INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Implementación de algoritmos de clasificación de patrones en FPGA.

T E S I S

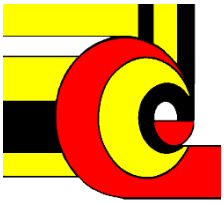
QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

SAMUEL MEJÍA ISLAS

DIRECTORES DE TESIS:

DR. AMADEO JOSÉ ARGÜELLES CRUZ
DR. ANTONIO HERNÁNDEZ ZAVALA



MÉXICO, D.F. ENERO DE 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 13:00 horas del día 13 del mes de noviembre de 2013 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Implementación de algoritmos de clasificación de patrones en FPGA"

Presentada por el alumno:

MEJÍA	**ISLAS**	**SAMUEL**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: **A 1 2 0 4 1 3**

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA
Directores de Tesis

Dr. Amadeo José Argüelles Cruz

Dr. Antonio Hernández Zavala

Dr. Oleksiy Pogrebnyak

Dr. Cornelio Yáñez Márquez

Dr. Oscar Camacho Nieto

M. en C. Romeo Urbieta Parrazales

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

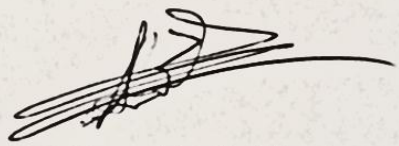
ESTADOS UNIDOS MEXICANOS
INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México D.F. el día 21 del mes de Noviembre del año 2013, el que suscribe Samuel Mejía Islas alumno del programa de Maestría en Ciencias de la Computación con número de registro A120413 adscrito a Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección de Dr. Amadeo José Argüelles Cruz y Dr. Antonio Hernández Zavala y cede los derechos del trabajo intitulado Implementación de algoritmos de clasificación de patrones en FPGA, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección smejiaislas@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Samuel Mejía Islas
Nombre y firma

Agradecimientos

Quiero agradecer a Dios por brindarme la vida y haberme brindado una familia que me ha ayudado a construir mis sueños en realidades y unos amigos que han marcado mi vida de una manera muy especial.

Formalmente quiero hacerle saber a mi núcleo familiar lo importante que son para mí, ya que sin ellos nada de todo lo que he logrado habría sido posible. Gracias.

Agradezco al Instituto Politécnico Nacional (COFAA, SIP y CIC) y al CONACyT por el apoyo recibido para la realización de este trabajo de tesis, de la misma forma que agradezco al pueblo de México por brindarme la oportunidad de continuar con mi preparación académica.

Es necesario dar reconocimiento a las grandes personas que han sido parte de mi viaje en el Centro de Investigación en Computación, ya que cada uno de ustedes me ha ayudado a descubrir que soy una persona con grandes aspiraciones y que los limites son una barrera imaginaria que puede ser derribada por el empeño y la constancia. Especialmente Joel Francisco Reyes gracias por brindarme una amistad sincera llena de lecciones de vida y humildad.

A mis directores de Tesis por el gran apoyo que me brindaron para la realización de este trabajo. Al Dr. Amadeo José Argüelles Cruz y al Dr. Antonio Hernández Zavala por la guía que me han brindado. A mis sinodales por sus valiosas aportaciones y críticas que han enriquecido este trabajo de tesis.

Resumen

En el presente trabajo de tesis se presenta una arquitectura con la que se realizaran las operaciones del clasificador Gamma mismo que se ubica dentro del enfoque asociativo de reconocimiento de patrones.

Se incluyen también las definiciones que permiten que el algoritmo funcione correctamente, así mismo se describe las etapas que requiere la arquitectura para entregar el resultado de clasificación.

Además se presenta un estudio experimental del desempeño del algoritmo incrustado en la arquitectura propuesta, comparando su rendimiento de clasificación, con el resultante de los otros clasificadores, trabajando con diversas bases de datos de acceso público ubicadas en el repositorio de la UCI *machine learning* y la página web del SIMAT (Sistema de Monitoreo Atmosférico de la Ciudad de México).

En los experimentos se muestra lo competitivo del clasificador, aunado se muestra un experimento en el cual el clasificador se emplea para la predicción de una serie de tiempo.

Con este trabajo de tesis se amplía la implementación de algoritmos del Enfoque Asociativo de Clasificación de Patrones haciendo uso de la tecnología FPGA (*Field Programmable Gate Array*); al mismo tiempo que se emplea un algoritmo de alto desempeño.

Abstract

In the current document of thesis an architecture that perform the same operations in Gamma Classifier is presented, which belongs to Associative Approach of Pattern Recognition.

Definitions allow the algorithm to operate properly, also describes the steps required by the architecture to deliver the result of classification.

Besides presents an experimental study of the performance of the algorithm embedded in the proposed architecture; classification performance is compared with the results from other classifiers with many databases located in the UCI machine learning repository and SIMAT (*Sistema de Monitoreo Atmosférico de la Ciudad de México*) web site.

The experiments shown which competitive is the classifier, also shown an experiment where the classifier is used to predict a time series.

This thesis extends the implementation of algorithms Associative Approach Pattern Classification technology using FPGA (Field Programmable Gate Array) while employing a high performance algorithm.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

Este trabajo trata sobre la implementación de un clasificador de patrones del enfoque asociativo en un FPGA (*Field Programmable Gate Array*), usando el lenguaje de programación VHDL (VHSIC —*Very High Speed Integrated Circuit*— HDL —*Hardware Description Language*).

1.1 Antecedentes

Los dispositivos lógicos programables, desde su creación física no contienen una función establecida; un contraejemplo de estos dispositivos, son las compuertas lógicas que tienen definida una función desde su fabricación. Antes de emplear un dispositivo lógico programable en cualquier aplicación, este debe de ser programado [1], [3], [4].

Las aplicaciones implementadas en dispositivos lógicos programables en estos días son innumerables, ya que la tendencia para resolver diversos problemas ha recaído en el desarrollo de esta tecnología [5].

Como respuesta a diversas problemáticas se han realizado avances en el desarrollo de estos dispositivos [6], con la misión de brindar cada vez mayores prestaciones, con el objetivo de implementar en estos dispositivos técnicas de diferente índole como la implementación de lógica difusa en un micro-controlador [7] o un sistema de decodificación de una cadena de bits [8] o un sintetizador de frecuencias [9].

De los dispositivos lógicos programables que nos interesa conocer su funcionamiento es el del FPGA, que se soporta en el álgebra de Boole y son constituidos por una matriz de celdas lógicas. El fin de este dispositivo, es que el cliente realice la programación del dispositivo y no el fabricante, dicha programación es realizada usualmente a través de una interfaz entre el software de la computadora y el hardware del FPGA.

Los dispositivos FPGA en la actualidad son reprogramables y su configuración normalmente es guardada en una memoria EEPROM (*Electrically Erasable Programmable Read-Only Memory*).

Por sus prestaciones los FPGA han sido ocupados en diversas aplicaciones y áreas de trabajo como lo son:

- Clasificación de células cancerígenas [10]
- Métodos de inferencia [11]
- Redes neuronales [12], [15], [16]
- Implementación de lógica difusa [56]
- Memorias Asociativas [13], [14]
- Manejo de Señales [18], [24], [27], [29]
- Implementaciones con entradas variables [20], [22] [26]
- Modelado computacional [23], [25]
- Modelos predictivos [28]

Aplicaciones como las realizadas anteriormente requieren que el cliente escoja un lenguaje de programación conocido como descripción de hardware de alto nivel. Existen varios lenguajes de descripción de *hardware*, uno de los más populares es VHDL que es una combinación de VHSIC—HDL. Ese lenguaje fue homologado por la IEEE y uso principal es el describir circuitos digitales [33], [34], una de sus opciones es describir la operación del circuito digital desde un diseño esquemático [35] he aquí algunas de sus aplicaciones:

-- Descripción de algoritmos paralelos. [36]
-- Descripción de entrenamiento de redes neuronales. [37]
-- Detección de movimiento en video en tiempo real. [38]

De las diferentes aplicaciones implementadas en los FPGAs que se han generado a lo largo de los años, se han ocupado miles de diferentes técnicas, entre ellas han sido empleados los clasificadores de patrones.

Dentro del enfoque de los clasificadores de patrones es importante resaltar, que una de las mejores habilidades del ser humano es el proceso de reconocer cosas, como un juguete, un color, una persona familiar y esto lo realizamos de forma inconsciente, sin embargo nos facilita nuestra forma de vivir y relacionarnos.[2]

Hemos de recordar que el objetivo de las computadoras a sus inicios era simular un cerebro humano, para que realizara tareas tediosas y de una alta complejidad de manera que nos redujera la carga de trabajo y pudiéramos hacer múltiples tareas simultáneamente, sin embargo en el proceso de reconocer existen muchas variantes, debido a ellas actualmente no existe un clasificador que tenga un rendimiento del 100% con cualquier tipo de banco de datos.

Por esta necesidad es que existen diferentes enfoques del reconocimiento de patrones:

- Enfoque estadístico-probabilístico. El principal clasificador de este enfoque es el bayesiano y como su nombre lo indica se basa en el teorema de Bayes.

- Clasificadores basados en métricas. Son basados en métricas y espacios métricos, que brindan la información del objeto, para realizar una clasificación.

- Enfoque sintáctico-estructural. Se basa en la teoría de autómatas y lenguajes formales para hacer la clasificación. Se enfoca principalmente en la estructura del objeto a clasificar, a diferencia de otros que emplean mediciones numéricas.

- Enfoque neuronal. Es basado en modelos matemáticos de neuronas que emulan el comportamiento o interacción de las neuronas de un ser racional. [17], [19]

- Enfoque Asociativo. Está soportado en un trabajo original que nació en el Centro de Investigación en Computación del IPN en 2002, donde se implementan los modelos de memorias asociativas para crear clasificadores robustos. [48]

Conocidas las dos ramas, FPGA y reconocimiento de patrones, el trabajo aquí presente bifurcará en un clasificador de patrones del enfoque asociativo y la implementación de esas operaciones en un FPGA, el cual es limitado a ciertas dimensiones de información.

1.2 Justificación

En la literatura científica se reportan trabajos en donde los clasificadores de patrones se relacionan con los FPGA [39], con base en esos trabajos se ha observado la necesidad de realizar diferentes variantes, con el fin de obtener un mejor resultado, puesto que no existe el clasificador perfecto *(NO FREE LUNCH)* [2], por ello es importante destacar que en este trabajo se implementará un algoritmo del enfoque asociativo en la búsqueda de una arquitectura que brinde los resultados de clasificación de dicho algoritmo.

1.3 Objetivo general

Generar una arquitectura de cómputo que permita la implementación de un clasificador de patrones del enfoque asociativo en un dispositivo de lógica reconfigurable FPGA, con el uso del lenguaje de descripción VHDL y que funcione como un clasificador de uso general.

1.4 Objetivos específicos

1. Realizar una descripción de la relación existente entre la dimensionalidad del banco de datos y la arquitectura que será creada.
2. Revisar la estructura del algoritmo seleccionado e identificar, que operaciones se pueden realizar en paralelo dentro de la arquitectura.
3. Implementar el algoritmo de clasificación de patrones, en una arquitectura de cómputo reconfigurable FPGA.
4. Comparar los resultados obtenidos, con otros mencionados en el estado del arte.

1.5 Contribuciones

Las contribuciones de esta tesis son:

- Desarrollo de una arquitectura donde se implementan las operaciones de un algoritmo de clasificación de patrones basado en el enfoque asociativo.
- Estudió donde se muestran los resultados de clasificación de la arquitectura en comparativa con algunos otros algoritmos de clasificación.

1.6 Organización del documento

En este capítulo se han presentado: los antecedentes, la justificación, el objetivo del presente trabajo de tesis y las contribuciones del mismo. El resto del documento de tesis está organizado de la siguiente manera:

En el capítulo 2 se presenta el estado del arte, donde se muestra un panorama general de los FPGA y los clasificadores de patrones.

El capítulo 3 presenta todas las herramientas (materiales y métodos) que van a ser utilizadas para el desarrollo del presente documento.

El capítulo 4 es la parte central de este trabajo. Es aquí, donde se describen los elementos de la arquitectura propuesta que es capaz de llevar a cabo la tarea de clasificación.

En el capítulo 5 se muestran los experimentos y resultados de la arquitectura propuesta, y un estudio comparativo respecto del rendimiento de otros clasificadores.

El capítulo 6 incluye las conclusiones, las aportaciones y el trabajo a futuro que se propone desarrollar. Finalmente se incluye un apéndice con la descripción de la arquitectura y las referencias.

Capítulo 2

Estado del arte

Dentro de este capítulo se contienen 4 secciones. En la sección 2.1, se muestran algunos acontecimientos históricos, así como ejemplificaciones de los desarrollos logrados con la tecnología de los FPGA y conceptos relacionados. La sección 2.2, abordará los principales modelos asociativos y en la sección 2.3 se describirá una herramienta llamada WEKA (*Waikato Environment for Knowledge Analysis*) y los algoritmos que servirán en la comparativa de resultados, finalmente en la sección 2.4 se contextualizará en el desarrollo de sistemas dedicados.

2.1 FPGA

Transcurriendo el año de 1983 es cuando la empresa *International CMOS Technology* (I.C.T.) desarrolla un dispositivo PEEL (*Programable Electrically Erasable Logic*) con tecnología especializada en el borrado de CMOS de manera eléctrica y este dispositivo fue llevado a la producción en el año de 1986. Durante el año de 1984 se anunció un nuevo concepto, que revolucionaría la forma de ver los PLD (P*rogrammable Logic Device*) por la empresa *Xilinix Corporation* el dispositivo que crearon fue la LCA (*Logic Cell Array*) que es compuesto de pequeñas celdas lógicas, donde cada celda es capaz de crear cuatro o cinco funciones de entrada y dos de salida, es cuando esta empresa tomo el liderato en el mercado de los FPGA.

La herramienta que representa el FPGA, es de alcances desconocidos, ya que día con día las aplicaciones desarrolladas en esta tecnología dirán hasta donde llega su verdadero potencial, dado que existen miles de desarrollos con diferentes fines y objetivos. Por ejemplo de las aplicaciones que nos competen, son la implementación de operaciones y existen trabajos como: un perceptron multicapa que es dependiente de múltiples pesos y una tolerancia neuronal digital [12]; como una memoria asociativa basada en la estabilidad celular de una red neuronal [16], un modelo de aprendizaje basado en una memoria asociativa con una eficiente implementación en *Hardware* [14], sin embargo existen implementaciones con mayor afinidad como el modelo de una memoria asociativa alfa-beta en paralelo dentro de una FPGA [13]. Estos son los trabajos donde existe una beta de investigación, ya que en la implementación de este tipo de modelos es posible implementar mejoras realizando arquitecturas específicas.

Es importante conocer, que se requiere de una representación física de las operaciones requeridas, para cualquier tipo de aplicación en los FPGA, a ese dispositivo electrónico se le conoce como puerta lógica o compuerta lógica.

Una sola compuerta lógica tiene una función, la cual forma parte del álgebra de Boole, que es la teoría matemática aplicada en la lógica combinatoria. Los valores que se pueden asignar a una variable booleana son: 1 (valor alto) ó 0 (valor bajo) y estos valores son utilizados para representar magnitudes lógicas.

Los operadores lógicos que tienen mayor relevancia para este trabajo son: "*OR*", "*AND*" y "*COMPLEMENTO*".

La operación lógica *OR* es representada mediante una tabla de verdad, en la que muestra cómo se realiza la operación con 2 entradas (**A** y **B**) y el resultado de la salida, dependerá totalmente de la combinación de las entradas.

Tabla 2.1: Operación lógica *OR*

ENTRADA A	**ENTRADA B**	**SALIDA A∨B X**
0	0	0
1	0	1
0	1	1
1	1	1

La compuerta lógica *OR* esquemáticamente se representa de la siguiente forma.

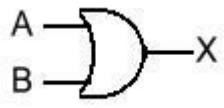


Figura 2.1 *Compuerta lógica OR*

La operación lógica *AND* es representada mediante una tabla de verdad, en la que muestra cómo se realiza la operación con 2 entradas (**A** y **B**) y el resultado de la salida, dependerá totalmente de la combinación de las entradas.

Tabla 2.2: Operación lógica *AND*

ENTRADA A	ENTRADA B	SALIDA A∧B X
0	0	0
1	0	0
0	1	0
1	1	1

La compuerta lógica *AND* esquemáticamente se representa de la siguiente forma.

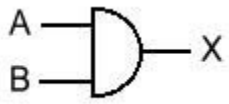


Figura 2.2 *Compuerta lógica AND*

La operación lógica *COMPLEMENTO* es representada mediante una tabla de verdad, en la que muestra cómo se realiza la operación con 1 entrada (**A**) y el resultado de la salida, que es el complemento de la entrada.

