

TP N°2 FPGA-UART

Alumnos:

Cristian Velazquez - cristian.velazquez@mi.unc.edu.ar

Muñoz, Saúl saul.munoz@mi.unc.edu.ar

TP-N° 2: FPGA-UART

Diciembre 2022

Introducción

El presente trabajo tiene por objeto el diseño e implementación de un módulo UART que comunica datos desde una ALU, este último módulo fué diseñado e implementado en el primer trabajo práctico.

Módulo UART

Un UART incluye un transmisor y un receptor. El transmisor es esencialmente un registro de desplazamiento especial que carga datos en paralelo y luego los desplaza poco a poco a una velocidad específica. El receptor vuelve a ensamblar los datos. La transmisión comienza con un bit de inicio, seguido de bits de datos y un bit de paridad opcional, y finaliza con bits de parada. El número de bits de datos puede ser 6, 7 u 8. El bit de paridad opcional se utiliza para la detección de errores. El número de bits de parada puede ser 1, 1,5 o 2.

Antes de que comience la transmisión, el transmisor y el receptor deben acordar un conjunto de parámetros por adelantado, que incluyen la velocidad en baudios (es decir, la cantidad de bits por segundo), la cantidad de bits de datos y bits de parada, y el uso del bit de paridad. Las velocidades de transmisión comúnmente utilizadas son 2400, 4800, 9600 y 19,200 baudios.

Ilustramos el diseño de los subsistemas de recepción y transmisión en las siguientes secciones. El diseño está personalizado para un UART con una velocidad de transmisión de 38400 baudios, 8 bits de datos, 1 bit de parada y sin bit de paridad.

Dado que no se transmite información de reloj desde la señal transmitida, el receptor puede recuperar los bits de datos sólo utilizando los parámetros predeterminados. Usamos un esquema de sobremuestreo para estimar los puntos medios de los bits transmitidos y luego los recuperamos en estos puntos en consecuencia.

Procedimiento de sobremuestreo

La tasa de muestreo más utilizada es 16 veces la tasa de baudios, lo que significa que cada bit serial se muestrea 16 veces. Suponga que la comunicación utiliza N bits de datos y M bits de parada. El esquema de sobremuestreo funciona de la siguiente manera:

1. Espere hasta que la señal entrante se convierta en 0, el comienzo del bit de inicio, y luego inicie el contador de muestreo.
2. Cuando el contador llega a 7, la señal entrante alcanza el punto medio del bit de inicio. Borre el contador a 0 y reinicie.
3. Cuando el contador llega a 15, la señal entrante avanza un bit y llega a la mitad del primer bit de datos. Recupere su valor, cámbielo a un registro y reinicie el contador.
4. El paso 3 se repite N-1 veces más para recuperar los bits de datos restantes.
5. Si se usa el bit de paridad opcional, el paso 3 se repite una vez para obtener el bit de paridad.
6. El paso 3 se repite M más veces para obtener los bits de parada.

El esquema de sobremuestreo básicamente realiza la función de una señal de reloj. En lugar de usar el flanco ascendente para indicar cuándo la señal de entrada es válida, utiliza marcas de muestreo para estimar el punto medio de cada bit. Si bien el receptor no tiene información sobre el tiempo de inicio exacto del bit de inicio, la estimación puede fallar como máximo 1/16. Las recuperaciones de bits de datos subsiguientes están desactivadas como máximo 1/16 y también desde el punto medio. Debido al sobremuestreo, la velocidad en baudios puede ser solo una pequeña fracción de la velocidad del reloj del sistema y, por lo tanto, este esquema no es apropiado para una velocidad de datos alta.

Generador de velocidad de transmisión

El generador de velocidad en baudios genera una señal de muestreo cuya frecuencia es exactamente 16 veces la velocidad en baudios designada por el UART.

