Universidad Nacional de Córdoba

Facultad de Cs. Exactas, Físicas y Naturales

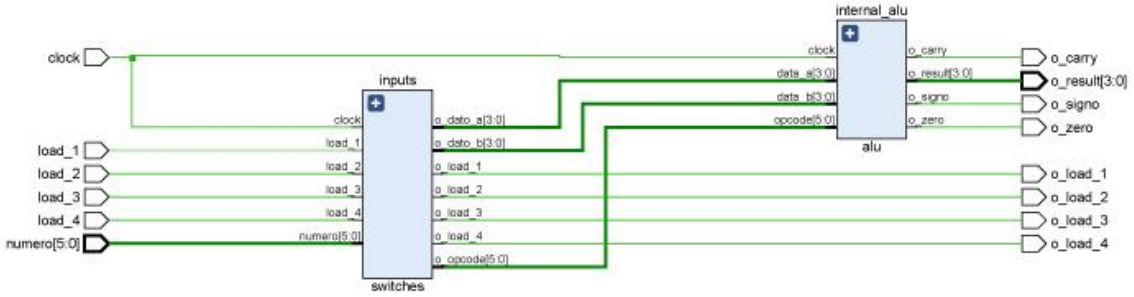
Arquitectura de Computadoras

Trabajo Práctico #1
Unidad Aritmético y Lógica

- Amallo, Sofía 41279731
- Raya Plasencia, Matías 40089058

Índice

Descripción



Se utilizó un esquema de módulos jerárquicos para permitir una abstracción de la implementación de cada uno. El módulo “switches” permite definir el valor de los registros para los datos o el opcode. El uso de los inputs `load_1, load_2 y load_3` es latchear el valor ingresado de acuerdo al destino que le corresponda, para conocer cual es el valor que se esta cargando, decidimos que cada uno tenga un led asociado para tal fin.

Una vez que se setean los valores de entrada, son enviados a la ALU, que se desarrolló siguiendo el diagrama brindado por la consigna. Se utiliza un bus de 6 bits para las operaciones, que son las siguientes:

Operación	Código
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
SRA	000011
SRL	000010
NOR	100111

Se parametrizan los tamaños de los buses para que sea más sencillo modificarlo en el caso que se desee utilizar otro tamaño de buses. En nuestro caso se utilizan buses de 4 bits para las entradas, uno de 5 bits para la salida y un tercer bus de 6 bits para el opcode, además contamos con un bit para el carry, otro para determinar si el resultado es 0 y un tercer bit para determinar si el resultado de la resta es negativo o no.

Se definieron parámetros para las operaciones, para hacer más legibles los switch-cases. Dentro de los mismos se consideraron los casos donde los operandos fueran

iguales, menores o mayores, para el caso de la resta, y así definir el bit del signo. No se utilizan números negativos, para evitar la complejidad de operaciones con signo.

Para las asignaciones se trató de minimizar el truncamiento utilizando el tamaño exacto de bits de los registros a asignar, ya que una asignación del tipo `zero = 0` indicaría el uso de un int, cuando en realidad se trata de un registro de un solo bit.

La salida del módulo ALU consiste en el resultado numérico, y flags para signo, cero y el carry.

Este es el repositorio de GitHub donde se encuentran los archivos .v y las imágenes de los test benchs.

Diagrama de bloque

El diagrama de bloque de nuestra ALU se encuentra aquí.

Test Bench

Cuando comenzamos a realizar el test bench de la suma, en la cual realizamos dos veces dicha operación, para comprobar el tema del flag de carry, nos dimos cuenta de que una vez realizada la operación teníamos que setear el opcode en 0, nuvamente, para que al introducir los valores de dato_a y dato_b, no se realice una operación que no querramos. Por este motivo es que se realiza una limpieza del opcode antes de introducir los nuevos datos. Se adjuntan imágenes por separado de la suma y la resta ya que se probaron los flags que se utilizan en ellas únicamente (dichas imágenes se encuentran en el aquí).

Para hacer la prueba asignamos diferentes parámetros en el archivo de simulación, esto nos permitió observar cómo se comportaba cada una de las funciones. Posteriormente descomentamos todas las operaciones y las probamos en su conjunto para observar la gráfica resultante.