Arquitectura de Computadoras

Ingeniería en Computación FCEFyN - UNC
Trabajo Práctico 2 - UART

Aguerreberry Matthew. Mat.: 93739112
Maero Facundo. Mat.: 38479441

28 de septiembre de 2017

1. Introducción

Para el segundo trabajo práctico de la materia se realizó la implementación en Verilog de un módulo UART. Se desarrolló para la placa Nexys 4 DDR utilizando el software Vivado 2017.2. El módulo se conecta a la ALU desarrollada en el primer práctico, y a una PC, desde la cual es posible enviar parámetros para que la unidad aritmética procese.

2. Desarrollo

Se definió un módulo Top, que contiene:

- Módulo UART, compuesto por:
  - Baud Rate Generator
  - Rx
  - Tx
- Circuito Interfaz
- ALU

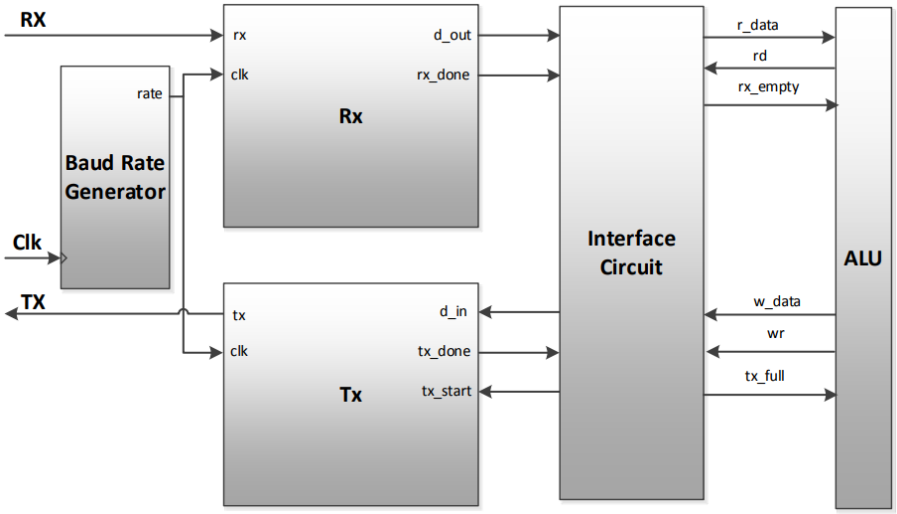

Figura 1: Diagrama de bloques del práctico

2.1. UART

2.1.1. Baud Rate Generator

Este módulo es utilizado para coordinar el muestreo de la señal recibida. Genera un tick 16 veces por Baud Rate. A partir de la entrada de clock (conociendo su frecuencia), y la velocidad de transmisión de la UART (en este caso, 9600 baudios), genera ticks que permiten al módulo Rx trabajar con la señal de entrada.

2.1.2. Rx

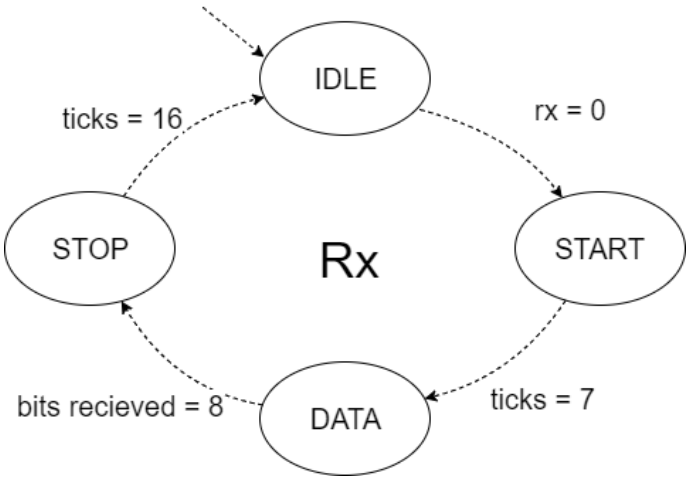

Figura 2: Diagrama de estados del receptor

Es una máquina de estados que modifica su comportamiento en función de sus entradas Rx (línea de comunicación) y tick (salida del Baud Rate Generator). Se encuentra en estado IDLE hasta que la entrada sea un 0 lógico, lo que marca el inicio del bit de start. Luego pasa al estado START, que espera 7 ticks hasta situarse en el centro del bit de start (que dura 16 ticks), lo que sincroniza el módulo con el transmisor. Una vez listo, se pasa al estado DATA, en el que toma muestras del canal Rx cada 16 ticks, que marcan el centro de cada bit recibido. El paso siguiente es pasar al estado STOP, donde se lee el único bit de stop, y finalmente se pasa al estado IDLE para comenzar el ciclo nuevamente. Al finalizar la recepción de un valor, emite un pulso en una señal de fin (rx_done_tick).

