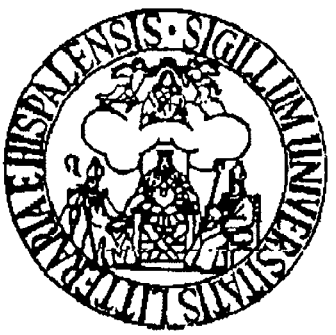


NUEVAS METODOLOGIAS HEURISTICAS APLICADAS AL DISEÑO DE CIRCUITOS INTEGRADOS

por

Miguel Angel Aguirre Echánove

Ingeniero Industrial por la Escuela Superior de Ingenieros Industriales

de la Universidad de Sevilla

Presentada en la

Escuela Superior de Ingenieros Industriales

de la

Universidad de Sevilla

para la obtención del

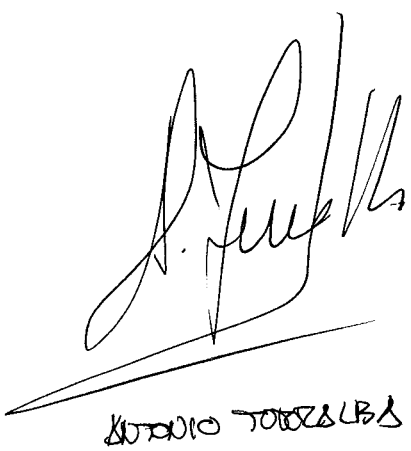
título de Doctor Ingeniero Industrial

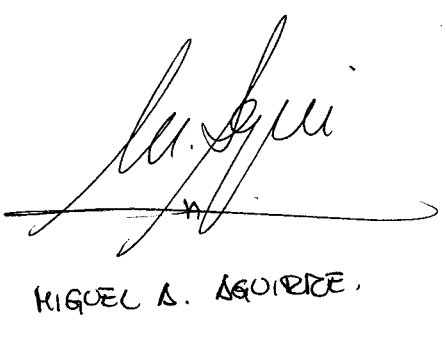
en Sevilla, Mayo de 1994

TESIS DOCTORAL

NUEVAS METODOLOGIAS HEURISTICAS APLICADAS AL DISEÑO DE CIRCUITOS INTEGRADOS

Autor: Miguel Angel Aguirre Echánove
Director: Antonio Jesús Torralba Silgado





Quiero dedicar la presente tesis a las personas que con su apoyo y comprensión más me han animado a concluirla.

En especial a mi amigo y director de tesis Antonio Jesús Torralba Silgado, por su paciencia y constante ánimo y a Leopoldo García Franquelo por sus buenos consejos.

A mis compañeros del Grupo de Tecnología Electrónica y del resto del Departamento de Ingeniería de Sistemas y Automática de la ESII.

A mis padres, hermanos y familiares que me han sabido aconsejar en los momentos más duros.

A mis buenos amigos compañeros de carrera, y en modo muy especial a Fernando Ruiz.

Por último expresar mi agradecimiento especial a D. Jorge Chávez Orzáez, ya que sin su cuidado de los sistemas hubiera sido imposible abordar este trabajo. A D. Juan Larrañeta Astola, catedrático del Dpto. de Organización Industrial de la ESII de Sevilla por sus indicaciones acerca del algoritmo *Búsqueda Tabú*.

NOTA DEL AUTOR

Dado que casi toda la literatura técnica que trata los temas expuestos en esta tesis se encuentra en lengua inglesa resulta tentador emplear la terminología inglesa, ampliamente aceptada en la comunidad científica, para designar los procesos y técnicas que aquí se describen. Así, podríamos decir que el problema del "placement" es una fase muy importante del problema más general del "layout" de un circuito VLSI, que suele ir acompañado de una fase de "routing" global. No obstante creemos que es nuestra responsabilidad defender el idioma de innecesarias intromisiones en el campo técnico, máxime cuando existen en lengua española términos equivalentes que pueden reemplazarlos. Así, diremos que el problema de la "colocación" de celdas es una fase muy importante del problema más general de la "disposición" de un circuito de alta escala de integración, que suele ir acompañado de una fase de "conexionado" global. Dada la escasez de bibliografía en lengua española, la traducción de los términos se ha hecho de una manera bastante libre. Para no confundir al lector, conocedor de la literatura técnica existente al respecto, cada uno de los términos propuestos se presenta en cursivas y se acompaña, al menos la primera vez que aparece en el texto, del término inglés correspondiente, encerrado entre paréntesis. En el apéndice V se relacionan los términos propuestos por el autor y su equivalente en lengua inglesa.

RESUMEN DE LA TESIS

La experiencia adquirida por el autor en el diseño de circuitos integrados de aplicación industrial, le mostró la necesidad de contar con herramientas eficientes de automatización del diseño de circuitos electrónicos. El objeto de esta tesis es el desarrollo de nuevas técnicas heurísticas que permitan obtener soluciones óptimas en aquellos problemas de gran complejidad que aparecen en el diseño automático de circuitos integrados. Las técnicas propuestas se aplican al problema de la *colocación* (*placement*) de celdas estándar en circuitos de muy alta escala de integración.

De particular interés resulta la aplicación de estas técnicas de optimización al diseño de circuitos integrados analógicos y mixtos, dada la relativa inmadurez de los paquetes de diseño existentes, sobre todo cuando se comparan con el nivel de automatización alcanzado en el campo digital. Por ello, conocidas la propiedades de los algoritmos de optimización, se han llevado a cabo nuevas aproximaciones al espacio continuo con el objetivo de aplicar estos algoritmos al problema de la síntesis de celdas analógicas. En este caso, el objetivo es obtener un circuito que cumpla un conjunto de especificaciones definidas por el diseñador, con una ocupación mínima de área de silicio.

La presente tesis se compone de las siguientes partes:

1. En el primer capítulo se presenta de un modo general el problema de diseño de un circuito electrónico y se definen los elementos básicos del problema de la *disposición* de los componentes de un circuito integrado.

2. Posteriormente se dedica un capítulo al estudio de los algoritmos de partición de circuitos, pues éstos constituyen la base de una buena parte de los algoritmos existentes de *colocación* de celdas. En este capítulo se propone un nuevo algoritmo de partición mediante agrupación de celdas, que proporciona buenos resultados, cuando se comparan con los obtenidos por los algoritmos clásicos.

3. En el siguiente capítulo se entra de lleno en el problema de la *colocación* de celdas y se describen los algoritmos existentes en la literatura. En particular, se hace un estudio detallado del algoritmo denominado *enfriamiento simulado*

(*simulated annealing*).

4. En el capítulo cuarto se propone una nueva técnica de *colocación* de celdas que combina las ventajas de los métodos de particionamiento y de enfriamiento simulado. El método propuesto se codifica en un conjunto de rutinas que se integran en las herramientas de diseño de la Universidad de Berkeley, conocidas como *Octtools*. Los resultados se comparan ventajosamente con los obtenidos por *TimberWolfSC*, considerado por la comunidad científica como un patrón de referencia.

5. En el quinto capítulo se expone el problema de la síntesis de celdas analógicas y se plantea como un problema de optimización paramétrica con restricciones.

6. En el capítulo seis se propone una aplicación de los algoritmos de búsqueda tabú y de enfriamiento simulado al espacio continuo. Se aprovechan las ventajas de esta aproximación para desarrollar un paquete de rutinas de síntesis de celdas analógicas. En particular, los resultados obtenidos con el algoritmo de búsqueda tabú, permiten alcanzar una solución aceptable en un número muy reducido de iteraciones, permitiendo la utilización de un simulador de circuitos tipo SPICE, para evaluar las prestaciones del circuito en cada iteración.

7. Finalmente, se enumeran las conclusiones más importantes de esta tesis, destacando las aportaciones originales en el campo de la automatización del diseño de circuitos integrados, y se plantean nuevas líneas de investigación.

Aparte de la labor de síntesis y sistematización de la bibliografía, las aportaciones originales del autor se encuentran en la sección 5 del capítulo 2, y en los capítulos 4 y 6.

Palabras clave: **Particionamiento de Circuitos, Colocación de celdas, Enfriamiento Simulado, Búsqueda Tabú, Mínimo Global, Optimización Combinatoria, Síntesis Analógica.**

Indice

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1 Diseño de Circuitos Integrados

Este primer apartado pretende situar el problema de la *colocación* de celdas en el contexto del diseño de un circuito integrado. En un capítulo posterior se realizará una descripción formal del problema.

La figura 1-1 muestra el procedimiento de diseño típico de un circuito integrado [123], [94], [36]. El diseñador parte de unas especificaciones y genera en una primera fase el diagrama funcional de un circuito que cumpla con dichas especificaciones. Esta descripción comienza por ser muy general y va ganando en detalle a medida que el diseño progresa. La introducción de lenguajes de definición hardware, tales como los lenguajes de transferencia de registros (RTLs) y, recientemente VHDL, proporcionan un entorno adecuado para realizar estas primeras fases del diseño.

En diseños de gran complejidad es aconsejable (y a veces necesario) realizar una primera simulación del comportamiento del sistema a partir de una definición de alto nivel de cada uno de sus módulos. De esta manera se puede identificar, desde el primer nivel de diseño, la existencia de caminos críticos que condicionan todo el proceso. A medida que el diseño progresa, los módulos que componen el circuito van ganando en detalle hasta llegar a su definición a nivel de bloques funcionales básicos. El empleo de herramientas de diseño automático permite obtener, de una

menera automática, esta definición de bajo nivel a partir de la descripción en alto nivel de cada uno de los bloques. A lo largo de este proceso descendente, se realizan continuas simulaciones y rediseños, hasta que se obtiene el circuito final.

Consideraciones de testabilidad del circuito deben formar parte del propio proceso de diseño y es aconsejable que la testabilidad del circuito sea considerada desde las primeras fases del mismo.

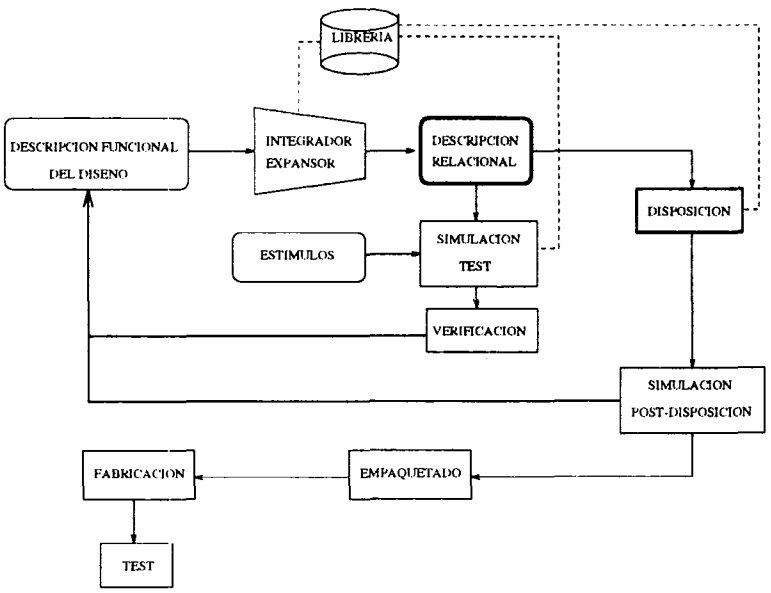


Figura 1-1: Esquema general del proceso de diseño

Ya sea de una forma manual o automática, al final llegamos a una descripción del circuito en forma de bloques funcionales básicos. La materialización del circuito sobre un bloque de silicio constituye el denominado problema de la *disposición* (*layout*).

1.2 El Problema de la *Disposición*

La *disposición* de un circuito integrado consiste en la colocación de los bloques que constituyen el diseño en un espacio bidimensional finito y en la interconexión de los mismos de acuerdo con el esquema del circuito. El objetivo de

este proceso es el de completar la *colocación* e *interconexión* del diseño en el menor área posible. No todas las disposiciones imaginables son posibles; las disposiciones posibles deben satisfacer un conjunto de restricciones de diseño (derivadas de la posición y tamaño de las celdas empleadas), restricciones tecnológicas (derivadas de un conjunto de reglas de diseño propias de cada tecnología) y restricciones de funcionamiento (tales como retrasos máximos permitidos) [99].

Históricamente los primeros problemas de *disposición* fueron resueltos de manera no automática, mediante la colocación manual de los componentes del circuito, transistor a transistor y conexión a conexión. Este proceso requiere un gran esfuerzo de diseño con una alta probabilidad de comisión de errores. Los conceptos introducidos en el ya clásico libro de Mead y Conway [76], facilitaron la aparición de los editores de máscaras, tales como KIC o MAGIC [74]. Mediante el empleo de estas herramientas, y otras presentadas por diferentes compañías comerciales, es posible el diseño de un circuito (o de partes de un circuito) con una geometría bastante libre. Este estilo de diseño es conocido como *diseño totalmente a medida* (*full custom design*).

La automatización del diseño requiere la disposición de los bloques del circuito en estructuras regulares, como se verá a continuación. Un avance importante en la dirección de una mayor escala de integración, se produce con la definición de librerías de celdas que contienen la *disposición* de los bloques funcionales básicos.

El concepto de celda permite la definición de un nuevo estilo de diseño: el diseño con *celdas estándar* (*standard cells*). En este estilo, las celdas (que tienen la misma altura) se disponen en una estructura de *filas* (*rows*) las cuales se encuentran separadas por unos *canales de conexión* (*routing channels*) por los que discurren las pistas de conexión entre las diferentes celdas.

Esta idea proporciona la posibilidad de automatizar el trazado de las máscaras, y permite avances espectaculares obtenido por la aplicación a este problema de diferentes algoritmos de optimización.

Simultáneamente, y centrando el problema en el campo digital, se propone la idea de predifundir en filas sobre la oblea parejas de transistores *pMOS y nMOS*,

de forma que mediante máscaras de conexiones se pueda definir la funcionalidad del circuito. Los canales de *conexión* pasan a ser de tamaño fijo; obteniéndose la estructura conocida como *matriz de puertas* (*gate array*). En esta estructura los problema de *colocación* y *conexionado* son similares al de un circuito con *celdas estándar* y por ello son susceptibles de ser automatizados, con las mismas técnicas. Los mismos programas de optimización son aplicables a otros estilos de diseño como los denominados de *macroceldas* (*macro-cells*) y de *mares de puertas* (*sea of gates*), aunque en éstos, una menor regularidad de su estructura, hace necesaria la introducción de otros aspectos que no son considerados en esta tesis.

1.2.1 Formulación del Problema de la *Disposición*

En este apartado se presenta una formalización del problema de la *disposición* que recoge las ideas básicas, que son comunes a todas las tecnologías, y se introduce la notación básica del problema [71]. Hay dos restricciones fundamentales que caracterizan la *disposición* de componentes en un circuito integrado:

1. Existe una dimensión finita λ, de manera que todas las medidas geométricas son múltiplos de λ. Es, por tanto, un problema de *Optimización Discreta.*

2. Es un problema plano. El proceso de fabricación se produce en un número finito de capas de metalización, mayor que 1, pero pequeño (4 como mucho).

Problema de la *Disposición*

- *Elementos:* Un hipergrafo $G = (V, E)$, donde V son los vértices (que representan las celdas con su tamaño) cuyo número es n, y E son los enlaces (que representan la conexiones, y que tienen asociados una función de pesos que representa su importancia relativa en el diseño final), cuyo número es m.

- *Configuraciones:* Cualquier *Colocación* de todas las celdas sobre una red de puntos que discretizan la oblea de silicio, conocida como *rejilla* (*grid*), de forma que los vértices de G se colocan sobre los puntos del *rejilla* sin posibilidad de solapamiento. Las conexiones, igualmente, se colocan de manera que sólo se

admiten cruces por medio de un punto de la *rejilla*. Formalmente un enlace $e \in E$ no incide en el vértice $v \in V$ al menos que un nodo de la *rejilla* correspondiente a v incida sobre el *árbol de Steiner* correspondiente a e.

- *Soluciones:* Todas las posibles combinaciones en las que no ocurre que dos *árboles de Steiner* diferentes tengan enlaces que descansan sobre los mismos puntos de la *rejilla*, si bien se admiten cruces entre ellos.
- *Objetivo:* Minimizar el área del menor rectángulo que contiene el circuito.

Kramer y Van Leewen [64] probaron que el problema de la *disposición* es del tipo *fuertemente NP-duro* [1].

1.3 Descomposición del problema

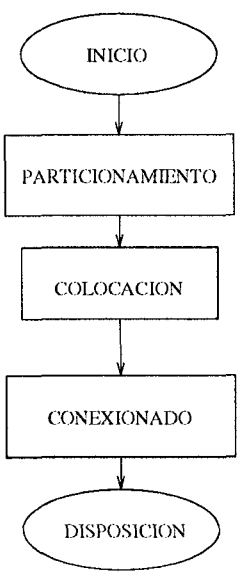


Figura 1-2: Partes de una disposición

La aproximación más común al problema de la *disposición* descompone el problema general en distintas fases que, a pesar de estar fuertemente relacionadas,

[1] La complejidad de los algoritmos computacionales es discutida en [35] y [3]. El apéndice I ofrece una breve definición de los términos principales

suelen resolverse de manera secuencial e independiente (figura 1-2). Tradicionalmente el problema de la *disposición* se descompone en [44]: *particionamiento* del circuito (*circuit partitioning*), *colocación* (*placement*) y *conexionado* (*routing*). En esta tesis tan sólo se consideran los problemas del *particionamiento* y la *colocación* en un estilo de diseño basado en *celdas estándar*, siendo los resultados extrapolables a un circuito diseñado con el estilo *matriz de puertas*.

El problema del *particionamiento* trata de establecer la jerarquía de los bloques que constituyen el circuito. Para ello se divide el circuito original en diferentes subcircuitos, de manera que la conexión entre ellos sea mínima. El objetivo del *particionamiento* puede ser la realización multichip de un circuito de gran dimensió. En esta tesis, nuestra preocupación en el problema del *particionamiento* es debida a que una gran parte de los algoritmos heurísticos empleados en el problema de la *colocación*, emplean, de una manera u otra, algoritmos de *particionamiento*.

En la fase de *colocación*, las bloques básicos del circuito son colocados (siguiendo el estilo de diseño seleccionado) con el objetivo de minimizar el área de las interconexiones (nótese que el área de las celdas es constante, salvo que sea necesario introducir *celdas de paso* (*feedthroughs*)).

Parece una paradoja intentar en esta fase una colocación con mínima interconexión, cuando el conexionado de las celdas se hace en una fase posterior. Como el proceso de *conexionado* es complejo y requiere un elevado tiempo de cálculo, no es posible (ni siquiera es interesante) realizar un *conexionado detallado* para cada posible solución que se considere durante el proceso de *colocación*. En esta fase nos contentaremos con una estimación del área de conexionado requerida para cada una de estas soluciones; estimación que, aunque grosera, sea fácil de calcular. Como mucho, a lo largo de esta fase, haremos una asignación de cada una de las conexiones a los canales de conexionado (lo que se conoce como *conexionado global* (*global routing*)), pero no resolveremos con detalle la distribución de las pistas en cada uno de estos canales (lo que se conoce como *conexionado de detalle o de canal* (*channel or detailed routing*)).

En su conjunto, y desgraciadamente en cada una de sus partes, el pro-

blema de la *disposición* resulta un problema combinatorio de una gran complejidad. Incluso versiones muy simplificadas del problema de la *disposición* resultan ser NP–completos o NP–duros. Técnicas clásicas de optimización basadas en información procedente del gradiente de la función de costes conducen con gran probabilidad a mínimos locales, que en la mayor parte de los casos, están muy lejos de ser la mejor solución del problema. En Lengauer [71] se ofrece un estudio muy detallado de la complejidad de cada una de las fases en que se divide el problema de la *disposición* y de las distintas simplificaciones con que han sido abordadas en la literatura.

Que el problema es de naturaleza combinatoria es la base del desarrollo de esta tesis. El clasificarse como NP–duro y tener una dimensión típica grande obliga a emplear técnicas heurísticas para su resolución. En esta tesis presentaremos las técnicas heurísticas que han sido empleadas para la resolución de los problemas de *particionamiento* y *colocación*. Mediante el estudio de estas técnicas descubriremos sus puntos débiles y presentaremos nuevos algoritmos que permiten obtener mejores resultados. Para dar una prueba de la bondad de los algoritmos propuestos, éstos se han codificado en un conjunto de rutinas y se han incorporado al conjunto de herramientas de diseño de la Universidad de Berkeley, denominado *Octtools 5.1*. Los resultados obtenidos se han comparado con los que resultan del paquete de *disposición* de circuitos de alta escala de integración, denominado *TimberWolfSC-4.2*. Los circuitos que se han empleado como patrones de prueba no han sido escogidos de una forma cualesquiera, sino que proceden de un banco de *circuitos patrones* (*benchmark circuits*) proporcionados por el MCNC (Microelectronic Center of Noth Carolina), algunos de los cuales fueron distribuidos en *ACM/IEEE Physical Design Workshop, celebrado en Abril de 1987*.

1.4 Diseño de celdas analógicas

Si bien el diseño de circuitos integrados digitales está en una fase muy madura de automatización, no ocurre lo mismo con el diseño analógico [53]. Aún hoy día, una gran parte del tiempo total de diseño de un circuito mixto se dedica al diseño de la parte analógica, aunque su tamaño relativo sea muy pequeño en

comparación con la parte digital.

Es difícil hablar de elevada escala de integración al referirnos a circuitos analógicos, pues éstos rara vez pasan de algunos centenares de transistores. De manera que el problema de la *disposición* en circuitos analógicos no suele tener la complejidad que se presenta en un circuito digital VLSI. Aún hoy día es común que el circuito analógico sea trazado manualmente con ayuda de algún programa de trazado de máscaras y compactación.

Un problema importante que se presenta en el diseño de celdas analógicas es el de elegir el tamaño óptimo de sus componentes (normalmente en un circuito CMOS, la relación W/L de sus transistores) para que el circuito cumpla unas ciertas especificaciones con mínima área. Nuevamente nos encontramos con un problema computacionalmente complejo que requiere el empleo de técnicas heurísticas para su resolución. Los mismos problemas de mínimos locales que comentamos para el problema de la *disposición* se presentan nuevamente, si bien la dimensión del problema (es decir, el número de variables a optimizar) es normalmente inferior.

En esta tesis hemos querido aprovechar la experiencia obtenida de la aplicación de algoritmos heurísticos de optimización al problema de la *colocación* de celdas, a la resolución del problema de la síntesis de celdas analógicas. De esta manera, en los capítulos finales de la tesis, se introduce el problema de la síntesis de celdas analógicas y se proponen diferentes heurísticas para su resolución.

Capítulo 2

El Problema del Particionamiento

2.1 Introducción

El objetivo del *particionamiento* de un circuito es su división en porciones, que van a ser realizadas de manera independiente, ya sea en bloques diferentes dentro de un mismo circuito integrado, ya sea en varios componentes discretos. En este caso la función de coste a minimizar es el número de conexiones que unen los diferentes bloques. Generalmente se incluye en la función de coste un término que tiene en cuenta la diferencia entre el tamaño de cada bloque y el tamaño deseado. Un segundo aspecto, aún más importante, del particionamiento, es su empleo como una herramienta por los algoritmos de *colocación* y *conexionado*, formando parte de técnicas del tipo "divide y vencerás". En este caso la función de coste no sólo incluye la capacidad del conexionado entre bloques, también puede incluir otros términos específicos del problema a resolver.

El problema del *particionamiento*, incluso en los casos más simples, es NP–duro, y en la literatura se han aplicado multitud de heurísticas para su resolución. De todas ellas es, sin lugar a dudas, la debida a Fiduccia y Mattheyses la que mayor aceptación ha encontrado, por lo que a ella y a sus diferentes versiones, mejoras y extensiones dedicaremos gran atención.

Al final de este capítulo se presenta una aportación del autor al problema del *particionamiento* que permite obtener muy buenos resultados cuando se compara

con los algoritmos clásicos.

Antes de continuar hemos de indicar que el problema del *particionamiento* no es exclusivo del problema de la *disposición* de circuitos integrados, también aparece en múltiples campos de la ingeniería. Poe ejemplo, en la ingeniería mecánica y civil se presenta en el problema de la partición de estructuras, en ingeniería eléctrica, en la partición de redes de transmisión de energía eléctrica, en ingeniería de las comunicaciones, en la división de redes de comunicaciones para maximizar el flujo de datos. Por ello, existen en la literatura multitud de trabajos de interés teórico y práctico sobre el problema del particionamiento. Se pueden encontrar estudios teóricos acerca de la complejidad del problema, existencia de separadores, algoritmos de aproximación, determinación de cotas superior e inferior, etc. Se pueden encontrar también heurísticas que tratan de resolver las diferentes versiones del problema. No es nuestra intención aquí hacer un estudio detallado de toda la bibliografía al respecto, de manera que tan sólo consideraremos aquellos algoritmos y resultados que son relevantes al problema de la *disposición* de circuitos integrados.

2.2 Definiciones y Complejidad

Todas las técnicas de particionamiento de circuitos consideran el circuito como un grafo o hipergrafo, donde los vértices son las celdas del circuito, y los enlaces o hiper–enlaces son las conexiones entre celdas. Por ello usaremos indistintamente los términos celda y vértice y los términos enlace y conexión.

Definición: *Problema de Particionamiento* [71]

- *Elementos:* Un hipergrafo $G = (V, E)$ con n vértices, una función que pondera los vértices $w : V \longrightarrow N$, una función que pondera los enlaces $c : E \longrightarrow N$ un número $r \in N$ de partes, un tamaño máximo de cada parte $B(i) \in N$ y un tamaño mínimo de cada parte $b(i) \in N$, con $i = 1..r$.

- *Configuraciones:* Todas las *r–particiones* $\Pi = (V_1, ..., V_r)$ de V, con $V_i \subset V$, $V_i \neq \emptyset$, y $V_i \cap V_j = \emptyset$ para $i, j \in \{1, ..., r\}$ y $\cup_{i=1}^{r} V_i = V$.

- *Soluciones:* Todas las particiones $\Pi = (V_1, ..., V_r)$ tales que $b(i) \leq w(V_i) \leq B(i), \forall i = 1, ..., r$.

- *Objetivo:* Minimizar $c(\Pi) = \frac{1}{2} \sum_{i=1}^{r} \sum_{e \in E_{ext,i}} c(e)$,
  donde $E_{ext,i} = \{e \in E \mid e \cap V_i \neq \emptyset, e \backslash V_i \neq \emptyset\}$, es el conjunto de enlaces externo a la *parte* V_i.

Cada partición está obligada a obedecer una restricción de tamaño denominado *criterio de balance*. En la definición anterior, esta restricción de tamaño se ha materializado en la forma de un tamaño máximo permitido $B(i)$ y un tamaño mínimo permitido $b(i)$, para cada parte. El tamaño de la partición se define como la suma de los pesos de cada uno de los vértices que contiene. De esta forma es posible trabajar con circuitos compuestos por celdas de diferente tamaño.

Los enlaces que conectan diferentes partes en Π, pertenecen al corte. El coste $c(\Pi)$ de la partición se llama *tamaño del corte*, y se define como la suma ponderada de los enlaces que conectan vértices que se encuentran en partes diferentes.

El problema del *particionamiento* es un problema NP–duro, incluso cuando se introducen severas restricciones. Por ejemplo, la versión más simple del problema, en que G es un grafo, $w \equiv 1$, $c \equiv 1$, $b(i) = 1$, y $B(i) = n$, $\forall i = 1, ..., r$, es aún fuertemente NP–completa.

Existen dos aproximaciones fundamentales al problema del *particionamiento*:

- Una aproximación *constructiva*, que parte de un conjunto de celdas *semilla* y va agrupando celdas alrededor de ellas, hasta formar la partición final. Esta aproximación recibe el nombre de *agrupamiento* (*clustering*).

- La segunda aproximación (que proporciona mejores resultados) parte de una solución inicial y va alcanzando mejores soluciones mediante el intercambio de celdas entre unos y otros bloques.

En lo que sigue tan sólo consideraremos esta segunda aproximación.

Bipartición. El caso r=2 merece especial atención y es conocido como *bipartición* o *bisección* (si ambas partes son de igual tamaño). El problema de la

bipartición es también un problema NP–duro. De hecho, la *bisección* restringida a grafos sigue siendo un problema fuertemente NP–completo.

2.3 Algoritmos de Búsqueda Dirigida por intercambio de celdas

Estas técnicas parten de una solución inicial escogida (o no) aleatoriamente y tratan de mejorar la función de costes mediante movimientos locales de las celdas de un bloque a otro. Son técnicas muy robustas que permiten considerar hipergrafos, costes arbitrarios de vértices y diferentes criterios de balance.

Se han propuesto mutitud de algoritmos de esta naturaleza. Casi todas las realizaciones prácticas son variaciones del algoritmo propuesto por Fiduccia y Mattheyses [34]. Para introducir el algoritmo de Fiduccia–Mattheyses es necesario exponer con anterioridad el algoritmo debido a Khernigan y Lin [55] que dió origen al mismo.

Algoritmo de Kernighan y Lin

Una primera heurística propuesta originalmente para la bipartición de grafos fue formulada por Kernighan y Lin [55], en 1970.

El algoritmo parte de una partición aleatoriamente escogida que cumpla el criterio de balance. La idea de este algoritmo es la de encontrar, de una forma ordenada, una secuencia de intercambios de parejas de celdas entre particiones que posibilite un corte de menor coste. Para ello se asigna un incremento de coste potencial a cada pareja de celdas. El algoritmo asume un coste unitario de cada vértice, de forma que el intercambio de dos vértices mantiene el criterio de balance. Se selecciona aquella pareja de celdas que produce el mayor decremento de la función de costes. Estas celdas se intercambian de partición y se marcan (bloquean) para evitar que vuelvan a ser consideradas durante esta etapa del algoritmo. Se repite el proceso de intercambio de celdas hasta acabar con todas las celdas no marcadas. Es necesario recalcar que se aceptan incluso decrementos negativos del coste, si no

hay un movimiento un mejor. La secuencia de movimientos se va guardando en una lista, y un puntero apunta a la posición de la lista en que se alcanzó el tamaño de corte mínimo.

Agotados todos los movimientos posibles, se vuelve a la situación inicial y se procede a intercambiar las celdas en el mismo orden que antes hasta alcanzar la configuración que durante el proceso de intercambio de celdas dió el menor coste. El algoritmo comienza ahora una nueva etapa tomando como solución inicial la solución obtenida en la etapa anterior. El algoritmo deja de realizar nuevas etapas cuando ya no encuentra una disminución del coste entre dos etapas consecutivas.

Nótese que la potencia del algoritmo radica en:

1. Marcar las celdas que se han movido (o marcar el propio movimiento) para evitar que vuelvan a producirse soluciones ya consideradas y se alcancen ciclos límites.

2. Admitir cambios de celdas aunque ello lleve aparejado un aumento del coste. Esta decisión dota al algoritmo de un mecanismo de escape de mínimos locales.

3. Una estructura de ganancias de celda que permite una eficiente actualización, y selección de las mejores celdas para el inetrcambio.

El algoritmo se expone de una manera esquemática en la figura 2-1.

Se puede comprobar fácilmente que en cada etapa no es necesario agotar todos los movimientos posibles de celdas, lo que permite acelerar el algoritmo.

Este algoritmo es muy robusto, y es posible fijar celdas a zonas determinadas. Sin embargo adolece de ciertas características, como la de no tratar el caso de vértices de distinto peso. El algoritmo se propone únicamente para grafos, y su complejidad es [107] $O(n^2 \times \log n)$. Es posible extender el algoritmo al caso de un hipergrafo.

Algoritmo de Fiduccia y Mattheyses

El algoritmo de Fiduccia–Mattheyses [34] recoge la idea del algoritmo de Kernighan–Lin, e introduce un conjunto de modificaciones que reducen su comple-

jidad hasta hacerlo lineal con el número de celdas, manteniendo la calidad de los resultados. Por ello maneja celdas individuales, de forma que es posible manejar particiones con balances no equilibrados y grafos con vértices ponderados (lo que es más parecido al concepto de celda física). El algoritmo se plantea directamente para tratar la bipartición de hipergrafos. Por último, maneja una estructura de datos muy eficiente que selecciona los movimientos y consigue optimizar el tiempo de ejecución del algoritmo.

Consideramos que la red $N = (V, E)$ se encuentra inicialmente dividida en dos partes, A y B.

Definimos el coste externo del vértice $v_i \in A$ como:

$$E(i) = \sum_{e \in E_{ext,i}} c(e) \tag{2.1}$$

donde: $E_{ext,i} = \{e \in E \mid \{v_i\} = e \cap A\}$.

$E(i)$ representa el número de enlaces que desaparecen del corte cuando v_i cambie de bloque.

Análogamente el coste interno del vértice $v_i \in A$ se define como:

$$I(i) = \sum_{e \in E_{int,i}} c(e) \tag{2.2}$$

donde: $E_{int,i} = \{e \in E \mid v_i \in e \ y \ e \cap B = \emptyset\}$

$I(i)$ representa la nueva contribución al corte debida a las conexiones de la celda en el interior de su bloque.

Por tanto, la ganancia (disminución del tamaño del corte) que se obtiene al cambiar de bloque el vértice v_i viene dado por:

$$D(i) = E(i) - I(i) \tag{2.3}$$

Una aportación de interés de este algoritmo está en la estructura de datos que selecciona la celda a mover. Se trata de dos matrices de punteros monodimensionales que denominaremos *Caja A* y *Caja B* cuyos índices se mueven en el rango $[-d_{max} \times b_{max}, d_{max} \times b_{max}]$, donde d_{max} es el grado máximo de una celda o vértice y b_{max} es el coste máximo de un enlace. Los punteros de la *Caja A* apuntan a

listas lineales que representan celdas no bloqueadas del bloque A, ordenadas por ganancia. Existe un puntero *Maxgan A* que apunta a la lista de mayor ganancia. Esto se repite igualmente para la *Caja B*. Otras dos listas representan las celdas bloqueadas de A y B: Son las denominadas *Bloq A* y *Bloq B*. El procedimiento de selección consiste en tomar una celda de la *Caja A* ó de la *Caja B*, si hay alguna, que esté apuntada por su respectivo puntero *Maxgan A* ó *Maxgan B* y que cumpla el criterio de balance. Si hay una celda en cada bloque que cumpla estos requisitos, se selecciona la de mayor ganancia. Si tienen igual ganancia, se escoge una de las dos de forma aleatoria. La celda seleccionada se cambia de parte y se marca (bloquea) para evitar ser seleccionada de nuevo en esta etapa del algoritmo. En este punto deben actualizarse las distintas listas y punteros. Una buena elección por parte de los autores, de la estructura de estas listas y punteros, permite una rápida actualización. Este proceso continúa hasta bloquear todas las celdas, quedando una lista de celdas bloqueadas que es, en realidad, una secuencia de movimientos. En este punto termina una etapa del algoritmo. Se vuelve a la situación de partida y se repiten los movimientos hasta alcanzar la configuración de mínimo coste. En la práctica es posible terminar la etapa sin necesidad de agotar todas las celdas, si mantenemos una lista con el número de enlaces que tienen celda bloqueadas en los dos lados de la partición, lo que permite acelerar el algoritmo.

Al igual que el algoritmo de Khernighan–Lin, este algoritmo tiene en su forma de selección de celdas a mover, un mecanismo inherente de escape de mínimos locales, al permitir incrementos del tamaño del corte.

El criterio de parada tiene lugar cuando una etapa no mejora el corte de la anterior. La complejidad de cada etapa es lineal con el número de celdas. La experiencia muestra que el algoritmo deja de mejorar cortes al cabo de unas 7 iteraciones, por lo que el algoritmo en su conjunto tiene una complejidad $O(n)$. La figura 2-2 muestra un diagrama simplificado del algoritmo de Fiduccia–Mattheyses.

2.3.1 Mejoras y Extensiones de los Algoritmos

Algoritmo de Krishnamurthy

En el algoritmo de Fiduccia–Mattheyses existe una indeterminación en el caso de que haya varias celdas con la misma ganancia y que satisfacen de igual manera el criterio de balance. En [66], su autor introduce una selección inteligente de entre todas las celdas candidatas al movimiento. En igualdad de condiciones, selecciona la celda que con mayor probabilidad reduce el tamaño del corte en futuros movimientos. De esta manera se extiende el concepto de ganancia de celda $D(i)$, a un vector de ganancias de celda $(D_1(i), ..., D_k(i))$. $D_1(i)$ es la ganancia original del algoritmo de Fiduccia–Matheyses y $(D_2(i), ..., D_k(i))$ ganancias de orden superior que indican el número de las celdas que reducen su ganancia $(D_1(i), ..., D_{k-1}(i))$ cuando la celda v_i cambia de partición.

En la figura 2-3 se reproduce una situación donde un criterio de segundo orden llevaría a seleccionar la celda v_2, dado que al mover esta celda, v_3 incrementa su ganancia D_1 en una unidad.

Algoritmos que emplean una modificación de la función de costes

DIferentes modificaciones al algoritmo de Fiduccia–Mattheyses se han publicado en la literatura. En [122], el criterio de balance se incluye en la función de coste. De esta manera, si A–B es una partición de la red N, entonces el objetivo es minimizar la función denominada *corte relativo*, definida como

$$R_{AB} = \frac{tamano\_corte(A, B)}{|A| \times |B|} \tag{2.4}$$

Recientemente, en [86] se propone relajar el criterio de balance, e incluir los desequilibrios en el tamaño de cada una de las partes, como un término adicional de la función de costes.

Otras modificaciones de la función de costes se dirigen hacia la solución de problemas específicos. Así, en [100] se modifica la función de costes para tener en cuenta ciertas particularidades del problema de la *colocación* de celdas estándar.

