ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN

Algoritmo de Detección de Bordes en Imágenes con NIOS II

TESINA DE SEMINARIO

Previa la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

Calle Cóndor Andrés Rafael

Pazmiño Villao Elina Paola

GUAYAQUIL – ECUADOR

AÑO 2012

AGRADECIMIENTO

Quiero agradecer a toda mi familia, en especial a mis padres que han sido han sido testigos de cada uno de mis logros y el pilar fundamental para seguir siempre adelante sin darme por vencido, sin ellos no hubiera podido llegar a cumplir este objetivo.

A todos mis compañeros y sobre todo a mis amigos que formaron parte de mi vida universitaria, contribuyendo día a día a que pueda llegar a este día tan importante.

Andrés Rafael Calle Cóndor

En primer lugar quisiera agradecer a Dios, porque sin él nada fuera posible.

Gracias a mis padres, que siempre han sido y son ese pilar fundamental en mi vida, sin sus sabios consejos, apoyo y orientación no sería la persona que soy ahora.

Elina Paola Pazmiño Villao.

DEDICATORIAS

A toda mi familia y amigos, en especial a mis padres ya que sin ellos no hubiera podido lograr con éxito todos mis objetivos planteados.

Andrés Rafael Calle Cóndor

A mis padres por toda su ayuda brindada ya que sin ellos nada de lo obtenido podría haber sido posible.

Elina Paola Pazmiño Villao

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Ponguillo

PROFESOR DEL SEMINARIO DE GRADUACIÓN

Dr. Daniel Ochoa

PROFESOR DELEGADO POR EL DECANO DE LA FACULTAD

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este trabajo, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”. (Reglamento de exámenes y títulos profesionales de la ESPOL)

Andrés Rafael Calle Cóndor

Elina Paola Pazmiño Villao

RESUMEN

El proyecto consiste en la implementación de un algoritmo de procesamiento de imágenes utilizando la tarjeta de desarrollo **DE2 ALTERA,** basada en un dispositivo **ALTERA CYCLONE II FPGA** junto con memorias embebidas y un **PROCESADOR NIOS II INTEGRADO,** que ayudará en el objetivo, el cual es detectar el borde de una imagen.

Para la realización del proyecto se aplican tres etapas. La primera etapa está basada en el almacenamiento de la imagen en la memoria SDRAM, la siguiente etapa se basa principalmente en leer los datos almacenados en la SDRAM y procesarlos correctamente para poder obtener su matriz, la cual contiene la información de la imagen, y una última etapa donde por medio de algoritmos implementados en código c, se procede a obtener la imagen resultante, la misma que mostrara solamente los bordes de la imagen originalmente almacenada.

Al proyecto se lo ha estructurado en 4 capítulos como se lo explica a continuación:

En el **primer capítulo**, se exponen los objetivos generales y específicos del proyecto, además de especificar claramente los alcances y limitaciones.

En el **segundo capítulo**, se da a conocer la parte teórica, explicando los conceptos básicos de los elementos y software usados para el desarrollo del proyecto, además de conocimientos adicionales que se tuvo en cuenta con lo que respecta a procesamiento de imágenes.

En el **tercer capítulo**, se señaló como se desarrolló el proyecto, para este caso se muestra el diseño base y se señala paso a paso las distintas etapas que se tuvo que ejecutar para implementar la detección de bordes dentro de los parámetros inicialmente establecidos.

Para el **cuarto capítulo**, se muestra una parte de la matriz resultante y se visualiza los bordes de la imagen en una pantalla,

Finalmente se realizo un análisis basado tanto en la matriz como en la imagen resultante, con lo que se pudo generar las respectivas conclusiones y recomendaciones.

ÍNDICE GENERAL

AGRADECIMIENTO ii

DEDICATORIAS iv

TRIBUNAL DE SUSTENTACIÓN v

DECLARACIÓN EXPRESA vi

RESUMEN vii

ÍNDICE GENERAL ix

ÍNDICE DE FIGURAS xii

IABREVIATURAS xviv

NTRODUCCIÓN xivi

CAPITULO 1 1

1. GENERALIDADES 1

1.1 Objetivos 1

1.1.1 Objetivos Generales 1

1.1.2 Objetivos Específicos 1

1.2 Alcance y Limitaciones del Proyecto 2

CAPITULO 2 4

2. MARCO TEÓRICO 4

2.1. INTRODUCCIÓN A QUARTUS II Y LENGUAJE VHDL 4

2.2. INTRODUCCIÓN A ECLIPSE 5

2.3. NIOS II SBT PARA ECLIPSE 6

2.4. TARJETA DE DESARROLLO ALTERA DE2 7

2.5. FPGA’s 8

CAPITULO 3 15

3. DISEÑO E IMPLEMENTACION 15

3.1. Almacenamiento de Imagen 16

3.2. Procesamiento de Imagen Matriz 19

3.3. Suavizado de Imagen 256

3.4. Procesamiento de Detección de Borde de Imagen 28

CAPITULO 4 32

4.1. Matriz Resultante de la Imegen con sus Bordes Detectados 32

4.2. Visualización de Imagen Resultante 32

CONCLUSIONES 35

RECOMENDACIONES 37

RECOMENDACIONES 42

ÍNDICE DE FIGURAS

Figura 2.1 NIOS II 9.1 for Eclipse ……..6

Figura 2.2 Kit Tarjeta DE2 de ALTERA……..7

Figura 2.3 Periféricos y E/S, Tarjeta Altera DE2……..8

Figura 2.4 Cyclone FPGA 2C35……9

Figura 2.5 NIOS II System……11

Figura 2.6 Procesador NIOS II……..12

Figura 3.1 Diseño de Algoritmo Detector de Bordes……..16

Figura 3.2 DE2 Control Panel……17

Figura 3.3 NIOS II, Agregando el .sof para DE2 Altera……18

Figura 3.4 DE2 Control Panel, Almacenamiento en SDRAM……..18

Figura 3.5 Proceso de Almacenamiento en SDRAM……19

Figura 3.6 NIOS II 9.1 Software Build Tools for Eclipse……20

Figura 3.7 Creación de Nuevo Proyecto en NIOS II……..20

Figura 3.8 Creación de Nuevo Proyecto en NIOS II , agregando el .sopcinfo……21

Figura 3.9 Nuevo Proyecto en NIOS II……..22

Figura 3.10 NIOS II, Agregando el .sof para Altera DE2……30

Figura 3.11 Compilación de Proyecto……30

Figura 3.12 Compilar en NIOS II Hardware……31

Figura 4.1 Imagen Resultante……32

ABREVIATURAS

ADC	Analog-to-Digital Converter
ASIC	Application Specific Integrated Circuit
CMOS	Complementary metal–oxide–semiconductor
DE2	Development and Education Board
ECJ	Eclipse *Compiler* for Java
EIA	Electronics Industry Association
E/S	Entrada y Salida
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GUIDE	Graphical User Interface Development Environment
HDL	Hardware Description Language
JTAG	Joint Test Action Group
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
OSGi	Open Services Gateway Initiative

PCB	Printed Circuit Board
PIC	Peripheral Interface Controller
RS-232	Recommended Standard 232
SBT	Build tool for Scala
SDRAM	Synchronous Dynamic Random Access Memory
SOPC	System on a Programmable Chip
SWT	Standard Widget Toolkit
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VGA	Video Graphics Adapter
VHDL	Very High Description Languaje
VHSIC	Very High Speed Integrated Circuit

INTRODUCCIÓN

En la actualidad, el diseño e implementación de los circuitos digitales ha cobrado mucha importancia en los programas de estudio de ingenierías y es parte fundamental de la formación básica del estudiante. La evolución en este tipo de dispositivos ha venido presentando cambios importantes para el desarrollo de la tecnología.

Actualmente los FPGA's representan dispositivos versátiles, de fácil programación y precio accesible. Una vez que los FPGA´s estuvieron al alcance de las universidades, muchas de ellas empezaron a incorporar en sus pensums de estudio cursos basados en el uso y diseño utilizando FPGA´s en carreras tecnológicas afines.

En el área de procesamiento de imágenes, la detección de los bordes de una imagen es de suma importancia y utilidad, pues facilita muchas tareas, entre ellas, el reconocimiento de objetos y la segmentación de regiones. Históricamente, se han desarrollado variedad de algoritmos que ayudan a detectar bordes en una imagen, entre los más conocidos tenemos: el Algoritmo de Prewitt, el Algoritmo Laplaciano, el Algoritmo de Sobel, etc.

CAPITULO 1

1. GENERALIDADES

En este capítulo se abarca el planteamiento del proyecto, se describe cuáles son sus objetivos, alcances y limitaciones.

1.1 Objetivos

1.1.1 Objetivos Generales

Desarrollar un código en NIOS II capaz de detectar el borde de una imagen que ha sido almacenada en la memoria SDRAM de un FPGA.

1.1.2 Objetivos Específicos

- Aprender el uso y manejo de la DE2 ALTERA.
- Estudiar la eficiencia de algoritmos para la detección de bordes.
- Implementar un proyecto que demuestre en base a los resultados la capacidad del algoritmo utilizado para detectar bordes.

