La ALU

Área de Arquitectura y Tecnología de Computadores – Versión 1.0.882, 04/02/2022

Índice

Objetivos

El objetivo de esta sesión práctica es comprender el funcionamiento de una ALU básica análoga a la empleada en el Computador Teórico (CT), un computador didáctico muy sencillo que se estudiará en la asignatura. Sobre esta ALU se llevarán a cabo las operaciones aritméticas de suma y resta, así como las operaciones lógicas AND, OR y XOR.

Como herramienta de prácticas se empleará el simulador de circuitos digitales introducido en la sesión anterior, el cual nos permitirá analizar la ALU no sólo como una "caja negra", sino a través del funcionamiento de sus componentes internos.

Conocimientos y materiales necesarios

Para aprovechar adecuadamente esta sesión de prácticas, el alumno necesita:

- Tema 2 del libro *Computadores y Redes*, *Sistemas Digitales*, en concreto el apartado 2.2.7, en el cual se describe el funcionamiento externo de la ALU. El alumno debe leer con detenimiento este apartado antes de asistir a la sesión práctica.
- Llevar a clase una memoria USB, o dispositivo análogo, para almacenar los circuitos que se desarrollarán en la práctica.

Durante la sesión se plantearán una serie de preguntas que deben responderse en el correspondiente cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923) del Campus Virtual. El cuestionario se puede abrir en otra pestaña del navegador pinchando en el enlace mientras se mantiene pulsada la tecla `Ctrl`.

1. Introducción

La ALU es un componente fundamental del computador, pues es la unidad encargada de llevar a cabo las operaciones aritméticas y lógicas. Por ejemplo, cada vez que se suman dos números dentro del computador estos números se llevan como entradas a la ALU, la cual genera a la salida no sólo el resultado de la suma, sino además unos bits de estado denominados *flags* que proporcionan información sobre el resultado de la operación.

La figura 1 muestra las entradas y salidas de la ALU de 4 bits empleada en esta práctica. Como entradas recibe los operandos `A` y `B`, ambos de 4 bits, así como 4 bits que seleccionan la operación a realizar: `Op1`, `Op0`, `Cin` y `Compl-1`. Como salidas se genera el resultado de 4 bits, `S`, y 4 bits de estado: `ZF`, `CF`, `OF` y `SF`.

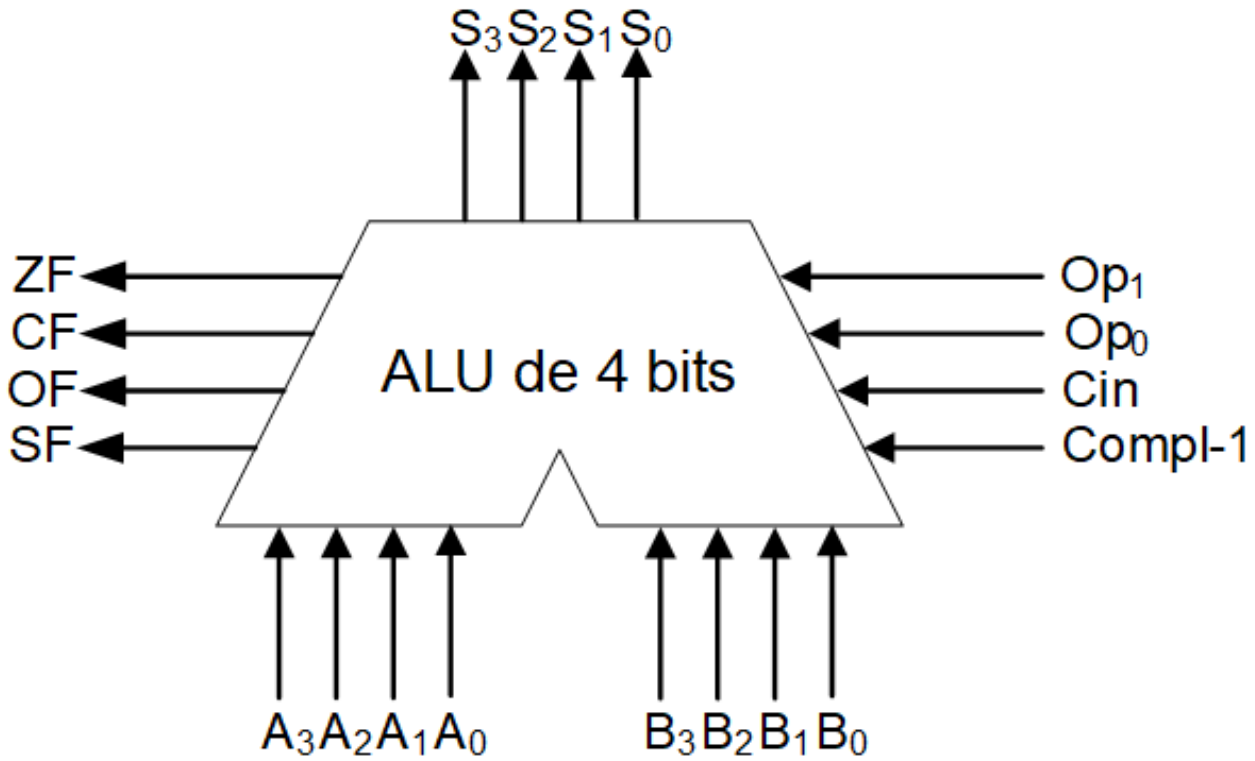

Figura 1. Entradas y salidas de una ALU de 4 bits

La tabla 1 indica la operación realizada en función del valor de los bits de operación.

