Laboratorio de Diseño Microelectrónico, 4º Curso, P94

FPGA: Nociones básicas e implementación

M. L. López Vallejo y J. L. Ayala Rodrigo

Abril 2004

Índice

1. Introducción **1**
1.1. Evolución de los dispositivos programables 1
1.2. Estructura general de las FPGAs 2

2. Arquitectura de las FPGA de Xilinx **4**
2.1. Tecnología de programación 4
2.2. Descripción de las principales familias 5
2.3. Arquitectura de la familia XC2000 6

3. Especificaciones **7**
3.1. Estructura 8
3.2. Bloques lógicos 8
3.3. Descripción detallada de la LUT 9
3.3.1. Implementación de memorias CMOS 10
3.4. Recursos de interconexión 11
3.5. Circuito de control de programación 12

Referencias **12**

1. Introducción

Cuando se aborda el diseño de un sistema electrónico y surge la necesidad de implementar una parte con *hardware* dedicado son varias las posibilidades que hay. En la figura 1 se han representado las principales aproximaciones ordenándolas en función de los *parámetros coste, flexibilidad, prestaciones* y *complejidad*. Como se puede ver, las mejores prestaciones las proporciona un diseño *full-custom*, consiguiéndose a costa de elevados costes y enorme complejidad de diseño. En el otro extremo del abanico de posibilidades se encuentra la implementación *software*, que es muy barata y flexible, pero que en determinados casos no es válida para alcanzar un nivel de prestaciones relativamente alto.

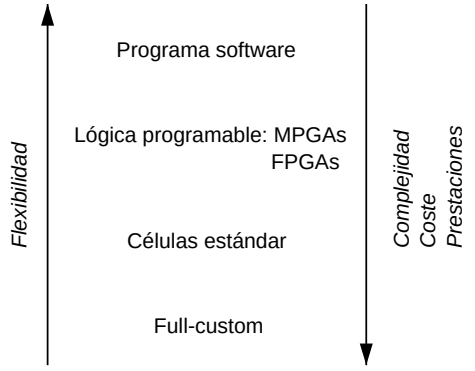

Figura 1: Diferentes soluciones para el diseño de circuitos digitales

Entre estas dos opciones se puede elegir la fabricación de un circuito electrónico realizado mediante diseño *semi-custom*, utilizando células estándar, o recurrir a un circuito ya fabricado que se pueda "programar" in situ, como son las FPGAs. De estas dos opciones la primera proporciona mejores prestaciones, aunque es más cara y exige un ciclo de diseño relativamente largo. Por otro lado, los dispositivos lógicos programables constituyen una buena oferta para realizar diseños electrónicos digitales con un buen compromiso coste-prestaciones. Y lo que es mejor, permiten obtener una implementación en un tiempo de diseño asombrosamente corto (con la consiguiente reducción del parámetro clave: *Time to market*).

Otro aspecto que se debe tener en cuenta para decidirse por este tipo de implementación es que el coste de realización es muy bajo, por lo que suele ser una buena opción para la realización de prototipos.

En estas notas vamos a describir de forma muy somera en qué consisten estos dispositivos, particularizando para una familia del fabricante Xilinx. La información contenida en ellas se basa en gran medida en las siguientes fuentes: sobre arquitectura de FPGAs [BFRV92, Xil91, CSR+99a]; sobre diseño de circuitos [WE94, CSR+99b].

1.1. Evolución de los dispositivos programables

Se entiende por dispositivo programable aquel circuito de propósito general que posee una estructura interna que puede ser modificada por el usuario final (o a petición suya, por el fabricante) para implementar una amplia gama de aplicaciones. El primer dispositivo que cumplió estas características era una memoria **PROM**, que puede realizar un comportamiento de circuito

utilizando las líneas de direcciones como entradas y las de datos como salidas (implementa una tabla de verdad). Hay dos tipos básicos de PROM:

1. Programables por máscara (en la fábrica), proporcionan mejores prestaciones. Son las denominadas de conexiones *hardwired*.
2. Programables en el *campo* (*field*) por en usuario final. Son las EPROM y EEPROM. Proporcionan peores prestaciones, pero son menos costosas para volúmenes pequeños de producción y se pueden programar de manera inmediata.

El **PLD**, Programmable Logic Device, es una matriz de puertas AND conectada a otra matriz de puertas OR más biestables. Cualquier circuito lógico se puede implementar, por tanto, como suma de productos. La versión más básica del mismo es una PAL, con un plano de puertas AND y otro fijo de puertas OR. Las salidas de estas últimas se pueden pasar por un biestable en la mayoría de los circuitos del mercado.

- Ventaja: son bastante eficientes si implementan circuitos no superiores a unos centenares de puertas.
- Inconvenientes: arquitectura rígida, y está limitado por un número fijo de biestables y entradas/salidas.

La **PLA**, Programmable Logic Array, es más flexible que la PAL: se pueden programar las conexiones entre los dos planos. Estos dispositivos son muy simples y producen buenos resultados con funcionalidades sencillas (sólo combinacional). Hace falta algo un poco más sofisticado y general: una matriz de elementos variados que se puedan interconectar libremente. Este es el caso de una MPGA (Mask-Programmable Gate Array), cuyo principal representante está constituido por un conjunto de transistores más circuitería de E/S. Se unen mediante pistas de metal que hay que trazar de forma óptima, siendo ésta la máscara que hay que enviar al fabricante.

