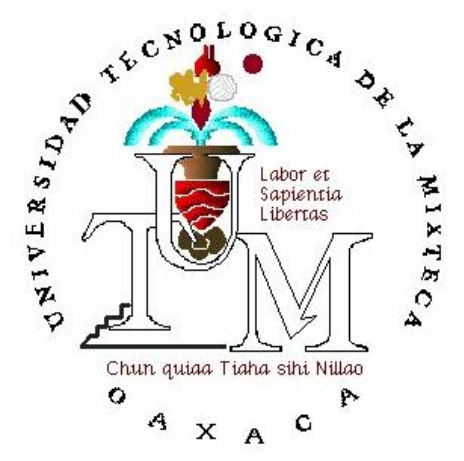


UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"SISTEMA DE VISUALIZACIÓN DE IMÁGENES A 8 COLORES EMPLEANDO LA TARJETA DE DESARROLLO DIGILENT SPARTAN-3"

TESIS

PARA OBTENER EL TÍTULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA
ROLANDO RUÍZ CARBAJAL

DIRECTOR DE TESIS
M.C. FELIPE SANTIAGO ESPINOSA

HUAJUAPAN DE LEÓN, OAXACA. OCTUBRE 2012

Tesis presentada el 26 de Octubre de 2012 ante los siguientes sinodales:

Dr. Enrique Guzmán Ramírez

Dr. Mikhail Arkhipov Alexandrovitch

Ing. Hugo Suárez Onofre

Director de Tesis:

M.C. Felipe Santiago Espinosa.

Dedicatoria

A mi madre, con cariño, respeto y amor.

Agradecimientos

Agradezco en particular al Prof. Felipe Santiago Espinosa, quien más allá de dirigir este proyecto con infinita paciencia, me motivo con sus conocimientos y experiencia; Pero sobre todo, agradezco su valiosa amistad.

Por ser mi primera maestra, consejera y el pilar fundamental de mis principios, educación y formación: A mi madre, quien me enseño que la libertad y la conquista más grande es hacer lo que amas.

A mi padre.

A Marvelia, por brindarme su cariño, confianza y apoyo incondicional.

A mis hermanos y amigos, compañeros de camino, siempre presentes; por su apoyo, entusiasmo, ganas y aliento.

Rolando.

Índice

Dedicatoria v
Agradecimientos vii
Índice ix
Lista de figuras xiii
Lista de Tablas xv
Introducción 1
Objetivos de la Tesis 1
Justificación 3
Estructura del documento de tesis 4
1. Marco Teórico 5
1.1 Dispositivos Lógicos Programables 5
1.1.1 Memoria Programable sólo de lectura (PROM) 5
1.1.2 Arreglo Lógico Programable (PLA) 6
1.1.3 Lógica de Arreglos Programables (PAL) 7
1.1.4 Arreglo Lógico Genérico (GAL) 7
1.1.5 Dispositivo Lógico Programable Complejo (CPLD) 8
1.1.6 Arreglo de Compuertas Programables en Campo (FPGA) 9
1.1.7 Tecnologías de fabricación y programación 10
1.1.7.1 Tecnología de FUSIBLES 10
1.1.7.2 Tecnología ANTIFUSIBLE 10
1.1.7.3 Tecnología EPROM 10
1.1.7.4 Tecnología EEPROM 11
1.1.7.5 Tecnología FLASH 11
1.1.7.6 Tecnología SRAM 11
1.1.8 Fabricantes y Manufatura 11
1.2 Entorno de desarrollo de la lógica programable 13
1.2.1 VHDL 13
1.2.1.1 Fundamentos del lenguaje 14
1.2.2 Herramientas y secuencia de diseño 15
1.3 Características de la tarjeta de desarrollo Digilab Spartan-3 18
1.3.1 Caracteristícas de los FPGA Spartan 3 de Xilinx 19
1.3.1.1 Bloques de Lógica Configurable (CLBs) 20
1.3.1.2 Bloques de Entrada/Salida (IOBs) 21

1.3.1.3 Bloques de memoria RAM (BlockRAM) 22
1.3.1.4 Bloques Multiplicadores Dedicados 23
1.3.1.5 Administrador de Reloj Digital 23
1.3.2 Programación de la Tarjeta Digilent Spartan-3 23
1.3.3 Comunicación Serie 24
1.3.3.1 Puerto Serie RS-232 24
1.3.3.2 Protocolo Interfaz PS/2 26
1.3.3.3 Ratón PS/2 29
1.3.4 Puerto VGA 31
1.3.5 Monitor 32
1.3.5.1 Monitor CRT 33
1.4 Estado del Arte 36
2. Diseño e Implementación del Sistema 39
2.1 Metodología de Diseño 39
2.2 Organización del Sistema 39
2.1 Módulo de Interfaz de Video 41
2.1.1 Componente Divisor de Frecuencia 41
2.1.1.1 Descripción Funcional 41
2.1.1.2 Descripción del Hardware, Diseño e Implementación 42
2.1.2 Componente de Memoria de Video SRAM. 42
2.1.2.1 Descripción Funcional 42
2.1.2.2 Descripción del Hardware, Diseño e Implementación 43
2.1.3 Componente Interfaz Mouse PS/2 45
2.1.3.1 Descripción Funcional 45
2.1.3.2 Descripción funcional del Hardware, Diseño e Implementación 46
2.1.4 Componente Cursor 52
2.1.4.1 Descripción Funcional 52
2.1.4.2 Descripción del Hardware, Diseño e Implementación 53
2.1.5 Componente Controlador VGA 56
2.1.5.1 Descripción Funcional 56
2.1.5.2 Descripción del Hardware, Diseño e Implementación 57
2.1.6 Componente Control de Video 61
2.1.6.1 Descripción funcional 61
2.1.6.2 Descripción de hardware, diseño e Implementación 62
2.1.7 Integración del Módulo Interfaz de Video 66
2.2 Módulo interfaz Serie 69

2.2.1 Componente UART ... 69
2.2.1.1 Descripción funcional ... 70
2.2.1.2 Descripción del Hardware e Implementación ... 71
2.2.2 Componente Control Serie ... 71
2.2.2.1 Descripción funcional ... 72
2.2.2.2 Descripción del Hardware, Diseño e Implementación ... 73
2.2.2 Integración del Modulo Interfaz Serie ... 74
2.3 Módulo para el Almacenamiento de Imágenes ... 76
2.3.1 Componente NVRAM ... 76
2.3.1.1 Descripción Funcional ... 77
2.3.1.2 Diseño e Implementación del Hardware ... 78
2.3.1.3 Descripción funcional del Hardware, Diseño e Implementación ... 80
2.3.2 Componente Control de Memorias ... 81
2.3.2.1 Descripción funcional ... 81
2.3.2.2 Descripción del Hardware, Diseño e Implementación ... 82
2.4 Integración del Sistema Visualización de Imágenes ... 83
3. Software ... 87
3.1 Ciclo de vida del Software ... 87
3.2 Análisis de Requisitos y Diseño ... 87
3.2.1 Panorama General ... 87
3.2.2 Metas ... 88
3.2.3 Requisitos Funcionales ... 88
3.2.4 Descripción del Sistema Software ... 89
3.2.4.1 Puerto Serie ... 89
3.2.4.2 Manejo de Archivos de Imagen ... 91
3.2.4.3 Simplificación de Imágenes ... 91
3.2.4.4 Transmisión de Datos ... 93
3.2.4.5 Manejo de la interfaz del Software ... 94
4. Resultados y Conclusiones ... 97
4.1 Resultados ... 97
4.2 Conclusiones ... 100
4.3 Trabajos futuros ... 100
Bibliografía ... 101
Sitios de Internet ... 103
Anexo A. Comandos Ratón PS/2 ... 105
Anexo B. Diagrama PCB Tarjeta Externa de Memoria NVRAM ... 108

Lista de figuras

Figura I.1. Diagrama funcional del sistema de proyección de imágenes basado en FPGA. 2

Figura 1.1. Arreglo AND fijo y OR Programable de una PROM ..……...…….. 6
Figura 1.2. Arreglo AND y OR programable de un PLA. 6
Figura 1.3. Arreglo AND programable y OR fijo con lógica de salida de un PAL. 7
Figura 1.4. Arreglo AND reprogramable y OR fijo con Macrocelda lógica de salida de un GAL. 8
Figura 1.5. Diagrama a bloques de la arquitectura de un CPLD [4]. 8
Figura 1.6. Elementos básicos en la arquitectura de un FPGA. 9
Figura 1.7. Estructura de diseño de una descripción en VHDL. 14
Figura 1.8. Flujo de diseño en lógica programable. 16
Figura 1.9. Componentes de la Tarjeta Digilent Spartan-3. 18
Figura 1.10. Arquitectura del FPGA Spartan-3. 19
Figura 1.11. Arreglo de Slices. 20
Figura 1.12. Estructura simplificada de un IOB [URL8]. 21
Figura 1.13. Paquete de comunicación serial empleando protocolo RS232. 26
Figura 1.14. Conector mini-DIN y funciones de sus terminales. 26
Figura 1.15. Trama serie de recepción de información del dispositivo PS/2. 28
Figura 1.16. Trama serie de envío de comandos al dispositivo PS/2. 28
Figura 1.17. Sistema de coordenadas relativas de un dispositivo señalador. 29
Figura 1.18. Información generada por un Ratón estándar PS/2. 29
Figura 1.19. Conector de puerto VGA. 32
Figura 1.20. Monitor de tubo de rayos catódicos [URL14]. 33
Figura 1.21. Recorrido del haz del electrones y demarcación de la pantalla. 34
Figura 1.22. Señales de sincronización Horizontal y R, G y B. 34
Figura 1.23. Señales de sincronización Vertical y R, G y B. 35

Figura 2.1. Diagrama a bloques General del Sistema. ……...……40
Figura 2.2. Diagrama a bloques de las descripciones de hardware. 40
Figura 2.3. Símbolo del componente Divisor de Frecuencia. 41
Figura 2.4. Diagrama a bloques de la descripción funcional del componente Divisor de Frecuencia. 42
Figura 2.5. Símbolo del módulo de memoria de video SRAM. 43
Figura 2.6. Diagrama a bloques de la descripción de hardware componente memoria de video SRAM. ...44
Figura 2.7. Máquina de estados de accesos para Lectura y Escritura de la memoria de video SRAM. 44
Figura 2.8. Símbolo del módulo de la interfaz Mouse PS/2. 45
Figura 2.9. Diagrama a bloques de la descripción funcional del módulo de la interfaz del Mouse PS/2. ...46
Figura 2.10. Máquina de estado del control de la interfaz mouse PS/2 47
Figura 2.11. Diagrama de flujo del proceso de transmisión de un comando al ratón. 49
Figura 2.12. Diagrama de flujo del proceso de recepción de 3 bytes de información del ratón. 50
Figura 2.13. Representación en una pantalla de los movimientos del ratón. 51
Figura 2.14. Símbolo del componente Cursor. 53
Figura 2.15. Diagrama a bloques de la descripción de hardware módulo Cursor. 54
Figura 2.16. Cursores del ratón y posición de selección. 54
Figura 2.17. Símbolo con la descripción del hardware del componente Controlador VGA. 56
Figura 2.18. Diagrama de tiempos de las señales de sincronismo horizontal y R, G y B. 57
Figura 2.19. Diagrama de tiempos de las señales de sincronismo vertical y R, G y B. 58
Figura 2.20. Diagrama a bloques de la descripción funcional del módulo Controlador VGA. 59
Figura 2.21. Símbolo con la descripción de hardware del componente Control de Video. 61
Figura 2.22. Representación de la información en pantalla. 63

Figura 2.23. Segmentación de la memoria NVRAM en páginas. 63
Figura 2.24. Esquemático de la conexión de los componentes en el Módulo Interfaz de Video. 67

Figura 2.25. Símbolo Descripción de hardware del módulo Interfaz de Video. 69
Figura 2.26. Símbolo con la descripción del hardware del Componente UART. 70
Figura 2.27. Diagrama a bloques de la descripción funcional del componente UART. 71
Figura 2.28. Símbolo Descripción del hardware del componente Control Serie. 72
Figura 2.29. Proceso de recepción de datos seriales y almacenamiento paginado en NVRAM. 73
Figura 2.30. Estructura del Byte que indica un comando de Operación. 73
Figura 2.31. Esquemático de la conexión de los componentes en el Módulo Interfaz Serie. 75
Figura 2.32. Símbolo Descripción de hardware del módulo Interfaz Serie. 75
Figura 2.33. Símbolo con la descripción del hardware del componente NVRAM. 77
Figura 2.34. Conexiones de FPGA y NVRAM. 78
Figura 2.35. Circuito Esquemático de la tarjeta externa de Memorias NVRAM. 79
Figura 2.36. Diagrama a bloques de la descripción de hardware componente NVRAM. 80
Figura 2.37. Máquina de estados del componente NVRAM. 80
Figura 2.38. Símbolo Descripción de hardware del componente Control de Memorias. 81
Figura 2.39. Esquemático de la conexión de los módulos interfaz de video, interfaz PS2 e interfaz de Memoria. 84

Figura 3.1. Representación numérica del espectro de Color Verde……92
Figura 3.2. Matriz bidimensional de manejo de los componentes de colores primarios de manera atómica. 93
Figura 3.3. Matriz bidimensional de manejo de los componentes de colores primarios de manera secuencial. 93
Figura 3.4. Byte de Comando y Número de página. 93
Figura 3.5. Opciones del Menú de la aplicación. 94
Figura 3.6. Barra de Progreso de transmisión. 94
Figura 3.7. Tira de imágenes con el arreglo de elementos imagen. 95
Figura 3.8. Visor de imagen. 95
Figura 3.9. Elementos de la interfaz visual de la aplicación. 96
Figura 3.10.Visualización de una imagen en formato original en 256 colores y simplificada a 8 colores. .96

Figura 4.1. Sistema de Visualización de Imágenes en funcionamiento con periféricos conectados……98
Figura 4.2. Sistema de Visualización de Imágenes desplegando una imagen. 99

Lista de Tablas

Tabla 1.1. Dispositivos Lógicos Programables. 5
Tabla 1.2. Tecnologías de fabricación de PLD. 10
Tabla 1.3. Fabricantes de Dispositivos Lógicos Programables. 11
Tabla 1.4. Herramientas y compañías de desarrollo EDA 15
Tabla 1.5. Características del FPGA Spartan-3 XC3S200FT256. 19
Tabla 1.6. Descripción de las señales del puerto JTAG. 23
Tabla 1.7. Resumen de las especificaciones básicas del Estándar RS-232-C. 25
Tabla 1.8. Estados del bus PS/2. 27
Tabla 1.9. 8 colores básicos RGB. 31

Tabla 2.1. Descripción de Puertos módulo Divisor de Frecuencia. 41
Tabla 2.2. Descripción del proceso divisor de frecuencia. 42
Tabla 2.3. Descripción de Puertos módulo memoria de video SRAM. 43
Tabla 2.4. Sentencias de control de individuales de la memoria SRAM. 44
Tabla 2.5. Descripción de Puertos módulo interfaz Mouse PS/2. 45
Tabla 2.6. Control de las Señales triestado del ratón. 46
Tabla 2.7. Filtro de la línea de reloj del ratón. 46
Tabla 2.8. Descripción del contador de inhibición de transmisión. 48
Tabla 2.9. Sentencia de un circuito de generación de paridad XOR. 48
Tabla 2.10. Sentencias aritméticas de la posición del cursor del ratón. 51
Tabla 2.11. Comparación del margen inferior del eje Y. 51
Tabla 2.12. Comparación del margen superior del eje Y. 52
Tabla 2.13. Comparación del margen inferior del eje X. 52
Tabla 2.14. Comparación del margen superior del eje X. 52
Tabla 2.15. Descripción de Puertos módulo Cursor. 53
Tabla 2.16. Selector de Cursores. 55
Tabla 2.17. Lógica combinacional para el cursor “000”. 55
Tabla 2.18. Sentencias de selección de salida de pixel. 56
Tabla 2.19. Descripción de puertos módulo controlador VGA. 57
Tabla 2.20. Tiempos de operación y ciclos de reloj de las señales de sincronismo. 58
Tabla 2.21. Descripción VHDL del contador horizontal y vertical. 60
Tabla 2.22. Descripción de las sentencias generadores de sincronismo y blanqueo. 60
Tabla 2.23. Sentencia de asignación del pixel a desplegar. 60
Tabla 2.24. Contadores Horizontales y Verticales de posición en pantalla activa. 60
Tabla 2.25. Descripción de Pines del componente Control de Video. 62
Tabla 2.26. Descripción de la secuencia de copiado de la imagen activa en la memoria de video. 64
Tabla 2.27. Descripción de la asignación de localidad a leer en la memoria de video. 65
Tabla 2.28. Proceso control de lectura en memoria de video y despliegue de pixeles. 65
Tabla 2.29. Descripción de la acción de los botones de ratón. 66
Tabla 2.30. Descripción de Pines del módulo Interfaz de Video. 68
Tabla 2.31. Descripción de Pines componente UART. 70
Tabla 2.32. Parámetro de Configuración para la velocidad de transmisión 71

Tabla 2.33. Descripción de Pines del componente Control Serie. 72
Tabla 2.34. Descripción de Proceso de Control de Escritura a la NVRAM..... 74
Tabla 2.35. Descripción de Pines del módulo Interfaz Serie..... 76
Tabla 2.36. Descripción de puertos módulo NVRAM. 77
Tabla 2.37. Conexiones de módulo externo de Memorias al FPGA Spartan-3..... 78
Tabla 2.38. Descripción de Pines del componente Control de Memorias..... 82
Tabla 2.39. Sentencias generadoras de las señales de control de memoria ocupada. 83
Tabla 2.40. Descripción de la selección de señales de control de la memoria. 83
Tabla 2.41. Descripción de Pines del Sistema de Visualización de Imágenes. 85

Tabla 3.1. Código de manejo del puerto serie. 90
Tabla 3.2. Extracto del Código de la función abrir imagen. 91
Tabla 3.3. Extracto del Código de la función Guardar imagen. 91
Tabla 3.4. Definición de la estructura tipo Tcolor. 92
Tabla 3.5. Dibujo de un pixel sobre el Canvas. 92
Tabla 3.6. Discriminador de Umbral de Color. 92
Tabla 3.7. Código para la transmisión por el puerto serie de una página. 94

Tabla 4.1. Tabla de los recursos empleados por el sistema de visualización de imágenes en un FPGA 3S200FT256. 97

Introducción

El auge de las tecnologías de la información se ha debido al extraordinario aumento de la capacidad de integración de los fabricantes de circuitos integrados digitales, que han pasado de contener 10 puertas lógicas a más de 1.000.000 en los últimos treinta y cinco años del siglo XX [1]. Actualmente, al momento de implementar un sistema electrónico digital, el diseñador dispone de un conjunto amplio de tecnologías. Una de las más populares son los dispositivos de lógica programable (PAL, CPLDs, FPGAs, etc).

Los dispositivos de lógica programable más versátiles son los FPGA (*Field Programmable Gate Array*), ya que permiten realizar diseños a medida, con bajo costo de desarrollo, incluso para la producción de pocas unidades. Sus bloques lógicos y sus recursos de interconexión son configurables, características por las que son ampliamente empleados al realizar prototipos express. Así mismo, la existencia de lenguajes de descripción de hardware (HDL, Hardware Description Languages) tales como ABEL, Verilog y VHDL hacen que los diseños sean archivos de texto, que contienen el "código fuente" del circuito. Esto ha llevado a que el proceso de diseño de hardware se parezca cada vez más al del software; lo que conduce, a que el hardware se convierta en algo que pueda ser compartido y reutilizado.

En el presente trabajo se plantea el desarrollo de un sistema basado en FPGA, que permita la visualización, mediante el uso de un monitor VGA de resolución de 640 x 480 pixeles, de un conjunto de imágenes simplificadas a 8 colores, almacenadas en un banco de memorias no volátiles; estableciendo una interacción con el usuario al permitirle el avance y retroceso entre un conjunto de imágenes, mediante el uso de un ratón de computadora. Así como la integración con el software de computadora que permita la simplificación de imágenes a 8 colores y el envío al FPGA, por medio del puerto serie RS-232, para su almacenamiento. Esto con la finalidad de desarrollar una herramienta alterna, educativa y de bajo costo para la realización de presentaciones.

Objetivos de la Tesis

El objetivo general que se persigue en este trabajo de tesis consiste en el desarrollo de un sistema que permita la visualización en un monitor o dispositivo estándar VGA de un conjunto imágenes, simplificadas a 8 colores, almacenadas en un banco de memorias NVRAM.

El sistema deberá contar con dos módulos principales:

- El primero, un módulo para la proyección de imágenes, que realizará la interfaz con el banco de memorias NVRAM y dispondrá de comunicación con un monitor o dispositivo VGA, permitiendo visualizar las imágenes almacenadas en el banco de memorias NVRAM; estableciendo la interacción del usuario mediante la interface con un ratón de computadora PS/2, por medio del que se realizarán las acciones de señalización e instrucción de control, como avance y retroceso de las imágenes.

- El segundo consiste en un Software de Computadora, que permita la selección de imágenes en formato BMP, con resolución de 640 x 480 pixeles, simplificación a una paleta de 8 colores y comunicación, mediante el puerto serie RS-232, con la tarjeta de desarrollo Digilent Spartan-3 para su descarga en el banco de memorias NVRAM; permitiendo de esta manera el almacenamiento y transporte de la información.

El conjunto de componentes serán desarrollados en el lenguaje para síntesis VHDL y se empleará como base de desarrollo al FPGA XC3S200, que se encuentra como componente principal en la tarjeta de desarrollo Digilent Spartan-3. En la Figura I.1 se muestra el diagrama funcional del sistema de proyección que se plantea.

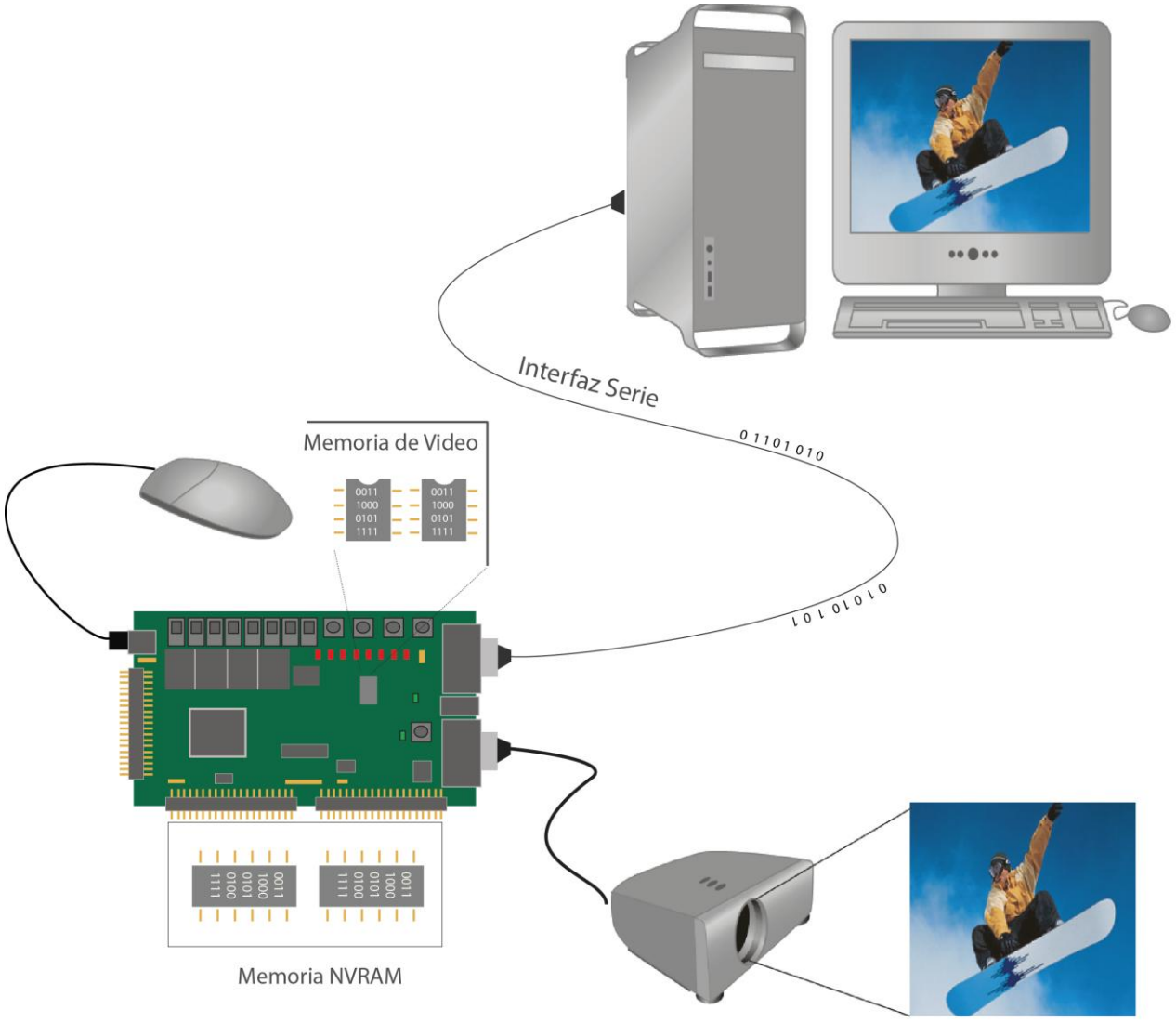


Figura I.1. Diagrama funcional del sistema de proyección de imágenes basado en FPGA.

Una vez planteado el objetivo principal, se establecen los siguientes objetivos específicos:

- Diseño y codificación de un software para PC que permita la selección, simplificación a una profundidad de color de 3 bits y comunicación para descarga de imágenes en formato BMP con resolución de 640 x 480 pixeles, mediante la interfaz serie RS-232, con el sistema de visualización de imágenes implementado en la tarjeta de desarrollo Digilent Spartan-3.

- Desarrollo de un componente que permita la interfaz de un FPGA con monitores que operen con el estándar de video VGA, bajo una resolución de 640 x 480 pixeles y una profundidad de color de 3 bits.

- Desarrollo de un componente que permita la comunicación de un FPGA con la memoria RAM incluida en la tarjeta de desarrollo Digilent Spartan-3, acondicionando su comportamiento al de una memoria de video.

- Desarrollo de un componente que permita la interfaz de un FPGA con un ratón de computadora mediante el estándar PS/2.

- Diseño y fabricación del circuito impreso de un banco de 2 memorias NVRAM DS1265W ó DS1270W como módulo de expansión para la tarjeta de desarrollo Digilent Spartan-3

- Desarrollo de un componente que permita la interfaz de un FPGA con el módulo de memorias NVRAM.

- Desarrollo de un componente de sincronización y control que permita la interacción de los diferentes componentes.

Justificación

El trabajo de investigación que se pretende llevar a cabo responde principalmente a una motivación personal para resolver el problema que específicamente se plantea, con la finalidad de obtener el conocimiento teórico y práctico que implica el proceso de desarrollo e investigación de un sistemas de visualización de imágenes sobre dispositivos lógicos programables, ya que esta capacidad se establece cada vez como una fuerte necesidad en las plataformas y sistemas electrónicos debido a que facilitan la operación, intuición de uso y proporcionan versatilidad y accesibilidad.

Así mismo, el diseño, implementación y resultados de este trabajo de tesis se pretende contribuir al desarrollo de proyectos de mayor envergadura, ya que permite la posibilidad de adaptar el sistema que se propone o módulos que lo conforman, a una amplia gama de aplicaciones, con lo que es posible ampliar, cambiar o modificar el paradigma que esta propuesta plantea. Con la ventaja de poder ser implementadas en cualquier FPGA con los recursos que las especificaciones del sistema demandan.

De la misma manera, el trabajo de tesis que se propone forma parte de una discreta línea de trabajos de investigación que se centran fundamentalmente en esta área del desarrollo tecnológico, para el fortalecimiento de los planes educativos y docentes que presenta la Universidad Tecnológica de la Mixteca (UTM), con la finalidad de describir el estado actual del conocimiento.

Aunque el trabajo en sí se plantea como un punto de partida para trabajos futuros, su aplicación es directa como una alternativa educativa de bajo costo para la visualización de imágenes simplificadas a 8 colores; con la evidente ventaja de ser un sistema de fácil manejo.

Estructura del documento de tesis

A continuación se detalla la estructura del documento de tesis:

El capítulo 1 presenta el marco teórico que incluye el estado del arte y entorno de desarrollo de la lógica programable, las características específicas de la tarjeta de desarrollo Digilent Spartan-3 y la información que representa el sustento teórico para el diseño de los componentes de los que se conforma el sistema de visualización de imágenes.

El capítulo 2 describe la metodología de desarrollo en sistemas digitales conocida como Top-Down, expone la organización del sistema y principalmente el diseño e implementación de los módulos que componen el sistema; describiendo finalmente, el proceso de integración de los diferentes módulos del sistema en una sola entidad, y el control central que permite la funcionalidad de conjunto.

El capítulo 3 describe el proceso de desarrollo del software mediante lenguaje C++, el proceso de simplificación y comunicación mediante el puerto RS-232.

El capítulo 4 presenta los resultados obtenidos, las conclusiones y trabajos futuros de investigación.

Finalmente, se presentan las referencias bibliográficas utilizadas en el desarrollo de este documento y los anexos.

1. Marco Teórico

En este capítulo se describen los fundamentos teóricos de los elementos requeridos para el desarrollo del presente trabajo.

1.1 Dispositivos Lógicos Programables

Los Dispositivos Lógicos Programables o PLD (*Programmable Logic Device*) proporcionan una solución al diseño de sistemas digitales mediante la integración de múltiples elementos lógicos en un circuito integrado. Permiten realizar cualquier sistema electrónico digital con base en una arquitectura formada por un arreglo de celdas lógicas configurables y un conjunto de líneas de interconexión, también programables. En la actualidad se dispone de muchos dispositivos de complejidad diversa que albergan esta estructura, tal como lo muestra la Tabla 1.1.

Tabla 1.1. Dispositivos Lógicos Programables.

Dispositivo	Descripción
PROM	Memoria Programable sólo de lectura (*Programmable Read-Only Memory*)
PLA	Arreglo Lógico Programable (*Programmable Logic Array*)
PAL	Lógica de Arreglos Programables (*Programmable Array Logic*)
GAL	Arreglo Lógico Genérico (*Generic Logic Array*)
CPLD	Dispositivo Lógico Programable Complejo (*Complex PLD*)
FPGA	Arreglo de Compuertas Programables en Campo (*Field Program Gate Array*)

1.1.1 Memoria Programable sólo de lectura (PROM)

Este dispositivo lógico se constituye por una arquitectura con un conjunto fijo de compuertas AND conectadas como decodificador y un arreglo OR (Figura 1.1); en donde el valor de cada bit depende del estado de un fusible, que puede ser quemado una sola vez[2, 3].

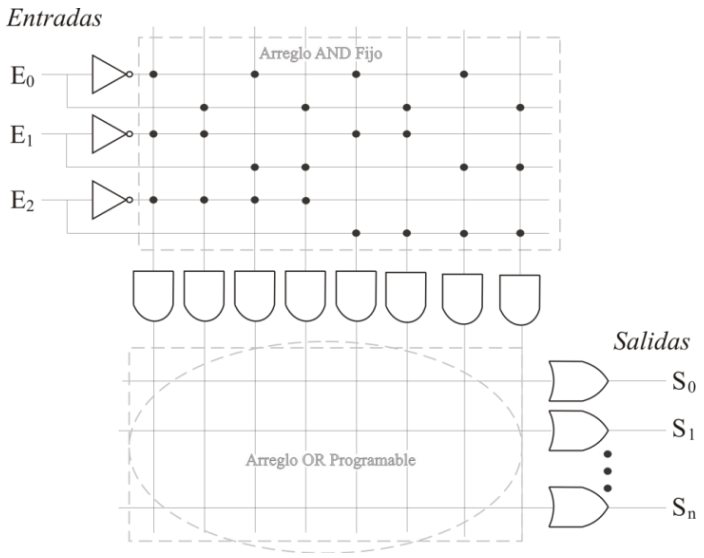


Figura 1.1. Arreglo AND fijo y OR Programable de una PROM.

1.1.2 Arreglo Lógico Programable (PLA)

Este dispositivo lógico se encuentra formado por arreglos AND y OR programables (Figura 1.2). Fue desarrollado para superar algunas de las limitaciones de la arquitectura de las memorias PROM, en una PROM se incluyen todos los términos producto debidos a las entradas, mientras que un PLA es más flexible, sólo pueden programarse aquellos que sean necesarios, generalmente son programados por máscara en el proceso de manufactura. La particularidad de estos dispositivos es que se encuentran embebidos en circuitos integrados más complejos, como microprocesadores [4]. Los PLA que pueden ser programados después del proceso de manufactura reciben el nombre de FPLA (*Field-programmable PLA*, PLA programable en campo).

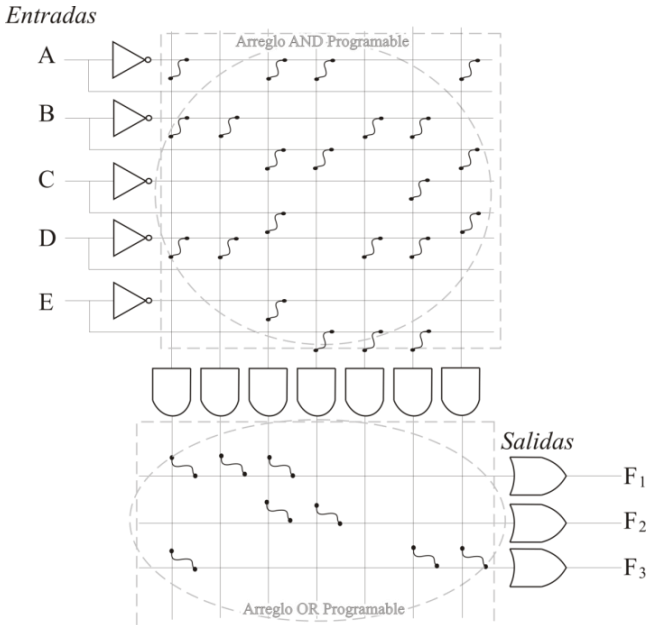


Figura 1.2. Arreglo AND y OR programable de un PLA.

1.1.3 Lógica de Arreglos Programables (PAL)

La PAL se desarrolló para superar ciertas desventajas del PLA, tales como los largos retardos debidos a los fusibles adicionales que resultan de la utilización de dos arreglos programables y la mayor complejidad del circuito. La PAL básica está formada por un arreglo AND programable y un arreglo OR fijo con la lógica de salida (Figura 1.3). Esta estructura permite implementar cualquier suma de productos con un número de variables definido. Su arquitectura se implementa con tecnología bipolar (TTL o ECL), lo que permite optimizar la velocidad del dispositivo.

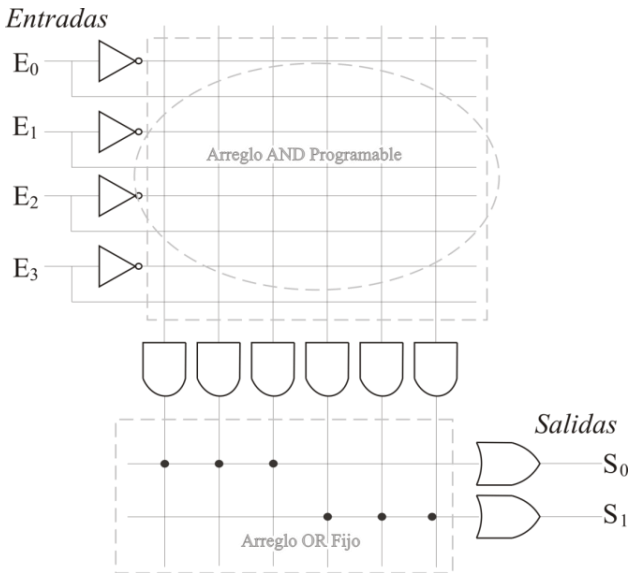


Figura 1.3. Arreglo AND programable y OR fijo con lógica de salida de un PAL.

1.1.4 Arreglo Lógico Genérico (GAL)

Este arreglo tiene las mismas propiedades lógicas que el PAL, ya que se forma por un arreglo AND reprogramable y un arreglo OR fijo, pero con la diferencia que puede ser borrado y reprogramado; ya que, en vez de fusibles como los dispositivos PAL, fundamenta su arquitectura en tecnología E^2CMOS (CMOS Borrable Eléctricamente , *Electrically Erasable CMOS*).

La innovación que representa este dispositivo, con respecto a la PAL, se encuentra en las Macroceldas lógicas de salida (OLMC, *Output Logic Macrocell*), formadas por circuitos lógicos que permiten programar las salidas como lógica combinacional o sencuencial [5], ya que cada Macrocelda incluye un Flip-Flop y un multiplexor para seleccionar el tipo de salida, como se muestra en la Figura 1.4.

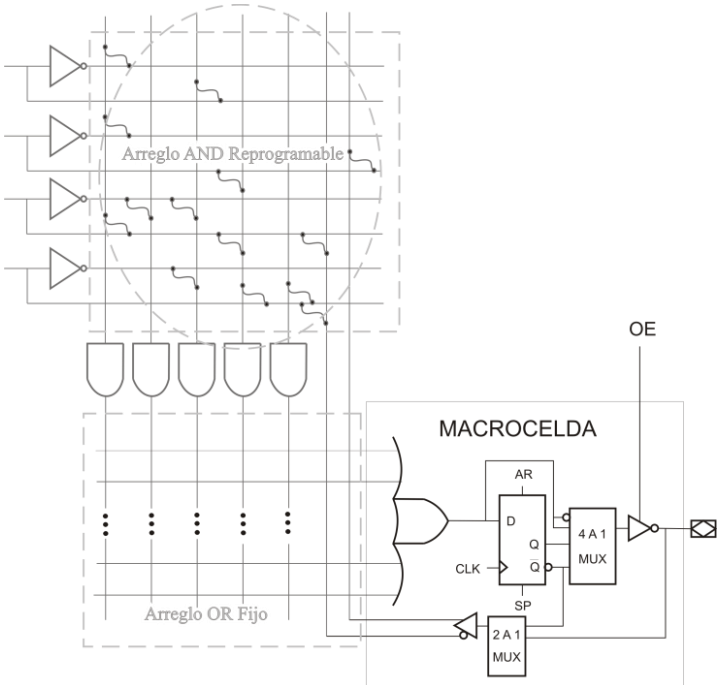


Figura 1.4. Arreglo AND reprogramable y OR fijo con Macrocelda lógica de salida de un GAL.

1.1.5 Dispositivo Lógico Programable Complejo (CPLD)

Los CPLD extienden el concepto de un PLD a un nivel mayor de integración –de ahí lo "complejo"-, ya que contienen el equivalente a varias PAL enlazadas por interconexiones programables (Figura 1.5), reemplazando miles o incluso cientos de miles de puertas lógicas en un mismo circuito integrado; la implementación de la tecnología E^2CMOS, les permite la capacidad de configurabilidad [6,7].

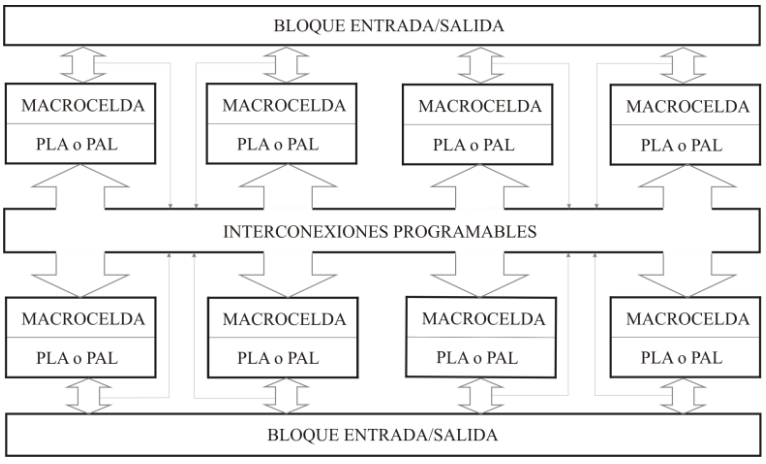


Figura 1.5. Diagrama a bloques de la arquitectura de un CPLD [4].