Tabla 2.3: Operación lógica *COMPLEMENTO*

ENTRADA A	SALIDA X
0	1
1	0

La compuerta lógica *COMPLEMENTO* esquemáticamente se representa de la siguiente forma.

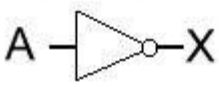


Figura 2.2 *Compuerta lógica COMPLEMENTO*

2.2 Modelos Asociativos

A continuación se presenta una breve historia de los modelos asociativos, así como los modelos más representativos para este trabajo.

En 1961 fue propuesto el primer modelo de memoria asociativa por Karl Steinbuch llamada la *Lernmatrix* [39] que es una memoria heteroasociativa, que trabaja con los patrones en su forma binaria.

Los escoceses Willshaw, Buneman y Longuet-Higgins en el año 1969 presentaron el *Correlograph* [40], un dispositivo óptico capaz de funcionar como una memoria asociativa.

Teuvo Kohonen [41] y James A. Anderson [42] generaron 2 trabajos que debido a su gran similitud, es reconocido con el nombre genérico de *Linear Associator* esto fue en el año de 1972, justo en ese año el profesor Shun-ichi Amari realizo aportes teóricos al área [43], su compatriota Kaoru Nakano mostro al mundo el *Associatron* [44], siendo este uno de los años con mayor actividad para el área de memorias asociativas.

La memoria de *Hopfield* [45], es uno de los modelos más conocidos de las memorias asociativas, aun cuando la memoria de Hopfield tiene un pobre desempeño de recuperación comparado con diferentes algoritmos.
Este trabajo tomó aportaciones del profesor Shun-ichi Amari que son fundamentales para el funcionamiento del modelo.

Uno de los modelos de memoria heteroasociativa con base en el modelo de *Hopfield* fue la memoria asociativa bidireccional *BAM*, presentada por Bart Kosko en 1988 [46], tuvo que pasar un década para que Ritter [47], presentará su modelo de memoria asociativas morfológicas, que al incorporar herramientas de la morfología matemática incrementó las capacidades de aprendizaje de los modelos conocidos hasta ese momento.

El enfoque asociativo fue creado en el año de 2002 por el grupo Alfa-Beta en el Centro de Investigación en Computación del IPN, engrosando así el área de reconocimiento de patrones.

El enfoque asociativo, se inició con el CHAT (Clasificador Híbrido Asociativo con Traslación) [57], desarrollado en 2002 por Raúl Santiago Montero en el CIC-IPN, como tema central de su tesis de maestría. Este enfoque se basa en las memorias asociativas, cuyo estado del arte incluye a las memorias asociativas Alfa-Beta [48].

Uno de los modelos del grupo Alfa-Beta que hace uso en el estado del arte de las memorias asociativas es el modelo asociativo Gamma [51], desarrollado en 2007 y mejorado en 2011 por Itzamá López Yáñez, siendo este el que se pretende implementar en la arquitectura propuesta en este trabajo.

2.3 Algoritmos de WEKA

En el capítulo 5, se presentan los resultados obtenidos con el modelo propuesto en este trabajo de tesis y se comparan dichos resultados con algunos algoritmos implementados en WEKA.

En la presente sección se realiza una descripción de dicho software y de los algoritmos utilizados para la comparación. El contenido de esta sección se encuentra basado en [59].

WEKA es una colección de algoritmos de aprendizaje de máquina y herramientas para el pre-procesamiento de datos. WEKA incluye los principales métodos para problemas de minería de datos, tales como: regresión, clasificación, clustering y selección de atributos. WEKA permite pre-procesar un banco de datos, introducirlo en un esquema de aprendizaje y analizar los resultados y el desempeño del clasificador resultante.

En seguida se realiza una breve descripción de los algoritmos que se utilizarán en el capítulo 5, para comparar con los resultados obtenidos por el modelo propuesto en este trabajo de tesis.

2.3.1 IBk

El IBk es un clasificador de k vecinos más cercanos. Una variedad de algoritmos de búsqueda pueden ser usados para agilizar la tarea de encontrar los vecinos más cercanos: búsqueda lineal (método por defecto), *kD-trees*, *ball-tress* y *cover-trees*.
La función de distancia es un parámetro que depende del método de búsqueda, el usado por defecto es la distancia euclidiana, pero existen otros distancias: Chebyshev, Manhattan y Minkowski. Por defecto el número de vecinos es $k = 1$.

2.3.2 RepTree

RepTree es un algoritmo de aprendizaje que construye un árbol de decisión usando reducción de ganancia/varianza de la información y poda basada en la reducción del error. Debido a que este algoritmo se encuentra optimizado para ser rápido, ordena los valores numéricos solo una vez. Los valores perdidos, se manejan dividiendo las instancias existentes en segmentos; al igual que lo hace el C4.5. Los parámetros que pueden ser establecidos son: número mínimo de instancias en cada hoja, la profundidad máxima del árbol y la cantidad de datos usada para la poda.

2.3.3 MultiLayerPerceptron

MultiLayerPerceptron (MLP) es una red neuronal que se entrena usando *backpropagation*. Los nodos de la red son sigmoidales, excepto los nodos de salida para clases numéricas que son transformados automáticamente en unidades lineales sin umbral. En WEKA este método tiene su propia interfaz de usuario, que permite alterar la estructura de la red. También se pueden ajustar otros parámetros como: velocidad de aprendizaje (*learning rate*), momentum y épocas.

2.4 Desarrollo de sistemas dedicados

Para comprender el desarrollo de sistemas dedicados es importante conocer los 3 procesos interactivos para el desarrollo de un sistema dedicado que son el modelado, el diseño y el análisis. [49], [50]

2.4.1 Modelado

El modelado es el proceso donde se deberá obtener la idea concreta del sistema, contemplando todas las variantes a su alrededor por medio de una imitación. Estas imitaciones brindaran las propiedades del sistema así como la especificación de lo que el sistema es capaz de hacer.

En otras palabras es preciso tener en cuenta el área de aplicación, en la cual se requiere el diseño de un sistema dedicado, ya que se deben determinar objetivos específicos para el proyecto y estimar su viabilidad. Para esto es necesario determinar los requisitos funcionales básicos de la aplicación, así como los requisitos no funcionales, algunos de las cuales se derivaran directamente de la aplicación y algunos de otros factores, tales como la comercialización.

2.4.2 Diseño

El diseño es el proceso de creación de manera estructurada de artefactos. En esta fase se especifica cómo es que el sistema desarrollara el trabajo.

Para realizar esta etapa de diseño se suelen emplear metodologías de diseño que entre las más conocidas figuran “cascada” y “espiral”. [49]

2.4.2.1 “Modelo cascada”

El modelo de cascada se divide en cinco grandes etapas: requisitos, especificaciones, arquitectura, codificación y mantenimiento. La aplicación final será refinada sucesivamente a través de estas etapas, con el

mantenimiento, incluida la entrega de la aplicación final y posibles correcciones o mejoras de esta. [50]

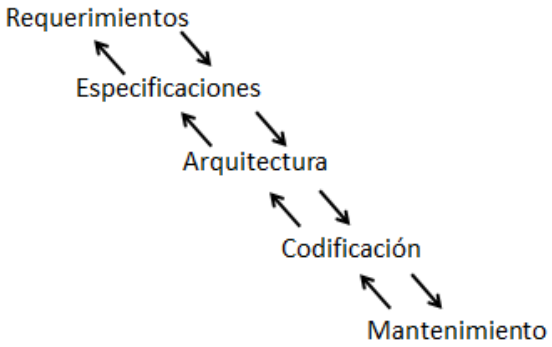


Figura 2.4 *Representación del modelo cascada.*

La mayor parte de la información contenida en esta metodología fluye desde la parte superior, es decir, que el flujo será de las etapas más abstractas hacia las etapas más concretas, aunque parte de la información puede fluir de nuevo a una etapa anterior para mejorar el diseño.

En la práctica de este modelo, los diseñadores pueden y deben utilizar la experiencia de las etapas de diseño y de esta forma reconsiderar las decisiones anteriores, y volver a hacer algo de trabajo.

2.4.2.2 "Modelo espiral"

El modelo en espiral, fue una manera de mejorar el modelo cascada por medio de una acción denomina refinamiento.

En este modelo se contempla que el proceso de diseño sea iterativo, de esa forma existirán varias versiones de la aplicación final, donde cada una de estas deberá ser mejor que la anterior. [50]

De tal modo que los diseñadores deberán cumplir con los 3 aspectos fundamentales de este modelo:

- Requisitos
- Arquitectura
- Codificación

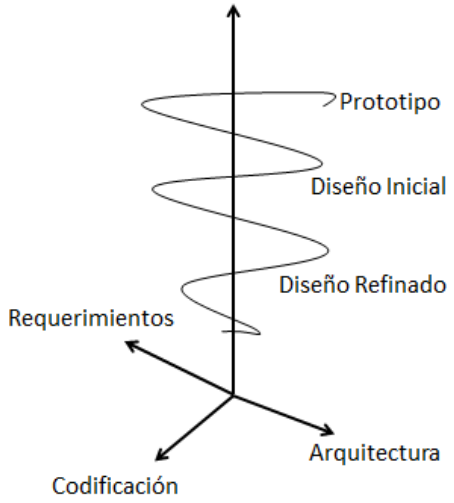


Figura 2.5 *Representación del modelo espiral.*

Toda vez que ocurra esto, los resultados de cada ciclo servirán como la guía para la próxima toma de decisiones antes de iniciar el próximo ciclo de desarrollo. La experiencia de una etapa debe ayudar a producir un mejor diseño en la siguiente etapa y permitir que el equipo de diseño cree el diseño mejorado de forma más rápida en comparación a la etapa previa.

2.4.3 Análisis

En el análisis se debe obtener una compresión profunda y detallada del comportamiento del sistema, si es que cumple con sus funciones que debería hacer de acuerdo a lo reportado en el modelado previo.

Dentro del análisis existen 3 métodos principales:

- Análisis estructurado

  Este método tiene como fin brindar las especificaciones que son requeridas por el sistema o la aplicación, ya que el desarrollo de esas especificaciones podría servir para el desarrollo de nuevos sistemas o efectuar modificaciones positivas a los ya existentes.

- Tablas de decisión

  Este método es elaborado por una matriz que describa las condiciones que se presentan en el sistema, aunado a las reglas de decisión que establecen o describen las acciones en las que desembocará el sistema.

- Arboles de decisión

  El árbol de decisión es un método que es representado por un diagrama secuencial, en que el que se describen las condiciones e indicando cual

será el camino de acciones que deberá de seguir el sistema, enumerando cual es la importancia de las condiciones; siendo indispensable enmarcar que el camino seguido dependerá del valor actual de la variable evaluada en ese instante.

Capítulo 3

Materiales y métodos

En este capítulo se describen los operadores Alfa y Beta, el operador u_β, la operación módulo, el algoritmo del código Johnson-Möebius modificado, que son indispensables para la descripción del clasificador Gamma que será utilizado para la propuesta de este trabajo.

Posteriormente en la sección 3.6 se expone los métodos de validación realizando énfasis en el que será empleado en el capítulo 4. En la sección 3.7 se explica los bancos de datos seleccionados para realizar las pruebas, después se explica la formación del conjunto de prueba, el conjunto de entrenamiento y las clases para una serie de tiempo. Finalmente en la sección 3.11 se expondrá el dispositivo FPGA donde se desarrollarán las pruebas correspondientes del trabajo presente.

3.1 Operadores Alfa y Beta

En esta sección se presentan los operadores fundamentales de los modelos asociativos Alfa-Beta, ya que son fundamentales para la implementación del clasificador Gamma, que será empleado en la propuesta de este trabajo de tesis.

Los operadores descritos en esta sección son la base para las memorias asociativas Alfa-Beta, presentados en [48].

A continuación se incluyen las tablas que representan a dichos operadores, dados los conjuntos:

$$A = [0; 1] \text{ y } B = \{0; 1; 2\}$$

Tabla 3.1: Definición del operador alfa.

$a: A \times A \rightarrow B$		
x	y	$a(x,y)$
0	0	1
0	1	0
1	0	2
1	1	1

Tabla 3.2: Definición del operador beta.

$\beta: B \times A \rightarrow A$		
x	y	$\beta(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1
2	0	1
2	1	1

3.2 Operador u_β

El operador aquí presentado fue definido en [51], ya que también es utilizado por el clasificador Gamma.

Definición 3.1 *Sean: el conjunto A = {0; 1}, un numero* $\boldsymbol{n} \in \mathbb{Z}^+$ y $\boldsymbol{x} \in A^n$ *un vector binario de dimensión n, con la i-ésima componente representada por* x_i. *Se define el operador* u_β(x) *de la siguiente manera:* u_β(x) tiene como argumento de entrada un vector binario n-dimensional x y la salida de un número entero no negativo que se calcula así:

$$u_\beta(\mathrm{x}) = \sum_{i=1}^{n} \beta\,(x_i\,, x_i)$$

Ejemplo 3.1 Sea x= $\begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}$; obtener u_β(x).

Dada la definición 3.1 $u_\beta(\mathrm{x}) = \sum_{i=1}^{5} \beta\,(x_i\,, x_i)$, por lo que:

u_β(x)= $\beta(0,0) + \beta(1,1) + \beta(1,1) + \beta(0,0) + \beta(1,1)$ = 0+1+1+0+0+1= 3
Entonces, u_β(x)= 3

Ejemplo 3.2 Sea x= $\begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$; obtener u_β(x).

Dada la definición 3.1 $u_\beta(\mathrm{x}) = \sum_{i=1}^{7} \beta\,(x_i\,, x_i)$, por lo que:
$u_\beta(\mathrm{x}) = \beta(1,1) + \beta(1,1) + \beta(0,0) + \beta(1,1) + \beta(0,0) + \beta(0,0) + \beta(1,1)$
= 1+1+0+1+0+0+1= 7
Entonces, $u_\beta(\mathrm{x}) = 7$.

3.3 Módulo

El contenido de esta sección fue tomado de [52]. El operador u_β de la sección anterior junto con el concepto de módulo, son relevantes en el desarrollo del clasificador Gamma.

Ya que hay situaciones en las que al utilizar el operador de división acostumbrado, resulta de más interés el residuo (entero) de dicha operación que el resultado (fraccional) mismo. Para resolver este problema existe el operador *módulo,* denotado por **mod**.

Definición 3.2 *Sean a un número entero y m un número entero positivo. Se denota por* α **mod** m *al residuo de dividir* α *por m.*

Dicho de otra manera α **mod** m es el número entero r tal que
$\alpha = qm + r$ y $0 \leq r < m$

Ejemplo 3.3 Se puede ver que 17 **mod** 5 = 2, 133 **mod** 9 = 7 y 2001 **mod** 101 = 82.

3.4 Código binario Johnson-Möebius modificado

Dado que los vectores con que trabaja el clasificador Gamma están codificados con el código Johnson-Möebius modificado, que a continuación es descrito, además se muestran algunos ejemplos numéricos [53]

Algoritmo 3.1 *Algoritmo del código Johnson-Möebius Modificado:*

1. Sea un conjunto de números reales

$$\{\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots\, \boldsymbol{r_i}, \ldots\, \boldsymbol{r_n}\}$$

donde n es un número entero positivo fijo.

2. Si uno de los número del conjunto (por ejemplo $\boldsymbol{r_i}$) es negativo, crear un nuevo conjunto transformado a través de la operación "restar $\boldsymbol{r_i}$ a cada uno de los n números"

$$\{\boldsymbol{t_1}, \boldsymbol{t_2}, \dots\ \boldsymbol{t_i}, \dots\ \boldsymbol{t_n}\}$$

donde $t_j = r_j - r_i \; \forall \, j \in \{1, 2, \dots, n\}$ y particularmente $t_i = 0$. Nota: si hay más de un negativo, se trabaja con el menor.

3. Escoger un número fijo d de decimales y truncar cada uno de los números del conjunto transformado (los cuales son no negativos) precisamente a d decimales.

4. Realizar un escalamiento de $10d$ en el conjunto del paso 3, para obtener un conjunto de n enteros no negativos

$$\{\boldsymbol{e_1}, \boldsymbol{e_2}, \dots\ \boldsymbol{e_i}, \dots\ \boldsymbol{e_n}\}$$

donde e_m es el número mayor.

5. El código Johnson-Möebius modificado para cada j = 1, 2, …, n se obtiene al generar $(\boldsymbol{e_m} - \boldsymbol{e_j})$ ceros concatenados por la derecha con $\boldsymbol{e_j}$ unos.

Ejemplo 3.4 Sea el conjunto $r = \{4.1, -0.4, 2.845, 1.6\}$; $r \in \mathbb{R}$.

