UNIVERSIDAD DE DEUSTO

Facultad de Informática

NUEVOS ALGORITMOS PARA EL DISEÑO OPTIMIZADO DE ARRAYS LINEALES BASADOS EN CÉLULAS SERIE-PARALELO

MEMORIA

QUE PARA OPTAR AL GRADO DE DOCTOR EN INFORMÁTICA

PRESENTA

José Luis Gutiérrez Temiño

DIRECTOR: Dr. D. José María Angulo Usategui

MAYO, 1995

ÍNDICE

RESUMEN

El diseño de circuitos integrados se ha convertido en la actualidad en un proceso tan complejo, que impone el uso casi obligatorio de algoritmos específicos desarrollados sobre herramientas informáticas. En este sentido se han planteado múltiples teorías que nos dan en la mayoría de los casos soluciones heurísticas más o menos acertadas. El problema fundamental en el desarrollo de dichos algoritmos estriba en la obtención de layouts óptimos desde el punto de vista de la superficie de silicio empleada, así como del tiempo de propagación de las funciones implementadas, consumo total, etc. Gran parte de los trabajos de investigación realizados en ésta línea se ocupan de la optimización de un único parámetro o de una mera aproximación al diseño óptimo, esto es debido a la dificultad que supone el tratamiento simultáneo de varios aspectos, por su interrelación.

Nuestra tesis plantea un nuevo método no heurístico con el cual se obtienen optimizaciones totales de la velocidad de propagación y de la superficie de silicio ocupada en la creación de layouts para arrays lineales, basados en células serie-paralelo, independientemente de la complejidad de los mismos. Además, se ha desarrollado un sistema de recorrido y análisis de grafos, con el cual se logra que nuestro método consuma un tiempo de proceso menor.

ABSTRACT

The design of integrated circuits has lately become a very complex process, which has made almost compulsory the use of specific algorithms developed using sophisticated tools. With this in mind, several theories have been proposed but give, in most cases, a near right heuristic solution. The fundamental problem in the development of these algorithms lies in obtaining the optimum layouts from the point of view of the silicon surface used, as well as the propagation time of the implemented functions, the total consumption of energy, etc. Most of the research done in this area deals with the optimization of only one parameter, or a simple approximation to the optimal design, this being so due to the difficulty that the simultaneous treatment of the different aspects implies because of its interrelation.

This Thesis proposes a new non-heuristic method that obtains a total optimization of the rate of propagation and silicon surface in the creation of layouts for lineal arrays, based on parallel-serial cells, regardless of their complexity. Besides, a system of traversal and analysis of graphs has been developed with which our method consume less time of process.

RELACIÓN DE FIGURAS

RELACIÓN DE TABLAS

1. INTRODUCCIÓN

El primer circuito integrado o C.I. se desarrolló en 1961 y estaba formado por unos pocos transistores. Con el desarrollo de la tecnología este número se ha incrementado sustancialmente. Así, hoy en dia la densidad de los C.I es tal que en muchos casos rebasan los 10^6 transistores; tal es el caso de los circuitos VLSI. La complejidad que supone su diseño hace necesaria la utilización de herramientas de diseño por ordenador y técnicas de diseño jerárquico (Top-Down) que partiendo de unas especificaciones dadas generan el layout correspondiente.

El proceso de diseño top-down se realiza en distintas fases, aplicándose en cada una de ellas, técnicas de optimización basadas en la programación dinámica que darán como resultado un C.I. ajustado a las especificaciones fijadas inicialmente. En la figura 1-1 se muestran las diferentes fases básicas que cubren el diseño de un C.I.

Inicialmente, se definen las especificaciones del C.I. (funciones, tiempos y consumos) mediante herramientas tales como lenguajes de síntesis, cronogramas, etc. Habitualmente estas especificaciones determinan una complejidad en el diseño que aconseja la realización del particionamiento del problema en módulos funcionales interconectados. A continuación, se ubican los módulos y sus conexiones en la superficie asignada al C.I. (Floorplanning). Dichos módulos denominados arrays celulares se implementan por medio de células básicas, definidas cada una de ellas por una

función lógica o puerta. Posteriormente, dichas células se convierten en circuitos basados en redes de transistores. Finalmente, estos transistores y sus interconexiones se representan por medio de un layout o máscara que proporciona, a través de procesos físico-químicos, las incrustaciones necesarias sobre la superficie activa del circuito integrado, de los distintos tipos de materiales que lo constituyen. En el capítulo 2, se detalla parte de este proceso con la notación y terminología utilizada en el presente trabajo de investigación.

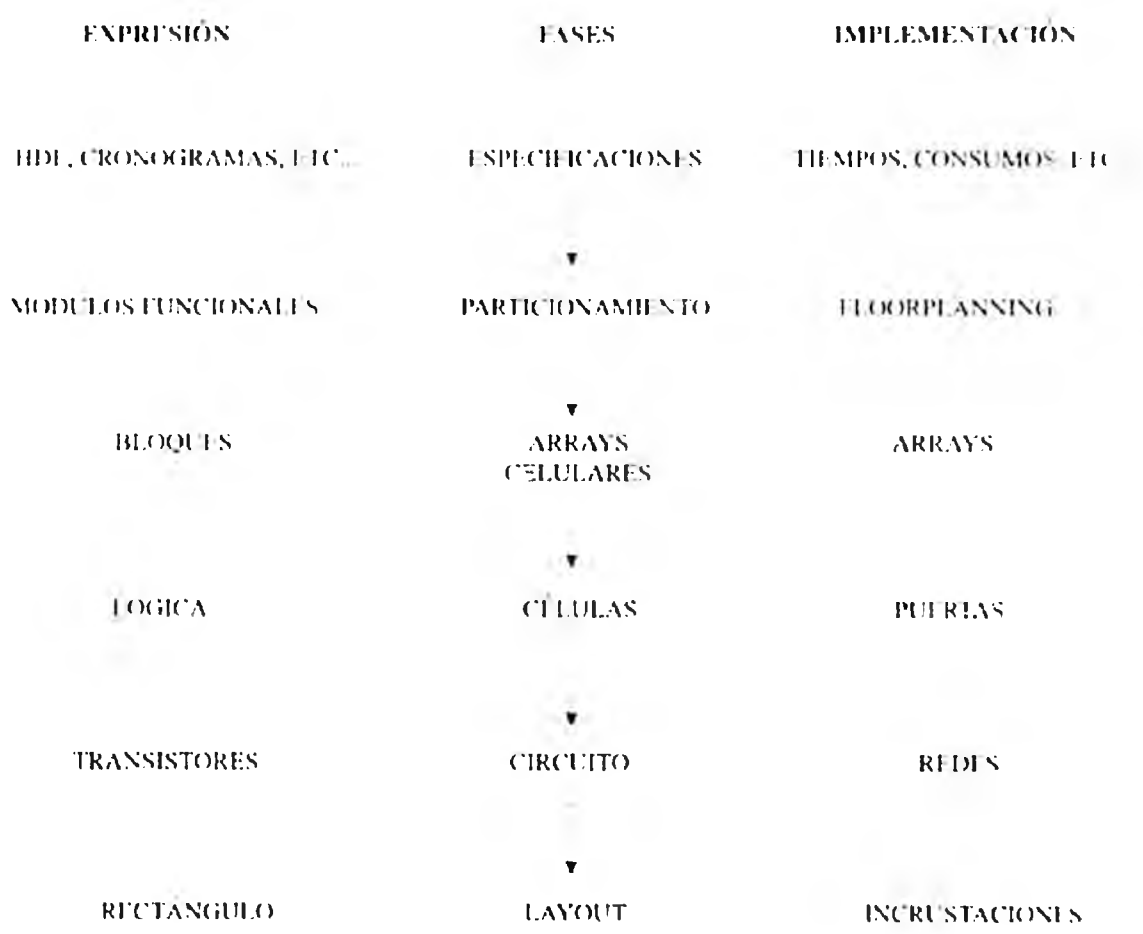
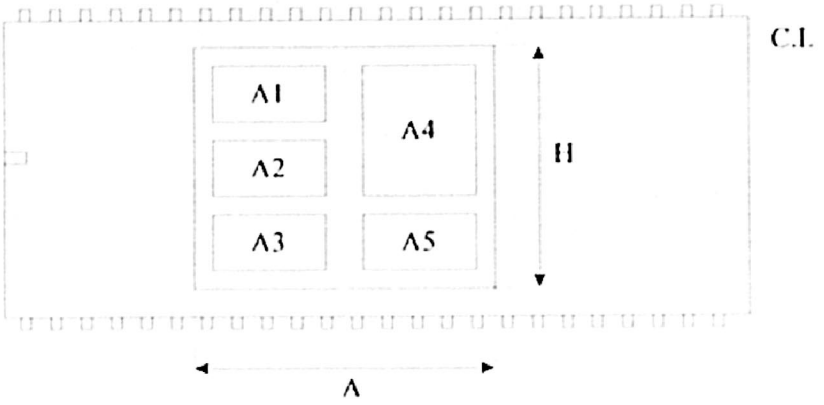


Figura 1-1 Fases de diseño de un CI

El diseño óptimo de un circuito integrado implica la resolución de una serie de problemas, al margen de las especificaciones funcionales (ecuaciones lógicas). Estos problemas pueden dividirse en dos grandes grupos: los intrínsecos al C.I. como son el área, la velocidad, el consumo, la tensión, etc., y los relacionados con el entorno tales como la inmunidad al ruido, las temperaturas operativas, los niveles de E/S, etc.

Cada fase de diseño requiere, según las especificaciones iniciales, unos valores para los parámetros asociados a estos problemas, que condicionan la totalidad del diseño. Así, el área de un array afecta al Floorplanning; el tiempo de propagación de una célula o puerta, puede determinar la velocidad del array al cual pertenece, etc. Resulta por tanto, muy complejo abordar simultáneamente la optimización de todos los parámetros que intervienen, debido a la fuerte interacción existente entre ellos. A medida que se abarca la resolución u optimización de más parámetros, el tiempo de diseño aumenta exponencialmente, pudiendo llegar a valores inoperantes si se pretende crear un algoritmo que alcance una optimización total; por consiguiente, se aceptan métodos heurísticos que manejan varias problemáticas al mismo tiempo. El objetivo del presente trabajo de investigación es la optimización de los parámetros de velocidad y área por lo que pasamos a describirlos en profundidad.

El área, a nivel de circuito integrado, está determinado por la anchura (A) y la altura (H) de la superficie activa (Figura 1-2). Lógicamente éstas serán el resultado del tamaño de los arrays, de la ubicación de los mismos y de la superficie ocupada por las zonas de conexión.



(A1-A5) : Arrays
( ) : Zonas de conexión
(H,A) : Altura y anchura respectivamente
Area de la superficie activa = HxA

Figura 1-2 Floorplanning o Planta de un C.I.

A nivel de array, el área se corresponde con la suma de las áreas de todas las células que lo componen, más el área perteneciente a la zona de conexiones entre las células.

Un tipo particular de array celular es el array lineal [MAZI91]; esta caracterizado por tener las células ubicadas secuencialmente en horizontal y con sus conexiones en las zonas superior e inferior. Horizontalmente el array queda delimitado por las líneas de alimentación de las células que lo componen (Figura 1-3).

La anchura de este tipo de array está determinada por la suma de las anchuras de las células. La optimización de este parámetro implica la optimización de la anchura de cada una de las células, más la necesaria, en ciertos casos, para el aislamiento entre ellas. La altura del array será la máxima de entre todas las alturas de las células, teniendo en cuenta las zonas de conexión y las constantes de altura para las alimentaciones.

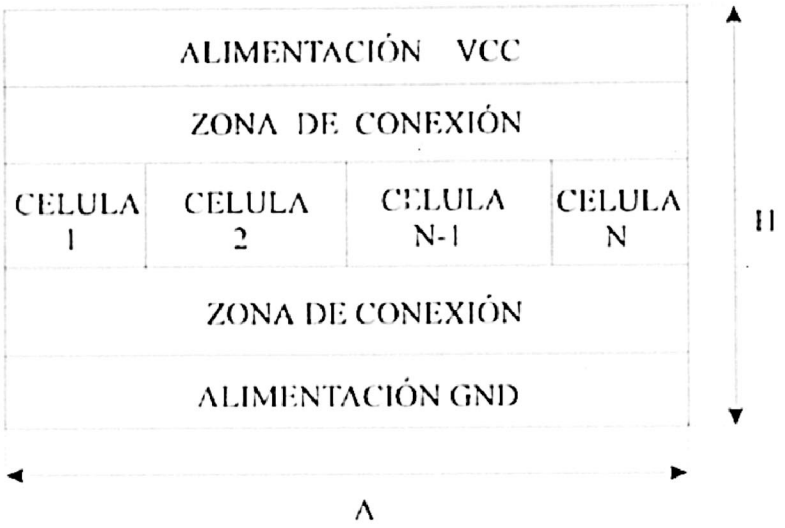


Figura 1-3 Composición de un array lineal.

La reducción de la altura para el array supondrá una reducción de la misma para las células y una optimización en las conexiones por medio de permutaciones, tanto de las células en el array como de los transistores que configuran cada una de las células. En la figura 1-4 se muestra un ejemplo de permutación celular y su impacto en la altura de las zonas de conexión.

Las células que forman los arrays están constituidas por transistores, y la problemática del área en ellas es muy similar a la que se representa en los arrays. Los transistores están distribuidos en dos líneas horizontales y cada una de ellas corresponde a una red, que en el caso de las células CMOS serie-paralelo son idénticas respecto al número de transistores y su ubicación, pero no en cuanto a las conexiones dentro de cada célula. La anchura de la célula será la suma de las anchuras de los transistores, igual para todos ellos, más la asignada a la separación entre los mismos mediante difusores de vacío. Análogamente al array, una correcta permutación de los transistores permite

reducir la anchura; la disposición adyacente del mayor número posible de transistores, anula las anchuras correspondientes a la separación entre ellos (difusores por incrustación).

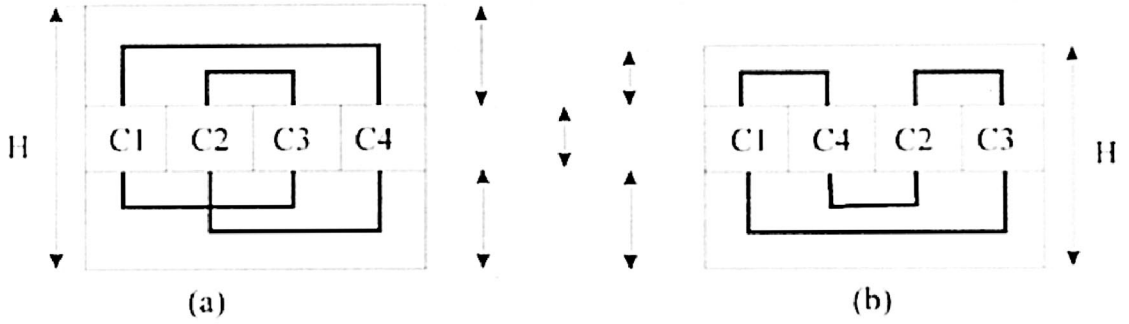


Figura 1-4 Conexiones entre células de un array.

Un problema distinto es la altura; en este caso, la optimización afecta a las conexiones entre los transistores. De la misma manera que ocurría en los arrays, una reordenación de los transistores puede implicar una reducción de la altura de alguna de las dos redes o de ambas como se indica en la figura 1-5.

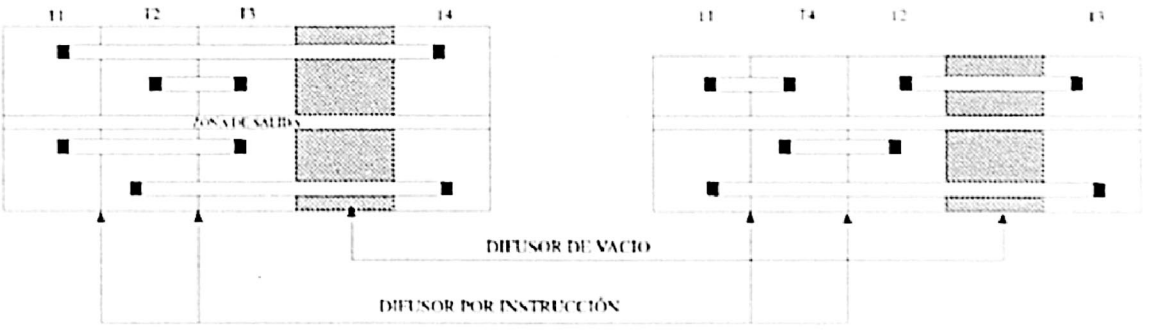


Figura 1-5 Conexión de los transistores en una célula.

Si se observa el array a nivel de circuito, todo él se resume en un conjunto de transistores y una red de conexiones entre ellos que genera un retraso resultante de la suma de los retrasos, tanto de los transistores que intervienen en el recorrido como del generado en las líneas de conexión que recorre un nivel lógico desde la entrada hasta la salida. Su cálculo es altamente complejo debido a la variedad de materiales y de tamaños de las uniones, así como a los distintos dimensionados de los transistores. Esto obliga a modelizar unas y otros, para obtener resultados comparativos y en su caso, conclusiones.

La razón de estos retardos hay que buscarla en el propio layout. Las distintas capas con los diferentes materiales utilizados en el diseño del circuito integrado generan entre ellas unas capacidades parásitas (Figura 1-6), las cuales se cargan o descargan con cada conmutación digital. Además, dichos materiales representan una resistencia específica que unida a las capacidades parásitas generan un conjunto de redes RC, cuyos valores son dependientes del recorrido realizado por una señal digital.

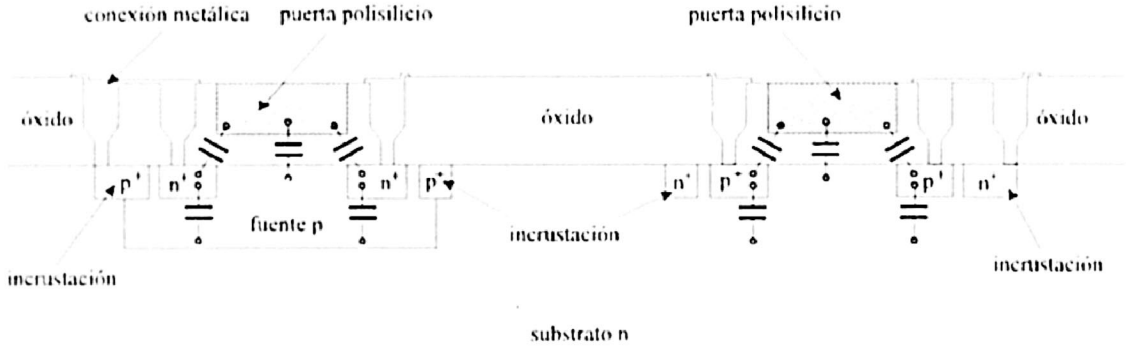


Figura 1-6 Capacidades presentes en un layout.

Para el estudio de la velocidad y del retardo, los transistores pueden modelizarse como una resistencia en serie y una capacidad en paralelo. De la misma manera, una conexión entre dos puntos del C.I. se puede considerar como una resistencia (R) y una capacidad (C) dependientes de la longitud de la conexión, del tipo de material y de la sección del mismo (Figura 1-7).

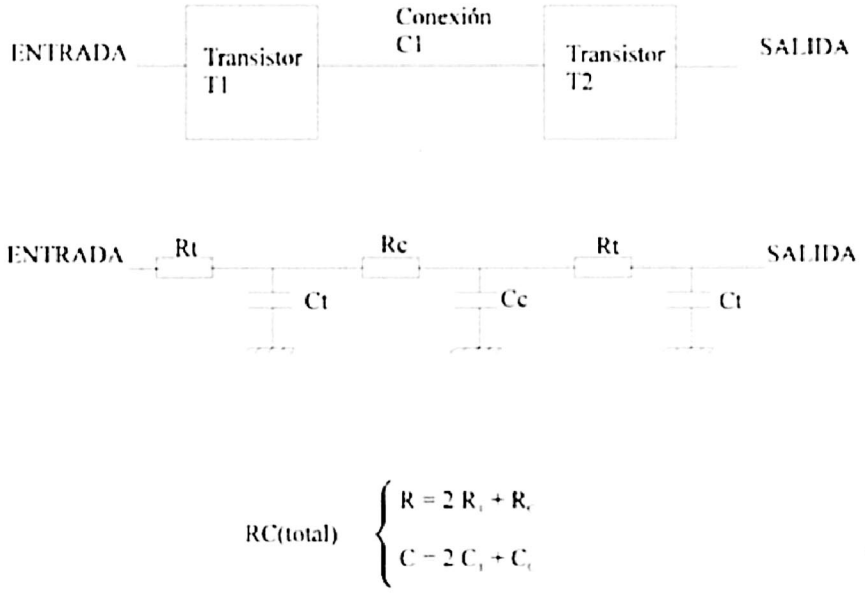


Figura 1-7 Equivalencia eléctrica de los elementos en un C.I.

En consecuencia se pueden afirmar las siguientes sentencias:

a) Considerando todas las conexiones con la misma sección y tipo de material, así como transistores con un tiempo de propagación constante, la velocidad de la célula, array o C.I., depende exclusivamente de la longitud de las conexiones en el nivel correspondiente.

b) Una reordenación de los transistores dentro de una célula y de éstas en los arrays, implicaría resultados funcionales idénticos con distintas velocidades.

c) Las reordenaciones anteriores afectan simultáneamente al área y velocidad en cualquiera de los niveles de diseño. Existe una relación muy estrecha entre ambos problemas.

1.1 DESARROLLO HISTORICO Y SITUACIÓN ACTUAL

Determinados autores han elaborado algoritmos de optimización de velocidad, anchura y altura. El presente trabajo de investigación presenta un nuevo método de optimización de arrays lineales que ofrece respecto de los modelos anteriores, entre otras, las ventajas siguientes:

- Mayor optimización de la relación velocidad-area.
- Optimización no heurística.
- Una mayor velocidad de proceso.

En la implementación de los arrays celulares (M) se presentan los problemas siguientes (Tabla 1-1):

- Optimización en Área logrando una:
  - a) anchura (A) mínima del array.
  - b) altura (H) mínima del array.
  - c) reordenación (R) del array celular para obtener la solución óptima.
- Optimización en Tiempo (T).

PROBLEMA	OPTIMIZACION EN AREA				OPTIMIZACION EN TIEMPO
	Minimizacion de la anchura	Minimizacion de la altura	Reordenacion del circuito	Numero de celulas	
AM	SI	NO	NO	ARRAYS	NO
AEM	SI	NO	SI	ARRAYS	NO
AHM	SI	SI	NO	ARRAYS	NO
AHKM	SI	SI	SI	ARRAYS	NO
AHKMT	SI	SI	SI	ARRAYS	SI

Tabla 1-1 Clasificacion de optimizaciones posibles

En un primer momento, el objetivo era conseguir un array de mínima anchura pero sin conceder especial atención a la altura. Tal es el caso de Wimer [WIME87], cuyo método minimiza el número de difusores por vacío para unos circuitos de dimensiones muy limitados; sin embargo no logró una optimización en altura y anchura para circuitos complejos. Posteriormente, se confirió idéntica importancia a la altura y a la anchura del array de células. El programa GENAC creado por Ong y Li [ONGL89], describe un sistema de arrays que logra reducir la anchura y la altura disminuyendo el número de canales. Este programa obtiene una solución, pero no la óptima, ya que no considera todas las combinaciones posibles resultantes de la reordenación. Con el propósito de diseñar un algoritmo de resultados óptimos, Maziasz [MAZI91] partió de los siete supuestos propuestos por Uehara y VanCleemput [UEHA81]. El método desarrollado por Maziasz minimiza, principalmente, la anchura de la célula mediante la reubicación de los transistores, reduciendo así el número de transistores conectados por difusores de vacío. Posteriormente, reduce la altura de la célula reordenando el circuito original de diferentes formas hasta dar con la combinación óptima. Una vez conseguida la

optimización celular se resuelve el problema para el array, buscando la combinación óptima de diferentes tamaños de células básicas tal que el conjunto de ellas resulte con un área mínima. Entre todas las implementaciones encontradas, con anchura mínima, selecciona aquella cuya altura sea mínima. Sin embargo, esta optimización a nivel de array resulta heurística por no contemplar todas las orientaciones posibles de las células que lo componen.

Kwong y Kyung [KWON88] desarrollaron un método para lograr un tiempo de proceso óptimo, que consiste en obtener reordenaciones útiles minimizando la anchura del array; sin embargo, no logra la optimización en altura. Otros, como F. Mailhot y G. DeMichelli [MAIL88] se centrarón en optimizar la anchura con respecto a la altura, o bien la altura con respecto a la anchura, pero con unas reordenaciones bastante limitadas. El algoritmo utilizado por H. T. Tu [TU93] minimiza la anchura total de las células dividiendo éstas en grupos, de manera que cambiando simultáneamente las células de un grupo se reduce la anchura total. Además asigna una velocidad a cada célula de tal forma que, reordenándolas y con las velocidades asignadas, logra disminuir la anchura de las células mayores. Para determinar las células que deben ser reordenadas, el algoritmo calcula mediante grafos la topología así como los requerimientos de distancia mínima y máxima entre células. El tiempo de ejecución de éste se calcula mediante la expresión O (n log n), siendo n el número de células.