Tabla 1. Tabla de operaciones de la ALU

Op1	Op0	Cin	Compl-1	Operación
0	0	0	0	`S = A AND B`
0	1	0	0	`S = A OR B`
1	0	0	0	`S = A XOR B`
1	1	0	0	`S = A + B`
1	1	1	0	`S = A + B + 1`
1	1	1	1	`S = A - B`

Para comprender el funcionamiento de la ALU se realizarán diferentes operaciones, siguiendo estos pasos:

- La operación planteada será realizada manualmente por el alumno, esto es, sin ayuda del simulador. Como resultado se obtendrán los 4 bits del resultado `S` y los 4 bits de estado: `ZF`, `CF`, `OF` y `SF`.
- Se realizará la operación en el simulador, para lo cual será necesario activar de forma adecuada las entradas con los bits necesarios. El resultado debería ser el mismo que el obtenido de forma manual.
- Con la ayuda del simulador se profundizará en el funcionamiento de la ALU observando el funcionamiento de sus componentes internos.

2. Funcionamiento de la ALU con operaciones de suma

En primer lugar, vamos a realizar la operación de suma con los operandos `A=0110` y `B=0111`.

- Realiza a mano sobre el papel la operación de suma de los operandos `A` y `B`. ¿Qué valor toma `S`, el resultado de la suma? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 1.
- ¿Qué valor deben tomar los bits de estado `ZF` y `SF`? ¿Por qué? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 2.
- Interpretando los operandos `A`, `B` y el resultado `S` como números naturales, ¿el resultado de la suma es correcto o incorrecto? ¿Qué valor debe tomar entonces el bit `CF` de acarreo? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 3.
- Interpretando ahora los operandos `A`, `B` y el resultado `S` como números enteros, ¿el resultado de la suma es correcto o incorrecto? ¿Por qué? ¿Qué valor debería tomar entonces el bit `OF` de *overflow*? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 4.

A continuación la operación de suma se realizará con la ayuda del simulador.

- Descarga el archivo `alu4.zip` del Campus Virtual y descomprímelo en tu directorio de trabajo. Dentro del directorio `alu4` creado aparecen varios archivos. Uno de ellos es el archivo `alu4_layout.dig`, el cual debes abrir empleando el simulador de circuitos digitales. En el área de trabajo observarás un componente de nombre `alu4` con sus entradas y salidas unidas a puntos de conexión, tal como se muestra en la figura 2.