Paso 1: $r = \{4.1, -0.4, 2.845, 1.6\}$.

Paso 2: Existe un número negativo (-0.4)

4.1 - (-0.4) = 3.7
-0.4 - (-0.4) = 0
2.845 - (-0.4) = 2.445
1.6 - (-0.4) = 1.2

Por lo que se obtiene el conjunto transformado $t = \{3.7,\ 0, 2.445, 1.2\}$.

Paso 3: Se escoge el número fijo d = 1 para obtener $t = \{3.7,\ 0, 2.4, 1.2\}$.

Paso 4: Se realiza el escalamiento de $10d$ para obtener $e = \{37,\ 0, 24, 12\}$, donde $e_m = 37$.

Paso 5: Para cada número $\boldsymbol{e_i}$ del conjunto e se generan $e_m - e_i$ ceros concatenados con $\boldsymbol{e_i}$unos.

1. Cada número será codificado con 37 bits.
2. Para el número $\boldsymbol{e_1}$ = 37, se tendrán 37 - 37 = 0 ceros concatenados de 37 unos.
3. Para el número $\boldsymbol{e_2}$= 0, se tendrán 37 - 0 = 37 ceros concatenados de 0 unos.
4. Para el número $\boldsymbol{e_3}$= 24, se tendrán 37 - 24 = 13 ceros concatenados de 24 unos.
5. Para el número $\boldsymbol{e_4}$ = 12, se tendrán 37 - 12 = 25 ceros concatenados de 12 unos.

La tabla 3.1 muestra los códigos correspondientes del conjunto *e*.

Tabla 3.3: Codificación obtenida para el ejemplo 3.4

Número	Códigos Johnson-Möebius modificado
37	1111111111111111111111111111111111111
0	0000000000000000000000000000000000000
24	0000000000000111111111111111111111111
12	0000000000000000000000000111111111111

3.5 Clasificador Gamma

Este modelo asociativo es un clasificador de alto desempeño presentado inicialmente en [51]. Hace uso de los operadores Alfa y Beta, descritos en la sección 3.1; el operador $\boldsymbol{u_\beta}$, descrito en la sección 3.2 y requiere que los patrones sean codificados en código Johnson-Möebius modificado algoritmo que fue descrito en la sección 3.4 y en esta sección se añade un elemento más a su composición: el operador gamma (γ) de similitud el cual es presentado igualmente en [51].

En esta sección se describe tanto el operador gamma (γ) de similitud como el correspondiente algoritmo para el desarrollo del clasificador Gamma.

3.5.1 Operador Gamma (γ) de similitud

De este operador resaltan las siguientes características:

1. Está basado en las operaciones fundamentales que dan lugar a las memorias asociativas Alfa-Beta (descritas en la sección 3.1).

2. El operador tiene la tarea de observar si dos vectores son parecidos o no, mediante un grado de disimilitud θ, indicando la tolerancia en

que al comparar los vectores sean considerados similares, no obstante que son diferentes.

3. El empleo de este operador permite trabajar con vectores binarios de dimensiones diferentes.

4. El nombre se le dio en base a los operadores Alfa-Beta, siendo gamma (γ) la letra griega que está a continuación de las otras dos.

Formalmente el operador se define de la siguiente manera: [51]

Definición 3.3 *Sean: el conjunto* $A = \{0, 1\}$*, un número* $n \in \mathbb{Z}^+$ $x, y \in A^n$ *dos vectores binarios de n-dimensionales, con la i-ésima componente representada por* x_i, y_i*, respectivamente, y además,* θ *un número entero no negativo. Se define el operador Gamma de similitud* $\gamma(x, y, \theta)$ *de la siguiente manera:* $\gamma(x, y, \theta)$ *tiene como argumentos de entrada dos vectores binarios n-dimensionales x, y, un número entero no negativo* θ*, y la salida es un número binario que se calcula así:*

$$\gamma(x, y, \theta) = \begin{cases} 1 \; si \; n - u_\beta[\alpha(x, y) \bmod 2] \leq \theta \\ 0 \; en \; otro \; caso \end{cases}$$

A continuación se muestran algunos ejemplos que emplean el operador Gamma de similitud.

Ejemplo 3.5 *Sean* $x = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$, $y = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ y $\theta = 3$*; calcular* $\gamma(x, y, \theta)$.

Paso 1 En este ejemplo se tiene que $n = 5$.

Paso 2 Calculando $a(x, y)$, obtenemos: $= \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Paso 3 Al aplicar el módulo 2 a cada elemento del vector resultante en el paso anterior, se obtiene el vector siguiente: $= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$

Paso 4 Aplicando el operador u_β en el vector del paso 3, se obtiene: $u_\beta = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 2$.

Del resultado obtenido, se realiza la resta con n = 5.

$$5 - 2 = 3$$
$$3 \leq \theta = 3$$

por lo tanto $\gamma(x, y, \theta) = 1$.

3.5.2 Algoritmo del Clasificador Gamma

El empleo del algoritmo se justifica con base en lo presentado en [30] al caracterizarse la operación del clasificador y definirse el conjunto fundamental ideal con el cual se realiza una recuperación correcta.

Definición 3.4 *Sea el conjunto fundamental del clasificador Gamma el conjunto de patrones asociados a una clase, de la forma* $\{(x^{\mu}, y^{\mu}) \mid \mu = 1,2, \ldots, p\}$; *donde* x^{μ} *es un patrón,* y^{μ} *es su clase correspondiente. Además, para este conjunto fundamental se cumplen las siguientes tres afirmaciones:*

$$x^i \neq x^j \; \forall i, j \in \{1,2, \ldots, p\} \text{ tal que } i \neq j$$

Esto implica que no hay patrones repetidos.

$$x^i = x^j \Longrightarrow y^i = y^j \; \forall i, j \in \{1,2, \ldots, p\}$$

Un patrón dado no puede tener asociada más de una clase.

$$y^i = y^j \Longrightarrow x^i = x^j \; \forall i, j \in \{1,2, \ldots, p\}$$

Clases diferentes tienen asociados patrones diferentes.

Dicho de otra manera, el conjunto fundamental debe incluir una relación entre el conjunto de patrones y el conjunto de clases, de tal manera que dicha relación cumpla con las características de una función.

Algoritmo 3.2 *Sea el conjunto fundamental del clasificador Gamma de acuerdo con la definición 3.2.2. Al presentarse un patrón a clasificar* $\tilde{x}$, *donde* $\tilde{x}$ *es un vector real n-dimensional* $\tilde{x} \in \mathbb{R}^n$, *con* $n \in \mathbb{Z}^+$, *se realiza lo siguiente:*

1. Codificar las componentes de cada patrón del conjunto fundamental con el código Johnson-Möebius modificado. Se resta el menor valor a todos los componentes de cada patrón y se obtiene un elemento trasladado $\boldsymbol{e_m} = \bigvee_{i=1}^{p} \boldsymbol{x_j^i}$ por cada componente esto con la finalidad de trabajar en un rango de 0 a $\boldsymbol{e_m}$. Así, la componente $\boldsymbol{x_j^i}$ se transforma en un vector binario de dimensión $\boldsymbol{e_m(j)}$.

2. Codificar las componentes de cada patrón a clasificar con el código Johnson-Möebius modificado, utilizando las mismas condiciones del paso 1. En caso de que alguna componente del patrón a clasificar sea mayor al elemento $\boldsymbol{e_m}$correspondiente, esto es $\tilde{\boldsymbol{x}}_{\boldsymbol{\xi}} > \boldsymbol{e_m}(\boldsymbol{\xi})$, igualar esa componente a $\boldsymbol{e_m}(\boldsymbol{\xi})$ y guardar su valor anterior en la variable $\boldsymbol{mgamma}_{\boldsymbol{\xi}}$. Por otro lado, si alguna componente da un valor negativo una vez desplazada, igualar esa componente a 0 y asignar el valor $\boldsymbol{e_m}(\boldsymbol{\xi}) + |\tilde{\boldsymbol{x}}_{\boldsymbol{\xi}}|$ a $\boldsymbol{mgamma}_{\boldsymbol{\xi}}$.

3. Calcular el parámetro de paro ρ y el parámetro de pausa ρ0. Dependiendo del problema a tratar, algunas posibilidades sugeridas para estos parámetros son las siguientes:

   - $\boldsymbol{\rho} = \wedge_{\boldsymbol{j=1}}^{\boldsymbol{n}}\left(\vee_{\boldsymbol{i=1}}^{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\right)$
   - $\boldsymbol{\rho} = \frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}}\left(\vee_{\boldsymbol{i=1}}^{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\right)$
   - $\boldsymbol{\rho} = \vee_{\boldsymbol{j=1}}^{\boldsymbol{n}}\left(\vee_{\boldsymbol{i=1}}^{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\right)$
   - $\boldsymbol{\rho_0} = \wedge_{\boldsymbol{j=1}}^{\boldsymbol{n}}\left(\vee_{\boldsymbol{i=1}}^{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\right)$, sobre todo si $\boldsymbol{\rho} = \vee_{\boldsymbol{j=1}}^{\boldsymbol{n}}\left(\vee_{\boldsymbol{i=1}}^{\boldsymbol{p}} \boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\right)$
   - $\boldsymbol{\rho_0} = \boldsymbol{\rho}$, cuando se desea asignar forzosamente una clase conocida a los patrones desconocidos.

4. Determinar el umbral de pausa u. Considerando que el valor de este umbral depende fuertemente de las características del problema y las propiedades del conjunto fundamental, se ofrecen las siguientes sugerencias como valores iníciales:

   - $u = 0$.
   - $u = n$.

5. Determinar los pesos de cada dimensión $\boldsymbol{w_i} \boldsymbol{\epsilon} \mathbb{R}^+ | \boldsymbol{i} = \boldsymbol{1, 2, \ldots, n}$. Dentro de este contexto, se sugieren los siguientes rangos como valores iníciales empíricos:

   - Dentro del rango [1.5, 2] a las dimensiones que sean puntualmente separables para todas las clases.
   - Dentro del rango [1, 1.5] a las dimensiones que sean puntualmente separables para algunas clases o bien, que sean puntualmente segmentables para todas las clases.
   - Dentro del rango [0.8, 1.2] a las dimensiones que sean puntualmente segmentables para todas o algunas clases.
   - Dentro del rango (0, 0.5] a las dimensiones que sean puntualmente no separables.

6. Realizar una conversión de índices en los patrones del conjunto fundamental, de manera que el índice que tenía un patrón originalmente, por ejemplo x^{μ}, se convierta en dos índices: uno para la clase a la que pertenece (ejemplo clase *i*) y otro para el orden que le corresponde dentro del conjunto (ejemplo orden *k*). Bajo estas condiciones, la notación para el patrón x^{μ} será ahora $x^{i\omega}$, esto es el patrón x pertenece a la clase i en la posición ω. Lo anterior se realiza para todos los patrones del conjunto fundamental.

7. Inicializar θ a 0.

8. Realizar la operación $\boldsymbol{\gamma_g}\left(\boldsymbol{x_j^{\mu}}, \boldsymbol{\tilde{x}_j}, \boldsymbol{\theta}\right)$para las componentes de cada uno de los patrones del conjunto fundamental y del patrón a clasificar, considerándose $\boldsymbol{mgamma_{\xi}}$ como la dimensión del patrón binario $\boldsymbol{\tilde{x}_{\xi}}$ en caso necesario.

9. Calcular la suma ponderada inicial $\boldsymbol{c_i^0}$ de los resultados obtenidos en el paso anterior, para cada patrón fundamental $\boldsymbol{\mu = 1, 2, \ldots, p}$

$$\boldsymbol{c_{\mu}^0 = \sum_{j=1}^{n} w_j \cdot \gamma_g\left(x_j^{\mu}, \tilde{x}_j, \theta\right)}$$

10. Evaluar si existe un máximo único, cuyo valor es además igual a *n*, asignando al patrón a clasificar la clase correspondiente a ese máximo: $\boldsymbol{\tilde{y} = y^i}$ tal que $\boldsymbol{\bigvee_{\mu=1}^{p} c_{\mu}^0 = c_i^0 = n}$. En otro caso, continuar.

11. Realizar la operación $\boldsymbol{\gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right)}$ para cada clase y para cada componente de cada uno de los patrones del conjunto fundamental que corresponden a esa clase, y del patrón a clasificar, considerándose $\boldsymbol{mgamma_{\xi}}$ como la dimensión del patrón binario $\boldsymbol{\tilde{x}_{\xi}}$ si es necesario.

12. Calcular la suma ponderada $\boldsymbol{c_i}$ de los resultados obtenidos en el paso 11, para cada clase $\boldsymbol{i = 1, 2, \ldots, m}$

$$\boldsymbol{c_i = \frac{\sum_{\omega=1}^{k_i} \sum_{j=1}^{n} w_j \cdot \gamma_g\left(x_j^{i\omega}, \tilde{x}_j, \theta\right)}{k_i}}$$

13. Evaluar, si existe más de un máximo entre las sumas ponderadas por clase, incrementar θ en 1 y repetir los pasos 11 y 12, hasta que:

    a. exista un máximo único;
    b. o se cumpla con la condición de pausa: $\theta = \rho 0$;
    c. o se cumpla con la condición de pausa: $\theta \geq \rho$.

14. Evaluar, si se cumple con la condición de pausa $\theta = \rho 0$, se compara el valor máximo de las sumas ponderadas con el umbral de pausa.

    a. Si $\bigvee_{i=1}^{m} \boldsymbol{c_i} \leq \boldsymbol{u}$ entonces se asigna la clase desconocida al patrón a clasificar: $\boldsymbol{C_{\tilde{x}} = C_0}$

    b. Si $\bigvee_{i=1}^{m} \boldsymbol{c_i} > \boldsymbol{u}$ entonces se continúa en el paso 11.

15. Evaluar, si existe un máximo único, asignar al patrón a clasificar la clase correspondiente a ese máximo: $\tilde{\boldsymbol{y}} = \boldsymbol{y^j}$ tal que $\bigvee_{i=1}^{m} \boldsymbol{c_i} = \boldsymbol{c_j}$.

16. En caso contrario: si $\boldsymbol{\lambda}$ es el índice más pequeño de clase que corresponde a uno de los máximos, asignar al patrón a clasificar la clase $\tilde{\boldsymbol{y}} = \boldsymbol{y^{\lambda}}$ tal que $\boldsymbol{C_{\tilde{x}} = c_{\lambda}}$.