Para la velocidad de 38.400 baudios, la velocidad de muestreo debe ser de 614400 (es decir, $38400 * 16$) impulsos por segundo. Dado que la velocidad del reloj del sistema es de 100 MHz, el generador de velocidad en baudios necesita un contador mod-163 (es decir, $(100 * 10 ^ 6) / (614400)$), en el que se afirma un tic de un ciclo de reloj una vez cada 163 ciclos de reloj.

Receptor UART

Para adaptarse a modificaciones futuras, se utilizan dos constantes en la descripción, la constante DBIT indica el número de bits de datos y la constante SB_TICK indica el número de ticks necesarios para los bits de stop, que es 16, 24 y 32 para 1, 1,5 y 2 bits de parada. DBIT y SB_TICK tienen un valor de 8 y 16 en este diseño.

Los estados inactivo (IDLE), inicio (START), datos (DATA) y parada (STOP), representan la situación de inactividad, el procesamiento del bit de inicio, los bits de datos y el bit de parada. s_tick es la señal de habilitación del generador de velocidad en baudios y hay 16 tics en un intervalo de bits. La máquina de estado permanece en el mismo estado a menos que se afirme la señal s_tick. Hay dos contadores, representados por los registros s y n. El registro s realiza un seguimiento del número de tics de muestreo y cuenta hasta 7 en el estado de inicio, hasta 15 en el estado de datos y hasta SB_TICK en el estado de parada. El registro n realiza un seguimiento del número de bits de datos recibidos en el estado de datos. Los bits recuperados se desplazan y se vuelven a ensamblar en el registro b. rx_done_tick se afirma durante un ciclo de reloj después de que se completa el proceso de recepción. El diagrama de estado para un receptor UART se muestran en la Figura 1.

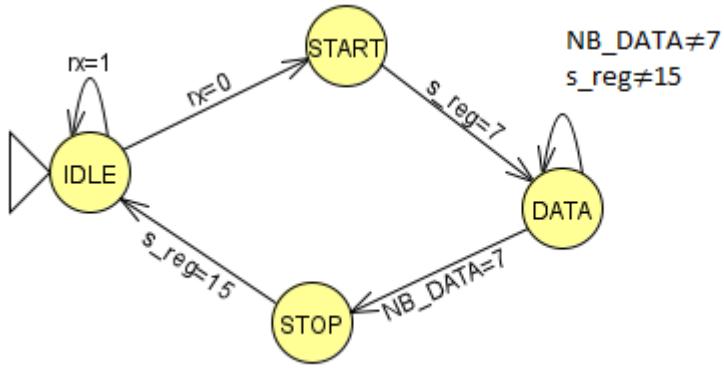


Figura 1. Diagrama de estados de un receptor

Transmisor UART

La organización del subsistema de transmisión UART es similar a la del subsistema de recepción. Consiste en un transmisor UART y en un generador de velocidad en baudios.

El transmisor UART es esencialmente un registro de desplazamiento que desplaza bits de datos a una velocidad específica. La velocidad se puede controlar mediante tics de habilitación de un ciclo de reloj generados por el generador de velocidad en baudios. Debido a que no se trata de sobremuestreo, la frecuencia de los tics es 16 veces más lenta que la del receptor UART. El transmisor UART comparte el generador de velocidad en baudios del receptor UART y usa un contador interno para realizar un seguimiento de la cantidad de tics habilitados. Se desplaza un bit cada 16 tics de habilitación.

La máquina de estado del transmisor UART es similar al del receptor UART. Despues de la afirmación de la señal de inicio de tx, se carga la palabra de datos y luego se avanza gradualmente a través de los estados de inicio, datos y parada. La afirmación de la señal `tx_done_tick` señala la finalización del proceso de recepción. El diagrama de estado para un transmisor UART se muestra en la **Figura 2**.

