

Bloque 2

Sistemas digitales

Prácticas de
Fundamentos de Computadores y Redes
7 de febrero de 2017

SESIÓN 1

Simulador educacional de circuitos digitales

Objetivos

En esta sesión se pretende introducir al alumno en la utilización de la herramienta de simulación SECD (Simulador Educacional de Circuitos Digitales). Esta herramienta permite diseñar en pantalla un circuito digital y simular su comportamiento paso a paso, probando diferentes combinaciones en sus entradas y permitiendo examinar el estado de sus salidas.

También se presentará el uso del “Generador de entradas” como elemento que facilita la generación de entradas para probar el circuito. Por último, se mostrará cómo realizar nuevos componentes definidos por el usuario para poder utilizarlos en futuros proyectos.

Conocimientos y materiales necesarios

Antes de comenzar esta práctica el alumno debe:

- Conocer la función desempeñada por un multiplexador.
- Comprender el concepto de “Tabla de verdad” de un circuito y ser capaz de escribir la tabla de verdad de un multiplexador.

Además, es conveniente llevar a clase una memoria USB para ir guardando los circuitos que se desarrollarán en prácticas.

Desarrollo de la práctica

1. Construcción de un circuito digital simple

El programa SECD es un programa desarrollado en Java y publicado como proyecto de software libre con licencia GPL. Se puede obtener la última versión en:

<http://sourceforge.net/projects/secd>

Para poder ejecutar el programa hace falta tener instalada la máquina virtual de java que se puede descargar de la siguiente dirección:

<http://www.java.com/getjava/>

Tu profesor te indicará dónde está ubicada la versión que se va a utilizar en prácticas. Haz doble clic sobre ella para arrancar el programa. Deberías ver una pantalla como la mostrada en la figura 1.1.

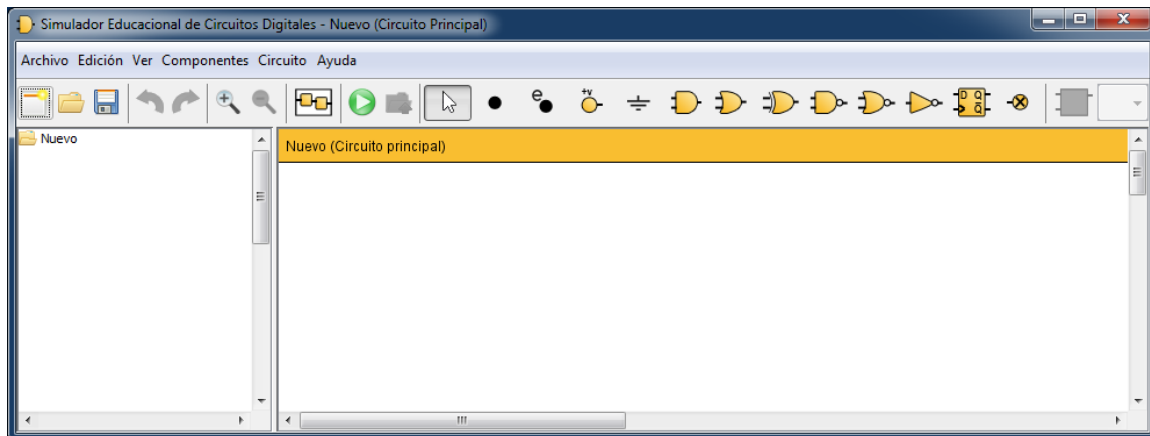


Figura 1.1: Pantalla principal del programa SECD

En la pantalla principal se pueden encontrar, de arriba a abajo, un menú con las distintas opciones del simulador, una barra de herramientas con algunas órdenes básicas (crear proyectos nuevos, abrir proyectos, deshacer y rehacer, y cambiar el grado de zoom, seleccionar e insertar componentes, etc.) y un área de trabajo dividida en dos zonas: a la izquierda un árbol donde se mostrarán los ficheros del proyecto y a la derecha un área para diseñar circuitos. En esta práctica aprenderás a utilizar todos estos elementos progresivamente.

Vamos a construir un multiplexador con dos canales de entrada (y, por tanto, con una línea de selección). El circuito final será el mostrado en la figura 1.2.

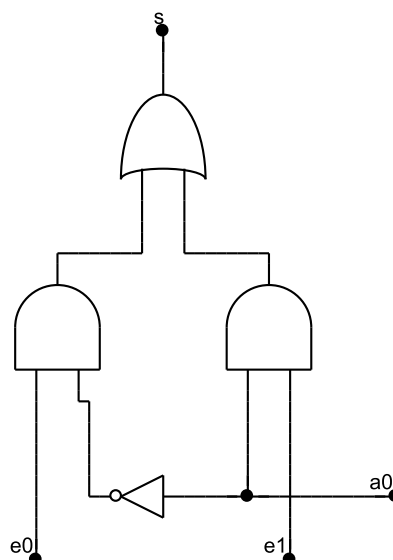


Figura 1.2: Esquema de un multiplexador de dos canales construido con puertas lógicas

Selecciona, en la barra de herramientas, la puerta AND y haz clic sobre el área de dibujo para poner dos puertas AND siguiendo el esquema de la figura 1.2. Utilizando el mismo

procedimiento, sitúa la puerta OR como en el esquema de la figura 1.2.

A continuación, selecciona en la barra de herramientas la puerta NOT y sitúala debajo de las dos puertas AND. Como puedes observar, las puertas no están orientadas igual que en la figura 1.2. SECD ofrece la posibilidad de girar puertas. Para ello, tienes que escoger, en primer lugar, la herramienta de selección, lo que puedes hacer pulsando sobre la flecha de la barra de componentes, o de manera aún más fácil, pulsando con el botón derecho del ratón o con la tecla `[ESC]`. Como verás, el cursor cambia a una flecha, lo que indica que estás trabajando con la herramienta de selección. Haz clic con el botón derecho sobre la puerta que deseas girar para seleccionarla; deberás ver que pasa a dibujarse en rojo para indicar que está seleccionada. A continuación, podrías girarla haciendo clic con el botón derecho y escogiendo en el menú emergente que aparece la opción “Girar”, pero es más fácil utilizar la combinación de teclas `[Ctrl+G]` o utilizando los iconos de la barra de tareas. En algunos casos, deberás girarla varias veces para conseguir la orientación adecuada.

A continuación, necesitarás colocar un punto de conexión (•) a la derecha de la puerta NOT (para poder unir allí dos cables) y puntos de conexión para que hagan de entradas y salidas. Para ello selecciona, en la barra de componentes, el punto de conexión (puedes utilizar el que tiene una “e” a la izquierda para que te pregunte directamente por la etiqueta) y sitúa varios tal como se muestran en la figura 1.3.

