FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES
UNIVERSIDAD NACIONAL DE CÓRDOBA

Arquitectura de Computadoras

Trabajo Práctico Nº2 - UART

Aagaard Martin - Navarro Matias

Universidad Nacional de Córdoba
31 de octubre de 2019

Índice

1. Introducción

En el siguiente trabajo, se realizó la implementación en Verilog de un módulo UART. Dicho módulo se conecta a la ALU desarrollada anteriormente en esta cátedra. El sistema se completa con una PC, a partir de la cual se envían los datos y operandos, para que el módulo ALU los procese y el resultado sea devuelto a la PC.

2. Implementación del practico

Entre los dispositivos disponibles para la Implementación del UART se seleccionó la placa de desarrollo Basys 3. La principal razón de la selección fue que esta placa es el único modelo disponible y que soporta la interfaz de desarrollo Vivado. La FPGA sobre la que se trabajara es la Artix-7 de Xilinx.

3. Desarrollo

3.1. Diagrama de bloques

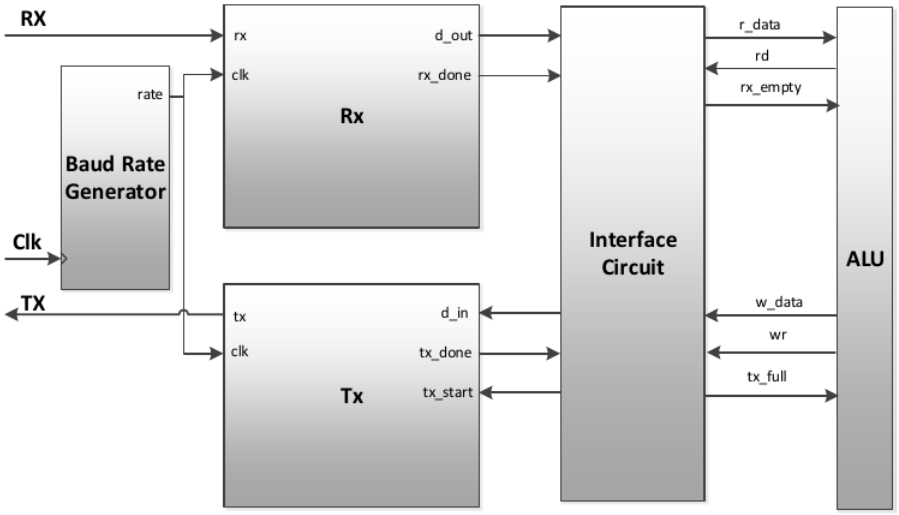

Figura 1: *Diagrama de bloques implementado*

3.2. Desarrollo del trabajo

Se definió un módulo Top, que contiene:

- Modulo UART, compuesto por:
  - Baud Rate Generator
  - Transmisor (TX)
  - Receptor (RX)
- Circuito Interfaz
- ALU

3.3. UART

3.3.1. 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 ese caso, 9600 baudios), genera ticks que permiten al módulo Rx trabajar con al señal de entrada.

