UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

DEPARTAMENTO DE ELECTRÓNICA Y TELECOMUNICACIÓN

TESIS DOCTORAL

SÍNTESIS Y COMPILACIÓN DE CÉLULAS EN TECNOLOGÍA GAAS

JUAN ANTONIO MONTIEL NELSON

Las Palmas de Gran Canaria, Julio de 1994

46/1993-94

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
UNIDAD DE TERCER CICLO Y POSTGRADO

Reunido el día de la fecha, el Tribunal nombrado por el Excmo. Sr. Rector Magfco. de esta Universidad, el aspirante expuso esta TESIS DOCTORAL.

Terminada la lectura y contestadas por el Doctorando las objeciones formuladas por los señores jueces del Tribunal, éste calificó dicho trabajo con la nota de APTO CUM LAUDE POR UNANIMIDAD.

Las Palmas de G. C., a 15 de Julio de 1.994.

El Presidente: Dr. D. Juan Figueras Pamies,

El Secretario: Dr. D. Antonio Hernández Ballester,

El Vocal: Dr. D. Juan Manuel Meneses Chaus,

El Vocal: Dr. D. Juan Carlos López López,

El Vocal: Dr. D. Aurelio Vega Martínez,

El Doctorando: D. Juan Antonio Montiel Nelson,

NEGOCIADO UNIDAD DE TERCER CICLO

UNIVERSIDAD DE LAS PALMAS DE G.C.
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES

Tesis Doctoral

Síntesis y compilación de células en tecnología GaAs

Autor: D. JUAN ANTONIO MONTIEL NELSON
Directores: DR. D. ROBERTO SARMIENTO RODRÍGUEZ
DR. D. ANTONIO NÚÑEZ ORDÓÑEZ

Las Palmas de Gran Canaria, Julio de 1994.

UNIVERSIDAD DE LAS PALMAS DE G.C.
DOCTORADO EN INGENIERÍA INDUSTRIAL

DEPARTAMENTO DE ELECTRÓNICA Y TELECOMUNICACIÓN
PROGRAMA DE INGENIERÍA ELECTRÓNICA

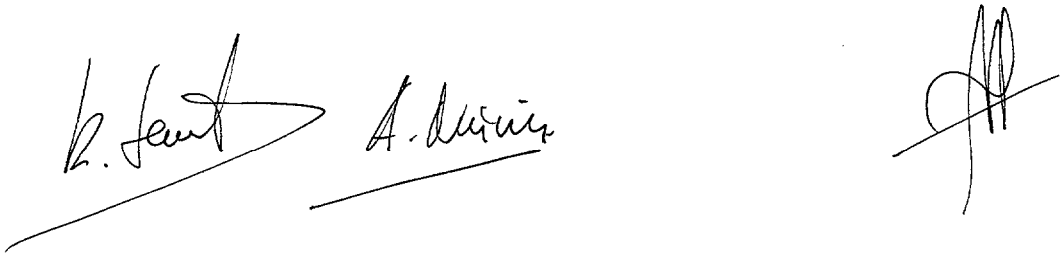
Tesis Doctoral

Síntesis y compilación de células en tecnología GaAs

Tesis Doctoral presentada por:
D. JUAN ANTONIO MONTIEL NELSON
Dirigida por los Doctores:
D. ROBERTO SARMIENTO RODRÍGUEZ
D. ANTONIO NÚÑEZ ORDÓÑEZ

Los Directores,

El Doctorando,



Las Palmas de Gran Canaria, Julio de 1994.

A Inma y Juan.

Resumen

Durante la década de los ochenta la tecnología de Arseniuro de Galio (GaAs) experimentó un crecimiento importante, debido a los avances en los procesos de fabricación, convirtiéndola en una solución viable en procesamiento y computación a muy alta velocidad. Las predicciones apuntaban que a principios de los noventa habría chips u obleas capaces de acomodar a varios cientos de miles de dispositivos. Las expectativas para la tecnología GaAs hacen pensar en un desarrollo similar al de la tecnología CMOS en Silicio. Por tanto, establecer metodologías que faciliten el diseño y desarrollar herramientas que permitan manejar la complejidad, es fundamental para el desarrollo de aplicaciones en tecnología GaAs.

El objetivo del presente trabajo es proponer una metodología de diseño en tecnología GaAs orientada a una estructuración sencilla y fácil, a la vez que robusta, del trazado. Ambos objetivos son muy sensibles a la aceptación industrial de la tecnología GaAs. La metodología de diseño se evalúa presentando tanto resultados experimentales como un estudio de inductancias series. Los resultados experimentales se obtuvieron diseñando y, caracterizando por medio de simulación, un conjunto de primitivas para procesamiento de señales bajo metodología tradicional y según la propuesta. Una vez abordado el aspecto metodológico del trazado se aborda el del dimensionamiento de los diferentes elementos del trazado físico.

Presentada la metodología de trazado físico, se propone la automatización del diseño, por medio de un entorno de síntesis de células para circuitos de muy alta velocidad en GaAs. **OLYMPO** es un conjunto completo de herramientas para planificación, colocado y ruteado de células y módulos orientado a tecnologías de altas prestaciones como GaAs y con especial énfasis en el desarrollo de entornos de compilación.

El proceso de evaluación de un entorno es una tarea que depende básicamente del uso del mismo, y por tanto, es un hecho contínuo, dinámico y a largo plazo. A pesar de esto, se propone validar el entorno implementando un generador de células (microrrotaciones), que forman parte de la sección de procesamiento de un procesador para la evaluación de la CFFT *Complex Fast Fourier Transform* en base 2 y precisión de 16 bits, basado en el algoritmo CORDIC (*COordinate Rotation DIgital Computer*).

Agradecimientos

Este trabajo es el punto de convergencia de muchos esfuerzos y aportaciones, que no son producto de una única persona. Desde estas líneas quiero agradecer a todos aquellos que han contribuido de una forma u otra.

Agradezco a Antonio y Roberto, la confianza puesta en mí cuando llegué al grupo. Sin sus ayudas y conocimientos este trabajo no hubiese cristalizado.

A Chiqui, porque desde que empezamos la carrera juntos he disfrutado de su amistad. A Jose por su buen humor y espíritu crítico. A Pedro por su hacer diario y desinteresado. A Tomás por su disponibilidad para con los demás. Gracias Alexis, por liberarme de muchas tareas tediosas del día a día. A Luis y Toni por acercarme sus conocimientos. A Francis por alumbrarme en el campo de la programación.

A todos, agradezco la ayuda que me han prestado.

Quiero agradecer especialmente a Alfonso lo mucho que me enseño en tan poco tiempo y a Carmen su incondicional apoyo y tesón.

A mi madre y a mi padre, porque siempre me han impulsado de forma decidida. Gracias a ellos soy lo que quise ser.

Todavía recuerdo aquellas largas tardes de verano del 90 en Tafira con Kamran Eshraghian, Roberto, Pedro, Chiqui Jose, Luis y Victor.

Sin duda aquello fué la semilla.

Mientras no se pueda medir
y expresar numéricamente
aquello de que se trata,
su conocimiento es imperfecto.

Lord Kelvin

Indice

Indice de Figuras

Indice de Tablas

Capítulo 1

Introducción.

Durante la década de los ochenta la tecnología de Arseniuro de Galio (GaAs) experimentó un crecimiento importante, debido fundamentalmente a los avances en los procesos de fabricación, lo que la ha convertido en una solución viable para muchas aplicaciones, por ejemplo, en procesamiento y computación a muy alta velocidad. Las expectativas para la presente década indican que se conseguirán densidades de integración que superan el millón de transistores. Es evidente que este nivel de integración requiere metodologías estructuradas para el diseño físico de los circuitos integrados en GaAs y, aún más, requiere la ayuda de herramientas que permitan una automatización del diseño y, en definitiva, que sitúen a la tecnología VLSI en GaAs a un nivel similar a la del Si en estos aspectos.

1.1 Planteamiento del problema. Necesidad y utilidad de esta investigación.

Las características inherentes de la tecnología GaAs y las soluciones que ofrece para la realización de sistemas electrónicos la hacen atractiva en muchas aplicaciones. Sin mencionar los campos analógicos y sobre todo de microondas (MMIC), dominados por el GaAs, podemos citar algunas de las aplicaciones donde éste adquiere especial relevancia o muestra potencialidades para el futuro: computación de alta velocidad, circuitos para comunicaciones e instrumentación.

El Arseniuro de Galio (GaAs) es un elemento semiconductor del grupo de los denominados compuestos III–V. Su principal ventaja respecto al Silicio es la mayor movilidad de los electrones, que permite realizar circuitos un orden de magnitud más rápidos. La tecnología GaAs ha evolucionado lentamente pero de forma constante en las dos últimas décadas. En los últimos cinco años, el avance tecnológico se ha acelerado debido a factores concurrentes como la disponibilidad de transistores MESFET de enriquecimiento, el aumento en el diámetro de las obleas (3 y 4 pulgadas), el uso de aluminio en múltiples capas de metal, procesos más estables y mejor controlados, mayor resolución litográfica y longitud de canal inferior a la mi-

cra, etc .... Todo ello ha llevado a esta tecnología a niveles de integración elevados (VLSI) en procesos maduros de alto rendimiento y costo competitivo.

El problema que plantea el diseño de circuitos integrados en GaAs empieza, por tanto, a tener cierta similitud con los planteados en la tecnología de Si. La complejidad de los diseños, la expansión de la tecnología a una amplia comunidad de diseñadores (debido a la difusión que están teniendo diversos procesos en GaAs mediante EUROCHIP y CMP en Europa ó MOSIS en Estados Unidos), la necesaria reducción de los tiempos y costes de diseño, etc. son problemas abordados previamente y que pueden servir de base para enfrentarnos a la nueva problemática contemplada. El diseñador de sistemas en GaAs utilizando la técnica *full–custom* a mano dispone de herramientas muy similares al Si, que le permiten realizar el trazado físico, el chequeo de reglas de diseño, la extracción de dispositivos, etc. La experiencia demuestra que la dificultad crece excesivamente cuando se desea manejar unas cuantas decenas de miles de dispositivos. Es en este supuesto en el que sería deseable que las facilidades (aún incompletas) que encuentra un diseñador en Si, tales como la generación y compilación de módulos, la síntesis de circuitos, etc .... estuviesen disponibles para el diseñador en GaAs.

Mientras el diseño progresa desde la especificación de alto nivel a la implementación física, la tecnología influye en muchas decisiones del diseño. En la implementación de un sistema VLSI, es preferible hacer tantos niveles de diseño independientes de la tecnología como sea posible. Sin embargo, el campo de aplicaciones posibles (más reducido en GaAs) y las características propias de la tecnología, hacen que muchos criterios y procedimientos utilizados en Si no sean directamente extrapolables al GaAs.

La razón principal para realizar sistemas en GaAs es la obtención de prestaciones mejoradas respecto a otras posibles soluciones tecnológicas. No valen soluciones que resuelvan algunos de los aspectos mencionados a costa de un perjuicio significativo de la frecuencia de trabajo. Los diseños se optimizan primordialmente en velocidad frente a otras optimizaciones (comprobabilidad, densidad, área, potencia etc ...) buscadas en Si. Es en este punto donde la decisión de traducir la descripcion funcional de un circuito a trazado físico usando librerías presenta deficiencias en GaAs. En Si el alto nivel de integración alcanzado hace que casi todo sistema pueda ser integrado monolíticamente. La concepción del circuito implementado a base de librerías parte de un marco mejor delimitado en viabilidad, cabida del sistema, interconexión, particiones y prestaciones esperadas, por lo que el diseñador le presta sólo una atención relativa. Esta idea de concebir sistemas en Si no es aplicable al GaAs, donde las implementaciones a nivel de células estándares no son del todo óptimas y la tendencia es la utilización de técnicas *full–custom.*

La base para el establecimiento de herramientas que permitan automatizar el diseño de circuitos integrados en GaAs, disminuyendo los tiempos y costos de diseño y haciendo los diseños mas robustos, es la creación de un sintetizador de células que permita la traducción de descripciones de alto nivel a trazado físico utilizando técnicas *full–custom.* Esta herramienta debe abordar los problemas nuevos que plantea la compilación de estructuras en GaAs, ya sean problemas relativos a las caracterísi-

cas diferenciadas de la tecnología o problemas generados por las altas frecuencias implicadas. Previamente, se hace necesario el desarrollo de una metodología de diseño físico que sea fácilmente estructurable y extendible a todas los procesos de fabricación y tecnologías existentes en GaAs.

1.2 Objetivos de la tesis y aportaciones.

La tesis aporta contribuciones a la metodología de diseño VLSI en tecnología GaAs, con especial énfasis en la compilación de estructuras. Ejemplos de algunos interrogantes cuya respuesta se aborda como objetivos son:

- Dado que la aproximación de librería de células estándares en GaAs es más comprometida que en Si, ¿qué metodología de trazado *full–custom* es la más adecuada en GaAs?

- Planteada una metodología de trazado, queda partido el espacio de diseño en subareas para acomodar dispositivos, zonas para el tránsito de señales y de distribución de la alimentación. Por tanto, ¿cómo se afronta el colocado automático de dispositivos? ¿los conocimientos adquiridos en Si en torno a la generación de células – matrices de puertas, redes de **Weinberger** – son extrapolables al GaAs?

- ¿Cómo resolver el trazado de señales de alta frecuencia – del orden de GHz – entre dispositivos?

- La conducción de la puerta del transistor MESFET en GaAs, es un problema que invita a realizar una planificación de la distribución de la alimentación, tanto a nivel de dispositivos como a nivel de células en un mismo chip. ¿Es posible obviar con alguna metodología de trazado el problema planteado?

- ¿Cuánto es posible independizarse de la tecnología y cuánto de las reglas de diseño de un proceso tecnológico, mediante el uso extensivo del diseño simbólico?

1.3 Método seguido.

El método seguido en este trabajo está basado en cinco fuentes de conocimiento: 1) reflexión sobre la amplia experiencia existente en Si y su extrapolación al GaAs, 2) la experiencia sobre entornos de diseño VLSI propios de Si, y la adaptación de algunos de sus elementos al GaAs, 3) la creación de un conjunto de herramientas para automatizar el diseño VLSI en GaAs, siguiendo técnicas *full–custom* y finalmente 4) la fabricación de circuitos aplicando la metodología y las herramientas desarrolladas.

La primera fuente de conocimientos ayuda a concebir la tecnología, y a establecer y definir las diferencias con las aproximaciones en Si. Permite también establecer ecuaciones y funciones que describan el comportamiento tecnológico a nivel de interconexiones entre dispositivos. Estos parámetros se fundamentan en otros proporcionados por los fabricantes. Esta vía permite concebir una metodología de trazado *full–custom* adecuada para tecnologías de alta velocidad SDCFL/DCFL GaAs MESFET o HEMT.

La segunda fuente recoge el estado del arte del diseño VLSI en Si y las herramientas disponibles en GaAs por simple adaptación de las de Si. Se plantea los nuevos problemas que surgen de elevar el diseño en GaAs al mismo nivel que el Si. Dada la metodología de diseño particular en GaAs según técnicas *full–custom*, se presenta una adaptación del algoritmo *alineamiento por enfriamiento simulado* para resolver el colocado de lógica (puertas y transistores). La adaptación de las técnicas de ruteado conocidas en Si es directa (con restricciones) en lo que se refiere a trazado de señales. Las alimentaciones son trazadas de forma distinta, mediante ruteadores no convencionales.

La tercera fuente de conocimientos ayuda a elegir el núcleo de implementación VLSI en GaAs. Mediante el estudio de las herramientas más ampliamente difundidas y su uso exhaustivo, se adquiere un fuerte experiencia en entornos VLSI y su implementación. Existen aproximaciones restrictivas, dependientes de la tecnología Si, que no permiten desarrollar el diseño VLSI en GaAs con el fin de situarlo en un nivel de implementación física parecida al del Si. Las experiencias en este sentido apuntan a la utilización de diseño simbólico y de disponer de un lenguaje de programación estándar, flexible y ampliamente conocido para la descripción del trazado físico. La elección del entorno adecuado es determinante para la viabilidad del trabajo.

El desarrollar un conjunto de herramientas para automatizar el diseño físico en GaAs, siguiendo técnicas *full–custom* es el resultado de los conocimientos adquiridos en las líneas anteriormente expuestas. Permite afianzar una metodología de desarrollo de entornos de diseño *bottom–up* y *top–down*, elegir el núcleo software de implementación y desarrollo VLSI y discernir hasta qué punto se puede ser independiente de la tecnología.

Por último, la fabricación y comprobación de circuitos es el elemento final en todo ciclo de diseño. Permite estimar, cuales son las desviaciones más importantes y reforzar los elementos dependientes de susodichas desviaciones. A la vez que sirve para demostrar y validar el conjunto de resultados obtenidos en este trabajo, corroborando en diversos aspectos la metodología.

1.4 Ordenación de la memoria.

La memoria de esta tesis se ha subdividido en siete bloques o capítulos. El primero lo constituye la actual introducción, el segundo es una aproximación al estado del arte de la tecnología y el diseño VLSI. Desde el tercero hasta el sexto se presentan las contribuciones realizadas. En el séptimo y último capítulo se describen las conclusiones, y líneas de investigación abiertas derivadas de este trabajo.

En el segundo capítulo se comienza haciendo una reflexión sobre las metodologías de trazado *full–custom*. Posteriormente se presenta una aproximación al estado del arte de la compilación en Si.

Los capítulos centrales de la tesis comienzan con una presentación breve de la tecnología GaAs, para posteriormente proponer una metodología de trazado físico *full–custom*. Este capítulo tercero termina enfocando la problemática asociada a la planificación de las interconexiones y de las alimentaciones. En el capítulo cuarto se establece una discusión acerca del desarrollo del proceso de síntesis en algunos entornos conocidos (Cadence/Edge, Mentor Graphics y OCTTOOLS), se presenta una adaptación de Cadence/Edge para diseño en GaAs y se plantea la problemática de la síntesis de alto nivel de VHSIC. Como colofón – capítulo quinto –, se analiza la automatización del trazado físico, resultando un conjunto de herramientas "OLYMPO" orientadas a la especificación de entornos de compilación. Estas herramientas se validan, en el capítulo sexto, comparando el trazado de la célula básica de una microrrotación de un procesador FFT, basado en algoritmo CORDIC, que hemos realizado a mano y el trazado generado mediante "OLYMPO".

El último capítulo detalla las aportaciones de esta tesis, así como, las líneas y trabajos futuros que vendrán a enriquecer y ampliar la línea de investigación abierta.

Capítulo 2

Estado del arte de las metodologías de diseño.

En la década de los ochenta, la complejidad de los circuitos integrados en Si, inducida por los avances en la tecnología VLSI, produjo una profunda crisis en el diseño. El tiempo tomado para diseñar chips llegaba a ser tan largo como la propia vida del producto. Esta situación desembocó en un bloqueo en el ciclo de desarrollo del producto. En la tecnología GaAs no es directa la aplicación de las herramientas desarrolladas en Si. Por otra parte, los avances tecnológicos permiten niveles de integración hoy en día del orden de 400.000 dispositivos activos [104].

Hay básicamente dos propuestas o escuelas de pensamiento [34] sobre la automatización del proceso de diseño:

- una, que mantiene que el proceso de diseño es difícil de automatizar y que el capital humano es la principal fuente de conocimientos. Por tanto, los diseñadores deben disponer de herramientas sofisticadas para la captura del diseño, verificación, análisis, y optimización. Estas herramientas suelen estar integradas en los entornos CAD para captura de esquemático o de trazado geométrico.

- La otra escuela de pensamiento mantiene que el conocimiento humano puede codificarse y que se pueden escribir compiladores que generen diseño VLSI de forma automática. Los compiladores de circuitos (Si o GaAs) son programas que generan trazado geométrico a partir de una descripción de nivel superior. La descripción de alto nivel puede ser un trazado simbólico, un esquemático de un circuito, un conjunto de expresiones Booleanas, un esquema lógico, una descripción a nivel de comportamiento de una microarquitectura, un conjunto de microinstrucciones, o un algoritmo para procesamiento de señales.

2.1 Trazado *full-custom*: metodologías.

El tamaño y complejidad de los circuitos en GaAs imponen el uso de la representación simbólica y la división jerárquica del espacio de diseño. Estas técnicas abogadas por **Mead & Conway** se han establecido difinitivamente en Si. Para la tecnología GaAs no se puede decir lo mismo, debido principalmente a la dificultad de adaptar herramientas del Si para esta tecnología.

2.1.1 Trazado físico manual.

El tamaño de los circuitos **VLSI** ha hecho que el trazado manual para un diseño completo sea casi imposible. A pesar de esto, sigue siendo utilizado. En la tecnología GaAs, a pesar del gran número de estructuras regulares (**ROM, RAM, ALU**) que pueden implementarse, muchos de los diseños todavía se hacen manualmente. La principal razón es la buena densidad del trazado hecho a mano y, por encima de todo, su aptitud de ocupar el área dejada por bloques regulares. No hay duda, sin embargo, que para circuitos **VLSI** es obligatorio desplazarse a métodos más sofisticados.

2.1.1.1 Entornos gráficos.

De entre los sistemas más conocidos desde mediados de los ochenta se encuentra **CAESAR** [74]. Su derivado **MAGIC** [75, 76], añade la comprobación de reglas de diseño de forma implícita, conexionado automático de capas entre posiciones definidas por el usuario, replicado, copia, etc ... **MAGIC** ha servido para que otros sistemas progresasen en la línea de pensamiento de disponer una interfaz de trazado potente al usuario. Esto requiere un gran esfuerzo, por parte del usuario, en adiestramiento de herramientas. Aparte de **MAGIC** – utilizado en GaAs, por su sencillez – a finales de los ochenta principios de los noventa, aparecen entornos que dominan el mercado, tales como **SDA** [97] (hoy en día *Design Framework* de **Cadence**) y **GDT** de **Mentor Graphics**. Las últimas versiones de estos poderosos entornos de diseño se ejecutan bajo *Windows* requiriendo mayores niveles de computación y de memoria, lo cual supone una inversión costosa en plataformas de diseño.

2.1.1.2 Lenguajes de descripción.

Hay otra forma de definir patrones geométricos; es utilizar un lenguaje de descripción, en vez de una entrada gráfica. Este método, tiene la ventaja de ofrecer todo lo concerniente a un lenguaje de programación procedural, esto es, parámetros, lazos condicionales e incondicionales, etiquetas, variables de computo de bucles y un sin fin de posibilidades. Mediante lenguajes de descripción se pueden definir células parametrizables independientes de la tecnología.

El método más obvio de añadir programabilidad al trazado es partir de un lenguaje completo de programación. Muchos sistemas empiezan con lenguajes conocidos y añaden subrutinas o macros [5, 109, 82]. Los lenguajes de programación resultantes se denominan **lenguajes empotrados**. Hay también lenguajes de programación que han sido creados exclusivamente para la tarea de diseñar. En el entorno **OCTTOOLS** desarrollado en **Berkeley** por el **Department of Electrical Engineering and Computer Sciences**, hay claros ejemplos de ambos posibilidades.

Los lenguajes de diseño pueden interaccionar de mejor forma si estos son interpretados por los sistemas CAD, más que si son compilados. Es obvio, desde el punto de vista del usuario de entornos interactivos, que los lenguajes intérpretes son más adecuados que los compilados. Sobre intérpretes de **LISP**, se han desarrollado lenguajes y sistemas tales como, **DPL** [5], **NS** [22], **Skill** [95]. Estos sistemas son adecuados cuando el diseñador interacciona con el entorno, puesto que una interfaz uniforme del lenguaje controla el proceso de edición, el acceso a la base de datos, herramientas internas, y algunas veces el sistema operativo.

2.1.2 Trazado simbólico.

Las metodologías basadas en el trazado simbólico son un medio de abstracción del detalle y de la tarea laboriosa del diseño de máscaras. Presenta ventajas sobre el trazado manual tales como una menor duración del tiempo de diseño y la facilidad de corrección del mismo. En esencia el uso de simbología reduce la complejidad del diseño, y permite a los diseñadores con experiencia acometer circuitos más complicados, y lo que es también importante, permite a los diseñadores noveles completar los diseños con un alto grado de seguridad.

2.1.2.1 Trazado sobre rejilla fija.

El sistema de trazado sobre rejilla fija, divide la superficie de diseño en una rejilla uniformemente espaciada tanto en las direcciones x como y. El tamaño de la rejilla representa la mínima regla de diseño que satisfacen todos los elementos del trazado. Se define un símbolo por cada combinación de capas permitidas.

La figura 2.1 muestra un ejemplo de uso de un lenguaje simbólico [111] basado en una rejilla fija y la geometría generada. Aunque el trazado basado en una rejilla fija simplifica las especificaciones, sacrifican eficiencia tecnológica por utilizar siempre un tamaño de rejilla del peor caso [34].

2.1.2.2 Trazado de tiras sobre rejilla virtual y rejilla relativa.

El término **sticks** [112, 113], es un concepto genérico dado a los sistemas de diseño simbólico que no restringen al diseñador a una rejilla fija, y que generalmente requieren que se introduzca una descripción **topológica** a través de un sistema **gráfico**

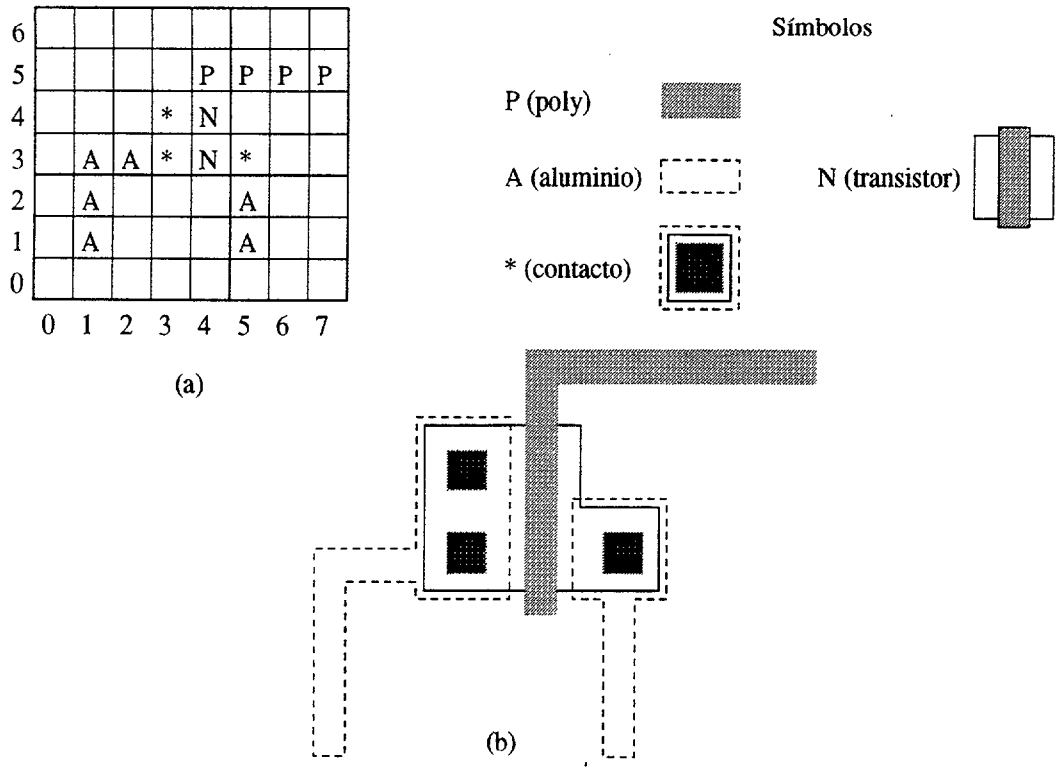


FIGURA 2.1: Arquitectura de rejilla fija: (a) Código y (b) trazado geométrico.

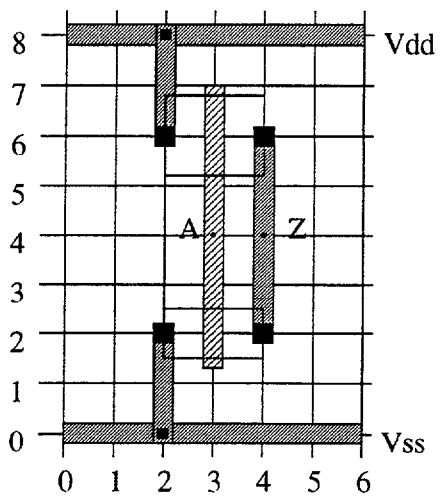
interactivo. Esta descripción gráfica tiene la forma de tiras geométricas en máscaras de trazado.

MULGA [110, 109, 30] creado en los laboratorios **Bell**, consiste en un entorno que permite la edición gráfica, compactación/espaciado, extracción de dispositivos, de un trazado simbólico sobre una rejilla virtual. Este entorno simbólico está escrito en lenguaje de programación C, sobre sistema operativo UNIX. El diseñador edita el trazado simbólico, en forma de **tiras** (*sticks*), por medio de un sistema gráfico interactivo. El diagrama resultante se traduce a un lenguaje intermedio **ICDL** (*Intermediate Circuit Description Language*) que también puede servir para la definición del trazado. En la figura 2.2, se presenta el ejemplo de un inversor y su descripción en **MULGA**.

LAVA [103] es un lenguaje tipo *sticks* basado en una rejilla virtual que se traduce a coordenadas reales. Este lenguaje de igual forma que **CHISEL** [51], está basado en lenguaje C, del cual hereda mucho de su estilo y sobre el que se implementa. **STICKS** de **CALMA** elimina la rejilla si el diseñador lo considera oportuno, resultando un proceso de diseño totalmente libre de reglas de diseño geométricas. El lenguaje procedural **ALI** [61] utiliza una rejilla para indicar únicamente la situación relativa de los transistores, contactos y conexiones, y para determinar la conectividad eléctrica del circuito. Este lenguaje basado en una rejilla relativa permite al usuario realizar el trazado geométrico a un nivel conceptual, en el cual sin dimensiones, ni posiciones de sus componentes se realiza la especificación.

```
BB0068
nt: device n(3,2) or = east
pt: device p(nt,6) or = east w = 2
    wire alum (0,0) (6,0)
    wire alum (0,8) (6,8)
    wire poly  (3,2) (3,6)
    wire alum (2,0) nt.s
    wire alum pt.s (2,8)
    wire alum nt.d pt.d
    contact md nt.d
    contact md nt.s
    contact md pt.d
    contact md pt.s
    contact vss (nt.d,0)
    contact vdd (nt.d,8)
vss: pin alum (0,0)
vdd: pin alum (0,8)
  a: pin poly (3,4)
  z: pin poly (4,4)
```

(a)



(b)

FIGURA 2.2: Arquitectura de rejilla virtual. (a) Lenguaje. (b) Trazado.

Cadence aboga por un entorno de edición simbólica como alternativa a su editor de polígonos [95, 97]. Mediante este entorno se puede diseñar, editar, y compactar el trazado sin tener en cuenta reglas de diseño. Las reglas que se utilizan durante el trazado manual, ahora forman parte del fichero tecnológico. Los dispositivos que se utilizan son comprobados automáticamente durante la fase de compactación. El trazado simbólico no soporta la conectividad eléctrica en su estructura, siendo esto una gran desventaja. Además la descomposición simbólica sobre cualquier dispositivo se hace hasta el nivel de máscaras. Mediante el lenguaje **Skill** se puede definir el trazado simbólico, además de añadir cualquier restricción tecnológica en forma de regla de diseño.

2.1.2.3 Ventajas y desventajas del trazado simbólico.

La simplificación de las reglas de diseño geométricas, alivia al diseñador de los detalles que pueden enturbiar otros objetivos más importantes y globales. El hacer el trazado transparente a las reglas de diseño hacen la tecnología más independiente del proceso. Si en un proceso las reglas se cambian, la descripción a nivel de máscaras de un circuito puede regenerarse con un mínimo esfuerzo relativo.

Los sistemas de diseño simbólico tienen, además de los beneficios de un entorno libre de reglas de diseño, las bases para capturar la conectividad del circuito, aunque muy pocos (Cadence) se han beneficiado de este detalle. Lo anterior se debe principalmente al hecho de que los problemas específicos, en particular la compactación,

se han dirigido más a completar el ciclo de diseño que a las herramientas relevantes. Como dato significativo, se puede perder de un 10 a un 20% en densidad, comparado con una reducción del 50 al 75% del tiempo de trazado [34].

2.2 Compiladores de circuitos.

El uso de los compiladores de circuitos permite a los diseñadores de sistemas trabajar sobre un nivel de abstracción superior, sin el conocimiento de los detalles del diseño de circuitos integrados y de la tecnología. La compilación mejora la calidad del diseño por ser **correcto por construcción**. Cada componente del diseño puesto por un compilador está libre de errores y satisface todas las reglas de diseño dependientes de la tecnología. Los errores de diseño introducidos por el diseñador a un nivel alto, más abstracto, son más fáciles de detectar y corregir.

2.2.1 Dominios de descripción.

El término compilación de Si fue introducido por Dave Johannsen en **Caltech** [50], donde lo utilizó para describir el concepto de ensamblado de piezas parametrizadas de trazado geométrico. El término ha ganado popularidad a través de la comunidad CAD de CI, donde ha sido utilizado con una gran variedad de matices diferentes.

En un sentido estricto, es una extensión del concepto de célula estándar, donde las células estándares son sustituidas por un compilador de células parametrizadas que permite al usuario adaptar la funcionalidad de la célula, sus parámetros eléctricos y de composición geométrica. En el caso de primitivas, tales como puertas NAND o NOR, el usuario puede especificar el número de entradas, elegir los *drivers* necesarios, y seleccionar la posición de algunos terminales de I/O en los bordes de la célula.

En un sentido más amplio, la compilación de Si se puede definir como un proceso de traducción desde un nivel superior de descripción a trazado geométrico. Este proceso de traducción se puede romper en algunos niveles, y cada nivel puede considerarse un compilador de más bajo nivel. De esta forma se puede definir un compilador lógico como el que traslada una descripción a un conjunto de puertas lógicas, o un compilador de microarquitecturas el que traduce un repertorio de instrucciones a un conjunto de registros, buses y unidades aritmético–lógicas.

Para representar estas diferentes propuestas de la compilación de Si, se describirá el diagrama en Y [33, 106]. Los ejes en el diagrama (véase figura 2.3) representan tres dominios diferentes de descripción: comportamiento, estructural, y físico. A lo largo de cada eje se sitúan los diferentes dominios de descripción. La raíz es el centro del eje Y, y representa el nivel más abstracto de descripción. Representando las herramientas de diseño como arcos en un mismo eje o entre ejes, se muestra qué información se utiliza en cada herramienta y que información se genera. Las herramientas de verificación se representan por lazos que empiezan y terminan en el mismo punto del eje. Los puntos equidistantes del centro del diagrama Y, represen-

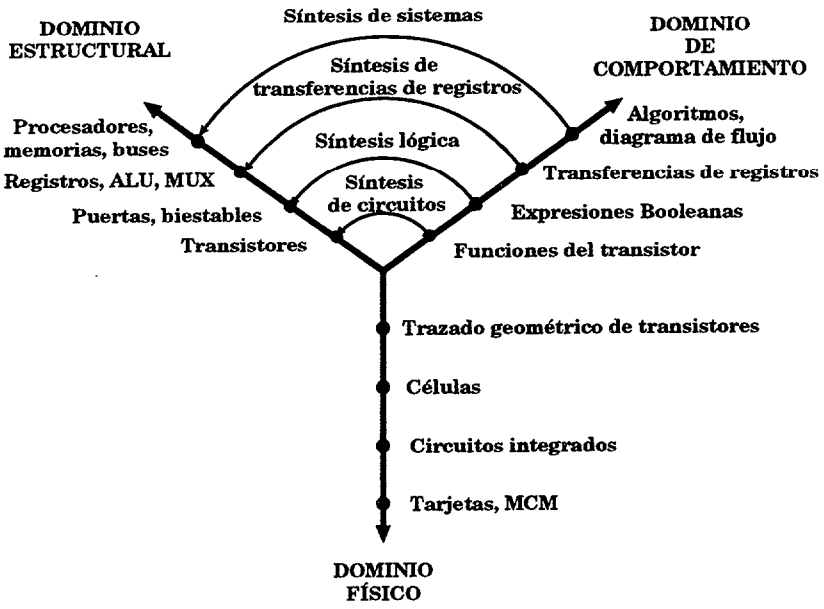


FIGURA 2.3: Diagrama en Y.

tan toda la información conocida del diseño en un punto y en el tiempo. El círculo más externo representa el nivel de sistema, el próximo es el nivel de algoritmo, seguido por el nivel de microarquitectura, el lógico, y el de circuito.

El diagrama en Y se puede utilizar para representar diferentes niveles de compilación en Si. Generalmente se denomina compilador de Si a los programas que traducen una descripción estructural o de comportamiento a un trazado geométrico. Esta traducción de comportamiento a trazado normalmente se hace en dos pasos. La traducción de comportamiento a estructura se denomina síntesis, puesto que la descripción del diseño se sintetiza a partir de un conjunto de componentes predefinidos. La traducción desde la estructura a la geometría se denomina planificación, colocado y conexionado o trazado geométrico, dependiendo del nivel de diseño.

Un compilador ideal generalizado puede constar de cuatro pasos o compiladores. El **compilador de sistema** que descompone un programa o algoritmo en un conjunto de procesos que se comunican. Un **compilador de procesador** que descompone cada proceso en un conjunto de componentes a nivel de microarquitectura o módulos. Cada **compilador de módulo** genera una matriz regular o irregular de células. Un **compilador de células** descompone una célula en puertas, transistores, polígonos. Cada paso de síntesis es seguido por un paso de colocado y conexionado, y la composición geométrica final se obtiene jerárquicamente a través de llamadas recursivas al *compilador de componentes*, el cual devuelve la composición geométrica en términos de los componentes respectivos.

La compilación en Si no se limita necesariamente a diseños de construcción completa o *full–custom*, también se aplica igualmente a metodologías predifundidas o *semi–custom*. En el caso de células estándares, el compilador descompone la descripción inicial en células a través de la síntesis de sistema, procesador, y módulo y

luego coloca y conecta esas células. La composición geométrica de las células procede de una librería de células estándares. La misma estrategia se utiliza para las redes de puerta, donde el diseño en su globalidad se descompone en puertas colocadas con anterioridad y posteriormente conectadas.

2.3 Compilación de células.

Un compilador de células traduce una descripción a nivel de comportamiento o una descripción estructural a trazado geométrico. Esta descripción estructural puede ser un esquema lógico o circuital. Puesto que la generación de trazado es difícil a partir de un esquema, el proceso de compilación se simplifica restringiendo la arquitectura del trazado geométrico, el dimensionamiento del mismo, la descripción de la entrada de la célula, el formato de salida, o el método de dimensionamiento de dispositivos.

2.3.1 Compilación y estilo de trazado geométrico.

El **estilo de trazado geométrico** queda definido por un conjunto de reglas globales de disposición geométrica y por la capacidad del compilador de satisfacer las restricciones de tamaño, forma, y posiciones de los terminales de I/O. Las reglas globales especifican las relaciones topológicas entre los objetos del trazado (transistores, contactos y conexiones), orientación de los objetos, y asignación de capas. Algunos ejemplos de restricciones globales son: el que todas las capas verticales se tracen en polisilicio y todas las conexiones horizontales en la capa de metal 1, o el que todos los dispositivos tipo **p** se sitúen en una fila y todos los tipo **n** en una fila paralela.

En una **arquitectura fija** el patrón de trazado geométrico se define al mismo tiempo que se escribe el compilador. Este patrón es una plantilla donde se sitúan los objetos cuya presencia o ausencia determina la funcionalidad y la interfaz de la célula. El principal problema es encontrar que asignación de lugares a objetos minimiza algunos aspectos del diseño, tales como el área o el tiempo de retardo. En una arquitectura fija el usuario del compilador tiene muy poca libertad en controlar la forma del trazado geométrico y la localización de los terminales. Consecuentemente, el trazado final no encaja muy bien en el entorno, y se necesitarán algunos canales adicionales para definir la interfaz de la célula de acuerdo con sus vecinas.

En una **arquitectura** de células **flexible** al diseñador se le permite especificar libremente la forma del trazado geométrico y la posición de los terminales. Ello sugiere el planteamiento de una estrategia descendente (*top–down*), donde el diseño se divide en células y se definen las interfaces antes de que se genere el trazado geométrico específico. De esta forma, la conexión de las células se realiza mediante alineado, adosado y conexionado a través o sobre células.

La composición geométrica se puede restringir a una sóla dimensión, dando lugar a **arquitecturas de trazado unidimensionales**. En este estilo los componentes

del trazado, puertas y transistores se sitúan en una matriz lineal con conexiones entre componentes en pistas paralelas o sobre la propia matriz. A medida que la complejidad del circuito crece, el tamaño de la matriz y la longitud media del conexionado crece linealmente con el número de componentes. En una **arquitectura bidimensional**, el incremento es proporcional a la raíz cuadrada del número de componentes.

La entrada a un compilador de células consta de dos partes. La primera define las **restricciones de entorno** que deben satisfacerse de acuerdo a la interfaz con las células vecinas. Estas restricciones incluyen el tamaño de las células y la especificación de I/O. El tamaño puede ser fijo, bien en ambas dimensiones o en una sola, dejando la otra dimensión libre.

La segunda parte de la especificación de entrada describe la **funcionalidad de la célula** o la **estructura** a implementar. La descripción funcional consiste en un conjunto de ecuaciones Booleanas o expresiones que definen la máquina de estados. Se utilizan operadores abstractos que tienen que ser sustituidos por los componentes reales mediante el sintetizador. La descripción estructural consiste en una lista de componentes y sus interconexiones. Los componentes pueden ir desde transistores, puertas, o biestables a alguna otra construcción de alto nivel.

La salida del compilador puede ser bien geométrica o simbólica. Una **descripción de trazado geométrica** es un producto preparado para fabricación que satisface las reglas de diseño dependientes de la tecnología. Una **descripción de trazado simbólica** utiliza símbolos para representar transistores, contactos, y conexiones así como sus posiciones relativas. Sin embargo, ni la posición absoluta es exacta, ni el tamaño es definido. Un paso de postcompilación es necesario para sustituir los símbolos en sus equivalentes geométricos y compactar la representación geométrica de acuerdo con las reglas particulares de la tecnología. Una representación simbólica descarga al compilador de la consideración de las reglas de diseño y hace el proceso de compilación más eficiente. La simplicidad del proceso, por otra parte, permite el uso de algoritmos más complejos para mejorar la eficiencia tecnológica y los tiempos de ejecución – para grandes circuitos –. Además, un compilador de células puede adaptarse más fácilmente a diferentes procesos de fabricación si se retrasa la dependencia del proceso de compactación/espaciamiento.

2.3.1.1 Células estándares.

La **arquitectura de célula estándar**, actualmente la más popular en los circuitos ASIC, divide el área de trazado geométrico en un conjunto de filas paralelas separadas por canales de conexionado, tal como se muestra en la figura 2.4. Cada fila consta de un número de células predefinidas obtenidas de una librería de células estándares.

El proceso de compilación consta básicamente de un colocado y conexionado. El colocado es la asignación de células estándares a posiciones en una fila, y el conexionado es la asignación de una interconexión en el canal de conexiones entre

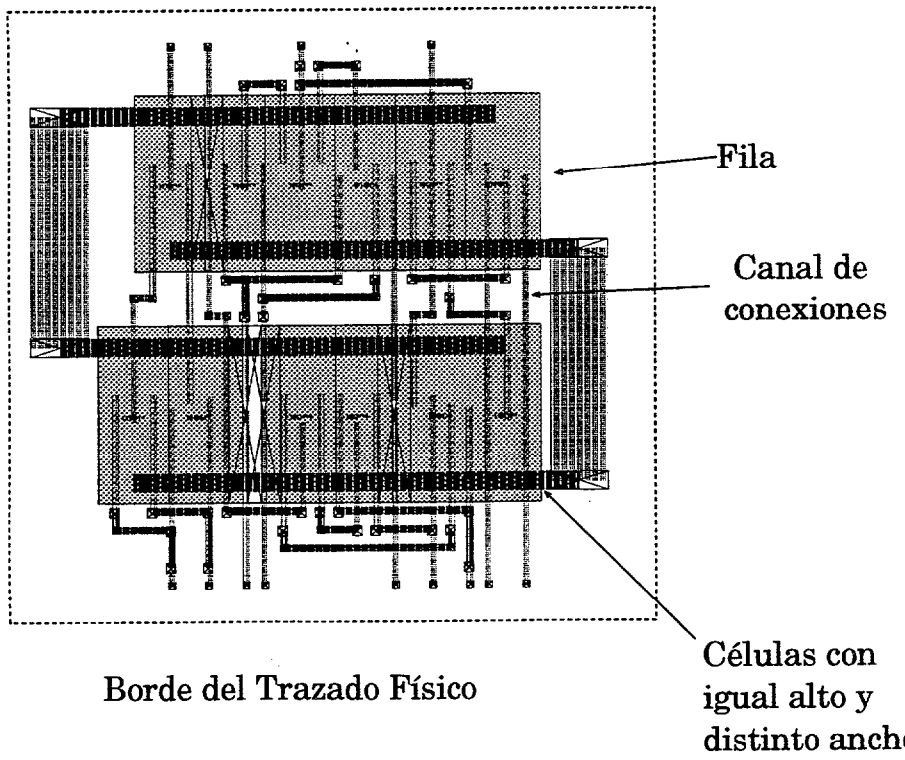


FIGURA 2.4: Arquitectura de células estándares.

dos filas. El objetivo del colocado y conexionado es minimizar la longitud media del cableado. Durante la fase de colocado se utilizan estimadores de la longitud del cableado de un nodo puesto que no se conoce el trazado de conexiones. Una buena aproximación es la suma de los altos y anchos de los rectángulos que encierran el nodo. El rectángulo que encierra un nodo es el rectángulo más pequeño que cubre a todos los terminales conectados al nodo.