PROBLEMA	REFERENCIA	NUMERO DE CELULAS
AR	Uehara y VanCleemput [UEHA81]	CELULA
AR	Nair, Brass y Reid [NAIR85]	CELULA
AHR	Leiserson, Chan y Martin [LEIS89]	CELULA
AHR	Madsen [MADS89]	CELULA
AHM	Wimer y otros [WIMI87]	ARRAYS
AHM	Ong, Li y Lo [ONGI89]	ARRAYS
AHRM	Mazumder [MAZU91]	ARRAYS
AHRT	Kwon y Kyung [KWON88]	CELULA
AHRMT	Maalbot y DeMicheli [MAIL88]	ARRAYS
AHRMT	Lo [LO91]	ARRAYS

Tabla 1-2 Relación de trabajos de optimización realizados

Una vez lograda la optimización del área, procede obtener la optimización en el tiempo. Como resultado de los avances tecnológicos, las dimensiones en área disminuyen y adquiere mayor relevancia la optimización de la velocidad. El tiempo es otro de los parámetros a resolver pero la información disponible es muy limitada. Además, las pruebas que se realizan se reducen a circuitos más o menos comunes, sin analizar lo que ocurriría con otros más complejos, aunque no por ello menos utilizados. Los algoritmos empleados tienen una curva exponencial para los tiempos de proceso y por lo tanto se hacen inviables. Resolver eficazmente el problema del tiempo no supone priorizar éste y dejar en un segundo plano la minimización del área, sino que ambos objetivos deben ser conjugados sutilmente para lograr una óptima implementación del array.

1.2 OBJETIVO DE LA TESIS

Ante la imposibilidad de lograr una optimización de todos los parámetros que intervienen en el diseño de un C.I., por la enorme complejidad que supone simultanear todos así como el elevado tiempo de proceso, el objetivo principal del presente trabajo de investigación es obtener el array lineal más optimo en velocidad con la menor anchura posible.

Diversos investigadores han logrado arrays optimos en velocidad y anchura (Tabla 1-2), sin embargo o no pueden garantizar que los arrays obtenidos sean los más óptimos por haber utilizado metodos heuristicos ó sólo son válidos para células muy sencillas.

Nosotros vamos a desarrollar un método no heuristico y válido para cualquier celula utilizando nuevas técnicas de construcción y recorrido de grafos. Este método implicará un proceso que cubrirá las siguientes etapas:

a) Optimización en velocidad de las células que constituyen el array.

b) Construcción del array con las células optimizadas.

c) Cálculo de la velocidad y anchura de este array y de las holguras en tiempo de cada célula en él.

d) Optimización en anchura de las células con holgura.

e) Sustitución en el array de estas células por las mismas ahora optimizadas.

f) Evaluación de la velocidad y anchura del nuevo array.

Este proceso se realizará para cada una de las posibles permutaciones de las células en cada array, así como para todas las soluciones optimas generadas para una misma célula.

Para su implementación se ha desarrollado un generador de arrays óptimos o GAO que se compone de un conjunto de generadores y tres algoritmos: A35 o algoritmo de optimización de velocidad y anchura de células serie-paralelo, ALG1 o algoritmo para el cálculo de la anchura del array y tiempo de conexiones entre células y ALG2 o algoritmo para el cálculo del tiempo de propagación y relación de células optimizables en anchura.

2. DEFINICIONES Y TERMINOLOGÍA

2.1 DEFINICIONES GENERALES

Presentamos una breve descripción de las estructuras empleadas en el diseño de las células y los arrays, objeto de la optimización propuesta, así como la terminología básica del diseño de circuitos. Las definiciones particulares de nuestra investigación, las iremos dando a medida que las necesitemos.

2.1.1 Transistores MOS

La tecnología CMOS proporciona dos tipos de transistores, un transistor de tipo N (NMOS) y un transistor de tipo P (PMOS). Son fabricados con silicio usando o bien silicio dopado negativamente, o bien silicio dopado positivamente.

La estructura del transistor N consiste en un substrato ó sección de silicio de tipo P (p-dopado) separando dos zonas de silicio de tipo N, y la del transistor P en un substrato de silicio de tipo N (n-dopado) separando dos zonas de silicio tipo P. En la figura 2-1 se muestra la estructura física típica de estos transistores MOS.

Para explicar su funcionamiento asumiremos que los transistores tienen dos conexiones adicionales llamadas fuente y drenador, que han sido formadas por regiones conductoras N o P en el caso de un dispositivo P.

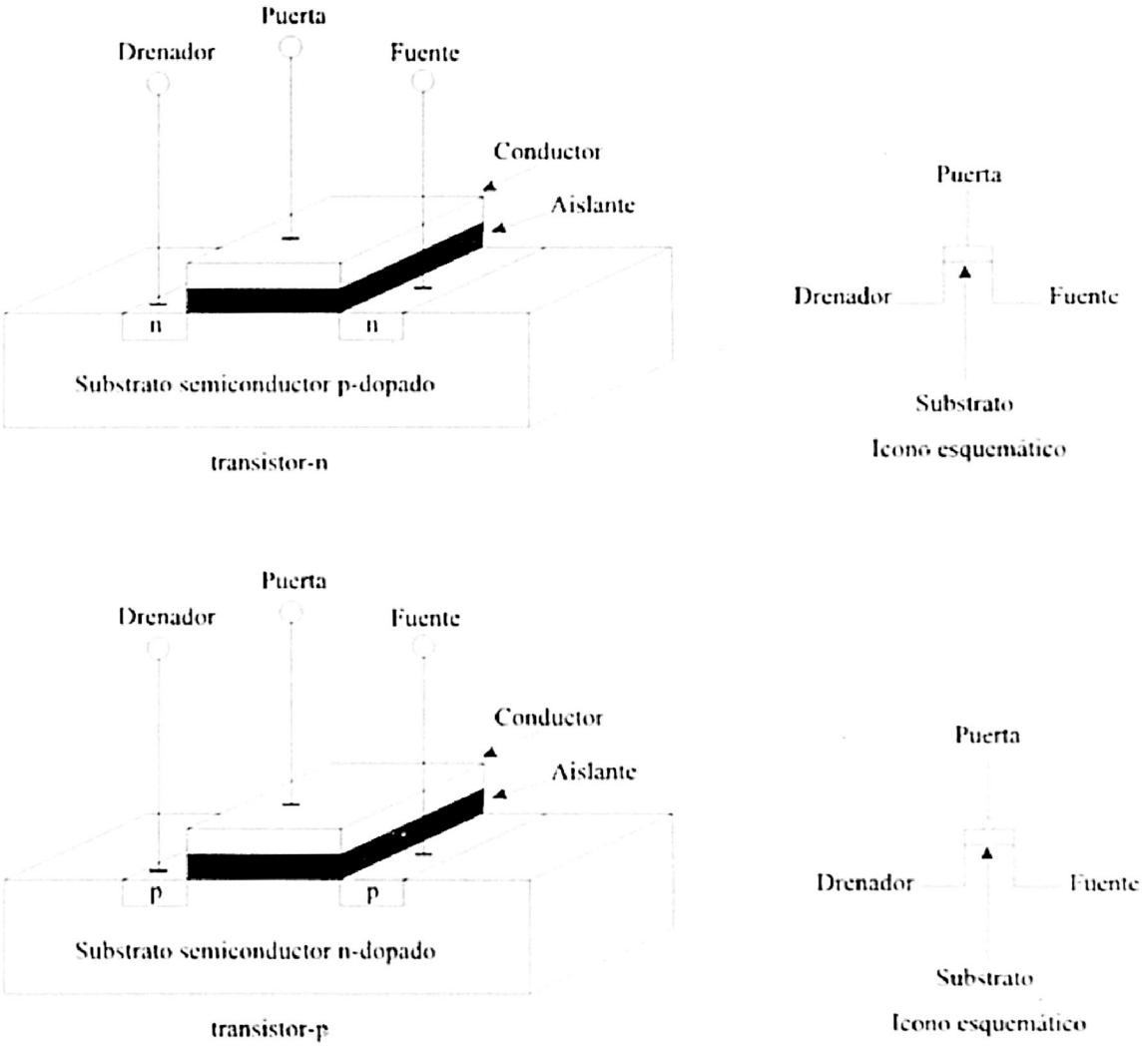


Figura 2-1 Estructura física y representación gráfica de los transistores MOS

El drenador y la fuente son físicamente equivalentes, el nombre asignado depende de la polarización. La puerta es un control de entrada que afecta al flujo de corriente

eléctrica entre ellos, permitiendo a los transistores MOS ser vistos como simples interruptores.

A partir de ahora supondremos que un "1" es un voltaje situado entre 4,37 y 5 voltios. A este voltaje se le designa VDD. El símbolo "0" corresponderá con un voltaje bajo GND, situado en 0 voltios.

Como se puede apreciar en la figura 2-2a el transistor NMOS puede ser expresado como un interruptor. En ella, la puerta se identifica con 's', el drenador con 'a' y la fuente con 'b'.

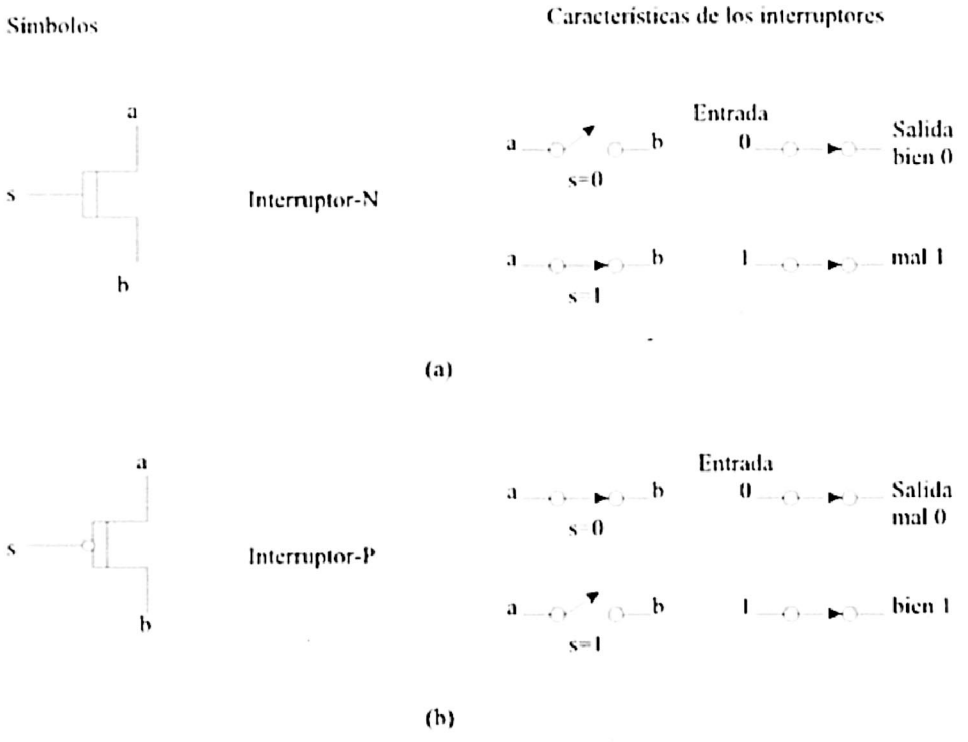


Figura 2-2 Interruptores NMOS y PMOS, símbolos y características.

Un interruptor N se considera cerrado o encendido si el drenador y la fuente están conectados, y se dice que es perfecto cuando se desea propagar un nivel 0. Sin embargo, éste presenta un malfuncionamiento si la propagación se realiza con un "1". Por otra parte, el transistor PMOS o interruptor P (Figura 2-2b), tiene propiedades inversas que su dual. La figura 2-2 muestra las conclusiones de funcionamiento de cada interruptor (tipo de transistor).

A continuación, se muestra un resumen (Tabla 2-1) de la respuesta de estos transistores frente a los niveles lógicos 0 y 1.

NIVEL	SÍMBOLO	CONDICIÓN INTERRUPTOR
1 Fuerte	1	P-INTERRUP puerta=0 fuente=VDD
1 Débil	1	N-INTERRUP puerta=1 fuente=VDD ó P-INTERRUP conectado a VDD
0 Fuerte	0	N-INTERRUP puerta=1 fuente=GND
0 Débil	0	P-INTERRUP puerta=0 fuente=GND ó N-INTERRUP conectado a GND
Alta impedancia	Z	N-INTERRUP puerta=0 ó P-INTERRUP puerta=1

Tabla 2.1. Niveles lógicos de salida para transistores PMOS y NMOS.

2.1.2 Redes CMOS

La tecnología CMOS se utiliza para representar el funcionamiento interno de cualquier célula. Una función lógica implementada con esta tecnología se basa en el siguiente principio:

"Cada unidad lógica se representa internamente por dos redes de transistores: Una red PMOS que permite conectar la salida de la célula a la tensión de alimentación VDD, y otra NMOS que posibilita la conexión de la salida a GND. Ambas redes son controladas por las puertas de los transistores, las cuales son variables de una función lógica" (Figura 2-3).

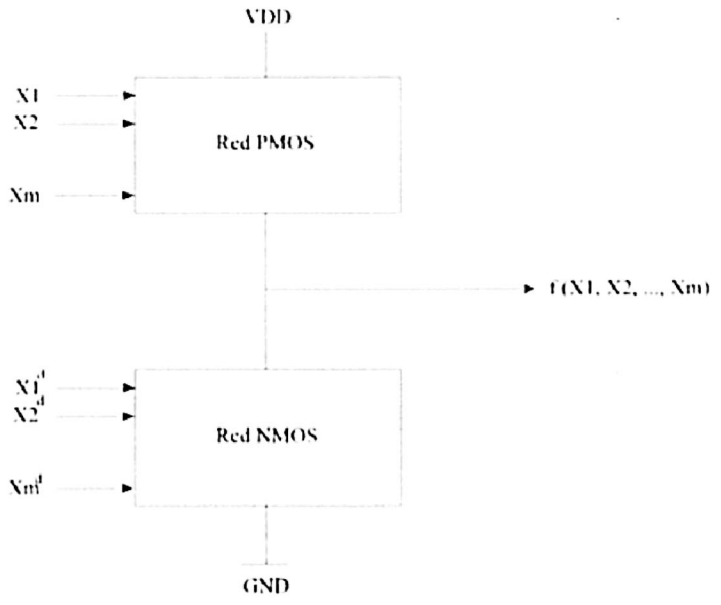


Figura 2-3 Modelización Neuronal de una red CMOS

El principio anterior se puede formalizar mediante el concepto de admitancias lógicas, de la siguiente forma:

- La red PMOS conecta su salida a VDD si su admitancia lógica es 1:

$$Yp\ (X1, X2, ..., Xm) = 1$$

- La red NMOS se conecta a GND si su admitancia lógica es 1:

$$Yn\ (X1^d, X2^d, ...Xm^d) = 1$$

Una condición indispensable en este tipo de redes es que sólo una de ellas debe conducir; en caso contrario, existiría un cortocircuito entre alimentación y tierra:

$$Yp * Yn = 0 \text{ o bien } Yp\ (X1, X2, ... Xm) * Yn\ (X1, X2, ..., Xm) = 0$$

2.1.3 Células funcionales y arrays celulares

Las células funcionales se consideran como implementaciones de unidades lógicas. Estas pueden implementarse como la evaluación de un conjunto de funciones lógicas; cada una de un nivel determinado, entendiendo por nivel el grado de profundidad que tiene en una representación en forma de árbol.

Las células CMOS estáticas consisten en dos subcircuitos, la red P y la red N. Las prestaciones de estas células pueden aumentar y el tamaño del mismo disminuir, si se usan células para implementar funciones lógicas de varios niveles, en lugar de utilizar un conjunto de ellas, interpretando cada una como célula funcional estándar.

Por ejemplo, sea la ecuación lógica en notación prefija:

$$Z = (+ (* a b (* (+ c d) (+ e f)))$$

Esta función puede ser implementada usando tres puertas OR y dos puertas AND de dos entradas cada una, o también utilizando una sola célula en cuya materialización podemos apreciar que el área de la misma es menor que en el caso de utilizar células diferentes para cada puerta.

El uso de este tipo de células funcionales en vez de sus equivalentes de células más primitivas (NAND, NOR, AND y OR) suele conllevar una reducción del área ocupada por los subcircuito.

Por ejemplo:

Sea la ecuación Z = -(* / + a b (+ / * c d e)) . Su representación con células independientes sería la siguiente:

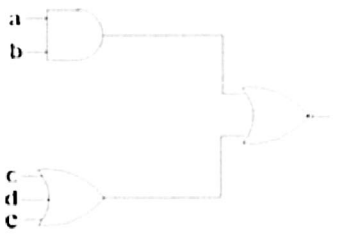


Figura 2-4 Implementación de la ecuación booleana Z con tres células

Si esta ecuación se implementa como una sola célula, su representación en red quedaría de la siguiente forma:

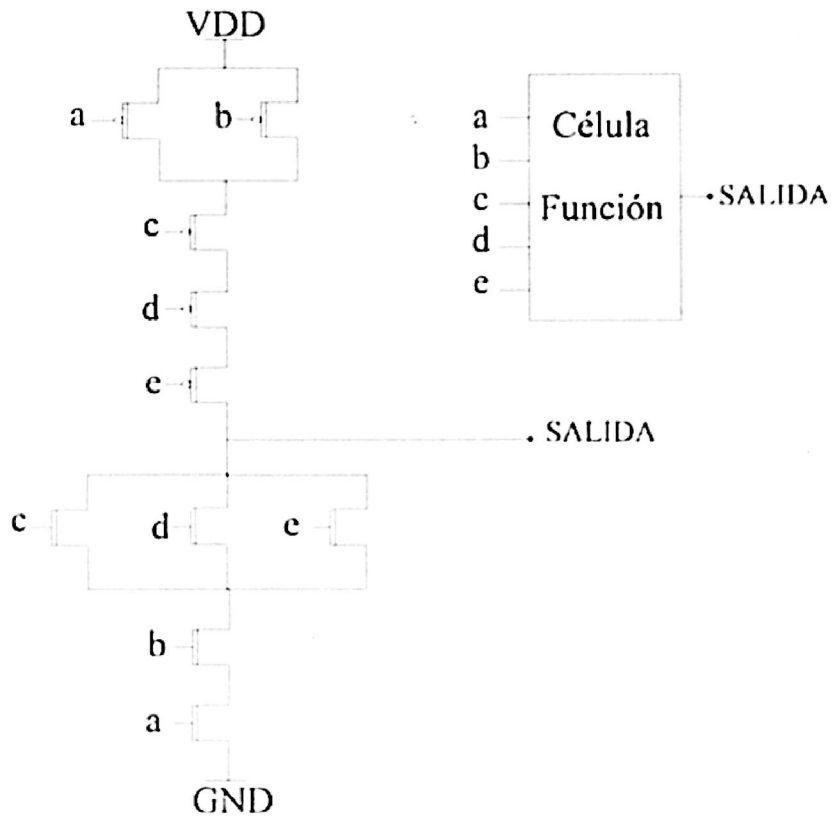


Figura 2-5 Implementación de la ecuación booleana Z con una única célula

Existen varios tipos de células que son de interés en nuestro estudio.

Células Serie-Paralelo / No Serie-Paralelo

Las células serie-paralelo implementan funciones lógicas en las que se alternan operadores de tipo serie y paralelo. Con tecnología CMOS toda operación serie en la red PMOS se traduce en una operación paralelo en la NMOS (Figura 2-6). Si en ambas redes las operaciones no son inversas las células se denominan No serie-paralelo.

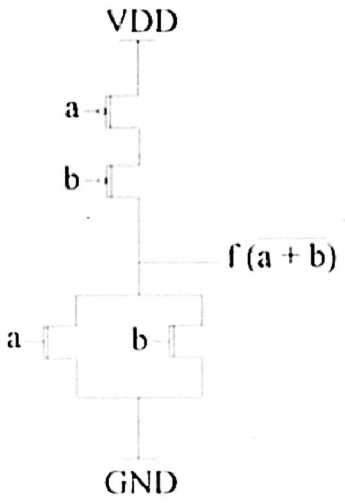


Figura 2-6 Implementación Neuronal de una Célula SP.

Células Unidimensionales / Bidimensionales

Una célula unidimensional es un array de transistores PMOS o NMOS en la que todos los terminales de drenadores y fuentes están dispuestas a lo largo de una misma fila de difusión. La representación de una célula básica unidimensional de un sólo transistor se puede ver en la figura 2-7.

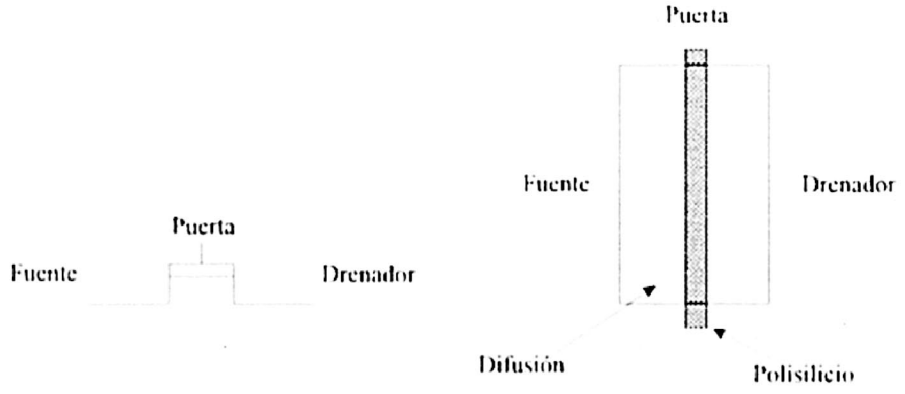


Figura 2-7 Célula básica unidimensional.

La columna de polisilicio se corresponde con la puerta del transistor y determina si circula intensidad entre fuente y drenador actuando como un simple interruptor. La difusión queda dividida gráficamente en dos partes, formando la fuente y el drenador.

Una célula unidimensional estática consiste en dos filas de transistores. La inferior corresponde al subcircuito o red NMOS y la superior a la red PMOS. Las puertas de los transistores que comparten una misma columna, o posición en ambas filas, suelen conectarse entre ellas con polisilicio (Figura 2-8).

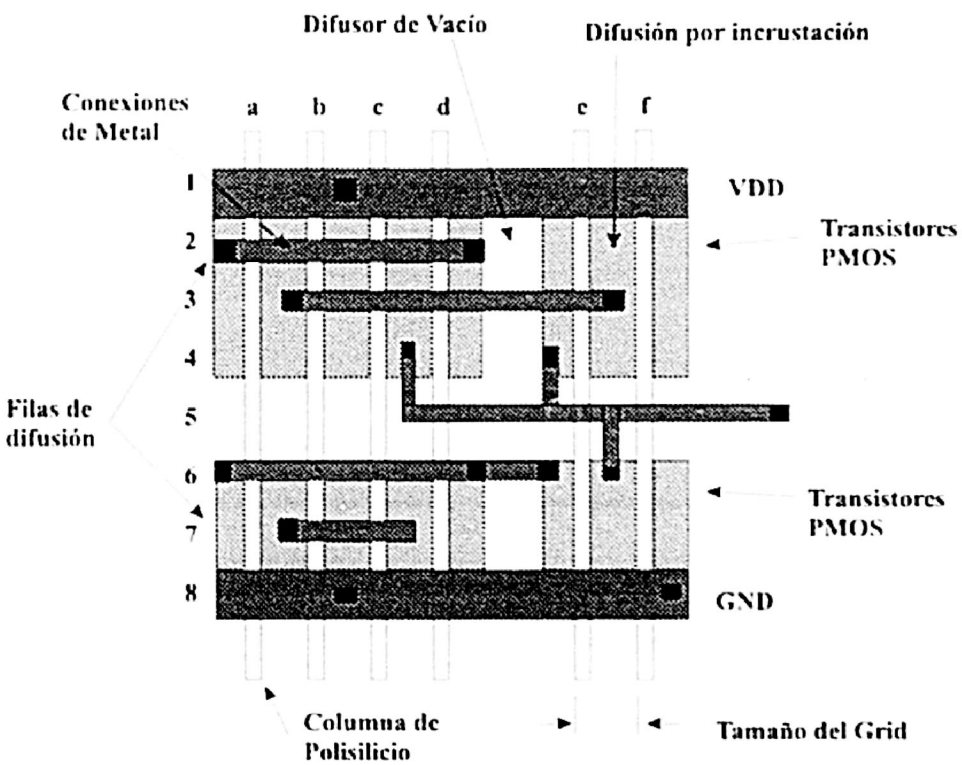


Figura 2-8 Célula unidimensional estática.

