LCSE

**ENTREGA 1: MÓDULO RS-232**

Miembros del grupo: Álvaro Montesano Martínez y Rafael Fernández Parra Fecha: 8/10/23

1) Introducción. (~ 1/2 página)

El objetivo de esta práctica inicial es la implementación de un controlador que permita a un sistema cliente transmitir y recibir datos por la línea serie asíncrona, siguiendo el estándar RS232 simplificado.

Esta interfaz se compone de una línea de transmisión otra de recepción y permite la configuración de cuatro características: velocidad de transmisión, numero de bits por palabras, uso de bit de paridad y numero de bits de parada.

Las funciones principales del controlador son:

1. Convertir la información recibida por el transmisor en una escritura asíncrona y enviarlo según se haya recibido.
2. Indicar mediante una señal el estado del transmisor.
3. Adquirir los datos recibidos por la línea de recepción y transmitirlo de manera síncrona al FIFO.
4. La FIFO almacena la información a la espera de la instrucción de lectura.

Para comenzar a desarrollar este sistema fijaremos unos valores determinados a algunas de las variables comentadas anteriormente, las cuales se podrán modificar en un fututo. Estableceremos una frecuencia de 20 MHz y una velocidad de transmisión de 115200 bps.

Debido a estos valores establecidos y los predeterminados de las demás variables modificables, la información para transmitir estará estructurada de la siguiente manera: 1 bit de Start, 8 bit de información y 1 bit de Stop.

Tabla

Descripción generada automáticamente con confianza media

2) Módulo RS-232

|  |  |
| --- | --- |
| Retos técnicos superados del RS-232 | Conseguido (Sí/No) |
| Implementación del RX en VHDL | SÍ |
| Implementación del TX en VHDL | Sí |
| Simulación Behavioral | Sí |
| Simulación con retardos | Sí |
| Sistema RS232 funcionando en placa | No |

|  |
| --- |
| Mejoras del RS-232 |
|  |
|  |
|  |

2.1) Descripción del sistema RS-232 (2 a 3 páginas)

El sistema, como se ha anticipado previamente, está formado por un transmisor y un receptor. Como se puede observar en el diagrama de bloques de la ilustración 1, el módulo de transmisión está compuesto por el bloque RS323\_TX, el cual representa la máquina de estados del transmisor. Mientras que el módulo receptor está compuesto al igual que el transmisor, por un bloque que representa la máquina de estados, además de un resgitro de desplazamiento llamado “ShiftRegister” y una memoria FIFO.

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 1:Diagrama de bloques

Para realizar el módulo de transmisión, hemos decidido separarlo en dos archivos VHDL independientes. Por una parte, estaría el contador, el cual está compuesto de un contador de pulsos de ciclos de periodos de CLK y luego tenemos otro contador, el cual lleva la cuenta del número de bits transmitidos.

Para poder dimensionar el contador de pulsos, hemos tenido en cuenta las especificaciones decididas al inicio. Sabiendo que la frecuencia de CLK es 20 MHz y con ello el periodo de los pulsos de CLK (TCLK) es de 50 ns. Teniendo una velocidad de transmisión de 115200 bps, calculamos que su periodo (Tbits) es de 8,68 us. Con estos datos podemos calcular el número de periodos de CLK que se suceden por cada bit de información transmitido.

(periodo CLK)/bit

Con estos datos sabemos el tamaño del vector a usar para contabilizar los pulsos de CLK, siendo este un vector de 8 bits. Usando en VHDL un vector ‘unsigned (7 downto 0)’, ya que nos proporciona una mayor facilidad a la hora de operar aritméticamente.

Cada vez que el contador llega 174 se el contador de bits, el cual cuenta de 0 a 7 con un vector de 3 bits. Este vector ‘unsigned (2 downto 0)’ es la señal que le proporcionamos a la máquina de estados de transmisor.

