

MAPEADO TECNOLÓGICO, SÍNTESIS LÓGICA Y HERRAMIENTAS DE CAD MICROELECTRÓNICO

J.L. Ramírez, J. Pallares, N. Cañellas y L.F. Marsal
Universitat Rovira i Virgili
Departament d'Enginyeria Electrònica
Escola Tècnica Superior d'Enginyeria
Carretera de Salou, s/n 43006 Tarragona
Tel.: (977) 55 96 29 Fax.: (977) 55 97 10
e-mail: jramirez@etse.urv.es

RESUMEN.- Es importante que, para conseguir la total comprensión de los conceptos teóricos, el alumno pueda realizar ejercicios prácticos. Estos ejercicios serán más útiles si, de forma paralela, sirven para aclarar conceptos relacionados, acercar al alumno a un entorno industrial real e incentivar su interés por la asignatura.

En la presente comunicación se describe una aplicación sencilla con una herramienta de CAD microelectrónico. En ella, el alumno tiene la posibilidad de ver como una misma función lógica puede presentar diversas realizaciones prácticas. Comprobará también que, las distintas formas de mapear esta función, presentan costes distintos en área y tiempo de retardo. Simultáneamente se iniciará en el uso de un gran entorno de diseño y podrá comprobar la utilidad de los nuevos lenguajes de descripción en alto nivel.

1.- INTRODUCCIÓN

En la actualidad los sistemas digitales han alcanzado un nivel de complejidad muy elevado. Esto hace necesario el uso de herramientas 'software' para llevar a cabo su diseño. Queda fuera de toda duda, que nuestros estudiantes deben tener, durante su formación, la posibilidad de tomar contacto con las herramientas que actualmente existen en el mercado.

Este mismo nivel de complejidad, y su constante crecimiento, es el que ha impulsado la aparición de lenguajes de alto nivel para la descripción de circuitos. Parece que, día a día, estos lenguajes van ganando terreno a la descripción convencional mediante esquemas circuitales.

Teniendo presentes los puntos anteriores, creemos que es una buena estrategia usar una herramienta de síntesis automática -síntesis lógica, en realidad- para asentar el concepto de mapeado tecnológico.

Proponemos la realización de un ejercicio práctico que, de una manera sencilla y sin suponer una gran carga para el estudiante, favorezca la total comprensión de como y porque una función lógica puede implementarse de distintas formas según las condiciones del problema. Al mismo tiempo nos servirá para introducir al alumno en el uso de herramientas de CAD microelectrónico y de síntesis lógica.

El ejercicio se llevará a cabo en una sesión de, aproximadamente, dos horas y utilizando el sintetizador (vhdlSyn 4.3) y optimizador (Sinergy 1.3) integrados en el entorno Design Framework II, de la firma Cadence. El mapeado se realizará sobre celdas estándar de las librerías de Mietec, tecnología CMOS de 2.4 micras.

Para enriquecer el ejercicio, y redundando en los conceptos anteriores, tendrán la posibilidad de comparar las soluciones obtenidas con distintas tecnologías. Para ello pueden repetir algún apartado usando tecnología CMOS de 0.7 μm o 1 μm de las librerías de ES2.

2.- OBJETIVOS

El objetivo, genérico, de algunos ejercicios prácticos es apoyar las exposiciones realizadas en las 'clases de teoría'. Facilitan la comprensión de algunos conceptos y permiten asegurar su correcta sedimentación.

En otros casos, lo que se pretende es poner al alumno en contacto con la realidad física que deberá manipular al llevar a cabo el desarrollo de un proyecto.

Nuestro ejercicio persigue ambos objetivos simultáneamente:

Convencerle de que la solución óptima a un problema no es universal, sino que depende de las condiciones del problema. Mostrar como una misma función se puede mapear de formas distintas sobre una misma librería tecnológica, o incluso sobre tecnologías distintas.

Habituarle al uso de herramientas 'software' de diseño microelectrónico de carácter industrial.

