CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

Desarrollo de un clasificador asociativo para el reconocimiento de caracteres implementado en FPGA

TESIS

QUE PARA OBTENER EL GRADO ACADÉMICO DE:

MAESTRO EN CIENCIA Y TECNOLOGÍA CON ESPECIALIDAD EN MECATRÓNICA

PRESENTA:

Alberto Vázquez Cervantes.

ASESORES

Dr. Hugo Jiménez Hernández.

M.C. Luciano Nava Balanzar.

SANTIAGO DE QUERÉTARO, QUERÉTARO,
16 de diciembre de 2013

CIENCIA Y TECNOLOGÍA

Director de Posgrado
PICYT – CIDESI
Querétaro

Los abajo firmantes, miembros del Comité Tutorial de la alumno **Alberto Vazquez Cervantes**, una vez leída y revisada la Tesis titulada **"Desarrollo de un clasificador asociativo para el reconocimiento de caracteres implementado en FPGA"**, aceptamos que la referida tesis revisada y corregida sea presentada por la alumno para aspirar al grado de **Maestría en Ciencia y Tecnología** en la opción terminal de **Mecatronica** durante el Examen de Grado correspondiente.

Y para que así conste firmo la presente a los 13 días del mes de diciembre del año dos mil trece.

Hugo Jiménez Hdz

Dr. Hugo Jiménez Hernández.
Tutor Académico.

M. en C. y T. Luciano Nava Balanzar.
Co-director.

CIENCIA Y TECNOLOGÍA

Director de Posgrado
PICYT – CIDESI
Querétaro

Los abajo firmantes, miembros del Jurado del Examen de Grado de la alumno **Alberto Vazquez Cervantes**, una vez leída y revisada la Tesis titulada **"Desarrollo de un clasificador asociativo para el reconocimiento de caracteres implementado en FPGA"**, aceptamos que la referida tesis revisada y corregida sea presentada por la alumno para aspirar al grado de **Maestría en Ciencia y Tecnología** en la opción terminal de **Mecatronica** durante el Examen de Grado correspondiente.

Y para que así conste firmamos la presente a los 13 días del mes de diciembre del año dos mil trece.

Dr. Jorge Alberto Soto Cajiga
Presidente

M. en C. y T. Leonardo Barriga Rodríguez
Secretario

Dr. Hugo Jiménez Hernández
Vocal

Resumen

En esta tesis se presenta una nueva propuesta para clasificación de caracteres en tiempo real, mediante una plataforma de propósito específico construida con FPGA (Field Programmable Gate Array). Este dispositivo se encarga de realizar el reconocimiento de cada carácter y de algunos otros procesos que pueden beneficiarse por un cómputo paralelo.

Los clasificadores clásicos de las Memorias Asociativas han tomado importancia en la actualidad por ser un tipo de reconocedor basadas en modelos de clasificación lineales. Sin embargo, al igual que la gran mayoría de los clasificadores, la capacidad de detectar en forma eficiente la clase de pertenencia depende totalmente de la expresividad codificada en los atributos usados para codificar los eventos. En este trabajo se presenta un clasificador basado en una memoria de aprendizaje Steinbuch. Funciona con la introducción de patrones binarios, los cuales son codificados con formando una matriz M, se multiplica por un patrón de prueba dando como resultado las clases fundamentales con una tolerancia al error. El método de muestreo utiliza el criterio de la *Regla Mayor* para determinar los patrones binarios que son distintivos de cada carácter a clasificar. Teniendo en cuenta estas técnicas, permite extraer las características de una imagen más manera que los métodos convencionales para una mejor categorización.

En el modelo experimental se analiza la reducción de complejidad computacional en dos ámbitos importantes, Disminuyendo las variables al usar técnicas de muestreo aleatorio, facilitando el procesamiento de la información. La segunda administra el uso efectivo de recursos en los dispositivos reconfigurables disminuyendo los ciclos utilizados para realizar las operaciones y por consecuencia el tiempo de ejecución.

Dedicatoria

Con todo mi cariño y mi amor para las personas que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre mi corazón y mi agradecimiento.

Papá y mamá

Agradecimientos

A CIDESI, por su espacio y material prestado para el desarrollo de esta tesis.

A Conacyt, quien me ofreció su beca de maestría para mi manutención a lo largo del proyecto de investigación.

A mi asesor de tesis, Hugo Jiménez H. por su esfuerzo y dedicación, quien con sus conocimientos, experiencia, paciencia y motivación ha logrado en mí que pueda terminar mis estudios con éxito.

A mi director de tesis, Luciano Nava por su confianza y apoyo durante el desarrollo de mi formación.

A mi asesor técnico, Jorge Alberto Soto por compartir su conocimiento y colaboración.

A mi padre Floriberto, que me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.

A mi madre Carmen, por su confianza y el apoyo, que sin duda alguna en el trayecto de mi vida me ha demostrado su amor, corrigiendo mis faltas y celebrando mis triunfos.

A mi hermana Ana, que con sus consejos me ha ayudado a afrontar los retos que se me han presentado a lo largo de mi vida.

Agradezco a todas las personas que de una u otra forma estuvieron conmigo, porque cada una aportó con un granito de arena; y es por ello que a todos y cada uno de ustedes les dedico todo el esfuerzo, sacrificio y tiempo que entregué a esta tesis.

Índice general

Índice de figuras

Capítulo 1

Introducción.

Uno de los principales aspectos de las aplicaciones de la visión artificial es la descripción de eventos en forma automática en un determinado fenómeno. Esta descripción comprende de la identificación y la localización de estructuras que denotan al evento a analizar. Las aproximaciones actuales ofrecen soluciones algorítmicas a una diversidad de eventos a identificar. Pero, el tiempo y los recursos de computo que requieren estos algoritmos es el principal problema de las aplicaciones en visión artificial. Especialmente en los escenarios donde el tiempo mínimo de respuesta es muy *corto* y la *complejidad* de los algoritmos muy grande. La mayor demanda de prestaciones de los algoritmos de procesamiento de imágenes, unido incremento en resolución espacial de las imágenes, hace que cada vez sean mayores las demandas de recursos computacionales. Además para trabajar en tiempo real, las exigencias de los tiempos de ejecución de los algoritmos son aún más críticas.

Habitualmente, las plataformas elegidas para hacer estos algoritmos de visión son las basadas en plataformas secuenciales utilizando un lenguaje de programación pensado en una sola ruta de ejecución. Sin embargo, éstas no resultan óptimas en muchas aplicaciones desde un punto de vista de rendimiento y eficiencia. Un caso particular dentro de el área de visión, son los algoritmos de clasificación e identificación de características, donde debido a la necesidad de usar eficientemente los recursos de cómputo se justifica la búsqueda de nuevos sistemas de clasificación lineales basadas en matrices. Por tanto, es importante seleccionar los recursos de hardware necesarios, en función del nivel de complejidad de procesamiento requerido y la forma de operar los datos. Esto permitirá explotar al máximo el rendimiento del sistema al utilizar En arquitecturas paralelas donde se asuma que cada componente interactúa independientemente de sus vecinos.

La mejora en los dispositivos reconfigurables ha posibilitado la implementación de sistemas

digitales de alto nivel, comúnmente denominados como SOC (System On Chip), con un ahorro económico y en tiempo de diseño. Resulta económico, porque el número de circuitos integrados se reduce al poder integrar en una FPGA numerosos sistemas digitales. Esto permite el desarrollo de clasificadores de gran eficiencia como el que se presenta en este trabajo.

1.1. Objetivos general

Generar la arquitectura paralela para reducir la complejidad computacional de un algoritmo clasificador basado en matrices de utilizando un dispositivo reconfigurable.
Este algoritmo de clasificación será utilizado para reconocer eficientemente caracteres digitalizados de letras de fuentes digitales.

1.2. Objetivos particulares

- Determinar un modelo de muestro para codificar la información de entrada.
- Obtener la base de entrenamiento para las clases.
- Realizar la codificación de patrones de entrada.
- Analizar el clasificador con la arquitectura computacional clásica.
- Implementar la matriz de aprendizaje en FPGA.
- Validar la reducción de complejidad.
- Comparar el rendimiento con respecto a una arquitectura secuencial.

1.3. Hipótesis

Si se tiene un clasificador basado en operadores matriciales, entonces es posible implementar un algoritmo en forma concurrente de tal manera que se administre la forma en que se realizan las operaciones, reduciendo el numero de ciclos de reloj de la maquina en un dispositivo reconfigurable al aumentar el número de operaciones que se efectúan por ciclo, y consecuentemente el tiempo de ejecución se ve disminuido.

1.4. Justificación

La solución práctica de un problema esta relacionado con el algoritmo; el cual está supeditado a la arquitectura de hardware y a la codificación. Por esta situación, se tiene que considerar la forma en que se administran los recursos en el hardware para definir la funcionalidad y forma en desarrollar un algoritmo. En este trabajo se presenta un esquema de como reducir la complejidad de clasificador asociativo tipo Steinbuch. La propuesta consiste en dos aspectos: (a) Expresar un conjunto de características distinguibles de forma en letras molde en forma compacta y (b) en utilizar en forma eficiente el paralelismo explicito en el operador producto matricial, al ser codificado en forma eficiente con FPGA, maximizando en cada ciclo de operación el uso de los recursos para disminuir el total de ciclos necesarios para realizar la clasificación de cada letra[1].

1.5. Planteamiento del problema

Esta tesis toma como referencia, el clasificador asociativo llamado *Lernmatrix* (Matriz de aprendizaje) el cual es la base de las memorias asociativas modernas [2][3]. Tiene la característica de clasificar patrones extraídos de imágenes que contienen grafías, que son el resultado de un muestreo aleatorio para codificar cada imagen [4]. Para lograr dicho propósito, esta aproximación utiliza operaciones matriciales. El uso intenso de operaciones matriciales causa que las implementaciones mediante algoritmos secuenciales este restringida a escenarios muy concretos y el uso en tiempos cortos se vea restringido. Por esta razón se plantea una nueva arquitectura para implementarse con FPGA. En esta plataforma se realiza la optimización de los recursos internos de dichos elementos. Por otra parte este dispositivo tiene la capacidad de realizar operaciones en forma paralela, permitiendo la reducción de ciclos de maquina y por disminuyendo la complejidad temporal. Para garantizar una correcta implementación de la arquitectura, en una primera etapa se realiza un análisis previo en simulación en computadora; luego se hace una implementación física en un FPGA y posteriormente un análisis de complejidad en la implementación en

Software (simple hilo de ejecución) y en FPGA (vesión paralela del algoritmo). Dentro la arquitectura se introducen datos de entrada del sistema que realizara la fase de aprendizaje y consulta en la FPGA para tener mayor velocidad de procesamiento en la clasificación. Finalmente se interpretan los datos obtenidos por el hardware. En Figura 1.1 se muestra el diagrama general, en este se aprecia que la memoria de Steinbuch en conjunto con la CPU.

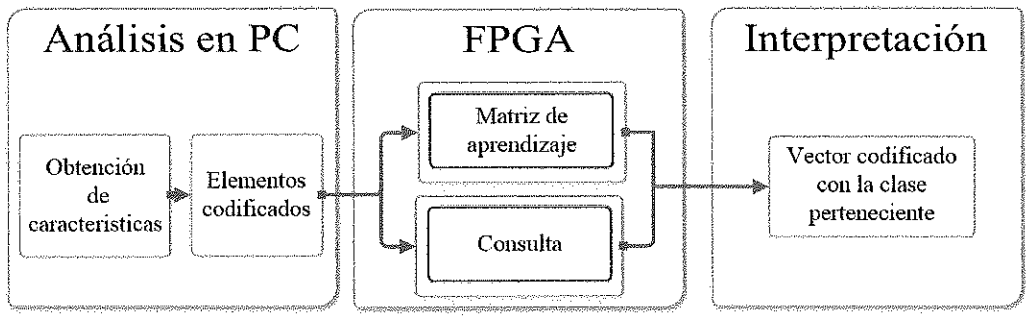


Figura 1.1: Esquema del proyecto.

1.6. Organización

El documento se organiza de la siguiente manera:
En §2 es un panorama general del uso de los dispositivos reconfigurables hasta el momento, además de las técnicas de clasificación y elementos que intervienen para realizar dicha tarea.

En §3 se describe el tema de clasificación de caracteres en FPGA. También los conceptos clave para comprender el objetivo del trabajo, y el análisis bibliográfico realizado para el trabajo, además de la estructuración lógica del material y el análisis crítico del mismo bajo el objetivo de esta tesis.

Después en §4 se introduce a los métodos que se seleccionaron para el desarrollo del proyecto, se presentan las fases que se llevan a cabo y el análisis de las técnicas usadas.

En §5 se evalúan los métodos descritos en el capitulo anterior mediante un proceso experimental, y se presentan los resultados obtenidos midiendo la complejidad en ciclos de reloj.

Finalmente se muestran las conclusiones de lo realizado, observaciones, trabajo a futuro y lineas de investigación que se desarrollan con la culminación de la tesis.

Capítulo 2

Estado del Arte

La clasificación de caracteres en tiempo real se requiere en muchas aplicaciones que demandan un rápido y continuo cálculo de operaciones a la hora de ejecutarse en un sistema determinado. Esto es causa de la elevada carga computacional que lleva asociada los algoritmos de reconocimiento. Recientemente los dispositivos hardware reconfigurables están siendo una alternativa en la clasificación debido a su bajo precio, y sus altas prestaciones. Sin embargo, este tipo de dispositivos presentan grandes problemas desde el punto de vista de implementación, porque los lenguajes para su desarrollo son de bajo nivel.