Como se puede observar en la máquina de estados de la ilustración 2, el sistema se inicia con un RST a nivel bajo que nos posiciona en el estado *Idle,* el cual tras recibir la señal de Start=’1’ pasa al estado *StartBit*. En este estado mandamos una señal de enable al contador y pasamos al estado de *SendData*. En el cual, el contador empieza a contar el número de bits transmitidos. Cuando la cuenta (dataCount) alcanza los 8 bit transmitidos (en el diagrama igualamos el contador a 7 porque esta inicializado a 0) el estado pasa a ser el de *StopBit* y TX envía un ‘1’ es cual es el bit de parada. Tras terminar el ciclo el estado vuelve a ser Idle a la espera de transmitir la siguiente palabra.

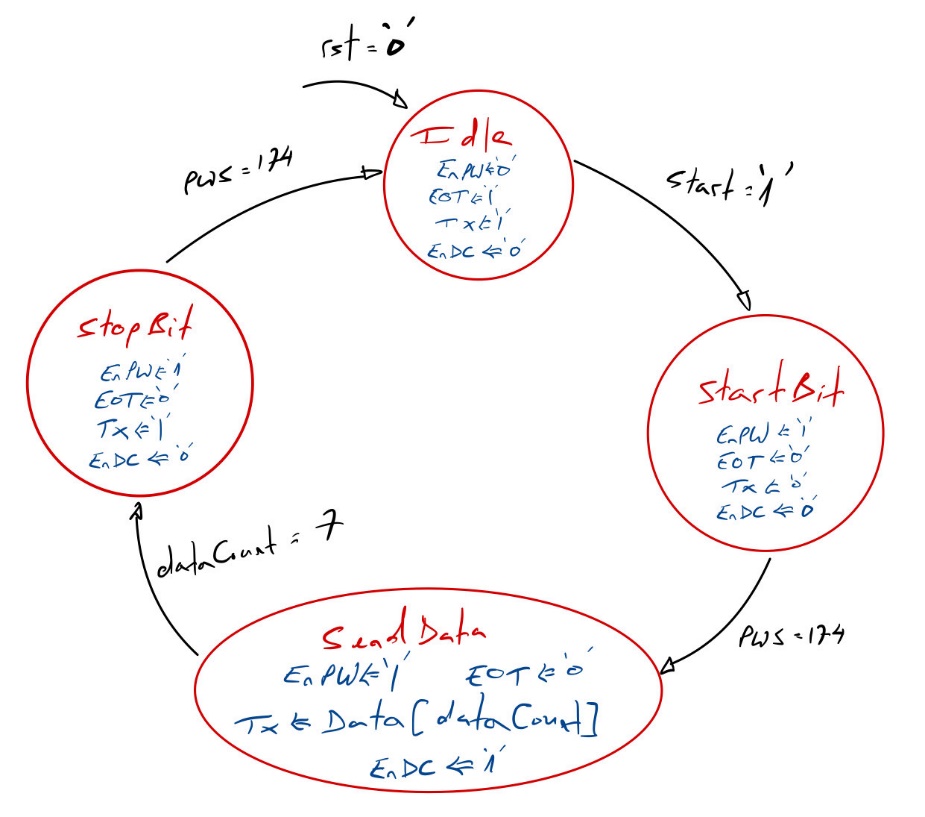


Ilustración 2: Máquina de estados RS232\_TX

Como se ha comentado antes, el módulo de recepción está formado por el módulo RS232\_RX, un shift register y una memoria FIFO (First In, First Out).

A su vez, el módulo RS232\_RX está formado por una máquina de estados y un contador (BitCounter). Este contador esta formado, al igual que en el caso del transmisor, por un contador de periodos, un contador de bits y añadiéndole un contador de medio periodo, el cual usaremos para retrasar al inicio de la recepción, la cuenta y así poder muestrear los bits recibidos en la mitad de un periodo de bit (N=87(periodo CLK) /bit).