1.1.6 Arreglo de Compuertas Programables en Campo (FPGA)

Los FPGAs son los dispositivos que en la actualidad alcanzan la mayor densidad de integración y complejidad de los circuitos de lógica programable, su arquitectura se basa en una matriz programable que contienen múltiples niveles de lógica, cuenta con un sistema de interconexión flexible que no se encuentra limitado a la típica matriz AND-OR. Estos dispositivos se caracterizan por la alta densidad de compuertas, alto rendimiento y un número grande de entradas/salidas definibles por el usuario. En la Figura 1.6 se muestra la arquitectura básica de un FPGA, la cual se encuentra formado tres elementos básicos [2]:

- CLBs (Bloques de Lógica Configurable , *Configurable Logic Blocks*), proporcionan los elementos funcionales para construir la mayoría de la lógica así como el almacenamiento de datos. Su complejidad puede variar desde un par de transistores hasta un conjunto de memorias de acceso aleatorio denominadas tablas de consulta (LUT, *Look-Up-Tables*).

- IOBs (Bloques de Entrada/Salida, *I/O Blocks*), estos recursos lógicos proveen el control de flujo de datos entre lógica interna y las terminales de entrada/salida del dispositivo.

- Recursos de interconexión, las entradas y salidas de los CLB e IOB se interconectan mediante líneas e interruptores programables que se ubican en los canales de cableado entre las filas y columnas de los bloques.

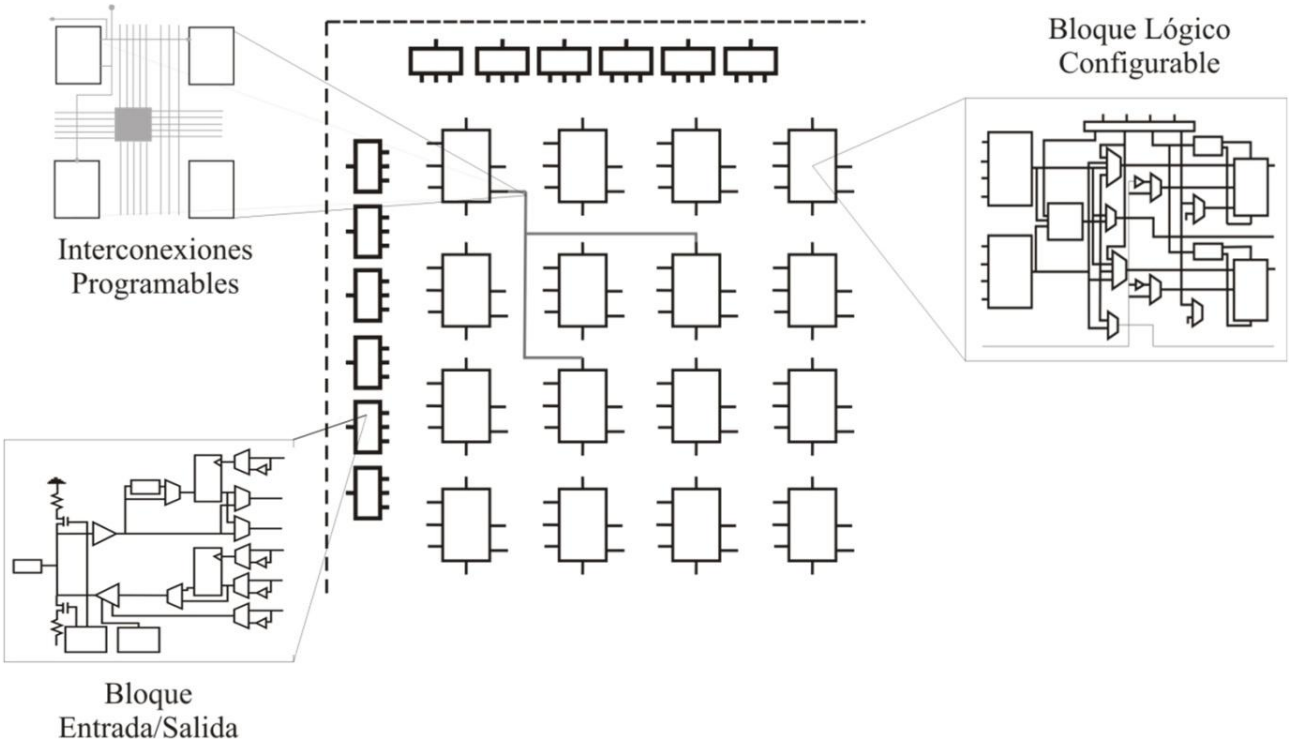


Figura 1.6. Elementos básicos en la arquitectura de un FPGA.

1.1.7 Tecnologías de fabricación y programación

La fabricación de dispositivos lógicos programables emplea diferentes tecnologías en la implementación de las celdas de memoria que permiten la configuración del PLD. La gran mayoría de estos procesos emplean tecnología unipolar CMOS (Tecnología Complementaria Metal Óxido Semiconductor, *Complementary Metal Oxide Semiconductor*) y en menor medida tecnología bipolar TTL (Lógica Transistor a Transistor , *Transistor-Transistor Logic*). Cada tecnología tiene sus ventajas y desventajas respecto a la fiabilidad, consumo o comportamiento de programación, tal como podemos apreciar en la Tabla 1.2.

Tabla 1.2. Tecnologías de fabricación de PLD.

Características	Fusibles	Antifusible	EPROM	EEPROM	FLASH	SRAM
Tiempo retención	Ilimitado	Ilimitado	10 - 20 años	10 - 20 años	10 - 20 años	Volátil
Ciclos de Programado	OTP	OTP	10,000	1,000 – 10,000	50 – 10,000	Ilimitado
Ciclos de Borrado	No Borrable	No Borrable	10,000	1,000-10,000	50-10,000	Ilimitado
Tiempo Programado	Minutos	Minutos	Minutos	Segundos	Segundos	mSeg
Tiempo Borrado	No Borrable	No Borrable	0.1 m Seg/celda	0.1m Seg/celda	0.1m Seg/celda	mSeg
Consumo Energético	Bajo	Bajo	Medio	Medio	Medio	Alto

1.1.7.1 Tecnología de FUSIBLES

Las celdas de FUSIBLES son celdas de memoria no volátil, programables una sola vez (OTP, *One-Time Programmable*). Esta es la tecnología original de programación de los dispositivos lógicos programables. Fabricadas bajo procesos de tecnología Bipolar (TTL), requieren de un programador especial para establecer la configuración del dispositivo. Los circuitos sintetizados en dispositivos lógicos con esta tecnología crean conexiones permanentes una vez programados, lo que genera implementaciones con retardos mínimos.

1.1.7.2 Tecnología ANTIFUSIBLE

Las celdas basadas en tecnología antifusibles son no volátiles, sólo son programables una vez (OTP) de manera permanente e irreversible y requieren de un programador para ser configuradas. Presentan las ventajas de la tecnología de fusibles, pero al estar fabricadas con un proceso CMOS la programación es más rápida.

1.1.7.3 Tecnología EPROM

Las celdas EPROM (Memoria Programable Borrable de Sólo Lectura, *Erasable Programmable Read-Only Memory*) son no volátiles, basadas en procesos CMOS, Los circuitos con esta tecnología emplean un programador especial para ser configurados y una vez programados solo pueden ser borrados mediante la exposición a una fuente de luz ultravioleta (UV).

1.1.7.4 Tecnología EEPROM

Las celdas EEPROM (ROM Programable y Borrable Eléctricamente, *Electrically Erasable Programmable Read-Only Memory*) son no volátiles, basadas en tecnología CMOS y físicamente más grandes que una celda EPROM, pero pueden ser reprogramadas y borradas eléctricamente; por lo que los circuitos fabricados con esta tecnología pueden ser programados en sistema, aunque requieren de lógica externa para este fin.

1.1.7.5 Tecnología FLASH

Las celdas FLASH son no volátiles, fabricadas en tecnología CMOS presentan los beneficios de borrado eléctrico de la EEPROM y sólo la mitad del espacio físico; ofrecen menores tiempo de retardo en los circuitos sintetizados. Algunos, no todos, los dispositivos Flash pueden ser programados en sistema. La diferencia fundamental entre una memoria FLASH y una EEPROM es que la primera se programa por bloques, mientras que en la segunda se pueden programar celdas individuales.

1.1.7.6 Tecnología SRAM

Las celdas de memoria estática de acceso aleatorio (SRAM, *Static Random Access Memory*) desarrolladas en tecnología CMOS, son volátiles, debido a que el contenido desaparece una vez que se pierde la alimentación y el dispositivo requiere ser configurado al ser alimentado nuevamente; ofrecen la ventaja que permite al dispositivo la programación en sistema y reconfiguración durante el funcionamiento. Esta tecnología se emplea por regla en los FPGAs, los dispositivos basados en esta tecnología pueden configurarse automáticamente una vez alimentados desde un dispositivo ROM externo o interno (*on-chip*).

1.1.8 Fabricantes y Manufatura

La versatilidad y el desarrollo que han alcanzado los dispositivos lógicos programables ha logrado que el mercado se amplíe significativamente, principalmente la oferta de FPGAs como dispositivos de propósito general, seguida muy de cerca por la oferta de dispositivos para entornos especializados. En la Tabla 1.3 se muestra un acercamiento de las principales compañías involucradas en la fabricación de PLDs, de las cuáles se puede encontrar información detallada en sus sitios web.

Tabla 1.3. Fabricantes de Dispositivos Lógicos Programables.

Compañía	URL
Achronix Semiconductor Corporation	http://www.achronix.com
Actel Corporation	http://www.actel.com
Altera Corporation	http://www.altera.com
Atmel Corporation	http://www.atmel.com
Cypress Semiconductor	http://www.cypress.com
Lattice Semiconductor Corporation	http://www.latticesemi.com
Quicklogic Corporation	http://www.quicklogic.com
Xilinx	http://www.xilinx.com

Achronix Semiconductor Corporation, fabricante de dispositivos FPGA, basados en tecnología en SRAM, con desempeño para aplicaciones de alta velocidad. Comercialmente la familia Speedster proporciona velocidades de operación a 1.5 GHz [URL1].

Actel Corporation, inicia como fabricante de dispositivos FPGA antifusibles para empleo en el sector aeroespacial. Actualmente oferta el FPGA proASIC basado en tecnología FLASH, que van de unas decenas de miles hasta tres millones de compuertas lógicas. También ofrece FPGAs que incluyen mezcladores de señales basados en tecnología FLASH [URL2].

Altera Corporation, el segundo fabricante líder de FPGA's basados en SRAM. En la familia Stratix se ubican los dispositivos de alto rendimiento que ofrecen alta densidad de compuertas. Para soluciones de bajo costo están los dispositivos Cyclone. Así mismo, desarrollan dispositivos denominados ASIC estructurados [URL3].

Atmel Corporation, fabricante de dispositivos con tecnología reconfigurable, SRAM, FLASH y EEPROM. Proveen ASICs de alta densidad de compuertas, microcontroladores AVR con FPGAs en el mismo encapsulado, denominados FPSLIC (*Field Programmable System Level Integrated Circuits*) y microcontroladores ARM con FPGA, denominados Atmel CAP (Procesador Avanzado Personalizable, *Customizable Advanced Processor*)[URL4].

Cypress Semiconductor, fabricante de CPLDs de propósito general con tecnología reconfigurable, con la familia de dispositivos Delta39K y Ultra37000, que integran de 32 hasta 3000 macroceldas [URL5].

Lattice Semiconductor, este fabricante líder ofrece una amplia gama de FPGAs basados en tecnología SRAM, FLASH e híbridos. Como innovación están los dispositivos digitales programables interconectados, denominados ispGDX y la gama de productos de propósito general se centra sobre la familia Lattice ECP [URL6].

QuickLogic, fabricante de dispositivos basados en antifusibles efocados a soluciones de bajo consumo de energía para dispositivos móviles, aplicaciones industriales, comunicaciones y militares. Actualmente oferta los dispositivos: PolarPro y ArcticLink II [URL7].

Xilinx, es posiblemente el más conocido fabricante de FPGA, con dispositivos basados en tecnología SRAM que varían de decenas de miles a varios millones de compuertas lógicas. Actualmente centran la comercialización en dos familias de productos: Virtex 4, dispositivos altamente complejos con alta densidad de compuertas y Spartan, orientados al bajo consumo de potencia y aplicaciones industriales [URL8].

1.2 Entorno de desarrollo de la lógica programable

El desarrollo de la tecnología y evolución en la lógica programable han hecho necesario un nuevo paradigma en las herramientas de diseño electrónico, que permitan al ingeniero abordar la descripción de sistemas electrónicos cada vez más complejos. Esta necesidad llevó al desarrollo de algunos lenguajes de programación especial, que consisten en definir las funciones lógicas que realizará un PLD mediante *la descripción del comportamiento del hardware*, conocidos como HDL (Lenguaje de Descripción de Hardware, *Hardware Description Language*)

Existe una amplia gama de HDLs, dentro de los más populares en la industria por su uso y estandarización se encuentran:

- VHDL
- Verilog-HDL
- ABEL

En un intento de reducir la complejidad y el tiempo de desarrollo en fases de prototipaje rápido, para validar un diseño en HDL, existen varias propuestas y niveles de abstracción del diseño. Entre otras, HANDEL-C y National Instruments LabVIEW FPGA, que proponen un acercamiento de programación gráfica de alto nivel.

1.2.1 VHDL

VHDL es el acrónimo que representa la combinación de VHSIC (*Very High Speed Integrated Circuit*, Circuitos Integrados de Muy Alta Velocidad) y HDL (*Hardware Despription Language*, Lenguaje de Descripción de Hardware), es decir, lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad; definido por el IEEE (Institute of Electrical and Electronics Engineers) en el estándar ANSI/IEEE 1076-1993, el lenguaje puede ser empleado para modelar, documentar, simular, verificar y sintetizar un sistema digital. Aunque puede ser usado de forma general para describir cualquier circuito se usa principalmente para programar ASIC, PLD, FPGAs y similares.

En el diseño de sistemas digitales, VHDL permite modelar y simular un sistema desde un alto nivel de abstracción hasta el nivel lógico más elemental con compuertas y biestables. Básicamente permite tres estilos de descripción [1, 8]

- Algorítmico o de comportamiento. Permite un alto nivel de abstracción, el diseñador sólo describe el comportamiento del sistema.

- RTL o flujo de datos. Proporciona cierto nivel de abstracción, es necesario que el diseñador describa las distintas señales que interactúan en un circuito y su comportamiento.

- Estructural o lógico. Es necesario describir las interconexiones entre los distintos componentes de un circuito.

Es posible, además, mezclar en un mismo diseño los distintos niveles de abstracción o estilos de descripción; la idea es definir la interfaz de un módulo de hardware mientras deja invisibles sus detalles internos. Las descripciones de los modelos creados pueden ser utilizadas en diferentes tecnologías (portabilidad y reutilización de código).

1.2.1.1 Fundamentos del lenguaje

Una descripción en VHDL, tal como se muestra en la Figura 1.7, comprende en su estructura de diseño al menos tres partes [8, 9]:

- Bibliotecas (*Libraries*), almacenan los componentes y elementos de diseño, organizados en unidades denominadas paquetes (*packages*). Cada biblioteca puede agrupar diferentes paquetes y, a su vez, cada uno de ellos contener diferentes componentes y/o elementos; permitiendo acceder a todos o cada uno de los elementos para ser compartidos o reutilizados por diferentes diseños.

- Entidades (*Entities*), es la declaración de las entradas y las salidas de un módulo hardware o de un sistema; solamente especifica la interfaz del componente, aún sin conocer la arquitectura.

- Arquitecturas (*Architectures*), cada una de ellas especifica, dependiendo del estilo de descripción, el comportamiento del circuito, sus interconexiones, funcionamiento interno y componentes.

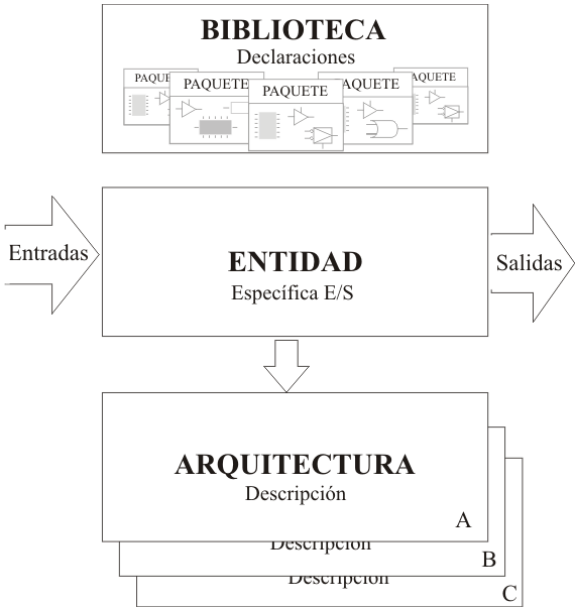


Figura 1.7. Estructura de diseño de una descripción en VHDL.

Es posible realizar diversas descripciones del funcionamiento de un circuito, por lo que pueden existir varias arquitecturas para una entidad.

En la descripción de un circuito, el hecho de que no todas las expresiones sean sintetizables se debe a que el VHDL es un lenguaje genérico para modelado de sistemas –no sólo para diseño de circuitos digitales-, por lo que hay expresiones que no pueden ser transformadas a circuitos digitales.

1.2.2 Herramientas y secuencia de diseño

Se denomina ciclo o flujo de diseño al proceso de crear un sistema de lógica programable que va desde la definición del sistema hasta su funcionamiento sobre un PLD, proceso que implica un conjunto de pasos intermedios y que no sería abordable sin la ayuda de un entorno de herramientas software que asisten al diseño, simulación, síntesis del resultado y configuración del hardware[4]; Este tipo de herramientas se denominan EDA (Automatización del Diseño Electrónico, *Electronic Design Automation*), en la Tabla 1.4 se muestran las herramientas principales para el desarrollo de sistemas con FPGAs y la compañía de desarrollo.

Tabla 1.4. Herramientas y compañías de desarrollo EDA .

Compañia	**Herramienta de Diseño**
Actel Corporation	Libero IDE
Achronix Semiconductor Corporation	Achronix CAD Enviroment (ACE)
Altera Corporation	Quartus II
Altium	Altium Designer
Atmel Coporation	Integrated Development System (IDS)
Cypress Semiconductor	Warp
Lattice Semiconductor Corporation	ispLEVER
Mentor Graphics	FPGA Advantage
Quicklogic Corporation Synplicity	QuickWorks
Synplicity	Synplicity Pro
Xilinx	ISE

El flujo de diseño se muestra en la Figura 1.8, describiendo los pasos requeridos a continuación:

Entrada de Diseño o **Descripción del diseño** (*Design entry*), para la creación del diseño pueden usarse diversos métodos, o una combinación, tal como una descripción con un HDL, captura de diagramas esquemáticos o incluso la representación gráfica de una máquina de estados.

Es conveniente establecer la división del diseño principal en módulos separados; ya que la modularidad es uno de los conceptos principales a tomar en cuenta en todo diseño, puesto que que permite crear y verificar especificaciones durante las fases iníciales de concepción y definición del sistema, lo que determina en gran medida el éxito de un proyecto. Principalmente se diferencia entre dos metodologías de diseño: Top-Down y Button-Up [8].

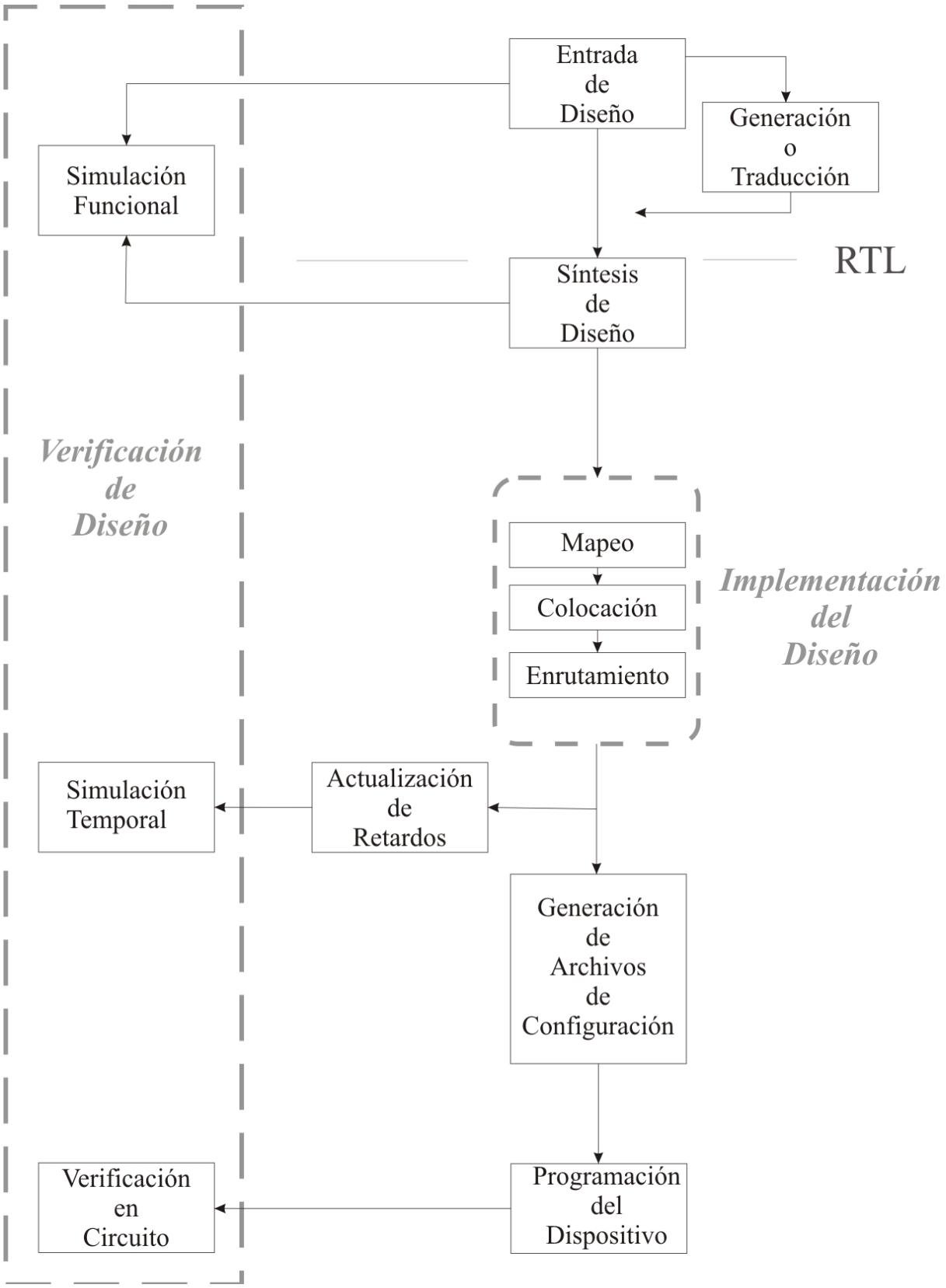


Figura 1.8. Flujo de diseño en lógica programable.

La **metodología Top-Down** consiste en que un diseño complejo, que parte desde lo más abstracto, se divide en diseños más sencillos que se puedan describir más fácilmente. En tanto, la **metodología Button-Up** consiste en describir un diseño complejo a partir de módulos más simples, partiendo de lo particular a lo general. En la práctica se hace uso de ambas metodologías, pero lo importante es establecer los mecanismos que permitan pasar de una descripción abstracta a una más detallada.

Generación o Traducción (*Translate*), los módulos son traducidos al HDL y se realiza un análisis para verificar la sintaxis y la semántica.

Simulación funcional y verificación (*Functional simulation* y *Verification*), se simula el diseño y se evalúa su comportamiento. Se comprueba que el diseño funcione adecuadamente, si no lo hace deberá modificarse.

Síntesis de Diseño (*Synthesis*), se realiza la conversión de lenguaje de alto nivel HDL, el cuál describe el circuito en RTL (*Register Transfer Level*), en una lista de conexiones a nivel de compuertas (*netlist*). Así mismo, se optimiza el desempeño del diseño por área o velocidad.

Mapeo (*Mapping*), se realiza el proceso de asignar a cada elemento lógico a un elemento físico específico que implemente la función lógica en un dispositivo configurable.

Colocación y Ruteo (*Placement* y *Routing*), en este paso se adapta el diseño a un hardware en concreto, ya sea un FPGA o un ASIC, considerando los recursos del dispositivo. Define que partes del dispositivo que contendrán las funciones del diseño y como estarán interconectadas.

Actualización de Retardos (*Back-Annotation*), se extraen los retardos de los bloques y sus interconexiones, que servirán para realizar la simulación temporal –también llamada simulación post-layout-. Estos retardos son anotados en un archivo SDF (*Standar Delay Format*, Formato Estándar de Retardos) que asocia a cada bloque o interconexión con un retardo mínimo/típico/máximo.

Simulación Temporal (*Temporal simulation*), el diseño se simula nuevamente, incluyendo los retardos; a pesar de la simulación funcional puede que el diseño no funcione cuando se programa, una posible causa es debido a los retardos internos del chip. Con esta simulación se puede comprobar el funcionamiento del sistema, y si hay errores se tiene que volver a uno de los pasos anteriores.

Generación de archivos de configuración (*Configuration file generation*), se crea un archivo con los datos para configurar el dispositivo, conocido como archivo binario (*bit-stream*).

Programación del dispositivo, se implementa el diseño en el dispositivo final, configurado por medio del archivo binario, y se comprueba el resultado.

1.3 Características de la tarjeta de desarrollo Digilab Spartan-3

La plataforma de desarrollo Digilab Spartan-3, manufacturada por la empresa Digilent[URL9], es una solución de bajo costo para el desarrollo e implementación de prototipos en circuitos digitales y de sistemas moderados a medianamente complejos; las características que la hacen atractiva para el desarrollo de prototipos en diseño lógico son las siguientes:

- FPGA Xilinx Spartan-3 XC3S200 con las siguientes características.
  - 200,000 compuertas lógicas.
  - 4,320 celdas lógicas equivalentes.
  - Doce bloques de RAM de 18K-bit (216K bits).
  - Doce multiplicadores de 18 bits.
  - Cuatro Bloques para administración digital de la señal de reloj (DCM, *Digital Clock Manager*).
- Oscilador basado en un cristal de 50 MHz, empleado como fuente principal de reloj.
- Memoria Flash de 2Mbit programable en sistema (XCF02S).
- SRAM asíncrona de 1Mbyte (256K x 32 bits) (ISSI IS61LV25616AL-10T).
- Puerto VGA de 3 bits, Puerto Serie RS-232 y Puerto PS/2.
- 8 Interruptores deslizables, 4 botones, 9 LEDs, 4 displays de 7 segmentos.
- Programación vía puerto JTAG.
- Tres reguladores de voltaje (3.3V, 2.5V, y 1.2 V).
- Tres conectores de expansión de 40 pines.

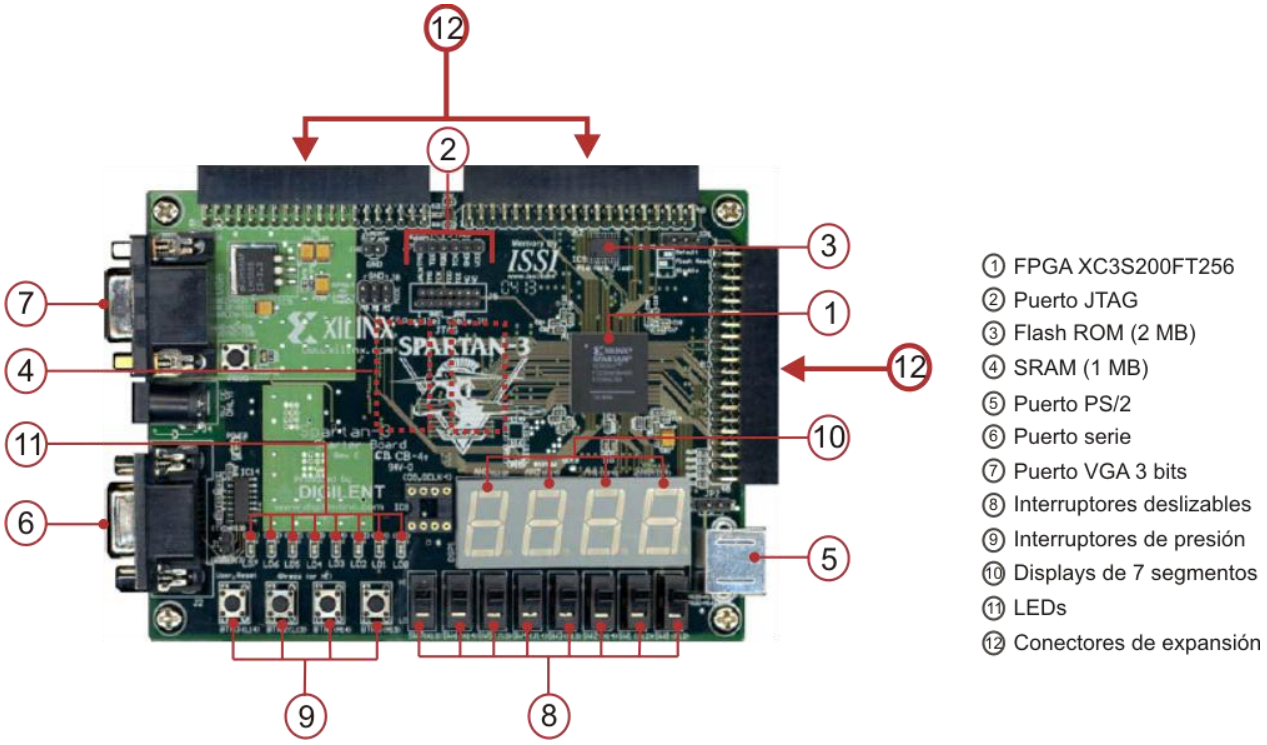


Figura 1.9. Componentes de la Tarjeta Digilent Spartan-3.

1.3.1 Caracteristícas de los FPGA Spartan 3 de Xilinx

La arquitectura de los dispositivos de la familia Spartan-3 consiste fundamentalmente de 5 elementos programables, organizados como se muestra en la Figura 1.10 [10]:

- Bloques de Lógica Configurable (CLBs, *Configurable Logic Blocks*)
- Bloques de Entrada/Salida (IOBs, *Input/Output Blocks*)
- Bloques de memoria RAM (*BlockRAM*)
- Bloques Multiplicadores Dedicados
- Administrador de Reloj Digital (DCM, *Digital Clock Manager*)

La Tabla 1.5 muestra las características del FPGA Spartan-3 XC3S200FT256 [10, URL8].

Tabla 1.5. Características del FPGA Spartan-3 XC3S200FT256.

Dispositivo	Compuertas	Celdas Lógicas	Arreglo CLB	Total CLBs	Bloque RAM (bits)	Multiplicadores Dedicados	DCMs	Total de E/S Disponibles
XC3S200	200K	4,320	24 x 20	480	216K	12	4	173

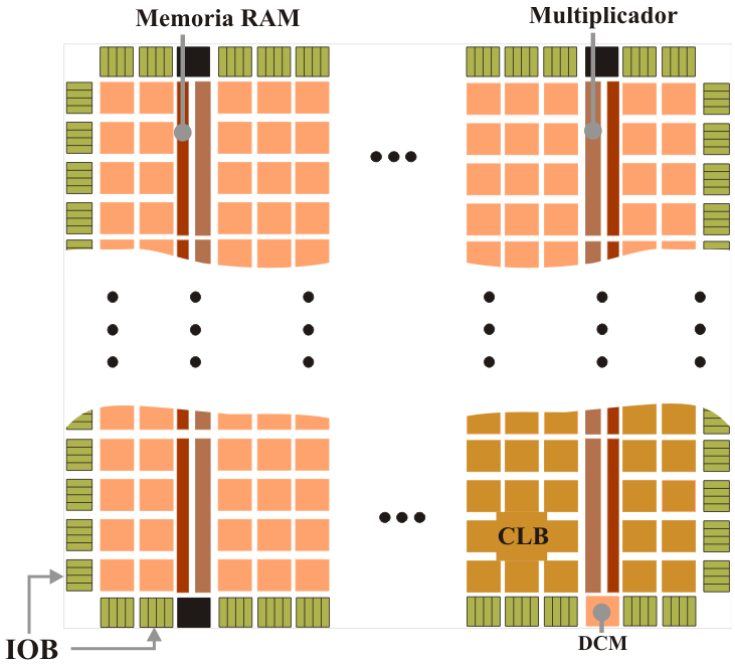


Figura 1.10. Arquitectura del FPGA Spartan-3.

1.3.1.1 Bloques de Lógica Configurable (CLBs)

El bloque de lógica configurable es el elemento básico de procesamiento en la arquitectura de los dispositivos Xilinx, basados en LUTs con en tecnología RAM permiten implementar los elementos de la lógica de ejecución y almacenamiento. Cada CLB se integra de un arreglo de 4 SLICEs (rebanadas), asociadas en pares e interconectadas tal cómo se muestra en la Figura 1.11.

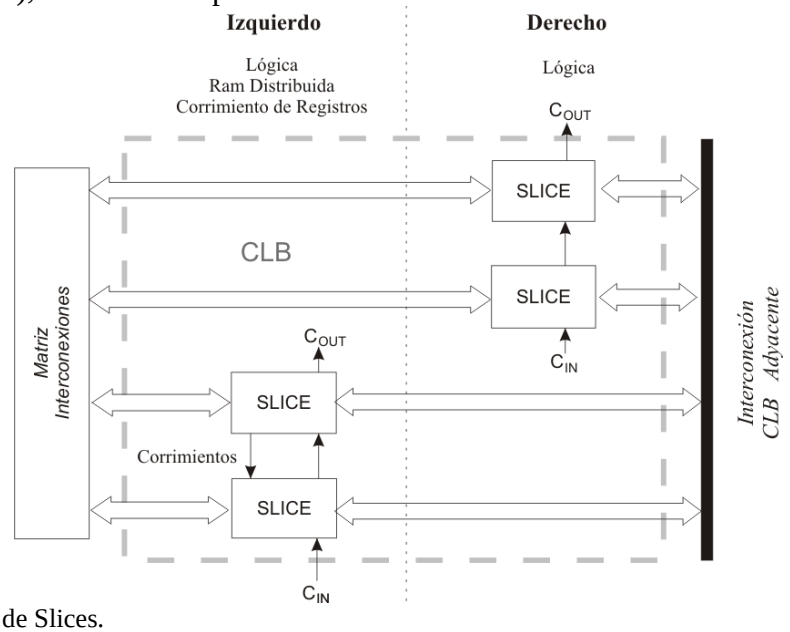


Figura 1.11. Arreglo de Slices.

Funcionalmente tanto el par derecho como el izquierdo de slices proveen las funciones lógicas, aritméticas y ROM[1]. Adicionalmente el par izquierdo soporta dos funciones añadidas, almacenamiento de datos empleando RAM distribuida y registros de corrimiento de 16 bits. Cada slice se compone de los siguientes elementos [10,11]:

2 Generadores de funciones lógicas, también conocidos como LUT, es el recurso principal para implementar funciones lógicas. Una LUT puede funcionar como cualquier función simple de cuatro entradas y una salida. La concatenación de LUTs y la interconexión de salidas y entradas permiten implementar cualquier función lógica. Adicionalmente, los LUTs dentro de cada par izquierdo de Slices pueden ser configurados como RAM distribuida, que proporcionan estructuras de memoria superficiales implementadas en CLBs, o un registro de desplazamiento de 16 bits, que es ideal para capturar datos a altas velocidades[2].

2 Elementos de almacenamiento o registros, que son programables tanto como Flip-Flop tipo D o latch (cierre) sensibles al nivel. La presencia de estos registros permite generar lógica síncrona, debido a que los elementos de registro proporcionan un medio para sincronizar los datos con la señal de reloj, lo que permite almacenar resultados intermedios, generar "iteraciones" y sobre todo, permite realizar "tuberías (*pipelines*)"[3].

[1] También conocida como ROM Distribuida, permite inicializar la memoria con datos durante la configuración del FPGA.
[2] Esta característica es muy utilizada en aplicaciones de procesamiento digital de señales (DSP, Digital Signal Processing)
[3] El concepto básico es genérico, y consiste en "desenrollar" algoritmos recursivos de ciclos de iteraciones predefinidos a un formato de "flujo de datos".

Multiplexores, elementos dedicados que combinan efectivamente LUTs de manera que permitan implementar complejas funciones lógicas. Estos recurso son empleados para optimizar una gran variedad de funciones lógicas de propósito general, en aplicaciones tales como comparadores, codificadores-decodificares y sentencias selectivas [9].

1.3.1.2 Bloques de Entrada/Salida (IOBs)

Los bloques de Entrada/Salida (IOB) proporcionan una interfaz bidireccional programable entre un pin de Entrada/Salida y la lógica interna del FPGA, con tasas de transferencia que pueden llegar a los 633 Mbit/Seg. Un diagrama simplificado de la estructura interna de un IOB aparece en la Figura 1.12, en donde se distinguen tres rutas principales de las señales dentro del IOB:

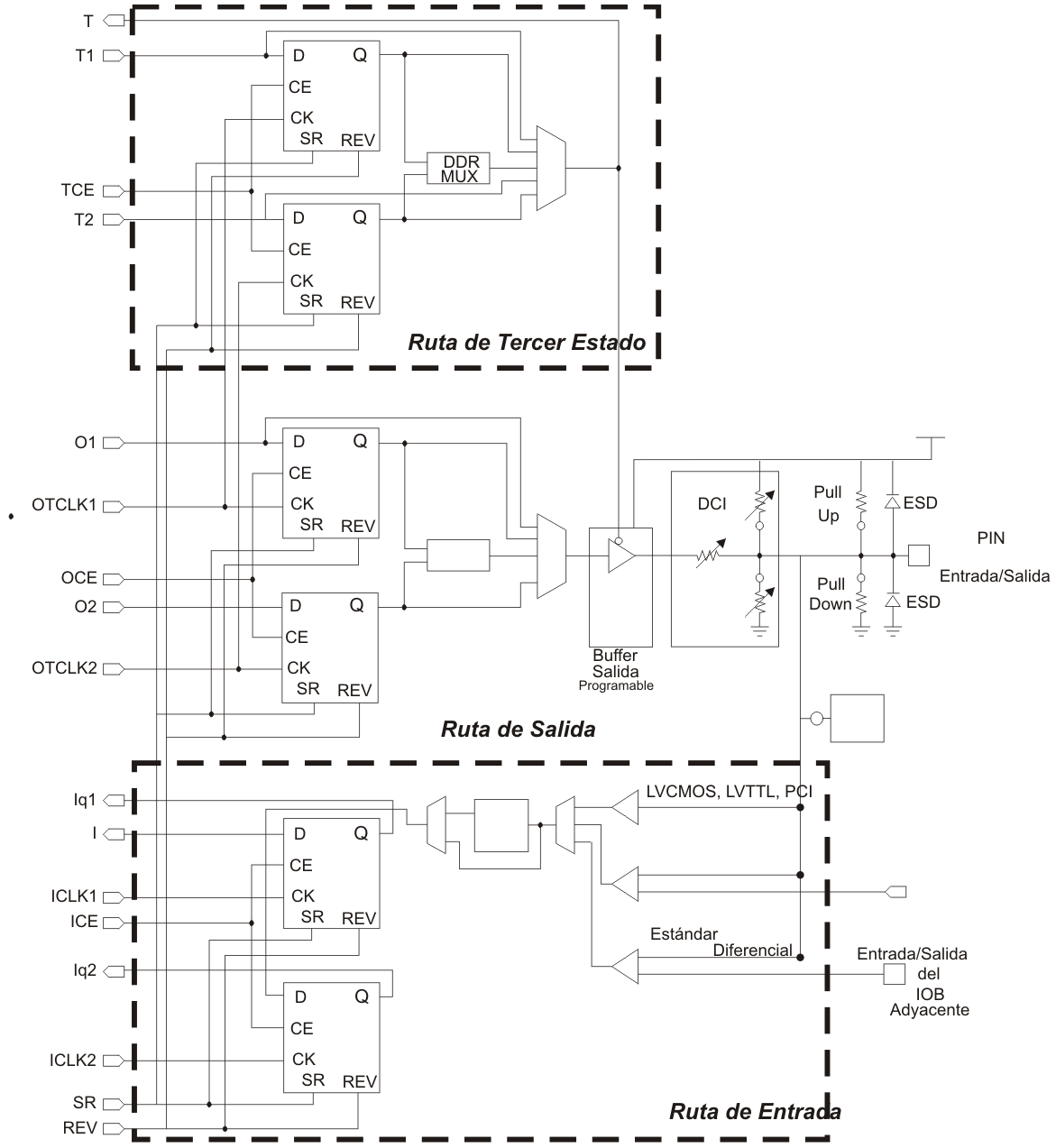


Figura 1.12. Estructura simplificada de un IOB [URL8].

