionar una mejor atención al bebé para su cuidado. La detección temprana de sordera ayuda para que el bebé tenga una mejor adaptación al medio que lo rodea, así como para evitar el retraso del aprendizaje lingüístico. Por otro lado, si al momento de nacer, el bebé tiene problemas de asfixia y no es tratado con los cuidados necesarios (por no haberse detectado este problema), corre el riesgo de que algunas de sus facultades mentales se vean disminuidas y en el peor de los casos, puede provocar la muerte del infante.

ABSTRACT

Cry is the only mean a baby has to communicate with the word. According to specialists, cry reflects his physical and pathological state. Because of this, several efforts had been realised to extract acoustic characteristics of crying and its relations with certain pathologies.

This thesis presents the develop of a Hardware architecture for babies cry recognition, with the goal of classifying different types of crying: healthy (normal) and pathologic, of audition problems babies (deafness) and asphyxia.

The acoustics characteristics are obtained using the Lineal Prediction Model and for its classification, we use an Hybrid Neural Network. The obtained results are very promising, with a 94.6% of correct classification in 2268 samples a second each one. The obtained results in software simulations were the same in the Hardware implementation.

This work presents the first stage of a system dedicated to automatic crying recognition, which would allow us to give the baby a better care. Early deafness detection help to improve the babies adaptation to his environment, and avoid linguistic learning delay. In the other hand, if a baby has asphyxia problems at his birth, and is not treated accordingly he has the risk that his mental faculties diminish, and in the worst of the cases he is in dead danger.

ÍNDICE

Capítulo 1.	INTRODUCCIÓN.....	1
1.1.	Antecedentes.....	2
1.2.	Objetivo.....	4
1.3.	Metodología.....	4
1.4.	Organización de la tesis.....	5
Capítulo 2.	MARCO TEÓRICO.....	7
2.1.	Hardware.....	7
2.1.1.	Arreglos de compuertas programables en campo (FPGA).....	9
2.2.	Herramientas utilizadas para la implementación en Hardware.....	11
2.3.	Modelos y métodos para la clasificación de patrones.....	15
2.3.1.	Comparación de patrones.....	15
2.3.2.	Aproximación estadística.....	15
2.3.3.	Modelos estocásticos.....	16
2.3.4.	Sistemas basados en conocimientos.....	17
2.3.5.	Modelos conexionistas.....	17
2.4.	Llanto del Bebé.....	18
2.4.1.	Fonación.....	18
2.4.2.	Aparato fonador.....	18
2.5.	Generación del llanto.....	19
2.5.1.	Modelo del llanto.....	20
2.6.	Parámetros fundamentales del llanto.....	20
2.6.1.	Intensidad.....	20
2.6.2.	Frecuencia fundamental.....	21
2.6.3.	Formantes.....	21
2.6.4.	Unidades de llanto.....	21
2.6.5.	Coeficientes de predicción lineal.....	22
2.6.6.	Coeficientes cepstrales en frecuencia de Mel.....	22
Capítulo 3.	RECONOCIMIENTO DEL LLANTO DE BEBÉ.....	23
3.1.	Procesamiento de la señal.....	24
3.2.	Base de datos de las muestras de llanto.....	24
3.3.	Análisis predictivo lineal.....	26
3.4.	Análisis de componentes principales.....	28
3.4.1.	Algoritmo.....	29
Capítulo 4.	CLASIFICACIÓN DE PATRONES.....	31
4.1.	Redes neuronales.....	31
4.1.1.	Entrenamiento.....	34
4.2.	Conjuntos difusos.....	35
4.3.	Relaciones matemáticas.....	36
4.4.	Productos relacionales.....	37
4.4.1.	Producto Círculo o Circlet (R o S).....	37
4.4.2.	Producto subtriángulo.....	37

4.4.3.	Producto supertriángulo	38
4.4.4.	Producto cuadrado.....	38
4.4.5.	Ejemplo de aplicación de productos relacionales	38
4.5.	Relaciones difusas	38
4.6.	Neurona difusa.....	39
4.7.	Red neuronal relacional.....	40
4.8.	Descripción del reconocimiento de llanto utilizando la red Fuzzy Neuronal Relacional	41
4.9.	Funciones de membresía.....	43
4.9.1.	Función de membresía triangular (Δ)	43
4.9.2.	Función de membresía trapezoidal (T)	45
4.9.3.	Función de membresía Trapezoidal Densa (T_D).....	45
4.9.4.	Cálculo del valor deseado de salida	46
4.10.	Entrenamiento.....	47
4.10.1.	Actualización de parámetros.....	47
Capítulo 5.	IMPLEMENTACIÓN	49
5.1.	Implementación en software.....	49
5.2.	Detección del tipo de llanto	50
5.3.	Implementación electrónica.....	52
Capítulo 6.	RESULTADOS	61
6.1.	Resultados obtenidos en Matlab	61
6.2.	Resultados obtenidos en el FPGA	65
Capítulo 7.	CONCLUSIONES.....	69
7.1.	Conclusiones.....	69
7.2.	Trabajo futuro	70
	LISTA DE FIGURAS.....	73
	LISTA DE TABLAS	75
	BIBLIOGRAFÍA.....	77
APÉNDICE A .	UNIDADES DE LLANTO	83
APÉNDICE B .	IMPLEMENTACIÓN DE UNA RED FEED FORWARD EN HANDEL-C.....	85
APÉNDICE C .	MATRIZ RELACIONAL Y VECTORES DE PRUEBA.....	89
APÉNDICE D .	RESULTADOS DE LA IMPLEMENTACIÓN ELECTRÓNICA	91

Capítulo 1 INTRODUCCIÓN

El llanto de un bebé recién nacido es el único medio de comunicación para expresar su estado anímico o físico tales como hambre, dolor, sueño, miedo, incomodidad entre otros. El llanto del bebé ha sido clasificado en un conjunto de clases de llanto acústicamente distintos, indicando necesidades específicas como el hambre o el sueño [1].

Actualmente el único medio que se tiene para reconocer el tipo de llanto es a través del oído entrenado de la madre o de la persona al cuidado del bebé y los gestos que realiza en ese momento; por consiguiente, esta interpretación es completamente subjetiva y se basa en la experiencia de la persona. Existen diversos trabajos para reconocer y clasificar el llanto no solo para identificar dolor, hambre y sueño, sino también dificultades o complicaciones que se presentan al momento de nacer como hipoxia y daños en el Sistema Nervioso Central (CNS) [2]; así como enfermedades y problemas que el neonato ya tenía tales como sordera, hidrocefalia y daños cerebrales [3] [4]. Estos trabajos han mostrado un nivel de reconocimiento muy aceptable con respecto al porcentaje de clasificación que se realiza de manera correcta el cual es de noventa o más, pero la mayoría se encuentra todavía a nivel software.

La onda de llanto contiene información relevante y se pueden distinguir los diversos tipos de llantos para reconocer las condiciones por las que el bebé lloró, con lo cual resulta viable su extracción, reconocimiento y clasificación por medios automáticos.

El presente proyecto está dirigido a crear un sistema en hardware que evalúe las características acústicas del llanto, con particular atención en los bebés con problemas de sordera (hipoacúsicos) y asfixia, dando como resultado el reconocimiento y clasificación automática de los tipos de llantos a partir de las características más relevantes. Los resultados servirán para que madres, enfermeras, médicos y en general personal que atienden bebés puedan saber si el neonato está normal o tiene sordera para darle el cuidado adecuado.

1.1 Antecedentes

El análisis del llanto de bebé ha sido el tema de numerosas investigaciones en los últimos años. Las primeras referencias que se tienen son de 1850 cuando Charles Darwin trató este tema, al diagnosticar el llanto a través de una serie de fotografías y dibujos los cuales ilustraban varias expresiones de emoción o dolor [5].

Casi un siglo después de que Darwin escribiera sus observaciones, surgen las primeras investigaciones que fueron realizadas por Hurlock en 1950, ya que supuso que el llanto de los niños era intencionado y contenía información de su estado [6]. En 1968, Wasz-Höckert realizó un estudio de los diferentes tipos de llanto infantil y para diferenciarlos se basaba en el análisis de espectrogramas [7]. Un grupo dirigido por Olé Wasz-Höckert determinó que diferentes tipos de llantos podían ser clasificados mediante técnicas auditivas [8]. Esta experiencia se considera como el punto de partida de la etapa moderna del análisis del llanto infantil. Después Golub en 1982 realizó una serie de pruebas a las que llamo “Primer Análisis Auditivo” donde noventa y tres madres con sus niños fueron entrenados durante dos semanas para que pudieran ser capaces de identificar a su bebé, escuchando el llantos de varios niños. Los resultados de esta prueba fueron que más del 95% de las madres pudo identificar a su propio bebé [9].

Otros experimentos probaron que el llanto no solo podía ser usado para identificar a un niño en especial, sino que además, también ayudaba para determinar el estado de salud del infante. Zeskind mostró que observadores entrenados podían identificar por medio del sonido del llanto si el neonato se encontraba normal o con algún factor de riesgo. Esta información llevó a creer a los investigadores que era posible extraer cierta información de la forma de onda del llanto para poder determinar de qué tipo era, por ejemplo si lloró por estar enfermo o por hambre [10].

Con el avance de la tecnología se pudieron extraer de forma automática características importantes de la onda de llanto, alentado a que un mayor número de investigadores se interesaran en el tema, con lo cual nació un nuevo campo de investigación llamado “Análisis de llanto infantil”.

Recientemente se han desarrollado trabajos de investigación utilizando RNA (Redes Neuronales Artificiales), descritas en un modelo computacional para su uso. Dentro de esta categoría destaca el trabajo realizado por Cano donde se utilizan Mapas Auto-Organizados de Kohonen para clasificar las unidades de llanto infantil, obteniendo resultados aceptables [11] [12]. En otros reportes dirigidos por Cano se ha probado la clasificación del llanto de bebés con diferentes técnicas basadas en RNA en las que destacan los Algoritmos de Cuantificación Vectorial Adaptable obteniendo diferentes resultados [13] [14].

Un trabajo muy completo fue realizado por M. Petroni et al. en el cual se clasifican llantos de infantes de dolor y sin dolor, utilizando diversas arquitecturas RNA entre ellas: red de alimentación hacia delante (feed-forward) completamente interconectada, redes neuronales recurrentes, redes neuronales con tiempos de retardo y redes neuronales correlacionadas en cascada. Para las cuales se emplearon diversos algoritmos de entrenamiento. El mejor resultado que reporta, se obtuvo con la red feed-forward alcanzando un 86.2% de precisión [5].

Últimamente se realizó un trabajo de tesis de maestría por Taco Ekkel, la cual es una investigación que se desprende del Grupo de Procesamiento de Voz (GPV) de Santiago en Cuba. Su objetivo fue clasificar llantos de tipo normal y anormal (Hipoxia) utilizando una red neuronal de base radial, en sus resultados reporta un $85\% \pm 5\%$ de exactitud utilizando todas las unidades de llanto [2].

Uno de los reportes más recientes es el que desarrolló José Orozco, como trabajo de tesis de maestría. Pudo clasificar entre niños normales y anormales (sordos), reconociendo llantos causados por hambre y dolor entre esas dos clases. Los resultados reportados son de 98.5% utilizando una red feed-forward entrenada con el método de Gradiente Conjugado Escalado (SCGM) [3].

Existen otros trabajos relacionados con el análisis del llanto de bebé, los últimos mencionados utilizan RNA como medio para el reconocimiento y clasificación de la onda de llanto, obteniendo resultados muy alentadores con algunas estructuras de redes neuronales aunque estos son todavía a nivel simulación.

Las Redes Neuronales se pueden implementar tanto en software como hardware; pero a nivel hardware se presenta el problema de interconectar un sistema complejo. Usualmente para los sistemas de propósito específico se emplea el diseño de un circuito integrado de aplicación específica (ASIC) donde el principal problema es el costo y tiempo de diseño. Para poder realizar una red neuronal a nivel hardware se necesita contar con un dispositivo capaz de crear hardware complejo [15].

En últimas fechas las RNA han sido implementadas en FPGA (Arreglo de compuertas programables en campo) [16]. Estos dispositivos tienen la capacidad de combinar programabilidad con el aumento en la velocidad de operación utilizando hardware en paralelo. Una de las restricciones es la limitada densidad lógica generada por las características propias del dispositivo [17]. Para solucionar este problema se crean las estructuras complejas del sistema y se llaman cuando son requeridas, en vez de crear muchas estructuras del mismo tipo [18].

Para el diseño de estos dispositivos se cuenta con herramientas de diseño basadas en descripción de los circuitos mediante lenguajes de alto nivel como Handel-C que permiten una rápida programación y simulación del sistema para su implementación en hardware [19].

1.2 Objetivo

El objetivo de este trabajo es diseñar e implementar una arquitectura electrónica dedicada a la clasificación de patrones del llanto de los bebés, seleccionando las características acústicas más adecuadas, para distinguir entre llantos de niños normales, con problemas de sordera y dificultades de respiración (asfixia).

1.3 Metodología

Debido a la complejidad del diseño de un sistema para el reconocimiento de patrones, esta tesis se enfoca solamente a la etapa de clasificación la cual se realizará tanto a nivel simulación como en un dispositivo electrónico, haciendo primero un estudio sobre el análisis acústico del llanto así como la elección de un método de extracción de parámetros que arroje buenos resultados y que su implementación no sea tan compleja de realizar físicamente.

La arquitectura propuesta es implementada en un FPGA. Para el diseño del sistema se busca paralelizar las operaciones con el objetivo de minimizar el tiempo del procesamiento, aprovechando las características propias del FPGA.

La actividad propuesta a desarrollar:

Análisis Acústico: Se examinarán las unidades de llanto como alternativa durante el procesamiento de la señal, comparando los resultados con una segmentación por intervalos de tiempo.

Etapas de clasificación: Se buscará un clasificador de patrones que sea óptimo para su implementación. El desarrollo y simulación del clasificador se realizan en Matlab. El diseño de cada bloque y la simulación para su implementación en Hardware se realizarán por medio del programa de alto nivel Handel C.

1.4 Organización de la tesis

Este trabajo se encuentra organizado en siete capítulos que comprenden una introducción, una visión general de los métodos que se emplean para la clasificación de llantos de bebé junto con las bases teóricas, el sistema propuesto para el reconocimiento de los patrones, la implementación, presentación de los resultados y conclusiones.

El capítulo 1 comprende la introducción al trabajo realizado, el objetivo del mismo. También, se mencionan algunos de los trabajos existentes relacionados con este trabajo.

El capítulo 2 presenta los dispositivos electrónicos disponibles, así como los métodos de clasificación y conceptos básicos de los llantos de bebé.

El capítulo 3 describe el proceso de reconocimiento automático del llanto de bebé, el análisis acústico realizado y la forma de extracción de características de la señal de llanto.

En el capítulo 4 se habla del proceso de clasificación de patrones para el reconocimiento automático de los tipos de llanto del bebé, también describe los

procedimientos de reducción, así como los parámetros para operación de la red neuronal híbrida artificial.

El capítulo 5 muestra todo el procedimiento para la implementación en Hardware. En este capítulo se describe de forma breve la plataforma de desarrollo utilizada, que en este caso fue la tarjeta RC100 de Celoxica.

En el capítulo 6 se presentan los resultados obtenidos y se comparan con algunos ya reportados.

El capítulo 7 contiene las conclusiones derivadas de la realización de este trabajo y se menciona el trabajo futuro.

Capítulo 2 MARCO TEÓRICO

2.1 Hardware

En la actualidad, el diseño de ASIC (circuitos integrados desarrollados para aplicaciones específicas) domina las tendencias en el desarrollo de aplicaciones a nivel de microelectrónica. Este tipo de diseño presenta varias opciones de desarrollo, como se observa en la Tabla 2.1. A nivel de ASIC los desarrollos completamente (full) y medianamente a la medida (semi-custom), ofrecen grandes ventajas en sistemas que emplean circuitos diseñados para una aplicación en particular. Sin embargo, su diseño ahora sólo es adecuado en aplicaciones que requieren un alto volumen de producción; por ejemplo, sistemas de telefonía celular, computadoras portátiles, cámaras de video, etcétera [16].

Los FPGA (arreglos de compuertas programables en campo) y CPLD (dispositivos lógicos programables complejos) ofrecen ventajas similares a las de un ASIC, sólo que con un menor tiempo de diseño; es decir, el tiempo que toma desarrollar un ASIC es mucho más alto que el que requeriría un FPGA o un CPLD, con la ventaja de que ambos son circuitos re-programables, en los cuales es posible modificar o borrar una función programada sin alterar el funcionamiento del circuito.

Categoría	Características
Diseño totalmente a la medida (Full-Custom)	Total libertad de diseño, pero el desarrollo requiere todas las etapas del proceso de fabricación: preparación de la oblea o base, crecimiento epitaxial, difusión de impurezas, implantación de iones, oxidación, fotolitografía, metalización y limpieza química. Los riesgos y costos son muy elevados; sólo se justifican ante grandes volúmenes o proyectos con restricciones (área, velocidad, consumo de potencia, etcétera).

Matrices de compuertas prefabricadas (Semi-custom/gate arrays)	Existe una estructura regular de dispositivos básicos (transistores) prefabricada que se puede personalizar mediante un conexionado específico que sólo necesita las últimas etapas del proceso tecnológico. El diseño está limitado a las posibilidades de la estructura prefabricada y se realiza con base en una biblioteca de celdas precaracterizadas para cada familia de dispositivos.
Celdas estándar precaracterizadas (Semi-custom/standard cells)	No se trabaja con alguna estructura fija prefabricada en particular, pero sí con bibliotecas de celdas y módulos precaracterizados y específicos para cada tecnología. Libertad de diseño (en función de las facilidades de la biblioteca); pero el desarrollo exige un proceso de fabricación completo.
Lógica programable (FPGA, CPLD)	Se trata de dispositivos que se pueden personalizar desde el exterior mediante diversas técnicas de programación. El diseño se basa en bibliotecas y mecanismos específicos de mapeado de funciones, mientras que su implementación tan sólo requiere una fase de programación del dispositivo, que por lo general realiza el diseñador en unos pocos segundos.

Tabla 2.1 Tecnologías de fabricación de circuitos integrados.

En la actualidad existe una gran variedad de dispositivos lógicos programables, los cuales se usan para reemplazar circuitos SSI (pequeña escala de integración), MSI (mediana escala de integración) e incluso circuitos VLSI (muy alta escala de integración), ya que ahorran espacio y reducen de manera significativa el tiempo y el costo del diseño. Estos dispositivos, llamados PLD (Tabla 2.2), se clasifican por su arquitectura y la forma funcional en la que se encuentran ordenados los elementos internos que proporcionan al dispositivo sus características.

Dispositivo	Descripción
PROM	Programmable Read-Only Memory: memoria programable de sólo lectura
PLA	Programmable Logic Array: arreglo lógico programable
PAL	Programmable Array Logic: lógica de arreglos programables
GAL	Generic Logic Array: arreglo lógico genérico
CPLD	Complex PLD: dispositivo lógico programable complejo
FPGA	Field Program Gate Array: arreglos de compuertas programables en campo

Tabla 2.2 Dispositivos lógicos programables.

2.1.1 Arreglos de compuertas programables en campo (FPGA)

Un FPGA (Field-Programmable Gate Array) es un dispositivo multi-nivel programable de propósito general, que integra una gran cantidad de bloques lógicos programables en un chip. El tamaño y velocidad de los FPGAs, dependiendo de la aplicación puede ser equiparable a los Circuitos Integrados para Aplicaciones Específicas (ASICs), pero los FPGAs son más flexibles y su ciclo de diseño es más corto [17].

El FPGA es un arreglo de bloques lógicos programables colocados en una infraestructura de interconexiones programables; es posible programar la funcionalidad de los bloques lógicos, las interconexiones entre bloques y las conexiones entre entradas y salidas [17]. Un FPGA es programable a nivel hardware, por lo que proporciona algunas de las ventajas de un procesador de propósito general y un circuito especializado.

Los dispositivos FPGA se basan en lo que se conoce como arreglos de compuertas los cuales consisten en la parte de la arquitectura que contiene tres elementos configurables: bloques lógicos configurables (CLB), bloques de entrada y de salida (IOB) y canales de ruteado para su interconexión [16].

Por dentro, un FPGA está formado por arreglos de bloques lógicos configurables (CLB), que se comunican entre ellos y con las terminales de entrada/salida (E/S) por medio de interconexiones llamados canales de ruteado. Cada FPGA contiene una matriz de bloques lógicos idénticos, por lo general de forma cuadrada, conectados por medio de líneas metálicas que corren vertical y horizontalmente entre cada bloque (Figura 2.1).

Empleando FPGAs para cómputo, guía el camino para una clase general de organización de computadoras, la cual puede ser llamada Arquitectura de Cómputo Reconfigurable [17]. Esto permite que una máquina pueda ser configurada de acuerdo a las necesidades que se presenten en una aplicación, mientras que permite la característica de ser reutilizada en cualquier momento. Esto se logra guardando el archivo que programa al FPGA en alguna parte de la memoria externa, el cual es llamado cuando se requiera reconfigurar al equipo para desempeñar una tarea distinta.

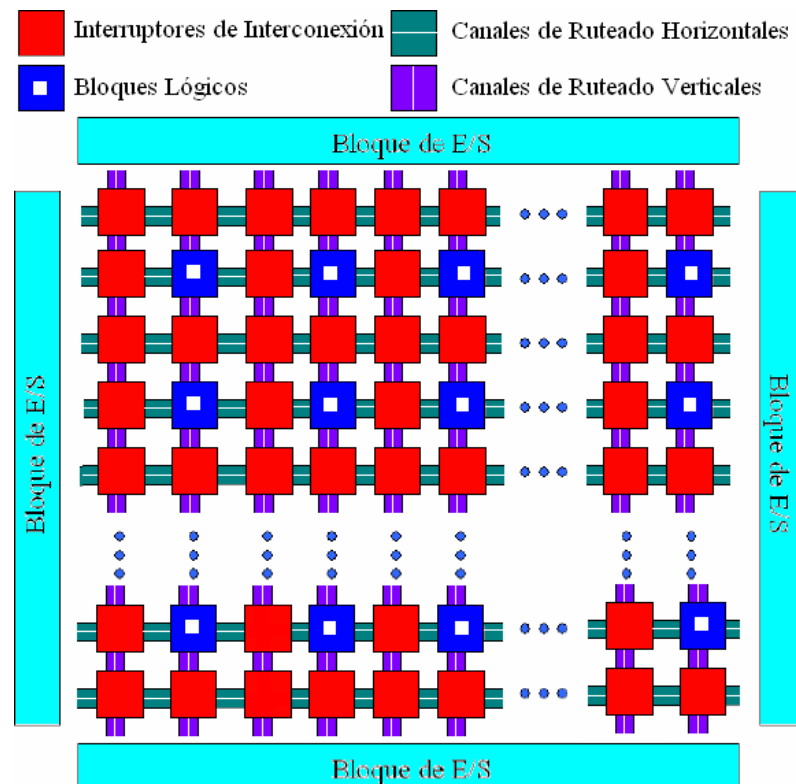


Figura 2.1 Estructura de un FPGA

2.2 Herramientas utilizadas para la implementación en Hardware

2.2.1 Handel-C

Handel-C es un lenguaje idóneo para el diseño de lógica síncrona y es particularmente adecuado para la implementación de algoritmos complejos sobre FPGAs [20], ya que cuenta con las siguientes características:

Permite crear estructuras que se ejecuten de manera paralela durante el mismo ciclo de reloj (Figura 2.2), con el comando par `{ }`.