Las **FPGA** (Field Programmable Gate Array), introducidas por Xilinx en 1985, son el dispositivo programable por el usuario de más general espectro. También se denominan LCA (Logic Cell Array). Consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una FPGA son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

1.2. Estructura general de las FPGAs

El proceso de diseño de un circuito digital utilizando una matriz lógica programable puede descomponerse en dos etapas básicas:

1. Dividir el circuito en bloques básicos, asignándolos a los bloque configurables del dispositivo.
2. Conectar los bloques de lógica mediante los conmutadores necesarios.

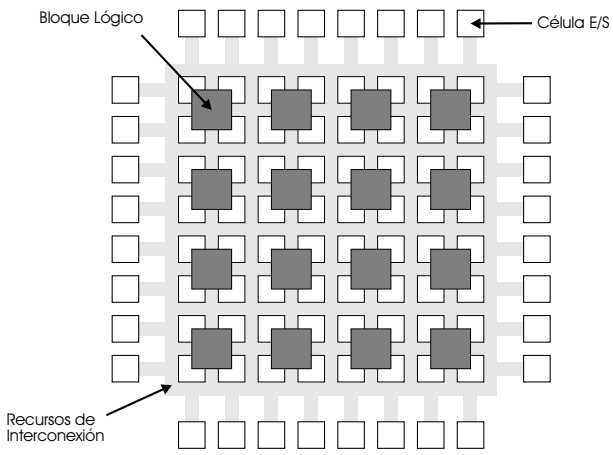

Figura 2: Estructura general de una FPGA (en concreto de XILINX)

Para ello el fabricante proporciona las herramientas de diseño adecuadas.

Los elementos básicos constituyentes de una FPGA como las de Xilinx se pueden ver en la figura 2 y son los siguientes:

1. Bloques lógicos, cuya estructura y contenido se denomina arquitectura. Hay muchos tipos de arquitecturas, que varían principalmente en complejidad (desde una simple puerta hasta módulos más complejos o estructuras tipo PLD). Suelen incluir biestables para facilitar la implementación de circuitos secuenciales. Otros módulos de importancia son los bloques de Entrada/Salida,

2. Recursos de interconexión, cuya estructura y contenido se denomina arquitectura de rutado.

3. Memoria RAM, que se carga durante el RESET para configurar bloques y conectarlos.

Entre las numerosas ventajas que proporciona el uso de FPGAs dos destacan principalmente: el bajo coste de prototipado y el corto tiempo de producción. No todo son ventajas. Entre los inconvenientes de su utilización están su baja velocidad de operación y baja densidad lógica (poca lógica implementable en un solo chip). Su baja velocidad se debe a los retardos introducidos por los conmutadores y las largas pistas de conexión.

Por supuesto, no todas las FPGA son iguales. Dependiendo del fabricante nos podemos encontrar con diferentes soluciones. Las FPGAs que existen en la actualidad en el mercado se pueden clasificar como pertenecientes a cuatro grandes familias, dependiendo de la estructura que adoptan los bloques lógicos que tengan definidos. Las cuatro estructuras se pueden ver en la figura 3, sin que aparezcan en la misma los bloques de entrada/salida.

1. Matriz simétrica, como son las de XILINX
2. Basada en canales, ACTEL
3. Mar de puertas, ORCA
4. PLD jerárquica, ALTERA o CPLDs de XILINX.

En concreto, para explicar el funcionamiento y la estructura básica de este tipo de dispositivos programables sólo se considerarán las distintas familias de XILINX.

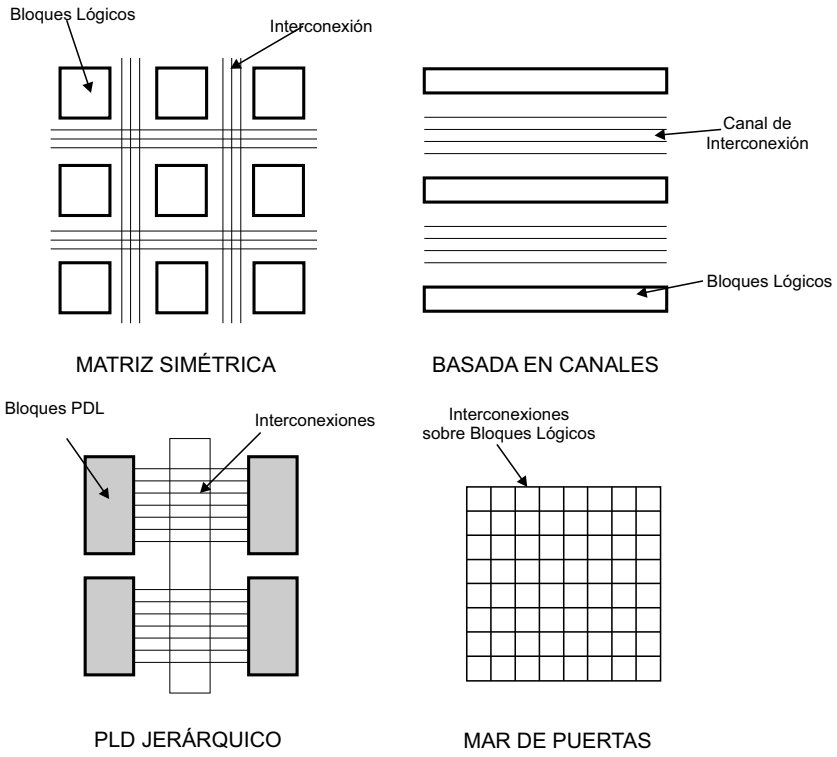

Figura 3: Tipos de FPGAs

2. Arquitectura de las FPGA de Xilinx

2.1. Tecnología de programación

Antes de continuar con conocimientos más avanzados acerca de FPGAs (de XILINX en concreto), hay que aclarar cómo se realiza el proceso de programación (ie., las conexiones necesarias entre bloques y pistas). En primer lugar, si se piensa que el número de dispositivos de conexión que hay en una FPGA es muy grande (típicamente superior a 100.000), es necesario que cumplan las siguientes propiedades:

- Ser lo más pequeños que posible.
- Tener la resistencia ON lo más baja posible, mientras la OFF ha de ser lo más alta posible (para que funcione como conmutador).
- Se deben poder incorporar al proceso de fabricación de la FPGA.

El proceso de programación no es único, sino que se puede realizar mediante diferentes "tecnologías", como son células RAM estáticas, transistores EPROM y EEPROM, etc. En el caso de las FPGAs de XILINX los elementos de programación se basan en células de memoria RAM que controlan transistores de paso, puertas de transmisión o multiplexores. En la figura 4 se puede ver esquemáticamente cómo son. Dependiendo del tipo de conexión requerida se elegirá un modelo u otro.

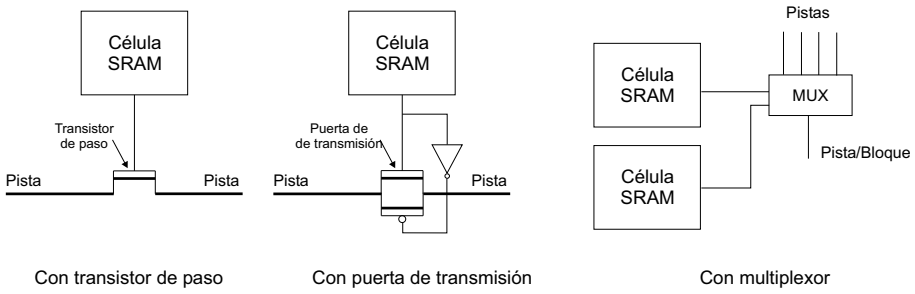

Figura 4: Tipos de conectores utilizados por XILINX

Es importante destacar que si se utilizan células SRAM la configuración de la FPGA será válida únicamente mientras esté conectada la alimentación, pues es memoria volátil. En los sistemas finales está claro que hace falta algún mecanismo de almacenamiento no volátil que cargue las células de RAM. Esto se puede conseguir mediante EPROMs o disco.

Este elemento de programación es relativamente grande (necesita por lo menos 5 transistores), pero se puede implementar en el proceso normal de fabricación del circuito (es CMOS). Además, permite reconfigurar la FPGA de una forma muy rápida.

2.2. Descripción de las principales familias

Hay múltiples familias lógicas dentro de XILINX. Las primeras que surguieron son: XC2000 (descatalogada en el año 1999), XC3000 y XC4000, correspondientes respectivamente a la primera, segunda y tercera generación de dispositivos, que se distinguen por el tipo de bloque lógico configurable (CLB) que contienen. En la actualidad existen también las familias de FPGA SpartanII, SpartanIII, Virtex, VirtexII y VirtexPro. La tabla 1 muestra la cantidad de CLBs que puede haber en cada FPGA de las familias base y ese mismo valor expresado en puertas equivalentes.

SERIE	Tipo CLB	Nº de CLBs	Puertas Equivalentes
XC2000	1 LUT, 1 FF	64-100	1.200-1.800
XC3000	1 LUT, 2 FF	64-484	1.500-7.500
XC4000XL	3 LUT, 2 FF	64-3.136	1.600-180.000

Cuadro 1: Familias del fabricante XILINX

El bloque lógico ha de ser capaz de proporcionar una función lógica en general y reprogramable. La mejor forma de realizar esto es mediante una tabla de valores "preasignados" o "*tablas de look-up*". Básicamente, una tabla de look-up (LUTs en adelante) es una memoria, con un circuito de control que se encarga de cargar los datos. Cuando se aplica en una dirección las entradas de la función booleana la memoria devuelve un dato, lo que se puede hacer corresponder con la salida requerida. Falta añadir los componentes necesarios para desempeñar funciones no implementables con una memoria, tales como una batería de registros, multiplexores, buffers etc. Estos componentes están en posiciones fijas del dispositivo.

La ventaja de la utilización de este tipo de tablas es su gran flexibilidad: una LUT de k entradas puede implementar cualquier función booleana de k variables, y hay 2^{2^k} funciones posibles. El inconveniente es obvio: ocupan mucho espacio y no son muy aprovechables.