Bidimensionales

En las células bidimensionales, a diferencia de las unidimensionales, no existe línea difusora común para la red NMOS y para la PMOS. Así, la difusión está presente en todo el circuito de transistores, es decir, se introduce en un plano de modo que la difusión afecta a todo el área ocupada.

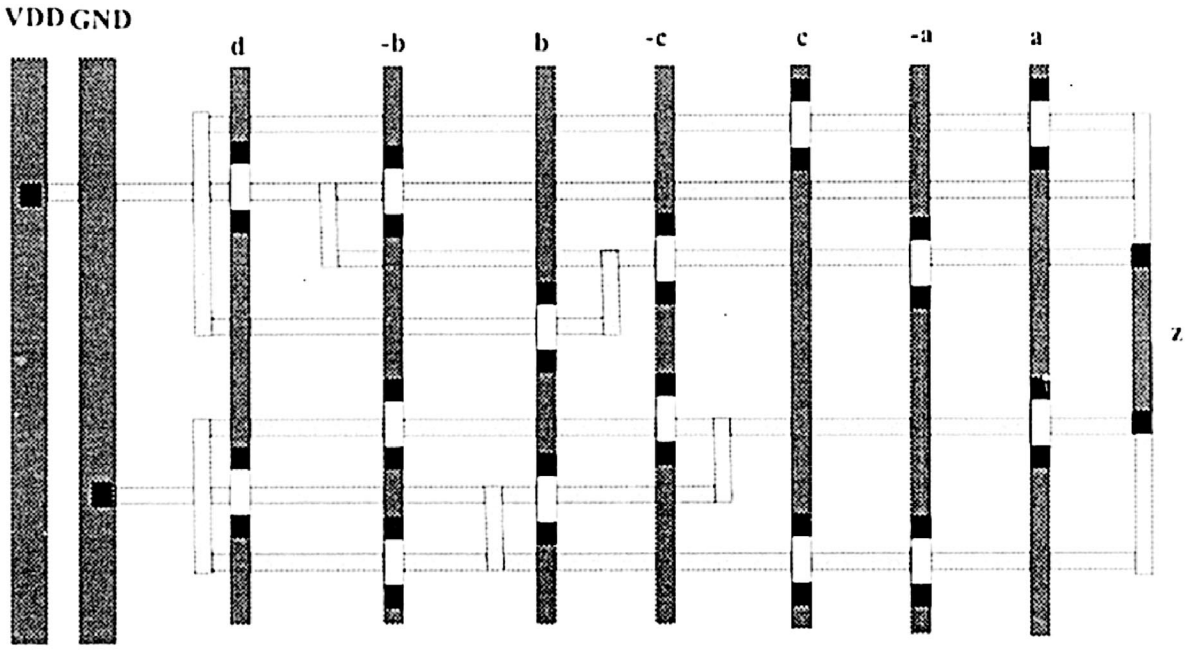


Figura 2-9 Implementación de una célula bidimensional.

En este tipo de células, las puertas son líneas metálicas que atraviesan la zona difusora sin crear transistores. Cuando se precise la creación de un transistor, se conectará la puerta al polisilicio.

Este tipo de implementación [WEIN67] utiliza un conjunto de células funcionales NMOS, atravesado tanto vertical como horizontalmente por líneas difusoras. Las filas polisilicias verticales son terminales puerta en la intersección de las dos capas anteriores.

Otra forma de implementar circuitos CMOS, es realizando una descomposición temporal en dos fases que son controladas por dos señales de sincronización: precarga y evaluación.

Las capacidades formadas por la unión de los distintos materiales que configuran un transistor, son cargadas en la fase de precarga y evaluadas (descargadas en función de los niveles lógicos de entrada) en la fase de evaluación. Esta técnica se conoce como Lógica Dinámica, y si el nivel de diseño es celular, el resultado será una célula dinámica. En estas células se utilizan menos transistores que en las descritas anteriormente. Esto supone una reducción importante de tamaño.

En las figuras 2-10 y 2-11 se muestra un ejemplo de la distribución de los transistores en una célula SP y en una célula dinámica CMOS, respectivamente, para la ecuación $Z = a * (b + c)$.

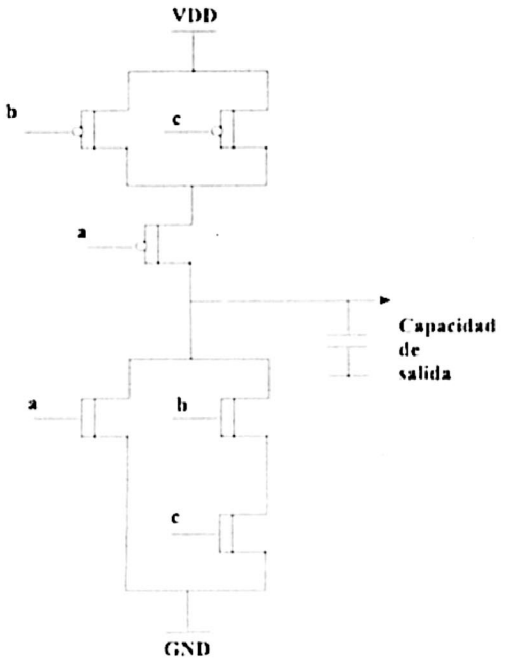


Figura 2-10 Diseño con células CMOS (SP).

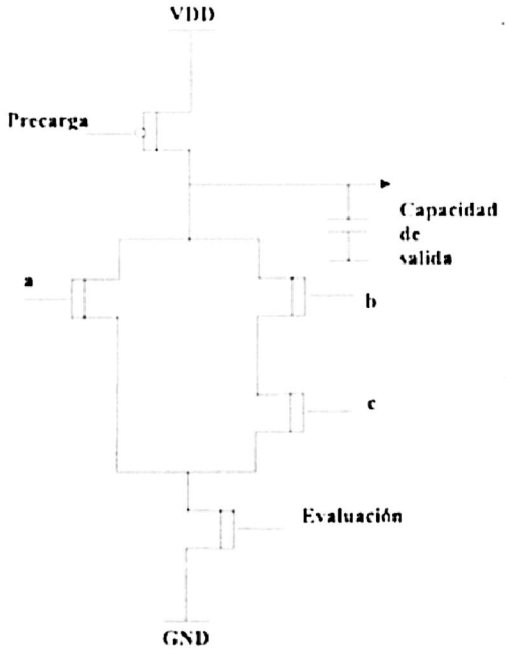


Figura 2-11 Diseño con célula dinámica CMOS

2.1.4 Modelización de células serie-paralelo

Las teorías expuestas por Uehara y VanCleemput [UEHA81] según las cuales las células serie-paralelo son un subconjunto de las estáticas, parten para su modelización de las siguientes premisas:

- Se utilizará la tecnología CMOS y dentro de ella las de tipo estático.
- Las redes PMOS y NMOS están constituidas como subcircuitos serie-paralelo de conexiones de transistores.
- Los subconjuntos PMOS y NMOS son geométricamente duales entre sí.
- El número de transistores en serie entre VDD y la salida está limitado a cuatro por dos tipos de limitaciones: limitaciones físico-eléctricas, y limitaciones técnico-procedurales. Al colocar varios transistores en serie se da una atenuación en cada transistor: limitación físico-eléctrica. Si se colocan cinco transistores en serie, el potencial de carga recibido por el quinto podría no superar su tensión de umbral. Modelizando cada uno de estos como una resistencia y un condensador, se obtiene el esquema de la figura 2-12.
- Los pares de transistores complementarios (PMOS y NMOS) están alineados. Esto permite que sus puertas estén interconectadas mediante una columna vertical de polisilicio sin usar conexiones.
- El drenador de un transistor y la fuente de otro, están conectados por difusión si los terminales son físicamente adyacentes en el layout. Si no fuesen adyacentes físicamente se conectarían por línea metálica.

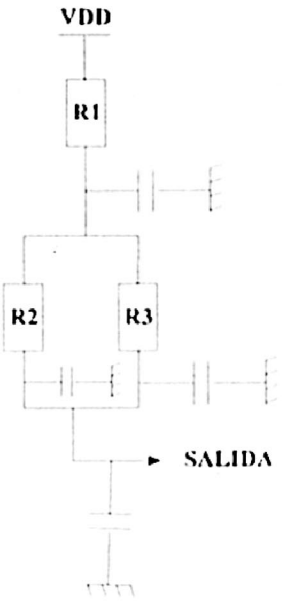


Figura 2-12 Modelización de una red CMOS con un circuito RC.

Partiendo de estas premisas Uehara y VanCleemput [UEHA81] desarrollarón algunos modelos de representación celular de cara a diferentes utilidades. Dichos modelos son los siguientes:

a) Modelo ecuacional.

Toda célula CMOS SP puede representarse funcionalmente por medio de una ecuación en notación post-fija que se puede formalizar de la siguiente manera:

[OP1 ... hijo +]

*hijo = hoja | NODOi

donde:

hoja = terminal o variable de la ecuación

NODOi = [OPi hijo*]

Por ejemplo:

Z1 = a * [b + c * (f + e)]

Z1 = - (*/+ a (+/* b (+/* c (+/* f e))))

La existencia de dos suboperadores está dada por dos redes o subcircuitos, y la dualidad entre ambos. Por otro lado, si la ecuación de partida es negada, el operador SO1/SO2 se corresponde con: SO1 para el circuito NMOS y SO2 para el circuito PMOS. Si la ecuación no es negada SO1 representa a PMOS y SO2 a NMOS.

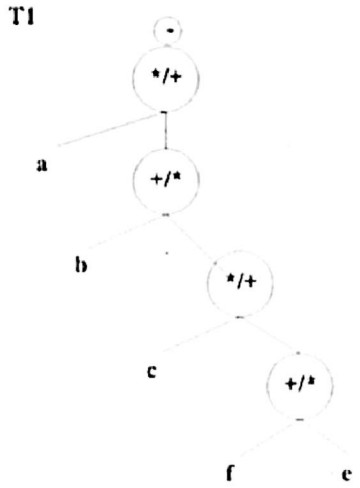
El operador OPi se formaliza de la manera siguiente:

- */+ operadores serie-paralelo (SP)
- +/* paralelo-serie (PS)

b) Modelo de árbol de composición.

Partiendo de la ecuación en notación post-fija, puede construirse una representación en forma de árbol, cuya ventaja estriba en su facilidad de proceso. Así, la representación de la ecuación dada sería la siguiente:

Z1 = - (*/+ a (+/* b (*/+ c (+/* f e))))



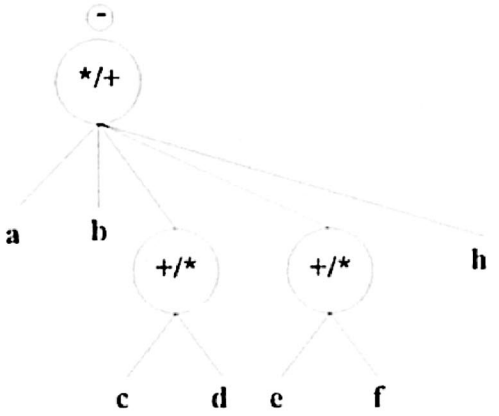
c) Modelo en multigrafos biterminales serie-paralelo: MBSP.

Consiste en una representación gráfica por medio de números de vértices y enlaces, que coincide con una de las posibles representaciones de las redes del circuito.

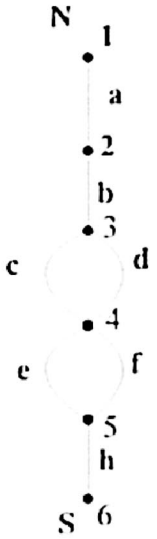
Su interpretación se lleva a cabo recorriendo el árbol de composición Ti en post-orden, y generando números de vértices en función de los operadores padre de un terminal.

Ejemplo 1: $\forall$ Z1 = - (*/+ a b (+/* c d) (+/* e f) h)

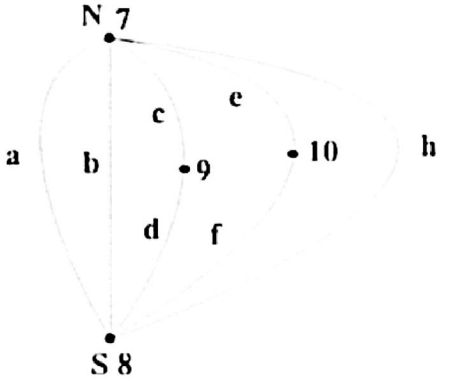
El grafo M surge del seguimiento del primer suboperador, y es la representación gráfica MBSP de la red NMOS si la ecuación va precedida de un '-', en caso contrario sería MBSP de la red PMOS.



Arbol de composición (T1)



Red P (grafo M)



Red N (grafo M^t)

De forma paralela, el grafo M^d surge del seguimiento del segundo suboperador. Representa un MBSP de la red NMOS si la ecuación va precedida de un '-' y sería el MBSP de red NMOS en caso contrario.

La orientación de los grafos se describe tipificando sus vértices. En cada grafo sólo existen dos terminales, uno de tipo N y otro de tipo S. N supone una conexión a

VDD si el grafo representa a la red PMOS y SALIDA si representa a NMOS. S es la conexión a SALIDA si el grafo representa a la red PMOS y GND si representa a NMOS.

Por otro lado, una ecuación puede tener varios árboles de composición Ti, y en definitiva varios MBSP (uno para cada árbol).

2.2 TERMINOLOGÍA

A continuación, pasamos a describir los términos, formatos y especificaciones que se utilizan en el desarrollo del presente trabajo de investigación.

2.2.1 Grafos MBSP

Un grafo MBSP y su dual los designaremos por M_i / M_i^d. Dichos grafos pueden tener diversos recubrimientos. Cr_i será el conjunto de dichos recubrimientos. Un recubrimiento $R_{i,j}$ de un grafo M_i / M_i^d es el conjunto de caminos (t_i) necesarios para recorrerlo por completo. Es decir, $R_{i,j} = \{t_1\#t_2\#...\#t_n\}$ donde '#' indica la posible separación entre dos caminos.

Un camino o recorrido del grafo será válido sólo si tiene correspondencia en el primal (M_i) y en el dual (M_i^d). Dicho camino lo representaremos como dos secuencias o conjuntos de vértices (V_i) y enlaces (e_i):

$$t_k = \{ V_1\, e_1\, V_2\, e_2\, V_3\, e_3 \ldots V_{n-1}\, e_n\, V_n \} \quad \{ V_1^d\, e_1^d\, V_2^d\, e_2^d\, V_3^d\, e_3^d \ldots V_{n-1}^d\, e_n^d\, V_n^d \}$$

donde V_i representa el vértice 'i' del grafo M_i y V_i^d el del grafo M_i^d formando un par vértice (V_i,V_i^d). Con el fin de facilitar el manejo y establecer una teoría para la compatibilidad de caminos, es necesario tipificar los vértices. Estos pueden ser de tipo N, S ó I según sea Norte, Sur o Interno, $T(V_i) \in \{N,S,I\}$ dependiendo de su ubicación.

El tipo de un par vértice final se denota por el par $(T(V_i),(V_i^d))$. Así, pueden darse los siguientes tipos de pares de vértice finales: {(N,N),(N,S),(S,N),(S,S),(I,I)} Los cuatro primeros son los tipos pares de vértices finales que tienen capacidad de concatenación y el último representa a todos los tipos de pares de vértices finales en los que alguno de sus vértices sea de tipo I.

El enlace entre los vértices (e_i para el grafo M_i) y (e_i^d para M_i^d) coincide con una de las variables de la ecuación y nunca aparece dos veces en un recubrimiento. El tipo de un camino t_k viene dado por el de los vértices inicial y final. Para expresar un tipo de camino usaremos la siguiente notación:

$$T(t_k) = (T(V_1), T(V_1^d)) / (T(V_k),T(V_k^d))$$

Los tipos de caminos más simples o de único enlace son (N,N)/(S,S) y (S,S)/(N,N) y con ellos se realizan operaciones */+ y +/* de forma repetitiva,

Los tipos de caminos más simples o de único enlace son (N,N)/(S,S) y (S,S)/(N,N) y con ellos se realizan operaciones */+ y +/* de forma repetitiva, consiguiendo así más tipos de caminos que varían en su descripción en función del grafo. Tomando como criterio el tipo de vértices finales de un camino, junto con el número de enlaces que recorre, puede establecerse la siguiente clasificación:

a) Camino Dual

Un camino dual posee dos subcaminos: t (subcamino para el grafo M) y t^d (subcamino para el grafo M^d).

Por ejemplo, sean los grafos M y M^d de la figura 2-13. Un posible camino dual en la notación de Uehara del grafo M/M^d es el siguiente:

$$t/td = \{(4, 7), (a, a), (3, 5), (b, b), (2, 8), (c, c), (3, 7)\}$$

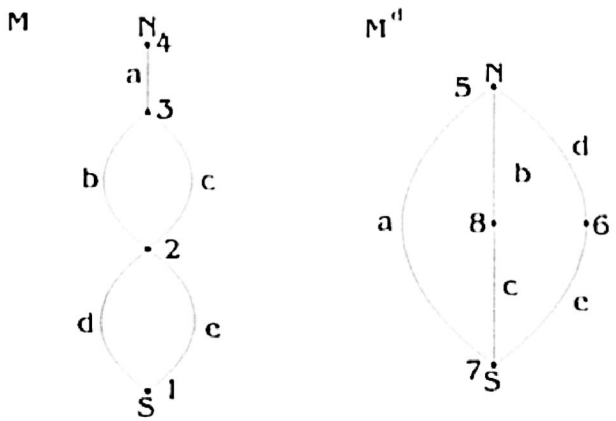


Figura 2-13 Grafo de MBSP

b) Camino Dual Distinguido

Un camino de este tipo es un camino dual que contiene al menos un par de vertices finales (N o S) del grafo M/M^d. Dado el grafo M/M^d de la figura 2-14, cuyos vértices finales son 1 y 3 para M, y 4 y 6 para M^d. Un posible camino es:

$$t = (1\ a\ 2\ b\ 3\ c\ 2)$$

$$td = (6\ a\ 4\ b\ 5\ c\ 6)$$

En él hay pares de vértices finales: (1, 6)/(2, 6). Por esta razón, el camino es un camino distinguido.

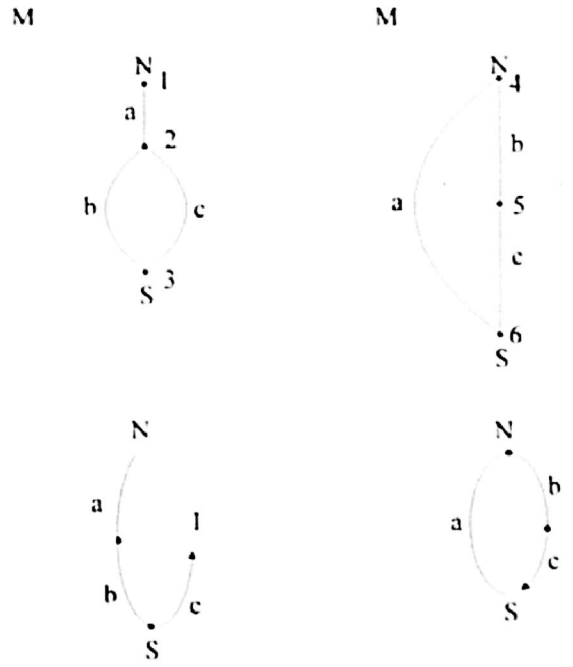


Figura 2-14 Ejemplo de un MBSP

c) Camino interno

Un camino interno es aquel que no tiene ningún par de vértices finales. Dado el grafo M/M^d de la figura 2-15, cuyos vértices finales son 1 y 3 para M, y 4 y 7 para M^d. Un posible camino es:

$$t = (2\ a\ 1\ b\ 2\ d\ 3\ c\ 2)$$

$$td = (4\ a\ 5\ b\ 7\ d\ 6\ c\ 4)$$

En él hay pares de vértices finales: (2, 4) y (2, 4). Ningún par es final, por lo que el camino es interno.

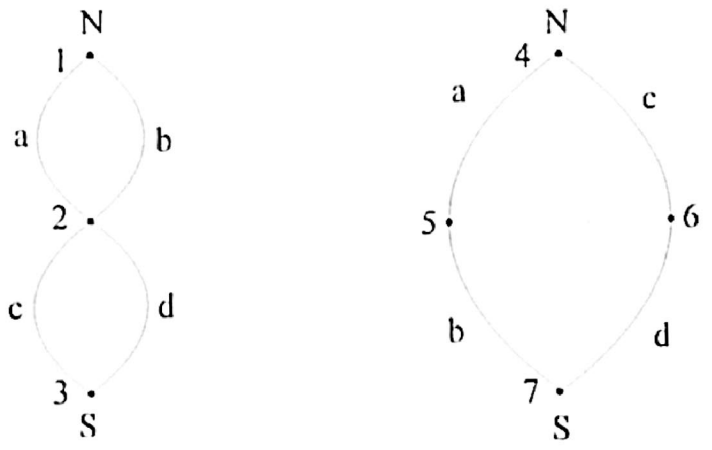


Figura 2-15 Un nuevo grafo MBSP para un ejemplo

d) Camino dual-Euler

El camino dual-Euler es el que contiene todos y cada uno de los enlaces de M y M^d una única vez. El camino descrito en la figura 2-13, correspondiente al grafo MSBP, es dual-Euler, ya que pasa por todos los enlaces una única vez.

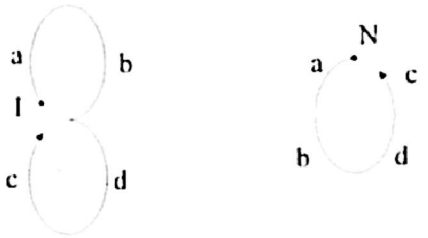


Figura 2-16 Recubrimiento posible del grafo MBSP

e) Caminos duales compatibles

Dos caminos t1 y t2 son concatenables o compatibles si poseen un vertice final en sus respectivos grafos (M1 y M2) que coincida en el grafo M3=M1#M2 (donde # puede ser * o +). En este caso t1 y t2 forman un único camino (t3) que recorre el grafo M3.

Si la compatibilidad se cumple para los caminos primales y duales, entonces se denominan Caminos Duales Compatibles. En la figura 2-17 representamos un ejemplo de caminos duales compatibles donde:

M3 = M1 * M2

t1 de M1 y t2 de M2, en notación 1:

t1 = {(2, 4), (a, a), (1, 5), (b, b), (2, 6)} = {2 a 1 b 2}

{4 a 5 b 6} en notación 2

t2 = {(2, 4), (c, c),(3, 4)} = {2 c 3}

{4 c 4} en notación 2

t3 = {(2, 4), (a, a), (1, 5), (b, b), (2, 6), (c, c), (3, 4)} = {2 a 1 b 2 c 3}

{4 a 5 b 6 c 4}

t3 = t2 */+ t1 = t2 */+ t1

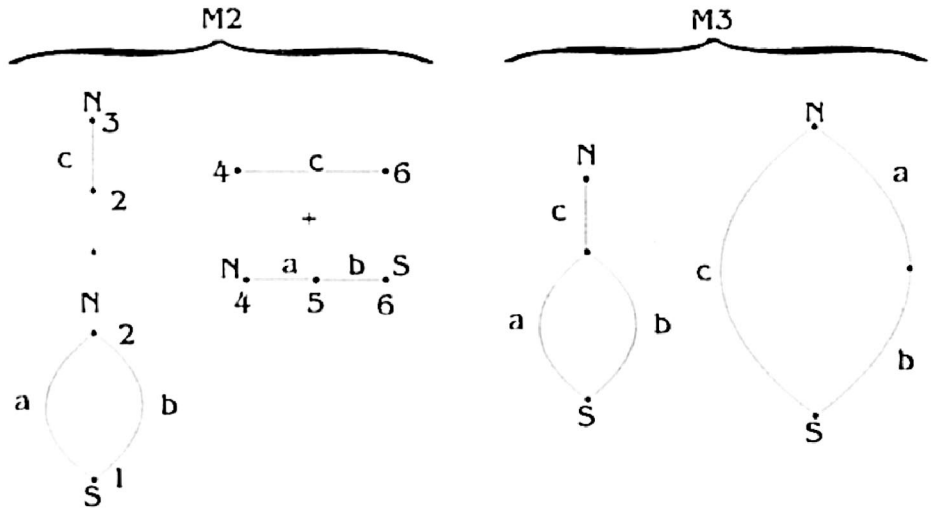


*Figura 2-17 Ejemplo de composición de un grafo M/Md con dos subgrafos más pequeños bajo el operador */+.*

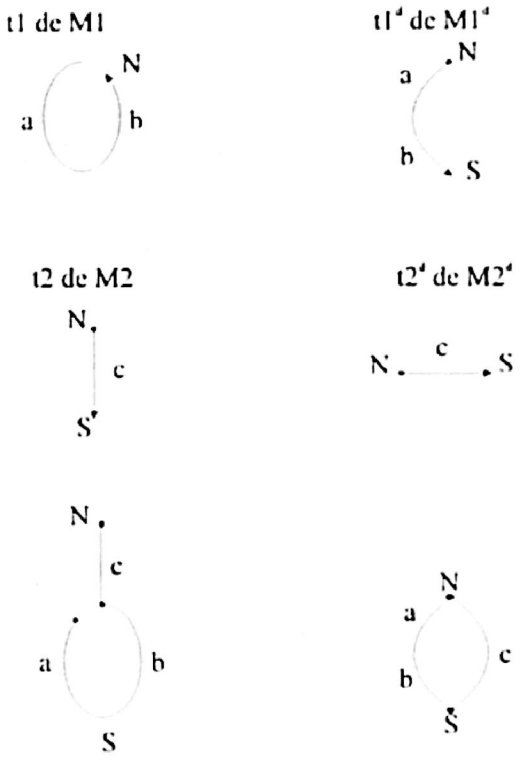


Figura 2-18 Ejemplo de recubrimiento de un MBSP con un camino formado por dos compatibles.