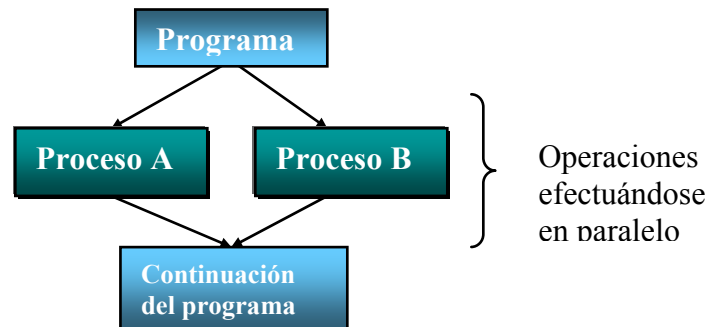


Figura 2.2 Ejemplo de bloques ejecutándose en paralelo

El código que se ejecuta tras un grupo de bloques de código de ejecución en paralelo sólo comenzará a ejecutarse si todos los procesos del bloque paralelo anterior han terminado de ejecutarse (Figura 2.3). Para ello, usa un modelo de concurrencia basado en CSP (Communicating Sequential Processes); de forma que consigue mediante una ejecución concurrente el efecto de una ejecución en paralelo real.

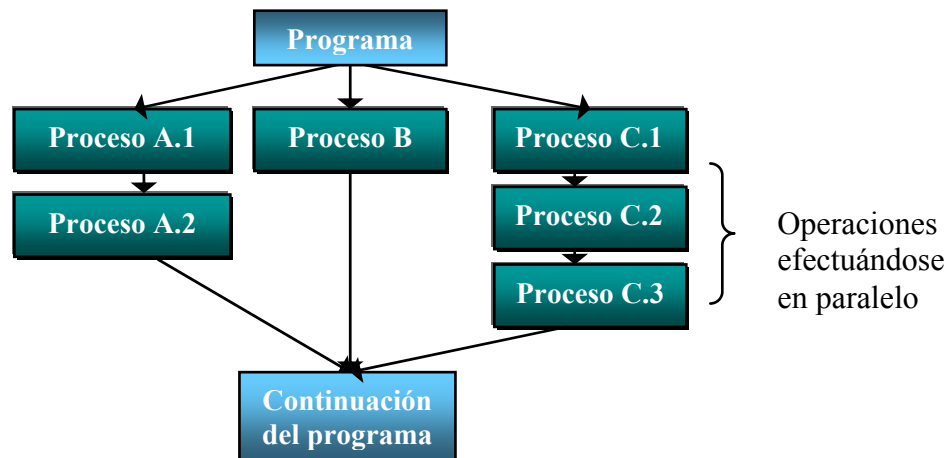


Figura 2.3 Ejemplo de bloques ejecutándose en paralelo de diferentes tamaños

Posee canales para la comunicación entre bloques de código (Figura 2.4), así como mecanismos de sincronización entre éstos.

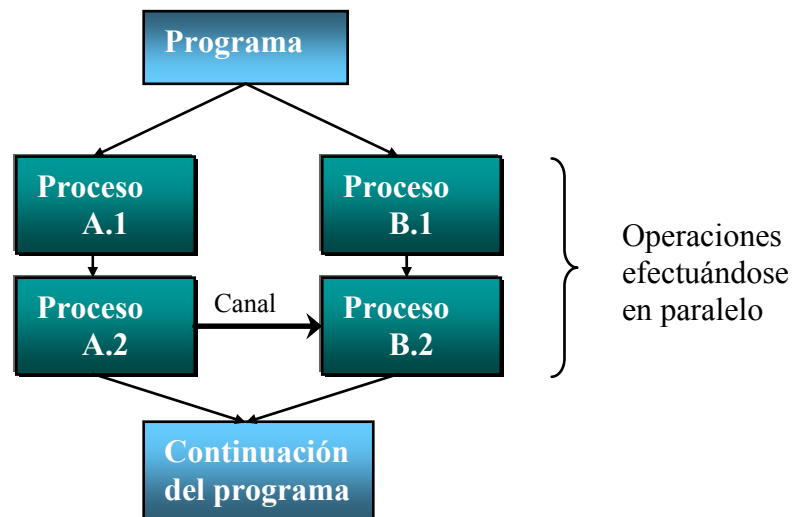


Figura 2.4 Canal de comunicación entre bloques de ejecución paralela

Posibilita la generación automática de máquinas de estado a partir de una descripción algorítmica del circuito, en términos de bloques de código paralelos y secuenciales.

Permite la generación automática de relojes, señales de habilitación y/o reset.

Además de trabajar a alto nivel, es posible controlar el funcionamiento a bajo nivel de los diseños:

El lenguaje lleva de forma implícita señales temporales muy precisas.

Es posible implementar lógica combinacional usando elementos de tipo bus, puerto e incluso señal.

En la Figura 2.5 se muestran las características principales entre ANSI-C y Handel-C [21]. Existen muchos elementos en común entre estos dos lenguajes, lo que facilita la programación y descripción en Hardware al utilizar Handel-C.



Figura 2.5 Elementos compartidos entre Handel-C y ANSI-C

2.2.2 Síntesis y ruteado del FPGA

La herramienta empleada para la síntesis y el ruteado de la arquitectura es el ISE 6.1.03 Project Navigator (Xilinx). Este programa es una de las herramientas proporcionadas por el fabricante de FPGAs para la síntesis del diseño implementado en Handel-C. Este programa se encarga de generar el archivo .bit para la configuración del FPGA a partir del archivo de descripción EDIF (.edf) generado por Handel-C.

2.2.3 Tarjeta RC100

La tarjeta RC100 de Celoxica es una de las partes más importantes en éste trabajo ya que en ella se implementó la red neuronal propuesta (Figura 2.6).

Esta tarjeta contiene dos Chips lógicos programables [22]:

- ✚ XILINX Spartan II FPGA: Es el componente principal de la tarjeta y cuenta con 200,000 compuertas. El número de parte es: XC2S200-5-FG456.
- ✚ XILINX XCR3128XL: El CPLD (Complex programmable logic device) es usado para manejar la configuración del FPGA Spartan II a través del puerto paralelo o por medio de memoria FLASH.

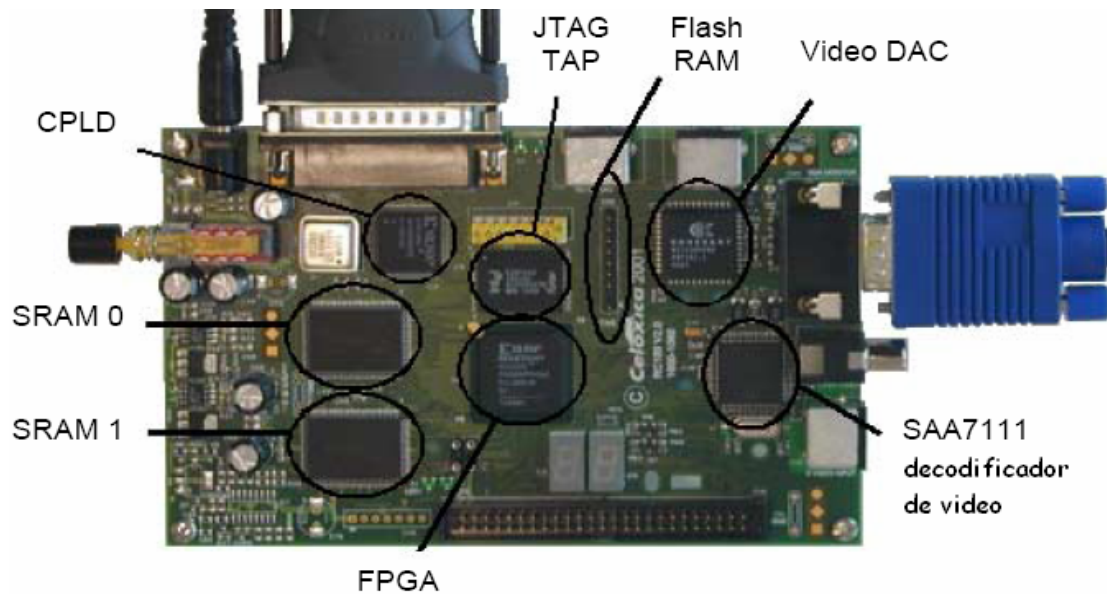


Figura 2.6 Tarjeta RC100

En la Tabla 2.3 se muestran algunos de los dispositivos y periféricos con los que la tarjeta RC100 de Celoxica cuenta [23].

Dispositivo	Función
8 Mbytes de FLASH RAM	Almacena múltiples configuraciones o puede servir para descargar datos de la PC a la tarjeta.
2 Bancos independientes de SSRAM (256k x 36 bits)	Es usada por el FPGA para guardar datos o simplemente para almacenar los datos que contiene la FLASH RAM.
video Decodificador Philips SAA7111A EVIP	Acepta señales NTSC/PAL/SECAM a través del RCA o S-video conector y sirve para la salida RGB del FPGA.
DAC (Convertidor Digital-Analógico) con mapa de colores de 24 bits	Este es usado por el FPGA para la salida de video al monitor VGA
2 puertos PS2 (mouse y teclado)	Para uso de propósito general.
Dos Displays de cátodo común	Para uso de propósito general.

Tabla 2.3 Dispositivos con los que cuenta la tarjeta RC100.

La tarjeta incluye un software de soporte que provee un controlador del dispositivo y una biblioteca para simplificar el proceso de inicialización y comunicación con la tarjeta. Entre las funciones que están incluidas en la biblioteca se encuentran las de manejo del decodificador de video, las de manejo de archivos de configuración del FPGA, las de control de relojes programables, las de transferencia de datos y las de manejo de los displays, etc. [24].

2.3 Modelos y métodos para la clasificación de patrones

Una de las etapas más importantes durante el reconocimiento del llanto de bebés, es determinar la clase o categoría a la que corresponde cada caso. A continuación se presentan algunas metodologías utilizadas con éste fin [25].

- ⊕ Comparación de patrones.
- ⊕ Aproximación estadística.
- ⊕ Modelos estocásticos.
- ⊕ Sistemas basados en conocimientos.
- ⊕ Modelos conexionistas.

2.3.1 Comparación de patrones

Consiste en comparar el patrón de entrada con una serie de plantillas o patrones que representan a las unidades a reconocer. La plantilla o patrón, no es más que un conjunto de características acústicas ordenadas en el tiempo (secuencia de vectores de parámetros o índices de una librería). La comparación de patrones incluye un alineamiento temporal no lineal y una medida de distancia [26].

2.3.2 Aproximación estadística

La aproximación más simple (y no por ello la menos eficiente) consiste en representar cada patrón mediante un vector de números, resultantes del muestreo y cuantificación (o binarización) de las señales externas (Figura 2.7) y cada clase por uno o varios patrones prototipo. Dado que existe variabilidad en las medidas registradas, cada componente del vector es una variable aleatoria y cada uno de sus valores es una realización de esa variable aleatoria [27].

Con esta aproximación un patrón no es más que un punto en el espacio de representación de los patrones, que es un espacio de dimensionalidad determinada por el número de variables consideradas. Esta aproximación concluye que es razonable que los patrones pertenecientes a una misma clase estén cercanos en el espacio de representación mientras que aquellos que pertenezcan a clases diferentes deberían estar en diferentes regiones del espacio de representación.

El estudio del conjunto apropiado de variables, la variabilidad de los patrones de una clase, las medidas de similaridad entre patrones y entre patrones y clases constituye el reconocimiento estadístico de patrones.

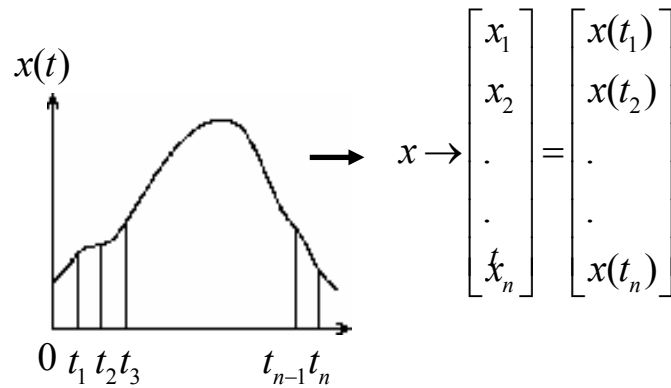


Figura 2.7 Patrones de naturaleza temporal

2.3.3 Modelos estocásticos

En la mayoría de los sistemas intervienen factores que no se pueden pronosticar con precisión. Para el estudio de estos sistemas, es conveniente medir la incertidumbre a través del uso de la probabilidad.

El estudio de sistemas donde está presente la incertidumbre ha dado lugar a los modelos estocásticos que tratan de explicar el funcionamiento de estos sistemas usando probabilidad [28].

Un proceso estocástico se define sencillamente como una colección indexada de variables aleatorias (X_t), donde el subíndice t toma valores de un conjunto T dado. Generalmente T se toma como el conjunto de enteros no negativos y X_t representa una característica de interés medible en el tiempo.

Dentro de los modelos estocásticos el más utilizado es el Modelo Oculto de Markov (Hidden Markov Model, HMM) el cual es un proceso estocástico que consta de un proceso de Markov no observado (oculto) q y un proceso observado O , cuyos estados son dependientes estocásticamente de los estados ocultos, es decir, es un proceso bivariado (q, O) . Los HMM se pueden considerar también como sistemas generativos estocásticos, los cuales se emplean para modelar series de tiempo [4].

2.3.4 Sistemas basados en conocimientos

El reconocimiento utilizando sistemas basados en conocimiento tiene el objetivo de incorporar la capacidad que tenemos para poder percibir patrones (como el llanto de bebés) en un algoritmo computacional [29]. El hecho de que el humano pueda inferir a partir del llanto del bebé su estado anímico o físico hace muy atractiva la idea de incorporar este tipo de sistemas en el reconocimiento de patrones. Aunque hasta el momento no existe un sistema de éste tipo que sea efectivo, esto puede deberse a la complejidad del cerebro humano en su habilidad de reconocer patrones.

2.3.5 Modelos conexionistas

Los modelos conexionistas o mejor conocidos como redes neuronales son estructuras de procesamiento paralelo de información, formadas por numerosos nodos simples conectados entre si mediante pesos y agrupados en diferentes capas. Debido a su naturaleza intrínsecamente no lineal, a su capacidad de clasificación y sobre todo a la capacidad que tienen para aprender una determinada tarea, se han convertido en una de las herramientas más atractivas para la solución del problema del reconocimiento de patrones [30]. Sin embargo, estos modelos presentan algunos inconvenientes como pueden ser: desconocimiento de la estructura de capas y número de nodos necesarios para cada problema; un tiempo, en algunas ocasiones, excesivamente elevado para su entrenamiento. Es debido a esto que se desarrollan combinaciones entre las diferentes técnicas como pueden ser: lógica difusa, algoritmos genéticos, etc. resultando así redes neuronales híbridas con el objetivo de aumentar cualidades y reducir sus defectos, obteniendo de esta manera muy buenos resultados dentro del reconocimiento de patrones.

2.4 Llanto del Bebé

El llanto del bebé, aunque muy limitado, es el único medio que tienen para comunicarse con el adulto y poder expresar algún estado ya sea de dolor, hambre, sueño, placer entre otros [8]. De acuerdo con médicos especialistas el llanto no solo contiene información acerca de su estado anímico, sino también de posibles patologías. Para poder analizar las muestras de llanto, primero hay que entender como se produce con el propósito de medir o extraer las principales características, para su análisis.

2.4.1 Fonación

La fonación es la producción de una onda sonora a partir de una fuente de sonido [31]. El llanto, como una manifestación sonora o acústica del lenguaje, se desarrolla en conjunto con otros órganos y funciones anatómicas por medio del aparato fonador.

2.4.2 Aparato fonador

Los órganos que intervienen en la producción del llanto son [32]:

- ✚ Boca (labios, lengua, paladar)
- ✚ Faringe (cuerdas vocales)
- ✚ Pulmones, diafragma (músculos abdominales)

El sonido que emiten los órganos del aparato fonador se amplifica en las cavidades de resonancia boca, garganta, fosas nasales y en el cráneo (Figura 2.8).

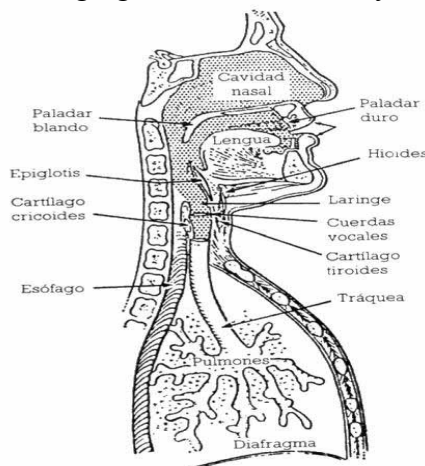


Figura 2.8 Aparato fonador

2.5 Generación del llanto

Se puede decir que la producción y emisión del llanto se deben a la acción o funcionamiento secuenciado, sincronizado y automático de los siguientes elementos:

- ✚ Una corriente de aire, la cual es producida por los pulmones y los músculos respiratorios.
- ✚ Un vibrador sonoro, constituido por las cuerdas vocales que se encuentran en la laringe.
- ✚ Un resonador, conformado por la boca, la nariz y la garganta (o faringe).
- ✚ Articuladores, conformado por los labios, dientes, paladar duro, velo del paladar, mandíbula.

Estos cuatro elementos generan los sonidos del llanto en el siguiente orden: en primer lugar, los pulmones suministran la columna de aire que, atravesando los bronquios y la tráquea, van a sonorizar las cuerdas vocales que se encuentran en la laringe. Es en la laringe donde propiamente se produce el llanto en su tono fundamental y sus armónicos; luego sufre una modificación en la caja de resonancia de la nariz, la boca y garganta (tracto vocal), en la que se amplifica y se forma el timbre de voz. Los órganos articuladores (labios, dientes, paladar duro, velo del paladar, mandíbula) van finalmente a moldear esa columna sonora, transformándola en sonidos. Este modelo muestra las partes principales del aparato fonador vinculadas con la producción del llanto (Figura 2.9). Los pulmones funcionan como fuente de energía acústica. La corriente de aire se desplaza por la tráquea y es modulada en las cuerdas vocales que vibran haciendo las veces de un oscilador [33].

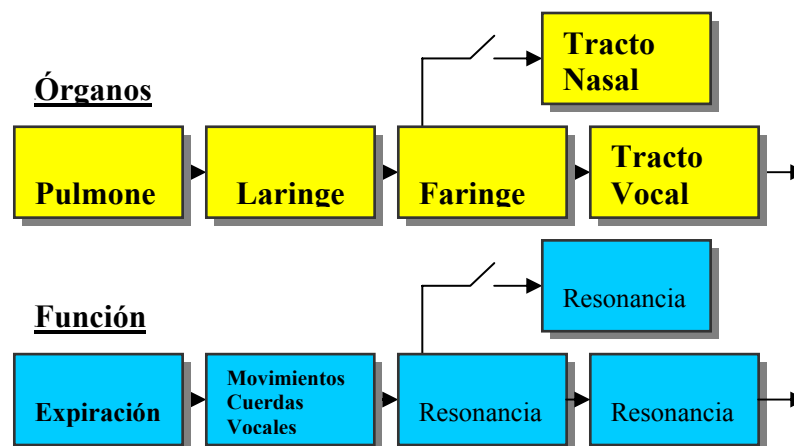


Figura 2.9 Diagrama del aparato fonador

2.5.1 Modelo del llanto

El llanto, como señal acústica, se produce a partir de las ondas de presión que salen de la boca y las fosas nasales de un bebé. Para tener un modelo exacto sería necesario modelar a los sonidos nasales, complicando en gran medida las matemáticas necesarias para representar dicho modelo. Por lo cual, en la Figura 2.10 se presenta un modelo basado en sonidos bucales únicamente. Es imprescindible señalar que los parámetros del modelo varían con el tiempo, ya que los parámetros articulatorios que pretendemos emular varían también con el tiempo, en función de los sonidos emitidos. Así, nuestro modelo será no estacionario, aunque se supone generalmente que si el intervalo de tiempo considerado es lo suficientemente pequeño (del orden de 10 ms, por ejemplo), la señal es estacionaria, con lo que los parámetros no variarían en dicho intervalo y el modelo pasaría a ser cuasi-estacionario, con las consiguientes ventajas en cuanto a sencillez [34].

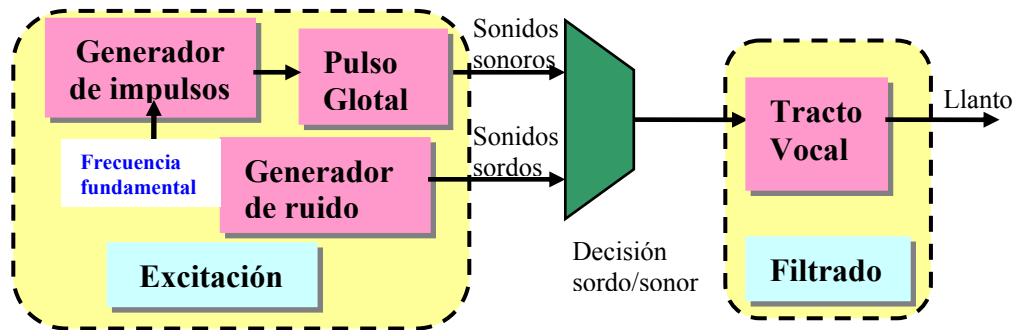


Figura 2.10 Modelo del llanto

2.6 Parámetros fundamentales del llanto

Usualmente, el llanto es generado por algún estímulo como dolor, hambre o alguna incomodidad, al analizar el llanto se extraen ciertas características las cuales contienen información que describen el llanto para su posterior clasificación. Estas características se describen a continuación.

2.6.1 Intensidad

Es el promedio de energía en una muestra de sonido por unidad de área. Esta se produce por la cantidad de aire y la presión subglotal que obligan a las cuerdas vocales a separarse. Como la señal de llanto no es estacionaria, la intensidad se define en función del tiempo [34].

2.6.2 Frecuencia fundamental

También conocida como tono o pitch, es la frecuencia con la que vibran las cuerdas vocales. Si suponemos que inicialmente las cuerdas vocales están juntas, la presión subglotal se incrementa lo suficiente para forzar a las cuerdas vocales a separarse. Al separarse, el aire pasa a través de ellas y la presión subglotal disminuye, momento en el que la fuerza de los músculos hace que las cuerdas vocales vuelvan a juntarse. Cuando las cuerdas vocales se juntan, el flujo de aire disminuye y la presión subglotal aumenta de nuevo, con lo que se vuelve a reproducir el ciclo, y la frecuencia de vibración de las cuerdas es conocida como frecuencia fundamental (F_0). Los valores típicos de F_0 para un niño están alrededor de los 300Hz [34]. La F_0 puede ser obtenida en el dominio del tiempo al medir el periodo de la onda del llanto del bebé o en el dominio de la frecuencia [35].

2.6.3 Formantes

Son las frecuencias amplificadas por el tracto vocal y tracto nasal. También conocidas como frecuencias de resonancia [35]. Normalmente se toman en cuenta los primeros tres, para el llanto de bebé el primer formante se encuentra entre los 800-1400 Hz, el segundo formante en el rango de 1800 y 3000 Hz, el tercero normalmente se encuentra entre los 3500 a 5000 Hz [2].

2.6.4 Unidades de llanto

Se pueden describir las unidades de llanto como la información que se encuentra entre el proceso de inspiración y expiración durante el llanto de un bebé [2]. No existe hasta el momento un programa que sea capaz de seleccionar las unidades de llanto automáticamente, esto se debe a que son de duración variable y solo una persona a través de la experiencia puede reconocer el inicio y final de una unidad de llanto ayudándose de programas que permitan reproducir la muestra en forma audible y que pueda desplegar en pantalla un espectrograma de la muestra analizada. Un ejemplo de la selección de unidades de llanto en una muestra se puede ver en la Figura 2.11. En el Apéndice A se muestran algunos resultados obtenidos utilizando la extracción de unidades de llanto durante el procesamiento acústico de la señal.

