Universidad Complutense de Madrid

Departamento de Informática y Automática

Técnicas de Estimación de Características Físicas en Síntesis de Alto Nivel

Hortensia Mecha López

TESIS DOCTORAL

Director: Dra. Milagros Fernández Centeno

Madrid, 1996

Memoria presentada por Hortensia Mecha López para optar al título de Doctor en Ciencias Físicas por la Universidad Complutense de Madrid.

Madrid, Abril 1996

Agradecimientos

En primer lugar quiero expresar mi agradecimiento a Dña. Milagros Fernández Centeno por la dedicación y entusiasmo que ha puesto en la elaboración de este trabajo de investigación, y por sus ánimos en los momentos más difíciles.

En segundo lugar, me gustaría mostrar mi agradecimiento a Don Julio Septién del Castillo y a Don Daniel Mozos Muñoz, por todas las horas que han pasado discutiendo distintos aspectos sobre la forma y contenido de este trabajo, así como por sus imprescindibles ánimos para conseguir la finalización del mismo.

Igualmente quiero agradecer a Don Román Hermida sus consejos e ideas, que me han permitido afianzar y explicar resultados de forma mucho más científica y clara.

También quiero agradecer a Don Francisco Tirado que contase conmigo para formar parte de su grupo de investigación, gracias a lo cual ha sido posible el desarrollo de este trabajo.

Tampoco puedo olvidar a Dña. Katzalin Olcoz Herrero, Don Victor Sánchez Canga y Don José Manuel Mendías Cuadros, que me han permitido discutir con ellos multitud de problemas e ideas, y han estado siempre dispuestos a darme ánimos y compañía.

Además, quiero agradecer a Don José Antonio López Orozco y Don Rafael Moreno Vozmediano las horas que han pasado ayudándome, para obtener los resultados sobre estimaciones de retardos presentados en este trabajo.

Por último, quiero dar las gracias a todos los demás miembros del Grupo de Arquitectura de Computadores del Departamento de Informática y Automática, por haberme escuchado y aconsejado siempre que lo he requerido.

Este trabajo ha sido posible gracias al Ministerio de Educación y Ciencia, por el soporte económico dado dentro del Plan de Becas de Formación de Personal Investigador, y gracias a la Comisión Interministerial de Ciencia y Tecnología, por las ayudas recibidas a través de los

proyectos CICYT TIC 92/0088 y CICYT TIC 94/0725-C03-02.

Indice

Resumen

El objetivo principal de este trabajo de investigación es estudiar las influencias de las características físicas del CI sobre el área y retardo de los diseños obtenidos en un proceso de Síntesis de Alto Nivel, y diseñar técnicas de estimación de dichas características, rápidas, precisas y fieles, para todas las fases de dicho proceso.

El primer problema abordado, es la elaboración de una función para medir la calidad de los diseños obtenidos durante la asignación de hardware en un proceso de Síntesis de Alto Nivel. Esta función debe ser una aproximación al área real del circuito, que a su vez es la suma del área de los módulos y de las interconexiones. Éstas áreas dependen de la tecnología que se esté utilizando, de la colocación de los módulos en el circuito final, y de cómo se realice el interconexionado de éstos (tanto de las interconexiones internas de los módulos como de las externas). Por tanto, depende de las características físicas del CI y de las herramientas y tecnologías de diseño, y es necesario estimarlas. En todos los casos las estimaciones se necesitan realizar muchas veces durante un proceso de síntesis, y por tanto deben ser muy **rápidas**. Además deben ser lo suficientemente **fieles** para dirigir correctamente el proceso de asignación de hardware.

En este trabajo se presenta un método de estimación de área que puede utilizarse durante las distintas fases de la síntesis, como en la preasignación y en la asignación de hardware, y en la generación del hardware de control, y que es lo suficientemente rápido y fiel para dirigir el proceso de diseño correctamente, sin incrementar la complejidad de éste. Como las influencias de las características físicas en el área del circuito, dependen de la tecnología de diseño utilizada, y el estudio para todas ellas es un trabajo de una extensión excesiva, se particulariza el estudio para celdas estándar. Sin embargo, muchas de las ideas propuestas pueden utilizarse para macroceldas y arrays de puertas.

El segundo problema que se trata en este trabajo es la obtención de circuitos con un funcionamiento eléctrico correcto. Para este fin, es necesario considerar los retardos de los módulos e interconexiones, que a su vez dependen de la tecnología utilizada y de los algoritmos de colocación e interconexionado de módulos. El retardo del interconexionado es un dato que no se conoce hasta que no se ha generado el layout, y por tanto es necesario estimarlo.

En este trabajo se presenta un algoritmo de selección del tiempo de ciclo que tiene en cuenta la biblioteca de módulos disponible, con información sobre los retardos de los módulos, y el retardo de las interconexiones, mediante estimaciones que consideran la tecnología utilizada y la forma de trabajo de los algoritmos de colocación e interconexionado de módulos. De esta forma, se asegura que los circuitos generados tienen un comportamiento eléctrico correcto. Además, para conseguir tiempos de ciclo óptimos, que permitan cumplir los objetivos del usuario en cuanto al área y tiempo de ejecución del circuito, se realiza un estudio global de la especificación dada y de la biblioteca de módulos.

La memoria que se presenta se divide en seis capítulos y tres apéndices. En el capítulo 1 se realiza una introducción a la Síntesis de Alto Nivel y se plantea la necesidad de realizar estimaciones de las características físicas del circuito para dirigir correctamente el proceso de diseño.

En el capítulo 2 se exponen los principales sistemas de Síntesis de Alto Nivel que realizan algún tipo de estimación del área de interconexionado, así como sus ventajas y sus inconvenientes, y la necesidad de realizar nuevas estimaciones más precisas. A continuación se presentan algunos métodos de estimación de área de circuitos integrados bastante precisos, pero muy lentos para incorporarlos dentro de una herramienta de Síntesis de Alto Nivel.

El capítulo 3 se centra en el sistema FIDIAS, que es un sistema de Síntesis de Alto Nivel dentro del cual se integra este trabajo de investigación. Se muestra que también en este sistema es necesario realizar estimaciones, y cómo y dónde se deben integrar éstas.

En el capítulo 4 se realiza un estudio de la metodología de trabajo de las herramientas de CAD, y de las características físicas de los circuitos integrados. Se han clasificado las interconexiones en función del número de módulos que conectan. Para cada uno de los tipos se ha obtenido una estimación de su longitud. Las interconexiones que hemos denominado cercanas tienen longitud constante. Para las lejanas y multimódulo se ha obtenido una fórmula en función del número total de interconexiones de cada tipo y del área de celdas estándar del diseño, datos conocidos cuando se ha terminado la generación de un diseño. Este método de estimación de la longitud de las interconexiones nos permite conocer el área de los módulos, siempre que en la biblioteca se disponga de información sobre el número y tipo de interconexiones que contiene cada uno de ellos. Las estimaciones se han integrado en el sistema FIDIAS y, mediante varios ejemplos, se ha demostrado la fidelidad y precisión de estas estimaciones de área, que en todos los casos tienen un error inferior al 5%.

En el capítulo 5 se presenta un algoritmo de selección del tiempo de ciclo, que tiene en cuenta la biblioteca de módulos, las restricciones del usuario y que estudia de forma global el GFD. El algoritmo de selección realiza una búsqueda de los caminos que para algún tiempo de ciclo pudieran ser críticos, puesto que son los únicos que influyen en el tiempo de ejecución del algoritmo. Sobre las operaciones situadas en estos caminos, se realiza la selección del tiempo de ciclo, permitiendo más de una Unidad Funcional para implementar cada operación, y mediante unos parámetros que miden los objetivos del usuario, en cuanto a la minimización del área y tiempo del circuito. De esta forma, el tiempo de ciclo utilizado para realizar la planificación depende de cada diseño y objetivos en particular, con lo cuál los resultados obtenidos se ajustan más a las necesidades del usuario.

Por último, en el capítulo 6, y a partir de las estimaciones del área de las interconexiones del capítulo 4, se presenta la forma de calcular el retardo de éstas. Se realiza un estudio completo sobre el retardo del interconexionado, obteniendo un modelo físico para una interconexión, y a partir de éste se calcula el retardo, utilizando el método del polo dominante. Además, se presenta el método para incluir los retardos de las interconexiones en el cálculo del tiempo de

ciclo, con el objetivo de obtener diseños con un funcionamiento eléctrico correcto.

Todos los resultados del capítulo 4 se basan, por una parte en la forma de trabajo de las herramientas de Síntesis de Bajo Nivel, y por otra parte en la tecnología utilizada. En el apéndice A se puede encontrar un estudio detallado sobre las distintas tecnologías de generación de Circuitos Integrados, y en el apéndice B se presentan los distintos algoritmos de colocación e interconexionado de módulos, cuyo conocimiento es necesario para entender los resultados de este trabajo.

Finalmente, en el apéndice C se explica la conexión de nuestra herramienta de Síntesis de Alto Nivel con una herramienta de Síntesis de Bajo Nivel, gracias a la cual se han podido realizar un gran número de experimentos, con los cuáles dirigir y validar la calidad de nuestras estimaciones.

Capítulo 1.

Síntesis de Alto Nivel y Características Físicas

1.1. Introducción

En la última década la tecnología de diseño de **Circuitos Integrados (CIs)** se ha desarrollado de forma espectacular, con la generación de múltiples herramientas de **Diseño Asistido por Computador (CAD)** para desarrollo de CIs y diseño lógico. Este desarrollo fundamentalmente es una consecuencia de dos factores:

- El gran avance de las tecnologías de fabricación de circuitos VLSI, que ha permitido incrementar de forma espectacular la densidad de encapsulamiento, elevando enormemente la complejidad de los diseños. Actualmente muchos diseños incorporan arquitecturas completas en un único chip.

- La necesidad de producir diseños en el menor tiempo posible y con el menor número de errores. Por tanto, se requiere utilizar herramientas para acelerar el proceso de diseño y verificación.

Dadas las consideraciones anteriores, se puede concluir que el nivel lógico no es un nivel natural para el diseñador de circuitos complejos. Es prácticamente inconcebible describir un sistema complejo en términos de expresiones booleanas. Además, la utilización cada vez mayor de **ASICs** (Circuitos Integrados de Aplicación Específica) en múltiples áreas de aplicación, implica el acceso a esta tecnología de usuarios con poca o ninguna experiencia en el diseño de CIs. Estos factores generan la necesidad de utilizar herramientas para automatizar el diseño de circuitos desde un nivel de abstracción más elevado, con el fin de ahorrar tiempo

y dinero, y permitir el acceso a las ventajas de la nueva tecnología a mayor número de usuarios, no necesariamente expertos.

Los posibles dominios de descripción de un diseño y sus niveles de abstracción respectivos han sido representados por Gajski [GaKu83] mediante un diagrama en forma de Y similar al de la figura 1.1. Cada uno de los tres ejes representa uno de los dominios: el **conductual**, el **estructural**, y el **físico**. En cada dominio, la descripción puede realizarse a distintos niveles, de mayor grado de abstracción cuanto más alejados se encuentren del origen.

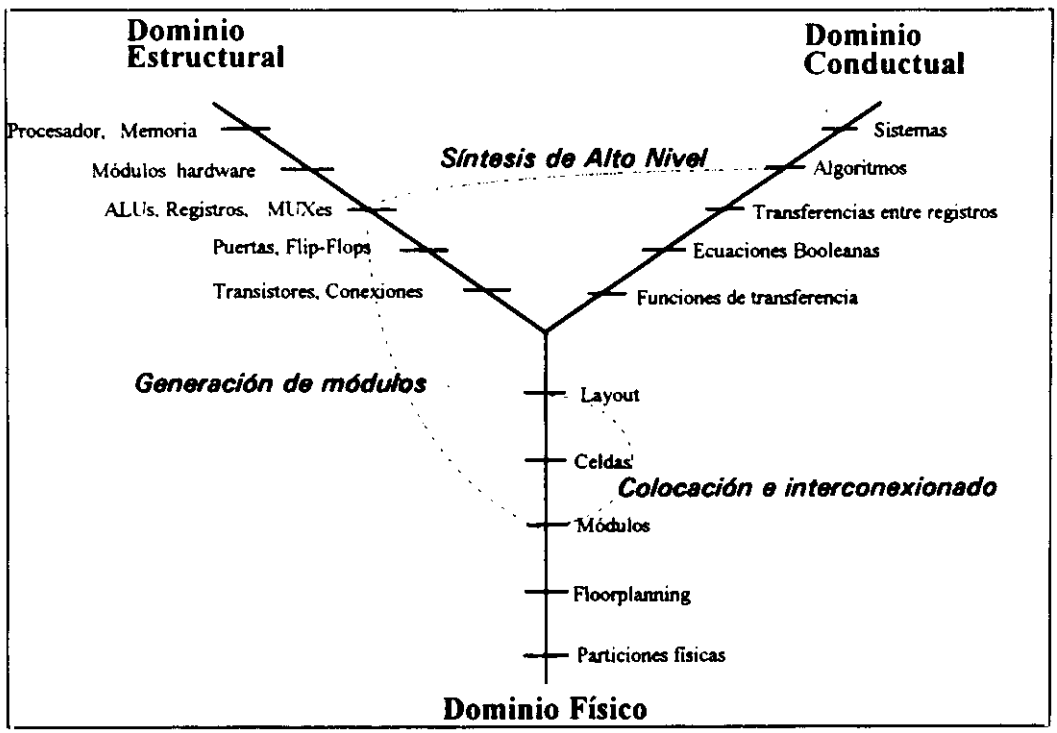


Figura 1.1 Dominios de Descripción y Niveles de Abstracción

En el dominio conductual el circuito se describe mediante una serie de relaciones entre las entradas y las salidas del mismo. En el dominio estructural se especifican los componentes y las interconexiones entre ellos. En el dominio físico se describen los módulos reales que integran el circuito y su distribución espacial.

En la figura 1.1 también se representan un conjunto de transformaciones entre distintos niveles de los diferentes dominios, que pueden constituir un proceso de diseño. Entre todos los

posibles puntos de partida y llegada dentro del desarrollo de CIs, la "**Síntesis de Alto Nivel**" (**SAN**) [GaRa94] parte de una descripción algorítmica del comportamiento del circuito. y obtiene automáticamente la estructura del sistema a nivel de transferencia de registros (RTL). La finalización del diseño se conseguiría mediante la generación de los módulos del nivel RTL, seguido de un proceso de colocación e interconexionado de éstos para generar el layout final (Apéndice B). Estos procesos suelen realizarse mediante la utilización de herramientas comerciales de CAD.

A continuación vamos a centrarnos en el proceso de SAN. El punto de partida, es decir, la descripción del comportamiento del circuito, debe realizarse en un lenguaje de alto nivel (**LAN**). Para facilitar la conexión con otras herramientas que trabajan en otros dominios descriptivos y/o niveles de abstracción, se pueden utilizar lenguajes estándar de descripción de hardware (**LDH**). Entre todos los LDH disponibles, uno de los más aceptados es **VHDL**. Además, el diseñador puede introducir un conjunto de restricciones y objetivos, básicamente de coste y velocidad del diseño, y el resultado proporcionado por la herramienta debe respetarlas.

Por otra parte, en su punto de llegada, el nivel RTL del dominio estructural, se especifican las unidades funcionales (sumadores-restadores, multiplicadores, comparadores, etc.), las unidades de almacenamiento (registros, memorias), y las interconexiones (buses, multiplexores) que componen el diseño, así como la secuencia de control, a partir de algún modelo de unidad de control dado.

Entre las ventajas aportadas por la Síntesis de Alto Nivel [McPC88] no sólo está la disminución del coste de los circuitos debido a la reducción del tiempo de diseño, sino también:

- **Reducción del número de errores**: Si se puede verificar que el proceso de SAN es correcto, podrá asegurarse la corrección por construcción del diseño obtenido. Esto permitirá reducir la probabilidad de aparición de errores y, por consiguiente, el

tiempo invertido en la depuración de los mismos.

- **Posibilidad de explorar el espacio de diseño**: Una buena herramienta de SAN deberá ser capaz de producir diferentes diseños, de distintos estilos, para una misma especificación de partida, siempre dentro del conjunto de restricciones establecido por el diseñador.

- **Posibilidad de auto-documentación**: El sistema de síntesis puede dejar constancia de cada una de las decisiones de diseño, y de los motivos que le indujeron a adoptarla.

- **Acceso de mayor número de usuarios a la tecnología VLSI**: La automatización del proceso de diseño se efectuará desde un nivel de abstracción mayor que los disponibles en las herramientas, y esto permitirá diseñar sus propios CIs a usuarios no necesariamente familiarizados con sus características de más bajo nivel.

1.2. Influencias de las características físicas en las distintas fases de la SAN

De las consideraciones realizadas en el apartado anterior se puede deducir que la SAN es una etapa automática dentro del proceso global de diseño del CI. Sin embargo no puede tratarse como una fase aislada del resto. Los estudios realizados por McFarland y Kowalski [McKo90] sobre las influencias de factores del nivel físico en los resultados finales, nos muestran que, **incluso en las decisiones que se toman en las primeras etapas de diseño, es necesario tener en cuenta las características físicas de los módulos**, tanto del área como del retardo.

En el Apéndice A se presenta un estudio de los diferentes estilos de diseño VLSI que existen hoy en día. Observando las características de cada uno de ellos, puede sacarse como conclusión que los resultados obtenidos para un mismo diseño utilizando diferentes estilos, difieren totalmente tanto en coste final como en rendimiento. Por tanto, durante la SAN es

crucial tener en cuenta el estilo de diseño que se va a utilizar. Al mismo tiempo, los resultados obtenidos por distintas herramientas CAD pueden modificar los resultados, dependiendo del tipo de algoritmos que utilicen para realizar la colocación de los módulos y su interconexionado [MSHO92].

Por consiguiente, para que la síntesis del circuito integrado esté debidamente optimizada **es necesario que durante todo el proceso de síntesis se tengan en cuenta:**

- **Las características físicas de la tecnología y de los módulos particulares que se van a utilizar.**
- **Las herramientas y estilo de diseño que se van a emplear para realizar la colocación e interconexionado de dichos módulos.**

El tema principal de este proyecto es el estudio de las mutuas influencias entre las características físicas de los CIs, las herramientas CAD con que se diseñan y los algoritmos de SAN.

Debido a la complejidad de estas mutuas influencias y, por tanto, de la necesidad de generar una enorme cantidad de experimentos para obtener conclusiones sobre su comportamiento, para una herramienta y un estilo de diseño en particular, existen muy pocos sistemas en la literatura que hayan realizado investigaciones relativas a este tema.

Aunque para cada estilo de diseño pueden existir diferentes tecnologías, cada una de ellas incluye todas las reglas para un estilo de diseño en particular. Por lo tanto, a partir de ahora utilizaremos la palabra tecnología para referirnos a un estilo de diseño y tecnología particulares.

Las características tecnológicas se conocen a priori, y pueden realizarse medidas sobre su influencia en el área y retardo de los módulos. Sin embargo, la colocación e interconexionado de los módulos no se conoce hasta que no se ha generado el layout final del circuito. Para obtener información sobre dicha colocación e interconexionado es interesante realizar la

conexión de la herramienta de SAN con la herramienta de CAD que va a generarlos. De esta forma, es posible experimentar y obtener el suficiente número de diseños para sacar conclusiones sobre la metodología de trabajo de los algoritmos de colocación e interconexionado de módulos y, a partir de ellas, realizar estimaciones sobre las influencias de las características físicas del circuito integrado y utilizarlas en las decisiones que se toman en SAN.

Además, la conexión de la herramienta de SAN con la herramienta de CAD permitirá comprobar si dichas estimaciones son correctas. La realización de estos experimentos es sólo posible si se automatiza en gran parte las interacciones con la herramienta de CAD, porque la interacción manual con el proceso incrementa enormemente el tiempo de diseño.

Como la SAN consta de un grupo de tareas con diferentes funciones, las estimaciones de las características físicas del CI hay que realizarlas para cada una de estas tareas. Seguidamente vamos a presentar las diferentes etapas de la SAN y las estimaciones necesarias en cada una de ellas.

En la figura 1.2 se muestra un esquema de un proceso de SAN. Podemos distinguir 4 tareas fundamentales:

1.- **Compilación**. La primera fase necesaria en cualquier proceso de SAN, es la traducción de la descripción inicial, que como hemos dicho viene expresada en un LAN, a una representación intermedia más fácil de manejar por el resto de los módulos del sistema. Normalmente la representación elegida es un **grafo de flujo de datos (GFD)** y control, en el cual los nodos representan las operaciones a realizar, y las aristas el flujo de los datos. Esta fase es la que se denomina compilación, que, además, es la encargada de analizar la descripción de partida y de realizar sobre ella un conjunto de transformaciones orientadas a su optimización.

2 y 3.- **Planificación y Asignación de Hardware**. Después de la compilación, la mayoría de las herramientas de SAN existentes distinguen dos subtareas que están muy

interrelacionadas entre sí: la planificación de operaciones y la asignación de hardware.

- La primera de estas dos subtareas se encarga de la ubicación temporal, es decir, de asignar las operaciones a "**pasos de control**". Un paso de control es el equivalente a un ciclo de reloj.

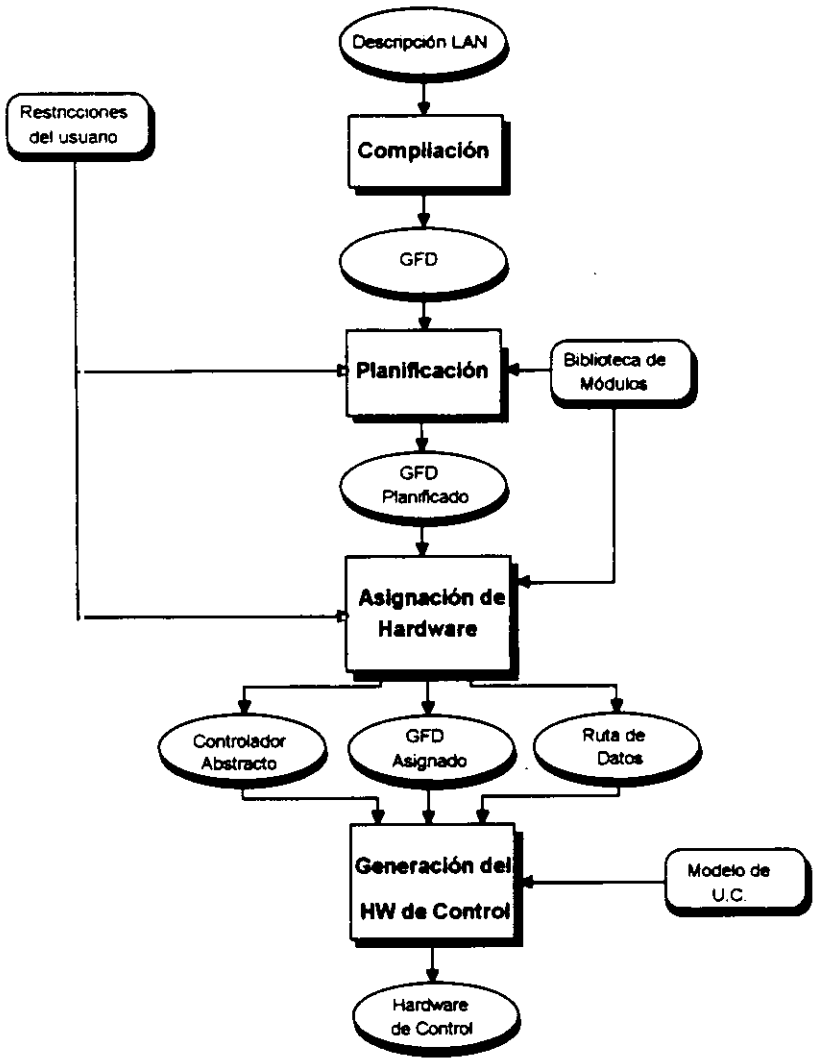


Figura 1.2 Tareas de un proceso de SAN

- La otra subtarea, la asignación de hardware, decide qué elemento hardware va a realizar cada operación, dónde se almacenará el resultado y los caminos de interconexionado necesarios. El módulo asignador recibe como entrada una

biblioteca de módulos, donde están almacenadas las Unidades Funcionales (**UFs**) disponibles para realizar las operaciones. El resultado de esta tarea es una **ruta de datos**, es decir, un conjunto de módulos conectados entre sí.

Esta división en dos subtareas es una consecuencia de que el proceso de síntesis consiste en una doble asignación de las operaciones incluidas en la descripción del comportamiento, a pasos de control y a módulos hardware. La complejidad de tratar estos dos problemas de una forma global es muy grande y por eso, aunque actualmente esta tendencia está cambiando, muchos de los sistemas no abordan la interdependencia existente entre ellos, y tratan cada una de estas tareas por separado. Aún así, cada una de ellas es un problema NP completo.

En una gran parte de los sistemas la planificación es una fase previa a la asignación de hardware, puesto que no es posible decidir qué UF puede realizar una operación si no se sabe de cuanto tiempo se dispone para realizarla. Sin embargo, en sistemas como Cathedral [DCGM90] esto no ocurre así. Primero se decide cuál es el mínimo hardware necesario para realizar todas las operaciones en el tiempo impuesto por el usuario, y luego se realiza la planificación en función de dicho hardware. En la actualidad, la mayoría de los planificadores deciden al menos qué tipo de UF va a realizar cada operación; es lo que se denomina **preasignación de hardware** o **asignación de tipos**. En este caso el asignador de hardware decide qué instancia en concreto realizará la operación (**asignación de instancias**).

4.- **Generación de control**. La última tarea de la SAN es la **generación del control**, que tiene como función la generación de la unidad de control, basándose en la planificación y asignación de hardware realizadas anteriormente.

En los apartados sucesivos vamos a explicar más detenidamente cada una de estas tareas, y las influencias de las características físicas del circuito integrado sobre cada una de ellas.

1.2.1. Compilación

La primera decisión a la hora de implementar un sistema de SAN es la selección del LAN

empleado para la descripción de partida. En los primeros sistemas de SAN, los lenguajes solían ser estándares de tipo procedural (como Pascal o C modificados para detallar algunos aspectos específicos del hardware). Sin embargo, la posibilidad de utilizar lenguajes estándares capaces de servir para la descripción de hardware en cualquiera de los niveles de varios dominios diferentes, así como de simular las descripciones, ha llevado a la mayoría de las herramientas de SAN a la utilización de VHDL como lenguaje de descripción del comportamiento de partida. Esto permite además, la simulación de dicha descripción a nivel de comportamiento, para asegurar que es correcta. Este lenguaje tiene un extenso soporte comercial y permite construcciones condicionales, lazos, procedimientos, funciones y asignaciones concurrentes y secuenciales.

El compilador también es el encargado de realizar transformaciones de alto nivel que tratan de optimizar la representación interna. Estas transformaciones pueden ser la eliminación de código muerto, extracción de código invariante de los bucles, etc.

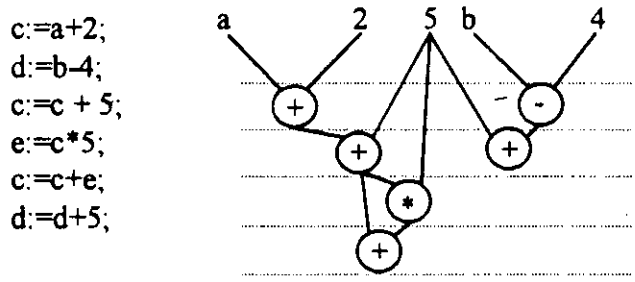


Figura 1.3 Ejemplo de GFD

En la figura 1.3 presentamos un ejemplo con un fragmento de una descripción de un circuito y el GFD correspondiente.

1.2.2. Planificación de Operaciones

Esta segunda tarea es, junto con la asignación de hardware, uno de los problemas clave de la SAN. Consiste en la distribución temporal de las operaciones del GFD, a partir de unas restricciones de área y tiempo dadas por el usuario, con un objetivo principal: obtener una

carga de trabajo equilibrada entre los distintos pasos de control, que permita el máximo aprovechamiento del hardware. El número total de pasos de control del algoritmo es lo que determinará la longitud de la memoria de la Unidad de Control -en el caso de un control microprogramado-, o el número de estados -para un control construido por una máquina de estados-. Por lo tanto, otro objetivo a minimizar por el planificador debe ser el número total de pasos de control del diseño final. Dentro de la planificación se pueden distinguir dos subtareas: la **selección del tiempo de ciclo** y la **asignación de operaciones a pasos de control**, que vamos a estudiar por separado a continuación.

1.2.2.1. Selección del tiempo de ciclo

Para realizar una planificación es necesario seleccionar el tiempo de ciclo que se va a utilizar en el diseño. Como veremos posteriormente, éste es uno de los factores que más van a influir en la obtención de la planificación óptima. En su selección hay que tener en cuenta los **tiempos muertos** de los operadores, es decir, el tiempo que transcurre desde que acaba de realizarse una operación hasta que se termina el tiempo de ciclo. Un tiempo de ciclo pequeño, en general, conduce a planificaciones donde los tiempos muertos de los operadores son pequeños, y por lo tanto, los tiempos de ejecución son menores. Sin embargo, el número de pasos de control aumenta al disminuir el tiempo de ciclo, puesto que se incrementa el número de etapas necesarias para realizar cada una de las operaciones. El incremento en el número de pasos de control produce a su vez un incremento en el área de la Unidad de Control.

Los tiempos muertos de los operadores para un determinado tiempo de ciclo dependen de los retardos de las UFs que los implementan. Por lo tanto es necesario tener en cuenta las UFs disponibles en la biblioteca en dicha selección.

Otro factor importante, que depende de la colocación de los módulos y de sus interconexiones en el layout final, es el retardo debido al interconexionado. Si éste no se tiene en cuenta, puede ocurrir que no se disponga de tiempo suficiente para la transmisión de todas las señales en un paso de control, lo cual conduciría a un mal funcionamiento del circuito. El **retardo de una**

interconexión depende, entre otros factores, de la tecnología que se vaya a utilizar, y de la longitud de dicha interconexión, desconocida hasta la generación del layout, y que será necesario estimar. Por lo tanto, en la selección del tiempo de ciclo, se deben tener en cuenta la biblioteca de módulos disponible, los retardos del interconexionado, y las restricciones del usuario en cuanto al área y tiempo de ejecución del circuito.

En la mayoría de los sistemas de SAN, la selección del tiempo de ciclo la realiza el diseñador, o se hace con un valor fijo. Esto puede dar lugar a diseños no óptimos, e incluso a diseños con una simulación eléctrica incorrecta, sobre todo si no se tiene en cuenta el retardo de las interconexiones.

En este trabajo presentaremos un método de selección del tiempo de ciclo y de estimación del retardo de interconexionado, que, además de permitir obtener planificaciones con tiempos de ejecución bastante aceptables, asegura el buen funcionamiento del circuito.

1.2.2.2. Asignación de operaciones a pasos de control

Una vez seleccionado el tiempo de ciclo el planificador debe asignar las operaciones a pasos de control tratando de minimizar su número. Podemos distinguir dos tipos de planificaciones:

- Si el número de módulos hardware de cada tipo (UFs y registros) ya se conoce antes de realizar la planificación, el planificador debe tratar de minimizar el número de pasos de control a partir de dicho hardware. Este tipo de planificaciones se presenta en sistemas donde el usuario restringe mucho el hardware disponible o en sistemas donde se realiza una preasignación de hardware antes de realizar la planificación.

- Si el número de módulos de cada tipo no es una restricción al planificador, también deberá equilibrar el paralelismo para poder realizar la implementación con el mínimo hardware posible, mediante la reutilización de UFs y registros en varios pasos de

control. En algunas planificaciones de este tipo, también se realiza simultáneamente una preasignación de hardware, es decir, se decide qué tipo de módulo va a realizar cada una de las operaciones.

En esta fase es necesario conocer el retardo de los módulos que van a realizar las operaciones, para estimar el número de pasos de control mínimo para cada operación.

Existen dos tipos de planificaciones muy fáciles de generar, si se asume un tiempo determinado de ejecución para cada operación, que proporcionan el mínimo número de etapas para ejecutar el algoritmo. Son las planificaciones **ASAP** ("As Soon As Possible") y **ALAP** ("As Late As Possible"). La planificación ASAP coloca cada nodo del GFD en la primera etapa donde pueda realizarse (cuando todos los operandos que necesita están disponibles). Un ejemplo se muestra en la figura 1.4a, donde se ha asumido que cada operación puede ejecutarse en una etapa de control. Por el contrario, la planificación ALAP coloca cada nodo en la última etapa donde pueda realizarse, sin incrementar el número de etapas dado por la planificación ASAP. La figura 1.4b nos muestra la planificación ALAP para el mismo GFD que la figura 1.4a. Estas planificaciones suelen utilizarse como base para generar otras nuevas, en general mejores en cuanto a la minimización del paralelismo.

Los dos objetivos de la planificación (la minimización del número de pasos de control y del paralelismo) son en muchos casos contrapuestos. Esto es debido a que, para reducir el paralelismo, puede ser necesario incrementar el número de pasos de control o viceversa. Sin embargo esto no es siempre así. Por ejemplo, en la figura 1.4 puede verse que la planificación a) sitúa en el mismo paso de control la realización de dos sumas, lo que obligaría a emplear dos sumadores diferentes, además de un restador y un multiplicador, o bien un sumador/restador, un multiplicador y un sumador. La planificación b) plantea un problema similar. En cambio, la c) permite emplear únicamente un sumador y un restador, además del multiplicador. En todos los casos, el número de pasos de control es el mínimo posible.

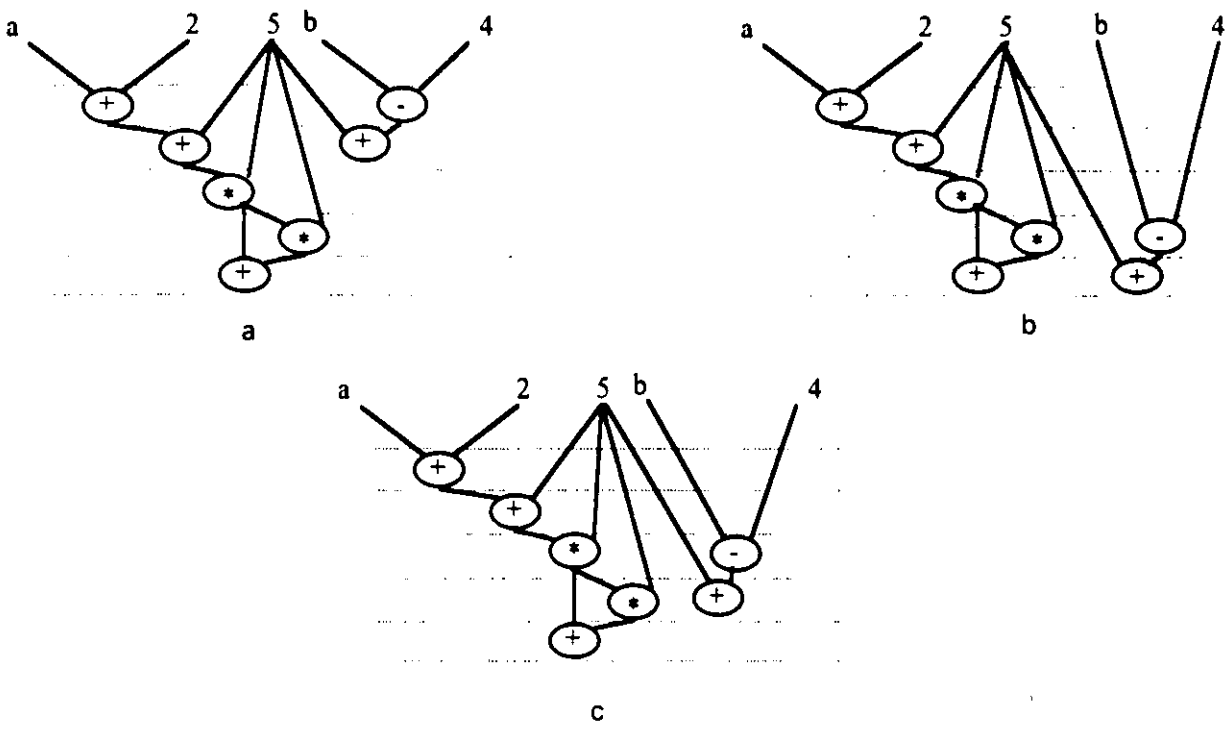


Figura 1.4. Tres planificaciones de un mismo GFD

Por lo tanto, en muchos casos, el planificador logra disminuir el paralelismo sin incrementar el número de pasos de control. La planificación óptima dependerá siempre de las restricciones impuestas por el usuario y de los objetivos principales que debe conseguir (de tiempo y área), y como hemos visto antes, de la biblioteca de módulos y de las características particulares del diseño.

La planificación de las operaciones en pasos de control es un tema bastante tratado ya por todos los sistemas de SAN, [PaKn87], [Paul88], [SSHF89], [WePa91], y por eso no profundizaremos más en ella.

1.2.3. Asignación del Hardware

En esta etapa se decide qué módulos hardware concretos van a realizar cada una de las operaciones del GFD proporcionado por el planificador, dónde se van a almacenar los resultados y qué tipo de interconexiones van a utilizarse para transmitir los datos. Las UFs

pueden escogerse entre las disponibles en una biblioteca, o bien generarlas automáticamente. Los elementos de almacenamiento serán registros y/o memorias. Las interconexiones entre unidades operacionales y unidades de almacenamiento pueden ser buses y/o multiplexores. El resultado es un conjunto de módulos e interconexiones que constituyen la **ruta de datos** del circuito. El objetivo principal de la asignación de hardware es conseguir un circuito con una ruta de datos que tenga un área final mínima.

Para un GFD planificado, se define un **espacio de diseño de asignación (EDA)** como aquel que contiene todos los posibles diseños que pueden implementar el comportamiento deseado, bajo una serie de restricciones de tiempo (el tiempo de ciclo y el número de etapas). Este EDA puede reducirse bajo ciertas restricciones, a un **espacio de diseño de asignación útil**.

El algoritmo de asignación debe recorrer este EDA útil, con el fin de encontrar una solución que minimice cierta **función de coste**. Esta función debe garantizar que los diseños que la minimicen sean aquellos que tienen un área menor. Por lo tanto, debe ser una **estimación del área del diseño**.

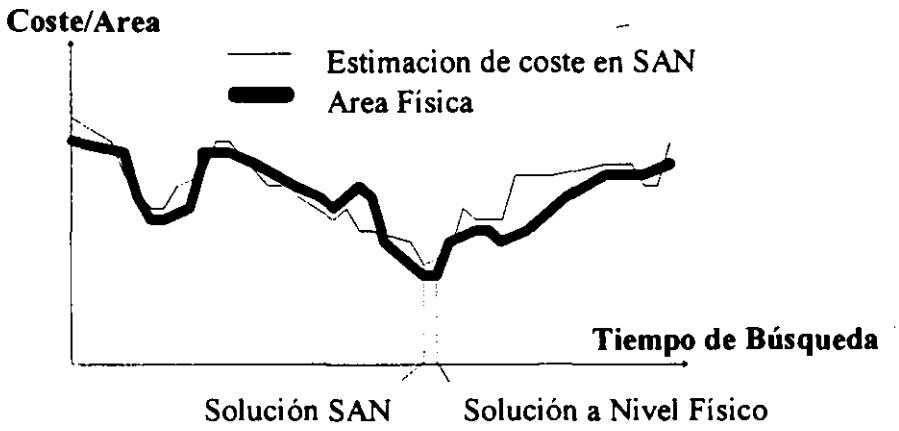


Figura 1.5 Area Física/Coste en SAN

En la figura 1.5 se muestra una representación de un proceso de exploración de un asignador de hardware genérico. En el eje de abcisas se han colocado los distintos diseños en el orden en el cual se encuentran (eje de tiempos), y en el eje de ordenadas el valor dado por la función de

coste. En la misma gráfica se ha incluído el área real, dada por una herramienta CAD, de los mismos diseños.

En una situación óptima, ambas funciones deberían ser iguales, o lo más parecidas posibles, de forma que la estimación del coste de cada diseño coincida con el área real del diseño. Es lo que se denomina una estimación **precisa**. Sin embargo, en la mayoría de los casos, es suficiente que ambas funciones tengan una forma similar. Así, un diseño con un valor mínimo de la función coste, tendrá también un área real mínima, y el mejor diseño obtenido por el algoritmo de asignación, será el diseño con área real mínima. En este caso diremos que la estimación es **fiel**.

Por lo tanto, durante un proceso de SAN se necesitan estimaciones fieles, que aseguren que el diseño seleccionado es el diseño con un área real mínima, y lo suficientemente precisas para conseguir diseños que estén dentro de unas especificaciones dadas por el usuario.

En la figura 1.6 vamos a comparar los resultados que se obtienen utilizando distintas funciones de coste. En la figura 1.6a se ha representado un proceso de obtención de sucesivos diseños (cada uno mejor que el anterior), mediante la exploración del EDA con nuestra herramienta de SAN, FIDIAS, para el ejemplo del filtro elíptico de 5° Orden [HLSB91]. Los algoritmos actuales de FIDIAS, con las estimaciones realizadas en este trabajo de investigación, utilizan una función de coste que contabiliza el área UFs, registros multiplexores e interconexiones. Por eso la gráfica 1.6a coincide con la del área real de los CIs.

En las otras tres gráficas se presentan los resultados generados por otras funciones de coste, que no tienen en cuenta todos los elementos del circuito. En la gráfica 1.6b se representan los costes de los distintos diseños si no se tiene en cuenta el área de las interconexiones. En la gráfica 1.6c sólo se considera el área de UFs y registros, y en la 1.6d sólo la de UFs.

Se puede observar que las gráficas tienen formas muy diferentes. Podemos deducir que la única función de coste fiel, que garantiza la obtención del diseño óptimo, es la que se muestra en la gráfica a, que tiene en cuenta todas las características físicas del diseño, e incluye tanto el

área de los módulos como de las interconexiones.

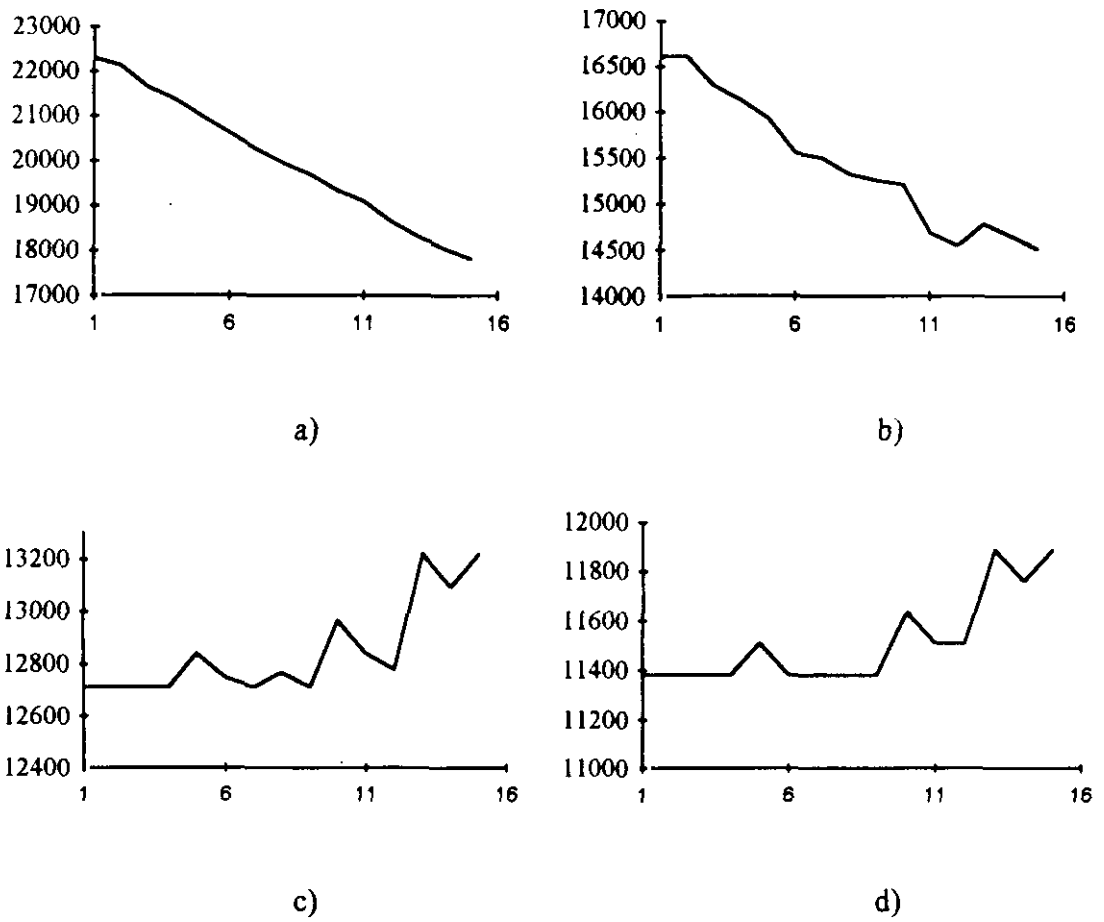


Figura 1.6 Resultados obtenidos por cuatro funciones de coste distintas

El resto de las funciones no sólo no son lo suficientemente precisas, sino que no permiten distinguir cuál es el diseño óptimo. Por ejemplo, si se usa la función de coste c o la d, la herramienta de SAN consideraría que uno de los diseños óptimos es el 1, cuando su área real es mayor que la del resto de los diseños explorados. De hecho, estas dos funciones dan por mejores precisamente los diseños que tienen mayor área, y por tanto son realmente peores. Por tanto, cualquier función de coste que se utilice en un asignador de hardware debe realizar una estimación global del área del circuito, donde se incluya el área de los módulos y de las interconexiones, y se tenga en cuenta la tecnología utilizada.

Sin embargo, en muchos sistemas de SAN, la asignación no se realiza de forma global. Como

ya se dijo anteriormente, dentro de la asignación de hardware se pueden encontrar dos subtareas: la preasignación de hardware y la asignación de instancias. En cada una de estas fases la información que se dispone del diseño es diferente, y por tanto la función de coste varía. Vamos a ver cuáles son las funciones de estas subtareas y cómo y dónde se necesitan hacer estimaciones.

1.2.3.1. Preasignación de Hardware

La subtarea de preasignación de hardware en algunos sistemas es previa a la planificación temporal y en otros sistemas se realiza conjuntamente con la asignación de hardware. Su función es determinar el número de registros y UFs de cada tipo que se van a utilizar para implementar el circuito. Sólo se decide el número y tipo de los módulos, no las operaciones del GFD que van a ejecutar cada uno de ellos en particular.

En muchos sistemas de SAN, el objetivo de esta tarea es la obtención de curvas área-rendimiento [JaPP87], [KuPa90a], [KuRa93], es decir, para cada una de las preasignaciones posibles estimar el área y el retardo del circuito final. De entre todas las posibles soluciones obtenidas, se elige aquella que cumple las restricciones y objetivos impuestas por el usuario (o la mejor de todas, en caso de que haya varias).

Para estimar el rendimiento de un circuito puede tenerse en cuenta el número de pasos de control y el tiempo de ciclo, datos conocidos para una determinada planificación. La estimación del área del circuito puede obtenerse calculando el área mínima de UFs y registros, también para una determinada planificación. Sin embargo, vimos en el apartado anterior que, para obtener soluciones fiables con las tecnologías actuales, es necesario tener en cuenta el área de las interconexiones y multiplexores, así como la tecnología en particular que se va a utilizar. La poca información que se dispone sobre el circuito en este momento dificulta mucho su estimación.

1.2.3.2. Asignación de instancias

Esta tarea tiene como función la asignación de variables a registros concretos y la asignación de operadores a UFs concretas. Dependiendo de cómo se realice la asignación, el número de buses, multiplexores e interconexiones punto a punto varía. Como ya se ha mencionado anteriormente, el área de interconexionado es un factor que influye notablemente en los resultados finales.

Esta subtarea puede partir de unas restricciones sobre el número de módulos de cada tipo disponibles, si se ha realizado una preasignación de hardware, o bien recibir una restricción en cuanto al máximo coste del diseño permitido. En cualquiera de los dos casos debe tratar de minimizar el coste total del diseño, que como ya hemos visto es una estimación del área física de éste.

El tipo de las estimaciones que se realizan durante la asignación de hardware varía dependiendo del tipo de algoritmo utilizado, como veremos a continuación. Existen tres tipos de algoritmos fundamentales:

- **Constructivos**: van asignando el hardware tratando de minimizar el coste final, y la solución que se obtiene es la mejor alcanzable por dicho algoritmo.
- **Iterativos**: a partir de una solución inicial se van generando nuevas soluciones realizando transformaciones que suponen una mejora de la anterior.
- **Iterativo-constructivos**: son una mezcla de los dos tipos anteriores. Crean una solución inicial por un método constructivo y van produciendo mejoras sobre ella.

Todos estos algoritmos tienen como finalidad minimizar el área total del CI. La mayoría de los sistemas SAN minimizan el hardware tratando de reutilizar al máximo los registros y las UFs de la ruta de datos. Pero esto no es suficiente. Por ejemplo, la reutilización de una UF lleva consigo la creación de un multiplexor a su entrada (o de un multiplexor con más entradas si ya existía uno), y una o más interconexiones. Las áreas de los módulos y de las interconexiones

dependen de la tecnología que se esté utilizando, de la colocación de los módulos en el circuito final, y de cómo se realice el interconexionado de éstos (tanto de las interconexiones internas de los módulos como de las externas). Por tanto depende de las características físicas del CI y de las herramientas de diseño, y es necesario estimarlas.

Para cada uno de estos tipos de algoritmos de asignación de hardware las necesidades y condiciones de las estimaciones de área son diferentes.

Algoritmos de tipo constructivo

Durante la asignación de hardware a cada uno de los nodos del GFD es necesario poder decidir entre varias opciones cuál conducirá a la obtención de área mínima. Por ejemplo, es necesario decidir entre reutilizar un determinado módulo o generar uno nuevo, teniendo en cuenta el consiguiente área de multiplexores e interconexiones para cada una de las posibles opciones. Si tenemos una UF cuya salida necesita ser almacenada y existe un registro disponible, es necesario elegir entre reutilizar el registro, al que será necesario añadir un multiplexor y una interconexión entre el multiplexor y el registro, o bien crear un nuevo registro. En ambos casos es necesario estimar el incremento en área que supondrá la creación de una interconexión, y de los módulos que se tengan que añadir, si bien todavía no se conoce el número total de módulos e interconexiones de la ruta de datos del diseño final. Esta estimación de área debe ser muy rápida, ya que se necesita realizar muchas veces durante la generación de un diseño, y debe ser lo suficiente fiel para permitir distinguir entre dos posibles decisiones cuál conducirá al mejor diseño final.

Algoritmos de tipo iterativo

Para los algoritmos de tipo iterativo se necesita poder distinguir entre dos o más diseños cuál es el que tiene el área mínima, y también escoger sólo aquellos

diseños que estén dentro de las especificaciones iniciales del sistema. Por lo tanto es necesario estimar el área total de la ruta de datos para cada uno de estos diseños, una vez que ya se conoce el número y tipo de módulos que lo van a formar y sus interconexiones. Esta estimación debe hacerse cada vez que se genere un nuevo diseño, y por lo tanto debe realizarse de forma rápida para que no incremente significativamente la complejidad del algoritmo.

Algoritmos iterativo-constructivo

Debido a que son un tipo mixto de los dos tipos anteriores, para este tipo de algoritmos es necesario realizar tanto la estimación del área final del circuito como del incremento en área que supone tomar una decisión durante el proceso de diseño.

Por lo tanto, sea cual sea el algoritmo de asignación de hardware utilizado, se deben realizar estimaciones de área rápidas, precisas y sobre todo fiables, que tengan en cuenta las influencias de las características físicas del circuito. Estas estimaciones unas veces se realizan sobre el diseño global y otras sobre partes del diseño. A lo largo de este trabajo de investigación veremos cómo realizar estas estimaciones de área en cualquiera de los casos anteriores.

1.2.4. Diseño de la Unidad de Control

La siguiente fase de un proceso de SAN es el diseño del controlador. Su función es generar la secuencia de control o microprograma (en el caso de control microprogramado) definitivos empleando la planificación y asignación realizadas anteriormente. Para ello es necesario seleccionar un modelo de Unidad de Control (UC) determinado, ya sea una máquina de estados finitos o algún modelo de unidad de control microprogramada. El área y retardo de la UC depende por una parte del diseño en particular (del número de etapas de control, el tiempo de ciclo y el número de módulos hardware utilizados por el asignador) y por otra de la

tecnología y tipo de UC que se utilice. Por lo tanto en esta fase también es necesario realizar estimaciones de área y retardo.

1.2.5. Control del Proceso de Diseño

Cada una de las subtareas anteriormente explicadas suele realizarse por medio de un módulo software independiente. Sin embargo, estas tareas son interdependientes y las decisiones que se toman en cada una de ellas afectan a los resultados obtenidos por las otras. Por eso, en algunos sistemas de SAN, se plantea la necesidad de un módulo adicional que controle el funcionamiento de todos los otros módulos y permita la interacción entre ellos. Esto posibilitará que todas las herramientas que participan en el diseño se orienten hacia la obtención de los objetivos dados por el usuario.

Este módulo debe decidir en qué momento se ejecuta cada una de las subtareas del proceso de síntesis, qué parámetros de funcionamiento interno utilizan, y cuáles son los parámetros globales del diseño.

También es el encargado de analizar cada uno de los diseños obtenidos, comparar los resultados con los objetivos deseados, y, en caso necesario, volver a diseñar con otros parámetros. De esta forma se consigue un sistema en lazo cerrado que permite la realimentación entre los distintos módulos operativos.

Este tipo de controlador necesita analizar los diseños obtenidos para ver si se cumplen los objetivos especificados, es decir, el coste en área y el rendimiento. También necesita realizar medidas sobre el retardo y área de los módulos hardware que deben ser tenidas en cuenta durante la planificación y la asignación de hardware a la hora de tomar decisiones.

1.3. Conexión con herramientas de Diseño

Todos las ventajas anteriormente descritas de una herramienta SAN se verían muy disminuidas

si no fuera posible generar automáticamente el layout del diseño a partir de la descripción estructural generada, ya que la interacción manual durante este último proceso incrementaría excesivamente el tiempo total de diseño. El módulo encargado de la asignación de hardware produce como salida la estructura definitiva del circuito digital, en forma de lista de componentes e interconexiones, o bien en forma gráfica. Por su parte, la salida del módulo encargado de la generación del controlador es la secuencia de estados o microprograma que, junto con el modelo de UC elegido, constituirá el hardware de control del sistema digital diseñado.

Si nos fijamos de nuevo en la figura 1.1, se puede observar que para obtener el layout final del circuito queda realizar la generación de módulos que lo componen y la colocación e interconexionado de estos, siempre cumpliendo unas reglas impuestas por la tecnología de fabricación. Aunque existen un conjunto amplio de estilos de diseño, que se presentan en el Apéndice A, esta última fase sólo tiene sentido realizarla en un estilo que tenga los mismos objetivos que la SAN. No tendría sentido automatizar el proceso de SAN para realizar luego el diseño con un estilo full-custom, que requiere una gran cantidad de tiempo, y además no es válido para diseños muy complejos. Tampoco sería lógico utilizar una tecnología basada en PLAs y ROM, donde se desaprovecha una gran cantidad de área, con lo cual todo el ahorro realizado por la herramienta de SAN se vería contrarrestado.

Por eso, los circuitos implementados con una herramienta de SAN debe fabricarse en un estilo de diseño basado en celdas estándar, macroceldas o arrays de puertas. Para estos estilos, las tareas de generación de módulos, colocación e interconexionado pueden realizarse por herramientas CAD de las que hemos hablado anteriormente, y que a partir de ahora también llamaremos herramientas de Síntesis de Bajo Nivel (**SBN**) o **herramientas de diseño**. Estas herramientas suelen ser específicas para cada uno de los estilos de diseño.

Por tanto, la generación del circuito final partiendo de la descripción de su comportamiento puede realizarse en dos fases:

- La primera utiliza una herramienta de SAN y obtiene como salida diseños a nivel

RTL.

- La segunda utiliza una herramienta CAD que, partiendo del nivel RTL generado por la herramienta de SAN, realiza la generación de módulos y su colocación e interconexionado final.

La conexión entre ambas herramientas puede hacerse automáticamente, con lo cual el proceso de diseño estaría totalmente automatizado.

La fase de generación de módulos puede ser independiente y previa a la Síntesis de Alto Nivel, formando una biblioteca de módulos, que estaría disponible para cualquier diseño, desde las primeras fases de la síntesis. Además, esto facilita la utilización de bibliotecas diferentes según la tecnología de fabricación, lo cual permite que el rápido avance de la tecnología no sea un factor crítico en la validez de la herramienta SAN.

Por otra parte, sabemos que el planificador y el asignador de hardware reciben como entrada una biblioteca de módulos (figura 1.2). Es posible almacenar datos en ella sobre cada uno de los distintos elementos hardware que la componen, que faciliten la estimación de su área y retardo. En [JRDK94] se demuestra experimentalmente la influencia de las variaciones del área y retardo de los módulos de la biblioteca en el layout final, y la imposibilidad de tratar los módulos como componentes con un tamaño y retardo constante. Como la colocación e interconexionado de módulos se realiza de forma global para todo el diseño RTL, la forma y tamaño de las componentes varía de unos diseños a otros. Por lo tanto, la biblioteca debe almacenar información física para una tecnología particular que permita estimar los valores de área y retardo de los módulos en el diseño final.

Por otra parte, se ha visto anteriormente que, debido a las influencias del nivel físico sobre el diseño, y para respetar las restricciones impuestas por el usuario en la SAN, así como para medir la calidad de los diseños obtenidos durante el proceso de síntesis, es necesario realizar una serie de estimaciones tanto del área de los módulos e interconexiones (**coste**) como sobre el retardo de estos, que nos dará una idea de la velocidad del diseño. De aquí surge la

necesidad de realizar la conexión de nuestra herramienta SAN con una herramienta de SBN, con la cual poder estudiar cómo trabajan las herramientas de diseño y poder evaluar nuestras predicciones.

El diseño físico previo a la SAN de cada uno de los módulos de la biblioteca permite almacenar en ésta información física sobre aquellos, como puede ser el área de sus celdas estándar (o macroceldas), el retardo, el consumo, las interconexiones internas, etc. Toda esta información la pueden utilizar los módulos de planificación, asignación de hardware, etc. para tomar sus decisiones. De esta forma tanto las medidas parciales como las finales sobre los diseños serán mucho más realistas.

La conexión entre una herramienta de SAN y otra de SBN se puede realizar traduciendo el camino de datos y la Unidad de Control que proporciona la herramienta SAN, a un lenguaje estándar como EDIF (Electrical Design Interchange Format) o VHDL, que son aceptados como entrada por la mayoría de las herramientas de SBN.

En definitiva, podemos concluir que el uso conjunto de herramientas automáticas de síntesis complementarias (herramientas de síntesis de alto nivel, por un lado, y de bajo nivel por otro) puede permitir automatizar el proceso de diseño prácticamente en su totalidad, y acceder a todas las ventajas que esta automatización conlleva. La automatización total debe estar controlada por el módulo controlador de diseño, que decidirá qué diseño es el que se va a sintetizar y una vez obtenido el layout comprobará que se cumplen las restricciones impuestas por el usuario

1.4.Conclusiones

A lo largo de este capítulo se ha justificado la necesidad de realizar estimaciones de área y retardo durante la SAN, teniendo en cuenta las influencias de las características físicas del circuito integrado.

En primer lugar hemos visto que para realizar la selección del tiempo de ciclo, se deben tener en cuenta la biblioteca de módulos disponible, los retardos del interconexionado, y las restricciones del usuario en cuanto al área y tiempo de ejecución del circuito, así como realizar un estudio global del GFD. El retardo del interconexionado es un dato que no se conoce hasta que no se ha generado el layout, y por tanto es necesario estimarlo.

En segundo lugar hemos demostrado como cualquier función de coste que se utilice en un asignador de hardware debe ser una aproximación al área real del circuito. Este área es la suma del área de los módulos y de las interconexiones, que a su vez dependen de la tecnología que se esté utilizando, de la colocación de los módulos en el circuito final, y de cómo se realice el interconexionado de éstos (tanto de las interconexiones internas de los módulos como de las externas). Por tanto depende de las características físicas del CI y de las herramientas y tecnologías de diseño, y es necesario estimarlas.

También se observó que durante la asignación de hardware se necesitan realizar diversos tipos de estimaciones, dependiendo del tipo de algoritmo utilizado:

- Para los algoritmos de tipo constructivo era necesario estimar qué decisión entre todas las posibles conduciría a un diseño con un área mínima. Estas estimaciones se realizan cuando todavía no se conocen todos los elementos que integran el circuito.

- Para los algoritmos iterativos era necesario estimar si un diseño es mejor o peor que otro. Estas estimaciones se realizan cuando se conocen todos los elementos que integran ambos diseños.

En todos los casos las estimaciones se necesitan realizar muchas veces durante un proceso de síntesis, y por tanto deben ser muy rápidas.

Por lo tanto, en SAN se necesita un método de estimación del área muy **rápido** (tanto del área de los módulos como de las interconexiones), que más que permitirnos obtener un valor exacto de área final del diseño, tenga suficiente **fidelidad** para:

- Permitir tomar decisiones sobre si un diseño es mejor que otro.
- Decidir si un diseño está dentro de unas especificaciones dadas por el usuario
- Permitir seleccionar entre dos posibles decisiones la que conduce a un área mínima

También se planteó la necesidad de realizar la conexión de la herramienta de SAN con una herramienta de diseño con dos objetivos fundamentales:

- Realizar una automatización total del proceso de diseño.
- Estudiar las influencias del nivel físico en las decisiones que se toman durante la SAN, y recopilar la información necesaria para poderlas tener en cuenta durante todas las fases de la síntesis.

Sin embargo, la mayoría de los sistemas de SAN no realizan todas las estimaciones anteriormente presentadas. No tenemos información sobre ningún sistema que realice estimaciones del retardo de las interconexiones para obtener el tiempo de ciclo y asegurar el correcto funcionamiento del circuito. Existen algunos sistemas que tienen en cuenta de alguna forma el área de las interconexiones, y otros que estiman el área de los módulos en el diseño final, pero en ningún caso estas estimaciones cumplen todos los requisitos anteriores.

El objetivo principal de este trabajo de investigación es estudiar las influencias de las características físicas del CI sobre el área y retardo del los diseños, y diseñar técnicas de estimación rápidas, precisas y fieles para todas las fases de la SAN. Se presentará un método de estimación de área durante la asignación de hardware válido para todos y cada uno de los tipos de algoritmos presentados. Como estas influencias dependen de la tecnología de diseño utilizada, y el estudio para todas ellas es un trabajo de una extensión excesiva, se particulizará el estudio para celdas estándar. Sin embargo, muchas de las ideas propuestas pueden utilizarse para macroceldas y arrays de puertas.

Estas estimaciones pueden utilizarse también durante la preasignación de hardware y la generación del control. También se verá cómo es posible estimar el retardo de las interconexiones. Además, se realizará la conexión de una herramienta de SAN con una

herramienta CAD, así como la automatización del proceso de diseño completo.

Pero antes de presentar estos métodos de estimación, en el capítulo 2 se realiza un repaso de los principales sistemas de que elaboran algún tipo de estimación del área de interconexionado. Veremos sus ventajas y sus inconvenientes, y la necesidad de realizar nuevas estimaciones más precisas.y rápidas.

En el capítulo 3 se estudiará el sistema FIDIAS, que es un sistema de SAN dentro del cual se integra este trabajo de investigación. Se verá que en este sistema, como en todos, también es necesario realizar estimaciones, y cómo y dónde se deben integrar éstas.

En el capítulo 4 se realizará un estudio de la metodología de trabajo de las herramientas de CAD, y de las características físicas de los CIs. A partir de este estudio, se realizarán estimaciones del área de las interconexiones y de los módulos del CI, y se integrarán dentro del sistema FIDIAS. Veremos también que estas estimaciones se pueden integrar en cualquiera de los tipos de algoritmos de asignación vistos anteriormente.

En los capítulos 5 y 6, y a partir de las estimaciones del área de las interconexiones del capítulo 4, veremos la forma de calcular el retardo de éstas. Se realizará un estudio completo sobre el retardo del interconexionado, y sobre cómo se puede incluir éste en el cálculo del tiempo de ciclo. También veremos un algoritmo de estimación del tiempo de ciclo que tiene en cuenta la biblioteca de módulos, las restricciones del usuario y que estudia de forma global el GFD.

Capítulo 2

Técnicas de Estimación de las Características Físicas

2.1. Introducción

Muy pocos sistemas de SAN tienen en cuenta las características del nivel físico. Sin embargo, con la tecnología VLSI, una gran parte del área del CI se consume en cableado y el retardo de las interconexiones es muy significativo. Ya McFarland [McFa87] apuntó la gran importancia de tener en cuenta las características físicas del diseño en SAN. Posteriormente Parker y colaboradores [PaGH91] pusieron de manifiesto las variaciones que dichas características producen en las curvas área-rendimiento de los diseños producidos.

Por otra parte, en el capítulo anterior vimos la necesidad de realizar estimaciones del área de los módulos y de las interconexiones en varias de las distintas fases de la SAN. También se planteó la importancia de considerar el retardo de las interconexiones durante la planificación de operaciones, a pesar de lo cual la mayoría de los sistemas sólo tienen en cuenta el de los módulos. La estimación de los retardos de las interconexiones sólo puede realizarse a partir de las longitudes de éstas, por lo cual sólo los sistemas que estiman la longitud de las interconexiones pueden calcular su retardo.

A lo largo de este capítulo vamos a revisar los sistemas que de alguna forman tienen en cuenta la influencia de las características físicas en el área de las distintas partes que integran el circuito. Entre todos estos sistemas, señalaremos los que estiman el retardo de las interconexiones.

Algunos de los sistemas de SAN sólo pueden generar las estimaciones de área a nivel RTL, cuando ya se conocen el número de módulos y de interconexiones. Otros permiten realizar algunas estimaciones durante la generación de un diseño, cuando la información sobre éste aún no es completa. Pero ninguna de estas estimaciones sobre área cumple los requisitos que vimos en el capítulo anterior: baja complejidad, fidelidad y precisión.

La mayoría de los métodos capaces de predecir el área de un diseño con suficiente exactitud, teniendo en cuenta el área de interconexionado, necesitan la información completa sobre el diseño y generan la estimación para un tipo de tecnología determinada. Estos métodos suelen estar integrados en las herramientas de SBN, y predicen el área de un circuito dada su descripción como un conjunto de celdas estándar (algunos permiten incluir macroceldas) o arrays de puertas y sus interconexiones, con el fin de disminuir el número de iteraciones necesarias para realizar un *floorplanning* (ver Apéndice B) correcto. Estas estimaciones suelen ser bastante precisas, con un error inferior al 10%, y la herramienta de SBN sólo necesita elaborarlas una vez. Pero como la información que necesitan manejar estos algoritmos de estimación es muy grande, la complejidad suele ser muy elevada, y no es conveniente integrarlos dentro de una herramienta de SAN, donde es necesario generar las estimaciones un gran número de veces. Sin embargo, es interesante repasar algunos de estos métodos para ver qué posibilidades presentan.

A lo largo de este capítulo vamos a exponer las principales técnicas de estimación de área clasificándolas en dos grupos fundamentales: las que se elaboran durante el proceso de SAN y las que se realizan a nivel de celdas estándar y arrays de puertas. Las primeras las denominaremos "Técnicas de Estimación durante la SAN" y las segundas "Técnicas de Estimación de Bajo Nivel". En el primer grupo presentaremos también los principales sistemas de SAN que, de alguna forma, tienen en cuenta las interconexiones en la optimización del diseño.

Como veremos, en general, las técnicas pertenecientes al primer grupo no son lo suficientemente fieles, mientras que las del segundo grupo son muy precisas, pero tienen una

complejidad demasiado elevada para ser utilizadas dentro de un proceso de SAN. Por eso surge la necesidad de obtener nuevas técnicas de estimación de área que reúnan las propiedades de ambos tipos: la precisión de las segundas y la baja complejidad de las primeras.

Nota. A lo largo de este capítulo utilizaremos las palabras conexión e interconexión como sinónimos. Una red se refiere a una conexión que una más de dos elementos. También serán sinonimos área de interconexionado y área de cableado.

2.2. Estimaciones de área durante un proceso de SAN

Todos los sistemas de SAN necesitan realizar estimaciones sobre la bondad de un circuito, pero sólo algunos de ellos tienen en cuenta la influencia de las interconexiones en los resultados finales. Como vimos en el capítulo anterior, en todos ellos existe una cierta función de coste que minimizar, y dependiendo de los factores que tenga en cuenta dicha función de coste, los diseños que la minimicen serán o no realmente óptimos. Esta función de coste debe ser una estimación lo más precisa posible del área de los módulos y de las interconexiones.

Por otra parte, también se planteó en el capítulo anterior, la limitación que supone tratar los módulos de la biblioteca como elementos con un retardo y área constantes. Por tanto, es necesario revisar, además del coste de las interconexiones, el coste de los módulos.

A continuación vamos a realizar una clasificación de los principales sistemas de SAN que incluyen el coste de las interconexiones del diseño en la función de coste. Posteriormente, y para cada uno de los sistemas que estudiemos, veremos si utilizan alguna estimación del coste de los módulos.

En primer lugar, existen sistemas que tratan las interconexiones exclusivamente desde el punto de vista del **número y/o coste de multiplexores** que se necesitan para construirlas. Estos sistemas minimizan el número de interconexiones del circuito con el fin de minimizar el

número de multiplexores, pero no realizan ninguna estimación del coste del cableado de las interconexiones.

Otros sistemas, además del número de multiplexores, tienen en cuenta el **número de interconexiones** del circuito, y tratan de minimizar ambos. Suelen ser sistemas que no obtienen una minimización global del área del circuito, ya que por una parte minimizan el número de UFs, por otra el de registros, etc.

En tercer lugar, existen sistemas que utilizan un **valor constante** para el coste de una interconexión; en algunos casos este coste es elegido por el diseñador, y en otros es un valor obtenido empíricamente para un estilo de diseño determinado.

Por último, existen sistemas que realizan un ***flooplanning*** (ver Apéndice B) conjuntamente o previo a la SAN, mediante el cual es posible estimar el área de interconexionado.

A continuación vamos a ver ejemplos de cada uno de estos tipos de sistemas, y veremos los pros y contras de las técnicas utilizadas.