La ruta de entrada, se conecta a la lógica interna del FPGA, a través de un buffer, por medio de la línea I, u opcionalmente a través de un par de elementos de almacenamiento en las líneas IQ1 e IQ2. Otro componente opcional es un elemento de retraso programable, que elimina los problemas de tiempo en la entrada, haciendo coincidir el retardo interno que existe en las líneas de reloj del FPGA, y de esta manera minimizar la diferencia en tiempo entre la entrada y las señales internas.

La ruta de salida, lleva datos desde la lógica interna del FPGA por las líneas O1 y O2, a través de un multiplexor y buffer de tercer estado, hacia los pines de salida. Además de esta trayectoria directa, el multiplexor proporciona la opción de insertar un par de elementos de almacenamiento, los cuáles además de emplearse como registros o latches sensibles al nivel, puden ser empleados para producir transmisiones de doble tasa de transferencia de datos (DDR, Double-Data-Rate).

La ruta de tercer estado, determina cuándo el buffer de salida se encuentra en alta impedancia; lleva los datos de la lógica interna del dispositivo por medio de las líneas T1 y T2, a través del multiplexor hacia el buffer de salida. Adicionalmente, el multiplexor permite incluir un par de elementos de almacenamiento [6, 10].

Las rutas de señales que entran al IOB, incluidas aquellas asociadas con los elementos de almacenamiento, tienen una opción de inversión. Cualquier inversor colocado en estas rutas es automáticamente absorbido dentro del IOB. En algunos casos, el buffer de entrada utiliza un voltaje de umbral proporcionado por el usuario, VREF. También dispone de resistores de Pull-Up[4] y Pull-Down[5], para hacer aún más flexible el tipo de entrada o salida. Por ejemplo, se puede configurar una entrada como Pull-Up para que pueda ser conectada a una salida de colector abierto.

1.3.1.3 Bloques de memoria RAM (BlockRAM)

Los Bloques de memoria RAM (*Random Access Memory*, Memoria de Acceso Aleatoria) de puerto doble, denominados por el fabricante como SelectRAM, proveen almacenamiento de grandes cantidades de datos en el dispositivo por medio de celdas de memoria RAM síncronas, incorporadas físicamente en una estructura de doble puerto –es decir, dos memorias en una e independientes-, y organizadas en bloques configurables de 18K-bits.

La cantidad total de bloques de memoria incorporados depende específicamente del tamaño del dispositivo, en el cuál los bloques RAM poseen un enrutamiento dedicado que provee la distribución de las señales de dirección y datos, así como la comunicación con el resto de los elementos en el FPGA. Adicionalmente, los bloques RAM son susceptibles de ser conectados en cascada con la finalidad de crear memorias de mayor ancho de palabra y/o capacidad de almacenamiento. Permiten implementar eficientemente RAM, ROM (*Read Only Memory*, Memoria de solo lectura) , FIFO (*First In, First Out* ; primero en entrar, primero en salir), convertidores de ancho de datos, buffers circulares, y registros de corrimiento, entre otras aplicaciones potenciales [10].

[4] Realiza una conexión del resistor a Vcco con el objetivo de a la línea un valor lógico bajo.
[5] Realiza una conexión del resistor a GND.

1.3.1.4 Bloques Multiplicadores Dedicados

Los multiplicadores dedicados que se incluyen en la arquitectura Spartan3 proporcionan la capacidad de implementar funciones aritméticas rápidas y eficientes con el mínimo uso de recursos de propósito general.

Entre las funciones de los bloques multiplicadores embebidos se encuentran la multiplicación de números de 18 bits con signo y sin signo, corrimientos, generadores de magnitud o el regreso a complemento a dos de un valor. Los multiplicadores se pueden conectar en cascada a otros bloques o con un CLB para implementar funciones más grandes y complejas [10].

1.3.1.5 Administrador de Reloj Digital

Los 4 módulos DCM (Administrador de Reloj Digital, *Digital Clock Manager*) con que cuenta la arquitectura Spartan 3, proveen soluciones de control y auto-calibración totalmente digitales para la distribución, retraso, multiplicación, división, y cambio de fase de la señal de reloj. Esto se logra mediante el empleo de un lazo de bloqueo de retraso[6] (DLL, *Delay-Locked-Loop*), que es un completo sistema de control digital que emplea la realimentación para mantener las características de la señal dentro de parámetros de retraso mínimos en distintas condiciones de voltaje y temperatura.

1.3.2 Programación de la Tarjeta Digilent Spartan-3

La tarjeta Digilab Spartan-3 incorpora la interfaz JTAG, lo que permite llevar a cabo los procesos de ISP (Programación en Sistema, *In System Programability*[7]) e ISR (Reprogramación en Sistema, *In System Reprogramability*[8]).

El puerto JTAG incorporado permite configurar directamente el FPGA Spartan-3, así como al dispositivo Flash ROM; El puerto cuenta con cuatro señales[12, 13], que se muestran en la Tabla 1.6:

Tabla 1.6. Descripción de las señales del puerto JTAG.

Señal	Descripción
TDI	Entrada a través de la cual se introducen los vectores de prueba
TDO	Salida a través de la cual se leen los vectores de prueba
TMS	Selector del Modo de prueba
TCLK	Reloj de Prueba

[6] Realizan la misma tarea que los tradicionales PLL pero de forma más robusta y menos susceptibles a las interferencias de ruido

[7] El término define la posibilidad de programar un dispositivo cuando se encuentra en una placa de circuito impreso.
[8] El término define la posibilidad de modificar la funcionalidad de un dispositivo dentro de la propia aplicación.

1.3.3 Comunicación Serie

La comunicación serial se basa en una interfaz de comunicaciones, conocida popularmente como puerto serie, en la cual los datos son enviados del emisor al receptor bit a bit empleando un único canal, esto es un enlace físico punto a punto. Los métodos de los que se hace uso en este *tipo de transmisión* son los siguientes:

Simplex , en este tipo de transmisión el transmisor y el receptor están claramente definidos y la información se envía en una sola dirección.

Duplex, Half-Duplex o Semi-Duplex, en este tipo de transmisión los datos pueden ser enviados en ambas direcciones entre dos sistemas, pero no simultáneamente, esto es, una dirección a la vez.

Full-Duplex, en este tipo de transmisión cada sistema puede enviar y recibir información al mismo tiempo, por lo que es necesario el uso de dos canales.

Así también, se hace uso de los *métodos de comunicación* que garantizan la fiabilidad en la recuperación de la información enviada por el emisor, para esto se emplean técnicas de sincronización, como son:

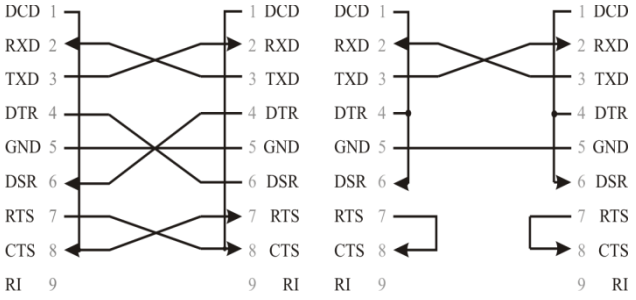
Transmisión Síncrona, método mediante el cual se envían los datos en bloques. El emisor y el receptor son sincronizados mediante una línea específica de reloj, síncrono etimológicamente significa "con reloj". En el canal de datos es necesaria la presencia de grupos de bits de comienzo y de final del bloque de datos, además de ciertos bits de corrección de errores y de control. A todo el conjunto de bits y datos se le llama trama. En este método no es necesario poner de acuerdo al emisor y al receptor de la transmisión en la velocidad de las transferencias de la información.

Transmisión Asíncrona, método en el que se añaden marcadores de comprobación y de control dentro del flujo de bits, sincronizándose al inicio de una cadena; se emplean bloques de datos pequeños, llamados palabras, así la posición de cada bit puede ser determinada temporizando los bits en periodos regulares, sin que el temporizador pierda la sincronía, con la clara desventaja de que al enviar múltiples tramas se emplean muchos bits de comprobación y control. En este método el canal permanece desocupado mientras no exista transmisión de datos y es imprescindible que el receptor y el emisor tengan configurada la misma velocidad de puerto.

1.3.3.1 Puerto Serie RS-232

Un puerto serie bajo el protocolo RS-232 (ANSI/EIA 232) define su funcionamiento por las características eléctricas, mecánicas, funcionales y modos de conexión de una interfaz de comunicación *serial asíncrona.* Las cuales se resumen como referencia en la Tabla 1.7. Para su implementación en sistemas de comunicación se requiere de un controlador electrónico denominado UART (*Universal Asynchronous Receiver and Transmiter*, Transmisor Receptor Asíncrono Universal), cuya función es convertir los bytes a un flujo de bits en serie. Además, cambia el voltaje utilizado en la señal para representar los bits y añade o extrae los bits de inicio y parada.

Tabla 1.7. Resumen de las especificaciones básicas del Estándar RS-232-C.

<table>
<tr><th>Especificación</th><th>Característica</th><th>Descripción</th></tr>
<tr><td rowspan="6">Eléctrica</td><td>‘0’ lógico en la salida</td><td>5 a 15 V</td></tr>
<tr><td>Estado indefinido</td><td>-5 a 5 V</td></tr>
<tr><td>‘1’ lógico en la salida</td><td>-5 a -15 V</td></tr>
<tr><td>‘0’ lógico en la entrada</td><td>3 a 15 V</td></tr>
<tr><td>Estado indefinido</td><td>-3 a 3 V</td></tr>
<tr><td>‘1’ lógico en la entrada</td><td>-3 a -15 V</td></tr>
<tr><td></td><td>Corriente (máx)</td><td>500 mA</td></tr>
<tr><td rowspan="2">Mecánica</td><td>Conector</td><td> 
<table>
<tr><th>Señal</th><th>DB 25</th><th>E/S</th><th>DB 9</th></tr>
<tr><td>Signal Ground</td><td>7</td><td>-</td><td>5</td></tr>
<tr><td>Transmitted Data</td><td>2</td><td>Salida</td><td>3</td></tr>
<tr><td>Received Data</td><td>3</td><td>Entrada</td><td>2</td></tr>
<tr><td>Data Terminal Ready</td><td>20</td><td>Salida</td><td>4</td></tr>
<tr><td>Data Set Ready</td><td>6</td><td>Entrada</td><td>6</td></tr>
<tr><td>Request To Send</td><td>4</td><td>Salida</td><td>7</td></tr>
<tr><td>Clear To Send</td><td>5</td><td>Entrada</td><td>8</td></tr>
<tr><td>Data Carrier Detect</td><td>8</td><td>Entrada</td><td>1</td></tr>
<tr><td>Ring Indicator</td><td>22</td><td>Entrada</td><td>9</td></tr>
</table></td></tr>
<tr><td>Longitud del bus (máx)</td><td>20 kbp - 15 m</td></tr>
<tr><td>Modos de Conexión</td><td>Full Duplex</td><td> 
</td></tr>
</table>

La transmisión serie empleando al protocolo RS-232, en su modo más simple, emplea 3 líneas de comunicación, transmisión, recepción y tierra; donde la transmisión de datos está dada por el patrón mostrado en la Figura 1.13, caracterizada en función de los siguientes parámetros [14, URL10] :

Velocidad de transferencia (Baud Rate): se determina por la tasa de baudios, unidad de medición para comunicación que indica el número de bits transferidos por segundo. Los más comunes son: 2400, 4800, 9600 y 19200.

Bits de Datos (Data bits): corresponde al total de bits con la información a transmitir. Los bits de información pueden ser 5, 6, 7 u 8.

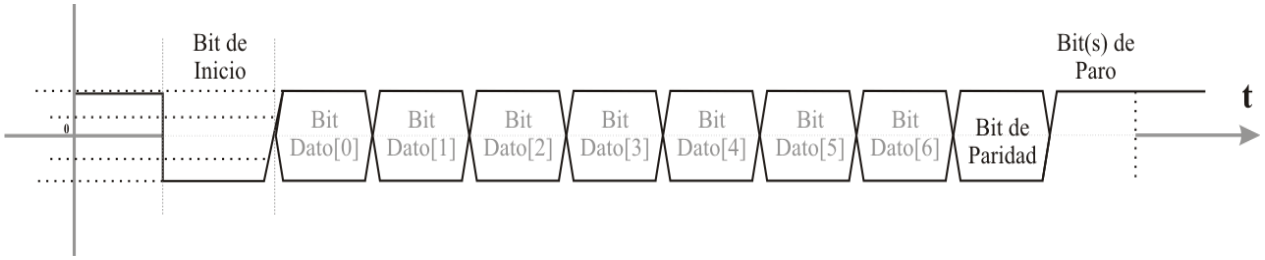


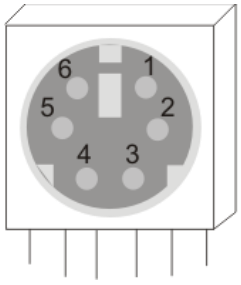
Figura 1.13. Paquete de comunicación serial empleando protocolo RS232.

Bit de Paro (Stop Bit): son empleados para señalar el término de comunicaciones en un paquete sencillo, los valores típicos 1, 1.5 ó 2 bits de paro.

Paridad (Parity): es una forma de revisión de error simple utilizada en la comunicación serial. Existen tres opciones para el bit de paridad: sin paridad, paridad par o paridad impar.

1.3.3.2 Protocolo Interfaz PS/2

El puerto PS/2 desarrollado por IBM se emplea para establecer una interfaz de comunicación *serie bidireccional síncrona* entre un dispositivo periférico de computadora, que puede ser *el teclado* o *el ratón*, eléctricamente son similares, y un controlador anfitrión: computadora, FPGA o microcontrolador, que se acoplan mediante un conector tipo mini-DIN (*Deutsches Institut für Normung eV*, Instituto Alemán de Normalización) (Figura 1.14), formado por 6 terminales de las cuales sólo 4 son operativas en el estándar PS/2 [13].



Pin	Señal	Función
1	DATA	Datos
2	n/c	Reservado
3	GND	Tierra
4	Vcc	Alimentación
5	CLK	Reloj
6	n/c	Reservado

Figura 1.14. Conector mini-DIN y funciones de sus terminales.

La comunicación físicamente se realiza por medio de dos líneas a colector abierto[9], con resistencias fijadas por el dispositivo al voltaje de alimentación, por lo que se encuentran por defecto en un estado lógico alto.

- Línea de Reloj (CLK), empleada para transmitir el reloj de sincronización.
- Línea de Datos (DATA), empleada para la transmisión de los datos serie.

[9] En una interfaz a "colector abierto" existen dos posibles estados lógicos: bajo o alta impedancia.

El protocolo de comunicación *serie bidireccional síncrona* que se emplea establece la comunicación por medio de una trama de 11 bits que presenta la siguiente estructura:

- 1 bit de inicio (0)
- 8 bits de datos (LSB-MSB)
- 1 bit de paridad (impar)
- 1 bit de paro (1)

Los datos son sincronizados mediante la señal de reloj, que tanto en la transmisión como en la recepción es generada por el dispositivo; no obstante que el controlador tiene prioridad en el control del bus, pudiendo inhibir la comunicación al introducir un *cero* -nivel lógico bajo- a la línea de reloj. En la Tabla 1.8 se muestran los estados que puede presentar el bus PS/2.

Tabla 1.8. Estados del bus PS/2.

Estado	PS/2 DATA	PS/2 CLK
Inactivo	Alto	Alto
Transmisión inhibida	Alto	Bajo
Petición de envío	Bajo	Alto

En tanto no exista comunicación se considera que el bus se encuentra en estado inactivo o en espera (*idle*) y las líneas de datos y reloj se encuentran en un nivel lógico alto; una vez que el dispositivo PS/2 envía datos, el controlador lee estos en el flanco de bajada de la señal de reloj, mientras que los datos enviados al dispositivo se leen en el flanco de subida.

La secuencia de eventos que se produce al momento que el dispositivo desea transmitir una trama al controlador (Figura 1.15), es la siguiente:

1. El dispositivo se cerciora que ambas líneas: la de Reloj y Datos se encuentran inactivas. El estado inactivo es indicado por un estado lógico ALTO.

2. Si ambas se encuentran inactivas, el dispositivo se prepara a tomar el control del bus estableciendo una *petición de envío*, cambiando el estado lógico de la *línea de datos* a BAJO, preparando así el envío del "bit de inicio".

3. El dispositivo controla el ciclo de trabajo de reloj para enviar los 11 bits de la trama serie a una frecuencia de 10 a 16.7 kHz. Esto implica que el periodo de la señal de reloj se establece entre 60 – 100 uS.

4. El controlador reconoce el inicio de la trama serial por el bit de inicio, seguido por 8 bits de datos, un bit de paridad impar y finalizando con un bit de paro en ALTO; si el dispositivo desea enviar más datos, la siguiente trama comenzará inmediatamente en el 12° bit, con el nuevo *bit de inicio*.

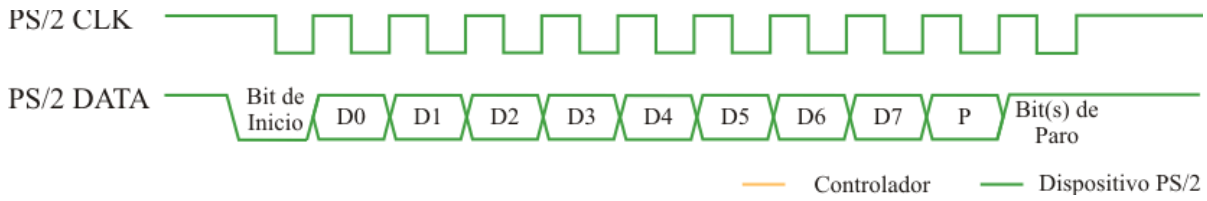


Figura 1.15. Trama serie de recepción de información del dispositivo PS/2.

La Figura 1.16 detalla la secuencia que se genera al momento que el controlador desea enviar un comando al dispositivo, la cual se establece de la siguiente manera:

1. El controlador lleva la línea de reloj en estado lógico BAJO al menos un periodo de reloj (60uS – 100uS) para inhibir cualquier nueva transmisión del dispositivo (estado transmisión inhibida) y asegurar el control del bus.

2. Se lleva la línea de datos al estado lógico BAJO proporcionando el bit de inicio de la trama, y se libera la línea de reloj para establecer una *petición de envío* e indicar al dispositivo que se enviará un comando.

3. Al detectar este estado, el dispositivo generará la señal de reloj para permitir la transmisión de los bits restantes del comando.

4. El controlador enviará los 8 bits de comando seguidos por un bit de paridad y un bit de paro

5. Una vez recibido el bit de paro, el controlador lleva la línea a un estado lógico ALTO y es liberada.

El dispositivo espera que el bus permanezca inactivo por un periodo del reloj y lleva la línea de datos a un estado lógico BAJO por un periodo de reloj, confirmando al controlador la recepción del comando (ACK, *acknowledge*). Finalmente, el bus el liberado.

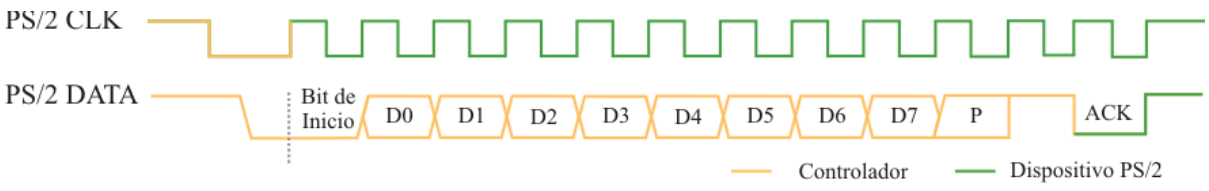


Figura 1.16. Trama serie de envío de comandos al dispositivo PS/2.

1.3.3.3 Ratón PS/2

El ratón o Mouse es un periférico de computadora que se considera, al mismo tiempo, como un dispositivo de entrada de datos y de control, dependiendo de las rutinas que maneje en cada momento. Tiene su origen en el proyecto dirigido por Douglas C. Engelbart durante la década de 1960 en el Instituto de Investigación de Stanford (SRI, *Stanford Research Institute*), donde se buscaban métodos de apuntar y señalar en un monitor de tubos de rayos catódicos (**CRT,** *Cathode Ray Tube***)** [URL11].

El principio de funcionamiento de este dispositivo es simple[10], consiste en asumir un sistema de coordenadas relativo en los planos X e Y (Figura 1.17) en el cuál, los desplazamientos del ratón son traducidos a valores numéricos que representan tanto la magnitud del desplazamiento como la dirección del mismo. En el plano X, el desplazamiento hacia la derecha genera un valor positivo y uno negativo hacia la izquierda; así en el plano Y, el desplazamiento hacia arriba genera un valor positivo y uno negativo hacia abajo. El formato de representación que se emplea en este caso, expresado en el sistema binario, es complemento a 2.

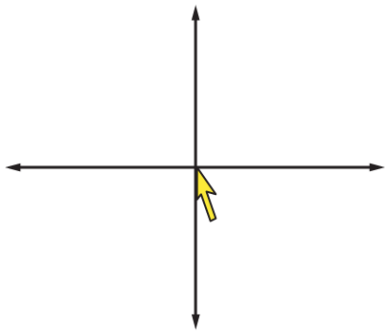


Figura 1.17. Sistema de coordenadas relativas de un dispositivo señalador.

La información del desplazamiento y el estado de los botones se agrupa en un paquete de 3 bytes, que se trasmite al controlador en la secuencia que se representa en la Figura 1.18.

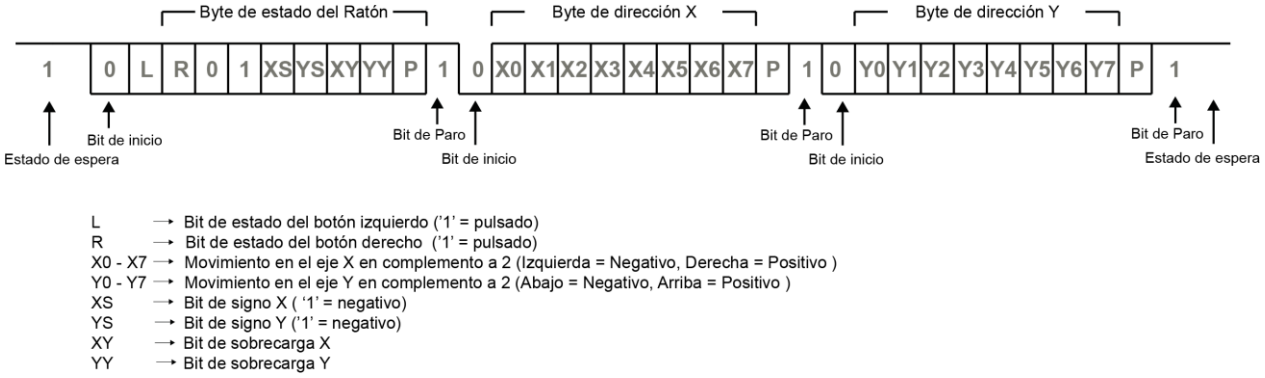


Figura 1.18. Información generada por un Ratón estándar PS/2.

[10] En la actualidad existen otros dispositivos señaladores derivados del concepto de funcionamiento del ratón: el track-ball, track-point, Glide-point, lápiz señalador; pantallas táctiles, etc;

De los datos que se transmiten, la magnitud de los índices en los campos X e Y representan el movimiento que el ratón ha experimentado desde la última transmisión de estado, por lo tanto, no representan posiciones absolutas, cuanto más rápido se mueva el ratón mayor será el número generado, pudiendo exceder el rango de valores que se comprende de -128 a 128, lo que causa que se active la bandera de sobreflujo asociada a ese eje.

Adicionalmente, es posible configurar el comportamiento del ratón, para brindarle flexibilidad en la aplicación que se desempeñe mediante la asignación de alguno de los siguientes modos de operación [15]:

Modo de flujo continúo
En este modo (*Stream Mode*), el más común y por lo tanto predefinido una vez inicializado el ratón, se envía al controlador anfitrión un reporte cada vez que detecta un movimiento o un cambio en el estado de los botones. La máxima frecuencia a la que se reportan esos datos se conoce como *tasa de muestreo*, este parámetro se encuentra en el rango de 10 a 200 muestras/seg.

Modo Remoto
En este modo (*Remote Mode*), las entradas del ratón, movimientos y botones, son leídas en la tasa de muestreo actual, actualizando contadores y banderas; pero la notificación del estado de los mismo se realiza sólo cuando es requerido por el anfitrión, haciendo uso del comando *read data*.

Modo de retorno automático
En este modo (*wrap mode*), el ratón entra en modo eco, ya que repite todos los bytes recibidos directamente al anfitrión, sin respuesta adicional o posterior, incluso si el byte representa un comando válido. Dos excepciones a esto son los comandos *reinicio* (FFh) y *reinicio en modo Wrap* (ECh). En estos casos, el ratón responde según las entradas para estos comandos descritos en la tabla A.2 en el Anexo A.

Operativamente, el ratón una vez energizado se inicializa en el modo de funcionamiento de *reset*, el cuál realiza una prueba de diagnóstico básico (BAT, *Basic Assurance Test*) y fija los valores de operación predeterminados:

- Modo Stream
- Escala 1:1
- Tasa de muestreo:100 muestras/segundo
- Reporte de datos deshabilitado.

La secuencia de interacción básica de inicialización que se da entre el ratón PS/2 y el controlador consiste de la siguiente secuencia de eventos [URL12]:

1. Al alimentarse el ratón, ejecuta una rutina interna de comprobación del dispositivo. Envía una un byte (AAh) indicando que el test se ha realizado satisfactoriamente, y envía un byte más con el identificador (ID) del dispositivo; que en este caso corresponde al dispositivo señalizador compatible con el estándar PS/2 (00h).

2. El controlador envía el comando (F4h) , habilitando el *reporte de datos* del modo stream , que por defecto se encuentra desactivado. El ratón confirma el comando mediante el byte de reconocimiento (FEh).

3. El ratón entra en modo Stream y envía el reporte de los datos por cada evento que registre el dispositivo.

Así mismo, de ser necesario, es posible forzar al ratón a regresar al estado inicial enviando el comando de *reset*:

1. El controlador envía el comando *reset* (FFh), indicando la secuencia de inicialización del dispositivo; el ratón confirma el comando mediante el byte de reconocimiento (FEh).

2. El ratón ejecuta la rutina de comprobación interna y envía 1 byte con el dato AAh y un byte con el dato 00h. El modo stream se deshabilita durante este proceso.

Los detalles de operación y comandos manejados por el ratón estándar PS/2 se presentan en el Anexo A.

1.3.4 Puerto VGA

La Matriz de Gráficos de Videos (VGA, *Video Graphics Array*) es un estándar para la gráfica de mapas de bits a color, introducido por IBM en 1987. Establece un modo de resolución de pantalla de 640 x 480 píxeles con una profundidad de color de 3 y 4 bits. De esta manera, al emplear 3 bits para señalar los componentes RGB (*Red, Green, Blue*; Rojo, Verde, Azul) que componen un píxel se obtiene una paleta de 8 (2^3) colores tal como se muestra en la Tabla 1.9. [13, URL13].

Tabla 1.9. 8 colores básicos RGB.

RGB	Color
0, 0, 0	Negro
1, 1, 1	Blanco
1, 0, 0	Rojo
0, 1, 0	Verde
0, 0, 1	Azul
1, 1, 0	Amarillo
0, 1, 1	Cian
1, 0, 1	Magenta

Un puerto VGA (Figura 1.19) estándar proporciona la interfaz de sincronización e información entre un controlador y un sistema gráfico de video (Monitor CRT, Monitor LCD, Video Proyector, etc.) mediante las siguientes señales analógicas: Rojo (R, Red), Verde (G, Green) y Azul (B, Blue), Sincronía Horizontal (HS, *Horizontal Sync*) y Sincronía Vertical (VS, *Vertical Sync*)

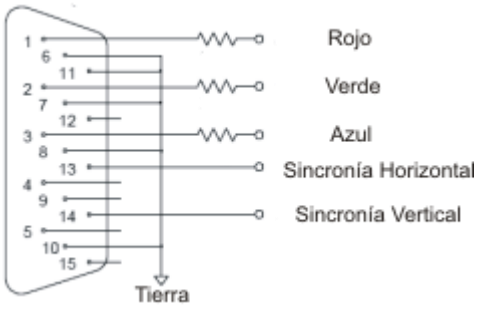


Figura 1.19. Conector de puerto VGA.

Los sistemas gráficos de visualización despliegan la información dividiendo la pantalla en cientos o miles de puntos gráficos, cada punto recibe el nombre de píxel (elemento de imagen, *picture element*). La calidad del sistema dependerá de la resolución o la cantidad de pixeles que permita visualizar; y la cantidad de bits que emplee para representar cada píxel, lo que determina cuantos colores o niveles de gris es posible desplegar.

1.3.5 Monitor

El monitor es un sistema gráfico de video, considerado como un dispositivo de salida, diseñado para mostrar las imágenes que en forma de señal provienen de un controlador gráfico; la evolución de la tecnología en estos dispositivos se ha desarrollado desde monitores de fósforo de un solo color hasta el desarrollo de monitores de pantalla de cristal líquido (LCD, *Liquid Crystal Display*, pantalla de plasma (PDP, *Plasma Display Panel*) e incluso los monitores basados en tecnología de diodo orgánico de emisión de luz (OLED, *Organic Led Emissor Diode*). Los principales parámetros que determinan el desempeño del funcionamiento de un monitor son:

Píxel, es la unidad mínima de imagen representable en un monitor.

Tamaño de punto, es el espacio entre cada píxel; entre menor sea, más uniforme es la representación de la imagen.

Resolución, es la cantidad de pixeles que es capaz de representar el monitor, se expresa por las dimensiones horizontales y verticales (ej. 640 x 480 píxeles).

Ancho de banda, se encuentra en función de la frecuencia máxima que es capaz de manejar el monitor, lo que determina la cantidad de datos que puede procesar.

Refresco de pantalla, es el número de veces por segundo que es posible redibujar la pantalla; la velocidad de refresco se mide en hertzios (Hz, 1/Segundo).

Luminosidad, es la cantidad de luz emitida desde la pantalla, representada en Lúmenes.

1.3.5.1 Monitor CRT

El monitor de Tubos de Rayos Catódicos (CRT, *Cathode Ray Tube*) se compone (Figura 1.20) de un sistema electrónico cuyo elemento básico es un tubo de rayos catódicos en el que se sitúan tres cañones de electrones que disparan constantemente un haz contra una pantalla recubierta de fósforo. El flujo de electrones de cada cañón apunta a un punto de fósforo de un color en específico: verde, rojo y azul, imprimiendo al incidir la unidad mínima de imagen: un píxel. Iluminando estos puntos con diferentes intensidades puede obtenerse cualquier color.

Para representar una imagen en la pantalla, el haz de electrones recorre sistemáticamente –barre o escanea- la superficie fluorescente de la pantalla, mediante el sistema de deflexión electromagnética que determina el trayecto del haz de electrones; comenzando por la esquina superior izquierda, recorre una fila de pixeles (deflexión horizontal) y al alcanzar el final, se apaga momentáneamente el cañón (retraso horizontal) y se coloca al principio de la siguiente fila (deflexión vertical). Cuando las filas han sido recorridas y se ha alcanzado la esquina inferior derecha, nuevamente se apaga el cañón (retraso vertical) y se retorna al comienzo. La información de cada píxel, acorde a la imagen a representar, se suministra por medio de las señales analógicas RGB para ser impresa en su posición específica en la pantalla durante el recorrido del haz de electrones, en la Figura 1.21 se ilustra el recorrido que realiza el haz de electrones.

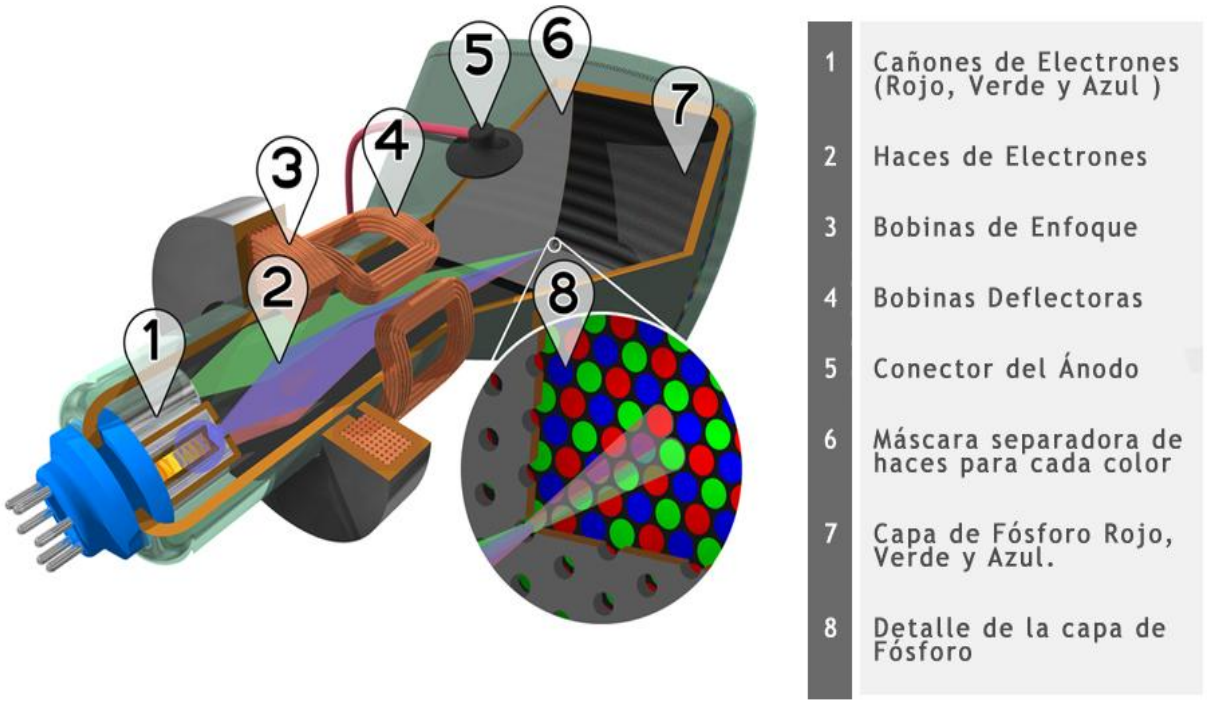


Figura 1.20. Monitor de tubo de rayos catódicos [URL14].

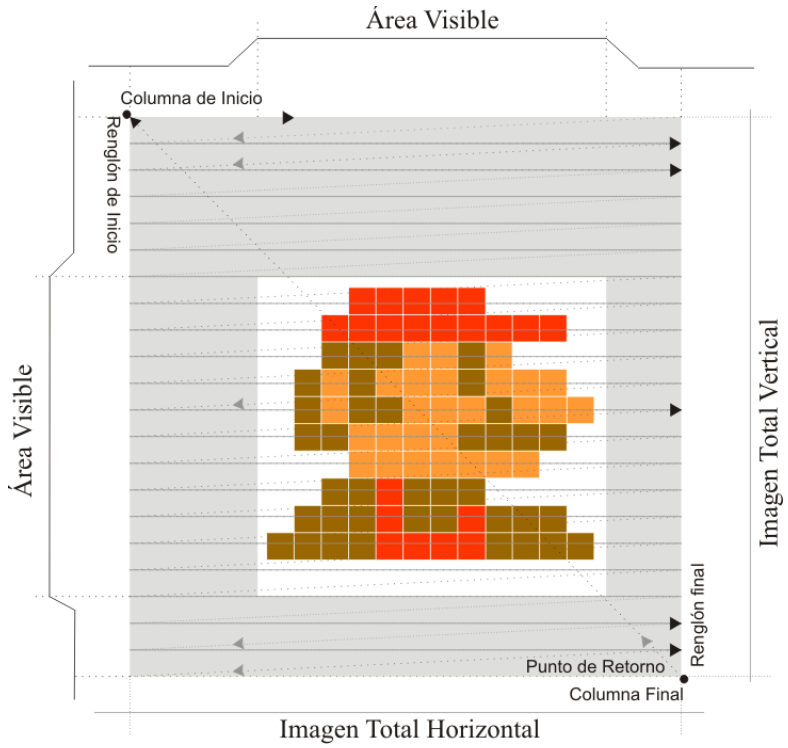


Figura 1.21. Recorrido del haz del electrones y demarcación de la pantalla.

El número de veces por segundo que la pantalla es recorrida completamente por el haz de electrones dibujando una imagen, se conoce como frecuencia de refresco, admitiendo actualmente los monitores, frecuencias que se encuentran en el rango de 50 a 120 Hz, en particular este valor se establece en un rango especificado por el fabricante. La frecuencia de operación depende directamente del cambio de línea y de pantalla, por ello, el control del sistema de deflexión precisa de las señales digitales de Sincronismo Horizontal (HS, *Horizontal Sync*) y Sincronismo Vertical (VS, Vertical Sync) para su operación [13, 2, URL13].

El Sincronismo Horizontal, que se ha adoptado también como Barrido Horizontal, es la señal que define el retraso de refresco horizontal, que es el número de veces por segundo que se traza una línea. En la Figura 1.22 se muestra el comportamiento temporal de las señal de sincronismo horizontal y las señales R,G y B.

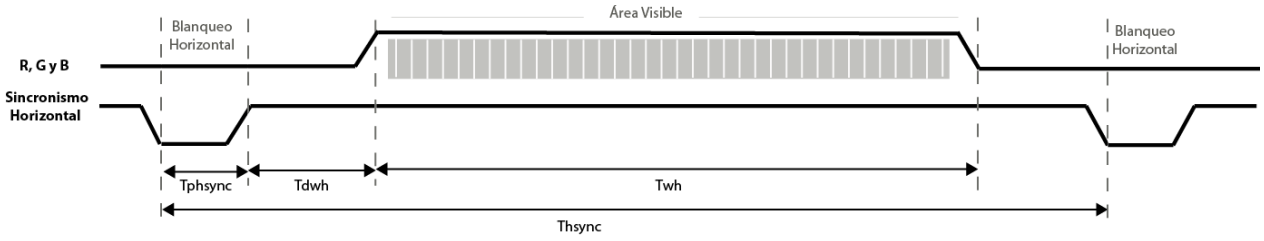


Figura 1.22. Señales de sincronización Horizontal y R, G y B.