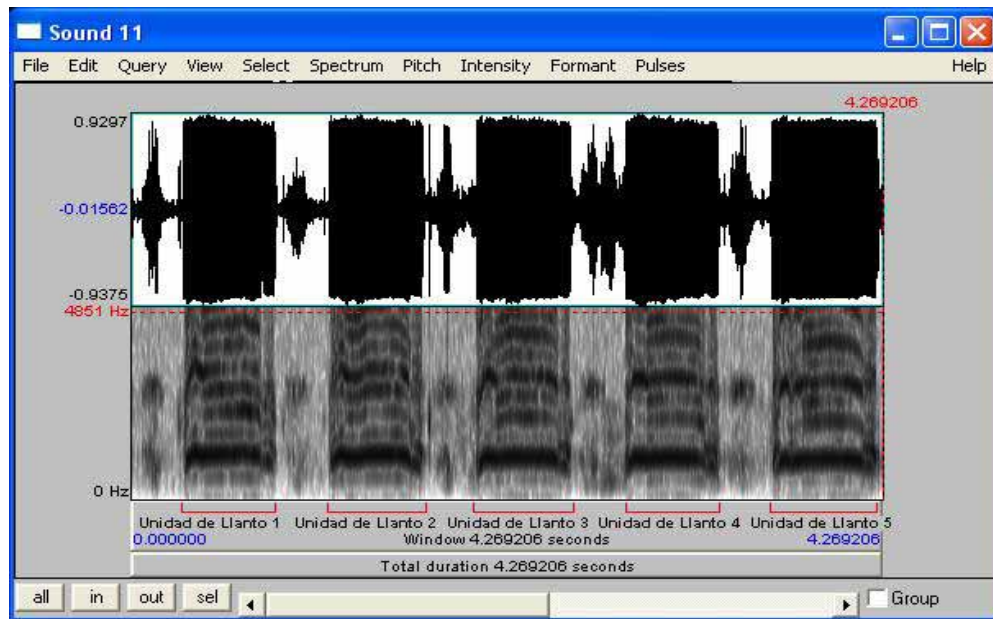


Figura 2.11 Unidades de llanto.

2.6.5 Coeficientes de predicción lineal

Los Coeficientes de Predicción Lineal (LPC), proveen una estimación de la función de transferencia del tracto vocal. El algoritmo para la extracción de los LPC consiste en construir un modelo lineal de la voz en un intervalo de tiempo y parametrizarla en dicho intervalo, reduciendo así la cantidad de bits que se necesitan para almacenar o transmitir la voz; ya que la voz no se representa utilizando la señal digitalizada, sino los parámetros calculados [36]. El principio de un LPC es que el valor actual de una muestra de señal puede predecirse a partir de un número finito de muestras anteriores [37].

2.6.6 Coeficientes cepstrales en frecuencia de Mel

Los cepstrum formalmente se definen como la transformada inversa del logaritmo del módulo de la transformada de la señal [38]. Los Coeficientes Centrales en Frecuencia de Mel (MFCC), son unos componentes cepstrum obtenidos basándose en las características perceptuales, es decir similar al funcionamiento del oído. Pueden obtenerse los cepstrum de la señal de llanto filtrándola a través de diferentes estados [38]. Este proceso pretende extraer características simulando el filtro que realiza el oído humano (el oído es más sensible a unas frecuencias que a otras).

Capítulo 3 RECONOCIMIENTO DEL LLANTO DE BEBÉ

Ya que no existe un modelo específico para el reconocimiento del llanto de bebés se parte del proceso de Reconocimiento Automático del Habla (RAH) para poder realizarlo, el cual usualmente se realiza en dos pasos. El primer paso es conocido como procesamiento de la señal teniendo como resultado la extracción de características, mientras que el segundo se conoce como clasificación de patrones.

El proceso del reconocimiento del llanto del bebé (Figura 3.1) es básicamente un problema de extracción y procesamiento de patrones, similar al proceso de Reconocimiento Automático del Habla (RAH). El objetivo es tomar la onda de llanto del bebé y a partir de éste extraer las características que aporten mayor información para obtener un patrón de entrada, al final detectar el tipo de llanto o patología que el bebé presenta. En el proceso del reconocimiento automático del llanto del bebé se tienen cuatro etapas: análisis acústico, extracción de características, reducción de las mismas y clasificación de patrones. Esta última parte es en la que esta enfocada la tesis.

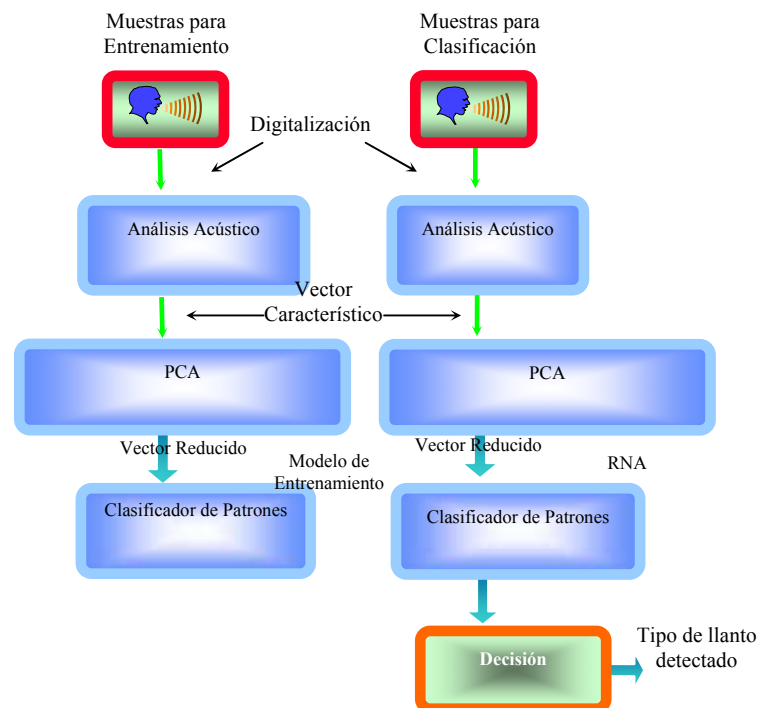


Figura 3.1 Proceso de reconocimiento del llanto de bebés

3.1 Procesamiento de la señal

La etapa del procesamiento de la señal implica realizar un filtrado a la muestra de entrada, segmentación de la señal, normalización, extracción de características acústicas, y reducción de datos.

En el análisis acústico o extracción de características, la onda del llanto del bebé es digitalizada a través de una grabadora digital. Una vez que se han grabado varias muestras de llanto, éstas se normalizan y se organizan para formar una base de datos. Los registros de los llantos son segmentados en señales de igual longitud. Las características más relevantes son obtenidas de la onda de llanto de los niños para utilizarlas posteriormente en el clasificador. Previamente se depura la señal con el fin de eliminar la información irrelevante e indeseable como el ruido de fondo, distorsión del canal, y características particulares de la señal. El conjunto de parámetros puede ser reducido al remover componentes repetitivos, es decir, que se conservan las características con la mayor varianza y se eliminan las que varían muy poco o nada. El conjunto de características obtenidas puede ser representado como un *vector*, y cada vector puede ser tomado como un patrón. El vector de parámetros en la mayoría de los casos resulta muy grande, por lo que necesita ser reducido, sin perder información que pueda ser útil en el reconocimiento. El vector de características reducido, con su etiqueta del tipo de llanto, es utilizado como un patrón de entrenamiento para el clasificador. Después, un nuevo conjunto de vectores reducidos de características es tomado por el clasificador entrenado, el cual compara cada vector con el conocimiento que tiene el clasificador para tomar una decisión de clasificación.

3.2 Base de datos de las muestras de llanto

El conjunto total de muestras consideradas en este trabajo es de 165 muestras con diversos tiempos de duración, las cuales han sido recolectadas de 88 bebés diferentes por médicos especialistas, con la grabadora de sonido digital ICD-67 Sony, éstas se tomaron a 8000 muestras por segundo (mps). Los médicos que realizaron dichas grabaciones, al final de cada una las etiquetaron en normal, sordo y asfixia; algunas

muestras también proporcionan el motivo del llanto tales como el dolor, hambre o incomodidad entre otras. Los bebés seleccionados para tomar las muestras son de hasta 6 meses de edad, sin tomar en cuenta el género. Las 165 muestras de las tres clases, se segmentaron en fragmentos de un segundo cada uno, obteniéndose 2,268 segmentos de las cuales, 1,049 son de bebés normales, 879 de bebés sordos y 340 bebés con asfíxia. Para realizar el experimento, se tomaron 340 muestras de cada clase debido a que es el número menor de muestras y corresponde a la clase de asfíxia, ésta cantidad se dividió en 5 grupos, donde se separaba un grupo y se entrenaba con lo demás, es decir 272 muestras de cada clase para el entrenamiento y 68 muestras para su clasificación.

Las muestras que se presentan en las Figuras 3.2, 3.3 y 3.4 corresponden al llanto causado por hambre, en donde se pueden apreciar las diferencias que existen entre las diferentes clases, las mismas que el clasificador debe de aprender a diferenciar.

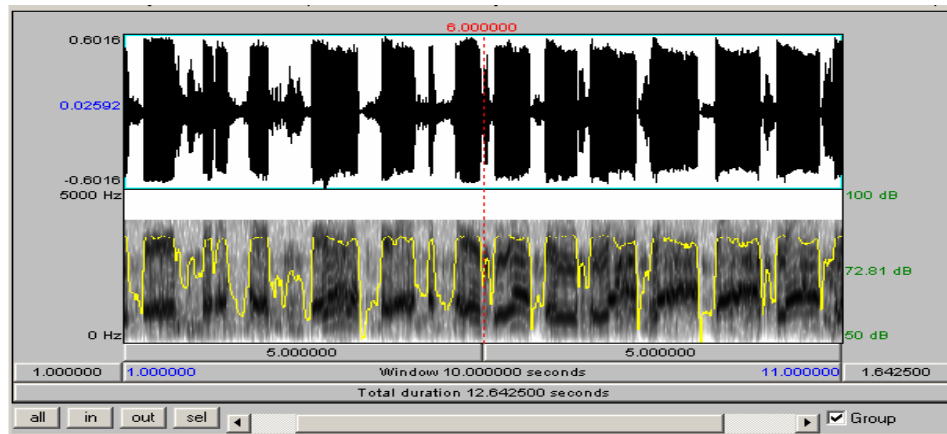


Figura 3.2 Forma de onda y espectrograma del llanto de un bebé normal

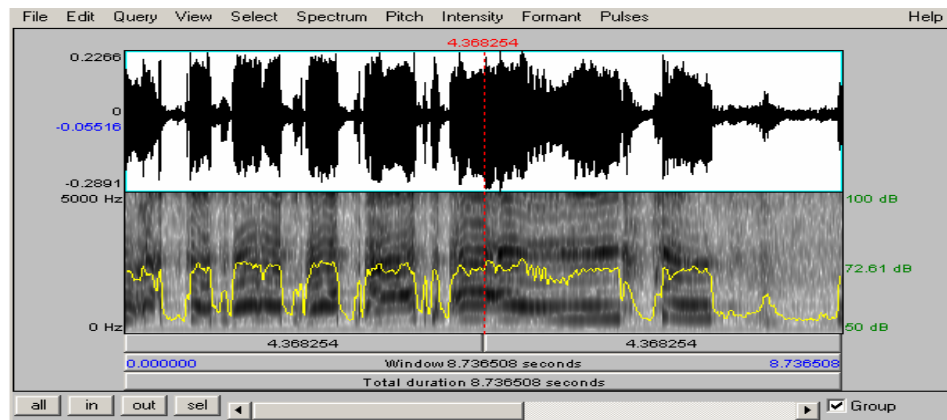


Figura 3.3 Forma de onda y espectrograma del llanto de un bebé sordo

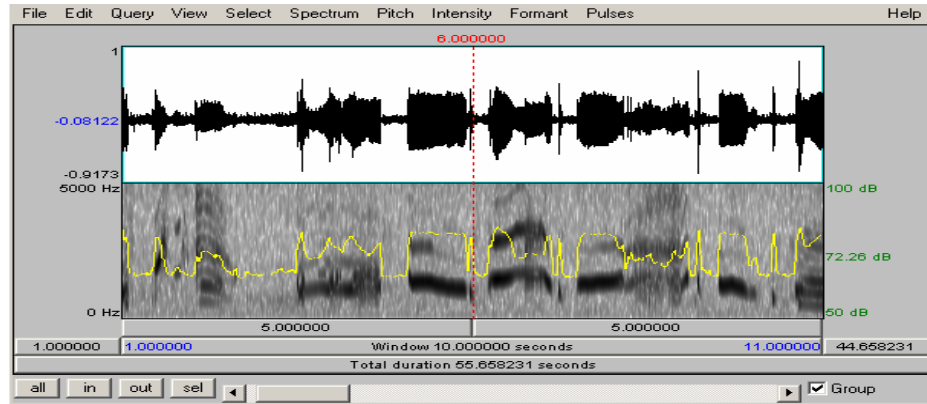


Figura 3.4 Forma de onda y espectrograma del llanto de un bebé con asfixia

La señal de audio fue digitalizada y filtrada dejando pasar solo las frecuencias menores a 8 kHz con el software Microsoft Sound Recorder en canal mono-aural a 8 bits y muestreada a una frecuencia de 16 kHz de acuerdo al *Criterio de Nyquist* [39]. La frecuencia de muestreo pudiera ser mayor, el problema que esto presenta es que el espacio de almacenamiento que se requiere también tendría que ser más grande.

3.3 Análisis predictivo lineal

Los vectores de características se obtuvieron utilizando los Coeficiente de Predicción Lineal (LPC). El número de coeficientes depende del tamaño de la ventana seleccionada, así como del número de coeficientes que se obtengan de cada una de ellas y de la duración en tiempo de la muestra.

Un método muy poderoso para el análisis del habla está basado en *codificación predictiva lineal*, [40] también conocido como análisis LPC, o modelo auto-regresivo. Este método es ampliamente usado en RAH porque es rápido, simple y efectivo para estimar los parámetros principales de la señal.

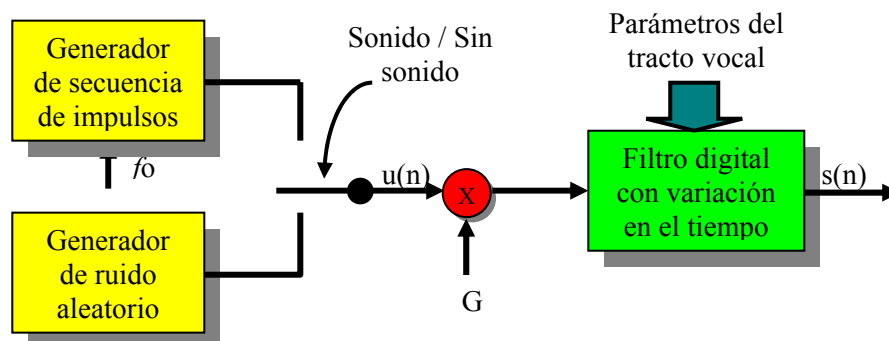


Figura 3.5 Diagrama a bloques del modelo simplificado del habla

El modelo de producci n del habla (Figura 3.5) correspondiente al an lisis LPC representa la funci n de excitaci n en el habla incluyendo una serie de impulsos o una fuente de ruido aleatorio. El sistema es excitado por una secuencia de impulsos para la se al con voz o una secuencia de ruido aleatorio para la se al sin voz. As , los par metros de este modelo son: clasificaci n de voz / sin voz, frecuencia para se al con voz, par metro de ganancia G y los coeficientes $\{\alpha_k\}$ del filtro con variaci n en el tiempo.

Las principales ventajas de los LPC son la precisi n de las estimaciones obtenidas y su rapidez de c lculo. La idea b sica de este tipo de an lisis es la suposici n de que el valor de una muestra de la se al puede ser predicho en base a una combinaci n lineal de los valores de muestras anteriores. Minimizando el error entre el valor de la muestra y la predicci n se puede obtener un conjunto  nico de valores para los coeficientes de la combinaci n lineal que refleja las caracter sticas de la se al [37].

El m todo LPC parte del postulado de que la se al de voz es producida por un zumbador al final de un tubo. La glotis (el espacio entre las cuerdas vocales) produce el zumbido que es identificado por su intensidad y frecuencia (pitch). El tracto vocal (la garganta y boca) forma el "tubo" que queda caracterizado por sus resonancias que se llaman formantes. Los LPC realizan el an lisis de la se al de voz mediante una estimaci n de los formantes, eliminando sus efectos de la se al de voz y estimando la intensidad y frecuencia del zumbido restante. El proceso de quitar el efecto de los formantes recibe el nombre de filtrado inverso y la se al restante se llama el residuo

Existen diferentes formas de obtener los LPC, los m todos m s comunes son los de covarianza y autocorrelaci n. El m todo utilizado en este trabajo es el de autocorrelaci n LP, debido a que presenta muy buenos resultados y su implementaci n ser a m s sencilla debido a que solo se realizan sumatorias, la funci n de autocorrelaci n para una se al de entrada $s(n)$ est  definida por [41]:

$$a = \sum_{n=0}^{N-1-l} s(n)s(n+1) \quad (l \geq 0) \quad (1)$$

donde, l es el retardo entre la se al actual y la muestra anterior, N es el n mero total de muestras.

La señal predicha $\hat{s}(n)$ esta dada por:

$$\hat{s}(n) = \sum_{i=0}^M a_i s(n+i) \quad (a_0 = 1) \quad (2)$$

for $n = 0, 1, \dots, N+M-1$

Mientras el orden M del modelo LP se incrementa, más detalles del espectro de la señal pueden ser aproximados. En nuestro caso se obtienen 16 LPCs de cada ventana. La señal de llanto original es segmentada en muestras de un segundo cada una, donde se obtuvieron 19 ventanas de 50ms con 16 coeficientes cada uno, resultando 304 coeficientes LPC por muestra de un segundo de llanto. Como se puede observar cada vector de entrada cuenta con 304 coeficientes y una etiqueta describiendo si el llanto pertenece a un bebé normal o patológico. En total cada vector de entrada contiene 305 elementos, los cuales deberán ser reducidos tratando de mantener la mayor cantidad de información del vector original.

3.4 Análisis de componentes principales

El Análisis de Componentes Principales o mejor conocido como “Principal Component Analysis” (PCA), tiene como objetivo reducir la dimensión de un conjunto d de variables a un conjunto m menor, para mejorar el manejo de los datos. En muchas situaciones prácticas se obtienen un gran número de variables correlacionadas es decir, el conjunto de variables o características originales en ocasiones son redundantes, por lo que es útil reducir la dimensión de los datos originales, sin perder información relevante.

El método PCA trata de hallar componentes (factores) que sucesivamente expliquen la mayor parte de la varianza total. Ésta técnica tiene tres efectos: ortogonaliza los componentes (parámetros) de vectores de entrada; ordena los componentes ortogonales resultantes (componentes principales) para que aquellos que tengan una variación mayor estén primero; y elimina aquellos componentes que contribuyen menos en la variación en el conjunto de datos. En el Análisis de Componentes Principales, el primer factor o componente sería el que contiene una mayor parte de la varianza total, el segundo factor sería el que tiene la mayor parte de la varianza restante, es decir, de la que no contenía el primero y así sucesivamente.

Si se trataran de analizar el conjunto total de datos extra dos de la se al de llanto sin reducir, ser a dif cil y lento el proceso de clasificaci n, por la gran cantidad de coeficientes obtenidos, es por esa raz n, que se debe reducir el conjunto de datos para facilitar al clasificador su tarea y ahorrar tiempo en el procesamiento. Los PCA han arrojado resultados satisfactorios en trabajos previos [3].

3.4.1 Algoritmo

El PCA se basa en representaciones estad sticas de una variable aleatoria. El objetivo es encontrar los *vectores propios* (eigenvectores) de la matriz de covarianza. Estos *eigenvectores* corresponden a las direcciones de los componentes principales de los datos originales, su importancia estad stica es dada por sus correspondientes *valores propios* (eigenvalores). En m s detalle, esta t cnica puede estructurarse como sigue:

Tomando el modelo PCA:

$$Y = W \cdot X \quad (3)$$

Donde Y es el vector de m elementos que se desea extraer del vector original X de tama o d .

El algoritmo de c lculo de W puede plantearse en 4 pasos:

1. Calcular la matriz de covarianza global Σ_X .
Para este c lculo se utilizan todos los patrones X .
2. Calcular los eigenvalores de Σ_X , $\lambda_1, \lambda_2, \dots, \lambda_d$.
3. Calcular los eigenvectores $\Phi_1, \Phi_2, \dots, \Phi_d$ asociados a $\lambda_1, \lambda_2, \dots, \lambda_d$.
4. Formar la matriz $W = [\Phi_1, \Phi_2, \dots, \Phi_d]$

Una vez formada la matriz de transformaci n W se procede a calcular los nuevos patrones, Y , a partir de cada X . Como $Y = W^T X$, y la matriz W es la matriz formada por los autovectores de Σ_X , sustituyendo en (3) se obtienen:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_m \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1d} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{d1} & \phi_{d2} & \cdots & \phi_{dd} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_d \end{bmatrix} \quad (4)$$

o lo que es lo mismo,

$$\begin{aligned}
 Y_1 &= \phi_{11}X_1 + \phi_{12}X_2 + \phi_{d1}X_d \\
 Y_2 &= \phi_{21}X_1 + \phi_{22}X_2 + \phi_{d2}X_d \\
 &\vdots \quad \quad \quad \vdots \\
 Y_d &= \phi_{d1}X_1 + \phi_{d2}X_2 + \phi_{dd}X_d
 \end{aligned} \tag{5}$$

Este m todo de reducci n se utiliza tanto para las muestras de entrenamiento como para las de prueba.

Capítulo 4 CLASIFICACIÓN DE PATRONES

4.1 Redes neuronales

Las redes neuronales artificiales (RNA) o “modelos conexionistas” son inspirados de modelos de neuronas biológicas humanas. Las redes neuronales se caracterizan por tener la capacidad de aprender y representar la información necesaria para desarrollar habilidades de reconocimiento sin instrucciones explícitas para ello. Las RNA son tan solo aproximaciones simplificadas de los modelos biológicos y han demostrado tener buen desempeño en clasificación con presencia de ruido [42].

La arquitectura de toda red se basa en unidades de procesamiento, cada unidad se comunica mediante el envío de señales a las demás unidades vía conexiones ponderadas. Las RNA (Figura 4.1) cumplen con los aspectos de un modelo distribuido, que son los siguientes [43]:

- ⊕ Una entrada externa.
- ⊕ Conexiones entre unidades (Pesos).
- ⊕ Un conjunto de unidades de Procesamiento.
- ⊕ Una función de activación.
- ⊕ Una regla de aprendizaje.
- ⊕ Una regla de propagación.
- ⊕ Un estado de activación por unidad (salida).

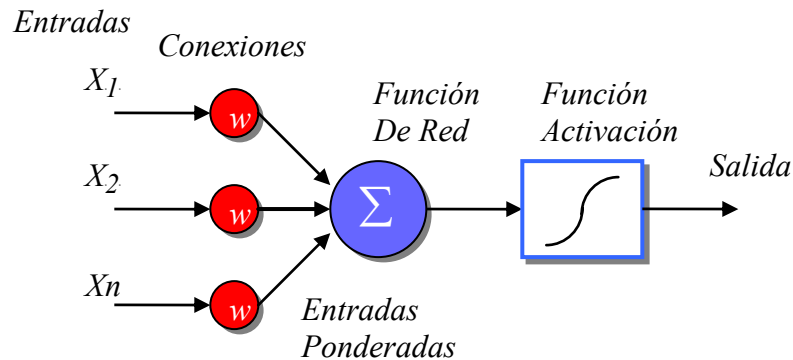


Figura 4.1 Modelo de una neurona artificial

La tarea de las unidades (Figura 4.2), es recibir una entrada del exterior o de otras unidades y generar una salida que se propagará a las otras unidades, la otra tarea es el ajuste de los pesos. *Entradas*

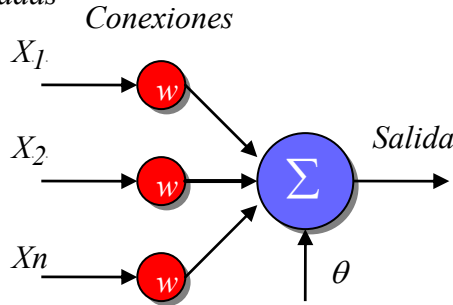


Figura 4.2 Neurona artificial simple

La salida total de una unidad es simplemente la suma de los pesos de las salidas de las unidades conectadas más el sesgo o corrimiento θ .

$$Salida = \sum_{i=0}^n w_i x_i - \theta \quad (6)$$

La salida de la ecuación 6 es pasada a través de una función de activación la cual se encarga de calcular el nivel o estado de activación de la neurona en función de la entrada total [43]. Entre las funciones de activación más comunes se encuentran:

- ⊕ Limitador Duro.
- ⊕ Umbral Lógico.
- ⊕ Función sigmoideal.