A continuación se presentan dos diagramas de bloques. El primero describe el pre-procesamiento (pasos 1 a 7) que debe realizarse para proseguir con el segundo diagrama que describe el algoritmo del clasificador Gamma

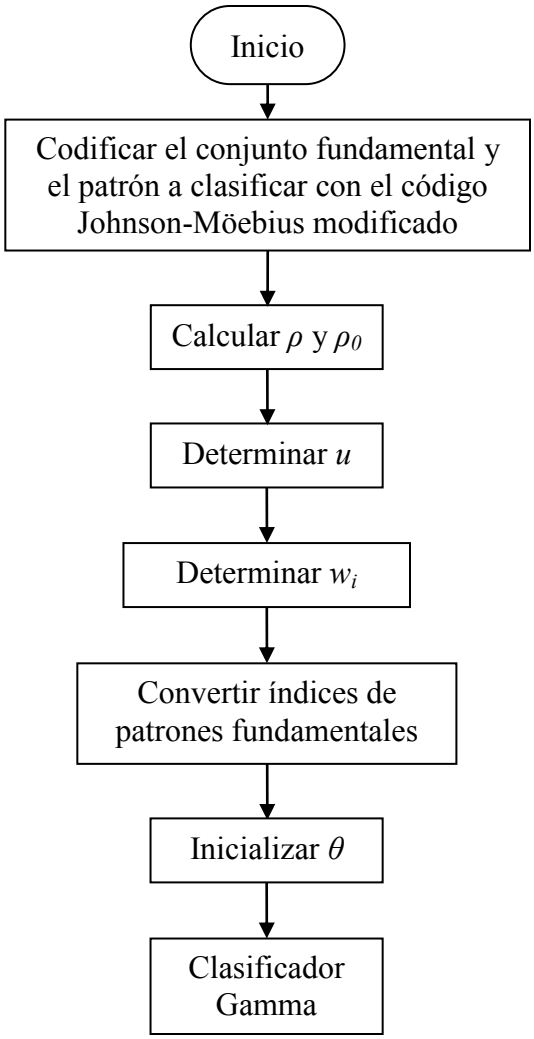


Figura 3.1 *Diagrama de bloques del algoritmo del clasificador Gamma, primera parte.*

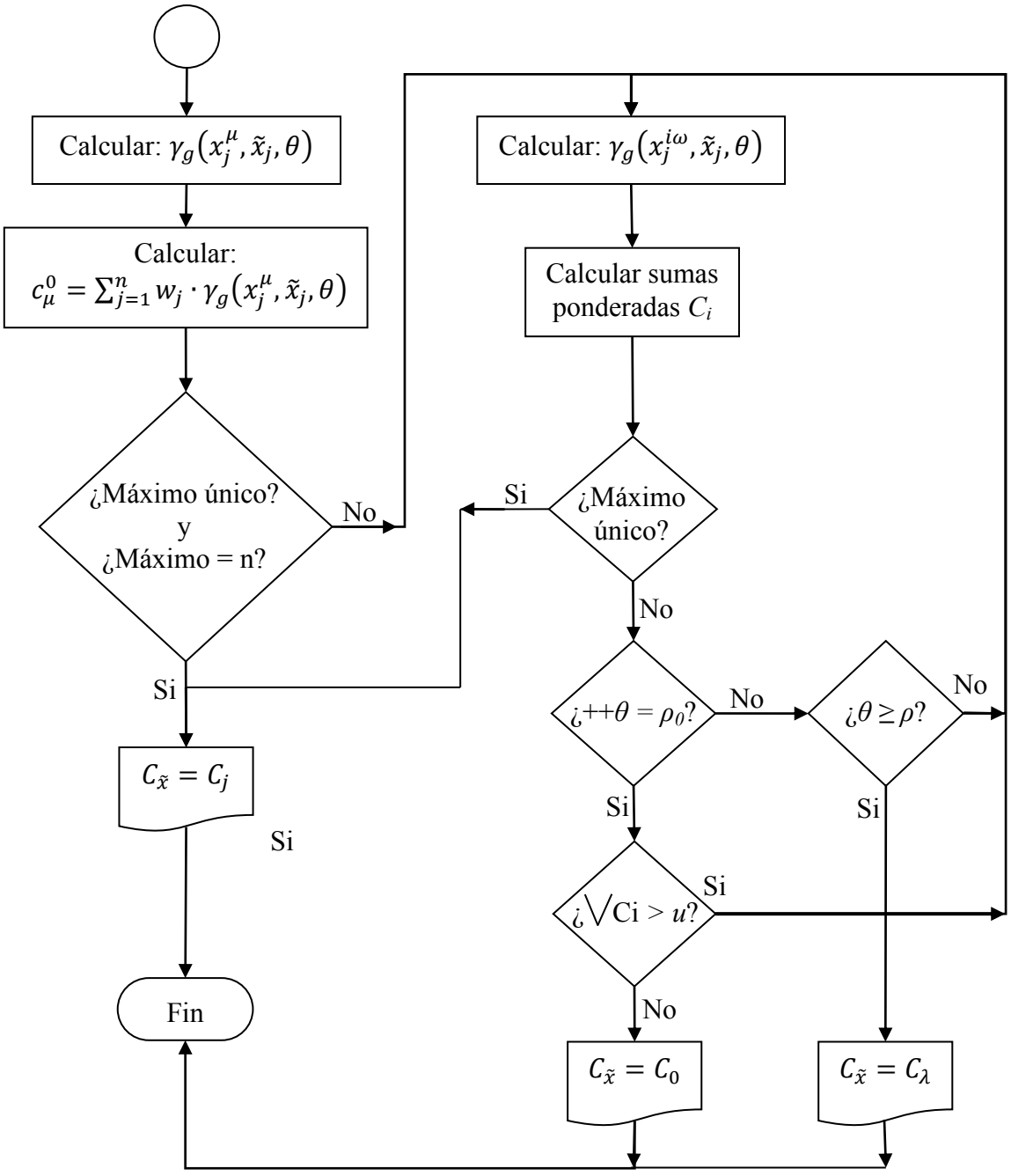


Figura 3.2 *Diagrama de bloques del algoritmo del clasificador Gamma, segunda parte.*

3.6 Métodos de validación cruzada

Los métodos de validación cruzada se aplican para conocer una estimación confiable del comportamiento de algún modelo propuesto en presencia de instancias no conocidas, es decir, se pretende saber si el subconjunto de rasgos seleccionados es el adecuado y si la precisión predictiva sobre instancias desconocidas es aceptable.

A continuación citaremos el método de validación cruzada que se empleará al modelo propuesto.

3.6.1 K-fold Cross Validation

En este método el conjunto de datos disponibles es dividido en *K* particiones que dan lugar a *K* subconjuntos mutuamente excluyentes.

Para cada una de las *K* estimaciones de error, uno de los *K* subconjuntos es usado como conjunto de prueba y los otros *K-1* restantes son agrupados para formar el conjunto de entrenamiento.

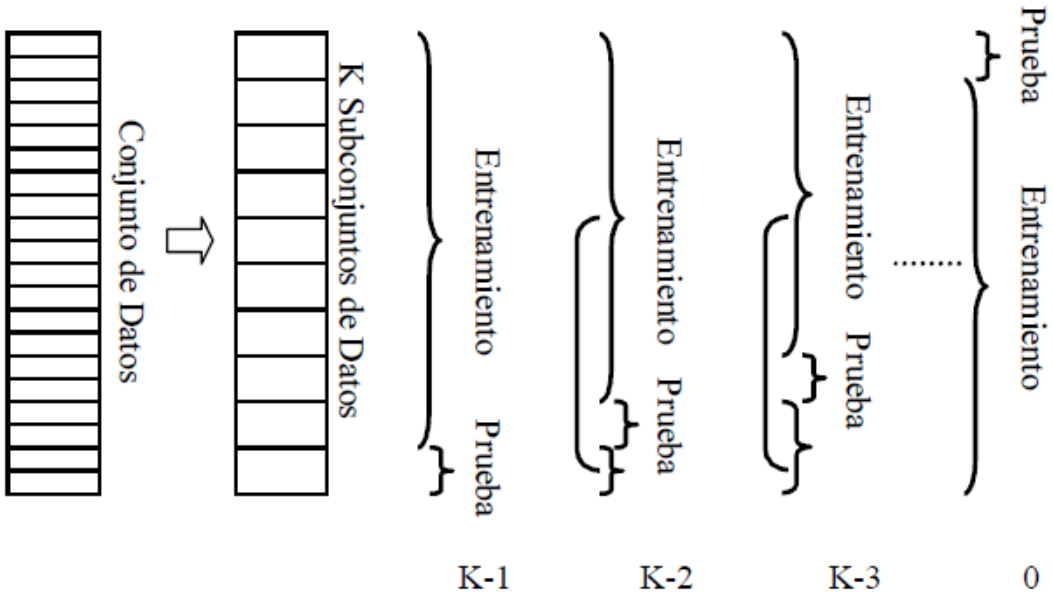


Figura 3.3 *Estimación del error usando K-fold cross validation.*

El procedimiento será repetido *K* veces para asegurar que los *K* subconjuntos hayan empleados en la fase de prueba, de modo tal, que se obtendrán *K* estimaciones de error E_i, que serán promediadas para obtener el error total de predicción E del modelo en cuestión.

$$\boldsymbol{E = \frac{1}{K} \sum_{i=1}^{K} E_i}$$

La estimación total del error E, es de mayor confiabilidad, debido a que todos los patrones son considerados en la fase de prueba, aun cuando se disponga de pocas instancias para ambas fases (entrenamiento/prueba).

Otra forma de expresarlo es que todos los patrones se consideran una vez en la fase de prueba y *K-1* veces en la fase de entrenamiento. Es importante mencionar que la varianza en la estimación del error de predicción disminuye conforme el valor de *K* aumenta.

Una desventaja evidente de este método es que la fase de entrenamiento tiene que ejecutarse en *K* ocasiones, lo que implica *K* veces más tiempo de computo con respecto a otras técnicas.

3.7 Análisis de bancos de datos seleccionado

Los bancos de datos seleccionados para los experimentos presentados en el capítulo 5 son 3, los primeros 2 son archivos extraídos del repositorio del UCI *machine learning* y el tercero de ellos es una serie de tiempo, que ha sido generada por el SIMAT (Sistema de Monitoreo Atmosférico de la ciudad de México) a través del subsistema RAMA (Red Automática de Monitoreo Atmosférico).

El primer banco de datos se llama "*Iris plant*", es quizás el más conocido que se encuentran en la literatura de reconocimiento de patrones. El conjunto de datos contiene 3 clases de 50 casos cada uno, donde cada clase se refiere a un tipo de planta iris. Una clase es linealmente separable de las otras 2, estos últimos no son linealmente separables una de otra.

El segundo banco de datos se llama "*Balance scale*", se generó para modelar unos resultados experimentales psicológicos. Cada clasificación es como la punta de una balanza, se inclina hacia la derecha, se inclina hacia la izquierda, o esta es equilibrada. Por lo que el conjunto de datos contiene 3 clases de 288 casos las primeras 2 clases y la tercera contiene 49 casos.

El tercer banco de datos es una serie de tiempo llamada "*12RAMA*" son series de tiempo, que han sido generadas por el SIMAT, a través del subsistema RAMA y son datos de las emisiones de contaminantes capturados en el año 2012, estos datos deberán ser codificados en forma de vectores que puedan ser utilizados por el clasificador Gamma.

En la sección 5.1 se brindará una descripción más detallada de cada banco de datos que utilizado para los experimentos de este trabajo de tesis.

Para evitar problemas potenciales con respecto a la no estacionalidad en la series de tiempo con referencia a los datos capturados, se trabaja con las diferencias entre las muestras consecutivas [54], [55].

Las diferencias son calculadas después de aplicar los pasos 3 y 4 del código Johnson-Möebius modificado, descrito en la sección 3.4. La tabla 3.4 muestra un ejemplo de este procedimiento.

Tabla 3.4: Ejemplo del cálculo de diferencias.

	Datos originales	Datos Escalados	Datos Truncados	Diferencias
1	2,1	21	21	
2	1,9	19	19	-2
3	1,0	10	10	-9
4	0,4	4	4	-6

Los datos tomados de una serie de tiempo serán codificados en k patrones de dimensión n representados de la siguiente forma:

$$x^k = \begin{pmatrix} x_1^k \\ x_2^k \\ x_3^k \\ \cdot \\ \cdot \\ \cdot \\ x_n^k \end{pmatrix}$$

donde

x^k es el k-ésimo patrón

x_n^k es el n-ésimo valor del patrón x^k

3.8 Conjunto de entrenamiento

El contenido de esta sección se encuentra basado fuertemente en el trabajo [60].
El conjunto de entrenamiento estará formado por patrones de dimensión n, que contendrá los valores de una serie de tiempo correspondiente a un año de monitoreo del contaminante CO (Monóxido de carbono) en una delegación de la ciudad de México, en la cual se tomaron p muestras de emisiones. Si se desea codificar una serie de tiempo en vectores de dimensión n = 5, el primer patrón contendrá los valores de los índices 0 al 4, el segundo patrón los valores de los índices 1 al 5 y así sucesivamente hasta llegar al último patrón que contendrá los valores de los índices $x_p - 4$ al x_p. La tabla 3.5 muestra la estructura de una serie de datos, mientras que la tabla 3.6 muestra la forma en que se organizan los patrones correspondientes a la serie de datos de la tabla 3.5.

Tabla 3.5: Serie de datos del conjunto de entrenamiento.

Índice	Valor
0	x_0
1	x_1
2	x_2
⋮	⋮
p-2	$x_p - 2$
p-1	$x_p - 1$
p	x_p

Tabla 3.6: Codificación de patrones de entrenamiento.

$$x^0 = \begin{pmatrix} x_0^0 \\ x_1^0 \\ x_2^0 \\ x_3^0 \\ x_4^0 \end{pmatrix} \quad x^1 = \begin{pmatrix} x_0^1 \\ x_1^1 \\ x_2^1 \\ x_3^1 \\ x_4^1 \end{pmatrix} \quad x^2 = \begin{pmatrix} x_0^2 \\ x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{pmatrix} \quad x^k = \begin{pmatrix} x_{p-4}^k \\ x_{p-3}^k \\ x_{p-2}^k \\ x_{p-1}^k \\ x_p^k \end{pmatrix}$$

Al codificar una serie de datos de la forma antes mencionada, el número de patrones para el conjunto de entrenamiento, está dado por:

$$\text{Total de Patrones} = p - n$$

Dónde:

p es el número de datos de la serie de tiempo.
n es la dimensión de los patrones.

3.9 Conjunto de prueba

El contenido de esta sección se encuentra basado fuertemente en el trabajo [60].
Los patrones en el conjunto de prueba tendrán la misma dimensión n que los patrones del conjunto de entrenamiento y contendrán datos del mismo contaminante que se utilizó en el conjunto de entrenamiento, pero de un mes específico de un año diferente a la utilizada durante la fase de entrenamiento. La codificación del conjunto de prueba será igual que la del conjunto fundamental pero los patrones se denotaran por $\tilde{x}$. La tabla 3.7 muestra la estructura de una serie de datos, mientras que la tabla 3.8 muestra la forma en que se organizan los patrones correspondientes a la serie de datos de la tabla 3.7.

Tabla 3.7: Serie de datos del conjunto de prueba.

Índice	Valor
0	$\tilde{x}_0$
1	$\tilde{x}_1$
2	$\tilde{x}_2$
⋮	⋮
p-2	$\tilde{x}_p - 2$
p-1	$\tilde{x}_p - 1$
P	$\tilde{x}_p$

Tabla 3.8: Codificación de patrones de prueba.

$$\tilde{x}^0 = \begin{pmatrix} \tilde{x}_0^0 \\ \tilde{x}_1^0 \\ \tilde{x}_2^0 \\ \tilde{x}_3^0 \\ \tilde{x}_4^0 \end{pmatrix} \quad \tilde{x}^1 = \begin{pmatrix} \tilde{x}_0^1 \\ \tilde{x}_1^1 \\ \tilde{x}_2^1 \\ \tilde{x}_3^1 \\ \tilde{x}_4^1 \end{pmatrix} \quad \tilde{x}^2 = \begin{pmatrix} \tilde{x}_0^2 \\ \tilde{x}_1^2 \\ \tilde{x}_2^2 \\ \tilde{x}_3^2 \\ \tilde{x}_4^2 \end{pmatrix} \quad \tilde{x}^k = \begin{pmatrix} \tilde{x}_{p-4}^k \\ \tilde{x}_{p-3}^k \\ \tilde{x}_{p-2}^k \\ \tilde{x}_{p-1}^k \\ \tilde{x}_p^k \end{pmatrix}$$

3.10 Clases

El contenido de esta sección se encuentra basado fuertemente en el trabajo [60]. Para la aplicación del clasificador Gamma a la tarea de predicción, las clases estarán codificadas como un vector unidimensional. Consideremos el i-ésimo patrón x^i, que se forma de una serie de datos; el patrón estará formado desde el valor i hasta el valor $i + (n - 1)$ de la serie de datos.

La clase correspondiente a x^i, denotada por C_i, será el valor que sigue al último elemento de x^i, es decir que C_i es igual al valor $i + n$. La figura 3.3 muestra un ejemplo de la forma en que son asignadas las clases.

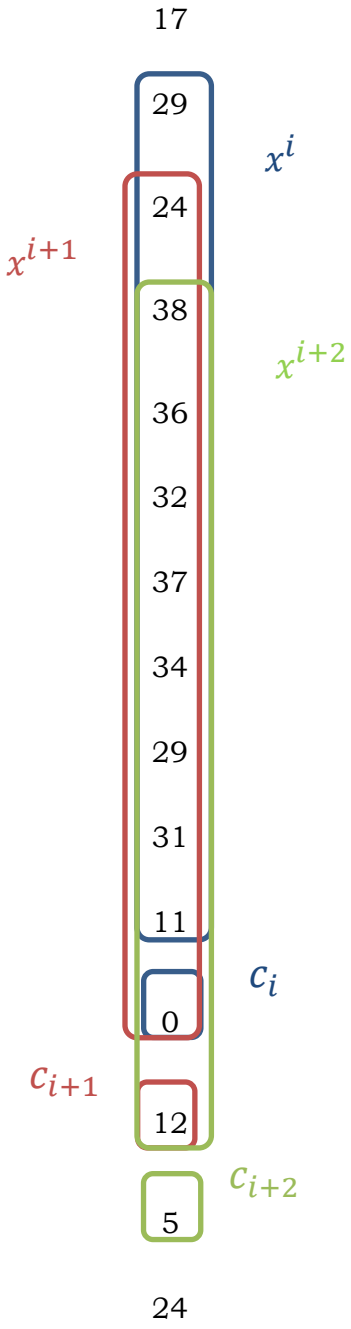


Figura 3.4 *Ejemplo de asignación de clases en series de tiempo.*

3.11 Dispositivo FPGA

Los FPGA son dispositivos semiconductores que están construidos por bloques lógicos que se interconectan. Para definir la funcionalidad del dispositivo, este debe ser configurado mediante un lenguaje de descripción de *hardware*.

