

Cambiar logotipos. Le mande la plantilla nueva,
espero que no le lleve mas de 5 minutos hacer
la adaptacion



UNIVERSIDAD POLITÉCNICA DE VICTORIA



**CLASIFICADOR DE NARANJAS POR TAMAÑO Y COLOR
IMPLEMENTADO EN FPGA USANDO APRENDIZAJE
SUPERVISADO**

T E S I S
QUE PARA OBTENER EL GRADO DE
MAESTRO EN INGENIERÍA
P R E S E N T A

Ismael Antonio Dávila Rodríguez

DIRECTOR DE TESIS:
DR. MARCO AURELIO NUÑO MAGANDA

CO-DIRECTOR DE TESIS:
DR. YAHIR HERNÁNDEZ MIER

© Derechos reservados por
Ismael Antonio Dávila Rodríguez
2020

Agradecimiento especial al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo financiero recibido durante los estudios de maestría.

La tesis presentada por Ismael Antonio Dávila Rodríguez fue aprobada por:

DR. MARCO AURELIO NUÑO MAGANDA, Director

DR. YAHIR HERNÁNDEZ MIER, Co-Director

Cd. Victoria, Tamaulipas, México, 7 de Octubre de 2020

Este trabajo de tanto esfuerzo lo dedico enteramente a mis padres por su amor incondicional en todos mis años de estudio. A mi familia por siempre confiar en mi.

Agradecimientos

- CONACYT

Índice General

Índice General	i
Índice de Figuras	v
Índice de Tablas	vii
Índice de Algoritmos	ix
Resumen	xii
Abstract	xiii
Nomenclatura	xv
1. Introducción	1
1.1. Antecedentes	1
1.2. Planteamiento del problema	3
1.3. Justificación	6
1.4. Estado del arte	9
1.5. Objetivos	11
2. Marco Teórico	13
2.1. Inteligencia artificial	13
2.2. Machine learning	14
2.3. Máquinas de aprendizaje supervisado	15
2.4. Árbol de decisión	16
2.5. Algoritmo de entrenamiento IDE3	17
2.6. Algoritmo de entrenamiento C4.5	17
2.7. Procesamiento de imágenes	18
2.7.1. Transformaciones de color	18
2.7.1.1. Transformación RGB a escala de grises	18
2.7.1.2. Transformación RGB a HSV	19
2.7.2. Segmentación	20
2.7.2.1. Binarización por umbral	20
2.7.2.2. Contornos	22
2.8. Visión por computadora	22
2.8.1. Histogramas	22
2.9. FPGA	22
2.9.1. Historia de los FPGAs	25
2.9.2. Arquitectura general de un FPGA	28

2.9.3. Componentes y características	29
2.9.4. Lenguajes de descripción de Hardware (HDL)	33
2.9.5. Implementación	35
2.9.6. Ventajas de FPGA	36
2.10. Procesamiento de imágenes en FPGA	38
2.10.1. Ventajas y desventajas de FPGA en procesamiento de imágenes	38
3. Desarrollo	39
3.1. Sistema propuesto	39
3.2. Maquinaria de flujo de naranja	41
3.2.1. Embudo de alimentación	42
3.2.2. Banda de giro	42
3.2.3. Banda de traslación	42
3.2.4. Banda elevadora	42
3.2.5. Embudo alineador	43
3.2.6. Banda principal	43
3.2.7. Cámara de proceso	43
3.2.8. Zona de actuadores	43
3.2.9. Zona de fruta Clasificada	44
3.3. Flujo de fruta en maquinaria	44
3.4. Sistema eléctrico/electrónico	44
3.4.1. Elementos mecánicos	45
3.4.1.1. Actuadores	45
3.4.1.2. Motores	45
3.4.1.3. Etapa de potencia	46
3.4.2. Procesador de video	46
3.4.3. Elementos auxiliares	47
3.4.4. Elementos de control y medición	48
3.4.4.1. Tablero de control	48
3.4.4.2. Sensores	48
3.4.4.3. Controlador General	49
3.5. Modelo ADD de segmentación	50
3.5.1. Captura de video	51
3.5.2. Captura de fotogramas de interés	51
3.5.3. Captura de data set de pixeles	54
3.6. Modelo ADD clasificador de color	55
3.6.1. Análisis de color	56
3.6.2. Captura de data set de histogramas	56
3.7. Modelo ADD clasificador por tamaño	59
3.8. Arquitectura genérica de árbol de decisión	59
3.8.1. Generación de un HDL a partir de un ADD	61
3.9. Arquitectura de procesador de video	62
3.10. Módulo Clasificador	63

3.10.1. Conversión de espacios de colores y cálculo de ROI	64
3.10.1.1. Módulo convertidor RGB a escala de grises	64
3.10.1.2. Módulo convertidor RGB a HSV	65
3.10.1.3. Módulo de cálculo de posición, ROI y fin de ROI	67
3.10.2. Segmentación	68
3.10.2.1. Módulo ADD segmentador	68
3.10.3. Cálculo de histogramas y área de segmentación	69
3.10.3.1. Histogramas de pixeles segmentados	69
3.10.3.2. Módulo de cálculo de área de segmentación	70
3.10.4. Clasificación de color y tamaño	71
3.10.4.1. Módulo ADD de clasificación de color	72
3.10.4.2. Módulo clasificador de tamaño	72
3.11. Módulos de comunicación y gestión de respuesta	73
3.11.1. Módulo de comunicación	73
3.11.2. Módulo de secuencia de respuesta	73
4. Resultados y discusión	75
4.1. ADD de segmentación	76
4.1.1. Captura de data set de pixeles	76
4.1.2. Entrenamiento y prueba por computadora	76
4.1.3. Implementación en FPGA	76
4.2. ADD de clasificación de color	76
4.2.1. Captura de data set de histogramas	76
4.2.2. Entrenamiento y prueba por computadora	76
4.2.3. Implementación en FPGA	76
4.3. Clasificación de tamaño	76
4.3.1. Captura de data set de áreas	76
4.3.2. Entrenamiento y prueba por computadora	76
4.3.3. Implementación en FPGA	76
5. Conclusiones	77
Bibliografía	79

Índice de Figuras

1.1. Fotografía de clasificación manual.	4
1.2. Dibujo de un clasificador mecánico.	5
1.3. Imagen de clasificación por visión artificial.	5
1.4. Condiciones poco probables.	10
2.1. Ejemplo de la estructura de un ADD.	15
2.2. Ejemplo de clasificación en 2 dimensiones.	16
2.3. Ejemplo de la estructura de un ADD.	17
2.4. Representación axial del espacio de color RGB.	19
2.5. Conversión de imagen a color a escala de grises.	20
2.6. Pirámide de espacio de color HSV.	21
2.7. Binarización de imágenes usando un umbral $t = 127$	21
2.8. Arquitectura más simple de un PAL.	27
2.9. Arquitectura genérica de un FPGA.	29
2.10. Arquitectura general de un bloque Configurable Lógico (CLB).	30
2.11. Arquitectura general de un CL formado con slices.	30
2.12. Arquitectura general de un módulo Lógico.	31
2.13. Arquitectura general de una LUT.	31
2.14. Esquema básico de un IOB.	32
2.15. Arquitectura general de una BSC.	33
3.1. Diagrama general del sistema.	40
3.2. Diagrama general de maquinaria.	41
3.3. Diagrama de flujo de procesamiento de fruta en la maquinaria.	45
3.4. Diagrama eléctrico/electrónico general.	46
3.5. Diagrama de conexión de elementos mecánicos	47
3.6. Configuración de terminales en MCU	50
3.7. Diagrama de flujo del programa del MCU	51
3.8. Escena de cámara dentro de cabina.	52
3.9. Captura para el set de datos y obtención de la máscara. Difuminación por movimiento.	53
3.10. Proceso de captura de imágenes.	54
3.11. Diagrama de flujo de captura de dataset de pixeles.	55
3.12. Resultados del análisis	57
3.13. Diagrama de flujo y ejemplo de captura de dataset de histogramas.	58
3.14. Modelos de ADD en Hardware.	60
3.15. Ejemplo de ADD resuelto con SCHADD.	62
3.16. Esquema general de la arquitectura.	63

Índice de Tablas

3.1. Características de motores	46
---	----

Índice de Algoritmos

1.	Algoritmo para módulo de histogramas	70
2.	Algoritmo para cálculo del área segmentada	71
3.	Secuencia de respuesta	74

Resumen

Clasificador de naranjas por tamaño y color implementado en FPGA usando aprendizaje supervisado

por

Ismael Antonio Dávila Rodríguez

Maestría en Ingeniería

Universidad Politécnica de Victoria, 2020

DR. MARCO AURELIO NUÑO MAGANDA, Director

DR. YAHIR HERNÁNDEZ MIER, Co-Director

AL FINAL

Abstract

Citrus fruit classifier by size and color FPGA implemented using supervised machine learning

by

Ismael Antonio Dávila Rodríguez

Master of Engineering

University Polytechnic of Victoria, 2020

DR. MARCO AURELIO NUÑO MAGANDA, Advisor

DR. YAHIR HERNÁNDEZ MIER, Co-advisor

AL FINAL

Nomenclatura

Creo que no deberias mezclar acronimos en español y en ingles
O todos en Español o todos en Ingles

ADD Árbol de decisión NO ABREVIAR Árbol de Decision (NI EN ESPAÑOL ni en INGLES)

FAO Food and Agriculture Organization

ML Machine Learning

RNA Red Neuronal Artificial

ARFF Attribute-Relation File Format

FPGA Field Programmable Gate Array

LSB Less Significant Bit

MSB Most Significant Bit

SVA Señal de Video Activo

SCHAAD Script Constructor HDL de Árbol De Desición

SSH Señal de Sincronización Horizontal

SSV Señal de Sincronización Vertical

BDS Bit de Segmentación

Reglas para estructura de un trabajo de tesis

* Un Capitulo, seccion, subseccion y subsubseccion deberia tener por lo menos una cuartilla de extension. En el caso de secciones , subsecciones y subsubsecciones, si la extension no sobrepasa 4 parrafos, no vale la pena ponerlo como una seccion , subseccion o subsubseccion, sino que debe incluirse como una lista de componentes con bullets.

* En el caso de los bullets, tampoco hay que abusar. Es preferible decir "La computadora tiene como perifericos, teclado, raton, camara USB e impresora" que hacer una lista con items tan cortos, a menos que lleven descripcion. Lo siguiente tiene la intencion de gastar espacio mas que expresar algo contundente.

"La computadora tiene los siguientes perifericos:

- ** Teclado
- ** Raton
- ** Camara USB
- ** Impresora

* Parrafos de menos de 3 lineas hacen ver mal al documento. Si tienes hay muchas ideas en parrafos de dos lineas, lo ideal es juntarlos en un parrafo aun mas grande

* Lo contrario se ve mal. Si se tiene un parrafo que abarca mas de tres cuartos de cuartilla, se ve mal, conviene dividirlo en dos parrafos

Otro punto que veo es que evites escribir parrafos redundantes. Por ejemplo:

~~1.2. Vision por computadora~~

~~En esta seccion hablaremos de vision por computadora. Es la rama de ...~~

~~1.2. Vision por computadora~~

La vision por computadora es la rama de la inteligencia artificial...

1

Introducción

1.1 Antecedentes

Desde la antigüedad los cítricos han sido parte fundamental de los alimentos, teniendo su origen desde las regiones tropicales y subtropicales del sureste de Asia y el archipiélago Malayo y esparcirse a todo el mundo. Partiendo históricamente por la planta del cidro vista por primera vez en Europa en 300 a.C. el cuál se utilizaba para el injerto y generación de especies híbridas de plantas y frutas, esto siendo una particularidad de los cítricos, ya que presentan mutaciones con mucha frecuencia incluso entre miembros de la misma especie [1].

Desde el siglo XVI, los cítricos fueron ganando popularidad por toda Europa aunque desde el siglo VI se cultivaban, hasta que en el siglo XIX fué el naranjo amargo el más popular entre el cultivo.

En la época actual los mayores productores de cítricos son China, Brasil, Estados Unidos, España, India, México y Argentina. De acuerdo a la Organización de Agricultura y Alimentos (**FAO** por sus siglas en inglés), México es uno de los 5 mayores productores de cítricos en el mundo [2].

1. Introducción

Actualmente, en el estado de Tamaulipas es el segundo productor de cítricos en México. En el 2011, la producción total fue de 500,000 toneladas aproximadamente [3].

A inicios de 1960, la visión por computadora iniciaba en las universidades de Estados Unidos, herramienta que hasta el día de hoy busca emular la visión de los humanos a través de computadoras para el proceso de interpretación de imágenes, las cuales pueden producir estímulos en procesos industriales de clasificación. Esta herramienta se ha utilizado en diferentes ámbitos industriales, incluyendo la alimentaria.

Hoy en día se cuenta con sistemas de visión en la industria alimentaria para la clasificación de frutas como los *cítricos*. Los cuales cuentan con bandas transportadoras las cuales hacen pasar la fruta por el sistema clasificador, el sistema normalmente se comunica con un sistema de movimiento de actuadores que realiza la tarea de separación. Por lo general estos sistemas sirven para determinar el tamaño, color, maduración y calidad de la fruta. Ya que una maquinaria trabaja más rápido que un sistema manejado enteramente por humanos la producción puede ser elevada, abasteciendo la gran demanda mundial antes mencionada.

Sin embargo en México no se tienen tecnologías muy desarrolladas en el ámbito, contando solamente con tecnologías extranjeras y obsoletas. Por eso la necesidad de desarrollar un proyecto que mediante tecnologías actuales y nacionales, solventar la producción de cítricos.

En las secciones posteriores de éste capítulo se aborda una pequeña introducción teórica acerca de los sistemas de clasificación actuales y su problemática nacional. En el Capítulo 2 se muestran todos los conceptos técnicos referentes a la realización del proyecto para poder comprender el principio y naturaleza de la solución propuesta. En el Capítulo 3 se muestra el desarrollo de la plataforma general del sistema propuesto, donde se explica el sistema mecánico, el flujo de información y la arquitectura propuesta del sistema. En el Capítulo 4 se demuestran los resultados obtenidos y la eficiencia detrás de la naturaleza de la solución propuesta desde los ADD. En el Capítulo 5, de

encuentran las conclusiones del uso de modelos de ADD en un FPGA.

1.2 Planteamiento del problema

Las clasificadoras de fruta son aquellas capaces de ordenar al producto en diferentes grupos los cuales pueden ser:

comentario de las listas de bullets sin descripción!

- Por color
- Por tamaño
- Por maduración

En México existen clasificadoras que pueden separar estas clases mencionadas. Generalmente la mayoría de las veces se utilizan clasificadores manuales o mecánicos, ya que no cuentan con tecnología electrónica actual por lo que presentan muchos inconvenientes, partiendo que la mayoría solamente desempeñan 1 sola clasificación a la vez. Existen de varios tipos de clasificadores:

✓ Clasificadora manual

Es literalmente un proceso artesanal (figura 1.1) en el cual un grupo de personas se encarga de seleccionar la fruta conforme a su criterio. Las mismas personas se encargan de llevar a cabo cada etapa en la selección de la fruta, desde la limpieza hasta el empacado, convirtiendo este proceso en largas jornadas de trabajo arduo y dependencia en un 100 % de la intervención del hombre.

Problemas frecuentes:

- Se necesita de empleados, y todas las responsabilidades que esto conlleva
- Velocidad de producción no estable
- Clasificación a criterio del personal

✓ Clasificadora mecánica

1. Introducción



Figura 1.1. Fotografía de clasificación manual.[Referencia]

Este tipo de clasificadora (figura 1.2) la comprende una estructura, que debido a su forma geométrica puede clasificar la fruta solamente por tamaño, ya que no cuenta con ningún dispositivo electrónico de procesamiento de datos. El movimiento de la fruta puede ser por gravedad o movida por motores.

Problemas frecuentes:

Una sola idea en un solo párrafo

- Degradación de calidad de fruta (golpes y cortes)
- Propensos a atascamientos
- Generadores de desperdicio

✓ **Clasificadora automática**

En los últimos años ha aumentado el uso de tecnología en la selección de fruta, donde predominan los métodos de inspección ópticos (cámaras) el cuál es un tipo de inspección no destructiva, aplicado

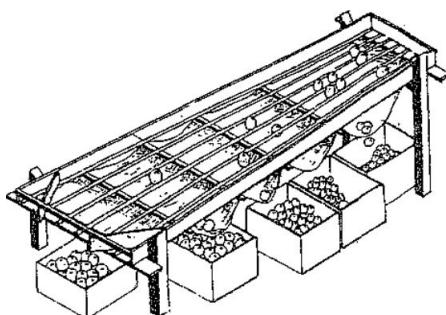


Figura 1.2. Dibujo de un clasificador mecánico.

Si es propio tu dibujo no incluyas referencia

a diversas frutas como manzanas, plátanos, fresas, limones, duraznos, naranjas, peras, papayas, tomates, papas, melones, sandía, etc como se verá en las secciones posteriores. En la figura 1.3 se puede ver una máquina que cuenta con la capacidad de inspección por cámaras asistido con visión por computadora.

Problemas frecuentes:

- Plataforma de implementación obsoleta (computadoras, tecnologías de comunicación, etc).
- Hardware obsoleto (difícil reparación).
- Clasificación lenta.

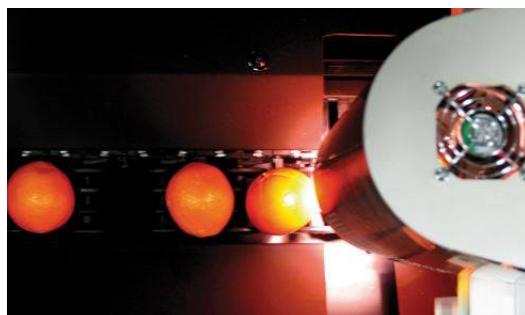


Figura 1.3. Imagen de clasificación por visión artificial. [\[Referencia\]](#)

Con el fin de mejorar el ~~y~~^h hardware existente en velocidad, usar un arreglo de compuertas lógicas programables (**FPGA** por sus siglas en inglés), parece una opción prometedora, ya que estos se destacan por su procesamiento en paralelo, y flexibilidad de construcción de ~~x~~^h hardware.

1. Introducción

1.3 Justificación

De acuerdo a la Organización de Agricultura y Alimentos (**FAO** por sus siglas en inglés), México es uno del los 5 mayores productores de cítricos en el mundo [2]. Actualmente, en el estado de Tamaulipas es el segundo productor de cítricos en México. En el 2011, la producción total fue de 500,000 toneladas aproximadamente [3].

El tratamiento post-cosecha se ha convertido en una etapa esencial en el mercado de fruta fresca, para mantener la frescura de los cítricos y proveer al consumidor en las mejores condiciones posibles. Comúnmente, la calidad de la clasificación de cítricos es realizado por humanos, basado en inspección visual de un criterio externamente visible de la fruta, como su tamaño, forma y color. Esta inspección manual recae mucho en la experiencia del observador, para mantener la consistencia y uniformidad en la clasificación.

Algunos autores han reportado algunos sistemas de clasificación. En [4], las técnicas de clasificación son revisadas. El autor menciona las características mas usadas para identificar pudrición, maduración y clase de frutas, en modelos de machine learning (**ML**) usadas para la tarea. Dos grupos grandes de enfoques fueron identificados:

1. Sistemas enfocados a la identificación de múltiples frutas (Por lo que no se revisa la calidad de la misma), alimentada con sets de miles de imágenes de una serie de diferentes frutas
2. Sistemas enfocados en una fruta específica, usando un set de imágenes de un solo tipo de fruta para entrenar y probar el clasificador. Aun y cuando el primer enfoque es más general, el segundo es una mejor opción para máquinas clasificadoras, diseñadas para clasificar una sola fruta.

Para sistemas específicamente diseñadas para clasificación de cítricos, una evaluación de diferentes clasificadores automáticos para naranja es reportado en [5], donde 5 arboles de decisión (ADD) diferentes, tres redes neuronales artificiales (RNA) y un clasificador por reglas fueron probados. En

[6], un clasificador de tipo árbol de reglas fue propuesto para clasificar la maduración de las naranjas. En [7], un algoritmo de segmentación multiespectral basada en una estructura de datos de arboles cuadruples es presentada.

Sistemas para clasificación de frutas que no son cítricos también han sido propuestos. En [8], un clasificador Bayes y un clasificador de manzanas basado en el espacio de colores de Ohta es presentado. En [9], un método para reconocer las regiones de tallo o cáliz en manzanas es propuesto. Eliminación de fondo y segmentación de objetos por umbralización son usadas para aislar el objetivo. Después por características estadísticas, texturas y forma son extraídas de cada objeto segmentado y usado para el entrenamiento de los modelos: lineal discriminante, vecino más cercano (KNN), vecino más cercano difuso (KNNF), máquinas de soporte vectorial (SVM) y clasificador adaboost.

En [10], un prototipo para clasificar fruta en base a las fecha de cosecha es propuesto. RNAs son entrenadas con el set de datos de 1800 frutas para obtener características como debilidad, tamaño, forma, intensidad y defectos.