La función sigmoideal es probablemente la función de activación más empleada en la actualidad (ecuación 7).

$$\sigma(y) = \frac{1}{1 + e^{-y}} \quad (7)$$

Se trata de una función continua no lineal. La función sigmoideal posee un rango comprendido entre 0 y 1 (Figura 4.3). Esto, aplicado a las unidades de proceso de una red neuronal artificial significa que, sea cual sea la entrada, la salida estará comprendida entre 0 y 1.

Esta función depende del parámetro y , el cual decide el estado de la neurona.

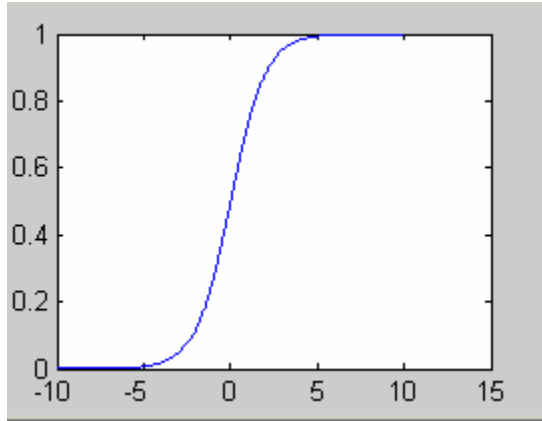


Figura 4.3 Función de activación sigmoideal

La topología de la red se refiere al número de capas (layers) y la cantidad de nodos en cada capa [44]. Los principales tipos de capas son:

- ⊕ Capa de Entrada: Reciben datos del exterior de la RNA.
- ⊕ Capas Ocultas: Cuyas entradas y salidas permanecen en la RNA.
- ⊕ Capa de Salida: Envían datos fuera de la RNA.

Dependiendo de las interconexiones entre las unidades y la propagación de los datos, existen al menos 2 topologías de red que se pueden distinguir [44]:

- ⊕ Redes en Cascada (Feed-Forward), donde el flujo de datos va desde las unidades de entrada hasta las unidades de salida. El procesamiento de los datos se puede extender a varias capas pero sin conexiones que puedan alimentar a la red.
- ⊕ Redes Recurrentes (feed-back), son las que estrictamente tienen conexiones de retroalimentación, aquí las características dinámicas de la red son importantes.

La conectividad se refiere al modo en que están conectados los nodos, se dice que hay conectividad completa cuando cada nodo en una capa esta conectado con cada nodo de la capa adyacente. Los pesos de las conexiones, regularmente, son ajustados durante el entrenamiento, pero algunos de ellos pueden permanecer fijos según los requerimientos del modelo [43]. En la Figura 4.4 se puede observar una red en cascada, mejor conocida como feed-forward de 3 capas totalmente interconectada, la cual fue implementada y los resultados obtenidos se muestran en el Apéndice B.

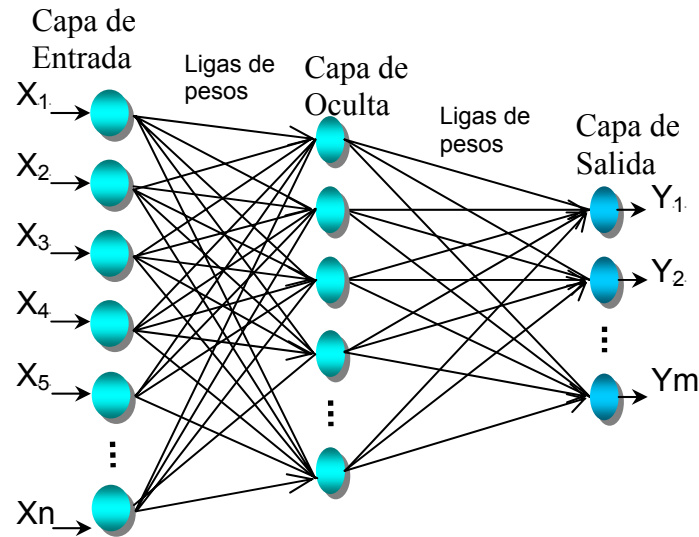


Figura 4.4 Red tipo feed-forward completamente interconectada

4.1.1 Entrenamiento

Una red neuronal debe ser configurada de tal manera que la aplicación de un conjunto de entradas produzca una salida deseada. Para tal tarea se pueden tener dos grandes formas de hacer el entrenamiento:

- ⊕ Entrenamiento Supervisado o Asociativo, donde la red se entrena dándole ejemplos y empatándolos con patrones de salida.
- ⊕ Entrenamiento No Supervisado o Auto-Organización, donde una unidad de salida es entrenada para responder a grupos de patrones de entrada. Aquí no se saben las clases a priori de los patrones a ser clasificados, la red hace sus propias clases en base a los estímulos.

Se puede resumir el aprendizaje, al ajuste de los pesos de las conexiones entre unidades, la idea básica es que si dos unidades son activadas simultáneamente, su interconexión debe ser reforzada. La regla de Aprendizaje de Hebbian se muestra a continuación, si una unidad “ j ” recibe una entrada de “ k ”:

$$\Delta w_{jk} = \gamma \cdot y_j \cdot y_k \quad (8)$$

donde γ es la tasa de aprendizaje. La regla de aprendizaje es la diferencia entre la activación actual y la deseada para el ajuste de pesos:

$$\Delta w_{jk} = \gamma \cdot y_j (d_k - y_k) \quad (9)$$

la ecuación anterior es llamada regla delta o la regla de Widrow-Hoff.

4.2 Conjuntos difusos

Un conjunto es tan solo una colección de elementos. Todo el tiempo los utilizamos cuando nos referimos a un objeto x y a un conjunto A , identificando si el objeto x pertenece o no al conjunto A [45]. La función de membresía (frecuentemente llamada función característica en conjuntos convencionales) es expresada como:

$$\mu(A) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{otros casos} \end{cases} \quad (10)$$

Por otro lado, en los problemas de la vida diaria las clases están mal definidas; generalmente, se sobreponen, o son difusas. Un patrón puede pertenecer a más de una clase con un grado de membresía diferente de cero. En la teoría de conjuntos difusos, un conjunto difuso A tiene una función característica de membresía que toma sus valores en el intervalo $[0,1]$. $\mu_A(x)$ es el grado de membresía de $x \in R$ en A . Donde A es definido como

$$A = \{(R, \mu_A(x))\}, \quad x \in R, \quad \mu_A(x) \in [0,1] \quad (11)$$

En un conjunto difuso la transición entre membresías es gradual en lugar de ser abrupta. Los grados de membresía reflejan un ordenamiento de los objetos en el universo inducido por el valor asociado en A . El valor del grado de membresía $\mu_A(x)$ de un objeto $x \in A$ puede ser interpretado como el grado de compatibilidad del valor asociado con A y con el objeto x [46].

Los valores de las variables lingüísticas son palabras u oraciones en un lenguaje natural. Un patrón lingüístico esta formado por variables lingüísticas. Una variable lingüística esta caracterizada por un quintuple $(H, T(H), U, G, M)$ en el cual H es el nombre de la variable; $T(H)$ es el conjunto de nombres de valores lingüísticos de H , donde cada valor es una variable difusa denotada genéricamente por X y con rango sobre un universo de discurso U que esta asociado con la variable base u ; G es una regla sintáctica para generar nombres, X , de valores de H ; y M es una regla semántica para asociar cada X con su significado, $M(X)$, el cual es un subconjunto difuso de U [47].

4.3 Relaciones matemáticas

La definición intencional de una relación $\mathbf{R}$ del conjunto x al conjunto y es como una oración abierta o incompleta, con dos espacios vacíos.

$$\mathbf{R} = \text{" ______ Es el tío de ______ "}$$

Cuando el primer espacio es llenado insertando un elemento x de X , y el segundo espacio insertando un elemento y de Y , resulta una proposición que puede ser cierta o falsa.

Si es cierta, se dice que x está en la relación $\mathbf{R}$ a y y sí $x \mathbf{R} y$ existe.

Si es falsa, se dice que x no está en la relación $\mathbf{R}$ a y y sí $x \mathbf{R} y$ no existe.

La definición de extensión está dada por el conjunto satisfacción. El conjunto satisfacción $\mathbf{R}_S$ de una relación $\mathbf{R}$ consiste en todos los pares (x, y) de elementos para los cuales $x \mathbf{R} y$, eso es

$$\mathbf{R}_S = \{(x, y) \in X * Y \mid x \mathbf{R} y\}$$

Este es claramente un subconjunto del producto cartesiano $X * Y$, que consiste en todos los pares posibles (x, y) . La relación también puede ser dada por medio de su matriz $\mathbf{R}_M$, con componentes [48].

$$R_{ij} = \begin{cases} 1, & \text{if } x_i \mathbf{R} y_j \\ 0, & \text{if } x_i \not\mathbf{R} y_j \end{cases}$$

Ejemplo:

$X =$ Gregorio, Jaime, Francisco

$Y =$ Ana, Lola, María, Sara

$\mathbf{R} =$ " ______ Asiste a clases con ______ "

$\mathbf{R}_S = \{(G, A), (G, M), (J, L), (J, M)\}$ Conjunto satisfacción.

$$R_M = \begin{array}{ccccc} & A & L & M & S \\ \begin{array}{c} G \\ J \\ F \end{array} & \begin{array}{c} 1 \\ 0 \\ 0 \end{array} & \begin{array}{c} 0 \\ 1 \\ 0 \end{array} & \begin{array}{c} 1 \\ 1 \\ 0 \end{array} & \begin{array}{c} 0 \\ 0 \\ 0 \end{array} \end{array} \text{ Matriz}$$

4.4 Productos relacionales

Sea $\mathbf{R}$ una relación de X a Y y $\mathbf{S}$ una de Y a Z , hay varias operaciones binarias aplicables sobre ellas; cada una resultando en una relación producto del conjunto X al conjunto Z .

4.4.1 Producto Círculo o Circlet ($\mathbf{R} \circ \mathbf{S}$)

El producto círculo para el cual x está en la Relación $\mathbf{R} \circ \mathbf{S}$ a z , si y sólo si hay al menos una y tal que $x \mathbf{R} y$ y $y \mathbf{S} z$:

$$x (\mathbf{R} \circ \mathbf{S}) z \Leftrightarrow \exists y \in Y \sqcup (x \mathbf{R} y \text{ y } y \mathbf{S} z) \quad (12)$$

Entonces $x (\mathbf{R} \circ \mathbf{S}) z$ si y sólo si hay un camino de x a z en la figura compuesta.

$$(\mathbf{R} \circ \mathbf{S})_{ik} = \max_j (\min(\mathbf{R}_{ij}, \mathbf{S}_{jk})) \quad (13)$$

Para la notación matricial $\mathbf{R}_M$ y $\mathbf{S}_M$

$$(\mathbf{R} \circ \mathbf{S})_{ik} = \bigvee_j (\mathbf{R}_{ij} \wedge \mathbf{S}_{jk}) \quad (14)$$

$$\text{Otros Operadores} \begin{cases} \triangleleft, & \text{Subtriángulo} \\ \triangleright, & \text{Supertriángulo} \\ \square & \text{Cuadrado} \end{cases}$$

4.4.2 Producto subtriángulo.

x está en la relación $\mathbf{R} \triangleleft \mathbf{S}$ a z sí, para cada y , $x \mathbf{R} y$ implica que $y \mathbf{R} z$. Para las matrices,

$$(\mathbf{R} \triangleleft \mathbf{S})_{ik} = \min(\mathbf{R}_{ij} \rightarrow \mathbf{S}_{jk}) \quad (15)$$

Donde $\rightarrow$ es el operador de implicación booleana dado en la Tabla 4.1.

p	q	$p \rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

Tabla 4.1 Implicación booleana $p \rightarrow q$.

Una implicación $p \rightarrow q$ es verdadera siempre que no se de el caso de que su antecedente p sea verdadero y su consecuente q falso.

4.4.3 Producto supertriángulo

Es un producto dual al anterior, dado por:

$$\begin{aligned}(R \triangleright S)_{ik} &= \min_j (R_{ij} \leftarrow S_{jk}) \\ &= (R^T \leftarrow S^T)^T_{ik}\end{aligned}\quad (16)$$

4.4.4 Producto cuadrado

Es esencialmente la intersección de los dos anteriores x esta en la relación $R \sqcap S$ con z , sí, para cada y , $x R y$ exactamente cuando $y S z$:

$$(R \sqcap S)_{ik} = \min_j (R_{ij} \leftrightarrow S_{jk}) \quad (17)$$

4.4.5 Ejemplo de aplicación de productos relacionales

X es un conjunto de pacientes.

Y es un conjunto de síntomas.

Z es un conjunto de enfermedades.

$x R y \Leftrightarrow$ El paciente x muestra el síntoma y

$y S z \Leftrightarrow$ Síntoma y es uno de los que caracterizan la enfermedad z .

$x R \circ S z \Leftrightarrow x$ tiene al menos uno de los síntomas de la enfermedad z .

$x R \triangleleft S z \Leftrightarrow$ Cada síntoma de x es uno de aquellos de la enfermedad z .

$x R \triangleright S z \Leftrightarrow x$ tiene todos los síntomas de la enfermedad z , y quizás más

$x R \sqcap S z \Leftrightarrow x$ tiene exactamente los síntomas de enfermedad z y no otros

4.5 Relaciones difusas

Una relación difusa es una que se establece entre dos elementos en algún grado entre 0 y 1 [48]. Así:

X es un conjunto de personas

Y es un conjunto de atributos o llaves

Z es un conjunto de objetos

$x R y$ se convierte en

"sujeto _____ posee la llave _____ al grado _____"

$y S z$ se convierte en

"la llave _____ es necesaria para acceder el objeto _____ al grado _____"

$x \mathbf{R} \circ \mathbf{S} z$	$\Leftrightarrow$	grado al cual sujeto x posee al menos una llave necesaria para acceder al objeto z .
$x \mathbf{R} \triangleleft \mathbf{S} z$	$\Leftrightarrow$	grado al cual las llaves de x están entre aquellas necesarias para acceder al objeto z .
$x \mathbf{R} \triangleright \mathbf{S} z$	$\Leftrightarrow$	grado al cual las llaves de x incluyen todas aquellas necesarias para acceder al objeto z .
$x \mathbf{R} \square \mathbf{S} z$	$\Leftrightarrow$	grado al cual las llaves de x son exactamente aquellas necesarias para acceder al objeto z .

4.6 Neurona difusa

Las metodologías neuronal y difusa pueden ser fundidas en una metodología unificada conocida como *Sistemas Neurodifusos* y *Sistemas Fuzzyneurales* en el primer caso se trata de un sistema difuso entrenado como una red neuronal, donde al final la parte *fuzzy* del sistema es la que se implementa, en el caso de los *Sistemas Fuzzyneurales* los parámetros de control en las neuronas y los pesos de las conexiones se remplazan por regiones difusas, y la red neuronal es la que funcionará, este último tipo de sistemas es el que se desarrolla en este trabajo.

Una neurona difusa tiene una arquitectura similar a la neurona clásica de MacCulloch Pitts como se ilustra en la Figura 4.5 y las principales diferencias son [49]:

1. En lugar de pesos escalares la neurona difusa utiliza conjuntos difusos.
2. La función de sumatoria recibe una medida central de los pesos difusos y suma de sus valores que son arbitrarios por un factor de atenuación o umbral.

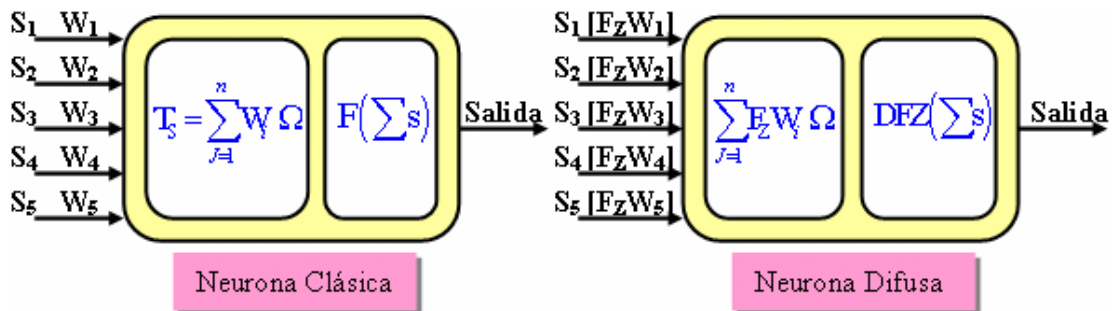


Figura 4.5 Arquitectura de una neurona clásica y una neurona Difusa

4.7 Red neuronal relacional

En este modelo, propuesto por Pedrycz, todo el proceso se realiza con operaciones de conjuntos difusos. El modelo se origina de la teoría de conjuntos difusos aprovechando las ventajas de manejo de problemas bajo condiciones de incertidumbre, junto con algunas estructuras relacionales básicas [49]. El algoritmo de aprendizaje utilizado es una modificación del método de retropropagación, adaptado para trabajar con valores difusos.

La notación utilizada es la siguiente. Sea $X = \{X_1, X_2, X_3, \dots, X_n\}$ un conjunto finito de nodos de un conjunto de nodos de entrada y $Y = \{Y_1, Y_2, Y_3, \dots, Y_m\}$ un conjunto de nodos de salida y se tiene la composición (producto relacional) Max-Min denotado por $X \bullet R$ que se aplica sobre un conjunto (Fuzzy) X y una relación (Fuzzy) R , y se obtiene otro conjunto (Fuzzy) Y tal que [49]:

$$Y = X \bullet Y \quad (18)$$

$$Y(y_j) = \max_{x_i} (\min(X(x_i), R(x_i, y_i))) \quad (19)$$

Se puede representar también como:

$$Y(y_j) = \bigvee_{x_i \in X} [X(x_i) \wedge R(x_i, y_i)] \quad (20)$$

En la Figura 4.6 se presenta una red neuronal con dos capas, de n nodos de entrada y m nodos de salida.

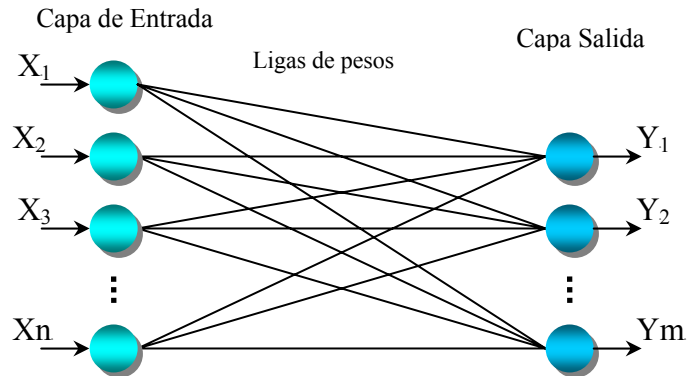


Figura 4.6 Red neuronal con dos capas

Para la operación de una red neuronal difusa el primer paso es *fuzzyficar* la entrada, esto es obtener para cada coeficiente de entrada su valor al conjunto difuso correspondiente de membresía de acuerdo a una serie de reglas previamente establecidas (Figura 4.7).

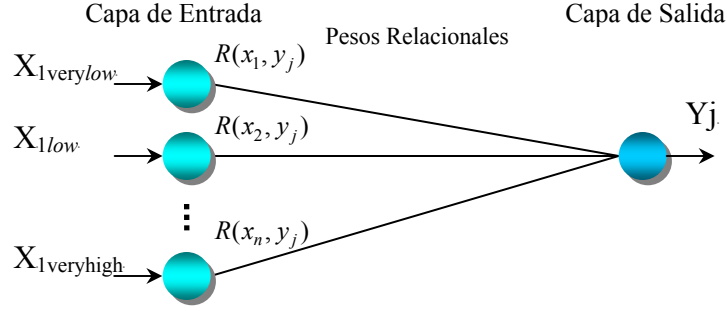


Figura 4.7 Red de una entrada simple fuzzy asociada a un nodo de salida

El producto relacional de la red difusa esta dada por $R: X \bullet R \rightarrow [0,1]$ describiendo todas las ligas entre los nodos de entrada y salida para una red como la Figura 4.6 ya que se han obtenido los valores de membresía para formar el vector de entrada. Para realizar este producto se utiliza la composición max-min representada en notación vector-matricial.

$$[X(x_1), X(x_2), \dots, X(x_n)] \bullet \begin{bmatrix} R(x_1, y_1) & \cdots & R(x_1, y_j) & \cdots & R(x_1, y_l) \\ \vdots & & \vdots & & \vdots \\ R(x_j, y_1) & \cdots & R(x_j, y_j) & \cdots & R(x_j, y_l) \\ \vdots & & \vdots & & \vdots \\ R(x_n, y_1) & \cdots & R(x_n, y_j) & \cdots & R(x_n, y_l) \end{bmatrix} \quad (21)$$

Donde X es un vector de activación para todos los nodos de entrada.

Incorporando un umbral a la ecuación (19) $u \in [0, 1]$ se obtiene

$$Y(y) = \max_{x_i} [\max(\min(X(x_i), R(x_i, y_i)), u(y))] \quad (22)$$

Esta relación es la representación básica de la red neuronal, en forma análoga a las redes neuronales típicas.

4.8 Descripción del reconocimiento de llanto utilizando la red

Fuzzy Neuronal Relacional

Se utilizó el mismo modelo empleado por Reyes [50], donde la red Fuzzy Neural Relacional (FNR), se compone básicamente de dos bloques: uno para el entrenamiento y otro para el procesamiento y clasificación una vez realizado el entrenamiento (Figura 4.8).

Para la aplicación propuesta, la sección de aprendizaje esta compuesta de cuatro bloques, nombrados Análisis Acústico (AA), PCA (reducción de datos), Estimador de la Salida Deseada (ESD), Entrenamiento de la Red Neuronal (ERN).

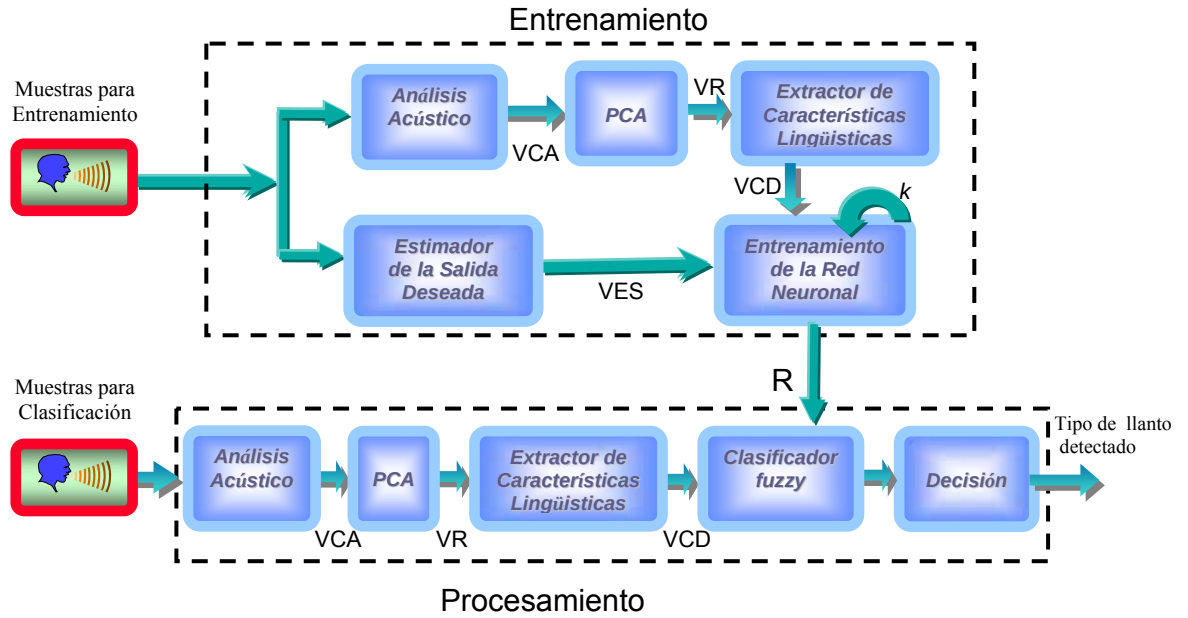


Figura 4.8 Diagrama a bloques del funcionamiento de la red FNR.