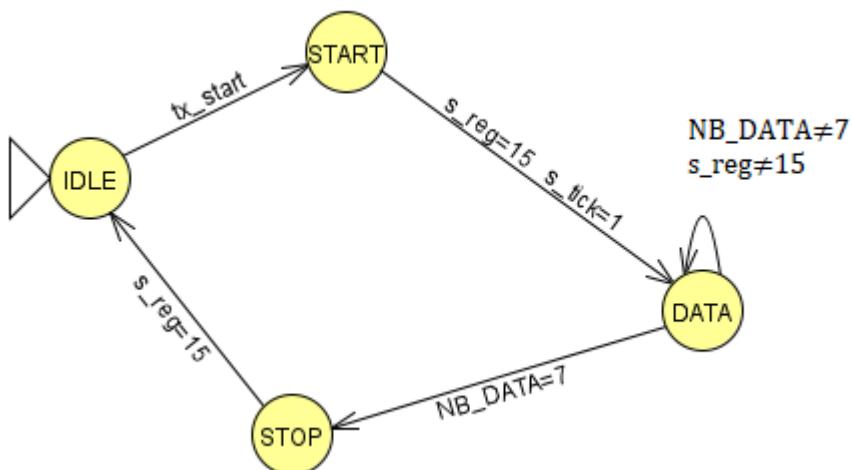


Figura 2. Diagrama de estados de un transmisor

Implementación

Síntesis

En el diagrama esquemático de la Figura 3 se puede apreciar la jerarquía de bloques y su interconexión

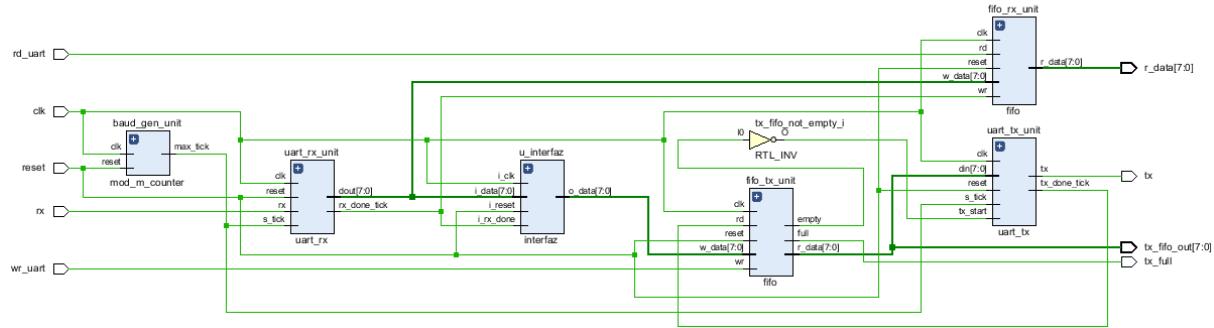


Figura 3. Diagrama esquemático del sistema

A continuación se observa sobre el interior de la interfaz donde se encuentra la alu