- Aprender como se codifica la información que posee el formato bmp.

1.2 Alcances y Limitaciones del Proyecto

Entre los alcances del proyecto se tiene:

- ✓ Almacenamiento de la imagen en la SDRAM de la Altera DE2.
- ✓ Lectura de los datos en un lenguaje de alto nivel.
- ✓ Estructuración de la matriz de la imagen a ser procesada.
- ✓ Obtener una matriz resultante con datos de la imagen con bordes detectados.
- ✓ Habilitar la salida VGA de la tarjeta DE2 Altera, para proceder a mostrar la imagen resultante.

Entre las limitaciones se tienes las siguientes:

- ✓ Se cuenta con un solo método para guardar las imágenes en la SDRAM, el mismo que es por medio del programa DE2 CONTROL PANEL.
- ✓ Se usa imágenes en formato BMP de 256 colores debido a la factibilidad de trabajar con sus datos, ya que este tipo de formato usan 8 bits por pixel, es decir que necesitan 1 byte para codificar cada pixel, además este formato presenta información del tamaño de imagen, tamaño de archivo, ancho (pixel), alto (pixel), etc. de tal manera que la información necesaria se encuentra en una posición

determinada, con lo que se puede formar una matriz de datos de la imagen.

CAPITULO 2

2. MARCO TEÓRICO

En este capítulo se abarcará los conceptos básicos y se hará una pequeña introducción a los programas y tecnologías que fueron se utilizaron durante el desarrollo del proyecto.

2.1. INTRODUCCIÓN A QUARTUS II Y LENGUAJE VHDL

Quartus es una herramienta de software producida por Altera para el análisis y la síntesis de diseños realizados en HDL. Dentro de las funciones que puede realizar Quartus, está el análisis de circuitos lógicos.

VHDL es un lenguaje usado para describir sistemas electrónicos digitales y es el acrónimo que representa la combinación de VHSIC y HDL. Este lenguaje permite la descripción de la estructura de un sistema, es decir, la forma en que se descompone en subsistemas y como estos subsistemas están interconectados. Además, permite la especificación de la función de un sistema mediante un lenguaje de programación. VHDL permite el diseño de un sistema para ser simulado antes de ser fabricado.

Los diseñadores pueden rápidamente comparar alternativas poniendo a prueba las correcciones sin los gastos asociados a la creación de un prototipo de hardware.

2.2. INTRODUCCIÓN A ECLIPSE

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse (y que son usados también para desarrollar el mismo Eclipse). Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus. [12]

Eclipse se encuentra estructurado de la siguiente manera:

- ✓ Plataforma principal para el inicio de Eclipse y ejecución de Plug-in’s.
- ✓ OSGi, permite diseñar plataformas compatibles que puedan proporcionar múltiples servicios.
- ✓ El SWT, que es un Widget Toolkit portable.
- ✓ JFace, para manejo de archivos, manejos de texto, editores de texto, etc.
- ✓ Workbench de Eclipse, para vistas, editores, perspectivas, asistentes, etc.

Este software usa lenguajes de programación como C/C++ y Phython, además de permitir trabajar con lenguajes para procesamiento de texto como LaTeX, aplicaciones de red como Telnet y sistema de gestión de base de datos, además de proveer al programador con frameworks para el

desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.

La definición que da el proyecto Eclipse acerca de su software es: "una especie de herramienta universal - un IDE abierto y extensible para todo y nada en particular".

2.3. NIOS II SBT PARA ECLIPSE

Nios II SBT para Eclipse es una pequeña capa de interfaz gráfica de usuario que ayuda a presentar un entorno unificado de desarrollo. El SBT para Eclipse proporciona una plataforma de desarrollo que funciona para todos los procesadores Nios II.

Se puede llevar a cabo todas las tareas de desarrollo de software dentro de Eclipse, incluyendo creación, edición, construcción, funcionamiento, depuración y perfilado de programas. Nios II SBT para Eclipse se basa en la estructura de Eclipse Framework y Eclipse Development Toolkit (CDT) Plugins.

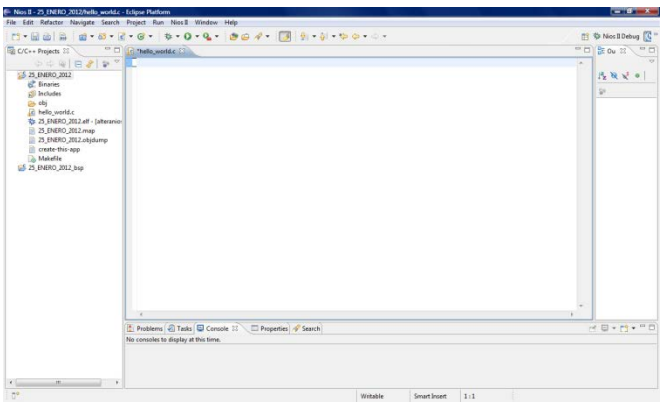

Figura 2.1 NIOS II 9.1 for Eclipse

2.4. TARJETA DE DESARROLLO ALTERA DE2

La tarjeta de desarrollo de Altera DE2 fue diseña con fines educacionales, debido a que es una creación de profesores, dirigida para profesores, investigadores y estudiantes.

La razón por la cual tiene fines educativos, es debido a que posee una amplia gama de aplicaciones libres y con demos gratis que se incluyen en el disco de la tarjeta DE2 ALTERA, para la fácil comprensión de lógica digital, organización de computadoras y FPGA`s. Además de poseer una amplia lista de ejercicios, donde también se pueden desarrollar tareas simples que ayudan a la fácil comprensión de conceptos fundamentales de cada uno de los distintos segmentos de la tarjeta. También se pueden llegar a realizar proyectos más complejos y avanzados.

Figura 2.2 Kit Tarjeta DE2 de ALTERA

2.5. FPGA's

Una FPGA es un dispositivo semiconductor que contiene bloques lógicos, cuya interconexión y funcionalidad se puede programar. La lógica programable puede reproducir desde funciones tan sencillas como las de una puerta lógica hasta complejos sistemas en un chip (SoC).

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

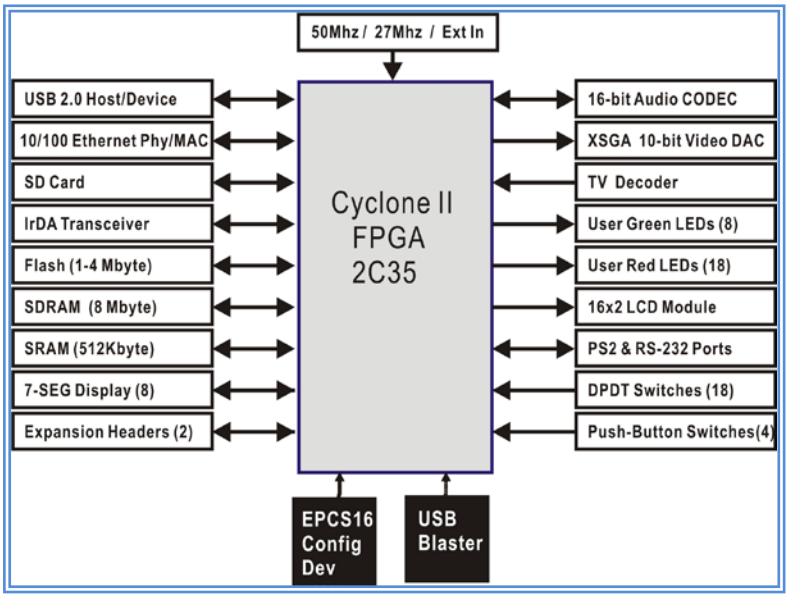

Figura 2.4 Cyclone FPGA 2C35 [2]

Los principales beneficios de los FPGAs son los siguientes:

- Flexibilidad y capacidad de desarrollo
- Precio
- Fiabilidad
- Reconfigurables

Hoy las FPGA’s están presentes en campos tan diversos como la electrónica de consumo, o la investigación espacial. La tecnología FPGA tiene una aplicación horizontal en todas las industrias que requieren computación a alta velocidad.

Tiene cabida en empresas dedicadas a la fabricación de sistemas electrónicos para: climatización de autobuses, comunicaciones por fibra óptica, conducción automática de trenes, control industrial, etc.

2.6. Procesador Embebido NIOS II

El Procesador Embebido Nios II es un sistema flexible capaz de ajustarse a las necesidades del diseñador, esta flexibilidad es gracias a la tecnología del SOPC builder que tiene como ventaja integrar en un mismo chip una o varias CPUs y sus periféricos asociados, lo cual optimiza el sistema.

El Sopc builder brinda un entorno específico permitiendo la definición y configuración a medida del microprocesador NIOS II y que por medio de la herramienta Quartus II puede ser directamente utilizado sobre el FPGA.