Relacionados a sistemas de multiples frutas, hay algunas aplicaciones en la literatura. En [11], una transformación curvelet para la clasificación de guayaba y limón es propuesto. Medidas de texturas basadas en la transformación curvelet como la energía, entropía, media y desviación estandar es usada para caracterizar textura de la superficie de la fruta. Modelos como SVM, y Redes Neuronales Artificiales Probabilisticas (RNAP) fueron entrenadas para distinguir entre frutas en buen estado y con defectos. En [12], un sistema para clasificar manzanas, frutas y naranjas es presentado. Utilizando una Transformación de Características Invariantes a Escalamiento (SIFT) fue usado para obtener características de las frutas presentadas, y un clasificador de tipo Bosque aleatorio fue aplicado de igual manera. En [13], un algoritmo KNN para clasificar frutas basado en su color, forma y tamaño es reportado. El enfoque propuesto fue aplicado para manzanas, limones, plátanos y fresas. En [14], descriptores de imágenes basadas en apariencia, color, textura, y forma fueron obtenidos y distinguidos de las técnicas de ML como SVM, LDA, arboles de clasificación, KNN, y ensambles de

1. Introducción

árboles fueron entrenados.

Recientemente, modelos de Deep Learning (DL) han sido intensivamente desarrolladas para la clasificación de problemas complejos. Esos modelos son mostrados para entender mejor las características en imágenes, y han sido usadas para afrontar con el rápido crecimiento de datos (big data). En [15], un clasificador basado en DL es propuesto. Algoritmos de DL requieren mucho tiempo entrenamiento de datos, una gran cantidad de datos de entrenamiento (set de datos), y un poder de procesamiento enorme. Generan una solución muy especializada en un dominio específico, donde la revaloración es necesaria para resolver problemas fuera del dominio.

El diseño de un sistema de visión para clasificación automática de cítricos es altamente deseada para reducir el costo de la fuerza laboral y problemas de mantener la consistencia y uniformidad en la clasificación. Un clasificador automático de cítricos basada en visión de computadora afronta algunos problemas técnicos como, el procesamiento de imágenes en tiempo real, baja precisión de clasificación y alto costo computacional. Un FPGA es ideal para esta tarea, ya que provee de procesamiento paralelo. En el enfoque presentado en [16], una arquitectura para clasificación de cítricos es presentada. El pre-procesamiento requerido para la clasificación es implementada en hardware. La clasificación de cítricos fue basada en una umbralización global, convirtiendo al sistema sensible al ruido generado por los objetos en movimiento y un modelo de ML más robusto no fueron explorados.

Algunos algoritmos de clasificación para frutas como ANN, SVN requieren mucha cantidad de memoria para almacenar diferentes coeficientes y complejos cálculos no lineales. Los árboles de decisión (ADD), son uno de los mejores algoritmos para implementación de hardware, ya que no requieren cálculos aritméticos, que son computacionalmente costosos, si no que solamente usa comparadores [17]. Los ADD han sido aplicados exitosamente en algunos dominios, como reconocimiento de objetos [18], identificación de gas [19], y reconocimiento de cantar de aves [20].

La comparación entre los sistemas de clasificación es difícil por su larga variedad y combinatoria de sets de imágenes de frutas. Precisión, especificidad y sensibilidad de mediciones son a menudo reportadas en diversos trabajos, pero no hay una medida real para medir su desempeño en el área, ya que existen sets de imágenes que son capturadas en entornos estáticos.

En esta tesis se propone una metodología para llegar al desarrollo de un clasificador de naranjas basado en ADD, usado para la segmentación de la fruta y su clasificación. El enfoque está dirigido para realizar una implementación sencilla en hardware, donde las arquitecturas de los modelos ADD se generan por medio de software.

1.4 Estado del arte

Por lo general los experimentos que se han realizado se usan fotografías (no video), de los cuales se extraen características, sin embargo las condiciones en las que se toman las fotografías son escenarios poco probables y poco prácticos para un problema real como lo es en un escenario industrial tal y como se ve en la figura 1.4 [21].

El canal de colores Hue, Saturation, Value (HSV) es pieza clave en la clasificación por color [22, 23, 24, 25] ya que se obtiene mucha información de uno de los canales ya que tanto H como S no son muy afectados por la iluminación.

La clasificación por umbrales fijos es difícil de lograr ya que se necesita mucha experimentación y desarrollo [25, 26], por lo que la implementación de un clasificador es necesaria.

Para la extracción de características para saber el estado de la fruta se utilizan ciertas características, tal es el caso de obtener desde el espacio de colores HSV de la imagen a procesar para obtener características sobre el color y la iluminación específica. De esto anterior obtener la media, y el histograma de los 3 canales correspondientes [22, 23], por otra parte el análisis es solamente para

1. Introducción

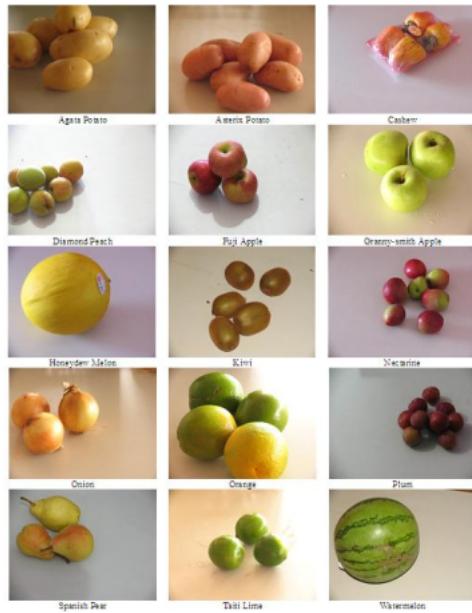


Figura 1.4. Condiciones poco probables. [\[Referencia\]](#)

detectar que se aprecia en mal estado, pero no se clasifica el color.

El obtener características complicadas de computar no siempre es la mejor opción para un problema como este, ya que el obtener una precisión de 90 % es aceptable, sin embargo el tiempo de procesamiento o el desarrollo de un algoritmo sofisticado puede ser costoso en recursos y tiempo de desarrollo [27]. Sin embargo, existen sistemas similares, los cuales están montados en FPGA, donde la velocidad de computo se mejora hasta 100 veces [28] además de también poder ser implementados clasificadores por arboles de decisión [29].

El uso de diferentes clasificadores para la clasificación de naranjas están los perceptrones multicapa (analizando histogramas sobre el canal RGB) [30, 22], arboles de decisión, Random Forest [22, 31].

Por otra parte, en la mayoría de los casos se tiene un set de datos muy reducidos, es por eso la elección de preparar un hardware industrial para tomar video, y obtener un set de datos más grande, a cambio de un poco de tiempo de desarrollo para el procesamiento del mismo y obtener fotografías

variadas.

Como se ha mencionado antes, un sistema como este, montado en un FPGA tiene resultados interesantes dado el poder de computo y la capacidad de funcionar en tiempo real [26], sin embargo el no utilizar algún tipo de memoria para almacenar las imágenes para un post-procesamiento, así como también el no utilizar clasificadores supervisados.

1.5 Objetivos

Objetivo general

- Desarrollar una implementación de hardware en FPGA capaz de clasificar naranjas con una precisión del 90 % con una rapidez de 1 naranja por segundo.

Objetivos específicos

- Implementación en FPGA de un clasificador por árbol de desición.
- Implementación en FPGA de un procesador de video extractor de características.
- Implementación de hardware auxiliar para mejorar el tiempo de desarrollo (microcontrolador).

2

Marco Teórico

En esta sección se abordan temas relacionados al trabajo realizado.

2.1 Inteligencia artificial

La inteligencia artificial en ciencia, es una rama que estudia desde un punto de vista de la ingeniería, la inteligencia en elementos artificiales, y que estos sean capaz de demostrar un comportamiento inteligente. Pretende construir sistemas que presenten un comportamiento que se pueda decir que es inteligente.

La inteligencia es un concepto difícil de definir, por lo que los consensos científicos no han establecido completamente su definición e identificación de un caso de inteligencia, osease, decir cuando algo es inteligente o no, sea artificial o no.

En el año de 1950, el padre de la computación Alan Turing, publicó un articulo relacionado a la computación y la inteligencia llamado “Maquinaria computacional e inteligencia”, donde

2. Marco Teórico

él argumentaba que siempre y cuando una máquina puede actuar como un humano, entonces es inteligente [32]. En el artículo se propuso una prueba llamada Test de Turing, que permitiría afirmar si una máquina es inteligente o no. El test es compuesto por 2 partes (individuos o cosas) donde se comunicarán por un medio informático, la primera parte debe ser un humano, y la segunda puede ser una máquina o un humano, la prueba consiste en que la primera parte defina si la segunda es una máquina o un humano. Si la primera parte identifica a una máquina como humano entonces esta máquina es considerada inteligente.

Hoy en día con los avances de la tecnología para que una máquina sea identificada como inteligente debe tener las siguientes características:

- Reconocimiento de lenguaje natural
- Razonamiento
- Aprendizaje
- Representación del conocimiento

El **aprendizaje automático** se refiere a la capacidad de la máquina para aprender cosas nuevas, de otro modo, no será capaz de adaptarse al medio o condiciones que exige el entorno. Las líneas de investigación que se revisan hoy en día se busca hacer que las máquinas capten generalizaciones a partir de ejemplos sacados del entorno. Como ejemplo, un niño pequeño al aprender a caminar, tiene caer muchas veces antes de calcular de manera exacta las instrucciones motrices para dar un paso, hasta lograr dar pasos con soltura.

2.2 Machine learning

Machine learning es una rama de la inteligencia artificial. Se define como la ciencia de hacer que sistemas de cómputo, aprendan y actúen como los humanos, mejorar su aprendizaje sobre el tiempo

2.3. Máquinas de aprendizaje supervisado

de forma autónoma, alimentándolos con datos e información que significan interacciones del mundo real [33].

Una máquina de aprendizaje es un modelo matemático o algoritmo capaz de predecir algún comportamiento a partir de datos que caracterizan el sistema (figura 2.1), los datos por lo general son clusters de características de que definen el comportamiento de un sistema, cada característica se administra en el modelo como una entrada. Después el modelo calcula una predicción y esta es su salida.

Oye pero esto mas que una estructura de un ADD, es la estructura de cualquier clasificador supervisado o no?

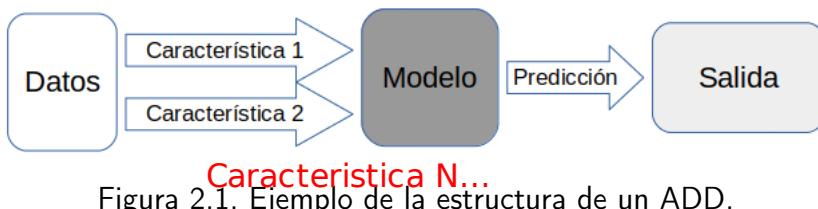


Figura 2.1. Ejemplo de la estructura de un ADD.

Las máquinas aprendizaje sirven para clasificar o realizar una regresión. En la figura 2.2 se puede ver el ejemplo de un set de datos, donde los ejes representan características de un sistema, por ejemplo: temperatura y humedad. Supongamos que se selecciona el 90 % de los datos para entrenar el modelo, el cual dependiendo del modelo se entrenará con uno u otro algoritmo. La linea punteada dibuja el resultado del entrenamiento el cual define la separación entre clases. Cuando un nuevo dato no conocido por el sistema ingresa al modelo, este será capaz de predecir la clase, entre más difícil sea de separar los datos se necesitan diferentes modelos.

Existen 3 tipos de máquinas de aprendizaje: máquinas de aprendizaje supervisado, máquinas de aprendizaje no supervisado, máquinas de aprendizaje por refuerzo. [\[Referencia\]](#)

2.2.1 Aprendizaje Supervisado

2.2.2 Aprendizaje no supervisado

2.3 ~~Máquinas de aprendizaje supervisado~~

2.2.3. Aprendizaje por refuerzo

En ellas la entrada y la salida deseada son administradas para el entrenamiento del modelo [34]. Los datos de entrada y salida son etiquetadas para la clasificación y con ellas se forma la base del entrenamiento para el procesamiento futuro de los datos tal y como se explicó en la sección anterior.

2. Marco Teórico

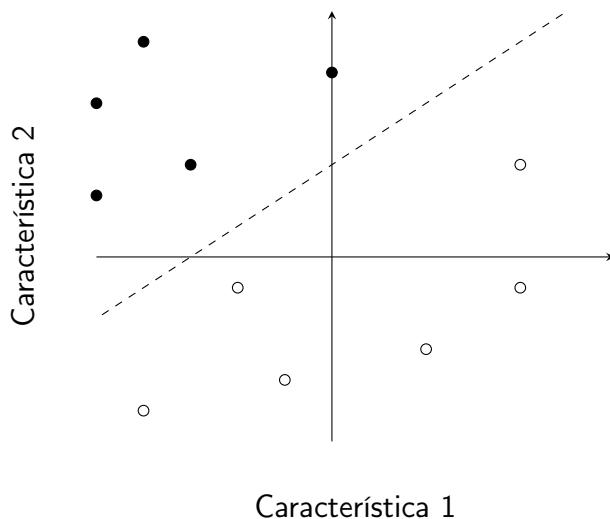


Figura 2.2. Ejemplo de clasificación en 2 dimensiones.

Algunos modelos de este tipo:

- Árbol de decisión (ADD) **
- Regresión Lineal
- Regresión logística
- Máquinas de soporte vectorial (SVM)
- Naive Bayes
- Algoritmo K-Vecinos más cercanos (KNN)
- Algoritmo de bosque aleatorio

2.4 Árbol de decisión

2.4. Árboles de decisión

2.4.1. Algoritmo IDE3

2.4.2. Algoritmo C4.5

Los Árboles de decisión (~~ADD~~) es una herramienta poderosa y popular para la clasificación y predicción [35]. Los ~~ADD~~ son estructuras de árboles enraizados, con hojas representando clasificaciones y nodos representando pruebas con características que definen las clasificaciones.

Un árbol puede ser entrenado dividiendo la fuente del set de datos de entrenamiento en otros pequeños basados en un atributo. Algunos algoritmos populares para entrenar un ~~ADD~~ son CART, ID3, C4.5 y Bosques aleatorios. Los ~~ADD~~ clasifican las instancias clasificándolas por el árbol desde la raíz hasta algún nodo hoja, lo que proporciona la clasificación de la instancia [36]. El algoritmo

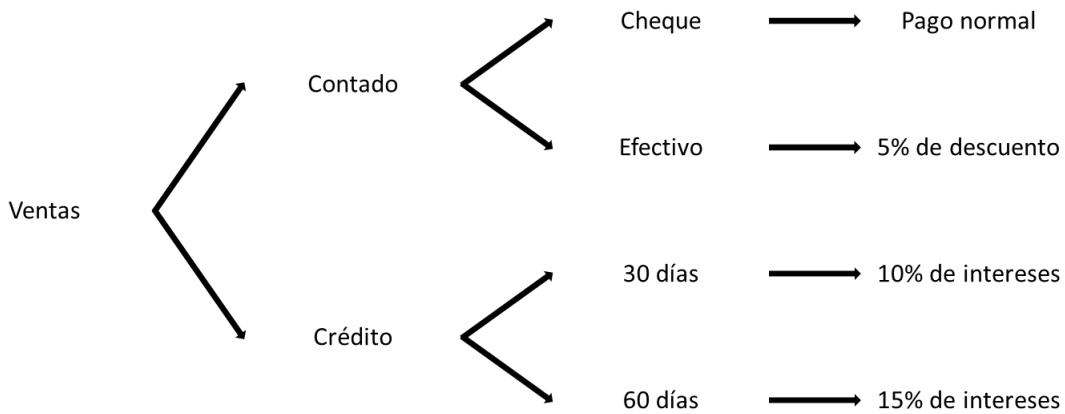


Figura 2.3. Ejemplo de la estructura de un ADD.

utilizado en este trabajo es el **algoritmo C4.5** el cuál es un derivado del algoritmo IDE3.

2.5 Algoritmo de entrenamiento IDE3

2.6 Algoritmo de entrenamiento C4.5

El algoritmo de entrenamiento C4.5 también llamado J48, es un algoritmo para construir ~~ADDs~~. Es el sucesor del algoritmo ID3 creado por J. Ross Quinlan, y es uno de los algoritmos para ~~ADDs~~ más populares [37].

Entropía

$$E(a) = \sum_{i=1}^{i=n} -p(c_i) \log_2 p(c_i)$$

a : Atributo

n : Número de clases del atributo a

c_i : la i -ésima clase del atributo

$p(x)$: La probabilidad de que ocurra la clase x

2. Marco Teórico

Ganancia

$$G(a, b) = E(a) - \sum p(a|b)E(a|b)$$

1. Buscar Entropia global
2. Buscar el atributo con mayor ganancia
3. Partir del atributo con mayor ganancia, e ir separando las clases por cada atributo de mayor a menor

2.7 Procesamiento de imágenes

algunas operaciones de procesamiento de imágenes son necesarias para la
Párrafo introductorio donde se dice que se ponen secciones necesarias.

implementacion del sistema de clasif. de naranjas. Entre las principales operaciones estan las transf. de color, las op morfológicas, etc, etc.

2.7.1 Transformaciones de color

En procesamiento de imágenes, es común que las imágenes estén en formato rojo-verde-azul (espacio de colores RGB), y se necesitan ser transformados en otros espacios de colores.

2.7.1.1. Transformación RGB a escala de grises

Una operación común para reducir los recursos computacionales en procesamiento de imágenes consiste en convertir una imagen RGB en una codificación diferencial como YC_bC_r , donde Y es la componente de luminancia, C_b y C_r son las componentes diferenciales de azul y rojo [38].

En los formatos digitales ITU-R BT-709, la luminancia puede ser calculado usando la ecuación 2.1, donde R, G y B son los componentes del espacio de colores RGB.

$$Y'_{709} = 0,2126R' + 0,7152G' + 0,722B' \quad (2.1)$$

Esta componente de luminancia corresponde a una imagen en escala de grises tal y como se muestra en la figura 2.5b para la imagen a color 2.5a.

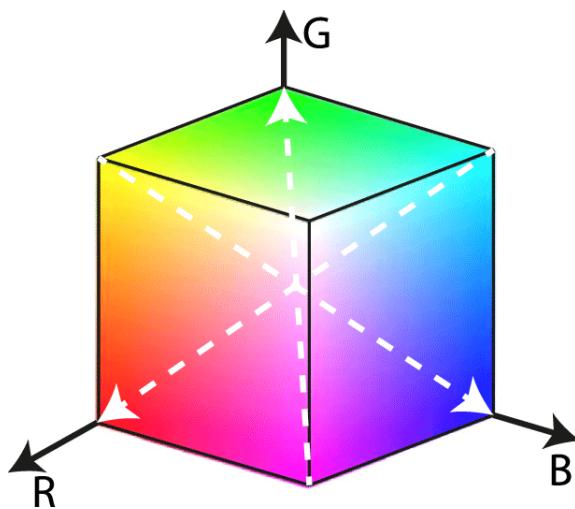


Figura 2.4. Representación axial del espacio de color RGB. [\[referencia\]](#)

2.7.1.2. Transformación RGB a HSV

Otro espacio de colores importante es Tono - Saturación - Valor (HSV). La particularidad de este espacio de colores es el campo del procesamiento de imágenes posible en él. Es una transformación no lineal del espacio RGB.

Las componentes R, G y B de un objeto en una imagen digital son todas correlacionadas a la cantidad de luz que refleja el objeto. Por otra parte las descripciones en términos de esos componentes puede hacer la discriminación del objeto complicada. En estas situaciones donde la descripción del color juega un rol importante, la descripción en términos del Tono (H) , saturación (S) y brillo (V) es comúnmente usada.

Esos valores son usados como ejes de coordenadas que proyectan el espacio RGB a lo largo de las diagonales blanco a negro (figura 2.4), resulta en un hexágono que forma la parte superior de una pirámide del espacio HSV (figura 2.6). H, es indicado como un ángulo alrededor del eje vertical. El color rojo es obtenido cuando $H=0^\circ$ o $H=360^\circ$, el verde cuando $H = 120^\circ$, y así sucesivamente [39].



Figura 2.5. Conversión de imagen a color a escala de grises.

2.7.2 Segmentación

Algo...

2.7.2.1. Binarización por umbral

Se trata de un método de segmentación que se rige por pixeles individuales. De una imagen por lo general en escala de grises se analiza pixel por pixel utilizando la ecuación 2.2, donde T es la función de binarización (por sus siglas en inglés de la palabra *Thresholding*), p como el valor del pixel y t como el umbral (por sus siglas en inglés de la palabra *threshold*).

$$T(p, t) = \begin{cases} 0 & \text{si } p < t \\ 255 & \text{si } p \geq t \end{cases} \quad (2.2)$$

El efecto de este método de segmentación da un realce de todos aquellos pixeles mayores al valor del umbral, tal y como se ve en la figura 2.7a. Por otra parte la condición de la ecuación 2.2 puede ser invertida, obteniendo la ecuación 2.3, donde el efecto se muestra en la figura 2.7b, la cual es una inversión de colores de la imagen 2.7a.

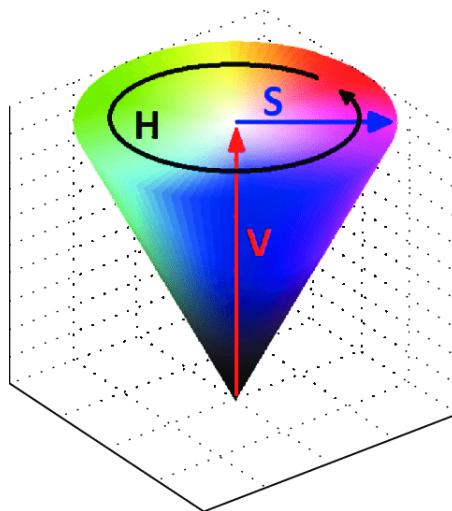


Figura 2.6. Pirámide de espacio de color HSV.[\[referencia\]](#)

$$T_{inv}(p, t) = \begin{cases} 255 & \text{si } p < t \\ 0 & \text{si } p \geq t \end{cases} \quad (2.3)$$



Figura 2.7. Binarización de imágenes usando un umbral $t = 127$

El cómputo de este método de segmentación es muy fácil de implementar en hardware ya que se trata de un comparador simple.

