

**Grado en Ingeniería de Tecnologías de Telecomunicación.
Escuela Técnica Superior de Ingeniería Industrial y de Telecomunicación.
Electrónica Digital I.**

Práctica nº 5. Herramientas CAD para el diseño de circuitos digitales combinacionales.

El objetivo de esta práctica es realizar el diseño de circuitos digitales combinacionales utilizando herramientas CAD, en concreto las herramientas del sistema SIS, en especial Espresso (minimizador de expresiones lógicas en dos niveles), con la que se pueden realizar operaciones de minimización lógica. SIS es un paquete para el diseño de circuitos digitales que contiene herramientas para la síntesis de circuitos digitales. SIS ha sido realizado para el sistema operativo UNIX y una versión para el sistema operativo MS-DOS de Windows se encuentra disponible en <http://www1.cs.columbia.edu/~cs6861/sis/sis.html>. Los resultados de la minimización lógica serán implementados con puertas lógicas y simulados mediante Circuit Maker.


En las prácticas con Espresso y SIS se deben plantear y resolver distintos problemas lógicos, los pasos a seguir en cada problema deben ser los siguientes:

- Editar el fichero de descripción del problema propuesto, según el formato que se indica en el manual de las herramientas de espresso y SIS. La descripción debe estar preparada antes de la sesión de prácticas correspondiente.
- Resolver el problema utilizando herramientas CAD y encontrar las ecuaciones lógicas que lo describen. Comparar con los resultados que se obtienen de resolver el problema manualmente.
- Editar un circuito que corresponda a las ecuaciones lógicas y comprobar su funcionamiento con Circuit Maker.

Todo el trabajo de la práctica debe realizarse en el directorio (o carpeta) Pr5 (o similar), que debe crearse en el directorio de trabajo de cada alumno. En la carpeta Pr5 debe utilizarse una subcarpeta para cada apartado cuando fuese necesario y trabajar en ella.

5.1. Introducción a Espresso y a Sis.

5.1.1. Comandos básicos del sistema operativo MS-DOS.

Las prácticas con el sistema SIS se realizan sobre el sistema operativo MS-DOS. Para activar una ventana con ese sistema operativo se debe hacer doble-click sobre el icono de nombre Sis que se encuentra en el escritorio . Se genera una ventana en la que está activo el sistema operativo MS-DOS. El MS-DOS de los ordenadores actuales no dispone de editor de textos. Para editar ficheros se deben usar los editores de Windows, por ejemplo, el *Bloc de Notas*.

Los comandos básicos que se utilizarán en MS-DOS durante esta práctica son:

- **dir**. Permite ver el nombre del directorio actual y su contenido.
- **cd**. Permite cambiar de directorio: **cd ..**, cambia al directorio superior, **cd nombre** cambia al directorio *nombre* inferior, **cd nombre1\nombre2\...\nombrex** cambia al directorio *nombrex* contenido en los directorios intermedios *nombre1*, *nombre2*,
- **mkdir**. Crea un nuevo directorio (o carpeta): **mkdir nombre**, crea el directorio *nombre* dentro del directorio actual. El directorio (o carpeta) puede crearse directamente desde Windows.
- **type**. Muestra el contenido de un fichero en pantalla: **type nombre** muestra el contenido del fichero *nombre*.

5.1.2. Descripción de circuitos en formato *pla*.

El fomato *pla* está permite definir la composición de un circuito programable de tipo PLA (Programmable Logic Array), aunque también puede utilizarse para describir funciones lógicas. Un documento sobre este formato se puede encontrar en <http://www1.cs.columbia.edu/~cs6861/sis/pla.txt>. La descripción de las funciones de conmutación en este formato se realiza mediante cubos o términos productos. En la descripción hay una serie de comandos básicos (nº de entradas, nº de salidas y términos productos), a los que se pueden añadir otros comandos optativos. La descripción básica para un circuito de N entradas y M salidas tiene el siguiente formato (los comandos optativos se muestran entre corchetes):

```
# Una línea que empiece por # es un comentario
.i N (donde N es el número de entradas)
.o M (donde M es el número salidas)
[.p P] (donde P es el número de términos producto)
Relación de términos producto
[.e]
```

Los términos producto se describen como una serie de líneas con un formato:

XXX... YYY.... (01- 1-0, por ejemplo)

Opcionalmente, los términos productos se indican entre los comandos **.p P** (donde P es el número de términos productos) y **.e**

En la descripción de os términos producto **XXX...** (N valores) representa un término producto, siendo cada X el valor que toma cada variable de entrada en el término producto. X puede tomar valor 1 (la entrada correspondiente está en el término producto sin complementar), 0 (la entrada correspondiente está en el término producto complementada) o - (para indicar que el término producto no depende de la variable de entrada, la entrada no está en el término producto). Las N entradas se referencian con nombres v_0 (la entrada más a la izquierda), v_1 , v_2 , ..., v_{N-1} , (la entrada más a la derecha).

