Enunciado del proyecto

Grupo 2

Prof. Ana Eugenia Sánchez Villalobos

Protocolo UART.

El protocolo UART (Universal Asynchronous Receiver-Transmitter) es uno de los métodos más comunes para la comunicación serial asíncrona en sistemas digitales. Permite la transmisión de datos entre dispositivos sin necesidad de una señal de reloj compartida, sincronizando en su lugar a través de configuraciones de velocidad de transmisión específicas en el transmisor y receptor. Cada unidad UART convierte los datos entre el formato paralelo usado por la CPU y el formato serial necesario para la comunicación, gestionando el flujo de bits mediante la inserción de bits de inicio y parada. Este protocolo es esencial en aplicaciones de microcontroladores, dispositivos integrados y sistemas de comunicación en donde la simplicidad y la efectividad en cortas distancias son primordiales.

Para este protocolo se va a tomar como ayuda SOLAMENTE los siguientes documentos. Solamente se toman estos documentos, debido a que al ser un protocolo extenso este puede agarrar distintos caminos, pero la idea del proyecto, es que tenga solamente ciertas especificaciones.

- https://youtu.be/sTHckUyxwp8?si=e8Mck4lRW_6d9aSE
- https://www.unm.edu/~zbaker/ece238/slides/UART.pdf
- https://cs140e.sergio.bz/notes/lec4/uart-basics.pdf
- https://www.rohde-schwarz.com/cz/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html#:~:text=UART%20stands%20for%20universal%20asynchronous,also%20have%20a%20ground%20connection.

Para este proyecto se va a utilizar el FULL DUPLEX, por lo que significa que ambos van a ser transmisores y receptores.

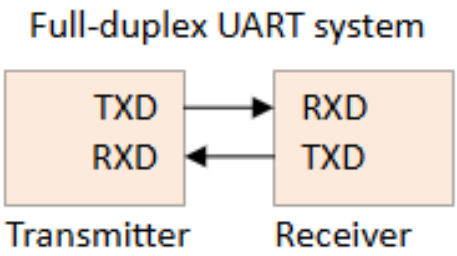

Imagen 1 Full-duplex UART system

El sistema tiene su propio clk, por lo que significa que existen dos clocks, el clock del sistema y el clock del protocolo. El clock del protocolo va a ser la mitad de la frecuencia que el clock del sistema.

El transmisor UART convierte datos paralelos en una señal serial para enviarlos a otro dispositivo. Aquí te explico paso a paso cómo funciona:

1. **Preparación de datos**: La señal al inicio cuando no transmite nada se encuentre en IDLE, que consiste en poner la señal en alto, por un tiempo indefinido hasta que se decida comenzar a enviar los datos. Va a existir una entrada llamada data_in que se encargue de tener los datos que se quieren enviar. Estos pueden ser colocados en un registro paralelo de datos (un registro temporal), que la unidad UART pueda leer y convertir.
2. **Inicio de transmisión**: Para señalar el inicio de los datos, el transmisor envía un "bit de inicio" (start bit), que es un bit de bajo nivel (0). Este bit alerta al receptor de que una nueva transmisión está comenzando.
3. **Transmisión de bits de datos**: A continuación, la UART transmite los bits de datos uno por uno, comenzando con el bit menos significativo (LSB, Least Significant Bit). La cantidad de bits de datos puede en este caso para este proyecto vamos a hacerlo de 8 bits.
4. **Bit de paridad:** Según la configuración, el transmisor puede agregar un "bit de paridad" para verificación de errores. Para el bit de paridad vamos a utilizar la configuración par. Por lo que significa que si el numero de 1's enviados en el DATA FRAME es par entonces el parity bit va a ser igual a 0, de otro modo si los bits de paridad son impares entonces el bit de paridad va a ser igual a 1.
5. **Bit de parada o finalización**: Una vez transmitidos los bits de datos (y de paridad, si aplica), el transmisor envía uno o dos "bits de parada" (stop bits), que son bits de alto nivel (1). Estos indican que la transmisión ha finalizado.
6. **Finalización y espera**: Una vez completada la transmisión, el transmisor puede quedar en un estado inactivo, a la espera de nuevos datos para transmitir. Para esto la la señal se va a quedar en IDLE, lo cual significa que se queda en 1, hasta que este vuelva a comenzar.