2. Marco Teórico

2.7.2.2. Contornos

~~2.8~~ Visión por computadora

~~2.7.2.3. Contornos~~

~~2.8.1 Histogramas~~

~~2.7.2.4. Histograma~~

~~2.9~~ FPGA

~~11 paginas de informacion relacionada al FPGA, se me hace EXCESIVO!~~

~~2.8~~

~~Un~~

~~En esta sección se hablará un poco de la historia, de los Arreglos de Compuertas Lógicas Programables llamadas en inglés Field Programmable Gate Array (llamadas FPGA de aquí en adelante), de como fué su introducción a la electrónica industrial, como es que funciona y que elementos lo componen para generar el conjunto que representan.~~

~~Es un circuito integrado...~~

~~Hablar de un FPGA, no es hablar de solo un chip más que se introdujo al área de la electrónica industrial, como pudo haber sido en sus inicios circuitos integrados (llamados ICs de aquí en adelante) con operaciones lógicas. Los FPGAs en el mundo de las simulaciones y en el prototipado, tiene un gran impacto desde hace algunas décadas. Los FPGAs las compañías de electrónicos las usan para poder generar su hardware digital, poderlo probar en una etapa temprana como los puertos de comunicación y los puertos de desarrollo que son usados durante todo el prototipado. Debido a su flexibilidad son muy requeridos en el área de la electrónica industrial.~~

2.8.1

Estructura de un FPGA

~~Un FPGA como todo circuito integrado está fabricado con silicio el cuál tiene la apariencia como cualquier otro circuito integrado. A diferencia de un IC convencional, los FPGAs son reprogramables, es decir, su funcionamiento interno o su propósito, puede ser modificado. Estos utilizan bloques pre-establecidos dispuestos de tal forma que la conexión entre ellos permiten ser variable, logrando así que el funcionamiento final pueda ser modificado dependiendo de como es que el usuario quiera esas conexiones.~~

El usuario puede configurar los FPGAs para implementar funciones que el mismo desee, sin tener que utilizar ICs adicionales para cambiar el comportamiento del circuito y por ende no utilizar otros dispositivos físicos que se pueden convertir en basura (circuitos de prueba que fallen por ejemplo). Por otra parte, el usuario deberá desarrollar ingeniería de computo digital en un software especificado para cada fabricante, este computo digital se desarrolla en archivos de texto en Lenguaje de Descripción de Hardware (llamado HDL a partir de aquí), el cual el software compilará y dará de salida un archivo tipo bitstream, el cual contiene toda la información necesaria que se utilizará para hacer las conexiones internas en el FPGA . Por ende, gracias a la explicación anterior los FPGAs son completamente flexibles ya que se puede cambiar su configuración cada que el usuario lo requiera.

~~En tiempos pasados solo había unos pocos ingenieros con grandes conocimientos los que podían trabajar con FPGAs pero conforme pasa el tiempo se van creando herramientas diseñadas en bloques las cuales permiten hacer un diseño en alto nivel haciendo más fácil generar los HDLs.~~

2.8.2 Utilización en la industria

Los FPGAs en la industria han sido pilar importante en el desarrollo de nuevos semiconductores (procesadores, Application-Specific-Integrated-Circuits o ASICs, etc.), ya que los FPGAs combinan lo mejor de los ASICs y de los sistemas en los que se basan procesadores ya que estos ofrecen frecuencias de reloj administradas por hardware, sin requerir altos volúmenes de requerimientos para poder fabricar un ASIC.

~~Para fabricar ASICs antes de que existieran los FPGAs significaba un verdadero problema, ya que no se contaba con algo flexible para primero hacer pruebas y después manufacturar la pieza de silicio. Las pruebas por lo general se realizan en tarjetas de desarrollo comerciales las cuales están formadas por un FPGA o más y tienen elementos que digitalizan todo el entorno del FPGA como pueden ser conversores analógicos-digitales (ADCs), ICs manejadores de puertos de comunicación (tal es el caso de un FTDI) y algunos conversores de voltaje, ya que por lo general un FPGA rápido, utiliza valores~~

2. Marco Teórico

digitales de 3.3V para ahorrar energía y calentamiento en el chip.

A diferencia de un procesador convencional que funcionan con una serie de instrucciones o pasos, los FPGAs funcionan en paralelo ya que están construidos solo por bloques lógicos o en más bajo nivel, solo están fabricados por compuertas lógicas las cuales están dispuestas de tal modo que se pueden fabricar series de bloques que ejecutan diversas tareas al mismo tiempo tal es la función de un procesador con más de un núcleo. Por otra parte, en los procesadores modernos existen hasta 8 o más núcleos, sin embargo en un FPGA se pueden hacer tantos bloques sean posibles físicamente dentro del FPGA que equivaldrían a núcleos que se dedican específicamente a una tarea en particular y no a tareas de distintos tipos lo cual es lo que mismamente desarrolla un procesador convencional de computadora.

Algunas veces, ciertos semiconductores cuando son manufacturados, una parte de ellos es un FPGA, generalmente las partes más problemáticas, más complicadas o aquellas que puede ser que con el tiempo necesiten ser modificadas. Esto se hace con el objetivo de que en un futuro el semiconducotor pueda ser reprogramado para hacer una actualización. En algunos casos, puede ocurrir que alguna parte del semiconductor esté en desarrollo, o se haya desarrollado anteriormente un sistema que es menos eficiente que uno que esté en pleno desarrollo, tal es el caso de algunas computadoras que tienen un BIOS que puede ser actualizado y por ende mejorar las capacidades de entrada y salida de una computadora.

La diferencia entre un ASIC y un FPGA es básicamente que un ASIC no puede cambiar su funcionamiento. Un ASIC como lo dice su nombre es un IC que tiene una aplicación en específico (por lo que su estructura no es flexible), su arquitectura es definida y finita. Los ASIC son ICs que son personalizables en su construcción, pero una vez manufacturados no pueden cambiar su arquitectura, por lo que para probar y hacer diseños de ASICs con otros ASICs es algo que los FPGAs han desplazado por completo. En un principio los FPGAs eran de baja potencia de computación, para desarrollar grandes volúmenes de diseño, pero hoy en día un FPGA puede llegar a tener un total de

500 millones de operaciones por segundo (un reloj de 500MHz) sin ningún problema y son capaces de almacenar diseños mucho más grandes comparándolos con los FPGAs de hace un par de décadas.

2.8.3.

~~2.9.1~~ Historia de los FPGAs

En 1984 Xilinx creó el primer modelo de FPGA, el modelo XC2046. Ellos no les llamaban como tal hasta que Actel popularizó el término en 1988. Desde entonces los fabricantes de FPGAs han incrementado las prestaciones increíblemente, tales son:

- Capacidad
- Costo
- Velocidad
- Consumo de energía

~~FPGAs contra ASICs~~

A principios de los años 80's las compañías que se dedicaban a fabricar ASICs vieron una buena oportunidad de negocio en la actividad. A mediados de los 80's muchas compañías se encontraban vendiendo ASICs, la competencia era dura en esos entonces. Los clientes buscaban al mejor postor, ya sea que el vendedor de ASICs mantuviera en sus productos a un costo bajo, alta capacidad del dispositivo y alta velocidad del semiconductor.

Cuando los FPGAs aparecieron, eran mucho más lentos, caros y pequeños (en capacidad) que cualquier ASIC en el mercado, ~~sin embargo prosperaron, lo cual resulta algo contradictorio~~. Los FPGAs prosperaron debido a que en la fabricación de ASICs había mucha ingeniería de por medio que podía ser reducida usando un ~~FPGA en el camino~~. Esta ingeniería de por medio resultaba algo cara, ya que se tenía que investigar, contratar personal capacitado y además algunas máquinas para la manufactura. Algunos fabricantes empezaron a usar los FPGAs, y se dieron cuenta que a cierto volumen de ventas, resultaba mucho más rentable usarlos en sus proyectos ya que se gastaban muchos recursos en la ingeniería. Cuando otros fabricantes empezaron a ver estos beneficios el volumen de ventas de ASICs ya no era justificable, volviendo así una tendencia el uso de FPGAs en el desarrollo

Traducción literal de "Field"

2. Marco Teórico

de semiconductores.

Aquellos fabricantes que con el tiempo se aferraron a la forma primitiva de fabricar ASICs empezaron a decaer, mucho del dinero invertido en algunos ASICs se perdían por errores de diseño, lotes mal manufacturados y carencia de clientes con demandas de altos volúmenes de dispositivos. Por otra parte, existían pequeños fabricantes los cuales generaban ingresos desarrollando pequeños volúmenes de ventas, solo por el hecho de usar FPGAs en sus procesos de diseño, por lo que ahorraban mucho en los errores de diseño que se pueden evitar en un FPGA, o que pueden ser reparados en un futuro.

~~FPGAs contra PALs~~

~~Los antecesores de un FPGA son los PALs~~

Antes de los FPGA existieron otro tipos de arreglos lógicos programables llamados PAL (Programmable Array Logic) los cuales eran programables con memorias EPROM. Estos fueron introducidos a principios de los años 80's. Algunos otros les llamaban PLA (Programmable Logic Array).

Que onda con esta IDENTACION?

Estos consistían en dos niveles lógicos de estructura, en la figura 2.8 se puede una estructura simple de un PAL, en el fondo podemos ver que están marcadas las entradas, en la parte izquierda se encuentra un arreglo AND programable, que produce operaciones AND de cualquier combinación incluyendo sus inversos. El bloque azul de la derecha corresponde a una compuerta de operación OR que completa la operación combinacional de la función lógica programada en el PAL. De esta manera es fácil de implementar máquinas de estados finitos en una versión muy primitiva.

Los PAL fueron muy eficientes desde un punto de vista de manufactura, ya que su estructura es bastante parecida a la de un arreglo de memorias EPROM. Gracias a esta arquitectura en un tiempo se tuvieron ideas nuevas sobre como expandir las memorias, y expandir los procesos de sus líneas de producción. El problema principal de esta arquitectura es bastante evidente, ya que para expandir

las funcionalidades de un PAL, este necesita convertirse en arreglos de PALs, dando como resultado una complicada programación por la cantidad exageradas de puntos que se debían de programar. Los arreglos de ANDs terminaban siendo demasiado grandes que la manufacturarlo el tamaño del dispositivo rebasaba a lo convencional.

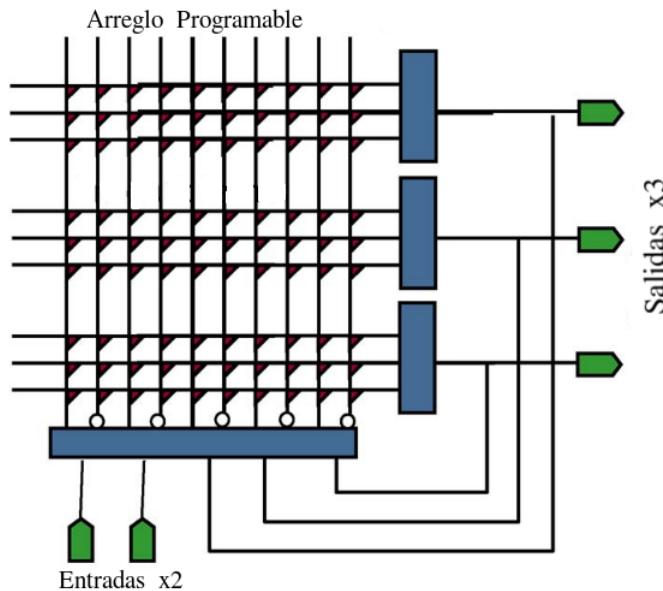


Figura 2.8. Arquitectura más simple de un PAL. [Referencia si la imagen no es de tu autoría]

~~El aumento de los arreglos en los PAL, causaban que los transistores usados en esos sistemas fueran más pequeños, por lo que la resistencia de cada uno bajaba, y por ende la capacitancia total aumentaba. Esto hacia que la velocidad y la potencia de consumo fueran opuestos en este tipo de arquitectura.~~

~~Principales fabricantes~~ Ya mencionaste arriba a Xilinx y a Altera (ahora Intel!!!)

~~Según la historia que se ha plasmado en la sección anterior~~, el primer FPGA fue creado por Xilinx la cual se encuentra hoy en día aún como vendedor más exitoso de FPGAs. Los principales

Ya lo dijiste (pagina 25).

2. Marco Teórico

vendedores de FPGA son:

(Ya dijiste que es SRAM y que es Flash y Antifuse?) - Por favor FPGAs basados en tecnología SRAM: no ocupes 10 paginas en explicarlo, procura SER Breve!

- Xilinx, Inc.
- Intel (Antes llamado Altera Corp.)
- Atmel
- Lattice Semiconductor

~~FPGAs basados en tecnología Flash y Antifuse:~~

- Actel Corp.
- Quick Logic Corp

Xilinx e Intel comparten el 60 % de las ventas.

2.9.2 Arquitectura general de un FPGA

La innovación de los FPGA fue la eliminación de la matriz de compuertas AND que se programaban, reemplazándolo por memorias de duración que fueron distribuidas entre cada nodo del arreglo de unidades lógicas. La arquitectura de un FPGA consiste en un arreglo de bloques programable los cuales se interconectan con unos switches programables entre unión de los bloques, los bloques están conectados entre si con estos switches, haciendo así el hardware reprogramable. El diagrama de la arquitectura de un FPGA convencional se puede ver en la figura 2.9, donde los bloques circulares son los switches programables, los bloques cuadrados son unidades lógicas y los bloques que se encuentran en las orillas son las entradas y salidas.

Un FPGA no cuenta con un procesador el cual siga una serie de instrucciones (software). El programador es quien diseña un circuito digital y es posible configurar un FPGA tan simple como una compuerta AND de 2 entradas o un complejo procesador multi núcleo para usarse como DSP. En [40] se puede ver que los FPGAs son idóneos para los DSPs, por lo que para aplicaciones de alta velocidad son una ventaja imprescindible.

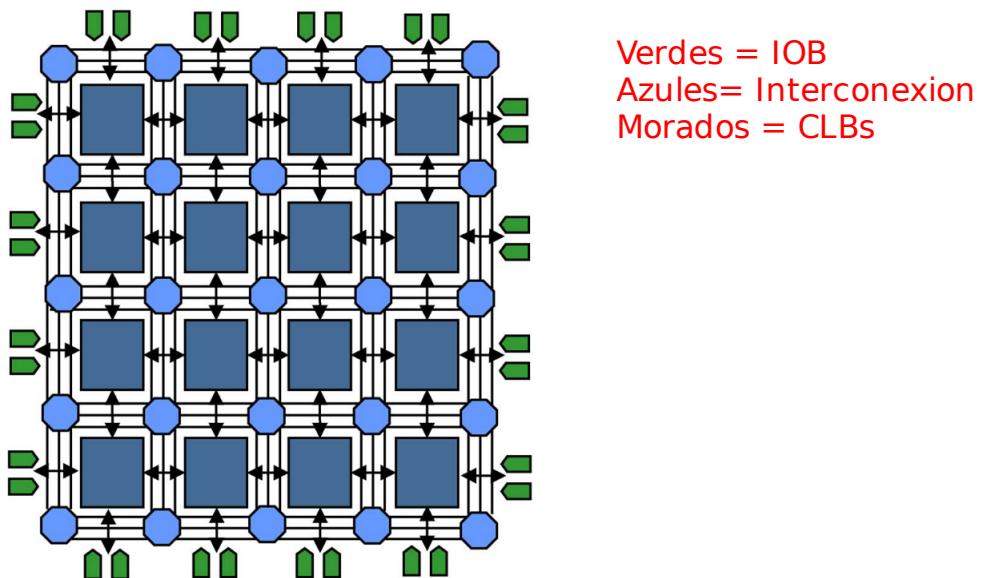


Figura 2.9. Arquitectura genérica de un FPGA.

[Referencia si la imagen no es de tu autoría]

2.9.3 Componentes y características

En general, hoy en día hay muchas otras formas de arquitectura de FPGAs sin embargo el en el modelo general la arquitectura de un FPGA está compuesto de los siguientes elementos:

CLB: Configurable-Logic-Block

La arquitectura de un CLB (figura 2.10) de un FPGA lo componen varios módulos lógicos más pequeños. Por lo general estos contienen LUTs, los cuales son una clase de memoria ROM (Read-Only-Memory), que son volátiles, ya que una vez programado perderá su valor cuando el FPGA sea desenergizado. Además cada CLB cuenta con un bloque de interconexiones, la cual tiene como tarea comunicar los bloques lógicos.

Slices

Algunos FPGAs en sus CLBs están divididos por slices (rebanadas) como se puede ver en la figura 2.11, un slice esta formado por dos módulos lógicos (o más dependiendo de su arquitectura), y entre

2. Marco Teórico

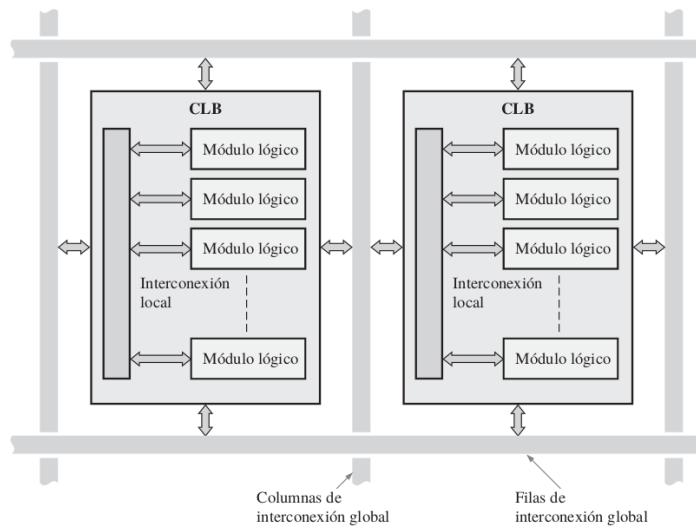


Figura 2.10. Arquitectura general de un bloques Configurable Lógico (CLB). [Referencia si la imagen no es de tu autoría]
ellos mismos la lógica combinacional se asocia dependiendo del problema en particular que se trate.

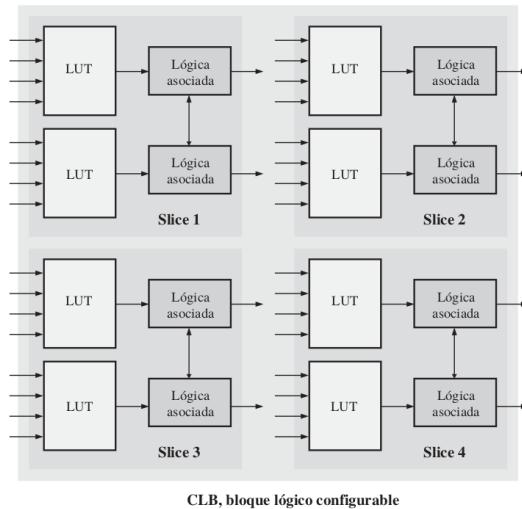


Figura 2.11. Arquitectura general de un CL formado con slices. [Referencia si la imagen Organización de un CLB en Capas (Slices) no es de tu autoría]

Módulo lógico

Un módulo lógico se configura para desarrollar circuitos de lógica combinacional por medio de las LUTs, o también como registros (localidades de memoria). En si, es un modelo reducido de memoria

y que en esencia una LUT realiza el mismo trabajo que una PAL.

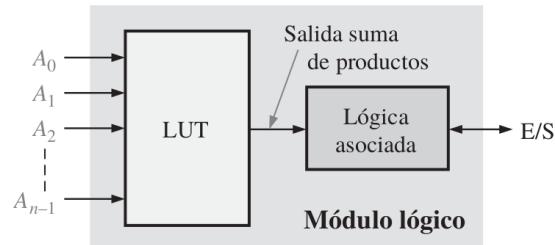


Figura 2.12. Arquitectura general de un módulo Lógico.

[Referencia si la imagen no es de tu autoría]

LUTs

Convencionalmente, la arquitectura de una LUT (figura 2.13) consiste en un arreglo de memorias, se tiene n como el numero de entradas, y una parte se comporta como un circuito combinacional convencional.

