UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Diseño de controladores para la tarjeta de desarrollo XSA-100

Tesis para obtener el título de

Ingeniero en Electrónica

Presenta

Juan Carlos Tepozán Ríos

Director de tesis

M.C. Felipe Santiago Espinosa

Huajuapan de León, Oaxaca.

Mayo 2004

DEDICATORIAS

Dedico este trabajo a mi madre por el esfuerzo que sólo ella pudo realizar.

A mi padre que siempre me brindó su apoyo incondicional.

A mis hermanos, que aprendimos y crecimos juntos.

A Luis que siempre estuvo presente.

A mis maestros, por formar parte de mi educación en la Universidad.

A mis amigos, sin olvidar a Sara.

Y especialmente a Janet.

AGRADECIMIENTOS

A mis padres por el esfuerzo y paciencia.

A la Universidad Tecnológica de la Mixteca por abrirme sus puertas.

A mi director de tesis, M.C. Felipe Santiago Espinosa por su apoyo.

A mis sinodales por su valiosa aportación a este trabajo.

CONTENIDO

INTRODUCCIÓN

La evolución de los dispositivos lógicos programables ha dado como resultado que éstos funcionen a frecuencias mayores y cuenten con una mayor cantidad de compuertas. Estos dispositivos tienen la capacidad de programarse para realizar una aplicación específica.

En la actualidad existen tarjetas de prototipado de sistemas digitales que emplean dispositivos lógicos programables para realizar, como su nombre lo indica, el prototipo de un sistema digital específico. Las empresas emplean estas tarjetas para probar los diseños de sus circuitos antes de sacarlos al mercado, sin embargo, también se utilizan en aplicaciones diversas independientes del diseño de circuitos lógicos comerciales.

En el mercado existe una gran variedad de tarjetas de prototipado de sistemas digitales que cuentan con algún dispositivo lógico programable y componentes adicionales como por ejemplo, memoria de acceso aleatorio, display de siete segmentos, microcontrolador 8051, conector de teclado, interruptores, etc.

Para algunos componentes de las tarjetas de prototipado de sistemas digitales, se requiere de controladores que configuren al dispositivo lógico programable de éstas, con el objeto de que se puedan emplear en aplicaciones diversas.

Para la programación de los dispositivos lógicos programables en las tarjetas de prototipado de sistemas digitales, se utiliza una computadora en la cual el diseñador realiza las especificaciones del sistema digital para, posteriormente, obtener un archivo que se descargue en el dispositivo lógico programable.

OBJETIVO GENERAL

La tarjeta de prototipado empleada en esta tesis es la XSA-100 y cuenta entre sus componentes con conectores de teclado y de video, sin embargo, para que éstos se puedan utilizar deben contar con controladores que los hagan funcionar de manera adecuada.

El objetivo del presente trabajo es desarrollar los controladores de teclado y de video, útiles en situaciones en que se requiera crear una interfaz de estos dispositivos con el usuario en la tarjeta XSA-100, y desarrollar un ejemplo que pruebe la funcionalidad de tales controladores.

Los controladores serán desarrollados en VHDL con el software Xilinx Foundation F3.i y se emplean para programar al arreglo de compuertas programable en campo (FPGA, Field Programmable Gate Array) que contiene la tarjeta.

METODOLOGÍA

Para el ejemplo que emplea los controladores de teclado y de video, se propone el planteamiento de un *sistema de consulta de calificaciones de alumnos*, donde un computador personal contendrá una base de datos basada en las carreras de la Universidad Tecnológica de la Mixteca, ésta tiene como propósito el contener las calificaciones de alumnos para que éstos puedan realizar una consulta.

Para ello, se emplea un protocolo de comunicación serial multipunto, en la cual el computador personal actuará como maestro y cada nodo esclavo (donde el alumno solicita la información al proporcionar su código y semestre en el que está interesado) se implementará en una tarjeta XSA-100 con la interfaz de teclado y video realizada por los controladores desarrollados.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos de esta tesis son:

- Desarrollar el controlador de teclado para el FPGA de la tarjeta XSA-100.
- Desarrollar el controlador de video alfanumérico (muestra caracteres en pantalla) para el FPGA de la tarjeta XSA-100.
- Desarrollar los módulos necesarios empleados en el protocolo de comunicación serial para comunicar los nodos esclavos con el nodo maestro.
- Acondicionar los controladores para que la tarjeta se comporte como un nodo esclavo en un *sistema de consulta de calificaciones*.
- Crear un programa para la PC que realice las funciones del nodo maestro para la comunicación con los nodos esclavos.

ORGANIZACIÓN DEL DOCUMENTO

El documento comprende 6 capítulos, el contenido de cada uno de ellos se describe a continuación:

Capítulo 1. Contiene la teoría de los diferentes elementos utilizados en el desarrollo del presente proyecto. Se da una breve descripción sobre VHDL, la tarjeta XSA-100, Modbus sobre línea serial (el protocolo de comunicación serial multipunto empleado), las señales de teclado y las señales de video.

Capítulo 2. Describe los módulos de interfaz con el usuario, es decir, los módulos controladores de teclado y de video en VHDL.

Capítulo 3. Describe los módulos en VHDL que realizan la comunicación serial en el nodo esclavo, así como la manera en que se relacionan.

Capítulo 4. Explica el módulo que realiza los accesos a memoria y la instanciación de componentes en la tarjeta, es decir, la interconexión de todos los módulos desarrollados.

Capítulo 5. Describe el funcionamiento del nodo maestro para que establezca comunicación con los nodos esclavos.

Capítulo 6. Está dedicado a las conclusiones y la descripción de los resultados obtenidos con ilustraciones del ejemplo planteado.

1. MARCO TEÓRICO

En este capítulo se ofrece una breve introducción acerca de los dispositivos lógicos programables y sobre VHDL, lenguaje en el que se diseñaron los controladores de la tarjeta de trabajo, así como los componentes más importantes de ésta. Para comprender el desarrollo del controlador de teclado, se explica el proceso de envío y recepción de datos en el teclado. En el caso del controlador de video, se explica el proceso para producir las señales necesarias para generar una imagen en un monitor. Finalmente se describe el protocolo de comunicación serial empleado para realizar la comunicación de las tarjetas.

1.1 HISTORIA DE LOS DISPOSITIVOS LÓGICOS PROGRAMABLES

Anteriormente para implementar una aplicación de un sistema digital se tenía que realizar la interconexión de varios circuitos lógicos integrales. Esto no era problema para aplicaciones sencillas, pero con la creciente necesidad de realizar diseños más complejos surgió un gran problema: se requería de un mayor número de circuitos integrales. Esto no sólo representaba un incremento en el costo de construcción del circuito, sino que se incrementó la dificultad para realizar los diseños, debido a que un pequeño error en la interconexión de los componentes empleados implicaba que toda la aplicación fallara. Buscar el error requería la revisión por partes del circuito. Por si fuera poco, los dispositivos deberían ser impresos y soldados en tarjetas. La suma de todos estos factores resultó en el empleo de mayor tiempo por parte del diseñador. Pero eso no era todo, los diseños no podían ser reconfigurados, es decir, sólo resultaban útiles para una aplicación específica.

Ante todas las desventajas mencionadas, en los setentas surgió una solución: la creación de circuitos digitales integrados programables por el usuario.

El dispositivo programable más antiguo (1970) es la memoria programable de sólo lectura (PROM, Programmable Read Only Memory) la cual consta de un arreglo (o matriz) AND fijo y un arreglo OR programable (figura 1.1) [1, URL1]. El arreglo AND genera 2^n

productos de *n* entradas, el arreglo OR permite incluir cualquier combinación de términos producto en cada término suma.

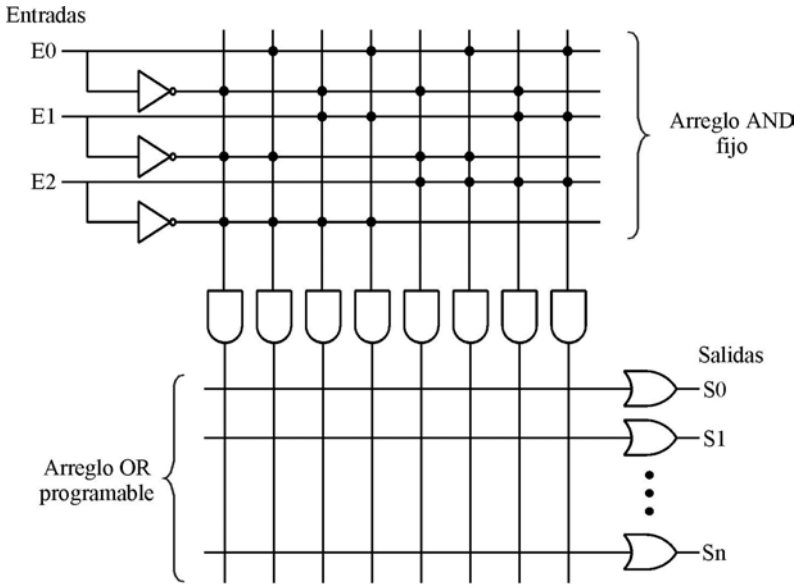

Figura 1.1. Arreglo AND fijo y OR programable de una PROM [1].

En cada línea vertical del arreglo AND fijo de la figura 1.1, el símbolo ● representa la entrada que forma parte del producto que realiza la compuerta AND. Las líneas horizontales del arreglo OR programable representan las salidas que se pueden programar para un producto de entradas realizadas por cada compuerta AND.

A mediados de la década de 1970 surgen los arreglos lógicos programables (PLA, Programmable Logic Array) en los cuales se tiene un arreglo AND programable seguido de un arreglo OR programable, éstos pueden configurarse para realizar operaciones lógicas AND y OR [1].

La figura 1.2 muestra los 2 arreglos programables de un PLA en el cual, a manera de ejemplo, el símbolo x representa la selección de entradas programadas que forman parte del producto realizado por la compuerta AND, así como la suma de productos realizados por cada compuerta OR.

A finales de la década de 1970 surge el arreglo lógico AND programable (PAL, Programmable Array Logic) mejorando los tiempos de retardo del PLA [URL1, URL2]. El PAL consta de una arreglo AND programable y un arreglo OR fijo (figura 1.3).

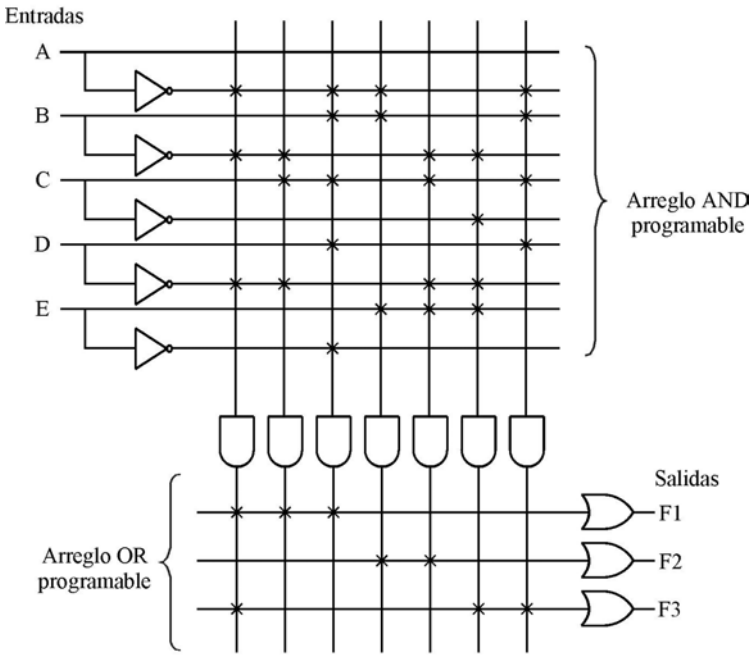

Figura 1.2. Arreglos AND y OR programables de un PLA [1].

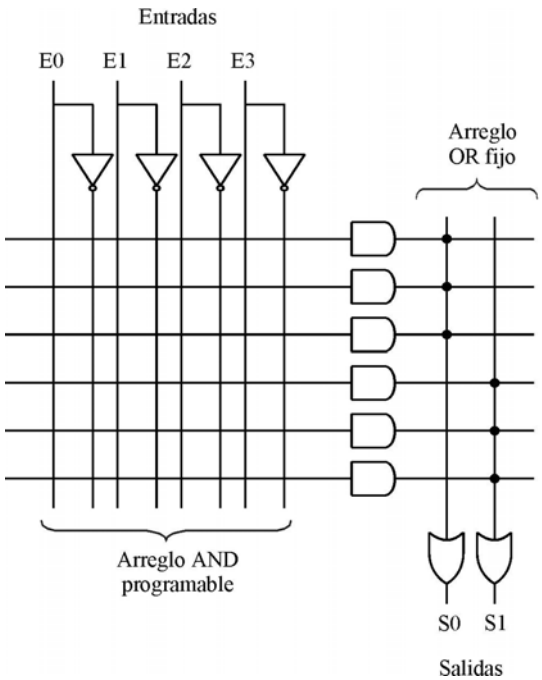

Figura 1.3. Arreglos AND y OR del PAL [1].

Para diseños lógicos de mayor complejidad los fabricantes de circuitos integrados desarrollaron los circuitos lógicos configurables (CLC). Éstos se caracterizan por:

- Contar con recursos lógicos divididos en bloques.
- Contar con recursos de interconexión entre los bloques lógicos.

El desarrollo de CLC está basado en dos tendencias (en la terminología anglosajona a los dispositivos que se basan en estas tendencias se les denomina dispositivos lógicos programables) [2]:

- A partir de matrices lógicas PAL y PLA con un registro de entrada y salida, se dio lugar a los secuenciadores lógicos programables (PLS, Programmable Logic Secuencer) mismos que son la base de los dispositivos lógicos programables (PLDs, Programmable Logic Devices). Los PLDs cuentan con recursos de interconexión concentrados o de organización matricial (figura 1.4) y, dependiendo de los elementos lógicos que los conforman, pueden ser: básicos, avanzados y complejos (CPLD).

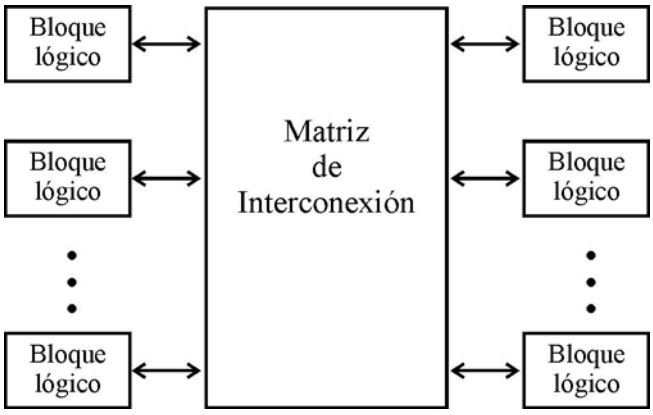

Figura 1.4. Organización matricial de los PLDs [2].

- A partir de las arquitecturas empleadas en los circuitos integrados digitales a semimedida, los cuales se caracterizan por contar con bloques funcionales que pueden ser configurados mediante las interconexiones entre ellos, se dio lugar al arreglo de compuertas configurable en campo (FPGA, Field Programmable Gate Array). Un FPGA está formado por 3 elementos básicos [2]:

- Bloques lógicos configurables (CLBs, Configurable Logic Blocks). Son recursos lógicos que realizan diferentes funciones lógicas. Algunos pueden tener memorias de acceso aleatorio llamadas tablas de consulta (LUT, Look-Up Table).
- Bloques lógicos de entrada y salida (IOBs, Input/Output Blocks). Son recursos lógicos que establecen el enlace entre los CLBs y las terminales de entrada y salida.
- Recursos de interconexión. Son líneas e interruptores programables que permiten transmitir las señales entre los CLBs y los IOBs.

La figura 1.5 muestra los elementos que conforman un FPGA.

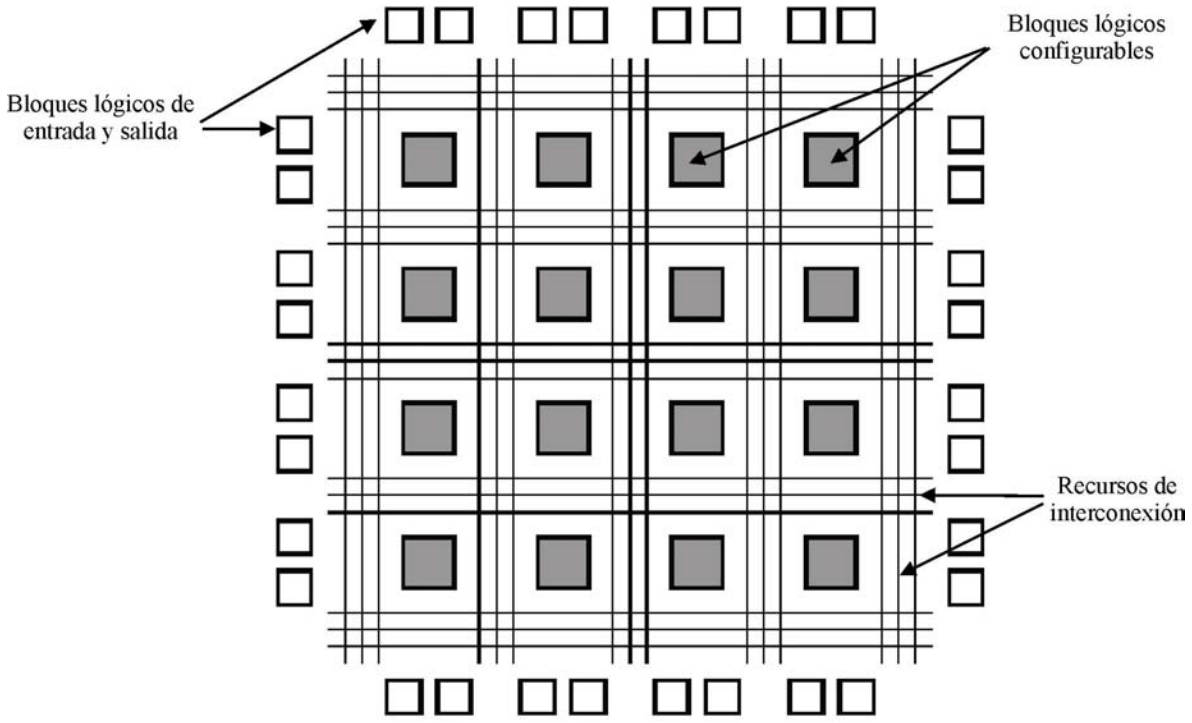

Figura 1.5. Elementos que contiene un FPGA [URL1].

1.2 VHDL

Con la aparición de los dispositivos lógicos programables surgen lenguajes para realizar la descripción de los sistemas digitales [2]. Éstos reciben el nombre de lenguajes de descripción de hardware (HDL, Hardware Description Language) y son de 2 tipos:

- No estructurados. Se caracterizan porque están orientados a la realización de un circuito o módulo por archivo.

- Estructurados. Permiten definir submódulos y enlazarlos jerárquicamente en un único fichero, además del uso de bibliotecas de circuitos. Algunos de estos lenguajes son: Verilog, Abel y VHDL.

Dado que la programación del FPGA de la tarjeta de desarrollo del presente trabajo se realiza con VHDL, en adelante la información está orientada a este lenguaje.

Las siglas de VHDL en inglés se traducen como *lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad*. El acrónimo se desprende de:

VHDL = VHSIC *Hardware Description Language*
VHSIC = *Very High Speed Integrated Circuits*

La primera versión de VHDL apareció en 1987 como Std 1076. Su propósito inicial fue el de crear diseños para la simulación, modelado[1] y documentación, sin embargo, con el paso del tiempo se ha colocado como una herramienta importante en el proceso de síntesis[2] de circuitos [3].

1.2.1 Entidad, arquitectura y paquetes

Los diseños en VHDL cuentan con 3 partes principales: la entidad, arquitectura y los paquetes (para mayor información puede consultar las referencias [4, URL3]).

Todo sistema digital cuenta con entradas que le indican que debe realizar una acción y con salidas que representan el resultado de las entradas aplicadas. Las entradas y salidas del sistema se declaran en VHDL en la parte de la entidad, con ella es posible realizar la interfaz de un sistema con otro.

La arquitectura, es la parte que describe el comportamiento del sistema con base en la entidad, es decir, describe funciones que dependen de las entradas de la entidad.

[1] Desarrollo de un modelo para la simulación de un circuito o sistema previamente implementado, por lo tanto se conoce [1].
[2] Proceso de diseño de un circuito digital a partir de cierta especificación inicial de un problema [1].

VHDL cuenta con bibliotecas estándar, sin embargo, si el diseñador requiere de descripciones externas a dichas bibliotecas puede crear paquetes.

La figura 1.6 ilustra la relación existente entre la entidad, arquitectura y los paquetes.

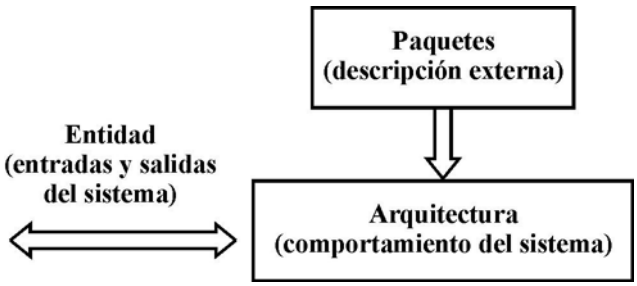

Figura 1.6. Relación de la entidad, arquitectura y paquetes en VHDL.

1.2.2 Señales

Las señales se emplean para comunicar bloques o circuitos, representan niveles lógicos y pueden ser individuales o buses.

Las señales pueden ser externas (declaradas en la entidad) o internas (declaradas en la arquitectura). Las señales externas se deben especificar de acuerdo a su modo como: entradas, salidas o bidireccionales.

1.2.3 Procesos

El comportamiento de un sistema puede ser descrito en los procesos. Los procesos se encuentran en la parte de la arquitectura. Éstos son concurrentes, por lo que, si se tienen varios procesos, todos se ejecutarán al mismo tiempo.

Los procesos cuentan con una lista de sensitividad, es decir, una lista de todas las señales que activan su ejecución [4], por ejemplo, al cambio de nivel de una señal (como en la ejecución de reset) o en el flanco de subida de la señal de reloj. A diferencia de lenguajes como C o Pascal, los procesos en VHDL nunca terminan. Cuando un proceso se ejecuta, está pendiente en las señales que forman parte de su lista de sensitividad.

1.2.4 Implementación de un diseño lógico con VHDL en la tarjeta XSA-100

Para crear e implementar un diseño lógico con VHDL se requiere de una serie de pasos. Al término de éstos, se obtiene un archivo para descargar en un FPGA o CPLD con el objeto de programarlo. Así, el programador debe [URL4]:

- Conocer los requerimientos del diseño a crear. Este punto abarca el conocimiento del problema, entradas y salidas del circuito lógico.
- Emplear VHDL para introducir la descripción de un circuito lógico. Con esto se describe el comportamiento del diseño. Para traducir el diseño descrito en VHDL en compuertas se debe realizar un proceso llamado síntesis.
- Para las síntesis se utiliza un programa sintetizador lógico (logic synthesizer) con el objeto de transformar el diseño descrito en HDL en una lista de conexiones (netlist). La netlist es un texto equivalente al circuito, es una forma compacta que describe qué compuertas se encuentran en el circuito, cómo se conectan y los nombres de las terminales de entrada y salida. Sin embargo, la netlist no describe el funcionamiento del circuito.
- Emplear herramientas de implementación (implementation tools) para trazar las compuertas lógicas y las interconexiones en el FPGA. Los CLBs en el FPGA pueden ser además descompuestos en LUTs que presentan operaciones lógicas. Los CLBs son interconectados con recursos de ruteo (routing resources). La herramienta de mapeo (mapping tool) toma las compuertas de la netlist en grupos que pueden ajustarse a las LUTs y entonces la herramienta de colocación y ruteo (place & route tool) asigna las compuertas a CLBs específicos mientras que abre o cierra los interruptores de las matrices para conectar las compuertas.
- Una vez que la fase de implementación está completa, un programa extrae el estado de los interruptores y genera los bits donde los unos y ceros corresponden a interruptores abiertos o cerrados.
- Los bits generados son cargados al FPGA. Los interruptores del FPGA se abren o cierran en respuesta de los bits que se cargaron. Hecho esto, el FPGA realizará las operaciones especificadas por el código HDL. Se pueden así aplicar señales de entrada a las terminales de entrada/salida del FPGA para verificar la operación deseada del diseño original.

La figura 1.7 muestra el flujo de los pasos a seguir para la implementación de un diseño lógico con VHDL, también indica qué pasos se implementan por las herramientas de diseño asistido por computadora (CAD, Computer Aided Design), término asociado al software encargado de realizar de manera automática todos o algunos pasos de la síntesis [1].

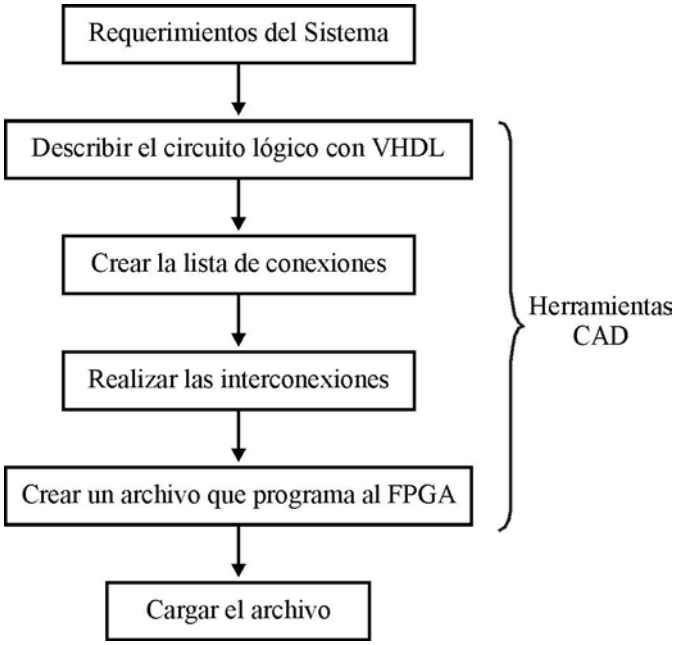

Figura 1.7. Pasos en la implementación de un diseño lógico con VHDL.

1.2.5 Xilinx Foundation

El software empleado en el presente trabajo es Xilinx Foundation 3.1i, éste cuenta con las herramientas necesarias para cubrir los pasos descritos en el apartado anterior y así obtener un archivo con extensión .bit el cual configura al FPGA. Para mayor información acerca de este software puede consultar la referencia [URL5].

1.3 COMPONENTES DE LA TARJETA XSA-100

La tarjeta de trabajo empleada (figura 1.8) cuenta con varios componentes que la hacen atractiva para desarrollar un diseño lógico, éstos son [URL6]:

- FPGA XC2S100 Spartan-II. Es el componente principal de la tarjeta, consta de 100 mil compuertas encapsuladas en un paquete QFP de 144 terminales.
- CPLD XC9572XL. Se usa para configurar al FPGA mediante el puerto paralelo o con la RAM tipo Flash.
- Oscilador Dallas DS1075. Es un oscilador programable de 100 Mhz que introduce una señal de reloj al FPGA y al CPLD. Su frecuencia máxima es de 100 Mhz sin embargo, la frecuencia por defecto de la tarjeta XSA-100 es de 50Mhz.
- Atmel AT49F002. Es una RAM tipo Flash que cuenta con una capacidad de almacenamiento de 256 KBytes (256K x 8) y está conectada al FPGA y al CPLD.
- SDRAM. Memoria de 16MBytes (8M x 16) que ofrece almacenamiento volátil de datos accesibles por el FPGA.
- Display de 7 segmentos. Permite visualizar algunos resultados de la tarjeta. Sin embargo, no se puede hacer uso de él cuando se utiliza la RAM tipo Flash debido a que comparten algunos pines del FPGA.
- Interruptor DIP. Configura la tarjeta XSA o controla los bits más significativos de la dirección de la RAM tipo Flash
- Pushbutton. Es un interruptor que envía información momentánea al FPGA.
- Puerto paralelo. Es la principal interfaz para configurar la tarjeta XSA por medio de un computador personal.
- Conector PS/2. Se utiliza para conectar un teclado o ratón.
- Conector VGA. Sirve para desplegar gráficos en el monitor.

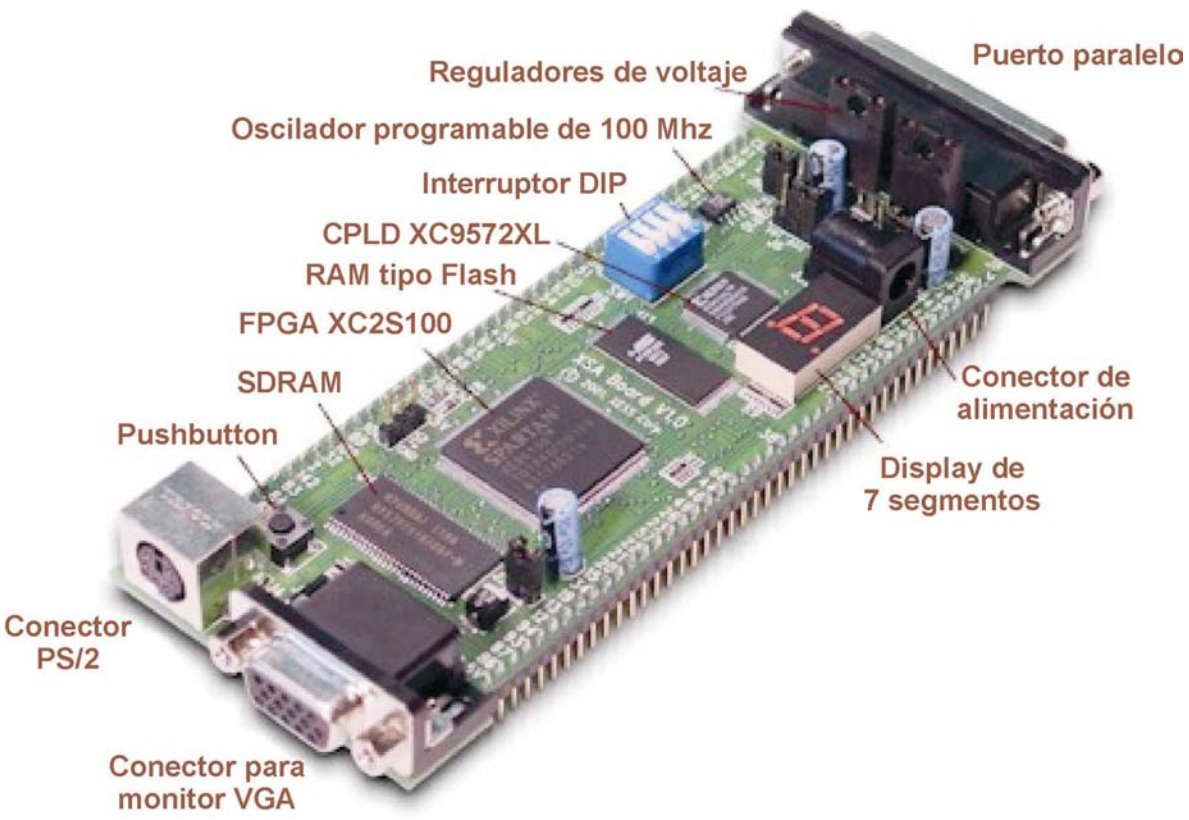

Figura 1.8. Tarjeta XSA-100 [URL6].

La tabla 1.1 muestra algunas de las tarjetas de XESS (la compañía en la cual se adquirió la tarjeta de desarrollo empleada en este trabajo). La tarjeta XSA-100 se encuentra, con respecto a la cantidad de compuertas que contiene el dispositivo que se programa en ella (CPLD o FPGA), en un rango relativo intermedio.

Tarjeta	**Cantidad de compuertas en el CPLD o FPGA**
XS95-108, XS95-108+	2,400
XSTE5	5,000
XS40-005XL, XS40-005XL+, XS40-005E, XS40-005E+	9,000
XSP-010, XSP-010+	10,000
XS40-010XL, XS40-010XL+, XS40-010E, XS40-010E+	20,000
XSV-50	57,906
XSA-100	100,000
XSV-100	108,904
XSV-300	322,970
XSV-800	888,439

Tabla 1.1. Tarjetas de XESS [URL7].

1.3.1 Características de los FPGAs Spartan II

La figura 1.9 muestra un diagrama a bloques del FPGA Spartan II.

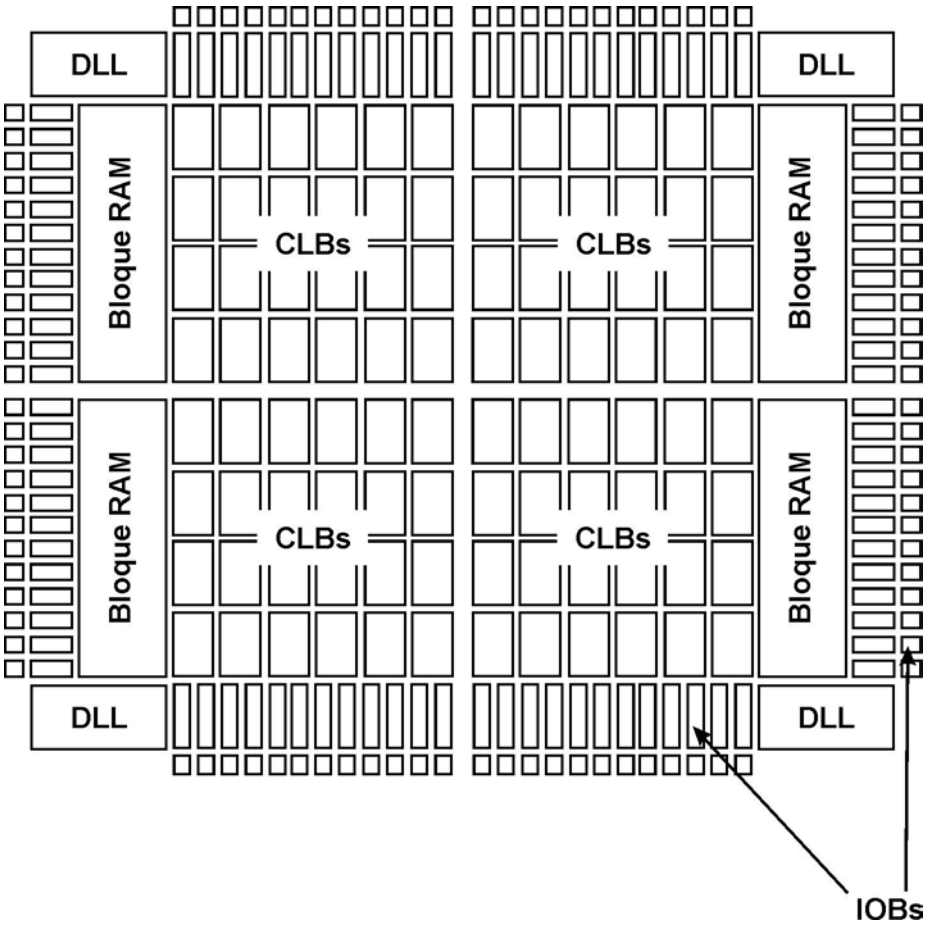

Figura 1.9. Diagrama a bloques del FPGA Spartan II [URL8].

Como se puede observar, la arquitectura de los FPGAs Spartan II se basa en CLBs, cada uno de ellos está formado por 2 pedazos (slices). Cada slice, a su vez, contiene 2 LUTs de 4 entradas y elementos de almacenamiento que pueden ser configurados ya sea como flip-flip D o como latch SR. La figura 1.10 muestra el diagrama a bloques de un slice.

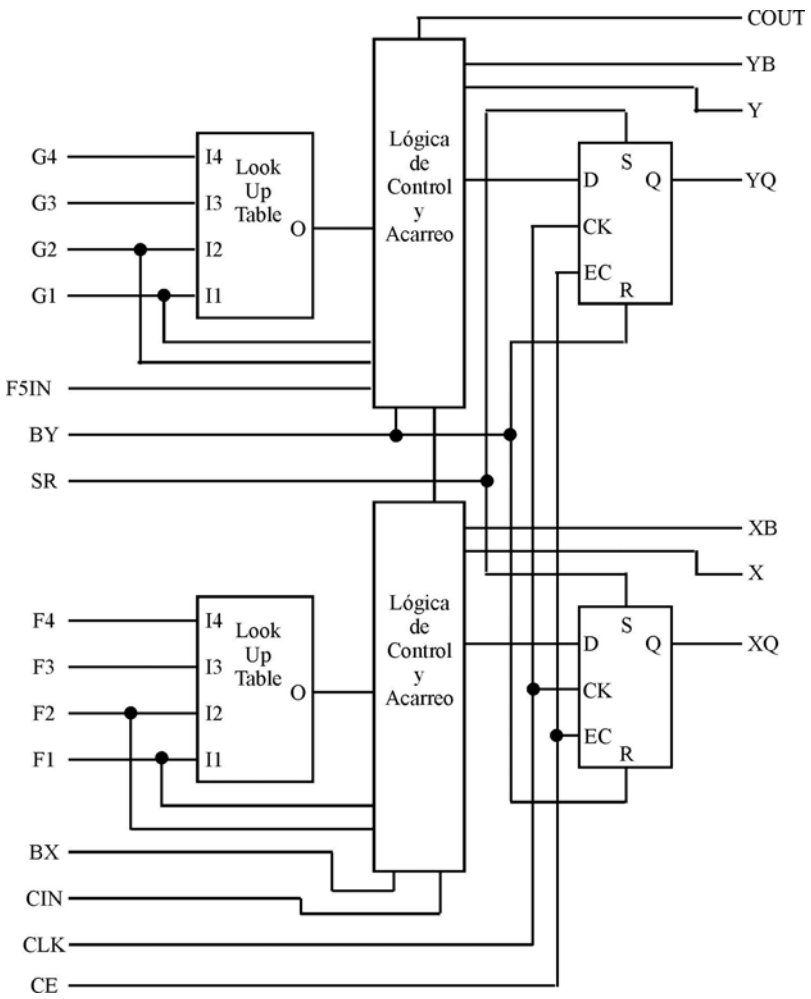

Figura 1.10. Slice de un CLB del FPGA Spartan II [URL8].

Otras características importantes del FPGA Spartan II son:

- Bloques de entrada y salida que se encuentran alrededor del FPGA.
- Bloques de memoria RAM, ubicados en columnas opuestas del FPGA que sirven para el almacenamiento y lectura de datos. En el FPGA XC2S100 de la tarjeta XSA, hay diez bloques de memoria RAM interna de 512 bytes cada uno.
- Cuatro lazos de seguimiento de retardo (DLLs, Delay-Locked Loops) se ubican en cada esquina del chip. Ofrecen un retardo de propagación de cero con lo que permiten la sincronización de las señales de reloj interna y externa. Por ejemplo, al emplear al FPGA en conjunción con dispositivos que tienen una entrada de señal de reloj, como la SDRAM. Además, ofrecen múltiples fases de la señal de reloj principal. Para mayor información acerca del uso de DLLs en el FPGA Spartan II puede consultar [URL9].

La tabla 1.2 muestra las características del FPGA XC2S100.

Dispositivo	Total de compuertas	Arreglo CLB	Total de CLBs	Cantidad de entradas y salidas disponibles	Total de bits en los bloques de memoria
XC2S100	100,000	20x30	600	196	40k

Tabla 1.2. Características del FPGA XC2S100 [URL8].

1.3.2 Programación de la tarjeta XSA-100

XESS ofrece el software llamado XSTOOLs que se encarga de configurar al CPLD de la tarjeta XSA-100 con el objeto de realizar la interfaz necesaria para:

- Realizar la prueba de que la tarjeta está en buenas condiciones, con alimentación de voltaje correcta y conectada al puerto paralelo de la computadora personal.
- Configurar la frecuencia del oscilador programable.
- Almacenar datos desde la PC a la SDRAM o la RAM tipo Flash.
- Programar al FPGA.
- Almacenar diseños en la RAM tipo Flash y, posteriormente, crear un circuito que programa al FPGA con el contenido de la RAM tipo Flash cada vez que la tarjeta se alimenta de energía, es decir, programa al FPGA sin necesidad de la PC. Esto es útil cuando se tiene el diseño final.

Para mayor referencia de la programación de la tarjeta XSA-100 puede consultar [URL6].

1.4 TECLADO

El teclado es uno de los principales dispositivos periféricos para introducir datos en un sistema. Básicamente un teclado realiza dos funciones, para ejemplificarlas se considerará que el teclado está conectado a un sistema al que se le llamará controlador (por ejemplo una PC), así las funciones que realiza son [URL10]:

- *Envío de códigos*. Cuando un usuario presiona una tecla, el teclado envía un código (correspondiente a la tecla presionada) al controlador, éste recibe el código y debe decidir si realizar una acción o permanecer sin cambios.
- *Ejecución de órdenes*. Cuando el controlador desea que el teclado realice una acción como, por ejemplo, encender sus LEDs, éste debe enviar una orden al teclado para que la ejecute.

1.4.1 Terminales del teclado

Para que el teclado realice las funciones mencionadas, a pesar de contar con más terminales, sólo utiliza cuatro de ellas:

- *Terminales de alimentación y de tierra*. Sirven para suministrar voltaje al teclado, el controlador debe proporcionar el voltaje (5Vcd) y tierra.
- *Terminal de datos*. Se emplea para enviar y recibir datos, es decir, de manera bidireccional. Cuando el teclado se conecta al controlador, y no hay datos a enviar o recibir, normalmente a esta terminal se le aplica un nivel lógico alto. Para el intercambio de datos se emplea un protocolo serial, cuando el teclado o el controlador envían datos, pueden forzar a que la línea de datos se ponga en un nivel lógico bajo si se transmite un ‘0’ o mantener el nivel lógico alto si se transmite un ‘1’.
- *Terminal de reloj*. En el envío y recepción de datos, el teclado cuenta con una señal de reloj para la transmisión serial previamente mencionada. La frecuencia máxima de la señal reloj es de 33 kHz, sin embargo, la mayoría de los dispositivos operan dentro del rango de los 10 kHz a 20 kHz. Al igual que en la terminal de datos, la señal de reloj se encuentra en un nivel lógico alto.

La figura 1.11 muestra un diagrama a bloques de la conexión de la terminal de datos y de reloj con el controlador, nótese que a las líneas de datos y de reloj se les aplica un voltaje que las mantiene en un nivel lógico alto.

