PRACTICA II

Controlador para puerto serie RS232

revisión octubre-2014: angelfh

Interfaz con un puerto serie RS232

1. Descripción del problema

El propósito de este ejercicio es implementar un controlador que ofrezca una interfaz clara a un sistema cliente de forma que puedan transmitirse y recibirse datos por una línea serie siguiendo el estándar RS232 simplificado. El esquema sería análogo al que aparece en la Figura 1.

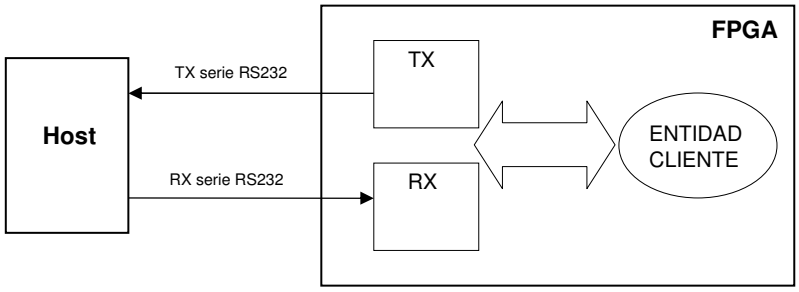

Figura 1. Esquema general del sistema de comunicación

Esta interfaz de comunicaciones serie asíncrona se compone, en su versión más simple, de una línea de transmisión y otra de recepción, siendo ambas configurables en los siguientes aspectos:

- Velocidad de transmisión;
- Número de bits de datos por palabra;
- Existencia o no de bit de paridad y tipo de ésta;
- Número de bits de parada.

El sistema que a continuación se describe convierte, por un lado, **la interfaz de transmisión en una escritura asíncrona con señales de negociación**, y por otro lado, **la recepción en una lectura de una memoria de tipo cola (FIFO) síncrona**. La frecuencia de trabajo del sistema, así como la velocidad de transmisión y recepción de los datos, son fácilmente controlables por medio de una serie de contadores internos al sistema que pueden configurarse antes de la síntesis de la arquitectura diseñada. En el caso concreto del controlador propuesto, se tomará una frecuencia de reloj de 20 MHz y una velocidad de transmisión de 115.200 bps. El resto de parámetros de la transmisión se han fijado a unos valores suficientemente genéricos como para permitir el uso del controlador en una amplia variedad de situaciones. Estos valores son:

- 8 bits de datos por palabra;
- Sin bit de paridad;
- 1 bit de parada.

De este modo, la palabra que se va a enviar y/o recibir tiene la forma de la Figura 2. Nótese que **se transmite el bit menos significativo en primer lugar.**

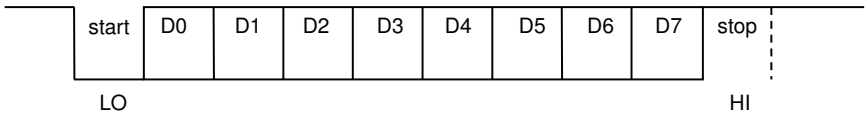

Figura 2. Palabra de datos RS232

2. Primera aproximación a la solución, diagrama de bloques

La propia definición del problema deja entrever que la solución ha de partir de la división clara entre sistema transmisor y sistema receptor, dado que ambos deben funcionar simultáneamente para que la comunicación sea bidireccional (*full-duplex*).

Para el *transmisor* se buscan las siguientes prestaciones:

- Recibir los datos del sistema cliente por un bus paralelo asíncrono con señales de negociación;
- Recibir un dato (*byte*) cada vez y enviarlo inmediatamente;
- Indicar mediante una línea si el sistema se encuentra listo para su uso u ocupado.

Las características que se piden al sistema *receptor* son algo diferentes:

- Disponer de una interfaz síncrona y paralela de retirada de datos por medio de una simple señal de lectura;
- Capacidad para recibir múltiples datos y almacenarlos internamente hasta que el sistema cliente desee retirarlos;
- Disponer hacia el sistema cliente de líneas que indiquen si existen datos por retirar y si el controlador ha agotado su memoria interna.

Teniendo en cuenta estos atributos, el controlador RS232top se compondrá de los bloques indicados en la Tabla 1.