Sólo pueden concatenarse los tipos de pares vértices finales (N, X) que sean operandos izquierdos del operador */+, con los operandos derechos de tipo (S, X), donde X es N o S. El tipo del camino resultante contiene dos tipos de pares de vértices finales.

Continuando con el grafo del ejemplo de la figura 2-17 la concatenación de los caminos t1 y t2 sería la siguiente:

$$T(t3) = T(t1) \,*/+\, T(t2) = (N, N)/(S, S) \,*/+\, (N, N)/(S, S) = (I, N)/(S, N)$$

En la figura 2-19 se presenta un ejemplo de Caminos Duales Incompatibles:

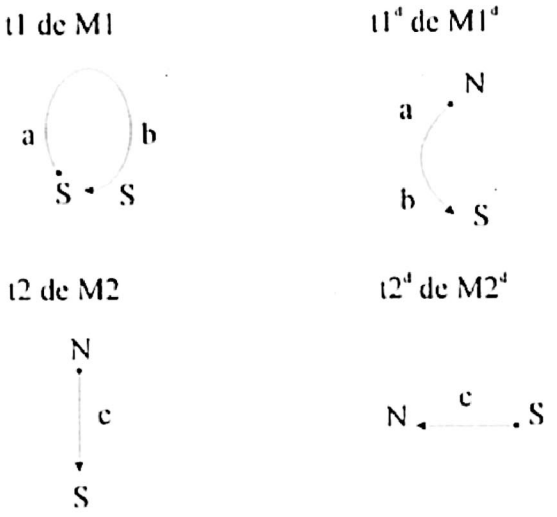


Figura 2-19 Caminos duales incompatibles.

Estos caminos no son compatibles, por lo que el recubrimiento para M3 es mediante 2 caminos incompatibles.

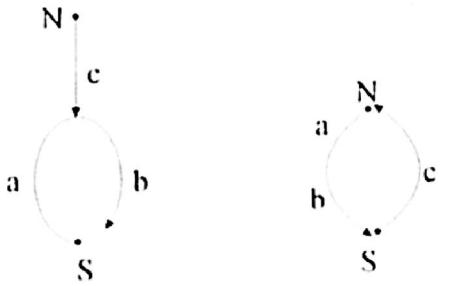


Figura 2-20 Ejemplo de recubrimiento formado por dos caminos incompatibles.

t1' = {(1, 4), (a, a), (2, 5), (b, b), (1, 6)}

t2' = {(2, 6), (c, c) (3, 4)}

t2' */+ t1' es una operación de caminos incompatibles.

El camino resultante para cubrir a M3 será un conjunto de caminos duales incompatibles y por tanto separados:

(t1'; t2') = {(1, 4), (a, a), (2, 5) (b, b), (1, 6); (2, 6), (c, c), (3, 4)}

Si la concatenación no es posible entonces se representa con I a todos los tipos de caminos que forman el recubrimiento. Todos los caminos que recubren el grafo M, que contengan exactamente el mismo número de pares de terminales, pueden ser considerados equivalentes. Con el fin de minimizar el conjunto de caminos que cubren M sin eliminar ninguno que lleve a la solución óptima, agruparemos el conjunto de caminos de recubrimientos en clases equivalentes. De este modo, se elige el camino de recubrimiento con el menor número de caminos equivalentes de cada clase equivalente.

Agrupados los caminos por la clase y por el recubrimiento al que pertenecen, estos últimos a su vez forman conjuntos de recubrimientos que recorren íntegramente un grafo MBSP. Los conjuntos de recubrimientos se clasifican según el número de caminos internos que posea cada recubrimiento de conjunto y en función de los tipos de los caminos integrantes de un recubrimiento (tipo de recubrimiento) de la siguiente forma:

a) Recubrimiento CR o completo.

Un recubrimiento CR es el conjunto de recubrimientos de M que contienen un representante de cada uno de los tipos que pueden recorrer M. Para denotarlo se utiliza el formato siguiente:

CR = {R1, R2, R3,...} donde cada Ri = {dt1, dt2, dt3, ..., dtj, ..., it1, it2, ...}

En un conjunto de recubrimientos CR, cada recubrimiento Ri posee un tipo formado por el tipo de todos los caminos que son incompatibles. Un CR está formado por todos los recubrimientos. aunque dos de ellos repitan su tipo.

b) Recubrimiento CRM o recubrimiento completo mínimo

Un recubrimiento CRM está formado por los recubrimientos de M, que contienen un representante de cada uno de los tipos de caminos que pueden recorrer M, que poseen el mínimo número de caminos internos. Su notación es la siguiente:

CRM = {tc1, tc2, tc3,...} donde cada

tci = {dt1, dt2, dt3, ..., dtj, it1, it2, ..., itn}, y n es el número mínimo de caminos internos.

CRM es un conjunto de recubrimientos en donde puede repetirse un tipo en dos recubrimientos, pero cada recubrimiento debe poseer un número mínimo de caminos internos. Este tipo de recubrimientos proporciona una mayor posibilidad de conexión

mayores para posteriores concatenaciones de recubrimientos, ya que en él están eliminados el máximo número de caminos internos, de cada uno de sus recubrimientos.

Por ejemplo:

Partiendo del grafo MBSP de la figura 2-21, que representa una célula SP, y de los tres posibles recubrimientos del grafo:

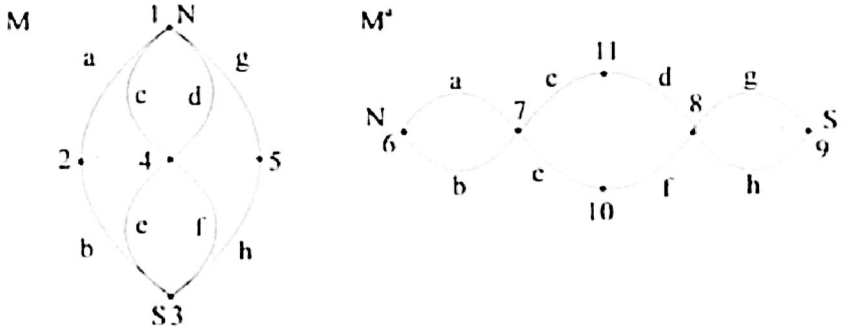


Figura 2-21 Recubrientos posibles del grafo.

$$R1 = \left\{ \begin{matrix} 1\,a\,2\,b\,3\,\#\,4\,c\,1\,d\,4\,f\,3\,e\,4\,\#\,1\,g\,5\,h\,3 \\ 6\,a\,7\,b\,6\,\#\,7\,c\,10\,d\,8\,f\,11\,e\,7\,\#\,8\,g\,9\,h\,8 \end{matrix} \right\}$$

$$R2 = \left\{ \begin{matrix} 1\,a\,2\,b\,3\,\#\,1\,c\,4\,d\,1\,\#\,3\,e\,4\,f\,3\,\#\,1\,g\,5\,h\,3 \\ 6\,a\,7\,b\,6\,\#\,7\,e\,10\,d\,8\,\#\,7\,e\,11\,f\,8\,\#\,8\,g\,9\,h\,8 \end{matrix} \right\}$$

$$R3 = \left\{ \begin{matrix} 1\,a\,2\,b\,3\,e\,4\,f\,3\,\#\,1\,c\,4\,d\,1\,g\,5\,h\,3 \\ 7\,a\,6\,b\,3\,e\,11\,f\,8\,\#\,7\,c\,10\,d\,8\,g\,9\,h\,8 \end{matrix} \right\}$$

Los tipos de los recubrimientos respectivos son los siguientes:

T(R1) = {(N,N)/(S,N) # (I,I)/(I,I) # (N,N)/(S,N)}

T(R2) = {(N,N)/(S,N) # (N,I)/(N,I) # (S,I)/(S,I) # (N,I)/(N,I)}

T(R3) = {(N,I)/(S,I) # (N,I)/(S,I)}

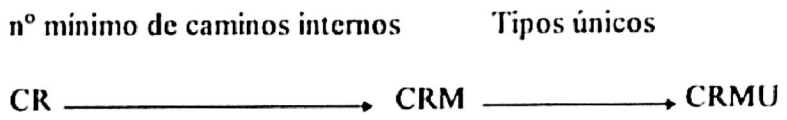
El CR correspondiente incluirá todos los recubrimientos, pues todos son distintos. Sin embargo, CRM sólo incluye R1, puesto que su recubrimiento únicamente proporciona un sólo camino interno. Este permite, en posteriores formaciones de grafos más grandes, una mayor concatenabilidad, lo que es fundamental para encontrar el recubrimiento óptimo. Así, si el grafo tomado es total, es decir, representa íntegramente una célula, el recubrimiento óptimo es R3, pues precisa de menos difusores de vacío.

c) Recubrimiento CRMU o recubrimiento completo mínimo unario.

Este es un CRM que contiene un único recubrimiento de cada tipo de camino posible. Es decir,

CRMU = {tc1, tc2, ...} donde tci = {dti}, o bien tci = {dtj}

Por tanto, en un CRMU no pueden existir dos recubrimientos de igual tipo y la relación entre los diferentes tipos de recubrimiento puede expresarse de la siguiente forma:



Cada tipo de recubrimiento es de especial utilidad en una parte del análisis arbóreo de la ecuación a implementar. Los CR se usan a nivel de hojas y los CRM a nivel de nodos.

En el caso de implementar una "célula a medida" que no va a ser gestionada por el GAO se usan los CRMU a nivel de raíz. En caso contrario, se define un nuevo tipo de recubrimiento llamado recubrimiento CRMC, que es el conjunto de recubrimientos mínimos completos (CRM), que incluyen propiedades como orientación de caminos de un recubrimiento y ordenación en los caminos de un recubrimiento. En consecuencia, el número de recubrimientos resultantes en un CRMC es elevado y todos ellos son empleados por el GAO.

2.2.2 Implementación de layouts

El objetivo final del diseño de un array, y por consiguiente de una célula, es la realización del layout; el cual estará determinado por la ubicación de los transistores necesarios y sus interconexiones.

La metodología de construcción del layout a partir de los grafos MBSP (M_i/M_i^d) pertenecientes a una célula es la siguiente:

Se seleccionan un conjunto de recorridos o recubrimientos comunes a ambos grafos. Esta selección será determinante en la optimización del layout puesto que permitirá definir el recorrido ($R_{i,j}$) para cada uno de los grafos (primal y dual) en base a sus vértices y enlaces. Si el recorrido elegido esté formado por varios caminos, éstos se separan con el símbolo '#'. Finalmente la representación gráfica del layout se realiza ubicando horizontalmente tantas zonas de difusión P como caminos formen el recorrido $R_{i,j}$ del grafo M_i y tantas zonas de difusión N como caminos en el grafo M_i^d, resultando una imagen simétrica sobre un eje horizontal separador de los dos tipos de difusión. Los transistores definidos en los caminos del recubrimiento $R_{i,j}$ como un enlace entre dos vértices (V_i e V_{i+1}) se implementan en el mismo orden sobre las difusiones de silicio aplicando una linea de polisilicio. A cada línea de polisilicio se le asigna el nombre del enlace (transistor) que representa. Todos los vértices con el mismo valor numérico, así como los correspondientes a las salidas de ambos tipos de difusión, se unen mediante conexiones metálicas.

El orden de colocación de los transistores en las dos filas horizontales de la implementación de la célula funcional afecta al ancho de la misma, ya que determina cuales de los transistores fisicamente adyacentes van a conectarse por incrustación. De esta manera, todos los enlaces del grafo M (despreciando por un instante su dual M^d) pueden recorrerse con un camino cuyo orden es el mismo que el de la implementación de las puertas en el layout. Cambiar el orden supone reordenar el árbol y el circuito de composición. Esto conlleva la formación de un nuevo árbol con la misma raíz pero con algunos nodos hijo permutados. Estos nodos pueden ser tanto hojas como nodos internos.

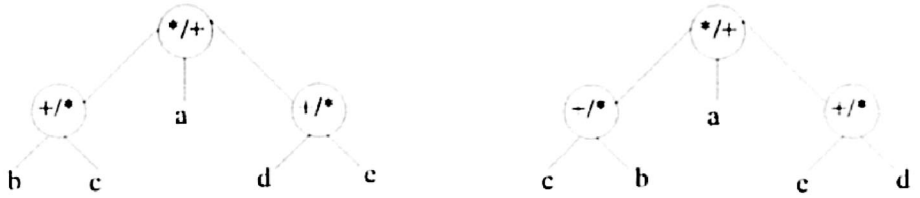


Figura 2-22 Reordenación de un árbol T que representa una ecuación.

Si transistores físicamente adyacentes no pueden estar unidos por incrustación, la separación de dichos transistores se lleva a cabo por un difusor de vacío. Es importante destacar que la anchura de un difusor de vacío es doble que la de un transistor. En un recubrimiento de un grafo pueden existir varios caminos duales, este hecho implica que en el layout, los transistores adyacentes de dos caminos duales se separen por un difusor de vacío. De esta manera, puede afirmarse que el problema de la minimización de la anchura celular se reduce a la búsqueda del mínimo número de caminos duales que cubran los multigrafos que representan las redes NMOS y PMOS. El número de caminos necesarios para cubrir un circuito de transistores dado depende de la ordenación y orientación de los subgrafos S/P en el modelo multigrafo. Una ordenación supone un nuevo conjunto de caminos que cubren a M_i/M_i^d. Una orientación también supone un nuevo conjunto de caminos que recubre a M_i/M_i^d, donde M_i/M_i^d será la reordenación M_i/M_i^d del grafo M/M^d de la ecuación original.

Observando el layout a nivel de array lineal el aislamiento entre células adyacentes se realiza bien mediante un difusor de vacío (Figura 2-23a), bien mediante un difusor por incrustación (Figura 2-23b). Lógicamente el segundo método es más óptimo

desde el punto de vista de la anchura total del array (recuérdese que la anchura de un difusor de vacío es el doble que la de un transistor).

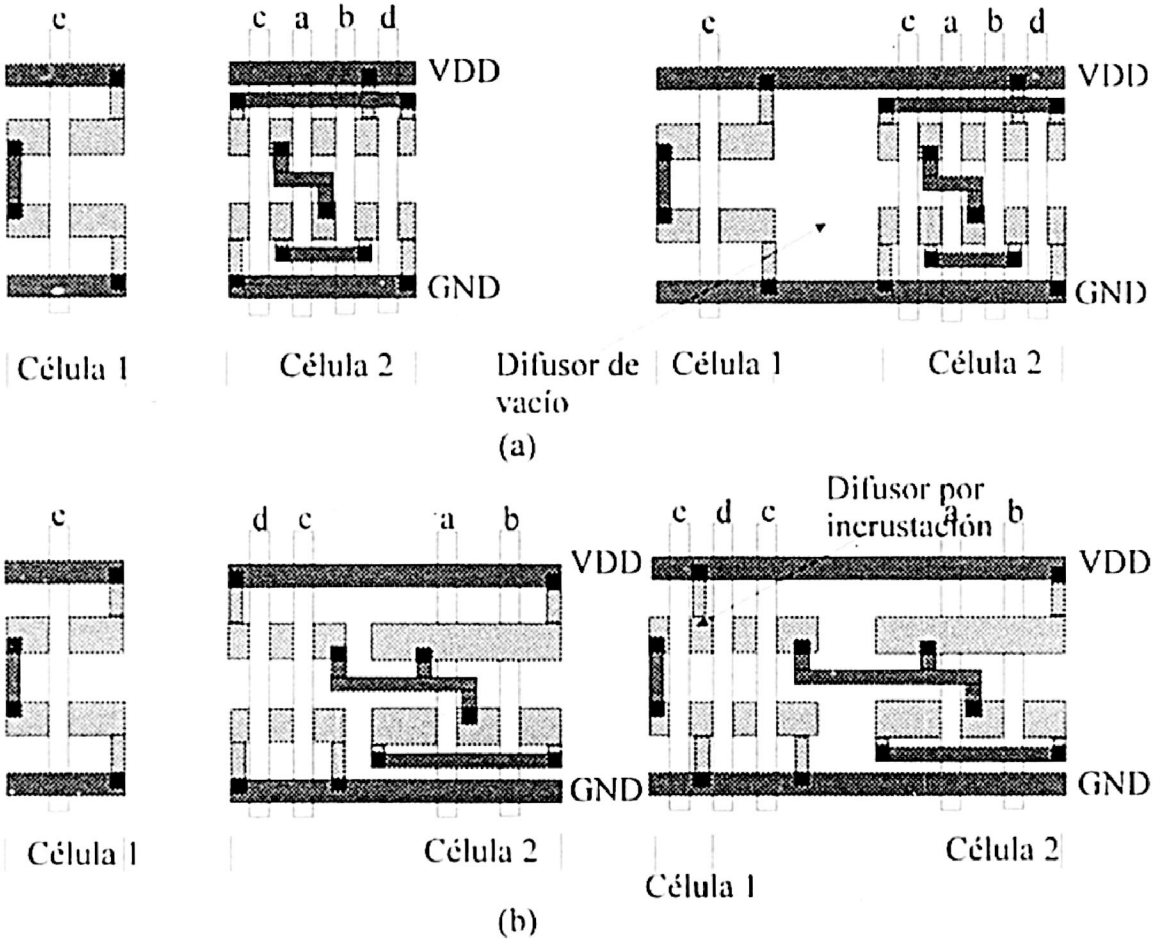


Figura 2-23 Unión de dos células adyacentes en un array lineal.

Para lograr la unión de células con difusión por incrustación es obligatorio que las alimentaciones de ambas redes de cada célula coincidan con los extremos de la misma. Para obtener un camino total que genere de partida el array más óptimo en

anchura, es necesario comprobar si en dos células adyacentes existe una coincidencia de alimentaciones y en caso afirmativo, unirlas directamente. La unión de células por difusión de incrustación proporcionará adicionalmente una reducción de la longitud de metal de los enlaces entre células participantes de la unión.

2.2.3 Cálculo de tiempos

Las características de los materiales citados determinan la velocidad de propagación de los transistores y por consiguiente de las células.

El tiempo de propagación es el que transcurre desde que se efectúa un cambio de las señales de entrada en una célula hasta que se recoge su evaluación en la salida de la misma.

Uno de los objetivos de la tésis actual es optimizar el tiempo de propagación celular, por lo que se necesita evaluar temporalmente cada solución celular con el fin de calcular el tiempo de propagación de cada una de ellas, seleccionando posteriormente aquellos que se correspondan con un tiempo menor.

Es necesario modelizar cada implementación como un circuito RC que permite calcular el tiempo de propagación celular mediante la expresión de la tensión de respuesta de un circuito RC equivalente:

$$V_{SALIDA}(t) = VDD\ e^{-t/RC}$$

Donde:

VDD es la tensión de alimentación

RC es la constante temporal asociado al circuito equivalente modelizador de la implementación celular.

De esta manera, aplicando esta expresión:

En to la V_{SALIDA} es 0.35 VDD

En tf la V_{SALIDA} es 4.65 VDD

La diferencia entre ambos tiempos es el tiempo de propagación de la red P.

$$t_{propagación\_red\_P} = tf - to = \frac{1}{RC} Ln \frac{4.65}{0.35} = \frac{2.587}{RC}$$

Puede calcularse igualmente el tiempo de propagación de la red N, considerando cargado (4.65VDD) inicialmente (to) el condensador ficticio de salida y descargándose (0.35VDD) a través de la red N en un tiempo tf.

3. OPTIMIZACIÓN TOTAL DE CÉLULAS SERIE-PARALELO

3.1 INTRODUCCIÓN

La optimización de los arrays celulares pasa por la optimización de las células que los componen, siendo ésta fase básica y de gran complejidad.

En esta capítulo describimos un método original y su implementación, el algoritmo A35, para la optimización celular.

3.2 PREMISAS DE PARTIDA

El presente trabajo de investigación toma como punto de partida las premisas establecidas por Uehara y VanCleemput [UEHA81] y otras originales para modelar la implementación celular. Estas últimas premisas se han añadido porque nuestra investigación se limita al diseño semicustom.

- **Premisa 1:** No se hacen distinciones entre tamaños de transistores. Por tanto, la anchura de los transistores será idéntica en ambas redes.

- **Premisa 2:** Debido a la atenuación que sufre una señal al pasar por varios elementos en serie, se limita el uso de circuitos CMOS duales serie/paralelo a una altura menor o igual a 4.

- **Premisa 3:** Las redes NMOS y PMOS se asignan cada una a una fila independiente en la implementación final, así los pares de transistores duales compartirán una misma columna polisilícea que cumplirá la función de sus respectivos terminales tipo puerta.

- **Premisa 4:** Los terminales fuente y drenador que deban conectarse en el circuito, lo harán mediante difusión por incrustación, en el caso de que se encuentren colocados adyacentemente dentro de una célula, y mediante hilos metálicos horizontales en el caso contrario. Los terminales dispuestos adyacentemente en una célula, pero que en el circuito no deban estar conectados, se separarán usando difusión por vacío.

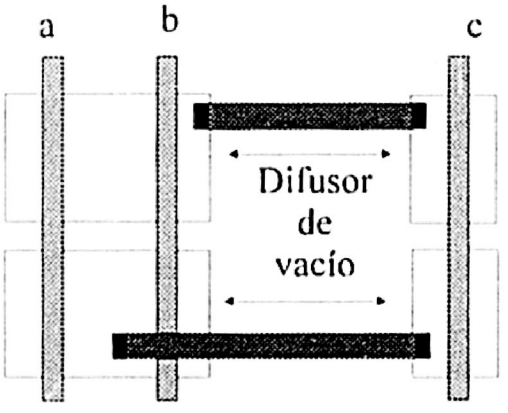


Figura -3-1 Separación física de transistores funcionalmente adyacentes.

Observando la figura 3-1, si el terminal 'c' debe ser adyacente al 'b', pero por requerimientos fisicos no es posible, entonces se utilizará un difusor de vacio.

3.3 MINIMIZACIÓN DE LA VELOCIDAD Y LA ANCHURA

Buscar una célula óptima total en velocidad y anchura supone encontrar la/s implementación/es más rápida/s y menos ancha/s. Es decir, hay que resolver el problema de la minimización de la velocidad operacional y posteriormente, se localizan las soluciones de menor anchura, y las de conexiones metálicas más cortas.

3.3.1 Velocidad operacional

La minimización de la velocidad operacional, más que un problema, es un critero de organización de los componentes de una célula, para lograr una mayor rapidez de la evaluación que implementa. Es decir, en el modelo de árbol de composición para representar ecuaciones habrá que evaluar lo más rapidamente posible los nodos más profundos (nodos urgentes).

Dada una ecuación lógica:

$$Z = (OP_1 (\ldots (OP_j e_j , 1e_j , 2\ldots) (OP_{j+1} e_{j+1}, 2e_{j+2}, 2\ldots e_{j+2}, m) \ldots))$$

y el árbol que le representa, si sus reordenaciones arbóreas poseen nodos profundos ubicados en las caras externas, entonces los transistores que implementan dichos nodos se sitúan junto a VDD/GND o SALIDA en los grafos M_i/M_i^d correspondientes a cada una de las reordenaciones.

Dado que cada transistor posee una resistencia y capacidad, la constante RC acumulada hasta la entrada del circuito que implementa un nodo profundo, es menor que en puntos del circuito global físicamente posteriores, por lo que el tiempo de carga de un hipotético condensador situado a la salida correspondiente a dicho nodo será mucho más elevado que en su salida.

En la figura 3-2 se muestra una red de transistores y la curva de respuesta tomada en los puntos A y B de dicha red.