Despertar su curiosidad respecto a las nuevas herramientas de síntesis lógica.

Esperamos, además, conseguir estos objetivos de una forma que resulte amena y estimulante para el alumnado.

3.- ELECCIÓN DEL SOFTWARE

Existe en el mercado una amplia diversidad de herramientas de síntesis lógica. Pero entre ellas el Design Framework II (DFII) se ajusta especialmente a nuestros objetivos:

Podemos trabajar con librerías de celdas estándar de varios fabricantes. Esto nos permitirá comparar los frutos de implementar una función con unas u otras.

Tiene la herramienta, de síntesis y optimización, integrada en un entorno de diseño industrial.

El único factor adverso que presenta la elección es su coste. Afortunadamente este punto viene amortiguado, en parte, por dos hechos:

Primero. Las universidades tienen la posibilidad de acceder a estas herramientas, de un modo ventajoso, mediante la acción, de la Unión Europea, Europractice.

Segundo. Su versatilidad lo hace idóneo para la realización de otras prácticas y para el desarrollo de algunas líneas de investigación. Resulta rentable, ya que las grandes posibilidades que ofrece - podemos realizar en él el desarrollo completo de un circuito integrado, desde su especificación, hasta la generación de sus vectores de test, pasando por el diseño de su layout y la posterior simulación eléctrica - lo hacen adecuado para otras muchas prácticas relacionadas con el diseño microelectrónico - en las que el alumno se enfrentará a un entorno ya conocido -

4.- DESARROLLO DE LA PRÁCTICA

4.1.- Preparación de la sesión

Por parte del alumno el trabajo previo a la práctica se limitará a:

Estudio de la documentación y relación de la práctica con los conocimientos previos.

Diseño esquemático de dos funciones lógicas sencillas. En este ejemplo:

$$Z = X0 \text{ xor } X1 \text{ xor } X2 \text{ xor } X3$$
$$Z = (X0 * X1 * X1) + (X3 * X4 * X5)$$

Modificación de una descripción VHDL , patrón facilitado junto con la documentación.

Por nuestra parte, tendremos que:

Realizar un fichero con las definiciones VHDL necesarias para que el alumno pueda completarlo, sin necesidad de que conozca todos los rudimentos de este lenguaje.

Crear las cuentas de trabajo necesarias e inicializar el entorno de la forma más adecuada.

Preparar una documentación suficientemente detallada como para que la práctica se pueda realizar en el periodo previsto de dos horas. Los detalles de uso del programa no deben desviar la atención del alumno del concepto principal, ni producir un efecto adverso de rechazo al uso de la herramienta.

4.2.- Sesión de trabajo

En este apartado presentamos, de forma detallada, los pasos a seguir en la realización de la práctica. Desde este punto, y para facilitar el seguimiento de dichos pasos, tomaremos el siguiente convenio: Las letras en negritas indicaran las opciones que aparecen de forma textual en los menús o ventanas de la aplicación. Los dos puntos ':' indicaran un anidamiento en menú desplegable, mientras que el mayor que '>' indicará un campo que debemos definir.

También se presentan algunas figuras de los pasos intermedios.

En primer lugar, habiendo ya iniciado una sesión de trabajo en el entorno UNIX, el alumno modificará el fichero patrón VHDL que se le facilita (Figura 1):

-- VHDL patron inicial --

```
library IEEE
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity XXX is
  Port ( X0 : In  std_logic;
        Z : Out std_logic );
end XXX;

architecture BEHAVIORAL of XXX is
  begin
    Z <= X0;
  end BEHAVIORAL;
```

-- VHDL modificado alumno --

```
library IEEE
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity XOR4 is
  Port ( X0 : In  std_logic;
        X1 : In  std_logic;
        X2 : In  std_logic;
        X3 : In  std_logic;
        Z : Out std_logic );
end XOR4;

architecture BEHAVIORAL of XOR4 is
  begin
    Z <= X0 xor X1 xor X2 xor X3;
  end BEHAVIORAL;
```

-- VHDL modificado alumno --

```
library IEEE
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity F1 is
  Port ( X0 : In  std_logic;
        X1 : In  std_logic;
        X2 : In  std_logic;
        X3 : In  std_logic;
        X4 : In  std_logic;
        X5 : In  std_logic;
        Z : Out std_logic );
end F1;

architecture BEHAVIORAL of F1 is
  begin
    Z <= (X0 and X1 and X2) or (X3
and X4 and X5);
  end BEHAVIORAL;
```

Figura 1.- Ficheros VHDL, patrón y modificados por el alumno.