2.1. Dispositivos reconfigurables

El uso de DSP's (Digital Signal Processor) en el reconocimiento de caracteres es otra alternativa empleada [5]. Las diferentes plataformas basadas en ellos proporcionan una buena autonomía y una elevada tasa de operaciones por unidad de tiempo. Esto es debido a la arquitectura interna de la CPU cuyas características se adaptan perfectamente al reconocimiento. Otra ventaja es la posibilidad de desarrollar diferentes algoritmos en lenguajes de alto nivel, dada la existencia de múltiples compiladores con elevado grado de prestaciones. Sin embargo, su naturaleza secuencial y de propósito específico no permite en muchas ocasiones alcanzar un procesamiento en tiempo real con estos dispositivos.
Otra alternativa a la clasificación, es el uso de dispositivos de hardware de propósito específico (ASIC) [6]. Este tipo de circuitos pueden proporcionar soluciones particulares a determinados algoritmos de procesamiento de imágenes. Sin embargo, presentan algunas desventajas:

- Puede tener un costo elevado si el volumen de elementos necesarios no justifica los costos de fabricación.

- El diseño y desarrollo de este tipo de circuitos requiere un tiempo excesivo para validar y fabricar el circuito. Este hecho causa demora en el desarrollo de proyectos. Por esta razón se está trabajando en el desarrollo de herramientas que realicen la implementación física automáticamente, partiendo de una netlist como la usada por las herramientas de emplazamiento de FPGAs.

- Si bien el rendimiento es óptimo, ya que se diseñan para una aplicación concreta, la falta de flexibilidad para modificar los algoritmos implementados constituye una desventaja.

Con objeto de evitar la rigidez de los circuitos ASIC, la alternativa que se están dirigiendo numerosas líneas de investigación es la de emplear dispositivos con mayor flexibilidad manteniendo las ventajas de los elementos ASIC en cuanto a rendimiento[7]. Un ejemplo de estos son los dispositivos FPGA, los cuales permiten reconfigurarse o reprogramarse múltiples veces, manteniendo la flexibilidad.

De la información presentada se comprueba como la opción de hardware reconfigurable ofrece buenos resultados para los diferentes índices evaluados. Una de las principales características de este tipo de sistemas, es la posibilidad de ejecutar algoritmos de forma concurrente y no de manera secuencial. Esto aplicado a algoritmos de reconocimiento, permite particionar su ejecución, siendo estas arquitecturas más propicias para la ejecución de procesos de manejo de imágenes.

Dentro de la clasificación de caracteres con dispositivos reconfigurables no existen muchos trabajos realizados hasta el momento. Esto es debido principalmente a dos razones: (a) la aparición de estos dispositivos es reciente (Aproximadamente hace veinte años), y (b) la falta de herramientas de programación de alto nivel. En ventaja que muchos algoritmos de clasificación permiten particionar su ejecución secuencial, facilitando diseñar versiones concurrentes y paralelas, acelerando así su tiempo de ejecución.

	Rendimiento	Coste	Consumo	Flexibilidad	Nivel de esfuerzo en el diseño
ASIC	Alto	Alto	Bajo	Bajo	Alto
DSP	Medio	Medio	Medio	Medio	Medio
Procesador	Bajo	Bajo	Medio	Alto	Bajo
Hw. Reconf.	Medio	Medio	Alto	Alto	Medio

Figura 2.1: Comparativa entre DSPs, ASICs, hardware reconfigurable y procesadores de propósito general

2.2. Modelos de memorias asociativas

Las actividades humanas han representado un tema de investigación en diversas áreas que tratan de entender los procesos internos involucrados. Sin embargo, la complejidad y la diversidad de este tema han permitido proponer enfoques en diferentes áreas [8]. Estos enfoques pretenden imitar a través de simulación/emulación algunos comportamientos específicos. En particular, los modelos cognitivos incluyen modelos para explicar el pensamiento, el raciocinio, la inferencia y la abstracción, (por mencionar algunos), representando un desafío por todos los factores y la complejidad de los procesos involucrados. Los primeros intentos fueron inspirados en los enfoques reduccionistas [9], donde los trabajos más significativos[10]. Este pretendía explicar los procesos cognitivos humanos como un conjunto de modelos monolíticos. Estos modelos son caracterizados debido a que se pueden reducir en una abstracción sintética definida, expresada por una aproximación matemática o lógica [11]. Sin embargo, estos enfoques sólo son útiles en escenarios específicos, en los que la información está libre de afectaciones como el ruido o datos incompletos. Después, los enfoques reduccionistas fueron sustituidos por enfoques conectivos. Los enfoques conectivos introducen un nuevo marco y una nueva metáfora para

emular y simular procesos cognitivos [10][12][13]. Este enfoque modela diferentes procesos cognitivos como estados interconectados. Los estados representan modelos simples de unidades de proceso de la información y la estructura de las interconexiones define topologías de interacción. Aspectos como la aparición, el cambio de fase, y la auto-organización se explican comúnmente con este enfoque[14]. Además, los modelos computacionales más simples resultan fáciles de implementar y de probar. Por esta razón, la introducción de la redundancia de los datos y la redundancia en el número de unidades de cálculo son capaces de manejar tareas más complicadas, que de otra forma, no se realizarían por una sola unidad de cálculo. Algunos estudios recientes que se ocupan de este tema incluyen [13][15][2], los cuales abordan los problemas teóricos modelados como una red y una interacción de unidades de computación más simples. Uno de estos problemas teóricos se relaciona con el proceso cognitivo realizado por el cerebro, que incluye la tarea de asociar y relacionar conceptos. Este proceso representa una oportunidad de investigación porque hay muchos factores involucrados [16][17] tales como la gestión de la información que incluye la forma de la representación y la forma de manipularlas. Adicionalmente él último factor incluye, es cómo se define el proceso de asociación usando las características codificadas. En este orden de ideas, algunos de los trabajos representativos incluyen [13][15][18], donde los autores proponen diferentes modelos basados en la linealidad de los datos expresada por matrices. En este sentido, el reto principal consiste en el establecimiento de un conjunto de características adecuadas para caracterizar un fenómeno a analizar, las cuales son invariantes a ciertas situaciones fenómeno [13][2][19]. En áreas como el análisis de imágenes, hay algunos enfoques como [20][21][22], donde los autores muestran algunos criterios, enfocados en la extracción y clasificación de características de la imagen y sus invarianzas para diferentes condiciones del escenario. Por otro lado, los clasificadores, que se utilizan para el agrupamiento de datos, utilizan características altamente confiables sobre su comportamiento, es decir, con características de alto grado de similitud, dentro del espacio de características, son dispersadas en cúmulos; los cuales son factibles para la agrupación a través de cualquier criterio de agrupamiento. En esta rama de

la investigación, los enfoques representativos [23][24][25][26] muestran diferentes criterios de agrupación para descubrir patrones en los datos brutos. Los fundamentos y paradigmas utilizados en estos trabajos son diferentes, y por lo tanto el resultado es ligeramente distinto en escenarios similares. Un clasificador muy aceptado es la memoria asociativa [13][2] como un tipo particular de red neuronal artificial (ANN). El enfoque de memoria asociativa (AMA) es de tipo conectivo. Este enfoque utiliza la linealidad expresada en un conjunto de datos, así como una transformación lineal. Este enfoque por lo general es referido como un tipo de red neuronal, es decir, representa un enfoque conectivo donde se utiliza una matriz W es usada para asociar las entradas y salidas. En varios escenarios podría representar una aproximación sólida porque soporta la interferencia de ruido en los datos [18][23]. Los primeros enfoques asociativos fueron desarrollados por Steinbuch [3], quién propone un criterio para definir un clasificador lineal de patrones binarios. Después, una red neuronal particular tuvo influencia para desarrollar otro modelo conectivo: Hoppfield Model[13]. Este modelo utiliza la idea de atractores lineales como una manera de codificar la información en clases. Luego, los modelos como [27] y [28] propusieron modelos asociativos basados en la separación de linealidad y la codificación de los datos. Esto es que representan los elementos de memoria como los patrones y establecen un criterio lineal para relacionar cada uno con una clase predefinida. Después, el modelo de ADAM [29] es presentado como un modelo de mejora del modelo de Hoppfield [13], añadiendo una matriz extra para indexar los elementos dentro de la memoria.

CAPÍTULO 3

FUNDAMENTO TEÓRICO.

En este capítulo se muestra el marco conceptual de los clasificadores y las herramientas para la reducción de complejidad. También se comenta la arquitectura física donde se implementará la propuesta y finalmente se da un panorama de las memorias asociativas.

3.1. Algoritmos y Complejidad computacional

Un algoritmo es un procedimiento computacional bien definido, el cual considera un valor o conjunto de valores como de entrada y a través de una secuencia de instrucciones organizadas produce un valor o conjunto de salida, en un tiempo determinado con la finalidad de dar solución a un problema específico. Las características principales de un algoritmo son:

- *Preciso y Definido.* Cada paso debe ser definido en forma precisa e indicar el orden de realización.
- *Datos de entrada* (input). El algoritmo recibe datos iniciales antes de su ejecución.
- *Datos de salida* (output). El algoritmo tiene una o más salidas, es decir, datos que tienen una relación específica respecto a los datos de entrada.
- *Generalidad.* Independientemente de las veces que se siga un algoritmo, se debe obtener el mismo resultado.
- *Finito.* Un algoritmo debe terminar siempre después de un número finito de pasos.

Al analizar un algoritmo se considera el tiempo de ejecución y consiste en el número de operaciones elementales que ejecuta en cada paso. Dicho análisis se concentra generalmente

en encontrar el peor caso de tiempo de ejecución, es decir, el mayor tiempo que tardaría el algoritmo en obtener los valores de salida.

En este contexto, la complejidad computacional estudia la eficiencia de los algoritmos estableciendo su efectividad de acuerdo a los ciclos y el espacio requerido en la computadora, ayudando a evaluar la viabilidad de la implementación práctica en tiempo y costo. Por otra parte, provee herramientas para clasificar la dificultad inherente de un problema, de esta manera se puede conocer previamente si la búsqueda de un algoritmo eficiente para la solución de dicho problema es posible o no. Existen problemas que únicamente pueden resolverse utilizando un algoritmo de tiempo exponencial o incluso, puede ser que no exista algoritmo alguno, por lo cual, al tener este tipo de información puede optarse por la búsqueda o aplicación de técnicas heurísticas existentes, que si bien no garantizan una solución óptima, si pueden proporcionar una buena solución o una aproximada.

3.2. Análisis asintótico de los algoritmos

El análisis asintótico permite conocer la eficiencia de un algoritmo en base al número de operaciones por ejecución. Cuando el tamaño de los datos de entrada es suficientemente grande de tal forma que las constantes y los términos de menor orden no afectan. Para expresar la tasa de crecimiento de una función se toma en cuenta el término dominante con respecto a n (donde n es el número de entradas) y se se ignoran las constantes.

Este tipo de análisis permite facilitar la elección del mejor algoritmo entre varios y, determinar cual es el más conveniente aplicar medidas para la eficiencia que implementarlo y medir la eficacia después de cada ejecución.

3.2.1. Notación asintótica

Las notaciones para el tiempo de ejecución asintótico de un algoritmo se definen en términos de funciones cuyo dominio es el conjunto de números naturales. Las notaciones se

utilizan para describir la función del peor caso del tiempo de ejecución, $T(n)$, normalmente definidas en tamaños de entrada enteros.

Notación O. *Representa una cota asintótica superior. Sea una función* $g(n)$, *se denota por* $O(g(n))$ *el conjunto de funciones:*

$$f(n) : \exists c > 0, \quad n_0 | 0 \leq f(n) \leq c(g(n)) \quad \forall n \geq n_0 \tag{3.1}$$

La notación O se utiliza para acotar el peor caso del tiempo de ejecución de un algoritmo como se muestra en la Figura 3.1.

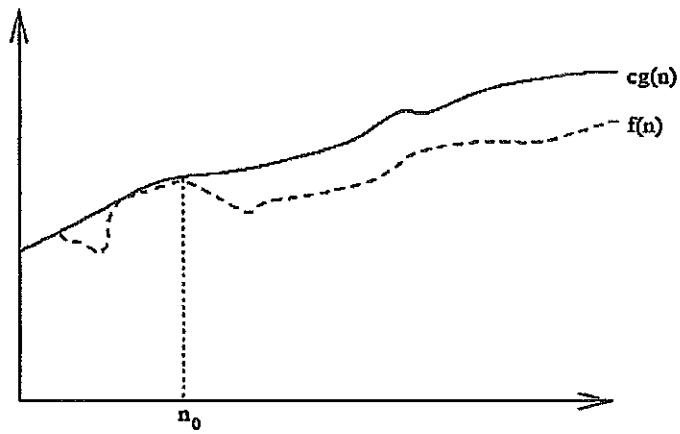


Figura 3.1: f(n)=O(g(n))

3.2.2. Análisis del peor caso del tiempo de ejecución

Los tiempos de ejecución de un algoritmo pueden dividirse en: el mejor, el probabilístico y, el peor. Pero por fines prácticos, para analizar un algoritmo normalmente se considera el peor caso, es decir, el tiempo más largo para cualquier entrada de tamaño n. Las razones son las siguientes:

- El mejor caso no es representativo porque se analiza una entrada para la cual el problema es trivial.

- El caso probabilístico sería una muy buena opción, no obstante, el problema principal es que matemáticamente puede resultar muy difícil de medir.
- El peor caso es muy práctico, debido a que, en algunos casos es muy cercano al probabilístico e incluso a observaciones experimentales. De hecho, es muy frecuente que el caso promedio sea tan malo como el peor caso.
- El peor tiempo de ejecución de un algoritmo es una cota superior del tiempo de ejecución para cualquier entrada, garantizando que el algoritmo no tardará más.
- Para algunos algoritmos, el peor caso ocurre muy rara vez.