Este tipo de dispositivos suelen ser programados con diferentes finalidades, dependiendo de las necesidades del diseñador y los requisitos de cada sistema que se quiera implementar en el FPGA, ya que una de sus características principales es el emplear circuitos del tipo ASIC (*Application Specific Integrated Circuits*) al diseño elaborado, con el fin de que el tiempo de implementación de los bloques lógicos tienda a ser menor.

Para la programación de estos dispositivos una de las herramientas más empleadas es VHDL, además de las herramientas del tipo EDA (*Electronics Design Automation*) que permiten al usuario el desarrollo del sistema digital usando símbolos esquemáticos.

3.11.1 Tarjeta de desarrollo Digilent Genesys™

Esta tarjeta de desarrollo, presentada en la figura 3.5, cuenta con el *FPGA* de *Xilinx Virtex-5 LX50T* de la serie *Virtex* el cual está optimizado para interactuar con todos los dispositivos de entrada-salida de la placa.

Las principales características de esta tarjeta de desarrollo son las siguientes:

- FPGA *Xilinx Virtex -5 LX50T* encapsulado de 1136-pin unido a la tarjeta por BGA (*Ball Grid Array*).
- 256*MByte* de memoria *DDR2 SODIMM* que tiene 64-bits de ancho de datos.
- Puerto 10/100/1000 Ethernet PHY y RS-232 puerto serie
- Múltiples puertos *USB2* para la programación, transferencia de datos y *hosting*.
- Puerto *HDMI* con una resolución de hasta 1600x1200 y color de 24 bits
- Códec AC-97 de muestreo 48 kHz para la línea de entrada y salida de micrófono y auriculares.
- Monitoreo en tiempo real del consumo de energía de las vías de alimentación.
- 100MHz en un oscilador interno y (hasta) 400 MHz en un generador de reloj programable.

- 112 puertos de E/S dirigidos a los conectores de expansión (dos conectores VHDC paralelo de alta velocidad y cuatro cabeceras de 8 pines).
- 8 LEDs, 2 botones, interruptores de navegación de dos ejes, 8 interruptores y un LCD de caracteres de 16x2.
- Su alimentación es con una fuente de alimentación de 20W, un cable USB y un protector de plexiglás.

Figura 3.5 *Tarjeta de desarrollo Digilent Genesys™ Virtex 5*

Capítulo 4

Modelo Propuesto

En este capítulo se establece el modelo y diseño de la arquitectura de cómputo de parámetros fijos, que tendrá la tarea de la clasificación de patrones, como siguiente punto se muestra la representación esquemática a nivel de compuertas lógicas del bloque principal que permite la implementación en hardware del modelo asociativo Gamma.

4.1 Modelo de la arquitectura propuesta

La idea de esta arquitectura es la implementación del algoritmo de clasificación Gamma, ya que el pre-procesamiento de la información se realizará en la computadora.

Por medio de una comunicación serie a 19200 *baudios* por segundo, serán enviados los datos del conjunto fundamental junto con el conjunto de prueba. Siendo esta comunicación usada para que el FPGA tenga a su disposición toda la información para realizar la tarea de clasificación.

Los datos enviados serán el flujo de datos para que la arquitectura realice las operaciones.

Finalmente el resultado se retornará por la comunicación serie hacia la computadora, donde esta última desplegará el resultado de la clasificación del conjunto de prueba.

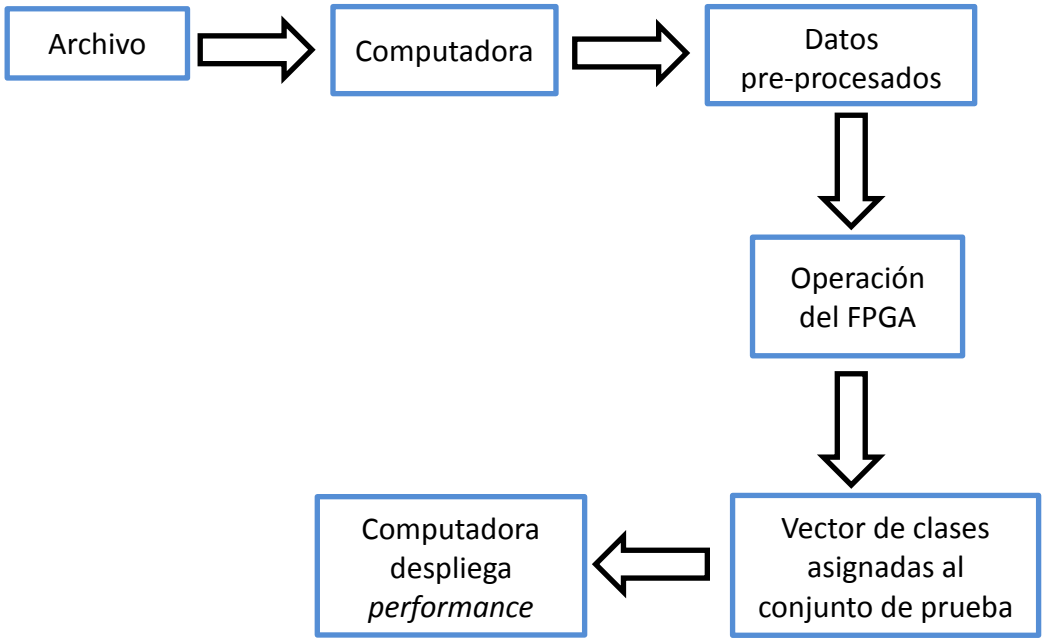


Figura 4.1 *Representación del modelo de la arquitectura propuesta.*

El parametrizado de la arquitectura dependerá totalmente del tamaño de conjunto de aprendizaje y el conjunto de prueba, ya que el arreglo que recibirá todos esos datos será de tamaño variable.

De esta arquitectura el objetivo específico será regresar un vector que contenga, la clase asignada a cada caso del conjunto de prueba, para que la computadora posteriormente entregue el *performance* de la evaluación realizada.

4.2 Diseño de la arquitectura propuesta

Para el diseño de esta arquitectura se deberá cumplir con el objetivo principal, el hacer fluir la información a través de la arquitectura, dicha información será entregada por la computadora para que posteriormente el FPGA realice las operaciones correspondientes a la fase de recuperación y envié la información de regreso a la computadora para completar el proceso.

El diseño de esta arquitectura se dividirá en:

- *Software*
- *Hardware*

Cada una de las divisiones requiere especificaciones para un correcto funcionamiento, tales como:

Especificaciones *Software*

- Características de los bancos de datos. (Tamaño, clases, rasgos, tipo de dato, tamaño de dato)

- Tasa de transferencia de datos. (Adecuada para una comunicación asíncrona entre el FPGA y la computadora)

- Tipo de archivo que incluye el banco de datos.

Especificaciones *Hardware*

- Características de los datos recibidos conjunto fundamental y conjunto de prueba. (Tamaño de datos)

- Tasa de transferencia de datos. (Adecuada para una comunicación asíncrona entre el FPGA y la computadora)

De tal forma que el diseño pretende realizar una interacción entre la computadora que tendrá la tarea de realizar el pre-procesamiento de los datos que posteriormente será entregados al FPGA, este último tendrá la tarea de realizar las operaciones del modelo asociativo Gamma, para que finalmente envié de regreso a la computadora un vector que contendrá las clases asignadas para el conjunto de prueba ingresado, la última tarea de la computadora será mostrar el *performance* del clasificador, para posteriormente poderlo comparar con otros algoritmos de clasificación.

El diseño propuesto se muestra en la *figura 4.2* en forma de un diagrama de bloques que se dilucidará más adelante.

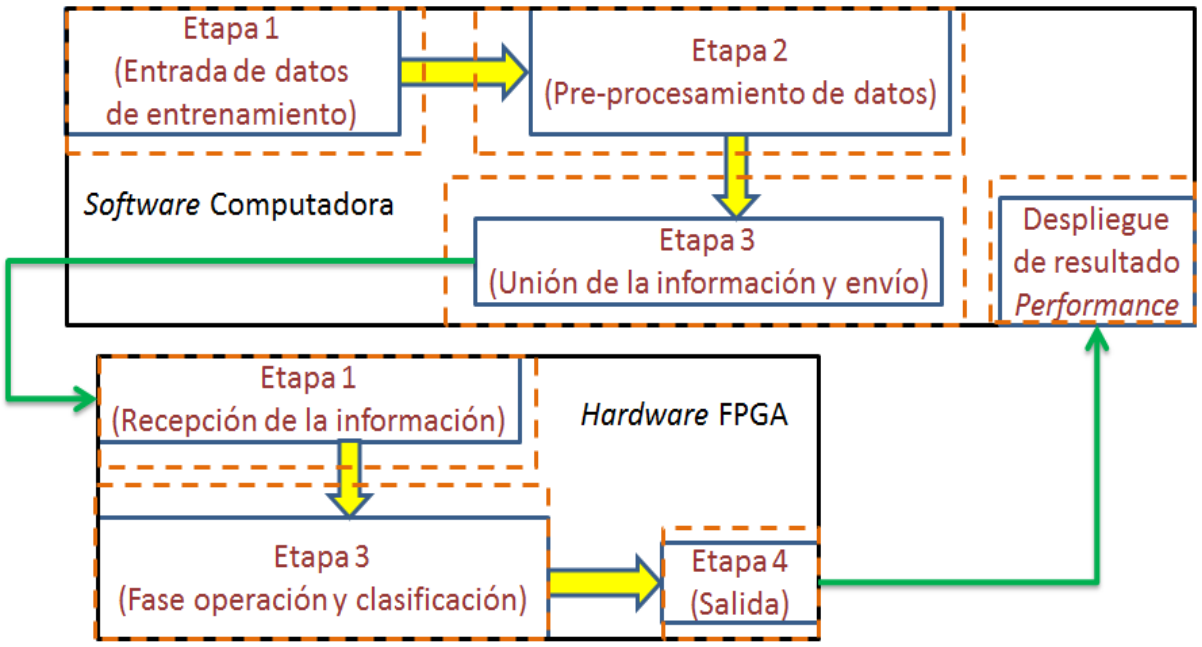


Figura 4.2 *Diagrama de bloques de la arquitectura propuesta.*

4.2.1 Acciones para el funcionamiento de la arquitectura

Este apartado se en listan las acciones necesarias para que la arquitectura antes propuesta funcione correctamente, así mismo se colocaran etapas para una mejor comprensión del funcionamiento.

1. Etapa 1 (Entrada de datos de entrenamiento) "*Software*-Computadora"

   - Se realizará la extracción de los datos provenientes del archivo que contiene el banco de datos.

2. Etapa 2 (Pre-procesamiento de datos) "*Software*-Computadora"

   - Se realiza un escalamiento del banco de datos original para trabajar con números enteros.
   - Se desordena el banco de datos.
   - Después se divide el banco de datos en conjunto fundamental y conjunto de prueba.

3. Etapa 3 (Unión de la información y envío) "*Software*-Computadora"

   - Los conjuntos son unidos en un arreglo.
   - El arreglo generado es el que se envía al FPGA por medio de la comunicación entre el dispositivo y la computadora.

4. Etapa 1 (Recepción de la información) "*Hardware*-FPGA"

   - Se recibirán los paquetes de información provenientes de la computadora por medio de la comunicación, dichos paquetes contendrán el conjunto de entrenamiento y el conjunto de prueba.
   - Dichos paquetes de información serán colocados en un arreglo.

5. Etapa 2 (Fase operación y clasificación) "*Hardware*-FPGA"

   - Este módulo será una caja negra, que tendrá como entrada los datos suficientes para realizar las operaciones del algoritmo Gamma.
   - La salida de este módulo será un vector que contendrá las clases designadas por el algoritmo.

6. Etapa 3 (Salida) "*Hardware*-FPGA"

   - En esta etapa se tomará la salida de la etapa anterior y será enviada a través de la comunicación a la computadora.

7. Despliegue de resultado *Performance*

   - Aquí la computadora será la encargada de calcular el rendimiento del clasificador y desplegarlo en la pantalla.

4.3 Algoritmo implementado

Este es el algoritmo que fue implementado en el FPGA, con el fin de reducir el número de compuertas empleadas y de esta forma tener mayores recursos para almacenar una mayor cantidad de información.

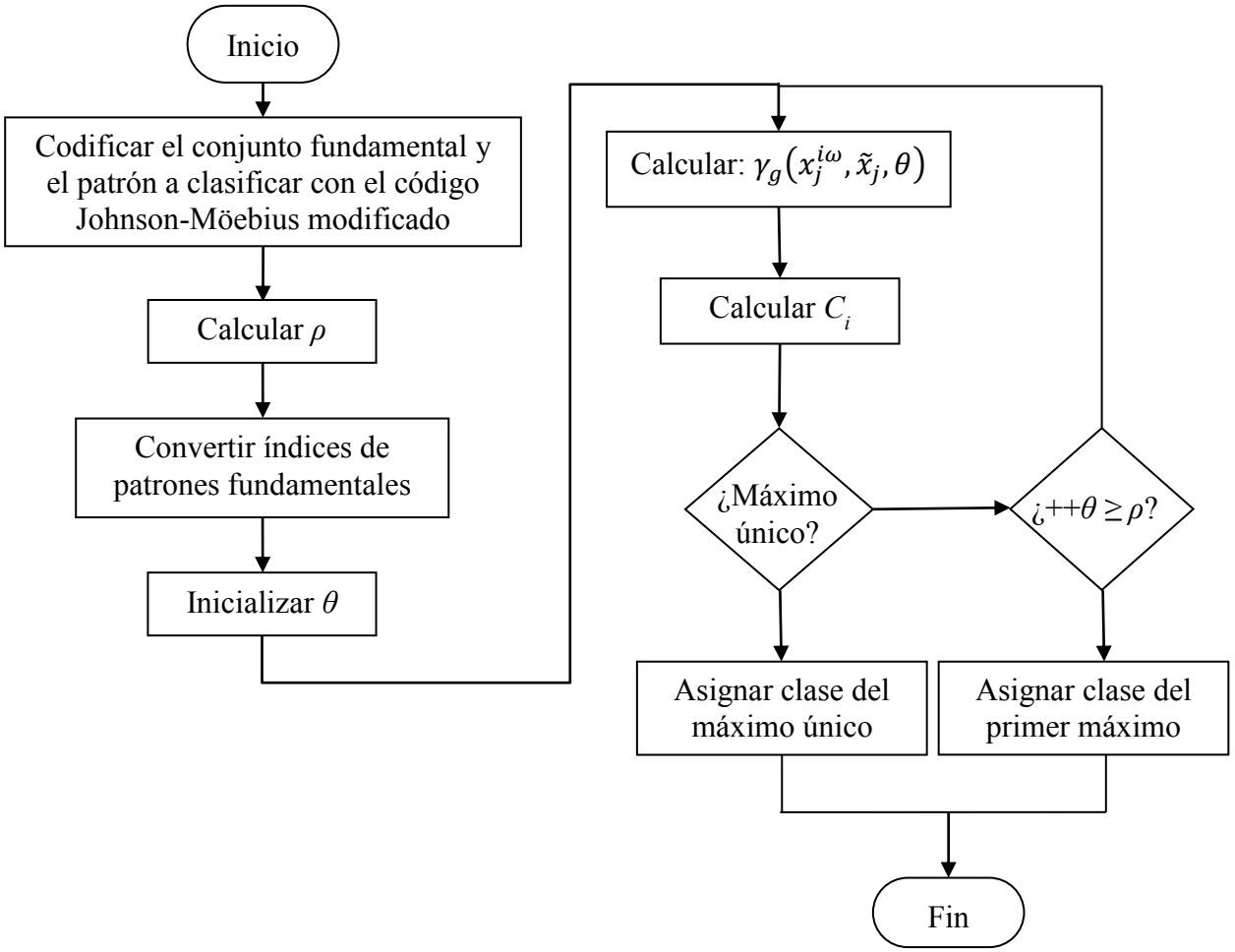


Figura 4.3 *Diagrama de bloques del algoritmo implementado en el* FPGA.

4.4 Módulo operación y clasificación

Este es el módulo donde el operador gamma (γ) de similitud, realizan las operaciones previas a la clasificación.

