Interfaz programable de comunicación serie asincrónica en FPGA

Alexander Suárez Léon1, Danelia Matos Molina2, Roger Rivero Labrada3, Rubén D. López Noa4, Berta Pallerols Mir5

1 Universidad de Oriente, aasl@fie.uo.edu.cu

2 Centro Povincial de Electromedicina Santiago de Cuba

3 Universidad de Oriente, roger@fie.uo.edu.cu

4 Universidad de Oriente, lnoa@fie.uo.edu.cu

5 Universidad de Oriente, bertapm@fie.uo.edu.cu

RESUMEN

El presente artículo muestra el diseño, implementación y prueba de una interfaz programable de comunicación serie asincrónica en un FPGA. El diseño está orientado a emular de forma parcial el funcionamiento de los dispositivos de recepción – transmisión asincrónica universal (UART) comerciales del fabricante Intel®. Se muestra la estructura interna de los bloques funcionales fundamentales y los resultados de la simulación funcional. Finalmente algunas aplicaciones y los principales aspectos distintivos del diseño son discutidos.

Palabras claves: UART, FPGA, Comunicaciones Asincrónicas.

ABSTRACT

This paper shows the design, implementation and test of an asynchronous serial communications programmable interface on FPGA device. The design is oriented to partially emulate the behavioural of the Intel's universal asynchronous receiver transmitter (UART) available in the market. The internal functional blocks structure and the functional simulation results are showed. Finally some applications and essentially subjects of the design are argued both.

KeyWords: UART, FPGA, Asynchronous Communications

Asynchronous serial communication programmable interface on FPGA

INTRODUCCIÓN

De los fundamentos de la transmisión de información es conocido que entre el transmisor y el receptor debe existir una sincronización para que la transmisión de la información se realice de manera exitosa. Esto es así debido a que en la transmisión de datos la información pasa por varias etapas como por ejemplo la codificación y decodificación en ambos extremos del canal lo cual exige que exista sincronismo entre la fuente y el destino.

En el caso de una transmisión digital este sincronismo puede ser de dos tipos: sincronismo de bit o sincronismo de caracter. Dependiendo de la forma en que se logra este la transmisión se denominará sincrónica o asincrónica.

El protocolo de transmisión asincrónica se desarrolló a comienzos de la historia de las telecomunicaciones. Se puso muy de moda a partir de la invención de los transmisores de telex o telegramas que se usaron para enviar telegramas alrededor del mundo.

En general el formato asincrónico requiere que delante de cada caracter se transmita un pulso de arranque y detrás del dato uno o más pulsos de parada. El valor del pulso de arranque es un cero (0) lógico de duración igual a un (1) bit y el valor de los pulsos de parada es un uno (1) lógico y de duración entre uno (1) y dos (2) bits. La sincronización de bits se logra a través del bit de arranque y la de caracter a través del conteo de los bits siguientes al bit de arranque.

En este tipo de sincronismo no todos los bits transmitidos son de datos debido a que hay bits de control (arranque, parada y opcionalmente paridad) por lo que no se puede aprovechar la capacidad total del canal.

Este formato de transmisión es apropiado para transmitir a velocidades lentas, por debajo de los 32000 baudios.

Como ventajas de este tipo de transmisión se pueden citar:

1. El formato asincrónico es ampliamente utilizado en terminales de baja velocidad donde lograr gran eficiencia en la utilización de la capacidad del canal no sea tan importante.
2. No se necesita excesiva estabilidad en el tiempo de duración de cada pulso.

3. La transmisión de datos se verifica con mayor rapidez debido a lo sencillo del sincronismo.
4. Este formato es muy fácil de implementar desde el punto de vista de la programación con respecto a otros formatos.

Las desventajas que sobresalen son:

1. Ineficiencia en la utilización de la capacidad del canal.
2. La distorsión o alteración del pulso de arranque causa errores en la recepción del dato.

