Implementación de MODBUS en FPGA mediante VHDL – Capa de Enlace –

Olmedo Sergio, Guanuco Luis, Panozzo Zenere Jonatan, Rubio Agustin
Centro Universitario de Desarrollo en Automación y Robótica "CUDAR"
Universidad Tecnológica Nacional - FRC
Córdoba, Argentina
solmedo@scdt.frc.utn.edu.ar, {lguanuco, 49190, 49286}@electronica.frc.utn.edu.ar

***Abstract*—La descripción de hardware, mediante la programación en VHDL, permite una amplia versatilidad en el diseño de circuitos digitales. En este artículo se presenta una descripción sobre la realización de una comunicación entre dispositivos lógicos programables, según el protocolo MODBUS. Este estándar de comunicación, de amplia aceptación, define protocolos para las capas de "Aplicación", "Enlace" y "Física". En este documento se aborda el desarrollo del mismo en "Capa de Enlace", y de manera resumida la forma en que esta interactúa con las otras dos capas. Esto se lleva a cabo mediante la descripción de los principales bloques, la síntesis, simulación y finalmente implementación en dispositivos FPGA.**

Keywords-MODBUS; Data-Link; FPGA; VHDL

I. INTRODUCCIÓN

En el desarrollo de cualquier protocolo de comunicación se deben considerar niveles de abstracción para el tratado de la información como así también diferentes formas de implementación tanto *hardware* como *software*. Para definir éstas pautas de diseño se considera el modelo OSI.

El modelo OSI (*Open System Interconnection*) es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones desarrollado por la Organización Internacional para la Estandarización [1]. Este permite al desarrollador seguir una determinada estructura para el manejo de la información en dicha red, Fig. 1.

Cada uno de los niveles de este modelo se regirá de acuerdo a las especificaciones del protocolo. Este modelo logra imponer un nivel de abstracción en el cual la comunicación es entre capas del mismo nivel de dos o más dispositivos. Sin embargo, la comunicación existe solo entre capas adyacentes de un mismo dispositivo, conectándose a otro únicamente a través de las capas físicas.

A. Capa de Enlace

MODBUS [2] [3] define un protocolo en esta capa para la comunicación serie entre un único dispositivo Maestro y entre uno a 247 Esclavos. En el caso de haber un único Esclavo, la comunicación se denomina "punto a punto" y si existe más de un Esclavo, la comunicación es "multipunto".

MODBUS define protocolo, en Capa de Enlace, para diferentes modos, como ser: "Maestro/Esclavo", "*Ethernet* II/802.3", "MODBUS+/HDLC", y otros. En este caso se utiliza "Maestro/Esclavo".

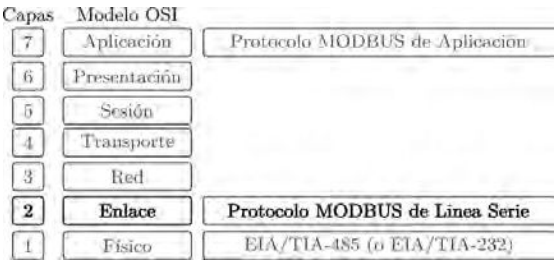

Figure 1. Modelo OSI con sus diferentes niveles.

Una comunicación siempre la inicia un Maestro, por lo que un Esclavo sólo transmite información luego de una petición; de lo cual se deduce que no es posible la comunicación directa entre Esclavos. Cada uno de estos dispositivos, tiene una dirección específica que los distingue.

El dispositivo Maestro puede transmitir datos en dos modos diferentes: *Unicast* o *Broadcast*. El primero, está dado por una petición del Maestro a un Esclavo específico y siempre la respuesta de este. El segundo es una transmisión del Maestro hacia todos los Esclavos al mismo tiempo, no habiendo respuesta alguna de ninguno de ellos.

MODBUS permite la codificación de la información en la red en dos formas diferentes, RTU (*Remote Terminal Unit*) y ASCII (*American Standard Code for Information Interchange*) [3].

RTU, los datos se presentan en bits consecutivos formando tramas de datos, cuyo inicio y fin son indicados por intervalos de tiempo.