Los parámetros que determinan el comportamiento de la señal de sincronismo horizontal corresponden a:

Tphsync, Tiempo del pulso de sincronización horizontal, es la duración del pulso de sincronización horizontal, medido en ciclos de reloj por pixel.

Tdwh, Tiempo de retardo de ventana horizontal, es la duración del tiempo entre el final del pulso de sincronización y el inicio de la ventana horizontal. La imagen puede ser recorrida a la izquierda o derecha de la pantalla mediante este parámetro. En diagramas de tiempo de video, también es referido como porche trasero.

Twh, Tiempo de ventana horizontal, es la duración del área visible en una línea de video, medida en ciclos de reloj por pixel. En diagramas de tiempo de video, es referido como tiempo activo.

Thsync, el tiempo de sincronismo horizontal, es la duración de una línea de video completa, desde el inicio del pulso de sincronización horizontal hasta el inicio el pulso de sincronización horizontal de la siguiente línea de video, medido en ciclos de reloj por pixel.

El sincronismo vertical, término adoptado también como Barrido Vertical, es la señal que define la frecuencia de "refresco" de la pantalla, que es el número de veces por segundo que se dibuja toda la pantalla. En la Figura 1.23 se muestra el comportamiento temporal de las señal de sincronismo vertical y las señales R,G y B.

Los parámetros que determinan el comportamiento de la señal de sincronismo vertical corresponden a:

Tpvsync, Tiempo del pulso sincronización vertical, es la duración del pulso de sincronización vertical, medido en líneas horizontales.

Tdwv, Tiempo de retardo de ventana vertical, es el tiempo entre el final del pulso de sincronización vertical y el inicio de la ventana vertical, medido en líneas horizontales. La imagen puede ser desplazada en la pantalla hacia abajo o arriba modificando este parámetro. En diagramas de tiempo de video, también es referido como porche trasero.

Twv, Tiempo de ventana vertical, es la duración del área visible de un cuadro ó frame de video, medido en líneas horizontales. En diagramas de tiempo de video, también es referido como tiempo activo.

Tvsync, Tiempo de sincronismo vertical, es la duración de un cuadro o frame de video completo, desde el inicio del pulso de sincronización vertical hasta el pulso de sincronización vertical siguiente, medido en líneas horizontales.

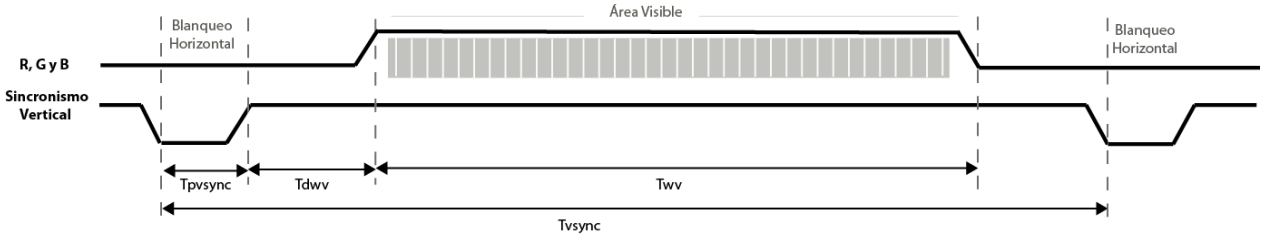


Figura 1.23. Señales de sincronización Vertical y R, G y B.

1.4 Estado del Arte

Con la finalidad de establecer marco de referencia que permita desarrollar una perspectiva del área y evaluar las tendencias en los sistemas de visualización, así como agregar claridad al proyecto de investigación que se plantea, esta sección incorpora algunas implementaciones e investigaciones relacionadas con el presente trabajo de tesis.

En [16] se plantea un proyecto de diseño de un controlador de video, que consiste en el envío de un archivo gráfico empleando el puerto serial RS-232 de la PC hasta un microcontrolador PIC16C773 que se encarga de almacenar la información en una memoria RAM; integrando un núcleo de control de video reside en un FPGA XC4010XL de Xilinx, en el que se implementa el control de video para la visualización de imágenes de 320 x 200 pixeles, con una profundidad de hasta 64 colores.

En [17] se describe la implementación sobre el FPGA de una tarjeta XSV-800 de un circuito de visualización y procesamiento de imágenes. Que consiste en una herramienta software que transforma una imagen en formato BMP a un formato propio y otra herramienta software que la guarda en la RAM, estableciendo comunicación entre la placa y la PC mediante el puerto paralelo. Una vez almacenada, la imagen se obtiene de la RAM para ser mostrada en un dispositivo VGA.

En lo que se refiere al procesamiento de imágenes, una vez que la imagen ha sido almacenada en el FPGA, se carga el filtro para tratar la imagen, se reconfigura el FPGA de acuerdo al filtro seleccionado y realizado el proceso de tratamiento de la imagen, se almacena de nuevo en la RAM para su posterior utilización. Los filtros implementados en este desarrollo son binarización, escala de grises, negativización, reducción del espectro de color a 8, suavizado y cálculo de los bordes de imagen.

En [18] se describe la implementación de un Osciloscopio Digital basado en una tarjeta de desarrollo Xess XSA-100; mediante el FPGA se realizan las funciones combinacionales y secuenciales para: capturar la señal mediante un acondicionador de señal, que consiste en un convertidor A/D para muestrear la señal y cuantificarla, guardar los datos adquiridos mediante una memoria SRAM sintetizada en el propio dispositivo lógico, tratamiento de los datos a representar, y generación de la imagen a visualizarse en un dispositivo VGA con resolución de 640 x 480 pixeles y profundidad de color de 6 bits.

El proyecto contempla también una interfaz del FPGA con un ratón de computadora, para establecer el control de las funciones del sistema como: escalas, disparo y tipo de acoplamiento.

El sistema desarrollado en [19], describe un programa con funciones similares al PAINT que incorpora el sistema operativo Windows, en un FPGA Cyclone II, fundamentado en un proyecto existente de la Universidad de Cornell [URL15].

En esta implementación se emplea una resolución de 640 x 480 pixeles en la operación del controlador VGA que despliega las funciones que son transmitidas de la tarjeta de desarrollo Altera DE2 a la pantalla. El controlador de un mouse USB se integra mediante el empleo del NIOS II y da la posibilidad de dibujar o pintar, o básicamente sólo manipular las funciones del sistema.

Los algoritmos de dibujo de las diferentes formas que adquiere el pincel de pintado y el dibujo de figuras y formas en la pantalla se realizan mediante el empleo de varios algoritmos gráficos; para líneas y círculos se emplea el algoritmo de Bresenham's [20], un algoritmo de relleno de bordes [21] se emplea para la detección de una figura cerrada y el color de relleno dentro de sus límites y la técnica de esgrima [22] se aplica para reducir el área de escaneo. Así, las funcionalidades proporcionadas por el sistema mediante el dibujo con el pincel son: dibujo de un punto, línea, cuadro, círculo, polígono, borrado con la goma, pintura en aerosol, relleno de color, selección de color, limpieza del área de dibujo y selección de tonalidad mediante una paleta de colores.

En el desarrollo que se establece en [23], se propone un sistema de captura y transmisión de imágenes para picosatélites Cubesat, implementado con tecnología FPGA y un procesador embebido; el sistema incorpora una cámara de video estándar como fuente de imágenes, limitando la resolución de captura a 240x270 pixeles; despliegue de las imágenes en una pantalla TFT, almacenamiento de la imagen capturada en una memoria SRAM y envío de los datos de la imagen empleando un puerto de salida RS-232 conectado a un radio de comunicación para permitir la transmisión a un dispositivo remoto.

Como parte de un proyecto de investigación de Maestría en [24], Armandas Jarusauskas, propone una implementación del juego clásico Pong, el cual consiste en un juego para dos personas, donde cada jugador intenta golpear la pelota hacia el oponente. El proyecto se desarrolla bajo la tarjeta de desarrollo Spartan-3E, contemplando las funciones de visualización de imágenes generadas para la dinámica del juego con resolución de 640x480 píxeles y profundidad de color de 3 bits, aceleración de la pelota, generación de sonido y visualización de información textual. En tanto, la interacción con el sistema se realiza integrando las señales generadas por un control de mandos de la plataforma de videojuegos NES de Nintendo.

En [25] describen el trabajo sobre una plataforma FPGA para el desarrollo de un IP core genérico para adquisición y desplegado de imágenes de 640 x 480 pixeles en escala de grises de 8-bits. Empleando para la adquisición de imágenes un decodificador de video compuesto configurado para recibir una señal de video estándar NTSC y convertirla en información de video compuesto en formato YCrCb 4:2:2, compatible con el estándar de 8-bits ITU-R BT.656. La información de luminancia de los pixeles de la imagen se almacena en una memoria SSRAM, desplegándose posteriormente por medio de un bloque controlador de monitor VGA.

En [26] proponen emplear un FPGA como un sistema de visión en tiempo real que simula una prótesis visual, transformando la imagen de una cámara de video en una imagen que representa la percepción de sensaciones visuales, que sería encausada para aplicar una estimulación eléctrica directamente sobre las capas neuronales del cerebro responsables de la visión.

La implementación en el FPGA efectúa la adquisición de las imágenes por medio de una cámara CMOS, a una resolución de 640 x 480 pixeles y una tasa de refresco de 60 Hz, los datos una vez procesados son enviados con la misma resolución a unos lentes de realidad virtual mediante un puerto VGA, permitiendo adicionalmente emplear un monitor VGA externo como visor dual o alternativo de la imagen de salida. El control e interacción de los parámetros del sistema se realiza por medio de un control remoto infrarrojo.

2. Diseño e Implementación del Sistema

En este capítulo se presenta el diseño e implementación de los módulos que forman el sistema de visualización de imágenes en el FPGA y la integración en una sola entidad, aplicando la metodología de desarrollo Top-Down.

Se presenta en primera instancia la descomposición jerárquica del sistema en módulos y componentes funcionales, seguido del diseño, implementación e integración de los componentes en módulos; y posteriormente la integración del módulo jerárquico en una sola entidad.

2.1 Metodología de Diseño

La metodología que se plantea a manera de directriz corresponde a la metodología de diseño *Top-Down*, que consiste en encontrar la solución de un problema mediante la aplicación sistemática de la descomposición en subproblemas cada vez más simples (aplicando la máxima de dividir para vencer); que, por la naturaleza del proyecto de tesis se considera la más adecuada y por tanto factible para el desarrollo de un sistema robusto.

Las principales ventajas que ofrece el desarrollo de un sistema mediante la metodología propuesta son:

Primero, la información se estructura en forma modular.

Segundo, clarifica la estructura y funciones de los módulos.

Tercero, la partición e independencia de módulos evita errores en el sistema.

Cuarto, la supresión de detalles hace los errores en la estructura más aparentes.

Quinto, incrementa la reutilización del diseño.

Sexto, incrementa la productividad del diseño.

Séptimo, el diseño puede ser probado módulo a módulo.

La aplicación de dicha metodología exige una forma clara de organizar el diseño, por lo que se plantea la creación de un *diseño modular jerárquico*, que consiste en construir un nivel de descripción funcional de diseño debajo de otro, de forma que cada nuevo nivel posea una descripción más detallada del sistema [8].

2.2 Organización del Sistema

En la creación de diseños jerárquicos es muy útil la realización de bloques funcionales o módulos, por ello, aplicando la metodología antes expuesta al problema planteado, en la Figura 2.1 se aprecia el marco de trabajo que permite definir los módulos que integran el sistema y sus interfaces, la división se realiza en partes funcionalmente independientes, especificando la división del sistema en sus componentes Hardware y Software.

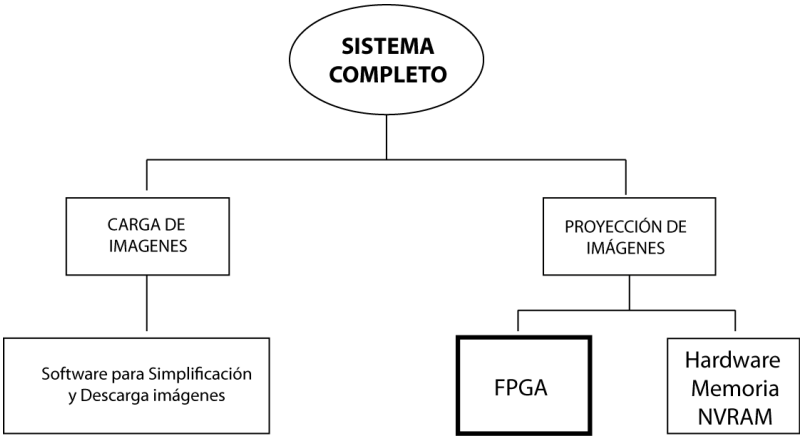


Figura 2.1. Diagrama a bloques General del Sistema.

Una gran parte de la funcionalidad del sistema radica en módulos que son descripciones de hardware implementadas en el FPGA, la funcionalidad con que cuenta se establece en 3 elementos y tareas principales:

1. Interfaz de Video: Proveer la funcionalidad de recuperar la información almacenada y desplegarla mediante el empleo de un puerto VGA, así como el control de una interfaz que permita el uso de un ratón estándar PS/2.
2. Interfaz Serie: Proveer capacidad de comunicación serial mediante el protocolo RS-232 (ANSI/EIA 232), recibe y discrimina la información para ser almacenada.
3. Almacenamiento de Imágenes: Proveer la funcionalidad de escribir y leer información en un arreglo de memorias NVRAM.

Al especificar la funcionalidad de cada bloque, estos se describen en entidades atómicas, que desempeñan una tarea en particular, permitiendo un diseño más sencillo y fácil de describir, con la ventaja adicional que pueda ser reutilizado en descripciones posteriores; aplicando esta metodología se obtiene el diagrama a bloques de la Figura 2.2, en el que cada módulo contiene uno o más componentes que proporcionan la funcionalidad, y un componente de control que permite la sincronización de operación entre las entidades, y a su vez la comunicación con el resto de los componentes de control para obtener una funcionalidad de conjunto.

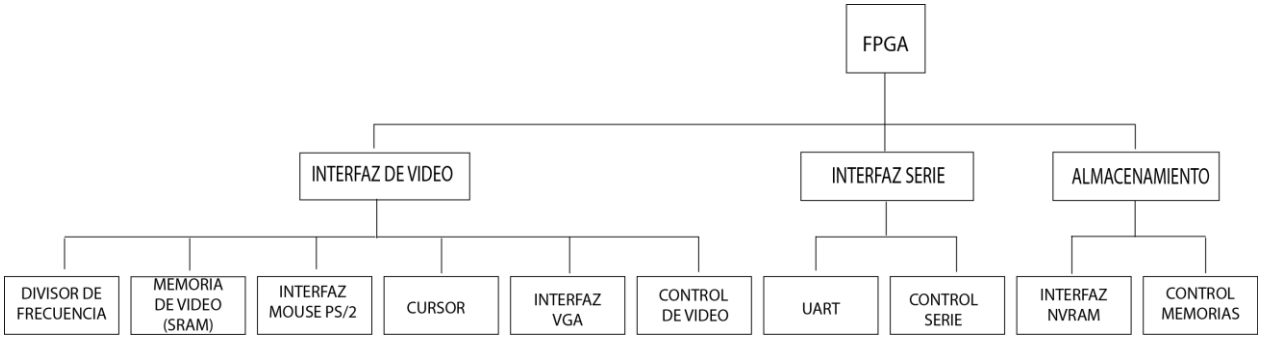


Figura 2.2. Diagrama a bloques de las descripciones de hardware.

2.1 Módulo de Interfaz de Video

Este módulo contiene los componentes de la interfaz de video: Divisor de Frecuencia, Memoria de Video (SRAM), Interfaz Mouse PS/2, Cursor e Interfaz VGA; que permiten crear una memoria de video para almacenar y refrescar una imagen a desplegar en un monitor VGA; así como establecer una interfaz que permita interactuar y representar en el monitor los movimientos y acciones de un ratón estándar PS/2.

2.1.1 Componente Divisor de Frecuencia

Componente cuya función consiste en dividir la frecuencia principal de la señal de reloj a la mitad de la frecuencia de la señal provista, considerando las siguientes características de operación:

- Divisor de frecuencia por 2
- Frecuencia de operación: 50 MHz
- Frecuencia de Salida: 25MHz

2.1.1.1 Descripción Funcional

En la Figura 2.3 se muestra el símbolo con las entradas y salidas del componente Divisor de Frecuencia.

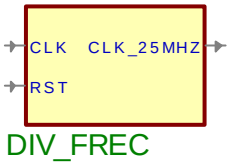


Figura 2.3. Símbolo del componente Divisor de Frecuencia.

En la Tabla 2.1 se describe la funcionalidad de cada uno de los puertos del componente,

Tabla 2.1. Descripción de Puertos módulo Divisor de Frecuencia.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de interfaz de memoria de video	
CLK_25MHZ	Salida	Nivel	Señal de reloj de 25 Mhz

2.1.1.2 Descripción del Hardware, Diseño e Implementación

En la figura 2.4 se muestra el diagrama a bloques del módulo Divisor de Frecuencia.

Figura 2.4. Diagrama a bloques de la descripción funcional del componente Divisor de Frecuencia.

La implementación se realiza mediante lógica secuencial síncrona, observándose en la Tabla 2.2 cómo en los flancos de subida de la señal de reloj, se niega el estado de salida, obteniendo como resultante una señal que duplica el ciclo útil o de trabajo de la señal original.

Tabla 2.2. Descripción del proceso divisor de frecuencia.

```
1  DIVISOR: process(RST, CLK)
2  Begin
3     if (RST = '1') then
4         CLK_DIV <= '0';
5     elsif (CLK='1' and CLK'event) then
6         CLK_DIV <= not CLK_DIV;
7     end if;
8  end process DIVISOR;
```

2.1.2 Componente de Memoria de Video SRAM.

Componente que proporciona la interfaz de acceso para la escritura y lectura de una memoria SRAM de 256Kx16 ISSI IS61LV25616AL-10T, la cuál, por las prestaciones de desempeño en los tiempos de acceso de lectura y escritura, se emplea como memoria de refresco de video, ofreciendo las siguientes características:

- Frecuencia de operación: 50 Mhz
- Bus de datos Unidireccional (Lectura/Escritura)
- Síncrono
- Tamaño de palabra de 2 bytes

2.1.2.1 Descripción Funcional

La figura 2.5 muestra el símbolo equivalente a la descripción de hardware en el diagrama esquemático del componente controlador de memoria de video SRAM.

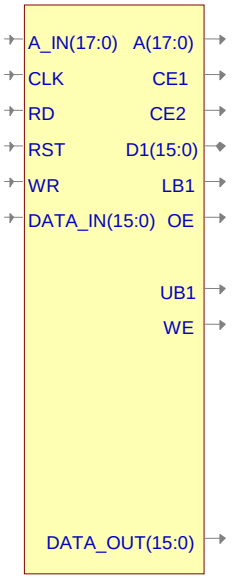


Figura 2.5. Símbolo del módulo de memoria de video SRAM.

En la Tabla 2.3 se describe la funcionalidad de cada uno de los puertos de los que se compone el componente de memoria de video SRAM.

Tabla 2.3. Descripción de Puertos módulo memoria de video SRAM.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de Interfaz Control de Video	
A_IN	Entrada	17:0	Bus de dirección de memoria
DATA_IN	Entrada	15:0	Datos para la escritura en memoria
DATA_OUT	Salida	15:0	Datos de lectura de memoria
RD	Entrada	Alto	Señal de control Lectura
WR	Entrada	Alto	Señal de control Escritura
		Señales de interfaz y Control SRAM	
A	Salida	17:0	Bus de dirección de memoria
D1	Salida	15:0	Bus de datos de memoria
CE1	Salida	Alta	Señal de habilitación (Chip Enable) memoria 1
CE2	Salida	Alta	Señal de habilitación (Chip Enable) memoria 2
LB1	Salida	Alta	Acceso al Byte de Datos menos significativo
UB1	Salida	Alta	Acceso al Byte de Datos más significativo
OE	Salida	Alta	Señal de Salida Habilitada (Output Enable)
WE	Salida	Alta	Señal de hablitación de Escritura
DATA_OUT	Salida	15:0	Bus de datos leídos

2.1.2.2 Descripción del Hardware, Diseño e Implementación

En la figura 2.6 se muestra en un diagrama a bloques la configuración de la memoria SRAM, permitiendo la funcionalidad de una memoria de video.

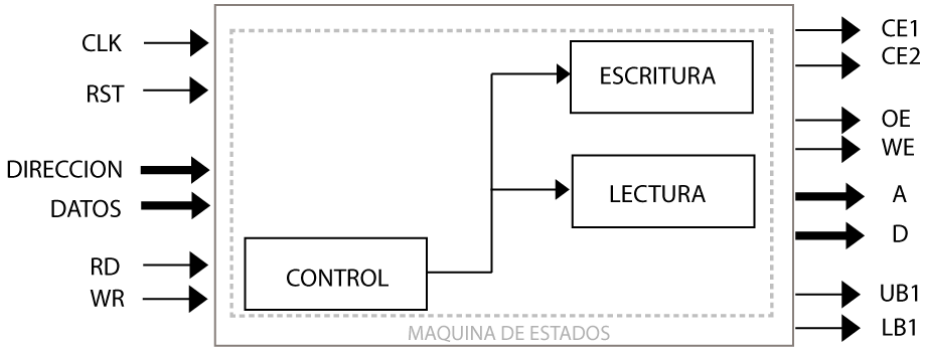


Figura 2.6. Diagrama a bloques de la descripción de hardware componente memoria de video SRAM.

La implementación remite al detalle esquemático, que se muestra en la guía de usuario de la tarjeta de Spartan-3[13], en el que se observa la capacidad de las memorias de configuración y operación de manera independiente, lo que permite la posibilidad de seleccionar el empleo de una u otra mediante el control individual de Selección (CE1, CE2). Para el manejo de un tamaño de palabra de 2 bytes se emplea el control compartido del byte alto (UB, *Upper Byte*) y byte bajo (LB, *Lower Byte*) activado, lo que permite el acceso conjunto al bus de datos bajos y altos (D1), como se muestra en las sentencias de la Tabla 2.4.

Tabla 2.4. Sentencias de control de individuales de la memoria SRAM.

```
1  CE1 <= '1';  --SRAM Desactivada
2  CE2 <= '0';  --SRAM Activa
3  UB1 <= '0';
4  LB1 <= '0';
5  D1 <= (others => 'Z');
```

La secuencia de control de acceso para la Lectura y Escritura de la SRAM se realiza mediante una máquina de estados, que se muestra en la Figura 2.7, la cual permite ejecutar la operación determinada en el puerto de entrada (Lectura –RD- ó Escritura –WR-) en un solo estado y regresar inmediatamente al estado inicial de espera.

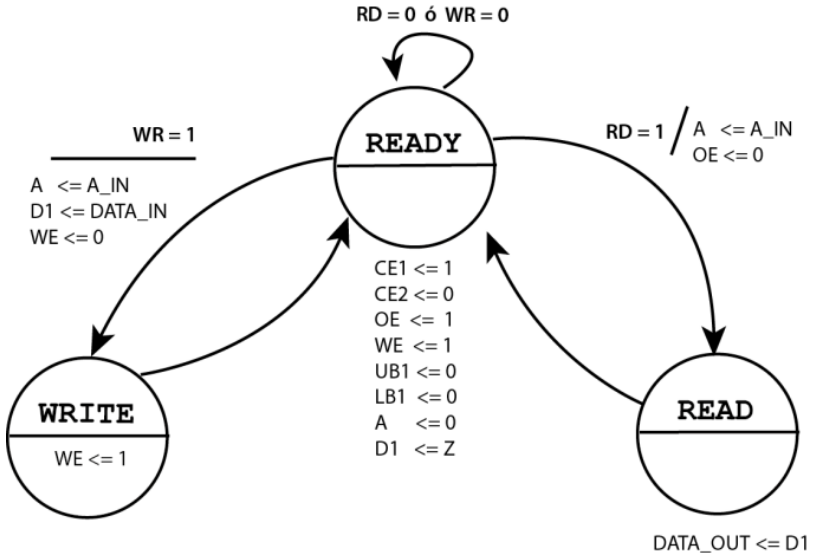


Figura 2.7. Máquina de estados de accesos para Lectura y Escritura de la memoria de video SRAM.

2.1.3 Componente Interfaz Mouse PS/2

Componente que proporciona una interfaz bidireccional síncrona entre el controlador y un ratón estándar PS/2, registrando los eventos del dispositivo: desplazamiento y pulsación del botón izquierdo o derecho, con las siguientes características:

- Frecuencia de Operación 50 Mhz
- Transmisión Bidireccional
- Compatible con Mouse estándar PS/2
- Paridad por Hardware (impar) en datos enviados
- Paridad por Hardware (impar) en datos recibidos

2.1.3.1 Descripción Funcional

En la figura 2.8 se muestra el símbolo con las entradas y salidas que esquematiza al componente de la interfaz Mouse PS/2.

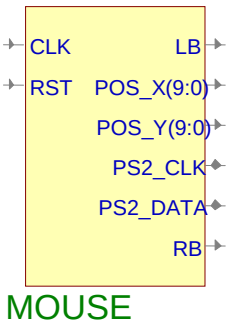


Figura 2.8. Símbolo del módulo de la interfaz Mouse PS/2.

En la tabla 2.5 se describe la funcionalidad de cada uno de los puertos de los que se compone el componente,

Tabla 2.5. Descripción de Puertos módulo interfaz Mouse PS/2.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset asíncrono de sistema externo
		Señales de Transmisión	
PS2_CLK	Bidireccional	Triestado	Señal de Reloj de sincronización
PS2_DATA	Bidireccional	Triestado	Señal de Datos
		Señal de Estado	
LB	Salida	Alto	Señal de estado de Botón Izquierdo
RB	Salida	Alto	Señal de estado de Botón Derecho
POS_X	Salida	9:0	Bus de posición eje X
POS_Y	Salida	9:0	Bus de posición eje Y

2.1.3.2 Descripción funcional del Hardware, Diseño e Implementación

De acuerdo al protocolo de interfaz PS/2 (1.3.3.2) y la secuencia de eventos que se establece para lograr la interacción de operación mediante el protocolo de interfaz PS/2 y entre el controlador (1.3.3.3), se establece el diagrama a bloques del hardware, en la Figura 2.9, que brinda la operación de la interfaz Mouse PS/2.

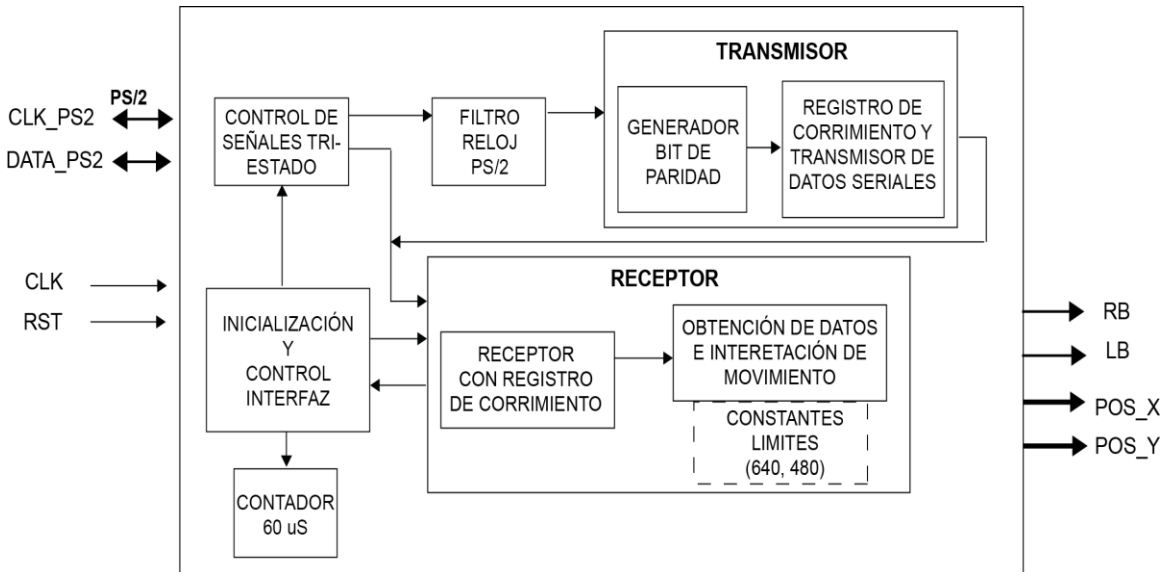


Figura 2.9. Diagrama a bloques de la descripción funcional del módulo de la interfaz del Mouse PS/2.

La primera consideración, con respecto a la implementación del módulo, se centra en la especificación del protocolo de comunicación PS/2, que establece que la interfaz con el ratón se realiza por medio de dos líneas a colector abierto, lo que hace necesario un control lógico de señales triestado para determinar el estado individual de cada línea (CLK, DATA) como entrada ó salida. El control de las señales se muestra en la Tabla 2.6.

Tabla 2.6. Control de las Señales triestado del ratón.

```
1  PS2_DATA <= 'Z' when DATO_DIR = '0' else DATO_SALIDA;
2  PS2_CLK <= 'Z' whe MOUSE_CLK_DIR = '0' else MOUSE_CLK_BUF
```

Es necesario considerar que la frecuencia de la señal de reloj del ratón, al ser mucho menor que la frecuencia de operación de la interfaz, presenta falsos positivos en los flancos; resultando en la necesidad de un filtro que garantice estabilidad en los datos de la línea, lo que se resuelve mediante un registro de corrimiento de 8 bits que responde al estado presente en la línea del reloj del ratón por cada flanco de subida del reloj de operación, el filtro se aprecia en la descripción de la Tabla 2.7.

Tabla 2.7. Filtro de la línea de reloj del ratón.

```
1  if (CLK'event and CLK = '1') then
2     fill_filter(7 downto 0) := fill_filter (6 downto 0) & PS2_CLK;
3     if (fill_filter = "11111111") then
4        MOUSE_CLK_FILTER <= '1';
5     elsif(fill_filter = "00000000") then
6        MOUSE_CLK_FILTER <= '0';
7     end if;
8  end if;
```

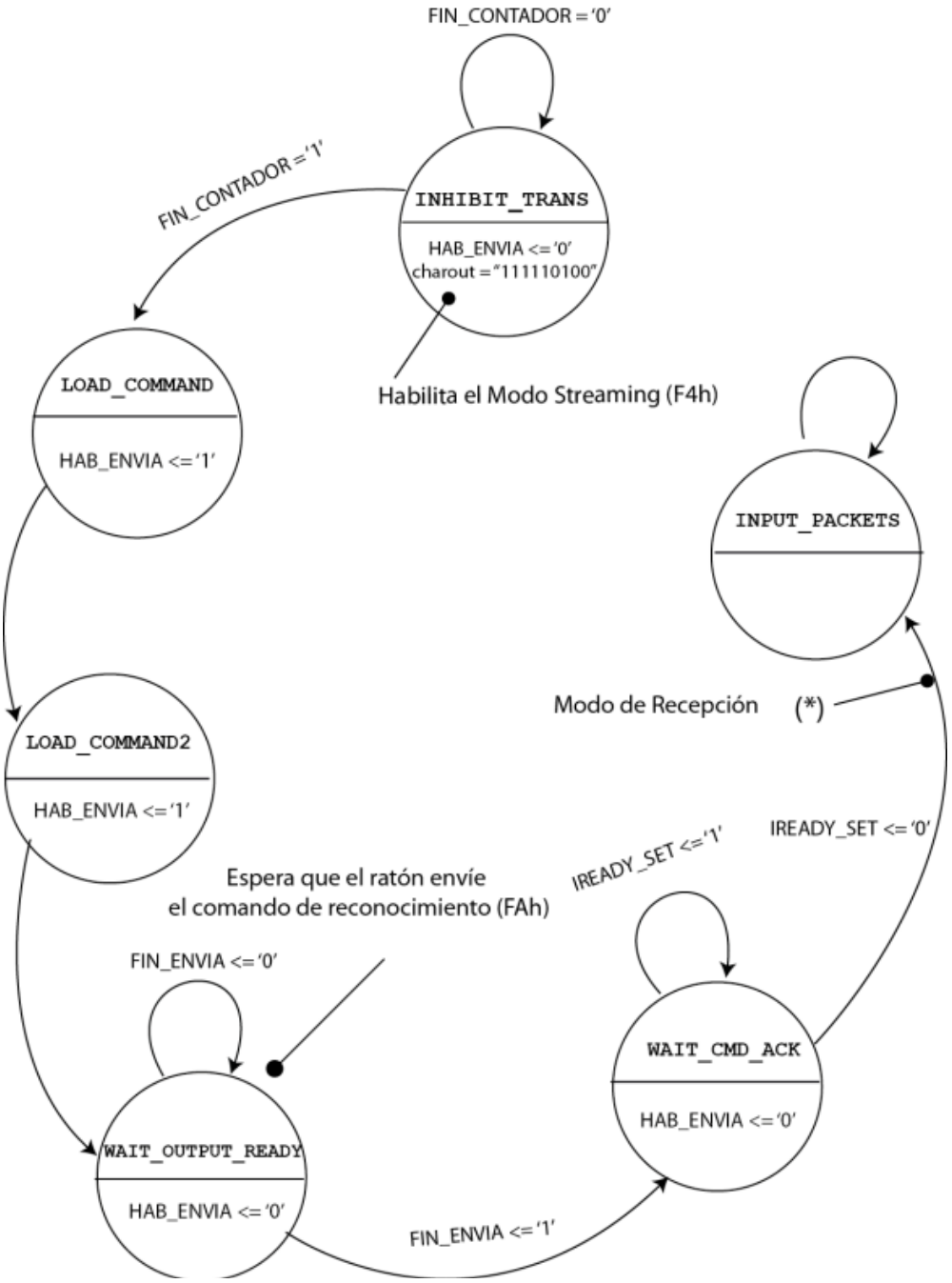


Figura 2.10. Máquina de estado del control de la interfaz mouse PS/2

Establecidas las consideraciones físicas que permitan la operación del puerto de comunicación, se identifica la necesidad de un proceso encargado de ejecutar las rutinas necesarias para la inicialización, envío y recepción de datos, apegado al protocolo de comunicación serie bidireccional síncrono PS/2. Este proceso se identifica en el diagrama a bloques de la Figura 2.9 como proceso de inicialización y control de interfaz y su funcionamiento se determina por la máquina de estados de la Figura 2.10.

En el estado INHIBIT_TRANS, ya que el ratón ha sido alimentado y ha envíado los códigos de reconocimiento AAh y 00h, el módulo debe llevar la línea de reloj a un estado lógico bajo, por al menos 60 uS, para inhibir cualquier nueva transmisión del ratón y asegurar que se toma control de la línea de datos; esto se realiza con ayuda del proceso que permite temporizar el cambio de estado del registro de control *FIN_CONTADOR* a partir de un contador de 1,536 ciclos de subida del reloj entrada, que en función del reloj de entrada en las especificaciones (50 MHz) ofrece 61.44 uS. La descripción VHDL del contador se presenta en la Tabla 2.8.

Tabla 2.8. Descripción del contador de inhibición de transmisión.

```
1  if RST = '1' then
2     wait_counter <= (others=>'0');
3     FIN_CONTADOR <= '0';
4  elsif CLK'event AND CLK = '1' then
5     if wait_counter(10 downto 9) = "11" then
6        FIN_CONTADOR <='1';
7     end if;
8  end if;
```

El estado *LOAD_COMMAND* y *LOAD_COMMAND2* habilita el envío del comando para inicializar el ratón en modo *Stream* (F4h), esto representa la configuración en la que el ratón enviará la información referente al movimiento y estado de los botones en paquetes de 3 bytes. En el proceso de envío de datos seriales por el ratón, la estructura de cada paquete o trama de comunicación serial que se emplea establece la estructura:

- 1 bit de inicio (0)
- 8 bits de datos (LSB-MSB)
- 1 bit de paridad (impar)
- 1 bit de paro (1)

Lo que hace necesario completar la trama con la generación del bit de paridad impar, mediante la aplicación en el byte de datos de un circuito de generación de paridad implementado por compuertas XOR, que ofrece un 1 si un número impar de sus entradas es 1. La sentencia que describe este circuito se muestra en la Tabla 2.9.

Tabla 2.9. Sentencia de un circuito de generación de paridad XOR.

```
1  not (charout(7) xor charout(6) xor charout(5) xor charout(4) xor
        charout(3) xor charout(2) xor charout(1) xor charout(0));
```

En la Figura 2.11 se aprecia el proceso necesario para el envío de un comando o un byte de información al ratón, operación que se realiza mediante la conmutación del puerto de datos triestado para establecer en la línea de datos del ratón, el valor del bit más significativo de la variable **cadena**, que contiene la trama, seguido del corrimiento de la variable, sucesivamente hasta completar la transmisión de los 11 bits.

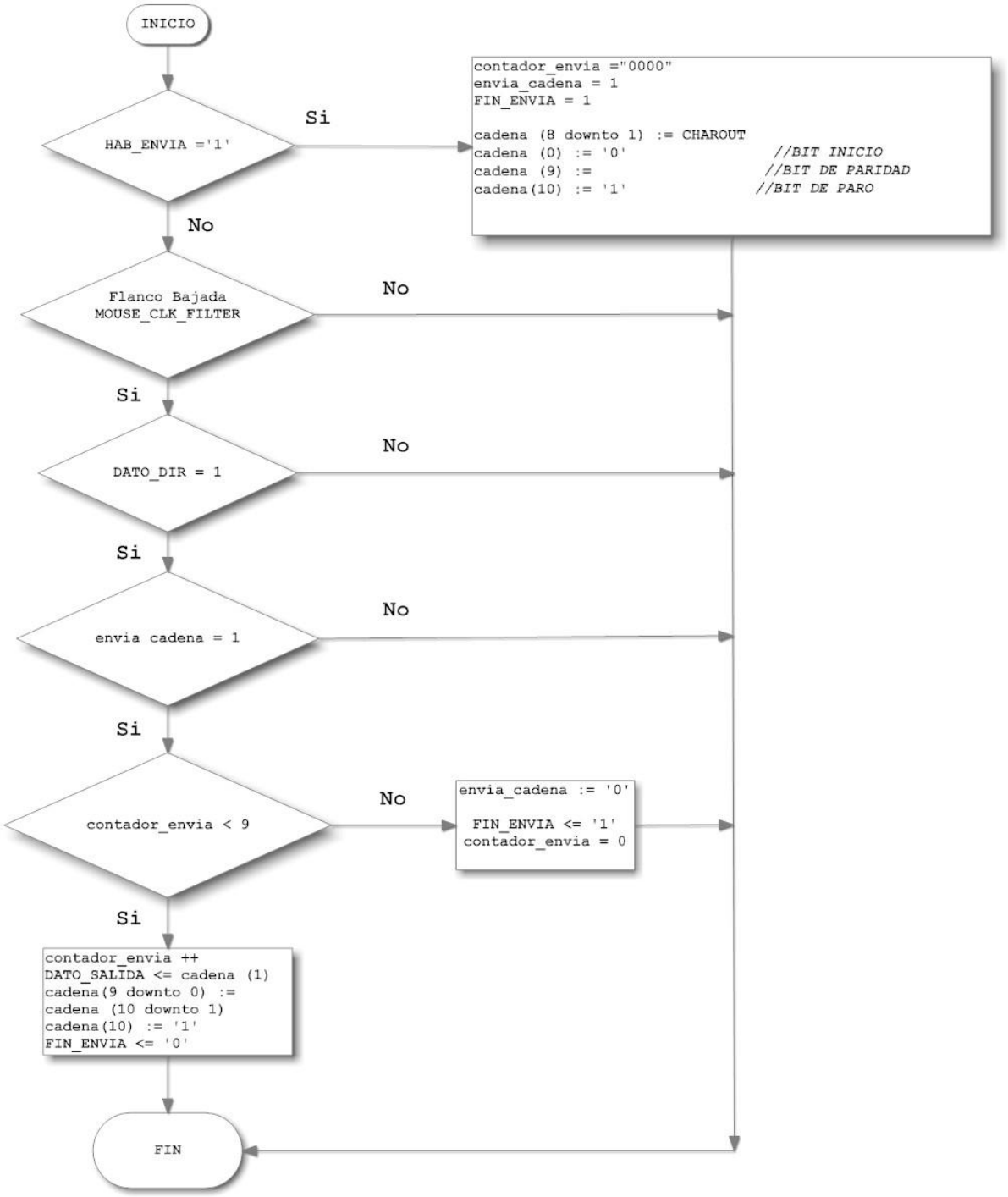


Figura 2.11. Diagrama de flujo del proceso de transmisión de un comando al ratón.