COMPONENTE	FUNCIÓN
RS232_TX	Máquina de estados que se encarga de controlar la transmisión de los datos enviados por el sistema cliente, así como de informar a éste de la disponibilidad de la línea de transmisión.
RS232_RX	Máquina de estados que controla la línea de recepción y se encarga de muestrear su valor a fin de extraer los datos de las palabras recibidas.
ShiftRegister	Registro de carga serie y salida paralelo que se encarga de situar en paralelo los datos recibidos en serie.
FIFO	Memoria interna de tipo cola para almacenar los datos recibidos. Se encarga además de generar las señales que indican si se encuentra llena o vacía (seleccionada dentro de las librerías *core* de Xilinx).

Tabla 1. Bloques de la interfaz RS232

El protocolo de recepción de datos del sistema cliente, así como ciertos registros internos, se encuentran implementados dentro de la entidad superior en la jerarquía (RS232top). La entidad del controlador RS232top dispone de las líneas de entrada/salida descritas en la Tabla 2.

LÍNEA	SENTIDO	FUNCIÓN
Reset	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
Data_in [7..0]	Entrada	Entrada del byte que el sistema cliente desea enviar.
Valid_D	Entrada	Validación del dato de entrada por parte del sistema cliente. Activa a nivel bajo.
ACK_in	Salida	Reconocimiento del dato para transmitir. Activa a nivel bajo. Se activa al guardar un nuevo dato válido y se desactiva al desactivarse Valid_D.
TX_RDY	Salida	Indica que el transmisor está disponible. Activa a nivel alto.
TD	Salida	Línea de transmisión RS232.
RD	Entrada	Línea de recepción RS232.
Data_out [7..0]	Salida	Datos de salida de la FIFO interna del subsistema receptor.
Data_read	Entrada	Petición del sistema cliente de lectura de un nuevo dato de los recibidos y almacenados. Activa a nivel alto.
Full	Salida	Indica que la memoria interna del subsistema receptor está llena. Activa a nivel alto.
Empty	Salida	Indica que la memoria interna del subsistema receptor está vacía. Activa a nivel alto.

Tabla 2. Líneas de RS232top

3. Bloques del diseño, subsistema transmisor

3.1. RS232_TX

Para la correcta transmisión de datos por la línea serie RS232 hay que diseñar una máquina de estados que se ocupe de construir la palabra de acuerdo con los parámetros ya mencionados. Con ayuda de contadores internos se generarán bits de adecuada longitud según la frecuencia de reloj y la velocidad de transmisión consideradas, y se transmitirán en el orden correcto.

La entidad RS232_TX dispone de las líneas de entrada/salida de la Tabla 3.

LÍNEA	SENTIDO	FUNCIÓN
Reset	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
Start	Entrada	Comando de comienzo de transmisión, activa a nivel alto.
Data [7..0]	Entrada	Byte de datos a transmitir.
EOT	Salida	Indica el final de una transmisión, activa a nivel alto.
TX	Salida	Línea de transmisión RS232.

Tabla 3. Líneas de RS232_TX

Internamente, serán necesarios dos contadores:

- `Data_count`: controla qué bit de datos se está enviando;
- `Pulse_width`: controla la longitud de los bits enviados según la frecuencia del reloj y la velocidad de transmisión. Para variar el límite de cuenta de este contador puede crearse una constante `PulseEndOfCount`.

Para la máquina se definen cuatro estados siguiendo el formato de la palabra: Idle, StartBit, SendData y StopBit. Como puede suponerse, la máquina permanece en el estado Idle hasta que recibe la señal de Start, momento en el que pasa a StartBit; en ese instante, la máquina empieza a contar con Pulse_width para enviar un bit de longitud adecuada (el bit de comienzo es siempre un 0) pasando después a SendData. En SendData la máquina va enviando los bits de datos según el contador Data_count, pasando al terminar al estado StopBit. Una vez enviado el bit de parada se regresa al estado Idle. Lógicamente, la salida EOT se encuentra negada en todos los estados exceptuando Idle. Esta línea se conecta directamente a la salida del controlador como TX_RDY.

3.2. Protocolo de escritura de datos para el sistema cliente

Para la construcción del protocolo interfaz entre el sistema cliente y el controlador RS232 se hace uso de un proceso dentro de la entidad 'top' del sistema, de modo que el transmisor pueda implementarse en otros diseños siguiendo otros protocolos de salida.

4. Bloques del diseño, subsistema receptor

4.1. ShiftRegister

Para la recepción de datos es necesario un sistema de memoria que se encargue de convertir los datos muestreados de la línea serie en un formato paralelo. Con este propósito se hace uso de un registro de desplazamiento con señal de habilitación de carga. Este sistema dispone de las líneas descritas en la Tabla 4.