El proceso de aprendizaje inicia con la entrada de muestras para el entrenamiento. En el AA las muestras de entrada son filtradas y segmentadas de donde el resultado es el Vector de Características Acústicas (VCA). En nuestro ejemplo el número de coeficientes obtenidos para el VCA es de 304 por muestra, los cuales se reducen a 10 con el método de Análisis de Componente Principal (PCA). El Vector Reducido (VR) es pasado por el Extractor de Características Lingüísticas (ECL), el cual toma un vector con n elementos de VR y los convierte en la forma $N \times n$ -dimensional dependiendo del número de propiedades lingüísticas deseadas: $3n$, $5n$, $7n$ -dimensional, etc. (Figura 4.9). El vector resultante $N \times n$ -dimensional es llamado Vector de Características Difuso (VCD). El ESD toma el vector de entrada y calcula su función de membresía para cada clase k , de un total de muestras l para el entrenamiento, por lo cual el ESD genera un vector de tamaño $l \times k$ llamado Vector Estimado de Salida (VES), con el cual es inicializada la matriz de pesos de la red. El VFC y VES son utilizados para el entrenamiento de la red, donde el propósito es ajustar la matriz de pesos, para que sea capaz de clasificar los patrones de entrada.

La red FNR consiste de dos capas [50], una capa de entrada y una capa de salida. El número de neuronas en la capa de entrada correspondiente esta dado por $N \times n$ elementos, donde N es el tamaño del VR y n el número de propiedades lingüísticas. La capa de salida tiene k neuronas, donde k es el número de total de clases.

La red FNR esta totalmente interconectada, las ligas o conexiones son descritas por pesos relacionales almacenados en la matriz R (ecuación 21).

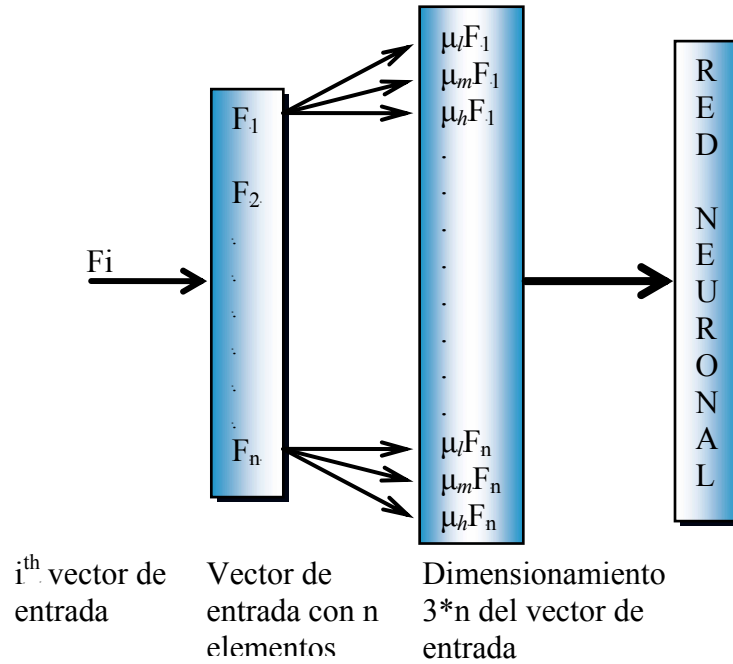


Figura 4.9 Diagrama a bloques de la obtención del Vector Final de Características.

4.9 Funciones de membresía

La asignación de la función de membresía para un conjunto difuso es subjetiva y aunque no existen reglas para realizarla, estas asignaciones no pueden ser hechas de manera arbitraria. En muchos casos se recomienda utilizar funciones estándares cuyos parámetros puedan ser ajustados ya que éstas pueden tener diferentes formas, y tanto el tipo de función como el número de las mismas son seleccionadas dependiendo de la experiencia del diseñador.

Después de realizar algunas pruebas se decidió emplear funciones triangulares y trapezoidales con tres, cinco y siete funciones de membresía para cada una. La descripción de ambas funciones se da a continuación.

4.9.1 Función de membresía triangular (Δ)

La función Δ es usada para obtener nuevos vectores de $N \cdot n$ a partir de los vectores de entrada, donde N es el número de membresías utilizadas y n el número de elementos que contiene cada vector de entrada.

La función Δ (Figura 4.10) con rango de μ $[0,1]$ y sea $x \in \mathfrak{R}$, esta definida como

$$\Delta(x:a,b,c) = \begin{cases} 0, & x < a \\ \frac{x-a}{b-a}, & a \leq x < b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & x > c \end{cases} \quad (23)$$

donde a y c es la base del triángulo y b es el centro.

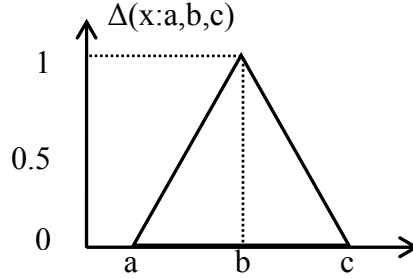


Figura 4.10 Función Δ

En la Figura 4.11 se observa un ejemplo con tres funciones de membresía donde cada una es expresada en términos lingüísticos, bajo, medio y alto. Las funciones se encuentran distribuidas uniformemente dentro del rango dinámico de los valores de entrada. Cada característica de entrada es pasada a través de estas funciones y se obtienen el grado de membresía para cada una de ellas, con este proceso un nuevo vector es obtenido con $3n$ elementos, donde n , es el número de características de entrada para cada $\bar{F}_i$. Se realizaron pruebas con cinco y siete funciones de membresía con el mismo grado de traslape del ejemplo que se presenta.

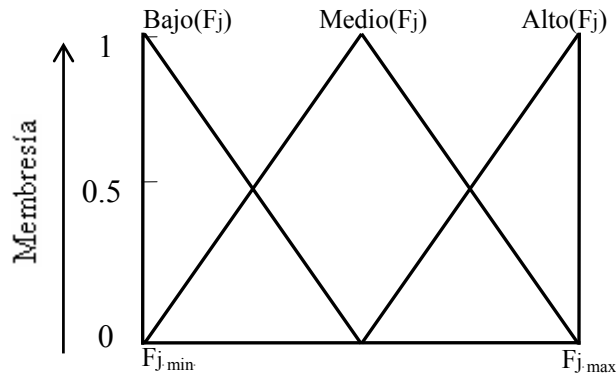


Figura 4.11 Representación de una estructura $3n$ -dimensional.

4.9.2 Función de membresía trapezoidal (T)

La función **T** es usada para obtener nuevos vectores de $N \cdot n$ a partir de los vectores de entrada, donde N es el número de membresías utilizadas y n el número de elementos que contiene cada vector de entrada [48].

La función **T** (Figura 4.12) con rango de μ $[0,1]$ y para $x \in \mathfrak{R}$, esta definida como

$$\Delta(x : a, b, \alpha, \beta) = \begin{cases} 1 - \frac{a-b}{\alpha}, & x \leq a, \alpha \leq 0 \\ 1, & a \leq x \leq b \\ 1 - \frac{x-b}{\beta}, & x \geq b, \beta > 0 \end{cases} \quad (24)$$

donde α y β son las extensiones izquierda y derecha de la función TD, y a y b son los puntos de referencia para calcular el valor máximo m (ejem., $\mu_m = 1$).

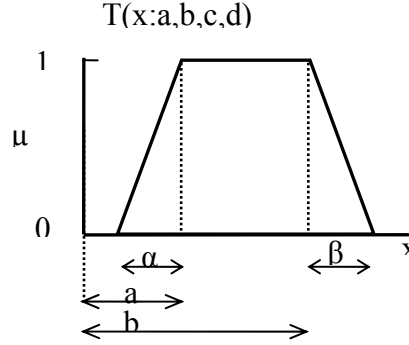


Figura 4.12 Función T

4.9.3 Función de membresía Trapezoidal Densa (T_D)

A partir de la función **T** se puede obtener la función de membresía Trapezoidal Densa (T_D), donde el espacio de características completo es dividido en $3n$, $5n$ y $7n$ subespacios sobrepuestos. Los cálculos de los valores de membresía de las características de entrada a las propiedades lingüísticas se hacen en una forma muy directa, como se da en las siguientes ecuaciones

$$\begin{aligned} X &= \frac{F_j}{D} * N, \\ i &= \text{int}(X), \\ d &= X - i, \end{aligned} \quad (25)$$

donde $D = F_j \max - F_j \min$, i es el entero más grande tal que $X - i \geq 0$, d es la parte decimal de X , y N es el número de las propiedades lingüísticas en el que se dividirá el espacio de las características.

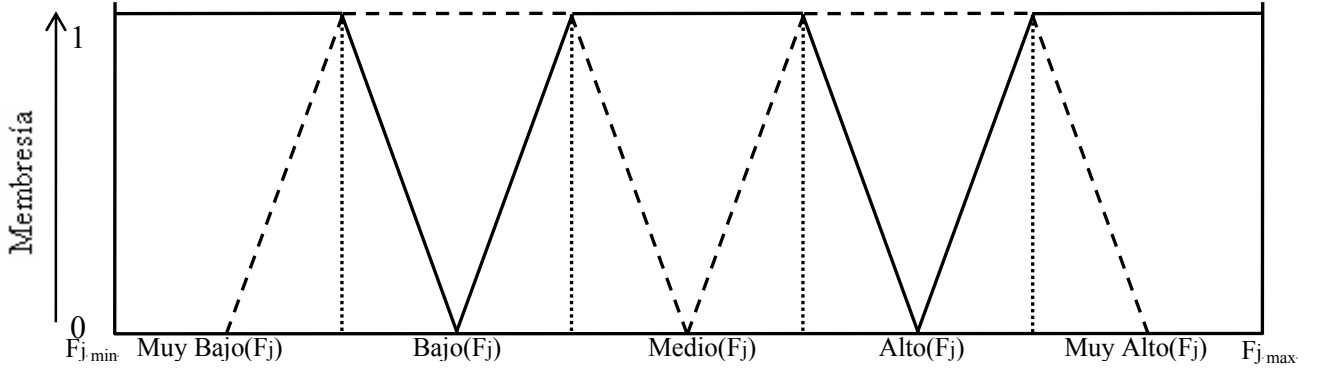


Figura 4.13 Estructura de la función TD con 5 propiedades lingüísticas

Un ejemplo para T_D con $N = 5$, donde F_j es representados por las propiedades lingüísticas *muy bajo*, *bajo*, *medio*, *alto*, *muy alto* (Figura 4.13). Donde las funciones se distribuyen de manera uniforme a través del rango dinámico de entrada.

4.9.4 Cálculo del valor deseado de salida

Para calcular el valor de salida deseada se toma cada vector de la muestra de entrenamiento y calcula su membresía a la clase que pertenece, en un problema con M clases. El vector resultante se conoce como Vector Deseado. A las clases a las que no pertenece el vector de entrada, se le coloca un valor muy pequeño casi cero.

1.- Se calcula una distancia ponderada del patrón F_j a la clase K en un problema con l clases [50].

$$Z_{ik} = \sqrt{\sum \left[\frac{F_{ij} - O_{kj}}{V_{kj}} \right]^2} \quad \text{Para } K=1,2,3,\dots,l \quad (26)$$

Donde:

O_{kj} = es la media

V_{kj} = es la desviación estandar de la característica j a la clase K

2.- Se calcula la membresía del vector i a la clase K .

$$\mu_k = \frac{1}{1 + \left(\frac{Z_{ik}}{fd} \right)^{fe}} \quad (27)$$

Donde:

fe = Generador Fuzzy exponencial.

fd = Generador Fuzzy denominacional.

4.10 Entrenamiento

El entrenamiento ocurre a través de la modificación de los pesos relacionales y de los umbrales de las neuronas que conforman la red. Para un entrenamiento supervisado, se presenta un vector F_i a la capa de entrada de la neurona y el resultado que la red arroja es comparado con una salida deseada, si la salida real y la salida deseada no son iguales, se procede a modificar los pesos relacionales de la red y se continua con este procedimiento hasta alcanzar el valor de salida deseado. Cuando la salida real se asemeja o es igual a la deseada se dice que la red convergió y se encuentra entrenada, para reconocer al vector F_i . Esto se repite para los i vectores de entrada que se utilicen durante el entrenamiento de la red [50].

4.10.1 Actualización de parámetros

Para una red relacional de n valores de entrada y l clases la salida real esta dada por:

$$y_i = f(x_i; r_k, u_k) = \left(\bigvee_{j=1}^n (r_{jk} \wedge x_{ij}) \right) \vee u_k \quad (28)$$

donde $r_k = [r_{j1}, r_{j2}, \dots, r_{ji}]$ es un vector que contiene todos los pesos relacionales de la red, $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]$ es el vector con los valores de entrada, y $u_k = [u_1, u_2, \dots, u_n]$ es el vector con los valores de umbral. Los parámetros r_m y u_m son actualizados en cada iteración por incrementos Δr_m y Δu_m resultado de las diferencias entre y_i y t_i , donde t_i es la salida deseada.

Una manera de obtener los incrementos Δr_m y Δu_m donde $m = 1, 2, 3, \dots, l$ es por medio de las siguientes derivadas

$$\Delta r_m = \frac{\partial Q}{\partial r_m} = \begin{bmatrix} \frac{\partial Q}{\partial r_1} \\ \frac{\partial Q}{\partial r_2} \\ \vdots \\ \frac{\partial Q}{\partial r_n} \end{bmatrix} \quad \Delta u_m = \frac{\partial Q}{\partial u_m} \quad (29)$$

De estas derivadas se obtienen las siguientes ecuaciones para Δr_m

$$\Delta a_m = \frac{\partial Q}{\partial r_m} = \sum_{i: y_i < t_i} \frac{\partial f(x_i; r_m, u_m)}{\partial r_m} - \sum_{i: y_i > t_i} \frac{\partial f(x_i; r_m, u_m)}{\partial r_m} \quad (30)$$

$$\frac{\partial f(x_i; r_m, u_m)}{\partial r_m} = \frac{\partial}{\partial r_m} \left[\left(\bigvee_{j=1}^n (r_{jm} \wedge x_{ij}) \vee u_m \right) \right] \quad (31)$$

$$\frac{\partial f(x_i; r_m, u_m)}{\partial r_m} = \begin{cases} 1, & \text{if (i) } \bigvee_{j=1}^n (r_{jm} \wedge x_{ij}) \geq u_m \\ & \text{(ii) } \bigvee_{j=1, j \neq s}^n (r_{jm} \wedge x_{ij}) \leq r_{sm} \wedge x_{is} \\ & \text{(iii) } r_{sm} \leq x_{is} \\ 0, & \text{otros casos} \end{cases} \quad (32)$$

El incremento para Δu_m esta dado por

$$\Delta u = \frac{\partial Q}{\partial u_m} = \sum_{i: y_i < t_i} \frac{\partial f(x_i; r_m, u_m)}{\partial u_m} - \sum_{i: y_i > t_i} \frac{\partial f(x_i; r_m, u_m)}{\partial u_m} \quad (33)$$

$$\frac{\partial f(x_i; r_m, u_m)}{\partial u_m} = \frac{\partial}{\partial u_m} \left[\left(\bigvee_{j=1}^n (r_{jm} \wedge x_{ij}) \vee u_m \right) \right] \quad (34)$$

$$\frac{\partial f(x_i; r_m, u_m)}{\partial u_m} = \begin{cases} 1, & \text{if } \bigvee_{j=1}^n (r_{jm} \wedge x_{ij}) \leq u_m \\ 0, & \text{otros casos} \end{cases} \quad (35)$$

Los valores r_m y u_m son actualizados por

$$\begin{aligned} r(k+1) &= r(k) + \Psi_1(k) \left[\frac{\Delta r(k+1)}{Nn} + \eta \frac{\Delta r(k)}{Nn} \right] \\ u(k+1) &= u(k) + \Psi_2(k) \left[\frac{\Delta u(k+1)}{Nn} + \eta \frac{\Delta u(k)}{Nn} \right] \end{aligned} \quad (36)$$

donde k es el número de iteración o paso de aprendizaje. $\Psi_1(k), \Psi_2(k)$ son funciones con valor fijo que influyen en los incrementos Δr_m y Δu_m y esta dado por

$$\Psi_1(k) = \Psi_2(k) = \frac{1}{\sqrt{x} + 1} \quad (37)$$

El entrenamiento finaliza al alcanzar un error deseado o al realizar un número determinado de épocas de entrenamiento. El resultado del entrenamiento es la matriz R capaz de clasificar una entrada en la clase correspondiente.

Capítulo 5 IMPLEMENTACIÓN

5.1 Implementación en software

Para el procesamiento acústico de las muestras de llanto se utilizó Praat 4.2 [51], así como para la obtención de coeficientes LPC. El software Praat versión 4.2 permite trabajar con señales acústicas o sonidos para su análisis y permite realizar rutinas (scripts). La extracción de parámetros se basó en la técnica LPC con muestras de un segundo, donde se obtuvieron 16 coeficientes de cada ventana (frame) de 50ms, el resultado es un vector de 304 características por muestra, tomando en cuentas las características mencionadas.

La implementación de la *fuzzyficación*, la red neuronal y el método de entrenamiento fueron hechos con Matlab 6.5.0.1 [52]. La misma versión de Matlab fue empleada para implementar el algoritmo PCA.

Después de obtener el vector con 304 coeficientes se redujo la dimensión de los vectores a un rango de 10 componentes principales. Por último cada vector es *fuzzificado* utilizando siete funciones de membresía triangulares (Figura 4.10), de donde se obtiene un vector con 70 elementos los cuales se utilizan ya sea para el entrenamiento o clasificación. Al final del mismo se agregó un elemento para etiquetarlo, es decir, para saber a que caso pertenece se coloca un 1 si el llanto pertenece a un niño normal, 2 si pertenece a un niño sordo y 3 si el vector pertenece al llanto de un niño con problemas de asfíxia. La configuración de la red, utilizando PCAs, finalmente consistió de setenta nodos en la capa de entrada y tres para la capa de salida.

Para evaluar el desempeño de la red neuronal, en términos de precisión, eficiencia, y confiabilidad se usó la técnica *5-fold cross validation* [53]. El conjunto de muestras fue asignado aleatoriamente en 5 subconjuntos, cada vez dejando un subconjunto para su clasificación y los otros restantes para entrenamiento. Este proceso se repite hasta que todos los conjuntos han sido clasificados una vez.

Siguiendo la metodología antes descrita, con la información que actualmente se posee se pueden obtener la cantidad de vectores que se muestran en la Tabla 5.1. Al momento de realizar alguna prueba la cantidad limitante es la de asfixia que contiene 340 muestras. Lo que se hace es tomar 340 muestras aleatorias de la clase sana y sorda para los experimentos.

	<i>Normales</i>	<i>Sordos</i>	<i>asfixia</i>
Número de Muestras	1049	879	340

Tabla 5.1 Número de muestras por cada categoría de llanto.

5.2 Detección del tipo de llanto

Para detectar el tipo de llanto o patología en el bebé se analizan los valores de salida de la red. Los valores en los nodos de salida de la red están entre 0 y 1. Para decidir a qué clase corresponde la muestra de entrada se selecciona al nodo con mayor nivel de activación, es decir, si el número mayor se encuentra en la neurona 1 se clasifica el llanto como normal, si el número mayor se encuentra en la neurona 2 se clasifica como sordo, de lo contrario se clasifica como asfixia, es decir si la neurona 3 contiene el número más grande.

La Tabla 5.2 muestra los valores de salida de la red para 30 muestras. En la mayoría de los casos, los valores mayores resultan en la clase a la que pertenecen realmente. Por ejemplo, en la muestra 1 (Tabla 5.2), en la columna 2, el valor: *0.77015* es mayor al de la columna 1: *0.06* y al de la columna 3: *0.09*; por lo tanto la muestra es considerada como llanto de niño sordo o hipoacúsico.

En ocasiones los valores de los dos nodos de salida de la red son casi similares o con una diferencia muy pequeña y son considerados como valores *críticos*. Tal es el caso de la muestra 15, la cual tiene una diferencia muy pequeña en los valores de salida, sin embargo el valor mayor determina a la clase que pertenece, el cual fue acertado. Otro caso aun más crítico se puede observar en la muestra 11 donde el valor de la etiqueta indica que el llanto pertenece a un bebé con problemas de asfixia y los cálculos realizados por la red FNR indican que es un niño normal, lo cual es erróneo. La solución para este tipo de problemas es aumentar el número de muestras durante el entrenamiento, para lo cual se están reuniendo más muestras.

No.	Salida de la Neurona 1	Salida de la Neurona 2	Salida de la Neurona 3	Clase a la que corresponde
1	0.06	0.77015	0.09	2
2	0.06	0.77015	0.09	2
3	0.91966	0.2038	0.01	1
4	0.91966	0.15	0.01	1
5	0.94	0.03	0.01	1
6	0.06	0.77015	0.11	2
7	0.91966	0.01735	0.01	1
8	0.06	0.77015	0.08	2
9	0.91966	0.08	0.01	1
10	0.01	0.2038	1	3
11	0.91966	0.2038	0.01	3
12	0.58	0.01	0.01	1
13	0.17	0.77015	1	3
14	0.06	0.77015	0.08	2
15	0.61	0.56	0.01	1
16	0.3	0.54	0.01	2
17	0.05	0.77015	1	3
18	0.51	0.16	0.01	1
19	0.01	0.2038	1	3
20	0.01	0.7	1	3
21	0.06	0.77015	0.12	2
22	0.06	0.77015	0.12	2
23	0.91966	0.01735	0.01	1
24	0.91966	0.01735	0.01	1
25	0.86	0.01	0.01	1
26	0.06	0.77015	0.11	2
27	0.53	0.01	0.01	1
28	0.06	0.77015	0.11	2
29	0.01	0.2038	1	3
30	1	0.01	0.01	1

Tabla 5.2 Valores de la capa de salida de la red para un conjunto de 30 muestras utilizando LPC.

5.3 Implementación electrónica

Para la implementación en Hardware es necesario manejar números del tipo flotante, aunque dentro de las bibliotecas se cuenta con una biblioteca de para el manejo de números de punto flotante [54], solo que aún se encuentra en desarrollo y presenta errores al momento de su implementación. Debido a esto se tuvieron que implementar algunas rutinas para el manejo de números del tipo flotante.

Para poder implementar el diseño digital en el FPGA, se realiza la descripción del diseño mediante Handel-C, después de lo cual se puede simular el diseño para determinar si cumple correctamente con las especificaciones de diseño (por bloques), posteriormente se realiza la síntesis del diseño para la RC100 (Figura 5.1).