Los valores en las salidas son **YYY...** (M valores) donde Y representa lo que produce un término producto en cada salida del circuito. Y puede tomar el valor 1 (el término producto produce 1s en la salida), 0 (el término producto produce 0s en la salida), - (el término producto produce "don't cares" en la salida) y ~ (el término producto no produce nada en la salida, es decir no se tiene en cuenta). Estos valores pueden ser modificados por el comando **.type** que se comenta seguidamente, y que permite activar el valor (el valor se comporta normalmente) o desactivarlo (el valor se comporta como si fuese ~); por defecto los valores 1 y - están activados, y el valor 0 está desactivado. Las M salidas se referencian con nombres $v_{N.0}$ (la salida más a la izquierda, N es el número de entradas), $v_{N.1}$, $v_{N.2}$, ..., $v_{N.M-1}$, (la salida más a la derecha).

Para obtener una mejor legibilidad al utilizar las herramientas espresso y sis con este formato, se pueden poner nombres a las entradas y las salidas del circuito mediante las sentencias:

```
.ilb A B C ... (v0 es A, v1 es B v2 es C, etc)
.ob F1 F2 F3 .... (vN.0 es F1, vN.2 es F2, vN.3 es F3 ....)
```

Ejemplo 1. $F(A, B, C) = \sum(0, 1, 7) + \sum\emptyset(3, 5)$. Tres entradas y una salida.

Cada fila de la descripción corresponde a un minterm de la función (de m0 a m7). En la salida se indica si el minterm toma valores 1, 0 o - (*don't care*)

```
.i 3
.o 1
.ilb A B C
.ob F
.p 8
000 1
001 1
010 0
011 -
100 0
101 -
110 0
111 1
.e
```

Ejemplo 2. Cuatro entradas y tres salidas. Los 1's, 0's y *don't cares* están definidos mediante términos producto.

```
.i 4
.o 3
.ilb A B C D
.ob F1 F2 F3
.p 4
1-00 -10
01-- 1-1
-010 10-
-0-1 001
.e
```

TP1: 1-00 => \overline{ACD} , (minterms m8 y m12) produce *don't cares* en F1 y 1s en F2.

TP2: 01-- => \overline{AB} , (minterms m4, m5, m6 y m7) produce 1s en F1 y F3, *don't cares* en F2.

TP3: -010 => \overline{BCD} , (minterms m2 y m10) produce 1s en F1 y *don't cares* en F3.

TP3: -0-1 => \overline{BD} , (minterms m1, m3, m9 y m11) produce 1s en F3.

Entonces: $F1(A, B, C, D) = \sum(2, 4, 5, 6, 7, 10) + \sum\emptyset(8,12)$

$F2(A, B, C, D) = \sum(8, 12) + \sum\emptyset(4, 5, 6, 7)$

$F3(A, B, C, D) = \sum(1, 3, 4, 5, 6, 7, 9, 11) + \sum\emptyset(2, 10)$

El formato por defecto para mostrar el resultado de una minimización es similar al formato de entrada, (nº de entradas, nº de salidas y términos productos en formato XXX... YYY...), donde los términos producto corresponden al resultado de la minimización y se muestran entre los comandos **.p** y **.e**. Los XXX... de los términos productos se comportan como en el formato de descripción, pero los términos YYY... solo toman los valores 1 (el término producto pertenece a la salida correspondiente del circuito) y 0 (el término producto no pertenece a la salida correspondiente del circuito). El orden y el nombre de referencia de las entradas (v0, v1, ...) y de las salidas (vN.0, vN.1, ...) en los términos producto son idénticos a los utilizados en el fichero de entrada.

Resultado de la minimización del ejemplo1:

```
.i 3
.o 1
.ilb A B C
.ob F
.p 2
--1 1
00- 1
.e
```

$F(A, B, C) = C + \overline{AB}$

Resultado de la minimización del ejemplo2:

```
.i 4
.o 3
.ilb A B C D
.ob F1 F2 F3
.p 4
1-00 010
-010 100
-0-1 001
01-- 101
.e
```

$F1(A, B, C, D) = \overline{BCD} + \overline{AB}$

$F2(A, B, C, D) = \overline{ACD}$

$F3(A, B, C, D) = \overline{BD} + \overline{AB}$

Existen otros comandos que permiten definir características de las funciones de conmutación o de su descripción. Estos comandos deben indicarse antes de la definición de los términos producto de las funciones. A continuación se citan dos comandos frecuentemente usados: **.type** y **.phase**.