2.2.1. Sistemas con minimización del coste de multiplexores

2.2.1.1. Sistema Olympus

Es un sistema de SAN desarrollado en la Universidad de Stanford bajo la dirección de Giovanni De Micheli. Está orientado a circuitos digitales síncronos de propósito general [MKMT90] [KuMi90a] [KuMi90b], [KuMi90c], con atención especial a los requerimientos de los diseños ASIC.

En la figura 2.1 se ofrece el diagrama de bloques de este sistema. Se puede observar que una de las novedades que presenta es la utilización de una herramienta de síntesis lógica para optimizar los circuitos. Esta también se utiliza para computar la información de área y retardo de los diseños y de las distintas partes de éstos, y esta información es realimentada al

Asignador de Hardware. Además, cada operador del GFD puede convertirse en un módulo de la biblioteca previamente diseñado, o bien implementarse en términos de expresiones lógicas que serán luego optimizadas por la herramienta de síntesis lógica.

La otra novedad importante de este sistema es que proporciona un interfaz para herramientas estándares de diseño físico.

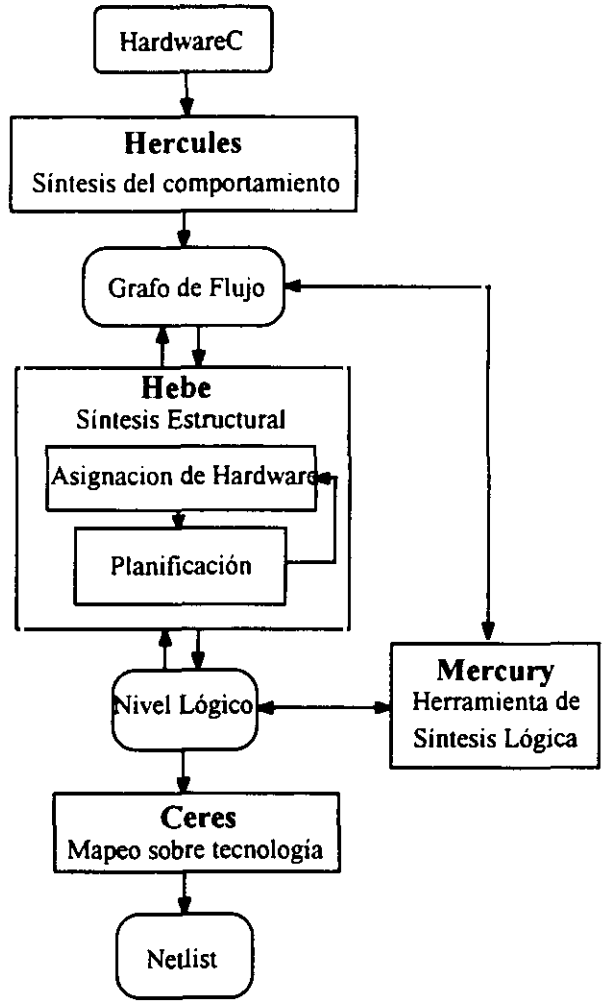


Figura 2.1 Sistema Olympus

El Asignador de Hardware utiliza heurísticas de búsqueda con unas funciones de coste que estiman el área y el retardo de interconexiones y módulos de la siguiente forma:

- Para computar el coste de una interconexión sólo se tiene en cuenta los multiplexores utilizados para realizarla, pero no estima el área de cableado necesario.

- Los módulos utilizados suelen tener una complejidad elevada, y su coste es el área total debida a registros, multiplexores y UFs que lo conforman, pero no se tiene constancia de que se hayan considerado el área de layout ni del cableado interno de éstos.

Una vez terminado el diseño lógico del circuito global, éste se transforma en una representación estructural definida en términos de celdas de una biblioteca predefinida para implementación semicustom. Esto se realiza de forma óptima, tratando de escoger las celdas que minimicen el área y el retardo para el diseño en particular.

La combinación de síntesis de alto nivel y síntesis lógica permite la generación de módulos de propósito particular para cada diseño, sin tener que depender de una biblioteca determinada, lo cual favorece la obtención de diseños óptimos. Además, la utilización de síntesis lógica para realizar estimaciones mejora la calidad de los diseños, y evita tomar decisiones erróneas a alto nivel debidas a la falta de datos sobre las características de los módulos.

Sin embargo, este sistema no explota todas las posibilidades que la generación del diseño sobre una tecnología determinada le ofrece, puesto que no utiliza la información del layout para generar estimaciones sobre el área del interconexionado. Cada una de las fases, la SAN y la SBN, realizan las optimizaciones por separado, pero no se interrelacionan convenientemente. El hecho de no tener en cuenta el área de cableado en las funciones de coste puede conducir a diseños que no sean realmente óptimos.

2.2.1.2. Sistema Elf

El sistema Elf [GiBK85] ha sido desarrollado por Girczyc en la Carleton University de Otawa. Este sistema otorga una atención especial a la minimización de las interconexiones y, con este fin, se pueden utilizar múltiples caminos para generarlas. Por ejemplo pueden realizarse a

través de UFs, sumando 0 o multiplicando por 1, o bien reutilizando otras interconexiones ya generadas anteriormente, etc. De todas las posibilidades se elige la más barata. Además, el modelo de interconexión [LyEG90] permite un número arbitrario de niveles de cascada de multiplexores y buses.

El coste de un bus depende de la anchura y del incremento en área al crear un nuevo camino, que es una función de la longitud de la interconexión, de las celdas necesarias para realizar conversiones, si se necesitaron, y del incremento en el número de entradas del multiplexor. La longitud de una interconexión se estima como el número de elementos que debe atravesar (muxes, UFs, etc.), pero no se tiene en cuenta el área de cableado.

Para un camino que ya existe, el coste de reutilizarlo es una función del número de caminos posibles que había antes de realizar esta interconexión ($N_{caminos\_antes}$) y del número de caminos que quedan para otras interconexiones después ($N_{caminos\_despues}$). Si una interconexión se ve afectada por el uso de un camino por otra interconexión, el coste probable que supone la utilización de dicho camino viene dado por:

$$Coste_{probable} = Coste_{nuevo\_camino} * \left(\frac{1}{N_{caminos\_antes}} - \frac{1}{N_{caminos\_despues}}\right)$$

donde $Coste_{nuevo\_camino}$ es el coste de crear el nuevo camino de la interconexión afectada. Si existe más de una interconexión afectada, el coste de reutilizar un camino existente es la suma de los costes probables de las transferencias cuyos caminos posibles son inutilizados por dicha asignación. El algoritmo trata de elegir caminos que inutilicen el menor número de caminos para posteriores conexiones.

Para tener en cuenta estas consideraciones, la lista de interconexiones se ordena según las prioridades, tratando de colocar primero aquellas interconexiones que tienen menos posibilidades de realizarse sin incrementar el hardware. Las prioridades vienen dadas por:

- La necesidad de hacer una transferencia de datos en el paso de control que se está tratando.

- El número de caminos existentes para realizarla. Cuantos más caminos haya menos prioridad tiene la interconexión considerada, puesto que existe más probabilidad de reusar un camino existente.

- La longitud media de estos caminos, medida como el número de elementos (redes, celdas, etc.) que debe atravesar la interconexión. La prioridad es directamente proporcional a este número.

- El grado de independencia de estos caminos. Cuanto más independientes sean estos caminos, menos prioridad tiene.

Una vez generada la lista ordenada de interconexiones, se van asignando por orden de prioridad. Si no es posible realizar una interconexión utilizando el hardware mínimo y se puede dejar ese nodo para un paso de control posterior, se pasa al siguiente de la lista. Tanto las interconexiones realizadas como las no realizadas se devuelven al asignador de UFs y registros, que decide qué asignaciones de la lista son válidas y cuáles no.

A continuación, se asigna el almacenamiento para las salidas de cada operación del paso de control, tratando de minimizar el número de interconexiones entre los registros y las UFs.

Una vez creadas todas las interconexiones del diseño, se reduce el área de interconexionado mezclando multiplexores y buses para generar interconexiones multi-nivel. De esta forma se supone que el sistema consigue diseños con un área y retardo de interconexionado mínima.

De todo lo dicho, se puede deducir que este sistema es uno de los que más se preocupan de minimizar el área de conexionado, permitiendo una amplia gama de posibilidades para implementarlas. Sin embargo, esta minimización del conexionado se hace al nivel RTL y no tiene en cuenta el área real de cableado, sino sólo el número de distintos módulos que atraviesan las interconexiones. Esta metodología no conduce, en la mayoría de los casos, a diseños óptimos, como ya vimos en el capítulo anterior.

Por otra parte, el coste de los módulos tampoco tiene en cuenta las características físicas del circuito. Por ejemplo, el coste de los multiplexores se mide en función del número de entradas que tengan, pero no tiene en cuenta el área de layout. Además, se plantea el problema de la optimización del diseño por partes. De esta forma se pueden obtener diseños donde, por ejemplo, el número y coste de multiplexores sea mínimo, pero el diseño globalmente no tenga un coste mínimo.

2.2.1.3. Sistema HIS

El sistema HIS [BeCP92], [CBHP91], [Camp88], [Camp91] es un proyecto de SAN desarrollado conjuntamente en diversos centros de IBM, bajo la dirección de Raul Camposano. Está orientado a microprocesadores o a diseños donde domina el control. En este tipo de sistemas las decisiones más importantes del diseño se dan en el tratamiento de los bloques condicionales y no en el posible paralelismo.

El objetivo de este sistema es conseguir el máximo rendimiento con la mínima cantidad de hardware, a partir de una restricción sobre el máximo número de UFs introducido por el usuario. Primero se realiza una asignación inicial de la ruta de datos, y a continuación se optimiza, comprimiendo registros, unidades funcionales y multiplexores mediante un algoritmo de coloreado de grafos.

Tanto para la comprobación de las restricciones, como para la optimización de la ruta de datos, necesita realizar estimaciones de tamaños y retardos. Con este fin, el área de los módulos se estima en función del área de las celdas estándar que los componen, y el retardo se computa para cada celda, teniendo en cuenta los niveles de carga y fanout.

Este sistema no realiza ninguna estimación del área ni del retardo de las interconexiones. Además, cuando estima el área de los módulos no tiene en cuenta la influencia de las interconexiones internas. Por otra parte, en la optimización de la ruta de datos no considera el

número de interconexiones, sólo el de multiplexores. Por todo esto no se puede garantizar que los diseños obtenidos sean los que tienen un área real mínima.

2.2.2. Sistemas con minimización del número de interconexiones

2.2.2.1. Sistema HAL

HAL, al igual que Elf, ha sido desarrollado en la Carleton University, Otawa. Emplea técnicas de programación orientada a objetos, como ha descrito P. E. Paulin en [PaKG86], [PaKn87], [Paul88], [PaKn89a], [PaKn89b] y [PaKn89c].

En la figura 2.2 se muestran las etapas del sistema.

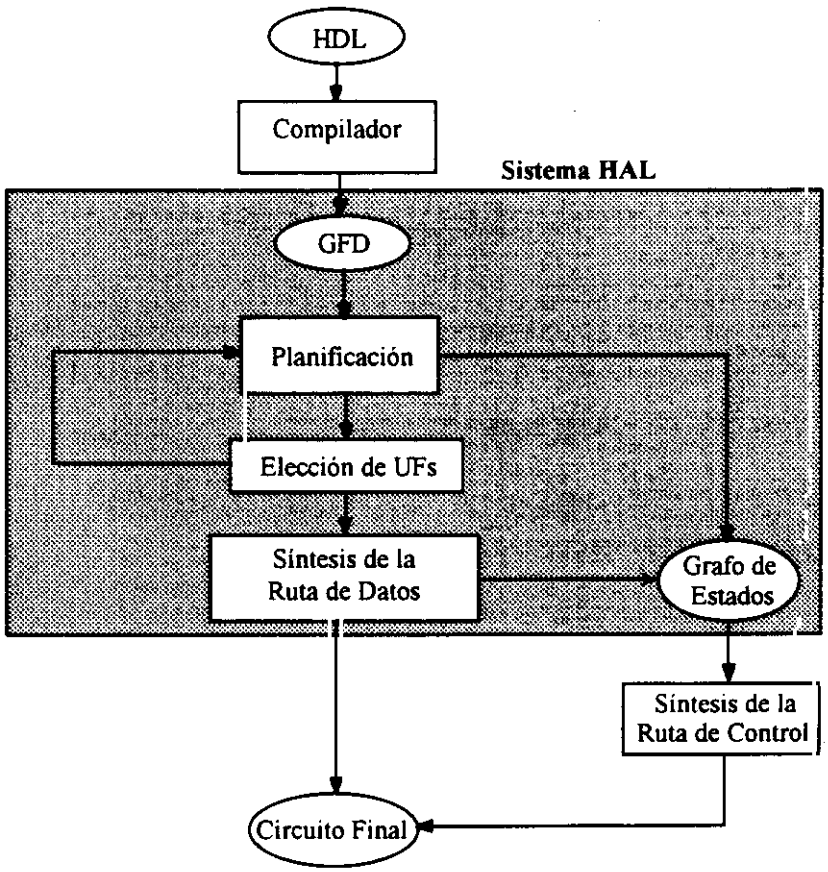


Figura 2.2 Etapas del Sistema HAL

Después de la asignación de UFs y registros, se reduce el número de estos últimos mediante un algoritmo de particionamiento de grafos, dando prioridad a las particiones que, además de disminuir el número de registros, suponen una minimización del número de interconexiones [PaKn89b].

El modelo de interconexión que emplea está orientado a multiplexores. Éstos se mezclan posteriormente para formar buses, mediante una técnica análoga a la de agrupación de registros.

Como vemos, este sistema minimiza separadamente el número de UFs, el número de registros y el número de muxes, tratando de reducir el numero de interconexiones. Sólo considera y minimiza el número de cada tipo de elemento, no su área. No tiene en cuenta el área de los módulos ni de las interconexiones, ni el peso que pueden tener cada uno de ellos en el área final del CI. Por lo tanto no es probable que los diseños obtenidos tengan un área mínima.

2.2.2.2. Sistema PARTHENON

Es una herramienta de SAN desarrollada en NTT Comunications and Information Processing Laboratories. Parte de una descripción del comportamiento y produce automáticamente circuitos lógicos [Naka87], [NaOg87],[NONN90].

La distintas partes del proceso global se presentan en la figura 2.3. Se puede observar que es un sistema muy completo, porque la herramienta de SAN está conectada a una herramienta de Síntesis Lógica y a otra de diseño de Bajo Nivel. SFLEXP genera los circuitos en forma de pseudo-celdas (puertas NOT, OR y AND) que no contienen ninguna información sobre tecnología, y optimiza el diseño a nivel lógico. Una vez terminada la síntesis del circuito, OPTMAP realiza el diseño sobre celdas reales, optimiza el circuito y hace los cambios necesarios para que se cumplan las restricciones y requerimientos. Además de las restricciones externas, también realiza un análisis para verificar que se cumplan las restricciones eléctricas de las celdas del diseño, tanto de carga como de tiempo. La información necesaria para

realizar este análisis la recibe de la biblioteca de celdas. Si alguna de las restricciones no se cumple, trata de modificar las componentes o añade otras nuevas hasta conseguir los objetivos.

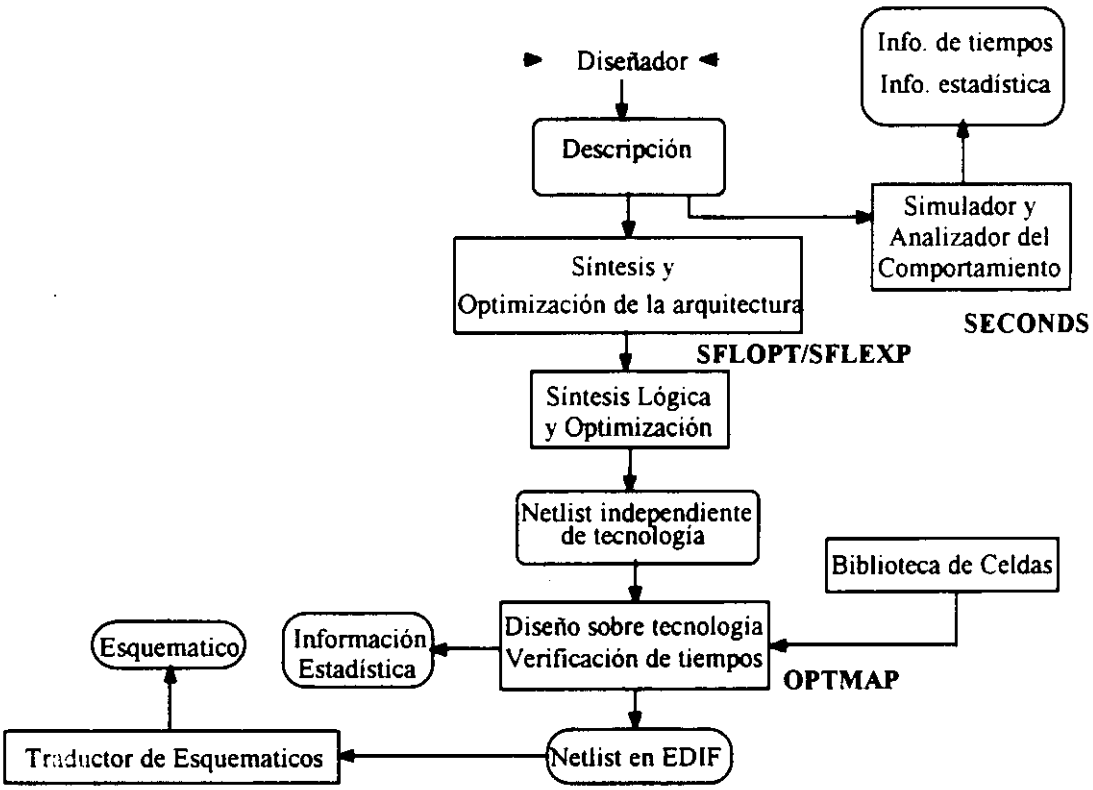


Figura 2.3 El Sistema Parthenon

A pesar de las posibilidades que presenta la conexión con la herramienta de Síntesis Lógica y con la de Bajo Nivel, este sistema de SAN no considera el área de interconexiones durante el proceso de diseño. Se minimiza el número de módulos e interconexiones por separado, pero la información sobre la tecnología y sobre las características físicas del circuito no se tiene en cuenta hasta las últimas fases del diseño, cuando se está generando el layout. Por lo tanto, no se puede decir con certeza que los resultados cumplan los objetivos iniciales. Incluso es necesario terminar todo el proceso, hasta el diseño para una tecnología determinada, para saber si se han cumplido las restricciones del usuario. En caso de que no se cumplan, se puede realimentar el proceso de nuevo, pero esto llevaría mucho tiempo de diseño.

2.2.3. Sistemas con coste de interconexión fijo

2.2.3.1. Sistema Chippe

Este sistema de síntesis fue desarrollado en la Universidad de Illinois por B. Pangrle y F. Brewer bajo la dirección de D. Gajski.

El sistema completo consta de los tres módulos: Chippe,[BrGa87],[Brew88], Slicer y Splicer, [PaGa86] [PaGa87] [Pang88], como se presenta en la figura 2.4.

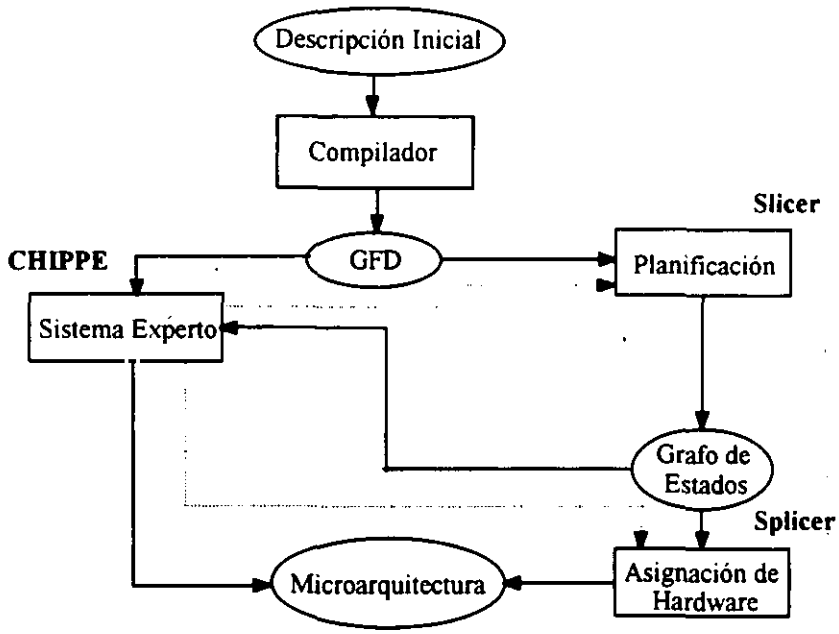


Figura 2.4 Sistema Chippe

Una de las principales novedades de este sistema, es la utilización de un sistema de control que permite la realimentación del proceso de diseño, si las restricciones y objetivos impuestos por el usuario no se han cumplido. Para esto utiliza una serie de funciones de coste que miden la bondad de los diseños obtenidos.

El algoritmo que emplea el asignador de hardware para recorrer el espacio de diseño es de tipo ramificar y acotar. También proporciona un conjunto de funciones de coste

seleccionables, que permiten acotar las ramas de diseño accesibles, y acelerar el proceso de búsqueda.

Las funciones de coste están basadas en posibles restricciones en el número máximo de recursos de cada tipo, en su coste, etc., y relaciones un tanto arbitrarias entre algunas de dichas restricciones. El coste de un circuito determinado se evalúa por el número de elementos de cada tipo, incluyendo el número de interconexiones, modulados mediante pesos elegidos libremente por el usuario, que asignan mayor o menor importancia relativa a cada tipo de elemento concreto. Los problemas de este método son que la obtención de resultados realistas depende de la habilidad y experiencia del usuario para fijar el valor de los parámetros, y que para un mismo diseño puede haber distintos costes dependiendo de la función elegida.

2.2.3.2. Sistema de la Universidad de California, Berkeley

Este sistema ha sido desarrollado por Devadas y Newton en la Universidad de California, Berkeley.

Trata el problema de planificación y asignación conjuntamente como un problema de colocación de operaciones en dos dimensiones: el espacio y el tiempo. La asignación de UFs, registros e interconexiones se hace simultáneamente. Utiliza un algoritmo de tipo *simulated-annealing* para resolver el problema, con una función de coste C que tiene en cuenta el coste del hardware y el tiempo de ejecución [DeNe89] y se define como:

$$C = \alpha * (n^{\circ}\,\text{ALUs}) + \beta * (\text{tiempo\_ejecución}) + \chi * (n^{\circ}\,\text{registros}) + \delta * (n^{\circ}\,\text{buses})$$

Las ALUs son unidades que pueden realizar diversas operaciones aritméticas y lógicas. α, χ y δ son parámetros de área y reflejan el área de layout de cada uno de los módulos. Para registros y ALUs se toman de la biblioteca de módulos. El área de los buses se calcula a partir de estimaciones empíricas: dado un estilo de layout se evalúa el incremento en área de interconexionado debido a la adición de registros e interconexiones; a partir de estas medidas se obtienen valores del área de buses en función del número de interconexiones y módulos.

Por otra parte, β es un parámetro del tiempo de ejecución y mide los objetivos de área/tiempo que deben conseguirse. Un β grande implica que se da más importancia a la minimización del tiempo que a la del área, y el resultado será una ruta de datos con menores tiempos de ejecución que para un β pequeño.

Este método puede presentar problemas en cuanto a la estimación del área de los módulos. Un módulo con un número elevado de interconexiones internas no ocupa la misma área cuando se realiza su layout aislado que cuando se realiza junto con otros módulos a los que está conectado. La existencia de interconexiones externas influye en la colocación relativa de las celdas del módulo y por tanto en el área de las interconexiones internas. Por otra parte, precisa la estimación del área de los buses para un número elevado de posibilidades, ya que el área de éstos depende mucho del número y tamaño de los módulos del circuito final. La precisión de los resultados finales depende de si se han obtenido previamente los valores del tamaño de los buses para el número de interconexiones y de módulos que tendrá el circuito final.

2.2.3.3. Sistema SAW

El System Architect's Workbench [TLWN90] (SAW) ha sido desarrollado en la Universidad Carnegie-Mellon.

En este sistema, el modelo de ruta de datos incluye tanto buses como multiplexores. El algoritmo de asignación emplea un criterio global para la elección del elemento siguiente. Dicho criterio se basa en el cálculo de los costes de asignación de los distintos elementos del grafo a componentes hardware. Este coste se calcula en cada paso del proceso de asignación, para todas las opciones posibles. Luego se calcula para cada elemento del grafo la diferencia entre las dos asignaciones menos costosas, se elige el elemento del grafo con una diferencia mayor entre estos dos costes, y se asigna al componente hardware de menor coste. Con esto se intenta reducir al mínimo el incremento potencial del coste de las asignaciones en cada paso futuro. Este criterio había sido propuesto anteriormente por McFarland, y se conoce como criterio **min-max** [McFa83].

Para el cálculo de los costes de las diferentes opciones, se dispone de una tabla de costes añadidos, donde se especifica el incremento de añadir un nuevo operador a una UF que realizaba otras operaciones.

Posteriormente se resintetizan las interconexiones utilizando un algoritmo de particionamiento de grafos. El coste de una interconexión es fijo, independientemente del tamaño final del circuito.

Este sistema trata de realizar una optimización del área de los módulos del diseño, y posteriormente minimiza el área de interconexiones. Pero dado que el coste de una interconexión es fijo, es equivalente a minimizar su número. Este método puede conducir a diseños no óptimos, puesto que la minimización se debe realizar globalmente para módulos e interconexiones, ya que ambos pueden tener un peso de la misma magnitud en el área final.

Por otra parte, la bondad del algoritmo depende de la validez de la tabla de costes añadidos. En esta tabla se suponen conocidos las áreas de los módulos y de las interconexiones, que hemos visto que no son datos disponibles hasta que no se realiza el layout, y por tanto sería necesario estimarlos.

2.2.3.4. Sistema CHARM

Este sistema ha sido desarrollado en los AT&T Bell Laboratories por Nam-Sung Woo y Hyunchul Shin [ShWo89].

El algoritmo de asignación de hardware [Woo90a] [Woo90b] y [WoSh89] es iterativo constructivo, y necesita conocer el coste de los registros, UFs, multiplexores e interconexiones cada vez que realiza una asignación. Inicialmente la ruta de datos del circuito está vacía, y en cada iteración se va construyendo, intentando minimizar el hardware. El número necesario de registros se determina dinámicamente, y para cada variable se elige el registro que minimice las interconexiones. Los costes de los registros, muxes, UFs e interconexiones se supone que están almacenados.

Aunque este sistema realiza una optimización global de hardware del sistema, no realiza ninguna estimación del área de los módulos ni de las interconexiones, y parte de unos supuestos valores que, como ya hemos visto, no es posible conocerlos hasta que no se ha generado el layout.

2.2.4. Sistemas con coste de interconexión obtenido por floorplanning

2.2.4.1. BUD

El sistema BUD (Bottom-Up Design) [McKo86], [McFa89], [McKo90], fue desarrollado por Kowalski y McFarland en la Universidad Carnegie-Mellon.

Se trata de una herramienta de estimación, que realiza un tratamiento global del diseño, obteniendo información sobre las características físicas de los módulos que se pueden utilizar, permitiendo orientar la síntesis y evaluar los diseños obtenidos. La información obtenida por BUD sobre el retardo de los caminos críticos y el área de las interconexiones y módulos, la utiliza el sistema de síntesis DAA [Kowa85] [KGWF85] para generar el circuito.

Como hemos dicho anteriormente, BUD realiza un preprocesamiento del diseño para obtener un análisis global y hacer estimaciones de bajo nivel. Dado que este sistema es uno de los que mejores estimaciones obtienen sobre las características físicas de los diseños, vamos a estudiarlo en más profundidad.

Las entradas a BUD son cuatro:

- Una representación del flujo de datos y de control, llamada Value Trace (VT).
- Un conjunto de ligaduras y criterios de optimización.
- Una traza dinámica de la VT, que indica cuantas veces se ejecuta cada operación a lo largo de una serie de ciclos de ejecución.
- La base de datos [Wolf86], con información física y lógica detallada del área, forma y retardo de las celdas disponibles para el diseño.

Con esta información, BUD realiza un particionamiento jerárquico del flujo de datos en elementos que tienen un significado tanto lógico como físico (cada uno de ellos representa una parte del chip final). El criterio para realizar el particionamiento está basado en una métrica que define la proximidad entre operaciones como una medida de la ventaja de realizarlas en la misma UF, considerando factores como el grado de interconexión, el paralelismo potencial entre ellas y la funcionalidad común.

A partir de esta métrica se calcula una matriz de distancias, que guarda para cada par de operaciones la distancia entre ellas. A partir de esta matriz se construye un árbol jerárquico de particiones. Cortando el árbol por distintos niveles se obtienen diferentes particiones, y por tanto diferentes diseños, que deben evaluarse.

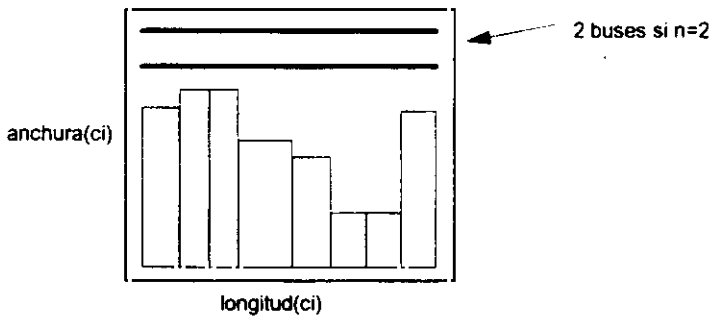


Figura 2.5 Estimación de área de un nodo hoja c_i

Supongamos una solución posible, es decir, un corte del árbol en un determinado nivel. Las particiones que quedan en el nivel del corte son los nodos hoja. Para cada una de ellas se realiza una evaluación sobre sus posibles formas, área y retardo. Primero se asignan las UFs necesarias para implementar todas las operaciones que contiene, teniendo en cuenta que las operaciones asignadas a la misma partición no pueden realizarse en paralelo a no ser que se puedan utilizar UFs diferentes. Seguidamente se realiza una planificación de las operaciones. A partir de la información disponible sobre el retardo y área de los módulos y buses existentes en las particiones, es capaz de estimar el tamaño y retardo de cada una de ellas (figura 2.5). La anchura de un nodo hoja $\mathbf{c_i}$ es la máxima de la anchura de todos los módulos que contiene más

la anchura de uno o de los dos buses (dependiendo si la función es unaria o binaria) de máxima anchura que contenga. La longitud es la suma de longitudes de todos los módulos.

Una vez que se han estimado las distintas formas y tamaños de las particiones hoja, se realiza un *floorplanning*. Se utiliza un algoritmo de programación lineal que recorre el árbol de forma ascendente, y decide el mejor tamaño y forma de todos y cada uno de los nodos del árbol. Para estimar la forma y tamaño de cada nodo utiliza la información sobre la forma y tamaño de sus nodos hijo, comenzando con la información obtenida sobre las particiones hoja. También se calculan el tamaño de las interconexiones entre nodos, suponiendo que cada interconexión está contenida en el menor rectángulo que contiene a todas las celdas conectadas.

A partir de esta colocación de los elementos en el circuito, es posible estimar el área del interconexionado y los retardos del diseño total, con los cuales se calculan el área total, el mínimo ciclo de reloj (se toma el máximo retardo a través de la ruta de datos para cualquier paso de control) y el tiempo de ejecución medio. El retardo de una interconexión se calcula mediante un modelo simple RC basado en su longitud y en la capacidad de salida de la fuente. La longitud de una interconexión se toma como la longitud de la partición, si es interna a ésta, mientras que si es entre dos particiones se toma la distancia Manhattan entre ellas.

Como hemos dicho, se pueden obtener distintos diseños a partir de diferentes cortes en el árbol de particiones. El mejor de todos los que se encuentran se pasa a DAA para que termine la asignación de hardware y la planificación.

Las estimaciones del área de los diseños tienen un error del 10-15% con respecto a layouts obtenidos por herramientas de diseño.

Aunque este sistema realiza estimaciones muy precisas sobre el área y retardo de los módulos, la complejidad del algoritmo es demasiado alta para un sistema de SAN que trate de recorrer un amplio Espacio de Diseño de Asignación (EDA). El tiempo de ejecución se dispara para un sistema con un GFD con un elevado número de nodos, cada uno de los cuales susceptible de ser implementado por varias UFs de la biblioteca.

2.2.4.2. Sistema ADAM

El sistema ADAM (Advanced Design AutoMation) [JaPP92] [KuPa90a] [JKMP89] se desarrolla en la Universidad de Southern California bajo la dirección de Parker. Está enmarcado dentro de otro proyecto más amplio, USC (Unified System Construction), que es un proyecto de investigación de síntesis a nivel de sistemas (figura 2.6), [KuPa91], [KuPa90b] y [WePa91].

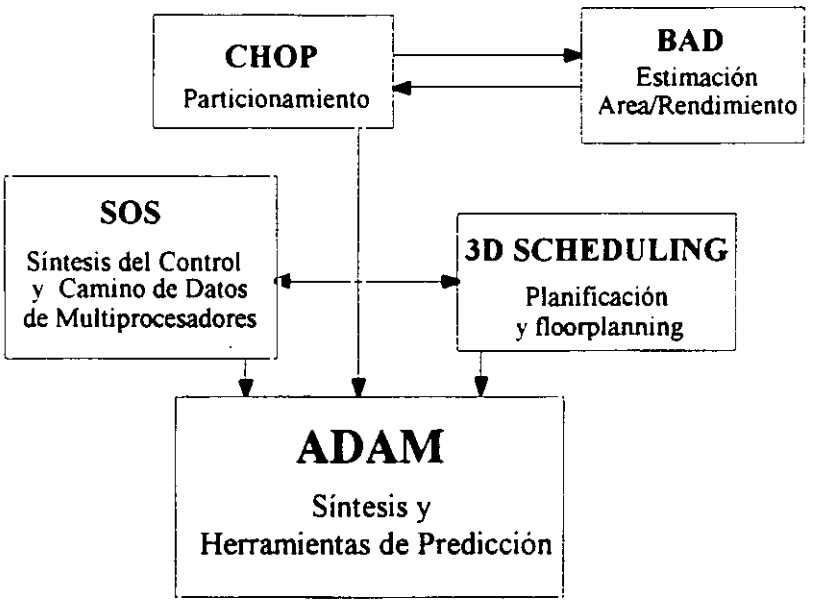


Figura 2.6 Sistema USC

ADAM consiste en dos subsistemas: una serie de herramientas de síntesis, que sintetizan desde una descripción del comportamiento, y una serie de predictores, que guían al diseñador.

Las predicciones se basan en un estudio del hardware mínimo necesario para implementar las operaciones del GFD. En [JaPP92] se deriva una predicción del área y retardo de los operadores del GFD. El algoritmo recibe como entrada el tipo de UF que va a realizar cada operación y supone que todas las operaciones deben ejecutarse en un ciclo de reloj. Primero se identifica el ciclo de reloj óptimo, y el mínimo número de UFs de cada tipo necesario. Seguidamente se utilizan los estimadores del área de interconexionado propuestos por Kurdahi

en PLEST [KuPa89], para un diseño RTL y un layout con celdas estándar (Apéndice A). Este sistema lo veremos en el apartado 2.3.

Una mejora de este sistema se propone en [ShJa93], donde se permiten operadores multiciclo y se estima qué tipo de módulo debe realizar cada una de las operaciones. Se deriva una relación entre el número de buses de un diseño y el área necesaria para interconexiones, multiplexores y drivers triestado. La longitud de un bus es una función lineal de la anchura del layout de las UFs conectadas al bus y su anchura es proporcional al número de bits. El área de UFs, registros, muxes etc. se obtiene de la biblioteca de módulos. El área mínima de registros se predice en [ShJa94].

Una vez realizadas las predicciones, se utilizan las herramientas de síntesis para generar el diseño. En [RMJL94] se propone un algoritmo que utiliza programación lineal para realizar la asignación de operaciones conjuntamente con un *floorplanning*. Para cada paso de control, se realiza la asignación de hardware y luego un *floorplanning*, y así va prediciendo el área de interconexiones. Para interconexiones que unen dos elementos, se toma la distancia Manhattan entre los centros de los dos elementos multiplicados por la anchura (que vienen dada por la tecnología). Si unen más de dos elementos, se calcula como la mitad del perímetro del rectángulo que contiene todos los elementos que une. Este algoritmo tiene una complejidad que depende del foorplanner utilizado, pero en general será elevada.

Este sistema tiene varios inconvenientes:

- Predice la longitud de bus como la anchura de layout de las UFs conectadas a él. Pero la anchura de un módulo que tenga interconexiones internas no se puede conocer antes de generar el layout del circuito total. Además, la colocación de una UF en el circuito dependerá de todos los módulos a los que esté conectada, y no se puede predecir que todas las UFs unidas por un bus van a estar unas al lado de otras.

- Utiliza PLEST para estimar el área de interconexiones (Apartado 2.3). Este estimador necesita conocer la estructura global del circuito y la longitud media de

una interconexión. Como valor medio de una interconexión utiliza valores obtenidos empíricamente para otros diseños. Sin embargo, este es un valor que depende de cada diseño particular, y no se puede garantizar que un valor obtenido para otros diseños se aproxime al valor real del diseño considerado.

- Las predicciones del número de módulos se realizan por separado para registros y UFs, con lo cuál no se puede garantizar una optimización global.

- La realización de *floorplanning* y asignación conjunta incrementa la complejidad del diseño y no permite explorar un amplio EDA. Además, este proceso se ejecuta para cada paso de control, con lo que no se estudia globalmente el GFD y no se pueden predecir asignaciones futuras.

2.2.4.3. Sistema de la Universidad de Cleveland (Case Western Reserve University)

Nourani [NoPa93] propone un método de estimación de área que pueda ser iterativamente llamado durante la SAN para obtener datos sobre el área de los diseños obtenidos y para optimizar los resultados. El estimador utiliza los módulos de la ruta de datos para realizar los cálculos, suponiendo:

- Que las interconexiones dentro de un módulo se realizan en capas diferentes a las interconexiones entre módulos.

- Que cada módulo se coloca en una única fila de celdas estándar.

El algoritmo se divide en dos partes:

- Primero se colocan todos los módulos en una fila, colocando más próximos aquellos que están más conectados, y calcula el número de pistas necesarios para realizar las conexiones.

- A continuación se pasa de un diseño en una fila a un diseño en 2 dimensiones, intentando conseguir un chip lo más cuadrado posible.

Para módulos muy grandes realiza una división en submódulos y aplica el mismo tratamiento.

Aunque este algoritmo es más rápido que los que utilizan celdas estándar para hacer los cálculos, sigue siendo demasiado complicado para realizar estimaciones dentro de un proceso de diseño. El máximo error que se produce en los experimentos presentados en la literatura es del 12%.

2.2.5. Conclusiones sobre las estimaciones de los sistemas de SAN

Todos los sistemas de SAN vistos en el apartado anterior, tratan de minimizar de una u otra forma el área de interconexionado. Sin embargo, muchos de ellos no realizan la minimización teniendo en cuenta el área que las interconexiones ocupan en el diseño final, sino que se preocupan sólo del número de multiplexores necesarios para realizarlas (Olympus, HIS). Existen otros sistemas que permiten utilizar multitud de posibles caminos para implementar las interconexiones (Elf), pero para medir el coste de las distintas opciones sólo tienen en cuenta el número de celdas que deben atravesar, pero no el de cableado propiamente dicho. Por último existen sistemas que, una vez asignado el hardware, tratan de minimizar el número de interconexiones (HAL y Parthenon). Estos sistemas no realizan ninguna estimación del área de interconexionado y, por tanto, los diseños consideran óptimos pueden no tener un área física óptima. No obstante, algunos de ellos (Olympus y Parthenon) sí realizan la conexión con herramientas de Bajo Nivel, de las cuales podrían obtener información muy valiosa para dichas estimaciones.

Otros sistema utilizan un **valor constante** para el coste de una interconexión; en algunos casos este coste es elegido por el diseñador (Chippe), en otros simplemente se supone conocido (SAW, CHARM) y en otros es un valor obtenido empíricamente para un estilo de diseño determinado (Sistema de la Universidad de California, Berkeley). Sin embargo, el valor del

coste de una interconexión depende de factores como la tecnología, la calidad de los algoritmos de colocación e interconexionado de módulos e incluso del diseño en particular, y por tanto, no es un valor conocido a priori, ni siquiera para un estilo de diseño.

Ninguno de estos sistemas estiman el área de los módulos, que sabemos que no se pueden tomar como elementos con un área constante, almacenada en la biblioteca de módulos. Tan sólo los sistemas que realizan un estudio del área/rendimiento de los circuitos mediante generación de *floorplanning*, obtienen estimaciones suficientemente precisas del área de los módulos e interconexiones (BUD, ADAM y Sistema de la Universidad de Cleveland). Sin embargo, el *floorplanning* es un proceso muy lento, que no permite explorar un amplio espacio de diseño, y por lo tanto, no consideramos apropiado incorporarlo en una herramienta SAN que pretenda analizar un gran número de diseños. Además, estos sistemas sólo pueden hacer predicciones reales cuando se conocen todos los módulos e interconexiones del circuito, que es cuando se puede realizar un *floorplan* correcto, pero no dentro del proceso de SAN, para poder tomar decisiones correctas.

Parece necesario estudiar otro tipo de estimaciones que nos permitan obtener datos sobre el valor del área de las interconexiones y no tengan una complejidad tan elevada.

2.3. Técnicas de Estimación de Bajo Nivel

Entendemos por técnicas de Estimación de Bajo Nivel aquellas que realizan los sistemas cuando la información del diseño es completa, es decir, se conocen todas y cada una de las celdas estándar y/o puertas de los módulos que lo integran y sus interconexiones. Estas estimaciones se generan al comienzar la tarea de *floorplanning*, que es una fase previa a la colocación e interconexionado de módulos. Durante esta fase se estima el número de filas necesarias para colocar todas las celdas del circuito, el ancho de las filas, el número de canales, etc. Una estimación de área precisa permite disminuir el número de iteraciones necesario para obtener un floorplanning correcto.

Basándonos en [PePr89], estos métodos se pueden clasificar en los siguientes grupos:

1- **Teóricos**, elaboran modelos matemáticos de las características físicas del diseño. Utilizan hipótesis sobre las distribuciones de las longitudes de las interconexiones y a partir de éstas estiman el área necesaria para interconexiones. Estos métodos no tienen en cuenta los detalles de cada diseño individual.

2- **Empíricos**: elaboran también modelos de las características físicas, pero extrayendo la información necesaria de los diseños. La mayoría de los sistemas pertenecientes a este grupo utilizan la regla de Rent [LaRu70]. Esta regla empírica relaciona el número de terminales (T) de una partición dentro de un circuito, que coincide con el número de conexiones externas, con el número de bloques que existen dentro de dicha partición (B), según:

$$T = K * B^r$$

donde K es el tamaño medio de los bloques y r es un parámetro empírico que toma los valores:

$$0.57 \leq r \leq 0.75$$

A partir del número de conexiones externas de cada partición, y utilizando diversas técnicas, veremos más adelante cómo se puede estimar la longitud media de las interconexiones.

3.- **Procedurales**: se basan en relaciones derivadas del conocimiento del diseño, de la estructura de las interconexiones y de las reglas de diseño. Incorporan muchos detalles propios de cada diseño.

A continuación vamos a estudiar algunos ejemplos de sistemas pertenecientes a los tres modelos.

2.3.1. Modelos teóricos

2.3.1.1. PLEST

PLEST (PLotting ESTimator) es un sistema desarrollado por Kurdahi y Parker [KuPa89] en la Universidad de Southern California. Es un estimador de área para diseños realizados con celdas estándar (Apéndice A). Se asumen filas de igual tamaño, celdas con entradas dobles, una en la parte de arriba y otra en la de abajo, e interconexiones de dos terminales que siguen caminos mínimos rectilíneos. El sistema recibe como entrada la anchura total de celdas W_{celdas}, el número de interconexiones N y la longitud media de una interconexión l_{media}. Este último valor se calcula a partir de valores obtenidos para otros diseños.

La probabilidad de que nazca una interconexión en un terminal i, **pb(*i*)** y su longitud **pL(*L*)** son variables aleatorias independientes. Se propone un distribución uniforme para pb(i) y geométrica para pL(L).

$$pb(i) = pb = \frac{N}{Wceldas}$$

$$pL(L) = \frac{1}{lmedia} * q^{L-1}$$

Primero se supone que se colocan todas las celdas en una fila y se estima el número de interconexiones que cruzan cada punto de la fila (figura 2.7). El máximo valor de este número, es el número de pistas necesarias para realizar el interconexionado para una sóla fila.

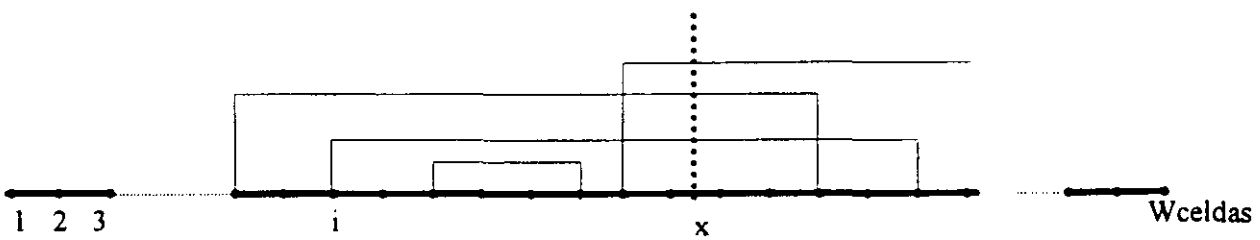


Figura 2.7 Número de interconexiones que cruzan un punto x

A continuación la fila se pliega en N_{filas} y se calcula el número de pistas para cada canal y el número de *feedthroughs* (Apéndice A). Conociendo la longitud máxima de una fila y el ancho de cada canal se calcula el área del circuito.

El error de este sistema es inferior al 10% y el tiempo de ejecución no muy elevado para diseños de pequeño tamaño (por ejemplo un sumador o un multiplicador). Pero, para diseños de mayor tamaño, como el tiempo de ejecución depende del número de celdas del circuito, se hace excesivamente grande para incorporarlo dentro de una herramienta de SAN que necesite llamarlo sucesivas veces. Además, el principal problema de este sistema es la necesidad de obtener como entrada el valor medio de una interconexión.

2.3.1.2. LAST

Este sistema también se ha desarrollado en la Universidad de Southern California [KuRa93]. Se encarga de predecir la función de forma del circuito para un rango de diferentes aspectos. El algoritmo se divide en tres partes:

- Primero un método constructivo para realizar particionamiento, construyendo un árbol de bloques, hasta llegar a una profundidad predefinida de particionamiento. En cada una de las divisiones trata de minimizar el número de interconexiones entre las particiones.

- A continuación se calcula el tamaño y forma de las distintas particiones mediante un método analítico llamado SCALE (Standard Cell Area and wire Length Estimation) que es una versión mejorada de PLEST [KuPa89], visto en el apartado anterior. SCALE lee la descripción RTL del circuito y estima los parámetros requeridos por PLEST, como son la anchura total de celdas estándar, el número de interconexiones punto a punto y una estimación de la longitud media de una interconexión. Para obtener esta última utiliza la regla de Rent con un valor

r=0.7. Con estos parámetros, PLEST realiza la estimación de área de cada uno de los bloques.

- Una vez que se conocen las funciones de forma de las distintas particiones, se recorre el árbol en "post orden" y se componen las funciones de forma de los nodos padre.

Este método desprecia el área de cableado en la dirección vertical en interconexiones entre bloques, y el área de los *feedthroughs* correspondientes. Este tratamiento es posible si se suponen dos capas de metal para realizar interconexiones. El error es del orden del 5%, y el tiempo de ejecución bastante elevado, dependiendo del tamaño del diseño, lo cual hace imposible llamarlo sucesivas veces desde una herramienta de SAN. Además, no permite realizar estimaciones hasta que no se conocen todos los elementos del circuito.

2.3.1.3. Sistema de la Universidad de California, San Diego

Hamada [HaCC92] propone un layout donde todas las celdas se distribuyen sobre una red de dos dimensiones. Dado que los algoritmos de colocación de módulos tratan de minimizar el área del CI, se supone que la estructura topológica del circuito se refleja en la estructura local de la colocación física de las celdas, y por tanto dos componentes que están topológicamente cercanas lo estarán también físicamente. El modelo de distribución de la longitud de interconexión entre dos celdas que propone Hamada es el de **Weibull**, que tiene la forma siguiente:

$$f(\alpha,\beta,l) = g * l^{\beta-1}\exp(-\alpha * l^{\beta})$$

donde α es el parámetro llamado *de escala*, β es el parámetro *de forma*, g es una constante y l es la longitud. Esta función f mide la probabilidad de que exista una interconexión de longitud l entre dos celdas.

Para interconexiones entre varias celdas supone que, dada una colocación determinada de las celdas obtenida por un algoritmo que minimiza la longitud de las interconexiones, una pequeña perturbación de las celdas adyacentes no cambia la longitud del cableado significativamente. Esto es debido a que los algoritmos de colocación de módulos tratan de minimizar una cierta función objetivo (la longitud de las interconexiones), y existen una serie de fuerzas contrapuestas (las distintas interconexiones de una celda), que alcanzan el equilibrio en ese estado de mínima longitud. Así aparece la existencia de un parámetro global, equivalente a la temperatura en termodinámica, y una función potencial, que es función de la posición (x,y) de la celda. Tomando como función objetivo para minimizar el medio perímetro del mínimo rectángulo que contiene todas las celdas interconectadas, obtiene como función potencial:

$$e(x,y) = \frac{3m}{m^2+2m-2}(|x|+|y|)$$

donde ***m*** es el número de celdas conectadas. A partir de esta función, y mediante argumentos similares a los de la mecánica estadística, obtiene la longitud media de la interconexión de ***m*** celdas como una función sencilla de ***m*** y de un parámetro, γ, que mide el estado de equilibrio, y cuanto mayor es su valor más se aproxima el sistema al estado de mínima energía.

Para calcular γ necesita realizar un estudio global del circuito, puesto que los algoritmos de colocación de módulos optimizan la colocación global (no local) de las celdas, y hay que tener en cuenta la influencia de la existencia de una interconexión sobre las otras. Para estudiar el circuito, lo descompone en una secuencia creciente de **vecindades multi-nivel**. Se entiende que una celda es vecina de otra, si existe una interconexión que las une. Se parte de una celda cualquiera O (figura 2.8). En el primer nivel se colocan todas las celdas conectadas a la primera (A, B y C). En el segundo nivel, las celdas conectadas a las del primer nivel (D,E,F), y así sucesivamente.

La distancia topológica entre dos celdas se mide por el número mínimo de niveles entre ellas. Por ejemplo, la distancia entre O y F en la figura 2.8 es 2.

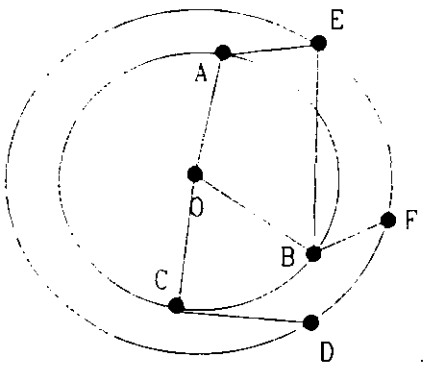


Figura 2.8 Descomposición de vecindades multi-nivel

Sobre cada una de estas vecindades se define la densidad de interconexiones $w_m(x,y)$ para redes de *m* terminales como la probabilidad de que una red de m terminales exista entre el origen y *(x,y)*, y supone que esta probabilidad sigue la distribución Weibull anterior, con

$$l=|x|+|y|$$

Aplicando de nuevo el principio de minimización de la energía, se obtiene el valor de una interconexión de m módulos.

De acuerdo con los datos suministrados en [HaCC92], el error medio cometido con esta técnica es del 9%, con una desviación máxima de 16%, y el tiempo de cálculo bastante elevado para diseños de complejidad media (como un sumador de 64 bits) y muy elevado para diseños de mayor complejidad. Además es necesario calcular algunos parámetros empíricamente.

2.3.2. Modelos empíricos

2.3.2.1. Modelo para Arrays de puertas en una dimensión

Sastry [SaPa86], desarrolló un sistema de estimación de área de interconexionado que es un modelo mixto, entre empírico y teórico. La solución que propone es válida para arrays de puertas (Apéndice A) suponiendo una distribución en una dimensión. El número de celdas en una partición es:

$$Nceldas = \lambda * x$$

donde x es la dimensión de la partición y λ es una constante de proporcionalidad.

Utilizando esta fórmula y la regla de Rent, explicada anteriormente, se llega a una distribución de Weibull para las longitudes de las interconexiones, con parámetro de escala $\alpha=K$ y parámetro de forma $\beta=r$, donde K y r son los parámetros de la regla de Rent. El principal problema de este método es la obtención empírica de los valores de r y de K. Según los resultados presentados en la literatura, el error en la estimación de la longitud media es superior al 10%.

2.3.2.2. Modelo para Arrays de Puertas en dos dimensiones

Gura [GuAb89] propone utilizar la regla de Rent para calcular el valor medio de una interconexión, pero con dos coeficientes r diferentes, según se realiza el estudio en 1 ó 2 dimensiones. En dos dimensiones el número de celdas en función de x, donde x se mide radialmente desde el origen, se aproxima por:

$$Nceldas \cong 2 * x^2$$

Utilizando la regla de Rent se llega también a una distribución de Weibull, con parámetro de escala $\alpha = k2^r$, y parámetro de forma $\beta=2r$. El problema principal de este método, como del anterior, es que los parámetros se calculan empíricamente. Además, el error cometido en algunos casos puede llegar a ser del 75%, ya que no tiene en cuenta las características propias de cada diseño.

2.3.3. Modelos procedurales

2.3.3.1. Sistema de la Universidad de Yale (New Haven)

Sechen [Sech87] plantea un sistema para calcular la longitud media de una interconexión para algoritmos de colocación de módulos aleatorios y optimizados. La longitud de una

interconexión se calcula como la mitad del perímetro del mínimo rectángulo que contiene todos sus terminales. Supone un diseño en el cual todas las celdas tienen el mismo tamaño. Para una red que conecta sólo dos celdas y un algoritmo de colocación de módulos aleatorio, obtiene como valor medio de una interconexión,

$$L_{media} = \frac{2}{3}\sqrt{N_{total}}$$

donde N_{total} es el número total de celdas.

Para interconexiones de m celdas y una colocación aleatoria, calcula todas las posibles posiciones de las m celdas, mide el valor de la longitud para cada una de esas posiciones y calcula el valor medio.

Para algoritmos de colocación optimizados utiliza el mismo método, pero considerando que si una celda está unida a $m-1$ celdas, estas m celdas estarán colocadas en un rectángulo de perímetro mínimo (es decir, lo mas parecido a un cuadrado), con el fin de minimizar el área de interconexionado. Primero calcula el valor medio de m para el circuito en cuestión, y el cuadrado mínimo que contendría esas celdas. A continuación calcula las diferentes posiciones de las celdas en ese cuadrado y el valor medio de la interconexión.

Como vemos, este sistema no tiene en cuenta la influencia que una interconexión tiene sobre otras cuando se realiza la colocación de módulos. Es decir, calcula el área de interconexiones para cada una de ellas en particular, sin tener en cuenta la influencia del resto sobre la posición de las celdas. Por eso, la aproximación para interconexiones que conectan un gran número de terminales, necesita una gran cantidad de cálculos y no resulta suficientemente precisa. En estos casos el error puede ser de hasta el 40%. Para circuitos con interconexiones entre 2 ó 3 celdas el error medio es el 7%.

2.3.3.2. Sistema de la Universidad de Rutgers, New Jersey

Chen [ChBu88] realiza una estimación del área del interconexionado para colocación aleatoria de celdas, utilizando cálculo combinatorio. Con la suposición inicial de que en cada fila se colocarán el mismo número de celdas, utiliza un algoritmo iterativo para calcular el número de filas N_{filas}, de forma que el diseño sea lo más cuadrado posible. Este algoritmo va obteniendo valores crecientes del número de filas, y para cada iteración se calcula la longitud de la fila L_{fila}.

$$Lfila = \frac{Nceldas}{Nfilas} * Wcelda$$

Donde N_{celdas} es el número total de celdas estándar, y W_{celda} es la anchura media de una celda.

Si para este valor de N_{filas} y la longitud de fila obtenida no se pueden colocar los puertos de E/S, se vuelve al principio del bucle incrementando el número de filas N_{filas}.

Una vez obtenido N_{filas}, se calcula el valor esperado de número de pistas N_{pistas}, utilizando cálculo combinatorio y bajo tres suposiciones:

- En cada pista hay una única interconexión.
- Los *feedthroughs* son líneas rectas que atraviesan una o más filas.
- Las filas son suficientemente largas para ser ocupadas por el máximo número de celdas que puede conectar una interconexión. Es decir, dada una interconexión de un determinado número de celdas, todas las celdas pueden estar colocadas en la misma fila.

Utilizando N_{pistas}, la altura de un pista H_{pista}, y el número de filas se calcula la altura del circuito $H_{circuito}$.

$$Hcircuito = Nfilas * Hcelda + Npistas * Hpistas$$

En el paso siguiente calcula, también por cálculo combinatorio, el número de *feedthroughs* en cada fila y en qué fila se obtiene el máximo N_{feed}. Con este valor, y la anchura de un *feedthrough* W_{feed}, se calcula el ancho de la fila más larga (la que más *feedthroughs* tenga), que será el ancho del circuito $W_{circuito}$.

$$Wcircuito = Wcelda * \frac{Nceldas}{Nfilas} + Nfeed * Wfeed$$

Este método no tiene en cuenta que se tienden a colocar próximas aquellas celdas más conectadas y por eso siempre sobrestima el área de cableado. El área sobrestimada para pequeños diseños es del 40-70% sobre la real del circuito. No existen resultados para diseños de tamaño mediano ó grande, puesto que la complejidad de utilizar combinatoria para un número grande de celdas es muy elevada.

2.3.3.3. Sistema de la Universidad de Kaiserslautern

Zimmerman [Zimm88] propone un algoritmo de estimación del área y funciones de forma de circuitos VLSI. El modelo asume una estructura jerárquica en forma de árbol donde se conocen las funciones de forma de los nodos hoja. Estos nodos pueden ser macroceldas (Apéndice A), celdas estándar o módulos de mayor complejidad cuya forma y área es conocida. También recibe como entrada un conjunto de **factores de demanda de pista**, calculados empíricamente para cada estilo de diseño, que miden, dependiendo del tipo de módulo (terminal o no) y de la orientación de la interconexión, el tanto por uno de pistas demandadas.

Mediante un algoritmo de min-cut (Apendice B) se genera el árbol de particiones, y para todos los nodos de éste se generan las diferentes funciones de forma, para orientación vertical y horizontal, y se escogen las de menor área para cada coordenada x (figura 2.9).

La estimación del área de interconexionado se realiza a partir de los factores de demanda de pista y del número de conexiones del nodo. Este área puede reducirse según el número de celdas *feedthroughs* disponibles.

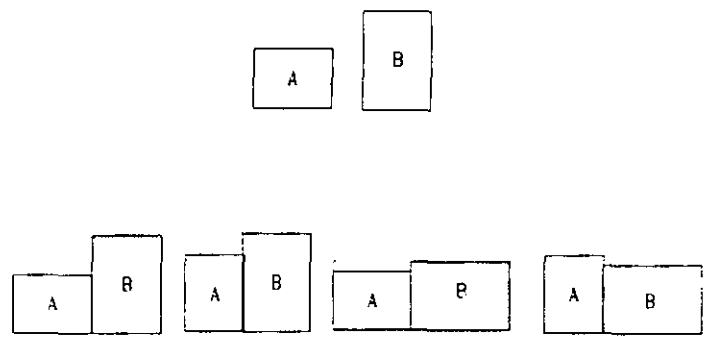


Figura 2.9 Funciones de forma para las celdas A y B

Subiendo hacia la raiz en la jerarquía del árbol se obtienen las distintas formas y tamaños del circuito global. El error máximo de este método para los experimentos presentados es del 10%, y su complejidad muy elevada, puesto que necesita calcular las distintas funciones de forma de todos los nodos del árbol

2.3.3.4. Sistema de la Universidad de California, Berkeley

Pedram y Preas [PePr89] proponen un modelo básico de estimación de área para colocación aleatoria de celdas y otro para colocación optimizada. En ambos casos se supone un algoritmo de interconexionado optimizado y dos capas de metal, una para realizar las interconexiones horizontales (***metal1***) y otro para las verticales (***metal2***). La anchura del chip es la anchura de la fila de celdas con mayor número de *feedthroughs*, y la altura es la suma de las alturas de las filas más las de los canales (Apéndice A). El layout de celdas estándar se modela como un array regular de $N_{cmedio}*N_{filas}$, donde N_{filas} es el número de filas y N_{cmedio} es el número medio de celdas por fila.

El algoritmo de interconexionado se supone que encuentra un árbol de "*spanning*" (ver Apéndice B) mínimo para realizar las conexiones.

Para una colocación aleatoria, se supone que los terminales están uniformemente distribuidos sobre la malla de celdas. Calcula por separado las longitudes de las interconexiones sobre el *metal1* y sobre el *metal2*, utilizando cálculo combinatorio. Por ejemplo, para una interconexiones de m terminales, la suma de longitudes sobre el metal1 (**LM1(*m*)**) será:

$$LM1(m) = \sum_{i=1}^{Min(m,Nfilas)} \left(\frac{1}{Nfilas}\right)^m * \binom{m}{i} * ContrLM1(i,m)$$

El primer término $\left(\frac{1}{Nfilas}\right)^m$ es la probabilidad de colocar m terminales en un subconjunto de N_{filas}; el segundo $\binom{m}{i}$ es el número de formas de seleccionar *i* filas entre *m*; y el tercero ***ContrLM1(i,m)*** es la contribución de m terminales que ocupan *i* filas ($i \leq m$). Para computar este tercer término, es necesario examinar todas las posibles configuraciones de m terminales sobre *i* filas, calcular la longitud en cada caso y hallar la media. Este cálculo lo realiza utilizando el mismo método. Realizando la suma para todas las interconexiones que existen, se obtiene una fórmula para calcular la longitud total de interconexionado sobre el *metal1*.

De forma similar se calcula para el *metal2* y también se obtiene el número de *feedthroughs* necesarios, el número de pistas, la longitud media de una interconexión, etc.

Para una colocación de módulos optimizada utiliza un método similar, pero para una interconexión de m terminales restringe el espacio de las posibles posiciones de éstos a una submalla de *xy*, donde *x* e *y* se computan teniendo en cuenta la influencia local de otras interconexiones sobre la interconexión en cuestión. De nuevo se utiliza el método de las vecindades, visto anteriormente para el sistema de la Universidad de California, San Diego, y así calcula la dimensión dicha submalla. Supone que los *m* terminales están uniformemente distribuidos sobre esta malla *x*y*.

Para diseños de tamaño medio (por ejemplo un sumador de 64 bits) y para pequeños valores de *m*, el error cometido por este método es inferior al 10% y el tiempo de ejecución no es muy

elevado. Pero para diseños donde existen celdas que conectan muchos pines, el tiempo de ejecución se dispara y el error es mucho mayor.

2.3.3.5. Sistema de la Universidad de California, Irvine

En [WuCG91] se propone un método basado en la descomposición de los módulos en *bit-slices*. Este sistema supone dos capas de metal para realizar las interconexiones: una para las internas al *bit-slice* y otra para las externas. Primero se divide la ruta de datos en *bit-slices* y mediante un algoritmo de min-cut (ver Apéndice B) se colocan las componentes de cada *bit-slice* en una fila (figura 2.10). La longitud del canal (que es igual que la de la fila) depende del número de celdas del *bit-slice* (que es la anchura del *bit-slice*), y puede obtenerse de una biblioteca de módulos. La anchura del canal depende de la densidad de pistas necesaria para conectar todas las redes del *bit-slice*. Esta a su vez depende de la tecnología y del número máximo de interconexiones que pasen por un punto del canal. Mediante el algoritmo del borde izquierdo (Apéndice B) realiza la asignación de interconexiones a pistas y así calcula la densidad de pistas para cada canal.

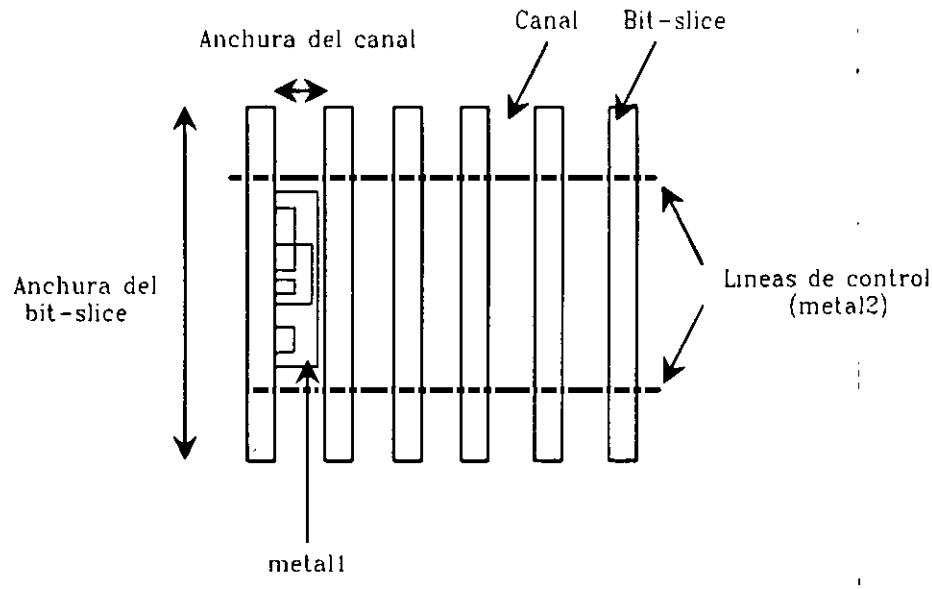


Figura 2.10 Colocación de *bit-slices*

Este algoritmo presenta un error inferior al 10% y una complejidad O(Mlog$N$), donde N es el número de interconexiones del circuito. Por lo tanto su complejidad es menor que la del resto de los algoritmos de estimación de SBN, que dependía del número de celdas. Sin embargo, sigue siendo demasiado elevada para integrarla en un sistema de SAN que explore un amplio número de diseños.

2.3.4. Conclusiones de las estimaciones de SBN

A lo largo de esta presentación de sistemas de estimación de SBN se ha comprobado el principal problema que tienen todos estos métodos: su elevada complejidad. Debido a que realizan las estimaciones trabajando con el conjunto de celdas e interconexiones del circuito, en cuanto los diseños son de un tamaño medio o grande, los tiempos de ejecución son demasiado elevados para poderlos utilizar dentro de una herramienta de SAN. Sin embargo, tienen una ventaja fundamental: su precisión. La mayoría de ellos son capaces de predecir el área de un circuito con una error inferior al 10%.

Entre los modelos teóricos, PLEST tiene la complejidad más baja, pero adolece de un problema fundamental: necesita recibir como entrada la longitud media de una interconexión del circuito, que es un dato que a su vez depende del propio circuito y no se conoce. Este problema es subsanado por LAST pero, a cambio, su tiempo de ejecución es bastante elevado para diseños de tamaño medio. Lo mismo ocurre para el método propuesto por Hamada, que además tiene un nuevo problema: necesita calcular varios parámetros empíricamente para cada estilo de diseño. Los errores máximos de estos tres sistemas, para los ejemplos dados por sus autores en la literatura, son 10%, 5% y 9% respectivamente, y los tres tienen en común que estiman el área de interconexionado mediante algoritmos que tienen en cuenta la colocación de los módulos en el diseño final, que a su vez viene determinada por las interconexiones entre dichos módulos.

Los modelos empíricos (propuestos por Sastry y Gura para arrays de puertas) necesitan conocer el valor de los parámetros de la regla de Rent, y además, no tienen la misma precisión

para todos los diseños. Estos sistemas parten de una función de distribución del interconexionado un tanto arbitraria, y en algunos casos el error puede ser hasta del 75%.

Los modelos procedurales son los que utilizan más información sobre la tecnología de diseño, la forma de trabajo de los algoritmos de colocación e interconexionado de módulos y el diseño en particular. En general, la utilización de cálculo combinatorio hace inutilizables estos sistemas para circuitos de tamaño grande, y son útiles para estimar, por ejemplo, el área de los módulos de una biblioteca. Por ejemplo, el sistema de la Universidad de Rutgers da buenos resultados para algoritmos de colocación no optimizados (es decir, si las celdas se colocan aleatoriamente en el circuito) y de pequeño tamaño (un sumador de 8 bits). Sin embargo no es válido si se utiliza un algoritmo que optimice dicha colocación (el error sería del 40%), y no hay resultados para diseños de mayor tamaño.

El resto de los métodos procedurales presentados en este capítulo sí tienen en cuenta el hecho de que los algoritmos de colocación de módulos tienden a minimizar el interconexionado. Esto se realiza, bien mediante un algoritmo de floorplanning (Universidad de Yale, Kaiserslautern y California -Irvine -), o bien mediante un estudio del diseño que considere las influencias de unas interconexiones sobre otras, como el sistema de la Universidad de California, Berkeley. Debido a este estudio del diseño, estos métodos obtienen resultados excelentes, con errores en todos los casos inferiores al 10%. Sin embargo, la mayoría de estos sistemas sólo son válidos para diseños de pequeño tamaño, y debido a su elevada complejidad no pueden aplicarse en un sistema de SAN.

Por ejemplo. el sistema desarrollado en la Universidad de Yale sólo es válido para diseños de pequeño tamaño, donde no existan redes que conecten un número grande de celdas. Si no es así, el error puede ser hasta del 40%.

Por su parte, el sistema desarrollado en la Universidad de California, Berkeley, tiene tiempos de ejecución aceptables para diseños de tamaño medio, pero se disparan cuando se incrementa

el número de celdas. Lo mismo ocurre con el método desarrollado por Zimmerman en la Universidad de Kaiserslautern, que además debe calcular empíricamente ciertos parámetros.

Por último, el sistema desarrollado en la Universidad de California -Irvine- hace una división de la ruta de datos en *bit-slices*, y supone que cada uno de ellos se coloca en una fila de celdas. Para un circuito con un número elevado de celdas es necesario colocar más de un *bit-slice* en cada fila y la validez de este método queda en entredicho.