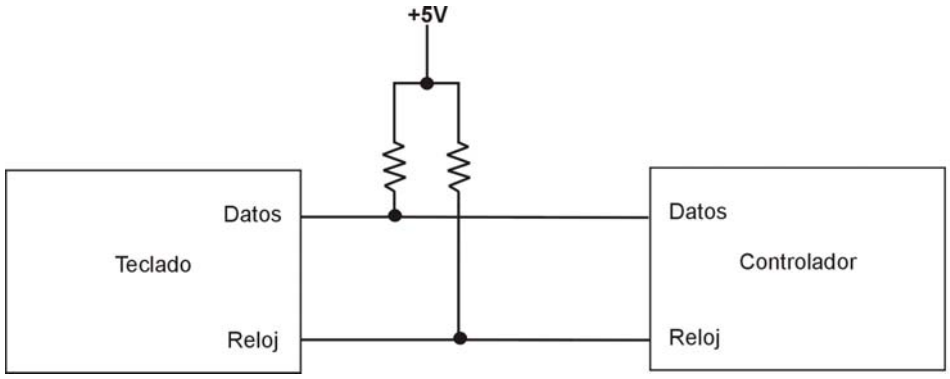

Figura 1.11. Conexión de la terminal de teclado con el controlador.

1.4.2 Envío de códigos

Se había mencionado previamente que al presionar una tecla, el teclado envía un código, en realidad existen 2 tipos de códigos:

- ***Código de Rastreo***. El código de rastreo es la representación de la tecla presionada. Cada tecla tiene un código de rastreo único. Si el usuario mantiene presionada la tecla, el teclado se mantiene enviando al sistema controlador una serie de códigos de rastreo hasta el momento en que se suelta la tecla. El código de rastreo no es código ASCII, por lo que, si el controlador necesita el código ASCII de una tecla presionada, debe capturar el código de rastreo para, posteriormente llevar a cabo la conversión de código de rastreo a código ASCII [5].

- ***Código de Liberación***. Al momento de soltar la tecla presionada, el teclado envía al sistema controlador el código de liberación. Con éste, el controlador se entera de que se liberó la tecla presionada.

Con los códigos de rastreo y de liberación un programa puede determinar si realizar una acción (por ejemplo, imprimir un carácter en la pantalla del monitor) cuando se presiona una tecla o cuando se suelta [6].

Existen 3 conjuntos de códigos de rastreo [URL10]:

- *Conjunto de Códigos de Rastreo 1*: Utilizado en el teclado XT de IBM, el cual consta de 83 teclas, este teclado es obsoleto.
- *Conjunto de Códigos de Rastreo 2*: Este conjunto es el más usado por los teclados modernos. Se emplea en teclados AT o PS/2.
- *Conjunto de Códigos de Rastreo 3*: Esta representación de códigos es opcional. Se creó para los teclados PS/2 con el objeto de soportar algunas órdenes adicionales al conjunto de códigos de rastreo 2. Este conjunto es el que se empleará en el presente trabajo debido a que, a diferencia del conjunto de códigos de rastreo 2, todos sus códigos de rastreo son de 1 byte y sus códigos de liberación de 2 bytes, con lo cual resulta más fácil el desarrollo del controlador. El apéndice A contiene las tablas de la representación de teclas de este conjunto.

Usualmente, el teclado está programado para que envíe sólo un tipo de conjunto, como es el caso de los teclados actuales que por defecto utilizan el conjunto de códigos de rastreo 2. Sin embargo, el teclado puede ser programado para que soporte el conjunto de códigos de rastreo 3.

Los datos que el teclado envía al dispositivo controlador son bytes. Debido a que éstos son enviados de manera serial por la línea de datos del teclado, debe agregar algunos bits de manera que el controlador se percate de que el teclado está enviando bits. Así, los datos están formados finalmente por tramas de 11 bits. El orden de envío es el siguiente:

- Bit de inicio. La línea de datos normalmente está en un nivel lógico alto, por lo que el bit de inicio obliga que ésta pase a un nivel lógico bajo con el objeto de indicar al controlador el inicio de la trama.

- Byte de datos. Es la representación de la información que el teclado envía al controlador. El dato se envía desde el bit menos significativo (LSB, Least Significant Bit) al bit más significativo (MSB, Most Significant Bit).
- Bit de paridad. Este bit lo utiliza el controlador para identificar si la trama es válida. Se utiliza paridad impar.
- Bit de paro. Después del bit de paridad, este bit coloca la línea de datos a un nivel lógico alto para indicar el final de la trama.

La terminal de reloj se utiliza durante el envío de los bits mencionados. Cuando se presiona la tecla, el teclado genera una señal de reloj y en cada flanco de bajada de dicha señal el controlador debe leer el bit de la línea de datos. El teclado envía el siguiente bit en el flanco de subida de reloj. Cuando se termina el envío de todos los bits de la trama (incluyendo el bit de paridad) el teclado deja de generar la señal de reloj.

La figura 1.12 muestra el envío de tramas por la línea de datos. Como se puede observar, en el flanco de bajada el bit de envío actual se mantiene estable por un lapso hasta el flanco de subida de la señal de reloj [URL11].

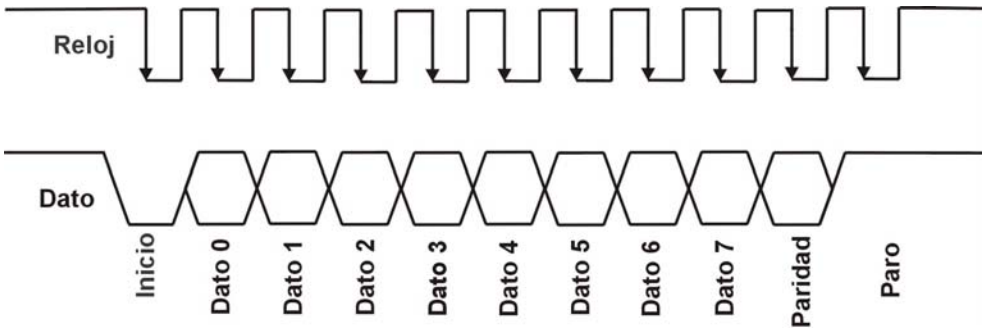

Figura. 1.12. Pulsos de las líneas de reloj y de datos en el envío de información del teclado al controlador.

1.4.3 Ejecución de órdenes

Una orden le indica al teclado que debe realizar una acción determinada, por ejemplo, cuando un usuario presiona la tecla de letras mayúsculas "Bloq Mayús", el sistema controlador recibe el código de rastreo de esa tecla, al detectar que se trata de la tecla "Bloq Mayús" envía una orden al teclado para indicarle que encienda el LED correspondiente, el teclado recibe la orden y la ejecuta.

Existen dos formas de hacer que un teclado ejecute órdenes:

1. Al alimentar de voltaje al teclado, éste ejecuta de manera automática la orden de reset.
2. Que el controlador le indique al teclado que ejecute una orden.

Dado que no se tiene control sobre el primer punto, la explicación siguiente se enfoca en el segundo punto.

Las órdenes, al igual que los códigos de rastreo, son datos de 1 byte, la trama se forma de manera similar:

- Bit de inicio. Pone la línea de datos en un nivel lógico bajo.
- Byte de datos. Bits que representan la orden enviada al teclado del LSB al MSB.
- Bit de paridad. Debe ser impar.
- Bit de paro. Mantiene la línea de datos en un nivel lógico alto.

Después del envío del bit de paro, el teclado envía al controlador un bit de reconocimiento para indicarle que recibió la orden de manera exitosa.

Para el envío de órdenes del controlador al teclado –al igual que en el envío de códigos del teclado al controlador– la señal de reloj también la genera el teclado. Cuando el controlador envía tramas lo hace en cada flanco de bajada de la señal de reloj, por otro lado, el teclado lee los bits en cada flanco de subida.

El controlador debe seguir los siguientes pasos para el envío de órdenes al teclado [URL11]:

1. Poner la línea de reloj en un nivel lógico bajo en por lo menos 100 μs, con esto el controlador toma el control sobre la comunicación.
2. Terminado ese lapso debe liberar la señal de reloj de teclado. Esto es porque el teclado es el único que genera la señal de reloj, necesaria para el envío y lectura de datos.
3. Poner la señal de dato en un nivel lógico bajo (bit de inicio) lo cual interpretará el teclado como una petición del controlador para enviar datos. Cuando el teclado detecte esto, comenzará a generar las señales de reloj para los 8 bits de datos, el bit de paridad y el bit de paro.
4. Para enviar los bits de la orden y el bit de paridad, el controlador debe esperar a que el teclado ponga la línea de reloj en bajo. En cada flanco de bajada el controlador debe enviar 1 bit, de esta manera se envía cada bit de la orden (comenzando desde el bit menos significativo y finaliza con el bit de paridad).
5. Para enviar el bit de paro, el controlador debe esperar nuevamente un flanco de bajada de la señal de reloj del teclado. Si el controlador después de esto no libera la línea de datos, el teclado continuará generando pulsos de reloj hasta que la línea de datos se libere.
6. Por último, el controlador debe esperar a que el teclado ponga la línea de reloj en un nivel lógico bajo para leer el bit de reconocimiento, el cual debe tener un nivel lógico bajo. (Después de que el controlador envía el bit de paro, el teclado enviará una señal de reconocimiento al poner la línea de dato en un nivel lógico bajo y generando un último pulso de reloj).

Existen además 2 parámetros en el teclado: el retardo *typematic*, el cual es un tiempo corto entre el primer y el segundo código de rastreo (de 0.25 segundos a 1.00 segundos), y el *typematic rate*, que indica cuantos caracteres por segundo se enviarán después del retardo typematic (2.0 a 30.0 caracteres por segundo) [URL10].

La lista de órdenes empleadas en este trabajo se muestra en el apéndice B.

1.5 MONITOR

Es el enlace visual del usuario con los programas que ejecuta un sistema. Una forma muy usada para el despliegue de imágenes en la pantalla es un monitor de tubo de rayos catódicos (CRT, Cathode-Ray Tube). El CRT es, básicamente, un gran tubo catódico al vacío en forma de botella. Por la parte trasera del tubo se encuentra un cañón de electrones el cual dispara un haz de electrones, cuando éstos hacen contacto con la parte interna de la pantalla se emite luz. El interior de la pantalla está recubierto con un tipo especial de fósforo, así el color de la luz emitido por la pantalla está determinado por el fósforo particular usado [5]. La figura 1.13 ilustra algunos componentes de un CRT.

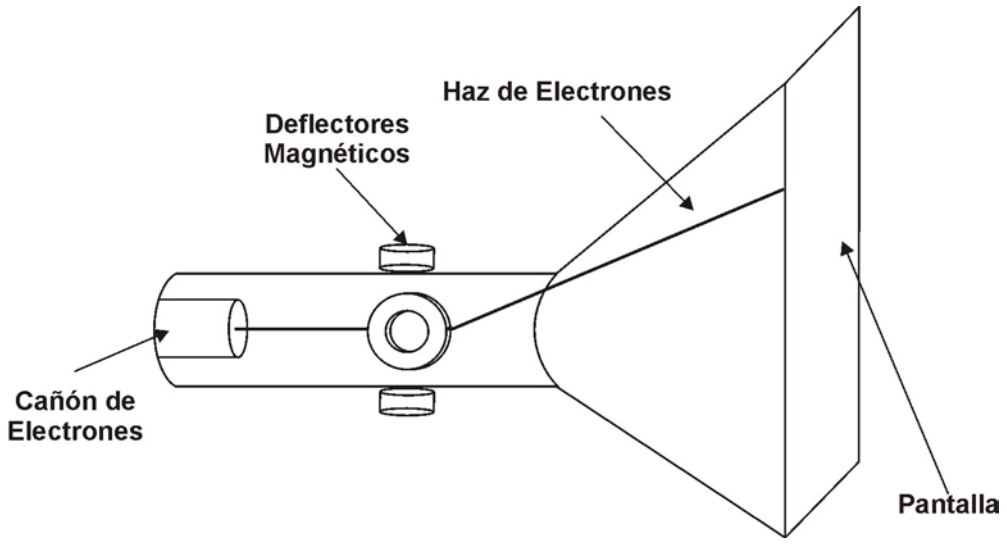

Figura 1.13. CRT del monitor.

1.5.1 Señales para producir una imagen

Una imagen en la pantalla del monitor no permanece fija, es decir, haciendo una analogía, no es como una hoja impresa sino que es una imagen "parpadeando" a tal velocidad que es imperceptible al ojo humano. En tales "parpadeos" la imagen que se muestra en el monitor es nula, por lo que la imagen que percibimos se está refrescando continuamente.

La imagen no es producto de una sola señal, de hecho es el resultado de la combinación de 3 señales independientes, que forman la llamada Señal de Video Compuesta [7]. Las señales que participan en esta señal combinada son:

1. Señal de Luminancia.
2. Pulsos de Sincronización.
3. Pulsos de Blanqueo.

Se puede considerar la pantalla como una malla de posiciones, por lo que una imagen está compuesta por los elementos de la malla. Cada elemento de la imagen representa un punto o píxel (también llamado pel). Para mostrar un elemento de la imagen en la pantalla, el haz de electrones excita al elemento generándose una señal que es proporcional a la intensidad de la luz que incide sobre él, a esta señal se le llama señal de luminancia.

La señal de luminancia es la señal de video, pero sólo representa un punto de la imagen. Para desplegar toda la imagen en la pantalla se debe mostrar cada elemento de la imagen, uno a la vez. Para llevar a cabo esto, se debe realizar la exploración.

La exploración es un método para desplegar imágenes en una pantalla de CRT. Su nombre se debe a que el haz de electrones explora la superficie de la pantalla, en otras palabras, realiza un barrido de la superficie de la pantalla comenzando de la parte izquierda a la derecha y desde arriba hacia abajo (figura 1.14), de esta manera se muestran todos los puntos de la malla [8]. La exploración se realiza de la misma manera en que se lee un libro, es por eso que se le llama exploración horizontal secuencial.

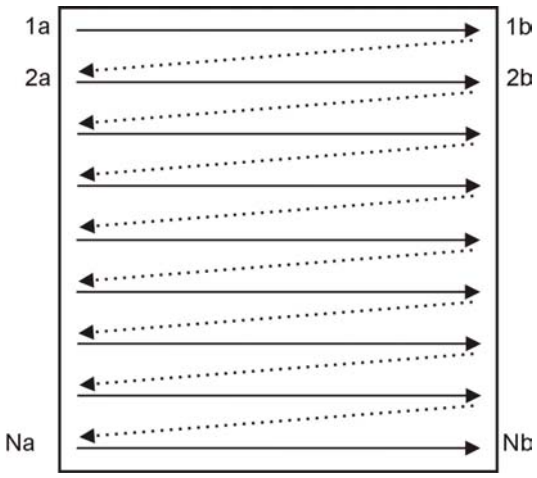

Figura 1.14. Exploración de la pantalla.

El origen de la exploración es en la parte superior izquierda de la pantalla (punto 1a). Para desplegar la primera línea de la pantalla, el haz de electrones se mueve en forma horizontal hasta llegar a la parte superior derecha (punto 1b). En ese momento se ha realizado el despliegue de la primera línea de la pantalla. Para desplegar toda la imagen se deben desplegar todas las líneas que conforman la imagen, por ello, el haz de electrones debe continuar con la siguiente línea en la pantalla, por lo que regresa a la parte izquierda pero ahora se posiciona en la segunda línea (punto 2a). El haz de electrones continua realizando el barrido y cuando llega a la parte derecha de la última línea de la pantalla (punto Nb), el haz regresa al origen de la pantalla (punto 1a). De esta manera la pantalla parecerá desplegar toda la imagen al mismo tiempo. A la imagen completa, compuesta por todas las líneas de exploración, se le llama trama (frame).

Para controlar la trayectoria del haz de electrones, los deflectores magnéticos utilizan los pulsos de sincronización y, como es de esperarse, hay pulsos de sincronización horizontal y vertical. Normalmente estas señales se encuentran en un nivel lógico alto, cuando se aplica el pulso de sincronización pasan a un nivel lógico bajo.

El pulso de sincronización horizontal marca el inicio y final de una línea de exploración, con esto garantiza el despliegue de píxeles entre los límites de izquierda a derecha del área visible de la pantalla. De igual forma, el pulso de sincronización vertical marca el inicio y fin de la imagen pero de manera vertical, es decir, garantiza que el monitor despliegue las líneas de exploración entre la parte superior e inferior de la pantalla.

Por otra parte, cuando el haz de electrones regresa de la parte derecha de la imagen a la parte izquierda de la siguiente línea, ocurre un lapso de tiempo llamado retraso horizontal. De la misma manera, al tiempo en que el haz regresa del extremo derecho inferior de la pantalla a la parte superior izquierda se llama retraso vertical [7].

Durante los tiempos de retraso horizontal y vertical no se despliega la imagen en la pantalla. Es en ese momento en el que se aplican los pulsos de blanqueo (llamados blankings en inglés).

El objetivo de los pulsos de blanqueo es poner la pantalla en negro cuando se presenta algún tiempo de retraso. Esto debido a que en ese lapso el haz de electrones no realiza ninguna exploración, es decir, no despliega imágenes en la pantalla por lo que no es necesario mostrar algo en ella.

Así, un monitor debe contar con 3 señales de entrada para desplegar una imagen: pulso de sincronización horizontal, pulso de sincronización vertical y señal de intensidad del haz.

1.5.2 Temporización de las señales

En la exploración horizontal el tiempo máximo en que se despliegan los píxeles es de 25.17 μs (dentro de ese tiempo el haz de electrones se desplaza horizontalmente). Después de desplegar el último píxel de la línea horizontal, se debe aplicar el blanqueo (llamado blanqueo horizontal), es decir, poner la pantalla en negro por 6.6 μs como mínimo.

Igualmente, al terminar los 25.17 μs se debe esperar un mínimo de 0.94 μs para aplicar el pulso de sincronización horizontal en un nivel lógico bajo y mantenerlo por 3.77 μs.

Después de aplicar el pulso de sincronización horizontal, se debe esperar por lo menos 1.89 μs antes de comenzar una nueva línea, es decir, antes de enviar los píxeles de la siguiente línea.

Así, el total de tiempo en que se explora la línea horizontal y antes de desplegar la siguiente es de 31.77 μs.

Para el pulso de sincronización vertical el tiempo máximo en que el monitor debe desplegar líneas es de 15.25 ms. Después de ese tiempo, se aplica el blanqueo (blanqueo vertical) por 1.534 ms para poner la pantalla oscura.

Antes de aplicar el pulso de sincronización vertical, se debe esperar 0.45 ms después de la última línea. El pulso de sincronización vertical permanece en un nivel lógico bajo por 64 μs, después de ese tiempo, se debe esperar por lo menos 1.02 ms antes de comenzar de nuevo con la primera línea.

Así, el tiempo total en que se despliegan todas las líneas y antes de desplegar la primera línea nuevamente es de 16.784 ms.

La figura 1.15 ilustra los tiempos de las señales que participan en el despliegue de la imagen. El blanqueo vertical se efectúa después de un lapso de varias líneas de video (líneas de exploración horizontal). Normalmente para un monitor VGA el número de píxeles desplegados horizontalmente es de 640 y el número de líneas visibles es de 480, por lo que su resolución es de 640x480.

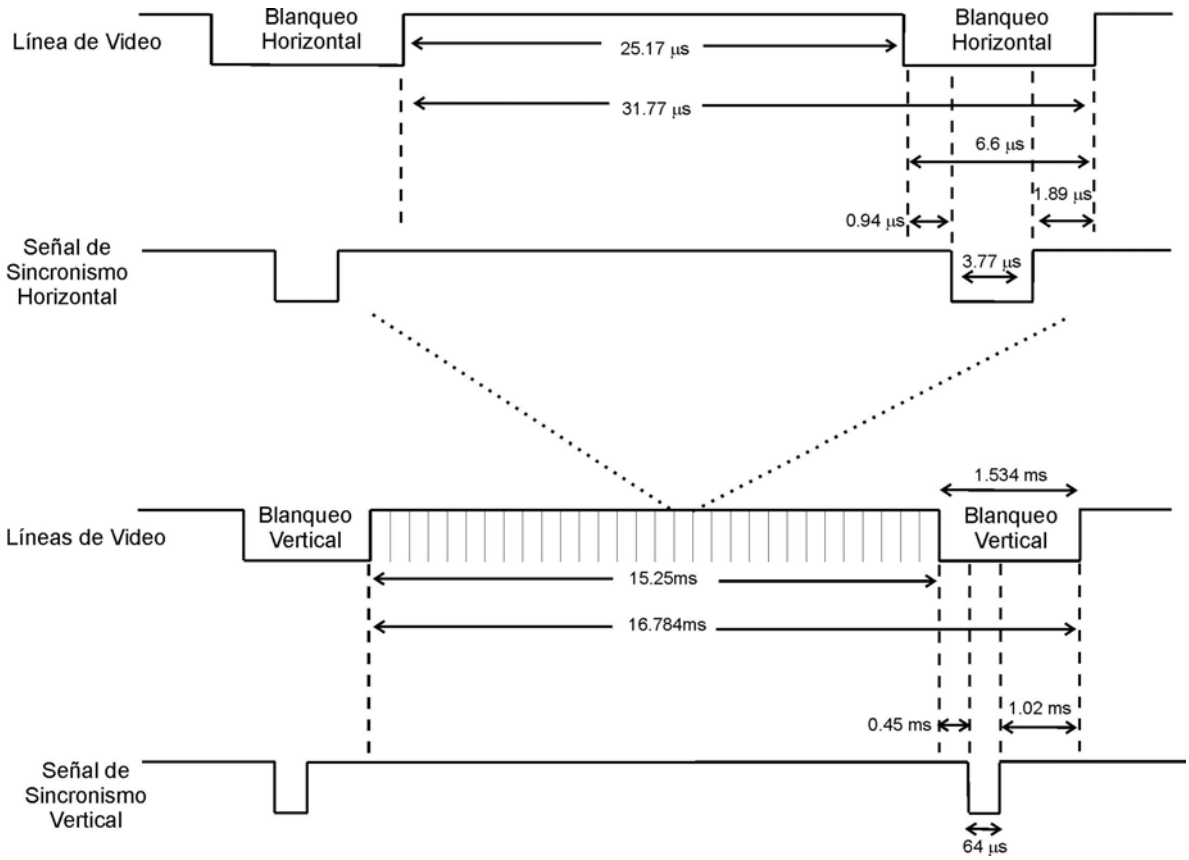

Figura 1.15. Temporización de las señales del monitor.

1.5.3 Representación de caracteres en la pantalla

Para representar los caracteres en la pantalla se emplean 2 memorias: la ROM generadora de caracteres y la RAM de refresco de pantalla [5].

1.5.3.1 ROM generadora de caracteres

La imagen se muestra en una pantalla de CRT como un patrón, ya sea de puntos de luz o de obscuros. Para representar los caracteres, se emplean matrices de puntos (píxeles), donde cada matriz representa un carácter. Por ejemplo, la figura 1.16 muestra la matriz de puntos de la letra A. Los puntos redondos representan el haz encendido, y los cuadros vacíos representan el haz apagado. El carácter es de 7 puntos de ancho por 9 puntos de alto (aunque existen varias combinaciones como, por ejemplo, de 7x12).

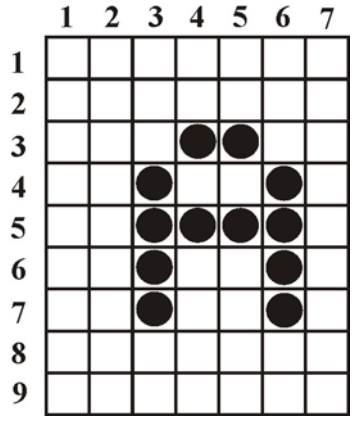

Figura 1.16. Matriz de puntos de la letra A.

La representación del patrón de puntos de cada carácter ASCII a ser desplegado se almacena en una ROM llamada ROM generadora de caracteres. En esta memoria cada punto de la matriz se representa por un bit, por lo que, si el bit es 1 éste corresponde a un haz encendido, y si es 0 representa un haz apagado en la línea de la matriz correspondiente.

Por ejemplo, si en la ROM generadora de caracteres las matrices están formadas por caracteres de 6x8 puntos y, además, almacena caracteres de letras mayúsculas iniciando de la localidad 0x00, entonces el contenido de la ROM estaría formado tal y como se ilustra en la figura 1.17, el espacio en blanco representa un bit 0 y el espacio en negro representa el bit 1. El objetivo de dejar espacios vacíos en la matriz es que cuando se muestran los caracteres contiguos en la pantalla no estén demasiado juntos. La ROM debe contener los datos para representar todas las matrices de puntos de todos los caracteres imprimibles.

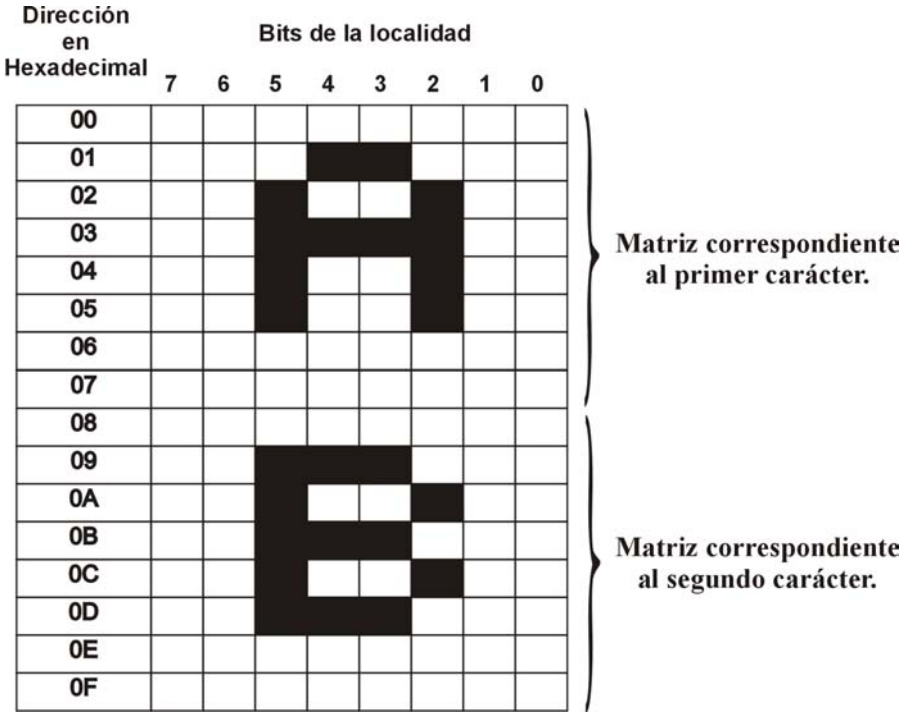

Figura 1.17. Matrices A y B en la ROM generadora de caracteres.

1.5.3.2 RAM de refresco de pantalla

La ROM generadora de caracteres sólo contiene las matrices de cada carácter que se puede presentar en la pantalla, pero la información que se muestra en pantalla debe estar almacenada en memoria RAM que a menudo se le llama frame buffer, RAM de refresco de pantalla o memoria de video.

La RAM de refresco de pantalla contiene almacenados los códigos ASCII de los caracteres que se están desplegando en la pantalla del monitor. Por ejemplo, si en pantalla aparece la secuencia de letras A, E, I, O y U en la memoria RAM se deben localizar los códigos ASCII de cada carácter y la pantalla debe mostrar la secuencia de letras.

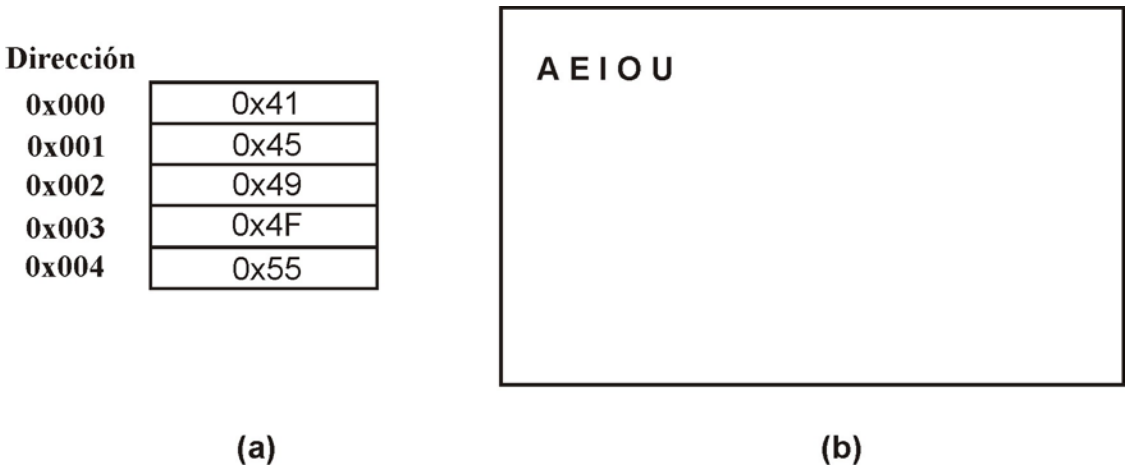

Figura 1.18. (a) RAM de refresco de pantalla que contiene los códigos ASCII de 5 vocales.
(b) Orden en el que aparecen los caracteres en la pantalla.

La figura 1.18(a) muestra que la RAM de refresco de pantalla contiene los códigos ASCII de los caracteres A, E, I, O y U; el resto de la memoria debe contener el código ASCII correspondiente al espacio en blanco. La figura 1.18(b) muestra los caracteres desplegados en la pantalla, para que suceda esto el sistema que genera la imagen (por ejemplo, una tarjeta de video) debe leer el contenido de la memoria RAM y, con base en ello, obtener los píxeles correspondientes de la ROM generadora de caracteres para desplegarlos uno a uno según la posición del haz en la línea de exploración.

1.5.4 Color en la pantalla

En un monitor monocromático la pantalla está recubierta con un solo tipo de fósforo. Cuando se le aplica el haz de electrones produce un color específico. Para desplegar una imagen a color, los monitores tienen pantallas recubiertas con puntos de fósforo rojo, verde y azul. El trío de puntos de fósforo conforma un píxel en la pantalla, es por ello que los puntos de fósforo deben encontrarse muy cercanos para que, de esta manera, se tenga la sensación de que forman un solo punto o píxel con color [URL12].

Como se puede ver en la figura 1.19, a cada punto de fósforo se le asigna un haz de electrones. La máscara se utiliza para que un haz de electrones asignado no se posicione en un punto que no le corresponde.

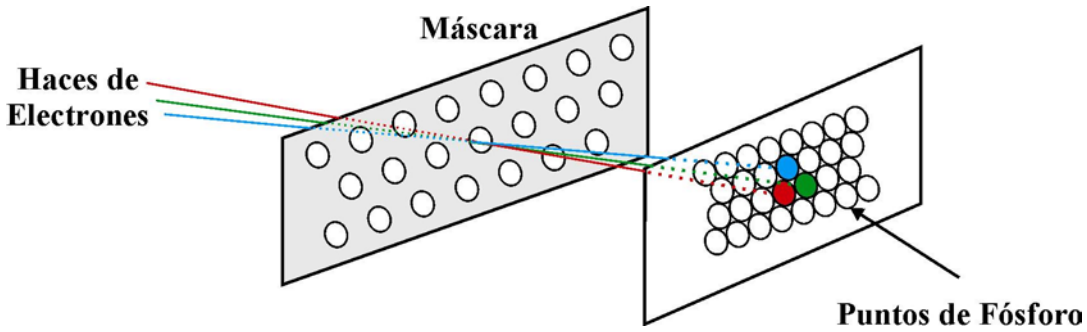

Figura 1.19. Haces de electrones que forman un punto o píxel.

Así, un monitor a color debe contar con las siguientes señales de entrada: intensidad roja, verde y azul, señal de sincronización horizontal y señal de sincronización vertical.

Las entradas del monitor para el color son analógicas e indican la intensidad del haz. Un controlador de video emplea un convertidor digital a analógico (DAC), el objetivo de esto es obtener un mayor rango de colores en la pantalla. La figura 1.20 muestra un bloque convertidor DAC.

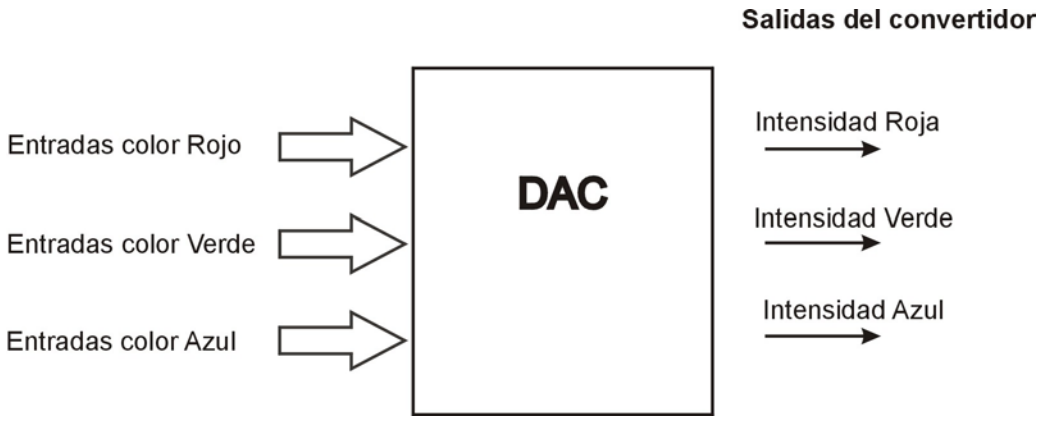

Figura 1.20. Convertidor DAC para obtener la intensidad de color.

A cada salida del DAC se le asigna una terminal del conector del monitor, es decir, son las entradas de video. Los niveles digitales indican la intensidad del haz de electrones, por lo que si las señales de entrada del convertidor están en un nivel lógico bajo la intensidad del píxel desplegado será baja (píxel negro) y, por el contrario, si las señales de entrada del convertidor están todas en un nivel lógico alto el píxel desplegado tendrá el máxima brillantez (luminancia). Los colores del píxel se encuentran dentro del rango de la intensidad mínima a la máxima intensidad.

El rango de colores depende de la resolución del convertidor, es decir, de la menor variación que se puede obtener en la salida analógica con respecto a la entrada digital. A más bits por píxel, se tiene un mayor número de colores. Por ejemplo, si el convertidor DAC que se emplea es 2 bits de entrada para cada color, el número total de colores que se pueden mostrar en la pantalla será 64 (tabla 1.3), esto debido a que cada una de las 3 entradas del DAC ofrece 4 combinaciones posibles.

Entradas	**Combinaciones para cada entrada**	**Total de colores**
Color Rojo 2	00 01 10 11	4x4x4 = 64
Color Verde 2		
Color Azul 2		

Tabla 1.3. Cantidad de colores para dos entradas por color en el DAC.

1.6 MODBUS

Modbus es un protocolo de mensajes, ubicado en la capa de aplicación del modelo de referencia de interconexión de sistemas abiertos (sección 1.6.1), desarrollado por Modicon en 1979 y que se usa para establecer una comunicación maestro-esclavo. Es un protocolo abierto usado en la industria, se emplea en [URL13]:

- TCP/IP sobre Ethernet.
- Transmisión serial asíncrona sobre una variedad de medios (EIA/TIA-232-E, EIA-422, EIA/TIA-485-A, fibra, radio, etc.).
- Modbus Plus, un token de alta velocidad pasando en la red.

Modbus es un protocolo de petición/respuesta, en el cual un nodo maestro envía una petición a un nodo esclavo, éste recibe la petición, realiza una función y envía una respuesta al nodo maestro.

1.6.1 Capas del modelo OSI en Modbus sobre línea serial

El modelo de referencia de interconexión de sistemas abiertos (OSI, Open Systems Interconection) define un conjunto jerárquico de capas que cubren las funciones necesarias para establecer la comunicación entre sistemas. Cada capa realiza funciones bien definidas, las cuales sirven de apoyo a la capa contigua. Las capas que el modelo OSI considera son siete [8]:

- Capa de Aplicación.
- Capa de Presentación.
- Capa de Sesión.
- Capa de Transporte.
- Capa de Red.
- Capa de Enlace de Datos.
- Capa Física.

En muchos sistemas de comunicaciones se omiten algunas capas dado que el mismo modelo OSI así lo permite. Para un sistema de comunicación serial, además de la capa de aplicación, Modbus describe 2 capas más del modelo OSI (y es llamado Modbus sobre línea serial), éstas son: la capa de enlace de datos y la capa física.

Las secciones 1.6.2 a la 1.6.4 describen estas capas.

1.6.2 Capa de aplicación

Proporciona a los programas de aplicación un medio para que accedan al entorno Modbus, así como los mecanismos necesarios en la implementación de aplicaciones distribuidas [8]. Modbus define en esta capa los mensajes de intercambio, así como el tipo de datos empleado [URL13].

1.6.2.1 Unidad de datos del protocolo

El protocolo Modbus define una unidad de datos del protocolo (PDU, Protocol Data Unit) que es independiente de las capas de comunicación subyacentes. La PDU está formada por 2 campos: el campo de código de función y el campo de datos (figura 1.21).

Código de Función 1 byte	Datos 0 a 252 bytes

Figura 1.21. PDU Modbus.

El nodo maestro construye una PDU para enviarla como una petición. El nodo esclavo la recibe y debe construir una nueva PDU que será parte de la respuesta que se envía al nodo maestro.

Campo Código de Función de la PDU. Se encuentra en el rango de 0x01 a 0xFF. Cuando el nodo maestro realiza la petición, este campo indica al nodo esclavo la función que debe realizar. En el caso de la respuesta del nodo esclavo, le indica al nodo maestro que se ha realizado la función deseada o que ocurrió un error.

Campo de Datos de la PDU. En la petición del nodo maestro, contiene la información necesaria para que el nodo esclavo realice una acción indicada por el código de función. En la respuesta del nodo esclavo contiene información relacionada a la acción realizada (descrita en la sección 1.6.2.3).

Si el nodo esclavo realiza la función con éxito, en la PDU de respuesta el código de función es el mismo que el código de función de la petición. De esta manera, el nodo maestro, al recibir la respuesta, sabe que el nodo esclavo tuvo éxito al procesar la función deseada (figura 1.22).

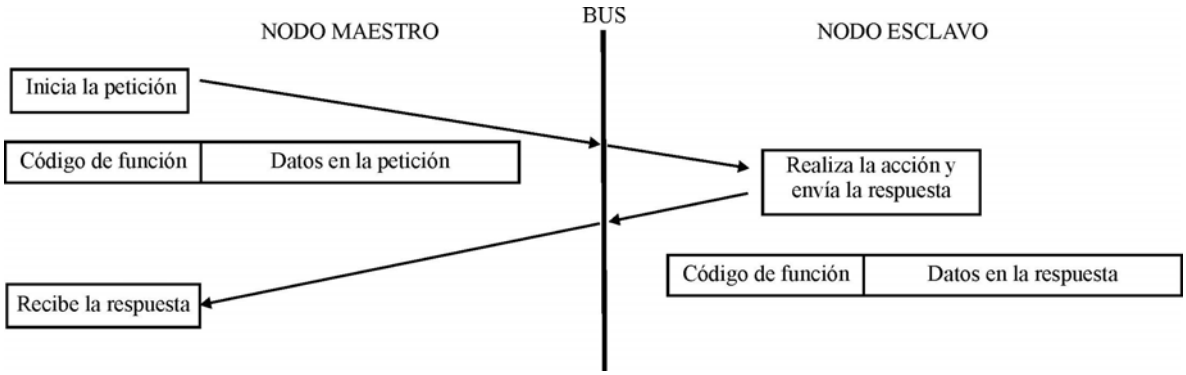

Figura 1.22. Petición Modbus realizada sin error [URL13].

Sin embargo, si en el nodo esclavo ocurre un error relacionado con la función descrita en el código de función, éste debe construir una PDU con el llamado código de error y con el código de excepción. A esta PDU se le llama respuesta de excepción y se muestra en la figura 1.23. Como se puede observar el código de error toma el lugar del campo código de función de la respuesta y, a su vez, el código de excepción toma el lugar del campo de datos.

El código de error es el resultado de la suma del código de función más 0x80. Gracias a él, el nodo maestro detecta en la PDU de respuesta que existió un error en el nodo esclavo al tratar de realizar la función. El código de excepción, por otro lado, indica al nodo maestro cuál fue el error que ocurrió.

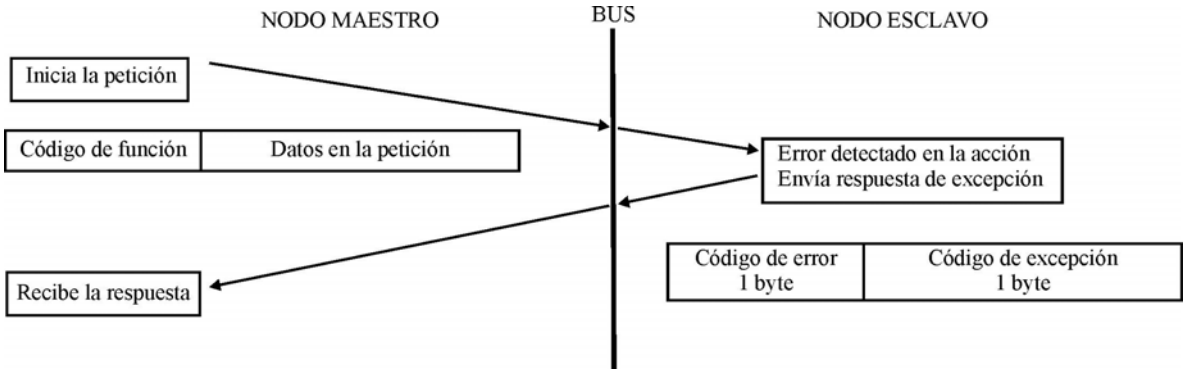

Figura 1.23. Respuesta de excepción Modbus [URL13].

1.6.2.2 Tipos de datos en Modbus

Los datos que Modbus utiliza son bits y registros, éstos se deben localizar en memoria del dispositivo al cual se le hace acceso. La capacidad de dicha memoria para cada tipo de dato es de 0 a 65535 localidades de dirección (0x0000 a 0xFFFF).

Modbus describe cuatro tablas primarias para los tipos de datos que maneja (tabla 1.4).

Tablas Primarias	Tipo de Objeto	Tipo de acceso
Entradas Discretas	Un solo bit	Sólo de Lectura
Coils	Un solo bit	Lectura – Escritura
Registros de Entrada	Palabra de 16 bits	Sólo de Lectura
Registros de Holding	Palabra de 16 bits	Lectura – Escritura

Tabla 1.4. Tipos de datos en Modbus.

1.6.2.3 Códigos de función

Como se ha descrito anteriormente, los códigos de función indican al nodo esclavo la acción que debe realizar. Éstos se dividen en tres categorías:

1. Públicos. Son códigos definidos por Modbus y únicos, es decir, que no se repiten. Los rangos de estos códigos en hexadecimal son: de 1 a 40, de 49 a 63 y de 6F a 7F.
2. Definidos por el usuario. Son códigos de función establecidos por el usuario y sin autorización de Modbus, por lo que, no hay garantía que el código de función se repita. Los rangos para estos códigos en hexadecimal son: de 41 a 48 y 64 a 6E
3. Reservados. Códigos de función que utilizan algunas compañías y no están disponibles al público.