3.2.3. Complejidad de los problemas

En complemento a la medición de complejidad de los algoritmos, se tiene la complejidad en representar ciertos problemas como un algoritmo. La clasificación de la complejidad de los problemas es en cuatro clases:

1. Problemas Indecidibles. No se puede escribir un algoritmo para su solución, por lo tanto son los problemas de complejidad más alta.
2. Problemas Intratables. No se puede desarrollar un algoritmo de tiempo polinomial, únicamente algoritmos exponenciales.
3. Problemas NP (Polinomial no determinístico). Son los problemas para los cuales la factibilidad del problema utilizando el correspondiente problema de decisión, puede ser verificada en tiempo polinomial, sin embargo, el problema solo puede resolverse con algoritmos no determinísticos. Un algoritmo no determinístico es un modelo teórico de computación donde la computadora debe adivinar que paso seguir, y en caso de existir un conjunto de adivinanzas que la computadora debiera resolver, entonces se supone que adivina correctamente. Obviamente este tipo de modelo es imposible de implementar.

4. Problemas P. Si un problema está en la clase P, se dice que es polinomial y significa que existe un algoritmo de tiempo polinomial para su solución.

Y todos estos problemas pueden ser relacionados como se muestra en la Figura 3.2

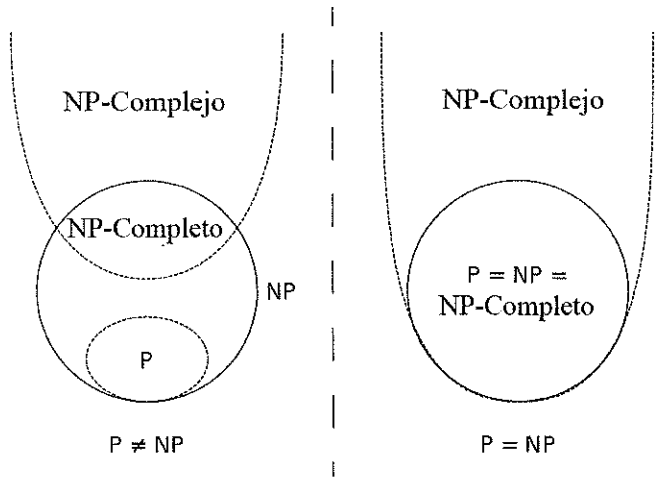


Figura 3.2: Diagrama de Euler de las familias de problemas P, NP, NP-Completo, y NP-Complejo.

3.3. Reconocedor de patrones

Establecer un modelo de análisis para clasificar y reconocer las estructuras de un fenómeno por sus características, es una tarea dependiente del contexto; las soluciones proporcionadas mediante un modelo son especificas a casos acotados, resultando que los modelos conceptuales no son generalizables por la diversidad de las situaciones de operación además muchas de las situaciones están limitadas por eventos externos al fenómeno, que no pueden ser caracterizados. Algunas de estas situaciones que influyen en el reconocimiento y clasificación son las siguientes:

- Se asume que el fenómeno tiene estructura.
- Ruido experimental (resultado de la medición del fenómeno).

- Capacidades de expresar el problema como un modelo abstracto en un esquema de representación.
- Existencia de variables externas.
- Estados o situaciones del fenómeno no contemplados.

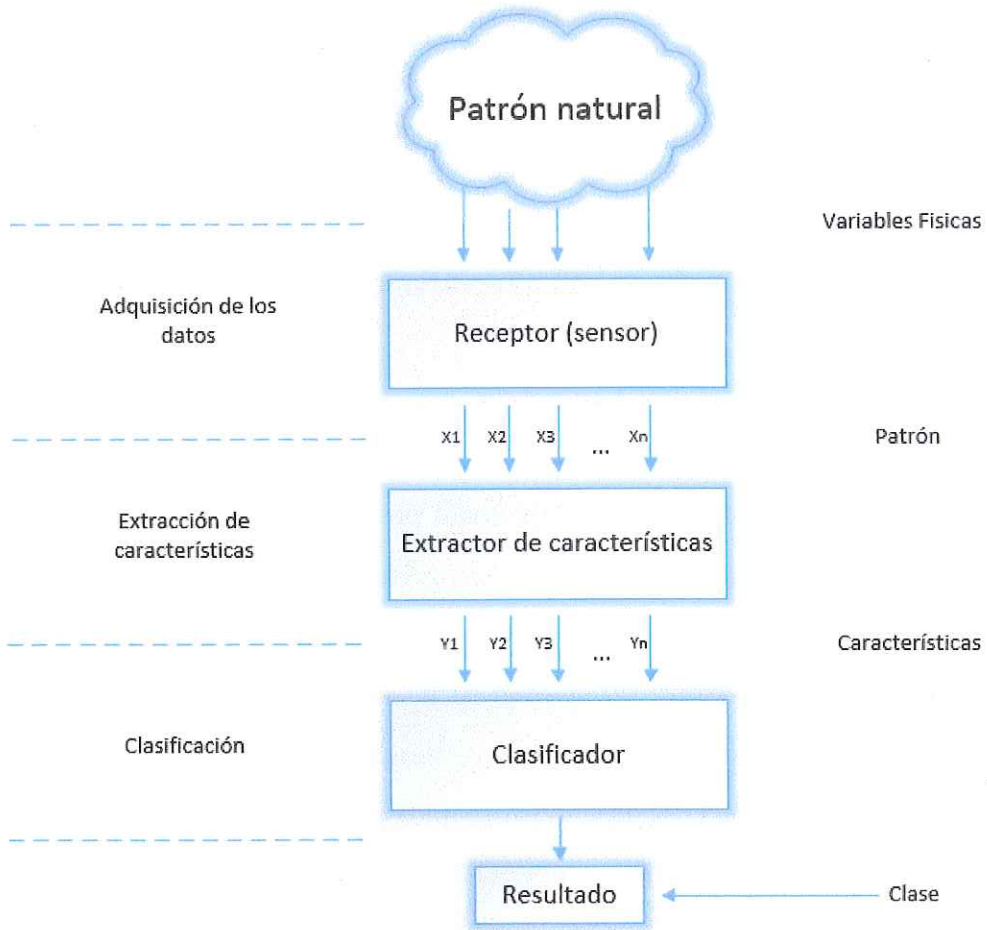


Figura 3.3: Esquema general de clasificador.

Estas situaciones en conjunto, permiten que el fenómeno a caracterizar tenga un número grande de situaciones distintas (grados de libertad) en donde los atributos necesarios para su análisis resulten de difícil obtención, sean insuficientes o no sean los adecuados para los fines que se presenta. Contar con modelos y técnicas con un costo computacional bajo, ayuda a dar soluciones a problemas de reconocimiento y clasificación.

3.3.1. Reconocimiento de caracteres

Un contexto particular del reconocimiento de patrones es el reconocimiento de caracteres en textos. El objetivo fundamental es extraer la información necesaria en forma de atributos que permita identificar y clasificar cada carácter que pueda estar presente en un documento escrito. Al símbolo o dibujo que denota a un carácter en particular se le llama grafía. El problema que se presenta al hacer un análisis sobre textos es que la forma, tamaño y estilos de las grafías no son uniformes y depende del tipo de documento que se desea analizar. Dentro del reconocimiento de caracteres se pueden hacer dos distinciones, de acuerdo al origen del escrito:

1. Caracteres generados de moldes artificiales (impresos).

2. Caracteres manuscritos (generadas por humanos).

Para este trabajo sólo se analiza el caso de caracteres generados por moldes artificiales. La variedad de las grafías depende de factores tales como: escala, rotación y texturizado dentro de un texto. Contar con un número definido de descriptores sobre las grafías no es posible debido a la gran variedad de situaciones en que una sola grafía puede presentarse. Por ejemplo, en el caso de la carácter "h"se pueden dar varios ejemplos de diferentes estilos de caracteres, algunos de estos se muestran en la Figura 3.4. El problema de identificación incluye la cantidad de variantes y el contexto. Por ejemplo cualquier grafía puede aparecer en contextos de difícil identificación, cuando esta presenta textura o entramados que hace complicado determinar los bordes de la grafía.

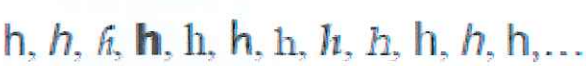


Figura 3.4: Varios tipos de "h"

Buscar métodos que ayuden a encontrar información que sea significativa para describir una grafía, combinado con métodos teóricos de clasificación, permite establecer las bases suficientes para un reconocedor de caracteres. En la literatura especializada existen varios

métodos que dan soluciones particulares a problemas que están acotados en un contexto definido, en la Figura 3.5 se muestran algunos métodos tradicionales para reconocer caracteres. Cada método ofrece buenos resultados en contextos particulares, pero ninguno ofrece soluciones generales. La capacidad de cómputo necesario varía en función de cada método.

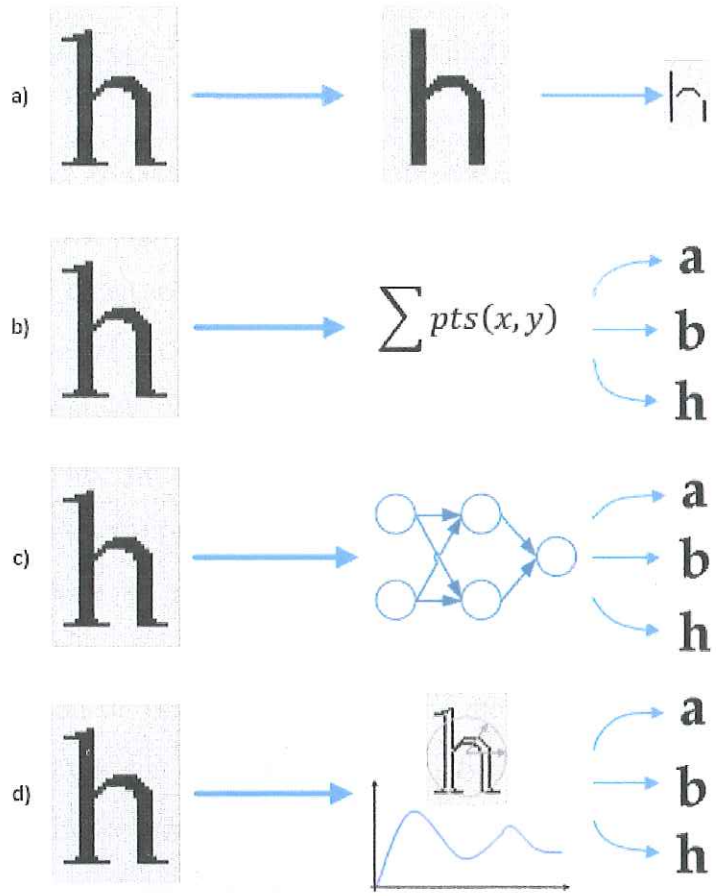


Figura 3.5: Diferentes maneras de realizar el reconocimiento: a) por descomposición de formas primitivas, líneas, y curvas; b) por áreas del carácter ocupado; c) utilizando redes neuronales; d) Gráficas de distancia a partir del centro de la imagen.

De los métodos descritos en la literatura, la mayoría de ellos caracterizan a un carácter por medio de la imagen de este, representada como un conjunto de datos y caracterizarlos por algún método de codificación en un conjunto de descriptores numéricos, que sirve de entrada

a algún método de clasificación. Este método usualmente es una transformación numérica que bajo ciertos umbrales indica la clase (letra) a la que es coincidente la entrada bajo un esquema de tolerancia.

Proponer maneras diferentes de resolver el problema, permite tener alternativas con un costo computacional distinto que ayuden a dar soluciones adecuadas a ciertos contextos. En este trabajo se muestra cómo, al emplear diferentes estructuras para codificar la información, y las diferentes maneras de representarla, se pueden construir maneras distintas con un costo computacional bajo.

Los descriptores, al ser empleados sobre características particulares cuantifican, bajo un esquema de codificación, los estados que presentan las grafías de los caracteres, sin embargo, la diversidad de situaciones de una grafía no esta limitada a un número definido, y no se puede saber *apriori*, qué descriptores son los adecuados para abstraer todas aquellas características que permitan identificar a cada carácter. El empleo de modelos simbólicos basados en dos estados permite codificar la información como cadenas simbólicas binarias[4], donde la estructura de las mismas, bajo algún modelo de codificación representa características particulares de los caracteres. Si la información codificada en cadenas es la necesaria por algún medio de codificación para representar a algún carácter, entonces la cadena es la abstracción de este carácter. Emplear modelos aleatorios ayuda a encontrar soluciones a problemas que son difíciles de resolver, dando soluciones tentativas subóptimas que en contextos definidos son muy útiles. Unir dos modelos de estas características proporciona las bases suficientes para proponer procesos distintos para el reconocimiento de caracteres. En el resto del capítulo se muestran algunos modelos teóricos de decisión, analizando en particular las memorias asociativas, de donde se toman algunos conceptos que ayudan a desarrollar un proceso automático para reconocer caracteres empleando cadenas simbólicas.

3.4. Memorias Asociativas

Las Memorias Asociativas (MA) son un modelo abstracto computacional, que toma de metáfora el concepto de asociación de conceptos en categorías. El funcionamiento de las MA's consiste en que dados dos conjuntos A y B, se asocian los elementos del conjunto A, con los elementos del conjunto B. En general se habla de dos conjuntos A y B, pero las asociaciones pueden hacerse sobre un mismo conjunto. La manera de asociar estos conjuntos es por medio de un conjunto de operaciones con los elementos de A y los elementos de B sobre una matriz denotada por M, estos conjuntos pueden ser de dominio real o entero. En la Figura 3.6 se observa un esquema de una Memoria Asociativa Clásica general.