El objetivo de este documento es mostrar el diseño e implementación en FPGA de una sencilla unidad de comunicación serie asincrónica para su uso con procesadores empotrados.

METODOLOGÍA

El protocolo de comunicación serie-asincrónica impone el siguiente formato:

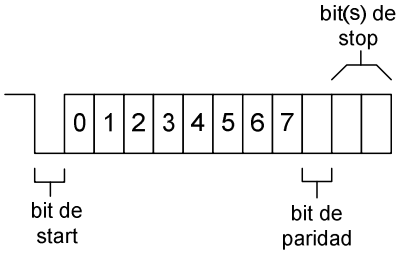

Figura 1

Protocolo serie asincrónico.

Como se aprecia los bits de datos y los de control viajan juntos por un solo conductor, la duración de cada bit es determinada por la velocidad de transmisión. En este protocolo se acuerda que cuando no haya datos en la línea se pondrá un nivel alto o uno (1) lógico en el canal. Cuando esto sucede el receptor se pone en espera del bit de arranque y a partir de éste captura el resto de la trama.

Los bits de datos pueden variar entre 5 y 8 como se muestra en la Tabla I. Los bits de parada varían de acuerdo con la cantidad de bits de datos que tenga la trama.

Tabla I: Relación cantidad de bits de datos / cantidad de bits de stop.

Bits de Datos	Bits de Stop
5 bits	1,5 bits
6, 7 y 8 bits	2 bits
8 bits	1 bit

A estos bits se puede sumar opcionalmente el bit de paridad, que permite implementar un rudimentario chequeo de errores en la transmisión. Este bit es configurable y la paridad puede ser fija, par, impar o prescindir de ella.

Dispositivos que implementen esta función existen muchos en el mercado, por ejemplo comercialmente están disponibles dos integrados muy conocidos como el UART Intel™ 8250 y el UART Intel™ 16550 que es el empleado en las PC actuales que disponen de interfaz RS232C.

Estos componentes son compatibles entre sí y permiten una amplia gama de velocidades de transmisión, destacando además el polimorfismo de las tramas y otras prestaciones como el soporte para interrupciones y **buffers** FIFO (16550).

Es por ello que son precisamente estos dispositivos los que han servido de modelo para el diseño que aquí se presenta.

Elementos generales

La idea básica del diseño está sustentada en el propósito de diseñar un circuito para hardware programable (léase FPGA) que dentro de lo admisible tenga un comportamiento similar a un 8250 clásico pero mucho más sencillo y menos general.

El circuito en forma de bloque funcional se muestra en la Figura 2. De forma general cuenta con 26 patillas divididas en 9 entradas, 1 salida y 16 líneas bidireccionales (el bus de datos).

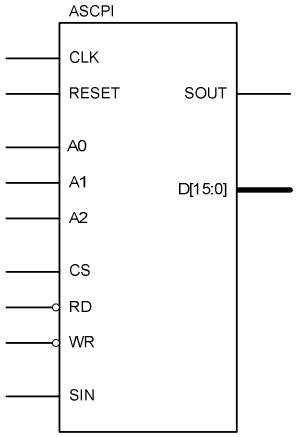

Figura 2

Bloque funcional de la interfaz.

Las líneas del bus de datos D[15:0] se han colocado convenientemente a la derecha.

Estructura interna

Internamente la unidad está compuesta por unidades funcionales más pequeñas como se muestra en la Figura 3.

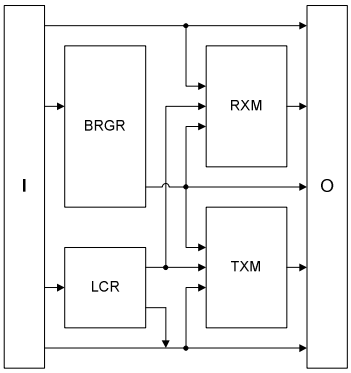

Figura 3

Estructura interna de la interfaz.