La figura 1.24 muestra la distribución de los códigos de función de Modbus con respecto a sus categorías.

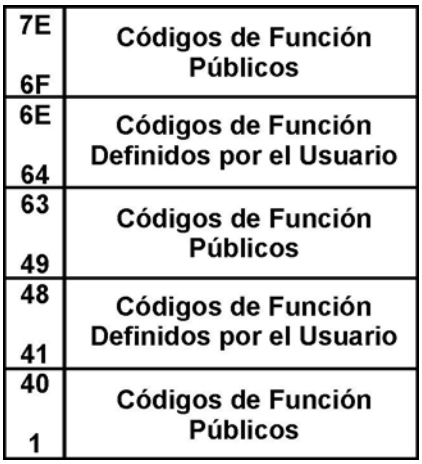

Figura 1.24. Categorías de códigos de función Modbus.

A continuación se describen los códigos de función que se emplearon en la realización del presente trabajo.

Código de Función 0x01

Indica al nodo esclavo que se lean desde 1 a 2000 estados continuos de bits. En el campo de datos de la PDU de petición, se especifica la dirección inicial de lectura y el número de bits. Así, la PDU de petición se forma de la siguiente manera:

Código de Función	1 Byte	0x01
Dirección de Inicio	2 Bytes	0x0000 a 0xFFFF
Cantidad de bits a leer	2 Bytes	1 a 2000 (0x7D0)

Tabla 1.5. Contenido de la PDU de petición del código de función 0x01.

En la PDU de respuesta, el campo de datos debe contener la lectura de los bits, así como el conteo de bytes, es decir, la cantidad de bytes que contienen la lectura de los bits. La información en el campo de datos se envía por bytes, por ello, los bits de lectura se almacenan desde la parte menos significativa del byte hasta la parte más significativa. Sin embargo, si la lectura realizada no es múltiplo de ocho, los bits restantes del byte toman el valor de cero.

La PDU de respuesta está formada por:

Código de Función	1 Byte	0x01
Cantidad de Bytes	1 Byte	N[3]
Bits leídos	N Bytes	n = N o N+1

Tabla 1.6. Contenido de la PDU de respuesta del código de función 0x01.

La PDU de respuesta de excepción:

Código de Función	1 Byte	Código de Función + 0x80
Código de Excepción	1 Byte	0x01 o 0x02 o 0x03 o 0x04

Tabla 1.7. Contenido de la PDU de respuesta de excepción del código de función 0x01.

Código de Función 0x10

Este código de función se usa para indicarle al nodo esclavo que debe escribir un conjunto de registros (de 1 a 120 registros). En el campo de datos de la PDU de petición se indica la dirección de inicio de escritura, cantidad de registros, el conteo de bytes y los registros a escribir. Un registro está formado por 2 bytes por lo que el valor del conteo de bytes es el doble del valor de la cantidad de registros. Así, la PDU de petición se forma de la siguiente manera:

Código de Función	1 Byte	0x10
Dirección de Inicio	2 Bytes	0x0000 a 0xFFFF
Cantidad de Registros	2 Bytes	0x0001 a 0x007B
Conteo de Bytes	1 Byte	2 x N[4]
Valor de Registros	N[4] x 2 Bytes	Valor

Tabla 1.8. Contenido de la PDU de petición del código de función 0x10.

.

En la PDU de respuesta sin errores se tiene el código de función, dirección de inicio y cantidad de registros escritos.

[3] Cantidad de salidas entre ocho, si el residuo es diferente de cero entonces se suma uno a la división.
[4] Cantidad de registros.

La PDU de respuesta está formada por:

Código de Función	1 Byte	0x10
Dirección de Inicio	2 Bytes	0x0000 a 0xFFFF
Cantidad de Registros	2 Bytes	1 a 123 (0x7B)

Tabla 1.9. Contenido de la PDU de respuesta del código de función 0x10.

Por otra parte, la PDU de respuesta con error contiene el código de error y el código de excepción.

Código de Error	1 Byte	0x90
Código de Excepción	1 Byte	0x01 o 0x02 o 0x03 o 0x04

Tabla 1.10. Contenido de la PDU de respuesta de excepción del código de función 0x10.

La tabla del apéndice C describe los códigos de excepción que se emplean en este trabajo.

1.6.3 Capa de enlace de datos

Proporciona un servicio de transferencia de datos seguro a través del enlace físico; se encarga del envió de tramas llevando a cabo la sincronización y el control de errores [8]. A continuación se presentan las especificaciones que Modbus sobre línea serial hace en esta capa.

1.6.3.1 Modos de petición del nodo maestro en Modbus sobre línea serial

El nodo maestro establece una petición que llega a todos los nodos esclavos. Esta petición puede ser de dos modos [URL14]:

- *Modo Unidestino***.** La petición del nodo maestro está destinada a un nodo esclavo, es decir, sólo un nodo esclavo debe realizar la función de la petición y debe enviar al nodo maestro una respuesta.

- *Modo Difusión***.** La petición del nodo maestro se dirige a todos los nodos esclavos, sin embargo, éstos no envían una respuesta al maestro puesto que todos los nodos comparten el medio y, de enviar una respuesta, se ocasionará una colisión de datos.

Los nodos esclavos están en espera de una petición del maestro, éste sólo se puede comunicar con uno de ellos a la vez. La comunicación siempre es maestro-esclavo, por lo que los nodos esclavos no se pueden comunicar entre ellos.

1.6.3.2 Modos de transmisión serial

El modo de transmisión serial se refiere a la manera en que se empaqueta la información para ser enviada. En Modbus sobre línea serial se definen 2 modos de transmisión: el modo unidad terminal remota (RTU, Remote Terminal Unit) y el modo código estándar estadounidense para intercambio de información (ASCII, American Standard Code for Information Interchange).

El modo de transmisión ASCII es opcional en Modbus sobre línea serial. Este modo es menos eficiente que el modo RTU ya que requiere de 2 códigos ASCII para representar un byte. Por ejemplo, para representar el byte 0xA8, se requiere del código ASCII 0x41, que representa al nibble más significativo del byte, y del código ASCII 0x38, que representa el nibble menos significativo del byte.

El modo de transmisión RTU es obligatorio. A diferencia del modo de transmisión ASCII, el byte de información que se envía no se fragmenta en 2 partes y, por lo tanto, resulta más eficiente. En adelante, la información que se presenta está orientada al modo de transmisión RTU.

1.6.3.3 Envío de datos

Los datos que se transmiten de un nodo maestro a un nodo esclavo o viceversa, se envían por bytes. La transmisión de un byte es asíncrona, por lo que se le debe agregar algunos bits de información útil para el nodo receptor. Estos bits son: bit de inicio, bit de paridad y bit de paro (figura 1.25).

Figura 1.25. Formato de un carácter en Modbus.

Inicialmente, cuando no hay transmisión de carácter, la línea entre el emisor y receptor se encuentra en un estado de reposo (un nivel lógico alto). Así que, cuando comienza la transmisión, el bit de inicio aplica un nivel lógico bajo a la línea con lo que indica al receptor que se está transmitiendo un carácter.

Los 8 bits de datos se trasmiten comenzando del bit menos significativo al bit más significativo. El receptor debe ordenar los bits de datos a su estado original.

El bit de paridad se usa para la detección de errores. Lo determina el nodo transmisor para indicar al receptor que el carácter resultante tenga una cantidad de bits uno par (paridad par) o una cantidad impar (paridad impar). Modbus sobre línea serial establece que se utilice paridad par, aunque opcionalmente se puede utilizar paridad impar o no utilizar paridad.

Finalmente, el bit de paro aplica un nivel lógico alto a la línea para indicar el fin del envío. Normalmente es de 1, 1.5 o 2 veces la duración de un bit convencional. Si se opta por no aplicar paridad, se deben considerar 2 bits de paro.

1.6.3.4 Trama Modbus

A la PDU descrita en la capa de aplicación se le agregan 2 campos: el campo de dirección y el campo de verificación de errores. A esta trama nueva se le llama unidad de datos de aplicación (ADU, Application Data Unit) y se ilustra en la figura 1.26, como se puede observar la cantidad máxima de bytes en una trama Modbus es de 256.

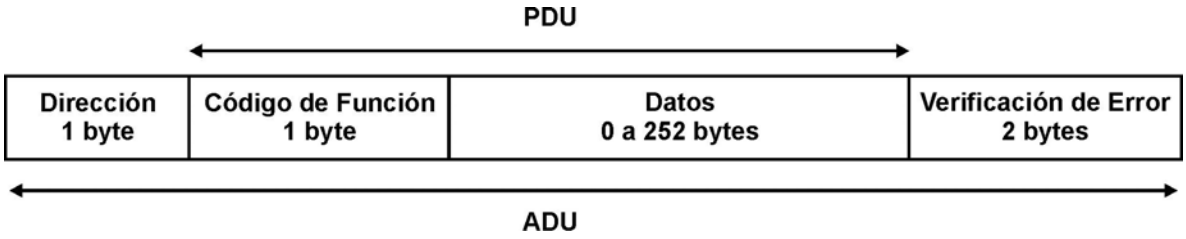

Figura 1.26. ADU Modbus.

Campo de Dirección de la ADU

El nodo maestro emplea este campo para indicar a qué nodo esclavo va dirigida su petición. Por otra parte, el nodo esclavo utiliza este campo para colocar su propia dirección en la respuesta que envía al nodo maestro. Algunas consideraciones con respecto al campo de dirección son las siguientes:

- Cada nodo esclavo debe contar con una dirección única que lo identifique de los demás nodos esclavos.
- Las direcciones de los nodos esclavos abarcan el rango de 1 a 247 (0x01 a 0xF7).
- La dirección 0x00 está reservada para una petición en modo difusión.
- El nodo maestro no tiene dirección.

Campo de Verificación de Error de la ADU

El campo de verificación de error se obtiene con la revisión de redundancia cíclica (CRC, Cyclical Redundancy Checking). Cuando un nodo transmisor desea enviar un mensaje, obtiene el valor de la CRC, lo agrega a la trama y lo envía. El nodo receptor recibe la trama y obtiene la CRC, si la CRC calculada no es igual a la recibida se considera que hay error en la trama. En el apéndice D se encuentra el algoritmo para obtener la CRC.

1.6.3.5 Envío de peticiones

Cuando un nodo maestro desea comunicarse con uno o varios nodos esclavos debe seguir el siguiente procedimiento:

1. Construir una ADU con la dirección destino del esclavo.
2. Enviar la ADU, emplear un temporizador y esperar:
   a) en modo difusión un tiempo (que depende de la aplicación de la petición) hasta que el temporizador expire.

b) en modo unidestino una respuesta del esclavo destino dentro de un tiempo (que depende de la aplicación) hasta que el temporizador expire.

El temporizador se emplea para prevenir que el nodo maestro permanezca en una espera indefinida de tiempo.

3. a) Si llega la respuesta, verifica que sea la trama correcta, es decir, que la respuesta recibida provenga del esclavo correcto y sin errores. Si no se detectan errores en la respuesta se concluye que la trama fue procesada de manera exitosa en el nodo esclavo (de otro modo se puede realizar nuevamente otra petición al esclavo) y procesa la respuesta.

   b) Si no se recibe una respuesta, expira el temporizador, y se genera un error (sólo en modo unidestino). Entonces el nodo maestro puede realizar de nuevo la petición. El número máximo de veces en que se realiza esto se deja a consideración del diseñador.

La figura 1.27 muestra el diagrama de estados del envío de peticiones en el nodo maestro.

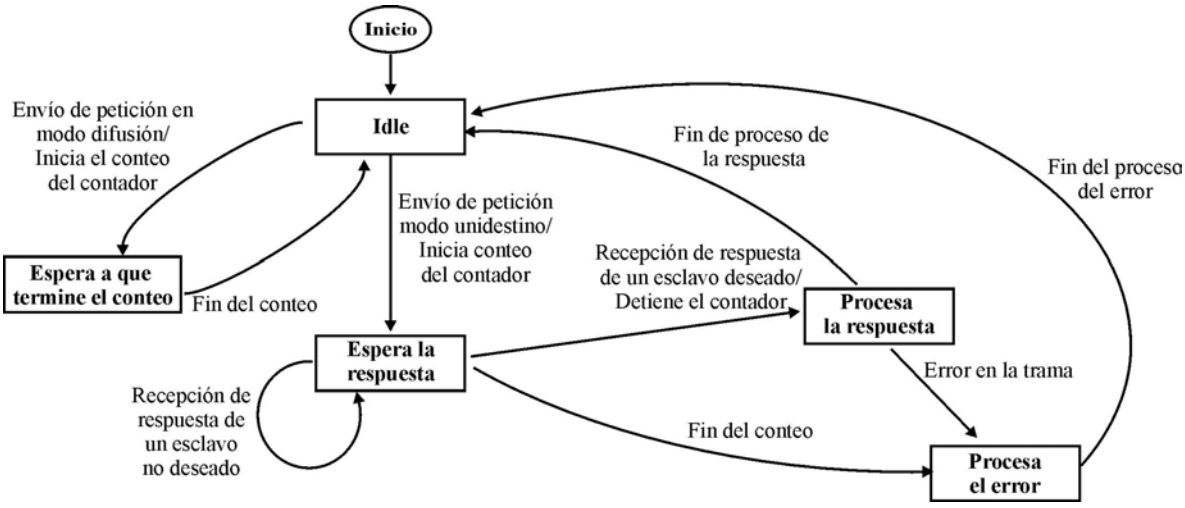

Figura 1.27. Diagrama de estados del envío de peticiones en el nodo maestro.

1.6.3.6 Envío de respuestas

Por otro lado, cuando un nodo esclavo recibe una trama de petición, el procedimiento es el siguiente:

1 El esclavo realiza el cálculo de la CRC de la trama recibida. Si detecta un error, no envía respuesta al maestro y se mantiene en espera de una nueva trama.

2 Si no hay error en la CRC verifica la trama antes de realizar una acción solicitada. Pueden ocurrir diferentes errores de excepción: error del formato en la petición, acción inválida, etc. En caso de que alguno de ellos se presente, envía una respuesta de excepción al nodo maestro.

3 Si no hay error de excepción, el nodo esclavo realiza la acción solicitada y, posteriormente, envía un mensaje de respuesta al nodo maestro.

La figura 1.28 muestra el diagrama de estados del envío de peticiones en el nodo esclavo.

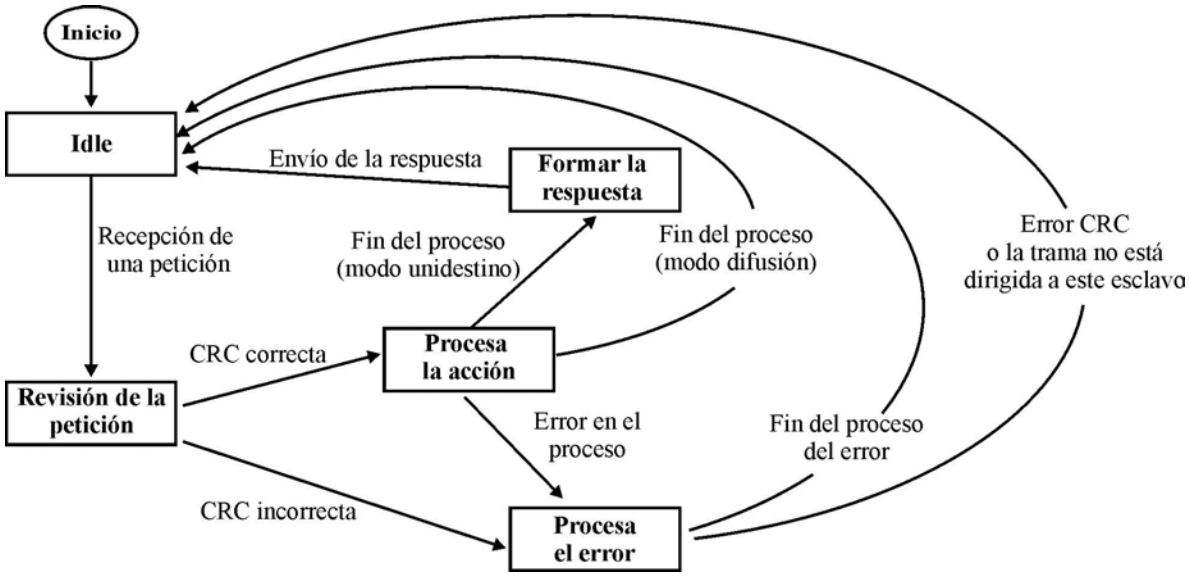

Figura 1.28. Diagrama de estados del envío de respuestas en el nodo esclavo.

1.6.3.7 Temporización de las tramas

Cuando un nodo envía información, es posible que requiera más de los 252 bytes que ofrece el campo de datos. Por otra parte, el nodo que recibe la información debe saber cuando termina el envío de la trama, es decir, el momento en el que el nodo transmisor deja de enviar información. Por lo tanto, si el nodo transmisor envía tramas contiguas el nodo receptor debe detectar el final de la última trama, para ello, debe emplear un contador o temporizador. La idea principal es la siguiente: cuando el nodo transmisor deja de enviar una trama (transmitir bytes) el nodo receptor activa un contador que cubra un intervalo de por lo menos 3.5 veces el tiempo que transcurre en el envío de un carácter, $t_{3.5}$; si el

contador cubre el intervalo sin que lleguen más datos, se considera que el transmisor terminó de enviar la trama, es decir, se detecta el final de la trama.

Sin embargo, si ocurre un intervalo sin que se reciban caracteres mayor a $t_{1,5}$ (1.5 veces el tiempo que transcurre en el envío de un carácter) y menor a $t_{3,5}$, la trama del mensaje se declara como incompleta y se descarta por el receptor.

1.6.4 Capa física

En la capa física se define la interfaz física entre los dispositivos. Consta de cuatro especificaciones [8]:

- Mecánicas: relacionadas con las propiedades físicas de la interfaz. Se incluye la especificación del conector que transmite señales.
- Eléctricas: están relacionadas con los niveles de tensión y la velocidad de transmisión.
- Funcionales: definen las funciones que se realiza en cada uno de los circuitos de la interfaz física entre el sistema y el medio de transmisión.
- De procedimiento: especifican la secuencia de eventos en el intercambio de flujo de bits a través del medio físico.

Modbus sobre línea serial incluye en esta capa una especificación de la configuración de la línea y el medio de transmisión, aunque esto se considera debajo de la capa física del modelo OSI [8]. Por ello se inicia con estas especificaciones para, posteriormente, describir las especificaciones mecánicas, eléctricas y funcionales empleadas en el presente trabajo (dado que Modbus sobre línea serial permite elegir entre varias opciones como tipo de conector, velocidad de transmisión, etc.). Finalmente se describen algunas especificaciones de un sistema multipunto que emplea Modbus sobre línea serial.

1.6.4.1 Configuración de la línea

Las características de la configuración de línea son la topología y su funcionamiento half-duplex o full-duplex [8] y a continuación se presentan.

1.6.4.1.1 Topología

La topología en Modbus sobre línea serial es en bus [URL14], es decir, todas las terminales están conectadas directamente a un medio de transmisión (o bus). De esta manera, cuando una terminal transmite datos, éstos se propagan por el bus ocasionando que todas las terminales restantes reciban la trama de datos, pero la trama va dirigida a sólo una terminal, así que las demás terminales ignoran la trama (sección 1.6.3.1). La figura 1.29, muestra un diagrama a bloques de la conexión Modbus con la topología en bus.

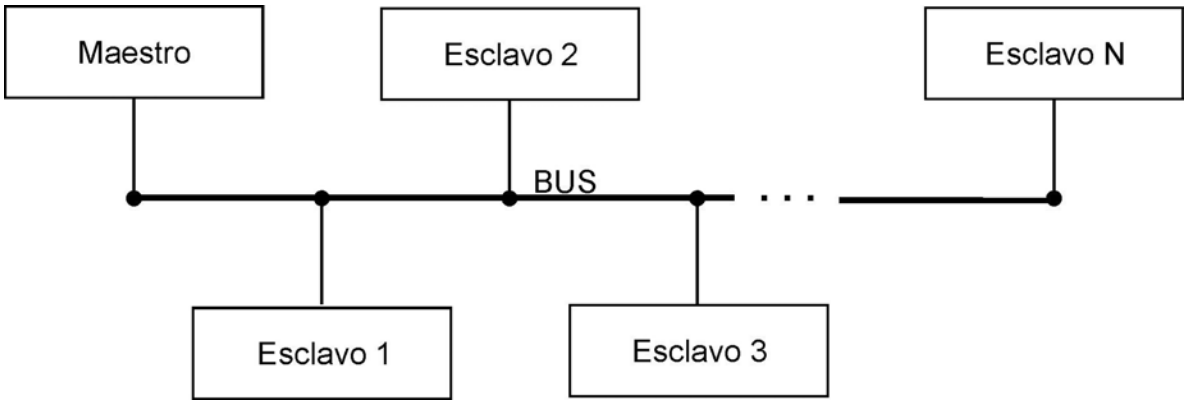

Figura 1.29. Topología en bus.

1.6.4.1.2 Transmisión half-duplex

En Modbus sobre línea serial todas las terminales del sistema multipunto pueden transmitir datos [URL14]. No obstante, en una transmisión half-duplex (o de 2 hilos) sólo una de ellas puede utilizar el medio para trasmitir, es decir, cualquier terminal puede transmitir pero no todas a la vez, de lo contrario habría colisión de datos. Para lograr la transmisión half-duplex, es necesario emplear un par trenzado (sección 1.6.4.2) y un cable de tierra (figura 1.30), cuando una terminal no está transmitiendo debe estar en alta impedancia para la recepción de datos (incluyendo al nodo maestro). Para la transmisión full-duplex (4 hilos) puede consultar la referencia [URL14].

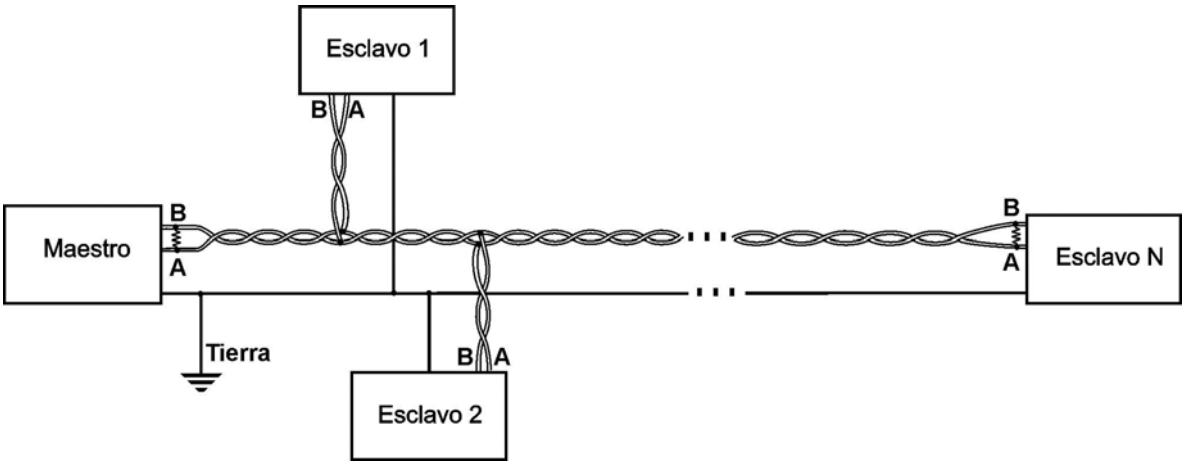

Figura 1.30. Conexión de los nodos en half-duplex.

1.6.4.2 Medio de transmisión

Modbus sobre línea serial especifica que para un sistema multipunto se debe utilizar como medio de transmisión par trenzado, la razón de su uso se explica en la sección 1.6.4.4.1.

1.6.4.3 Especificaciones mecánicas

Se emplea un conector RJ45, la asignación de terminales se ilustra en la figura 1.31 (cada nodo del sistema multipunto debe contar con esta asignación).

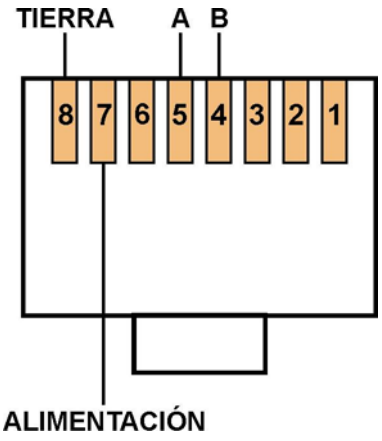

Figura 1.31. Asignación de terminales en un conector RJ45 tipo macho en Modbus sobre línea serial.

1.6.4.4 Especificaciones eléctricas

Modbus sobre línea serial emplea el estándar RS-485 para la comunicación en un sistema multipunto, éste se describe a continuación.

1.6.4.4.1 Estándar RS-485

Modbus sobre línea serial establece que se puede implementar una interfaz eléctrica de acuerdo al estándar EIA/TIA-485 (también conocido como estándar RS-485), o bien, implementar una interfaz RS-232.

Para el intercambio de datos, el estándar RS-232 ha sido uno de los más usados y se ha empleado frecuentemente en el computador para realizar comunicaciones punto a punto. Sin embargo, para lograr una comunicación multipunto, es posible emplear los estándares RS-422 y RS-485. La diferencia principal de éstos con RS-232, es que realizan una transmisión de datos balanceada mientras que en RS-232 los niveles de voltaje son con respecto a tierra.

En la transmisión balanceada se requieren 2 señales opuestas, a las cuales se les llama comúnmente como A y B. Cuando el voltaje en la terminal A (V_A) es mayor (por más de 200mV) que el voltaje en la terminal B (V_B), el dato que se transmite es un 0 binario. Por otra parte, cuando $V_B > V_A$, el dato que se transmite es un 1 binario.

Para la transmisión balanceada se emplea el par trenzado, la finalidad de éste es:

- Reducir radiaciones emitidas por el cable. En el momento de transmisión de datos a altas frecuencias y con longitudes grandes de cable, las líneas actúan como antenas, por lo tanto, emiten radiaciones. Sin embargo, con el par trenzado las señales opuestas tienden a cancelar estas emisiones [URL15].
- Tener inmunidad ante el ruido. Si un agente externo induce ruido, éste se inducirá en las 2 líneas del par trenzado, sin embargo la relación entre la señal A y B se mantendrá. Por ejemplo, si A tiene un voltaje de 2Vcd y B tiene 4Vcd y se induce un voltaje externo de 2Vcd, las señales A y B tendrán 4 y 6Vcd respectivamente, pero la relación A<B se mantiene [URL16].

1.6.4.4.2 Velocidad de transmisión y longitud

Se requiere una velocidad de transmisión de 19.2Kbps por defecto aunque se pueden utilizar opcionalmente otras velocidades de transmisión. La longitud máxima del bus es de 1200 metros, sin embargo, ésta depende de varios aspectos: velocidad de transmisión, calibre del cable, impedancia característica, configuración de la red (2 hilos o 4 hilos). Por otro lado, las derivaciones (cable del bus al nodo) deben ser cortas, no mayores a 20m.

1.6.4.5 Especificaciones funcionales

La tabla 1.11 muestra las especificaciones funcionales de acuerdo al conector RJ45, así como su nivel de requerimiento[URL14].

Terminal en el conector RJ45	Nivel de Requerimiento	Señal	Descripción
4	**Requerido**	B	Si B>A la señal de transmisión es un nivel lógico alto
5	**Requerido**	A	Si A>B la señal de transmisión es un nivel lógico bajo
7	Recomendado	Alimentación	Alimentación positiva de 5...24Vcd
8	**Requerido**	Común	Tierra de las señales B, A y alimentación.

Tabla 1.11. Descripción de las terminales del conector RJ45.

1.6.4.6 Especificaciones de procedimiento

Un nodo o dispositivo final, llamado equipo terminal de datos (DTE, Data Terminal Equipment), hace uso del bus mediante el empleo de un equipo de terminación de circuitos (DCE, Data Circuit-Terminating Equipmet), el cual es el encargado de recibir y transmitir datos [8]. Para que el DTE interaccione con el DCE se requiere el intercambio de datos y de control mediante cables llamados circuitos de intercambio. Sin embargo, Modbus sobre línea serial no especifica el DCE que se debe utilizar, por ello es necesario emplear un dispositivo que realice la conversión de datos binarios a las señales A y B y que cumpla con las secuencias de eventos para la transmisión de datos que, en este caso, son 2: la transmisión y la recepción. El dispositivo que cumple con estos requerimientos y que se emplea en el presente trabajo se describe en la sección 1.6.5.

1.6.4.7 Requerimientos del sistema multipunto en Modbus sobre línea serial

Para un sistema multipunto EIA/TIA-485 se deben aplicar los siguientes requerimientos:

- *Número máximo de dispositivos sin un repetidor*: Se pueden conectar hasta 32 dispositivos en un bus sin repetidores, sin embargo, esto no es estrictamente necesario, ya que hay dispositivos que cuentan con más unidades de carga (unidad que especifica el número de dispositivos que se puede conectar al bus).
- *Terminación de línea*: Con el incremento de la velocidad de transmisión y de la distancia del bus, al transmitir datos se pueden originar reflexiones en el extremo de éste. Las reflexiones son resultado de una discontinuidad de impedancia. Bajo estas condiciones, para minimizarlas se recomienda colocar resistores terminales en cada extremo del bus. El valor de los resistores terminales debe ser idealmente el mismo que el valor de la impedancia característica del cable.
- *Polarización de la línea*: Cuando no se están transmitiendo datos, el bus está en un estado desconocido, esto debido a que todos los nodos están en estado de alta impedancia, dando como resultado que la línea sea susceptible al ruido externo. Para solucionar esta situación se colocan dos resistores al par trenzado, generalmente en la parte del nodo maestro: uno de ellos desde la terminal B a 5V y el otro resistor de la terminal A a tierra. De esta manera cuando el bus está en alta impedancia, se tiene un estado conocido en el estado receptor. El valor de los resistores debe ser entre 450 Ohms y 650 Ohms.

1.6.5 Circuito integrado MAX483E

Se emplea el circuito integrado MAX483E para realizar la interfaz de los nodos esclavo y maestro con el bus. Sus principales características son [URL17]:

- Receptor/Transmisor de comunicaciones con base en el estándar RS-485.
- Comunicación half-duplex.
- Soporta hasta 250Mbps.
- Alimentación de 5v.

En la figura 1.32 se ilustra la configuración de terminales de este circuito integrado.

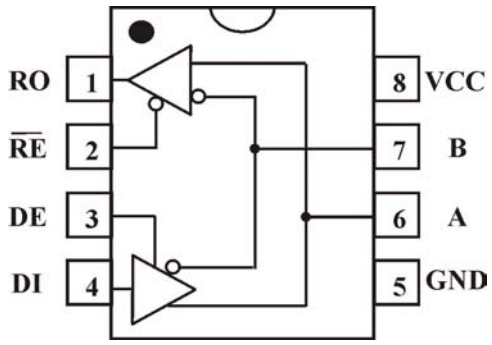

Figura 1.32. Terminales del circuito integrado MAX483E

Descripción de las terminales:

VCC. Alimentación del circuito integrado de 4.75V a 5.25V.

GND. Tierra.

$\overline{\textbf{RE}}$ (Receiver Output Enable). Para habilitar la recepción de datos.

RO (Receiver Output). Terminal de recepción de datos. Si A>B por más de 200mV, RO tendrá un nivel lógico alto y si B>A por más de 200mV RO tendrá un nivel lógico bajo.

DE (Driver Output Enable). Terminal de habilitación del transmisor. Debe tomar un nivel lógico alto en la transmisión de datos y bajo en la recepción.

DI (Driver Input). Entrada para la transmisión.

B. Entrada de recepción inversa y salida de transmisión inversa.

A. Entrada de recepción sin invertir y salida de transmisión sin invertir.

Las siguientes tablas muestran las salidas del circuito dadas las combinaciones de las entradas y el modo de operación.

ENTRADAS			**SALIDAS**	
$\overline{\textbf{RE}}$	**DE**	**DI**	**B**	**A**
X	1	1	0	1
X	1	0	1	0
0	0	X	Alta Impedancia	Alta Impedancia
1	0	X	Alta Impedancia	Alta Impedancia

Tabla 1.12. Salidas en la transmisión del circuito integrado MAX483E.

ENTRADAS			SALIDA
$\overline{RE}$	DE	A-B	RO
0	0	$\geq$+0.2V	1
0	0	$\leq$-0.2V	0
0	0	Entradas Abiertas	Alta Impedancia
1	0	X[5]	Alta Impedancia

Tabla 1.13. Salidas en la recepción del circuito integrado MAX483E.

[5] Donde X representa un valor que no importa.

2. MÓDULOS DE INTERFAZ

Los módulos de interfaz son los controladores de teclado y de video de la tarjeta XSA-100. La función de éstos es servir de interfaz en aplicaciones en las cuales el diseñador de circuitos lógicos, que emplea la tarjeta XSA-100, requiera que el usuario introduzca y visualice datos. A continuación se describe cada módulo.

2.1 MÓDULO CONTROLADOR DE TECLADO

2.1.1 Descripción general

El módulo controlador de teclado se encarga de recibir códigos de rastreo y enviar órdenes por la línea de datos del teclado. Está orientado a teclados con conector PS/2, éstos cuentan con teclas de las siguientes categorías [6]:

1. Teclas de caracteres imprimibles: letras, números y otras como ¿, $, (, ”, etc.
2. Teclas de función extendida: están conformadas por:
    a. Teclas de función de programa (F1, F2, etc.)
    b. Teclas del panel numérico con BloqNum apagado (Inicio, Flechas, Supr, Ins, RéPag y AvPág) y las teclas repetidas en el teclado de 101 teclas.
3. Teclas de control: Alt, Ctrl, y Shift.

Este controlador abarca prácticamente todas las funciones del teclado, es decir, reconoce órdenes, teclas extendidas, teclas de caracteres imprimibles. La implementación del usuario determinará la forma en que se emplea. Por ejemplo, en una aplicación en que sólo requiera teclas numéricas ignorará los códigos de rastreo de las teclas restantes.

El teclado envía un solo código de rastreo por cada tecla. El hecho de que se encienda o apague un LED no significa que automáticamente envíe otro código de rastreo. Por ejemplo, si un usuario presiona la tecla “E” con la intención de obtener el código de rastreo del carácter ‘e’, presiona la tecla BloqMayús para encender el LED correspondiente y de nuevo presiona la tecla “E” pero ahora con la intención de obtener un código de rastreo para la letra mayúscula ‘E’, descubrirá que el código de rastreo es el mismo. Los LEDs del teclado son sólo indicadores visuales, asimismo, no hay que confundir el hecho de que los

códigos de rastreo sólo identifican la tecla presionada y no tienen relación con los códigos ASCII: corresponde a la aplicación que esté haciendo uso del teclado encargarse de realizar la conversión código de rastreo a código ASCII.

2.1.2 Descripción del funcionamiento del controlador de teclado

Para el envío y recepción de datos, el teclado emplea 2 terminales: de datos y de reloj. Por medio de la terminal de datos se intercambia la información con el controlador y, por otro lado, la terminal de reloj la emplea tanto el controlador como el teclado en dicho intercambio. Estas terminales son asignadas a 2 terminales del FPGA de la tarjeta XSA-100 (puede consultar el apéndice E donde se encuentran las asignaciones de las terminales del presente trabajo; o el manual de la tarjeta XSA-100 en la referencia [URL6], la cual contiene las asignaciones de las terminales del teclado en el FPGA).

La terminal de datos en el FPGA debe encontrase normalmente en alta impedancia, de tal forma que la línea de datos tenga un nivel lógico alto (sección 1.4.1) y así el controlador esté listo para recibir en cualquier momento.

Cuando el controlador deba enviar órdenes, sólo debe enviar un bit cero (en la parte del dato que corresponda a un bit cero, como el bit de inicio) en el dato para forzar a que la línea de datos se ponga en un nivel lógico bajo. Esto también se aplica a la línea de reloj. En la figura 2.1 se muestra la descripción de está asignación en VHDL y que forma parte del programa de este controlador, la terminal de datos corresponde DATO_PS y la terminal de reloj es TEC_CLK, ambas están en alta impedancia mientras que las señales internas RELOJ_TECLADO o DATO_TECLADO permanecen en un nivel lógico alto.

```
-------------------------------------------------------------------------------------------------------
-- ENTRADAS / SALIDAS DE LAS LINEAS DE DATOS Y RELOJ DEL TECLADO
-------------------------------------------------------------------------------------------------------
TECA:
tec_clk   <= 'Z' when Reloj_Teclado   = '1' else '0';
dato_Ps  <= 'Z' when Dato_Teclado    = '1' else '0';
```

Figura 2.1. Descripción de las terminales de datos y de reloj del teclado en VHDL.

La descripción del controlador del teclado en VHDL se encuentra en el archivo Teclado.vhd (todos los archivos descritos en VHDL del presente trabajo se encuentran en la carpeta VHDLs del disco adjunto a este documento).

El controlador cuenta con un detector de flancos de bajada, de esta forma se determina que la línea de datos del teclado ha enviado el bit de inicio y comienza la recepción de bits del controlador en una máquina de estados (para la recepción y envío de datos). Al finalizar la recepción la máquina de estados habilita por un ciclo de reloj la señal interna CR_DETECTADO que se emplea en el proceso TEC6, si existió un error en la recepción habilita la señal de ERROR[6].

Por otro lado, si el controlador debe enviar una orden activa una señal llamada ENVIAR_ORDEN y comienza el proceso de envío de bits al teclado en la máquina de estados (sección 1.4.3).

En la recepción de datos o el envío de ordenes se debe obtener la paridad, para ello hay un proceso que cuenta en su lista de sensitividad con una señal interna llamada INICIALIZA_PARIDAD. La figura 2.2 muestra la descripción en la máquina de estados que determina si se deben recibir o enviar bits, así como la indicación de inicializar la paridad (las faltas de ortografía son debido a que el editor de texto del software Foundation no acepta acentos o letras ñ).

```
WHEN INICIAL =>
    if (Flanco_Bajada_Clk_Tec and NOT dato_ps) = '1' then    -- Si hay un flanco de bajada y hay bit de inicio
    inicializa_paridad   <='1';                  -- Inicializa bit de paridad
    CR_Int_Sig(0)        <= dato_ps; -- Almacena el bit recibido
    edo_sig                                                  <= DATO0;
elsif Enviar_Orden = '1' then  -- Si hay una orden lista a enviar al teclado...
    inicializa_paridad   <=          '1';          -- Inicializa bit de paridad
    Retardo_sig          <= '1';     -- Inicio de retardo para la peticion para enviar
    Reloj_Teclado_sig  <= '0';     -- senal de reloj en bajo
    Edo_Sig              <= PETICION_PARA_ENVIAR;
end if;
```

Figura 2.2. Descripción de la acción a realizar en el controlador del teclado en VHDL.

[6] En todo el documento se considera que una señal se habilita cuando pasa de un estado lógico bajo a un estado lógico alto, a menos que se indique lo contrario

El proceso TEC6 (figura 2.3) desecha el dato recibido si corresponde al código de liberación, de otra forma, activa la señal interna TECLA_PRESIONADA que se emplea en el proceso TEC7.

```
TEC6:process(clk, reset)
begin
    if reset='1' then
        Tecla_Presionada  <= '0';     -- Por defecto, no hay datos recibidos
        Tecla_Liberada    <= '0';
    elsif clk'event and clk = '1' then
        Tecla_Presionada  <= '0';     -- Inicializa indicador de codigo de rastreo (CR)
        if CR_DETECTADO= '1' then              -- Si se detecta un CR
            Tecla_Presionada          <= '1';     -- Indica que se esta presionando la tecla
            if CR(8 downto 1) = F0 then -- Si detecta que el CR es un el 1er byte del codigo de liberacion...
                Tecla_Presionada      <= '0';     -- No esta listo el CR de salida
                Tecla_Liberada        <= '1';     -- Activa senal a '1' para esperar el 2o byte del codigo de liberacion
            elsif Tecla_Liberada = '1' then        -- Si despues de F0 se recibe el siguiente CR...
                Tecla_Presionada      <= '0';     -- No se toma en cuenta ese CR
                Tecla_Liberada        <= '0';
            end if;
        end if;
    end if;
end process;
```

Figura 2.3. Descripción del controlador de teclado que detecta
si se está presionando una tecla o se ha soltado.

El proceso TEC7 utiliza la señal interna TECLA_PRESIONADA para verificar si el código de rastreo corresponde a alguna tecla Bloq Mayús, Bloq Núm o Bloq despl (figura 2.4); si el dato recibido no corresponde a dichas teclas se habilita por un ciclo de reloj la salida CODIGO_RASTREO_LISTO, indicando que el bus de salida CODIGO_RASTREO contiene el código de rastreo. Por otra parte, al alimentar el FPGA con voltaje, el proceso TEC7 ordena al proceso TEC5 (donde se encuentra la máquina de estados de recepción de datos y envío de órdenes) que programe al teclado con el conjunto de códigos de rastreo 3. El proceso TEC7 da también la orden del envío de la orden correspondiente si el código de rastreo recibido es de la tecla Bloq Mayús, Bloq Núm o Bloq Despl, con esto se apagan o se encienden los LEDs y el bus de salida CNS tiene el estado de éstos: el bit más significativo corresponde al estado del LED de la tecla Bloq Mayús, el siguiente corresponde al estado LED de la tecla Bloq Núm y el bit menos significativo corresponde al estado del LED Bloq Despl.

```
IF Tecla_Presionada ='1' THEN -- Si aparece un CR de teclado
   CASE CR (8 DOWNTO 1) IS
      WHEN OK | NOK => -- AA prueba exitosa de reset o FC (no exitosa al conectar mal el teclado)
         LEDS          := (OTHERS =>'0');
         ORDEN         <= SSCS; -- orden de asignacion del conjunto de codigos de rastreo (Set Scan Code Set)
         Enviar_ORDEN <= '1';
         AUXILIAR      <= "00000010"; -- Guarda byte 0x02 que indicara la accion a realizar despues de la orden
                                      -- de asignación de conjunto de codigos de rastreo (SSCS)
      WHEN CAPS | NUM | SCROLL =>     -- Si es SCROLL, NUM o CAPS...
         ORDEN         <= SR_LEDS;     -- Orden para encender/apagar los LEDS
         Enviar_ORDEN <= '1';          -- Indica que se envie la orden
         AUXILIAR      <= CR(8 downto 1);-- Guarda CR
      WHEN ACK_T =>  -- Byte de reconocimiento del teclado
         AUXILIAR      <= (others=>'1');    --Incializa AUXILIAR para que en sig. ciclo entre a others del case
         IF AUXILIAR=CAPS THEN
            LEDS(2):=NOT LEDS(2); ORDEN <="00000"& LEDS; Enviar_ORDEN<='1';
         ELSIF AUXILIAR=NUM THEN
            LEDS(1):=NOT LEDS(1); ORDEN <="00000"& LEDS; Enviar_ORDEN<='1';
         ELSIF AUXILIAR=SCROLL THEN
            LEDS(0):=NOT LEDS(0); ORDEN <="00000"& LEDS; Enviar_ORDEN<='1';
         ELSIF AUXILIAR="00000010" THEN -- Despues del reset y de la orden el SSCS...
            Enviar_ORDEN <='1';
            ORDEN  <=  SCS3;  -- Orden de asignar el conjunto de codigos de rastreo 3
         END IF;
      WHEN OTHERS => Codigo_Rastreo_LISTO<='1'; -- Activa la bandera de CR listo
   END CASE;
END IF;
```

Figura 2.4. Descripción en el controlador de teclado que determina la orden a enviar, el estado de los LEDs o si se ha recibido un código de rastreo.