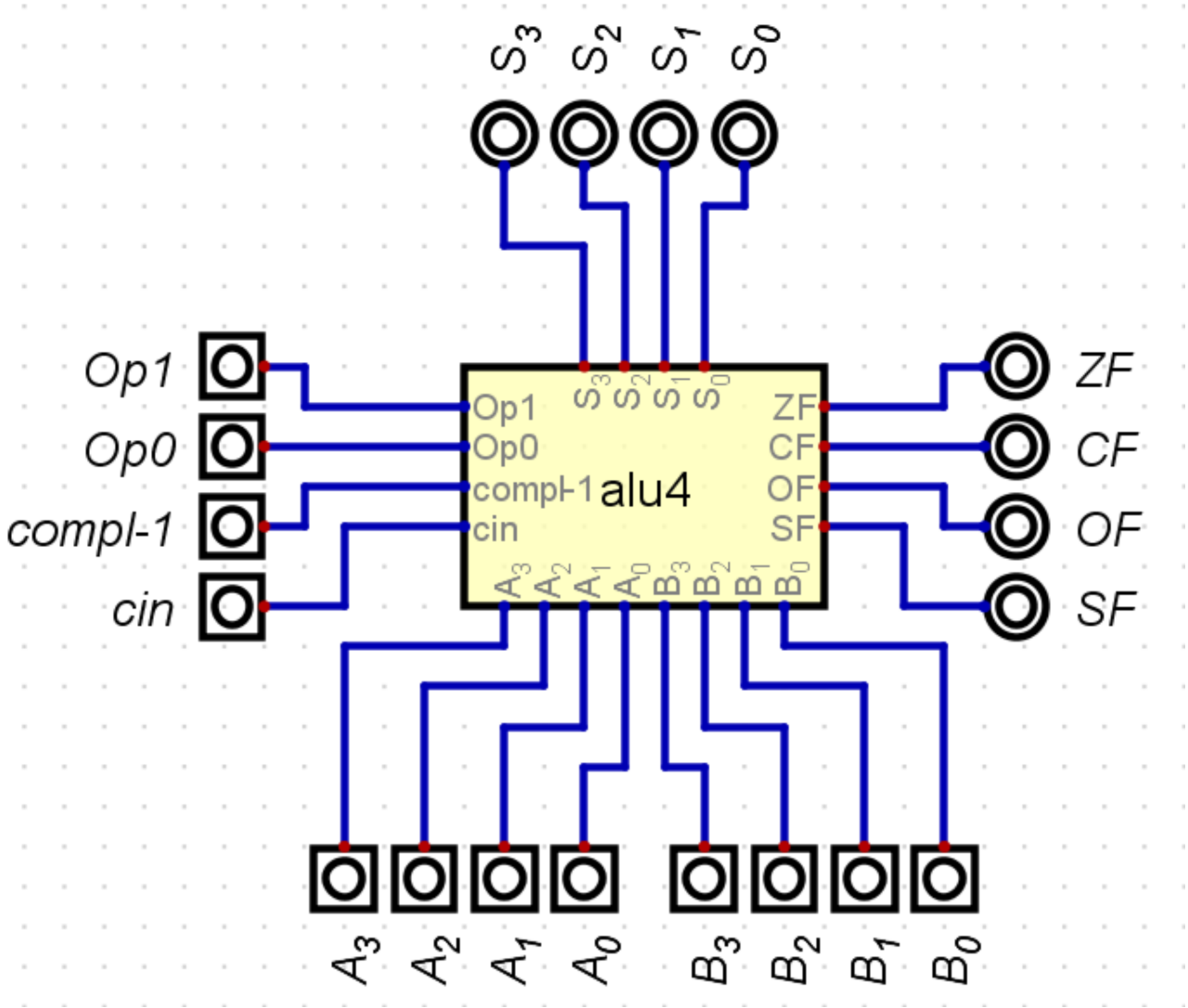

Figura 2. La ALU de 4 bits en el simulador

- En el simulador, activa el modo simulación.
- Coloca en las entradas correspondientes los valores `1` y `0` adecuados para realizar la suma con los operandos `A=0110` y `B=0111`.
- Observa el resultado de la suma y el de los bits de estado. ¿Coinciden con los que obtuviste manualmente?
- Desactiva el modo simulación.
- Haz clic con el botón derecho del ratón sobre el componente `alu4`. En la ventana que aparece haz clic con el botón izquierdo en la opción Abrir circuito. El simulador muestra el interior de la ALU en el cual se observan sus componentes y los cables de conexión como se muestra en la figura 3.

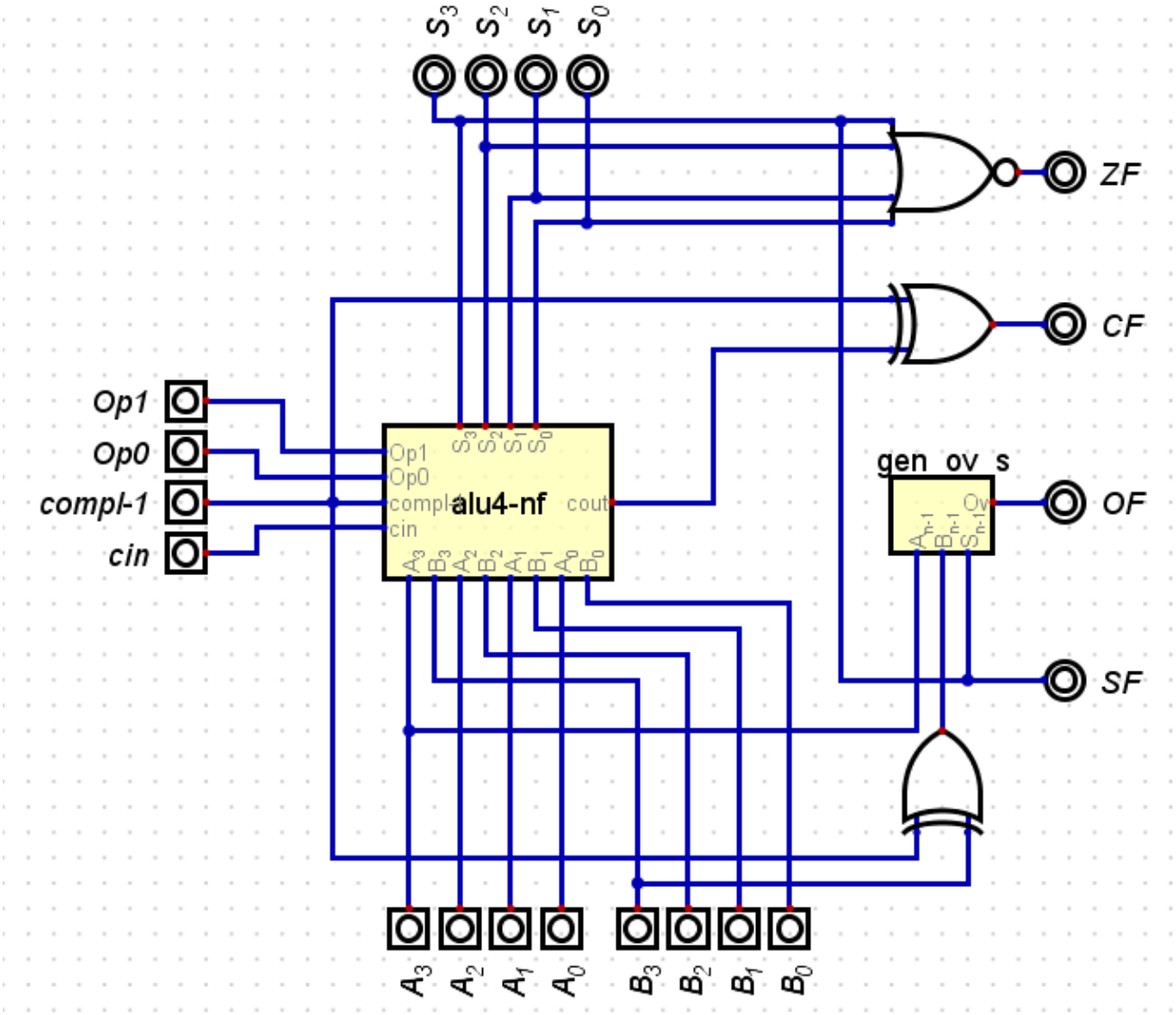

Figura 3. Interior de la ALU de 4 bits

El resultado de la suma, `S`, es generado por el componente `alu4-nf`, el cual implementa la ALU de 4 bits sin *flags* (bits de estado). En la parte derecha aparecen 4 componentes que implementan los bits de estado. Debe tenerse en cuenta que el bit de estado `SF` se obtiene directamente del bit más significativo del resultado, pues es el que indica si el resultado es negativo (`SF` =1) o positivo (`SF` =0). Estos son los componentes que aparecen a la derecha:

- Una puerta NOR de 4 entradas. Esta puerta genera un uno a la salida sólo cuando todas sus entradas están a cero, es decir, cuando el resultado de la suma es `S=0000`. Este es justo el resultado esperado del bit de estado `ZF`.
- Una puerta XOR que genera el bit de acarreo `CF`. Durante las operaciones de suma la entrada de operación `compl-1` toma el valor 0, por lo que el bit de acarreo coincide con el proporcionado por la ALU de 4 bits sin *flags*, es decir, `CF = cout`. Si el bit `compl-1` fuese 1, como ocurre en el caso de las operaciones de resta, el bit de acarreo `CF` sería justo `cout` negado.
- Para la detección de *overflow* se emplea el generador de *overflow* para la suma, de nombre `gen_ov_s`, y la puerta XOR que está justo debajo. La función de la puerta XOR es invertir el bit más significativo del operando `B` cuando se hace una operación de resta, es decir, cuando `compl-1` es 1. En el caso de la suma no se invierte el bit de signo de `B`, por lo que `gen_ov_s` recibe como entradas el bit más significativo del operando `A`, el bit más significativo del operando `B` y el bit más significativo del resultado `S`. La salida `Ov` del generador de *overflow*, bit `OF`, debe reflejar la coherencia entre los signos de los operandos de entrada y el resultado. Es decir, indica si el resultado de la suma es correcto interpretando los operandos y el resultado como enteros en complemento a 2.

- En el nuevo circuito que has abierto, `alu4.dig`, activa el modo simulación.
- Vuelve a colocar en las entradas correspondientes los valores de `1` y `0` adecuados para realizar la suma con los operandos `A=0110` y `B=0111`.
- Observa el resultado y cómo se generan las salidas correspondientes a los bits de estado.
- En el caso del bit de estado `ZF`, ¿qué valores toman las entradas y salidas de la puerta NOR? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 5.
- En el caso del bit de estado `CF`, ¿qué valor toman los bits `cout` y `compl-1`? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 6.
- En el caso del bit de estado `OF`, ¿qué signos tienen `A`, `B` y `S`? ¿Es coherente el signo de `S` con el de los operandos `A` y `B`? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 7.

Vamos a profundizar un paso más en el diseño del circuito de la ALU.

- Desactiva el modo simulación en el circuito `alu4.dig`, si no lo has hecho ya.
- Haz clic con el botón derecho del ratón sobre el componente `alu4-nf`. En la ventana que aparece elige de nuevo la opción Abrir circuito. Aparecerá ahora un nuevo circuito que representa la estructura interna de la ALU de 4 bits sin bits de estado, como se muestra en la figura 4.
- En el nuevo circuito que has abierto, `alu4-nf.dig`, activa el modo simulación.
- Vuelve a colocar en las entradas correspondientes los valores de `1` y `0` adecuados para realizar la suma con los operandos `A=0110` y `B=0111`.
- Observa el resultado y cómo ahora solo estaría disponible el acarreo de salida de la ALU, `cout`.

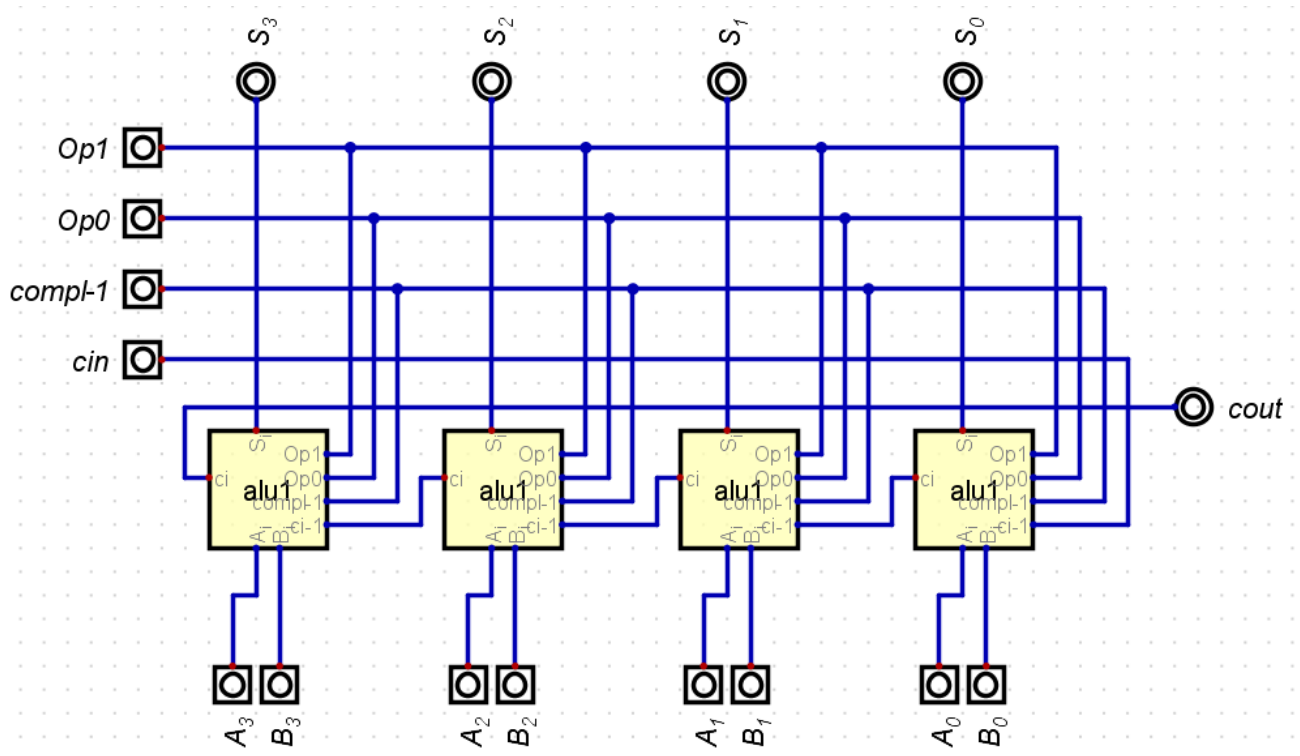

Figura 4. Interior de la ALU de 4 bits sin flags

En el nuevo circuito, se observa cómo las operaciones que realiza la ALU se realizan bit a bit mediante circuitos más simples. Para entender el porqué de esta implementación, la figura 5 ilustra cómo la operación de suma anterior se puede llevar a cabo con 4 ALU de 1 bit, cada una de las cuales trabaja sobre una pareja de bits de `A` y `B`.