El estado WAIT_OUTPUT_READY espera que la trama de datos que contiene el comando se envíe al ratón, una vez que se ha enviado, el estado WAIT_CMD_ACK espera que el ratón responda con el comando de reconocimiento (FAh), indicador de que se ha habilitado el modo de *Stream*; por ello, el siguiente estado, INPUT_PACKETS, libera la línea de datos y entra en estado perpetuo de recepción de paquetes, que el ratón transmite a una tasa de 100 muestras/segundo; en la Figura 2.12 se aprecia el proceso y las operaciones que se emplean al recibir la trama serial de 3 bytes de información, proveniente del ratón, almacenarla en la variable de lectura, realizar el corrimiento y contabilizarlo; sucesivamente hasta ubicar el último de los bits recibido en su posición original dentro de la estructura de la trama.

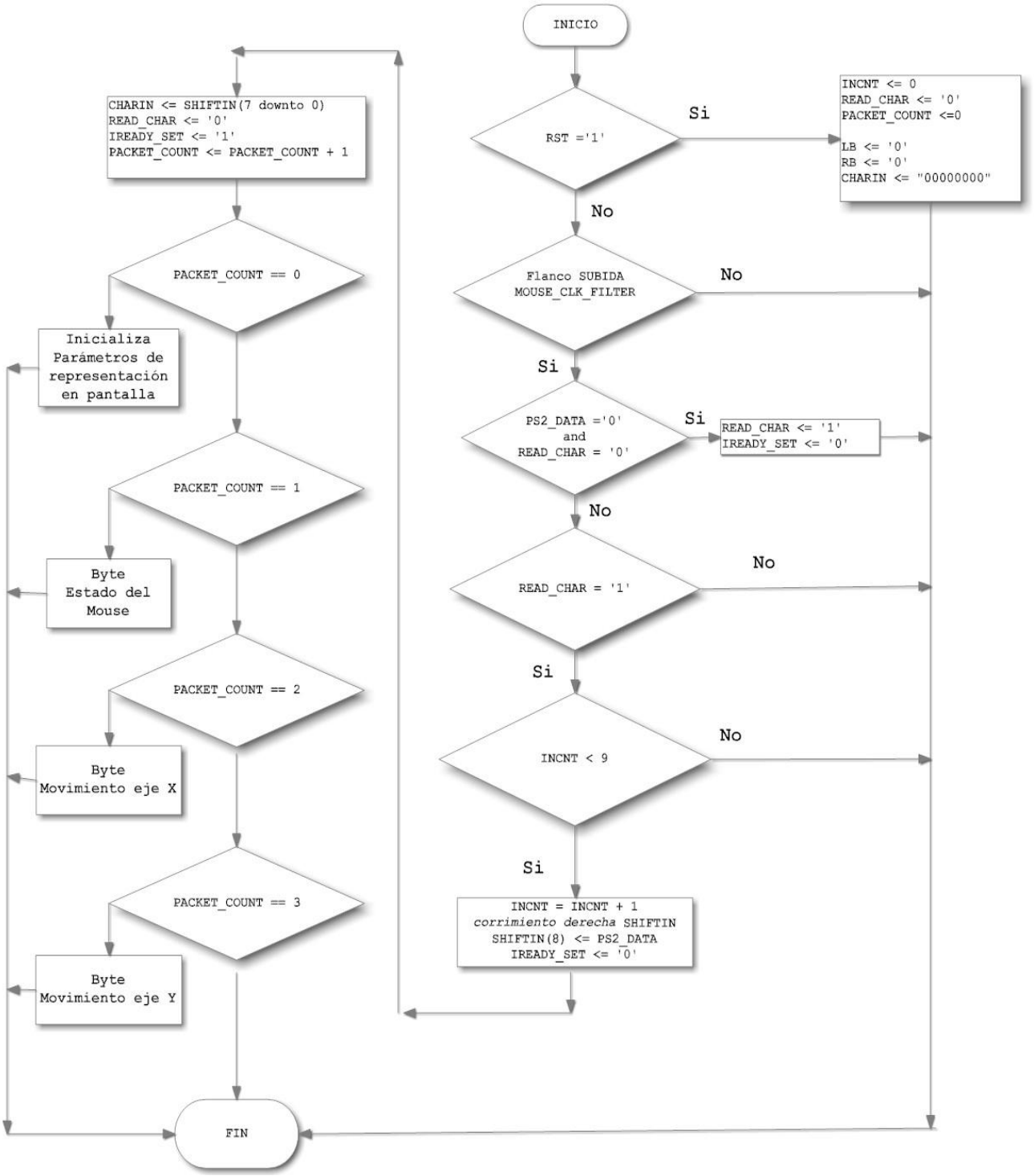


Figura 2.12. Diagrama de flujo del proceso de recepción de 3 bytes de información del ratón.

La trama de 3 bytes recibida contiene la información generada respecto al desplazamiento relativo y el estado de los botones del ratón, la Figura 1.18 hace referencia a esta información; los valores de desplazamiento obtenidos al no ser absolutos, requieren que sean interpretados y adecuados al marco de referencia en el que operarán, y al ser este bidimensional, se representa en sistema cartesiano con dos constantes: MAX_X y MAX_Y, máxima posición en X e Y respetivamente. Los cuales por consideraciones de especificación del proyecto, se establecen en 640 y 480, que representa las proporciones de la norma VGA.

La inicialización de parámetros de representación en la pantalla se realiza al recibir por única ocasión el paquete 0, que corresponde al byte de reconocimiento (FAh) al configurarse al ratón en modo streaming; se establecen las coordenadas iniciales en el punto medio de los valores máximos en X e Y (320, 240).

Del byte de estado del ratón se obtienen los bits de estado de los botones derecho (RB) e izquierdo (LB) del ratón.

Los Bytes de movimiento del eje X e Y, deben traducirse a la abstracción en que se representan los movimientos del ratón en el IV cuadrante del eje cartesiano, Horizontal y Vertical, como se muestra en la Figura 2.13.

Para determinar la posición absoluta en la representación, se realiza la operación aritmética del diferencial de posición, que en el caso del eje X se obtiene mediante la suma y en caso del eje Y, al ser la parte superior la menos negativa, mediante resta. Las sentencias expresas se muestran en la Tabla 2.10, en donde se observa la extensión del signo de desplazamiento.

Tabla 2.10. Sentencias aritméticas de la posición del cursor del ratón.

```
1   NEW_cursor_vertical <= cursor_vertical – (PACKET_3(7) & PACKET_3(7) &
    PACKET_3 )

2   NEW_cursor_horizontal <= cursor_horizontal + (PACKET_2(7) & PACKET_2(7)
    & PACKET_2 )
```

Los márgenes mínimos y máximos en que se circunscribirá el ratón, se establecen considerando que el desplazamiento máximo que se muestrea por paquete es de 127 pixeles y que los desplazamientos hacia abajo e izquierda tienen una representación negativa (complemento a 2); por tanto, una representación negativa que excede los márgenes mínimos, considerándose para fines prácticos de comparación como positivos, establece el margen inferior del eje Y y superior de la pantalla, como se muestra en la Tabla 2.11.

Tabla 2.11. Comparación del margen inferior del eje Y.

```
1   if cursor_vertical < 128 and NEW_cursor_vertical > 256 or
    NEW_cursor_vertical < 2 then
2      cursor_vertical <= (others => '0');
3   end if;
```

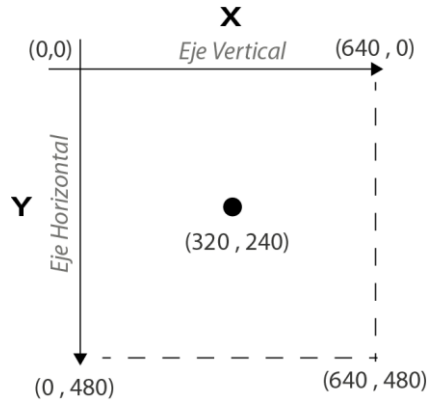


Figura 2.13. Representación en una pantalla de los movimientos del ratón.

La comparación con la constante máxima en Y determina el límite en la parte inferior de la pantalla, se muestra en la Tabla 2.12

Tabla 2.12. Comparación del margen superior del eje Y.

```
1  if NEW_cursor_vertical > MAX_Y then
2     cursor_vertical <= MAX_Y;
3  Else
4     cursor_vertical <= NEW_cursor_vertical;
5  end if;
```

En la Tabla 2.13 se muestra, la restricción para el margen inferior del eje X en la parte izquierda de la pantalla.

Tabla 2.13. Comparación del margen inferior del eje X.

```
1  if cursor_horizontal < 128 and NEW_cursor_horizontal > 256
      or NEW_cursor_horizontal < 2   then
2        cursor_horizontal <= (others=>'0');
3  end if;
```

Finalmente, en la Tabla 2.14 se muestra como se establece el límite superior en X, o parte derecha de la pantalla

Tabla 2.14. Comparación del margen superior del eje X.

```
1  if NEW_cursor_horizontal > MAX_X then
2     cursor_horizontal <= MAX_X;
3  Else
4     cursor_horizontal <= NEW_cursor_horizontal;
5  end if;
```

2.1.4 Componente Cursor

El componente cursor crea un puntero señalizador que se sobreimprime a la imagen en función de las coordenadas de posición del ratón con las siguientes características de operación:

- Diseño asíncrono
- 8 Cursores seleccionables
- Color del cursor dinámico

2.1.4.1 Descripción Funcional

El símbolo que esquematiza las entradas y salidas del componente Cursor, se muestra en la Figura 2.14.

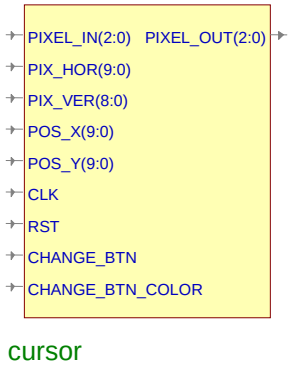


Figura 2.14. Símbolo del componente Cursor.

La funcionalidad de cada uno de los puertos de los que se compone el componente se describe en la Tabla 2.15.

Tabla 2.15. Descripción de Puertos módulo Cursor.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 25 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de Interfaz de Operación	
CHANGE_BTN	Entrada	Alto	Señal de Cambio de tipo de Cursor
CHANGE_BTN_COLOR	Entrada	Alto	Señal de Cambio de Color de Cursor
		Señales de Información y Control Interfaz Video	
PIXEL_IN	Entrada	2:0	Señal con información del píxel activo RGB
PIXEL_HOR	Entrada	9:0	Posición horizontal del píxel actual
PIXEL_VER	Entrada		Posición vertical del píxel actual
		Señales de Información de Interfaz de Mouse	
POS_X	Entrada	9:0	Información de la posición horizontal del cursor del Mouse
POS_Y	Entrada	9:0	Información de la posición vertical del cursor del Mouse
		Señales de Información de interfaz de Cursor	
PIXEL_OUT	Salida	2:0	Bus con información a desplegar del píxel activo RGB en función

2.1.4.2 Descripción del Hardware, Diseño e Implementación

En la Figura 2.15 se muestra un diagrama a bloques del componente generador del cursor, en el que el comparador evalúa la posición del ratón y la posición que se está representando en el monitor, lo que permite determinar la salida.

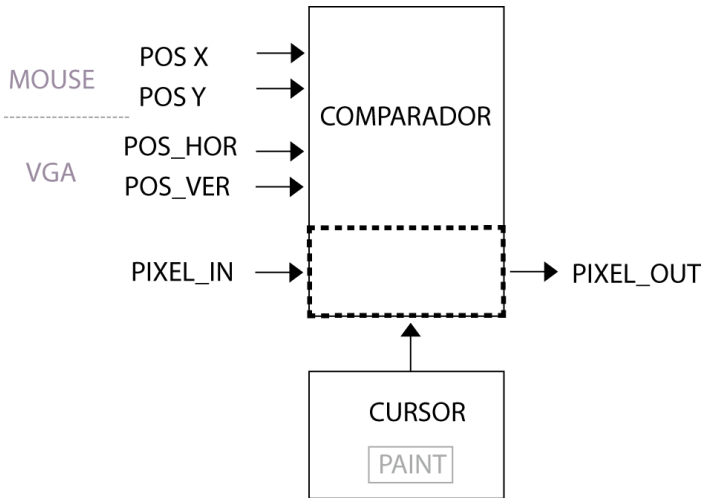


Figura 2.15. Diagrama a bloques de la descripción de hardware módulo Cursor.

El Cursor realiza la representación de la posición del ratón en la pantalla, generando las imágenes que se sobrepondrán en la posición del cursor a partir de lógica combinacional; el diseño de la imagen se realiza en el eje cartesiano, lo que permite determinar la posición de cada pixel referenciado al punto cartesiano (0,0), que será la posición actual del ratón, para dar forma a los cursores que se muestran en la Figura 2.16.

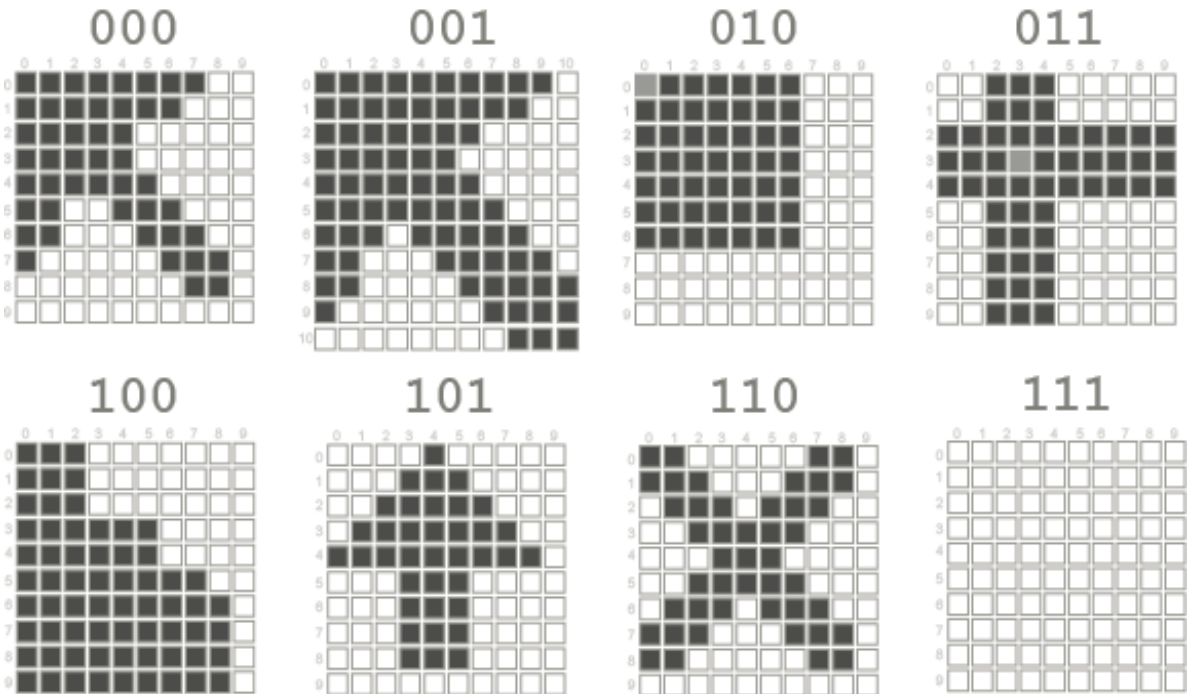


Figura 2.16. Cursores del ratón y posición de selección.

La selección de los cursores, en la Figura 2.16 se realiza por medio de la señal *PAINT1*, un contador realiza un incremento al detectar un nivel alto en la entrada *CHANGE_BTN* y un selector conmuta entre cada ecuación de generación de imagen de acuerdo al estado del contador. La Selección de cursores se muestra en la Tabla 2.16.

Tabla 2.16. Selector de Cursores.

```
1  with COUNTER select
2        PAINT <= PAINT1 WHEN "000",
3        PAINT2 WHEN "001",
4        PAINT3 WHEN "010"
5        PAINT4 WHEN "011",
6        PAINT5 WHEN "100",
7        PAINT6 WHEN "101",
8        PAINT7 WHEN "110",
9        PAINT8 WHEN others;
```

Por ejemplo el cursor de la posición “000” se genera a partir de la lógica combinacional de la Tabla 2.17.

Tabla 2.17. Lógica combinacional para el cursor “000”.

```
1  PAINT1 <= '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 7) and
                      ('0'&PIX_VER = POS_Y )else
2            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 6) and
                      ('0'&PIX_VER = (POS_Y + 1)) else
3            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 4) and
                      ('0'&PIX_VER = (POS_Y + 2)) else
4            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 4) and
                      ('0'&PIX_VER = (POS_Y + 3)) else
5            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 5) and
                      ('0'&PIX_VER = (POS_Y + 4)) else
6            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 1) and
                      ('0'&PIX_VER = (POS_Y + 5)) else
7            '1' when (PIX_HOR >= POS_X + 4 and PIX_HOR <= POS_X + 6) and
                      ('0'&PIX_VER = (POS_Y + 5)) else
8            '1' when (PIX_HOR >= POS_X and PIX_HOR <= POS_X + 1) and
                      ('0'&PIX_VER = (POS_Y + 6)) else
9            '1' when (PIX_HOR >= POS_X + 5 and PIX_HOR <= POS_X + 7) and
                      ('0'&PIX_VER = (POS_Y + 6)) else
10           '1' when (PIX_HOR = POS_X ) and ('0'&PIX_VER = (POS_Y + 7))
                       else
11           '1' when (PIX_HOR >= POS_X + 6 and PIX_HOR <= POS_X + 8) and
                      ('0'&PIX_VER = (POS_Y + 7)) else
12           '1' when (PIX_HOR >= POS_X + 7 and PIX_HOR <= POS_X + 8) and
                      ('0'&PIX_VER = (POS_Y + 8)) else
13           '0';
```

El pixel que se desplegará se discrimina mediante la comparación del cursor seleccionado, si existe coincidencia, se toma el valor inverso de los bits más significativos del pixel actual y el bit menos significativo de la selección de color, generando un tonalidad diferente de la imagen a la

que se sobrepone el cursor, resultando en la simplificación de la ubicación visual; en caso de que no exista coincidencia, la salida del pixel desplegado corresponde al de entrada del componente. Estas sentencias combinacionales se muestran en la Tabla 2.18.

Tabla 2.18. Sentencias de selección de salida de pixel.

```
1  PIXEL_OUT <= INVERSO when PAINT = '1' else PIXEL_IN;
2  INVERSO <=  NOT PIXEL_IN (2 downto 1)  & COLOR(0);
```

2.1.5 Componente Controlador VGA

El componente controlador VGA es compatible con el estándar VGA, funcional por lo tanto en la mayoría de monitores CRT y LCD, provee las señales de sincronismo y tiempos de video para generar una imagen de video en modo de 3 bits, o profundidad de 8 colores, con una resolución de 640 x 480 píxeles. Por flexibilidad la implementación considera la ubicación de la memoria de video fuera del componente así, ésta puede ser una memoria de video dedicada o compartida en el sistema. El componente contempla las siguientes características:

- 3 bpp(bits por píxel) VESA interfaz estándar VGA
- Resolución de 640 x 480 píxeles
- Diseño síncrono
- Frecuencia de operación: 25 Mhz

2.1.5.1 Descripción Funcional

En la Figura 2.17 se muestra el símbolo equivalente a la descripción de hardware en el diagrama esquemático del componente VGA.

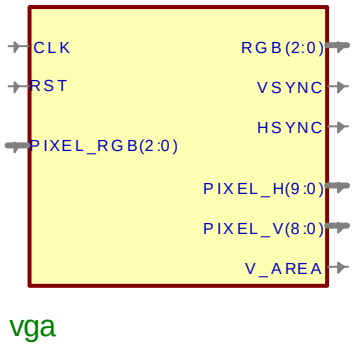


Figura 2.17. Símbolo con la descripción del hardware del componente Controlador VGA.

En la tabla 2.19 se describe la funcionalidad de cada uno de los puertos de los que se compone el componente controlador VGA.

Tabla 2.19. Descripción de puertos módulo controlador VGA.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 25 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de interfaz de memoria de video	
PIXEL_RGB	Entrada	0:3	Información del píxel activo en formato RGB
		Señales de Interfaz control de video	
PIXEL_H	Salida	0:10	Posición horizontal del píxel actual
PIXEL_V	Salida	0:9	Posición vertical del píxel actual
V_AREA	Salida	Nivel	Señal de habilitación usada para indicar si la posición actual en la pantalla corresponde a: 1 = Área visible 0 = Fuera del Área visible
		Señales de Interfaz VGA	
RGB	Salida	0:3	Señal con información del píxel activo RGB
HSYNC	Salida	Nivel	Señal de Control de sincronismo Horizontal
VSYNC	Salida	Nivel	Senal de Control de sicronismo Vertical

2.1.5.2 Descripción del Hardware, Diseño e Implementación

En el diseño del controlador se considera el uso de un monitor con una frecuencia de barrido vertical de 60 Hz (1 pantalla cada 16.6 ms) y una frecuencia de barrido horizontal de 31.5 KHz con una resolución de 640 x 480 pixeles, especificaciones estándar de la norma VGA. La frecuencia de barrido horizontal acorde a estos parámetros se muestra en la Figura 2.18 , en la que se aprecia que el trazo de una línea está definido por la señal de *sincronismo horizontal (HSYNC)*, para la que es necesario considerar que el barrido de una línea debe durar 31.77 µS; de los que 25.17 us corresponden a la ventana de tiempo durante la que es posible desplegar los pixeles que se visualizarán en esa línea, seguido de 6.6uS que corresponde al tiempo mínimo considerado para un *intervalo blanco horizontal*, en el cual se envían puntos negros (R,G,B)=(0,0,0), así también, durante este intervalo se debe establecer la señal de sincronismo horizontal en nivel lógico bajo al menos por 3.77 uS.

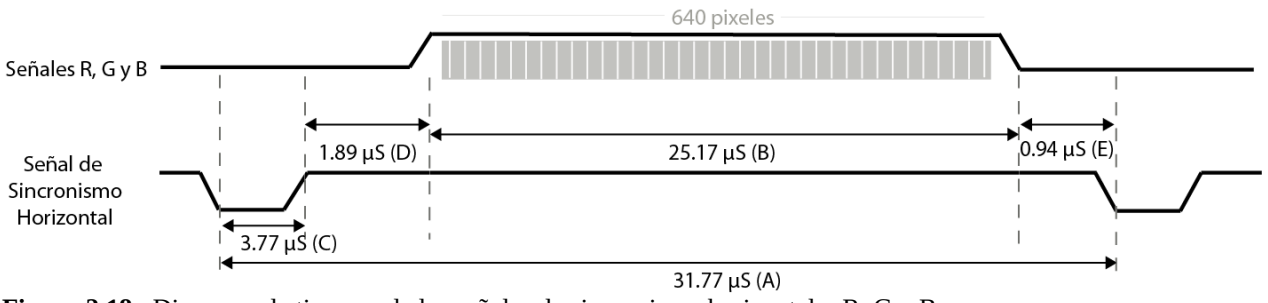


Figura 2.18. Diagrama de tiempos de las señales de sincronismo horizontal y R, G y B.

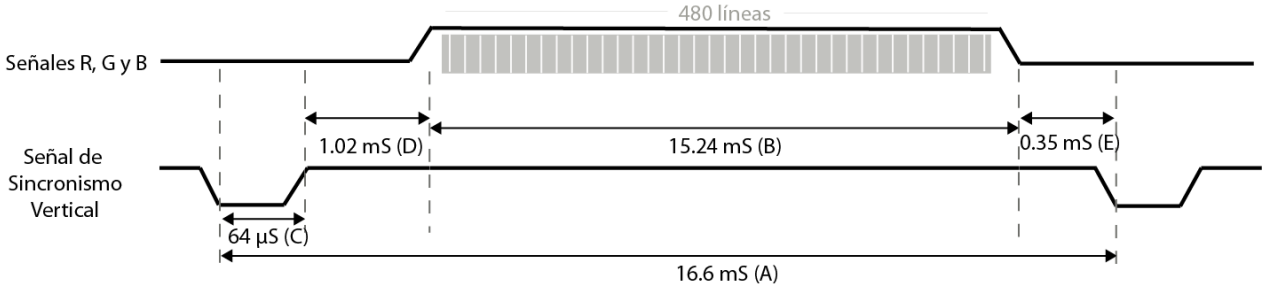


Figura 2.19. Diagrama de tiempos de las señales de sincronismo vertical y R, G y B.

En la Figura 2.19 se observa que el trazo o barrido de una imagen completa está definido por la señal de *sincronismo vertical (VSYNC)*, y esta contempla 16.784 ms, de los cuales 15.25 ms corresponden a la ventana durante la que se deben desplegar los pixeles de la imagen que se visualizará, y como mínimo 1.534 ms se debe aplicar un intervalo de blanco vertical, en el cual se envían puntos negros (R,G,B)=(0,0,0); así también, durante este intervalo se debe establecer la señal de sincronismo vertical en nivel lógico bajo al menos durante 0.064 ms.

Para dibujar la imagen se cuenta con las señales R, G yB que actúan para durante el tiempo que el cañón de electrones realiza su recorrido por el área visible de la pantalla, por lo que para mostrar 640 pixeles de manera horizontal, se dispone de 39.32 nS para cada punto, lo que corresponde a efectos prácticos de 2 ciclos de reloj.

Con base en las consideraciones de tiempo, se trasladan los tiempos establecidos para los impulsos de sincronismo horizontal y vertical a ciclos del reloj de entrada (CLK), ya que el diseño queda estrechamente ligado a la frecuencia del reloj con que opera el controlador, arrojando los tiempos que se muestran en la Tabla 2.20.

Tabla 2.20. Tiempos de operación y ciclos de reloj de las señales de sincronismo.

	Sincronismo Horizontal		Sincronismo Vertical		
Símbolo	**Tiempo (µS)**	**Ciclos Reloj**	**Tiempo (mS)**	**Ciclos Reloj**	**Líneas**
A	32	800	16.7	416,800	521
B	25.6	640	15.36	384,000	480
C	3.84	96	64	1,600	2
D	0.64	16	320	8,000	10
E	1.92	48	928	23,200	29

A partir de las especificaciones para este módulo, funcionalmente debe contener los elementos que se muestran en el diagrama a bloques de la Figura 2.20.

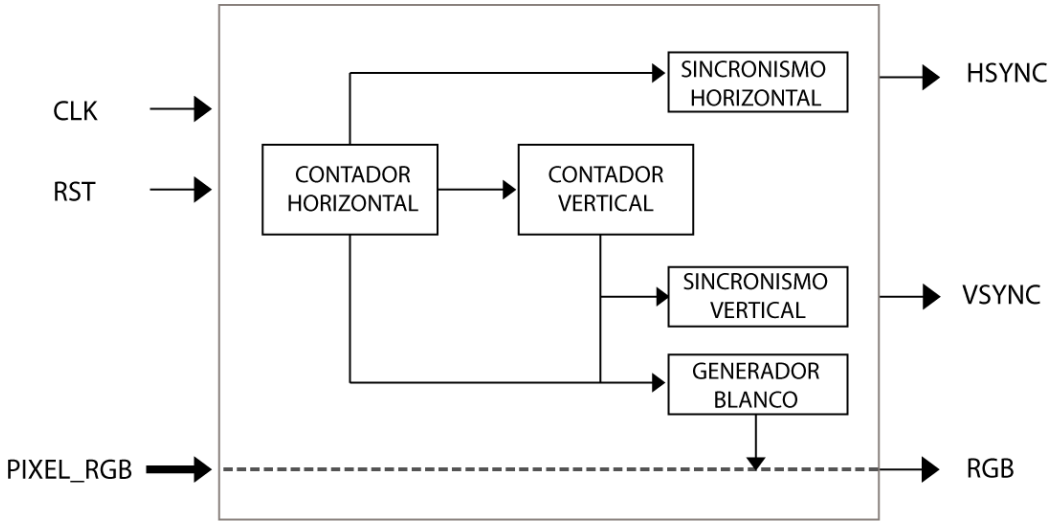


Figura 2.20. Diagrama a bloques de la descripción funcional del módulo Controlador VGA.

El *Contador Horizontal*, corresponde a un contador de flancos de reloj principal (CLK) y proporciona, mediante la métrica numérica, la información necesaria para generar los impulsos de sincronismo horizontal y la información de los pixeles.

El *Contador Vertical*, es la parte encargada de contar los flancos de la señal de sincronismo horizontal (HSYNC), proporcionando la información necesaria para generar los impulsos de sincronismo vertical y el conteo de las líneas que forman la imagen.

El *Sincronismo Horizontal*, consiste en un generador de impulsos de sincronismo horizontal, mediante la señal HSYNC se indica el cambio de la señal de sincronismo horizontal, respondiendo al estado del contador horizontal.

El *Sincronismo Vertical*, consiste en un generador de impulsos de sincronismo vertical, mediante la señal VSYNC se indica el cambio de la señal de sincronismo vertical, de acuerdo al estado del contador vertical.

El *Generador de Blancos*, con la señal de salida BLANK indica, mediante un nivel lógico bajo, que la señal de color PIXEL_RGB[0:3] debe pasar a un nivel lógico bajo para crear un intervalo blanco.

En la implementación del *contador Horizontal* se parte de que 32 uS equivalen a 800 ciclos de reloj, por lo que se emplea un contador de 800 estados, de tal manera que en el ciclo 800 se reinicia a 0 para volver a comenzar la cuenta de la siguiente línea horizontal.

El contador vertical de 16.7mS corresponde a un contador de 521 líneas horizontales, cuenta cada que el contador horizontal reinicia su cuenta; estableciendo así, la cuenta para una pantalla nueva. La descripción de ambos contadores se especifica en la Tabla 2.21.

Tabla 2.21. Descripción VHDL del contador horizontal y vertical.

```
1  Constant HORIZONTAL: STD_LOGIC_VECTOR (9 downto 0) := "1100100000";  --800
2  Constant VERTICAL: STD_LOGIC_VECTOR (9 downto 0) := "1000001001";    --521
3  if (CLK = '1' and CLK'event) then
4    if (HC = HORIZONTAL) then
5        HC <= "0000000000";
6    if(VC = VERTICAL) then
7        VC <= "0000000000";
8    else
9        VC <= VC + 1;
10   end if;
11 else
12   HC <= HC + 1;
13 end if;
```

Los *generadores de sincronismo vertical y horizontal*, y *generador de blancos* se realizan mediante lógica combinacional, coincidiendo con los tiempos de reloj obtenidos de los tiempos característicos mostrados en la Tabla 2.20. El comportamiento que estas señales presentan se rige por las ecuaciones que determinan las sentencias de la Tabla 2.22.

Tabla 2.22. Descripción de las sentencias generadores de sincronismo y blanqueo.

```
1  HSYNC <= '0' when (HC > 0 and HC <= 96 ) else '1';
2  VSYNC <= '0' when (VC > 0 and VC <= 2) else '1';
3  BLANK <= '1' when (HC >= 144 and HC < 784 and VC >= 39 and VC < 519)
              else '0';
```

La información de las señales RGB que forman los pixeles que se representa en el monitor, es desplegada durante los intervalos en los que el haz se encuentra recorriendo el área visible de la pantalla, y deben cambiar a 0 durante el intervalo de blanqueo; lo que se aprecia en la Tabla 2.23.

.

Tabla 2.23. Sentencia de asignación del pixel a desplegar.

```
1  RGB<= PIXEL_RGB when (BLANK = '1')  else "000";
```

Así, se genera un contador horizontal y vertical que permite conocer la posición precisa del pixel que debe representarse en la pantalla dentro del área visible de (640 horizontal y 480 vertical), tal como muestra la descripción de la Tabla 2.24.

Tabla 2.24. Contadores Horizontales y Verticales de posición en pantalla activa.

```
1  if (CLK = '1' and CLK'event and BLANK = '1')then
2      PIXHOR <= PIXHOR + 1;
3     if (PIXHOR = 639) then
4        PIXHOR <= (others=>'0');
5        PIXVER <= PIXVER + 1;
6       if (PIXVER = 479) then
7          PIXVER <= (others=>'0');
8       end if;
9     end if;
10 end if;
```

2.1.6 Componente Control de Video

El componente control de video tiene como propósito generar el control de los componentes memoria NVRAM, memoria de video, VGA, cursor e interfaz PS2; con la finalidad de establecer las secuencias que permitan obtener la información de la memoria NVRAM, que contiene imágenes descompuestas en sus elementos básicos, y copiar la información de la imagen activa a la memoria de video, que por su alta tasa de refresco, proporciona la información que debe desplegar el Controlador VGA. Integrando la interfaz Mouse PS2, que establece la interfaz con el usuario, para permitir con la presión del botón derecho del ratón el retroceso de la imagen o el avance con la presión del botón izquierdo, así como el despliegue y redibujo del desplazamiento del puntero sobre la imagen activa.

2.1.6.1 Descripción funcional

En la Figura 2.21 se muestra el símbolo, equivalente a la descripción de hardware en el diagrama esquemático, del componente de Control de Video.

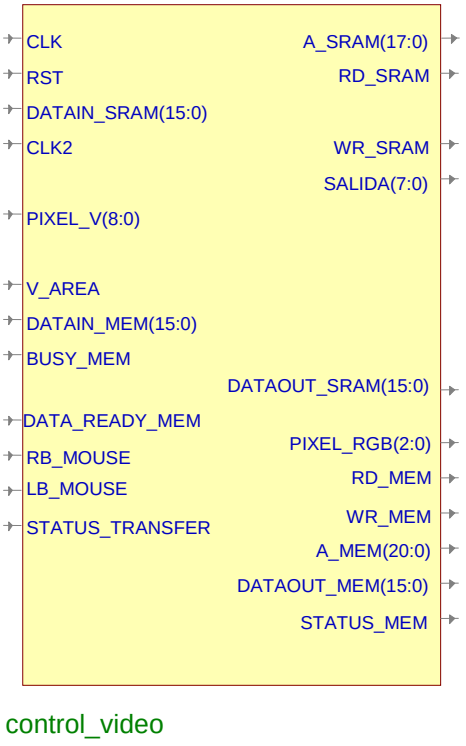


Figura 2.21. Símbolo con la descripción de hardware del componente Control de Video.

La Tabla 2.25 describe la funcionalidad de cada uno de los puertos que se integran en el componente Control de video.

Tabla 2.25. Descripción de Pines del componente Control de Video.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
Señales de Información y Control Interfaz Video			
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
CLK2	Entrada	Flanco	Reloj de sistema externo de 25 Mhz
RST	Entrada	Alto	Reset de sistema externo
Señales de control y datos de memoria de video			
A_SRAM	Salida	17:0	Bus de dirección memoria de video
DATAIN_SRAM	Entrada	15:0	Bus de datos leídos memoria de video
DATAOUT_SRAM	Salida	15:0	Bus de datos a escribir memoria de video
RD_SRAM	Salida	Alto	Señal de control Lectura
WR_SRAM	Salida	Alto	Señal de control Escritura
Señales de control de video			
PIXEL_V	Entrada	8:0	Bus de la línea vertical desplegándose
PIXEL_RGB	Salida	2:0	Bus de información de pixel a desplegar
V_AREA	Entrada	Alto	Señal indicadora de blanqueo de video
Señales de control de memoria NVRAM			
STATUS_MEM	Salida	Alto	Señal de escritura activa en la memoria
RD_MEM	Salida	Alto	Señal de control de solicitud de lectura
WR_MEM	Salida	Alto	Señal de control de solicitud de escritura
A_MEM	Salida	20:0	Bus de dirección
DATAIN_MEM	Entrada	15:0	Bus de datos leídos
DATAOUT_MEM	Salida	15:0	Bus de datos escritos
BUSY_MEM	Entrada	Alto	Señal de control de memoria ocupada
DATA_READY_MEM	Entrada	Alto	Señal de control de dato disponible
Señales de estado del ratón			
RB_MOUSE	Entrada	Alto	Señal de estado de botón derecho
LB_MOUSE	Entrada	Alto	Señal de estado de botón izquierdo
Señales de monitoreo			
SALIDA	Salida	7:0	Bus de visualización de estados

2.1.6.2 Descripción de hardware, diseño e Implementación

En el proceso de diseño del componente de control, es necesario especificar los parámetros de representación de la información con la que opera el sistema, tanto en pantalla como en memoria. Considerando que para representar una imagen de 640 x 480 pixeles, con una profundidad de color de 3bits, son necesarios 921,600 bits; estos datos requieren ser organizados de manera que permitan, sin un algoritmo complejo, simplificar la interpretación por los procesos involucrados en el controlador de video, por ello, considerando que la longitud de palabra de la memoria de video es de 16 bits, se asignan los 15 primero bits para representar 5 pixeles consecutivos, tomando la palabra de la siguiente localidad de memoria para los siguientes 5 pixeles; con este manejo se hacen necesarias 61,440 palabras de 16 bits para representar una imagen completa, lo que se representa en la Figura 2.22.

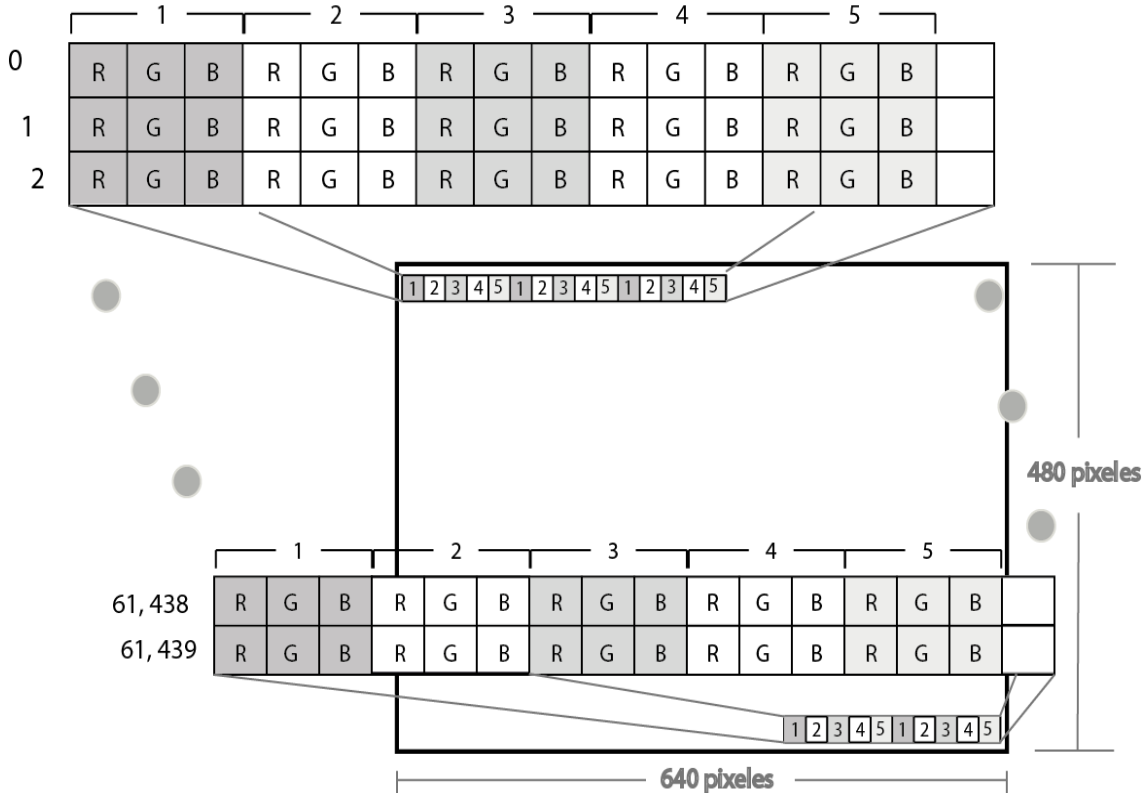


Figura 2.22. Representación de la información en pantalla.