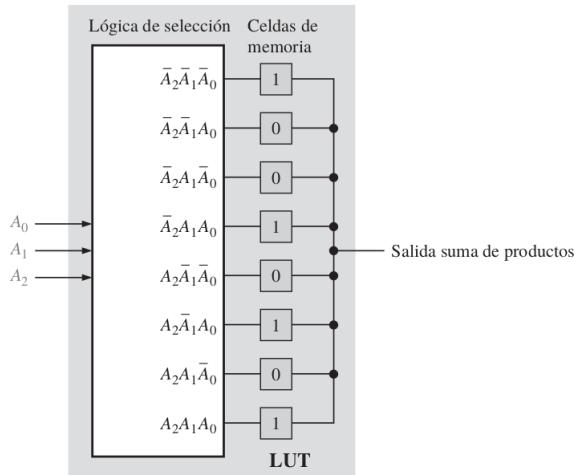
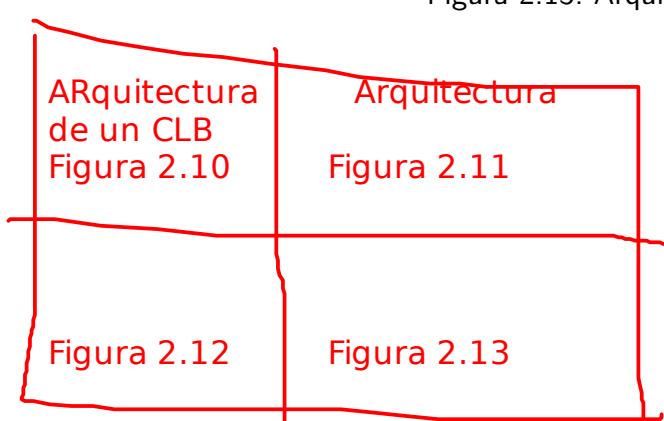


Figura 2.13. Arquitectura general de una LUT. [Referencia si la imagen no es de tu autoría]



En vez de poner 4 figuras, recomiendo poner una sola figura con 4 subapartados incluyendo las figuras 2.10 a 2.13



2. Marco Teórico

IOB: Input/Output Block

Reducirlo a una idea sin dar tanto detalle de su funcionamiento: UN IOB es un bloque cuya funcionalidad (E/S) es configurada por el usuario

Un IOB es un bloque de entrada o salida. A diferencia de un microcontrolador, la mayoría de los pines de un FPGA puede ser configurado de diferentes maneras. En algunas tarjetas de desarrollo algunos pines están reservados, por lo regular son entradas o salidas especializadas para ciertos periféricos, o señales de alta frecuencia, tal es el caso de las señales de reloj que utilizan los FPGAs. Los microcontroladores la mayoría de las veces cuentan con diferentes modalidades, tal es el caso que una misma terminal puede servir como un convertidor analógico digital (ADC), o incluso ser una salida o entrada digital, sin embargo esta función no puede ejecutarse al mismo tiempo.

A diferencia de un Microcontrolador, todas las terminales de un FPGA son digitales, las tarjetas de desarrollo, tienen algunas veces ASICs que digitalizan cierta información y se introduce por ciertos pines de un FPGA por lo que al dispositivo en si nunca se le introduce una señal analógica. Un IOB permite que el usuario defina como es que se comportará esa terminal, la cual puede comportarse como entrada, salida, entrada-salida ó triestado, buffer de entrada/salida de señal de reloj, de tipo latch y de tipo pull up/down.

El valor lógico "Z" se refiere a que en el nodo habrá una alta impedancia. Estos son usados por bloques con múltiples nodos para poder direccionar señales correctamente.

En la figura 2.14, se puede ver la arquitectura general de un IOB en donde se muestran los pines de entrada y salida (I/O), entradas de reloj y bus del modo el pin, esto conectado a su respectivo pad (o pin) del FPGA .

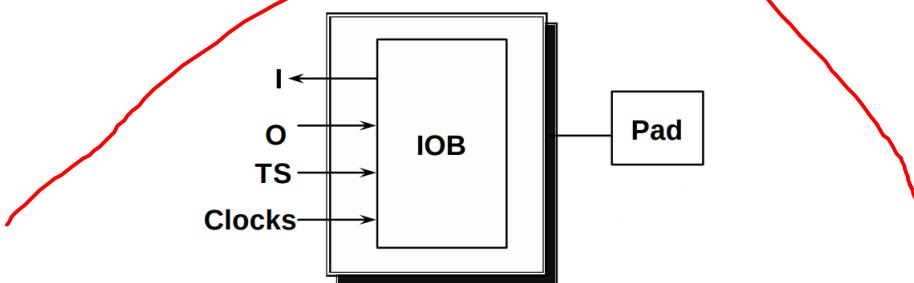


Figura 2.14. Esquema básico de un IOB.

~~Celda BSC~~

Una celda BSC (Boundary Scan Cell), es un bloque que está entre un IOB y una terminal física de un FPGA, estas celdas obtienen el valor inmediato de la terminal. Estas celdas están interconectadas en serie con todos los pines de un FPGA. Por lo regular se usan con el entorno de comunicación JTAG que se ve en una sección posterior de este trabajo. En algunos FPGAs se utilizan estos bloques como entrada de programación.

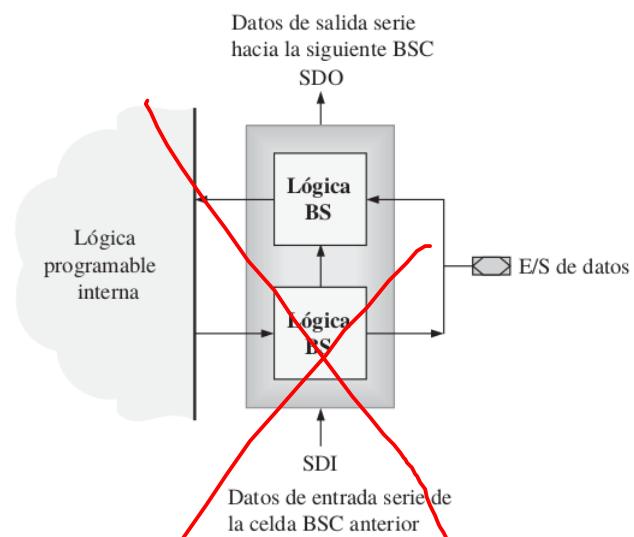


Figura 2.15. Arquitectura general de una BSC.

2.9.4 Lenguajes de descripción de Hardware (HDL)

Como hemos visto en secciones anteriores, los FPGAs son dispositivos flexibles, ya vimos por qué lo son, sin embargo ¿cómo es que una computadora hace todo lo posible por resolver los circuitos que deben realizarse a bajo nivel?. Existe software que se usa para calcular todas estas cosas, pero de todo ello depende un código escrito en un HDL. El código es interpretado por un compilador y posteriormente se resuelve toda la lógica y circuitos a bajo nivel hasta convertirse en un bitstream, el cual es un sistema de archivos que sirve para programar las celdas programables dentro del FPGA .

Existen unos cuantos HDLs, de entre los que más destacan son:

2. Marco Teórico

■ VHDL

Este HDL es un estándar en Europa. Su estilo tiene un parecido al lenguaje C, sin embargo muchos de los operadores son bastante distintos. Se suelen definir los bloques con las palabras reservadas "begin" y "end". Por lo general a los principiantes les resulta más fácil entender un código en este HDL.

■ Verilog

Este HDL es un estándar en América. Su estilo tiene aún mas parecido al lenguaje C ya que tiene prácticamente los mismos operadores. Se suelen declarar bloques interdependientes con el operador "@". Las sentencias igualmente se parecen mucho al lenguaje C.

■ System Verilog

Este HDL es bastante peculiar, y como se puede intuir es una clase de "extensión" del HDL Verilog. Este HDL suele no ser sintetizable, se usa en su mayoría para declarar simulaciones de bloques temporizados. Se utiliza mucho este HDL para la depuración de grandes proyectos. Sin duda es una herramienta muy buena de alto nivel que ayuda a depurar código rápidamente.

Cada HDL tiene su estilo y ninguno es mejor que otro. Estos 3 son de los más usados, sin embargo existen algunos más. Por estandarización mundial, la mayoría de los desarrolladores utilizan estos 3 HDLs.

Recordemos que este código describe circuitos y hardware, por lo que no es nada parecido a la programación de software. El flujo de datos no es secuencial, por lo que es un error común en los principiantes tratarlo como tal. Los ciclos "for" por ejemplo, no describen una secuencia, si no un patrón de repetición para módulos, o dicho de otra manera, reproducir circuitos iguales como un arreglo.

2.9.5 Implementación

La implementación de un diseño básicamente consiste en 3 pasos:

- **Síntesis**

Este proceso compila todo el circuito lógico diseñado en un HDL, en donde se revisa la sintaxis del mismo. Después se revisa si el HDL puede ser sintetizable, es decir, que se pueda construir con Hardware. Existen algunos elementos que se pueden describir con hardware pero no pueden ser construidos físicamente, como por ejemplo pueden ser pulsos temporizados sin base de alguna señal de reloj.

- **Mapeo**

Al ya estar resuelto el circuito lógico puede que algunos circuitos no quepan en CLBs únicos, por lo que el circuito se divide en sub-bloques y trata de encajar con la arquitectura del FPGA. Esto se hace en una computadora con un compilador especial, esta compilación es distinta de cada fabricante. Una vez realizado el mapeo, el usuario por medio del software puede ser capaz de realizar análisis de tiempos de señales de mapeo estático.

- **Posicionamiento y Ruteo**

En esta etapa se necesita una lista de red (NetList) la cual en ella, están descritos cada entrada y salida del módulo de mayor nivel y lo conecta a cada pin físico del FPGA. Una vez reestructurado todo el circuito en bloques pequeños que pueden recrearse en CLBs, el compilador busca la manera de colocar todos estos bloques de tal manera que ahorre espacio entre conexiones de bloques. Debido a que la arquitectura varía entre fabricantes e incluso entre modelos del mismo fabricante pero con modelos diferentes de FPGAs, el ruteo es específico para cada modelo, por lo que se necesita un compilador diferente para saber al final como obtener conexiones idóneas entre bloques o poder ahorrar espacio para no sufrir un sobrecargo de Hardware. Al final de este paso se obtiene un bitstream.

2. Marco Teórico

2.9.6 Ventajas de FPGA

Como hemos visto en las secciones anteriores, un FPGA tiene una estructura única y flexible. Gracias a estas características los FPGAs ofrecen muchos beneficios, como lo son:

1. **Rendimiento.** Dada la arquitectura de un FPGA y que puede ejecutar tareas de forma paralela, los FPGAs son candidatos excelentes para computar operaciones que realizan los procesadores digitales de señales (por sus siglas en inglés Digital-Signal-Processing llamados DSPs de aquí en adelante). ~~Algunos DSPs son logrados en computadoras, esto causa que esos DSPs sean lentos o con una velocidad de muestreo muy lenta, dado que una computadora funciona de manera secuencial.~~ Con un FPGA se pueden controlar entradas y salida a nivel de hardware propiamente, y esto ofrece tiempos de respuesta mucho más rápidos. En algunos osciloscopios se utilizan FPGAs en sus sistemas para realizar cálculos a velocidades increíbles.

2. **Tiempo en llegar al mercado.** Gracias a las ventajas de diseño que ofrece un FPGA el experto puede probar una idea o un concepto sobre la construcción de un circuito digital, y verificarlo por su propia cuenta, sin tener que llegar a fabricar o manufacturar el IC. ~~Muchos de los ICs convencionales que hoy se utilizan para diversas tareas (amplificadores operacionales, arreglos de flip-flops, etc.) tuvo que pasar algún tiempo hasta que un fabricante decidió manufacturarlo en masa para su venta. Por otra parte, a diseños o ideas de otros ingenieros se pueden refinar o mejorar en un FPGA y todo esto en solo unos cuantos días, ya que usando un FPGA se ahorra el tiempo de manufactura de un IC que está en prueba.~~

~~La comunidad que trabaja sobre FPGAs existen algunos núcleos prefabricados (llamados IP cores de aquí en adelante) como lo pueden ser filtros, procesadores de datos, etc. Usando estos IP cores, el experto puede ahorrarse algo de tiempo en el diseño, por lo que también la curva de aprendizaje disminuye y la complejidad de un proyecto también.~~

3. **Precio.** De primera mano comparar el precio de costo de fabricación de un ASIC contra la de un FPGA comercial es totalmente diferente. Para un ASIC se requieren varios ingenieros tanto para desarrollar la arquitectura interna como su manufactura. Las soluciones basadas en FPGAs son bastante simples ya que en una primera parte solamente se necesita la ingeniería detrás de la arquitectura del IC, sin tener que gastar aún en la manufactura. Una vez refinado por completo el modelo puede ser manufacturado, esto puede ahorrar bastante en scrap realizado en pruebas.

Los fabricantes de ICs por lo general hacen lo contrario, sin embargo la fuerte inversión inicial es justificable al manufacturar miles de chips que se venden por año. Por otra parte si solo se necesitan algunos cuantos ICs de un solo tipo un FPGA puede resultar en una solución abismalmente más barata.

4. **Fiabilidad.** Como sabemos hay muchas soluciones basadas en software que funcionan muy bien y que se encuentran en un entorno de programación, es obvio que estas soluciones están logradas en computadoras, las cuales como se ha comentado anteriormente cuentan con un procesador que ejecuta solo una tarea a la vez.

Los circuitos realizados internamente en un FPGA se pueden considerar que son implementaciones seguras, ya que no existen sistemas operativos de por medio, además el paralelismo ofrecen un mayor rendimiento en cualquier tarea. Como podremos recordar, un sistema operativo administra recursos de una computadora para poder realizar cálculos de diferentes programas, eso implica que en una computadora algunas veces existen conflictos entre uno y otro programa debido a que el ancho de banda puede variar para ambos. El hardware implementado en un FPGA es preciso y dedicado para cada tarea, todos trabajando juntos al mismo tiempo.

2. Marco Teórico

5. Mantenimiento a largo plazo. ~~Como se ha mencionado en todo este escrito,~~ los FPGAs son personalizables y actualizables en una implementación. No requieren el precio y el tiempo en rediseñar un nuevo chip. Algunas bloques con el tiempo pueden requerir una actualización, tal es el caso de los protocolos de comunicación los cuales con el tiempo requieren ciertos cambios, ya sean cambios de "timing", o de transferencia en la secuencia.

~~sistema modelado mediante~~

Una ~~máquina~~ con un FPGA puede ser continuamente actualizable, mientras que una basada en ASICs no lo puede ser debido a que una gran parte de la electrónica requeriría ser rediseñada y verificada.

2.10 Procesamiento de imágenes en FPGA

Creo haberte pasado un LIBRO para que complementes

El procesamiento de imágenes en un FPGA es bastante distinto de entornos computacionales, y dependiendo del procesamiento que se va a realizar se necesita cierto hardware. Por lo general

1. Procesar una imagen guardada en una memoria
2. Procesar una imagen codificada
3. Procesar una imagen escaneada

2.10.1 Ventajas y desventajas de FPGA en procesamiento de imágenes

Algo...

3

Desarrollo

empleados

En este capítulo se muestran los componentes y la solución para el proyecto. Se empieza hablando por la idea general, y el flujo de la solución ya que el proyecto está compuesto de diferentes temas, tales como ~~consideraciones~~ ~~mecánicas~~, eléctricas, electrónicas y computacionales. Despues se explican particularmente cada sección que compone el proyecto, para el entendimiento de cada una de las consideraciones de diseño tomadas.

3.1 Sistema propuesto

La idea general del trabajo se puede apreciar en la figura 3.1, el cual es un diagrama general del sistema, donde de manera general se compone de los siguientes componentes ~~con las siguientes actividades particulares~~: continuación.

~~descritos a~~

- **FPGA.** Encargado de las tareas relacionadas con el procesamiento de video y clasificación, ya que cuenta con un poder de computo superior a los demás componentes. ~~Este será comandado por el microcontrolador.~~ Menciona su ubicacion , y describe de manera breve las operaciones de proc. de imagen y de clasificacion de las que esta encargado

3. Desarrollo

- **Microcontrolador.** Encargado de todo el control de la maquinaria, tanto sensores como actuadores, encendido y secuencias. (**Conviene mencionar primero al MC y despues al FPGA, ya que es el "jefe" - que controla a todos**)
- **Computadora.** Utilizada para analizar y desarrollar el proyecto, sin embargo no forma parte del la maquinaria final.
- **Circuitos auxiliares.** Encargados de interactuar con el usuario y la interconexión entre dispositivos.
- **Actuadores.** Se encargan de mover la fruta (motores y actuadores).
- Camara de VIDEO. Se encarga de capturar



Figura 3.1. Diagrama general del sistema.

3.2 Maquinaria de flujo de naranja

Esta maquinaria es la encargada de mover toda la fruta por un circuito para ser procesada. En la figura 3.2 se muestra un diagrama general de la misma, en ella se aprecian todos los componentes de que componen el flujo de naranja, empezando por el embudo de alimentación y terminando en la zona de fruta clasificada. Las flechas, muestran el sentido y la dirección de el flujo de las naranjas y en el caso de la flecha con textura punteada se trata de una realimentación de fruta que puede que no haya sido procesada correctamente para dar una nueva oportunidad de ser clasificada.

** Lastima
que no se pueda
sacar una
fotografia
"aerea" de
dicha maquina, si
no seria un PLUS
interesante tanto
para tu tesis como
para el articulo

** De hecho sospecho
que si hay un bosquejo
de la figura que mencione
anterior ,es la tan mencionada
pero faltante Figura "X"

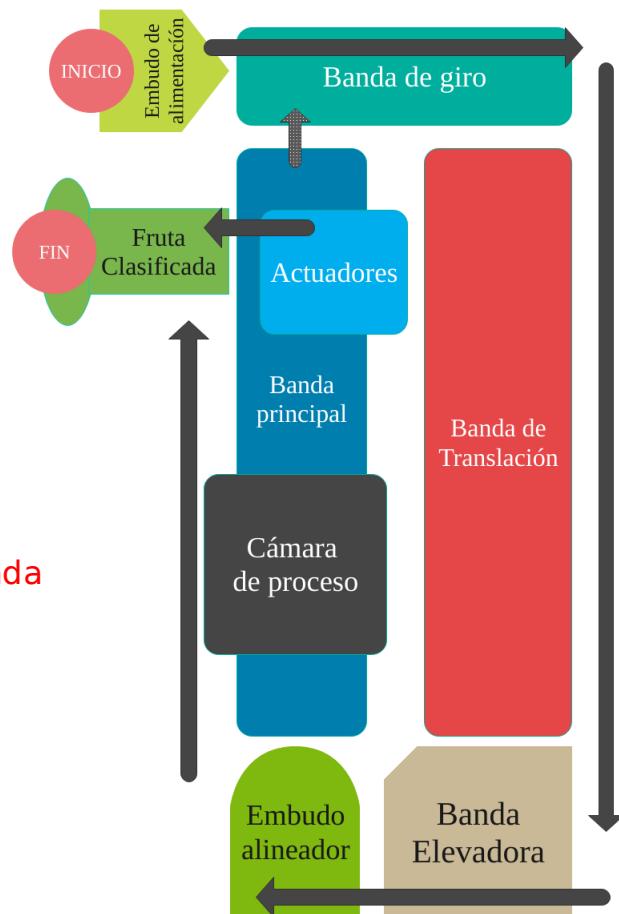


Figura 3.2. Diagrama general de maquinaria. **para el sistema de separacion automatica de naranjas..**
A continuación se explica de manera particular cada parte de la maquinaria.

Una subsección de un párrafo no es en general buena idea,
mas bien si los componentes se describen en un párrafo, debe
ir una lista con bullets

3. Desarrollo

3.2.1 Embudo de alimentación

Es el inicio de del circuito de flujo de naranja, en el se dispondrá la fruta, y gracias a su forma, la cual es un embudo con una pendiente negativa, deposita la fruta en la banda de giro. Su diseño no es final, y puede mejorar bastante, ya que puede ser más larga y mucho mas grande, dependiendo de la capacidad a la que se quiera clasificar. Es un elemento puramente mecánico y no cuenta con elementos eléctricos (figura X).

3.2.2 Banda de giro

Es una banda pequeña la cual se puede ver en la figura X, movida por un motor DC de 90V, la cual es controlada por un driver de 0.5HP por lo que la velocidad de giro puede ser controlada para suministrar un empuje diferente a la fruta. La transferencia del giro es por medio de una banda mecánica genérica. Al final de la banda cuenta con un riel el cual proporciona un giro de 90 grados para posteriormente la fruta caer a la siguiente etapa.

3.2.3 Banda de traslación

Es una banda larga la cual se puede ver en la figura X, movida por un motor AC monofásico. Este motor tiene una velocidad constante, sin embargo la banda cuenta con mucho espacio para la fruta y el motor es de carga pesada.

3.2.4 Banda elevadora

Es una banda con paletas la cual se encarga de subir la fruta a la siguiente etapa. Es movida por un Motor AC monofásico. El diseño de la banda permite subir alrededor de 5 naranjas por paleta (esto dependiendo del tamaño de la fruta), posteriormente la fruta cae en un riel que tiene la misma estructura de la banda, la cual sirve para suministrar la siguiente etapa.

3.2.5 Embudo alineador

Es una estructura metálica la cual tiene forma de embudo con una pendiente negativa, y unos rieles intermedios la cual se muestra en la figura X. Esta estructura es la encargada de hacer que la fruta sea alineada o colocada en hilera para posteriormente alimentar la banda principal, esto con el fin de evitar atascos en el flujo de la naranja. Es un elemento puramente mecánico por lo que no lleva motores.

3.2.6 Banda principal

Es una estructura con un riel compuesta de rodillos capaces de crear zocos de fruta, los cuales sirven para transportar fruta por toda la banda principal. El riel es movido por un motor DC de 90V, la cual es controlada por un driver de 0.5HP por lo que la velocidad de giro puede ser controlada para suministrar un empuje de diferente velocidad a la fruta. En la figura X se puede ver una fotografía de la banda, y en ella esta montada la zona de actuadores y la cámara de proceso que se describirán a continuación.