Algoritmos de Compactación

Son algoritmos que buscan trabajar con redes densas, es decir, redes cuyas celdas tienen un elevado número de conexiones. Para ello realizan una compactación de celdas para formar bloques de mayor tamaño. Luego particionan varias veces la nueva red compactada (en donde cada bloque es una celda), partiendo de diferentes soluciones aleatorias. Se escoge la mejor de las particiones de la red compactada. Finalmente, se deshace la compactación , y se utiliza la red resultante como solución inicial a un algoritmo del tipo Fiduccia–Mattheyses. Se comprueba experimentalmente que al compactar la red desaparecen muchos mínimos locales no deseados, manteniéndose la calidad de las soluciones. Además, la red compactada es de una dimensión menor que la red original, por lo que resulta muy ventajoso realizar varios intentos de partición sobre la red compactada y no sobre la original.

Los algoritmos de *particionamiento* por compactación se diferencian entre sí por los diferentes métodos empleados para compactar la red y por los mecanismos que se emplean para probar distintas particiones en la red compactada. Ejemplos de estos algoritmos se pueden encontrar en [42], [15] y [16], que emplean un algoritmo de *acoplamiento perfecto* (*perfect matching*) para la compactación de las redes. En [81] se propone una compactación mediante un algoritmo de *agrupación*, basado en la regla de Rent [67]. En [26] se emplea un algoritmo recursivo que utiliza el algoritmo de *corte relativo*, anteriormente comentado para particionar la red en bloques.

Extensión al caso de *particionamiento* en múltiples bloques

Una extensión inmediata al problema del *particionamiento* en r bloques consiste en aplicar de manera jerárquica un algoritmo de bisección. Otras aproximaciones parten de una partición aleatoria en r bloques y utilizan alguna modificación de las técnicas iterativas de intercambio de celdas antes comentadas. En este sentido, el algoritmo de [110] es una extensión directa del algoritmo de Fiduccia–Mattheyses. El algoritmo de [97] extiende el concepto de *vector de ganancia* introducido por Krishnamurthy al caso de r bloques. En [98] la misma autora considera otras funciones de coste.

2.4 Otros Algoritmos de *particionamiento*

En la literatura existen otras muchas soluciones al problema del *particionamiento*, algunas de las cuales se comentan a continuación.

1. Técnicas de *flujo máximo* (*maxflow*). El teorema de corte mínimo y flujo máximo establece que el problema del flujo máximo en un grafo es equivalente al problema del corte mínimo. Esto ha llevado a aplicar técnicas de flujo máximo al problema del *particionamiento*. De especial relevancia son los trabajos de Bui [15], Kahng [54], Leighton y Rao [70] y C.K.Chen [25]. Los trabajos en esta dirección han permitido un mejor conocimiento del problema del *particionamiento*, pero con escasos resultados prácticos.

2. Técnicas de optimización no lineal. Bajo ciertas restricciones, el problema del *particionamiento* puede ser formulado como un problema de *asignación cuadrática* . Al igual que en el caso anterior, los trabajos en este sentido [5], [8], han permitido un mayor conocimiento del problema (determinación de cotas superior e inferior,...etc), pero escasos resultados prácticos.

3. Técnicas de *enfriamiento simulado* (*simulated annealing*). Las técnicas de *enfriamiento simulado* se basan en el algoritmo de Metrópolis que asemejan un problema de optimización al proceso de enfriamiento lento de un gas hasta formar un sólido cristalino. Por su importancia los algoritmos de enfriamiento simulado, de aplicación muy general, son estudiados en detalle en el apartado 3.5. Aunque existen muchas aplicaciones del algoritmo al problema de *particionamiento* ([57], [46] y [69], entre otros), en la práctica los resultados obtenidos difícilmente justifican los elevados tiempos de computación. Como es conocido, los algoritmos de enfriamiento simulado no compiten ventajosamente en aquellos problemas para los que se conocen heurísticas muy eficientes, como es el caso del *particionamiento*. La potencia de estos algoritmos se pondrá de manifiesto cuando tratemos los problemas de la *colocación* de celdas y de la síntesis de celdas analógicas. De mayor interés resultan otras técnicas probabilísticas de menor complejidad computacional como, por ejemplo, el algoritmo

denominado *evolución estocástica* de Saab y Rao [96]–[97].

4. Técnica de *búsqueda tabú* (*tabu–search*). La técnica de búsqueda tabú fue propuesta por F.Glover [40]–[41] para resolver problemas de naturaleza combinatoria. En el capítulo 6 se hace una descripción detallada de esta técnica. En [4] se presenta una aplicación de este algoritmo al problema del *particionamiento.*

5. Uso de *redes neuronales.* El problema del *particionamiento* en grafos puede formularse como la minimización de una función hamiltoniana de las que es capaz de optimizar una red de Hopfield [50], [114]. Por tanto, es posible emplear una red de este tipo al problema de *particionamiento.* No obstante, esta resolución no evita la elevada posibilidad de quedar atrapados en un mínimo local de la función de coste. El empleo de redes neuronales de *campo medio normalizado* (*normalized mean field nets* [9], [118]) que introduce un término de enfriamiento simulado en la red neuronal permiten evitar la caída en mínimos locales, aunque los parámetros del proceso son empíricos y muy dependiente del problema. Además, la extensión al caso de hipergrafos no es inmediata [125]. La compeljidad de la red crece con una potencia mayor que 2 con el tamaño del circuito.

2.5 Aportaciones al problema del corte mínimo

Dada la gran importancia que tienen los algoritmos de corte mínimo en la resolución del problema de la *disposición* de circuitos integrados, a lo largo del desarrollo de esta tesis se han hecho diferentes pruebas con el objetivo de obtener algoritmos de corte eficientes y estables.

Para ello, se codificó en un paquete de rutinas el algoritmo de Fiduccia–Mattheyses y algunas de sus variante. En general, se observó la gran dependencia de estos algoritmos con la solución de partida. Las variaciones en el corte final pueden ser de hasta el 50 % en dos veces sucesivas que se corre el algoritmo. Esto nos hizo ver la necesidad de contar con algún mecanismo que permitiera asegurar un coste final más próximo al valor óptimo.

Por otra parte, las experiencias realizadas con el empleo de los algoritmos de compactación, nos permiten afirmar que una buena compactación de la red permite conservar el valor del corte óptimo en la red compactada, a la vez que se eliminan mínimos locales no deseados. Por ello, decidimos intentar una mejora del algoritmo de Fiduccia–Mattheyses mediante un algoritmo de compactación que emplea bloques compactados casi óptimos.

Para conseguir que los bloques compactados sean casi óptimos, emplearemos el propio algoritmo de Fiduccia–Mattheyses para su generación. Esta aproximación es similar a la realizada por Cheng en [26], donde emplea el algoritmo de *corte relativo* para la generación de los bloques compactados.

Una vez planteada la idea general, a continuación se describe en detalle el algoritmo propuesto y se discuten los resultados obtenidos.

Como circuitos de prueba se han tomado unos *circuitos patrón* (*benchmarks*) que se encuentran en la base de datos del MCNC (Microelectronic Center of North Carolina). Las características de estos circuitos se detallan en el apéndice III.

2.5.1 Algoritmo propuesto

Consideremos una red $N = (V, E)$, donde V es el conjunto de nodos de la red y E es el conjunto de enlaces (hiperenlaces en el caso general). Se pretende obtener una bipartición de la red N con el criterio de balance

$$tam1 < tamano\ de\ cada\ particion < tam2$$

En lo que sigue g y $nveces$ son parámetros del algoritmo.

El algoritmo propuesto consta de las siguientes fases:

1. Inicializar $\Phi = V$ y calcular el tamaño total del circuito $|V| = tam\_tot$.

2. Sea V^* un subconjunto de Φ, tal que $|V^*| = max_{\{V_i \in \Phi\}}|V_i|$. Mientras $|V^*| > tam\_tot/g$, repetir el paso 3, en caso contrario ir al paso 4.

3. Hacer $\Phi = \Phi - V^*$. Aplicar el algoritmo de Fiduccia–Mattheyses a V^* hasta conseguir un corte (A, A'), donde $V^* = A \cup A'$. Hacer $\Phi = \Phi \cup \{A, A'\}$. En

esta fase, el criterio de balance del algoritmo de Fiduccia–Mattheyses es muy poco restrictivo, de manera que se permiten particiones muy desequilibradas.

4. Construir la red compactada $H = (\hat{V}, \hat{E})$, donde cada elemento de Φ es una celda.

5. Aplicar *nveces* el algoritmo de Fiduccia–Mattheyses a la red H con el criterio de balance $tam1$, $tam2$.

6. Usar el mejor resultado del paso anterior como solución inicial de la red original N. Aplicar el algoritmo de Fiduccia–Mattheyses una vez a la red N con el criterio de balance $tam1$, $tam2$.

2.5.2 Selección de los parámetros del algoritmo

Para estudiar el efecto de los parámetros del algoritmo (*nveces* y g) en la solución obtenida, se han realizado numerosas pruebas sobre circuitos de prueba con diferentes valores de los parámetros. La figura 2-4 muestra los resultados obtenidos en la bisección de tres circuitos, (cuyas características se detallan en el apéndice III), para *nveces* = 10 y diferentes valores de g. Puede apreciarse que el valor de g que proporciona resultados óptimos se encuentra entre 50 y 100, dependiendo del circuito. Nótese que un valor de g elevado, proporciona un número muy grande de bloques, por lo que el algoritmo propuesto se reduce a repetir 10 veces el algoritmo de Fiduccia Mattheyses. Por ello, los tiempos de cálculo son elevados, sobre todo si el circuito es grande, como pasa con el denominado *biomed*. Un valor demasiado pequeño de g, conduce a una compactación en muy pocos bloques.

En la figura 2-5 se muestra el efecto de *nveces* para $g = 50$. Por supuesto, cuanto mayor sea el valor de *nveces*, mejor será la solución obtenida. Valores de *nveces* entre 10 y 50 proporcionan buenos resultados con tiempos de ejecución no muy altos.

2.5.3 Resultados obtenidos sobre un conjunto de circuitos patrones

La tabla 2.1 muestra una comparación entre los resultados obtenidos con el algoritmo de Fiduccia–Mattheyses, y el algoritmo propuesto. Los parámetros del problema de corte han sido los siguientes:

1. criterio de balance: $tam_1 = 0.9 * tam\_tot$, $tam_2 = 1.1 * tam\_tot$.
2. número de repeticiones: 10 veces.
3. Para el algoritmo propuesto, el paso 5 se repite 10 veces ($nveces = 10$). El criterio de balance para obtener una red compactada es: $0.5 * tam\_tot$ y $1.5 * tam\_tot$. El valor de g se fija en 50.

En la tabla se observan resultados muy similares a los que aparecen en [26], lo que indica que cualquier algoritmo bueno de compactación da resultados finales buenos y estables.

Circuito	AFM			Propuesto		
	C. med.	C. mín.	D. tip.	C. med.	C. mín.	D. típ.
Struct (s109)	16.7	12	4.19	13.42	11	2.77
Struct (mult16)	75	46	23.09	48.25	41	6.05
Biomed (fan)	462.8	402	30.5	111.75	86	13.9
Primary1 (CIRCUITX)	74.6	56	11.58	52.62	47	3.35
Primary2 (RPROC)	257	197	34.96	173.4	156	12.26

Tabla 2.1: Resultados comparativos frente al algoritmo de Fiduccia–Mattheyses para el caso de la bisección

2.5.4 Extensión al caso de multipartición

El algoritmo presentado en el apartado anterior puede ser extendido al caso de multipartición, de la manera propuesta en [97]. Aquí estamos sobre todo

interesados en la partición bidimensional en cuatro bloques o *cuadrisección*. Para este caso, el algoritmo propuesto es el siguiente:

1. Particionar la red de manera recursiva mediante el algoritmo de Fiduccia–Mattheyses, hasta que los bloques sean de un tamaño menor que $tamano\_total/g$.
2. Formar la red compactada, donde cada bloque es una celda.
3. Cuadriseccionar la red compactada (por el algoritmo de Sanchis [97] o por el algoritmo más simple de Suaris y Kedem [110]). Repetir esta cuadrisección *nveces* y tomar el mejor resultado.
4. Descompactar la red y tomar el resultado como una solución inicial a un algoritmo de cuadrisección sobre la red original.

En la tabla 2.2 se muestra una comparación entre los resultados obtenidos por el algoritmo de Suaris y Kedem [110], y el algoritmo aquí propuesto, para los mismos circuitos del apartado anterior. Nuevamente, se ha escogido $g = 50$ y $nveces = 10$. Puede observarse la notable mejora de las soluciones, y la estabilidad de las mismas.

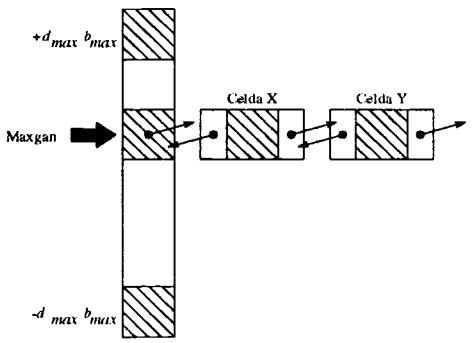
El algoritmo aquí propuesto se ha empleado en el proceso de *colocación* de celdas que se presenta en el capítulo 4.

Circuito	Alg. de Suaris-Kedem			Alg. Propuesto		
	C. med.	C. mín.	D. tip.	C. med.	C. mín.	D. típ.
Struct (s109)	39.2	31	5.67	28.71	28	1.03
Struct (mult16)	391.4	338	26.18	175.37	147	19.5
Biomed (fan)	1068.5	1021	30.56	528.62	497	6.26
Primary1 (CIRCUITX)	348.35	302	16.15	140.0	132	5.76
Primary2 (RPROC)	865.23	756	23.37	497.87	480	4.97

Tabla 2.2: Resultados comparativos frente al algoritmo de Suaris y Kedem para el caso de la cuadrisección

```
    Par:                Matriz [ 1: n/2] de pares de Celdas;
    Coste:              Matriz [ 1: n/2] de Enteros;
    Bloqueado:          Matriz [ 1: n ] de Boolean;
    D:                  Matriz [ 1: n ] de Enteros;
    c:                  Matriz [ 1 : n ][ 1 : n ] de Enteros;
    Mejor Intercambio:  [ 1 : n/2 ] ;
    Mejor Coste:        Entero;
    imin, jmin:         Entero [ 1 : n ];
Calcular   c(i), D(i);
      c(i) es el valor de los pesos de los enlaces;
      D(i) es el valor de   D(i)= Σ c(e) - Σ c(e)
                                  e={v_i,v_j}  e={v_i,v_j}
                                  ∀ v_j≠v_i εB ∀ v_j≠v_i εA
      Se desbloquean todos los nodos de la red
      Mejor Corte=Coste[0]= Corte(A,B);
      Mejor Cambio= [0];
for ( s=1,s<=n/2,s++  ){
   Coste[s]=Infinito;
   for( i=0,j=0;i<n,j<n;i++,j++ ){
      if( v_i εA y Bloqueado[i]=false )&&( v_j ε B y Bloqueado[j]=false ){
         if( 2c(i,j)-D[i]-D[j] < Coste[s]  ){
            Par[s]=(i,j);
            Coste[s]=2c(i,j)-D[i]-D[j];  }
         (imin,jmin)=Pair[s];
         Bloqueado[i]=Bloqueado[j]=True;  }}
   for( i=1;i<=n;i++ ) if( Bloqueado[i]=False  ){
      if( v_i ε A ) D[i]=D[i]-c[i,jmin]+c[i,imin];
      else( D[i]=D[i]-c[i,imin]+c[i,jmin]  ;}
   Coste[s]=Coste[s]+Coste[s-1];
   if( Coste[s]< Mejor Corte  ){
      Mejor Cambio=s;
      Mejor Coste=Coste[s]; }}
for(i=s;s>=1;s--  ){ CambiarPar[s];}
```

Figura 2-1: Algoritmo de Kernighan & Lin



a)

```
Inicializar D(i);
while (haya movimientos posibles){
   seleccionar  v  no bloqueado;
   actualiza la estructura
      reflejando el movimiento de v  ;
   a ade v a la lista de movimientos; }
computar la secuencia de movimientos que
   mejora el corte;
```

b)

Figura 2-2: a) Estructura de Datos Utilizada. b) Esquema General del Algoritmo Fiduccia–Mattheyses

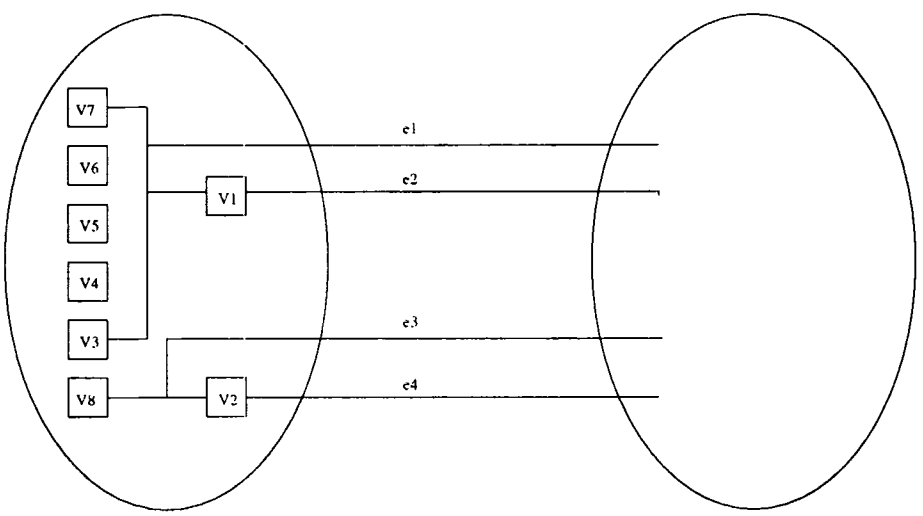


Figura 2-3: Esquema de la mejora de Krishnamurthy

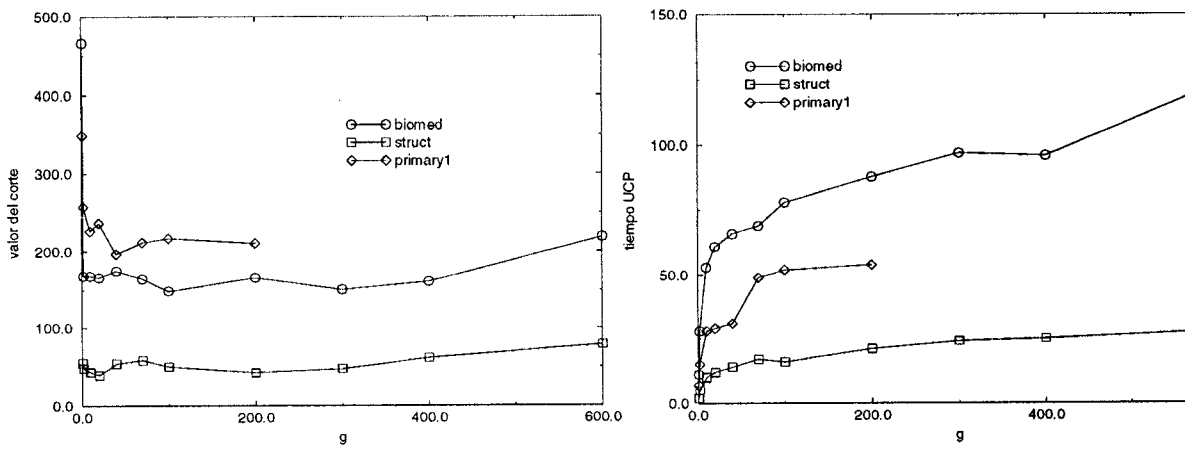


Figura 2-4: Resultados comparativos del algoritmo propuesto para la determinación de g

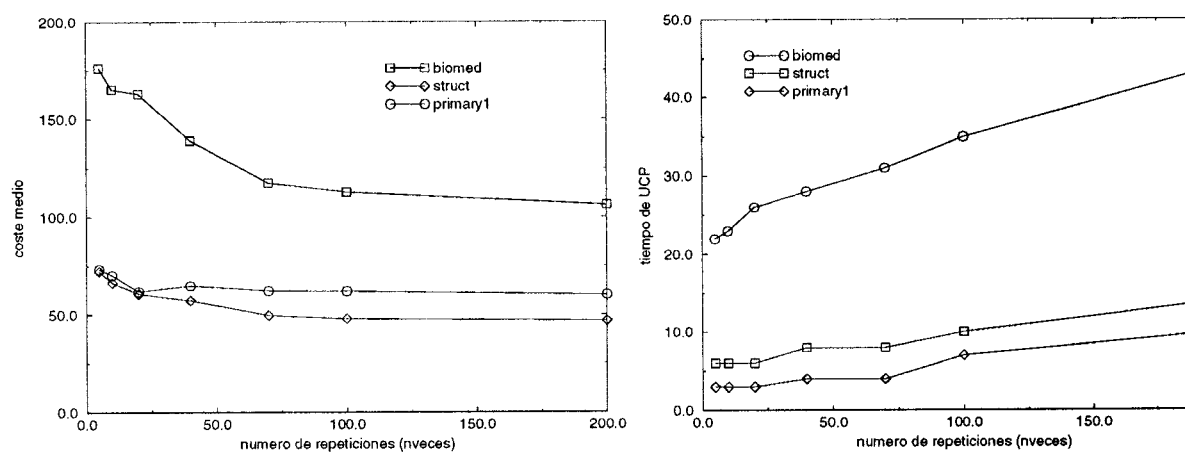


Figura 2-5: Resultados comparativos del algoritmo propuesto para resolver el número de repeticiones ($nveces$)

Capítulo 3

El Problema de la Colocación

3.1 Introducción

Tras la etapa de diseño de un circuito integrado se plantea el problema de distribuir los transistores a lo largo de la oblea. Usualmente esta tarea se asigna a herramientas automáticas, en las que se discurre por diversas etapas, muy dependientes del estilo de diseño.

El caso de las *celdas estándar* consiste básicamente en colocar las celdas de la librería, de manera que el área de la *disposición* completa sea mínima. Nótese que el área total no es sólo la que ocupan los transistores de las celdas por sí mismos, hemos de añadir el área que ocupa el conexionado, sobre la que podemos influir para minimizar el área total.

En el caso de la *matriz de puertas* se plantea una cuestión ligeramente diferente ya que el área del circuito viene dada por un patrón de transistores predifundidos, quedando los *canales de conexionado* delimitados en un área fija. El problema consiste, pues, en asignar las posiciones geométricas de las conexiones, de forma que el *conexionado* quepa en el área reservado para ello.

La similitud de ambos problemas permite comprender que los algoritmos de resolución que se emplean para *celdas estándar* pueden aplicarse igualmente al caso de las *matrices de puertas*. En esta tesis nos limitaremos a este caso y no consideraremos otros estilos como *macroceldas* o *mares de puertas*.

El problema planteado es de tipo combinatorio. Se trata de encontrar aquella secuencia de celdas tal que con esa colocación el área total, incluidas las conexiones, sea mínima. Es un problema de optimización con restricciones.

Habitualmente se describe un circuito por medio de una descripción formal o *referencial* (*netlist*), es decir, mediante una enumeración de las celdas y sus conexiones. En el problema de la *colocación* se pretende buscar una asignación de las coordenadas geométricas de estos elementos en el plano, que satisfaga unos ciertos requerimientos de fabricación y que optimice un conjunto de criterios de área mínima. La bondad de una determinada colocación se mide usualmente en términos del valor de una función de coste.

Debido a las magnitudes que usualmente se manejan en el problema (entre 100 y 100.000 celdas) éste se vuelve matemática y computacionalmente intratable, siendo, para los casos de interés, un problema NP–duro. Por ello hay que buscar métodos heurísticos para su resolución y esto justifica la búsqueda de algoritmos computacionalmente eficientes que proporcionen soluciones casi–óptimas.

El hecho de que el problema de la *disposición* se resuelva en dos etapas (*colocación* de las celdas y *conexionado*) no implica necesariamente que éstas hayan de ser independientes: ambos procesos están íntimamente relacionados y tan sólo problemas computacionales aconsejan la separación de ambos procesos. De hecho, simultáneamente con la fase de *colocación*, se suele realizar una primera asignación de conexiones a los *canales de conexión*, proceso denominado *conexionado global*. La disposición final de las conexiones dentro de cada canal (*conexionado de canal*) se realiza en una fase posterior.

3.2 Estilos de Diseño

En esta sección se describen con detalle los estilos de diseño que se emplean para la *disposición* de los circuitos integrados de muy alta escala de integración. En la figura 3-1 se presenta una clasificación–enumeración de los métodos que se emplean industrialmente. A medida que se desciende en esta clasificación, se incrementa al grado de automatización y se reducen los grados de libertad del diseñador.

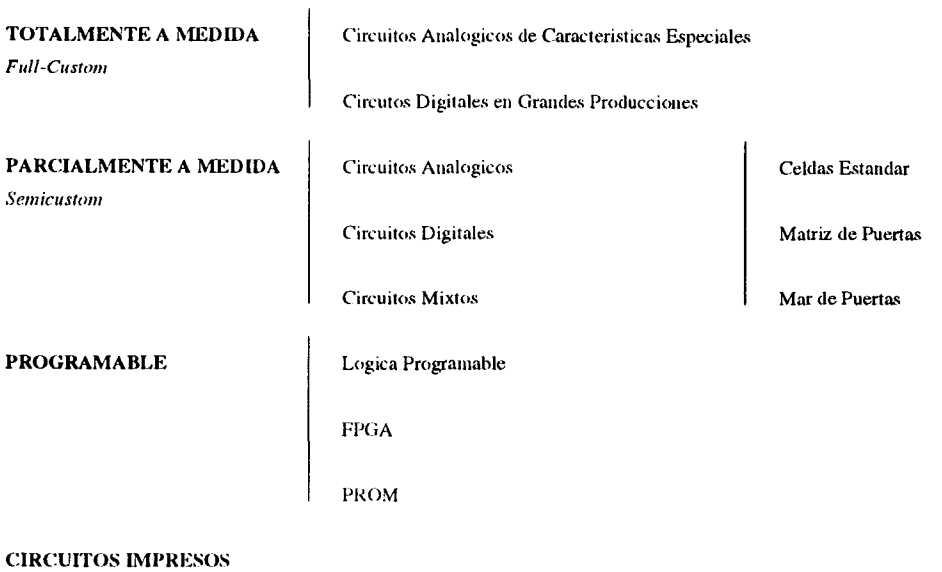


Figura 3-1: Estilos típicos de Disposición

3.2.1 Diseño *totalmente a medida*

Es el método que más trabajo requiere para el diseñador. En él la unidad mínima de trabajo es el transistor. La labor del diseñador consiste en el trazado de cada una de las *máscaras*, conforme a las especificaciones funcionales del circuito.

Si se plantea la cuestión desde el punto de vista de la oblea de silicio, la organización del circuito es absolutamente libre. Ni que decir tiene que el aspecto económico juega un papel fundamental: la compactación obtenida puede ser máxima y por tanto el área de silicio empleada, mínima. Para que el elevado costo del diseño resulte rentable, ha de verse compensado con un número muy elevado de piezas.

Actualmente existen herramientas de apoyo que permiten al diseñador verificar el trabajo a medida que se traza el circuito. Hoy día el diseño *totalmente a medida* se reserva al diseño de los elementos funcionales de las librerías, al diseño de aquellas partes de un circuito que son poco regulares o representan una parte crítica por sus requerimientos de velocidad o tamaño (por ejemplo, el controlador de una UCP), y al diseño de circuitos y bloques analógicos.

En la figura 3-2 se representa un amplificador CMOS hecho *totalmente a medida*. Se observa la ausencia de regularidad en la posición y las dimensiones de los elementos.

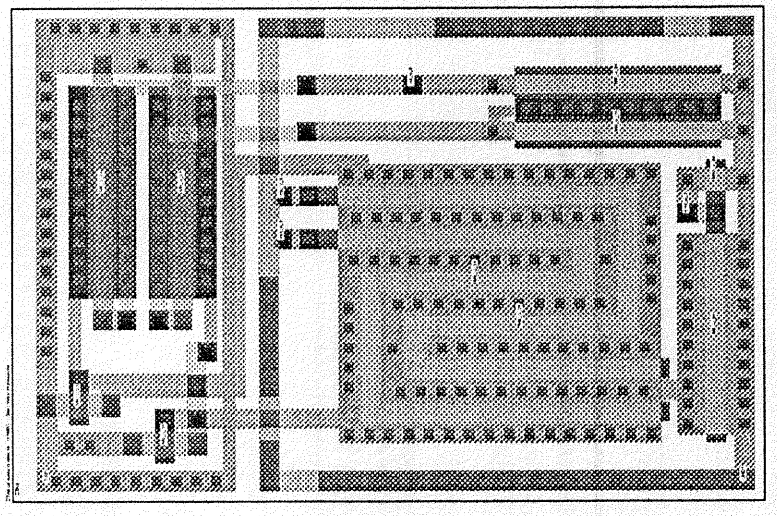


Figura 3-2: Diagrama de máscaras de un bloque diseñado *totalmente a medida*. (Realizado en el GTE por J. Chávez)

3.2.2 Diseño *parcialmente a medida*

Para describir la idea de un circuito integrado diseñado con el estilo *parcialmente a medida* es preciso introducir el concepto de diseño jerárquico. Se agrupan las unidades funcionales en conjuntos superiores o bloques. De esta forma es posible crear nuevas agrupaciones que a su vez relacionan estos bloques con otros de su mismo nivel o inferior. Las unidades funcionales básicas se agrupan en una librería de celdas.

El diseñador se preocupa, exclusivamente, del correcto ensamblaje de los módulos conforme a lo especificado funcionalmente. El resultado final, una vez expandida la red, es un conjunto de referencias entre unidades funcionales y de conexiones entre ellas, llamada *referencial*.

Celdas Estándar

Los elementos de la librería contienen la *disposición* de sus celdas. Por norma general se les dota de una dimensión que es común a todos los elementos: la altura. El ancho depende del tipo de celda; por ejemplo, un biestable JK ocupa obviamente mucho más área que un simple inversor y, para la misma altura, la celda del biestable será más ancha.

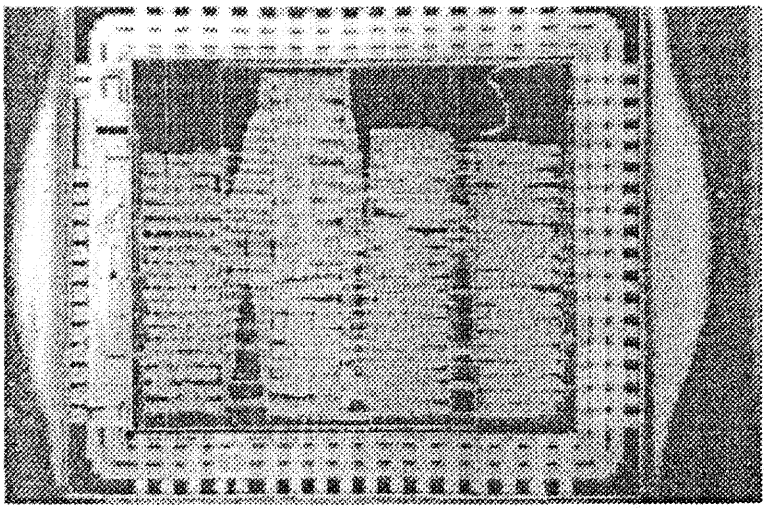


Figura 3-3: Circuito mixto basado en *celdas estándar*. Diseñado por el GTE y grupo de empresas INNOVA, 1992

El estilo de *disposición* es más regular que en el caso *totalmente a medida*. Las celdas se agrupan en *filas* horizontales, quedando unos *canales de conexión* entre ellas. El espacio dedicado al *conexionado* puede ser variable de un canal a otro. Buscaremos aquella disposición de las celdas que haga mínima el área de conexionado. La figura 3-3 muestra un circuito industrial con este estilo de diseño. En este caso, el circuitos contiene partes digitales y analógicas que se agrupan en distintas filas (las partes centrales del circuito contienen zonas en las que se observa una mayor regularidad en la *disposición*, coincidiendo con las partes digitales).

Las conexiones entre celdas que no pertenecen a filas ubicadas consecutivamente (o a la misma fila) pueden realizarse a través de canales verticales destinados

al efecto, denominados *corredores* (*highways*), aunque esta técnica de diseño no se usa hoy día por su mal aprovechamiento del área de silicio. Una técnica más empleada, usa un tipo de celda especial llamada *celda de paso* (*feedthrough*) que hace posible el cruce de la conexión a través de una fila. Dado que su presencia supone un ligero incremento en el área de la *colocación*, minimizar el número de *celdas de paso*, necesarias es otro criterio que ha de tenerse en cuenta a la hora de formular el problema de optimización. Tecnologías de fabricación más modernas permiten el empleo de múltiples capas de metalización, haciendo innecesario el uso de las *celdas de paso*.

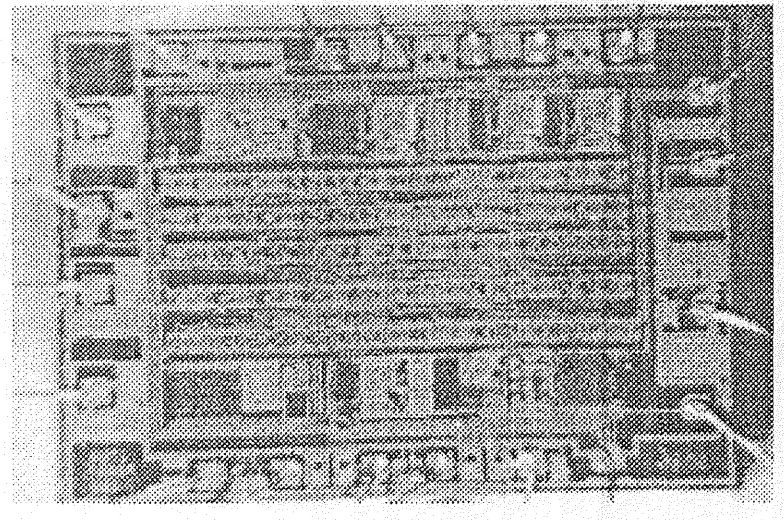


Figura 3-4: Circuito mixto basado en *celdas estándar*. Diseñado por el GTE y la empresa Landis & Gyr española, 1992

Dado que la oblea de semiconductor no tiene elementos predefinidos, es posible emplear este método con celdas digitales y analógicas en el mismo circuito.

Macroceldas

Es un caso especial dentro del estilo de *celdas estándar* donde se admiten celdas con diferente tamaño y alturas. Un ejemplo típico es el caso de las celdas digitales parametrizables (RAM, ROM, operadores aritméticos,...). El tratamiento

de este caso es ligeramente diferente, y el problema de la colocación se complica.

Matriz de Puertas

Este es un estilo de diseño que se aplica exclusivamente a diseños digitales, aunque recientemente han aparecido estructuras de un estilo parecido en el campo analógico. Se predifunden en la oblea de silicio filas formadas por pares de transistores p–MOS y n–MOS. Mediante conexiones metálicas de estos transistores se consiguen formar bloques funcionales equivalentes a las celdas de un estilo de *celdas estándar*. El proceso de personalización de la oblea de silicio se reduce a las últimas máscaras del proceso tecnológico que realizan las conexiones, tanto dentro de las filas (para formar las celdas) como en el espacio entre filas (para realizar el conexionado de las celdas).

En este estilo de diseño, el área de *conexionado* está fijada previamente, y el objetivo del problema de optimización es conseguir introducir el conexionado del circuito en el espacio reservado para ello. No obstante, aunque de una ,manera indirecta, el objetivo final es reducir la longitud total de las conexiones, pues de esta manera aumenta la probabilidad de que el conexionado quepa en el área reservada para ello.

En la figura 3-5 se presenta una *disposición* de un circuito industrial en estilo *matriz de puertas*. Se puede observar la disposición regular de los canales de conexión.

Mar de Puertas

Es una generalización del estilo *matriz de puertas*. En él la oblea de silicio está totalmente compuesta por transistores, sin ningún espacio específicamente reservado para el *conexionado*. Cuando se establecen la conexiones se hace a costa de inutilizar los transistores que quedan debajo. El problema de la *disposición* es, en este caso, de una mayor complejidad, al aumentar los grados de libertad del problema. En general, se adoptan algoritmos que introducen algún tipo de regularidad en la estructura, para hacerlos computacionalmente eficiente.

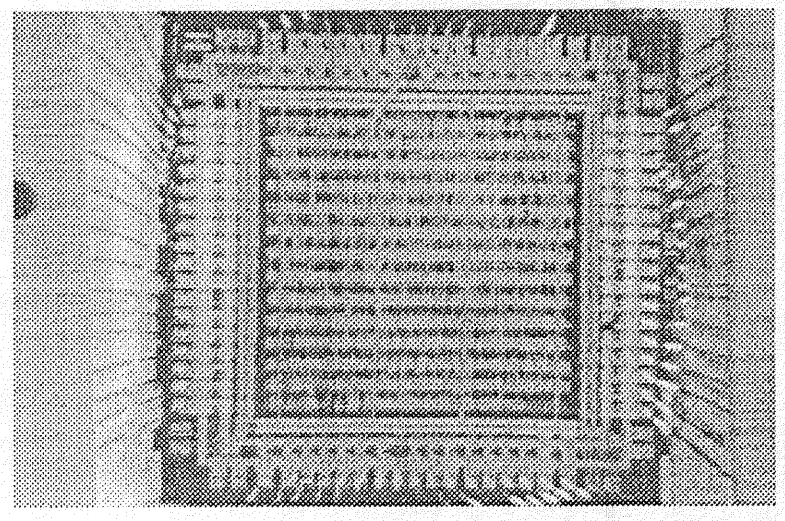


Figura 3-5: Circuito en el estilo matriz de puertas. Diseñado por el GTE y la empresa GHERSA, 1991

La figura 3-6 presenta la *disposición* de un circuito industrial en el estilo *mar de puertas*. En él puede observarse la estructura regular de transistores y líneas de alimentación que forman la base del diseño y la estructura irregular de su conexionado. En particular, no se observa la disposición de celdas en filas, ni la existencia de los *canales de conexión*, que son típicos de los estilos *celdas estándar* y *matriz de puertas*.

3.3 Definición del Problema de la *Colocación*

El problema de la *colocación* es un problema de optimización con restricciones. Para ello, se define una función de costes en la que intervienen los siguientes términos:

1. Área: su influencia es fundamental en el coste económico del circuito integrado.