El comando **.type** se utiliza para definir los tipos de datos que producen los términos productos. Para definir un circuito digital los términos productos se incluyen en tres tipos de conjuntos: el ON (los 1s), el OFF (los 0s) y el DC (“don't cares”). Dentro del fichero de entrada de Espresso puede determinarse los conjuntos que se indican en la descripción mediante la orden:

.type [f | r | fd | fr | rd | fdr] (entre corchetes se muestran las opciones posibles *f*, *r*, *fr*, etc)

donde las opciones indican los conjuntos que están activos al leer la descripción (*f* el ON, *d* el DC y *r* el OFF). Por defecto, está activa la opción *fd*, de forma que al leer el fichero de entrada solo se reconocen los términos productos que producen 1 (ON) y - (DC) en las salidas, y no se toman en cuenta los que producen 0 (OFF). La orden **.type fdr** permite reconocer todos los conjuntos 1 (ON), 0 (OFF) y - (DC), mientras que, por ejemplo, la opción **.type r** permite reconocer solo a los términos que generan 0 (OFF). Cuando el valor ~ aparece en un término producto indica que el término producto en la salida correspondiente no es ni ON ni OFF ni DC, cuando un valor no aparece en el comando **.type**, todos los valores de ese tipo se comportan como ~.

Al aplicarse Espresso, el programa lee de la descripción los conjuntos ON, OFF y DC que estén activos, donde al menos uno de los conjuntos ON u OFF deben estar activos. El programa genera un conjunto no activo (DC solo si ON y OFF están activos; ON u OFF, el no activo, en los demás casos) en función de los conjuntos activos leídos.

El comando **.phase** permite definir si obtiene en la minimización la función lógica de una de las salidas o la función lógica complementada, mediante la orden:

.phase PPP...

donde cada P toma valor 1 si se desea que la señal de salida tome la función lógica (se minimiza el conjunto ON), y 0 si se desea que la señal de salida tome la función lógica complementada (se minimiza el conjunto OFF). Se deben incluir M valores P (uno por cada salida). Cuando no se usa este comando se presupone que ninguna salida está complementada.

5.1.3. Descripción básica del minimizador en dos niveles Espresso.

Un documento de ayuda más completo para la ejecución de espresso puede encontrarse en <http://www1.cs.columbia.edu/~cs6861/sis/espresso.txt>. La ejecución básica del programa se realiza desde el sistema operativo MS-DOS mediante la orden básica: **espresso mi_fichero**, donde *mi_fichero* contiene la descripción de entrada. Esta ejecución realiza una minimización

conjunta de todas las funciones de conmutación (o salidas) definidas. Los resultados de la ejecución de Espresso aparecen en el terminal gráfico, para almacenarlos en un fichero, se debe utilizar la opción de redireccionamiento del sistema operativo UNIX: **espresso *mi_fichero* > *fich_salida***. Por defecto, este comando genera una minimización de tipo heurístico con un método iterativo de aproximación a la función óptima, pero que no garantiza que en todos los casos se obtenga la función mínima. El criterio de coste que se utiliza es el de conseguir el menor número de términos producto en la solución final, aunque se incluye un procedimiento final para reducir el número de literales.

Espresso tiene algunas opciones de ejecución. Para obtener una ayuda en pantalla de estas opciones de Espresso hay que introducir la orden **espresso -h**. Las opciones de ejecución más interesantes en cuanto a la ejecución tienen un formato del tipo:

espresso -*Dopcion* -*oformato* *mi_fichero*

donde *opcion* representa las diferentes opciones de ejecución de Espresso y *formato* representa el formato esperado en la salida.

Algunas de las opciones de ejecución de Espresso que se utilizan son:

- **Dcheck**: comprueba si las funciones de conmutación están mal definidas. No realiza la minimización. Conviene situar el comando **.type** al valor **fdr** con esta opción.
- **Dexact**: realiza una minimización exacta, que garantiza encontrar la solución óptima. En problemas grandes el tiempo de cómputo requerido es alto.
- **Dopo**: realiza una minimización conjunta en la que se permite complementar o no las señales de salida para mejorar la minimización.
- **Dso**: realiza una minimización para cada salida por separado.
- **Dso_both**: realiza una minimización para cada salida por separado, obteniendo la función o la función complementada para cada salida, según cuál sea la mínima.

El formato de salida de Espresso tiene varias opciones posibles. Al igual que en la definición de **.type** se pueden indicar los valores **-o[f | d | r | fd | fr | dr | fdr]**, que indica los conjuntos que se van a mostrar: f el ON, r el OFF y d el DC. Por defecto, se tiene la opción **-of** que muestra el conjunto ON resultante de la minimización. Otra opción que se puede utilizar es **-oeqntott**, que muestra la salida en un formato algebraico, en el que la operación AND se representa por **&**, OR por **|**, y NOT por **!**.

5.1.4. Breve descripción del sistema SIS.

SIS es un paquete para el diseño de circuitos digitales que contiene herramientas para la síntesis de circuitos digitales. SIS ha sido realizado para el sistema operativo UNIX y una versión se encuentra disponible en <http://www1.cs.columbia.edu/~cs6861/sis/sis.html> para diversos sistemas operativos, incluido Windows. Se puede encontrar un documento de ayuda en: <http://www1.cs.columbia.edu/~cs6861/sis/sis.1.html>.