Podemos concluir que estos métodos no son aplicables a diseños con un gran número de celdas, cuando las estimaciones se necesitan realizar un gran número de veces, como ocurre en un sistema de SAN. Sin embargo son bastante precisos, gracias a que consideran las características físicas del diseño y la metodología de trabajo de los algoritmos de colocación e interconexionado. Por tanto parece necesario generar nuevas estimaciones, de baja complejidad y con suficiente fidelidad para guiar correctamente el proceso de síntesis, que tengan en cuenta las consideraciones de estos sistemas.

2.4. Conclusiones generales

En la tabla 2.1 se presenta un resumen de las estimaciones realizadas por los sistemas de SAN y SBN revisados a lo largo de este capítulo. Para cada uno de los sistemas se muestra información sobre cómo estima el coste los módulos e interconexiones, el retardo considerado de estas últimas, el error y la complejidad de los algoritmos utilizados para obtener estos valores y, por último, si son aplicables o no en un proceso de SAN. En la columna donde se muestran los errores, existen casillas donde no se tienen los valores aproximados y se ha puesto un "?". En todos los casos donde aparece éste, aunque el valor del error es indeterminado, dada la funcionalidad de los algoritmos no puede tener un valor óptimo.
De esta tabla se puede deducir varias conclusiones:

- Que ningún sistema de SAN realiza estimaciones sobre el área de los módulos que tengan en cuenta las características físicas de estos, el área de las interconexiones

internas, etc. La mayoría de ellos supone que tienen un área fija o bien sólo consideran su número. Los sistemas de SBN, como realizan estimaciones a nivel de celdas (celdas estándar o arrays de puertas) o de bit-slices, conocen dicha área.

Sistema	Coste Módulos	Coste Interconexiones	Retardo Interc.	Error	Complejidad	Aplicable en SAN
Olympus	fijo	área muxes	muxes	?	baja	Sí
Elf	fijo	área muxes	-	?	baja	Sí
HIS	fijo	número muxes	No	?	baja	Sí
HAL	número	número interconex.	No	?	baja	Sí
Parthenon	número	número interconex.	No	?	baja	Sí
Chippe	fijo	fijo	No	?	baja	Sí
Berkeley	fijo	fijo	No	?	baja	Sí
SAW	fijo	fijo	No	?	baja	Sí
CHARM	fijo	fijo	No	?	baja	Sí
BUD	fijo	**floorplan**	RC	**15%**	muy alta	No
ADAM	fijo	**floorplan**	No	**10%**	alta	No
Cleveland	fijo	**floorplan**	No	**12%**	alta	No
PLEST	celdas	**floorplan**	-	**10%**	alta	No
LAST	celdas	**floorplan**	-	**5%**	muy alta	No
San Diego	celdas	**distrib./Vecindades**	-	**9%**	muy alta	No
Array 1dim	celdas	distrib./R. Rent	-	10%	alta	No
Array 2dim	celdas	distrib./R. Rent	-	75%	alta	No
U. Yale	celdas	**floorplan**	-	**7%**	alta	No
U. Rutgers	celdas	combinatoria	-	40%	muy alta	No
U. Kaisersla	celdas	**floorplan**	-	**10%**	muy alta	No
U. Berkeley	celdas	**combin. /vecindades**	-	**10%**	muy alta	No
U. Irvine	bit-slices	**floorplan**	-	**10%**	alta	No

Tabla 2.1 Estimaciones que realizan los sistemas SAN y SBN

- Todos los sistemas que obtienen estimaciones en dos dimensiones, con un error inferior o igual al 15% (la tabla están marcados en negrita), son aquellos que consideran de alguna forma cómo se colocan los módulos en el diseño final. Si para estos sistemas observamos los algoritmos que estiman el coste de las interconexiones, bien realizan un floorplaning, o bien utilizan algún método que

tiene en cuenta la influencia de las interconexiones en la colocación de los módulos, por medio de un sistema de vecindades multinivel, etc. Sin embargo, todos estos algoritmos que realizan un estudio tan exhaustivo del diseño final, tienen una complejidad alta o muy alta, y no pueden utilizarse en un sistema de SAN. Para todos los demás algoritmos los resultados obtenidos distan mucho de ser óptimos.

- Sólo el sistema BUD realiza estimaciones sobre el retardo de las interconexiones. Para las tecnologías actuales este retardo tiene una influencia notable en el rendimiento del circuito, y no considerarlo puede dar lugar a circuitos con un funcionamiento incorrecto. La estimación de estos retardos se realiza a partir del conocimiento del valor de las longitudes de las interconexiones. Por eso es necesario tener un método que calcule estos valores con suficiente precisión. Por otra parte, el modelo RC, que utiliza BUD, no siempre es adecuado para calcular los retardos de las interconexiones.

- Todos los sistemas revisados que realizan algún tipo de estimación, tratan de calcular el área total del circuito, cuando se conocen todos sus elementos, pero no permiten estimar el área de una sola interconexión, o de un módulo, cuando aún no se tiene una información completa del diseño. Esta información es necesaria dentro de un algoritmo de SAN que necesite tomar decisiones durante la generación de un diseño.

Podemos deducir que no existen sistemas de SAN que realicen estimaciones del área y retardo de los circuitos con la suficiente fidelidad para guiar correctamente el proceso, y es necesario encontrar métodos para obtener resultados realistas.

Por otra parte, todos los métodos de estimación precisos tienen en cuenta **la tecnología de diseño** y la **metodología de los algoritmos de colocación e interconexionado**. Parece por lo tanto indispensable considerar estos factores para realizar estimaciones acertadas. Sin embargo, los métodos de este tipo que aparecen en la literatura utilizan algoritmos muy complejos, y no pueden utilizarse en SAN.

Basándonos en estas características, en este proyecto de investigación se va a tratar de encontrar un método de estimación que, además de tener una **complejidad baja**, para poder

utilizarlo sucesivas veces en un proceso de SAN, tenga en cuenta todos los factores necesarios para que sea **preciso** y, sobre todo, **fiel**.

Pero, antes de pasar a buscar este método, vamos a enmarcar este trabajo de investigación dentro del sistema de SAN donde se ha incluido, el sistema FIDIAS, que presentamos en el próximo capítulo.

Capítulo 3.

El Sistema FIDIAS

3.1. Estructura del Sistema FIDIAS

Este trabajo de investigación está enmarcado dentro de una línea de trabajo sobre Síntesis de Alto Nivel de sistemas digitales, cuyo resultado práctico más notable es una herramienta denominada FIDIAS (Flexible Intelligent Designer for Integrated Architecture Synthesis) [SMTH95], y que se lleva a cabo en el seno del grupo de Arquitectura de Computadores del Departamento de Informática y Automática de la Universidad Complutense de Madrid.

Como ya se mencionó en el capítulo 1, el objetivo principal de la SAN es, dado el comportamiento de un sistema digital descrito a nivel algorítmico, y un conjunto de restricciones y objetivos, generar una estructura a nivel de transferencia de registros (RTL) que implemente dicho comportamiento y cumpla las restricciones y objetivos.

También se explicó que la SAN consta de una serie de tareas interrelacionadas, como son la planificación de las operaciones en pasos de control y la asignación de elementos hardware para realizar dichas operaciones y almacenar los resultados. El sistema FIDIAS está formado por una serie de módulos que realizan estas tareas de forma coordinada y conjunta. En la figura 3.1 se muestra el diagrama de bloques de este sistema, con los distintos módulos y las subtareas que realiza cada uno de ellos.

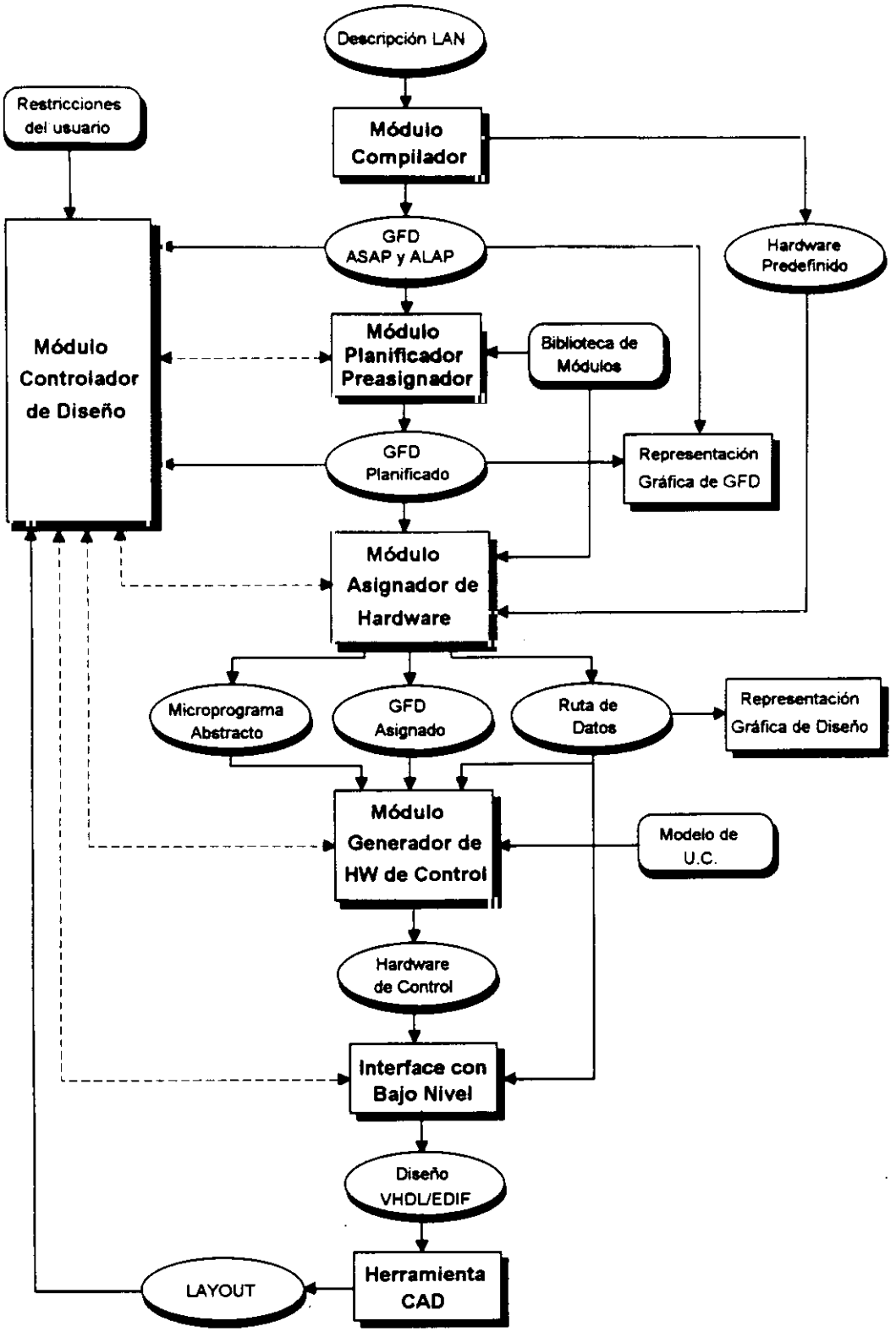


Figura 3.1 Diagrama de Bloques del Sistema Fidias

Estos módulos son:

- El Compilador
- El Planificador-Preasignador
- El Asignador de Hardware
- El Generador del Hardware de Control
- El Experto de Diseño
- El Interfaz con Bajo Nivel

El sistema dispone además de una serie de fuentes de información:

- La biblioteca de módulos
- Las restricciones y objetivos impuestos por el usuario
- El Hardware predefinido
- El modelo de Unidad de Control deseado

Por otra parte, se cuenta con una serie de herramientas gráficas que permiten visualizar el grafo de flujo de datos con las diferentes planificaciones, la ruta de datos con diferentes asignaciones de hardware, etc.

A continuación vamos a tratar brevemente cada uno de estos módulos.

- **El Compilador.** Este módulo [SSMT89] transforma la descripción inicial del comportamiento del circuito, que viene dada en VHDL, en un grafo de flujo de datos (GFD), que es una estructura más fácilmente manejable por los demás módulos del sistema. También realiza una doble planificación del GFD: las planificaciones ASAP y ALAP, que fueron vistas en el capítulo 1. Estas planificaciones se realizan

suponiendo que todas las operaciones son monociclo, puesto que todavía no se conoce el tiempo de ciclo, y sirven como entrada a otros módulos del sistema para realizar un estudio global del GFD.

- **El Planificador-Preasignador**. Este módulo se encarga de asignar a cada nodo del GFD una etapa de control y un tipo de UF. La asignación de hardware y la planificación de operaciones están muy interrelacionadas entre sí, y realizarlas de forma totalmente aislada puede conducir a la obtención de diseños que no cumplan realmente los objetivos del usuario. Esto se ha tratado en el sistema FIDIAS mediante la creación de este módulo, que al mismo tiempo que realiza la planificación temporal obtiene la información del GFD necesaria para preasignar las UFs [More95].

- **El Asignador de Hardware**. Este módulo [SMTH90][SMHS91][SMHT91] [SMTH92] [Sept90] realiza la asignación de operaciones a UFs, de resultados a registros y de transferencia de operadores a interconexiones. Su objetivo fundamental es minimizar el hardware del circuito cumpliendo la planificación temporal del GFD. La metodología utilizada se basa en una exploración del espacio de diseño mediante un algoritmo de tipo ramificar y acotar, con una serie de heurísticas de guía y acotación que permiten orientar la búsqueda y reducir el número de posibles diseños explorados.

- **El Generador de Control**. Es el módulo [Mend93] encargado de generar la Unidad de Control para el camino de datos obtenido por el asignador de hardware, de acuerdo con un tipo de modelo predefinido.

- **El Interfaz con Bajo Nivel**. Este módulo se encarga de trasladar la salida de FIDIAS a los formatos VHDL y EDIF, que pueden utilizarse como entrada de una herramienta CAD para generar el layout final. La conexión de FIDIAS con una herramienta de CAD se realizará en este trabajo de investigación, con el fin de

obtener información para realizar las estimaciones necesarias durante la SAN, como veremos a lo largo de los próximos capítulos.

- **El Experto de Diseño**. La tarea de este módulo [SMTH92] [MSTF93] [MSTH92] [Mozo92] es coordinar la operación del resto de los módulos, conduciendo el proceso de síntesis como lo haría un usuario experto, para cumplir los objetivos impuestos por el usuario.

3.2. El Proceso de Síntesis en FIDIAS

En la figura 3.2 se muestra el proceso global de SAN con el sistema FIDIAS. Como ya se ha dicho, el módulo Experto controla dicho proceso. La primera tarea llevada a cabo es la compilación. A continuación se realizan el resto de las subtareas y después de cada una de ellas el módulo Experto evalúa los resultados y decide si debe realizarse de nuevo alguna de las subtareas, modificando alguno de los parámetros de entrada, o bien puede continuarse con la siguiente.

Podemos distinguir dos bucles principales. El primero realiza la tarea de planificación-preasignación, dando como resultados un **GFD planificado**. Este grafo es evaluado por el Experto antes de pasar a la siguiente subtarea.

El segundo lazo realiza la asignación de hardware, tratando de conseguir un diseño que cumpla las restricciones del usuario con un área lo más pequeña posible. De nuevo el resultado, la **ruta de datos** y el **GFD asignado**, es evaluado por el módulo Experto, que puede decidir volver a comenzar el proceso, si los objetivos no se han cumplido, o bien pasar el diseño a la herramienta de síntesis de bajo nivel. En el primer caso, sería necesario seleccionar un nuevo tiempo de ciclo y/o incrementar el número de pasos de control, y realizar de nuevo la planificación, antes de realizar la asignación de hardware.

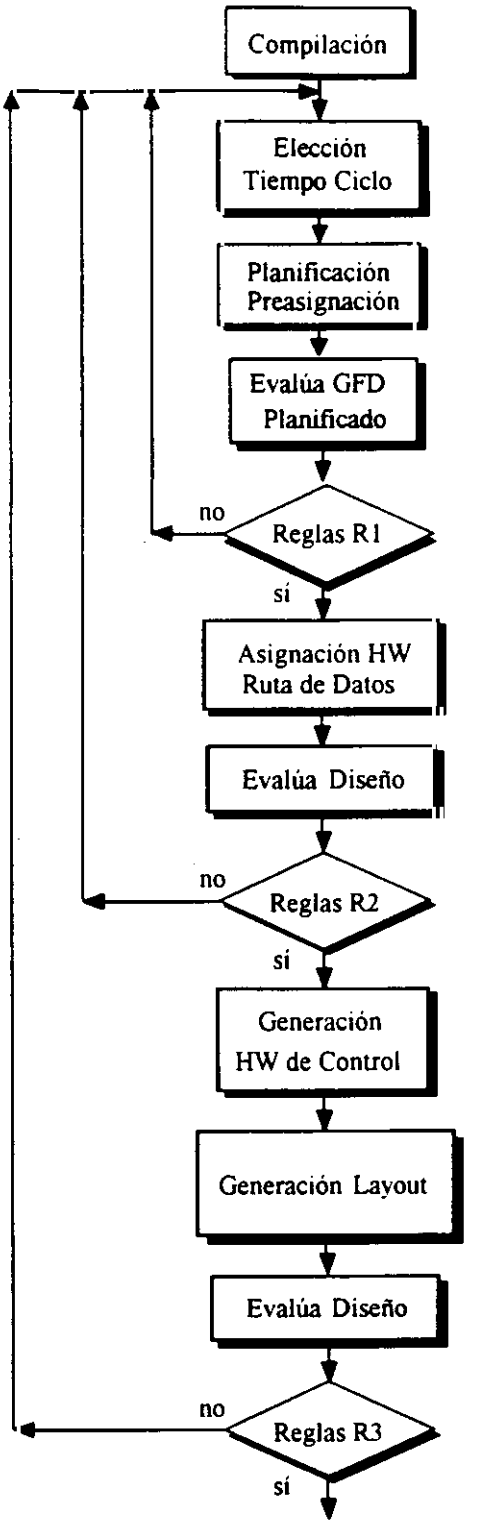


Figura 3.2 Proceso de Síntesis con FIDIAS

Estos dos lazos constituyen el cuerpo principal del sistema FIDIAS. Cuando se encuentra una solución válida, se pasa el diseño a un generador de módulos, que produce como salida un fichero VHDL o EDIF, lenguajes que sirven de fuente a herramientas CAD, para generar el layout final. Si durante este último paso se obtiene un circuito que no cumple los objetivos dados (debido a que las estimaciones realizadas no han tenido la suficiente precisión), es posible volver a realimentar el proceso total, calculando de nuevo los parámetros globales del sistema. Este último proceso, es decir la posibilidad de conectar con una herramienta CAD y volver a diseñar el circuito en caso necesario, obteniendo información del diseño físico para modificar los parámetros del sistema adecuadamente, es uno de los temas tratados en este trabajo, y será desarrollado en capítulos posteriores.

A continuación vamos a tratar brevemente las principales tareas del sistema FIDIAS.

3.3. Subtarea de planificación-preasignación

La primera fase dentro de esta tarea es la **selección del tiempo de ciclo**. Como vimos en el capítulo 1, éste es uno de los parámetros con más influencia en el tiempo total de ejecución del circuito (o rendimiento), ya que su valor viene dado por el producto de dicho tiempo de ciclo y del número de etapas de control. Esta selección es realizada por el módulo Experto [*Mozo92*].

El siguiente paso es llamar al módulo **planificador-preasignador**, que realiza la asignación de pasos de control a los operadores del GFD y al mismo tiempo decide qué tipo de UF va a implementarlos. Una vez realizada la planificación, el Experto de Diseño realiza **la evaluación del GFD planificado** mediante un conjunto de reglas R1 (ver figura 3.2) y decide si el resultado cumple las especificaciones dadas por el usuario o es necesario realizar de nuevo la planificación-preasignación. A continuación vamos a tratar cada una de estas fases por separado.

3.3.1. Selección del tiempo de ciclo

El sistema FIDIAS dispone como información de entrada de una biblioteca de módulos donde se tiene almacenado el retardo asociado a cada operador. Este retardo se supone que incorpora el retardo debido a la UF que lo implementa, el retardo de almacenamiento en un registro y el tiempo de propagación de las señales debido a interconexiones.

En primer lugar se calcula un tiempo de ciclo inicial a partir de estos datos, tomando como retardo "base" de cada operador del GFD, el retardo de la UF más rápida que puede implementarlo. Entonces se elige como tiempo de ciclo **tciclo** el retardo del operador más lento, de forma que todas las operaciones puedan realizarse en un solo ciclo.

Con *tciclo* se realiza la planificación ASAP y se evalúa el tiempo total de ejecución. Si no se cumplen los objetivos del usuario, hay que recalcular *tciclo*. La técnica se basa en la minimización de los **tiempos muertos** de los operadores, es decir, del tiempo que transcurre desde que termina de realizar la operación hasta que se termina el ciclo de reloj. El algoritmo utilizado computa los tiempos muertos de los operadores del circuito que se encuentren en el camino crítico, y calcula el valor medio. El nuevo valor de *tciclo* es el anterior menos dicho tiempo muerto medio.

Este algoritmo puede repetirse sucesivas veces, hasta que se encuentra el valor adecuado y como la complejidad es baja, O(N), donde N es el número de operadores del circuito, no se penaliza excesivamente el tiempo de ejecución global.

Además presenta dos ventajas fundamentales:

- Tiene en cuenta una biblioteca de UFs, con varias UFs para implementar cada operación.
- El tiempo de ciclo se computa teniendo en cuenta los operadores del camino crítico, que son los que tienen influencia sobre el tiempo total de ejecución.

Sin embargo presenta una serie de problemas:

- Supone que se dispone de información sobre los retardos, pero realmente esta información no es conocida. El retardo de una interconexión depende de su longitud, y ésta es un dato que no se puede calcular hasta que no se ha generado el layout del circuito. Además, no es posible tratar a los módulos como componentes con un retardo constante [JRDK94], sino que hay que tener en cuenta los retardos debidos a las interconexiones internas. Por lo tanto, sería necesario realizar una estimación de los retardos de las interconexiones (tanto internas como externas), considerando el circuito en particular, la tecnología de diseño y la biblioteca de módulos.

- Aunque utiliza una biblioteca de módulos, sólo tiene en cuenta para cada operador, la UF más rápida que pueda implementarlo. Dependiendo de los objetivos y/o restricciones del usuario, esta opción puede ser real o totalmente equivocada, con lo cual los resultados obtenidos pueden no ser los óptimos, y, en el peor de los casos, conducir a un valor del tiempo de ciclo para el cual todas las UFs sean multiciclo. Es necesario tener en cuenta tanto los objetivos del usuario, como la biblioteca de módulos a la hora de realizar esta selección.

- El camino crítico varía dependiendo del tiempo de ciclo. Por tanto, no puede utilizarse sólo el camino crítico para un cierto valor de *tciclo*, para calcular otro *tciclo*, sino que es necesario realizar un estudio global del GFD para obtener los posibles caminos críticos.

Estas, y otras desventajas que veremos posteriormente, revelan la necesidad de realizar una mejora de este algoritmo, incluyendo estimaciones sobre los retardos de interconexiones y, en general, sobre las características físicas del circuito. Por eso, uno de los objetivos de este trabajo de investigación es obtener un algoritmo mejorado de selección del tiempo de ciclo, que tenga en cuenta las características físicas del circuito y elabore un estudio global del GFD, de la biblioteca de módulos y de los objetivos del usuario.

3.3.2. Planificación-Preasignación

El sistema FIDIAS realiza la asignación de operaciones a pasos de control simultáneamente a la asignación de tipos de UFs, mediante una técnica de exploración exhaustiva del espacio de diseño [More95]. Dado que este algoritmo de Planificación-Preasignación conjunta es una de las aportaciones más novedosas dentro de este sistema, a continuación vamos a desarrollarlo un poco más en profundidad.

El algoritmo utiliza una serie de técnicas de acotación y guía basadas en información probabilística obtenida de un análisis del GFD, de la biblioteca de módulos y de los objetivos del usuario. Los resultados del análisis se cuantifican en unos factores, llamados **probabilidades**, asociados a cada alternativa de planificación y asignación de cada nodo del GFD (ver figura 3.3).

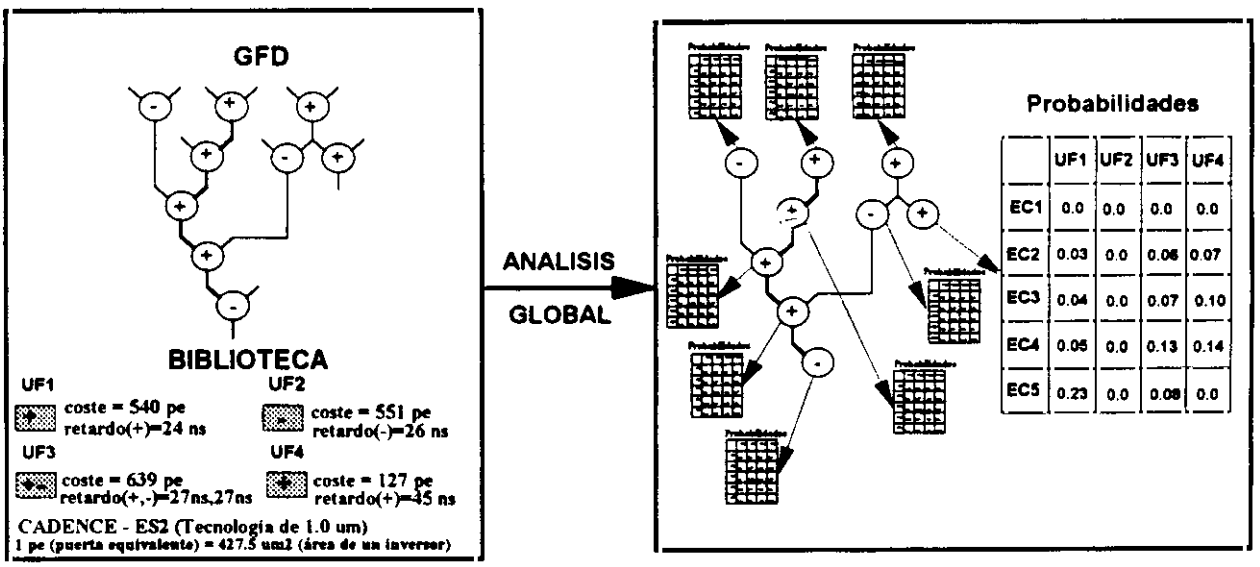


Figura 3.3 Probabilidades: resultado del análisis global del GFD y la biblioteca

A cada nodo N del GFD, para cada etapa de control EC y para cada UF de la biblioteca de módulos, se calcula $\mathbf{Prob_N(EC,UF)}$, como la probabilidad de que el nodo N sea simultáneamente planificado en la etapa de control EC y asignado a dicha UF. Estas probabilidades pretenden caracterizar la calidad de las asignaciones correspondientes.

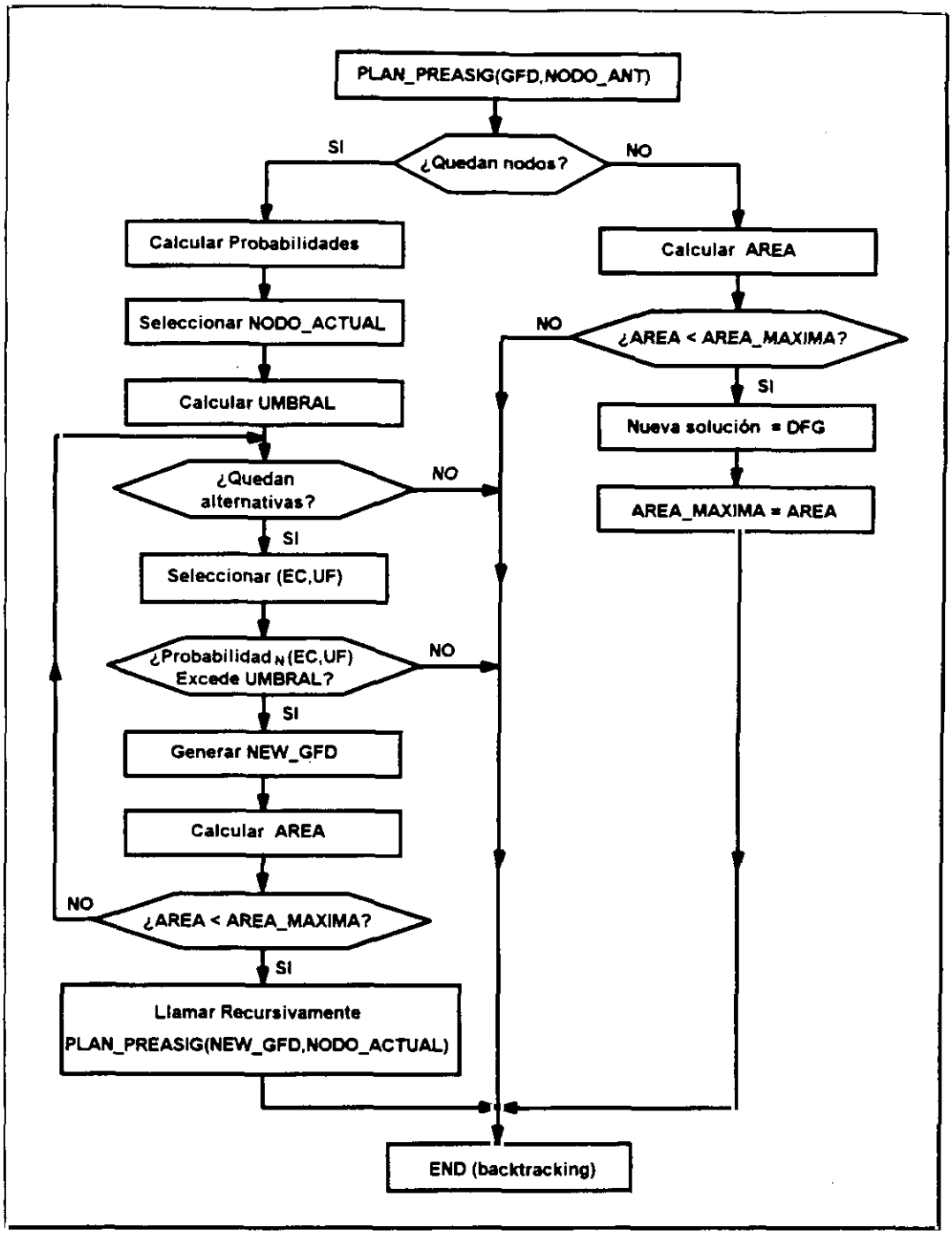


Figura 3.4. Diagrama de flujo del algoritmo de planificación-preasignación

El algoritmo de exploración del espacio de soluciones es de tipo ramificar y acotar, guiado por una serie de técnicas que acotan el número de soluciones exploradas, y guían la búsqueda, permitiendo obtener soluciones de buena calidad en tiempos de búsqueda moderados. En la figura 3.4 se muestra el diagrama de flujo de dicho algoritmo.

Las **técnicas de acotación** consisten en explorar sólo aquellas alternativas de cada nodo cuya probabilidad supera un cierto umbral, que depende de un factor de tolerancia dado por el usuario.

Las **técnicas de guía** indican el orden en que se seleccionan los nodos, y el orden en el que se seleccionan las distintas alternativas para cada nodo. El orden de selección de los nodos viene dado por una función de prioridad, que optimiza la reusabilidad del hardware y minimiza la reducción de alternativas para el resto de los nodos del GFD. Para cada nodo, las alternativas se seleccionan en orden decreciente de probabilidades.

3.3.3. Evaluación del GFD planificado

Una vez realizada la planificación-preasignación, por cualquiera de los dos métodos vistos anteriormente, el módulo Experto realiza una evaluación del GFD planificado con el objetivo de comprobar el cumplimiento de las restricciones del usuario.

El chequeo del área se realiza mediante una estimación del área mínima, a partir del GFD planificado y la biblioteca de módulos. Para realizar esta estimación, se calcula un número mínimo de registros, UFs e interconexiones necesarias para implementar el diseño.

El número mínimo de registros se calcula mediante un algoritmo modificado de REAL (algoritmo del borde izquierdo [KuPa87] y Apéndice B), que incorpora el tratamiento de condicionales y lazos.

Aunque el área de UFs se puede calcular fácilmente si se ha realizado la preasignación de éstas, el Experto dispone además de un algoritmo para el caso de que no se hubiera realizado esta preasignación (por ejemplo si se utiliza otro planificador). Este algoritmo se basa en el máximo paralelismo de cada uno de los operadores del GFD planificado y la UF más barata capaz de implementar cada uno de ellos.

También se estima el área mínima de interconexiones, calculando:

- El número mínimo de éstas, basado en el máximo número de entradas/salidas de UFs en un paso de control;
- El área de una interconexión, basándose en consideraciones geométricas que estudiaremos posteriormente.

Si el área total mínima estimada es superior a la restricción máxima impuesta por el usuario, el Experto realimenta el lazo de planificación-preasignación, recalculando de nuevo los parámetros necesarios. Una posible modificación es incrementar el número de pasos de control, si no se violan las restricciones de tiempo, para reducir el paralelismo y por tanto el área. Otra posibilidad es preasignar UFs más baratas, antes de realizar la siguiente iteración.

3.4. Subtarea de Asignación de Hardware

Como hemos dicho anteriormente, esta tarea tiene como función asignar UFs, registros e interconexiones a todos los operadores del GFD planificado. El módulo Asignador tiene en cuenta la información sobre UFs preasignadas que obtiene del GFD planificado, pero puede realizar también la asignación de UFs si éstas no han sido preasignadas.

El algoritmo de asignación debe recorrer el **EDA** (Espacio de Diseño de Asignación) útil, con el *fin de encontrar una solución que minimice cierta* ***función de coste***.

El tiempo de búsqueda de soluciones se puede disminuir guiando dicha búsqueda mediante heurísticas que modifican el orden en el cual se toman las diferentes opciones. Son las llamadas **heurísticas de guía**. Además, el tiempo de búsqueda global se reduce mediante las **heurísticas de acotación**, que limitan el EDA útil.

El algoritmo utilizado en FIDIAS [SMTH92] para recorrer el EDA útil es de tipo ramificar y acotar, como se muestra en la figura 3.5.

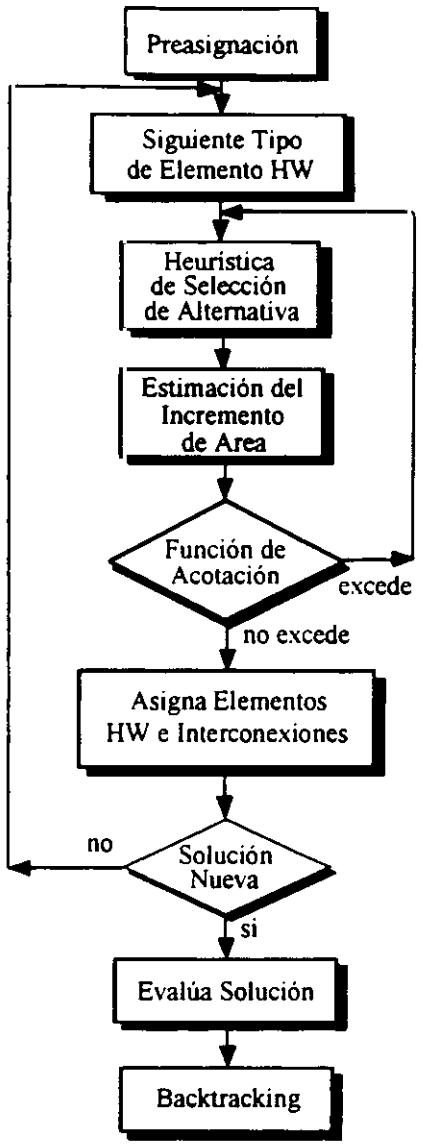


Figura 3.5. Modo de operación del asignador de hardware

Para cada tipo de elemento que se asigna, examina todas las alternativas posibles, primero tratando de reusar hardware y luego añadiendo nuevos módulos. El orden de selección de las alternativas se decide mediante las heurísticas de guía. Para cada alternativa se calcula el incremento en la función de coste que se produce, y mediante las heurísticas de acotación se decide si se puede elegir o no. Durante la asignación, también se incluyen medidas de testabilidad del circuito [OTMS93] y [OTMM94]. A continuación vamos a presentar brevemente la reducción del EDA, las heurísticas de guía y acotación, la función de coste utilizada y las medidas de testabilidad mencionadas.

3.4.1. Técnicas de reducción del EDA

Estas técnicas permiten disminuir el EDA, para obtener lo que hemos llamado EDA útil, con lo que se reduce también el tiempo de búsqueda del diseño óptimo. Se basan principalmene en la **preasignación de módulos.** Como la creación de un nuevo módulo sólo se realiza cuando se han tratado ya todas las posibilidades de reuso de hardware, los primeros módulos que se crean en un diseño suelen reutilizarse más que los módulos que se crean en las últimas fases. Esto produce una concentración de interconexiones alrededor de ciertos módulos y un crecimiento del número y tamaño de los multiplexores. Para evitar este hecho, se realiza una preasignación de módulos, que permite equilibrar las interconexiones y optimizar la reutilización del hardware.

Para los registros se preasigna el número mínimo estimado en la fase de validación del GFD planificado, explicada anteriormente. La preasignación de buses se realiza mediante un análisis del máximo paralelismo del GFD planificado en un paso de control.

3.4.2. Función de coste

La función de coste [SMHT91] debe garantizar que los diseños que la minimizan son los que tienen menor área; por lo tanto esta función debe ser una **estimación** lo más precisa posible del **área** final del diseño. Esta estimación debe ser fiel, es decir, que garantice que si un diseño se supone que es mejor que otro, realmente lo sea.

Por otra parte, para cada alternativa posible se calcula el incremento en la función de coste que produce, y mediante las heurísticas de acotación se decide si se elige o no. Este incremento en la función de coste debe ser el incremento en área de la alternativa elegida.

Uno de los puntos más importantes de este algoritmo de asignación es, por tanto, la estimación de área, que presentamos a continuación.

3.4.2.1. Estimación de área

La estimación del área en el sistema FIDIAS no se realiza para un diseño completo, sino sólo para una decisión, cada vez que se debe optar por una de las posibles alternativas. La suma de todos los incrementos de área de las sucesivas decisiones que se toman para un diseño, es el área total de éste. A continuación vamos a ver las distintas estimaciones de área según las posibles alternativas.

Por un lado, si la decisión es de creación o adición de nuevo hardware al diseño, el coste parcial del mismo se verá incrementado en el coste de dicho elemento o módulo hardware. Esto es lo que sucede, por ejemplo, para registros o UFs. Para todos los módulos el coste que se considera es equivalente al área de silicio que ocupa el módulo, que se supone almacenado en la biblioteca.

Por otro lado, tanto si la decisión ha sido de creación de un nuevo elemento hardware como de reuso de un elemento ya asignado, es necesario estimar cómo influye esta decisión en el interconexionado. Si no se crea una interconexión nueva, debido a que los elementos fuente y destino estaban ya conectados, esta influencia es nula. Si la interconexión no existe previamente, es necesario crearla, y evaluar el incremento del coste parcial que va a suponer dicha creación. En este caso, se debe calcular el incremento del coste que puede suponer la adición de nuevos multiplexores, el aumento del número de entradas de los ya existentes y el trazado de las nuevas conexiones. El coste de los primeros es también el coste en área de silicio que ocupan, dependiendo dicha área del número de entradas del multiplexor, y se supone almacenado en la biblioteca. Sin embargo el coste de la interconexión debe ser estimado.

Para saber si una interconexión dada va a ser más cara o barata que otras interconexiones (es decir, más larga o más corta), es necesario conocer la disposición física de los módulos que intervienen en cada una de ellas. Dado que a este nivel es imposible conocer esta disposición, lo que se ha realizado en el sistema FIDIAS [SMHT91] es una estimación de la longitud

media de las interconexiones, basándose en métodos geométricos y estadísticos, como los presentados en [PrLo88].

Se ha considerado el diseño como un cuadrado compuesto de N*N módulos, cada uno de ellos con un área promedio A_p. El área total del diseño A_{total} vendrá dada entonces por:

$$Atotal = Ap * N * N$$

La distancia media D_m entre dos módulos cualesquiera, es proporcional al lado del cuadrado L_c ($L_c=\sqrt{Atotal}$), con un factor de proporcionalidad igual a 2/3 [PrLo88]. Este cálculo se ha realizado suponiendo que la distancia entre dos elementos está compuesta por un tramo horizontal y otro vertical. Por tanto tenemos:

$$Dm = \frac{2}{3}\sqrt{Atotal} = \frac{2}{3}\sqrt{Ap} * N$$

La longitud de una interconexión promedio $\mathbf{L_{int}}$ se toma igual a esta distancia promedio Dm entre módulos.

Por otra parte, el tiempo de ciclo debe ser suficiente largo para garantizar que se puede realizar la transmisión a lo largo de la interconexión. El retardo de una interconexión se considera fundamentalmente debido a la resistencia total de la interconexión **R** y a las capacidades de entrada a los módulos que conecta **C**, utilizando un modelo simple RC La resistencia es directamente proporcional a la longitud e inversamente a la anchura de la interconexión. Dado un cierto C, para poder garantizar un valor constante del retardo independiente de la longitud de las interconexiones, la resistencia entre los extremos de las mismas debe mantenerse igual o por debajo de un valor máximo, que vendrá determinado por el tiempo de ciclo y por las capacidades parásitas. Por esta razón, la anchura mínima de la interconexión W_{int} deberá ser proporcional a su longitud total L_{int}:

$$W\,int = Ka * L\,int$$

En donde K_a, que determina la relación entre la anchura y la longitud de la interconexión, depende de la tecnología y de la capacidad C. Finalmente, el incremento del coste parcial del diseño debido al coste de la interconexión A_{int}, será el de la superficie ocupada por la misma, es decir, el producto de la longitud por la anchura.

$$A\,int = Ka\left[\frac{2}{3}\sqrt{Ap} * N\right]^2 = K * Ap * N^2$$

donde K es una constante que incluye el factor 2/3 y la constante Ka.

Se puede observar que el tamaño de la interconexión es proporcional al número de módulos del diseño, y al valor promedio del área de éstos. Por tanto, el valor medio de una interconexión debe recalcularse cada vez que el módulo asignador obtenga un nuevo diseño, con los datos actualizados sobre el número y tamaño de los módulos. En [Sept91] puede encontrarse un desarrollo más preciso sobre este método de estimación.

Si bien este método de estimación de área ha tratado de tener en cuenta consideraciones físicas del circuito, como son el tamaño del circuito final, características tecnológicas etc., se han realizado una serie de suposiciones imprecisas. En primer lugar se considera fija el área de los módulos almacenada en la biblioteca. Esto ya hemos dicho que no es posible, puesto que los módulos no pueden suponerse componentes con un área constante [JRDK94]. Es necesario estimar también el área de los módulos, teniendo en cuenta sus interconexiones internas. Por otra parte, se ha supuesto que las interconexiones se realizan en una única capa, y normalmente se utilizan dos, una para las horizontales y otra para las verticales (ver Apéndice A). Veremos que no es necesario tener en cuenta los tramos verticales para estimar el incremento en área del circuito. También se ha supuesto que el ancho de una interconexión es variable, y, salvo algunas excepciones como la tierra, alimentación y algunas otras determinadas por el diseñador, el resto tienen una anchura fija. Por último no se ha tenido en cuenta el estilo de diseño utilizado, que como se muestra en el Apéndice A influye notablemente en los resultados finales.

En general, asumiendo estas suposiciones, y utilizando celdas estándar, arrays de puertas o macroceldas para generar los diseños, los resultados de las áreas estimadas de los circuitos son mucho mayores que las reales, y además no son fieles, puesto que sobrestiman el área de interconexionado frente a la del resto de los módulos. Todas estas consideraciones y los resultados obtenidos, nos llevan a la conclusión de que es necesario realizar un estudio más profundo sobre las características físicas del circuito para poder realizar estimaciones de área. En el próximo capítulo veremos qué otros factores hay que tener en cuenta para obtener resultados más fieles y precisos.

3.4.3. Heurísticas de Acotación

Ademas de estimar el incremento en área del circuito, cada vez que se estudia una nueva alternativa es necesario decidir si se selecciona o no. Las heurísticas de Acotación [SMTH92], nos permiten reducir el tamaño del EDA, eliminando aquellas alternativas que no van a conducir a un diseño óptimo. Existen tres técnicas principales de acotación:

- **Acotación global**. Cada vez que se toma una alternativa, se chequea el coste del diseño parcial obtenido hasta el momento con el valor del coste total de la última solución encontrada, que coincide con el área del mejor diseño obtenido hasta el momento. Como valor inicial, antes de encontrar la primera solución, se toma el valor máximo de área dado por el usuario. Esta heurística es muy poco restrictiva, por lo tanto recorre una gran parte del EDA útil. Esto permite encontrar siempre el diseño óptimo, pero utilizando mucho tiempo de búsqueda.

- **Acotación distribuida**. Para cada etapa de control, se tiene un valor de acotación igual al coste parcial obtenido para dicha etapa para el mejor diseño encontrado hasta el momento. Esta acotación es muy restrictiva, y tiene el peligro de caer en mínimos locales, con lo cual no siempre es posible encontrar la solución óptima.

- **Acotación distribuida con margen**. Esta heurística utiliza también el coste parcial de cada etapa de control, pero, con el objetivo de escapar de mínimos locales, le añade un cierto margen a cada etapa. Este margen se calcula para cada etapa de control, y su valor depende del incremento del coste entre la etapa anterior y la actual, y de un parámetro de proporcionalidad dado por el usuario. De esta forma, el usuario decide la capacidad de escape de los mínimos locales, según quiera incrementar o no el tiempo de búsqueda de soluciones. Esta heurística es la que mejores resultados consigue para tiempos moderados de búsqueda.

3.4.4. Heurísticas de guía

Permiten decidir el orden de selección de alternativas, con el fin de dirigir la búsqueda hacia la solución óptima. Es lo que se denomina la reordenación del espacio de diseño. Existen dos heurísticas:

- **Reordenamiento de registros y buses**. Se ordenan las alternativas de reuso según el orden en el que producen menos incremento de coste. Así no se reusan siempre los mismos módulos (los que se crearon al principio del diseño) y se equilibran las interconexiones.

- **Reordenamiento de UFs**. Se ordenan las alternativas según un estudio del GFD planificado y de la biblioteca de UFs, midiendo la compatibilidad de operadores y el posible reuso de UFs.

3.4.5. Medidas de testabilidad durante la asignación de hardware

Una de las más recientes líneas de investigación que se está desarrollando e integrando en el sistema FIDIAS es la introducción de medidas de testabilidad de circuitos durante la fase de asignación de hardware [OTMS93] y [OTMM94].

Debido a la creciente complejidad de los circuitos VLSI, se hace necesario mejorar la testabilidad de los mismos para poder garantizar su calidad. La introducción de testabilidad en circuitos resulta ser más efectiva y barata si se realiza durante la fase de diseño y no tras ella. De ahí la importancia de su relación con la síntesis de alto nivel.

La técnica utilizada para el desarrollo de *circuitos testables* ha sido la metodología de diseño BIST ("Built-In Self-Test"). Los circuitos BIST son capaces de generar su propio test y analizar la salida del mismo, características muy convenientes para circuitos sintetizados automáticamente. A cambio de facilitar el test, las técnicas de testabilidad introducen en el circuito retardos adicionales y aumentan su área, y como consecuencia es necesario alcanzar un compromiso entre la testabilidad y otras ligaduras del diseño.

La técnicas de testabilidad en FIDIAS se integran en la tarea de la asignación de hardware. Esta integración se realiza introduciendo estimaciones de testabilidad junto con las estimaciones de área de los circuitos. A partir de estas estimaciones, se definen nuevas funciones de acotación de área y de testabilidad que deciden qué ramas del árbol de diseño no conducen a diseños aceptables para las ligaduras.

3.4.6. Evaluación del diseño

Una vez que el Asignador de Hardware ha concluido su tarea, el Experto evalúa el diseño resultante. Como ya se conoce el número y tipo de módulos del circuito, la estimación de área es similar a la realizada durante la asignación de hardware. El valor del área de una interconexión se recalcula, utilizando los datos del circuito obtenido. El valor del resto de los módulos se toma de la biblioteca de módulos. La estimación de tiempo se realiza de forma análoga a cómo se hizo después de la planificación. Si se han cumplido los objetivos, el diseño, junto con la UC que genera el generador de Control, se pasa a un lenguaje VHDL/EDIF, que puede servir de entrada a una herramienta CAD para generar el layout. Si no es así, debe realimentarse el proceso al comienzo del algoritmo, calculando de nuevo los

parámetros de entrada a los módulos. Por ejemplo, se puede incrementar el número de pasos de control, si es posible, para reducir el paralelismo [Mozo92].

3.5. Generación del control

La Unidad de Control [Mend93] puede realizarse de dos formas diferentes: microprogramada o cableada (FSM). La unidad de control microprogramada responde a un modelo segmentado de dos etapas (ejecuta microoperación actual y calcula microinstrucción siguiente en paralelo), con un formato de microinstrucción horizontal. La unidad de control cableada corresponde a una máquina de estados finita de tipo Mealy.

3.6. Conclusiones

A lo largo de esta exposición hemos visto de nuevo la importancia de las estimaciones durante el proceso de SAN. Se ha presentado un sistema que trata de tener en cuenta el retardo de las interconexiones para calcular el tiempo de ciclo, y, por tanto, su influencia en el tiempo total de ejecución del algoritmo. También trata de estimar el área del interconexionado durante la asignación de hardware y al final de ésta. Aunque este sistema de SAN se caracteriza por la atención que se ha prestado a las estimaciones de las características físicas, la metodología utilizada tiene varias desventajas:

- Se supone que en la biblioteca de módulos existe información sobre el área y retardo de los módulos. Esto sólo es posible si los módulos son Macro-celdas con una forma y tamaño definidos (ver Apéndice A) o sólo están formados por una celda estándar. Pero si cada módulo está formado por varias celdas conectadas entre sí, la longitud de las interconexiones internas del módulo no se puede conocer hasta que no se ha generado el layout, y por tanto no es un dato disponible. De la misma forma, el

retardo de las interconexiones externas (entre módulos) no es un dato conocido a priori.

- Las suposiciones que se realizan para estimar el área de una interconexión tampoco son totalmente correctas. No se han tenido en cuenta la forma de colocar los módulos de los algoritmos de colocación de módulos, ni tampoco el número de capas que se utilizan para realizar las interconexiones. En resumen, no se ha considerado la metodología de trabajo de las herramientas de diseño de circuitos ni la tecnología utilizada.

De este modo, la conexión de la salida del sistema de SAN con herramientas CAD surge como necesidad natural, no sólo para terminar el proceso de diseño y llegar a la generación del layout, sino también para obtener información sobre la metodología de trabajo de estas herramientas y sobre la tecnología utilizada, y mantener un flujo de intercambio de información en ambos sentidos. A partir de esta información es posible mejorar las estimaciones realizadas durante la SAN. Además, como ventaja final, puede realizarse la evaluación del diseño real, de forma que, si no se cumplen los objetivos deseados, se puede volver al inicio del proceso, pero ahora disponiendo de mucha más información sobre las características físicas del circuito. Esto nos permitirá mejorar la fidelidad de estimaciones de las características físicas. De esta forma, se puede guiar mejor el siguiente proceso y aumentar la precisión de los resultados obtenidos en las siguientes fases del diseño.

En el próximo capítulo se presentará la conexión del sistema FIDIAS con una herramienta CAD. Esta herramienta utiliza celdas estándar para realizar los diseños, y todos los estudios sobre área y retardos de este trabajo se referirán a este estilo de diseño.

A partir de dicha conexión se verá la forma de mejorar las estimaciones de área del circuito, de forma que sin ser más complejas sean mucho más fiables y precisas. Estas estimaciones de área se pueden utilizar en cualquiera de los tipos de algoritmos de asignación de hardware presentados en el capítulo 1, es decir, en los iterativos, en los constructivos y en los

constructivo-iterativos. Como ejemplo de integración en un sistema, se verá cómo estas estimaciones se han incluido en el sistema FIDIAS, cuyo algoritmo de asignación podemos decir es de tipo constructivo-iterativo, ya que necesita realizar estimaciones tanto durante la creación de un diseño, como al final del mismo.

En los capítulos 5 y 6, y basándose en los resultados obtenidos en el capítulo 4, se realizarán las estimaciones del retardo de las interconexiones. A partir de estos retardos, se presenta un algoritmo mejorado de selección del tiempo de ciclo, que tiene en cuenta todos los factores que influyen en el correcto funcionamiento del circuito.

Resumiendo, a lo largo de los próximos capítulos, se presentarán todas las consideraciones que son necesarias sobre la influencia de las características físicas en los diseños, primero sobre el área y luego sobre el tiempo, asegurando así el correcto funcionamiento de los circuitos diseñados y la consecución de los objetivos del usuario, si ésta es posible.

Capítulo 4

Estimaciones de Area en SAN

4.1. Introducción

A lo largo de los capítulos anteriores se ha visto que en un proceso de SAN se deben tomar una serie de decisiones teniendo en cuenta las características físicas del circuito. En particular, durante la asignación de hardware, es necesario estimar el área de los módulos y de las interconexiones del circuito. En el sistema FIDIAS, que realiza una exploración del EDA (Espacio de Diseño de Asignación) mediante un algoritmo de tipo constructivo-iterativo, estas estimaciones se tienen que hacer en dos momentos diferentes:

- Durante la generación de un diseño, cuando el número y tipo de módulos no se conocen totalmente, para poder elegir entre diferentes opciones la que conducirá al diseño con área mínima. El problema para realizar esta estimación es que la información que se tiene sobre el diseño final no es completa.

- Cuando se ha terminado de generar un diseño, para calcular el área global y decidir entre varios diseños cuál es el que cumple mejor las restricciones y objetivos del usuario.

Algunos autores suponen que el área de los módulos puede saberse fácilmente si se dispone de una biblioteca de módulos, como en [Pang88], [DeNe89] y en [JaPP92]. Esto sólo será cierto si se utilizan macroceldas con una estructura definida (Apéndice A) para implementar cada uno de los módulos. Pero como ya hemos visto, la mayoría de las herramientas de SAN que realizan alguna estimación de las características físicas, suponen el diseño generado con celdas

estándar. En este caso, el área de las celdas que forman los módulos es constante, pero, debido al interconexionado externo del módulo, cuando se colocan las celdas en el circuito aparecen dos factores que hay que tener en cuenta:

- Las posiciones relativas de las celdas cambian;
- El ancho de los canales necesario para realizar el interconexionado aumenta, como se explica en el Apéndice B.

Estos dos factores provocan que la longitud de las interconexiones internas del módulo, y por tanto el área final de éste, sea diferente cuando se coloca el módulo en un circuito que cuando se tiene aislado.

Por lo tanto necesitamos elaborar una estimación del área de las interconexiones tanto internas como externas del módulo. Y esta estimación debe realizarse en dos situaciones:

- La estimación del incremento en área que supone la creación de cada una de las interconexiones sin conocer todavía el número final de módulos y de interconexiones, basándonos en qué tipo de módulos va a interconectar, en el conocimiento de la forma de trabajo de los algoritmos de colocación e interconexionado de módulos y en la tecnología de diseño.
- La estimación del área final del circuito sin necesidad de generar el layout ni de realizar cálculos complicados. Esto será posible utilizando la estimación del área de una interconexión para afinar la estimación del área que van a ocupar los módulos en el diseño final.

En el capítulo 1 vimos las características que debían tener estas estimaciones. En primer lugar los algoritmos de estimación deben tener una **complejidad baja**, puesto que es necesario utilizarlos muchas veces durante el proceso de diseño. En segundo lugar deben ser **fieles**, para permitir dirigir el proceso de diseño a la obtención del circuito con área mínima. Y por último deben ser los suficientemente **precisas** para distinguir si un circuito cumple o no las

restricciones del usuario.

A lo largo del capítulo 2 se presentaron distintas técnicas de estimación de área de CIs. Las técnicas más precisas se realizaban a nivel de celdas estándar (o arrays de puertas), y su complejidad era demasiado elevada para poderlas integrar dentro de un proceso de SAN. Sin embargo, vimos que su precisión viene dada por el hecho de que tienen en cuenta la tecnología de diseño y la forma de trabajo de los algoritmos de diseño de CIs. Por esta razón vamos a comenzar nuestro proceso de búsqueda de métodos de estimación con un análisis de estos dos últimos factores. Puesto que estos algoritmos son dependientes del estilo de diseño y de la herramienta de composición de circuitos integrados, nos centraremos en un estilo de diseño, las celdas estándar, y en una herramienta en particular, CADENCE, que conectaremos a nuestro sistema de SAN.

4.2. Conexión con una herramienta de SBN

La conexión de la herramienta de SAN con una herramienta de composición de circuitos integrados, aparte de permitir la conclusión del proceso de diseño con la generación del layout, se hace imprescindible en el marco de este trabajo para poder estudiar cómo trabajan las herramientas de diseño y poder evaluar la calidad de nuestras predicciones. La realización de los experimentos sólo es viable mediante la creación de un sistema para automatizar en gran medida la interacción con la herramienta de diseño, puesto que la interacción manual, a base de teclado y ratón, consume una gran cantidad de tiempo.

La herramienta de diseño elegida fue CADENCE, ampliamente utilizada para el diseño de CIs, que emplea celdas estándar y macroceldas para generar el layout de los circuitos. El proceso de diseño total, desde la descripción del comportamiento del diseño hasta la generación del layout se muestra en la figura 4.1.

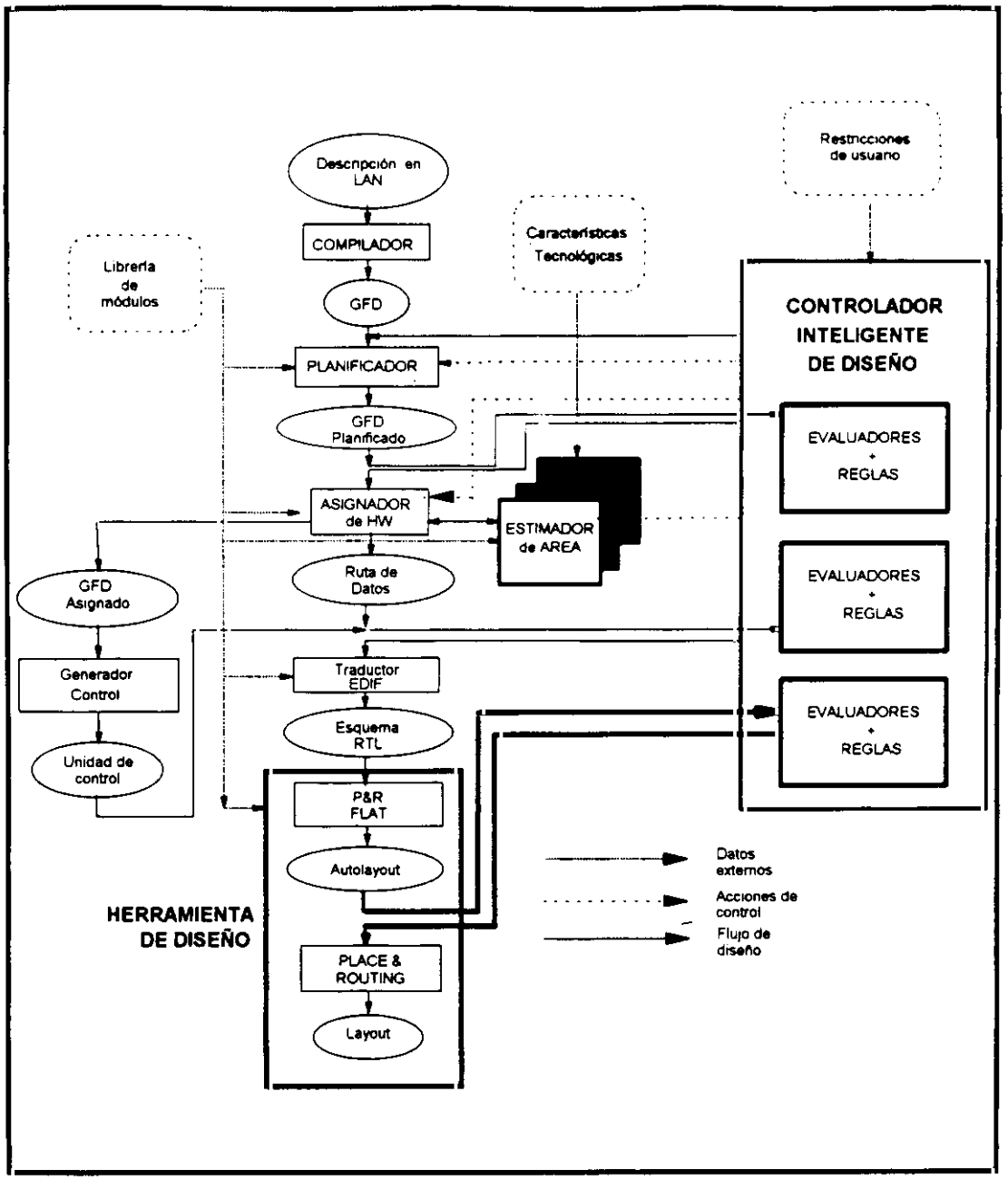


Figura. 4.1 Proceso de diseño global

Aunque en el Apéndice C se realiza un estudio exhaustivo de esta conexión, aquí resaltaremos las características principales (figura 4.2).

Los componentes que intervienen en cualquier ruta de datos generada por la herramienta de

SAN se encuentran diseñados previamente en una o varias bibliotecas propias de la herramienta de SBN (en la figura 4.2 son las que se nombran biblioteca 1..N). Podemos encontrar, entre otros, sumadores, multiplicadores, divisores, registros y multiplexores. Como hemos dicho, vamos a utilizar celdas estándar, por eso estos módulos están diseñados jerárquicamente a partir de otros más sencillos, que también se encuentran en la biblioteca, hasta llegar al nivel de celdas estándar. Ésta últimas se encuentran en una biblioteca propia de CADENCE, y en nuestro caso se trata de la biblioteca de ES2.

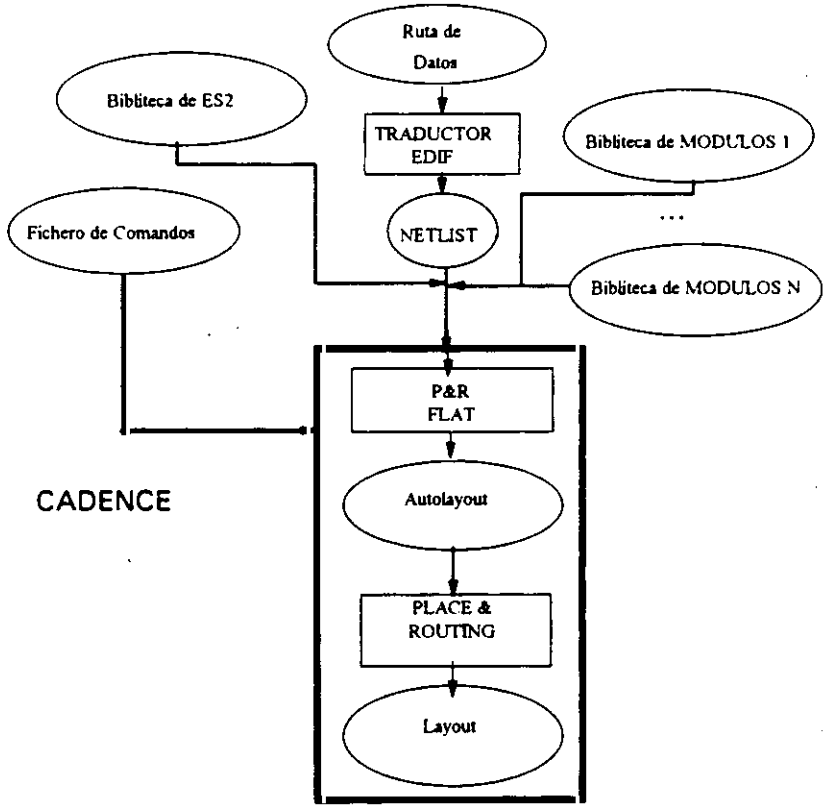


Figura 4.2 Conexión de una herramienta de SAN con CADENCE

La salida RTL del sistema de SAN se traduce a un diseño en EDIF (Electrical Design Interchange Format), donde se indica las bibliotecas que tienen módulos presentes en el diseño y los interfaces de estos módulos, seguido de una lista con el conjunto de instancias de módulos y sus interconexiones. Este formato sirve de entrada a CADENCE para generar el diseño en una configuración propia, que en principio es jerárquico. Es importante que todos

los tipos de módulos que aparecen instanciados, se encuentren diseñados hasta el nivel de celdas en una de las bibliotecas dadas en el fichero EDIF. En el Apéndice B se presenta el aspecto general de una descripción en este formato.

Las herramientas CAD para celdas estándar disponen de algoritmos que realizan cada una de las fases de SBN (Apéndices A y B). Sin embargo, el proceso de diseño total necesita cierta intervención manual, para dar la entrada de ciertos comandos que, ordenadamente, van ejecutando cada una de las fases de diseño. Para automatizar todo el proceso de colocación e interconexionado de módulos se elabora un fichero con todos los comandos necesarios en un lenguaje interno de CADENCE, denominado SKILL [SKILL 93]. De esta forma, a partir de la lista de componentes e interconexiones y recibiendo como entrada el fichero de comandos, CADENCE produce la colocación e interconexionado de módulos sin necesidad de ninguna intervención manual. El proceso de diseño total está así automatizado.

Los algoritmos de estimación deben tener en cuenta tanto las características de la tecnología utilizada, como las técnicas de colocación e interconexionado de módulos, para poder hacer estimaciones fiables. La validación de tales estimaciones sólo sería posible gracias a la automatización del proceso de generación de layout. Además, una vez que todo el proceso es automático, es posible generar un gran número de diseño en tiempos razonables, y a partir de ellos sacar conclusiones. A lo largo de este capítulo veremos estos resultados.

4.3. Características tecnológicas de un diseño con celdas estándar

Un diseño generado con celdas estándar está formado por una serie de filas de celdas, todas ellas de la misma longitud (L_{fila}) y altura, y una serie de canales entre las filas, para las interconexiones (para más información ver Apéndice A). Todas las celdas estándar tienen la misma altura, que es igual que la de la fila.

Normalmente existen dos capas de metal para realizar las interconexiones. La capa inferior,

que es la que discurre por los canales, se utiliza para las interconexiones horizontales (paralelas a las filas). La mayoría de las conexiones verticales se generan en la capa superior. De esta forma no se producen cruces de dos interconexiones en un mismo punto.

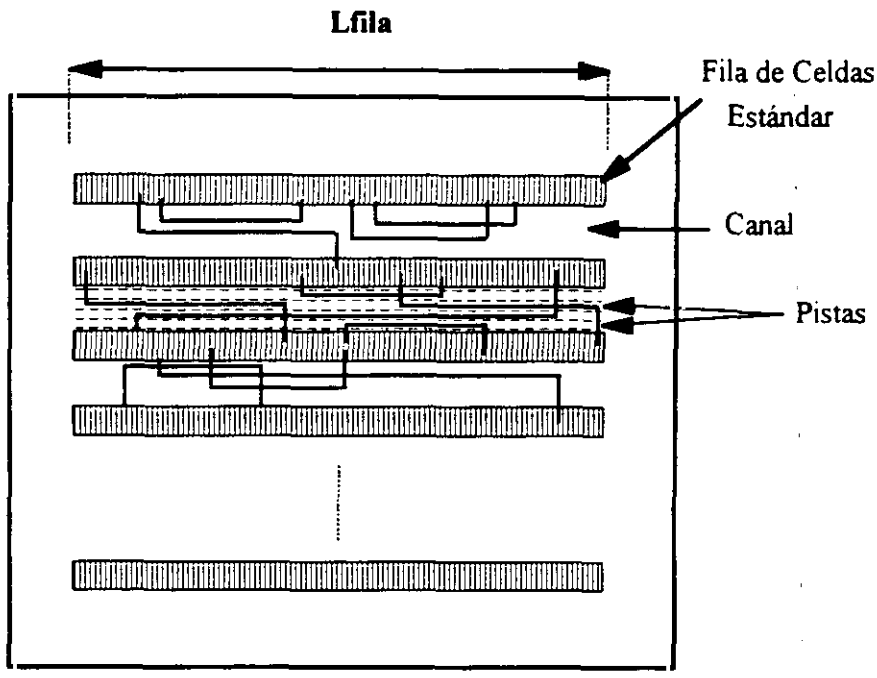


Figura 4.3 Diseño con celdas estándar

Cada canal tiene un número variable de pistas, paralelos a las filas de celdas estándar, y en cada una de las pistas pueden ir una o varias interconexiones, siempre que no se solapen. En la figura 4.3 tenemos un esquema de un diseño con celdas estándar.

Aunque casi todas las pistas tienen el mismo ancho, que viene dado por la tecnología, existen pistas especiales para las interconexiones que conectan un elevado número de celdas, como por ejemplo las de tierra y alimentación. Para asegurar el correcto funcionamiento eléctrico del circuito necesitan ser más o menos anchas, dependiendo del número de celdas que conectan, y normalmente su anchura viene tabulada según rangos. Los algoritmos de colocación e interconexionado deben consultar dicha tabla para reservarles el espacio suficiente. El resto de las interconexiones, si el diseñador no especifica lo contrario, tienen la misma anchura.

Vamos a considerar que la anchura de una interconexión estándar viene dada por la suma de la anchura de metal más la anchura mínima necesaria entre dos pista, que también viene dada por la tecnología. Ambos datos son conocidos y están almacenados en la biblioteca de módulos.

A partir de estos datos es posible realizar una primera estimación de las longitudes de las interconexiones, que será un promedio de entre todos los posibles valores.

4.4. Modelo simple: Estimación de la longitud de una interconexión media

Un diseño generado por una herramienta de SAN viene dado por un conjunto módulos y sus interconexiones. Si el diseño se implementa con celdas estándar, en la biblioteca de módulos se puede almacenar información precisa sobre el número y área de celdas estándar y el número de interconexiones intermas de cada módulo. A partir de esta información se puede calcular el número total de celdas e interconexiones del circuito.

El tipo de tecnología más usual en diseño de ASICs utiliza dos capas de metal para realizar las interconexiones, una para las interconexiones horizontales (***metal1***) y otra para las verticales (***metal2***). En la capa 1 se encuentran las filas de celdas estándar y una serie de canales entre ellas por donde se realizan las interconexiones (ver figura. 4.3). La capa de *metal2* sólo se utiliza para hacer interconexiones, por lo tanto hay espacio suficiente y no nos preocupamos de la influencia en el área final. Sólo hay que considerar los tramos horizontales de las interconexiones.

Este modelo es adecuado, puesto que es el que utilizan la mayoría de las herramientas CAD para celdas estándar y en especial el que se utiliza en CADENCE.