La figura 2.5 muestra la descripción en el controlador de la asignación de la salida que contiene el código de rastreo capturado y la salida que se habilita cuando esto sucede.

```
-------------------------------------------------------------------------------------------
--          SALIDAS DE CODIGO DE RASTREO DEL MODULO
-------------------------------------------------------------------------------------------
TECB:
CR_Listo          <= Codigo_Rastreo_LISTO;  -- Indica que el Codigo de Rastreo esta listo
Codigo_Rastreo    <= CR(8 downto 1);        -- Codigo de Rastreo de Salida
```

Figura 2.5. Parte de la descripción del controlador de teclado que asigna las salidas de código de rastreo e indicación de código de rastreo listo.

La figura 2.6 muestra el diagrama de flujo para la recepción de códigos de rastreo y envío de órdenes.

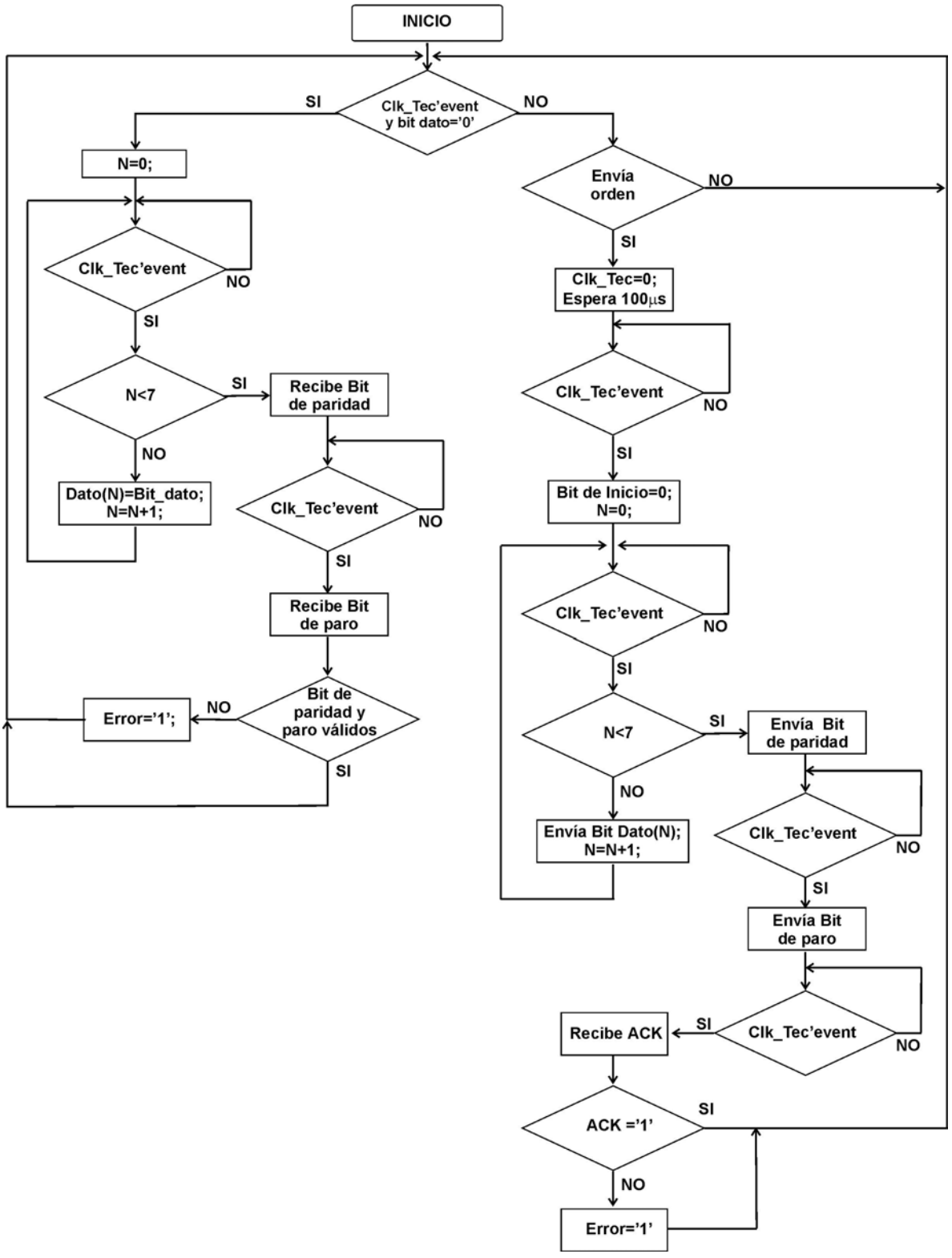

Figura 2.6 Diagrama de flujo de la recepción y envío de datos del teclado

2.1.3 Entradas y salidas

En la figura 2.7 se muestra un símbolo que representa las entradas y salidas del módulo controlador del teclado. El bloque puede ser considerado como una caja negra que actúa conforme a las entradas y, con base en ello, obtiene salidas.

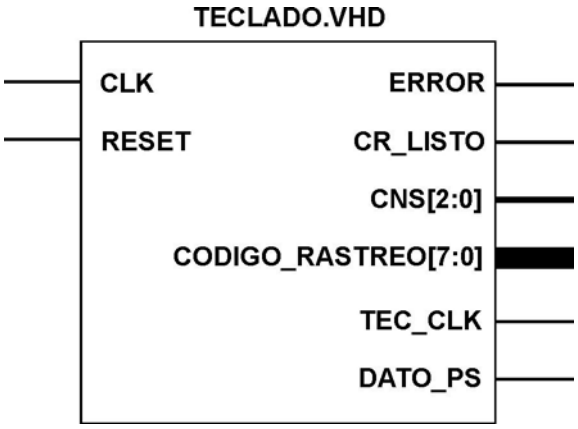

Figura 2.7. Símbolo del controlador del teclado.

Entradas:

- CLK. Frecuencia de reloj a la que opera el módulo.
- RESET. Entrada para aplicar reset asíncrono al módulo

Salidas:

- CNS[2:0]. Bus de 3 bits que indica qué LEDs del teclado están encendidos o apagados.
- CODIGO_RASTREO[7:0]. Bus de 8 bits el cual contiene el código de rastreo correspondiente a la tecla presionada.
- CR_LISTO. Se habilita por un ciclo de reloj cuando termina la recepción del un código de rastreo.
- ERROR. Se habilita por 1 ciclo del reloj cuando se detecta un error de paridad o de bit de paro.

Entradas/Salidas:

- TEC_CLK. Línea de reloj del teclado.
- DATO_PS. Línea de datos del teclado.

2.2 MÓDULO CONTROLADOR DE VIDEO

2.2.1 Descripción general

El módulo controlador de video es alfanumérico, es decir, sólo muestra caracteres en la pantalla. Para ello, se encarga de generar las señales necesarias para desplegar los píxeles, éstas son: las señales RGB y los pulsos de sincronización.

2.2.2 Descripción del funcionamiento del controlador de video

A continuación se presentan algunos tópicos útiles para comprender el funcionamiento de este controlador.

2.2.2.1 Frecuencia de reloj y resolución

La resolución que puede ofrecer el controlador depende de la resolución del monitor (por ejemplo 640x480) y de la frecuencia de reloj a la que trabaja.

Los tiempos que se emplean para aplicar las señales en una exploración horizontal (tiempo de blanqueo horizontal, tiempo de sincronización horizontal, tiempo de despliegue de píxeles, etc.) pueden expresarse en términos de cantidad de ciclos de reloj. Para esto, se divide cada uno de los tiempos que participan en la exploración horizontal (THor) entre el periodo de la señal de reloj a la que funciona el controlador (Ec.2.1). Determinar la cantidad de ciclos es importante para aplicar el pulso de sincronización horizontal, enviar la cantidad correcta de píxeles al monitor y realizar el blanqueo horizontal.

$$\text{Cantidad de Ciclos} = \frac{\text{THor}}{\text{Periodo de la frecuencia principal}} \qquad \text{(Ec.2.1)}$$

Asimismo, para obtener la cantidad de líneas correspondientes a los tiempos de las señales que participan en la exploración vertical (tiempo de despliegue de líneas, tiempo de blanqueo vertical, tiempo de sincronización vertical, etc.), se divide cada uno de los tiempos que participan en la exploración vertical (TVert) entre el tiempo total invertido en una exploración horizontal (Ec.2.2). Al determinar el número de líneas se conoce el número de líneas de exploración se pueden desplegar en la pantalla, además el número de líneas para aplicar la señal de sincronismo y de blanqueo vertical.

$$\text{Cantidad de Líneas} = \frac{\text{TVert}}{\text{Tiempo en que se realiza la exploración horizontal}} \qquad \text{(Ec.2.2)}$$

Por ejemplo, supóngase que el controlador emplea una frecuencia de reloj de 12.5Mhz, esto implica que el periodo de la señal es 80 ns. El tiempo en que se puede explorar una línea horizontal y que sea visible en la pantalla es de 25.17 μs, de la ecuación 2.1 se puede concluir que se pueden desplegar 314 píxeles (25.17 μs/80 ns) como máximo, para ello en cada flanco de la señal de reloj se muestra un píxel en la pantalla. De igual forma, el tiempo para desplegar las líneas es de 15.25 ms, de la ecuación 2.2 el resultado es de 480 líneas visibles en pantalla (15.25 ms/31.77 μs). Por lo tanto, la resolución con esta frecuencia es de 314x480, lo que quiere decir que si el monitor es de resolución de 640x480 cada píxel que se muestre en pantalla corresponde a aproximadamente 2 píxeles en la exploración horizontal.

La tabla 2.1 muestra de manera más detallada, usando la ecuación 2.1, la cantidad de ciclos requeridos para los tiempos en la exploración horizontal a 12.5 MHz.

Tiempo (μs)	**Descripción**	**Tiempo dividido entre 80 ns**	**Cantidad de ciclos**
31.77	Tiempo de exploración horizontal.	397.12	397
25.17	Tiempo máximo en que se puede desplegar píxeles en una línea.	314.62	314
0.94	Tiempo mínimo antes de aplicar el pulso de sincronización horizontal y después de desplegar el último píxel.	11.75	12
3.77	Tiempo en que el pulso de sincronización horizontal debe permanecer en un nivel lógico bajo.	47.125	47
1.89	Tiempo mínimo antes de terminar el blanqueo horizontal.	23.625	24

Tabla 2.1. Cantidad de ciclos que generan las señales de exploración horizontal a 12.5Mhz.

La tabla 2.2 resume estos resultados a una frecuencia de reloj de 25MHz.

Tiempo (µs)	**Descripción**	**Tiempo dividido entre 40 ns**	**Cantidad de ciclos**
31.77	Tiempo de exploración horizontal.	794.25	794
25.17	Tiempo máximo en que se puede desplegar píxeles en una línea.	629.25	629
0.94	Tiempo mínimo antes de aplicar el pulso de sincronización horizontal y después de desplegar el último píxel.	23.5	24
3.77	Tiempo en que el pulso de sincronización horizontal debe permanecer en un nivel lógico bajo.	94.25	94
1.89	Tiempo mínimo antes de terminar el blanqueo horizontal.	47.25	47

Tabla 2.2. Cantidad de ciclos que generan las señales de exploración horizontal a 25Mhz.

De igual forma, y empleando la ecuación 2.2, la tabla 2.3 tiene la cantidad de líneas necesarias para generar las señales en la exploración vertical.

Tiempo (ms)	**Descripción**	Tiempo dividido entre **31.77 µs**	**Cantidad de líneas**
16.784	Tiempo para realizar la exploración vertical.	528.29	528
15.25	Tiempo máximo en que se pueden desplegar todas las líneas en la pantalla.	480.01	480
0.45	Tiempo mínimo antes de aplicar el pulso de sincronización vertical y después de desplegar el último píxel.	14.16	14
0.064	Tiempo en que el pulso de sincronización vertical debe permanecer en un nivel lógico bajo.	2.01	2
1.02	Tiempo mínimo antes de terminar el blanqueo vertical.	32.1	32

Tabla 2.3. Cantidad de líneas que generan las señales de exploración vertical.

2.2.2.2 Consideraciones del controlador de video

Para implementar el controlador de video alfanumérico, se consideran los datos obtenidos de las tablas 2.2 y 2.3 para determinar la cantidad máxima de caracteres que pueden aparecer en la pantalla.

En adelante, para explicaciones posteriores, a la cantidad de caracteres que se muestran horizontalmente en la pantalla se les hará referencia como cantidad de columnas; y a la cantidad de caracteres que aparezcan verticalmente se les llamará renglones. Por ejemplo, en la figura 2.8 hay 4 renglones de caracteres imprimibles en pantalla: en el primero hay 4 columnas de caracteres imprimibles y en el resto sólo hay una columna. Los caracteres restantes de la pantalla corresponden al carácter del espacio vacío.

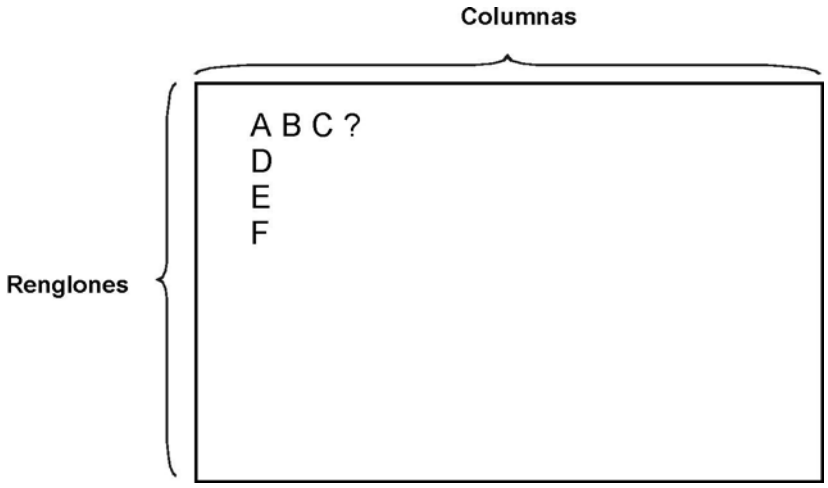

Figura 2.8. Muestra de caracteres visibles en la pantalla

Asimismo, se emplearán matrices de 8x16 píxeles (figura 2.9). Donde al conjunto de píxeles horizontales se le llamará fila, por lo tanto, son 16 filas por matriz, donde la fila inicial se referencia como 0 y se encuentra en la parte superior de la matriz, y la última fila es la 15.

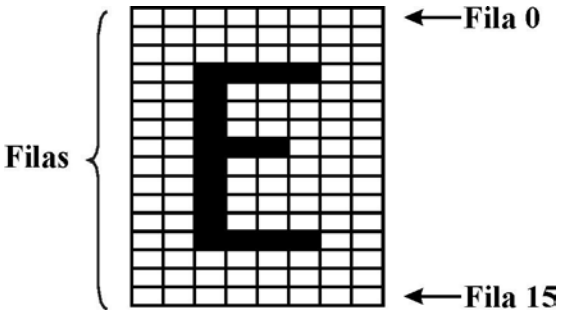

Figura 2.9. Matriz de 8x16 píxeles.

Por último, la frecuencia del controlador desarrollado es de 25MHz, sin embargo, los resultados obtenidos pueden extenderse a otras frecuencias.

2.2.2.3 Cantidad de columnas y renglones

La tabla 2.2 contiene el número de ciclos necesarios para obtener los tiempos de interés descritos a una frecuencia de 25Mhz. Estos resultados son necesarios para obtener, de nueva cuenta, los tiempos pero ahora en términos de cantidad de columnas. Dado que cada columna requiere de 8 píxeles, la cantidad de columnas se obtiene como sigue:

$$\text{Cantidad de Columnas} = \frac{\text{Cantidad de ciclos de la señal}}{8} \qquad \text{(Ec.2.3)}$$

Así, de la ecuación anterior se obtienen los resultados de la tabla 2.4.

Tiempo (µs)	Descripción	Cantidad de ciclos	Cantidad de ciclos entre 8	Cantidad de columnas
31.77	Tiempo de exploración horizontal.	794	99.25	99
25.17	Tiempo máximo en que se puede desplegar píxeles en una línea.	629	78.625	78
0.94	Tiempo mínimo antes de aplicar el pulso de sincronización horizontal y después de desplegar el último píxel.	24	3	3
3.77	Tiempo en que el pulso de sincronización horizontal debe permanecer en un nivel lógico bajo.	94	11.75	12
1.89	Tiempo mínimo antes de terminar el blanqueo horizontal.	47	5.875	6

Tabla 2.4. Cantidad de columnas que generan las señales de exploración horizontal a 25 Mhz.

Se puede observar en la tabla 2.4 que es posible desplegar hasta 78 columnas como máximo en un renglón.

Para obtener la cantidad de renglones se realiza el mismo procedimiento, sólo que se toman las cantidades de líneas necesarias para generar la exploración vertical y se dividen entre la cantidad de filas que hay en la matriz, en este caso entre 16 (ecuación 2.4).

$$\text{Cantidad de Renglones} = \frac{\text{Cantidad de líneas de la señal}}{16} \qquad \text{(Ec.2.4)}$$

La tabla siguiente muestra los resultados de estos cálculos.

Tiempo (ms)	Descripción	Cantidad de líneas	Cantidad de líneas entre 16	Cantidad de renglones
16.784	Tiempo para realizar la exploración vertical.	528	33	33
15.25	Tiempo máximo en que se pueden desplegar todas las líneas en la pantalla.	480	30	30
0.45	Tiempo mínimo antes de aplicar el pulso de sincronización vertical y después de desplegar el último píxel.	14	0.875	1
0.064	Tiempo en que el pulso de sincronización vertical debe permanecer en un nivel lógico bajo.	2	0.125	1
1.02	Tiempo mínimo antes de terminar el blanqueo vertical.	32	2	1

Tabla 2.5. Cantidad de renglones que generan las señales de exploración vertical.

De las 2 tablas anteriores se concluye que, dado que la cantidad máxima de columnas que se pueden desplegar en la pantalla es de 78 y la de renglones es 30, entonces es posible mostrar en la pantalla hasta 78x30=2340 caracteres.

La obtención de la cantidad de columnas y de renglones es necesaria para, en la descripción en VHDL, utilizar contadores de estos parámetros y aplicar con base en ellos las señales necesarias en la generación de la imagen en la pantalla, pero no sólo eso, en la sección siguiente se explica como estos parámetros también se emplean para tener acceso a la RAM de refresco y la ROM generadora de caracteres.

En la figura 2.10 se muestran las señales de video con base en la cantidad de columnas y renglones, la figura 2.10(a) muestra la cantidad de columnas que se requieren para cumplir con los tiempos de la exploración horizontal, a su vez, la figura 2.10(b) muestra la cantidad de renglones necesarios para cumplir con la exploración vertical.

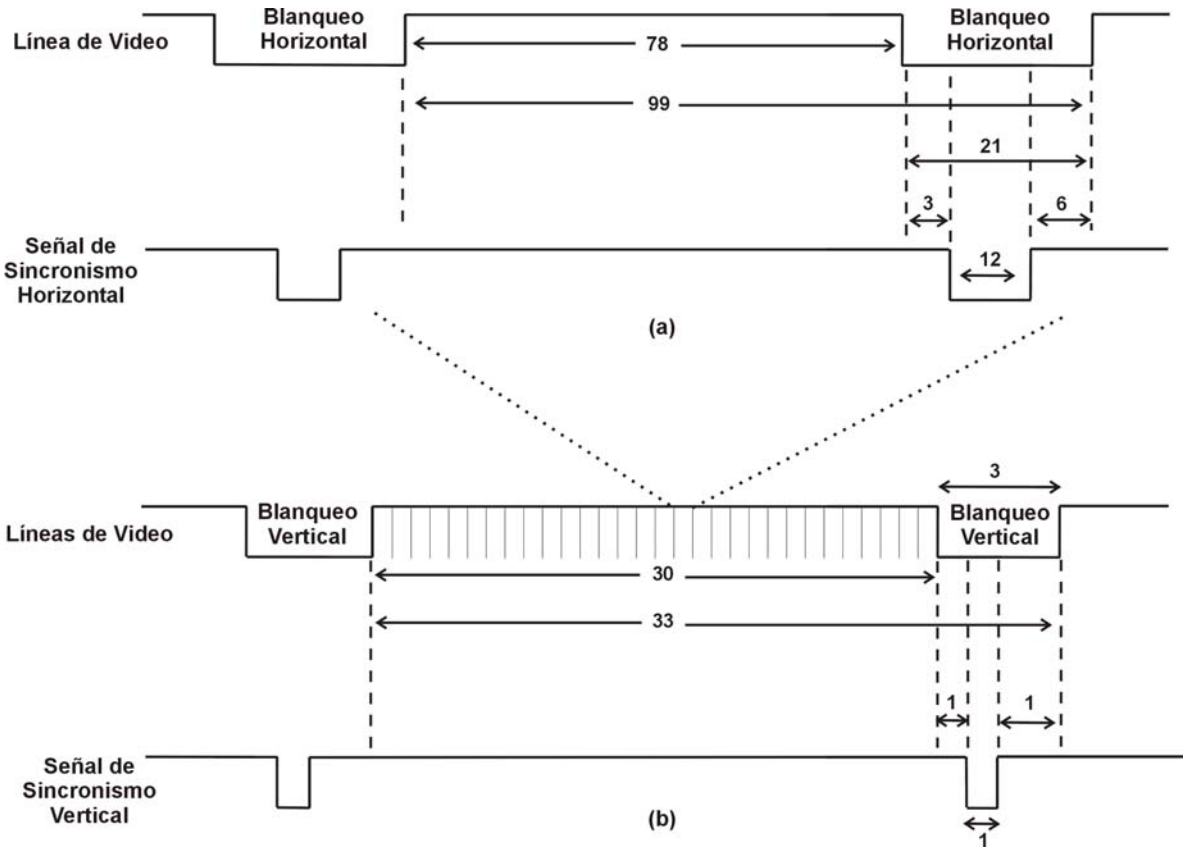

Figura 2.10. Señales de video en términos de: (a) cantidad de columnas, (b) cantidad de renglones.

2.2.2.4 Acceso a la RAM de refresco de pantalla y la ROM generadora de caracteres

Este controlador de video no tiene acceso directo a memoria, para obtener el valor de los píxeles a desplegar se emplea el módulo de Memoria que se explicará en el capítulo 4 (Instanciación de Componentes). Sin embargo, el controlador de video proporciona el bus de direcciones para que el módulo de Memoria lea los píxeles correctos.

Se hace uso de 4 contadores anidados para obtener los píxeles que se deben desplegar, la dirección de acceso en la RAM de video y los pulsos de sincronización y blanqueo. Éstos son:

- Contador de Píxeles (PixelCnt): Se incrementa cada flanco de subida de reloj. Realiza un conteo desde cero hasta el número máximo de píxeles que hay en cada fila de la matriz que representa un carácter, en este caso en la matriz de 8x16 este contador alcanzará como máximo el 7 y después se reinicia a 0.

- Contador de Columnas (ColCnt): Se incrementa cada vez que el contador de píxeles llega a 0 y su valor máximo es 98. Indica la columna en la pantalla en la cual el carácter se está desplegando en el monitor en el momento actual.

- Contador de Filas (FilaCnt): Se incrementa en uno cada vez que el contador de columnas es cero, su valor máximo es 15. Representa la fila de la matriz que se está refrescando actualmente.

- Contador de Renglones (RengCnt): Se incrementa en uno cada vez que el contador de filas se reinicie a cero, su valor máximo es 32. Representa el renglón en el que se encuentra el carácter que se está desplegando en la pantalla.

El diagrama de flujo de estos contadores se representa en las figuras 2.11(a) y 2.11(b). En la figura 2.11(a) hay un arreglo llamado Pixel que contiene una de las filas de la matriz de píxeles y que tiene como índice el contador PixelCnt para mostrar de su contenido en cada flanco de la señal de reloj. Cuando el contador PixelCnt vuelve a ser cero, la matriz Pixel debe contener la fila de otro carácter.

Nótese también que los blanqueos horizontal y vertical, así como las señales de sincronización, dependen de los valores de los contadores; todos estos procesos se describen en las siguientes secciones.

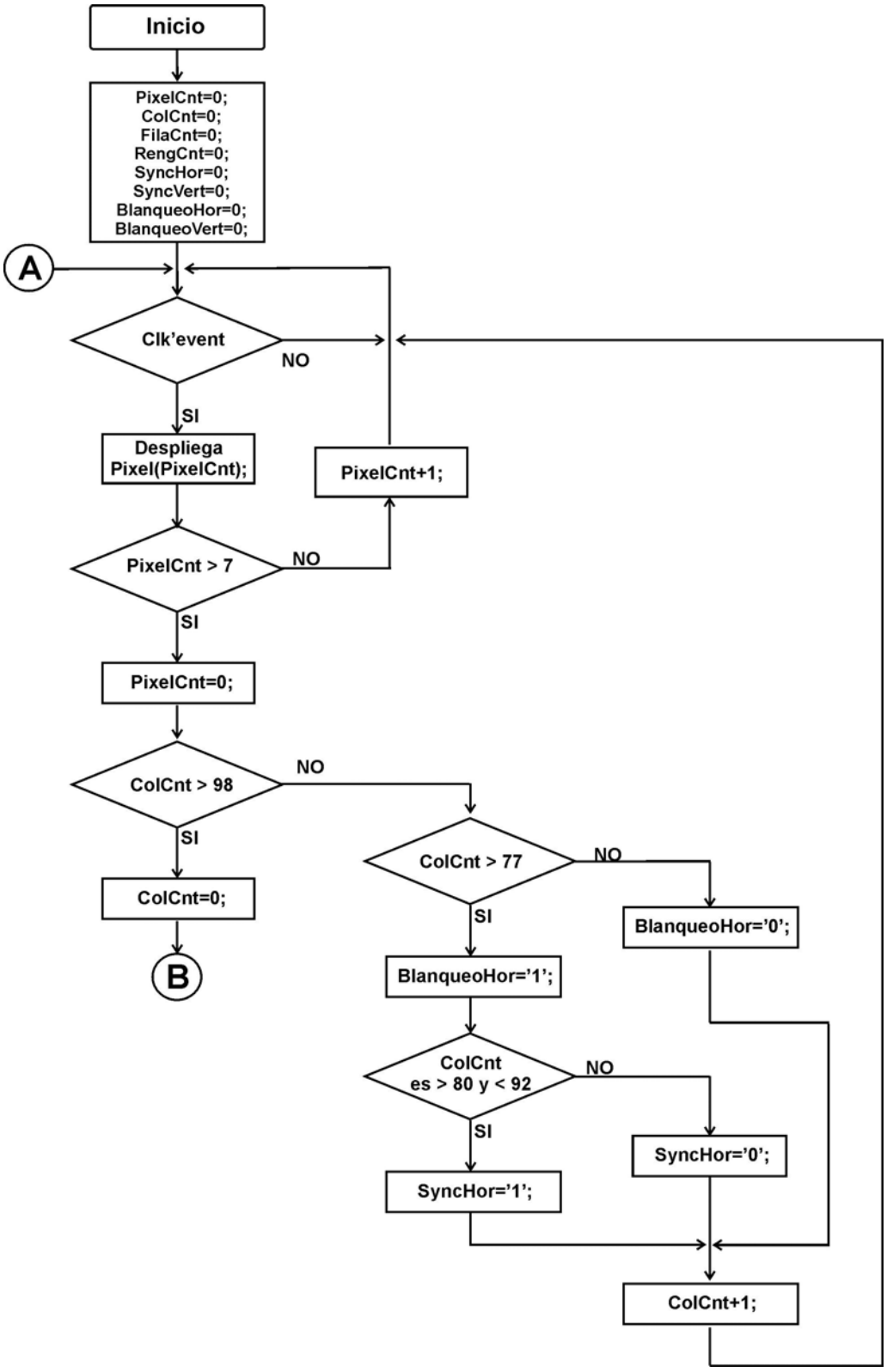

Figura 2.11 (a) Primera parte del diagrama de flujo de los contadores anidados que generan las señales de video.

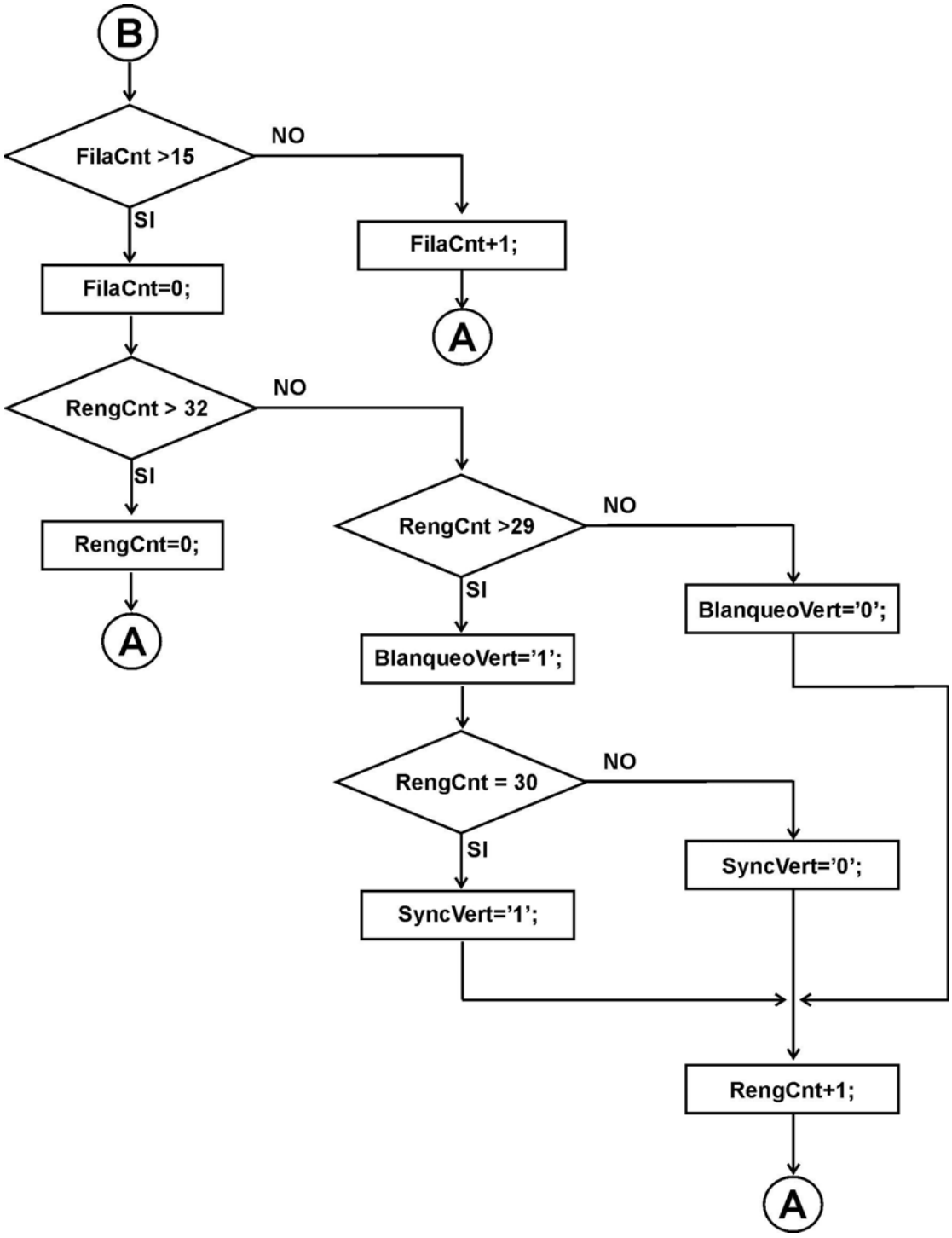

Figura 2.11 (b) Segunda parte del diagrama de flujo de los contadores anidados que generan las señales de video.

2.2.2.5 Pulsos de sincronización y de blanqueo

Es posible observar de la figura 2.10(a) y 2.11(a) que el blanqueo horizontal se aplica cuando el contador de columnas (ColCnt) llega a 78, y tiene una duración que corresponde a 21 incrementos de columnas. De manera análoga, el pulso de sincronización horizontal se aplica cuando ColCnt llega a 81 y permanece en un nivel lógico bajo mientras que ColCnt se incrementa 12 columnas más. Una nueva línea de exploración comienza cuando ColCnt llega a 99. Estos pasos se aplican de manera similar para las señales de sincronización y blanqueo vertical pero con el contador de renglones.

La figura 2.12 muestra la descripción en VHDL que aplica los pulsos de sincronización del controlador de video.

```
----------      OBTIENE SENAL DE SINCRONIZACION HORIZONTAL
VGA2: process(clk_vga,reset)
begin
    -- Reset asincrono establece la senal de sincronizacion horizontal a inactiva
    if reset='1' then
        Sync_Hor <= '1';
        -- La senal horizontal sync se calcula cada ciclo de reloj
        elsif (clk_vga'event and clk_vga='1') then
        if (colcnt>TO_UNSIGNED(80,colcnt'length) and colcnt<TO_UNSIGNED(92,colcnt'length)) then
            Sync_Hor <= '0';
        else
            Sync_Hor <= '1';
        end if;
    end if;
end process;

----------      OBTIENE SENAL DE SINCRONIZACION VERTICAL
VGA3: process(clk_vga,reset)
begin
    -- Reset asincrono que establece la senal vertical de sincronizacion a inactiva
    if reset='1' then
        Sync_Vert <= '1';
    elsif (clk_vga'event and clk_vga='1') then
        if rengcnt=TO_UNSIGNED(30,rengcnt'length) then
            Sync_Vert <= '0';
        else
            Sync_Vert <= '1';
        end if;
    end if;
end process;
```

Figura 2.12. Descripción para aplicar los pulsos de sincronización en VHDL.

2.2.2.6 Bus de direcciones

La dirección de la RAM de refresco de pantalla, empleada en el módulo de Memoria, está conformada por los contadores de renglones y columnas:

Dir_RAM_refresco <= RengCnt (4 downto 0) &ColCnt (6 downto 0); (Ec.2.5)

En VHDL el carácter '&' corresponde al operador de concatenación de señales y es aplicable a bits o vectores siempre que exista compatibilidad del tamaño y tipo de los operandos. Por su parte, el operador '<=' se emplea para realizar la asignación de señales.

Si las posiciones de los caracteres de acuerdo al renglón y columna se ubican como coordenadas (x,y), entonces el valor ColCnt corresponde a la posición **x** en la que se encuentra el carácter en la pantalla, mientras que el valor de RengCnt indica la posición **y** en el monitor.

La tabla 2.6 contiene algunos ejemplos de la concatenación de los contadores ColCnt y RengCont para formar el bus de direcciones de la RAM de refresco dada una posición del carácter en la pantalla.

Posición	**Renglón**		**Columna**		**Dirección**
(x,y)	**Decimal**	**Binario**	**Decimal**	**Binario**	**Hexadecimal**
(0,0)	0	00000	0	0000000	000
(0,1)	0	00000	1	0000001	001
(0,77)	0	00000	77	1001101	04D
(1,0)	1	00001	0	0000000	080
(1,77)	1	00001	77	1001101	0CD
(29,0)	29	11101	0	0000000	E80
(29,77)	29	11101	77	1001101	ECD

Tabla 2.6. Concatenación de algunos valores de ColCnt y RengCnt.

Cuando los contadores exceden la cantidad de columnas y renglones visibles en la pantalla, se realiza el blanqueo y no es necesario el acceso a memoria.

La figura 2.13 muestra las localidades RAM de refresco de pantalla que se utilizan para almacenar los caracteres que se despliegan en la pantalla, las líneas diagonales indican la parte de memoria que no se emplea, ya que en ese lapso los contadores se utilizan para realizar el blanqueo. Nótese como algunas de las localidades están conformadas por la concatenación de la tabla anterior.

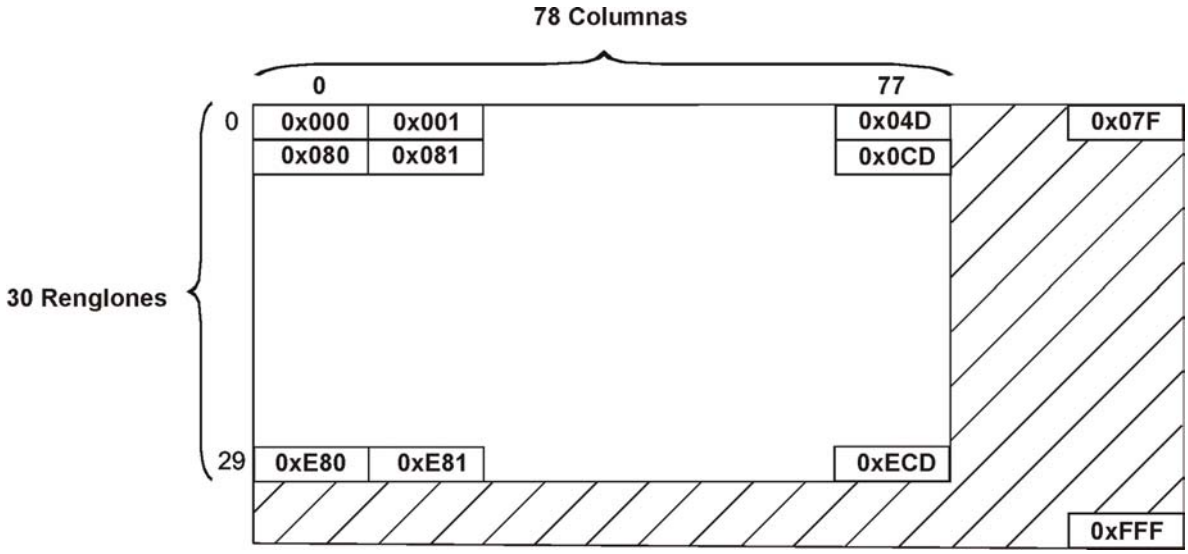

Figura 2.13. Localidades de la RAM de refresco que almacenan caracteres.

En cada localidad de la RAM de refresco se almacena código ASCII, éste se emplea en conjunción del contador de filas (FilaCnt) para formar el bus de direcciones que indica la dirección a leer en la ROM generadora de caracteres y así obtener los bits de la matriz de píxeles que se encuentra (ecuación 2.6).

Dir_ROM_caracteres <= Codigo_ASCII (7 downto 0) & FilaCnt (3 downto 0); (Ec.2.6)

Por ejemplo, para la fila 7 del carácter ‘E’, la dirección de la ROM es:

Dir_ROM_caracteres <= “01000101” & “0111”;

De la figura 2.9, el contenido de la fila 7 es “00111000”, este dato está almacenado en la ROM generadora de caracteres en la dirección obtenida recién concatenada.

Con el fin de ejemplificar esto para una cadena de caracteres en la pantalla, se describen los pasos requeridos para mostrar la palabra HOLA en la parte superior la pantalla (figura 2.14), con el acceso a memoria realizado por el módulo de Memoria.

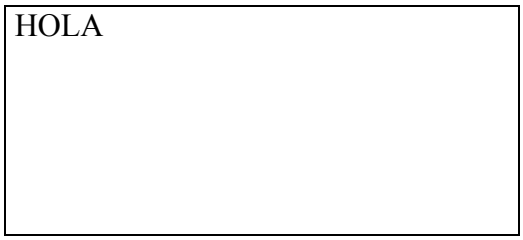

Figura 2.14. Palabra para el ejemplo mostrada en la pantalla del monitor.

Para comenzar a desplegar en la esquina izquierda de la parte superior de la pantalla, todos los contadores se inicializan con 0. Así el contador de columnas y el contador de renglones tienen cero y por lo tanto, la posición (x,y) en la pantalla es (0,0), en esta posición de la RAM de refresco se encuentra el 1er código ASCII, es decir, el código ASCII correspondiente al carácter ‘H’, de la ecuación 2.5 el bus de direcciones para la RAM de refresco tendrá:

Dir_RAM_refresco <= “00000” & “000000”;

Ya que el módulo de Memoria ha leído el código ASCII del 1er carácter (0x48), éste se emplea con el contador de filas como parte del bus de direcciones (ecuación 2.6) para obtener los píxeles de la ROM generadora de caracteres. Por lo tanto la dirección ROM será:

Dir_ROM_caracteres <= “01001000” & “0000”;

Como resultado de este proceso, la lectura de la ROM generadora de caracteres es de la fila inicial de la matriz (FilaCnt 0), es decir, un conjunto de 8 bits de 0’s (figura 2.15).

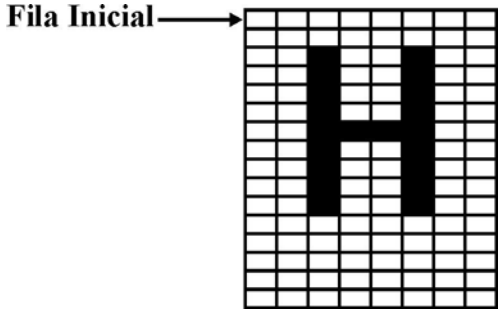

Figura 2.15. Matriz de puntos del carácter 'H' indicando el contenido de la fila inicial.

Después de desplegar los 8 píxeles de la fila inicial de la posición (0,0), el contador PixelCnt regresa a su valor inicial 0 y se incrementa en 1 el contador ColCnt. Ahora el carácter al que se hace referencia en la RAM de refresco de pantalla está en la posición (0,1) y corresponde a la letra 'O'. De nuevo con la ecuación 2.5 se obtiene el código ASCII de esa posición con los contadores ColCnt y RegCnt:

Dir_RAM_refresco <= "00000" & "000001";

El código ASCII para el segundo carácter (0x4F) junto con el contador de filas forman el contenido del bus de direcciones de la ROM generadora de caracteres. Debido a que el contador de filas de la matriz es todavía 0 (ya que se incrementa hasta que el contador de columnas vuelva a ser cero), la ROM tendrá en su salida el patrón de puntos de 8 bits en la fila 0 de la matriz. La dirección de la ROM es:

Dir_ROM_caracteres <= "01001111" & "0000";

Después de desplegar los 8 bits de la matriz, se incrementa ColCnt en 1 para que el leer el carácter en la posición (0,2), es decir, la letra 'L', nuevamente se obtienen los bits de la fila inicial de la matriz. El proceso continua hasta que el contador de columnas llega a su valor máximo para desplegar en la pantalla (77), es decir, se alcanza la parte superior derecha de la pantalla.