2. Longitudes máximas de las conexiones: influye en los retrasos de las señales y en el propio área del circuito.

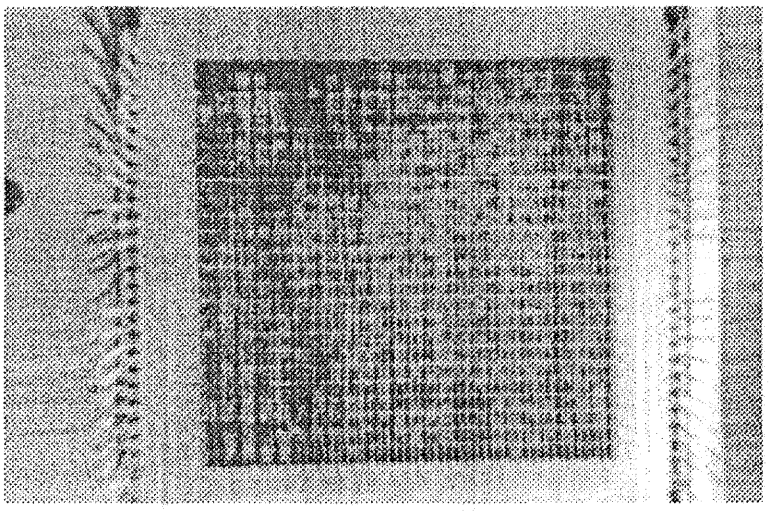


Figura 3-6: Circuito en el estilo mar de puertas. Diseñado por el GTE y la empresa MAC S.A., 1993

3. Número de Cruces entre Conexiones: afectan a la robustez eléctrica del circuito.

4. Número de vías o contactos metal-semiconductor: estos contactos introducen inestabilidades eléctricas.

5. Número de cambios de dirección de las conexiones: los cambios de dirección son zonas de mayor resistencia.

El problema de la *colocación* de *celdas* minimiza, además, los cruces de *filas*, si estos se producen a través de *celdas de paso* (*feedtroughs*).

3.3.1 Formulación del Problema de la *Colocación*

A continuación se presenta una formulación del problema de la *colocación* que recoge las ideas principales que aparecen en el caso del diseño con *celdas estándar* y *matrices de puertas*. Esta formulación del problema realiza algunas simplificaciones cuyo objetivo es el de intentar identificar el problema de la *colocación* con otros problemas de optimización con restricciones existentes en la bibliografía [71].

Definición y Complejidad

Como se ha comentado en apartados anteriores, el tratamiento del problema de la *colocación* de celdas, tanto en el estilo de *celdas estándar* o de *matrices de puertas*, no es un problema independiente de la etapa posterior. La formulación del problema lleva inherente la presencia del coste de las conexiones, aunque sea mediante una estimación poco precisa de la misma. En cada solución del problema no se asignan posiciones detalladas a las líneas de conexión, sino que se estima una distancia de la misma en función de la posición de cada celda. A veces la estimación de esa distancia se realiza mediante una asignación de conexiones a *canales*. En este caso, el propio algoritmo de *colocación* lleva incluido una primera etapa de *conexionado*,

Desde un punto de vista formal, el problema de la *colocación* se plantea de la forma siguiente:

Problema de la *colocación* rectangular no restringida [71]

- *Elementos:* Un hipergrafo $G = (V, E)$ con una función de coste $l(e) \in R^+$, para cada $e \in E$, $|V| = n$. (En el caso de *matrices de puertas* y *celdas estándar* se consideran, ademas, dos constantes $r, s \in N$, $r, s \geq n$ indicando el tamaño de la matriz.)

- *Configuraciones:* Todas las posibles *colocaciones* p, esto es, todas las posibles combinaciones que representan la posibles distribuciones de las celdas a lo largo y ancho de la oblea. (En el caso de *matrices de puertas* y *celdas estándar* es una relación inyectiva $p : V \rightarrow [1, r] \times [1, s]$ entre la posición de las celdas y los mapas del circuito que representan la *rejilla*. Se admite que una misma celda pueda ocupar varias posiciones contiguas de la *rejilla*).

- *Soluciones:* Todas las configuraciones posibles.

- *Minimizar:* Una función de costes suma de todos los elementos que pueden influir en el área total. (En el caso de las *matrices de puertas* se busca que el área de las conexiones sea menor que el fijado por la matriz).

3.3.2 Elección de la Función de Costes

A lo largo de esta sección se dará una definición más precisa de la función de costes y de sus componentes.

Función de Costes Basada en la Longitud de la Conexión

Por simplicidad, las funciones de coste de un hiper–enlace están basadas en la consideración que la celda colocada realiza todas la conexiones desde su centro. Es posible dar una estimación de la longitud de la red mediante el *semiperímetro del menor rectángulo* que la contiene, es decir:

$$c_1(e,p) = \max_{v,w \in e} |p(v)_x - p(w)_x| + |p(v)_y - p(w)_y| \tag{3.1}$$

Donde $p(v) = (p(v)_x, p(v)_y)$ indica la posición de la celda v en el área, dada por sus coordenadas cartesianas y $c_1(e,p)$ hace referencia a la norma L_1 .

Una mejor estimación se puede obtener calculando el árbol de Steiner de cada conexión. No obstante, a pesar de la complejidad de esta aproximación, tampoco es un método muy seguro, ya que, por un lado las conexiones se realizan en puntos que se sitúan en la periferia, y no en el centro, de cada celda. Por otro, no se tienen en cuenta la interrelación con otras conexiones. Por ejemplo, el conexionado a través de una zona muy congestionado dará lugar a longitudes finales de conexión que serán elevadas. La medida del coste mediante la determinación de la longitud del árbol de Steiner es, obviamente, una cota inferior de la longitud de las conexiones que se va a resultar finalmente tras la etapa de *conexionado en detalle* (*detailed routing*). En su contra tiene el ser un método complejo de estimar el coste de una conexión.

También se pueden emplear estimaciones del coste basadas en una estructura en estrella de la conexión, donde el centro de la estrella se fija en su centro de gravedad, o mediante el cálculo de la longitud del *árbol rectilíneo de mínima cobertura* (*minimum spanning tree*).

Dado que ninguna de las aproximaciones anteriores da una estimación precisa del coste final del conexionado, normalmente se escoge la más simple de calcular, es decir, el *semiperímetro del menor rectángulo* que engloba a todos los tramos de la

conexión. Así, para evaluar el coste total de la red se puede utilizar una expresión simple:

$$c_1(p) = \sum_{e \in E} l(e) c_1(e, p) \tag{3.2}$$

donde $l(e)$ representa el coste de la conexión e, que en la mayor parte de los casos se hace igual a la unidad y c_1 se define en (3.1).

El problema de colocación que utiliza (3.2) como función de costes se conoce como *colocación óptima rectangular*. En el caso unidimensional se denomina *colocación óptima lineal*.

Este tipo de funciones de coste plantea dos problemas:

1. En la práctica, no todas las conexiones tiene una anchura uniforme.

2. El minimizar la longitud de las redes no necesariamente tiene por qué minimizar el área final, ya que no se maximiza la trazabilidad de la red (es decir, la facilidad para poder ser colocada geométricamente). En este sentido puede ser más conveniente obtener una distribución uniforme de las conexiones a lo largo del circuito.

Algunas aproximaciones dan prioridad a la eliminación de conexiones muy largas, lo que se traduce en uniformizar los retrasos debidos al *conexionado*. Para conseguir esto, se modifica la función de costes en la forma:

$$c_k(p) = \sum_{e \in E} l(e) c_k(e, p) \tag{3.3}$$

donde $c_k(p)$ vale:

$$c_k(e, p) = \max_{v,w \in e} |p(v)_x - p(w)_x|^k + |p(v)_y - p(w)_y|^k \tag{3.4}$$

Esta función de costes es no-lineal. El caso más habitual se obtiene para $k = 2$ y, en este caso, el problema se conoce como *colocación rectangular cuadrática no restringida*; en el caso unidimensional, se denomina *colocación lineal cuadrática no restringida*. En caso particular de que G sea un grafo, el problema de la *colocación rectangular cuadrática no restringida* se reduce a uno de *asignación cuadrática*, que es un problema muy estudiado en la literatura.

Función de Costes Basadas en Cortes

Una forma de mejorar la distribución de los enlaces a lo largo de la oblea es la de utilizar funciones de coste basadas en el número de enlaces que atraviesa un determinado corte de la misma. En este caso el coste es una medida de la *congestión* de los enlaces. Se consigue una mejor distribución considerando como función de costes, no la suma de los cortes, sino el máximo de todos los cortes, es decir:

$$c_{MC}(p) = \max_{h \in H} c(h) \tag{3.5}$$

Donde h representa todos los posibles cortes en el área de la *disposición*, y $c(h)$ el tamaño del corte. Esta función de costes se conoce como *densidad*. El problema de la *colocación*, planteado con esta función de costes se conoce como *colocación rectangular del mínimo corte*, y su problema unidimensional *colocación lineal del mínimo corte*. Ambos son problemas NP–duros.

3.4 Heurísticas de búsqueda dirigida

3.4.1 Heurísticas del corte mínimo

Estas técnicas tratan de resolver el problema de la *colocación* minimizando la conexión entre subcircuitos. Aunque no está demostrado que haciendo esto se minimice el área total, el sentido común apunta a que, al menos, se camina en la dirección adecuada. De hecho, los resultados obtenidos con esta aproximación así lo prueban.

El problema de la *colocación* puede ser resuelto mediante partición recursiva. El proceso genera un árbol de particiones siguiendo un modelo descendente. Algunos de los algoritmos más antiguos de *colocación* emplean estas técnicas. La idea original fue propuesta por Breuer en [14] y [13]. Para tener en cuenta la naturaleza bidimensional del problema, el autor propone escoger direcciones alternadas de corte (figura 3-7).

En [30] se introduce el concepto de *propagación de terminales*. Con este mecanismo, es posible tener en cuenta (al menos, parcialmente) la sitiuación del

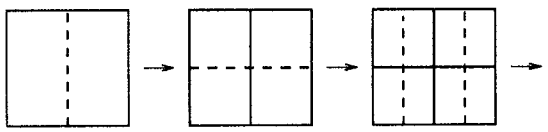


Figura 3-7: Secuencia de Particionamiento

resto de las celdas del circuito cuando se realiza la partición de un bloque. En [110], [111], [118], se emplea como algoritmo de corte un algoritmo de división en cuatro bloques (o *cuadrisección*), lo que permite acercar la solución a la naturaleza bidimensional del problema. En [115] y [121] se incluyen en el algoritmo de corte, ciertas restricciones en la longitud máxima de conexiones críticas.

En [49] se emplea un algoritmo de corte mínimo para formar una solución inicial a un proceso de intercambio de celdas. Por ello se escoge la celda peor posicionada del circuito y se intercambia la posición con otra celda de un entorno cercano a la posición óptima.

3.4.2 Técnicas que minimizan la longitud de las conexiones

La mayor parte de las aplicaciones de interés práctico que tratan de minimizar la longitud de las conexiones, emplean una función cuadrática para evaluar el cost. Hay varias razones para ello:

1. La función cuadrática, en comparación con la lineal, penaliza las distancias muy largas.

2. Es posible establecer una analogía entre la energía de un sistema de muelles [89] o de un circuito resistivo [24] y el coste de una *colocación* que emplea una función de costes cuadrática. Aunque no parece que esta analogía tenga alguna utilidad práctica.

3. Escogiendo adecuadamente las restricciones del problema, se puede conseguir que la función de coste sea convexa y tenga un único mínimo global. De

esta manera se pueden emplear técnicas de cálculo de autovectores para la resolución del problema.

Resolución del problema *relajado*

El problema se resuelve en dos partes. En una primera fase se eliminan las restricciones que hacen que el problema sea especialmente duro, es decir, se ignora el hecho de que las celdas deben estar alineados en una *fila* y no solapar entre ellas. Por ello, se admite cualquier posición de la celda en el rectángulo de la oblea.

La función objetivo en el caso unidimensional y para un coste cuadrático con la distancia, puede formularse como [71]:

$$f(x) = p^T \ C \ p \tag{3.6}$$

donde C es una matriz de costes semidefinida positiva y p es el vector de posición de las celdas.

Para normalizar el problema, se introduce la restricción esférica

$$p \ p^T \ = 1 \tag{3.7}$$

y para evitar la solución trivial (todos los autovalores 0, es decir todas las celdas apiladas unas encima de otras en el origen) se introduce otra restricción

$$\sum_{i=1}^{n} p_i = 0 \tag{3.8}$$

Ninguna de estas restricciones supone una restricción física real al problema de la *colocación*

La solución a este problema se encuentra haciendo p igual al autovector correspondiente al mínimo autovalor no nulo de la matriz C [47].

El caso bidimensional puede tratarse como la combinación de dos problemas unidimensionales distintos. Si no se introduce ninguna restricción adicional, ambos problemas unidimensionales serán idénticos y proporcionarán la misma solución. Es decir, todas las celdas se encontrarán agrupadas en unaa diagonal del rectángulo. Para evitar este problema se introduce una restricción adicional, que asegura la no–colinealidad de la solución

$$p_x^T \ p_y = 0 \tag{3.9}$$

Ahora el mínimo para la segunda dimensión se produce en el autovector correspondiente al segundo autovalor de menor valor, no nulo.

Blanks [7] extiende el problema al caso en que existan celdas con posiciones fijas. En [59], los autores emplean la misma aproximación en el primer paso de un proceso recursivo. En cada paso, el circuito se va dividiendo en bloques de menor tamaño. En los pasos siguientes, la restricción (3.8) se transforma en la expresión siguiente

para $j = 1, .., numero\_de\_bloques$

$$\sum_{celdas\_en\_bloque\_j} tamano\_celda\_i \times p_i =$$

$$centro\_gravedad\_bloque\_j \times \sum_{celdas\_en\_bloque\_j} tamano\_celda\_i \qquad (3.10)$$

Eliminando el solape entre celdas

La solución del problema *relajado* no asigna celdas a posiciones físicas reales, ni elimina el solape entre ellas. Para conseguir esto, se emplea normalmente un algoritmo de cortes recursivo que emplea como partida la solución relajada del problema. Este método es el que se sigue en [85], [116], [124] y [59].

3.4.3 Otras técnicas

Entre otras aproximaciones al problema de la *colocación*, destacamos los algoritmos conocidos como *redes como puntos* (*nets–as–points*), que tratan de colocar las conexiones, en vez de las celdas. Para ello, construyen una red dual de la original en la que cada conexión es representada por una celda y viceversa [127].

3.5 Heurísticas de búsqueda aleatoria

La aparición de las técnicas de *enfriamiento simulado* (*simulated annealing*), introducidas por Kirpatrick, Gelatt y Vecchi en [57] da lugar a la aplicación de metodos de búsqueda aleatoria a la resolución de problemas de optimización. De

especial interés son las aplicaciones al problema de la *colocación* de celdas en un circuito integrado.

Ya el el citado artículo [57] se describe la aplicación de esta técnica a la colocación de un conjunto de bloques, componentes de un circuito de mayor dimensión. No mucho más tarde [103], C. Sechen y A. Sangiovanni-Vincentelli presentan *TimberWolf*, un paquete de programas para la *colocación* y el *conexionado* de *celdas estándar* que emplea *enfriamiento simulado*. Este paquete de rutinas ha sufrido desde entonces sucesivas mejoras y ampliaciones [105], [100] y [101].

TimberWolfSC fue puesto a libre disposición de las industrias y universidades, lo que unido a los excelentes resultados que con él se obtienen, le ha convertido en un patrón de referencia para medir la calidad de la *disposición* de un circuito integrado.

3.5.1 Heurística del Enfriamiento Simulado

Este algoritmo fue propuesto por Kirpatrick, Gelatt y Vecchi [57] en 1983, basándose en estudios del método Monte Carlo propuesto por Metrópolis en el Año 1969. Posteriores estudios teóricos [37], [78] demostraron que el algoritmo es capaz de encontrar el óptimo global de un problema de optimización con probabilidad 1, siempre que durante el proceso de enfriamiento se cumplan una serie de condiciones. En [57] se propone un paralelismo con el enfriamiento cuasiestático (en el que se produce una sucesión de estados de equilibrio termodinámico) de un sólido, desde un estado líquido o gaseoso hasta su estado de mínima energía reticular. En ese estado final la estructura del sólido es absolutamente ordenada. La idea propuesta allí es la de aplicar un procedimiento de enfriamiento equivalente al problema de la *colocación* de celdas en un circuito integrado (y a otros problemas de igual complejidad). El algoritmo emplea un parámetro de control llamado *Temperatura* que será responsable del proceso de optimización. A cada temperatura, para que el proceso sea cuasiestático, debe alcanzarse un equilibrio equivalente al equilibrio termodinámico del sólido. Para evaluar la energía del sistema se utiliza la propia función de costes, en la que se introducen términos referentes a la posición de las celdas, al número

de *celda de paso* y a la longitud de las conexiones. A cada temperatura se produce una generación de cambios aleatorios o perturbaciones en el sistema, que simula el movimiento caótico entre los átomos de la red a causa de su energía térmica.

La condición de enfriamiento cuasiestático que requieren los estudios teóricos de convergencia del algoritmo, en la práctica exige:

1. Un número muy elevado de movimientos en cada temperatura para garantizar que se ha alcanzado el equilibrio y
2. Una disminución muy suave de la misma.

Por tanto, para conseguir buenos resultados es necesario un empleo masivo de recursos de cálculo y un elevado tiempo de optimización. En [65] y [17] se propone el empleo de un sistema multiprocesador para resolver el problema de la *colocación* en circuitos con un número elevado de celdas. Otros investigadores propusieron políticas de enfriamiento que permiten acelerar el proceso de convergencia. Entre ellas destacan las propuestas de por Huang, Romeo, y Sangiovanni–Vincentelli en [51] y la propuesta por Lam y Delosme en [68].

A fin de concretar ideas describimos a continuación la realización propuesta en [51], y que ha sido utilizada en esta tesis como referencia. También se describe en cada caso, la elección de los parámetros que utiliza TimberWolfSC–4.2.

3.5.2 Realización propuesta por Huang, Romeo, y Sangiovanni–Vincentelli

El algoritmo propuesto por estos autores [51] consta de cinco elementos:

1. Obtención de la temperatura inicial.
2. Criterio de descenso de temperatura.
3. Condición de equilibrio de temperatura.
4. Criterio de aceptación de movimientos.
5. Condición de congelación o criterio de parada del algoritmo.

```
T=TEMPERATURA INICIAL();
WHILE( Condicion de Congelacion=FALSE) {
  WHILE( Condicion de Equilibrio=FALSE ){
    B= MUEVE(A);
    IF( ACEPTAR(A;T)=TRUE ){ A=B } }
  T=DESCIENDE TEMPERATURA();}
```

Figura 3-8: Algoritmo Básico de Enfriamiento Simulado

El algoritmo en su esquema más general se presenta en la figura 3-8.

Dentro del esquema básico del algoritmo, la condición de aceptación de movimientos tiene especial relevancia, ya que es la responsable de la capacidad del algoritmo de salir de un mínimo local.

En lo que sigue se presentan cada uno de los elementos del Algoritmo.

Cálculo de la Temperatura Inicial (T_∞)

En la comparación con el ejemplo físico que da lugar al algoritmo, se buscaría un valor de la temperatura tal que cualquier movimiento de las partículas se acepte. Por ello, se realiza una primera exploración del espacio de configuraciones aceptando todos los movimientos posibles y se calcula la desviación típica σ de la función de coste. Como temperatura inicial se adopta un valor $T_\infty >> \sigma$ tal como se propuso en [78]. De esta forma se asegura la aceptación de todos los posibles movimientos del sistema para T_∞.

Desde un punto de vista práctico se establece como criterio, una temperatura inicial de:

$$T_\infty = 20 \times \sigma \tag{3.11}$$

C. Sechen, en TimberWolfSC4.2, fija la temperatura inicial de manera ar-

bitraria en 500.

Criterio de Descenso de la Temperatura

Es el criterio básico para garantizar una convergencia al óptimo global. En el apéndice B de esta tesis se resume una condición suficiente del descenso de la temperatura para garantizar que el óptimo global se alcanza con probabilidad 1. El problema es que esta condición choca frontalmente con el carácter de los problemas a resolver. Suelen ser problemas de gran dimensión, con espacios de soluciones muy grandes. Se exige, pues, una búsqueda ágil que permita encontrar buenas soluciones lo antes posible.

Si se produce un descenso muy lento, el algoritmo, sería muy poco ágil, pero se obtendrían unos resultados bastante buenos. En la literatura se pueden encontrar diferentes aproximaciones que utilizan la desviación típica de la función de costes como medio para determinar el decremento de la temperatura. En [51] se emplea la curva de enfriamiento, es decir, la curva de coste medio frente a $\log T$ para generar ese decremento. El objetivo es buscar una función de descenso de temperatura que haga que la media de la función de costes $< c >$ decrezca con el logaritmo de una manera uniforme. Para ello se obliga a que la pendiente de la curva valga:

$$\frac{d < c >}{d(\ln T)} = T \times \frac{d < c >}{dT} = Cte \tag{3.12}$$

De consideraciones aportadas por la mecánica estadística:

$$\frac{d < c >}{dT} = \frac{\sigma^2}{T^2} \tag{3.13}$$

De aquí se obtiene la relación:

$$\frac{d < c >}{d \ln T} = \frac{\sigma^2}{T} \tag{3.14}$$

que, desde un punto de vista práctico, se reduce a:

$$\frac{\Delta < c >}{\ln T - \ln T'} = \frac{\sigma^2}{T} \tag{3.15}$$

es decir:

$$T' = T \times e^{\frac{\Delta < c > T}{\sigma^2}} \tag{3.16}$$

Para conseguir cuasi–equilibrio se fuerza a que el incremento del coste medio sea menor que la desviación típica, es decir $\Delta < c >= -\lambda\sigma$, con $\lambda \leq 1$, y:

$$T' = T \times e^{-\frac{\lambda T}{\sigma}} \tag{3.17}$$

Un valor típico es $\lambda = 0.7$. Los autores proponen que, en cualquier caso, T'/T no sea inferior a 0.5.

En TimberWolfSC–4.2 se toma una expresión simplificada, que realiza una aproximación escalonada a la función exponencial, de manera que se desciende según: $T' = 0.9825 \times T$, cuando la temperatura en superior a 90 y $T' = 0.915 \times T$ cuando es inferior a 90 y superior a 20. Cuando T en menor que 20 se desciende mediante $T' = 0.7 \times T$. Cuando se alcanza $T \leq 10$ realizan tres iteraciones más con $T' = 0.1 \times T$, para, seguidamente, concluir el proceso con un *conexionado de detalle.*

Condición de Equilibrio Local

Probablemente es la condición más compleja de modelar. La idea básica consiste en encontrar una situación estacionaria a una *Temperatura* dada, en que el número de movimientos aceptados no aumente ni disminuya el valor de la energía global del sistema. Es decir:

$$E(c,T) = \int_{-\infty}^{+\infty} \Delta c(T) \times N(c,T) \times dc = 0 \tag{3.18}$$

Esta situación en términos de nuestro problema es el valor de la esperanza matemática de Δc:

$$E(c,T) = \int_{-\infty}^{+\infty} \Delta c(T) \times P_{acept}(c,T) \times dc = 0 \tag{3.19}$$

Donde P_{acept} y Δc dependen de la temperatura. Este detalle es de especial importancia para nuestro posterior desarrollo.

En la literatura este problema se resuelve aceptando que el equilibrio se ha alcanzado cuando se ha realizado un cierto número mínimo de movimientos aceptados. En el desarrollo del programa TimberWolf este número de intentos es función del tamaño del problema, y depende de si el usuario desea una optimización rápida o lenta.

La propuesta de Huang y col. se basa en que, una vez alcanzado el equilibrio, la razón entre el número de estados aceptados con un coste c dentro de un cierto rango δ de la media $< c >$, y el número total de estados aceptados alcanzará un valor estable χ. El valor de χ depende de la naturaleza la distribución de coste y del parámetro δ. A elevadas temperaturas, en que la distribución de costes es normal, $\chi = errf(\delta/\sigma)$, donde $errf$ es la función de error y σ la desviación típica de la distribución de costes a esa temperatura. Un valor típico para δ es $0.5 \times \sigma$, entonces $\chi = errf(0.5) = 0.38$. Basándose en esta estimación se definen dos contadores: el *contador interior* y el *contador de máximo alcance*. El primero cuenta el número de estados aceptados dentro del intervalo $(< c > -0.5 \times \delta, c > +0.5 \times \delta)$, y el segundo cuenta el número de estados aceptados fuera del intervalo. Si el *contador interior* alcanza el valor $0.38 \times 3 \times$ *numero de celdas* antes de que el *contador de máximo alcance* llegue a $0.62 \times 3 \times$ *numero de celdas* entonces se considera que se ha alcanzado el equilibrio. En caso contrario, se inicializan ambos contadores a cero y se comienza de nuevo el proceso a esa misma temperatura.

Este criterio se conoce como el del *contador Interno*. Representa, en realidad, un ajuste dinámico de la longitud de la cadena de Markov.

En cualquier caso el número de movimientos generados se acota superiormente para evitar tiempos excesivos de cálculo, sobre todo, a bajas temperaturas.

Criterio de Aceptación de Movimientos

La aceptación de movimientos es el mecanismo básico de salida de óptimos locales y responde a la propuesta inicial de Kirkpatrick [57]. Sea Δc el incremento de costes debido a un movimiento aleatorio en la posición actual, entonces:

1. Si Δc es negativo, el nuevo movimiento se acepta incondicionalmente.

2. Si Δc es positivo, el nuevo movimiento se acepta con una cierta probabilidad, que depende de la temperatura.

   El movimiento se acepta si:

$$RANDOM[0,1] > e^{-\frac{\Delta c}{T}} \tag{3.20}$$

Donde $RANDOM[0,1]$ es una función que genera aleatoriamente un número real en el intervalo $[0,1]$. La función exponencial es siempre positiva, y, su forma permite una alta aceptación a altas temperaturas y baja aceptación cuando T se acerca a 0.

Condición de Congelación

Es el criterio de parada del algoritmo. Usualmente se utiliza un criterio de imposibilidad de mejora de la solución, es decir, si la media de la función de costes no varía en las dos o tres últimas iteraciones, puede considerarse que el algoritmo no ha mejorado y ha encontrado su óptimo.

En la versión de TimberWolfSC-4.2 aquí estudiada el criterio de parada es, simplemente, el alcance de un valor determinado de la temperatura, fijado en 0.01, como resulta del mecanismo de descenso de temperatura descrito en 3.5.2.

3.5.3 Otras heurísticas de búsqueda aleatoria

Una de las críticas más repetidas sobre el comportamiento de este algoritmo es que requiere un tiempo de computación excesivo. Se han propuesto otras heurísticas de búsqueda aleatoria que buscan un proceso de optimización más rápido sin perder calidad en las soluciones finales.

Heurística de Evolución Simulada

En [60] se presenta una heurística conocida como Evolución Simulada. Esta es una técnica iterativa en que, a partir de una solución inicial, se van alcanzando soluciones mejores mediante una búsqueda aleatoria. La posición de cada una de las celdas es evaluada respecto de su posición óptima (considerando como tal el centro de

gravedad de sus conexiones). Aquellas celdas con peor evolución son aleatoriamente escogidas para cambiar de posición.

Heurística de Evolución Estocástica

Un ejemplo de otro tipo de heurísticas basadas en búsqueda aleatoria es la llamada "Evolución Estocástica" introducida por Saab & Rao [96]. La idea que se baraja bajo esta realización es que durante el recorrido por el espacio de las configuraciones S es probable que se pase por estados muy próximos al óptimo global del problema. Por ello diseñan un algoritmo de búsqueda dotado de un elemento que permita "escalar" a costes mayores cuando no se consigue una mejora de la solución. Por el contrario, el espacio de búsqueda se reduce si se logran encontrar nuevos mínimos. La ventaja de este algoritmo es que es más general, independiente del problema que se desea optimizar, ya que no hay parámetros de control.

Heurística de Apagado Simulado

La heurística de Apagado Simulado (o *Simulated Quenching*) es recogida por Ingber [52] en sus estudios sobre el algoritmo de *Enfriamiento Simulado* y aparece en esta tesis como un ejemplo de la relajación de la condición de descenso de la temperatura buscando una evolución más rápida del algoritmo. De hecho al aproximación de Sechen en el desarrollo del TimberWolf es uno de los casos que se plantean en el mencionado trabajo.

3.6 Algoritmos basados en Métodos de Inteligencia Artificial

En esta sección se mencionan otras técnicas de optimización utilizadas en otros tipos de problemas que se han intentado trasladar al problema de la *colocación de celdas*.

3.6.1 Algoritmos Basados en Redes Neuronales

Bajo un conjunto de simplificaciones, es posible reducir el problema de la *colocación* de celdas al problema de una optimización con retricciones como el resuelto por una red de Hopfield [50], [114]. En [58] se emplea la red de Hopfield para resolver de una forma iterativa el problema de la colocación de celdas.

Aunque el empleo de la red neuronal puede parecer una aproximación muy prometedora al problema. en la práctica no lo es tanto por una serie de razones:

1. Para que el problema pueda ser resuelto mediante una red de neuronas es necesario realizar un conjunto de simplificaciones poco realistas.

2. La soluciones obtenidas mediante una red de Hopfield son muy dependientes de la solución inicial, pues carece de un mecanismo de escape de mínimos locales.

3. La red de neuronas necesita de n^2 neuronas para resolver el problema de n celdas, el número de conexiones es, como mínimo, de $O(n^3)$. Esto reduce su campo de aplicación a redes muy pequeñas.

Más prometedor parece el uso de redes neuronales auto–organizantes [62], [63], como se demuestra en [56], [18] y [126].

3.6.2 Algoritmos Genéticos

Surgen de un nuevo símil, esta vez de tipo biológico. Se trata de algoritmos que, a patir de una población aleatoria de soluciones (*individuos*), y tras una inspección detallada de cada una de ellas, reconocen secciones de las mismas que son localmente óptimas, y a partir de una combinación de dos generan una tercera que preserva la bondad de las primeras, y es más parecida a la óptima.

La terminología de los algoritmos genéticos denomina *genes* a tales caracaterísticas, *cromosomas* a una solución compuesta de genes y *esquema* a una solución que está parcialmente compuesta de genes. Los operadores básicos de estos algoritmos son tres:

- *Cruce*: Es el operador que combina dos individuos para generar un tercero. Diversos métodos de cruces han sido utilizados en el problema de la *colocación* de celdas.

- *Mutación*: Este operador permite la salida de mínimos locales, generando nuevas características o genes, mediante cambios aleatorios en los individuos.

- *Inversión*: Consiste en modificar el individuo mediante un simple movimiento de una cadena de genes dentro del mismo.

Existen pocas aplicaciones de Algoritmos Genéticos al problema de la colocación [27], [106]. En ambos casos los autores argumentan que esta aproximación permite visitar un número de soluciones muy inferior al que visitan otros algoritmos de búsqueda aleatoria, como el *enfriamiento simulado*. Si bien esto es cierto, los tiempos de ejecución y los resultados obtenidos por estos algoritmos no mejoran los resultados de *TimberWolfSC*

Capítulo 4

Aportaciones al Problema de la Colocación

4.1 Introducción

En este capítulo se presenta una aproximación original al problema de la colocación de celdas empleando *enfriamiento simulado*. La idea que subyace en la aproximación propuesta, ya fue apuntada en el artículo original de Kirpatrick, Gelatt y Vecchi [57]. Estos autores comprobaron que el proceso de enfriamiento simulado aplicado a la colocación de un conjunto de bloques presenta dos fases bien definidas. En la primera, caracterizada por elevadas temperaturas, las celdas se agrupan en bloques. A esta fase se le denomina fase de *agrupación*. En la segunda fase, las celdas se reordenan dentro de cada bloque, hasta alcanzar la colocación final. Estas dos fases se ponen de manifiesto cuando, siguiendo el símil termodinámico, se presenta la evolución con la temperatura de la función *calor específico* $\frac{d<c>}{dT}$. La presencia de un valle acusado en esta curva indica un "cambio de fase" del sistema.

Con esta idea en mente pretendemos sustituir la primera fase de un algoritmo de enfriamiento simulado por un proceso de particionamiento siguiendo la heurística propuesta en la sección 5 del capítulo 2, que, como vimos, es muy eficiente. De esta manera esperamos eliminar (al menos en gran medida) el tiempo que requiere la fase de aproximación del algoritmo, reduciendo los tiempo finales

de optimización. Por otra parte, al partir de una buena solución inicial cabe esperar que los resultados finales sean iguales, o mejores que la solución del algoritmo estándar de enfriamiento simulado.

El capítulo sigue las siguientes directrices:

1. Se comprueba la existencia de las diferentes fases antes apuntadas con medidas reales sobre circuitos de prueba. Para ello se emplea un algoritmo de enfriamiento simulado con un buen control del proceso, siguiendo el escalado de temperaturas propuesto por Huang, Romeo y Sangiovanni–Vincentelli [51] y que fue detallado en el capítulo anterior.

2. En el siguiente apartado, se propone un método combinado que toma como solución inicial del problema del enfriamiento, el resultado de un particionamiento del circuito siguiendo la técnica heurística del apartado 2.5. El problema de la combinación de métodos consiste en determinar la temperatura inicial que permita conservar los beneficios del algoritmo de partición, sin sacrificar la fase final del proceso de enfriamiento que asegura la obtención de un buen mínimo. En este apartado se examina la influencia de una buena solución inicial en un proceso de enfriamiento simulado.

3. Posteriormente, se describen distintos intentos extraídos de la literatura, o propuestos por el autor, para determinar una temperatura inicial adecuada. El estudio se apoya en numerosas simulaciones de distintas circuitos con diferentes particiones iniciales y temperaturas de partida, que permiten obtener experimentalmente un rango de temperaturas adecuado para realizar la aproximación.

4. Se presenta el método finalmente propuesto para la determinación de la temperatura inicial y se justifica en base a la teoría de [51]. Se demuestra su validez mediante una comparación de los resultados con los valores experimentales.

5. Finalmente, el método combinado se compara con los resultados de TimberWolfSC–4.2 mostrando que se alcanzan casi siempre mejores resultados, en tiempos muy inferiores.

Todas las simulaciones tienen como base el paquete de rutinas TimberWolfSC-4.2, convenientemente adaptado a cada método. Este paquete de síntesis incorpora también algoritmos de *conexionado global* y *de detalle* que permiten observar la calidad de la solución final. De esta manera se miden resultados sobre *disposiciones* (*layouts*) finales de los circuitos que dan la calidad real de las soluciones obtenidas.

En este capítulo se han empleado los circuitos de prueba que se encuentran disponibles en la base de datos del Microelectronic Center of North Carolina (MCNC), cuyas características se detallan en el apéndice III. Como función de costes se ha utilizado la función de costes que emplea *TimberWolfSC* que tiene tres términos [101]:

- C_1 es una función que tiene en cuenta el coste debido al semiperímetro (X_s, Y_s) del rectángulo que engloba a cada una de las conexiones del circuito, incluyendo distintos costes para las pistas horizontales (H_w) y verticales (V_w).

$$C_1 = \sum_{n=1}^{N_n} [X_s(n) \times H_w(n) + Y_s(n) \times V_w(n)] \tag{4.1}$$

  N_n es el número total de conexiones.

- C_2 es una función que tiene en cuenta el solape entre celdas y que tiene importancia en los pasos intermedios del algoritmo.

$$C_2 = p_2 \sum_{i \neq j} [O_l(i,j)]^2 \tag{4.2}$$

  $O_l(i,j)$ es el solape entre las celdas i y j. p_2 es una constante de normalización.

- C_3 es una función que tiene en cuenta la diferencia entre las longitudes de las distintas filas.

$$C_3 = p_3 \sum_{r=1}^{N_r} |L_a(r) - L_d(r)| \tag{4.3}$$

  N_r es el número total de filas, $L_a(r)$ es la longitud actual de la fila y $L_d(r)$ es la longitud deseada. p_3 es una constante de normalización.

4.2 Evolución del Enfriamiento Simulado

La figura 4-1 presenta la evolución típica de un proceso de optimización utilizado el algoritmo de *enfriamiento simulado*. Representando la media de la función de costes frente al número de iteraciones es posible distinguir tres fases:

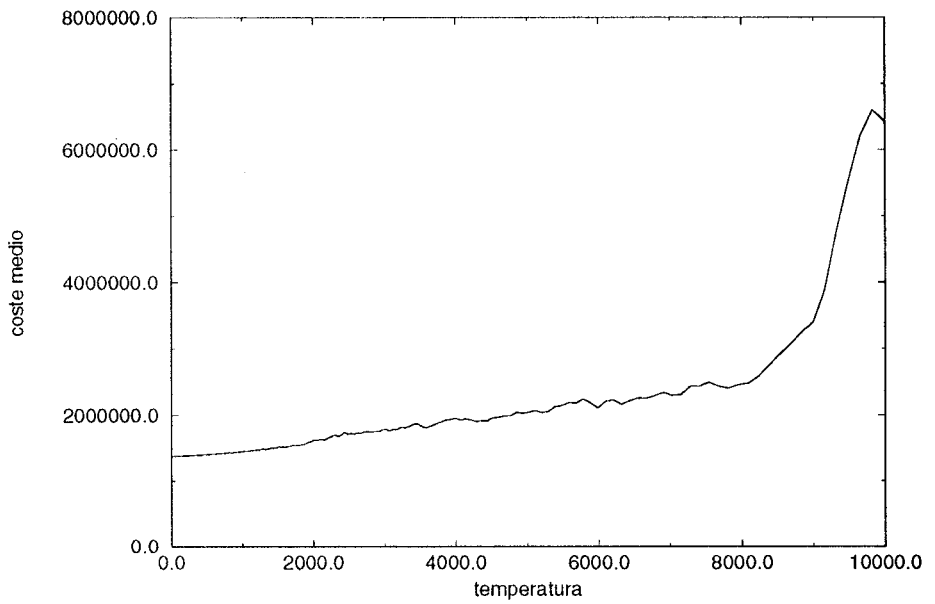


Figura 4-1: Curva típica de enfriamiento (media de costes frente a la temperatura) para el circuito Primary1

1. *Fase de aproximación.* En esta fase, la temperatura es suficientemente alta para controlar algún movimiento. En esta fase todos los movimientos son aceptados y el descenso de la temperatura es rápido. La media de la función de costes no varía mucho. Hay versiones del *enfriamiento simulado* que determinan la temperatura inicial del algoritmo en base a conideraciones relativas a la desviación típica de la distribución de coste en esta fase. TimberWolfSC fija el valor de esta temperatura en 500, de manera arbitraria.