Sea una fila de celdas estándar de longitud L_{fila}. Cada una de las celdas tiene una longitud media L_{celda} y en total hay un número de celdas igual a N_{celdas} (figura 4.4).

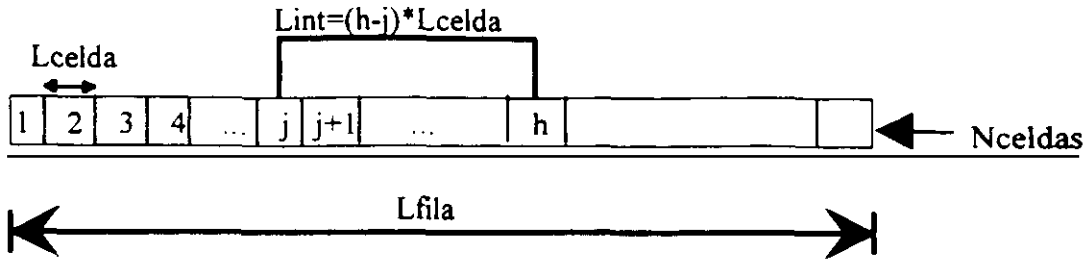


Figura 4.4 Fila de celdas estándar

Suponiendo equiprobables las conexiones entre cualquier par de celdas, la distancia media de una interconexión entre dos celdas de una fila (L_{media}) se calcula sumando la longitud de todas las diferentes interconexiones (L_{int}) y dividiendo entre el número total de interconexiones (N_{total}).

$$Lmedia = \frac{\sum_{n=1}^{Ntotal} Lint_n}{Ntotal}$$

Una interconexión entre la celda de posición j y la celda de posición h ($h>j$), tendrá una longitud horizontal dado por:

$$Lint = (h - j) * Lcelda$$

Como hay que sumar todas las posibles interconexiones, j puede variar desde 1 hasta N_{celdas}-1, y, para cada valor de j, h puede variar desde $j+1$ hasta N_{celdas}.

$$Lmedia = \frac{\sum_{j=1}^{Nceldas-1} \sum_{h=j+1}^{Nceldas} (h - j) * Lcelda}{\sum_{j=1}^{Nceldas-1} \sum_{h=j+1}^{Nceldas} 1}$$

Recordemos que la fórmula de una serie aritmética viene dada por

$$\sum_{n=n1}^{n2} n = \frac{n1+n2}{2}(n2-n1+1)$$

Si resolvemos los sumatorios en *h* utilizando esta fórmula nos queda:

$$Lmedia = \frac{\sum_{j=1}^{Nceldas-1} Lcelda * \left[\left[\frac{Nceldas + j + 1}{2}(Nceldas - (j+1) + 1)\right] - [j * (Nceldas - (j+1) + 1)]\right]}{\sum_{j=1}^{Nceldas-1} Nceldas - (j+1) + 1}$$

Simplificando y sacando factor común:

$$Lmedia = \frac{\sum_{j=1}^{Nceldas-1} Lcelda * \left[\left[\frac{Nceldas + j + 1}{2} - j\right] * (Nceldas - j)\right]}{\sum_{j=1}^{Nceldas-1} Nceldas - j}$$

$$Lmedia = \frac{\sum_{j=1}^{Nceldas-1} Lcelda * \left[\left[\frac{Nceldas - j + 1}{2}\right] * (Nceldas - j)\right]}{\sum_{j=1}^{Nceldas-1} Nceldas - j}$$

Sacamos las constantes fuera de los sumatorios:

$$Lmedia = \frac{1}{2} Lcelda * \frac{\sum_{j=1}^{Nceldas-1} (Nceldas^2 + j^2 - 2 * Nceldas * j) + \sum_{j=1}^{Nceldas-1} (Nceldas - j)}{\sum_{j=1}^{Nceldas-1} Nceldas - j}$$

Volvemos a simplificar:

$$Lmedia = \frac{1}{2} Lcelda * \left[\frac{\sum_{j=1}^{Nceldas-1} (Nceldas^2 + j^2 - 2 * Nceldas * j)}{\sum_{j=1}^{Nceldas-1} Nceldas - j} + 1\right]$$

Podemos utilizar la siguiente fórmula [Spie70]:

$$\sum_{n=1}^{N} n^2 = \frac{N(N+1)(2N+1)}{6}$$

Utilizando las fórmulas para cada sumatorio tenemos:

$$Lmedia = \frac{1}{2} Lcelda * \left[\frac{(Nceldas^2 * (Nceldas-1) + \frac{(Nceldas-1)(Nceldas-1+1)*(2(Nceldas-1)+1)}{6} - 2\,Nceldas * \frac{Nceldas-1+1}{2}(Nceldas-1))}{Nceldas*(Nceldas-1) - \frac{Nceldas-1+1}{2}(Nceldas-1)} + 1 \right]$$

Dividimos arriba y abajo por $(N_{celdas} - 1)$

$$Lmedia = \frac{1}{2} Lcelda * \left[\frac{(Nceldas^2 + \frac{(Nceldas-1+1)*(2(Nceldas-1)+1)}{6} - 2\,Nceldas * \frac{Nceldas-1+1}{2})}{Nceldas - \frac{Nceldas-1+1}{2}} + 1 \right]$$

Simplificando tenemos:

$$Lmedia = \frac{1}{2} Lcelda * \left[\frac{(Nceldas^2 + \frac{2\,Nceldas^2 - Nceldas}{6} - Nceldas^2)}{\frac{Nceldas}{2}} + 1 \right]$$

$$Lmedia = \frac{1}{2} Lcelda * \left[\frac{\frac{2\,Nceldas-1}{6}}{\frac{1}{2}} + 1 \right]$$

Y operando queda

$$Lmedia = \frac{1}{2} Lcelda * \frac{1}{6} * [4\,Nceldas - 2 + 6]$$

Simplificando de nuevo tenemos:

$$Lmedia = Lcelda * \frac{Nceldas + 1}{3}$$

como $N_{celdas} >> 1$ para un diseño de tamaño medio o grande, tenemos:

$$Lmedia = Lcelda * \frac{Nceldas}{3}$$

Como $\boldsymbol{L_{fila}} = L_{celda} * N_{celdas}$ obtenemos:

$$Lmedia = \frac{1}{3} Lfila$$

Normalmente los circuitos son cuadrados o casi cuadrados. Si suponemos el circuito cuadrado, L_{fila} es el lado del circuito, y para calcular la longitud de una interconexión sustituimos L_{fila} por la raíz del área total (A_{total}) del circuito.

$$Lmedia = \frac{1}{3}\sqrt{Atotal}$$

El área total viene dada por el área de los módulos más el área de las interconexiones. El área de los módulos está formada por el área de las celdas estándar más el área de las interconexiones internas de los módulos. Si suponemos que hay en total N interconexiones (entre interconexiones internas y externas de los módulos), todas de la misma longitud, la longitud de una interconexión media es:

$$Lmedia = \frac{1}{3}\sqrt{Aceldas + N * Lmedia * Wint}$$

donde $\mathbf{W_{int}}$ es la anchura de una interconexión, que depende de la tecnología, y su valor es conocido. En esta fórmula la única incógnita es L_{media}. Así obtenemos una ecuación de segundo grado en L_{media}:

$$9 * Lmedia^2 - N * Lmedia * Wint - Aceldas = 0$$

de donde podemos despejar el área de una interconexión media en una fila.

$$Lmedia = \frac{N * Wint + \sqrt{(N * Wint)^2 + 4 * 9 * Aceldas}}{18}$$

Como hemos supuesto que los tramos verticales de una interconexión se realizan en una capa diferente, la longitud de una interconexión media entre celdas colocadas en filas diferentes sobre la capa de *metal1* se reduce a los tramos horizontales, y se puede calcular suponiendo que las dos celdas se encuentran en la misma fila.

Hemos conseguido estimar el valor medio de una interconexión. Para estimar el área de un módulo hay que calcular el número de interconexiones internas que tiene, multiplicar por

L_{media} y añadir el área de celdas estándar, que es un dato conocido y disponible en la biblioteca de módulos.

Esta estimación tiene una complejidad N, donde N es el número de módulos. Por lo tanto cumple el primero de los requisitos necesarios en una buena estimación: es rápido. Sin embargo falta saber si es preciso y fiable.

4.5. Modelo refinado: Clasificación de las interconexiones

Para comprobar la validez de la estimación obtenida por el modelos simple, se realizan una serie de diseños con FIDIAS y se generan los layouts con CADENCE. A partir de estos datos se compara el área estimada con la que propone el estimador de la herramienta de diseño y con el área real del diseño.

Con este objetivo vamos a generar el diseño de uno de los ejemplos más frecuentemente utilizados en SAN ("benchmark"), el filtro elíptico de 5° Orden [HLSB91]. Utilizamos FIDIAS para realizar un proceso de diseño durante el cual se generan un total de 13 soluciones intermedias, hasta conseguir los objetivos dados por el usuario. Para cada uno de estos diseños se estima su área, utilizando como longitud de una interconexión la longitud media. Igualmente se genera el layout y se mide el área real.

Los resultados están representados en la figura 4.5. El área viene expresada en **puertas equivalentes** (p.e.). Una puerta equivalente es el área de un inversor en la tecnología utilizada. Utilizando puertas equivalentes el área de los módulos es independiente de la tecnología.

En la gráfica se presenta además el área estimada por CADENCE antes de generar el *floorplanning*. Podemos observar que para todos los diseños obtenidos, el área estimada utilizando como longitud de una interconexión el valor medio es superior al área real del circuito. El área estimada es también superior a la obtenida por los estimadores de CADENCE. Además, tampoco es una estimación fiable, ya que existen diseños como el nº 10,

cuya área estimada es menor que la estimada para el nº 11, y sin embargo ocurre al contrario con los valores reales de sus áreas respectivas. Este caso nos indica que no se puede utilizar estas estimaciones para guiar un proceso de diseño.

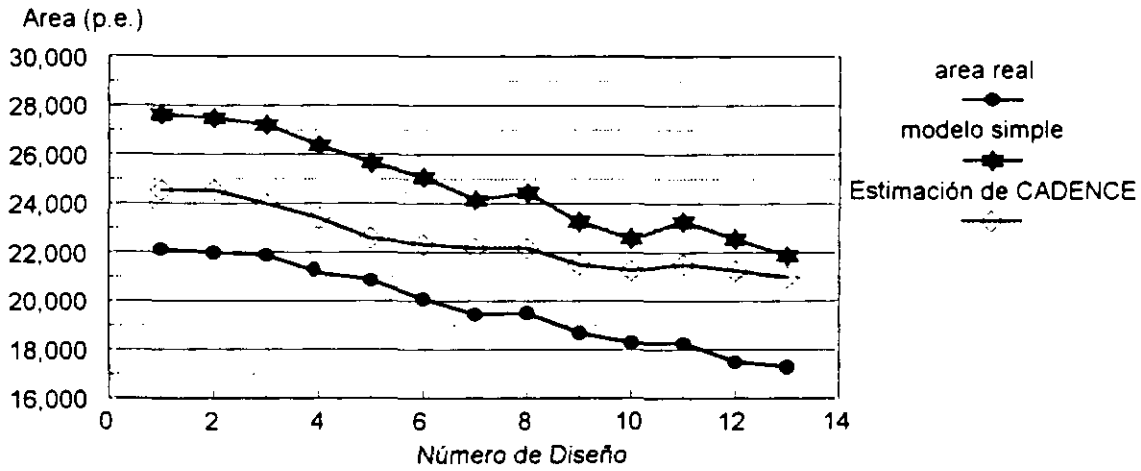


Figura 4.5 Estimación de Área para el Filtro Elíptico de 5º Orden usando L_{media}

Vamos a tratar de encontrar las razones de que la estimación no sea la adecuada. Si en todos los casos los resultados obtenidos son superiores a los reales, es porque el valor medio de la longitud de una interconexión en el diseño final, una vez generado el layout, es inferior a la longitud media obtenida por las fórmulas anteriores. La razón es que las celdas no están colocadas aleatoriamente en el circuito, sino que se tiende a colocarlas de forma que se disminuye el área del interconexionado.

Para comprobar estas suposiciones observamos el diseño de los circuitos después de realizar la colocación e interconexionado de los módulos y dedujimos que, efectivamente, los módulos conectados entre sí estaban colocados próximamente, y por tanto sus interconexiones tenían un valor muy inferior al valor medio estimado por la fórmula. Esto ha producido que se haya sobrestimado el área del interconexionado.

La explicación de la observación anterior se basa en el objetivo que tratan de alcanzar los

algoritmos de colocación e interconexionado de módulos: la minimización del área total del circuito (como se muestra en el Apéndice B). Debido a esto, cuando se realiza la colocación de los módulos en un CI se trata de conseguir que la longitud de las interconexiones sea mínima, y por tanto aquellos módulos más conectados entre sí se colocan más próximos que aquellos que no lo están.

En el caso ideal, si dos celdas están conectadas entre sí, se deberán colocar una al lado de la otra, y el valor de una interconexión será la longitud media de una celda estándar. Sin embargo, en muchos casos una celda está unida a varias celdas, y cada una de estas celdas a su vez está unida a otras celdas, y así sucesivamente. Y no es posible colocar juntas todas las celdas que están unidas entre sí. Para clarificar estos casos proponemos el siguiente ejemplo.

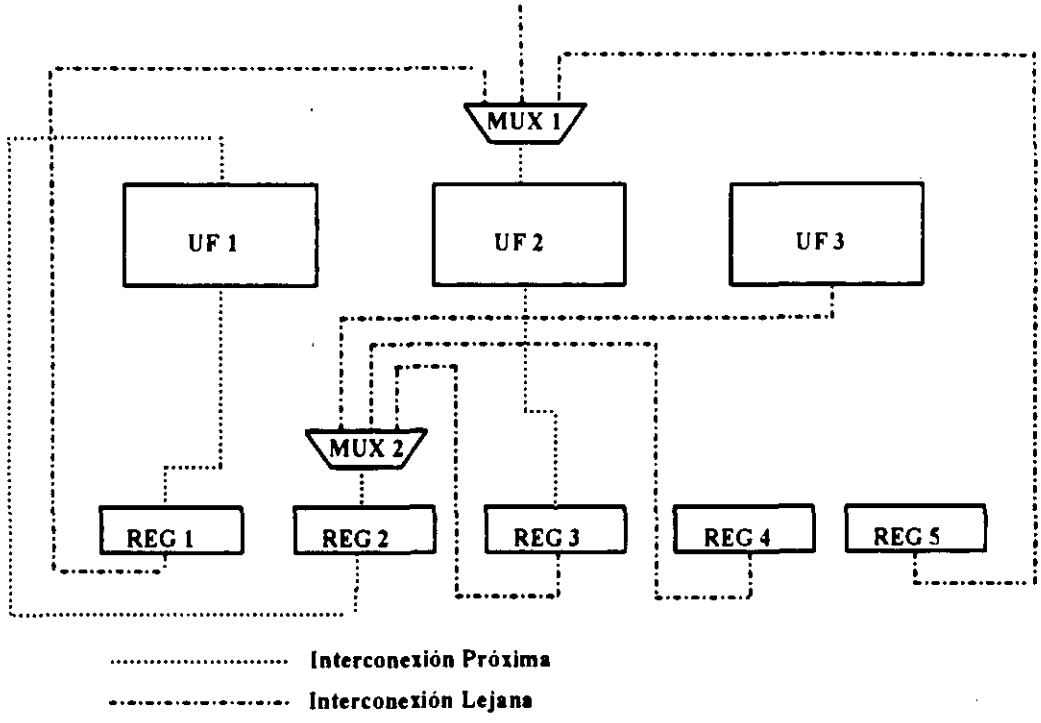


Figura 4.6 Fragmento de ruta de datos

La figura 4.6 muestra un fragmento de una ruta de datos. Suponemos que los registros, UFs y multiplexores de esta ruta de datos son directamente implementables mediante celdas estándar, es decir, cada módulo de la ruta de datos corresponde a una celda estándar de la biblioteca de

celdas.

Existen conexiones que van desde la salida de un módulo a la entrada de otro, (por ejemplo desde la UF1 al REG 1) y ni la salida del primer módulo ni la entrada del segundo están conectados a ningún otro módulo. Estos dos módulos se pueden colocar uno al lado del otro en el circuito final (ver figura 4.7) y su interconexión tendrá el tamaño mínimo dado por la longitud media de las celdas que une.

Existen otras interconexiones que van desde la salida de un módulo a la entrada de otro módulo, pero este último tiene a su vez otros módulos conectados a su entrada (por ejemplo, la conexión que une la salida de la UF3 y la entrada del REG2). Debido a que existen varios módulos conectados a la entrada del módulo destino, no todos los módulos fuente pueden colocarse junto al destino (ver figura 4.7), y por eso no todas estas interconexiones tienen un tamaño mínimo.

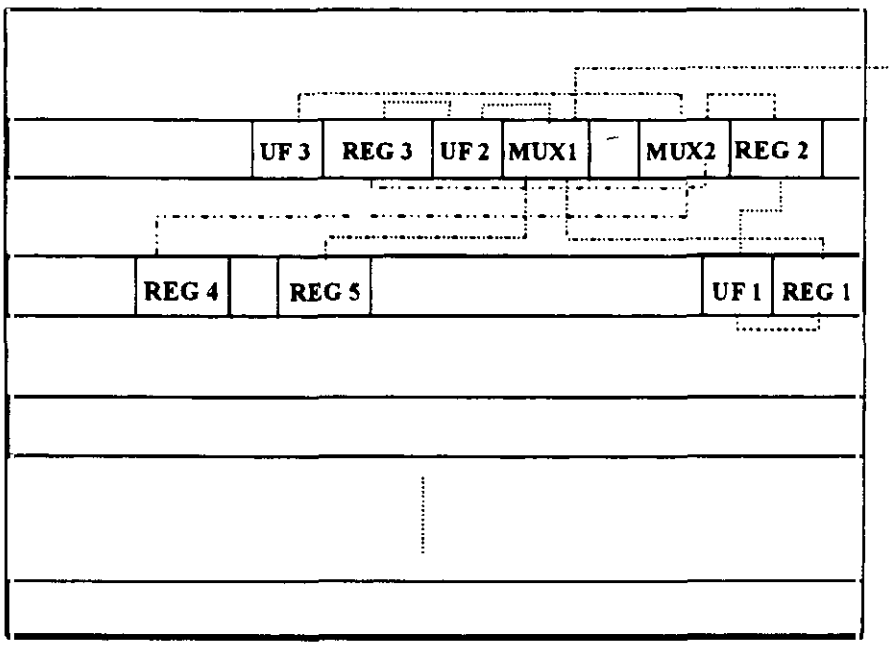


Figura 4.7 Layout de la ruta de datos

En el caso de una ruta de datos donde existan módulos que están formados por más de una

celda estándar, cuando la herramienta de SBN realiza el floorplanning sustituye cada módulo por las celdas que lo componen. Por lo tanto, al final lo que tenemos son conexiones entre celdas, y todas las ideas presentadas hasta el momento siguen teniendo la misma validez.

Todas estas observaciones nos llevan a distinguir dos tipos fundamentales de interconexiones:

- **Interconexiones cercanas**. Son aquellas que unen la salida de un módulo a la entrada de otro que no está conectado a ningún otro módulo. Por ejemplo la salida de un multiplexor conectada a la entrada de una UF o de un registro (figura 4.8). Este tipo de interconexiones es muy corta, ya que los algoritmos de colocación tienden a colocar estos dos módulos muy próximos entre sí.

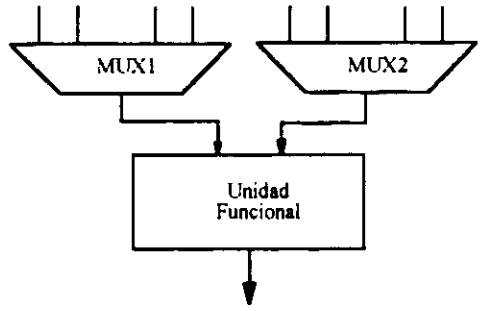


Figura 4.8 Interconexión cercana

Podemos suponer que su longitud es fija y con un valor igual al valor medio de las longitudes de las celdas estándar. Dentro de este grupo también están algunas de las interconexiones internas de los módulos. Por ejemplo, un multiplexor de 4 a 1 de 16 bits está formado por 4 multiplexores de 4 a 1 de 4 bits, que a su vez contienen 2 multiplexores de 2 a 1 de 4 bits (que son celdas estándar), conectados a la entrada de otro multiplexor de 2 a 1 de 4 bits. Estos 3 multiplexores es lógico que se coloquen muy próximos entre sí y por tanto las interconexiones internas de cada uno de los multiplexores de 4 a 1 de 4 bits son interconexiones cercanas. No ocurre lo mismo de las interconexiones entre los 4 multiplexores de 4 a 1 de 4 bits.

- **Interconexiones lejanas**: unen varios módulos entre sí, como por ejemplo la salida de varias UFs conectadas a la entrada de un mismo registro (figura 4.9). Cada una de estas interconexiones tiene un valor mayor que en el caso anterior, puesto que todos los módulos fuente no pueden colocarse próximos al destino, y por eso la calificamos como **lejana**. Como una primera estimación de su longitud tomamos el valor medio de una interconexión l_{media}, calculado de forma similar a la explicada en el apartado 4.4. A este grupo también pertenecen las interconexiones internas de los módulos que conectan varios bit-slices entre sí.

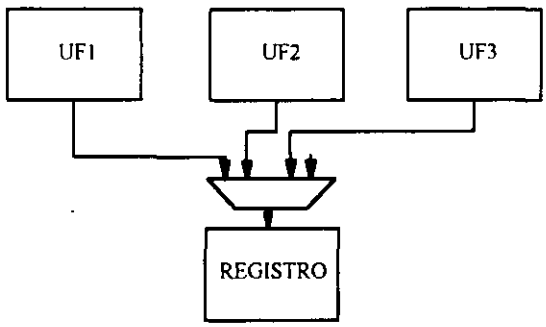


Figura 4.9 Interconexión lejana

4.6. Nueva estimación de la longitud de una interconexión

La clasificación de las interconexiones presentada en el apartado anterior nos permite realizar una nueva estimación del área de las interconexiones y módulos. El área de una interconexión cercana es un dato conocido que depende de la tecnología de diseño. Hay que estimar la longitud de una interconexión lejana, que hemos dicho que se toma como la longitud media de una interconexión, que depende de cada diseño. Vamos a utilizar el método del apartado 4.4 para calcular la longitud media de una interconexión, pero ahora teniendo en cuenta las interconexiones cercanas. Habíamos llegado a la siguiente fórmula:

$$Lmedia = \frac{1}{3}\sqrt{Atotal}$$

donde A_{total} es el área del circuito debida a módulos e interconexiones. Los módulos a su vez están formados por celdas estándar e interconexiones internas. En la biblioteca de módulos disponible para la herramienta de SAN, debe existir información sobre el área de celdas estándar y el número de interconexiones lejanas y cercanas internas de cada módulo. Todos estos datos son constantes para cada módulo, y se conocen cuando se diseñan.

Por otra parte, las interconexiones externas pueden ser también lejanas o cercanas, pero este es un dato conocido cuando se han terminado de crear todas las interconexiones.

Con todos estos datos vamos a calcular la longitud de una interconexión lejana. Supongamos que el área de celdas estándar total es A_{celdas}, el número total de interconexiones lejanas (internas y externas) es N_{lej} y el número total de interconexiones cercanas es N_{cerc}. La anchura de todas las interconexiones (lejanas y cercanas) es la misma, W_{int}. Sustituyendo en la fórmula anterior tenemos que la longitud de una interconexión lejana, L_{lej}, viene dada por:

$$Llej = \frac{1}{3}\sqrt{Aceldas + (Ncerc * Lcerc + Nlej * Llej) * Wint}$$

donde L_{cerc} es la longitud de una interconexión cercana, que es un dato conocido y constante para cada tecnología.

De nuevo tenemos una ecuación de segundo grado donde la única incógnita es L_{lej}. Despejando obtenemos:

$$Llej = \frac{Nlej*Wint + \sqrt{(Nlej*Wint)^2 + 4*9*(Aceldas + Ncerc * Lcerc * Wint)}}{18}$$

Esta fórmula permite calcular el área de una interconexión lejana, L_{lej}, con una complejidad proporcional a N, donde N es el número de módulos.

Para calcular el área de un módulo, utilizamos los datos sobre el área de celdas estándar,

número de interconexiones lejanas y cercanas, que son conocidos desde el momento en el que se diseña el módulo, y deben almacenarse en la biblioteca de módulos. Para calcular el área total de un módulo $Area_{modulo}$ se suma el área de celdas estándar ($Area_{activa}$), el área de interconexiones cercanas ($N_{cerc}*W_{int}*L_{cerc}$), y el área de interconexiones lejanas ($N_{lej}*W_{int}*Llej$):

$$Area_{modulo} = Area_{activa} + (N_{cerc} * W_{int} * L_{cerc}) + (N_{lej} * W_{int} * L_{lej})$$

Este nuevo método de estimación de área, además de tener en cuenta la tecnología de diseño, tiene en cuenta los objetivos de los algoritmos de colocación e interconexionado de módulos. Por eso suponemos que va a ser mucho más preciso que la estimación que trataba a todas las interconexiones de la misma forma, con un valor promedio.

Sin embargo, no todas las interconexiones existentes en un diseño pertenecen a los dos grupos anteriores. Existen interconexiones que unen la salida de un módulo con la entrada de varios módulos, y que no podemos clasificar como lejanas ni como cercanas. Por eso, antes de comprobar la fidelidad de esta nueva estimación sobre algunos ejemplos, es necesario realizar un tratamiento de este nuevo tipo de interconexiones.

4.7. Modelo completo: interconexiones multimódulo

Hasta ahora todas las estimaciones que hemos realizado se han dirigido hacia las interconexiones punto a punto, es decir, interconexiones que unen dos módulos únicamente, por ejemplo la salida de un registro con la entrada de una UF, la salida de un multiplexor con la entrada de un registro, etc. Pero no se han considerado las interconexiones que van desde la salida de un módulo a varias entradas de otros módulos, que llamaremos **interconexiones multimódulo** [MFSH94] [MFSM96]. Para clarificar este concepto observemos el fragmento de ruta de datos de la figura 4.10.

La salida de la UF2 está unida a la entrada de tres registros (REG3, REG4 y REG5). Este tipo

de interconexión no pertenece a ninguno de los tipos anteriores. No es cercana, puesto que la salida de la UF está conectada a más de un módulo. Y no es lejana porque los elementos destino sólo están conectados a una fuente. Hay que estudiar estas conexiones separadamente de las anteriores.

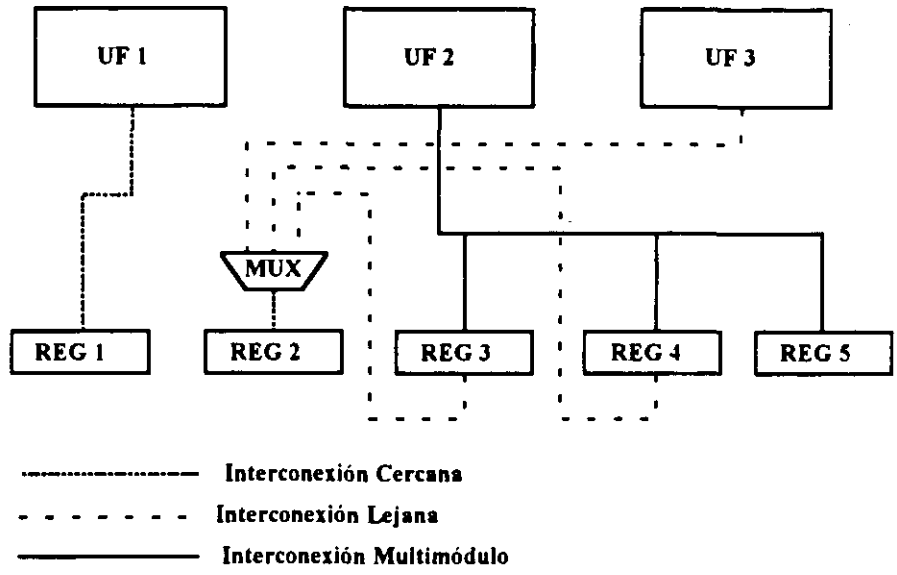


Figura 4.10 Fragmento de una ruta de datos con interconexiones multimódulo

Como las conexiones de este tipo conectan varios módulos, que a su vez estarán interconectados a otros módulos, los algoritmos de colocación e interconexionado no pueden colocarlos todos juntos en el layout final, y por tanto, podríamos realizar un tratamiento para ellas similar al de las interconexiones lejanas. Esto es lo mismo que suponer los módulos conectados uniformemente distribuidos en un cuadrado de área A_{total}, y tratar cada una de las conexiones como si el módulo de salida y el de entrada estuvieran en la misma fila de celdas estándar, de longitud L_{fila}, donde L_{fila} es la raíz cuadrada de A_{total}. Sin embargo, si tratamos una interconexión que conecta una fuente a m destinos como m interconexiones punto a punto, cada una de ellas con una longitud L_{lej}, obtenemos una sobreestimación del área, como puede observarse en la figura 4.11.

De nuevo hemos utilizado el filtro elíptico de 5° Orden como ejemplo. En la gráfica aparecen las áreas reales y estimadas por CADENCE para cada uno de los diseños, y además, el área estimada suponiendo todas las interconexiones iguales (modelo simple), y el área estimada utilizando longitudes diferentes para las interconexiones cercanas y para las lejanas (modelo refinado).

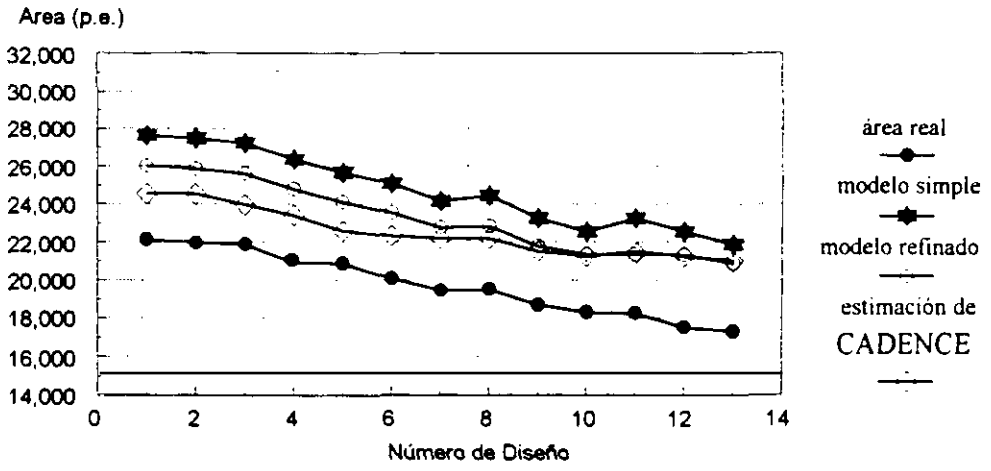


Figura 4.11 Estimación de Area para el Filtro Elíptico de 5° Orden

Podemos observar que la estimación obtenida con el modelo refinado ha mejorado sensiblemente respecto a la del modelo simple, y se aproxima notablemente al área estimada por CADENCE. Además, es suficientemente fiel para dirigir el proceso de diseño, puesto que permite distinguir entre dos diseños cuál es el de área mínima.

Sin embargo, en todos los casos el área estimada está alrededor de un 10% por encima del área real del circuito. Esto puede dar lugar a que, diseños que cumplen las restricciones de área mínima del circuito, sean desechados, puesto que el área estimada sería superior a dichas restricciones.

Esta sobreestimación es debida a que la ruta para las interconexiones multimódulo se puede generar como se muestra en la figura 4.12, donde un mismo tramo de la interconexión pertenece a varias de las *m* conexiones. Por eso la longitud de la interconexión total no es la

suma de las *m* conexiones punto a punto, sino que es menor.

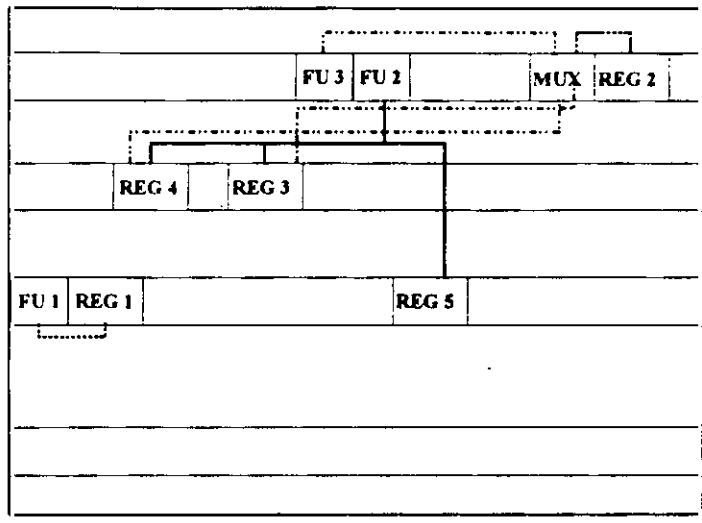


Figura 4.12 Ruta de Interconexiones multimódulo

Debido a esto, la longitud de una interconexión que une m módulos no crece linealmente con *m*. A medida que *m* se incrementa, el incremento que sufre la longitud total de la interconexión es menor, puesto que se pueden aprovechar más tramos de la interconexión de los *m-1* módulos anteriormente conectados.

Para calcular la longitud total de una interconexión multimódulo L_{mult} vamos a utilizar una función de interpolación. Esta función deberá cumplir dos requisitos principales:

- A medida que *m* crece, el incremento en la longitud de la interconexión al añadir un nuevo módulo debe disminuir.
- Cuando el número de módulos conectados crece, la longitud de la interconexión debe tender a la longitud de la fila de celdas estándar.

Existen múltiples funciones que cumplen estos dos requisitos. Pero nosotros necesitamos encontrar una que, además, sea sencilla de calcular y estime de forma lo más precisa posible el área real de este tipo de interconexiones.

Vamos a realizar un estudio intuitivo de qué forma podría tener esta función. Para esto, vamos a ir colocando los módulos de uno en uno. Primero colocamos el elemento fuente. Como no está conectado a ningún elemento el valor de la interconexión vale 0.

A continuación colocamos el primer elemento destino, que marcamos con "**1**" (ver figura 4.13). Podemos considerar esta primera interconexión como lejana, ya que, como hemos dicho anteriormente, los *m* módulos no pueden estar próximos y por tanto la distancia promedio entre ellos será L_{lej}.

Por tanto para $m=1$ la longitud será

$$Lmult = \frac{1}{3} Lfila$$

según calculamos en el apartado 4.5.

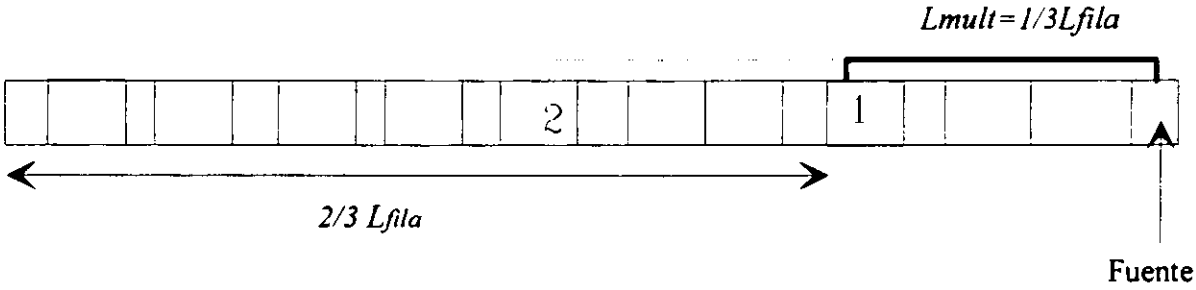


Figura 4.13 Interconexión multimódulo

Colocamos ahora el siguiente elemento (el marcado con "**2**" en la figura 4.13). La longitud de la interconexión debe experimentar, en término medio, un incremento menor que su valor, es decir, una fracción del valor que tenía. Asignamos a la siguiente interconexión hacia la izquierda lá longitud promedio correspondiente al tramo de pista que queda, es decir, la longitud media de los $2/3 L_{fila}$ restantes.

$$\Delta Lmult\,2 = \frac{1}{3} * \frac{2}{3} * Lfila$$

Es decir, el incremento que ha tenido lugar en la longitud de la interconexión vale 2/3 del

incremento que había experimentado la interconexión al colocar el primer elemento destino.

Si suponemos que este mismo comportamiento lo seguirán teniendo los incrementos a medida que añadimos más módulos, cuando coloquemos el siguiente elemento destino, es decir el "**3**", el incremento será la longitud media del tramo de pista que queda, es decir:

$$\Delta Lmult\,3 = \left(\frac{2}{3}\right)^2 * \frac{1}{3} * Lfila$$

Siguiendo el mismo razonamiento, el incremento en la longitud de la interconexión cuando se coloca el elemento *m* sería:

$$\Delta Lmult_m = \left(\frac{2}{3}\right)^{m-1} * \frac{1}{3} * Lfila$$

Podemos observar que, cuando *m* crece, el incremento cada vez es menor, lo que intuitivamente es correcto, y cumple el primer requisito que propusimos.

La longitud total de una interconexión L_{mult} que conecta una fuente a *m* módulos será la suma de todos los incrementos:

$$Lmult = \sum_{j=1}^{m}\left(\frac{2}{3}\right)^{j-1} * \frac{1}{3} * L\,fila$$

La suma de *m* términos de una serie geométrica es:

$$\sum_{j=1}^{m} ar^j = a\frac{r - r^{m+1}}{1 - r}$$

Sustituyendo en la fórmula anterior tenemos que la longitud de una interconexión entre un módulo fuente y *m* módulos destino viene dada por:

$$Lmult = \left[1 - \left(\frac{2}{3}\right)^m\right] * Lfila$$

Intuitivamente, si el número de módulos conectados crece, la longitud de la interconexión

debe tender a L_{fila}, como nos pedía nuestro segundo requisito. En la fórmula anterior, cuando m se hace muy grande, $Lmult$ tiende a L_{fila}, lo cual es correcto.

Ahora necesitamos relacionar todos los tipos de interconexiones. L_{fila} no es un valor conocido, sino que depende del área de las celdas estándar y de las interconexiones. La longitud de una interconexión lejana y multimódulo depende a su vez de L_{fila}. Es necesario obtener las fórmulas de L_{lej} y L_{mult} en función de valores conocidos una vez terminado de diseñar el circuito. En el próximo apartado presentamos las fórmulas finales para el cálculo de longitudes de todos los tipos de interconexiones, y algunos ejemplos que demuestran su fidelidad y precisión.

4.8. Estimaciones finales para el modelo completo

Hasta ahora hemos presentado tres tipos de interconexiones:

- Interconexiones **cercanas**: tienen una longitud fija L_{cerc} que depende de la tecnología y se conoce a priori.

- Interconexiones **lejanas**: tienen una longitud que depende del diseño en particular y estimamos su longitud como la longitud media de una interconexión, dada por:

$$Llej = \frac{1}{3} Lfila$$

- Interconexiones **multimódulo**: tienen una longitud que dependen del diseño en particular y del número de módulos que conectan, y estimamos su longitud por:

$$Lmult = \left[1 - \left(\frac{2}{3}\right)^{m}\right] * Lfila$$

Donde L_{fila} depende del área total del circuito por:

$$Lfila = \sqrt{Atotal} = \sqrt{Areaceldas + Areainterc}$$

A su vez el área de las interconexiones será la suma del área de las interconexiones cercanas, lejanas y multimódulo. Para facilitar el cálculo vamos a poner la longitud de una interconexión multimódulo en función de la longitud de una interconexión lejana:

$$Lmult = 3 * \left[1 - \left(\frac{2}{3} \right)^m \right] * \frac{1}{3} Lfila = 3 * \left[1 - \left(\frac{2}{3} \right)^m \right] * Llej$$

Es decir, una interconexión multimódulo que conecta una fuente a m módulos, es lo mismo que $3 * \left[1 - \left(\frac{2}{3} \right)^m \right]$ interconexiones lejanas. Es como si cada interconexión multimódulo pudiera sustituirse por un número de interconexiones lejanas dado por $3 * \left[1 - \left(\frac{2}{3} \right)^m \right]$.

En el apartado 4.6, obtuvimos la longitud de una interconexión lejana en función del área de celdas estándar, el área de interconexiones cercanas y el número de interconexiones lejanas, que venía dada por:

$$Llej = \frac{Nlej * Wint + \sqrt{(Nlej * Wint)^2 + 4 * 9 * (Aceldas + Ncerc * Lcerc * Wint)}}{18}$$

En esta fórmula no se consideraban las interconexiones multimódulo. Pero esta misma fórmula sigue siendo válida para calcular la longitud de una interconexión lejana, si sustituimos el número de interconexiones lejanas, N_{lej}, por N_{lej} más el equivalente en interconexiones lejanas de las interconexiones multimódulo. Este número total es conocido en el momento de terminar la generación del diseño.

A partir de la longitud de la interconexión lejana puede calcularse el área de cualquier interconexión multimódulo utilizando la fórmula que relaciona sus longitudes. De esta forma obtenemos de nuevo un método de una complejidad proporcional a N, donde N es el número de módulos del diseño, para calcular el área de cualquier tipo de interconexión del circuito, y

por tanto del área final.

El cálculo del área de los módulos del circuito se puede realizar de forma análoga a como se presentó en el apartado 4.6 pero teniendo en cuenta las interconexiones multimódulo internas a los módulos. En la biblioteca de FIDIAS se almacena el número de interconexiones cercanas y lejanas, y en estas últimas se incluye las interconexiones multimódulo, con lo cuál las fórmulas para calcular el área son idénticas a las del apartado 4.6.

Una vez obtenidas las longitudes de todos los tipos de interconexiones de cualquier diseño, vamos a generar de nuevo las áreas de los distintos diseños para el ejemplo que hemos utilizado en los apartados anteriores. En la figura 4.14 tenemos la estimación obtenida para el Filtro Elíptico de 5° Orden, utilizando una clasificación completa de las interconexiones y sus fórmulas correspondientes (modelo completo). También presentamos las estimaciones obtenidas anteriormente (modelo simple y modelo refinado), para que se pueda observar cómo nos hemos ido aproximando al área real del circuito.

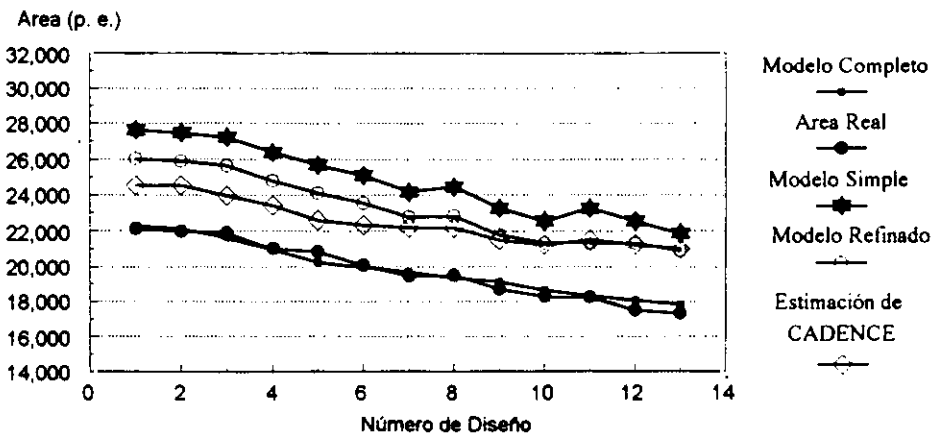


Figura 4.14 Estimación de Área para el Filtro Elíptico de 5° Orden

Se puede comprobar que la última estimación realizada, utilizando los tres tipos de interconexiones, es fiable, como lo era la obtenida con el modelo refinado, pero además es suficientemente precisa para conseguir los objetivos que nos habíamos propuesto. Para este

ejemplo en particular, el error es siempre inferior al 5%.

Antes de presentar más ejemplos para comprobar con la validez de estas estimaciones, vamos a hacer algunas consideraciones. Hasta ahora, todas las comprobaciones se han basado en la estimación global del área del circuito, una vez se conocen todos los módulos e interconexiones que lo componen. Pero las estimaciones en SAN son mucho más útiles si también se pueden realizar durante la generación de un diseño, para guiar correctamente el proceso de toma de decisiones. Por eso, en el próximo apartado vamos ver cómo se integran las estimaciones dentro de nuestra herramienta de SAN, con el fin de comprobar que estas estimaciones también se pueden utilizar dentro de un proceso y cumplen los objetivos deseados.

4.9. Estimaciones de la longitud de una interconexión durante el proceso de diseño

Como presentamos en el capítulo 3, en el sistema FIDIAS se necesitan estimaciones de área en varios puntos a lo largo del proceso de diseño: después de la planificación, durante y después de la asignación de hardware y durante la generación del control. En los próximos apartados veremos cómo se realiza cada una de ellas.

4.9.1. Estimaciones después de la planificación-preasignación

En primer lugar se necesitan estimaciones de área después de la planificación-preasignación de hardware. El Experto de Diseño precisa evaluar la calidad de la planificación obtenida, para prever si se cumplirán las restricciones del usuario. Para esto, es necesario estimar el área del circuito cuando todavía no se ha realizado la asignación de hardware. Lo que se calcula en este caso es el área mínima del circuito [Mozo92], que vendrá dada por:

1.- **El área mínima de registros**. El número mínimo de registros se calcula mediante un algoritmo basado en REAL [KuPa87], con modificaciones para tratamiento de bucles y

condicionales. Cada uno de los registros está formado por un conjunto de celdas estándar y un conjunto de interconexiones, cuya área no es conocida, pero sí se conoce su tipo y número. El área de las interconexiones se calculará posteriormente.

2.- **El área mínima de UFs**. Después de la preasignación se conoce el número y tipo de las UFs que van a realizar cada una de las operaciones del GFD planificado. Como para los registros, se conoce el área de celdas estándar de cada UF, pero no de las interconexiones internas. Sin embargo, si se conoce el número y tipo de las interconexiones de cada módulo, que se utilizará posteriormente para calcular el área de las interconexiones.

3.- **El área mínima de interconexiones**. De las interconexiones internas a los módulos se conoce su tipo y número, calculado según se ha explicado anteriormente. Para calcular el número mínimo de interconexiones externas, se utiliza el máximo número de entradas/salidas de UFs que se utilizan en un paso de control. Desde luego esta estimación es un poco burda, pero la información disponible sobre el circuito sólo permite obtener una aproximación. Una vez que tenemos el número total de interconexiones de cada tipo, se estima el área de una interconexión lejana, mediante las fórmulas presentadas en el apartado anterior. Conocido este valor, se puede calcular el área mínima del CI.

Si este valor de área mínima estimada es superior a la restricción impuesta por el usuario, la planificación se desecha y se vuelve a realizar utilizando otro valor del tiempo de ciclo o bien incrementando el número de etapas de control, con el objetivo de disminuir el paralelismo y aumentar la reutilización de hardware. Si, por el contrario, es menor, se pasa a la asignación de hardware.

4.9.2. Estimaciones durante y después de la asignación de hardware

Ya vimos durante el capítulo 3 que la estimación del área durante la asignación de hardware en el sistema FIDIAS no se realiza sólo para un diseño completo, sino también para una decisión, cada vez que se debe optar por una de las alternativas posibles, como la reutilización

de un hardware existente o la creación de nuevos módulos. En cada uno de los casos es necesario estimar el área de los nuevos módulos (si se crea alguno), y el área de las interconexiones que se necesitan añadir a los diseños.

Como se ha señalado anteriormente, el área de los módulos (registros, UFs y multiplexores) está formada por el área de las celdas estándar y de las interconexiones. De éstas últimas sólo conocemos su número y tipo, que está almacenado en la biblioteca de módulos. El área de las interconexiones cercanas es posible conocerlo, puesto que la longitud es un valor fijo para cada tecnología, y está almacenado también en la biblioteca de módulos. Sin embargo, el área de las interconexiones lejanas es dependiente del diseño final, cuya composición todavía no es conocida. Recordemos que el número de interconexiones multimódulo se considera integrado dentro del número total de interconexiones lejanas, por tanto su área se incluye en la de aquéllas.

También hay que estimar el área de una interconexión cada vez que se crea. Si es lejana o multimódulo, el área depende del diseño final, que todavía no se conoce. Por tanto, vemos la necesidad de obtener un valor inicial del área de una interconexión lejana antes de comenzar la asignación. Es posible utilizar la estimación obtenida por el Experto de Diseño una vez acabada la planificación, basada en el área mínima del circuito integrado. Esta estimación obtiene siempre un valor inferior al real, pero como tiene en cuenta las características del diseño en particular, además de consideraciones sobre la tecnología y modo de funcionamiento de los algoritmos de colocación e interconexionado, será una aproximación mejor que cualquier otro valor tomado aleatoriamente.

Con este primer valor se produce la primera asignación de hardware. Al final de ésta se conoce el número real de módulos e interconexiones del circuito para el primer diseño obtenido, y se recalcula el valor del coste de una interconexión lejana. El valor que se obtiene ya es suficientemente fiable, puesto que se ha computado con un número real de módulos e interconexiones y permite recalcular el área total del circuito.

Como el proceso de diseño es iterativo, partimos del valor de la longitud de una interconexión lejana obtenido en este primer diseño para realizar la asignación en el segundo diseño y así sucesivamente.

Cada vez que se obtiene un nuevo diseño se recalcula el valor de la longitud de la interconexión y se actualiza su valor, así como la estimación del área total del circuito. Aunque el valor estimado para el primer diseño difiere notablemente del valor recalculado después de la asignación de hardware, debido a la falta de suficientes datos sobre el circuito, para el resto de los diseños la estimación es suficientemente buena, ya que de un diseño al siguiente el número de unidades funcionales, registros e interconexiones no varía de forma drástica, con lo cual, el error cometido al utilizar como valor medio del coste de una interconexión el obtenido en un diseño anterior, es mínimo.

En la gráfica de la figura 4.15 podemos observar la variación del área estimada de una interconexión durante el proceso de diseño para el ejemplo del filtro Elíptico de 5° Orden, presentado anteriormente.

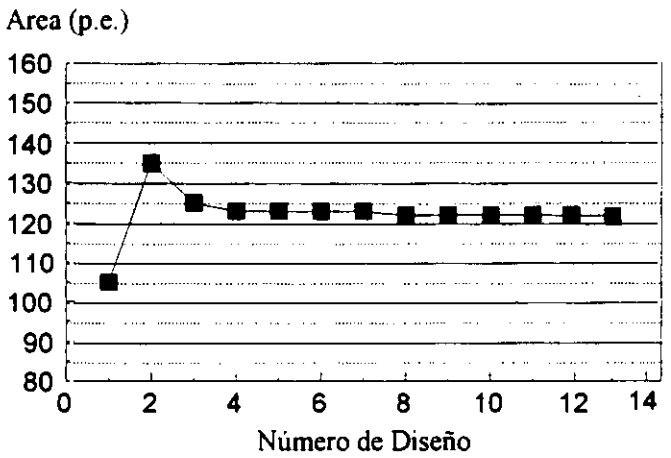


Figura 4.15 Variación del Área de una interconexión lejana durante el proceso de diseño

Se observa que en el primer diseño la estimación que se tiene del área de una interconexión es de 106 p.e. y cuando se recalcula su valor, una vez finalizada la primera asignación de

hardware, es de 136 p.e.. La variación del valor respecto a la primera estimación es del orden del 23%. Como era de suponer el área estimada de una interconexión antes de realizar la asignación de hardware es inferior a la obtenida después, puesto que sólo se había evaluado el área mínima del CI. Además, no se había considerado el área de los multiplexores, que influyen notoriamente en el área total del circuito y, por tanto, en el área de una interconexión lejana.

En el diseño número 2 utilizamos el valor de 136 p.e. como estimación (obtenido a partir del diseño número 1) y al recalcularlo obtenemos 122 p.e.. La variación es del 11%. Además, en este y en posteriores diseños se ha utilizado un valor del área de una interconexión mayor que el real, puesto que el asignador de hardware obtiene diseños cada vez más baratos, y el área de la interconexión lejana cada vez es menor.

Aunque una variación del 11% puede parecer excesiva, para los diseños siguientes el valor *estimado de una interconexión a partir de un diseño y el recalculado en el diseño siguiente* permanece aproximadamente constante, con una variación inferior al 1%. Por lo tanto, podemos utilizar las estimaciones de área de una interconexión basándonos en el valor del área del diseño anterior sin pérdida de la precisión deseada.

Una vez terminada la asignación de hardware, el Experto de Diseño utiliza la estimación de área del último diseño obtenido, con el valor de una interconexión lejana debidamente actualizado, para evaluar la calidad de la asignación y decidir si se puede proseguir con la siguiente tarea o es necesario reiniciar de nuevo el proceso.

4.9.3. Estimación del área del controlador

Una vez que se ha realizado la planificación de operaciones y la asignación de hardware, la generación del control es un proceso automático y no se puede optimizar su tamaño. Sin embargo, el tamaño de la Unidad de Control también influye en la del CI y es necesario tenerla en cuenta.

En el sistema FIDIAS sólo se considera esta influencia durante la planificación de operaciones. En esta etapa se trata de minimizar el número de pasos de control, tanto más cuanto más importante sea la minimización de área. Esto es debido a que el número de etapas determina el tamaño de la ROM de microinstrucciones, si se trata de un control microprogramado, o de la máquina de estados, si es cableado. Este método lo veremos en el próximo capítulo.

La estimación del área de los módulos de la Unidad de Control puede realizarse de forma análoga a la del resto de los módulos del circuito, disponiendo en la biblioteca de información sobre el área de celdas estándar de cada uno de ellos, así como del tipo y número de interconexiones. Si la ROM de microinstrucciones se implementa con una macrocelda, es fácil obtener una estimación de área en función del ancho de la microinstrucción y del número de éstas. El tamaño de las macroceldas es fijo, y su valor no se modifica cuando éstas se colocan en el diseño final.

Por otra parte, las señales de control son interconexiones de un bit, mientras que, en nuestro sistema, las interconexiones de datos son de 16 bits. Si se estima el área de las señales de control tratándolas como interconexiones lejanas o multimódulo, según el caso, hay que considerar que su anchura es 16 veces menor. Por esta razón, para los ejemplos considerados el número de señales de control es bastante bajo y se puede suponer que equivalen a 1 ó 2 interconexiones de datos de tipo lejano, con lo cual su influencia en el área final es muy pequeña.

4.10. Ejemplos

Vamos a presentar algunos ejemplos más para mostrar la calidad de las estimaciones explicadas anteriormente. Nuestro objetivo principal es que se dirija correctamente el proceso de diseño, es decir, que cada diseño obtenido sea realmente mejor que el anterior.

Las estimaciones están integradas dentro del sistema FIDIAS, según el método presentado en

el apartado anterior. Se introducen los parámetros al sistema para que obtenga el diseño con un área mínima. Así el asignador de hardware genera una serie de diseños, cada uno mejor que el anterior, hasta que no puede encontrar un diseño mejor que la última solución propuesta, se le ordena parar o bien se sobrepasa un determinado tiempo máximo de búsqueda. Para todos los diseños obtenidos, se genera el layout con CADENCE y se mide el área.

En las gráficas se representan los diseños en el eje de abcisas, en el orden que los ha generado el asignador de hardware, es decir, en orden decreciente de área. En el eje de ordenadas se presenta el área estimada por el asignador para cada uno de los diseños, una vez recalculada el área de las interconexiones lejanas, y el área real del circuito. Como en los ejemplos anteriores, las áreas están medidas en puertas equivalentes.

4.10.1. Filtro Elíptico de 5º Orden

De nuevo vamos a utilizar el filtro elíptico de 5° orden, pero ahora con una planificación diferente. Como hemos dado prioridad al área frente al tiempo, se ha incrementado el número de pasos de control de 17, para el ejemplo anterior, a 19, con el objetivo de minimizar el paralelismo. En la figura 4.16 se puede observar que realmente se ha conseguido disminuir el área, de 18000 p.e. para el mejor diseño obtenido con 17 pasos de control, hasta 12000 p.e. con 19.

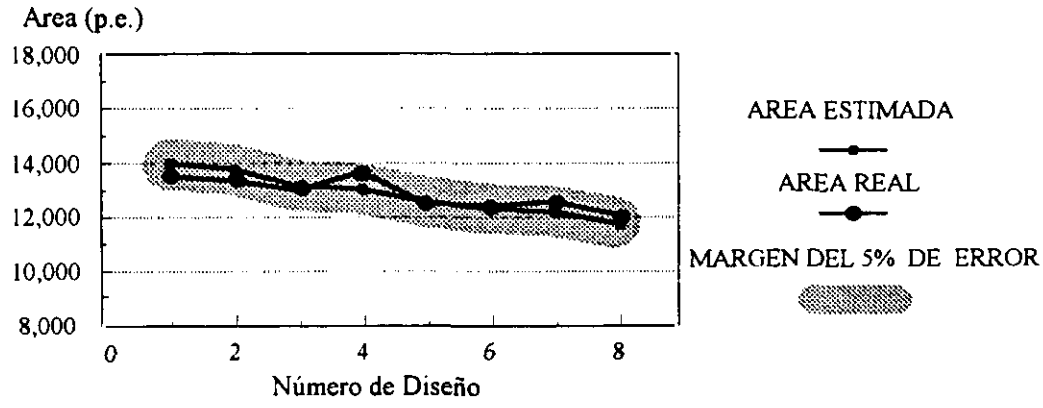


Figura 4.16 Filtro Elíptico de 5° Orden con 19 pasos de control

En todos los casos el error cometido es inferior al 5%, como se puede observar en la gráfica donde se ha dibujado la banda que marca el margen de dicho error. Todas las áreas reales de los diseños se encuentran colocados dentro de dicha franja. Es importante tener en cuenta este margen para juzgar la fidelidad de las estimaciones. Así por ejemplo, el diseño 4 se estimó que tenía un área inferior al 3 y sin embargo el área real del diseño 4 es un poco mayor, menos de un 5%, que el área real del diseño 3. Esa diferencia se encuentra dentro del margen de error, y por eso la estimación no ha podido precisarla.

Como el asignador de hardware distingue incluso entre dos diseños cuya área estimada se diferencie en tan solo una única puerta equivalente, puede darse el caso, como el que hemos visto, de que no realice una opción correcta.

Pero esto realmente no es un problema, puesto que si dos diseños se diferencian en menos de un 5%, para un diseñador suelen ser igualmente válidos. Al final del proceso de diseño para este ejemplo en particular, la diferencia entre el primer y el último diseño obtenidos con 19 pasos de control es del orden del 15% , y en la mayoría de los casos el algoritmo de asignación produce mejoras muy superiores a ese valor.

4.10.2. Ecuación diferencial

A continuación presentamos otro ejemplo frecuentemente utilizado en la .literatura sobre SAN [PaKG86]: la ecuación diferencial.

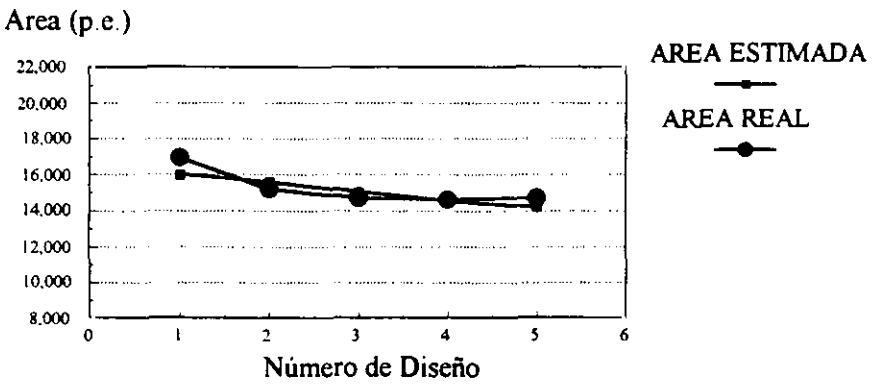


Figura 4.17 Ecuación diferencial

La planificación se realizó en 9 pasos de control y el asignador generó un total de 5 diseños. Las dos gráficas, la del área estimada y el área real se muestran en la figura 4.17. Se puede comprobar que ambas tienen la misma forma, y la precisión de las estimaciones ha sido del orden del 95%.

4.10.3. Filtro-Ar

El siguiente ejemplo que presentamos es el filtro-Ar [JaPP87]. La planificación se realizó en 11 pasos de control. La figura 4.18 muestra los resultados: se generaron 9 diseños. Los errores en todos los casos son inferiores al 5% En los puntos donde las gráficas no tienen exactamente la misma forma, como para el caso del filtro Elíptico de 5° Orden, se debe a que las área de los diseños obtenidos por el asignador diferían en menos de un 5%, lo cual está dentro del margen de error que el estimador ha mostrado en los experimentos.

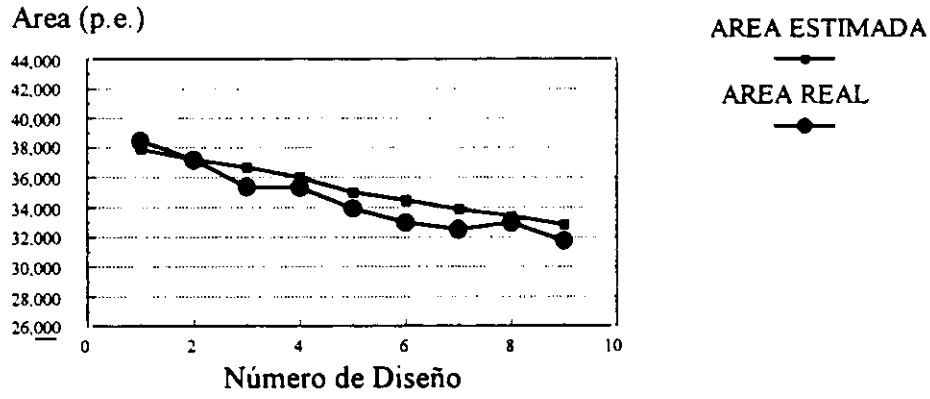


Figura 4.18 Filtro-Ar para 11 pasos de control

Pero se puede observar que el proceso de diseño se ha guiado correctamente, y al final se ha obtenido el diseño con área mínima.

Seguidamente, vamos a presentar otro ejemplo con el filtro Ar, pero ahora con una planificación diferente (figura 4.19). Se ha colocado una restricción más fuerte del área del circuito, con lo cual la herramienta de SAN ha incrementado el número de pasos de control a 14, con el fin de disminuir el paralelismo. En la figura 4.19 se puede observar cómo los

diseños obtenidos tienen un área inferior a los generados para la planificación anterior, y se ha pasado de 32000 p.e. a 16000 p.e. para el mejor diseño. El error máximo cometido por las estimaciones en este caso es del 5%, y el proceso de diseño se ha guiado correctamente.

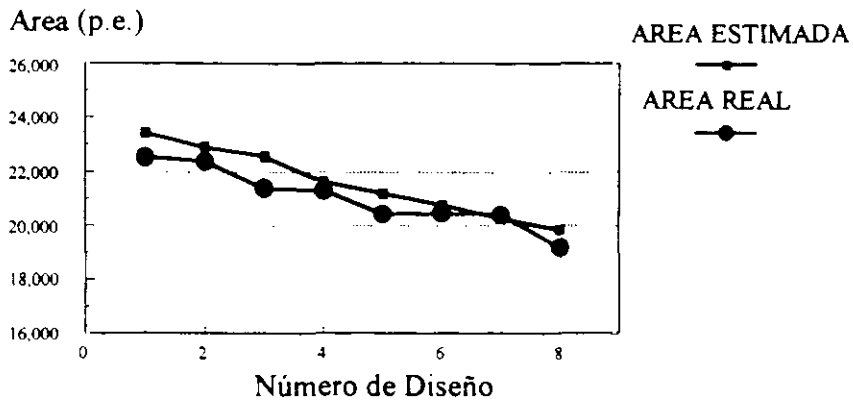


Figura 4.19 Filtro-Ar en 14 pasos de control

4.11. Conclusiones

A lo largo de este capítulo se ha presentado un método para realizar estimaciones del área de un circuito durante el proceso de SAN. Este método está basado en un estudio de:

- la tecnología de diseño de celdas estándar,
- la forma de trabajo de los algoritmos de colocación e interconexionado de módulos,
- el diseño en particular.

Gracias a ello se obtienen estimaciones muy precisas y que pueden generarse mediante cálculos simples, lo cual supone un avance respecto de otros métodos propuestos en el campo de la SAN.

En primer lugar se han clasificado las interconexiones en función del número de módulos que conectan. Para cada uno de los tipos se ha obtenido una estimación de su longitud. Las

interconexiones que hemos denominado cercanas tienen longitud constante. Para las lejanas y multimódulo se ha obtenido una fórmula en función del número total de interconexiones de cada tipo y del área de celdas estándar del diseño, datos conocidos cuando se ha terminado la generación de un diseño.

Este método de estimación de la longitud de las interconexiones nos permite conocer el área de los módulos, siempre que en la biblioteca se disponga de información sobre el número y tipo de interconexiones que contiene cada uno de ellos.

De esta forma, el cálculo del área total o parcial del circuito se convierte en un algoritmo de complejidad proporcional a N, donde N es el número de módulos, y una precisión del 95%.

Dado que este método permite conocer el área del circuito una vez que se conocen sus componentes, es directamente aplicable para algoritmos de tipo iterativo, que realizan los cálculos del área para comparar diferentes diseños, y por tanto conocen perfectamente su estructura completa.

Para los algoritmos de tipo constructivo es necesario obtener una estimación inicial del área mínima del circuito, a partir de la cual se genera una primera aproximación del área de las interconexiones lejanas y multimódulo. Una vez se ha terminado la composición del circuito, se recalcula dicho valor y se estima el área del circuito final.

Por último, para los algoritmos de tipo constructivo-iterativo se ha visto la forma de combinar ambos métodos. Primero se realiza una estimación inicial del área mínima del circuito y, a partir de ella, se calcula una primera aproximación del valor de las interconexiones lejanas y multimódulo, con el cual se genera el primer diseño. Para el resto de los diseños, se utiliza la estimación de la longitud de las interconexiones del último circuito obtenido.

Mediante varios ejemplos hemos comprobado la fidelidad y precisión de estas estimaciones de área, que en todos los casos tienen un error inferior al 5%. Como el retardo de una interconexión depende de su longitud, utilizaremos estas estimaciones en los próximos

capítulos, para realizar estimaciones sobre los retardos de las interconexiones y su influencia en el rendimiento de los circuitos diseñados.

Capítulo 5.

Selección del Ciclo de Reloj

5.1. Introducción

Dado que la selección del ciclo de reloj es un proceso previo a la planificación de las operaciones en pasos de control, la información del circuito disponible en el momento de realizarla es muy limitada: un grafo del flujo de datos y de Control (**GFDC**), una biblioteca de módulos que pueden realizar todas y cada una de las operaciones del GFDC y una serie de especificaciones dadas por el usuario como pueden ser el área máxima, el tiempo de ejecución máximo, etc. A pesar de la falta de información, la selección debe realizarse cuidadosamente por dos razones fundamentales:

- Es una de las decisiones que se toman al comenzar el proceso de síntesis, y en la cual se basan la planificación y la asignación de hardware. Un cambio en el valor del tiempo de ciclo implica que se tenga que volver a diseñar completamente el circuito: nueva planificación y, por tanto, nueva asignación de hardware.

- Debido a la influencia del tiempo de ciclo en el tiempo total de ejecución de la descripción del circuito y en el tamaño final de la Unidad de Control. Una selección inadecuada puede conducir a diseños que no cumplan los objetivos, e incluso las restricciones impuestas por el usuario.

La mayoría de los sistemas de síntesis [WaPa95], [WaCa91], [PaKn89b], [BaMa89] reciben como entrada el valor del tiempo de ciclo. Es el diseñador el que, basándose en su intuición, debe seleccionar el valor más apropiado. Sin embargo, una mala selección

puede conducir a diseños con tiempos de ejecución excesivos o con Unidades de Control muy grandes.

Por eso es importante que, una elección con tantas repercusiones en los resultados finales, se realice dentro del proceso de síntesis mediante un estudio de todos los factores influyentes.

Uno de los factores que hay que tener en cuenta en la selección del tiempo de ciclo es el tiempo muerto de las operaciones del GFD. Se define el **tiempo muerto** de una operación como el tiempo que transcurre desde que se finaliza la ejecución de una operación hasta que se termina el ciclo. Por ejemplo, si una operación dura 17 ns y el tiempo de ciclo es de 4 ns, necesitamos cinco ciclos de reloj para la ejecución de dicha operación. Los cuatro primeros ciclos se utilizan totalmente, pero del quinto sólo se usa 1 ns y los otros 3 ns son lo que se denomina tiempo muerto. El valor del tiempo de ciclo determina los tiempos muertos de las operaciones, y por tanto influye en el tiempo total de ejecución de la especificación del circuito. Cuanto más pequeño sea el tiempo de ciclo, los tiempos muertos de todas las operaciones tienden a ser menores (como mucho pueden ser del tamaño del tiempo de ciclo menos uno), y por tanto el tiempo de ejecución total suele ser menor. Sin embargo, el tamaño de la Unidad de Control crece con el número de pasos de control, y éste es inversamente proporcional al tiempo de ciclo. Dependiendo de las características del diseño se debe dar más o menos prioridad a cada uno de estos factores: el área final de la Unidad de Control y el tiempo total de ejecución de la especificación del circuito.

En este capítulo se presenta un método de selección del tiempo de ciclo que tiene en cuenta las restricciones y características del diseño, la tecnología de fabricación, la biblioteca de módulos disponibles, el tiempo total de ejecución y el tamaño de la Unidad de Control. Como estos dos últimos factores tienen influencias muy distintas en el valor óptimo del tiempo de ciclo, se realiza un estudio de cada uno de ellos por separado.

En el próximo apartado se estudian los métodos de selección del tiempo de ciclo que se han incluido en algunos sistemas de SAN. En el siguiente se deducen los tiempos de ciclo que minimizan el tiempo total de ejecución. El apartado cuarto trata la influencia del tiempo de ciclo en el tamaño de la unidad de control. En el quinto se presenta el algoritmo de selección del tiempo de ciclo que considera estos dos factores y las características de cada diseño.

5.2. Trabajo Previo

La mayoría de los sistemas de SAN, incluso recientemente desarrollados, reciben el tiempo de ciclo como dato de entrada [WaPa95]. Es por tanto el diseñador el que, basándose en su experiencia, debe decidir su valor. Sólo algunos sistemas tratan de realizar la selección basándose en las características del circuito. Por ejemplo, en el sistema MAHA [PaPM86] se elige como tiempo de ciclo el máximo retardo de cualquier operador del camino crítico.