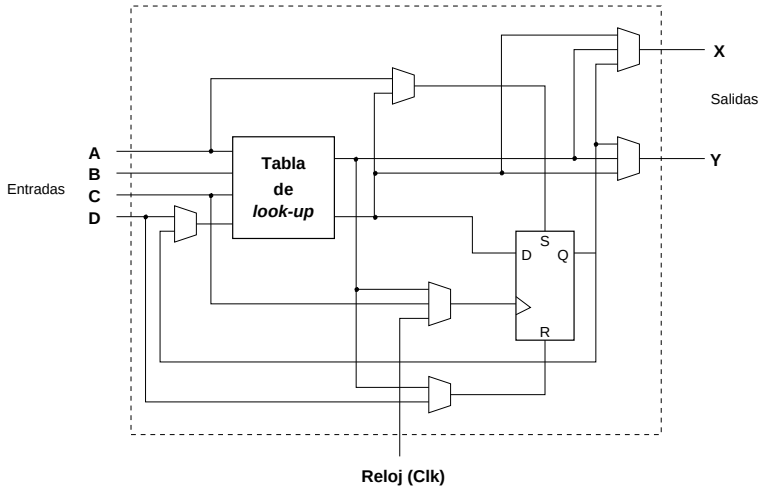

Figura 5: Arquitectura del CLB de la XC2000

Los bloques lógicos configurables de la familia XC2000 se componen de una tabla de look-up con cuatro entradas y un biestable, con lo que puede generar cualquier función de hasta 4 variables o dos funciones de 3 variables. El de la familia XC3000 es más complejo: permite implementar una función de 5 variables o dos funciones de 4 variables (limitadas a 5 diferentes entradas, claro). Además contiene dos biestables y cierta lógica. La familia XC4000 es ya mucho más sofisticada. Tiene tres tablas de look-up dispuestas en dos niveles, llegando a poder implementar funciones de hasta 9 variables.

En general, los recursos de interconexión son de tres tipos:

- Conexiones directas, permiten la conexión de las salidas del CLB con sus vecinos más directos (N, S, E y O).
- Interconexiones de propósito general, para distancias superiores a un CLB (más allá del vecino). Son pistas horizontales y verticales del tamaño de un CLB, pero que se pueden empalmar para crear pistas más largas.
- Líneas de largo recorrido, suelen cubrir lo ancho o largo de la pastilla. Permiten conexiones con un retardo mucho menor que uniendo las anteriores.

El camino crítico de un circuito es el recorrido que, desde una entrada hasta una salida, presenta un retardo máximo.

2.3. Arquitectura de la familia XC2000

Aunque hoy en día no se encuentran disponibles las FPGAs de esta familia, dado que contienen la arquitectura más sencilla, vamos a utilizarlas como base para comprender el funcionamiento de este tipo de dispositivos.

En la figura 5 se puede ver cómo es el bloque configurable básico de las XC2000. Contiene como elementos principales una tabla de *look-up* de 4 entradas y un biestable D. La tabla de *look-up* puede reproducir cualquier función de cuatro variables o dos funciones de tres variables. De las dos salidas del CLB una se puede registrar, o se pueden dejar las dos combinacionales.

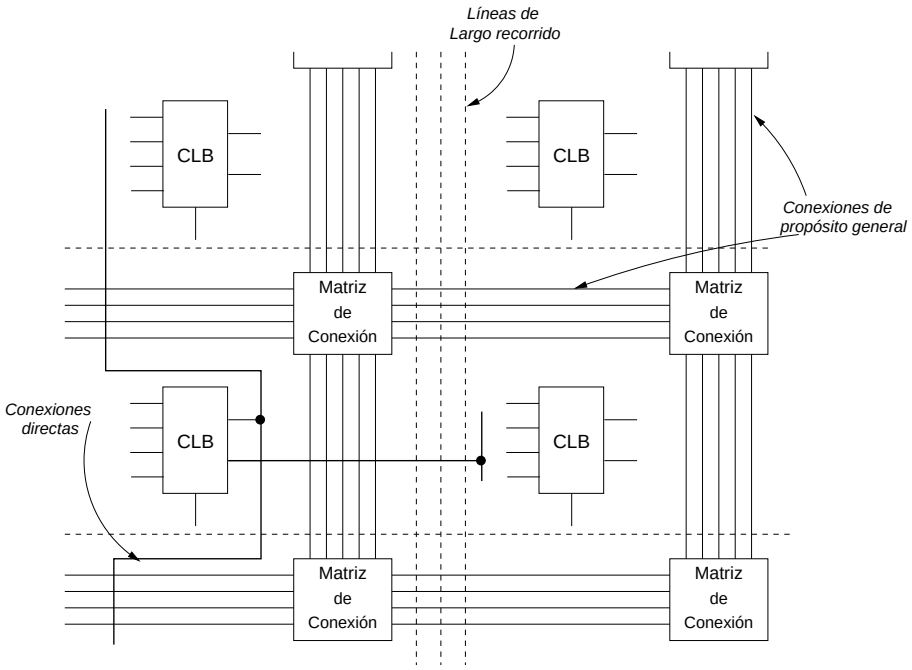

Figura 6: Recursos de interconexión en la familia XC2000

Adicionalmente, en el bloque hay 6 multiplexores que permitirán seleccionar las conexiones que se desea hacer dentro de cada CLB particular. Por ello, en sus terminales de selección necesitarán un elemento de memoria con el valor deseado (ver figura 4). Nótese que la salida del biestable se puede llevar de vuelta a una de las entradas de la LUT, siempre y cuando se configuren adecuadamente los selectores oportunos. Esto es muy útil, pues permite implementar estructuras realimentadas como son contadores o máquinas de estados.