2. *Fase intermedia.* En esta fase, el descenso del valor medio de la función objetivo es bastante acusado. La explicación es que los movimientos permitidos están cada vez más restringidos. Dentro de esta fase, a una temperatura intermedia, se produce un fenómeno de formación de *Agrupaciones* [57]. En la figura 4-2 se muestra una curva teórica que relaciona lo que Kirpatrick llama "calor específico" y la temperatura, asociada a un cambio de fase (de líquido a sólido) característico de un proceso de enfriamiento real [1], donde se aprecian las dos sub–fases.

   El "calor específico" tiene la expresión:

$$\frac{d < c >}{dT} \tag{4.4}$$

   A temperaturas altas se admiten movimientos de alto incremento de coste y por tanto entre largas distancias. A partir de cierta temperatura, que denotaremos por T_c, la optimización se realiza por zonas, es decir, los movimientos son de corta distancia y con pocas mejoras en la función de costes.

3. *Fase de parada.* Los movimientos admitidos son más bien cortos, es decir, intercambios entre celdas adyacentes, rotaciones, y poco más.

4.2.1 Influencia del escalado

Ya vimos en el capítulo anterior la importancia que tiene la elección de un método de enfriamiento ("escalado") que garantice las propiedades del algoritmo de enfriamiento simulado.

En el planteamiento teórico del capítulo anterior y otras publicaciones ya mencionadas, se determina que la elección de uno u otro tipo de escalado es fundamental para la consecución de un buen o mal resultado final. El escalado de Huang [51] reduce la temperatura del sistema lentamente y su propuesta de condición de equilibrio a cada una de las temperaturas, obliga a que el número de movimientos

[1] Esta curva fue introducida sobre una realización que resolvía una *colocación* de circuitos integrados discretos, en los que el número de conexiones es regular y en torno a 16

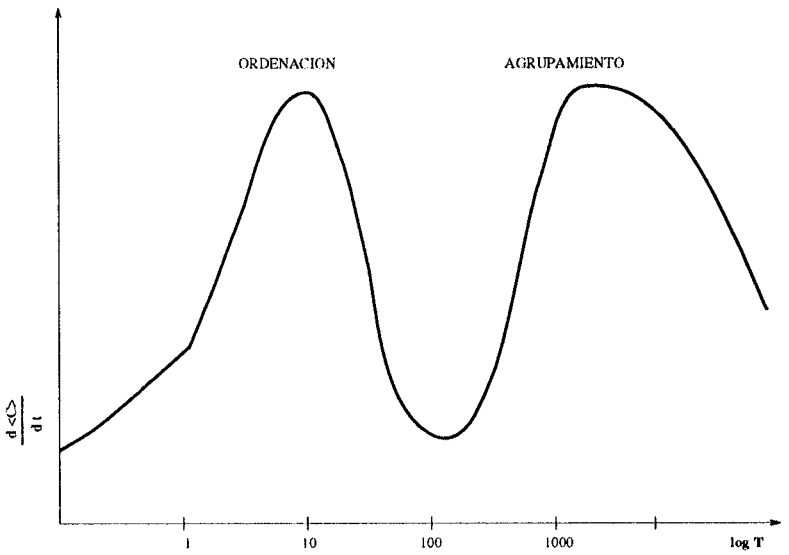


Figura 4-2: Curva de Calor Específico frente a la Temperatura

ensayados sea extraordinariamente grande. Por tanto, los tiempos de computación son también muy grandes (del orden de 20 horas para un circuito de unas 6000 celdas).

Las simplificaciones internas de *TimberWolfSC* reducen los tiempos de computación, si bien a costa de empobrecer la solución final (ver tabla 4.1).

El número de iteraciones que aparecen en las columnas de la tabla 4.1 es un valor meramente informativo, ya que debido al tipo de condición de equilibrio de cada caso, una iteración del escalado de Huang es normalmente mucho más costosa en tiempo que una iteración en el escalado de TimberWolfSC, sobre todo a bajas temperaturas.

En la figura 4-3 se presenta una curva de enfriamiento con ambas propuestas de escalado, donde se comprueba la evolución de las soluciones en una prueba sobre el circuito Primary1 (833 celdas). Los tiempos obtenidos son del orden de 300 segundos para el escalado simplificado de TimberWolfSC, frente a unos 30 minutos que precisó el escalado de Huang.

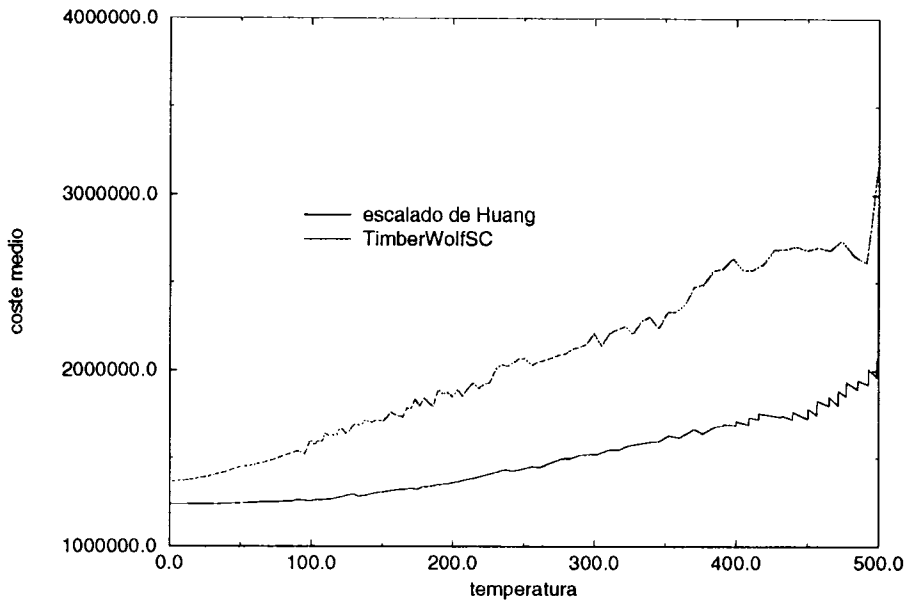


Figura 4-3: Comparación entre ambas propuestas de Escalado sobre el circuito Primary1

4.2.2 Influencia de la Temperatura Inicial

La temperatura de partida es un parámetro que influye enormemente en la calidad de la solución final. Una temperatura excesivamente alta hace que el sistema se pierda, ensayando soluciones de una manera aleatoria, sin mejorar el valor medio de la función de costes. Una temperatura de partida excesivamente baja impide un correcto funcionamiento de la técnica de enfriamiento simulado, quedando el sistema atrapado en un mínimo local.

La figura 4-4 presenta curvas de enfriamiento a distintas temperaturas iniciales, mostrando cómo la elección de la temperatura inicial es crítica a la hora de no caer en mínimos locales.

Circuito	TimberWolfSC		Esc.	Huang.
	Coste final	n^o iteraciones	Coste final	n^o iteraciones (aprox)
Fract (s109)	45000	123	34306	1300
Struct (mult16)	705279	123	586370	2500
Biomed (fan)	3940855	123	2193138	8900
Primary1 (CIRCUITX)	1499962	123	1219258	1800
Primary2 (RPROC)	5902171	123	4251079	3500

Tabla 4.1: Resultados comparativos entre el escalado de Huang y TimberWolfSC

4.3 Propuesta de un Algoritmo Combinado

La fase de aproximación se puede obviar si partimos de la certeza de que estamos en una situación parcialmente óptima, es decir, si aplicamos otra heurística que aporte con poco esfuerzo una buena solución inicial, y somos capaces de obtener un valor de la temperatura inicial del algoritmo de enfriamiento simulado que permite obtener nuevas soluciones, sin perder los beneficios de la solución inicial. En particular, lo que proponemos es el empleo de un algoritmo de colocación basado en multiparticiones que emplea como base el algoritmo de corte descrito en el capítulo 2 para generar la solución inicial al problema del enfriamiento simulado. Usando este algoritmo, se produce una distribución previa de las celdas más fuertemente conectadas del circuito, en regiones. Esta situación se traslada al enfriamiento simulado, pero con una temperatura T_0 lo suficientemente baja como para no destruir esta configuración previa. Obviamente partiremos de un valor de la función de costes bastante más bajo que el que se alcanzaría como solución de partida de un problema normal de enfriamiento simulado.

Hemos de tener cuidado en elegir una temperatura inicial demasiado baja, ya que en este caso no se alcanzaría un número suficiente de movimientos como para salir de mínimos locales.

El problema se reduce a encontrar una temperatura en la que se den dos

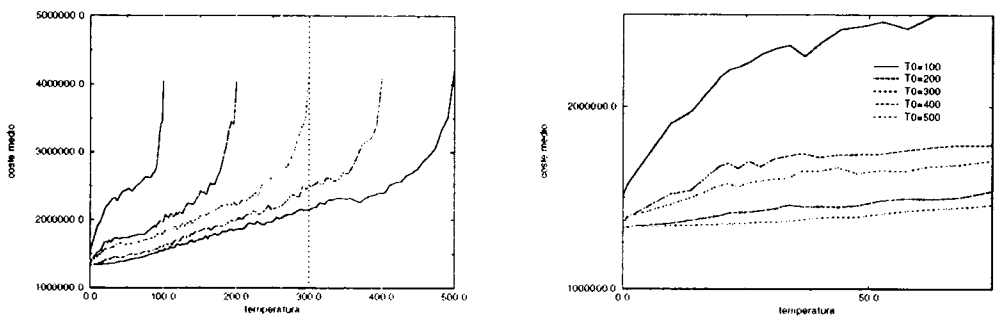


Figura 4-4: Comparación entre diferentes temperaturas iniciales sobre el circuito Primary1 con escalado simplificado de *TimberWolfSC*

condiciones:

1. se preserven los beneficios de la solución inicial.
2. se asuma un número suficiente de movimientos como para no quedar atrapado en mínimos locales.

A priori, se proponen dos tipos de métodos para obtener esa temperatura:

1. métodos basados en al análisis de la función de distribución
2. métodos que valoran las propiedades de la solución.

4.3.1 Métodos basados en la Función de Distribución

Para obtener la temperatura a la cual se conservan las propiedades heredadas del algoritmo de particiones [92], se puede partir del siguiente supuesto: la solución aportada es solución de equilibrio a esa temperatura. De esta manera los movimientos que se produzcan a esa temperatura harán que el valor de la función de costes media no se diferencie mucho del coste de la solución aportada. La forma típica de la función de distribución se representa en la figura 4-5, y es diferente a cada temperatura. A altas temperaturas los incrementos de costes son grandes y la

probabilidad de aceptar incrementos de costes de signo positivo es bastante alta. La función de distribución será, pues, simétrica en torno al 0, pero muy "amplia". A bajas temperaturas es difícil mejorar el coste, por ello la función de distribución será "delgada". Nótese que estas funciones se calculan cuando se ha alcanzado el equilibrio en la temperatura correspondiente, por lo que la distribución de los incrementos de coste debe tener media cero.

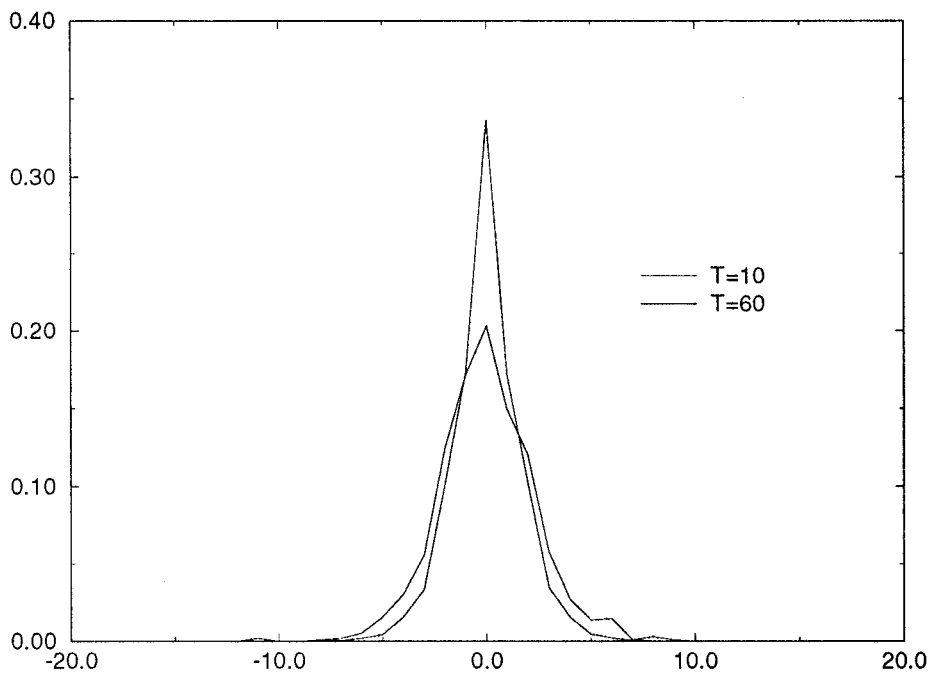


Figura 4-5: Forma típica de la función de Distribución de los incrementos de Costes

En [92] para caracterizar la temperatura del sistema se emplea una aproximación basada en la función de *distribución estática*. Esta función es la que se obtiene cuando tan sólo se consideran movimientos aleatorios, generados siempre a partir de la misma solución de partida. Los autores afirman que esta distribución difiere poco de la *función de distribución dinámica* en que cada movimiento se genera a partir de la configuración alcanzada en el movimiento anterior. Esta afirmación

plantea ciertas dudas; por ejemplo, piénsese en el caso en que, desafortunadamente, la situación de partida sea un mínimo local; sin embargo, los resultados presentados por los autores parecen justificar esta aproximación. En cualquier caso, el empleo de este método obliga a calcular la integral de ΔC a cada temperatura, lo que es bastante costoso.

La propuesta realizada por el autor de esta tesis en [1] es menos pretenciosa que la anterior, ya que se trabaja directamente con la función de distribución dinámica. En este caso el planteamiento es ligeramente diferente, ya que se busca, no la propia función de distribución, sino un parámetro que la caracterice. Si este parámetro es más sencillo de calcular que la propia función de distribución es posible obtener una temperatura característica del proceso de enfriamiento con un reducido coste computacional. El parámetro que caracteriza esta función de distribución puede ser de distinta naturaleza: puede ser el valor máximo de la misma, puede ser el valor del incremento máximo del coste aceptado, etc. Nótese que todos estos valores dependen de la temperatura.

A pesar de los esfuerzos realizados para conseguir identificar un parámetro característico que permita estimar una temperatura del sistema mediante el método antes comentado, no hemos sido capaces de encontrar uno que sea independiente del problema a optimizar. En particular, hemos constatado una gran dependencia con el tamaño del circuito.

4.3.2 Método de la *temperatura crítica*

Como ya se mencionó en la sección 4.2, en el planteamiento original del algoritmo de *enfriamiento simulado*, ya se hacía mención a la formación de *agrupaciones fuertemente interconectadas*. A una cierta temperatura T_c estas agrupaciones hacen que la variable *calor específico* alcance un "valle". Por tanto, como la partición inicial ha sido construida con bloques fuertemente interconectados, podemos emplear como temperatura inicial del proceso de enfriamiento la temperatura crítica de cambio de fase de un proceso de enfriamiento.

En la figura 4-6 se representa una curva típica real del comportamiento del

calor específico frente a la Temperatura T, en escala logarítmica para un circuito de unas 2000 puertas (circuito primary2). En la misma figura se representa la curva para otro circuito de unas 6000 puertas (biomed). En ellas se observa que se repite la circunstancia ya apuntada en [57] como una sucesión de valles más o menos acusada. Estos valles representan la transición entre soluciones poco organizadas y soluciones con una organización por bloques. Ahora, nuestro objetivo es aproximarnos a la temperatura característica T_c en que el *calor específico* genera los valles.

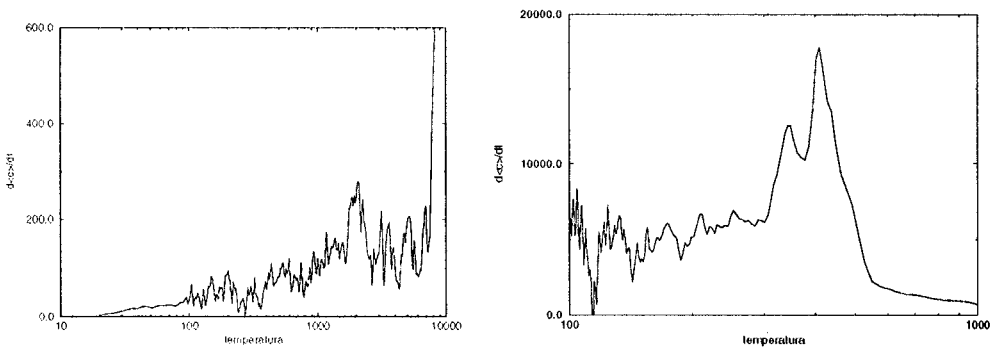


Figura 4-6: Forma de la curva de calor específico frente a la temperatura, a) para el circuito *Primary2*, b) para el circuito *biomed*.

Nuevamente, a pesar de los esfuerzos realizados, no es fácil encontrar un método simple que permita predecir la temperatura crítica del sistema, si no es realizando todo el proceso de enfriamiento simulado desde una temperatura elevada, por lo que tampoco haremos uso de esta aproximación.

4.3.3 Método basado en el escalado propuesto por Huang

La introducción de un escalado uniforme en la *curva de enfriamiento* propuesta por Huang [51] establece que:

$$\frac{d < c >}{d \ln T} = Cte \tag{4.5}$$

Esta constante, en principio es desconocida y depende de muchos factores, entre ellos del circuito (de su tamaño y de su estructura). Por consideraciones de

mecánica estadística:

$$\frac{d < c >}{d \ln T} = \frac{\sigma^2}{T} \tag{4.6}$$

Por tanto, resoviendo la ecuación diferencial se puede obtener la relación entre T y c de la manera siguiente:

$$T' = T \times e^{-\frac{\Delta C}{Cte}} \tag{4.7}$$

La idea del método propuesto consiste en realizar un número de exploraciones a fin de determinar el valor de la Cte del circuito que cumple dicha ecuación. Luego se extrapola la curva de enfriamiento y se toma como temperatura inicial el valor de la temperatura que tendría una situación de equilibrio de igual coste. Nótese que al realizar unas cuantas iteraciones hasta alcanzar el equilibrio en altas temperaturas contribuye poco al tiempo total del proceso, ya que los equilibrios a altas temperaturas se alcanzan muy rápidamente.

Al adoptar esta aproximación se cometen varios errores: 1) el valor de esta temperatura depende del error cometido en el procedimiento para evaluar la Cte y, sobre todo, 2) se considera que la solución que procede del algoritmo de particiones es una situación de equilibrio, obtenida por el escalado de Huang, lo que, evidentemente, no es cierto.

No obstante, lo único que pretendemos con este método es obtener una estimación de la temperatura inicial. Los resultados finales que se obtienen y que se muestran en los siguientes apartados avalan la eficiencia del método propuesto.

4.4 Realización del Algoritmo Combinado

En esta sección se aportan nuevas ideas sobre el algoritmo combinado propuesto. El objetivo final es el de realizar una combinación de algoritmos que mejore la eficiencia de cálculo del proceso de *Enfriamiento Simulado*, tanto en calidad de soluciones como en tiempo de computación.

4.4.1 Estructura General del Algoritmo Combinado

La estructura general del algoritmo propuesto consta de las siguientes etapas (figura 4-7):

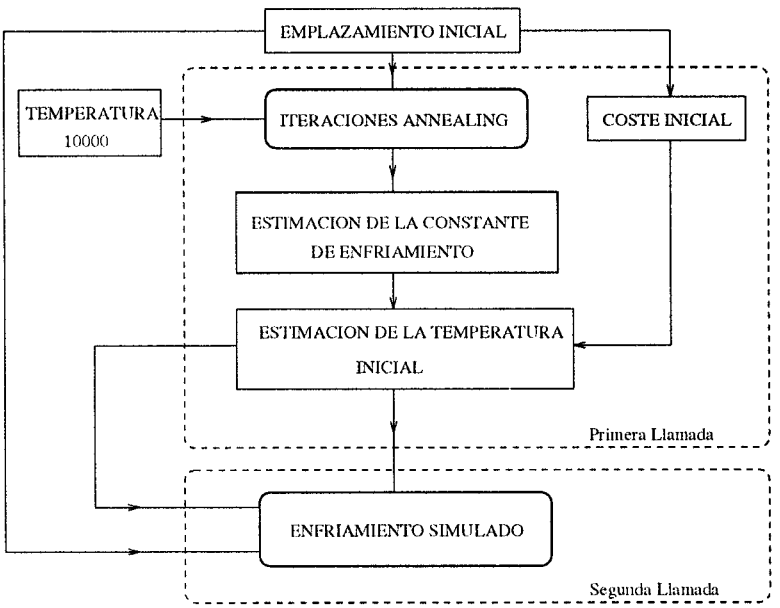


Figura 4-7: Esquema General del Desarrollo del Programa

1. Particionamiento del circuito en $n \times m$ bloques de similar tamaño y colocación inicial de esos bloques. El particionamiento se realiza mediante cuadrisección recursiva con propagación de terminales siguiendo un esquema similar al propuesto por Suaris y Kedem [110]. Para mejorar los resultados de la cuadrisección se realiza una compactación previa del circuito empleando el método de particionamiento propuesto en el capítulo 2.

2. A partir de la solución obtenida, se colocan las celdas en cada bloque de una manera aleatoria. El resultado se emplea como solución inicial en un proceso de enfriamiento simulado.

3. Se determina una temperatura inicial adecuada para comenzar el algoritmo de enfriamiento simulado, empleando el método propuesto en el apartado ante-

rior. Para ello se realizan unas pocas iteraciones del algoritmo de enfriamiento simulado partiendo de una temperatura elevada, con el objeto de estimar la constante de escalado. Estimada esta constante, se extrapola la curva de enfriamiento y se determina la temperatura a la cual se obtendría (en equilibrio) un coste igual al de la solución inicial, y

4. A partir de esta solución y esta temperatura inicial se continúa el proceso de enfriamiento simulado, siguiendo el escalado simple de TimberWolfSC, para no alargar excesivamente los tiempos de computación.

4.4.2 Cálculo de la Temperatura T_0

La tabla 4.2 presenta los resultados de calcular T_0 a los diferentes *circuitos patrón* de que disponemos. Nótese que este valor no tiene por qué ser necesariamente una estimación exacta de la temperatura T_c. El punto de partida puede ser simplemente cercano a T_c, debido a la posible dispersión en los valores de equilibrio para cada temperatura. El error cometido puede acotarse por la desviación típica a esa temperatura cuyo valor estimado es de:

$$\sigma = \sqrt{Cte \times T} \tag{4.8}$$

Es conveniente indicar que TimberWolfSC fija la temperatura inicial del propceso de enfriamiento, de una manera arbitraria en 500.

4.4.3 Estructura General del Algoritmo Combinado

El algoritmo de exploración se encuentra representado en la figura 4-11. En él se integran dos llamadas al *Enfriamiento Simulado*, siendo la primera para realizar las estimaciones propuestas, y la segunda para realizar la optimización.

El cálculo de la estimación de la temperatura de partida se hace a partir de una caracterización del propio sistema cuando alcanza diversas situaciones de equilibrio. Por ello se adopta una temperatura de partida de valor muy alto (10000). A esta temperatura, el sistema acepta cualquier movimiento propuesto. El sistema se va enfriando y, a partir de una cierta temperatura, el proceso comienza a rechazar

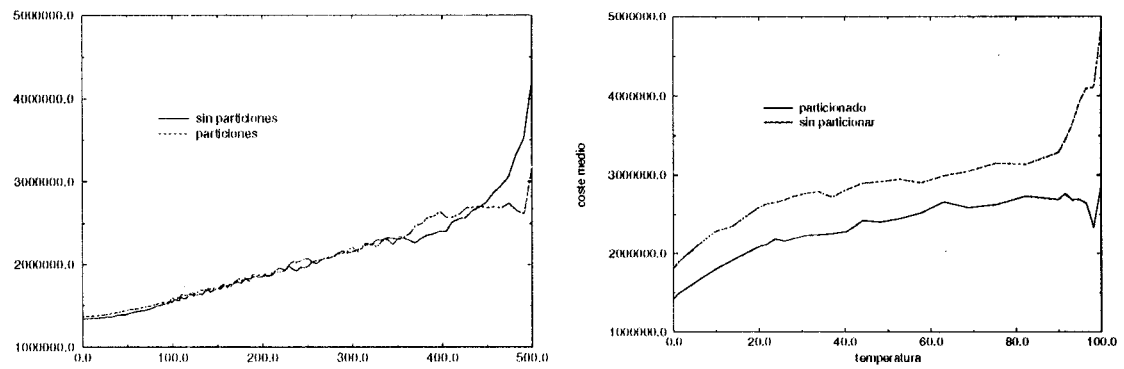


Figura 4-8: Influencia de la temperatura inicial. Gráficos comparativos entre curvas a diferentes temperaturas de partida (500 y 100)

algunos movimientos. A partir de esta temperatura se continúa enfriando hasta unas 10 iteraciones (es una cantidad arbitraria), donde se puede ya considerar que el sistema está bien controlado por la temperatura. En este punto se estima la constante de enfriamiento, del sistema y, extrapolando la curva de enfriamiento, se calcula la temperatura inicial adecuada.

Obviamente esta temperatura de partida es estimada, y se obtienen valores que simplemente la aproximan. Experimentamente se comprueba que estos valores son adecuados.

Circuito	4 bloques	16 bloques	64 bloques
Fract (s109)	21	80	-
Struct (mult16)	88	80	81
Biomed (fan)	80	82	73
Primary1 (CIRCUITX)	150.7	135	139
Primary2 (RPROC)	149	135	157

Tabla 4.2: Resultados del cálculo de la temperatura inicial T_0

Se han realizado diversas pruebas sobre cada uno de los circuitos patrón. Las características de estos circuitos, las condiciones de ensayo y los resultados se encuentran detallados en el apéndice III. Se han dispuesto como soluciones de partida, multisecciones en 4, 16 y 64 bloques.

Circuito	Coste Inicial	Coste final	T_0	n^o iteraciones	tiempo
Fract (s109)	65791	44818	21.8	7	4.8
Struct (mult16)	1907973	729117	88	22	288.9
Biomed (fan)	11627216	3514981	80	22	1849.1
Primary1 (CIRCUITX)	3215739	1391150	150	50	258.5
Primary2 (RPROC)	14627079	5508134	149	52	3025.5

Tabla 4.3: Resultados obtenidos con el algoritmo combinado propuesto (todos los ejemplos sobre particiones en 4 bloques)

Circuito	Coste Inicial	Coste final	T_0	n^o iteraciones	tiempo
Fract (s109)	62393	35338	80	22	9.2
Struct (mult16)	1759185	665355	80	22	287.8
Biomed (fan)	12205817	3341821	82	22	21016.3
Primary1 (CIRCUITX)	2923128	1420563	135	48	279.5
Primary2 (RPROC)	12731496	5199174	155	53	2690.0

Tabla 4.4: Resultados obtenidos con el algoritmo combinado propuesto (todos los ejemplos sobre particiones en 16 bloques)

Figura 4-9: a) Colocación con el algoritmo propuesto. b) Resultado con *wolfe*

Circuito	Coste Inicial	Coste final	T_0	n^o iteraciones	tiempo
Fract (s109)	-	-	-	-	-
Struct (mult16)	1731107	646982	81	22	292.9
Biomed (fan)	9357507	3353480	73.2	21	20474.8
Primary1 (CIRCUITX)	2461699	1438057	139	49	280.2
Primary2 (RPROC)	11050820	5299792	157	54	2752.9

Tabla 4.5: Resultados obtenidos con el algoritmo combinado propuesto (todos los ejemplos sobre particiones en 64 bloques)

4.4.4 *Colocación* Inicial

Empleando un algoritmo de colocación por cuadrisección recursiva se han realizado particiones de 4,16 y 64 bloques. Posteriormente se ha realizado un programa que coloca las celdas dentro de su bloque de forma aleatoria, obteniendo una *Colocación* inicial, tras la cual se ejecuta el programa TimberWolfSC tal y como se ha descrito en la sección 4.4.3.

La selección del número de particiones en que se distribuye la *Colocación* inicial es función del número de celdas. Se han realizado pruebas de comportamiento del algoritmo con distintos números de celdas y se ha comprobado, como se muestra en la Tabla 4.6 que pocas particiones en circuitos muy grandes mejoran poco la solución final. De la misma manera, un gran número de particiones en circuitos pequeños tampoco aumenta la eficiencia.

Las tablas 4.3 a 4.6 resumen los resultados obtenidos con el método propuesto. En estas tablas se muestra igualmente el porcentaje de disminución en la función de costes conseguido respecto a la solución obtenida por TimberWolfSC–4.2 y el ahorro en tiempo. Se puede comprobar que, en casi todos los casos, las soluciones obtenidas son mejores que las de TimberWolfSC, y que, sobre todo, los tiempos requeridos son muy inferiores. En la tabla 4.6 se emplean los coeficientes de mejora

Circuito	4 bloques			16 bloques			64 bloques		
	Coste final	%c	%t	Coste final	%c	%t	Coste final	%c	%t
Fract (s109)	44818	1.8	89.23	35338	21.5	79	-	-	-
Struct (mult16)	729117	-3.4	81.9	665355	5.66	81.93	646982	8.62	81.61
Biomed (fan)	3514981	10.8	24.97	3341821	15.2	20.56	3353480	14.9	22.611
Primary1 (CIRCUITX)	1391150	7.25	47.23	1420563	5.29	42.94	1438057	4.12	42.08
Primary2 (RPROC)	5508134	6.67	32.26	5199174	11.91	39.77	5299792	10.2	38.37

Tabla 4.6: Resultados obtenidos con diferentes números de particiones. Se muestra, en cada caso, el coste final, el % de mejora en coste (%c) y el % de mejora en tiempo (%t)

Circuito	4 bloques	16 bloques	64 bloques
Fract (s109)	33	-	-
Struct (mult16)	-1	2	-
Primary1 (CIRCUITX)	16	8	6
Primary2 (RPROC)	-1	9	2

Tabla 4.7: Procentaje de memoria en área, medido sobre la disposición final de los circuitos.

en coste, $\%c$, y en tiempo, $\%t$, que se definen como:

$$\%c = 100 \times (1 - \frac{coste\ algor.\ prop}{coste\ TimberWolfSC}) \tag{4.9}$$

$$\%t = 100 \times (1 - \frac{tiempo\ algor.\ prop}{tiempo\ TimberWolfSC}) \tag{4.10}$$

En la figura 4-10 se muestra de una manera gráfica, la evolución de $\%c$ y $\%t$ frente al tamaño del circuito. En esta figura y en la tabla 4.6 no se incluye el tiempo que se emplea en realizar la colocación inicial y en determinar la temperatura inicial, pero este tiempo es pequeño en comparación con el tiempo total de colocación. Para los casos contemplados en la tabla 4.6, este tiempo representa, en el peor de los casos, menos de un 10 % del tiempo total.

Como es sabido, TimberWolfSC también incorpora rutinas que ejecutan el proceso de *Conexionado Global* y *de Detalle*. Los tiempos de computación de las tablas 4.3, 4.4 y 4.5 incluyen el proceso completo, de forma que se puede comprobar la eficiencia de la etapa anterior sobre el tiempo total requerido para la *Disposición* de los circuitos.

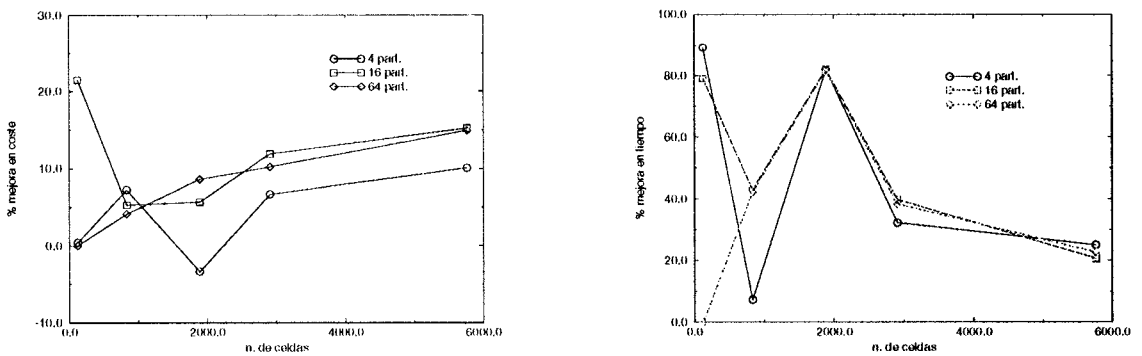


Figura 4-10: Mejoras porcentuales en coste y tiempo del algoritmo propuesto

En la tabla 4.7 se muestra el procentaje de mejora obtenido en el área de los circuitos, medidas realizadas sobre las disposiciones finales. Estas áreas se han obtenido con ayuda del programa *chipstats* de las herramientas *Octtools*. No se incluye el circuito *biomed(fan)*, ya que, debido a su gran tamaño, los ficheros temporales necesarios no caben en los directorios correspondientes.

En el apéndice VI se muestran las disposiciones finales obtenidas por *TomberWolfSC*, y por el algoritmo propuesto para los circuitos denominados *fract* y *struct*. Estas disposiciones se han obtenido con el programa *oct2ps* de las herramientas *Octtools*.

4.5 Conclusiones

De las experiencias realizadas con el algoritmo *Enfriamiento Simulado* se puede desprender que, a pesar de ser un algoritmo muy eficaz en la resolución de problemas de *colocación* de celdas, realiza una gran cantidad de movimientos a tem-

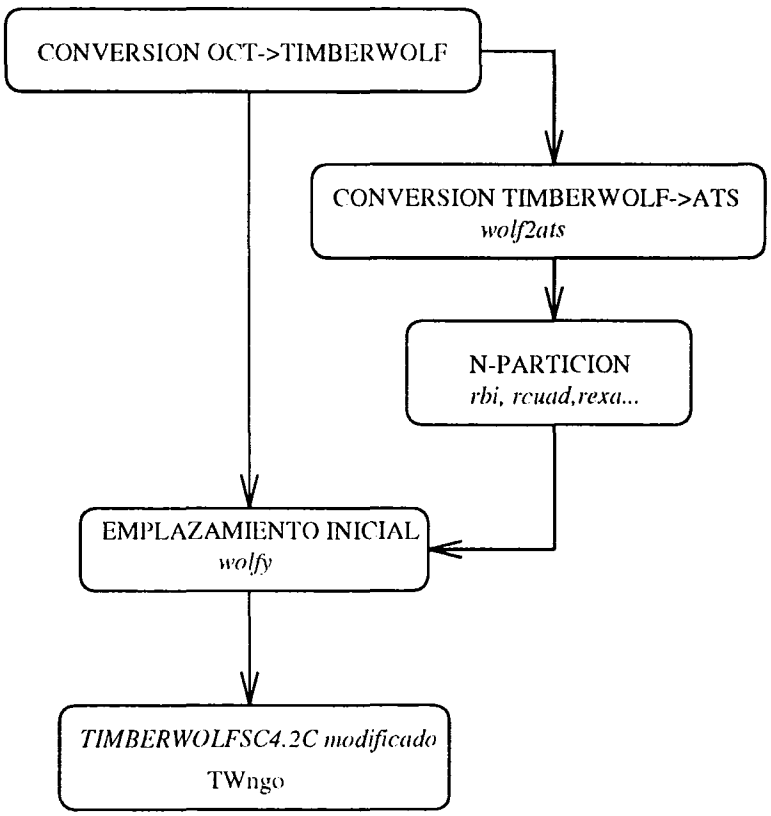


Figura 4-11: Esquema General del Desarrollo del Programa

peraturas elevadas que contribuyen pobremente a la mejora de la solución. Una forma de mejorar este pobre comportamiento es la combinación de este algoritmo con otras técnicas que aproximen, al menos parcialmente, a la solución final, y así evitar las etapas iniciales del proceso. El problema consiste ahora en encontrar una situación de partida favorable. Por ello se ha empleado una algoritmo de cuadrisección recursiva como el propuesto por Suaris y Kedem [110] para obtener una buena colocación inicial y se ha ensayado una idea simple para aproximar la temperatura inicial de un proceso de enfriamiento simulado que permite obtener un mínimo de gran calidad en la función de coste, sin destruir los beneficios de la situación de partida. Para ello se estima la constante de enfriamiento del circuito a partir de unas iteraciones a temperaturas elevadas. Estimada esta constante, se

estrapola la curva de enfriamiento para determinar la temperatura de partida. Se ha demostrado de manera experimental que se alcanzan iguales o mejores soluciones que TimberWolfSC, con un gran ahorro en tiempo.

Se ha desarrollado un paquete de síntesis que emplea el entorno de desarrollo Octtools 5.1. La estructura de los programas se muestra en la figura 4-11

Capítulo 5

El Problema de la Síntesis Analógica

5.1 Introducción

Este capítulo esta dedicado a exponer una faceta del diseño de ciruitos integrados CMOS muy diferente a la hasta ahora expuesta de la *Colocación*. Nos disponemos a abordar el problema de sintetizar celdas analógicas a partir de unas especificaciones de diseño.