La ventaja derivada es la pronta evaluación de los nodos más profundos en el arbol T, resultando una velocidad de caracter operacional mayor. Según el número de nodos profundos existentes en la representación arbórea de una ecuación, se distingen entre ecuaciones con uno, con dos o con más nodos profundos

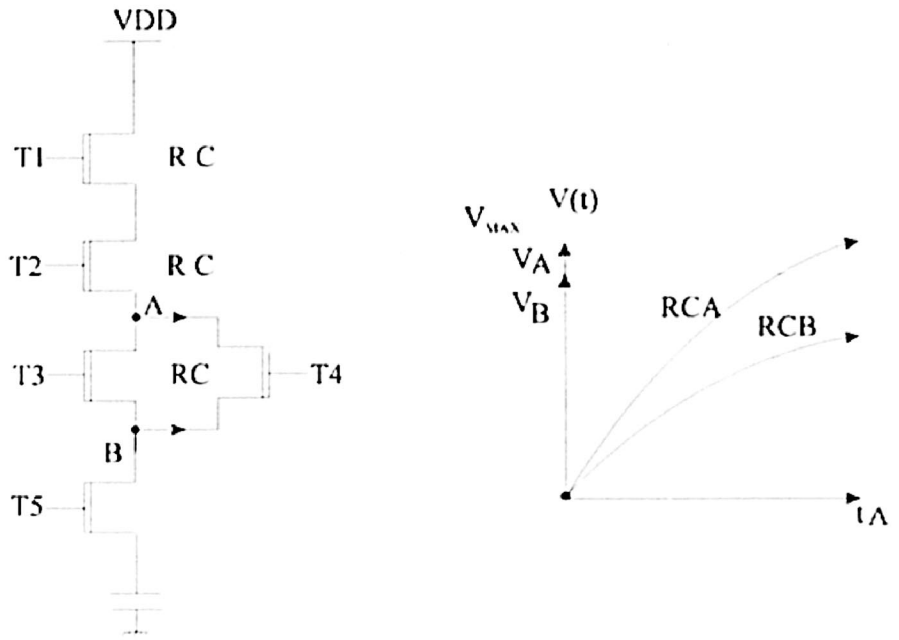


Figura 3-2 Modelización RC de una red complementaria de transistores y curva de respuesta tomada en diferentes puntos de la red.

- **Ecuación con un nodo profundo**

  Dada la siguiente ecuación:

  Z = (OP1 NX NX (..(...(OP$_j$ e_j,1 e_j,2 ... e_j,n)...)...)...

  NX

)

El nodo (OP$_j$ e_j,1 e_j,2 ... e_j,n) es el más profundo. Un árbol correspondiente a dicha ecuación es el de la figura siguiente:

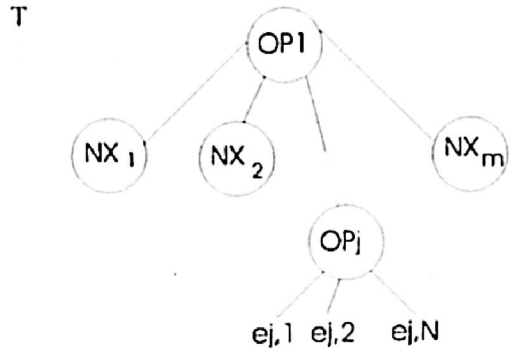


Figura 3-3 Árbol inicial de una ecuación dada (un nodo profundo)

Para obtener los grafos MBSP (M_i/M_i^d) de la ecuación, se recorre el árbol en post-orden de izquierda a derecha (Figura 3-3). En este caso, el nodo OP_j no es adyacente a VDD/GND ni SALIDA en ninguno de los grafos; por lo tanto, la reordenación i no es de interés en cuanto a velocidad operacional.

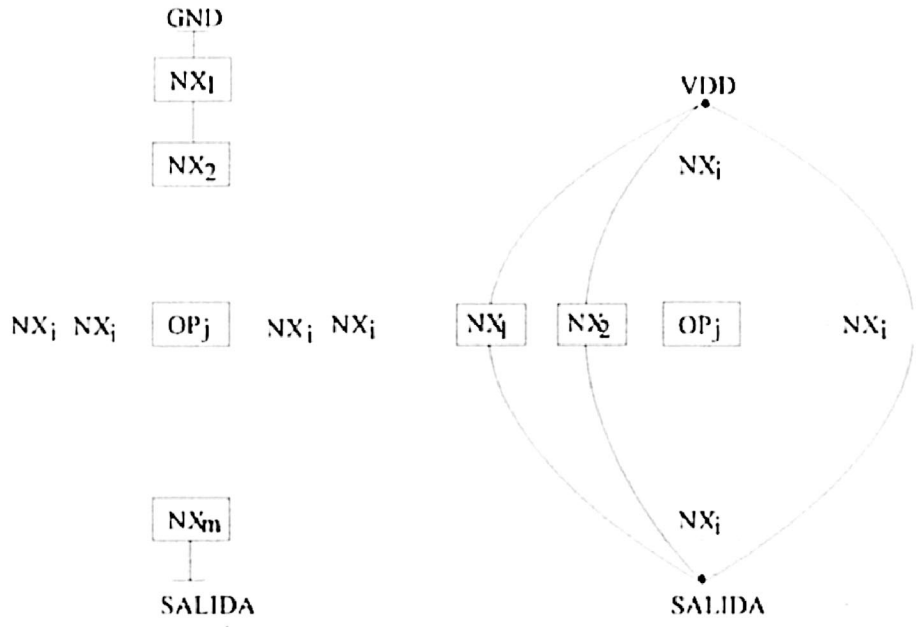


Figura 3-4 Ubicación del subgrafo correspondiente al nodo OP_j en el grafo MBSP.

Por el contrario, si se elige cualquier reordenacion de las que situan el nodo OP_j en cualquiera de las dos caras externas del árbol, entonces se logrará la ubicación gráfica y física del nodo Op_j cerca de VDD / GND o SALIDA. En este caso la reordenación o permutación considerada es de interés. Una permutación de interés (PI) correspondiente al árbol inicial es la mostrada en la figura 3-5.

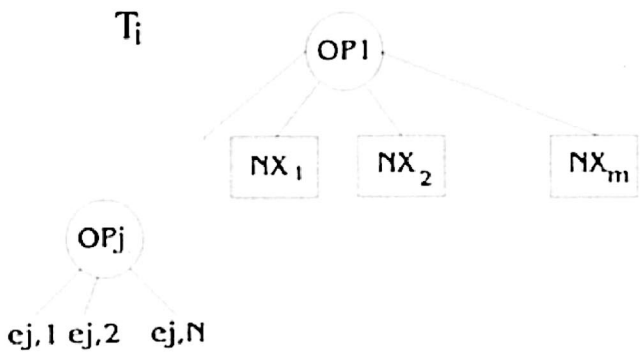


Figura 3-5 Permutación de interés para ecuaciones con un nodo profundo

- **Ecuación con dos nodos profundos**

Dada la ecuación con dos nodos profundos OPj y OPj+1

Z= (OP1 ... NXi...(NXi...(OPj ej,1 ej,2 ... ej,n)

(OPj+1 ej+1,1 ej+2,1... ej+1,n)...

) NXi)...

...NXi

)

Del mismo modo que para las ecuaciones con un nodo profundo, cualquier permutación del árbol inicial que sitúe los dos nodos profundos OPj y OP j+1 en sendas

caras externas del árbol, supone una permutación de interés. Fijando los nodos profundos en caras externas y permutando hojas de estos nodos profundos se obtiene un CPI, cada permutación de estas hojas es una PI (Figura 3-6).

Recorrer el árbol en post-orden de izquierda a derecha implica situar OP_j, OP_{j+1} adyacentemente a VDD / GND o SALIDA en el grafo MBSP.

- **Ecuaciones de tres o más nodos profundos**

Si la ecuación tiene tres o más nodos profundos entónces todas las permutaciones son de interés y no hay posibilidad de optimizar la velocidad operacional de dicha célula, dado que un árbol sólo tiene dos caras externas.

3.3.2 Anchura celular

A partir de una solución óptima en velocidad operacional (V_i), se consiguen las células óptimas en anchura.

La anchura de una célula viene dada por la fórmula siguiente:

$$W = T + G + I$$

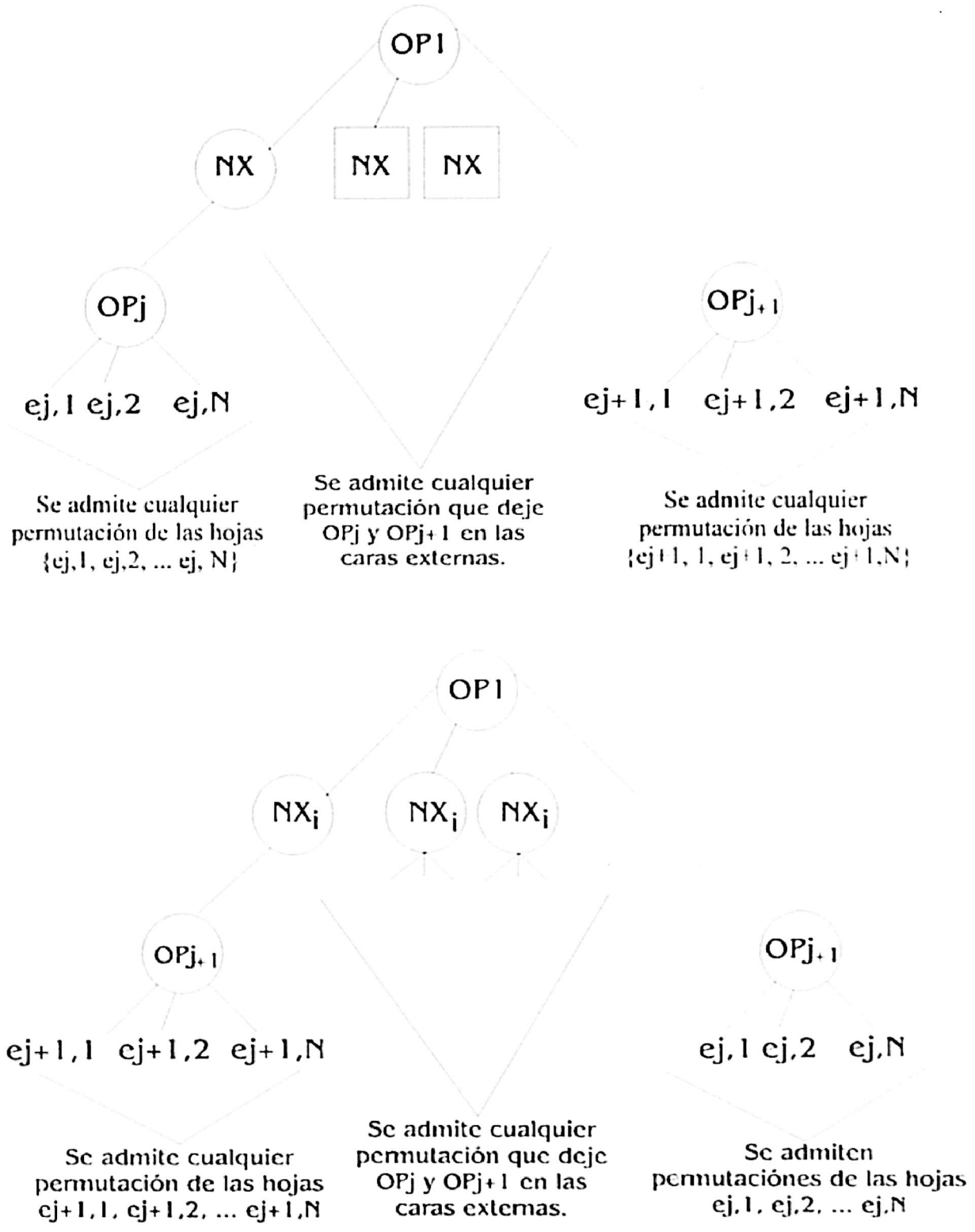


Figura 3-6 Permutación de interés para ecuaciones con dos nodos profundos

donde:

T: Número de terminales de la célula.

G: Número de difusores de vacío en la célula.

Por tanto, para minimizar la anchura de una célula habrá que reducir los difusores de vacío.

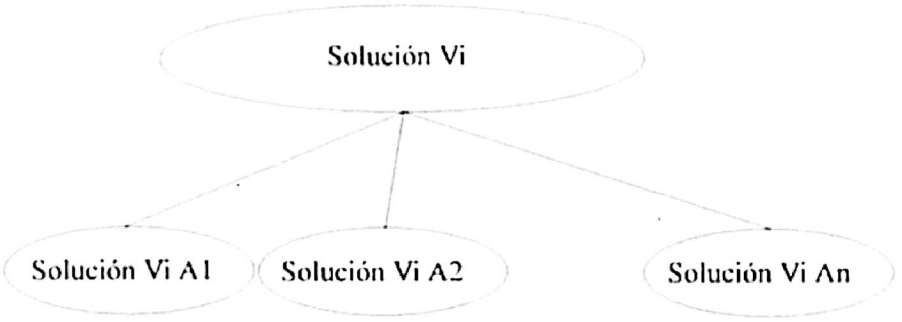


Figura 3-7 Soluciones de anchura de una solución Vi de velocidad operacional

El algoritmo A35 halla, para cada PI, los recubrimientos óptimos del grafo de la permutación y posteriormente compara los recubrimientos de todas las PI, escogiendo aquella o aquellas soluciones que recubren su grafo con el menor número de caminos posibles.

3.3.3 Concepto de velocidad por longitud de metal.

Obviamente, cuanto mayor sea la longitud de una conexión metálica (CM) en una implementación, mayor será el tiempo de propagación y en consecuencia menor velocidad de respuesta del circuito. Por esta razón la optimización consistirá en escojer aquellas soluciones óptimas VA cuyas conexiones metálicas sean mínimas en longitud.

Los caminos que forman el recubrimiento de una permutación ofrecen inherentemente la cuantificación de la longitud. La longitud de una línea metálica viene dada por el número de transistores y/o difusores de vacío dispuestos entre dos vértices coincidentes.

Por ejemplo, sea el grafo de una permutación de interés correspondiente a la ecuación: Z = (+/* d e (*/+ a (+/* b c)))

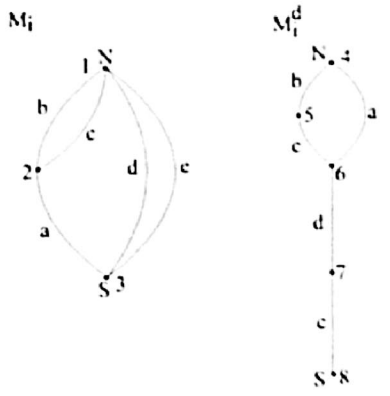


Figura 3-8 Grafo de una permutación de interes para la ecuación Z = (+/ d e (*/+ a (+/* b c)))*

Un camino dual de un recubrimiento posible del grafo anterior es:

{2 c 1 b 2 a 3 a 1 e 3} ⟶ Red N

{6 c 5 b 4 a 6 a 7 e 8} ⟶ Red P

A partir de él se obtienen las longitudes de las conexiones metálicas. Su formato es L_v donde L es la longitud de la conexión metálica y v el vértice que genera la conexión.

Red NMOS : 2_2 2_3 3_1

Red PMOS : 3_6

El algoritmo A35 selecciona soluciones VA en base al tiempo estimado de respuesta de cada implementación y no en base a las longitudes de las conexiones metálicas, aunque tiempo y longitud están íntimamente relacionadas.

3.4 ALGORITMO DE OPTIMIZACIÓN DE CÉLULAS

A partir de la teoría necesaria para modelizar un circuito de tipo planar serie palalelo mediante un grafo y su equivalente arbóreo computacional, hemos desarrollado un algoritmo optimizador de células con el objetivo de optimizar el tiempo de propagación de la célula (que denominamos A35).

Este algoritmo permite dos tipos de optimizaciones celulares, una de ellas nos entrega todas las soluciones óptimas en velocidad operacional (V), entre todas esas soluciones selecciona las óptimas en anchura (VA), y por último entre ellas escoje las de longitudes metálicas más cortas (VAV), lo que intrínsecamente implica una mayor velocidad. Las células asi optimizadas se denominan VAV. La otra nos dará como resultado las soluciones óptimas en anchura (A) y entre todas ellas las de menor velocidad por conexión metálica (AV), es decir, los caminos óptimos en anchura; en este caso las células optimizadas se denominan AV.

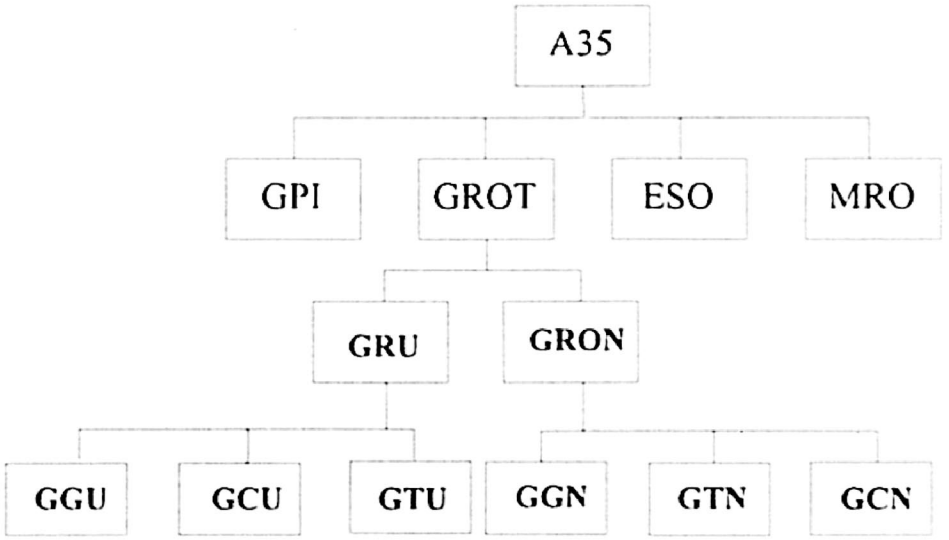


Figura 3-9 Diagrama de bloques del algoritmo A35.

Como se observa en la figura anterior, el algoritmo A35 se compone de varios módulos:

- Módulo 1: GPI o Generador de Permutaciones de interés
- Módulo 2: GROT o Generador de recubrimientos óptimos totales

- Submódulo 2.1: GRU o Generador de recubrimientos unienlace

-Submódulo 2.1.1: GGU o Generador de grafo unienlace

-Submódulo 2.1.2: GCU o Generador de caminos unienlace

-Submódulo 2.1.3: GTU o Generador de tipos de caminos unienlace

-Submódulo 2.2: GRON o Generador de recubrimientos óptimos por nodo

-Submódulo 2.2.1: GGN o Generador de grafo de nodo

-Submódulo 2.2.2: GTN o Generador de tipos de caminos de nodo

-Submódulo 2.2.3: GCN o Generador de caminos de nodo

-Módulo 3: ESO o Explosión de soluciones óptimas

-Módulo 4: MRO o Modelador de recubrimientos óptimos

Los datos de entrada para éste algoritmo son la ecuación en notación postfija y el tipo de optimización deseada.

3.4.1 Generación de PI: módulo GPI

El generador de permutaciones de interés realiza, en función del tipo de optimización celular y del número de nodos profundos de la ecuación a implementar, una de las siguientes acciones:

a) Si la ecuación posee un nodo profundo (NP), genera todas las permutaciones posibles que situen dicho nodo profundo en una de las dos caras profundas de un árbol correspondiente a una reordenación. Así, se consigue que el nodo urgente quede

implementado cerca de VDD/GND o SALIDA, según la reordenación o permutación de interés escogida.

Ejemplo: Sea el PI de la ecuación: Z = (+/* (*/+ (+/* bc) a) def), Partiendo del árbol inicial Ti:

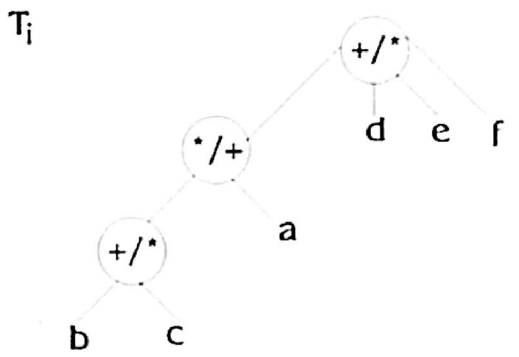


Figura 3-10 Ejemplo de un árbol y sus permutaciones de interés.

Las permutaciones correspondientes a la ubicación de un nodo profundo en una de las caras externas del árbol son:

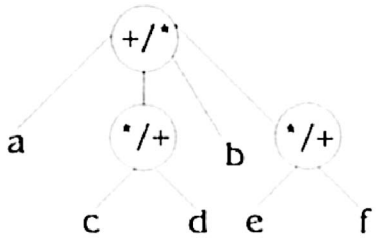
def	dfe	edf	efd	fed	fde
a	a	a	a	a	a
bc	bc	bc	bc	bc	bc

def	dfe	edf	efd	fed	fde
a	a	a	a	a	a
cb	cb	cb	cb	cb	cb

El hecho de posicionar el nodo profundo en una de las caras externas reduce el número de permutaciones posibles. Para optimizar la anchura, ésta es simplemente una regla heurística.

b) Si la ecuación posee dos nodos profundos, genera todas las permutaciones posibles situando cada nodo en una de las caras externas; de esta forma la implementación de los nodos urgentes quedan en las zonas más rápidas de evaluación del circuito.

Ejemplo: Sea el CPI para la evaluación de la ecuación: Z=(+/*a (*/+cd)b (*/+ef))
Partiendo del árbol inicial Ti se obtienen todas las permutaciones de interés. Arbol inicial:



Permutaciones de interés:

ab	ab	ab	ab
cd	ef	dc	ef
dd	fe	dc	fe
ba	ba	ba	ba
cd	ef	dc	ef
cd	fe	dc	ef
ab	ab	ab	ab
ef	de	ff	cd
ef	dc	fe	dc

ba ; ba ; ba ; ba
ef cd fe cd ef dc fe dc

c) Si la ecuación posee más de dos nodos profundos, genera todas las permutaciones posibles para la ecuación dada sin tener en cuenta las caras externas. En el mejor de los casos la evaluación de la implementación de un nodo urgente se retrasará, no siendo posible optimizar la velocidad operacional.

En este caso, todas las permutaciones posibles son PI, por lo que es indiferente el tipo de optimización. La salida de este módulo es, por tanto, el conjunto de permutaciones árboles de interés. A partir de éste conjunto el algoritmo genera el grafo de permutación y sus recubrimientos óptimos posibles, y consecuentemente, todos los recubrimientos óptimos para el grafo de cada reordenación o permutación de interés.

3.4.2 Generador de recubrimientos óptimos totales: Módulo GROT

Por cada permutación o árbol el módulo GROT lo recorre en post-orden y genera un conjunto de recubrimientos óptimos. Este módulo implementa una función recursiva y llama a los submódulos GRON y GRU. El módulo GRU genera los recubrimientos unienlace (a nivel de hoja) y el GRON los recubrimientos óptimos por nodo (Figura 3-11)

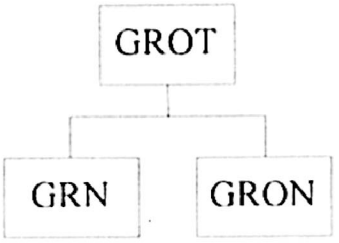


Figura 3-11 Módulo GROT

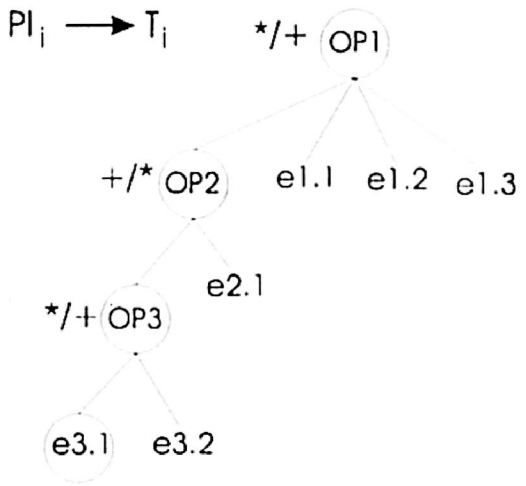


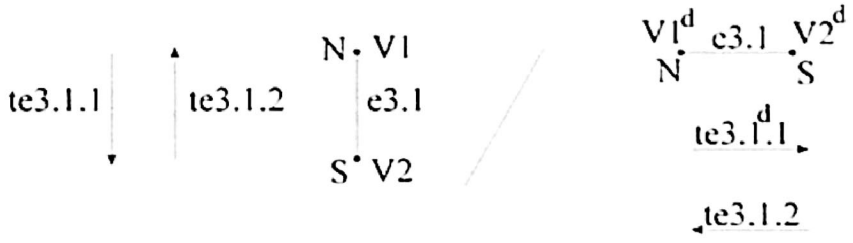
Figura 3-12 Árbol correspondiente a una permutación (PI_i).