Figura 3.6: Esquema de una memoria asociativa.

Formalmente un elemento de A se denota por $a \in A$, y un elemento de B se denota por $b \in B$, entonces las asociaciones son denotadas por un conjunto donde cada elemento es una 2 tupla de la forma (a_i, b_i), donde i representa la asociación *i-ésima*. El conjunto de relaciones de asociaciones se denota por R. Una MA esta constituida por una matriz M, que contiene en ella (bajo algún conjunto de operaciones) codificadas las asociaciones en R, de modo que cuando se opera sobre M, se puede distinguir si el elemento ha sido codificado o no en M. Las MA, se clasifican en dos grupos:

- Heteroasociativa: Se dice que una MA es heteroasociativa si para el conjunto de asociaciones R, de los conjuntos A y B, para alguna asociación (a_i, b_i) contenida en R se tiene que $a_i \neq b_i$, donde $a_i \in A$ $b_i \in B$.

- Autoasociativas: Se dice que una MA es autoasociativa si para el conjunto de asociaciones R, se tiene que $A = B$ y para toda relación (a_i, b_i) contenida en R donde $a_i \in A$ $b_i \in B$ se tiene que $a_i = b_i$.

Las MA, son empleadas como clasificadores universales, porque se puede ver al conjunto A, como patrones de entrada, asignados a una clase, dentro del conjunto B, entonces, es posible tener subconjuntos de A, que están asociados a algún elemento en B, donde B es una agrupación que denota al conjunto A. Los conjuntos A y B usualmente son vistos como algún producto n dimensional del espacio $\{0, 1\}$, entonces los elementos que están asociados son vectores que contienen 0 y 1, algunos modelos expanden a conjuntos mayores como los naturales, o continuos como los reales. La dimensionalidad de A y B se denota por n y m respectivamente. Cada elemento de entrada (del conjunto A) es un vector n dimensional, que está asociado por operaciones sobre la matriz M, al elemento m dimensional (conjunto B).
Una de las principales ventajas de una MA es que tiene tolerancia a variaciones sobre cada componente de los vectores empleados de entrada, es decir, tiene tolerancias a ruido aditivo o sustractivo y esto queda expresado como:
Si se tiene la MA denotada por M y $(a_i, b_i) \in \mathbb{R}$, y se tiene $a'_i \in A$, que denota un versión alterada por ruido (puede ser sustractivo o aditivo) de a_i, al operar sobre a'_i en M, el resultado obtenido es b_i. Es necesario notar que el nivel de intensidad de ruido que presente a'_i, que pueda tolerar M, depende de como fue introducido el vector a_i en M.

3.4.1. Dispositivos basados en una CPU.

La mayoría de las CPU's, y la mayoría de los dispositivos de lógica secuencial, son de naturaleza síncrona. Es decir, están diseñados y operan en función de una señal de sincronización. Esta señal, conocida como señal de reloj, usualmente toma la forma de una onda cuadrada periódica. Calculando el tiempo máximo en que las señales eléctricas pueden moverse en las varias bifurcaciones de los muchos circuitos de una CPU, los diseñadores pueden seleccionar un período apropiado para la señal del reloj.
Este período debe ser más largo que la cantidad de tiempo que toma a una señal moverse, o propagarse en el peor de los casos. Al fijar el período del reloj a un valor bastante mayor sobre el retardo de la propagación del peor caso, es posible diseñar todo la CPU y la manera que mueve los datos alrededor de los francos de subida y bajada de la señal del reloj. Esto

tiene la ventaja de simplificar la CPU significativamente, tanto en una perspectiva de diseño, como en una perspectiva de cantidad de componentes. Sin embargo, esto también tiene la desventaja que todo la CPU debe esperar por sus elementos más lentos, aún cuando algunas unidades de la misma son mucho más rápidas. Esta limitación ha sido compensada en gran parte por varios métodos de aumentar el paralelismo de la CPU.

3.5. Aspectos para implementación en Hardware

En esta sección se describen diferentes aproximaciones a nivel Hardware de implementación de algoritmos. Cada aproximación emplea en forma distinta los recursos y por consecuencia el algoritmo, para que resulte eficiente debe de considerar diferentes aspectos.

- **FPGA(*Field Programmable Gate Array*):** Permite implementar cualquier circuito digital de aplicación específica. Las aplicaciones donde más comúnmente se utilizan con *FPGA* incluyen procesamiento digital de señales, sistemas aeroespaciales y de defensa, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, prototipos de *ASICs* entre otras. Cabe notar que su uso en otras áreas es cada vez mayor, sobre todo en aquellas aplicaciones que requieren un alto grado de paralelismo. Se estima que un *FPGA* supera 500 veces o más el rendimiento de un *DSP*, por lo tanto son usados en sistemas en tiempo real o sistemas que requieren una alta velocidad de procesamiento.

- ***Look-up Table* (*LUT*):** En esencia es una memoria *RAM* con valores predefinido.

- **Celdas Lógicas**: La unidad más pequeña de un FPGA es la celda lógica (*Xilinx*) o elemento lógico (Altera). Una celda lógica contiene principalmente una *LUT* de 4 entradas (la cual se puede usar como una *RAM* de 16x1, o un registro de corrimiento de 16 bits), un multicanalizador y un registro como se muestra en la Figura 3.8 El registro se puede configurar como *flip-flop* (activo por flanco) o como *latch* (activo

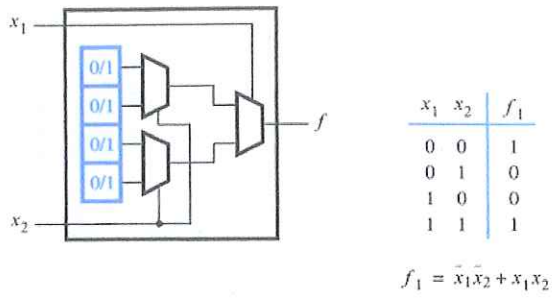


Figura 3.7: Ejemplo de LUT

por nivel). Se pueden configurar las polaridades del reloj (*Clock*), habilitación (*Clock enable*) y señales *set/reset*.

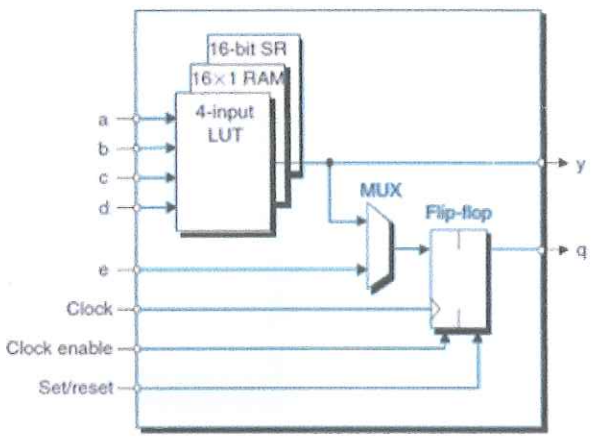


Figura 3.8: Celda lógica o elemento lógico

- **Slices**: En esencia es un bloque de *look-up tables* enlazadas con una salida que puede ser registrada o directa, pudiendo implementarse diversas funciones lógicas.

- **Arquitectura de un *FPGA***: Está formada por una matriz de bloques lógicos configurables (*CLB*),a su vez cada *CLB* esta formado por *Slices* y cada *Slice* esta formado por Celdas Lógicas *(Logic Cells)*. Las FPGAs se utilizan cuando se necesita procesamiento muy rápido y paralelo.

- **Sumadores, Multiplicadores y *MAC* Embebidos**: Los circuitos sumadores y multiplicadores son lentos ya que generan largos retrasos de propagación al conectar

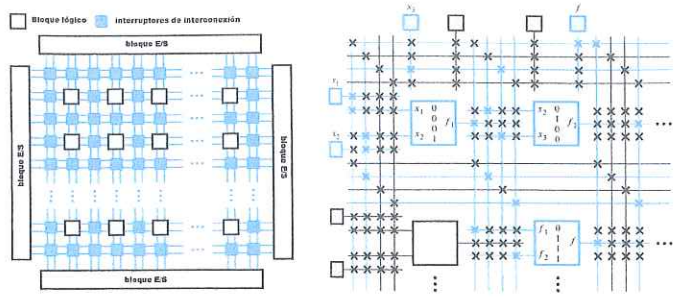


Figura 3.9: Arquitectura general y seccion de FPGA

un gran número de CLBs juntos. Debido a esto y a que son operaciones muy comunes, muchos *FPGAs* incorporan bloques sumadores y multiplicadores. Estos se encuentran comúnmente localizados cerca de la memoria *RAM* embebida debido a que comúnmente se usa en conjunto con esta.

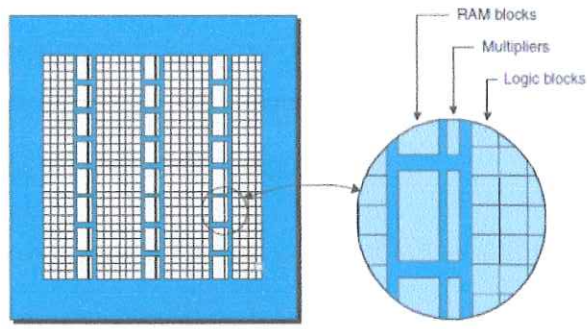


Figura 3.10: Multiplicadores embebidos

Una operación muy común en las aplicaciones de procesamiento digital de señales (*DSP*) es llamada Multiplica y acumula (*MAC*). Como su nombre indica, esta función multiplica dos números y luego suma el resultado a un acumulador como en la Figura. 3.10. Esta operación es usada para el cálculo de la convolución, *FFT* (transformada rápida de *Fourier*), *FWT* (transformada rápida de *Wavelet*), etc.

3.5.1. Programación concurrente

En diversos procesos computacionales es necesario aplicar diversos paradigmas de programación para enfocar el problema, es así que en algunas aplicaciones se necesita

aplicar concurrencia y paralelismos para aumentar la eficiencia y potencial de los programas. En este contexto se va a desarrollar los principales conceptos de la programación concurrente y paralela, así como una introducción a la comunicación entre procesos y teoría de hilos.

- **Computación concurrente:** Es la simultaneidad en la ejecución de múltiples tareas en interacción. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa [30]. La programación concurrente está relacionada con la programación paralela, pero enfatiza más la interacción entre tareas. Así, la correcta secuencia de interacciones o comunicaciones entre los procesos y el acceso coordinado de recursos que se comparten por todos los procesos o tareas son las claves de esta disciplina.

- **Programación secuencial:** Es aquel que especifica la ejecución de una secuencia de instrucciones que comprenden el programa. Los procesos secuenciales se han usado tradicionalmente para estudiar la concurrencia.

- **Programación concurrente:** Es un programa diseñado para tener 2 o más contextos de ejecución decimos que este tipo de programa es multi-hilo, porque tiene más de un contexto de ejecución. Un programa concurrente es un programa que tiene más de una línea lógica de ejecución, es decir, es un programa que parece que varias partes del mismo se ejecutan simultáneamente.

- **Programación paralelo**: Es un programa concurrente en el que hay más de un contexto de ejecución o hebra activo simultáneamente; desde un punto de vista semántica no hay diferencia entre un programa paralelo y concurrente. Algunas características de la programación paralela son:

  - Resolver problemas que no caben en una CPU.
  - Resolver problemas que no se resuelven en un tiempo razonable.
  - Se pueden ejecutar problemas mayores en forma más rápida

- El rendimiento de los computadores secuenciales está comenzando a saturarse, una posible solución sería usar varios procesadores, sistemas paralelos como *FPGA's*

3.5.2. *VHDL*

VHDL es el acrónimo que representa la combinación de *VHSIC* (*Very High speed Integrated Circuit*) y *HDL* (*Hardware description language*). Originalmente *VHDL* fue patrocinado por el Departamento de Defensa de E.U. y posteriormente transferido al *IEEE* (*Institute of Electrical and Electronics Engineers*). El *IEEE* lo ratificó como el estándar 1076 en 1987, al cual se le llama *VHDL-87*. En 1993 el *IEEE* hizo una nueva revisión del estándar *(VHDL-93)* y en el 2001 *(VHDL-2001)*. El *VHDL* originalmente surgió como un lenguaje para simulación de circuitos digitales y para el diseño se usaban otras herramientas como esquemáticos y *Netlist*. Conforme los circuitos digitales fueron haciéndose más complejos surgió la necesidad de poder describir los circuitos con un alto grado de abstracción, no desde el punto de vista estructural, sino desde el punto de vista funcional. Este nivel de abstracción ya se había alcanzado con las herramientas de simulación (como *VHDL*), ya que para poder simular partes de un circuito era necesario disponer de un modelo que describiera el comportamiento del circuito o sus componentes. Fue entonces cuando se empezó a usar el *VHDL* para el diseño de circuitos digitales, ya que surgieron herramientas que realizan la síntesis a partir de la descripción en *HDL*. El lenguaje *VHDL* es un lenguaje muy extenso y complejo, sin embargo para la síntesis solo se una pequeña porción del lenguaje.

Capítulo 4

PROPUESTA.

En este capítulo se describe los métodos que se utilizan para el desarrollo del proyecto, se divide en tres etapas:

- Modelo de aleatorio de imágenes.
- La creación matriz de aprendizaje.
- Generación de la arquitectura en FPGA.

4.1. Modelo aleatorio de imágenes

Dependiendo de la manera de representar, codificar y operar la información de algún fenómeno es necesario contar con un método para extraer y evaluar los datos en una imagen. El método de muestreo probabilístico es un método adecuado para la selección de elementos de una población obteniendo la información relevante para trabajar con una fracción de la información original.