Un modelo de estimación área-tiempo se presenta en [JMPP88]. El GFD se divide en un conjunto de pasos de control ***npasos***. Se elige un tiempo de ciclo igual al máximo entre el retardo del camino crítico dividido entre *npasos*, y el retardo máximo de un operador en dicho camino.

Estos dos sistemas no permiten multiciclo y, por tanto, el mínimo ciclo de reloj permitido es el máximo retardo de un operador del GFD. Además, no permiten módulos que realicen más de una operación, y sólo tienen en cuenta el retardo de las UFs, pero no el de los registros, multiplexores e interconexiones. La elección del máximo retardo de un operador como tiempo de ciclo puede generar tiempos muertos excesivos, que incrementarían el tiempo de ejecución total.

El sistema HIS [WaCa91] optimiza el número de pasos de control y, por tanto, también

el tamaño de la Unidad de Control. Utiliza la planificación AFAP, que considera restricciones de hardware para determinar el mínimo número de pasos de control, y el máximo retardo de un paso de control se elige como ciclo de reloj. De esta forma se penaliza el tiempo de ejecución total.

Existen sistemas que utilizan un valor inicial del tiempo de ciclo, elegido sin considerar el GFDC, para obtener una planificación, y posteriormente se optimiza dicho valor. Por ejemplo, en [BhDB94] se presenta un algoritmo para sistemas que no utilizan multiciclo, que parte de un GFDC planificado y preasignado, y realiza una asignación que minimiza el ciclo de reloj.

Por otra parte, en [PaJD94] se presenta un algoritmo que realiza una selección del tiempo de ciclo para una ruta de datos y una planificación dada. Primero se demuestra que los valores del tiempo de ciclo que pueden conducir a tiempos de ejecución mínimos son los divisores del retardo de cada uno de los estados. El método utilizado es realizar una nueva planificación para todos estos divisores y tomar como tiempo de ciclo el que produce menor tiempo de ejecución.

El problema que presentan estas soluciones es la necesidad de realizar inicialmente una selección del tiempo de ciclo, en la cual se basa la obtención de buenos resultados. Además, en el último algoritmo es necesaria una nueva planificación, con lo cual la asignación de hardware podría no ser óptima y sería conveniente hacer una nueva. Por otra parte, la utilización de los divisores como posibles tiempos de ciclo puede dar lugar a opciones con valores muy pequeños, que hagan crecer de forma excesiva el número de estados.

Muy pocos sistemas realizan un estudio inicial del GFDC y de la biblioteca de módulos para obtener una buena selección. En [NaGa92] se presenta un algoritmo basado en la minimización de los tiempos muertos. Para cada tipo de operación del GFD se computa su retardo, teniendo en cuenta el retardo de los drivers, de los operadores y de los

registros utilizados. Se define el máximo tiempo de ciclo posible como el máximo retardo de una operación del GFD; y el mínimo tiempo de ciclo como el mínimo valor permitido por la tecnología. Para cada valor posible del tiempo de ciclo entre el mínimo y el máximo, con un incremento igual a 1, se calcula el tiempo muerto para cada operador, el tiempo muerto medio y el porcentaje de utilización de operadores. El valor del ciclo que maximiza dicha utilización se elige como ciclo de reloj. En este algoritmo existe sólo un tipo de UF para implementar cada operación y sólo trata de minimizar el tiempo de ejecución, pero no el número de pasos de control, con lo cual el área de la UC puede dispararse. Además no tiene en cuenta que el camino crítico depende del tiempo de ciclo, y que existen operadores que no pertenecen a ningún camino crítico, independientemente del tiempo de ciclo, y por tanto no es necesario considerarlos en la selección.

Una mejora de este algoritmo se presenta en [SJWL93]. Se parte de un camino crítico y una serie de operadores con un retardo fijo, y se permite multiciclo y encadenamiento. Se realiza la selección planificando de nuevo el camino para un conjunto de valores del tiempo de ciclo, que se obtiene a partir de los retardos de los operadores. De todos los posibles valores se selecciona el que produce una relación tiempo-área que se adapte mejor a las especificaciones del usuario. Sin embargo, esta nueva planificación podría dar lugar a la aparición de nuevos caminos críticos, distintos a los iniciales, con lo cual los resultados no serían los esperados.

La mayoría de estos sistemas no tienen en cuenta el retardo debido a interconexiones. Weng en [WePa91] considera este retardo realizando una asignación de hardware y un floorplanning después de la planificación. Los operadores del camino crítico se colocan lo más cerca posible de sus predecesores. Se calcula el máximo retardo posible en el camino crítico, teniendo también en cuenta el retardo de las interconexiones. Si las restricciones de tiempo no se cumplen, se realizan cambios en el diseño para asegurar la correcta simulación eléctrica. Sin embargo, este sistema no permite estimar estos retardos sin realizar el floorplanning, que incrementa considerablemente la complejidad

del proceso de síntesis.

Revisados los métodos utilizados por los distintos sistemas de SAN para seleccionar el tiempo de ciclo, vemos que no hay ninguno que cumpla todos los requisitos necesarios: que tenga en cuenta el GFDC, especialmente los nodos que se encuentran en los caminos críticos, la biblioteca de módulos, las especificaciones del usuario y los retardos de los módulos e interconexiones. Por tanto, parece clara la necesidad de generar un algoritmo que considere todos estos factores.

5.3. Minimización del tiempo de ejecución

A priori parece que, para tener un tiempo de ejecución mínimo, es necesario escoger un tiempo de ciclo que produzca tiempos muertos muy pequeños (o nulos en el mejor caso) en cada una de las operaciones del GFD. Sin embargo, una vez realizada la planificación, se observa que sólo tienen influencia en el tiempo total de ejecución los tiempos muertos correspondientes a operaciones situadas en el camino crítico. Antes de la planificación no es posible saber cuál es el camino crítico, ya que depende del tiempo de ciclo, por lo cual es interesante desarrollar un algoritmo para detectar aquellos caminos que, en función del tiempo de ciclo, podrían ser críticos. Eliminar del conjunto de operadores del GFD aquellos que no se encuentren en estos posibles caminos críticos, simplifica los cálculos y conduce más rápidamente a buenos resultados. Esto sobre todo ocurre en GFDs con una gran diversidad de operadores con tiempos de ejecución diferentes. La selección del tiempo de ciclo debe ser tal, que produzca tiempos muertos lo más pequeños posible en las operaciones que se encuentren en los posibles caminos críticos del GFD.

Por tanto, para encontrar el tiempo de ciclo que minimiza el tiempo de ejecución debemos:

- Estimar el retardo de las operaciones del GFD.
- Encontrar los posibles caminos críticos.

- Seleccionar el tiempo de ciclo que minimiza el tiempo muerto de las operaciones a lo largo de los posibles caminos críticos.

En los apartados 5.3.1 al 5.3.3 se explica cómo realizar cada uno de estos tres pasos.

5.3.1. Retardo de las operaciones del GFD

Para estimar el retardo de las operaciones del GFD debemos basarnos en un determinado modelo de ruta de datos. En nuestro sistema de SAN, este modelo (ver figura 5.1) consta de una serie de registros donde están almacenados los datos, y una serie de UFs, que realizan las operaciones. También consta de dos niveles de multiplexores, uno a la entrada de los registros y otro a la de las UFs.

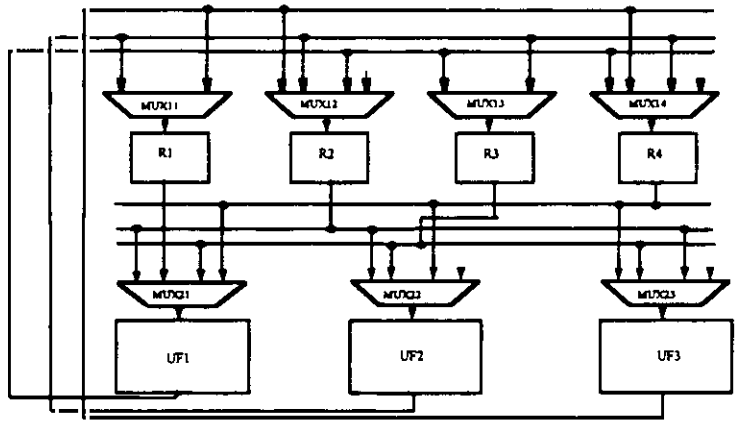


Figura 5.1. Modelo de Ruta de Datos

Las interconexiones parten de la salida de una UF o de un registro, y pueden ir a una o varias entradas de registros y de UFs, respectivamente. Existe la posibilidad de utilizar buses además de multiplexores. En este caso, la salida de un módulo que utilice un bus debería ir conectada a un driver y éste al bus. Como el cálculo del retardo de la transmisión no cambia significativamente si se utilizan buses (sólo hay que añadir el retardo del driver al retardo de la interconexión), vamos a presentar todo el desarrollo para un modelo sólo con multiplexores.

La existencia de interconexiones de registro a registro no influye significativamente en la selección del tiempo de ciclo, porque la transmisión de una señal es siempre mucho más rápida que la ejecución de una operación. Por tanto, para simplificar el modelo, se considera que no existen.

Para este modelo de ruta de datos, vamos a ver qué factores influyen en el tiempo de ejecución de una operación t_{ejec}, que se realiza sobre unos operandos disponibles en registros, y cuyo resultado se almacena en otro registro. Además, se permite que una operación tarde varios pasos de control en ejecutarse. Las fases a seguir y los retardos asociados a ellas son:

a) Transmisión de los operandos a las UFs donde se va a realizar la operación. Esta transmisión consta a su vez de tres retardos:

   - Transmisión desde el registro donde está almacenado el dato al multiplexor colocado a la entrada de la unidad funcional. Este retardo depende de la resistencia de salida del registro, de la capacidad de entrada del multiplexor y de la longitud de la interconexión. Tiene un valor que debe estimarse.

   - El retardo producido por el multiplexor t_{mux}. Es un dato que puede conocerse realizando una simulación eléctrica de los módulos del circuito, y tenerse almacenado en la biblioteca.

   - Transmisión desde el multiplexor a la unidad funcional. Este retardo depende de la resistencia de salida del multiplexor, de la capacidad de entrada de la UF y de la longitud de la interconexión. Tiene un valor que debe estimarse.

b) Ejecución de la operación, t_{UF}. Este retardo depende del tipo de unidad funcional que realice la operación y suponemos que es un dato que se tiene en la

biblioteca. Pueden existir varias alternativas, que serán consideradas a la hora de realizar la selección del tiempo de ciclo.

c) Transmisión del resultado desde la salida de la UF a la entrada del registro donde se va a almacenar. Esta transmisión consta también de tres retardos:

- Transmisión desde la UF al multiplexor colocado a la entrada del registro. Este retardo depende de la resistencia de salida de la UF, de la capacidad de entrada del multiplexor y de la longitud de la interconexión. Tiene un valor que debe estimarse.
- El retardo producido por el multiplexor, $\mathbf{t_{mux}}$.
- Transmisión desde el multiplexor al registro. Este retardo depende de la resistencia de salida del multiplexor, de la capacidad de entrada del registro y de la longitud de la interconexión. Tiene un valor que debe estimarse.

d) Almacenamiento en registro. Este valor también es fijo para una biblioteca dada y se tiene como dato. Se denomina $\mathbf{t_{almac}}$.

Para calcular correctamente $\mathbf{t_{ejec}}$ es necesario estimar de alguna manera los retardos asociados a las interconexiones. Como hemos visto, en cada operación hay cuatro interconexiones implicadas, y su retardo depende de la longitud de cada una de ellas. En principio, el valor asociado a todas ellas lo denominaremos $\mathbf{t_{prop}}$. En el capítulo 6 presentaremos un método para estimar su valor.

Por tanto, el tiempo de ejecución de una operación es:

$$t_{ejec}=t_{prop} + t_{UF} + t_{almac} + 2*t_{mux}$$

Una vez calculados los valores de los tiempos de ejecución de todas las operaciones del GFD pasamos a buscar los posibles caminos críticos.

5.3.2. Búsqueda de los posibles caminos críticos

Llamamos **posible camino crítico** a un camino del GFD que, para algún determinado valor del tiempo de ciclo, sea un **camino crítico**. La búsqueda del tiempo de ciclo óptimo sólo es necesario realizarla con los operadores que se encuentren en los posibles caminos críticos, puesto que los caminos que no son críticos no influyen ni en el tiempo total de ejecución, ni en el número de pasos de control. La eliminación de operadores que no se encuentren en dicho grupo facilita y acelera la búsqueda del valor del tiempo de ciclo óptimo.

En este apartado se presenta un método para generar una lista de posibles caminos críticos, sobre la cual aplicar un algoritmo de selección del tiempo de ciclo. En primer lugar, presentaremos un método suponiendo que los retardos de las operaciones del GFDC son fijos, es decir, que en la biblioteca sólo existe una UF capaz de realizar cada una de ellas. A continuación, en el apartado 5.3.2.4, veremos las modificaciones necesarias para considerar múltiples UFs para cada operación.

La lista de posibles caminos críticos debe tener el mínimo número de elementos posible, puesto que la complejidad del algoritmo de selección es proporcional a dicho número. Con este fin, si dos posibles caminos críticos son iguales, uno de ellos no es necesario colocarlo en la lista. Con el mismo objetivo, si dos caminos A y B son críticos para un determinado tiempo de ciclo, pero para cualquier otro sólo A puede ser crítico, sólo éste último se añadirá a la lista.

Nuestro sistema de SAN permite multiciclo, es decir, que una operación utilice varios pasos de control para ejecutarse. Sin embargo, no está prevista la utilización de encadenamiento, con lo cual una operación debe utilizar al menos un ciclo para ejecutarse. Si cada operación necesita al menos un ciclo de reloj para ejecutarse, el **número de etapas mínimo** para un determinado GFD es el número de operaciones (o nodos) del camino más largo. Llamaremos $\mathbf{N_{nodosmax}}$ al número de nodos del camino o

los caminos más largos del GFD, que como hemos visto coincide con el número de etapas mínimo $\mathbf{N_{etapasmin}}$.

Como actualmente los retardos de los módulos se miden en nanosegundos, y para las bibliotecas utilizadas todos los módulos tienen un retardo superior a 1ns, no parece lógico utilizar un tiempo de ciclo inferior a la unidad, puesto que no se mejora el tiempo de ejecución y se incrementa inútilmente el número de etapas de control. Por otra parte, existe un tiempo de ciclo mínimo posible dado por la tecnología, que en general es inferior a 1ns. El **tiempo de ciclo mínimo utilizado** ($\mathbf{tc_{mu}}$) será el máximo entre el dado por la tecnología y 1ns. Este valor puede modificarse fácilmente para futuras tecnologías, que permitan generar módulos con tiempos de ejecución inferiores a 1ns. En los ejemplos que presentaremos a lo largo de este trabajo tomaremos siempre $\mathbf{tc_{mu}}$ **igual a 1ns**. El número de etapas máximo de un GFD $\mathbf{N_{etapasmax}}$, se obtiene planificando para un tiempo de ciclo igual a tc_{mu}.

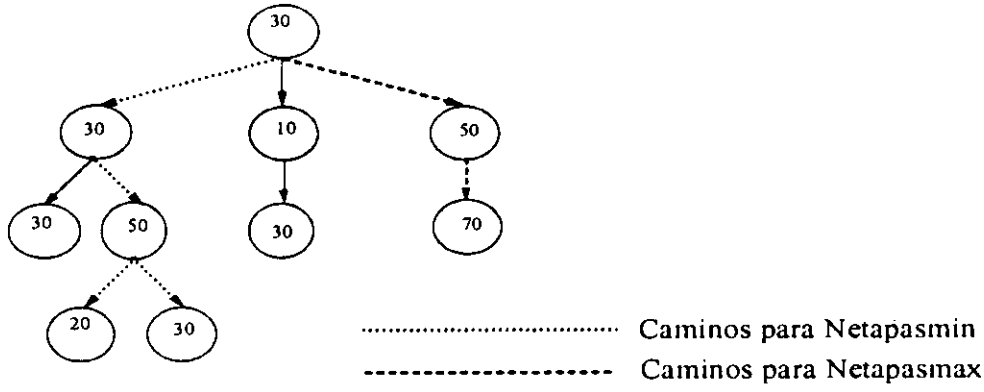


Figura 5.2 Cálculo de $\mathbf{N_{etapasmax}}$ y $\mathbf{N_{etapasmin}}$

En el GFD de la figura 5.2, cada nodo presenta el tiempo de ejecución de la operaciones asociadas a él. También se muestran los caminos que permiten calcular $N_{etapasmax}$ y $N_{etapasmin}$. En primer lugar, $N_{etapasmin}$ viene dada por el máximo número de nodos en un camino. En este ejemplo existen dos caminos que tienen 4 nodos, $N_{nodosmax}=4$ y por tanto $N_{etapasmin}=4$.

En segundo lugar, $N_{etapasmax}$ es el número de etapas del camino que más tarda en ejecutarse para un tiempo de ciclo de 1ns (tc_{mu}). En este caso, para un tiempo de ciclo de 1ns el retardo máximo es de 150ns, y por tanto $N_{etapasmax} = 150$.

Para encontrar los posibles caminos críticos de cualquier GFD, en primer lugar vamos a encontrar todos los caminos e insertarlos en una lista de caminos **LC**. Esta lista la ordenamos colocando primero los caminos con mayor retardo para un tiempo de ciclo igual a tc_{mu}. Si dos caminos tienen el mismo retardo, se coloca primero el que más operaciones tiene.

En el ejemplo de la figura 5.2 existen 5 caminos de datos, por lo tanto LC tendrá 5 elementos, que ordenados de izquierda a derecha según se presentan en la figura, son:

LC= {(30,30,30) (30,30,50,20) (30,30,50,30) (30,10,30) (30,50,70)}

Si esta lista la ordenamos de mayor a menor retardo, obtenemos para un tiempo de ciclo igual a 1ns (tc_{mu}):

LC= {(30,50,70) (30,30,50,30) (30,30,50,20) (30,30,30) (30,10,30)}

A partir de esa lista vamos a generar otra lista de posibles caminos críticos **LPCC**. En el próximo apartado presentamos algunos teoremas que nos servirán de ayuda para construir la LPCC.

5.3.2.1. Teoremas de selección de posibles caminos críticos

En los teoremas que se presentan a continuación se suponen conocidos y fijos los retardos de todas las operaciones del GFDC. Posteriormente veremos qué modificaciones deben realizarse para considerar una biblioteca de módulos con varias UFs que implementen cada operación.

Teorema 1

Todos los caminos que tengan un número de nodos igual a $\mathbf{N_{nodosmax}}$ pueden ser caminos críticos para algún tiempo de ciclo.

Demostración

Si se elige un tiempo de ciclo igual al valor del retardo máximo de todos los operadores del grafo, cada operación tarda exactamente un ciclo en ejecutarse, y el número de etapas totales es $N_{\mathbf{etapasmin}}$. Para este valor del tiempo de ciclo, los caminos con un número de nodos igual a $\mathbf{N_{nodosmax}}$ serán los caminos críticos.

Para el ejemplo de la figura 5.2, con un tiempo de ciclo de 70ns, que es el máximo retardo de todos los operadores del GFD, los caminos críticos serían (30, 30,50,20) y (30,30,50,30), cada uno de los cuales tardaría cuatro ciclos en ejecutarse. Ambos caminos son posibles caminos críticos.

Teorema 2

Sea t_i la suma de los retardos de todos los nodos pertenecientes al camino i para un tiempo de ciclo igual a tc_{mu}. Aquellos caminos con el valor máximo de t_i son posibles caminos críticos.

Demostración

Para un tiempo de ciclo igual a tc_{mu}, los caminos que tienen el máximo valor de t_i, necesitan un número de etapas igual a $\mathbf{N_{etapasmax}}$, y por tanto son caminos críticos.

Estos caminos son los primeros de la LC, puesto que está ordenada de mayor a menor retardo. Para el ejemplo anterior sería (30,50,70).

Teorema 3

Sea **(A,B)** el par ordenado de caminos del GFD. Sean $\{a_1, .., a_n\}$ las operaciones de A y

$\{b_1, .. b_m\}$, las operaciones de B, con $m \geq n$. Si para cada operación $a_i \in A$ se puede encontrar un subconjunto de k operaciones de B, $B_i = \{b_{i1},...,b_{ik}\}$, cuya suma de retardos sea mayor o igual que el retardo de a_i, y siendo los n subconjuntos B_i disjuntos, se puede asegurar que, para cualquier tiempo de ciclo, el tiempo de ejecución de A será menor o igual que el de B y, por tanto, $A \notin$ LPCC.

Demostración

Vamos a demostrar que, para cualquier tiempo de ciclo, el tiempo de ejecución de las k operaciones de B_i es mayor o igual que el de a_i. Si esto es cierto para cualquier i, entonces el retardo de A será menor o igual que el de B para cualquier tiempo de ciclo.

Sea ta_i el retardo de la operación a_i, tb_{ij} el retardo de la operación $b_{ij} \in B_i$ y tB_i la suma de retardos de las k operaciones de B_i.

$$tBi = tbi1 + tbi2 + \ldots + tbik$$

Se sabe que $ta_i \leq tB_i$. Supongamos un tiempo de ciclo tc. Pueden presentarse dos casos:

- $tc > ta_i$. En este caso a_i necesita un ciclo para ejecutarse, mientras que B_i necesita k ciclos como mínimo. Como $k \geq 1$, el retardo de B_i será mayor o igual que el de a_i.

- $tc \leq ta_i$. El número de etapas Na_i que necesita a_i para ejecutarse será

$$Nai = \left\lceil \frac{tai}{tc} \right\rceil$$

y el número de etapas para ejecutar las k operaciones de B_i, NB_i, será:

$$NBi = \left\lceil \frac{tbi1}{tc} \right\rceil + \ldots + \left\lceil \frac{tbik}{tc} \right\rceil \geq \left\lceil \frac{tBi}{tc} \right\rceil \geq \left\lceil \frac{tai}{tc} \right\rceil = Nai$$

Como el número de etapas NB_i es siempre mayor o igual a Na_i, las k operaciones de B_i

requieren el mismo, o más tiempo para ejecutarse que a_i. Por esta razón, concluimos que si B ∈ LPCC entonces A∉ LPCC y el teorema queda demostrado.

Ejemplos

Para clarificar el significado del teorema 3 vamos a presentar algunos ejemplos.

Ejemplo 1 Aplicación del teorema 3 sobre un camino B con 2 sumas y 3 multiplicaciones, y otro camino A con 3 sumas y 2 multiplicaciones.

$$B=\{\text{suma}_{b1}, \text{suma}_{b2}, \text{mult}_{b1}, \text{mult}_{b2}, \text{mult}_{b3}\}$$

$$A=\{\text{suma}_{a1}, \text{suma}_{a2}, \text{suma}_{a3}, \text{mult}_{a1}, \text{mult}_{a2}\}$$

Podemos asociar cada operación de A con una operación de B de la siguiente forma:

$\text{suma}_{a1} \Leftrightarrow \text{suma}_{b1}$ $\text{suma}_{a2} \Leftrightarrow \text{suma}_{b2}$ $\text{suma}_{a3} \Leftrightarrow \text{mult}_{b1}$

$\text{mult}_{a1} \Leftrightarrow \text{mult}_{b2}$ $\text{mult}_{a2} \Leftrightarrow \text{mult}_{b3}$

En cada uno de estos grupos se puede asegurar que, para cualquier tiempo de ciclo, el tiempo de ejecución de la operación de B es mayor o igual que el de la operación de A, puesto que una multiplicación necesita igual o mayor tiempo para realizarse que una suma. Si para cada operación de A existe una operación de B, que necesita igual o mayor tiempo para ejecutarse, y A y B pertenecen al mismo grafo, se puede asegurar que A no influye en el tiempo total de ejecución y no es necesario insertarlo en la LPCC.

Hay que notar que, en este caso, A puede tardar en ejecutarse el mismo tiempo que B, si se escoge un tiempo de ciclo igual al de la multiplicación (o mayor). Entonces ambos caminos necesitan cinco etapas para ejecutarse. Sin embargo, para cualquier tiempo de ciclo menor, A tardará menos que B. Como queremos que la LPCC tenga el mínimo número de elementos posibles, A no se introduce en la lista.

□

Ejemplo 2 Aplicación del teorema 3 sobre el grafo de la figura 5.3

En el ejemplo 1, hemos podido asociar cada operación de A con una única operación de B. Para el ejemplo de la figura 5.3, es necesario asociar cada operación de A, a_i, con un subconjunto de operaciones de B, cuya suma de retardos sea mayor o igual que el retardo de a_i.

La asociación realizada ha sido

$a_1 \Leftrightarrow B_1=\{b_1,b_4\}$ $ta_1=6ns$, $tB_1=4ns+2ns=6ns$

$a_2 \Leftrightarrow B_2=\{b_2,b_5\}$ $ta_2=5ns$, $tB_2=4ns+1ns=5ns$

$a_3 \Leftrightarrow B_3=\{b_3\}$ $ta_3=3ns$, $tB_3=4ns$

En todos los casos puede comprobarse que $ta_i \leq tB_i$.

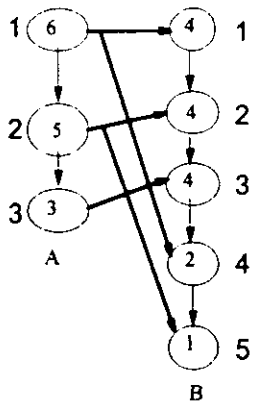


Figura. 5.3 Eliminación de posibles caminos críticos

Vamos a comenzar por el grupo a_1, B_1.

- Si se utiliza un tiempo de ciclo mayor o igual que seis, que es el retardo de a_1, ésta necesita 1 ciclo para ejecutarse, mientras que B_1 necesita dos ciclos, uno para ejecutar b_1 y otro para b_4. Por tanto las 2 operaciones de B necesitan más ciclos que a_1.

- Si se utiliza un tiempo de ciclo inferior a 6ns, existen las siguientes posibilidades:

$tciclo = 5ns$	$Na_1 = 2$	$NB_1 = 1 + 1 = 2$
$tciclo = 4ns$	$Na_1 = 2$	$NB_1 = 1 + 1 = 2$
$tciclo = 3ns$	$Na_1 = 2$	$NB_1 = 2 + 1 = 3$
$tciclo = 2ns$	$Na_1 = 3$	$NB_1 = 2 + 1 = 3$
$tciclo = 1ns$	$Na_1 = 6$	$NB_1 = 4 + 2 = 6$

En cualquiera de los casos anteriores, el número de etapas para ejecutar b_1 y b_4 es mayor o igual que para ejecutar a_1. Un análisis similar puede obtenerse para el grupo a_2 y B_2, y para a_3, B_3. Por lo tanto, para cada operación de A existen un grupo de operaciones de B que necesitan igual o más ciclos para ejecutarse, independientemente del ciclo de reloj, y por tanto A no debe pertenecer a la lista de posibles caminos críticos.

□

Ejemplo 3 *Aplicación del teorema 3 sobre el los caminos del grafo de la figura 5.4.*

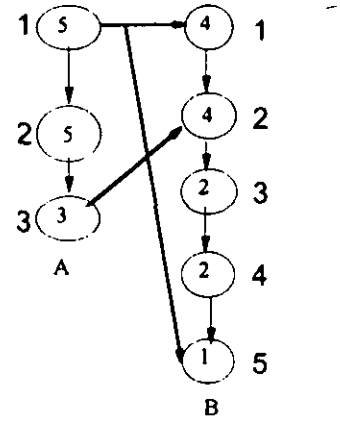


Figura. 5.4

En este caso no se puede decir que el nodo 2 del camino A tiene un retardo inferior que los nodos 3 y 4 del camino B para cualquier tiempo de ciclo, y no existe ninguna posible reagrupación de nodos que nos permita asegurar que el camino A tiene una duración

igual o inferior que el camino *B* para cualquier tiempo de ciclo. Por lo tanto, no se puede eliminar ninguno de los dos caminos como posibles caminos críticos.

□

Corolario 1

Si dos caminos tienen el mismo número de operaciones de cada tipo, implementadas en igual o distinto orden, uno de ellos se puede eliminar de la lista de posibles caminos críticos.

La demostración de este corolario puede realizarse aplicando el teorema 3, eligiendo para cada operación de *A*, una operación de *B* del mismo tipo. Puede asegurarse que, para cualquier tiempo de ciclo, la duración de los dos caminos es la misma y, por tanto, se puede dejar uno de ellos como posible camino crítico y eliminar el otro de la lista.

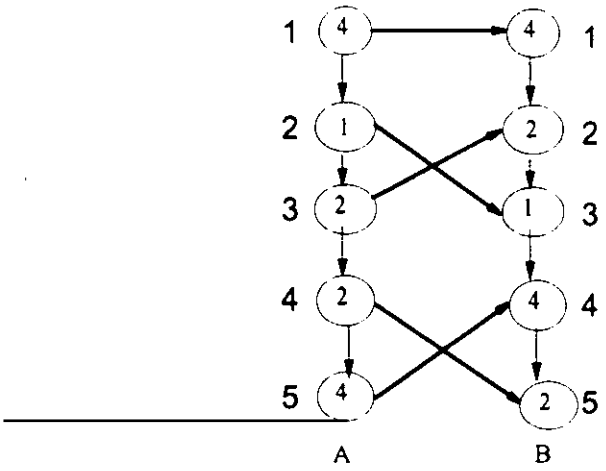


Figura 5.5 Caminos equivalentes

Por ejemplo, para los dos caminos de datos que se muestran en la Figura 5.5 los retardos son iguales, puesto que existe una correspondencia (flechas cruzadas) 1 a 1 entre las operaciones de ambos caminos.

Corolario 2

Si dos caminos *A* y B tienen el mismo retardo para un tiempo de ciclo igual a tc_{mu}, y tienen el mismo número de operaciones, pero no existe una correspondencia unívoca entre los retardos de las operaciones de *A* y las de B, existe la posibilidad de que ambos sean caminos críticos para distintos tiempos de ciclo.

Demostración

La demostración de este corolario vamos a realizarla con un ejemplo. Sea la figura 5.6. Los caminos *A* y *B* tienen un retardo de 18ns y un total de 5 operaciones. Sin embargo, no existe una correspondencia unívoca entre los retardos de las operaciones de *A* y *B*, puesto que las operaciones a_2 y a_3 no tienen su análoga en *B*, y b_2 y b_3 no tienen su análoga en *A*. Si elegimos un tiempo de ciclo igual a 4ns, *A* necesita 6 ciclos, mientras que *B* necesita 5, por tanto *A* sería camino crítico. Sin embargo, si elegimos un tiempo de ciclo de 3, *B* necesita 9 ciclos y *A* necesita 8. En este caso *B* sería camino crítico. Por tanto, dependiendo del tiempo de ciclo, uno u otro pueden ser caminos críticos, y ambos deben estar en la LPCC.

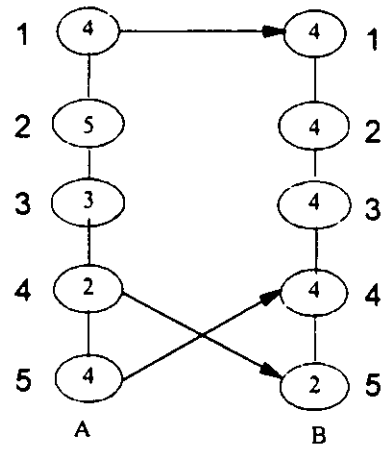


Figura 5.6 Caminos no equivalentes

5.3.2.2. Algoritmo de Unificación de Caminos

El teorema 3 nos permite, para un par de caminos (A,B), eliminar A como posible camino crítico si se consigue asociar o unificar cada operación de A con una o varias de B. Esto es lo que denominamos **unificación de caminos**. Si para cada operación de A tratamos de encontrar una solución realizando una búsqueda exhaustiva entre las operaciones de B, nos encontramos ante un problema NP completo. Sin embargo, nuestro objetivo principal no es eliminar totalmente todos los caminos no críticos, sino eliminar el mayor número de operadores posibles, que se encuentren en caminos nunca críticos, con el fin de facilitar y acelerar la selección del tiempo de ciclo. Por tanto, una búsqueda exhaustiva de soluciones complica el problema de selección en lugar de simplificarlo.

Por eso se plantea la necesidad de encontrar un algoritmo de unificación que cumpla una serie de requisitos:

- No elimine nunca un camino que para algún tiempo de ciclo sea crítico.
- Elimine un porcentaje considerable de caminos no críticos.
- Tenga una complejidad moderada.

Una información muy valiosa que se puede utilizar para acelerar el algoritmo es la que se obtiene de la biblioteca de módulos. En los caminos de datos sólo pueden aparecer tiempos de ejecución de UFs que existan en la biblioteca de módulos. Por esta razón, resulta interesante realizar un estudio de las posibles asociaciones entre las operaciones de cada biblioteca de módulos, y ordenarlas según un criterio que guíe correctamente la unificación.

El **algoritmo de unificación** de un par de caminos (A,B) recibe como información de entrada una **tabla de posibles unificaciones TPU**, donde las distintas opciones están ordenadas según un criterio determinado. Existe una entrada a la tabla para cada

operación que pueda haber en el grafo. Inicialmente se unifican las operaciones comunes. A continuación se recorre A ordenadamente, comenzando con la operación con mayor tiempo de ejecución. Para cada operación se escoge la primera unificación de la lista que se pueda utilizar (según las operaciones de *B*), y se pasa a la siguiente operación de *A*. Si en dicha lista no hay ninguna opción posible, se recorren los operadores de B de mayor a menor tiempo de ejecución, y se van cogiendo todos los operadores existentes, no unificados, hasta conseguir un retardo mayor o igual que el de la operación de *A*. Si no se consigue, *A* se inserta en la LPCC.

La tabla de posibles unificaciones es necesario realizarla para cada biblioteca de módulos que se pueda utilizar. Con esta información, para cada operación de A se conoce exactamente con qué operaciones de B debe unificarse. Así el algoritmo de unificación en lugar de ser NP completo, será lineal con el número de operaciones de A. Aunque esta solución no consigue siempre la unificación óptima, si el criterio de ordenación dirige correctamente la búsqueda, podemos encontrar una solución, si la hay, en la mayoría de los casos. Además, aseguramos que todos los caminos que son críticos para algún tiempo de ciclo se insertarán en la LPCC.

A continuación vamos a ver cómo generar la tabla de posibles unificaciones. Dada una biblioteca de módulos hay que encontrar las posibles asociaciones entre sus operaciones y ordenarlas en función de algún criterio que guíe correctamente la unificación global de dos caminos.

A partir de la biblioteca construimos la lista de operaciones con sus retardos asociados:

$$L=\{op_1, ..., op_k\}$$

Suponemos que las operaciones están ordenadas de mayor a menor retardo en la biblioteca. Para cada operación op_i debemos obtener una serie de posibles unificaciones, con las op_j $(j \neq i)$. Estas posibles unificaciones se ordenan de tal forma que permitan el mayor número de posibilidades para el resto de las unificaciones, utilizando el siguiente

criterio:

- **Paso 1**: Se eligen las posibles unificaciones con una única operación op_j que tenga mayor o igual retardo que op_i Las opciones posibles de unificación con una única operación no es necesario ordenarlas. Como los nodos en A se recorren de mayor a menor retardo, para un nodo a_i no existen en A operaciones con mayor retardo que no estén unificadas y, por tanto, seleccionar una única operación de B con mayor retardo que a_i siempre será una opción que conduce a una solución, si existe.

- **Paso 2**: Se proponen las unificaciones con dos o tres operaciones. Permitir todas las posibles combinaciones con más de tres operaciones incrementa la complejidad del algoritmo y, para los ejemplos de GFDs presentados en la literatura, no se ha visto que incremente su bondad en la misma medida. Sin embargo, este algoritmo es fácilmente modificable para permitir un mayor número de operaciones en una combinación.

En el paso 2, las distintas opciones de dos o tres operaciones se ordenan siguiendo dos criterios:

- **El retardo total**. Se ordenan de menor a mayor retardo total.

- **El número de operaciones**. Se ordenan de menor a mayor número de operaciones.

Si se plantea el caso de que exista una opción con menor número de operaciones que otra, y mayor o igual retardo total, para ordenarlas seguiremos el siguiente criterio:

1.-Se elimina de las dos opciones la operación con más retardo.

2.-Se suman los retardos del resto de las operaciones de cada una de las opciones.

3.-La opción que tenga esta suma menor, se coloca primero. Si ambas sumas son iguales, se realiza el mismo proceso con la siguiente operación con más retardo.

Ejemplos

Ejemplo 1: Ordenación de posibles opciones de unificación

Supongamos op_i=35ns y las posibles unificaciones

$$35 \Rightarrow (30,8)\ (20,10,5).$$

Eliminamos de ambas opciones la primera operación y nos queda

$$(8)\ (10,5).$$

Como la primera opción tiene de suma 8 y la segunda 15, colocamos la opción (30,8) antes que la (20,10,15).

□

Ejemplo 2. Generación de la TPU

Sea la biblioteca de módulos que se presenta en la tabla 5.1.

modulo	op	retardo
sumador	+	20ns
restador	-	21ns
multiplicador	*	100ns
divisor	/	120ns

Tabla 5.1 Biblioteca de Módulos

La lista de operaciones ordenadas según el tiempo de ejecución es:

$$L=(120,100,21,20)$$

Veamos las unificaciones posibles para la división. No es posible encontrar ninguna unificación con una única operación. Pasamos a las posibilidades con 2 operaciones.

$$120 \quad \Leftrightarrow \quad (100,20)\ (100,21)\ (100,100)$$

No existe ninguna posibilidad con tres operaciones, por lo tanto el conjunto de posibles unificaciones para la división es el que aparece en la lista anterior.

La multiplicación se puede unificar con una división. No existe ninguna solución con dos o tres operaciones.

$$100 \quad \Leftrightarrow \quad (120)$$

La resta se puede unificar con una multiplicación, con una división o bien con dos sumas. La soluciones ordenadas son:

$$21 \quad \Leftrightarrow \quad (100)\ (120)\ (20,20)$$

Por último la suma se puede unificar con cualquier operación.

Todas las unificaciones para cada operación de la biblioteca se encuentran en la tabla 5.2.

/	*	-	+
(*,+)	(/)	(*)	(-)
(*,-)		(/)	(*)
(*,*)		(+,+)	(/)

Tabla 5.2 TPU para la biblioteca de la tabla 5.1

□

Mediante la generación de la TPU, hemos conseguido un conjunto de unificaciones para cada operación, que nos marcan el camino a seguir en la unificación entre dos caminos. Para cualquier par de caminos *(A,B)* que tengan operaciones implementadas con módulos de esa biblioteca, se realizará la unificación recorriendo los nodos de A de mayor a menor retardo, y tratando de realizar las unificaciones dadas en la tabla, en el orden señalado.

Este algoritmo de generación de la TPU sólo es necesario ejecutarlo cuando se va a utilizar una nueva biblioteca de módulos.

5.3.2.3. Algoritmo de búsqueda de posibles caminos críticos

Basándonos en los teoremas de selección de posibles caminos críticos, en la TPU y en el algoritmo de unificación, obtenemos el algoritmo de búsqueda de caminos críticos para un GFDC con retardos fijos para cada una de las operaciones. Partimos de una lista de caminos del GFD. El algoritmo se divide en tres partes:

- **Paso 0:** **Creación de la tabla de posibles unificaciones** para la biblioteca de módulos que se esté utilizando. Este paso sólo es necesario realizarlo si la biblioteca es nueva.

- **Paso 1**: **Ordenación de la LC**. Todos los caminos de la LC se ordenan de mayor a menor retardo para un tiempo de ciclo igual a tc_{mu}. Si dos caminos tienen el mismo retardo, se coloca primero el de mayor número de nodos. Dentro de cada camino, los nodos se ordenan de mayor a menor retardo. Al mismo tiempo se realiza la **reducción de la LC,** porque si existen dos caminos de datos con el mismo retardo y el mismo número de operaciones de cada tipo, por el corolario 1 uno de ellos puede eliminarse.

- **Paso 2:** **Insertar el primer elemento de la LC en la LPCC**. Por el teorema 2, el primer elemento de la LC, que tiene un retardo máximo para un

tiempo de ciclo igual a tc_{mu}, es un posible camino crítico. Como la LPCC está vacía, no existe ningún camino que permita aplicar el teorema 3, y por tanto este camino se inserta en la LPCC.

- **Paso 3: Resto de posibles caminos críticos.** Sea A un elemento de LC. Si no existe ningún camino B de la LPCC, tal que el par (A,B) cumpla el teorema 3 utilizando el algoritmo de unificación, A debe añadirse a la LPCC. El algoritmo recorre ordenadamente la LC y termina cuando queda vacía. Sobre las operaciones pertenecientes a la LPCC aplicamos el algoritmo de selección del tiempo de ciclo.

El primer paso, la ordenación de la LC, no sólo es útil para facilitar la reagrupación de los nodos, sino que es crucial para obtener resultados correctos, puesto que sólo si se recorre la LC de forma ordenada, se puede asegurar que para cualquier camino A que se inserta en la LPCC no existe en la LC otro camino B tal que el par (A,B) cumple el teorema 3. Si esto no fuera así, sería necesario sacar A de la LPCC, con lo cual el algoritmo no funcionaría correctamente, ya que una vez insertado un elemento en la LPCC nunca se extrae.

Para demostrar que esta afirmación es correcta, veamos que no se puede dar el caso contrario. Sea A un camino de la LPCC. Si existiera en LC algún camino B tal que el par (A,B) cumpliera el teorema 3, el camino B tendría un retardo total mayor o igual que el de A, y un número de operaciones también mayor o igual que el A. Como los caminos de LC se ordenan de mayor a menor retardo para un tiempo de ciclo igual a tc_{mu}, se puede asegurar que no existe ningún camino en la LC con mayor retardo que A para cualquier tiempo de ciclo, puesto que, al menos para un tiempo de ciclo igual a tc_{mu}, es el que más retardo tiene. Por tanto, sólo podría ocurrir que B tuviera el mismo retardo que A para un tiempo de ciclo igual a tc_{mu}, y un número de operaciones menor o igual. Si es menor, no se puede aplicar el teorema 3. Si es igual, por el corolario 2 ambos pueden pertenecer a la LPCC, y no hay que sacar A de ésta.

El algoritmo en pseudocódigo será:

Obtención LC
Ordenación y Reducción de LC
B=Sacar_cabecera(LC)
Insertar B en LPCC
Mientras haya caminos en LC
    A=Sacar_cabecera(LC)
    Si no existe otro camino B en LPCC | (A,B) cumpla Teorema 3
        Insertar A en LPCC

Ejemplos

Ejemplo 1 Algoritmo de búsqueda de posibles caminos críticos sobre el grafo de la figura 5.2.

LC={(30,50,70) (30,30,50,30) (30,30,50,20) (30,30,30) (30,10,30)}

La TPU correspondiente a la biblioteca que se está utilizando se muestra en la tabla 5.3. La lista con los caminos ordenados sería:

LC={(70,50,30) (50,30,30,30)(50,30,30,20) (30,30,30) (30,30,10)}

No se puede reducir la lista de caminos, puesto que no existen dos caminos con las mismas operaciones.

Insertamos el primer elemento en la LPCC

LPCC= {(70,50,30)}

Para cada camino de la LC, lo sacamos de ésta y buscamos si existe algún camino de la LPCC que permita aplicar el teorema 3. Si no es así lo insertamos en la LPCC.

LC={(50,30,30,30) (50,30,30,20) (30,30,30) (30,30,10)}

LPCC= {(70,50,30)}

100	70	60	50	30	20	10
(70,30)	(100)	(100)	(60)	(50)	(30)	(20)
(50,50)	(60,10)	(70)	(70)	(70)	(70)	(70)
(70,20,10)	(50,20)	(50,10)	(100)	(60)	(50)	(30)
(60,30,10)	(60,20)	(30,30)	(30,20)	(100)	(60)	(50)
(60,20,20)	(50,10,10)	(50,20)	(30,10,10)	(20,10)	(100)	(60)
(70,20,20)	(50,30)	(50,30)	(30,30)	(20,20)	(10,10)	(100)
(60,50)	(60,30)	(30,20,10)	(20,20,10)	(10,10,10)		
(70,50)	(30,30,10)	(20,20,20)	(20,20,20)			
(60,30,20)	(30,20,20)	(30,20,20)				
(60,60)	(50,50)	(50,50)				
(70,60)	(60,50)					
(60,30,30)	(30,30,20)					
(50,30,20)	(60,60)					
(50,30,30)	(30,30,30)					
(70,70)						

Tabla 5.3 TPU para el la biblioteca del grafo de la figura 5.2

El camino (50,30,30,30) tiene 4 nodos, y no existe en LPCC ningún elemento con 4 o más nodos. Por tanto ningún elemento puede cumplir el teorema 3.

LPCC= {(70,50,30) (50,30,30,30)}

El camino (50,30,30,20) no lo incluimos en la LPCC porque se puede unificar con (50,30,30,30). Primero asociamos operaciones comunes y a continuación utilizamos las de la tabla 5.3.

$50 \Rightarrow 50 \qquad 30 \Rightarrow 30 \qquad 30 \Rightarrow 30 \qquad 20 \Rightarrow 30.$

LC={(30,30,30) (30,30,10)}

LPCC= {(70,50,30) (50,30,30,30) }

El siguiente camino de LC, A=(30,30,30) lo unificamos con B=(70,50,30):

$30 \Rightarrow 30 \qquad 30 \Rightarrow 50 \qquad 30 \Rightarrow 70$

Para el siguiente camino de LC, A=(30,30,10) existe el camino B=(70,50,30), que permite aplicar el teorema 3. La unificación puede realizarse:

$30 \Rightarrow 30 \qquad 30 \Rightarrow 50 \qquad 10 \Rightarrow 70$

Por lo tanto, no queda ningún camino en la LC que pertenezca a la LPCC. Y de esta forma obtenemos:

LPCC= {(70,50,30) (50,30,30,30) }

□

Ejemplo 2 Algoritmo de búsqueda de posibles caminos críticos sobre el grafo de la figura 5.7

En este grafo sólo se representan los retardos de las operaciones, y la biblioteca de módulos es la misma que para el ejemplo anterior.

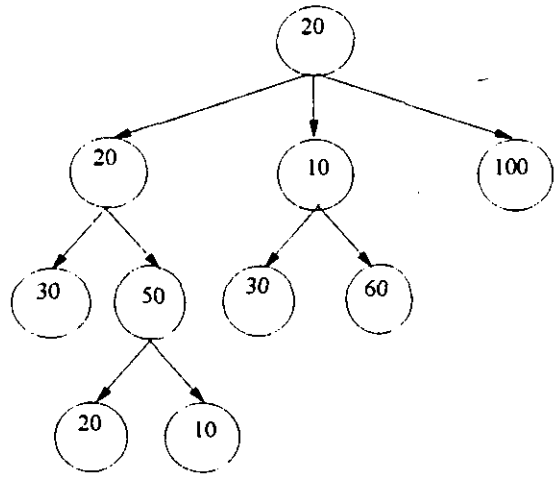


Figura. 5.7

La lista de caminos de este grafo ordenada de mayor a menor retardo es la siguiente:

LC={(20,100) (20,20,50,20) (20,20,50,10) (20,10,60) (20,20,30) (20,10,30)}

Ordenamos los nodos de cada camino de mayor a menor retardo:

LC={(100,20) (50,20,20,20) (50,20,20,10) (60,20,10) (30,20,20) (30,20,10)}

No existen dos caminos con el mismo número de operaciones de cada tipo, luego no se puede reducir LC. En esta lista tenemos: $N_{nodosmax}$=4 y el máximo retardo para tiempo de ciclo=1 es 120. Por lo tanto inicialmente:

LPCC={(100,20) }

El camino (50,20,20,20), tiene un número de nodos igual a 4, y el único camino de LPCC tiene 2 nodos, por tanto no puede aplicarse el teorema 3 y debe insertarse en la lista.

LPCC={(100,20),(50,20,20,20)}

El camino (50,20,20,10), que también tiene un número de nodos igual a 4, lo unificamos con el camino (50,20,20,20).

50⇒50 20⇒20 20⇒20 10⇒20

Con esto nos queda

LC={(60,20,10) (30,20,20) (30,20,10)}

Extraemos el camino (60,20,10), y tratamos de unificarlo con (100,20). Como éste último tiene 2 nodos y el primero 3, no es posible. Pasamos a unificarlo con (50,20,20,20) según se muestra en la figura 5.8.

Ahora nos queda LC={(30,20,20) (30,20,10)}. Unificamos *A*=(30,20,20) con *B*= (50,20,20,20) según:

30⇒50 20⇒20 20⇒20

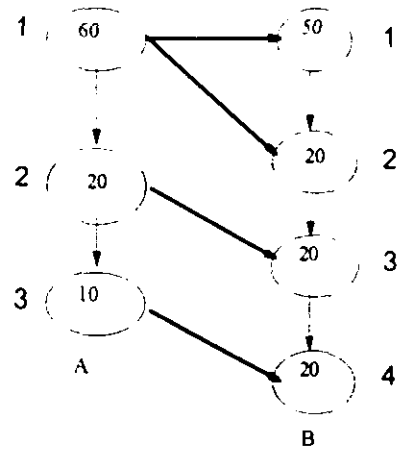


Figura. 5.8

y no es necesario insertar *A* en la LPCC. Por último unificamos (30,20,10) con (50,20,20,20) y tenemos que:

$$30 \Rightarrow 50 \qquad 20 \Rightarrow 20 \qquad 10 \Rightarrow 20$$

y tampoco lo insertamos en LPCC. Por lo tanto sólo existen 2 posibles caminos críticos:

$$LPCC=\{(100,20)\ (50,20,20,20)\}$$

sobre los que aplicaremos el algoritmo de selección del tiempo de ciclo.

□

Por otra parte, hemos comprobado experimentalmente, que los retardos de las operaciones lógicas no deben tenerse en cuenta para calcular el ciclo de reloj, porque son muy pequeños frente al retardo del resto de los operadores, con lo cual los resultados de tiempo de ciclo obtenidos por nuestro algoritmo suelen ser menores que los óptimos. Esto produce un incremento en el número de etapas necesarias para realizar el resto de los operaciones. El hecho de no tenerlos en cuenta en especificaciones donde predomina el número de operaciones aritméticas, favorece la obtención de tiempos de ciclo óptimos.

Un último problema que se presenta en la búsqueda de caminos críticos es la posibilidad

de disponer de una biblioteca de módulos más amplia, donde cada operación del GFD puede ser implementada por más de un operador. En este caso, los retardos de los operadores no estarían prefijados. En el apartado 5.3.2.4 se presenta un algoritmo modificado que tiene en cuenta este factor.

5.3.2.4. Algoritmo modificado de búsqueda de caminos críticos

Si existe una biblioteca de módulos con más de una UF para implementar cada operación, sería necesario utilizar el algoritmo de búsqueda de posibles caminos críticos para cada una de las UFs posibles, lo cual aumentaría significativamente la complejidad del algoritmo.

Sin embargo, podemos modificar levemente el algoritmo de búsqueda para no incrementar tanto su complejidad.

- **Paso 0:** **Generación de la TPU.** No es necesario tener en cuenta todas las UFs de la biblioteca. Consideramos dos listas: una lista $\mathbf{UF_{Lentas}}$ con las UFs más lentas para implementar cada operación del GFD, y otra lista $\mathbf{UF_{Rapidas}}$ con las más rápidas. Se trata de generar una TPU que contenga las posibles unificaciones entre cada elemento de la lista de UF_{Lentas}, con los elementos de la lista de $UF_{Rapidas}$, teniendo en cuenta los criterios dados para el algoritmo de unificación.

- **Paso 1:** **Búsqueda de caminos**. En esta fase cada nodo del grafo se trata como una operación sin un retardo fijo.

- **Paso 2:** **Ordenación y reducción de LC**. Se hace de la misma forma que en el algoritmo inicial, pero ordenando todos los nodos de los caminos de la LC de mayor a menor retardo, utilizando para cada uno de ellos la UF más lenta de la biblioteca que pueda implementar la operación. Si dos caminos son iguales, uno de ellos se elimina.

- **Paso 3:** **Se inserta el primer elemento de LC en la LPCC.** El razonamiento es análogo al del algoritmo inicial. En la LPCC el retardo de cada operación se toma como el de la UF más rápida que pueda implementarlo.

- **Paso 4:** **Resto de posibles caminos críticos**. El algoritmo recorre ordenadamente la LC, toma el camino A de la cabecera, y busca si existe algún elemento B de la LPCC tal que el par (A,B) cumpla el teorema 3. Si no es así, el camino A se inserta en la LPCC, sustituyendo los retardos de sus operaciones por los retardos de las UFs más rápidas que puedan implementarlas.

La razón por la cual sustituimos cada operación de la LPCC por el retardo de la UF más rápida que pueda realizarlos, es para asegurar que en las comparaciones entre un camino *A* de la LC y un camino *B* de la LPCC, tenemos en cuenta todas las posibilidades. El caso más desfavorable se produce cuando las operaciones de *A* se implementan con las UFs más lentas, y las de *B* con las más rápidas. Si en ese caso *A* no puede ser camino crítico, en ningún otro caso lo será.

Ejemplo.

Vamos a realizar la búsqueda de caminos críticos sobre el GFD del algoritmo de Resolución de la Ecuación Diferencial [PaKG86], cuyo GFD se muestra en la figura 5.9. Vamos a utilizar la biblioteca de UFs que se presenta en la tabla 5.4. Para esta biblioteca presentamos la TPU en la tabla 5.5.

La lista de caminos es:

LC={(*,+), (*,*,-,-), (*,*,-,-), (*,*,-) (+,=)}

Eliminando los caminos que son iguales y las operaciones lógicas, y ordenando la lista según los retardos de las UFs más lentas de la biblioteca obtenemos:

LC={ (*,*,-,-), (*,*,-),(*,+), (+)}

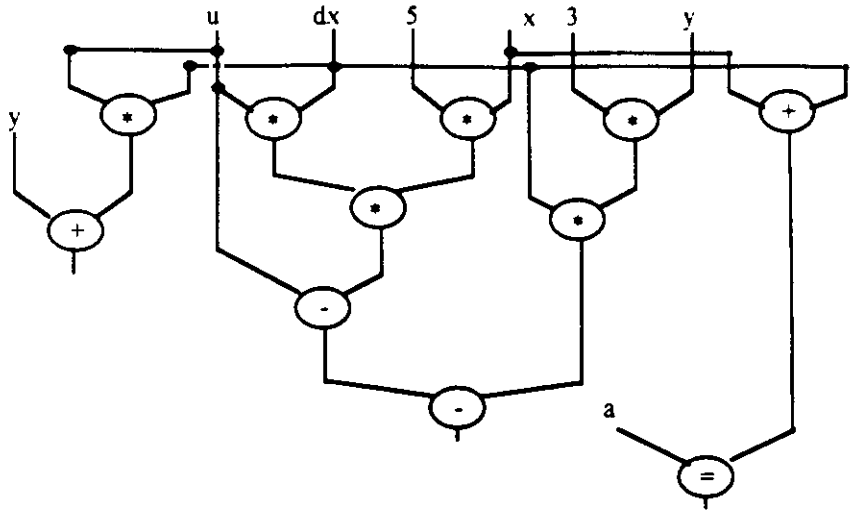


Figura. 5.9 GFD de la Ecuación Diferencial

Por tanto: LPCC={ (*,*,-,-)}

modulo	op1	retardo	op2	retardo
sumador	+	20ns		
restador	-	21ns		
multiplicador	*	100ns		
divisor	/	103		
sumador/restador	+	21ns	-	25ns

Tabla 5.4 Biblioteca de Módulos

Tomamos ahora el camino (*,*,-). Como el camino que se encuentra en la LPCC tiene dos multiplicaciones y dos restas, eliminando de ambos las operaciones comunes podemos unificarlos.

Tomemos ahora (*,+). Unificamos las operaciones comunes (una multiplicación de cada camino). Según la TPU la suma se unifica con la siguiente multiplicación.

/ (103ns)	* (100ns)	- (25ns)	+(21ns)
(*,+) (100ns+20ns)	(/) (103ns)	(/) (103ns)	(/) (103ns)
(*,-) (100ns+21ns)		(*) (100ns)	(*) (100ns)
(*,*) (100ns+100ns)		(+,+) (20ns+20ns)	(-) (21ns)

Tabla 5.5 TPU para la biblioteca de la tabla 5.4

Por último, tomamos (+), y su única operación se unifica con una de las multiplicaciones. Por tanto:

LPCC={ (*,*,-,-)}

5.3.3. Obtención del tiempo de ejecución mínimo

Una vez conocidos los tiempos de ejecución de todas y cada una de las operaciones del diseño, el tiempo de ciclo que produce tiempos muertos nulos es cualquier divisor común a dichos tiempos de ejecución. Si sólo tenemos en cuenta los elementos de la LPCC, será un divisor común de los tiempos de ejecución de las operaciones de dichos caminos. Cualquier planificación que minimice el número de pasos de control (por ejemplo la planificación ASAP) y utilice como tiempo de ciclo uno de estos divisores comunes, produce un tiempo de ejecución mínimo, puesto que no existen tiempos muertos en ninguno de los pasos de control. Como se trata de minimizar también el número de etapas de control, el máximo común divisor parece el valor óptimo entre los divisores.

En general, cualquier especificación del comportamiento de un circuito suele tener operaciones con tiempos de ejecución muy diferentes, y el máximo común divisor de todos esos valores es la unidad o próximo a ella. Un tiempo de ciclo tan pequeño da lugar a muchos pasos de control y, en consecuencia el tamaño de la Unidad de Control puede crecer de forma excesiva. Además, la potencia disipada aumenta inversamente al

tiempo de ciclo, y para tiempos demasiado pequeños, puede dar lugar a un mal funcionamiento del circuito, debido a la propagación de señales. Por eso, en la mayoría de los casos, no es posible realizar esta simple selección para el tiempo de ciclo, y es necesario considerar en dicha selección los factores área y potencia disipada. En el apartado 5.4 presentamos los factores que influyen en el tamaño de la UC.

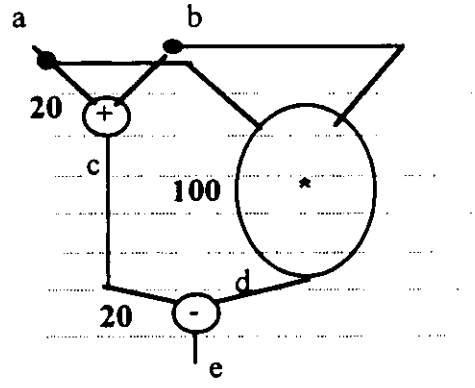
Por otra parte, antes de realizar la asignación de hardware no se sabe qué tipo de UF va a ejecutar cada una de las operaciones, y por tanto, tampoco se sabe el tiempo de ejecución de cada operación. Es necesario realizar un estudio del grafo y considerar qué tipos de UFs son las que pueden utilizarse, según los objetivos y restricciones del usuario. En el apartado 5.5 se presenta un algoritmo de selección del tiempo que tiene en cuenta estos factores, y además considera el tiempo total de ejecución y el área del circuito.

5.4. Minimización del área de la Unidad de control

El tamaño de una Unidad de Control crece con el número de estados correspondiente a la especificación del circuito, para un hardware y una planificación determinados. Si escogemos un tiempo de ciclo pequeño, el número de estados aumenta, y por tanto el tamaño de la Unidad de Control. Sin embargo, el área y tiempo no siempre son contrapuestos. Es posible realizar una selección inteligente del tiempo de ciclo, que tenga en cuenta los factores de área y tiempo de ejecución del circuito, y así conseguir soluciones que cumplan los objetivos de minimización dados por el usuario. Para ilustrar este hecho suponemos la siguiente especificación del comportamiento de un circuito:

```
program ejem
port a,b;
var c, d;
bgn
   c:=a+b;
   d:=a*b;
   e:=c-d
end
```

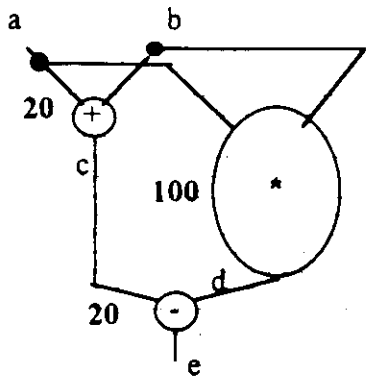
La biblioteca de módulos dispone de un multiplicador, de un sumador y de un restador para realizar las operaciones. Supongamos que, teniendo en cuenta los retardos de todas las fases de una operación explicadas en el apartado 5.3.1, el sumador y el restador tienen un retardo de 20ns y el multiplicador 100ns. Si se utiliza un tiempo de ciclo de 20ns (figura 5.10) se necesitan 5 etapas de control para realizar la multiplicación y 1 para la suma y para la resta. Como la suma y la multiplicación se pueden realizar en paralelo, en total necesitamos 6 pasos de control y el tiempo total de ejecución es 120ns.



ciclo 20ns
ttotal 120ns n° etapas6
FU: SUM(+), MUL(*), SUB (-)

Figura. 5.10

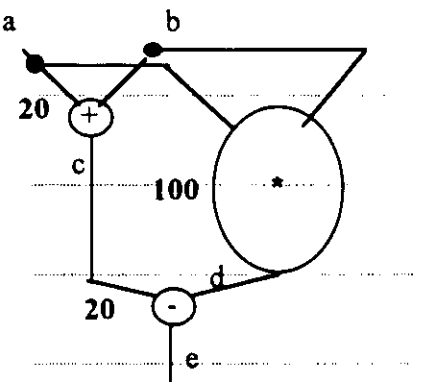
Sin embargo, si utilizamos un tiempo de ciclo de 10ns (figura 5.11), necesitamos dos pasos de control para realizar la suma y la resta, y 10 para la multiplicación. Como la suma y la multiplicación pueden realizarse en paralelo, en total necesitamos 12 etapas de control, y el mismo tiempo de ejecución total.



ciclo 10ns
ttotal 120ns n° etapas12
FU: SUM(+), MUL(*), SUB (-)

Figura. 5.11

En este caso, parece mucho mejor tomar como tiempo de ciclo 20ns puesto que no se incrementa el tiempo de ejecución y se minimiza el tamaño de la Unidad de Control. Si se utilizan tiempos de ciclo divisores de 20 (por ejemplo 10, 5, 4, etc.), lo único que se consigue es incrementar el número de microinstrucciones necesarias para realizar las 3 operaciones, sin mejorar el tiempo de ejecución total.



ciclo 50ns
ttotal 150ns netapas 3
UF: SUM(+), MUL(*), SUB (-)

Figura. 5.12

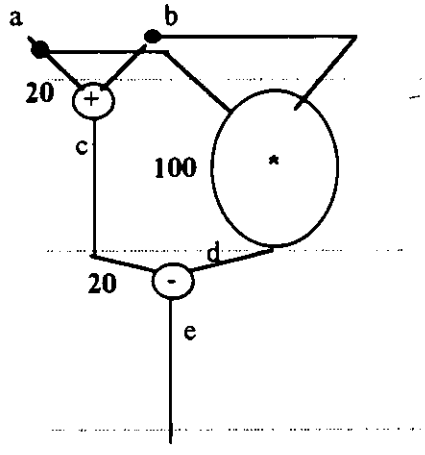
Sin embargo, si utilizamos un tiempo de ciclo de 50ns (figura 5.12) necesitaremos un paso de control para realizar la suma y la resta, y sólo 2 pasos de control para realizar la

multiplicación. En este caso el número total de etapas de control es de 3 y el tiempo total de ejecución es de 150ns, frente a los 120 necesarios para el caso anterior.

Dependiendo de las necesidades del usuario esta selección será peor o mejor que la de 20ns. Si el usuario quiere optimizar el tiempo, esta selección es peor, pero si lo más importante es el área, ésta puede ser una selección mejor.

Si seleccionamos un tiempo de ciclo de 100ns, el número total de pasos de control necesarios será de 2, pero en este caso el tiempo total de ejecución es de 200ns (figura 5.13). Como vemos esta selección reduce un 30% el número de etapas de control, y empeora un 30% el tiempo total de ejecución (de 150ns a 200ns). Sólo si fuera totalmente prioritaria el área frente al tiempo sería una buena selección.

Cualquier tiempo de ciclo múltiplo de 100ns no mejora el número de etapas de control, pero empeora el tiempo total de ejecución.

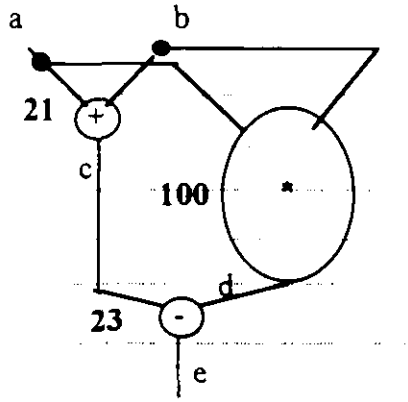


ciclo 100ns netapas 2
ttotal 200ns
UF: SUM(+), MUL(*), SUB (-)

Figura. 5.13

Viendo estos ejemplos podría pensarse que cuanto menor es el tiempo de ciclo mayor es el número de pasos de control y menor el tiempo de ejecución. Sin embargo, no ocurre

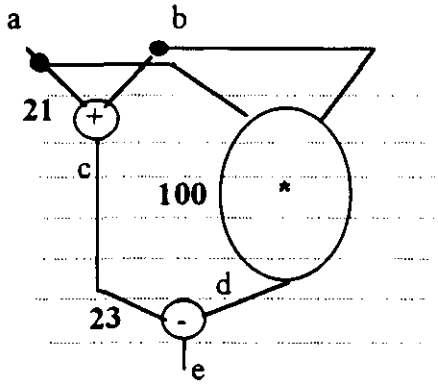
así siempre. Veamos el siguiente caso.



ciclo 25ns
ttotal 125ns netapas 5
UF: ALU(+,-), MUL(*)

Figura. 5.14

Se considera una biblioteca de módulos que dispone de una ALU que puede realizar la suma en 21ns y la resta en 23ns. Si el objetivo es optimizar el hardware, puede utilizarse esta ALU para realizar la suma y la resta. Si el tiempo de ciclo es de 25ns (figura 5.14) se necesitan 4 etapas de control para la multiplicación, y 1 para la suma y la resta. En total tendremos 5 etapas de control y un tiempo total de ejecución de 125ns.



ciclo 23ns
ttotal 138ns netapas 6
UF: ALU(+,-), MUL(*)

Figura 5.15

Sin embargo, si utilizamos un tiempo de ciclo de 23ns (Figura 5.15) se necesitarán, como antes, 1 paso de control para realizar la suma y la resta, pero 5 pasos de control para realizar la multiplicación. En este caso el tiempo total de ejecución sería de 138ns (frente a los 125ns de antes) y el número total de etapas de control de 6 (frente a los 5 de antes). Podemos deducir que la selección de un tiempo de ciclo de 25ns frente a la selección de 23ns mejora tanto el número total de etapas de control, como el tiempo total de ejecución.

Una consecuencia inmediata de las consideraciones basadas en el ejemplo anterior es que, dependiendo de las características del diseño, de la biblioteca de módulos y de las restricciones del usuario, el tiempo de ciclo óptimo es diferente. Otra observación importante es que, en general, la disminución del tiempo de ciclo conduce a Unidades de Control con mayor número de microinstrucciones. Sin embargo esto no es siempre cierto, y si no se realiza la selección adecuadamente, se pueden obtener resultados peores tanto en área como en tiempo.

5.5. Algoritmo de Selección del tiempo de ciclo

La finalidad de la selección del tiempo de ciclo es tener en cuenta los dos factores anteriormente explicados:

- **Minimización de los tiempos muertos**, para obtener un tiempo de ejecución total mínimo. Esta minimización es necesario realizarla en todos los elementos de la LPCC.
- **Minimización del número de etapas de control**, para minimizar el tamaño de la Unidad de Control.

Nuestro objetivo es obtener una lista reducida de valores del tiempo de ciclo que optimicen estos dos factores y, una vez realizada la planificación ASAP, elegir el que

produzca un tiempo mínimo de ejecución [MFHM94]. Como la planificación ASAP tiene complejidad lineal, si la lista de posibles tiempos de ciclo tiene un tamaño asequible, la complejidad de este método no será excesiva.

Para conseguir este objetivo suponemos que disponemos de una biblioteca de módulos para realizar todas y cada una de las operaciones del grafo, y que puede existir en esta biblioteca más de una UF para realizar una determinada operación. Para cada operación que puede implementar una UF, existe la información del tiempo de ejecución, y para los registros y multiplexores sus retardos respectivos.

Existe también una serie de restricciones impuestas por el usuario, como el tiempo de ejecución máximo del circuito $\mathbf{t_{max}}$, el área máxima permitida $\mathbf{área_{max}}$, y la prioridad del área frente al tiempo. Esta prioridad se mide mediante dos parámetros dependientes entre sí.

α es una medida del peso del tiempo muerto y

β mide el peso del tamaño de la Unidad de Control.

La relación entre ellos viene dada por

$$\alpha + \beta = 1$$

El usuario puede dar el valor de uno de los dos parámetros.

Si $\alpha=\beta=0.5$ se dará la misma prioridad al área que al tiempo. Si $\alpha=0$ y $\beta=1$ el área tendrá prioridad total frente al tiempo, siempre teniendo en cuenta las restricciones del tiempo total de ejecución. Si $\alpha=1$ y $\beta=0$ se deberá dar total prioridad al tiempo, intentando su optimización total, aun a costa de incrementar el tamaño de la Unidad de Control, pero siempre respetando las restricciones de área.

El siguiente paso es examinar el GFD para obtener la LPCC mediante el algoritmo modificado de búsqueda de caminos críticos.

También es necesario conocer el rango de posibles valores del tiempo de ciclo, para lo cual definiremos un valor mínimo $\mathbf{t_{ciclo_{min}}}$ y un valor máximo $\mathbf{t_{ciclo_{max}}}$.

El valor de $\mathbf{t_{ciclo_{min}}}$ es el retardo de la UF más rápida $\mathbf{t_{UF_{rap}}}$, dividido por el factor de fraccionamiento $\mathbf{f_f}$.

$$tciclo_{min} = \frac{tUF_{rap}}{ff}$$

El factor de fraccionamiento $\mathbf{f_f}$ es función del número de etapas máximo recomendable para ejecutar una operación. Su valor depende únicamente de las restricciones del usuario. Si el usuario da más importancia al área frente al tiempo, es decir $\beta>\alpha$, el número de etapas recomendables para ejecutar una operación debe ser pequeño, próximo a 1, para que el número de etapas total también lo sea; en este caso el factor de fraccionamiento tiene valor 1. Sin embargo, si se da más importancia al tiempo que al área ($\beta<\alpha$), el número de etapas recomendables puede incrementarse, y el factor de fraccionamiento se hace mayor que 1. Para los ejemplos que hemos estado realizando, los siguientes valores de *ff* dan buenos resultados:

$$\beta > \alpha \Rightarrow ff = 1$$
$$\beta = \alpha \Rightarrow ff = 2$$
$$\beta < \alpha \Rightarrow ff = 3$$

El valor máximo del tiempo de ciclo $\mathbf{t_{ciclo_{max}}}$ es el retardo de la UF más lenta de la biblioteca. Este valor se elige así porque un valor del tiempo de ciclo superior al de la UF más lenta, dará lugar a tiempos muertos en todos las UFs de la biblioteca, sin disminuir el número de pasos de control, y por lo tanto no es aconsejable.