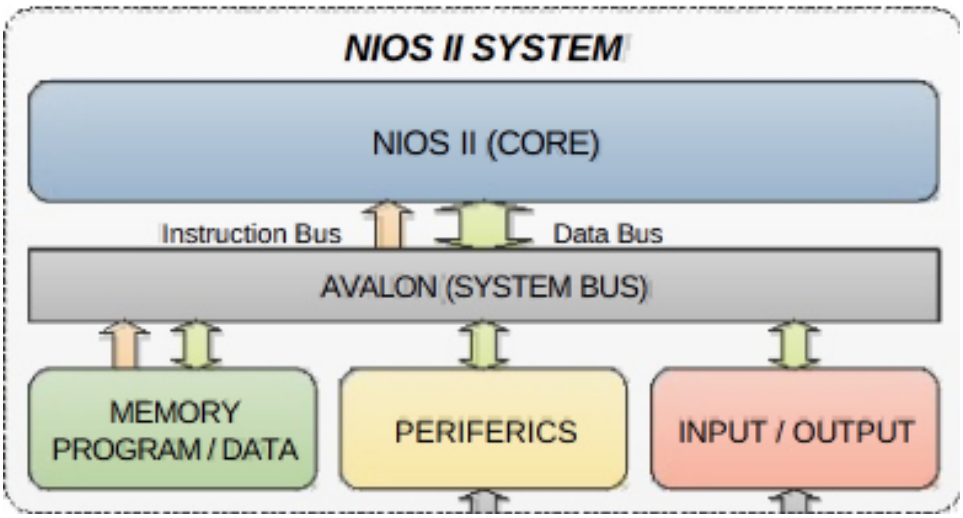

Figura 2.5 NIOS II System [13]

NIOS II es un procesador cuya característica principal es que posee un grupo de registros de propósito general de 32 bits. Está compuesto por: el núcleo procesador NIOS II, memoria interna de programa y de datos, periféricos integrados e interfaces para memoria externa y/o entrada/salida.

Existen tres versiones del procesador Nios II, las cuales tienen la misma arquitectura de instrucciones de 32 bits:

- NIOS II /f (rápido): Es de alto rendimiento, con pipeline (conjunto de elementos procesadores conectados en serie) de 6 etapas aumenta su

desempeño con opciones específicas como: memorias caché de instrucciones y datos o una unidad de manejo de memoria.

- NIOS II /s (estándar): Tiene un *pipeline* de 5 etapas que con una unidad aritmética lógica busca combinar rendimiento y consumo de recursos.
- NIOS II /e (económico): Requiere de menos recursos de la FPGA, no posee *pineline* y es muy limitada porque carece de las operaciones de multiplicación y división.

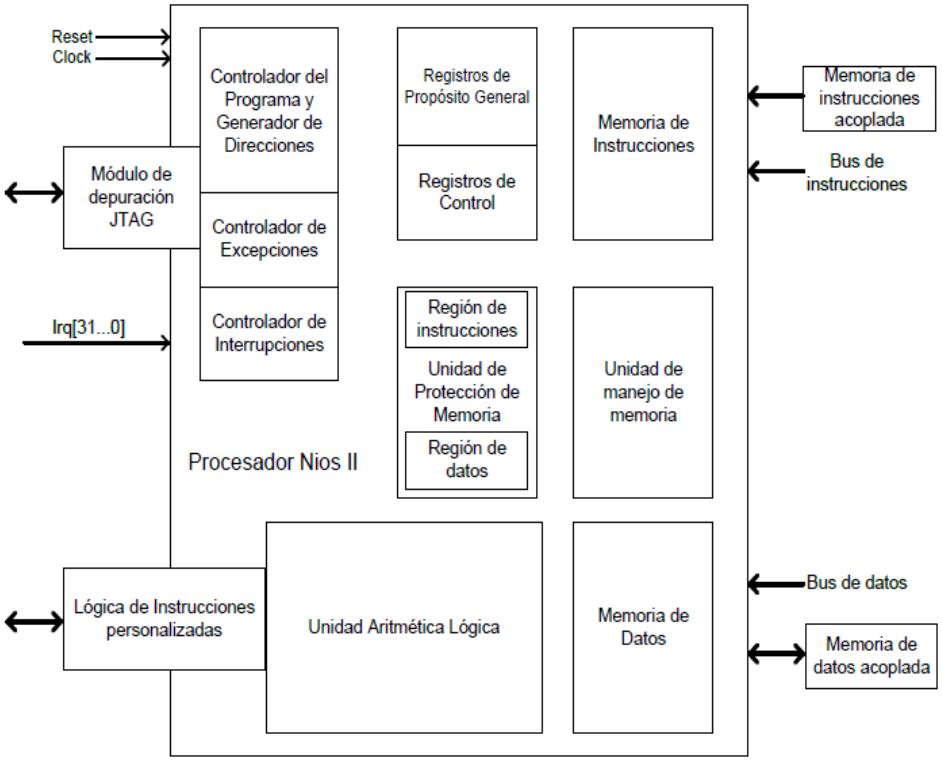

Figura 2.6 Procesador NIOS II[13]

El Nios II está formado por unidades funcionales dependiendo de la versión implementada, entre las fundamentales tenemos: registros, unidad

aritmético-lógica, interfaz para instrucciones definidas por el usuario, controlador de excepciones, bus de instrucciones, bus de datos, memoria caché de instrucciones y datos, interfaz de acoplamiento directo de memoria de instrucciones y datos y módulo de depuración JTAG.

2.7. Procesamiento Digital de Imágenes

Actualmente el procesamiento Digital de Imágenes es una herramienta muy específica en computación, se puede definir como las técnicas que se aplican a las imágenes digitales. Este procedimiento se utiliza en dos importantes áreas que son:

- mejorar la calidad de la información de una imagen
- procesamiento de los datos de una imagen de forma autónoma.

Para el procesamiento de la imagen es necesario tratar a la imagen como una matriz de valores numéricos, cada elemento de la matriz representa un pixel. Los valores de cada pixel se encuentran entre 0 y 255, en donde el 0 corresponde al negro que es el tono más oscuro y el 255 corresponde al blanco que es el tono más claro.

2.8. Detección de Bordes

Los bordes de una imagen son las secciones en donde ocurre un cambio brusco de intensidad, es decir en donde existe una transición entre dos

partes que tienen diferentes niveles de gris. El resultado de este proceso sirve como entrada para tareas adicionales como reconocer objetos, compresión de imágenes, registro y alineación.

Hasta el momento se cuenta con varios algoritmos de detección de bordes, entre los más simples están:

- Algoritmo de Sobel
- Algoritmo de Prewitt
- Algoritmo Laplaciano

2.8.1. Algoritmo de Sobel y Prewitt

Ambos algoritmos son operadores basados en la primera derivada (gradiente). Detectan el borde de una imagen, calculando el gradiente de la misma en dos direcciones ortogonales

Los dos operadores pueden formularse de forma conjunta con las siguientes máscaras de convolución mostradas a continuación:

Gradiente fila

$$\frac{1}{2+k}$$

1	0	-1
K	0	-K
1	0	-1

Gradiente columna

$$\frac{1}{2+k}$$

-1	-K	-1
0	1	0
1	K	1

- En el operador Prewitt (K=1) se involucran a los vecinos de filas/columnas adyacentes para proporcionar mayor inmunidad al ruido.

- El operador Sobel (K=2), se supone que es más sensible a los bordes diagonales que el de Prewitt aunque en la práctica hay poca diferencia entre ellos.

2.8.2. Algoritmo Laplaciano

El Laplaciano es un operador basado en la segunda derivada. El requerimiento básico en la definición del Laplaciano digital es que los coeficientes asociados con el pixel central sean positivos y que coeficientes asociados con los pixeles externos sean negativos. Porque el Laplaciano es una segunda derivada, la suma de los coeficientes tiene que ser cero. Por lo tanto la respuesta es cero siempre que el punto tratado y sus linderos tengan el mismo valor.

La máscara de convolución más usada es la siguiente:

0	1	0
1	-4	1
0	1	0

CAPITULO 3

3. DISEÑO E IMPLEMENTACION

En este capítulo se explica cómo se diseñó el proyecto y cuáles fueron los parámetros utilizados para poder implementarlo.