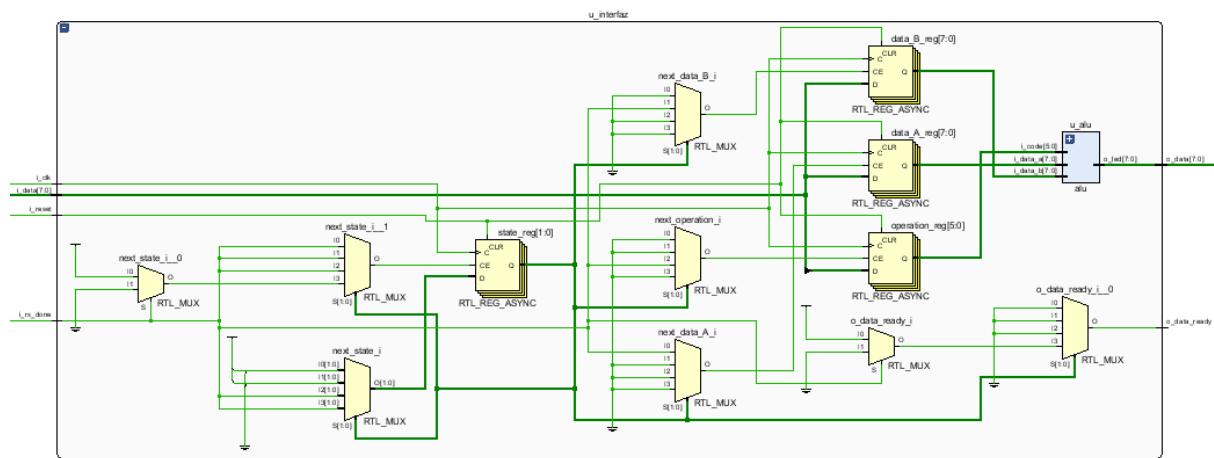


Figura 4. Diagrama esquemático de la interfaz

Ejecución del programa

En la pc se ejecuta un programita en python el cual espera que se ingrese el dato A, luego B y por último el tipo de operación deseada. Para el caso de la operación se usa el nombre de la op

En la imagen se puede ver que se envía un 3 y un 1 para realizar una resta (SUB).

```

PS E:\CRISTIAN\FACULTAD\cursadas\arquitectura> python traductor.py
Primer se envia en dato A, luego el B y por ultimo la operacion
Escribe el dato a enviar: 3
3
writed = 1
Primer se envia en dato A, luego el B y por ultimo la operacion
Escribe el dato a enviar: 1
1
writed = 1
Primer se envia en dato A, luego el B y por ultimo la operacion
Escribe el dato a enviar: SUB
SUB
writed = 1
binary: 00000010      Decimal: 2      Hex:0x2 Primer se envia en dato A, luego el B y por ultimo la operacion
Escribe el dato a enviar:

```

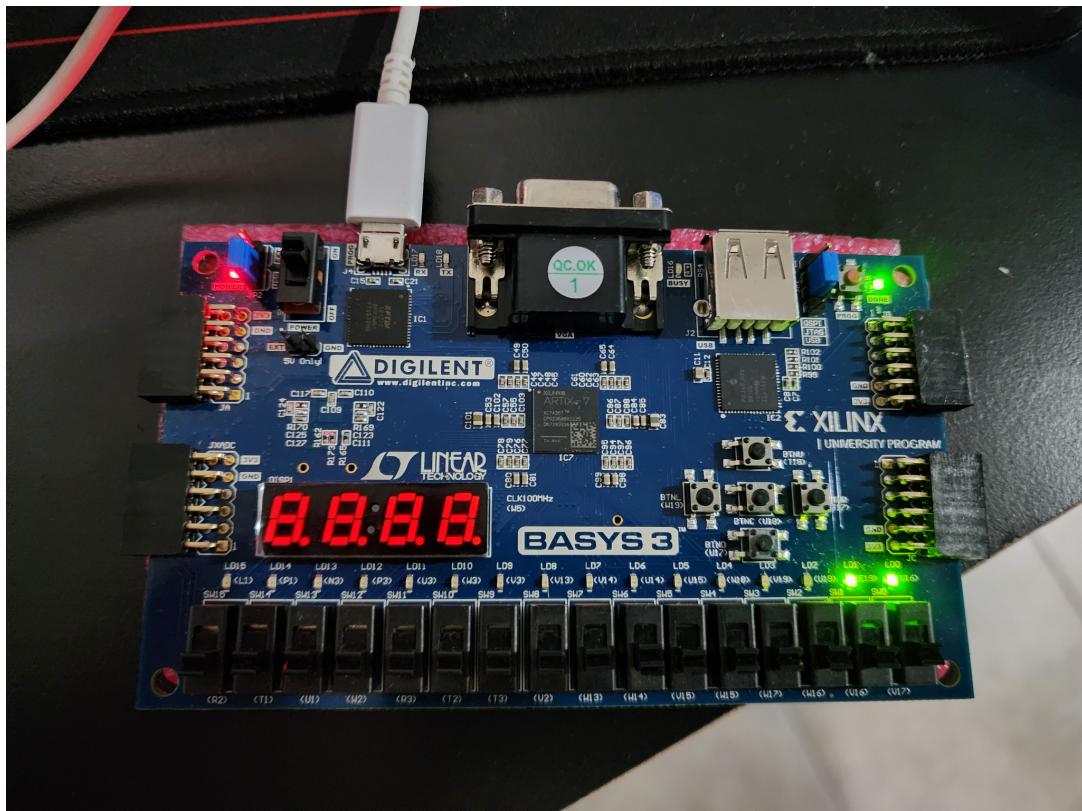
Respecto al programa que corre la pc, en este se realiza una búsqueda de ciertos valores. Estos son los números de 0 a 9 y las palabras ADD, SUB, AND, OR, XOR, SRL,SRA y NOR.

