Bloque 2

Sistemas digitales

Prácticas de
Fundamentos de Computadores
Curso 2009-2010

SESIÓN 1

Introducción a SECD

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 (ya que se va a construir uno con dos canales de entrada).
- Comprender el concepto de "Tabla de verdad" de un circuito y ser capaz de escribir la tabla de verdad de un multiplexador de dos canales de entrada.

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:

`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.

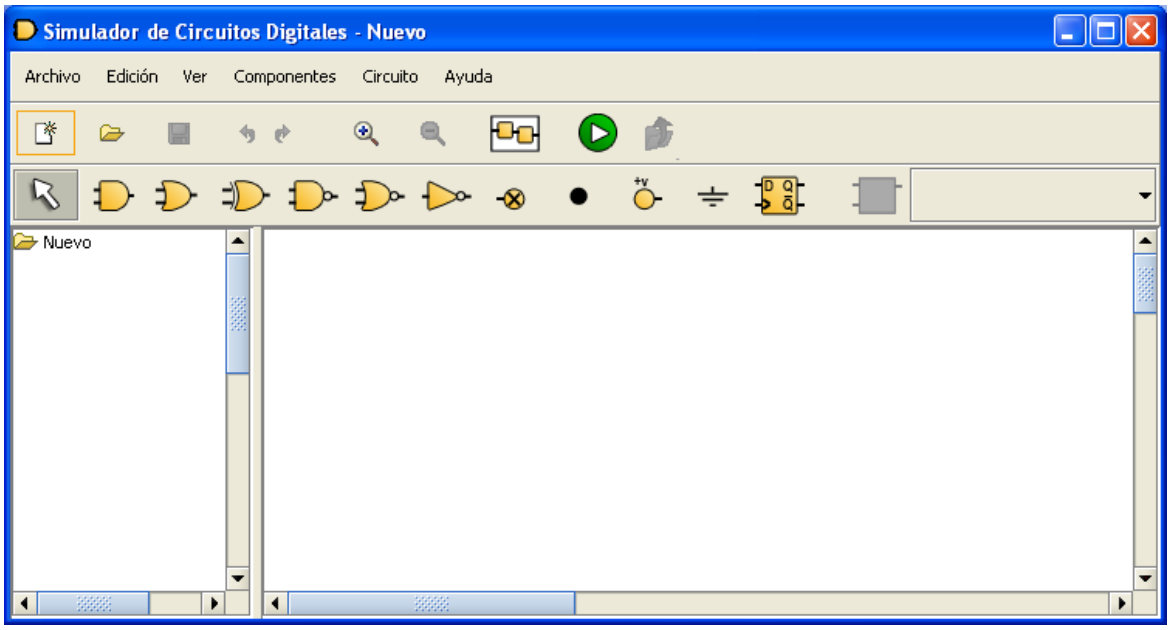

Figura 1.1: Pantalla principal del programa SECD

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, entre otras), otra barra de herramientas para seleccionar e insertar componentes (tiene los componentes básicos: puertas AND, OR, NOT, etc.) y un área de trabajo dividida verticalmente 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.

Selecciona, en la barra de herramientas de componentes, 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

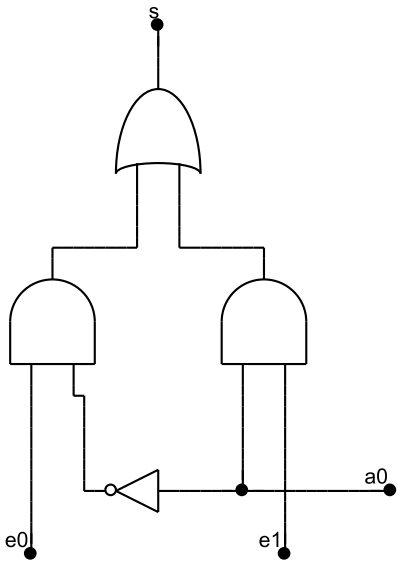

Figura 1.2: Esquema de un multiplexador de dos canales construido con puertas lógicas

cursor cambia a una flecha, lo que indica que estás trabajando con la herramienta de selección. Haz clic con el botón izquierdo 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. 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 y sitúa varios tal como se muestran en la figura 1.3.

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.

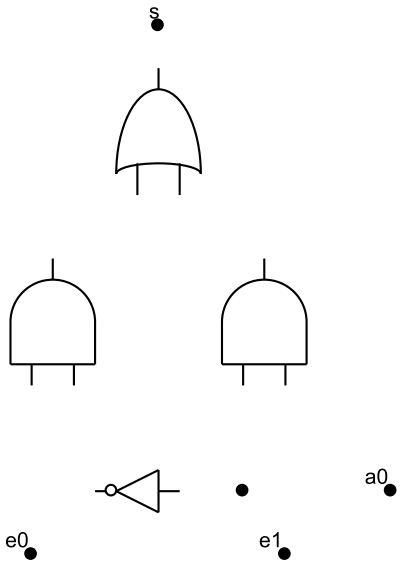

Figura 1.3: Puntos del mux2

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 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, ⊗. 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[1].

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 principal. 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.

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

[1]En el menú *Ver* de la barra de herramientas, a través de la opción *Configurar Colores Simulación* se pueden modificar los colores asignados por defecto a cada uno de los estados lógicos de las líneas del circuito durante la simulación.

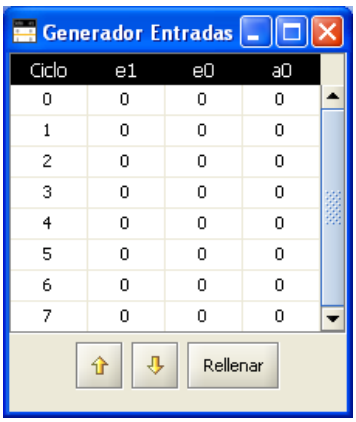

Figura 1.4: Generador de entradas

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 ↓ 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. Para ello, 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 dos ficheros: "mux2.pro" y "mux2.pri". El primero es el fichero del proyecto. Más adelante vas a definir varios componentes nuevos y en este archivo se guardará información de dónde están los componentes que se utilizan en un circuito. El fichero ".pri" es el fichero del circuito que has hecho, que se considerará el circuito principal del proyecto.

3. Creación de nuevos componentes definidos por el usuario

En la sección anterior hemos creado un multiplexor de 2 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 Supr.

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. Introduce "mux2" como nombre del nuevo componente.
- ❏ 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 contienen los puntos que están arriba, abajo, a la izquierda o a la derecha del 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 un *combo-box* y que la aplicación lo ha seleccionado por ti, lo que se muestra con el icono cambiado al nuevo componente, que ya podrías poner sobre el área de dibujo.

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:

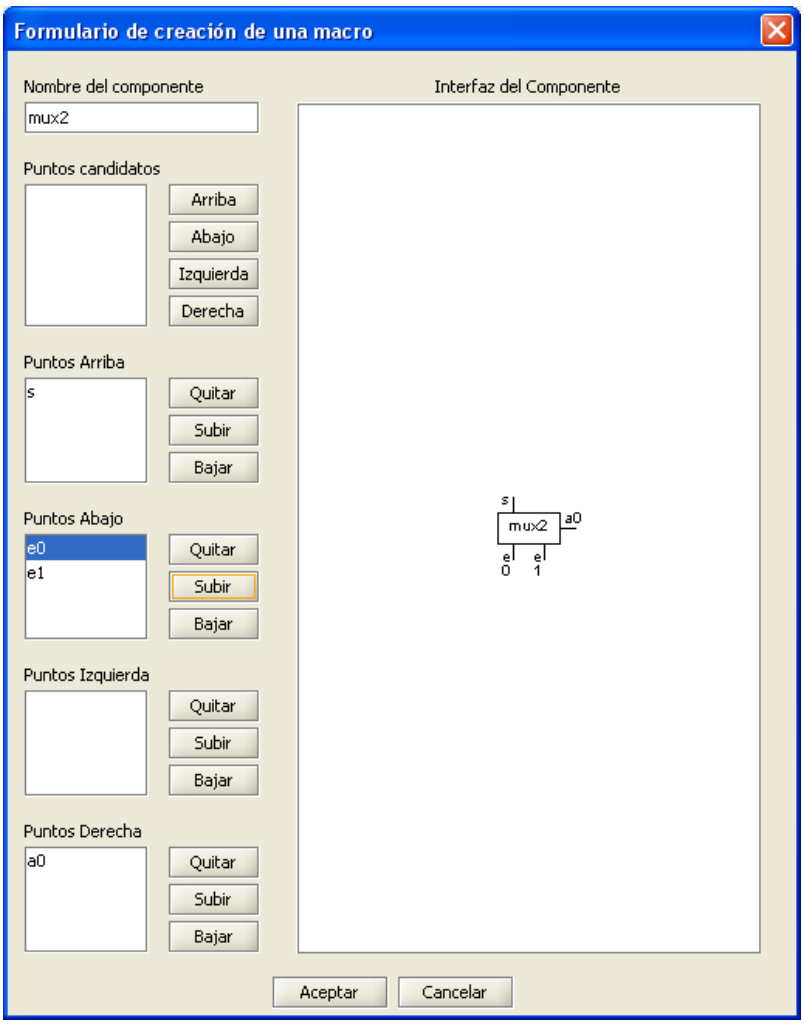

Figura 1.5: Creación del componente mux2

- ❏ 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.
- ❏ Sitúa el componente en el área de dibujo.
- ❏ Crea puntos para las entradas y salidas y dales nombre como se muestra en la figura 1.6. Coloca también 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, 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 correspondien-