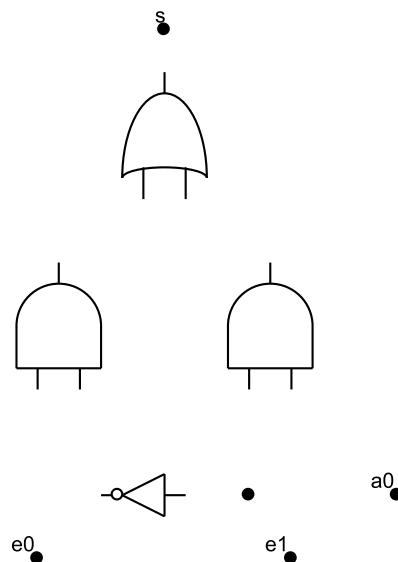


Figura 1.3: Puntos del multiplexador de dos entradas de datos

Como puedes observar, en esta figura las entradas y las salidas están etiquetadas. Para introducir la etiqueta se debe utilizar la herramienta de selección y hacer doble clic sobre el punto a etiquetar. Para simular y para hacer nuevos componentes es obligatorio que las entradas y salidas estén etiquetadas, así que etiqueta las de tu circuito.

Finalmente, debes conectar con cables las puertas y los puntos para obtener el esquema del multiplexador. Para conectar un cable, acércate a un borde de una patilla de una puerta o al borde de un punto, hasta que veas que aparece un cuadrado y el cursor cambia a una mano (puedes utilizar el zoom si tienes alguna dificultad). A continuación, haz clic con el botón izquierdo y, sin soltarlo, arrastra hasta otro punto o patilla y suelta el botón cuando veas aparecer la mano de nuevo.

2. Verificación del circuito

Para comprobar si el circuito se comporta realmente como un multiplexador, será necesario introducir en sus entradas todas las combinaciones de unos y ceros posibles, y examinar qué ocurre con su salida en cada uno de estos casos. Podremos así comparar la salida que produce nuestro circuito con la tabla de verdad de un multiplexador y, si coinciden, concluiremos que nuestro circuito se comporta como un multiplexador y, por tanto, *es* un multiplexador.

Para examinar la salida del circuito, conecta a dicha salida una bombilla, que se representa en la barra de componentes como un aspa rodeada por un círculo, \otimes . Cuando esta bombilla se pone verde, la salida tiene un uno. Si se pone gris, la salida tiene un cero. El mismo convenio de colores se utiliza en los cables. Cuando no se sabe si un componente tiene uno o cero, se pinta de azul.

Para generar diferentes combinaciones de entradas, usaremos el componente llamado “generador de entradas”, que aparece cuando intentamos simular un circuito y hay entradas que no están conectadas a nada. Se considerará entrada cualquier punto que **esté etiquetado, del que sólo salga un cable y que esté conectado a una entrada de otro componente**.

Pulsa el botón verde de la barra de herramientas. Ese botón se utiliza para simular el circuito. Al pulsarlo, como hay entradas que no tienen valor, aparecerá el generador de entradas con un aspecto como el mostrado en la figura 1.4.

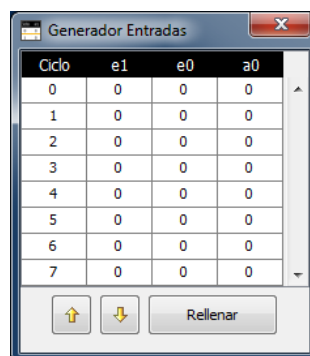




Figura 1.4: Generador de entradas

En el generador hay una columna etiquetada con el nombre de cada una de las entradas. Puedes escribir ceros o unos pulsando sobre las celdas de la tabla para escribir las 8 combinaciones de entradas posibles de un multiplexor, pero el generador proporciona el botón “Rellenar”, que genera todas las combinaciones posibles de forma automática. Pulsa este botón y comprueba cómo aparecen.

En este momento los cables del circuito están en azul, lo que indica que todavía no tienen valor y, por lo tanto, se encuentran en estado indefinido. Pulsa el botón \downarrow en el generador de entradas y observa que ocurren dos cosas:

- Se resalta en verde la primera fila de la tabla del generador. Esto indica que el generador está generando los valores de esa fila, es decir, en cada una de las entradas del circuito se está colocando el valor lógico que indica la columna correspondiente del “generador de entradas”.

- En el circuito puedes ver que los cables que salen de todas las entradas han pasado a color gris, lo que indica que tienen un cero, que es lo que está produciendo el generador de entradas en la fila resaltada. Esos valores se propagarán por el resto de elementos hasta llegar a la bombilla, que también está en gris, indicando que tiene un cero.


Vete avanzando con los botones del generador de entradas,  y , probando todas las combinaciones y comprobando que la salida de tu circuito es correcta según la tabla de verdad de un multiplexor. ¿Cuántas combinaciones generan un uno a la salida?^[1]

1

Llegado a este punto, vamos a guardar el circuito. Antes de poder hacerlo, debes parar la simulación pulsando en el botón de stop de la barra de herramientas. Para guardar, pulsa sobre el icono del disquete en la barra de herramientas. Te aparecerá un cuadro de diálogo preguntándote dónde quieres guardar el proyecto. Selecciona la carpeta donde desees guardarlo (fíjate en cuál es), introduce como nombre de archivo “mux2” y pulsa “Guardar”. Para comprobar que el proyecto se ha guardado correctamente, abre el Explorador de Windows y vete hasta la carpeta donde has guardado el proyecto. En ella deberás ver un fichero “mux2.pro”, el fichero del proyecto. Un proyecto es un circuito principal que puede usar otros componentes nuevos definidos por el usuario. Estos componentes deberán ser añadidos al proyecto, como verás más adelante.

3. Creación de nuevos componentes definidos por el usuario

En la sección anterior hemos creado un multiplexor de dos canales y hemos comprobado que funcionaba correctamente. Ahora añadiremos este circuito como un componente nuevo, de modo que si en el futuro necesitamos un multiplexor de dos canales como parte de un circuito mayor, no tendremos que volver a crearlo y probarlo, sino que reutilizaremos el que acabamos de crear.

El proceso para crear un nuevo componente (también denominado macro) requiere que lo que vayan a ser entradas y salidas del nuevo componente no estén conectadas a ningún elemento. Por lo tanto, habrá que borrar la bombilla que está conectada a la salida. Para ello, selecciónala y pulsa la tecla .

Una vez que el circuito está preparado, para convertirlo en componente se deben realizar los siguientes pasos:

- ❑ Pulsa en la barra de herramientas el botón de creación de componente nuevo: el rectángulo blanco con dos rectángulos naranja en su interior.
- ❑ Te aparecerá el cuadro de diálogo de creación de macros. SECD toma el nombre del proyecto (en este caso “mux2”) para el nuevo componente, pero podrías cambiarlo si quisieses. En este caso queremos que el nuevo componente se llame “mux2”, así que no se debe cambiar.
- ❑ En el cuadro de diálogo puedes observar varias listas. La primera contiene los puntos candidatos a ser salidas o entradas antes de que se les asigne posición. Las siguientes listas contienen los puntos que están arriba, abajo, a la izquierda o a la derecha de la interfaz que tendrá el nuevo componente. Utilizando los botones que están a la derecha de la lista de puntos candidatos, crea la macro para que tenga el aspecto presentado en la figura 1.5. Fíjate en que e0 tiene que estar a la izquierda de e1.

- ❑ Pulsa “Aceptar” para crear el componente. Deberás observar que en la barra de herramientas de componentes ha aparecido el nuevo componente en el *combo-box* de la derecha.