Por lo que el proceso que seguiría sería el siguiente, el contador de pulsos cuenta hasta 174, activa el contador de medio periodo que incrementa su contador hasta 87 una única vez (al inicio de la palabra). Este, tras haber alcanzado la mitad del bit, activa el contador de bits. El contador de bit al ser activado tendrá como señal de referencia el contador de pulsos, por lo que incrementará su contador cada 174 periodos de CLK, hasta contabilizar los 8 bits de datos de la palabra.

Como se puede observar en la máquina de estados de la ilustración 3, el sistema comienza en el estado *Idle* cuando el sistema recibe un RST a nivel bajo. Si recibimos un ‘0’ en la línea de recepción, el estado transita a *StartBit*, tras el cual pasamos al *RevData*. En este estado es donde activamos el contador mediante la señal EnCnt a nivel alto. Una vez recibidos todos los bits de información pasamos al estado *StopBit* activando la señal de almacenamiento (Store\_tmp). En este estado es donde esperamos la recepción de una señal a nivel bajo de la línea de recepción. Tras su recepción volvemos al estado *Idle* donde esperamos el comienzo de la siguiente palabra y comenzar de nuevo el proceso.

­­­ Diagrama

Descripción generada automáticamente

Ilustración 3: Máquina de estados RS232\_RX

Durante el estado de *RevData* los bits de información se envían al ShiftRegister el cual los va concatenando en un vector 8 bits. Según recibe un bit nuevo, desplaza el vector una posición a la derecha y concatena el bit recibido en la posición más significativa del vector.

Una vez que se ha recibido la palabra entera y la máquina de estados está en el estado *StopBit*, esta activa el FIFO mediante la señal Store que activa el enable del modo escritura (Wr\_en). Tras activarse este enable el FIFO almacena el bus de 8 bits recibidos del ShiftRegister, quedándose a la espera de recibir el enable de lectura (RD\_en) y así proporcionar el bus de datos almacenados.

Si la FIFO alcanza límite de memoria máximo se activa la salida Full, mientras que si se resetea la FIFO se activa la señal Empty.

2.2) Simulaciones y prueba en placa del RS-232 (≤ 1 página)

.

Primero se comprueba el correcto funcionamiento de los distintos módulos implementados mediante una *behavioral simulation* del fichero RS232top.vhd y RS232top\_tb.vhd. El resultado de esta simulación se observa en la ilustración 4 y se puede comprobar el correcto funcionamiento de la misma. Para comprobar que realiza correctamente su objetivo, se aprecia que a través de la señal TD salen los valores de Data\_in en su correspondiente estado y que en Data\_out se obtienen los valores que se trasmiten a través RD de bit en bit. Se comprueba que el dato es correctamente adquirido por el registro de desplazamiento en el estado de *RevData* y van entrando en *din* de la unidad FIFO, por último se guarda correctamente en la memoria FIFO al leer el correspondiente bit de parada en el estado de *StopBit,* sacando así el dato correspondiente a través de Data\_out.