Cuando el contador de columnas llega a 78 se produce un pulso de blanqueo para poner la pantalla en negro. Esta es una fracción muy pequeña de tiempo por lo que no es visible al ojo humano. Dentro de este tiempo se produce un pulso de sincronización horizontal para causar que el haz regrese a la parte izquierda de la pantalla. Los contadores siguen incrementándose pero en la pantalla no hay despliegue. En el momento que el contador de columnas vuelve a ser 0 el haz de electrones se encuentra en la siguiente línea de exploración, es decir, en la posición (0,0) pero ahora el contador de filas se incrementa a 1.

El proceso se repite en la segunda línea de exploración (FilaCnt es 1), ahora los píxeles que se obtienen de las matrices corresponden a la segunda fila.

Cuando el haz de electrones llega a la tercer línea de exploración, los contadores de columnas, renglones y píxeles son 0, pero el contador de filas, FilaCnt, vale 2. Se obtiene el código ASCII de la posición (0,0) (del carácter ‘H’), y con él, como en los pasos anteriores, se obtienen los bits de la ROM generadora de caracteres:

Dir_ROM_caracteres <= “01001000” & “0010”;

Ahora los bits que se leen de la primer matriz son 00100100, éstos corresponden a la segunda fila del carácter ‘H’. Los píxeles se muestran en la pantalla y posteriormente se obtienen los bits de las matrices restantes.

Al final de desplegar todas las filas de las matrices del primer renglón los contadores PixelCnt, ColCnt y FilaCnt son 0, pero el contador RengCnt se incrementa en 1. Ahora se hace referencia a la posición (1,0), en este caso el carácter es un espacio en blanco y todos los bits de esta matriz son 0. El proceso se repite hasta mostrar los 30 renglones y las 78 columnas. Cuando termina de desplegar el último píxel de la última fila del carácter de la posición (29,77) se realiza un blanqueo vertical y la pantalla se apaga, el haz regresa a la parte izquierda de la parte superior de la pantalla, ahí la posición es (0,0) y se empieza de nuevo.

2.2.2.7 Color en la pantalla

En la tarjeta XSA-100 se encuentra un convertidor DAC de tipo resistivo con una red de escalera R-2R, éste tiene como entradas las salidas del FPGA que representan 2 bits por color (Rojo, Verde y Azul; RGB en inglés). Las salidas del DAC se ubican en las terminales del conector de video de la tarjeta. La figura 2.16 muestra las terminales correspondientes al FPGA y el DAC.

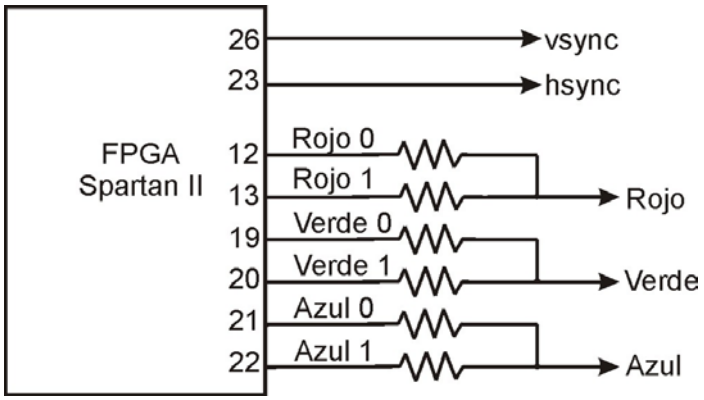

Figura 2.16. Conexión del conector VGA con el FPGA [10].

Las salidas del DAC le indican al monitor la intensidad de cada punto de fósforo en la pantalla, que al combinarse produce el píxel de un color. Con este DAC se pueden obtener hasta 64 colores diferentes en la pantalla.

Asimismo, los pulsos de sincronización horizontal y vertical son señales activas en nivel lógico bajo, y se conectan directamente al conector de video.

2.2.3 Entradas y salidas

Las entradas y salidas del controlador de video se pueden observar en la figura 2.17.

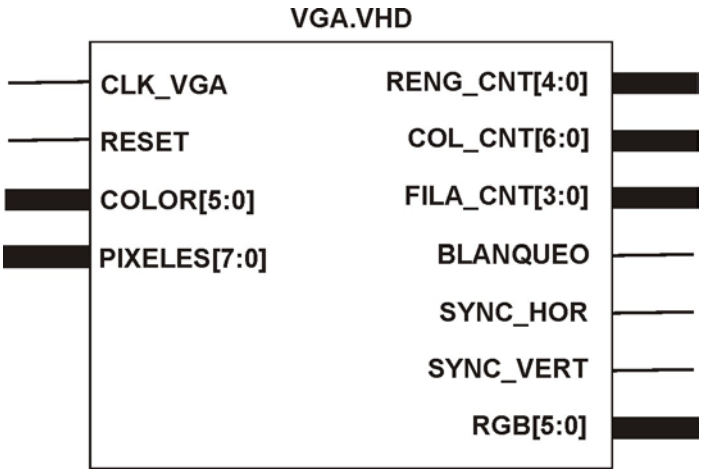

Figura 2.17. Símbolo del controlador de video.

Entradas:

- CLK_VGA. Frecuencia de reloj a la que opera el módulo.
- COLOR[5:0]. Bus que contiene información acerca del color de los píxeles a desplegar.
- PIXELES[7:0]. Bus que contiene una fila de la matriz leída de la ROM generadora de caracteres en el módulo de Memoria, es decir, los 8 píxeles a desplegar.

Salidas:

- RENG_CNT[4:0]. Bus de 5 bits que contiene el valor de contador de renglones y que el módulo de Memoria usa para leer un carácter de la RAM de refresco.
- COL_CNT[6:0]. Bus de 7 bits que contiene el valor de contador de columnas y que el módulo de Memoria usa para leer un carácter de la RAM de refresco.
- FILA_CNT[3:0]. Bus de 4 bits que representa el valor del contador de filas y que se emplea para formar el bus de direcciones de la ROM generadora de caracteres.
- BLANQUEO. Indicador de que se está realizando un blanqueo. Esta salida la toma el módulo de Memoria para saber cuando es necesario el acceso a memoria.
- SYNC_HOR. Proporciona la señal de sincronización horizontal al conector del monitor.

- SYNC_VERT. Proporciona la señal de sincronización vertical al conector del monitor.
- RGB[5:0]. Bus de 6 bits que el DAC emplea como entradas para obtener las intensidades de color del píxel mostrado en la pantalla.

Las salidas SYNC_HOR y SYNC_VERT de la figura 2.17 corresponden a las salidas hsync y vsync de la figura 2.16, respectivamente. Asimismo, los 2 bits más significativos del bus RGB corresponden a las líneas Rojo 1 y Rojo 0, los siguientes 2 bits representan a Verde 1 y Verde 0. Finalmente, los bits menos significativos corresponden a Azul 1 y Azul 0.

El archivo del módulo controlador de video se llama Vga.vhd y realiza todos los procedimientos necesarios para generar las señales de sincronización y mostrar los caracteres en la pantalla.

3. MÓDULOS MODBUS

La comunicación serial multipunto vía Modbus sobre línea serial requiere de un nodo maestro y varios esclavos, en el presente trabajo cada terminal esclavo está basada en una tarjeta XSA-100 programada en VHDL y, por otra parte, una PC es la encargada de realizar las funciones propias del nodo maestro. Los módulos que a continuación se describen pertenecen al esclavo con transmisión half-duplex, para las funciones del maestro se dedica un capítulo posterior.

Nota. En lo que resta del documento se empleará el término Modbus o Modbus sobre línea serial indistintamente, asumiendo que se está empleando éste último en el ejemplo desarrollado en el presente trabajo.

3.1 DESCRIPCIÓN GENERAL

La terminal esclavo consta de 4 módulos en VHDL:

- Módulo de recepción-transmisión serial
- Módulo de espera fin de trama.
- Módulo de almacenamiento de la trama
- Módulo de transacción Modbus.

La figura 3.1 es un diagrama a bloques la relación existente entre estos módulos, las flechas angostas muestran la conexión de una señal de un módulo a otro y, por su parte, las flechas más grandes representan la conexión de varias señales entre módulos.

El módulo de recepción-transmisión serial recibe y envía bytes por medio del bus de Modbus. Cuando recibe un byte habilita una señal que toma como entrada el módulo de espera fin de trama, la función de éste es “avisar” al módulo de almacenamiento de la trama que puede recibir los datos y guardarlos en bloques de memoria RAM del FPGA (o RAM interna). Finalmente, cuando se recibe y guarda toda la trama, el módulo de almacenamiento de la trama le indica al módulo de transacción Modbus que realice una acción, ya sea de lectura o escritura de datos en la SDRAM, esto con base en el código de

función de la trama almacenada en la RAM interna, cuando termina la transacción envía una respuesta al maestro por medio del módulo de recepción-transmisión serial.

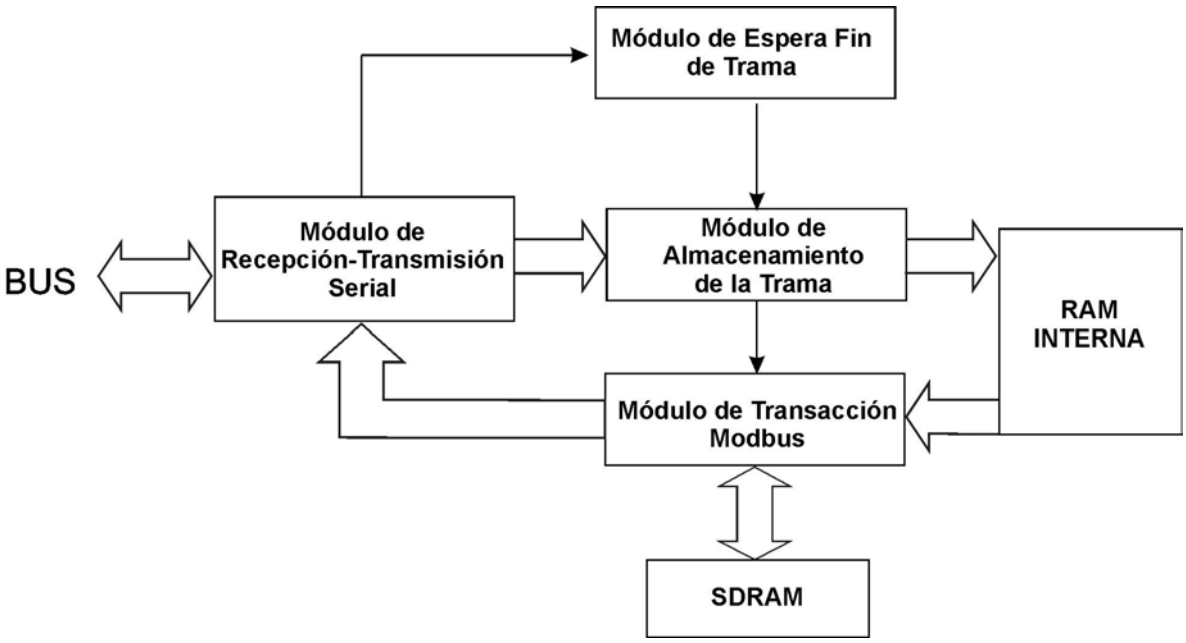

Figura 3.1 Diagrama a bloques de los módulos Modbus en el nodo esclavo.

3.2 MÓDULO DE RECEPCIÓN-TRANSMISIÓN SERIAL

3.2.1 Descripción general

Este módulo es el encargado de realizar operaciones de recepción y transmisión de datos de manera serial, por lo que requiere de una entrada para la recepción y una salida para la transmisión, así como un bus de salida que proporcione el dato recibido y un bus de entrada que con el cual se proporciona el dato a transmitir.

En Modbus sobre línea serial se requieren de LEDs que funcionan como indicadores visuales del momento en que se está recibiendo o transmitiendo, por ello también se necesitan 2 salidas: una de ellas se habilitará (con un nivel lógico alto) cuando se realice una operación recepción, y la otra se habilitará durante la transmisión. En ambos casos las salidas se deshabilitan cuando termina su operación ya sea de recepción o transmisión.

3.2.2 Descripción del funcionamiento del módulo de recepción-transmisión serial

Al recibir o enviar datos se requiere de un contador para determinar (habilitando una señal interna) en qué momento se debe guardar un nuevo bit recibido o transmitir el siguiente bit, respectivamente, es decir, el instante en que hay un cambio de bit. La cantidad de ciclos del contador se obtiene de la ecuación 3.1.

$$\text{Cantidad de Ciclos} = \frac{\dfrac{1}{\text{Velocidad de trasmisión}}}{\dfrac{1}{\text{Frecuencia de operación}}} = \frac{\text{Frecuencia de operación}}{\text{Velocidad de transmisión}} \qquad \text{(Ec.3.1)}$$

Por ejemplo, a una frecuencia de 50Mhz el periodo es de 20 ns y con una velocidad de transmisión de 2400bps, es decir, la duración en la que se transmite un bit es de 416.666 μs (1/2400); el valor máximo del contador debe ser:

$$\text{Cantidad de Ciclos} = \frac{1/2400\text{bps}}{1/50\text{Mhz}} = \frac{416.666\mu\text{s}}{20\text{ns}} = 20833.33$$

Dado que no pueden contarse fracciones de ciclos se toma la parte entera del resultado, por lo que cada 20833 ciclos a 50Mhz hay un cambio de bit.

La figura 3.2 muestra la descripción en VHDL del contador de ciclos. El conteo comienza cuando la señal interna CTA_EN se habilita. El contador (llamado CTA_CICLOS) se reinicia cuando: CTA_EN tiene un nivel lógico bajo, se aplica reset o cuando el conteo alcanza su máximo valor, representado por MAXIMO_CICLOS (obtenido de la ec. 3.1). En el momento de recepción de bit (cuando el conteo ha llegado al máximo y se reinicia el contador) se habilita la señal LPRB (Listo Para Recibir Bit) por un ciclo de reloj indicando que se debe guardar el bit de la entrada, de manera similar, en el momento de transmitir un nuevo bit se habilita la señal ENVIA_BIT indicando que se debe enviar el siguiente bit.

```
-------------------------------------------------------------------------------------------------------------------
--------    Contador que indica cuando recibir o enviar un bit de manera serial
-------------------------------------------------------------------------------------------------------------------
Rx_Tx_3:PROCESS(clk,reset,cta_en)
BEGIN
   IF reset='1' OR cta_en='0' THEN   -- Reset al contador cuando hay caracter disponible o reset
      Cta_Ciclos   <= 0;      -- Contador de ciclos es 0
      LPRB                 <= '0';     -- No Recibe Bit
      envia_bit            <= '0';     -- No envia Bit
   ELSIF (clk'event and clk='1') THEN
      LPRB                 <= '0';
      envia_bit            <= '0';
      IF Cta_Ciclos /= maximo_ciclos THEN      -- Si no ha llegado al maximo
         Cta_Ciclos        <= Cta_Ciclos+1;                    -- Suma 1 al contador
         IF edo_espera='1' and Cta_Ciclos = maximo_ciclos/2 THEN  -- Esta en el edo. espera y es la maximo/2
            Cta_Ciclos     <= maximo_ciclos; -- Asigna el maximo valor del contador
         END IF;
      ELSE
         Cta_Ciclos        <= 0;      -- Reinicia el contador
         LPRB              <= '1';     -- Listo para Recibir bit
         envia_bit         <= '1';     -- Envia el bit en la transmision
      END IF;
   END IF;
END PROCESS;
```

Figura 3.2. Descripción en VHDL del contador de ciclos del módulo de recepción-transmisión serial.

Hay una máquina de estados que se encarga de guardar los bits recibidos (en el proceso RX_TX_4A) y otra que se encarga de transmitir bits (en el proceso RX_TX_5A). Cuando se recibe el bit de inicio la máquina de recepción se encuentra en un estado llamado ESPERA, sólo en ese estado el valor del contador toma su valor máximo a la mitad del conteo actual (figura 3.3) para que: en el siguiente ciclo se reinicie el contador, la maquina de estados pase al siguiente estado y de esa manera los siguientes bits recibidos se guarden cuando haya pasado la mitad de tiempo del bit que se recibe, es decir, cuando el contador se haya reiniciado después de haber alcanzado su máximo (líneas punteadas figura 3.3).

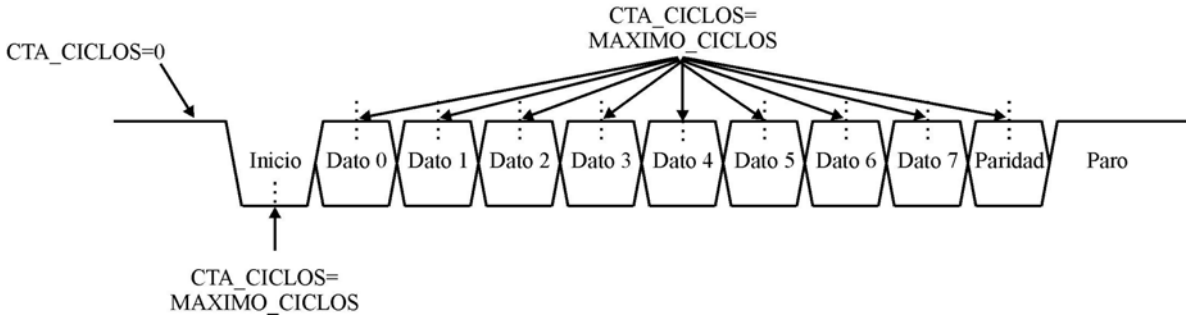

Figura 3.3. Reinicio del contador a la mitad del bit de inicio en la recepción de datos.

La figura 3.4 muestra diagrama de flujo del funcionamiento de este módulo, obsérvese que en la recepción o transmisión de datos se emplea un contador de ciclos, Clk'event representa un evento de la señal de reloj que puede ser un flanco de subida o de bajada (pero no ambos). MAX es la constante obtenida por la ecuación 3.1 que determina la velocidad de transmisión y recepción.

Inicialmente el módulo inicializa las salidas DATO_RECIBIDO, ERROR, ENVIADO, RECIBIENDO y TRANSMITIENDO a 0. Además cuenta con un detector de flancos (proceso Rx_Tx_1A) que revisa si la entrada Rx está en un nivel lógico bajo (bit de inicio) para iniciar la recepción de un dato.

Durante la recepción de bits se pone en un nivel lógico alto la salida RECIBIENDO. Al finalizar la recepción se habilita la salida DATO_RECIBIDO por un ciclo de reloj, indicando que el bus de salida DATO_Rx contiene el dato recibido. Si hay un error de paridad o de bit de paro se activa también la salida ERROR (esta detección de error funciona de igual manera que en el controlador de teclado descrito en el capitulo 2).

En la transmisión primero se revisa el estado de la entrada ENVIA_DATO, si está habilitada se toma el dato a enviar del bus de entrada DATO_Tx y se envía de manera serial por la salida Tx. Al término de la transmisión se activa la salida ENVIADO durante un ciclo de reloj para indicar el fin de la operación. También se habilita la salida TRANSMITIENDO durante la duración de la transmisión.

Cuando se está transmitiendo no es posible recibir datos hasta que la transmisión termine (dado que el contador se emplea tanto para la recepción como la transmisión). En caso de que se presente un dato en la línea de recepción al momento de transmisión, este módulo hará caso omiso de los bits que se reciban en la entrada. Esto se aplica de igual manera en la recepción de datos: no se puede transmitir cuando se están recibiendo datos. Es por ello que la aplicación que emplee este módulo debe sincronizar sus procesos de transmisión y recepción.

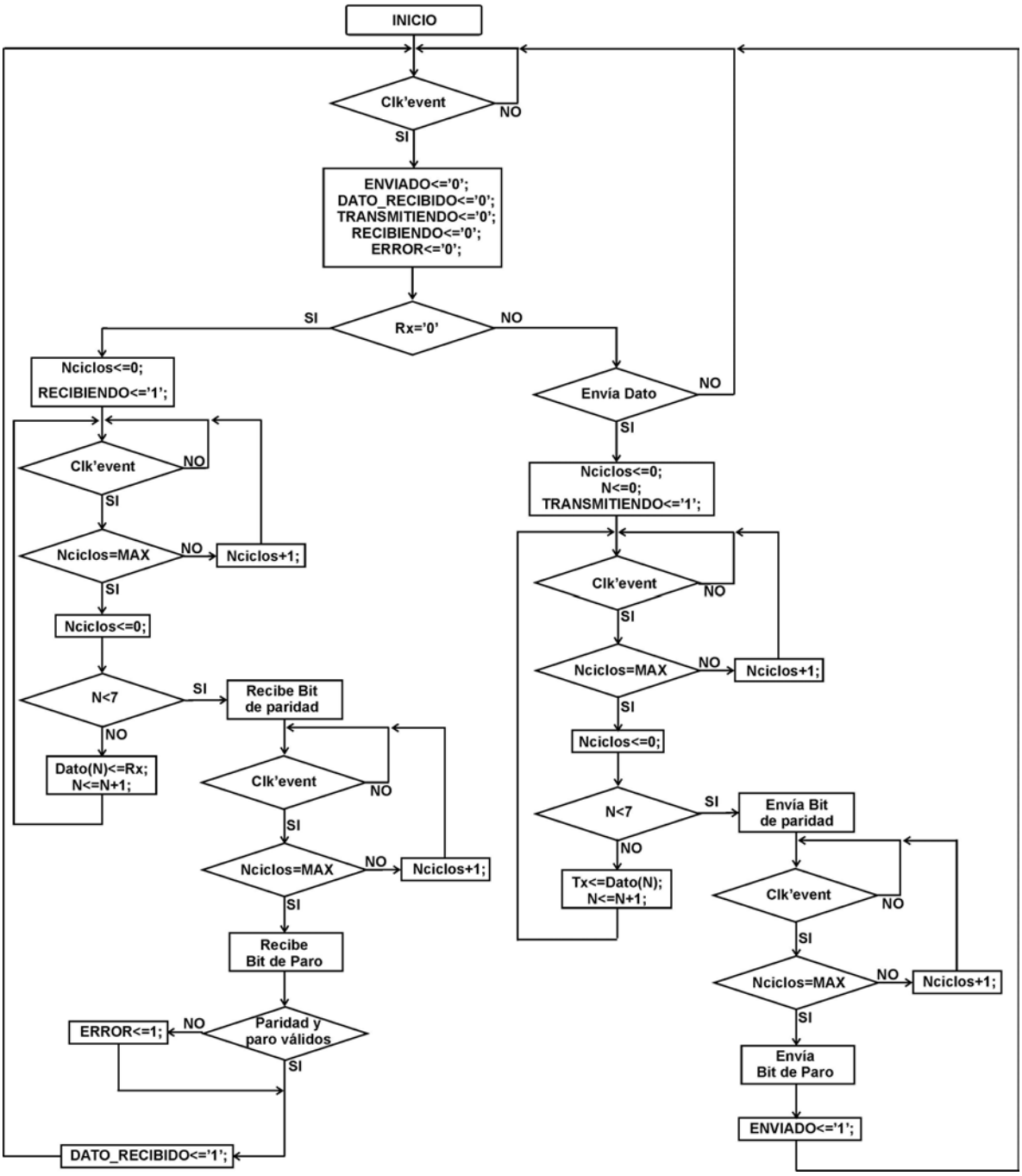

Figura 3.4. Diagrama de flujo del módulo de recepción-transmisión serial.

3.2.3 Entradas y salidas

Gracias a la flexibilidad de VHDL el módulo de recepción-transmisión serial cuenta con una constante con la que es posible definir la velocidad de transmisión sin necesidad de realizar más cambios. La figura 3.5 muestra el símbolo que representa a este módulo.

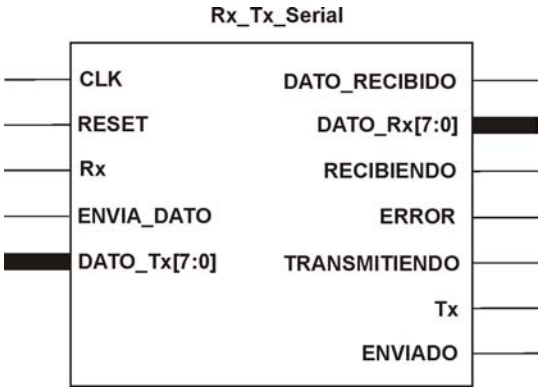

Figura 3.5. Símbolo del módulo de recepción-transmisión serial.

Entradas:

- CLK. Señal de reloj para el controlador.
- RESET. Señal de reset asíncrono.
- Rx. Línea de recepción de datos.
- ENVIA_DATO. Activa el envío del dato contenido en DATO_Tx.
- DATO_Tx[7:0]. Bus de datos los cuales que se transmiten de manera serial.

Salidas:

- DATO_RECIBIDO. Indicador de que se ha recibido un dato.
- DATO_Rx[7:0]. Bus que contiene el dato recibido de manera serial.
- RECIBIENDO. Indicador de que se está recibiendo un dato por la línea Rx.
- ERROR. Indicador de error de recepción del dato.
- TRANSMITIENDO. Indicador de que se está transmitiendo un dato por la línea Tx.
- Tx. Línea de transmisión de datos
- ENVIADO. Indicador de que se ha transmitido el dato.

3.3 MÓDULO DE ESPERA FIN DE TRAMA

3.3.1 Descripción general

Es un contador que habilita una salida llamada FIN_TRAMA cuando han transcurrido 3.5 veces el tiempo en que un dato se envía de manera serial (denominado $t_{3,5}$). Se utiliza para indicar el comienzo y final de la trama transmitida por el nodo maestro.

3.3.2 Descripción del funcionamiento del módulo de espera fin de trama

Para detectar el final de una trama enviada por el maestro tiene que transcurrir un lapso de $t_{3,5}$ (sección 1.6.3.7), por lo que para realizar la descripción de esto en VHDL se emplea un contador. Cada vez que se reciba un dato de manera serial debe comenzar el conteo, por ello se requiere de una entrada que será llamada DATO_RECIBIDO. Si el conteo llega hasta $t_{3,5}$ sin que se reciban más datos puede considerarse que el nodo maestro ha terminado el envío de la trama y la salida FIN_TRAMA se habilita. Si hay una recepción de dato cuando el contador se encuentra entre un lapso mayor de $t_{1,5}$ y menor a $t_{3,5}$, se considera que hay un error, por lo que se requiere de una salida llamada ERROR. Las salidas ERROR y FIN_TRAMA se activan sólo por un ciclo de reloj.

Al igual que en el módulo de recepción-transmisión serial, hay un contador de ciclos, sólo que en este caso tiene como máximo la cantidad de ciclos que conforman un lapso de $t_{3,5}$. Para obtener la cantidad de ciclos se considera la frecuencia y la velocidad transmisión (Ec. 3.2 y Ec. 3.3).

$$t_{1,5} = (1.5)(11)\left(\frac{\text{Frecuencia de operación}}{\text{Velocidad de transmisión}}\right) \qquad \text{(Ec.3.2)}$$

$$t_{3,5} = (3.5)(11)\left(\frac{\text{Frecuencia de operación}}{\text{Velocidad de transmisión}}\right) \qquad \text{(Ec.3.3)}$$

Así, para una velocidad de transmisión de 19200bps y una frecuencia de operación de 50Mhz, el valor máximo para cada contador es:

$$t_{1,5} = (1.5)(11)\left(\frac{50\text{Mhz}}{19200\text{bps}}\right) = 42968.75$$

$$t_{3,5} = (3.5)(11)\left(\frac{50\text{Mhz}}{19200\text{bps}}\right) = 100260.41$$

Al multiplicar la cantidad de ciclos por el periodo de la señal de frecuencia de operación se obtiene el lapso en segundos:

$$t_{1,5} = 42968 * 20\text{ ns} = 0.859\text{ ms}$$

$$t_{3,5} = 100260 * 20\text{ ns} = 2\text{ ms}$$

La figura 3.6 muestra la descripción del contador en VHDL, el valor de $t_{1,5}$ y $t_{3,5}$ está representado por num_ciclos_1_5 y num_ciclos_3_5, respectivamente.

```
PROCESS(clk,reset)
begin
   IF reset ='1' THEN
      FIN_TRAMA          <= '0';
      ERROR              <= '0';
      Realiza_conteo     <= '0';
      Cta_Ciclos         <=  0;
      Termina_conteo     <= '0';
   ELSIF clk'event and clk = '1' THEN
      FIN_TRAMA          <= '0';
      ERROR              <= '0';
      IF Dato_Recibido='1' THEN       -- Si llega un caracter nuevo...
         Cta_Ciclos      <= 0;                  -- Reincia el contador de ciclos
         Realiza_conteo  <= '1';    -- Realiza conteo de espera
         IF Cta_Ciclos >=num_ciclos_1_5 and Cta_Ciclos < num_ciclos_3_5 THEN -- Si llega un caracter en t>1.5
                                                                              -- y t<3.5
            ERROR        <= '1';
         END IF;
      ELSIF Realiza_conteo='1' THEN--Si se esta realizando el conteo
         IF Cta_Ciclos < num_ciclos_3_5 THEN-- si el lapso es menor a t3.5...
            Cta_Ciclos   <= Cta_Ciclos+1;   -- Incrementa el contador
         ELSE                    -- Si  t>3.5 entoces es el fin de la trama...
            Cta_Ciclos   <= 0;   -- Inicializa el contador
            FIN_TRAMA    <= '1';  -- Fin de trama detectado
            Realiza_conteo <= '0';  -- Ya no se debe realizar el conteo
         END IF;
      END IF;
   END IF;
END PROCESS;
```

Figura 3.6. Descripción en VHDL del contador de ciclos del módulo de espera fin de trama.

En la figura 3.7 se muestra el diagrama de flujo correspondiente a este módulo, nótese el uso del contador Nciclos según se reciben los datos en la condición Dato_Recibido.

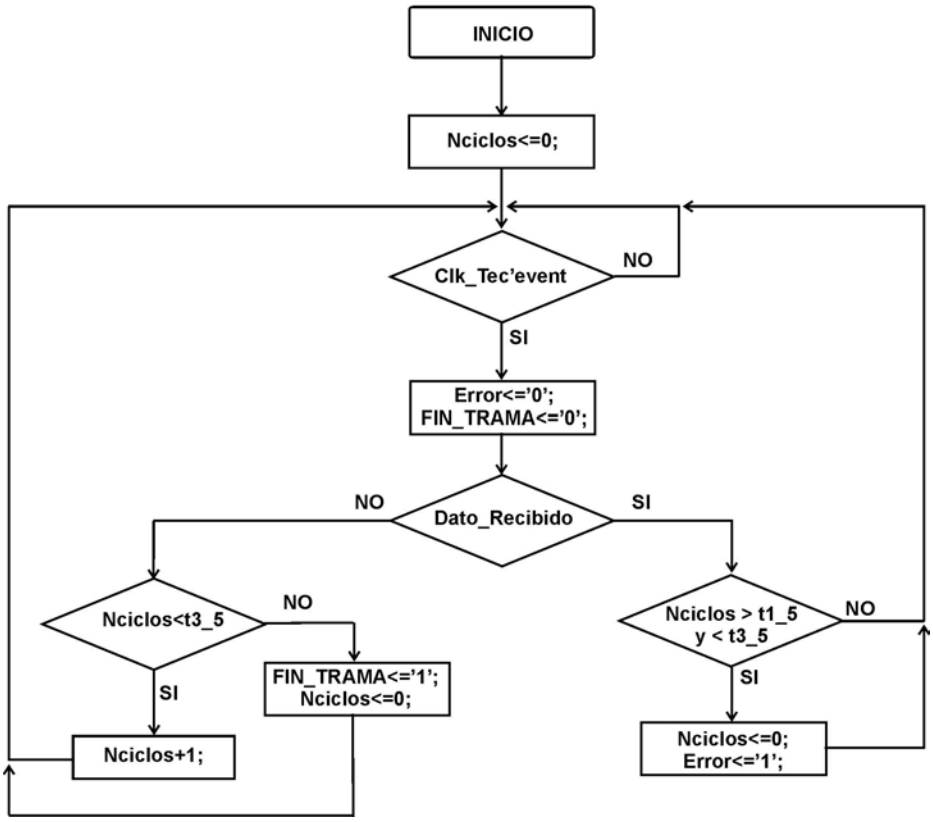

Figura 3.7. Diagrama de flujo del módulo de espera fin de trama.

3.3.3 Entradas y salidas

Las entradas y salidas del módulo de espera fin de trama se muestran en la figura 3.8.

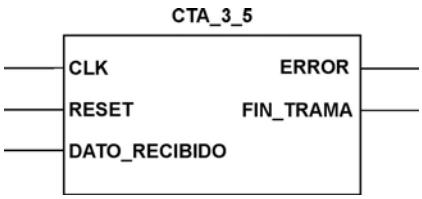

Figura 3.8. Símbolo del módulo de espera fin de trama.

Entradas:

- CLK. Señal de reloj para el controlador.
- RESET. Señal de reset asíncrono.
- DATO_RECIBIDO. Indica que se ha recibido un dato de manera serial.

Salidas:

- ERROR. Señal que indica que se ha recibido un dato después de $t_{1,5}$ veces el carácter y antes de $t_{3,5}$ veces
- FIN_TRAMA. Se habilita cuando se ha detectado un final de la trama recibida.

3.4 MÓDULO DE ALMACENAMIENTO DE LA TRAMA

3.4.1 Descripción general

Este módulo se encarga de habilitar las señales necesarias para guardar los datos que se reciben serialmente (la trama). Para este propósito emplea bloques de memoria RAM del FPGA (sección 1.3.1). Al bloque donde se almacena esta trama se le llama RAM_TRAMA, haciendo referencia a su uso. Además este módulo se encarga de hacer el cálculo de la CRC para indicar que la trama recibida es correcta.

3.4.2 Descripción del funcionamiento del módulo de almacenamiento de la trama

Se requiere de una entrada que indique que se ha recibido un dato de manera serial para almacenarlo, además de una entrada que indique que se ha recibido la trama completa y un bus de entrada que contenga el dato a escribir en la memoria.

Para almacenar datos en RAM_TRAMA este módulo debe esperar el primer byte de la trama, es decir, la dirección del esclavo destino, si ésta corresponde al nodo esclavo comienza la escritura de los datos que se reciben. El final de la escritura se detecta cuando en la entrada FIN_TRAMA (proveniente del módulo de espera fin de trama) se habilita indicando que se ha detectado el fin de la trama.

El byte que se escribe en RAM trama se toma del bus de una entrada llamada DATO_ENTRADA cuando la entrada DATO_RECIBIDO indica que hay un nuevo dato a escribir. El cálculo de la CRC de los datos recibidos se realiza de manera concurrente en que se están escribiendo. Al término de la recepción de la trama, salida CRC_VALIDO se habilita por un ciclo de reloj para indicar que la trama recibida está libre de errores.

La figura 3.9 muestra la descripción que habilita la escritura en RAM_TRAMA, así como el cálculo de la CRC. Esta descripción pertenece a una máquina de estados. El estado ESPERA_DIRECCION recibe el primer byte de la trama y determina si ésta debe escribirse en memoria o ignorarse (estado ESPERA) hasta que se termine el envío de la trama por parte del nodo maestro. El estado ESPERA_DATO revisa la señal FIN_TRAMA; si es cero, indica que no se ha terminado el envío de la trama y espera a recibir un dato, cuando éste llega habilita las señales para la escritura en RAM_TRAMA y el cálculo de la CRC.

En la figura 3.10 se muestra el diagrama de flujo correspondiente a este módulo. La dirección de inicio de escritura en la memoria RAM_TRAMA “00000000”, si el primer dato no corresponde a la dirección del esclavo o no es una transmisión tipo difusión (dirección 0x00) no se guarda la trama, de otra manera continua para almacenar los datos hasta que FIN_TRAMA sea ‘1’.

```
WHEN INICIAL =>
    Inicializa_CRC <= '1';              -- Inicializa la CRC
    dir_sig        <= (others=>'0'); -- Direccion RAM donde se comienza escribir la TRAMA
    EDO_SIG        <= ESPERA_DIRECCION;


WHEN ESPERA_DIRECCION => -- Revisa si la direccion recibida correponde al esclavo
    IF DATO_RECIBIDO='1' THEN                                          -- Si llego un Dato (1er byte)
       IF (DATO_ENTRADA=TO_UNSIGNED(DIR_ESCLAVO,DATO_ENTRADA'length) or --  Si es unidestino
          DATO_ENTRADA = TO_UNSIGNED(0,DATO_ENTRADA'length)                --  o difusion
       ) THEN
          calcula_CRC_sig <= '1';                  -- Calcula la CRC del dato
          REG_CRC_sig   <= DATO_ENTRADA; -- Dato a calcular en la CRC
          EDO_SIG         <= ALMACENA_DATO;
       ELSE
          EDO_SIG         <= ESPERA;
       END IF;
    END IF;

       WHEN ESPERA_DATO =>
          IF FIN_TRAMA='0' THEN
             IF DATO_RECIBIDO='1' THEN  -- Si llego un Dato serial
                calcula_CRC_sig   <= '1';
                REG_CRC_sig       <= DATO_ENTRADA;
                Escribiendo_trama <= '1';     -- Indica que esta escribiendo en RAM_TRAMA
                we                <= '1';     -- Habilita la escritura
                EDO_SIG           <= ALMACENA_DATO;
             END IF;
          ELSE
             Rx_Terminada  <= '1';
             EDO_SIG       <= INICIAL;
          END IF;

       WHEN ALMACENA_DATO =>
          we                <= '1';
          Escribiendo_trama <= '1';
          Escribe_dato      <= '1';
          dir_sig           <= dir_act+1;    -- Aumenta 1 localidad de RAM_TRAMA
          EDO_SIG           <= ESPERA_DATO;

       WHEN ESPERA =>            -- Espera el final de la Trama para ir al estado Inicial
          IF FIN_TRAMA='1' THEN
             EDO_SIG       <= INICIAL;
          END IF;
```

Figura 3.9. Descripción en VHDL que habilita la escritura en memoria del módulo de almacenamiento de trama.

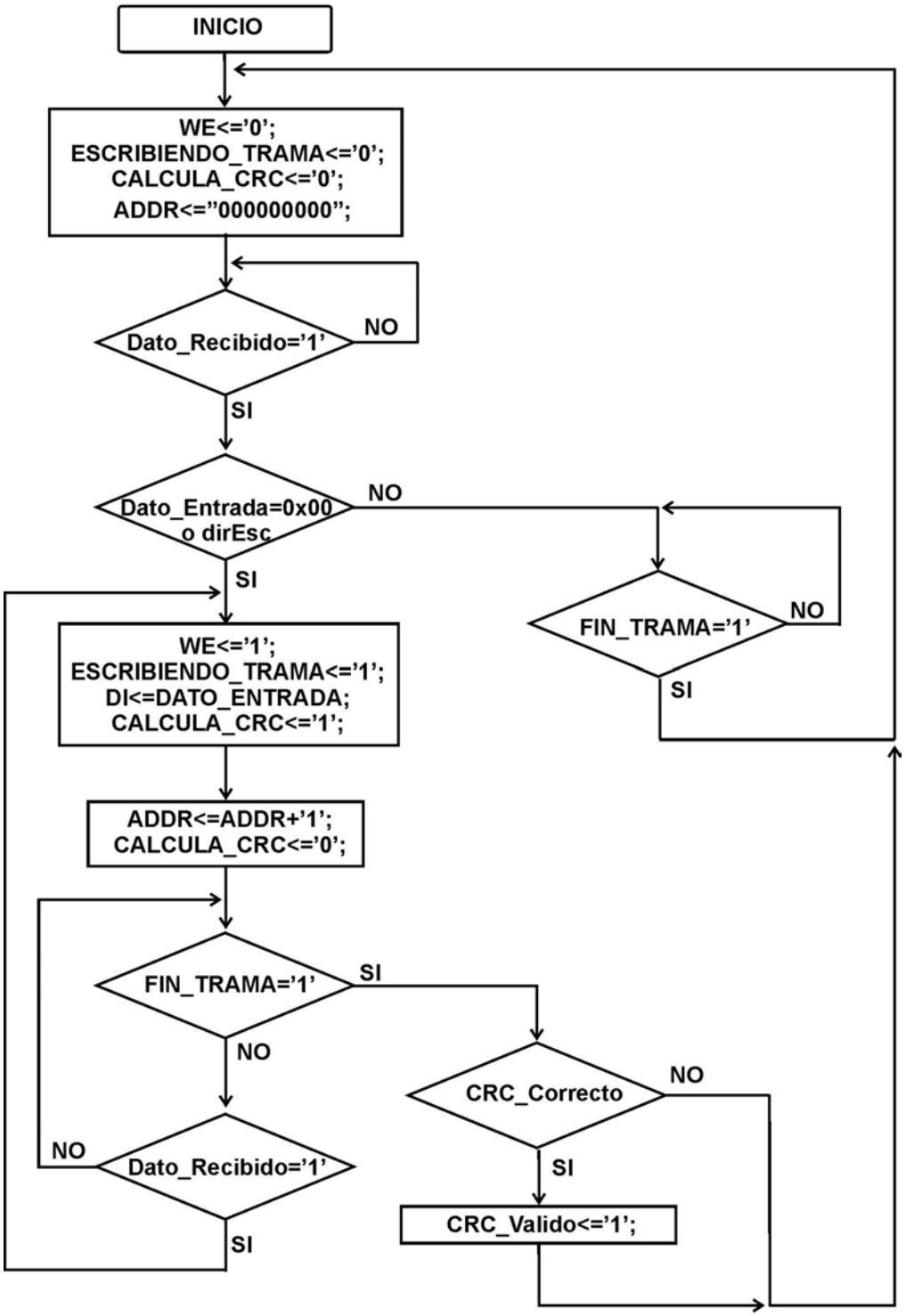

Figura 3.10. Diagrama de flujo del módulo de almacenamiento de la trama.

3.4.3 Entradas y salidas

Las entradas y salidas del módulo de almacenamiento de la trama se muestran en la figura 3.11.

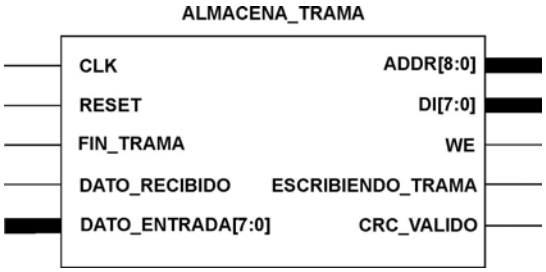

Figura 3.11. Símbolo del módulo de almacenamiento de la trama.

Entradas:

- CLK. Señal de reloj para el controlador.
- RESET. Señal de reset asíncrono.
- FIN_TRAMA. Habilita el final del almacenamiento.
- DATO_RECIBIDO. Indica que se ha recibido un dato de manera serial y se debe escribir en la memoria.
- DATO_ENTRADA[7:0]. Dato a escribir en la memoria.

Salidas:

- ADDR[8:0]. Bus de direcciones de RAM_TRAMA.
- DI[7:0]. Bus de datos de la RAM interna.
- WE. Habilita la escritura del dato en el bloque de memoria RAM.
- ESCRIBIENDO_TRAMA. Indica que se está realizando la escritura del dato.
- CRC_VALIDO. Se habilita al final de la escritura de la trama e indica que ésta no tiene errores.