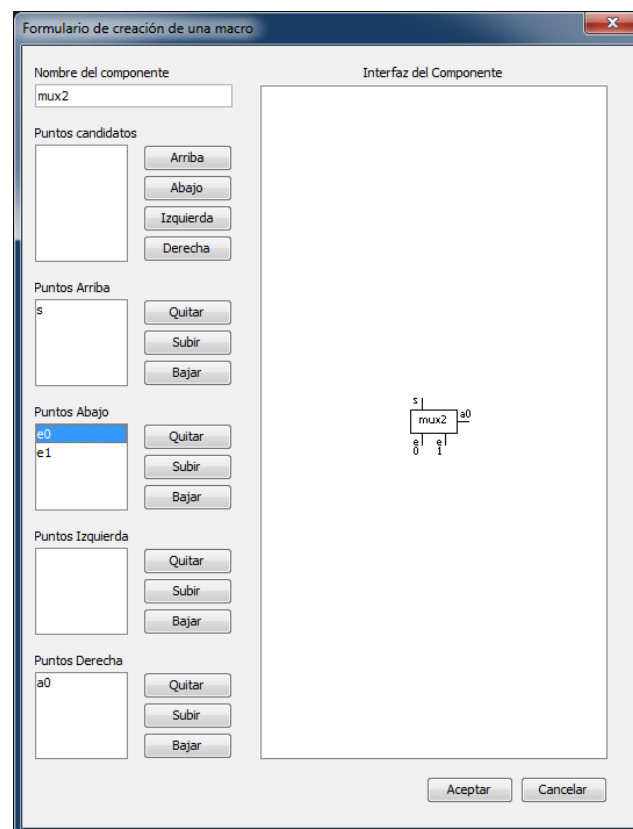


Figura 1.5: Creación del componente mux2


Guarda el circuito en este momento. Vete con el Explorador de Windows a la carpeta donde estás guardando tus circuitos. Como verás, ha aparecido un nuevo archivo, “mux2.cir”, que contiene el componente definido que acabas de crear.

4. Simulación con componentes definidos

Para ver cómo funcionan los componentes definidos, vamos a hacer un circuito nuevo a partir del proyecto que ya está hecho:

- ❑ Borra el circuito que tienes en el circuito principal. Para ello, utiliza la herramienta de selección y, trazando un rectángulo, selecciona todos los componentes. A continuación, pulsa la tecla **Supr**.
- ❑ Selecciona, en la barra de herramientas, el componente mux2. Para ello debes pulsar sobre el icono que hay al lado de su nombre. El cursor cambiará para indicar que tienes el componente seleccionado.
- ❑ Sitúa el componente en el área de dibujo.
- ❑ Ahora deberías crear puntos para las entradas y salidas y darles nombre como se muestra en la figura 1.6. Como es muy habitual esta operación de

añadir puntos a las entradas y salidas de un componente definido y darles nombre, SECD tiene una forma rápida de llevarla a cabo: pulsar con el botón derecho del ratón sobre el componente y escoger la opción “Situar puntos”. Pulsa la tecla **[ESC]** para que el cursor pase al modo selección y utiliza la función “Situar puntos” como se acaba de explicar.

- ❑ Coloca una bombilla a la salida como en la figura.
- ❑ Guarda el proyecto, lo que te guardará también el circuito principal.
- ❑ Simula el circuito con el botón de simular. Te aparecerá el generador de entradas. Presiona el botón de “Rellenar” para generar todas las combinaciones posibles y, con el botón , vete avanzando por la tabla comprobando que todo sale bien.
- ❑ Vete a la fila correspondiente al ciclo 3 en el generador. Si no estuviera escogida, escoge la herramienta de selección en la barra de componentes de la ventana principal y haz doble clic sobre el dibujo del componente “mux2” que tienes en el área de dibujo. Se abrirá el circuito correspondiente a “mux2” y en él puedes ver el valor de sus entradas, salidas y cables. Esta es una información muy útil para comprender cómo funcionan circuitos complejos hechos a partir de otros circuitos más simples.
- ❑ Para volver al circuito principal, haz clic en el botón con la flecha amarilla hacia la izquierda de la barra de herramientas. Fíjate que el circuito principal de un proyecto muestra en la parte superior del área de trabajo una banda naranja con el nombre del proyecto seguido de «(Circuito principal)» y el fondo es blanco, mientras que cuando se está mostrando un componente, la banda es gris, el texto mostrado es el nombre del componente seguido de «(Componente definido)» y el fondo es gris. Esta información te puede ser muy útil para no confundirte a la hora de crear componentes definidos, que deberían ser creados sólo en el circuito principal de un proyecto. Ten cuidado cuando trabajes con los ficheros de los circuitos. Si los renombas, pueden dejar de funcionar ellos mismos u otros circuitos que dependan de ellos. Además, todos los componentes de un proyecto deben estar en la misma carpeta.
- ❑ Para moverte entre los distintos circuitos de un proyecto (el principal, que es el correspondiente al proyecto, y los componentes que tenga) también puedes utilizar el árbol que se muestra en la parte izquierda de la ventana del programa. Sin embargo, si lo intentas ahora, el programa te mostrará un mensaje de error porque cuando se está simulando puede haber varias instancias del mismo componente con distintos estados y desde el árbol no se puede saber cuál quieres abrir. Por lo tanto, antes de abrir un componente mediante el árbol, debes parar la simulación. Detén la simulación y haz doble clic sobre “mux2”. Verás que se abre el fichero de componente “mux2”.

5. Construcción de un multiplexador de cuatro canales de entrada

El componente que vamos a crear a continuación, visto como una “caja negra”, tiene 6 entradas (4 canales de entrada y 2 de selección) y una salida. Llamaremos a los canales de entrada e_0 , e_1 , e_2 y e_3 , y a las líneas de selección a_0 y a_1 (el subíndice indica el peso del correspondiente bit). La salida será s .

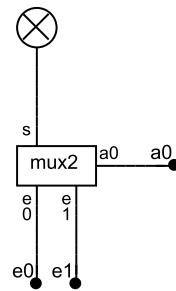


Figura 1.6: Prueba del componente mux2

Existen dos posibles enfoques para construir un multiplexador de cuatro canales de entrada:

- Escribir su tabla de verdad y a partir de ella deducir su función lógica.
- Tratar de aprovechar el componente mux2 desarrollado anteriormente.

El primer enfoque es directo y automático, pero tiene el inconveniente de ser extremadamente tedioso en este caso, debido a que la tabla de verdad de un multiplexador de cuatro canales es muy larga (constaría de $2^6 = 64$ combinaciones de entrada). Se deja como ejercicio para el alumno interesado.

En esta sesión de prácticas se utilizará el segundo enfoque. Para ello se necesitarán tres multiplexadores de 2 entradas (es decir, tres copias del componente mux2). En la figura 1.7 se muestra la forma de conectar entre sí estos tres componentes.