Iniciaremos la sesión de Cadence. En nuestro caso tenemos definido un alias para las herramientas de síntesis. Ejecutamos el programa en 'back-ground' de forma que podremos seguir usando la ventana de comandos:

```
sun> vhdlsyn &
```

En la primera fase no tomamos decisiones que afecten a la realización del diseño. Estructuramos los datos, los guardamos en los directorios adecuados y comprobamos su corrección:

Open: VHDL Shell

Hemos iniciado la sesión. Ahora definimos donde tenemos los ficheros VHDL y donde guardaremos los ficheros intermedios:

Libraries: Logicals: Define

Logical Name > Nombre de la librería a crear.

Value > Directorio donde almacenarla.

Display Comprobamos que la hemos creado.

Seleccionamos el fichero a tratar y las opciones de análisis.

Files: Analyze:

Options

> **+syn: Analyze for synthesis.**

VHDL Library > Librería a usar.

Display File List Seleccionamos el fichero a usar.

Comprobamos y seleccionamos los módulos que usamos en el diseño:

Libraries: List Units Seleccionamos, en nuestro caso 'fichero.Behavioral'

Units: Ready to Synthesize.

Hemos comprobado si podemos realizar la síntesis, en caso afirmativo pasamos a ello:

Units: Synthesize

Vemos como en este momento cambia el 'prompt'. Esta es la fase en la que tomaremos las decisiones que realmente influyen en la síntesis del diseño. En primer lugar elegimos que tipo de circuito queremos sintetizar y sobre que librería lo mapearemos (Figura 2):

Session: Options: Synthesis Library > mie24Lib

Indicamos cuales son las restricciones que queremos imponer al diseño:

Constraints: Global: Maximum Cost > 100000

Y realizamos, por fin, una ejecución del sintetizador y optimizador. Se le indica si queremos optimizar, si queremos generar un esquemático, los tipos de optimización, el nivel de 'esfuerzo' de computación - tiempo de CPU consumido -, etc. (Figura 3):

Run: Synthesizer

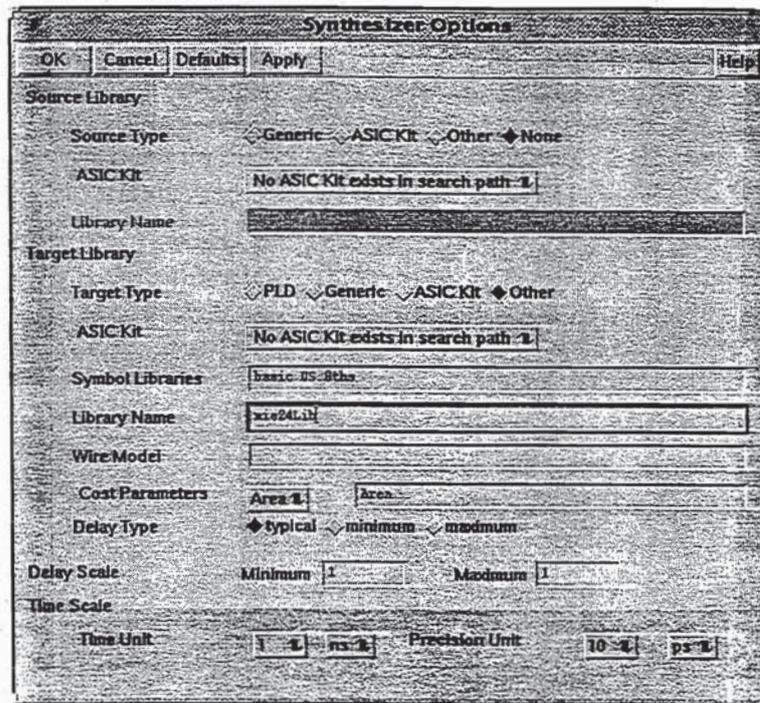


Figura 2.- Definición de la librería de celdas estándar sobre las que queremos mapear el diseño.