3.2.7 Cámara de proceso

Es una estructura metálica en la cual se contiene un ambiente con iluminación constante con LEDs de color blanco, toda la iluminación externa es eliminada en este lugar. En la entrada de la cámara de proceso se encuentra un sensor de presencia, para detectar si una naranja está por entrar, por otra parte en la parte central y superior se encuentra la cámara digital que se conecta al FPGA. Lo anterior se puede apreciar en la figura X.

3.2.8 Zona de actuadores

A lo mejor en vez de decir Zona, puedes decir "Bloque"

Se trata de una estructura pequeña montada sobre la banda principal (figura X), la cual alberga actuadores eléctricos los cuales empujan la naranja hacia la siguiente etapa, estos serán controlados por el microcontrolador mediante relevadores de estado sólido. La estructura es puramente ajustable,

3. Desarrollo

para colocarse cada actuador como sea conveniente en la posición conveniente.

3.2.9 Zona de fruta Clasificada

Es una zona que aun no se construye. Su función será recolectar la fruta ya clasificada, en diferentes rieles, se tiene un diseño preliminar el cual se puede ver en la figura X.

3.3 Flujo de fruta en maquinaria

En la figura 3.3 se muestra un diagrama de flujo el cual explica el proceso realizado por la maquinaria en general. Primeramente se recibe la fruta no clasificada, la cual debe haber sido lavada con anterioridad, puede tener polvo, sin embargo no puede estar manchada con lodo o algún agente que haga que cambie de color la misma, ya que esto puede afectar el rendimiento de la clasificación. Posteriormente, la fruta es suministrada al circuito de bandas la cual va a ser la encargada de alinear y acomodar la fruta para su procesamiento de video y decidir la clasificación que se esté tomando. Si la clasificación es válida, pasará a la zona de actuadores y al final del proceso, de otra manera la fruta será suministrada nuevamente al circuito de bandas para ser procesada nuevamente.

3.4 Sistema eléctrico/electrónico

~~En esta sección~~ se explican los componentes eléctricos y electrónicos y su conexión propuesta. El sistema eléctrico/electrónico general está compuesto por diversas partes las cuales serán explicadas a detalle en las siguientes secciones. Primeramente veamos un esquema general del sistema (figura 3.4) en donde se puede apreciar los componentes. Las flechas describen el flujo de señales que son emitidas o recibidas en el microcontrolador (MCU), el cuál es el que maneja los estímulos proporcionados por los elementos de control y medición, además maneja las salidas para estimular los componentes mecánicos y de procesamiento de video. Se puede apreciar también que algunos componentes solo se usarán como auxiliares en el desarrollo (DEBUG), por lo que no forman parte del sistema final y cuando estos son eliminados las flechas con textura punteada son utilizadas.



Figura 3.3. Diagrama de flujo de procesamiento de fruta en la maquinaria.

A continuación será explicada cada parte en particular.

3.4.1 Elementos mecánicos

~~3.4.1.1.~~ Actuadores

Como se comentó en la sección anterior, son actuadores eléctricos de AC controlados mediante actuadores de estado sólido (incluidos en la etapa de potencia), esto para mejorar la velocidad de conmutación ya que necesitan ser rápidos.

~~3.4.1.2.~~ Motores

Las características de los motores son los siguientes se pueden ver en la tabla 3.1. Los drivers de motores DC cuentan con rampa de aceleración para una transición suave y el contacto es controlado por un relevador mecánico activado por el MCU.

En vez de tener subsubsecciones, sugiero que ponga una lista de bullets

- * Actuadores
- * Motores
- * Etapa de Potencia

3. Desarrollo

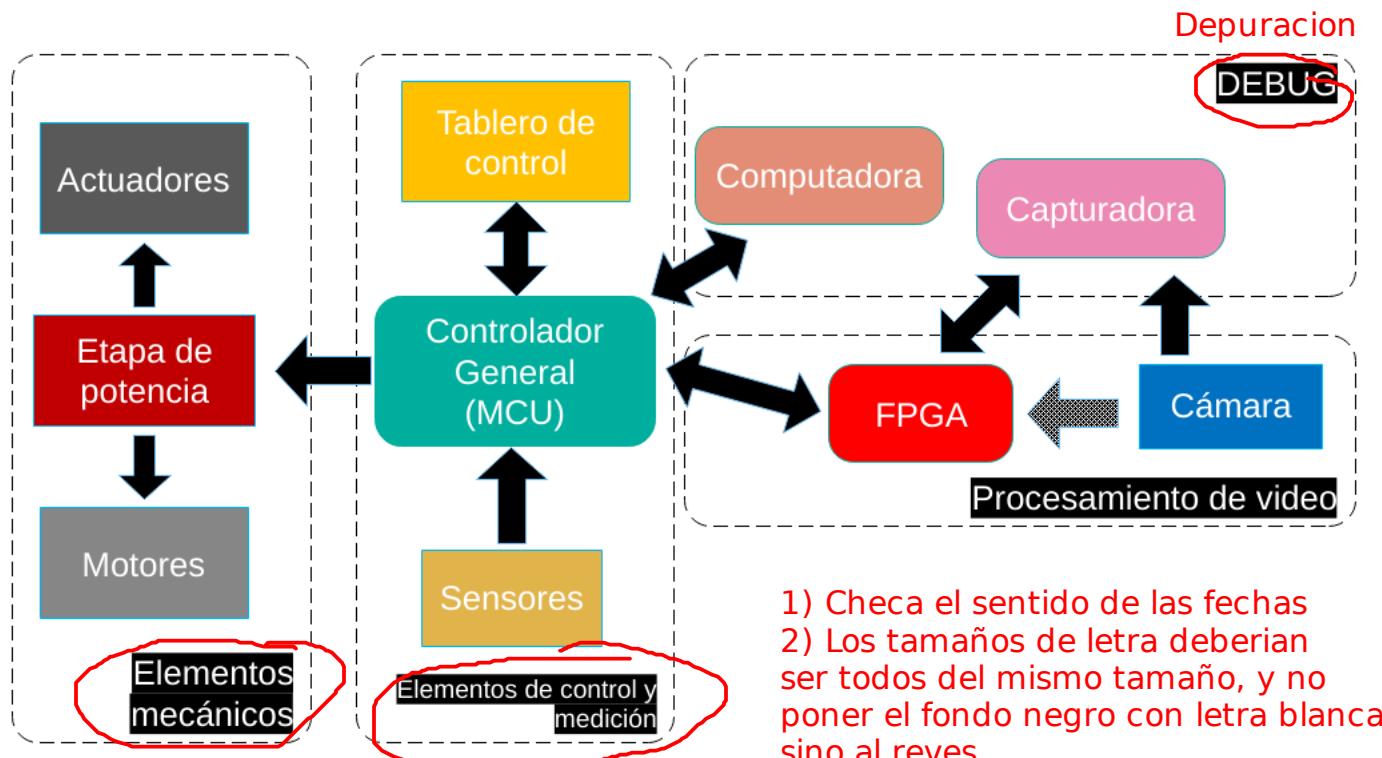


Figura 3.4. Diagrama eléctrico/electrónico general.

Tabla 3.1. Características de motores

Posición	Características eléctricas	Controlado por
Banda principal	90VDC 0.5HP	Driver de contacto seco
Banda de giro	90VDC 0.5HP	Driver de contacto seco
Banda elevadora	120 VAC 1HP	Relevador mecánico activado por MCU
Banda de translación	120 VAC 1 HP	Relevador mecánico activado por MCU

~~3.4.3. Etapa de potencia~~

La etapa de potencia está encargada de el accionamiento de los elementos anteriores, sus conexiones se pueen observar en la figura 3.5, en donde se puede ver el uso de los relevadores mecánicos y de estado sólido.

3.4.2 Procesador de video

- Cámara digital Es requerida cualquier camara digital que pueda capturar
Es capaz de grabar a una resolución de 1080p, a 60 fps, con salida de HDMI. Está colocada video usando una resolucion de 1080p...

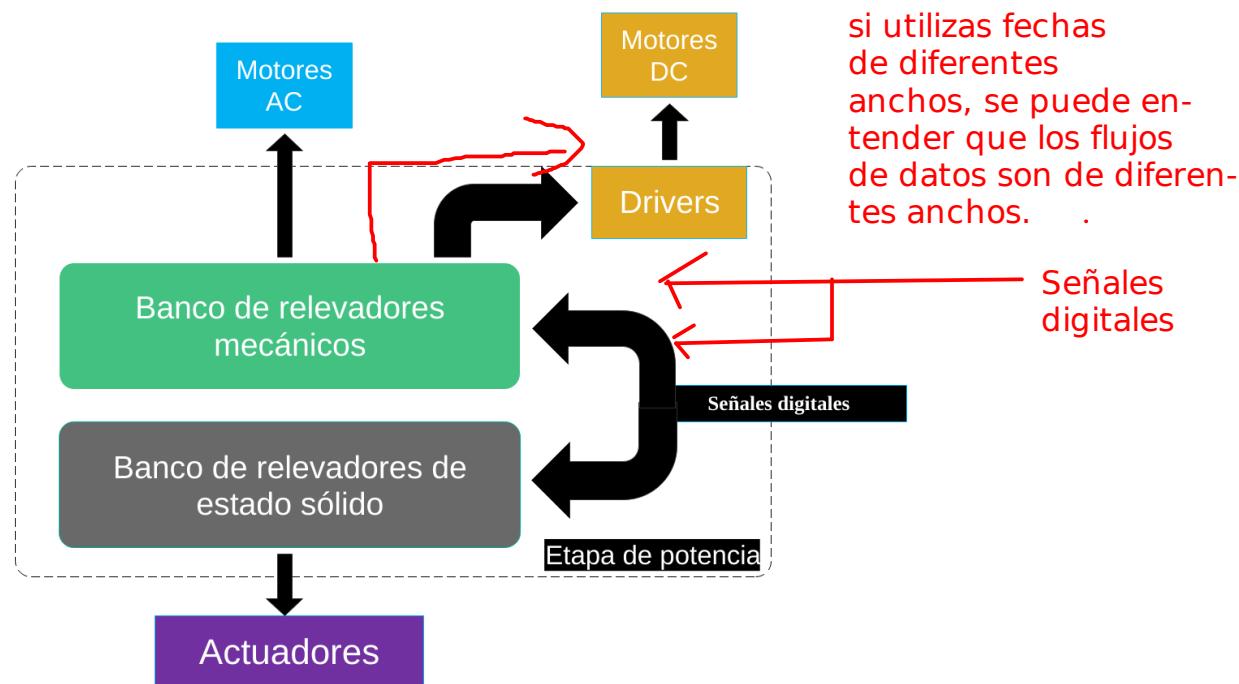


Figura 3.5. Diagrama de conexión de elementos mecánicos

en la cámara de proceso y alimenta con video al FPGA.

■ **FPGA**

Se trata de la tarjeta de desarrollo Industrial Video Processing Kit (IVPK) de Xilinx, con un FPGA Spartan-6 de nivel industrial, además cuenta con una tarjeta I/O de video con puertos HDMI. Su capacidad es mayor a la requerida en el proyecto. Cuenta con su propia fuente de alimentación externa. Actualmente está descontinuada.

3.4.3 Elementos auxiliares

Estos elementos no forman parte del proyecto final, sin embargo son parte fundamental del desarrollo de etapas computacionales y para conocer algunos datos de la maquinaria.

■ **Capturadora (En el mismo renglon la descripción)**

Utilizada para la captura de los datasets necesarios para generar los modelos de ADD.

■ **Computadora (En el mismo renglon la descripción)**

3. Desarrollo

La computadora es utilizada para recabar cierta información útil, tal como la frecuencia del sensor de sincronía y el resultado de la clasificación.

3.4.4 Elementos de control y medición

Estos elementos serán los encargados de la sincronización y activación de secuencias de encendido de la maquinaria.

En una lista con bullets
* Tablero de Control
~~3.4.4.1.~~ * Sensores
* Controlador General

Este se encarga del control general del sistema y de lamparas de señalización para el usuario el cual se puede ver en la figura X. Cuenta con los siguientes componentes:

ENTRADAS

- Botón de Inicio
- Selector de 3 posiciones.
- Botón de paro de emergencia
- x2 Botones de propósito general

SALIDAS

- x3 Lamparas de señalización.

Para el proyecto solo se utilizaron los botones de inicio y paro. Los demás componentes están disponibles para desarrollos futuros a este trabajo.

~~3.4.4.2.~~ **Sensores**

Los sensores en este proyecto solo son 2, los cuales son de proximidad fotoeléctrico Infrarrojo (E18-D80NK), su funcionamiento es detectar si existe un obstáculo frente al sensor, el obstáculo debe estar dentro del umbral de distancia el cual es ajustable desde 1cm hasta 100cm, si existe el obstáculo dentro del umbral, entonces el sensor mandará una señal de 5V. Los sensores tienen los siguientes propósitos:

1. Detección de naranja.

Está colocado dentro de la cámara de proceso, justamente en el primer zócalo donde entra la fruta. Su actividad principal es detectar si en el zócalo de entrada hay una naranja, el zócalo de detección está desplazado por 3 zócalos de donde se encuentra el objetivo central de la cámara digital, esto con el objetivo de no interferir en la imagen suministrada al procesador de video.

2. Sincronía con banda principal.

Está colocado en la banda principal cerca de la zona de actuadores, aunque puede ser instalado en cualquier parte de la banda principal. Este se encarga de sentir cada zócalo de naranja, y con ello obtener una onda cuadrada, la cual servirá como disparador para el inicio de una clasificación.

En conjunto los sensores definen en que momento procesar la imagen obtenida por la cámara y si es necesario realizar el proceso, ya que si no hay una naranja en el objetivo del zócalo no es necesario realizarlo y así ahorrar recursos computacionales.

~~3.4.4.3.~~ Controlador General

El MCU usado para el proyecto es un ATMEGA 2560, el cual cuenta con los GPIO necesarios con los pins de interrupciones necesarias. En la figura 3.6 se puede apreciar un diagrama con los tipos de conexiones utilizadas.

Se optó por un puerto SPI en el FPGA debido a la cantidad reducida de GPIOs en la tarjeta IVPK, debido a que el MCU es el maestro de la conexión, se pueden disparar instrucciones desde el microcontrolador hacia el FPGA, para iniciar una clasificación, esperar un momento y recibir la respuesta.

En la figura 3.7 se puede ver un diagrama de flujo del programa que contiene el MCU. Es un diagrama de flujo reducido, debido a que faltan la parte de la comunicación con la computadora y el cálculo de frecuencia la señal de sincronía, sin embargo es un diagrama muy cercano a la realidad.

3. Desarrollo

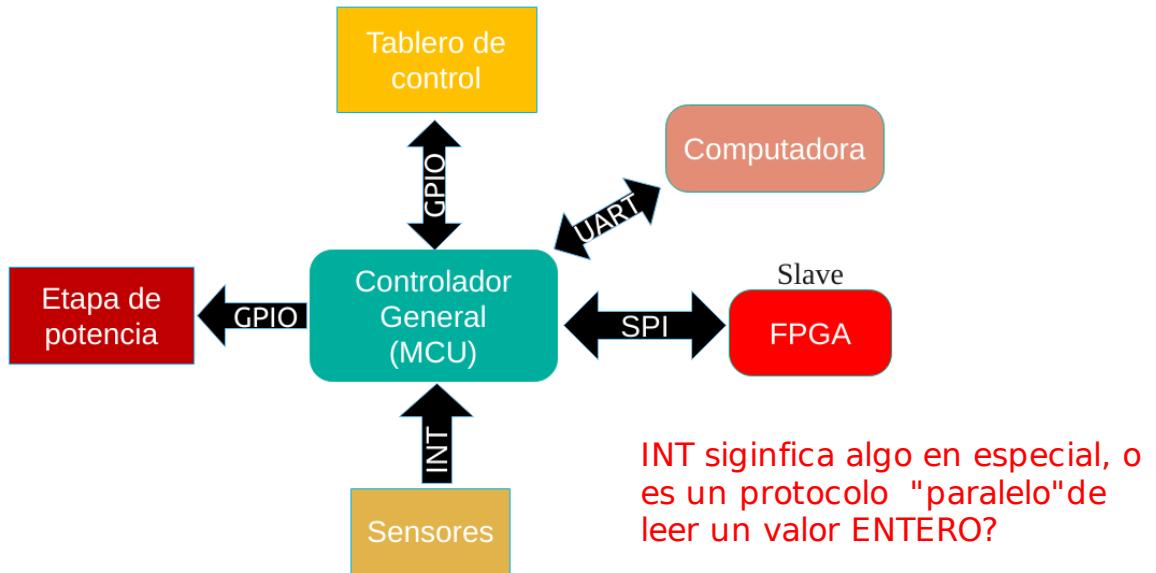


Figura 3.6. Configuración de terminales en MCU **y los protocolos de comunicación empleados**

En una primera instancia se detecta el pulso de inicio y paro para dar a lugar las secuencias de encendido y apagado, ya que la maquinaria es de alta potencia se necesitan encender motores a tiempos espaciados, ya que de otra manera se producen transitorios de alta corriente que pueden dañar la instalación eléctrica. Después se detecta el pulso de sincronía, y al ser detectado, se sensa si hay una naranja en el zócalo de la cámara de proceso, de haberlo, se manda al FPGA una instrucción de procesamiento, se espera un pulso proveniente del FPGA para saber que ya ha acabado el procesamiento, esto debe de durar alrededor de 64ms ya que es la tasa de refresco que tiene la cámara es de 16ms por cuadro y el retraso es de 4 cuadros posteriores, y debe de analizar la imagen en el momento justo. Al recibir el pulso **de termino**, el MCU recibe el resultado de la clasificación por medio de un byte, el cual contiene la información de tamaño y color de la fruta. Dependiendo de la respuesta, se computa el movimiento de empuje para el actuador.

3.5 Modelo ADD de segmentación

En esta sección se describen los pasos necesarios para construir el ADD de segmentación hecho por computadora y en una sección posterior se explica como se realiza este modelo en un módulo

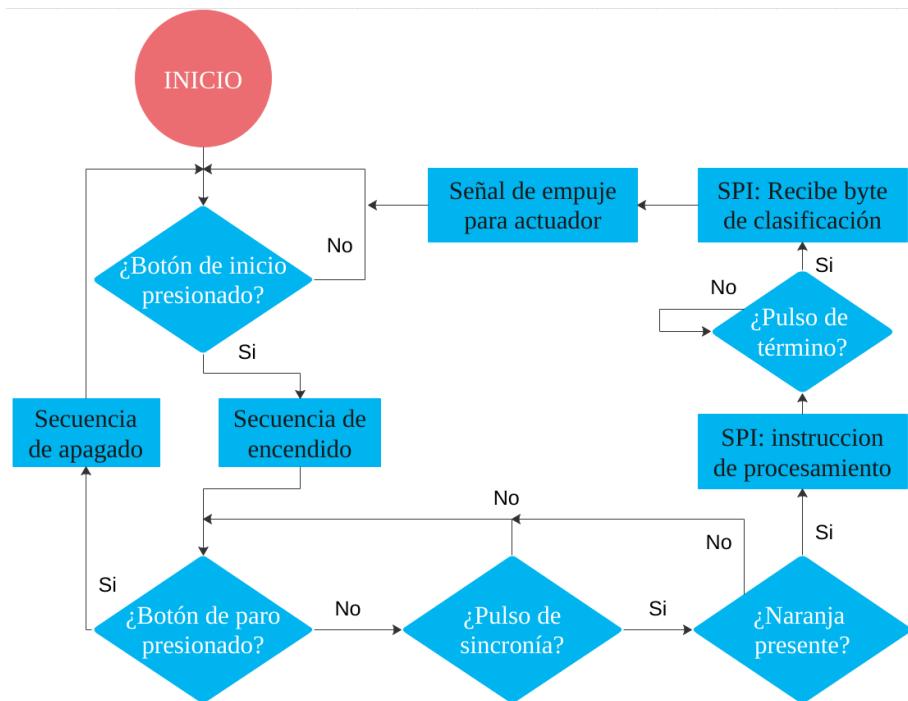


Figura 3.7. Diagrama de flujo del programa del MCU

* Ver artículo enviado

sintetizable de HDL.

3.5.1 Captura de video

Para el desarrollo del modelo, primeramente se obtuvieron los datos para poderlos manejar y alimentar el modelo. Para ello se accionó la maquinaria con fruta en los circuitos de bandas sin activar los actuadores, para que estas giraran entre el circuito de bandas, con esto y los elementos auxiliares vistos en la sección anterior y con ayuda de la cámara se tomaron videos de la fruta pasando por el objetivo de la cámara para de ellos obtener los datos necesarios para fabricar el modelo.