3.3.2. Receptor (RX)

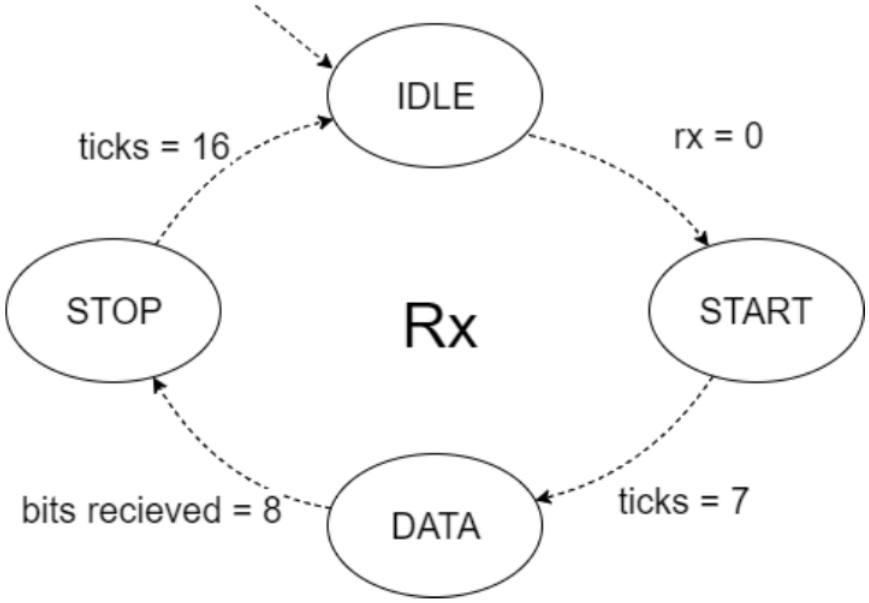

Figura 2: *Diagrama de estados del receptor*

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 y asegura que el dato no se muestrea cerca del cambio de símbolo. 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).

3.3.3. Transmisor (TX)

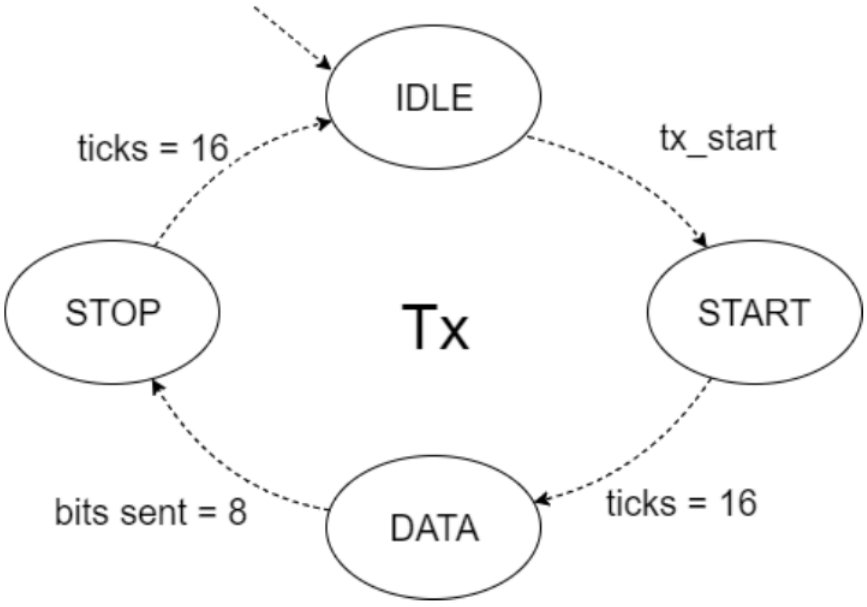

Figura 3: *Diagrama de estados del transmisor*

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.

3.3.4. Circuito Interfaz

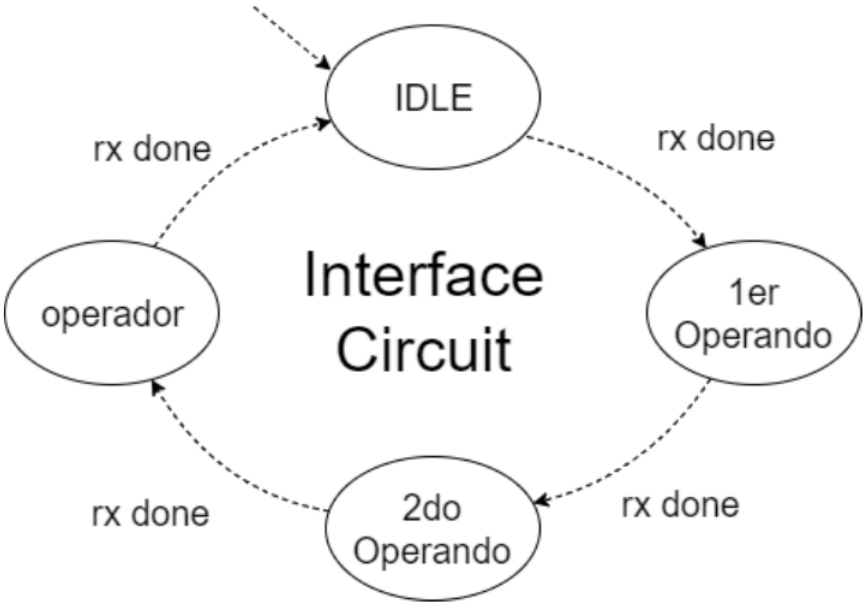

Figura 4: *Diagrama de estados del circuito interfaz*

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.