Las ventajas de la arquitectura de células estandares son su simplicidad y popularidad, lo cual ha liderado el desarrollo de muchas herramientas y algoritmos tanto de colocado como de conexionado. Las desventajas son la baja utilización del área y la necesidad de mantener la librería de células. Debido al tamaño fijo de las células, posiciones de terminales I/O, y funcionalidad de las células estándares, normalmente del 60% al 80% del área de composición geométrica se destina a conexionado [101]. Además, cualquier cambio en el proceso de fabricación requiere rediseñar las células puesto que éstas han sido desarrolladas para un proceso particular. Para obviar la dependencia con cambios de proceso en una misma tecnología, se han desarrollado librerías de células simbólicas estándares y modelos de alto nivel parametrizados.

2.3.1.2 Colocado.

El colocado automático es difícil, debido al problema extremadamente complejo que se debe resolver. Por ejemplo, obviando el conexionado, el problema de colocado es simplemente un problema de optimización de área del tipo **NP–completo** [35].

El procesamiento puede tomar un tiempo irrazonable para resolver el problema. Además si se añaden consideraciones de conexionado, entonces para cada combinación de posiciones de células que se proponga también se requerirá una gran cantidad de tiempo para conectar los diferentes elementos. Debido a esto, el colocado no se puede realizar óptimamente. Mediante procedimientos heurísticos se han de generar resultados tolerables en un tiempo razonable.

Uno de los procesos más conocidos de colocado automático es el *min–cut* [10]. Esta técnica divide el conjunto de células sin posicionar en dos grupos que pertenecen a lados distintos del conjunto. Cada lado es posteriormente subdividido hasta que se obtenga un grafo con estructura de árbol. El objetivo es hacer la división de células de tal forma que se corte el menor número de conexiones (figura 2.5).

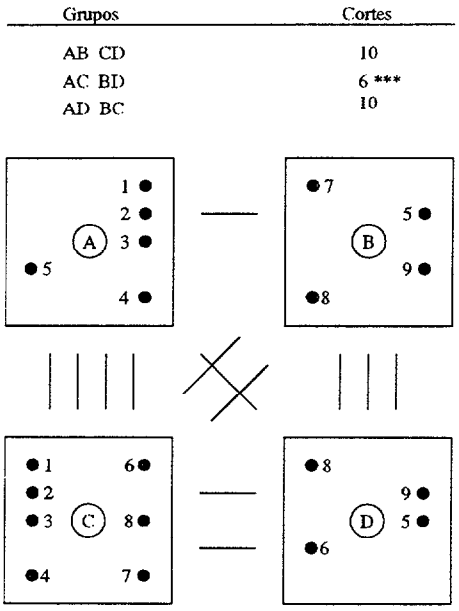


FIGURA 2.5: Método de colocado *min–cut*.

Otra manera de hacer lo mismo, denominada *bottom–up* [79, 44] es conceptualmente lo opuesto al método *min–cut*, dado que lo que se hace es ir creciendo los grupos en vez de irlos dividiendo. El método *bottom–up* asume que todas las células son distintas y deben agruparse de acuerdo con sus conexiones. Se utiliza una función de peso que indica la proximidad de las células. Esta función, como la del método *min–cut*, tiene en cuenta el número de conexiones que deben eliminarse para agrupar dos células. Esta función también puede tener en cuenta tamaños de células y otros factores. La figura 2.6 muestra esta técnica.

Junto con los métodos *min–cut* y *bottom–up* aparecen aquellos que tratan de encontrar analogías con sistemas físicos que tienen características similares al problema planteado. Por ejemplo, si un circuito se ve como una red resistiva, en el que cada célula es un nodo de tensión constante y cada grupo de interconexiones es una resistencia, entonces el problema del colocado es análogo al de determinar las tensiones correctas en cada nodo. Los valores de las tensiones que estén próximas

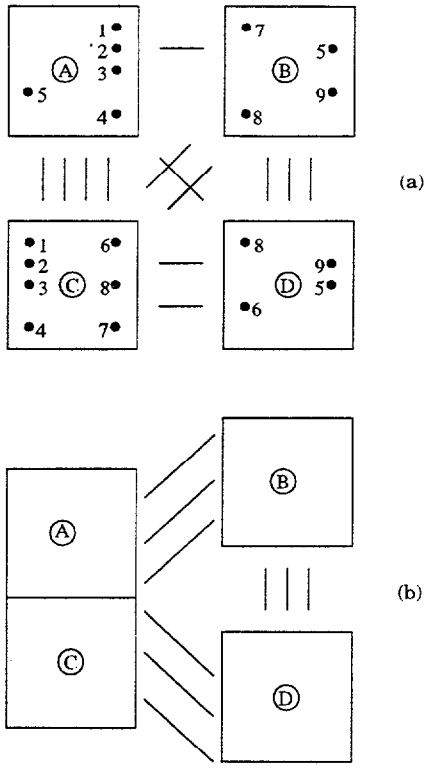


FIGURA 2.6: Método de colocado *bottom–up*.

indican células que deben estar próximas. Para implementar esta técnica, el cuadrado de la longitud de las conexiones es la función objetivo, la cual se resuelve con métodos matriciales [21].

Otra emulación física es la denominada alineamiento por enfriamiento simulado [55, 88]. Este método también es muy útil en compactación y conexionado. Dado un posicionamiento inicial, se realizan varios cambios evaluando los resultados. La severidad de los cambios disminuye hasta que no haya detalle que ajustar. Al igual que otras tareas de alineamiento, este método consume muchos recursos de computación pero produce muy buenos resultados.

2.3.1.3 Ruteado.

A un ruteador se le da un conjunto de terminales de células que deben conectarse y éste debe resolver el problema. El espacio entre esas células puede ser tan simple como un rectángulo o tan complejo como un laberinto (véase figura 2.7). Cuando el espacio de conexionado es rectangular y contiene puntos de conexión en las dos caras se denomina canal, y cuando contiene puntos de conexión en las cuatro caras se denomina *switchbox* (caja de conexión). Las áreas de conexión complejas se descomponen en canales.

El conexionado de laberinto (*maze routing*) es la manera más simple de conectar, puesto que sólo considera una conexión a la vez. Dados dos puntos de conexión

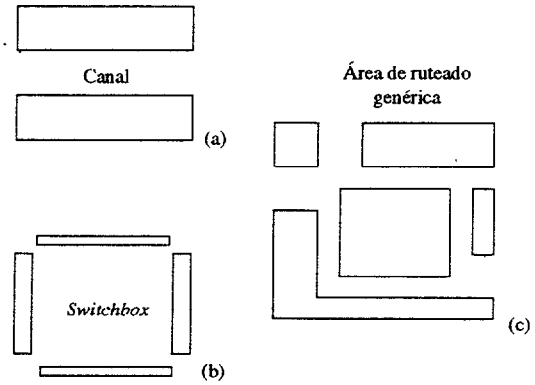


FIGURA 2.7: Áreas de ruteado. (a) Canal. (b) *Switchbox*. (c) Area de ruteado genérica.

y un conjunto de obstáculos, el algoritmo encuentra un camino. Los obstáculos pueden ser células, los contactos o las conexiones que ya existan y que no pueden ser atravesadas. La primera técnica que se conoce se basa en una rejilla de puntos que debe ser cruzada por las conexiones [71]. El algoritmo conocido con el nombre de **Lee–Moore**, funciona disponiendo anillos concéntricos de radios monótonamente crecientes en torno a un valor central de partida S. Cuando un anillo alcanza el punto final F, se define una ruta contando hacia atrás a través de los anillos (véase figura 2.8). Este método es tan simple que hay desarrollos a nivel de circuitos

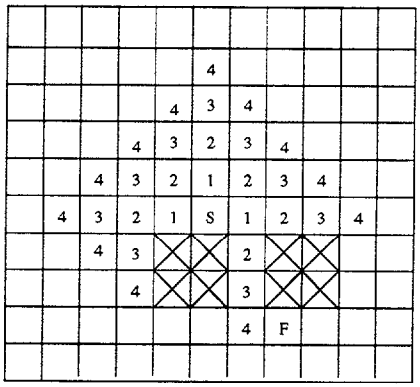


FIGURA 2.8: Ejemplo de aplicación del algoritmo de **Lee–Moore**.

integrados [7].

Estas técnicas básicas se pueden extender hacia el uso de múltiples capas. Generalmente la conexión cambia de capa en el trayecto donde se cruzan otras conexiones. En algunos esquemas una capa va horizontal y otra vertical [29, 54]. Conectar en una capa no es difícil, puesto que todas las conexiones van paralelas. Por lo que la única cuestión es cómo ordenar los conjuntos de rutas paralelas. Entre otras consideraciones, lo más importante es minimizar el número de cambios de capas, puesto que un excesivo cambio de capas puede retrasar las señales. Algunas herramientas [29] tienen parámetros que permiten elegir bien longitudes extras de conexión o cambio

de capas. Otra consideración es introducir una prioridad en las capas que se han de utilizar, e incluso restringir a unas capas determinadas la conexión de algunas señales [53].

Generalmente, las conexiones de alimentación van siempre en una capa de metal específica. En Si y para asegurar lo anterior lo primero que se conecta son las alimentaciones. El sistema de conexión de alimentaciones tipo **PI** utiliza el algoritmo del cartero (*traveling–salesman*). Esta técnica encuentra la ruta que recorre a cada célula una sola vez de forma óptima [78].

Como alternativa al procedimiento "conexión a conexión", está el método de realizar todas las conexiones a la vez. Estos métodos pueden utilizar bien dos lados enfrentados del canal (*channel router*) o bien todos los lados (*switchbox router*). Si el área de conexión no es rectangular, debe partirse en múltiples áreas rectangulares asignando puntos de conexión a lo largo de la línea de división (véase figura 2.9), a esta fase se le denomina ruteado global [57].

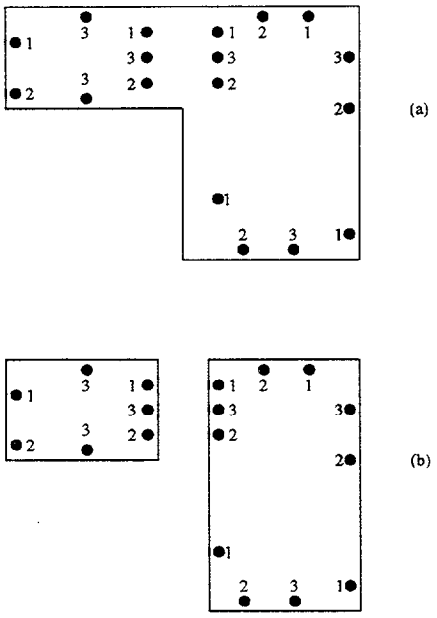


FIGURA 2.9: Ruteado global. (a) Antes. (b) Después.

En los ruteadores de canal o de caja de conexión todas las conexiones van sobre una rejilla de tamaño fijo. En algunas implementaciones se crean nuevas líneas de la rejilla entre las filas y columnas iniciales, si son necesarias para completar la conexión sastifactoriamente [66, 57].

Aunque hay muchas implementaciones [28, 79, 29], se discutirá brevemente una implementación muy conocida y que engloba la mayor parte de las técnica [41]. Se trabaja con un *switchbox* que contiene puntos de conexión, obstáculos, y obstrucciones, haciendo las conexiones con dos capas. La obstrucciones, según muestra la figura 2.10, son aquellas áreas que pueden cruzarse sólo por un capa determinada y los obstáculos bloquean el cruce de todas las capas, forzando a que las conexiones las rodeen.

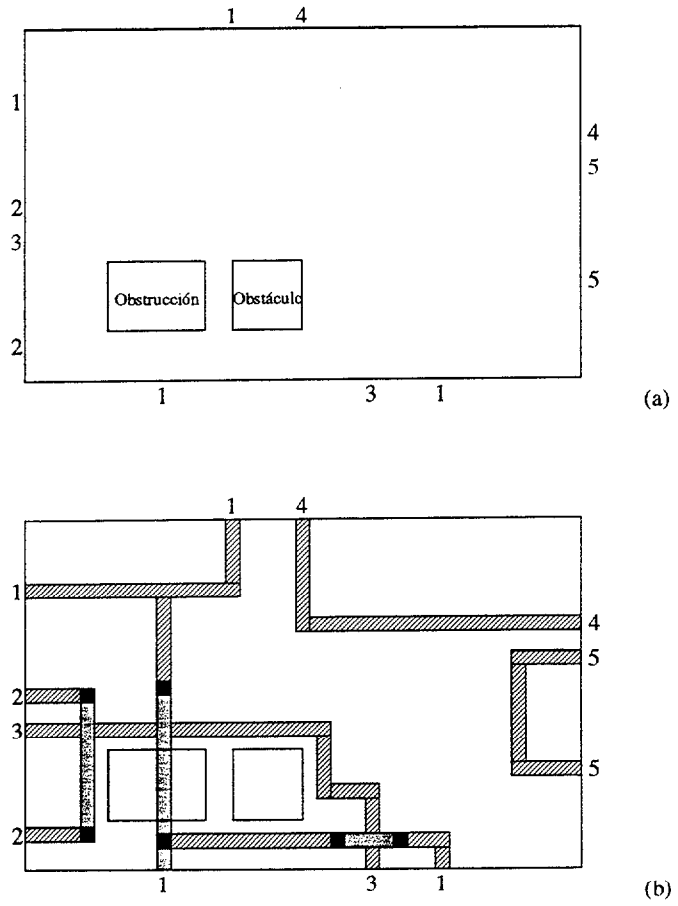


FIGURA 2.10: Ruteado de un canal *switchbox*. (a) Configuración inicial. (b) Después del ruteado.

Cada nodo está en una pista horizontal particular y debe cambiarse eventualmente de una a otra para completar la conexión. Las conexiones se desarrollan sobre una columna, empezando desde la izquierda y barriendo el área hacia la derecha. En cada columna, hay reglas que sirven para determinar qué nodos saltarán de unas pistas a otras. Cuando se alcanza el lado derecho del área, el trazado queda completado y todos los nodos quedan conectados. Para poder implementar este modelo, los nodos conectados en la parte superior e inferior del canal deben llevarse hasta las primeras pistas horizontales, y cualquier conexión a los nodos que no se encuentra en el centro de la pista se cambia de dirección hasta llevarla a su lugar correcto. Mientras se va procesando cada columna, se va dando una prioridad a aquellas conexiones que pueden colapsar un nodo. Las capas van cambiando cuando se producen cambios de la dirección horizontal a la vertical, permitiendo que las conexiones verticales y horizontales se puedan cruzar. Para evitar excesivos cambios de capas, hay un procesamiento posterior que elimina los cambios innecesarios minimizando el uso de vías.

El incremento en el número de máscaras dedicadas a interconexión ha supuesto

el desarrollo de ruteadores de tres máscaras, que incluyen conexiones sobre las células [24, 48, 59, 114, 99].

2.4 Compilación de módulos.

Los módulos pueden definirse como entidades de microarquitectura que llevan a cabo funciones, constituidas por una o más matrices de células de un tipo específico. Ejemplos de estos son ROM, RAM, PLA, registros, multiplicadores, sumadores, contadores, ALU y rutas de datos. El diseño de un compilador de módulos consiste en describir la plantilla, las células, interfaces, personalización de la plantilla y modelos.

2.4.1 Implementaciones.

Hay algunas implementaciones que utilizan editores de trazado geométrico para la especificación de la célula y descripciones textuales para la plantilla. Existen entornos que utilizan menús [58] o gráficos [67, 97, 95, 96] para especificar la plantilla. En estos sistemas gráficos, se utilizan **diagramas de ensamblaje** donde se muestran las posiciones relativas y las orientaciones de las células que componen un determinado bloque del módulo. Cada célula puede ser bien otro bloque de células o un trazado.

El propio diagrama de ensamblado puede tomar parámetros, computar variables locales, y pasar nuevos valores hacia los niveles de jerarquía inferior. De igual forma, los componentes de niveles inferiores pueden acceder a parámetros y variables definidas en niveles jerárquicos superiores. Además el diseñador debe verificar que para toda combinación de células, no se violan las reglas de diseño. Estas tareas pueden simplificarse y codificarse dentro del compilador [20]. En vez de especificar varias versiones de una misma célula, sólo se especifica una versión, y el compilador llevará el ajuste correspondiente, tanto mediante el alineamiento de los terminales de I/O, como mediante el uso de un ruteador. Posteriormente el compilador puede realizar la compactación solapando células para compartir buses y líneas de alimentación o espaciarlas para evitar violaciones de reglas de diseño.

Capítulo 3

Metodología de trazado físico propuesta para GaAs.

En este capítulo presentamos una metodología de diseño físico orientada a una estructuración sencilla y fácil a la vez que robusta del trazado. Esta metodología permitirá la automatización del trazado físico en tecnología GaAs, tal como se verá en el capítulo quinto. La metodología de diseño se evalúa presentando tanto resultados experimentales para diversas unidades funcionales, como un estudio de inductancias series en el trazado de interconexiones y alimentaciones. Los resultados experimentales se obtuvieron diseñando *full–custom* y caracterizando por medio de simulación un conjunto de primitivas para procesamiento de señales, tanto la metodología tradicional usada en Si, y según la desarrollada para circuitos en GaAs. Una vez abordado el aspecto metodológico del trazado – planificación de interconexiones y notación de topología en anillo – se aborda el dimensionamiento de los diferentes elementos del trazado físico.

Introducida la metodología, analizaremos algunos entornos de diseño existentes presentando una adaptación de Cadence/Edge para trazado físico en GaAs. Finalmente pasaremos a presentar en el capítulo quinto la herramienta desarrollada para sintetizar células con esta metodología. Esta presentación incluirá tanto aspectos formales generales sobre estructura, gramática, transformaciones y funcionalidad de la herramienta, como ejemplos y detalles prácticos de diseño.

3.1 Tecnologías GaAs para diseño VLSI.

La tecnología GaAs ha avanzado notablemente en los últimos años. Actualmente existen tres generaciones de dispositivos. Los MESFETs (*MEtal–Semiconductor FETs*) y los HEMTs (*High Electron Mobility Transistors*) que han alcanzado un nivel comercial, y los HBTs (*Heterostructure Bipolar Transistors*) que están en fase de laboratorio. En MESFETs, los procesos de fabricación sobre $1\mu m$, han madurado suficientemente, y muchas empresas del sector han introducido CI digitales de alta densidad. Además, existen diversos fabricantes ofreciendo procesos tecnológicos por debajo de $0.5\mu m$ que permiten niveles de integración similares a los alcanzados por el Si. Por otra parte, son diversos los laboratorios que disponen de tecnología suficiente para realizar HEMTs con longitudes de puertas inferiores a 0.3 μm.

En uno y otro caso las mayores densidades se consiguen utilizando lógica DCFL (*Direct Coupled FET Logic*). DCFL es la familia más compacta en GaAs, y la más adecuada para sistemas VLSI de altas prestaciones [62]. Su estructura re-

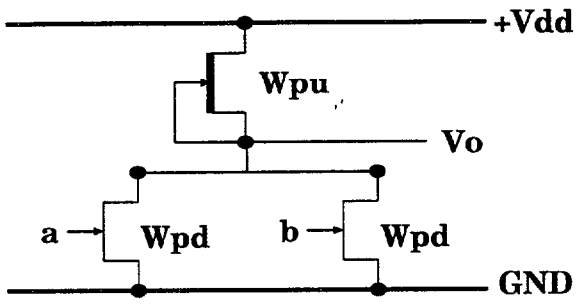


FIGURA 3.1: Representación de una puerta NOR en lógica DCFL.

cuerda la lógica nMOS en Si, utilizando un transistor de deplexión como carga activa y un transistor de enriquecimiento que actúa de conmutador. La figura 3.1 muestra un ejemplo de puerta NOR de dos entradas. En este tipo de lógica, al igual que en la nMOS, la dimensión de los dispositivos determina las características de la señal eléctrica de salida (niveles lógicos, márgenes de ruido, etc ...). Es por tanto, una lógica de ratios. Se define como relación de aspecto al coeficiente $\beta = \frac{W_{\text{pull-down}}/L_{\text{pull-down}}}{W_{\text{pull-up}}/L_{\text{pull-up}}}$. La lógica DCFL presenta algunas limitaciones, que afectan la

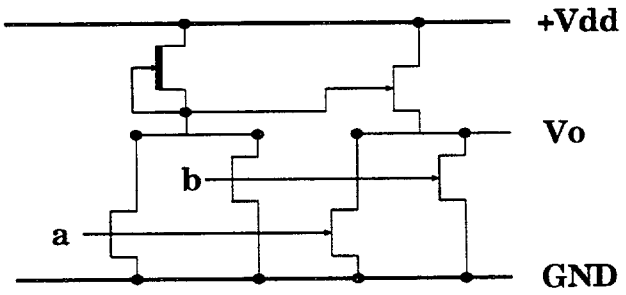


FIGURA 3.2: NOR de dos entradas en *superbuffer FET logic*.

forma de concebir e implementar los diseños. Aunque algunas de estas limitaciones están generadas por el uso de metal para la puerta (caso de los MESFETs), por ejemplo los reducidos márgenes dinámicos de ruido ($\sim 100mV$) y las elevadas corrientes de fuga, otros problemas son debidos a la topología de la lógica. En efecto, los circuitos DCFL tienen gran sensibilidad a la carga (bien debida al *fan–out* o al cableado) y al *fan–in*. El desarrollo de sistemas complejos en DCFL requiere en algunas partes del diseño del desarrollo de técnicas de amplificación y recuperación de los niveles lógicos. Este es uno de los campos donde mayor cantidad de estudios ha habido últimamente [62].

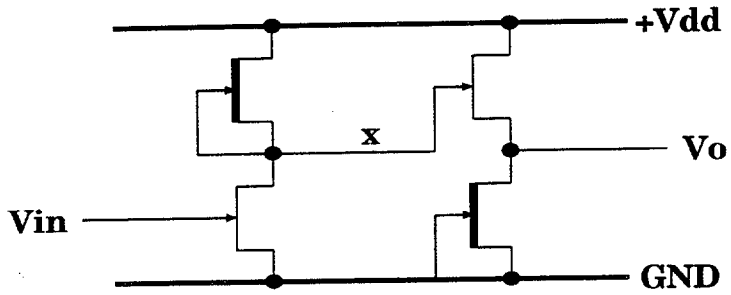


FIGURA 3.3: Inversor con *buffer*, lógica SDCFL.

Una de las más conocidas técnicas de amplificación es el uso del denominado *super–buffer* (véase figura 3.2). Con esta solución la capacidad de carga se mejora notablemente. Sin embargo, el uso de esta lógica debe restringirse debido principalmente al ruido que genera en los buses de alimentación y de la depencia que presentan sus prestaciones frente a la temperatura. Nuestro grupo ha venido desarrollando otra solución [84] para mejorar los márgenes de ruido y la robustez del circuito, que consiste en utilizar un seguidor de fuente como *buffer*. La figura 3.3 muestra un inversor en lógica SDCFL.

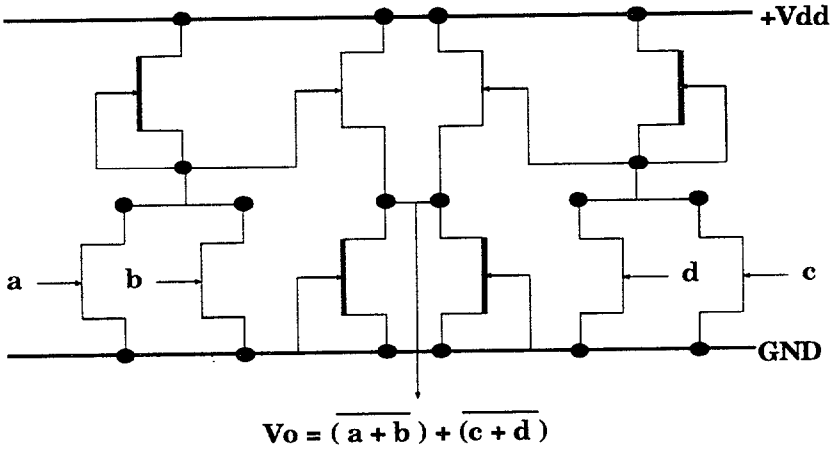


FIGURA 3.4: Representaciones de una puerta O–A–I en lógica SDCFL.

Por otro lado, la lógica SDCFL aporta cierta flexibilidad en el diseño digital en GaAs. Una estructura básica muy importante realizada con SDCFL es la O–A–I.

Con ella se pueden obtener algunos bloques funcionales (MUX, XOR, etc.) de forma más eficiente que en DCFL. La O–A–I se realiza utilizando dos puertas NOR SDCFL y una OR cableada a la salida (véase figura 3.4).

La estructuración de las interconexiones se puede extender para incluir una tercera fuente de alimentación ($-V_{ss}$) muy utilizada en las etapas de amplificación de una puerta en lógica SDCFL, como aparece en la figura 3.5.

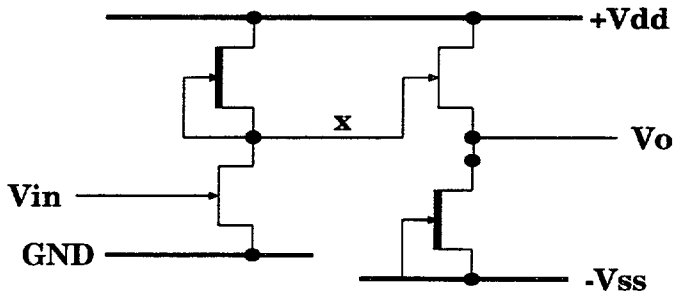


FIGURA 3.5: Inversor en lógica SDCFL con tres alimentaciones.

Nuestro objetivo es realizar un sintetizador de células totalmente independiente de la tecnología y utilizando lógica DCFL y SDCFL. Se hace necesario clasificar las tecnologías según sus características más importantes, o al menos las más relevantes en relación al desarrollo de un sintetizador de células. La primera división la realizamos entre dispositivos MESFETs y HEMTs. En MESFETs para diseño digital existe una amplia gama de tecnologías:

Vitesse presenta una tecnología para diseño digital donde compromete algo la ventaja en velocidad del GaAs con el fin de obtener un mayor nivel de integración. La tecnología H–GaAsII, $0.8\mu m$ de longitud de puerta, permite unos niveles de integración de hasta 400000 transistores. El proceso admite hasta cuatro niveles de metalización, aunque el último está especialmente pensado para planos de alimentación. A la vez cabe la posibilidad de utilizar diferentes longitudes de puertas. El proceso H–GaAsIII, mejora el anterior, disminuyendo hasta $0.6\mu m$ la longitud de puerta.

Fujitsu dispone de un proceso de fabricación MESFET para circuitos digitales, de $0.6\mu m$ de puerta y hasta cuatro niveles de metalización.

TriQuint se centra en aplicaciones digitales por encima de $1GHz$. A diferencia de **Vitesse**, **TriQuint** no mantiene una línea de producto digital estándar, muchas de sus implementaciones de alta frecuencia son diseños basados en células estándares, para fibra óptica y aplicaciones de procesamiento de señales.

Thomsom es segunda fuente de Vitesse y por tanto su proceso tiene las mismas características. En resumen, se nos presentan dos casos bien diferenciados. El primero de ellos comprende a todos los fabricantes que proporcionan más de dos niveles de metal, y el segundo a los que disponen de dos niveles de metal. En cuanto a los HEMTs, existen en Europa dos procesos muy avanzados:

Philips Microwave Limeil ofrece la tecnología con transistores HEMT de $0.2\mu m$ de longitud de puerta (ED0.2AH) que alcanza una frecuencia de hasta $53GHz$. Esta tecnología está disponible en nuestro laboratorio a través del proyecto GARDEN.

El Instituto Fraunhofer de Friburgo (IAF) proporciona mediante EUROCHIP un proceso en GaAs de transistores HEMT digital, de dos niveles de metalización (una de ellas con *airbridge*) que permite realizar divisores dinámicos de frecuencia de hasta $50GHz$. Este proceso es de $0.3\mu m$ de longitud de puerta.

No existen diferencias claras en los procesos en HEMTs anteriormente mencionados. Desde el punto del sintetizador estas tecnologías no ofrecen diferencias y pueden tratarse unificadamente, aunque la utilización de la herramienta que se pretende desarrollar requerirá ficheros tecnológicos que expresen sus especificidades.

3.2 Metodología utilizada para la automatización del trazado físico.

3.2.1 Estilo de trazado.

Para la realización del trazado geométrico de circuitos en lógica DCFL, puede utilizarse la metodología usada en tecnología nMOS. En ella el transistor de conmutación (*pull-down*) se sitúa por debajo del de carga (*pull-up*). El bus de alimentación V_{dd} se localiza en el parte superior y el bus de retorno GND en la parte inferior del trazado, tal y como se muestra en la figura 3.6.

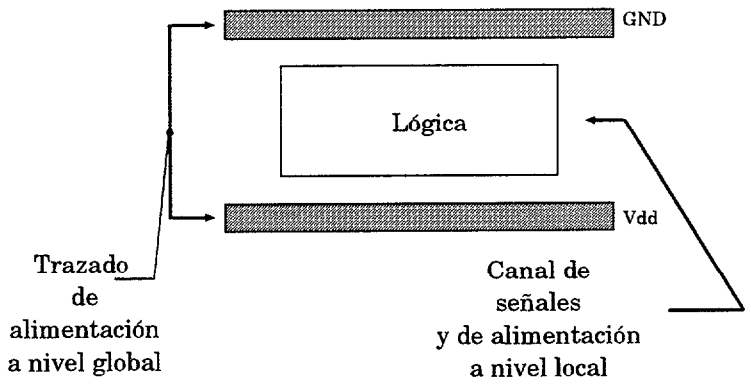


FIGURA 3.6: Estilos de trazado: propuesta nMOS.

El principal problema que presenta esta metodología, cuando se aplica al diseño en GaAs y se utiliza la lógica DCFL, es que el transistor E–MESFET es de un orden de magnitud mayor que el D–MESFET, por lo que el espacio perdido es importante. Cuando se utiliza lógica SDCFL, la disposición de los transistores de la etapa de amplificación atenúa esa pérdida de área. Sin embargo, para facilitar la automatización del trazado físico, hemos desarrollado y evaluado una metodología

para diseño de circuitos integrados en GaAs denominada **Notación en Anillo** [32]. En esta metodología, el bus GND se sitúa entre la lógica y el bus V_{dd}, tal y como se ilustra en la figura 3.7(a). Una segunda opción de este estilo de trazado se muestra en 3.7(b). Incluso esta metodología permite reducir las inductancias series de las

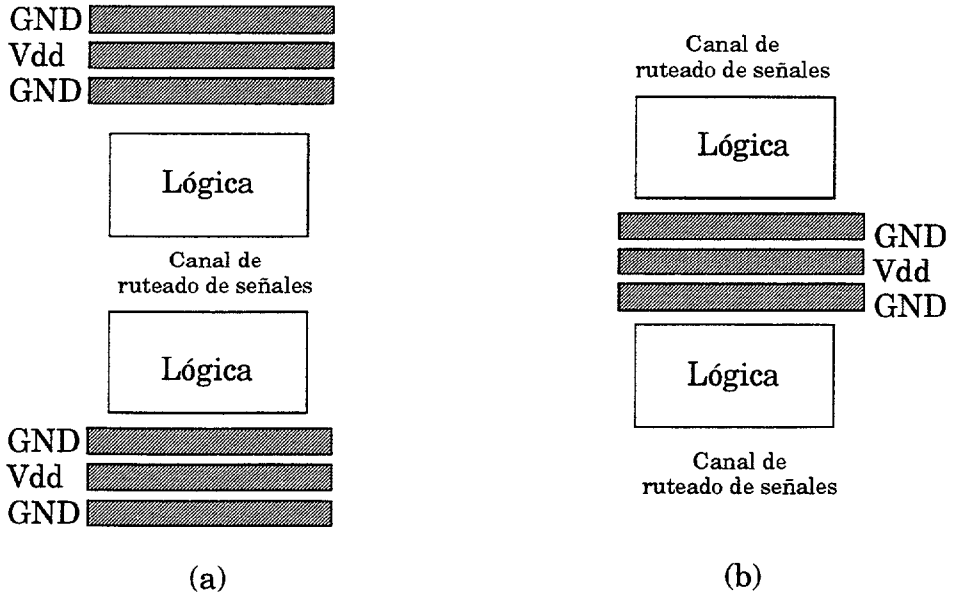


FIGURA 3.7: Estilos de trazado: propuesta Notación en Anillo. (a) Notación en Anillo utilizando buses laterales. (b) Notación en Anillo utilizando buses centrales.

líneas de alimentación y los acoplamientos con las señales, tal y como se presentará en la sección 3.2.2.2.

La **Notación en Anillo** permite una formulación intermedia que se muestra en el inversor de la figura 3.8(b). En esta figura la línea de puntos representa un transistor **E–MESFET** de conmutación, mientras que la línea sólida representa el transistor **D–MESFET** de carga. Puesto que en esta tecnología sólo se admiten ramas en paralelo en la ruta de entrada, es decir, solamente puertas NOR, la representación de las puertas NOR puede simplificarse eliminando las ramas paralelas de las entradas, según se muestra en la figura 3.9(a),(b).

En la figura 3.10 mostramos distintas representaciones de la puerta OR en DCFL. La traducción desde la **Notación en Anillo** a un trazado simbólico convencional y posteriormente a un trazado geométrico es directa. Lo que viene a ser importante en la **Notación en Anillo**, como método intermedio de describir un trazado físico, es que se pueden destacar las rutas de las señales y a continuación formular una estrategia de interconexión, antes de traducirla en un trazado definitivo. En una tecnología de dos niveles de metalización, como la de **TriQuint**, hay un sólo nivel de alimentación (alimentación local). Los buses se distribuyen horizontalmente y se dimensionan de acuerdo a las necesidades de alimentación. En la figura 3.11, se muestra el área dedicada a la lógica, los canales de conexión de señales y la

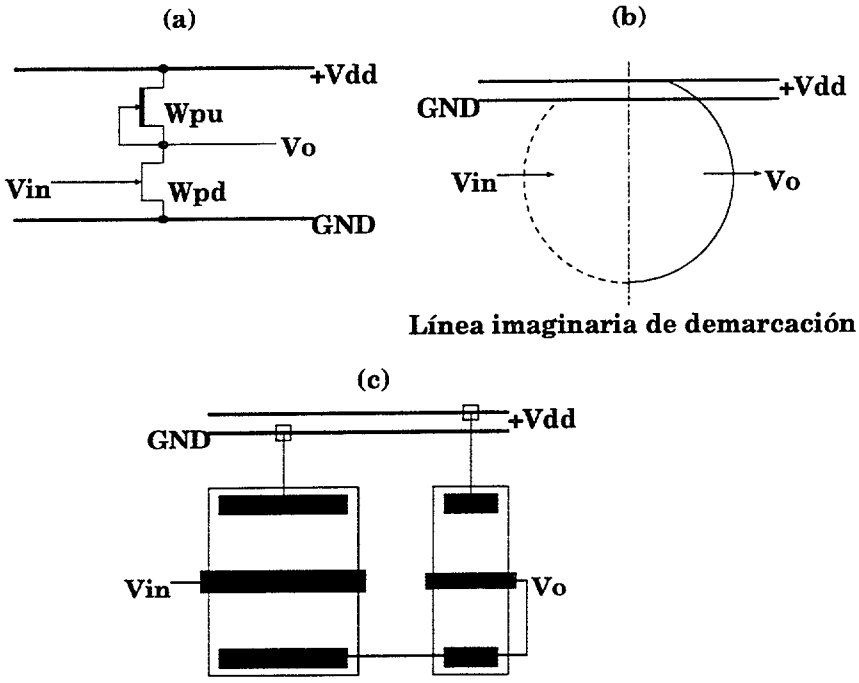


FIGURA 3.8: Representaciones de un inversor en lógica DCFL. (a) Circuito. (b) Notación en Anillo. (c) Notación Simbólica.

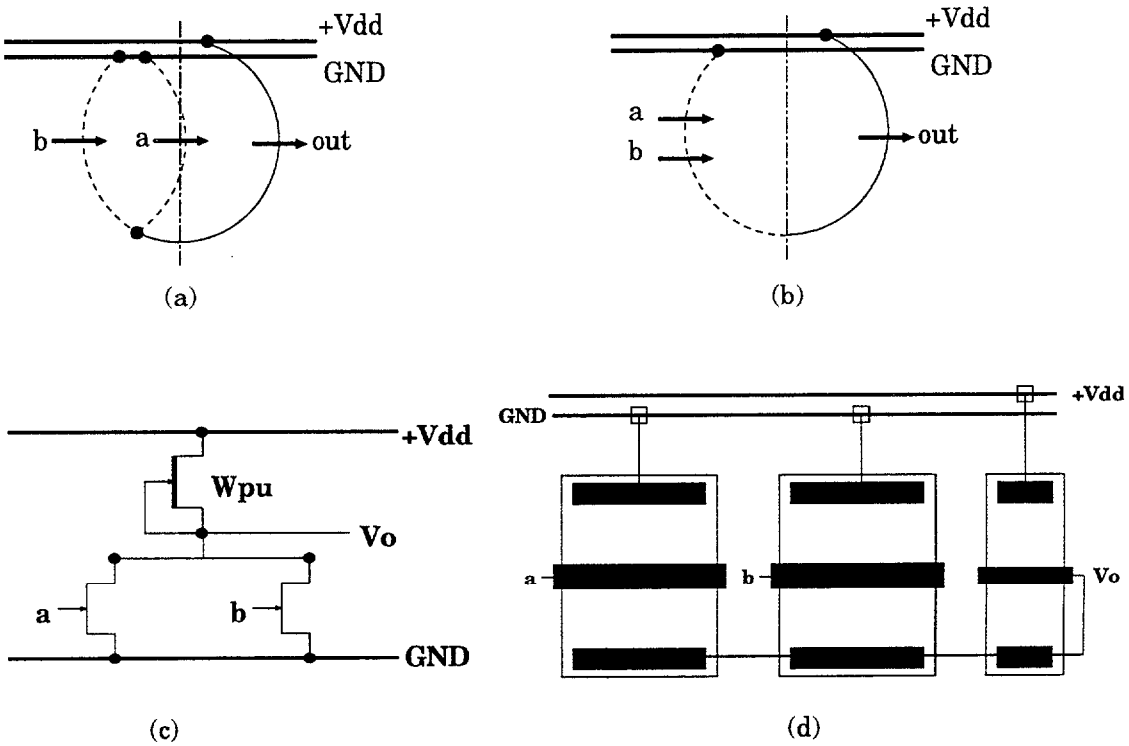


FIGURA 3.9: Topología de la Notación en Anillo para una puerta NOR en DCFL: (a) y (b) Simbología, (c) circuito y (d) trazado simbólico.

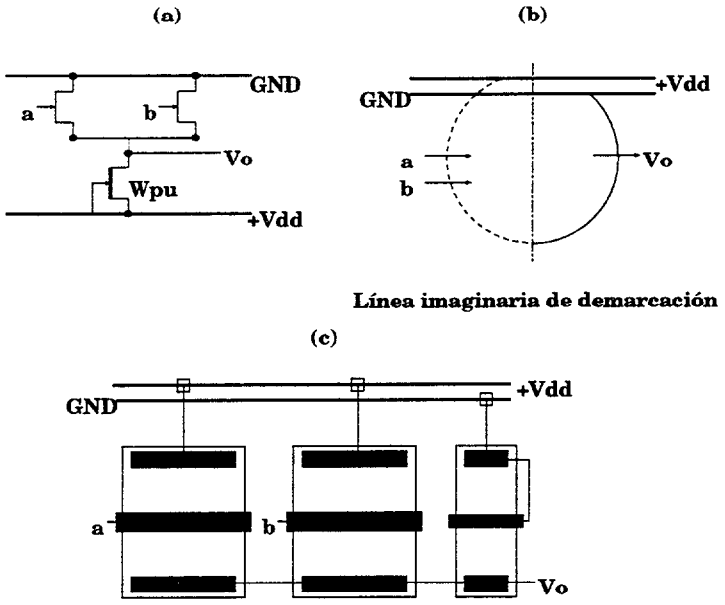


FIGURA 3.10: Representaciones de una puerta OR. (a) Circuito. (b) Notación en Anillo. (c) Notación Simbólica.

distribución de los buses de alimentación local.

Cuando la tecnología permite hasta tres niveles de metalización, como es el caso de **Vitesse**, los buses de alimentación local se dimensionan al mínimo. Los planos en metal de nivel 3 distribuyen la alimentación a nivel global, tal y como se muestra en la figura 3.12. Los fabricantes recomiendan que se utilice este tercer nivel de metal sólo para distribuir las alimentaciones.

En algunas tecnologías GaAs con transistores HEMT, como la que distribuye **IAF**, la metalización a nivel de puerta del transistor es diferente para el transistor de empobrecimiento y el transistor de enriquecimiento. Por otra parte, los anchos de los transistores de empobrecimiento y de enriquecimiento son de igual orden de magnitud. Ambas razones expuestas, conllevan a que la puerta del transistor debe orientarse perpendicular a los buses, tal y como se muestra en la figura 3.13. Las interconexiones entre los metales de puerta y el primer nivel de metalización es directa, utilizándose el canal entre lógica para ruteado de señales.

3.2.2 Evaluación de la metodología propuesta.

En este apartado haremos una evaluación de la metodología de diseño propuesta. En primer lugar se presentan los resultados de la comparación entre diseños realizados con la metodología tipo nMOS, y con la de **Notación en Anillo**. Posteriormente se realiza un estudio de inductancias series producidas por las líneas de alimentación. La bondad de la metodología se medirá no sólo en términos de las prestaciones logradas en los diseños realizados, sino también respecto a la facilidad de trazado.

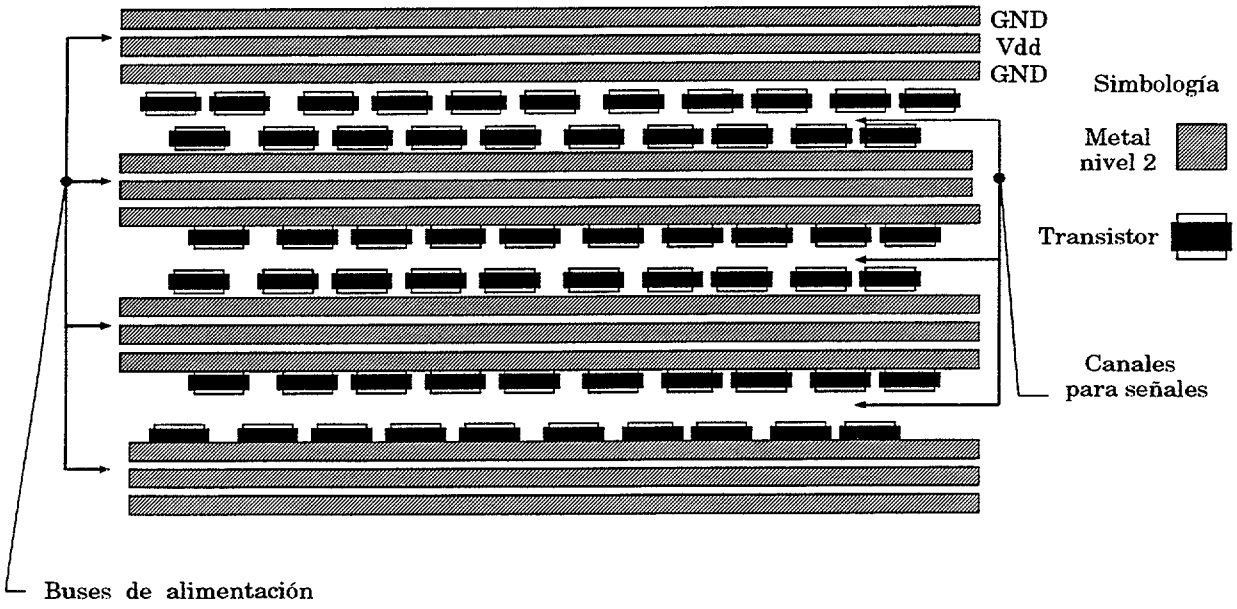


FIGURA 3.11: Estructuración del trazado en tecnologías de hasta dos niveles de metalización.