te 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 verde hacia arriba de la barra de herramientas.
- ❏ 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. Haz doble clic sobre "mux2" y verás que se abre el fichero de componente "mux2". Puedes comprobar que ya no se muestra el valor de las entradas y los cables (no están coloreados). Eso es porque al abrir ese circuito se ha parado la simulación. Este comportamiento del SECD tiene su razón de ser en que puedes tener muchos componentes "mux2" en un mismo circuito y si haces clic en el árbol, no es posible saber cuál de esos componentes estás abriendo.

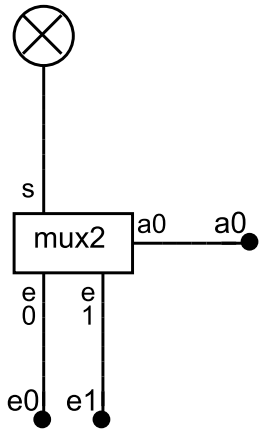

Figura 1.6: Prueba del componente mux2

5. Modificación de componentes definidos por el usuario

En este apartado se va a hacer un componente definido por el usuario incorrecto y luego se va a arreglar. El objetivo es aprender a solucionar problemas similares si te ocurren más adelante durante el desarrollo de las prácticas.

Vas a hacer, a continuación, una puerta NOR de cuatro entradas. Las entradas de la puerta se llamarán `e0`, `e1`, `e2` y `e3`, y la salida, `s`. ¿Cuál es la función lógica que debe implementar el circuito?[2]

[2]

Realiza los siguientes pasos:

- ❏ Guarda el proyecto como "nor4".

- ❏ Borra los elementos que tienes en el circuito principal.
- ❏ Sitúa puertas y cables para realizar el circuito mostrado en la figura 1.7. Este circuito, aunque combina puertas NOR de dos entradas, no implementa la función de una puerta NOR de cuatro entradas. ¿Qué función lógica implementa en realidad el circuito?[3]

3

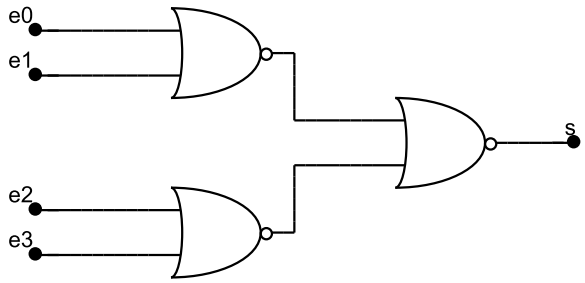

Figura 1.7: Circuito incorrecto de una puerta NOR de 4 entradas

- ❏ Crea el componente nor4 a partir de este circuito. Deberá tener el interfaz mostrado en la figura 1.8.

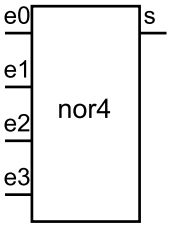

Figura 1.8: Componente nor4

- ❏ Borra el circuito principal, sitúa en él el componente nor4 creado y conecta puntos a las entradas y una bombilla a la salida. Etiqueta las entradas con el mismo nombre que en el componente.
- ❏ Guarda el proyecto.
- ❏ Antes de simular el circuito, rellena esta tabla escribiendo la salida que debería generar la función nor4 si estuviera correctamente implementada:

e3	*e2*	*e1*	*e0*	*s*
0	0	0	0	
0	0	1	0	
1	1	1	1	

- ❏ Abre el generador de entradas pulsando el botón de simular. Escribe las combinaciones mostradas en la tabla anterior. ¿cuántas salen distintas en el circuito de lo que has puesto en la tabla?[4]

4

- ❏ Vamos a arreglar el componente nor4. Para ello, haz doble clic sobre su nombre en el árbol de componentes que aparece a la izquierda de la ventana del SECD. Se deberá abrir el circuito del componente nor4. Para arreglarlo, debes borrar las dos puertas NOR de la izquierda y sustituirlas por dos puertas OR. Cuando lo hayas hecho y hayas vuelto a conectar los cables, guarda el circuito, con lo que habrás modificado el componente.
- ❏ Abre el circuito principal haciendo doble clic sobre el nombre del proyecto en el árbol de componentes.
- ❏ Vuelve a probar las combinaciones de la tabla anterior y comprueba que la puerta ahora se comporta como debería.

Aunque, como se ha visto, la aplicación permite modificar componentes definidos por el usuario, estas modificaciones a veces generan problemas, especialmente si se cambian entradas o salidas, ya que los circuitos que utilicen el componente con el interfaz viejo no podrán funcionar con el nuevo componente. Por ello, es muy recomendable que prestes atención a este guión e intentes no cometer fallos cuando desarrolles componentes nuevos.

6. Ejercicios adicionales

6.1. Ficheros en el disco

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

- `mux2.pro`, `mux2.pri` y `mux2.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `mux2`.
- `nor4.pro`, `nor4.pri` y `nor4.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `nor4`. Comprueba abriendo con el SECD el proyecto `nor4.pro` que incluye los dos componentes que has hecho: `mux2` y `nor4`.

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`. Cuándo presiones el botón ↓ para que se genere esa combinación, ¿de qué color deberá quedar la bombilla?[5]

5

✏ Crea un proyecto nuevo ("Archivo→Nuevo Proyecto"). A continuación, crea el circuito de la figura 1.9, que tiene tres entradas y una salida.

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

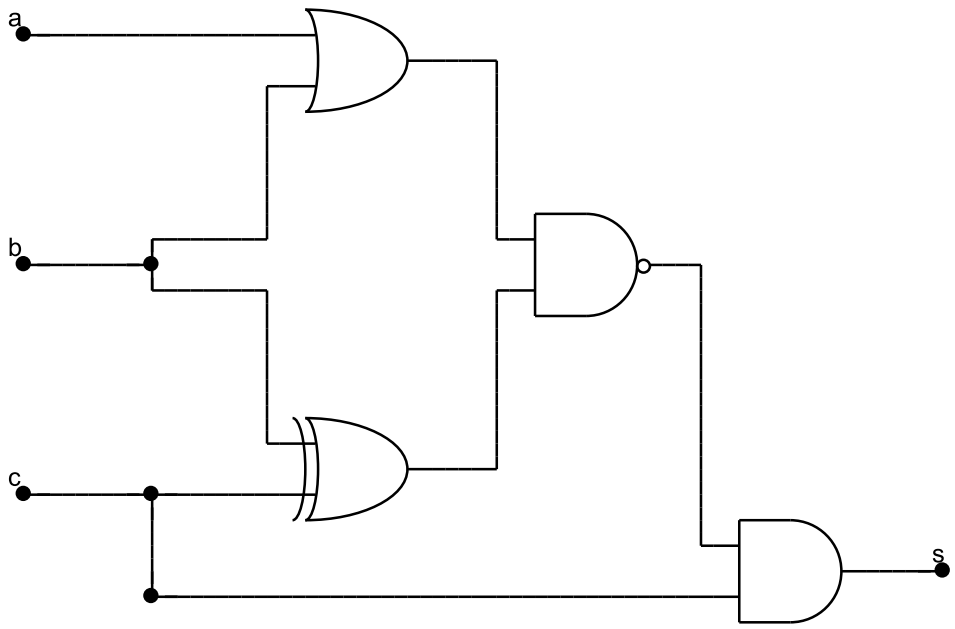

Figura 1.9: Circuito del ejercicio propuesto

a	*b*	*c*	*s*
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

Construcción de componentes básicos: El multiplexador y el sumador elemental

Objetivos

El objetivo de esta sesión es la construcción de dos componentes básicos que serán utilizados en sesiones posteriores. Se trata de un multiplexador de 4 canales y de un sumador elemental.