Entre las herramientas incluidas en SIS se encuentran manipuladores de expresiones lógicas que se utilizan en el diseño de circuitos combinacionales, tales como minimizadores de expresiones lógicas en dos niveles (como *Espresso* por ejemplo), o como operaciones para síntesis multinivel como factorización (*factor*), descomposición (*decomp*), colapsado (*collapse*), sustitución (*resub*), extractor de cubos comunes y de múltiples cubos comunes (*gcx* y *gkx*), obtención de divisores comunes (*fx*), etc. También existen opciones típicas del diseño de circuitos secuenciales, como son la minimización de estados (*state_minimize*, *stamina* por defecto o *sred*) y la asignación secundaria de estados (*state_assign*, *nova* por defecto o *jedi*, o *one_hot*). SIS también realiza la tarea de mapear las expresiones lógicas, dado un catálogo concreto de dispositivos (puertas lógicas y flip-flops), modificándolas en expresiones directamente implementables por los dispositivos del catálogo (*read_library* para seleccionar el catálogo y *map* para realizar la implementación). Además, SIS contiene utilidades que permiten simular el comportamiento del circuito (*simulate*), comprobar si dos descripciones son equivalentes en circuitos combinacionales (*verify*) y secuenciales (*verify_fsm*), hacer una estimación de la potencia disipada por el circuito (*power_estimate*) o encontrar una secuencia de prueba para fallos en el circuito (*atpg*).


SIS permite varios formatos de descripción de expresiones lógicas y circuitos. El formato propio de SIS es el formato blif (Berkeley Logic Interchange Format), que es relativamente complejo y permite hacer relación a alguna característica eléctrica (cargabilidad) o temporal. También se permiten otros formatos como el pla indicado en el apartado 5.1.2, típico de Espresso, para definir de forma sencilla expresiones lógicas combinacionales en dos niveles, o el formato kiss, que permite definir máquinas de estados finitos. Dentro de SIS, las expresiones lógicas se muestran en formas SOP o factorizada, donde los nudos utilizados se muestran con el nombre fijado en la descripción de entrada, o como IN_X para las entradas sin nombre (X representa el índice de la entrada), {OUT_X} para las salidas sin nombre y LatchOut_vX para las variables de estado sin nombre, y [XXX] para los nudos intermedios (XXX es el índice del nudo); se utiliza el carácter ‘ para indicar la complementación de un literal y el carácter + para la operación OR.

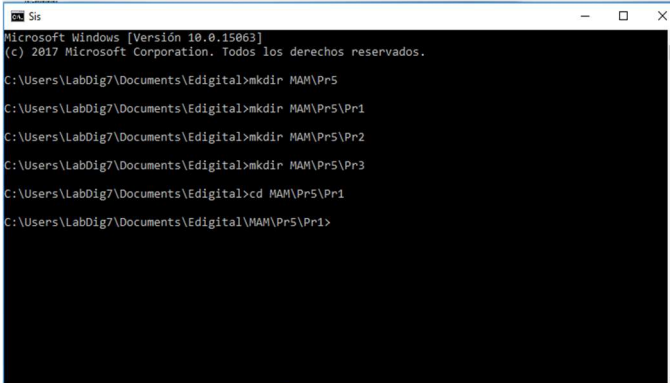
5.2. Minimización en dos niveles de una función lógica mediante espresso.

En este apartado se va a trabajar con una función lógica y se va a obtener sobre ella una forma SOP mínima en dos niveles mediante el minimizador lógico espresso, y una forma factorizada mediante las opciones de SIS. La función a utilizar en este apartado es:

$$F(A, B, C, D, E) = \sum(2,3,4,5,7,8,11,13,18,20,22,27,28,31) + \sum\phi(1,6,10,12,15,16,19,21,25,30)$$

- Trabajo previo. Encontrar una forma SOP mínima con ayuda de los mapas de Karnaugh de la siguiente función lógica. Encontrar una buena expresión factorizada a partir de la forma SOP encontrada, utilizando el algoritmo recursivo de literal de mayor aparición. Preparar la descripción de tipo pla para esta función lógica. Una buena forma de hacerlo es incluir en la descripción una fila por cada minterm que genera 1s y *don't cares* en la salida.