Para cada uno de los valores comprendidos entre $\mathbf{t_{ciclo_{min}}}$ y $\mathbf{t_{ciclo_{max}}}$, con un incremento de 1, se calcula un coste que tiene en cuenta la influencia de los tiempos muertos y del número de etapas en los resultados finales. Para encontrar este coste primero es necesario realizar una serie de definiciones.

Se define $\mathbf{t_{ij}}$ como el retardo de la Unidad Funcional i (**UFi**) al realizar la operación j

(incluyendo el retardo de los multiplexores y de almacenamiento del resultado en un registro como ya hemos dicho).

Se define $\mathbf{n_{etapas_{ij}}}$ como el número de etapas de control necesarias para realizar la operación j con la UF de tipo i, para un determinado valor del tiempo de ciclo.

Se define el tiempo muerto asociado a **UFi** al realizar una operación j ($\mathbf{t_{muertoij}}$), como el tiempo que transcurre desde que se acaba de realizar la operación j con la **UFi** hasta que se termina el ciclo. Para un tiempo de ciclo **tciclo** es

$$\mathbf{t_{muertoij}} = t_{ciclo} - t_{ij} \bmod(t_{ciclo})$$

donde la función mod es el resto de la división entera.

Se define **FCij** para un determinado tiempo de ciclo, como el coste que supone utilizar la Unidad Funcional de tipo i de la biblioteca para realizar la operación j. La función de coste **FCij** consta de dos factores: por una parte el coste en área y por otra parte el coste en tiempo. Estos costes se definen de la siguiente forma:

- El coste en **tiempo** es la relación entre el tiempo muerto de la UFi al realizar la operación j, y el tiempo total de ejecución de dicha operación, para un tiempo de ciclo determinado.

$$\frac{tmuerto_{ij}}{t_{ij}}$$

Si el tiempo muerto es cero, el coste en tiempo es cero. Si el tiempo muerto no es cero, pero es pequeño frente al tiempo de ejecución de dicha operación, el coste en tiempo también es pequeño. Pero si el tiempo muerto es del mismo orden que el tiempo de ejecución, el coste en tiempo se acerca a la unidad, y puede ser superior a ella.

- El coste en **área** es la relación entre el incremento en el número de etapas de

control que va a necesitar la UF_j para realizar la operación j, para un tiempo de ciclo determinado, dividido entre el factor de fraccionamiento $\mathbf{f_f}$. Como el mínimo número de etapas para una operación es 1, el coste en área viene dado por:

$$\frac{\text{netapas}_{ij} - 1}{\text{ff}}$$

Si el número de etapas de la UF_j, para realizar la operación j es 1, el coste en área es 0. Cuanto más se aleja este valor de 1, mayor es el coste en área, puesto que se necesitan más microinstrucciones para realizar la operación.

Estos dos costes los modulamos con los valores de α y β, que miden la importancia de uno de los dos factores frente al otro.

$$FC_{ij} = \alpha \frac{\text{tmuerto}_{ij}}{t_{ij}} + \beta \frac{\text{netapas}_{ij} - 1}{\text{ff}}$$

Dada una operación j definimos ($\mathbf{Ocur_j}$) como el número de veces que aparece dicha operación en el GFD.

Si inicialmene suponemos que todas las UFs capaces de realizar una operación j tienen la misma probabilidad de ser utilizadas, y que en total son $\mathbf{NUF_j}$, la probabilidad de utilizar la UFi para realizar una operación j es:

$$\text{Probij} = \frac{\text{Ocurj}}{\text{NUFj}}$$

Esta probabilidad no depende de i porque es la misma para todas las UFi que pueden realizar la operación j. El hecho de que unas UFs son más apropiadas que otras para implementar las operaciones del GFD, en función de los objetivos del usuario, se mide mediante el coste de utilizarlas, que sí depende de i.

Para normalizar se define la probabilidad total como la suma de todas las probabilidades:

$$Ptotal = \sum_{j=1}^{Noper} \sum_{i=1}^{NUFj} Probij$$

donde **Noper** es el número total de operaciones distintas del GFD. Si sustituimos en la fórmula anterior:

$$Ptotal = \sum_{j=1}^{Noper} \sum_{i=1}^{NUFj} \frac{\cancel{Ocurj}}{NUFj} = \sum_{j=1}^{Noper} \frac{Ocurj * NUFj}{NUFj} = \sum_{j=1}^{Noper} Ocurj$$

Es decir, la P_{total} es igual al número de total de operaciones o al número de nodos del GFD.

El coste asociado a un determinado tiempo de ciclo es :

$$FC = \frac{\sum_{j=1}^{Noper} \sum_{i=1}^{NUFj} FCij * Probij}{Ptotal}$$

Para todos los valores entre $\mathbf{t_{ciclomin}}$ y $\mathbf{t_{ciclomax}}$ con un incremento igual a 1, se calcula FC. De esta forma se obtiene una lista con los costes asociados a cada uno de los posibles tiempos de ciclo. Los tiempos de ciclo que producen menor coste son los mejores para el diseño que se está tratando, con una biblioteca particular y con las restricciones dadas por el usuario. Sin embargo, el coste no es una medida exacta sobre la bondad de un tiempo de ciclo, sino que representa un factor aproximado. En la lista de costes suele haber tiempos de ciclo que producen costes muy similares, y es difícil saber cual es el mejor de todos, ya que esto no se conoce hasta que no se obtiene una planificación. Por esta razón, entre todos los posibles tiempos de ciclo se escogen los que menor coste producen, particularmente en nuestro sistema de SAN escogemos los 5 tiempos de ciclo de menor coste, y para todos ellos se realiza la planificación ASAP. El tiempo de ciclo elegido es aquel que produce un tiempo de ejecución mínimo. El algoritmo de selección de tiempo de ciclo será:

Desde t=tciclomin hasta tciclo_max

Obtener FC(t)

Ordenar de menor a mayor coste FC(t)

Planificación ASAP con los 5 tiempos de ciclo con menor FC

Seleccionar de esos 5 tiempos de ciclo el que produzca un tiempo mínimo de ejecución

5.6. Ejemplos

5.6.1. FACET

El primer ejemplo que vamos a utilizar para verificar este algoritmo es el utilizado en FACET [TSSi86]. El GFD se muestra en la Figura 5.16.

En el GFD existen 2 operaciones lógicas: una AND (&) y una OR (|). El retardo de estas operaciones es muy pequeño comparado con el resto, puesto que es de un único nivel de puertas. Como ya explicamos, deben eliminarse del GFD para calcular el tiempo de ciclo.

En la Figura 5.16 se puede observar que, una vez eliminados los 2 operadores lógicos, existen 3 caminos de datos:

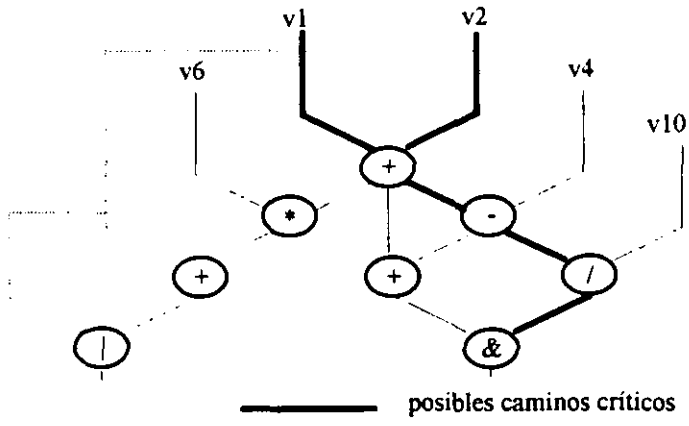


Figura. 5.16 GFD del ejemplo FACET

I) - formado por suma-multiplicación-suma.

II).- formado por suma-suma

III).- formado por suma-resta-división

La biblioteca de módulos disponible para realizar estas operaciones se muestra en la tabla 5.4 y la TPU en la tabla 5.5. La lista de caminos ordenada, utilizando la UF más lenta para implementar cada operación es:

LC={(división, resta, suma) (multiplicación, suma, suma) (suma, suma)}

Inicialmente LPCC={(división,resta,suma)}

Tomamos *A*= (multiplicación, suma, suma) y *B*=(división,resta,suma). Podemos unificar una suma de cada camino antes de utilizar la TPU. Para el resto de las operaciones las unificaciones son:

multiplicación $\Rightarrow$ división suma $\Rightarrow$ resta

Aplicando el teorema 3 el camino *A* no se inserta en la LPCC.

A continuación tomamos *A*= (suma, suma)

Podemos unificar una suma de cada camino, y la suma de *A* con la división de *B*. De nuevo aplicando el teorema 3, *A* no se inserta en la LPCC, que queda reducida a un camino.

LPCC={(división,resta,suma)}

Vamos a aplicar el algoritmo de selección del tiempo de ciclo a este único camino para diferentes valores de α y β, y nos quedamos en cada caso con los 5 tiempos de ciclo que producen menos coste.

Caso 1 $\alpha = 0.1$ *y* $\beta = 0.9$.

El área tiene mayor prioridad que el tiempo y el algoritmo debe obtener una lista de ciclos que minimicen el número de etapas de control. Para este valor de β, f_f es igual a 1 y el $t_{ciclo_{min}}$ es el retardo de la UF más rápida, es decir 20ns. El $t_{ciclo_{max}}$ es el retardo de la UF más lenta, es decir, 103ns. Entre estos dos valores, y con un incremento igual a 1, se calculan los costes FC, y se eligen los 5 tiempos de ciclo que dan menor coste. En la tabla 5.6 se muestran los resultados.

Se puede observar que, en este caso, el tiempo de ciclo con menor coste es 103ns, que permite realizar todas las operaciones en un solo paso de control, con lo cual el número total de etapas de control es el mínimo 4. Además, de todos los tiempos de ciclo que permiten realizar todas las operaciones en un solo paso de control, es el que produce los menores tiempos muertos, puesto que cualquier otro valor superior a 103ns no disminuye el número de etapas, e incrementa los tiempos muertos de todas las operaciones.

ciclo (en ns)	FC	tiempo muerto			nº de etapas por UF			nº etapas total	tiempo de ejecución (en ns)
		+	-	/	+	-	/		
103	0.24	83	82	0	1	1	1	4	412
52	0.39	32	31	1	1	1	2	5	260
53	0.39	33	32	3	1	1	2	5	265
54	0.40	34	33	5	1	1	2	5	270
55	0.40	35	34	7	1	1	2	5	275

Tabla 5.6 Costes para $\alpha = 0.1$ y $\beta = 0.9$.

Los siguientes valores de la lista permiten realizar la suma y la resta en un solo paso de control y la división en 2. Los tiempos de ciclo que permiten esto son los que están en el rango 52ns-102ns, y puede comprobarse que todos los demás valores de la tabla pertenecen a este rango y son los que producen menores tiempos muertos (lo más

próximos a 52ns). Cualquier valor inferior a 52ns necesitaría 3 pasos de control para la división, y como se trata de optimizar el área no aparece ninguno en la lista.

Caso 2 $\alpha=0.3$ *y* $\beta=0.7$.

El área sigue siendo más importante que el tiempo, pero no tanto como en el caso anterior. Los valores de $t_{ciclo_{min}}$ y de $t_{ciclo_{max}}$ son los mismos que en el anterior. Los resultados se muestran el la tabla 5.7.

ciclo (en ns)	FC	tiempo muerto			nº de etapas por UF			nº etapas total	tiempo de ejecución (en ns)
		+	-	/	+	-	/		
52	0.50	32	31	1	1	1	2	5	260
53	0.51	33	32	3	1	1	2	5	265
54	0.52	34	33	5	1	1	2	5	270
55	0.55	35	34	7	1	1	2	5	275
56	0.56	36	35	9	1	1	2	5	280

Tabla 5.7 Costes para α= 0.3 y β=0.7.

En este caso la opción tiempo de ciclo igual a 103ns no se presenta, porque se da más importancia al tiempo que en el caso anterior, y para esta opción los tiempos muertos eran bastante mayores que para las otras opciones. Aparece la opción 56ns que también, como todas las otras opciones, permite realizar la suma y la resta en un ciclo y la división en dos, y es la siguiente opción que produce menores tiempos muertos.

Caso 3 $\alpha = 0.7$ *y* $\beta= 0.3$

Para estos valores de α *y* β se está otorgando más importancia al tiempo que al área. Los tiempos de ciclo obtenidos conducen a tiempos muertos menores, aunque se incrementa el número de etapas de control. Para este valor de β, f_f es igual a 3 y el $t_{ciclo_{min}}$ es el retardo de la UF más rápida dividido entre 3, es decir 6ns. La tabla 5.8 muestra los

resultados.

En este caso los tiempos de ciclo son menores que en casos anteriores y los tiempos muertos se reducen sustancialmente. Por ejemplo, para la primera opción, la suma tiene un tiempo muerto de 6ns frente a los 32ns del caso anterior. Puede observarse que el número de etapas para la suma y la resta sigue siendo 1, pero para la división se ha incrementado a 4. El primer valor de la lista permite un tiempo de ejecución de 182ns frente a los 260ns del caso anterior, y el número de etapas se ha incrementado de 5 a 7. Se ha optado por soluciones que aumentan el área de la UC y disminuyen el tiempo de ejecución.

ciclo	FC	tiempo muerto			nº etapas por UF			nº etapas máximo	tiempo de ejecución
(en ns)		+	-	/	+	-	/		(en ns)
26	0.02	6	5	1	1	1	4	7	182
27	0.21	7	6	5	1	1	4	7	189
28	0.24	8	7	9	1	1	4	7	196
25	0.24	5	4	22	1	1	5	8	200
29	0.27	9	8	13	1	1	4	7	203

Tabla 5.8 Costes para α= 0.7 y β=0.3.

Caso 4 $\alpha = \beta = 0.5$

En este caso el área y el tiempo tienen la misma importancia. Para este valor de β, f_f es igual a 2 y el $t_{ciclomin}$ 10. En la tabla 5.9 se muestran los resultados para esta opción.

Todas las posibles opciones que se presentan como soluciones equilibran el coste del número de etapas y tiempo de ejecución. Se puede observar que el máximo número de etapas para la división es 4 frente a 5 del caso anterior. También aparece la opción tc=35ns, que permite una planificación en 6 etapas frente a las 7 del resto de las posibilidades, sin incrementar excesivamente el tiempo total de ejecución (210ns frente a

196ns, que se obtiene con la opción tc=28ns).

ciclo	FC	tiempo muerto			nº etapas por UF			nº etapas máximo	tiempo de ejecución
(en ns)		+	-	/	+	-	/		(en ns)
26	0.31	6	5	1	1	1	4	7	182
27	0.33	7	6	5	1	1	4	7	189
28	0.35	8	7	9	1	1	4	7	196
35	0.36	15	14	2	1	1	3	6	210
29	0.37	9	8	13	1	1	4	7	203

Tabla 5.9 Costes para α=0.5 y β = 0.5

5.6.2. Ecuación diferencial

El ejemplo de la ecuación diferencial [PaKG86] lo utilizamos en el apartado 5.3.2.4 para obtener los posibles caminos críticos de un GFD (figura 5.9). Si utilizamos la misma biblioteca de módulos que en aquel caso, el único posible camino es (*,*,-,-).

Vamos a aplicar el algoritmo de selección del tiempo de ciclo a este único camino para diferentes valores de α y β y, como en el ejemplo anterior, nos quedamos en cada caso con los 5 mejores valores.

Caso 1 α= *0 y* β=*1.*

Con estos valores queremos presentar un caso en el que el área tiene prioridad total sobre el tiempo, y sólo hay que minimizar el número de etapas de control. En la tabla 5.10 se muestran los resultados.

Se puede observar que, en este caso, el tiempo de ciclo que conduce a un menor coste (0) es 100, y permite realizar todas las operaciones en un solo paso de control, con lo cual el número total de etapas de control es el mínimo, 4. El resto de las soluciones,

desde un tiempo de ciclo de 50ns hasta 99ns, permiten realizar la resta en un paso de control y la multiplicación en dos. Para todas ellas el coste es el mismo, 0.5, puesto que el número de etapas de control total es siempre 6, y el tiempo de ejecución no importa (α =0)

ciclo (en ns)	FC	tiempo muerto		nº etapas por UF		nº etapas total	tiempo de ejecución (en ns)
		-	*	-	*		
100	0	79	0	1	1	4	400
50	0.5	29	0	1	2	6	300
51	0.5	30	2	1	2	6	306
52	0.5	31	4	1	2	6	312
53	0.5	32	6	1	2	6	318

Tabla 5.10 Costes para α= 0 y β=1.

Caso 2 α=0.3 y β= 0.7.

El área es más importante que el tiempo, pero no es totalmente prioritaria, como en el caso anterior. Los resultados se muestran el la tabla 5.11.

ciclo (en ns)	FC	tiempo muerto		nº etapas por UF		nº etapas total	tiempo de ejecución (en ns)
		-	*	-	*		
100	0.51	79	0	1	1	4	400
50	0.53	29	0	1	2	6	300
51	0.54	30	2	1	2	6	306
52	0.55	31	4	1	2	6	312
53	0.56	32	6	1	2	6	318

Tabla 5.11 Costes para α= 0.3 y β=0.7.

Los tiempos de ciclo que producen menos coste siguen siendo los mismos que para α=0,

pero la solución tciclo=100ns no tiene un coste 0, como ocurría en el caso anterior, puesto que, aunque da lugar a sólo 4 etapas de control, es la que produce mayor tiempo de ejecución.

Caso 3 $\alpha = \beta = 0.5$.

En este caso el tiempo y el área tienen la misma importancia. Los tiempos de ciclo obtenidos conducen a tiempos muertos menores, aunque se incrementa el número de etapas de control. La tabla 5.12 muestra los resultados.

ciclo (en ns)	FC	tiempo muerto		nº etapas por UF		nº etapas total	tiempo de ejecución (en ns)
		-	*	-	*		
34	0.38	13	2	1	3	8	272
35	0.40	14	5	1	3	8	280
25	0.40	4	0	1	4	10	250
26	0.42	5	4	1	4	10	260
50	0.42	29	0	1	2	6	300

Tabla 5.12 Costes para $\alpha = 0.5$ y $\beta = 0.5$.

Podemos observar que las cinco soluciones propuestas equilibran el número de etapas de control y el tiempo de ejecución. En todos los casos, este último se ha reducido sustancialmente respecto a la solución para tciclo=100ns, que se proponía como óptima para el caso $\alpha = 0$. Por ejemplo, la solución tciclo=34ns ha disminuido el tiempo de ejecución de 400ns a 272ns, mientras que el número de etapas se ha incrementado de 4 a 8. Las cinco soluciones propuestas permiten realizar la resta en una etapa de control cada una, independientemente del operador utilizado, el restador o el sumador/restador.

Caso 4 $\alpha = 0.7$ $\beta = 0.3$.

En este caso se otorga más importancia al tiempo que al área. En la tabla 5.13 se

presentan los resultados.

ciclo (en ns)	FC	tiempo muerto		nº etapas por UF		nº etapas total	tiempo de ejecución (en ns)
		-	*	-	*		
25	0.18	4	0	1	4	10	250
26	0.21	5	4	1	4	10	260
27	0.24	6	8	1	4	10	270
28	0.27	7	12	1	4	10	280
34	0.28	13	2	1	3	8	272

Tabla 5.13 Costes para α= 0.7 y β=0.3.

Los tiempos de ciclo son menores que en casos anteriores, y los tiempos muertos se reducen sustancialmente. La primera opción permite un tiempo de ejecución de 250ns frente a los 272ns del caso anterior, y se ha incrementado el número de etapas para la multiplicación a 4, con lo cual el número de etapas total pasa de 8 a 10.

Caso 5 α=1 y β=0.

El tiempo es totalmente prioritario frente al área. Las soluciones se presentan en la tabla 5.14.

ciclo (en ns)	FC	tiempo muerto		nº etapas por UF		nº etapas total	tiempo de ejecución (en ns)
		-	*	-	*		
25	0.05	4	0	1	4	10	250
7	0.06	3	5	4	15	38	266
26	0.09	5	4	1	4	10	260
13	0.09	5	4	2	8	20	260
6	0.10	5	2	5	17	44	264

Tabla 5.14 Costes para α= 1 y β=0.

En este caso, aparecen soluciones que necesitan realizar la multiplicación y la resta en varias etapas, porque los tiempos de ciclo son menores (6ns, y 7ns), con lo cual el número de etapas de control total se incrementa notablemente (38 y 44). Sin embargo, puede observarse que el tiempo de ciclo que minimiza el tiempo de ejecución sigue siendo 25ns, que es la primera solución propuesta.

5.6.3. Filtro Ar

A continuación vamos a presentar el ejemplo del Filtro-Ar [JaPP87], cuyo GFD se muestra en la figura 5.17.

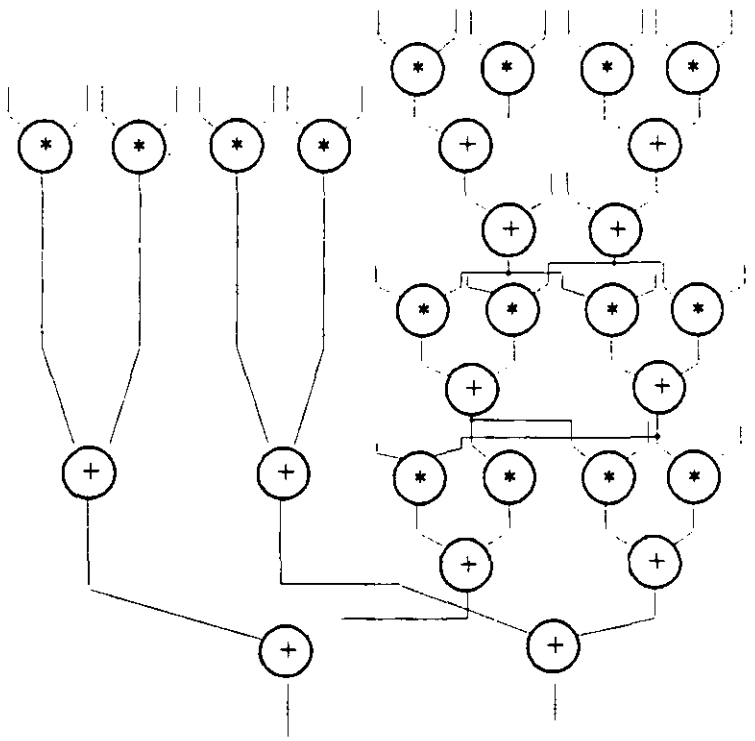


Figura. 5.17 GFD del ejemplo Filtro Ar

La lista de caminos para este GFD es:

LC={(*,+,+) (*,+,+) (*,+,+) (*,+,+) (*,+,+,*,+,*,+,+) (*,+,+,*,+,*,+,+) (*,+,+,*,+,*,+,+) (*,+,+,*,+,*,+,+)}

Eliminando los caminos que son iguales nos queda:

LC={(*,+,+) (*,+,+,*,+,*,+,+)}

Vamos a comparar los resultados que se obtienen aplicando el algoritmo de selección del tiempo de ciclo para distintas bibliotecas de módulos. Usaremos en primer lugar la misma biblioteca que en los ejemplos anteriores, **bibl1**. Por otra parte, utilizaremos la biblioteca que presentamos en la tabla 5.15, **bibl2**.

modulo	opl	retardo
sumador	+	30ns
sumador	+	24ns
multiplicador	*	138ns

Tabla 5.15 Biblioteca *bibl2*

Para esta biblioteca la TPU se presenta en la tabla 5.16.

*(138ns)	+(30ns)
	* (138ns)

Tabla 5.16 TPU para la biblioteca de la tabla 5.15

Para ambas bibliotecas, la LC ordenada es:

LC={(*,*,*,+,+,+,+,+,+) (*,+,+)}

Utilizando las TPUs, en ambos casos se obtiene:

LPCC={(*,*,*,+,+,+,+,+)}

Como en los demás ejemplos, vamos a aplicar el algoritmo de selección del tiempo de ciclo a los posibles caminos de críticos para diferentes valores de α y β, y nos quedamos en cada caso con los 5 mejores valores.

Caso 1 α= 0.1 y β=0.9.

En la tabla 5.17 se muestran los resultados para las dos bibliotecas utilizadas.

ciclo bibli1	nº etapas por UF		nº etapas total	tiempo ejecución	ciclo biblio2	nº etapas por UF		nº etapas total	tiempo ejecución
(en ns)	+	*		(en ns)	(en ns)	+	*		(en ns)
100	1	1	8	800	138	1	1	8	1104
50	1	2	11	550	69	1	2	11	759
51	1	2	11	561	70	1	2	11	770
52	1	2	11	572	71	1	2	11	781
53	1	2	11	583	72	1	2	11	792

Tabla 5.17 Costes para α= 0.1 y β=0.9.

Para ambas bibliotecas se obtienen soluciones que minimizan el número de etapas de control. La mejor solución propuesta en ambos casos permite una planificación en 8 etapas de control. Este tiempo de ciclo viene determinado por el retardo del multiplicador, que para un caso es 100ns y para el otro 138ns. El resto de las soluciones producen un total de 11 etapas de control, y dependiendo de la biblioteca son unos valores diferentes. El tiempo de ejecución es siempre mayor para la bibl2 que para la bibl1, puesto que el retardo del multiplicador es mayor.

Caso 2 α=0.3 y β= 0.7.

El área sigue siendo más importante que el tiempo, pero no tanto como en el caso anterior. Los resultados se muestran el la tabla 5.18.

En este caso las opciones tiempo de ciclo igual a 100ns para la bibl1, y 138ns para la bibl2 no se presentan, porque se da más importancia al tiempo que en el caso anterior y con esta opción los tiempos de ejecución eran bastante mayores que las otras. Aparece la opción 54ns para la bibl1 y 73ns para la bibl2 que también, como todas las otras opciones, permite realizar la suma en un ciclo la multiplicación en dos y es la siguiente opción que produce menores tiempos muertos.

ciclo bibli1 (en ns)	nº etapas por UF		nº etapas total	tiempo ejecución (en ns)	ciclo biblio2 (en ns)	nº etapas por UF		nº etapas total	tiempo ejecución (en ns)
	+	*				+	*		
50	1	2	11	550	69	1	2	11	759
51	1	2	11	561	70	1	2	11	770
52	1	2	11	572	71	1	2	11	781
53	1	2	11	583	72	1	2	11	792
54	1	2	11	594	73	1	2	11	803

Tabla 5.18 Costes para α= 0.3 y β=0.7.

Caso 3 α= 0.5 y β=0.5.

Los resultados se presentan en la tabla 5.19.

ciclo bibli1 (en ns)	nº etapas por UF		nº etapas total	tiempo ejecución (en ns)	ciclo biblio2 (en ns)	nº etapas por UF		nº etapas total	tiempo ejecución (en ns)
	+	*				+	*		
25	1	4	17	425	35	1	4	17	595
26	1	4	17	442	36	1	4	17	612
34	1	3	14	476	46	1	3	14	644
27	1	4	17	459	37	1	4	17	629
35	1	3	14	490	47	1	3	14	658

Tabla 5.19 Costes para α= 0.5 y β=0.5.

Puede observarse que los tiempos de ejecución han disminuido notablemente, y que el número de etapas de control totales se ha incrementado. También puede observarse que los valores que conducen a buenas soluciones para una biblioteca son muy diferentes que los que dan buenos resultados para la otra. Por ejemplo, el tiempo de ciclo 25ns que produce un número total de etapas de 17 para la *bibl1*, si se utilizara para la *bibl2* se necesitarían 6 etapas para realizar la multiplicación, y dos para la suma, por tanto en total 28 etapas de control y un tiempo de ejecución de 700ns, frente a las 17 etapas y 595ns que produce la selección 35ns. Queda por tanto demostrado lo importante que es tener en cuenta la biblioteca de módulos en la selección del tiempo de ciclo.

Caso 4 $\alpha = 0.9$ *y* $\beta = 0.1$

El tiempo tiene más importancia que el área. Los tiempos de ciclo obtenidos conducen a tiempos de ejecución menores, aunque se incrementa el número de etapas de control. La tabla 5.20 muestra los resultados. Puede comprobarse de nuevo que los tiempos de ciclo que producen buenos resultados para una biblioteca no sirven si se utiliza la otra. En este caso los tiempos de ciclo son menores que en casos anteriores y los tiempos de ejecución que producen las distintas opciones también son menores, aunque el número de etapas de control se ha incrementado.

ciclo bibli1 (en ns)	nº etapas por UF +	nº etapas por UF *	nº etapas total	tiempo ejecución (en ns)	ciclo biblio2 (en ns)	nº etapas por UF +	nº etapas por UF *	nº etapas total	tiempo ejecución (en ns)
25	1	4	17	425	35	1	4	17	595
23	1	5	20	460	15	2	10	40	600
26	1	4	17	442	36	1	4	17	612
24	1	5	20	480	30	1	5	20	600
27	1	4	17	459	31	1	5	20	620

Tabla 5.20 Costes para $\alpha = 0.9$ y $\beta = 0.1$.

5.7. Conclusiones

A lo largo de este capítulo se han presentado diversos ejemplos que demuestran cómo varían los tiempos de ciclo óptimos en función de las restricciones del usuario, del diseño en particular y de la biblioteca de módulos que se utilice. También se ha comprobado que una selección inteligente del tiempo de ciclo puede conducir a planificaciones que, sin incrementar el número de etapas, disminuyan el tiempo de ejecución, o bien sin aumentar el tiempo de ejecución se disminuya el número de etapas. Por lo tanto, parece necesario emplear un cierto tiempo en realizar una buena selección antes de hacer la planificación y asignación de hardware, dada la complejidad de estos algoritmos y su gran dependencia de dicho valor.

Se ha presentado un algoritmo de selección que tiene en cuenta el GFD, la biblioteca de módulos y las restricciones del usuario. En primer lugar, la biblioteca de módulos se utiliza para realizar una búsqueda de los posibles caminos críticos del circuito, que varían dependiendo del tiempo de ciclo. A partir de esta lista de caminos críticos, se aplica un algoritmo de selección que tiene en cuenta los objetivos del usuario. Como la complejidad de este algoritmo de selección es baja, puede utilizarse sin incrementar excesivamente el tiempo de ejecución del algoritmo de planificación.

Sin embargo, este algoritmo necesita conocer los retardos de las operaciones del GFD, que dependen de los tiempos de ejecución de los operadores que los implementen y también de los retardos asociados a las transmisiones de señales. Como este último valor no es conocido, en el capítulo 6 presentaremos un método que nos permite realizar una estimación.

Capítulo 6

Estimación del Retardo de las Interconexiones

6.1. Introducción

En el capítulo anterior surgió la necesidad de conocer el tiempo de ejecución de las operaciones del GFD. Vimos que, para el modelo de ruta de datos que se utiliza en FIDIAS, una estimación de este retardo ***tejec*** viene dada por:

$$t_{ejec}=t_{prop} + t_{UF} + t_{almac} + 2*t_{mux}$$

donde t_{UF} es el retardo debido a la UF que implementa la operación, t_{almac} es el retardo debido a la carga del registro que almacena el resultado, t_{mux} es el retardo debido a un multiplexor y t_{prop} es el retardo debido a las transmisiones de las señales. Los retardos de los módulos son conocidos y pueden tenerse almacenados en la biblioteca de módulos. Sin embargo, t_{prop} es un valor desconocido, y vimos que depende de los retardos de las diversas transmisiones de señales que tienen lugar en la ejecución de una operación.

Por lo tanto, para tener toda la información necesaria para calcular el tiempo de ejecución de una operación, es imprescindible realizar una estimación del retardo que supone la transmisión de una señal a través de una interconexión, y para esto es necesario encontrar un modelo físico que represente dicha transmisión.

Supongamos una interconexión (figura 6.1) que conecta dos módulos.

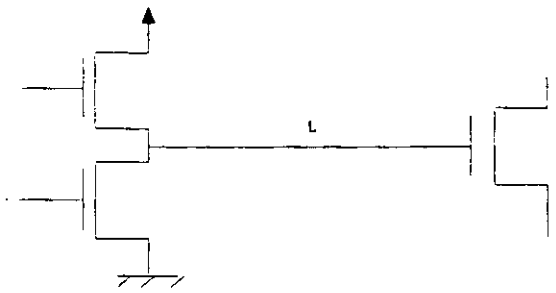


Figura 6.1 Línea de transmisión

Desde el nivel de abstracción físico, la interconexión parte de un transistor con una resistencia de salida R_t, tiene una longitud L y termina en un transistor de capacidad de entrada C_t (figura 6.2). La capacidad y la resistencia del cable por unidad de longitud son c y r y por tanto, la resistencia total del cable es $R=r*L$ y la capacidad total $C=c*L$

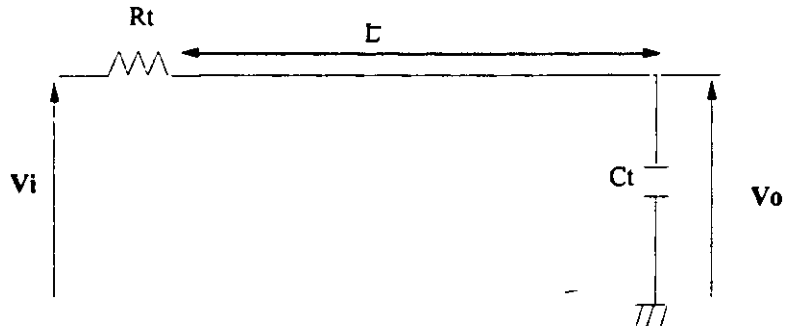


Figura 6.2 Modelo físico de una línea de transmisión

En las tecnologías SSI, MSI y LSI la capacidad de la línea era perfectamente despreciable frente a las capacidades de los transistores ($C << C_t$) y el modelo para calcular el retardo se obtenía sustituyendo la línea de transmisión por una resistencia R (figura 6.3).

Este modelo se corresponde con un típico circuito RC cuyo retardo es proporcional a $(R+R_t)*C_t$, [Kuo87] y por tanto el cálculo del retardo de la línea tiene una complejidad constante.

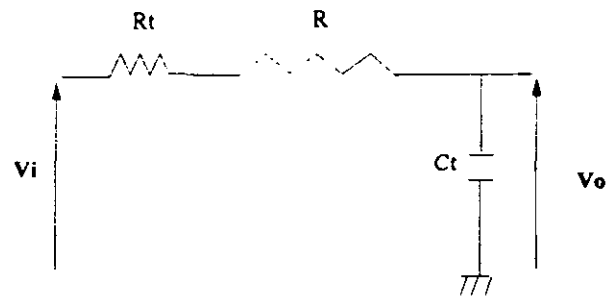


Figura 6.3. Modelo de una línea para tecnología LSI

Sin embargo, el área de las interconexiones en la tecnología VLSI es normalmente superior al área de los transistores [WeEs94], por lo cual sus efectos de carga son muy importantes y es necesario considerarlos. Al introducir C en el modelo de la línea, hay que considerar que la resistencia R y la capacidad C se distribuyen de forma homogénea a lo largo de la longitud L, y no se pueden considerar concentrados en ninguna posición concreta. Así obtenemos el **modelo ideal de una línea de transmisión** que se muestra en la figura 6.4. Este modelo es un circuito en escalera de infinitos tramos, donde los elementos en las ramas en serie son resistencias y en las paralelas son capacidades. La suma total de todas las capacidades es C y la de todas las resistencias es R.

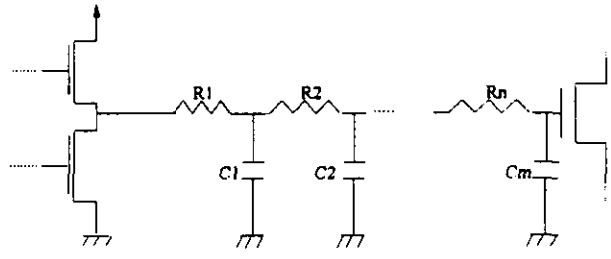


Figura 6.4. Modelo ideal de una línea de transmisión

Como la resolución de este modelo es demasiado compleja, existen diversos modelos alternativos en la literatura mucho más simples, cada uno de los cuales se comporta de forma aceptable para ciertos valores de C_t y R_t frente a C y R. Cualquier modelo no ideal tendrá ***n*** ramas en serie, que serán resistencias, y ***m*** ramas en paralelo, que serán capacidades. Cuanto mayores sean n y m más se aproxima este modelo al modelo ideal de una línea de transmisión.

A lo largo de este capítulo veremos algunos de los modelos comúnmente utilizados, y cómo pueden emplearse para obtener las estimaciones del retardo con errores aceptables.

6.2. Modelos para una línea de transmisión

Para cuantificar la aceptabilidad de un modelo se ha seguido a [Saku83], que considera que un modelo es **aceptable** cuando su utilización produce errores inferiores al 3% respecto al comportamiento real de la línea de transmisión.

El modelo más simple de todos es aquel que considera que la línea tiene una resistencia R=0 y una capacidad C=0. Este modelo se denomina **N** y es aceptable para valores de R_l *C_l del orden de 2500 veces mayores que R*C (Figura 6.5).

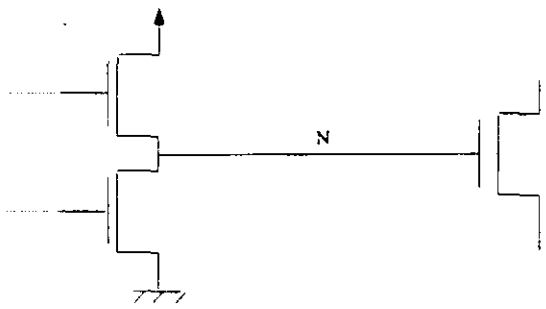


Figura 6.5. Modelo N

Cuando el orden de magnitud de R_l y C_l disminuye frente al de R y C, el modelo aceptable se va complicando.

En el **Modelo C** (figura 6.6) la interconexión viene representada por un condensador de capacidad C. Es aceptable cuando R_l es del orden de 50 veces mayor que R.

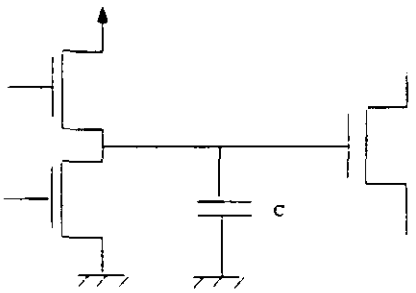


Figura 6.6. Modelo C

En el **Modelo R** (figura 6.7) el cableado se representa por una resistencia R y resulta aceptable cuando C_l es del orden de 50 veces mayor que C.

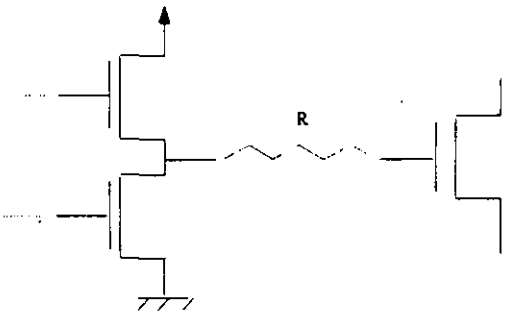


Figura 6.7. Modelo R

En el **Modelo CPL** (figura 6.8) el cableado se representa por una escalera formada por una rama paralela C y por una rama serie R. Este modelo no suele dar buenos resultados frente a los que presentamos a continuación, y es raramente utilizado.

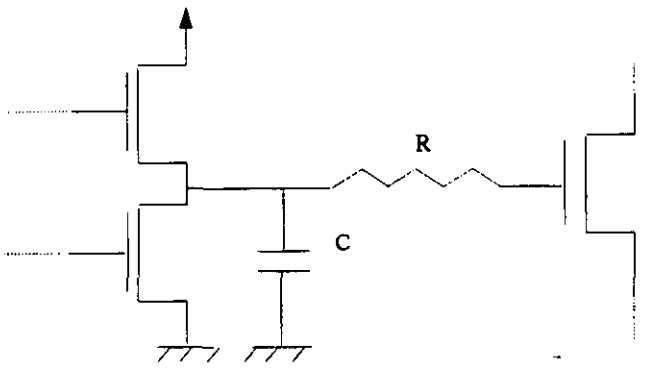


Figura 6.8. Modelo CPL

En el **Modelo L** de un paso la representación viene dada por una escalera formada por una rama serie R y una rama paralela C (figura 6.9). Suele utilizarse cuando R_t*C_t es del orden de 200 veces superior a R*C.

Figura 6.9. Modelo L

El **Modelo T** es un modelo escalera que comienza y termina con una rama serie. En el modelo $\mathbf{T_i}$, el subíndice i indica el número de módulos resistencia-condensador-resistencia, y coincide con el número de condensadores. Cada módulo tiene un condensador de valor C/i, y dos resistencias $R/(2*i)$. La última rama resistencia de un módulo y la primera rama resistencia del siguiente están en serie, y por tanto se suman dando lugar a una resistencia de valor R/i. En el modelo $\mathbf{T_1}$ habrá un único módulo, formado por dos resistencias de valor $R/2$ y un condensador de valor C, como se muestra en la Figura 6.10.

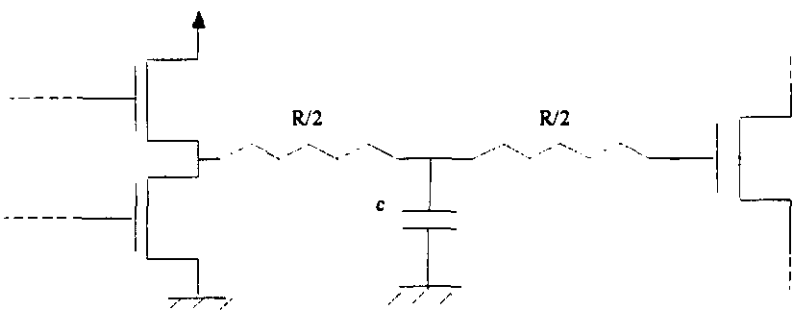


Figura 6.10. Modelo T1

Los modelos T_2, T_3, T_4, ..., se forman colocando en cascada varios módulos. Por ejemplo, el modelo T2 estará formado por dos módulos, cada uno de los cuáles tiene dos resistencias de

valor *R*/4 y un condensador de valor *C*/2, como se muestra en la figura 6.11.

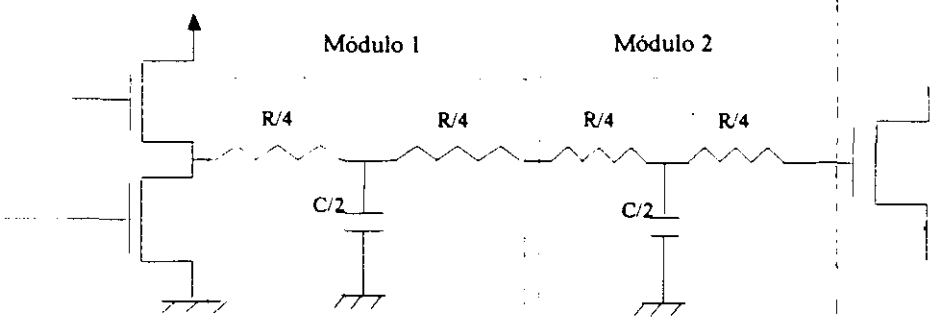


Figura 6.11. Modelo T2

Si las dos resistencias centrales se suman, obtenemos el circuito de la figura 6.12

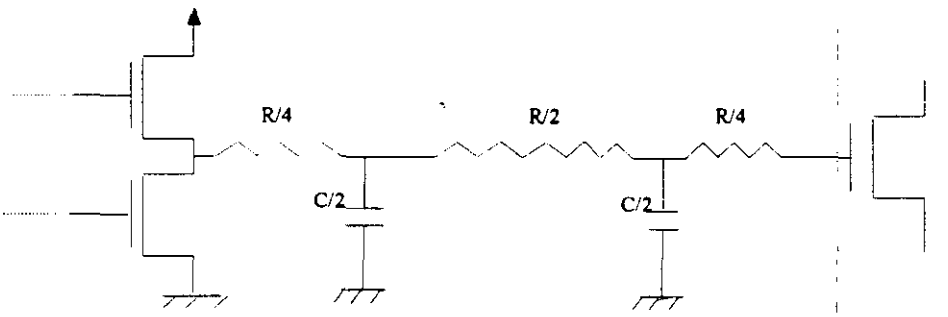


Figura 6.12. Modelo T2 simplificado

El rango de aplicabilidad del modelo T depende del número de módulos que contenga, y posteriormente veremos cuándo se utiliza cada uno de ellos.

El **Modelo Π** es un modelo escalera que comienza y termina con una rama paralela. En el modelo $\mathbf{\Pi_i}$, el subíndice *i* indica el número de módulos condensador-resistencia-condensador, y coincide con el número de resistencias. Cada módulo tiene dos condensadores de valor *C/(2*i)*, y una resistencia *R/i*. Los condensadores de un módulo y del siguiente, cada uno de un valor *C/(2*i)*, quedan en paralelo, y por tanto se suman dando lugar a un condensador de valor *C/i*. El modelo $\mathbf{\Pi_1}$ tendrá un único módulo con dos condensadores de valor *C*/2 y una resistencia de *R*, como se puede ver en la Figura 6.13.

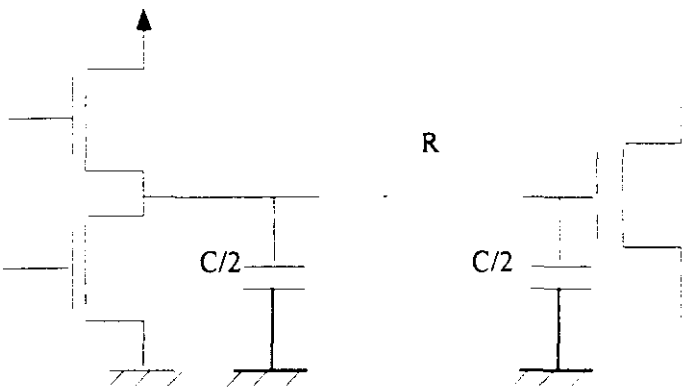


Figura 6.13 Modelo Π_1

El modelo Π_2 (figura 6.14) tiene dos módulos en cascada, cada uno con dos condensadores de valor *C*/4 y una resistencia de valor *R*/2. Los condensadores centrales pueden sumarse dando lugar a un condensador de valor *C*/2, y por tanto tendrá 3 condensadores de valor *C*/4, *C*/2 y *C*/4, y dos resistencias de valor *R*/2.

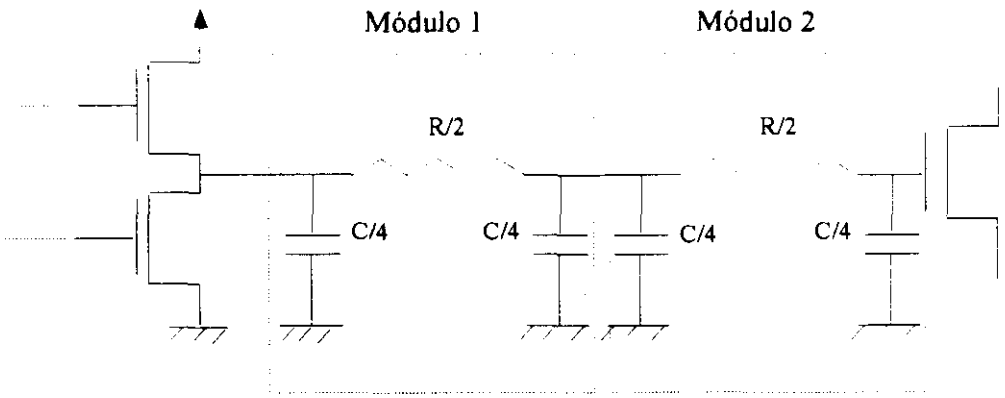


Figura 6.14 Modelo Π_2

El modelo Π_3, que se muestra en la Figura 6.15, tiene los condensadores de los extremos con un valor *C*/6, los centrales de *C*/3, y las resistencias valen *R*/3.

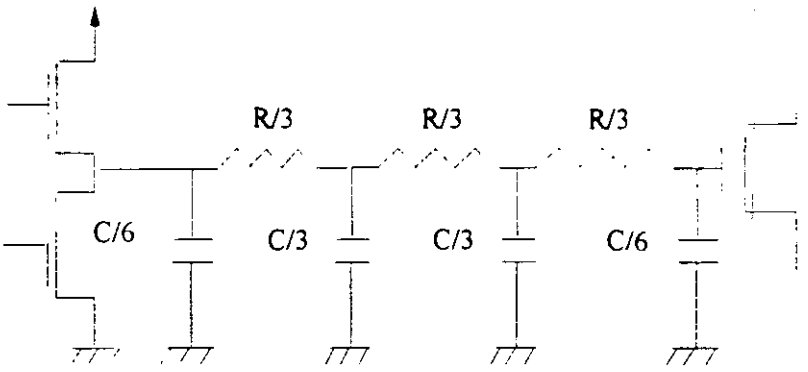


Figura 6.15. Modelo Π3

$C_t/C \backslash R_t/R$	0	0.01	0.1	0.2	0.5	1.0	2.0	5.0	10.0	20.0	50.0	100
0	Π3	Π3	Π2	Π2	Π1	Π1	Π1	Π1	Π1	C	C	C
0.01	Π3	Π3	Π2	Π2	Π1	Π1	Π1	Π1	Π1	C	C	C
0.1	T2	T2	Π2	Π2	Π1	Π1	Π1	Π1	Π1	C	C	C
0.2	T2	T2	Π2	Π2	Π1	Π1	Π1	Π1	Π1	C	C	C
0.5	T1	T1	T1	T1	Π1	Π1	Π1	Π1	Π1	C	C	C
1.0	T1	T1	T1	T1	Π1	Π1	Π1	Π1	Π1	C	C	C
2.0	T1	T1	T1	T1	Π1	Π1	Π1	Π1	L1	L1	C	C
5.0	Π1	Π1	Π1	Π1	Π1	Π1	Π1	L1	L1	L1	C	C
10.0	Π1	Π1	Π1	Π1	Π1	Π1	L1	L1	L1	L1	C	C
20.0	R	R	R	R	R	R	L1	L1	L1	L1	C	C
50.0	R	R	R	R	R	R	R	R	R	R	C	N
100	R	R	R	R	R	R	R	R	R	R	N	N

Tabla 6.1 Modelos aceptables

En la tabla 6.1 se muestran los resultados de un estudio realizado por Sakurai [Saku83], sobre qué modelos, de los anteriormente descritos, son aceptables para cada rango de valores de R_t, C_t, R y C, es decir, aquellos cuya utilización da resultados con errores inferiores al 3% en el cálculo del retardo de la línea. Si un modelo es aceptable para un determinado rango de

valores, todos los que son más complejos que él también lo son.

Como vemos en la tabla, el modelo Π3 es válido para cualquier valor de R_t/R y de C_t/C. Experimentalmente se comprueba que la diferencia de utilizar el modelo Π3 y el Π4 es del 0.1%. Por otra parte el modelo Π3 es más simple que el T3, puesto que tiene un condensador menos, y por tanto los cálculos son más sencillos. Como no se gana prácticamente nada en exactitud, y el modelo Π3 es más simple que el Π_4 y el T_3, en nuestro sistema FIDIAS **se utilizará el modelo Π3 como modelo de una línea de transmisión**. En el próximo apartado calcularemos el retardo asociado.

6.3. Retardo del modelo Π_3

El modelo Π3, incluyendo la resistencia de salida del transistor fuente, y la capacidad de entrada del transistor destino se muestra en la Figura 6.16.

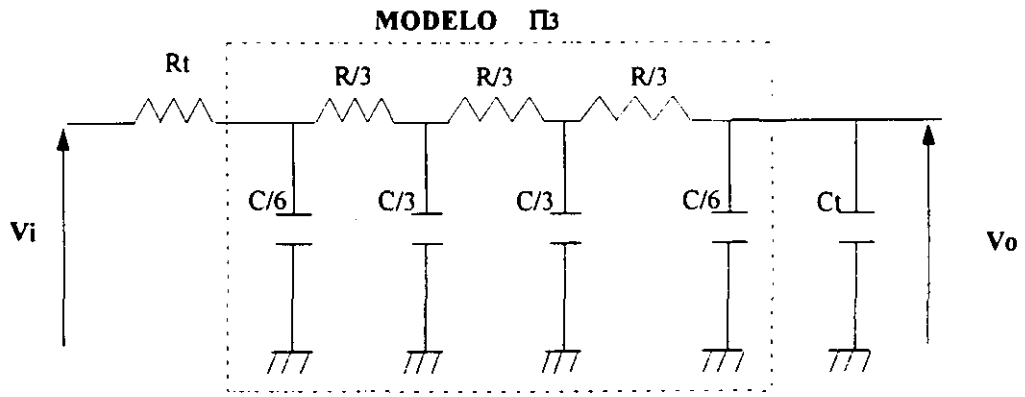


Figura 6.16. Modelo Π3

Si se realiza un estudio de la respuesta del circuito, es ampliamente aceptado que cuando la salida toma un valor igual al 90% del valor de la entrada, la señal ha terminado de transmitirse. A este tiempo se le denomina $\mathbf{t_{0,9}}$, y es el que se utilizará como retardo de la señal en una línea de transmisión.

Para calcular $t_{0,9}$ se puede estudiar la respuesta en frecuencia del circuito [Kuo87]. El primer paso necesario es calcular la función de transferencia del circuito, que para este modelo es:

$$Ft = \frac{Vo}{Vi} = \frac{F}{A s^4 + B s^3 + D s^2 + Es + F}$$

donde:

$$A = C^3 R^3 Rt[C + 6Ct]$$
$$B = 6C^2 R^2[C[R + 12Rt] + 6Ct[R + 9Rt]]$$
$$D = 81C\ R\ [C[4R + 19Rt] + 2Ct[8R + 27Rt]]$$
$$E = 4374[C[R + 2Rt] + 2Ct[R + Rt]]$$
$$F = 8748$$

Como el denominador de la función de transferencia es un polinomio de 4° grado, existen 4 polos. Los valores de estos polos dependen de la relación de $C_t{*}R_t/C{*}R$, que a su vez depende de la longitud de la interconexión y de la tecnología utilizada. La respuesta temporal del circuito depende de la posición de los polos. Se puede demostrar [Kuo87], [Frie87] que cuando la relación entre los dos polos más significativos es de al menos un orden de magnitud, el polo dominante es el que determina la respuesta del sistema, y es posible simplificar el sistema de 4° orden a uno de 1er orden.

Por esta razón, es interesante conocer la variación de los dos polos más significativos en función de la relación $C_t{*}R_t/C{*}R$. Con este objetivo, vamos a realizar un estudio a partir de unos valores típicos de C_t y R_t de la biblioteca de ES2, que es la biblioteca de celdas estándar que proporciona CADENCE [ES2L93], y para un rango de posibles valores de la longitud de una interconexión L. De los datos que proporciona [ES2L93] se han obtenido como valores típicos, para tecnología de 1μ, e interconexiones sobre metal, R_t= 1000Ω, C_t=0.05pF, r=0.04 Ω/μ y c=0.09fF/μ. Con estos valores:

$$\frac{Ct{*}Rt}{C{*}R} \approx \frac{14{*}10^6}{L^2}$$

donde L viene dada en micras.

Se realiza una variación de L desde 1cm, hasta 1μ, que es el rango de valores que suelen tomar las longitudes de las interconexiones para los circuitos que se realizan sobre un único CI. En la tabla 6.2 se presentan los resultados.

Se puede observar que en todos los casos la frecuencia del polo dominante es al menos 25 veces menor que la del siguiente polo y, por lo tanto, es posible utilizar el método del polo dominante para calcular $t_{0,9}$.

El cálculo de los polos de un sistema de 4° orden es un problema de complejidad constante, con lo cual pasar a un sistema de 1er orden a partir del polo dominante del sistema de 4° orden también lo es.

L(μ)	R_f*C_f/R*C	polo2 (rad/ps)	polo1 (rad/ps)	polo2/polo1
1	1,39E+07	-6,71E+05	-2,00E-02	3,36E+07
2	3,47E+06	-1,68E+05	-1,99E-02	8,43E+06
3	1,33E+06	-6,42E+04	-1,99 E-02	3,23E+06
5	5,56E+05	-2,70E+04	-1,98E-02	1,36E+06
10	1,39E+05	-6,79E+03	-1,96E-02	3,46E+05
20	3,47E+04	-1,72E+03	-1,93E-02	8,93E+04
30	1,54E+04	-7,76E+02	-1,90E-02	4,09E+04
50	5,56E+03	-2,87E+02	-1,83E-02	1,57E+04
100	1,39E+03	-7,61E+01	-1,69E-02	4,51E+03
500	6,86E+01	-5,09E+00	-1,09E-02	4,65E+02
1000	1,39E+01	-1,33E+00	-7,00E-03	1,90E+02
2000	3,47E+00	-4,20E-01	-4,21E-03	9,99E+01
5000	5,56E-01	-8,57E-02	-1,86E-03	4,60E+01
7000	2,44E-01	-4,09E-02	-1,24E-03	3,31E+01
10000	1,39E-01	-2,45E-02	-9,22E-04	2,66E+01

Tabla 6.2 Relación de polos más significativos

Para un sistema de primer orden la respuesta en el tiempo es del tipo:

$$Vo = Vi(1 - e^{-\alpha t})$$

donde α es la frecuencia del polo dominante.

Para calcular $t_{0.9}$ sustituimos Vo por 0.9 Vi y tenemos:

$$0.9 * Vi = Vi(1 - e^{-\alpha * t_{0.9}})$$

despejando $t_{0.9}$ obtenemos:

$$t_{0.9} = \frac{Ln(10)}{\alpha}$$

Como se ha dicho anteriormente el cálculo de α es un problema de complejidad constante y, por tanto, el cálculo del retardo de una línea de transmisión es también de complejidad constante, una vez conocidos R, C, R_t y C_t. Esta solución es válida para el rango de valores de L presentados en la tabla 6.2. Sin embargo, para interconexiones de mayor longitud, la relación entre las frecuencias del polo dominante y del siguiente puede hacerse menor de un orden de magnitud, con lo cual este método no sería viable. En el próximo apartado presentamos un método alternativo, y un estudio sobre el rango de valores que permiten la utilización de cada uno de estos métodos.

6.4. Modelo para interconexiones de cualquier longitud

En [Saku83] se propone una solución analítica para calcular $t_{0.9}$, mediante la fórmula de aproximación dada por

$$t_{0.9} = 1.02RC + 2.21(R_tC_t + C_tR + R_tC)$$

Esta solución, que a partir de ahora llamaremos **método analítico**, tiene un error inferior al 1.1 % para valores de C_t y R_t inferiores a C y R, y del orden del 4% para el resto de los casos. El comportamiento de este método es contrario al que presentamos nosotros, puesto que obtiene mejores resultados cuanto mayor es la longitud de la interconexión (mayores son C y

R). Por eso lo proponemos como alternativa para interconexiones muy largas, que es donde nuestro método tiene un error mayor del 4%. Es interesante comparar el error de los dos métodos, con el fin de utilizar para cada valor de *L* el que produzca menos error.

En la tabla 6.3 se realiza un estudio comparativo entre ambos métodos. Para calcular el valor real del retardo se ha utilizado una herramienta de tratamiento de sistemas lineales (MATLAB) [MATL94].

			Método Polo Dominante		**Método Analítico**	
L(μ)	$R_t*C_t/R*C$	t0,9 real (ps)	t0,9 (ps)	error	t0,9 (ps)	error
1	1,39E+07	1,153E+02	1,153E+02	0,05%	1,107E+02	3,98%
2	3,47E+06	1,155E+02	1,155E+02	0,05%	1,109E+02	3,98%
3	1,33E+06	1,158E+02	1,158E+02	0,05%	1,11E+02	3,98%
5	5,56E+05	1,161E+02	1,162E+02	0.05%	1,115E+02	3,98%
10	1,39E+05	1,172E+02	1,172E+02	0.05%	1,125E+02	3,98%
20	3,47E+04	1,193E+02	1,194E+02	0.05%	1,146E+02	3,98%
30	1,54E+04	1,214E+02	1,215E+02	0.05%	1,166E+02	3,98%
50	5,56E+03	1,257E+02	1,257E+02	0.04%	1,207E+02	3,98%
100	1,39E+03	1,362E+02	1,363E+02	0.03%	1,309E+02	3,97%
500	6,86E+01	2,11E+02	2,110E+02	0.08%	2,027E+02	3,86%
1000	1,39E+01	3,30E+02	3,287+02	0.27%	3,175E+02	3,68%
2000	3,47E+00	5,506E+02	5,474E+02	0.59%	5,318E+02	3,41%
5000	5,56E-01	1,254E+03	1,236E+03	1,42%	1,219E+03	2,80%
7000	2,44E-01	1,898E+03	1,860E+003	2,01%	1,85E+03	2,37%
10000	1,39E-01	2,563E+03	2,498E+03	2,54%	2,511E+03	2,03%

Tabla 6.3 Comparación de resultados obtenidos con el Método del Polo Dominante y el Método Analítico

En la figura 6.17 se presentan las gráficas con la variación del error para ambos métodos en función de $C_t*R_t/R*C$. Se puede observar que para pequeños valores de $C_t*R_t/R*C$, correspondientes a interconexiones muy largas, el método analítico de Sakurai da mejores resultados que el método del polo dominante. Sin embargo, para valores de $C_t*R_t/R*C$ del

orden de 0,1 (correspondiente a interconexiones de longitud inferior a 10000 μ) ambos métodos obtienen resultados similares, y al aumentar dicho valor, es decir, al disminuir el tamaño de las interconexiones, el método del polo dominante produce errores mucho menores, que tienden rápidamente a 0.

En los ejemplos que se han tratado a lo largo de este trabajo, las longitudes de las interconexiones tienen un amplio rango de valores, pero todas ellas tienen longitudes inferiores a 10000μ. Por ejemplo, el área de una interconexión lejana para el filtro elíptico de 5° Orden era del orden de 120 p.e, lo que nos da una longitud de 1078μ. La longitud de las interconexiones cercanas tiene un valor muy inferior a éste (del orden de 80μ). Por otra parte, el área máxima de todos los ejemplos presentados era de 38000p.e., o lo que es lo mismo, un lado igual a 4mm. En el peor de los casos presentados, pueden existir interconexiones con longitudes de hasta 8000μ.

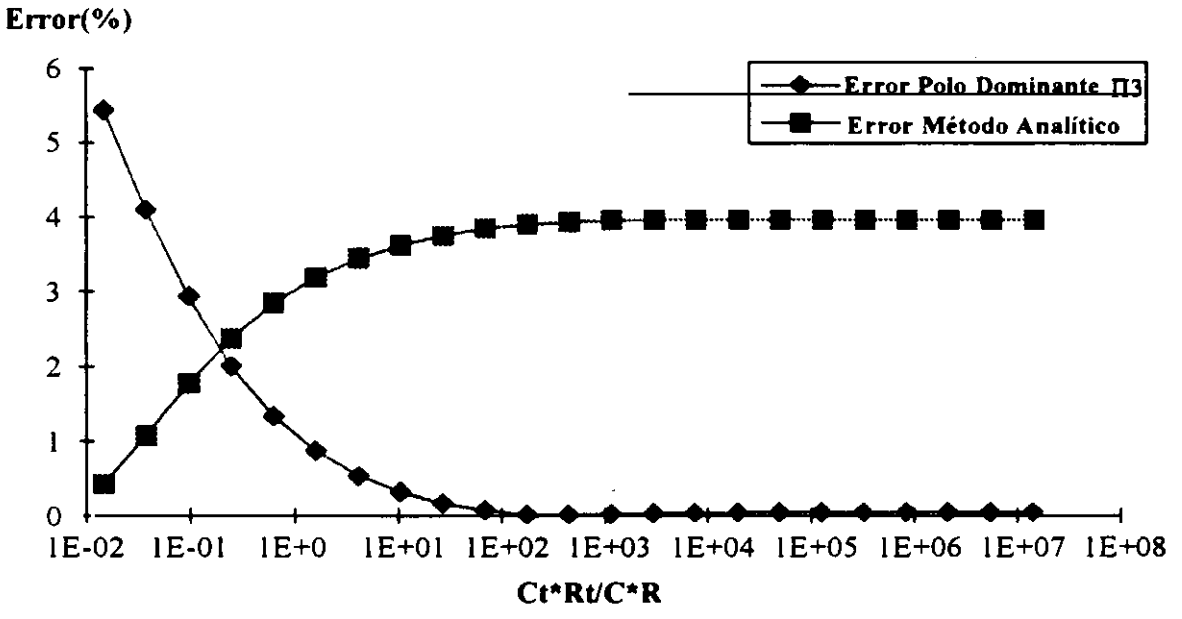


Figura 6.17 Variación del error para los dos métodos presentados

De los datos anteriores se puede deducir que, en la mayoría de los casos presentados en SAN, el método del polo dominante es el que produce errores menores. Sin embargo, en diseños muy grandes pueden existir interconexiones donde sea más apropiado utilizar el método

analítico de Sakurai. También será más apropiado para interconexiones entre distintos CIs. Como esto depende de cada caso particular, podemos dar como regla general que para obtener un error inferior al 2,5% se utilice el método del polo dominante si $R_t*C_t/R*C>0,25$, y el método analítico de Sakurai si $R_t*C_t/R*C<0,25$. Ambos métodos tienen una complejidad constante, por lo tanto la complejidad de la estimación no depende del método utilizado.

Sin embargo, hay que señalar que en la práctica, las interconexiones muy largas, donde sea necesario aplicar el método analítico de Sakurai, producen retardos excesivamente grandes, con lo cual el tiempo de ciclo del circuito debería ser también muy grande para asegurar un correcto funcionamiento. Esto se suele solucionar mediante otros métodos, haciendo más ancha la interconexión para disminuir la resistencia y con ello el retardo total, o bien añadiendo buffers intermedios que aceleran el proceso [WeEs94]. La primera solución produce una disminución en el valor de r (resistencia por unidad de longitud), y un incremento menor de c (capacidad por unidad de longitud), con lo cual la relación $R_t*C_t/R*C$ aumenta. En el segundo caso, la interconexión queda dividida en varias partes, cada una de una longitud menor. Ambas soluciones llevan a situaciones donde es preferible aplicar el método del polo dominante.

Una vez estimado el retardo de una interconexión de longitud L, necesitamos saber cuál es el valor de L que debemos utilizar para calcular el retardo debido a la propagación de las distintas señales en la ejecución de una operación. La longitud de cada interconexión es distinta y no se conoce hasta que no se ha realizado la colocación e interconexionado de módulos, por lo cual debemos estimar un valor que nos permita asegurar el correcto funcionamiento.

6.5. Estimación del retardo de propagación de señales en SAN

Los valores de C_t y R_t dependen únicamente de la tecnología de diseño y son conocidos. Sin embargo R y C dependen de la longitud de la interconexión ($C=c*L$ y $R=r*L$), y ésta no es

conocida hasta que no se ha realizado la colocación e interconexionado de módulos del circuito. En el capítulo anterior se vio que había cuatro retardos asociados a interconexiones: dos debidos a la transmisión de una señal desde un multiplexor a una UF o a un registro ($t_{MUX\_a\_UF}$ y $t_{MUX\_a\_REG}$), otro asociado a la interconexión desde la salida de la UF a la entrada del registro (o a su multiplexor), $t_{UF\_a\_MUXdeREG}$, y otro desde la salida del registro a la entrada del multiplexor de la UF, $t_{REG\_a\_MUXdeUF}$. Estos tipos de interconexiones pueden verse en la Figura 6.18.

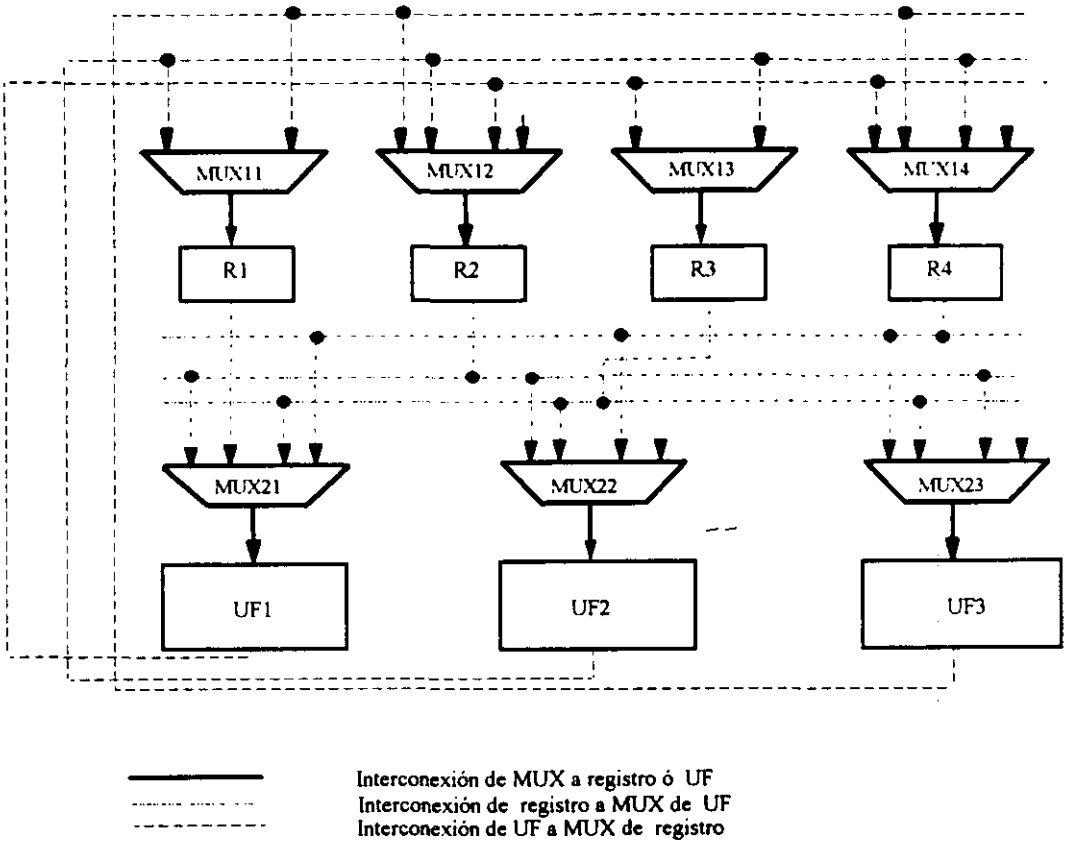


Figura 6.18 Tipos de Interconexiones

$$tprop = tUF\_a\_MUXdeREG + tREG\_a\_MUXdeUF + tMUX\_a\_UF + tMUX\_a\_REG$$

Si existen varias operaciones ejecutándose en paralelo en un paso de control, todas tendrán un retardo de interconexionado igual, y como todas las transmisiones se deben poder realizar en paralelo, **tprop** sólo se debe tener en cuenta una vez en cada paso de control. En el caso de

utilizar buses en el diseño, es necesario añadir también el retardo de un driver al tiempo de ciclo. El retardo asociado a un bus se puede considerar el mismo que el de una interconexión multimódulo.

