INGENIERÍA
EN AUTOMATIZACIÓN

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INGENIERÍA

Sistemas digitales con lógica reconfigurable ll

Proyecto parcial 1 | Implementación de circuito con comunicación serial asíncrona para la detección de errores en la transmisión de bloques de datos.

Profesor: Dr. Miguel Ángel Martínez Prado

Alumno: Lorena Fuentes Flores Expediente: 263769

Santiago de Querétaro, Qro., 22 de febrero 2021.

Contenido

Objetivo

Diseñar e implementar un circuito con comunicación serial asíncrona para la detección de errores en la transmisión de bloques de datos.

Especificaciones del proyecto

- La plataforma de experimentación está configurada para que el FPGA reciba un número arbitrario de bytes con una configuración serial de 9600 bits por segundo, 7 bits de datos, 1 bit de parada y paridad impar.
- El sistema enviará de forma intencional una cantidad variable de tramas de datos con errores en la paridad. El FPGA debe detectar dichas tramas y llevar la cuenta del número de tramas erróneas.
- La cantidad de tramas erróneas debe visualizarse en formato decimal con ayuda de los displays de siete segmentos de la tarjeta.
- La señal RXD está conectada al pin del FPGA PIN_AA2.

Se realizarán cinco pruebas y el resultado de cada una es exhibido en un LCD externo y éste debe coincidir con un margen máximo de 1% de error relativo en todas las pruebas.

Marco teórico

Protocolo UART

El receptor/transmisor asíncrono universal (Universal Asynchronous Receiver/Transmitter, UART) es el dispositivo clave de un sistema de comunicaciones serie. Su función principal es convertir los datos serie a paralelos cuando se trata de datos recibidos (de entrada) y de convertir datos paralelos a serie para transmisión (de salida). En la figura 1 se muestra el esquema general con los bloques básicos de un UART. Se distinguen los registros de datos, tanto de recepción como de transmisión y sus correspondientes registros de desplazamiento (RxD, TxD), los registros de control de transmisión y recepción y señales de sincronización para comienzo de la transmisión/recepción (RTS, CTS). Aunque la transmisión es realmente asíncrona, tradicionalmente se les ha llamado asíncrona para hacer referencia al hecho de que existe flexibilidad para la transmisión de la siguiente palabra de datos.

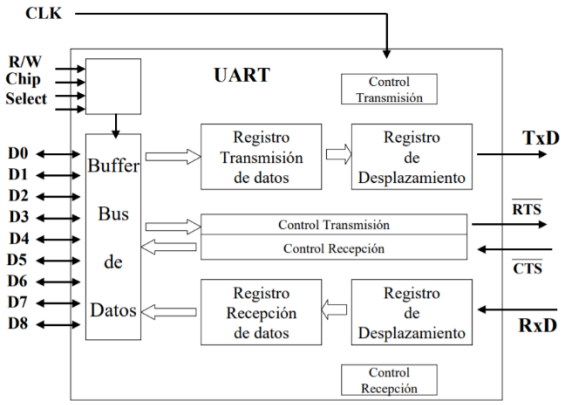

Figura 1. Diagrama de bloques de un Receptor/Transmisor Asíncrono Universal

Los parámetros que se han de definir para el correcto diseño de un interfaz de comunicaciones usando un UART son los siguientes:

- Sincronismo entre el receptor y el transmisor.
- Codificación de los datos.
- Prioridad en el envío de datos.
- Tasa de envío de datos.
- Señales eléctricas de los valores lógicos.

La sincronización en la transmisión de los datos se lleva a cabo colocando en primer lugar un bit de comienzo (start bit), después se envían los datos (data bits) usualmente entre 5 y 9 bits empezando siempre por el bit menos significativo, LSB, y por último, se envía un bit de para (stop bit). Los

UART's que trabajan con 8 bits añaden un bit de detección de error o bit de paridad. Esto se realiza secuencialmente hasta completar la transmisión. En la figura 2 se muestra el esquema de transmisión.