La segmentación del almacenamiento de la memoria NVRAM, con fines de ordenamiento y manejo, se determina estableciendo 65,535 direcciones para cada página, Figura 2.23, con la subsecuente subutilización de espacio de almacenamiento, pero con la clara ventaja de establecer un control de paginación con los restantes bits de dirección, transparentando la manipulación de la información por el controlador de video y controlador serie, que realizan accesos al banco de imágenes almacenadas en la memoria; adicionalmente, esto permite la implementación de memorias DS1265 ó DS1270, estas últimas con el doble de capacidad de almacenamiento, ampliando las posibilidades de utilizar al componente en otras aplicaciones.

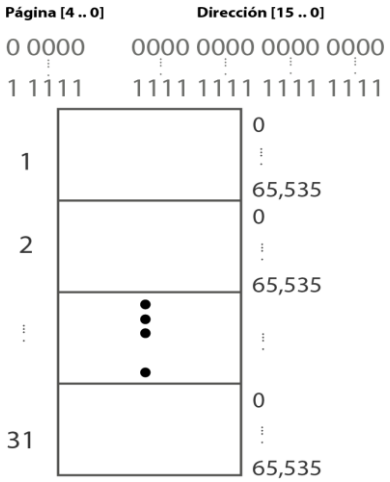


Figura 2.23. Segmentación de la memoria NVRAM en páginas.

La información que se procesará por la interfaz de video se encuentra almacenada en la memoria de video NVRAM, agrupada en un arreglo de 61,440 direcciones por imagen, considerando 65,535 direcciones por página, tal como se muestra en la Figura 2.23.

Para obtener la imagen activa ó imagen a desplegar, se utiliza la variable *counter_pag*; en tanto la variable *counter*, funcionará como indicador de la localidad en la que se almacena la información de esa imagen.

La información de la imagen activa se lee localidad a localidad y se escribe en la memoria de video, que observa el mismo patrón de almacenamiento que la memoria NVRAM, pero para una sola página ó 65,535 palabras. Para fines prácticos, el final del copiado de la información de la imagen actual se establece en 61,440 palabras, que corresponde a la información efectiva. La Tabla 2.26 muestra la descripción, que forma parte de una máquina de estados, que permite copiar la imagen activa en la memoria de video.

Tabla 2.26. Descripción de la secuencia de copiado de la imagen activa en la memoria de video.

```
1   when UNO =>
2     Addrs <= counter_pag & counter;    -- Asigna la dirección a leer
3     read_memory <= '1';
4     STATUS_MEM <= '1';                 -- Bandera de operación en NVRAM
5   when UNO_B =>
6     read_memory <= '0';
7     if (flag_nvram = '0') then         -- Fin de Lectura ?
8        control_state <= UNO_B;
9     else
10       control_state <= UNO_C;
11    end if;
12  when UNO_C =>
13    A_SRAM <= "00" & counter;
14    DATAOUT_SRAM <= rDByte;             -- Escritura del Dato leído
15    WR_SRAM <= '1';
16    control_state <= DOS;
17  when DOS =>
18    WR_SRAM <= '0';
19    counter <= counter  + 1;
20    control_state <= TRES;
21  when TRES =>
22    if (counter < 61440) then
23       control_state <= UNO;
24    else
25       control_state <= CUATRO;
26    end if;
```

La información que es desplegada por el componente VGA, se lee de la memoria de video, donde cada solicitud de lectura es establecida mediante la activación del registro *read_ready*; la ubicación de los pixeles a desplegar; puede determinarse orgánicamente al considerar que, la longitud de una línea horizontal requiere 640 píxeles y una palabra en la memoria de video representa 5 pixeles, lo que equivale a 128 palabras por línea horizontal; así, un contador de 7 bits que incremente por cada palabra desplegada, *counter2*, y el contador de líneas verticales, *PIXEL_V*, del componente VGA, determinarán la localidad de la palabra actual en la memoria de video. La descripción en VHDL se aprecia en la Tabla 2.27.

Tabla 2.27. Descripción de la asignación de localidad a leer en la memoria de video.

```
1  if (read_ready = '1') then
2     A_SRAM(6 downto 0) <= counter2;
3     A_SRAM (15 downto 7) <= PIXEL_V(8 downto 0);
4     RD_SRAM <= '1';
5  end if;
```

La memoria de video requiere de un proceso que establezca el ciclo de lectura y presente el pixel a desplegar al controlador VGA, para que opere propiamente como memoria de refresco de video. Este proceso, del que se muestra la descripción en la Tabla 2.28, comparte la señal de reloj de 25 Mhz con el componente VGA para sincronizar la asignación de los pixeles; y emplea como referencia, para sincronizar las secuencias de operación, la señal de blanqueo, *V_AREA*, reiniciando el contadores de pixeles, *counter3*, y el contador de palabras, *counter2*. Los pixeles se despliegan mediante un registro de corrimiento, *data_buff*.

Tabla 2.28. Proceso control de lectura en memoria de video y despliegue de pixeles.

```
1   process (RST, CLK2)
2     begin
3        if (RST ='1') then
4              stand <= '0';
5              counter2 <= (others=>'0');
6              counter3 <= (others=>'0');
7        elsif (CLK2 = '1' and CLK2'Event) then
8              if ( V_AREA = '0') then            --PERIODO DE BLANQUEO
9              counter2 <= (others=>'0');
10             counter3 <= (others=>'0');
11                   if (stand = '0') then
12                         stand  <= '1';
13                         read_ready <= '1';
14                   Else
15                         read_ready <= '0';
16                   end if;
17             else
18                   stand <= '0';
19                   counter3 <= counter3 + 1;
20                   if (counter3 = 0) then
21                         read_ready <= '1';
22                         data_buff <= DATAIN_SRAM ;
23                   elsif (counter3 = 4 ) then
24                         counter3 <= (others=>'0');
25                         counter2 <= counter2 + 1;
26                   else
27                         data_buff <= "000" & data_buff (15 downto 3);
28                         read_ready <= '0';
29                   end if;
30             end if;
31       end if;
32 end process;
```

El ratón controla la visualización de la página activa, mediante el contador de página *counter_pag*, ya que si se presiona el botón derecho, la cuenta será positiva, adelantando la paginación de imagenes; si el que se presiona es el botón izquierdo, la cuenta es negativa, retrocediendo en uno la paginación.

Operacionalmente se emplea una bandera que permite conocer el estado de los botones, *flag_pressed*. Si uno de los botones se encuentra presionado, el componente seguirá refrescando

la pantalla, realizando el cambio de imagen hasta que el botón sea liberado, ya que en otro caso se forzaría a visualizar el retardo inherente a la lectura, escritura y despliegue de la imagen, en una pantalla negra. Otra consideración importante es que el cambio se realice en tanto la memoria no se encuentre realizando algún proceso de acceso a memoria, indicado por *BUSY_MEM*, o todavía aún más importante, algún ciclo de escritura, indicado por la señal de entrada *STATUS_TRANSFER* que se encuentra ligada directamente al proceso de control serie. La descripción de la acción de las señales de estado de los botones del ratón se muestra en la Tabla 2.29.

Tabla 2.29. Descripción de la acción de los botones de ratón.

```
1   if (LB_MOUSE = '1' and BUSY_MEM = '0') then
2      if(STATUS_TRANSFER = '0') then
3        if (flag_pressed = '0' and STATUS_TRANSFER = '0') then
4           counter_pag <= counter_pag + 1;
5        end if;
6        flag_pressed <= '1';
7      end if;
8   elsif (RB_MOUSE = '1' and BUSY_MEM = '0') then
9       if(STATUS_TRANSFER = '0') then
10        if (flag_pressed = '0') then
11           counter_pag <= counter_pag - 1;
12        end if;
13      flag_pressed <= '1';
14  end if;
```

2.1.7 Integración del Módulo Interfaz de Video

El módulo de Interfaz de Video, de acuerdo a la metodología jerárquica empleada, corresponde a un bloque de jerarquía superior, el cual se genera de la instanciación de los componentes:

- Divisor de Frecuencia
- Memoria de Video SRAM
- Interfaz Mouse PS/2
- Cursor
- Controlador VGA
- Controlador de Video

Teniendo en cuenta que la descripción en VHDL se desarrolla con el estilo estructural, la integración de instancias, como se muestra en el esquemático de la Figura 2.24, dota al módulo de la funcionalidad especificada en el proceso de diseño a partir del comportamiento de conjunto de cada uno de los componentes que se han detallado hasta el momento.

En el esquemático se observa que el componente *Divisor de frecuencia*, opera como reloj principal en el componente *Controlador VGA* y reloj secundario en el componente *Controlador de video*, lo que le permite sincronizar los datos que se obtienen de la *Memoria de video SRAM* por el proceso interno de control de lectura de video y despliegue de pixeles.

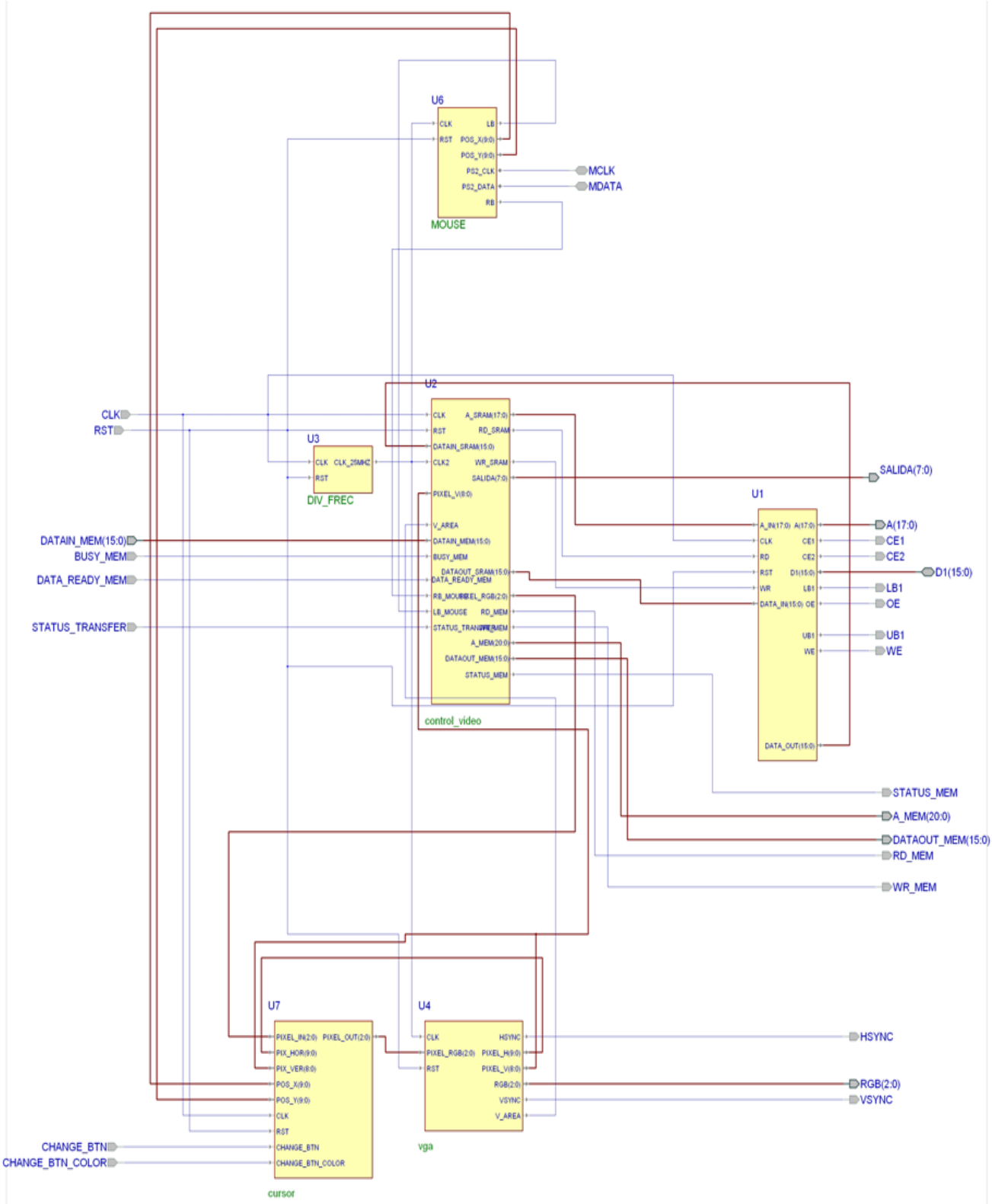


Figura 2.24. Esquemático de la conexión de los componentes en el Módulo Interfaz de Video.

El componente *Interfaz mouse PS/2* está directamente ligado los puertos triestado *MCLK* y *MDATA*, que proporcionan la comunicación bidireccional con el ratón; los detalles de la posición del cursor, *POS_X* y *POS_Y*, se conectan al componente *Cursor*, que compara la posición actual que se reporta en el ratón con la posición que se encuentra representando el componente

Controlador VGA y genera así la forma del cursor seleccionado a partir de esta posición. El evento de botón de mouse, *RB* y *LB*, se establece como señal indicadora para el componente *Control de video*, que permite establecer el proceso de solicitud de datos y ciclo de escritura en el componente *Memoria de Video SRAM*.

El componente *Controlador VGA*, obtiene el pixel a representar, *PIXEL_RGB*, directamente del componente *Cursor*; retroalimentado con la posición horizontal y vertical, *PIXEL_H* y *PIXEL_V*. En tanto la señal de control *V_AREA* se conecta con el componente *Controlador de video*, para indicar los estados de blanqueo y con esto proporcionar la ventana de tiempo seguro para operaciones sobre la memoria de video.

La Tabla 2.30 se describe la funcionalidad de cada uno de los puertos del módulo resultante,

Tabla 2.30. Descripción de Pines del módulo Interfaz de Video.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de interfaz y Control SRAM	
A	Salida	17:0	Bus de dirección de memoria
D1	Salida	15:0	Bus de datos de memoria
CE1	Salida	Alta	Señal de habilitación memoria 1
CE2	Salida	Alta	Señal de habilitación memoria 2
LB1	Salida	Alta	Acceso Byte de Datos menos significativo
UB1	Salida	Alta	Acceso Byte de Datos más significativo
OE	Salida	Alta	Señal de Salida Habilitada (Output Enable)
WE	Salida	Alta	Señal de habilitación de Escritura
		Señales de Interfaz VGA	
RGB	Salida	0:3	Señal con información del píxel activo RGB
HSYNC	Salida	Nivel	Señal de Control de sincronismo Horizontal
VSYNC	Salida	Nivel	Senal de Control de sicronismo Vertical
		Señales de Interfaz Mouse PS/2	
MCLK	Bidireccional	Triestado	Señal de Reloj de sincronización
MDATA	Bidireccional	Triestado	Señal de Datos
		Señales de Interfaz Botones de Presión	
BTN_CURSOR	Entrada	Alto	Señal Selección de Cursor
CHANGE_BTN_COLOR	Entrada	Alto	Señal de inversión de bit de Color
		Señales de Interfaz UART	
TX	Salida	Alto	Señal Transmisión de Datos
RX	Entrada	Alto	Señal Recepción de Datos
		Señales Badera	
STATUS_TRANSFER	Entrada	Alto	Señal de control petición acceso NVRAM
		Señales Interfaz NVRAM	
A_MEM	Salida	20:0	Bus de Dirección
DATAIN_MEM	Entrada	15:0	Bus de Datos Leídos
DATAOUT_MEM	Salida	15:0	Bus de Datos a Escribir
RD_MEM	Salida	Alto	Señal de operación de Lectura
WR_MEM	Salida	Alto	Señal de operación de Escritura
STATUS_MEM	Salida	Alto	Señal de Petición del recurso
BUSY_MEM	Entrada	Alto	Señal de memoria ocupada
DATA_READY_MEM	Entrada	Alto	Señal de bus de datos disponible

La Figura 2.25 muestra el símbolo que representa al módulo Interfaz de video con las entradas y salidas del bloque.

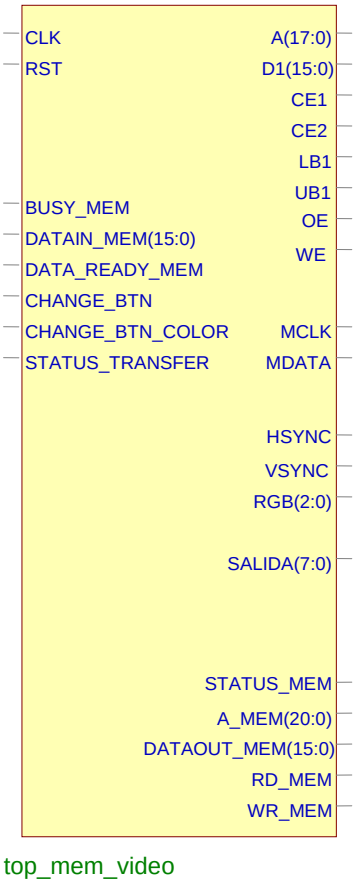


Figura 2.25. Símbolo Descripción de hardware del módulo Interfaz de Video.

2.2 Módulo interfaz Serie

Este módulo contiene los componentes de la interfaz de comunicación serial asíncrona y es el encargado de establecer la comunicación a través de la interfaz de hardware RS232C, permitiendo intercambiar datos binarios entre el FPGA y otro dispositivo a través de una conexión punto a punto.

2.2.1 Componente UART

Este componente provee una interfaz programable de comunicación serie asíncrona. El diseño está orientado a emular de forma parcial el funcionamiento de los dispositivos de recepción-transmisión asincrónica universal, UART, con una PILA (FIFO, *First Input First Output*) de datos, que permite convertir los datos seriales a paralelos; y debe ofrecer las siguientes características de operación:

- Frecuencia de operación: 50 MHz – 25 MHz
- Velocidad de transferencias configurables : 9600 bps a 115.2 Kbps
- Conversión de datos Seriales a Paralelo (Recepción)
- Conversión de datos Paralelos a Seriales (Transmisión)
- Control de errores y paridad

2.2.1.1 Descripción funcional

En la Figura 2.26 muestra el símbolo, equivalente a la descripción de hardware en el diagrama esquemático, del componente UART.

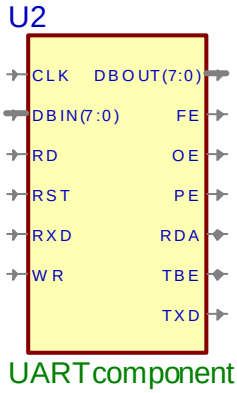


Figura 2.26. Símbolo con la descripción del hardware del Componente UART.

En la Tabla 2.31 se describe la funcionalidad de cada uno de los puertos que se integran en el componente UART.

Tabla 2.31. Descripción de Pines componente UART.

Puerto	Dirección Puerto	Polarida/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de Sistema externo asíncrono
		Transmisión Serial	
TXD	Salida		Señal para la Transmisión de Datos
RXD	Entrada		Señal para la Recepción de Datos
		Buses de Datos	
DBIN	Entrada	7:0	Bus de datos de entrada
DBOUT	Salida	7:0	Bus de datos de salida
		Señales de Control Serial	
RDA	Bidireccional	Alto	Señal de dato transmitido disponible
TBE	Bidireccional	Bajo	Señal de transferencia
RD	Entrada	Alto	Señal de control de lectura
WR	Entrada	Alto	Señal de control de escritura
		Señales bandera	
PE	Salida	Alto	Señal de bandera de Error de Paridad
FE	Salida	Alto	Señal de bandera de Error de Trama
OE	Salida	Alto	Señal de Error de SobreEscritura

2.2.1.2 Descripción del Hardware e Implementación

En el caso de este componente y dadas las especificaciones requeridas, se emplea la implementación del *componente de referencia RS232* de libre distribución proporcionado por la empresa Digilent Inc. [URL9] y desarrollado por Dan Pederson, con revisión de 25 de Julio de 2008; el cuál ha sido comprobado en la práctica y ofrece un rendimiento de recepción y transmisión confiable.

En el diagrama de la Figura 2.27 se observan los bloques que proporcionan la funcionalidad del componente UART.

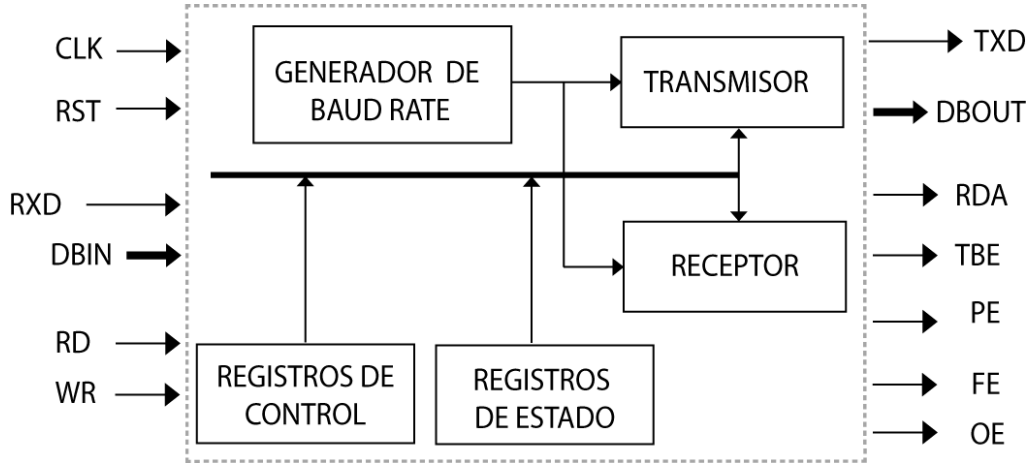


Figura 2.27. Diagrama a bloques de la descripción funcional del componente UART.

La configuración de la velocidad de transmisión para la implementación, se realiza en el bloque Generador de Baud Rate, mediante una constante, *baudDivide*, la cual se obtiene al dividir la frecuencia del reloj principal (CLK) –en este caso 50 MHz- entre la frecuencia de transmisión deseada y entre un ciclo de operación seguro de 16 ciclos; con lo que se configura la velocidad de operación del componente de acuerdo al parámetro resultante de la división anterior, en la Tabla 2.32 se muestra el valor de *baudDivide* para las diferentes frecuencias de transmisión.

Tabla 2.32. Parámetro de Configuración para la velocidad de transmisión

Constante *baudDivide*	**BaudRate**
1010 0011	9600
0101 0001	19200
0010 1001	38400
0001 1010	57600
0000 1101	115200

2.2.2 Componente Control Serie

El componente de Control serial tiene como finalidad integrar el componente serial e interactuar con el componente NVRAM, para establecer las secuencias que permitan recibir datos del puerto serial y almacenar esta información en la memoria, segmentando el total de la memoria en páginas de 61,440 direcciones ó 983,040 bytes.

2.2.2.1 Descripción funcional

En la Figura 2.28 se muestra el símbolo equivalente a la descripción de hardware en el diagrama esquemático, del componente de Control Serie.

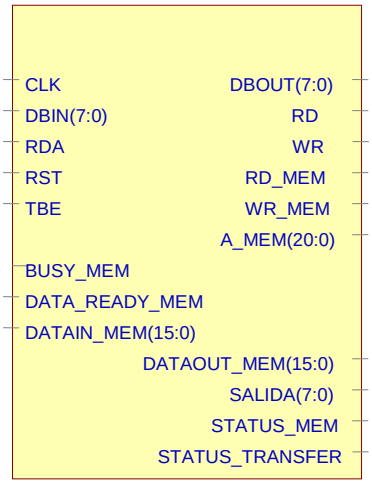


Figura 2.28. Símbolo Descripción del hardware del componente Control Serie.

La Tabla 2.33 describe la funcionalidad de cada uno de los puertos que se integran en el componente Control Serie.

Tabla 2.33. Descripción de Pines del componente Control Serie.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de Sistema externo asíncrono
		Señales de Control UART	
RD	Salida	Alto	Señal de control de lectura UART
WR	Salida	Alto	Señal de control de escritura UART
RDA	Entrada	Alto	Señal de dato transmitido disponible
TBE	Entrada	Bajo	Señal de transferencia de UART
		Buses de Datos UART	
DBOUT	Salida	7:0	Bus de datos de salida
DBIN	Entrada	7:0	Bus de datos de entrada
		Señales registro de petición NVRAM	
RD_MEM	Salida		Señal de control solicitando Lectura
WR_MEM	Salida		Señal de control solcitando Escritura
		Señales de control y bus NVRAM	
A_MEM	Salida	20:0	Bus de dirección
DATAIN_MEM	Entrada	15:0	Bus de datos leídos
DATAOUT_MEM	Salida	15:0	Bus de datos a escribir
BUSY_MEM	Entrada	Alto	Señal de memoria ocupada
DATA_READY_MEM	Entrada	Alto	Señal de dato disponible
		Señales bandera y monitoreo	
STATUS_TRANSFER	Salida	Alto	Señal de registro memoria ocupada
SALIDA	Salida	7:0	Señal de monitoreo de estados

2.2.2.2 Descripción del Hardware, Diseño e Implementación

El proceso que se establece mediante este componente, se describe operacionalmente de acuerdo al diagrama de flujo de la Figura 2.29.

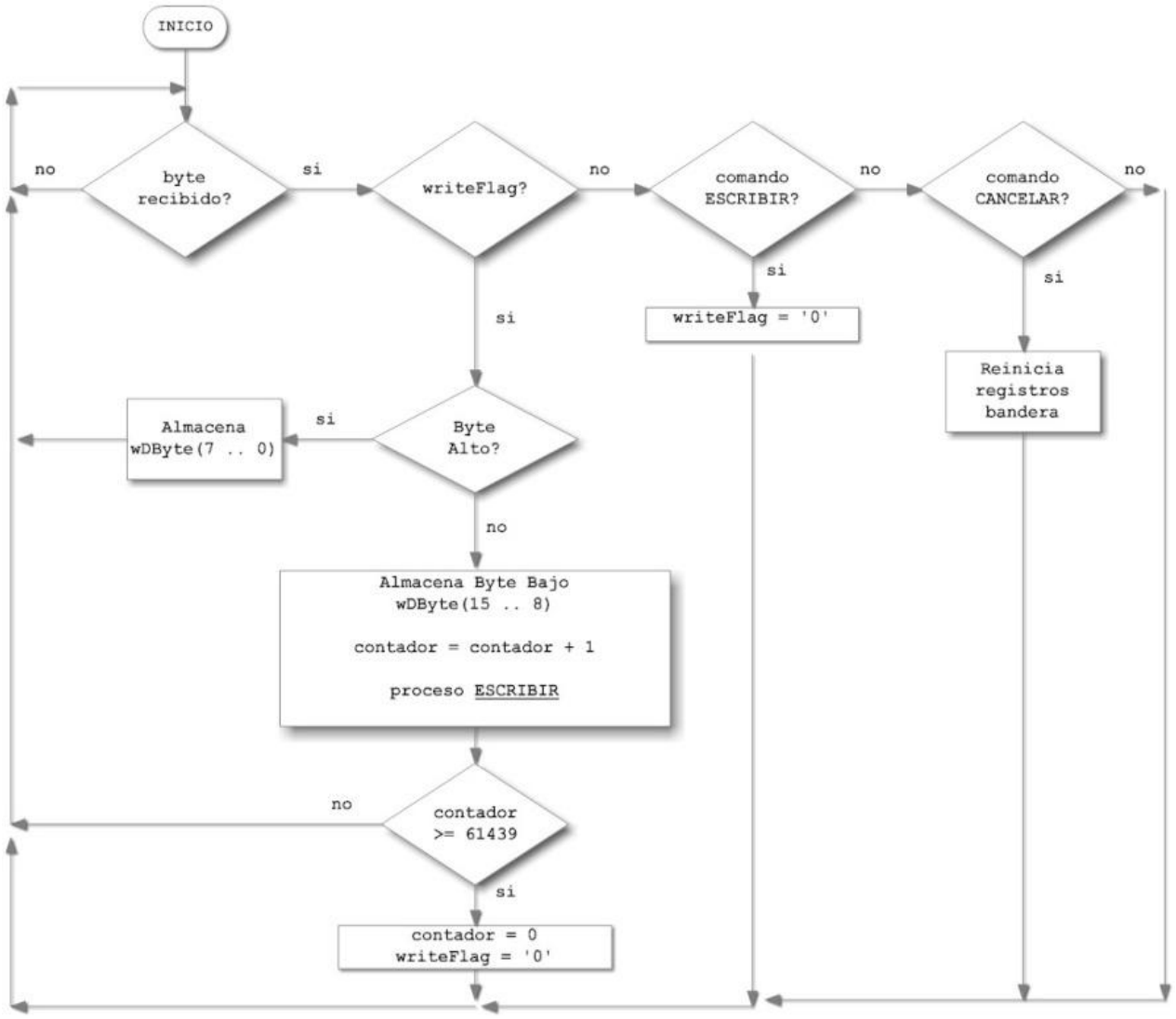


Figura 2.29. Proceso de recepción de datos seriales y almacenamiento paginado en NVRAM.

El primer de byte de información que debe recibir el controlador en modo de espera corresponde al comando de la operación requerida, comprendido por la estructura de la Figura 2.30.

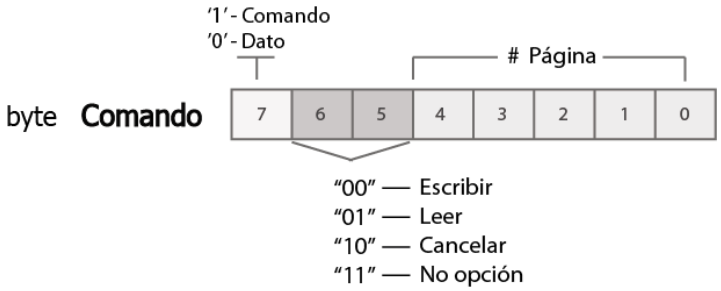


Figura 2.30. Estructura del Byte que indica un comando de Operación.

Si el comando que se recibe corresponde a una solicitud de escritura, se activa la bandera indicando que el proceso se debe ciclar en la recepción de datos hasta completar una cuenta de 61,440 bytes, que corresponde a una página; este proceso, en términos de operación, considera al módulo NVRAM como una sola memoria con longitud de palabra de 16 bits, por ello la escritura se realiza hasta completar la parte alta y baja del dato, cada 2 recepciones. La descripción de la Tabla 2.34 forma parte de la máquina de estados que se ejecuta en el proceso de escritura.

Tabla 2.34. Descripción de Proceso de Control de Escritura a la NVRAM.

```
1   Addrs (15 downto 0) <= counter;      //Dirección - contador
2   selector <= not selector;            //selector byte alto y bajo
3   if (selector = '1') then             //si palabra completa
4      wDByte (7 downto 0) <= sByte;
5      stNext <= stWriteUno;
6   else
8      wDByte(15 downto 8) <= sByte;
9      stNext <= stUno;                    //Estado inicial
10  end if;

11  when stWriteUno =>
13     write_memory <= '1';
14     stNext <=  stWriteDos;
15  when stWriteDos =>
16     write_memory <= '0';
17     stNext <= stWriteTres;
18  when stWriteTres =>
19     stNext <= stUno;
20     if (counter >= 61439) then          //Cuenta de Página
21         writeFlag <= '0';               //Bandera escritura Desactivada
22         counter <= (others => '0');
23     else
24         counter <= counter + 1;         // Incremento de contador
25     end if;
```

2.2.2 Integración del Modulo Interfaz Serie

El módulo de Interfaz Serie, de acuerdo a la metodología jerárquica empleada, corresponde a un bloque de jerarquía superior, el cual es el resultado de la instanciación de los componentes:

- Componente UART
- Componente Control Serie

Teniendo en cuenta que la descripción en VHDL se desarrolla con el estilo estructural, la integración de instancias, como se muestra en el esquemático de la Figura 2.31, dota al módulo de la funcionalidad especificada en el proceso de diseño a partir del comportamiento de conjunto de cada uno de los componentes que se han detallado hasta el momento.

En el esquemático se observa que el componente *Control serie* establece la comunicación mediante las señales de control con el componente *UART*, para establecer el proceso de inicialización del ratón y a su vez, recibir la indicación de nuevo dato recibido, *RDA*. Así mismo, establece los puertos de conexión hacia el componente externo *módulo NVRAM*, ya que establecen las señales de control y los ciclos de escritura de los datos recibidos mediante el componente *UART*.

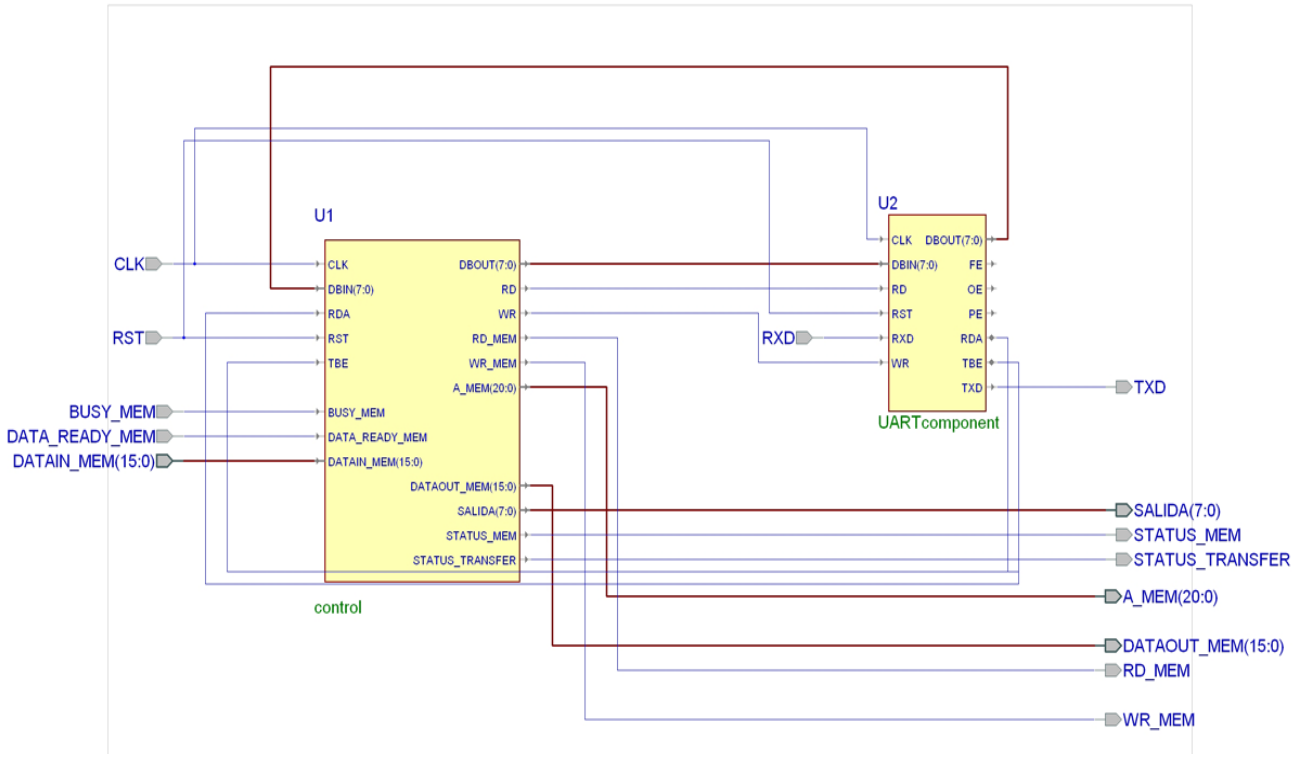


Figura 2.31. Esquemático de la conexión de los componentes en el Módulo Interfaz Serie.

La Figura 2.32 muestra el símbolo que representa al módulo Interfaz Serie con las entradas y salidas del bloque.

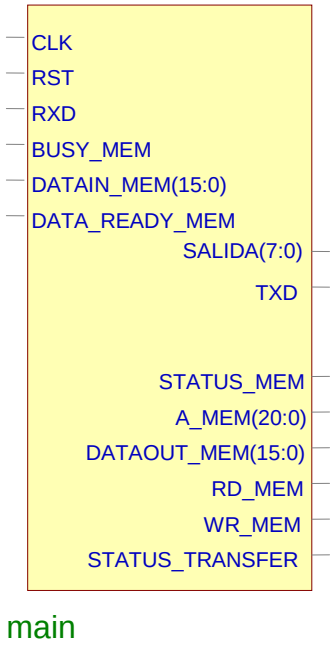


Figura 2.32. Símbolo Descripción de hardware del módulo Interfaz Serie.

La Tabla 2.35 describe la funcionalidad de cada uno de los puertos del módulo resultante,

Tabla 2.35. Descripción de Pines del módulo Interfaz Serie.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de Interfaz UART	
TX	Salida	Alto	Señal Transmisión de Datos
RX	Entrada	Alto	Señal Recepción de Datos
		Señales Badera	
STATUS_TRANSFER	Salida	Alto	Señal de control petición acceso NVRAM
		Señales Interfaz NVRAM	
A_MEM	Salida	20:0	Bus de Dirección
DATAIN_MEM	Entrada	15:0	Bus de Datos Leídos
DATAOUT_MEM	Salida	15:0	Bus de Datos a Escribir
RD_MEM	Salida	Alto	Señal de operación de Lectura
WR_MEM	Salida	Alto	Señal de operación de Escritura
STATUS_MEM	Salida	Alto	Señal de Petición del recurso
BUSY_MEM	Entrada	Alto	Señal de memoria ocupada
DATA_READY_MEM	Entrada	Alto	Señal de bus de datos disponible

2.3 Módulo para el Almacenamiento de Imágenes

Este módulo contiene los componentes para el acceso a la memoria no volátil externa a la tarjeta Spartan-3. Se integra de dos componentes, uno para la interfaz física y otro para el control, los cuales se integran directamente en el módulo de mayor jerarquía.

2.3.1 Componente NVRAM

El componente NVRAM provee la interfaz de acceso para la escritura y lectura en un arreglo de 2 memorias DALLAS NVRAM (*Non-volatile random Access memory*, Memoria de acceso aleatorio No Volátil) DS1265W de 8 Mb ó DS1270W de 16 Mb, un tipo de memorias que no pierde la información almacenada al cortar la alimentación eléctrica y por sus características será empleada como memoria de almacenamiento de imágenes de 16 Mb ó 32 Mb de capacidad.

El componente se incorpora de manera física, mediante los conectores de expansión, a la tarjeta de desarrollo Digilent Spartan-3; basado en las especificaciones opera bajo las siguientes características:

- Frecuencia de operación: 50 Mhz
- Síncrono
- Almacenamiento de 16 Mb ó 32 Mb
- Operación de memorias independiente

2.3.1.1 Descripción Funcional

En la Figura 2.33 se muestra el símbolo equivalente a la descripción de hardware en el diagrama esquemático del componente NVRAM.

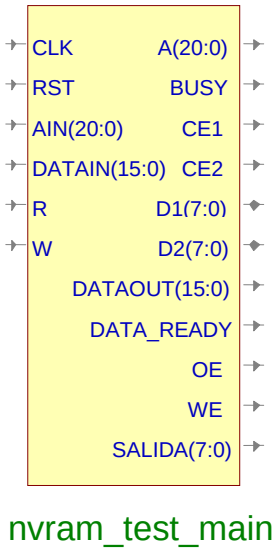


Figura 2.33. Símbolo con la descripción del hardware del componente NVRAM.