- Abrir la ventana MS-DOS pulsando sobre el icono Sis . Por defecto el directorio de trabajo es Edigital, Crear en el directorio de trabajo de cada alumno la carpeta Pr5 y dentro de ella las subcarpetas Pr1, Pr2 y Pr3 para cada subapartado de la práctica. También puede hacerse desde MS-DOS usando los comandos **mkdir XXX\Pr5** (donde XXX es el nombre de la carpeta del alumno) y sucesivamente **mkdir XXX\Pr5\Pr1**, **mkdir XXX\Pr5\Pr2** y **mkdir XXX\Pr5\Pr3**. Moverse al directorio de trabajo **Pr1** de este apartado de la práctica. Para ello indicar el comando: **cd XXX\Pr5\Pr1** (donde XXX es el nombre la carpeta del alumno),



```

Microsoft Windows [Versión 10.0.15063]
(c) 2017 Microsoft Corporation. Todos los derechos reservados.

C:\Users\LabD1g7\Documents\Edigital>mkdir MAM\Pr5
C:\Users\LabD1g7\Documents\Edigital>mkdir MAM\Pr5\Pr1
C:\Users\LabD1g7\Documents\Edigital>mkdir MAM\Pr5\Pr2
C:\Users\LabD1g7\Documents\Edigital>mkdir MAM\Pr5\Pr3
C:\Users\LabD1g7\Documents\Edigital>cd MAM\Pr5\Pr1
C:\Users\LabD1g7\Documents\Edigital\MAM\Pr5\Pr1>

```

- Editar el fichero de descripción en formato espresso de nombre *entrada.txt* desde el *Bloc de Notas*. Para arrancar el editor pulsar el icono de Windows, seleccionar *Accesorios de Windows* y pulsar ahora sobre *Bloc de Notas*. Introducir en el fichero la descripción del circuito en formato pla.

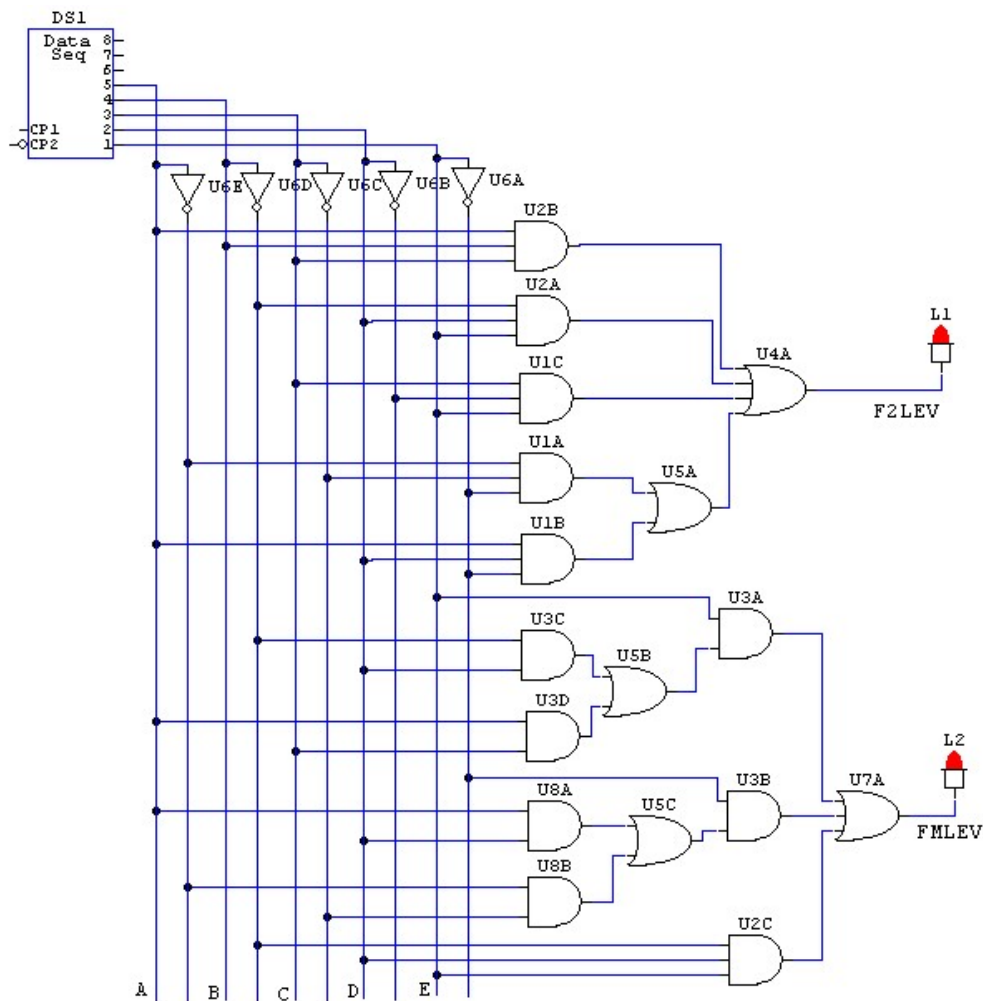
- Ejecutar la orden **espresso –Dexact entrada.txt > resul1** para realizar la minimización y almacenar los resultados en el fichero *resul1*. Comparar los resultados mediante la orden **type resul1** con los que se obtuvieron de la minimización por Mapa de Karnaugh. Las expresiones algebraicas pueden obtenerse mediante la orden **espresso –Dexact –oeqntott entrada.txt > resul2** (! NOT, & AND, | OR), los resultados en ecuaciones algebraicas se pueden visualizar mediante la orden **type resul2**. Comparar el tamaño de la expresión generada por espresso con la generada mediante mapas de Karnaugh.