El diseño de circuitos integrados digitales ha sido desde hace unos diez años una cuestión que no plantea excesivos riesgos, desde el punto de vista del diseñador. Prácticamente es posible abordar un proyecto con una probabilidad del 99% de éxito en el primer envío al fabricante con apenas una experiencia corta en el campo del diseño CMOS. Sin embargo, el campo analógico se encuentra aún bastante limitado en el campo del diseño industrial de Aplicación Específica (ASIC). Usualmente se trabaja con librerías de celdas cerradas (o en el mejor de los casos parametrizables) absolutamente garantizadas por el fabricante. La geometría de la celda está definida por un diseñador con la suficiente experiencia. No obstante, existe una tendencia creciente a incluir bloques analógicos en circuitos ASIC's [53]. Debido a la necesidad de contar con celdas de distintas especificaciones se hacen necesarias herramientas capaces de automatizar el diseño de circuitos analógicos y mixtos. El diseño de la

parte analógica de un chip sigue precisando de mayores esfuerzos que la digital.

Cuando se compara con un circuito digital, el diseño de un circuito analógico resulta difícil, debido a la compleja dependencia que existe entre los parámetros de diseño y las prestaciones del mismo. El proceso típico de diseño parte de una aproximación a la solución inicial y sigue con continuas simulaciones y rediseños hasta alcanzar un circuito que cumpla las especificaciones. Este proceso requiere una experiencia por parte del diseñador y un elevado tiempo dedicado a cálculo. Todas estas razones confirman la necesidad de desarrollar herramientas eficientes para el diseño automático de circuitos analógicos.

La síntesis de celdas analógicas se puede descomponer en dos etapas (figura 5-1):

1. Selección de la topología más apropiada para las condiciones requeridas.

2. Dimensionado de los elementos de esa topología que mejor responden a estos requerimientos.

De la primera cuestión existe una limitada bibliografía al respecto: [32], [48], [61], [112], [22] y no nos vamos a ocupar en esta tesis. Dedicaremos nuestra atención al segundo de los problemas. Su peculiar problemática permite ser atacado por algoritmos de los hasta ahora expuestos, y de ahí su inclusión como una parte importante en esta tesis.

En este capítulo vamos a abordar el problema del diseño automático de celdas analógicas desde el punto de vista de la optimización de sus prestaciones. Nos restrigiremos a tres topologías básicas de amplificadores operacionales, aunque las soluciones propuestas pueden extenderse a otras topologías de circuitos analógicos.

Mostraremos que desde el punto de vista matemático es un problema en el espacio continuo (en la práctica es discreto debido a las unidades de medida, que están en función de la unidad tecnológica λ), multivariable y con restricciones, suficientemente complejo como para aplicar las soluciones heurísticas de la optimización de problemas combinatorios.

5.2 Síntesis de Celdas Analógicas

Una celda digital se especifica con unos pocos parámetros, por ejemplo, un inversor puede especificarse por sus retrasos de puerta y su *fan–out* o cargabilidad, además del área. Sin embargo para especificar una celda analógica se necesitan gran cantidad de parámetros: un amplificador operacional puede especificarse mediante su disipación de potencia, ganancia en pequeña señal, ancho de banda, margen de fase, tiempo de establecimiento, $1/f$, ruido, área, ...etc .

El bloque analógico básico es el amplificador operacional. Convertidores, comparadores, filtros, contienen este tipo de celdas y por tanto será el objeto de nuestro estudio.

Si se trabajara con una librería fija de celdas, sería necesario disponer de una extensa gama de celdas prediseñadas con un amplio espectro de comportamientos que cubra todas las posibilidades. Esto no es posible debido al enorme número de celdas que habrían de incluirse en la librería. Se precisa, pues, de *generadores automáticos de celdas* capaces de sintetizar celdas analógicas a partir de las especificaciones del usuario. El programa de síntesis devolverá la topología más apropiada a las especificaciones y los valores de los parámetros geométricos de esa topología. La selección de la topología se realiza a partir de los contenidos de una base de datos y con criterios de selección basados en la experiencia y los estudios teóricos.

Una vez determinada la topología podemos disponer de un *referencial* del circuito a optimizar. Es deseable que, además de cumplir las especificaciones, el circuito tenga muy bajo consumo y con área mínima.

5.2.1 Modelos escogidos para la Optimización

En este estudio nos limitaremos, por simplicidad, a tres topologías de Amplificadores Operacionles: el Amplificador Operacional Básico de Dos Etapas (Basic Two Stage, BTS), el Amplificador Operacional de Salida en Transconductancia (Output Tranconductance Amplifier, OTA) y el OTA Cascodo Plegado. La figura 5-2 muestra un esquema de cada una de las topologías ensayadas.

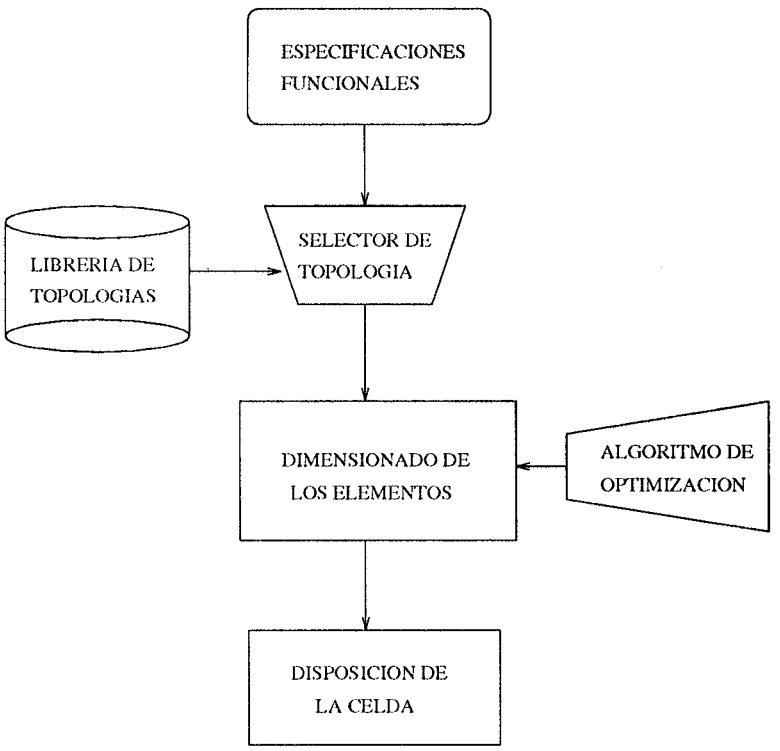


Figura 5-1: Proceso de diseño analógico.

5.3 Problema de Optimización

La síntesis de celdas analógicas es un problema multivariable, donde las características a obtener son de magnitudes muy diferentes y por tanto es difícil realizar una búsqueda por el espacio de configuraciones que sea numéricamente eficiente. Por ello es necesario una adimensionalización de las variables.

Las variables a determinar son los parámetros geométricos mientras que las especificaciones son características eléctricas. Por tanto se hace necesario un mecanismo que relaciones las variables independientes o geométricas con las variables dependientes o eléctricas.

La bondad de la solución encontrada se evalúa a través de una función de costes. Esta función es el resultado de comparar cada una de las características que

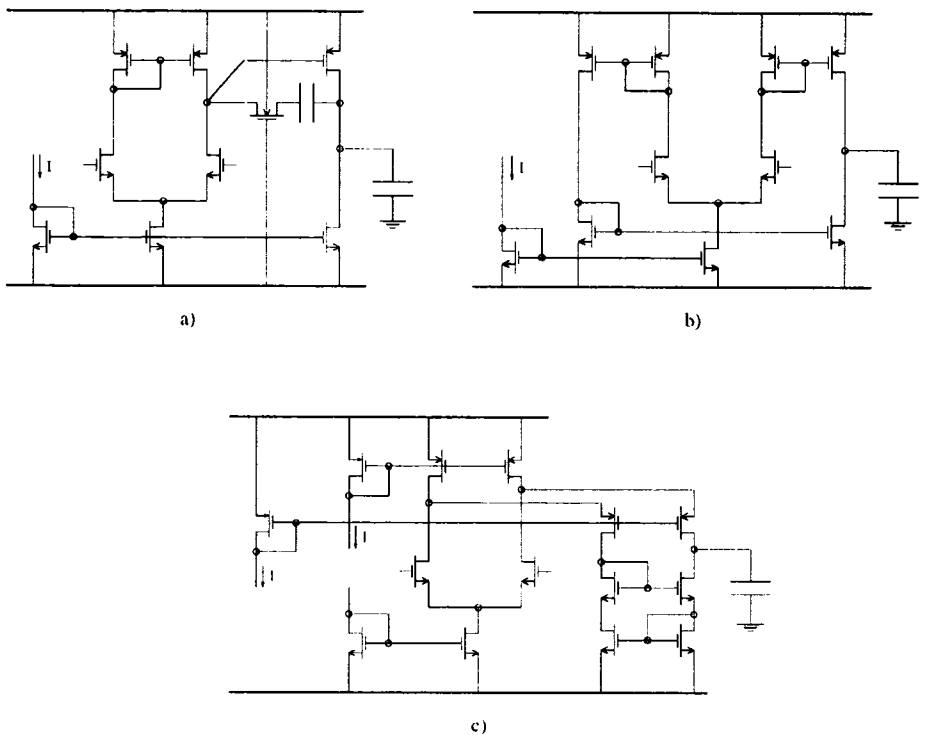


Figura 5-2: Topologías Básicas de Amplificadores Operacionales: a) Básico de Dos Etapas (BTS), b) Amplificador de Salida Transconductancia (OTA) y c) OTA Cascodo Plegado (CAS).

se han pedido en las especificaciones con la solución que se ensaya en un paso del algoritmo.

Existen dos aproximaciones para la evaluación de la función de costes. La primera de ellas emplea modelos aproximados de los dispositivos que conducen a expresiones simples para cada una de las especificaciones requeridas [45], [36]. Si bien esto permite formular de una manera analítica la dependencia entre la función de costes y los parámetros físicos de la celda, los resultados obtenidos tan sólo son aproximados. Además, la tendencia a disminuir las dimensiones geométricas de los dispositivos a medida que los procesos tecnológicos mejoran, hace que estos modelos simplificados sean cada vez más inexactos.

La segunda aproximación emplea modelos muy detallados de los dispositivos. Dado que la formulación analítica de estos modelos es muy compleja [117], se suelen emplear simuladores de circuitos como SPICE que incorporan rutinas de modelado muy detallado, para determinar los parámetros del circuito y evaluar la función de coste.

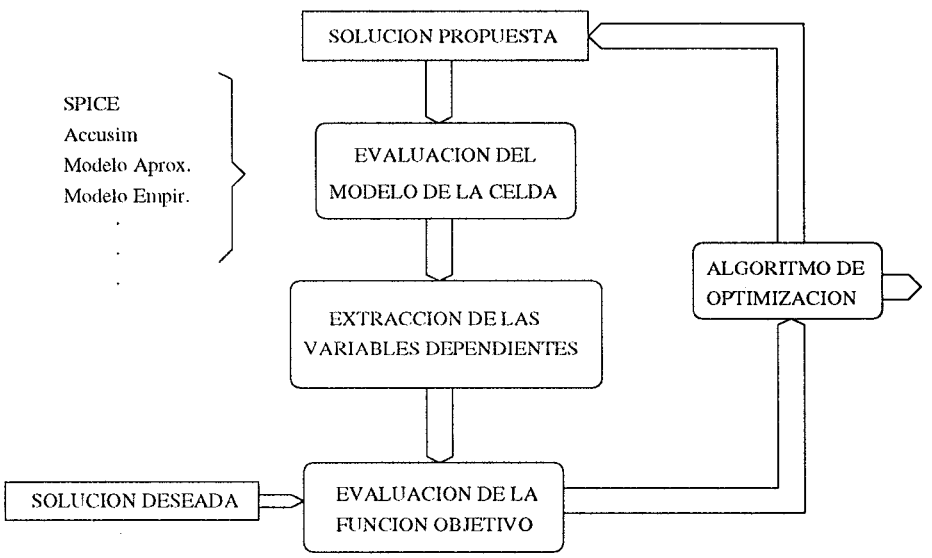


Figura 5-3: Esquema General de la Optimización.

Diferentes técnicas de optimización se han empleado en el diseño de celdas analógicas. Una primera revisión de estas técnicas se puede encontrar en [12]. Estas técnicas se pueden englobar en tres grandes grupos:

1. Técnicas que emplean reglas sencillas de diseño, basadas en la experiencia de los diseñadores. Entre estas aproximaciones podemos citar IDAC [28], y los módulos expertos de BLADES [32] y OASYS [48].

2. Técnicas de optimización sin restricciones. Estas técnicas emplean algoritmos gradenciales, con o sin evaluación del gradiente. Para evaluar la función objetivo pueden emplear un modelo aproximado del circuito en cada iteración,

como en el caso de [61] y la primera fase de [84]. Otros, en concreto, emplean simuladores de circuitos (p.e. SPICE) [79], como la segunda fase de [84].

3. Técnicas de optimización con restricciones. En este caso, se suelen emplear simuladores de circuitos para evaluar las prestaciones de la celda. En este apartado podemos incluir a DELIGHT-SPICE [82], ECTASY [108], DONALD [113] y la referencia [48]

4. Técnicas de búsqueda aleatoria. Estre éstas predomina el empleo de las técnicas de enfriamiento simulado. Generalmente emplean un modelo aproximado del circuito, dado que el número de iteraciones necesarias para alcanzar el óptimo hace inviable el uso de un simulador de circuitos. Un ejemplo de esta aproximación se encuentra en [38]. La solución del enfriamiento simulado puede acelerarse introduciendo una información gradencial, como se hace en [83] y [20].

5.3.1 Algoritmos Aplicados a la Síntesis de Celdas Analógicas

Hasta ahora se ha hecho mención a que el campo de aplicación de los algoritmos es el espacio continuo. Es posible trabajar sobre un espacio discretizado de forma que los movimientos de las variables se realizan a través de una *parrilla*. Esta aproximación se acerca más a la realidad, dado que las dimensiones geométricas de los elementos se refieren siempre a una dimensión mínima de las máscaras. Sin embargo la visión geométrica del espacio continuo no se pierde en el sentido de que las tolerancias de las geometrías son comparables a la precisión de la *parrilla* y por tanto se pueden redimensionar, a "posteriori", los valores continuos a los discretos sin pérdida de generalidad, ya que siempre se trabajará bajo una cierta incertibumbre de resultados. Por otra parte el espacio continuo ofrece una serie de ventajas de codificación del algoritmo que le permiten ser más rápido y simple: no hay que preocuparse por el tipo de variables ni el valor de éstas al codificar. En el caso discreto habría que operar sobre un espacio con valores fijos, lo que presenta el problema de estar constantemente corrigiendo el valor de las variables.

5.4 Desarrollo del Programa de Síntesis

En el capítulo siguiente se presenta el resultado de aplicar distintas técnicas inteligentes de búsqueda aleatoria al problema de la síntesis de celdas analógicas. Los algoritmos de síntesis se han codificado en unas rutinas en lenguaje C que se han empleado como base de un paquete de diseño de celdas analógicas, realizado por D. J. Chávez en el desarrollo de su tesis, en el Grupo de Tecnología Electrónica de la E.S. de Ingenieros de Sevilla [23] cuyas características más sobresalientes se discuten a continuación.

5.4.1 Adimensionalización de las Variables

Se ha mencionado la necesidad de poder comparar dentro de la función de costes variables de muy diferente orden de magnitud. Para poder realizar esta comparación se adimensionalizan las variables mediante la expresión:

$$x[j] = X_{min}[j] + (X_{max}[j] - X_{min}[j]) \cdot \sin^2(w[j]) \tag{5.1}$$

cuya función inversa es:

$$w[k] = \arcsin \sqrt{\frac{x[j] - X_{min}[j]}{(X_{max}[j] - X_{min}[j])}} \tag{5.2}$$

donde $w[j]$ es la variable adimensionalizada. El valor de $w[j]$ está comprendido entre $[0, \frac{\pi}{4}]$. Los valores X_{min} y X_{max} de cada variable no son más que valores extremos posibles de las mismas.

5.4.2 Función de Costes

Los algoritmos propuestos tratan de minimizar una función de costes sin restricciones. La función de costes elegida para la optimización es la definida en [84], si bien cualquier otra función de costes podría haberse empleado con resultados similares.

$$f_{obj} = \sum_{i=1}^{N_{esp}} {f_i}^2 \tag{5.3}$$

$$f_i = W_i \cdot (\sqrt{Err_i{}^2 + \frac{1}{W_i{}^2}} + Sign_i \cdot Err_i) \tag{5.4}$$

$$Err_i = \frac{P_i(\vec{x})}{P_i{}^{esp}} - 1 \tag{5.5}$$

N_{esp} es el número de especificaciones deseadas a optimizar en la celda. W_i es un factor de ponderación de cada una de ellas (por defecto $W_i = 1$). $\vec{x}$ es el vector de parámetros físicos (dimensiones) de la celda. $P_i(\vec{x})$ es el valor de la especificación i como función de los parámetros físicos. Esta relación se obtiene a través del modelo analítico o mediante la simulación seguida de un post–análisis. $P_i{}^{esp}$ es el valor de la especificación i pedida por el usuario. $Sign_i$ es un parámetro que toma el valor $+1$ o -1 dependiendo de que los requerimientos sean $P_i(\vec{x})$ mayor que $P_i{}^{esp}$, o menor que $P_i{}^{esp}$.

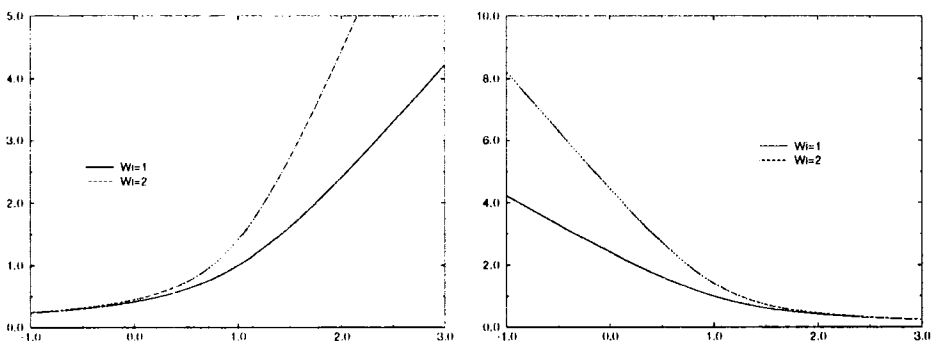


Figura 5-4: Forma de la curva función objetivo para cada variable. En abscisa se representan $\frac{P_i(\vec{x})}{P_i{}^{esp}}$ y en ordenadas el valor de la función. A la izquierda suponiendo que se desea un valor mayor que la especificación y a la derecha menor.

La figura 5-4 representa la forma de la función de costes para una de las especificaciones. Se comprueba que su característica más sobresaliente es la de penalizar mucho la especificación que no se cumple, y recompensar ligeramente aquella que se cumple en exceso.

5.4.3 Generación de Movimientos

La generación de movimientos se produce de manera aleatoria en el espacio de las variables adimensionales. La expresión que obtiene el punto siguiente es:

$$\omega_{k+1}[i] = \omega_k[i] + rnd_k[i] \cdot coef(k) \tag{5.6}$$

Donde $\omega_k[i]$ es el valor de la variable adimensionalizada en la iteración k. $rnd_k[i]$ es un generador de números aleatorios que toma valores en el intervalo $[0, 1]$.

La función $coef(k)$ tiene la misión de actuar sobre el espacio de búsqueda. Cuando se alcanza un nuevo valor del óptimo, se reduce el valor del coeficiente, de forma que centraliza la búsqueda en un entorno reducido del punto actual. A medida que el número de iteraciones aumenta y no se encuentran nuevos mínimos, el coeficiente aumenta. Se ha escogido una función que cumple esta características:

$$coef(j) = \frac{\pi}{2} \cdot 0.917^{(50-j)} \tag{5.7}$$

con $coef(j) \in [0.01, \frac{\pi}{2}]$. El valor 50 se ha escogido debido a que el algoritmo *Búsqueda Tabú* no permite que el argumento j tome valores mayores que 50. Si se tratara del *Enfriamiento Simulado* la función dependería de la Temperatura T. El valor 0.75 se corresponde a un valor menor que el argumento máximo posible de la función de adimensionalización, que es periódica y no puede valer más que $\frac{\pi}{4}$. Por último el 0.917 es un valor empírico.

Capítulo 6

Aportaciones a la Síntesis Analógica

6.1 Introducción

En este capítulo se presentan aportaciones originales al problema de la síntesis de celdas analógicas. Se presenta un programa de optimización mediante un algoritmo mejorado de enfriamiento simulado, que emplea una técnica gradencial para orientar la búsqueda del algoritmo a bajas temperaturas.

También se presenta una aplicación de los métodos de "búsqueda tabú" (*tabu search*) al mismo problema. Tal como se demuestra en este capítulo, los resultados de aplicar esta técnica con modelos simplificados del comportamiento de los circuitos dan resultados similares a los obtenidos con las rutinas de enfriamiento simulado, pero con un número muy reducido de iteraciones. Esto permite emplear un simulador de circuitos como SPICE para evaluar la función de costes en cada iteración, con lo que se evitan los errores que aparecen cuando se emplean modelos simplificados del circuito.

6.2 *Enfriamiento Simulado* y Síntesis Analógica

Existen algunas aplicaciones del algoritmo *Enfriamiento Simulado* a la síntesis de celdas analógicas con unos resultados finales bastante interesantes. En particular, en [77] se utiliza un espacio discretizado de parámetros, y en [20]–[21] se utiliza un espacio continuo. Debido al elevado número de iteraciones que caracteriza al algoritmo de enfriamiento simulado, casi todas estas referencias emplean un modelo aproximado del circuito para estimar el valor de la función de costes en cada iteración. Modelos aproximados de bloques analógicos simples pueden encontrarse en [36] y [45].Tan sólo en [77] se emplea SPICE en cada iteración, pero para que los resultados sean buenos es necesario un número tan elevado de iteraciones que hace inviable esta aproximación.

Los resultados que a continuación se exponen emplean un algoritmo de enfriamiento simulado para la optimización de los parámetros del circuito, cuyos elementos se detallan en el capítulo 5.

6.2.1 Introducción de una técnica gradencial para acelerar el proceso de optimización

En [20] un grupo de autores, entre los que se encuentra el autor de esta tesis, proponen un método que mejora el proceso de enfriamiento simulado, mediante la introducción de un término gradencial en el proceso de búsqueda. Para ello en la función que perturba la solución en cada iteración (expresión (5.6) que es repetida aquí por completitud):

$$\omega_{k+1}[i] = \omega_k[i] + rnd_k[i] \cdot coef(k) \tag{6.1}$$

el término $rnd_k[j]$ se evalúa mediante la expresión:

$$rnd_k[j] = (1 - t_c)\ \ unrnd_k[j] - t_c\ \ grad_k[j] \tag{6.2}$$

donde $unrnd_k[j]$ es un número aleatorio normalizado, uniformemente distribuido y $grad_k[j]$ es el gradiente normalizado de la función objetivo en el punto

k–ésimo. t_c es un factor de ponderación en el intervalo $[0, 0.8]$ que se ajusta por la expresión

$$t_c = \frac{1}{1 + temp/k} \tag{6.3}$$

El valor de k determina el punto de arranque del término gradiente. Nótese que si $k = 0$, el algoritmo se convierte en un enfriamiento simulado convencional. Si $k = \infty$, se convierte en un algoritmo gradencial clásico. Valores intermedios, (p.e. $k = 1$) hacen que el algoritmo se comporte como un enfriamiento simulado a altas temperaturas y vaya sesgando las soluciones propuestas hacia la dirección opuesta al gradiente, a medida que el sistema se enfría. De esta manera, se pretende evitar el elevado número de iteraciones que caracteriza al *enfriamiento simulado* en temperaturas próximas a la congelación. Los resultados indicados en [20] y resumidos en la tabla 6.1 confirman que el algoritmo proporciona resultados similares a los obtenidos con enfriamiento simulado, con un número de iteraciones reducido en casi un orden de magnitud. Los resultados que se muestran en [20] se refieren a la síntesis de Amplificadores Operacionales, cuyas topologías se representan en la figura 5-2. A pesar de la reducción del número de iteraciones del algoritmo propuesto, en algunos casos se hicieron necesarias más de 30.000 llamadas a la rutina de evaluación de la función de costes.

Modelo	BTS		OTA	
	ES	Comb	ES	Comb
n^o medio iteraciones	11800	4150	82000	35354
f_{obj} mínima	1.572	1.578	1.514	1.556
f_{obj} media	1.584	1.568	1.516	1.560

Tabla 6.1: Comparación entre *enfriamiento simulado* y el algoritmo propuesto

Queremos llamar la atención de que este elevado número de iteraciones hace inviable aplicar un modelo más exacto para evaluar el comportamiento del circuito, como el que se obtiene de una simulación SPICE, ya que los tiempos de

computación serían excesivos. Además, el autor ha comprobado experimentalmente que la función de costes se vuelve mal condicionada cuando se emplea SPICE como evaluador de las prestaciones del circuito. Esto hace que, en este caso, el número de iteraciones necesarias para alcanzar el óptimo aumente respecto al tiempo que se emplea con un modelo aproximado del mismo. La inexactitud de la predicción realizada por el modelo aproximado se muestra en la tabla 6.2, siendo especialmente importante el error en el ancho de banda.

Modelo	BTS			OTA		
Especificación	Esp.	Pred	SPICE	Esp.	Pred	SPICE
Ganancia DC(dB)	90	99.4	93.7	60	62.6	59.5
Frec. Gan. Unidad(MHz)	1	2.00	0.64	1	5.57	2.77
Margen de Fase	60	70.4	63	60	76.0	75.5
Slew Rate	1	2.55	1.66	1	1.87	1.5
Disipación de Potencia(mW)	1	0.55	0.33	5	0.29	0.21
Area activa	min.	1312	-	min.	3144	-

Tabla 6.2: Comparación de la solución obtenida analítica frente a una evaluación de SPICE.

6.3 Heurística de Búsqueda Tabú

En la próxima sección se presenta el desarrollo de una solución original al problema de la síntesis de celdas analógicas mediante el empleo de un algoritmo de *Búsqueda Tabú*. Esta sección contiene una breve descripción de los elementos básicos de esta heurística.

6.3.1 Características Generales

El algoritmo de "Búsqueda Tabú" (*tabu Search*) [40]-[41] fue introducido por F. Glover para la resolución de problemas de naturaleza combinatoria y ha sido aplicado en diferentes problemas tales como el reparto de tareas, coloreado de grafos, organización industrial [29], ... etc.

El algoritmo evita caer atrapado en un mínimo local mediante un mecanismo que emplea una lista de movimientos prohibidos. Mediante una adecuada elección del tamaño de esta lista es posible igualmente evitar la exploración cíclica de una misma zona. Otros mecanismos avanzados dotan al algoritmo de diferentes "visiones" del problema. Por ejemplo, es posible mantener una *memoria a corto plazo* que determina las nuevas direcciones de búsqueda y *memorias de medio y largo plazo* con las que se observan las zonas del espacio de configuraciones en que existe una alta probabilidad de localizar un mínimo absoluto.

Elementos Básicos de la Heurística Búsqueda Tabú

El algoritmo de *Búsqueda Tabú* en su presentación más simple está dotado de dos elementos básicos:

1. Un elemento que permite clasificar movimientos como prohibidos, dirigiendo la búsqueda hacia zonas no exploradas y
2. Un elemento que, por medio de un "olvido estratégico", permite liberar la búsqueda.

Estos dos conceptos se encuentran hábilmente combinados para generar un sencillo algoritmo que permite salir de los óptimos locales a los que nos conduciría un proceso puramente gradencial. En la figura 6-1 se muestra un esquema básico del algoritmo de *Búsqueda Tabú*.

El algoritmo introduce un elemento que es capaz de memorizar un número de movimientos previos. Estos movimientos son, en realidad, elementos del espacio de las configuraciones que ya han sido explorados. Los movimientos se almacenan en una lista denominada "Tabú" T, que es de un tamaño finito λ (la elección de

```
BUSQUEDA TABU()
{
    S0=Soptima= Punto de Partida Aleatorio;

    while( CRITERIO DE PARADA = false )
    {
        S= CALCULA_ENTORNO(S0);
        ∀ Si ∈ S
        {
            O= f_objetivo(Si);
            if( TABU_STATUS(Si)=false )
            {
                S0=Si;
                Ccorriente= Oi
                if( Ccorriente<Coptimo) Soptima=S0; Coptimo=Corriente;
                AÑADIR_LISTA_TABU( S0 );
            } else {
                if( Ccorriente<Coptimo) Soptima=S0; Coptimo=Corriente;
            }
        }
    }
}
```

Figura 6-1: Algoritmo básico Búsqueda Tabú

λ no parece que sea crítica para asegurar el buen comportameinto del algoritmo siempre que sea suficientemente grande. La elección de $\lambda = 7$ parece ser óptimo en la mayor parte de los casos). Hay dos aspectos básicos del algoritmo que merecen ser destacados:

1. El uso de T obliga a una búsqueda restringida por el espacio de las configuraciones.

2. El método no genera información acerca de si un óptimo es local o global. Unicamente devuelve el punto que encontró el mejor valor de la función de coste durante el proceso de exploración.

El algoritmo realiza una búsqueda aleatoria a través del espacio de las configuraciones, mediante un barrido local en torno a la solución actual. La existencia de la lista tabú impide visitar soluciones ya exploradas. El criterio de parada al que hace referencia la figura 6.1, puede llevarse a cabo de diferentes maneras, pero, en general, consiste en superar un cierto número de iteraciones sin mejorar la solución óptima.

La Lista Tabú

Un aspecto importante para el desarrollo del algoritmo es la posibilidad de que aparezcan comportamientos cíclicos. Existe una cierta probabilidad de que una zona del espacio de configuraciones ya visitada previamente lo sea de nuevo tras λ iteraciones. La probabilidad de que esto ocurra está relacionada inversamente con el propio valor de λ. Esto puede dar a entender que λ ha de ser muy grande. Sin embargo existen otros factores en contra de un λ elevado, como son el hecho de que un λ grande puede restringir excesivamente el espacio de búsqueda en cierto tipo de problemas y aumentar la complejidad del algoritmo.

Otro aspecto de importancia en el desarrollo de la búsqueda tabú es la forma en que se contruye la lista tabú. La forma más simple consiste en almacenar los últimos movimientos ya visitados e impedir que aparezcan de nuevo en la siguiente iteración. Si el espacio de búsqueda es muy denso (en el caso extremo, si es continuo) ésta no parece ser una buena elección para la lista tabú, ya que la probabilidad de se visite *exactamente* el mismo punto es muy baja (o cero en el caso continuo), y la lista tendría poca utilidad. Por tanto, más que movimientos en sí mismos, la lista debe almacenar características de los movimientos o áreas de búsqueda que no deben volver a visitarse.

Niveles de Aspiración y Memorias a Corto y Largo Plazo

Un concepto interesante que se puede aplicar a esta heurística es el de *Nivel de Aspiración*. Consideremos una función $A(s, x)$, donde s es el movimiento y x el

punto bajo estudio. Un nivel de aspiración se alcanza en s si:

$$c(s(x)) < A(s,x) \tag{6.4}$$

Si una determinada solución exploratoria s alcanza un nivel de aspiración, entonces ésta se adopta como nueva solución actual, independientemente de su estado tabú. El papel que juega el *Nivel de Aspiración* es olvidar el *status Tabú* de un movimiento y adoptarlo bajo ciertas condiciones ventajosas. Este mecanismo permite también actuar sobre los posibles comportamientos cíclicos. Como inconveniente, será necesario el evaluar el coste de todas las soluciones exploratorias, sin tener en cuenta previamente su carácter *Tabú*. El propio *Nivel de Aspiración* puede "aprender" y actualizarse cada vez que se cumple.

Es de destacar el hecho de que un ser humano, cuando razona suele tener en consideración diversas pautas de comportamiento que la psicología denomina *memoria de corto plazo* y *memoria de largo plazo.*

Este inciso viene a colación por lo siguiente: un *Nivel de Aspiración* no es más que una norma de memoria de corto plazo al ser una norma de visión muy puntual.

Sin embargo esta idea permite sugerir otros tipos de funciones más complejas que juegan el papel de memoria a *medio* y *largo plazo.* La primera es un mecanismo que intensifica la búsqueda sobre regiones que en las que se sospecha la presencia de un óptimo, y la segunda es un mecanismo llamado de "diversificación" con el que se reinicia la búsqueda desde puntos elegidos aleatoriamente. Esta idea sugiere la introducción de una *Lista Tabú de Largo Plazo* en la que se consideren cierto atributos parciales propiciados por cada búsqueda iniciada. Volveremos a estas ideas más adelante.

6.3.2 Comparación *Búsqueda Tabú* frente a *Enfriamiento Simulado*

Se ha llegado a sugerir [4] que la técnica de *Búqueda Tabú* es una versión determinista del *Enfriamiento Simulado* en un sentido amplio. Se ha hecho mención a la naturaleza Markoviana del equilibrio a cada temperatura. Está claro que existen

elementos de gran potencia y elementos de debilidad en cada uno de los algoritmos, por ello es posible dotar de ciertas características de la *Búsqueda Tabú* al *Enfriamiento Simulado* y viceversa. También se encuentran realizaciones que combinan ambos algoritmos a lo largo de toda la optimización, tomando el control del proceso el algoritmo más eficiente en cada caso. Así se puede responsabilizar de los primeros movimientos (hasta llegar a una temperatura relativamente baja) al *Enfriamiento Simulado*, para luego encomendar la fase final del algoritmo a la *Búsqueda Tabú*. Esta aproximación va en la misma línea del algoritmo combinado enfriamiento simulado–gradiente, propuesto en el primer apartado de este capítulo.

Por otra parte, la mayoría de las experiencias de aplicación de *Búsqueda Tabú* a los diversos problemas, certifican la menor cantidad de iteraciones que éste necesita para dar buenas soluciones. No todo son ventajas para el algoritmo de búsqueda tabú, el algoritmo de enfriamiento simulado garantiza alcanzar un óptimo global con probabilidad 1 si se escogen adecuadamente los parámetros de enfriamiento.

6.4 Optimización Mediante Búsqueda Tabú

En esta sección se presenta una aplicación del algoritmo de búsqueda tabú a la síntesis de celdas analógicas. Para evaluar la calidad de la solución propuesta, emplearemos el problema de optimización de amplificadores operacionales ya presentado en el capítulo anterior. Emplearemos la misma rutina base de adimensionalización de variables y la misma definición de la función de costes presentada en el capítulo 5.

Por ello tan sólo discutiremos aquí la parte del proceso de optimización que lleva a cabo el algoritmo de búsqueda tabú.

6.4.1 Generación de Movimientos

La generación de movimientos está basada en la expresión (5.6) de la sección 5.4.3. Sin embargo se han realizado una serie de consideraciones adicionales. Dado

que la *Búsqueda Tabú* precisa de un conjunto de puntos para trabajar, se ha adoptado el criterio de generar una nube de puntos alrededor de la *Solución Actual* de una manera lo más uniforme posible. Para evitar que el algoritmo explore soluciones del espacio que estén muy próximas, no todos los puntos generados aleatoriamente en torno a la solución actual son aceptados como movimientos exploratorios. El criterio de aceptación del movimiento s en el conjunto de movimientos exploratorios viene definido por la expresión:

$$d(sa, s) > d(sa, s) + d(s, s_i), \quad \forall s_i \in S \tag{6.5}$$

donde sa es la llamada *Solución Actual*, a partir de cual se obtiene el conjunto de soluciones S. s_i es el conjunto de soluciones ya seleccionadas para S. Cuando se introduce una solución nueva dentro del conjunto S, se repasa este conjunto para desechar aquellas soluciones generadas anteriormente que dejan de ser compatibles con el criterio (6.5). La función $d(x, y)$ obtiene la distancia al cuadrado entre los puntos x e y. Al ser un espacio de dimensión 4 ó 5, en realidad sa y s_i están acotando una región del espacio determinada por una hiper–esfera.

La figura 6-2 presenta un esquema gráfico para un espacio bidimensional, del criterio de aceptación de soluciones dentro del conjunto S. Nótese que aún no se ha aplicado ninguna evaluación de la función objetivo. De esta manera la exploración del espacio de soluciones se hace más eficiente al no tener que hacer llamadas a la función objetivo en puntos que se encuentran muy próximos.