3.5.2 Captura de fotogramas de interés

** Estrictamente hablando, no capturas una foto ya que esto da a entender que la guardas para su posterior proc.*

Dado que en no todos los fotogramas de los videos capturados se encuentran imágenes de la fruta, y además solo se tomará una fracción del fotograma para realizar la clasificación se decidió construir un algoritmo de detección de la fruta para que de esta manera se lograra obtener un dataset

** Fotorama => Imagenes

3. Desarrollo

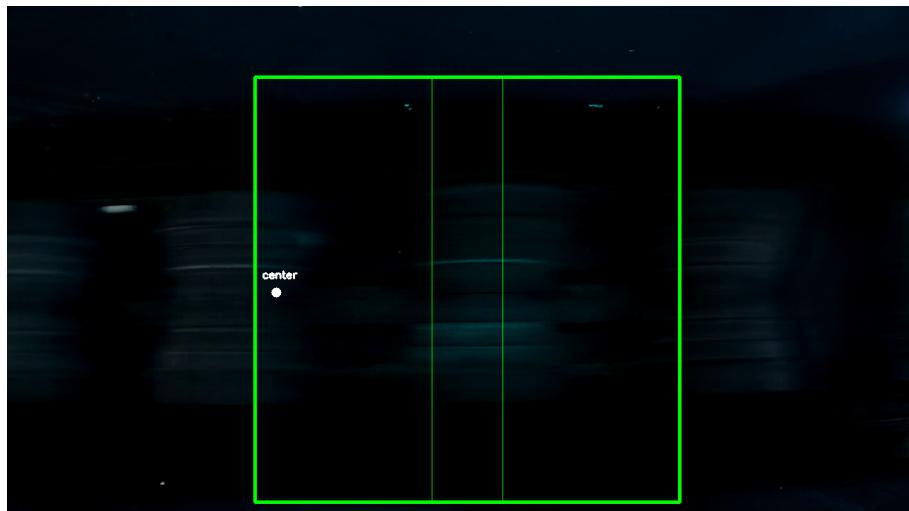


Figura 3.8. Escena de cámara dentro de cabina. **y recuadro con la region de interes.**

de imágenes de fruta.

Primeramente se decidió la región de interés (ROI), la cual se puede ver en la figura 3.8 como un recuadro de color verde en la parte central. La región tiene una medida de 600x600 pixeles en una región conveniente del fotograma ya que en ella se encuentra el zócalo de captura de manera centrada. Dentro de ROI, se encuentran unos límites centrales las cuales son unas barras verticales de color verde. Entre estos límites se tomará el fotograma una vez que sea detectado el centroide de la fruta.

Para la segmentación de la naranja simplemente la imagen de entrada para cada cuadro fue convertida a escala de grises y binarizada con un umbral. Posteriormente encontrar el contorno con mayor área y llenar el contorno, con esto se obtiene una máscara tal como se ve en la figura 3.9b. Con ésta máscara, se calcula el centroide y el área de la misma, si el área es mayor **a 10000 pixeles**, quiere decir que se encuentra una fruta dentro de ROI. Por otra parte, se calcula si el centroide en el eje horizontal está entre los límites centrales, y de esa manera se toma la captura de la imagen tal y como se ve en la figura 3.9a. Debido a las diferentes velocidades de la banda las capturas pueden tener una difuminación tal y como se ve en la figura 3.9, sin embargo esto no afecta en la toma del set de imágenes.

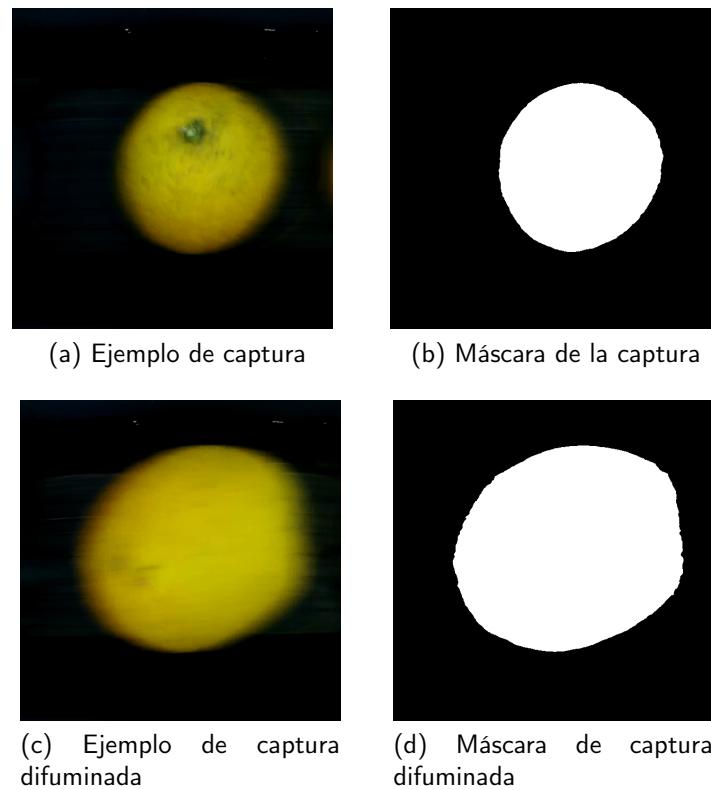


Figura 3.9. Captura para el set de datos y obtención de la mascara. Difuminación por movimiento.

Ejemplo de ruido de movimiento (a) y (c) ocasionado por el traslado a altas velocidades de la fruta objetivo. Mascada de captura obtenida (b) y (d) en ambos casos

3. Desarrollo

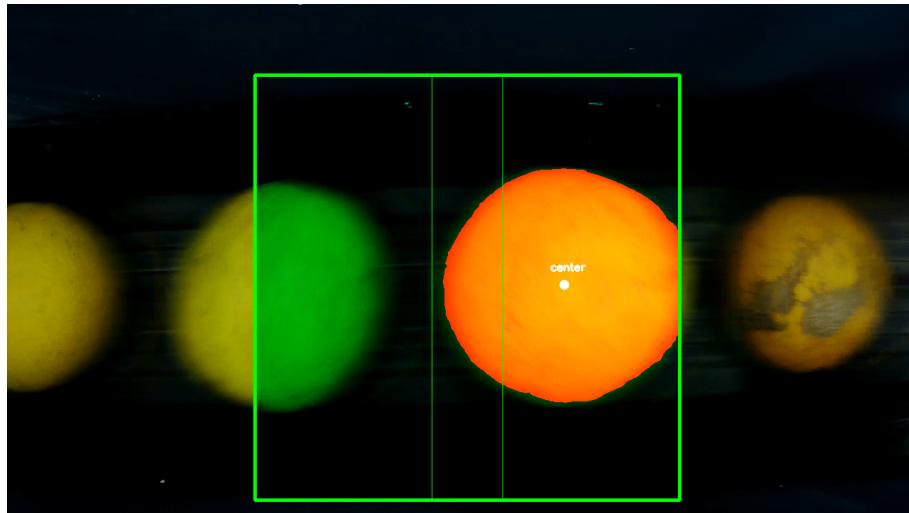


Figura 3.10. Proceso de captura de imágenes. **includiendo los límites centrales para ubicar el centro de la ROI**

En la figura 3.10 se puede ver una imagen completa del proceso antes descrito, donde la parte enrojecida es la máscara, además se encuentra indicado tanto el ROI, como los límites centrales y el centroide de la máscara.

3.5.3 Captura de data set de pixeles

generar

Una vez obtenidas las imágenes visto en la sección anterior, estas se utilizan para realizar el data set de pixeles. En la figura 3.11 se muestra un diagrama de flujo el cual explica el proceso de la captura del data set. Para cada imagen obtenida, se obtiene la segmentación, la conversión a espacios de color a escala de grises y HSV. Una vez obtenidos, cada canal es separado y guardado como imágenes independientes, para ser convertidas de matrices a vectores planos verticales. Por último, estos vectores son apilados horizontalmente para crear una matriz donde se tiene cada canal como una columna. Realizar esto para cada imagen e ir apilando los resultados en una matriz donde se guarda el resultado de todos las imágenes, para al final remover todos los duplicados. Como paso final, se eliminaron todos los pixeles los cuales mostraran los mismos valores en los canales de color pero diferente segmentación, por ejemplo, en la figura 3.11, para el pixel 1 y 4 no son válidos, debido a que en todos los demás canales tienen los mismos datos, sin embargo en la segmentación son diferentes.

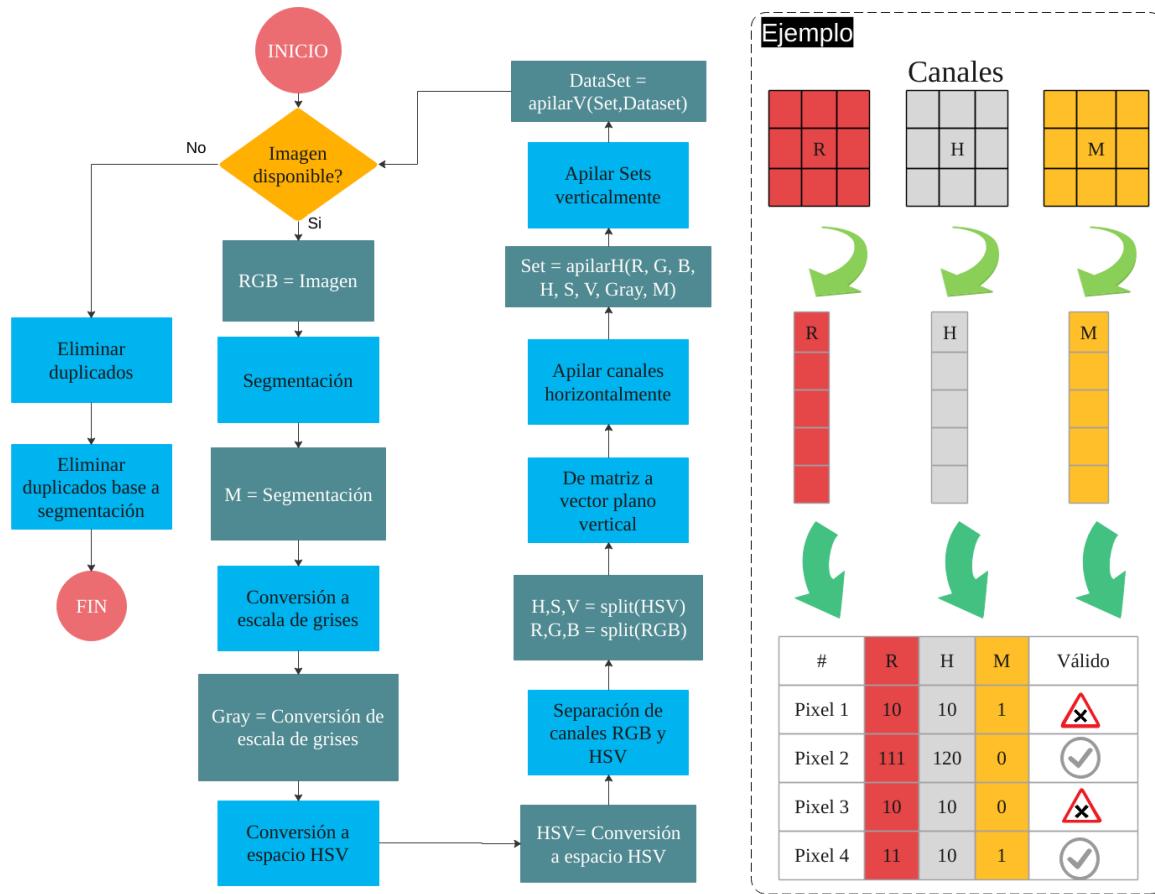


Figura 3.11. Diagrama de flujo de captura de dataset de pixeles.

Esto es un problema grave y no existe una separación espacial entre los datos. Estos pixeles son eliminados del dataset para mostrar una tendencia entre lo que es un pixel que pertenece a la fruta y uno que no.

De última instancia, se procedió al entrenamiento en WEKA, lo cual se realizó con un modelo del tipo J48 de ramas binarias con un costo $C = 0,25$.

3.6 Modelo ADD clasificador de color

Para el desarrollo de este modelo, se utilizó el dataset de imágenes anterior. Primeramente se analizó el dataset de imágenes y se clasificó de manera manual, separando entre las clases "naranjas

3. Desarrollo

"verde". Una vez clasificadas se realizó una captura de histogramas que se verá en las secciones siguientes.

3.6.1 Análisis de color

Dada la naturaleza del canal H, el cual describe el color del pixel, dependiendo de su iluminación (ya que el negro se interpreta como ausencia de luz y blanco como presencia de todos los colores) se decidió realizar un pequeño análisis de colores el cual se puede ver en la figura [3.12](#) en donde se muestran 4 muestras de fruta, con 4 casos posibles diferentes donde 2 de ellas están muy bien definidas entre su clase([3.12a](#) y [3.12d](#)) y otras cerca del umbral de lo que se puede considerar como naranja o verde ([3.12c](#) y [3.12d](#)).

En cada una de ellas, se analizó un histograma solo sobre los pixeles que son segmentados. Ya que se hace muy complejo el análisis de un histograma completo como arquitectura sintetizable, se decidió optar por uno de 5 bins. En cada una de las muestras los acompaña un histograma completo, donde las barras verticales son las divisiones, y las otras barras el resultado del histograma, también lo acompaña la representación de un histograma de 5 bins, y una representación de la muestra en el canal H. Como se puede notar, en la representación del canal H, el color verde cuenta con una intensidad más alta, que la del color naranja. Además se puede ver un patrón diferente del histograma de 5 bins para cada caso, por lo que el canal H es un buen candidato para la separación de datos, sin embargo se optó por realizar por un histograma de 5 bins de todos los canales disponibles, ya que se vio que de esta manera es un buen método, por otra parte, si los datos llegasen a ser irrelevantes, el algoritmo C.45 se encargará de eliminar todos aquellos datos que no aporten a la separación entre clases.

3.6.2 Captura de data set de histogramas

Debido al tamaño distinto para cada muestra, se notó que un problema fuese el rango de valores que toman los histogramas por lo que se decidió tomar histogramas normalizados. La norma se tomaría a partir de el total de pixeles segmentados. En la figura [3.13](#) se puede ver un diagrama de

3.6. Modelo ADD clasificador de color

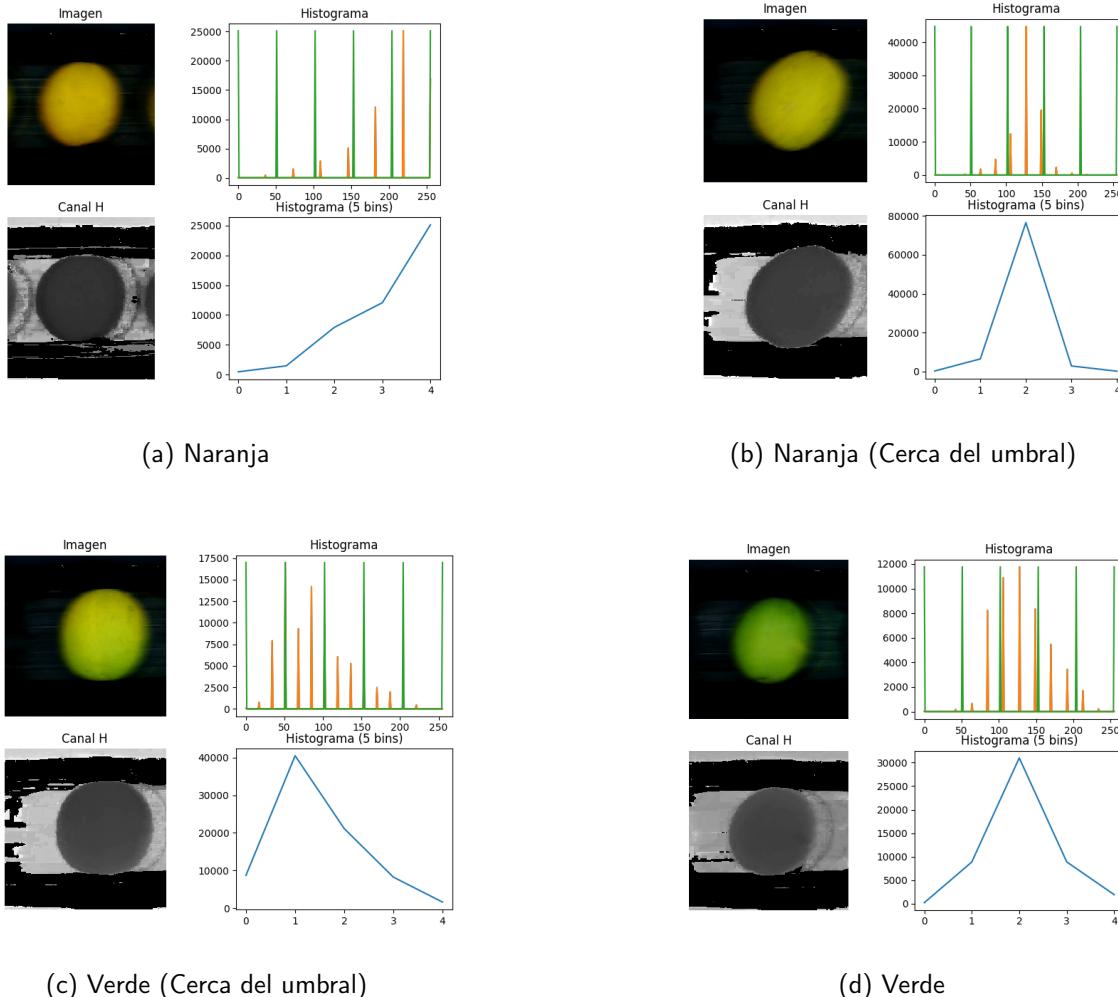


Figura 3.12. Resultados del análisis para cuatro posibles escenarios de color

flujo de como se capturó el dataset.

Primeramente, para cada imagen de cada clase primero se calcula la segmentación y el área de la misma, osease todos los pixeles de color blanco. Posteriormente se calculan todos los espacios de colores en base a la imagen de entrada, para eliminar todos los elementos de pixeles no segmentados, para solo tomar en cuenta los pixeles que pertenecen a la fruta y con ello calcular el histograma de 5 bins obteniendo un vector con los valores del histograma. Posteriormente cada elemento del histograma es multiplicado por 10000 para cambiar el valor de la precisión del histograma, esto con

3. Desarrollo

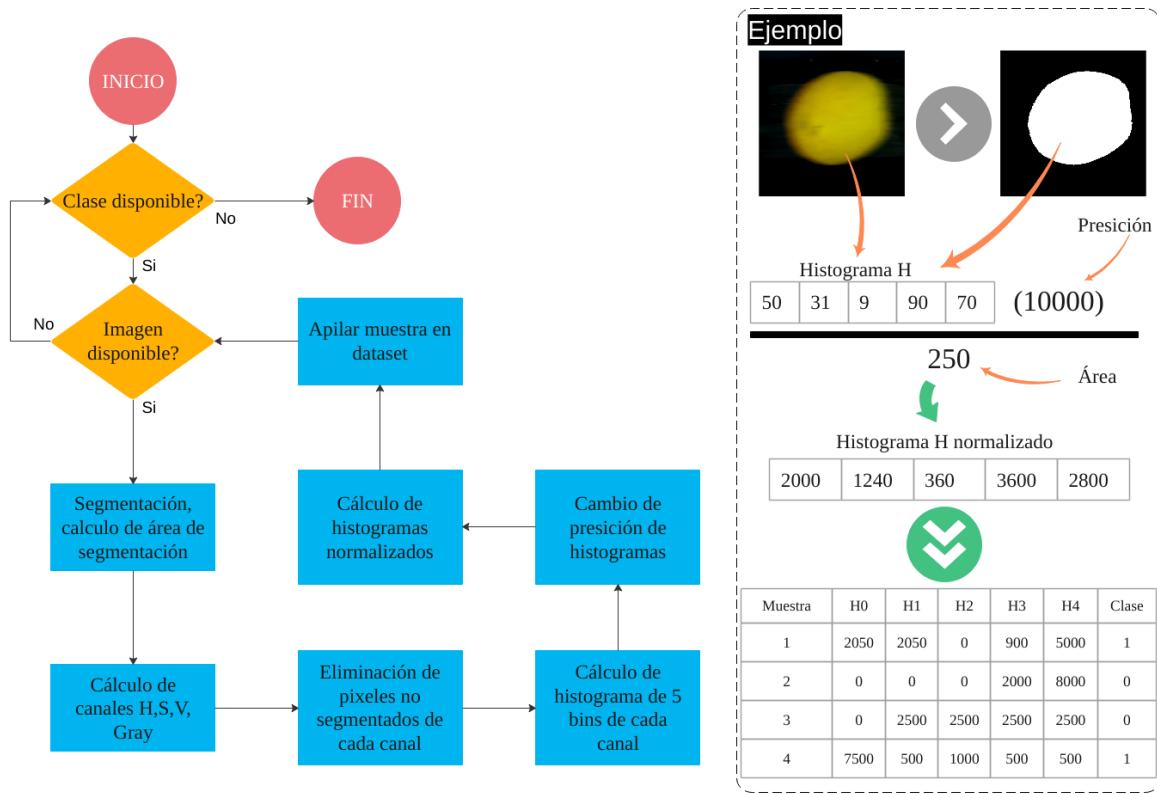


Figura 3.13. Diagrama de flujo y ejemplo de captura de data set de histogramas.

el fin de solamente manejar números enteros en los cálculos, ya que se tornan difíciles de manejar en un modelo HDL sintetizable. Por último para normalizar el histograma se divide cada elemento entre el área, y esto apilarlo en una matriz en donde se almacenan los datos para cada muestra.

En la figura 3.13 se muestra un ejemplo gráfico de como se efectúan los cálculos. Dado que es un ejemplo, solo se muestra un caso hipotético con un solo canal, obteniendo un dataset de 4 muestras con 5 características y una clase. Sin embargo en el experimento real, dado que los canales utilizados fueron R, G, B, H, S, V y Gray, por lo tanto se obtiene un dataset del número de imágenes con 35 características y una clase. La clase está representada por un número, siendo el número 1 para el color verde y 0 para el color naranja.

De última instancia, se procedió al entrenamiento en WEKA, de la misma manera que el modelo ADD de segmentación.