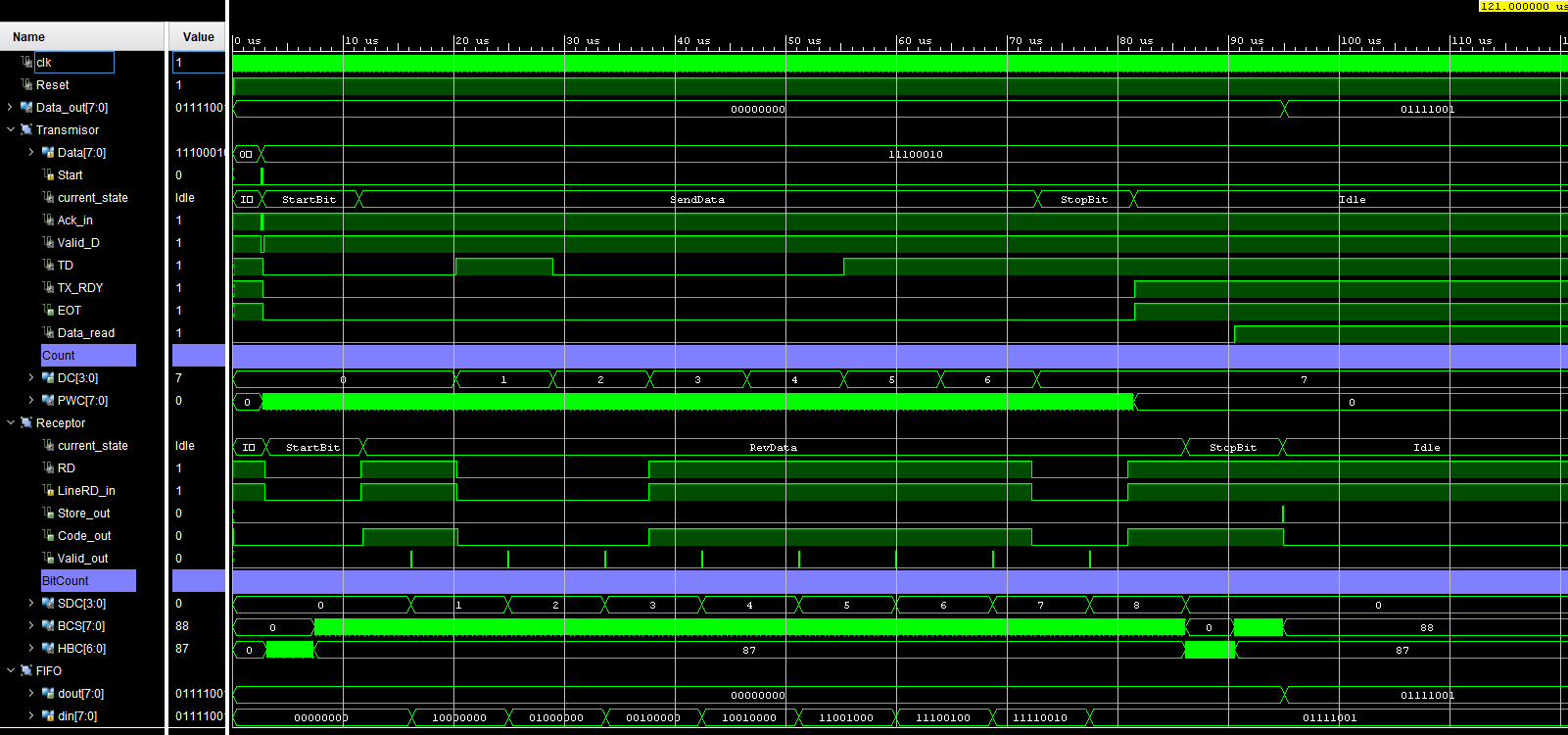


Ilustración 4: Behavioral simulation

Luego se ha comprobado el funcionamiento del programa con retardos. Para poder realizar esta simulación se han tenido que seleccionar como top los ficheros denominados Nexys\_232.vhd y tb\_nexys\_RS232.vhd. Después, se realiza la síntesis y se realiza la implementación del programa. Por último, se corre la simulación con retardos después de ser implementado el programa, el resultado de esta se puede observar en la Ilustración 5. En esta se puede apreciar que el dato transferido es correcto mediante la señal TD y que al en Data\_out se obtiene el resultado del dato generado en el test bench y guardado en la FIFO. Cabe destacar que en esta simulación se pueden observar ciertos pulsos en las señales que no están programados, estos ocurren debido a los glitches surgidos al implementar retardos en la simulación.