Naturalmente, el objetivo no es sólo la construcción de estos elementos, sino también su verificación mediante el generador de entradas y la comprensión de su funcionamiento.

Conocimientos y materiales necesarios

- Comprensión de la función de un multiplexador de cuatro canales. El alumno debe ser capaz de generar sobre el papel la tabla de verdad de este componente.
- Capacidad para realizar sobre el papel una suma de tres cantidades de 1 bit cada una de ellas, lo que involucra el concepto de acarreo.
- Proyecto "nor4", previamente generado por el alumno en la sesión 1 de este bloque. Este proyecto contiene el componente `mux2` que será necesario para la creación del multiplexador de cuatro canales.

Asimismo, se recomienda leer los puntos 1 y 2 de este documento antes del día de la práctica, para agilizar la realización de la misma, ya que algunos de los conceptos expuestos en este texto pueden necesitar una lectura atenta.

Desarrollo de la práctica

1. Construcción de un multiplexador de cuatro canales de entrada

El componente que queremos crear, 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 `e0`, `e1`, `e2` y `e3`, y a las líneas de selección `a0` y `a1` (el subíndice indica el peso del correspondiente bit). La salida será `s`.

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 en la práctica anterior.

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.

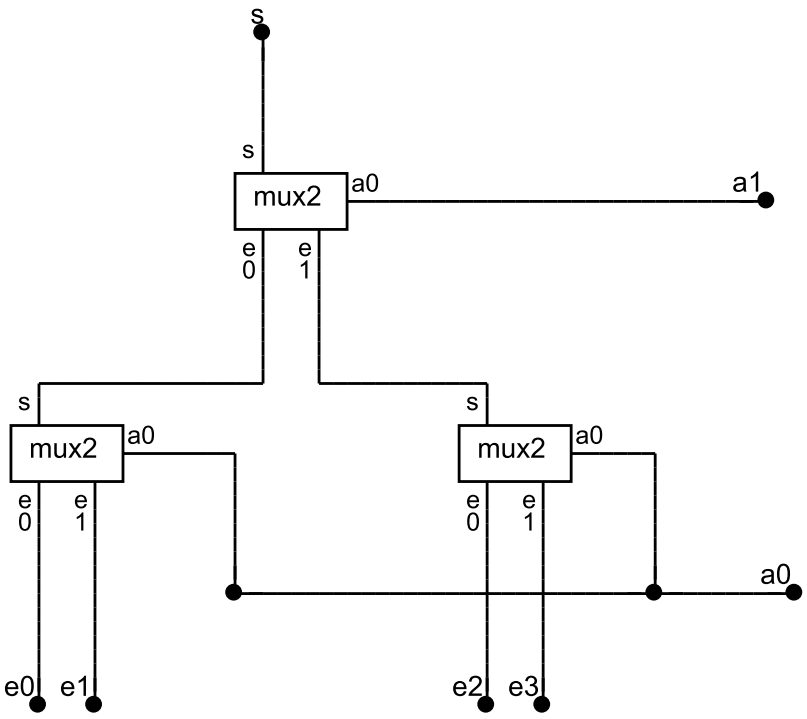

Figura 2.1: Conexión de tres multiplexadores de 2 canales para crear un multiplexador de 4 canales

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 2.1 se muestra la forma de conectar entre sí estos tres componentes. Debes realizar los siguientes pasos:

❑ Arranca el programa `SECD` de la forma habitual y carga el proyecto "nor4" creado en la sesión 1 de este bloque. Este proyecto contiene el componente `mux2`.

❑ 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 2.1 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 2.2

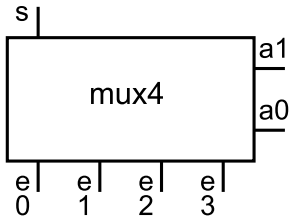

Figura 2.2: Macro del multiplexador de 4 canales

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 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. Coloca también una bombilla a la salida.
- ❏ Guarda el proyecto, 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	1	0	0	0	1	
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.

2. Construcción de un sumador elemental

Un sumador elemental es un componente que recibe tres entradas y genera dos salidas. Un esquema de este componente se muestra en la figura 2.3. Las tres entradas (que denominaremos `ai`, `bi` y `ci-1`) representan tres cantidades de 1 bit cada una, que deben ser sumadas por el circuito. Las dos salidas representan los dos bits de la respuesta, `si` es el resultado de la suma y `ci` el "acarreo".

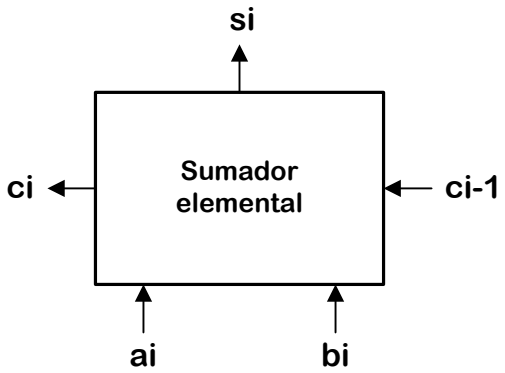

Figura 2.3: El sumador elemental como "caja negra"

Para crear este componente, al igual que ocurría en el caso del multiplexador de cuatro canales, tenemos dos opciones:

- Escribir la tabla de verdad del circuito y, a partir de ella, deducir su función lógica.
- Tratar de crear un componente más simple aún (el semisumador, que suma sólo dos cantidades de 1 bit, en lugar de tres) y pensar cómo combinar varios de ellos para lograr el resultado deseado.

En este caso el primer enfoque es admisible, pues la tabla de verdad tendría sólo ocho filas. De todas formas, ya que este enfoque es automático y no conlleva una dificultad excesiva, se deja como ejercicio para el alumno interesado. En este punto utilizaremos el segundo enfoque. Para ello debes hacer lo siguiente:

- ❏ Carga el proyecto "mux4".
- ❏ Guarda el proyecto con el nombre "semisum" y borra los elementos que tienes en el circuito principal.
- ❏ Conecta una puerta lógica AND y otra XOR como se muestra en la figura 2.4 en la página siguiente. Escribe, para cada posible combinación de las entradas `a` y `b`, el valor de ambas salidas (hazlo de memoria, no necesitas usar el generador de entradas para un circuito tan simple). Observarás que, efectivamente, el resultado en la salida `suma` es la suma de los dos bits de entrada, y la salida `carry` es el acarreo de esa suma.

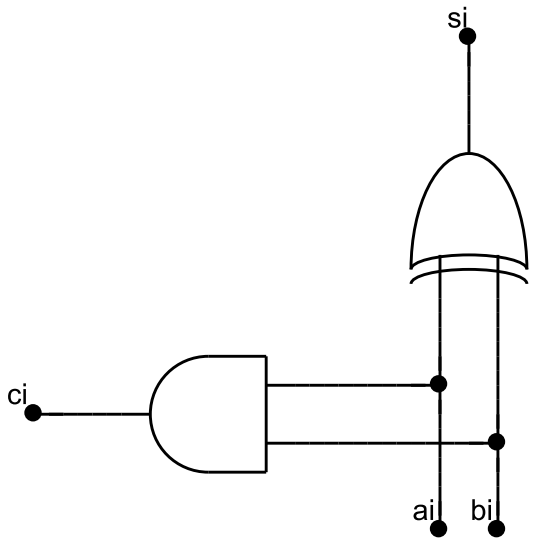

Figura 2.4: Esquema interno de un semisumador

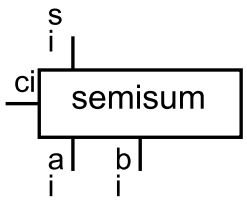

Figura 2.5: Macro del semisumador

- ❏ Convierte el circuito anterior en una *macro* y llámala *semisum*. Realiza las conexiones de tal forma que el componente resultante tenga las entradas y salidas como en la figura 2.5.

A continuación, vamos a utilizar el componente `semisum` para crear el sumador de 1 bit.

- ❏ Guarda con el proyecto con el nombre "sum1" y borra los elementos que tienes en el circuito principal.
- ❏ Conecta dos semisumadores junto con una puerto OR, tal y como se muestra en la figura 2.6 en la página siguiente.
  Observa cómo primero se suman dos de las entradas en un semisumador y el resultado es sumado a la tercera entrada en el otro semisumador. Si se produce acarreo en cualquiera de estas sumas parciales habrá un acarreo como resultado final, de ahí la puerta OR.
- ❏ Convierte el circuito anterior en una *macro* y llámala `sum1`. Realiza las conexiones de tal forma que el componente resultante tenga las entradas y salidas como en la figura 2.7 en la página siguiente:

Ahora ha llegado el momento de comprobar si el circuito funciona realmente como un sumador.