3.7 Modelo ADD clasificador por tamaño

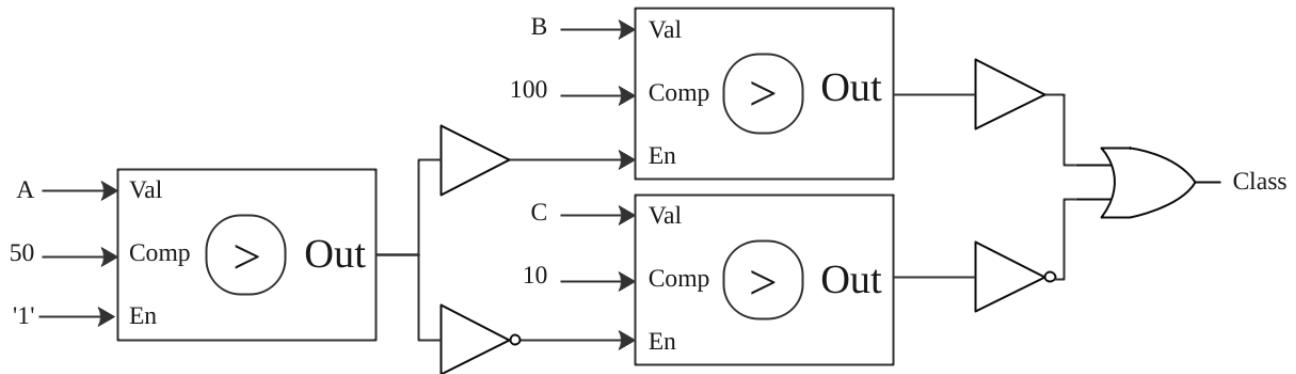
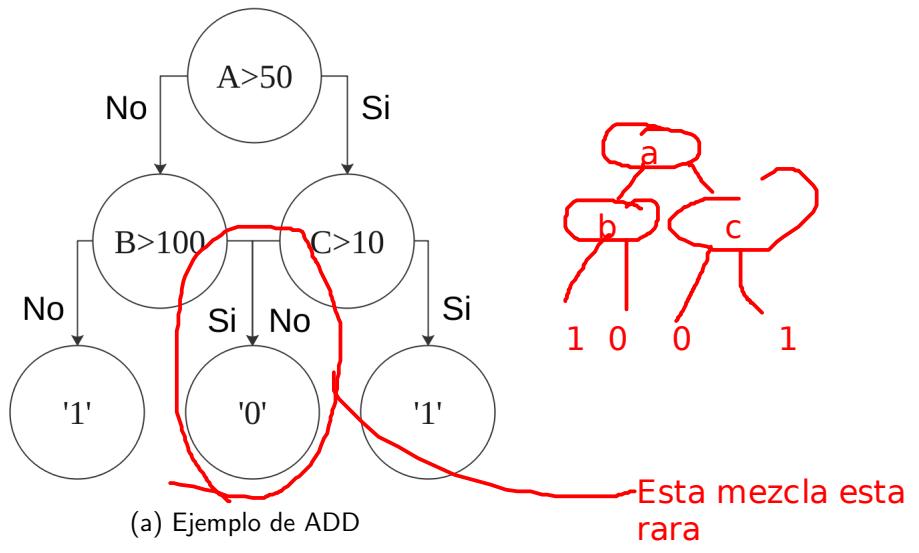
Para el desarrollo de este modelo, se utilizó el dataset de imágenes anterior. Primeramente se clasificó cada imagen de manera manual, bajo el criterio del autor, obteniendo 3 clases: chico, mediano y grande. De esta manera para cada imagen de cada clase se tomó como característica principal el área de segmentación.

De última instancia, se procedió al entrenamiento en WEKA, de la misma manera que el modelo ADD de segmentación. Éste árbol se generó con el fin de únicamente obtener los umbrales de área, y debido a que es un árbol sumamente pequeño, este no se realizará con los métodos de ADD sintetizables explicados en secciones siguientes, si no, con comparadores fijos.

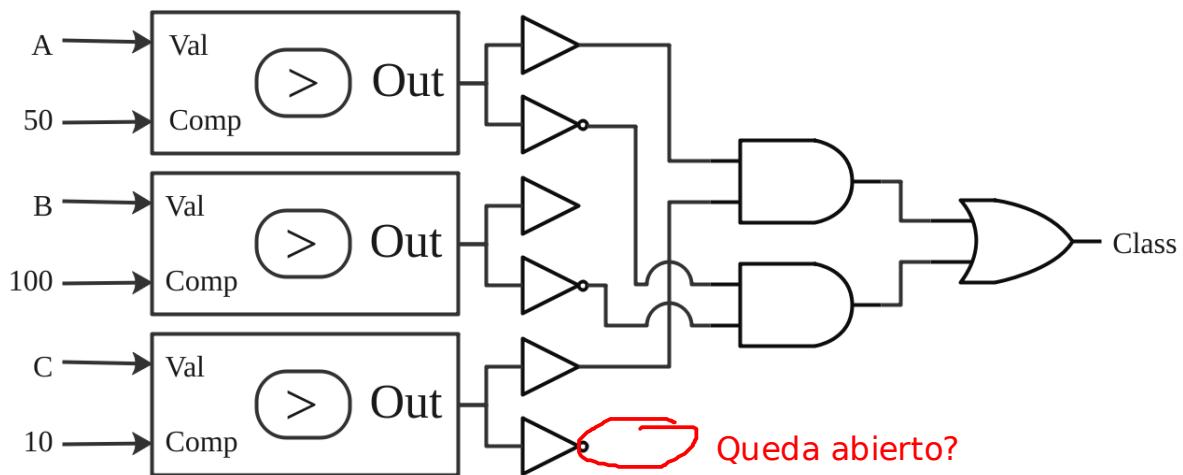
3.8 Arquitectura genérica de árbol de decisión

Un ADD puede ser fácilmente aplicado a hardware mediante HDL. Para explicar la arquitectura se tomará como referencia el ADD de la figura 3.14a, además, se tomará que todos los componentes sencillos tales como buffers, y compuertas lógicas comprenden solamente 1 unidad de retardo (UDR). El modelo encontrado en la literatura corresponde al de la figura 3.14b, en donde se puede ver bloques generales los cuales corresponden a comparadores con habilitación (el pin 'En'), donde sus entradas corresponden el valor a comparar (Val) y el valor fijo de comparación (Comp), estos, comprenden 2 UDR ya que primeramente se calcula si debe realizarse la comparación, y si esta se va a realizar se computa la salida. Sabiendo esto, se puede calcular que se tiene en el circuito completo un retardo de 7 UDR desde la entrada hasta la salida. Se puede reducir el circuito en 1 UDR reemplazando el primer comparador por uno sin habilitación, sin embargo, entonces un componente del módulo sería diferente y habría que realizar el cambio manualmente y correr riesgo de equivocación por parte del programador.

3. Desarrollo



(b) Ejemplo de arquitectura de ADD en serie



(c) Ejemplo de arquitectura de ADD en paralelo (propuesto)

Figura 3.14. Modelos de ADD en Hardware.

3.8. Arquitectura genérica de árbol de decisión

Por otra parte la arquitectura que se propone una forma más eficiente de realizar el cálculo de la clasificación, colocando algunos bloques en paralelo. Ésta se puede ver en la figura 3.14c la cual se comprende de 3 etapas. La primera, se compone de todos los comparadores, ya que estan en paralelo y no cuentan con un pin de habilitación estos solo tienen 1 UDR, después se tiene la etapa de compuertas AND, el cual resuelve la ramificación del ADD, y por último, la etapa de compuertas OR donde se adjuntan todas las posibilidades donde la salida es la clase '1'. Con esta forma, se tienen 4 UDR en el circuito, reduciendo el computo de la clase en 3 UDR.

De tenerse un ADD más profundo en el caso del sistema propuesto, la etapa de comparación no retrasaría más el computo, ya que la comparación se resuelve en la primera etapa de manera paralela, por otra parte la solución de la ramificación de la segunda etapa se puede reducir las UDR siempre y cuando se conecten la mayor cantidad de compuertas AND posible en paralelo, igualmente para la última etapa. Esto claramente no es posible en la arquitectura en serie.

3.8.1 Generación de un HDL a partir de un ADD

Debido a que para las pruebas cada ADD es diferente, se decidió realizar un Script que genera el HDL desde un modelo de ADD generado en WEKA el cuál tiene como nombre Script Constructor HDL de árbol de desición (SCHADD). Para la explicación del algoritmo se usará como referencia la figura 3.15, donde se muestra un modelo ADD de WEKA. El algoritmo funciona de la siguiente manera. Primeramente se identifican todas las ramificaciones en donde se obtiene la clase '1', como en la linea 5 y 6 del ejemplo, y se generan todas los comparadores que comprenden esas ramificaciones, junto con sus buffers e inversores, esto para ahorrar hardware en el ADD excluyento todas las compuertas restantes. Después para cada ramificación por medio de una compuerta de n entradas, donde n es el número de comparadores necesarios para resolver la ramificación, se conectan todas las salidas invertidas o no invertidas del comparador que resuelve cada ramificación por ejemplo la ramificación de la linea 5 es la inversión de la linea 1,2 y 4, Por último, las salidas de cada compuerta AND es conectada a una compuerta OR de m entradas, donde m es el número de ramificaciones

3. Desarrollo

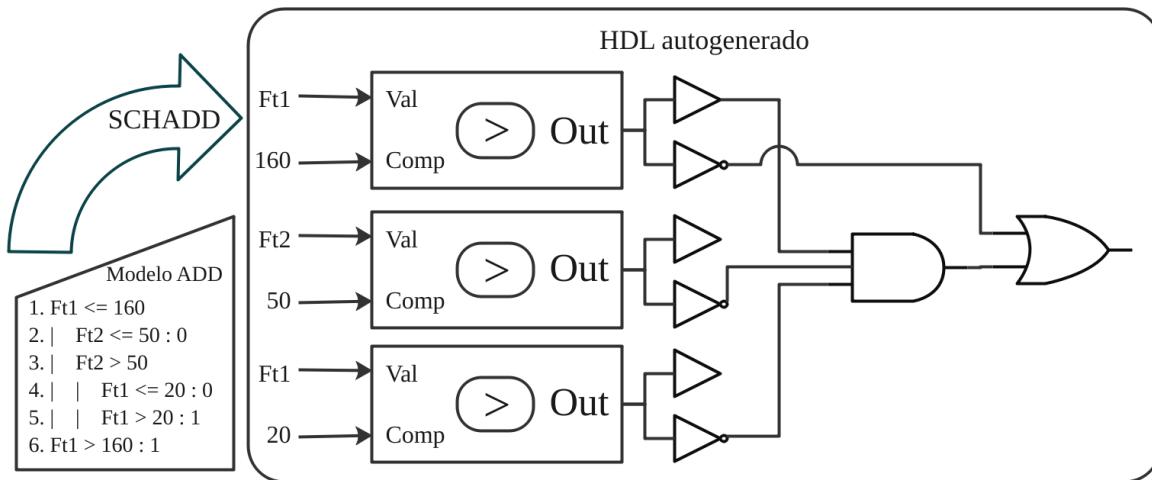


Figura 3.15. Ejemplo de ADD resuelto con SCHADD.
generado mediante

donde la clase es '1' en el ADD, y la salida de la compuerta OR es la clase computada.

3.9 Arquitectura ~~de~~ procesador de video

La arquitectura propuesta en el proyecto está compuesta de varias etapas, las cuales independientemente cumplen con una tarea en específico en donde se va resolviendo por etapas la clasificación. En la figura 3.16 se puede apreciar un esquema general de la arquitectura, en donde se aprecia las capas de comunicación y de clasificación.

La capa de comunicación se encarga de manejar las señales entre el MCU y el FPGA, para la gestión de la clasificación, ya que la capa de clasificación estará en reposo y solo funcionará cuando el MCU lo indique.

La capa de clasificación está compuesta por 4 etapas, en las cuales se implementó una arquitectura en pipeline, por lo que entre cada capa se encuentran módulos de registro controlados por el reloj de video. Debido a las etapas de registros es que el sistema estará retrasado 4 ciclos de reloj de video, lo cual es despreciable teniendo un reloj a una frecuencia de 155MHz normalmente. Las capas y etapas de la arquitectura serán explicadas en esta sección.

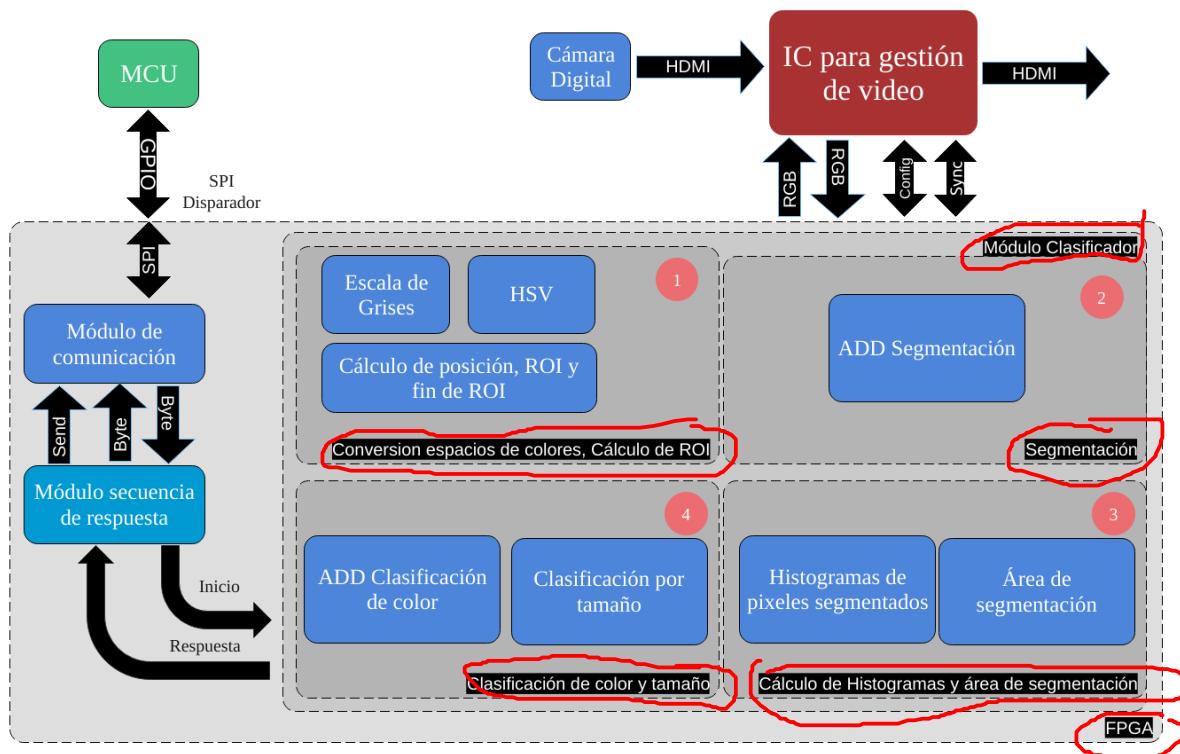


Figura 3.16. Esquema general de la arquitectura.

3.10 Módulo Clasificador

~~Como se mencionó en la sección 3~~ El módulo clasificador es parte de la capa de clasificación la cual está compuesta de 4 etapas que serán explicadas en esta sección.

razón/motivo
 La ~~cause~~ de dividirlo en etapas, es que la arquitectura de algunos módulos es compleja, aún tratándose de circuitos puramente combinacionales, por lo que la salida se vería afectada si se realizara un circuito combinacional general. Además, la naturaleza de algunos componentes son secuenciales por lo que registros entre las etapas, facilitan el manejo y orden de los datos.

La primera etapa está encargada del cálculo de algunas métricas utilizadas en etapas posteriores, la segunda está encargada puramente de la segmentación, la cual es de vital importancia para las siguientes etapas. La tercera, está encargada de la recolección de datos para la clasificación por color

3. Desarrollo

y tamaño, y la última es la encargada de realizar la clasificación.

3.10.1 Conversión de espacios de colores y cálculo de ROI

En esta etapa se realizan los cálculos de espacios de colores y posicionamiento de píxeles. Está posicionado en primer lugar ya que la conversión de espacios de colores será la entrada de módulos posteriores, así como el conocimiento del posicionamiento de la ROI. La etapa está compuesta por los siguientes módulos.

Cualquier libro de OpenCV

3.10.1.1 Módulo convertidor RGB a escala de grises

Este módulo tiene como entradas los canales del pixel entrante y la señal de reloj, y simplemente se encarga de realizar la operación que se aprecia en la ecuación 3.1 (la que corresponde como salida del módulo), la cual es la implementada en las librerías de OpenCV **aquí faltareía**, esto con el fin de coincidir con en cálculo de datos obtenidos desde la computadora. El módulo está sincronizado por lo que se hará el cálculo cada flanco positivo del reloj.

$$Gray(R, G, B) = (0,299)R + (0,587)G + (0,114)B \quad (3.1)$$

Sin embargo, en este módulo no se utilizaron cálculos con números de punto flotante por lo que se calculó con números de punto fijo utilizando la ecuación 3.2. Para determinar esta ecuación primeramente se obtuvo la precisión necesaria para el cálculo, es decir, el número de bits necesarios después del punto en el formato de punto fijo. Como cada coeficiente de la ecuación 3.1 es menor que cero, entonces, no se necesitarán bits de la parte entera por lo que solo hay que poder representar los números en enteros, y para definir el número de bits es necesario poder representar el más grande de ellos, en este caso el numero 587 el cual se representa con mínimo 10 bits. Posteriormente al realizar las multiplicaciones, cada número resultante es de 18 bits, por lo que al hacer la suma, resulta en un número con la misma cantidad de bits. Finalmente se descartan todos los bits de precisión, tomando los 8 bits más significativos, o lo que es lo mismo, dividir el la suma entre 2^{10} .

$$Gray(R, G, B) = \frac{(299)R + (587)G + (114)B}{2^{10}} \quad (3.2)$$

3.10.1.2. Módulo convertidor RGB a HSV

Este módulo tiene como entradas los canales del pixel entrante y la señal de reloj, y simplemente se encarga de realizar las operaciones que se aprecian en las ecuaciones 3.3, 3.4, 3.5, 3.6. Las cuales también son las implementadas en las librerías de OpenCV [aquifaltaref](#). El módulo está sincronizado por lo que se hará el cálculo cada flanco positivo del reloj.

Quienes son:
 R, R', G, G', V, V'
 D', S', S, H' (es obvio,
 pero en alguna
 parte específicas
 que son RGB, R'G'yB'
 son las normalizaciones, etc
 etc

$$\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned} \quad (3.3)$$

$$\begin{aligned} V' &= \max(R', G', B') \\ V &= (255)V' \end{aligned} \quad (3.4)$$

$$D' = V' - \min(R', G', B')$$

$$S' = \begin{cases} 0 & \text{si } V' = 0 \\ \frac{D'}{V'} & \text{si } V' \neq 0 \end{cases} \quad (3.5)$$

$$S = (255)S'$$

$$H' = \begin{cases} 0 + \frac{60(G' - B')}{D} & \text{si } V' = R' \\ 120 + \frac{60(B' - R')}{D} & \text{si } V' = G' \\ 240 + \frac{60(R' - G')}{D} & \text{si } V' = B' \end{cases} \quad (3.6)$$

$$\text{si } H' < 0 \text{ entonces } H' = H' + 360$$

$$H = \frac{H'}{2}$$

Sin embargo no se utilizaron cálculos con números de punto flotante ni enteros negativos. Como

3. Desarrollo

se puede ver en la ecuaciones 3.7, 3.8, 3.9, no se utilizan los componentes R' , G' y B' .

Para la ecuación 3.8 para el cálculo de S , primeramente D tiene que ser multiplicado por 255, debido a que tanto D como V son de 8 bits, por lo que al dividirse primero resultaría en un numero de 0 bits lo cual es inconsistente, por otra parte la operación consigue obtener una relación entre los dos números con un rango $[0, 1]$ y posteriormente multiplicarlo por 255 originalmente en la ecuación 3.5, si se realiza esta operación previamente, el resultado será de 8 bits obteniendo el resultado buscado.

Para el cálculo de H' y evitar usar números negativos se propone la ecuación 3.9. Esta se determinó debido a que dependiendo de los valores del pixel el segundo término para cada caso en la ecuación 3.6, puede tomar un valor positivo o negativo, y esta sustraería o añadiría del primer término para cada caso, por eso simplemente se reescribió la forma del segundo término para quedar de manera equitativa exceptuando de todo esto al primer caso ya que para $D = 0$ resulta en una división indeterminada. Para finalmente calcular H , se necesita extraer el LSB de H' ya que el rango de H es $[0, 360)$ y para poder representarse con 8 bits es necesario este descarte.

$$V = \max(R, G, B) \quad (3.7)$$

$$\begin{aligned} D &= V - \min(R, G, B) \\ S &= \begin{cases} 0 & \text{si } V = 0 \\ \frac{(255)D}{V} & \text{si } V \neq 0 \end{cases} \end{aligned} \quad (3.8)$$

$$H' = \begin{cases} 0 & \text{si } S = 0 \\ 0 + \frac{60(G-B)}{D} & \text{si } (V = R) \& (G > B) \\ 360 - \frac{60(B-G)}{D} & \text{si } (V = R) \& (B \geq G) \\ 120 + \frac{60(B-R)}{D} & \text{si } (V = G) \& (B > R) \\ 120 - \frac{60(R-B)}{D} & \text{si } (V = G) \& (R \geq B) \\ 240 + \frac{60(R-G)}{D} & \text{si } (V = B) \& (R > G) \\ 240 - \frac{60(G-R)}{D} & \text{si } (V = B) \& (G \geq R) \end{cases} \quad (3.9)$$

$$H = \frac{H'}{2}$$

3.10.1.3. Módulo de cálculo de posición, ROI y fin de ROI

Este módulo es necesario para computar ROI, donde se define un bit que cuando es 1, significa que está dentro de ROI. Se trata de un módulo donde internamente se encuentran 2 contadores de pulsos, uno con habilitación y otro no, y además un circuito combinacional.