3.3.5. ALU

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

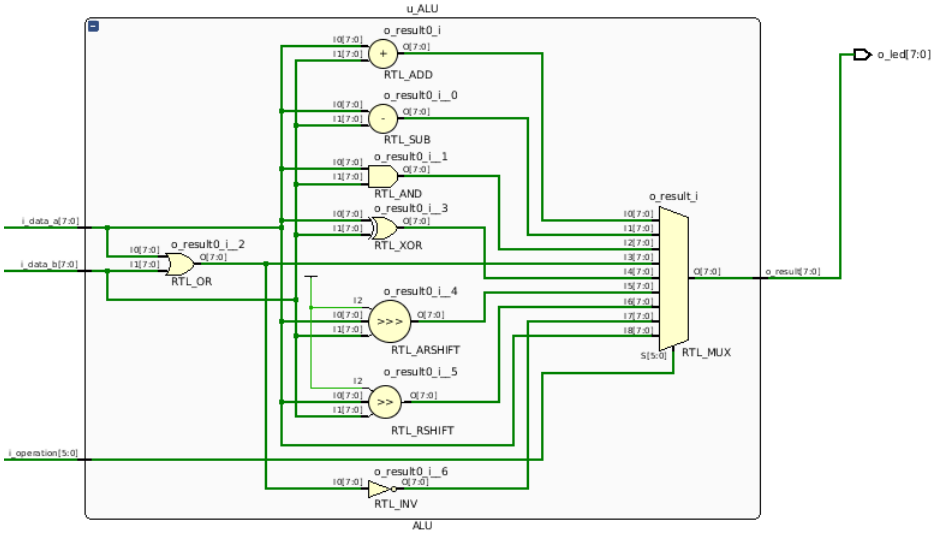

Figura 5: *Estructura RTL del módulo ALU obtenida de la herramienta Verilog*

4. Simulación de los módulos

Se realizaron las siguientes simulaciones para comprobar el correcto funcionamiento de cada uno de los bloques del sistema.

4.1. Top_UART

Se comprueba el reset y que el sistema devuelva el resultado correcto cuando se ingresan operandos y códigos de operación determinados.