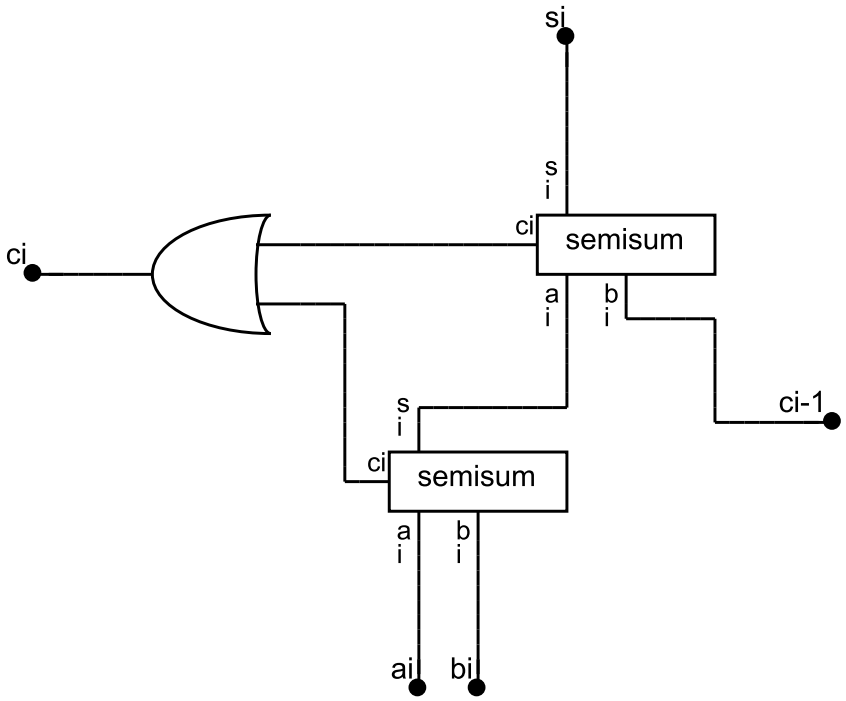

Figura 2.6: Esquema interno del sumador de 1 bit

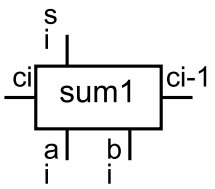

Figura 2.7: Macro del sumador de 1 bit

- ❏ 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 `sum1`. 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 salidas, y dales nombre. Coloca también una bombilla en cada una de las salidas.
- ❏ Sin usar aún el simulador, rellena la siguiente tabla de verdad del sumador (usa para ello tu conocimiento adquirido en las clases de teoría de qué debería haber en la salida como resultado de la suma de las entradas):

ci-1	ai	bi	ci	si
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

❑ Una vez que has rellenado la tabla anterior, pulsa el botón de simular, te aparecerá el generador de entradas. Presiona el botón de "Rellenar" para generar todas las combinaciones posibles. A continuación, verifica que tus respuestas han sido correctas comparándolas con el valor de las salidas del simulador.

3. Ejercicios adicionales

3.1. Ficheros en el disco

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

- `mux4.pro`, `mux4.pri` y `mux4.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `mux4`.
- `semisum.pro`, `semisum.pri` y `semisum.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `semisum`.
- `sum1.pro`, `sum1.pri` y `sum1.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `sum1`.

3.2. Ejercicios

✏ Examina de nuevo la figura 2.1 en la página 13 y asegúrate de que comprendes cómo funciona el circuito representado. Observa que el multiplexador de arriba elige como su salida una de las dos salidas de los multiplexadores de abajo. Y cada uno de estos elige entre dos posibles líneas de entrada. Para verificar si has comprendido bien su funcionamiento, intenta diseñar ahora (sobre el papel) un multiplexador de 8 canales de entrada (y, por tanto, tres líneas de selección), usando para ello dos `mux4` y un `mux2`.

✏ (Opcional) Construye tu diseño con el SECD y haz algunas pruebas para verificar que funciona correctamente. Por ejemplo, introduce en las entradas de selección la combinación `011` (usando el componente "Uno lógico" para el `1` y "Cero lógico" para el `0`), y utiliza el generador de entradas para introducir la secuencia `01010101` en la entrada `e3`. Simula las filas de la tabla de verdad donde has introducido la secuencia para `e3` ¿luce la bombilla en la secuencia apropiada?

✏ Si en el sumador que has desarrollado combinando dos semisumadores se cambiara la puerta OR por una puerta XOR, el circuito así resultante ¿seguiría comportándose como un sumador? ¿por qué? Compruébalo.

✏ En el primer ejercicio se pedía diseñar un multiplexador de ocho canales usando dos `mux4` y un `mux2`. Ésta es la forma más simple de hacerlo, pero también podría lograrse mediante cuatro `mux2` y un `mux4` que elegiría entre las salidas de los cuatro `mux2`. Diseña sobre el papel cómo se haría el multiplexador de ocho canales utilizando este enfoque, y (opcionalmente) realízalo con el SECD para comprobar su funcionamiento con las mismas combinaciones para las entradas que en el segundo ejercicio.

SESIÓN 3

Construcción de un sumador de cuatro bits

Objetivos

En esta sesión se construirá un circuito capaz de sumar dos cantidades binarias de cuatro bits cada una, generando un resultado, también de cuatro bits, y completando el resultado con un indicador de *carry* (que indica desbordamiento en operaciones con números naturales) y un indicador de *overflow* (que indica desbordamiento en operaciones con números enteros codificados en complemento a 2).

El sumador de cuatro bits se construirá combinando cuatro sumadores elementales. El detector de *overflow* se sintetizará a partir de su tabla de verdad, de la que se obtendrá la función lógica y, finalmente, el circuito necesario.

Al finalizar esta sesión el alumno deberá ser capaz de sintetizar cualquier circuito combinacional a partir de su tabla de verdad y de comprender la suma de cantidades en binario natural y en complemento a 2, siendo capaz de anticipar el resultado y la activación de los bits de *carry* y de *overflow*.

Conocimientos y materiales necesarios

- Comprensión de cómo se realiza una suma en binario "sobre el papel", pues ese mismo esquema será el que se aplique para realizarla mediante un circuito digital.
- Capacidad de codificar cantidades positivas y negativas usando la representación en complemento a 2.
- Comprensión del concepto de desbordamiento y conocimiento del método para detectar el *overflow* en complemento a 2, mediante el examen de los bits de signo de los operandos y del resultado.
- Proyecto "sum1", previamente generado por el alumno en la sesión 2 de este bloque. Este proyecto contiene, entre otras, la macro `sum1`, necesaria para desarrollar el sumador binario de cuatro bits.

Desarrollo de la práctica

1. Construcción de un sumador de cuatro bits

Es importante comprender que un sumador de cuatro bits no es un circuito que suma cuatro cantidades de 1 bit, sino dos cantidades de cuatro bits cada una expresadas en binario. Por tanto, deberá tener ocho entradas (cuatro bits para cada una de las cantidades) y cuatro salidas (los cuatro bits del resultado).

Adicionalmente, incluiremos una entrada más que llamaremos *cin* (*carry in*), y una salida más que llamaremos *cout* (*carry out*). La salida *cout* será el bit de *carry* del resultado, mientras que la entrada *cin* permite añadir `1` ó `0` a la suma, lo cual tendrá su utilidad más adelante.

El sumador de cuatro bits, como "caja negra", se muestra en la figura 3.1. Fíjate en la disposición de las entradas y salidas.

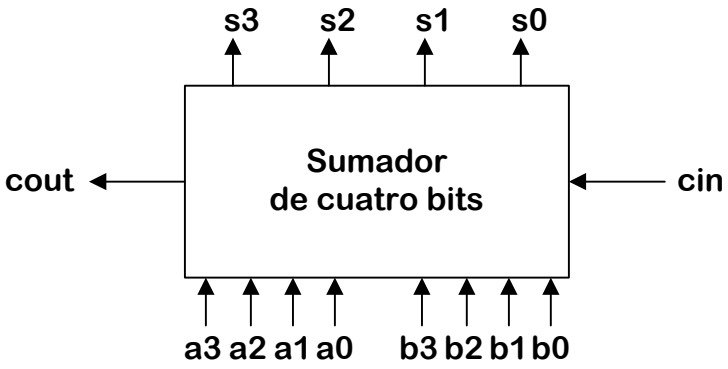

Figura 3.1: El sumador de cuatro bits como "caja negra"

- ❏ Observa cómo se realiza una suma binaria "sobre el papel". Para ello realiza la suma de dos cantidades de cuatro bits, como por ejemplo: `0101 + 0111`. Expresa el resultado en binario natural[1].
- ❏ Fíjate que en cada columna de la suma estás sumando tres bits: dos de esos bits son los datos a sumar y el tercero es el acarreo que se ha producido en la suma de la columna anterior. El resultado de sumar cada columna es un bit (que forma parte de la respuesta) y un acarreo que pasa a la columna siguiente.