3.5 MÓDULO DE TRANSACCIÓN MODBUS

3.5.1 Descripción general

Este módulo se encarga de realizar acciones determinadas por el código de función de la trama enviada por el nodo maestro. Por ello deben leer los datos escritos por el módulo de almacenamiento de la trama y así determinar la acción a realizar. Los códigos de función incorporados son 2 públicos:

- 0x01. Lectura de bits.
- 0x10. Escritura de registros.

Este módulo no realiza los accesos a memoria SDRAM, sólo habilita las señales necesarias para que otro módulo se encargue de ello.

Además se incorpora un código de función creado con el propósito de visualizar la información en la pantalla:

- 0x41. Define página. Una página es una porción de memoria que forma parte de la RAM de refresco que emplea la interfaz del monitor. En total para el consultor de calificaciones serán 4 páginas descritas el capítulo siguiente.

Después de realizar la acción, se debe enviar una respuesta de manera serial por lo que se proporcionan las salidas para que el módulo de recepción-transmisión serial envíe la respuesta al nodo maestro.

3.5.2 Descripción del funcionamiento del módulo de transacción Modbus

Se requiere de una entrada que le indique a este módulo que se tiene una trama en memoria, esta entrada se llama INICIA_TRANSACCION, y la habilita el módulo de almacenamiento de la trama cuando ésta corresponde al nodo esclavo correcto (al nodo destino) y sin error de CRC. Cuando se tiene una nueva trama, es necesario proporcionar el bus de direcciones para la lectura de la trama, así el bus de datos que contenga el dato leído en RAM_TRAMA.

El primer dato que se debe leer de RAM_TRAMA corresponde a la dirección del esclavo, si ésta es 0x00 entonces es una transmisión tipo difusión en la que el maestro ordena que se reinicien los esclavos. De otra forma se lee el siguiente dato de RAM trama que contiene el código de función, el cual puede ser:

- ***Lectura de bits.*** Hace una petición al módulo de Memoria (descrito en el próximo capítulo) de lectura de bits en la SDRAM. Por lo que se requiere de una entrada (llamada LECTURA_MB_TERMINADA) que indique que la lectura de bits se ha completado.
- ***Escritura de registros***. En este caso se realiza una petición de escribir datos en la SDRAM. Al término de esta operación el módulo de Memoria habilita la entrada ESCRITURA_MB_TERMINADA.
- ***Definición de Página***: Este código de función determina el área de memoria a la cual la RAM de refresco tiene acceso para mostrar información en la pantalla.

Para cada código de función (a excepción del 0x41) se debe revisar que la dirección de lectura o escritura, según sea el caso, sea válida y que la cantidad de bytes a leer o escribir sea mayor que cero y menor a la cantidad soportada (sección 1.6.2.3). Además, la suma de la dirección más la cantidad de bytes debe ser válida debido a que puede haber sobreflujo.

La figura 3.12 muestra la descripción que revisa que la suma de direcciones más la cantidad de bytes a leer o escribir (DIR_FINAL) sea válida, además que la cantidad máxima de bytes (NUM_REGS) sea la permitida.

```
WHEN LIMITE_SUMA =>-- Valida la suma del estado SUMA
   IF NUM_REGS >= TO_UNSIGNED(1, NUM_REGS'length) AND
      (
      (NUM_REGS <= TO_UNSIGNED(250, NUM_REGS'length) and CF = CF_0x01 ) or -- Rango permitido en
                                                                              -- el CF 0x01
      (NUM_REGS <= TO_UNSIGNED(123, NUM_REGS'length) and CF = CF_0x10 ) --Rango en el CF 0x10
      ) THEN
      IF DIR_FINAL(15 DOWNTO 0) > DIR_INI(15 DOWNTO 0) THEN  --Si no hay sobreflujo en la suma...
         DIR_FINAL_SIG <= DIR_FINAL-1;       -- Resta 1 a la direccion final
         IF CF= CF_0x01 THEN
            dir_sig_RAM  <= TO_UNSIGNED(0, dir_ACT_RAM'length); -- Direccion inicial de RAM_TRAMA
            EDO_SIG      <= ENVIA_DIR_CF;   -- Comienza la transaccion CF 0x01
         ELSIF CF= CF_0x10 THEN
            dir_sig_RAM <= dir_act_RAM+1;-- Incrementa la direccion de RAM_TRAMA para leer el BC
            EDO_SIG      <= ASIGNA_BC_CF_0x10;
         END IF;
      ELSE                  --Si hay sobreflujo en la suma...
         C_EXCEPCION_sig  <= CodEx2;    -- Codigo de excepcion 2: Limites de memoria erroneos
         EDO_SIG               <= COD_EXCEPCION;
      END IF;
   ELSE                     --La cantidad de resgistros es erroneo
      C_EXCEPCION_sig  <= CodEx3;        -- Codigo de excepcion 3: Cantidades de registros erroneos
      EDO_SIG               <=  COD_EXCEPCION;
   END IF;
```

Figura 3.12. Descripción en VHDL que revisa la suma de direcciones más la cantidad de bytes a acceder en la memoria SDRAM.

La descripción de la figura 3.12 pertenece a una máquina de estados, si la suma o la cantidad de registros no fue válida se procede a construir una respuesta de excepción que será enviada al nodo maestro. De otra manera se debe realizar la acción solicitada.

Para enviar una respuesta del nodo esclavo se debe realizar una petición de envío al módulo de recepción-transmisión serial al habilitar la señal ENVIA_DATO_SERIAL y proporcionar el dato a enviar con el bus de salida DATO_A_ENVIAR. Al término del envío, el módulo de transmisión serial habilita la señal ENVIADO que este módulo toma como entrada y así ordenar el siguiente envío de byte o permanecer en modo de espera de la siguiente trama si es que ha terminado el envío.

La figura 3.13 muestra de manera general el funcionamiento de este módulo.

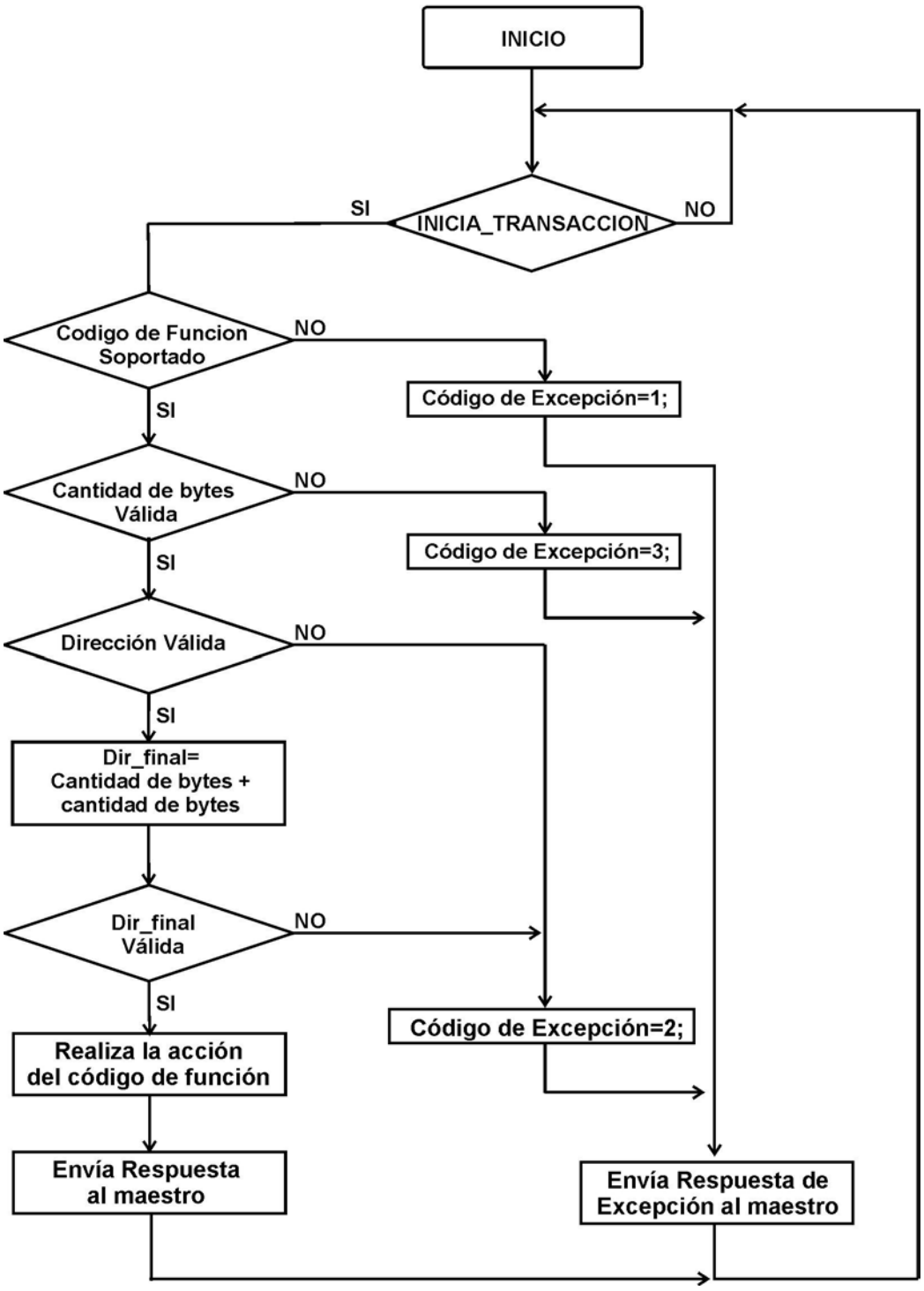

Figura 3.13. Diagrama de flujo del módulo de transacción Modbus.

3.5.3 Entradas y salidas

Las entradas y salidas del módulo de transacción Modbus se muestran en la figura 3.14.

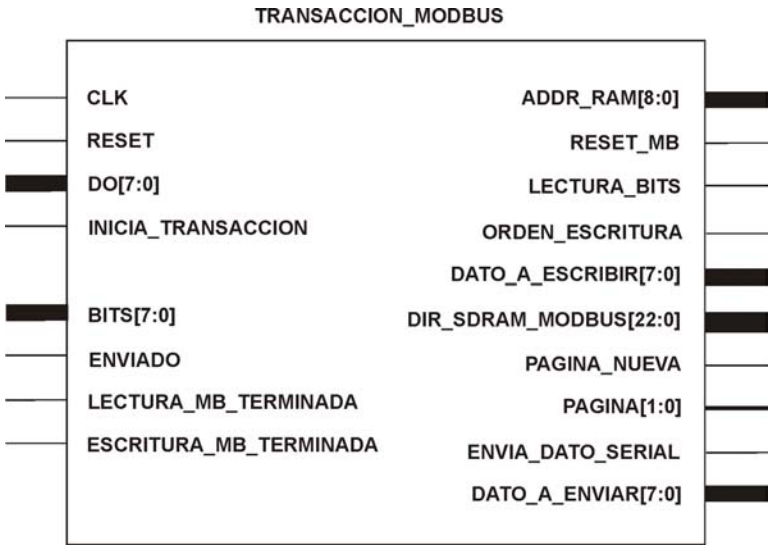

Figura 3.14. Símbolo del módulo de transacción Modbus.

Entradas:

- CLK. Señal de reloj para el controlador.
- RESET. Señal de reset asíncrono.
- DO[7:0]. Bus de datos de RAM_TRAMA.
- INICIA_TRANSACCION. Habilita el inicio de la transacción.
- ESCRITURA_MB_TERMINADA. Indica que se ha terminado la operación de escritura en la memoria SDRAM.
- LECTURA_MB_TERMINADA. Indica que se ha terminado la operación de lectura de la memoria SDRAM.
- BITS[7:0]. Bits leídos de la transacción del código de función 0x01.
- ENVIADO. Indica que se ha enviado un dato de manera serial.

Salidas:

- ADDR_RAM[8:0]. Bus de direcciones de RAM_TRAMA.
- RESET_MB. Señal de reset aplicada desde el nodo maestro cuando transmite en modo difusión.

- LECTURA_BITS. Orden de lectura de bits (código de función 0x01) al módulo de memoria en la SDRAM.
- ORDEN_ESCRITURA. Orden de escritura de registros (código de función 0x10) al módulo de memoria en la SDRAM.
- DATO_A_ESCRIBIR[7:0]. Dato que forma parte del registro a escribir en la SDRAM.
- DIR_SRAM_MODBUS[22:0]. Bus de direcciones de la SDRAM empleada en las operaciones de lectura o escritura.
- PAGINA_NUEVA. Indica que se tiene una página nueva a desplegar en el monitor (código de función 0x41).
- PAGINA[1:0]. Arreglo que indica cuál de las cuatro páginas se va a mostrar en la pantalla.
- ENVIA_DATO_SERIAL. Orden de enviar un dato de manera serial.
- DATO_A_ENVIAR [7:0]. Dato a transmitir de manera serial.

3.6 CONEXIÓN DE LOS MÓDULOS MODBUS

Aunque en el capítulo Instanciación de Componentes se interconectan todos los módulos, la figura 3.15 muestra la conexión de los símbolos que representan los módulos Modbus con todas sus entradas y salidas. Cada símbolo tiene el nombre asignado en las figuras anteriores y con la extensión .VHD, indicando que son módulos VHDL. La única salida que no está conectada es ERROR del módulo recepción-transmisión serial y está considerada como “abierta”, esto debido a que con el cálculo de la CRC del módulo de almacenamiento de la trama se realiza la revisión de errores.

La salida RECIBIENDO del módulo de recepción y transmisión serial no se conecta a algún módulo, esto a causa de que se le asigna a una terminal del FPGA, misma que estará conectada a un LED como indicador visual de la recepción de datos.

Debido a que la SDRAM también se utiliza como memoria de refresco, el módulo de Memoria se encarga de sincronizar los accesos a la SDRAM, es decir, de realizar una operación de escritura o lectura a la vez. En este bloque sólo se muestran las entradas y salidas relacionadas con los módulos de Modbus.

Los módulos de almacenamiento de la trama y de transacción Modbus conectan sus salidas de bus de direcciones a un multiplexor, de esta forma sólo uno de ellos tiene acceso a RAM_TRAMA. Dicho multiplexor acepta las entradas del bus de direcciones del módulo de almacenamiento de la trama cuando su salida WE está activa en un nivel lógico alto, de otro modo, las direcciones de entrada que el multiplexor acepta son las del bus de direcciones del módulo de transacción.

El bloque denominado RS-485 corresponde a la interfaz física para comunicarse con el bus, tiene 2 señales de entrada (Tx y TRANSMITIENDO) y una de salida Rx, todas se conectan con el módulo de recepción-transmisión serial.

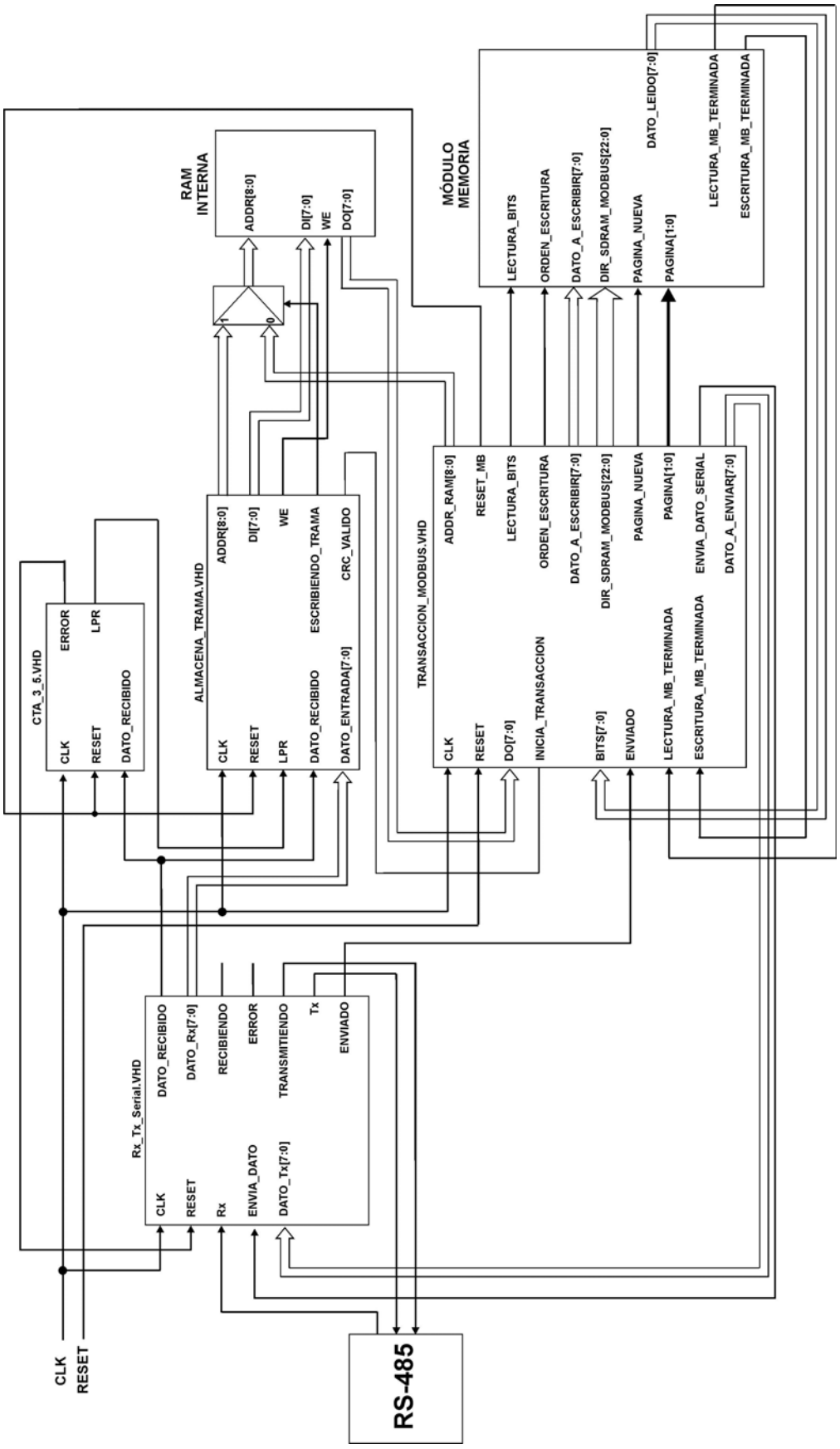

Figura 3.15. Conexión de los módulos Modbus.

3.7 INTERFAZ ESCLAVO/RS-485

Cada esclavo se conecta con el circuito integrado MAX483E tal y como se puede observar en la figura 3.16.

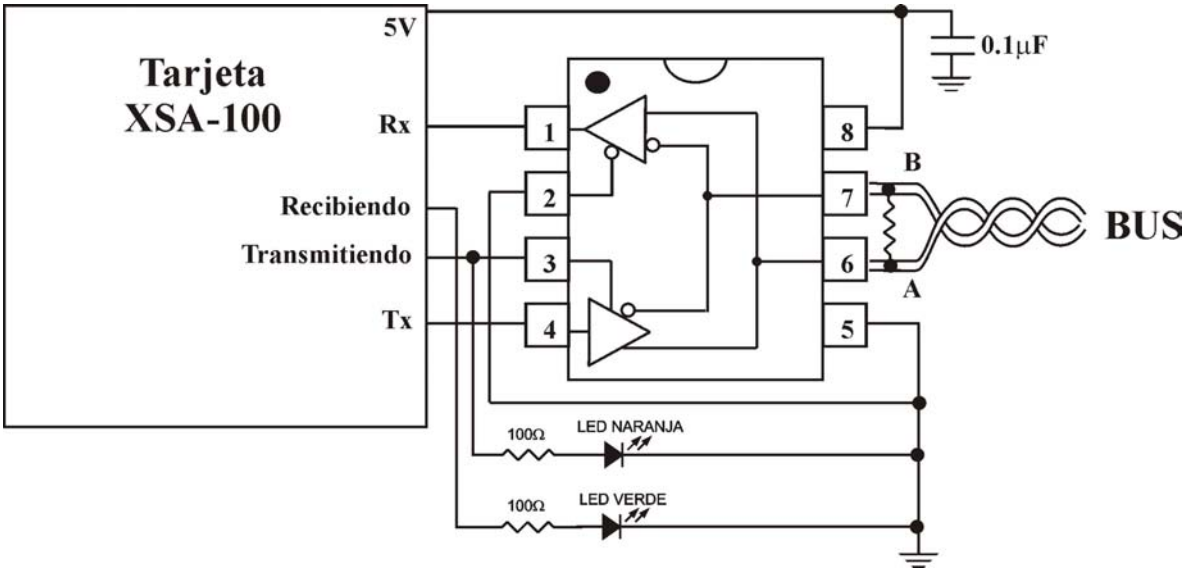

Figura 3.16. Conexión de la tarjeta XSA-100 con el circuito integrado MAX483E

El resistor que se conecta en las terminales del bus es de 100 ohms y se conecta en el esclavo del extremo del bus, tal y como lo especifica Modbus sobre línea serial (sección 1.6.4.7).

La tarjeta proporciona una salida de 5Vcd y tierra que se utilizan para alimentar al MAX483E. La salida **Transmitiendo** (del módulo de recepción-transmisión serial) tiene un nivel lógico alto cuando se transmiten datos y un nivel lógico bajo cuando no transmite, con esto las terminales B y A se encuentran en alta impedancia y la terminal RO puede recibir datos. Asimismo, las salidas Rx y Tx se conectan (descritas en la sección 3.2) a las terminales RO y DI, respectivamente. Los LEDs son indicadores visuales de la recepción y transmisión de datos. Finalmente, la terminal 2 se encuentra conectada a tierra, de esta forma el MAX483E siempre está habilitado para recibir datos, a menos que DE esté habilitada.

4. INSTANCIACIÓN DE COMPONENTES

Un sistema en VHDL puede estar formado por múltiples componentes para realizar funciones complejas. De esta forma se puede ver cada componente como una caja negra con entradas y de salidas que reaccionan a ellas.

La instanciación de componentes relaciona los módulos, es decir, los conecta mediante señales para conseguir que un sistema más complejo funcione. A un diseño basado en módulos se le llama diseño jerárquico. En el capítulo anterior, la figura 3.15 mostró la instanciación de los módulos de Modbus, el módulo encargado de instanciar todos los módulos es el módulo de más alto nivel.

Antes de comentar la instanciación necesaria para el diseño de la aplicación, se explica el módulo de Memoria desarrollado ya que forma parte fundamental de la mayoría de los módulos empleados en el presente trabajo.

4.1 MÓDULO DE MEMORIA

4.1.1 Descripción general

Este módulo se encarga de la sincronización de los accesos a la SDRAM por parte de los módulos de Interfaz y Modbus, esto mediante la habilitación de señales, además de:

- Inicializar la Memoria de Refresco de Pantalla.
- Inicializar la Memoria de Teclado.
- Obtener los píxeles correspondientes de la ROM generadora de caracteres.

Para ello emplea 3 tipos de memoria:

- RAM tipo Flash.
- SDRAM.
- Bloques de Memoria del FPGA.

4.1.1.1 RAM tipo Flash

Una parte de ella se utiliza para programar al FPGA en el momento de suministrar voltaje a la tarjeta, es decir, sin necesidad de utilizar una PC; la otra parte contiene la información de Memoria de Refresco y la Memoria de Teclado que será almacenada en la SDRAM. La figura 4.1 muestra el rango de direcciones para cada bloque de memoria, el espacio con líneas diagonales corresponde a memoria no asignada.

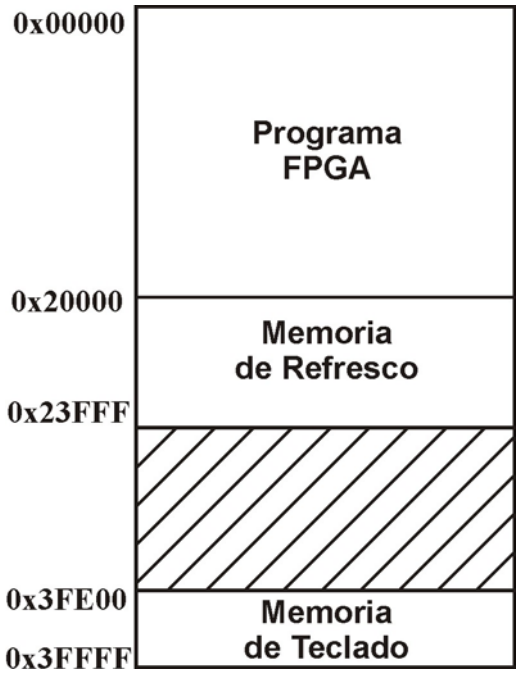

Figura 4.1. Contenido de la RAM tipo Flash.

Para almacenar tanto el diseño que programa al FPGA como la Memoria de Teclado y de Refresco de Pantalla (Memoria de Video), se emplea el software XSTOOLS que ofrece el vendedor de la tarjeta XSA-100. Esta operación sólo se realiza una vez ya que la RAM tipo Flash no es volátil. El archivo que programa al FPGA se llama ESCLAVO.EXO[7], asimismo el archivo que contiene a la Memoria de Teclado y de Video se llama MVIDETEC.MCS, ambos se encuentran en el disco anexo a este documento en la carpeta VHDLs. Para programar configurar la RAM tipo Flash con estos archivos puede consultar la referencia [URL6].

[7] Este archivo corresponde al primer esclavo, para obtener otro esclavo puede consultar la sección 4.2.

4.1.1.2 SDRAM

4.1.1.2.1 Módulo controlador de la SDRAM

Este módulo fue desarrollado por XESS Corporation (la compañía en la cual se adquirió la tarjeta XSA-100) y se llama Sdramcntl.vhd, la descripción de este archivo en VHDL está en el disco anexo a este documento en la carpeta VHDLs. Gracias a él se puede utilizar la SDRAM como si fuera una simple RAM estática ya que se encarga de realizar los refrescos de memoria, la asignación de bancos, comandos, etc. y proporciona un rango de direcciones de 0x000000 a 0x7FFFFF con palabras de 16 bits. Para más detalles sobre este controlador puede consultar la referencia [URL18], y la referencia [URL19] contiene información útil para comprender el uso de una SDRAM.

4.1.1.2.2 Asignación de memoria en la SDRAM

La SDRAM se utiliza para almacenar la Memoria de Teclado y la Memoria de Refresco que contiene la RAM tipo Flash las razones son:

- Las operaciones Modbus afectarán el contenido de la Memoria de Video (el maestro enviará información vía Modbus que será visible en la pantalla), sin embargo, Modbus especifica que la memoria empleada para sus operaciones de lectura y escritura debe comenzar en la dirección 0x0000, por ello, no es posible utilizar la RAM tipo Flash porque esas localidades de memoria almacenan los datos que programan el FPGA.
- Aunque se utilice la memoria desde la dirección 0x20000 (utilizando los 4 bytes menos significativos como dirección inicial Modbus), el proceso para reprogramar un byte en la Flash requiere de borrar todo un bloque y programar byte a byte [URL20]. Debido a que la Memoria de Refresco debe actualizarse constantemente, es más eficiente escribir en la SDRAM por velocidad y facilidad de acceso.
- En el caso de la Memoria de Teclado se emplea también la SDRAM debido a que es más práctico el acceso a una sola memoria que usar 2 memorias, con lo que se habilitan menos señales de lectura, se emplean menos buses de datos y de direcciones.

La asignación de la SDRAM se muestra en la figura 4.2.

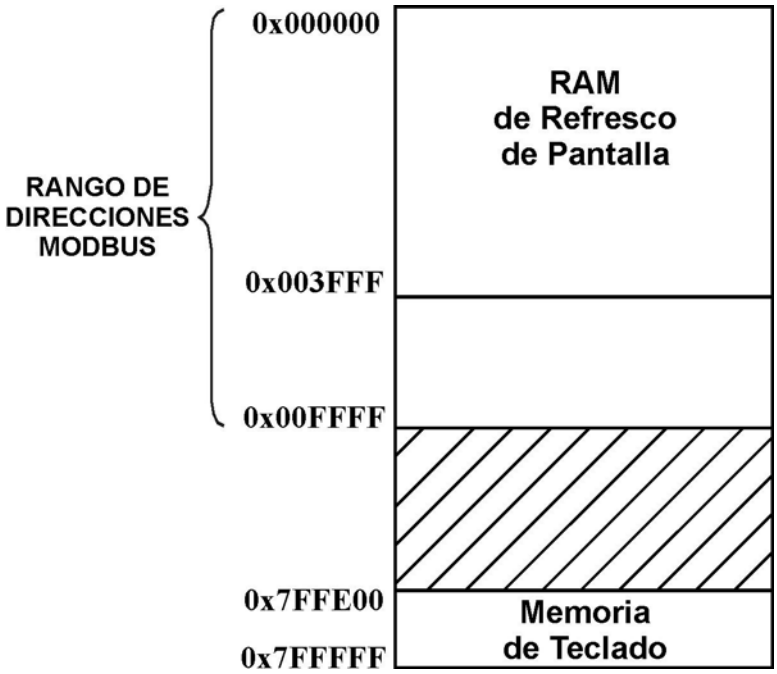

Figura 4.2. Contenido de la SDRAM.

La RAM de refresco está formada por 16Kx16 palabras que representan 4 páginas a mostrar en la pantalla, es decir, a cada página le corresponden 4Kx16 palabras. En la figura 4.3(a) se puede observar parte del contenido (en hexadecimal) de la primera página. El primer renglón comienza en la dirección 0x000000, y contiene los códigos ASCII correspondientes al margen superior mostrado en la figura 4.3(b). El segundo renglón comienza en la dirección 0x000080, únicamente para fines demostrativos considérese que hay sólo un espacio en blanco (0x20) entre el margen (carácter 0xBA), y la palabra CONSULTA (caracteres 0x43, 0x4F, 0x4E, 0x53, 0x55, 0x4C, 0x54, 0x41) de la figura 4.3(b). El tercer renglón contiene los caracteres de la línea de división.

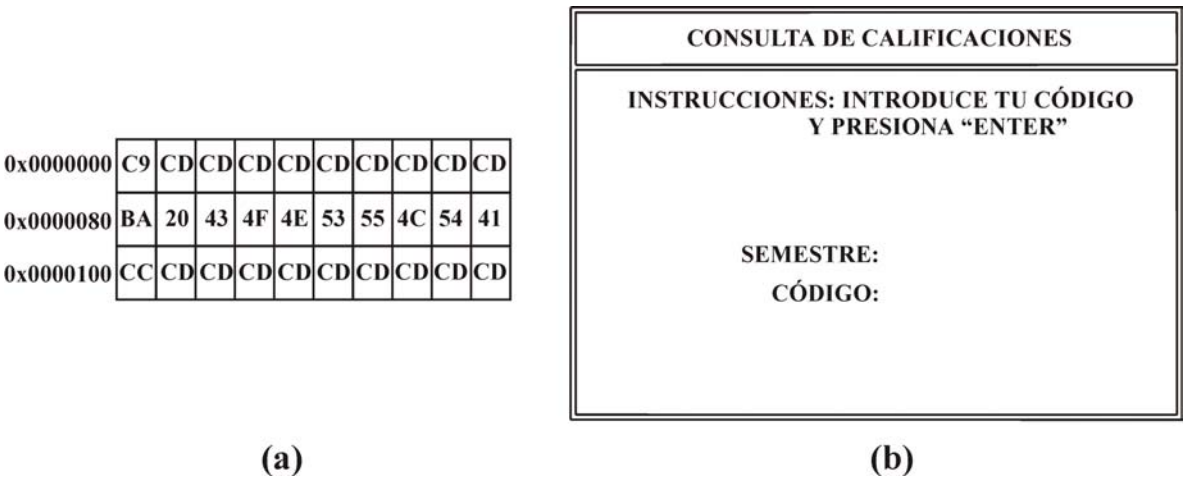

Figura 4.3. (a) Contenido de la SDRAM correspondiente a la esquina izquierda superior de primera página. (b) Imagen de la primera página mostrada en la pantalla del monitor.

Las páginas restantes muestran:

- Mensaje de que no existe el código.
- Calificaciones del alumno.
- Mensaje de que no existe el semestre.

Estas páginas se ilustran en la sección de resultados del capítulo 6.

Por otro lado, la Memoria de Teclado consta de 512 localidades de memoria que almacenan los códigos ASCII correspondientes a cada código de rastreo de las teclas imprimibles. La mitad de la memoria corresponde a los caracteres de las letras minúsculas y aquellos que en un procesador de texto se muestran cuando no está presionada la tecla Shift. El resto corresponde a los caracteres de letras mayúsculas y los demás caracteres. La dirección SDRAM está determinada por el código de rastreo, y si el estado del LED correspondiente a BLOQ MAYÚS es encendido. Así el bus de direcciones será:

```
DIR_SDRAM <= "11111111111111" &  MAYUSC &
                    Codigo_Rastreo(7 DOWNTO 0);          (Ec.4.1)
```

donde:

MAYUSC es un bit que representa el estado del LED mencionado como encendido (’0’) o apagado (‘1’).

Codigo_Rastreo es un arreglo unidimensional de 8 bits que representa el código de rastreo correspondiente a la tecla presionada.

Por ejemplo, si el código de rastreo es 0x33 (letra H) y el LED BLOQ MAYÚS está apagado, la dirección SDRAM es:

DIR_SDRAM <= "11111111111111" & ‘0’ & “00110011”;

La parte menos significativa de esa localidad debe contener 0x68, el cual es el código ASCII de la letra H.

Si la tecla presionada no corresponde a un carácter imprimible, la localidad de la SDRAM contiene sólo bits ‘0’.

Una característica a considerar es que la Memoria de Teclado está orientada a teclados en español, además de que los códigos de rastreo mostrados en el apéndice A incluyen y modifican algunos códigos del conjunto de códigos de rastreo original. Con estas modificaciones al presionar la tecla ñ se obtiene el código de rastreo 0x4C (correspondiente al código de rastreo de la tecla ‘;’ en los teclados en inglés) y con la ecuación 4.1 en la Memoria de Teclado se lee código ASCII correspondiente al carácter ‘ñ’ y no el código ASCII del correspondiente al carácter ‘;’.

4.1.1.3 Bloques de memoria del FPGA

El FPGA contiene 10 bloques de 512 bytes de memoria, un bloque se emplea para almacenar la trama recibida de manera serial (sección 3.4) y 8 bloques forman la ROM generadora de caracteres, llamada ROM_Caracteres. Sin embargo, se pueden considerar como 2 bloques independientes de memoria, así RAM_TRAMA inicia desde la dirección 0x000 y ROM_Caracteres inicia también en la dirección 0x000.

La ROM generadora de caracteres almacena todas las matrices (de 8x16) que representan los caracteres imprimibles en la pantalla. La parte más significativa de la dirección de memoria identifica el carácter que se representa, por ejemplo, la figura 4.4 ilustra la matriz correspondiente al código ASCII del carácter 'A' (0x41).

Dirección ROM	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0x410	0	0	0	0	0	0	0	0
0x411	0	0	0	0	0	0	0	0
0x412	0	0	0	0	0	0	0	0
0x413	0	0	0	0	0	0	0	0
0x414	0	0	0	0	0	0	0	0
0x415	0	0	0	0	**1**	0	0	0
0x416	0	0	0	**1**	0	**1**	0	0
0x417	0	0	**1**	0	0	0	**1**	0
0x418	0	0	**1**	**1**	**1**	**1**	**1**	0
0x419	0	0	**1**	0	0	0	**1**	0
0x41A	0	0	**1**	0	0	0	**1**	0
0x41B	0	0	**1**	0	0	0	**1**	0
0x41C	0	0	0	0	0	0	0	0
0x41D	0	0	0	0	0	0	0	0
0x41E	0	0	0	0	0	0	0	0
0x41F	0	0	0	0	0	0	0	0

Figura 4.4. Matriz de puntos del carácter 'A' en la ROM generadora de caracteres.

Para usar los bloques de memoria existe una utilería de Xilinx Foundation llamada CORE GENERATOR. Con esta utilería fue posible instanciar un módulo que realiza las operaciones de lectura y escritura. El origen de los datos que inicializarán a los bloques de memoria RAM se determina en la fase de diseño del programa por medio de archivos con extensión .coe, esta inicialización se realiza durante la programación del FPGA. Así, las matrices de puntos de la ROM generadora de caracteres están contenidas en el archivo ROM_CAR.coe. Un aspecto importante es que incluye matrices de caracteres con acento que no están presentes en el código ASCII extendido.

A su vez, el archivo TRAMA.coe contiene los datos que inicializarán a la memoria RAM_TRAMA durante la programación del FPGA.

Los archivos ROM_CAR.coe y RAM_TRAMA.coe se encuentran en el disco adjunto a este documento en la carpeta VHDLs.

4.1.2 Descripción del funcionamiento del módulo de Memoria

Inicialmente el módulo de Memoria habilita las señales necesarias para inicializar la SDRAM con la Memoria de Refresco de Pantalla y de Teclado que está contenida en la RAM tipo Flash. Para leer los datos de la Flash se requiere habilitar una salida (RD_FLSH), así como contar con un bus de direcciones de salida y un bus de datos de entrada que contenga el dato leído. Durante este tiempo se aplica una señal de reset interno al resto de los procesos de este módulo.

El módulo controlador de la SDRAM realiza las operaciones de lectura y escritura, éstas son habilitadas por el módulo de Memoria. Cuando el módulo de controlador de la SDRAM ha realizado la operación de lectura o escritura habilita una señal llamada ACK, de manera el módulo de Memoria puede habilitar la siguiente lectura o escritura.

Para realizar los accesos a memoria se debe tomar en cuenta si el módulo controlador de video está aplicando blanqueo a la pantalla.

Si no hay blanqueo se requiere saber la posición actual del carácter a desplegar por lo que se toman las entradas COLCNT y RENGCNT (provenientes del controlador de video) como bus de direcciones de la SDRAM para obtener el código ASCII de la Memoria de Refresco de Pantalla.

La página a mostrar se obtiene del bus de entrada PAGINA (proveniente del módulo de transacción Modbus) y se guarda en el registro interno PAGINA_ACTUAL cada vez que se habilita la entrada PAGINA_NUEVA, así el bus de direcciones para leer el código ASCII de la memoria de refresco será:

```
dir_sig_sdram<="000000000" & PAGINA_ACTUAL(1 downto 0) & rengcnt(4 downto 0)
                & colcnt(6 downto 0);          (Ec.4.2)
```

En la parte más significativa del dato leído de la dirección anterior está el color del carácter a mostrar, así mismo en la parte menos significativa se encuentra el código ASCII del carácter que se almacena en un registro interno llamado CARACTER y que formará parte del bus de direcciones junto con el bus de entrada FILACNT (proveniente del controlador de video e indica la fila que actualmente se está desplegando de la matriz de puntos) y, de esta manera, obtener los puntos a desplegar en la pantalla de la memoria interna ROM_Caracteres, así el bus de direcciones será:

dir_sig_rom <= caracter(7 downto 0) & filacnt(3 downto 0); (Ec.4.3)

La figura 4.5 muestra la descripción en VHDL de la lectura del carácter ASCII y de la lectura de los píxeles a desplegar de la ROM generadora de caracteres. Esta descripción forma parte de una máquina de estados que ejecuta cuando no hay blanqueo.

```
WHEN INICIAL =>      -- Asigna la direccion de la memoria de video
   Guarda_columna       <= '1';     -- Guarda la columna del caracter actual mostrado
   -- Direccion del caracter actual mostrado (memoria de video), notese que esta direccion depende de la pagina
   -- actual
   dir_sig_sdram    <= "000000000" & PAGINA_ACTUAL & rengcnt(4 downto 0) & colcnt(6 downto 0);
   edo_sig1         <= LEE_POSICION;

WHEN LEE_POSICION=> -- Lee el caracter actual en la pantalla que esta en la posicion XY
   IF ack = '0' THEN  -- Operacion de lectura no completada...
      RD_SDRAM1       <= '1';     -- Mantiene la senal de  lectura activa (o se activa)
   ELSE          -- Se ha leido el ASCII de la memoria de video...
      RD_SDRAM1       <= '0' ;
      caracter        := dIn_sdram(7 downto 0);  -- Asigna el caracter a desplegar
      COLOR_SIG       <= dIn_sdram(13 downto 8); -- Asigna el color de los pixeles
      dir_sig_rom     <= caracter(7 downto 0) & filacnt(3 downto 0);    -- Direccion de ROM_Caracteres
      edo_sig1        <= LEE_PIXELES;
   END IF;
```

Figura 4.5. Descripción en VHDL de la lectura del carácter ASCII que se muestra en la pantalla, así como de los píxeles que forman la matriz de puntos del carácter.

Por otra parte, **cuando hay blanqueo** no es necesario el acceso a la Memoria de Refresco de Pantalla para la lectura de códigos ASCII, en ese momento se puede utilizar la memoria SDRAM para:

- Escribir el cursor de posición: Hay un cursor en la pantalla del monitor que indica la posición en que se ubicará el siguiente carácter introducido por el teclado. Cuando no se presiona una tecla, el cursor se encuentra "parpadeando". Para ello, se emplea un contador que habilita una señal interna llamada REALIZA_PARPADEO_CURSOR y así ordenar el cambio de estado del cursor de presente a ausente. Cuando hay un "parpadeo", se debe escribir en la Memoria de Refresco de Pantalla el código ASCII correspondiente al cursor presente (un guión bajo, 0x5F) o ausente (un espacio vacío, 0x20), el bus de direcciones de la SDRAM está formado por los registros internos RENGLON y COLUMNA que representan la posición XY de la pantalla del cursor.

- Leer los bits Modbus: Cuando el módulo de transacción Modbus indica que se deben leer bits de la memoria SDRAM, habilita su salida ORDEN_LECTURA_BITS_MB misma que el módulo de Memoria toma como entrada para habilitar la señal interna ORDEN_LEE_BITS_MB_INT, esto debido a que ORDEN_LECTURA_BITS_MB sólo está habilitada por un ciclo de reloj y es posible que en ese momento no haya blanqueo o se esté realizando un acceso a la SDRAM por lo que no se podrá realizar la lectura. En cambio la señal interna ORDEN_LEE_BITS_MB_INT se deshabilita sólo cuando lectura solicitada se efectúa. La lectura se realizará en la dirección dada por el bus de entrada DIR_SDRAM_MB. Al término de la ésta el módulo de Memoria proporciona en el bus de salida DATO_LEIDO los bits que se leyeron por el controlador de la SDRAM y la salida LECTURA_MB_TERMINADA se habilita durante un ciclo de reloj.

- Escribir registros Modbus: Cuando el módulo de transacción Modbus indica que se deben escribir registros, habilita su salida ORDEN_ESCRITURA_MB, misma que el módulo de Memoria toma como entrada, así como el registro proporcionado por la entrada DATO_MB_A_ESCRIBIR. De manera similar al punto anterior, se emplea una señal interna llamada ORDEN_ESCRITURA_MB_INT que se deshabilita sólo cuando la operación solicitada se efectúa. Al término de la operación de escritura el módulo de Memoria habilita la salida ESCRITURA_MB_TERMINADA durante un ciclo de reloj.