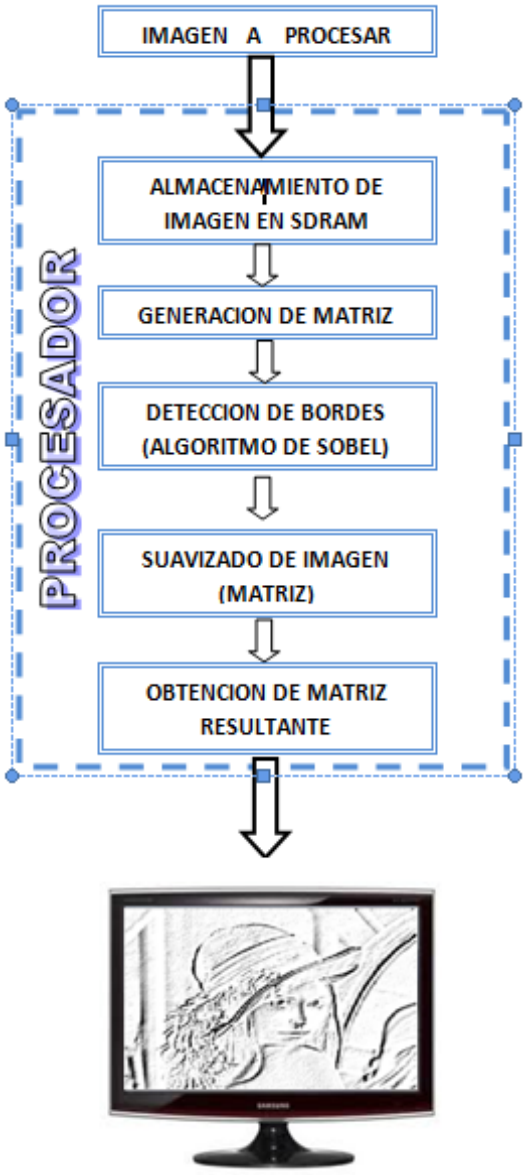

Figura 3.1 Diseño de Algoritmo Detector de Bordes

3.1. Almacenamiento de Imagen

En esta sección se hizo uso del disco que viene incluido dentro del Kit de la Tarjeta Altera DE2, se usó el programa DE2_Control_Panel que facilitó el almacenamiento de la imagen dentro de la SDRAM, además que permite especificar en qué sección de memoria dentro de la SDRAM, se quiere comenzar a almacenar la imagen y al finalizar el almacenamiento nos indica el tamaño de la misma.

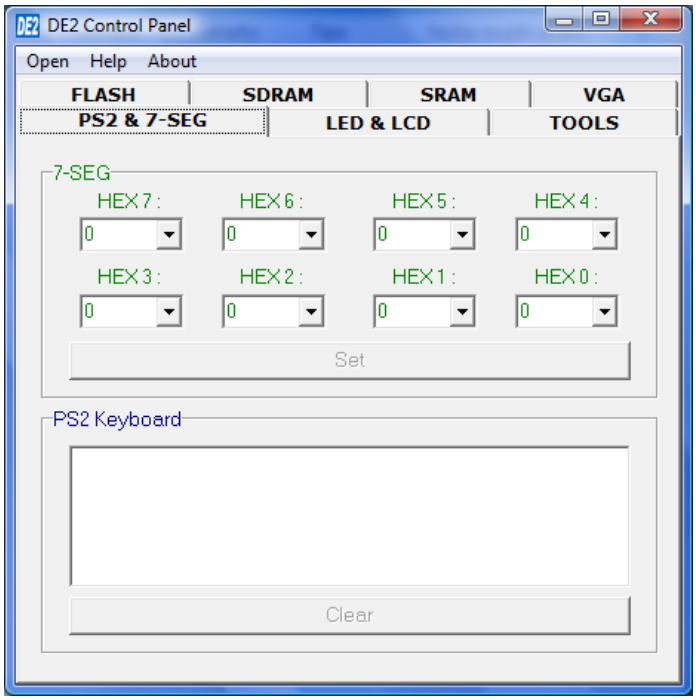

Figura 3.2 DE2 Control Panel

En la Fig. 3.2 se muestra el programa DE2_Control_Panel,

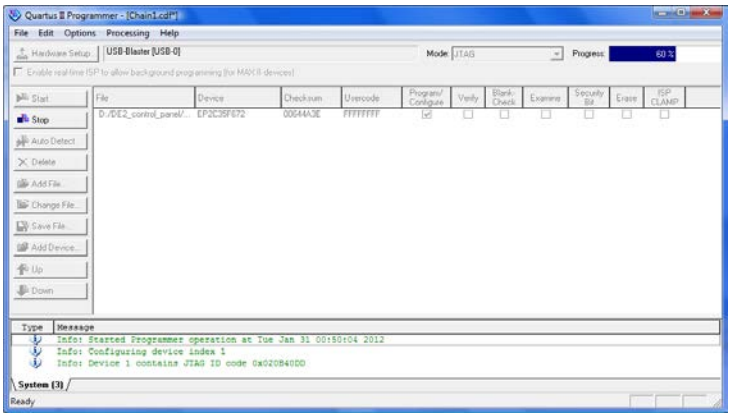

Figura 3.3 NIOS II, Agregando el .sof para Altera DE2

Para poder almacenar la imagen por medio del programa DE2_Control_Panel, se debe habilitar los periféricos I/O por medio del DE2_Control_panel.sof, cuyo archivo lo encontramos en el disco que viene incluido con la tarjeta DE2 ALTERA, debido a que contiene los periféricos necesarios para el almacenamiento de la imagen en la SDRAM, una vez habilitado los periféricos para usar el software en mención como se muestra en la figura 3.3, se puede proceder a almacenar la imagen a procesar.

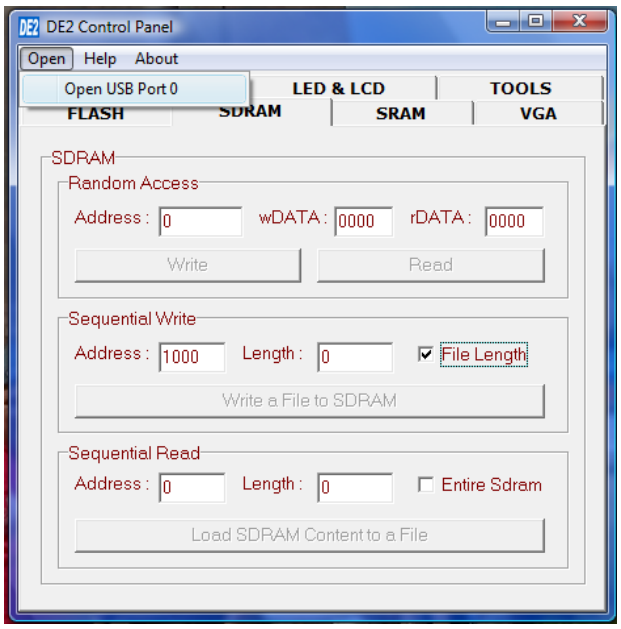

Figura 3.4 DE2 Control Panel, Almacenamiento en SDRAM

En la Fig. 3.4 mostramos la imagen del programa DE2_Control_Panel, se procede a dar clic en Open para habilitar el puerto USB y poder guardar la imagen, ubicándose sobre la pestaña SDRAM.

En la sección de *Sequential Write*, procedemos a seleccionar "*file lenght*", y en el campo *Address* escribimos la dirección donde se va a comenzar a guardar la imagen, en nuestro caso utilizamos la dirección 0x1000, una vez llenado estos campos damos clic sobre *WRITE A FILE TO SDRAM*, y seleccionamos la imagen que se va a procesar, de esta manera la imagen seleccionada se irá almacenando secuencialmente en la SDRAM desde la dirección indicada.

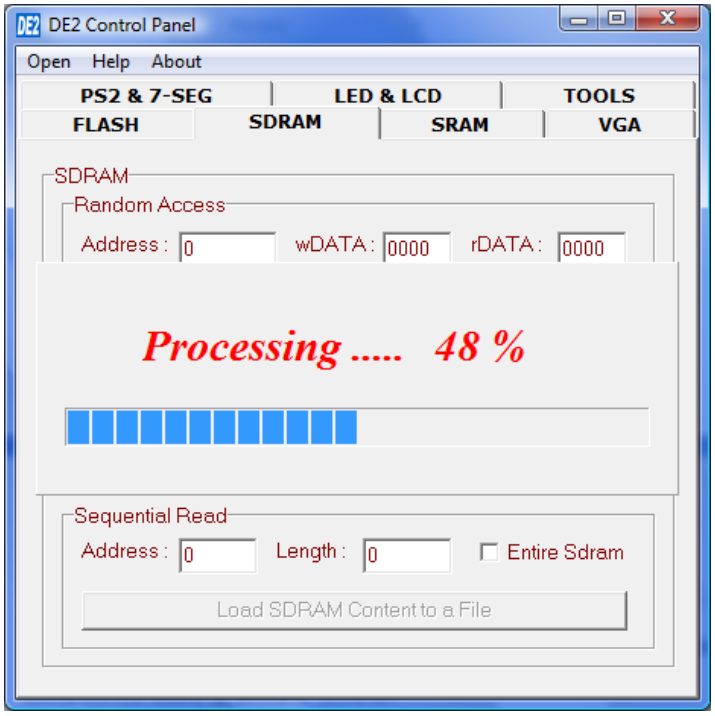

Figura 3.5 Proceso de Almacenamiento en SDRAM

3.2. Procesamiento de Imagen Matriz

Una vez almacenada la imagen en la SDRAM, procedemos a iniciar NIOS II 9.1 Software Build Tools for Eclipse.