1

Un sumador elemental, como el `sum1` desarrollado en la sesión anterior, es capaz de sumar tres bits, por lo que podría hacer la suma de una columna. Ya que hay cuatro columnas a sumar, podemos usar un sumador elemental para cada una de las sumas.

- ❏ Arrancar el programa `SECD` en la forma habitual y carga el proyecto "sum1", creado en la sesión 2.

- ❏ Guarda el proyecto como "sum4" y borra los elementos que tienes en el circuito principal.
- ❏ Conecta cuatro sumadores elementales (`sum1`) de modo que puedan resolver una suma de cuatro bits como la que has hecho sobre el papel.
- ❏ Convierte el circuito en una *macro* llamada `sum4`. Configura las entradas y salidas tal y como se muestran en la figura 3.2.

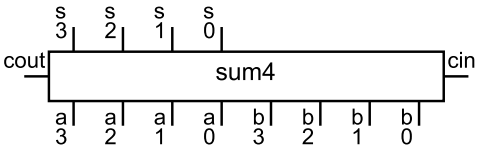

Figura 3.2: Macro del sumador de 4 bits

- ❏ Borra el circuito que tienes en pantalla.
- ❏ Selecciona en la barra de herramientas el componente `sum4` y sitúalo en el área de dibujo.
- ❏ Crea puntos para las entradas y salidas, y dales nombre. Coloca también una bombilla a cada bit del resultado y otra bombilla más al acarreo resultante final.
- ❏ La entrada *cin* no es utilizada de momento, pero no puede dejarse "al aire". Obligaremos a que esta entrada tenga el valor lógico `0` conectando a ella el componente "Cero lógico".
- ❏ Programa el generador para que el circuito así creado calcule la suma `0101 + 0111`, que es la que has hecho antes sobre el papel. Al pulsar [↓] las bombillas de salida mostrarán el resultado de la suma. ¿Coincide con lo que tú has obtenido sobre el papel?[2] Si no es así, revisa tu circuito.

[2]

1.1. Pruebas con el sumador de cuatro bits

Primero probaremos si el sumador resuelve correctamente sumas expresadas en binario natural.

- ❏ Haz que el sumador resuelva la suma 6+3. Para ello debes codificar el 6 y el 3 en binario, usando cuatro bits, y después programar el generador de entradas para que envíe a cada sumador elemental un bit a sumar de cada dato.
- ❏ ¿Cuál es el número más grande que puede representarse con cuatro bits?[3] ¿Qué pasará si intentamos realizar una suma cuyo valor exceda este máximo?[4] Vamos a intentarlo. Elige dos números cualesquiera cuya suma exceda el valor del máximo representable. Programa el generador de entradas para que envíe estos datos al sumador y pulsa [↓]. ¿Qué ha salido como resultado? Escríbelo en base 10[5].

[3]

[4]

[5]

¿Qué ha ocurrido con el bit de *carry*?[6] La bombilla de *carry* debe haberse encendido para indicar que el resultado no cabe, interpretado en binario natural. Si no fuera así revisa tu circuito.

[6]

Ahora probaremos si el sumador es capaz de resolver también sumas realizadas en complemento a 2.

- ❏ Haz que el sumador resuelva la suma 2+(-2). Para ello debes codificar el 2 y el -2 en complemento a 2, usando cuatro bits. Programa el generador de entradas de acuerdo con estos datos y pulsa [↓]. Puesto que el resultado de 2+(-2) es cero, todas las bombillas de salida deberían permanecer apagadas.
- ❏ Haz que el sumador resuelva la suma 5+(-3). El resultado debe ser 2. Comprueba que es así.
- ❏ Usa el sumador para calcular (-5)+3. En este caso el resultado es negativo (-2). Esto lo reconocerás porque la bombilla más significativa de la respuesta estará encendida.
- ❏ ¿Cuál es el mayor positivo representable en complemento a 2 con cuatro bits?[7] ¿Qué pasará si el resultado de la suma excede este máximo?[8] Vamos a comprobarlo. Elige dos números positivos cualesquiera que al ser sumados excedan el máximo representable. Usa el sumador con estos datos ¿qué sale en el resultado?[9]

[7]

[8]

[9]

El último experimento muestra que si el resultado no cabe la respuesta será errónea porque, entre otras cosas, es de signo contrario al que debería ser. Pero en este caso la bombilla asociada al *carry* no sirve para detectar el problema. Es necesario construir un detector de *overflow*, lo cual se hará en la segunda parte de esta sesión.

2. Construcción de un detector de *overflow*

El *overflow* ocurre al sumar dos cantidades en complemento a 2 sólo en los siguientes casos:

- Si las cantidades a sumar son ambas positivas y el resultado sale negativo.
- Si las cantidades a sumar son ambas negativas y el resultado sale positivo.

Por tanto, para detectar el problema del *overflow* debemos mirar los bits de signo de los operandos y del resultado. Según qué combinaciones de signos encontremos, habrá *overflow* o no.

- ❏ Completa la tabla siguiente, indicando para qué combinaciones de signos ha ocurrido *overflow*.

an-1	bn-1	sn-1	ov
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- ❏ A partir de la tabla de verdad anterior, deduce la función lógica `ov` = f(`an-1`, `bn-1`, `sn-1`).
- ❏ Crea un nuevo proyecto y guardalo con el nombre "gen-ov-s".
- ❏ Construye, mediante puertas AND, OR y NOT, un circuito que lleve a cabo la función lógica que has deducido anteriormente.
- ❏ Convierte el circuito que has creado en una macro, dándole el nombre `gen-ov-s` (Generador de *overflow* para la suma). Asegúrate de que las entradas y salidas de este nuevo componente quedan tal y como se muestra en la figura 3.3.

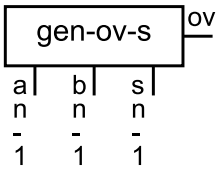

Figura 3.3: Macro del generador de *overflow* de la suma

- ❏ Guarda el proyecto.

2.1. Verificación del detector de *overflow*

Vamos a comprobar ahora si el detector de *overflow* detecta los casos en los que el resultado de la operación no cabe (cuando los datos están representados en complemento a 2). Para ello debes hacer lo siguiente:

- ❏ Carga el proyecto "sum4".
- ❏ Agrega el componente `gen-ov-s` al proyecto. Para ello pulsa con el botón derecho del ratón sobre la carpeta del proyecto, dentro del árbol de fichero, y selecciona la opción "Añadir archivo existente". A continuación, selecciona el fichero `gen-ov-s.cir`, que es donde está implementado el generador de *overflow* de la suma.
- ❏ Lleva a la zona de trabajo el componente *gen-ov-s* que has creado antes y conecta sus entradas apropiadamente al circuito que ya tenías en pantalla. Conecta una bombilla a la salida del generador de *overflow*.

- ❏ Programa el generador de entradas para que envíe al sumador dos cantidades positivas cuya suma exceda el rango del complemento a 2 para cuatro bits. Pulsa [↓] para que estos datos sean enviados al sumador. La bombilla conectada a la salida del generador de *overflow* debe encenderse. Si no lo hace revisa este componente.
- ❏ Haz, además, las siguientes pruebas:
  - Suma una cantidad positiva con una negativa (la bombilla del detector debe permanecer apagada).
  - Suma una cantidad negativa con una positiva (la bombilla del detector debe permanecer apagada).
  - Suma dos cantidades positivas cuyo resultado quepa en el rango del complemento a 2 (la bombilla del detector debe permanecer apagada).
  - Suma dos cantidades negativas cuyo resultado quepa en el rango del complemento a 2 (la bombilla del detector debe permanecer apagada).
  - Suma dos cantidades negativas cuyo resultado exceda el menor negativo representable en complemento a 2 con 4 bits (la bombilla del detector debe encenderse).
- ❏ Si tu detector pasa todas las pruebas anteriores, funciona correctamente.