El módulo GROT comienza la exploración del arbol por el nodo raiz (OP1). Puesto que éste es un nodo interno, GROT explora el nodo hijo de OP1 situado más a la izquierda (OP2). Este es también un nodo interno por lo que como en el caso anterior se pasa a explorar el primer hijo de OP2 situado más a la izquierda (OP3). Como OP3 es un nodo interno se explora su primer hijo situado más a la izquierda (e3.1) que es una hoja. Una vez localizado un nodo hijo tipo hoja se llama al módulo GRU (Figura 3-13). Este

módulo para obtener los recubrimientos y tipos correspondientes para dicho nodo, realiza las siguientes llamadas:

a) A la función GGU que genera el grafo unienlace para la hoja dada.

b) A la función GCU que genera los caminos para el grafo unienlace calculado por GGU, considerando las secuencias de vértices y los tipos de operadores (+/*, */+) de sus nodos padres.



c) A la función GTU que crea los tipos de cada camino generado, según los tipos de vértices del grafo recubierto.

$$\begin{Bmatrix} NS \\ NS \end{Bmatrix} \quad \begin{Bmatrix} SN \\ SN \end{Bmatrix}$$

$T(te_{31.1})$ $T(te_{31.2})$

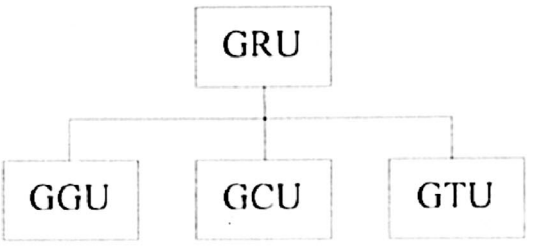
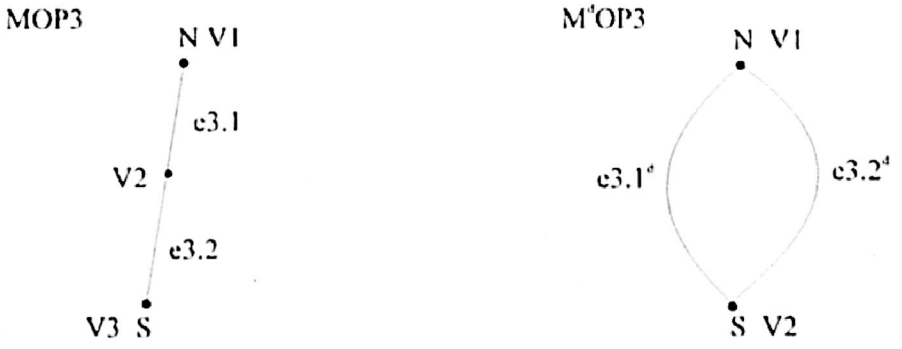


Figura 3-13 Módulo GRU.

Una vez procesada la hoja e3.1, GROT (OP1.OP2.OP3) explora el siguiente hijo de OP3 a la derecha (e3.2), repitiéndose las operaciones anteriores al ser un nodo tipo hoja.

Terminada la exploración de todos los nodos hijos del OP3, se procede a generar el recubrimiento óptimo en dicho nodo. Esto lo realiza el módulo GRON llamando a los módulos siguientes:

a) GGN, que genera los grafos primal y dual del nodo en cuestión según los grafos de sus hijos y tipo de operador (+/*,*/+).



b) GTN, que se encarga de la formación de los recubrimientos óptimos y de sus tipos a partir de los grafos generados por el módulo GGN.

c) GCN, que genera los caminos para el grafo del nodo dado, a partir del conjunto de recubrimientos (CRM) obtenidos en el paso anterior.

A partir de estos caminos para cada recubrimiento, se procede a realizar una permutación de los mismos y de sus orientaciones, para obtener todas las posibles interconexiones celulares.

3.4.3 Explosión de soluciones óptimas

Uno de los objetivos del presente trabajo de investigación es la optimización en velocidad y anchura, lo que implica ciertos requisitos para la posterior interconexión celular. Así, el algoritmo A35 calculará a partir de CRM el conjunto total de caminos posibles. Para ello, el módulo ESO de este algoritmo permuta la orientación de cada camino de cada recubrimiento contenido en CRM asociado a una permutación dada. Dicha operación afecta a las posibilidades de interconexión entre terminales, obteniéndose posibles conexiones metálicas más o menos largas.

Por ejemplo, dados los grafos Mi/Mi^{d} correspondiente a una Pii (Figura 3-14), y el recubrimiento Ri,j de dicha permutación:

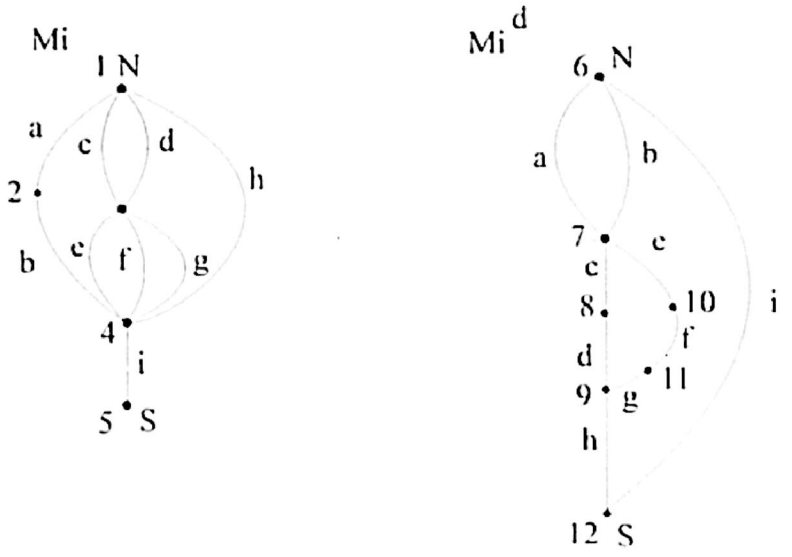


Figura 3-14 Grafos Mi Mi^d correspondiente a una Pli

$$Rij = \left\{ \begin{matrix} 1\ a\ 2\ b\ 4\ i\ 5\ \#\ 3\ d\ 1\ c\ 3\ e\ 4\ f\ 3\ g\ 4\ h\ 1 \\ 6\ a\ 7\ b\ 6\ i\ 12\ \#\ 9\ d\ 8\ c\ 7\ e\ 10\ f\ 11\ g\ 9\ h\ 12 \end{matrix} \right\} = \left\{ \begin{matrix} t1\ \#\ t2 \\ t1\ \#\ t2 \end{matrix} \right\}$$

Las longitudes NMOS son: 6_4, 2_4, 2_3, 2_3, 6_1, 5_1 y las longitudes PMOS: 2_6, 9_{12}, 5_9, 7_7

Permutando la orientación del camino t2 se obtiene otro recubrimiento:

$$Ri,j+1 = \left\{ \begin{matrix} 1\ a\ 2\ b\ 4\ i\ 5\ \#\ 1\ b\ 4\ g\ 3\ f\ 4\ e\ 3\ c\ 1\ d\ 3 \\ 6\ a\ 7\ b\ 6\ i\ 12\ \#\ 12\ h\ 9\ g\ 11\ f\ 10\ e\ 7\ c\ 8\ d\ 9 \end{matrix} \right\} = \left\{ \begin{matrix} t1\ ,\ t1p \\ t1\ ,\ t2p \end{matrix} \right\}$$

Ahora las longitudes de las CM son para la red NMOS: 5_1, 5_1, 4_4, 4_2, 3_2, 3_2 y para la red PMOS: 6_2, 8_7, 9_5, 2_{12}.

El resto de posibilidades de interconexión dentro de una célula las calcula éste módulo mediante la permutación de los caminos dentro de un recubrimiento.

Por ejemplo, suponiendo el recubrimiento Rij={t1#t2}, el módulo ESO calcula un nuevo recubrimiento Rij+1={t2#t1}, donde se habrán cambiado la orientación de los caminos t2 y t1, para no variar las longitudes de conexión. Posteriormente vuelve a permutar la orientación de cada camino con el fin de buscar CM mínimas.

En resumen, el módulo ESO permuta en cada camino su orientación y en el recubrimiento la orientación de los caminos, resultando así una auténtica explosión de recubrimientos. A partir de éstas soluciones óptimas en velocidad y anchura, se seleccionan las óptimas en velocidad por longitud de conexión metálica.

3.4.4 Selección de recubrimientos óptimos VA por longitudes de conexiones metálicas

El módulo MRO a partir del conjunto CRMU selecciona los recubrimientos que impliquen conexiones metálicas mínimas.

Dado un recubrimiento (dual o primal) es posible conocer las longitudes de las líneas de metal por el número de transistores o enlaces existentes entre vértices coincidentes. Así, en el recubrimiento siguiente, {V1 e1 V2 ... m terminales.. V2 V3 ... n

terminales ... V3 ... Vm} la longitud metálica entre los vértices V2 será 'm' y entre los vértices V3 será 'n'.

Frecuentemente, las soluciones consideradas óptimas en velocidad operacional y en anchura son desechadas tras un estudio cuantitativo del tiempo, dado que las longitudes utilizadas en sus conexiones metálicas implican una modelización de la solución muy lenta.

Por cada recubrimiento de cada permutación habrá una lista de longitudes metálicas, que se utilizará para el cálculo del tiempo de respuesta de una solución celular.

Una permutación de un árbol original (T_i) tiene un único circuito equivalente, en consecuencia, a un grafo le corresponde un único circuito. Todas las soluciones afines al grafo en cuestión están asociadas al mismo circuito.

En cada solución, varían únicamente las longitudes entre los transistores del circuito. Esta información queda detallada, para cada recubrimiento que se está evaluando, en la tabla 3-1.

La tabla 3-1 representa inherentemente a la tabla correspondiente al recubrimiento de la red P, y a la tabla correspondiente al recubrimiento de la red N.

XMOS	e1	e2	e3	e4	e5	e6	...	ei	...	em
e1		L12	L13	L14	L15	L16		L1i		L1m
e2	L21		L23	L24	L25	L26		L2i		L2m
e3	L31	L32		L34	L35	L36		L3i		L3m
e4	L41	L42	L43		L45	L46		L4i		L4m
e5	L51	L52	L53	L54		L56		L5i		L5m
e6	L61	L62	L63	L64	L65			L6i		L6m
. . .										
ei	Li1	Li2	Li3	Li4	Li5	Li6				Lim
. . .										
em	Lm1	Lm2	Lm3	Lm4	Lm5	Lm6		Lmi		

Tabla 3-1 Tabla de conexiones entre transistores

Ambas tablas se construyen en el momento de analizar cada recubrimiento solución, con lo cual sólo existe una única tabla actualizada para cada red con los valores L_{ij}. Estas tablas se actualizan dinámicamente de modo que en cada instante de ejecución del módulo MRO tienen los valores L_{ij} correspondientes al recubrimiento que se está analizando en dicho instante, donde L_{ij} es la longitud de conexión del drenador del transistor ei a la fuente del transistor ej.

Por ejemplo, sea el recubrimiento siguiente:

$$Ri = \left[\begin{Bmatrix} 2a1b2c3d4e3\#4f5 \\ 6a7b8c6d7e8\#6f8 \end{Bmatrix} \begin{matrix} 5,1,4 \\ 6,8,6 \end{matrix} \right]$$

Estudiando el corespondiente a la red P calcula su tabla asociada. Este cálculo implica la localización de los vértices de números coincidentes y el número de enlaces y/o difusores de vacío que existen entre cada par de ellos (tabla Lp). Del mismo modo, se procede con el recubrimiento de la red N (tabla L_N).

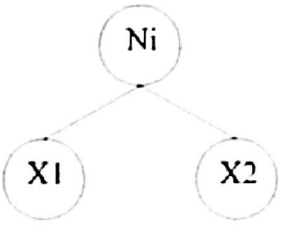
TABLA L_N

NMOS	a	b	c	d	e	f
a	X	0	3	0	0	0
b	0	X	0	3	0	0
c	0	0	X	0	3	6
d	3	0	4	X	0	0
e	0	0	0	0	X	0
f	0	0	4	0	0	X

TABLA L_P

PMOS	a	b	c	d	e	f
a	X	0	0	0	0	0
b	2	X	0	0	0	0
c	0	0	X	0	2	0
d	0	0	0	X	0	3
e	2	0	0	2	X	0
f	0	0	0	0	0	X

A partir de estas tablas se modeliza el grafo correspondiente mediante el cálculo de la expresión de la resistencia y capacidad de cada nodo Ni.



A continuación, se modeliza el circuito equivalente para cualquier tipo de nodo mediante las expresiones de sus resistencia y capacidad resultantes.

Sea el nodo genérico Ni, donde X1, X2 son dos hijos (nodos u hojas) con ciertas resistencias y capacidades (R_{NX1}, R_{PX1}, C_{NX1}, C_{PX1}) y (R_{NX2}, R_{PX2}, C_{NX2}, C_{PX2}), donde, Rjxi es la resistencia de la red j (P o N) asociada al nodo Xi, y Cjxi es la capacidad de la red j de dicho nodo.

Partiendo de esta información se calcula la resistencia y capacidad del nodo Ni.

R_{Ni} = ▼ R_{pNi} Para la red PMOS

▲ R_{nNi} Para la red NMOS

C_{Ni} = ▼ C_{pNi} Capacidad para la red PMOS

▲ C_{nNi} Capacidad para la red NMOS

Si Ni es de tipo */+ sus subcircuitos pueden representarse de la forma siguiente:

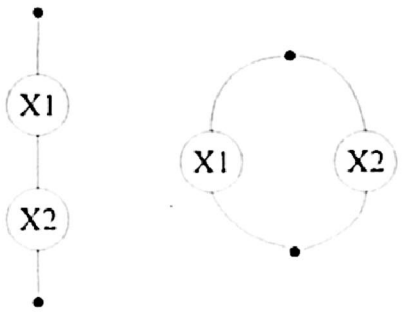


Figura 3-15 Descripción gráfica de los subcircuitos a nivel de nodo y sin tener en cuenta interconexiones entre transistores.

En el circuito del nodo Ni existen siempre dos conexiones lógicas, de X1 a X2 y de X2 a X1. Si estas conexiones son por incrustación, los subcircuitos X1 y X2 están físicamente adyacentes, si no están conectados por conexión metálica. Sus resistencias y capacidades las expresamos así: R_{X1X2}, C_{X1X2}, R_{X1X2}, C_{X2X1}.

Por ejemplo, si el nodo N_i es de tipo */+ puede calcularse la resistencia equivalente del circuito asociado a cada red del mismo mediante las expresiones siguientes:

$R_{PNi} = R_{PX1} + R_{PX1X2} + R_{PX2}$ para la red PMOS

$$R_{XX1} = \frac{(R_{XX1} + R_{XX1X2}) \cdot (R_{XX2} + R_{XX2X1})}{(R_{XX1} + R_{XX1X2}) + (R_{XX2} + R_{XX2X1})}$$ para la red PMOS

A partir del circuito capacitivo se calculan las capacidades de las redes mediante las siguientes expresiones:

$$\text{para la red PMOS} \quad C_{PN1} = \frac{\left(\frac{C_{X1} \cdot C_{X1X2}}{C_{X1} + C_{X1X2}}\right) \cdot C_{X2}}{\left(\frac{C_{X1} \cdot C_{X1X2}}{C_{X1} + C_{X1X2}}\right) + C_{X2}}$$

$$\text{para la red PMOS} \quad C_{nN1} = \left(\frac{C_{X1} \cdot C_{X1X2}}{C_{X1} + C_{X1X2}}\right) + \left(\frac{C_{X2} \cdot C_{X2X1}}{C_{X2} + C_{X2X1}}\right)$$

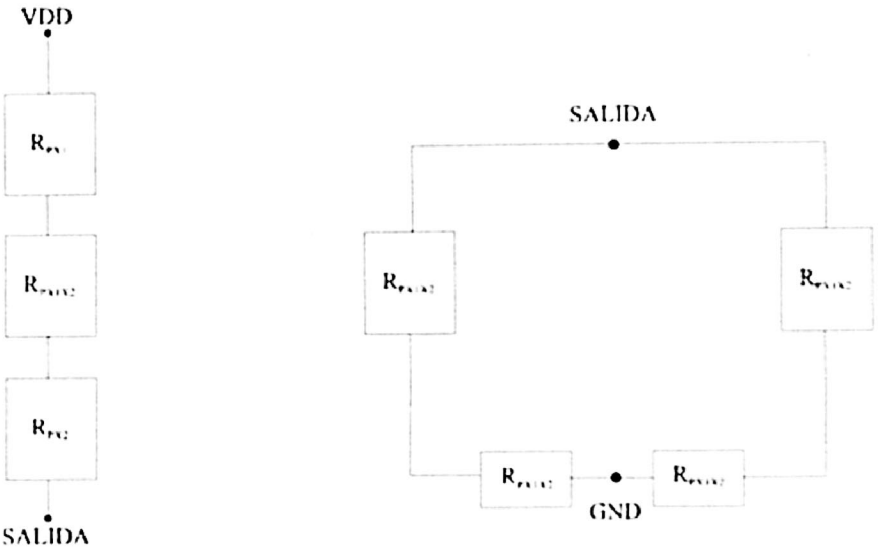


Figura 3-16 Descripción global del circuito resistivo para un operador

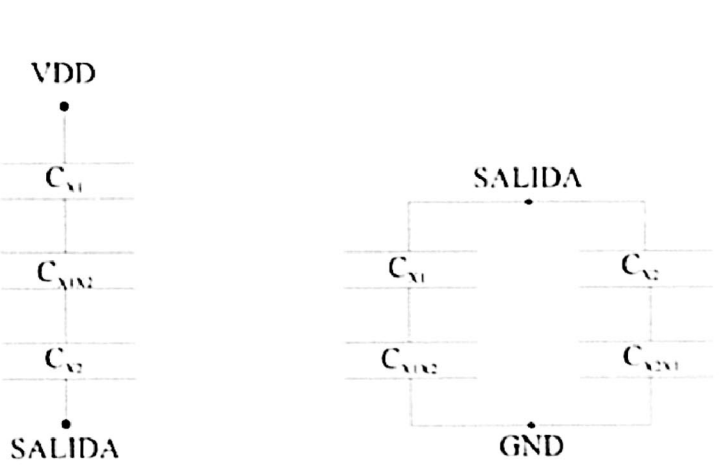


Figura 3-17 Descripción global del circuito capacitivo del nodo

Calculadas estas expresiones el nodo Ni puede tratarse como un nodo hijo de un hipotético nodo superior.

Al recorrer un árbol en post-orden, el módulo GRON además de generar el grafo correspondiente y sus recubrimientos, calcula los valores correspondientes a $R_{N/I,\,X1}$, $R_{N/I,\,X2}$, $C_{N/I,\,X1}$ y $C_{N/I,\,X2}$ y las sustituye en las expresiones de cálculo de resistencia y capacidad de un nodo que utiliza los nodos X1 y X2. Si X1 y X2 son de tipo hoja, $R_{N/I,X1}$, $R_{N/I,X2}$, $C_{N/I,X1}$, y $C_{N/I,X2}$ son parámetros λ y por tanto constantes.

A partir de las longitudes de conexión de cada solución, determinados por las tablas de longitudes de conexión, se calculan $R_{N/I,X1X2}$ y $C_{N/I,X1X2}$. Es decir:

$$R_{N\,X1X2} = L_N\,[X1, X2] \cdot RM_N$$

$$C_{NX1X2} = L_N\,[X1, X2] \cdot CM_N$$

donde RM_N y CM_N son la resistencia y capacidad de una conexión metálica de longitud mínima.

Si la conexión se establece por incrustación, su capacidad y resistencia se consideran 'nulas' en comparación con las de una conexión metálica. Así, la longitud de estas conexiones 'ε' se considera comprendido en el intervalo (0,1).

Finalizado el recorrido en post-orden de un árbol Ti se consigue una constante RC total del circuito. Esta constante la sustituye en la expresión para la tensión de salida:

$$V_{SALIDA}\,(t) = VDD\; e^{-t/RC}$$

A partir de ella se obtiene el tiempo de respuesta para cada red.

$$t_P = RC\ L_N\ V_{SALIDA} / VDD$$

$$t_N = RC\ L_N\ V_{SALIDA} / GND$$

El tiempo mayor de los calculados se seleccionará como representativo de la solución. La razón de eligir el más restrictivo es la necesidad de dar tiempo suficiente a la descarga a través de una red mientras se produce la carga a través de la otra.

3.4.5 Resultados del Algoritmo A35

Este algoritmo selecciona un conjunto de caminos para cada reordenación T_i de un árbol T que representa una ecuación, o lo que es lo mismo, un conjunto de recubrimientos para la reordenación M_i/M_i^d del grafo M/M^d original, considerando cada reordenación como una permutación de interés (Pi_i).

Para obtener células AV, A35 considera todas las permutaciones posibles (m) de interés. En cambio, para células de tipo VAV el número de permutaciones será menor o igual al posible. Así, si el árbol para una célula VAV tiene más de dos nodos profundos, todas las permutaciones (m) son de interés, pero si posee uno o dos nodos profundos, el número de permutaciones de interés (n) será menor que en los casos anteriores ($n \leq m$).

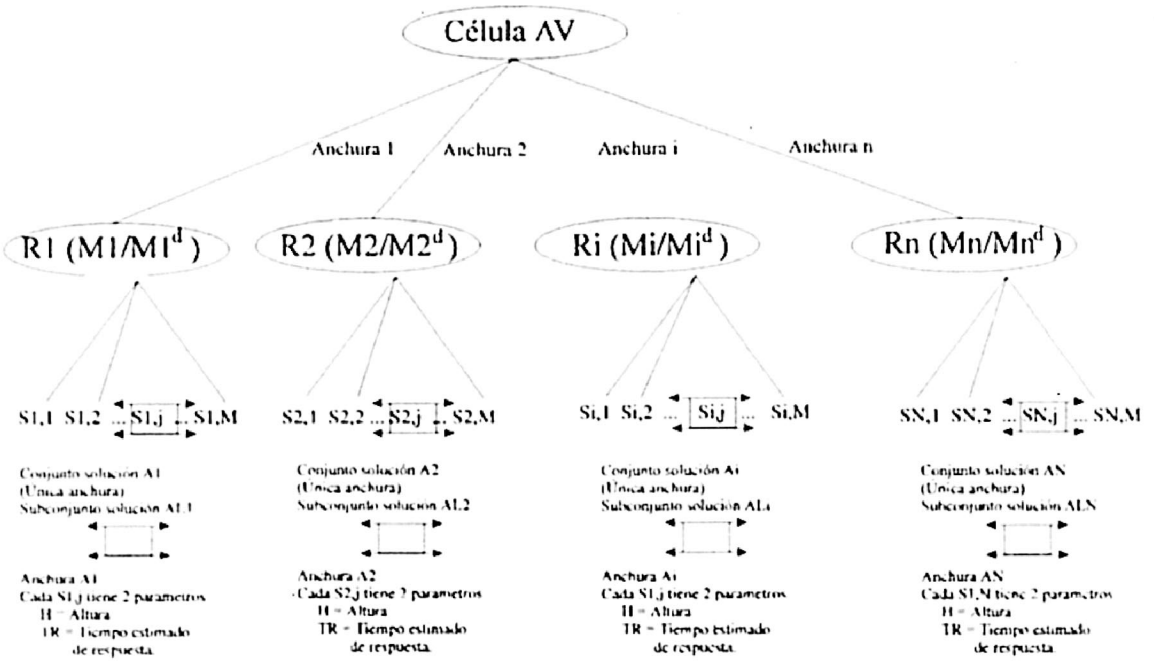


Figura 3-18 Perspectiva gráfica de entrega de resultados del algoritmo A35 para células AV.

El algoritmo A35 entrega para cada recubrimiento R_{ij} de una permutación Pi_i, los siguientes parámetros:

- Tiempo estimado de respuesta.

- Anchura del recubrimiento.

- Altura de las regiones P y N resultantes.