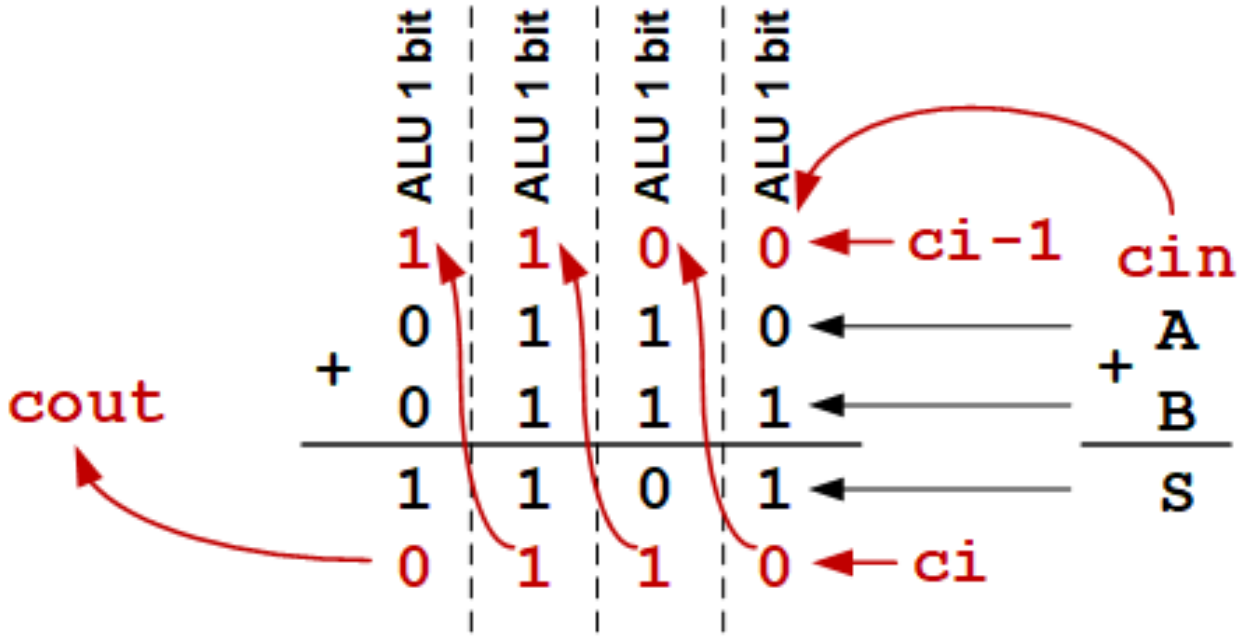

Figura 5. Ejemplo de suma de 4 bits

Así por ejemplo, la ALU de 1 bit menos significativa, la que está más a la derecha, realiza la operación con los bits A_0 y B_0, generando un resultado S_0 y un acarreo de salida `ci` que pasa a ser el acarreo de entrada `ci-1` de la ALU de 1 bit siguiente. El acarreo de salida de la ALU de 1 bit más significativa será el acarreo de salida `cout` de la ALU de 4 bits sin *flags*. De forma análoga, el acarreo de entrada de la ALU menos significativa es el acarreo de entrada `cin` de la ALU de 4 bits sin *flags*. Resulta conveniente reseñar que todas las ALU de 1 bit emplean los mismos bits de operación `Op1`, `Op0` y `compl-1`. Esta estructura es razonable si se piensa cómo se lleva a cabo por ejemplo una suma de 4 bits como la indicada en la figura 5.

- Desactiva el modo simulación en el circuito `alu4-nf.dig`, si no lo has hecho ya.
- Haz clic con el botón derecho del ratón sobre la ALU de 1 bit (componente `alu1`) más significativa, la que aparece más a la izquierda. En la ventana que aparece elige de nuevo la opción Abrir circuito. Aparecerá su estructura interna, como se muestra en figura 6.