El dispositivo está compuesto por las unidades funcionales que a continuación se enuncian:

1. Unidades de entrada/salida. Bloques I/O en el diagrama.
2. Registro de almacenamiento del divisor para el generador de baudios. Bloque BRGR.
3. Registro de Control de Línea. LCR es análogo al que aparece en el 8250.
4. Módulo de recepción. RXM.
5. Módulo de transmisión. TXM.

Las unidades de entrada/salida tienen la función de establecer las entradas y constituyen la interfaz entre el dispositivo que solicita la transmisión serie y el transmisor en sí.

Los registros BRGR y LCR son dos registros de 16 y 8 bits respectivamente con la particularidad de que BRGR tiene un valor de reinicio por defecto distinto de cero.

El LCR a diferencia de su análogo en el 8250 sólo es configurable en los bits 7, 5, 4 y 3. Los tres últimos corresponden a selección de la paridad; y el más significativo, que en el 8250 es **DLAB**, aquí es el equivalente al bit 4 del Modem Control Register (MCR), que permite hacer un diagnóstico del dispositivo colocándole en un modo en que internamente se conecta SOUT a SIN permitiendo que un caracter enviado por SOUT sea recibido en SIN.

Los elementos más importantes de esta estructura son el módulo de transmisión que como su nombre lo indica es el elemento encargado de garantizar la recepción de los datos que entran por la patilla SIN; y el módulo de transmisión que se encarga de transmitir los datos por la patilla SOUT.

Se verán a continuación más detalles de cada uno de estos dispositivos.

Módulo de recepción

Internamente el módulo de recepción está estructurado como se muestra en la Figura 4.

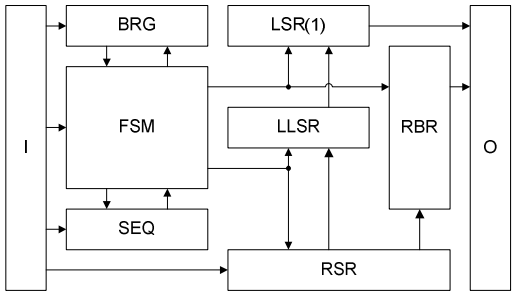

Figura 4

Estructura interna del módulo de recepción (RXM).

Está formado por un bloque de entrada/salida I/O, un generador de baudios (BRG), una máquina de estado finito (FSM), un contador de bits, (SEQ), un registro de desplazamiento de recepción (RSR), un buffer de recepción, una lógica de chequeo de la recepción (LLSR) y parte del registro de estado de línea, LSR.

La tarea del BRG es generar pulsos con la frecuencia indicada por la velocidad de transmisión y entregarlos a la máquina de estado para que genere las órdenes que deben ejecutar el resto de los dispositivos en el próximo ciclo.

La máquina de estado finito (FSM) actúa como unidad de control (UC, de aquí en adelante se llamará indistintamente como máquina de estado o como unidad de control) de todo el sistema controlando los recursos de recepción de datos auxiliada por el generador de baudios y el contador de bits.[1] – [4].

El diagrama de estados de la máquina se muestra en la Figura 5.

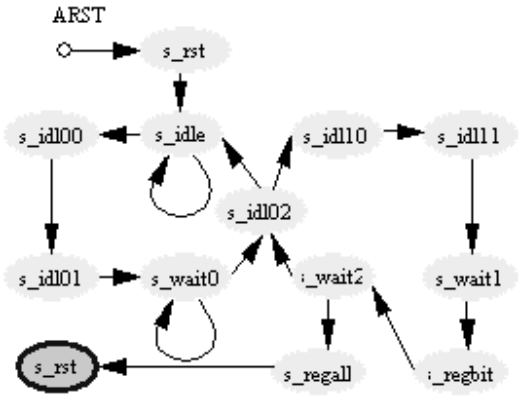

Figura 5

Diagrama de estados de la unidad de control de recepción.