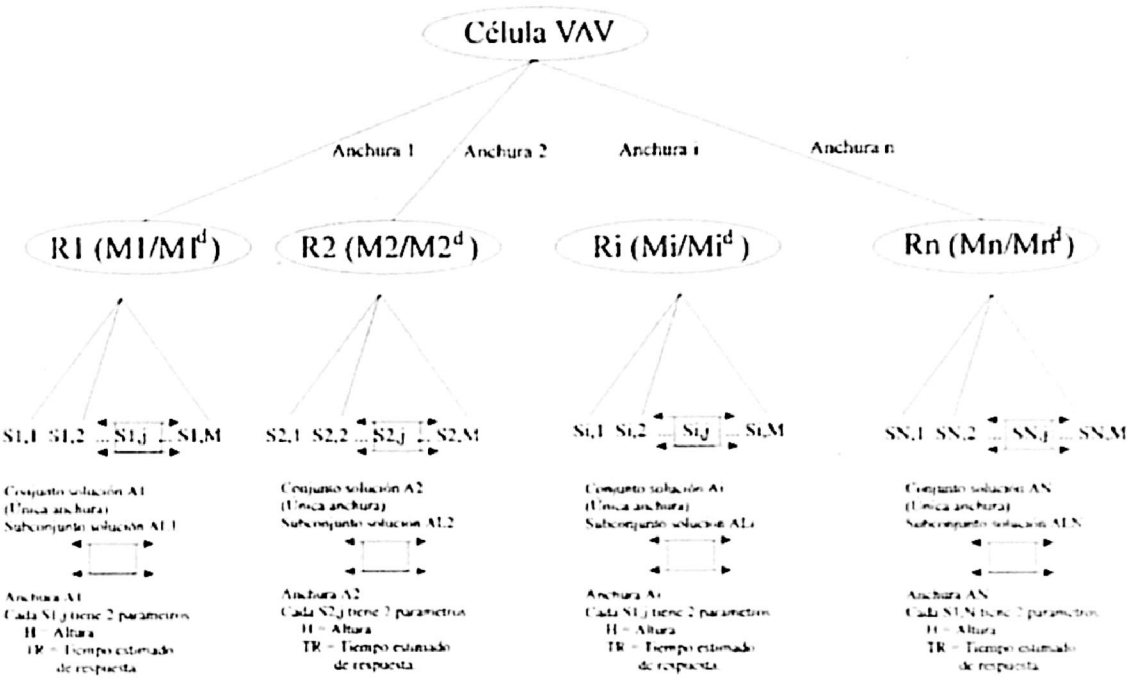


Figura 3-19 Perspectiva gráfica de entrega de resultados del algoritmo A35 para células [11]

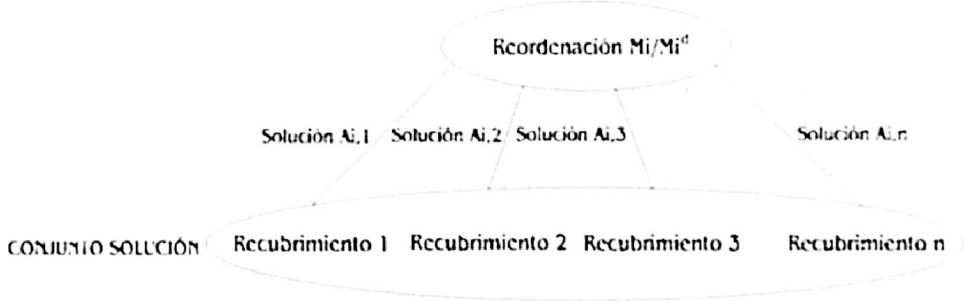


Figura 3-20 Grafo que indica el resultado de una permutación en células [11]

4. ALGORITMO PARA LA OPTIMIZACIÓN DE LOS ARRAYS LINEALES DE CÉLULAS SERIE - PARALELO

4.1 INTRODUCCIÓN

Una vez optimizada cada célula mediante el algoritmo A35, explicado en el capítulo anterior, hemos desarrollado y aplicado un nuevo generador que denominamos GAO (Generador de Arrays Óptimos) para la optimización de los arrays lineales componentes de un CI.

A lo largo de la exposición del mismo, para su mejor comprensión y seguimiento, utilizamos un ejemplo ilustrativo de un array. Este array está formado por cuatro células, cada una de las cuales se representa funcionalmente por una ecuación Booleana:

F1 = (+/* (*/+ a e') (*/+ b e))

F2 = (+/* (*/+ y e') (*/+ d e))

F3 = (+/* (*/+ h f) (*/+ i f))

F4 = (*/+ f g)

4.2 DEFINICIÓN DE ARRAY

Un array está formado por un conjunto de elementos digitales definibles mediante una ecuación booleana e interconectados de forma que la unión de todos ellos realiza una función compleja de 'n' entradas y 'm' salidas. En la figura 4-1 se muestra el esquema electrónico del array ejemplo.

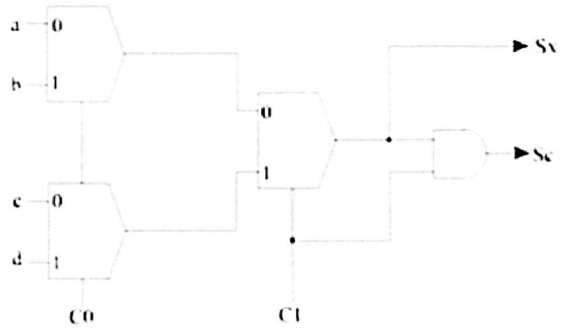


Figura 4-1 Esquema electrónico de un array.

La representación gráfica de un array corresponde al esquema electrónico de sus elementos y conexiones. Los elementos digitales que lo forman pueden representarse mediante bloques funcionales (Figura 4-2). Cada conexión entre bloques funcionales, incluidas sus bifurcaciones, posee una letra única e identificativa de dicha conexión.

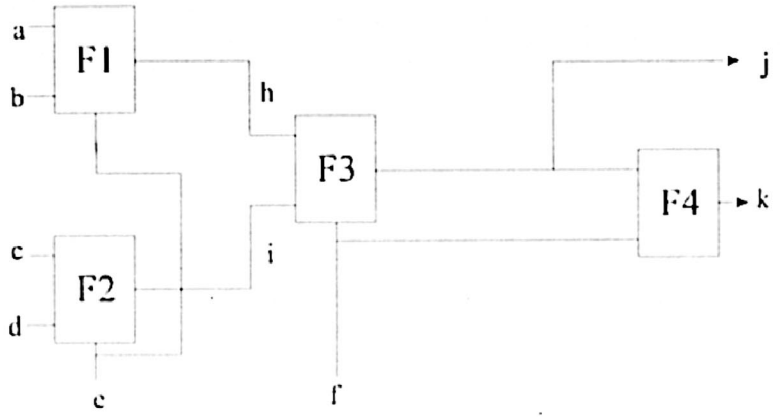


Figura 4-2 Esquema a nivel de bloques funcionales del array.

Un array lineal forma parte de la estructura completa del circuito integrado, donde se aplica un criterio de interconexión entre arrays (Floorplanning) que puede dar origen a conexiones cuya longitud sea crítica para el tiempo de propagación de las señales. Por esta razón, es necesario adoptar las siguientes consideraciones en el diseño de los arrays:

a) Ubicar determinadas células en posiciones concretas del array con el fin de reducir la longitud de conexiones con arrays adyacentes.
b) Cada entrada común a varias células en un mismo array se asocia a una y sólo a una de las células que comparten dicha entrada, con el fin de que la longitud de la conexión con ella de la entrada común sea mínima.

En el array ejemplo de la figura 4-2 se suponen F1 y F4 situados de forma inamovible en los extremos izquierdo y derecho respectivamente del array, al requerir así una longitud mínima su conexión con arrays adyacentes F1 Fx Fx F4. Las funciones

F2 y F3 podrán tomar cualquier ubicación (excepto las de F1 y F4) lo que originará dos posibles permutaciones de las células del array.

Permutación 1.

A R R A Y

CÉLULA F1	CÉLULA F2	CÉLULA F3	CÉLULA F4

Permutación 2.

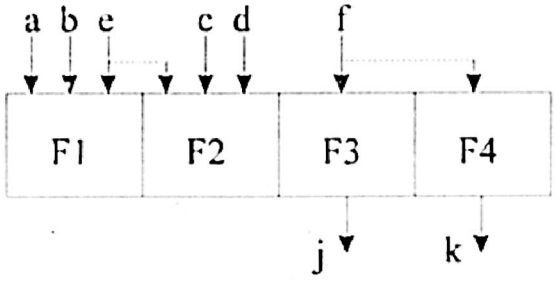
A R R A Y

CÉLULA F1	CÉLULA F3	CÉLULA F2	CÉLULA F4

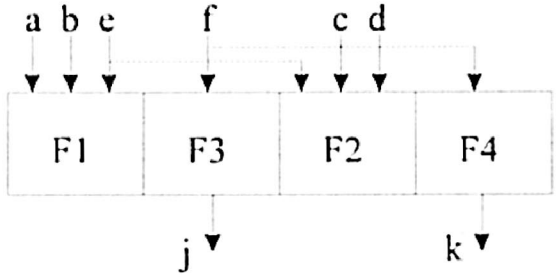
Cuando una misma variable de entrada al array tiene como destino varias células, puede interesar que la longitud de la conexión desde la propia entrada del array a una determinada célula sea lo más corta posible, en cuyo caso se asocia la entrada a dicha célula en cuestión. En el del ejemplo se asume que la entrada común 'e' del array está lo más próxima posible a F1, y la entrada 'f' a F3. Las funciones F2 y F4 no requieren una conexión de longitud mínima con ninguna entrada. En cualquier caso, si una entrada del array tiene como destino varias funciones (células), siempre debe asociarse la entrada a una de las células. Por lo tanto, las permutaciones de las células del array quedan como sigue:

F1 (e) F2 ( ) F3 (f) F4 ( )

Permutación 1.



Permutación 2.

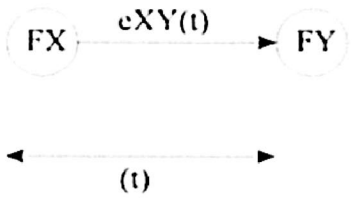


Con el fin de modelizar un array de manera que queden representadas las relaciones existentes entre las células que lo componen, hemos utilizado la técnica de grafos de precedencia. Con dicha técnica se representan los arrays mediante nodos y flechas, donde los nodos simbolizan las funciones (células) y las flechas indican conexiones entre ellas.

La construcción del grafo de precedencia ó grafo relacional de enlaces, para un array dado puede realizarse aplicando las siguientes directrices:

- Todas las permutaciones se representan con un mismo grafo puesto que la única diferencia existente entre dos permutaciones es el tiempo.

- El grafo representa las funciones reales F1... Fm, además de dos funciones virtuales con tiempo de propagación 0. Estas funciones son FE y FS, donde FE determina el nodo temporal del que parten las entradas al array, y FS el nodo final al que confluyen las salidas del mismo.
- Los enlaces entre funciones se codifican de la siguiente forma:



Donde:

- e es el enlace.
- t es el tiempo de propagación para dicho enlace.
- FX, FY son dos células SP cualesquiera.

  Si X=E se trata de una célula o función virtual de entrada.

  Si Y=S se trata de una célula o función virtual de salida.

- En los enlaces donde la función origen es FE, el valor de 't' lo determinará exclusivamente el tiempo de propagación para el enlace.
- En los enlaces donde la función destino es FS, el valor de 't' lo determinará el tiempo de propagación de la función origen.

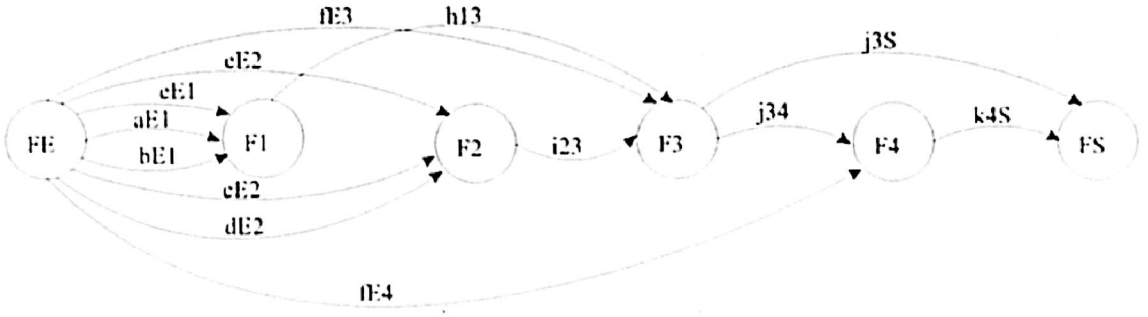


Figura 4-3 Grafo de precedencia para el array ejemplo

4.3 CAMINOS TOTALES

Para el cálculo de los distintos parámetros de un array (Anchura, Altura y Tiempo), este es definido como una macrocélula compuesta por los caminos de las redes P y N de todas las células que lo componen, resultando un camino total (CT) para la red P y red N del array en cada una de las posibles permutaciones.

El primer paso para la construcción de los caminos totales del array consiste en la obtención de los caminos de cada célula o función, con el mínimo tiempo de propagación posible, y la mínima anchura para dicho tiempo. El algoritmo A35 es el responsable de generar éstos caminos en base a la ecuación de la célula y a la premisa especificada como clave V (caminos óptimos en velocidad) asociada a la misma. Por tanto, un array puede representarse de la siguiente manera:

F1 = V (ecuación)

.

.

.

Fm = V (ecuación)

La información a nivel celular, que proporciona el algoritmo A35 es la siguiente:

	[{Camino 1 de la red P} S, P, Hp]		[{Camino n de la red P} S, P, Hp]
F1		. . . .	
	[{Camino 1 de la red N} S, P, Hn]		[{Camino n de la red N} S, P, Hn]
	.		.
	.		.
	.		.
	[{Camino 1 de la red P} S, P, Hp]		[{Camino n de la red P} S, P, Hp]
Fm		. . . .	
	[{Camino 1 de la red N} S, P, Hn]		[{Camino n de la red N} S, P, Hn]

Donde:

- S es la salida celular de la implementación del camino.

- P indica la situación de la alimentación VDD para la red P y GND para la red N.

- Hp y Hn son la altura de la región P y región N respectivamente.

Adicionalmente este algoritmo calcula el tiempo de propagación (T) y anchura (A) de cada célula:

F1 = T, A

.

.

.

Fm = T, A

En la figura siguiente se muestra el organigrama del proceso descrito:

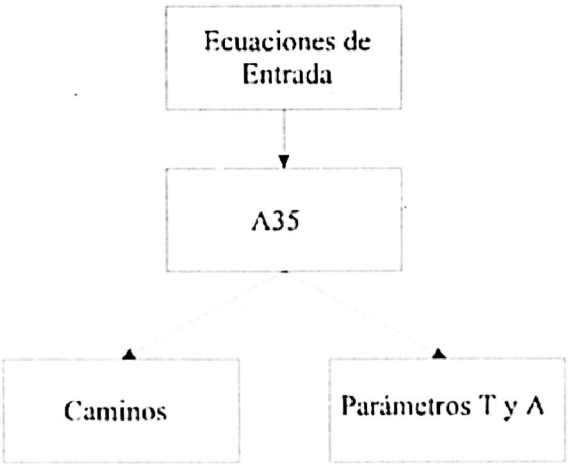
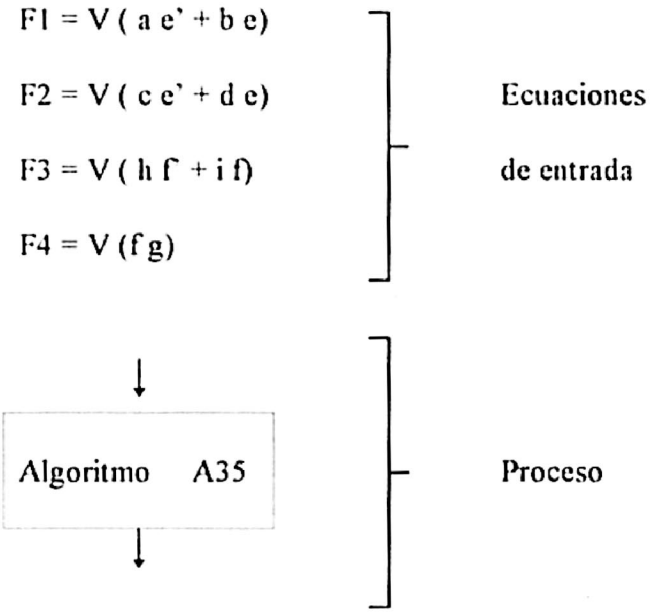


Figura 4-4 Organigrama del proceso

Siguiendo el ejemplo:



F1 [{0Z0#7e'6e5b6a7}7,5,3] . . . . [{0Z0#4e'1e2b3a4}1,3,2]

F2 [{0Z0#7e'6e5d6c7}7,5,3] . . . . [{0Z0#4e'1e2d3c4}1,3,2]

F3 [{0Z0#7f'6f5i6h7}7,5,3] . . . . [{0Z0#4j1f2i3h4}1,3,2]

F4 [{3f4j5}3,5,1] . . . . [{1f2j1}1,2,2]

Salida

Resultados de tiempo y anchura:

F1 = 344'787,7

F2 = 344'787,7

F3 = 344'787,7

F4 = 156'7,2

A partir de todas las posibles combinaciones de los caminos de cada función obtenidos por el algoritmo A35, y mediante nuestro generador de permutaciones (GPCT), se obtiene un conjunto de caminos totales.

CT (P,N) | {Camino red P(F1) ... Camino red P(Fm)} | {Camino red N(F1) ... Camino red N(Fm)}

Ordenando estas combinaciones en función de las posibles permutaciones se generan grupos de caminos.

Permutación 1 | CT(1, 1) = {Camino redes P / Camino redes N} | ... ... | CT(1, N) = {Camino redes P / Camino redes N}

Permutación P | CT(P, 1) = {Camino redes P / Camino redes N} | ... ... | CT(P, N) = {Camino redes P / Camino redes N}

- P es el número de permutación.
- N es el número de camino total para una permutación P.

Los caminos totales obtenidos los utilizará el algoritmo ALG1 para calcular los tiempos correspondientes a los enlaces. El formato de dichos caminos es el siguiente:

CT(Pi,Nj) = (FX,S,P,T) Camino red P para FX (FY,S,P,T) Camino red P para FY... //

(FX,S,P,T) Camino red N para FX (FY,S,P,T) Camino red N para FY...

Donde:

- Pi es el número de la permutación.
- Nj es el número de camino total.
- FX, FY es el número de función.
- S es el terminal del camino correspondiente a la salida de la célula.
- P es el terminal del camino correspondiente a VDD (red P) o GND (red N).
- T es el tiempo de propagación para esa función (célula).

Un camino total del array ejemplo propuesto es:

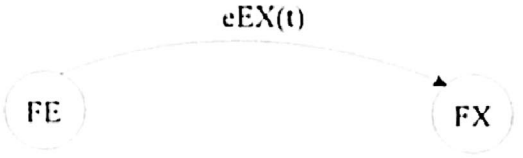
CT (1, 1) = (1, 2, 1, 4) 1 a 2 # 3 b 4 # (2, 3, 1, 5) 1 c 2 d 3 //

(1, 8, 7, 4) 8 a 9 # 7 b 8 # (2, 9, 8, 5) 8 c 9 d 7

4.4 TIEMPOS Y TIPOS DE ENLACES

Analizando los posibles tipos de enlaces puede establecerse la siguiente clasificación en función de las entradas y tipos de células pertenecientes al enlace:

- Enlace tipo A

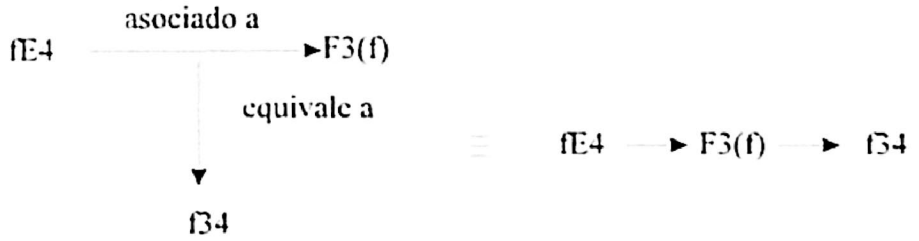


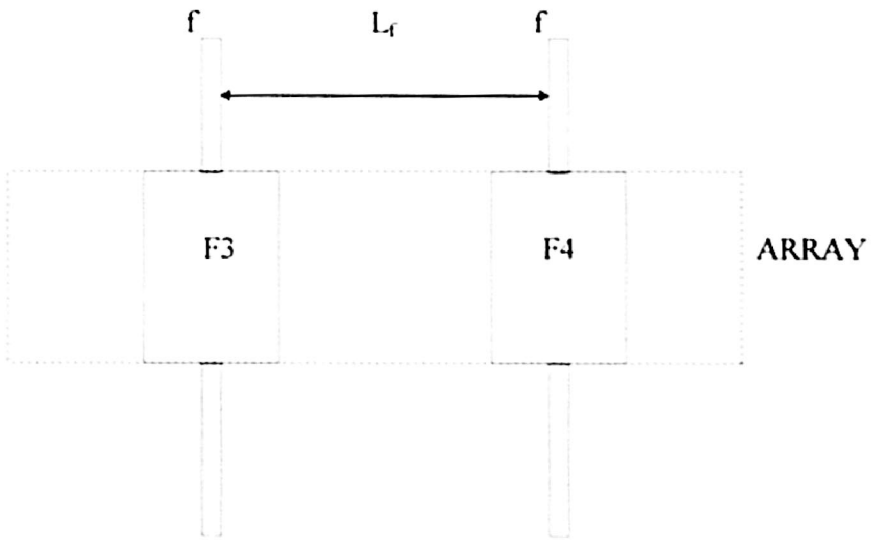
Un enlace 'e' es de tipo A si 'e' es la entrada a una sola función o si dicha entrada está asociada a FX. El tiempo de propagación para dicho enlace es cero: $t = 0$.

Estos enlaces corresponden a las entradas del array siendo su tiempo de propagación igual a cero a nivel de array, al no considerarse, en principio, las longitudes de estas conexiones con el resto de los arrays en un hipotético floorplanning.

- Enlace tipo B

Un enlace 'e' es de tipo B si 'e' es la entrada a varias funciones y si dicha entrada no está asociada a FX. En éste caso, se genera una unión con metal desde la función asociada al resto de las funciones que comparten la misma entrada. El tiempo de propagación para éste enlace es t = Te, y viene determinado por la longitud de metal entre las columnas de polisilicio correspondientes a la entrada de la función asociada al enlace y a la entrada de la función destino, siempre y cuando dicho enlace no esté asociado a una función. En el array ejemplo propuesto al comienzo del capítulo, un enlace de tipo B sería:





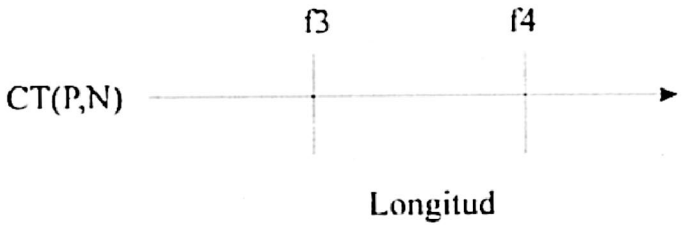
Como el enlace f34 está asociado a F3, la longitud del metal Lf estará determinada por la distancia entre la entrada 'f' de la función F3 y la de la función F4 (f34).

Tanto en la función origen (F3) como en la función destino (F4) sólo puede existir una columna de polisilicio con el nombre del enlace (un sólo transistor para cada entrada en la célula). La longitud de los enlaces de tipo B la calcula el algoritmo ALG1 realizando un rastreo del camino total de izquierda a derecha. Una vez localizada la célula fuente o destino, este algoritmo se posiciona en la entrada del enlace y contabiliza las unidades de longitud hasta encontrar una entrada con nombre idéntico correspondiente a la otra célula perteneciente al enlace. El rastreo del camino total puede realizarse indiferentemente con cualquiera de las dos redes P o N del camino total CT, puesto que las columnas de polisilicio y los difusores de vacío son comunes a ambas redes.

El tiempo para las conexiones metálicas $L^2 \cdot 2'2$, corresponde al tiempo de una conexión de longitud L entre dos columnas de polisilicio. En relación a los difusores de vacío, el tiempo equivalente a la longitud del metal entre dos columnas de polisilicio con un difusor de vacío entre ellas será de $2^2 \cdot 2'2$ (El difusor de vacío debe tener el doble de anchura que un transistor).

Continuando con el ejemplo:

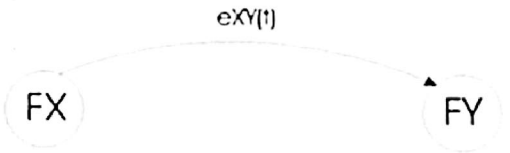
Enlace fE4 -> F3(f) -> f34



f3 : entrada f de la célula origen (F3)

f4 : entrada f de la célula destino (F4)

Enlace tipo C



Un enlace 'e' es de tipo C, si 'e' es el enlace entre la salida de la función FX y una entrada de la función FY. El tiempo de propagación para dicho enlace es t=T FX + Te, es decir, es el resultado de la suma del tiempo de propagación de la célula origen más el de la conexión de metal desde la salida de la célula origen hasta la entrada de la célula destino. Su cálculo es más complejo debido a que la salida de una célula corresponde a una conexión entre la red P y la red N de la misma. Puede darse el caso de que esta conexión establezca además uniones entre varios transistores de una o ambas redes, esto significa que la salida de una célula se puede tomar en varios puntos. Será necesario comprobar cual de ellos supone una conexión óptima (la más corta con la entrada de la célula destino).

Considerando la célula de la figura 4-5 como función origen de un enlace de tipo y asumiendo que la función destino está situada a la izquierda de la misma se comprueba que la salida óptima corresponde a la unión de los transistores (c,d) a través de la región U. Si por el contrario, la célula destino está ubicada a la derecha, la salida óptima será la formada por la unión de los transistores (e,f) a través de la región L.