Cuando se encuentra ese rango de números y palabras, envía sus respectivos valores en ASCII, ya que cuando se escribe por ejemplo un 0, este se envía como un char 0 el cual tiene otro valor en ASCII

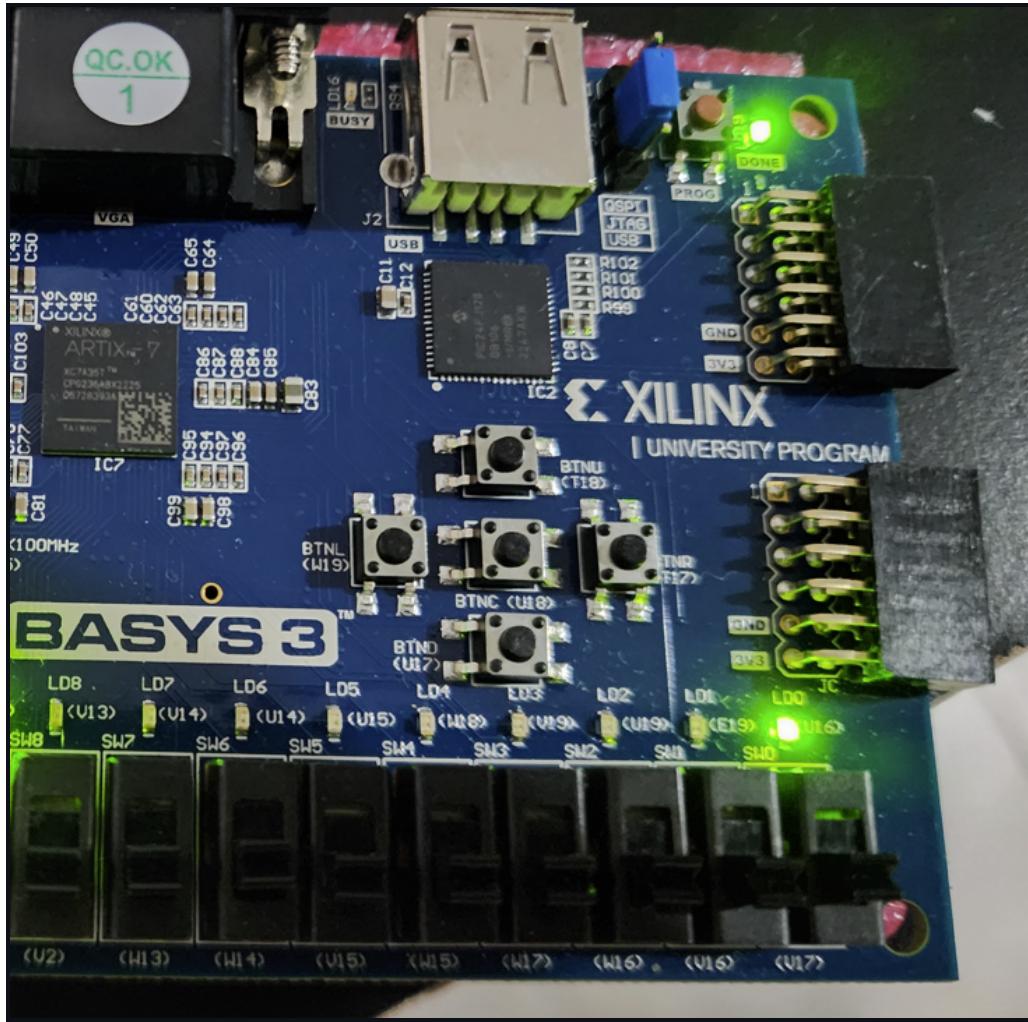
FPGA

La FPGA espera el primer dato (A), luego se debe presionar el pulsador T17 para que el FIFO espere el segundo dato (B), por último se presiona de nuevo el pulsador para el dato C (op).