El contador de bits, SEQ, genera un nivel bajo en su salida cuando se ha alcanzado el último bit o bit de stop de la trama, permitiendo así que la unidad de control conozca que se ha recibido el último bit de la trama.

Es necesario para que el contador de bits funcione correctamente si la paridad está habilitada.

El registro de desplazamiento de recepción (RSR) es un registro de desplazamiento a la derecha de 10 bits (el bit de START no cuenta) cuyo propósito es recibir cada uno de los bits que componen la trama de datos

El registro buffer de recepción RBR se encarga de almacenar el dato recibido de 8 bits.

La lógica de chequeo de recepción (LLSR) está diseñada para determinar las condiciones de error (paridad, trama, etc) que serán registradas en el registro de estado de línea de recepción, incluye una puerta ***XOR*** para la obtención de la paridad de los bits recibidos.

El registro de estado de línea (LSR) es un subconjunto de las señales de este registro en el 8250 que pertenecen a la operación de recepción. Estas son:

DR: Data Ready. (BIT 0)

OE: Overrun Error. (BIT 1)

PE: Parity Error. (BIT 2)

FE: Framing Error. (BIT 3)

El BIT 4, **BI**: Break Interrupt no está soportado.

Módulo de Transmisión

El módulo de transmisión (TXM) como ya se dijo es el encargado de transmitir los datos por la patilla SOUT. En la Figura 6, se muestra la estructura interna de este módulo.
Este módulo cuenta con algunos de los elementos del módulo de recepción, como FSM, BRG y SEQ, que en el caso de los dos últimos operan exactamente igual a como lo hacen en el módulo de recepción, no siendo el caso de la máquina de estado que es también la unidad de control, pero en este caso de transmisión. Ver Figura 7.

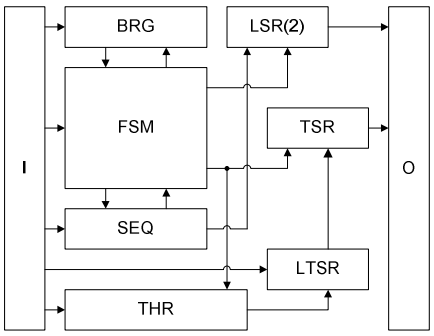

Figura 6

Estructura interna del módulo de transmisión (TXM).

A los elementos antes mencionados se suman el registro de retención de transmisión (THR), la lógica de chequeo de transmisión, que en este caso estaría más bien llamarle lógica de generación de paridad, (LTSR), el registro de desplazamiento de transmisión (TSR) y el registro de estado de línea (LSR), en los bits que competen a la transmisión.

El registro de retención de transmisión (THR) es un registro común de 8 bits que realiza la misma función que su homólogo en el 8250, la cual no es más que almacenar (retener) temporalmente el valor del **byte** a transmitir en el caso de que se esté produciendo una transmisión, hasta que esta concluya. De no estarse efectuando una transmisión casi inmediatamente su valor se escribirá en el TSR.

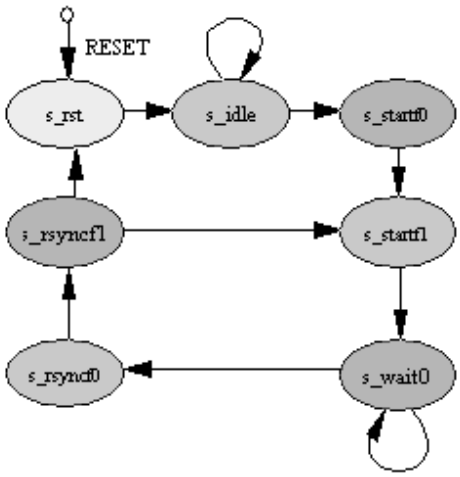

Figura 7

Diagrama de estados de la unidad de control de transmisión.

La lógica de chequeo de transmisión (LTSR), es la lógica necesaria para generar la paridad y el bit de stop para las diferentes variantes disponibles, de este módulo los elementos más sobresalientes son una puerta ***XOR***, de 8 bits y dos (2) multiplexores.