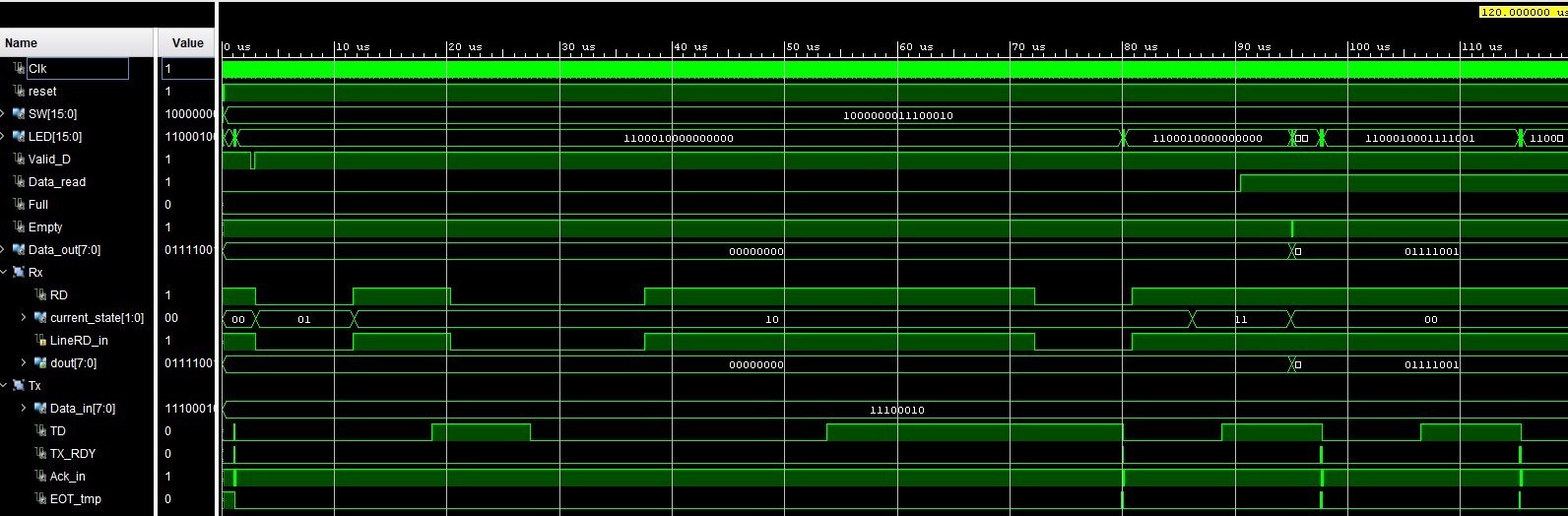


Ilustración 5: Simulación con retardos.

Por último, se intenta cargar el programa en la placa Nexys 4 DDR, para ello una vez implementado en programa se genera el bitstream y se carga en la placa. Sin embargo, la placa reconoce el programa, pero no tiene el funcionamiento esperado y visto en las distintas simulaciones.

2.3) Resultados de la síntesis (~ 1/2 página)

Una vez realizada la síntesis se piden los datos de utilización a Vivado, los datos aportados por el programa se encuentran en la Ilustración 6 y 7. En estas ilustraciones se observa una utilización de 102 LUTs, 122 FFs, 0 DSP slices y solo una BRAM. La frecuencia de reloj utilizada por el reloj generado es de 50 ns y la del reloj del sistema de 10 ns. Como último comentario se puede observar que no hay ningún latch lo cual es buena señal ya que implica que toda señal esta correctamente asignada y conectada.

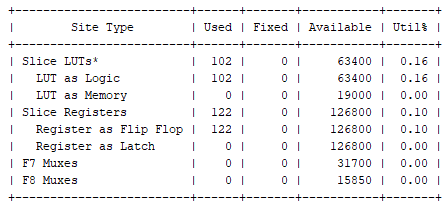


Ilustración 6: Slices.

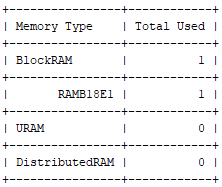


Ilustración 7: BRAM.

2.4) Mejoras del RS-232 (≤ 1 página por mejora)

(En este apartado, explicad las mejoras del RS232 realizadas).

(Al terminar de escribir este documento, eliminad todos los comentarios en azul. La longitud máxima de este documento es de 5 páginas más las mejoras).