Por otro lado, la arquitectura de rutado de la familia XC2000 utiliza tres tipos de recursos de interconexión: conexiones directas, conexiones de propósito general y líneas de largo recorrido. Todos estos recursos se pueden ver en la figura 6. Las conexiones directas (en la figura aparecen sólo para un CLB) proporcionan enlace desde la salida de un CLB hasta sus vecinos superior, inferior y a la derecha. Si hay que conectar una red a un bloque más lejano hay que utilizar las conexiones de propósito general, que son segmentos de pista dispuestas horizontal y verticalmente a lo largo de toda la FPGA. En particular, en esta familia hay cuatro segmentos horizontales y cinco verticales por canal. Su longitud está limitada siempre a la distancia fija entre 2 CLBs, por lo que para realizar conexiones más largas hay que utilizar las matrices de interconexión. Es importante observar que la utilización de estos recursos repercutirá negativamente en las prestaciones del diseño, pues los conectores de la matriz introducen forzosamente un retardo. Las líneas de largo recorrido se utilizan para conexiones que han de llegar a varios CLBs con bajo *skew*.

3. Especificaciones

Como ya se ha comentado, vamos a diseñar una versión simplificada de una FPGA de la familia XC2000 de Xilinx. Todo el diseño debe seguir las especificaciones que acompañan

este enunciado (sacadas de [Xil91]), permitiéndose la realización de las simplificaciones que se consideren oportunas. A continuación se darán algunas directrices sobre qué simplificaciones se pueden hacer y cuál es la forma de abordarlas.

Ha de quedar claro desde el principio que este diseño tiene un margen de libertad bastante grande. Es decir, NO hay que hacer una FPGA idéntica a una de la familia XC2000, sino que se deben adaptar las especificaciones de la misma para poder utilizar las técnicas vistas en clase: diseño con doble fase de reloj, elementos CMOS de cualquiera de los tipos estudiados, etc.

3.1. Estructura

La FPGA que se debe realizar constará de 16 CLBs. En principio basta con este tamaño, pues el diseño ha de ser completamente regular y estructurado, caracterizando perfectamente a nivel de módulos, lo que lo hará fácilmente ampliable. Recordamos que la meta que se persigue es un diseño correcto, con un trazado simple y bien estructurado.

Además de los 16 CLBs, es necesario implementar los recursos de conexión (tanto las líneas como las matrices de interconexión) y el sistema de control que gestione la programación de toda la FPGA.

Para facilitar la comprensión del problema, hay que ver en el diseño dos estructuras *"superpuestas"*: una correspondiente a la FPGA propiamente dicha, como la ve el diseñador final; otra estructura constituye el entramado necesario para realizar la programación de la FPGA. Esta última parte es transparente para el usuario.

Es importante destacar que la programación de la FPGA no es crítica en cuanto a prestaciones, porque se hace una única vez y no es determinante. Sólo ha de ser **correcta**. Sin embargo, sí interesa optimizar el funcionamiento de la FPGA una vez programada para la implementación final.

3.2. Bloques lógicos

La arquitectura del CLB de la XC2000 se presentó en la figura 5, siendo sus componentes principales:

- Una LUT de 4 entradas y 2 salidas
- Un biestable.
- Seis multiplexores, cada uno con sus correspondientes elementos de memoria.

La LUT que vamos a implementar es como se describió en el ejercicio 4 (apartado 3.3). Es decir, ha de ser capaz de implementar dos funciones de 4 entradas independientes[1].

En Xilinx, el biestable del CLB dispone de una entrada de "reloj" o *enable* que se puede excitar con tres líneas:

- La entrada de reloj `clk`.
- La entrada de propósito general `C`.
- La función combinacional `G` (salida de la LUT).

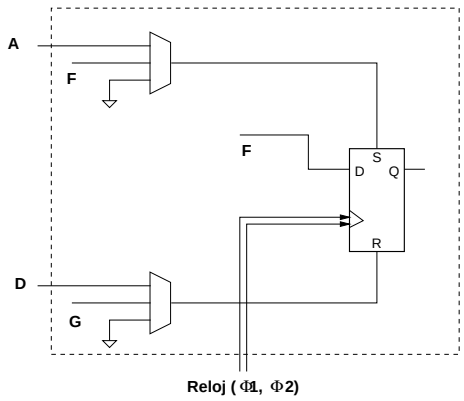

Figura 7: Biestable del CLB

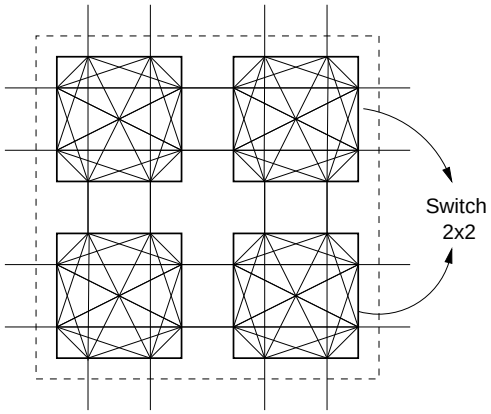

Figura 8: Matriz de interconexión

Dado que nosotros trabajamos con dos fases de reloj, no tiene mucho sentido que se puedan utilizar otras señales como reloj, por lo que una implementación perfectamente válida del CLB sería la de la figura 7. En cuanto al tipo de biestable, el que se encuentra realmente en el CLB de Xilinx es configurable, de forma que se puede programar como *latch* D sensible al nivel o biestable D sensible al flanco. Para nuestros propósitos sería suficiente implementar uno de los dos tipos de biestable[2], sin que sea programable. Con esto el número de multiplexores del CLB que nos hacen falta es menor.

Cualquier otra simplificación que se haga al CLB habrá de justificarse de forma similar a la realizada en el párrafo anterior.