6.4.2 Mecanismo de Aceptación de la Nueva Solución Actual

Una vez generado el conjunto S, se calcula el valor de la función objetivo a cada una de las soluciones en S. Nótese que para el cálculo de la función objetivo en cada punto de S, se evalúan los parámetros que informan sobre el comportamiento de la celda generada. Nótese también que la evaluación de la función de coste se realiza antes de comprobar si la nueva solución es o nó tabú. Ello es debido a que es preciso evaluar todos los puntos de S para poder admitir los niveles de aspiración. Si no se cumple la condición del nivel de aspiración (que se estudiará posteriormente) en ninguno de los puntos de S, entonces la nueva solución aceptada será aquella que:

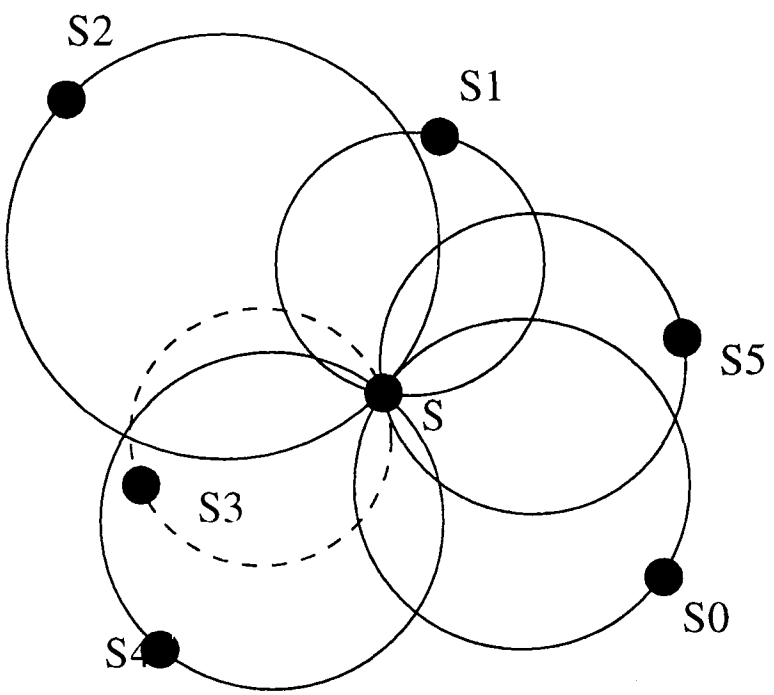


Figura 6-2: Criterio de Aceptación de la rutina de generación

1. No es tabú.

2. Tiene menor coste de entre todas las soluciones de S.

6.4.3 La lista Tabú

En nuestra aplicación, la lista tabú es un registro *FIFO* que contiene los últimos punto visitados y evaluados. Esta lista (de tamaño $\lambda = 7$) tiene por objeto impedir el retroceso de la *Solución Actual* y que sean de nuevo visitadas soluciones previamente exploradas. Para ello, lo que se propone es un criterio de asignación de áreas, similar al de la rutina de generación de movimientos aleatorios anteriomente descrita. Cada dos puntos consecutivos de la lista tabú delimitan una región del espacio de búsqueda, declarada *tabú*. De esta forma la lista, representada por una secuencia de puntos, determina un conjunto de regiones *tabú* (circulares en el espacio bidimensional) que han de declarar rechazados a los puntos del conjunto S que

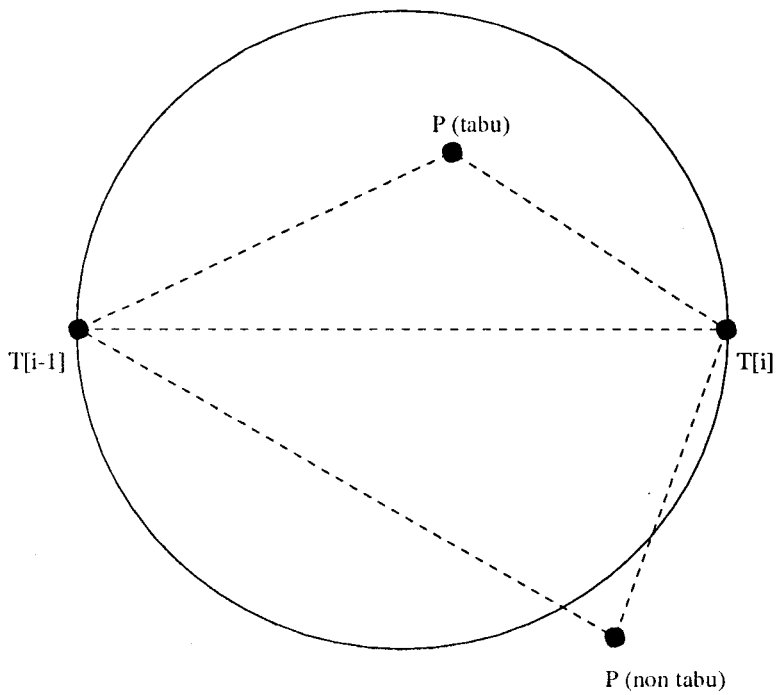


Figura 6-3: Declaración del estado Tabú

pertenezcan a ella. El criterio de rechazo de puntos es:

$$d(T_i, T_{i+1}) < d(T_i, s) + d(s, T_{i+1}) \tag{6.6}$$

La figura 6-3 representa un esquema en dos dimensiones del criterio expresado matemáticamente en la ecuación (6.6). La figura 6-4 muestra una representación bidimensional del área cubierta por la lista tabú. En esta figura, $T[i]$ es el *i–ésimo* punto almacenado en la lista tabú.

6.4.4 Salida de Mínimos Locales

Una de las ventajas mas interesantes a la hora de aplicar el algoritmo *Búsqueda Tabú* es su capacidad para salir de óptimos locales. En la figura 6-5 se muestra de una manera gráfica el mecanismo de escape de mínimos locales en el caso bidimensional. En esta figura, $T[t-1]$, $T[t-2]$ y $T[t-3]$ son los últimos movimientos aceptados incluidos en la lista tabú, SA es la solución actual y $W1$

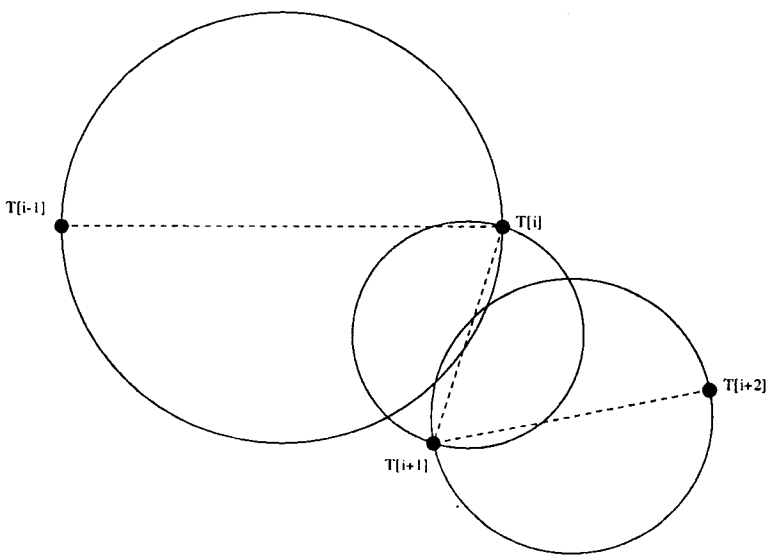


Figura 6-4: Cobertura del Aŕea de Búqueda por la Lista Tabú

y $W2$ son movimientos exploratorios alrededor de SA. $W2$ será seleccionada como nueva solución actual, a pesar de su mayor coste, debido a que $W1$ está en el interior del área tabú y no es globalmente óptima. Siguiendo un esquema simular, U y V serán los siguientes movimientos seleccionados.

Cabe la posibilidad de que nos encontremos en una situación en la que el tamaño de la lista tabú no cubra el área suficiente para escapar del óptimo local. En nuestra aplicación se han realizado ensayos con un tamaño mayor de la lista, sin una mejora sustancial de las soluciones.

6.4.5 Niveles de Aspiración

Uno de los elementos más característicos del Algoritmo *Búsqueda Tabú* son los llamados *Niveles de Aspiración*. El objeto de estos elementos no es más que el de aceptar, bajo ciertas condiciones, soluciones que debieran ser rechazadas por su estado tabú. Este mecanismo se ha mostrado muy indicado para dar mayor flexibilidad al algoritmo. En nuestro caso hemos escogido la forma más simple de nivel de aspiración. Un movimiento exploratorio W_i de S se acepta si $coste(W_i) <$

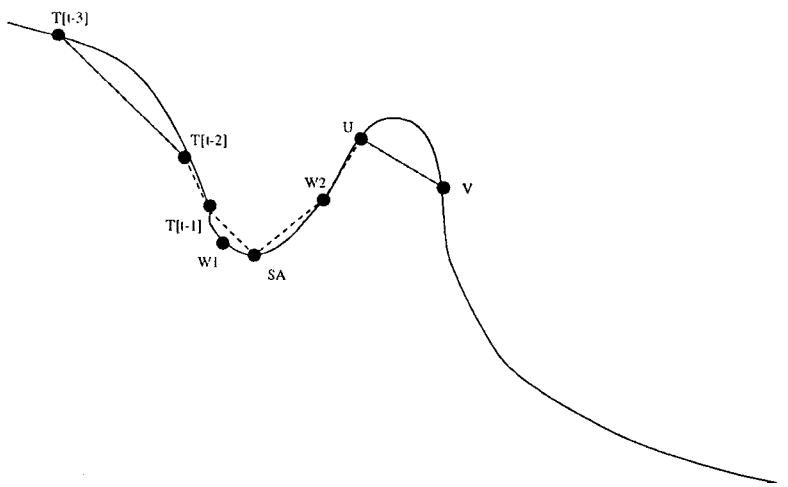


Figura 6-5: Mecanismo de Salida de Mínimos Locales

coste_optimo, independientemente de su estado tabú. En la figura 6-6 se muestra un ejemplo. En esta figura, $T[t-1]$ es el último movimiento aceptado en la lista tabú y SA la solución actual. $W1$ a $W5$ son movimientos exploratorios. Nótese que $coste(W2)$ es menor que el coste óptimo. De acuerdo con nuestra definición de nivel de aspiración, $W2$ será seleccionado como nueva solución actual, a pesar de su estado tabú, y el coste óptimo será actualizado para futuros movimientos.

6.4.6 Memoria a Medio Plazo

La memoria a medio plazo consiste en un conjunto de reglas heurísticas que proporcionan una visión menos local del problema. Estas reglas dirigen la búsqueda sobre aquellas zonas donde aparentemente puede haber un mínimo global. Es nuestro problema hemos considerado dos elementos como *Memoria a Medio Plazo*:

1. Un elemento que recupera la búsqueda sobre el último punto encontrado, y que permite intensificar la búsqueda alli donde mejor resultado se obtuvo. La regla seguida es la siguiente: si al cabo de N iteraciones (en nuestro caso $N = 50$) no se encuentra un óptimo mejor, reiniciar la búsqueda a partir del último óptimo encontrado.

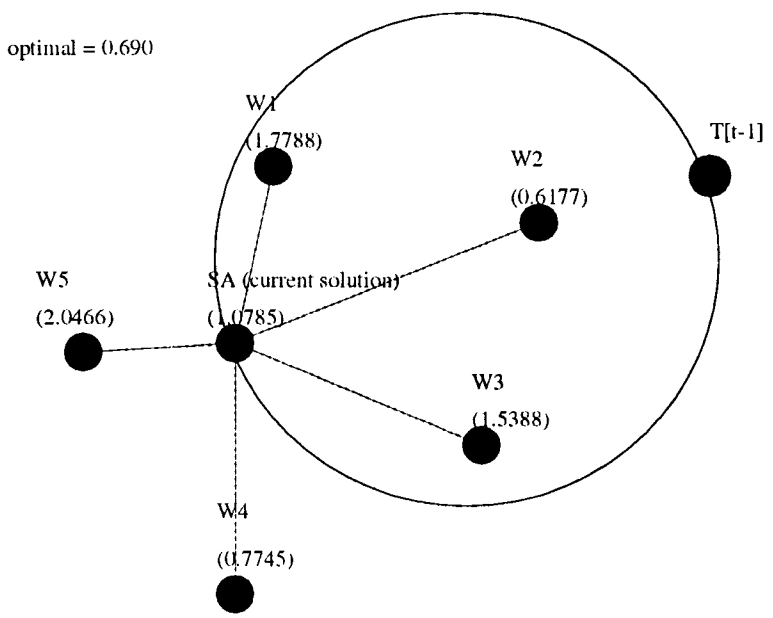


Figura 6-6: Nivel de Aspiración

2. Un elemento que cada vez que encuentra un óptimo hace que los movimientos generados sean cercanos a ese óptimo, y amplían el entorno de búsqueda en caso de no encontrar un nuevo óptimo. En nuestro caso, este concepto se ha introducido mediante una modificación dinámica del coeficiente *coef* que gobierna la amplitud de la búsqueda. Este coeficiente se modifica en la forma indicada en la sección 5.4.3.

Existen consideraciones de *Memoria a Largo Plazo* que no hemos introducido en esta primera realización de la técnica. Básicamente consisten en generar aleatoriamente distintos puntos de partida y correr el algoritmo desde cada uno de ellos. Una vez que se han realizado varios intentos, se decide un último intento sobre un punto predeterminado, bien con las soluciones de partida generadas, bien desde los puntos finales obtenidos en intentos anteriores.

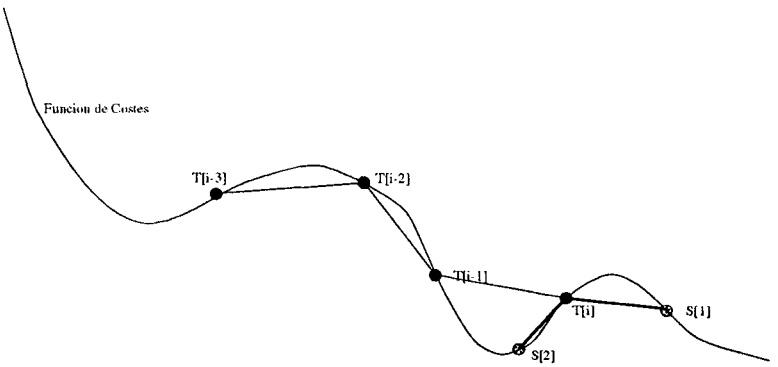


Figura 6-7: Actuación de los Niveles de Aspiración

6.4.7 Criterio de Parada

El algoritmo original de Glover no posee ningún criterio de parada definido, tan sólo se limita a indicar un número máximo de iteraciones. Sin embargo se puede establecer un criterio de parada relativamente simple: se debe detener el algoritmo cuando al cabo de un cierto número M de iteraciones la solución óptima no ha mejorado.

Nuestra realización limita el número máximo de iteraciones a 600 y limita el número de iteraciones sin mejorar el coste óptimo a 150.

6.5 Evaluación de la Función Objetivo

En el plantemiento inicial del algoritmo el cálculo de la función de coste estaba determinado por un modelo analítico de la celda del cual se extraían las propiedades mediante fórmulas aproximadas [20]. En la tabla 6.2 vimos que los resultados de una optimización que utilizan un modelo aproximado, difieren sobre todo, en el ancho de banda, con respecto a los valores obtenidos de la simulación SPICE.

Como se demostrará más adelante, los resultados del algoritmo de búsqueda tabú muestran una mejora muy importante en cuando al número de iteraciones respecto al *Enfriamiento Simulado*. Esto significa que los tiempos de computación

del algoritmo son lo suficientemente cortos como para integrar dentro del algoritmo, el cálculo de la función objetivo mediante simulación SPICE, dando por ello un resultado de simulación más exacto. La figura 6-8 presenta un esquema de cómo se evalúan los parámetros deseados a partir de una salida SPICE.

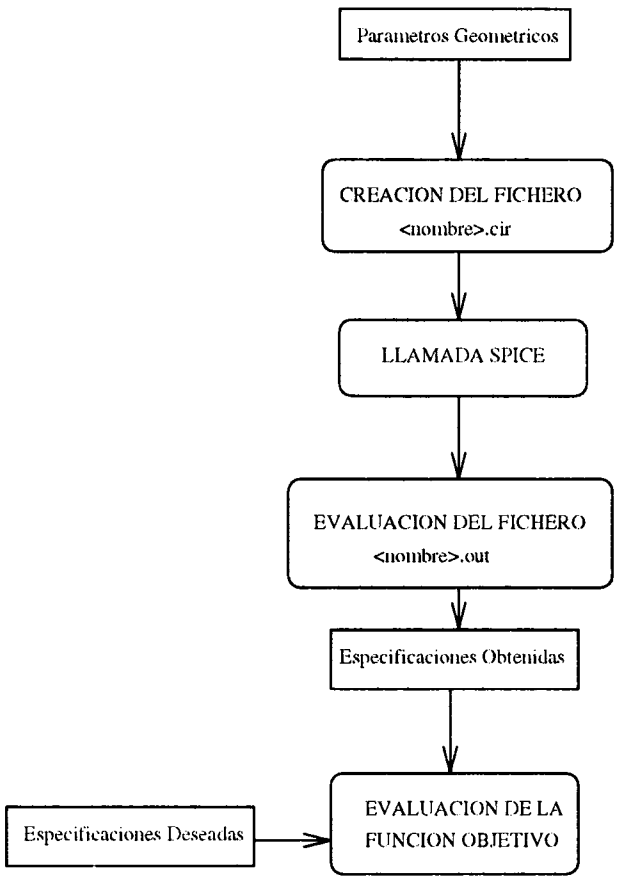


Figura 6-8: Esquema de la Evaluación de la Función Objetivo

6.6 Pruebas Realizadas con Modelos Aproximados

Todas las consideraciones mencionadas hasta este momento han sido integradas en el algoritmo de la figura 6-9. La tabla 6.3 muestra los resultados de 10 optimizaciones escogidas al azar de entre un conjunto de especificaciones típicas. Es

de destacar la propiedad ya mencionada de poder obtener soluciones de buena calidad con un número reducido de iteraciones. Los tiempos de cálculo del algoritmo de búsuqeda tabú es de unos pocos segundos en una estación de trabajo SUN–SPARC–10.

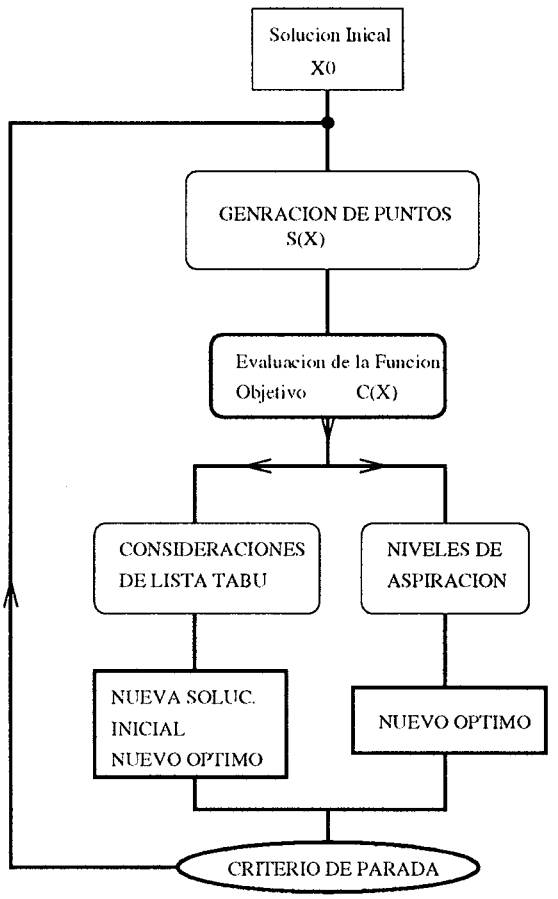


Figura 6-9: Esquema Global del Algoritmo propuesto

6.7 Pruebas Realizadas con Modelos SPICE

Dado el reducido número de iteraciones que precisan las técnicas de búsqueda tabú, es posible emplear SPICE para evaluar el comportamiento de la celda. La ta-

bla 6.4 muestra los resultados devueltos por la optimización empleando SPICE para evaluar la función objetivo en cada iteración. La tabla 6.5 muestra el valor medio, el valor mínimo y la desviación típica obtenida en las pruebas, con un conjunto de especificaciones típicas cada una repetida 10 veces. Se emplea SPICE para calcular la función de coste. La disparidad de valores obtenida, es debido a que la función objetivo se vuelve muy mal condicionada cuando se emplea SPICE para calcularla. El algoritmo, por otra parte, está ajustado para optimizar la función objetivo cuando se emplea un modelo simplificado de la celda, en cuyo caso la función objetivo es suave y tiene pocos mínimos.

Se han intentado algunas mejoras de diseño, como el mantenimiento de estadísticas para llevar un cierto control del proceso pero no hemos conseguido con ello una mejora sustancial. Se ha comprobado también que el tamaño de la lista $\lambda = 7$ proporciona buenos resultados, además de verificar las propiedades de la *Búsqueda Tabú* en el sentido de ser capaz de escapar de mínimos locales.

6.8 Algoritmo combinado basado en la *Búsqueda Tabú*

Una de las críticas que de manera más inmediata se pueden alegar al comportamiento de la *Búsqueda Tabú* observado en la tabla 6.4 es el hecho que no existe una medida explícita ni analítica de la cercanía al óptimo absoluto de la función objetivo. Es más, existe una probabilidad bastante alta de que el mínimo encontrado no sea el mejor, e incluso que el algoritmo haya explorado entornos cercanos a ese mínimo sin detectarlo debido a un excesivo paso de búsqueda. Para mejorar la confianza en la solución se ha diseñado una técnica combinada de búsqueda, en la que se aplica un algoritmo puramente gradencial a la búsqueda local, reservando la estrategia de selección del espacio, al algoritmo de búsqueda tabú. La idea básica se sugiere en 6.4.3, donde se expone la posibilidad de crear listas *Tabú* a largo plazo, que gobiernen el proceso de búsqueda, dejando a una técnica gradencial la responsabilidad de encontrar el óptimo (se podría decir que la memoria a corto plazo de la *Búsqueda Tabú* es sustituida por otra técnica más eficiente).

6.8.1 Descripción del Algoritmo Combinado

La técnica gradencial empleada es el método de Powell recogido en [39]. En ella se obtienen las direcciones de búsqueda de forma que se realizan muy pocos cálculos para ello.

La *Búsqueda Tabú* tiene como misión la selección inteligente de los puntos de partida de la rutina gradencial. Es decir, tenemos, por así decirlo, dos visiones del problema de búsqueda. La visión de corto plazo con un "paso" corto y la de largo plazo, con una visión más general del problema. Debido a que se han respetado los criterios de adimensionalización de las variables, el espacio de variabilidad de w_k sigue siendo $[0, \pi]$. Por otra parte, para reducir el número de llamadas a *SPICE* se ha limitado el número de pasos del algoritmo de *Búsqueda Tabú* a 20. La rutina del método del *Gradiente Conjugado* emplea en una optimización una media de 80 llamadas a la evaluación de la función objetivo. Esto implica que, se emplean del orden de unas 1600 llamadas a SPICE.

6.8.2 Desarrollo de la Técnica Combinada

Para que la búsqueda sea eficiente desde el punto de vista de la calidad de las soluciones precisamos que:

- El número de pruebas Tabú sea grande, ya que exploraremos más exahustivamente toda el área.
- Los pasos de búsqueda sean cortos, ya que no obviaremos ningún punto de descenso.
- La rutina de gradiente descienda lo más lentamente posible para encontrar el óptimo de todo el proceso, y no un punto localmente cercano.

Todas estas consideraciones van a favor de unos tiempos de computación grandes, es decir, muchas llamadas a SPICE. La solución de compromiso es la de fijar el coeficiente de búsqueda global en un valor que permita explorar bien todo el espacio (en nuestro caso es de $\pi/2$) y el de búsqueda en detalle, un paso de gradiente que varía entre $\pi/2$ y $\pi/20$ con un error máximo estimado de un 1%.

Para involucrar al algoritmo gradencial en la propia búsqueda se asocia el óptimo obtenido con cada uno de los respectivos puntos de partida de la lista *Tabú* y estos son los valores que se barajan en la lista para evaluar su estatus y explorar el mejor resultado. En la tabla 6.5 se muestra una relación de diferentes síntesis. Se indican resultados estadísticos relativos a la confianza de las soluciones encontradas. Se han realizado diversas pruebas sobre tres especificaciones diferentes en cada una de las topogías. Se muestran, en cada caso, el menor de los óptimos encontrados, la media de los mismos y la desviación típica. También se presentan la media de iteraciones y el tiempo medio de cada ejemplo. Cada iteración supone un análisis del circuito por parte de SPICE. Por último es preciso comentar que, al haberse escogido como error máximo admisible del algoritmo gradencial el valor 0.01, los valores obtenidos de la desviación típica entre las diferentes pruebas, nos permiten garantizar un buen resultado a la hora de integrar el algoritmo en un paquete de síntesis analógica completo.

Los resultados obtenidos mejoran la confianza en las soluciones, desde el punto de vista de que los valores finales sobre las mismas especificaciones han devuelto resultados muy similares entre sí. Sin embargo, hay una pérdida apreciable en tiempos de computación del algoritmo, cosa bastante evidente si se tiene en cuenta que el algoritmo gradencial consume entre 50 y 120 llamadas al simulador.

6.9 Conclusiones

La síntesis de celdas analógicas es un serio problema de optimización en un espacio n–dimensional continuo. La complejidad del mismo y las características especiales del problema de sintetizar celdas analógicas, aconsejan la aplicación de técnicas de búsqueda aleatoria lo suficientemente eficientes como para que se pueda emplear como evaluador de la función objetivo un simulador en vez de una fórmula que sea consecuencia de un análisis aproximado de la celda en cuestión, y con ello dar una mayor garantía de los resultados. Se propone una técnica basada en el *Enfriamiento Simulado* combinada de un modo interactivo con un algoritmo gradencial. Esta técnica reduce el tiempo de trabajo del enfriamiento simulado, pero

aún persiste un número de iteraciones muy elevado para pensar en introducir un simulador.

Posteriormente se ha aplicado la técnica *Búsqueda Tabú*, más novedosa, que también es original del campo de la resolución de problemas combinatorios y que se propone con una aproximación al espacio continuo. Las soluciones obtenidas mediante la aplicación de este algoritmo son de una calidad similar a las que proporciona el *Enfriamiento Simulado*. Sin embargo, dado que el control sobre el espacio de búsqueda es escaso, el conjunto de soluciones que genera este algoritmo, aún siendo aceptables, presenta una cierta dispersión en los resultados finales, lo que hace pensar en la posibilidad de emplear algoritmos combinados.

Dado que la *Búsqueda Tabú* posee ciertos *campos de visión* más o menos amplios se ha optado por una técnica combinada, donde la responsabilidad de recorrer el espacio se cede a la *Búsqueda Tabú* y la de encontrar el mejor mínimo local se cede a un algoritmo gradencial. Los resultados obtenidos responden a lo esperado. El número de iteraciones *Tabú* es reducido, de manera que penalizamos ligeramente el número de iteraciones, pero ganamos en confianza en la solución final obtenida.

OTA:	Simulated Annealing		Tabu Search	
	Coste Final	Iteraciones	Coste Encontrado	Iteraciones
1	0.0919	7140	0.0916	274
2	0.1579	8456	0.1645	204
3	0.1108	8427	0.0891	232
4	0.1627	5532	0.1805	191
5	0.0918	5682	0.0893	207
6	0.1337	12491	0.1402	161
7	0.0911	11624	0.0882	204
8	0.1430	4919	0.1334	189
9	0.0916	8049	0.0899	207
10	0.1243	7242	0.1246	245
BTS:	Simulated Annealing		Tabu Search	
	Coste Final	Iteraciones	Coste Encontrado	Iteraciones
1	0.2646	6169	0.2619	189
2	0.2650	9310	0.2637	254
3	0.3688	14670	0.3880	217
4	0.3766	13838	0.3788	311
5	0.2245	9521	0.2273	393
6	0.2320	7834	0.2382	178
7	0.4359	10355	0.4423	191
8	0.4496	9392	0.4524	232
9	0.2348	13686	0.2252	186
10	0.2236	14120	0.2229	232

Tabla 6.3: Resultados sobre el Modelo Simplificado (de ahí las discrepancias en los valores de la función objetivo) (SA versus TS).

	BTS		OTA		FCO	
Especificación	Deseada	Obtenida	Deseada	Obtenida	Deseada	Obtenida
Iteraciones		158		87		238
Ganancia DC(dB)	80	104.8	60	75.15	100	144.3
Frec. Gan. Unidad(MHz)	1	2.18	2.5	3.08	1	1.7
Márgen de Fase (deg)	60	89	60	75	60	84
Area Activa(μm^2)	< 5000	4840	< 10000	7010	< 5000	3720
Cap. (μF)	10		10		10	

Tabla 6.4: Resultados del algoritmo *Búsqueda Tabú* sobre especificaciones típicas utilizando modelos SPICE.

	Mínimo	Media	Desv. Típ.	n. iter	tiempo (minut.)
Diseño 1 (OTA)	1.19646	1.2369	0.028	2525	59
Diseño 2 (OTA)	1.6353	1.6133	0.01629	2542	59
Diseño 3 (OTA)	1.91293	1.877	0.031	2689	63
Diseño 4 (BTS)	1.8801	1.8805	0.0003	2818	66
Diseño 5 (BTS)	3.0722	3.078	0.0036	2949	49
Diseño 6 (BTS)	2.5335	2.54032	0.0021	2670	44
Diseño 7 (FCO)	0.9388	0.93558	0.0017	1864	31
Diseño 8 (FCO)	1.28267	1.2779	0.0077	1568	26
Diseño 9 (FCO)	1.09611	1.0944	0.0022	1444	24

Tabla 6.5: Resultados basados en las diversas pruebas realizadas con el algoritmo mixto tabú–gradiente

Capítulo 7

Conclusiones

Esta tesis presenta un conjunto de contribuciones en el campo del diseño asistido de circuitos electrónicos. El trabajo de esta tesis se ha centrado en el problema de la *colocación* de celdas estándar en circuitos $VLSI$ y en la optimización de celdas analógicas. Las principales aportaciones que aquí se presentan son:

1. Una síntesis de la bibliografía existente relativa a la optimización del diseño de circuitos electrónicos.

2. Un algoritmo de partición de circuitos de gran dimensión mediante compactación previa. Este algoritmo ha sido empleado en un paquete de *colocación* de celdas en diseño VLSI.

3. Un estudio de la aplicación de la técnica de enfriamiento simulado al problema de la colocación de celdas. Fruto de esta estudio es el desarrollo de un paquete de rutinas que utiliza un método original para la colocación de celdas consistente en una posición previa obtenida por partición recursiva del circuito, cuyos resultados se emplean como solución inicial del problema de la *colocación*. El algoritmo incluye la determinación de una temperatura inicial que permite alcanzar un óptimo global sin deteriorar los beneficiosos resultados de la partición inicial. Los resultados obtenidos demuestran que, con esta aproximación se consiguen soluciones mejores (en la mayor parte de los casos) a las obtenidas por TimberWolfSC, y en tiempos muy inferiores.

4. La aplicación de técnicas de búsqueda aleatoria al problema de la optimización de celdas analógicas. En este sentido, se presenta un algoritmo combinado con una técnica gradencial que permite acortar los tiempos de optimización manteniéndo los excelentes resultados del enfriamiento simulado.

5. El desarrollo de un método original de optimización de celdas analógicas que emplea un algoritmo de búsqueda tabú. Los resultados de este método, cuando se aplican a modelos aproximados de celdas, alcanzan soluciones óptimas en tan sólo unas pocas centenares de iteraciones, lo que permite sustituir el modelo aproximado de la celda por una simulación SPICE en cada iteración.

6. Finalmente, para reducir la dispersión de los resultados finales cuando se emplea SPICE para evaluar la función de coste en cada iteración, se propone un método combinado de búsqueda tabú y gradiente conjugado. En este método, el algoritmo de búsqueda tabú se emplea como un método inteligente de selección de zonas de búsqueda, mientras que en cada iteración tabú se emplea el método del gradiente conjugado para localizar el óptimo de esa zona. Los resultados obtenidos con este método dan una dispersión mínima de los resultados, si bien a costa de un incremento en el número total de llamadas a SPICE (que oscila entre 1500 y 3500, dependiendo del modelo).

La complejidad creciente de los circuitos ASIC y la extensión de su diseño a un elevado número de potenciales usuarios hace que el desarrollo de nuevas herramientas de diseño asistido de circuitos electrónicos sea una necesidad creciente. Esto nos anima a proseguir los trabajos aquí presentados proponiendo las siguientes líneas de investigación futura:

1. Desarrollo de nuevos algoritmos de partición de circuitos que incluyan en la función de coste características del problema de la *colocación.*

2. Resolución jerárquica del problema de la *colocación* de celdas mediante compactación de celdas, posicionamiento de bloques y descompactación. Este método parece muy interesante en circuitos de gran dimensión (> 1000 celdas).

3. Empleo de nuevas técnicas en la resolución del problema de la *colocación*. Por ejemplo, algoritmos de búsqueda tabú.

4. Estudio de los resultados de las técnicas de optimización de celdas analógicas en otras celdas como comparadores, filtros, convertidores, etc.

5. Introducir en el proceso de optimización de celdas analógicas, la geometría de la celda, obtenida mediante un generador automático y un paquete de extracción de parámetros de la celda.

Bibliografía

[1] M.A.Aguirre y A.Torralba. "Algoritmo combinado de partición y enfriamiento simulado para el "placement" de circuitos VLSI". *Actas VIII Congreso de Diseño de Circuitos Integrados*, Málaga, Nov. 1993, pp 33–36.

[2] M.A.Aguirre, J.Chávez, A.Torralba and L.G.Franquelo. "Analog design optimization by means of a tabu search approach". Accepted for presentation at the *IEEE Int. Symp. on CAS*, London, June 1994.

[3] A.V.Aho, J.E.Hopcroft and J.D.Ullman. *The design and analysis of computer algorithms*. Addison–Wesley series in computer science and engineering. Addison–Wesley, Reading, MA, 1974.

[4] S.Areibi and A.Vanelli. "Circuit partitioning using a tabu search approach". *Proc. of the IEEE Int. Symp. on CAS*, 1993, pp 1643–1646.

[5] E.R.Barnes, A.Vannelli and J.Q.Walker. "A new heuristic for partitioning the nodes of a graph". *SIAM Journal of Disc. Math.*, vol. 1, no. 3, Aug. 1988.

[6] F.J.Belaza. "Fundamentos del análisis y diseño de circuitos ayudado por ordenador". *Serv. Pub. E.T.S. Ingenieros de Telecomunicación*, Madrid, 1981.

[7] J.P.Blanks. "Near optimal placement using a quadratic objective function". *Proc. of the ACM/IEEE 22nd Design Automation Conf.*, pp. 609–615, 1985.

[8] R.B.Boppana. "Eigenvalues and graph bisection: an average case analysis". *Proc. of the IEEE 28th Ann. Symp. on Foundations in Computer Science*, pp. 280–285, 1987.

[9] D.E.van den Bout and T.K.Miller. "Graph partitioning using annealed neural networks". *IEEE Trans. on Neural Networks*, vol. 1, np. 2, 1990.

[10] R.J.Bowman and D.J.Lane. "A knowledge–based system for analog integrated circuit design". *Proc. of the IEEE Int. Conf. Computer–Aided Design*, 1985.

[11] D.R.Brasen and M.L.Bushnell. "Mhertz: A new optimization algorithm for flooplanning and global routing". *Proc. of the ACM/IEEE 27th Design Automation Conference*, 1990, pp. 107–110.

[12] R.K.Brayton, G.D.Hatchel and A.L.Sangiovanni–Vincentelli. "A suvey of optimization techniques for integrated circuit design". *Proc. IEEE*, vol 69, pp 1334–1362, Oct. 1981.

[13] M.Breuer. "A class of min–cut placement algorithms". *Proc. of the 14th Design Automation Conf*, pp. 284–290, 1977.

[14] M.Breuer. "Min–cut placement". *J. Design Automation and Fault Tolerant Computing*, vol. 1, no.4, pp. 343–362, Oct. 1977.

[15] T.N.Bui, S.Chaudri, F.T.Leighton and M.Sipser. "Graph bisection algorithms with good average behavior". *Combinatorica*, vol. 7, no. 2, pp. 171–191, 1987.

[16] T.Bui, C.Heigham, C.Jones and T.Leighton. "Improving the performance of the Kernighan–Lin and simulated annealing graph bisection algorithms". *Proc. of the 26th ACM/IEEE Design Automation Conf.*, pp. 775–778, 1989.

[17] A.Cassotto, F.Romeo and A.L.Sangiovanni–Vincentelli. "A parallel simulated annealing algorithm for the plcement of macro–cells". *IEEE Trans. on Computer–Aided Design*, no. 5, Sept 1987, pp 838–1987.

[18] R.I.Chang and P.Y.Hsiao. "Arbitrarily sized cell placement by self–organizing neural networks". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2043–2046, Chicago, May 1993.

[19] A.Chatterjee and R.Hartley. "A new simultaneous circuit partitioning and chip placement approach based on simulated annealing". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp 36–39, 1990.

[20] J.Chávez, M.A.Aguirre and A.Torralba. "Analog design optimization: A case study". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2083–2085, Chicago, May 1993.

[21] J.Chávez, M.A.Aguirre, A.Torralba y L.G.Franquelo. "Diseño automático de celdas analógicas. Aplicación al diseño de amplificadores operacionales". *Actas del VIII Congreso de Diseño de Circuitos Integrados*, pp 217–221, Málaga, Nov. 1993.

[22] J.Chávez, A.Torralba and L.G.Franquelo."A fuzzy–logic based tool for topology selection in analog synthesis". *Accepted for presentation at the IEEE Int. Symp. on CAS*, London, May 1994.

[23] J.Chávez. "Nuevas aportaciones al diseño óptimo de celdas analógicas ".*Tesis Doctoral en preparación*, Dpto. de Ingeniería de Sistemas y Automática, ESII, Sevilla, 1994.

[24] C.K.Cheng and E.S.Kuh. "Module placement based on resistive network optimization'. *IEEE Trans. on Computer–Aided Design*, vo. 3, pp. 218–225, July 1984.

[25] C.K.Cheng. "The optimal circuit decompositions using network flow formulations". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2650–2653, New Orleans, May 1990.

[26] C.K.Cheng. "An improved two–way partitioning algorithm with stable performance". *IEEE Trans. on Computer–Aided Design*, vo. 10, no. 12, pp. 1502–1511, Dec. 1991.

[27] J.P.Cohoonaud and W.D.Paris. "Genetic Placement". *IEEE Trans. on Computer–Aided Design*, vol. 6, no. 6, pp. 956–964, Nov. 1987.

[28] M.G.R.Degrauwe, O.Nys, E.Dijkstra, J.Rijmenants, S.Bitz, B.L.A.G.Goffart, E.A.Vittoz, S. Cserveny, C.Meixenberger, C.V. det Stappen and H.J.Oguey. "IDAC: An interactive design tool for analog CMOS circuits". *IEEE J. Solid–State Circuits*, vol. 22, no. 6, pp. 1106–1117, Dec. 1987.

[29] B. Díaz Fernández. *Control dinámico de la producción en entornos industriales mediante la aplicación de heurísticas basadas en recocido simulado y búsqueda tabú.* Tesis Doctoral, Oviedo, 1991.

[30] A.E.Dunlop and B.W.Kernighan. "A procedure for placement of standard–cell VLSI circuits". *IEEE Trans. on Computer–Aided Design*, Vol 4, No. 1, pp 92–98, Jan. 1985.