2.1.3. Tx

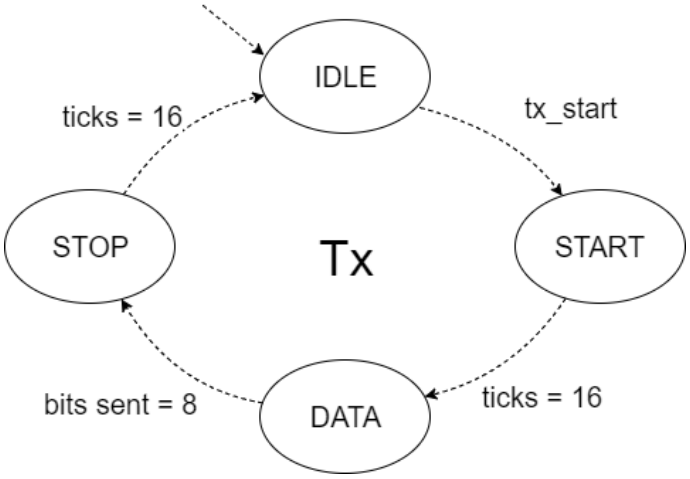

Figura 3: Diagrama de estados del transmisor

Este módulo es muy similar al anterior. Posee los mismos estados, y también utiliza la señal proveniente del Baud Rate Generator para actualizar su salida (Tx) acorde a la información que se desea transmitir. Al finalizar la transmisión emite un pulso en una señal de fin (tx_done_tick). Comienza en el estado IDLE, en el que envía por Tx un 1 lógico (canal inactivo). Al recibir la señal tx_start pasa al estado START, en el que se envía un 0 lógico por el canal durante 16 ticks del Baud Rate Generator. Acto seguido se pasa al estado DATA, en el que se envía por 16 ticks cada bit del byte a transmitir, comenzando por el menos significativo. Cuando la cantidad de bits enviados sea 8 (toda la información disponible), se pasa al estado STOP, en el que se envia un bit de stop por 16 ticks del Generator. Finalmente se pasa al estado IDLE nuevamente, a la espera de más bits para transmitir.

2.2. Circuito Interfaz

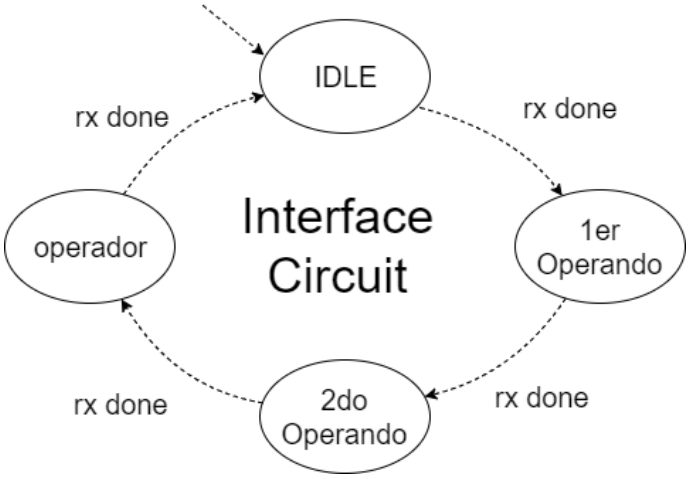

Figura 4: Diagrama de estados del receptor

Permite comunicar la ALU con el módulo UART. Recibe los bytes de Rx, los almacena en tres registros (dos para los operandos, uno para el operador), y una vez tiene todos los valores, obtiene el resultado de la ALU. Seguidamente comunica al módulo Tx que el valor está listo, y lo envía mediante un bus de 8 bits.

2.3. ALU

El módulo ALU es el desarrollado en el primer práctico. Es puramente combinacional, con dos entradas para operandos, y una entrada para el operador.

3. Test Bench

Se realizaron tres archivos de Test Bench:

3.1. Baud Rate Generator

Comprueba el funcionamiento del generador de ticks, que constituye la base de todo el módulo UART. Permite verificar que luego del número de ciclos de clock especificados, se emite un tick.

3.2. Interface

Permite verificar el comportamiento de la interfaz y la ALU. Simula la entrada de tres valores desde Rx (operandos y operador), y permite ver que efectivamente son recibidos por la unidad aritmética.

3.3. UART

Comprueba el correcto funcionamiento del bloque UART. Al conectar la salida Tx con la entrada Rx, se simula que un dato sale del bloque Rx, pasa por la interfaz y la ALU, e ingresa por el bloque Tx. Ese dato es enviado a través del conector Tx, y se ve nuevamente por Rx. En el gráfico se incluyen los estados de los bloques emisor y receptor, así como los valores de data_in y data_out, que muestran la evolución del proceso de transmisión de información.