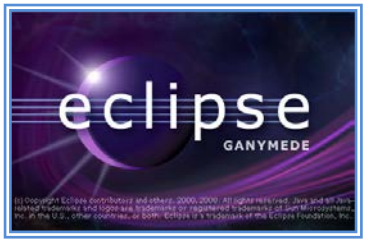

Figura 3.6 NIOS II 9.1 Software Build Tools for Eclipse [14]

Para poder cumplir con el propósito del proyecto una vez iniciado el programa procedemos a crear un nuevo proyecto, para esto vamos a:

File->New-> Nios II Application and BSP From Template

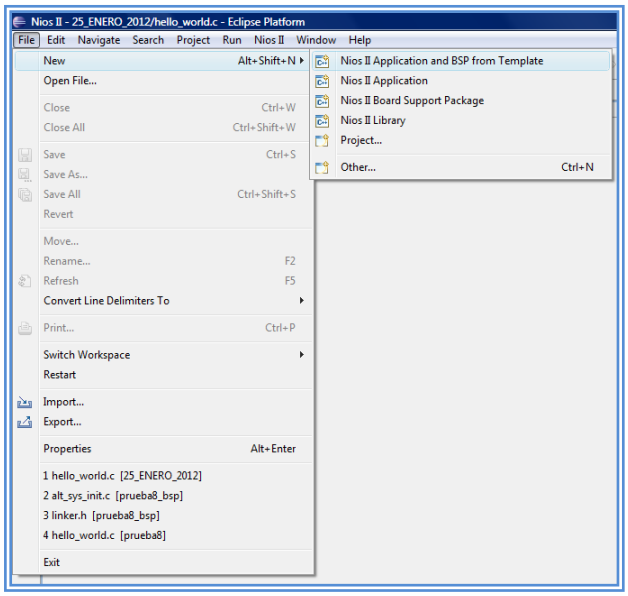

Figura 3.7 Creación de Nuevo Proyecto en NIOS II

Luego que se abra la ventana correspondiente, hay que ingresar el archivo .sopcinfo, que es donde se almacena la información del hardware, en este caso se ha usado la configuración DE2 Media Computer de Altera, que se encuentra localizada en la siguiente dirección:

C:\altera\91\University_Program\NiosII_Computer_Systems\DE2\DE2_Media_Computer.

Luego se le asigna un nombre al proyecto y se da click en **NEXT,** como se muestra en la figura.

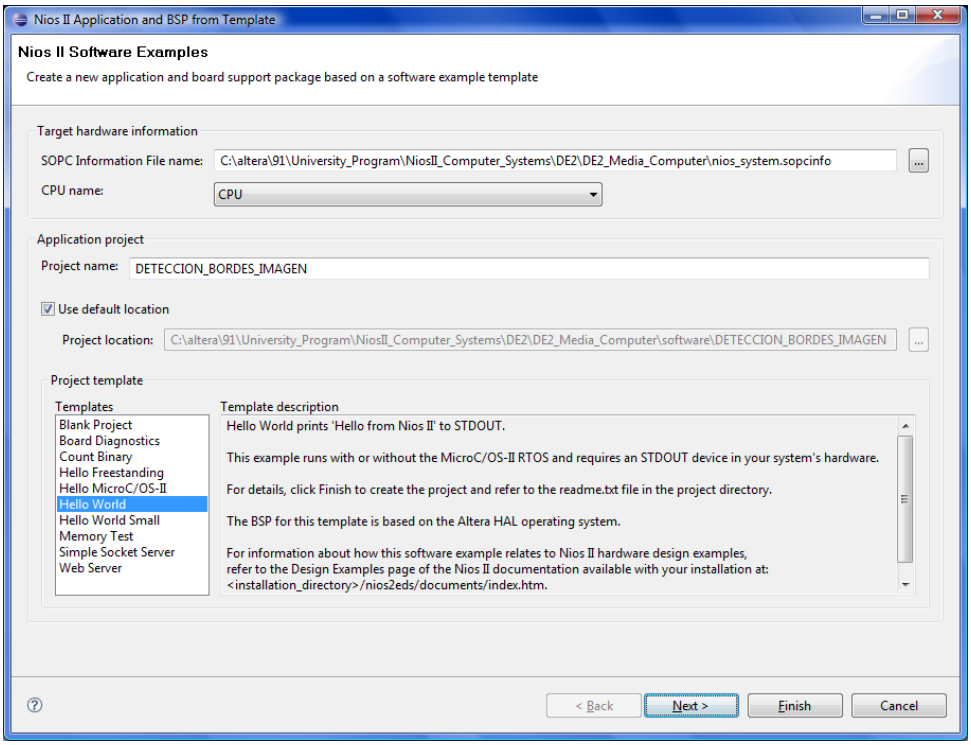

Figura 3.8 Creación de Nuevo Proyecto en NIOS II , agregando el .sopcinfo

En el siguiente recuadro se procede a dar click en **FINISH** y de esta manera se inicia con la programación.

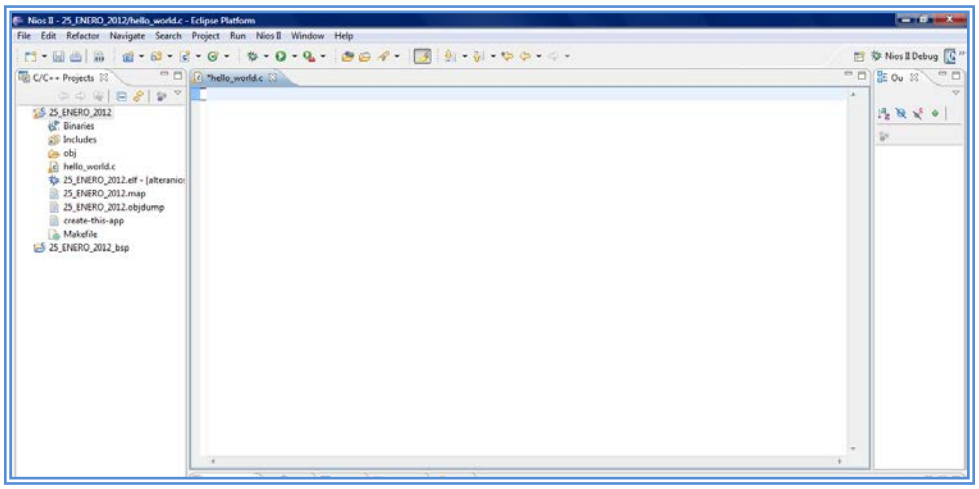

Figura 3.9 Nuevo Proyecto en NIOS II

Una vez que se tiene el nuevo proyecto creado, se empieza a agregar las librerías que se va a utilizar.

```
#include <stdio.h>
#include <alt_types.h>
#include <io.h>
#include <system.h>
```

En la sdram la imagen es almacenada en memoria de una forma especial, es decir que cuando se guarda en la posición 0x1000 como se observa en la ***Figura 3.4***, en realidad el archivo comienza a ser almacenado en memoria desde la posición 0x4000 cuyo valor decimal es 16384. El número de filas se encontrará almacenado desde la posición 16406 hasta la 16407, y el número de columnas desde la posición 16402 hasta la 16403. A continuación se muestran las funciones utilizadas para calcular el número de filas y columnas que posee la imagen almacenada en memoria:

FUNCION GENÉRICA: IORD_8DIRECT(BASE, OFFSET)

```
int f =     (IORD_8DIRECT((SDRAM_BASE),16406))+
          (256*IORD_8DIRECT((SDRAM_BASE), 16407));
int c =   (IORD_8DIRECT((SDRAM_BASE), 16402))+
          (256*IORD_8DIRECT((SDRAM_BASE), 16403));
```

Posteriormente se empieza a leer los datos de la imagen que son necesarios para detectar el borde de la imagen, los mismos que se encuentran a partir de la posición 18486. Además la imagen se encuentra distribuida internamente en memoria en saltos de 512, es decir desde donde empieza a almacenarse la imagen (16384) hasta la posición 16895 se van a encontrar los primeros 512 datos(pixeles), luego en las siguientes 512 posiciones vamos a encontrar datos que no pertenecen a la información de la imagen (16896 – 17407), de igual manera desde la posición 17408 hasta la posición 17920 nuevamente se encuentran almacenados datos(pixeles)

correspondientes a la imagen, y asi sucesivamente hasta terminar con la lectura de todos los datos de la imagen dependiendo del tamaño de la imagen.

Debido a que los datos que contienen la información de la imagen se encuentran almacenadas a partir de la posición 18486, los primeros datos que obtendremos no serán 512 sino 458, es decir que se empezará a leer desde la posición 18486 hasta la posición 18944.

Por tal razón el código para obtener solo los datos válidos lo realizamos de la siguiente manera:

```
start = 18485;
if(tam<=458){
        tope=start+tam;
        for(i=(start+1); i<=tope ; i++){
        hexadecimal[h] = IORD_8DIRECT((SDRAM_BASE), i);
  h++;
  res=0;
        }
  }
  if(tam>458){
   tope=start+458;
          for(i=(start+1); i<=tope ; i++){
                hexadecimal[h] = IORD_8DIRECT((SDRAM_BASE), i);
```

```
                h++;
            }
            res=tam-458;
            start=tope+512;
 }
 while(res>0){
    if(res>512){
            tope=start+512;
            for(i=(start+1); i<=tope ; i++){
                hexadecimal[h]=IORD_8DIRECT((SDRAM_BASE), i);
                    h++;
    }
    res=res-512;
    start=tope+512;
    }
    else{
            tope=start+res;
            for(i=(start+1); i<=tope ; i++){
                hexadecimal[h] = IORD_8DIRECT((SDRAM_BASE), i);
                h++;
                res=0;
     }
  }
}
matriz = crearMatriz(hexadecimal,f,c);
```

Con este procedimiento se obtiene la matriz de los valores válidos para luego realizar el suavizado de la imagen.

