Electrónica Digital - Laboratorio III

Implementación de Unidad Aritmética Lógica en Formato Signo-Magnitud

Kioskovic, Maksim

Centro Atómico Bariloche y Instituto Balseiro, Comisión Nacional de Energía Atómica

Arquitectura

Se requiere implementar una ALU que reciba dos bits de instrucción y dos operandos en formato signo-magnitud.

Las operaciones se implementan en formato complemento a 2. Para no procesar operaciones que no corresponden a la instrucción, se incluye una etapa de demultiplexión que fija las operaciones no seleccionadas a alta impedancia.

La figura 1 presenta un esquemático del top level implementado.

- **Formato Entrada:** Traduce la entrada en signo-magnitud al formato complemento a 2, y luego la extiende a N+1 bits.
- **Selección Entrada:** Enruta las entradas al módulo correspondiente a la instrucción.
- **Aritmética:** Implementa las operaciones aritméticas y detecta errores de overflow. Los módulos aritméticos comparten interfaz de entrada N+1 bits, salida de N+1 bits, y flag de error.
- **Selección Salida:** Multiplexa el bus de resultados y de flags de overflow a las salidas con la palabra de instrución.
- **Formato Salida:** Traduce la salida en complemeto a 2 al formato signo-magnitud.

Las instrucciónes se codifican con la siguiente tabla

Código	Instrucción	Módulo
00	$+$	`add`
01	$-$	`subt`
10	$\times$	`mult`
11	$=$	`eq`

Para la evaluación de igualdad, la salida se define como 0x00 si las entradas son diferentes y 0x01 si son iguales.

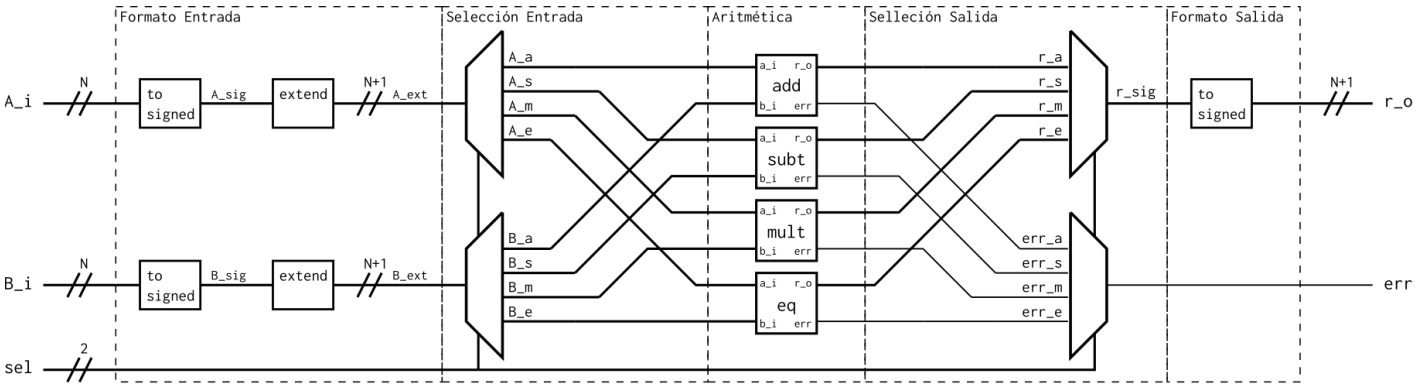

Figura 1: Esquemática del módulo top level implementado en VHDL con las correspondientes señales y sub-módulos.

Implementación

Algunas consideraciones del diseño justifican la elección de complemento a 2.

Ambas etapas de formato instancian el mismo componente `tosigned`. Esto lo permite la propiedad que el complemento a 2 es su propia inversa. El algoritmo auto-inverso implementado para cambiar el formato de un número de N bits signado es

$$Y = X_{N-1}\,\mathrm{Mux}\left\{0 : \left[0\ X_{[N-2:0]}\right],\ 1 : \overline{\left[0\ X_{[N-2:0]}\right]}_2\right\}$$

Es decir, retorna el mismo número si es positivo, y el complemento a 2 si es negativo.

Además de estandarizar las operaciones aritméticas, en complemento a 2 extender palabras y detectar errores de overflow es fácil.

Las palabras extendidas a N+1 bits se obtienen repitiendo el bit de signo

$$A_{ext} = \left[A_{N-1} \quad A_{[N-1:0]}\right]$$

El resultado de una operación tiene overflow si la palabra extendida de resultado cambia de signo al quitarle los bits de extensión, lo que sucede si los respectivos bits más significativos no coinciden, por lo que el flag de overflow se implementa con un XOR.[1]

[1] Una consideración especial se hace si el resultado es número más negativo -2^{N-1} que no es overflow en complemento a 2, pero va a ser un resultado incorrecto al convertirlo a signo-magnitud.

Tests

Algunos de los tests más importantes del diseño son el test del módulo `tosigned` y de los módulos aritméticos, además del módulo top level.

Para probar el módulo `tosigned` se instanció un componente de 3 bits y se probaron las entradas por extensión, generando la figura 2.

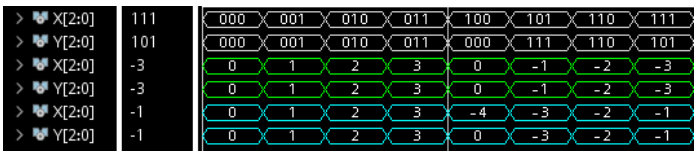

Figura 2: Simulación de un componente `tosigned` de 3 bits. En gris, las señales en radix binario. En verde la conversión a `signed`. En cian la conversión a `sign-magnitude`.

Se valida que la conversión es correcta exceptuando la conversión del número $-2^{N-1} = -4$, que conviete a 0, este es el resultado esperado y la responsabilidad de considerarlo como resultado inválido recae sobre los módulos aritméticos.

Se toma el ejemplo el test del multiplicador, se instanció un componente de 4 bits y se probaron diferentes combinaciones de signos de entrada que retornen resultados esperados con y sin overflow, multiplicación por 0, y el caso especial de una operación con resultado $-2^{N-1} = -8$. La simulación genera la figura 3.

Figura 3: Simulación de un componente `mult` de 4 bits. Entradas en verde y salidas en cian expresadas en radix `signed`.

Notar como el módulo está diseñado para encender el flag de overflow en el resultado más negativo a pesar de ser un resultado válido en su formato, anticipando que el resultado va a ser convertido a signo-magnitud.

Finalmente el test del módulo top level que cubre los casos mínimos de prueba y un caso adicional, el caso en el que el resultado es $-2^{N-1} = -16$ que va a ser error por no poder ser representado en formato signo-magnitud de 5 bits.

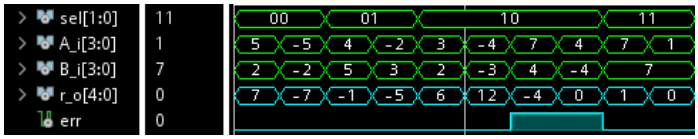

Figura 4: Simulación del componente `ALU` de 4→5 bits. Entradas en verde y salidas en cian. Los valores numéricos expresados en radix `sign-magnitude` y el código de instrucción en radix binario.