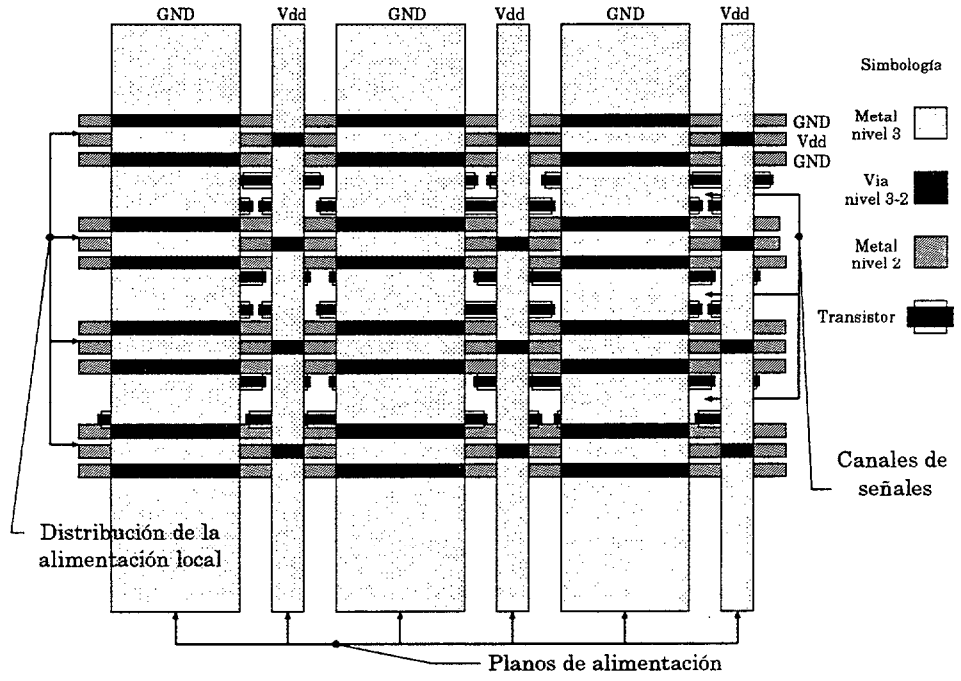


FIGURA 3.12: Estructuración del trazado en tecnologías de hasta tres niveles de metalización.

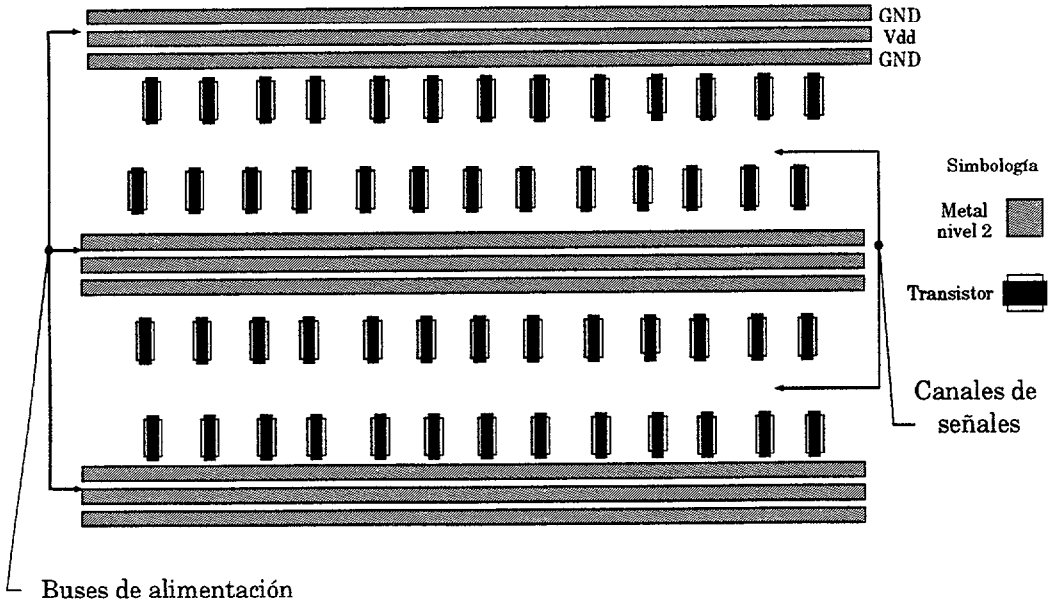


FIGURA 3.13: Estructuración del trazado en tecnologías GaAs HEMT, con distintos niveles de metalización de puerta.

3.2.2.1 Estudio de prestaciones.

Aunque hay muchas opciones para comparar las prestaciones de los diferentes circuitos y estilos de trazado, se propone una **Figura de Mérito** [84, 86, 32] o índice de prestaciones – $P_{\text{ind.}}$ – en términos del número de puertas, la máxima frecuencia del circuito, y el área utilizada (véase ecuación 3.1).

$$P_{\text{ind.}} = \frac{N_{\text{gates}} \times \hat{f}_{\text{chip}}}{A_{\text{chip}}} \tag{3.1}$$

siendo:

- $\hat{f}_{\text{chip}}$ la máxima frecuencia del circuito,
- A_{chip} el área utilizada.

Las unidades de este índice de prestaciones nos dan una medida de la densidad computacional en $\frac{puertas-GHz}{mm^2}$.

Utilizando los estilos representados en la figura 3.7, hemos diseñado un conjunto de sumadores de 8 y 16 bit y multiplicadores de 8×8 bit, con el fin de comparar sus prestaciones.

En la tabla 3.1 se presentan los resultados de algunos sumadores de acarreo serie, que han sido optimizados para bajo consumo de potencia y área. La tecnología es de

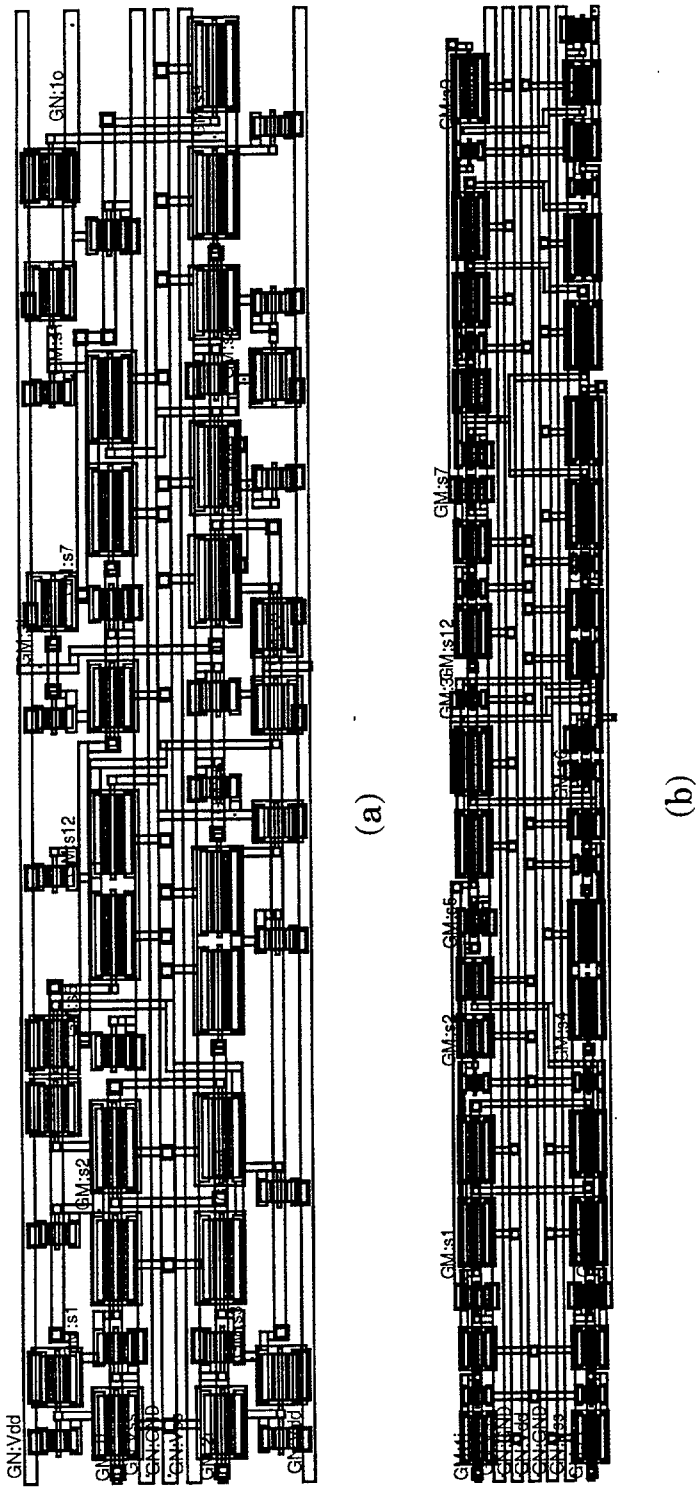


FIGURA 3.14: Trazado geométrico del sumador serie de un bit utilizando: (a) diseño tradicional y (b) Notación en Anillo.

TriQuint y la familia lógica es SDCFL. Los resultados han sido obtenidos sin tener en cuenta los efectos inductivos derivados del estilo de trazado, pero considerando las capacidades a tierra y entre líneas. A pesar de ello, se muestra que el estilo de **Notación en Anillo** genera circuitos más rápidos que los generados con el estilo tipo **nMOS**, principalmente debido a una menor longitud en las rutas de interconexión. Por tanto, se aprecia que el retardo en que se incurre por el efecto capacitivo (CC en la tabla 3.1) es menor en estilo de trazado de **Notación en Anillo** y además, la disposición de los buses de alimentación es más adecuada tanto para la reducción de las inducciones como para el efecto de apantallamiento que se produce entre lógica. En la figura 3.14 se muestran los trazados geométricos de las realizaciones en: (a) estilo nMOS y (b) estilo Notación en Anillo. El trazado al estilo nMOS (figura 3.14(a)) es más irregular, debido a que la disposición de los buses obliga a una distribución no lineal de la lógica. Esta irregularidad complica el proceso de automatización del diseño. Sin embargo, la disposición de los buses de alimentación en la figura 3.14(b) y la regularidad en la disposición de la lógica, facilita el trazado geométrico y la distribución de la alimentación. En ambas implementaciones hemos utilizado tres niveles de alimentación (Vdd, GND y Vss).

Estilo de Trazado	bits	Retardo (ns) PC	CC	Consumo (mW)	Área (mm^2)	P_{ind} $\frac{puertas--GHz}{mm^2}$
Tipo nMOS	8	3.9	4.6	48	0.116	195
	16	8.0	9.4	98	0.233	95
Notación	8	3.7	4.3	48	0.141	171
en Anillo	16	7.4	8.6	98	0.282	86

TABLA 3.1: Parámetros de algunos sumadores de acarreo serie. PC simulación de peor caso. CC simulación teniendo en cuenta el *crosstalk*

Los mismos resultados se obtienen en el diseño de multiplicadores [86]. Nuestro laboratorio ha estado desarrollando y evaluando la metodología de **Notación en Anillo**, mediante el diseño y fabricación de varios circuitos integrados en GaAs [65, 64, 3, 4]. En la figura 3.15 [65], se presenta la implementación de un multiplicador usando el algoritmo de **Booth** y un sumador de **Brent & Kung**, en tecnología H-GaAsII de Thomson CSF y lógica DCFL.

Utilizando un sumador de **Brent & Kung** en la etapa final se pueden multiplicar dos operandos de 8 bit en $5.4ns$. Si se compara esta realización con la implementación del multiplicador de **Booth** realizado por Sing et al. [91] se observa que éste último tiene un retardo de $3.2ns$, con un consumo de potencia de $510mW$ y un área de $2.18mm^2$. El índice de prestaciones P_{ind} de la realización de Sing es de $101\ \frac{puertas-GHz}{mm^2}$. Comparando consumo de potencia y área, nuestro diseño demanda menos de la mitad de potencia y área. Para nuestra realización el consumo de potencia es de $267mW$, el área utilizada de $1.82mm^2$ y el índice de prestaciones de

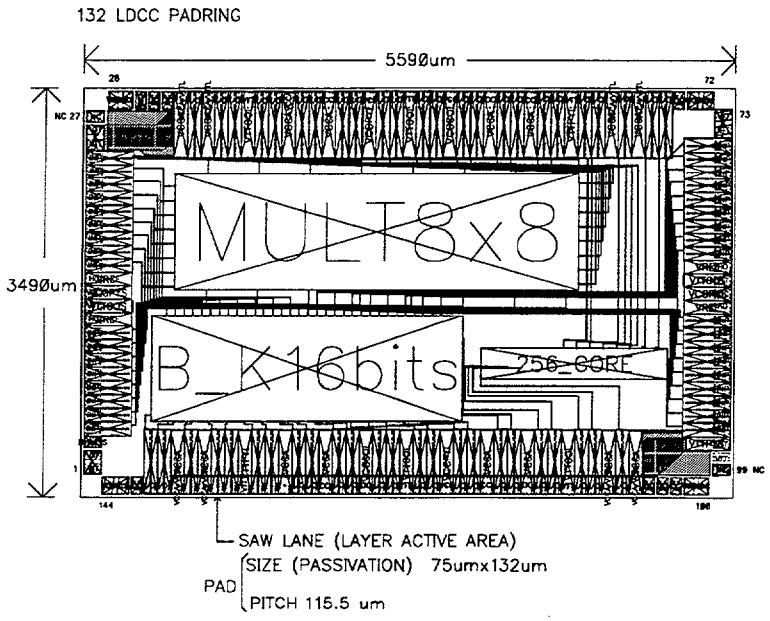


FIGURA 3.15: Circuito integrado incluido en el proyecto multichip digital organizado por el CMP en Enero de 1993, con tecnología H-GaAsII de Thomson CSF. Comprende un sumador de **Brent & Kung**, un multiplicador de **Booth**, y una memoria ROM.

Tipo	Retardo (ns)	Consumo (mW)	Área (mm^2)	P_{ind}
Booth/B & K	5.4	267	1.820	126
Booth [91]	3.2	510	2.180	101

B & K ≡ **Brent & Kung**.

TABLA 3.2: Parámetros de algunos multiplicadores de arquitectura paralela.

126 $\frac{puertas-GHz}{mm^2}$, tal y como se muestra en la tabla 3.2. Concluimos esta evaluación subrayando que se han diseñado algunas estructuras básicas de computación utilizando una metodología de diseño *full-custom* denominada **Notación en Anillo**. Los resultados muestran que el estilo de trazado en el diseño en GaAs tiene una fuerte influencia en las prestaciones del circuito. En particular el estilo de diseño propuesto, permite al diseñador optimizar el trazado de acuerdo a las prestaciones velocidad–área–tiempo, obteniendo una disposición más regular en el espacio.

3.2.2.2 Reducción de las inductancias serie.

El segundo estudio se refiere a la optimización, incorporada en la metodología, de las inductancias serie del trazado de interconexiones y alimentación. Hay muchas opciones relativas a la disposición de las tiras de alimentación a nivel global y local. Si

la inductancia no fuera importante, los buses estarían colocados de forma arbitraria. Sin embargo, los modelos existentes de línea de transmisión aconsejan la forma de realizar un trazado de baja inductancia. Para ello, comparamos las distribuciones mostradas en las figuras 3.16, 3.17 y 3.18.

En la primera disposición 3.16, las tiras de alimentación se presentan alternadas y ampliamente separadas. El modelo ideal de guía onda coplanar [36, 37], puede

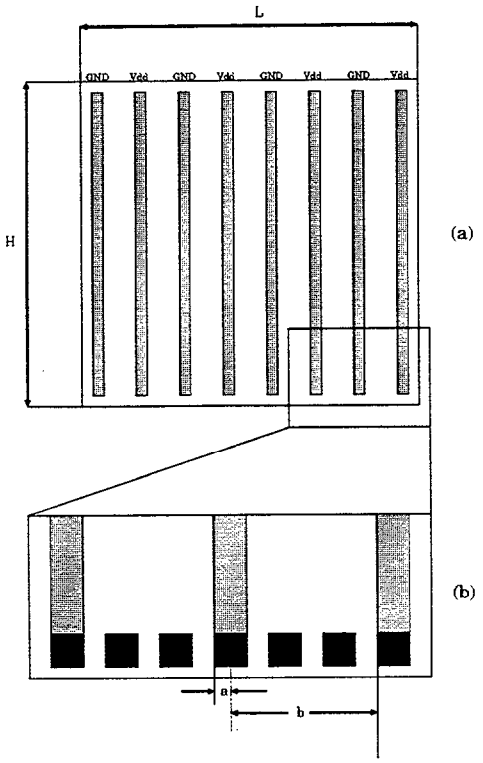


FIGURA 3.16: Distribución de la alimentación alternada y ampliamente espaciadas.(a) Vista de un chip con V_{DD} y GND distribuidas de forma alternada, ampliamente espaciadas. (b) Vista ampliada de la esquina derecha.

aplicarse a esta disposición, tendiendo a sobrestimar la capacidad total de la línea y subestimar la impedancia característica; por tanto, llevará a una estimación conservativa, aunque se desprecie el acoplamiento línea–línea (*crosstalk*). Para esta geometría las ecuaciones son las mostradas a continuación.

$$Z_0 = \frac{30\Pi}{\sqrt{\xi \mathrm{eff}}} \frac{K(k')}{K(k)} \tag{3.2}$$

$$\xi \mathrm{eff} = 1 + \frac{\xi \mathrm{r} - 1}{2} \frac{K(k')}{K(k)} \frac{K(k_1)}{K(k_1')} \tag{3.3}$$

siendo:

$$k = \frac{a}{b}$$

$$k_1 = \frac{sh\ (\frac{\Pi a}{2h})}{sh\ (\frac{\Pi b}{2h})}$$

h es la profundidad del substrato (3.4)

Es muy usual la situación $h \gg b > a$, $sh\ x \approx x$ y $k_1 \simeq \frac{a}{b}$. Por tanto, la ecuación de la ξ_{eff} se reduce a:

$$\xi\text{eff} = 1 + \frac{\xi\text{r} - 1}{2} \tag{3.5}$$

donde ξ_r es la constante dieléctrica relativa del GaAs.

En estas ecuaciones, $K(k)$ es una integral elíptica completa de primera clase y $K'(k) = K(k')$ es la función complementaria. k' se define como:

$$k' = \sqrt{1 - k^2} \tag{3.6}$$

En [47] se presenta una aproximación para la función elíptica que se reproduce a continuación.

$$\begin{aligned} \frac{K(k)}{K'(k)} &= \left[\frac{1}{\Pi}\ln\left(2\frac{1+\sqrt{k'}}{1-\sqrt{k'}}\right)\right]^{-1} \\ 0.707 &\leq k \leq 1 \\ \frac{K(k)}{K'(k)} &= \frac{1}{\Pi}\ln\left(2\frac{1+\sqrt{k}}{1-\sqrt{k}}\right) \\ 0.000 &\leq k \leq 0.707 \end{aligned} \tag{3.7}$$

Desarrollando un sencillo ejemplo numérico, si $a = 50\mu m$ (medio ancho de un *pad* de conexión) y $b = 383\mu m$ (dos *pads* y medio, además de tener en cuenta reglas de separación mínima entre metales) y consideramos que no hay dieléctrico sobre el nivel de metal considerado ($\xi_\text{r} = 1$) se deduce una $Z_0 = 205\Omega$ y una $L_0 = 6.8\frac{nH}{cm}$.

En la disposicion de líneas de alimentación en paralelo próximas, agrupadas en parejas ampliamente espaciadas (al menos $10b$ de separación entre las tiras más próximas), tal y como se muestra en la figura 3.17, se puede aplicar las expresiones deducidas del modelo de tira coplanar [36]:

$$\begin{aligned} L_0 &= \frac{Z_0\sqrt{\xi_{\text{eff}}}}{c} \\ Z_0 &= \frac{120\Pi}{\sqrt{\xi_{\text{eff}}}}\frac{K(k)}{K(k')} \end{aligned} \tag{3.8}$$

Las ecuaciones 3.3, 3.4 y 3.5 también se aplican a este modelo.

Continuando con el ejemplo de cálculo propuesto, si $a = 2.0\mu m$ (regla de separación mínima) y $b = 102\mu m$ (dos *pads* y la separación mínima) se deduce una $Z_0 = 106\Omega$ para $\xi_\text{r} = 1$ y una $L_0 = 3.5\frac{nH}{cm}$, casi dos veces inferior que en la

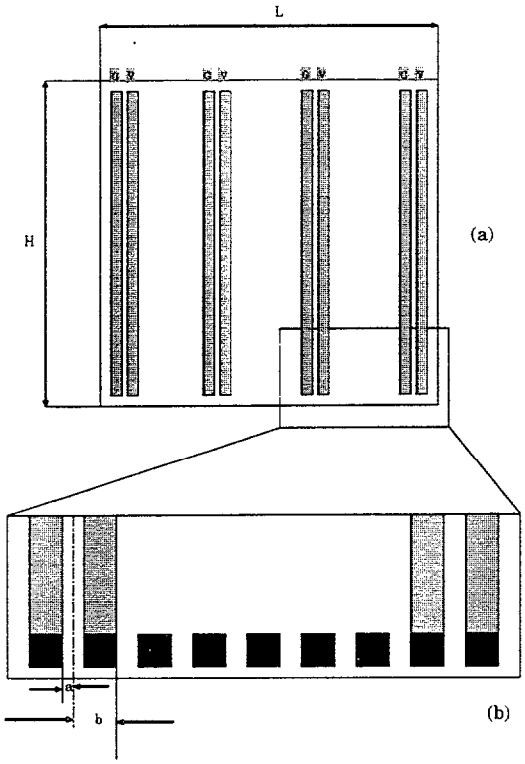


FIGURA 3.17: Distribución de la alimentación alternada y agrupada en parejas. (a) Vista de un chip con V_{DD} y GND agrupadas en parejas.(b) Vista ampliada de la esquina derecha.

primera disposición donde las tiras estaban ampliamente espaciadas. Por tanto es ventajoso agrupar los conductores de alimentación en tiras paralelas lo más próximas posibles y alternando las tensiones de alimentación para de esta forma reducir las inductancias serie.

En la última disposición 3.18, las tiras de alimentación se presentan alternadas y próximas.

El modelo ideal de guía onda coplanar [36], sigue siendo totalmente válido en esta disposición.

Continuando con el ejemplo numérico, sea $a = 50\mu m$ y $b = 52.0\mu m$, se obtiene una $Z_0 = 58\Omega$ para $\xi_r = 1$ y una $L_0 = 1.2\frac{nH}{cm}$, casi tres veces inferior que en la segunda disposición donde las tiras estaban agrupadas en parejas ampliamente espaciadas.

Comparando analíticamente los modelos de guía onda (CPW) y de tira coplanar (CPS), se deduce la siguiente interrelación 3.9:

$$Z_{0,\mathrm{CPW}} = 4 \times \frac{K(k_{\mathrm{CPS}})}{K'(k_{\mathrm{CPS}})} \times \frac{K(k_{\mathrm{CPW}})}{K'(k_{\mathrm{CPW}})} \times Z_{0,\mathrm{CPS}} \tag{3.9}$$

A partir de la figura 3.19, que representa las dimensiones que forman parte del

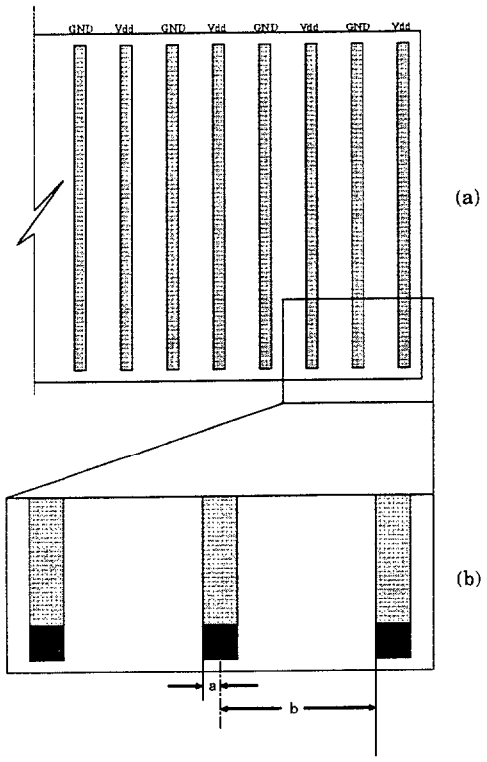


FIGURA 3.18: Distribución de la alimentación alternada y próximas. (a) Vista de un chip con V_{DD} y GND distribuidas de forma alternada.(b) Vista ampliada de la esquina derecha.

cálculo de las constantes k_{CPW} y k_{CPS} se obtiene la siguiente relación 3.10:

$$k_{CPW} = \frac{(1 - k_{CPS})}{(21 - k_{CPS})} \tag{3.10}$$

de forma tal, que en la figura 3.20 se puede observar que dependiendo del factor $k = \frac{a}{b}$ y por tanto de la separación entre líneas de alimentación, dado un ancho constante de las mismas:

- debe predominar la geometría CPS cuando los buses están ampliamente separados ($\frac{a'}{b'} < 0.5$),
- o CPW cuando la distancia entre buses empieza a disminuir ($\frac{a'}{b'} > 0.5$) acercándose al mínimo de regla de diseño.

Es decir, si las líneas de alimentación han de estar separadas ampliamente, es muy beneficioso agruparlas alternándolas. Sin embargo, si éstas pueden ir próximas se prefiere que lo hagan alternadas, cuanto más próximas mejor.

Lo anterior corrobora el hecho de que las líneas de alimentación trazadas en un nivel de metal 3, al poder ir tan cerca como las reglas de fabricación permitan,

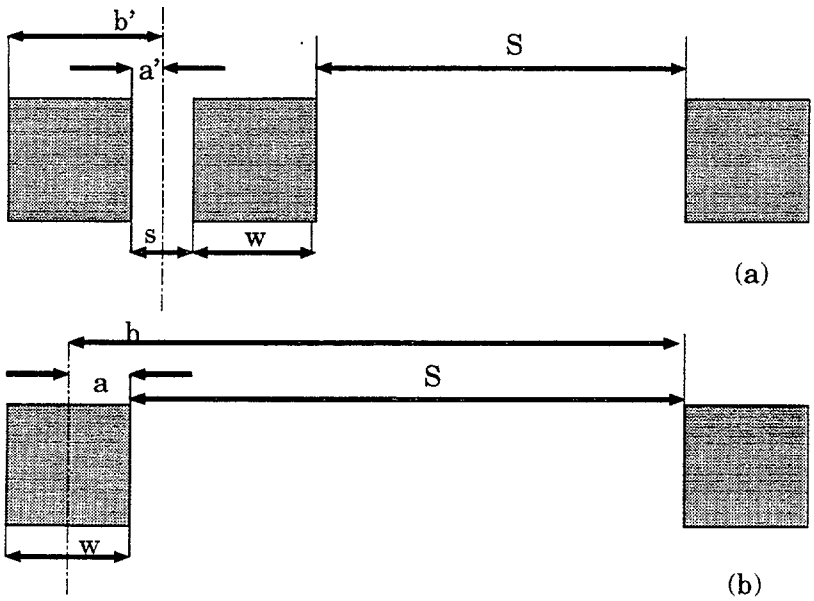


FIGURA 3.19: Dimensiones para el cálculo de (a) k_{CPS} y (b) k_{CPW}. S expresa la separación entre buses y s la regla de separación mínima. W es al ancho de una línea.

irán alternadas y con separación mínima (geometría CPW). El mínimo número de líneas requeridas de este tercer nivel de metal de un ancho determinado se calcula considerando, en el peor caso, la caída de tensión y la limitación en la densidad de corriente de electromigración ($\hat{J} = 2\frac{mA}{\mu m^2}$ para el aluminio).

Respecto al siguiente nivel de metal inferior (metal de nivel 2) todo lo anterior es igualmente aplicable. Sin embargo, dado que en este segundo nivel influyen los canales de ruteado de señales y el colocado de lógica, de forma que tienden a separar ampliamente los buses de alimentación, la mejor disposición es la agrupación y la separación de los grupos formados (geometría CPS). Para corroborar lo anterior, desarrollamos los siguientes ejemplos numéricos.

En la figura 3.21, mostramos la disposición para el trazado geométrico de acuerdo con el estilo nMOS, y en la figura 3.22 la disposición para la composición geométrica según la metodología propuesta. Para simplificar, no consideramos el ancho del canal de interconexión y el ancho de alimentación W lo establecemos a $5\mu m$. La regla de separación mínima entre metales de nivel 1 es de $2\mu m$. Los anchos de las puertas, dispuestas de forma horizontal, son variables. Los altos de las puertas varían con el ancho del canal, pero consideraremos que se mantienen a $10\mu m$.

Aplicando el modelo gemétrico CPW (figura 3.21), deducimos que $a = 2.5\mu m$ y $b = 22.5\mu m$. Dado que la relación $\frac{a}{b} = 0.10$, y aplicando las expresiones 3.7 y 3.2 considerando que no hay dieléctrico sobre este segundo nivel de metal, se deduce una $Z_{0,CPW} = 220\Omega$.

Si aplicamos el modelo geométrico CPS (figura 3.22), deducimos que $a' = 1\mu m$

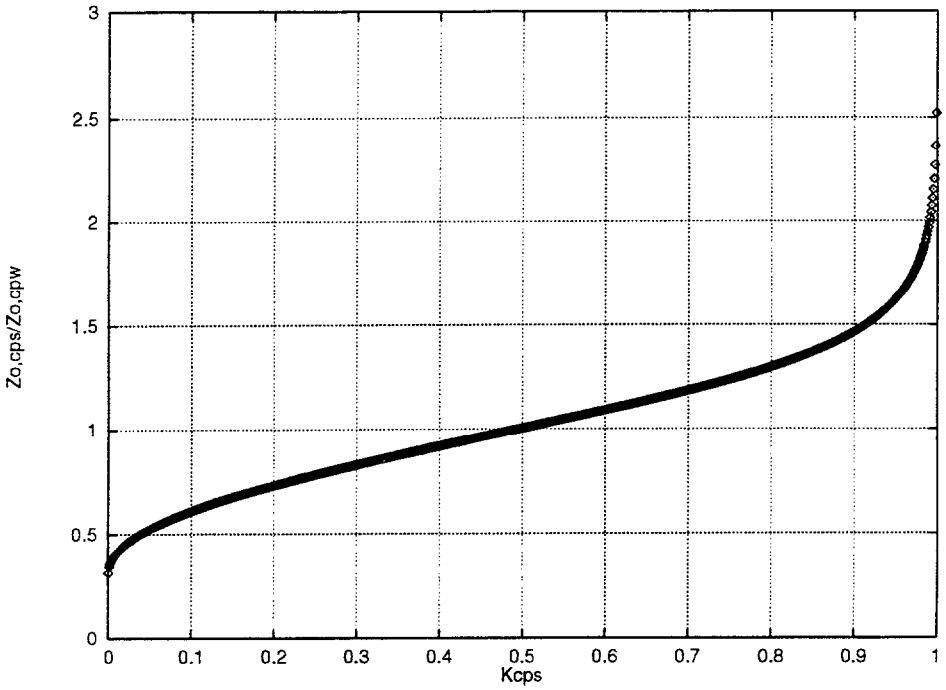


FIGURA 3.20: Relación entre impedancias series para los modelos CPW y CPS y el factor geométrico k_{CPS}.

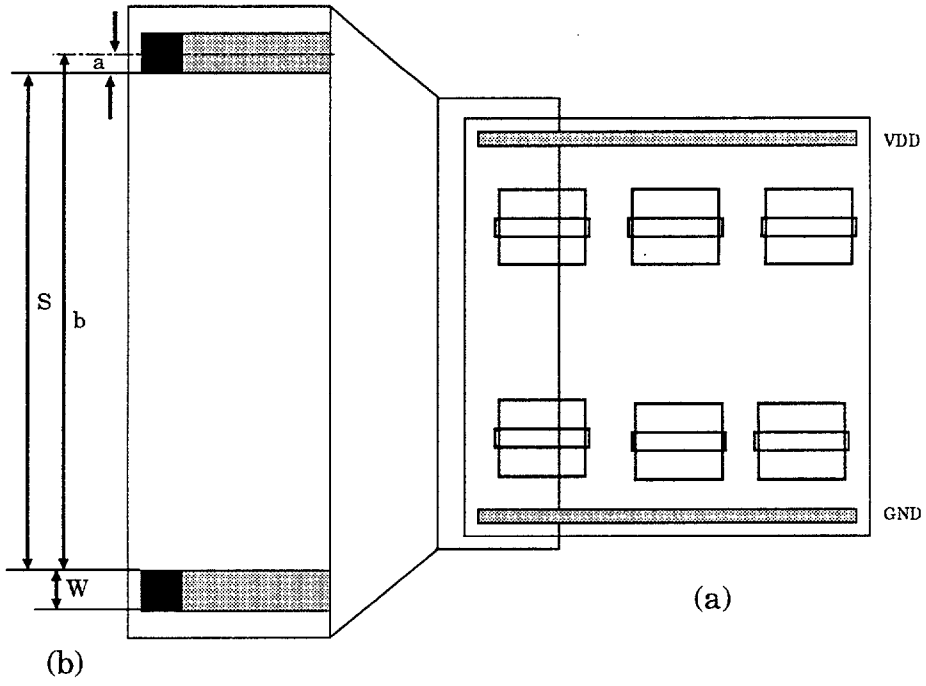


FIGURA 3.21: Disposición de buses de alimentación local según el estilo nMOS: (a) Vista global, (b) detalle lateral.

y $b' = 6\mu m$, siendo la relación $\frac{a'}{b'} = 0.17$. Al aplicar las expresiones 3.7 y 3.8, considerando también que no hay dieléctrico sobre este segundo nivel de metal, se deduce una $Z_{0,CPS} = 187\Omega$. Lo anterior se ha calculado sin tener en cuenta el canal de interconexión. Si aumentamos en $10\mu m$ la separación entre buses para dar cabida al canal de interconexión entre lógica, observamos que $Z_{0,CPW}$ aumenta hasta 248Ω, mientras que $Z_{0,CPS}$ se mantiene constante.

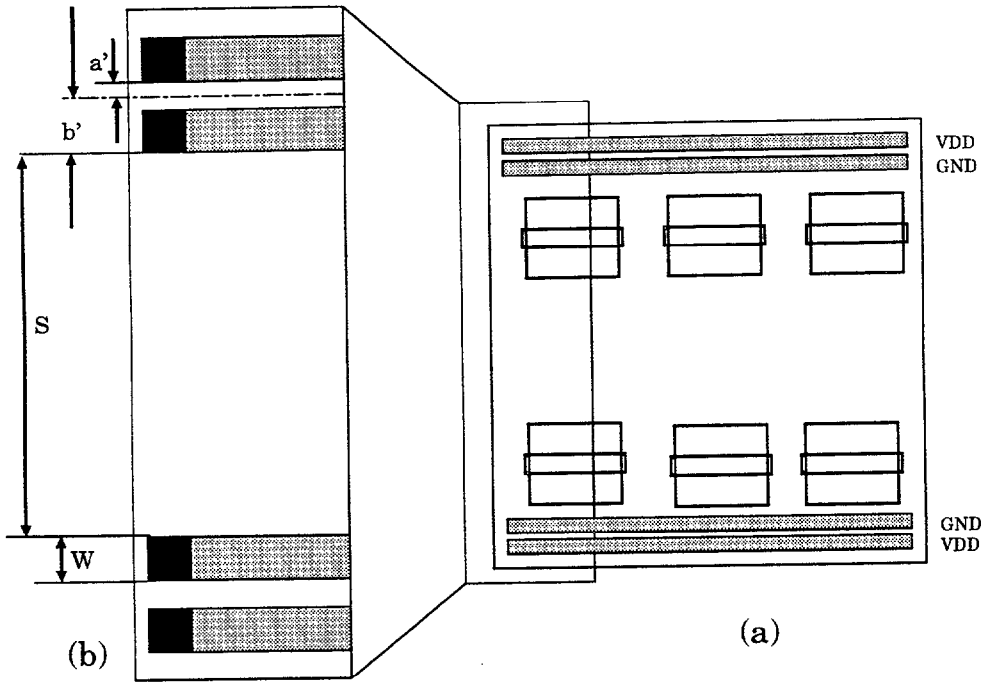


FIGURA 3.22: Disposición de buses de alimentación local según el estilo Notación en Anillo: (a) Vista global, (b) detalle lateral.

Si se disponen múltiples líneas de metal de nivel 2 perpendicular al nivel superior, como se muestra en la figura 3.23, se configura la alimentación a nivel local. Las conexiones entre los metales de nivel 3 y 2 se representan mediante el patrón de vías en la figura. El ancho requerido para los buses de alimentación dependerá del número de puertas a alimentar en cada fila entre las conexiones anteriormente mencionadas, de forma que si se distribuye las tiras de metal de nivel 3 uniformente a lo largo del todo el chip, las caídas de tensión pueden mantenerse bastante bajas en estos buses locales, con unos anchos mínimos.

En la alimentación a nivel de transistor, perpendicular al nivel inmediatamente superior, los problemas de inductancias series son despreciables debido a la corta dimensión de las tiras entre transistores y buses locales de alimentación.

3.3 Dimensionamiento de los elementos del trazado físico.

En esta sección presentamos:

- El dimensionamiento a nivel de puertas,
- el dimensionamiento de las interconexiones entre puertas,
- la estrategia de dimensionamiento de alimentación a nivel local,
- la estrategia de dimensionamiento de la alimentación a nivel global, para tecnologías de hasta tres niveles de metalización.

3.3.1 Dimensionamiento a nivel de puertas.

Cuando se habla de dimensionamiento a nivel de puerta, se refiere a la optimización de las características eléctricas de los transistores, por medio de sus variables geométricas – dimensiones –. Desde finales del noventa hasta nuestros días se ha realizado un esfuerzo considerable en optimización y análisis temporal de circuitos DCFL/SDCFL en GaAs [38, 45, 52].

El problema de la optimización eléctrica presupone un diseño óptimo desde el punto de vista del análisis circuital, tratando de mejorar sus prestaciones velocidad, área y potencia. Para abordar este fin se requieren modelos precisos orientados a estimar el retardo de propagación, el área ocupada y la potencia consumida. Las técnicas para optimizar un circuito son fundamentalmente el dimensionado y la recuperación de los márgenes de ruido. Sobre esta última técnica y con lógica SDCFL de **TriQuint** existen herramientas de optimización eléctrica como la presentada en [38], donde se hace uso del método de gradiente descendente con condiciones de contorno, con el fin de minimizar una función objetivo multidimensional. En lógica DCFL de **Vitesse** se aplicó esta estrategia para obtener el conjunto de soluciones no inferiores de una función objetivo [70]. En esta tecnología de Vitesse (H–GaAsII), y debido a la mejora del proceso tecnológico alcanzado, se está en posición de dar mayor variabilidad al estudio de relaciones entre variables geométricas y eléctricas y, en consecuencia, de obtener mayores márgenes de optimización.

En documentación generada por el fabricante **Vitesse** [104], se presenta un método de dimensionamiento robusto pero no optimizado, basado en las consideraciones de salida del circuito y de las prestaciones de cada etapa lógica.

3.3.2 Dimensionamiento de las interconexiones.

Las interconexiones entre transistores deben ser cortas, ocupando la menor área posible. Ambos objetivos se abordan haciendo uso de al menos dos niveles de metal.

Metal de puerta, para las conexiones de señales entre transitores próximos, con un predominio de trazado horizontal. Y metal de nivel 1 para las conexiones más críticas, con tendencia de trazado vertical. Debido a las intensidades de corriente que circulan por la puerta del transistor y a la corta longitud de las interconexiones, generalmente con la mínima dimensión de ancho de ambos metales se satisface con holgura el dimensionado. De todas formas, durante la fase de minimizado de vías se comprueban estas dimensiones, con el fin de asegurar el funcionamiento del circuito. Cabe la posibilidad de utilizar metal de nivel 2, para la transmisión de aquellas señales de alto *fan–out* – señales de reloj, señales de entrada –, lo que requiere un ruteador de tres metales. El proceso de dimensionamiento se realiza evaluando una reducción aceptable del margen de ruido con los requerimientos de intensidad de corriente, de igual forma que si del dimensionado de una línea de alimentación se tratase. Es decir, para cada señal de estas características, se calcula cuál debe ser el ancho de metal de nivel tres que produce una caída de tensión aceptable. En la sección 3.3.3.4 se muestra una relación útil entre las variables de interés para este cálculo.

3.3.3 Dimensionamiento de las alimentaciones.

3.3.3.1 Estrategia Global.

Si se consideran los efectos de la resistencia y de la electromigración, la estrategia global de trazado será utilizar al menos un segundo nivel de metal para la distribución de potencia a lo ancho del chip. Debido a que el segundo nivel de metal es más grueso que el primer nivel de metal, es más adecuado para suministrar mayores niveles de corriente, manteniendo un control más aceptable de la tensión con anchos menores. Si se realiza una planificación adecuada, la mayor caída de tensión tendrá lugar en esta segunda capa [92]. Si el proceso tecnológico dispusiera de una tercera capa de metal (por ejemplo, H–GaAsII de Vitesse), ésta podría realizar la misma función que realizó el segundo nivel de metal, pero suministrando intensidad al nivel inferior. De esta forma, se soportan las mayores caídas de tensión en un tercer nivel de metal, aún más adecuado para éste propósito. El incremento capacitivo que supone esta decisión es beneficioso para atenuar los efectos de la conmutación de puertas [62]. La estrategia se presenta en la figura 3.23, donde:

El primer nivel de metal se usa para la alimentación de las puertas lógicas, conectando los drenadores y surtidores de los transistores a un

segundo nivel de metal que distribuirá localmente las alimentaciones, y éste a su vez a un

tercer nivel de metal que distribuirá a nivel de chip las alimentaciones.

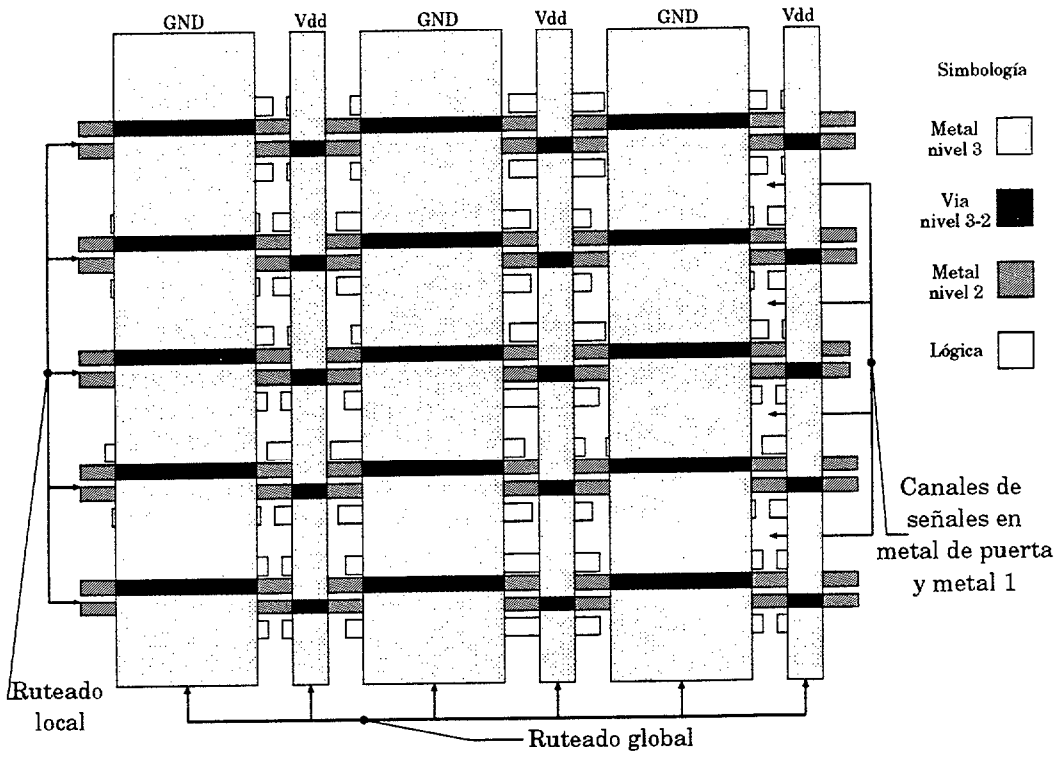


FIGURA 3.23: Estrategia global de distribución de alimentaciones, señales y lógica.

3.3.3.2 Dimensionamiento de la alimentación a nivel de transistor.

El problema que aquí se describe trata el dimensionamiento de las tiras de alimentación que van desde un transistor al bus local inmediato. Los problemas de inductancias series son despreciables debido a la corta longitud de la tira de alimentación.