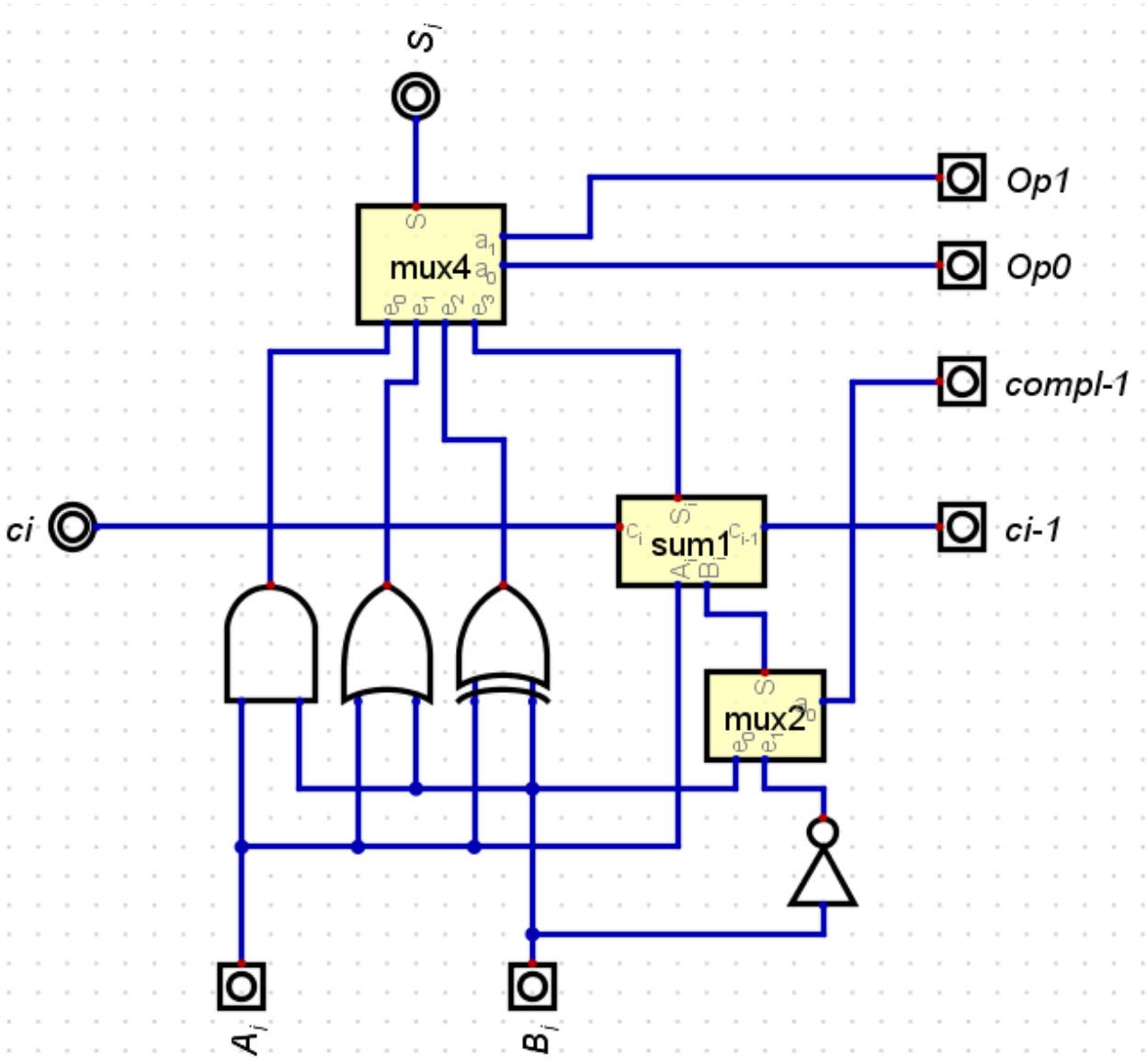

Figura 6. Circuito interno de la ALU de 1 bit más significativa

El funcionamiento de la ALU de 1 bit es simple. Realiza 4 operaciones a la vez sobre los bits A_i y B_i, los cuales en nuestro caso coinciden con A_3 y B_3. Las operaciones son AND, OR, XOR y SUMA. Los resultados de las 4 operaciones llegan a un multiplexor de 4 canales de entrada. Con las líneas `Op1` y `Op0` se selecciona la operación deseada. Por ejemplo, para seleccionar la suma hay que poner en estas líneas la combinación `11` pues el resultado del sumador de 1 bit llega a la entrada e_3 del multiplexor.

Durante las operaciones de suma la entrada de operación `compl-1` toma el valor 0, por lo que a la entrada B_i del sumador llega B_3 (en caso contrario llegaría B_3 negada, como ocurre en las operaciones de resta).

El funcionamiento interno de los multiplexores, `mux2` y `mux4`, y del sumador de 1 bit `sum1` es conocido, por lo que no se profundizará dentro de ellos.

3. Funcionamiento de la ALU con operaciones de resta

En este apartado se realizará la operación de resta con los mismos operandos anteriores, `A=0110` y `B=0111`.

- Realiza a mano sobre el papel la operación de resta de los operandos `A` y `B`. Recuerda que para llevar a cabo la resta debes sumarle al operando `A` el complemento a 2 del operando `B`, tal como puedes observar en la figura 7. ¿Qué valor toma `S`, el resultado de la resta? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 8.