ASCII, la información se encuentra codificada en caracteres ASCII. La trama de datos comienza y termina con caracteres definidos.

Los tiempos de transmisión y recepción de una trama en cada uno de éstos modos de codificación difieren en gran medida. En modo ASCII los datos deben ser convertidos en su correspondiente caracter además de ser ponderados en formato hexadecimal. Por ejemplo, el byte 0x5B es codificado como dos caracteres: 0x35 y 0x42 (0x35 = "5", y 0x42 = "B" en ASCII) [3]. Por el contrario, en el modo RTU la información se encuentra en forma de bits consecutivos, permitiendo que para

un mismo tiempo, haya un mayor flujo de información por la red que en el modo ASCII.

B. Codificación en modo ASCII

Si bien el modo RTU debe implementarse en todo los dispositivos, se elige en primera instancia y para el presente desarrollo el modo ASCII, debido a la mejor legibilidad de la información. En este modo se puede apreciar la trama circulante por el *bus*, conectando a él un dispositivo con las capacidades de interpretar caracteres ASCII. Esta es una característica fundamental si se quiere realizar un análisis en cualquier punto de una red donde se encuentra aplicado MODBUS.

La codificación en modo ASCII cuenta con una trama limitada por un caracter de comienzo ":" y dos de fin "CR (*Charriage Return*) – LF (*Line Feed*)". El mensaje se encuentra dentro de éstos caracteres distribuido como se observa en la Fig. 2. Los cuatro campos que forman el mensaje son:

Dirección; del dispositivo esclavo que está actuando en la comunicación.

Código de Función; códigos preestablecidos por MODBUS que establecen las operaciones que debe llevar a cabo el esclavo.

Datos; es la información.

CRC (*Cyclic Redundancy Check*) /LRC (*Longitudinal Redundancy Check*); campo que sirve para la detección de errores, y no para la corrección de estos.

Mensaje del MODBUS

comienzo	dirección	función	datos	LRC	fin
1 caracter ":"	2 caracteres	2 caracteres	0 hasta 2x252 caracter(es)	2 caracteres	2 caracteres "CR,LF"

Figure 2. Trama de MODBUS en modo ASCII.

II. DISEÑO

La implementación de un protocolo MODBUS en FPGAs requiere un diseño en algún lenguaje de descripción de hardware, basado en gran medida, en el desarrollo de máquinas de estados finitas.

La generación de una trama comienza con el envío de un caracter que define el principio de la misma. En forma consecutiva se transmiten los campos de dirección, función, datos, chequeo de error LRC y para terminar los caracteres de fin de trama. De forma semejante se plantea para la recepción de la trama. En forma general se definen los estados de codificación/decodificación de la trama en la Fig. 3.

Como bloques específicos de mayor relevancia en el diseño, se considera los de Recepción y Transmisión, los que se definirán como máquinas de estados, a nivel de componentes, dentro de la descripción principal en VHDL.

Las máquinas de estados se clasifican en dos tipos: "*Moore*" y "*Mealy*" [4]. Ambas se diferencian por la dependencia o no, de las salidas con respecto al estado de las entradas.

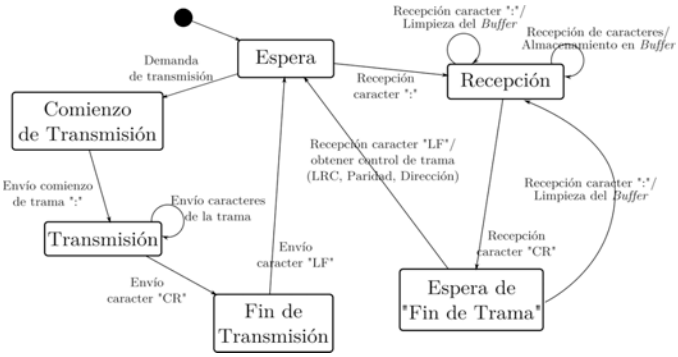

Figure 3. Diagrama de estados en transmisión y recepción.

En virtud de los requerimientos necesarios para la capa de enlace del MODBUS, se opta por la implementación de máquinas de estados tipo *Mealy*.