2.2. Construcción de un detector de *overflow* genérico

A continuación, vamos a construir un detector de *overflow* genérico. Esto es, un detector de *overflow* que sirva tanto para las operaciones de suma como para las de resta. Debes seguir los pasos siguientes pasos:

- ❏ Carga el proyecto "sum4".
- ❏ Guarda el proyecto como "gen-ov".
- ❏ Borra los elementos que tienes en el circuito principal.
- ❏ El detector de *overflow* genérico se diferencia del detector de *overflow* para la suma en que posee una entrada adicional que indica si se está realizando una suma o una resta. Puede construirse a partir de su tabla de verdad, o bien, a partir del detector de *overflow* para la suma, `gen-ov-s`. Esta última alternativa será la que seguiremos debido a su mayor simplicidad. El circuito correspondiente al detector de *overflow* genérico se obtiene añadiendo simplemente una puerta XOR a la entrada `bn-1` del detector de *overflow* para la suma, tal como se indica en la figura 3.4 en la página siguiente.
  Si `resta=1`, la puerta XOR funciona como un inversor que lleva a la entrada `bn-1` del `gen-ov-s` el signo contrario al del operando `bn-1`. Si por el contrario `resta=0`, `bn-1` no se modifica. En este último caso el detector de *overflow* genérico funciona exactamente igual que el detector de *overflow* para la suma. Una vez construido el circuito selecciónalo y crea la macro `gen-ov`, con las entradas y salidas situadas tal como aparecen en la figura 3.5 en la página siguiente.

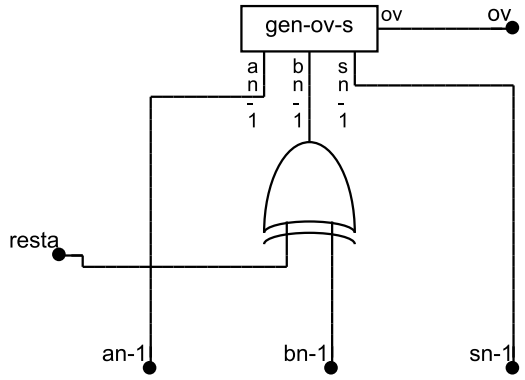

Figura 3.4: Esquema interno del detector de *overflow* genérico

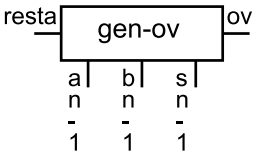

Figura 3.5: Macro del detector de *overflow* genérico

3. Ejercicios adicionales

3.1. Ficheros en el disco

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

- `sum4.pro`, `sum4.pri` y `sum4.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `sum4`.
- `gen-ov-s.pro`, `gen-ov-s.pri` y `gen-ov-s.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `gen-ov-s`.

3.2. Ejercicios

✏ Rellena la tabla siguiente, haciendo las operaciones "en papel", es decir, sin utilizar el `SECD`. Para ello deberás realizar lo siguiente:

- Codificar cada dato, en complemento a 2, con 4 bits. Escribe esta codificación en las columnas tituladas "**A** (binario)" y "**B** (binario)".
- Realizar la suma de las cantidades binarias que has obtenido, escribiendo los cuatro bits del resultado en la columna titulada "Suma (binario)".
- Decodifica este resultado, pasándolo de binario a decimal y teniendo en cuenta que está codificado en complemento a 2. Escribe lo que obtengas en la columna "Suma (decimal)".

- Escribe también el acarreo resultante de la suma (aunque este acarreo no significa nada cuando se opera en complemento a 2) y el *overflow* resultante (que indica si el resultado es incorrecto por haber superado el rango).

A	**A** (binario)	**B**	**B** (binario)	Suma (binario)	Suma (decimal)	C_{out}	*Ov*
2		4					
-2		4					
-5		-5					
3		5					

✏ (Opcional) Carga el proyecto "sum4" en el SECD y programa el generador de entradas para que el sumador realice las cuatro sumas anteriores una tras otra. Comprueba que tanto el resultado de cuatro bits como las bombillas de *carry* y *overflow* coinciden con lo que tú habías escrito en la tabla.

✏ Observa de nuevo las columnas tituladas "**A** (binario)" y "**B** (binario)" de la tabla anterior. Los cuatro bits que contienen son la codificación de una cantidad en complemento a 2, pero también podrían ser la codificación de una cantidad en binario natural. Si así fuera ¿cuáles serían esas cantidades? Escríbelas en la tabla siguiente:

A (binario)	**A** (natural)	**B** (binario)	**B** (natural)

✏ Señala en esta tabla qué sumas darían un resultado que no cabría en cuatro bits (interpretados como binario natural). Comprueba que las filas que has señalado coinciden justamente con aquellas que producían *carry* en la primera tabla. Por tanto el *carry* indica cuándo el resultado no cabe, interpretando todos los datos como binario natural, mientras que el *overflow* indica cuándo el resultado no cabe interpretando todos los datos como complemento a 2.

✏ Piensa cómo conectarías dos *sum4* para lograr un sumador de cantidades de ocho bits. ¿Dónde debería conectarse el detector de *overflow*?

✏ (Opcional) Construye con el `SECD` el diseño que has ideado, y utilízalo para sumar las cantidades 37 y 20, por ejemplo.

- ❑ ¿Cuál es el resultado obtenido? Expresarlo en decimal[10].
- ❑ ¿Se ha producido *carry*?[11]
- ❑ ¿Se ha producido *overflow*?[12]

10

11

12

SESIÓN 4

Construcción de una ALU de 4 bits

Objetivos

El objetivo de esta sesión práctica es la construcción y simulación de la ALU de la CPU teórica. Esta ALU es capaz de realizar las operaciones aritméticas suma y resta, así como las operaciones lógicas AND, OR y XOR.

Para no complicar en exceso la simulación, se considera una ALU de sólo 4 bits, pero totalmente análoga a la ALU de la CPU teórica (de 16 bits).

Después de realizar la sesión práctica el alumno debería ser capaz de:

- Construir la ALU sin necesidad del esquema de la misma.
- Simular las operaciones SUMA, RESTA, AND, OR y XOR, con dos operandos de 4 bits cualesquiera.

Conocimientos y materiales necesarios

- Apuntes del tema de sistemas digitales donde se explica la construcción de la ALU. El alumno debe leer con detenimiento éstos antes de asistir a la sesión práctica. La lectura de los mismos le permitirá comprender el funcionamiento interno de la ALU de la CPU teórica.
- Proyecto "gen-ov", previamente generado por el alumno en la sesión 3 de este bloque. Este proyecto contiene, entre otras, las macros `mux2`, `mux4`, `sum1`, `gen-ov-s` y `gen-ov`.

Desarrollo de la práctica

1. Introducción

En esta sesión se pretende construir una ALU de 4 bits análoga a la ALU de 16 bits de la CPU teórica. Para ello se construirá, en primer lugar, una ALU de 1 bit. A continuación, empleando cuatro ALUs de 1 bit, un detector de *overflow* y cierta lógica adicional, se construirá la ALU de 4 bits. Finalmente, se probará la ALU simulando varias operaciones.

2. Construcción de la ALU

Debes realizar los pasos siguientes para la construcción de la ALU:

- ❏ Arranca el programa `SECD` en la forma habitual y carga el proyecto "gen-ov", creado en la sesión 3.
- ❏ Guarda el proyecto como "alu4" y borra los elementos que tienes en el circuito principal
- ❏ Construye el circuito de la ALU de 1 bit. Puedes usar como referencia la figura 4.1.