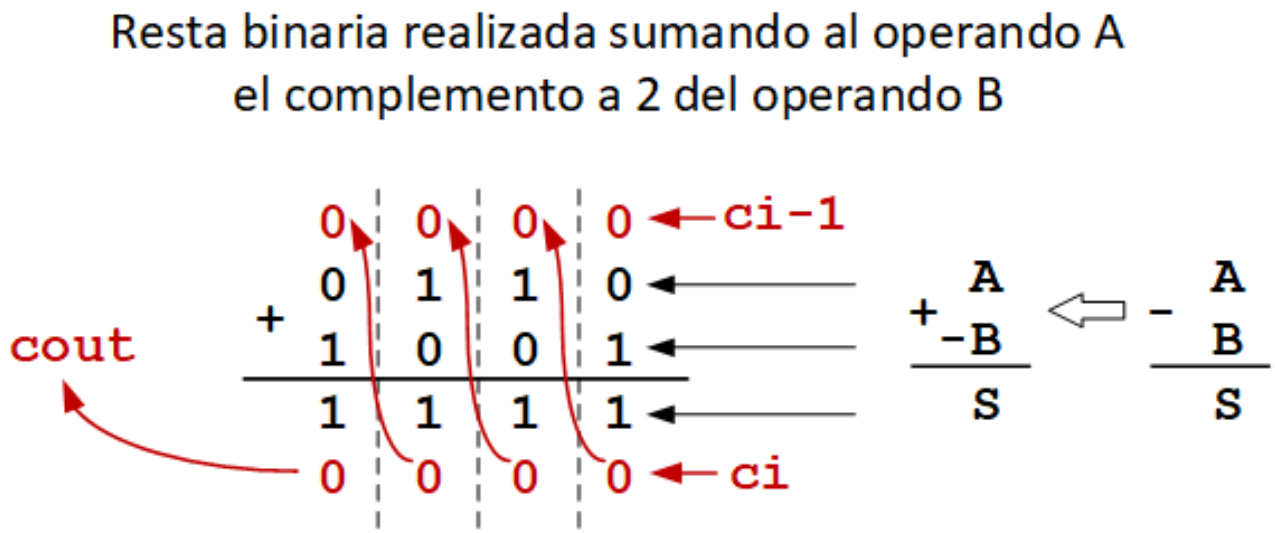

Figura 7. Ejemplo de resta de 4 bits realizada "manualmente"

- ¿Qué valor deben tomar los bits de estado `ZF` y `SF`? ¿Por qué? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 9.
- Interpretando los operandos `A` y `B` como números naturales, ¿cuáles serían respectivamente sus valores? ¿es posible llevar a cabo la operación de resta? ¿Por qué? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 10.
- Al sumar al operando `A` el complemento a 2 del operando `B`, ¿se ha producido acarreo? ¿cuál es el valor del resultado `S` interpretado como número natural? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 11. Cuando no se produce acarreo al hacer esta operación con el complemento a 2, la operación de resta original no es posible entre los operandos interpretados como naturales. En caso contrario, si se genera acarreo en la suma con el complemento a 2, sí es posible la operación de resta original.
- Interpretando ahora los operandos `A` y `B` como números enteros, ¿cuáles serían respectivamente sus valores? ¿cuál es el valor del resultado `S` interpretado como número entero? ¿el resultado de la resta es correcto o incorrecto? ¿Por qué? ¿Qué valor debería tomar entonces el bit `OF` de overflow? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 12.

A continuación la operación de resta se realizará con la ayuda del simulador.

- Abre con el simulador de circuitos digitales el archivo `alu4.dig`, si no lo tienes abierto ya.
- Activa el modo de simulación. Coloca en las entradas correspondientes los valores de `1` y `0` adecuados para realizar la resta con los operandos `A=0110` y `B=0111`.
- ¿Qué valor toma el bit de acarreo a la salida de la ALU de 4 bits sin *flags*? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 13. ¿Qué valor toma el bit de acarreo a la salida de la ALU de 4 bits completa? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 14. Observa cómo en el caso de resta (`compl-1` es 1) el valor del bit de acarreo se invierte para indicar si el minuendo es menor que el sustraendo.
- Observa el resultado de la resta y el de los bits de estado. ¿Coinciden con los que obtuviste manualmente al sumar `A` más el complemento a 2 de `B`? Deberían coincidir todos excepto el bit de acarreo. La razón es que en el caso de resta la ALU genera el bit de acarreo para indicar si el minuendo es menor que el sustraendo, interpretados minuendo y sustraendo como naturales. Cuando el minuendo es menor que el sustraendo la ALU pone el bit de estado `CF` a 1 indicando un error en la resta de naturales. Cuando la ALU genera un bit de estado `CF` a 0 esto indica que la operación interpretada entre naturales es correcta; si se trata de suma el resultado está dentro del rango y si se trata de resta el minuendo es mayor que el sustraendo.
- Desactiva la simulación.

Vamos a repetir ahora la operación a nivel de los componentes internos de la ALU.

- Pulsa con el botón derecho sobre el componente `alu4-nf`, elige la opción Abrir circuito. El simulador muestra el interior de la ALU en el cual se observan sus cuatro ALU elementales y los cables de conexión.

El circuito digital de la ALU realiza la resta de forma diferente a como la hemos hecho sobre el «papel». La ALU para realizar la resta, suma el operando `A` con el negado del operando `B` (`~B`), que también se denomina complemento a 1 del operando `B`. Esta suma se realiza bit a bit en cada ALU elemental. Para completar la operación de resta, se suma `1` al bit memos significativo de los operandos. Esto equivale a activar la señal `ci-1` de la ALU elemental menos significativa, es decir, la de más a la derecha. El proceso puede verse esquematizado en la figura 8. Como puedes deducir, el complemento a 2 que realizamos en la resta sobre el «papel» equivale a realizar el complemento a 1 + 1.