De acuerdo con el trazado físico de un transistor en tecnología GaAs, con un ancho W_{txtor} y una longitud de puerta L_{txtor} como caraterísticas geométricas y una intensidad de saturación $I_{\text{sat.}}$ como parámetro eléctrico, el ancho de la tira de alimentación que va desde el terminal fuente del transistor al bus local tiene una longitud L_{t}, determinándose el ancho de la tira W_{t} según la expresión 3.11,

$$\hat{I}_{\text{t}} = I_{\text{t,lc}}(W_{\text{t}} - \Delta W_{\text{t}}) \tag{3.11}$$

donde:

- $\hat{I}_{\text{t}}$ es la máxima corriente que pasa por la tira, y que es equivalente a la de saturación del transistor,
- $I_{\text{t,lc}}$ es la densidad de corriente límite de la tira de metal, medida en $\frac{mA}{\mu m}$,

- W_t es el ancho de la tira y,

- ΔW_t es un ancho de corrección que la mayoría de los fabricantes distribuyen como consecuencia del error cometido cuando se procesan las máscaras.

La existencia de una vía en la conexión entre el terminal del transistor y la tira de metal de conexión al bus local, corrige la anchura anterior de acuerdo con las reglas de diseño. Quedando la siguiente relación:

$$W_t' = W_t - 2 \times E\rceil_{\text{via1}}^{\text{met1}} \tag{3.12}$$

donde:

- $E\rceil_{\text{via1}}^{\text{met1}}$ es el solapamiento mínimo que ha de existir entre el metal y la vía

En la tabla 3.3 se presentan las densidades de corriente máxima para los metales implicados en la conexión transistor metal de alimentación y metal de alimentación local, para la tecnología H–GaAsII.

Capa	$R_S(\frac{\Omega}{\square})$	Límite de corriente I_{lc} DC	AC	Pico	$\Delta W(\mu m)$
Metal de Puerta	$0.5 - 1.5$	5.0	5.0	25.0	0.4
Implantación (n+) Óhmica	$190 - 230$	1.0	1.0	2.0	0.0
Metal Óhmico	< 10.0	0.3	0.3	0.6	0.0
Metal 1	< 0.070	1.0	1.0	5.0	0.2
Metal 2	< 0.035	2.0	2.0	10.0	0.0
Metal 3	< 0.025	2.8	2.8	14.0	0.0
Metal 4	< 0.025	2.8	2.8	14.0	0.0

TABLA 3.3: Resistencia y límites de corrientes de las máscaras de interconexión ($T = 85°C$).

De entre estos metales, el que dispone de una mínima densidad de corriente es el óhmico, siendo la relación entre anchos de éste metal y el metal de conexión de más de tres veces para la misma intensidad. El utilizar el ancho de metal igual al ancho del transistor permite que la longitud de alimentación sea de al menos 10 veces la mínima, tal y como se muestra en la tabla 3.4. Los datos están calculados a partir de tablas de fabricantes [104]. Son igualmente aplicables a tecnologías GaAs tanto comerciales como militares.

Tamaño txtor $(\frac{W}{L})$ $(\frac{\mu m}{\mu m})$	$T = 70°C$		$T = 125°C$	
	$I_{DS}(\mu A)$	$L_t(\mu m)$	$I_{DS}(\mu A)$	$L_t(\mu m)$
$\frac{3}{10}$	43	997	49	875
$\frac{3}{8}$	53	808	60	714
$\frac{3}{6}$	70	612	78	549
$\frac{3}{4}$	101	424	113	379
$\frac{3}{3}$	167	256	181	236
$\frac{3.5}{2.3}$	201	248	217	230
$\frac{6}{2.3}$	369	134	400	214
$\frac{8}{2.3}$	504	226	546	209
$\frac{10}{2.3}$	639	223	693	206
$\frac{15}{2.3}$	976	219	1058	202
$\frac{20}{2.3}$	1313	217	1423	200

TABLA 3.4: Longitud de tira de metal de nivel 1 para producir una caída de tensión de menos de $1mV$.

3.3.3.3 Dimensionamiento de las alimentaciones a nivel local.

Bajo el estilo de trazado propuesto, donde la lógica se dispone paralela a los buses locales, se estudiarán los requerimientos de alimentación de este segundo nivel de metal, considerando únicamente los efectos resistivos y electromigratorios, ya que la reducción de inductancia se consigue en base a la disposición alternada y agrupada de las alimentaciones.

Si no se utiliza el nivel de metal 3 en la alimentación, la tira horizontal de metal de nivel 2 tiene que soportar toda la caída de tensión según se expresa en la ecuación 3.13:

$$\Delta V = \frac{\rho L_t I_{power}}{2 W_t t_t} \tag{3.13}$$

donde :

- I_{power} es la intensidad que debe suministrarse en cada extremo,
- L_t es la semi–longitud de la tira de alimentación,
- ρ es la resistividad del metal,
- W_t es el ancho de la tira de metal, y
- t_t es el grueso de la tira.

Generalmente, la expresión 3.13 se implementa en un entorno de compilación durante la fase de ruteado de alimentación, para determinar el ancho de la líneas.

Si se utiliza el nivel de metal 3 para distribuir la alimentación a nivel de chip, entonces los condicionantes que rodean a las dimensiones del nivel 2 de metal para distribuir la alimentación localmente quedan muy relajados. De hecho la longitud máxima de las tiras en este nivel 2 es el ancho de la tira de alimentación en nivel 3 – teniendo en cuenta las reglas de diseño – tal y como muestra la figura 3.23.

El estilo de distribución aquí presentado tiene la ventaja de adecuarse a las tecnologías existentes en GaAs – TriQuint y Vitesse –. En los diseños desarrollados en el proceso tecnológico de TriQuint, se dispone sólo de dos niveles de metal, mientras que en el desarrollado por Vitesse (H–GaAsII) se dispone de hasta cuatro niveles de metalización. De aquí la importancia y adaptabilidad de las implementaciones que tengan en cuenta estas diferencias entre fabricantes.

3.3.3.4 Dimensionamiento de las alimentaciones a nivel global.

Suponiendo que la alimentación a nivel de módulo está dispuesta en tiras de metal de nivel 3, y además éstas se distribuyen perpendiculares a las tiras de alimentación en metal de nivel 2, tal y como se muestra en la figura 3.18, aunque la densidad máxima de corriente permita reducir el ancho a un mínimo, ésto no es lo adecuado para producir una caída de tensión mínima. Con una distribución uniforme de la corriente sobre la totalidad del chip, y suponiendo que en ambos extremos se suministra alimentación, la caída de tensión mayor tiene lugar en el centro del chip. Para determinar esta caída de tensión se utiliza la expresión 3.13

De entre ese conjunto de variables en una tecnología determinada, y sobre un trazado específico, son datos: ΔV, ρ, L_t, y la intensidad a suministrar I_{power}, siendo consecuentemente W_t la incógnita. Es posible llegar a una relación entre la intensidad a suministrar en cada extremo de la tira y el ancho de la misma a partir de las consideraciones del trazado. Dada una célula de alto H_{chip} y ancho W_{chip} – relación de aspecto r_{aspecto} –, que consuma una corriente máxima I_{chip} en una tecnología de hasta tres niveles de metal (H–GaAsII de Vitesse), con la distribución uniforme y alternada de alimentación (véase figura 3.23), la expresión que relaciona el ancho de la tira de metal de nivel 3 (W_t) y las cantidades mencionadas es:

$$\Delta V = \frac{\rho H_{\text{chip}} I_{\text{chip}}}{2W_t} \tag{3.14}$$

El W_t obtenido se ha de dividir entre un número de planos para suministrar a todas las secciones del chip las alimentaciones. Para atenuar el efecto de realimentación en la conmutación de las puertas generalmente se suele multiplicar por un factor de 2 a 3 el ancho de los planos de alimentación con tensión más negativa.

Con ello han quedado completamente definidas la notación, topología, dimensionado y expresiones de la metodología presentada en este capítulo.

Capítulo 4

Análisis de entornos de diseño VLSI y su adecuación a herramientas para GaAs.

El alto costo derivado del diseño de un CI complejo se debe al gran volumen de detalle que debe manejarse. El diseño de un CI conlleva la especificación de las características lógicas y físicas de un gran número de subsistemas funcionales relacionados unos con otros. Para cada uno de ellos, el diseñador debe desarrollar la circuitería asociada, teniendo en cuenta las características peculiares de la tecnología, de forma que la realización física satisfaga todas las restricciones en prestaciones, potencia, área y velocidad. Para manipular la complejidad asociada con el diseño VLSI se han desarrollado metodologías basadas en jerarquías, maximizando la regularidad en las estructuras de trazado. Sin embargo, todos los esfuerzos se han orientado hacia la aproximación de células estándares y se han desarrollado pocos entornos para el desarrollo de módulos *full–custom*. Desde el punto de vista de la síntesis de alto nivel son importantes los progresos conseguidos, pero a la hora de implementar un CI se hace uso extensivo de la aproximación de células estándares. En tecnologías de alta velocidad, como es el caso que nos ocupa, esta aproximación no es válida como se ha indicado, porque sus costos obligan a un diseño optimizado en prestaciones de velocidad y área, y estos objetivos no se alcanzan por las restricciones de la metodología de células estándares y por la poca variedad y flexibilidad de las primitivas en GaAs. Analizamos por ello las opciones disponibles en entornos de diseño para desarrollar en *full–custom* células, módulos y bloques circuitales completos.

4.1 Discusión del desarrollo de la síntesis de células sobre entornos comerciales y universitarios.

A pesar de la proliferación de entornos de diseño en estos últimos años, bajo este apartado nos limitaremos a analizar dos sistemas comerciales distribuidos por EUROCHIP: **Cadence/Edge** y **Mentor Graphics**. Como aportación presentamos una adaptación del primero para el diseño en GaAs. De los entornos universitarios, hemos tomado de referencia a **OCTTOOLS**.

4.1.1 Cadence/Edge

El sistema Cadence/Edge ofrece una gran variedad de herramientas programables por el usuario. Esta programabilidad permite adaptar Cadence a las necesidades especiales de la interfaz de usuario que se desee. Las siguientes facilidades son posibilidades programables por el usuario:

- definición de nuevos comandos,
- acceso y modificación de la base de datos del diseño,
- realización de simulaciones,
- mantenimiento de las librerías de diseño.

Las interfaces a muchas herramientas del entorno de diseño **Design Framework** se basan en el lenguaje IL, el cual se desarrolló especialmente para este cometido. IL permite modificar y extender el sistema añadiendo una programabilidad al desarrollo de interfaces. IL está basado en el lenguaje de programación LISP. Sin embargo, el lenguaje IL también soporta una sintaxis más al estilo de C. Esta sintaxis al estilo C permite a los usuarios adaptarse con relativa facilidad al sistema, mientras que los programadores expertos pueden hacer uso de la potencia del LISP. Para cada herramienta de Cadence que tenga una interfaz programable por el usuario, existen funciones en IL para acceder a las posibilidades de esa herramienta.

Skill es un lenguaje procedural, interpretado y orientado a la ejecución interactiva, que permite utilizar las posibilidades que brinda la librería de funciones de trazado de geometría básica – líneas, polígonos–, explotar las posibilidades del compilador de estructuras propio del sistema – **Structure Compiler** –, extiende las posibilidades del lenguaje IL para el desarrollo de procedimientos, y es el lenguaje sobre el que se apoya el generador de módulos de Cadence – **ModuleMaker** –.

El compilador de estructuras no es más que un replicador en el espacio de un patrón definido de células. Está especialmente indicado para el desarrollo de redes de puertas típicas en núcleos de memorias, PLA y de algunos módulos muy homogéneos.

El entorno de generación de módulos combina el diseño basado en procedimientos, células prediseñadas, y herramientas de aplicación. Todo ello controlado por una interfaz desarrollada a medida de la aplicación. Las células predefinidas podrían ser trazados *full–custom* realizados en el editor de polígonos, o bien trazados simbólicos Las herramientas de aplicación típicas que se invocan dentro del generador de módulos de Cadence pueden incluir el compilador de estructuras, compactador, extractor, analizador temporal y varios simuladores.

La principal carencia del entorno de generación automática de trazado – ModuleMaker –, está en la propia generación de trazado a nivel de células. Es decir, en lo relativo a la implementación de una metodología de diseño diferente a la de células estándares. Por tanto entre las dificultades que presenta se puede detallar:

- Necesidad de aprender y adiestrarse en lenguajes nuevos, mixtos, no estándares e interpretados, los cuales son útiles para desarrollos interactivos, pero que suponen una incomodidad en la ejecución cuando la interacción con el entorno no es necesaria. Además, la funcionalidad del lenguaje para manejar ciertas estructuras – grafos, matrices – no es fácil, impidiendo implementar algoritmos típicos en el diseño VLSI.
- Las funciones que añaden programabilidad al entorno o bien son muy básicas o son muy dependientes del diseño en CMOS.
- La independencia tecnológica no está muy lograda, arriesgándose el diseñador a generar trazados que no pueden seguir la evolución de la tecnología.

4.1.1.1 Creación de un entorno de diseño *full–custom* en Cadence/Edge para tecnología GaAs

Impulsados por el uso extendido en el entorno universitario Europeo, hemos desarrollado en Cadence/Edge un entorno para trazado físico *full–custom* [17, 16] en tecnología GaAs. En este entorno creado para el diseño con tecnología de Gigabit Logic (ahora TriQuint) y H–GaAsII de Thomson CSF podemos realizar:

- la captura del diseño a nivel de esquemático y de trazado físico – trazado geométrico y simbólico –,
- el análisis del circuito,
- la verificación del trazado y
- la simulación eléctrica mediante HSPICE.

En todo momento hemos tenido en cuenta el hecho de crear un entorno que nos permita cubrir todas aquellas fases del diseño, sin abandonar la misma interfaz de usuario y en una única base de datos.

Hemos implementado las capas necesarias para la representación de las diferentes máscaras que intervienen en la composición geométrica de células y bloques, esforzándonos en mantener colores y tramados estándares (véase figura 4.1).

DESCRIPCION DE CAPAS					
Nombre	**CIF**	**Color**	**Función**	**Tramado**	**Comentarios**
Difusión	GD	Borde Amarillo	Implantac.		Area activa
nplus	GI	Borde Blanco	Implantación n+		Implantación n+ para MESFETs
Metal de puerta	GGM	Verde	Metal de Puerta		Metal para puerta e interconexiones locales cortas
Metal 1	GM1	Rojo	Metal de nivel 1		Metal para interconexiones locales
Metal 2	GM2	Azul	Metal de nivel 2		Metal para alimentación local
Metal 3	GM3	Gris	Metal de nivel 3		Metal para alimentación global
Contacto óhmico	GC	Celeste	Contacto		Contacto entre difusión y metal1
via 1	GV1	Naranja	Vía		Contacto entre metal de puerta y metal 1
via 2	GV2	Marrón	Vía		Contacto entre metal 1 y metal 2
via 3	GV3	Violeta	Vía		Contacto entre metal 2 y metal 3

FIGURA 4.1: Descripción de las máscaras utilizadas en el entorno.

Cualquier esfuerzo orientado a la automatización del diseño en nuevos entornos ha de respetar el trabajo desarrollado con anterioridad. Por esta razón, no se concibe una mejora sin la creación de conversores de formatos de intercambio de datos de trazado físico, que permitan reutilizar las células y módulos diseñados. Para ello se han desarrollado un conjunto de utilidades de conversión que se muestran en la tabla 4.1.

Con objeto de conocer el comportamiento del circuito que se diseña, se realiza una captura y dimensionado previo al trazado físico, a nivel de transistores MESFETS, que se verifica utilizando HSPICE como simulador eléctrico. Este

Conversor	Descripción
CDS2HSPICE	Cadence a lista de nodos HSPICE
CDS2STRM	Cadence a Stream
CIF2CDS	CIF estándar a Cadence
CIF2STRM	CIF estándar a Stream
STRM2CDS	Stream a Cadence
STRM2CIF	Stream a CIF estándar

TABLA 4.1: Conversores de datos desarrollados para el entorno.

esquema de partida se utilizará también para la comparación esquemático versus trazado geométrico, verificación topológica y dimensional. Para ello el diseñador dispone de un conjunto de puertas básicas parametrizadas y fácilmente configurables, incluyendo opciones por defecto. En dicha lista se incluyen puertas NOR, OR, inversores, *buffers* y otras puertas lógicas basadas en el transistor MESFET – DCFL y SDCFL –. En la figura 4.2(a) se muestra un esquema de puertas realizado con las utilidades que hemos desarrollado, donde se observa las cargas en las salidas, los estímulos en las entradas, etc. En la figura 4.2(b) aparece la lógica a nivel de transistores de una puerta NOR. Y finalmente, en la figura 4.2(c) el resultado de simular con HSPICE el esquema anterior.

En el entorno Cadence/Edge existen diferentes ayudas para la captura del trazado geométrico (véase figura 4.3). Todas ellas tienen en común, que soportan jerarquía, presentación multiventana, edición simultánea de varios circuitos con diferentes representaciones y acceso a herramientas de verificación en línea.

La comprobación de estructuras y dimensiones de los dispositivos diseñados a nivel geométrico, se realiza mediante un fichero tecnológico que hemos definido, y que el fabricante generalmente suele suministrar. La compilación del fichero tecnológico de descripción de dispositivos, genera un subconjunto de reglas de diseño que pueden ser utilizadas en la comprobación anterior. Sin embargo, la complejidad del transistor en GaAs induce reglas tecnológicas muy difíciles de implementar en el fichero tecnológico para la especificación de dispositivos. En este sentido, Cadence/Edge esta orientado a tecnología CMOS.

La extracción de dispositivos y parásitos, y de información de conectividad, a partir de la composición geométrica, se realiza con una herramienta de extracción y un fichero de formación de dispositivos y parásitos para cada tecnología. Hemos escrito diversas versiones de extractores de dispositivos y parásitos para tecnología GaAs de los fabricantes TriQuint y Thomson CSF. Este extractor tiene funciones especiales para generar representaciones a partir de la composición geométrica, que podemos resumir en:

- Reconocimiento de dispositivos activos del circuito,
- extracción y medida de parámetros de los dispositivos,

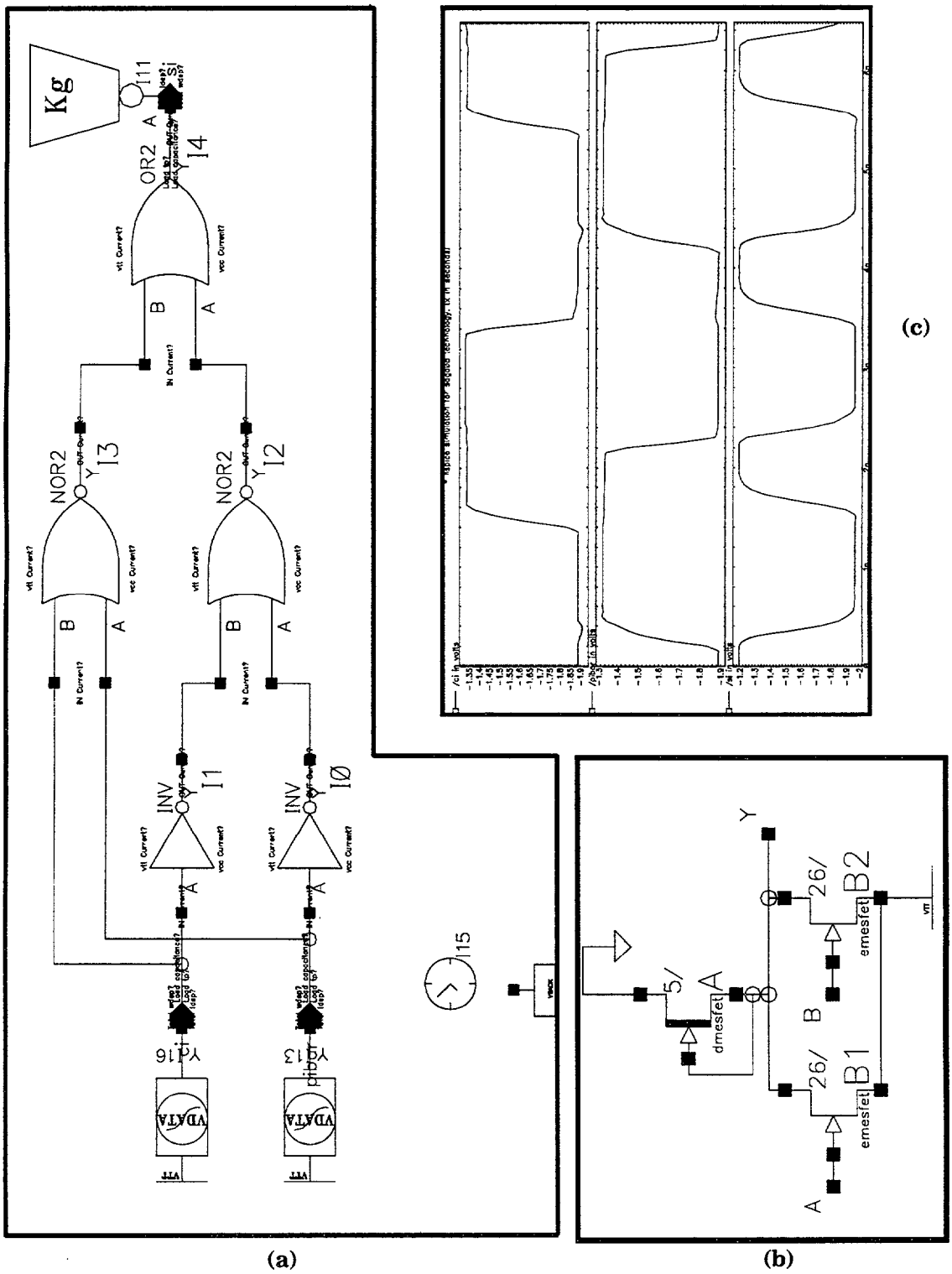


FIGURA 4.2: Captura y simulación del diseño a nivel de esquemático: (a) Esquema lógico, (b) esquema de transistores de una puerta NOR y (c) simulación con HSPICE.

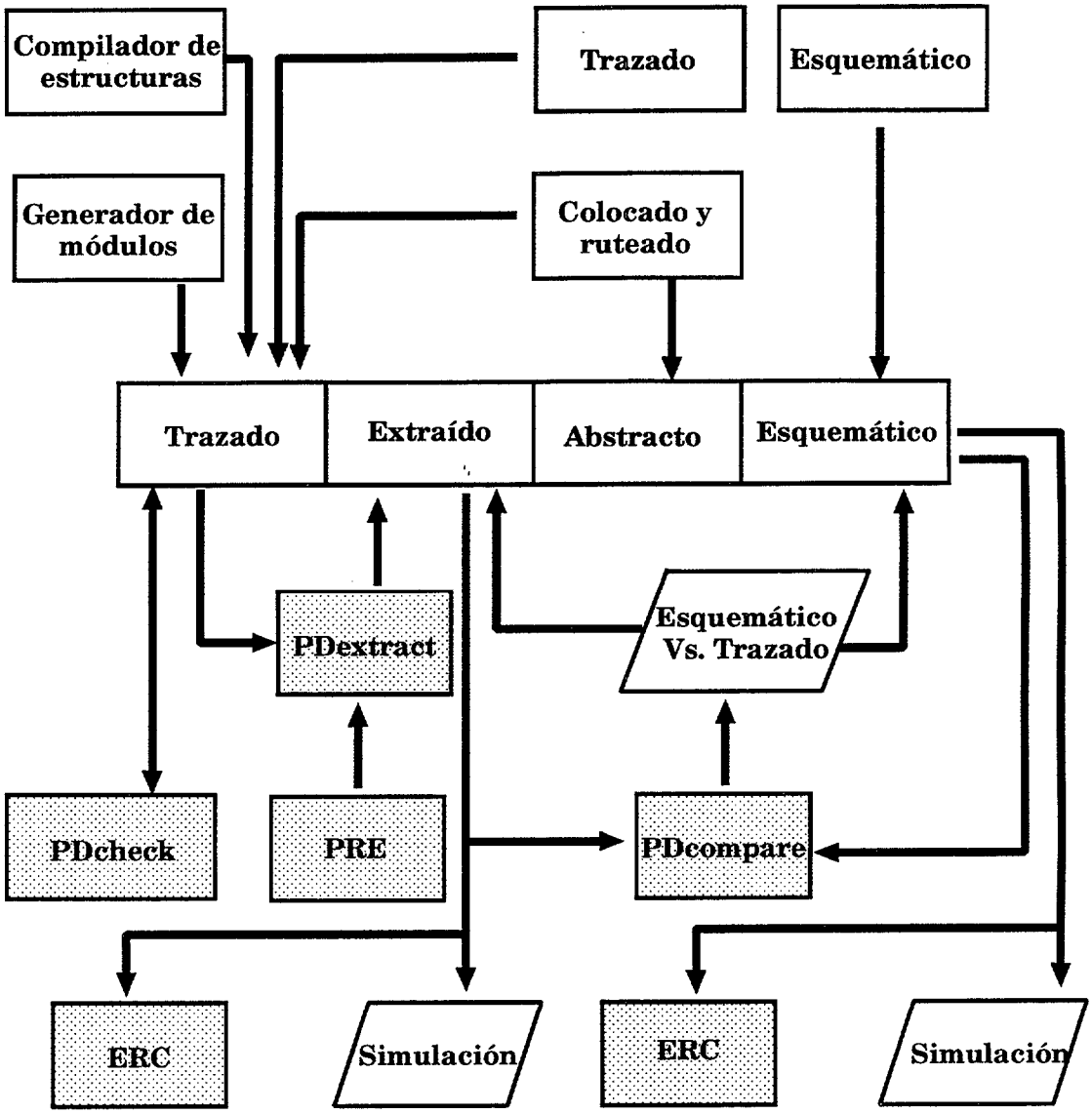


FIGURA 4.3: Herramientas en el entorno Cadence/Edge para el diseño físico.

- reconocimiento y extracción de dispositivos parásitos,
- modelado RC de interconexiones y vías,
- almacenamiento y transformación de todos los datos o de un subconjunto de ellos.

En la extracción de parámetros físicos de dispositivos activos medimos la longitud y ancho de la puerta del transistor. De estas medidas se obtiene la capacidad de la puerta y cuantos parámetros dependientes se deduzcan. Las capacidades parásitas resultantes de la interconexión de dispositivos dependen de tres magnitudes: el área, el perímetro y la longitud equivalente de la interconexión. El método para determinar la longitud equivalente de interconexión, consiste en encontrar la longitud de un rectángulo equivalente, cuya área y perímetro coincidan con el área y perímetro de la interconexión (véase figura 4.4). En las siguientes expresiones 4.1, se deducen la longitud y el ancho equivalente.

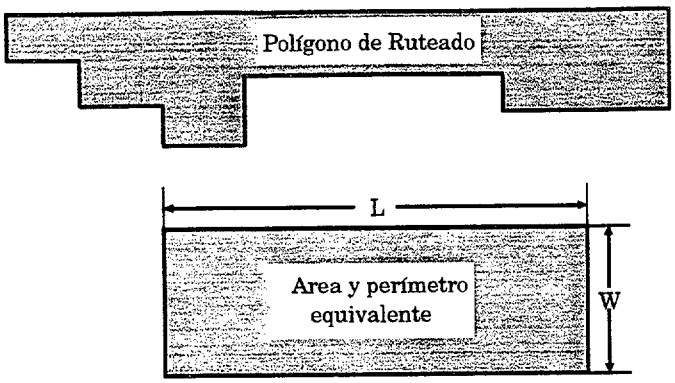


FIGURA 4.4: Rectángulo equivalente para la estimación de parásitos.

$$\begin{aligned} L_{\text{equivalente}} &= \frac{\frac{P_{\text{ruteado}}}{2} + \sqrt{(\frac{P_{\text{ruteado}}}{2})^2 - 4 \times A_{\text{ruteado}}}}{2} \\ W_{\text{equivalente}} &= \frac{A_{\text{ruteado}}}{L_{\text{equivalente}}} \end{aligned} \tag{4.1}$$

La capacidad entre cualesquieras dos nodos se calcula mediante la siguiente expresión 4.2:

$$\begin{aligned} C &= L_{\text{equivalente}} \times W_{\text{equivalente}} \times C_{\text{Area}} \\ &+ 2 \times (L_{\text{equivalente}} + W_{\text{equivalente}}) \times C_{\text{Perímetro}} \\ &+ \text{Máximo}(W_{\text{equivalente}}, L_{\text{equivalente}}) \times C_{\text{Longitud}} \end{aligned} \tag{4.2}$$

En la figura 4.5 se muestra el resultado de aplicar uno de nuestros extractores, a una porción de circuito. En esta versión sólo se han extraído capacidades parásitas y dispositivos activos.

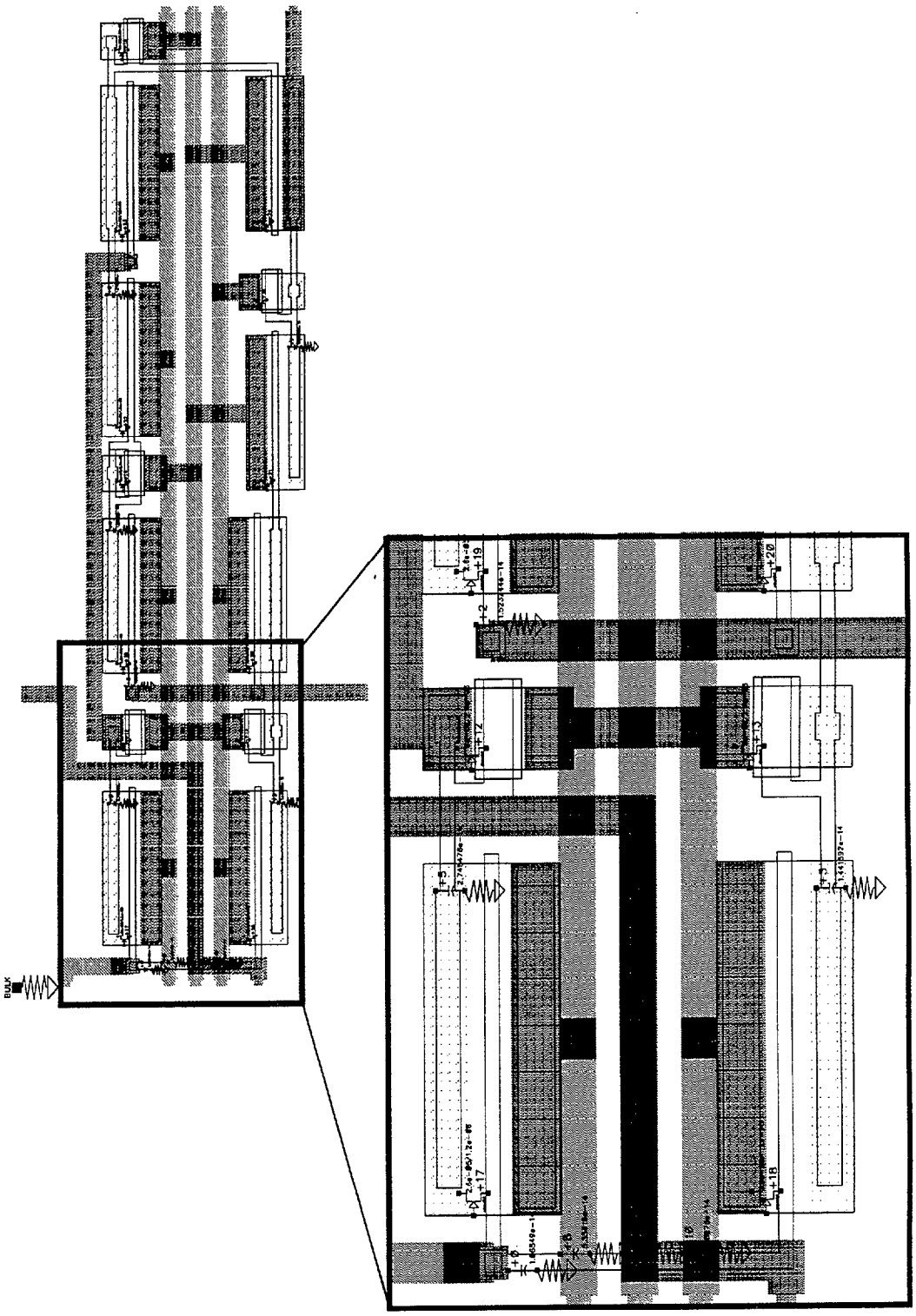


FIGURA 4.5: Extracción de dispositivos activos y parásitos.

En este entorno específico para tecnología GaAs, y como herencia de la estructura de Cadence/Edge, existen diferentes ayudas para la captura del trazado físico. Todas ellas tienen en común que soportan jerarquía, multiventana, edición simultánea de varios circuitos con diferentes representaciones y acceso a herramientas de verificación. Se dispone de editor de polígonos, editor de células parametrizables y de herramientas para diseño simbólico y compactación y/o espaciamiento. Debido a la rapidez con que la tecnología maduró, pronto se iba a desarrollar una segunda adaptación del entorno orientado a una utilización intensiva del diseño simbólico y paramétrico, disponiendo de transistores, vías, contactos, puertas básicas etc ..., parametrizados según dimensiones especificadas por el usuario.

Las principales ventajas a destacar de esta adaptación son las derivadas del uso del diseño simbólico, la independencia de las reglas de diseño tecnológicas y la facilidad a la hora de realizar un trazado físico. En Cadence/Edge la independencia tecnológica no está bien conseguida, puesto que no se mantiene la conectividad en la base de datos cuando estamos realizando el trazado simbólico. Es a posteriori y precisamente cuando se va a proceder a la compactación y/o espaciamiento cuando se extrae la conectividad del circuito. Durante la fase de trazado y de extracción no hay información acerca de la conectividad y la base de datos no entiende más que de objetos geométricos tales como, líneas, polígono etc .... Lo anterior supone un grave perjuicio en contra de la reutilización de los diseños, incluso cuando varían las reglas tecnológicas en un mismo proceso, lo cual hace inviable la posibilidad de que un diseño siga fielmente las variaciones dentro de una misma tecnología. En la figura 4.6(a) se muestra el trazado simbólico del esquema de puertas presentado en la figura 4.2.

Las ventajas alcanzadas con la creación parametrizada de dispositivos simbólicos, se pierden en el momento de la compactación o espaciamiento. Si se permite que la compactación o espaciamiento de reglas de diseño alcance niveles jerárquicos que incluyan reglas de formación de los dispositivos básicos – transistores – el tiempo consumido en compactar o espaciar una célula de hasta 5000 transistores puede eternizarse incluso sobre las estaciones de trabajo con más capacidad de computación. Por otra parte, el entorno de creación de dispositivos simbólicos parametrizables está muy orientado hacia diseño CMOS, lo cual se deja notar en la descripción y formación de otros tipos de transistores, como es el caso de los MESFETS. Para poder adaptar de forma adecuada el diseño simbólico a la tecnología GaAs, hemos tenido que realizar especificaciones tecnológicas de formación de dispositivos simbólicos de forma muy restrictiva y poco ortodoxa.

Un claro ejemplo de lo anterior, es la propia especificación del transistor MESFET. Mientras la sintaxis para la especificación de capas, sólo admite los tipos propios del diseño CMOS: conducción, vía, pozo, difusión e implantación; para especificar las reglas de formación de un MESFET se necesita distinguir entre dos tipos de difusiones – n^+ y n^- –. Con el fin de obviar esta carencia, se han de crear dos máscaras distintas de tipo difusión, y crear un contacto entre ellas para mantener la conectividad. Otro ejemplo, es la propia especificación de la puerta de un transistor MESFET. Debido a que entre el metal de puerta y el canal no existe

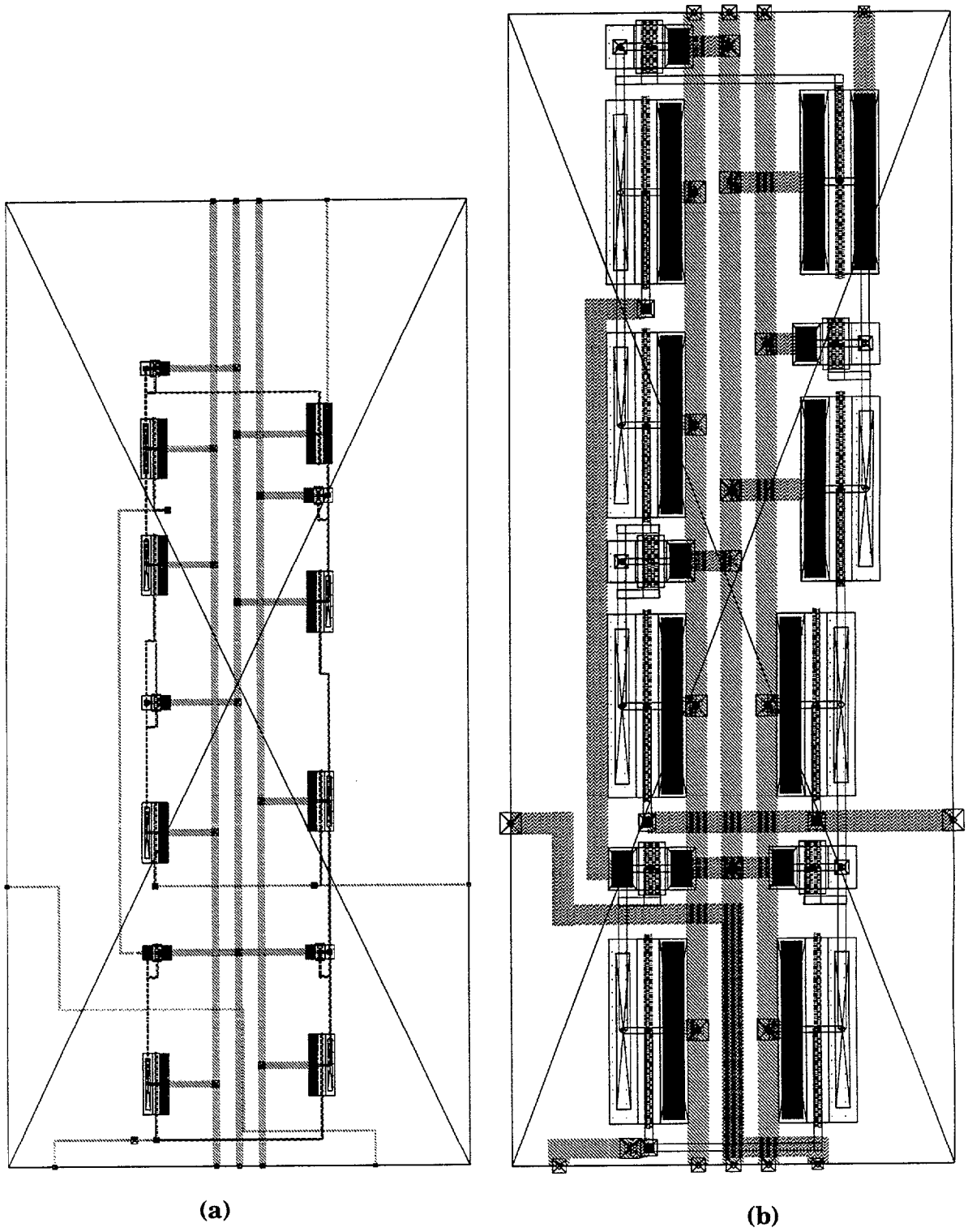


FIGURA 4.6: Trazado simbólico del esquema de puertas presentado en la figura 4.2: (a) Composición y (b) trazado compactado y espaciado.

ningún elemento de contacto – en Si existe el contacto entre polisilicio y difusión –, hemos tenido que crear una nueva máscara, que permita definir la zona de contacto. Ello hace inviable la extracción de dispositivos, si no se tiene en cuenta lo anterior.

4.1.2 Menthor Graphics.

GDT Designer es un conjunto de herramientas de **Mentor Graphics** para diseño VLSI y desarrollo de librerías de células estándares. Dispone de interfaces programables por el usuario escritas bien en lenguaje C (**Ldbi**), o en una extensión del lenguaje **L** al estilo de programación de **GENIE** denominado **Lx**. El editor gráfico **Led** permite la realización del trazado físico al mismo estilo que el de otras interfaces como Cadence/Edge. El lenguaje básico para la generación de circuitos es L, siendo procedural y estando orientado principalmente a la generación de módulos, mientras que el lenguaje de generación de células Lx sólo abarca la metodología de células estándares. A diferencia de Cadence/Edge, en el entorno GDT el diseño simbólico se basa en la descripción del trazado sobre una rejilla virtual, con posterior compactación y/o espaciamiento, y manteniendo siempre la conectividad. Al igual que Cadence/Edge dispone de conversores de datos, de ficheros tecnológicos configurados por el usuario, utilidades de compactación, extracción y simulación. Todas las herramientas están orientadas básicamente a la metodología de células estándares, disponiendo de herramientas de colocado y conexionado muy elementales. Las principales desventajas son:

- las derivadas de disponer de lenguajes no estándares y distintos para la descripción del trazado,
- la orientación al estilo de células estándares de determinadas funciones de colocado y conexionado y la baja programabilidad de las mismas,
- la fuerte dependencia tecnológica del entorno hacia CMOS, y
- la no definición del sistema para el desarrollo de herramientas de compilación de circuitos con una metodología distinta a la de células estándares.

4.1.3 OCTTOOLS.

OCTTOOLS es una colección de programas y librerías que forman un sistema integrado para diseño VLSI, desarrollado en la Universidad de Berkeley. Está escrito enteramente en lenguaje C, y es de dominio público. El sistema incluye herramientas para síntesis lógica, asignación de estados, células estándares, diseño *full–custom*, circuitos, simulación a nivel lógico y de conmutación, y una gran variedad de programas de utilidad para manipulación de esquemáticos, diseño simbólico y geométrico. Las herramientas están integradas dentro del administrador de datos de OCTTOOLS y la interfaz de usuario **VEM**.

Como sistema de diseño tiene todas las ventajas de herramientas comerciales tales como Cadence y Mentor, además de contar con una gran tradición en investigación sobre diseño VLSI. En su definición se recogen trabajos desde principios de los ochenta hasta nuestros días de autores que han contribuido de forma importante en el desarrollo de herramientas para el diseño VLSI. Pero la principal ventaja, es que dispone una gran librería de programas orientados al diseño *full-custom* fácilmente adaptables a distintas metodologías de trazado, incluyendo programas para colocado y conexionado modificables por el usuario, en función de la metodología a implementar. Uno de los aspecto decisivos, lo conforma la política de trazado simbólico que implementa. Los dispositivos – transistores, vías, contactos – más comunes son compartidos por todos los usuarios, accediendo a distintas vistas función de la aplicación, mientras que cada aplicación puede generar sus dispositivos específicos. La conectividad forma parte integrada de la base de datos, y no se obtiene como un resultado de un proceso de análisis del trazado. Lo anterior permite seguir fielmente los cambios dentro de una misma tecnología, y a la vez simplificar el proceso de compactación y espaciamiento.

La tradición con la que se ha desarrollado el sistema OCTTOOLS, el lenguaje de programación único y estándar sobre el que se fundamenta, y el éxito conseguido en el desarrollo de compiladores de Si – **MOSAICO** –, son las principales razones que han influido en la elección de este entorno, para la consecución del presente trabajo. Además, la distribución OCTTOOLS–5.1 se ha construido y comprobado en la siguiente combinación de máquinas y sistemas operativos: DECstation 3100, 5000 bajo Ultrix 4.1 y 4.2; DEC VAX bajo Ultrix 4.1 y 4.2; Sun 3 y 4 bajo OS 4.0; Sun SparcStation bajo OS 4.0; Sequent Symmetry, IBM RS/6000 bajo AIX 3.1.

4.2 Problemática de la síntesis de alto nivel en VHSIC.

En la síntesis de circuitos, desde una descripción de alto nivel hasta un nivel de transferencia de registros, o de puertas lógicas, se han desarrollado numerosos trabajos que han definido claramente el proceso de traducción. El trazado basado en células estándares innegablemente desperdicia área del circuito integrado. Hasta un 80% del área del circuito se utiliza para el conexionado. Por otra parte el sistema de colocado y ruteado no es capaz de generar circuitos integrados con un tamaño y prestaciones deseados, y con unos límites de potencia [101]. Por ello, la compilación basada en células estándares no es más efectiva que la fundamentada en el estilo *full–custom.*

La tecnología GaAs presenta la problemática derivada del diseño de circuitos de alta velocidad, que fundamentalmente se orienta hacia la reducción de la inductancia de las líneas de alimentación y el apantallamiento de las líneas de señal y lógica. Por otra parte, hay una sensibilidad a la carga que limita el *fan–out* y el *fan–in* de cada puerta. Aunque existen familias lógicas que pueden soportar bien altas cargas a la entrada y salida, su consumo de potencia hace que los niveles de integración

bajen considerablemente. Todos estos aspectos corroboran el hecho de que el proceso de síntesis convencional que hace uso extensivo de la aproximación de células estándares y de redes de puertas no es lo adecuado para obtener altas prestaciones de la tecnología GaAs.