En la Tabla 2.36 se describe la funcionalidad de cada uno de los puertos de los que se compone el componente NVRAM.

Tabla 2.36. Descripción de puertos módulo NVRAM.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de Control Memoria	
AIN	Entrada	20:0	Bus de dirección
DATAIN	Entrada	15:0	Bus de datos de Entrada
R	Entrada	Alto	Habilita Operación de Lectura
W	Entrada	Alto	Habilita Operación de Escritura
		Señales de Interfaz y Control NVRAM	
A	Salida	20:0	Bus de dirección
D1	Salida	7:0	Bus de datos de la memoria 1
D2	Salida	7:0	Bus de datos de la memoria 2
CE1	Salida	Alto	Habilitación de Integrado 1
CE2	Salida	Alto	Habilitación de Integrado 2
OE	Salida	Alto	Habilitación de Salida
WE	Salida	Alto	Habilitación de lectura
DATAOUT	Salida	15:0	Bus de datos de Salida
BUSY	Salida	Alto	Bandera de operación en proceso
DATA_READY	Salida	Alto	Dato listo para ser leído

2.3.1.2 Diseño e Implementación del Hardware

El módulo de memorias será agregado como un componente externo a la tarjeta de desarrollo Digilent Spartan-3 haciendo uso de los puertos de Entrada/Salida disponibles en el FPGA.

El arreglo de memorias considera un diseño de 2 memorias NVRAM DS1270W (16 Mb) que manejan 20 líneas de dirección ó DS1265W (8 Mb) que manejan 19 líneas de dirección. La operación se plantea por medio de líneas de control comunes y el empleo de señal de habilitación independientes (CE1 y CE2).Un esquemático detallado se muestra en la Figura 2.34.

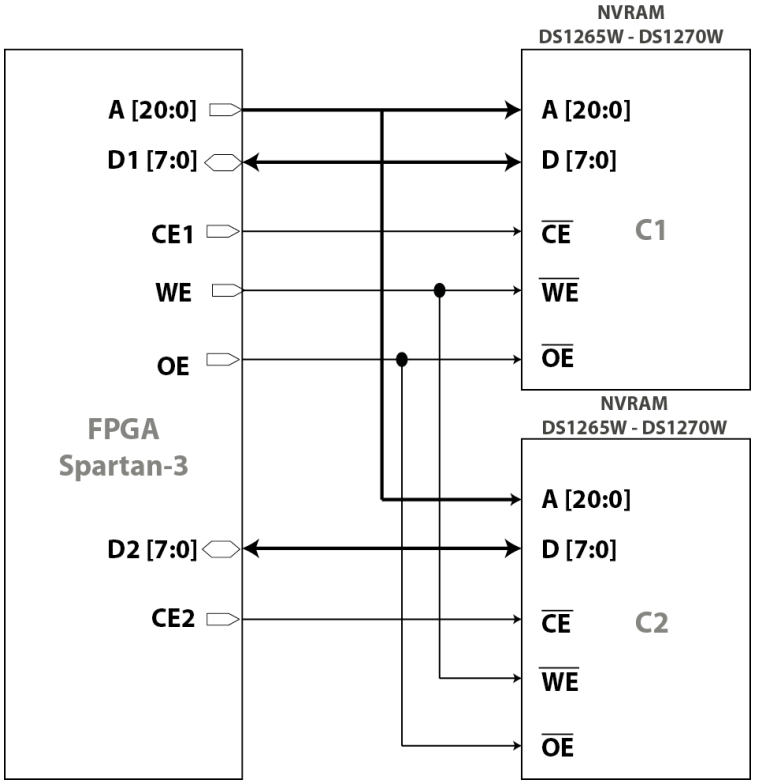


Figura 2.34. Conexiones de FPGA y NVRAM.

Las líneas de dirección se conectan mediante los conectores de expansión A2 y B1, que pueden ser consultados en el manual de referencia del fabricante [13], estableciéndose la configuración de las conexiones como se muestra en la Tabla 2.37.

Tabla 2.37. Conexiones de módulo externo de Memorias al FPGA Spartan-3.

Señal	Pin del FPGA	Conector - Pin	Señal	Pin del FPGA	Conector - Pin
A[0]	A10	A2 – 28	CE1	A8	A2 – 25
A[1]	A9	A2 – 26	CE2	C12	B1 – 12
A[2]	B8	A2 – 24	WE	D7	A2 – 11
A[3]	B7	A2 – 22	OE	B6	A2 – 21
A[4]	A5	A2 – 20			
A[5]	A4	A2 – 18	D1[0]	B12	A2 – 30

Tabla 2.38. (Continuación) Conexiones de módulo externo de Memorias al FPGA Spartan-3.

A[6]	A3	A2 – 16	D1[1]	B13	A2 – 32
A[7]	C9	A2 – 14	D1[2]	B14	A2 – 34
A[8]	D10	A2 – 15	D1[3]	C10	B1 – 4
A[9]	B4	A2 – 17	D1[4]	A13	A2 – 33
A[10]	A7	A2 – 23	D1[5]	A12	A2 – 31
A[11]	B5	A2 – 19	D1[6]	B11	A2 – 29
A[12]	C8	A2 – 12	D1[7]	B10	A2 – 27
A[13]	D8	A2 – 13	D2[0]	E10	B1 – 6
A[14]	C7	A2 – 10	D2[1]	C11	B1 – 8
A[15]	D6	A2 – 7	D2[2]	D11	B1 – 10
A[16]	C6	A2 – 8	D2[3]	C16	B1 – 22
A[17]	E7	A2 – 9	D24]	C15	B1 – 21
A[18]	C5	A2 – 6	D2[5]	B16	B1 – 18
A[19]	D5	A2 – 5	D2[6]	E11	B1 – 16
A[20]	E6	A2 - 4	D2[7]	D12	B1 - 14

En la Figura 2.35 se muestra el circuito esquemático de la tarjeta externa de Memorias NVRAM, así como las conexiones por medio de los conectores de expansión de la tarjeta de desarrollo; en el Anexo B se ofrece el diagrama PCB resultante.

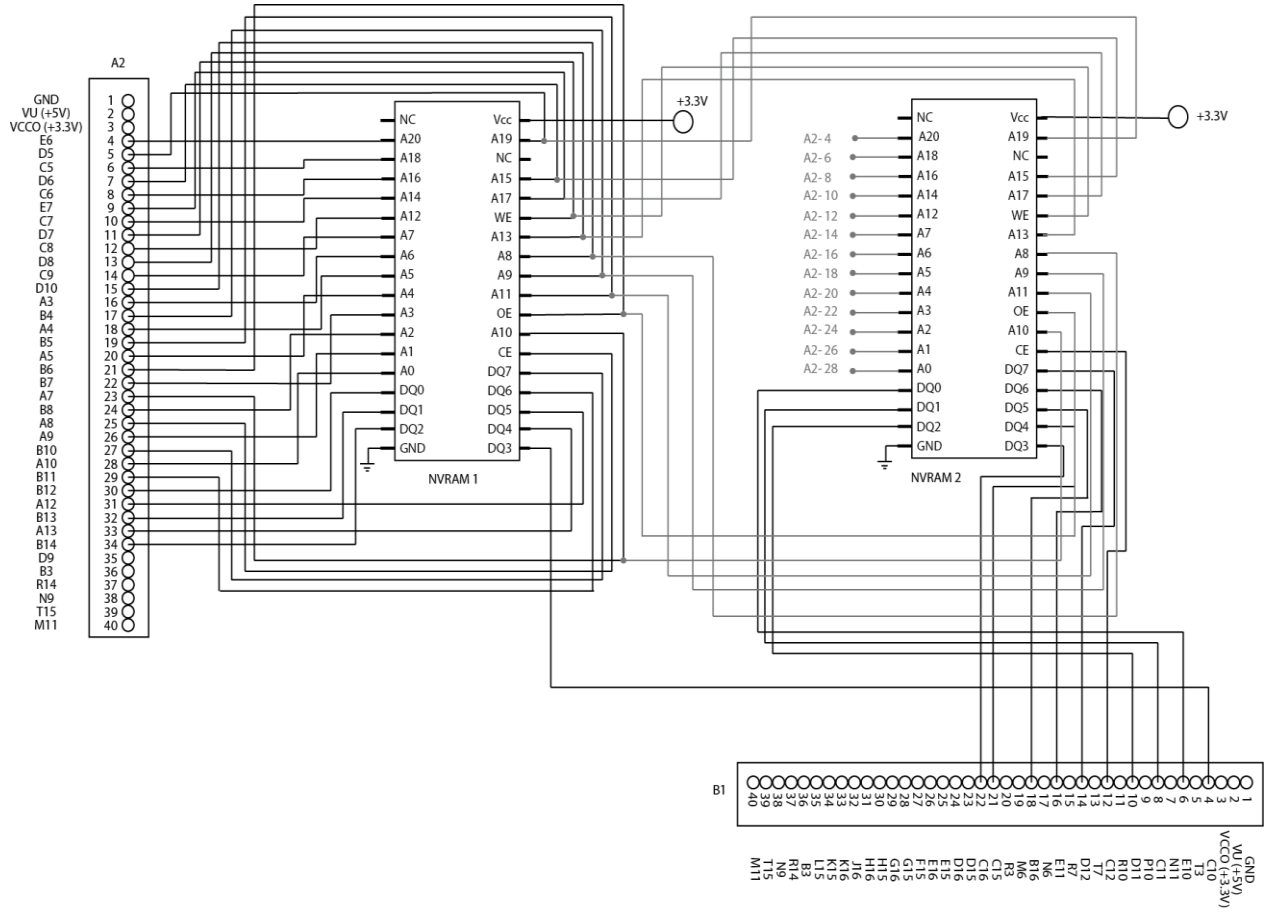


Figura 2.35. Circuito Esquemático de la tarjeta externa de Memorias NVRAM.

2.3.1.3 Descripción funcional del Hardware, Diseño e Implementación

En la figura 2.36 se muestra en un diagrama a bloques la configuración de la memoria NVRAM, permitiendo las funcionalidades de lectura y escritura para datos persistentes.

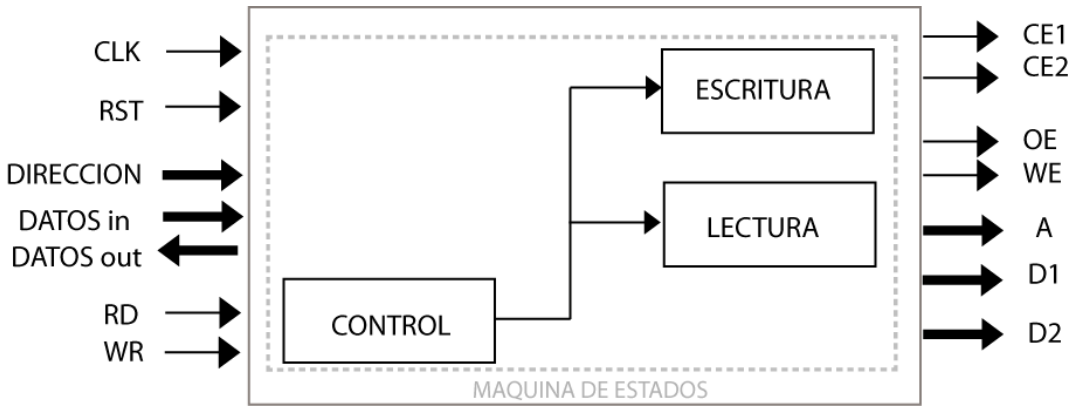


Figura 2.36. Diagrama a bloques de la descripción de hardware componente NVRAM.

La secuencia de control de acceso para Lectura y Escritura de la NVRAM se realiza mediante la máquina de estados que se muestra en la Figura 2.37, que corresponde a las secuencias de acceso y tiempos especificados por el fabricante de este componente electrónico.

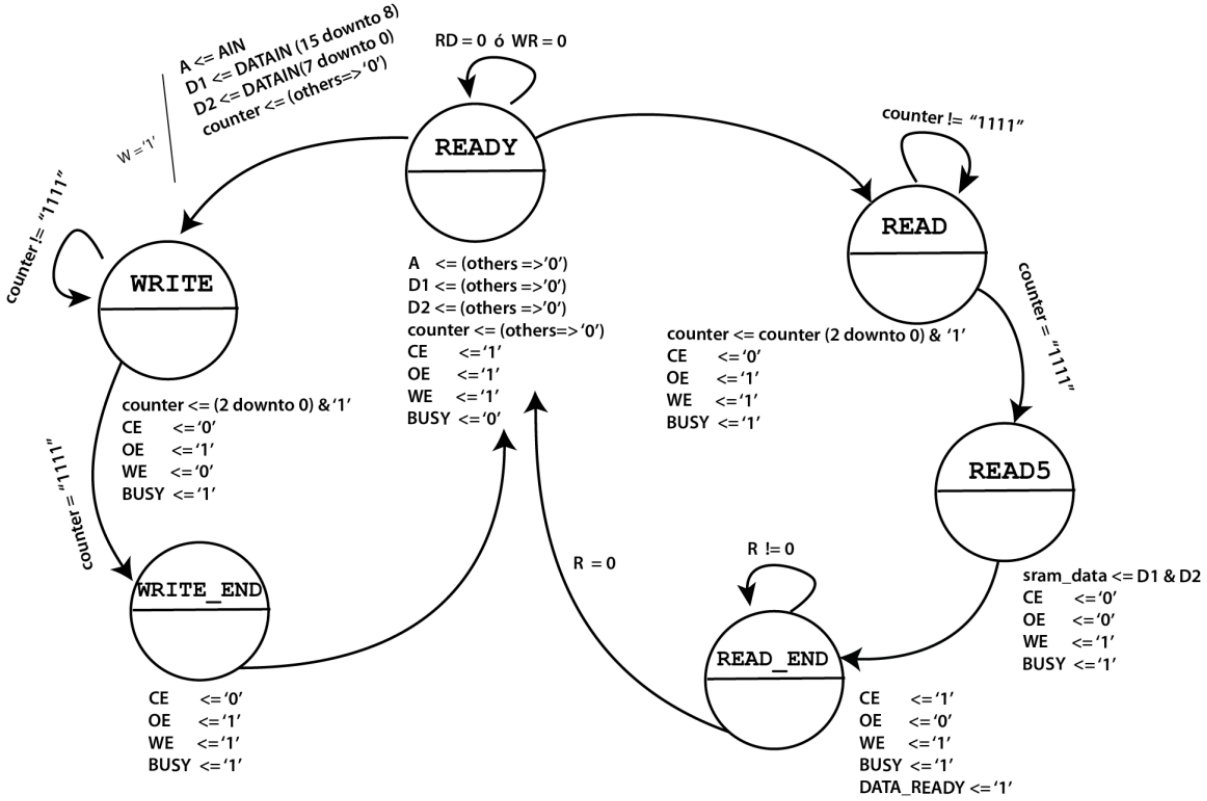


Figura 2.37. Máquina de estados del componente NVRAM.

El puerto de datos, para las operaciones de lectura y escritura, se considera de una longitud de palabra de 2 bytes, considerando que el módulo se comporta como una sola memoria, operacionalmente se escribe cada byte en diferente memoria, pero en la misma dirección.

2.3.2 Componente Control de Memorias

Este componente se concibe para interactuar con la parte superior de la estructura jerárquica, debido a su funcionalidad, que consiste en evitar que exista un conflicto en el acceso por parte de los módulos Serie e Interfaz de Video, ya que los módulos son concurrentes en su operación, el acceso para lectura y/o escritura en la memoria NVRAM debe coordinarse.

2.3.2.1 Descripción funcional

En la Figura 2.38 se muestra el símbolo, equivalente a la descripción de hardware en el diagrama esquemático, del componente Control de Memorias.

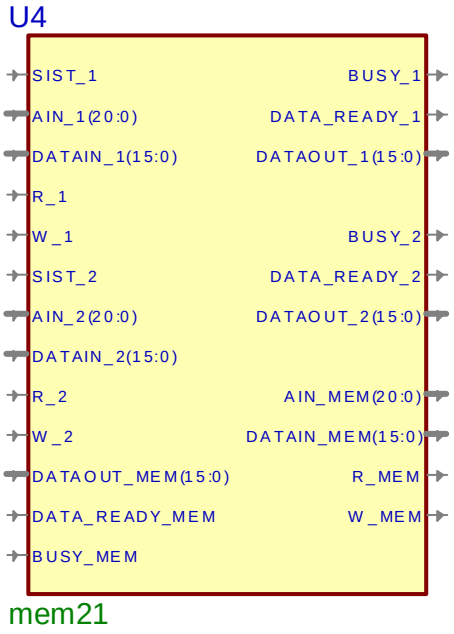


Figura 2.38. Símbolo Descripción de hardware del componente Control de Memorias.

La Tabla 2.38 describe la funcionalidad de cada uno de los puertos que se integran en el componente Control de Memorias.

Tabla 2.39. Descripción de Pines del componente Control de Memorias.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
Señales de Control y Datos Sistema 1			
SIST_1	Entrada	Alto	Señal de petición de uso de memoria
AIN_1	Entrada	20:0	Bus de Dirección
DATAIN_1	Entrada	15:0	Bus de Datos a Escribir
DATAOUT_1	Salida	15:0	Bus de Datos Leídos
R_1	Entrada	Alto	Señal de operación de Lectura Sistema
W_1	Entrada	Alto	Señal de operación de Escritura
Señales de bandera Sistema 1			
BUSY_1	Salida	Alto	Señal de memoria ocupada
DATA_READY_1	Salida	Alto	Señal de bus de datos disponible
Señales de Control y Datos Sistema 2			
SIST_2	Entrada	Alto	Señal de petición de uso de memoria
AIN_2	Entrada	20:0	Bus de Dirección
DATAIN_2	Entrada	15:0	Bus de Datos a Escribir
DATAOUT_2	Salida	15:0	Bus de Datos Leídos
R_2	Entrada	Alto	Señal de operación de Lectura
W_2	Entrada	Alto	Señal de operación de Escritura
Señales de bandera Sistema 2			
BUSY_2	Salida	Alto	Señal de memoria ocupada
DATA_READY_2	Salida	Alto	Señal de bus de datos disponible
Señales de Control y Datos NVRAM			
AIN_MEM	Salida	20:0	Bus de Dirección de memoria
DATAIN_MEM	Salida	15:0	Bus de Datos Leídos
DATAOUT_MEM	Entrada	15:0	Bus de Datos a Escribir
R_MEM	Salida	Alto	Señal de operación de Lectura
W_MEM	Salida	Alto	Señal de operación de Escritura
Señales de Bandera			
BUSY_MEM	Entrada	Alto	Señal de memoria ocupada
DATA_READY_MEM	Entrada	Alto	Señal de dato disponible

2.3.2.2 Descripción del Hardware, Diseño e Implementación

El diseño de este componente es completamente combinacional, por tanto el control es exclusivo del valor de las señales de control de entrada, en este caso la señal de control *BUSY* indica que la memoria está ocupada, por ende no puede ser empleada. *BUSY_1*, indica al *SIST_1*, interface de video, que la interface serial está empleando la memoria; y viceversa en el caso de BUSY_2.

En caso de que ninguna de las interfaces asociadas solicite control de la memoria, se asigna la señal de operación del controlador de la memoria que se establece en estado lógico bajo cuando se encuentre disponible, como se aprecia en las sentencias de la Tabla 2.39.

Tabla 2.40. Sentencias generadoras de las señales de control de memoria ocupada.

```
1  BUSY_1 <= '1' when SIST_2 = '1' else BUSY_MEM;
2  BUSY_2 <= '1' when SIST_1 = '1' else BUSY_MEM;
```

En las sentencias de la Tabla 2.40 se aprecia la asignación de los buses de dirección, datos y control a la interface que solicita acceso a los recursos de la memoria; y el control de esta selección es determinado por *SIST_1*, que corresponde al estado de la solicitud de control de la memoria de la interface de video.

Tabla 2.41. Descripción de la selección de señales de control de la memoria.

```
1  selec <= SIST_1;

2  AIN_MEM <= AIN_1 when selec = '1' else AIN_2;
3  DATAIN_MEM <= DATAIN_1 when selec = '1' else DATAIN_2;
4  R_MEM <= R_1 when selec = '1' else R_2;
5  W_MEM <= W_1 when selec = '1' else W_2;
```

2.4 Integración del Sistema Visualización de Imágenes

El sistema de visualización de imágenes, se genera, de acuerdo a la metodología jerárquica, de la instanciación de los módulos que en la estructura de diseño le preceden:

- Módulo interfaz de video
- Módulo interfaz serie
- Componente Controlador de Memorias
- Módulo NVRAM

En el esquemático de la Figura 2.39 se muestra la integración del sistema completo, y la forma en que se integran y conectan los módulos, identificados en el proceso de diseño y desarrollados estructuralmente para generar la funcionalidad global; así podemos apreciar, que la señal generada por el módulo Interfaz de video, STATUS_TRASNFER, es la única relación que establece hacia el módulo Interfaz Serie; la cual indica que se está realizando uso del recurso de la memoria NVRAM; así la operación de ambos módulos es de naturaleza concurrente.

En tanto, las señales de control y datos del módulo NVRAM se encuentran conectadas al componente Controlador de Memorias, para determinar la prioridad de acceso y que no exista contienda por el recurso.

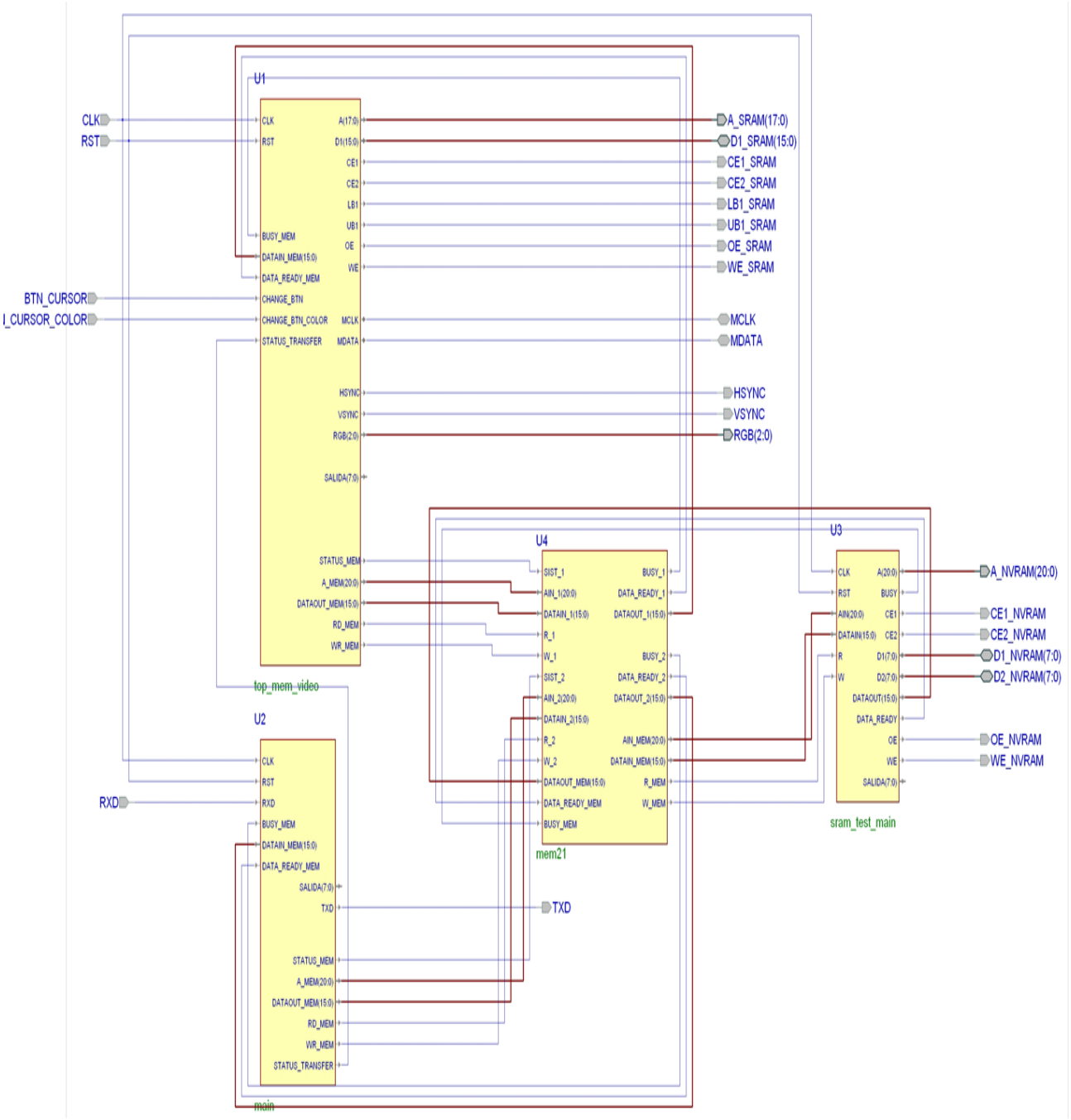


Figura 2.39. Esquemático de la conexión de los módulos interfaz de video, interfaz PS2 e interfaz de Memoria.

Finalmente, los puertos de salida y entrada, detallados en la Tabla 2.41, deben ser conectados a los puertos físicos del FPGA como parte de la implementación, conocida en el ciclo de diseño como: Colocación y Ruteo; que básicamente consiste en adaptar el diseño a un hardware en concreto, que en este caso corresponde al FPGA Xilinx Spartan-3 XC3S200, que corresponde al núcleo de la tarjeta sobre la que se implementará la arquitectura.

Tabla 2.42. Descripción de Pines del Sistema de Visualización de Imágenes.

Puerto	Dirección Puerto	Polaridad/ Ancho Bus	Descripción
		Señales de Control	
CLK	Entrada	Flanco	Reloj de sistema externo de 50 Mhz
RST	Entrada	Alto	Reset de sistema externo
		Señales de interfaz y Control SRAM	
A	Salida	17:0	Bus de dirección de memoria
D1	Salida	15:0	Bus de datos de memoria
CE1	Salida	Alta	Señal de habilitación (Chip Enable) memoria 1
CE2	Salida	Alta	Señal de habilitación (Chip Enable) memoria 2
LB1	Salida	Alta	Acceso Byte de Datos menos significativo
UB1	Salida	Alta	Acceso Byte de Datos mas significativo
OE	Salida	Alta	Señal de Salida Habilitada (Output Enable)
WE	Salida	Alta	Señal de hablitación de Escritura
		Señales de Interfaz VGA	
RGB	Salida	3	Señal con información del píxel activo RGB
HSYNC	Salida	Nivel	Señal de Control de sincronismo Horizontal
VSYNC	Salida	Nivel	Senal de Control de sicronismo Vertical
		Señales de Interfaz Mouse PS/2	
MCLK	Bidireccional	Triestado	Señal de Reloj de sincronización
MDATA	Bidireccional	Triestado	Señal de Datos
		Señales de Interfaz Botones de Presión	
BTN_CURSOR	Entrada	Alto	Señal Selección de Cursor
CHANGE_BTN_ COLOR	Entrada	Alto	Señal de inversión de bit de Color
		Señales de Interfaz UART	
TX	Salida		Señal Transmisión de Datos
RX	Entrada		Señal Recepción de Datos
		Señales de Interfaz NVRAM	
A_NVRAM	Salida	20:0	Bus de dirección
D1_NVRAM	Salida	7:0	Bus de datos de la memoria 1
D2_NVRAM	Salida	7:0	Bus de datos de la memoria 2
CE1_NVRAM	Salida	Alto	Habilitación de Integrado 1
CE2_NVRAM	Salida	Alto	Habilitación de Integrado 2
OE_NVRAM	Salida	Alto	Habilitación de Salida
WE_NVRAM	Salida	Alto	Habilitación de lectura

3. Software

En este capítulo se describe la metodología y las funciones del sistema software de simplificación y transmisión de imágenes.

3.1 Ciclo de vida del Software

El sistema de visualización de imágenes requiere del desarrollo de un sistema de Software específico para la aplicación, este proceso requiere de varias etapas que en conjunto se denominan el ciclo de vida del software[11] y en cada caso, en función de cuales sean las características del proyecto, se configurará el ciclo de vida de forma diferente.
En el caso que ocupa al presente documento se optó por emplear el modelo incremental, ya que por la naturaleza del Sistema de Visualización de Imágenes y del Sistema Software es necesario que en el proceso de desarrollo sea posible:

- Contar con flexibilidad, de manera que el sistema se desarrolle en versiones y cada una satisfaga un subconjunto de los requisitos especificados para realizar pruebas con la contraparte en Hardware.
- Aumentar las funcionalidades en cada versión o bien mejorarlas para que satisfagan más requisitos.
- Evaluar cada desarrollo y planear el siguiente incremento.

3.2 Análisis de Requisitos y Diseño

El estudio de las posibilidades con las que se cuenta para el desarrollo del sistema, así como sus limitaciones, se establece una lista de requisitos, tanto a nivel funcional como no funcional.

3.2.1 Panorama General

Este proyecto tiene por objeto crear un sistema que permita transferir imágenes BMP simplificadas a 8 colores por el puerto serie al sistema de proyección de imágenes, así también, que permita ver la imagen original, almacenarla y/o reemplazarla.

[11] Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso. ISO 12207-1

3.2.2 Metas

La meta que se persigue con el desarrollo de este sistema es obtener imágenes de resolución de 640X480 pixeles, simplificarlas y transferirlas mediante el puerto Serie al sistema de proyección de imágenes, lo que incluye:

- Almacenamiento de Imágenes
- Simplificación de las imágenes
- Transferencia mediante el puerto serie
- Gestión visual de las imágenes

3.2.3 Requisitos Funcionales

Dentro de los requisitos que especifican la funcionalidad que debe realizar el sistema o un componente en específico, se divide en grupos lógicos y se listan de acuerdo a las funcionalidades esperadas:

Funciones de Imágenes:

1. Almacenar y Visualizar 32 imágenes en formato BMP de tamaño estándar 640X480.
2. Reemplazar una imagen a selección del usuario.
3. Simplificar la imagen seleccionada a 8 colores (3 bits) mediante la aplicación de un filtro.
4. Visualizar una imagen a selección del usuario en resolución de colores y tamaño original.
5. Visualizar una imagen a selección del usuario en resolución de 3 bits y tamaño original.

Funciones de Transferencia:

1. Comunicación mediante puerto serie a una velocidad de 115200 bps.
2. Transmisión de una imagen al Sistema de Proyección de imágenes en la dirección de imagen seleccionada.

Funciones de Usabilidad:

1. Mostrar un indicador de transferencia de información y avance de la misma.
2. Mostrar una imagen vacía cuando en una dirección de imagen no se encuentre almacenada alguna.

Requisitos no funcionales:

1. El sistema será desarrollado con compatibilidad para el Sistema Operativo Microsoft Windows 95, 98, 2000, XP, 7.
2. El sistema se codificará en Lenguaje C++, empleando la herramienta Builder 6.0 de Borland, Inc.

3.2.4 Descripción del Sistema Software

El Sistema Software se estructura de manera que permita el mantenimiento y modificación de la funcionalidad durante el ciclo de desarrollo y pruebas, por ello se emplean funciones que realizan tareas específicas y funciones que responden a eventos concretos. Dentro de las funciones se tienen:

- Gestión del Puerto Serie
- Manejo de Archivos de Imágenes
- Simplificación de imágenes
- Transmisión de Datos
- Manejo de la Interfaz del Software

3.2.4.1 Puerto Serie

Permite realizar una conexión mediante el Puerto Serial, interfaz RS-232, para que el software pueda emitir algún tipo de salida hacia el exterior del mismo hardware donde se ejecuta.
El puerto serie se puede acceder mediante cualquier versión de Windows compatible con RAD Studio C++ Builder empleando algunas funciones y métodos de la Interfaz de Programación de Aplicaciones (API, Application Programming Interface).

La estructura del Bloque de Control de Dispositivo (DCB, Device Control Block) define la configuración de control para un dispositivo serie.

1. Se declara una variable del tipo de la *estructura **DCB***, que define los valores de control para dispositivos de comunicaciones series, para establecer la configuración del puerto.

2. Se requiere abrir el puerto serie a emplear, utilizando **CreateFile** que devuelve un apuntador que se usa para acceder al objeto, puede ser COM1, COM2, etc.

3. Se asignan los parámetros de TimeOut para todas las operaciones de lectura y escritura en el dispositivo de comunicaciones especificado mediante la función *SetCommTimeouts*, que escribe en la estructura **COMMTIMEOUTS**. Los parámetros determinan el comportamiento de las operaciones de lectura y escritura en el dispositivo.

4. Se configura el dispositivo de comunicaciones inicializando la estructura DCB y obteniendo mediante la función **GetCommState** un apuntador al puerto abierto. Teniendo acceso al puerto, se establecen los parámetros que determinan la comunicación y se aplican mediante la función **SetCommState** que configura el dispositivo de acuerdo a las especificaciones en el DCB; esta función inicializa el hardware y maneja los parámetros mediante una cadena que contiene el Baud Rate, la Paridad, el tamaño de palabra y la cantidad de bits de paro; que en este caso corresponde:

- Baud Rate : 115200 bps
- Paridad: Sin Paridad
- Tamaño de Bytes: 8 Bits
- Bits de Paro: 2 Bits

5. Para escribir un dato en el puerto serie, se utiliza la función **TransmitCommChar** que transmite un carácter específico por delante de cualquier dato pendiente en el buffer de salida en el dispositivo de comunicación especificado.

6. Finalmente, es necesario *Cerrar* el Puerto Serie indicando que se finaliza la transmisión.

El código que permite establecer la comunicación mediante el puerto serie, se muestra en la Tabla 3.1

Tabla 3.1. Código de manejo del puerto serie.

```
1  DCB dcbCommPort;
2   hComm = CreateFile("COM3",
3                      GENERIC_READ | GENERIC_WRITE,
4                      0,
5                      0,
6                      OPEN_EXISTING,
7                      0,
8                      0);
9
10 // SI EL PUERTO NO PUEDE ABRIRSE
11  if(hComm == INVALID_HANDLE_VALUE) Application->Terminate();
12
13 // ESTABLECEMOS LOS TIMEOUTS
14  GetCommTimeouts(hComm,&ctmoOld);
15  ctmoNew.ReadTotalTimeoutConstant = 100;
16  ctmoNew.ReadTotalTimeoutMultiplier = 0;
17  ctmoNew.WriteTotalTimeoutMultiplier = 0;
18  ctmoNew.WriteTotalTimeoutConstant = 0;
19  SetCommTimeouts(hComm, &ctmoNew);
20
21 // ESTABLECEMOS BAUD RATE, PARITY, WORD SIZE, Y STOP BITS.
22 dcbCommPort.DCBlength = sizeof(DCB);
23 GetCommState(hComm, &dcbCommPort);
24 BuildCommDCB("115200,N,8,2", &dcbCommPort);
25 SetCommState(hComm, &dcbCommPort);
26
27 // ENVIA EL DATO "128" MEDIANTE EL PUERTO SERIE
28 TransmitCommChar(hComm, 128);
29 // SE CIERRA EL DISPOSITIVO EMPLEADO
30 PurgeComm(hComm, PURGE_RXABORT);
31  SetCommTimeouts(hComm, &ctmoOld);
32  CloseHandle(hComm);
```

3.2.4.2 Manejo de Archivos de Imagen

Los archivos de imagen son una parte necesaria dentro del Sistema Software, ya que por principio, es más cómodo al usuario del sistema visualizar y controlar el conjunto de imágenes, que establecer operaciones individuales. En consecuencia, las imágenes se manejan como un arreglo de 32 elementos, sobre los que se establece la convención, para facilitar su ubicación y manejo, que el identificador de archivo corresponda a su posición en el arreglo; así también, se establece una carpeta por defecto, que sirve de referencia y almacén de los archivos de imagen a los que se referencia el arreglo.

El arreglo de imágenes es cargado una vez iniciado el sistema, en este proceso solamente se asignan los elementos existentes, esto es, los que hayan sido previamente guardados, en caso contrario, el elemento se encuentra aún sin asignar, por lo que se establece una imagen por defecto para hacer notar visualmente este hecho. En la Tabla 3.2 se muestra el código sintetizado para la función de abrir archivos de imagen.

Tabla 3.2. Extracto del Código de la función abrir imagen.

```
1  if (FileExists(exePath + "\imagenes\\" + num_image + ".bmp")){
2   imagenBmp->LoadFromFile(exePath + "\imagenes\\" + num_image + ".bmp");
3   lstPictures->Add(imagenBmp, imagenBmp);
4   imagenOpen->Picture->Graphic = imagenBmp;
5   imagenOpen->Canvas->Draw(0, 0, imagenBmp);
6  }
7  else  {
8   imagenBmp->LoadFromFile(exePath + "\imagenes\\empty.bmp");
9   lstPictures->Add(imagenBmp, imagenBmp);
10  imagenOpen->Picture->Graphic = imagenBmp;
11  imagenOpen->Canvas->Draw(0, 0, imagenBmp);
12 }
```

En la función guardar se establece el identificador de archivo, por el índice que determina la posición del elemento sobre el que se desea guardar la imagen, sobrescribiendo el archivo previo o creando uno nuevo en caso de no existir. Este código se muestra en la Tabla 3.3.

Tabla 3.3. Extracto del Código de la función Guardar imagen.

```
1  Imagen->Picture->LoadFromFile(OpenDialog1->FileName);
2  Imagen->Picture->SaveToFile(ExtractFilePath(Application->ExeName) +
    "\imagenes\\" + indice +".bmp");
3  Original->LoadFromFile(OpenDialog1->FileName);
4  lstPictures->Replace(indice-1, Original, Original);
```

3.2.4.3 Simplificación de Imágenes

La simplificación de imágenes se realiza tomando cada uno de los pixeles que estructura la imagen (640 x 480 = 307,200 pixeles), para luego descomponerlo en sus componentes de color (RGB) , para así discriminar cada una de esas componente fijando un umbral que corresponde a la mitad del espectro con que se represente el color, que corresponde a un valor numérico que se encuentra dentro del rango de 0-254, como se muestra en la figura 3.1 ; así, la componente se

incluye si presenta una saturación por encima del umbral, ya que indica mayor presencia que ausencia en ese componente de color, en caso contrario se descarta.

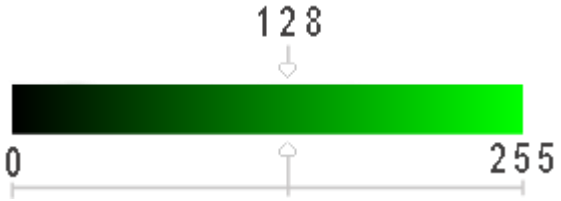


Figura 3.1. Representación numérica del espectro de Color Verde.

El tipo Tcolor, Tabla 3.4, permite especificar el color de un objeto, puede especificarse en un valor decimal o definirlo por sus componentes RGB.

Tabla 3.4. Definición de la estructura tipo Tcolor.

```
1  union  tRGB {
2     unsigned char    bases[4];
3     TColor  result;
4  } Fuente;
```

Se asigna un pixel específico de la imagen original dibujada en el Canvas, como se describe por la línea de código de la Tabla 3.5.

Tabla 3.5. Dibujo de un pixel sobre el Canvas.

```
1  Fuente.result = Original->Canvas->Pixels[j][i];
```

Si la saturación del componente de color es mayor que 128 se coloca un 1 en el bit indicando la presencia de ese color; en otro caso, se coloca un 0 indicando la ausencia de ese componente de color. Esto se describe con el extracto de código específico de la operación que se muestra en la Tabla 3.6.