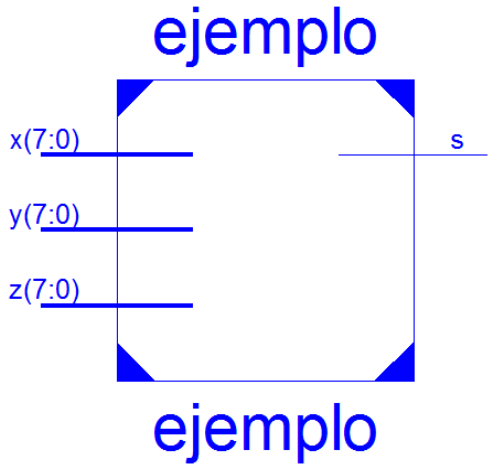


Figura 4.4 *Módulo del operador gamma (γ) de similitud.*

Esta es la vista interna del módulo donde se procesan todos los patrones con el fin de encontrar el resultado correcto para la clasificación de cada patrón.

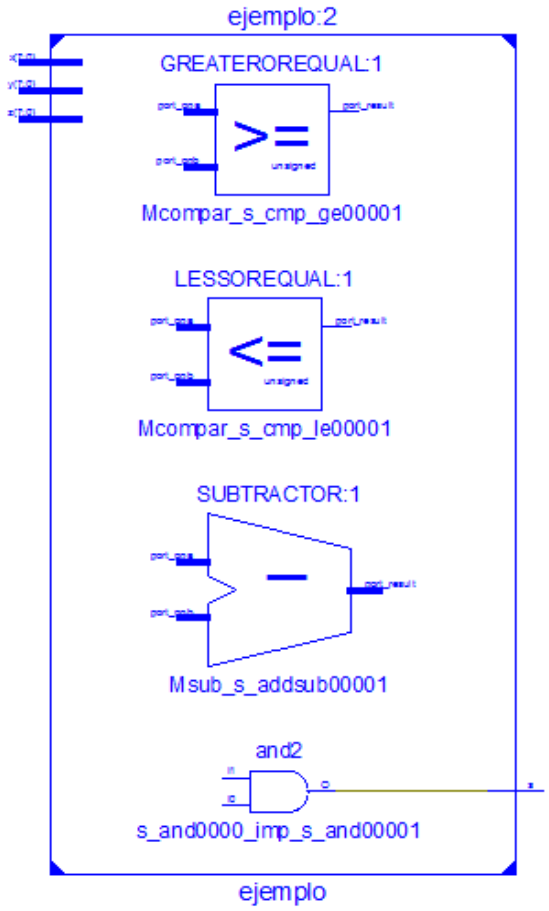


Figura 4.5 *Implementación del operador gamma (γ) de similitud.*

4.5 Arquitectura Propuesta

Esta es una imagen de la arquitectura generada e implementada en el FPGA.

Esta arquitectura hace uso de diferentes contadores, multiplexores, banderas auxiliares, cargadores de registro, sumadores, el módulo del operador gamma (γ) de similitud, una unidad de control y registros generales.

La unión de los elementos anteriores permite que la arquitectura funcione y entregue un resultado de clasificación.

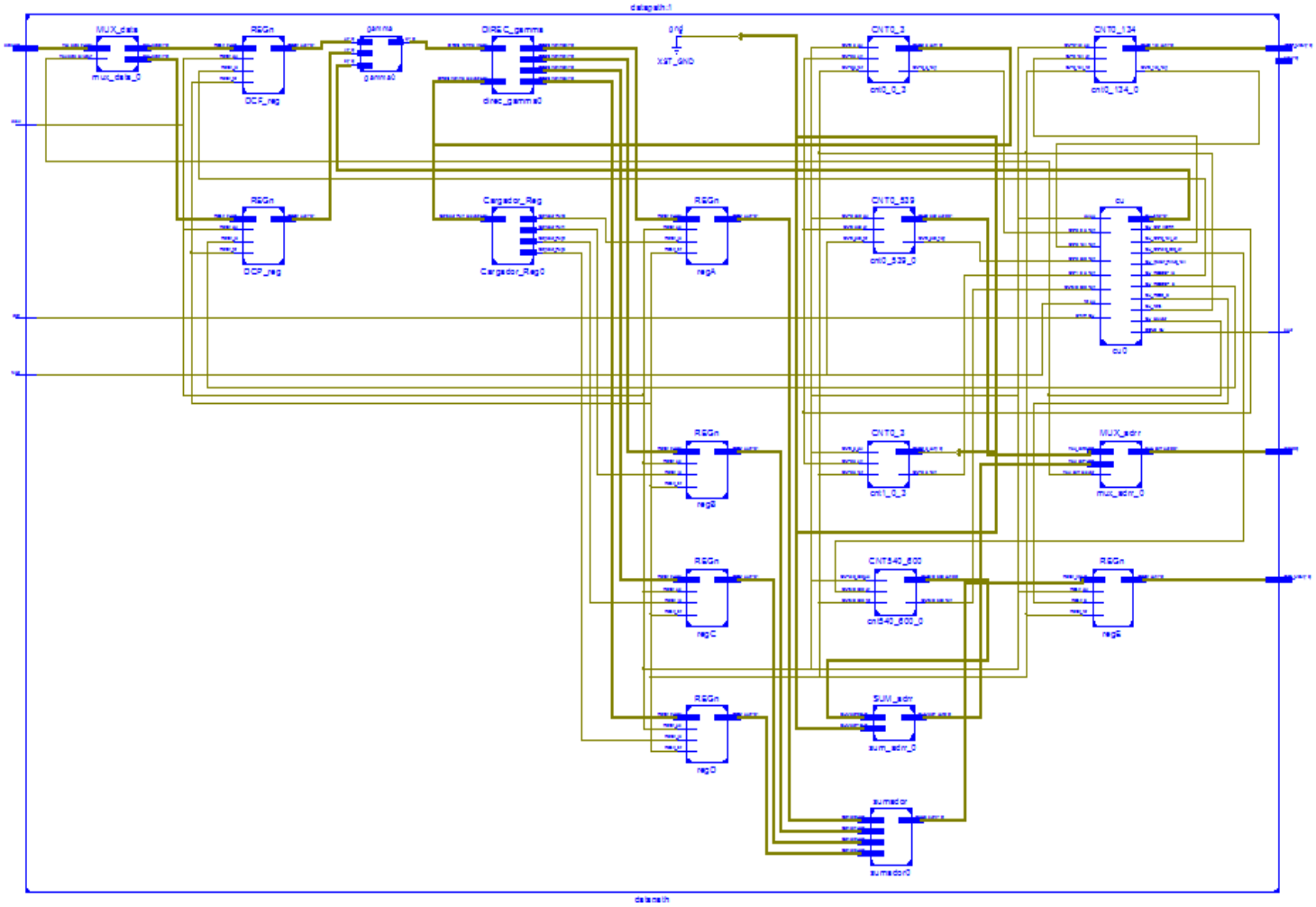


Figura 4.6 *Implementación de la arquitectura propuesta.*

Capítulo 5

Resultados y Discusión

En este capítulo se presentan los resultados correspondientes a las pruebas realizadas con el clasificador Gamma, así mismo, se muestran los resultados que corresponden al empleo de otros algoritmos de reconocimiento de patrones con los cuales se estableció una comparación de rendimiento.

5.1 Desempeño de la Implementación en *Hardware*

Aquí se presenta una prueba, en la cual se demuestra el desempeño del módulo del operador gamma (γ) de similitud, ya que este módulo es el encargado de entregar los resultados que posteriormente un sumador procesará para determinar la clase, resulta de gran importancia que este módulo brinde la información correcta acorde a su fundamentación matemática antes expuesta.

Recordemos la definición 3.3 que nos brinda esta representación:

$$\gamma(x, y, \theta) = \begin{cases} 1 \; si \; n - u_\beta[\alpha(\mathrm{x},\mathrm{y}) \bmod 2] \leq \theta \\ 0 \; en \; otro \; caso \end{cases}$$

Prueba 5.1: Ahora tomaremos un patrón (*x*, *y*) de las siguientes características donde *n*=*m*, como lo indica el algoritmo $\theta = 0$, se incrementara $\theta + 1$ hasta alcanzar *ρ=4* si fuera el caso.

$$x = \begin{pmatrix} 43 \\ 50 \\ 35 \\ 22 \\ 32 \end{pmatrix} \qquad y = \begin{pmatrix} 41 \\ 50 \\ 38 \\ 20 \\ 37 \end{pmatrix}$$

A continuación se muestran la simulación de la operación rasgo a rasgo.

Dónde:

x es un bus de 7 hasta 0 bits
y es un bus de 7 hasta 0 bits
z es un bus de 7 hasta 0 bits representa θ
s es una salida de un bit

En esta figura podemos observar que θ=2, por lo que se obtuvo a la salida un $\gamma(x, y, \theta)=1$.

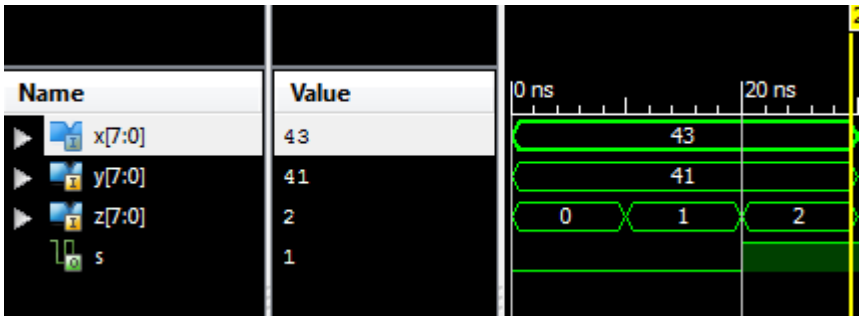


Figura 5.1 *Simulación del primer rasgo de la prueba 5.1*

En esta figura podemos observar que θ=0, por lo que se obtuvo a la salida un $\gamma(x, y, \theta)=1$

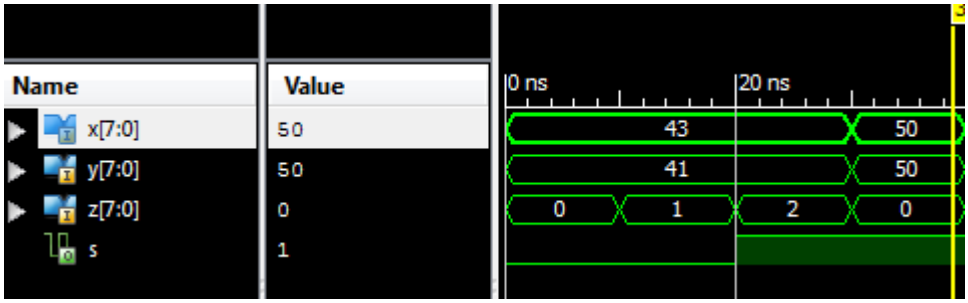


Figura 5.2 *Simulación del segundo rasgo de la prueba 5.1*

En esta figura podemos observar que θ=3, por lo que se obtuvo a la salida un $\gamma(x, y, \theta)=1$.

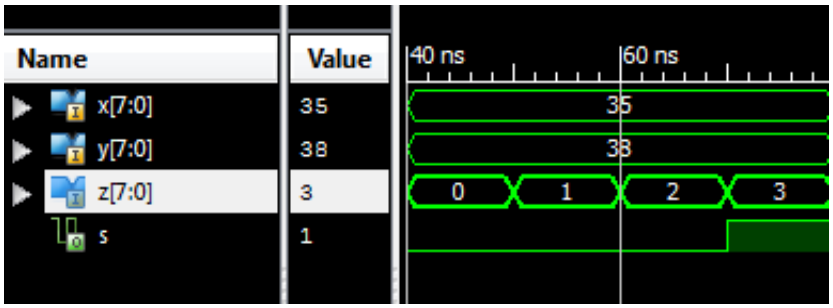


Figura 5.3 *Simulación del tercer rasgo de la prueba 5.1*

En esta figura podemos observar que θ=2, por lo que se obtuvo a la salida un $\gamma(x, y, \theta)=1$.

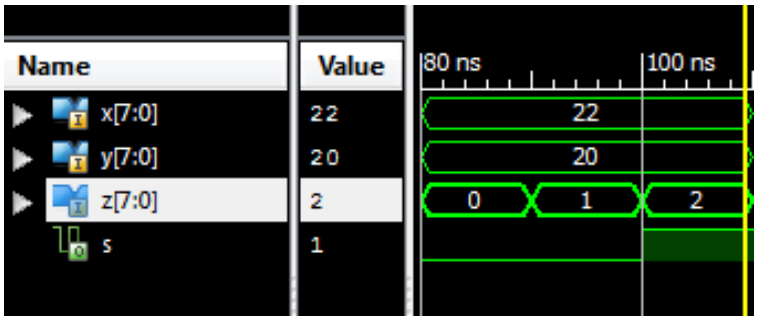


Figura 5.4 *Simulación del cuarto rasgo de la prueba 5.1*

En esta figura podemos observar que θ=4, por lo que alcanzo la condición ρ=4 por lo tanto en la salida un $\gamma(x, y, \theta)=0$.

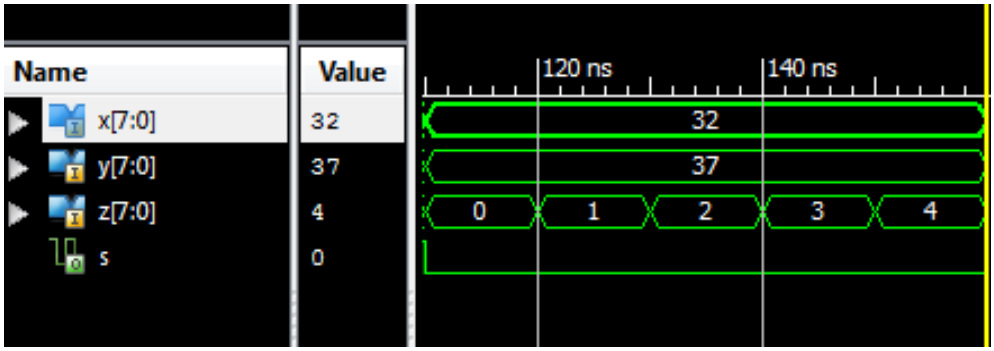


Figura 5.5 *Simulación del quinto rasgo de la prueba 5.1*

Después de este ejemplo es posible aseverar que las condiciones necesarias para el operador gamma (γ) de similitud se cumplen en la implementación

por lo que ahora se presentaran los resultados obtenidos por el clasificador en comparación con otros clasificadores.

5.2 Banco de datos

Los bancos de datos utilizados para las pruebas realizadas en esta sección fueron obtenidos de http://archive.ics.uci.edu/ml/machine-learning-databases/iris/, de http://archive.ics.uci.edu/ml/machine-learning-databases/balance-scale/ y http://www.calidadaire.df.gob.mx/calidadaire/productos/basesdedatos/bd_rama.php.

Estos bancos no tienen relación ya que una de las ideas es poner a prueba el clasificador con diferentes datos y áreas de aplicación.
Es por eso que se utilizaron 2 bancos de datos para clasificar y el tercero para la tarea de predicción.
Como se comentó en la sección 3.7 el primer banco de datos es quizás el más conocido que se encuentran en la literatura de reconocimiento de patrones; el segundo banco de datos se generó para modelar unos resultados experimentales psicológicos. Y El tercer banco de datos es una serie de tiempo llamada "*12RAMA*" y de las series de tiempo que contiene se empleó el CO (Monóxido de carbono) que han sido generadas por el SIMAT, a través del subsistema RAMA, desde la estación Merced, ubicada en la delegación Venustiano Carranza, perteneciente al Distrito Federal.

5.2.1 Clasificación de "*Iris plant*"

En primera instancia se presenta la base de datos *Iris Plant Database*, que fue tomada del Repositorio de *Machine Learning* del Departamento de Información y Ciencias de la Computación de la *University of California, Irvine* [58]. Esta es una base de datos clásica en el área de Reconocimiento de Patrones y, a pesar de ser una de las primeras bases de datos utilizadas para probar algoritmo del área, sigue siendo muy utilizada.

Cada patrón que representa una planta de iris tiene 4 atributos más uno que representa la clase. En esta base de datos los rasgos que representan el largo y ancho de los pétalos están altamente correlacionados. Asimismo, se conoce que mientras la clase 1 (Iris Setosa) es linealmente separable de las otras dos clases, las clases 2 y 3 (Iris Versicolor e Iris Virginica, respectivamente) no son linealmente separables entre sí.

Se realizaron 100 experimentos, donde fue particionado aleatoriamente la base de datos en dos conjuntos mutuamente excluyentes: un conjunto fundamental y un conjunto de prueba, en donde la cantidad de patrones fundamentales que aportó cada clase fue igual al aportado por las otras dos clases. De esos 100 experimentos, se tomaron los 10 mejores resultados de clasificación y se muestran en la Figura 5.6.