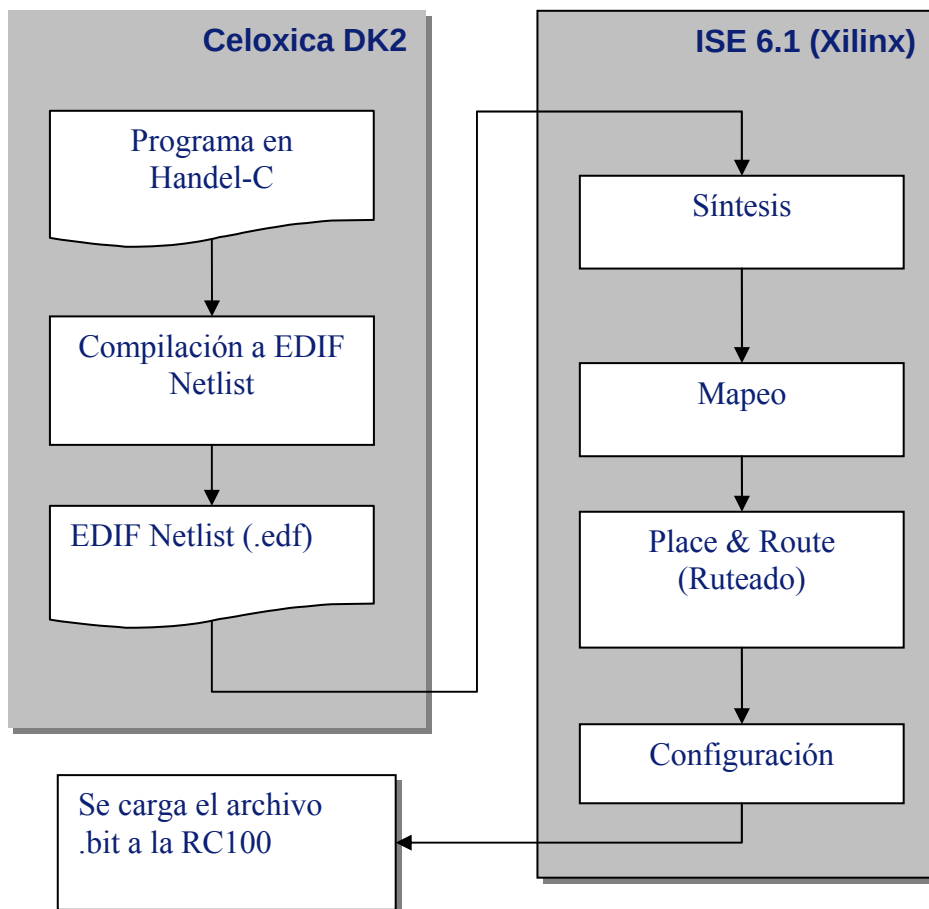


Figura 5.1 Diagrama de flujo para la implementación de la arquitectura.

5.2.1 Números de punto flotante

El objetivo es dado un número convertirlo a una representación del estándar IEEE 754 de punto flotante de 32 bits. La representación de los 32 bits requeridos es :

signo(s)	exponente (e)						mantisa (f)				
x_{31}	x_{30}	...	x_{26}	x_{25}	x_{24}	x_{23}	x_{22}	...	x_2	x_1	x_0

Tabla 5.3 Componentes de un número flotante de 32 bits en formato IEEE 754.

El bit de signo está determinado por el bit x_{31} , que toma los valores:

1 : número negativo

0 : número positivo

El exponente está determinado por los bits $[x_{30}...x_{23}]$ y cuyo rango es: 00000 , 00001, , 11111.

Teniendo una totalidad de 32 casillas o números que podemos representar y siguiendo el enfoque del estándar IEEE 754, se requiere definir un exceso para representar exponentes negativos y positivos, para ello definimos el exceso como 125 obtenido un método para la conversión:

$$\text{exponente} = \text{exp} - 127 \quad (38)$$

siendo exp el exponente del número que queremos convertir. Por ejemplo si tenemos el número $0.678e+02$, se tiene : $\text{exponente} = 02 + 127 = 27$. Con base en ello se pueden detectar un *overflow* o *underflow* para los cuales se cumple:

overflow : si $\text{exponente} > 255$

underflow : si $\text{exponente} < 0$

De la se pueden calcular el valor v de un número representado en formato IEEE 754.

sí	&	entonces
$e=255$	$f \neq 0$	$v = \text{No existe}$
$e=255$	$f=0$	$v = (-1)^s \cdot \infty$
$0 < e < 255$		$v = (-1)^s \cdot 2^{e-127} (1.f)$
$e=0$	$f \neq 0$	$v = (-1)^s \cdot 2^{e-126} (0.f)$
$e=0$	$f=0$	$v = (-1)^s \cdot 0$

Tabla 5.4 Valor v para un número flotante representado en formato 32 bits de IEEE 754.

Por lo expuesto anteriormente es claro que la representación de números en punto flotante de 32 bits es un mapeo $F: \mathbb{R} \rightarrow PF_{32bits}$, que pierde precisión en el mapeo, pero resulta atractivo desde el punto de vista de tiempo computacional.

Para llevar el número dado x a una representación de 32 bits se sigue el siguiente algoritmo.

1. Determinar el signo de x , $s = 1$ o 0 . Imprimo su valor.
2. Obtener su valor absoluto $x = \text{abs}(x)$.
3. Dividir a x mientras se satisfaga $x > 1$, por cada división incremento exp .
4. Obtener el valor de exponente, $\text{exponente} = \text{exp} + 127$. Imprimo su valor binario considerando los primeros 8 bits.
5. Obtener la representación en 23 bits del número x . Este proceso es iterativo y consiste en ir sumando $t += 1/2^i$, $i = 1, \dots, 23$, aquellos números que no sobrepasen el valor de x , por cada número sumado se escribe un 1 en la consola de salida y 0 en caso contrario.

5.2.2 Matriz de pesos y vectores de prueba

Para poder descargar la matriz de pesos (previamente obtenida de Matlab) y los vectores de entrada, previamente convertidos en formato de 32 bits hexadecimal, es necesario convertirlos en un mapa de bits. La manera más sencilla de descargar un archivo a la memoria flash es convirtiéndola en una imagen .bmp y utilizando la herramienta bmp2raw [20], para convertir el archivo en .raw y pueda ser bajado a la FLASH RAM en la dirección de memoria indicada.

Se utilizan imágenes RGB utilizando 8 bits para definir cada color. Por ejemplo si se desea utilizar el número 14.13557720184327, primero es convertido en el formato IEEE 754 de 32 bits y se obtiene 0x41622B53 (El 0x indica hexadecimal). Ahora cada elemento de 8 bits es representado en un color pero como un píxel solo contiene 24 bits es necesario tomar los siguiente 8 bits faltantes de un segundo píxel (Figura 5.2). En éste ejemplo si se desea seguir codificando otro número se toman los siguientes 2 bytes y haría falta un tercer píxel del cual se tendrían que tomar otros 2 bytes para codificar los 32bits y así sucesivamente.

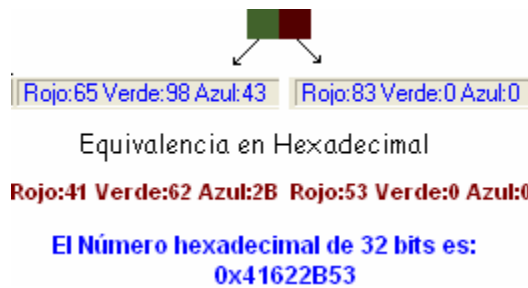


Figura 5.2 Representación de un número flotante de 32 bits en píxeles.

Dentro del FPGA el mapa de bits es almacenado de byte en byte siguiendo el orden RGB. Se leen 4 bytes como se muestra en Tabla 5.5 y se almacenan en una memoria que puede ser la SSRAM1 o en un arreglo previamente declarado.

```

MemoryLock = 1;
par
{
    a = RC100FlashReadData(0 @ FlashAddress);

    FlashAddress++;
}
par
{
    b = RC100FlashReadData(0 @ FlashAddress);

    FlashAddress++;
}
par
{
    c = RC100FlashReadData(0 @ FlashAddress);

    FlashAddress++;
}
par
{
    d = RC100FlashReadData(0 @ (FlashAddress));

    FlashAddress++;
}
par
{
    Matriz_relacional[i] = (a @ b @ c @ d);
    i++;
}
MemoryLock = 0;

```

Tabla 5.5 Código en Handel-C para el almacenamiento en memoria de los números flotantes

5.2.3 Diagrama a bloques de la arquitectura implementada

En la Figura 5.3 se muestra el diagrama a bloques de la manera en que funciona la arquitectura propuesta.

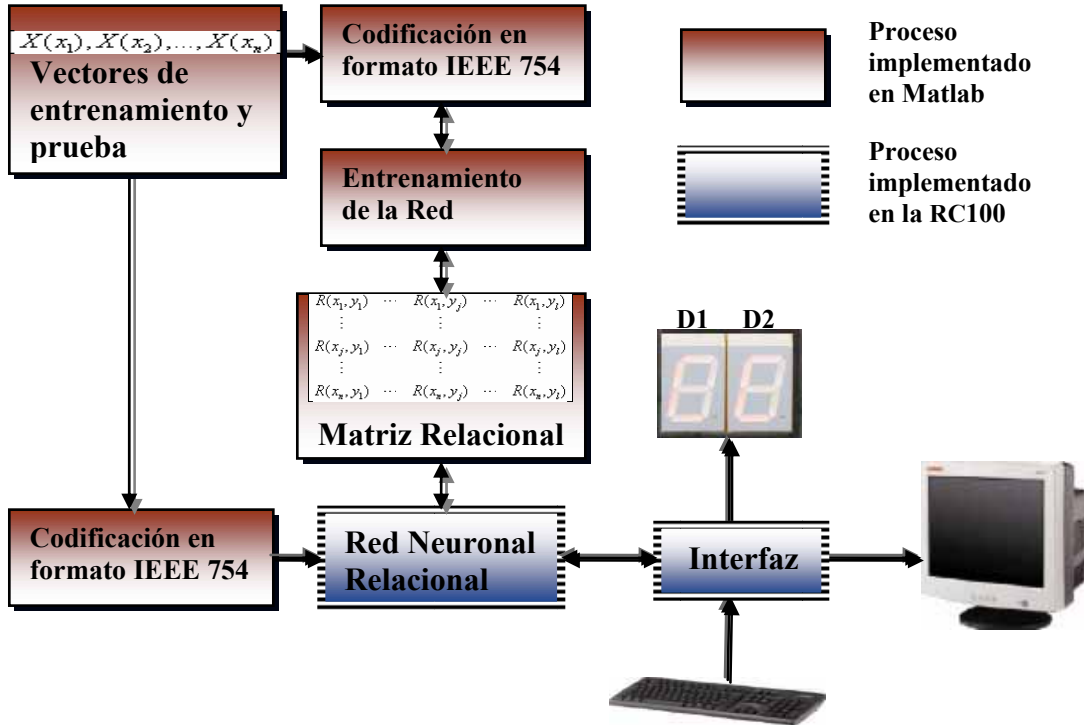


Figura 5.3 Diagrama a bloques de la arquitectura implementada.

Lo primero que se lleva a cabo es la descarga de la imagen que se utilizara como fondo en la pantalla de 512 x 480 píxeles de tamaño (Figura 5.4), la cual se encuentra en formato .raw al igual que la matriz de pesos y los vectores de prueba (Apéndice C) las cuales se colocan en las direcciones 0x000000, 0x140000, 0x0160000, respectivamente, de la FLASH RAM. Por último se descarga el archivo .bit generado por ISE, el cual configura al FPGA para ejecutar la arquitectura mostrada en la Figura 5.3 y que a continuación se describe.

Esta arquitectura se divide básicamente en dos etapas. En la primera de ellas se inicializa el sistema donde se cargan todos los datos externos y necesarios para que la red pueda realizar su función de clasificación. Después en la segunda etapa se ejecuta la red neuronal relacional para la clasificación de las muestras y se muestran los resultados obtenidos.

En la primera etapa de inicialización se carga la imagen que se desplegará como fondo de pantalla (Figura 5.4). El código es similar al presentado en la Tabla 5.5 con la diferencia que los datos de entrada son almacenados en la SSRAM0. Lo siguiente es cargar la matriz de pesos (la cual fue ajustada previamente durante el entrenamiento en Matlab) y los vectores de prueba, estos dos últimos son guardados en la SSRAM1, ya que la SSRAM0 la mayor parte del tiempo se mantiene ocupada con los datos que se despliegan en el monitor. Una vez que se han cargado los que vienen de la PC, se procede a detectar el teclado y a esperar que la tecla enter sea pulsada.



Figura 5.4 Imagen de fondo desplegada por la RC100 en el monitor.

En la segunda etapa lo que se ejecuta es la ecuación (22) implementada en Hardware. Al pulsar la tecla “enter” el programa llama al primer vector de prueba, el cual contiene 71 elementos de los cuales el último es la etiqueta, la cual es desplegada en el display 1 de la tarjeta. A continuación se procede a evaluar el vector entrante y se despliega el resultado de cada neurona en la pantalla, estos son números del 0 al 1, codificados en formato punto flotante IEEE 754 de 32 bits, por lo cual un algoritmo se encarga de comparar cual es la neurona con mayor nivel de activación y dependiendo si fue la 1, 2 o 3 despliega en el monitor NORMAL, SORDO o ASFIXIA, respectivamente. Por último en el display 2 se imprime el número de la neurona con mayor nivel de activación.

El programa se queda en espera de que la tecla “enter” sea oprimida para continuar con el siguiente vector de prueba, esto se repite de manera iterativa hasta que todos los vectores se hayan analizado. Cuando el último vector es analizado y la tecla “enter” es pulsada de nuevo, se vuelve a cargar el primer vector y se repite la operación antes descrita.

Una opción fue implementada para limpiar la pantalla, ésta se llama al presionar la tecla “Delete” o “Supr”. Lo que hace el algoritmo es volver a cargar la imagen original almacenada en la FLASH RAM, y con eso se logra borrar todos los números desplegados en pantalla y almacenados en la SSRAM0.

En el monitor solo se despliega el resultado de la red, para saber si es correcto se tiene que comparar con el valor desplegado en el display 1, aunque en el display 2 se muestra el número de la neurona ganadora (él cual es el mismo que el desplegado en pantalla).

Todos los caracteres que aparecen en el monitor durante la ejecución del programa, están almacenadas en memorias RAM declaradas internamente. Ya que es necesario desplegar píxel por píxel cada carácter que se desee mostrar en la pantalla.

5.2.4 Diagrama esquemático de la red Fuzzy Neural Relacional

En la Figura 5.5 se encuentra el diseño esquemático empleado para la implementación de la red FNR. A continuación se describe el funcionamiento de la misma. La matriz de pesos y los vectores de entrada se encuentran almacenados en distintas localidades de la flash RAM, la primer tarea que se realiza es la lectura de la matriz de pesos: $R(x_i, y_j)$, donde $x_{i...(n-1)}$ indica la fila en que se encuentra el actual valor y $y_{j...l}$ se refiere a la clase que pertenece. Después se lee el vector de entrada X_i el cual cuenta con n elementos, donde $X_{i=n}$ es la etiqueta que indica la clase a la cual pertenece de un total de l clases. Ambos valores son almacenados en registros y se pasan por el comparador “Min”, obteniendo como resultado el valor mínimo entre el valor de vector de entrada y el del peso relacional actual. El resultado es almacenado en el registro $\text{Min } x_{ij}$, el cual es comparado con el umbral U_j , usando el comparador “Max”, donde se selecciona el valor numérico mayor y es guardado en el registro $\text{Max } x_i$ el cual es comparado con el valor anterior y se selecciona al más grande, en caso de ser el primer valor de entrada se compara con un 0. La misma operación se repite hasta que los $n-1$ valores del vector X_i para la clase y_j han sido procesados.

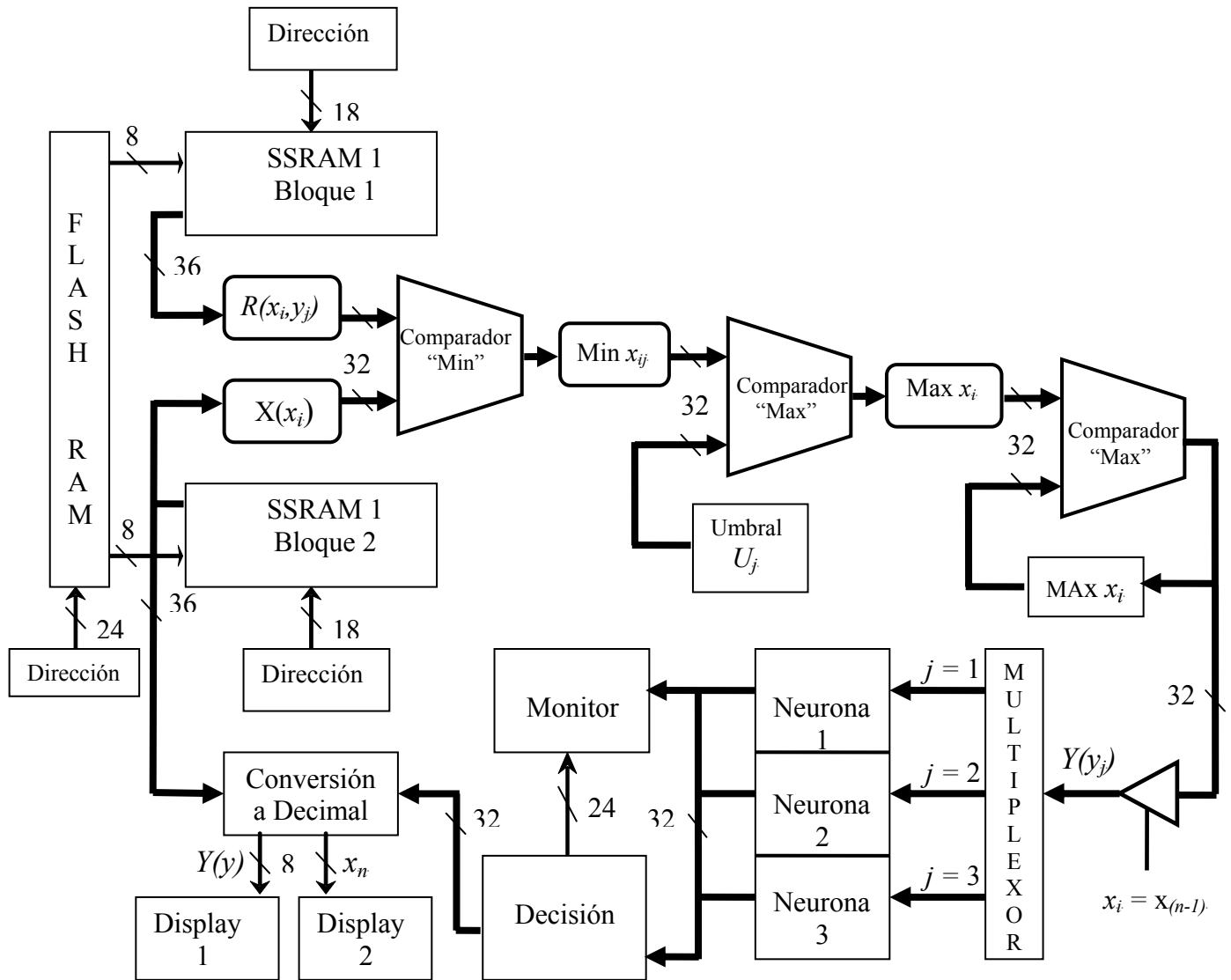


Figura 5.5 Diagrama esquemático de la red FNR implementado en el FPGA.

Al final el resultado obtenido es nombrado $Y(y_j)$. Un multiplexor se encarga de direccionar a $Y(y_j)$ para que sea guardado en el registro $Neurona_{(j...l)}$ correspondiente, por último, el resultado se envía al monitor sin decodificar. Si $X_{i=n}$ el valor es trasladado a un codificador decimal y enviado al Display 1 de la RC100 para su visualización, el cual es la etiqueta de la muestra y nos indican la clase a la cual pertenece. La operación se repite hasta que se cumpla $Y(y_{j=l})$, cuando esto sucede los valores obtenidos para cada $Y(y_j)$ se comparan y el último bloque de la red FNR se encarga de tomar una decisión para poder enviar el resultado al Display 1 (previamente decodificado) y en la pantalla se imprime “NORMAL”, “SORDO” o “ASFIXIA”, dependiendo de cual sea la neurona con mayor nivel de activación.

La frecuencia de operación esta dada por un reloj de 20 MHz y el número total de ciclos de reloj que la red requiere para probar cada vector de entrada son 64,365, es decir 3.2 milisegundos. Aunque el resultado no esta disponible en pantalla hasta que han transcurrido 122 milisegundos, debido a que es el tiempo que le toma al FPGA el refrescar la pantalla del monitor. Estos resultados son comparados con los que se obtienen al simular en Matlab, donde transcurren 2.5 segundos para obtener el resultado de un solo vector de entrada.

Capítulo 6 RESULTADOS

El análisis se realizó en base a muestras de un segundo y se dividió en dos grupos. El Grupo A, contiene mil veinte muestras y al experimentar con este grupo se aplicó la técnica de validación cruzada, resultando 5 subgrupos con 68 muestras de cada clase, en cada ocasión dejando un subgrupo para prueba y el resto para el entrenamiento. Este proceso es repetido hasta que todos los conjuntos han sido usados una vez para prueba. Para el Grupo B, se consideró el número total de muestras (ver Tabla 6.1). La selección de las muestras fue realizada al azar. Para el uso de validación se dividió en 3 subgrupos con 250 muestras para el entrenamiento y el resto de las muestras cada una de las clases para prueba (clasificación), el experimento se repitió en cinco ocasiones consecutivas y se tomó el promedio.

	Muestras <i>normal</i>	Muestras <i>sordos</i>	Muestras <i>asfixia</i>	Total
Grupo A	340	340	340	1020
Grupo B	1049	879	340	2268

Tabla 6.1 Número de muestras de llanto en los grupos A y B.

La red neuronal relacional se entrenó para clasificar llanto de niños normales, llanto de niños sordos y el llanto de niños con problemas de asfixia. Los conjuntos de datos que sirven de entrada a la red son obtenidos aplicando la técnica LPC (obteniendo 304 coeficientes en Praat), después son reducidos mediante el PCA a diez elementos y posteriormente son fuzzificados.

6.1 Resultados obtenidos en Matlab

Los resultados se presentan en una matriz de confusión para cada caso. En primer lugar se probaron 5 funciones de membresía para la trapezoidal densa (T_D) y triangular. Los resultados fueron el 91.1765% para la trapezoidal densa (ver Tabla 6.2) y 93.1373 para la funciones de membresías tipo triangular (ver Tabla 6.3), para clasificación correcta. Ambos resultados fueron obtenidos utilizando el Grupo A.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	70	61	9		
Sordos	72		72		
Asfixia	62	2	7	53	
Total	204				91.1765 %

Tabla 6.2 Resultado para el Grupo A utilizando 5 funciones de membresía de la trapezoidal densa.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	82	82			
Sordos	60		60		
Asfixia	62	5	9	48	
Total	150				93.1373%

Tabla 6.3 Resultado para el Grupo A utilizando 5 funciones de membresía triangulares.

Los resultados para el Grupo B aplicando 5 funciones para trapezoidal densa es 87.945% de precisión (ver Tabla 6.4), y utilizando 5 funciones triangulares se obtiene un 90.909% de exactitud (ver Tabla 6.5).

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	799	635	150	14	
Sordos	629		629		
Asfixia	90	3	16	71	
Total	1518				87.945 %

Tabla 6.4 Resultado para el Grupo B utilizando 5 funciones de membresía de la trapezoidal densa.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	799	670	90	39	
Sordos	629	0	629	0	
Asfixia	90	4	5	81	
Total	1518				90.909%

Tabla 6.5 Resultado para el Grupo B utilizando 5 funciones de membresía triangulares.

Utilizando 7 funciones de membresía en el Grupo A, el resultado obtenido usando la TD es 92.1569% de clasificación correcta (ver Tabla 6.6) y el resultado aplicando funciones triangulares es del 94.6078% de precisión (ver Tabla 6.7).

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	69	69			
Sordos	69		69		
Asfixia	66	10	6	50	
Total	204				92.1569 %

Tabla 6.6 Resultado para el Grupo A utilizando 7 funciones de membresía de la trapezoidal densa.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	72	68	4		
Sordos	64		64		
Asfixia	68	3	4	61	
Total	204				94.6078 %

Tabla 6.7 Resultado para el Grupo A utilizando 7 funciones de membresía triangulares.