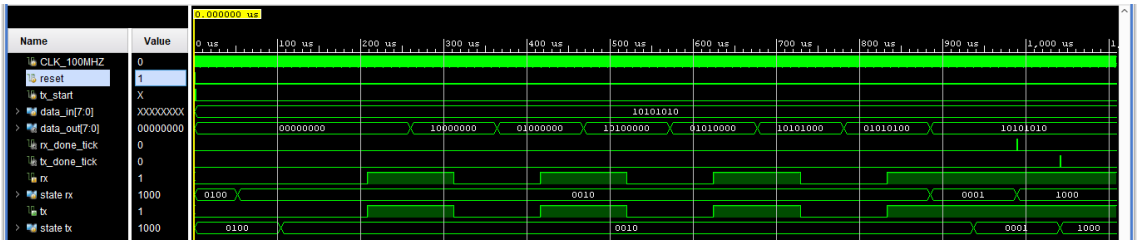

Figura 5: Test Bench del bloque UART

4. Interfaz con la PC

Para probar el funcionamiento de todo el proyecto, se programó un script en Python que se comunica con la FPGA mediante conexión USB, utilizando la librería PySerial. El script espera dos operandos y un operador, y una vez tenga los tres valores los transmite a la placa, mostrando el resultado en pantalla.

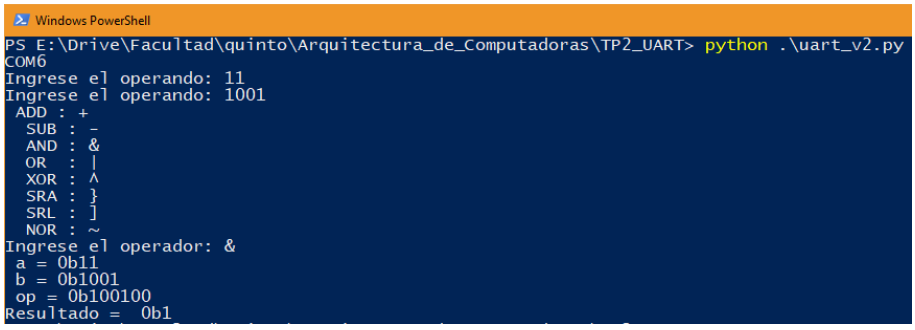

Figura 6: Prueba de funcionamiento en Python

5. Verificación con Osciloscopio

Se conectó un osciloscopio a la salida del transmisor para observar el flujo de datos luego de enviar dos operadores y un operando a la ALU. En el ejemplo de la imagen se envió lo siguiente:

- A = 0111 0000
- B = 0000 0001
- OP = 100000 (suma)

El resultado esperado es: 0111 0001. Como puede verse, a la izquierda se encuentra el canal inactivo (valor 1 lógico). Luego se observa el bit de start, y luego los bits de datos, comenzando desde el menos significativo.

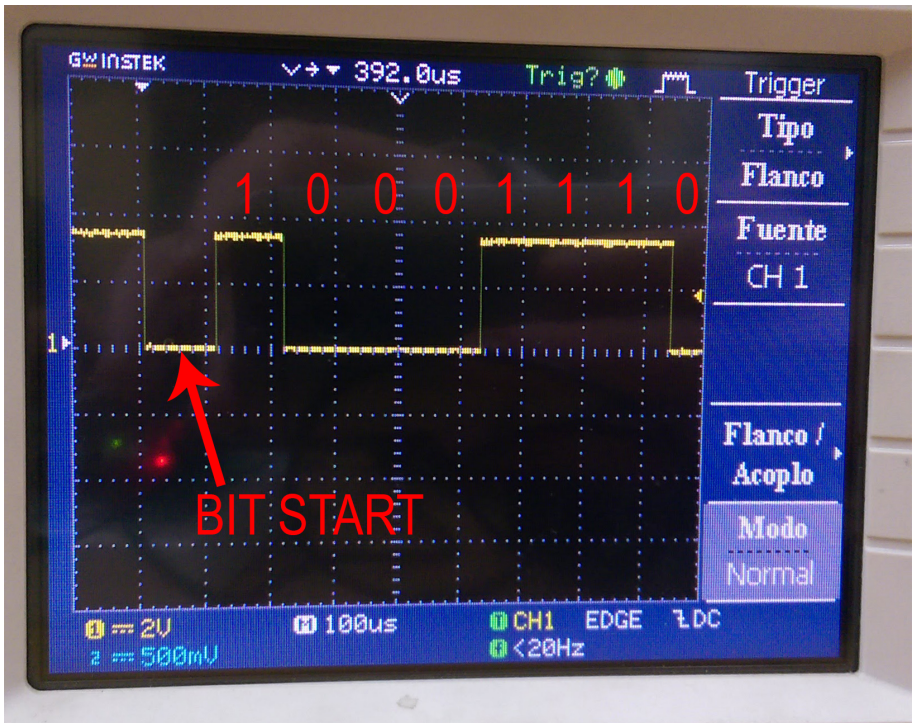

Figura 7: Resultado observado en osciloscopio