El registro de desplazamiento de transmisión (TSR) es análogo al RSR con la diferencia que este es utilizado para la transmisión de los datos serie. Es un registro de 10 bits con desplazamiento a la derecha.

El registro de estado de línea (LSR) para la transmisión está compuesto por los bits:

THRE: Transmitter Holding Register Empty. (BIT 5)

TEMT: Transmitter Empty. (BIT 6)

RESULTADOS

Las tablas de costo de implementación en el FPGA se muestran, Tabla I y Tabla II, en cada caso se muestran los costes de implementación de los bloques funcionales y finalmente para la estructura completa.
Como se muestra en ambas tablas el resultado de la implementación no supera el 2% de los recursos para una tarjeta con el XC3S1000FT256.

Tabla II: Tabla de costos de implementación XC3S1000FT256

Elemento	S	%	SFF	%	4ILUT	%
BRGR	9	0.12	9	0.10	-	-
LCR	5	0.07	8	0.05	-	-

RXM	68	0.89	64	0.42	125	0.81
TXM	56	0.73	57	0.37	101	0.66
ASCPI	146	1.9	145	0.94	263	1.71

S: Slices
SFF: Slices Flip/Flops
4ILUT: 4 - Input LUTs

Tabla III: Tabla de costos de bloques de entrada/salida y señales de reloj XC3S1000FT256

Elemento	IOBs	%	GCLK	%
BRGR	34	19	1	12.5
LCR	18	10	1	12.5
RXM	35	20	1	12.5
TXM	32	18	1	12.5
ASCPI	25	14	1	12.5

IOBs: Bloques de entrada/salida
GCLK: Buffers globales de reloj.

El análisis de respuesta en frecuencia da igualmente resultados positivos, aunque en este caso no es de suma importancia que el dispositivo sea muy rápido ya que los autores ven como característica más útil la posibilidad de interconexión con computadoras que incorporen UART y cuyas velocidades más comunes no son precisamente altas.

Las Tablas IV y V muestran el resumen de la respuesta en frecuencia del dispositivo para cada una de las especificaciones temporales y los peores caminos respectivamente.

Tabla IV: Retrasos en ns de la lógica del sistema.

C2S	C2P	P2S
5.369ns	8.674ns	6.307ns

Tabla V: Peores caminos en cada caso.

C2S	C2P	P2S
cnt16-cnt4	CLK – D[3]	A[2]– CLK

C2S: Clock to setup
C2P: Clock to pad
P2S: Pad to setup

La Figura 8 muestra los resultados de una simulación funcional. Para ello se ha montado un banco de prueba (testbench) con los datos de la simulación hasta el pulso número 769 de un reloj de 200 ns de período (5 MHz). La simulación incluye la configuración del dispositivo, así como las lecturas de encuesta al pin **DR** (**DATA READY**), para conocer cuando ha llegado un nuevo dato al módulo de recepción.

Los parámetros de configuración del dispositivo se muestran en la Tabla VI. La velocidad se ha tomado grande para que la simulación no se extienda demasiado en el tiempo de manera que sea más rápido y fácil de comprobar el correcto funcionamiento del dispositivo.

Se ha elegido el modo **loop** para comprobar el funcionamiento del sistema debido a que permite en una misma simulación comprobar tanto el sistema de transmisión como el de recepción.

Tabla VI: Parámetros de configuración del dispositivo para la simulación.

Parámetro	Valor
Velocidad	98000 b
Divisor	2
Paridad	Sin paridad
Modo	Loop
Información a transmitir	04h
Información a recibir	04h

Obsérvese cómo las patillas de entrada/salida (SIN/SOUT) permanecen en nivel alto todo el tiempo de simulación, mientras la señal ISIN es la que recibe el dato serie internamente.