3.3. Descripción detallada de la LUT

Hasta ahora, se ha presentado el CLB de la XC2000 desde el punto de vista de un usuario final. Dado que ahora se va a implementar siguiendo una metodología *full-custom*, es necesario conocer más detalles. En la figura 9 se ve el conjunto de entradas y salidas que tiene realmente la LUT. Como se puede observar, en el símbolo aparecen dos entradas nuevas, las relacionadas con la programación de la LUT:

- Las líneas de datos `Din`, sirven para introducir en la LUT los datos de programación.
- La línea de programación `Prog`, su valor será diferente en función del modo en que se esté (programación/funcionamiento normal).

Como ya se ha explicado previamente, la forma inmediata de implementar una LUT es utilizar una memoria RAM estática. Por ejemplo, considera la tabla de verdad de la función $f = ab + \bar{c}$ que aparece en la figura 10. Si esta función lógica se implementa con una LUT de tres entradas, será necesaria una RAM de $2^3 = 8$ posiciones, y almacenaremos 1 en la posición 000, 0 en la posición 001, y así sucesivamente.

[1]NOTA: las salidas de la LUT en las especificaciones de Xilinx se llaman **F** y **G**, siendo **X** e **Y** las salidas del CLB; adoptaremos esta notación a partir de ahora.

[2]Atención, un biestable no es un registro dinámico.

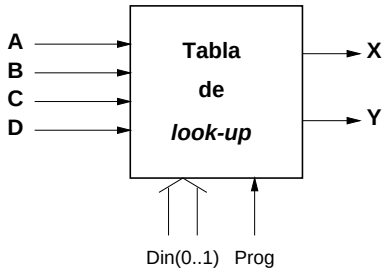

Figura 9: Símbolo de la LUT

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

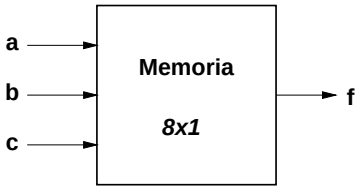

Figura 10: Implementación LUT con SRAM

Si queremos realizar una LUT de dos salidas lo podemos hacer con 2 memorias de 2^k posiciones, o lo que es mejor, con una memoria de k líneas de dirección y 2 bit de palabra. Por supuesto, se está ocupando mucha área, pero esta es la mejor forma de realizar una función booleana cualquiera. Se deja al alumno decidir la mejor forma de implementar la memoria para conseguir un buen compromiso área-tiempo de acceso.

La equivalencia de terminales entre la LUT y la memoria son los siguientes. Las líneas de entrada corresponden al bus de direcciones; las líneas de datos equivalen a bus de datos de entrada; las salidas (X, Y en nuestro caso) corresponden a bus de datos de salida; y la señal de programación será la línea de lectura/escritura, $\mathtt{R}/\bar{\mathrm{W}}$ de la memoria. Dependiendo del tipo de implementación de la memoria se puede necesitar alguna entrada más (por ejemplo, se puede utilizar alguna señal de reloj para temporizar el funcionamiento de la memoria).

3.3.1. Implementación de memorias CMOS

Una memoria CMOS[3] integra los siguientes componentes básicos:

La célula de memoria. Puede ser estática o dinámica dependiendo de la forma de almacenamiento del dato. Las células estáticas son más grandes que las dinámicas, pero tienen la ventaja de no necesitar refresco. La forma más eficiente de distribuir las células es colocarlas en una matriz cuadrada, como se muestra en la figura 11.

El decodificador de filas, que permite seleccionar la fila de la matriz de células de memoria a la que se quiere acceder (tanto para lectura como para escritura). Para ello se utilizarán parte de las líneas de dirección.

El decodificador de columnas, que permite seleccionar las células de la palabra que se va a leer o escribir entre los datos de la fila proporcionada por el decodificador de filas.

Circuito de escritura de las células adecuadas (las de la palabra deseada).

Circuito de lectura de la fila adecuada (*sense amplifier*).

Circuito de control de línea de bit, que acondiciona esta línea para que sean factibles las operaciones de lectura y escritura.

La forma de distribuir las células en la matriz se describe a continuación. Hay que intentar hacer una matriz lo más cuadrada posible. Siguiendo la terminología de la figura 11, supongamos que tenemos una memoria con 2^k palabras (por lo tanto con k líneas de dirección) y cada palabra

[3]El capítulo 8 de [WE94] contiene una excelente descripción de la implementación de memorias CMOS.