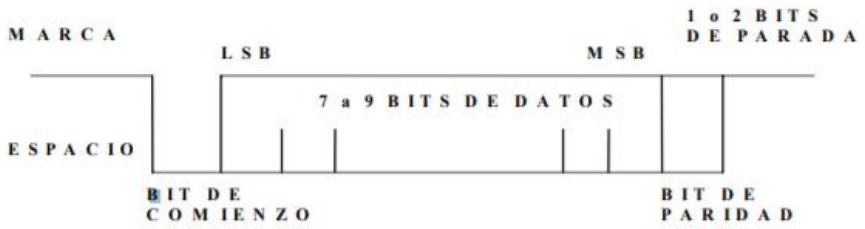

Figura 2. Esquema de transmisión/recepción asíncrona serie UART.

La codificación puede ser cualquier código binario. El más utilizado es el código ASCII (American Standard Code dor Information Interchage) que utiliza 77 bits para codificar 96 caracteres imprimibles y 32 caracteres de control.

Los niveles eléctricos para definir valores lógicos se suelen llamar marcas (1-lógico) (*mark*) y espacios (0-lógico) (*space*). Cuando no se realiza transmisión se mantiene el estado de marca (llamado también *iddle*). El comienzo de la transmisión se realiza con el estado lógico de espacio o 0-lógico (figura 2).

Bit de paridad

El bit de paridad es un parámetro con valor 0 o 1 que se utiliza en un método de detección de errores de transmisión en el que se agrega un 0 o un 1 a cada grupo de 7-8 bits (byte). El fin es que así cada byte siempre tenga una cantidad total impar de "1" o una cantidad total par de "1", según la paridad establecida.

La paridad es una técnica de detección de errores que se usa en las comunicaciones asíncronas. Se utiliza para verificar la integridad de cada byte dentro del flujo transmitido. Por ejemplo, si se establece una paridad impar, cualquier byte que se reciba de una transmisión con una cantidad total de "1" que sea par debe contener un error.

Metodología

El objetivo principal del proyecto fue encontrar paquetes de datos erróneos al contener un bit de paridad par, puesto que la configuración de la transmisión serial estaba configurada a un bit de paridad impar. Para ello recurrí al uso de electrónica digital, realizando primero la tabla de verdad de dos casos, entrada A y entrada B, por simplicidad. En la salida, realice el caso para paridad par y paridad impar.

Tabla 1. Tabla de verdad para bit de paridad con 2 bits de datos.

Entradas		**Salidas**	
A	**B**	**P**	**I**
0	**0**	**0**	**1**
0	**1**	**1**	**0**
1	**0**	**1**	**0**
1	**1**	**0**	**1**

- P = paridad par, es decir un número de 1 par.
- I = paridad impar, es decir un número de 1 impar.

Funciones:

$$P = \bar{A}B + A\bar{B} = A \oplus B$$
$$I = \bar{P}$$

Como se muestra en las funciones lógicas de resultado, para la salida de paridad impar es la misma función que la de paridad par pero negada. Basta con observar la tabla realizada para darse cuenta que concuerda con la tabla de verdad de la compuerta XOR que se muestra en la figura 3.

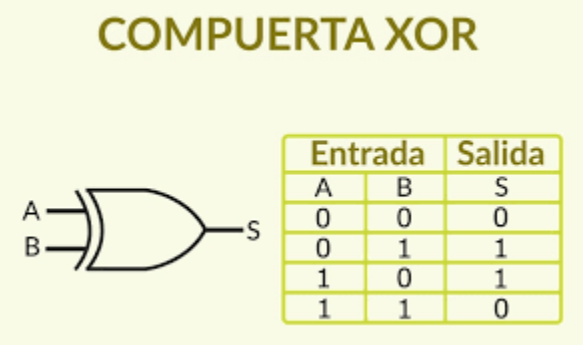

Figura 3. Tabla de verdad de compuerta lógica XOR

Aplicando la solución encontrada a un caso de 7 bits de datos con 1 bit de paridad impar, se tuvo como resultado el diagrama de la figura 4.