- Encontrar mediante SIS una expresión factorizada de la función. Introducir el comando **sis**, y sobre el prompt de la herramienta (*sis>*) que aparece en pantalla, introducir la siguiente secuencia de órdenes, que realizan la minimización en dos niveles seguida de una factorización:

read_pla entrada.txt (lee el fichero *entrada.txt* en formato pla)
full_simplify (realiza una minimización en dos niveles de tipo espresso)
print_factor (muestra una factorización de la expresión en dos niveles)
quit (sale de Sis y vuelve al sistema operativo)

Entre cada orden se puede introducir la orden **print** para observar cómo se modifican las expresiones lógicas. Al introducir la última orden aparece en pantalla la expresión factorizada.

Comparar el tamaño de esta expresión (número de literales) con el tamaño de la expresión en dos niveles y con el tamaño de la expresión factorizada obtenida manualmente.



• Diseñar un circuito que implemente las expresiones lógicas SOP (obtenida mediante espreso) y factorizada (obtenida mediante Sis), utilizando puertas AND, OR y NOT, e implementarlo en Circuit Maker en el fichero **Sim1.ckt**. Si no hubiese puertas con el número de entradas suficiente (AND/OR de 5 entradas o más) se pueden construir mediante árboles de AND o de OR, o mediante puertas NOR, NAND de 8 entradas seguidas por un inversor, con las entradas no utilizadas conectadas a valor no controlante (0 o GND en puertas OR, 1 o Vcc en puertas AND). Realizar las conexiones de forma que se pueda rehacer el circuito fácilmente en caso de error, por ejemplo, situar etiquetas, realizar líneas verticales paralelas para las entradas complementadas y sin complementar y conectar las entradas de las puertas AND mediante líneas horizontales. El circuito de la figura muestra una forma aproximada de conexión, aunque no se corresponde con el circuito a diseñar.

Simular en modo digital, verificando que los dos circuitos realizan la tabla de verdad del problema. Utilizar un Data Sequencer (*hotkey G*) para aplicar las señales de entrada y Logic Displays (*hotkey 9*) para observar el valor en las salidas. Programar el Data Sequencer de forma que se puedan aplicar desde las cinco salidas más bajas del generador los 32 valores de la tabla

de verdad (en hexadecimal de 00 a 1F, se puede hacer inmediatamente desde la pestaña *Pattern* del *Data Sequencer* escogiendo *Count Up*) a las entradas del circuito de la dirección 1 (*Start Address*) a la 32 (*Stop Address*); tener en cuenta que el campo *Tick Increment* debe estar a un valor suficiente (10 ticks) para permitir que el valor de las entradas llegue a la salida antes de aplicar un valor nuevo. Utilizar la simulación por pasos (*Step Size* a valor 10) para sincronizar las medidas con el *Tick Increment* del generador de secuencias.

5.3. Minimización de un problema lógico de múltiples salidas.

El objetivo de este apartado es minimizar las expresiones lógicas que definen un circuito digital de varias salidas

$$F1(A, B, C, D) = \sum(0, 4, 6, 9, 10, 11) + \sum\emptyset(2, 13, 14)$$

$$F2(A, B, C, D) = \sum(2, 3, 5, 7, 9, 10, 11, 14) + \sum\emptyset(1, 6)$$

$$F3(A, B, C, D) = \sum(0, 1, 3, 6, 7, 14) + \sum\emptyset(2, 4, 5)$$

- Trabajo previo. Realizar una minimización en dos niveles por separado de cada salida por separado con ayuda de los mapas de Karnaugh, e intentar encontrar una solución mínima conjunta. Preparar una descripción de tipo *pla* para este problema de 4 entradas y 3 salidas. Se puede hacer fácilmente editando un término producto por cada minterm, e indicando en cada salida el valor: 1, 0 o *don't care*.

- Moverse al directorio de trabajo **Pr2** (o el nombre que se elija para la carpeta de este apartado) en la ventana MS-DOS, suponiendo que éste ya ha sido creado desde el sistema operativo Windows. Si no, usar desde el directorio Pr1 del apartado anterior la orden **mkdir ..\Pr2** y moverse a él mediante la orden **cd ..\Pr2**.

- Editar con el *Bloc de Notas* el fichero de descripción de tipo *pla* de nombre *entrada1.txt* a partir de la tabla de verdad del problema (introduciendo también las condiciones don't care).