El reconocimiento de caracteres es un proceso de extracción de descriptores útiles para discriminar cada elemento. La información de un carácter se representa con un total de puntos de la imagen. El método de Muestreo tiene como objetivo extraer puntos que ayuden a discriminar codificando en una cadena simbólica con la información. Cuando la información contenida en la imagen es considerablemente grande se usa un muestro aleatorio probabilístico obteniendo una información más compacta de los datos.

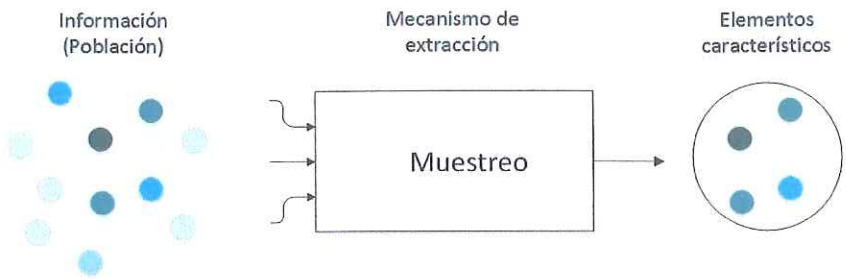


Figura 4.1: Proceso de muestreo aleatorio

4.1.1. Muestreo aleatorio

Para realizar el método de muestreo aleatorio se considera tomar muestras de la imagen donde se encuentra el carácter y codificarla en cadenas binarias. Si el muestro resulta representativo, este representa de manera más compacta los datos (en este caso información de los caracteres). El nivel y calidad de la abstracción dependerá de la calidad de la muestra seleccionada. En la Figura 4.1 a través de un mecanismo de muestreo se seleccionan los elementos característicos de la población. La extracción de información de las diferentes grafías en las imágenes en el reconocimiento de caracteres puede considerarse como un problema de muestreo, el objetivo es extraer aquellos puntos que conforman la imagen de los caracteres de manera que la identifique. Un muestreo en reconocimiento de caracteres es un subconjunto de puntos de una imagen que contiene la información necesaria para representar un carácter, o, en forma general, a un conjunto de caracteres, entonces un muestreo adecuado permite que no se use la totalidad de la información de cada variante de carácter, si no una fracción representativa. El muestreo de una imagen obtiene un subconjunto de puntos que representa un patrón de este carácter. En el caso que se tengan diferentes variantes de los caracteres al aplicar el mismo muestreo de puntos, la información será muy parecida.

Entonces para este trabajo, el reconocimiento de caracteres consiste en extraer la información por medio de un muestreo que caracterice a cada símbolo del alfabeto escrito como se muestra en la Figura 4.2 y un método de clasificación que identifique a cada carácter de acuerdo a las características que fueron codificadas por el muestreo.

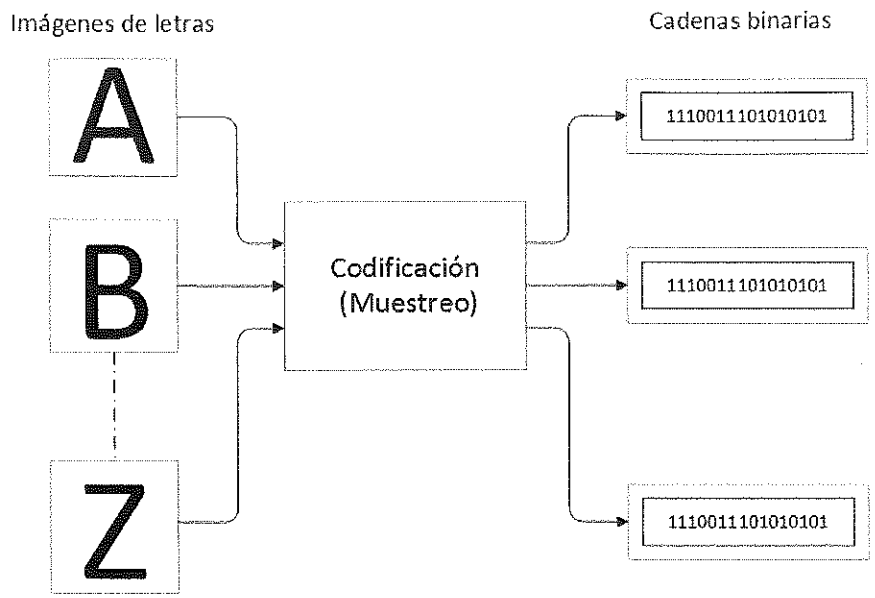


Figura 4.2: Diagrama del proceso

4.1.2. Representación.

Un modelo de representación simbólica de dos estados ayuda a codificar problemas específicos. Estos contienen atributos de cada carácter. Las cadenas simbólicas permiten tener una representación en pocos estados de cada carácter, no es necesario definir los descriptores gracias a la técnica de muestreo la cual reduce la información. Como resultando nos entrega una manera más compacta para expresar los datos en situaciones de una excesiva cantidad de información.

Esta representación simbólica se expresa con nociones de ausencia o existencia de información [4], es decir, en un carácter dado, indica el conjunto de píxeles que coinciden con la posición de los puntos generados por el muestreo aleatorio, de manera que esta distribución de píxeles permite identificar a dicho carácter como se muestra en la Figura. 4.3.

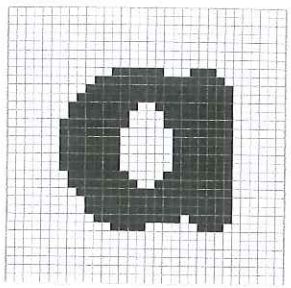


Figura 4.3: Representación binaria de un carácter

4.1.3. Codificación

La codificación es la forma de caracterizar el conjunto de información empleando cadenas binarias, es una manera extraer información que representaran una población representativa del carácter. La codificación consiste en generar un conjunto de puntos aleatorios dentro de los rangos de la imagen binarizada.

Cada imagen asociada a un carácter se representa como una matriz M, tal que para todo elemento $M_{ij} \in \mathbb{R}$, el valor del punto j,i se representa por el contenido de la posición M_{ij}. Sea n el número de puntos que se desean muestrear de la matriz M, entonces es necesario generar $2n$ puntos aleatorios para conformar las n posiciones a muestrear dentro de la matriz M de la forma (i, j) en el rango. Este conjunto de puntos es almacenado en una lista L en el orden en que son generados. Para cada tupla contenida (i, j) en L, se obtiene el valor de la matriz M, aplicando la función de transformación, el resultado es concatenado con el resto de los símbolos de los puntos. Se representa la imagen de un carácter que tiene asociada una

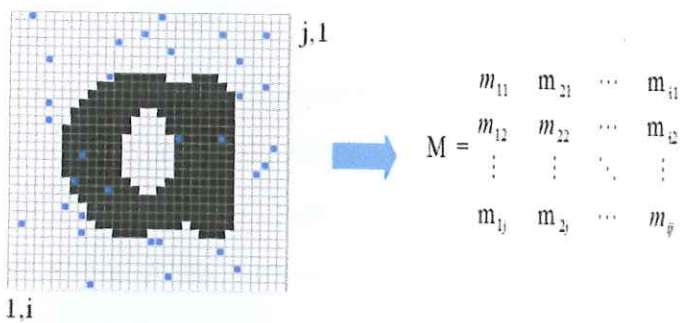


Figura 4.4: Representación binaria de un carácter

matriz M_{ij}. Las cadenas generadas a partir de M, representan a su vez un punto p en un espacio $\{0,1\}^n$, donde n es la longitud de la cadena, es decir, el tamaño de la muestra como se muestra en la Figura 4.4.

Tomando como población inicial el conjunto de posiciones de la matriz M, que corresponden a los distintos puntos de la imagen, se escogen aleatoriamente un conjunto de n puntos de muestreo en M, de manera que bajo el orden en que son seleccionados, se construye una cadena de longitud n

En el caso de la Figura 4.5, $f(x)$ representa la función de transformación, que codifica un punto de la imagen en un símbolo. Si se emplean los mismos puntos de muestreo, se observa que ante pequeñas variaciones sobre el contenido de la imagen, la cadena simbólica generada sufre cambios mínimos con esta codificación.

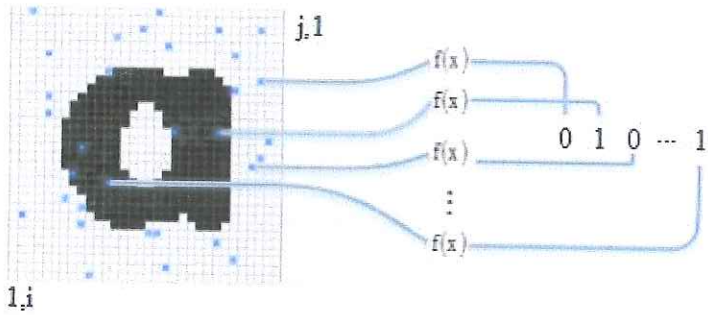


Figura 4.5: Proceso de generación de la cadena

4.1.4. La regla del mayor

También conocida como *Majority rule*, consiste en encontrar un punto p de un conjunto C a partir de la frecuencia en que ocurre alguno de los dos símbolos del modelo de representación. Para esto se toma cada elemento de C y se calcula la frecuencia de cada símbolo $C \in \{0,1\}$ obteniendo un nuevo vector de las mismas dimensiones que C. Aplicando una codificación de dos estados, donde para valores mayores a $|C|/2$ se asigna "1", caso contrario "0çomo se muestra en la Figura. 4.6.

Sea un conjunto $C \subseteq \{0,1\}^n$ y $m_y \in \{0,1\}$, donde el punto significativo determinado por

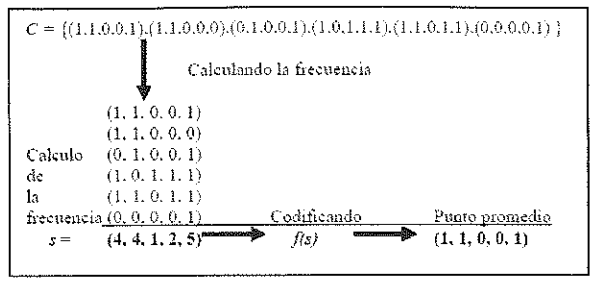


Figura 4.6: Ejemplo de Majority Rule

Majority Rule esta definido como:

$$\forall_{i\in 1\ldots n} my_i = \begin{cases} 1 & si \quad \sum_{\forall_{c\in C}} c_i > \frac{|C|}{2} \\ 0 & si \quad \sum_{\forall_{c\in C}} c_i \leq \frac{|C|}{2} \end{cases} \tag{4.1}$$

Como conclusión del muestreo aleatorio tenemos que existen ventajas al emplear un modelo de muestreo aleatorio y una codificación simbólica, ya que no es necesario tener la totalidad de información lo que nos permite reducir procesamiento y al ser una codificación simbólica binaria por consecuencia nos permite trabajar con la lógica digital. La caracterización de la información como una sola cadena, permite representar a un carácter como una cadena, en donde de acuerdo a los criterios de codificación puede representar a la imagen completa sin la necesidad de tener un conjunto de descriptores de características especificas sobre el carácter que se esta codificando, ofreciendo alternativas distintas.

4.2. Lernmatrix

Es una memoria heteroasociativa que puede funcionar como un clasificador de patrones binarios si se escogen adecuadamente los patrones de salida; es un sistema de entrada y salida que al operar acepta como entrada un patrón binario $x^\mu \in A^n, A = 0, 1$ y produce como salida la clase $y^\mu \in A^p$ que le corresponde (de entre p clases diferentes), codificada ésta con un método simple, a saber : para representar la clase $k \in 1, 2, \ldots, p$, se asignan a las componentes del vector de salida y^μ los siguientes valores: $y_k^\mu = 1 y y_j^\mu = 0$, y para

$j = 1, 2, \ldots, k-1, k+1, \ldots p.$

4.2.1. Fase de entrenamiento

Esquema de la fase de aprendizaje al incorporar la pareja de patrones de entrenamiento $(x^\mu, y^\mu) \in A^n \times A^p$.

Opera con entrada binaria tal que

$$x^\mu \in A^n, A = \{0, 1\} \tag{4.2}$$

Y una salida que corresponde a un vector de clases constituido con la siguiente ecuación.

$$y^\mu \in A^p \tag{4.3}$$

La memoria asociativa M se representa mediante una matriz cuya componente ij-ésima es m_{ij}; la matriz M se genera a partir de un conjunto finito de asociaciones conocidas de antemano: éste es el conjunto fundamental de asociaciones, o simplemente conjunto fundamental. Se denota por p la cardinalidad del conjunto fundamental (p es un número entero positivo). Si es un índice, el conjunto fundamental se representa de la siguiente manera:

$$\{(x^\mu, y^\mu) | \mu = 1, 2, \ldots, p\} \tag{4.4}$$

Cada uno de los componentes m_{ij} de M, la *Lernmatrix de Steinbuch*, tiene valor cero al inicio, y se actualiza de acuerdo con la regla $m_{ij} + \Delta m_{ij}$, donde:

$$\nabla m_{ij} = \begin{cases} +\varepsilon \quad si \quad y_i^\mu = 1 = x_j^\mu \\ -\varepsilon \quad si \quad y_i^\mu = 1 \quad y \quad x_j^\mu = 0 \\ 0 \; en \; otro \; caso \end{cases} \tag{4.5}$$

En la tabla se esquematiza la fase de aprendizaje para la Lernmatrix de Steinbuch, con la pareja de patrones fundamentales.

	x_1^μ	x_2^μ	...	x_j^μ	...	...	x_n^μ
y_1^μ	m_{11}	m_{12}		m_{12}			m_{1n}
y_2^μ	m_{21}	m_{22}		m_{2j}			m_{2n}
y_i^μ	m_{1i}	m_{i2}		m_{ij}			m_{in}
y_m^μ	m_{m1}	m_{m2}		m_{mj}			m_{mn}

4.2.2. Fase de recuperación.

La fase de recuperación consiste en encontrar la clase a la que pertenece un vector de entrada $x_\omega \in A^n$ dado. Encontrar la clase significa obtener las coordenadas del vector $y_\omega \in A^p$ que le corresponde al patrón x_ω; en virtud del método de construcción de los vectores y^μ la clase debería obtenerse sin ambigüedad. La i-ésima coordenada yl_ω i del vector de clase $y_\omega \in A^p$ se obtiene como lo indica la siguiente expresión:

$$x^\omega \in A^m \tag{4.6}$$

Esto significa encontrar las coordenadas correspondientes:

$$y^\omega \in A^m \tag{4.7}$$

Para recuperar la información se efectúa la siguiente operación:

$$y^\omega = M \bullet x^\omega \tag{4.8}$$

Se obtiene con la siguiente expresión donde U es un operador de máximo.

$$y_i^\omega = \begin{cases} 1 \quad si \quad \sum_{j=1}^n m_{ij} x_j^\omega = U_{j=1}^m \left[\sum_{j=1}^n m_{hj} x_j^\omega \right] \\ 0 \text{ en otro caso} \end{cases} \tag{4.9}$$

La forma en como Steinbuch planteó las fases de aprendizaje y recuperación para su memoria son poco útiles para la descripción de su funcionamiento. De esta forma, lo primero que se plantea es una caracterización alterna de dichas fases, de modo que su

manipulación matemática sea más fácil. Dicha caracterización introduce el concepto de función de Steinbuch.

