Practica 3: Diseño de una ALU de 16 bits para el procesador ICAI-RiSC-16

1. Objetivos

El objetivo de esta práctica es el diseño de una ALU de 16 bits, la cual será utilizada en la ruta de datos del procesador ICAI–RiSC–16. El diseño ha de incluir también un *testbench* para verificar el funcionamiento correcto del circuito en simulación. Como es tedioso comprobar todos los resultados de la ALU "a mano", el "testbench" ha de verificar el resultado del circuito mediante sentencias `assert`.

2. Tiempo de laboratorio

1 sesión.

3. Trabajo previo

Ha de realizarse toda la fase de diseño del sistema.

4. Documentación final de la práctica

Se entregará un diagrama de bloques del diseño así como todos los esquemas y el código VHDL usado para la implantación de la ALU. También ha de incluirse el código del *testbench* usado para la verificación del circuito.

5. Introducción

En esta práctica se va a implementar una ALU similar a la que se realizó en el curso de Sistemas Digitales I en 1º de ITL. La principal diferencia es que esta ALU está pensada para ser usada en la implantación de un microprocesador, por lo que sus entradas y salidas serán de 16 bits y las operaciones serán las marcadas por la arquitectura del microprocesador, las cuales se resumen en la figura 1a.

En las siguientes secciones se exponen las operaciones a realizar por la ALU

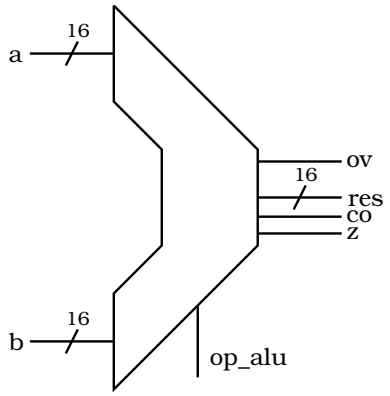
5.1. Operaciones aritméticas

La ALU ha de realizar cuatro tipo de operaciones aritméticas. En todas ellas la salida de la ALU será $\mathrm{ALU} = \mathrm{A}\ op\ \mathrm{B}$. Los operandos podrán ser números sin signo o con signo codificados en complemento a 2. En el caso de la suma recuerde que la única diferencia es a la hora de averiguar si ha ocurrido un error de desbordamiento. La resta se realizará sumando el complemento a 2 del operando B. Como requisito de diseño **es obligatorio usar un sólo sumador para realizar las operaciones de suma y resta**. La señal `co` será el acarreo del sumador, la señal `ov` se activará cuando se produzca un desbordamiento en operaciones de suma o resta con signo y la señal `z` se activará cuando el resultado de la operación sea un cero.

Como para realizar la multiplicación de números con signo es necesario extender el signo de los operandos es necesario distinguir la multiplicación con signo o sin signo. En ambos casos el resultado se truncará a 16 bits, activándose la señal `ov` en caso de que dicho resultado no pueda representarse en 16 bits.

op_alu	Operación
0000	Suma
0001	Resta
0010	Multiplicación con signo
0011	Multiplicación sin signo
0100	NAND
0101	Desplazamiento de 1 bit a izq.
0110	Desplazamiento aritmético de 1 bit a der.
0111	Desplazamiento lógico de 1 bit a der.
1000	*Set on Less Than*.

(a) Operaciones de la ALU



(b) Símbolo

Figura 1: ALU para el procesador ICAI-RiSC-16.

5.2. Operaciones lógicas

La ALU ha de realizar también cuatro tipos de operaciones lógicas. La primera es una NAND bit a bit. La segunda desplaza el operando A un bit a la izquierda, no usando el operando B. La tercera realiza un desplazamiento aritmético de un bit a la derecha del operando A. Recuerde que el desplazamiento aritmético copia el bit de signo en el bit más significativo después de desplazar. La cuarta operación es similar a la anterior pero introduciendo un cero en el bit más significativo al desplazar.

5.3. Operaciones de comparación

Las última operación a realizar por la ALU es la denominada *Set on Less Than Unsigned*, que sirve para implantar instrucciones de toma de decisión (`if`, etc.). La instrucción compara las dos entradas de la ALU: A y B y pone a 1 su salida si $\mathrm{A} < \mathrm{B}$ o a 0 en caso contrario. En la comparación se toman A y B como números sin signo.

6. Implantación del circuito

Como en todo diseño complejo, es necesario dividir el circuito en bloques más sencillos para que el diseño sea abordable. Es recomendable al menos implantar un bloque con el sumador/restador. El resto puede realizarse en un mismo archivo, aunque es recomendable separar cada sección adecuadamente para mantener el código organizado.

Por otro lado es **obligatorio** que el ancho de palabra de los operandos sea un parámetro genérico para así poder hacer una simulación exhaustiva en un tiempo razonable.

6.1. Implantación en la placa

Para verificar el funcionamiento en la placa se usaran los interruptores para introducir dos números de 5 bits, los pulsadores para introducir el código de operación y los LEDs para visualizar las salidas. Para ello será necesario generar un archivo tope de jerarquía para implantar la ALU con sólo 5 bits aprovechando que se ha diseñado el componente de forma genérica.