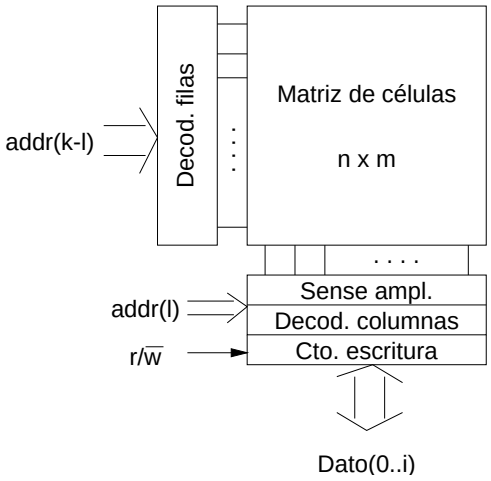

Figura 11: Estructura de una memoria

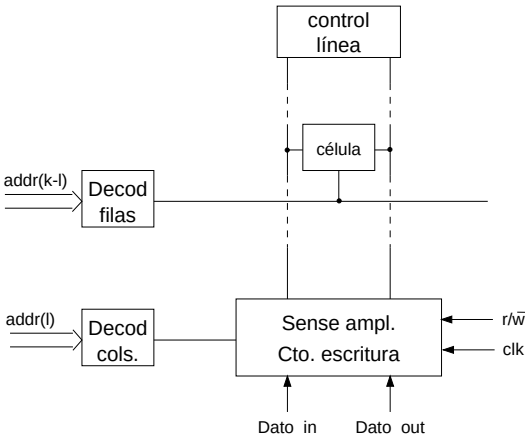

Figura 12: R/W de una célula de memoria

tiene b bits. En total tendremos $2^k \times b$ células. Todas estas células se ordenan en una matriz de dimensiones $n \times m$, donde $m = 2^{k-l}$ y $n = 2^l \times b$, siendo l una parte de las líneas de dirección necesaria para hacer la selección de las columnas. Hay que intentar que n y m sean lo más parecido posible para tener forma cuadrada.

En la figura 12 aparece representada la relación existente entre todos los elementos constituyentes de la memoria y su disposición en las operaciones de lectura y escritura.

3.4. Recursos de interconexión

Como se expuso en la sección 2.3, la arquitectura de rutado de la familia XC2000 utiliza tres tipos de recursos de interconexión: conexiones directas, conexiones de propósito general y líneas de largo recorrido (ver figura 6). Toda esta estructura se puede simplificar para hacerla más fácil de implementar. En esta sección hacemos una propuesta de cómo se pueden realizar.

Comenzando por las conexiones de propósito general, sabemos que se basan en la unión de segmentos de pista entre CLBs. Para simplificar el rutado vamos a considerar que son 4 líneas tanto en sentido horizontal com en vertical. Necesitaremos dos células especiales para realizarlas:

- La *matriz de interconexión* (figura 8), que permite el cruce de pistas horizontales y verticales. No tiene por qué posibilitar la conexión de todas las líneas con todas. Se implementará utilizando 4 *switch* de 2x2.
- *Módulos de conexión* de las entradas y salidas del CLB a los segmentos de pista de las conexiones de propósito general. Un ejemplo de cómo se puede hacer estas conexiones se ve en la figura 13. Como se ve, hay que facilitar el acceso a las entradas del CLB y la salida del mismo. No tiene por qué ser el acceso sólo a las líneas verticales; esto dependerá fundamentalmente del trazado que se realice del CLB.

Siguiendo esta propuesta, las conexiones directas que vamos a implementar son las descritas en el manual de Xilinx:

- La salida `X` se puede conectar a las entradas `A` o `B` del CLB inferior o a las entradas `C` o `D` del CLB superior.

- La salida `Y` se puede conectar a la entrada `B` del CLB situado a la derecha.

Evidentemente esto es para los CLBs internos de la FPGA. Los que estén en la periferia se podrán conectar directamente a los pads de E/S. Por supuesto, las líneas de E/S del CLB no tienen por qué tener la orientación mostrada en la figura, cada proyecto se realizará con la orientación que cada grupo estime mejor.

En cuanto a las líneas de largo recorrido, se deja a voluntad del diseñador implementar las que son programables. En caso de considerarlas, se tendría que elegir qué entradas y salidas del CLB tendrán acceso a las mismas. En la figura 13 se ha presentado como ejemplo dos líneas de largo resorrido y algunas entradas accediendo a las mismas. Hablamos de líneas *programables* porque hay 2 líneas de largo recorrido que en nuestro caso NO serán programables y sí son obligatorias: las líneas que llevan las dos fases del reloj, que como se sabe, se deben rutar con especial cuidado.

3.5. Circuito de control de programación

Esta parte de la FPGA debe ser una extensión del circuito de programación de una LUT. Ha de ser un circuito único que configure forma secuencial toda la FPGA.

Aunque el la familia XC2000 hay varias formas de programar la FPGA, nosotros nos centraremos en el modo más sencillo: el denominado *modo Serial/Slave*. En este caso los datos llegan en serie por un pin de entrada de datos de forma síncrona: un reloj (`CCLK`) valida los datos de la entrada serie (`DIN`). Se deja a alumno definir el formato de los datos de entrada cuando se está en configuración. El control de la FPGA se puede hacer desde un microprocesador, resultando el esquema de conexión más sencillo el que aparece en la figura 14.

Resulta evidente que hay que utilizar los recursos de interconexión para realizar la programación, por lo que será necesario configurar primero las interconexiones para realizar la programación de los CLBs.

Para programar las LUTs, se sugiere como forma sencilla de implementación comenzar configurando todas las líneas para que lleven `a`, `b c` y `d` a todos los CLBs de la FPGA. Luego se utilizarían unas *"líneas de selección de CLB"* que vayan habilitando cada CLB mientras se hace la programación.

Son necesarios unos decodificadores globales que sirvan para llevar el dato a cada fila/columna de CLBs. Esto se puede ver como el acceso en escritura a una memoria.

Si se realiza la escritura por filas o columnas, harían falta unos registros de desplazamiento globales.

Todos los elementos de memoria se deben enlazar siguiendo una filosofía "scan-path".

Referencias

[BFRV92] Stephen D. Brown, Robert J. Francis, Jonathan Rose, and Zvonko G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, Norwell, Massachusetts, 1992.

[CSR+99a] P. Chow, S. Ong Seo, J. Rose, K. Chung, G. Páez-Monzón, and I. Rahardja. The design of an sram-based field programmable gate array-part i: Architecture. *IEEE Trans. on VLSI Systems*, 7(2):191–197, June 1999.