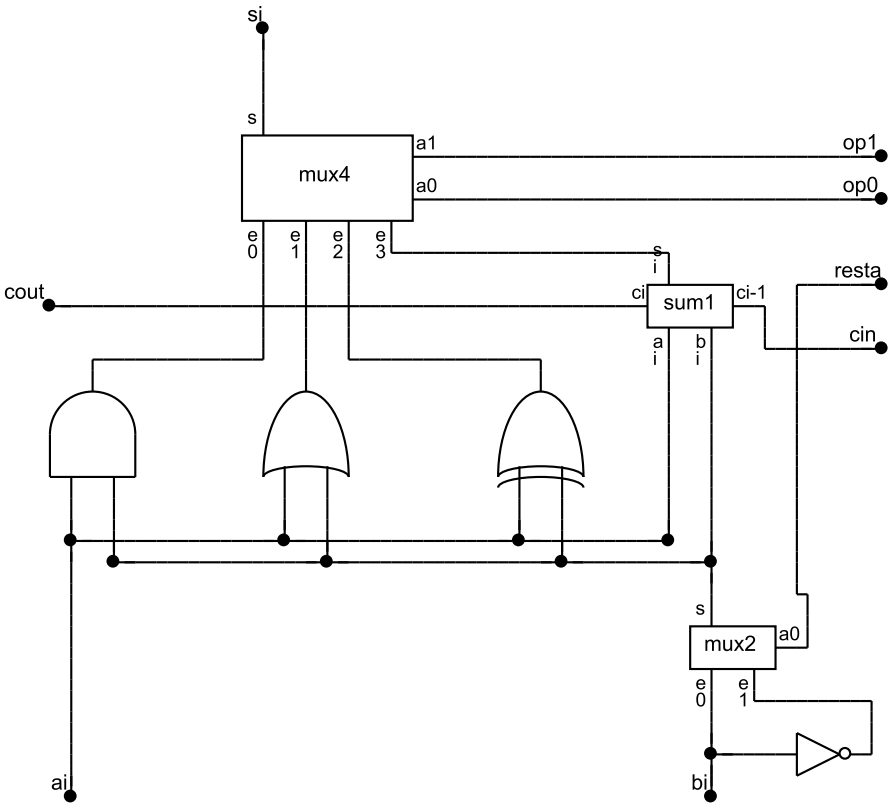

Figura 4.1: Esquema interno de la la ALU de 1 bit

- ❏ Crea la macro `alu1`, con las entradas y salidas situadas tal como aparecen en la figura 4.2 en la página siguiente.
- ❏ Repasa de nuevo que las entradas y salidas de la macro que has creado coinciden con las de la figura 4.2 en la página siguiente. Es un error muy común cambiar la posición de la entrada `resta` con la entrada `cin`.

Ahora vamos a empezar con la ALU de 4 bits. Vamos a implementarla en dos pasos, en primer lugar diseñaremos una ALU sin bits de estado y, a continuación partiendo de ésta, se implementará la ALU completa. Para ello, se deben realizar los siguientes pasos:

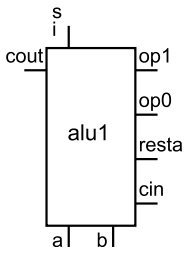

Figura 4.2: Macro de la ALU de 1 bit

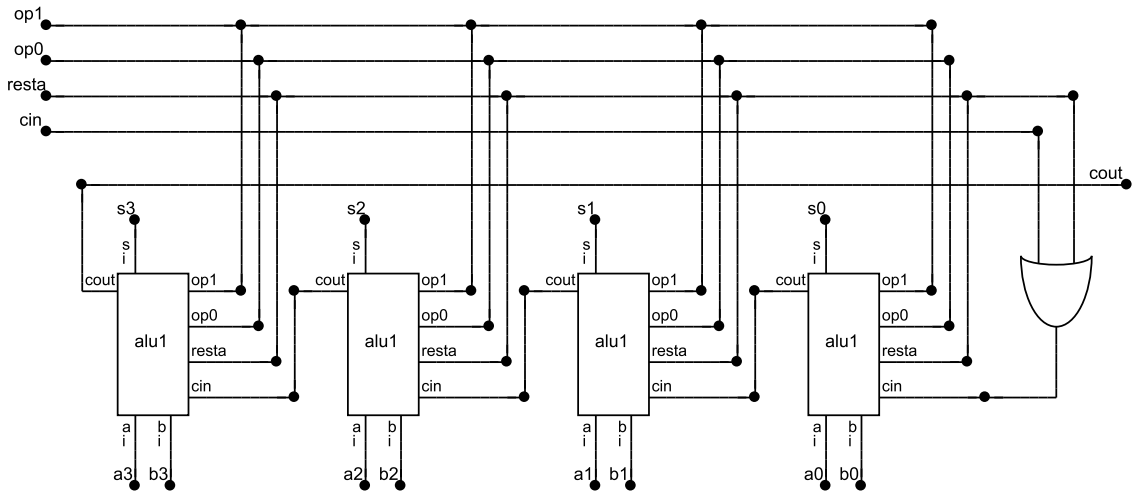

Figura 4.3: Esquema interno de la ALU de 4 bits sin bits de estado

- ❏ Borra los elementos que tienes en el circuito principal.
- ❏ Construye el circuito de la ALU de 4 bits pero sin bits de estado.
- ❏ Si quieres, te puedes basar como referencia a la hora de colocar los componentes y conectar los cables en la figura 4.3. En caso de que el SECD de error al conectar dos puntos remotos, la única solución es añadir puntos intermedios manualmente y facilitar al programa el enrutamiento de las líneas de conexión.
- ❏ Una vez construido el circuito crea la macro `alu4-nf` (ALU 4 bits No Flags, es decir, ALU de 4 bits sin bits de estado), con las entradas y salidas situadas tal como aparecen en la figura 4.4.

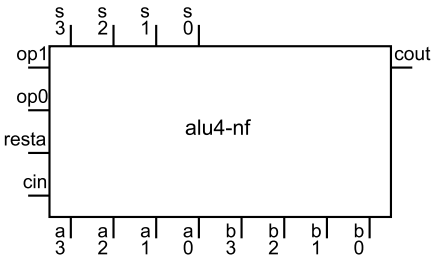

Figura 4.4: Macro de la ALU de 4 bits sin bits de estado

Por último, vamos a construir la ALU completa con los bits de estado realizando los siguientes pasos:

- ❏ Borra los elementos que tienes en el circuito principal.
- ❏ Construye el circuito de la ALU de 4 bits, en este caso con bits de estado. Para ello debes partir de la macro `alu4-nf`.
- ❏ Una vez construido el circuito crea la macro `alu4`, con las entradas y salidas situadas tal como aparecen en la figura 4.5.

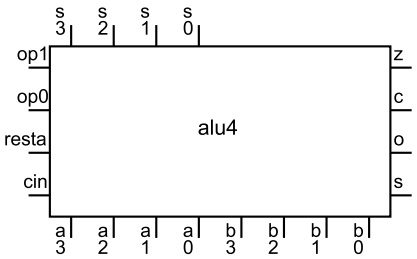

Figura 4.5: Macro de la ALU de 4 bits

3. Pruebas de la ALU

En esta sección se simularán varias operaciones con la ALU recién construida, tal como se indica a continuación.

- ❏ Rellena la tabla siguiente, haciendo las operaciones "en papel", es decir, sin utilizar el `SECD`.

A	B	Operación	Resultado	z	c	o	s
0110	0111	Suma					
0110	0111	Resta					
0110	0111	OR					
0110	0111	AND					
0110	0111	XOR					

- ❏ Borra el circuito que tienes en pantalla.
- ❏ Selecciona en la barra de herramientas el componente `alu4` y sitúalo en el área de dibujo.
- ❏ Crea puntos para las entradas y salidas, y dales nombre. Coloca también una bombilla a cada salida.
- ❏ Pulsa el botón de simular, te aparecerá el generador de entradas. Escribe los valores necesarios en las entradas para realizar las operaciones que se indicaban en la tabla anterior.
- ❏ Verifica que tus respuestas son correctas comparándolas con el resultado que ofrece el SECD.

4. Ejercicios adicionales

4.1. Ficheros en el disco

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

- `alu4.pro`, `alu4.pri` y `alu4.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `alu4`.
- `alu1.cir`, que contiene el componente `alu1`.
- `gen-ov.cir`, que contiene el componente `gen-ov`.

4.2. Ejercicios

✏ Rellena la tabla siguiente, haciendo las operaciones "en papel", es decir, sin utilizar el `SECD`.

`A`	`B`	Operación	Resultado	`z`	`c`	`o`	`s`
0010	1100	Suma					
0010	1100	Resta					
0010	1100	OR					
0010	1100	AND					
0010	1100	XOR					

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 positivos cuya suma genere acarreo. ¿Existen tales operandos? ¿Por qué?

✏ Busca dos operandos negativos cuya suma no genere acarreo. ¿Existen tales operandos? ¿Por qué?

✏ Elige, sobre el papel y de forma razonada, dos operandos 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 positivos cuya resta genere *overflow*. ¿Existen tales operandos? ¿Por qué?

SESIÓN 5

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 teórica.

Para no complicar en exceso la simulación se considera una CPU de sólo 4 bits, pero totalmente análoga a la CPU teórica (de 16 bits). 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 teórica.

Conocimientos y materiales necesarios

- Comprensión por parte del alumno del funcionamiento de un registro y de la ALU de la CPU teórica.
- Proyecto "cpu", ubicado en un directorio que el profesor indicará al principio de la sesión práctica.
- Proyecto "alu4", previamente generado por el alumno en la sesión 2-4. 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:

- ❏ Copia los ficheros `cpu.pro` y `cpu.pri` 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 5.1.