Los resultados en el Grupo B empleando 7 funciones de T_D es 88.735% de precisión (ver Tabla 6.8), y utilizando 7 funciones triangulares se obtiene un 92.292% de exactitud (ver Tabla 6.9). En la Tabla 6.1, se puede observar que el número de muestras para el entrenamiento se ve limitado por la clase de asfixia ya que ésta contiene un total de 340 muestras y las muestras que se utilizaran en la fase de prueba no son ocupadas durante el entrenamiento.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	799	646	85	68	
Sordos	629		629		
Asfixia	90	4	14	72	
Total	1518				88.735%

Tabla 6.8 Resultado para el Grupo B utilizando 7 funciones de membresía de la trapezoidal densa.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	799	691	87	21	
Sordos	629	0	629	0	
Asfixia	90	2	7	81	
Total	1518				92.292%

Tabla 6.9 Resultado para el Grupo B utilizando 7 funciones de membresía triangulares.

Los resultados del Grupo A se pueden ver de manera gráfica en Figura 6.1 y también los resultados para el Grupo B (Figura 6.2).

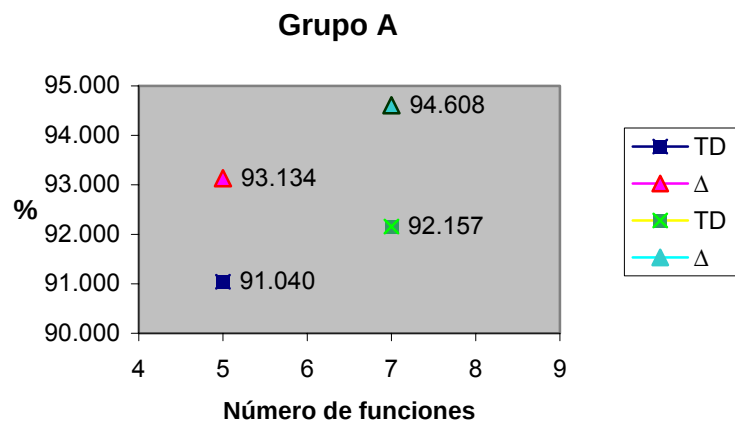


Figura 6.1 Resultados obtenidos para el Grupo A.

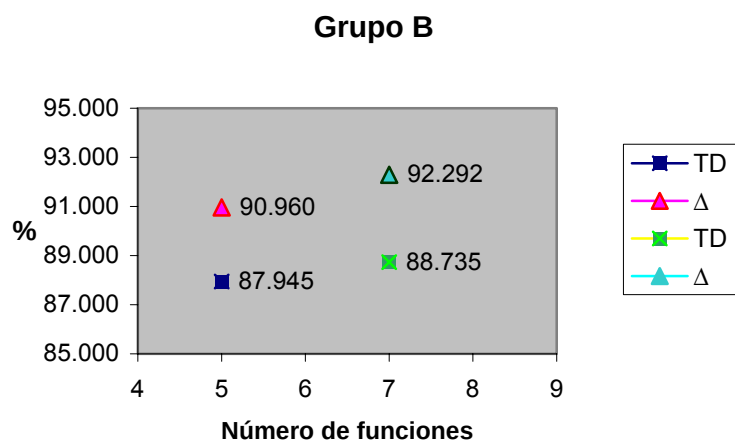


Figura 6.2 Resultados obtenidos para el Grupo B.

6.2 Resultados obtenidos en el FPGA

Después de la implementación de la arquitectura se obtienen los resultados para los vectores de prueba. Al realizar la síntesis del sistema se obtiene el reporte de los recursos de hardware empleados.

```
The AVERAGE CONNECTION DELAY for this design is:      2.102
The MAXIMUM PIN DELAY IS:                             6.704
The AVERAGE CONNECTION DELAY on the 10 WORST NETS is:  6.297

Listing Pin Delays by value: (nsec)
```

Figura 6.3 Sección del reporte de ruteo.

Estos datos nos indican que la frecuencia máxima de operación de la arquitectura es de aproximadamente 1/8.906 ns equivalente a 60 MHz.

```
Device utilization summary:

Number of External GCLKIOBs      1 out of 4      25%
Number of External IOBs         221 out of 284    77%
Number of LOCed External IOBs   221 out of 221  100%
Number of BLOCKRAMs             1 out of 14       7%
Number of SLICES                 2321 out of 2352  96%
Number of GCLKs                  2 out of 4       50%
```

Figura 6.4 Sección del reporte de mapeo.

Para mostrar la matriz de pesos y los vectores de entrada almacenados en la memoria de la tarjeta RC100, se creo un programa que simplemente los cargara y enviara a la pantalla, esto se puede apreciar en la Figura 6.5.

El resultado al probar la red FNR en el FPGA fue el mismo que el obtenido en Matlab 94% de precisión al utilizar 10 PCAs y 7 funciones de membresía triangulares. El programa manda a la pantalla el resultado de cada una de las neuronas Normal, Sordo y Asfixia respectivamente, con lo cual toma una decisión en base a la que contiene la cantidad mayor y despliega NORMAL, SORDO o ASFIXIA según sea el caso. El valor que el display 1 (el de la izquierda Figura 6.6) se refiere es a la etiqueta de la muestra y el display 2 muestra el resultado de la clase que la red FNR arrojo. Tomando en cuenta que Normal es 1, Sordo es 2 y Asfixia es 3.

En la **Tabla 6.10** se muestra una comparación del resultado que se obtuvo durante el desarrollo de esta tesis, con algunos ya reportados, para los cuales se describe detalladamente en la introducción la clase de llantos que clasificó y la metodología utilizada.

Comparación de Resultados	Clasificación
Red <i>feed forward</i> (José Orozco)	98.0%
Red FNR	94.6 %
Red de base radial (Taco Ekkel)	85.0%
Modelos Ocultos de Markov Dror Lederman	83.0%

Tabla 6.10 Comparación del resultado obtenido con algunos ya reportados

Capítulo 7 CONCLUSIONES

7.1 Conclusiones

El uso de las unidades de llanto durante el procesamiento acústico de la señal de llanto no aportó resultados mayores al 80%, por lo que se descartó su uso. Lo cual nos hace suponer que los espacios de silencios son también parte de la información útil, la cual nos permite una mejor clasificación en la detección de patologías.

Se implementó una red feed-forward en Handel-C. Debido a que se utilizan números flotantes en el formato IEEE 754 de 32 bits y se realizan operaciones algebraicas, además de emplear la función sigmoideal, el resultado obtenido en Matlab del 95.33% de eficiencia se ve reducido considerablemente a 87.33% de precisión conseguido en Handel-C, por lo que se tuvo que buscar otra alternativa.

Los resultados obtenidos al implementar la red Fuzzy Neural Relacional (FNR) en Matlab, utilizando 10 PCAs y 7 funciones de membresía triangulares, es del 94.6078% de clasificación correcta. Lo cual motivo a su implementación en Hardware.

El método utilizado para la extracción de características son los LPC esto es pensando en su futura implementación en Hardware, ya que la manera de calcularlos es mucho más sencilla que los MFCC, esto es debido a que solo se utilizan series de sumatorias.

Los resultados obtenidos al implementar la red FNR en la tarjeta de desarrollo RC100 de Celoxica es el mismo que el obtenido en Matlab, 94% de precisión. Los requerimientos para su prueba son la matriz de pesos ya entrenada y los vectores de entrada.

Las ventajas que se encontraron al utilizar este tipo de red es que no se requiere de operaciones algebraicas como en una feed-forward ya que como se están manejando números flotantes de 32 bits éstos ya contienen cierta cantidad de error, y al realizar una operación el error no solo se conserva sino que en la mayoría de los casos aumenta. Otro problema encontrado en la implementación de una feed-forward es la función de activación sigmoideal (ecuación 7). La función de salida de la red FNR solo se basa en comparaciones de máximos y mínimos (ecuación 22), es decir

que el error que cada elemento trae implícitamente no aumenta. Una ventaja extra que se obtiene al utilizar una etapa de fuzzificación es que los valores de entrada son reducidos al rango entre 0 y 1, lo cual hace que el error al manejar números flotantes en su equivalente a 32 bits se reduzca y no se requiere una etapa de desfuzzificación.

Se ha comprobado que la aplicación de una red neuronal del tipo híbrida puede ser la solución para implementar un sistema en Hardware dedicado al reconocimiento o clasificación del llanto de bebés.

Debido a la propia naturaleza de los FPGAs, se ha podido realizar el manejo de la red FNR en paralelo a otros procesos como son el despliegue de datos en el monitor, el manejo del teclado y las memorias internas. Lo que reduce el tiempo de cómputo desde que el dato esta siendo procesado hasta que se obtiene el resultado.

Los bloques del programa desarrollado en Handel-C se realizaron de tal manera que puedan ser re-usables, para facilitar una modificación futura o poder cambiar de aplicación.

7.2 Trabajo futuro

Se considera como trabajo futuro la recolección de muestras de llanto de niños patológicos tanto de sordera así como de asfixia, ya que el entrenamiento se ve limitado por la clase con el menor número de muestras disponibles. Con lo que se pretende mejorar el porcentaje de eficiencia en cuanto a clasificación de la Red.

Para mejorar el desempeño de la red FNR es recomendable implementar un algoritmo genético que la provea con las mejores características. Por ejemplo encontrar el número óptimo de PCAs a utilizar, el número de funciones de membresía así como el tipo (probando entre triangulares, gaussiana, trapezoidal, trapezoidal densa, pi, etc.), ajustar los parámetros internos de la red FNR y por último buscar el número adecuado de épocas para el entrenamiento.

El actual trabajo, es el inicio de un sistema en tiempo real que permitirá reconocer el tipo de llanto del bebé y se pretende que emita un mensaje de voz sintetizado. Los bloques a construir son la digitalización del llanto, extracción de parámetros LPC y su reducción por PCA para entregarle estos vectores al bloque actualmente diseñado y tomar una decisión.

Finalmente adecuar el sistema para que además de clasificar si el estado del bebé es normal o tiene alguna patología, se pudiera hacer una subclasificación para detectar la causa del llanto, como hambre, dolor, incomodidad, etc. o utilizar el sistema para otras aplicaciones relacionadas.

LISTA DE FIGURAS

Figura 2.1 Estructura de un FPGA.....	10
Figura 2.2 Ejemplo de bloques ejecutándose en paralelo.....	11
Figura 2.3 Ejemplo de bloques ejecutándose en paralelo de diferentes tamaños.....	11
Figura 2.4 Canal de comunicación entre bloques de ejecución paralela.....	12
Figura 2.5 Elementos compartidos entre Handel-C y ANSI-C.....	13
Figura 2.6 Tarjeta RC100.....	14
Figura 2.7 Patrones de naturaleza temporal.....	16
Figura 2.8 Aparato fonador.....	18
Figura 2.9 Diagrama del aparato fonador.....	19
Figura 2.10 Modelo del llanto.....	20
Figura 2.11 Unidades de llanto.....	22
Figura 3.1 Proceso de reconocimiento del llanto de bebés.....	23
Figura 3.2 Forma de onda y espectrograma del llanto de un bebé normal.....	25
Figura 3.3 Forma de onda y espectrograma del llanto de un bebé sordo.....	25
Figura 3.4 Forma de onda y espectrograma del llanto de un bebé con asfixia.....	26
Figura 3.5 Diagrama a bloques del modelo simplificado del habla.....	26
Figura 4.1 Modelo de una neurona artificial.....	31
Figura 4.2 Neurona artificial simple.....	32
Figura 4.3 Función de activación sigmoideal.....	33
Figura 4.4 Red tipo feed-forward completamente interconectada.....	34
Figura 4.5 Arquitectura de una neurona clásica y una neurona Difusa.....	39
Figura 4.6 Red neuronal con dos capas.....	40
Figura 4.7 Red de una entrada simple fuzzy asociada a un nodo de salida.....	41
Figura 4.8 Diagrama a bloques del funcionamiento de la red FNR.....	42
Figura 4.9 Diagrama a bloques de la obtención del Vector Final de Características.....	43
Figura 4.10 Función Δ	44
Figura 4.11 Representación de una estructura 3n-dimensional.....	44
Figura 4.12 Función T.....	45
Figura 4.13 Estructura de la función TD con 5 propiedades lingüísticas.....	46
Figura 5.1 Diagrama de flujo para la implementación de la arquitectura.....	52
Figura 5.2 Representación de un número flotante de 32 bits en píxeles.....	55
Figura 5.3 Diagrama a bloques de la arquitectura implementada.....	56
Figura 5.4 Imagen de fondo desplegada por la RC100 en el monitor.....	57
Figura 5.5 Diagrama esquemático de la red FNR implementado en el FPGA.....	59
Figura 6.1 Resultados obtenidos para el Grupo A.....	64
Figura 6.2 Resultados obtenidos para el Grupo B.....	64
Figura 6.3 Sección del reporte de ruteo.....	65
Figura 6.4 Sección del reporte de mapeo.....	65
Figura 6.5 Matriz Relacional y Vector de entra en el FPGA.....	66
Figura 6.6 Clasificación de una muestra realizada por el FPGA.....	66

LISTA DE TABLAS

Tabla 2.1 Tecnologías de fabricación de circuitos integrados.....	8
Tabla 2.2 Dispositivos lógicos programables.....	9
Tabla 2.3 Dispositivos con los que cuenta la tarjeta RC100.	14
Tabla 4.1 Implicación booleana $p \rightarrow q$	37
Tabla 5.1 Número de muestras por cada categoría de llanto.....	50
Tabla 5.2 Valores de la capa de salida de la red para un conjunto de 30 muestras utilizando LPC.....	51
Tabla 5.3 Componentes de un número flotante de 32 bits en formato IEEE 754.....	53
Tabla 5.4 Valor v para un número flotante representado en formato 32 bits de IEEE 754.....	53
Tabla 5.5 Código en Handel-C para el almacenamiento en memoria de los números flotantes.....	55
Tabla 6.1 Número de muestras de llanto en los grupos A y B.....	61
Tabla 6.2 Resultado para el Grupo A utilizando 5 funciones de membresía de la trapezoidal densa.	62
Tabla 6.3 Resultado para el Grupo A utilizando 5 funciones de membresía triangulares.....	62
Tabla 6.4 Resultado para el Grupo B utilizando 5 funciones de membresía de la trapezoidal densa.....	62
Tabla 6.5 Resultado para el Grupo B utilizando 5 funciones de membresía triangulares.....	62
Tabla 6.6 Resultado para el Grupo A utilizando 7 funciones de membresía de la trapezoidal densa.....	63
Tabla 6.7 Resultado para el Grupo A utilizando 7 funciones de membresía triangulares.....	63
Tabla 6.8 Resultado para el Grupo B utilizando 7 funciones de membresía de la trapezoidal densa.....	63
Tabla 6.9 Resultado para el Grupo B utilizando 7 funciones de membresía triangulares.....	64
Tabla 6.10 Comparación del resultado obtenido con algunos ya reportados	67

BIBLIOGRAFÍA

- [1] Soltis, J. "The signal functions of early infant crying". Behavioral and Brain Sciences, Cambridge, 2003.
- [2] Ekkel, T. "*Neural Network-Based Classification of Cries from Infants Suffering from Hypoxia-Related CNS Damage*". Tesis de Maestría. University of Twente. The Netherlands, 2002.
- [3] Orozco, J., Reyes, C. "Extracción y Análisis de Características Acústicas del Llanto de Bebés para su Reconocimiento Automático Basado en Redes Neuronales". Tesis de Maestría, INAOE, Puebla, Mex., 2002.
- [4] Lederman, D. "*Automatic Classification of Infants' Cry*". Tesis de Maestría. University of Negev. Israel, 2002.
- [5] Petroni, M., Malowany, A., Johnston, C., Stevens, B. "*Identification of Pain from Infant Cry Vocalizations Using Artificial Neural Networks (ANNs)*". The International Infant Cry Research Group. Applications and Science of Artificial Neural Networks. The International Society for Optical Engineering. Volumen 2492. 1995. pp.729-738.
- [6] Hurlock, E.B. "*Child Development*". Nueva York. McGraw-Hill. 1950.
- [7] Wasz-Höckert, O., Lind, J., Vuorenkoski, V., Partanen, T., Valanne, E. "*The Infant Cry: A Spectrographic and Auditory Analysis*". Spastics International Medical Publication, Lavenham, U.K., 1968.
- [8] Wasz-Höckert, O., Lind, J., Vuorenkoski, V., Partanen, T., Valanne, E. "*El Llanto en el Lactante y su Significación Diagnóstica*", Editorial Científico-Médica, Barcelona, 1970.
- [9] Golub, H., Corwin, M. "*Infant Cry: a Clue to Diagnostic*". Pediatrics, vol. 69. pp. 197-201. 1982.
- [10] Zeskind P.S., Lester B. "Acoustic features and auditory perceptions of the Cries of Newborns with Prenatal Complications". Child development, pp. 580-589. 1978.

- [11] Cano, Sergio D., Escobedo, D., Coello, Eddy. “*El Uso de los Mapas Auto-Organizados de Kohonen en la Clasificación de Unidades de Llanto Infantil*”. Grupo de Procesamiento de Voz, 1er. Taller AIRENE, Universidad Católica del Norte, Chile, 1999. pp.24-29.
- [12] Cano, S., Escobedo, D., Coello, Eddy. “*Clasificación de Unidades de Llanto Infantil Mediante el Mapa Auto-Organizado de Kohonen*”. Capítulo 6 del texto: Reconocimiento de Patrones con Redes Neuronales del Proyecto Competitivo CYTED VII (Subprograma de Electrónica e Informática Aplicadas). Salamanca. España, 2001. pp.137-148.
- [13] Castillo, A., Chivas, P., Torres, M., Cano, S., Escobedo, D. “*Clasificación de Unidades de Llanto Infantil Mediante Algoritmos de Cuantificación Vectorial Adaptable*”. Taller AIRENE CYTED, U. do Minho, Braga, Portugal, 2001.
- [14] Torres, M., Et. al. “*Una Aproximación al Diagnóstico del Llanto Infantil Basado en Redes Neuronales Supervisadas*”. 1er. Taller AIRENE sobre Reconocimiento de patrones con Redes Neuronales. Universidad Católica del Norte, Antofagasta. Chile, 1999.
- [15] Amanatidis, D. “FPGA Implementation of Cellular Neural Networks: And Initial Study”. University of Kingston. 2000.
- [16] Maxinez, D., Alcalá, J. “VHDL EL arte de programar Sistemas Digitales”. Ed. CECSA. México, 2002.
- [17] Pérez, G. M., “*Introducción a la Tecnología FPGA*”. Coordinación de Ciencias Computacionales, INAOE, Puebla, México, 2003.
- [18] Lysaght, P., Stokwood, J., Law, J., Girma, D. “Artificial Neural Network Implementation on a Fine-Grain FPGA”. University of Strathclyde. 1994.
- [19] Handel-C. Language Refence Manual, Celoxica Ltd. Version 1.2. United Kingdom, 2001.
http://www.celoxica.com/support/view_article.asp?ArticleID=518
- [20] DK 2 Users Manual. Celoxica. Version 1.2. United Kingdom, 2002.
http://www.celoxica.com/support/view_article.asp?ArticleID=518

- [21] Handel-C. Language Refence Manual, Celoxica Ltd. Version 1.2. United Kingdom, 2001. <http://www.celoxica.com/methodology/handelc.asp>
- [22] RC100 User Manual. Celoxica. Version 1.2. United Kingdom, 2001. http://www.celoxica.com/support/view_article.asp?ArticleID=376
- [23] RC100 Hardware Manual. Celoxica. Version 1.2. United Kingdom, 2001. http://www.celoxica.com/support/view_article.asp?ArticleID=376
- [24] RC100 Function Library Manual. Celoxica. Version 1.2. United Kingdom, 2001. http://www.celoxica.com/support/view_article.asp?ArticleID=376
- [25] Reyes, C.A., “Sistemas Difusos (Fuzzy Systems): Fundamentos y Aplicaciones”, INAOE, Puebla, 2004.
- [26] Sakoe, H. "Two-Level DP-Matching- A Dynamic Programming - Based Pattern Matching Algorithm for Connected Word Recognition", IEEE Trans. on Acoustic, Speech and Signal Processing, vol. ASSP-27, num.6, pp.588-595, December 1979.
- [27] Devijver P.A., y Kittler J.V., “Pattern Recognition. A Statistical Approach”, Prentice Hall- Englewood Cliffs, 1982.
- [28] Rabiner, L.R., A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", in Readings in Speech Recognition, A.Waibel and K.F. Lee, eds., Morgan Kau_mann Publishers Inc., San Mateo, Calif, 1990.
- [29] Zue, V.W., \The Use of Speech Knowledge in Automatic Speech Recognition", in Readings in Speech Recognition, A. Waibel and K.F. Lee, eds., Morgan Kau_mann Publishers Inc., San Mateo, Calif, 1990.
- [30] Fu, Limin. “*Neural Networks in Computer Intelligence*”. Singapore. McGraw-Hill. 1994.
- [31] Peláez, C. “Tratamiento de la Información Multimedia”. Dpto. Teoría de la Señal y Comunicaciones. Madrid, 2003.
- [32] Cuétara, J., Margarita, P. “Aparato Fonador”, Dpto. Fonética y fonología. UNAM, México, 2003.
- [33] Castañeda, F. “*El Lenguaje Verbal del Niño*”. Ed. UNMSM. 1999.

-
- [34] “*El Sistema de Producción de Habla*”. Dpto. de Ingeniería Eléctrica. Madrid, 2003.
- [35] Hernáez, I. “*Modelos de Producción de Voz*”. Dpto. Electrónica y Telecomunicaciones. España, 2001.
- [36] Gutiérrez, G., Salvador, R. “*Desarrollo de un Compresor LPC de Voz en Tiempo Real*”. Instituto Tecnológico de Chihuahua, México, 2001.
- [37] Bustos, J. “*Estudio de Sistemas de compresión de voz digital orientado a telefonía celular*”. Dpto. De Informática y Ciencias de la Computación. Chile, 2002.
- [38] Moreno, D. “*Harmonic Decomposition Applied to Automatic Speech Recognition*”. Universidad de Birmingham. UK, 2002.
- [39] Bernal, J., Bobadilla, J. y Gomez, P. “*Reconocimiento de Voz y Fonética Acústica*”. Editorial Alfaomega, México, 2000.
- [40] Atal, B. S. “*Automatic Speaker Recognition Based on Pitch Contours*”. Journal of the Acoustical Society of America, volumen 52. 1972. pp 1687-1697.
- [41] Markel, John D., Gray, Augustine H., *Linear prediction of speech*. New York: Springer-Verlag, 1976.
- [42] Kröse B., Van der Smagt P., “*An introduction to neural computing*”. The University of Amsterdam, Octava edición, 1996.
- [43] Reyes, C. A., “*Redes Neuronales*”, Material de apoyo para curso de Sistemas Híbridos, INAOE, Puebla, 2002.
- [44] Gómez M., “*Teoría y simulación de Redes Neuronales Artificiales*”. Tutorial, Barcelona, España, 1999.
- [45] Goodrich A. Michael, “A Fuzzy Logic Tutorial”, Julio, 2001.
- [46] Kandel, A., “*Fuzzy Techniques in Pattern Recognition*”, John Wiley & Sons, New York, 1982.
- [47] Zadeh, L.A., “*The Concept of Linguistic Variable and its Application to Approximate Reasoning*”, Information Sciences 8, 1977, pp 301-357.
- [48] Reyes, C. A., “*Sistemas Difusos (Fuzzy Systems): Fundamentos y Aplicaciones*”, INAOE, Puebla, 2004.