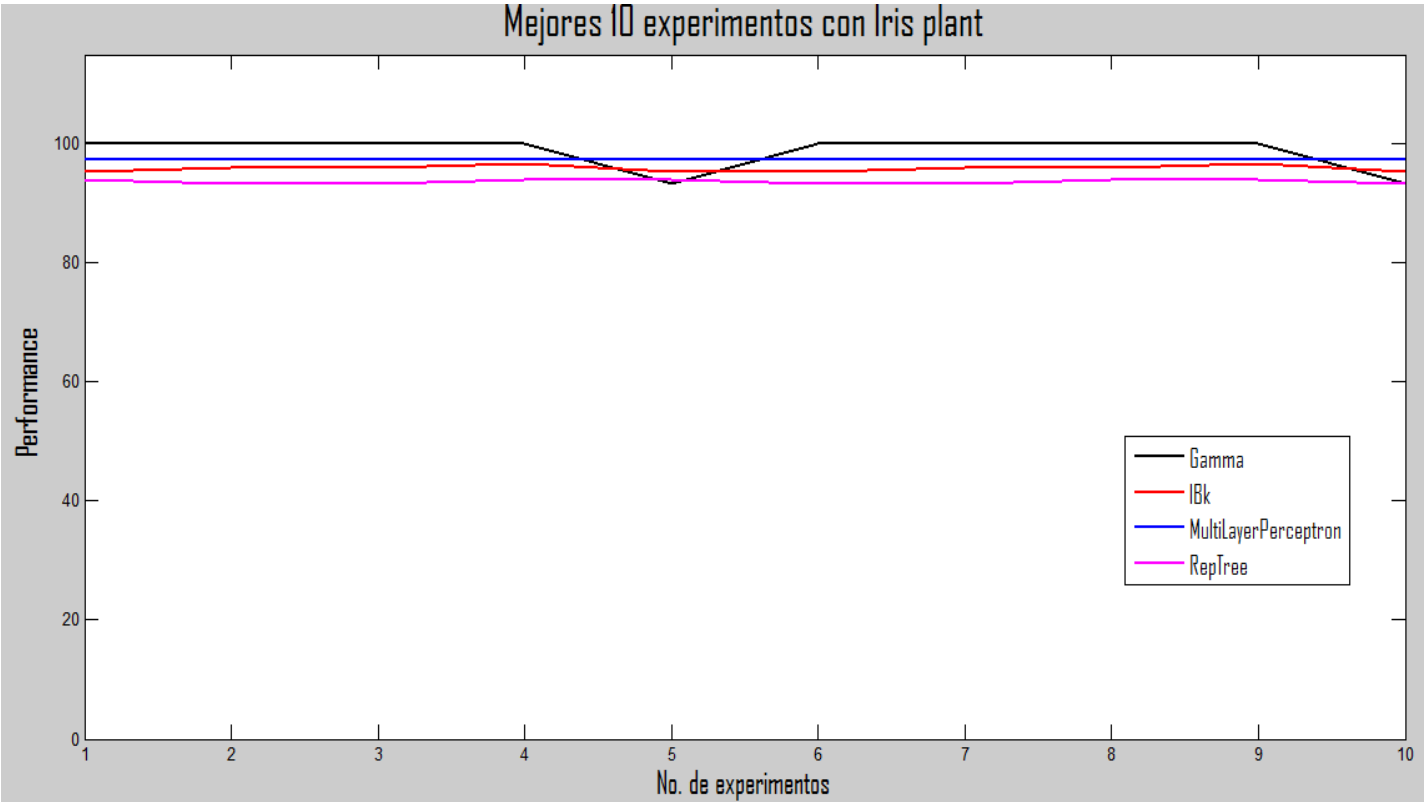


Figura 5.6 *Mejores 10 experimentos con Iris plant*

En la Tabla 5.1 se presenta una comparación de los rendimientos reportados del desempeño de varios clasificadores con respecto al bancode datos *Iris plant*.

En particular, es importante hacer notar que los dos mejores algoritmos según la Tabla 5.1 de la mencionada tesis, al tomar los mejores 10 resultados de 100 experimentos, son el 1-NN con una clasificación correcta de 95.3 % y el *Multi-Layer Perceptron* con la mejor clasificación de 97.3 % hasta ese momento ya que el clasificador Gamma con una clasificación del 98.66 % es importante mencionar que los algoritmos fueron probados con un proceso de 10-*fold cross validation*.

Tabla 5.1: Comparación del rendimiento con el banco de datos "*Iris Plant*"

Clasificador	Mejor Rendimiento
Gamma	98.66 %
MultiLayerPerceptron	97.3 %
IBk	95.3 %
RepTree	94 %

5.2.2 Clasificación de "*Balance scale*"

En primera instancia se presenta la base de datos *Balance scale Database*, que fue tomada del Repositorio de *Machine Learning* del Departamento de Información y Ciencias de la Computación de la *University of California, Irvine* [58]. Esta es una base de datos que resulto de modelar los resultados experimentales psicológicos.

Cada patrón indica la tendencia de la inclinación de la balanza por medio de 4 atributos más uno que representa la clase. Las clases son conocidas como R, L, B, haciendo alusión si, se inclina hacia la derecha, se inclina hacia la izquierda, o esta es equilibrada, por sus siglas en inglés. El banco de datos contiene 3 clases de 288 casos las primeras 2 clases y la tercera contiene 49 casos.

Se realizaron 100 experimentos, donde fue particionado aleatoriamente la base de datos en dos conjuntos mutuamente excluyentes: un conjunto fundamental y un conjunto de prueba, en donde la cantidad de patrones fundamentales que aportó cada clase fue igual al aportado por las otras dos clases. De esos 100 experimentos, se tomaron los 10 mejores resultados de clasificación y se muestran en la Figura 5.7.

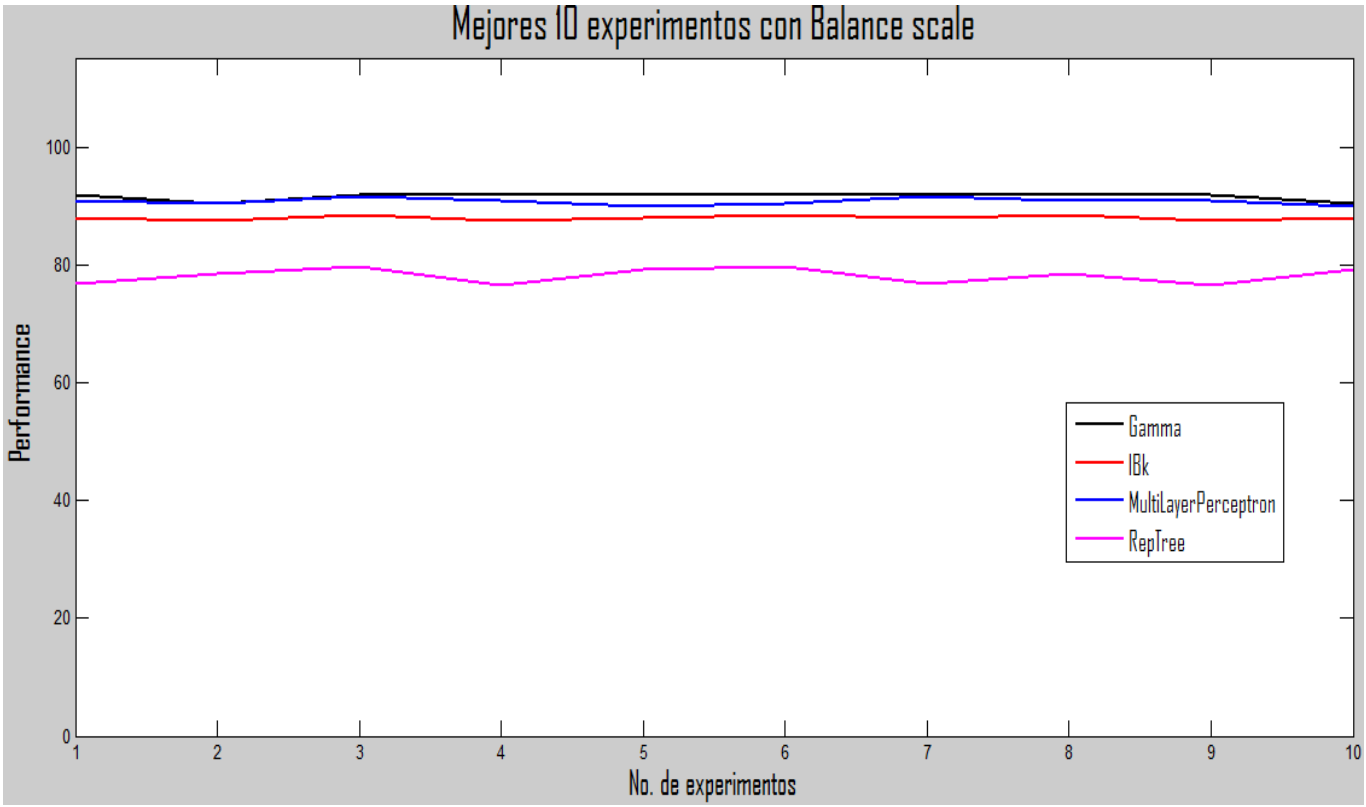


Figura 5.7 *Mejores 10 experimentos con Balance scale*

En la Tabla 5.2 se presenta una comparación de los rendimientos reportados del desempeño de varios clasificadores con respecto a la banco de datos *Balance scale*.

En particular, es importante hacer notar que los dos mejores algoritmos según la Tabla 5.2 de la mencionada tesis, al tomar los mejores 10 resultados de 100 experimentos, son el 1-NN con una clasificación correcta de 88 % y el *Multi-Layer Perceptron* con la mejor clasificación de 90.5 % hasta ese momento ya que el clasificador Gamma con una clasificación del 92.0 % es importante mencionar que los algoritmos fueron probados con un proceso de 10-*fold cross validation*.

Tabla 5.2: Comparación del rendimiento con el banco de datos "*Balance scale"*

Clasificador	Mejor Rendimiento
Gamma	92.0 %
MultiLayerPerceptron	90.5 %
IBk	88 %
RepTree	78.4 %

5.2.3 Predicción de "CO *(*Monóxido de carbono*)*"

Dado que el banco de "CO (Monóxido de carbono)" no tienen una relación directa con las tareas de clasificación o predicción, no es posible comparar los resultados obtenidos con otras publicaciones; por lo que se utilizará el *Root Mean Square Error* (RMSE) ya que esta es una herramienta que sirve para medir el desempeño de las predicciones y de esta forma poder realizar comparaciones con algunos algoritmos implementados en WEKA.

$$\boldsymbol{RSME} = \sqrt{\frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{P_i} - \boldsymbol{O_i})^2}$$

En esta sección se presentan los resultados de la predicción realizada para monóxido de carbono [59], donde el conjunto fundamental son las mediciones obtenidas en el año 2012 y el conjunto de prueba son las mediciones obtenidas en el mes abril del presente.

El resultado del experimento se detalla en la tabla 5.3. Las figuras 5.8, presentan las gráficas de los valores reales contra los valores predichos, por el modelo asociativo gamma.

Tabla 5.3: Comparación de resultados para la predicción de MO

Clasificador	Tamaño del conjunto fundamental	Tamaño del conjunto de prueba	Desempeño (RSME)
RepTree	8749	661	5.2876
MiltiLayerPerceptron	8749	661	5.6717
Gamma	8749	661	6.0086
IBk	8749	661	6.1444

De la tabla anterior podemos observar que el resultado presentado por el clasificador Gamma es competitivos, en relación con otros algoritmos tradicionalmente utilizados para la realizar predicciones; como son las redes neuronales y la regresión.

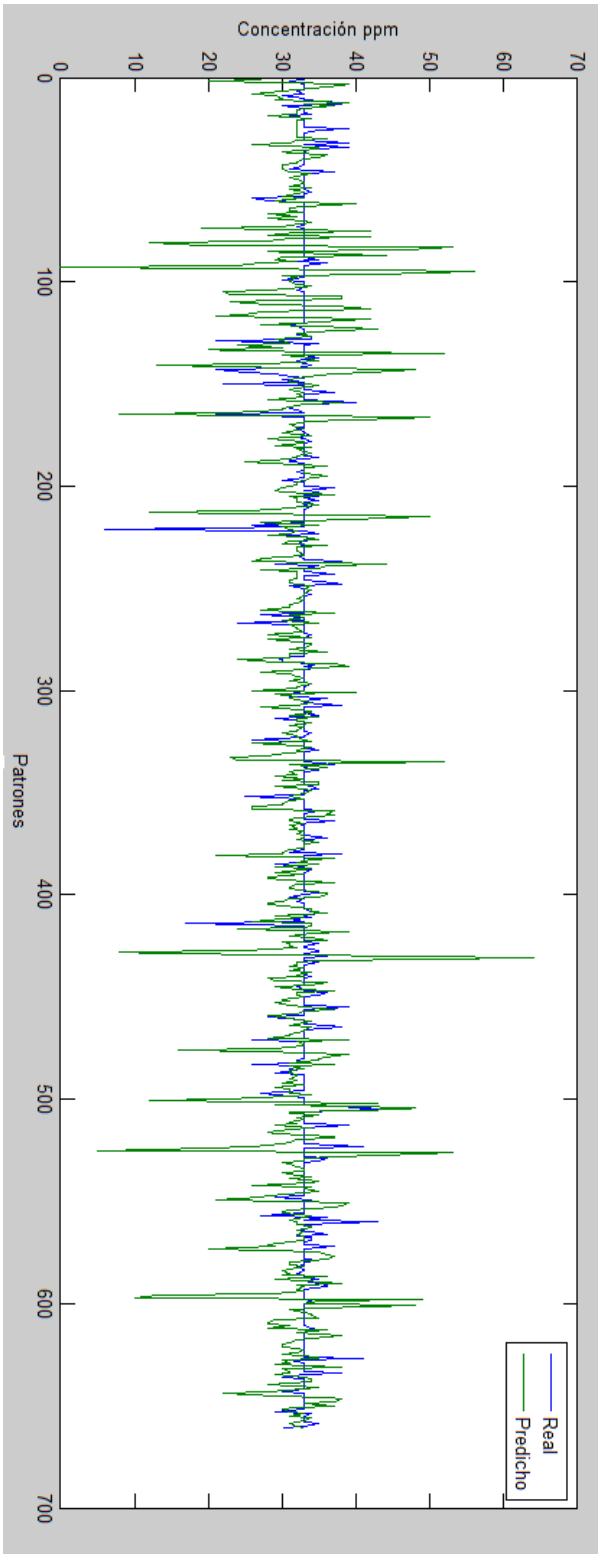


Figura 5.8 *Predicción de monóxido de carbono*

Capítulo 6

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones derivadas de los experimentos y resultados presentados en este trabajo de tesis. Además se proponen algunas ideas de trabajos futuros que se podrán realizar, que den la pauta para desarrollar nuevos trabajos de investigación que abarquen puntos no cubiertos en la presente tesis.

6.1 Conclusiones

Es imprescindible divisar que existe una relación directa entre la dimensionalidad de los datos y el tamaño de las unidades funcionales requeridas para su procesamiento, otro punto remarcable es que la lógica sintetizable, como los recursos de almacenamiento, ambos se encuentran enlazados a la dimensionalidad de los datos y por lo mismo toda la arquitectura depende de la dimensionalidad de los datos directamente.

Un aspecto importante, es que al diseñar la arquitectura es posible implementar operaciones en paralelo, ya que al instanciar múltiples unidades funcionales operando en paralelo, el tiempo de ejecución considerablemente y no por ello se afecta el resultado de la clasificación.

Al analizar diferentes métodos del estado del arte, resulta evidente que el modelo matemático del clasificador Gamma, es en general más sencillo que la mayoría de los métodos usados para la tarea de predicción.

El clasificador Gamma en su tarea original de clasificación los resultados que presenta son buenos ya que es un clasificador competitivo, además se demuestra que puede ser aplicado para la tarea de predicción y que los resultados que entrega en ella también son resultados competitivos.

A pesar de los buenos resultados que presenta la implementación del clasificador Gamma en una arquitectura dedicada, aun se exhiben algunas limitaciones:

- La arquitectura generada funciona como un clasificador de uso general, siempre y cuando se realice un acoplamiento al banco de datos. Para que funcione con el flujo de datos de la arquitectura. Ya que en el caso de no realizarse, la arquitectura funcionara, pero el resultado de la operación de clasificación no se garantiza.

- Los valores para el tamaño de los patrones, *p*, *p0* y *u*, son determinados de forma empírica, basado en el conocimiento acumulado durante los experimentos realizados.

6.2 Trabajo Futuro

- Desde la arquitectura aquí propuesta generar un diseño el cual no dependa de la dimensionalidad de los bancos de datos ingresados. Que sea ajustable sin necesidad de modificar el código.

- Sea implementada una comunicación diferente, esperando que el tiempo total de ejecución reduzca.