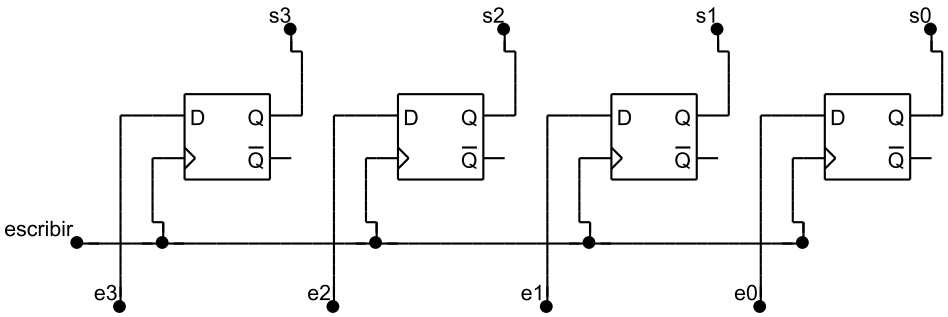

Figura 5.1: Esquema interno de un registro de 4 bits

- ❏ 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 5.2.

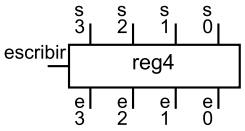

Figura 5.2: Macro del registro de 4 bits

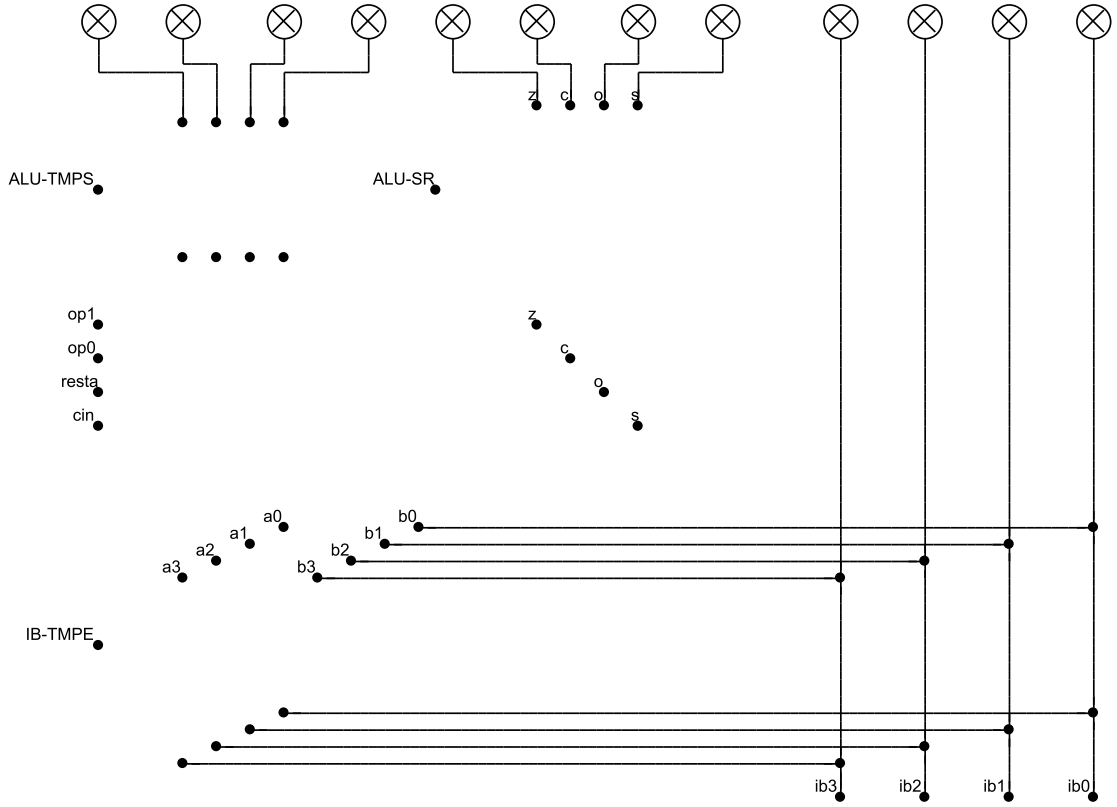

Figura 5.3: Circuito de la CPU

- ❏ Carga el proyecto "cpu". Aparecerá en la pantalla el circuito de la figura 5.3.

  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.
  - 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.
- ❏ Sitúa en el circuito, en la posición indicada por el texto ALU, una copia de la macro `alu4`.
- ❏ Sitúa en el circuito, en la posición indicada por el texto TMPE, una copia de la macro `reg4`.
- ❏ Sitúa en el circuito, en la posición indicada por el texto TMPS, una copia de la macro `reg4`.
- ❏ Sitúa en el circuito, en la posición indicada por el texto SR, una copia de la macro `reg4`.
- ❏ Realiza las conexiones apropiadas entre las macros y los puntos de conexión del circuito. Una vez llegados a este punto el circuito se corresponde con el diagrama de bloques de la figura 5.4.

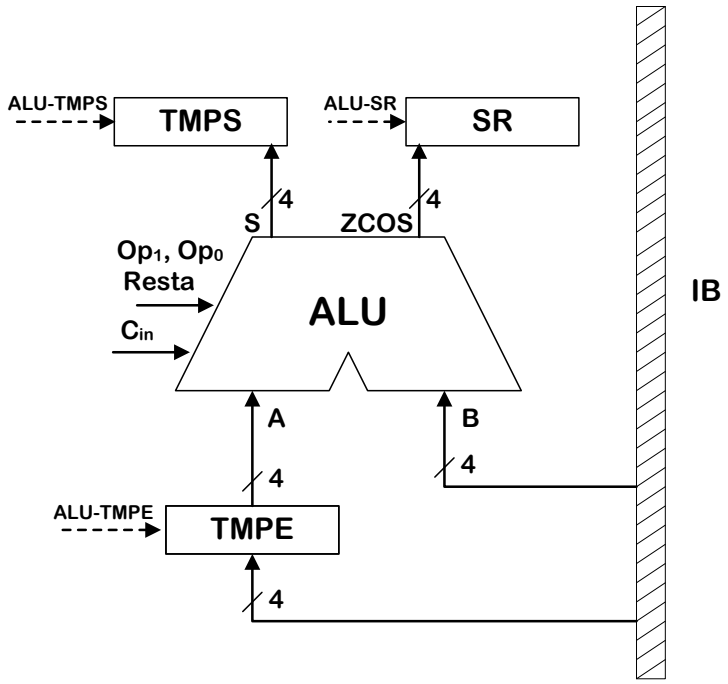

Figura 5.4: Diagrama de bloques de la CPU

❏ Realiza, sobre el papel, la resta de los operandos *A*=5 y *B*=4, tal y como se muestra en la figura 5.5.

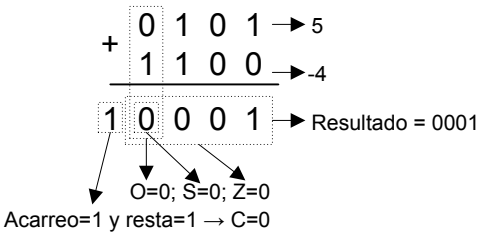

Figura 5.5: Operación $5 - 4$

❏ Simula la resta de los operandos *A*=5 y *B*=4. Para ello debes realizar los siguientes pasos:

1. Pulsa el botón de simular. Te debe salir una tabla con una columna para cada una de las siguientes señales: `resta`, `op1`, `op0`, `ib3`, `ib2`, `ib1`, `ib0`, `IB-TMPE`, `cin`, `ALU-TMPS` y `ALU-SR`. Si falta alguna de estas señales o aparece alguna adicional es que has hecho algo mal y debes revisar el circuito antes de continuar.
2. Sitúa el primer operando en el bus interno, esto es, escribe el valor 0101 en la fila del ciclo cero del generador de entradas. Cada bit debe ir en las columnas correspondientes a `ib3`, `ib2`, `ib1` y `ib0`.
3. Pulsa [↓] en el generador de entradas para que se vuelquen al circuito los valores de la fila correspondiente al ciclo 0.
4. 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`.

5. 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.
6. Sitúa el segundo operando sobre el bus interno, esto es, el valor 0100, así como las entradas `cin`=0, `resta`=1, `op0`=1 y `op1`=1. Estos valores se deben cargar en la fila del ciclo 2.
7. Pulsa [↓] en el generador de entradas para que se vuelquen al circuito 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.
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 como 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`, `reg4.pri` y `reg4.cir`, que contienen el proyecto, el circuito principal y el circuito del componente `reg4`.
- `cpu.pro`, `cpu.pri`, que contienen el proyecto de la CPU.

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].

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?

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