[31] M. Edahiro, T. Yoshimura. "New placement and global routing algorithm for standard cell layouts". *Proc. of the 27th ACM/IEEE Design Automation Conference, pp. 107–110*, 1990.

[32] F.El–Turkey and E.E.Perry. "BLADES: An artificial intelligence approach to analog circuit design". *IEEE Trans. on Computer–Aided Design*, vol. CAD–8, no. , pp. 680–6926, June 1989.

[33] E.Fernández Camacho. *Indentificación de sistemas lineales discretos no lineales. Ajuste y estimación de parámetros.* Tesis Doctoral, ESII, Sevilla, Feb. 1977.

[34] M.Fiduccia and R.M.Mattheyses. "A linear–time heuristic improving network partitions". *Proc. of the 19th Design Automation Conference*, 1982, pp 175–181.

[35] M-R.Garey and D.S.Johnson. *Computers and Intractability: A guide to the theory of NP–Completness.* Freeman, San Francisco CA, 1979.

[36] R.L.Geiger, P.E.Allen and N.R.Strader. *VLSI design techniques for analog and digital circuits.* McGraw–Hill, 1990.

[37] S. Geman and D. Geman. "Stochastic relaxation, Gibbs distributions, and the bayesan restauration of images". *IEEE Trans. Pattern An. and Mach. Intel.*, vol 6, pp 721–741, 1984.

[38] G.G.E.Gielen, H.C.C.Walscharts and W.M.C.Sansen. "Analog circuit design optimization based on symbolic simulation and simulated annealing". *IEEE J. Solid–State Circuits*, vol. 25, no. 3, pp. 707–713, June 1990.

[39] P.E.Grill, W.Murray, and M.H.Wright. *Practical Optimization.* Academic Press: New York, 1981.

[40] F. Glover. "Tabu Search–Part I". *ORSA Journal on Computing*, Vol 1, No 3, pp 190–206, Sum. 1989.

[41] F. Glover. "Tabu Search–Part II". *ORSA Journal on Computing*, Vol 2, No 1, pp 4–32, Win. 1990.

[42] M.K.Goldberg and M.Burstein. "Heuristic improvement technique for bisection of VLSI networks", in *Proc. of the IEEE Int. Conf. on Computer Design: VLSI in Computers*, pp. 122–125, 1983.

[43] O.Goldschmidt and D.S.Hochbaum. "A polynomial algorithm for the *k–cut* problem". *Technical Report, Department of Indutrial Engeneering and Operations Research and School Business Administration*, University of California, Berkeley, CA, 1987.

[44] S.Goto, T.Matsuda. " Partitioning, Assignment and Placement". in *Layout Desing and Verification*, Advances in CAD for VLSI, vol. 4, T.Ohtsuki Ed., pp. 55–98, North Holland, 1986.

[45] P.R.Gray and R.G.Meyer. *Analysis and design of analog integrated circuits.* John Wiley, 1984.

[46] J.W.Greene and K.J.Supowit. "Simulated annealing without rejected moves". *Proc. of the IEEE Int. Conf. on Computer Design: VLSI in Computers*, pp. 658–663, 1984.

[47] K.M.Hall. "An r–dimensional quadratic placement algorithm". *Management Science*, 17, pp. 219–229, 1970.

[48] R.Harjani, R.A.Rutenbar and L.R.Carley. "OASYS: A framework for analog circuit synthesis". *IEEE Trans. on Computer–Aided Design*, vol. 8, no. 12, pp. 1247–1266, December 1989.

[49] M.R. Hartoog. "Analysis of placement procedures for VLSI standard cell layouts" *Proc. of the 23rd ACM/IEEE Design Automation Conference*, pp. 314–319, 1986.

[50] J.J.Hopfield and D.W.Tank. "Neural computation of decisions in optimization problems", *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.

[51] M.D.Huang, F.Romeo and A.Sangiovanni-Vincentelli. "An efficient general cooling schedule for simulated annealing". *Proc. of the vInt. Conf. on CAD*, pp. 381–384, 1986.

[52] L.Ingber. "Simulated Annealing: practice versus theory", *Journal Mathl. Comput. Modelling*, 1993.

[53] M. Ismail and J. Franca, Eds. *Introduction to analog VLSI design automation.* Kluwer Academic Publishers, 1990.

[54] A.B.Kahng. "Fast Hypergraph partition". *Proc. of the 25th ACM/IEEE Design Automation Conf.*, pp. 762–766, 1989.

[55] B.W. Kernighan and S. Lin. "An efficient heuristic procedure for partitioning graphs". *Bell Syst. Technologies Jour.*, Vol 49. pp. 291–307, Feb. 1970.

[56] S.S. Kim and C.M. Kyung. "Circuit placement in arbitrarily-shaped regions using the self-organization principle". *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 7, pp. 844–854, 1992.

[57] S.Kirpatrick, C.D.Gelatt Jr. and M.P.Vecchi "Optimization by simulated annealing". *Science*, Vol 220, pp 671–680, May 1983.

[58] H.Kita, H.Odani and Y.Nishikawa. "Solving a placement problem by means of an analog neural network". *IEEE Trans. on Industrial Electronics*, vol. 39, no. 6, pp 543–550,Dec 1982.

[59] J.M.Kleinhans, G.Sigl and F.M.Johannes. "GORDIAN: A new global optimization / Rectangle dissection method for cell placement". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp. 506–509, 1988.

[60] R.M.Kling and P.Bannerjee. "ESP: Placement by simulated evolution". *IEEE Trans. on Computer-Aided Design*, vol 8, no 3, March 1989.

[61] H.Y.Koh, C.H.Séquin and P.R.Gray. "OPASYN: A compiler for CMOS operational amplifiers". *IEEE Trans. on Computer-Aided Design*, vol. CAD-9, no. 2, pp. 113–125, February 1990.

[62] T.Kohonen. *Self-organization and associative memory*, 3rd. ed., New York: Springer–Verlag, 1989.

[63] T.Kohonen. "Self–organizing maps". *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[64] M.R.Kramer and J.van Leewen. "The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits". *F.P. Prep. Advances in Computing Research*, Vol 2, VLSI Theory, pp 129–146.

[65] S.A.Kravitz and R.A.Rutenbar. "Placement by simulated annealing on a multiprocessor". *IEEE Trans. on Computer–Aided Design*, Vol 9, No. 10, pp 534–549,October 1990.

[66] B.Krishnamurthy. "An improved min–cut algorithm for partitioning VLSI networks". *IEEE Trans. on Computers*, vol. 33, no. 5, pp. 438–446, May 1984.

[67] B.Landman and R.Russo. "On a pin versus block relationship for partitions of logic graphs". *IEEE Trans. on Computers*, vol. 20, pp. 1469, 1971.

[68] J.Lam and J.Delosme. "Performance of a new annealing schedule". *Proc. of the 25th ACM/IEEE Design Automation Conference*, pp 306–311, 1988.

[69] J.Lam and J.Delosme. "Simulated annealing: A fast heuristic for some generic layout problems". *Proc. of the 25th ACM/IEEE Design Automation Conference*, pp 510–513, 1988.

[70] T.Leighton and S.Rao. "An approximate max–flow min–cut theorem for uniform multicommodity flow problems with applications to approximation algorithms". *Proc. IEEE 28th Ann. Symp. on Foundations of Computing*, pp. 422–431, 1988.

[71] T.Lengauer. *Combinatorial algorithms for integrated circuits*, Wiley–Teubner, 1990.

[72] I.Lin and D.H.C.Du. "Performance–driven constructive placement". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp 103–106, 1990.

[73] S.Mallela and L.K.Grover. "Clustering based simulated annealing for standard cell placement". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp 36–39,1990.

[74] *Magic 1990 DECWRL/Livermore Release.*

[75] P.C.Manlik, L.R.Carley and D.J.Allstot. "Sizing of cell-level analog circuits using contrained optimization techniques". *IEEE J. Solid-State Circuits*, vol 28, n. 3, pp 233–241, March 1993.

[76] C.Mead and L.Conway. *Introduction to VLSI Systems*, Reading, MA:Addison–Wesley, 1980.

[77] F. Medeiro, R. Domínguez–Castro, A. Rodríguez and J.L.Huertas. "A prototype tool for optimum analog sizing using simulated annealing". *Proc. of the IEEE Int. Symp. on CAS*, 1992.

[78] D.Metra, F.Romeo and A.Sangiovanni–Vincentelli. "Convergence and finite–time behaviour os simulated annealing" *Proc of the 24th IEEE Conf. Decision and Control*, pp 761–767, Dec. 1985.

[79] N.S.Nagarj. "A new optimization for performance optimization of integrated circuits by device sizing". *Proc. of the IEEE Int. Symp. on CAS*, pp 2102–2105, Chicago, May 1993.

[80] S.Nahar, S.Sahni and E.Shragowitz. "Simulated annealing and combinatorial optimization". *Proc. of the 23rd ACM/IEEE Design Automation Conf.*, pp. 293–299, 1986.

[81] T.K.Ng, J.Oldfield and V.Pitchumani. "Improvements of a mincut partition algorithm". *Proc. of the Int. Conf. on Computer–Aided Design*, pp. 470–473, 1987.

[82] W.Nye, D.C.Riley, A.Sangiovanni–Vincentelli and A.L.Tits. "DELIGHT.SPICE: An optimization–based system for the design of integrated circuits". *IEEE Trans. on Computer–Aided Des.*, vol. CAD–7, no. 4, pp. 501–519, April 1998.

[83] E.S.Ochotta, R.A.Rutenbar and L.R.Carley. "Equation free synthesis of high performance linear analog circuits". *Proc. of the 1992 Brown/MIT Conf. Advanced Research in VLSI and Parallel Systems*, Cambridge, Ma:MIT Press, pp 129–143, 1992.

[84] H.Onodera, H.Kanbara and K.Tamaru. "Operational-amplifier compilation with performance optimization". *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 466–473, April 1990.

[85] R.H.J.M.Otten. "Automatic floorplan design". *Proc. of the 19th ACM/IEEE Design Automation Conf.*, pp. 261–267, 1982.

[86] C.I.Park and Y.B.Park. "An efficient algorithm for VLSI network partitioning problem using a cost function with balancing factor". *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 11, pp. 1086–1094, Nov. 1993.

[87] R.G.Parker and R.L.Rardin. *Discrete optimization*, Academic Press, 1988.

[88] L.T.Pillage and R.A.Rohrer. "A quadratic metric with a simple solution scheme for initial placement". *Proc. of the 25th ACM/IEEE Design Automation Conference, pp 324–327*, 1988.

[89] N.R.Quinn. "The placement problem as viewed from the physics of classical mechanics". *Proc. of the 12th ACM/IEEE Design Automation Conf.*, pp. 173–178, 1975.

[90] M.Razaz. "A fuzzy c-means clustering placement algorithm". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2051–2054, Chicago, May 1993.

[91] J.Rose, W.Klebsch and J.Wolf. "Temperature measurement of simulated annealing placements". *Proc. of the 25th ACM/IEEE Design Automation Conference*, pp 514–517, 1988.

[92] J.Rose, W.Klebsch and J.Wolf. "Temperature measurement and equilibrium dynamics of simulated annealing Placements". *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 3, pp 253–259, March 1990.

[93] J.Rose. "Parallel global routing for standard cells". *IEEE Trans. on Computer-Aided Design*, vol. 6, o.. 4, pp 1085–1095, July 1987.

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

[95] Y.G.Saab and V.B.Rao. "Fast effective heuristics for the graph bisectioning problem". *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 1, Jan. 1990.

[96] Y.G.Saab and V.B.Rao. "Combinatorial optimization by stochastic evolution". *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 4, pp 525–535, April 1991

[97] L.A.Sanchis. "Multiple-way network partitioning".*IEEE Trans. on Computers*, vol. 38, no. 1, pp 62–81, Jan 1989.

[98] L.A.Sanchis. "Multiple-way partitioning with different cost functions". *IEEE Trans. on Computers*, vol. 42, n. 12, pp 1500–1504, Dec. 1993.

[99] A.Sangiovanni-Vincentelli. "Automatic layout of integrated circuits" in *Design Synthesis and Silicon Compilation*, G.DeMicheli, A.Sangiovanni-Vincentelli and P.Antognetti Ed., Martinus Nithjhoff Publishers, 1987.

[100] C.Sechen and D.Chen. "An improved objective function for mincut circuit partioning". 1988.

[101] C.Sechen. *VLSI placement and global routing using simulated annealing*, Kluwer Academic Publishers, 1988.

[102] C.Sechen. "Chip-planning, placement, global routing of macro-custom cell integrated circuits using simulated annealing". *Proc. of the 25th ACM/IEEE Design Automation Conference*, pp 73–80, 1988.

[103] C.Sechen and A.Sangiovanni-Vincentelli. "The TimberWolf placement and routing package". *IEEE J. Solid-State Circuits*, vol. 20, pp. 512–522, 1985.

[104] C.Sechen and A.Sangiovanni-Vincentelli. "TimberWolf3.2: a new standard cell placement and global routing package". *Proc. 23rd ACM/IEEE Design Automation conf* pp. 432–439. June 1986.

[105] C.Sechen and K.W.Lee. "An improved simulated annealing algorithm for row based placement". *Proc. of the Int. Conf. Computer-Aided Design*, pp 478–481. June 1987.

[106] K.Shahookar and P.Mazumder. "A genetic approach to standard cell placement using meta-genetic parameter optimization". *IEEE Trans. on Computer-Aided Design*, vol. 9, no. 5, pp 500–511, May 1990.

[107] D.G.Schweikert and B.W.Kernighan. "A proper model for the partitioning of electrical circuits". *Proc. of the 9th Ann. Design Automation Conf.*, pp. 57–62, 1972.

[108] J.M. Shynand and A.L.Sangiovanni–Vincentelli. "ECTASY: a new environment for IC design optimization". *Proc. of the IEEE Int. Conf. on CAD* pp. 484–487, 1988.

[109] J.Skorin–Kapov. "Tabu search–part applied to the quadratic assigment problem". *ORSA Journal on Computing*, vol. 2, no. 1, pp 33–40, Win. 1989.

[110] P.R.Suaris and G.Kedem. "An algorithm for quadrisection and its application to standard cell placement". *IEEE Trans. on Circuits and Systems*, vol. 35, no. 3, pp 294–303, March 1988.

[111] P.R.Suaris and G.Kedem. "A quadrisection–based combined place and route scheme for standard cells". *IEEE Trans. on Computer–Aided Design*, vol. 8, no. 3, pp 234–244, March 1989.

[112] K.Swings, S.Donway and W.Sansen. "HECTOR: A hierarchical topology–construction program for analog circuits based in a declaration approach to circuit modeling". *Proc. of the IEEE CICC'91* San Diego, Ca., pp 5.3.1–5.3.4, 1991.

[113] K.Swings and W.Sansen. "Ariadne: A constraint–based approach of computer–aided synthesis and modelling of analog integrated circuits". *Analog Integrated Circuits and Signal Processing*, vol 3, no. 3, pp 197–216, May 1993.

[114] D.A.Tank and J.J.Hopfield. "Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit". *IEEE Trans. on Circuits and Systems*, vol. 33, pp. 533–541, May 1986.

[115] M.Terai, K.Takahashi and K.Sato. "A new min–cut placement algorithm for timing assurance layout design meeting net lencth constraint". *Proc. of the 27th ACM/IEEE Design Automation Conf.*, pp. 96–102, 1990.

[116] R.Tsay, E.S.Kuh and C.Hsu. "PROUD: A fast sea–of–gates placement algorithm". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp. 318–323, 1988.

[117] Y.Tsividis. "The MOS transistor". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp. 318–323, 1988.

[118] M.K.Unamuna and V.Pitchumani. "Quadrisectioning based placement with a normalized mean field neural network". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2047–2050. Chicago, May 1993.

[119] M. Upton, K. Samii and S. Sugiyama. "Integrated placement for mixed macro cell and standard cell". *Proc. of the 27th ACM/IEEE Design Automation Conference*, pp 32–35, 1990.

[120] J.P.Uyemura. *Fundamentals of MOS digital integrated circuits*, Addison–Welsey, 1988.

[121] S.Wakabayashi, H.Kusumoto, M.Mishima, T.Koide and N.Yoshida. "Gate–array placement based on mincut partitioning with path delay constraints". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2059–2062. Chicago, May 1993.

[122] Y.C.Wei and C.K.Cheng. "Ratio–cut partitioning for hierarchical designs". *IEEE Trans. on Computer–Aided Design*, vol. 10, no. 7, pp. 911–921, July 1991.

[123] N. Weste and K. Eshraghian. *Principles of CMOS VLSI design*. Addison Wesley, 1988.

[124] G.J.Wipfler, M.Wiesel and D.A.Mlynski. "A combined force and cut algorithm for hierarchical VLSI layout". *Proc. of the 19th ACM/IEEE Design Automation Conf.*, pp. 671–677, 1982.

[125] J.S.Yi and P.Mazumder. "A neural network design for circuit partitioning". *IEEE Trans. on Computer–Aided Design*, vol. 9, no. 12, pp. 1265–1271, Dec. 1990.

[126] C.X.Zhang. "Timing–, Heat–, and Area–driven placement using self–organizing semantic maps". *Proc. of the IEEE Int. Symp. on CAS*, pp. 2067–2070. Chicago, May 1993.

[127] X.Zhang, L.T.Pillage and R.A.Roher. "Efficient final placement based on net–as–points". *Proc. of the 26th ACM/IEEE Design Automation Conf.*, pp. 578–581, 1989.

Apéndice I

Optimización de Problemas Combinatorios

En esta sección pretendemos sentar las bases formales de nuestro problema. Estableceremos la notación y definiremos las propiedades que caracterizan su complejidad. Se define, en particular, la categoría de NP–Duro y se muestra cómo el problema del *Emplazamiento* y *Trazado General* de circuitos *VLSI* pertenece a ella y por tanto nos exige la aplicación de algoritmos de especiales características.

I.1 Definiciones Básicas de Análisis Algorítmico

I.1.1 Problema Algorítmico

Definición: Problema Algorítmico. Un problema Π se define como **Algorítmico** si es posible establecer la relación $\Pi : I \rightarrow 2^S$, siendo I el conjunto de posibles problemas que resuelve el algoritmo, y S el espacio de las configuraciones de las soluciones posibles. Para un problema concreto $p \in I$ a las configuraciones $\pi(p)$ se les denominan soluciones de p o posibles configuraciones para p. I y S pueden ser finitos o infinitos.

Un caso particular interesante es aquel en que el espacio de las configuraciones es unitario. A este problema se le denomina *de decisión.*

I.1.2 Complejidad de un Algoritmo

¿Cómo evaluar un algoritmo y su eficacia? ¿Qué medidas podemos asignar para conocer su capacidad, y más aún, con independencia de la buena o mala técnica de programación que se introuzca?

La medidas fundamentales son dos: el tiempo de computación y la memoria de almacenamiento involucrada en su ejecución. La primera variable es la que en realidad se usa ya que los modernos computadores admiten con las técnicas de "swapping" memorias casi ilimitadas.

Definición: Complejidad de un Algoritmo. Un algoritmo A resuelve un problema algorítmico Π si, dada una codificación para el problema p, con $p \in I$, el algoritmo computa y encuentra la solución $S \in \Pi(p)$. El tiempo de ejecución del algoritmo A sobre un elemento p del conjunto I se denota por $T(A,p)$; los requisitos de memoria se denotarán por $S(A,p)$.

Para evaluar más fácilmente el S y T se asocia a cada elemento del conjunto I un valor llamado *tamaño del problema*, $l(p)$.

Dado que los algoritmos pueden hacer depender su tiempo de ejecución del propio problema planteado hemos de definir una cota superior a la complejidad referida al tiempo de ejecución, llamada Tiempo del *peor caso*:

$$T_{PC}(A,n) := max\{T(A,p) \mid p \in I, l(p) = n\} \tag{I.1}$$

Dado que el peor caso puede ser un elemento que nunca se evalúe es mejor proponer un tiempo característico, dado por la probabilidad de que un problema p sea evaluado al ser escogido del conjunto I. Definimos un tiempo medio:

$$T_{med}(A,n) := \sum_{p,l(p)=n} \pi(p)T(A,p) \tag{I.2}$$

I.1.3 Algoritmos con mecanismos aleatorios

Existen problemas en los que inevitablemente el algoritmo no puede evaluar y decidir el siguiente paso por si mismo. Ha de dotársele de un mecanismo aleatorio con el cual le sea posible adoptar una nueva configuración sin perjuicio de la solución final.

Formalmente se puede decir que el algoritmo tiene dos entradas, $A(p, \sigma)$, siendo σ una cadena aleatoria que utilizará el algoritmo para resolver "encrucijadas".

La complejidad del algoritmo toma de nuevo una matiz diferente, ya que la secuencia aleatoria podría resolver el problema "acertando" siempre en sus decisiones o bien acertando solamente al final. Por ello se define la complejidad en tiempo *Las Vegas* como:

$$T_{LV}(A,p) := \lim_{k\to\infty} 2^{-k} \sum_{p,l(p)=n} T(A,(p,\sigma)) \tag{I.3}$$

$$T_{LV}(A,n) := \max_{p,l(p)=n} T_{LV}(A,p) \tag{I.4}$$

El valor exacto de estas medidas son, en general difíciles de determinar, ya que dependen del tipo de implementación del problema, de la buena o mala codificación del algoritmo, del código obtenido, del entorno de trabajo,... etc. En el campo del análisis algorítmico no se evalúan estos problemas, sino que se busca determinar la complejidad intrínseca, esto es, ver en qué medida crece la complejidad del problema con el parámetro que mejor lo caracteriza, como es su tamañoo la *longitud* $n = l(p)$. Para poder establecer unos criterios de evaluación prácticos se precisa la ayuda del *Análisis Asintótico*

I.1.4 Análisis Asintótico

Este análisis establece unas cotas sobre la complejidad del algoritmo, determina cómo varía $T(A,n)$ y lo compara con funciones. Fué introducido por Knutt y tiene la siguiente notación:

- $f(n) = O(g(n))$ si existen constantes $n_0 \in N$ y $c > 0$, tales que $\forall n > n_0 \Rightarrow f(n) \leq c \cdot g(n)$
- $f(n) = \Omega(g(n))$ si existen constantes $n_0 \in N$ y $\epsilon > 0$, tales que $\forall n > n_0 \Rightarrow f(n) \geq \epsilon \cdot g(n)$

Las funciones $f(n)$ y $g(n)$ se las caracteriza por funciones que acotan su comportamiento, tanto inferior como superiormente. Al crecimiento asintótico de $T(A,n)$ se le denomina *complejidad asintótica en tiempo del peor caso* del algoritmo A.

Puede ocurrir que el problema Π se pueda resolver por diferentes algoritmos, por tanto se define la complejidad del peor caso de un problema algorítmico minimizando la complejidad para todos los algoritmos, es decir, será la del algoritmo que mejor resuelve el problema:

$$T_{PC}(\Pi, n) := \min_{A resuelve \Pi} T_{PC}(A, n) \tag{I.5}$$

I.1.5 Problema de Optimización

Un problema de optimización es un problema algorítmico Π para el cual existe una función de costes $c : S \rightarrow R$. Un algoritmo A tal que dado $p \in I$, computa una configuración válida s tal que c(s) es mínimo se dice que *minimiza* Π.

I.2 Caracterización como NP–Duro

Con objeto de justificar la complejidad del problema del *Emplazamiento* y *Trazado General* de circuitos *VLSI* vamos a establecer categorías a los algoritmos basándonos en la idea anterior de funciones que acotan la complejidad.

I.2.1 Problema Factible y no–Factible

Un algoritmo A se dice que es *factible* (o de tiempo polinomial) si su $T_{PC}(A, n) = O(n^{\alpha})$ para algún $\alpha \geq 0$. Un problema algorítmico Π se le denomina *factible* (o de tiempo polinomial) si existe un algoritmo A factible que lo resuelve. A la clase de problemas de decisión factibles se la denota por **P**.

La definición de algoritmo factible es, en cierto modo, pobre, ya que pueden existir algoritmos que resuelvan problemas en tiempo no polinomial, pero con la función de acotación del tipo $2^{n/1000}$, que, según lo expuesto no es factible. Sin embargo pueden existir otros algoritmos que lo resuelvan en tiempo n^{1000} que sí es factible, y resulta que el algoritmo no factible es más rápido siempre que $n \leq 2.4 \times 10^7$. Si la dimensión más frecuente del problema es menor que este número parece razonable emplear el algoritmo no factible.

I.2.2 Algoritmos no deterministas

Sea Π un problema de decisión. Un algoritmo *no determinista* A para Π es aquel que, a cada paso, se permite hacer una *disquisición binaria* de como proceder. El algoritmo *resuelve* el problema si, para cada problema $p \in I$, ocurre lo siguiente:

- Si $\Pi(p) = 1$ se concluye que existe un modo en A para determinar que un 1 se ha computado. En este caso, se define $T_{nodet}(A, p)$ como el tiempo de cálculo más corto para tal computación.
- Si $\Pi(p) = 0$ se concluye que no existe un modo en A para determinar que un 1 se ha computado. Definimos $T_{nodet}(A, p)$=1.

La clase de problemas de decisión que se resuelven mediante algoritmos factibles y no-deterministas se llama **NP**.

I.2.3 Problema Umbral

Sea $\Pi : I \rightarrow 2^S$ un problema de minimización con una función de costes c. El *problema umbral* $\Pi_{UMB} : I \times N \rightarrow \{0, 1\}$ es el problema de decisión tal que dado un elemento $(p, k) \in I \times N$, toma el valor 1 si existe una configuración $s \in \Pi(p)$ tal que $c(s) \leq k$.

La idea que soporta esta definición es que el problema Π será *umbral* cuando se encuentre un valor de la función de costes que sea inferior a la constante k dada.

I.2.4 NP–Completo y NP–Duro

Definición: NP–Completo. Una Transformación Polinomial Φ de un problema de decisión en otro $\Pi_1 : I_1 \rightarrow \{0, 1\}$ a otro problema de decisión $\Pi_2 : I_2 \rightarrow \{0, 1\}$ es una función que puede ser computada por un algoritmo determinista y en tiempo polinomial de forma que para cada $p \in I_1$ tenemos $\Pi_1(p) = \Pi_2(\Phi(p))$. Si tal transformación Φ existe, Π_1 se llama *reducible* a Π_2 (se denota por $\Pi_1 \leq_m \Pi_2$. Un problema $\Pi' \in NP$ pertenece a la clase **NP–Completo** si $\Pi \leq_m \Pi'$, cualquiera que sea Φ polinomial,$\forall \Pi \in NP$.

Significa que una transformación polinomial siempre conserva la categoría del problema. Por tanto si se llega a formular un problema algorítmico nuevo siempre sera

del tipo NP hasta que se encuentre una transformación polinomial que lo relacione con un problema P.

Los problemas de optimización cuyo problema umbral es **NP–Completo** se le denomina **NP–Duro**

I.3 Resolución de Problemas *NP–Duro*

I.3.1 Algoritmos pseudopolinomianles

Corre en tiempo polinomial desde el punto de vista de la longitud del problema y desde el punto de vista de los números que concurren en la codificación del problema. Tales algoritmos son eficientes si los números que intervienen no son grandes. Si un problema *NP–completo* se resuelve mediante un algoritmo pseudopolinomial se llama *débilmente NP–completo*. El problema del placement de celdas es *fuertemente NP–completo*, incluso con una codificación unaria del problema (entendiéndose por tal, una codificación de los número a base de cantidad de caracteres p.e. '/').

I.3.2 Algoritmos aproximativos

Se pueden diseñar algoritmos para problemas de optimización que aproximen a la solución final eficientemente. Para medir la calidad de la solución se utiliza la expresión:

$$ERROR(A) := max_{p \in I} c(A(p))/c(Opt(p)) \tag{I.6}$$

Se puede aproximar en tiempo polinomial a una cierta cota de este error.

I.3.3 Aleatoriedad

Es posible introducir la idea de *aleatoridad* del algoritmo en problemas de optimización, ya que así se consigue mejorar el tiempo de computación para aproximarse al coste óptimo. Existen varias posibilidades de caracterización de un algoritmo de búsqueda aleatoria:

1. Que el algoritmo siempre determina una solución óptima.

2. Que el algoritmo determina una solución óptima, la encuentre con una alta probabilidad.

3. Si el algoritmo computa una solución óptima, produce una pruba de que es óptima.

4. El algoritmo resuelve en tiempos pequeños en el peor caso.

5. El tiempo de computación del algoritmo, promediado sobre la aleatoriedad interna es pequeño para cada problema.

Los algoritmos que se caracterizan por las propiedades 1 y 5 se denominan tipo *Las Vegas* y los de las propiedades 2 ó 3 y 4 son del tipo *Monte Carlo*

I.3.4 Heurísticas

La última posibilidad es proceder heurísticamente. Son algoritmos que, dotados de un mecanismo aleatorio pueden llevar a obtener valores de la función de costes cercanos al óptimo, pero sin saber medir la cercanía del mismo, ni hasta qué punto es posible, que si continuamos probando, se encuentren mejores configuraciones.

I.4 Definición del Problema de la Disposición

El diseño automático de una *Disposición* es un problema de optimización con restricciones. Ver sección 2.2.

Apéndice II

Más sobre el Enfriamiento Simulado

II.1 Convergencia del algoritmo

El algoritmo fué inicialmente concebido por Kirpatrick a partir del estudio del algoritmo de integración numérica de Metrópolis Monte Carlo.

Un aspecto interesante del algoritmo conocer los órdenes de magnitud de lo parámetros a buscar. En espacios continuos, suele buscarse una discretizaciónde todo el espacio.

Si nos basamos en las formas funcionales derivadas del comportamiento de los sistemas físicos que pertenecen a la clase de sistemas Gaussianos–Markovianos, la función densidad de probabilidad $g(x)$, donde x son los D parámetros de búsqueda ($x = x^i; i = 1..D$), vale:

$$g(\Delta x) = (2\pi T)^{-D/2} e^{\frac{-\Delta x^2}{2T}} \tag{II.1}$$

Dada $g(\Delta x)$, se ha probado que es condición suficiente para alcanzar un mínimo global de la función de coste $c(x)$ que T seleccionada no sea más rápida que:

$$T(k) = \frac{T_0}{\ln k} \tag{II.2}$$

con T_0 suficientemente grande. Se muestra que el óptimo de $c(x)$ se alcanza, es suficiente probar que la probabilidad de no alcanzar un estado x en un número suficientemente

grande de intentos posteriores a k_0 vale cero:

$$\prod_{k=k_0}^{\infty} (1 - g_k) = 0 \tag{II.3}$$

Que es equivalente a decir que:

$$\sum_{k=k_0}^{\infty} g_k = \infty \tag{II.4}$$

Por tanto si $T(k)$ cumple la condición anterior:

$$\sum_{k=k_0}^{\infty} g_k \geq \sum_{k=k_0}^{\infty} e^{-\ln k} = \sum_{k=k_0}^{\infty} 1/k = \infty \tag{II.5}$$

Lundi y Mees (1986) proporcionan una costa de cercanía al óptimo:

$$\frac{\alpha}{\alpha - 1} > (k - 1) \cdot e^{-\epsilon/T} \tag{II.6}$$

donde α es un índice de fiabilidad y ϵ es la cota de la distancia al óptimo.

Apéndice III

Resultados sobre los circuitos de prueba

III.1 Experiencia con *TimberWolfSC-4.2*

Circuito	Celdas	Redes	Conexiones	Pads	n. filas
Fract (s109)	125	147	1268	24	4
Struct (mult16)	1888	1920	9559	64	32
Biomed (fan)	6417	5766	39994	32	32
Primary1 (CIRCUITX)	833	985	5186	81	16
Primary2 (RPROC)	2907	3029	24773	107	24

Tabla III.1: Características generales de los circuitos de prueba utilizados

Condiciones de las pruebas: Número de filas según la tabla III.1. Computador: SUN SPARC 10-30. S.O. SunOs 4.1.3.

Circuito	tiempo	C. Horiz.	C. Vert.	C. Final	Feed	n. pista	n. final	n.i.r.	iterac
Fract (s109)	44.6	28048	16952	45000	129	77	72	100	123
Struct (mult16)	1592.9	382210	323069	705279	2646	411	383	1750	123
Biomed (fan)	26457.2	2222687	1718168	3940855	11951	1274	1148	12150	123
Primary1 (CIRCUITX)	489.9	611447	888515	1499962	1100	384	347	450	123
Primary2 (RPROC)	4466.8	2331392	3570779	5902171	1640	1455	1294	2350	123

Tabla III.2: Resultados con *TimberWolfSC-4.2*

III.2 Experiencia con el Algoritmo Combinado *foxy*

Circuito	tiempo	C. Horiz.	C. Vert.	C. Final	Feed	n. pistas	n. final	n.i.r.	iterac	T_0
Fract (s109)	4.8	27112	17706	44818	121	89	85	150	7	21.8
Struct (mult16)	288.9	486485	242632	729117	2100	532	492	2000	22	88.3
Biomed (fan)	19849.1	2275488	1239493	3514981	10732	2042	1892	8700	22	80.0
Primary1 (CIRCUITX)	258.5	618654	772496	1391150	1157	457	429	450	50	150.0
Primary2 (RPROC)	3025.5	2459812	3048322	5508134	1288	1440	1322	2200	52	149.3

Tabla III.3: Resultados con partición en 4 bloques

Circuito	tiempo	C. Horiz.	C. Vert.	C. Final	Feed	n. pistas	n. final	n.i.r.	iterac	T_0
Fract (s109)	9.2	21394	13994	35338	122	81	74	150	22	80.08
Struct (mult16)	287.8	432873	232482	665355	2038	480	438	2000	22	80.0
Biomed (fan)	21016.3	2201966	1139855	3341821	9873	1944	1809	8250	23	82.0
Primary1 (CIRCUITX)	279.5	612073	808490	1420563	1122	481	449	500	48	135.3
Primary2 (RPROC)	2690.2	2542745	2656429	5199174	1020	1516	1390	2050	53	155.6

Tabla III.4: Resultados con partición en 16 bloques

Circuito	tiempo	C. Horiz.	C. Vert.	C. Final	Feed	n. pistas	n. final	n.i.r.	iterac	T_0	C.
Struct (mult16)	292.9	402494	244488	646982	2017	510	473	2000	22	81.2	173
Biomed (fan)	20474.8	2275588	1094669	3353480	9477	2024	1903	8000	21	73.2	935
mary1 (CIRCUITX)	280.2	6009152	828905	1438057	1212	482	456	400	49	139.0	246
rimary2 (RPROC)	2752.9	2551470	2748332	5299792	1274	1646	1544	2200	54	157.2	1105

Tabla III.5: Resultados con partición en 64 bloques

Apéndice IV

Resultados de las Pruebas de Síntesis Analógica

Los resultados que se presentan en este apéndice están simulados con la versión SPICE2GV. La simulación se ha realizado sobre una estación SUN SPARC 10–30.