Con el fin de acercar ambos niveles de descripción (comportamiento–estructural–físico) en tecnología GaAs, debe resolverse la automatización del diseño bajo una perspectiva metodológica *full–custom.* Un primer avance lo supone la generación de células con especificaciones *top–down* y *bottom–up*, para posteriormente pasar a la generación de módulos. En este trabajo se hace principal hincapié en la generación de células, y resuelto este problema la generación de módulos se simplifica, pudiéndose abordar con los procedimientos desarrollados. De hecho, si los módulos son estructurados, esto es, están particionados regularmente en células, una vez generadas las células lo que procede es replicarlas y ensamblarlas. Para el ensamblado y replicado de un conjunto de células, se necesitan herramientas que manejen estructuras al estilo de matrices y que además permitan realizar el conexionado (si no es posible el simple adosado) de los componentes de la matriz.

En todo este proceso que va desde la generación de células al replicado y ensamblado de las mismas y posterior adosado y conexionado para constituir módulos, no se obtendrían resultados satisfactorios si no se introduce la planificación. El trabajo que nos ocupa, hereda del sistema OCTTOOLS el carácter de aproximación *top–down* con que están pensadas sus herramientas. De hecho, cabe la posibilidad de planificar la relación de aspecto y la posición de los terminales de entrada y salida de cada célula, con el fin de que el módulo resultante satisfaga ciertos condicionantes de estilo. O bien, dadas unas especificaciones de relación de aspecto y posición de los terminales, se podría alcanzar una partición en células que una vez generadas se aproximasen a lo exigido en un nivel de jerarquía superior. El conjunto de herramientas desarrolladas carece de un proceso continuo, de iteraciones de planificación y colocado, controlado por una herramienta de más alto nivel, para implementar las ideas anteriores.

Las iteraciones de planificación y colocado pueden comenzar con una planificación inicial, en la que muchos detalles para cada célula se obtienen a partir de la base de datos de **OCTTOOLS**. Para algunas células el trazado completo puede estar perfectamente definido. Para otras, sólo se define un conjunto de restricciones, tales como posición de los terminales o relación de aspecto. Otras células pueden describirse a nivel lógico. Para tales células se lleva a cabo una estimación del área, basada en la complejidad de la célula y en las características del generador que se utilizará. Los terminales de las células pueden moverse libremente en el borde de la misma. Estos terminales se denominan flotantes.

Después de cada iteración, el planificador generaría como salida la posición y orientación de cada célula y también la forma y posición de los terminales para las células que no fueron especificadas completamente. Esta información se pasa al generador de módulos. La entrada al generador de módulos viene de dos fuentes, una del planificador que especifica las restricciones geométricas como se describió. El otro dato es la descripción lógica que se introduce a través de una lista de no-

dos a nivel de transistores o de puertas. Esta descripción puede proceder de una herramienta de más alto nivel como un compilador de estructuras, de rutas de datos, de memorias, etc ... Si el generador de módulos es incapaz de satisfacer todas las restricciones impuestas sobre una célula particular, la forma actual de la célula generada se realimenta a la herramienta de planificado y colocado y se inicia una nueva iteración. El módulo de planificación y colocado puede interaccionar con un verificador temporal que admita las lista de nodos en los formatos más usuales (HSPICE, SPICE). El propósito del analizador temporal es criticar la planificación. La información temporal se utiliza para evaluar las restricciones impuestas por la herramienta de planificación y colocado en las longitudes máximas y mínimas de ciertos nodos.

Capítulo 5

Desarrollo de un entorno de síntesis de células para circuitos de muy alta velocidad en GaAs: OLYMPO.

OLYMPO es un conjunto completo de herramientas para planificación, colocado y ruteado de células, orientado a tecnologías de altas prestaciones como GaAs, y con especial énfasis en el desarrollo de entornos de compilación. Este conjunto de herramientas están escritas en lenguaje C, desarrolladas en el sistema operativo UNIX, sobre la base de la distribución **OCTTOOLS 5.1** de Berkeley. La elección de **OCTTOOLS** como entorno de diseño para la realización de este trabajo fué estudiada en el capítulo cuarto. Para cada herramienta se ha desarrollado el comando de ejecución a nivel de sistema operativo y el paquete de librería de funciones, compilables en todas las máquinas de la distribución **OCTTOOLS 5.1** [1], siendo totalmente compatibles con esta distribución.

Sobre la misma distribución **OCT**, el Departamento de **Electrical Engineering and Computer Sciences** de la Universidad de Berkeley desarrolló un entorno para compilación en Si [49] – **MOSAICO** –.

[1] Se utilizará el término OCT, como alias.

5.1 Estructura de OLYMPO.

OLYMPO consta de una estructura básica, capaz de generar células utilizando el estilo de la Notación en Anillo, generar módulos resultantes de una partición estructurada o tecnológica – teniendo en cuenta o no la distribución de interconexiones propuesta –, y ofrecer al diseñador de sintetizadores en GaAs todo un entorno para generación o compilación final del circuito.

Para desarrollar **OLYMPO** hemos incorporado librerías de programas en:

el núcleo, desarrollado sobre librerías estándares y de dominio público **OCT,**

un nivel intremedio, donde hemos desarrollado un paquete de rutinas para descripción de células y desarrollo tecnológico en GaAs, y

el nivel más externo, formado por las herramientas de **OLYMPO.**

5.1.1 Núcleo de librerías para desarrollo de OLYMPO.

En este apartado se describe en detalle las aportaciones realizadas desde el núcleo de **OCT** hasta el nivel intermedio de librerías, sobre el cual hemos desarrollado **OLYMPO.** Las regiones sombreadas en la figura 5.1 son los paquetes añadidos a la distribución **OCTTOOLS 5.1.**

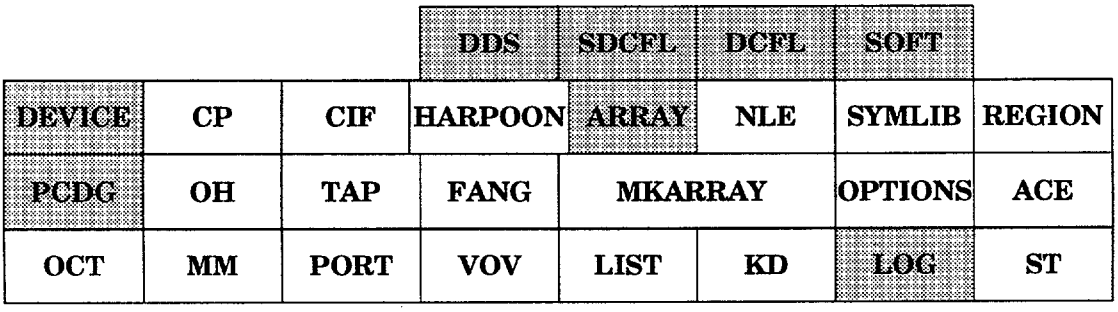


FIGURA 5.1: Estructura jerárquica de la herramienta **OLYMPO.**

En el núcleo de la distribución **OCT** hemos incorporado, en la fase más temprana del desarrollo, **un paquete de librerías – LOG – para gestión del software** que incluye: control del nivel de trazas, tratamiento y recuperación de errores, generación de mensajes y avisos.

Con el fin de disponer de procedimientos para la descripción y generación de transistores, puertas básicas y puertas complejas, **hemos desarrollado en dos capas consecutivas de software, los paquetes de librerías PCDG y DEVICE.** Utilizando paradigmas de la programación orientada a objeto y sobre lenguaje C, hemos desarrollado un conjunto de procedimientos que aportan un nuevo estilo en la descripción del trazado geométrico y simbólico. En la figura 5.2 mostramos como

generar una célula inversora en tecnología H–GasII. Y en la figura 5.3, mostramos el resultado. Este paquete está especialmente indicado para generar librerías parametrizables de dispositivos, que es una carencia que notamos en la distribución **OCT**.

```
main(argc,argv)
int argc;
char **argv;
{
layer gate,met1,met2;
instance inst1,inst2;

TurnOnEnv();

Set(Technology(Name("H-GaAsII")));
Set(Facet(Name("juan:symbolic:contents")));

gate=Layer(Name("GATE"));
met1=Layer(Name("MET1"));
met2=Layer(Name("MET2"));

ints1=Instance(Master(Name("emesfet.8x4:physical:contents")),
                Translation(Point(0,2000)),Name("abc"));

inst2=Instance(Translation(Point(1000,2000)),
               Master(Name("load.4x4:physical:contents")));

Path(Terminal(inst1,Name("DRAIN")),gate,Terminal(inst2,Name("SOURCE")));

Pin(Terminal(inst1,Name("DRAIN")),Name("OUT"));

exit(1);

}
```

FIGURA 5.2: Generación de un inversor en lógica DCFL y tecnología H–GaAsII mediante **PCDG**.

El paquete **DEVICE**, lo hemos construido sobre el paquete **PCDG**, pues necesitamos parametrizar dispositivos básicos, tales como vías, contactos, transistores, diodos, etc ...e independizarlos de las reglas tecnológicas del fabricante. La distribución **OCT** carece de un paquete de librerías para generación automática de dispositivos. **DEVICE** incorpora la generación paramétrica e independiente de la reglas del fabricante de los siguientes dispositivos: EMESFET, DMESFET, contacto entre metal de puerta y metal óhmico, contacto entre metal óhmico y difusión n^-, vía entre metal óhmico y metal de nivel 1, vía entre metal de puerta y metal 1, vía entre metal 1 y metal 2, vía entre metal 2 y metal 3, vía entre metal 3 y metal 4. En la figura 5.4 mostramos algunos de los dispositivos generados.

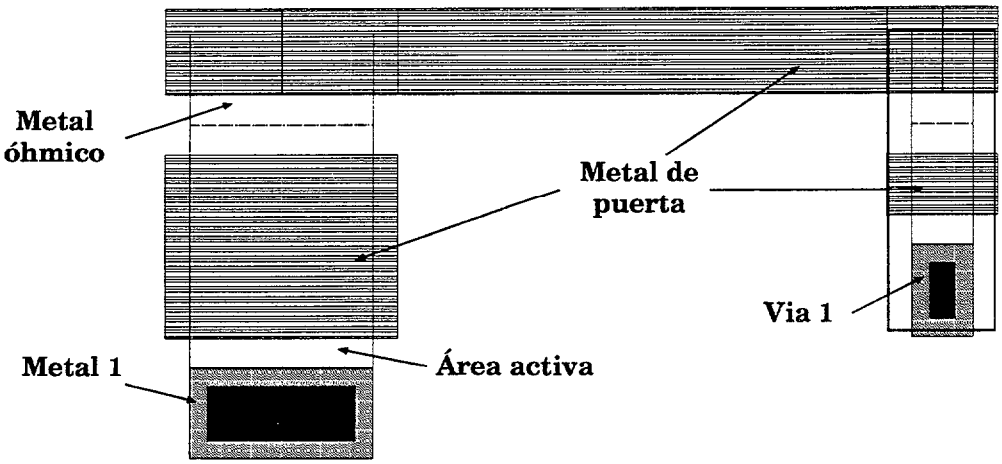


FIGURA 5.3: Inversor en lógica DCFL y tecnología H–GaAsII generado por **PCDG**.

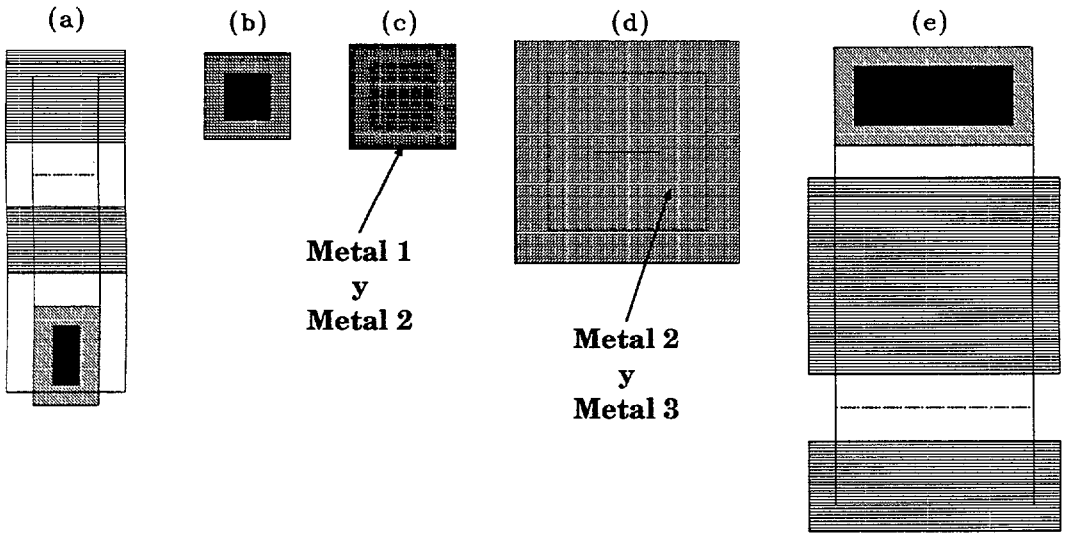


FIGURA 5.4: Algunos dispositivos generados por **DEVICE**: (a) DMESFET, (b) via 1, (c) via 2, (d) via 3 y (e) EMESFET.

A partir del paquete de dispositivos básicos **DEVICE**, hemos desarrollado los paquetes de librerías para generación paramétrica e independiente de las reglas del fabricante, de dispositivos y puertas lógicas. El paquete **DCFL** genera puertas NOR, OR, inversores y *buffers* de acuerdo con la topología de la familia lógica DCFL en cualquier tecnología GaAs. El paquete **SDCFL** genera puertas NOR, OR e inversores según la familia lógica SDCFL. En la figura 5.5 se muestran algunos dispositivos.

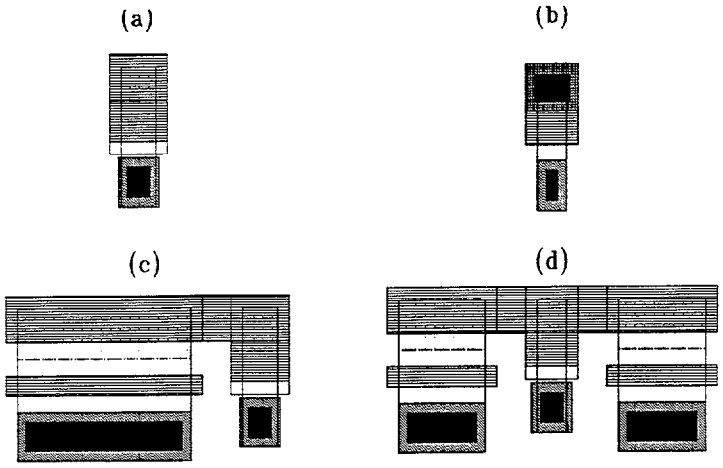


FIGURA 5.5: Algunas primitivas generadas a partir de las librerías **DCFL**, **SDCFL**: (a) *pull-up*, (b) *pull-down*, (c) inversor y (d) NOR.

En otra línea diferente a la de creación de dispositivos parametrizables e independientes de las reglas de diseño del fabricante, hemos desarrollado un paquete de librerías de procedimientos en C para abarcar la compilación de estructuras. **ARRAY** se utiliza para colocar, en una rejilla definida sobre el espacio de diseño, elementos de trazado – transistores, puertas, células, módulos, etc ... – al estilo de una matriz bidimensional. Además, incluye la definición de área de conexionado, permite expandir el área que rodea un elemento, insertando espacio entre elementos, e incluso realizar conexiones ortogonales y sencillas. La herramienta de **OLYMPO** denominada **Baco** la hemos desarrollado utilizando este paquete de librerías.

5.1.2 El nivel más externo en OLYMPO.

La estructura general de **OLYMPO** se representa en la figura 5.6, junto con la secuencia de vistas simbólicas que son almacenadas en cada paso o proceso, por el administrador de datos de **OCT** [43]. Cada paso se describe en detalle en las siguientes secciones.

En la figura 5.6, la columna central representa la secuencia de actividades. Las flechas a la izquierda de esta columna representan los flujos de realimentaciones hacia la tarea inicial, bien por no cumplirse alguna restricción, o bien por no poderse

llevar a cabo la actividad. A la derecha de esta secuencia de actividades aparece el conjunto de vistas generadas, y entre estas vista y la descripción de actividades el flujo entre ambas. Así por ejemplo, la actividad de "planificación y colocado" genera una vista denominada "place", que es la vista de partida para la actividad de "estructuración matricial". Esta última genera una vista denominada "tritón". En la definición de canales y ruteado global de alimentaciones, si el resultado no es satisfactorio se procede de nuevo a la planificación y colocado.

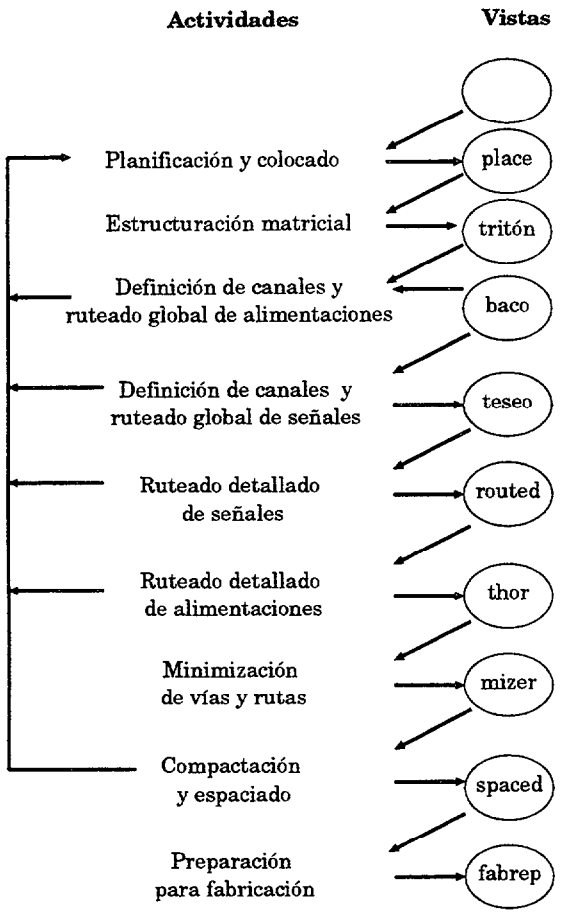


FIGURA 5.6: Estructura de **OLYMPO**.

La figura 5.6 ilustra una de las características más importantes de **OLYMPO**, su modularidad. La modularidad se consigue asegurando que toda la información producida en cada etapa de la estructura, se almacene en la base de datos de **OCT**. Cada una de las herramientas en la estructura obtiene sus datos de entrada a partir de una vista **OCT** y genera como salida otra vista que contiene la información actualizada.

MOSAICO consta de cinco actividades, cuya secuencia se representa en la figura 5.7, junto con el conjunto de vistas simbólicas que genera. La forma de entender la figura 5.7 no difiere de lo explicado para la figura 5.6.

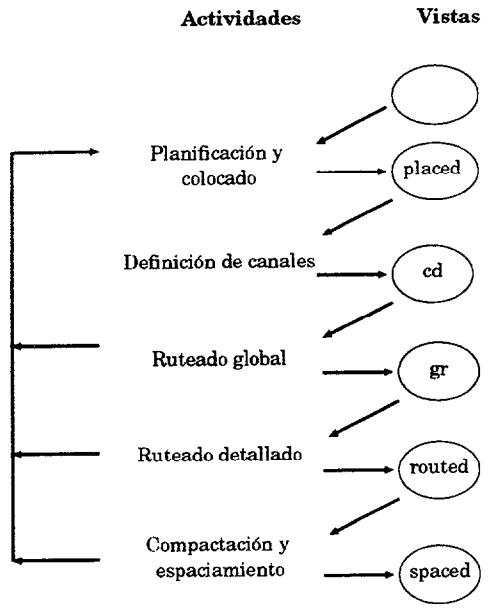


FIGURA 5.7: Estructura de **MOSAICO**.

5.2 Planificación y colocado.

En el entorno **OLYMPO** la actividad de planificación y colocado se realiza unificadamente, de hecho no hay distinción entre planificación y colocado a nivel de implementación. En la etapa de planificación algunos de los parámetros de las células (relación de aspecto, posición de terminales, estimación de área) pueden variarse. Por otra parte, a nivel de colocado todos los parámetros de la célula son fijos. Mientras las tareas de planificación y colocado se vayan realizando y repitiendo en el tiempo, se generan representaciones más refinadas de la célula. Mientras las células ganan en definición física, se incrementa la información disponible para la herramienta, y la planificación se va conviertiendo en una tarea de colocado.

El conjunto de herramientas relacionadas con el proceso de planificación y colocado – Atenea, Rea, Anteo y Tritón – abarcan desde la especificación de las propiedades pre–planificación, hasta la conversión en una estructura más manejable para proseguir el proceso de compilación. En la figura 5.8 se muestra la estructuración de esta actividad. A continuación, se describe brevemente la funcionalidad de cada una de las partes para acometer el proceso de planificación y colocado.

Atenea es una herramienta para asignar propiedades previas a la planificación. Puede evaluar el área de la célula mediante un estimador de lógica cableada. Podemos especificar la relación de aspecto deseada de la célula, o bien especificar su ancho y alto. También podemos definir los lados de la célula sobre los que asignar terminales flotantes o fijos.

El estimador de área de esta herramienta lo hemos elaborado a partir del conocimiento a priori de la metodología de diseño –Notación en Anillo–. En la figura 5.9, se observa que para cada puerta lógica hay una porción de área de

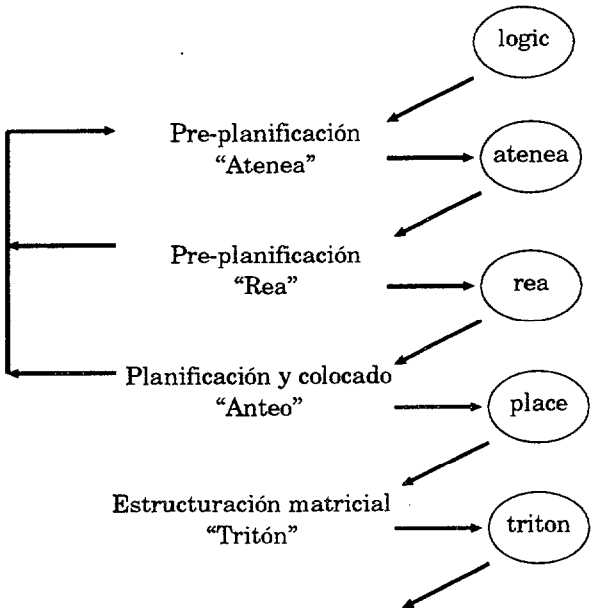


FIGURA 5.8: Estructura de la actividad de planificación y colocado.

interconexión dedicada a alimentación local, y otra porción de área variable de interconexión función del número de terminales de la puerta lógica. Por tanto, el área total estimada, para cada puerta lógica se expresa en la ecuación 5.1:

$$\begin{aligned} A_{\text{total}} &= A_{\text{alimentación}} + A_{\text{señales}} + A_{\text{lógica}} \\ A_{\text{alimentación}} &\propto W_{\text{lógica}} \times H_{\text{alimentación}} \\ A_{\text{señales}} &\propto \log(\text{número de terminales} + 1) \\ A_{\text{lógica}} &\propto W_{\text{lógica}} \times H_{\text{lógica}} \end{aligned} \tag{5.1}$$

siendo:

- A_{total}, es el área total estimada de una puerta lógica.
- $A_{\text{alimentación}}$, es el área que ocupan los buses de alimentación de la puerta lógica. Este área depende del ancho de la puerta lógica y del alto de la alimentación.
- $A_{\text{señales}}$, es la porción de área ocupada por las señales con la que contribuye la puerta lógica.
- $A_{\text{lógica}}$, es el área que encierra a la puerta lógica.
- $W_{\text{lógica}}$, es el ancho del área que encierra a la puerta lógica.
- $H_{\text{alimentación}}$, es el ancho del área que ocupan los buses de alimentación local. La contribución al área de alimentación ($A_{\text{alimentación}}$) de cada puerta lógica es: $\frac{H_{\text{alimentación}}}{2}$.
- $H_{\text{lógica}}$, es el alto del área que encierra a la puerta lógica.

El área total estimada para una célula resulta de la suma de cada una de las contribuciones de las áreas totales estimadas para cada puerta lógica.

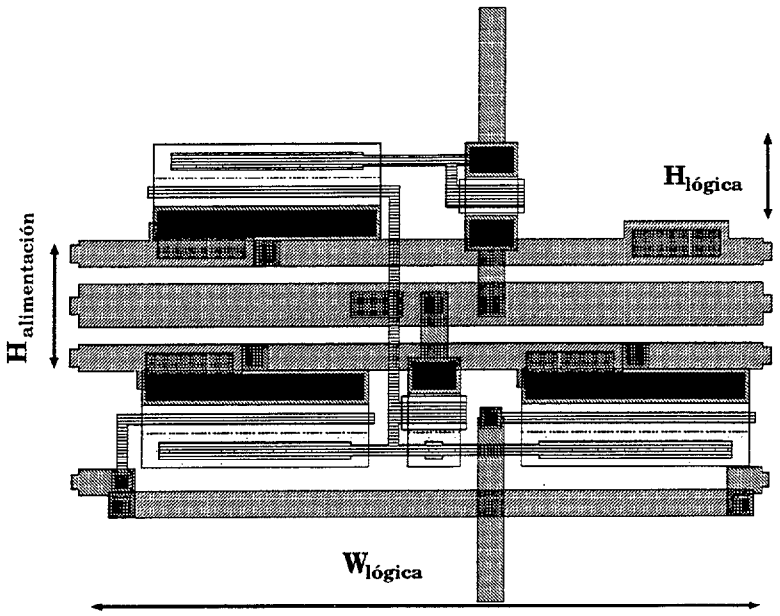


FIGURA 5.9: Dimensiones geométricas para la estimación del área.

Rea es una herramienta que hemos desarrollado para llevar a cabo todo un conjunto de comprobaciones antes y después de la planificación, así como el cambio de alguna información orientada al colocado. También permite especificar y ejecutar particiones tanto estructurales como tecnológicas.

Dado que las propiedades pre–planificación no tienen representación gráfica, la forma de conocer los valores asociados a las misma se realiza mediante **Rea**. Así podemos especificar como propiedad asociada a una célula el generador de módulos para crearla, o bien ejecutar la partición estructural. Esto facilita el mantenimiento en la fase de síntesis y compilación de módulos, pues permite integrar en la base de datos del diseño los procedimientos para su generación.

Una vez asignadas las propiedades pre–planificación, se procede a ejecutar **Anteo**. Dado un área y una relación de aspecto de la célula, el resultado de la planificación de la célula es la posición de los terminales.

Anteo es la herramienta que engloba la aplicación del algoritmo de alineamiento por enfriamiento simulado tanto en la planificación como en la ejecución del estilo de trazado de Notación en Anillo. Puede utilizarse en niveles de descripción jerárquicos superiores, y es compatible con la herramienta **puppy** [18] del entorno **MOSAICO**. La interfaz de usuario es la misma que la utilizada en **puppy** con ligeras modificaciones, y las diferencias estriban en que:

- Hemos cambiado la estimación de las longitudes de nodos por el método del árbol expandido (véase figura 5.10). En ambos programas **puppy** y **Anteo**, se estima la longitud de cada subnodo como la distancia rectilínea entre las coordenadas de los centros de los terminales. La longitud del nodo es la suma de las longitudes de los subnodos. Dado que el árbol expandido de un nodo no es necesariamente el árbol de expansión mínimo,

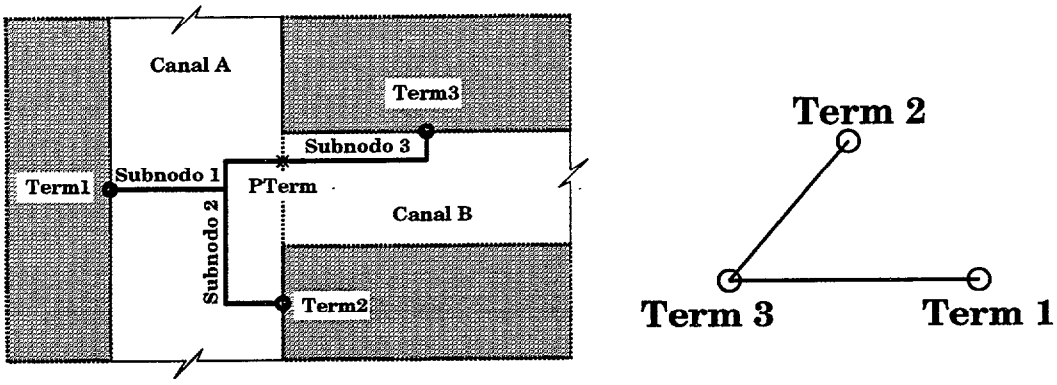


FIGURA 5.10: Árbol expandido de un nodo de conexión.

en **puppy** se realiza un alineamiento por enfriamiento simulado en la forma del árbol. En **Anteo** hemos implementado la solución al problema de **Steiner** para árboles expandidos de tres y cuatro subnodos. Dado que en tecnología GaAs, para las familias lógicas DCFL y SDCFL, el *fan-out* se limita entre 2 y 3, la mayor parte de los nodos tienen tres o cuatro subnodos y la solución al problema de **Steiner** es directa. Para aquellos nodos que no podamos resolver de la forma anterior, se aplica un alineamiento por enfriamiento simulado en la forma del árbol de expansión. En la figura 5.11, se muestra la solución del problema de **Steiner** para un nodo constituido por tres subnodos, y en la figura 5.12 la solución para un nodo formado por cuatro subnodos.

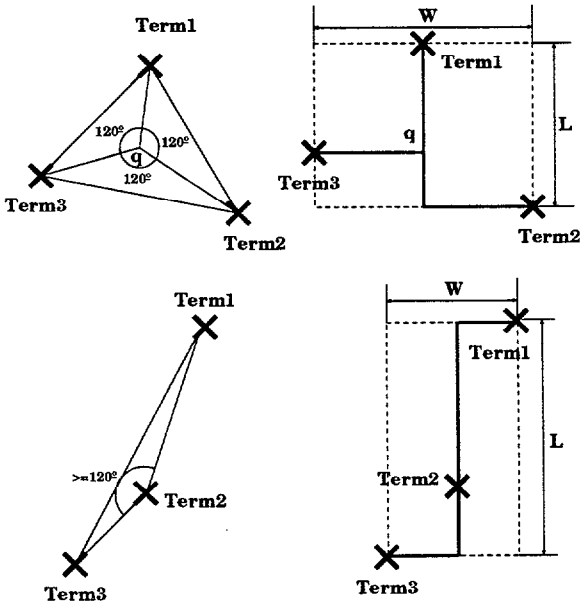


FIGURA 5.11: Solución del problema de **Steiner** con tres subnodos.

- Hemos incorporado en la estimación del área de ruteado, el área dedicada a alimentación local, de acuerdo con la metodología de Notación en Anillo.

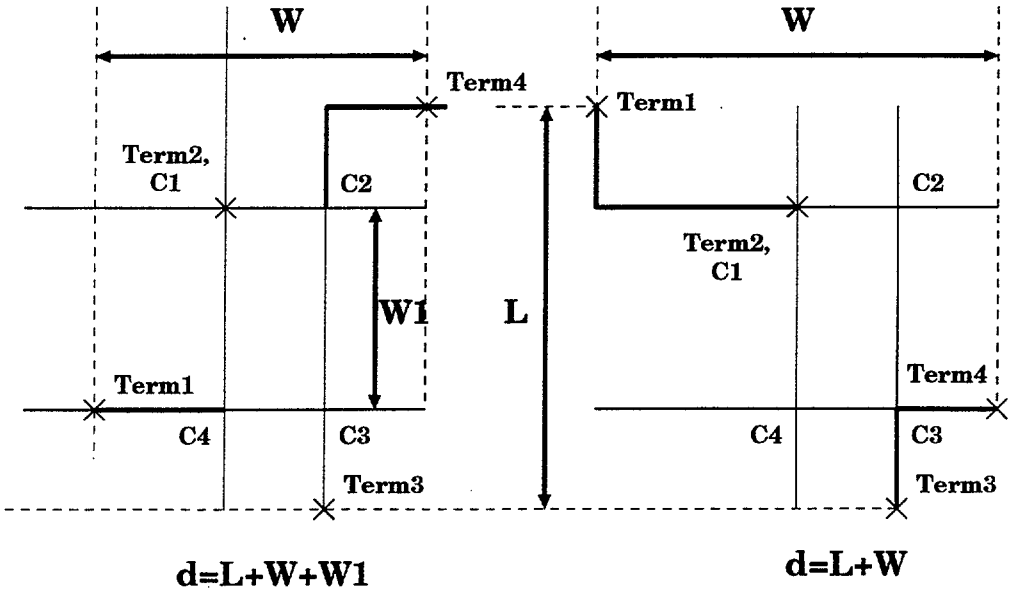


FIGURA 5.12: Solución del problema de **Steiner** con cuatro subnodos.

Las expresiones para el cálculo ya han sido mostradas en la presentación de **Atenea.**

- Hemos incorporado un estimador de capacidades al estilo del método del rectángulo equivalente. En el capítulo cuarto, sección 4.1.1.1, se describió detalladamente este método para estimar las capacidades del cableado. El hecho de incorporar el método del rectángulo equivalente responde a que, si se ha de realizar un análisis de prestaciones con la finalidad de realizar optimizaciones locales, extraeremos con ese método. Si cambiamos el estimador de capacidades, tendremos mayor variabilidad en la estimación y el proceso de optimización puede ser erróneo.

Tritón lo hemos desarrollado con el fin de transformar un colocado de células a cualquier nivel – transistores y puertas lógicas – en una disposición matricial. Esta organización es típica de los generadores de estructuras.

En el espacio de diseño utilizado en el colocado de transistores y puertas, hemos considerado el área requerida para el trazado de la alimentación a nivel local de cada uno de ellos. El área total de ruteado, para cada transitor y puerta, la hemos repartido equitativamente, como si se tratara de una expansión de los bordes de las entidades. Con esta disposición resultante, vamos a realizar la partición en forma de tiras de puertas lógicas con el fin de facilitar los canales para alimentación.

Mediante **Tritón**, podemos crear una rejilla sobre el área resultante del colocado, cuya resolución horizontal y vertical sea igual a la dimensión mínima de ancho y alto, respectivamente, del conjunto de puertas que forman parte del colocado. **Tritón** asigna fila y columna a cada puerta, de acuerdo con la posición de su centro geométrico sobre la rejilla. En el cálculo de las dimensiones mínimas, se tiene en cuenta la expansión de la célula. En la figura 5.13,

observamos la rejilla sobre una matriz de transistores o puertas previamente colocadas.

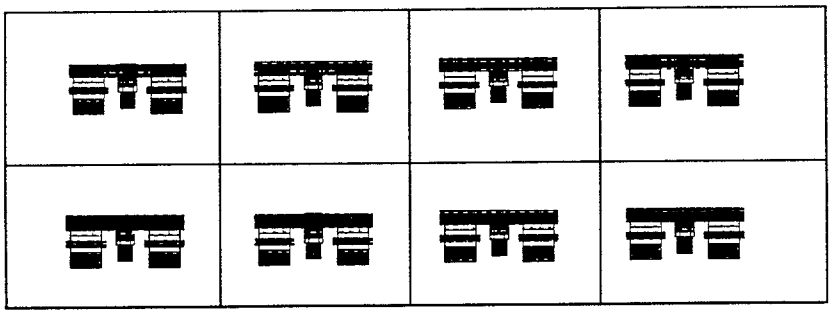


FIGURA 5.13: Rejilla para transformación de cordenadas ortogonales a filas y columnas.

5.2.1 Transformación del esquema lógico.

El procedimiento genérico para transformar un esquema a nivel de puertas, en una representación más susceptible de planificarse bajo la metodología de Notación en Anillo, se expone a continuación:

- Como primer paso, la lógica se transforma en su diagrama de flujo de datos equivalente utilizando las transformaciones básicas que se muestran en la figura 5.14. De esta forma, cada una de las puertas se representan mediante nodos equivalentes, con tantos arcos de llegada como entradas y un arco de salida.

  En la figura 5.14 el símbolo "<", a la salida de la figura de las puertas, representa la existencia de un *buffer* de salida (lógica SDCFL). El símbolo equivalente en el operador es un círculo a la salida.

- A partir de este diagrama se asignan pesos a cada arco, de acuerdo con los requerimientos de *fan–out*, *fan–in*, capacidad, longitud, etc ..., obteniéndose un grafo orientado y con pesos positivos en cada arco.

- Las rutas lógicamente equivalentes pueden optimizarse sobre el grafo con el fin de convertirlas en rutas eléctricamente equivalentes.

- La simetría en el diagrama o en el grafo equivalente, puede ayudar para generar una representación simbólica simétrica. Sin embargo es difícil de encontrar en diagramas con cierta complejidad.

Como ejemplo vamos a aplicar este procedimiento al diseño de un divisor por cuatro complementario, que se presenta en la figura 5.15(a):

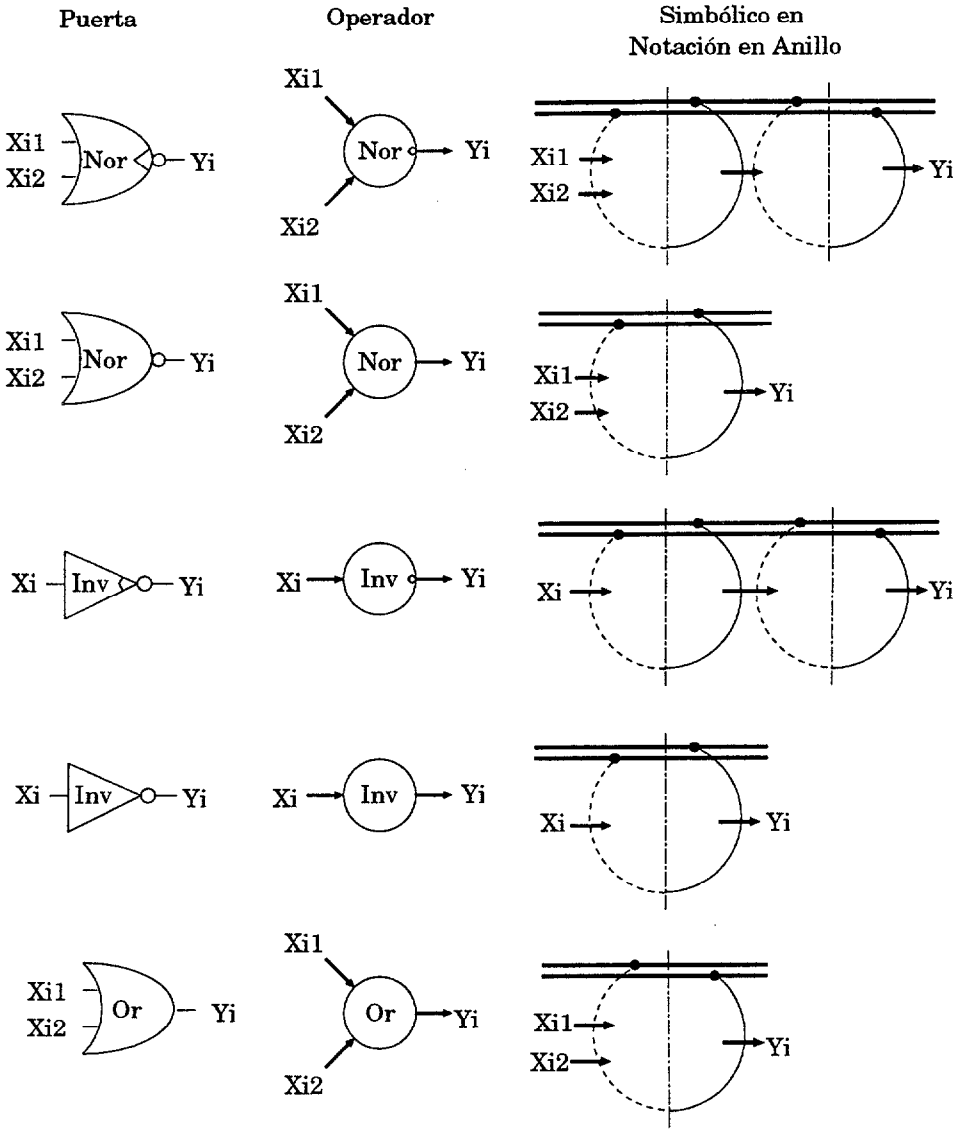


FIGURA 5.14: Representación de puertas.

Primer paso: Obtención del diagrama lógico, figura 5.15(a).

Segundo paso: Transformación en su diagrama de flujo de datos equivalente, figura 5.15(b).

Tercer paso: Replanteo del diagrama observando dónde se pueden optimizar las interconexiones, figura 5.15(c).

Cuarto paso: Creación de la representación en notación en anillo, estableciendo los ejes de simetría, si estos existen.

Quinto paso: Obtención de la representación simbólica.

Entre todo este conjunto de pasos del procedimiento propuesto, el tercero es dependiente de la experiencia del diseñador. Cada diseñador puede encontrar una solución distinta y aproximada, tanto más diferenciada cuanto más compleja sea la estructura. Situaciones parecidas se han superado por la implementación de procedimientos heurísticos fuertemente dependientes de la experiencia. El primer paso no es más que una transformación en forma de grafo, donde la puerta se transforma en un nodo con más de una entrada y una sola salida. La asignación de pesos a los arcos de conexión entre nodos, es lo siguiente. En esta asignación se refleja el énfasis de posteriores procesamientos.

5.2.2 Colocado de transistores y puertas bajo la metodología propuesta.

El algoritmo de alineamiento por enfriamiento simulado es un procedimiento ampliamente utilizado en la planificación, colocado e incluso en el trazado de conexiones de circuitos al estilo de diseño basado en módulos [55, 40, 19]. El ámbito de aplicación anteriormente relacionado, se extiende en el presente trabajo, a la planificación y colocado de transistores y puertas lógicas, para la generación de células, bajo restricciones formalizadas en una función de costo.

La información inicial en un problema de colocado consiste en una lista de nodos, que describe el conjunto de elementos a conectar y sus conexiones. Estos elementos a conectar – transistores, puertas, células – pueden ser de cualquier forma, aunque son más comunes los de forma rectangular. Cada célula tiene un número de terminales, que representan los lugares donde los nodos implementan las interconexiones. Los terminales pueden tener una posición fija o pueden estar situados a lo largo de algunos de los lados de la célula. En este segundo caso se dice que los terminales son flotantes. Para describir un colocado se debe especificar la posición de cada elemento. En general, un colocado se puede considerar como un punto **s** en un espacio **S** de todos los posibles colocados. El punto $s \in S$ se especifica por los valores asignados a algunas variables, las cuales representan la posición de una célula o de un terminal. Por el hecho de que se está utilizando el algoritmo de alineamiento por enfriamiento simulado para calcular el valor óptimo de estas variables, se denominan **variables**

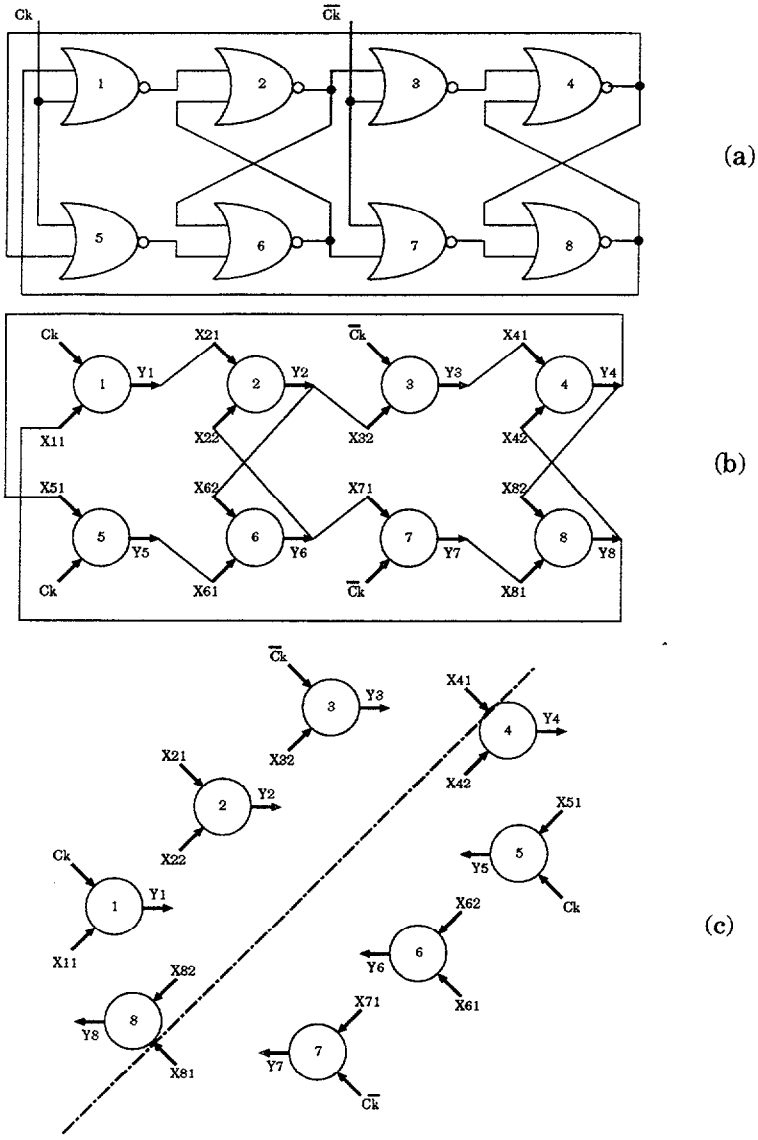


FIGURA 5.15: Divisor de reloj complementario. (a) Diagrama lógico. (b) Diagrama de flujo de datos equivalente. (c) Optimización de interconexiones.

```
mientras no condición final hacer empezar
        desde i := 1 hasta n movimientos hacer bucle interior
            para cadavariable de la lista de variables de alineamiento hacer
                mover variable (figura 5.25);
            fin para
        fin desde
        costo anterior := costo actual;
        costo actual := costo de la actual configuración;
        actualizar la temperatura;
        actualizar el limitador de rango;
        generar perturbación;
        evaluar condición final;
fin mientras
```

FIGURA 5.16: Fundamentos algorítmicos del alineamiento por enfriamiento simulado.

de alineamiento. Las variables de alineamiento a las que se hace referencia en este tipo de implementación son:

Transistores y puertas, para los cuales se selecciona una posición, representada por una tripleta formada por las coordenadas X e Y del centro del rectángulo que los limita y la orientación del mismo;

terminales flotantes, que deben situarse a lo largo del perímetro de la célula a la que pertenecen, sujetos a algunas restricciones durante la planificación;

nodos, para los que debe determinarse la forma del árbol de expansión mínimo, o la mínima longitud.