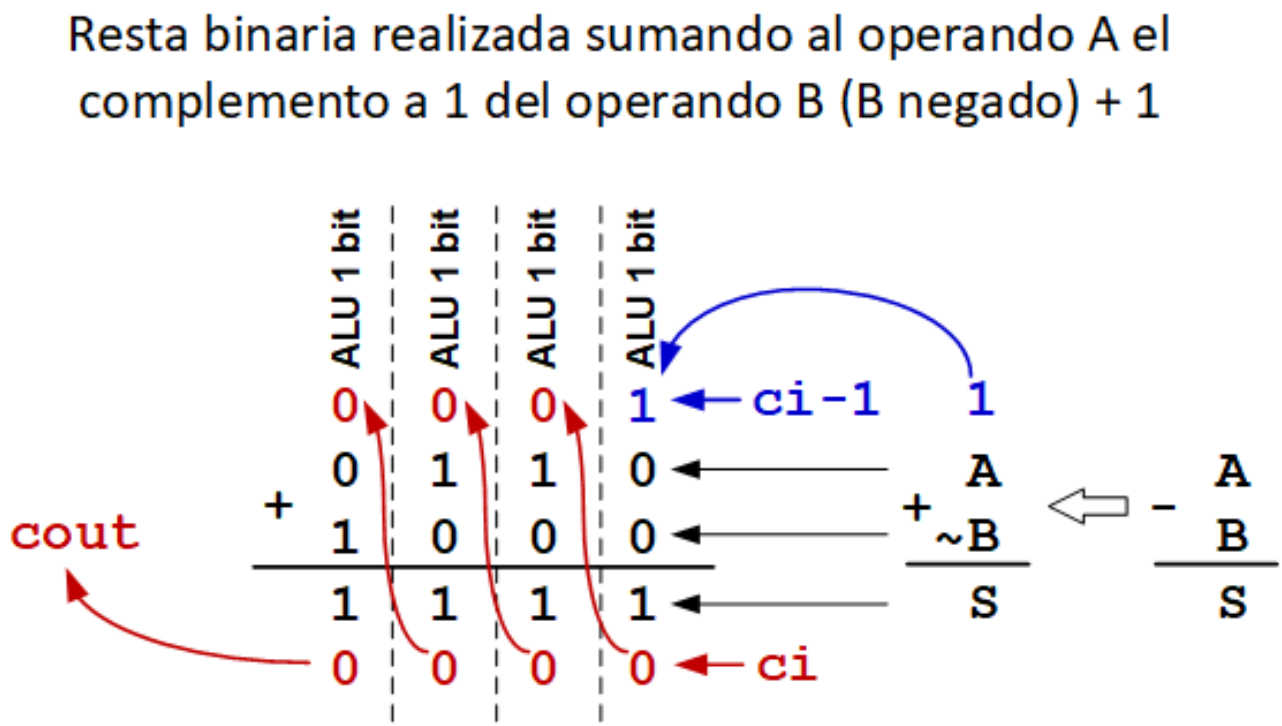

Figura 8. Ejemplo de resta de 4 bits realizada por la ALU

Vamos a centrarnos ahora en lo que sucede en la ALU elemental menos significativa.

- En el circuito `alu4-nf.dig`, observa cómo la línea de entrada `cin` entra solamente en el componente `alu1` situado más a la derecha, la menos significativa. En cambio las otras tres señales de control (`Op1`, `Op0` y `compl-1`) llegan a todos los componentes `alu1`.
- Pulsa con el botón derecho sobre el componente `alu1` situado más a la derecha en el circuito `alu4-nf`. Esta ALU es la que opera con los bits A_0 y B_0 de los operandos. Elige la opción Abrir circuito.
- Activa el modo de simulación. Coloca en las entradas A_i y B_i los valores correspondientes a los bits A_0 y B_0 de los operandos `A` y `B`. Coloca en las entradas de control los valores necesarios para ordenar la operación de resta: `Op1` = `1`, `Op0` = `1`, `compl-1` = `1` y `ci-1` = `1`.
- ¿Qué operación se está seleccionada dentro de la ALU de 1 bit? Responde en el cuestionario (https://www.campusvirtual.uniovi.es/mod/quiz/view.php?id=143923): pregunta 15.
- Observa lo que sucede. La señal `compl-1` llega al multiplexor de dos entradas haciendo que en lugar de tomar el valor del operando B_i tome el de su negado ~ (B_i). Es decir, la señal `compl-1` indica a las ALU elementales que obtengan el complemento a 1 del operando `B`.
- Observa que la señal `ci-1`, cuyo valor es `1`, entra al sumador de la ALU elemental menos significativa. Se sumará de esta forma `1` al valor del complemento a 1 del operando `B`. De esta forma la ALU realiza el complemento a 2 del número para realizar la resta.
- Desactiva la simulación y cierra el circuito `alu1.dig`.

En esta sesión se ha ilustrado el funcionamiento de la ALU con 4 bits. Como habrás podido observar, el funcionamiento es perfectamente escalable a cualquier número de bits encadenando componentes del tipo `alu1`. Cada componente `alu1` o ALU elemental opera con un bit de cada operando (A_i y B_i), teniendo en cuenta el valor del acarreo previo (`ci-1`) y produciendo un resultado (S_i) y un acarreo de salida (`ci`).

4. Ejercicios adicionales

- Realiza las operaciones recogidas en la tabla 2 «en papel», es decir, sin utilizar el simulador. En el caso de las operaciones lógicas el valor de los bits de estado *carry* y *overflow*, aunque la ALU los genera, no son significativos y deben ignorarse.

Tabla 2. Operaciones propuestas

A	B	Operación	Resultado	ZF	CF	OF	SF
0010	1100	OR					
0010	1100	AND					
0010	1100	XOR					
0010	1100	Suma					
0010	1100	Resta					

- Verifica que tus respuestas son correctas comparándolas con el resultado que ofrece el simulador.

- Elige, sobre el papel y de forma razonada, dos operandos cuya suma genere acarreo; calcula su suma y los bits de estado. A continuación, mediante simulación comprueba que el resultado de su suma y los bits de estado (*carry* y *overflow*) son iguales a los obtenidos sobre el papel.
- Busca dos operandos enteros positivos cuya suma genere acarreo. ¿Existen tales operandos? ¿Por qué?
- Busca dos operandos enteros negativos cuya suma no genere acarreo. ¿Existen tales operandos? ¿Por qué?
- Elige, sobre el papel y de forma razonada, dos operandos enteros positivos cuya resta dé un valor negativo. Calcula su resta y los bits de estado. A continuación, mediante simulación comprueba que el resultado de su suma y los bits de estado (*carry* y *overflow*) son iguales a los obtenidos sobre el papel.
- Busca dos operandos enteros positivos cuya resta genere *overflow*. ¿Existen tales operandos? ¿Por qué?