Tabla 3.6. Discriminador de Umbral de Color.

```
1  if (Fuente.bases[f] >= 0x80)
2             tempore = tempore |  0x80;
3          Else
4             tempore = tempore & 0x7F;
```

Los pixeles que se simplifican mediante sus componentes de color se almacenan en dos matrices bidimensionales de tipo *unsigned char*, una matriz, que se identifica como ***m***, que almacena la imagen simplificada para ser visualizada en el componente TCanvas; y una matriz que se identifica como ***y***, que almacena la imagen simplificada que posteriormente será transmitida.
Ambas matrices manejan la misma información, una imagen simplificada, pero la almacenan de maneras diferentes:
La matriz bidimensional ***m*** destina un byte por componente, como se muestra en la Figura 3.2, lo que permite facilitar su reconstrucción, y si se requiriera, el tratamiento de la información por componente de color. En 3 bytes se tiene la información de 8 pixeles.

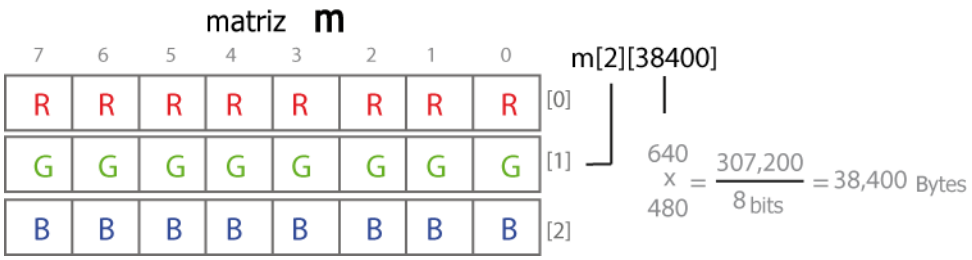


Figura 3.2. Matriz bidimensional de manejo de los componentes de colores primarios de manera atómica.

La matriz bidimensional ***y*** almacena la información en dos arreglos, uno que representa la parte alta y otro la parte baja de una palabra de 16 bits. En la Figura 3.3 se observa cómo se organizan 5 pixeles en una palabra de 16 bits desechando el bit 15, para obtener una representación ordenada de la información. La finalidad del manejo de esta matriz es que el hardware maneja la información mediante un ancho de palabra de 16 bits por dirección.

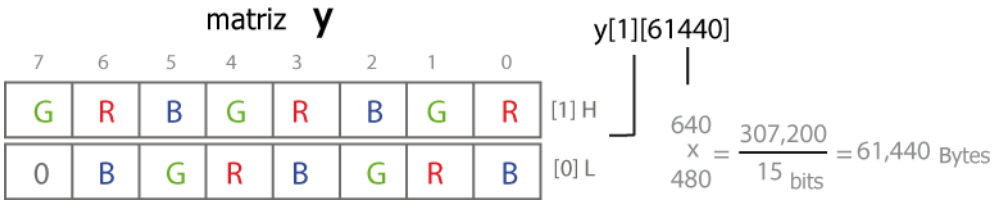


Figura 3.3. Matriz bidimensional de manejo de los componentes de colores primarios de manera secuencial.

3.2.4.4 Transmisión de Datos

La transmisión de se establece tomando como base a la imagen actualmente seleccionada, de la cual se envía la versión simplificada, de ésta, se toma la ubicación que se le dará en la memoria, a la que se denomina *página.* En el hardware del sistema se tiene una capacidad de 32 posiciones, de la 0 a la 31, lo que permite almacenar 32 imágenes.

El primer byte que se envía al hardware se compone del comando de operación a realizar y el número de página sobre la cual se va a efectuar la operación. Tal como se muestra en la figura 3.4.

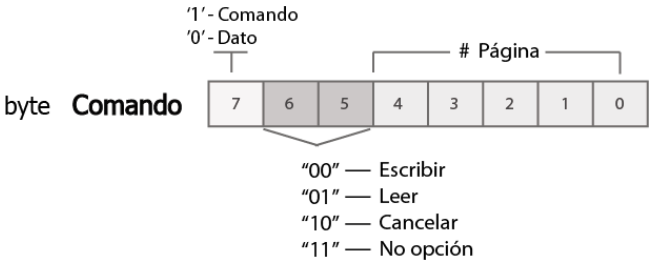


Figura 3.4. Byte de Comando y Número de página.

Para el comando que corresponde con la operación de escribir, se inicia el bucle que permite la lectura de cada uno de los bytes almacenados durante el proceso de simplificación de imagen, información correspondiente a la matriz ***y***, los que se envían hasta completar los 61,440 bytes que componen a la imagen, como se observa en la Tabla 3.7. Durante dicho proceso se hace uso del hardware y se emplea el hilo de ejecución actual, por lo que el programa se ocupa completamente de este proceso; limitando con esto las acciones que pueda realizar el usuario durante la ejecución de la transmisión, que corresponde al proceso crítico del sistema.

Tabla 3.7. Código para la transmisión por el puerto serie de una página.

```
1   a = actual;  // Comando Escribir en la página => 128 + # de página
3   TransmitCommChar(hComm, a);  //Transmisión Puerto Serie
4   for (b =0 ; b < 61440; b++)              //  61, 440 palabras
5    for (c = 0; c < 2 ; c ++) {               // 2 palabras de 8 bits
6        TransmitCommChar(hComm, y[c][b]);
9        ProgressBar1->Position = b*.00163;     //Barra de Progreso
10   }
```

3.2.4.5 Manejo de la interfaz del Software

La interfaz del Software se compone de la parte de la aplicación que el usuario ve y con la cual interactúa. La interfaz está relacionada con las funciones que realizan las diferentes operaciones del software e incluye los siguientes elementos:

Menú, es la lista de opciones que presenta el software, mediante una interfaz desplegable con las opciones que se muestran en la Figura 3.5.

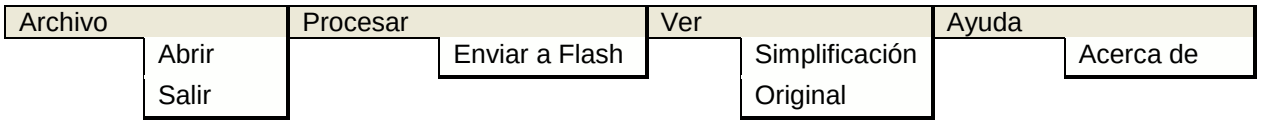


Figura 3.5. Opciones del Menú de la aplicación.

Barra de Progreso, corresponde al indicador visual del progreso de transmisión de la imagen, se muestra en la Figura 3.6.

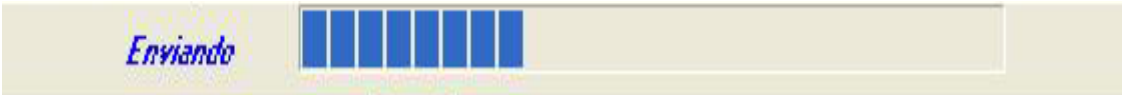


Figura 3.6. Barra de Progreso de transmisión.

Tira de Imágenes, arreglo de 32 imágenes, que muestra las imágenes de acuerdo a su posición, con desplazamiento vertical, como se muestra en la Figura 3.7.

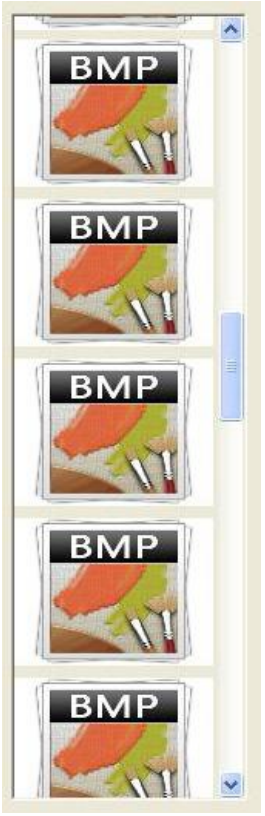


Figura 3.7. Tira de imágenes con el arreglo de elementos imagen.

Visor de Imágenes, es la representación a escala real, 640 x 480 pixeles, de la imagen seleccionada en formato original o simplificado; que en términos de componentes, corresponde al dibujo de la imagen sobre un elemento Canvas. El cual se muestra en la Figura 3.8 con la imagen que se ha establecido por defecto.

Figura 3.8. Visor de imagen.

La interfaz completa se muestra en la Figura 3.9, así como la manera en que se integran los elementos para crear la interfaz visual del sistema, y que permite interactuar con el arreglo de imágenes en la pantalla y no solo presentarlas.

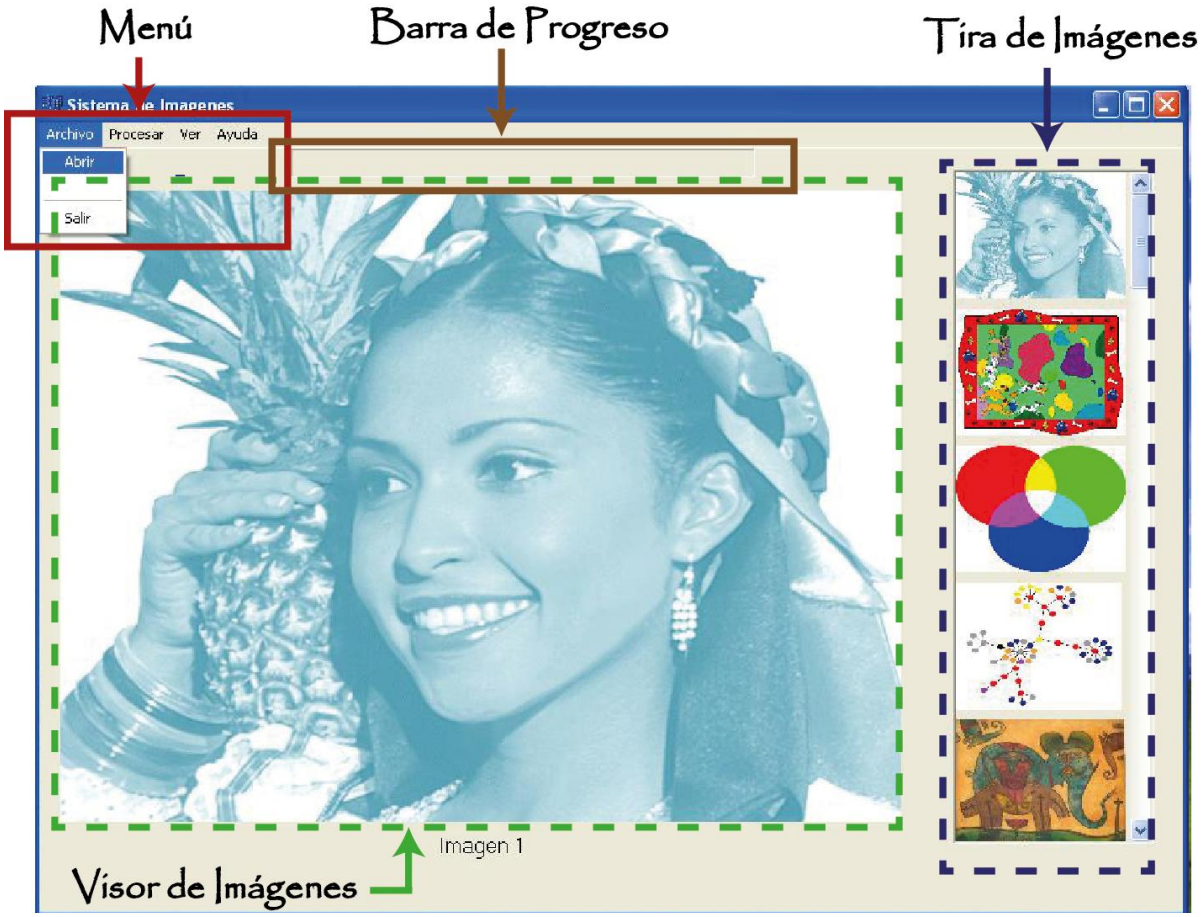


Figura 3.9. Elementos de la interfaz visual de la aplicación.

Finalmente, en la Figura 3.10 se muestra la comparación entre las represetaciones de una imagen original, formato BMP con 256 colores, y una imagen simplificada a 8 colores; dentro del lienzo establecido como visor de imágenes.

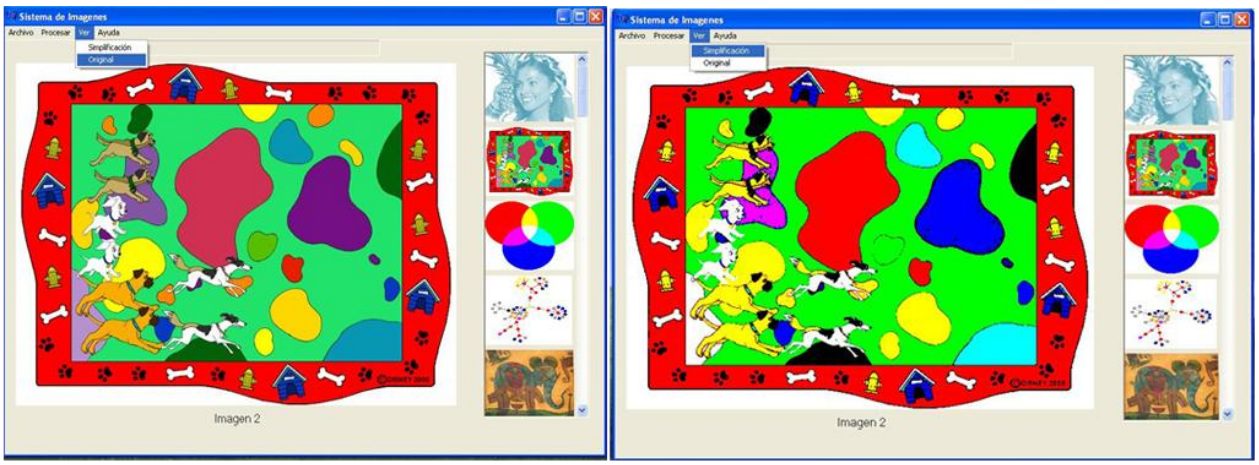


Figura 3.10. Visualización de una imagen en formato original en 256 colores y simplificada a 8 colores.

4. Resultados y Conclusiones

El presente trabajo de tesis consistió en la implementación de diversos módulos con el objetivo de llegar a la implementación final de un sistema de visualización de imágenes flexible y escalable, sintetizado en la tarjeta de desarrollo Digilent Spartan-3. Diseñando y generando genéricamente los principales componentes, con la intención de que puedan seguir empleándose, reutilizando el código en otras aplicaciones o integrando los componentes en bloques de mayor complejidad; ya que por la naturaleza inherente, el diseño e implementación sobre un dispositivo reconfigurable proporciona la capacidad de realizar rediseño, cambios o modificaciones en pro de su optimización.

En la fase de pruebas de los bloque identificados en el ciclo inicial de diseño, se realizaron pruebas de caja negra a cada uno de los bloques y en caso de que la funcionalidad no fuera la esperada, se hicieron las modificaciones de código y la síntesis hasta obtener su correcto funcionamiento con base a las especificaciones originales bajo las que se planteó. Teniendo en cuenta que la descripción en VHDL se desarrolló con un estilo estructural, se probaron los sistemas más pequeños, luego los sistemas donde están integrados los anteriores y así hasta simular el sistema completo.

4.1 Resultados

El sistema resultante cumplió con las expectativas planteadas, la implementación del diseño alcanza perfectamente en el FPGA 3S200FT256, en la Tabla 4.1 se muestran los recursos que el sistema ocupa dentro del dispositivo.

Tabla 4.1. Tabla de los recursos empleados por el sistema de visualización de imágenes en un FPGA 3S200FT256.

Recursos	Cantidad Usada	Porcentaje
SLICEs	865 de 1920	45%
IOBs	93 de 173	53%
GCLKs	1 de 8	12%

En términos de almacenamiento de información, la implementación del sistema con memorias NVRAM DS1265W permite el almacenamiento de 32 imágenes y de 64 imágenes con la memoria DS1270W.

La transferencia máxima que se establece en la transmisión de datos, se limita por el puerto serie a 115,200 bps, ya que durante la fase de pruebas este es el valor que permite la estabilidad de la aplicación y transferencia efectiva de la información en los diferentes equipos que se realizaron

las pruebas de comunicación; adicionalmente de que este valor, se establece como la máxima tasa de transferencia de la gran mayoría de puertos seriales incorporados en PC comerciales. Obteniendo bajo esta configuración, la transferencia de una imagen completa en 12 segundos por los tiempos de retraso agregados en Software.

El sistema Software presenta compatibilidad de operación con los sistemas Windows 95, 98, 2000, XP y 7; adicionalmente se empleo un adaptador de USB macho tipo “A” a serial (BD9) USB-RS232 para la transferencia de imágenes en PCs que únicamente contaban con puerto USB; permitiendo la transferencia de las imágenes con un demora de 5 segundos durante el proceso de la transmisión en sistemas con Windows 7.

En la Figura 4.1 se muestra el sistema de visualización de imágenes en funcionamiento, conectado a un monitor LCD, una computadora y un ratón PS/2. La imagen desplegada corresponde a la transferencia realizada por el software vía interfaz serie de la computadora en la primera posición de la memoria de video.

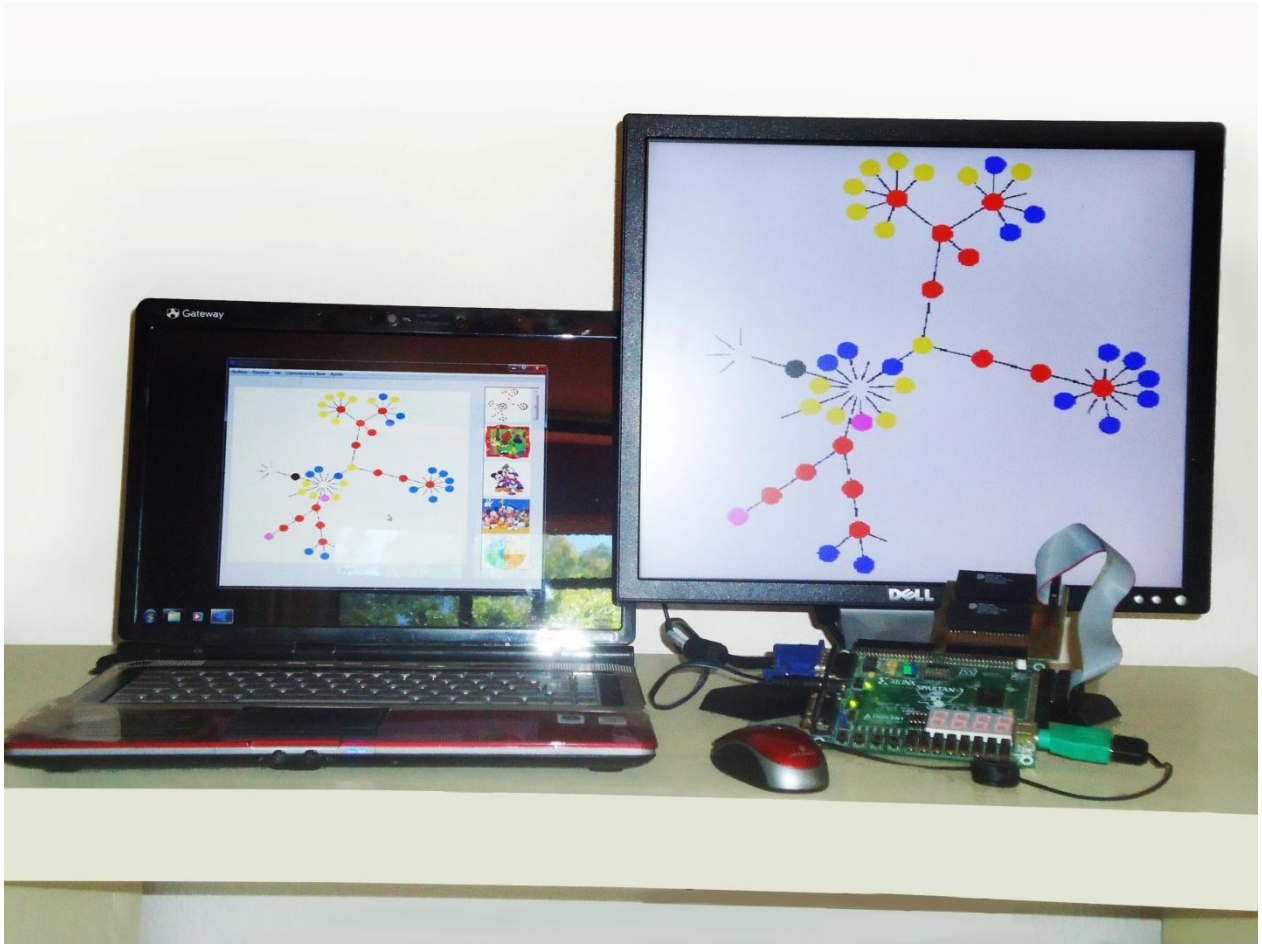


Figura 4.1. Sistema de Visualización de Imágenes en funcionamiento con periféricos conectados.

La Figura 4.2 muestra el sistema de visualización de imágenes conectado únicamente a un monitor LCD, que despliega la imagen almacenada en la primera posición de la memoria de video, y a un ratón PS/2 que presenta la funcionalidad de señalización y control de la imagen visualizada.

Se implementó una metodología de diseño descendente, Top-Down, para el proceso de diseño e identificación de los módulos en el sistema; la descripción estructura, síntesis y simulación de los componentes se realizó con la herramienta.

La integración de uso del sistema visualización de imágenes se completa con un software para computadora, que permite la manipulación de imágenes y la comunicación con el puerto serial. El software se desarrolló empleando el modelo incremental y como herramienta de desarrollo el entorno de programación para C++.

En el FPGA, el uso de la metodología de diseño jerárquica permitió implementar los bloques de hardware atómicamente, realizar pruebas de caja negra y funcionalidad individualmente en cada uno de los bloques.

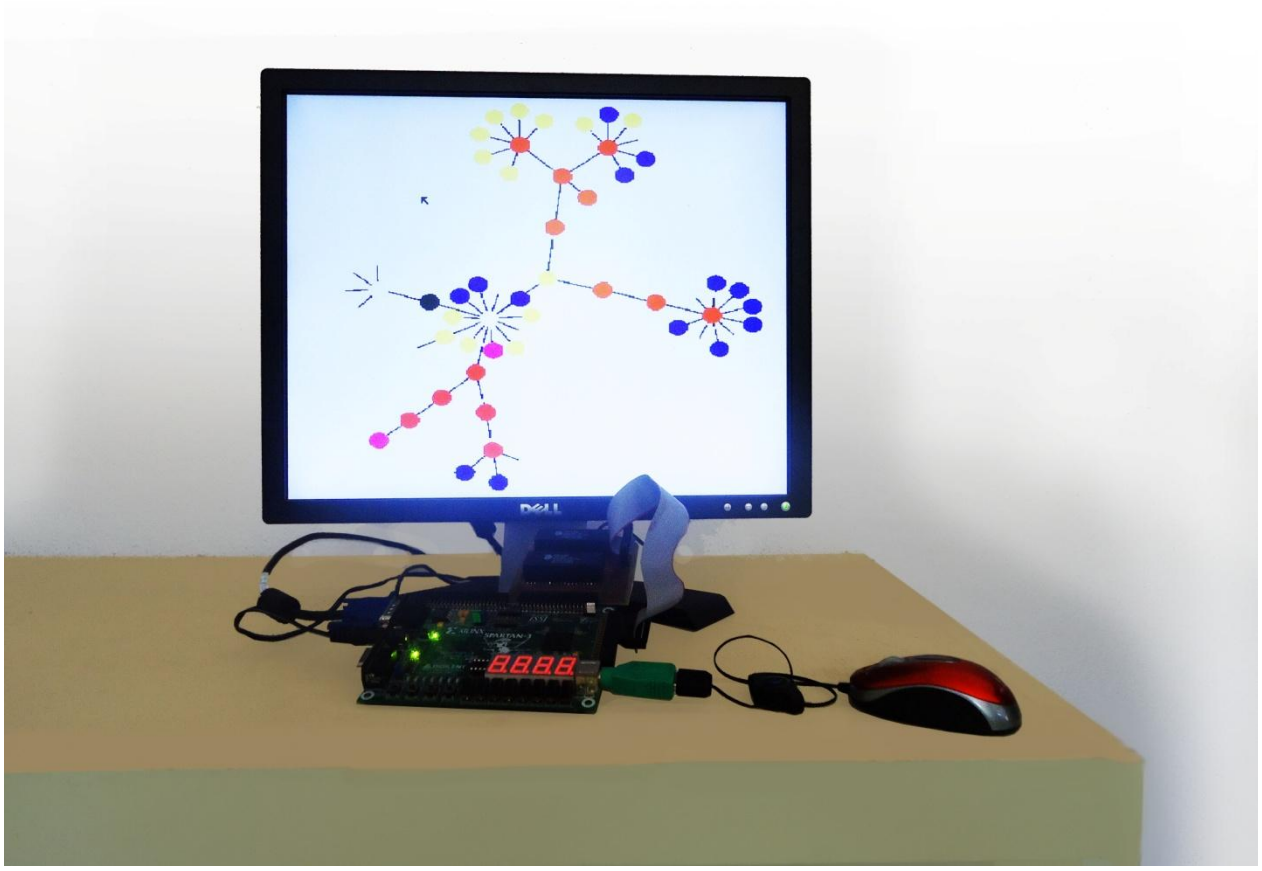


Figura 4.2. Sistema de Visualización de Imágenes desplegando una imagen.

Los resultados que se han obtenido permiten demostrar que el trabajo de desarrollo presentado cumple con el propósito específico de crear un sistema de visualización de imágenes a 8 colores en la tarjeta Digilent Spartan-3, a partir de un conjunto de módulos diseñados con la característica de ser reutilizables en diferentes aplicaciones y tecnologías. Lo que permite que el sistema, parte del sistema o módulos en específico, sean implementados en otros prototipos de desarrollo.

4.2 Conclusiones

Una vez concluido el presente trabajo, se tienen las siguientes conclusiones:

- La arquitectura del sistema de visualización de imágenes implementada sobre un dispositivo reconfigurable le proporciona la capacidad de realizar rediseños, cambios o adaptaciones en pro de su optimización.

- La metodología de diseño descendente, Top-Down, es adecuada durante el proceso de diseño e identificación de los módulos en sistemas basados en FPGAs. Las herramientas de software apoyan en la realización automática de descripciones estructurales de mayor jerarquía, así como en la simulación y síntesis de los componentes individuales.

- La versatilidad en los FPGAs permite la creación de sistemas complementados con interfaces basadas en programas de computadora, en este caso, el sistema se completa con un software que permite la manipulación de imágenes y la comunicación con el puerto serial. El modelo incremental resulta conveniente para el desarrollo de aplicaciones de esta naturaleza, en donde el programa se va adecuando para ir cubriendo los diferentes requerimientos.

- En el FPGA, el uso de la metodología de diseño jerárquica permite implementar los bloques de hardware atómicamente, ya que es posible realizar pruebas de caja negra para evaluar la funcionalidad de cada uno de los bloques.

4.3 Trabajos futuros

El análisis de los resultados permite el planteamiento de mejoras sobre el sistema implementado, perspectivas de trabajos, aplicaciones y desarrollos a futuro en diversas líneas de investigación, tales como:

- Implementación de procesamiento de imágenes dentro del FPGA, que permita obtener características relevantes de las imágenes, para ser empleadas en otros procesos, o en sistemas de reconocimiento de patrones, formas o entornos.

- Empleo del sistema de visualización de imágenes para pantallas informativas, donde se integre un módulo y controlador de interfaz de red Ethernet, permitiendo a un software central proveer información relevante mediante imágenes del clima, programación televisiva, recordatorios, estado del tráfico, etc.

- La evaluación de técnicas que permitan la generación de un número mayor de colores, variando dinámicamente los colores básicos en un pixel.

Bibliografía

[1] Pérez, S., Soto, E. y Fernández, S.: Diseño de sistemas digitales con VHDL. THOMPSON, 2002.

[2] Nelson V., Troy H. e Irwin J., Análisis y diseño de circuitos lógicos digitales. Prentice Hall, 1996.

[3] Mandado, E., Álvarez J. y Váldes, M., Dispositivos lógicos programables y sus aplicaciones, THOMPSON, 2002.

[4] Grout, Ian: Digital Systems Design with FPGAs and CPLDs.ELSEVIER, 2008

[5] Floyd T.L.: Fundamentos de Sistemas Digitales. Prentice Hall, 1998.

[6] Zoran, Salsic and Smailagic, Asim: Digital Systems Design and Prototyping Using Field Programmable Logic and Hardware Description Languages, 2 ed., Kluwer Academic Publishers, 2002.

[7] Brown, Stephen, D., & Vranesic, Zvonko, G.: Fundamentals of Digital Logic with VHDL Design, 2 ed., MacGraw-Hill, 2004.

[8] Pardo, F. y Boluda, J.: VHDL, Lenguaje para síntesis y modelado de circuitos. Alfaomega, 3ª edición, 2000.

[9] Pedroni, Volnei A. : Circuit Design with VHDL, MIT Press, 2004

[10] Xilinx: Spartan-3 FPGA Family, Complete Data sheet, Xilinx, August, 2004.

[11] Jean-Pierre Deschamps, Gery Jean Antoine Bioul, Gustavo D. Sutter: Synthesis of arithmetic circuits: FPGA, ASIC and embedded systems, 2006.

[12] Berger A., Embedded systems design, An introduction to process, tools and techniques. CMP Books, 2002.

[13] Digilent: User Guide Digilent Spartan-3 Starter Kit Board Board, Digilent, April, 2004.

[14] Dallas Semiconductor: “Aplication Note 83”, 1998

[15] HOLTEK Semiconductor Inc: “Datasheet HT6523”, 2003.

[16] Rodríguez, O.: "Diseño de un controlador de video para procesamiento digital de imágenes basado en un FPGA", 1er Taller de Cómputo Reconfigurable y FPGAs dentro del ENC 2003, Apizaco, Tlaxcala, México, 2003.

[17] Álvarez, I., Bernabé, S., Robledillo, G. y Rodríguez, M.:"Procesado de imágenes digitales sobre un sistema hardware dinámicamente reconfigurable", Facultad de Informática, Universidad Complutense de Madrid, 2003.

[18] Fajardo, D., Rubén, R., Vicedo, F.: "Osciloscopio digital mediante FPGA", Dpto. Física y Arquitectura de Computadores, Universidad Miguel Hernández, 2004.

[19] Edwards, V., Courtney, M. y Yang, K.: "A FPGA Paint Brush Application", Information Systems Education Journal, Vol. 7, No. 36, Abril, 2009.

[20] Harris, M., Reingold E.: "Line Drawing, Leap Years, and Euclid. "ACM Computing Surveys (CSUR), Vol. 36, No. 1, pp. 68-80, 2004.

[21] Smith, A.: "Tint fill." ACM SIGGRAPH Computer Graphics, Vol. 13, No. 2, pp. 276-283, 1979.

[22] Chalmers, A., Cater, K., Maflioli, D.: "Visual attention models for producing high fidelity graphics efficiently", Proceedings of the 19th Spring Conference on Computer Graphics (SCCG'03), pp. 39-46, 2003.

[23] Obando, A., Ariza, M.: "Sistema de adquisición y transmisión de imágenes basado en FPGAs para prototipado de pico satélites tipo Cubesat" , 8th Latin American and Caribbean Conference for Engineering and Technology, Arequipa, Perú, 2010.

[24] Jarusauskas, A.: "FPGA based VGA driver and arcade game", School of Engineering & Design University of Sussex, 2010.

[25] González, M., Morales, L., Osornio, R. y Morales, L.: "IP core Genérico para Adquisición y Desplegado de Imágenes en Plataforma Basada en FPGA", 10° Congreso Nacional de Mecatrónica, Puerto Vallarta, Jalisco, México, 2011.

[26] Josh, H., Yong, B., Kleeman, L.: "A Real-time FPGA-based Vision System for a Bionic Eye", Monash Vision Group and Department of Electrical and Computer Systems Engineering, Monash University Wellington Road, Clayton, Australia, 2011.

Sitios de Internet

[URL1] http://www.achronix.com Página de la Coporación Achronix Semiconductor, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL2] http://www.actel.com Página de la Corporación Actel, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL3] http://www.altera.com Página de la Corporación Altera, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL4] http://www.atmel.com Página de la Corporación Atmel, fabricante de microcontroladores y dispositivos lógicos programables. Febrero 2011.

[URL5] http://www.cypress.com Página de la Compañia Cypress Semiconductor, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL6] http://www.latticesemi.com Página de la Compañía Lattice Semiconductor, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL7] http://www.quicklogic.com Página de la Compañía QuickLogic, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL8] http://www.xilinx.com Página de la Compañía Xilinx, fabricante de dispositivos lógicos programables. Febrero 2011.

[URL9] http://digilentinc.com/, Página de la Empresa Digilent, Enero 2012.

[URL10] http://www.lammertbies.nl/comm/info/RS-232.html, Página con especificaciones del estándar RS-232, Febreo-Marzo 2009.

[URL11] http://sloan.stanford.edu/MouseSite/MouseSitePg1.htmlhttp://www.computer-engineering.org/ps2protocol/, Página de la universidad de Stanford dedicada a la historia del mouse. Enero 2009

[URL12] http://www.computer-engineering.org/ps2protocol/, Página de Computer-Engineering de Adam Chapweske, USA, Enero 2009.

[URL13] http:// www.vesa.org, Página de la VESA, Video Electronics Standards Association, USA, Febrero 2011.

[URL14] http://commons.wikimedia.org/wiki/Main_Page, Página de la organización wikimedia Commons. Enero 2012.

[URL15] http://www.ece.cornell.edu, Página de la Escuela de Ingeniería Eléctrica y Computación (ECE, Electrical and Computer Engineering) de la Universidad de Cornell. Enero 2012.

Anexo A. Comandos Ratón PS/2

Comandos enviados de un Ratón PS/2

La Tabla A.1 muestra los comandos comúnmente enviados por un ratón PS/2 a un controlador anfitrión. Los comandos se encuentran listados en términos de sus códigos Hexadecimales. Todos los comandos son de un byte de longitud.

Tabla A.1 Comandos enviados de un Ratón PS/2

Código	*Descripción*
00h	Identificación del Ratón (*Mouse ID*)
AAh	Prueba básica de integridad Post Alimentación exitosa
FCh	Prueba básica de integridad Post Alimentación fallida
FAh	Acuse de recibo de Comando
FEh	Reenvío. Tras la recepción de este código, el controlador PS2 retransmite el byte previo.

Comandos enviados del Controlador al Ratón PS/2

La tabla A.2 lista los comandos que se pueden enviar de un controlador anfitrión a un ratón PS/2. Se enumeran en términos de sus códigos hexadecimales. Todos los comandos son de un byte de longitud, pero muchos requieren enviar al anfitrión un byte adicional de datos cuando determinan opciones específicas.

Tabla A.2 Comandos de Control

Código	*Descripción y Acción del ratón*
E6h	Este comando se emplea para fijar la escala 1:1. Después de la recepción del comando, el ratón responde con el código de reconocimiento (FAh) y habilita la escala 1:1.
E7h	Este comando se emplea para fijar la escala 2:1. Después de la recepción del comando, el ratón responde con el código de reconocimiento (FAh) y fija la escala 2:1.
E8h	Este comando se emplea para fijar la resolución del ratón. El ratón envía el código de reconocimiento (FAh) y después espera por otro byte de datos del

	anfitrión, en el cuál se especifica la resolución a ser empleada: 00h – 1 incremento por milímetro 01h – 2 incrementos por milímetro 02h – 4 incrementos por milímetro 03h – 8 incrementos por milímetro. Después de la recepción de este segundo byte, el ratón envía nuevamente el código de reconocimiento y reinicia los contadores de movimiento.
E9h	Este comando se emplea para solicitar el estado actual del ratón. Después que recibe el comando, el ratón envía el código de reconocimiento (FAh), seguido por el paquete de estado en tres Bytes, estructurado de la siguiente forma: *Byte 1* Bit0 – Estado del botón derecho del ratón (1=presionado; 0=no presionado) Bit1 – Estado del botón medio del ratón (1=presionado; 0=no presionado) Bit2 – Estado del botón izquierdo del ratón (1=presionado; 0=no presionado) Bit3 – No empleado (fijado a 0) Bit4 – Escala actual (1=2:1; 0=1:1); Bit5 – Estado del reporte de datos (1=habilitado; 0=deshabilitado) Bit6 – Modo actual (1=modo remoto; 0=modo *stream*) Bit7 – No empleado (fijado a 0) Byte 2 – Resolución actual Byte 3 – Tasa de muestreo actual Después de enviar el paquete de estado, el ratón reinicia los contadores de movimiento.
EAh	Este comando se emplea para fijar al ratón en modo *Stream*. El ratón responde enviado el código de reconocimiento (FAh), reinicia los contadores de movimiento y habilita el modo *stream*.
EBh	Este comando se emplea para leer la tasa de muestreo del ratón mientras se encuentra en *modo remoto*. El ratón responde enviando el código de reconocimiento (FAh) y el paquete actual de los datos de movimiento. Los contadores de movimiento se reinician posteriormente.
ECh	Este comando se emplea para reiniciar el modo *wrap*. El ratón responde enviando el código de reconocimiento (FAh), reinicia los contadores de movimiento y regresa al modo en que se encontraba antes de habilitar el modo el Wrap (remoto o *stream*)
EEh	Este comando se emplea para fijar el ratón en modo *wrap*. El ratón responde enviando el código de reconocimiento (FAh), reinicia los contadores de movimiento y entra en modo *wrap*.
F0h	Este comando se emplea para fijar el ratón en modo remoto. El ratón responde enviando el código de reconocimiento (FAh), reinicia los contadores de movimiento y habilita el modo remoto.
F2h	Este comando se emplea para leer el ID (identificador) del ratón. El ratón responde con el código de reconocimiento (FAh), seguido por el código que representa su identificador y lo distingue como un ratón estándar PS/2 – 00h;

	los contadores de movimiento también son reiniciados en este momento.
F3h	Este comando es empleado para fijar la tasa de muestreo para el ratón. El Mouse envía en respuesta el código de reconocimiento (FAh) y entonces espera del anfitrión otro byte de datos especificando la tasa de muestreo a emplear, como sigue: 0Ah – 10 muestras por segundo. 14h – 20 muestras por segundo. 28h – 40 muestras por segundo. 3Ch – 60 muestras por segundo. 50h – 80 muestras por segundo. 64h – 100 muestras por segundo. C8h – 200 muestras por segundo. Después de la recepción de este segundo byte, el ratón envía nuevamente el código de reconocimiento y reinicia los contadores de movimiento.
F4h	Este comando se emplea para habilitar el reporte de datos cuando el ratón se encuentra en modo *Stream*. Tras la recepción del comando, el ratón responde enviando el código de reconocimiento (FAh) y reinicia los contadores de movimiento.
F5h	Este comando se emplea para deshabilitar el reporte de datos cuando el ratón se encuentra en modo *Stream*. Una vez recibido el comando, el ratón responde enviando el código de reconocimiento (FAh) y reinicia los contadores de movimiento.
F6h	Este comando se emplea para cargar el ratón con los valores por defecto. Una vez reconocido el comando, el ratón responde con el código de reconocimiento (FAh) y entonces es inicializado con lo siguiente: Tasa de muestreo – 100 muestras por segundo. Resolución – 4 incrementos por milímetro. Escala – 1:1 Reporte de datos – Deshabilitado Después de cargar los valores, el ratón reinicia los contadores de movimiento y entra en modo *Stream*.
FEh	Este es el comando de reenvío y se emplea cuando el anfitrión requiere que el ratón retransmita el último paquete de los datos enviados. Después de reconocer el comando, el ratón transmite el paquete enviado previamente.
FFh	Este comando es empleado para reiniciar el ratón. Después de que el comando es identificado, el ratón se reinicia y posteriormente entra en modo de reset.

Anexo B. Diagrama PCB Tarjeta Externa de Memoria NVRAM