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.

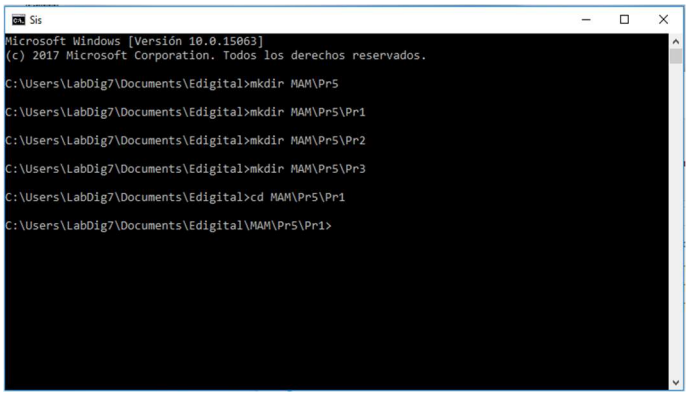
5.2. Minimización en dos niveles de una función lógica mediante espresso.

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),



• 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:

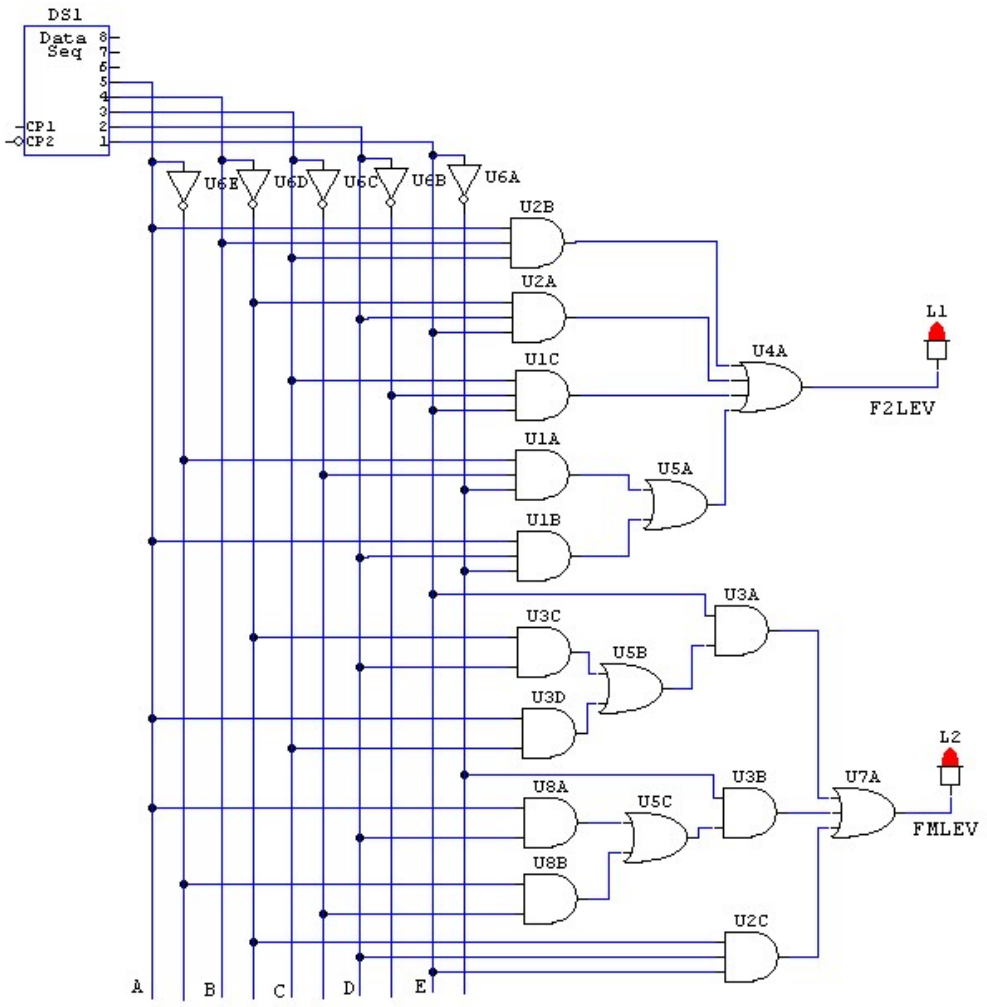
read_pla entrada.txt (lee el fichero *entrada.txt* en formato pla)
full_simplify (realiza una minimización en dos niveles de tipo espresso)
print_factor (muestra una factorización de la expresión en dos niveles)
quit (sale de Sis y vuelve al sistema operativo)

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 espresso) 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) = ∑(0, 4, 6, 9, 10, 11) + ∑ø(2, 13, 14)
F2(A, B, C, D) = ∑(2, 3, 5, 7, 9, 10, 11, 14) + ∑ø(1, 6)
F3(A, B, C, D) = ∑(0, 1, 3, 6, 7, 14) + ∑ø(2, 4, 5)

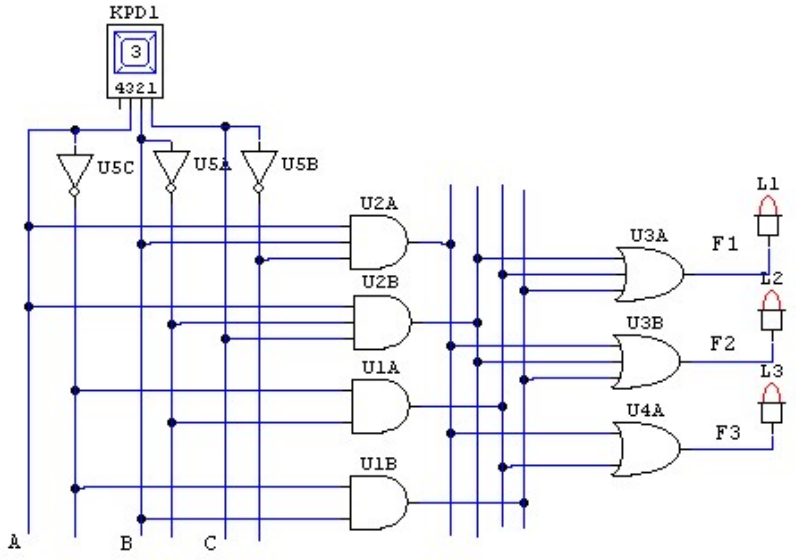
• Trabajo previo. Realizar una minimización en dos niveles por separado de cada salida por separado con ayuda de los mapas de Karnaugh, e intentar encontrar una solución mínima conjunta. Preparar una descripción de tipo *pla* para este problema de 4 entradas y 3 salidas. Se puede hacer fácilmente editando 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 en las entrada 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.

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 las 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=!(1A*(2B+2C)) por las leyes de DeMorgan realiza equivalentemente la función O=!1A+ !2B*!2C. La puerta 1890 se debe implementar con una OR y una NAND.