En la figura aparecen también los bits del registro de estado de línea (LSR) correspondientes a la recepción (DR, PE, OE, FE) y los bits del registro de control de línea (LCR) que se relacionan con la paridad.

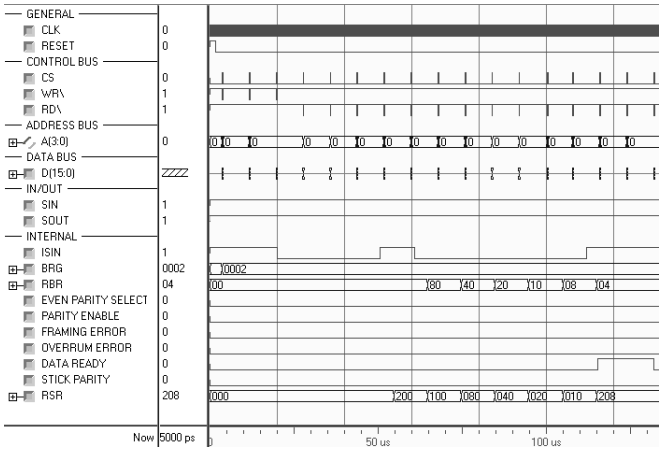

Figura 8

Resultados de la simulación funcional.

El RSR muestra la secuencia de entrada de los bits desde el LSB hasta el bit de STOP, mientras RBR es el registro buffer de recepción a la entrada es por ello que se ve como cambia la señal, pues en realidad no debe cambiar hasta que sea recibido el bit de STOP que marca el instante de registrar el contenido del RSR y modificar si es necesario el estado del registro de estado de línea (LSR).

También se muestran las señales de control del dispositivo en general, parte del bus de direcciones (A[2:0]) y el bus de datos bidireccional (D[15:0]).

DISCUSIÓN

Para discutir las características del módulo los autores se han basado en dos (2) elementos fundamentales que tienen que ver con:

1. Semejanzas, diferencias, ventajas y desventajas del módulo frente a un UART clásico (modelo 8250 ó 16550).
2. Aplicaciones de prueba

Comenzando con el primer aspecto, el módulo ha sido construido teniendo en cuenta el protocolo serie asincrónico y como modelo los UART de Intel™ por ello existe bastante similitud en la manera de programar el dispositivo pues se ha tratado, salvando las diferencias, de mantener las mismas direcciones de los registros. Estas direcciones se mantienen inalterables para el registro de estado de línea (LSR), el registro buffer de recepción (RBR), el registro de transmisión (THR) y el registro de control de línea (LCR).

Esta situación permite que a pesar de no ser un UART completo el dispositivo siga parte del algoritmo clásico de programación de un 8250 o un 16550.

Esta filosofía le da la posibilidad al diseño de seguir creciendo en el futuro hasta dejar de ser una interfaz bastante simple para convertirse en un verdadero UART.

Las diferencias más significativas están dadas en la imposibilidad de la interfaz de variar el tamaño de la trama (fija en 8) y el número de bits de stop (fijo en 1), la inexistencia de los registros asociados al tratamiento de interrupciones (IER, IIR) y señales y control de módem (MCR, MSR).

Otras diferencias se aprecian en que la interfaz no especifica reloj así que este puede tener en general una frecuencia distinta a la clásica 1.8432 MHz, no existe soporte para el BIT 4 del LSR (**BI**) y por último debido al tamaño del bus de datos (16 bits) no tiene efecto utilizar **DLAB** al programar la velocidad, lo que trae como consecuencia que para programar los divisores esto se realiza en una sola operación de transferencia de 16 bits y como si se estuviera escribiendo el bit más significativo de la velocidad en un UART clásico, esto es, en la misma dirección.

Pasando al tema de la aplicaciones, para discutir los resultados los autores se auxiliaron de dos mini plataformas distintas, la plataforma número uno se muestra en la Figura 9 y no es más que una extensión real del **testbench** realizado en el acápite anterior.