Por otra parte, si se estudia la forma de trabajo de los algoritmos de colocación e interconexionado de módulos, se observa que en la mayoría de los casos los multiplexores se encuentran colocados muy próximos a las UFs y registros asociados a ellos. La longitud de estas interconexiones, que denominamos cercanas en el capítulo 4, L_{cerc}, se estimó que venía dado por la longitud media entre celdas estándar, cuyo valor es conocido. Por tanto:

$$tprop = tUF\_a\_MUXdeREG + tREG\_a\_MUXdeUF + 2 * t(Lcerc)$$

El valor del retardo de los otros dos tipos de interconexiones debe calcularse, pero hasta que no se ha realizado la colocación de los módulos en el diseño final no se sabe cuál va a ser la posición de los registros respecto a las UFs. Estas interconexiones deben pertenecer a uno de los tres tipos estudiados en el capítulo 4, cercanas, lejanas o multimódulo. A diferencia de las interconexiones cercanas, la longitud de los otros dos tipos de interconexiones varía con el tamaño del circuito y con las fórmulas dadas en el capítulo 4 lo que se obtiene es un valor medio.

Una posible solución que asegura un correcto funcionamiento, es tomar el valor de la longitud de una interconexión para el caso peor, que es el que determina el tiempo máximo necesario para transmisiones de señales. Este valor es el del medio perímetro del circuito, L_{max}, y su valor debe estimarse en cada caso. Por lo tanto, la longitud de una interconexión de cualquiera de estos dos tipos se toma como la mitad del perímetro máximo del circuito.

$$tUF\_a\_MUXdeREG = tREG\_a\_MUXdeUF = t(Lmax)$$

Y por tanto:

$$tprop = 2 * t(Lmax) + 2 * t(Lcerc)$$

donde: $Lmax = 2\sqrt{Area}$

Entre los ejemplos típicos de SAN que se han tratado a lo largo de este trabajo de investigación, el circuito de mayor tamaño tenía un medio perímetro de 8mm, lo cual equivale a un retardo de 2ns. Este es un valor aceptable si se tiene en cuenta los retardos de los módulos que se han utilizado, como por ejemplo los 30ns de un sumador o los 138ns de un multiplicador. Por tanto, consideramos que tomar como retardo de una interconexión de UF a registro o de registro a UF, el retardo de una interconexión con una longitud igual al medio perímetro del circuito, puede ser una aproximación válida, que asegura un correcto funcionamiento del circuito.

Para circuitos de gran tamaño, o cuando se desee minimizar el tiempo de ejecución lo más posible, o para tecnologías futuras donde la relación $C_l{*}R_l/C{*}R$ tiende a disminuir, este valor máximo de longitud de una interconexión puede ser "demasiado malo" para obtener los resultados fiables. Sin embargo, hay que tener en cuenta que las herramientas de diseño disponen de métodos para disminuir el retardo de las interconexiones. Como ya dijimos, es posible aumentar la anchura de estas, o bien introducir buffers que dividen la longitud total en dos o más partes, con lo cuál el retardo de cada una de estas partes se divide entre 4 o más. Además, es posible dirigir el floorplanning, para colocar próximos aquellos módulos que se encuentren en el camino crítico. Sólo utilizando estos métodos se puede conseguir que sistemas con interconexiones muy largas funcionen a frecuencias elevadas.

Durante un proceso de SAN podemos suponer que la herramienta de SBN dispone de distintos métodos para asegurar que el retardo asociado a las interconexiones no va a superar un cierto valor. Por ejemplo, puede disponer de un algoritmo que, una vez realizada la colocación de módulos, recorra todas las interconexiones del circuito, y cuando encuentra una con un retardo superior al máximo permitido, inserta los buffers necesarios para disminuir este valor. Si se utiliza este algoritmo, es posible fijar otro valor para el retardo de una interconexión de UF a registro o de registro a UF, como por ejemplo el retardo de una interconexión lejana, que es inferior al del medio perímetro del circuito. Cualquier

interconexión con una longitud mayor que la de una interconexión lejana debe de llevar un buffer que acelere la propagación de las señales. Sin embargo, para la tecnología de 1μ y los ejemplos típicos de SAN estas medidas no son necesarias, y puede utilizarse el medio perímetro para calcular el retardo de una interconexión, dando lugar a circuitos con un funcionamiento eléctrico correcto.

Por otra parte, como el área del circuito no se conoce hasta que no se ha realizado la asignación de hardware, y necesitamos calcular el retardo de las interconexiones para estimar el tiempo de ciclo, es decir, antes de la planificación, es imprescindible estimar dicha área. Una posible solución, que nos permite acotar el valor del medio perímetro, es estimar el área máxima del circuito, para lo cual necesitamos conocer el número máximo de registros, UFs e interconexiones que se van a utilizar. Sin embargo, estos valores no es posible estimarlos con un error dentro de unos márgenes razonables antes de realizar la planificación. La solución que nosotros proponemos es:

A.- **Estimación de *tprop***: Esta fase se realiza en los siguientes pasos:

- **Paso 1**: Estimar un tiempo de ciclo inicial sin tener en cuenta el retardo debido a la transmisión de señales, usando los algoritmos presentados en el capítulo anterior.

- **Paso 2**: Realizar la planificación, mediante un algoritmo de planificación por listas [SSHF89], para dicho tiempo de ciclo inicial, usando para implementar cada operación el módulo más rápido de la biblioteca.

- **Paso 3**: Estimar el área máxima para esa planificación, ***Area_max***, y calcular el valor máximo del medio perímetro del circuito, L_{max}***Plan_Listas.*** Este valor se toma como la longitud de una interconexión de registro a MUX de UF o de UF a MUX de registro, y se calcula el retardo asociado a ella.

- **Paso 4**: Calcular *tprop* utilizando:

$$tprop = 2 * t_{0.9}(LmaxPlan\_Listas) + 2 * t_{0.9}(Lcer)$$

Si el valor de t_{prop} obtenido es fraccionario, redondeamos siempre tomando el entero mayor más cercano, con el fin de asegurar un correcto funcionamiento del circuito.

B.- **Estimación del tiempo de ciclo, planificación y asignación de hardware**. Teniendo en cuenta el retardo de las interconexiones, se recalcula el valor del tiempo de ciclo y se vuelve a realizar la planificación, utilizando el algoritmo de planificación-preasignación de FIDIAS. A continuación se realiza la asignación de hardware y se obtiene el área final del circuito, ***A_final***.

C.- **Recálculo de *tprop* y ajuste del tiempo de ciclo**. Con el valor final del área del circuito se recalcula el medio perímetro del circuito, ***L_final***, y se obtiene el nuevo valor del retardo de propagación de señales t_{prop_nuevo}:

$$t_{prop\_nuevo} = 2 * t_{0.9}(L\_final) + 2 * t_{0.9}(Lcer)$$

Con este dato, se genera la diferencia entre el valor estimado anteriormente y el nuevo, $\mathbf{t_{dif}}$.

$$t_{dif} = t_{prop} - t_{prop\_nuevo}$$

Es posible disminuir el tiempo de ciclo en un valor igual a t_{dif} dividido entre el máximo número de etapas que necesitan las UFs del circuito para ejecutar una operación, sin que sea necesario modificar la planificación ni la asignación de hardware.

A continuación presentamos los pseudocódigos de cada una de estas fases.

El pseudocódigo de la fase A, es decir, el algoritmo de estimación de t_{prop}, parte de un GFD sin planificar y una biblioteca de módulos donde se conocen los retardos de los operadores, sin tener en cuenta el retardo de la propagación de señales.

Estima_tprop(GFD,Biblioteca)
tciclo=Estimacion_tciclo(GFD,Biblioteca)
GFD_Planificado=Planificación_Listas(GFD,Biblioteca,tciclo)
Area_max=Estima_ÁreaMax(GFD_Planificado)
$LmaxPlan\_Listas = 2\sqrt{Area\_max}$
$tprop=2*t_{0.9}(LmaxPlan\_Listas)+2*t_{0.9}(Lcer)$

En la fase B, con el valor estimado de t_{prop}, se recalculan los retardos de todas las UFs de la biblioteca, teniendo en cuenta el retardo de las interconexiones; así se obtiene lo que denominamos la ***Nueva_Biblioteca***, que es exactamente igual que la inicial pero con los retardos de las UFs actualizados. Con esta *Nueva_Biblioteca,* se recalcula el tiempo de ciclo, y luego se realiza la planificación y asignación de hardware. El algoritmo global de SAN es:

Nueva_Biblioteca=Añadir_retardo(Biblioteca,tprop)
t_{ciclo}*=Estimacion_tciclo(GFD,Nueva_Biblioteca)*
GFD_Planificado=Planificación_Preasignación(GFD,Nueva_Biblioteca,tciclo)
GFD_Asignado=Asignación_Hardware(GFD_Planificado, Nueva_Biblioteca)
A_final=Calcula_Area(GFD_Asignado)

Por último, el pseudocódigo de la fase C es:

$L\_final = 2\sqrt{A\_final}$
$t_{prop\_nuevo}=2*t_{0.9}(L\_final)+2*t_{0.9}(Lcer)$
$t_{dif}=t_{prop}-t_{prop\_nuevo}$
t_{ciclo_nuevo}*=Recalcula_ciclo(tciclo,tdif,GFD_Asignado)*

La planificación por listas nos proporciona sólo un área aproximada del circuito final. Como se trata de optimizar el área, el área estimada para esta planificación en ningún caso será inferior al área final del circuito, es decir:

$$Area\_max >= Afinal$$

Por lo tanto, la longitud del medio perímetro calculada para la planificación por listas, será siempre mayor que la calculada para la planificación final del diseño:

$$LmaxPlan\_Listas = 2\sqrt{Area\_max} > L\_final.$$

Teniendo esto en cuenta, se puede deducir que el retardo máximo asociado a la propagación de señales, estimado para la planificación por listas, es siempre superior al retardo de propagación de señales estimado para el circuito final. Esta conclusión nos permite asegurar un correcto funcionamiento del circuito, con las suposiciones realizadas.

Esta estimación inicial del retardo de una interconexión es un valor superior al real. Como nos interesa obtener diseños con tiempos totales de ejecución optimizados, es necesario revisar este valor una vez realizada la planificación final y la asignación de hardware y, si es posible, reducir el tiempo de ciclo según se ha explicado anteriormente. Después de la asignación de hardware, el área total del circuito puede estimarse con una precisión superior al 95%, como vimos en el capítulo 4, con lo cual el retardo máximo de una interconexión puede calcularse con mucha precisión. En aquellos casos en los que el tiempo de ciclo varíe mucho, puede ser interesante realizar una nueva planificación y por tanto una nueva asignación de hardware.

6.6. Ejemplo

Para clarificar el proceso total de SAN, teniendo en cuenta el retardo de las interconexiones, vamos a utilizar el Filtro Elíptico de 5° Orden. La biblioteca de módulos se presenta en la tabla 6.4. Vamos a tratar de obtener un diseño donde el área sea prioritaria frente al tiempo, pero no sea totalmente prioritaria.

modulo	op	retardo
sumador	+	30ns
sumador	+	24ns
multiplicador	*	138ns

Tabla 6.4 Biblioteca para el Filtro Elíptico de 5° Orden

A.- **Estimación de *tprop*.**

- **Paso 1**: Como la minimización de área es más importante que la de tiempo, elegimos los parámetros de estimación del tiempo de ciclo con valores:

$$\alpha=0.3 \text{ y } \beta=0.7$$

Para estos valores, y la biblioteca de la tabla 6.4 obtenemos un tiempo de ciclo de 69ns.

- **Paso 2**: El resultado de la planificación por listas se presenta en la figura 6.19. Se obtienen un total de 18 pasos de control.

- **Paso 3**: Para esta planificación se necesitan al menos dos sumadores y dos multiplicadores. En total obtenemos un área de 20069 p.e.. El medio perímetro del circuito es:

$$L_{max}Plan\_Listas = 5{,}858\text{mm}.$$

- **Paso 4**: Estimamos el retardo asociado a una interconexión de longitud 5,858mm, y obtenemos:

$$t_{0.9}(LmaxPlan\_Listas) = 1{,}412\text{ns}.$$

El retardo de una interconexión cercana para la tecnología de 1μ es un dato conocido:

$$t_{0.9}(Lcer)=0{,}127\text{ns}.$$

Con estas estimaciones,

$$t_{prop}=2*1{,}412\text{ns} + 2*\ 0{,}127\text{ns} = 3{,}078\text{ns}.$$

Para asegurar un correcto funcionamiento del circuito tomamos

$$t_{prop}=4\text{ns}.$$

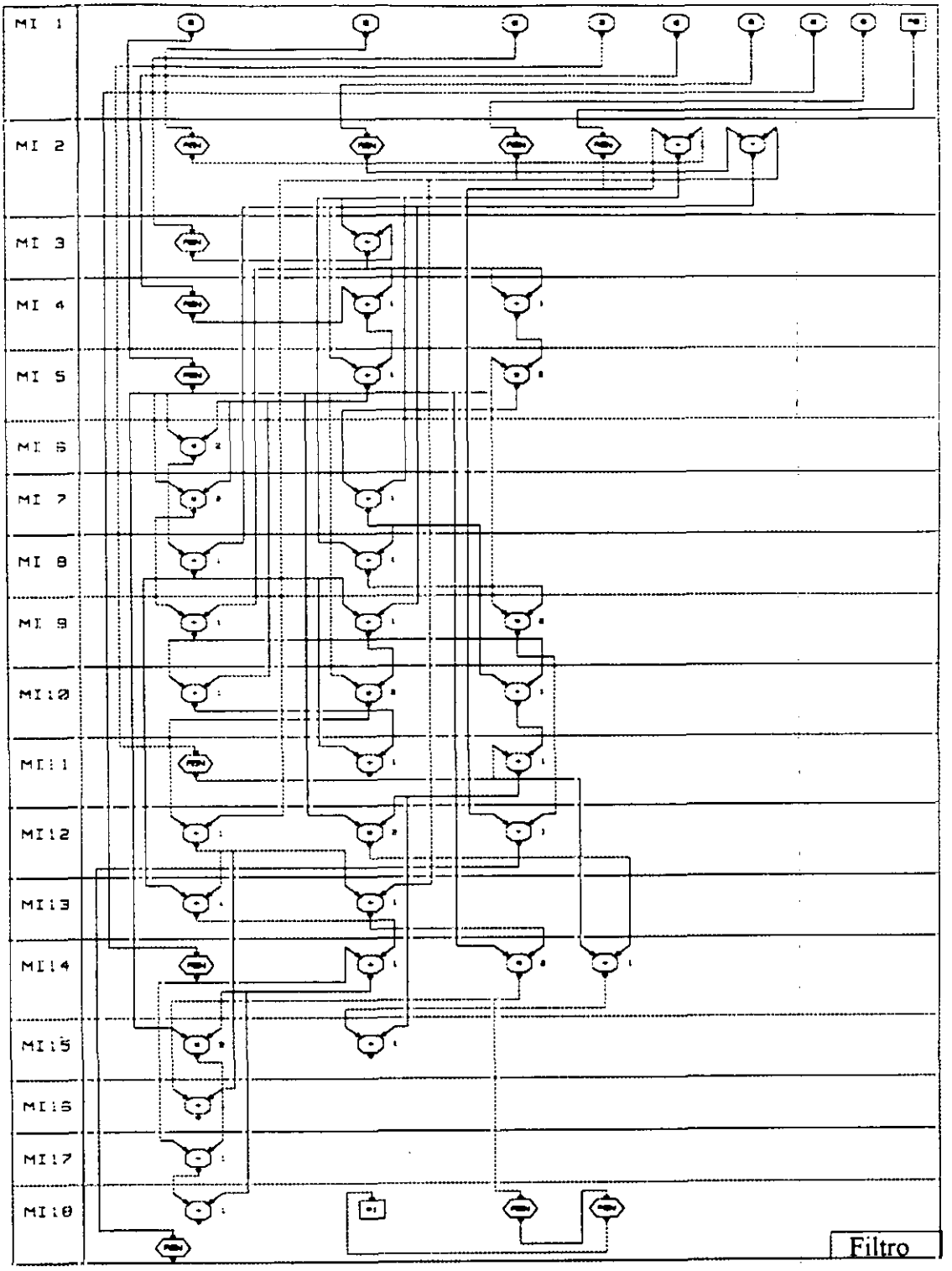


Figura 6.19 Planificación para el Filtro Elíptico de 5° Orden usando un Planificador por Listas y un tiempo de ciclo de 69ns.

B.- **Estimación del tiempo de ciclo, planificación y asignación de hardware**. Calculamos el tiempo de ejecución de todas las operaciones, teniedo en cuenta el retardo de las interconexiones. Los resultados se presentan en la tabla 6.5.

modulo	op1	retardo
sumador	+	34ns
sumador	+	28ns
multiplicador	*	142ns

Tabla 6.5 Biblioteca con retardos de interconexiones para el Filtro Elíptico de 5° Orden

El tiempo de ciclo obtenido para esta biblioteca, con los mismos valores de α y β es:

$$t_{ciclo} = 71\text{ns}.$$

La nueva planificación se muestra en la figura 6.20. Se puede observar que se ha incrementado el número de microinstrucciones de 18 a 23. Esto es debido a que la minimización de área es más importante que la de tiempo, y de esta forma sólo es necesario utilizar un multiplicador, con lo cual el área se reduce notablemente.

C.- **Recálculo de *tprop* y ajuste del tiempo de ciclo**.

Después de realizar la asignación de hardware, se obtiene

$$A\_final = 10.777 \text{ p. e.}.$$

Con lo cual:

$$L\_final = 4{,}294 \text{ mm}.$$

El retardo asociado a esta longitud es de

$$t_{0.9}(L\_final) = 1{,}051\text{ns}$$

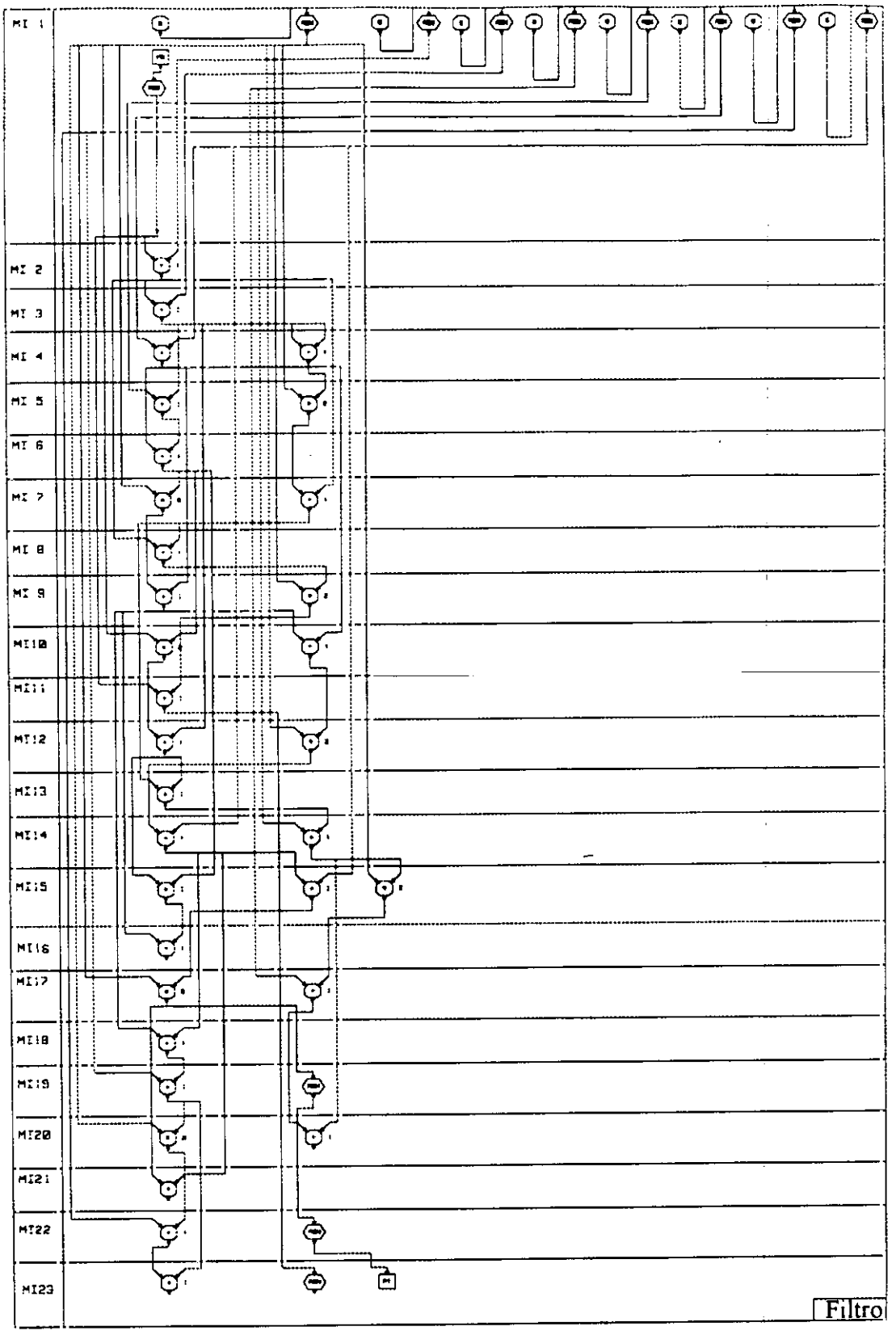


Figura 6.20 Planificación para el Filtro Elíptico de 5° Orden usando un Planificador-Preasignador y un tiempo de ciclo de 71ns.

por tanto:

$$t_{prop\_nuevo} = 2 * 1{,}051\text{ns.} + 2 * 0{,}127\text{ns} = 2{,}356\text{ns.}$$

Con este nuevo valor del retardo de propagación de señales obtenemos:

$$t_{dif} = t_{prop} - t_{prop\_nuevo} = 4ns - 2{,}356ns = 1{,}644\text{ns.}$$

Como el máximo número de etapas de cualquier operación del grafo es 2, se podría disminuir el tiempo de ciclo en un valor igual t_{dif} dividido entre 2, es decir, 0,822ns. Como estamos midiendo el tiempo de ciclo en valores enteros de nanosegundos, y el valor obtenido es inferior a la unidad, dejamos el tiempo de ciclo como está.

$$t_{ciclo\_nuevo} = t_{ciclo}$$

La planificación y asignación de hardware finales son las obtenidas en la fase B.

6.7. Conclusiones

El objetivo principal de este capítulo era la estimación del retardo debido a propagación de señales, para calcular el retardo de todas las operaciones del GFDC y seleccionar un ciclo de reloj, según los algoritmos vistos en el capítulo anterior.

Como el retardo asociado a la propagación de señales depende de la longitud de las interconexiones implicadas, en primer lugar se ha presentado un método para calcular el retardo de una interconexión de longitud L. El modelo físico utilizado es un Π3. El algoritmo computa la frecuencia del polo dominante y a partir de ahí el retardo asociado. Se ha comprobado que este método permite obtener resultados con errores inferiores al 2% para interconexiones de tamaño inferior a 0,7 cm., que corresponde a la mayoría de los ejemplos tratados a lo largo de este trabajo de investigación. También se presenta otro método alternativo para interconexiones de mayor longitud.

Como la longitud de las interconexiones no se conoce hasta que no se ha realizado la

colocación e interconexionado de módulos, es necesario estimar un valor que nos permita obtener un valor para el tiempo de ciclo y que asegure un correcto funcionamiento del circuito. Para este fin, se ha utilizado el retardo de la interconexión de máxima longitud, es decir, el medio perímetro del circuito.

Para circuitos de gran tamaño o para tecnologías inferiores a 1μ, este caso peor puede ser "demasiado malo", y sería necesario acotar el retardo máximo de una interconexión mediante otros valores, como por ejemplo el retardo de una interconexión de longitud media. Aunque en este caso no se puede asegurar el correcto funcionamiento del circuito, pueden utilizarse algoritmos en las herramientas de SBN, que cuando encuentran interconexiones con longitudes superiores a las medias, introduzcan buffers que disminuyan el retardo de éstas. Sin embargo, para la tecnología de 1μ y los ejemplos habituales presentados en SAN, esto no es necesario.

Dado que la información que se dispone sobre el circuito antes de realizar una planificación es muy poca, las estimaciones que se pueden realizar sobre la longitud máxima de una interconexión antes de realizar una planificación tienen un error demasiado elevado para poderlas utilizar en el resto del proceso. Por eso, en nuestro sistema FIDIAS se realiza una primera selección del ciclo de reloj sin tener en cuenta el retardo de las interconexiones. Con este valor se realiza una planificación por listas, se estima el área máxima del circuito para esta planificación, y se calcula el medio perímetro del circuito. A partir de este valor se estima el retardo debido a propagación de señales y se calcula el tiempo de ciclo, que se utiliza para realizar la planificación y la asignación de hardware.

Una vez terminada la asignación de hardware, el área del circuito se puede estimar con una precisión del 95%. Con este valor, se recalcula el retardo máximo de una interconexión, el retardo debido a propagación de señales y la diferencia entre el valor estimado inicialmente y el final. Es posible disminuir el tiempo de ciclo en un valor igual a esta diferencia dividida por el máximo número de etapas necesaria para ejecutar una operación, sin que sea necesario realizar otra planificación ni otra asignación de hardware.

Con las estimaciones realizadas a lo largo de los capítulos 4, 5 y 6 es posible disponer de información, durante todo el proceso de SAN, sobre el retardo y área de todos los módulos e interconexiones del circuito, lo cual nos permite dirigir correctamente el proceso de diseño, obtener resultados que se ajusten a los objetivos del usuario, y diseños con una simulación eléctrica correcta.

Principales Aportaciones

- En primer lugar se ha presentado un método para realizar estimaciones del área de un circuito durante un proceso de Síntesis de Alto Nivel. Este método está basado en un estudio de la tecnología de diseño de celdas estándar, la forma de trabajo de los algoritmos de colocación e interconexionado de módulos y el diseño en particular. Gracias a ello, se obtienen estimaciones muy **precisas**, que pueden generarse mediante **cálculos simples**, lo cual supone un avance respecto de otros métodos propuestos en el campo de la SAN.

- Se han **clasificado las interconexiones** en función del número de módulos que conectan. Para cada uno de los tipos se ha obtenido una estimación de su longitud. Las interconexiones que hemos denominado cercanas tienen longitud constante. Para las lejanas y multimódulo se ha obtenido una fórmula, en función del número total de interconexiones de cada tipo, y del área de celdas estándar del diseño, datos conocidos cuando se ha terminado la generación de un diseño.

- Este método de estimación de la longitud de las interconexiones nos permite conocer el **área de los módulos**, siempre que en la biblioteca se disponga de información sobre el número y tipo de interconexiones que contiene cada uno de ellos. De esta forma, el cálculo del área total o parcial del circuito se convierte en un algoritmo de complejidad proporcional a N, donde N es el número de módulos, y una precisión del 95%.

- Además, se ha presentado un método para estimar el área de los módulos y de cada una de las interconexiones del circuito durante la generación de éste, cuando aún no se conocen el número y tipo de módulos e interconexiones totales. Por eso, las estimaciones pueden utilizarse tanto para los algoritmos iterativos, para comparar diseños completos entre si, como para los constructivos, que realizan las estimaciones durante la generación de un

circuito, para guiarla correctamente, y, por tanto, para los constructivo-iterativos. Como ejemplo de integración en un sistema de Síntesis de Alto Nivel, estas estimaciones se han incluido en el sistema FIDIAS, cuyo algoritmo de asignación podemos decir es de tipo constructivo-iterativo, ya que necesita realizar estimaciones tanto durante la creación de un diseño, como al final del mismo.

- Se ha realizado la conexión de la salida del sistema de SAN con herramientas CAD, con el fin de terminar el proceso de diseño y llegar a la generación del layout, y también para obtener información sobre la metodología de trabajo de estas herramientas y sobre la tecnología utilizada. Esto nos ha permitido mejorar la fidelidad de las estimaciones realizadas sobre las características físicas. Además, esta conexión permite un flujo de intercambio de información en ambos sentidos: puede realizarse la evaluación del diseño real, de forma que, si no se cumplen los objetivos deseados, se puede volver al inicio del proceso, pero ahora disponiendo de mucha más información sobre las características físicas del circuito. De esta forma, se puede guiar mejor el siguiente proceso y aumentar la precisión de los resultados obtenidos en las siguientes fases del diseño.

- Se ha presentado un método para realizar una selección inteligente del tiempo de ciclo, que tiene en cuenta el GFD, la biblioteca de módulos y las restricciones del usuario. Se ha demostrado que el tiempo de ciclo óptimo es muy diferente dependiendo del GFD, de la biblioteca de módulos y de las restricciones del usuario. En primer lugar, a partir del GFD y de la biblioteca de módulos, se realiza una búsqueda de los posibles caminos críticos del circuito, que varían dependiendo del tiempo de ciclo. A partir de esta lista de caminos críticos, se aplica un algoritmo de selección que tiene en cuenta los objetivos del usuario y la biblioteca de módulos. Así se obtiene un tiempo de ciclo óptimo para cada diseño en particular. Como la complejidad de este algoritmo de selección es baja, puede utilizarse sin incrementar excesivamente el tiempo de ejecución del algoritmo de planificación.

- Se ha realizado una estimación del retardo de una interconexión de longitud L. El modelo físico utilizado para modelar una interconexión es un Π3, y para calcular su retardo se

utiliza el método del polo dominante. Se ha comprobado que este método permite obtener resultados con errores inferiores al 2% para interconexiones de tamaño inferior a 0,7 cm., que corresponde a la mayoría de los ejemplos tratados a lo largo de este trabajo de investigación. También se presenta otro método alternativo para interconexiones de mayor longitud.

- También se ha presentado un método para incluir los retardos de la interconexiones en la selección del tiempo de ciclo. Como la longitud de las interconexiones no se conoce hasta que no se ha realizado la colocación e interconexionado de módulos, es necesario estimar un valor que nos permita obtener un valor para el tiempo de ciclo y que asegure un correcto funcionamiento del circuito. Para este fin se ha utilizado el retardo de la interconexión de máxima longitud, es decir, el medio perímetro del circuito.

- Por último apuntar que, con las estimaciones realizadas a lo largo de este trabajo de investigación sobre las características físicas del diseño, es posible disponer de información, durante todo el proceso de Síntesis de Alto Nivel, sobre el retardo y área de todos los módulos e interconexiones del circuito, lo cual nos permite dirigir correctamente el proceso y obtener resultados que se ajusten a los objetivos del usuario y que tengan una simulación eléctrica correcta.

Trabajo Futuro

Las estimaciones sobre las características físicas de un diseño presentadas en este trabajo de investigación, se han basado en un tipo de tecnología de diseño de Circuitos Integrados: las celdas estándar. Aunque éste es uno de los estilos más usuales para el diseño de ASICs, también existen otros estilos de diseño, como son los arrays de celdas, las macroceldas o las FPGAs, que también se utilizan con bastante frecuencia en la actualidad. Por esta razón, una de las líneas futuras de investigación será ampliar las técnicas de investigación de las características físicas a otros estilos de diseño.

El diseño con arrays de celdas tiene unas características similares al diseño con celdas estándar, lo cual nos induce a pensar que las variaciones que se deberán introducir en nuestro sistema de Síntesis para esta nueva tecnología serán mínimas. Tampoco creemos que el diseño mixto con celdas estándar y macroceldas necesite consideraciones muy diferentes a las realizadas en este trabajo, con lo cuál sólo será necesario realizar algunos ajustes en las fórmulas de las estimaciones de área.

Sin embargo, el diseño con FPGAs tiene unas características muy diferentes. Se trata de circuitos que contienen un conjunto de bloques de lógica configurables, y para los cuáles el proceso de fabricación ya ha finalizado. Por lo tanto el proceso de Síntesis de Alto Nivel debe ser muy diferente al de los estilos anteriores, ya que no se trata de conseguir un circuito con área mínima, sino un circuito que pueda diseñarse en una FPGA. Consecuentemente, la adaptación de nuestro sistema a este nuevo estilo de diseño, no sólo será un ajuste de las funciones de estimación de área y retardo, sino una adaptación del sistema completo de Síntesis de Alto Nivel a la nueva tecnología.

Por otra parte, también pretendemos seguir investigando sobre las nuevas tecnologías de

celdas estándar, inferiores a 1μ y con dos o más capas de metal para realizar interconexiones. Nuestros primeras investigaciones sobre el área de los circuitos se realizaron para la tecnología de 1,2μ, y cuando se comenzó a utilizar la tecnología de 1μ, sólo tuvimos que variar algunos datos, como el ancho de una celda estándar o el ancho de una interconexión. Por esta razón, suponemos que para las tecnologías inferiores a 1μ, que utilicen dos capas de metal, nuestras estimaciones de área seguirán siendo válidas, y sólo será necesario medir los nuevos parámetros.

Sin embargo, no ocurrirá lo mismo con las estimaciones del retardo de las interconexiones. Para las tecnologías inferiores a 1μ el área de interconexionado es cada vez mayor respecto al área de celdas estándar, lo cual produce una disminución de la relación de la capacidad de entrada a los transistores frente a la capacidad de la interconexión. Esto conduce a diseños donde el retardo de las interconexiones es cada vez más importante, e incluso dominante, en el retardo final del circuito.

Por lo tanto, utilizar el máximo retardo de una interconexión como retardo de cualquier interconexión, puede llevar a resultados poco fiables, o a sistemas con tiempos de ejecución demasiado elevados. Por esta razón, y como ya se apuntó en el capítulo 6, será necesario acotar el retardo máximo de una interconexión mediante otros valores, como por ejemplo el retardo de una interconexión de longitud media. Aunque en este caso no se puede asegurar el correcto funcionamiento del circuito, pueden utilizarse algoritmos en las herramientas de diseño de circuitos, que cuando encuentren interconexiones con longitudes superiores a las máximas permitidas, introduzcan buffers que disminuyan el retardo de éstas, o bien incrementen el ancho de la interconexión.

En este sentido, nuestro trabajo de investigación seguirá dos líneas principales: por un lado la realización de simulaciones eléctricas en circuitos con tecnologías inferiores a 1μ , para obtener resultados que nos permitan acotar el retardo medio de una interconexión de forma más fiable. Por otro lado, la generación de un algoritmo que permita dirigir la colocación de módulos e interconexiones, con el objetivo de que no se realicen interconexiones con retardos

mayores que los estimados.

También será necesario realizar un nuevo estudio sobre la forma de trabajo de los algoritmos de colocación e interconexionado de módulos, para tecnologías que utilicen más de dos capas de metal.

Por otra parte, también pretendemos que el flujo de información de la herramienta de Síntesis de Alto Nivel hacia las herramientas de diseño aumente. Hay que tener en cuenta que la información disponible en Síntesis de Alto Nivel sobre un circuito es muy grande, puesto que se conocen los caminos críticos, los retardos máximos de las UFs e interconexiones, el tiempo de ciclo, etc. Por esta razón, parece lógico que esta información se utilice para dirigir las fases de colocación e interconexionado de módulos, con el fin de obtener circuitos que cumplan las restricciones y objetivos del usuario y tengan un funcionamiento eléctrico correcto.

Apéndice A

Estilos de Diseño de Circuitos VLSI

A.1. Diseño VLSI

Desde que en 1960 surgió el concepto LSI (Large Scale Integration), las herramientas de CAD (*Computer Aided Design ó* Diseño Asistido por Computador) se han convertido en ayudas indispensables para disminuir el esfuerzo en diseño y verificación de Circuitos Integrados. Esto se ha acentuado aún más con la tecnología VLSI (Very Large Scale Integration), que ha tenido su desarrollo en los últimos años, y que ha dado lugar al nacimiento de los ASICs (Circuitos Integrados de Aplicación Específica).

Las herramientas CAD posibilitan la automatización del proceso total de generación de layout que sigue a la fase de diseño VLSI. Esta automatización es posible también gracias a distintas técnicas de estandarización de la metodología de diseño, como son los arrays de puertas o el diseño con celdas estándar. Cada una de estas técnicas de estandarización, también conocidas como **estilos de diseño**, está unida a la existencia de diferentes paquetes software que realizan las fases de colocación e interconexionado de módulos.

El coste de un circuito integrado es el coste de diseño mas el coste de fabricación. La estándarización permite disminuir el esfuerzo de diseño, y por tanto el coste de diseño. Pero a cambio, disminuye la densidad de empaquetamiento y la velocidad del circuito. Como la densidad de empaquetamiento está relacionada con el coste de fabricación del circuito, y a más densidad del circuito menos coste, según aumenta el nivel de estandarización el coste de fabricación de cada circuito se incrementa, pero el de diseño disminuye. En la tabla A.1 se

presenta una lista con los principales estilos de diseño y su nivel de estandarización correspondiente, ordenados de menor a mayor.

Estilo de diseño	**Nivel de estandarización**	**Objeto de estandarización**
Full-custom	Elemento	Regla de diseño
Diseño simbólico	Dispositivo	Simbolización de transistores, vías, conexiones
Celdas estándar	Celda	Normalización de altura de la celda, localización de terminales, anchura de interconexiones
Macro-celdas	Bloque	Normalización de posición de terminales de bloques
Arrays de puertas	Colocación de celdas y bloques	Estandarización de celdas y colocación de bloques
PLA-ROM-LCA	Cableado	Estandarización de conexiones entre dispositivos

Tabla A.1 Estilos de diseño

Si se fabrican N circuitos iguales, el coste de fabricación total es el coste de cada uno de ellos por N. Sin embargo, el coste de diseño es el mismo para un circuito que para N. Por lo tanto, el coste global viene dado por el coste de fabricación de los N circuitos más el coste del diseño. Para pequeños valores de N, el coste de diseño y de fabricación tienen el mismo peso, y no resulta rentable que el primero sea muy grande, por eso se suelen utilizar técnicas con alto nivel de estandarización. Sin embargo, para grandes valores de N, el coste de fabricación total tiene mayor peso que el de diseño, y resulta rentable emplear más tiempo en el diseño, utilizando técnicas de nivel de estandarización poco elevado, con lo cual el coste de fabricación disminuye.

Los sistemas que necesitan una gran velocidad o gran densidad de empaquetamiento (la demanda es muy elevada) suelen utilizar las técnicas de full-custom, mientras que los que

necesitan generar un circuito en el menor tiempo de fabricación posible, donde la velocidad de ejecución del propio circuito no es primordial, utilizan PLAs o ROMs. En una situación intermedia se encuentra el diseño con celdas estándar, macro-celdas y arrays de puertas, que suelen conseguir gran densidad de empaquetamiento y el tiempo de diseño es considerablemente inferior al del diseño full-custom. Suelen utilizarse en circuitos donde el volumen de producción no es muy elevado, por lo cual el coste de desarrollo tiene un gran peso sobre el coste total de los circuitos. En estos casos no resulta rentable emplear el tiempo de diseño necesario en full-custom para conseguir más densidad de empaquetamiento.

En los apartados siguientes se tratarán detalladamente cada una de estas técnicas. Una buena revisión sobre los diferentes estilos de diseño se puede encontrar en [Muro82] y [UeKS86].

A.2. Full-custom

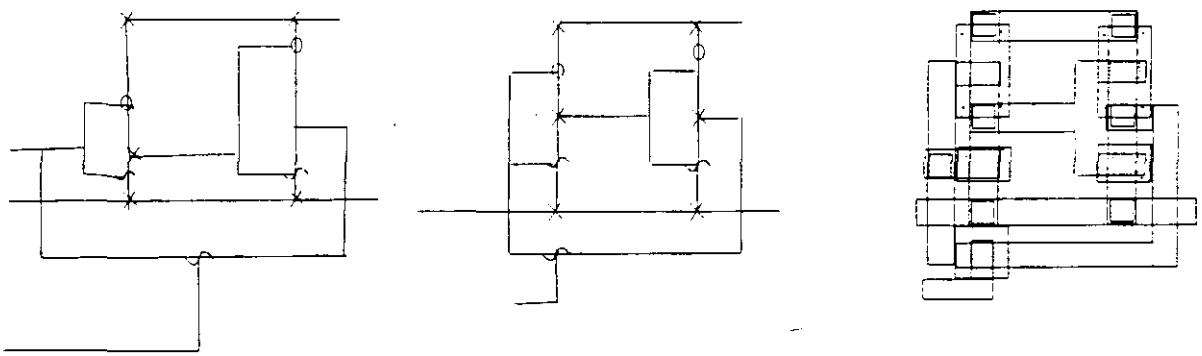
En el estilo de diseño full-custom o diseño a medida, el empaquetamiento es realizado por el diseñador, con la ayuda de herramientas gráficas. No se dispone de ninguna restricción estructural sobre el diseño. La minimización de área (o densidad de empaquetamiento) es máxima y por tanto el rendimiento del circuito es muy alto. Sin embargo, se precisa una gran inversión en tiempo de desarrollo y fuerza de trabajo, que se incrementa enormemente con la complejidad de los circuitos. Además es muy importante la experiencia del diseñador o diseñadores para obtener buenos resultados.

Debido al elevado tiempo de diseño no es posible utilizar este estilo a gran escala, ni para redes muy complejas. Sin embargo, cuando la demanda es elevada, el coste de desarrollo del circuito puede incrementarse si se consigue disminuir notablemente el coste de producción, con lo cual el coste total del CI se abarataría. En este caso surge la necesidad de obtener una gran densidad de empaquetamiento, y es necesario optimizar los circuitos individuales a nivel de transistor.

Otra aplicación que se presta a un diseño full-custom es cualquier diseño regular, formado por un elemento base de lógica que, una vez optimizado, puede repetirse varias veces en el CI.

A.3. Diseño simbólico

Se utilizan componentes simbólicas, como transistores, vías y conexiones. La imagen del layout se diseña manualmente, y la conversión de los símbolos en modelos físicos y su compactación está automatizada (ver figura A.1). De esta forma el esfuerzo de diseño se reduce, pero se mantiene alta la densidad de empaquetamiento, aunque menor que para el diseño full-custom. Suele utilizarse en sistemas donde la demanda es muy elevada.



Entrada del diseñador | 1er Paso: espaciamiento | 2do Paso: Generación de máscaras

Figura A.1 Estilo de diseño simbólico

A.4. Celdas estándar

Las celdas estándar son módulos lógicos, en general simples (puertas lógicas, flip-flops), que tienen un layout interno prediseñado. Todas son de la misma altura, pero diferente anchura, dependiendo de la funcionalidad. Una herramienta CAD suele tener una biblioteca de unas 30-40 celdas, que es necesario actualizar cada vez que cambia la tecnología. La colocación e

interconexionado se suele realizar automáticamente.

Las celdas se colocan en filas, todas ellas de una longitud similar, entre las cuales quedan los canales para realizar las interconexiones (ver figura A.2). En cada uno de estos canales pueden existir un número variable de pistas, en cada una de las cuales pueden realizarse varias interconexiones siempre que no se solapen.

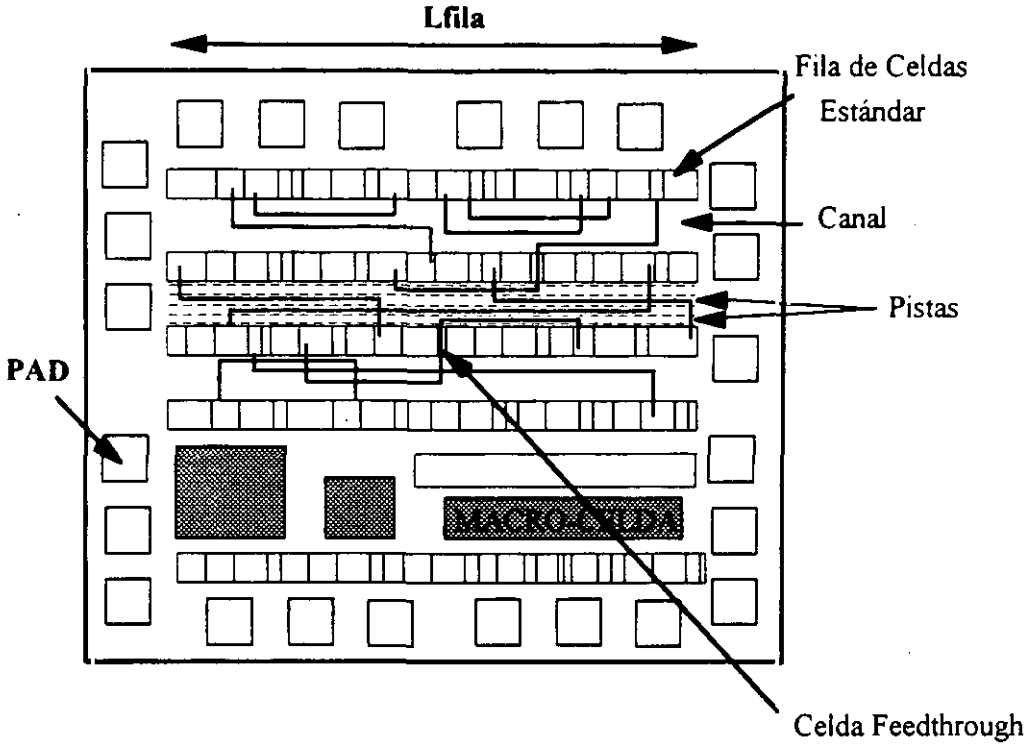


Figura A.2 Diseño con celdas estándar

Las celdas estándar suelen diseñarse para que las conexiones de alimentación y tierra puedan correr horizontalmente en la parte de arriba y abajo de las celdas, como se muestra en la figura A.3. Como las celdas se conectan una adyacente a la siguiente, estas interconexiones forman una pista continua en cada fila. Las entradas y salidas lógicas de las celdas son los terminales, que se colocan en el borde superior y/o inferior.

Las conexiones entre una fila y otra se realizan mediante canales verticales en los bordes del circuito, o mediante celdas *feedthrough o* "celdas de atravesamiento", que son celdas estándar

(con la misma altura que el resto), con una serie de interconexiones que corren verticalmente a través de ellas (ver figura A.2). Normalmente se utilizan dos capas de metal para realizar las interconexiones: una para los tramos horizontales, que se sitúan en las pistas de los canales, y otra para los verticales. La unión entre ambos tipos de tramos se realiza mediante una vía. Es aconsejable utilizar el mínimo número de vías posibles, ya que penalizan el retardo de la transmisión.

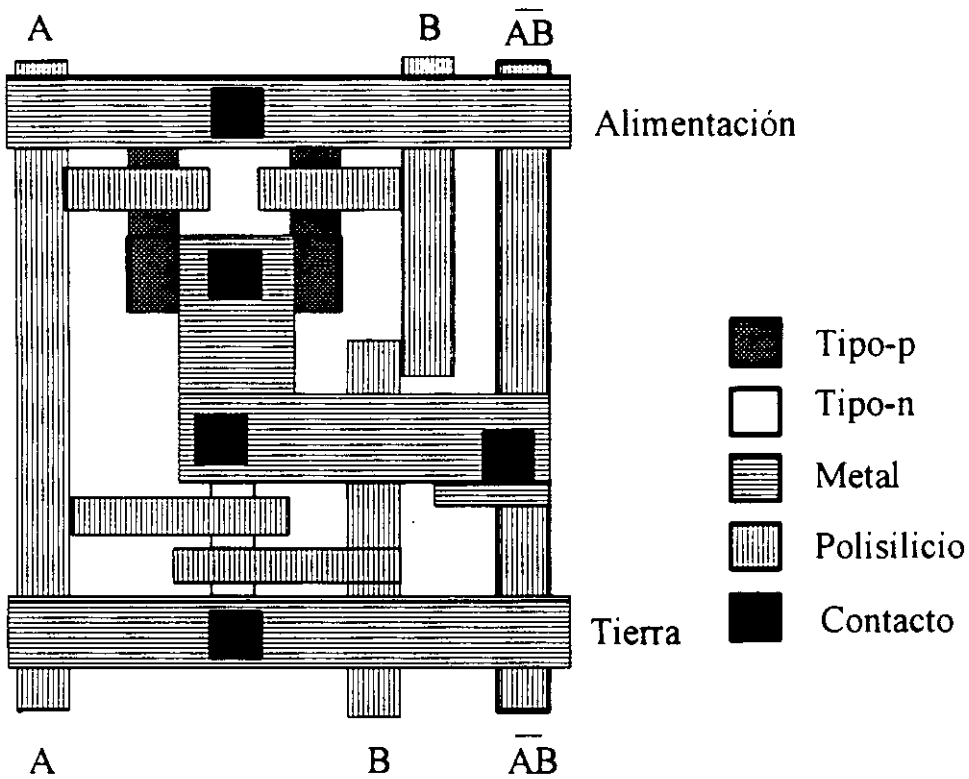


Figura A.3 Layout de una celda estándar

El proceso de diseño con celdas estándar tiene como objetivo la generación de todas las máscaras necesarias para la fabricación del circuito. Una vez que el diseñador introduce el esquema del circuito, las fases de colocación de módulos, generación de las rutas de interconexiones y generación de máscaras pueden realizarse de forma automática mediante una herramienta CAD.

En general, en el diseño con celdas estándar no se desperdicia un área excesiva, como veremos

que ocurre en los próximos estilos de diseño. Esto es debido a que los canales tienen el dimensionado necesario para permitir realizar todas las interconexiones, que se ajusta para cada caso. Además, cada una de las celdas estándar puede diseñarse de forma óptima para minimizar su área. Sin embargo, no es posible conseguir la misma densidad de empaquetamiento que con el diseño full-custom, con lo cual el coste de fabricación del circuito es mayor. Por otra parte, el coste de desarrollo es considerablemente inferior a aquél, y suele utilizarse en aplicaciones donde la demanda no es muy elevada. En ocasiones el diseño con celdas estándar se intercala con macro-celdas.

A.5. Macro-celdas

Las macroceldas son módulos lógicos que no tienen el formato de las celdas estándar, es decir, no tienen una altura predefinida, y normalmente suelen ser bloques de mayor tamaño (ver figura A.4). Suelen estar parametrizados según el número de entradas/salidas.

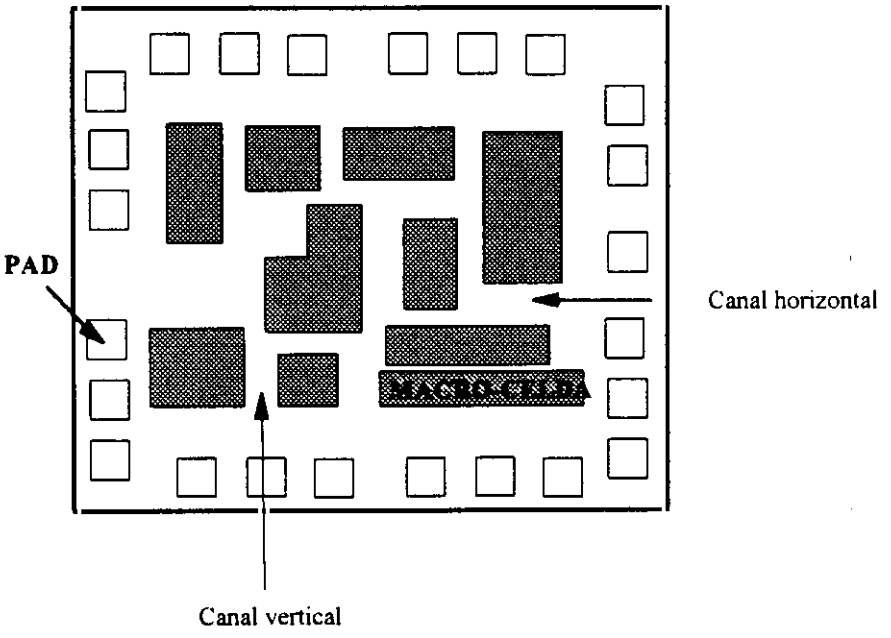


Figura A.4 Diseño con macroceldas

La colocación e interconexionado de las macroceldas en ocasiones se realiza manualmente en

cualquier lugar del circuito, porque la dirección de los terminales de éstas no está estandarizado. Sin embargo, ya existen algunas herramientas CAD capaces de realizarlo de forma automática. Es necesario dejar espacio entre los bloques para realizar las interconexiones. Si el diseño es mixto, una vez colocadas las macroceldas (manualmente o no) puede realizarse automáticamente la colocación de las celdas estándar.

A.6. Arrays de puertas

Consisten en un array rectangular de celdas básicas o puertas lógicas primitivas (tipo NAND) prediseñadas, cada una de las cuales tiene un modelo uniforme de transistores PMOS y NMOS, antes del último paso, previo a la metalización. (ver figura A.5).

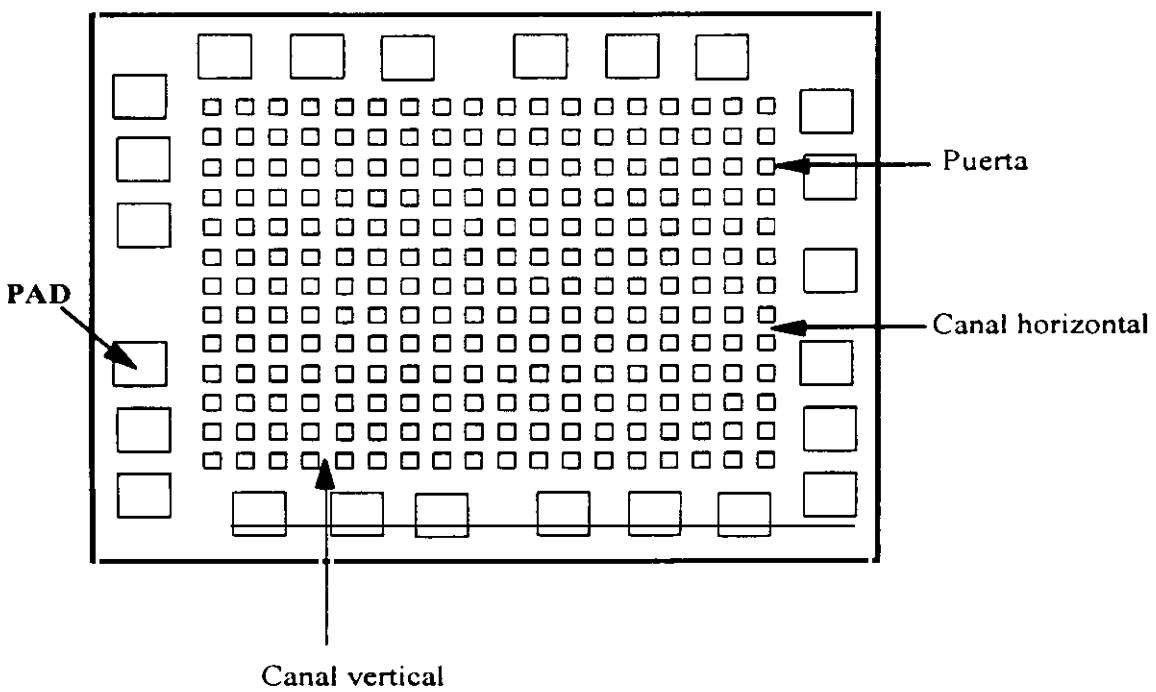


Figura A.5 Array de puertas

Existen canales horizontales y verticales entre las puertas reservados para el interconexionado. Además se pueden definir las interconexiones internas de cada celda básica y así modificar su funcionalidad. Un ejemplo de una posible celda básica con sólo dos transistores PMOS y

NMOS se presenta en la figura A.6.

El diseño del CI consiste en el diseño de las máscaras para las fases de metalización, donde se generan las interconexiones de acuerdo al diagrama del circuito. Estas máscaras de capas seleccionan las interconexiones tanto internas como externas de las puertas. Normalmente las herramientas de CAD disponen de una biblioteca con una serie de módulos con funciones lógicas definidas suministrada por el fabricante del array de puertas. La colocación de celdas y generación de interconexiones puede realizarse automáticamente.

La anchura de los canales es fija, y para asegurar que se pueden realizar todas las interconexiones externas a las puertas, se suele recomendar no superar un cierto porcentaje de celdas básicas utilizadas, dado por el fabricante. Por eso se desperdicia mucha área, y la densidad de empaquetamiento puede llegar a ser la mitad que para celdas estándar.

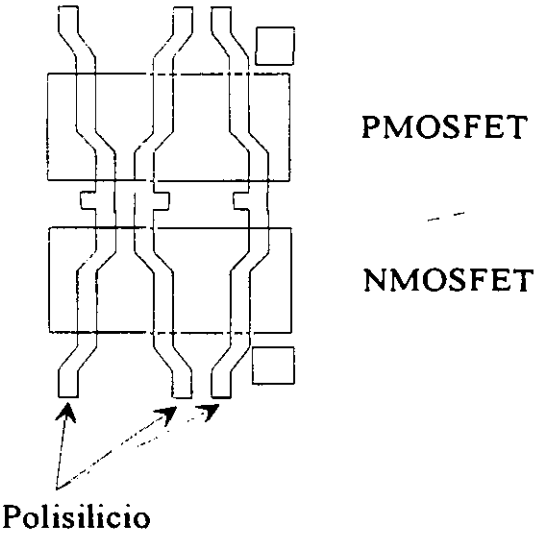


Figura A.6 Estructura de una puerta

Sin embargo, el tiempo de diseño es similar al de aquellas y el coste de fabricación menor, y por tanto la elección de un estilo u otro depende de la demanda del circuito. Si ésta es grande y el tamaño del circuito se puede reducir considerablemente utilizando celdas estándar, entonces esta última elección será la preferida. Si no es así, el diseño con arrays de puertas

dará resultados similares.

A.7. PLA, ROM y LCAs

El diseño de las PLAs y ROMs se obtiene conectando/desconectando las líneas horizontales/verticales de la entrada/salida de los circuitos de dispositivos, como MOSFETs y diodos colocados en un array de dos dimensiones (ver figura A.7). En general los circuitos son sencillos de realizar, pero la densidad de empaquetamiento es muy baja.

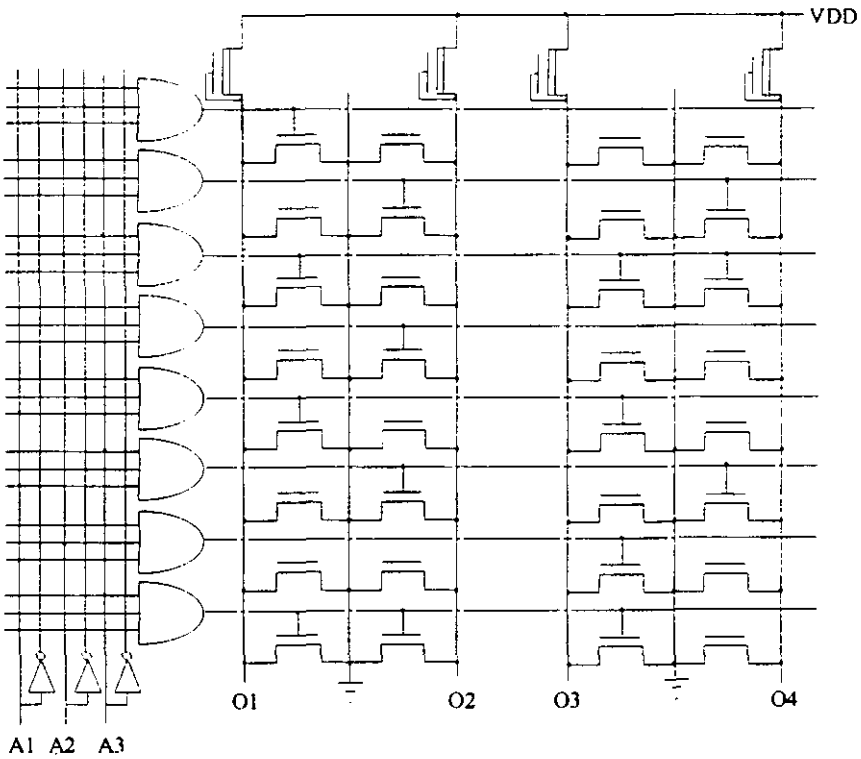


Figura A.7 ROM de MOS

Una posibilidad aún más rápida de diseño la proporcionan las LCAs (Logic Cell Array). Se trata de circuitos que contienen un conjunto de bloques de lógica configurables. Tienen el proceso de fabricación ya finalizado y puede ser personalizados para cumplir la función deseada mediante un fichero de datos que se carga en células de almacenamiento RAM propias de cada LCA. Tanto la definición de la función lógica como sus interconexiones

quedan determinadas por los datos contenidos en dichas células RAM internas.

Los cambios en la lógica se realizan cargando un nuevo fichero en dicha RAM. Existen herramientas CAD que permiten la entrada de las funciones lógicas que deben realizarse con la LCA y trasvasan esta información a un fichero de datos que puede cargarse directamente en la RAM.

El proceso de diseño es por tanto muy rápido, pero la densidad de empaquetamiento muy baja y el funcionamiento más lento que en el resto de los estilos de diseño. Por eso suele utilizarse para sistemas relativamente pequeños con cantidades totales bajas, donde la rapidez de diseño es esencial.

Apéndice B

Colocación e interconexionado de módulos

B.1. Metodología de las Herramientas CAD

El desarrollo de la tecnología VLSI ha dado lugar a la creación de herramientas CAD que sirven para automatizar parte o todo el proceso de diseño del circuito. Cada estilo de diseño (ver apéndice A) tiene sus propios paquetes software para la generación del layout final. Pero en especial los diseños con celdas estándar, arrays de puertas y macroceldas son los que necesitan de un soporte software más complejo para su desarrollo, ya que el tamaño de los circuitos para los que se utilizan suele ser bastante grande y se necesita crear los diseños en un tiempo suficientemente corto.

En general las herramientas CAD para este tipo de estilos de diseño suelen constar de un interfaz gráfico, mediante el cual el diseñador puede introducir el esquema del circuito. También puede introducirse mediante distintos lenguajes de descripción de circuitos, como por ejemplo VHDL ó **EDIF** (Electronic Design Interchange Format), que trataremos más detalladamente en el apartado B.6. En esta fase se dispone además de una biblioteca de módulos de diversa complejidad que sirven al diseñador para facilitar su tarea. Estos módulos pueden haberse diseñado anteriormente (mediante esa misma herramienta CAD y con el estilo de diseño apropiado) o bien pertenecer a la biblioteca original dada por el fabricante de la herramienta. El esquema es un conjunto de módulos y sus interconexiones y no tiene por qué tenerse en cuenta ni el estilo de diseño ni las reglas de layout.

Es la herramienta CAD la que se encarga de generar el layout con un estilo definido a partir de

dicho esquema, teniendo en cuenta un conjunto de reglas de diseño que permite asegurar el buen funcionamiento del circuito. En primer lugar debe haber una fase de transformación del diseño jerárquico, constituido por módulos más o menos complejos, a elementos propios del estilo de diseño, como pueden ser celdas estándar, arrays de puertas o macroceldas.

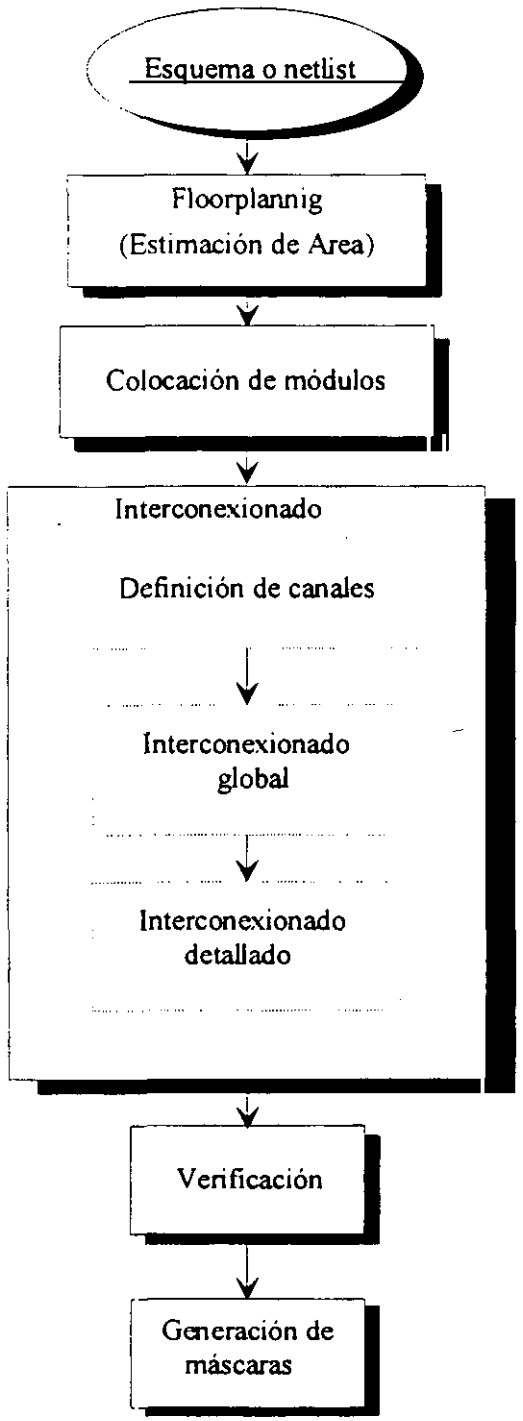


Figura B.1 Esquema general de un proceso de diseño de CI

Un esquema general de un proceso de diseño a partir de este formato se muestra en la figura B.1. Existen tres fases principales de diseño: la generación del floorplanning, la colocación de módulos y el interconexionado. Una vez obtenido el diseño final del circuito, donde se conocen la posición de todos los elementos y sus interconexiones, puede pasarse a la fase de verificación, y si el diseño es correcto se realiza la generación de máscaras.

A continuación vamos a tratar las fases que tienen más influencia en el rendimiento y área del circuito: el floorplanning, la colocación de módulos y el interconexionado. También trataremos brevemente las reglas de diseño, que son muy importantes dentro del proceso de diseño. Por último veremos uno de los lenguajes de descripción de CIs, el lenguaje EDIF, por ser el que hemos utilizado en este trabajo de investigación.

B.2. Floorplanning

En esta primera fase se parte de un conjunto de elementos, que pueden ser macroceldas, celdas estándar o arrays de puertas, según el estilo de diseño, y un conjunto de interconexiones entre ellos. En el caso de los dos últimos la geometría es fija, pero las macroceldas pueden tener distintas formas y la colocación de los terminales en ellas no está predefinida. Además se dispone de estimaciones sobre el retardo y consumo de los distintos elementos, que se puede obtener a través de simulación o bien de la biblioteca. Esta información sirve para dirigir la colocación de módulos, con el fin de que se cumplan las restricciones de tiempo y consumo dadas por el usuario. Por otra parte, el diseñador puede dar restricciones sobre el área total del circuito, algunas posiciones de los elementos y/o sus terminales, y sobre su forma geométrica. A partir de estos datos, el algoritmo de floorplanning debe determinar la geometría, posiciones y formas de todos los elementos, y las posiciones de los terminales que no se conozcan, de forma que las restricciones sean satisfechas y se cumplan los objetivos.

Algunos algoritmos de floorplanning se presentan en [BBCM85] y [OIST86]. Los algoritmos

longitudes de las interconexiones cada vez que se obtiene una colocación determinada de módulos. En el próximo apartado se presentan estas estimaciones.

B.3.1 Técnicas de estimación de la longitud de una interconexión

La mayoría de las herramientas de interconexionado utilizan geometría Manhattan, es decir, sólo se pueden utilizar líneas horizontales y verticales para conectar dos puntos. Además se suelen utilizar dos capas de metal: una para realizar las líneas horizontales y otro para las verticales.

La ruta más corta para conectar un conjunto de terminales es un **árbol de Steiner**. En este método una interconexión puede ramificarse en cualquier punto de su longitud (ver figura B.2).

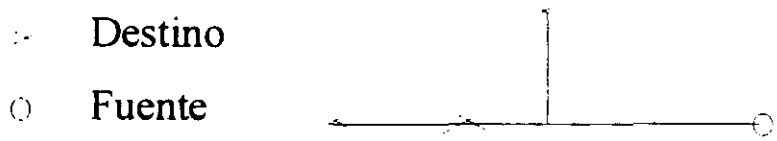


Figura B.2 Árbol de Steiner

La complejidad del cálculo del punto de ramificación que produce una longitud mínima es muy elevada [Chan72],[Chen83], [Hwan79], y por eso este método no suele utilizarse por los algoritmos de interconexionado. En su lugar, el mínimo árbol de "spanning", las conexiones en cadena y el método del semiperímetro son más comunes.

El algoritmo del mínimo **árbol de "spanning"** [Krus56] sólo permite ramificaciones en las posiciones donde existen terminales (ver figura B.3)

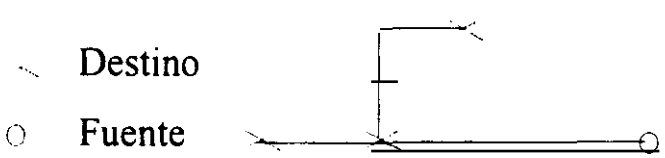


Figura B.3 Mínimo Árbol de spanning

Las **conexiones en cadena** [ver figura B.4] no permiten ramificaciones. Cada terminal se conecta al siguiente según una cadena. En general es muy fácil de implementar pero las longitudes obtenidas son más largas que para los otros casos.

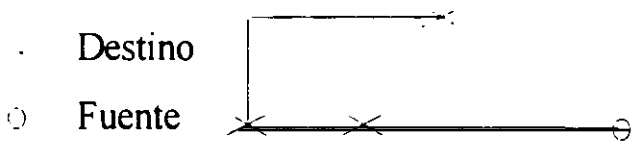


Figura B.4 Conexiones en cadena

En el **método del semiperímetro** la longitud se aproxima por el medio perímetro del mínimo rectángulo que contiene todos los terminales (ver figura B.5). Es el método que más se aproxima a la longitud obtenida por el árbol de Steiner y en general es el más utilizado.