3.3. Suavizado de Imagen

El objetivo del suavizado es aumentar o disminuir ciertas características que se encuentran presentes en la imagen, el filtro que hemos usado está basado en la convolución de una máscara y la media aritmética que tiene como fin reducir presencia de ruido, es decir eliminar componente de altas frecuencias y mantener las de bajas frecuencias.

```
tam = 3;
radio = (tam-1)/2;
matrizReflejada2 = reflejarMatriz(matriz,f,c);
fila=f;
columna=c;
f=f+2;
c=c+2;
for(i=(1+radio);i<(f-radio);i++){
  for(j=(1+radio);j<(c-radio);j++){
          fil=1;
          for(a=(i-radio);a<=(i+radio);a++)  {
                  col=1;
```

```
                for(b=(j-radio);b<=(j+radio);b++)  {
                        matrizAux[fil][col] = matrizReflejada2[a][b];
                                col++;
                }
                fil++;
        }
        sumaentera=0;
        suma=0;
        for(ma=1;ma<=tam;ma++){
                for(mb=1;mb<=tam;mb++){
                        suma=suma+matrizAux[ma][mb]*W[ma][mb];
                }
        }
        suma= (int)suma;
        sumaentera=suma;
        G[i-radio][j-radio]=sumaentera;
    }
}
```

3.4. Procesamiento de Detección de Borde de Imagen

El procesamiento de detección de bordes se podría decir que es nuestro último paso, antes de mostrar los resultados finales en pantalla, Para realizar este paso de detección nos basarernos en el principio del operador de Sobel, debido a que entre los operadores en procesamiento Digital de Imágenes para la detección de bordes en Imágenes es uno de los más usados.

```
matrizReflejada = reflejarMatriz(G,f,c);
matrizTransformada = malloc(sizeof(int) * (f));
for(i=0;i<f;i++)
{
matrizTransformada[i] = malloc(sizeof(int) * (c));
}
    for(i=0;i<f;i++)
     {
        for(j=0;j<c;j++)
        {
            sumFil= (matrizReflejada[i+2][j] + 2*matrizReflejada[i+2][j+1] +
            matrizReflejada[i+2][j+2]) - (matrizReflejada[i][j] +
            2*matrizReflejada[i][j+1] + matrizReflejada[i][j+2]);
            sumCol= (matrizReflejada[i][j+2] + 2*matrizReflejada[i+1][j+2] +
            matrizReflejada[i+2][j+2]) - (matrizReflejada[i][j] +
            2*matrizReflejada[i+1][j] + matrizReflejada[i+2][j]);
            matrizTransformada[i][j]=sumFil+sumCol;
          }
      }
```

```
 for(i=0;i<f;i++)
{
    for(j=0;j<c;j++)
    {
        if(255-matrizTransformada[i][j]>255)
            matrizTransformada[i][j]=255;
        else if(255-matrizTransformada[i][j]<0)
            matrizTransformada[i][j]=0;
        else
            matrizTransformada[i][j]=255-matrizTransformada[i][j];
      }
   }
```

Una vez que hemos terminado de programar tenemos que ir a:

NIOS II->QUARTUS II PROGRAMMER

En esa ventana tenemos que añadir la DE2_Media_computer.sof, cuyo archivo lo encontramos en el programa universitario de Altera para la tarjeta DE2, en este caso se ha utilizado la DE2 MEDIA COMPUTER debido a que contiene los periféricos necesarios para el proyecto, luego damos clic en **START**, esperamos que se ejecute al 100% y volvemos a la ventana de NIOS II.

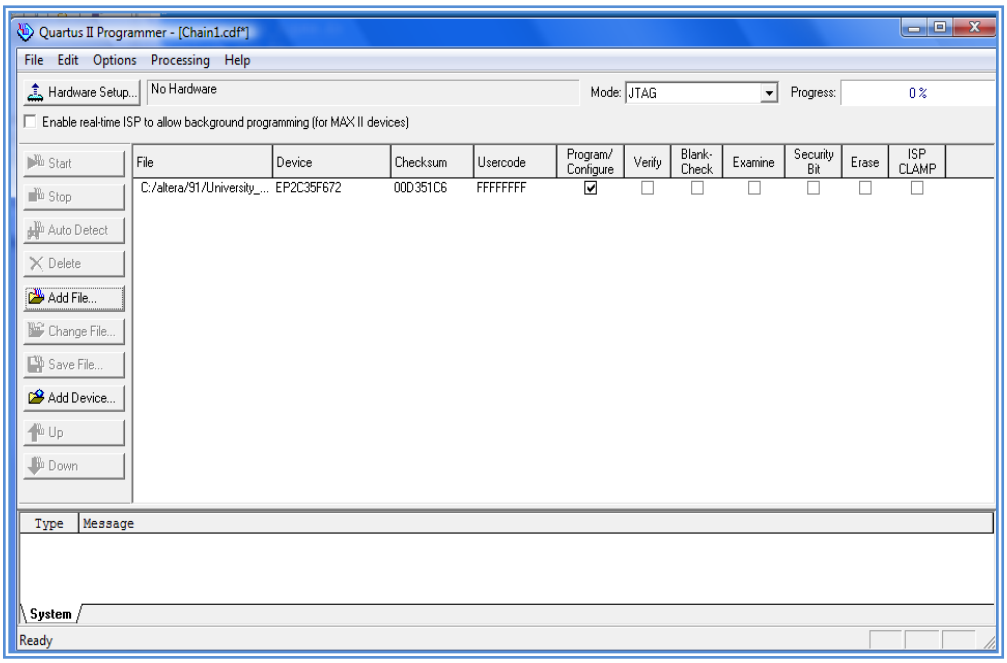

Figura 3.10 NIOS II, Agregando el .sof para Altera DE2

Una vez realizado este paso, procedemos a ejecutar nuestro proyecto en el procesador NIOS II de la FPGA dando clic sobre RUN, como se muestra en la figura.

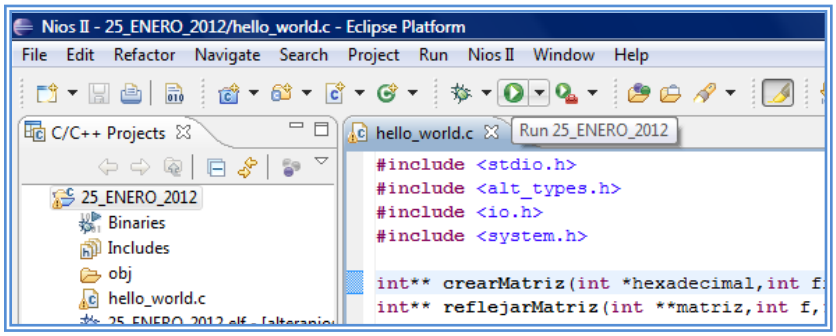

Figura 3.11 Compilación de Proyecto

A continuación de realizar este paso nos aparecerá una ventana donde debemos seleccionar NIOS II Hardware y el programa empezara a ejecutarse.

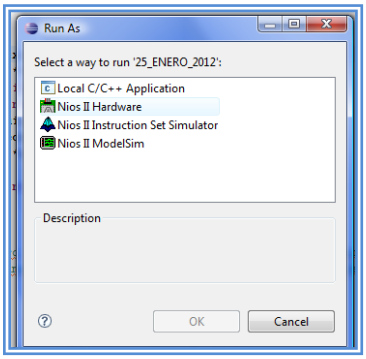

Figura 3.12 Compilar en NIOS II Hardware

CAPITULO 4

4.1. Matriz resultante de la imagen con sus bordes detectados

Una vez obtenida la matriz resultante con la información de la imagen con los bordes detectados gracias al Operador de Sobel, procedemos a mostrar la imagen por medio de la salida VGA de la tarjeta DE2 ALTERA.

4.2. Visualización de Imagen Resultante

Figura 4.1 Imagen resultante

Para poder mostrar la imagen resultante en pantalla, necesitaremos habilitar la salida VGA, recordando que esta salida soporta hasta una resolución de 320x240, en este código lo primero que se realiza es un limpiado de pantalla, la manera como realiza el limpiado de pantalla es recorriendo toda la pantalla y asignándole el color blanco “0”, debido que para que resalten los otros colores es suficiente tener una pantalla de fondo blanco, de otra manera, sino se realizaba este paso se iba a mostrar un fondo de múltiples colores debido a que siempre hay datos en memoria y esto hace que se muestre una pantalla no apta para trabajar, finalmente procedemos a llenar la pantalla con los resultados obtenidos.

clear_screen(); //Limpiado de pantalla, poniendo todo el fondo de color blanco

```
 int x,y;
 color=0;
 for (x=0;x<f;x++)
 {
    for(y=0;y<c;y++)
    {
  if(matrizTransformada[x][y]>100)
     color=0xffff;
    else
    color=0x0000;
    write_pixel(y,x, color);
     }
 }
```

void clear_screen()

```
{
    int x, y;
    for (x = 0; x < 320; x++)
     {
            for (y = 0; y < 240; y++)
    {
                    write_pixel(x,y,0);
            }
    }
 }

void write_pixel(int x, int y, int colour)
{
  volatile short *vga_addr=(volatile short*)(0x08000000);
  *vga_addr=colour;
}
```

CONCLUSIONES

1. Se pudo desarrollar un programa para NIOS II que facilitó la respectiva visualización y comparación de resultados en las distintas etapas del proyecto.