Llamaremos función de Steinbuch a una función $f : \mathbb{R} \longrightarrow \mathbb{R}$ que cumpla con la siguiente propiedad:

$$\begin{aligned} f(0) &= -1 \\ f(1) &= 1 \end{aligned} \tag{4.10}$$

Llamaremos función vectorial de Steinbuch con respecto a f a una función $F : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, tal que:

$$F(x) = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \cdots \\ f(x_n) \end{pmatrix} \tag{4.11}$$

Así, se está en condiciones de proponer una caracterización alterna para las fases de aprendizaje y recuperación que tiene una interpretación equivalente que se adapta mejor a la propuesta.

Siendo como en (4.4) un conjunto fundamental y F una función vectorial de Steinbuch con respecto a f . La Lernmatrix M para el conjunto fundamental se construye de acuerdo con la siguiente regla:

$$M = \sum_{\mu=1}^{m} y^{\mu} \bullet (F(x^{\mu}))^T \tag{4.12}$$

Sea M una Lernmatrix y x^{ω} un patrón de dimensión n. El patrón $y^{\sim\omega}$ recuperado a partir de x^{ω} y M se determina de la siguiente forma:

$$y^{\omega} = M \bullet x^{\omega} \tag{4.13}$$

Donde:

$$y_i^{\omega} = \begin{cases} 1 \quad si \quad z_i^{\omega} = U_{h=1}^{m} z_h^{\omega} \\ 0 \text{ en otro caso} \end{cases} \tag{4.14}$$

Donde $y^{\sim\omega}$ no es necesariamente igual a y^{ω}. En particular, si$y^{\sim\omega} = y^{\omega}$ es una recuperación perfecta.

4.3. Arquitectura en FPGA

En esta se realiza el clasificador aprovechando las cualidades que tiene este dispositivo reconfigurable. Se basa en una composición de componentes digitales. En la fase de adquisición se diseña un *DEMUX* (Demulticanalizador, selector de múltiples salidas con lógica digital como se aprecia en la Figura 4.7) para poder intercambiar la información de que se introduce. Esta causa que la cadena seleccione entre ser un patrón fundamental a un vector de prueba.

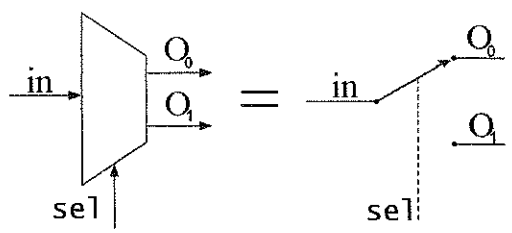


Figura 4.7: Estructura de un demulticanalizador

En cuanto la fase de aprendizaje de *Steinbuch* se considera la salida del DEMUX cuando el selector esta en bajo, entonces tomando en cuenta el criterio del complemento *A2* y el de *Steinbuch* se hace una asignación según la información que llega de la cadena y se guarda en una matriz (M) como en la ec. (4.15).

$$M = \begin{array}{c|ccccccc} & x^{1\mu} & x^{2\mu} & \cdots & x^{j\mu} & \cdots & x^{n\mu} \\ y^{1\mu} & m_{11} & m_{12} & \cdots & m_{1j} & \cdots & m_{1n} \\ y^{2\mu} & m_{12} & m_{22} & \cdots & m_{2j} & \cdots & m_{2n} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{i\mu} & m_{i1} & m_{i2} & \cdots & m_{ij} & \cdots & m_{in} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ y^{p\mu} & m_{p1} & m_{p2} & \cdots & m_{pj} & \cdots & m_{pn} \end{array} \tag{4.15}$$

Dicha matriz esta constituida por el número de clases contenidas en el modelo además de la resolución de la cadena binaria y es almacenada en una memoria temporal de la FPGA la

matriz M donde se hace la fase de consulta. La consulta esta dividida en secciones, en una de ellas se puede apreciar que la suma y multiplicación se realizan en el mismo ciclo.

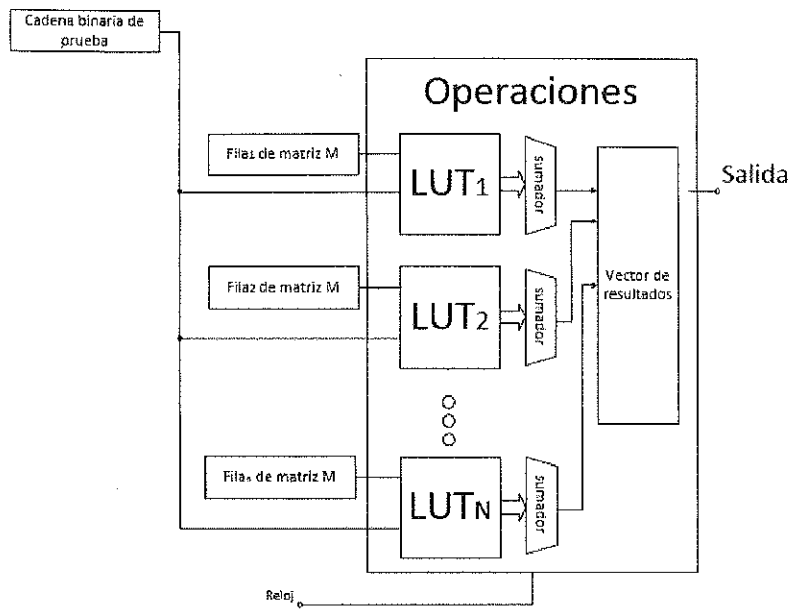


Figura 4.8: Proceso de consulta en FPGA

Se tiene un detector de máximo haciendo comparaciones sucesivas en parejas de números hasta encontrar la solución.

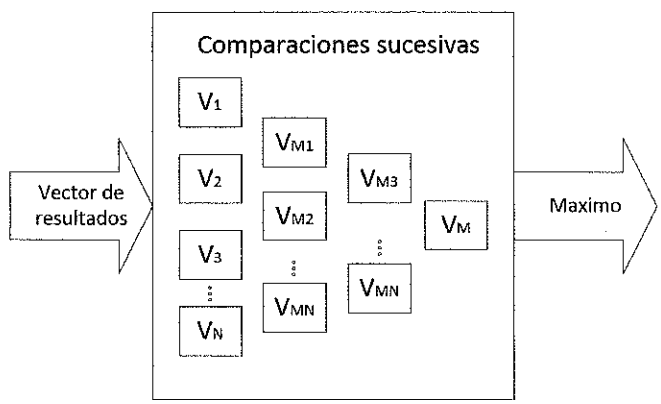


Figura 4.9: Diagrama de comparaciones sucesivas

Por ultimo se tiene la integración de todas estas secciones antes mencionadas como se muestra en la Figura. 4.10, cabe mencionar que la resolución es una aspecto importante considerar

para la síntesis por motivo que el desborde de estos variables puede causar lecturas erróneas en los resultados del clasificador. Cabe mencionar que se uso la matriz de aprendizaje de Steinbuch ya que es la base de posteriores memorias que se han desarrollado a lo largo del tiempo, también tiene características que favorecieron el desarrollo en Hardware como son que utiliza cadenas binarias para su funcionamiento, otro aspecto que fue considerado fue su linealidad.

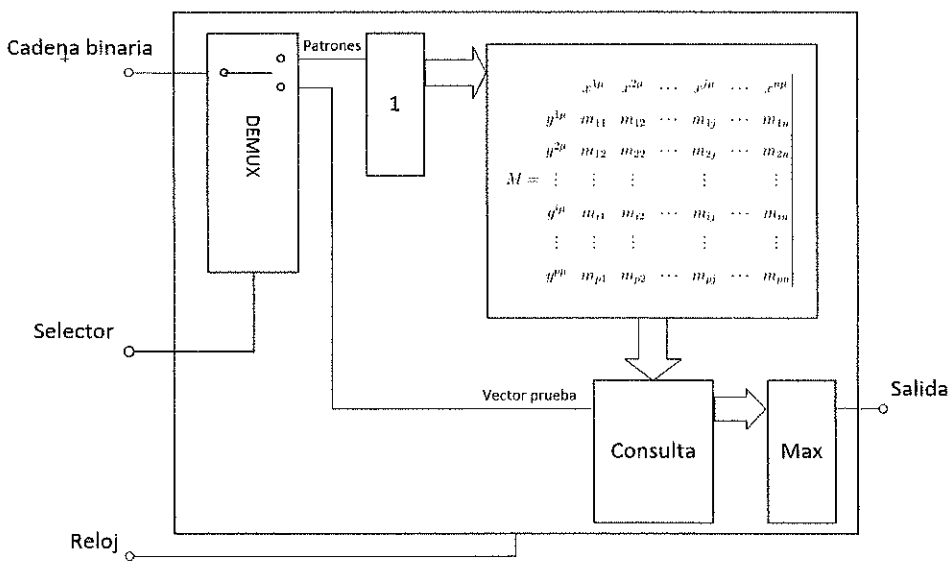


Figura 4.10: Arquitectura completa en FPGA

Capítulo 5

Modelo Experimental y resultados

Al emplear maneras distintas de representar y codificar la información se pueden construir métodos diferentes para resolver problemas particulares. En el capítulo anterior se mostró cómo combinando un modelo de representación simbólico y un modelo de muestreo se pueden establecer las bases suficientes para definir un método de reconocimiento de caracteres basado en la matriz de aprendizaje de Steinbuch; en este capítulo se presenta una propuesta de método para el reconocimiento de caracteres basado en una codificación por un muestreo alcatorio y donde la información se representa como cadenas simbólicas de longitud definida. Para evaluar el método propuesto se ha desarrollado un modelo experimental, donde se mide el porcentaje de reconocimiento, al reconocer un conjunto de diferentes tipos de caracteres.La parte de modelo se realizaron pruebas en un software de implementación en el cual se probo el funcionamiento en la ejecución de un solo hilo. Posteriormente se propone un una versión en hardware donde se muestra las capacidades en cuestión de operaciones paralelas y por ultimo un análisis de reducción de complejidad de reconocedor basado en procesador y el generado en la FPGA utilizando el modelo del peor de los casos polinomial $O(n)$

5.1. Optimización de la codificación

Con el fin de obtener una optimización del método desarrollado en software de un solo hilo de ejecución se aplica un criterio para garantizar la máxima eficiencia. Este consiste en que la matriz generada por la codificación contenga una distribución uniforme de los datos binarios. Se hace un análisis de la distribución de los datos el cual consiste en un muestreo se genera una lista L de longitud n, donde n representa el número de puntos a muestrear de las imágenes, sin embargo, el uso de una distribución aleatoria uniforme no siempre garantiza encontrar los puntos óptimos, se pueden dar casos en que resulten repetidos, o las partes

muestreadas dentro de la imagen no ayuden para discriminar a un carácter. La definición de un método de optimización garantiza, en la mayoría de los casos, que se tenga buenos resultados. La primera parte del proceso consiste en adaptar la distribución de los puntos relevantes. Supóngase que se generan los n puntos, en su mayoría en áreas donde siempre permanece constante el carácter, estos puntos no tienen información directa que ayude en la tarea de reconocimiento. En Fig 5.1 se muestra un patrón correspondiente a la letra "a". El análisis de los patrones ayuda a no manejar el total de los caracteres, los puntos significativos a muestrear dentro de un patrón p.

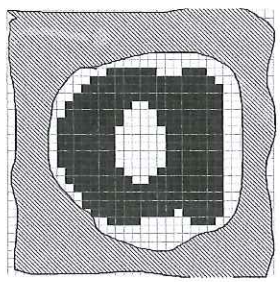


Figura 5.1: Zona que no proporciona directamente información del carácter.

Después se hace una prueba en la cual se suman los puntos de todas la fuentes para ver la zona que nos proporciona más información como se muestra en la Fig
Posteriormente al garantizar las zonas que nos proporcionan la mayor información se procede a hacer un análisis de cada tipo de fuente como se observa en el ejemplo de la Fig. 5.2. En el mismo proceso se implementa el criterio de *Majority rule* para encontrar un carácter promedio que sera introducido en la matriz de aprendizaje.