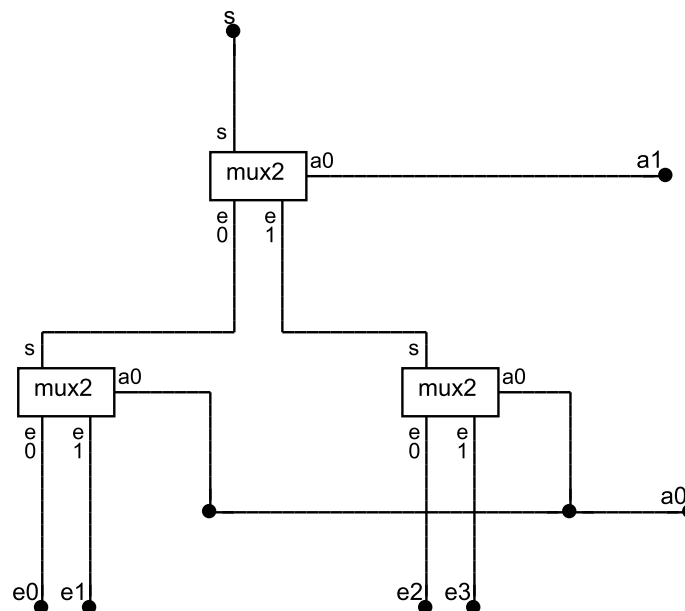


Figura 1.7: Conexión de tres multiplexadores de 2 canales para crear un multiplexador de 4 canales

Debes realizar los siguientes pasos:

- ❑ Carga el proyecto “mux2” si no lo tuvieses abierto.

- ☐ Guarda el proyecto con el nombre “mux4” y borra los elementos que tienes en el circuito principal.
- ☐ Crea un circuito con la estructura mostrada en la figura 1.7 en la página anterior, usando el componente mux2 del banco.
- ☐ Convierte el circuito en una *macro* llamada mux4. Configura las entradas y salidas tal y como se muestran en la figura 1.8

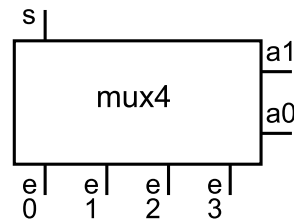


Figura 1.8: Macro del multiplexador de 4 canales

Para ver cómo funciona el componente definido, vamos a hacer un circuito nuevo a partir del proyecto que ya está hecho:

- ☐ Borra el circuito que tienes en el circuito principal. Para ello, utiliza la herramienta de selección y, trazando un rectángulo, selecciona todos los componentes. A continuación, pulsa la tecla **Supr**.
- ☐ Selecciona, en la barra de herramientas, el componente mux4. Para ello, debes pulsar sobre el icono que hay al lado de su nombre.
- ☐ Sitúa el componente en el área de dibujo.
- ☐ Crea puntos para las entradas y la salida y dales nombre usando la función “Situación puntos”. Coloca una bombilla a la salida.
- ☐ Guarda el proyecto, lo que guardará también el circuito principal.

A continuación, y antes del proceso de simulación, rellena lo que crees que aparecerá en la salida *s* del circuito ante las siguientes combinaciones de sus entradas:

e3	e2	e1	e0	a1	a0	s
0	0	0	0	0	1	
0	0	0	0	1	1	
0	0	1	0	1	0	
0	1	0	0	1	0	
0	1	1	1	1	1	
1	1	1	1	0	0	

Una vez que has rellenado la tabla, pulsa el botón de simular; te aparecerá el generador de entradas. Introduce en el generador de entradas las combinaciones de las entradas propuestas en la tabla anterior y verifica que tus respuestas han sido correctas.

Recuerda que mientras estás simulando puedes hacer doble clic sobre el dibujo de un componente para comprobar su estado interno.


6. Ejercicios adicionales

6.1. Ficheros en el disco

En la carpeta donde guardes los circuitos debes tener estos ficheros:

- `mux2.pro`, y `mux2.cir`, que contienen el proyecto con el circuito principal y el circuito del componente `mux2`.
- `mux4.pro`, y `mux4.cir`, que contienen el proyecto con el circuito principal y el circuito del componente `mux4`.

6.2. Ejercicios

- ⇒ Abre el proyecto `mux2`. Pulsa el botón “Simular” y escribe en la primera línea los valores `e0=1`, `e1=0` y `a0=0`. Cuando presiones el botón  para que se genere esa combinación, ¿de qué color deberá quedar la bombilla?^[2]
- ⇒ Crea un proyecto nuevo (“Archivo→Nuevo Proyecto”) con un circuito que implemente una puerta NAND de cuatro entradas. A partir de este circuito, crea un componente definido llamado `NAND4`.
- ⇒ Crea un proyecto nuevo. A continuación, crea el circuito de la figura 1.9, que tiene tres entradas y una salida.

2

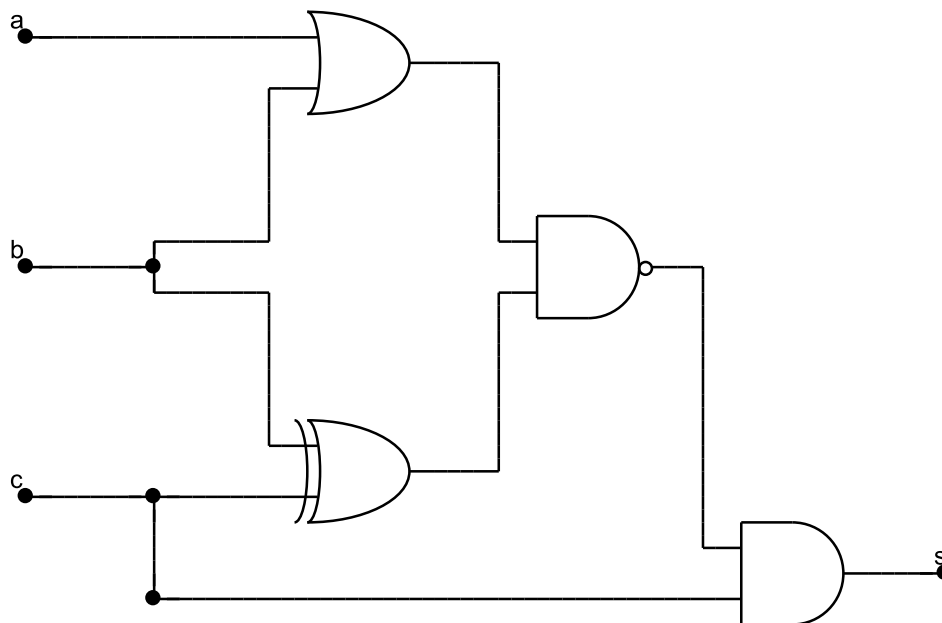


Figura 1.9: Circuito del ejercicio propuesto

Conecta la salida del circuito a una bombilla. Antes de simular el circuito, rellena su tabla de verdad:

<i>a</i>	<i>b</i>	<i>c</i>	<i>s</i>
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Simulando el circuito, comprueba que concuerda lo simulado con la tabla de verdad.

- ⇒ Crea una macro que implemente la función lógica $s = a \cdot (\overline{b + c})$ y comprueba que funciona correctamente.

SESIÓN 2

La ALU

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 suma y resta, así como las operaciones lógicas AND, OR y XOR.

Como herramienta de prácticas se empleará el simulador de circuitos digitales SECD, 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

- Tema 2 del libro *Fundamentos de 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.