El algoritmo de alineamiento por enfriamiento simulado intenta llevar a cabo una minimización de una función definida sobre un espacio finito. En este caso el espacio finito es el conjunto **S** de todos los colocados y la función es una función de costo definida por el usuario, tal que el resultado final tiene alguna propiedad determinada y deseada, como minimizar la longitud total de los nodos, o minimizar el área, o una combinación de ambas, entre otras. Esta caracterización de la función de costo es obviamente experimental, pues se trata de caracterizar la función por medio del resultado final, el cual está influenciado a su vez por la elección de la función de costo. Sin embargo, esta caracterización encaja con los aspectos empíricos que existen en el estado del arte del uso del alineamiento por enfriamiento simulado para la minimización de una función de costo compleja. En la figura 5.16, se presenta un esbozo del algoritmo.

5.2.2.1 La función de costo.

La función de costo que el algoritmo de alineamiento por enfriamiento simulado trata de minimizar es la suma ponderada de cuatro componentes genéricos:

- La longitud, bien resultado de una estimación de las interconexiones, o de las violaciones a las restricciones impuestas en la longitud de algunos nodos;
- el área, de la célula y/o de solapamiento entre células;
- medidas de la asimetría y/o dislocamiento de posiciones relativas entre células;
- medidas de la degradación de las prestaciones debidas a las capacidades estimadas del trazado de conexiones.

Precisando, sea **S** el conjunto de todos los posibles colocados de las células que pertenecen a $\mathbf{C_T}$, y que define una función de costo $\mathbf{v} : S \rightarrow \Re^+$. La función de costo v es la suma sopesada de los componentes, según se expresa en la ecuación 5.2:

$$\begin{aligned} v(s) &= \alpha_{wl} f_{wl}(s) + \alpha_a f_a(s) + \alpha_{ov} f_{ov}(s) + \\ &\quad + \alpha_{sy} f_{sy}(s) + \alpha_{ma} f_{ma}(s) + \alpha_{cd} f_{cd}(s) \end{aligned} \tag{5.2}$$

donde:

- $s \in S$;
- $f_{\mathrm{w}} l$ es la **longitud total de interconexión estimada**; la longitud de un nodo se estima computando el semi–perímetro del rectángulo que incluye todos los terminales conectados al nodo, o la suma de las longitudes de los subnodos del árbol de expansión de los terminales del nodo;
- f_{a} es el **área total del chip**;
- f_{ov} es el **área total de solapamiento** entre las células; este componente tiene que ser cero a la finalización del algoritmo, para llevar acabo un colocado factible;
- f_{sy} es una medida de la asimetría del circuito.
- f_{ma} es una medida del dislocamiento de los dispositivos del circuito.
- f_{cd} es una medida de la degradación de las prestaciones del circuito.
- $\alpha_{\mathrm{wl}}, \alpha_{\mathrm{a}}, \alpha_{\mathrm{ov}}, \alpha_{\mathrm{sy}}, \alpha_{\mathrm{ma}}, \alpha_{\mathrm{cd}}$ son pesos no negativos, que pueden establecerse experimentalmente; normalmente, realizando varias experiencias se puede determinar un conjunto de pesos adecuados. Los pesos se eligen por el usuario, usando criterios empíricos.

5.2.2.2 Estimación de la longitud de interconexión.

El cálculo estimado de la longitud de los nodos se fundamenta en dos métodos básicos. El primer método, es el clásico procedimento del rectángulo lindero, mediante el cual la longitud de un nodo se estima como el semi–perímetro del mínimo rectángulo que incluye todos los terminales que pertenecen al nodo. En la figura 5.17, la longitud estimada del nodo que conecta a los terminales $Term1$, $Term2$ y $Term3$, es $L_{rect} + W_{rect}$.

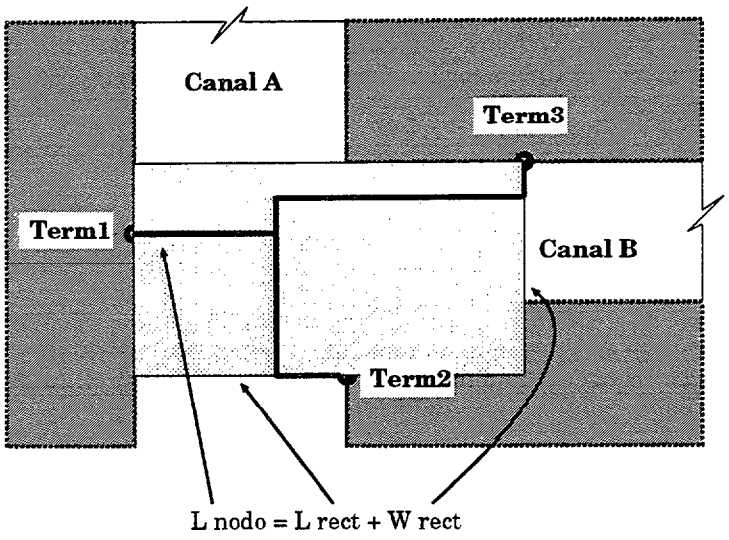


FIGURA 5.17: Método del rectángulo que encierra los terminales del nodo.

En el segundo método, cada nodo se representa utilizando un árbol expandido de sus terminales (véase figura 5.10). Cada flanco en el árbol se denomina subnodo. La longitud de un subnodo es la distancia rectilínea entre los dos terminales del subnodo. La longitud del nodo la estimamos mediante la suma de las longitudes de sus subnodos. El árbol expandido de un nodo no es necesariamente el árbol expandido mínimo. Generalmente, se ha aplicado el algoritmo de alineamiento por enfriamiento simulado a la forma del árbol expandido con el fin de minimizar su longitud. En términos de velocidad de procesamiento, el método segundo es intrínsecamente más rápido que el método del rectángulo lindero. Sin embargo, debido a la necesidad de llevar a cabo movimientos extras para alinear los árboles de expansión, no hay una clara ventaja en velocidad de procesamiento, y por tanto, en el uso de este método. Generalmente, el método del subnodo es más recomendable, puesto que es más preciso. Una alternativa, que hemos desarrollado en la herramienta **Anteo**, es la solución al problema de **Steiner** para árboles expandidos de cuatro subnodos o menos.

5.2.2.3 Estimación del área

En **puppy**, para estimar el área necesaria para el trazado de conexiones, se expande el perímetro de cada célula dependiendo: de la densidad de terminales en cada lado

de la célula, de un parámetro denominado **factor de expansión de la célula**, y del tipo de ruteador que se vaya a utilizar. Lo normal es considerar que se va a utilizar un ruteador de canales. La ley que rige, en esta herramienta, la expansión del área por lado con el número de terminales es de tipo logarítmico (véase ecuación 5.3). La estimación es buena cuando se aplica al colocado de módulos, puesto que el número de terminales suele ser alto y el área ocupada por el módulo es significativa respecto al total.

$$\text{expansión} \propto \text{factor de expansión} \times \log(\text{número de terminales} + 1) \qquad (5.3)$$

En nuestra implementación del cálculo del área de conexionado en **Anteo**, el área ocupada por los buses de alimentación contribuye de manera importante en el área total. Esto atenúa las variaciones introducidas por el factor de expansión en el cálculo del área de trazado de señales. Además, dado que el número de terminales de las puertas lógicas es pequeño si no se tiene en cuenta el área para la alimentación local, los mejores resultados se obtienen estableciendo valores del factor de expansión en el rango 2–4. En **puppy** se recomienda no superar el valor de 2. En la figura 5.18 presentamos varios ejemplos, sobre el divisor de reloj complementario presentado bajo la sección 5.2.1 (figura 5.15)

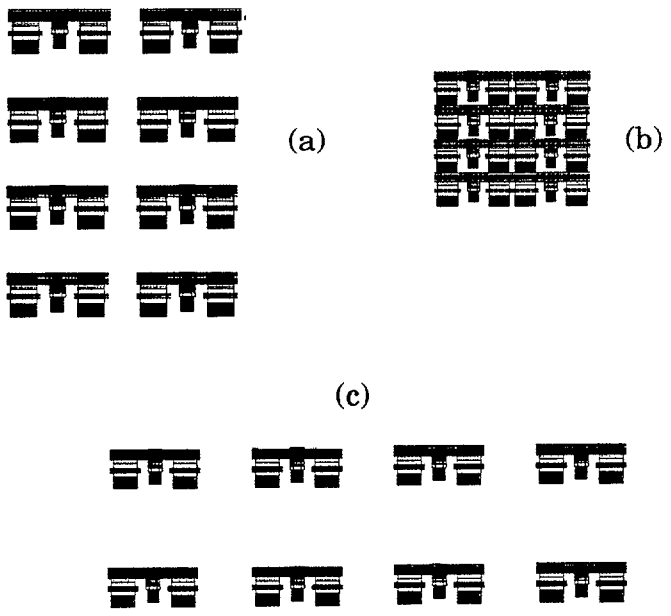


FIGURA 5.18: Evaluación del factor de expansión: (a) Factor de expansión = 2, (b) factor de expansión = 0 y (c) factor de expansión = 2 y teniendo en cuenta la contribución de la alimentación.

Esta estimación del área necesaria a veces no es satisfactoria. En particular, no es válida cuando se quiere adosar células, puesto que el factor de expansión previene del adosado de células. Por otra parte, el hecho de incorporar los principios del trazado simbólico, hace que estas estimaciones no deban tener un carácter marcadamente cuantitativo en nuestra herramienta.

5.2.2.4 Solapamiento.

El solapamiento de áreas tiene una doble naturaleza (véase figura 5.19): por una parte, se refiere a la intersección entre el área externa a un borde o límite preestablecido y la superficie realmente ocupada; y por otra a la intersección entre áreas ocupadas de células – teniendo en cuenta la expansión para conexiones –. El colocado final no debería tener solapamientos, pero la experiencia demuestra que el tiempo de ejecución del algoritmo de alineamiento por enfriamiento simulado, para colocado, mejora si se permite que haya solapamientos en los pasos intermedios de ejecución del algoritmo [20].

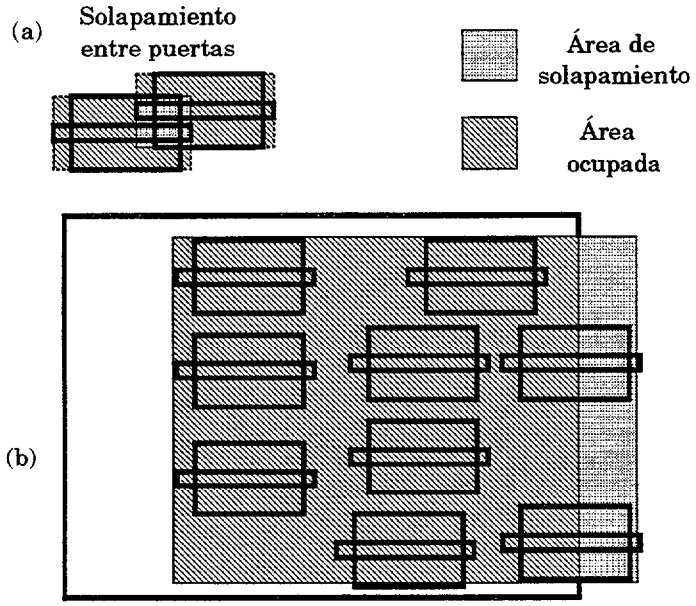


FIGURA 5.19: Solapamiento de áreas: (a) Entre puertas, (b) respecto al límite preestablecido.

El costo por solapamiento es proporcional al área de solapamiento. Para mantener la posición de las células dentro de algún límite preestablecido, se ha de computar una penalización por solapamiento, que es proporcional a la cantidad de área fuera del límite. Este tipo de solapamiento es más costoso que el solapamiento entre células. El que en el colocado final haya una pequeña cantidad de solapamiento, no significa necesariamente que el colocado no es adecuado, sino que dicho solapamiento se debe a una cierta intersección en las regiones en torno a las células. Si no aplicamos ningún peso al costo por solapamiento las células se solapan unas sobre otras.

5.2.2.5 Restricciones aplicadas a longitud de los nodos y al emparejamiento de las células.

En algunas aplicaciones se desea que un nodo crítico no tenga una longitud superior a un cierto límite. Esto se puede especificar en la implementación del algoritmo asignando una propiedad a ese nodo crítico. En la figura 5.20, mostramos el resultado de aplicar un alineamiento por enfriamiento de los nodos ck y $\overline{ck}$ del ejemplo del divisor de reloj complementario presentado en la sección 5.2.1. En la simulación se ha limitado la longitud del nodo ck a $5\mu m$ y la longitud de $\overline{ck}$ a $30\mu m$, y la estimación de la longitud se hace por el método del subnodo. La propiedad asignada al nodo es un valor entero que representa la máxima longitud del nodo en unidades **lambdas**. La interpretación de esta propiedad es diferente dependiendo del método que se utilice para estimar la longitud del nodo. Si se utiliza el método del rectángulo lindero, entonces se entiende que se viola el principio especificado cada vez que el nodo tenga una longitud estimada que exceda la longitud máxima. Si se utiliza el método del subnodo, entonces se viola el principio especificado cada vez que se exceda la longitud especificada. En este último caso podría ocurrir que dado un nodo crítico con muchos subnodos, cada uno de ellos fuera más corto que la longitud máxima de nodo especificada $\hat{L}_{\text{net}}$, pero el total de longitud excediera a $\hat{L}_{\text{net}}$. Tales situaciones no están penalizadas por el método del subnodo en **puppy**, pero en nuestra implementación (**Anteo**) sí se tienen en cuenta.

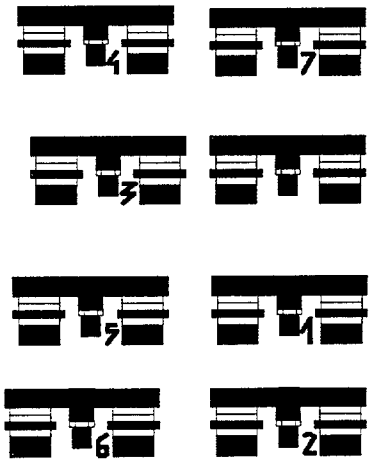


FIGURA 5.20: Alineamiento por enfriamiento penalizando la longitud de un nodo.

5.2.2.6 Restricciones topológicas.

Algunos trazados en GaAs, son total o parcialmente simétricos, mejorando consecuentemente las prestaciones (en términos de dispersión de señal, márgen de ruido, etc ...). La componente f_{sy} en la expresión 5.2 penaliza las puertas y transistores simétricos que no estén equidistantes con respecto a un eje de simetría y las que no

se han alineado respecto al mismo. La orientación del eje y la posición en el espacio son definidas por el usuario. La especificación se puede mantener fija durante la ejecución del algoritmo o bien su posición puede actualizarse periódicamente. En la figura 5.21 se presenta un ejemplo de evaluación de la componente f_{sy} de costo. Para ello, hemos definido un eje de simetría horizontal (véase figura 5.15(c)) sobre la lista de puertas del divisor de reloj complementario definido al principio de la sección 5.2.1. La definición del eje conduce a los emparejamientos de las células $(1,7),(2,6),(3,5)$, quedando libres las células 4 y 8. La definición de un eje de simetría no supone que las primitivas emparejadas deban estar lo más proxima unas de otra, sino que deban situarse a un lado y otro del eje.

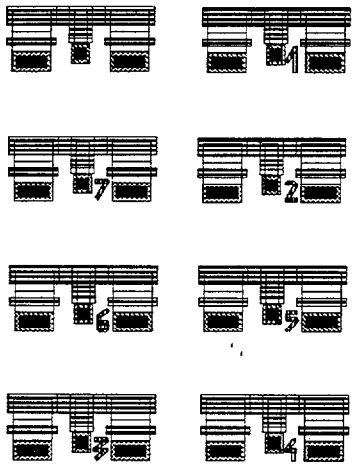


FIGURA 5.21: Alineamiento por enfriamiento evaluando la componente de simetría.

Realizar un colocado a nivel de transistores sin tener en cuenta la reagrupación a nivel de puertas lógicas es un modo de proceder muy tosco. Además, la mayor parte de las funciones de prestaciones de los circuitos en GaAs, dependen de la satisfacción de ciertos comportamientos eléctricos de los dispositivos del circuito. Tales comportamientos eléctricos tienen que ver con la separación física entre dispositivos y sus orientaciones. La componente f_{ma} en la función de costo $v(s)$ penaliza el colocado cuando las células que deben emparejarse están demasiado apartadas o tienen diferentes orientaciones.

5.2.2.7 Restricción a nivel de prestaciones.

El colocado definitivo puede estar dirigido hacia la satisfacción de restricciones de alto nivel sobre las prestaciones, que dependen de la capacidad de los nodos (esta aproximación se puede extender también a considerar la resistencia de los nodos).

Durante la fase de colocado, las capacidades de los nodos no se conocen de forma precisa, puesto que dependen de los detalles del trazado de conexiones. Sin embargo, para un colocado determinado pueden hacerse estimaciones conservativas

de los valores máximo y mínimo de la capacidad de cada nodo, sobre la base de la longitud estimada del nodo y los posibles niveles de metal en las que el nodo se puede trazar. Por lo tanto, los valores máximo y mínimo pueden incidir en la degradación de la función de prestaciones durante cada iteración del algoritmo.

- Si la máxima degradación para una prestación dada está por debajo del límite especificado, entonces no se ha de aplicar ningun costo por penalización. En este caso la restricción siempre se satisface durante la fase de trazado de conexiones si el máximo estimado es conservativo.
- Si el límite establecido está entre las degradaciones máximas y mínimas, se contribuye proporcionalmente a la violación de restricciones, lo cual se añade a la función de costo. En este caso hay posibilidad de que la restricción se satisfaga durante la fase de trazado de conexiones.
- Sin embargo, cuando la degradación mínima posible excede el límite preestablecido, la restricción no se satisface durante el trazado de conexiones.

Para estas tres posibilidades se asigna un valor a la componente $\mathbf{f}_{cd}$ de la función de costo $v(s)$ que dirige el algoritmo de alineamiento por enfriamiento simulado.

5.2.2.8 El conjunto de movimientos.

Para cada tipo de variable de alineamiento hay un conjunto de movimientos posibles. Una "puerta lógica o un transistor" puede trasladarse, rotarse o intercambiarse por otra. Un "terminal flotante" puede moverse de un sitio a otro, un "nodo" con más de dos terminales puede modificar su árbol expandido.

Traslación: El módulo del vector traslación se controla por un mecanismo denominado limitador de rango [88, 89]. Este limitador de rango es un par de números positivos (R_x, R_y), que sirven para calcular la ventana que limita el movimiento. Esta ventana para un elemento de colocado – transistor, puerta lógica u otro elemento de jerarquía superior – es un área rectangular de ancho R_x y alto R_y centrada en el elemento. Sea W y H el alto y ancho del límite de la célula.

En **puppy** el limitador de rango se establece en $(\frac{W}{8}, \frac{H}{8})$. Este rango inicial decrece si la fracción de traslaciones aceptadas sobre intentadas es menor del 0.25%. El limitador de rango no puede ser menor de $(4, 4)$. En nuestra herramienta **Anteo** seguimos la misma política.

En una traslación, el centro de la puerta lógica o transistor A se traslada a una posición aleatoria dentro de un área determinada por la intersección del borde de la célula y la ventana limitadora de rango. Si la intersección es vacía, el centro de la puerta lógica o transistor se sitúa de forma aleatoria dentro de los límites de la célula. La intersección puede ser vacía si las puertas o transistores están inicialmente fuera de los límites de la célula o si el límite de la célula ha cambiado dejando a la puerta o transistor fuera.

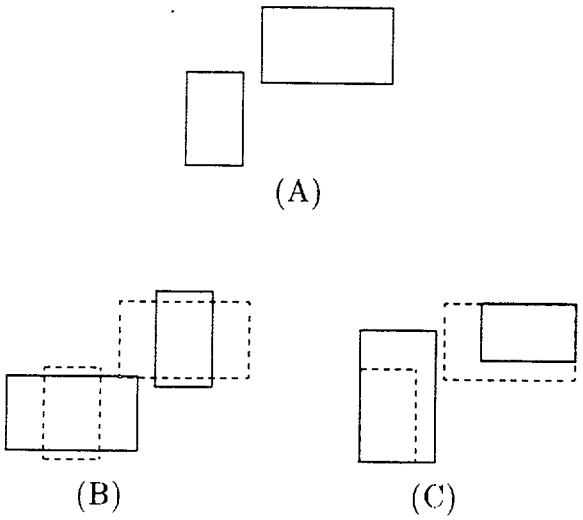


FIGURA 5.22: Tipos de intercambio de pares de objetos. (a) Posición inicial. (b) Intercambio de centros. (c) Intercambio manteniendo el borde exterior.

Transformación: todas las transformaciones ortogonales $(90°, 180°, 270°)$ son posibles para una puerta lógica o transistor, a menos que sean prohibidas por el usuario. Las coordenadas del elemento permanecen fijas durante una transformación.

Las características peculiares de los procesos tecnológicos en GaAs – TriQuint, Vitesse –, obligan a que la orientación de los transistores debe ser horizontal, restringiendo este tipo de movimiento a múltiplos de 180°. Estas características son debidas a las diferentes estructuras cristalinas del Ga y As. Ello hace que el compuesto GaAs presente diferentes distancias entre átomos de Ga y As en planos de distintas direcciones. A la hora de crear el canal la orientación de la oblea es importante.

Traslación y Transformación: Este movimiento es una combinación de una traslación y una transformación.

Intercambio de pares: El objeto a intercambiar con el objeto A se selecciona aleatoriamente de entre el conjunto de elementos que pueden moverse. Hay dos tipos de intercambio (figura 5.22), uno de los cuales lleva consigo el intercambio de los centros, con o sin información adicional de transformación. La otra posibilidad es la de mantener constante el rectángulo exterior de los objetos a intercambiar. Esta última transformación está prohibida para algunas tecnologías GaAs, puesto que se producen transformaciones no múltiplo de 180°.

5.2.2.9 Movimiento de terminales flotantes.

Una célula con terminales flotantes está restringida a tener forma rectangular. Los terminales flotantes de una célula de este tipo se sitúan sobre un número de sitios predefinidos a lo largo de los bordes de la célula, como se muestra en la figura 5.23. Este número de lugares se controla indirectamente por una variable que representa el máximo número de terminales flotantes que pueden ocupar un sitio. Cuanto más grande sea este número, serán necesarios menos sitios para mantener todos los terminales flotantes.

En cada lado de una célula con terminales flotantes, se situarán el menor número de sitios, igualmente espaciados, tal que el número de terminales que pueden ser físicamente apiñados en el lado es menor que la capacidad combinada de los lugares y cada lugar tiene una capacidad no mayor que la máxima.

Si los lugares se distribuyen simétricamente alrededor de la célula, todas las orientaciones que pueden obtenerse por reflejo o por rotaciones de 180° serán equivalentes, y el algoritmo podría tomar un tiempo considerable en la selección de una orientación entre todas las posibles orientaciones. Para romper la simetría, al lado izquierdo y al lado superior de la célula se asigna un lugar más que a los lados derecho e inferior.

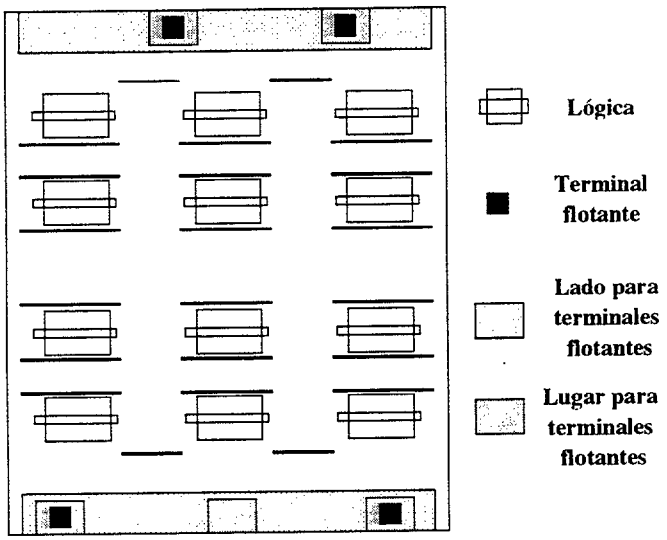


FIGURA 5.23: Célula con terminales flotantes.

Tras inicializar los lados de la célula:

- La nueva posición de un terminal flotante se selecciona aleatoriamente entre todos los lugares posibles de la célula, excepto el lugar actual que ocupa el terminal.
- Si la capacidad del lugar que va a ocupar el terminal está completa, o si está en un lado no permitido para el terminal flotante, el movimiento no se realizará.

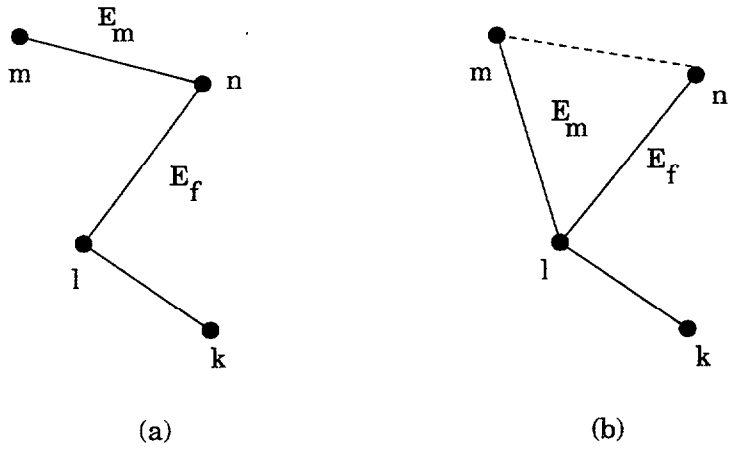


FIGURA 5.24: Ejemplo de movimiento de subnodo: (a) Árbol de expansión de partida, (b) movimiento del subnodo m–n.

- Si se rechaza el nuevo sitio, entonces se selecciona uno de los dos sitios adyacentes al sitio rechazado. Si este segundo sitio también se rechaza, entonces el movimiento queda abortado.

5.2.2.10 Movimientos de un nodo.

Cuando se estima la longitud de un nodo utilizando el método del subnodo y si el número de subnodos es superior a cuatro, la implementación **Anteo** lleva a cabo un alineamiento por enfriamiento simulado en la forma del árbol expandido de los nodos. Los pasos llevados a cabo en este tipo de movimiento son:

- Aleatoriamente, se selecciona un nodo n en el árbol de expansión con dos o más lados incidentes;
- De nuevo aleatoriamente, se seleccionan dos lados $E_{\mathrm{m}} = (n, m)$ y $E_{\mathrm{f}} = (n, l)$ que inciden en el nodo n; sea E_{m} el lado que se mueve y E_{f} el lado fijo;
- La idea es permitir que E_{m} se desplace a lo largo de E_{f} desde n a l, tal que $E_{\mathrm{m}} = (n, m)$ se reemplaza por $\hat{E}_{\mathrm{m}} = (m, l)$. Este movimiento mantiene la conectividad del grafo.

En la figura 5.24 se muestra un ejemplo del movimiento de un subnodo.

5.2.2.11 Perturbación.

Una perturbación corresponde a una pequeña desviación en la dirección de la traslación. Un "vector de perturbación" de longitud especificada se suma vectorialmente a todos los vectores de traslación. Este vector rota 90° en sentido contrario a las

```
desde i := 1 hasta n movimientos hacer  bucle interior
      para cada variable de la lista de variables de alineamiento hacer
            mover variable;
      fin para
fin desde
```

FIGURA 5.25: Lazo interno del algoritmo de alineamiento por enfriamiento simulado.

agujas del reloj, cada vez que la temperatura del algoritmo de enfriamiento se actualiza. La longitud del vector se limita internamente a un rango entre 0 y 10. Estas pertubaciones tienen dos posibles efectos: ayudan a eliminar el solapamiento en las últimas etapas del algoritmo de alineamiento por enfriamiento simulado y ayudan a conseguir una célula más cuadrada situando las células hacia las esquinas del área disponible, cuando no hay restricción de área [20].

5.2.2.12 Esquema de enfriamiento.

El algoritmo de alineamiento por enfriamiento simulado (figura 5.16) consta de dos bucles (externo e interno). Durante cada paso del bucle externo la temperatura de enfriamiento se mantiene constante hasta que se actualiza. En cada paso, del bucle interno (figura 5.25) para cada variable de una lista de todas las variables de alineamiento se intenta realizar un movimiento de acuerdo con su naturaleza. El bucle interno se repite, al menos, un número de veces controlado por un parámetro n. La temperatura utilizada en el paso $k+1$ de ejecución del bucle externo se obtiene a partir de la temperatura utilizada en el paso k multiplicando por una constante α cuyo valor por defecto es 0.9. De esta manera el algoritmo va realizando un enfriamiento progresivo con la consiguiente disminución de entropía y la posibilidad de progresar hacia un mejor "ordenamiento" final. El alineamiento termina cuando las siguientes condiciones se satisfacen:

- la temperatura alcanzada es menor que la temperatura mínima, que es un parámetro controlado por el usuario;
- el costo del colocado medido al final de los últimos cuatro pasos no ha cambiado más de un 0.1%.

Si se alcanza la temperatura mínima pero el costo varía mucho, se aconseja volver a ejecutar el algoritmo con temperatura inicial superior. Si a pesar de lo anterior, el sistema se sigue comportando de igual forma, entonces algún peso de la función de costo hace oscilar el sistema. En este caso, se aconseja ajustar los pesos.

Si la función de costo se estabiliza lejos de la temperatura mínima, la solución obtenidad por el algoritmo corresponde a un mínimo local. Dado que el algoritmo está gobernado por procesos aleatorios, es suficiente volver a ejecutarlo.

La temperatura inicial o bien se especifica o si no fuera así se determina automáticamente. Para el cálculo automático de la temperatura inicial, el algoritmo lleva a cabo un número de movimientos a temperatura infinita (representada por -0.1), y entonces se calcula la temperatura inicial basada en estadísticas sobre el costo de los movimientos llevados a cabo. Como regla general, la temperatura inicial debe ser muy alta.

5.3 Definición de canales y trazado de la alimentación.

En la definición de canales cabe distinguir los procesos orientados a la distribución de alimentación y los orientados al conexionado de las señales.

En el caso del trazado de la alimentación, cabe distinguir dos niveles de interconexión:

- el nivel intracélula o local, donde se orientan las entidades en función de la definición de los canales locales e
- intercélula o global.

El primer nivel es el que requiere la definición de canales para el trazado de los buses de alimentación, de acuerdo con la metodología. El segundo nivel no siempre es posible realizarlo, pues depende de los niveles de metalización de la tecnología. El programa **Baco** lo hemos desarrollado para definir la estrategia de trazado de alimentación local. El hecho de separar la definición de canales para la conexión de señales de la definición de canales para la alimentación, responde a la estrategia introducida en la metodología de Notación en Anillo. La forma del trazado de la alimentación y de las señales hacen incompatible la definición de canales para ambos tipos de interconexionado. Además, hay solapamientos entre los canales definidos para ambos tipos de conexiones, tal y como se muestra en la figura 5.27.

En la figura 5.26 mostramos la secuencia en la ejecución de esta actividad.

Baco es la versión de definición de canales y ruteado global de las alimentaciones a nivel intracélula. De acuerdo con la metodología de Notación en Anillo, hemos desarrollado este programa con el fin de definir los canales para ruteado de la alimentación y trazar los buses de alimentación.

Para ambas opciones de la metodología de Notación en Anillo – buses lateral y centrales –, el proceso llevado a cabo por **Baco** es el mismo y se resume a continuación:

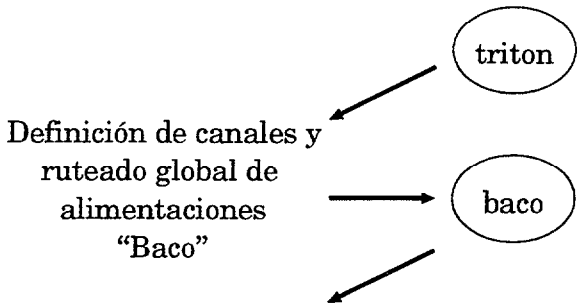


FIGURA 5.26: Actividad de definición de canales y ruteado global de la alimentación local.

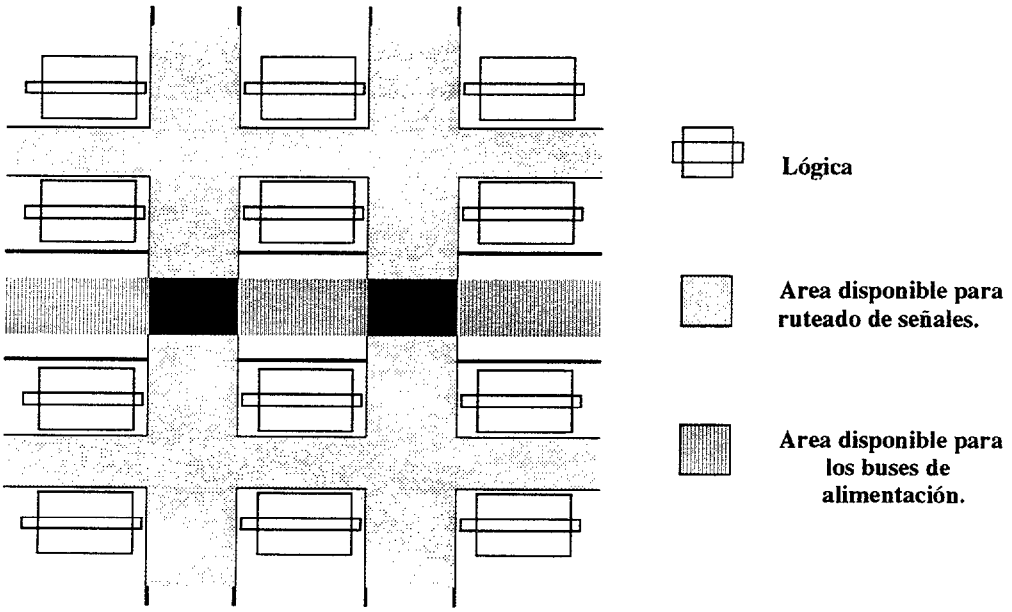


FIGURA 5.27: Definición de canales para la alimentación.

- Las puertas lógicas o los transistores que forman parte de una fila – recordemos que **Tritón** lleva a cabo la transformación de un colocado a una estructura matricial en forma de filas y columnas –, se agrupan con la fila adyacente con la que comparten alimentación. Si no es posible lo anterior, por haber un número impar de filas, el procedimiento continúa.

- Las áreas horizontales que quedan entre los agrupamientos anteriores definen los canales por los que los buses de alimentación van a trazarse. Los reagrupamientos permiten conocer exactamente las necesidades de alimentación a nivel de intensidad y a partir de aquí se dimensionan los anchos de los buses para que la caída de tensión sea menor que un cierto porcentaje del margen de ruido (1%).

5.4 Definición de canales para señales.

En general la tarea de definición de canales para señales no es más que la partición del área de ruteado disponible en regiones más pequeñas para las subsecuentes fases de ruteado global y detallado. Una vez han sido determinadas las regiones, éstas tienen que ser ordenadas de forma tal que cada una pueda conectarse siguiendo una secuencia donde se realice su ajuste independientemente de las regiones previamente ruteadas.

Otten [73] presenta un algoritmo de definición de canal que produce una ordenación del trazado de conexiones factible sobre un tipo de disposición espacial de objetos, denominado estructura de rebanada rectilínea. La ventaja del método de rebanada es que induce una descomposición jerárquica sencilla del problema de ruteado de señales en subproblemas independientes. La célula se divide por la primera rebanada en dos bloques; el procedimiento de rebanar se repite recursivamente en todos los subbloques. Cada rebanada representa un área en la cual el conexionado es simple. El orden en el que se encuentran las rebanadas, es el inverso al orden de realización del conexionado.

En la figura 5.28 mostramos la secuencia en la ejecución de esta actividad.

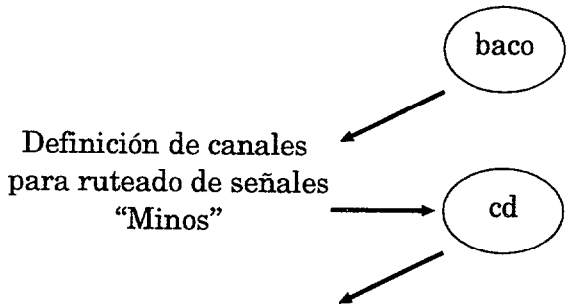


FIGURA 5.28: Estructura de la actividad de definición de canales para señales y alimentación.

Minos es el programa de definición de canales para ruteado de señales. En la distribución **OCT** disponemos de la implementación del algoritmo de Otten, para la definición de canales, denominada **atlas** [20]. Sobre esta implementación hemos introducido una modificación. Se trata de disminuir la densidad de interconexión de los canales de señales, en las zonas de trazado de la alimentación. En la figura 5.29, se muestra el objetivo perseguido.

Dada una célula, la definición de los canales por el programa **Minos** – adaptación para GaAs de **atlas** – obtiene dónde se localizan los canales de ruteado de señales y en qué orden deben rutearse. A diferencia de **atlas**, Minos no incluye en la definición de canales para ruteado de señales, las zonas dedicadas a alimentación disminuyendo la densidad de interconexiones en estas zonas. De esta forma, se obliga al ruteador

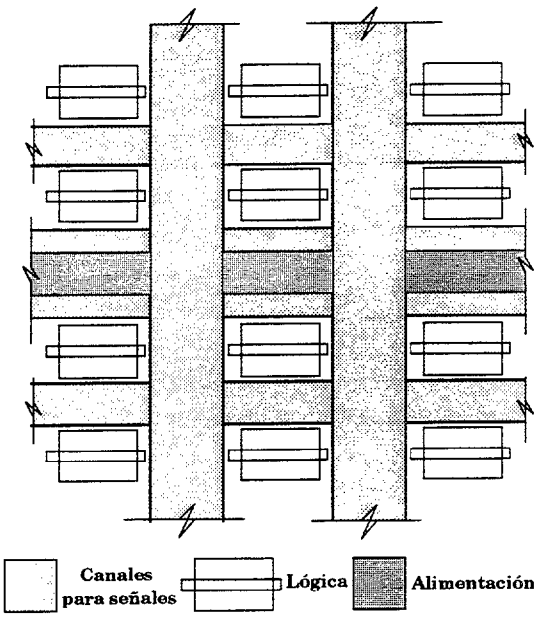


FIGURA 5.29: Definición de canales para ruteado de señales.

global de señales a utilizar los canales para ruteado de señales horizontales (véase figura 5.29), y todos los canales verticales.

Minos tratará de utilizar un árbol para la definición de canales. Cuando es imposible crear el árbol, se procede a utilizar canales irregulares, es decir tipo "Z". En el árbol de canales, los canales verticales quedan definidos cuando al menos uno de los lados del canal coincide con el borde de una entidad. En la figura 5.30 se representa el árbol de definición de canales.

5.5 Trazado global de señales.

Después de la definición y ordenamiento de los canales para señales, los canales deben transformarse antes de proceder al ruteado detallado. El grafo de canales, con las posiciones de los terminales en sus arcos, y una lista de nodos, constituyen los datos de entrada necesarios para el trazado global.

En el grafo de canales, los nodos representan las intersecciones entre canales, mientra los arcos representan canales o secciones de ellos (véase figura 5.31). A cada arco en el grafo se le asigna un peso que representa el máximo número de pistas que podemos trazar en el canal. Una vez el grafo de canales se haya construido, todos los terminales de cada entidad que tengan conexión se proyectan al arco más próximo en el grafo. Los pesos en los arcos, se interpretan como restricciones de capacidad. Es importante tener en cuenta que la entrada al ruteador global es completamente simbólica, y por tanto totalmente independiente de las reglas de diseño de fabricación. Cuando finaliza el ruteado global, cada nodo se representa por una secuencia de subnodos, cada uno de los cuales se asigna a uno de los arcos

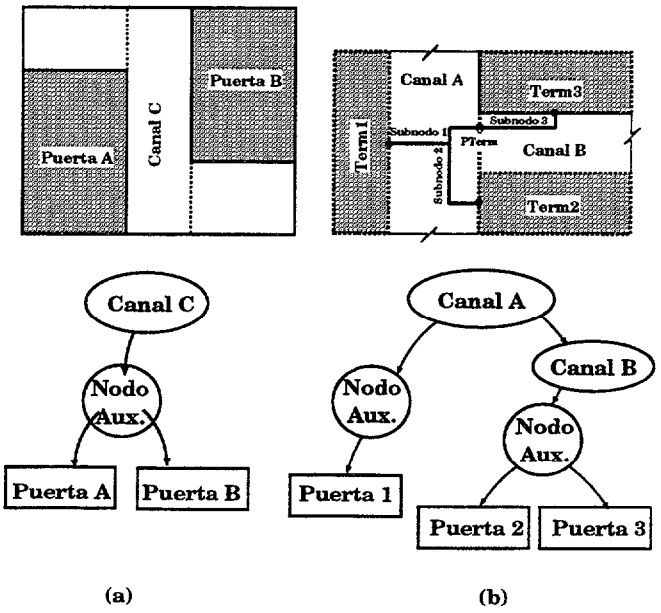


FIGURA 5.30: Árbol de definición de canales: (a) Canal tipo "Z", (b) canales rectangulares.

del grafo de canales. Cuando existe un nodo que pasa de un canal a otro se crea un pseudonodo. En la figura 5.32 se muestra un ejemplo aclarativo.

Después del ruteado global, cada nodo consta de un número de segmentos (correspondientes a las combinaciones de los terminales tomados de dos en dos para cada subnodo), cada uno de los cuales se asigna a un canal. La conectividad entre nodos se obtiene con pseudoterminales situados en la intersección del canal. Estos pseudoterminales se consideran como lugares para conectores que debe crear el ruteador detallado. Ni los pseudoterminales ni los pseudonodos tienen vinculada ninguna geometría.

En el ruteado global es necesario minimizar la longitud de los nodos, eligiendo la posición de los pseudoterminales de forma adecuada y además minimizar el número de cambios de metales. Se utilizaron dos ruteadores globales, uno procedente de una herramienta de ruteado de células estándares muy conocida **TimberWolfMC** [87], y otro **mercury** [49] que minimiza la longitud de los nodos de interconexión y los cambios de metales. Ambos ruteadores globales pertenecen a la distribución **OCT**. Para poder utilizar el ruteador **TimberWolfMC** hemos desarrollado una interfaz entre la base de datos de **OCT** y la entrada al ruteador. Respecto a **mercury** hemos tenido que modificar el código en la sección de proyección de terminales sobre los arcos del grafo de canales. En ambas implementaciones hemos tenido en cuenta que podemos trazar las interconexiones cercanas (entre puertas adyacentes) en metal de puerta, y utilizar metal de nivel 1 para ruteado vertical e interconexiones lejanas. En la figura 5.33 mostramos la secuencia en la ejecución de esta actividad.

Jano lo hemos desarrollado como interfaz del sistema de definición de canales al

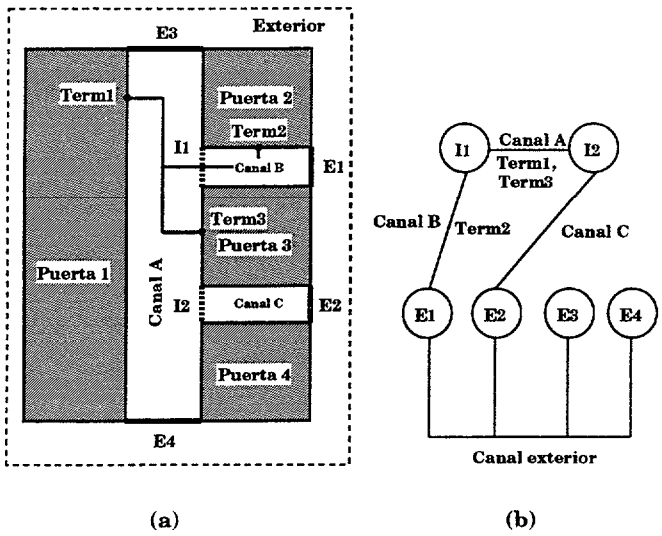


FIGURA 5.31: Grafo de canales: (a) Definición, (b) grafo.