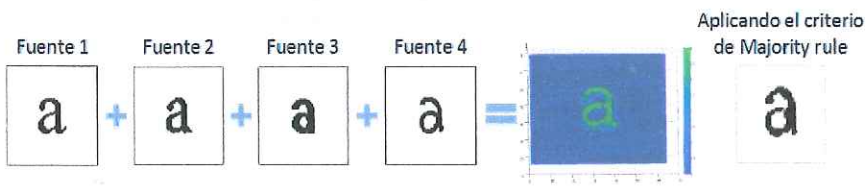
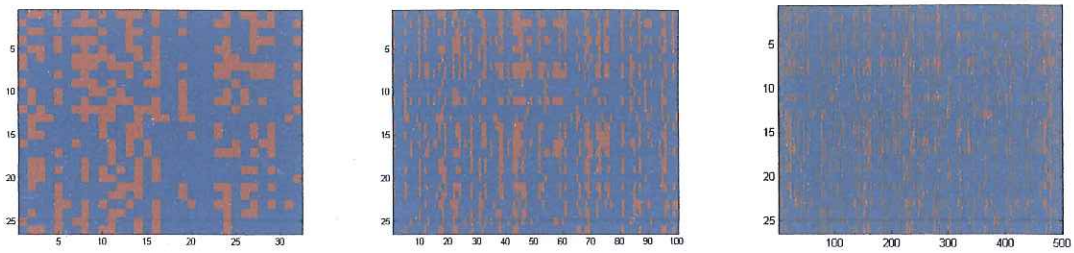


Figura 5.2: Suma de caracteres "a" de distintas fuentes

Dentro de las pruebas que se aplican en el algoritmo para construir la matriz de aprendizaje, se

aplican varias cadenas binarias dentro de un ciclo donde una condición garantiza la longitud adecuada para diferente resolución como se muestra en la Figura 5.3 estas tienen como objetivo encontrar la distribución de puntos que contenía mejores resultados



(a) Con 32 puntos (b) Con 100 puntos (c) Con 500 puntos

Figura 5.3: Experimentación con múltiples puntos

En las siguientes gráficas se puede observar los índices reconocimiento de 80 pruebas que se realizaron en alfabetos que contienen 26 caracteres diferentes como se muestra en la Figura 5.5, se comprobó que el método tiene buenos resultados con una cadena binaria con longitud de 32 que correspondería a el 3 % de los datos de la imagen (Considerando una imagen de 32x32 pixeles).

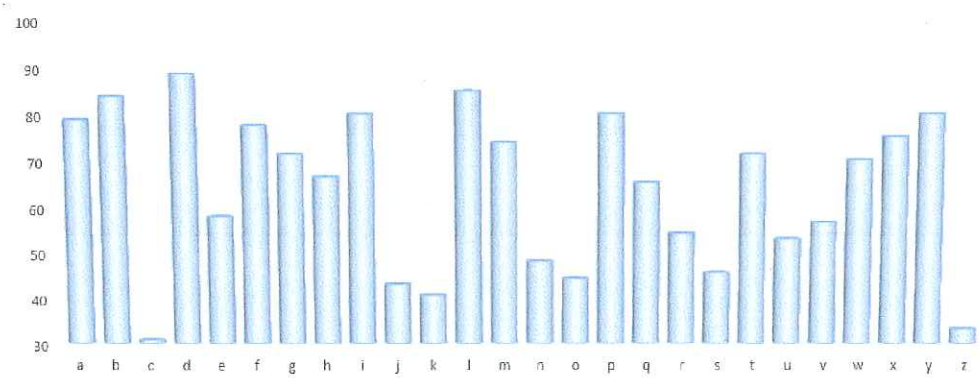


Figura 5.4: Eficiencia con 32 puntos uniformemente distribuidos

Continuando con el análisis ponemos una longitud de 100 puntos aleatorios que corresponden al 9 % de la información, aumenta el índice de reconocimiento de algunos caracteres, en algunos casos el clasificador.

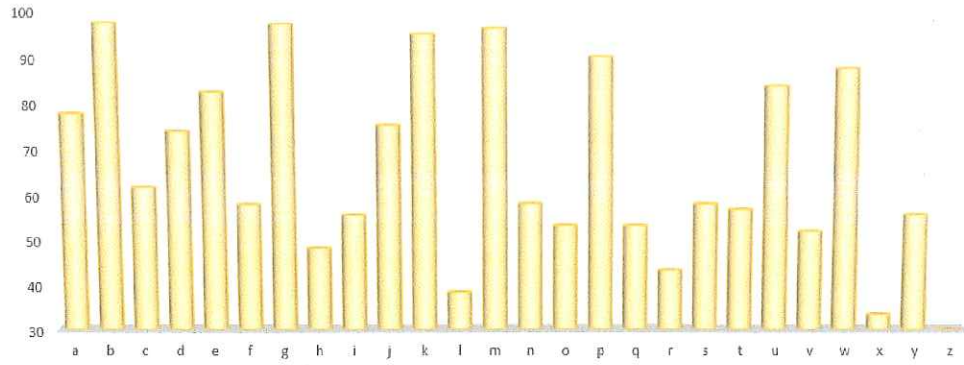


Figura 5.5: Eficiencia con 100 puntos uniformemente distribuidos

Por último se ingresan 500 puntos (50 % de la información) teniendo mejores resultados que los anteriores. Por tanto podemos concluir que si disminuimos el número de puntos podemos tener una aproximación rápida de los caracteres presentes en un texto, esto podría corregirse con el uso de diccionarios de palabras aunque también al ir aumentando el número de puntos se tiene más certeza.

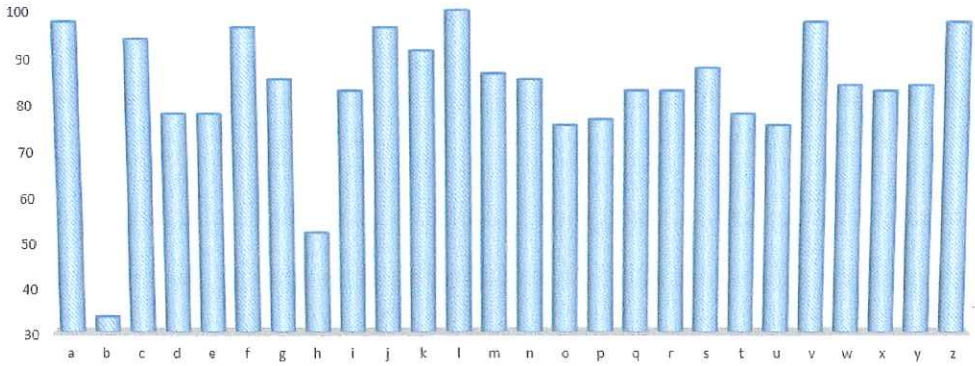


Figura 5.6: Eficiencia con 500 puntos uniformemente distribuidos

5.2. Comparativa de complejidad computacional

Para el análisis de complejidad se toma como principal característica el número de ciclos que se necesita para realizar una parte del algoritmo. Como se puede apreciar en la Figura (5.7) se obtienen primeramente la adquisición de los datos en forma serial y se guardan variables temporales.

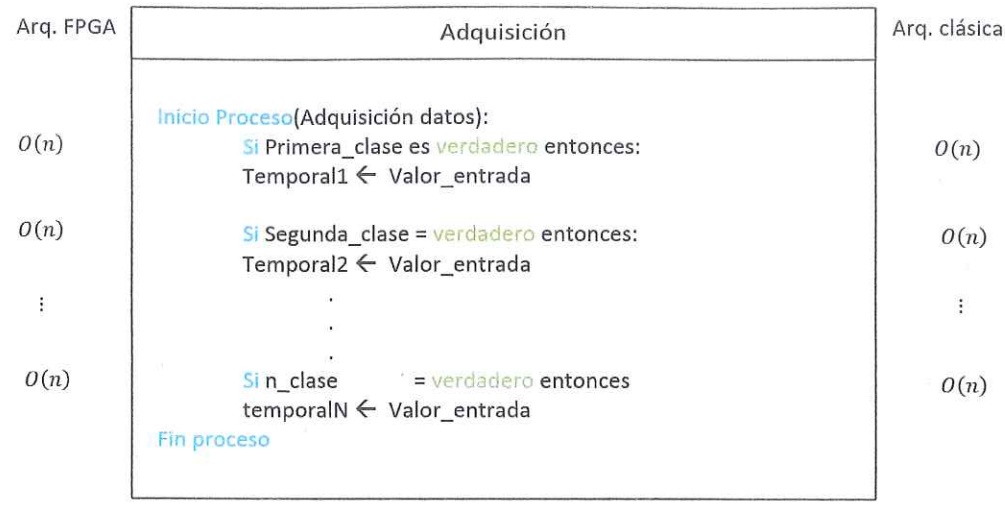


Figura 5.7: Análisis de complejidad en la adquisición

Despues la parte de aprendizaje se basa en el criterio de *Steinbuch* donde se obtiene:

$$\begin{aligned} f(0) &= -1 \\ f(1) &= 1 \end{aligned} \tag{5.1}$$

En ambos casos es similar el número de ciclos utilizado, ya que la forma en que se recorre cada elemento bajo este criterio.

Cabe mencionar que en la esta parte se utiliza el complemento A2 para la representación de números negativos en el dispositivo reconfigurable. Aumentando un bit de signo y cambiando el criterio a:

$$\begin{aligned} f(0) &= 11 \\ f(1) &= 01 \end{aligned} \tag{5.2}$$

Todo esto de describe en la parte del pseudocódigo de la Fig (5.8)

Por ultimo, la fase de reconocimiento toma la mayor importancia en este trabajo ya que se realiza disminución de ciclos más significativa de trabajo. Describiendo el procedimiento comenzamos con insertar un vector que contiene la información de un carácter, este previamente codificado fuera de linea, se multiplica por la matriz de aprendizaje contenida en la memoria temporal de la *RAM* de la *FPGA*. La operación se hace independientemente,

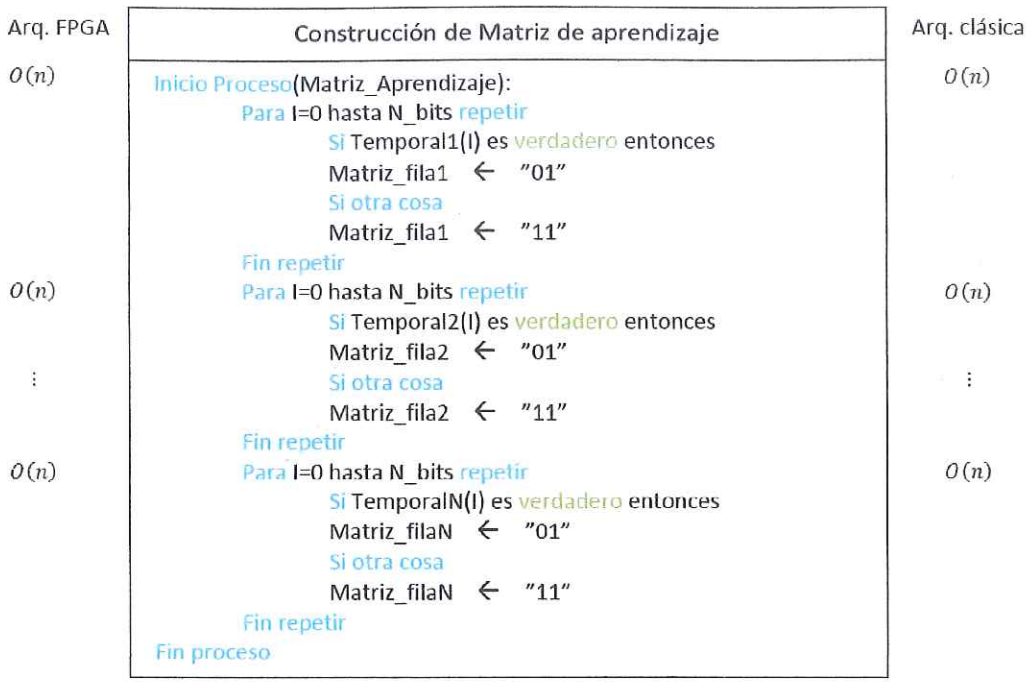


Figura 5.8: Análisis de complejidad en la construcción de la matriz

lo que hace que el resultado se obtenga en el siguiente ciclo de reloj. Posteriormente se localiza el máximo de los resultados creando un árbol de decisión que nos proporciona el valor y lo coloca en un vector de salida, se localiza el posición del máximo y se procesa fuera de del FPGA.

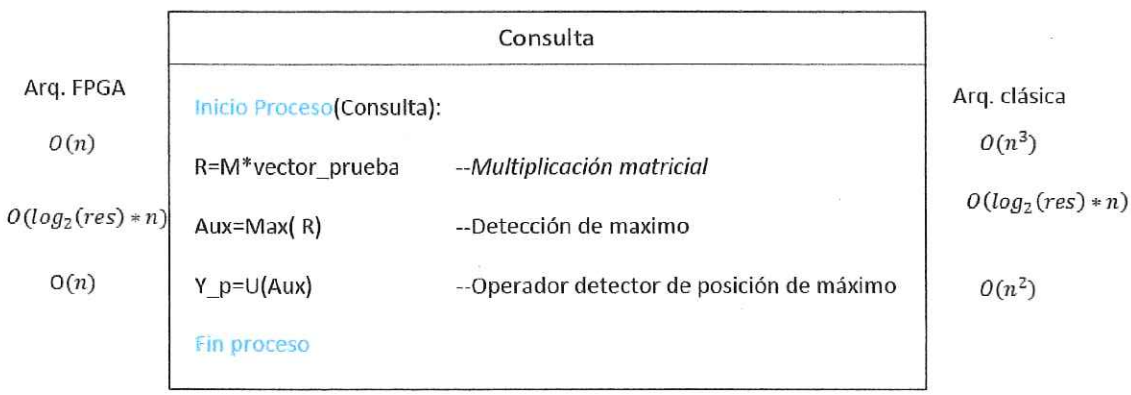


Figura 5.9: Análisis de complejidad en la consulta

5.3. Resultados experimentales de evaluación de complejidad.

Al obtener los parámetros de la complejidad observamos que en ambos algoritmos se muestra un análisis basado en la eficiencia temporal que se presenta una simulación de tiempos. El primer aspecto a considerar es analizar la adquisición de los datos el cual genera una pequeña latencia descrita por la siguiente ecuación. Cabe mencionar que el aprendizaje de la matriz de Steinbuch se realiza al mismo tiempo que se van ingresando lo que nos permite la utilización de la matriz cuando un ciclo después.

$$2 \times O(k \times n) \tag{5.3}$$

Donde k es el numero de clases.