2. El formar nuevas matrices para poder trabajar con la detección de bordes fue más que necesario, debido a que el manejo de datos desde la sdram donde se encontraban inicialmente daba dificultades por la forma como se encontraba almacenada.

3. La calidad de la imagen resultante no es tan precisa debido a que se está usando imágenes en escala de grises formato BMP de 256 colores, lo que indica que se están usando 8 bites por pixel .

4. La unidad de almacenamiento SDRAM fue una gran limitación en el desarrollo de nuestro proyecto debido a su capacidad de almacenamiento para guardar datos, siendo esta una de las razones principales por la cual se usaron imágenes de formato BMP de 8 bits (256 colores).

5. El uso del suavizado de imagen en conjunto con el umbral, nos fue de suma ayuda para poder eliminar ligeramente el ruido que se presentaba en la imagen, logrando así de obtener de una manera un poco más precisas los bordes de imágenes.

RECOMENDACIONES

1. Se recomienda usar imágenes pequeñas para el procesamiento de imágenes debido a la capacidad de almacenamiento, ya que internamente adicional al almacenamiento de imagen hay que tomar en cuenta que también se ocupa espacio en memoria para el desarrollo de las matrices y proceso de detección de bordes.

2. Se recomienda saber con precisión donde se encuentran almacenado cada uno de los datos para poder realizar el respectivo proceso, para nuestro desarrollo del proyecto nos fue de gran ayuda analizar usar un convertidor de imagen a números hexadecimales, ya que con esto nos podíamos guiar si las posiciones que estábamos tomando en consideración eran las indicadas.

3. Se recomienda tener presente que el algoritmo implementado solo sirve para imágenes formato BMP de 256 colores, debido que dependiendo del formato de la imagen el algoritmo cambia, ya que las posiciones de almacenamiento cambian para cada formato tales como el número de filas, número de columnas, cuerpo de la imagen, etc. Adicional a esto para el proceso de detección de bordes en otros formatos de imágenes se recomendaría usar algún medio de almacenamiento externo ya que la cantidad de espacio que ocuparía la imagen seria mayor, por ende el desarrollo del algoritmo

internamente necesitaría mayor memoria para realizar el respectivo procesamiento de imagen.

4. En la SDRAM la información se va guardando en forma de vector por lo que es recomendable guardar los datos en forma de matriz ya que de otra manera no se podrían realizar los cálculos con las máscaras de convolución y el proceso de detección de bordes no se hubiera podido realizar.

ANEXOS

ANEXO A

DE2 Development and Education Board

User Manual

Chapter 1
DE2 Package

The DE2 package contains all components needed to use the DE2 board in conjunction with a computer that runs the Microsoft Windows software.

1.1 Package Contents

Figure 1.1 shows a photograph of the DE2 package.

Figure 1.1. The DE2 package contents.

Figure 1.2. The feet for the DE2 board.

Chapter 2

Altera DE2 Board

This chapter presents the features and design characteristics of the DE2 board.

2.1 Layout and Components

A photograph of the DE2 board is shown in Figure 2.1. It depicts the layout of the board and indicates the location of the connectors and key components.

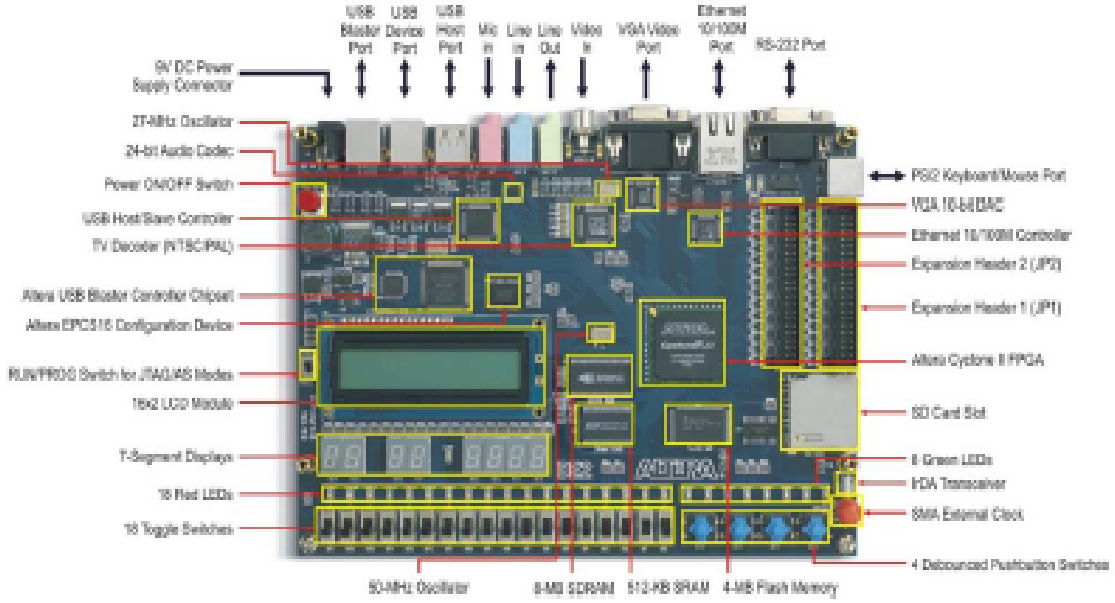

Figure 2.1. The DE2 board.

The DE2 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware is provided on the DE2 board:

- Altera Cyclone® II 2C35 FPGA device
- Altera Serial Configuration device - EPCS16
- USB Blaster (on board) for programming and user API control; both JTAG and Active Serial (AS) programming modes are supported
- 512-Kbyte SRAM
- 8-Mbyte SDRAM

- 4-Mbyte Flash memory (1 Mbyte on some boards)
- SD Card socket
- 4 pushbutton switches
- 18 toggle switches
- 18 red user LEDs
- 9 green user LEDs
- 50-MHz oscillator and 27-MHz oscillator for clock sources
- 24-bit CD-quality audio CODEC with line-in, line-out, and microphone-in jacks
- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL) and TV-in connector
- 10/100 Ethernet Controller with a connector
- USB Host/Slave Controller with USB type A and type B connectors
- RS-232 transceiver and 9-pin connector
- PS/2 mouse/keyboard connector
- IrDA transceiver
- Two 40-pin Expansion Headers with diode protection

In addition to these hardware features, the DE2 board has software support for standard I/O interfaces and a control panel facility for accessing various components. Also, software is provided for a number of demonstrations that illustrate the advanced capabilities of the DE2 board.

In order to use the DE2 board, the user has to be familiar with the Quartus II software. The necessary knowledge can be acquired by reading the tutorials *Getting Started with Altera's DE2 Board* and *Quartus II Introduction* (which exists in three versions based on the design entry method used, namely Verilog, VHDL or schematic entry). These tutorials are provided in the directory *DE2_tutorials* on the DE2 System CD-ROM that accompanies the DE2 board and can also be found on Altera's DE2 web pages.

2.2 Block Diagram of the DE2 Board

Figure 2.2 gives the block diagram of the DE2 board. To provide maximum flexibility for the user, all connections are made through the Cyclone II FPGA device. Thus, the user can configure the FPGA to implement any system design.

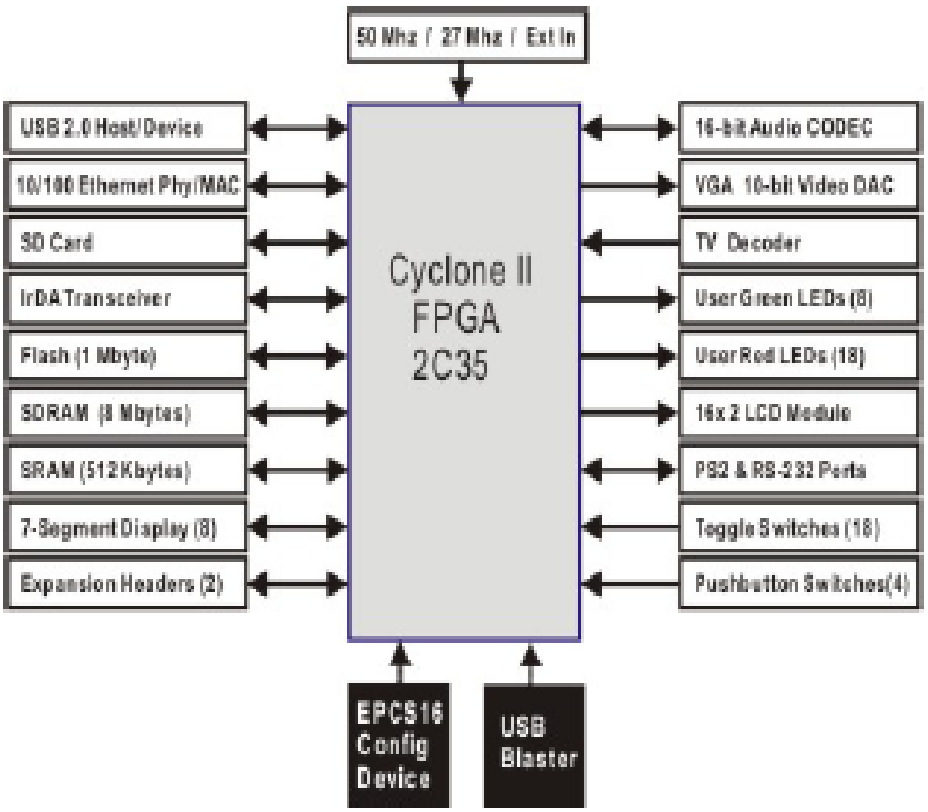

Figure 2.2. Block diagram of the DE2 board.