LÍNEA	SENTIDO	FUNCIÓN
Reset	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
Enable	Entrada	Permiso de carga y desplazamiento, activa a nivel alto y de forma síncrona.
D	Entrada	Bit de llegada.
Q[7..0]	Salida	Byte en recepción por la línea RS232.

Tabla 4. Líneas de ShiftRegister

4.2. FIFO

Para almacenar los datos recibidos hasta que el sistema cliente desee retirarlos se opta por utilizar una memoria que sigue el modelo FIFO de manera síncrona. Para la construcción de este componente se ha preferido hacer uso de los modelos que el fabricante de la FPGA, Xilinx, ofrece optimizados para cada uno de sus dispositivos programables. El software para la generación de estos módulos optimizados es el CoreGen, incluido en el sistema de desarrollo de Xilinx, el ISE. El modelo a partir del cual se generó la FIFO es un *core*, que se configuró según los requisitos del sistema: una memoria de bloque de 16 bytes que contuviera las líneas indicadas en la Tabla 5.

LÍNEA	SENTIDO	FUNCIÓN
Rst	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
Din[7..0]	Entrada	Puerto de escritura de datos.
Wr_en	Entrada	Habilitación de escritura síncrona y activa a nivel alto.
Rd_en	Entrada	Habilitación de lectura síncrona y activa a nivel alto.
Dout[7..0]	Salida	Puerto de lectura de datos.
Full	Salida	Indica que la FIFO está llena.
Empty	Salida	Indica que la FIFO está vacía.

Tabla 5. Líneas de la memoria FIFO

4.3. RS232_RX

Del mismo modo que para el transmisor, es necesario disponer de una máquina de estados que en este caso se dedicará a muestrear la línea de recepción serie en los momentos oportunos. Para ello se dota a este bloque con las líneas de entrada/salida indicadas en la Tabla 6.

LÍNEA	SENTIDO	FUNCIÓN
Reset	Entrada	Reset asíncrono y activo a nivel bajo.
Clk	Entrada	Reloj del sistema.
LineRD_in	Entrada	Línea de recepción RS232.
Valid_out	Salida	Indica que el bit recién muestreado es válido y debe ser almacenado en el registro de desplazamiento, activa a nivel alto.
Code_out	Salida	Dato de salida (la línea RS232 es directamente pasada a la salida).
Store_out	Salida	Indica que el byte recién recibido es válido y debe pasar del registro de desplazamiento a la memoria interna, activa a nivel alto.

Tabla 6. Puertos de RS232_RX

Para cumplir su propósito la máquina dispone de tres contadores:[1]

- `DataCount`, que cuenta los bits de datos recibidos;
- `HalfBitCounter`, que cuenta ciclos de reloj desde la detección del flanco inicial del bit de comienzo hasta su centro, donde debe ser muestreado;
- `BitCounter`, que cuenta también ciclos de reloj, pero en este caso los correspondientes a la longitud de un bit completo.

La máquina dispone de cuatro estados: Idle, StartBit, RcvData, StopBit, permaneciendo en el primero hasta que detecta un nivel bajo en la línea, momento en el que se inicia la recepción.

Como puede imaginarse, el control, cuando se encuentra en el estado RcvData, aumenta el contador DataCount cada vez que BitCounter llega al final de su cuenta, momento en el cual se muestrea la señal serie y se da la orden de almacenarla en el registro de desplazamiento (Valid_out='1'). Una vez recibidos todos los bits, se comprueba que no hay errores de alineamiento en la palabra mediante la comprobación del bit de parada y, si éste es correcto, se da la orden de que el dato recibido y almacenado en el registro de desplazamiento pase a la memoria interna (Store_out = '1').

5. Ficheros disponibles

Estarán disponibles los siguientes ficheros:

- `RS232top.vhd`: definición de la entidad RS232top de la Tabla 2 junto con una implementación haciendo uso de los módulos de la Tabla 1. Incluye en un proceso llamado Clocking el protocolo asíncrono de comunicación entre cliente y transmisor.
- `tb_RS232top.vhd`: fichero de testbench para el anterior.
- `RS232top.ucf`: fichero de restricciones para una frecuencia de reloj de 20 MHz.
- `RS232_test.ucf`: definición del procedimiento Transmit() para facilitar la generación de señales con temporización RS232 a 115.200 bps en un testbench.

[1] Los dos últimos pueden implementarse como uno solo, ya que no se emplean nunca simultáneamente.