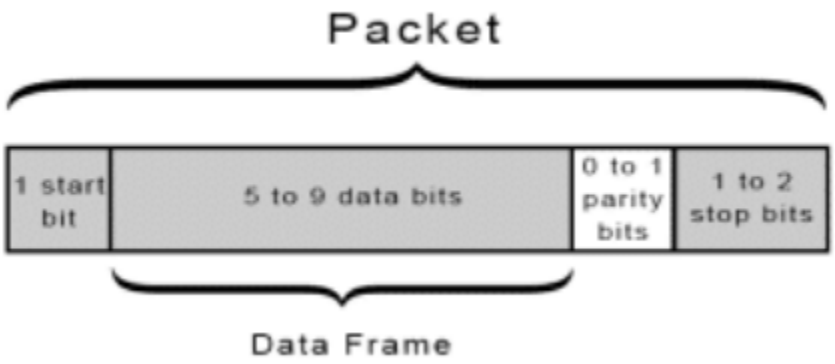

Imagen 2 Paquete UART enviado por el transmisor

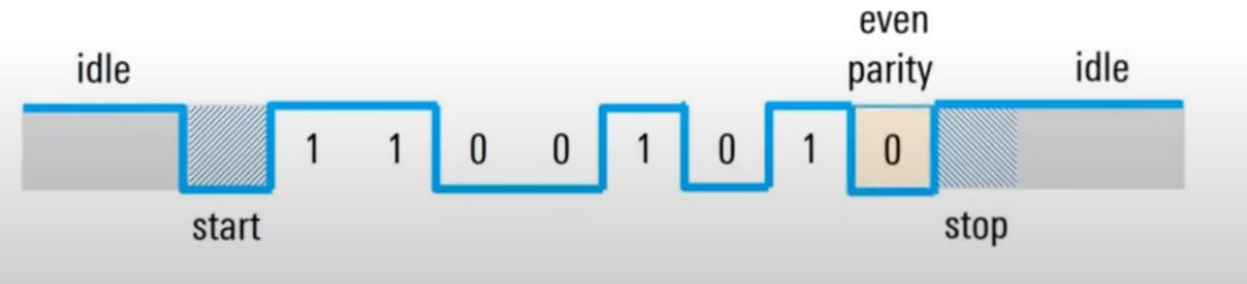

Imagen 3 Ejemplo de transmisor.

El receptor va a recibir los datos de forma serial, y va a sacar los datos de forma paralela, SIPO. Como es full dúplex significa que ambos reciben y transmiten, por lo tanto, esto se debe ver en el tester, este comportamiento no sé da al mismo tiempo.

Condiciones:

- Los datos enviados deben ser distintos, no pueden ser iguales.
- Cada bloque que contenga el receptor y un transmisor debe ser UN módulo, debido a que es full dúplex significa que son dos módulos. Este bloque debe ir luego encerrado por uno más grande que permita que se vea la comunicación total de ambos.

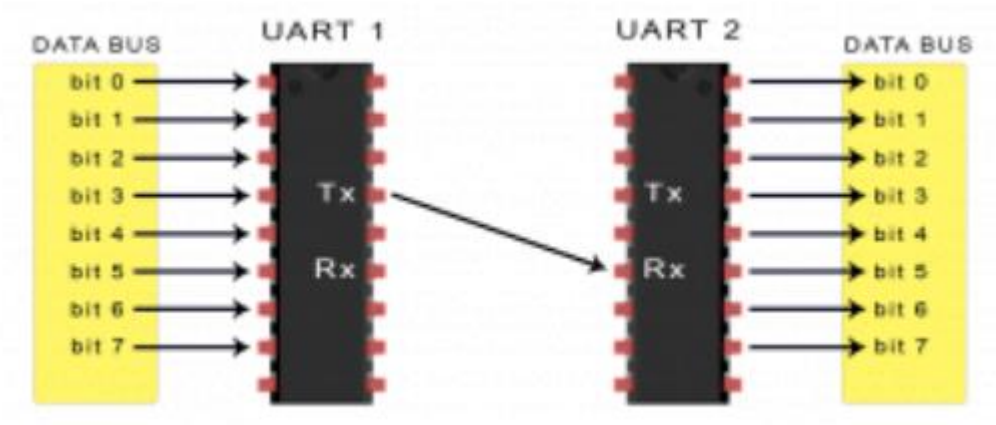

Imagen 4 Protocolo UART Full-Duplex

Ejemplo de transmisor y receptor de UART.

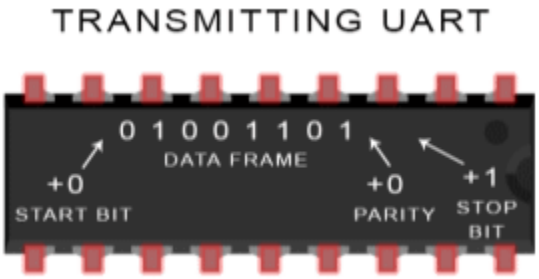

Imagen 5 Transmisor UART

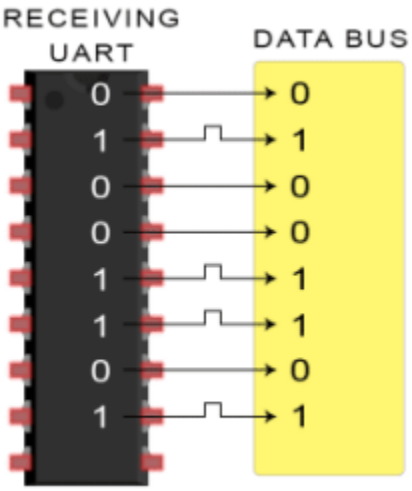

Imagen 6 Receptor UART

Para el reporte debe incluir figuras explicativas no solamente del waveforms obtenido si no también utilizando la pagina https://wavedrom.com/ . También se debe entregar un makefile, que logre ejecutar todo el protocolo. Así como un testbench y un tester. Los módulos de los dos UART deben entregarse en el mismo archivo, así como la caja que encapsula todo el sistema, como lo hemos visto en evaluaciones pasadas, esta no afecta el resultado.

El reporte debe tener

- Portada
- Introducción del protocolo y como este funciona. Aquí se deben incluir las imágenes de wavedrom.
- Forma de correr los archivos
- Trabajo en GitHub, aquí va compartido su repositorio, nombre de este, y explicación de como trabajaron todos los integrantes del equipo.
- Resultados, puede incluir aquí los screenshots de su waveform.
- Conclusiones

Puntos:

Reporte: 30ptos

Repositorio de Git: 20ptos (Si se observa que el equipo no trabajo equitativamente, se le restara puntos de esta sección a quien no haga la misma cantidad de commits que el resto. Si una persona decide realizar todo el trabajo no se le dará más puntos, al contrario perjudicará a sus compañeros, dándoles a ellos un puntaje en esta sección inferior a los 20 puntos.) El repositorio debe ser compartido con la profesora, que tiene un usuario *eusanchez* .

Programa UART: 50ptos

- **Cada UART servicio de transmisión y receptor vale 25 ptos. En esto se incluye el uso correcto de todos los bits que forman parte del paquete.**