Following is more detailed information about the blocks in Figure 2.2:

Cyclone II 2C35 FPGA

- 33,216 LEs
- 105 M4K RAM blocks
- 483,840 total RAM bits
- 35 embedded multipliers
- 4 PLLs
- 475 user I/O pins
- FineLine BGA 672-pin package

Serial Configuration device and USB Blaster circuit

- Altera's EPCS16 Serial Configuration device
- On-board USB Blaster for programming and user API control
- JTAG and AS programming modes are supported

SRAM

- 512-Kbyte Static RAM memory chip
- Organized as 256K x 16 bits
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

SDRAM

- 8-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
- Organized as 1M x 16 bits x 4 banks
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

Flash memory

- 4-Mbyte NOR Flash memory (1 Mbyte on some boards)
- 8-bit data bus
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

SD card socket

- Provides SPI mode for SD Card access
- Accessible as memory for the Nios II processor with the DE2 SD Card Driver

Pushbutton switches

- 4 pushbutton switches
- Debounced by a Schmitt trigger circuit
- Normally high; generates one active-low pulse when the switch is pressed

Toggle switches

- 18 toggle switches for user inputs
- A switch causes logic 0 when in the DOWN (closest to the edge of the DE2 board) position and logic 1 when in the UP position

Clock inputs

- 50-MHz oscillator
- 27-MHz oscillator
- SMA external clock input

Audio CODEC

- Wolfson WM8731 24-bit sigma-delta audio CODEC
- Line-level input, line-level output, and microphone input jacks
- Sampling frequency: 8 to 96 KHz
- Applications for MP3 players and recorders, PDAs, smart phones, voice recorders, etc.

VGA output

- Uses the ADV7123 240-MHz triple 10-bit high-speed video DAC
- With 15-pin high-density D-sub connector
- Supports up to 1600 x 1200 at 100-Hz refresh rate
- Can be used with the Cyclone II FPGA to implement a high-performance TV Encoder

NTSC/PAL TV decoder circuit

- Uses ADV7181B Multi-format SDTV Video Decoder
- Supports NTSC-(M,J,4.43), PAL-(B/D/G/H/I/M/N), SECAM
- Integrates three 54-MHz 9-bit ADCs
- Clocked from a single 27-MHz oscillator input
- Supports Composite Video (CVBS) RCA jack input.
- Supports digital output formats (8-bit/16-bit): ITU-R BT.656 YCrCb 4:2:2 output + HS, VS, and FIELD
- Applications: DVD recorders, LCD TV, Set-top boxes, Digital TV, Portable video devices

10/100 Ethernet controller

- Integrated MAC and PHY with a general processor interface
- Supports 100Base-T and 10Base-T applications
- Supports full-duplex operation at 10 Mb/s and 100 Mb/s, with auto-MDIX
- Fully compliant with the IEEE 802.3u Specification
- Supports IP/TCP/UDP checksum generation and checking
- Supports back-pressure mode for half-duplex mode flow control

USB Host/Slave controller

- Complies fully with Universal Serial Bus Specification Rev. 2.0
- Supports data transfer at full-speed and low-speed
- Supports both USB host and device
- Two USB ports (one type A for a host and one type B for a device)

- Provides a high-speed parallel interface to most available processors; supports Nios II with a Terasic driver

- Supports Programmed I/O (PIO) and Direct Memory Access (DMA)

Serial ports

- One RS-232 port
- One PS/2 port
- DB-9 serial connector for the RS-232 port
- PS/2 connector for connecting a PS2 mouse or keyboard to the DE2 board

IrDA transceiver

- Contains a 115.2-kb/s infrared transceiver
- 32 mA LED drive current
- Integrated EMI shield
- IEC825-1 Class 1 eye safe
- Edge detection input

Two 40-pin expansion headers

- 72 Cyclone II I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
- 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives
- Diode and resistor protection is provided

2.3 Power-up the DE2 Board

The DE2 board comes with a preloaded configuration bit stream to demonstrate some features of the board. This bit stream also allows users to see quickly if the board is working properly. To power-up the board perform the following steps:

1. Connect the provided USB cable from the host computer to the USB Blaster connector on the DE2 board. For communication between the host and the DE2 board, it is necessary to install the Altera USB Blaster driver software. If this driver is not already installed on the host computer, it can be installed as explained in the tutorial *Getting Started with Altera's DE2 Board*. This tutorial is available on the DE2 System CD-ROM and from the Altera DE2 web pages.
2. Connect the 9V adapter to the DE2 board
3. Connect a VGA monitor to the VGA port on the DE2 board
4. Connect your headset to the Line-out audio port on the DE2 board
5. Turn the RUN/PROG switch on the left edge of the DE2 board to RUN position; the PROG position is used only for the AS Mode programming

6. Turn the power on by pressing the ON/OFF switch on the DE2 board

At this point you should observe the following:

- All user LEDs are flashing
- All 7-segment displays are cycling through the numbers 0 to F
- The LCD display shows **Welcome to the Altera DE2 Board**
- The VGA monitor displays the image shown in Figure 2.3.
- Set the toggle switch SW17 to the DOWN position; you should hear a 1-kHz sound
- Set the toggle switch SW17 to the UP position and connect the output of an audio player to the Line-in connector on the DE2 board; on your headset you should hear the music played from the audio player (MP3, PC, iPod, or the like)
- You can also connect a microphone to the Microphone-in connector on the DE2 board; your voice will be mixed with the music played from the audio player

Figure 2.3. The default VGA output pattern.

REFERENCIAS BIBLIOGRAFICAS

[1] Altera, DE2 Development and Education Board Altera, http://www.altera.com/education/univ/materials/boards/de2/unv-de2-board.html?GSA_pos=1&WT.oss_r=1&WT.oss=de2, Mayo 2011 2011.

[2] Altera, DE2 Development and Education Board Altera, ftp://ftp.altera.com/up/pub/Webdocs/DE2_introduction.pdf, Mayo 2011.

[3] Nios II C2H Compiler User Guide, www.altera.com/.../ug_**nios2_c2h_compiler**.pdf, Abril 2011.

[4] Vargas Javier, Mohammed Gharbi y Lucas Alejandro, Un nuevo algoritmo de detección de bordes en imágenes con ruido Gaussiano, https://opera-portal.us.es/pid/public/default/grupo/ver/id/30, Junio 2011.

[5] Universidad de Sevilla, Algoritmos genéticos: Detección de bordes de imágenes, www.sav.us.es/formaciononline/textosz/gen**bordes**03/presenta.ppt, Junio 2011.

[6] Asociación Española de Teledetección, Detección de bordes en imágenes SST, www.aet.org.es/congresos/ix/Lleida126.pdf, Junio 2011

[7] Bravo Ignacio, Arquitectura basada en FPGA`s para la detección de objetos en movimiento, utilizando visión computacional y técnicas PCA,

http://dspace.uah.es/dspace/bitstream/handle/10017/1381/tesis_ignacio_bravo.pdf?sequence=1, Julio 2011.

[8] Mosquera Oscar Leonardo, Convolución con máscaras para detectar bordes, http://es.scribd.com/doc/37184885/Detectar-Bordes-de-una-Imagen, Julio 2011

[9] Valverde Rebaza Jorge, Detector de Bordes de Sobel en Matlab, http://jc-info.blogspot.com/2011/06/detector-bordes-sobel-codigo-matlab.html, Julio 2011.

[10] Interactive and Cooperative Technologies Lab, Formatos de Imagen, http://ict.udlap.mx/people/raulms/avances/formatos.html, Agosto 2011

[11] Aula Clic SL, Formatos de Imagen, http://www.aulaclic.es/html/a_5_1_1.htm, Agosto 2011

[12] Wikipedia, Eclipse (software) http://es.wikipedia.org/wiki/Eclipse_%28software%29, Agosto 2011

[13] Altera, Nios II processor performance benchmarks data sheet, http://www.altera.com/literature/ds/ds_nios2_perf.pdf, Agosto 2011

[14] Ramblingbytes World, Eclipse Ganymade, http://ramblingbyte.wordpress.com/2008/07/09/pdt-on-ganymede/ganymede_splash/, Diciembre 2011