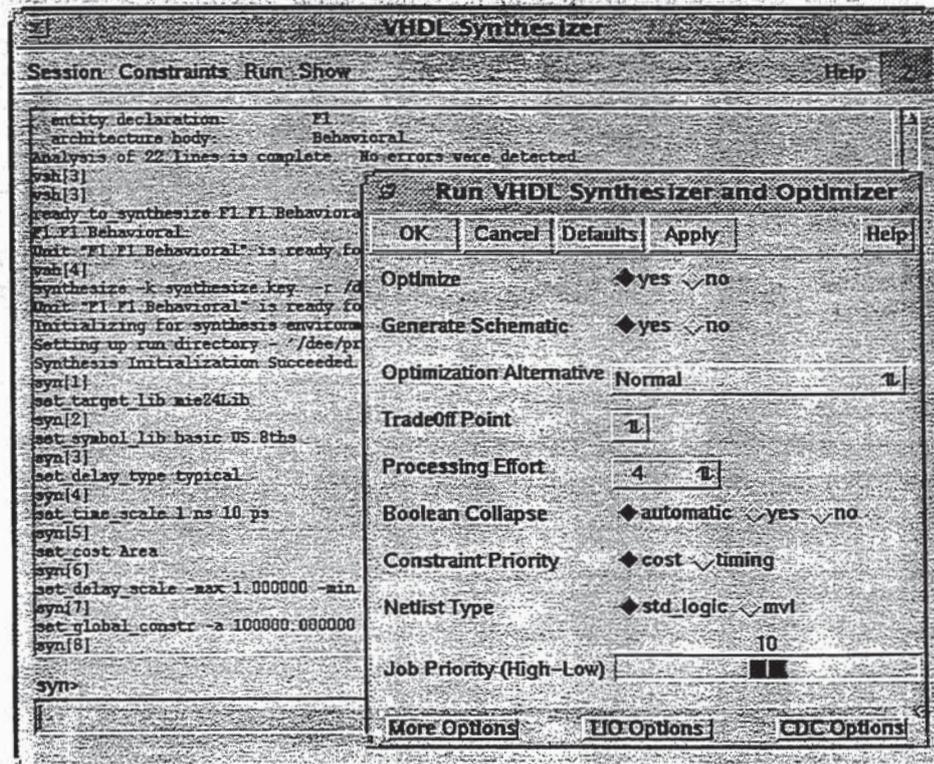


Figura 3.- Definición de los parámetros de síntesis y opciones de optimización.

Por fin podremos ver el esquema de nuestro circuito o los resultados en área y tiempo:

Show: Output: Schematic

Show: Report:

4.3.- Resultados de la síntesis. Circuitos obtenidos

En este apartado presentamos algunos de los resultados de la práctica. Las siguientes figuras, Figuras 4 y 5, muestran el esquema circuital obtenido tras repetir los pasos, anteriormente descritos, en dos casos diferentes: Una función OR exclusiva de cuatro entradas, buscando la solución óptima según consumo de área o según tiempo de propagación. Nótese que, para una misma función lógica, cuando buscamos las dos soluciones extremas, solo es necesario repetir el paso de síntesis (Run: Synthesizer) modificando la opción 'Constraint Priority Cost / Timing' (Figura 3).