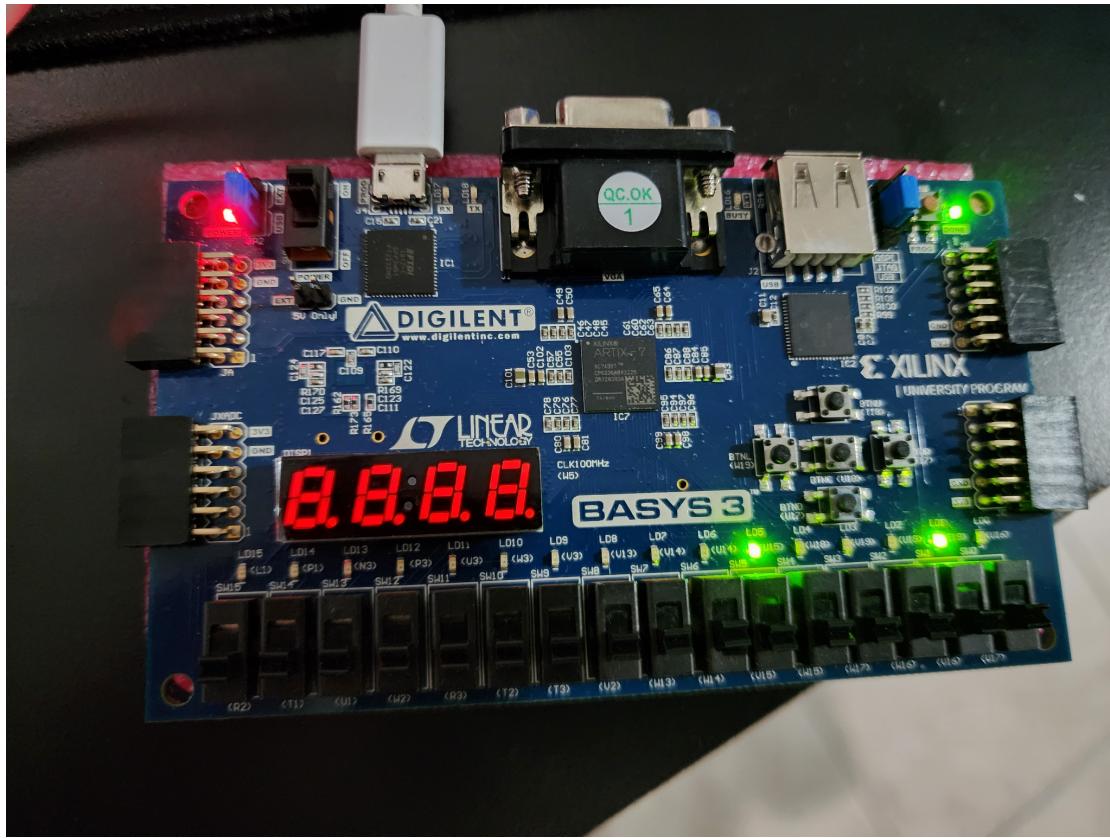
DATO A (3)



DATO B (1)



DATO C (6'b100010)



En los leds se puede ver los datos que van llegando a la fifo. Para enviar el resultado se presiona el pulsador W19. En este momento se puede ver en los leds el resultado de la salida del fifo_tx y se envia el dato por tx.

RESULTADO 2 (v3=1 y v13=0)