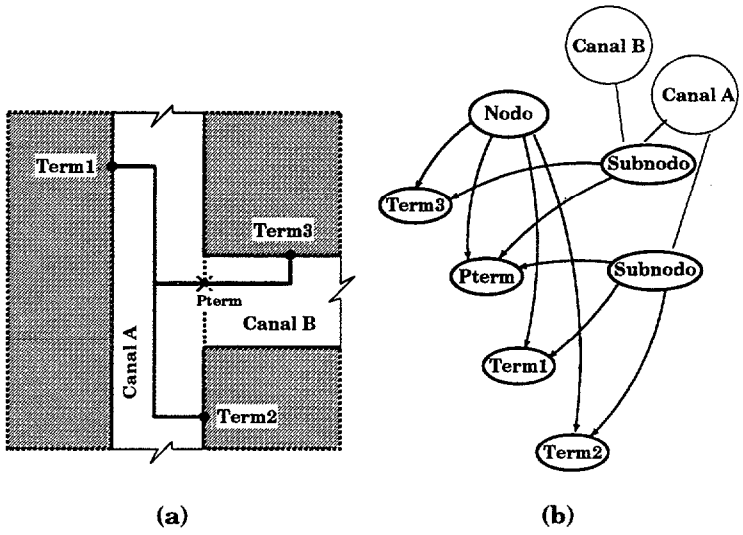


FIGURA 5.32: Ruteado global de un nodo: (a) Conectividad entre terminales, (b) árbol de ruteado global del nodo.

ruteador global para señales **TimberWollfMC**. **Jano** consta de dos fases: la primera fase genera los datos en un formato adecuado para el uso del ruteador global. En la segunda fase, **Jano** lee los resultados producidos por el ruteador global y los almacena en la base de datos de **OCT** para que puedan ser utilizados por el ruteador detallado.

Pan es una adaptación de **mercury** del entorno **MOSAICO**, que realiza el ruteado global optimizando la longitud de los nodos. La adaptación la hemos realizado en la fase de asignación de terminales a los arcos del grafo del canal. En la figura 5.34, mostramos la diferencia que resulta de rutear con **mercury** y con la adaptación **Jano**. Mientras que **mercury** proyecta el punto medio del terminal de conexión en el canal horizontal adyacente, con **Jano** proyectamos el terminal sobre los canales adyacentes verticales.

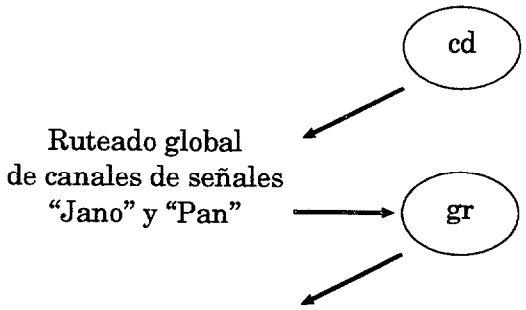


FIGURA 5.33: Estructura de la actividad de ruteado global para señales.

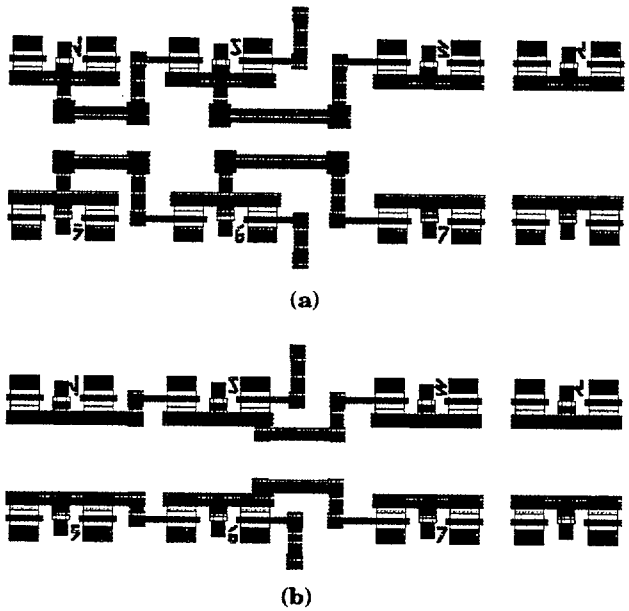


FIGURA 5.34: Diferencia entre el ruteado global con (a) **mercury**, y (b) **Jano**.

5.6 Trazado detallado de las conexiones.

A partir de la información proporcionada por el ruteador global y de la fase de definición de canales y ordenamiento, se generan especificaciones de canales para la interfaz **Teseo**. **Teseo** nos permite llevar acabo el trazado detallado de las conexiones de cada canal utilizando la información y las reglas de diseño de la tecnología. Puede manipular regiones con lados irregulares, y conexiones con diferentes anchos.

Teseo lo hemos desarrollado a partir de la interfaz **Spider** del entorno **MOSAICO**. **Spider** es un traductor del formato de datos tipo **OCT** a los ruteadores simbólicos **YACR** [77] y **mighty** [90], en ambas direcciones. A partir de la descripción de un canal, se extrae una rejilla simbólica no uniforme. Las líneas que constituyen la rejilla, se posicionan de forma que se obtenga el mejor alineamiento posible con la posición de los terminales a rutear. El problema que presenta **Spider** es que depende fuertemente de la tecnología. Básicamente los defectos son:

- Sólo permite las siguientes combinaciones de metales:
  - Polisilicio vertical y metal 1 horizontal,
  - metal 1 vertical y metal 2 horizontal, y
  - metal 2 horizontal y metal 1 vertical.
- Transforma los terminales de polisilicio tal y como se muestra en la figura 5.35. Si el canal es horizontal o el ruteador es del tipo *switch–box*, los terminales en polisilicio se cambian a metal de nivel 1. Si el canal es vertical, antes de rotarlo para convertirlo en horizontal (cambio de los metales de interconexión) los terminales de polisilicio se cambian a metal de nivel 2.

Estas dos actuaciones no están de acuerdo:

- Por una parte con la filosofía de trazar conexiones con cualquier combinación de dos metales, compatibles con los terminales de las puertas o de la célula y,
- por otra parte, en el caso de que se utilice metal de puerta para conexión, no se ha de cambiar esta capa, salvo que la interconexión sea demasiado larga.

En la figura 5.36 mostramos la secuencia en la ejecución de la actividad para trazado detallado de conexiones.

Egeo obtiene información de cada canal de la base de datos en el orden apropiado y genera la información necesaria para el ruteador detallado.

Teseo es la interfaz con los ruteadores simbólicos **mighty** y **YACR**.

YACR es un ruteador de canales de dos capas. Es rápido y potente, dado que siempre traza rutas en los canales con el mínimo número de pistas de conexionado. Esta especialmente orientado para la conexión de señales en canales entre células estándares (*channel–routing*).

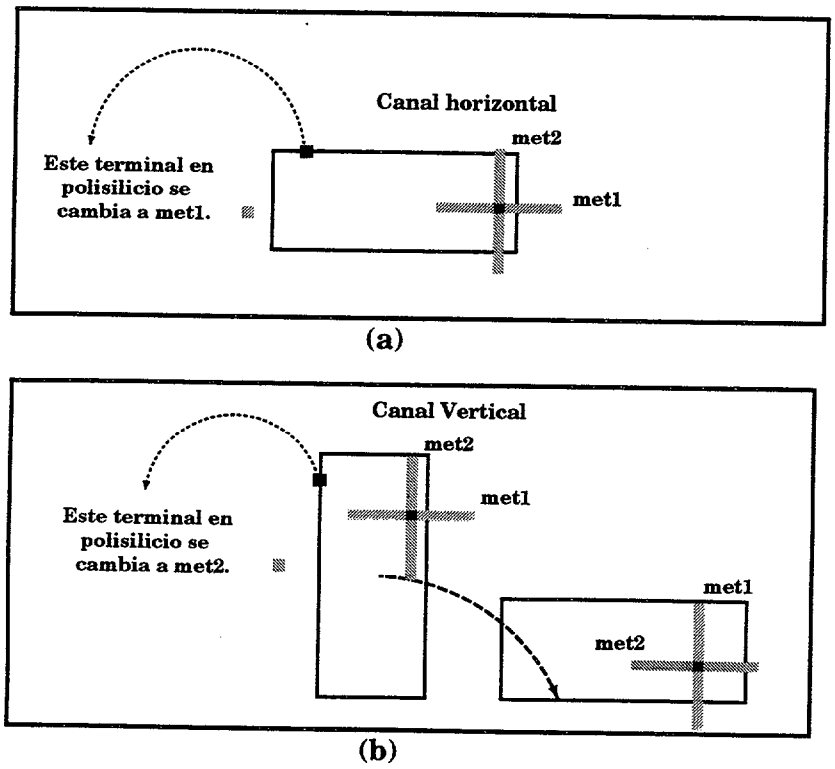


FIGURA 5.35: Transformación realizada por **Spider** de los terminales de polisilicio, en un canal: (a) Canal horizontal o ruteador *switch-box*, (b) canal vertical.

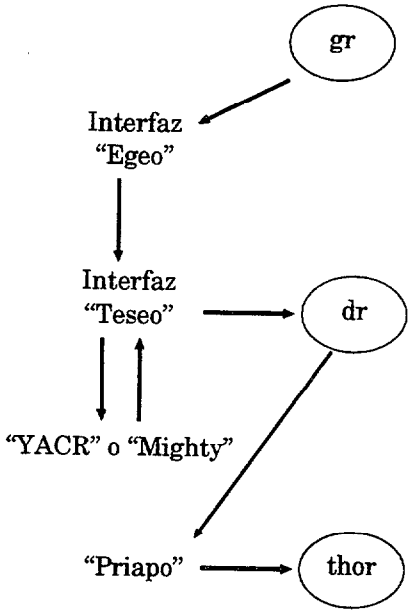


FIGURA 5.36: Estructura de la actividad de ruteado detallado para señales.

Mighty es un ruteador tipo *switch–box* y de laberinto. Se utiliza para trazar conexiones en los canales con terminales en todos sus lados, con el mínimo número de pistas, aunque consume más recursos de computación que YACR.

Priapo realiza el trazado de las alimentaciones globales, locales y lógicas.

Finalizado el ruteado detallado para las señales, volvemos al conexionado de los terminales de alimentación de cada puerta con el bus de alimentación más cercano. Para realizar lo anterior hemos desarrollado la utilidad **Priapo**. La secuencia algorítmica llevada a cabo se muestra en la figura 5.37.

```
crear la lista de buses de alimentación local;
crear la lista de puertas a conectar con el bus de alimentación;
ordenar los elementos de la lista de puertas en función
        de la fila y dentro de cada fila en función de la columna;
para cada puerta de la lista de puertas a conectar hacer
empezar
        determinar los terminales que se conectan a alimentación;
        encontrar en la lista de buses de alimentación el más cercano;
        obtener la posición de la vía de conexión entre metal 1 y metal 2;
        colocar la vía 2 de conexión;
        almacenar la vía 2 en la lista de buses locales;
        conectar en metal 1 desde la vía 2 al terminal de la puerta;
terminar
fin para
ordenar las vías 2 de la lista de buses de alimentación por fila y
dentro de cada fila por columna;
para cada fila de vías 2 de la lista de buses de alimentación hacer
empezar
        para cada columna de vías hacer
        empezar
                conectar en metal de nivel 2 via 2 con via 2 anterior;
        terminar
        fin para
terminar
fin para
```

FIGURA 5.37: Algoritmo de Priapo.

5.7 Minimización de vías y compactación.

La entrada para la minimización de vías es un trazado sin jerarquía que contiene los dispositivos de la célula y el conexionado asociado. En este punto, todas las conexiones horizontales son de un metal distinto a las conexiones verticales. Reasignando a algunos segmentos la capa opuesta, el minimizador de vías reduce el número total de vías, minimizando los cambios de metal. Las localizaciones de los segmentos no

cambian, sólo las capas asignadas, y en todo momento el cambio tiende a disminuir la resistencia. Durante la especificación de los terminales de una célula, normalmente se debe utilizar más de una máscara, puesto que estas asignaciones de los terminales a máscaras determinan durante la minimización qué nivel de metal debe utilizarse para conectar ese terminal.

Todos los pasos en **OLYMPO** utilizan representaciones a nivel simbólico. El espaciamiento–compactación del trazado simbólico es de hecho una parte esencial del sistema, entre otras razones, puesto que:

Primero, todos los diseños deben espaciarse para asegurar que son correctos desde el punto de vista de las reglas de diseño; esto tiene la ventaja de eliminar la comprobación de tales reglas, puesto que el trazado es correcto por construcción.

Segundo, el uso del trazado simbólico generalizado, proporciona un mecanismo para producir diseños independientes del proceso tecnológico y de su evolución, aunque dependientes del dispositivo.

La técnica de compactación utilizada en **OLYMPO** es la misma que la utilizada en **MOSAICO**, la cual es capaz de actualizar las primitivas simbólicas de trazado (transistores, contactos, vías), así como el espaciamiento entre ellas. **Sparcs** [12] es un espaciador–compactador simbólico, basado en la determinación de rutas críticas, usado extensamente en el entorno **OCT**.

5.8 Obtención del diseño *full–custom.*

En esta fase se traduce el trazado simbólico a físico, ejecutando una serie de operaciones sobre las máscaras (véase figura 5.38) con el fin de eliminar muescas, entrantes, etc .... A pesar de que el proceso de fabricación rellenará los espacios del tipo que

```
para cada máscara de la lista de máscaras para conexión hacer empezar
    extensión := regla de diseño de separación - unidad de resolución;
    extender los polígonos de la máscara según extensión;
    encoger los polígonos de la máscara según extensión;
fin para
```

FIGURA 5.38: Procesamiento de traducción simbólico a físico.

se muestran en la figura 5.39, es una buena práctica remitir el diseño a fabricación sin presuponer lo anterior. Además, los propios comprobadores de reglas de diseño, tienen la obligación de detectar este tipo de geometrías.

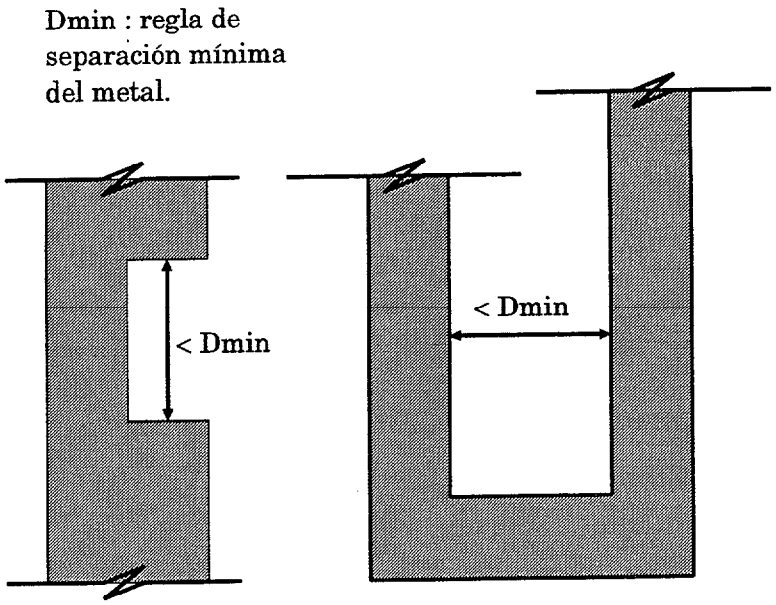


FIGURA 5.39: Ejemplos de muescas en un layout.

5.9 Traducción a otros entornos.

La distribución **OCTTOOLS 5.1**, dispone de utilidades de conversión bidireccional a otros formatos tales como **CIF**, **EDIF** etc ..., así como una interfaz bidireccional entre **MAGIC** y **OCT**. Esta distribución también puede generar información en formato **SILOS**, **HILO** y **SPICE** entre otros. En el entorno de diseño *full-custom* para tecnología GaAs [17, 16] desarrollado por nosotros en Cadence/Edge, y referido en el capítulo cuarto es posible realizar la captura de un diseño tanto a nivel de esquemático como a nivel de trazado simbólico y físico, su análisis, optimización y verificación, así como su simulación eléctrica; y está conectado a este conjunto de herramientas **OLYMPO** a través de una interfaz bidireccional **Cadence** y **OCT**.

El planificador de **OLYMPO** produce especificaciones a nivel de posicionamiento de los terminales dada una relación de aspecto y un área de la célula. La entrada al generador de células es la combinación de un listado de nodos de transistores o puertas lógicas, y la información de la geometría y posicionamiento de los terminales. La salida del planificador se utiliza en la fase de colocado de **OLYMPO**. Esta estrategia de generación combina las ventajas propias de una metodología *top–down* con la precisión de un planteamiento *bottom–up*. **OLYMPO** está concebido para el colocado de puertas lógica y transistores al estilo de la metodología de Notación en Anillo. Esto lo diferencia de otras herramientas para planificación y colocado de módulos – **puppy** planifica y coloca módulos en **MOSAICO** –.

El colocado de puertas y transistores lo realizamos en todo el espacio de diseño

que abarca la célula. Una vez hemos obtenido el colocado de los elementos, traducimos sus coordenadas (x, y) del espacio de diseño, a fila y columna (r, c) de una matriz. Con esta representación manejamos la lógica agrupada en filas y realizamos la definición de canales de buses de alimentación al estilo de la metodología de Notación en Anillo.

El trazado de las interconexiones en **OLYMPO** sigue el planteamiento definido en la metodología de composición geométrica propuesta. En la definición de canales para trazado de señales, reducimos la densidad de cableado de señales en las zonas dedicadas a la alimentación. El trazado de la alimentación es independiente del trazado de las señales. Aunque hay una estructuración de la alimentación en tres niveles: a nivel de transitor, local y global; se ha integrado el trazado en dos fases.

Con el fin de conseguir la independencia de las reglas de diseño, el sistema completo **OLYMPO** opera a un nivel simbólico, más que a nivel físico. Hemos desarrollado dentro del núcleo de la distribución **OCT** diferentes paquetes de librerías, con el fin de disponer en el nivel más externo, de generadores de dispositivos y primitivas propios de familias lógicas en GaAs. Antes de realizar la generación de máscaras se lleva acabo la compactación y espaciamiento para asegurar que el trazado es correcto, de acuerdo con las reglas de diseño. Toda la información referente a reglas de fabricación se soporta mediante un sistema de ficheros en *ASCII*, configurables y modificables por el usuario. Sobre el trazado simbólico definitivo realizamos la transformación a geométrico eliminando imperfecciones – muescas – resultado de la compactación.

Capítulo 6

Validación de la herramienta de compilación de células en OLYMPO.

El proceso de evaluación de una herramienta es una tarea que depende básicamente de su uso y por tanto es un hecho contínuo, dinámico y a largo plazo. A pesar de esto, se propone validar la herramienta implementada desarrollando las microrrotaciones, que forman parte de la sección de procesamiento de un procesador para la evaluación de la CFFT (*Complex Fast Fourier Transform*) en base 2 y precisión de 16 bits, basado en el algoritmo CORDIC (*COordinate Rotation DIgital Computer*).

Los resultados de esta evaluación se presentan en términos de prestaciones (área, potencia y retardo) alcanzadas en ambos diseños manual y automático; y en la medida de tiempos de diseño y procesamiento en uno y otro caso.

Se ha demostrado que el algoritmo CORDIC es el más eficiente para la computación de funciones básicas de tipo trigonométrico, exponencial y logarítmico. Aunque el algoritmo se propuso en 1959 [105], la evolución en el diseño VLSI ha abierto de nuevo el interés a un cierto número de aplicaciones. Además de su importancia e interés, se ha elegido como elemento de referencia para el proceso de evaluación porque sobre su arquitectura e implementación existen resultados extensos y muy actuales en nuestro grupo [6, 85].

6.1 Arquitectura del procesador FFT basada en CORDIC.

La Transformada de Fourier Compleja juega un papel clave en el campo del procesamiento de señales. En 1965 Cooley y Tukey [25] desarrollaron un algoritmo para acelerar el cálculo de la Transformada de Fourier Discreta, que supuso una revolución en la computación numérica de las transformadas ortogonales. Este algoritmo, conocido como Transformada Rápida de Fourier en base 2 y decimación en tiempo, está basado en el método de doblamiento sucesivo [2]. Si se utiliza el algoritmo de

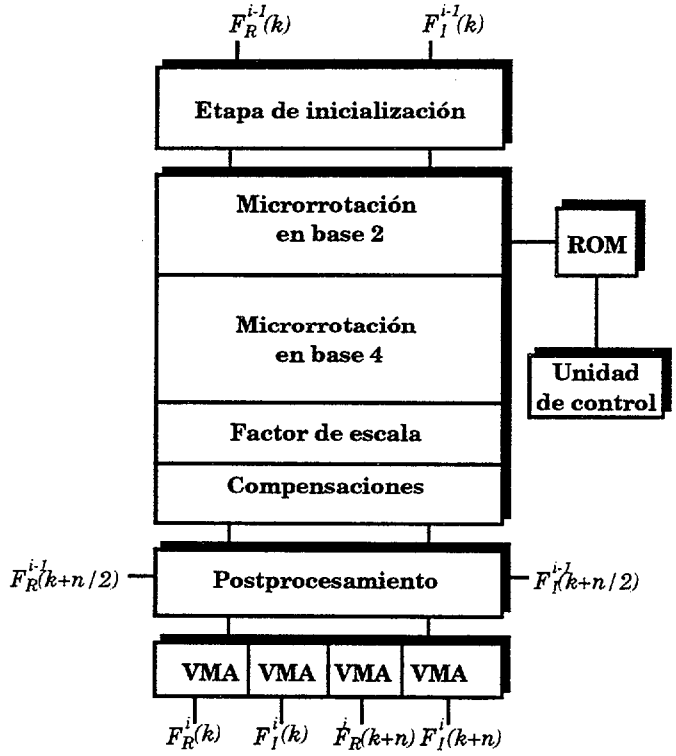


FIGURA 6.1: Diagrama de bloque de la computación FFT basada en CORDIC.

Cooley y Tukey, el procesador para la CFFT debe incluir dos subprocesadores:

La sección de procesamiento encargada de computar y,

la sección de ruteado para la recirculación de los datos.

La sección de procesamiento consta de un procesador CORDIC y una unidad de postprocesamiento. El cálculo de la CFFT se basa en una pareja de puntos $(F_R^{i-1}(k), F_I^{i-1}(k)), (F_R^{i-1}(k+n/2), F_I^{i-1}(k+n/2))$. En el CORDIC se procesa un conjunto de puntos cuyo resultado es un vector $(F_R^i(k), F_I^i(k)), (F_R^i(k+n), F_I^i(k+n))$. Este proceso es independiente de la transformada ortogonal implementada en la unidad de postprocesamiento.

El algoritmo CORDIC es un medio eficiente para calcular funciones trigonométricas rotando el vector un cierto ángulo, especificado por sus coordenadas. Esta rotación se obtiene llevando a cabo un número de microrrotaciones a través de ángulos de rotación elementales ϕ_i, resultantes de la descomposición del ángulo de rotación total ϕ. El carácter específico de la arquitectura se fundamenta en el conocimiento a priori de los ángulos que intervienen en todas las etapas de la CFFT [115]. El número de microrrotaciones del algoritmo CORDIC se puede reducir mediante la utilización de representaciones mixtas en base 2 y 4. Además de las microrrotaciones CORDIC convencionales existen microrrotaciones adicionales para reducir el factor de escala a la unidad. Los detalles del algoritmo CORDIC y arquitecturas posibles caen fuera del objeto de esta tesis, pero pueden consultarse en [2, 6, 85, 115]

La planificación del CORDIC se muestra en la figura 6.1. Consta de una estapa de inicialización, seguida por las microrrotaciones en base 2 y en base 4 y las etapas de compensación.

La operación básica de la microrrotación CORDIC en base 2 es la suma/resta. Para acelerar el procesamiento, se utiliza aritmética redundante y segmentación. En notación redundante cada vector se representa por sus palabras de *suma* y *acarreo*. Por lo tanto, la célula básica para la implementación de microrrotaciones es un sumador/restador $4-2$. Para la segmentación se necesita un registro.

Un procesador CORDIC para el cómputo de 1024 puntos CFFT de datos de 16 bit incluirá más de 600 sumadores/restadores $4-2$ y 1250 registros. La necesidad de un diseño optimizado de ambos elementos es evidente. La complejidad del procesador hace que la optimización deba hacerse en área y potencia. Esta filosofía permite integrar la unidad de procesamiento en un solo circuito integrado.

Para la realización de las microrrotaciones hemos tomado la familia lógica DCFL, aunque la OR exclusiva se ha realizado con la estructura O–A–I de la lógica SDCFL, que mostramos en la sección 3.1. Esta combinación proporciona el mejor compromiso entre prestaciones y robustez [85].

Los sumadores/restadores 4–2 en notación redundante se implementan usando dos niveles de sumadores *carry–save* (CSA) como se muestra en la figura 6.2. En el primer sumador los operandos A, B y C se suman. La segunda etapa añade la *suma* y el *acarreo* de la etapa anterior a D. A la salida del sumador/restador el dato está en forma de representación redundante.

6.2 Generación automática.

En este apartado se expone la forma en la que hemos realizado la generación automática de la unidad CORDIC, que constituye la unidad principal del procesador CFFT, basándonos en las herramientas desarrolladas. Para no hacer muy extenso el capítulo nos centraremos en las microrrotaciones en base 2, porque tanto el resto de las microrrotaciones (radix4 y escalado), como la unidad de postprocesamiento siguen el mismo procedimiento.

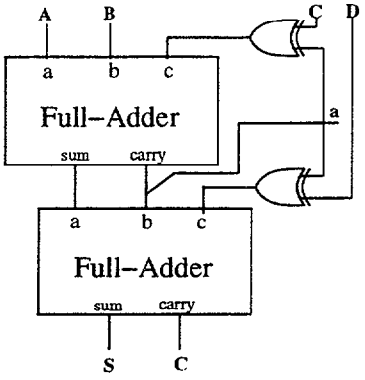


FIGURA 6.2: Sumador/restador 4 – 2.

Con el fin de evaluar los procesos de colocado y de trazado de conexiones de forma independiente, hemos realizado el conexionado automático partiendo del colocado de puertas que forma la célula ejemplo en [85]. A posteriori, hemos relizado el proceso completo (planificación, colocado y ruteado). En la descripción siguiente nos hemos centrado en este último proceso.

6.2.1 Lista de nodos a nivel de puertas lógicas.

Como interfaz de entrada para la generación automática hemos elegido el entorno de captura a nivel de esquemático que hemos desarrollado sobre Cadence y que fué presentado en la sección 4.1.1.1. En la figura 6.3 presentamos el esquemático que corresponde al sumador/restador 4–2. El dimensionamiento previo de este esquema lo hemos realizado en base al estudio presentado en [31]. Las dimensiones asignadas inicialmente a las puertas NOR e inversores son:

- $\frac{W_{pu}}{L_{pu}} = \frac{2}{2.4}(\frac{\mu m}{\mu m})$ para el transistor de *pull–up*, siendo W_{pu} el ancho del transistor y L_{pu} la longitud de puerta, y
- $\frac{W_{pd}}{L_{pd}} = \frac{10}{1.2}(\frac{\mu m}{\mu m})$ para el transistor de *pull–down*, siendo W_{pd} el ancho del transistor y L_{pd} la longitud de puerta.

Para las puertas OR estas dimensiones son:

- $\frac{W_{pu}}{L_{pu}} = \frac{5}{1.2}(\frac{\mu m}{\mu m})$ para el transistor de *pull–up* y,
- $\frac{W_{pd}}{L_{pd}} = \frac{3}{1.2}(\frac{\mu m}{\mu m})$ para el transistor de *pull–down*.

En la figura 6.4 se muestra el resultado de la simulación realizada con HSPICE. En esta simulación no se tiene en cuenta las capacidades parásitas introducidas por el cableado. A partir del esquema a nivel de puertas se obtienen dos listas de nodos: una constituida por las puertas lógicas que configuran la célula y otra a nivel

FIGURA 6.3: Esquemático a nivel de puertas del sumador/restador 4–2.

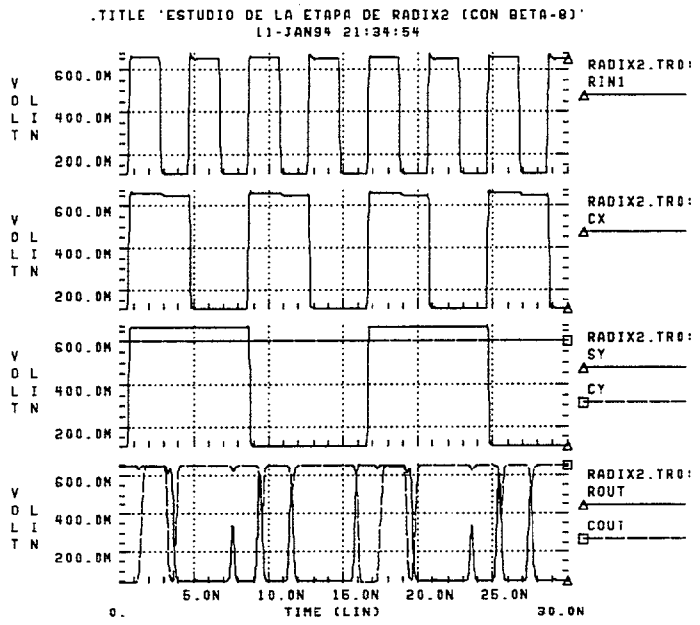


FIGURA 6.4: Resultados de la simulación con HSPICE antes del trazado geométrico.

de transistores en formato HSPICE. Ambas especificaciones de la lista de nodos (a nivel de puertas lógicas y a nivel de transistores) son fácilmente traducibles al formato intermedio BDNET (véase figura 6.5). Para ello hemos desarrollado diversas utilidades en **Skill** y **C**. Las especificaciones a nivel de transistor generalmente no ofrecen buenos resultados; es preferible introducir una descripción a nivel de puertas. En la figura 6.5 se representa parcialmente una lista de nodos a nivel de puertas lógicas.

Esta lista constituye un bit de la microrrotación bajo estudio, contituida por 91 transistores E/DMESFETS, en lógica SDCFL, agrupados en 34 puertas lógicas. En la cabecera se describe el modelo de lógica que se va a generar, indicando la tecnología, el tipo de vista y el tipo de trazado (simbólico, físico, esquemático, etc ...). Posteriormente, se definen los nodos de los terminales de señal y su tipo (entrada, salida, entrada/salida). En esta fase no se especifican los terminales para alimentación, puesto que la propia metodología de diseño y sus realizaciones (buses laterales y centrales), definen la conectividad y posición de los buses de alimentación de forma implícita.

Una vez definido los nodos de los terminales de señal, se especifican los dispositivos (transistores, contactos, vías, puertas), indicando el nombre del dispositivo y la conectividad. Obsérvese, que los nodos de alimentación de las puertas figuran como "no conectados", pues la especificación de la conectividad es implícita en la metodología propuesta. Una vez escrita esta lista de nodos, se procesa mediante la herramienta **bdnet** que comprueba la sintaxis, la existencia de las entidades descritas, la conectividad, y se genera la vista lógica de la célula a generar.

```
model radix2AA:logic;

TECHNOLOGY H_GaAsII;
VIEWTYPE SYMBOLIC;
EDITSTYLE SYMBOLIC;

INPUT  rin1;
INPUT  cx;
INPUT  sy;
INPUT  cy;
INPUT  alpha;
INPUT  noalpha;
OUTPUT out_rout;
OUTPUT cout;

INSTANCE tech/H_GaAsII/contacts/met1.18x18:physical
          NAME = rout CONNECTOR
          t # noalpha ;
                              ...
                              ...
                              ...

INSTANCE tech/H_GaAsII/dcfl/nor.10x10.30x6:physical
         NAME = nor14 PROMOTE
         inA:17;
         inB:rin1;
         out:6;
         sourceA:UNCONNECTED;
         sourceB:UNCONNECTED;
         drain:UNCONNECTED;

INSTANCE tech/H_GaAsII/dcfl/nor.10x10.30x6:physical
         NAME = nor1 PROMOTE
         inA:cy;
         inB:noalpha;
         out:35;
         sourceA:UNCONNECTED;
         sourceB:UNCONNECTED;
         drain:UNCONNECTED;

ENDMODEL;
```

FIGURA 6.5: Lista de nodos a nivel de puertas: formato "bdnet".

Para especificar las posiciones de las diferentes entidades, basta incluir en el fichero anterior la siguiente sintaxis $[X_{Coordenada}, Y_{Coordenada}]$, en la descripción de cada entidad, como se muestra en la figura 6.6.

```
INSTANCE tech/H_GaAsII/dcfl/nor.10x10.30x6:physical NAME = nor1 PROMOTE
         [400, 1200]
         inA:cy;
         inB:noalpha;
         out:35;
         sourceA:UNCONNECTED;
         sourceB:UNCONNECTED;
         drain:UNCONNECTED;
```

FIGURA 6.6: Especificación de la posición en formato "bdnet".

6.2.2 Terminales y relación de aspecto.

La especificación de la relación de aspecto se debe realizar antes de proceder a posicionar los terminales de la célula a sintetizar. La especificación de la relación de aspecto tiene sentido sobre un área específica, que se puede estimar de muy diversas formas. El entorno dispone de un estimador de área para lógica cableada, basado en una expansión de las células para acomodar el trazado de conexiones, tal como se presenta en la ecuación 6.1, y teniendo en cuenta el área disponible para las alimentaciones expuesta en la sección 5.2.

$$\text{expansión} \quad \propto \quad \text{factor de expansión} \times \log(\text{número de terminales} + 1) \qquad (6.1)$$

La especificación de los terminales (posición y dirección) se realiza con una sintaxis preestablecida, donde se describe la posición de los terminales y el objeto de trazado que representa (trozo de metal, vía). La posición se puede dar respecto a cada una de las esquinas de la célula de forma relativa, o bien de forma absoluta. Se ha de definir la naturaleza del terminal (señal, reloj, alimentación, GND), la dirección de acceso (entrada, salida, entrada/salida), el lado de la célula donde se posiciona (superior, inferior, izquierda, derecha), tal y como se muestra en la figura 6.7. Finalizada la especificación, se procesa la información mediante la herramienta **padlist** del entorno **OCT** pasando a formar parte de la estructura del diseño, en la base de datos y, en la vista correspondiente.

La posición de los terminales puede establecerse con la herramienta de planificación y colocado. Si se conoce el entorno que rodea a la célula, y especificando los terminales como flotantes, se realizan alineamientos por enfriamiento simulado, obteniéndose la posición de éstos. El resultado obtenido, situaría la especificación en un punto similar a lo definido en la sección anterior.

```
        TERM_RELATIVE_POSITION 0.20
        TERM_RELATIVE_POSITION_STEP 0.20
        TERMTYPE SIGNAL
        DIRECTION INPUT
        TERM_EDGE TOP
FORMAL_TERMINAL rin1
FORMAL_TERMINAL cx
FORMAL_TERMINAL sy
FORMAL_TERMINAL cy
        TERM_RELATIVE_POSITION 0.33
        TERM_RELATIVE_POSITION_STEP 0.33
        DIRECTION OUTPUT
        TERM_EDGE BOTTOM
FORMAL_TERMINAL cout
FORMAL_TERMINAL rout
        DIRECTION INPUT
        TERM_EDGE LEFT
FORMAL_TERMINAL alpha
FORMAL_TERMINAL noalpha
```

FIGURA 6.7: Especificación de la posición de los terminales de I/O.

6.2.3 Procedimiento de planificación y colocado.

Definida la posición de los terminales y especificadas las puertas se procede al colocado de estos últimos mediante alineamiento por enfriamiento simulado. Las mismas herramientas (**Atenea, Rea, Anteo**) para planificación de la posición de los terminales, se utilizan para obtener un colocado de los dispositivos.

En la figura 6.8, se muestra el resultado del enfriamiento simulado aplicado a los dispositivos que conforman la célula. En este proceso, se utiliza toda el área disponible de la célula para posicionar las puertas lógicas, lo que conlleva a un posterior procesamiento para definir los canales de señales y alimentación, planificados en la metodología propuesta. En los procesos de alineamiento por enfriamiento si-

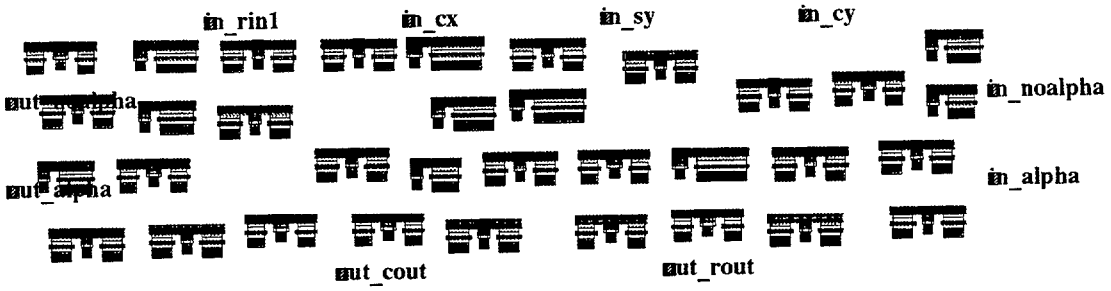


FIGURA 6.8: Colocado de células para microrrotación de CORDIC.

mulado aparecen una serie de pesos que se han de especificar como resultado de la experiencia de utilizar el algoritmo. Nuestra experiencia en el uso del algorimo de alineamiento por enfriamiento simulado para estas aplicaciones indica que las inecuaciones a cumplir son:

- para el peso del área total de solapamiento; $4 < \alpha_{ov} \leq 6$,
- para el peso de la longitud total de interconexión estimada; $6 < \alpha_{wl} \leq 10$,
- $\alpha_a \leq 2$; para el peso de área total de la célula,
- para el peso de la simetría del circuito; $\alpha_{sy} \leq 2$,
- para el peso del dislocamiento de los dispositivos del circuito, cuando su utilización sea el emparejamiento de transistores para formar puertas; $6 < \alpha_{ma} \leq 10$, siendo generalmente $\alpha_{ma} \leq 2$.
- el peso de la medida de la degradación de las prestaciones del circuito es $\alpha_{cd} \leq 2$, si no se dispone de un buen estimador de capacidades.

Obtenido el colocado de los dispositivos que conforman la célula, lo que procede es dividir el espacio de diseño en una rejilla de paso constante, con el fin de transformar la representación inicial en una estructura matricial. El objetivo, es manejar los dispositivos de forma agrupada en filas y columnas, con objeto de definir las filas de lógica y el área de los canales de alimentación al estilo de un compilador de estructuras. La expansión de los dispositivos en la fase de planificación y colocado de la célula, para acomodar el conexionado, define las áreas para conexionado de señales.

6.2.4 Procedimiento de trazado de conexiones: señales y alimentaciones.

Mediante la estructura matricial del espacio de diseño se simplifica la especificación de los anchos de canal vertical y horizontal para el conexionado de las alimentaciones. En la figura 6.9 se muestra el resultado de esta fase, cuando a la microrrotación se le aplica una definición de canales al estilo de buses laterales. La propia definición del área para la alimentación magnifica el área existente para tránsito de señales, orientando la definición de canales para ruteado de señales. El ruteado global de la alimentación subdivide el espacio donde transitan las líneas de alimentación en tiras horizontales con nodos de conexión en los extremos.

Para el ruteado global de señales, se prefiere utilizar un ruteador que minimize la longitud de los nodos y el número de cambios de máscaras de metal, y que incluya métodos de programación lineal para encontrar la solución más óptima. En la distribución OCTTOOLS-5.1, existe un ruteador global para tecnología CMOS, denominado **mercury**, que hemos adaptado para el trazado de señales en GaAs como hemos indicado en la sección 5.5 del capítulo anterior. Las soluciones obtenidas mediante el ruteador global para células estándares de la herramienta TimberWolf, de la misma distribución, no es óptima para los requerimientos especificados, puesto que los nodos tienen longitudes excesivas y no se minimiza el cambio de metales de ruteado, además el trazado global llevado a cabo por TimberWolf está orientado a canales entre células estándares.

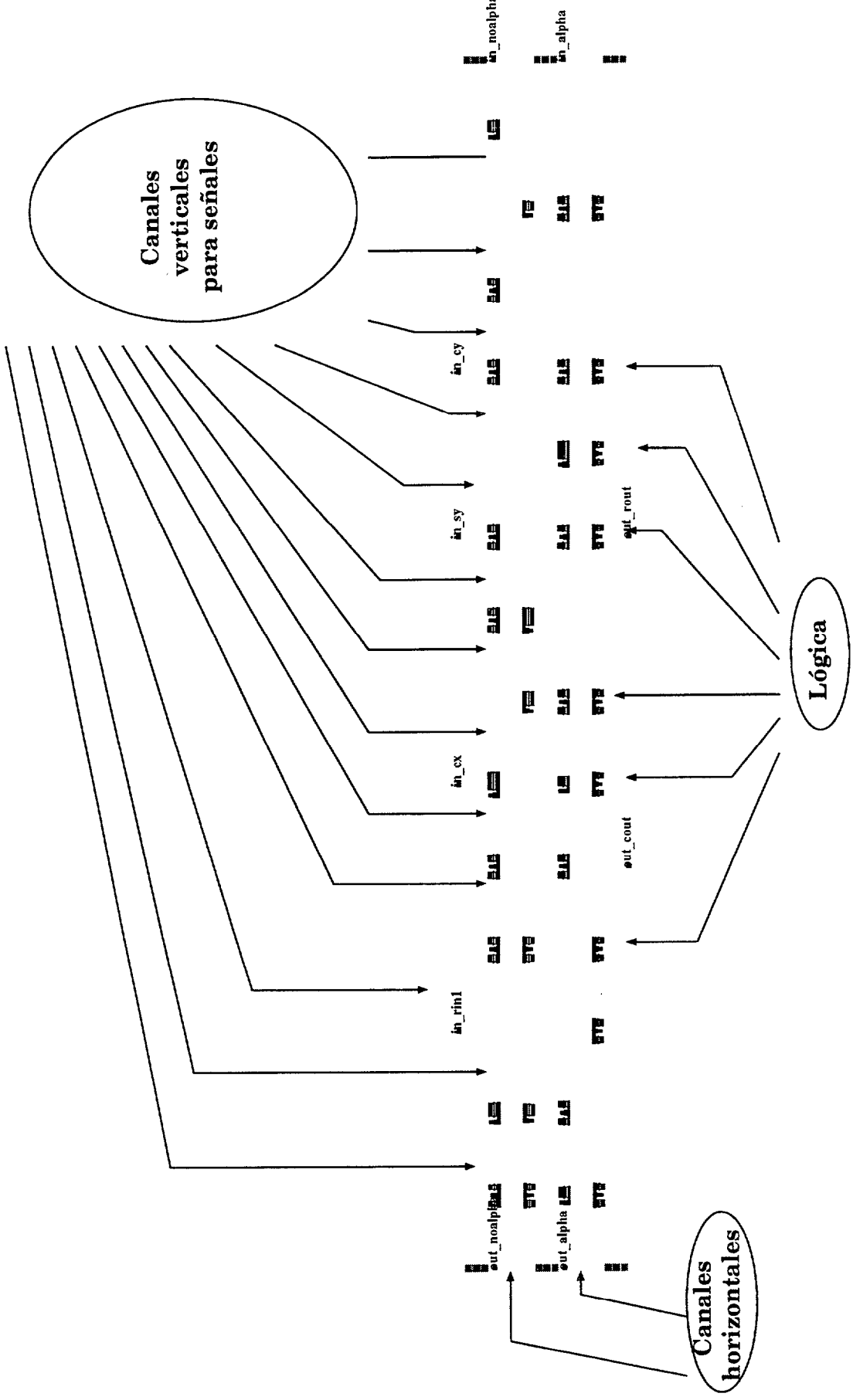


FIGURA 6.9: Definición de canales para señales y alimentaciones.

De entre las propuestas de herramientas para trazado detallado de señales (YACR, mighty), si se utiliza el ruteador simbólico **mighty** se debe calcular inicialmente el espacio necesario. En la figura 6.10 se muestra el trazado de señales (optando por el ruteador simbólico detallado mighty) de la célula ejemplo.