-
- [49] Pedrycz W., "Neurocomputations in Relational Systems", in Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 3, March 1991, pp. 289-296.
- [50] Reyes, C.A, "*ON the Design of a Fuzzy Relational Neural Network for Automatic Speech Recognition*", Tesis de doctorado, The Florida State University, E.U, 1994.
- [51] Boersma, P., Weenink, D., *Praat v. 4.2 "Sistema para hacer fonética por computadora"*. University of Amsterdam. The Netherlands. 1992-2004.
- [52] The MathWoks, Inc., MATLAB[®]. Version. 6.5.0.1. 1984 - 2002.
- [53] Haykin, Simon S. "*Neural networks: A comprehensive foundation*". New York: Macmillan College Publishing Company, Inc., 1994.
- [54] Floating Point and Fixed Point Libraries. Celoxica. Version 1.0. United Kingdom, 2001. <http://www.celoxica.com/support>
- [55] Kadlec, J., Hermanek, A., Softley, Ch., Matousek, R., Licko, M. "*32-bit Logarithmic ALU for Handel C 2.1 and Celoxica DKI*". Presentado en conferencia, Celoxica. Stratford, 2001.
- [56] Schafer, R. W., and Rabiner, L. R. "*Digital Representations of Speech Signals*". Proceedings IEEE, vol. 63. 1975. pp. 662-677.
- [57] Atal, B.S., M.R. Schroeder. "*Predictive Coding of Speech Signals and Subjective Error Criteria*". IEEE Transactions on Acoustics, Speech and Signal Processing. 1979..
- [58] John D., Gray, Augustine H. "*Linear Prediction of Speech*". New York: Springer-Verlag. 1976.
- [59] Huang, X., Acero, A., Hon, H. "*Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*". Prentice-Hall, Inc., USA, 2001.
- [60] Davis, S., and Mermelstein, P. "*Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences*". IEEE Transactions on Acoustics, Speech and Signal Processing. 1980. 28(4), pp. 357-366.

- [61] Fu, K.S. "*Digital Pattern Recognition*". Communication and cybernetics ; v.10. 2nd ED. - New York: Springer-Verlag, Inc., 1980.
- [62] Mitchell, Tom. "*Machine Learning*". Boston, Massachusetts: WCB / McGraw-Hill. 1997.
- [63] Haykin, Simon S. "*Neural networks: A comprehensive foundation*". New York: Macmillan College Publishing Company, Inc., 1994.
- [64] Bishoop, Chris. "*Neural Networks for Pattern Recognition*". Oxford University Press Inc., New York. 1995.
- [65] Orozco, J., Reyes, C. "*Classification of Infant Cry Using a Neura Network*". III Symposium Internacional en Tecnologías Inteligentes. Apizaco, Tlax., Méx., 2002.
- [66] Orozco, J., Reyes, C. "*Clasificación de Llanto del Bebé Utilizando una Red Neuronal de Gradiente Conjugado Escalado*". Mexican International Conference on Artificial Intelligence. Mérida, Yucatán, Méx., 2002.
- [67] Torres, M., Cano, S., Escobedo, D., Castillo, A. "*Reconocimiento de Clases de Llanto Infantil Aplicando una Red Neuronal de Kohonen*". 2do. Taller AIRENE sobre Reconocimiento de patrones con Redes Neuronales. Salamanca España, 2000.

APÉNDICE A UNIDADES DE LLANTO

Se realizó un estudio donde se exploraba la posibilidad de utilizar las unidades de llanto durante el procesamiento acústico en vez de realizar segmentaciones de un segundo como se realiza hasta el momento. Para este experimento se utilizaron las muestras proporcionadas por el Grupo de Procesamiento de Voz (GPV) de la Universidad de Oriente (UO) en Santiago de Cuba.

Las grabaciones de llanto procedentes del GPV fueron realizadas en el Hospital Materno Este de Santiago de Cuba. El llanto fue inducido por un estímulo estandarizado: un toque del dedo índice en el talón del infante. Los niños estuvieron en posición de cúbito supino, piernas abiertas y extendidas y no se encontraban llorando en el momento del estímulo. Se grabaron para cada infante 12 segundos de llanto con un equipo SONY CFS-210. Se usó un micrófono PHILIPS SBC-3040 ubicado a una distancia de los labios del niño de 17 cm más o menos. Luego estos llantos fueron digitalizados por el sistema de adquisición de habla PCVOX. Los archivos originales tienen la extensión .MST (Fig. 3) y están codificados a 16 bits y muestreados a 8000 Hz, siendo en total 26 casos, 13 normales y 13 patológicos.

En cuanto a las muestras provenientes del INAOE para crear un estándar del conjunto total se escogieron 13 muestras de llantos de niños normales y 13 de niños sordos, con diferentes tiempos de duración. Las muestras seleccionadas debían de ser mayor a 12 segundos para poder reducirlas a la cantidad de tiempo mencionada, después de segmentarlas se filtraron para reducir el ruido de fondo y se colocaron en dos carpetas: normales y sordos.

Las cantidad de unidades de llanto obtenidas fueron: 56 de hipoxia, 177 de normales del GPV y 118 del INAOE) y 84 de asfixia.

El entrenamiento se ve limitado por la clase de hipoxia ya que es la clase que cuenta con el menor número de muestras, donde se toman 56 muestras de cada clase donde 46 de ellas se utilizan para el entrenamiento y las sobrantes se utilizan de prueba. El resultado que la red FNR arroja se muestra en la Tabla A.1, éste fue obtenido utilizando el método de validación cruzada y en promedio es de 66% de clasificación correcta.

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Hipoxia	
Normal	10	6		4	
Sordos	10	3	7		
Hipoxia	10	2	2	6	
<i>Total</i>	30				63.33 %

Tabla A.1. Matriz de confusión resultante al utilizar unidades de Llanto.

APÉNDICE B IMPLEMENTACIÓN DE UNA RED FEED FORWARD EN HANDEL-C

En una red neuronal feed forward (RNFF) Cada neurona contiene un número de entradas, las cuales son multiplicadas por un valor al cual se le nombra peso, el resultado de cada producto se adiciona con el siguiente, el resultado es pasado a través de una función de activación no lineal, en este caso se utilizó la función de activación sigmoideal debido a los resultados que arroja (Figura B.1).

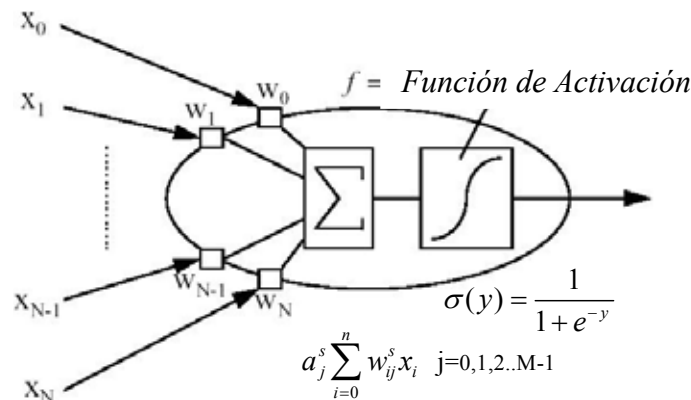


Figura B.1. Arquitectura de una neurona.

La red neuronal feed-forward implementada en Matlab, contiene 3 capas, una de entrada, una capa oculta y una de salida. Es fácil establecer el número de nodos que se utilizan a la entrada de la red ya que es el mismo que número de datos que contiene la muestra de entrada en este caso se utilizan 50 PCAs, utilizando muestras de 1 seg. y LPC como método de extracción de características de donde por cada muestras se obtienen 304 coeficientes. La capa de salida contiene tres neuronas, una para cada tipo de llanto (normal, sordo y asfixia). Establecer cuantas neuronas debe de tener la capa oculta es relativamente complejo ya que no existe una formula matemática con la cual se pueda conocer la cantidad de neuronas que deban de tener las capas oculta. En base a la experiencia se ha visto que la siguiente capa debe de ser del mismo tamaño o menor a la anterior. Se realizaron diferentes pruebas llegando a la conclusión que 15 neuronas en la capa oculta es la cantidad adecuada. El método utilizado en el entrenamiento fue el de Gradiente Conjugado Escalado. El resultado obtenido fue de 95.33% de precisión (Tabla B.1)

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	50	47	3		
Sordos	50		50		
Asfixia	50	4		46	
Total	150				95.33 %

Tabla B.1. Resultado obtenido en Matlab.

El modelo que implementado en Handel-C se muestra en la Figura B.2, el cual realiza todas las operaciones equivalentes a una capa de RNA.

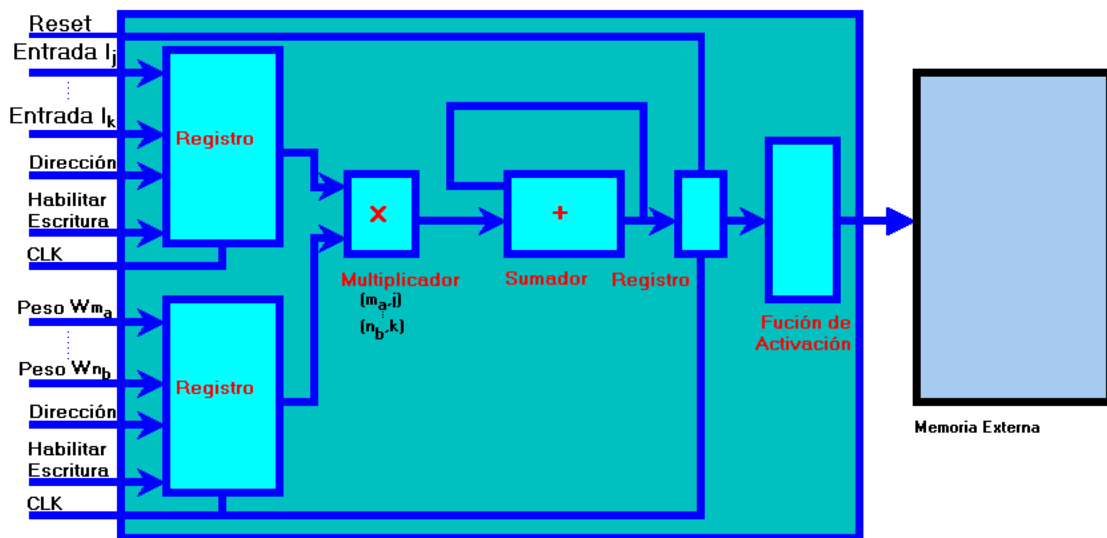


Figura B.2. Diagrama a bloques correspondiente a una capa de la RNA.

En la arquitectura propuesta se almacenan en un registro los valores de entrada correspondientes a la neurona m y se realizan las operaciones de multiplicación con sus respectivos pesos (almacenados en un registro diferente al de las entradas), el resultado de cada producto se guarda y se va sumando hasta realizar todas las operaciones para dicha neurona y el resultado se pasa por la función de activación, el valor de salida es guardado en una memoria externa, este valor si no corresponde a la última capa de neuronas, servirá como entrada para la siguiente capa. Este proceso se repite hasta completar las n neuronas contenidas en cada capa de la red. La función de activación sigmoideal es con la que se obtienen los mejores resultados y el problema ocasionado al tratar de implementar esta función fue la exponencial que contiene, como se puede observar en la ecuación B.1.

$$\sigma(y) = \frac{1}{1 + e^{-y}} \quad (\text{B.1})$$

Se probaron algunas alternativas como la interpolación y la aproximación, encontrando buenos resultados en la sig. serie:

$$e^y = 1 + \frac{y}{1!} + \frac{y^2}{2!} + \frac{y^3}{3!} + \dots + \frac{y^n}{n!} \quad (\text{B.2})$$

Utilizando n igual a 20 y la matriz de pesos entrenada de la RNFF por Matlab, se obtiene un resultado de 87.33% de eficacia al momento de clasificar (Tabla B.2).

Tipo de Llanto	Muestras	Matriz de Confusión			Clasificación
		Normal	Sordos	Asfixia	
Normal	50	43	4	3	
Sordos	50	3	46	1	
Asfixia	50	5	3	42	
<i>Total</i>	150				87.33 %

Tabla B.2. Resultado obtenido en Handel-C.

APÉNDICE C MATRIZ RELACIONAL Y VECTORES DE PRUEBA

En la Figura C.1 (a) se puede apreciar la matriz de pesos que se utiliza dentro del FPGA y en la Figura C.1 (b) se encuentra 94 vectores de prueba con las tres clases a clasificar. De esta manera fácilmente se pueden bajar los vectores a probar que se deseen o cambiar la matriz de pesos relacional $\mathbf{R}$ si es que se cuenta con una que proporcione mejores resultados.

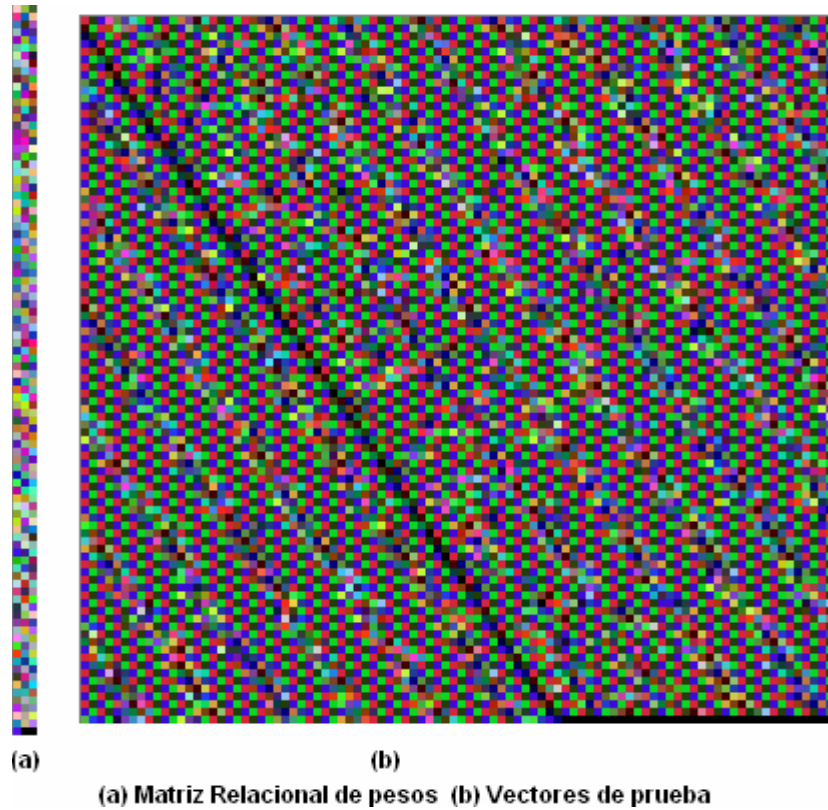


Figura C.1. Matriz de pesos y vectores de prueba en mapa de bits.

En las tres primeras columnas de la Tabla C.1 se puede ver la matriz original que se obtiene en Matlab después de que la red ha sido entrenada. Esta tiene un tamaño de 70×3 , ya que para cada elemento del vector de prueba le corresponde una entrada y el 3 es debido al número de clases a distinguir. En las siguientes tres columnas se puede ver la conversión de la matriz $\mathbf{R}$ en un formato punto flotante hexadecimal de 32 bits, del cual se parte para obtener la matriz de pesos que finalmente se descarga en la memoria del FPGA (Figura C.1 (a)).

-1591.9	-4.5053	17.272	C4C6FE3C	C0902B62	418A2CF4
-3.516	-0.48002	2.1754	C0610592	BEF5C578	400B393D
-0.8151	-0.004603	1.1107	BF50AAB8	BB96D519	3F8E2B4A
-0.51796	-0.018668	0.076058	BF049917	BC98EE4A	3D9BC46A
-0.1212	-0.052151	-0.025404	BDF83946	BD559BEE	BCD01BC5
0.0086765	-0.068262	-0.032475	3C0E27DF	BD8BCCC4	BD050476
0.91966	-0.05392	-0.044217	3F6B6EE5	BD5CDB48	BD351D61
0.81076	-0.040455	-0.019197	3F4F8E31	BD25B3F0	BC9D4326
1.078	-0.031702	-0.010075	3F89FC51	BD01DA00	BC250F93
1.1995	-0.024676	-0.0034216	3F998A27	BCCA2465	BB603D91
-0.42204	0.77015	-0.0039109	BED81633	3F452891	BB802742
-0.6072	0.2038	-0.0044154	BF1B716A	3E50B08F	BB90AF13
-0.58457	0.01735	-0.0057692	BF15A65C	3C8E229D	BBBD0BC1
-0.53899	0.0097623	-0.0060507	BF09FB75	3C1FF234	BBC644B2
-0.48046	0.0047612	0.0024659	BEF5FEAA	3B9C03E8	3B219B24
-0.46948	0.0036773	-0.0017153	BEF05F20	3B70FEBB	BAE0D374
-0.44253	-0.0073253	-0.012967	BEE2937C	BBF008D5	BC547198
-0.41892	-0.0051838	-0.016378	BED67C5E	BBA9DD39	BC862A20
-0.38298	-0.0080614	-0.016084	BEC4164C	BC041419	BC83C1EC
-0.35163	-0.011659	-0.012828	BEB40840	BC3F04F1	BC522D7D
-0.32403	-0.0050577	-0.0096193	BEA5E7C2	BBA5BB11	BC1D9A1B
-0.29958	-0.0036479	-0.0040342	BE9961E9	BB6F1162	BB84313F
-0.27775	-0.0068354	0.0012709	BE8E353A	BBDFFB83	3AA695DC
-0.26829	-0.0096658	-0.01457	BE895D09	BC1E5D40	BC6EB7BC
-0.26012	-0.012632	-0.016992	BE852E51	BC4EF853	BC8B336C
-0.25257	-0.0114	-0.019208	BE815154	BC3AC512	BC9D591E
-0.23655	-0.010079	-0.018895	BE723955	BC2520A5	BC9AC998
-0.2219	-0.0054608	-0.014258	BE633993	BBB2F03A	BC699BBB
-0.21631	-0.0045073	-0.014175	BE5D7FD1	BB93B260	BC683D03
-0.21143	-0.013314	-0.014104	BE588075	BC5A22A8	BC671494
-0.19935	-0.015288	-0.017991	BE4C23A0	BC7A796C	BC93625D
-0.18818	-0.017078	-0.017855	BE40B0F2	BC8BE7CC	BC9243BB
-0.17779	-0.015912	-0.015874	BE360F8A	BC825900	BC820AFD
-0.16813	-0.012026	-0.012201	BE2C29E9	BC450867	BC47E83B
-0.15911	-0.0083905	-0.0087535	BE22ED63	BC097856	BC0F6B00
-0.15067	-0.0075917	-0.0055108	BE1A49B1	BBF8C383	BBB49458
-0.14276	-0.0094208	-0.0023876	BE12308D	BC1A59A9	BB1C78DC
-0.15214	-0.013616	-0.0058285	BE1BC9A4	BC5F15EB	BBBEFD2F
-0.16103	-0.020027	-0.010632	BE24E651	BCA40EF2	BC2E329F
-0.15883	-0.021416	-0.010705	BE22A34A	BCAF70CB	BC2F6426
-0.15672	-0.022726	-0.010771	BE207A08	BCBA2C78	BC3079DA
-0.14963	-0.019472	-0.0079641	BE1937AF	BC9F8320	BC027BCE
-0.14292	-0.016384	-0.0052955	BE125890	BC8638A4	BBAD85A3
-0.13655	-0.013452	-0.0041037	BE0BD4F9	BC5C6582	BB867846
-0.13517	-0.012742	-0.0042945	BE0A6A82	BC50C539	BB8CB8C1
-0.13384	-0.012065	-0.0031888	BE090D7D	BC45AD9F	BB50FB33
-0.12814	-0.0094376	-0.0046663	BE0337F7	BC1AA000	BB98E811
-0.12272	-0.0069297	-0.0048487	BDFB5230	BBE31297	BB9EE179
-0.11754	-0.0064248	-0.0025885	BDF0B7FF	BBD2873D	BB29A48C
-0.1126	-0.0040917	-0.00042369	BDE6987B	BB8613E7	B9DE22B4
-0.10787	-0.0018584	0.0016518	BDDCEB8C	BAF3945F	3AD8810A
-0.11105	-0.0014503	0.0013668	BDE36EB3	BABE172B	3AB32769
-0.11416	-0.0010979	0.0010918	BDE9CBAF	BA8FE812	3A8F1B3B
-0.11341	-0.0024157	0.00082419	BDE84505	BB1E50C8	3A580E9A
-0.10904	0.002844	0.00060429	BDDF4DCA	3B3A6232	3A1E698C
-0.10483	0.0047281	-0.00062623	BDD6B267	3B9AEDDB	BA2429C2
-0.10079	0.0065407	0.001225	BDCE6D7A	3BD65327	3AA08EAB
-0.096912	0.0067788	0.0030093	BDC67A0A	3BDE20FF	3B45381F
-0.09654	0.0038659	0.00083091	BDC5B6F1	3B7D5B2C	3A59D183
-0.096175	0.0010949	-0.00032666	BDC4F79E	3A8F8423	B9AB4436
-0.092583	-0.00010346	-0.00047589	BDBD9C22	B8D8F760	B9F980FB
-0.089119	-0.0012521	0.0012154	BDB6841F	BAA41EED	3A9F4F6A
-0.085777	0.00047641	0.0028508	BDAFAC07	39F9C696	3B3AD4C6
-0.082551	0.0021471	0.004433	BDA9108D	3B0CB5EC	3B914270
-0.079435	0.0023821	0.0041843	BDA2AE9C	3B1C1C40	3B891CD9
-0.076422	-0.00011373	0.00048418	BD9C8353	B8EE815C	39FDD903
-0.082132	-0.002522	-0.003105	BDA834A7	BB254857	BB4B7D7D
-0.084809	-0.0035392	-0.0040747	BDADB07D	BB67F21E	BB85857A
-0.068134	-0.0019349	-0.0033563	BD8B89B7	BAFD9DA6	BB5BF4DB
-0.065486	-0.00037951	-0.0018239	BD861D81	B9C6F979	BAEF0F53

Tabla C.1. Matriz Relacional y su equivalencia en punto flotante de 32 bits

APÉNDICE D RESULTADOS DE LA IMPLEMENTACIÓN ELECTRÓNICA

En este apéndice se analizan los primeros 11 resultados correspondientes a los vectores de prueba de la Figura C.1 (b). Los resultados desplegados en el monitor por la RC100 se pueden observar en la Figura D.1.



Figura D.1. Resultado de 11 muestras analizadas por la red FNR implementada.

Los resultados de la Tabla D.1 se pueden comparar con los primeros 11 de la Tabla 5 y se puede ver que los resultados son similares, la diferencia es que los primeros fueron obtenidos de la implementación electrónica y los otros de la simulación en Matlab. En la imagen se puede apreciar que el display de la izquierda indica 3 y el de la derecha 1, debido a que la red ha fallado al momento de clasificar.

Número de Muestra	Resultado desplegado en el monitor	Conversión	Etiquetas
1	3D75C28F 3F452891 3DB851EB	0.0600 0.7702 0.0900	
	SORDO		SORDO
2	3D75C28F 3F452891 3DB851EB	0.0600 0.7702 0.0900	
	SORDO		SORDO
3	3F6B6EE5 3E50B08F 3C23D70A	0.9197 0.2038 0.0100	
	NORMAL		NORMAL
4	3F6B6EE5 3E199999 3C23D70A	0.9197 0.1500 0.0100	
	NORMAL		NORMAL
5	3F70A3D7 3CF5C28F 3C23D70A	0.9400 0.0300 0.0100	
	NORMAL		NORMAL
6	3D75C28F 3F452891 3DE147AE	0.0600 0.7702 0.1100	
	SORDO		SORDO
7	3F6B6EE5 3C8E229D 3C23D70A	0.9197 0.0174 0.0100	
	NORMAL		NORMAL
8	3D75C28F 3F452891 3DA3D70A	0.0600 0.7702 0.0800	
	SORDO		SORDO
9	3F6B6EE5 3DA3D70A 3C23D70A	0.9197 0.0800 0.0100	
	NORMAL		NORMAL
10	3C23D70A 3E50B08F 3F800000	0.0100 0.2038 1.0000	
	ASFIXIA		ASFIXIA
11	3F6B6EE5 3E50B08F 3C23D70A	0.9197 0.2038 0.0100	
	NORMAL		ASFIXIA

Tabla D.1, Análisis de los resultados obtenidos de la implementación electrónica.