IV.1 Resultados para CAS, OTA, BTS

****	fobj	A0=20	GB=1	MF=60	AREA=5000					
1790	0.941207	68.68	4.60189	86.7544	416	0.119	283.48	1.34099	1.49617e-05	4.85296e-06
1501	0.939003	67.25	4.56445	87.2635	416	0.124	283.25	1.16401	1.45796e-05	6.7354e-06
2086	0.939043	70.49	4.02785	87.1564	416	0.105	279.69	2.00771	1.10734e-05	4.82241e-06
2033	0.939003	67.25	4.56445	87.2635	416	0.124	283.25	1.26284	1.45084e-05	6.49321e-06
1911	0.935845	69.3	4.75455	86.7123	400	0.105	281	1.41326	1.0818e-05	4.95951e-06
****	fobj	A0=20	GB=5	MF=60	AREA=5000					
1465	1.09447	72.18	15.5706	79.9792	648	0.146	275.68	7.02309	1.96039e-05	8.26606e-06
1331	1.09564	71.09	15.3096	80.5032	656	0.165	276.79	6.15766	2.46506e-05	8.30476e-06
1435	1.0952	74.27	13.9949	80.6234	652	0.136	275.97	7.63208	1.77613e-05	7.95213e-06
1476	1.09482	73.43	14.9779	79.8858	628	0.135	276.33	6.64972	1.86771e-05	7.52367e-06
1517	1.10046	71.74	14.6706	80.6453	680	0.165	276.74	6.86899	2.40761e-05	9.09373e-06
****	fobj	A0=20	GB=10	MF=60	AREA=5000					
1366	1.27719	73.84	24.4746	73.8245	868	0.176	275.64	10.934	2.59945e-05	1.05142e-05
1377	1.29287	72.48	19.5114	77.0888	748	0.172	275.58	8.87376	2.60887e-05	8.6188e-06
1742	1.27879	73.96	19.7059	76.8607	704	0.146	275.56	8.66062	2.03393e-05	8.57902e-06
1921	1.27948	73.74	23.8122	74.3347	896	0.186	276.44	10.7922	2.82462e-05	1.18532e-05
1466	1.29058	72.15	25.0447	73.748	924	0.211	276.35	10.1708	3.50591e-05	1.18627e-05
****	fobj	A0=60	GB=1	MF=60	AREA=5000					
1591	1.57729	74.08	8.18648	83.2217	540	0.124	277.88	4.86888	1.68242e-05	4.73434e-06
1434	1.58664	74.43	4.17158	82.6552	532	0.0941	279.93	2.23257	9.55564e-06	2.29524e-06
1710	1.57513	73.77	8.12581	83.8914	544	0.124	276.67	5.26908	1.45458e-05	5.67094e-06
1429	1.54287	77.73	3.17495	82.7646	536	0.0842	281.73	2.11374	6.85742e-06	2.20312e-06
1529	1.54287	77.73	3.17495	82.7646	536	0.0842	281.73	2.03026	6.65434e-06	2.19348e-06
1676	1.56379	77.78	10.3261	79.5349	612	0.109	275.58	7.57452	1.22586e-05	4.35951e-06
****	fobj	A0=60	GB=5	MF=60	AREA=5000					
1944	1.68198	76.28	13.9449	78.4778	644	0.129	276.18	7.89122	1.67889e-05	5.65096e-06
1350	1.67425	75.85	14.148	79.2739	656	0.125	274.95	8.86581	1.54705e-05	6.44329e-06
1375	1.68202	76.17	14.1598	78.3998	636	0.129	276.37	7.76819	1.65294e-05	5.86458e-06
1819	1.67702	75.41	14.553	79.2707	620	0.133	276.71	7.21485	1.69188e-05	6.40693e-06
1759	1.67702	75.41	14.553	79.2707	620	0.133	276.71	7.16962	1.67443e-05	6.70726e-06
****	fobj	A0=60	GB=10	MF=60	AREA=5000					
1365	1.87066	75.22	23.305	72.9461	824	0.173	276.32 1	1.0227	2.87475e-05	7.9747e-06
1307	1.87148	74.56	24.6969	73.0467	864	0.166	275.36	11.3831	2.32965e-05	1.01219e-05
1815	1.87116	74.54	24.4916	73.4133	860	0.176	276.04	10.9594	2.55257e-05	1.02531e-05
1593	1.87142	75.29	23.6762	73.0053	832	0.173	276.29	11.2333	2.81217e-05	8.12297e-06
2038	1.8705	75.59	24.5696	72.1939	840	0.163	275.19	12.071	2.3611e-05	8.74661e-06
****	fobj	A0=100	GB=1	MF=60	AREA=5000					
1622	2.57211	76.89	10.0416	80.7107	656	0.105	272.59	9.57209	1.09769e-05	5.01051e-06
2007	2.60718	76.86	12.6198	79.0744	700	0.129	275.06	9.85151	1.66312e-05	5.80455e-06
2098	2.60642	77.02	10.6381	78.3116	640	0.104	272.82	8.59213	1.13758e-05	3.7573e-06
1364	2.59447	75.48	10.2944	81.987	584	0.124	276.58	6.4923	1.5004e-05	5.5669e-06
1584	2.56996	77.69	2.89993	81.5751	548	0.0785	284.19	1.36866	5.91943e-06	1.94563e-06
****	fobj	A0=100	GB=5	MF=60	AREA=5000					
1285	2.69343	76.02	13.7734	79.3988	672	0.125	274.62	9.36782	1.50583e-05	6.55256e-06
1578	2.70282	75.41	14.553	79.2707	620	0.133	276.71	7.12991	1.68503e-05	6.6452e-06
1528	2.69849	77.34	12.8684	77.6182	692	0.124	275.14	9.49164	1.65744e-05	4.90727e-06
2055	2.6923	76.87	13.1424	78.6086	704	0.124	273.97	10.2487	1.51565e-05	5.65034e-06
1516	2.69298	76.73	13.4753	78.4396	688	0.124	274.33	9.83821	1.45579e-05	5.65713e-06
****	fobj	A0=100	GB=10	MF=60	AREA=5000					
1511	2.88481	76.33	19.9069	74.3347	700	0.133	275.03	9.70071	1.65167e-05	6.60315e-06
1487	2.89877	75.22	23.305	72.9461	824	0.173	276.32	11.116	2.81543e-05	7.86083e-06
1375	2.89772	75.46	23.5827	72.8705	852	0.168	275.26	12.2	2.61217e-05	8.32075e-06
1659	2.89638	75.03	19.7607	76.1339	716	0.136	274.73	9.77958	1.76113e-05	7.91381e-06
1694	2.89206	76.88	22.7265	71.7518	940	0.153	273.18	15.94	2.09823e-05	8.22191e-06

Tabla IV.1: Resultados para el Amplificador Cascodo Plegado

****	fobj	A0=20	GB=1	MF=60	AREA=5000						
2372	1.19646	51.75	4.13595	78.6497	596	0.119	215.75	10.4917	0.85818	1.71117e-05	2.85774e-06
2818	1.26929	46.94	5.45706	78.4811	512	0.249	207.84	6.79561	0.754713	3.80684e-05	7.1707e-06
2762	1.20291	49.66	3.2966	79.8557	484	0.121	213.76	6.72784	0.731091	1.7065e-05	2.84185e-06
2367	1.22146	51.54	4.18001	79.0543	848	0.17	216.34	16.2107	0.850079	2.91545e-05	3.15555e-06
2357	1.27121	41.53	3.2659	84.6596	416	0.318	209.43	3.0534	0.799057	4.9663e-05	9.0628e-06
****	fobj	A0=20	GB=5	MF=60	AREA=5000						
3035	1.63496	48.04	11.8514	72.698	904	0.199	161.14	18.1398	1.65188	2.79838e-05	5.04386e-06
2597	1.61033	45.21	12.731	74.6679	768	0.233	158.81	14.9327	2.29494	2.87019e-05	9.39711e-06
2650	1.62675	44.72	12.6395	73.5152	580	0.21	157.42	9.99976	2.49496	2.11859e-05	9.66671e-06
2329	1.64697	47.13	9.96121	73.4947	444	0.116	158.83	7.27027	1.51201	1.28373e-05	4.23509e-06
2102	1.65754	44.88	9.97688	74.9257	536	0.0979	154.68	8.70722	2.14926	9.93272e-06	3.07503e-06
****	fobj	A0=20	GB=10	MF=60	AREA=5000						
3002	1.91233	46.23	17.3941	68.3069	872	0.24	160.63	18.5682	2.34286	2.92103e-05	1.01651e-05
2418	1.877	43.46	18.668	71.3377	680	0.382	159.06	11.912	2.0821	4.55589e-05	1.65108e-05
3021	1.89302	45.47	19.2553	68.376	1104	0.296	160.57	23.3522	2.14879	4.07546e-05	8.75906e-06
2486	1.90837	41.23	19.3823	70.1512	344	0.387	155.53	4.37169	1.60448	3.29976e-05	2.66417e-05
2519	1.96807	38.28	17.5493	75.3265	432	0.609	172.08	4.95651	1.23546	6.27961e-05	3.72193e-05
****	fobj	A0=60	GB=1	MF=60	AREA=5000						
3342	2.22133	78.1	2.51161	69.4505	3344	0.134	228.4	70.0446	0.108133	2.49613e-05	1.24374e-06
2336	2.24608	74.48	1.92962	65.6831	1820	0.107	236.78	16.5117	0.0924447	1.68876e-05	1.48548e-06
2773	2.14527	73.01	2.41924	68.9557	2052	0.171	238.41	30.0007	0.111676	3.07262e-05	2.09511e-06
2072	2.30132	72.16	2.21919	62.1341	1376	0.0748	225.66	20.3773	0.171012	1.00268e-05	1.24392e-06
2849	2.22113	78.47	2.43454	65.1536	2884	0.127	252.57	45.0492	0.0824378	2.22283e-05	1.55968e-06
****	fobj	A0=60	GB=5	MF=60	AREA=5000						
3004	2.89079	55.44	7.62365	66.5409	912	0.108	219.84	20.4737	0.811546	1.61401e-05	2.89607e-06
3320	2.86419	57.44	8.90843	61.978	1180	0.1	205.74	29.8264	0.85649	1.42532e-05	3.36002e-06
2399	2.98807	52.4	10.246	66.3324	912	0.0708	160.9 21.	6481	1.88307	6.52375e-06	2.79412e-06
2601	2.94728	53.91	10.4112	66.5492	1452	0.181	169.31	37.0818	0.972788	2.6093e-05	4.43569e-06

Tabla IV.2: Resultados para el Amplificador Operacional de Transconductancia

****	fobj	A0=20	GB=1	MF=60	AREA=5000							
2633	1.0606	75.84	2.76362	80.6522	1160	0.0421	397.24	1.47081	52.1546	6.40915e-06	9.95021e-05	1.04032e-12
2669	1.05728	79.13	2.69567	80.3016	1056	0.029	394.53	2.26825	46.1204	4.16166e-06	8.99828e-05	1.09136e-12
2418	1.06043	76.34	2.79843	79.8589	1040	0.0359	394.94	1.84249	45.9204	5.65407e-06	8.51465e-05	1.04356e-12
3144	1.06577	74.09	3.10402	79.3189	1124	0.0523	396.39	1.19436	49.832	8.70597e-06	9.99149e-05	1.22823e-12
3228	1.07501	78.56	3.25739	78.4657	1272	0.0269	394.86	4.08736	54.6976	3.76786e-06	9.96647e-05	1.0074e-12
****	fobj	A0=20	GB=5	MF=60	AREA=5000							
2705	1.72829	74.82	8.57317	62.0208	1452	0.124	408.02	4.38174	53.8946	2.31357e-05	9.99994e-05	1.00101e-12
3317	1.73975	73.78	7.67326	64.3721	1324	0.113	405.28	2.71514	50.9951	2.13272e-05	9.64974e-05	1.08169e-12
3557	1.70469	78.28	7.82081	64.1214	1288	0.0527	402.18	3.92356	54.5117	8.36027e-06	9.93093e-05	1.00219e-12
2407	1.71885	75.74	7.79691	63.815	1220	0.073	403.24	3.46078	48.4588	1.37115e-05	9.65685e-05	1.00216e-12
2763	1.72575	76.96	8.62568	61.5793	1444	0.0935	407.36	5.73399	54.7814	1.73554e-05	0.0001	1.00408e-12
****	fobj	A0=20	GB=10	MF=60	AREA=5000							
3048	2.54032	77.16	11.5838	52.3248	1452	0.104	408.66	7.163	51.0697	2.07144e-05	9.95828e-05	1.00203e-12
2475	2.52325	78.29	11.6901	52.2588	1476	0.0834	407.29	7.34527	55.6059	1.48034e-05	9.98283e-05	1.00302e-12
2593	2.52741	78.77	11.799	51.6162	1384	0.0834	407.77	6.9916	50.1342	1.6081e-05	9.90365e-05	1.00725e-12
2351	2.55906	80.56	12.3385	49.5414	1560	0.063	407.16	8.90308	59.2756	1.13798e-05	9.99936e-05	1.00014e-12
2853	2.53233	76.42	12.0736	51.2132	1532	0.114	408.92	6.19692	56.3467	2.09441e-05	9.99933e-05	1.00869e-12
****	fobj	A0=60	GB=1	MF=60	AREA=5000							
2888	1.56952	80.54	2.42149	79.4039	900	0.0239	391.54	1.44423	38.7006	3.11927e-06	6.98208e-05	1.07212e-12
2291	1.56923	81.51	2.50801	79.1072	1176	0.0219	393.61	1.786	51.2099	2.80568e-06	9.99286e-05	1.07606e-12
2165	1.55454	83.39	2.826	77.8219	1388	0.0205	394.99	3.68858	62.2208	2.35628e-06	8.32881e-05	1.00371e-12
2545	1.58858	78.2	2.5535	80.2223	976	0.0319	393.9	2.33335	41.9997	5.05504e-06	7.91906e-05	1.13127e-12
3497	1.57751	79.42	3.25746	78.342	1244	0.0319	396.22	4.19982	52.5629	5.13705e-06	9.94663e-05	1.23156e-12
****	fobj	A0=60	GB=5	MF=60	AREA=5000							
2520	2.20725	80.46	7.80919	63.47	1296	0.0422	401.86	5.62685	52.1605	6.43937e-06	9.9822e-05	1.03596e-12
2465	2.21253	80.71	7.83064	63.5074	1452	0.0422	403.01	5.38039	61.9253	6.57442e-06	9.9754e-05	1.00104e-12
2576	2.24505	80.16	8.5029	61.159	1620	0.063	405.76	9.18628	62.5895	1.01827e-05	9.8807e-05	1.12456e-12
2060	2.25087	79.92	7.87105	62.5437	1432	0.0527	401.92	6.46855	58.2684	7.85002e-06	9.59431e-05	1.30949e-12
2186	2.24369	81.21	7.91233	61.7263	1380	0.0422	400.61	8.654	51.5408	6.69215e-06	9.95646e-05	1.31986e-12
****	fobj	A0=60	GB=10	MF=60	AREA=5000							
2400	3.03708	79.78	11.5346	52.2784	1476	0.0732	407.48	8.71972	53.7553 1	.27437e-05	9.88855e-05	1.00585e-12
3343	3.04095	80.66	11.7288	51.2403	1516	0.063	406.86	10.1855	54.5256	1.12325e-05	9.89793e-05	1.00943e-12
2632	3.03658	80.27	12.4103	49.4531	1480	0.063	406.47	8.81529	55.227	1.11202e-05	9.91297e-05	1.00006e-12
3053	3.04045	80.18	11.9405	50.8337	1544	0.063	406.38	9.87092	57.2908	1.01206e-05	9.87569e-05	1.00051e-12
3438	3.03443	79.47	12.0302	50.7579	1408	0.0732	407.27	7.73292	51.4964	1.38461e-05	9.95105e-05	1.00011e-12
****	fobj	A0=100	GB=1	MF=60	AREA=5000							
2064	2.49601	87.13	2.66746	75.5049	2028	0.017	393.13	10.7289	87.6035	1.39224e-06	6.89378e-05	1.18609e-12
3088	2.47061	86.62	2.36696	75.8284	1380	0.0164	389.12	4.42003	61.4469	1.20338e-06	4.25175e-05	1.0365e-12
2354	2.48498	87.53	2.2505	75.1994	1488	0.0156	388.03	6.39686	63.2858	1.00149e-06	3.64576e-05	1.06519e-12
3201	2.48811	84.03	2.33055	77.8771	1152	0.0183	390.43	3.03774	51.2767	1.80423e-06	5.36301e-05	1.02752e-12
2075	2.48088	86.03	2.22478	76.2211	1152	0.0173	387.73	5.57246	46.6902	1.47365e-06	3.97493e-05	1.21842e-12
****	fobj	A0=100	GB=5	MF=60	AREA=5000							
2482	3.17519	81.13	8.16244	62.1781	1556	0.0528	405.53	10.353	58.2069	8.43633e-06	9.79107e-05	1.00449e-12
2747	3.18136	81.93	8.3544	60.424	1492	0.0423	403.43	13.0834	50.3047	6.80914e-06	9.59674e-05	1.00004e-12
2688	3.20267	82.64	7.79365	61.115	1548	0.036	400.94	10.2072	59.7973	5.54048e-06	8.97167e-05	1.22775e-12
3215	3.19974	83.05	8.44276	59.786	1876	0.0423	405.15	15.6498	67.5695	6.73907e-06	9.99256e-05	1.09315e-12
2353	3.23024	78.06	8.86989	43.012	1476	0.247	408.35	9.92369	54.1064	8.60195e-06	9.69506e-05	1.00906e-12
****	fobj	A0=100	GB=10	MF=60	AREA=5000							
2697	4.00204	80.33	12.0957	50.3363	1520	0.063	406.53	9.61118	56.2469	1.07666e-05	9.84509e-05	1.00675e-12
2849	4.01412	79.89	11.7361	51.3848	1464	0.0733	407.19	9.10913	52.5483	1.28074e-05	9.91731e-05	1.04876e-12
2479	4.00496	81.16	12.2857	49.0665	1460	0.0528	405.16	11.2131	50.7681	9.17967e-06	9.99987e-05	1.00251e-12
3893	4.00796	80.94	12.5224	48.557	1472	0.0528	404.84	10.7298	52.5641	8.58251e-06	9.9788e-05	1.00181e-12
2943	4.00397	80.46	12.0991	50.2016	1520	0.063	406.46	9.69024	55.8435	1.08176e-05	9.99656e-05	1.02297e-12

Tabla IV.3: Resultados para el Amplificador Operacional de Dos Etapas

Apéndice V

Glosario de Términos Traducidos

Apagado Simulado: Simulated Quenching. Relajación del criterio de descenso de la temperatura del algoritmo Enfriamiento Simulado.

Búsqueda Tabú: Tabu Search. Algoritmo sugerido por F.Glover de búsqueda aleatoria.

Celda de paso: Feedthrough. Celdas especial que permiten el paso de conexiones a través de las filas de una disposición.

Celdas Estándar: Standard–Cell. Estilo de emplazamiento de celdas que permite el uso de librerías de celdas precaracterizadas con una buena compactación. Las celdas se ubican en filas.

Circuitos patrón: Benchmarks. Circuitos de prueba aceptados como evaluadores de los nuevos algoritmos.

Colocación, Emplazamiento: Placement. Es el proceso de situar en la oblea de silicio los elementos que componen un circuito integrado.

Conexionado: Routing. Fase de la disposición destinada al posicionamiento de las conexiones una vez emplazada cada celda.

Conexionado Global: Global Routing. Fase del conexionado en la que se asigna un canal de conexiones a cada conexión, sin especificar exactamente cuál es el camino que recorre. También se determina la posici—'on de las celdas de paso.

Conexionado de Detalle: Detailed Routing. Fase del conexionado en la que se determina exactamente la posición y la trayectoria de cada conexión entre cada celda.

Conexiones Externas: Pad's. Son elementos que permiten conectar mediante hilos a las patas del encapsulado el "dado" de silicio.

Contador Interno: Within Count. Es un criterio utilizado en el algoritmo Enfriamiento Simulado.

Corredores: Highways. Son espacios donde se ubican las conexiones verticales de un circuito.

Disposición: Layout. Se conoce como el proceso global de disponer, situar y unir todos y cada uno de los elementos de un circuito.

Distribución: Floorplanning. Es la planificación de la distribución desde un punto de vista estratégico de los elementos del circuito.

Enfriamiento Simulado: Simulated Annealing. Algoritmo de Búsqueda Aleatoria aplicado a la disposición de circuitos integrados, basado en un símil físico de enfriamiento de un sólido.

Evolución Estocástica: Stochastic Evolution. Algoritmo de Búsqueda aleatoria aplicado a la disposición de Circuitos Integrados.

Filas: Rows. Disposición de las celdas característica de los circuitos parcialmente a medida.

FIFO: First Input First Output. Es una estructura de cola de elementos de tamaño fijo, donde el primer elemento que entra es el primero en salir, cuando toda la cola está llena y no hay posiciones vacías.

Matriz de Puertas: Gate Array. Estilo de disposición en el que se predifunden sobre la oblea parejas de transistores p y n, reservando espacios para el conexionado. La disposición de los transistores es en filas.

Mar de Puertas: Sea of Gates. Estilo de disposición en el que se cubre la oblea con transistores p y n y se metaliza posteriormente sobre ellos, utilizando unos e inutilizando otros debido al conexionado del circuito.

Parcialmente a Medida: Semicustom. Estilo de disposición en el que las celdas a las que se hace referencia son las precaracterizadas por el fabricante.

Parrilla de puntos: grid. Es una parrilla imaginaria donde se situan las diferentes geometrías de los circuitos.

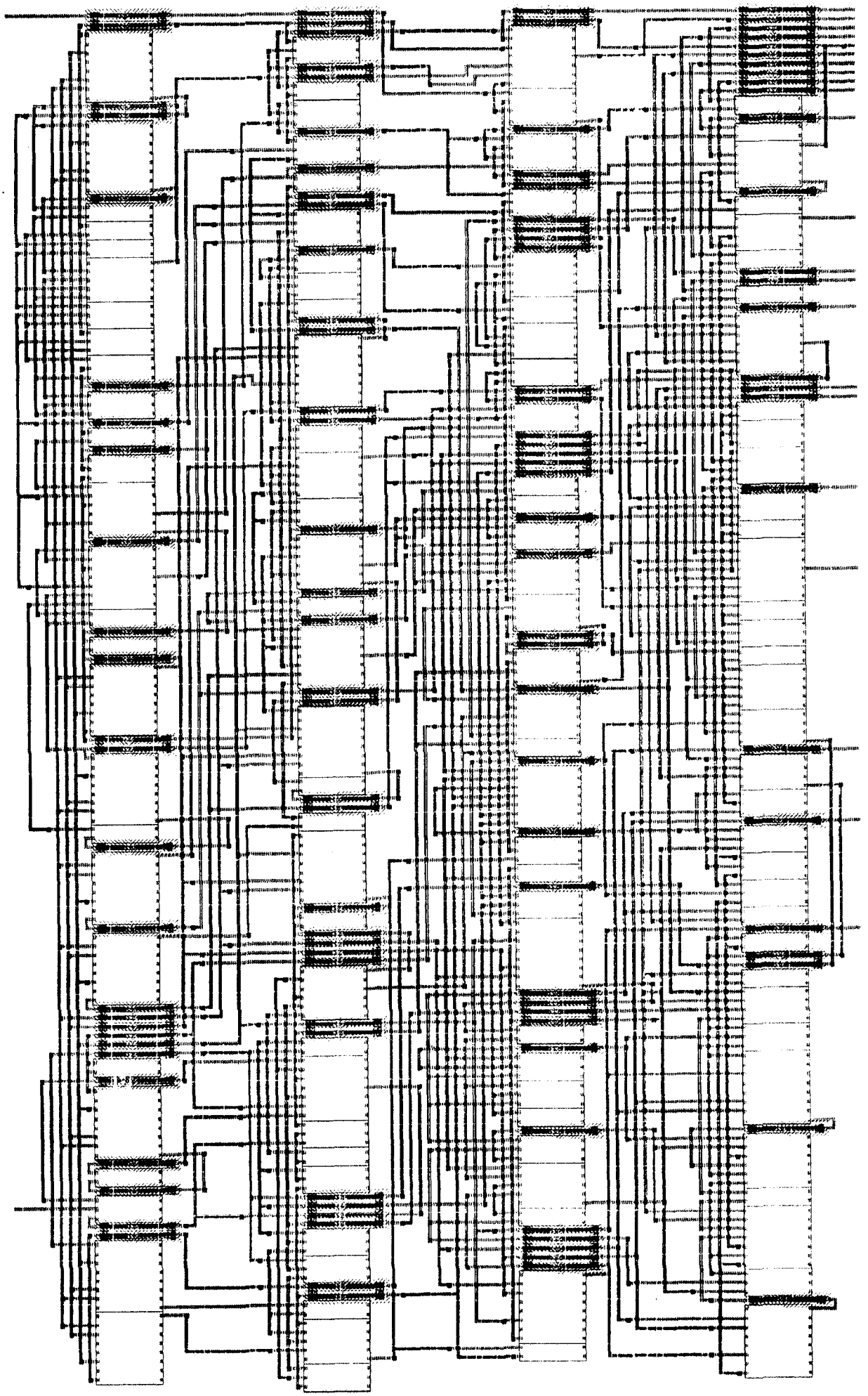
Subagrupaciones de celdas fuertemente interrelacionadas: Clusters

Totalmente a Medida: Full–Custom. Estilo de disposición en el que el diseño que recibe el fabricante es un conjunto de dimensiones de transistores y conexiones.

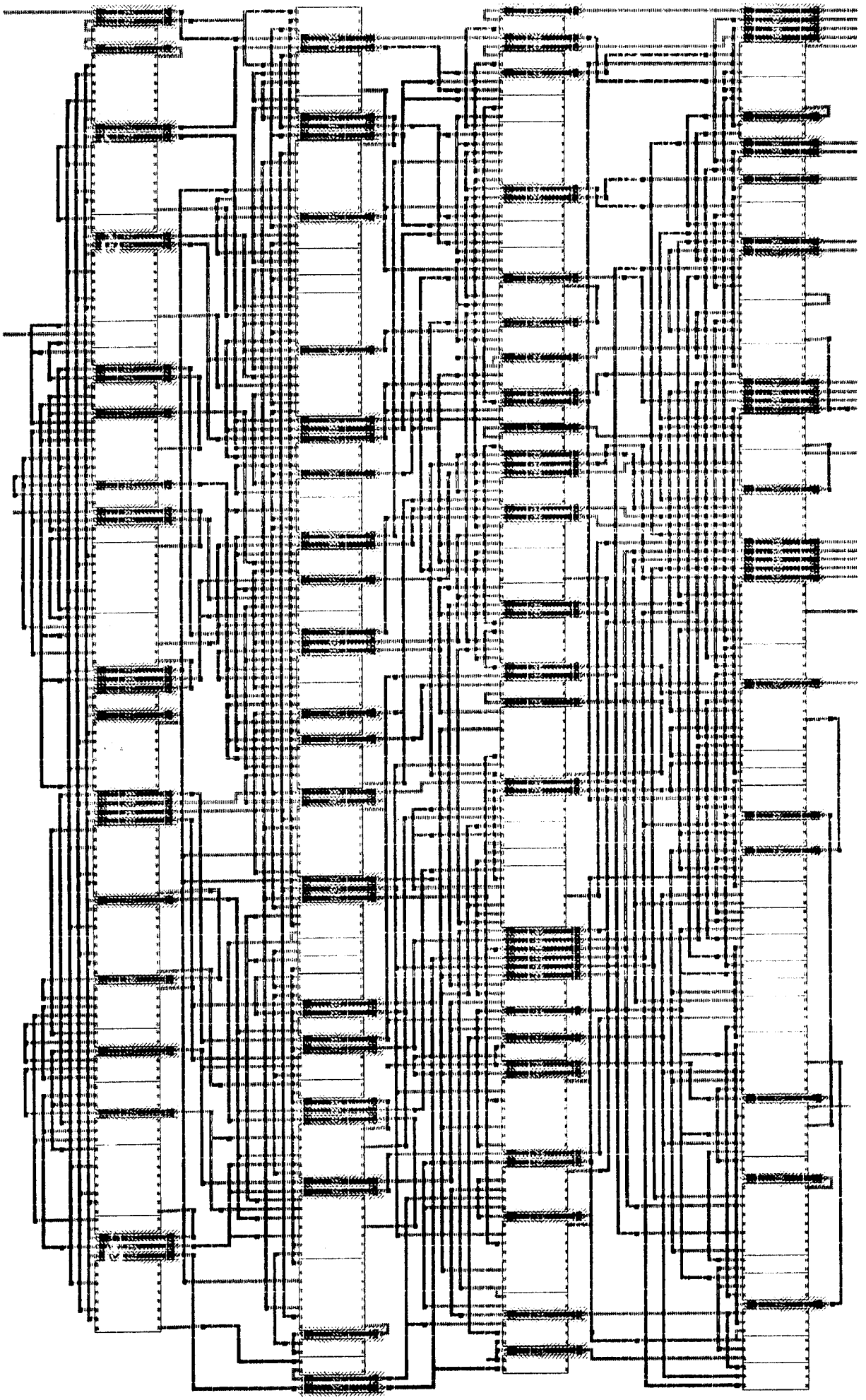
Referencial del Circuito: netlist. Es un conjunto de referencias entre redes y celdas que describe el conexionado eléctrico del circuito.

Apéndice VI

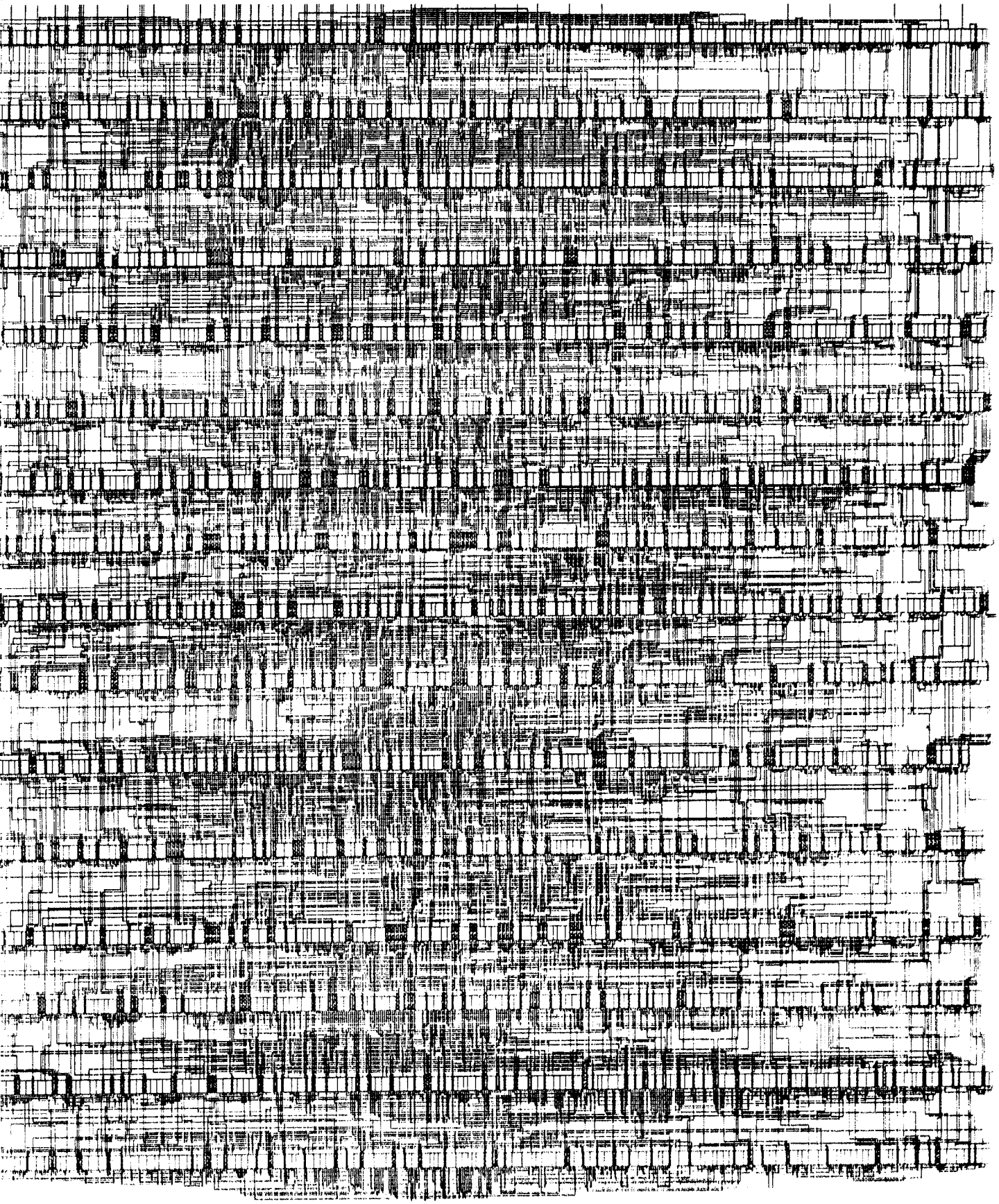
Disposiciones finales de algunos circuitos patrones (layouts).



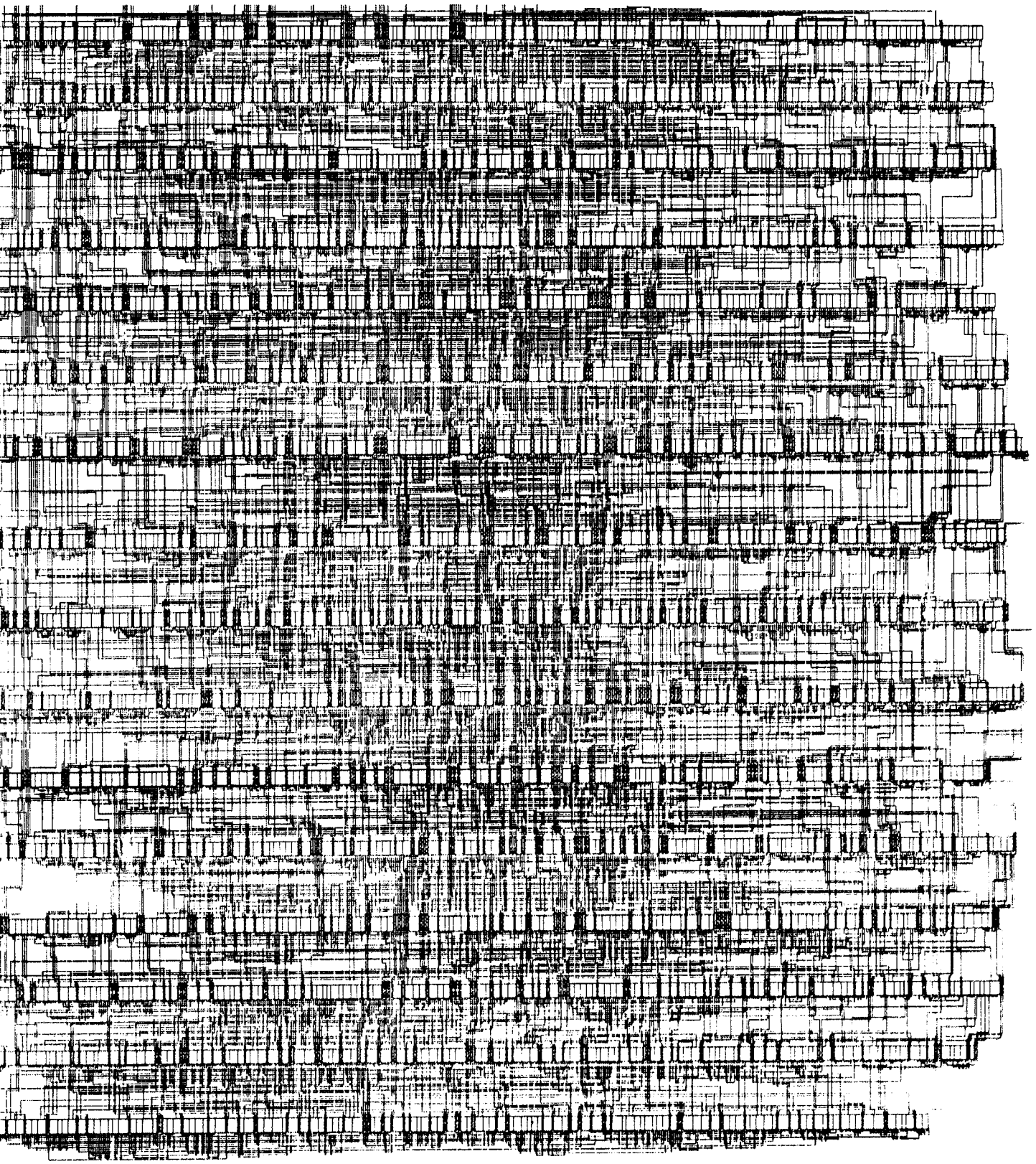
Resultado de la disposición del circuito *fract* por TimberWolfSC (área calculada: 1095057).



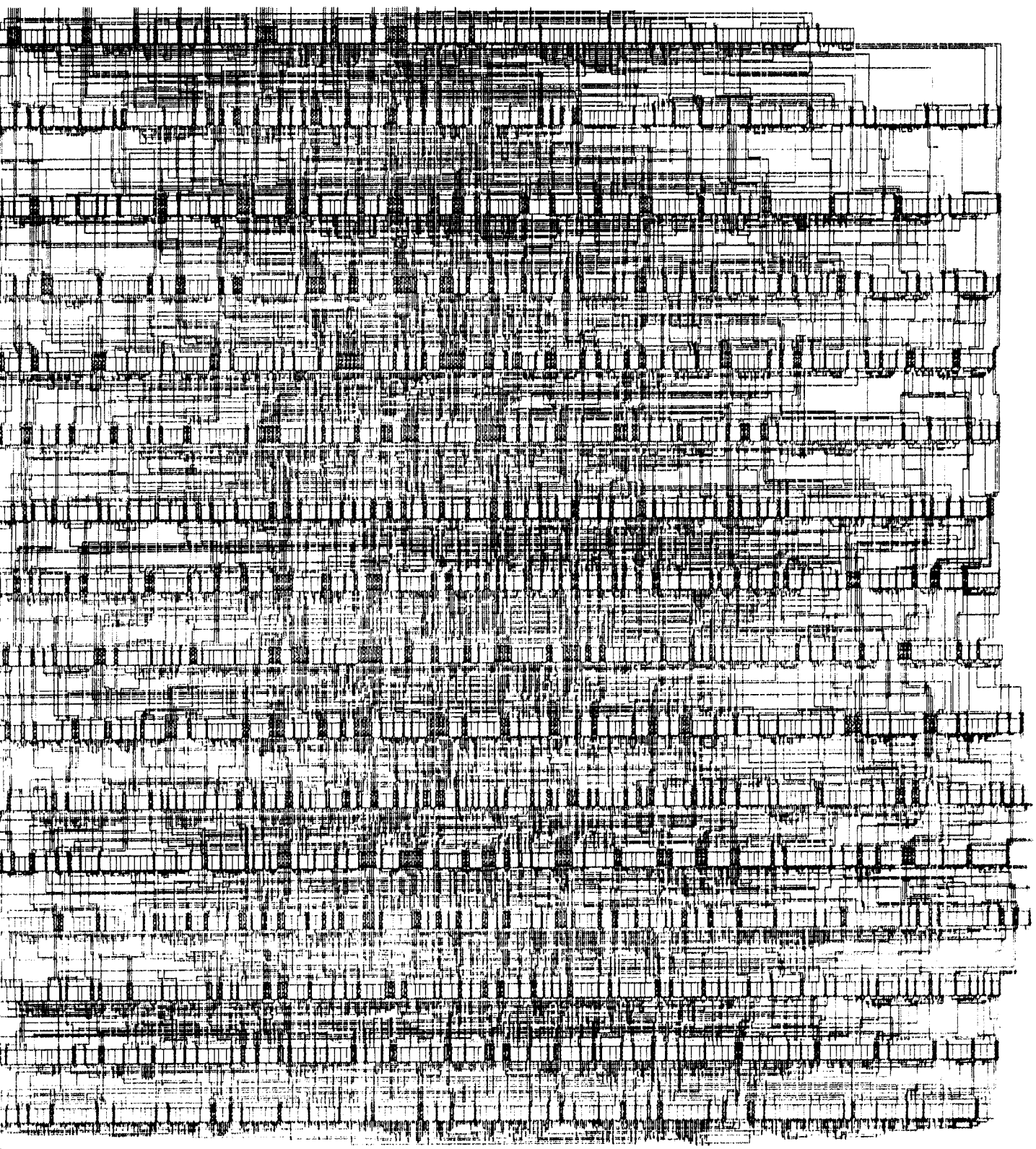
Resultado de la disposición del circuito *fract* por foxy (4 bloques) (área calculada: 946770).



Resultado de la disposición del circuito *struct* por (4 bloques) (área calculada: 20500121).



Resultado de la disposición del circuito *struct* por TimberWolfSC (área calculada: 20264114). .



Resultado de la disposición del circuito *struct* por (16 bloques) (área calculada: 19822491).