Escuela Politécnica Superior,
Grado en Informática

Asignatura: Diseño Automático de Sistemas

Diseño de un receptor UART

11 de abril de 2022

UNIVERSIDAD NEBRIJA

UNIVERSIDAD
NEBRIJA

Pablo Sarabia Ortiz

Índice/Tabla de contenidos

1. Introducción

1.1. Presentación

En este ejercicio se va a implementar un diseño de un receptor UART en VHDL. Se va a proporcionar el diagrama de bloques del conjunto y la máquina de estados del receptor. El alumno debe implementar tanto el código del receptor como el testbench para verificar su funcionamiento.

1.2. Bibliografía recomendada

Las guías previas de cómo realizar diseños en VHDL y los recursos en la plataforma. Referencia en inglés al protocolo UART.

1.3. Entrega y evaluación

Este problema deberá ser entregado en la plataforma de la asignatura con los mismos grupos de prácticas el viernes **29 de abril a medianoche**. Estoy disponible tanto en clases como en tutorías para cualquier duda que tengáis.

Se debe entregar el código: **testbench/source y un informe**. Se puntuará sobre 10. Este problema lo tiene que entregar cada miembro del grupo en la entrega de la plataforma.

El código debe ser lo suficientemente claro para entender el funcionamiento, por favor usad nombres de variables apropiados, indentación y comentarios cuando sea necesario.

Este ejercicio se enmarca en la puntuación por participación del alumno y os sirve como preparación de cara al examen final.

El informe debe tener al menos una descripción de las entradas y salidas, máquina de estados y cualquier detalle sobre el funcionamiento o la implementación que consideréis necesario.

2. Definición del protocolo UART

El protocolo UART (Universal Asynchronous Receiver Transmitter) utiliza una conexión serie, es un protocolo full dúplex que permite enviar y recibir a la vez. El esquema de las conexiones de bus sería el siguiente Figura 1**Error! Reference source not found.**:

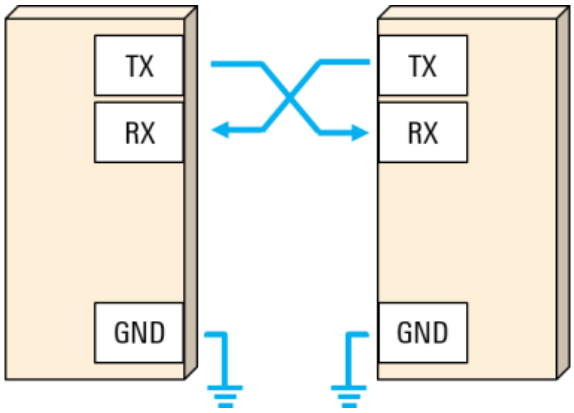

Figura 1. Interconexión de dos sistemas mediante UART.

En este problema solo se va a implementar el receptor luego nos basta con la entrada RX de cualquiera de los dos sistemas mostrados en la Figura 1

A continuación, vamos a definir el formato de los datos y la trama de la comunicación. Por defecto la línea se encuentra al nivel lógico alto ('1'), lo que se conoce como el estado de IDLE.

Primero se envía el bit de START que consiste en bajar el nivel de la línea de datos de alto a bajo. Este bit de comienzo sirve para que el receptor sepa cuando comienza la transmisión. La transmisión consta de 8 bits que se enviarán a una frecuencia acordada. Una vez realizado este envío de datos se incluirá un bit de paridad par o impar (acordado con el emisor) para que el receptor pueda verificar que los datos se han recibido correctamente. Por ultimo se incluye un bit de STOP, que consiste en un bit a nivel alto para indicar que se ha terminado la comunicación.