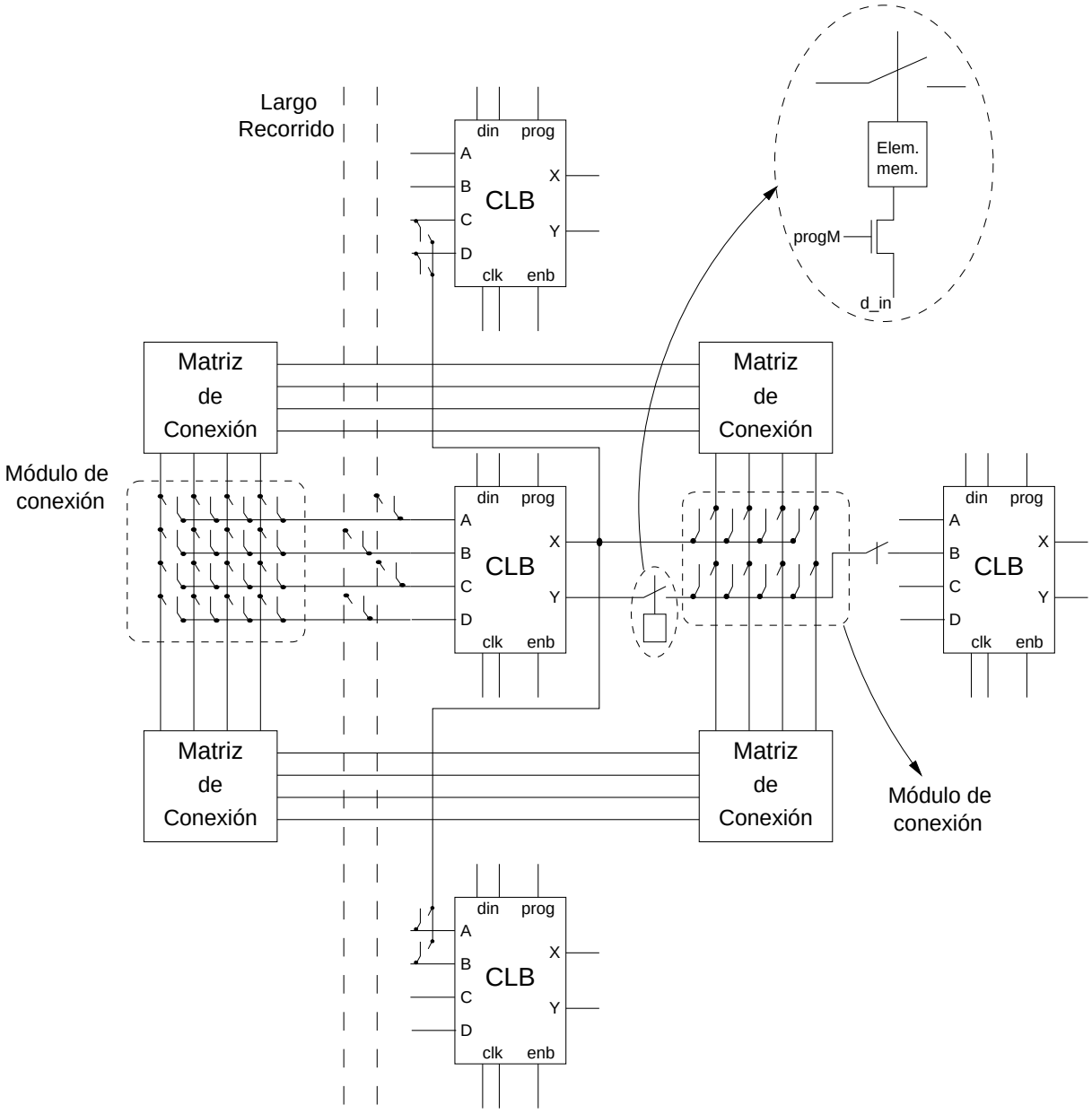

Figura 13: Esquema de posibles conexiones en la FPGA. Detalle de conexión con su elemento de memoria.

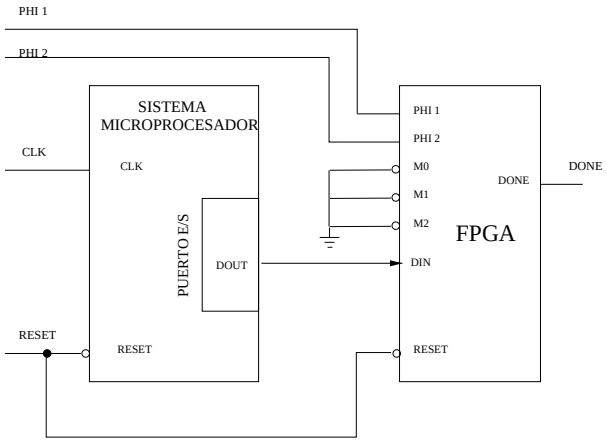

Figura 14: Conexión de programación.

[CSR+99b] P. Chow, S. Ong Seo, J. Rose, K. Chung, G. Páez-Monzón, and I. Rahardja. The design of an sram-based field programmable gate array-part ii: Circuit design and layout. *IEEE Trans. on VLSI Systems*, 7(3):321–330, September 1999.

[WE94] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, 2nd edition, 1994.

[Xil91] Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124. *The Programmable Gate Array Data Book*, 1991.