Desarrollo de la práctica

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 2.1 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, R, y 4 bits de estado, ZCOS.

La tabla 2.1 indica la operación realizada en función del valor de los bits de operación.

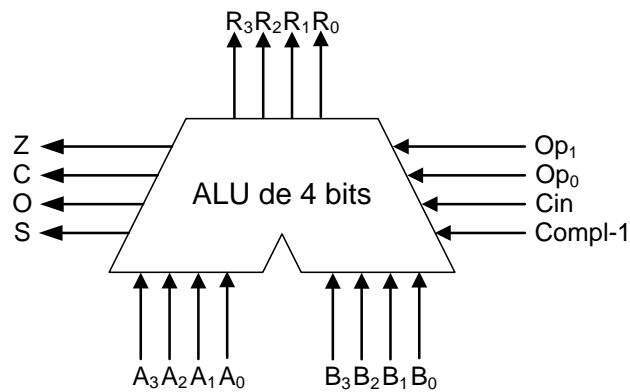


Figura 2.1: Entradas y salidas de una ALU de 4 bits

Op1	Op0	Cin	Compl-1	Operación
0	0	0	0	$R = A \text{ AND } B$
0	1	0	0	$R = A \text{ OR } B$
1	0	0	0	$R = A \text{ XOR } B$
1	1	0	0	$R = A + B$
1	1	1	0	$R = A + B + 1$
1	1	1	1	$R = A - B$

Cuadro 2.1: Operaciones de la ALU

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 R y los 4 bits de estado ZCOS.
- Se realizará la operación en el simulador, para lo cual será necesario programar de forma adecuada el generador de 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 R, el resultado de la suma?^[3]
- ¿Qué valor deben tomar los bits de estado Z y S?^[4]. ¿Por qué?^[5]
- Interpretando los operandos A, B y el resultado R como números naturales, ¿el resultado de la suma es correcto o incorrecto?^[6] ¿Qué valor debe tomar entonces el bit de C de acarreo?^[7]. ¿Has obtenido ese valor de acarreo?

3

4

5

6

7

- ❑ Interpretando ahora los operandos A, B y el resultado R como números enteros, ¿el resultado de la suma es correcto o incorrecto? ^[8] ¿Por qué? ^[9] ¿Qué valor debería tomar entonces el bit 0 de *overflow*? ^[10]

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.pro`, el cual debes abrir empleando el simulador de circuitos digitales SECD. 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 2.2. Para el nombrado de señales en el simulador se ha tomado el convenio de escribirlas todas en minúsculas.

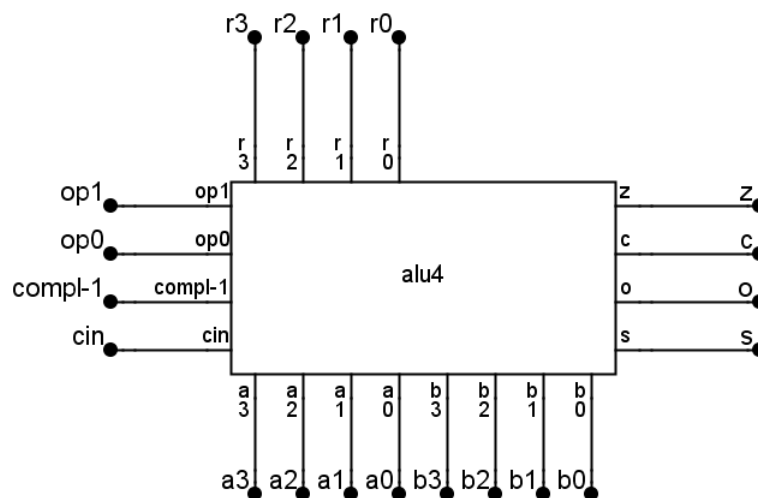


Figura 2.2: La ALU de 4 bits en el simulador

- ❑ Configura el generador de entradas del simulador para llevar a cabo la operación de suma de A y B.
- ❑ Observa el resultado de la suma y el de los bits de estado. ¿Coinciden con los que obtuviste manualmente?
- ❑ Haz doble clic sobre el componente `alu4`. El simulador muestra el interior de la ALU en el cual se observan sus componentes y el estado lógico de los cables de conexión como en la figura 2.3. El resultado de la suma, R, 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 *s* se obtiene directamente del bit más significativo del resultado, pues es el que indica si el resultado es negativo (*s*=1) o positivo (*s*=0). Estos son los componentes que aparecen a la derecha:
 - Una puerta XOR que genera el bit de acarreo *c*. 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, *c*=*cout*. Si el bit `compl-1` fuese 1, como ocurre en el caso de las operaciones de resta, el bits de acarreo *c* sería justo *cout* negado. ¿Qué valor toman los bits *cout* y *c*? ^[11]