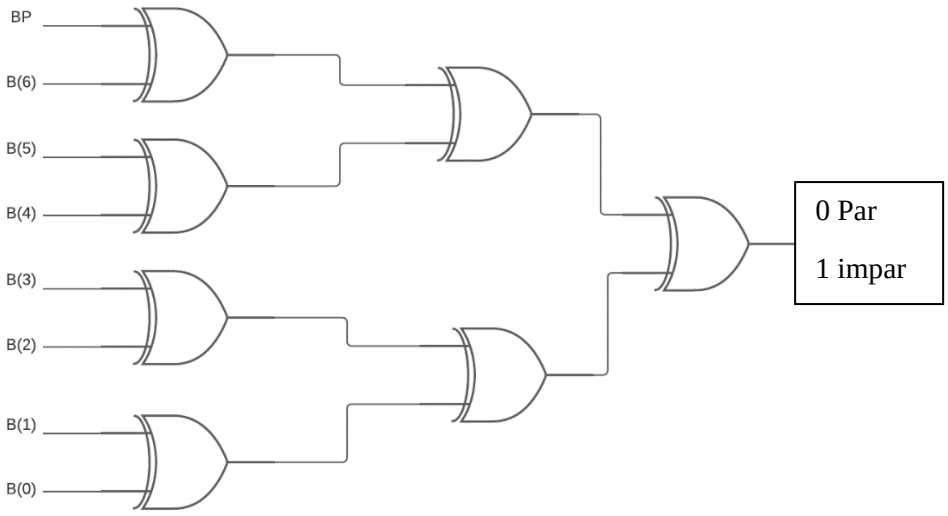

Figura 4. Diagrama del diseño del circuito para detectar el bit de paridad en un paquete de datos de 7 bits.

Teniendo el circuito listo y probado su funcionamiento, es necesario agregar un contador donde éste aumentara cuando el detector encuentre un bit de paridad par en el paquete de datos recibido, el diagrama del circuito completo final se muestra en la figura 5.

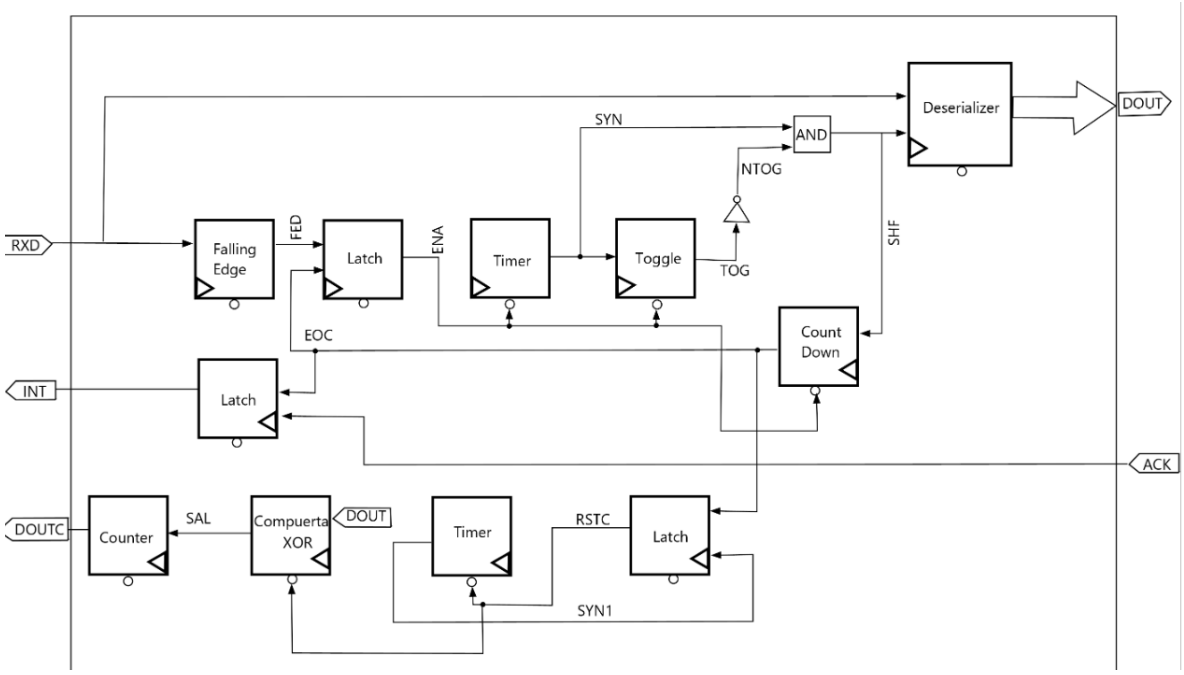

Figura 5. Diagrama a bloques UAR con detector de error de paridad.