Se trata de una plataforma que además de la interfaz de comunicación que se analiza, incorpora el procesador Silmaril 1.1, y un módulo de visualización para cuatro lámparas de 7 segmentos. [5] - [6].

El objetivo de la plataforma es comprobar prácticamente el modo de funcionamiento en lazo (**loop**) y en general de toda la estructura de la interfaz.

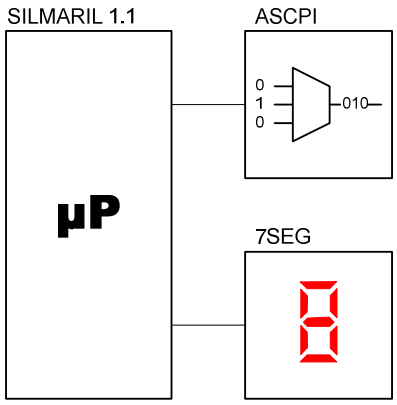

Figura 9

Mini plataforma 1.

El programa que incorpora el procesador implementa el algoritmo que se muestra en la Figura 10.

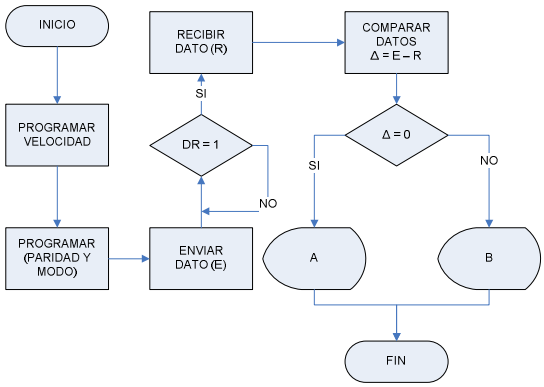

Figura 10

Algoritmo de prueba en la mini plataforma 1.

Básicamente se trata de programar el procesador para que a su vez programe la interfaz en modo **loop** y seguidamente transmitir un dato, recibir el dato y comparar el dato enviado con el recibido (autodiagnóstico) y como señal de éxito o fallo encender una de las lámparas con un valor específico para cada caso.

Lo primero a hacer es programar la velocidad de transmisión y recepción, a continuación se programan las características de la trama, esto es, el tipo de paridad, pues es fijo 8 bits de datos, 1 bit de stop; y el modo.

Seguidamente se envía un **byte** de información (E) al THR para inmediatamente comenzar a encuestar el bit DR (DATA READY) del Registro de Estado de Línea (LSR). Cuando se halla recibido el dato se lee el contenido del Registro Buffer de Recepción (RBR) que es en definitiva el **byte** recibido (R).

El próximo paso es comparar el dato enviado (E) con el dato recibido (R), si la diferencia es cero entonces son iguales y se procede a encender una de las lámparas con el carácter A, si no es así se ilumina la lámpara con el valor B.

La mini plataforma 2 se muestra en la Figura 11, como se ve se trata de un procesador, conectado a un adaptador de vídeo VGA con su correspondiente conector, un adaptador de teclado, similar al utilizado en el IBM™ PC AT y la interfaz serie conectada a un adaptador de nivel físico (conector RS232C) y un cable común para conectar al PC.

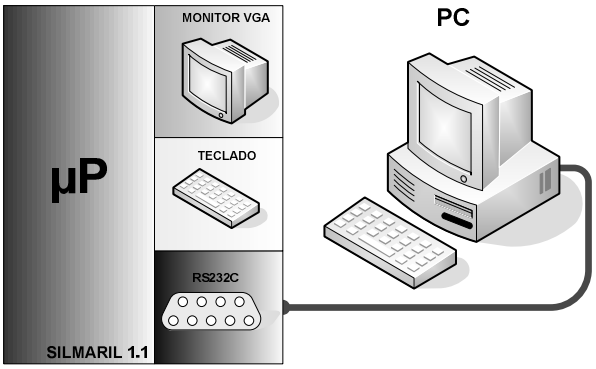

Figura 11

Mini plataforma 2.