8

9

10

11

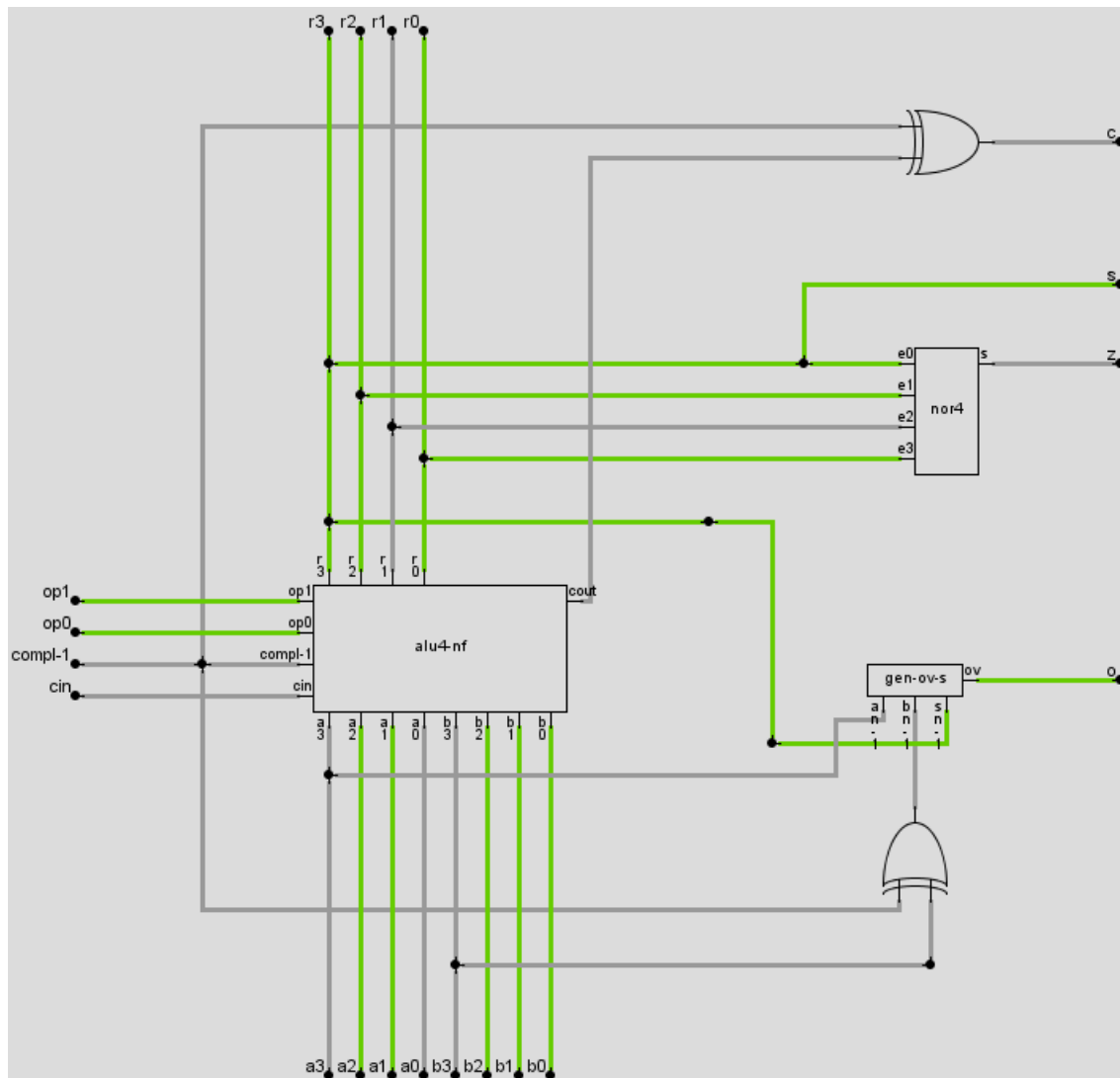


Figura 2.3: Interior de la ALU de 4 bits durante la suma

- 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 $R=0000$. Este es justo el resultado esperado del bit de estado z . ¿Qué valores toman las entradas y salidas de la puerta NOR?^[12]
- 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 R. ¿Qué signos tienen A, B y R?^[13] ¿Es coherente el signo de R con el de los operandos A y B?^[14] La salida `ov` del generador de *overflow* debe reflejar precisamente esa coherencia y por consiguiente debe indicar si el resultado de la suma es correcto interpretando los operandos y el resultado como enteros en complemento a 2.

12

13

14

- Haz doble clic ahora sobre el componente alu4-nf, aparecerá su estructura interna, como se muestra en la figura 2.4. Para entender el porqué de esta implementación, la figura 2.5 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.

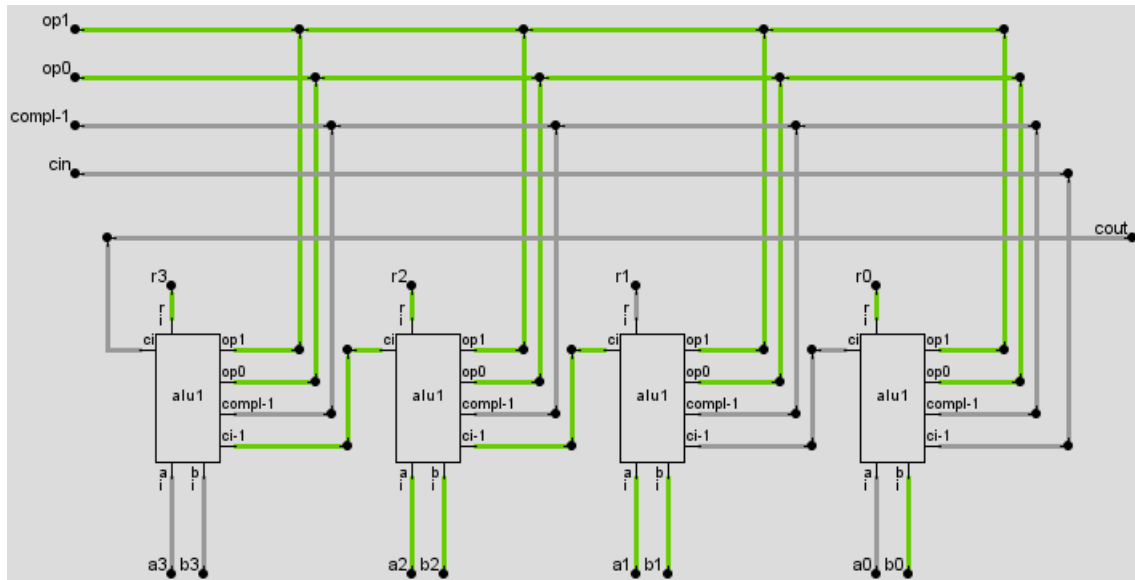


Figura 2.4: Estado interno de la ALU sin *flags* durante la suma

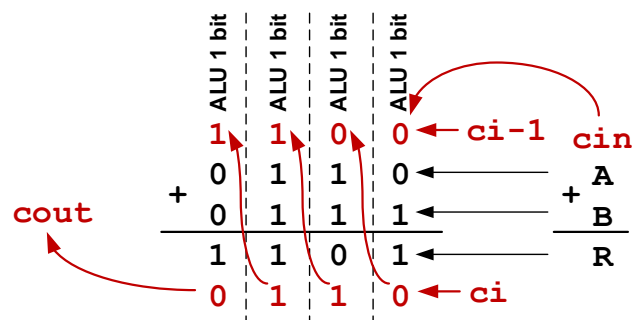


Figura 2.5: Ejemplo de suma de 4 bits

Por ejemplo, la ALU de 1 bit menos significativa, la que está más a la derecha, realiza la operación con los bits a0 y b0, generando un resultado r0 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, compl-1 y cin. 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 2.5.

- Haz doble clic ahora sobre la ALU de 1 bit más significativa, la que aparece más a la izquierda, aparecerá su estructura interna, como se muestra en la figura 2.6.

El funcionamiento de la ALU de 1 bit es simple. Realiza 4 operaciones a

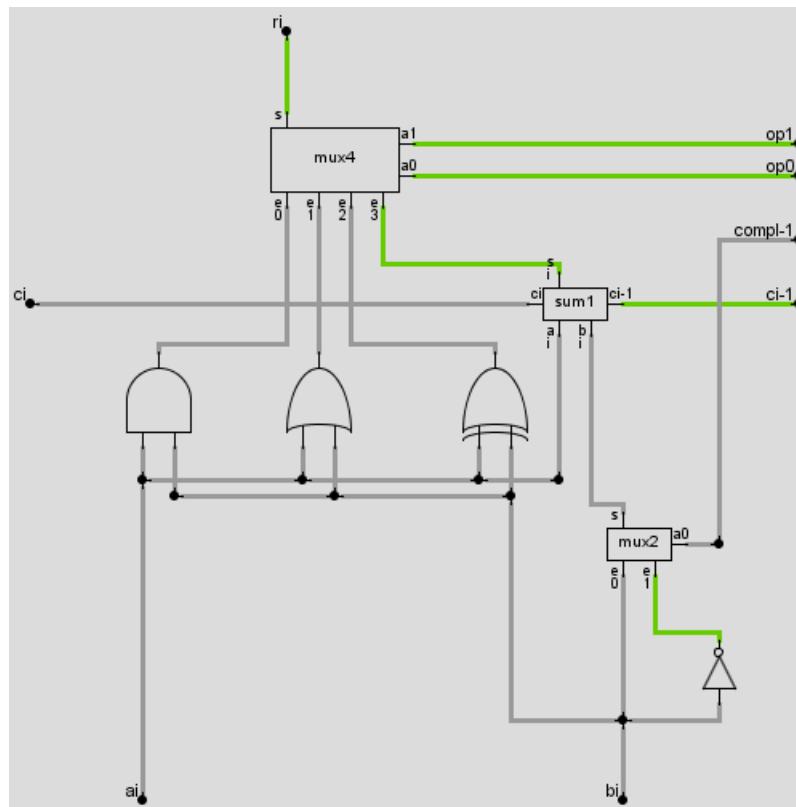


Figura 2.6: Estado interno de la ALU de 1 bit más significativa

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 operandos 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. ¿Qué valor toma R, el resultado de la resta?^[15]
- ❑ ¿Qué valor deben tomar los bits de estado Z y S?^[16]. ¿Por qué?^[17]