Uno de los contadores es el que calcula la coordenada horizontal X (definida en este trabajo como pX), el pulso a contar es el reloj de video, la habilitación es la SVA y su reset es la SSH.

El otro contador es el que calcula la coordenada vertical Y (definida en este trabajo como pY), el pulso a contar es la inversión de SVA y su reset es la SSV.

cojones sois la hostia!!

Para ~~computar~~ el bit de ROI, es necesario hablar de los parámetros Xm , Xmy , Ym , Ymy , Xp , Yp ; los cuales simplemente son las coordenadas que representan los bordes de ROI, y la cantidad de pixeles del fotograma, donde significan lo siguiente:

- Xm : Coordenada horizontal del punto izquierdo superior de ROI
- Ym : Coordenada vertical del punto izquierdo superior de ROI

3. Desarrollo

- X_{my} : Coordenada horizontal del punto derecho inferior de ROI
- Y_{my} : Coordenada vertical del punto derecho inferior de ROI
- pX : Coordenada horizontal actual
- pY : Coordenada vertical actual
- X_p : Número de pixeles en eje horizontal
- Y_p : Número de pixeles en eje vertical

Los parametros de ROI deben seguir las sentencias de 3.10.

cumplir las siguientes restricciones:

$$\begin{aligned} 0 \leq X_m < X_{my} < X_p \\ 0 \leq Y_m < Y_{my} < Y_p \end{aligned} \quad (3.10)$$

Y para el cálculo de el Bit de ROI descrita en la ecuación 3.11, solamente es necesario comparar las coordenadas actuales con los parametros de borde de ROI. FDR?

$$Bit_{ROI} = \begin{cases} 1 & \text{si } (X_m \leq pX < X_{my}) \& (Y_m \leq pY < Y_{my}) \\ 0 & \text{de otra manera} \end{cases} \quad (3.11)$$

$$FDR = \begin{cases} 1 & \text{si } (pX = X_{my}) \& (pY = Y_{my}) \\ 0 & \text{de otra manera} \end{cases} \quad (3.12)$$

3.10.2 Segmentación

La segmentacion requiere

Como se vió en la sección 3.5 esta etapa, en su totalidad está compuesta de un ADD y es la encargada de realizar la segmentación pixel a pixel y está compuesto por el siguiente módulo. ADD, el cual esta

~~3.10.2.1. Módulo ADD segmentador~~

~~Este módulo~~ está preparado para recibir las características computadas en la etapa anterior.

El módulo tiene como entradas los canales R, G, B, H, S, V, y Gray, donde todos ellos son de 8 bits y tiene solo una salida de 1 bit, el bit de segmentación llamado a partir de aquí como BDS. Es un circuito puramente combinacional por lo que no cuenta con registros sincronizados por el reloj de video, sin embargo estos están implementados entre capas lo que se podría decir que cuenta con ellos para un manejo ordenado.

3.10.3 Cálculo de histogramas y área de segmentación

En esta etapa se implementaron histogramas de 5 bins para cada canal computado en la primera etapa. Además de un módulo el cual se encarga de computar el área definida por la segmentación computada en la etapa 2.

~~3.10.3.1. Histogramas de pixeles segmentados~~

Como lo visto en la sección [3.6](#), para alimentar el ADD para la clasificación de color es necesario el cálculo de histogramas de 5 bins para cada canal de color. En este módulo se tiene como entrada solamente el valor de 8 bits a clasificar, el reset y el reloj, y como salida 5 números de 15 bits donde se representan los 5 bins del histograma normalizado. Para ello se implementó en hardware el algoritmo [1](#), el cuál tiene un estilo de escritura del lenguaje C. El funcionamiento se basa por contadores con habilitación, donde su habilitación se determina por BDS y Bit_{ROI} , para al final habilitando así el contador que representa al bin correspondiente. Por otra parte, al recibir el flanco de bajada de FDR, se calcula el histograma normalizado y se activa el bit HL, el cual representa que el histograma está listo o calculado. Todo lo antes descrito sincronizado por el reloj de video y con un reset asíncrono implementado.

Los bordes determinados en la variable B, fueron los mismos indicados en el algoritmo para todos los canales, a excepción de el canal H, ya que el rango de H es [0, 180).

En esta etapa, se construyeron 7 módulos de histograma que a su vez como salida se obtienen

3. Desarrollo

Algoritmo 1 Algoritmo para módulo de histogramas

```
B[6] = {0, 51, 102, 153, 204, 256}; // Bordes de los bins
cntpix = 0; //Contador de pixeles
Ct [5] = {0}; // Contadores de histogramas
Hs [5] = {0}; // Histogramas definidos
HL = 0; // Bit para indicar que está listo el histograma
si reset entonces
    Ct [5] = {0};
    Hs [5] = {0};
    cntpix = 0;
    HL = 0;
en otro caso
    si Flanco de subida (Reloj) entonces
        si BDS && BitROI entonces
            Leer Valor;
            para i=0 ; i<5 ; i++ hacer
                si (B[i] ≤ Valor) && (Valor < B[i+1]) entonces
                    Ct[i] = Ct[i] + 1;
                fin
            fin
            cntpix = cntpix + 1;
        fin
    fin
    si Flanco de bajada (FDR) entonces
        para i=0 ; i<5 ; i++ hacer
            Hs[i] = (Ct[i]*10000)/cntpix;
        fin
        HL = 1;
    fin
fin
```

35 buses con los histogramas normalizados por canal. Debido a que la señal HL es la misma para cada caso, se utiliza una solamente.

3.10.3.2. Módulo de cálculo de área de segmentación

Este módulo tiene como entradas, a las señales BDS, Bit_{ROI} , reset y reloj de video, y como salida un número de 15 bits que corresponde al área normalizada. El funcionamiento del módulo se describe con el algoritmo 2, y está compuesto de un contador por habilitación, donde su habilitación

Algoritmo 2 Algoritmo para cálculo del área segmentada

```

cntpix = 0; //Contador de pixeles segmentados
pixTot = (Ymy-Ym) * (Xmy - Xm); // Cálculo del área de ROI
area = 0; //Area resultante
si reset // SSV entonces
    cntpix = 0;
    area = 0;

en otro caso
    si Flanco de subida (Reloj) entonces
        si BDS && BitROI entonces
            | cntpix = cntpix + 1;
        fin
    fin
    si Flanco de bajada (FDR) entonces
        | area = (10000*cntpix)/pixTot;
    fin
fin

```

es calculada por la operación $BDS \&\& Bit_{ROI}$, la cuál significa que si se está dentro de ROI, y además es un pixel segmentado, entonces se contabiliza el pixel. Al obtener un flanco de bajada de la señal FDR, se calcula el área normalizada. Todo lo antes descrito sincronizado por el reloj de video y con un reset asíncrono implementado.

3.10.4 Clasificación de color y tamaño

En esta última etapa, se calcula la clasificación. De parte del clasificador de tamaño por medio de comparadores y en la parte de color por medio de un ADD generado. Esta etapa tiene como salida un número de 8 bits compuesto por las señales BCC y BTC ~~que se ven en las secciones 3.10.4.1 y 3.10.4.2~~ quedando compuesto el número de la siguiente manera:

- Bit 0: Bit de Clase de Color (BCC)
- Bit 2-1: Bits de clase de Tamaño (BCT)
- Bit 7-3: Bits constantes a 0, estos se dejan con fines de diseño estandar que se ven en la sección 3.11

3. Desarrollo

Este bus es denominado como la Respuesta del Módulo Clasificador o RMC, visto en la figura 3.16 como "respuesta". **Este modulo contiene los siguientes bloques:**

A)

3.10.4.1. Módulo ADD de clasificación de color

Este módulo tendrá como entradas los 35 números de 15 bits correspondientes a los histogramas normalizados vistos en el módulo de la sección 3.10.3.1 y como salida un bit que corresponde a la clase de color llamado Bit de Clase de Color o BCC, donde su salida es interpretada como:

- 0: Color naranja
- 1: Color verde

Al tratarse de un circuito combinacional, no es necesaria la adición de la señal de reloj.

B)

3.10.4.2. Módulo clasificador de tamaño

El módulo tiene como entrada únicamente el área calculada por el modulo mostrado en la sección 3.10.3.2, y por salida un número de 2 bits llamado Bits de Clase de Tamaño o BCT, el cual es interpretada su salida como:

- 0: Tamaño chico
- 1: Tamaño mediano
- 2: Tamaño grande
- 3: Inválido

Al tratarse de una simple comparación la cual es tratada como un circuito combinacional, no es necesaria la adición de la señal de reloj.

3.11 Módulos de comunicación y gestión de respuesta

La capa de comunicación del proyecto, es la encargada de administrar las señales que provee el MCU, sincronizandolas con las internas del FPGA para finalmente regresar una respuesta, la cuál será la clasificación resultante, esto con el fin de realizar una clasificación de la manera más rápida posible. Cabe mencionar, que esta capa está regida por diferentes señales de reloj para diferentes etapas. El funcionamiento de cada módulo será explicado en esta sección.

A)

~~3.11.1 Módulo de comunicación~~

Disparo de clasificación (DDC) -> **Elaborar**

B)

~~3.11.2 Módulo de secuencia de respuesta~~

Este módulo es el encargado de administrar la clasificación correctamente. Tiene como entrada, las señales Reloj de video, reset, y reloj interno DDC, SSV, HL (de la sección 3.10.3.1) , y RMC , y como salida la señal de Inicio de clasificación (mostrado en la figura 3.16 como "Inicio") o también llamado INC, el Byte de respuesta del SPI (BRSPI), y el disparo de clasificación lista (DCL).

El reloj interno mencionado, corresponde al del FPGA, siendo este de 150MHz. El funcionamiento del módulo, se muestra en el algoritmo3, el cuál está compuesto por una Máquina de estados finitos. La secuencia está controlada por el reloj interno, y con el, se encarga de que por medio la señal DLC, esperar un nuevo fotograma para ser contemplado en su totalidad, ya que puede ser que el disparo se realice a mediación del fotograma, es por eso que la secuencia espera uno nuevo, para ser procesado, y con el, se genera la señal para el inicio de la clasificación, y la correcta administración de la respuesta al módulo de comunicación.

.

3. Desarrollo

Algoritmo 3 Secuencia de respuesta

```
e = 0; //Estado actual
ef = 0; // Estado futuro
si Flanco de subida (Reloj interno) entonces
    switch(e)
        0: //Estado de reinicio
        INC = 0;
        BRSPI = 0;
        DCL = 0;
        ef = 1;
        1: //Estado en reposo
        INC = 0;
        DCL = 0;
        si DDC entonces
            |   ef = 2;
        fin
        2: //En espera de un fotograma nuevo
        si SSV entonces
            |   ef = 3;
        fin
        3: //En espera de finalizar la clasificación
        INC = 1;
        si HL entonces
            |   Leer RMC;
            |   ef = 4;
        fin
        4: //Mandar respuesta
        BRSPI = RMC;
        DCL = 1;
        ef = 1;
    fin
    si reset entonces
        |   e = 0;
    en otro caso
        si Flanco de subida (Reloj interno) entonces
            |   e = ef;
        fin
    fin
```

4

Resultados y discusión

En la parte de la toma del dataset de imagenes hay que poner que se usaron 7 videos para el data set, y aparte de los otros 2 videos se tomaron para el el traning set

Principalmente deberias poner Tablas en esta seccion, ademas de secuencias del resultado de segmentacion/clasificacion por tus modulos propuestos

4. Resultados y discusión

4.1 ADD de segmentación

4.1.1 Captura de data set de pixeles

4.1.2 Entrenamiento y prueba por computadora

4.1.3 Implementación en FPGA

4.2 ADD de clasificación de color

4.2.1 Captura de data set de histogramas

4.2.2 Entrenamiento y prueba por computadora

4.2.3 Implementación en FPGA

4.3 Clasificación de tamaño

4.3.1 Captura de data set de áreas

4.3.2 Entrenamiento y prueba por computadora

4.3.3 Implementación en FPGA

* Tabla de recursos empleados por la computadora

5

Conclusiones

Algo...

Trabajos futuros

Algo...

**** IMPORTANTE ****

Falta un gran Diagrama asociado a una sección (pudee ir al inicio del capitulo 3) cuyo titulo puede ser "Metodología", que intente resumir como sigue:

Fase 1: Captura del dataset de naranjas (se utiliza la camara y la capturadora de video, requiere solo la capturadora)

Fase 2: Generacion del Dataset (A partir de los videos, se procesan las naranjas y se construye el dataset. Requiere de una PC , Python para aislar cada naranja y para generar el Dataset - eliminando repetidos (supongo en formato arff))

Fase 3: Entrenamiento a partir del dataset (Requiere Weka y genera como salida el/los arbol(es) de decision)

Fase 4: Generacion del HW para clasificar naranjas en base al modelo generado por Weka (Requiere ademas de la PC, el Script para generar el HW del arbol de decision, y la herramienta del fabricante del FPGA (ISE) para generar el .BIT. Una vez finalizado, se configura el FPGA y queda listo para la fase de pruebas

Fase 5: Clasificacion de las naranjas (Se requiere el FPGA y a partir del modelo previamente entrenado y las imagenes de entrada, lleva a cabo la clasificacion). La PC deja de ser indispensable (podria utilizarse para actualizar el arbol de decision como trabajo futuro), pero se puede conectar un monitor para depurar

Fase 6: (Por implementar): Instruir a los actuadores para llevar a cabo la separacion de la fruta en base a los criterios establecidos. Se requiere los actuadores, sensores y procesar la salida del FPGA.

(Me da la impresion que esta lista de referencias pudiera ser mas larga)

Recomiendo revisar cada referencia para no dejar datos faltantes: Congreso/Revista, Numero de Paginas, Año, etc. Ver Entradas con CIRCULO

Bibliografía

Té encargo checar porque los meses aparecen con dos digitos como 04 y no como apr (o abr)

- [1] G. Ancillo, *Los ciótricos*. Valencia: Universitat de Valeónia, Jardíó Botaónic, 2014.
- [2] Food and A. O. of the United Nations, "Citrus fruit fresh and processed statistical bulletin." www.fao.org/3/a-i8092e.pdf, 2016.
- [3] SAGARPA, "National agricultural planning 2016-2030." https://www.gob.mx/cms/uploads/attachment/file/257073/Potencial-C_tricos-parte_uno.pdf, 2016.
- [4] Seema, A. Kumar, and G. S. Gill, "Automatic fruit grading and classification system using computer vision: A review," in *2015 Second International Conference on Advances in Computing and Communication Engineering*, pp. 598–603, May 2015.
- [5] J. P. Mercol, M. J. Gambini, and J. M. Santos, "Automatic classification of oranges using image processing and data mining techniques," 2008.
- [6] Y. Sirisathitkul, N. Thumpen, and W. Puangtong, "Automated chokun orange maturity sorting by color grading," *Walailak Journal of Science and Technology*, vol. 3, 06 2006.
- [7] A. Martínez-Usó, F. Pla, and P. García-Sevilla, "Multispectral image segmentation for fruit quality estimation," in *Proceedings of the 2005 Conference on Artificial Intelligence Research and Development*, (Amsterdam, The Netherlands, The Netherlands), pp. 51–58, IOS Press, 2005.
- [8] G. Feng and C. Qixin, "Study on color image processing based intelligent fruit sorting system," vol. 6, pp. 4802 – 4805 Vol.6, 07 2004.
- [9] D. Unay and B. Gosselin, "Stem and calyx recognition on 'jonagold' apples by pattern recognition," *Journal of Food Engineering*, vol. 78, no. 2, pp. 597 – 605, 2007.

BIBLIOGRAFÍA

- [10] Y. A. Ohali, "Computer vision based date fruit grading system: Design and implementation," *Journal of King Saud University - Computer and Information Sciences*, vol. 23, no. 1, pp. 29 – 36, 2011.
- [11] S. Khoje, S. Bodhe, and A. Adsul, "Automated skin defect identification system for fruit grading based on discrete curvelet transform," *International Journal of Engineering and Technology*, vol. 5, pp. 3251–3256, 08 2013.
- [12] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, "Automatic fruit classification using random forest algorithm," in *2014 14th International Conference on Hybrid Intelligent Systems*, pp. 164–168, Dec 2014.
- [13] Woo Chaw Seng and S. H. Mirisaei, "A new method for fruits recognition system," in *2009 International Conference on Electrical Engineering and Informatics*, vol. 01, pp. 130–134, Aug 2009.
- [14] A. Rocha, D. C. Hauagge, J. Wainer, and S. Goldenstein, "Automatic fruit and vegetable classification from images," *Computers and Electronics in Agriculture*, vol. 70, no. 1, pp. 96 – 104, 2010.
- [15] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic fruit classification using deep learning for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 1027–1034, Feb 2019.
- [16] M. A. Nuño-Maganda, Y. Hernández-Mier, C. Torres-Huitzil, and J. Jiménez-Arteaga, "Fpga-based real-time citrus classification system," in *2014 IEEE 5th Latin American Symposium on Circuits and Systems*, pp. 1–4, Feb 2014.
- [17] F. Saqib, A. Dutta, J. Plusquellic, P. Ortiz, and M. S. Pattichis, "Pipelined decision tree classification accelerator implementation in fpga (dt-caif)," *IEEE Transactions on Computers*, vol. 64, pp. 280–285, Jan 2015.

- [18] D. Wilking and T. Röfer, "Realtime object recognition using decision tree learning," in *RoboCup 2004: Robot Soccer World Cup VIII* (D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, eds.), (Berlin, Heidelberg), pp. 556–563, Springer Berlin Heidelberg, 2005.
- [19] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 45–58, 12 2011.
- [20] E. Stattner, P. Hunel, N. Vidot, and M. Collard, "Acoustic scheme to count bird songs with wireless sensor networks," in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1–3, June 2011.
- [21] S. Arivazhagan and R. Newlin Shebiah and S. Selva Nidhyanandhan and Lakshmanan Ganesan, "Fruit Recognition using Color and Texture Features," 2010.
- [22] J. Pablo Mercol, J. Gambini, and J. Miguel Santos, "Automatic classification of oranges using image processing and data mining techniques," 04 2019.
- [23] Y. SIRISATHIKUL, N. THUMPEN, and W. PUANGTONG, "Automated chokun orange maturity sorting by color grading," *Walailak Journal of Science and Technology (WJST)*, vol. 3, no. 2, pp. 195–205, 2011.
- [24] G. Feng and C. Qixin, "Study on color image processing based intelligent fruit sorting system," vol. 6, pp. 4802 – 4805 Vol.6, 07 2004.
- [25] G. Kay and G. de Jager, "A versatile colour system capable of fruit sorting and accurate object classification," in *Proceedings of the 1992 South African Symposium on Communications and Signal Processing*, pp. 145–148, Sep. 1992.
- [26] M. A. Nuño-Maganda, Y. Hernández-Mier, C. Torres-Huitzil, and J. Jiménez-Arteaga, "Fpga-based real-time citrus classification system," in *2014 IEEE 5th Latin American Symposium on Circuits and Systems*, pp. 1–4, Feb 2014.

REPETIDA [26] con [15]

BIBLIOGRAFÍA

- [27] S. Khoje, D. S. K. Bodhe, and A. Adsul, "Automated skin defect identification system for fruit grading based on discrete curvelet transform," 2013.

FPGA-based

- [28] M. Hervas, R. Alsina-Pagès, F. Alías, and M. Salvador, "An fpga-based wasn for remote real-time monitoring of endangered species: A case study on the birdsong recognition of *botaurus stellaris*," *Sensors*, vol. 17, p. 1731, 06 2017.

- [29] Q. Li and A. Bermak, "A low-power hardware-friendly binary decision tree classifier for gas identification," *Journal of Low Power Electronics and Applications*, vol. 1, pp. 45–58, 12 2011.

- [30] J. Gill, P. S. Sandhu, and T. Singh, "A review of automatic fruit classification using soft computing techniques," 2014.

- [31] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, "Automatic fruit classification using random forest algorithm," in *2014 14th International Conference on Hybrid Intelligent Systems*, pp. 164–168, Dec 2014.

- [32] A. García Serrano, *INTELIGENCIA ARTIFICIAL. Fundamentos, práctica y aplicaciones*. 07 2012.

- [33] V. Jatana, *Machine Learning For Beginners*, 10 2018.

- [34] A. Criollo, L. Suárez Zambrano, and O. Oña, *Machine learning: Importance, advances, techniques and applications*, 04 2018.

- [35] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

- [36] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

- [37] S. Salzberg, "C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993," *Machine Learning*, vol. 16, pp. 235–240, 1994.

- [38] J.-Y. Dufour, *Intelligent Video Surveillance Systems*. Wiley-ISTE, 2012.

- [39] A. Koschan and M. A. Abidi, *Digital Color Image Processing*. New York, NY, USA: Wiley-Interscience, 2008.
- [40] R. Bajaj and S. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," vol. 35, pp. 1–1, 01 2015.