Cabe señalar que se agregó un bloque donde se implementó el circuito diseñado de la figura 4, ya que al intentar realizar las operaciones en el circuito de mayor jerarquía en el software Active-HDL la operación se realizaba con cada dato al ser recibido y no al tener el paquete de datos total. Para ello fue necesario agregar un timer el cual fue configurado a 2 ticks de reloj, tiempo necesario para que el bloque “Compuerta XOR” realizara el proceso. Este timer es activado con la señal “RSTC” la cual

proviene de un Latch el cual a su vez es activado por la señal “EOC”, esta señal se activa cuando el paquete de datos ya fue recibido completamente. Todo esto es descrito en el diagrama a bloques de la figura 5.

Resultados

Para probar el funcionamiento del circuito implementado, fue programado un microcontrolador el cual enviaba cierto número de paquetes y entre ellos un numero aleatorio de paquetes eran erróneos pues no cumplían con la paridad impar, estos datos el microcontrolador los desplegaba en un display de 4 dígitos. La cantidad de errores detectados por la FPGA también eran mostrados en los displays de esta. La cifra mostrada debía concordar con la cifra mostrada por el microcontrolador.

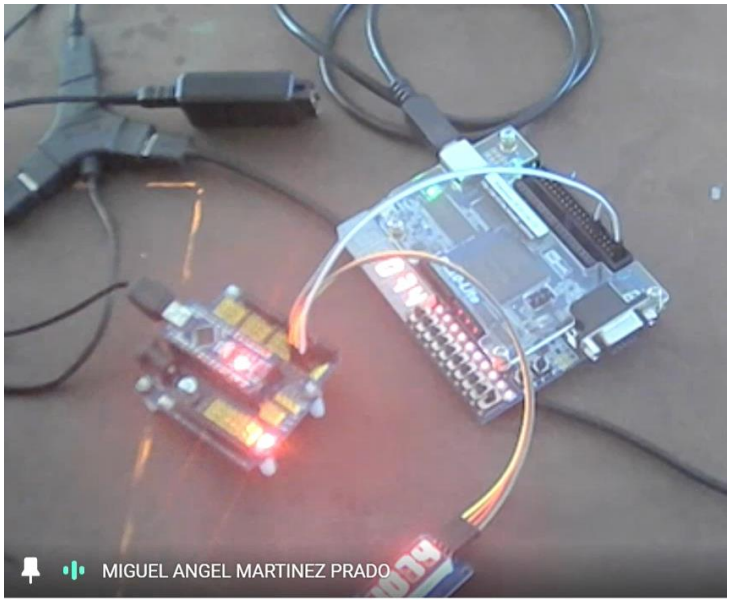

Figura 6. Primera prueba realizada con 0% de error.

Figura 7. Segunda prueba realizada con 0% de error.

En total se realizaron 5 pruebas en las cuales el circuito implementado en la FPGA tuvo 0% de error al realizar la detección de errores en el bit de paridad.

Conclusiones

El proyecto llevado a cabo tiene un 100% de funcionalidad ya que cumplió con el objetivo planteado y especificaciones teniendo un 0% de error en todas las pruebas realizas.

Fue interesante el poder realizar la plataforma del proyecto ya que se aplicaron los conocimientos vistos en el primer parcial. Algo que es sumamente necesario es el realizar el análisis de las señales para que de acuerdo a su comportamiento se encuentren errores o la manera de interconectar los bloques de circuitos.

Anexos

Repositorio donde se encuentran los archivos que componen el proyecto:

https://github.com/LoreFuentes/UAR.git