- Mostrar caracteres introducidos por el teclado: Cuando se presiona una tecla, la entrada de este controlador llamada CR_NUEVO se habilita e indica que se debe leer de la Memoria de Teclado el código ASCII correspondiente al código de rastreo, proporcionado por el controlador de teclado, y si es un carácter imprimible, mostrarlo en la pantalla en la posición actual de cursor. Si la entrada MAYSC es '1', quiere decir que el LED de Mayúsculas esta encendido y los caracteres introducidos a mostrar serán de letras mayúsculas o los representados en la parte superior de las teclas numéricas. Si la tecla introducida no tiene código ASCII correspondiente (no es imprimible) se toma como carácter el código ASCII correspondiente al espacio en blanco.

La figura 4.6 muestra la descripción en VHDL que indica la acción a realizar. Esta descripción pertenece a una máquina de estados, por lo que si se debe realizar alguna lectura o escritura de memoria SDRAM, se asigna el estado siguiente en el cual se efectuará dicha acción.

```
IF Realiza_Parpadeo_Cursor='1' and PIDE_CS='1' THEN -- Si se debe realizar un parpadeo del cursor...
    Reset_Realiza_Parpadeo_Cursor  <= '1';        -- Indica que se inicialice la senal Realiza_Parpadeo_Cursor
    Parpadeo_En_Proceso_sig        <= '1';        -- Habilita el indicador de se esta aplicando un parpadeo
    dir_sig_sdram2         <= "11111111111111" & Cur_Pres & "00000000"; -- Dirección caracter del cursor
    edo_sig2               <= OBTIENE_ASCII;
ELSIF ORDEN_LEE_BITS_MB_INT ='1' THEN   -- La transaccion desea usar la SDRAM
    dir_sig_sdram2         <= DIR_SDRAM_MB;  --Direccion SDRAM de la transaccion Modbus
    edo_sig2               <= ESPERA_FIN_DE_LECTURA;
    LIMPIA_ORDEN_LEE_BITS_MB_INT <='1';
ELSIF ORDEN_ESCRITURA_MB_INT='1' THEN-- La transaccion desea usar la SDRAM
    LIMPIA_ORD_ESCRITURA_INT    <= '1';
    ESCRIBE_REG_MB              <= '1'; -- Indica que el registro a escribir sea DATO_MB_A_ESCRIBIR
    dir_sig_sdram2              <= DIR_SDRAM_MB;      --Direccion de la transaccion Modbus
    edo_sig2                    <= ESPERA_FIN_DE_ESCRITURA;
ELSIF Codigo_Rastreo(7 DOWNTO 0) /="00000000" THEN        -- Hay un nuevo codigo de rastreo
    IF Codigo_Rastreo = ENTER and semestre_Capturado ='0' THEN        -- Si corresponde a ENTER...
        dir_sig_sdram2 <= "11111111111111" & '1' & "00000000"; --Direccion de un caracter no imprimible
    ELSE
        dir_sig_sdram2 <= "11111111111111" & MAYSC & Codigo_Rastreo(7 DOWNTO 0); -- MAYUS determina si
                                                                                  -- son mayusculas o no
    END IF;
    edo_sig2 <= OBTIENE_ASCII;   -- Obtiene el ASCII
ELSE
    Cursor_sig<='0';
    Limpia_CR<='1';
END IF;
```

Figura 4.6. Descripción en VHDL que indica la operación a realizar en la SDRAM cuando hay blanqueo.

El diagrama de flujo correspondiente al módulo de Memoria se ilustra en la figura 4.7.

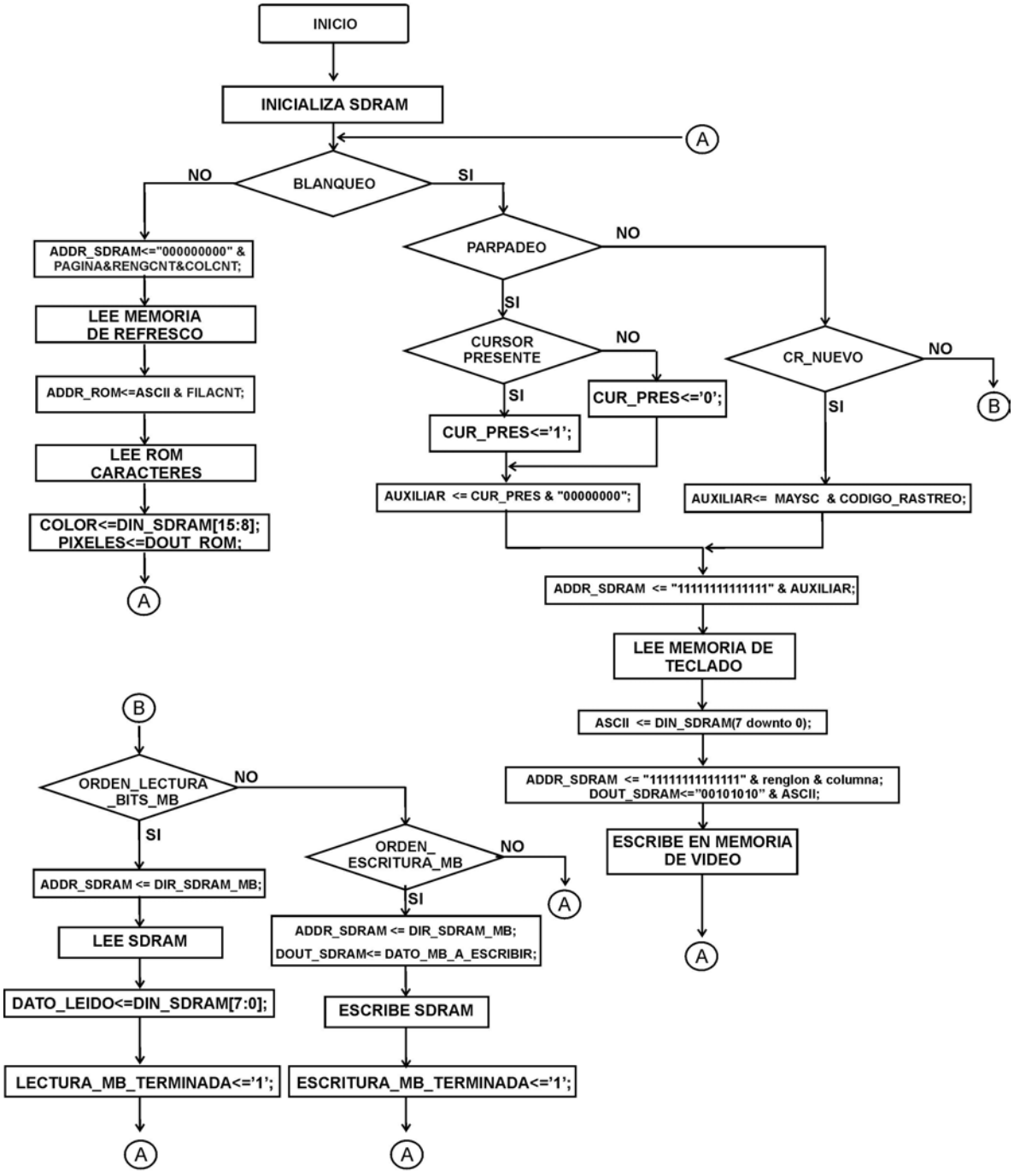

Figura 4.7. Diagrama de flujo del módulo de Memoria.

4.1.3 Entradas y salidas

La figura 4.8 muestra el bloque que representa el símbolo del módulo de Memoria.

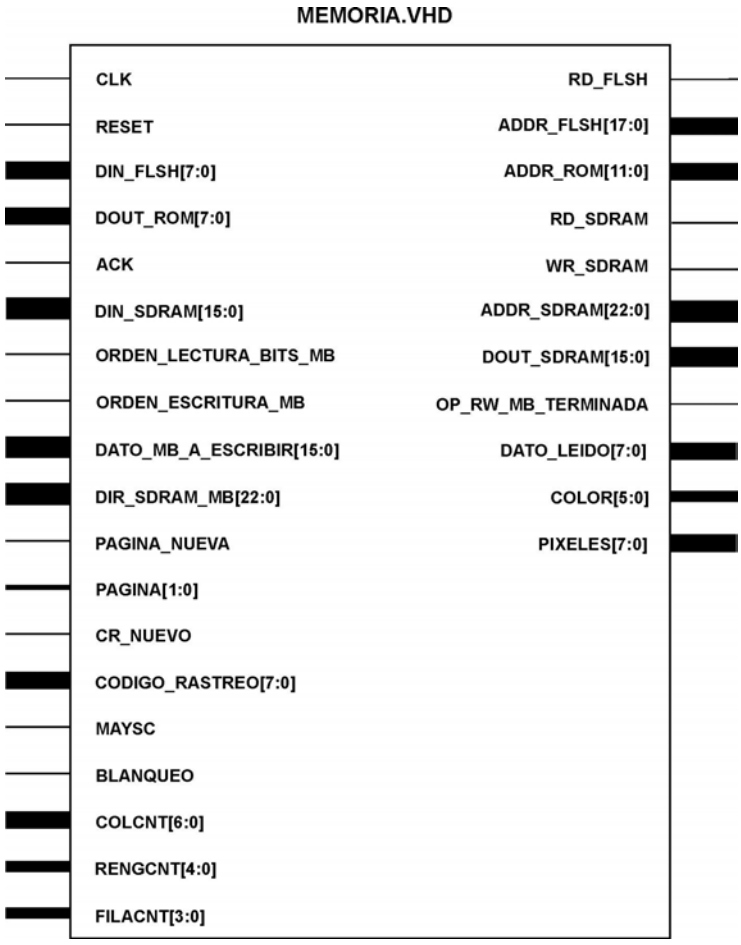

Figura 4.8. Símbolo del módulo de Memoria.

Entradas:

- CLK. Frecuencia de reloj a la que opera el módulo.
- RESET. Entrada para aplicar reset asíncrono al módulo.
- DIN_FLSH[7:0]. Bus de datos de la memoria tipo Flash.
- DOUT_ROM[7:0]. Bus de datos de la ROM generadora de caracteres.
- ACK. Indicador de operación de lectura/escritura terminada en la SDRAM.
- DIN_SDRAM[15:0]. Dato de lectura SDRAM.
- ORDEN_LECTURA_BITS_MB. Orden de lectura de la SDRAM por el código de función 0x01.

- ORDEN_ESCRITURA_MB. Orden de escritura en la SDRAM por el código de función 0x10.
- DATO_MB_A_ESCRIBIR[7:0]. Dato a escribir en la SDRAM de acuerdo al código de función 0x10.
- DIR_SDRAM_MB[22:0]. Dirección de lectura/escritura para Modbus.
- PAGINA_NUEVA. Indicador de que se tiene una nueva página.
- PAGINA[1:0]. Página a mostrar en pantalla.
- CR_LISTO. Indicador de que se tiene un código de rastreo nuevo.
- CODIGO_RASTREO [7:0]. Código de rastreo obtenido por el controlador de teclado.
- MAYSC. Indicador de LED BLOQ MAYÚS encendido/apagado.
- BLANQUEO. Indicador de que se está realizando un blanqueo.
- COLCNT[6:0]. Valor de contador de columnas del controlador de video.
- RENGCNT[4:0]. Valor de contador de renglones del controlador de video.
- FILACNT[3:0]. Valor del contador de filas del controlador de video y que se emplea para formar el bus de direcciones de la ROM generadora de caracteres.

Salidas

- RD_FLSH. Habilita la lectura de RAM tipo Flash.
- ADDR_FLSH[17:0]. Bus de direcciones de la RAM tipo Flash.
- ADDR_ROM[11:0]. Bus de direcciones de la ROM generadora de caracteres.
- RD_SDRAM. Habilita la lectura de memoria en el controlador de SDRAM.
- WR_SDRAM. Habilita la escritura de memoria en el controlador de SDRAM.
- ADDR_SDRAM[22:0]. Bus de direcciones del controlador de SDRAM.
- DOUT_SDRAM[15:0]. Dato a escribir por el controlador de SDRAM.
- LECTURA_MB_TERMINADA. Indicador de lectura terminada al módulo de transacción Modbus.
- ESCRITURA_MB_TERMINADA. Indicador de escritura terminada al módulo de transacción Modbus.
- DATO_MB_LEIDO[7:0]. Dato de lectura SDRAM para el código de función 0x01.

- COLOR[5:0]. Bus que contiene información acerca del color de los píxeles a desplegar, empleado por el controlador de video.
- PIXELES[7:0]. Bus que contiene una fila de la matriz leída de la ROM generadora de caracteres en el módulo de memoria, es decir, los 8 píxeles a desplegar.

4.2 INSTANCIACIÓN

4.2.1 Paquetes

Para la instanciación de componentes se pueden utilizar varios métodos, el empleado aquí considera que los módulos a instanciar forman paquetes en la biblioteca XSLIB, todos ellos están en el disco anexo en la carpeta VHDLs y son los siguientes:

Teclado_pckg: Es el módulo controlador de teclado.
Vga_pckg: Es el módulo controlador de video.
Rx_Tx_Serial_pckg: Módulo de transmisión-recepción serial.
Cta_3_5_pckg: Módulo de espera fin de trama.
Almacena_Trama_pckg: Módulo de almacenamiento de la trama.
Transaccion_Modbus_pckg: Módulo que realiza la transacción Modbus.
Memoria_pckg: Módulo de sincronización de accesos a memoria SDRAM.
Common y xilinx: Contienen una función y algunas declaraciones del módulo controlador de la SDRAM.
Sdram_pckg: Controlador de la SDRAM.

4.2.2 Módulo de alto nivel

El módulo de alto nivel realiza la instanciación de los componentes, es decir, realiza la descripción de las conexiones. Para la aplicación desarrollada, lleva a cabo las siguientes funciones:

- Declara las bibliotecas.
- Establece los valores constantes utilizados tal como la dirección del esclavo destino y la velocidad de transmisión, esto en la parte de declaraciones genéricas.
- Declara los puertos de entrada, salida y bidireccionales.

- Declara las señales de interconexión y define las señales de los componentes de RAM_TRAMA y ROM_Caracteres.
- Instancia los bloques de memoria RAM_TRAMA, ROM_Caracteres y al resto de los componentes.

El archivo del módulo de alto nivel es Modulo_Alto_Nivel.VHD y se encuentra en el disco anexo en la carpeta VHDLs.

Finalmente, la figura 4.9 muestra la conexión de todos los módulos Modbus y el módulo de Memoria, así mismo la figura 4.10 muestra la conexión de los módulos de Interfaz con el módulo de Memoria. El sistema es uno solo pero se divide en dos figuras, esto debido a que no se puede ilustrar en una sola página de manera que se puedan distinguir todos los elementos.

Nótese como en la figura 4.9 la señal de reset RST_INT2 proviene del reset interno de la figura 4.10 y sólo afecta al módulo de transacción Modbus. Por su parte, el reset Modbus (reset_MB) que se aplica en el módulo de transacción afecta al resto de los módulos.

La figura 4.10 además muestra el símbolo del controlador de la SDRAM, los bloques en gris representan los periféricos a los que se tiene acceso.

La asignación de terminales del FPGA (en la forma en que se declaran) se encuentra en el apéndice E.

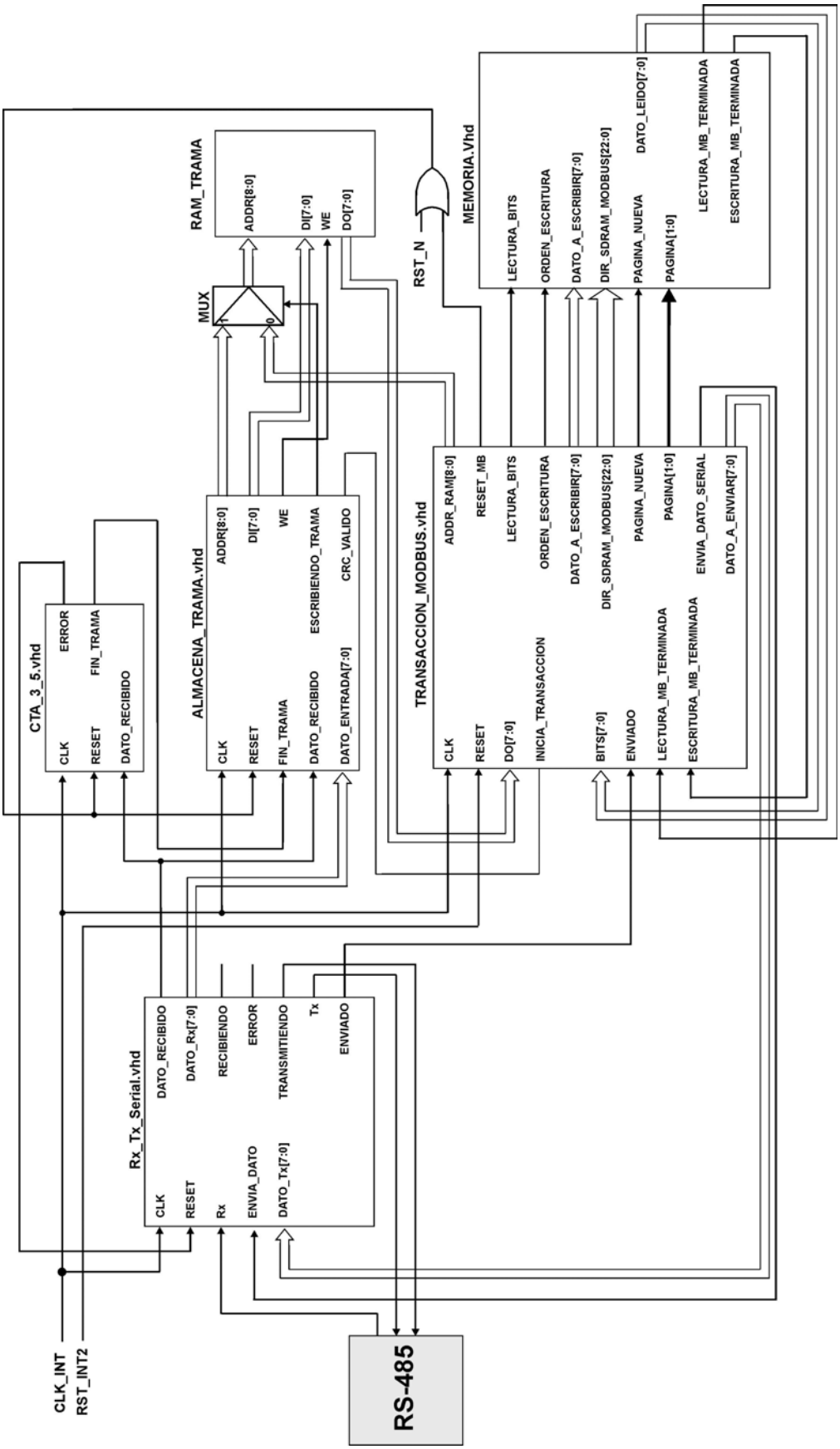

Figura 4.9. Conexión de los módulos Modbus con el módulo de Memoria.

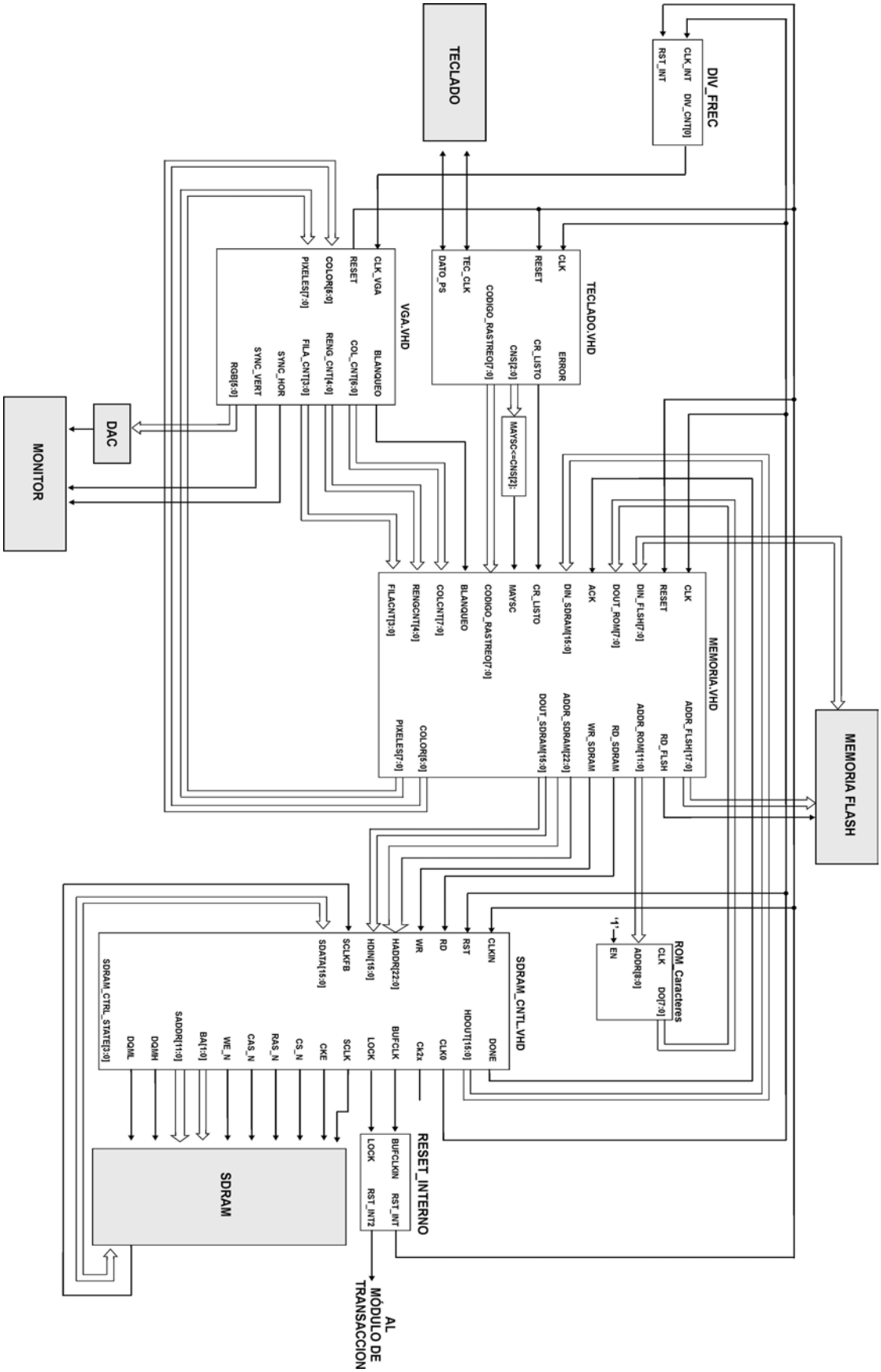

Figura 4.10. Conexión de los módulos de Interfaz con el módulo de Memoria.

5. NODO MAESTRO

El nodo maestro está programado en C++ Builder (Versión 6.0). Su función principal es el envío de tramas a los nodos esclavos, aunque también realiza otras acciones tal y como se describe en la sección 5.4. Para el ejemplo del sistema de consultor de calificaciones planteado, el nodo maestro debe:

- Enviar a cada esclavo una trama de *solicitud de calificaciones*.
- Acceder a una base de datos para obtener las calificaciones y enviarlas al esclavo por medio de tramas.

5.1 SONDEO

El nodo maestro realiza sondeo a cada esclavo, es decir, envía a cada esclavo una trama de *solicitud de calificaciones* la cual tiene un código de función de 0x01 con el objeto de leer los 8 bits menos significativos de la localidad 0x4000 de la SDRAM. La razón es la siguiente: cuando el usuario ha introducido su semestre y código en el nodo esclavo, el módulo de Memoria escribe el byte 0x5A en la dirección 0x4000 para hacerle "saber" al nodo maestro (por medio de la trama de *solicitud de calificaciones*) que está solicitando información.

5.2 ACCESO A LA BASE DE DATOS

Al momento que el nodo maestro detecta la solicitud de información del nodo esclavo, envía nuevamente 2 tramas:

- La primera pide al nodo esclavo que lea desde la dirección de inicio 0x7A7 (localidad donde se encuentra el semestre) 16 bits de la Memoria de Refresco, es decir, los 2 bytes que corresponden al semestre introducido por el usuario. La memoria SDRAM tiene localidades de 16 bits, pero con la lectura de bits sólo se leen los 8 bits menos significativos (LSB) de cada localidad, es decir, con esta trama se lee la parte menos significativa de 2 localidades de memoria. Esto es así, puesto que la parte menos significativa de las direcciones de la RAM de Refresco contienen los códigos ASCII de los caracteres que se muestran en la pantalla, y la parte más significativa contiene

la información del color del carácter desplegado, dato que no es importante para el nodo maestro en el sondeo.

- La segunda trama solicita la lectura del código del alumno, es decir, que lea 64 bits comenzando en la dirección 0x8A7, o lo que es lo mismo, que lea los 8 bits menos significativos de 8 localidades de la memoria de refresco.

El nodo maestro revisa en las tramas de respuesta que la información solicitada sea la correcta, es decir: se realiza la CRC, que el semestre sea mayor que cero y menor que once, que el código sea numérico y exista en la base de datos. En caso de que alguna de las condiciones anteriores no se cumpla, el nodo maestro envía al esclavo una trama con código de función 0x41 que contiene en el campo de datos ya sea 0x01 o 0x03, el módulo de transacción del esclavo toma la parte menos significativa de este byte y la coloca en el registro PAGINA (capítulo 4) para indicar la página a mostrar en la pantalla. La página 0x01 corresponde a la aparición del mensaje de *código inexistente* y la página 0x03 al mensaje de *semestre inexistente*. Así, de ser el caso, el usuario puede observar en el monitor que ha introducido algún dato de manera errónea.

Si por el contrario, el semestre y código introducidos por el usuario son correctos, el nodo maestro prepara una trama con código de función 0x10 para que escriba en la SDRAM sólo el código del alumno en la dirección 0x228D (que pertenece a la página 2) de la RAM de refresco de pantalla. No se envía más información, como el nombre del alumno, por discreción ante otros usuarios que intenten acceder a calificaciones que no les corresponden. Después el nodo maestro envía las calificaciones de los alumnos.

5.2.1 Envío de calificaciones al nodo esclavo

Todas las tramas con la información de las calificaciones que el nodo maestro envía al nodo esclavo tienen el código de función 0x10 para que éste las escriba en el rango de direcciones que se indique en el campo de datos de la trama. La RAM de refresco emplea la parte menos significativa de una localidad para almacenar el código ASCII del carácter que se muestra en la pantalla, y la parte más significativa contiene el color del carácter. Por ello, el nodo maestro sigue este patrón al enviar un registro a escribir en el esclavo: en la parte

más significativa del registro la información del color y en la parte menos significativa el carácter que se mostrará en la pantalla.

Inicialmente la tabla que se muestra en la pagina de calificaciones (página 2) del esclavo está vacía, y tiene la forma que se presenta en la tabla siguiente.

Materia	Parcial			Final	Extraord		Esp
	1ro	2do	3ro		1ro	2do	

Tabla 5.1. Forma inicial de la tabla de la página de calificaciones

La parte menos significativa de los registros que el nodo maestro envía para su escritura en el nodo esclavo corresponden a los códigos ASCII de:

- Los nombres de las materias. Se escriben en la memoria de video de manera que en la pantalla aparezcan en la columna *Materia* de la tabla anterior. La cantidad máxima de caracteres por materia es de 21.
- Líneas divisorias. Son las líneas que dividen las calificaciones de la tabla de manera vertical, es decir, son líneas del carácter '|'. La razón del envío de estas líneas es que el código de función 0x10 indica la escritura de registros contiguos y, dado que los caracteres mostrados en la pantalla se encuentran en localidades continuas, el carácter que divide a las calificaciones también se envía. Es posible no enviar este carácter, sin embargo, esto resultará en el envío de una mayor cantidad de tramas.
- Calificaciones. Se ubican debajo del resto de las columnas de la tabla, de no existir la calificación, se envía el carácter de espacio vacío 0x20.

La tabla tiene una capacidad para mostrar hasta 6 materias con sus respectivas calificaciones, es decir, el número máximo de tramas con calificaciones que el nodo

maestro envía es de 6. Si el número de materias es menor, el nodo maestro debe enviar tramas con caracteres del espacio vacío.

El nodo esclavo envía una trama de respuesta para cada trama de escritura de registros enviada por el maestro, con ello se le indica al nodo maestro que puede enviar la siguiente trama.

Cuando se han enviado las 6 tramas de calificaciones, el nodo maestro envía una última trama con código de función 0x41 que en el campo de datos tiene 0x02, con esto el esclavo muestra al alumno la página 2 con las calificaciones solicitadas. El nodo maestro dirige una trama de *solicitud de calificaciones* al siguiente esclavo y el proceso descrito anteriormente se repite de manera cíclica, es decir, al terminar con el último esclavo se realiza el sondeo para el primer esclavo.

El proceso del envío de tramas se muestra en el diagrama de flujo de la figura 5.1. La primera trama que se envía realiza una petición de lectura al esclavo N (esclavo destino) y espera su respuesta, cuando ésta llega se revisa el byte leído en la localidad 0x4000: si no es 0x5A, se incrementa el contador N para pasar al esclavo siguiente, de otro modo, el esclavo destino desea información de la base de datos, así el nodo maestro envía una trama para leer el semestre del esclavo y posteriormente una trama para leer el código. Si éstos no son correctos, se envían tramas para que el esclavo asigne una página que es la que se mostrará en la pantalla e indicará un mensaje de error; si por el contrario, los datos son correctos, el nodo maestro envía una trama para escribir el código del alumno en la página 2 y después 6 tramas con los nombres de las materias, líneas divisorias y las calificaciones deseadas. Finalmente, el nodo maestro envía una trama para que el nodo esclavo asigne la página 2 y el usuario pueda observar sus calificaciones en el nodo esclavo.

El nodo maestro agrega la CRC a cada trama que envía, tal como Modbus sobre línea serial lo establece (sección 1.6.3.4), de igual forma, se obtiene la CRC de cada trama recibida. Puesto que pueden ocurrir errores en la transmisión o recepción de datos, el nodo maestro vuelve a enviar la trama cuando no recibe una respuesta o cuando hay error en la CRC de la respuesta, esto se realiza hasta 7 veces (Modbus describe que el diseñador del sistema

multipunto establezca el número de errores permitido), es decir, si se detectan menos de 8 errores en la recepción de datos el nodo esclavo vuelve a enviar la trama, de otra forma, se considera que el nodo esclavo no tiene alimentación.

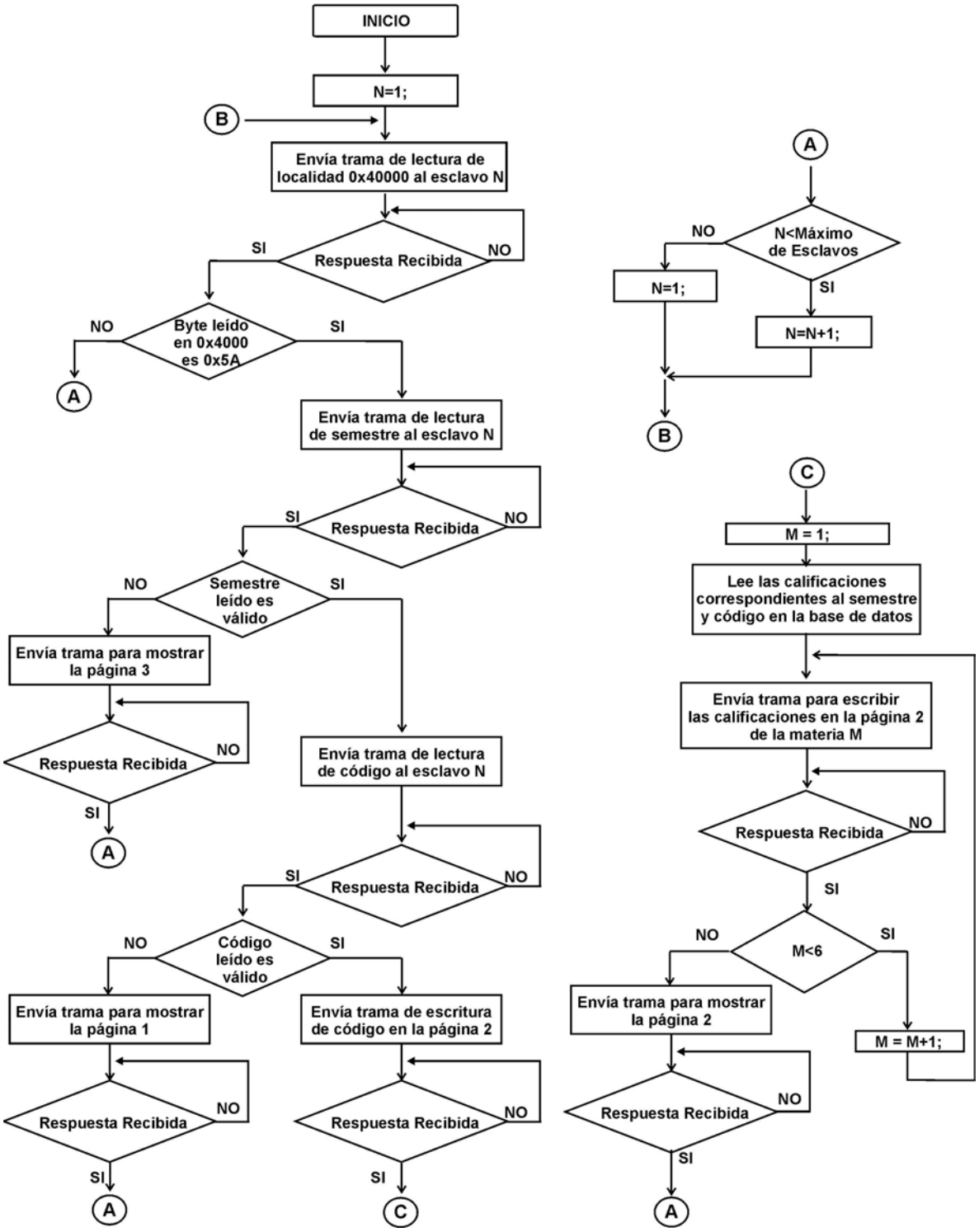

Figura 5.1. Diagrama de flujo del envío de tramas del nodo maestro

5.3 INTERFAZ PC/RS-485

Para la interfaz con de la PC con el estándar RS-485 se emplea un convertidor de niveles RS-232 (del puerto serial de la PC) a niveles TTL, para la transmisión y recepción de datos mediante el chip MAX232 (figura 5.2).

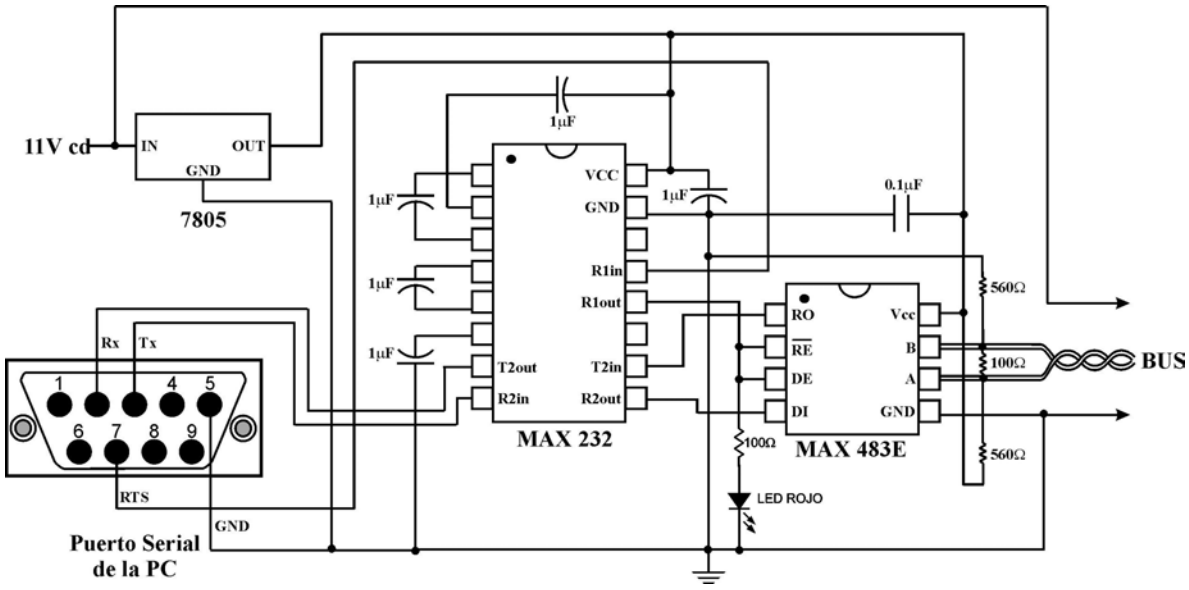

Figura 5.2. Conexión de circuitos electrónicos en el nodo maestro.

El circuito 7805 proporciona 5Vcd a su salida de los 11Vcd de alimentación, con los cuales se alimenta a cada circuito integrado. El puerto serial proporciona la tierra, misma que, en conjunción con los 11Vcd, se dirige al bus para alimentar a cada nodo esclavo. La señal RTS (request to send) que proviene del puerto serie habilita al MAX483E para que éste funcione como receptor o transmisor, e ilumina al LED rojo indicando que se está transmitiendo. Cuando el nodo maestro desea enviar tramas, pone la señal RTS en un nivel lógico alto para habilitar la transmisión en el MAX483E y al finalizar inmediatamente deshabilita RTS.

5.4 ACTUALIZACIÓN DE LA BASE DE DATOS

Además de las acciones descritas, el nodo maestro para el ejemplo planteado en este trabajo también se encarga de la actualización de la base de datos, permitiendo:

- Introducir nuevos alumnos.
- Modificar calificaciones de un alumno.
- Eliminar un alumno de la base de datos.
- Asignar las materias de determinada carrera.

La actualización de la base de datos es muy sencilla, en cada caso sólo se deben asignar los datos que se piden en los campos que aparecen en las ventanas para cada opción del programa del nodo maestro.

Las acciones descritas son independientes de la relación maestro-esclavo, esto gracias al uso de hilos en C++ Builder con los que es posible ejecutar dos aplicaciones con un programa, de tal forma que puede actualizar la base de datos sin afectar la relación de transmisión y recepción del nodo maestro con los nodos esclavo.

El programa del nodo maestro se llama Maestro.exe y se encuentra, junto con su código, en la carpeta C del disco anexo a este documento. Este programa emplea los archivos alumnos.$$$ y carreras.$$$ que contienen los datos de los alumnos y las materias de las carreras, respectivamente. Al ejecutar el programa estos archivos deben encontrarse en la misma carpeta que Maestro.exe del disco duro de la computadora.

6. RESULTADOS Y CONCLUSIONES

6.1 RESULTADOS OBTENIDOS

Para que funcione un nodo esclavo, en el sistema multipunto deben cumplirse las siguientes condiciones:

- Alimentarlo con el conector adjunto en su terminal de alimentación.
- Que se conecte el circuito del nodo maestro (figura 5.2) al puerto serial COM1.
- Que el programa maestro.exe se ejecute en el nodo maestro.
- Que el circuito del nodo maestro se encuentre encendido para que proporcione el voltaje y tierra en el bus.
- Que las terminales del bus se conecten al esclavo de acuerdo con la figura 3.16.

La figura 6.1 muestra a un esclavo conectado a un monitor y un teclado, el nodo esclavo cumple con las condiciones anteriores.

Figura 6.1. Nodo esclavo en funcionamiento.

Inicialmente el monitor muestra la página correspondiente a la petición de datos en el nodo esclavo, la figura 6.2 muestra parte de esta página.

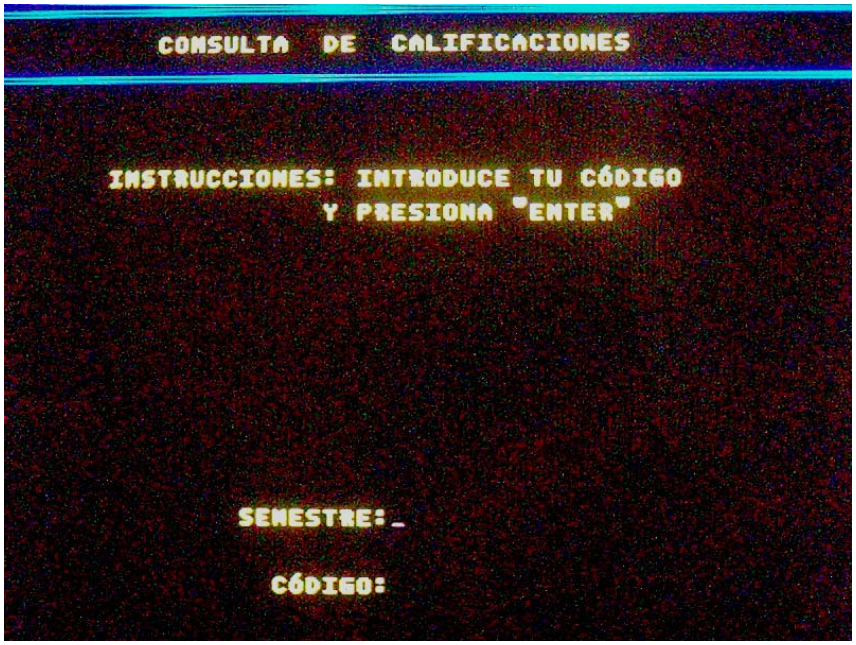

Figura 6.2. Página de petición de datos al usuario.

Al presionar una tecla de un carácter imprimible, éste se muestra en la posición en la que se encuentra el cursor. Los LEDs del teclado se encienden o apagan al presionar las teclas BLOQ MAYÚS, BLOQ NÚM y BLOQ DESPL. Aunque no se requiere para esta aplicación, se implementó (por el módulo de Memoria) el uso de la tecla BLOQ MAYÚS, es decir, se pueden introducir letras mayúsculas y minúsculas. Esto sólo con fines demostrativos, debido a que la entrada de datos del usuario únicamente necesita de teclas numéricas. También se implementó el uso de la tecla BackSpace para borrar un carácter introducido; algunas teclas que no se emplean, como las de función, introducen un espacio en blanco al presionarlas.

Al introducir el semestre de manera errónea (caracteres que no sean numéricos, espacios en blanco, un semestre mayor a 10 o menor que 1) el nodo maestro envía una trama indicando que aparezca la página de semestre inexistente en el nodo esclavo (figura 6.3).