A. Bloque de RAM

La información que contiene la trama del MODBUS debe ser almacenada en registros para su tratado en los diferentes niveles. En este sentido, este bloque funciona como puente entre las capas de "Enlace" y "Aplicación". La primera, guarda en la RAM los datos recibidos en el mensaje, preparando el servicio de la capa de aplicación. Esta toma los datos desde la RAM, los procesa y escribe en ella la información a transmitir.

Se logra realizar la descripción de *hardware* de un bloque de memoria RAM mediante la sintaxis VHDL. Existen dos posibilidades de llevar dicha descripción al dispositivo lógico, bloques lógicos reconfigurables o dispositivos primitivos. La elección de uno u otro no se encuentra, en general, al alcance del diseñador, sino que será el sintetizador quién infiera en su elección.

Los bloques de RAM embebidos en FPGAs, llamados también bloques de RAM primitivos, se encuentran físicamente en el chip [6]; compuestos de entradas/salidas, bus de direccionamiento y señales de control. La limitación en su utilización, es que no se cuenta con un modelo descriptivo de los mismos. Restringe el diseño, al ser de un tamaño ya determinado por el dispositivo FPGA en donde se encuentra, en definitiva, hace dependiente la descripción al hardware a utilizar.

En el bloque de RAM descriptivo se puede llevar a cabo análisis de tiempo y reducir la cantidad de bloques lógicos en función de la necesidad de la implementación. Considerando un bloque de RAM primitiva instanciado mediante el uso de librería o un bloque de RAM descriptivo, se opta en el presente trabajo por este último, en base a lo explicado anteriormente. Sin embargo, el diseño global ocupa mayores recursos dado que las RAM primitivas están igualmente incorporadas y disponibles en el chip.

B. Transmisor y Receptor

El Transmisor funcionalmente debe generar la trama a ser enviada, esto, tanto en el Maestro como en los Esclavos. Se diseña una máquina de estados, pendiente del proceso de

escritura del bloque de RAM, llevada a cabo por la capa de aplicación.

La máquina de estados realiza las lecturas sucesivas desde el bloque de RAM hasta enviar uno a uno los caracteres, respetando los marcadores de comienzo y fin de trama.

El Receptor, al igual que el Transmisor, utiliza nuevamente una máquina de estados, que deberá cumplir con las especificaciones del modo de codificación. En este caso se cuenta con la información en forma serial recibida por la capa "Física". Los datos son almacenados en el bloque de RAM, momento en el que el bloque de recepción posee el control absoluto de escritura en la memoria.

Por lo expuesto, resulta necesaria la presencia de un control de accesibilidad del bloque de RAM, dado que varios componentes precisan de la escritura y/o lectura de dicho bloque.

Tanto el Transmisor como el Receptor deben chequear la presencia de un error en la trama, lo que se representa por los *bytes* del LRC.

Como se presentó anteriormente, el cálculo del LRC resulta sencillo en su formulación. Esta compuesto el complemento a dos de la sumatoria de todos los elementos de la trama. En el diseño comportamental se traducirá a unas simples líneas de código que en forma lógica será compuesta por un sumador digital y simples compuertas. Aquí se puede ver la versatilidad del lenguaje y la forma abstracta del diseño, donde se utiliza recursos primitivos de la FPGA con una descripción básica.

C. UART

MODBUS define para las capas 1 y 2 del modelo OSI, el "Protocolo MODBUS de Línea Serial" [3]. Esto implica la utilización de una UART (*Universal Asynchronous Receiver Transmitter*) para poder transmitir y recibir los datos en forma serie.

La UART constituye entonces la conexión de la capa de "Enlace" con la capa "Física". Esta última puede ser RS232 o el estándar RS485 adoptado en el presente desarrollo.

Este bloque se realiza al igual que los demás de manera descriptiva en VHDL, y en forma general presenta el dato recibido en forma serial, como salida en paralelo. De forma análoga, recibe el dato a transmitir en paralelo y envía los bits de información en forma serie atendiendo las configuraciones de velocidad elegidas, y las condiciones preestablecidas por el protocolo MODBUS sobre la conformación de la palabra a enviar: *bits* de comienzo, datos, paridad y parada [3].