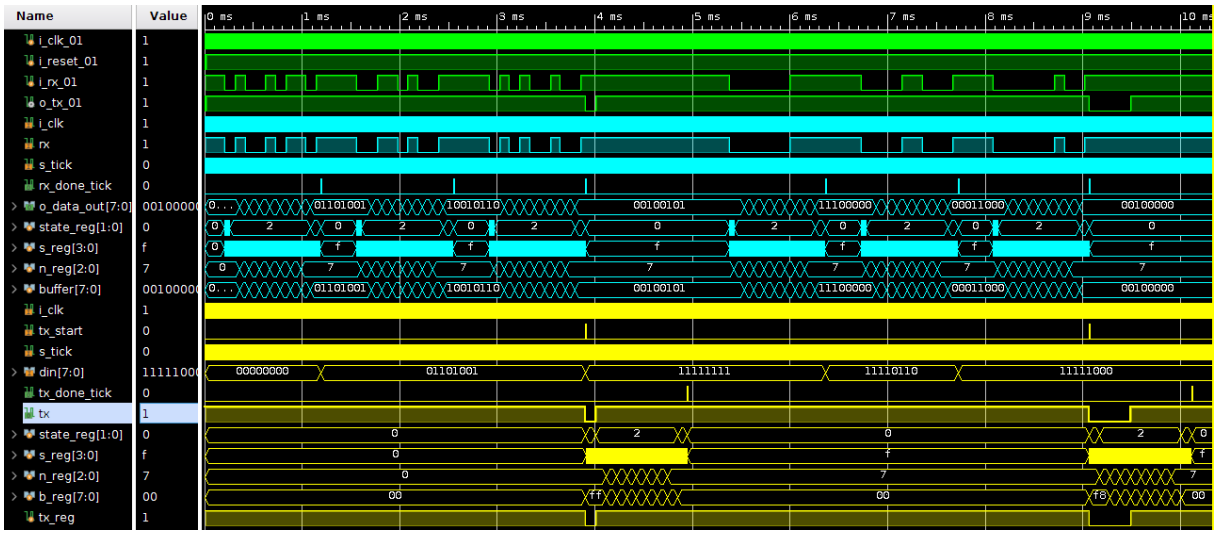

Figura 6: *Simulación del top_ UART*

4.2. Baud Rate Generator

Se comprueba la generación de los ticks con el intervalo correcto con respeto a los parámetros de entrada.

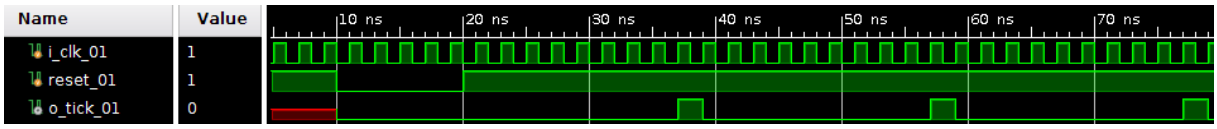

Figura 7: *Simulación del módulo Baud Rate Generator*

4.3. Receptor (RX)

Se envían bits y se comprueba el pasaje entre los estados del módulo y las salidas en cada uno de dichos estados. Se envían tramas correctas e incorrectas.

4.4. Transmisor (TX)

Se ingresa un valor a transmitir y se eleva en alto el bit de Tx_start. Se comprueba el pasaje entre los estados del módulo y las salidas en cada uno de dichos estados.

4.5. Circuito Interfaz

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.

4.6. ALU

Se comprobaron que el modulo sea capaz de realizar correctamente las operaciones, recibiendo los operandos y el operador para generar la salida esperada.

5. Interface 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.

```
~/git/Arquitectura_de_Computadoras_2019/TP2-UART-2019/Script master* 9s
❯ sudo python2 Serial_TP2_2019.py
/dev/ttyUSB1
Ingrese el operando: 11110000
Ingrese el operando: 00001111
 ADD : +
  SUB : -
  AND : &
  OR  : |
  XOR : ^
  SRA : }
  SRL : ]
  NOR : ~
Ingrese el operador: |
 a = 0b11110000
 b = 0b1111
 op = 0b100101
Resultado =  0b11111111
```

Figura 8: *Prueba de funcionamiento en Python*

6. Conclusión

Una vez terminado el práctico, a través de la herramienta Vivado se obtuvieron las especificaciones en la utilización de los recursos de la FPGA y en la potencia que consume dicho circuito instanciado. Además, en este práctico se afianzaron los conceptos en cuanto al lenguaje de descripción de hardware Verilog, incluyendo nuevos conceptos (en el contexto de verilog) como por ejemplo las maquinas de estados.

Se logro exitosamente realizar una comunicación anacrónica en serie entre la computadora y el sistema desarrollado en FPGA, pudiendo enviar operaciones simples de aritmética al modulo ALU y recibiendo el resultado de dichas operaciones en la computadora conectada.

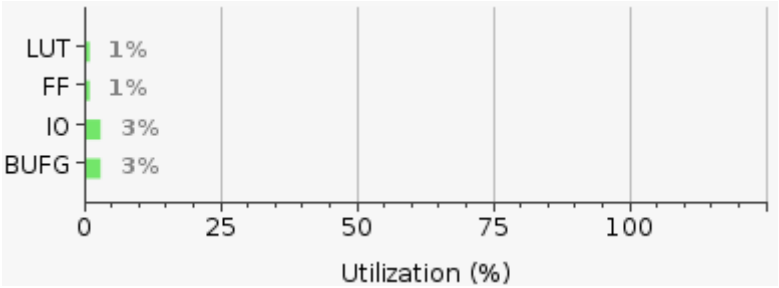

Figura 9: *Utilización de la FPGA*