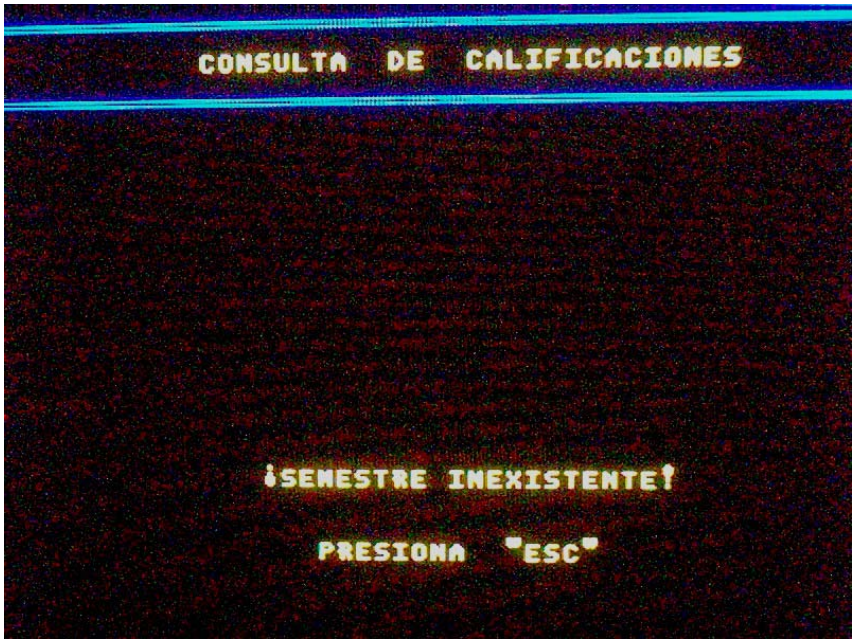

Figura 6.3. Página de semestre inexistente.

De manera similar, al introducir el código de manera errónea, el nodo maestro envía una trama indicando que aparezca la página de código inexistente (figura 6.4).

Figura 6.4. Página de código inexistente.

Si los datos introducidos son correctos, el nodo maestro envía el código y los datos de calificaciones al nodo esclavo para que se desplieguen en la página de calificaciones (figura 6.5). Como se puede observar, las calificaciones menores a 6 se muestran en color rojo y las restantes, así como el nombre de las materias, en color blanco.

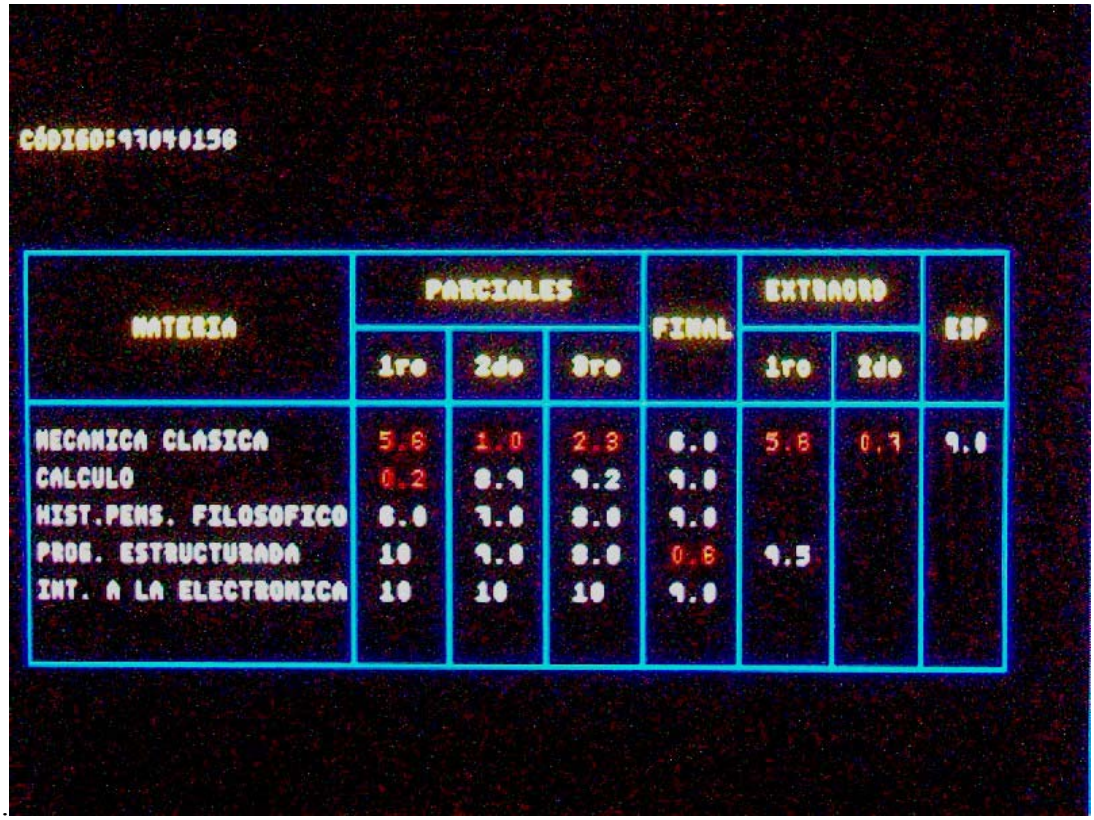

Figura 6.5. Página de calificaciones.

En la figura 6.6 se puede observar a 2 nodos esclavos conectados al bus.

Figura 6.6. Conexión de 2 esclavos al bus.

El programa del nodo maestro en ejecución se muestra en la figura 6.7

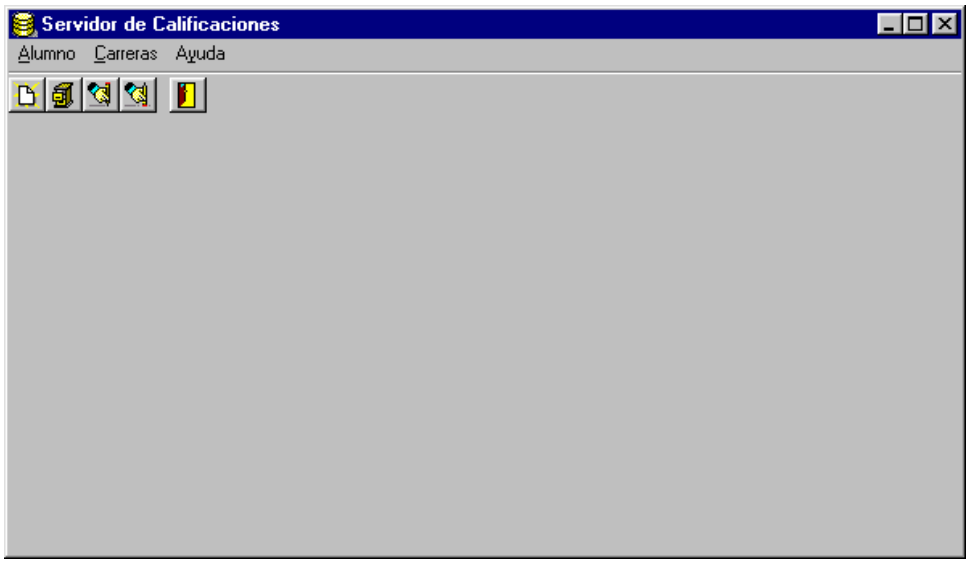

Figura 6.7. Programa del nodo maestro en ejecución.

Con sólo ejecutar el programa del nodo maestro, éste envía y recibe tramas sin la necesidad de elegir una opción, esto debido a que al crear la ventana principal, el programa maestro.exe declara un hilo encargado de la comunicación serial y se pone en ejecución con una prioridad alta para minimizar errores durante la transmisión. Los botones y el menú que

ahí aparecen se emplean para modificar la base de datos, para ello se debe seleccionar la acción a realizar con el ratón y llenar los campos que aparecen.

La figura 6.8 muestra la tabla de calificaciones en el programa del nodo maestro que corresponde a los datos mostrados en el nodo esclavo de la figura 6.5.

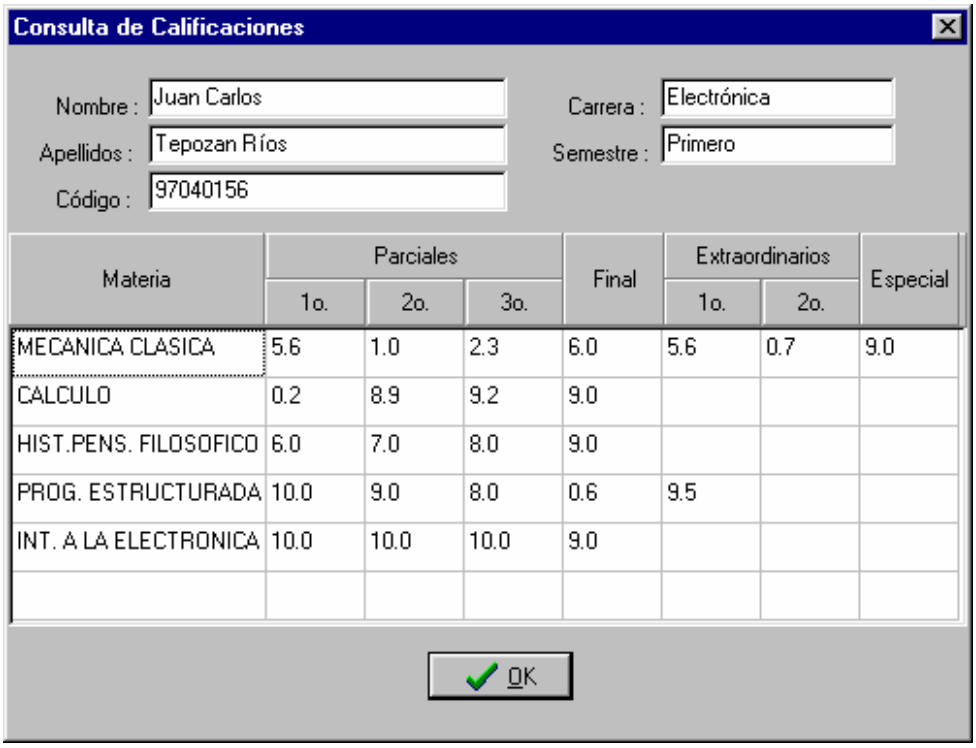

Figura 6.8. Calificaciones mostradas en el nodo maestro que se despliegan en el nodo esclavo de la figura 6.5.

Para finalizar con la ejecución del programa se debe cerrar la ventana o elegir la opción salir (figura 6.9), con ello se destruye al hilo encargado de la comunicación serial, de manera que también termina la transmisión y recepción de tramas. En ese momento los nodos esclavos sólo mostrarán la página en la que se encuentren (o la página de petición de calificaciones si se presiona ESC) y si el usuario intenta llevar a cabo una consulta, ésta será ignorada porque el nodo maestro ha dejado enviar tramas a los nodos esclavos.

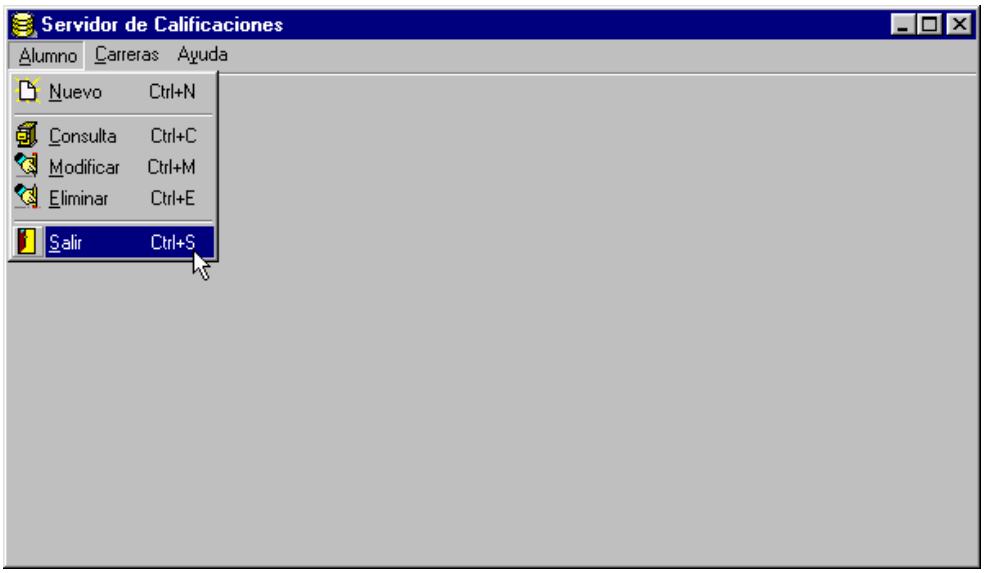

Figura 6.9. Elección de la opción salir del programa del nodo maestro.

En la tabla 6.1 se muestra el porcentaje de utilización de los diferentes elementos que integran al FPGA para el nodo esclavo, con ello la frecuencia máxima obtenida es de 55.429Mhz (si se emplea una frecuencia mayor puede ser que el diseño no funcione de manera adecuada, dependiendo de factores como la variación de temperatura y de voltaje), la cual es mayor a la frecuencia por defecto (sección 1.3).

Elementos	Cantidad usada	Porcentaje	Descripción
Slices	951 de 1200	79%	Partes pertenecientes a los CLBs
LUTs de 4 entradas	1317 de 2400	54%	Tablas de consulta
IOBs	83 de 92	90%	Bloques de entrada y salida
Bloques de RAM	9 de 10	90%	Bloques de RAM del FPGA
GLKIOBs	2 de 4	50%	Señal de reloj principal y señal de reloj con retardos (usada por los DLLs)
DLLs	2 de 4	50%	Bloques para sincronizar la señal de reloj principal con la SDRAM

Tabla 6.1. Porcentaje de los elementos empleados en el nodo esclavo.

Las tablas 6.2 a 6.7 muestran los porcentajes empleados en el módulo correspondiente que ahí se indica[8].

Módulo controlador de teclado		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	117 de 1200	9%
LUTs de 4 entradas	195 de 2400	8%
IOBs	15 de 92	16%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.2. Porcentaje de los elementos empleados en el módulo controlador de teclado.

Módulo controlado de video		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	39 de 1200	3%
LUTs de 4 entradas	66 de 2400	2%
IOBs	40 de 92	43%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.3. Porcentaje de los elementos empleados en el módulo controlador de video.

Módulo de recepción-transmisión serial		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	59 de 1200	4%
LUTs de 4 entradas	75 de 2400	3%
IOBs	14 de 92	15%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.4. Porcentaje de los elementos empleados en el módulo de recepción-transmisión serial.

[8] Para obtener los porcentajes se consideró cada módulo de manera independiente, es decir, que en cada módulo sus salidas y entradas se asignan a terminales del FPGA. Por lo tanto, se puede observar que la cantidad de bloques de entrada y salida corresponden a la cantidad de entradas y salidas declaradas en la entidad y, a causa de ello, no aparece el módulo de Memoria ya que su cantidad de entradas y salidas supera al número de terminales del FPGA.

Módulo de espera fin de trama		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	38 de 1200	3%
LUTs de 4 entradas	36 de 2400	1%
IOBs	4 de 92	4%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.5. Porcentaje de los elementos empleados en el módulo de espera fin de trama.

Módulo de almacenamiento de la trama		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	50 de 1200	4%
LUTs de 4 entradas	42 de 2400	1%
IOBs	31 de 92	33%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.6. Porcentaje de los elementos empleados en el módulo de almacenamiento de la trama.

Módulo de transacción Modbus		
Elementos	**Cantidad usada**	**Porcentaje**
Slices	293 de 1200	24%
LUTs de 4 entradas	377 de 2400	15%
IOBs	84 de 92	91%
Bloques de RAM	0 de 10	0%
GLKIOBs	1 de 4	25%
DLLs	0 de 4	0%

Tabla 6.7. Porcentaje de los elementos empleados en el módulo de transacción Modbus.

Finalmente la tabla 6.8 muestra la frecuencia máxima correspondiente a cada módulo.

Módulo	Frecuencia Máxima (Mhz)
Teclado	72.296
Video	86.237
Recepción-transmisión serial	114.561
Espera fin de trama	104.822
Almacenamiento de la trama	95.841
Transacción Modbus	48.952

Tabla 6.8. Frecuencia máxima correspondiente a cada módulo.

Note como en la tabla 6.8 la frecuencia máxima del módulo de transacción Modbus es menor a la frecuencia por defecto, sin embargo al realizar la instanciación la frecuencia máxima cambia, tal es el caso del presente trabajo en el que al realizar la instanciación de todos los módulos se supera la frecuencia por defecto.

6.2 CONCLUSIONES

El controlador de teclado desarrollado (con las entradas y salidas que ofrece) cubre las necesidades que pueda tener el diseñador al realizar una aplicación. La característica principal de este controlador es que configura al teclado con el conjunto de códigos de rastreo 3, ya que es más simple manejar este conjunto con respecto al conjunto de códigos de rastreo 2. Una característica de este controlador es que sólo ofrece los códigos de rastreo y no los códigos de liberación, ya que detecta cuando se ha dejado de presionar una tecla pero no proporciona en sus salidas los códigos correspondientes a esa acción.

El controlador de video genera las señales necesarias para que el monitor muestre los caracteres que se encuentran en la RAM de refresco de pantalla y emplea una frecuencia de 25Mhz con la que se obtiene una resolución de 629x480 píxeles. Para aplicaciones que no requieran de una gran cantidad de RAM de refresco, ésta puede ser asignada en bloques de la RAM del FPGA, el principio para realizar las operaciones de lectura y escritura de este tipo de memoria se puede ver en el módulo de almacenamiento de la trama, el módulo de transacción Modbus y módulo de Memoria. Este controlador genera las señales de sincronización y de blanqueo con contadores que además se emplean para obtener el carácter y los píxeles que se muestran en el monitor, es decir, sirven para realizar accesos a la memoria de teclado y de video. Esto representa una ventaja frente a otros controladores que tienen contadores de ciclos que: sólo se emplean para generar las señales de sincronización y blanqueo, que sólo muestran líneas verticales de colores en la pantalla (tal es el caso del controlador de video que se realiza en [2]) o que sólo tienen acceso a una memoria.

En el caso de los módulos Modbus sólo se implementaron los códigos de función necesarios en la aplicación de este trabajo (2 códigos de función públicos y uno definido para mostrar las páginas en la pantalla) puesto que la capacidad del FPGA no permite que se definan todos los códigos de función de Modbus. Sin embargo, el módulo de transacción funciona de manera adecuada ante tramas con los códigos de función no implementados, es decir, responde con una respuesta de excepción al maestro indicando que no soporta un código de función no definido.

Por otra parte, el módulo de Memoria realiza el enlace de los otros módulos, con él es posible realizar los accesos a la SDRAM de manera ordenada. Cuenta máquinas de estados que determinan el orden en que se realiza una operación de lectura o escritura a la SDRAM, esto debido a que puede haber una petición de una de estas operaciones por 2 módulos al mismo tiempo.

Con el ejemplo desarrollado se demuestra que los módulos cuentan con las señales suficientes para coordinarse con el objetivo de cumplir con un propósito específico. El ejemplo sólo muestra el uso de los módulos, sin embargo, se considera que la principal aportación del presente trabajo es la de proporcionar un conjunto de controladores de fácil manejo para que puedan ser utilizados en diferentes aplicaciones, para el mismo FPGA o para cualquier otro de características similares (realizando los cambios pertinentes), esto por la portabilidad que tiene VHDL en forma implícita.

6.3 EXPECTATIVAS

Con el avance de la tecnología se espera la fabricación de FPGAs de mayor capacidad y de tarjetas de prototipado de sistemas digitales más complejas. Sin embargo, muchas de las tarjetas nuevas siguen teniendo conectores de teclado y/o de video entre sus componentes. Así que los controladores desarrollados pueden emplearse en estas tarjetas, teniendo en cuenta la frecuencia a la cual se utilizan y que las terminales del FPGA varían de acuerdo a la tarjeta.

De esta manera, es posible emplear los controladores de interfaz de manera independiente:

- El controlador de teclado puede ser útil en aplicaciones que requieran introducir datos para que se realicen ciertas acciones, por ejemplo iniciar o detener la ejecución de una orden que afecte a la aplicación desarrollada. Además, este controlador puede modificarse, según las necesidades del diseñador, para que acepte más órdenes o utilice una frecuencia de trabajo diferente.
- El controlador de monitor se puede configurar a otra frecuencia, lo que traerá consigo una nueva resolución en la pantalla; o bien puede servir como base para el desarrollo de un controlador que despliegue imágenes (con las limitantes de resolución y color que esto implica). Si se desea tener una mayor variedad de estilos de los caracteres

desplegados será necesario almacenar sus matrices de puntos en la ROM generadora de caracteres.

Con las modificaciones adecuadas, los controladores de teclado y video se pueden emplear para realizar aplicaciones que requieran una interfaz de entrada de datos y visual con el usuario como, por ejemplo: la comunicación entre 2 tarjetas de prototipado de sistemas digitales, en aplicaciones con fines educativos, en prototipos de procesadores RISC, etc.

Por otra parte, los módulos Modbus que se desarrollaron pueden emplearse en aplicaciones distintas en las que se requiera: transmisión serial, almacenamiento y lectura de datos en la memoria interna, transacciones Modbus diferentes a las desarrolladas, etc.

Por su parte, en el módulo de Memoria se puede aprovechar la máquina de estados utilizada para leer la memoria RAM tipo Flash en diseños que requieran leer datos no volátiles.

Por último, es posible modificar los módulos del ejemplo desarrollado en este trabajo para realizar una aplicación distinta en cada nodo esclavo (esto implicaría cambios en el programa del nodo maestro, agregar o quitar transacciones en el nodo esclavo, modificar el contenido de las páginas, etc.) y que no necesariamente requiera de todos los módulos de este trabajo. Por ejemplo: un nodo esclavo que realice el monitoreo de señales y las envíe al nodo maestro, otro nodo esclavo que sólo muestre información en la pantalla, un esclavo que tenga una aplicación de control, etc.

APÉNDICE A

Conjunto de códigos de rastreo 3 (todos los valores están en hexadecimal).

Tecla	Código de rastreo	Código de liberación	Tecla	Código de rastreo	Código de liberación
A	1C	F0,1C	0	45	F0,45
B	32	F0,32	1	16	F0,16
C	21	F0,21	2	1E	F0,1E
D	23	F0,23	3	26	F0,26
E	24	F0,24	4	25	F0,25
F	2B	F0,2B	5	2E	F0,2E
G	34	F0,34	6	36	F0,36
H	33	F0,33	7	3D	F0,3D
I	43	F0,43	8	3E	F0,3E
J	3B	F0,3B	9	46	F0,46
K	42	F0,42	BackSpace	66	F0,66
L	4B	F0,4B	Space	29	F0,29
M	3A	F0,3A	TAB	0D	F0,0D
N	31	F0,31	Bloq Mayús	14	F0,14
Ñ	4C	F0,4C	SHIFT Izquierdo	12	F0,12
O	44	F0,44	CTRL Izquierdo	11	F0,11
P	4D	F0,4D	WIN Izquierdo	8B	F0,8B
Q	15	F0,15	ALT Izquierdo	19	F0,19
R	2D	F0,2D	SHFT Derecho	59	F0,59
S	1B	F0,1B	CTRL Derecho	58	F0,58
T	2C	F0,2C	WIN Derecho	8C	F0,8C
U	3C	F0,3C	ALT Derecho	39	F0,39
V	2A	F0,2A	APPS	8D	F0,8D
W	1D	F0,1D	ENTER	5A	F0,5A
X	22	F0,22	ESC	08	F0,08
Y	35	F0,35	°	0E	F0,0E
Z	1A	F0,1A	Ç	5C	F0,5C

Tecla	Código de rastreo	Código de liberación	Tecla	Código de rastreo	Código de liberación
F1	07	F0,07	Bloq Num	76	F0,76
F2	0F	F0,0F	KP /	77	F0,77
F3	17	F0,17	KP *	7E	F0,7E
F4	1F	F0,1F	KP -	84	F0,84
F5	27	F0,27	KP +	7C	F0,7C
F6	2F	F0,2F	KP Enter	79	F0,79
F7	37	F0,37	KP .	71	F0,71
F8	3F	F0,3F	KP 0	70	F0,70
F9	47	F0,47	KP 1	69	F0,69
F10	4F	F0,4F	KP 2	72	F0,72
F11	56	F0,56	KP 3	7A	F0,7A
F12	5E	F0,5E	KP 4	6B	F0,6B
Impr Pant	57	F0,57	KP 5	73	F0,73
Bloq Despl	5F	F0,5F	KP 6	74	F0,74
Pausa	62	F0,62	KP 7	6C	F0,6C
Insert	67	F0,67	KP 8	75	F0,75
Inicio	6E	F0,6E	KP 9	7D	F0,7D
Re Pág	6F	F0,6F	`	4E	F0,4E
Supr	64	F0,64	¡	55	F0,55
Fin	65	F0,65	,	41	F0,41
Av Pág	6D	F0,6D	.	49	F0,49
Flecha arriba	63	F0,63	-	4A	F0,4A
Flecha izquierda	61	F0,61	`	54	F0,54
Flecha abajo	60	F0,60	+	5B	F0,5B
Flecha derecha	6A	F0,6A	´	52	F0,52

APÉNDICE B

Órdenes del teclado utilizadas en el controlador de teclado.

Nota: Cuando el controlador envía una orden, debe esperar a recibir el byte de reconocimiento 0xFA.

Las órdenes empleadas son:

Reset (0xFF). Aplica reset al teclado, con esto el teclado realiza una prueba de diagnóstico. Durante este lapso, enciende sus LEDs y programa los siguientes valores:

Retardo Typematic 500 ms.

Typematic rate 10.9 caracteres por segundo.

Conjunto de código de rastreo 2.

Teclas en modo typematic/rastreo/liberación.

Al finalizar, el teclado apaga los LEDs y envía al controlador el código de prueba exitosa 0xAA, o de error 0xFC.

Encender/Apagar los LEDs. Para Indicarle al teclado que encienda o apague sus LEDs, el controlador debe enviarle 2 órdenes:

La primera es 0xED, que le indica al teclado que se trata de una orden de encender o apagar sus LEDs.

La segunda le indica al teclado los LEDs específicos que se encienden o apagan, este byte es "00000CNS". Donde:

C representa el LED "Bloq Mayús"

N representa el LED "Bloq Núm"

S representa el LED "Bloq Despl"

Por ejemplo, si se desea que el teclado tenga encendidos los LEDs "Bloq Núm" y "Bloq Despl", el byte será 00000011. Si se desea, apagar los LEDs mencionados y encender el LED "Bloq Mayús" el byte será 00000100.

Asignación del conjunto de Códigos de Rastreo (0xF0). El controlador debe enviar esta orden y enseguida un dato que especifica el conjunto de códigos de rastreo a usar, este dato puede ser 0x01, 0x02 o 0x03 para seleccionar el conjunto de códigos de rastreo 1, 2 o 3, respectivamente.

APÉNDICE C

Descripción de los códigos de excepción utilizados.

Código	Nombre	Significado
01	Función ilegal	El código de función recibido en la petición no es una acción permitida para el nodo esclavo. Esto puede ser porque el código de función no fue implementado en la unidad seleccionada.
02	Dirección de datos ilegal	La dirección de datos recibida en la petición no es válida para el nodo esclavo. Más específicamente, la combinación de número de referencia y la longitud de transferencia es inválida.
03	Valor de dato ilegal	Un valor contenido en el campo de dato de la petición no es valor permitido para el nodo esclavo. Esto indica una falla en la estructura del residuo de una petición compleja, tal como que la longitud implicada es incorrecta. Esto específicamente no significa que un dato destinado a almacenarse en un registro tiene un valor fuera del esperado del programa de aplicación.

APÉNDICE D

Cálculo de la revisión de redundancia cíclica (CRC)

Los pasos a seguir para obtener la CRC son los siguientes:

1. Se emplea un registro de 16 bits que representa la CRC, éste debe tener inicialmente todos sus bits en 1.
2. Se aplica una operación tipo XOR al primer byte de datos del mensaje con el byte de menor orden del registro CRC.
3. Al resultado se le aplica un corrimiento en la dirección del bit menos significativo (LSB), insertando un bit cero en la posición del bit más significativo (MSB). Cada bit que se extrae del corrimiento se examina:
    a. si fue 1, entonces se realiza la operación XOR con un registro que contiene 0xA001.
    b. si fue 0, no se repite este paso, es decir, se realiza otro corrimiento.

   Este proceso se repite hasta que se realicen 8 corrimientos.
4. Repetir los pasos 2 y 3 pero ahora con el siguiente byte, hasta que se haya procesado todo el mensaje, cuando esto suceda el contenido del registro CRC será el valor CRC calculado.
5. A la CRC calculada se le debe intercambiar el byte más significativo con el byte menos significativo (sólo para el envío de tramas).

La figura A.1 muestra la descripción de la función que obtiene la CRC en VHDL, ésta toma como parámetros un registro que representa el dato al cual se le va aplicar la CRC y la CRC anterior (cuando se realiza por primera vez el cálculo la CRC anterior debe ser 0xFFFF, tal como lo indica el paso 1).

Por otra parte la figura A.2 muestra la función para obtener la CRC en C++ Builder 6, en este caso se pasa como parámetros un arreglo que contiene toda la trama recibida o a transmitir y la cantidad de bytes que la conforman.

```
-- FUNCION QUE CALCULA LA CRC
FUNCTION CRC_loop(Dato: in UNSIGNED; CRC_ant:in UNSIGNED) return UNSIGNED is
    variable LSB:    STD_LOGIC;
    variable CRC_aux:   UNSIGNED(15 downto 0);
     constant POLI: UNSIGNED(15 downto 0):="1010000000000001";
     begin
        CRC_aux:= CRC_ant(15 downto 8) & (CRC_ant(7 downto 0) xor Dato);
             for i in 1 to 8 loop
                  LSB := CRC_aux(0);
                  CRC_aux := '0' & CRC_aux(15 downto 1);
                  IF LSB='1' THEN
                      CRC_aux:= CRC_aux xor POLI;
                  END IF;
               END loop;
         return CRC_aux;
END FUNCTION CRC_loop;
```

Figura A.1. Descripción de la función que obtiene la CRC en VHDL.

```
//---------------------------------------------------------------------------
//----------   FUNCIÓN PARA OBTENER LA CRC
//---------------------------------------------------------------------------
    unsigned short CRC16(unsigned short *, short);  //PROTOTIPO DE LA FUNCIÓN

    unsigned short CRC16( unsigned short *puchMsg, short TAM) {
    unsigned int CRC    = 0xFFFF;      // CRC inicial
    unsigned int P_Value = 0xA001;   // Inicializa valor polinomio
    int n_corr; // Número de Iteraciones
    int LSB;    // Bit Menos Significativo

    for (;TAM>0;TAM--) {        //Realiza operacion XOR y corrimientos
        CRC = CRC ^ *puchMsg++;  // calcula el primer CRC (XOR)
        for (n_corr=1;n_corr<9;n_corr++) {
            LSB = CRC & 0x01; // Obtiene LSB
            CRC = CRC>>1;     // Corrimiento 1 bit a la derecha
            if (LSB)          // Si el LSB fue 1...
                CRC = CRC ^ P_Value; //Realiza otra oeración XOR
        }
    }
  return (CRC);
}
```

Figura A.2. Función que obtiene la CRC en C++Bulider 6.

APÉNDICE E

Asignación de terminales del FPGA

```
net rst_n          loc="p85";      # Reset de Usuario

# Asignacion de terminales para la memoria SDRAM
net clkin          loc="p88";      # reloj principal
net sclkfb         loc="p91";      # reloj de retroalimentacion SDRAM despues de retardos debidos a PCB
net sclk           loc="p129";     # reloj a la SDRAM
net cke            loc="p131";     # clock enable
net cs_n           loc="p132";     # chip-select
net ras_n          loc="p130";     # row address strobe
net cas_n          loc="p126";     # column address strobe
net we_n           loc="p123";     # write enable
net ba<0>          loc="p134";     # Bancos
net ba<1>          loc="p137";
net sAddr<0>       loc="p141";     # Bus de direcciones
net sAddr<1>       loc="p4";
net sAddr<2>       loc="p6";
net sAddr<3>       loc="p10";
net sAddr<4>       loc="p11";
net sAddr<5>       loc="p7";
net sAddr<6>       loc="p5";
net sAddr<7>       loc="p3";
net sAddr<8>       loc="p140";
net sAddr<9>       loc="p138";
net sAddr<10>      loc="p139";
net sAddr<11>      loc="p136";
net sData<0>       loc="p95";      # Bus de datos
net sData<1>       loc="p99";
net sData<2>       loc="p101";
net sData<3>       loc="p103";
net sData<4>       loc="p113";
net sData<5>       loc="p115";
net sData<6>       loc="p117";
net sData<7>       loc="p120";
net sData<8>       loc="p121";
net sData<9>       loc="p118";
net sData<10>      loc="p116";
net sData<11>      loc="p114";
net sData<12>      loc="p112";
net sData<13>      loc="p102";
net sData<14>      loc="p100";
net sData<15>      loc="p96";
net dqmh           loc="p124";
net dqml           loc="p122";
```

```
# Asignacion de terminales para la memoria RAM FLASH
net  oe_n          loc="p43";       # output-enable
net  ce_n          loc="p41";       # chip-enable
net  we_flash      loc="p58";       # write enable
net  rst_flash     loc="p59";       # reset

net  sAddr_flash<17>     loc="p56";       # Bus de direcciones
net  sAddr_flash<16>     loc="p63";
net  sAddr_flash<15>     loc="p64";
net  sAddr_flash<14>     loc="p54";
net  sAddr_flash<13>     loc="p51";
net  sAddr_flash<12>     loc="p65";
net  sAddr_flash<11>     loc="p47";
net  sAddr_flash<10>     loc="p42";
net  sAddr_flash<9>      loc="p48";
net  sAddr_flash<8>      loc="p50";
net  sAddr_flash<7>      loc="p66";
net  sAddr_flash<6>      loc="p76";
net  sAddr_flash<5>      loc="p75";
net  sAddr_flash<4>      loc="p74";
net  sAddr_flash<3>      loc="p27";
net  sAddr_flash<2>      loc="p28";
net  sAddr_flash<1>      loc="p29";
net  sAddr_flash<0>      loc="p40";
net  sData_flash<7>      loc="p67";       # Bus de datos
net  sData_flash<6>      loc="p62";
net  sData_flash<5>      loc="p60";
net  sData_flash<4>      loc="p57";
net  sData_flash<3>      loc="p49";
net  sData_flash<2>      loc="p46";
net  sData_flash<1>      loc="p44";
net  sData_flash<0>      loc="p39";

# Asignacion de terminales para el teclado
net   tec_clk   loc = "p94";                  # Linea de reloj
net   dato_ps  loc = "p93";                   # Linea de datos

# Asignacion de terminales para el Monitor
net sync_hor       loc="p23";       # Sincronizacion horizontal
net sync_vert      loc="p26";       # Sincronizacion vertical
net  RGB<5>        loc="p13";       # Salidas RBG al DAC
net  RGB<4>        loc="p12";
net  RGB<3>        loc="p20";
net  RGB<2>        loc="p19";
net  RGB<1>        loc="p22";
net  RGB<0>        loc="p21";

# Asignacion de terminales de Transmision y Recepcion Serial
net Rx              loc = "p77";       # Línea de recepcion de datos serial
net Tx              loc = "p83";       # Línea de transmision de datos serial
net Transmitiendo   loc = "p84";   # Indicador de transmision de datos
net Recibiendo      loc = "p79";   # Indicador de recepcion de datos
```

APÉNDICE F

Contenido del disco anexo.

El disco anexo cuenta con tres carpetas, éstas son:

1. **VHDLs.** Contiene los siguientes archivos:
   - TECLADO.VHD. Es el módulo controlador de teclado.
   - VGA.VHD. Módulo controlador de video.
   - RX_TX_SERIAL.VHD. Módulo para la recepción y transmisión serial.
   - CTA_3_5.VHD. Módulo de espera el fin de trama.
   - ALMACENA_TRAMA.VHD. Módulo de almacenamiento de la trama.
   - TRANSACCION_MODBUS. Módulo que realiza el proceso de la trama recibida y construye una trama de respuesta.
   - SDRAMCTRL.VHD. Módulo controlador de la SDRAM de la tarjeta.
   - MEMORIA.VHD. Módulo que sincroniza los accesos a las memorias empleadas en el presente trabajo.
   - MODULO_ALTO_NIVEL.VHD. Módulo que realiza la interconexión de los módulos en VHDL.
   - XS_PCKG.VHD. Contiene una función y declaraciones empleadas por el módulo controlador de la SDRAM.
   - RAM_TRAMA.COE. Contenido de la memoria donde se almacena la trama.
   - ROM_CARACTERES.COE. Matrices de puntos de los caracteres que se muestran en el monitor.
   - MVIDETEC.MCS. Contiene la Memoria de Video y de Teclado que se guardará en la RAM tipo Flash.
   - ESCLAVO.EXO. Archivo se almacena en la RAM tipo Flash para que se programe la tarjeta al momento de suministrarle voltaje.

2. **C.** Contiene los siguientes archivos[9], los cuales muestran diversas ventanas:
   - PRINCIPAL.CPP. Ventana principal con las opciones.
   - UCONSULTA.CPP. Consulta de calificaciones de un alumno.
   - UTABLA.CPP. Actualiza la tabla de calificaciones que se muestra en pantalla.
   - MODCAL.CPP. Modifica las calificaciones de los alumnos.
   - MOSTCAL.CPP. Muestra las calificaciones de los alumnos
   - UNUEVO.CPP. Introduce un alumno a la base de datos.
   - UPEDCAL.CPP. Actualiza los nombres de las materias de las carreras.
   - UACERCADE.CPP. Información al usuario.
   - UINSUSO.CPP. Informa al usuario sobre las opciones del programa.

[9] Para cada archivo con extensión .CPP hay un archivo con el mismo nombre y con extension .h.

Además de los siguientes archivos:

- AYUDA.RTF. Contiene la información que se muestra al solicitar ayuda por el usuario.
- ALUMNOS.$$$. Contiene la base de datos con los nombres y calificaciones de los alumnos.
- CARRERAS.$$$. Contiene la base de datos con los nombres de las carreras y materias de la Universidad Tecnológica de la Mixteca.
- MAESTRO.EXE. Programa del nodo maestro.

3. **DOCUMENTOS.** Archivos que contienen este trabajo en WORD, éstos son:
   - PORTADA.DOC
   - ÍNDICE.DOC
   - INTRODUCCIÓN.DOC
   - CAPÍTULO 1 MARCO TEÓRICO.DOC
   - CAPÍTULO 2 MÓDULOS DE INTERFAZ.DOC
   - CAPÍTULO 3 MÓDULOS MODBUS.DOC
   - CAPÍTULO 4 INSTANCIACIÓN DE COMPONENTES.DOC
   - CAPÍTULO 5 NODO MAESTRO.DOC
   - CAPÍTULO 6 RESULTADOS Y CONCLUSIONES.DOC
   - APÉNDICES.DOC

LISTA DE FIGURAS

LISTA DE TABLAS

BIBLIOGRAFÍA

[1] Análisis y Diseño de Circuitos Lógicos Digitales; Victor P. Nelson, H. Troy Nagle, Bill D. Carroll, J David Irwin; Editorial Prentice Hall, México; 1996.

[2] Dispositivos Lógicos Programables y sus Aplicaciones; Enrique Mandado, L. Jacobo Álvarez, Maria Dolores Valdés; Thomson Editores, España; 2002.

[3] VHDL Lenguaje para síntesis y modelado de circuitos; Fernando Pardo, José A. Boluda; Editorial Alfaomega, México; 2000.

[4] Essential VHDL, RTL Síntesis Done Right; Sundar Rajan; Editorial Prentice Hall, USA; 1998.

[5] Microprocessors and Interfacing: Programing and Hardware; Douglas V. Hall; Editorial McGraw-Hill, Edición Internacional; Segunda Edición; 1992.

[6] Lenguaje Ensamblador y Programación para PC IBM® y Compatibles; Abel Peter; Editorial Prentice Hall Hispanoamericana; Tercera Edición; 1996.

[7] Sistemas de Comunicaciones Electrónicas; Wayne Tomasi; Editorial Prentice Hall, México; Segunda Edición; 1996.

[8] Comunicaciones y Redes de computadores; William Stallings; Editorial Prentice Hall, España; Sexta Edición; 2000.

URL

[URL1] Architecture of FPGAs and CPLDs: A Tutorial.
http//www.eecg.toronto.edu/~jayar/pubs/brown/survey.pdf; 2003.

[URL2] Programmable Logic Design Quick Start Hand Book.
http://www-ee.ccny.cuny.edu/www/web/xchen/EE598.67syllabus.html; 2002.

[URL3] Directorio de recursos VHDL.
http://www.terra.es/personal/zyryab/listado.htm; 2003.

[URL4] What are CPLDs and FPGAs?.
http://www.xess.com/fpgatut.htm; 2002.

[URL5] Pragmatic Logic Design with Xilinx Foundation 2.1i.
http://www.xess.com/pragmatic-2_1.htm; 2002.

[URL6] XSA Board V1.0 User Manual.
http://www.xess.com/manuals/xsa-manual-v1_2.pdf; 2002.

[URL7] http://www.xess.com/ho01001.html; 2004.

[URL8] Spartan-II 2.5V FPGA Family:Complete Data Sheet.
http://direct.xilinx.com/bvdocs/publications/ds001.pdf; 2003.

[URL9] Using Delay-Locked Loops in Spartan-II FPGAs.
http://direct.xilinx.com/bvdocs/appnotes/xapp174.pdf; 2003.

[URL10] The AT-PS/2 Keyboard Interface.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/keyboard/
atkeyboard.htm; 2002.

[URL11] PS/2 Mouse/Keyboard Protocol.
http://panda.cs.ndsu.nodak.edu/~achapwes/PICmicro/PS2/ps2.htm; 2002.

[URL12] Conceptos Básicos de Vídeo.
http://spanish.doom9.org/video-basics.htm; 2002.

[URL13] MODBUS Application Protocol Specification V1.1.
http://www.modbus.org; 2003.

[URL14] MODBUS over Serial Line Specification & Implementation guide V1.0.
http://www.modbus.org; 2003.

[URL15] Explanation of Maxim RS-485 Features.
http://pdfserv.maxim-ic.com/en/ds/MAX1487E-MAX491E.pdf; 2003.

[URL16] Example Applications: RS-485 Two-Wire.
http://www.robustdc.com/?libsect=appnotes; 2003.

[URL17] Low-Power, Slew-Rate-Limited RS-485/RS-422 Transceivers.
http://pdfserv.maxim-ic.com/en/ds/MAX1487E-MAX491E.pdf; 2003.

[URL18] XSA SDRAM Controller.
http://www.xess.com/fpgatut.htm/appnotes/an-092601-sdramcntl.pdf; 2002.

[URL19] HOW TO USE SDRAM.
http://www.elpida.com/pdfs/E0123N40.pdf; 2002.

[URL20] ATMEL 2-megabit (256K x 8) 5-volt Only Flash Memory AT49F002.
http://www.atmel.com/dyn/resources/prod_documents/DOC1017.pdf; 2003.