III. SÍNTESIS E IMPLEMENTACIÓN

La implementación se realiza en una FPGA Xilinx Spartan 2E XC2S200E [6]. La síntesis se realiza con el XST (*Xilinx® Synthesis Technology*) [7], herramienta que forma parte del paquete ISE Xilinx versión 6.0 [8] disponible en el centro de investigación donde se lleva a cabo el desarrollo.

La FPGA cuenta con una gran cantidad de recursos físicos, los principales se detallan a continuación:

- Bloques de entradas y salidas.
- Bloque lógico configurable.
- Bloques de RAM.
- Distribución de Clock: DDL (*Delay-Locked Loop*).
- *Boundary Scan*.

Con las pautas de diseño ya presentadas, como así también la identificación de los distintos bloques que componen nuestra descripción, se presenta el resultado de la síntesis, TABLA I.

TABLA I. RESUMEN DE UTILIZACIÓN DE RECURSOS

Dispositivo FPGA: 2S200EPQ208-6Q			
Recurso	**Utilizado**	**Disponible**	**Porcentaje**
Slices	349	2352	14%
Flip Flops	296	4704	6%
LUTs	496	4704	10%
IOBs	42	146	28%
GCLKs	1	4	25%

De la TABLA I se aprecia los escasos recursos utilizados, ya que se cuenta con un dispositivo con gran número de CLBs (*Configurable Logic Block*). Igualmente es de suma importancia la simulación, verificación y posterior simplificación de la descripción, para lograr un mejor rendimiento de los recursos en vista de su implementación en diferentes dispositivos lógicos.

La utilización de un único reloj para el sincronismo de los CLBs resulta ser más flexible en el diseño que disponer de varios clocks externos conectados a la FGPA. Sin embargo, debe tenerse presente que esto se logra con el correspondiente consumo de recursos físicos, ya que un divisor de clock, implementado con bloques lógicos, se sintetiza como un contador lógico.

RTL (*Register Transfer Level*) permite la representación gráfica del diseño descrito en VHDL. En la Fig. 4 se presente la entidad de la arquitectura del proyecto.

En la presentación de las características del protocolo MODBUS se ha señalado que le mismo especifica las capas de "Aplicación", "Enlace" y "Física". Se deja en claro que se ha desarrollado la capa de Enlace en función de éstas especificaciones. Ahora, la implementación física en FPGA se realizó en un prototipo de desarrollo, en el cuál se adaptó una línea serie mediante RS232 conectando entre sí dos FPGA. Uno de ellos implementados como Maestro y el otro como Esclavo.

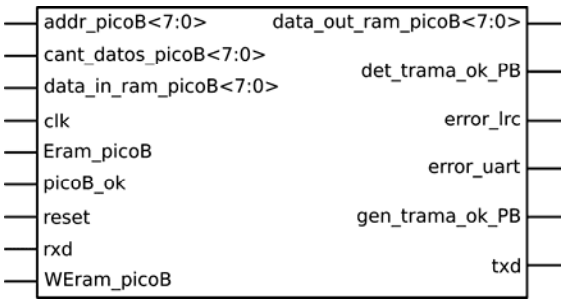

Figure 4. Entidad de la implementación de MODBUS en FPGA.

La entidad de la capa de Enlace descripta presenta puertos de control para comunicarse con la capa de Aplicación. Debido a la ausencia de la implementación de la capa de Aplicación, se ha reemplazado él mismo por llaves que hacen de control en la interacción con la capa de Enlace.

La implementación de la capa de Aplicación forma parte de un proyecto global que englobaría la implementación completa del protocolo MODBUS. Debido a la complejidad que esto presenta, se ha considerado como primera opción el uso de microprocesadores embebidos en FPGA. Para ser más preciso, *cores* tales como PicoBlaze o MicroBlaze. Mediante el empleo de compiladores destinados a éstos microcontroladores se puede realizar procesos secuenciales escritos en *assembler* al igual que cualquier microprocesador. Desde luego, se debe tener en cuenta las limitaciones en uso de la memoria de programa y demás información proporcionada por el fabricante.