2.1. Bit de paridad

El bit de paridad tiene como objetivo que el numero de unos en la transmisión (palabra de 8 bits + bit de paridad), sea par o impar según lo acordado con el emisor. Para no complicar la práctica más, está disponible en el repositorio del profesor el circuito combinacional que dada una palabra te devuelve el valor del bit de paridad para el tipo de paridad seleccionado. Después deberás comprobar que el valor es el mismo que el que se ha leído en la trama de datos recibida. La interfaz de la entidad aparece en la Figura 2.

```
entity parity_checker is
      generic(
          g_size_word : integer := 8;
--Number of bytes to check parity
          g_parity_type : std_logic := '0'
-- '0' for even '1' for odd
      );
      port(
          input_bits : in std_logic_vector(g_size_word-1 downto 0);
          parity : out std_logic
      );
end parity_checker;
```

Figura 2. Interfaz del comprobador de paridad.

3. Descripción detallada del funcionamiento del receptor.

En esta práctica se pide implementar el código y los ficheros de prueba (testbench), necesarios para implementar y verificar el funcionamiento del receptor UART. El receptor va a constar de diferentes bloques, mostrados en la Figura 3.

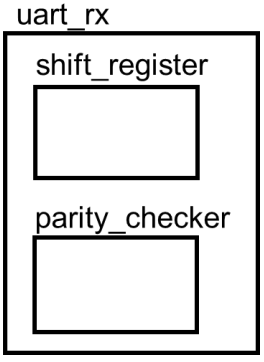

Figura 3. Diagrama de bloques del receptor UART.

3.1. Interfaz

La interfaz del receptor se muestra en la figura 4.

El receptor que se va a implementar va a permitir cambiar mediante un genérico la frecuencia de los datos (baudrate) y el tipo de paridad (par/impar).Las entradas son: rx, clk y rst. Las salidas van a ser: word (8 bits) y valid (1 bit).

La señal de rx son los datos que entran al receptor, la señal word es la palabra completa que se encuentra disponible a la salida del receptor (8 bits). La señal de valid está a 1 cuando el dato a en word es válido, más detalles a continuación. El rst es **síncrono a nivel bajo**.

```
entity UART_rx is
      generic(
            g_baudrate        : integer := 10000;
            g_parity_type     : std_logic := '0' -- '0' for even '1'
for odd
      );
      port(
            rx, clk, rst      : in  std_logic;
            valid             : out std_logic;
            word              : out std_logic_vector(7 downto 0)
      );
end UART_rx;
```

Figura 4. Interfaz del receptor UART.

3.2. Máquina de estados

Para el control del receptor se pide implementar una máquina de estados que controle el proceso de recepción de datos. **Incluid el diagrama de la maquina de estados en el informe.**

El funcionamiento del receptor es el siguiente:

1. Espera a detectar una bajada de 1 a 0 (bit de START).
2. Activa el contador con la frecuencia deseada, se debe implementar un divisor de reloj como el que hemos hecho en otros ejercicios (práctica 1 por ejemplo). Cuenta 8 bits y los almacena y desplaza los almacenados en el registro de desplazamiento (shift register).
3. Recibe el bit de paridad, si es correcto, pone la señal de valid a 1.
4. Cuenta un bit más y vuelve al estado inicial (bit de STOP).

La señal de valid solo está a 1 cuando el dato se ha comprobado que es correcto y no se ha empezado a recibir otra transmisión.

3.3. Calculo del periodo de los bits usando el baudrate

Como se ha mencionado se va a usar el baudrate (frecuencia del UART) para generar un reloj secundario que funcione a la frecuencia de la transmisión. Para ello vamos a generar un contador que cuente hasta 10 (8 bits de datos + 1 de paridad + 1 de STOP) con la frecuencia definida por el baudrate. Podéis revisar la forma de hacer un contador a una frecuencia distinta a la del sistema en la práctica 1 y 2. Tomad como valor para el genérico 10 000.

3.4. Registro de desplazamiento

Se pide implementar un registro de desplazamiento serie paralelo: Consiste en un registro con una entrada en serie que permite desplazar los bits de forma síncrona, revisad lo explicado en la práctica 2. En este caso el registro basta con que tenga como entradas una señal de shift y una de datos en serie (aparte del rst síncrono y el clock). Como salida necesita de una salida de 8 bits que después se conectará a la salida del receptor UART. Podéis reutilizar el código de las prácticas.