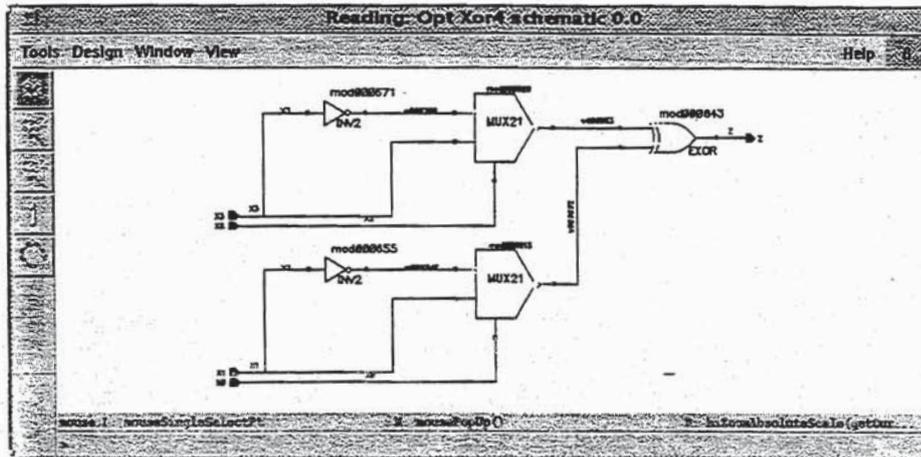


Figura 4.- OR exclusiva de 4 entradas , optimizada por tiempo de propagación (Mietec, CMOS 2.4μ).

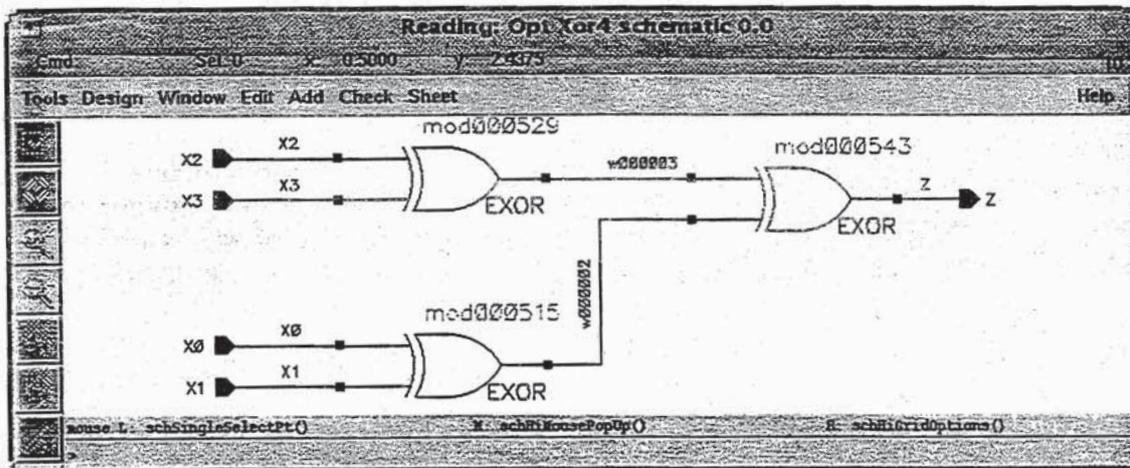


Figura 5.- XOR de 4 entradas, optimizada por área consumida (Mietec, CMOS 2.4 μ)

4.5.- Comparación entre tecnologías.

Ahora podemos mapear cualquiera de los diseños en otra tecnología totalmente distinta. Por ejemplo para obtener F1 en una tecnología ES2 CMOS de 1 μ, solo tendríamos que repetir el paso de selección de librería (Figura 2) substituyendo, en este caso, 'mie24Lib' por 'Syn_ind' (al preparar la sesión, el responsable debe haber proporcionado la localización de esta librería de ES2 en el sistema de ficheros). El siguiente paso es de nuevo realizar la llamada al sintetizador (Figura 3). El resultado sería el mostrado en la Figura 6.

Pese a la diferencia de aspecto, debido al distinto proveedor de las librerías, el alumno puede comprobar como el esquema lógico es el mismo que en la realización con Mietec 2.4μ. De todos modos, al cambiar de tecnología, aparecen grandes diferencias en los resultados de consumo de área y tiempo de propagación (Figura 7).