15

16

17

- ☐ Interpretando los operandos A, B y el resultado R como números naturales, ¿es posible llevar a cabo la operación de resta? ¿Por qué? ^[18].
- ☐ Al sumar al operando A el complemento a 2 del operando B, ¿se ha producido acarreo? ^[19]. Cuando no se produce acarreo al hacer esta operación con el complemento a dos, 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 dos, sí es posible la operación de resta original.
- ☐ Interpretando ahora los operandos A, B y el resultado R como números enteros, ¿el resultado de la resta es correcto o incorrecto? ^[20] ¿Por qué? ^[21] ¿Qué valor debería tomar entonces el bit 0 de overflow? ^[22].

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

- ☐ Configura el generador de entradas del simulador para llevar a cabo la operación de resta de A y B.
- ☐ 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 c a 1 indicando un error en la resta de naturales. Cuando la ALU genera un bit de estado c 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.
- ☐ Haz doble clic sobre el componente alu4. El simulador muestra el interior de la ALU en el cual se observan sus componentes y el estado lógico de los cables de conexión. ¿Qué valor toma el bit de acarreo a la salida de la ALU de 4 bits sin *flags*? ^[23] ¿Qué valor toma el bit de acarreo a la salida de la ALU de 4 bits completa? ^[24]. Observa como 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.
- ☐ Haz doble clic ahora sobre el componente alu4-nf. ¿Qué valor toma el acarreo de entrada de la ALU de 1 bit menos significativa? ^[25].
- ☐ Haz ahora doble clic sobre esa misma ALU. ¿Qué operación está seleccionada dentro de la ALU de 1 bit? ^[26]. Fíjate en el valor de las entradas del multiplexor. ¿Qué valores toman las entradas bits ai y bi que llegan al sumador? ^[27]. ¿Coinciden con las que llegan a la ALU de 1 bit a través de sus líneas ai y bi? ^[28]. Como te habrás dado cuenta, en el caso de las operaciones de resta se suma al operando A el complemento a 2 del operando B, o lo que es lo mismo, se le suma B negado más 1 (*ci-1=1* en la ALU menos significativa).

18
19

20

21

22

23

24

25

26

27

28

4. Ejercicios adicionales

4.1. Ejercicios

- ⇒ Rellena la tabla siguiente, haciendo las operaciones “en papel”, es decir, sin utilizar el SECD. En el caso de las operaciones lógicas el valor de los bits de estado C y 0 no tiene sentido y debe ignorarse.

A	B	Operación	Resultado	Z	C	0	S
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 SECD.

- ⇒ 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?

SESIÓN 3

Integración de la ALU en la CPU teórica

Objetivos

El objetivo de esta sesión práctica es la simulación por parte del alumno de operaciones aritméticas y lógicas dentro de la CPU del Computador Teórico. Para no complicar en exceso la simulación se considera una CPU de sólo 4 bits, pero totalmente análoga a la del Computador Teórico. Además, sólo se consideran los siguientes elementos de la CPU:

- Bus interno.
- ALU.
- Registro temporal de entrada.
- Registro temporal de salida.
- Registro de estado.

Después de realizar la sesión práctica el alumno debería ser capaz de definir la secuencia de señales necesaria para realizar cualquier operación aritmética o lógica empleando la CPU del Computador Teórico.

Conocimientos y materiales necesarios

- Comprensión por parte del alumno del funcionamiento de un registro y de la ALU del Computador Teórico.
- Proyecto “alu4”, generado por el alumno en la sesión anterior. Este proyecto contiene, entre otras, la macro alu4.

Desarrollo de la práctica

1. Introducción

En esta sesión se pretenden conectar la ALU, el registro temporal de entrada, el registro temporal de salida, el registro de estado y el bus interno. El objetivo es realizar con estos elementos de la CPU un conjunto de operaciones aritméticas y lógicas propuestas. Para facilitar la construcción del circuito se proporciona el proyecto cpu.

2. Construcción del circuito

Los pasos a seguir para la realización de esta sesión práctica son los siguientes:

- ❑ Descarga del Campus Virtual el fichero `cpu.pro` al directorio donde tengas los demás ficheros que has ido generando en las anteriores sesiones.
- ❑ Arranca el programa SECD en la forma habitual.
- ❑ Crea un nuevo proyecto y guárdalo como “reg4”.
- ❑ Construye el circuito correspondiente a un registro de 4 bits. Toma como referencia el circuito de la figura 3.1.

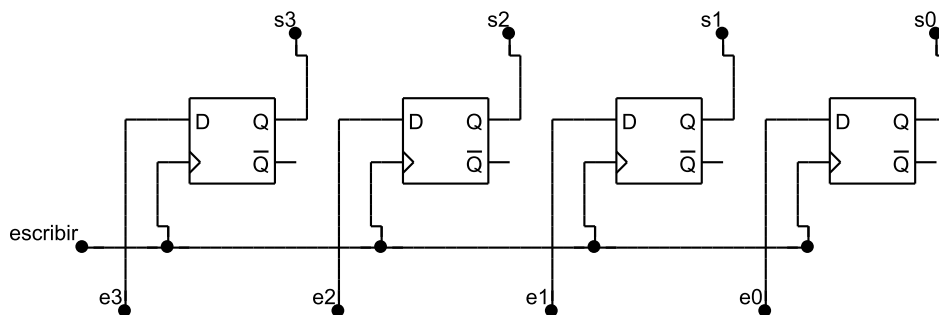


Figura 3.1: Esquema interno de un registro de 4 bits

- ❑ Genera, a partir del circuito, la macro `reg4` correspondiente a un registro de 4 bits. Las entradas y salidas deben disponerse tal como aparecen en la figura 3.2. Guarda el proyecto.

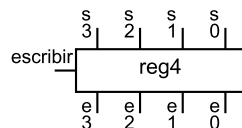


Figura 3.2: Macro del registro de 4 bits

- ❑ Carga el proyecto “cpu”. Aparecerá en la pantalla el circuito de la figura 3.3. En este circuito, ALU, TMPE, TMPS y SR representan el lugar que van a ocupar las macros `alu4` y `reg4`.

Asimismo, se han situado grupos de 4 bombillas en puntos interesantes del circuito. Estos puntos son:

- El bus interno (las cuatro líneas verticales de la derecha).
- Salida del registro de estado.
- Salida del registro temporal de salida.