El objetivo de la prueba es realizar un pequeño Chat entre el ordenador (PC) y la plataforma para el intercambio de información entre ambas. Lo que requiere hardware y software del lado de la plataforma y software del lado del PC.

La plataforma ha sido construida y se encuentra funcionando en una tarjeta de desarrollo que incorpora además de los adaptadores de nivel físico para monitor VGA y teclado, un Spartan™ - 3 XC3S1000-4FT256 de Xilinx®.

CONCLUSIONES

Hasta aquí se han mostrado el diseño, la implementación y algunas plataformas de prueba de la interfaz tratada.

Se han examinado las principales características que acercan y alejan la interfaz al comportamiento clásico de los dispositivos de transmisión serie que se encuentran en los ordenadores.

A pesar de no contar con la universalidad de un UART se puede decir que salvando la primera letra de esta sigla el dispositivo cumple con el resto y puede (como se ha mostrado) ser utilizado en numerosas aplicaciones donde se requiera la transmisión de datos hacia un PC u otro dispositivo similar.

Como se ha visto además es un dispositivo que requiere de relativamente pocos recursos y tiene una respuesta en frecuencia que puede calificarse como aceptable.

También se ha visto en el caso de la mini plataforma 2 cómo combinado con un procesador y algunos otros dispositivos se puede crear un pequeño terminal con comunicación a un PC que puede ser utilizado en las dos direcciones para intercambiar datos, lo que cobra utilidad en aplicaciones como dispositivos de control inteligentes y configurables.

REFERENCIAS

1. **XILINX, INC:** "Xilinx UG130 Spartan-3 Starter Kit Board User Guide". 2004.
2. **XILINX, INC:** "Xilinx Libraries Guide". 2004.
3. **WAKERLY, J. F:** "Digital Design. Principles and Practices". 4ed. New Jersey: Pearson Perentice Hall, pp. 553 – 587. 2006.
4. **XILINX, INC:** "StateCAD® Release 6.2i Help". 2004.
5. **ASHENDEN, P. J:** "The VHDL Cookbook". 1990.
6. **SUÁREZ LEÓN, A. A:** "Diseño del microprocesador Silmaril 1.1 para sistemas empotrados en FPGA" en *Conferencia Internacional FIE 2008*. 2008

AUTORES

Alexander A. Suárez León, Ing. en Automática, Instructor recién graduado, Universidad de Oriente. FIE, Dpto. Ingeniería Biomédica. aasl@fie.uo.edu.cu Miembro del Grupo Científico de Electrónica Programable *Ni*.

Danelia Matos Molina, Ing. en Automática, Profesor Instructor, Centro Provincial de Electromedicina, Santiago de Cuba, Dpto. Electro-óptica. Miembro del Grupo Científico de Electrónica Programable *Ni*.

Roger E. Rivero Labrada, Ing. en Materiales y Componentes Electrónicos, Profesor Asistente, Master en Ciencias, Universidad de Oriente. FIE, Dpto. Ingeniería Biomédica. roger@fie.uo.edu.cu. Subdirector Grupo Científico de Electrónica Programable *Ni*. Jefe de Carrera Ingeniería Biomédica.

Rubén D. López Noa, Ing. en Automática, Profesor Auxiliar, Master en Ciencias. Universidad de Oriente FIE, Dpto. Ingeniería Biomédica. lnoa@fie.uo.edu.cu Director del Grupo Científico de Electrónica Programable *Ni*. Jefe de Departamento Ingeniería Biomédica.

Berta Pallerols Mir, Ing. en Telecomunicaciones, Profesora Auxiliar, Universidad de Oriente. FIE, Dpto. Ingeniería en Telecomunicaciones bertapm@fie.uo.edu.cu. Miembro del Grupo Científico de Electrónica Programable *Ni*. Jefa de Departamento Ingeniería en Telecomunicaciones.