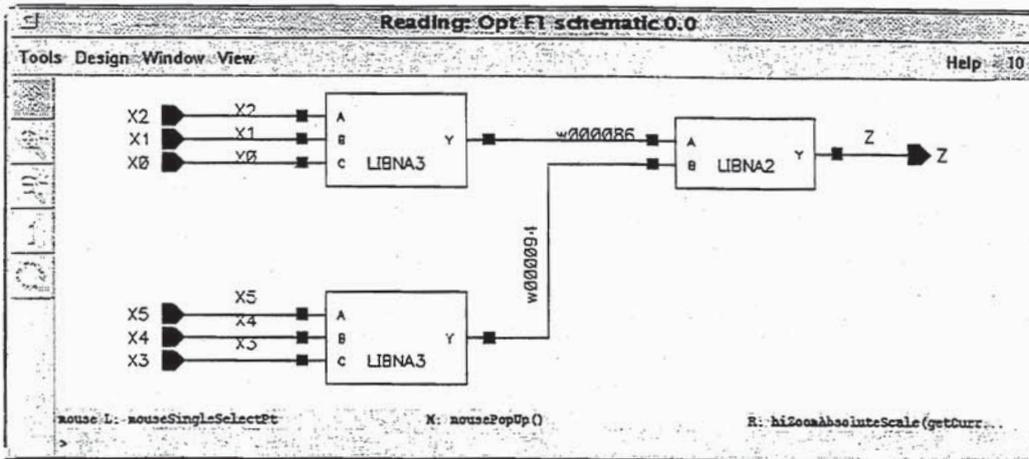


Figura 6.- Función F1, solución óptima por área consumida (ES2, CMOS 1 μ).

File Help 10

.....
Cadence Design Systems, Inc.
Optimizer Area Report File
.....
Circuit: Work.F1.Behavioral
.....

Module: Work.F1.Behavioral

Cell	Count	Area each	Area total
HAND2	1	3180.00	3180.00
HAND3	2	5300.00	10500.00
Total	3		13780.00

Block Summary:
No blocks

Node Summary:
Z output wire 1 bit
X5 input wire 1 bit
X4 input wire 1 bit
X3 input wire 1 bit
X2 input wire 1 bit
X1 input wire 1 bit
X0 input wire 1 bit

Resource Summary:
No resource limits

File Help 5

.....
Cadence Design Systems, Inc.
Optimizer Area Report File
.....
Circuit: Work.F1.Behavioral
.....

Module: Work.F1.Behavioral

Cell	Count	AreaS each	AreaS total
LIBNA2	1	593.75	593.75
LIBNA3	2	819.38	1638.75
Total	3		2232.50

Block Summary:
No blocks

Node Summary:
Z output wire 1 bit
X5 input wire 1 bit
X4 input wire 1 bit
X3 input wire 1 bit
X2 input wire 1 bit
X1 input wire 1 bit

Figura 7.- Consumo de área de F1, realizado con Mietec 2.4 μ y ES2 1 μ .

5.- CONCLUSIONES

En este momento tenemos unos resultados que nos permitirían realizar sucesivas prácticas, simulación lógica, ensamblado de celdas, simulación eléctrica... Pero estos puntos, además de requerir sucesivas sesiones, quedan fuera de nuestros objetivos iniciales.

Podemos concluir que se han obtenido unos resultados correctos, y suficientemente diferenciados como para llamar la atención del alumno. El alumno ha podido comprobar los distintos resultados obtenidos al implementar una misma función. Ha podido comprobar las diferencias entre dos tecnologías, ambas CMOS pero de distinto nivel de integración. Y por último ha podido comprobar como, independientemente de la complejidad de la función, el esfuerzo realizado para su síntesis se reduce al mínimo gracias a las herramientas utilizadas. De esta forma consideramos cumplidos los objetivos marcados.