Para observar el valor que tienen el resto de elementos del circuito durante la simulación se puede observar el color del cable.

- ❑ Añade, mediante el menú Archivo→Añadir archivo existente, el archivo `alu4.cir`. La herramienta te informará de que también se han añadido todos los componentes que utiliza el componente `alu4`.
- ❑ Añade también el fichero para el componente `reg4`. Guarda el proyecto.

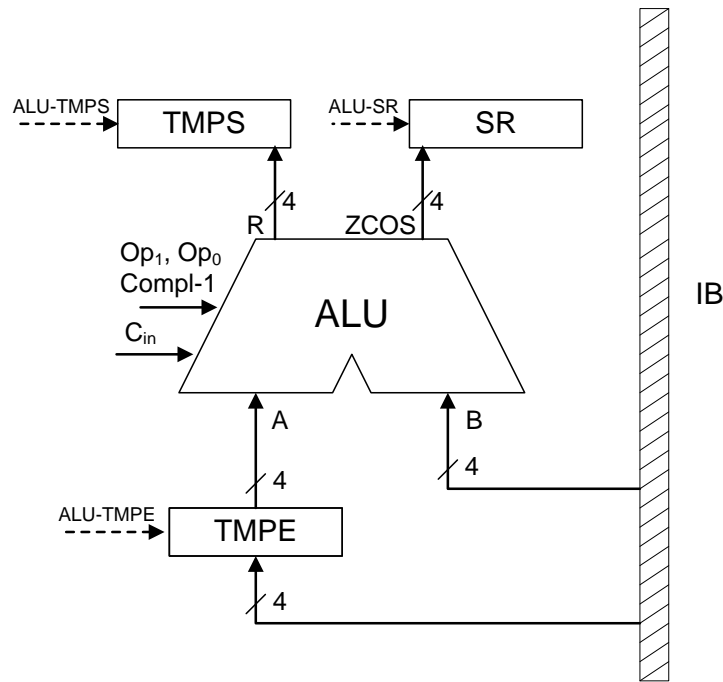


Figura 3.4: Diagrama de bloques de la CPU

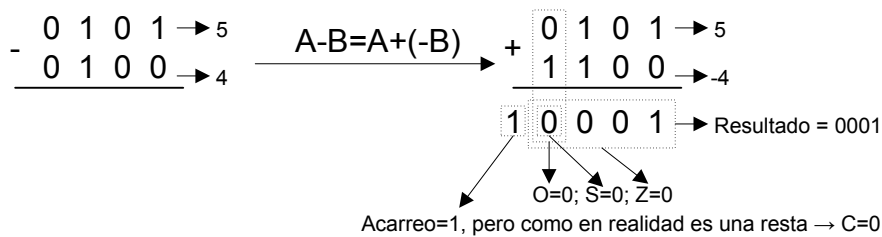



Figura 3.5: Operación 5 - 4

- Pulsa en el generador de entradas para que se vuelquen al circuito los valores de la fila correspondiente al ciclo 0. Como puedes observar, en el bus interno está el valor 0101 y este llega hasta la entrada del registro TMPE, pero la salida del registro está indefinida porque todavía no se ha escrito ningún valor en él.
- A continuación, vamos a escribir en el registro temporal de entrada el valor que hay en el bus interno. Para ello, escribe un 1 en la fila correspondiente al ciclo 1 de la señal IB-TMPE, que como puedes observar está conectada a la señal escribir del registro TMPE.
- Pulsa en el generador de entradas para que se vuelquen al circuito los valores de la fila correspondiente al ciclo 1. Observa cómo cambian los valores a la salida del registro temporal de entrada: coinciden con los que había en el bus interno cuando ocurrió el flanco (el cambio de 0 a 1).
- Sitúa el segundo operando sobre el bus interno, esto es, el valor 0100, así como las entradas de control de la ALU necesarias para hacer una resta, es decir, cin=1, compl-1=1, op0=1 y op1=1. Estos valores se deben cargar en la fila del ciclo 2.
- Pulsa en el generador de entradas para que se vuelquen al circui-

to los valores de la fila correspondiente al ciclo 2. En este momento se muestra a la salida de la ALU el resultado y los bits ZCOS. Comprueba que son correctos. Si no es así, comprueba que las entradas a la ALU son correctas; si no lo son, es que has cometido un fallo en los pasos anteriores; si las entradas a la ALU son correctas y el resultado es incorrecto, tienes algo mal hecho en la ALU. Abre su circuito y vete comprobando la generación de cada valor para descubrir el fallo.

8. Por último, vamos a escribir el resultado en el registro temporal de salida y en el registro de estado. Para ello, pon a 1, en la fila correspondiente al ciclo 3, las señales ALU-TMPS y ALU-SR.
 9. Pulsa  en el generador de entradas. Observarás cómo se almacena el resultado y los bits ZCOS en el TMPS y en el SR, respectivamente.
- ☐ Comprueba que el resultado y los bits de estado obtenidos sobre el papel coinciden con los obtenidos por simulación.
 - ☐ Guarda el proyecto.

3. Ejercicios adicionales

En este tipo de sesiones prácticas en las cuales se simulan circuitos digitales, es fundamental que trates de predecir el resultado de la simulación antes de llevarla a cabo. A continuación, debes comparar tu predicción con el resultado proporcionado por la simulación y extraer tus propias conclusiones.

Por ejemplo, si tu predicción no coincide con el resultado de la simulación podría suceder que no tengas claros los conceptos teóricos relacionados, o bien que hayas diseñado mal el circuito. En cualquier caso, debes dedicar el tiempo que sea necesario hasta conseguir que tus predicciones coincidan con los resultados de simulación.

En particular, en esta sesión práctica debes predecir correctamente el resultado y bits de estado correspondientes a cualquier operación *antes de realizar simulación alguna*. Además, con cada operación debes predecir correctamente el estado que tendrán todos los elementos del circuito *justo después de llevar a cabo cualquiera de los pasos anteriores*.

3.1. Ficheros en el disco

En la carpeta donde guardes los circuitos debes tener estos ficheros:

- `reg4.pro` y `reg4.cir`, que contienen el proyecto con el circuito principal y el circuito del componente `reg4`.
- `cpu.pro`, que contiene el proyecto de la CPU con su circuito principal.

3.2. Ejercicios

- ⇒ Realiza las operaciones aritméticas y lógicas indicadas en la tabla *en primer lugar sobre el papel y finalmente mediante simulación*, rellenando los campos resultado y bits de estado ¹.

¹En las operaciones lógicas es necesario especificar únicamente el bit de estado Z, pues los demás bits son irrelevantes.

Operando A	Operando B	Operación	Resultado	ZCOS
-3	7	Or		
5	3	Suma		
-6	-8	Resta		
4	-6	Suma		
-3	7	Resta		
6	-2	And		
7	3	Resta		
-1	-5	Xor		
5	-4	And		

- ⇒ Como has podido observar, para realizar una operación aritmética o lógica se requieren 4 ciclos. Sin embargo, una opción para reducir el número de ciclos es hacer que se escriba en los registros a la mitad de un ciclo en lugar de al principio. Usando esta técnica, ¿cuántos ciclos serían necesarios para realizar una operación aritmética o lógica?