IV. SIMULACIÓN

La simulación resulta fundamental en el proceso de síntesis e implementación. La estructura del proceso de simulación es acorde al esquema de la Fig. 5. De esta manera se crea un lazo que permite llegar al correcto funcionamiento del sistema. La herramienta de *software* con el que se llevó a cabo éstas simulaciones es ModelSim SE versión 6.0a.

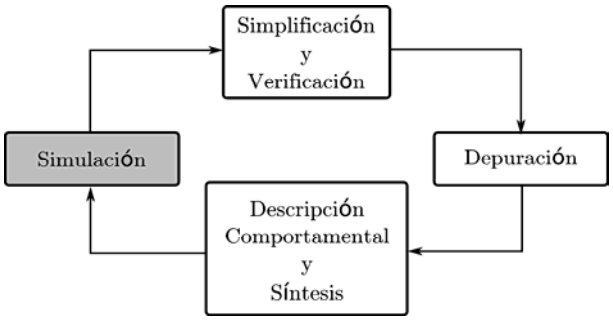

Figure 5. Proceso de validación del diseño digital

En función de las especificaciones de la capa de "Enlace" del protocolo MODBUS, se presenta un caso posible de comunicación tanto para la transmisión como la recepción de datos.

El proceso de simulación se basa en la instanciación de la descripción a simular en otra arquitectura, llamada *testbench*, donde se permite el uso de librerías de simulación que no son posibles sintetizar en un dispositivo lógico. El ejemplo más significativo es la descripción de periodos de tiempo, clave en el presente proyecto.

Dentro de la arquitectura del *testbench* se inyectan señales de entrada que sean representativas de una trama MODBUS tal y como se describió en la introducción de éste trabajo. Luego que se obtienen los datos decodificados de la trama, la capa de Enlace avisa a la capa superior, Aplicación, que los datos se encuentran listos para ser procesados. La capa de Aplicación realiza la operación correspondiente, lo que se traduce en la simulación como un determinado periodo de tiempo estipulado como el tiempo de respuesta. Para finalizar, se envían los datos escritos en el bloque de RAM por la capa de Aplicación. Se resume éste proceso en la Fig. 6.

La simulación no sólo ofrece información útil para corregir problema en la síntesis, sino que además permite validar la trama. En la Fig. 7 se observa la decodificación de una trama de recepción acorde lo dicho en el párrafo anterior, en la Fig. 8 se ha re-transmitido la trama recibid, almacenada en la RAM, lo que demuestra el correcto funcionamiento de la descripción.

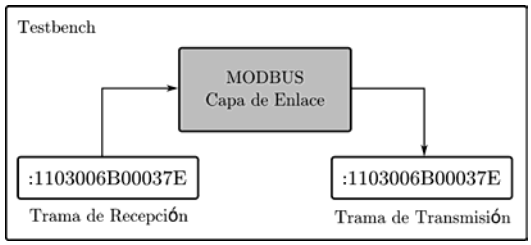

Figure 6. Datos de entrada y salida en simulación.

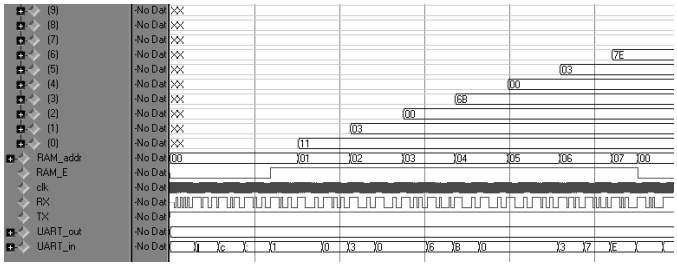

Figure 7. Resultado de simulación de una trama de recepción en Capa de Enlace del MODBUS.

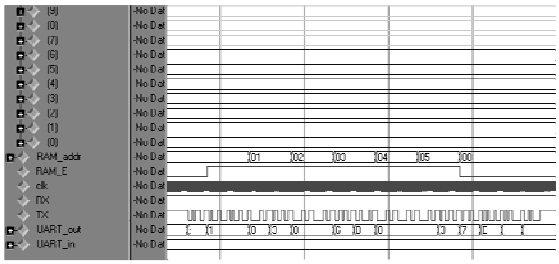

Figure 8. Resultado de simulación de una trama de transmisión en Capa de Enlace del MODBUS.