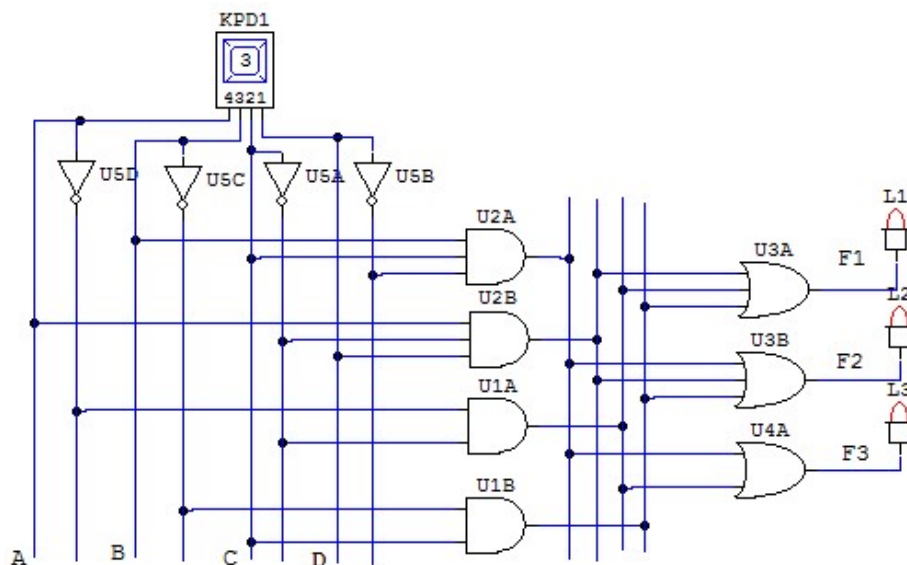
- Utilizar la orden **espresso -Dso entrada1.txt > resul1** para realizar una minimización por salidas individuales. Tener en cuenta que en los resultados obtenidos por Espresso, un término producto puede aparecer varias veces si aparece en distintas salidas (aparece una vez por cada salida). Comparar el resultado obtenido con el realizado con ayuda de los mapas de Karnaugh.

- Realizar una minimización conjunta de las funciones lógicas utilizando las órdenes:

a) **espresso entrada1.txt > resul2**

b) **espresso -Dexact entrada1.txt > resul3**

ya que estos dos comandos utilizan algoritmos distintos. Comparar si las soluciones son las mismas, utilizando las que generen un circuito más pequeño, y comparar el tamaño de este circuito con el resultante de la minimización individual de cada salida.



• Utilizar las ecuaciones lógicas generadas en el punto anterior para realizar la implementación del circuito, utilizando únicamente puertas lógicas AND, OR y NOT. Simular la implementación en modo digital mediante Circuit Maker en el fichero **Sim3.ckt**, verificando la tabla de verdad. Utilizar la llave hexadecimal (Hexadecimal Key, *hotkey H*) para aplicar las señales de entrada y tres displays lógicos (Logic Display, *hotkey 9*) para comprobar el resultado. Realizar las conexiones de forma que se pueda rehacer el circuito fácilmente en caso de error: situar etiquetas, realizar líneas verticales paralelas para las entradas complementadas y sin complementar y para las salidas de cada término producto, y conectar las entradas de las puertas AND mediante líneas horizontales. Realizar la simulación verificando el funcionamiento del circuito para las combinaciones de entrada válidas. El circuito de la figura no corresponde al circuito final de la práctica.

5.4. Obtención de una forma multinivel.

El objetivo de este apartado es generar un circuito digital multinivel para un circuito sumador completo de 1 bit. Este circuito tiene 3 entradas A, B y Ci y dos salidas Co y S, de forma que realiza la suma aritmética de A, B y C y genera el resultado en un número binario de 2 bits formado por $(Co S)_2$. Este circuito se va a implementar de forma multinivel, utilizando las puertas disponibles de un catálogo de dispositivos. Existen varios catálogos en la versión de Sis disponible en el laboratorio. En esta práctica se utilizará en concreto el catálogo *msu.gen*. En este catálogo se muestran los dispositivos disponibles referenciados por un número, la función lógica que realizan y alguna de sus características físicas como tamaño, tiempo de retraso del cambio de las entradas a las salidas y la influencia de las conexiones en dicho tiempo.