Por lo que respecta a la consulta se calculo la ecuación correspondiente al análisis de complejidad por parte de la arquitectura clásica.

$$O(n^2) + O(n^3) + O(log_2(res) \times n) \tag{5.4}$$

Donde *res* corresponde a la resolución de la cadena binaria.

Los resultados de la arquitectura implementada en el dispositivo reconfigurable generaron la siguiente ecuación polinomial.

$$O(n) + O(n) + O(log_2(res) \times n) \tag{5.5}$$

Como se puede observar se disminuye la complejidad de una arquitectura clásica con $O(n^3)$ a tener una en la arquitectura en FPGA con $O(n)$.
Se implemento una prueba experimental en la cual se probó el algoritmo con los siguientes parámetros.

- Un reloj de 100kHz correspondientes a $10\mu s$

- Un clasificador de un abecedario con 26 clases
- Resolución de la codificación de cada carácter de 32 bits

Para observar la disminución de la temporalidad como consecuencia de la reducción de ciclos de reloj sustituimos en el polinomio del peor de los casos. Lo correspondiente a la latencia de adquisición de cadenas:

$$T(n) = 2 \times (26 \times 10\mu s) = 0{,}26ms \tag{5.6}$$

La fase Consulta con arquitectura clásica

$$T(n) = (10\mu s)^2) + (10\mu s)^3 + log_2(32) \times 10\mu s = 11{,}5ms \tag{5.7}$$

Consulta con FPGA

$$T(n) = 2 \times (10\mu s) + (log_2(32) \times 10\mu s) = 0{,}07ms \tag{5.8}$$

Dichos parámetros fueron metidos al simulador de VHDL obteniendo lo resultados mostrados en la Fig 5.10 observando con la fecha roja el tiempo transcurrido en el aprendizaje y adquisición de las cadenas binarias.

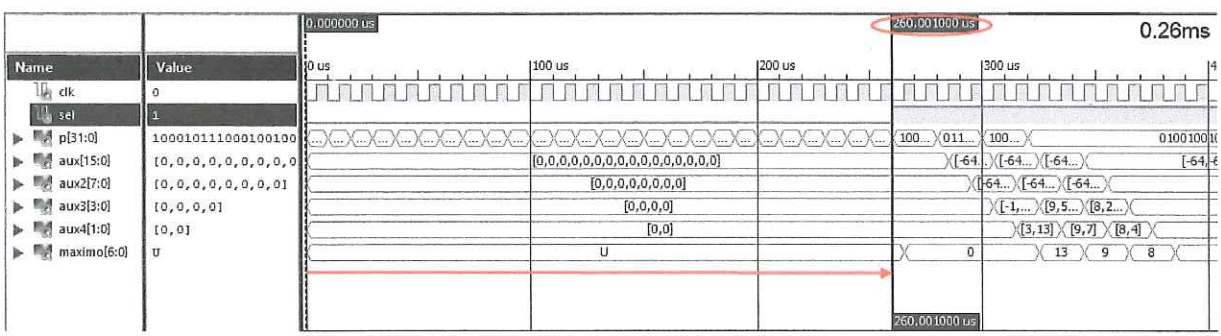


Figura 5.10: Simulación de adquisición en conjunto con aprendizaje

Continuando con la simulación se muestra en la Fig 5.11 que las operaciones matriciales se hacen en un ciclo de reloj y la detección de máximo según la resolución de la cadena binaria lo cual hace comparaciones para resolverse. Como podemos notar los valores se acercan a los calculados lo que nos da una idea de las reducción de ciclos de maquina.

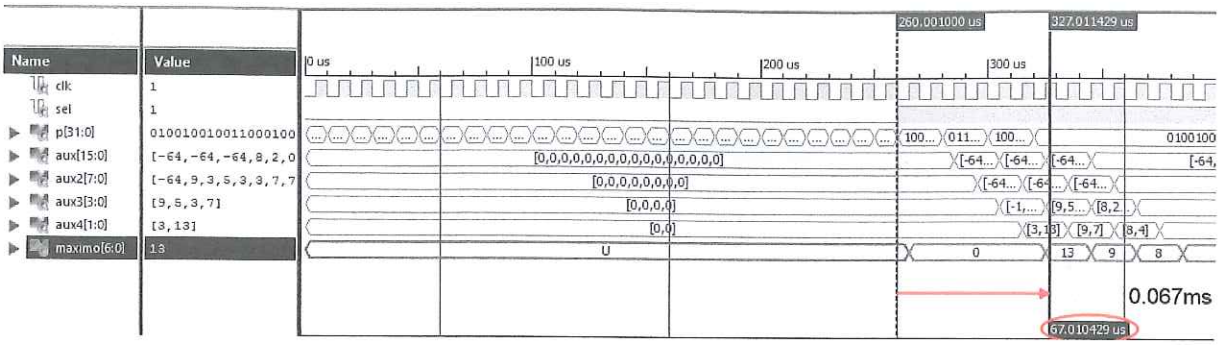


Figura 5.11: Simulación de consulta en FPGA

Tomamos un fragmento de texto al azar para analizar la temporalidad del clasificador.

> *Las FPGAs se utilizan en aplicaciones similares a los ASICs sin embargo son más lentas, tienen un mayor consumo de potencia y no pueden abarcar sistemas tan complejos como ellos. A pesar de esto, las FPGAs tienen las ventajas de ser reprogramables (lo que añade una enorme flexibilidad al flujo de diseño), su costo de desarrollo y adquisición son mucho menores para pequeñas cantidades de dispositivos y el tiempo de desarrollo es también menor.*

En este corresponden 74 palabras y 345 letras sin considerar los espacios se hace una aproximación de tiempos considerando los siguientes parámetros:
Una latencia asociada al reconocimiento del alfabeto de 26 ciclos correspondientes a cada letra diferente presente en ambos casos, y el reconocimiento de cada carácter multiplicado por los ciclos usados para la detección de máximo dando como resultado:

En una arquitectura clásica basada en un procesador con un hilo de ejecución, las matrices se realizan con varios ciclos de reloj secuencialmente.

$$26\ clases \times 10\mu s + 345\ letras \times 11{,}5ms = 3{,}96s \approx 4s \tag{5.9}$$

Por otra parte en la arquitectura implementada en FPGA con operadores matriciales independientes.

$$26\ clases \times 10\mu s + 345\ letras \times 0{,}07ms = 0{,}0244s \approx 0{,}03s \tag{5.10}$$

Dadas los resultados obtenidos por los calculo del polinomio $O(n)$, por lo que podemos garantizar la reducción de complejidad utilizando un dispositivo reconfigurable con operadores matriciales independientes en comparación con un procesador que utiliza un hilo de ejecución y operadores secuenciales.

Capítulo 6

Conclusiones.

Podemos concluir a partir de los resultados obtenidos que se comprueba la posibilidad de reconocer caracteres empleando una representación por cadenas largas binarias (cadenas simbólicas). Utilizando el modelo de muestreo aleatorio en imágenes nos permite extraer información para clasificar diversos caracteres. Así se puede evitar la necesidad de definir descriptores basados en descomposición a formas básicas, esquinas o morfología de los caracteres.

Utilizando el muestreo aleatorio se reduce la dimencionalidad de los datos convirtiendo al modelo en una versión computacionalmente más económica en cuestión de utilización de recursos. Se observo que la longitud (dimensionalidad del espacio) de las cadenas tiene relación con la cantidad de símbolos codificados y afecta las capacidades de reconocimiento, es decir, si los símbolos a caracterizar son demasiados, en relación a la longitud de las cadenas, se tienen índices de reconocimiento bajos.

El tiempo de cómputo se reduce significativamente cuando se utiliza un hardware reconfigurable para la realización de las operaciones matriciales. Demostrando que se pueden realizar en paralelo dichas operaciones disminuyendo los ciclos de reloj, por consecuencia el tiempo de ejecución de los clasificadores basados en la memoria asociativa de Steinbuch.

Otra consideración al implementar clasificadores con FPGA es considerar número de clases para evitar el desbordamiento de bits provocado por la resolución de los puntos aleatorios generados.

Como trabajo a futuro producido por el proyecto se enumeran algunos puntos:

- Una posibilidad es la ampliación de los clasificadores de Steinbuch para otras

aplicaciones combinando la velocidad y eficiencia de este método.

- La implementación de clasificadores con memorias asociativas más robustas.
- Combinación de procesadores ARM y FPGA para aprovechar las ventajas de ambos sistemas, ya que en algunos casos no es posible una paralelización total por la naturaleza del problema, se consideran paralelizaciones parciales en algunas partes de los procesos de adquisición de señales y operaciones matriciales del algoritmo.
- Programación de sistemas distribuidos en dispositivos reprogramables.

Bibliografía

[1] S.-H. Liu, J.-S. Lin, and S.-Y. Huang, "Character recognition with cmac on field programmable gate array," in *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*, vol. 2, pp. 1109–1112, IEEE, 2004.

[2] P. Kanerva, *Sparse distributed memory*. The MIT Press, 1988.

[3] K. Steinbuch, "Die lernmatrix," *Biological Cybernetics*, vol. 1, no. 1, pp. 36–45, 1961.

[4] H. J. y J. Figueroa, "Reconocimiento de caracteres por cadenas binarias aleatorias," vol. 2, pp. 125–130, CNyCIIC2003, Avances en informática y computación.

[5] J. Naguil, E. Brac, P. Ferreyra, and C. Marques, "Implementacion de un red neuronal de hopfield ultradiluida para reconocimiento de rostros en un dsp,"

[6] A.-N. Suen, J.-F. Wang, and T.-D. Wang, "A programmable application-specific vlsi architecture and implementation for speech word-recognizer," in *Design Automation Conference 1997. Proceedings of the ASP-DAC'97. Asia and South Pacific*, pp. 71–75, IEEE, 1997.

[7] D. Wentzlaff and A. Agarwal, "A quantitative comparison of reconfigurable, tiled, and conventional architectures on bit-level computation," in *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*, pp. 289–290, IEEE, 2004.

[8] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*, vol. 74. Prentice hall Englewood Cliffs, 1995.

[9] J. P. Sterba and C. Quinn, "Davis baird on nano tech," *Social Theory and Practice*, vol. 29, no. 2, 2003.

[10] M. Minsky, *Society of mind*. SimonandSchuster. com, 1988.

[11] T. S. Kuhn, *The structure of scientific revolutions* . University of Chicago press, 1996.

[12] B. J. Copeland, *The Essential Turing*. Oxford University Press, 2004.

[13] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[14] Y. Bar-Yam, "Dynamics of complex systems (studies in nonlinearity)," 2003.

[15] B. J. Copeland and R. Sylvan, "Beyond the universal turing machine," *Australasian Journal of Philosophy*, vol. 77, no. 1, pp. 46–66, 1999.

[16] P. Dayan, L. F. Abbott, and L. Abbott, "Theoretical neuroscience: Computational and mathematical modeling of neural systems," 2005.

[17] F. Rieke, D. Warland, R. De Ruyter van Steveninck, and W. Bialek, "Exploring the neural code," 1997.

[18] M. Minsky and S. Papert, "Perceptrons: an introduction to computational geometry (expanded edition)," 1988.

[19] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[20] S. Santini and R. Jain, "Similarity measures," *Pattern analysis and machine intelligence, IEEE transactions on*, vol. 21, no. 9, pp. 871–883, 1999.

[21] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," *Acm Computing Surveys (CSUR)*, vol. 35, no. 4, pp. 399–458, 2003.

[22] C. Zhang and Z. Zhang, "A survey of recent advances in face detection," tech. rep., Tech. rep., Microsoft Research, 2010.

[23] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[24] C. BenAbdelkader, R. Cutler, H. Nanda, and L. Davis, "Eigengait: Motion-based recognition of people using image self-similarity," in *Audio-and Video-Based Biometric Person Authentication*, pp. 284–294, Springer, 2001.

[25] J. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[26] R. T. Collins, R. Gross, and J. Shi, "Silhouette-based human identification from body shape and gait," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pp. 366–371, IEEE, 2002.

[27] T. Kohonen, "Self-organization and associative memory," *Self-Organization and Associative Memory, 100 figs. XV, 312 pages.. Springer-Verlag Berlin Heidelberg New York. Also Springer Series in Information Sciences, volume 8*, vol. 1, 1988.

[28] B. Kosko, "Bidirectional associative memories," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 18, no. 1, pp. 49–60, 1988.

[29] J. Austin, "Adam: A distributed associative memory for scene analysis," in *Proceedings of First International Conference on Neural Networks*, vol. 4, p. 285, 1987.

[30] J. T. Palma, M. C. Garrido, F. Sánchez, and A. Quesada, "Programación concurrente," *Thomson*, 2003.