- Se documente una metodología para la asignación de los valores *p*, *p0* y *u*.

- Usar esta la arquitectura propuesta en diferentes problemas de aplicación y realizar un estudio para conocer los casos exitosos y los casos donde se requiere de otro tipo de herramientas.

Apéndice A Descripción de la arquitectura

El contenido de este Apéndice está integrado por una descripción de la arquitectura que fue propuesta en este trabajo de tesis, el clasificador Gamma fue el escogido para ser implementado en el FPGA. Dicha arquitectura fue elaborada en el software ISE Disign Suite 14.5, siendo parte fundamental del trabajo el programa Matlab R2011b (*Matrix Laboratory*).

El proceso inicia cuando se ingresa el banco de datos a la computadora por medio de un archivo que es leído por Matlab.

Posteriormente es realizado un pre-procesamiento del banco de datos original, se comienza con un escalamiento del banco de datos para trabajar con números enteros. Se ordena de forma aleatoria el banco de datos. A continuación se divide el banco de datos en conjunto fundamental y conjunto de prueba.

Una vez hecho lo anterior se realiza un arreglo que contenga el conjunto fundamental y el conjunto de prueba, el arreglo generado se envía al FPGA por medio de la comunicación entre el dispositivo y la computadora.

El FPGA recibirá los paquetes de información provenientes de la computadora por medio de una comunicación, dichos paquetes contendrán el conjunto fundamental y el conjunto de prueba. Estos paquetes de información serán colocados en una memoria RAM (*Random Access Memory*) compuesta por un arreglo bidimensional de 8x600.

Una vez llena la memoria RAM, se enviará una señal a la unidad de control (CU) diseñada dentro del FPGA. Dicha unidad controla todas las operaciones para la generar un vector con el resultado de la clasificación de patrones del conjunto de prueba.

El siguiente paso que realiza la CU, es activar dos contadores que serán los encargados de proporcionar la dirección de memoria de la cual se quieren extraer los datos para realizar las operaciones del operador gamma de similitud, en el módulo denominado como Gamma dentro de la arquitectura.

Es imprescindible indicar que los patrones se codifican de la siguiente manera:
Los primeros cuatro bloques de memoria forman el primer patrón, los siguientes cuatro forman el segundo patrón y así sucesivamente hasta obtener el total de patrones del banco de datos.

El resultado obtenido del módulo Gamma se almacena en un registro de propósito general, teniendo cuatro diferentes registros para almacenar el resultado correspondiente de cada rasgo de los patrones comparados.

El contenido de los registros previamente mencionados, solo pueden adquirir los valores de 1 ó 0, pues guardan el resultado del operador gamma de similitud, posteriormente se suman los contenidos de los cuatro registros y el resultado es enviado a una nueva memoria.

Se evalúa el contenido de dicha memoria, una vez que se tienen los resultados de las comparaciones entre el patrón de prueba y el conjunto fundamental, descubriendo si existe un máximo único, en caso que así sea se asigna la clase del patrón; en caso contrario —es decir que no exista un máximo único— se levanta una bandera que le indica a la CU que se tiene que realizar nuevamente el proceso incrementado por uno el valor de teta.

Una vez teniendo los resultados finales para el conjunto de prueba, la CU levanta una bandera que sirve para iniciar el envío de dichos resultados de vuelta a la computadora.

Posteriormente la computadora será la encargada de calcular el rendimiento del clasificador y desplegarlo en la pantalla.

Referencia

[1] **Morris, M. M. & Ciletti, M. D. (2011).** *Digital Design With An Introduction to the Verilog HDL*. USA: Pearson.

[2] **Duda, R.O., Hart, P.E. & Stork, D.G. (2001).** *Pattern Classification*. USA: John Wiley & Sons.

[3] **Young, M. H. & Muroga, S. (1985).** Minimal Covering Problem and PLA Minimization. *International Journal of Computer and Information Sciences*, vol. 14, no. 6. Plenum Publishing Corporation.

[4] **Abdollahi, S.R., Fakhraie, S.M. & Kamarei, M. (2005).** A Crystal-Based Low-Voltage All-Digital Programmable Ring Oscillator. *Analog Integrated Circuits and Signal Processing*, vol. 43, no. 2, pp. 147-157. Springer.

[5] **Wang, B., Cao, Z., Yan, Y., Liu, W. & Wang, Z. (2011).** Fundamental technology for RFID-based supervisory control of shop floor production system. *The International Journal of Advanced Manufacturing Technology*, vol. 57, no. 9-12, pp. 1123-1141. Springer.

[6] **Schwarz, E. M. & Flynn, M. J. (1991).** Cost-Efficient High-Radix Division. *Journal of VLSI Signal Processing*, vol. 3, no. 4, pp. 293-305. Springer.

[7] **Yahyaei, M., Jam, J. E. & Hosnavi, R. (2010).** Controlling the navigation of automatic guided vehicle (AGV) using integrated fuzzy logic controller with programmable logic controller (IFLPLC)—stage 1. *The International Journal of Advanced Manufacturing Technology*, vol. 47, no. 5-8, pp. 795-807. Springer.

[8] **Wang, Z. M., Yao Z. B., Guo H. X. & LV, M. (2012).** Bit stream decoding and SEU-induced failure analysis in SRAM-based FPGAs. *SCIENCE CHINA*, vol. 55, no. 4, pp. 971-982. Springer-Verlag.

[9] **Ching-Yuan, Y. (2008).** A high-frequency CMOS multi-modulus divider for PLL frequency synthesizers. *Analog Integrated Circuits and Signal Processing*, vol. 55, no. 2, pp. 155-162. Springer.

[10] **Tahir, M. A., Bouridane, A. & Kurugollu, F. (2005).** An FPGA Based Coprocessor for GLCM and Haralick Texture Features and their Application in Prostate Cancer Classification. *Analog Integrated Circuits and Signal Processing*, vol. 43, no. 2, pp. 205-215. Springer.

[11] **Zierke, S. & Bakos, J. D. (2010).** FPGA an acceleration of the phylogenetic likelihood function for Bayesian MCMC inference methods. *BMC Bioinformatics*, vol. 11, pp. 184. Biomed Central LTD.

[12] **Horita, T. & Takanami, I. (2013).** An FPGA-based multiple-weight-and-neuron-fault tolerant digital multilayer perceptron. *Neurocomputing*, vol. 99, pp. 570-574. Elsevier.

[13] **Aldape-Pérez, M., Yánez-Márquez, C. & Argüelles-Cruz, A. J. (2008).** FPGA Implementation of Parallel Alpha-Beta Associative Memories. *Image Analysis and Recognition. Lecture Notes in Computer Science*, vol. 5112, pp. 1081-1090. Springer.

[15] **Han, Q., Liao, X., Huang, T., Peng, J., Li, C. & Huang, H. (2012)**. Analysis and design of associative memories based on stability of cellular neural networks. *Neurocomputing*, vol. 97, pp. 192-200. Elsevier.

[16] **Lotrič, U. & Bulić, P. (2012).** Applicability of approximate multipliers in hardware neural networks. *Neurocomputing*, vol. 96, pp. 57-65. Elsevier.

[17] **Misra, J. & Saha, I. (2010).** Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, vol. 74, no. 1-3, pp. 239-255. Elsevier.

[18] **Jacobsson, R. (2010).** Central FPGA-based destination and load control in the LHCb MHz event readout. *Nuclear Instruments and Methods in Physics Research A*, vol. 688, pp. 41-50. Elsevier.

[19] **Nedjah, N., Martins da Silva, R. & Mourelle, L. de M. (2012).** Compact yet efficient hardware implementation of artificial neural networks with customized topology. *Expert Systems with Applications*, vol. 39, no. 10, pp. 9191-9206. Elsevier.

[20] **He, K., Crockett, L. & Stewart, R. (2012).** Dynamic Reconfiguration Technologies Based on FPGA in Software Defined Radio System. *The Journal of Signal Processing Systems*, vol. 69, pp. 75-85. Springer.

[21] **Watson, S. N., Risling, T. E., Hermann, P. M. & Wildering, W. C. (2012).** Failure of delayed nonsynaptic neuronal plasticity underlies age-associated long-term associative memory impairment. *BMC Neuroscience*, vol. 13, pp. 103. Biomed Central LTD.

[22] **Cao, T. P., Elton, D. & Guang, D. (2012).** Fast buffering for FPGA implementation of vision-based object recognition systems. *Journal of Real-Time Image Processing*, vol. 7, pp. 173-183. Springer.

[23] **Herbordt, M. C., Gu, Y., VanCourt, T., Model, J., Sukhwani, B. & Chiu, M. (2008).** Computing Models for FPGA-Based Accelerators. *Computing in Science & Engineering*, vol. 10, no. 6, pp. 35-45. IEEE Computer SOC.

[24] **Bhattacharya, R., Biswas, S. & Mukhopadhyay, S. (2012).** FPGA based chip emulation system for test development of analog and mixed signal circuits: A case study of DC–DC buck converter. *Measurement*, vol. 45, no. 8, pp. 1997–2020. Elsevier.

[25] **Toledo-Moreo, F. J., Martínez-Alvarez, J. J., Garrigós-Guerrero, J. & Ferrández-Vicente, M. J. (2012).** FPGA-based architecture for the real-time computation of 2-D convolution with large kernel size. *Journal of Systems Architecture*, vol. 58, no. 8, pp. 277–285. Elsevier

[26] **Wu, S., Wu, M., Huang, C. & Yang, J. (2012).** FPGA-based implementation of steerable parametric loudspeaker using fractional delay filter. *Applied Acoustics*, vol. 73, no.12 pp. 1271–1281. Elsevier

[27] **Naouar, M.W., Hania, B. B., Slama-Belkhodja, I., Monmasson, E. & Naassani, A.A. (2013).** FPGA-based sliding mode direct control of single phase PWM boost rectifier. *Mathematics and Computers in Simulation*, vol. 91, pp. 249-261. Elsevier.

[28] **Yang, N., Li, D., Zhang, J. & Xi, Y. (2012).** Model predictive controller design and implementation on FPGA with application to motor servo system. *Control Engineering Practice*, vol. 20, no. 11, pp. 1229-1235. Elsevier.

[29] **Kral, J., Awes, T., Muller, H., Rak, J. & Schambach, J. (2012).** L0 trigger for the EMCal detector of the ALICE experiment. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 693, pp. 261–267. Elsevier.

[30] **Park, H., Shannon, V., Biggan, J. & Spann, C. (2012).** Neural activity supporting the formation of associative memory versus source memory. *Brain Research*, vol. 1471, pp. 81–92. Elsevier

[31] **Esmaeili-sani, V., Moussavi-zarandi, A., Akbar-ashrafi, N., Boghrati, B. & Afarideh, H. (2012).** Neutron–gamma discrimination based on bipolar trapezoidal pulse shaping using FPGAs in NE213. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment,* vol. 694, pp. 113–118. Elsevier.

[32] **Cavuslu, M. A., Karakuzu, C. & Karakaya, F. (2012).** Neural identification of dynamic systems on FPGA with improved PSO learning. *Applied Soft Computing*, vol. 12, no. 9, pp. 2707–2718. Elsevier.

[33] **Boccuni, I., Gulino, R. & Palumbo, G (2002)**. Behavioral Model of Analog Circuits for Nonvolatile Memories with VHDL-AMS. *Analog Integrated Circuits and Signal Processing*, vol. 33, pp-19-28. Springer

[34] **Calhoun, J. S., Madisetti, V. K., Reese, R. B. & Egolf, T. (1997)**. Developing and Distributing Component-Level VHDL Models. *Journal of VLSI Signal Processing*, vol. 15, no. 1-2, pp. 111–126. Springer.

[35] **Li, S., Okoon, B., Hella, M., Ismail, M. & Rubeiz, M. (1999).** The Implementation of a VHDL-AMS to SPICE Converter. *Journal of VLSI Signal Processing*, vol. 20, no. 3, pp. 113-121. Springer.

[36] **Bibilo, P. N. (2006).** PRALU-to-VHDL Conversion of Parallel-Algorithm Descriptions for Control Units. *Russian Microelectronics*, vol. 35, no. 4, pp. 262–275. Springer.

[37] **Çavuşlu, M. A., Karakuzu, C, Sahin, S. & Yakut. M. (2011).** Neural network training based on FPGA with floating point number format and it's performance. *Neural Computing & Applications*, vol. 20, num. 2, pp. 195–202. Springer.

[38] **Meagher, R., Sushmitha, M., Rizkalla, M. E., Salama, P. & El-Sharkawy M. (2009).** VHDL Design for Real Time Motion Estimation Video Applications. *The Journal of Signal Processing Systems*, vol. 57, no. 3, pp. 339–348. Springer.

[39] **Steinbuch, K. (1961).** Die Lernmatrix. *Kybernetik*, vol. 1, no. 1, pp. 36-45. Springer.

[40] **Willshaw, D. J., Buneman, O. P. & Longuet-Higgins, H. C. (1969)** Non-holographic associative memory. *Nature*, no. 222, pp. 960-962. Nature Publishing Group.

[41] **Kohonen, T. (1972).** Correlation matrix memories. *IEEE Transactions on Computers*, vol. C - 21, no. 4, pp. 353-359. IEEE Computer SOC.

[42] **Anderson, J. A. (1972).** A simple neural network generating an interactive memory. *Mathematical Biosciences*, vol. 14, no. 3-4, pp. 197-220. Elsevier.

[43] **Amari, S. (1972).** Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, vol. C-21, no. 11, pp. 1197-1206. IEEE Computer SOC.

[44] **Nakano, K. (1972).** Associatron-A model of associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 2, no. 3, pp. 380-388. IEEE Systems, Man, and Cybernetics SOC.

[45] **Hopfield, J. J. (1982).** Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554-2558.

[46] **Kosko, B. (1988).** Bidirectional associative memories. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60. IEEE Systems, Man, and Cybernetics SOC.

[47] **Ritter, G. X., Sussner, P. & Diaz-de-Leon, J. L. (1998).** Morphological associative memories. *IEEE Transactions on Neural Networks*, vol. 9, no.2, pp. 281-293. IEEE Computational Intelligence SOC.

[48] **Yánez M. C. (2002).** *Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios*. Tesis de Doctorado en Ciencias de la Computación, Instituto Politécnico Nacional, Centro de Investigación en Computación. D.F., México.

[49] **Lee, A. E. & Seshia, S. A. (2011)**. *Introduction to Embedded Systems A Cyber-Physical Systems Approach.* USA: LeeSeshia.org.

[50] **Wolf, W**. **(2007)**. *In Praise of High-Performance Embedded Computing: Architectures, Applications, and Methodologies.* USA: Elsevier.

[51] **López Y. I. (2011).** T*eoría y aplicaciones del clasificador Gamma*. Tesis de Doctorado en Ciencias de la Computación, Instituto Politécnico Nacional, Centro de Investigación en Computación. D.F., México.

[52] **Rosen, K. H**. **(2012)**. *Discrete Mathematics and Its Applications.* USA: McGraw-Hill.

[53] **Flores C. R. (2006).** *Memorias asociativas Alfa-Beta basadas en el código Johnson-Möbius modificado*. Tesis de Maestría en Ciencias de la Computación, Instituto Politécnico Nacional, Centro de Investigación en Computación. D.F., México.

[54] **Schelter, B., Winterhalder, M. & Timmer, J**. **(2006)** *Handbook of Time Series Analysis*. Germany: Wiley-Verlag.

[55] **Palit, A. K. & Popovic, D. (2005)** *Computational Intelligence in Time Series Forecasting.* England: Springer-Verlag.

[56] **Hernández Zavala, A., Batyrshin, Z. I., Camacho Nieto, O. & Castillo, O. (2013)** *Conjunction and disjunction Operations for Digital Fuzzy Hardware*. Applied Soft Computing, vol. 13, no. 7, pp. 3248-3258. Elsevier.

[57] **Santiago-Montero, R. (2003).** *Clasificador híbrido de patrones basado en la Lernmatrix de Steinbuch y el Linear Associator de Anderson-Kohonen.* Tesis de Maestría en Ciencias de la Computación, Instituto Politécnico Nacional, Centro de Investigación en Computación. D.F., México.

[58] **Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J. (1998).** UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html

[59] **Sistema de Monitoreo Atmosférico (SIMAT). (1986).** Red Automática de Monitoreo Atmosférico (RAMA). Available at http://www.calidadaire.df.gob.mx/calidadaire/productos/basesdedatos/bd_rama.php

[60] **Uriarte Arcia, A. (2012).** *Procesamiento de datos de monitoreo atmosférico usando clasificación no convencional*. Tesis de Maestría en Ciencias de la Computación, Instituto Politécnico Nacional, Centro de Investigación en Computación. D.F., México.