GATE	"1310:physical"	16	O=!1A;
GATE	"1120:physical"	24	O=!(1A+1B);
GATE	"1130:physical"	32	O=!(1A+1B+1C);
GATE	"1140:physical"	40	O=!(1A+1B+1C+1D);
GATE	"1220:physical"	24	O=!(1A*1B);
GATE	"1230:physical"	32	O=!(1A*1B*1C);
GATE	"1240:physical"	40	O=!(1A*1B*1C*1D);
GATE	"1660:physical"	32	O2=1A*1B;
GATE	"1670:physical"	40	O2=1A*1B*1C;
GATE	"1680:physical"	48	O2=1A*1B*1C*1D;
GATE	"1760:physical"	32	O1=1A+1B;
GATE	"1770:physical"	40	O1=1A+1B+1C;
GATE	"1740:physical"	48	O=1A+1B+1C+1D;
GATE	"1870:physical"	40	O=!(1A*1B+2C*2D);
GATE	"1880:physical"	32	O=!(1A+2B*2C);
GATE	"1860:physical"	40	O=!(1A+1B)*(2C+2D);
GATE	"1890:physical"	32	O=!(1A*(2B+2C));
GATE	"1970:physical"	56	O=1A*1B+2C*2D;
GATE	"1810:physical"	72	O=1A*1B+2C*2D+3E*3F;
GATE	"1910:physical"	96	O=1A*1B+2C*2D+3E*3F+4G*4H;
GATE	"1930:physical"	64	O=1A*1B*1C+2D*2E*2F;
GATE	"2310:physical"	40	O=1A*!1B+!1A*1B;
GATE	"2310:physical"	40	O=!(1A*1B+!1A*!1B);
GATE	"2350:physical"	48	O=1A*!1B+!1A*!1B;
GATE	"2350:physical"	48	O=!(1A*!1B+!1A*!1B);
GATE	"1610:physical"	32	O=!1A*2B;
GATE	"1620:physical"	32	O=1A+!2B;
GATE	"1350:physical"	48	O=1D1*3SEL+2D2*!3SEL;
GATE	"1430:physical"	8	O=CONST1;
GATE	"1440:physical"	8	O=CONST0;

En la figura se muestra parte del catálogo *msu.gen* (faltan las referencias a los tiempos de retraso), en el que se muestran los dispositivos disponibles, sus números de referencia, sus tamaños y las funciones lógicas que realizan. Los símbolos !, * y + son respectivamente las operaciones lógicas NOT, AND y OR. En este catálogo están definidas las puertas lógicas básicas: NOT (1310), NOR (1120, 1130, 1140), NAND (1220, 1230, 1240), AND (1660, 1670, 1680), OR (1760, 1770, 1740), XOR (2310), XNOR (2350), además de otras puertas complejas que incluyen en el mismo dispositivo puertas AND-OR-NOT (1870, 1880, 1860, 1890, 1970, 1810, 1910, 1930, ...).

- Trabajo previo. Preparar una descripción de tipo *pla* para este problema de 3 entradas y 2 salidas, correspondiente a una sumador completo de 1 bit. Se puede hacer fácilmente editando un término producto para cada combinación en las entradas, e indicando en cada salida el valor: 1, 0.
- Moverse al directorio de trabajo **Pr3** (o el nombre que se elija para la carpeta de este apartado) en la ventana MS-DOS, suponiendo que éste ya ha sido creado desde el sistema operativo Windows. Si no, usar desde el directorio Pr2 del apartado anterior la orden **mkdir ..\Pr3** y moverse a él mediante la orden **cd ..\Pr3**.
- Editar con el *Bloc de Notas* el fichero de descripción de tipo *pla* de nombre *full_adder.txt* a partir de la tabla de verdad del sumador completo de 1 bit.
- Introducir el comando **sis**, y sobre el prompt de la herramienta (*sis>*) que aparece en pantalla, introducir la siguiente secuencia de órdenes, anotando la evolución de las funciones lógicas en

el circuito. Al ejecutar sis puede que cambie la configuración del teclado de forma que el carácter '-' está en la tecla de comilla simple, '_' está en la tecla '?', y '*' en la tecla '('.

read_pla full_adder.txt (lee el fichero *full_adder.txt* en formato pla)
print (muestra en pantalla las ecuaciones en dos niveles de cada nudo del circuito)
print_factor (muestra las formas factorizadas de cada nudo del circuito)
simplify -m nocomp -d (realiza una minimización en dos niveles del tipo espresso)
print (muestra en pantalla las ecuaciones en dos niveles de cada nudo del circuito)
print_factor (muestra las formas factorizadas de cada nudo del circuito)
decomp -g * (descompone las expresiones de las salidas generando nudos internos)
print_factor (los nudos internos aparecen descritos como números entre corchetes)
read_library c:\sis\sis_lib\msu.gen (lee el catálogo msu.gen de puertas lógicas disponibles)
map (genera los nudos internos con puertas lógicas del catálogo)
print_factor (los nudos internos aparecen descritos como números entre corchetes)
print_gate (muestra el número de referencia de la puerta con la que se genera cada nudo interno según el catálogo msu.gen)
quit (sale de Sis y vuelve al sistema operativo)

- A partir de las ecuaciones del último comando *print_factor* y de las puertas del último comando *print_gate*, construir el circuito en Circuit Maker y comprobar que funciona como un sumador completo. Para hacer bien la construcción hay que darse cuenta que la puerta 2310 es la definición de una puerta XOR, y que la puerta 1890 además de realizar la función $O = \neg(1A * (2B + 2C))$ por las leyes de DeMorgan realiza equivalentemente la función $O = \neg 1A + \neg 2B * \neg 2C$. La puerta 1890 se debe implementar con una OR y una NAND.