Figura B.5 Método del medio perímetro

B.3.2 Métodos de colocación de módulos

Existe una gran diversidad de métodos para resolver el problema de colocación de módulos, como son la optimización numérica [Wals75], sobre todo para macroceldas, algoritmos genéticos [GGRV85] y [Gref87], simulated-annealing, particionamiento, dirigidos por fuerzas, etc. A continuación vamos a presentar brevemente los algoritmos que son más utilizados por las herramientas de CAD, puesto que se ha hecho mención a ellos a lo largo de este trabajo de investigación.

B.3.2.1 Simulated annealing

Este método es uno de los mejores sistemas para resolver problemas de optimización, como

colocación de módulos [KiGV83], particionamiento [ChRa86], interconexionado [VeKi83], minimización lógica [LaDe86].

Se basa en el proceso de cristalización de los metales. Para conseguir un cristal perfecto se calienta a temperatura muy alta y se enfría lentamente. A alta temperatura, los átomos tienen suficiente energía cinética para saltar de sus posiciones actuales (e incorrectas) a otras nuevas. Cuando se enfría, poco a poco se quedan atrapados en las posiciones correctas.

En el algoritmo de simulated annealing, existe un parámetro que se denomina temperatura y sirve para controlar la probabilidad de aceptación de los movimientos que suponen un incremento del coste. Inicialmente este parámetro toma un valor elevado, con lo cual hay muchas posibles permutaciones (por ejemplo en colocación de módulos, particionamiento, etc.) que son aceptadas. Además todos los movimientos que suponen una disminución del coste se aceptan. Poco a poco se disminuye el valor de dicho parámetro, de forma que cada vez se aceptan menos cambios, y los elementos se quedan en sus lugares adecuados.

En el caso de los algoritmos de colocación de módulos, para evaluar el cambio en el coste, se suele utilizar el método del semiperímetro.

B.3.2.2 Colocación dirigida por fuerzas

Los algoritmos dirigidos por fuerzas, además de para colocación de módulos, también se utilizan en otras aplicaciones, como por ejemplo en particionamiento [HaKu72] y en la planificación temporal de grafos [PaKn87] [PaKn89a].

Se parte de una solución inicial y se asume que todos los elementos conectados se atraen unos a otros con una determinada fuerza. La magnitud de la fuerza entre dos módulos es directamente proporcional a la distancia entre ellos, como en la ley de Hooke para la fuerza de un muelle [figura B.6]. En el caso de colocación de módulos esta distancia vendría dada por la distancia física de los módulos.

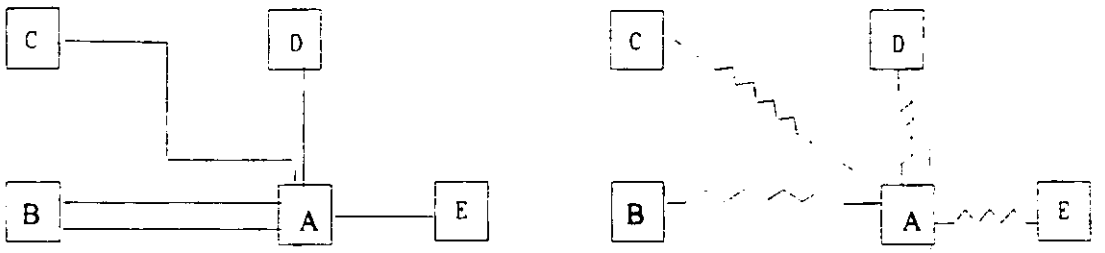


Figura B.6 Algoritmo dirigido por fuerzas

Si se dejara a los módulos libres se moverían en la dirección de la fuerza hasta que el sistema llegara al equilibrio, en un estado de mínima energía, donde la fuerza resultante sería 0. Mediante estas leyes físicas se trata de encontrar la solución óptima.

B.3.2.3 Particionamiento o algoritmo de min-cut

Este tipo de algoritmos suele aplicarse al particionamiento de grafos [KeLi70] [FiMa82] [ScKe72], aunque tiene otras muchas aplicaciones, por ejemplo la colocación de módulos [DuKe85] [SuKe87].

El algoritmo comienza con una solución, obtenida por cualquier otro método. Entonces se realiza repetidamente una división del circuito (o el grafo) en subcircuitos (subgrafos) densamente conectados, de tal forma que el número de redes que cruzan entre particiones (en el caso general serían conexiones entre los elementos del grafo) sea mínimo. Con cada partición del circuito, el área del CI también se divide, y cada subcircuito se asigna a una de estas particiones. Este proceso se repite hasta que cada subcircuito es un único elemento (puerta, celda estándar o macrocelda). Después se recorre de forma ascendente el árbol de particiones y a cada elemento se le asigna una única posición.

B.4. Interconexionado

Los algoritmos de interconexionado se encargan de definir los caminos por donde van a correr las interconexiones. El sistema de partida es un circuito con todos los módulos colocados en

posiciones fijas y una serie de regiones para realizar las interconexiones. Su objetivo principal es conseguir realizar todas las interconexiones en el mínimo área posible.

La primera fase es descomponer la región de interconexionado en regiones rectangulares llamadas **canales**. En el caso de celdas estándar y arrays de celdas estas regiones suelen estar bien definidas, puesto que son los espacios que quedan entre las filas de celdas. Sin embargo para macroceldas pueden ser muy irregulares.

El proceso se divide en dos partes: el interconexionado global y el interconexionado detallado. Además las interconexiones de tierra y alimentación suelen tratarse separadamente, debido a sus características especiales. A continuación tratamos cada una de estas fases separadamente.

B.4.1 Interconexionado global

Durante esta fase a cada red del circuito se le asigna una región particular de las reservadas para interconexionado. Se debe realizar esta operación consiguiendo un 100% de asignaciones de redes a regiones y minimizando el área de interconexionado. El sistema utilizado es diferente dependiendo del estilo de diseño.

Para arrays de puertas [TiTi83] [KLRT83] [AoKu83], y celdas estándar [Supo83], el espacio de interconexionado está principalmente en los canales de interconexionado horizontal. Los terminales de las celdas/puertas se colocan en la parte superior e inferior de éstas. La conexión entre canales puede realizarse por medio de pistas verticales en los lados o en el centro del circuito, o bien mediante celdas *feedthroughs* o "de atravesamiento", que son de la misma forma que el resto de las celdas, pero sólo tienen dentro interconexiones que las atraviesan.

La principal diferencia entre ambos estilos es que el circuito de array de puertas está prefabricado, y los canales tienen una altura fija. Se trata por tanto de asignar un 100% de interconexiones con la restricción del número de pistas. Para celdas estándar la anchura de los canales está sin especificar, y el objetivo principal es realizar la asignación dentro de un área mínima. Esto se consigue minimizando el número de pistas y realizando una asignación

juiciosa de celdas de atravesamiento y de los terminales equivalentes que pertenecen a cada celda. Para estos estilos suelen utilizarse técnicas como simulated annealing (ver apartado B.3), programación lineal, etc.

En el caso de macroceldas [CHKC83], los módulos tienen un tamaño y forma irregular, y las regiones de interconexionado pueden tener formas muy variadas, como rectángulos, formas de L, etc. Estas regiones pueden expandirse o contraerse en función de las necesidades. Se utilizan algoritmos que minimizan la longitud de interconexionado mediante búsqueda del árbol de Steiner.

En general, además de minimizar el número de pistas necesarias, también es deseable minimizar el número de esquinas de las interconexiones. Cada esquina supone el paso de un tramo horizontal (que se realiza en una capa) a un tramo vertical (que se realiza en otra distinta) o viceversa. El paso de una capa a otra supone la utilización de una vía, que es costosa y menos fiable que el resto de la conexión.

B.4.2 Interconexionado detallado

Una vez definidas las regiones por donde se va a realizar cada una de las interconexiones, hay que asignarles un camino específico dentro de cada región. De esta tarea se encarga la fase de interconexionado detallado. Existen diversos algoritmos que tratamos brevemente a continuación.

El algoritmo más conocido para esta tarea es el algoritmo de **recorrido del laberinto** [Ohts86], que es una extensión del algoritmo de Lee [Lee61]. Este método garantiza el encuentro de una solución, si ésta existe. El problema de este algoritmo es que requiere una gran cantidad de memoria y tiempo. Otro algoritmo muy utilizado es el de búsqueda de líneas [High69], que requiere menos memoria y tiempo que el anterior.

Un caso especial de interconexionado detallado es el **interconexionado en canales** (Channel router), en el cual las interconexiones se realizan en una región rectangular sin obstrucciones,

entre terminales colocados en los lados opuestos del rectángulo. Es el caso de celdas estándar y arrays de puertas, donde suelen utilizarse dos capas diferentes, una para tramos horizontales y otra para verticales. Dos segmentos en la misma dirección, deben estar en la misma capa, y por tanto es necesario que no se toquen. Dos segmentos de una misma red en distintas direcciones, que se tocan en un punto determinado, se dice que se conectan por una vía.

Para este tipo de interconexionado existen principalmente dos algoritmos [HaSt71], que garantizan la utilización del mínimo número de pistas posibles:

1.- El **algoritmo del borde izquierdo**. Este tipo de algoritmo también se utiliza para asignación de módulos, planificación, etc. Los segmentos horizontales se ordenan en orden ascendente de abscisas de su borde izquierdo. A continuación se van asignando de uno en uno a las pistas disponibles.

2.- **Algoritmo de empaquetamiento en línea**. Sea I_1 el intervalo con mínimo lado izquierdo. Sea I2 el intervalo con mínimo lado izquierdo mayor que el derecho de I_1. Sea I_k el intervalo con mínimo lado izquierdo mayor que el derecho de I_{k-1}. Los intervalos I_1....I_k pueden colocarse en la misma pista. Este proceso se repite para el resto de los intervalos.

Otros algoritmos de interconexionado de canales se presentan en [Deut76], [Yosh84] y [YoKu82].

B.4.3 Interconexionado de alimentación y tierra

Normalmente debido a la conductancia finita de las interconexiones, al ruido de los contactos y a que este tipo de redes deben conectar muchas celdas, su anchura debe ser variable. Por eso suele haber unos anillos de alimentación y tierra alrededor del CI. Todos los pads deben tener 3 conexiones, una a tierra, otra a alimentación y otra a la celda a la que estén conectados. Además, para las celdas estándar, suelen existir conexiones en la parte superior e inferior de las celdas, de forma que estas redes pueden colocarse en la fila de celdas estándar.

B.5. Reglas de diseño

Las reglas de diseño para cada uno de los estilos se determinan por la *mínima* distancia que puede haber entre cada uno de los dispositivos del circuito. Varían dependiendo de la tecnología de diseño. En general tienen en cuenta los siguientes factores:

- Mínima distancia entre dos capas.
- Mínima distancia de un elemento en una misma capa
- Mínima dimensión de un elemento determinado por restricciones físicas o eléctricas para cada una de las distintas máscaras.

Para conseguir una alta densidad son necesarias muchas reglas que tratan todos los casos posibles por separado (para celdas estándar, macroceldas, arrays de puertas, para cada una de las posibles capas etc.). También deben existir otros valores "por defecto" como, por ejemplo, el valor del ancho de una interconexión. Como este dato es una función de su longitud o del número de elementos que conecta, normalmente se obtiene para distintos rangos de valores.

Otro problema importante es el número de capas para realizar las conexiones entre elementos. En el caso general y como ya hemos dicho anteriormente se utilizan dos capas de metal.

B.6. Lenguaje EDIF

Como consecuencia de la utilización de diversas herramientas para las distintas fases del diseño de circuitos, han surgido problemas de interfaz entre ellas. Por ejemplo, las bibliotecas de celdas, macroceldas, netlists, etc., son utilizadas por varias herramientas y algoritmos, y debe existir la posibilidad de describirlas en algún formato que permita pasarlas de unas herramientas a otras (y de unos algoritmos a otros). De aquí la aparición, entre otros, de EDIF (Electronic Design Interchange Format).

Este lenguaje soporta toda la información que es necesaria para describir un circuito eléctrónico. Su sintaxis está basada en el lenguaje de programación LISP, en el cual todos los datos se representan como expresiones simbólicas. La semántica está dada por las palabras clave. La descripción se expresa en forma jerárquica, abstracta a alto nivel y se convierte progresivamente en más detallada, según se desciende en la jerarquía.

Para mostrar el aspecto general de una descripción es este lenguaje, presentamos el esquema de un circuito. En negrita están las palabras clave.

```
(edif X
  |(external Nombre_Libreria_Externa
    |(cell Nombre_celda
      |(view Nombre_view
        (viewType       /* sólo NETLIST o SCHEMATIC */
        (interface
          |(port Nombre_port
            (direction     /* sólo INOUT, INPUT o OUTPUT*/
  |(library  Nombre_Libreria
    |(cell Nombre_celda
      |(view Nombre_view
        (viewType       /* sólo NETLIST o SCHEMATIC */
        (interface
          |(port Nombre_port
            (direction     /* sólo INOUT, INPUT o OUTPUT*/
        (contents
          |(instance Nombre_instancia
            (viewRef Nombre_view_referencia
              [(cellRef    Nombre_celda_referencia
                [(libraryRef  Nombre_libreria_referencia
          |(net Nombre_red
            (joined
              |(portRef     Nombre_port_referencia
                [(instance Nombre_instancia
                  [(viewRef Nombre_view_referencia
                    [(cellRef  Nombre_celda_referencia
                      [(libraryRef  Nombre_libreria_referencia
  |(design Nombre_diseño_referencia
    (cellRef       Nombre_celda_referencia
      [(libraryRef  Nombre_libreria_referencia
```

Apéndice C

Conversión de FIDIAS a CADENCE

C. 1. De la estructura RTL al layout

A lo largo de este apéndice vamos a presentar el método utilizado para terminar la generación del diseño a partir de la estructura RTL generada por FIDIAS. Esta estructura, que consta de un conjunto de módulos e interconexiones, viene definida en un formato propio de nuestro sistema, que debe ser convertido a un lenguaje de descripción de CIs. El lenguaje elegido fue EDIF (ver Apéndice B), que es aceptado directamente por CADENCE, y así es posible realizar la entrada del diseño a la herramienta de SBN. El diseño generado para CADENCE llega en forma de **netlist**, que es un conjunto de módulos e interconexiones sin información gráfica.

Previamente se ha generado el diseño de todos los módulos utilizados en el circuito. El estilo elegido ha sido celdas estándar, y los módulos se encuentran en una biblioteca de CADENCE y en un formato propio de esta herramienta.

A partir de la netlist comienza la generación del layout. El primer paso, **flattening**, es convertir el diseño jerárquico en un diseño plano, sólo formado por celdas estándar e interconexiones.

Una vez generado este diseño plano, se realiza un floorplanning (ver Apéndice B). Esto lo realiza automáticamente la herramienta cuando a una celda, previamente diseñada como netlist o esquemático, se le otorga la forma denominada **autolayout**.

Los siguientes pasos, como ya vimos en el Apéndice B, son la colocación de módulos, la generación de canales, y los algoritmos de interconexionado global y detallado. Todos estas funciones se ejecutan manualmente, llamándolas una a una desde el sistema de menús que nos presenta la herramienta. Sin embargo, también es posible generarlas automáticamente, como veremos en los próximos apartados.

C. 2. La ruta de Datos

La estructura de salida de FIDIAS, formada por un conjunto de módulos e interconexiones, aparece descrita en un fichero, como el que se muestra en la figura C.1.

Los datos aparecen agrupados en constantes, UFs, buses, registros, registros preasignados (estos últimos bajo el epígrafe *pregistros*), puertos, multiplexores e interconexiones. Cada uno de estos grupos viene precedido de una cabecera, que indica el nombre del grupo, el número de elementos que existen pertenecientes a ese grupo, y una lista de elementos. Para cada elemento existe una serie de campos que lo describen.

```
* Coste Total: 32402
*
* CONSTANTES
    5
* Const.   Valor
    0            1
    1            2
    2            4
    3            6
    4            10
*
* UNIDADES FUNCIONALES
    3
* Num Tipo Entr. Num. Op. Operaciones
    0    10    2    4        AND OR +
* Entr. Conex. Mux.
    1    2    2
    2    4    4
* Num Tipo Entr. Num. Op. Operaciones
    1    18    2    2            + *
```

```
* Entr. Conex. Mux.
    1     2     2
    2     2     2
* Num Tipo Entr. Num. Op. Operaciones
   2     12    2    1              /
* Entr. Conex. Mux.
    1     1     1
    2     1     1
*
*BUSES
    0
Bus n. Conex. Mux
*
* REGISTROS
    3
* Reg n. Conex Mux.
     0    2     2
    1     1     1
    2     1     1
*
* PREGISTROS
    0
*
* PORTS
    0
* Port n. Tipo  Nombre  Conex. Mux.
*
* MULTIPLEXORES
     5
* Mux n. Entr.
    0    2
    1    4
    2    2
    3    2
    4    2
*
* INTERCONEXIONES
    16
* CONS0 - FU0,  e1, c1
   2 6 0   0 0 0 1  1
* CONS1  - FU0, e2, c1
   2 6 1   0 0 0 2  1
* FU0   -  REG0, e1, c1
   0 0 0   2 4 0 1  1
* REG0  -  FU0.  e1, c2
   2 4 0   0 0 0 1  2
* CONS2  -FU0.  e2, c2
   2 6 2   0 0 0 2  2
```

```
* REG0 - FU1. e1, c1
  2 4 0   0 0 1 1 1
* CONS3 -FU1. e2, c1
  2 6 3   0 0 1 2 1
* FU0  -  REG1, e1, c1
  0 0 0   2 4 1 1 1
* REG1 - FU0. e1, c3
  2 4 1   0 0 0 1 3
* CONS4 -FU2. e1, c1
  2 6 4   0 0 2 1 1
* REG1 - FU2 e2, c1
  2 4 1   0 0 2 2 1
* FU2  -  REG2, e1, c2
  0 0 2   2 4 2 1 2
* CONS0 - FU1, e1, c2
  2 6 0   0 0 1 1 2
* REG2 - FU1. e2, c2
  2 4 2   0 0 1 2 2
* REG2 - FU0. e2, c4
  2 4 2   0 0 0 2 4
```

Figura C.1 Ruta de Datos de FIDIAS

Para las constantes existen 2 campos: el identificador que, como para el resto de los tipos de elementos, es un número, y el valor de la constante.

Para describir las UFs existen 5 campos: el identificador, el tipo, que sirve para reconocerla dentro de una biblioteca de módulos, el número de entradas de la UF, el número y las operaciones que realiza. Además, para cada entrada existen dos campos: el número de conexiones que tiene y el tipo de multiplexor que debe utilizarse. Los multiplexores se clasifican según el número de entradas que tengan.

Los buses, registros y registros preasignados tienen 3 campos: el identificador, el número de elementos conectados y el tipo de multiplexores, si tienen alguno.

Los puertos tienen los tres campos anteriores, y, además, el tipo (entrada, salida o entrada/salida) y el nombre.

Los multiplexores tienen dos campos: el número que los identifica y el tipo de multiplexor,

que coincide con el número de entradas.

```
* LISTA DE COSTES DE LOS MULTIPLEXORES
*       n_entr= número de entradas
*       c_mux= coste del multiplexor
*       c_min= coste mínimo del multiplexor (el inicial)
*       c_celdas= coste de celdas estándar del multiplexor
*       n_it_p= número de interconexiones próximas
*       n_it_l= número de interconexiones lejanas
*
* La inicialización se realiza con la subrutina recu_mux a partir
*
*       multiplexor de 1 entrada
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       1       0.0     0       0       0     0
*
*       multiplexor de 2 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       2       41.0    41      40      0       3
*
*       multiplexor de 4 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       4       202.0   202     120     32      10
*
*       multiplexor de 8 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       8       446.0   446     281     96      25
*
*       multiplexor de 16 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       16      1025.0  1025    603     224     56
*
*       multiplexor de 32 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       32      2247.0  2247    1247    480     119
*
*       multiplexor de 64 entradas
*       n_entr c_mux c_min c_celdas n_it_p n_it_l
+       64      4534.0  4534    2534    992     246
```

Figura C.2 Biblioteca de multiplexores de FIDIAS

Para cada interconexión existen dos líneas con toda la información necesaria. En la primera

aparece el tipo y el número del elemento fuente y del destino de la interconexión, el número de entrada de la interconexión (e1 o e2 para UFs y e1 para registros y buses), y la entrada del multiplexor destino si existe. En la segunda línea aparecen 8 números que son la codificación de la información anterior.

Cada uno de los módulos existentes en la ruta de datos ha sido generado previamente por CADENCE, y se encuentra dentro de una biblioteca de módulos propia de la herramienta. Además, las características de estos módulos se describen en varios ficheros accesibles por el sistema FIDIAS, de donde se puede obtener la información sobre las operaciones que realizan cada uno de ellos, el retardo, el número de celdas estándar, las interconexiones lejanas y cercanas, etc. Un ejemplo de este fichero para los multiplexores se encuentra en la figura C.2.

C. 3. Generación del fichero EDIF

La conversión del formato de ruta de datos de FIDIAS a EDIF se realiza de forma automática. Durante la conversión también se añaden al circuito las celdas de alimentación y tierra necesarias. En el fichero EDIF, cuyo formato se ha descrito en el apéndice B, deben aparecer todas las bibliotecas que se van a utilizar para realizar el diseño, como son la biblioteca de las celdas de alimentación y tierra y la biblioteca donde están almacenados todos los módulos que pueden utilizarse. Para cada una de estas bibliotecas, que se denominan externas, deben listarse los módulos o celdas que pertenecen a ellas y aparecen en el circuito. Además, para cada uno de estos módulos o celdas debe describirse su interfaz. Toda esta información es conocida una vez generados los módulos.

Una vez descritas las bibliotecas externas, se nombra la biblioteca donde debe colocarse el diseño que se va a generar, seguido del nombre del diseño. A continuación se describe el interfaz del circuito, listando todos los puertos que contenga, y su contenido. Deben nombrarse todos los módulos del circuito, haciendo referencia a la biblioteca externa donde están generados. Por último se colocan las interconexiones entre los módulos.

Este fichero EDIF se introduce en CADENCE mediante el comando **edifin**. Es necesario utilizar un fichero auxiliar, llamado *edifin.template* (figura C.3).

```
edifInKeys = list(nil
    'searchPath          ". $Path "
    'inFile        ""
    'loadRefLib          "StdLib "
    'techFile            "$FichTech"
    'userSkillFile       ""
    'setSchematicType          "FALSE"
    'caseSensitivity           "preserve"
    'fontStyle           "stick"
)
```

Figura C.3 Fichero *edifin.template*

En este fichero se dan valores a las distintas opciones del comando *edifin*. La variable **Path** contiene todos los caminos donde existen bibliotecas o comandos que son necesarios dentro de CADENCE. La variable **FichTech** contiene el nombre completo (incluido el camino) del fichero de tecnología que se desea utilizar.

El comando *edifin* debe utilizarse con dos argumentos: el fichero auxiliar, *edifin.template*, y el fichero con la descripción en EDIF del circuito, *circuito.edif*.

Con este comando se genera una descripción del circuito, en forma de netlist, en un formato propio de CADENCE. A partir de aquí comienza el proceso de generación del layout utilizando esta herramienta. Pero antes de llamar a este sistema recomendamos borrar el fichero *si.env*, en caso de que existiera en el directorio *./adpFlatten*, porque causa problemas cuando se realiza el *flattening*.

C.4. Generación automática del layout

Para automatizar el proceso de generación del layout vamos a utilizar una herramienta que nos ofrece el sistema CADENCE, que permite la generación de flujos de diseño. Un flujo está formado por una serie de cajas conectadas unas a otras. Cada una de las cajas se corresponde

a un paso de control dentro del flujo de diseño. En cada paso puede llamarse a una o varias herramientas, externas o internas al sistema. Se pueden generar dependencias entre los distintos pasos, de forma que uno no empieza a ejecutarse hasta que no ha terminado el anterior. También es posible ramificar en un punto el diagrama de flujo y generar una condición para seguir un camino u otro.

Nuestro diagrama de flujo debe tener tres pasos: la generación del diseño plano, la realización de la colocación de módulos y el interconexionado (ver figura C.4). El primer paso se realizará automáticamente cuando se inicialice el sistema; una vez terminado este paso se realizará el segundo y después el tercero.

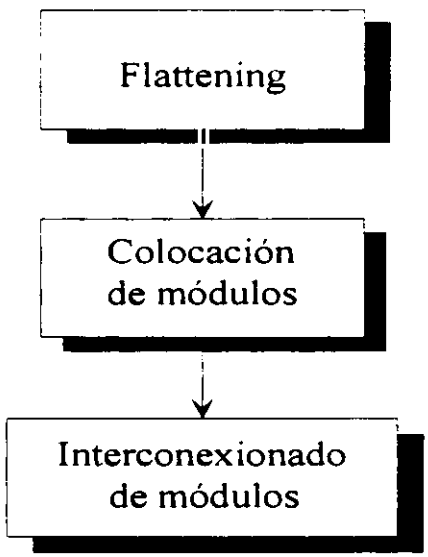


Figura C.4 Diagrama de flujo

De esta forma es posible secuenciar el proceso de diseño global.

En primer lugar, vamos a ver la forma de generar este diagrama de flujo y las dependencias entre los distintos pasos. A continuación, veremos la forma de generar los comandos para realizar las diferentes acciones. Estos comandos se generan en **SKILL** [SKILL93], que es un lenguaje propio de CADENCE.

Tanto los comandos para la generación del diagrama de flujo, como la definición de funciones que se realizan en los pasos de control, se almacenan en ficheros que se cargan automáticamente cuando se llama a CADENCE. Esto se consigue introduciendo comandos de carga de ficheros (*load(NombreFich)*) en el fichero *.cdslocal*.

C.4.1 Generación del diagrama de flujo

En primer lugar hay que definir los pasos de control que van a formar el diagrama de flujo. Como son tres, hay que definir tres pasos de control, a cada uno de los cuales asociamos un nombre y un puntero. Los comandos son:

```
PasoFlattening=dfEncapsulate(?name "Flattening")
PasoColocacion=dfEncapsulate(?name "Colocación")
PasoInterc=dfEncapsulate(?name "Interconexionado")
```

El diagrama de flujo debe almacenarse en una biblioteca, como el resto de los diseños del sistema. Por eso el siguiente paso es abrir esta biblioteca en modo escritura. Puede utilizarse una biblioteca existente o bien crear una nueva. En este caso generamos una biblioteca que denominamos **BiblGenDis** mediante el comando:

```
dfInitLib(?libName "BiblGenDis")
```

Seguidamente se crea el diagrama de flujo en dicha biblioteca. El diagrama lo llamamos "**GenDis**". El comando necesario es:

```
GenDis = dfCreateFlowchart(?name "GenDis" ?libName "BiblGenDis" ?lowerLeft
                           150:150)
```

El último parámetro, **lowerLeft**, es la posición de la esquina inferior izquierda de la ventana donde se quiere que aparezca el diagrama de flujo cuando se abra. Esta posición es relativa a la pantalla y viene dada en coordenadas absolutas.

Una vez creado el diagrama y los pasos de control por separado es necesario relacionarlos. Para esto añadimos los pasos de control al diagrama en unas posiciones determinadas.

dfAddStepToFlowchart(?flowchart GenDis ?step PasoFlattening ?xy 0:1800)
dfAddStepToFlowchart(?flowchart GenDis ?step PasoColocacion ?xy 0:800)
dfAddStepToFlowchart(?flowchart GenDis ?step PasoInterc ?xy 0:100)

A continuación se crean enlaces entre los distintos pasos de control. Hay que enlazar el paso de *flattening* con el de colocación de módulos, y éste último con el de interconexionado. Los comandos son:

dfLinkFlowchartSteps(?flowchart GenDis ?parent PasoFlattening ?child PasoColocacion)
dfLinkFlowchartSteps(?flowchart GenDis ?parent PasoColocacion ?child PasoInterc)

El siguiente paso es generar las dependencias para los enlaces creados. Queremos que los pasos segundo y tercero no se inicien hasta que no hayan terminado todos los anteriores. Los comandos son:

dfSetDependency(?step PasoColocacion ?flowchart GenDis ?dependency 'allParents)
dfSetDependency(?step PasoInterc ?flowchart GenDis ?dependency 'allParents)

Con estos comandos ya se ha creado el diagrama de flujo. Ahora interesa que este diagrama se abra automáticamente cuando comienza el sistema de CADENCE. Para esto hay que generar una instancia de este diagrama, que llamamos **GenDisInst**.

dfDisplayFlowchart(?flowchart GenDis ?libName "BiblGenDis" ?cellName "GenDisInst")

Si el diseño no se había abierto anteriormente, se crea una celda con este nombre y se muestra en la ventana. Una vez abierto el diagrama, el primer paso comienza a ejecutarse automáticamente.

Una vez generado el esqueleto del diagrama de flujo, hay que añadir la funcionalidad a cada paso de control. La ventaja de utilizar este método es que la dirección de flujo y las dependencias entre los distintos pasos se verifican automáticamente, con lo cual se asegura la correcta secuencialidad del proceso. La funcionalidad se añade asociando a cada paso de control una función mediante los comandos:

PasoFlattening->function='funcFlattening
PasoColocacion->function='funcColocacion
PasoInterc->function='funcInterconexionado

Todas las funciones definidas por el usuario deben utilizar el argumento **stepInst**. Este argumento se utiliza para pasar el puntero del paso de control a la función. La función se invoca automáticamente cuando se llega al paso de control asociado. La definición de estas funciones se realiza en el subapartado C.4.2. Deben usarse siempre comandos *SKILL*.

C.4.2 Definición de la función para el paso *Flattening*

Supongamos que el diseño que vamos a generar se llama **MiDiseño** y la biblioteca donde los almacenamos **MiBiblioteca**.

El comando *SKILL* que realiza el *flattening* es *adpFlatten()*. Sin embargo antes de llamarlo se necesitan realizar algunas operaciones, como abrir la biblioteca donde se encuentra el diseño en forma de netlist y borrar posibles versiones antiguas del diseño en forma *autoLayout*.

Además surge un nuevo problema. Dentro de una función, los comandos se van ejecutando de uno en uno, en el orden en que aparecen. Pero cuando un comando necesita que se abran menús para introducir datos, no espera a que se terminen de abrir los menús para ejecutar el siguiente comando. Como el siguiente comando suele ser rellenar las distintas opciones del menú, si éste no existe se produce un error. Por eso, para este tipo de comandos, se necesita introducir un retardo que sea suficientemente grande para que puedan abrirse los menús asociados, antes de pasar al comando siguiente.

Para esto vamos a utilizar la variable **delay_form**, que inicializamos con un número muy grande. Entonces se genera un bucle del cual se sale, bien cuando el menú ya se ha abierto, o bien cuando el bucle se ha ejecutado un número de veces igual a *delay_form*. En este último caso hay que dar un mensaje de error, puesto que no se han abierto los menús correctamente. Por ejemplo, para el caso del comando *adpFlatten*, se abre un menú llamado *adpFlattenForm*. El retardo en este caso se generaría mediante los siguientes comandos:

```
cont = delay_form
while( ( symeval( 'adpFlattenForm ) == 'unbound )
  cont--
  if( cont == 0 then
    printf( " ERROR: PRflatten (form)" )
    return( nil )))
```

Esta misma estructura para generar un retardo se utilizará para el resto de las funciones en las que haya que esperar a que se abran menús. Para no repetir código simplemente la denominaremos retardo. Cada vez que se llame a un comando que tenga asociado un menú se creará una función (en *SKILL* se denomina *prog*) que generará el retardo y luego rellenará los valores de las distintas opciones del menú. Todos los comandos dentro de un procedimiento deben incluirse dentro de uno o varias funciones. En la figura C.5 se muestran los comandos para el *flattening*.

```
procedure(funcFlattening(@key (stepInst nil))
prog(()
                ; Carga los nombres del diseño y de la biblioteca
biblioteca = "MiBiblioteca"
diseño = "MiDiseño"
                ; Inicializa la variable delay_form
delay_form = 100000
                ; Abre la biblioteca donde está el diseño
biblioteca_cv = dmOpenLib( biblioteca "" "" "a" )
                ;Se borra el autoLayout del diseño en caso de que hubiera uno anteriormente
generado.
 dmDeleteCellView( biblioteca_cv diseño "autoLayout" )
                ; Se llama al comando que genera el flattening
 adpFlatten()
prog((cont)
retardo(cont)
        hiiSetCurrentForm('adpFlattenForm)
        adpFlattenForm->adpFlattenLibName->value = biblioteca
        adpFlattenForm->adpFlattenCellName->value = diseño
        adpFlattenForm->adpFlattenViewName->value = "netlist"
        adpFlattenForm->adpFlattenVersionName->value = ""
         hiFormDone(adpFlattenForm)
)
return(t)))
```

Figura C.5 Comandos para realizar el flattening

Cerrando un menú (mediante *hiFormDone()*), el comando asociado a dicho menú empieza a ejecutarse, y hasta que no termina no se pasa al comando siguiente. Así se asegura que la colocación de módulos, que es el siguiente paso, comience a ejecutarse cuando ha terminado de realizar el *flattening*.

C.4.3 Definición de la función para el paso de colocación de módulos

La colocación de módulos es un proceso complejo que se realiza mediante sucesivos comandos, que colocan los puertos del diseño, generan las celdas *feedthroughs*, las celdas de las esquinas para distribuir las redes de alimentación y tierra, etc. Por eso, los comandos *SKILL* asociados a cada una de estas funciones los hemos almacenado en ficheros distintos, que se van cargando en el orden adecuado, mediante el comando *load(NombreFichero)*. En la figura C.6 se presentan estos comandos.

A continuación vamos a tratar cada uno de estos pasos por separado.

C.4.3.1 . Comandos de definición de prioridades de las redes

Se encuentran almacenados en el fichero *netprior.ldi* (figura C.7). Su función es leer un fichero (*netData*) donde se almacena la información sobre las prioridades de las distintas redes del circuito.

C.4.3.2. Inicialización del algoritmo de colocación de módulos

Los comandos se hallan en el fichero *inicIO.ldi*. El objetivo de esta fase es iniciar la colocación de los puertos de entrada/salida. Se puede utilizar un fichero para colocar algunos de estos módulos, si bien nosotros no hemos utilizado ninguno, por eso esta opción del menú la colocamos a *nil*. Los comandos se muestran en la figura C.8.

```
procedure(funcColocacion(@key (stepInst nil))
prog((maxScreen result ck ck_feed resp)
                ; se generan las coordenadas de la ventana donde se abrirá el diseño
if( ! boundp('windowLocation) || windowLocation == nil
        then
        maxScreen = hiGetMaxScreenCoords()
          windowLocation = list(0:0 list(car(maxScreen)/2 cadr(maxScreen)/2)))
if( ! boundp('windowScrollBars) || windowScrollBars == nil
then  windowScrollBars = t)
                ; creacion de la ventana
winp = hiCreateWindow(windowLocation "graphics" "autolayout" "" windowScrollBars)
                ; apertura del diseño
dmOpenLib(biblioteca "" "" "a")
result = deOpenCellview(biblioteca disenno "autoLayout" "" winp "append")
if(!result then  return(nil))
ck = dbOpenCellView(biblioteca disenno "autoLayout" "" "a" nil)
                ; Instalación de la herramienta de colocación e interconexionado
deInstallApp(getCurrentWindow() "Cell Ensemble ")
load(" netprior.ldi")
load(" inicIO.ldi")
icEnvMakeForm0- deselectBy- value="All"
let( ((form icEnvMakeForm0) (field 'allnone)) adpCmdFilterNoneCB(form) )
icEnvMakeForm0- region- region- value=t
icEnvMakeForm0- selectBy- value="All"
load(" snap.ldi")
load(" IOPlace.ldi")
load(" feed.ldi")
load(" corners.ldi")
load(" justify.ldi")
icEnvMakeForm0- region- region- value=t
icEnvMakeForm0- selectBy- value = "All"
load(" leftcap.ldi")
auiGlueAddForm = 'unbound
auiGlueCapCellSubForm = 'unbound
load(" rightcap.ldi")
load(" savep.ldi")
return(t)))
```

Figura C.6 Comandos para realizar la colocación de módulos

```
auiNetPropertiesFileMC()
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiNetsPropertiesFileForm)
auiNetsPropertiesFileForm->auiActionFD->value = "read"
auiNetsPropertiesFileForm->auiNetPropWhichNetsFD->value = "selected"
auiNetsPropertiesFileForm->auiPropertiesFileFD->value= "$path/netData"
auiNetsPropertiesFileForm->auiWhichPropertiesFD->value="P&R"
hiFormDone(auiNetsPropertiesFileForm)
return( t )
)
```

Figura C.7 Comandos para definir las prioridades de las redes

```
adpfpInitFPMC()
prog((cont)
retardo(cont)
hiiSetCurrentForm('adpfpInitFPCeBeForm3)
adpfpInitFPCeBeForm3->fpInitChoice->all->value = t
adpfpInitFPCeBeForm3->fpInitPreserve->IO->value = nil
adpfpInitFPCeBeForm3->fpInitPreserve->macro->value = nil
adpfpInitFPCeBeForm3->fpInitPreserve->stdCell->value = nil
adpfpInitFPCeBeForm3->fpInitFPEstSize->value = t
adpfpInitFPCeBeForm3->fpInitFPAR->value = 1
adpfpInitFPCeBeForm3->fpInitFPLoadPFile->value = nil
adpfpInitFPCeBeForm3->fpInitFPLoadSDF->value = nil
hiFormDone(adpfpInitFPCeBeForm3)
return( t ))
```

Figura C.8 Fichero *inicIO.ldi*

C.4.3.3. Definición de la malla para realizar la colocación

Los comandos se encuentran en el fichero *snap.ldi* (figura C.9). El objetivo es definir la malla sobre la que se colocan las celdas e interconexiones y que todas las celdas se coloquen correctamente sobre esta malla. Para la tecnología de *ES2* de 1.0 micras el valor del ancho de la malla es de 0.125.

C.4.3.9. Salvar el diseño

Una vez terminada la colocación de módulos salvamos el diseño con el nombre de "*placed*". Los comandos se encuentran en el fichero *savep.ldi* (Figura C.15).

```
prog(()
dmDeleteCellView( biblioteca_cv disenno "placed" )
icSaveAsMC(winp))
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiSaveDesignForm)
auiSaveDesignForm- auiMgrViewName- value = "placed"
hiFormDone(auiSaveDesignForm)
return( t ))
```

Figura C.15 Fichero *savep.ldi*

C.4.4 Definición de la función para el paso de interconexionado

La realización del interconexionado tiene lugar en varias fases. En primer lugar hay que borrar todas las regiones generadas para la colocación de módulos. A continuación vienen las fases de generación de canales, interconexionado global e interconexionado detallado. Por último se salva el diseño y la definición de su tecnología. Todas estas fases se realizan cargando distintos ficheros que tienen los comandos asociados a cada una de ellas. En la figura C.16 se muestra el conjunto de comandos para realizar el interconexionado. A continuación veremos cada una de las etapas por separado.

C.4.4.1. Generación de canales

Los comandos para generar los canales se encuentran en el fichero *genchan.ldi*, que se muestra en la figura C.17.

```
procedure(funcInterconexionado(@key (stepInst nil))
prog(()
                ;borrar regiones de placement
let( ((form icEnvMakeForm0) (field 'allnone)) adpCmdFilterAllCB(form) )
icEnvMakeForm0- deselectBy- value="All"
let( ((form icEnvMakeForm0) (field 'allnone)) adpCmdFilterNoneCB(form) )
icEnvMakeForm0- region- region- value=t
icEnvMakeForm0- selectBy- value="All"
let( ((form icEnvMakeForm0) (field 'delete)) adpCmdDeleteCB(form field) )
load( " genchan.ldi" )
load( " globalh.ldi" )
load( " detailedh.ldi" )
load(" saver.ldi")
load(" savetech.ldi")
load(" quit.ldi")
return(t)))
```

Figura C.16 Comandos para realizar el interconexionado

```
auiChnGenChannels()
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiChnGenChanelsForm)
auiChnGenChanelsForm- auiMainTreeDir- value = "automatic"
auiChnGenChanelsForm- auiChannelsBlockNamePrefix- value = "Channels"
hiFormDone(auiChnGenChanelsForm)
return( t ))
```

Figura C.17 Fichero *genchan.ldi*

C.4.4.2. Interconexionado global

Los comandos para generar esta etapa se encuentran en el fichero *globalh.ldi*, que se muestra en la figura C.18.

C.4.4.3. Interconexionado detallado

La última etapa del interconexionado se realiza mediante los comandos del fichero *detailedh.ldi*, que se presenta en la figura C.19.

```
auiGlrAutoGlobalRouteMC()
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiGlrAutoGlobalRouteForm)
auiGlrAutoGlobalRouteForm- auiGlrMethod-> value = "both"
hiFormDone(auiGlrAutoGlobalRouteForm)
return( t ))
```

Figura C.18 Fichero *globalh.ldi*

```
auiDtrRtCompact()
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiDtrRtCompactForm)
auiDtrRtCompactForm- auiDtrChanCompactionMode- value = "automatic"
auiDtrRtCompactForm- auiDtrCenterContacts- value = "offCentered"
auiDtrRtCompactForm- auiConditionalVia?- boolean- value = nil
auiDtrRtCompactForm- auiRouteTLMoption- value = "2 or 2 1/2 Layers"
hiFormDone(auiDtrRtCompactForm)
return( t ))
```

Figura C.19 Fichero *detailedh.ldi*

C.4.4.4. Salvar el diseño

La última fase dentro de este proceso es salvar el diseño y la tecnología. El diseño lo salvamos como *routed*. Los comandos para estas dos fases se encuentran en los ficheros *saver.ldi* y *savetech.ldi*, que se muestran en las figuras C.20 y C.21.

```
prog(()
dmDeleteCellView( biblioteca_cv disenno "routed" )
icSaveAsMC(winp))
prog((cont)
retardo(cont)
hiiSetCurrentForm('auiSaveDesignForm)
auiSaveDesignForm- auiMgrViewName- value = "routed"
hiFormDone(auiSaveDesignForm)
return( t ))
```

Figura C.20 Fichero *saver.ldi*

```
tcDisplaySaveForm()
prog((cont)
retardo(cont)
hiiSetCurrentForm('tcSaveTechForm)
tcSaveTechForm-··tcLibNameField-··value = biblioteca
hiFormDone(tcSaveTechForm)
return( t ))
prog((cont)
retardo(cont)
hiDBoxCancel(tcSaveDBox)
return( t ))
```

Figura C.21 Fichero *savetech.ldi*

Para terminar el proceso hace falta salir de CADENCE y cerrar todas las ventanas. Los comandos para realizar esta fase se encuentran en el fichero *quit.ldi* (figura C.22).

```
exit()
prog((cont)
retardo(cont)
hiiSetCurrentForm('geSaveAllForm)
hiFormDone(geSaveAllForm)
return( t ))
```

Figura C.22 Fichero *quit.ldi*

BIBLIOGRAFÍA

[AoKu83] K. Aoshima, E. S. Kuh, "Multi-channel optimization in gate-array layout" Proc. International Symposium of Circuits and Systems, IEEE ISCAS 83, pp. 1000-1008, 1983

[BaMa89] M. Balakrishnan, P. Marwedel "Integrated Scheduling and Binding: A Synthesis approach for design space exploration", 26th ACM/IEEE Design Automation Conference pp. 68-74, 1989

[BBCM85] R. K.Brayton, N. L. Brenner, C. L. Chen, G. De Micheli, C. T. McMullen, R. H. J. M. Otten"The Yorktown Silicon Compiler" Proc. International Symposium of Circuits and Systems, ISCAS 85, pp. 391-394, June 1985, Kyoto

[BeCP92] R.A. Bergamaschi, R. Camposano, M. Payer "Allocation algorithms based on path analysis" *INTEGRATION, the VLSI journal* 13, pp. 283-299, 1992

[BhDB94] S. Bhattachrya, S. Dey, F. Brglez, "Clock Period Optimization During Resource Sharing and Assigment", Proc. Design Automation Conference, DAC-94, San Diego, pp. 195-200, 1994

[Brew88] F. D. Brewer "Constraint Driven Behavioral Synthesis", Tesis Doctoral, University of Illinois at Urbana-Champaign, 1988

[BrGa87] F.D.Breuer, D. Gajski, "Knowledge Based Control in Micro-Architecture Design", 24th Design Automation Conference, pp. 203-209,1987

[Camp88] R. Camposano "Design Process Model in the Yorktown Silicon Compiler", in Proc. 25th Design Automation Conference, ACM/IEEE California, pp. 489-494, June 1988

[Camp91] R. Camposano,"Path-Based Scheduling for Synthesis", IEEE Transactions on Computer-Aided Design, vol. 10, n°1, pp. 85-93, Enero 1991

[CBHP91] R. Camposano, R.A. Bergamaschi, C.E. Haynes, M. Payer, S.M. Wu, "The IBM High-Level Synthesis System", en "High Level VLSI Synthesis", Kluwer Academic Publisher, pp. 79-104, 1991

[Chan72] S. Chang "The generation of minimal trees with a Steiner topology" ACM Computing Surveys, vol. 19, n°4, pp. 699-711, Octubre 1972

[ChBu88] Chen, X., Bushnell, M.L. "A module area estimator for VLSI layout", in Proc. 25th Design Automation Confer., pp. 54-59, June 1988

[Chen83] N.P. Chen "New algorithms for Steiner tree on graphs" Proc. International Symposium on Circuits and Systems" ISCAS 1983, pp. 1217-1219, 1983

[CHKC83] N.P. Chen, C. P. Hsu, E. S. Kuh, C. C. Chen, M. Takahashi "BBL: A building block layout system for custom chip design" Proc. IEEE International Conference on Computer Aided Design, ICCAD 83, pp. 40-41, 1983

[ChRa86] M. J. Ching, K. K. Rao, "Pararell simulated annealing for partitioning and routing" Proc. IEEE International Conference on Computer Design, pp. 238-242, 1986

[DCGM90] H. de Man, F. Catthoor, G. Goossens, J. Van Meerbergen, J Rabaey, J. Huisken "Architecture-driven synthesis techniques for mapping digital signal processing algorithms into silicon", special issue on Proc. IEEE Computer-Aided Design, vol 78 n° 2, pp. 319-335, Febrero 1990.

[DeNe89] S. Devadas, Newton. "Algortihms for Hardware Allocation in Data Path Synthesis" IEEE Trans on Computer-Aided Design, vol8 n° 7, pp. 768-781, Julio 1989

[Deut76] D. N. Deutsch "A dogleg channel router" Proc. 13th Design Automation Conference, DAC 76, pp. 425-433, 1976

[Dona80] W.E. Donath "Complexity theory and design automation" Proc. 17th Design Automation Conference, DAC 80, pp. 412-419, 1980

[DuKe85] A. E. Dunlop, B. W. Kernigham "A procedure for placement of standard cell VLSI circuits" IEEE Trans. on Computer Aided Design, CAD-9, pp. 92-98, January 1985

[ES2L93] ES2 ECPD10 Library Databook, 1993

[FiMa82], C. M. Fiduccia, R. M. Mattheyses "A linear-time heuristic for improving network partitions" Proc. 19th Design Automation Conference, DAC 82, pp. 175-181, 1982

[Frie87] B. Friedland "Control System Design: An Introduction to Space Methods" McGraw Hill International Editions, 1987

[GaKu83]D. Gajski, R. Kuhn "New VLSI Tools" IEEE Computer, vol. 16, nº 12, pp. 11-14, Diciembre 1983

[GaRa94] D. Gajski, L. Ramachandran "Introduction to High-Level Synthesis" IEEE Design&Test of Computers, pp. 44-54, 1994

[GiBK85] E. F. Girczyc, R. J. Buhr, J. P. Knight "Applicability of a Subset of Ada as an Algorithmic Hardware Description Language for Graph-Based Hardware Compilation", IEEE Trans. on Computer-Aided Design, vol. CAD-4, nº 2, pp. 134-142, Abril 1985

[GGRV85] J. J. Grefenstetter, R. Gopal, B. J. Romaita, D. Van Gucht, "Genetic Algorithms for the Traveling Salesman Problem" Proc. of First International Conference on Genetic Algorithms and their Applications, 1985

[Gref87] J. J. Grefenstetter "Incorporating problem specific knowledge into genetic algorithms" in "Genetic Algorithms and Simulated Annealing", cap. 4, pp. 42, 1987

[GuAb89] C. V. Gura, J. A. Abraham "Average Interconnection Length and Interconnection Distribution Based on Rent's Rule", 26th ACM/IEEE Design Automation Conference, pp. 574-577, 1989.

[HaCC92] T. Hamada, C. Cheng, P. M. Chau, "A Wire Length Estimation Technique Utilizing Neighborhood Density Equations" 29th ACM/IEEE Design Automation Conference, pp. 57-61,1992

[HaKu72] M. Hanan, J.M. Kurkzberg "Placement techniques In Design Automation of Digital Systems" M. A. Breuer, Ed. Prentice Hall, Englewood Cliffs, N. I., Cap. 5, pp. 213-282, 1972

[HaSt71] A. Hashimoto, J. Stevens "Wire Routing by Optimizing Channel Assigment within Large Apertures" Proc. 8th Design Automation Workshop, pp. 155-169, 1971

[High69] D. W. HighTower "A solution to Line-Routing Problem on the Continous Plane" Proc. 6th Design Automation Workshop, pp. 1-24,1969

[HLSB91] High Level Synthesis Benchmarks, 1991

[Hwan79] F.K. Hwang "An O(NLogN) algorithm for suboptimal rectilinear Steiner trees". IEEE Transactions on Circuits and Systems, vol 26, n°1, pp. 75-77, 1979

[JaPP87] R. Jain, A. Parker, N. Park "Predicting Area-Time Tradeoffs for Pipelined Designs" Proc. 24th Design Automation Conference, DAC-87, pp. 35-41, 1987

[JaPP92] R. Jain, A. Parker, N. Park "Predicting System-Level Area and Delay for Pipelined and Nonpipelined Designs" IEEE Transactions on Computer-Aided Design, Vol. 11, n°8, pp. 955-965, Agosto 1992

[JKMP89] R. Jain, K.Kucukcakar, M.J.Mlinar, A. Parker "Experiences with the ADAM System" Proc. 26th Design Automation Conference, DAC-1989, pp. 55-61, New York 1989

[JMPP88] R. Jain, M.J.Mlinar, A. Parker, N. Park "Area Time Model for Synthesis of Non-pipelined Designs" Proc. IEEE International Conference on Computer-Aided Design, ICCAD-88, pp. 48-51, Nov. 1988

[JRDK94] P.K.Jha, C. Ramachandran, N.D. Dutt, F. J. Kurdahi "An Empirical Study on the Effects of Physical Design in High Level Synthesis" 7th International Conference on VLSI Design, pp. 11-16, Enero 1994

[KeLi70] B.W.Kerniighan, S. Lin "An efficient heuristic for partitioning graphs" Bell Systems Technology, vol. 49, n° 2, pp. 291-308, 1970

[KGWF85] T.J. Kowalski, D. J. Geiger, W. H. Wolf, W. Fichtner, "The VLSI Design Automation Assistant: From Algorithms to Silicon", IEEE Design and Test, Agosto 1985

[KiGV83] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi "Optimization by simulated annealing" Science 220, 4598, pp. 671-680, Mayo 1983

[KLRT83] R. M. Karp, R. L. Leighton, R. Rivest, C. D. Thompson, U. Vaizirani, V. Vazirani "Global wire routing in two dimensional arrays" Ann. Symposium on Foundations of Computers Science, vol 24, pp. 453-459, 1983

[Kowa85] T.J. Kowalski "An Artificial Intelligence Approach to VLSI Design" Boston, MA: Kluwer Academic, 1985

[KuMi90a] D. C. Ku, G. De Micheli "High Level Synthesis and Optimization Strategies in Hercules and Hebe" Proc. European ASIC Conference, pp. 124-129, 1990

[KuMi90b] D. C. Ku, G. De Micheli "HardwareC - A lenguage for hardware design (version 2.0)". Stanford University CSL Technical Report, CSL-TR-90-419, April 1990

[KuMi90c] D. C. Ku, G. De Micheli "Control Optimization based on resynchronization of operations" Proc. Design Automation Conference, pp. 59-64, June 1990

[Kuo87] B. C. Kuo "Automatic Control Systems" Prentice-Hall International Editions, 1987

[KuPa87] Kurdahi,F.J. and Parker,A.L. "REAL: A Program for REgister ALlocation". In Proc. ACM/IEEE 24rd Design Automation Conference, pp. 210-215, June 87

[KuPa89] Kurdahi,F.J. and Parker,A.L."Technique for area estimation of VLSI layouts", in IEEE Transactions on Computer-Aided Design, vol. CAD 8, n° 1, pp. 81-92, Jan 1989

[KuPa90a] K.Kucukcakar, A. Parker. "Data Path Tradeoffs using MABAL" Proc. 27th ACM/IEEE Design Automation Conference, pp. 511-516, 1990

[KuPa90b] K.Kucukcakar, A. Parker. "BAD: Behavioral Area Delay Prediction". Technical Report CENG-90-31, Dep. Electrical Engineering, University of Southern California, November 1990

[KuPa91] K.Kucukcakar, A. Parker. "CHOP: A Constraint-Driven System-Level Partitioner". Proc. 28th ACM/IEEE Design Automation Conference, pp. 514-519, 1991

[KuRa93] F.J. Kurdahi, C. Ramachandran "Evaluating Layout Area Tradeoffs for High Level Applications", in IEEE Trans. on VLSI Systems, vol. 1, n° 1, pp. 46-55, March 1993

[Krus56] J. Kruskal, "On the shortest spanning subtree of a graph and the travelling salesman problem" In Proc. of the American Mathematical Society, vol. 7, n°1, pp. 48-50, 1956

[LaDe86] J. Lam, J. Delome "Logic minimization using simulated annealing" Proc. IEEE International Conference on Computer Aided Design, pp. 378, 1986

[LaRu70] B. Landman, R. Russo, "On a Pin Versus Block Relationship For Partitions of Logic Graphs". IEEE Trans. on Computers, vol c-20, n° 12, Dec. 1970

[Lee61] C. Y. Lee "An Algorithm for Path Connections and its Applications" IRE Trans. on Electronics Computer vol EC-10, pp. 346-365, 1961

[Leig83] F.T.Leighton "Complexity Issues in VLSI" MIT Press, Cambridge, 1983

[LyEG90] T.A.Ly, W.L.Elwood, E.F. Girczyc "A Generalized Interconnect Model for Data Path Synthesis" Proc. 27th ACM/IEEE Design Automation Conference, pp. 168-173, 1990

[MATL94] "*High Performance Numeric Computation and Visualization Software*", Matlab ver 4.2, Ed. The MathWorks, Inc, 1994

[McFa83] M.C. McFarland "Computer-Aided Partitioning of Behavioral Hardware Descriptions" Proc. 20th Design Automation Conference, pp. 472-478, 1983

[McFa87] M.C. McFarland "Reevaluating the Design Space for Register-Transfer Hardware Synthesis" International Conference on Computer-Aided Design, pp. 262-265, IEEE, Nov. 1987

[McFa89] M.C. McFarland "A Fast Floor Planning for Architectural Evaluation" International Conference on Computer-Aided Design, pp. 96-99, IEEE, Nov. 1989

[McKo86] M.C. McFarland, T.J. Kowalski "Assisting DAA: The Use of Global Analisys in an Expert System". International Conference on Computer Aided Design, IEEE, pp. 482-485, Octubre 1986

[McKo90] M.C. McFarland, T.J. Kowalski "Incorporating Bottom-Up Design into Hardware Synthesis". IEEE Transactions on Computer Aided Design. Vol. 9, n°9, pp.938-950. Sep 1990.

[McPC88] M.C. McFarland, A.C. Parker, R. Camposano, "Tutorial on High Level Synthesis", Proc. 25th ACM/IEEE Design Automation Conference, pp. 330-336, 1988

[Mend93] J. M. Mendias Cuadros "Implementación de Sistemas Digitales mediante la *interacción entre herramientas de Síntesis de Alto Nivel y lógica*", *Trabajo de tercer ciclo*, Departamento de Informática y Automática, 1993

[MFHM94] H. Mecha, M. Fernández, R. Hermida, D. Mozos, K. Olcoz "Clock Cycle Estimation Based on Dead Time and Control Unit Area Minimization" Microprocessing and

Microprogramming, The Euromicro Journal, vol. 40, Ed. North-Holland, pp. 821-824, Septiembre 1994, Holanda

[MFSH94] H. Mecha, M. Fernández, J. Septién, R. Hermida, R. Moreno "Un Método de Estimación de Área en Síntesis de Alto Nivel" Actas del IX Congreso de Diseño de Circuitos Integrados. Canarias, pp. 89-94, Noviembre 1994.

[MFSM96] H. Mecha, M. Fernández, J. Septién, D. Mozos, F. Tirado, K. Olcoz "A Method for Area Estimation of Data-path in High Level Synthesis" IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 15, nº 2, Febrero 1996

[MKMT90] G. De Micheli, D. C. Ku, F. Mailhot, T. Truong, "The Olympus Synthesis System for digital design", IEEE Design and Test Magazine, pp. 37-53, Octubre 1990

[More95] R. Moreno Vozmediano "Integración de Planificación Temporal y Asignación de Hardware", Tesis Doctoral, Universidad Complutense de Madrid, Octubre 1995

[Mozo92] D. Mozos Muñoz "Un Sistema de Control Inteligente para Síntesis de Alto Nivel", Tesis Doctoral, Universidad Complutense de Madrid, 1992

[MSHO92] H. Mecha, J. Septién, R. Hermida, K. Olcoz "Influencias de factores del Nivel Físico en la Estimación de Area y Tiempo en Síntesis de Alto Nivel" Actas del VII Congreso de Diseño de Circuitos Integrados. pp. 521-522, Toledo, Noviembre 1992.

[MSTF93] D.Mozos, J. Septien, F. Tirado, M. Fernández "Guidance for Optimization-Based Synthesis Tools", Microprocessing and Microprogramming 37, pp. 95-96, 1993

[MSTH92] D.Mozos, J. Septien, F. Tirado, R. Hermida "Design Control in a High Level Synthesis System", Microprocessing and Microprogramming 34, pp. 93-96., 1992

[Muro82] S.Muroga "VLSI System Design" John Wiley, New York, 1982, Cap. 9, pp. 365-395, 1982

[NaGa92] S. Narayan, D. Gajski "System Clock Estimation based on Clock Slack Minimization" EuroVHDL 92, Hamburgo, pp. 66-71, Septiembre 1992

[Naka87] Y. Nakamura "An Integrated Logic Design Environment Based on Behavioral Description", IEEE Trans. on CAD, vol. CAD-6, n°3, pp. 322-336, 1987

[NaOg87] Y. Nakamura, K. Oguri "An RTL Logic Design Aid for Pararell Control VLSI Procesors", IFIP VLSI 87, pp. 12-28, Agosto 1987

[NONN90] Y. Nakamura, K. Oguri, A. Nagoya, R. Nomura "A Hierarchical Behavioral Description based CAD System", EURO ASIC-90, pp. 282-287, Mayo 1990

[NoPa93] M. Nourani, C. Papachistou, "A Layout Estimation Algorith for RTL Datapaths", 30th ACM/IEEE Design Automation Conference, DAC-93, pp. 285-291, 1993

[OIST86] Y. Ogawa, T. Ishii, Y. Shiraishi, H. Terai, T. Kozawa, K. Yujama, K. Chiba "Efficient Placement Algorithms Optimizing Delay for High-speed ECL Master-slice LSIS" Proc. 23rd Design Automation Conference DAC-86, pp. 404-411, June 1986

[Ohts86] T. Ohtsuki, "Layout Design and Verification" Ed. Elsevier Science, North Holland, 1986

[OTMM94] K. Olcoz, F. Tirado, H. Mecha, D. Mozos, J. M. Mendías, "Integración del Análisis y Mejora de la Testabilidad en una Herramienta de SAN" Actas del IX Congreso de Diseño de Circuitos Integrados. Las Palmas, pp. 325-330, Noviembre 1994

[OTMS93] K. Olcoz, F. Tirado, D. Mozos, J. Septién, R. Moreno, "Data-Path Structures and Heuristics for Testable Allocation in High Level Synthesis" Microprocessing and Microprogramming 39, pp. 263-266, 1993

[PaGa86] B.M. Pangrle, D. Gajski "State Synthesis and Connectivity Binding for Microarchitecture Compilation", Proc. of IEEE International Conference on Computer-Aided Design ICCAD-86, pp. 210-213, 1986

[PaGa87] B.M. Pangrle, D.D. Gajski " Designs Tools for Intelligent Silicon Compilation", IEEE Transactions on Computer-Aided Design Vol.CAD-6, n°6, pp. 1098-1112, Nov. 1987

[PaGH91]A. Parker, P. Gupta, A. Hussain "The Effects of Physical Characteristics on the Area-Performance Tradeoff Curve" Proc. of 28rd Design Automation Conference, pp. 530-534, 1991

[PaJD94] S. Parameswaran, P. Jha, N. Dutt "Resynthesizing controllers for minimum execution time" Proc. Second Asian Pacific Conference on Hardware Description Languages, APCHDL 94, pp. 111-117, Japon, Octubre 1994

[PaKG86] P.G. Paulin, J.P. Knight, E.F. Girczyc, "HAL: A Multi-Paradigm Approach to Automatic Data Path Synthesis", Proc. of 23rd Design Automation Conference, pp. 263-270, July 1986.

[PaKn87] P.G. Paulin, J.P. Knight, "Force-Directed Scheduling in Automatic Data Path Synthesis", Proc. of 24rd Design Automation Conference, Miami Beach, pp. 195-202, July 1987.

[PaKn89a] P.G. Paulin, J.P. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASICs", IEEE Transactions on CAD of ICs and Systems, Vol. 8, pp. 661-679, June 1989.

[PaKn89b] P.G. Paulin, J.P. Knight, "Scheduling and Binding Algorithms for High Level Syntesis", Proc. of the 26th Design Automation Conference, DAC-89, pp. 1-6, 1989.

[PaKn89c] P.G. Paulin, J.P. Knight, "Algorithms for High Level Syntesis", IEEE Design & Test of Computers, pp. 18-31, Diciembre 1989

[Pang88] B.M. Pangrle, "Splicer: A Heuristic Approach to Connectivity Binding", 25th Design Automation Conference, pp. 536-541, 1988

[PaPM86] A.C. Parker, J. T. Pizarro, M. Mlinar, "MAHA: A Program for Data Path Synthesis", Proc. of 23rd ACM/IEEE Design Automation Conference, pp. 461-466, 1986.

[Paul88] P.G. Paulin,"High-Level Synthesis of Digital Circuits Using Global Scheduling and Binding Algorithms", Ph.D. Thesis, Carlenton University, Otawa, Canada, February 1988.

[PePr89] Massoud Pedram, Bryan Preas, "Accurate Prediction of Physical Design Characteristics for Random Logic", Proc IEEE International Conference on Computer Design, pp. 100-108, ICCD-1989.

[PrLo88] B. Preas, M. Lorenzetti, "Physical Design Automation of VLSI Systems", Benjamin/Cummings Publishing Company, 1988

[RMJL94] M. Rim, A. Mujumdar, R. Jain, R.D. Leone "Optimal and Heuristic Algorithms for Solving the Binding Problem" IEEE Trans on VLSI Systems, vol2, n°2, pp. 211-215, Junio 1994

[SaBh80] S. Sahni, A. Bhatt "The complexity of design automation problems" Proc. 17th Design Automation Conference DAC80, pp. 402-411, 1980

[Saku83] T. Sakurai "Approximation of Wiring Delay in MOSFET LSI" IEEE Journal of Solid-Statre Circuits, Vol. SC-18, n°4, pp. 418-426, Aug. 1983

[SaPa86] Sastry, S. and Parker, A.C. "Stochastic models for wireability analysis of gate arrays", IEEE Trans. on Computer-Aided Design. CAD-5,n°1, pp. 52-65, January 1986

[ScKe72] D. G. Schweirkert, B. W. Kernighan "A proper model for partitioning of electrical circuits" Proc. 9th Design Automation Workshop, pp. 57-62, 1972

[Sech87] Sechen, C. "Average interconnection length estimator for random and optimized Placements", in IEEE Int. Conf. on Computer-Design, pp. 190-193, November 1987

[Sept90] J. Septién del Castillo, "Un Sistema de Asignación de Hardware para la Síntesis de Alto Nivel", Tesis Doctoral, Universidad Complutense de Madrid, 1990

[ShJa93] A. Sharma, R. Jain, "Estimating Architectural Resources and Performance for High-Level Synthesis Applications", IEEE Transactions on Very Large Scale Integration, vol.1 n°2, pp. 175-190, June-1993

[ShJa94] Alok Sharma, Rajiv Jain, "Register Estimation from Behavioral Specifications", IEEE Int. Conference On Computer Design ICCD 94, pp. 576-580, 1994

[ShMa91] K. Shahookar, P. Mazumder "VLSI Cell Placement Techniques" ACM Computing Surveys, vol. 23, n°2, pp. 143-221, June 1991

[ShWo89] H. Shin, N. S. Woo "A cost function based optimization technique for scheduling in Data-Path Synthesis" Proc IEEE International Conference on Computer Design: VLSI in Computers and Porcessors, Massachusetts ICCD 1989, pp. 424-427, 1989.

[SJWL93] M. Sheu, Y. Jeang, J. Wang, J. Lee "The determination of the cycle length in High Level Synthesis" Integration, The VLSI journal, vol 16, pp. 131-148, Ed. Elsevier, 1993

[SKILL93] "Design Framework II Commands". Vol. 1 y 2, "CADENCE Reference Manual", 1993

[SMHS91] J. Septien, D.Mozos, R. Hermida, A. Sotelo, "Bounding the Design Space in Hardware Allocation", Proc. IEEE International Conference on Circuits and Systems, pp. 913-916, 1991

[SMHT91] J. Septien, D.Mozos, R. Hermida, F. Tirado, "A Hardware Allocator Guided by Cost Functions", Microprocessing and Microprogramming 32, pp. 185-192, 1991

[SMTH90] J. Septien, D.Mozos, F. Tirado, R. Hermida, A. Sotelo, "Hardware Allocation in FIDIAS System", Proc. COMPEURO'90, pp. 55-59, 1990

[SMTH92] J. Septien, D.Mozos, F. Tirado, R. Hermida, M. Fernández, "Heuristics for Branch and Bound Global Allocation", EuroDac 92, Hamburgo, pp. 334-340, Sep. 1992.

[SMTH95] J. Septién, D. Mozos, F. Tirado, R. Hermida, M. Fernández, H. Mecha "FIDIAS: an Integral Approach to High Level Synthesis" IEE Proceedings Circuits, Devices and Systems, vol. 142, n°4, pp. 227-235, Agosto 1995

[Spie70] M. R. Spiegel "Manual de Fórmulas y Tablas Matemáticas" Libros McGraw-Hill, 1970

[SSHF89] A. Sotelo, J. Septien, R. Hermida, M. Fernández, "And Approach to Minimal-Time Scheduling of Micro-Operations" Microprocessing and Microprogramming 28, pp. 301-304, 1989.

[SSMT89] J. Septien, A. Sotelo, D.Mozos, F. Tirado, M. Fernández, "Behavioural Specification and Internal Representation in FIDIAS System", Proc. IASTED 89, pp.179-182, 1989.

[SuKe87] P. Suaris, G. Kedem "Quadrisection: A new approach to Standard Cells layout" Proc. International Conference on Computer Aided Design, pp. 474-477, 1987

[Supo83] K. J. Supowit, "Reducing channel density in Standard Cell Layout" Proc. 20th Design Automation Conference, DAC 83, pp. 263-269, 1983

[TiTi83] B. S. Ting, B. N. Tien "Routing Techniques for gate array" IEEE Trans. on Computer Aided Design, CAD-2, pp. 301-312, 1983

[TLWN90] D.E. Thomas, E.D. Lagnese, R.A. Walker, J.A. Nestor, J.V. Rajan, R.L. Blackburn, "Algorithmic and Register-Transfer Level Synthesis: The System Architect's Workbench", Kluwer Academic Publisher 1990

[TSSi86] C. TSeng, D.P.Siewiorek, "Automated Synthesis of Data Paths in Digital Systems", IEEE Trans. on CAD of Integrated Circuits and Systems, CAD-5, n°3, pp. 379-395, July 1986

[UeKS86] K. Ueda, R. Kasair, T. Sudo "Layout strategy, standarization and CAD tools" En "Layout Design and Verification", T. Ohtsuki, Ed. Elsevier Science, cap.1, New York 1986

[VeKi83] M. P. Vecchi, S. Kirkpatrick "Global wiring by simulated annealing" IEEE Trans on Computer Aided Design, vol 2, pp. 215-222, 1983

[WaCa91] R. Walker, R. Camposano. "A Survey of High Level Synthesis Systems" Kluwer Academic Publishers, 1991

[Wals75] G. R. Walsh "Methods for Optimization" John Wiley and Sons, New York 1975

[WaPa95] C. Wang, K.K.Parhi, "High Level DSP Synthesis Using Concurrent Transformations, Scheduling and Allocation" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, nº 3, pp. 274-295, Marzo 1995

[WeEs94] N.H.E. Weste, K. Eshraghian "Principles of CMOS VLSI Design" 2ª Edición, cap. 4, 1994

[WePa91] J.Pj Weng, A.C.Parker "3D Scheduling: High-Level Synthesis with floorplanning" Proc. of 28rd ACM/IEEE Design Automation Conference, pp. 668-673, 1991

[Wolf86] W. H. Wolf "Fred: A procedural database for VLSI design" in Proc. ACM/IEEE 23rd Design Automation Conference, pp. 744-751, June 1986

[Woo90a] N. Woo, "SAM: A Data Path Allocation System", Proc. of 1990 Custom Integrated Circuits Conference, May 1990

[Woo90b] N. Woo, "A Global, Dynamic Register Allocation and Binding for Data Path Syntesis System", Proc. of 27th ACM/IEEE Design Automation Conference, pp. 505-510, 1990

[WoSh89] N. Woo, H. Shin "A Technology-Adaptive Allocation of Functiona Units and Connections", Proc. of 26th ACM/IEEE Design Automation Conference, pp. 602-605, 1989.

[WuCG91] Allen C-H Wu,Viraphol Chaiyakul, Daniel D. Gajski "Layout-Area Models for High-Level Synthesis", in IEEE Int. Conf. on Computer Aided -Design, pp. 34-37, November 1991

[YoKu82] T. Yoshimura, E. S. Kuth "Efficient Algorithms for Channel Routing" IEEE Trans on Computer Aided Design of Integrated Circuits and Systems, CAD-82, pp. 25-35, 1982

[Yosh84] T. Yoshimura "An efficient channel router" Proc. of 21th Design Automation Conference, DAC 84, pp. 38-44, 1984

[Zimm88] G. Zimmerman "A New Area and Shape Function Estimation Technique for VLSI Layouts" Proc. of 25th ACM/IEEE Design Automation Conference, pp. 60-65, 1988.