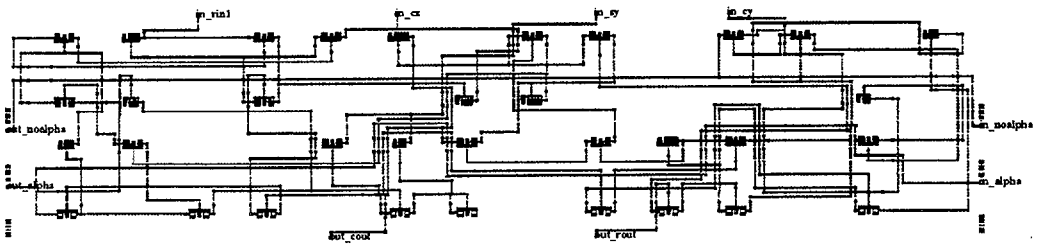


FIGURA 6.10: Trazado de señales.

Una vez trazadas las señales, se procede al trazado de las alimentaciones, obteniéndose el resultado que se muestra en la figura 6.11. Obsérvese como la definición de los canales (bajo una estrategia de buses laterales) define zonas de máxima densidad de señal, distintas a las dedicadas a alimentación.

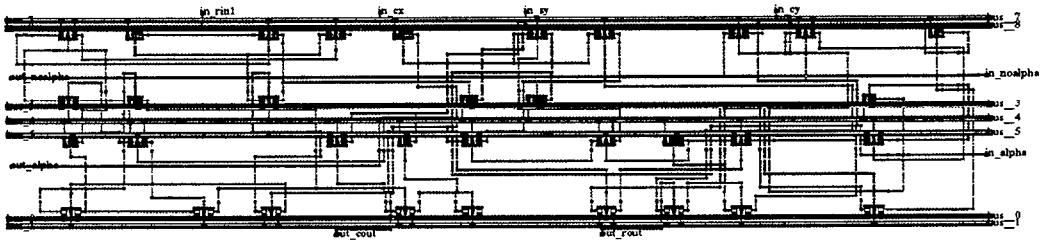


FIGURA 6.11: Trazado de alimentaciones.

6.2.5 Minimización de vías y compactación.

En la figura 6.12 se obtiene el resultado de compactar el ejemplo hasta ahora desarrollado. Previamente se han minimizado vías, reduciendo el número de cambios de metales.

Para simular el trazado teniendo en cuenta las capacidades introducidas por el cableado, cambiamos el diseño de formato, de OCT a Cadence. Sobre Cadence, disponemos de diversos extractores de dispositivos y parásitos, como indicamos en la sección 4.1.1.1. Una vez extraído el diseño procedemos a la simulación mediante HSPICE, cuyos resultados se muestran en la figura 6.13. Sobre esta base, podemos realizar un segundo dimensionamiento. Dado que se dispone de una versión del trazado geométrico en forma simbólica, podemos dimensionar las puertas y de nuevo compactar, minimizar vías, extraer y simular.

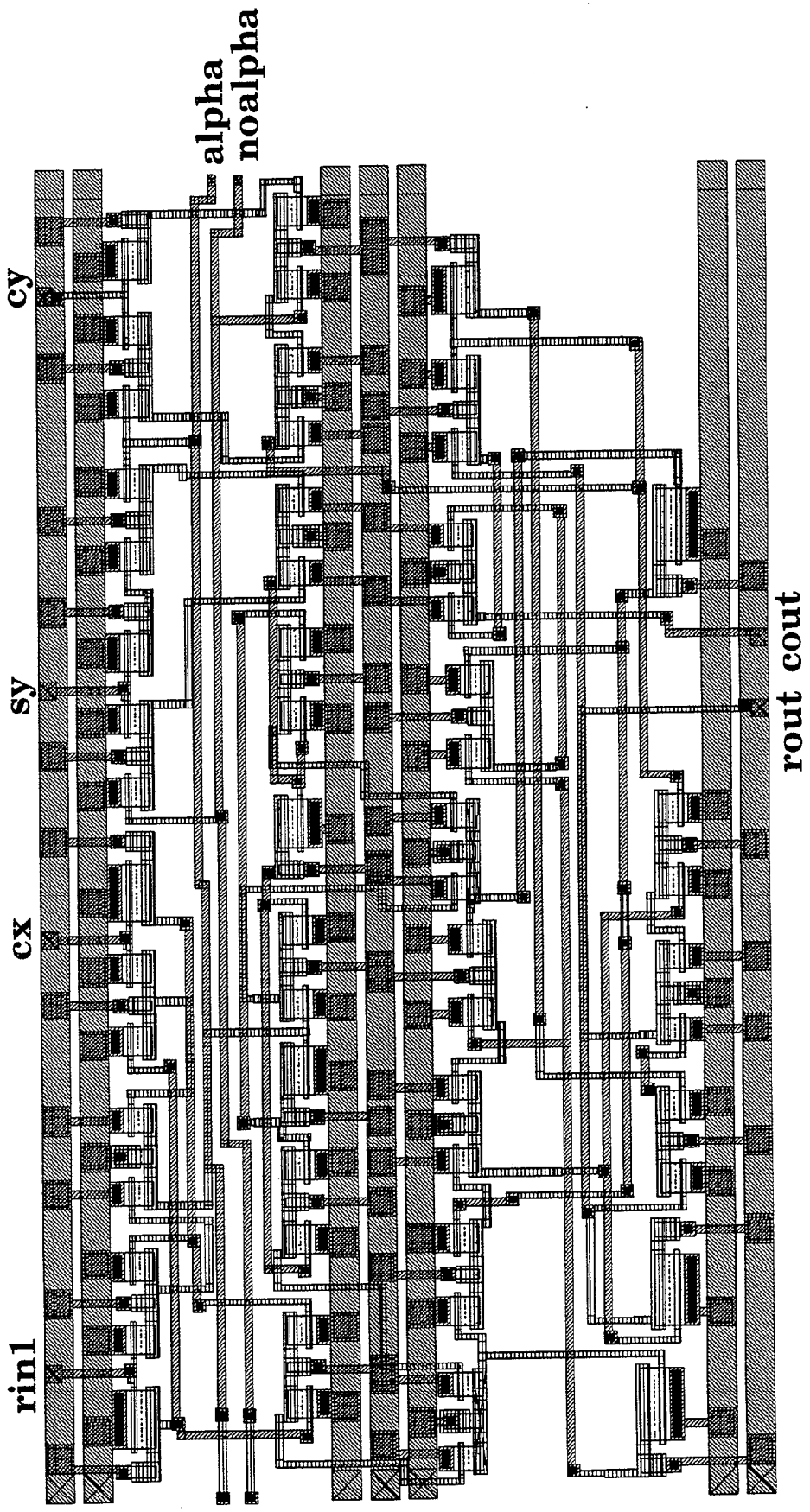


FIGURA 6.12: Sumador/restador 4–2 realizado por **OLYMPO**.

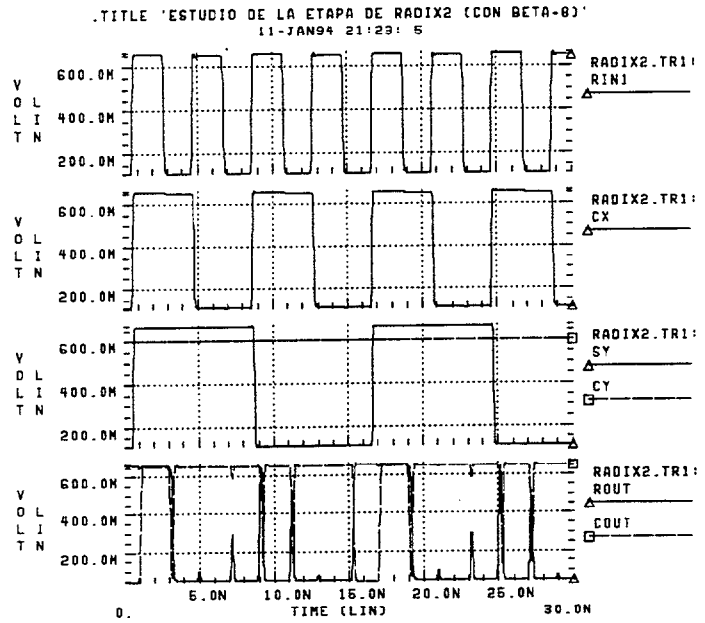


FIGURA 6.13: Resultados de la simulación con HSPICE teniendo en cuenta las capacidades introducidas por las conexiones.

6.3 Procedimiento de diseño manual.

El diseño manual de la microrrotación en base 2, lo realizamos sobre el entorno MAGIC. Todos los terminales de entrada y salida los hemos ajustado a una rejilla de $5\mu m$, con el fin de facilitar el ruteado de señales entre microrrotaciones. Todas las conexiones horizontales dentro de la célula se rutean en metal de nivel 1 y 2, excepto las que son cortas que se hacen en metal de puerta. La alimentación local se suministra en metal de nivel 2, mientras la global se hace en metal de nivel 3. De esta forma las tiras de tensión V_{DD} y GND mantienen las dimensiones mínimas, generando trazados que son eficientes en área. El resultado se muestra en la figura 6.14, estimándose el tiempo de diseño de aproximadamente dos semanas de trabajo de un diseñador experto, con una jornada de ocho horas, haciendo un total de aproximadamente unas 80 horas. El área ocupada por el trazado geométrico es de $230\mu m$ de largo por $90\mu m$ de alto, lo que supone un área de $0.0207mm^2$. La frecuencia de funcionamiento es de $1.33GHz$, siendo el retardo de la ruta crítica de $750ps$.

6.4 Generación de la microrrotación.

Una microrrotación de un CORDIC se compone de n células del tipo generado (n depende del número de bit que constituye el dato). Puesto que todas las células tienen alineados los terminales de la derecha e izquierda, para componer la microrrotación basta con realizar un simple adosado. En la figura 6.15 se muestra el código escrito

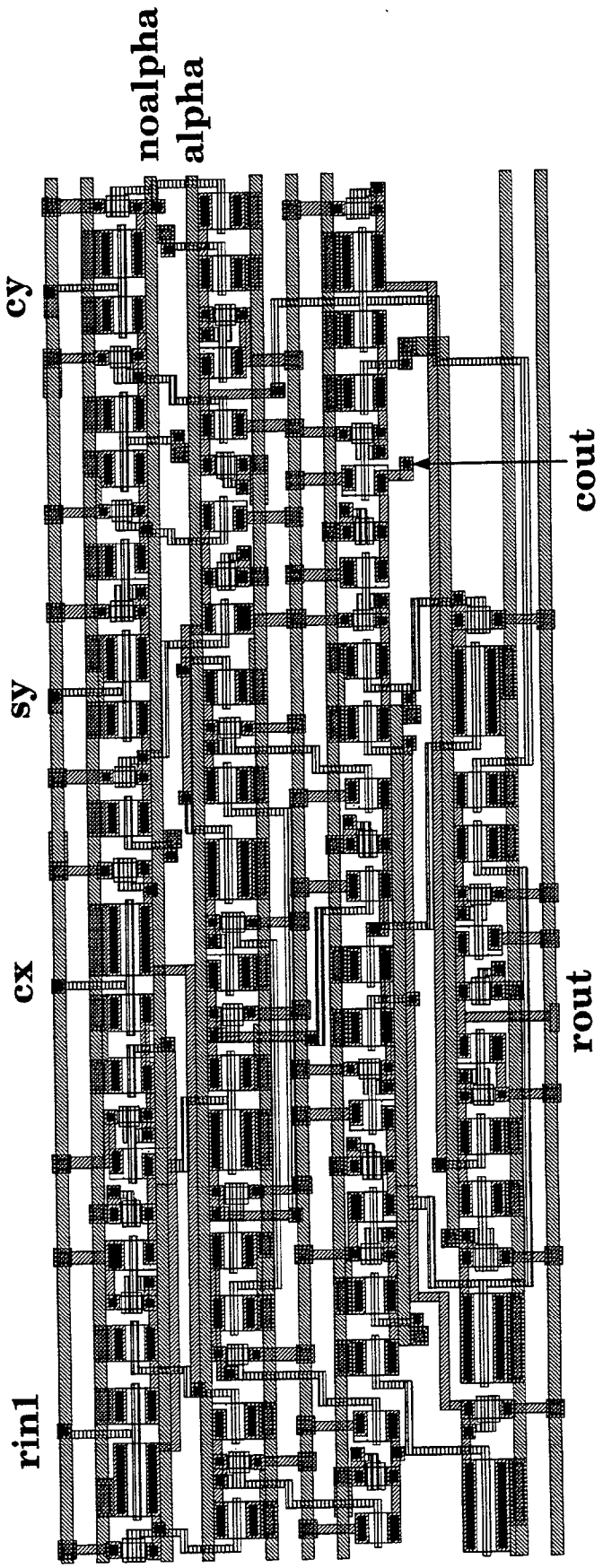


FIGURA 6.14: Sumador/restador 4–2 realizado a mano.

en formato BDNET para generar una microrrotación de datos de 3 bit. En la figura 6.16 se muestra el resultado de procesar esta especificación mediante **bdnet**.

```
model urot:place;

TECHNOLOGY H_GaAsII;
VIEWTYPE SYMBOLIC;
EDITSTYLE SYMBOLIC;

INPUT  rin1<2:0>;
INPUT  cx<2:0>;
INPUT  sy<2:0>;
INPUT  cy<2:0>;
OUTPUT rout<2:0>;
OUTPUT cout<2:0>;

ARRAY %i FROM 0 TO 2 OF
        INSTANCE radix2MABL:spaced PROMOTE
                [(27660*%i), 0]
                rin1: rin1<%i>;
                cout: cout<%i>;
                cx:   cx<%i>;
                sy:   sy<%i>;
                cy:   cy<%i>;
                rout: rout<%i>;
ENDMODEL;
```

FIGURA 6.15: Especificación para la generación de una microrrotación.

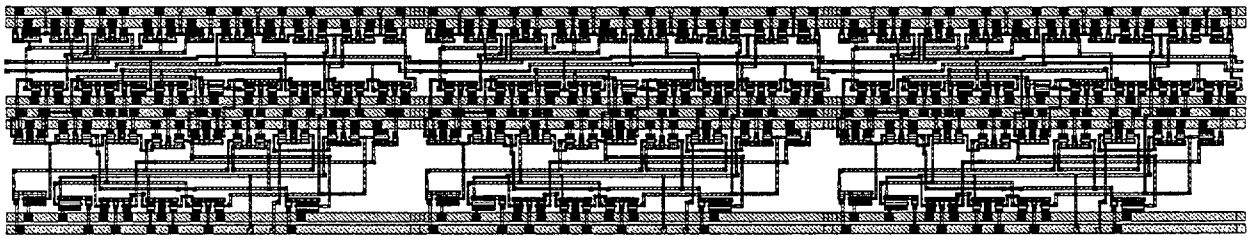


FIGURA 6.16: Microrrotación de dato de 3 bit de un procesador CORDIC.

6.5 Resultados y discusión comparativa.

Si se parte del colocado de las puertas del trazado realizado a mano, ejecutando las tareas de ruteado de señales y alimentación, minimización de vías y espaciado los resultados son

incremento en la longitud de un 10% respecto a la célula realizada a mano, resultando una longitud de $254\mu m$,

incremento en el alto de un 19%, resultando una altura de $107\mu m$,

incremento en el área de un 35%, lo que supone un área ocupada de $0.0270mm^2$.

De los resultados expuestos, se deduce que la estrategia de ruteado es adecuada y responde a la metodología de partida. No hay que perder de vista que el hecho de utilizar trazado simbólico supone perder de un 10 a un 20% en densidad, comparado con una reducción del 50 al 75% del tiempo de trazado. De entre todas estas cifras, la realmente significativa es la drástica reducción en el tiempo de diseño. La ejecución de las diferentes fases desde ruteado hasta compactación no sobrepasa el minuto de tiempo de *CPU* sobre una plataforma **SPARC** 10 de **Sun MicroSystem Corp.**

Si se parte de la lista de puertas que constituye el sumador/restador (figura 6.5), el tiempo de ejecución del algoritmo de alineamiento por enfriamiento simulado lleva aproximadamente $1min40s$ de tiempo de *CPU* sobre una plataforma **SPARC** 10 de **Sun MicroSystem Corp.** Este tiempo adicional, hace que el proceso completo (colocado, ruteado y compactación) no sobrepase los tres minutos de tiempo de *CPU*. El aumentando de área respecto al trazado realizado a mano es de un 45%. En la tabla 6.1 se compara esta última realización con los resultados obtenidos en la implementación manual.

Tipo de implementación	Retardo (ps)	Area (mm^2)	Potencia (mW)	Densidad ($\frac{trtos.}{mm^2}$)	Tiempo de diseño
Manual	750	0.0207	8.5	4396	80h
OLYMPO	860	0.0303	8.5	3003	3min

TABLA 6.1: Prestaciones finales.

Con este proceso de evaluación concluimos las contribuciones a la implementación de la metodología propuesta, resaltando que en una primera aproximación las prestaciones del entorno **OLYMPO** son competitivas con otras metodologías de trazado. La drástica reducción del tiempo de diseño, a costa de unas pérdidas muy moderadas en prestaciones (comparado con la metodología de células estándares), resalta la capacidad de **OLYMPO**.

Capítulo 7

Conclusiones y líneas futuras.

7.1 Conclusiones.

Hemos diseñado algunas estructuras básicas de computación utilizando una metodología de diseño *full–custom* denominada **Notación en Anillo**. Los resultados muestran que el estilo de trazado en el diseño en GaAs tiene una fuerte influencia en las prestaciones del circuito. El estilo de diseño propuesto, permite al diseñador optimizar el trazado de acuerdo con las prestaciones en velocidad, área y tiempo, obteniendo una disposición más regular en el espacio y por tanto más fácil de automatizar. Incluso esta metodología permite reducir las inductancias series de la líneas de alimentación y los acoplamientos con las señales, mejorando los retardos.

Según la metodología de **Notación en Anillo**, la distribución de alimentación la hemos dividido en dos niveles diferenciados:

- un primer nivel (global) a nivel de chip o módulo donde se concentran las mayores caídas de tensión, así como las mayores inductancias serie y
- un segundo nivel (local y transistor) donde se distingue la alimentación local, propiamente dicha, a lo largo de cada célula que forma parte del chip o módulo y la alimentación a nivel del transistor. Respecto a la alimentación local de la célula, la disposición de los buses permite una reducción de las inductancias series.

Para cada uno de estos niveles de alimentación hemos realizado un estudio de dimensionamiento. En general, si la tecnología permite hasta tres niveles de metalización, las dimensiones de la alimentación a nivel local son las mínimas según reglas de diseño. El ancho de la alimentación a nivel global está relacionado con las dimensiones del circuito. Si la tecnología es de dos niveles de metalización, los anchos de la alimentación dependen de la longitud de la célula, y de la intensidad que demandan las filas de lógica a las cuales está conectada.

Hemos desarrollado en Cadence/Edge un entorno para trazado físico *full–custom* en tecnología GaAs. En este entorno podemos realizar:

- la captura del diseño a nivel de esquemático y físico (trazado geométrico y simbólico),
- el análisis del circuito,
- la verificación del trazado y
- la simulación eléctrica mediante HSPICE.

En Cadence/Edge, a pesar de los beneficios derivados del trazado simbólico y de la creación parametrizada de dispositivos, las ventajas alcanzadas se pierden en el momento de la compactación. El proceso de compactación/espaciamiento se extiende innecesariamente hacia el interior del dispositivo. Por otra parte, la orientación hacia tecnología CMOS complica la especificación de reglas de formación de los dispositivos. Desde el punto de vista de la programabilidad de las herramientas en Cadence/Edge y Mentor, las principales desventajas son:

- Las derivadas de disponer de lenguajes no estándares y distintos para la descripción del trazado. En algunos casos estos lenguajes son interpretados, lo que suponen una incomodidad cuando se desea desarrollar un entorno de compilación.
- La orientación al estilo de células estándares de determinadas funciones de colocado y conexionado y la baja programabilidad de las mismas,
- la fuerte dependencia tecnológica del entorno hacia CMOS, y
- la no definición del sistema para el desarrollo de herramientas de compilación con una metodología distinta a la de células estándares.

La síntesis de células siguiendo la metodología de Notación en Anillo se concreta en el entorno **OLYMPO**, desde la planificación hasta la obtención del diseño *full–custom*. **OLYMPO** es un conjunto completo de herramientas para planificación, colocado y ruteado de células orientado a tecnologías de altas prestaciones como GaAs y con especial énfasis en el desarrollo de entornos de compilación. Este conjunto de herramientas están escritas en lenguaje C, desarrolladas en el sistema operativo UNIX, sobre la base de la distribución **OCTTOOLS 5.1** de Berkeley.

Para validar el entorno **OLYMPO** hemos realizado un generador de microrrotaciones que forman parte de la sección de procesamiento de un procesador para la evaluación de la CFFT (*Complex Fast Fourier Transform*) en base 2 y precisión de 16 bits, basado en el algoritmo CORDIC (*COordinate Rotation DIgital Computer*). Como elemento de referencia y comparación hemos implementado una microrrotación *radix* 2 al estilo *full–custom* totalmente a mano. Hemos simulado y medido ambos trazados físicos. El trazado generado por **OLYMPO** ocupa un 45% más de área, con un retardo de la ruta crítica superior en un 15%. Disminuyendo el tiempo de diseño de $80h$ de trabajo de un diseñador experto a $3min$ de CPU.

7.2 Líneas futuras.

El trabajo de investigación presentado en esta memoria puede continuarse en las siguientes líneas:

7.2.1 Compilación y generación de módulos.

Parte del conjunto de desarrollos mostrados en el presente trabajo, son extensibles fácilmente a la compilación de módulos. Un módulo puede ser definido como un elemento de trazado estructurado y de mayor jerarquía. Desde este punto de vista, puede construirse de la misma forma que cualquier célula. Inicialmente se han de generar los elementos de trazado que forman parte del módulo. Posteriormente, conocida a priori la posición relativa de las células, se procede al colocado de las células y ruteado tanto de señales como de la alimentación entre células.

Otro modo de proceder, es el que tiene en cuenta el adosado y alineamiento de los terminales de las células que configuran el módulo. Esta segunda forma de proceder, precisaría de información acerca de la célula crítica en dimensión, para cada una de las filas y columnas que configuran el módulo. Este conocimiento es necesario a fin de realizar la compactación de todas las células no críticas, con alineamiento de los terminales. Con este segundo enfoque se elimina la fase de ruteado.

7.2.2 Trazado de conexiones en tres niveles de metal.

En la metodología de Notación en Anillo cabe la posibilidad de utilizar metal de nivel 2, para la transmisión de aquellas señales de alto *fan–out* (señales de reloj, señales de entrada). A fin de poder trazar las señales que forman parte de las células en tres niveles de metal, hemos de desarrollar un ruteador de tres capas. Esta línea está soportada por los recursos materiales y humanos de la red Europea GARDEN de la que nuestro grupo forma parte.

7.2.3 Proceso contínuo de planificación y colocado guiado por el análisis temporal.

Las fases de planificación y colocado carecen de un proceso continuo e iterativo, controlado por una herramienta de más alto nivel, que incluya el análisis temporal.

Después de cada iteración, el planificador generaría como salida la posición y orientación de cada célula y también la forma y posición de los terminales para las células que no fueron especificadas completamente. Esta información se pasa al generador de células. Si el generador de células es incapaz de satisfacer todas las restricciones impuestas sobre una célula particular, la forma actual de cada célula generada se realimenta a la herramienta de planificado y colocado y se inicia una

nueva iteración. El módulo de planificación y colocado debe interaccionar con un verificador temporal. El propósito del analizador temporal es criticar la planificación. La información temporal se utiliza para evaluar las restricciones impuestas por la herramienta de planificación y colocado en las longitudes máximas y mínimas de ciertos nodos.

Esta línea está soportada por los recursos de dos redes Europeas (GARDEN y GRASS) de la que nuestro grupo forma parte.

Bibliografía

[1] Applicon. *IAGL User's Guide.* Applicon Incorporated, Jun. 1983.

[2] F. Argüello. *Application Specific Array Processor for Fast Orthogonal Transforms.* Tesis Doctoral, Univ. de Santiago de Compostela, Ene. 1992.

[3] V. Armas. GaAs HEMT double modulus frecuency divider. Infome técnico, Centro de Microelectrónica Aplicada, Mar. 1994.

[4] V. Armas, J.F. López, J.A. Montiel, y R. Sarmiento. Unidad Aritmético Lógica de la Arquitectura SPARC. En *Actas del XIX Congreso de Diseño de Circuitos Integrados*, (aceptado), 1994.

[5] J. Batali y A. Hartheimer. The Design Procedure Language Manual. Ai memo 598, Massachusetts Institute of Technology, 1980.

[6] T. Bautista, V. De Armas, R. Sarmiento, J.A. Montiel, F. Argüello, J.D. Bruguera, y E.L. Zapata. Procesador FFT, parte II: sección de procesamiento basada en CORDIC. *Actas del VIII Congreso de Diseño de Circuitos Integrados*, 1993.

[7] T. Blank. A Survey of Hardware Accelerators Used in Computer-Aided Design. *IEEE Design and Test*, 1(3):21–39, Ago. 1984.

[8] D. Bostick, G. Hachtel, R. Jacoby, M. Lightner, P. Moceyunas, C. Morrison, and D. Ravenscrof. The Boulder Optimal Logic Design System. En *Proceedings of the International Conference on Computer Aided Design*, pp. 62–65, Nov. 1987.

[9] R. Brayton, R. Rudell, A. Sangiovanni-Vicentelli, y A. Wang. MIS: A Multiple–level Logic Optimization System. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, pp. 1062–1081, Nov. 1987.

[10] M.A. Breuer. A Class of Min–Cut Placement Algorithms. En *Proc. 14Th Design Automation Conference*, pp. 284–290, Jun. 1977.

[11] M.R. Burich. *Design of Module Generators and Silicon Compilers*, capítulo 2, pp. 49–94. Kluwer Academic, 1992.

[12] J.L. Burns y A.R. Newton. Sparcs: A new constraint–based ic symbolic layout spacer. En *Proc. Custom Integrated Circuit Conference*, pp. 24–27, May. 1986.

[13] M. Burstein, S.J. Hong, y R. Pelavin. Hierarchical VLSI Layout: Simultaneous Placement and Wiring of Gate Arrays. En Anceau y Aas, *VLSI'83*, pp. 45–60, Amsterdam, Ago. 1983. North Holland.

[14] CAE. *CAE 2000 Command Language User's Manual.* CAE Corporation, Ago. 1984.

[15] Calma. *GPL II Programmers Reference Manual.* GE Calma Company, Feb. 1981.

[16] P. Carballo, J.A. Montiel, R. Sarmiento, y A. Núñez. Setting Up a Full–Custom Design Environment on Cadence For GaAs Technology. En IEEE-MMI, *Proceedings of the European Gallium Arsenide and Related III–V Compounds Applications Symposium*, 1994.

[17] P.P Carballo, J.A. Montiel, R. Sarmiento, J. López, y A. Núñez. Creación de un Entorno de Diseño Full–Custom en Cadence/Edge para Tecnología GaAs. *Actas del VII Congreso de Diseño de Circuitos Integrados*, pp. 333–338, 1992.

[18] A. Casotto. *puppy reference manual.* Dept. of Electrical Engineering, University of California, Berkeley, Sep. 1993.

[19] A. Casotto y A. Sangiovanni-Vincentelli. Placement of Standard Cells Using Simulated Annealing on the Connection Machine. En *The Proceedings of the International Conference on Computer-Aided Design.* IEEE, 1987.

[20] A. Cassoto. *Octtools 5.1 User Guide.* Electronic Research Laboratory, University of California, Berkeley, Oct. 1991.

[21] K.C. Chen y E.S. Kuh. Module Placement Based on Resistive Network Optimization. *IEEE Transaction on CAD*, 3(3):218–225, Jul. 1984.

[22] J. Cherry, H. Shrobe, N. Mayle, C. Baker, H. Minsky. K. Reti, y N. Weste. NS: An Integrated Symbolic Design System. En Horbst, *VLSI'85*, pp. 325–334, 1985.

[23] Computervision. *CADDS II/VLSI Integrated Circuit Programming Language User's Guide.* Computervision Corporation, Abr. 1986.

[24] J. Cong, B. Preas, y C.L. Liu. General models and algorithms for over–the–cell routing in standard cell design. En *Proc. 27th Design Automation Conference*, pp. 709–715, June 1990.

[25] J.W. Cooley y J.W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.*, 19(4):297–301, 1965.

[26] J. Darringer, D. Brand, J. Gerbi, W. Joyner, y L. Trevillyan. LSS: A System for Production Logic Synthesis. *IBM J. Res. Develop.*, C-18:537–545, Sep. 1984.

[27] A.R. Deas y I.M. Nixon. Chromatic Idioms for Automated VLSI Floorplanning. En Horbst, *VLSI'85*, pp. 61–70, Ago. 1985.

[28] D.N. Deutsch. A "Dogleg" Channel Router. En *Proc. 13Th Design Automation Conference*, pp. 425–433, Jun. 1976.

[29] M.T. Doreau y P. Koziol. TWIGY: A topological Algorithm Based Routing System. En *Proc. 18Th Design Automation Conference*, pp. 746–755, Jun. 1981.

[30] M.I. Elmasry. *Virtual Grid Symbolic Layout*, capítulo 2, pp. 263–271. IEEE PRESS, 1985.

[31] K. Eshraghian, A. Blanksby, R. Sarmiento, y C.C. Lim. GaAs Methodology & Perfomance Estimates for Very High Speed Circuits Using Normally–Off Classes of Logic. En IEEE-MMI, *Proceedings of the European Gallium Arsenide and Related III–V Compounds Applications Symposium*, pp. 203–206, 1994.

[32] K. Eshraghian, R. Sarmiento, P.P. Carballo, y A. Núñez. Speed–Area–Power Optimization for DCFL and SDCFL Class of Logic Using Ring Notation. *Microprocessing and Microprogramming*, 3(2):75–82, 1991.

[33] D. Gajski y R.H. Kuhn. New VLSI Tools. *Computer*, 16(12):11–14, Dic. 1983.

[34] D.D. Gajski y D.E. Thomas. *Introduction to Silicon Compilation*, capítulo 1, pp. 1–48. Kluwer Academic, 1992.

[35] M.R. Garey y D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP–Completeness.* W. H. Freeman, 1979.

[36] G. Ghione y C.U. Naldi. Analytical Formulas for Coplanar Lines in Hybrid and Monolithic MICs. *Electronics Letters*, 20:179–181, Feb. 1984.

[37] G. Ghione y C.U. Naldi. Coplanar Waveguides for MMIC Applications: Effect of Upper Shielding, Conductor Backing, Finite-Extent Ground Planes, and Line-to-Line Coupling. *IEEE Trans. Microwave Theory and Techniques*, MTT-35:260–267, Mar. 1987.

[38] L. Gomez. *Optimización multiobjetivo de circuitos digitales DCFL/SDCFL en GaAs.* Tesis Doctoral, ETS Ingenieros de Telecomunicación, Univ. de Las Palmas de G.C., Dic. 1992.

[39] D. Gregory, K. Barlett, A. DeGeus, y G. Hachtel. SOCRATES: A System for Automatically Synthesizing and Optimizing Combinational Logic. En *Proc. 23rd Design Automation Conference*, pp. 78–85, 1986.

[40] B. Hajek. A tutorial survey of theory and applications of simulated annealing. En *Proc. IEEE Conf. on Decision and control*, Dic. 1985.

[41] G.T. Hamachi y J.K. Ousterhout. A Switchbox Router with Obstacle Avoidance. En *Proc. 21St Design Automation Conference*, pp. 173–179, Jun. 1984.

[42] A. Hanczakowski. TRICKY symbolic layout system for integrated circuits. En *Proc. Spring Compcon 81*, pp. 374–376, 1981.

[43] D. Harrison, P. Moore, R.L. Spickelmier, y A.R. Newton. Data management and graphics editing in the berkeley design environment. En *Proc. ICCAD*, pp. 24–27, Jun. 1986.

[44] W.R. Heller, G. Sorkin, y K. Maling. The Planar Package Planner for System Designers. En *Proc. 19Th Design Automation Conference*, pp. 153–260, Jun. 1982.

[45] A. Hernández. *Modelado del tiempo de propagación para análisis y verificación temporal de circuitos DCFL en GaAs.* Tesis Doctoral, ETS Ingenieros de Telecomunicacion, Univ. de Las Palmas de G.C., Dic. 1992.

[46] D.W. Hightower. A Solution to Line–Routing Problems in the Continuous Plane. En *Proc. 6Th Design Automation Workshop*, pp. 1–24, Jun. 1969.

[47] W. Hilberg. From Approximation to Exact Relations for Characteristic Impedances. *IEEE Trans. Microwave Theory and Technique*, MTT-17:259–265, May 1969.

[48] N.D. Holmes, N.A. Sherwani, y M. Sarrafzadeh. Algorithms for three–layer over–the–cell channel routing. En *Proc. IEEE Int. Conf. Computer–Aided Design*, pp. 428–431, Nov. 1991.

[49] J.Burns, A.Casotto, M.Igusa, F.Marron, F.Romeo, A.Sangiovanni-Vincentelli, C.Sechen, H.Shin, G.Srinath, y H.Yaghutiel. Mosaico: An Integrated Macro–Cell Layout System. En C.H Sequin, *Proceedings of VLSI '87*, Vancouver, Canada, Ago. 1987.

[50] D. Johannsen. Bristle Blocks: A Silicon Compiler. En *Proc. 16Th Design Automation Conference.*, pp. 310–313, 1979.

[51] K. Karplus. *CHISEL, an extension to the programming language C for VLSI layout.* Tesis Doctoral, Stanford University, 1982.

[52] A.I. Kayssi y K.A. Sakallah. Delay Macromodels for the Timing Analysis of GaAs DCFL. *IEEE Trans. oc Computer–Aided Design*, pp. 142–145, Jul. 1992.

[53] J. Keller. Power and Ground Requirements for a High Speed 32 Bit Computer Chip Set. Informe técnico, University of California at Berkeley, Ago. 1985.

[54] B.W. Kernighan, D.G. Schweikert, y G. Persky. An Optimun Channel–Routing Algorithm for Polycel Layout of Integrated Circuits. En *Proc. 10Th Design Automation Workshop*, pp. 50–59, Jun. 1973.

[55] S. Kirkpatrick, C.D.Jr. Gelatt, y M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, May. 1983.

[56] P.W. Kollaritsch y N.H.E. Weste. TOPOLOGIZER: An Expert System Translator of Transistor Connectivity to Symbolic Cell Layout. *IEEE Journal of Solid–State Circuits*, 1986.

[57] U. Lauther. Channel Routing in a General Cell Environment. En Horbst, *VLSI'85*, pp. 393–403, Ago. 1985.

[58] H-F.S. Law y J.D. Mosby. An Intelligent Composition Tool for Regular and Semi–Regular VLSI Structures. En *Proc. ICCAD–85*, pp. 169–172, 1985.

[59] M.S. Lin, H.W. Perng, C.Y. Hwang, y Y.L. Lin. Channel density reduction by riuting over the cells. En *Proc. 28Th Design Automation Conference*, pp. 120–125, 1991.

[60] Y-L.S. Lin y D.D. Gajski. LES: A Layout Expert System. En *Proc. 24Th Design Automation Conference*, pp. 672–678, Jun. 1987.

[61] R.J. Lipton, S.C. North, R. Sedgewick, J. Valdes, y G. Vijayan. ALI: A Procedural Language to Describe VLSI Layouts. En *Proc. 19Th Design Automation Conference*, pp. 467–473, 1982.

[62] S.I. Long y S.E. Butner. *Gallium Arsenide Digital Integrated Circuit Design.* McGraw–Hill Publishing Company, 1990.

[63] A. Lopez y H. Law. A Dense Gate Matrix Layout Method for MOS VLSI. *IEEE Trans. on Electronics Devices*, ED–27(8):1671–1675, Ago. 1980.

[64] J.F. López. GaAs ROM Memories. Infome técnico, Thomson Composants Microondes, Jul. 1992.

[65] J.F. Lopez, J.A. Montiel, y V. Armas. VLSI GaAs Experience Using H–GaAsII Technology. *Eurochip Workshop on VLSI Design Training*, Oct. 1993.

[66] W.K. Luk. A Greedy Switch–box Router. VLSI Document VLSI Documnent V158, Carnegie–Mellon University, Department of Computer Science, May 1984.

[67] R.N. Mayo. Mocha Chip: A System for the Graphic Design of VLSI Module Generators. En *Proc. ICCAD–86*, pp. 74–77, 1986.

[68] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, y E. Teller. Equation of State Calculation by Fast Computing Machine. *Journal of Chemical Physics*, 21(6):1087–1092, Jun. 1953.

[69] M.R. Misha y T.G. Matheson. Silicon Compilation Environments. En *Proc. Custom Integrated Circuits Conference*, pp. 208–212, May. 1985.

[70] J.A. Montiel, V. De Armas, J.F. López, y T. Bautista. Tecnología H–GaAsII de Vitesse: Especificaciones y Reflexiones. *Actas del VIII Congreso Diseño de Circuitos Integrados*, pp. 68–74, Nov. 1993.

[71] E.F. Moore. Shortest Path Through a Maze. En Harvard University Press, *Infome técnicos*, pp. 285–292. Harvard University Press, Cambridge, Massachusetts, 1959.

[72] A.R. Newton. Symbolic Layout and Procedural Design. En DeMichelli, Sangiovanni-Vicentelli, y Antognetti, *Design System for VLSI Circuits: Logic Synthesis and Silicon Compilation*. Kluwer Academic Publishers, 1986.

[73] R.H. Otten. Automatic floorplan design. En *Proc. 19Th Design Automation Conference*, pp. 261–267, Jun. 1982.

[74] J.K. Oustherhout. The User Interface and Implementation of an IC Layout Editor. *IEEE Trans. on Computer-Aided Design*, pp. 242–249, Jul. 1984.

[75] J.K. Oustherhout, G.T. Hamachi, R.N. Mayo, W.S. Scott, y G.S. Taylor. Magic: A VLSI Layout System. En *ACM/IEEE 21st Design Automation Conf. Proc.*, pp. 152–159, 1984.

[76] J.K. Oustherhout, G.T. Hamachi, R.N. Mayo, W.S. Scott, y G.S. Taylor. The MAGIC VLSI Layout System. *IEEE Design and Test*, pp. 19–26, Feb. 1985.

[77] J. Reed. Yacr2: Yet Another Channel Router. Informe técnico, Univ. of California, Berkeley, Feb. 1985.

[78] R.L. Rivest. The PI (Placement and Interconnet) System. En *Proc. 19Th Design Automation Conference*, pp. 475–481, Jun. 1982.

[79] R.L. Rivest y C.M. Fiduccia. A Greedy Channel Router. En *Proc. 19Th Design Automation Conference*, pp. 418–424, Jun. 1982.

[80] J. Rosenberg y N. Weste. The ABCD Language. Infome técnico 82–01, Microelectronics Center of NC, Microelectronics Center of NC, 1982.

[81] J.B. Rosenberg. CAD Tools for Mask Generation. En Wolfgang Fichtner y Martin Morf, *Proceedings of the Summer School on VLSI CAD Tools and Applications*. Kluwer Academic Publishers, 1987.

[82] S.M. Rubin. *Computer Aids for VLSI Design*. Addison-Wesley, 1987.

[83] R. Sarmiento, V. Armas, P.P. Carballo, J. López, y A. Núñez. Diseño y Optimización de Sumadores en Tecnología GaAs. *Actas del VI Congreso Diseño de Circuitos Integrados*, pp. 32–38, 1991.

[84] R. Sarmiento, P.P. Carballo, y A. Núñez. High speed primitives of hardware accelerators for DSP in GaAs technology. *IEE PROCEEDINGS–G*, 139(2):205–216, Abr. 1992.

[85] R. Sarmiento y K. Eshraghian. Implementation of a CORDIC Processor for CFFT Computation in Gallium Arsenide Technology. En IEEE Press, *Proc. Of the Europen Desing Automation Conference*, pp. 238–244, Dic. 1994.

[86] R. Sarmiento, J.A. Montiel, P.P. Carballo, J.F. López, y A. Núñez. Diseño y Optimización de Multiplicadores para Procesado Digital de Señales en Tecnología GaAs. *Actas del VI Congreso de Diseño de Circuitos Integrados*, pp. 20–26, 1991.

[87] C. Sechen, K-W Lee, B. Swartz, D. Chen, y M. Lee. *The TimberWolfSC Standard Cell Placement and Global Routing Package.* Yale University, Yale University, Oct. 1987.

[88] C. Sechen y A. Sangiovanni-Vicentelli. The Timberwolf Placement and Routing Package. En *Proc. Custom Integrated Circuit Conference*, pp. 522–527, May. 1984.

[89] Carl Sechen. *Placement and Global Routing of Integrated Circuits Using Simulated Annealing.* Tesis Doctoral, University of California, Berkeley, 1987.

[90] H. Shin y A. Sangiovanni-Vicentelli. Mighty: A 'Rip–up and Reroute' Detailed Router. En *Proc. IEEE International Conference on ComputerAided Design*, pp. 2–5, Nov. 1986.

[91] H.P. Sing, R.A. Burrier, y R.A. Sadler. A 6.5-ns GaAs 20x20-b Parallel Multiplier with 67-ps Gate Delay. *IEEE Journal of Solid-State Circuits*, 25(5):1226–1231, Oct. 1990.

[92] W.S. Song y L.A. Glaser. Power Distribution Techniques for VLSI Circuits. *IEEE Journal Solid State Circuit*, SC–21:150–156, Feb. 1983.

[93] J. Soukup. Circuit Layout. *Proc. IEEE*, 69(10):1281–1304, Oct. 1981.

[94] K.J. Supowitz y E.A. Slutz. Placement Algorithms for Custom VLSI. En *Proc. 20Th Design Automation Conference*, pp. 164–170, Jun. 1983.

[95] Cadence Design Systems. *Physical Design. Reference Manual.* Cadence Design Systems, Inc., 1989.

[96] Cadence Design Systems. *DIVA Reference Manual.* Cadence Design Systems, Inc., 1991.

[97] SDA Systems. *SDA Languages. Reference Manual.* SDA System Incorporation, 1987.

[98] S. Teig, R.L. Smith, y J. Seaton. Timing–Driven Layout of Cell–Based ICs. *VLSI Systems Design*, 7(5):63–73, May 1986.

[99] M. Terai, K. Nakajima, K. Takahashi, y K. Sato. A New Approach to Over–the–Cell Channel Routing with Three Layers. *IEEE Transactions on Computer–Aided Design of Integrated Circuits and Systems*, 13(2):187–200, February 1994.

[100] M. Tompa. An Optimal Solution to a Wire–Routing Problem. En *Proc. 12Th Annual ACM Symposium of Theory of Computing*, pp. 161–176, 1980.

[101] S. Trimberger. Automatic chip layout. En M.I. Elmasry, *Digital VLSI Systems*, capítulo Design Methodologies, pp. 76–83. IEEE PRESS, Ene. 1985.

[102] T. Uehara y W.M. vanCleemput. Original Layout of CMOS Functional Arrays. *IEEE Trans. on Computers*, C–30(5):305–311, 1979.

[103] J.D. Ullman. *Computational Aspect of VLSI.* Computer Science Press, 1984.

[104] Vitesse. *Foundry Design Manual.* Vitesse Semiconductor Corporation, Mar. 1991.

[105] J.V. Volder. The CORDIC Trigonometric Computing Techniques. *IEEE Transactions on Electronic Computing*, EC–8(4):330–334, Sep. 1959.

[106] R.A. Walker y D.E. Thomas. A Model for Design Representation and Synthesis. En *Proc. 22Nd Design Automation Conference*, pp. 453–459, 1985.

[107] C.L. Wardle, C.R. Watson, C.A. Wilson, J.C. Mudge, y J.B. Nelson. A Declarative Design Approach for Combining Macrocells by Directed Placement and Constructive Routing. En *Proc. 21St Design Automation Conference*, pp. 594–601, Jun. 1984.

[108] A. Weinberger. Large Scale Integration of MOS Complex Logic: A Layout Method. *IEEE Journal of Solid–State Circuits*, SC–2(4):182–190, Dic. 1967.

[109] N. Weste. MULGA: An interactive symbolic layout system for the design of integrated circuit. *Bell System Technical Journal*, 60(6), 1981.

[110] N. Weste. Virtual Grid Symbolic Layout. En *ACM/IEEE 18th Design Automation Conf. Proc.*, pp. 225–233, 1981.

[111] N. Weste y K. Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley, 1985.

[112] J. Williams. *STICKS: A new approach to LSI design.* Tesis Doctoral, Massachusetts Institute of Technology, Jun. 1977.

[113] J. Williams. STICKS- a Graphical Compiler for High Level LSI Design. En *Proceedings of the 1978 NCC*, pp. 289–295, May. 1978.

[114] B. Wu, N.A. Sherwani, N.D. Holmes, y M. Sarrafzadeh. Over–the–cell routers for new cell model. En *Proc. 29Th Design Automation Conference*, pp. 604–607, Jun. 1992.

[115] E.L. Zapata y F. Argüello. Application–specific Architecture for Fast Transforms based on the Successive Doubling Method. *IEEE Trans. Signal Processing*, 41(3):1476–1481, 1993.