V. CONCLUSIÓN

En función a las especificaciones del protocolo MODBUS, se ha logrado un desarrollo totalmente descriptivo en el lenguaje VHDL. Lo que permite mejorar el rendimiento y eficiencia en su implementación como así también diferentes análisis funcionales mediante el uso de *software*. De ésta forma, se puede dar un carácter didáctico al presente trabajo. A continuación se resume éstos conceptos.

A. Implementación mediante VHDL (concurrencia)

La descripción de hardware mediante VHDL permite la flexibilidad en el diseño de sistemas digitales, dado que el mismo se realiza independientemente del dispositivo a utilizar,

por lo que se logra portabilidad en la implementación sobre PLDs. La concurrencia otorga un mejor aprovechamiento del tiempo, que se traduce en mayor velocidad de operación, en desmedro de una utilización de recursos también mayor.

B. Recursos de hardware de la FPGA

Con el avance tecnológico, los PLDs logran alcanzar velocidades de hasta los Giga Hertz, característica funcional de los bloques lógicos que componen el dispositivo.

C. Implementación de Dispositivos Lógicos Programables vs. Microcontroladores

En el proceso de investigación a cerca de la implementación de MODBUS en sistemas embebidos, presenta una preferencia en la utilización de microcontroladores para llevar adelante su desarrollo. El avance tecnológico de los microcontroladores, su evolución en nuevas arquitecturas y las herramientas de *software* han incrementado ésta tendencia.

Resulta complejo realizar una comparación directa entre la implementación de MODBUS en FPGA y microcontroladores, ya que se presentan muchas variables que caracterizan cada una de éstas tecnologías. En forma objetiva se pueden enunciar ítems que se han considerado en la comparación:

- Nivel de abstracción en la programación.
- Portabilidad de código.
- Herramienta de software.
- Consumo de recursos físicos.

Agregado a lo anteriormente expuesto, el presente trabajo posee un enfoque didáctico en la implementación de VHDL con dispositivos lógicos programables. Con lo que se incentiva más aún el desarrollo, además de su continuo avance tanto en la mejora de su descripción como el diseño del *hardware*.

AGRADECIMIENTOS

El desarrollo de la implementación de MODBUS en FPGA forma parte de los proyectos desarrollados en El CUDAR "Centro Universitario de Desarrollo en Automación y Robótica". Se agradece a los Directivos y Miembros.

REFERENCIAS

[1] MODBUS-IDA.ORG, "Modelo OSI". http://es.wikipedia.org/. 2010.

[2] MODBUS-IDA.ORG, "*MODBUS application protocol specification*", V1.1b. http://www.MODBUS.org, 2010.

[3] MODBUS-IDA.ORG, "*MODBUS over serial line specification and implementation guide*", V1.02. http://www.MODBUS.org, 2010.

[4] K. Kuusilinna, V. Lahtinen, T. Hämäläinen, J.Saarinen, "*Finite state machine encoding for VHDL synthesis*", IEEE Proc.-Comput. Digit. Tech, Vol. 148, No. 1, Enero 2001.

[5] A. Iborra y J. Suardiaz, "Diseño de Sistemas Electrónicos-DB4", Diseño Basado en Máquinas de Estado Finitas, Uni. 8. Mayo 2003.

[6] Xilinx® Inc., "*Spartan-IIE 1.8V FPGA Family: Functional Description*", v2.1, Product Specification. Julio 2003.

[7] ©2002-2008 Xilinx, "ISE 10.1 *Quick Start Tutorial*". http://www.xilinx.com/. Agosto 2010.

[8] Xilinx® Inc. "ISE WebPACK *Design Software*". http://www.xilinx.com/. Agosto 2010.

[9] J. Jiménez, E. Fernández, J. Martin, U. Bidarte,A. Zuloaga. "*Simulation environment to verify industrial communication circuits*". University of the Basque Country, Department of Electronics and Telecommunications, 2002.