En resumen, el algoritmo ALG1 entregará para un enlace tipo C, además del tiempo, la región y la salida óptima.

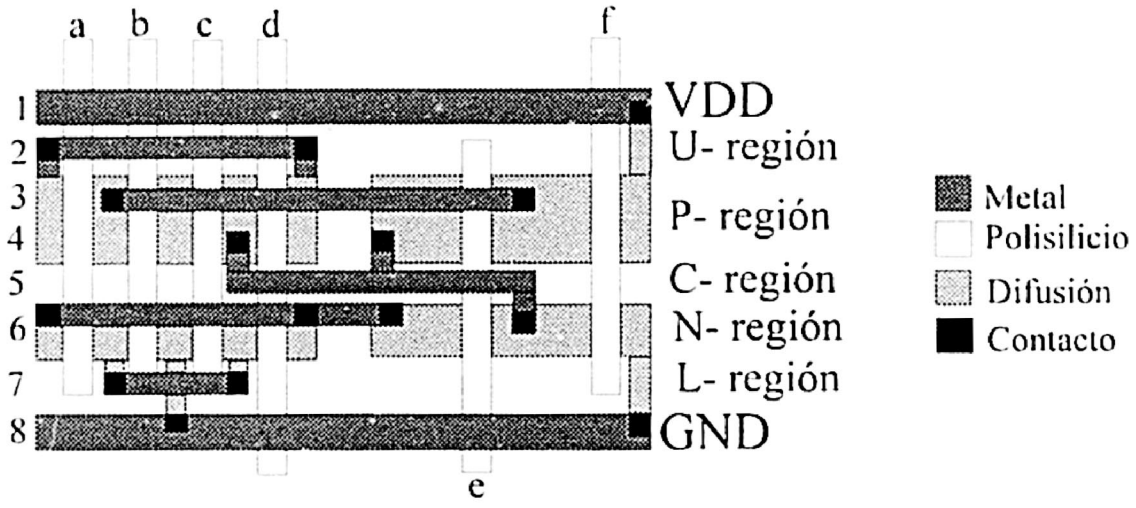


Figura 4-5 Layout de una célula SP.

Con el fin de poder abordar todas estas posibilidades de conexión en un enlace de tipo C, representamos con una flecha la región seleccionada para la salida celular:

$eXY(t)^{\leftarrow}$ Selección de la salida más a la izquierda en la región U

El tiempo corresponde al enlace establecido a través de la región U y tomando la salida situada más a la izquierda.

$eXY(t)^{\rightarrow}$ Selección de la salida más a la derecha en la región U

El tiempo corresponde al enlace establecido a través de la región U y tomando la salida situada más a la derecha.

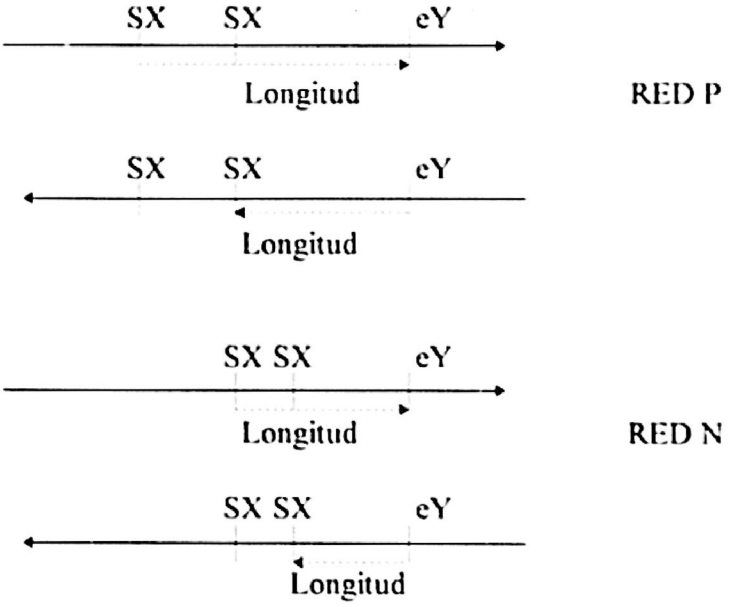
$eXY(t)_{\leftarrow}$ Selección de la salida más a la izquierda en la región L

El tiempo corresponde al enlace establecido a través de la región L y tomando la salida situada más a la izquierda.

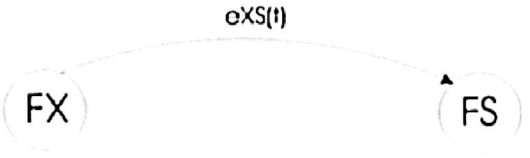
$eXY(t)_{\rightarrow}$ Selección de la salida más a la derecha en la región L

El tiempo corresponde al enlace establecido a través de la región L y tomando la salida situada más a la derecha.

Para obtener la longitud más corta posible, ALG1 realiza un rastreo bidireccional en ambas redes.



- Enlace tipo D



Un enlace 'e' es del tipo D si 'e' es la salida del array. El tiempo asociado a dicho enlace es (t) = T FX.

El algoritmo ALG1 obtiene a partir de los enlaces clasificados de un camino, los tiempos de propagación 't' de los mismos y la anchura total del array. El tiempo de propagación de cada enlace se calcula aplicando un algoritmo específico según su tipo.

El valor T FX (tiempo de propagación de la función x) viene especificado en los caminos totales, por lo que el cálculo de 't' sólo será necesario para los enlaces de tipo B y C.

Los enlaces del grafo deben agruparse en función del tipo al que pertenecen. Así tomando como ejemplo el grafo de la figura 4-3 se obtiene la siguiente especificación:

TIPO A	TIPO B	TIPO C	TIPO D
e E 1 *	e E 2 * *	h 1 3	j 3 S
f E 3 *	f E 4 * *	i 2 3	k 4 S
a E 1			
b E 1			
c E 2			
d E 2			

* Entradas a varias funciones, pero asociadas a la función destino F1(e) F2( ) F3(f) F4( )

** Entradas a varias funciones no asociadas a la función destino.

Este algoritmo considera como longitud óptima la de la red N (región L) calculando el tiempo (t) para dicha conexión y entregando eXY(t) $\rightarrow$ donde (t) será el tiempo de la longitud más el de la célula origen.

Para obtener el tiempo del enlace eXY(t) (salida de la célula X y entrada 'e' de la célula Y) debe calcularse la longitud de metal para la conexión en la región U (red P del camino total), y posteriormente, la longitud del metal para la conexión en la región L (Red N del camino total).

El cálculo de la longitud de metal, en cualquiera de las dos redes, se realiza contabilizando los números de vértices y las variables (transistores) existentes desde el número de vértice correspondiente a la salida de la célula origen hasta la variable entrada de la célula destino. Para realizar este calculo se prescinde de las informaciones auxiliares y se añaden cuatro unidades por cada difusor de vacío (#) encontrado en el recorrido.

Aunque intracelularmente es posible realizar el cálculo de la longitud del enlace en unidades de longitud de transistor, a nivel intracelular no se utilizará esta unidad sino la correspondiente a medio transistor. Si no, sería imposible detallar que la conexión metálica correspondiente a un enlace se extiende desde la salida de una célula (número de vértice) hasta la entrada a una función (barra polisilicia de un transistor).

Para poder utilizar las expresiones de cálculo de tiempos, que utilizan como unidad la longitud de un transistor, se divide entre dos el resultado de dicho cálculo, obteniéndose la longitud de ese recorrido (L). Aplicando la expresión $L^2 \cdot 2.2$ al tiempo relativo de la conexión del metal, se obtiene su tiempo de propagación. La suma del tiempo de propagación celular (función origen) más el tiempo correspondiente a la conexión metálica, dará como resultado el valor 't' para ese enlace.

Considerese el siguiente camino total del array ejemplo, formado por dos células F1 y F2 situadas respectivamente a izquierda y derecha de un enlace tipo C.

F1	[{6c5b6a7d6}6,5]	Red P	F1 = 9,4
	[{1c2b3a1d4}1,3]	Red N	

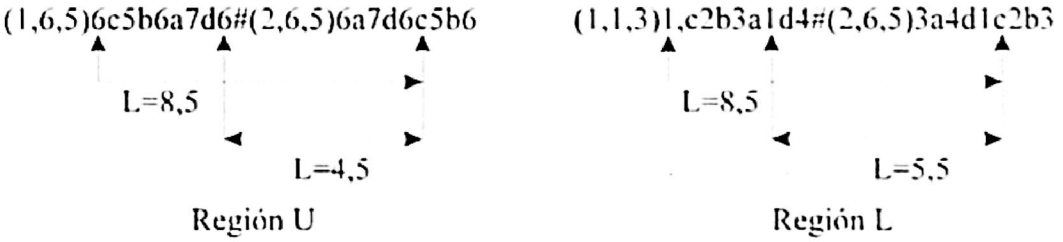
F2	[{6a7d6c5b6}6,5]	Red P	F2 = 7,4
	[{3a4d1c2b3}1,3]	Red N	

El camino total es el siguiente:

CT (1,1) = {6c5b6a7d6#6a7d6c5b6//1c2b3a1d4#3a4d1c2b3}

CT (1,1) = (1,6,5,9)6c5b6a7d6#(2,6,5,7)6a7d6c5b6//
(1,1,3,9)1c2b3a1d4#(2,6,5,7)3a4d1c2b3

Para obtener el tiempo del enlace eXY(t) (salida de la célula 1 hacia la entrada c de la célula 2) debe calcularse la longitud de metal para la conexión en la región U (red P del camino total), y posteriormente, la del metal para la conexión en la región L (Red N del camino total).



El cálculo de la longitud de metal se realiza contando todas las letras y números comprendidos entre la conexión origen y destino, dependiendo del sentido del rastreo, prescindiendo de los contenidos entre paréntesis y añadiendo cuatro unidades por cada símbolo # (difusor de vacío) encontrado en el rastreo. El resultado de esta operación se divide entre dos, siendo el valor obtenido la longitud (L) para ese recorrido. Aplicando la expresión $L^2 \cdot 2^{-}2$ al tiempo relativo de la conexión de metal se obtiene el tiempo de la

conexión metálica. La suma del tiempo de propagación de la célula (función) origen más el tiempo correspondiente a la conexión metálica dará como resultado el valor 't' para ese enlace. La evaluación de las posibilidades de salida del enlace de tipo C en el camino anteriormente elegido es:

c12(17.5)→	Conexión por región U
c12(13.5)←	
c12(17.5)→	Conexión por región L
c12(14.5)←	

De estos resultados se deduce que el enlace óptimo será c12(13.5)←

4.5 ANCHURA TOTAL DEL ARRAY

El algoritmo ALG1 proporciona, además del tiempo de los enlaces, la anchura total del array (ACA) para el camino total en análisis. El cálculo de este parámetro se realiza contando todos los transistores de una de las redes del camino total (las dos son idénticas en anchura), y añadiendo dos unidades por cada difusor de vacío (#) encontrado en el camino.

La anchura del camino total elegido en el apartado anterior para el array ejemplo es:

Red P del camino CT (1,1)

(1,6,5,9)6c5b6a7d6#(2,6,5,7)6a7d6c5b6

▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲ ▲

1 1 1 1 2 1 1 1 1

Anchura del camino total : ACA(1,1) = 10

La unidad para el cálculo de la anchura corresponde a la de un transistor. recuérdese que la anchura de todos los transistores es idéntica.

El resultado del cálculo de (t) para los enlaces del ejemplo propuesto en la permutación F1, F2, F3, F4 es el siguiente:

TIPO A	TIPO B	TIPO C	TIPO D
eE1(0)	eE2(198`55)	h13(1138`987) →	j3S(344`787)
fE3(0)	fE4(55)	i23(385`587) →	k4S(156`7)
aE1(0)		j34 (351`524) →	
bE1(0)			
cE2(0)			
dE2(0)			

4.6 CÁLCULO DE LOS ENLACES CRÍTICOS

El algoritmo ALG2 partiendo de los enlaces etiquetados en tiempo proporcionados por ALG1, calcula las holguras de los mismos y el tiempo de propagación del array.

4.6.1 Caminos críticos

En un array celular, la salida viene determinada por la evaluación de 'n' variables de entrada a través de 'm' células. El tiempo empleado en estas evaluaciones no tiene porqué ser el mismo ya que el tiempo más largo para obtener un resultado, partiendo de una variable de entrada, lo determinarán ciertas células y enlaces.

Por ejemplo, dado el grafo de un array, mostrado en la figura 4-6 siguiente, para poder llevar a cabo la evaluación de la célula ó función F3 es necesario que se hayan evaluado previamente las funciones F1 y F2. Luego, que la evaluación de una de ellas sea más rápida es irrelevante.

El recorrido elegido como camino crítico será el más lento (Recorrido B de la figura) y el tiempo empleado en el mismo será el invertido en la evaluación del array. Para las células no pertenecientes al camino crítico es posible realizar una optimización en anchura. Si esta optimización no supera el tiempo de holgura de las mismas, el tiempo total del array no se verá afectado.

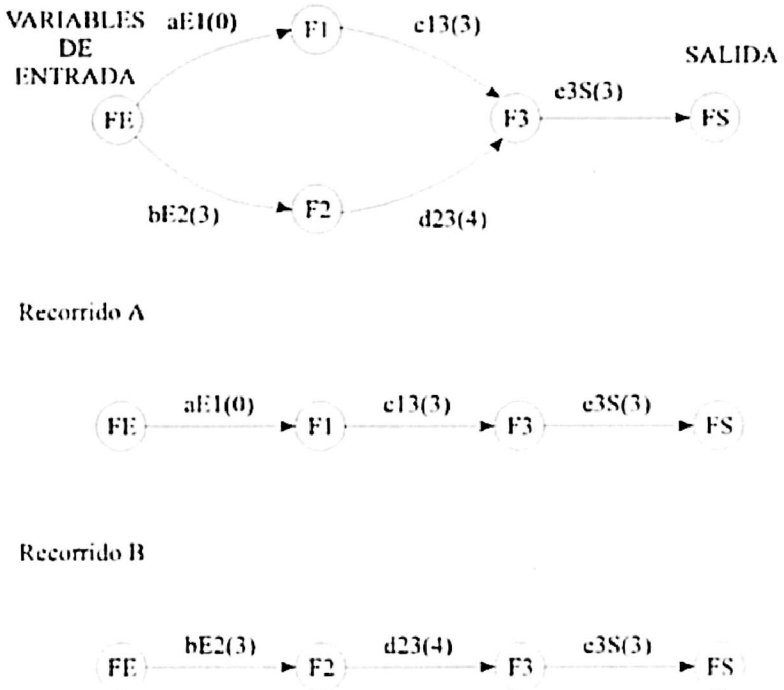


Figura 4-6 Ejemplo de recorridos pertenecientes a un grafo.

La generación de los caminos críticos del array, permitirán obtener el tiempo total de propagación del mismo y el cálculo de las holguras de los enlaces. Para determinar estos caminos hemos empleado el método PERT [CHUE72].

Para aplicar el método PERT hay que evitar la unión de dos funciones mediante varios enlaces; por ello, se eliminan todos los enlaces del tipo A cuyo tiempo es igual a cero, esto no afecta ni al tiempo total ni a las holguras de los restantes enlaces. Sin embargo, puesto que este método exige que toda función tenga al menos una entrada, los enlaces 'eE1' y 'eE2' de la figura 4-3 no son eliminados a pesar de ser de tipo A. Aplicando lo expuesto al ejemplo en desarrollo, se obtiene el siguiente grafo:

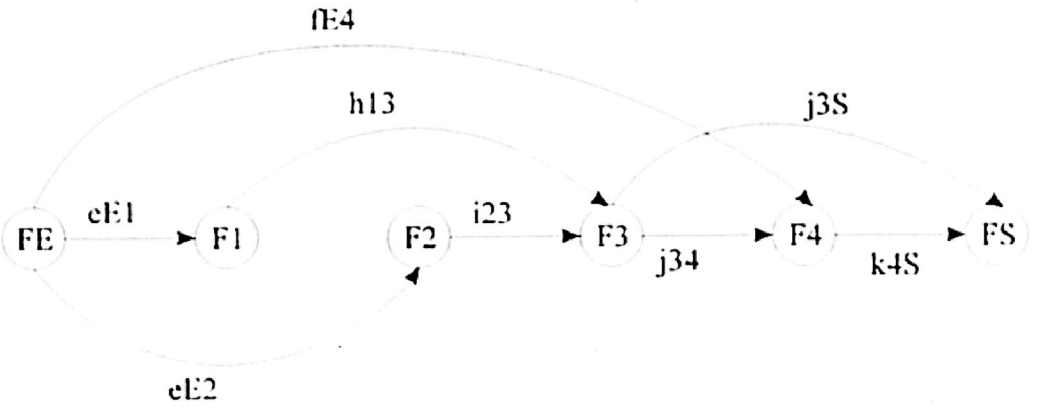


Figura 4-7 Grafo relacional de enlaces adaptado al método PERT.

Una vez adaptado el grafo se elabora la matriz relacional de enlaces, que representa por medio de un 1 la conexión entre dos células. El triángulo inferior izquierdo será simétrico, no siendo necesaria su resolución.

	FE	F1	F2	F3	F4	FS
FE		1	1		1	
F1				1		
F2				1		
F3					1	1
F4						1
FS						

Tabla 4-1 Matriz relacional de enlaces del grafo.

Para la obtención de la tabla de tiempos se construye otra matriz, a partir de la anteriormente calculada, llamada matriz para la reordenación del grafo. Esta matriz se obtiene de la siguiente forma:

a) En la primera columna se dispone la suma de los unos correspondientes a las filas de la matriz relacional de enlaces.

b) Las posiciones de las columnas cuyo valor ha resultado cero, determinan las funciones cuyas columnas en la matriz relacional de enlaces deben eliminarse dichas funciones .

Este proceso, se repite para la obtención de las restantes columnas de la matriz para la reordenación.

	FE	F1	F2	F3	F4	FS
FE	3	3	2	2	0	
F1	1	1	1	0		
F2	1	1	1	0		
F3	2	1	0			
F4	1	0				
FS	0					

FS F4 F3 F1 FE

F2

Tabla 4-2 Matriz para la reordenación del grafo

Como resultado se obtiene un nuevo orden en el grafo. En el ejemplo en desarrollo, la nueva ordenación es la siguiente:

$$FE\begin{Bmatrix} F1 \\ F2 \end{Bmatrix} F3, F4, FS$$

Ordenado el grafo, se procede a la construcción de la tabla de tiempos, a partir del cual se obtendrán los caminos críticos y las holguras de los enlaces.

F.Origen	F.Destino	Enlace	T.Inic	T.Proc	T.Fin	T.Fout	Holgura
FE	F1	eE1	0	0	0	0	0
FE	F2	eE2	0	198`55	198`55	753`4	554`85
F1	F3	h13	0	1138`987	1138`987	1138`987	0
F2	F3	i23	198`55	385`587	584`137	1138`987	554`85
FE	F4	fE4	0	55	55	1490`511	1435`511
F3	F4	j34	1138`987	351`524	1490`511	1490`511	0
F3	F3	j3S	1138`987	344`787	1483`774	1647`211	163`437
F4	FS	k4S	1490`511	156`7	1647`211	1647`211	0

T.Inic: Lo más pronto posible que puede comenzar una función.

T.Proc: Duración de cada proceso.

T.Fout: Lo más tarde posible que puede acabar una función.

T.Fin: Lo más pronto posible que pueden comenzar las funciones que partan de la función origen.

Tabla 4-3 Tabla de tiempos para el grafo.

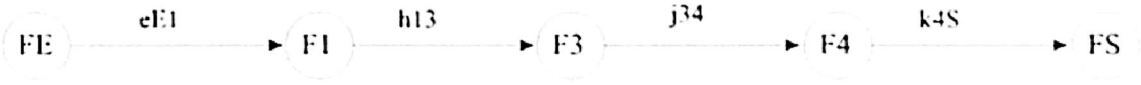
En esta tabla se observa que los los enlaces críticos resultantes son:

[eE1] , [h13] , [j34] , [k4S]

4.6.2 Funciones Optimizables en Anchura

Todos los enlaces cuya holgura sea igual a cero se consideran enlaces críticos y cualquier modificación de su tiempo afectará directamente al tiempo total del array; por consiguiente, en el caso de ser enlaces del tipo C, sus funciones origen no podrán ser optimizadas en anchura.

A partir de los enlaces críticos calculados en la tabla de tiempos anterior resulta el camino crítico para una permutación (P,N), con un tiempo igual a la suma de los tiempos de proceso (T.Proc) correspondientes a los enlaces que lo constituyen:



Así, el tiempo del camino crítico (CCT(P,N)) ó tiempo total del array será de 1687.211 unidades de tiempo.

La tabla de tiempos permite extraer información sobre los enlaces clasificados cuyas holguras son superiores a cero, sobre el máximo tiempo que pueden emplear las funciones origen de dichos enlaces, y sobre el número de difusores de vacio de estas células (Tabla 4-4).

Tipo	Nombre	Holgura	Máxima Duración	Función Origen	Máxima Duración	Difusores de Vacio
B	eE2	554'85	753'4	F1	-	-
B	fE4	1435'511	1490'511	F3	-	-
C	i23	554'85	940'437	F2	799'637	2
D	j3S	163'437	508'437	F3	508'224	2

Tabla 4-4 Tabla de enlaces para el grafo.

La máxima duración del enlace: Corresponde a la suma del tiempo para ese enlace más su holgura (tiempo máximo que puede dedicarse a dicho enlace).

La máxima duración: Indica el tiempo máximo que puede asignarse a la propagación de la función origen. En los enlaces de tipo B este tiempo hace referencia sólo a la conexión metálica. Con respecto a los enlaces de tipo C, el tiempo se obtiene a partir de la duración máxima del enlace con menor tiempo de conexión metálica. Para los de tipo D este valor se corresponden con el de máxima duración de los enlaces.

Partiendo de estos parámetros se genera la lista de funciones optimizables en anchura. Estas funciones deberán cumplir los tres requisitos siguientes:

a) No pertenecer al camino crítico.

b) Poseer al menos un difusor de vacío.

c) Poseer enlaces de tipo C ó D.

En la tabla 4-2 la única función optimizable en anchura es F2. La optimización se realiza invocando al algoritmo A35 el cual dará como resultado los caminos celulares óptimos en anchura para esta célula, de los cuales se seleccionan, mediante el generador de caminos válidos (GCV), los que mejor se ajustan en tiempo al parámetro Función máxima duración.

El generador de camino total (GCT) sustituye el camino de la célula correspondiente a dicha función por cada uno de los caminos seleccionados, obteniéndose un camino total por cada sustitución. A continuación, por medio de los algoritmos ALG1 y ALG2, como se ha descrito anteriormente se obtiene el tiempo y anchura total del array para cada uno de ellos.

4.7 ESTRUCTURA DEL ALGORITMO GENERADOR DE ARRAYS ÓPTIMOS

El algoritmo Generador de Arrays Óptimos (GAO) se basa en la ejecución iterativa de un conjunto de algoritmos agrupados en cinco bloques (Apéndice B). En la figura 4-8 estos bloques están representados por los diagramas de flujo B1, B2, B3, B4. Para realizar estos diagramas hemos partido del diagrama de contexto básico del mencionado algoritmo. En ellos no quedan reflejadas las condiciones de iteración.

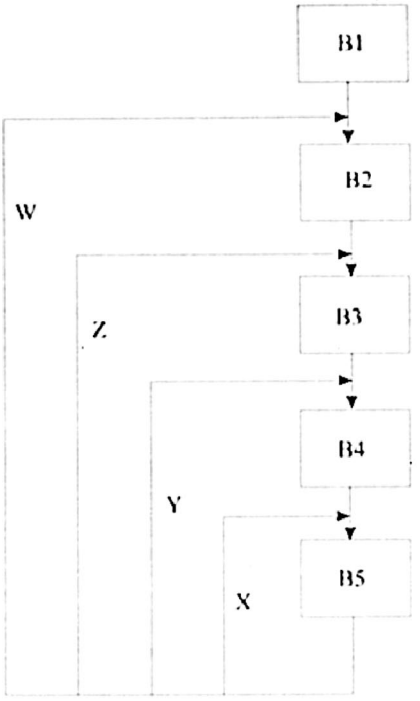


Figura 4-8 Particionamiento en bloques del Generador de Arrays Óptimos

Los flujos de ejecución recibidos por cada diagrama son:

X - Flujo para el análisis de los caminos de una misma función optimizada.

Y - Flujo para el análisis de las funciones optimizables.

Z - Flujo para el análisis de los caminos totales de una permutación.

W - Flujo para el análisis de las permutaciones posibles de un array.

Los datos de entrada al algoritmo son los enlaces, las funciones con premisas y las ecuaciones de todas las funciones. En el diagrama B1, se refleja como el generador de enlaces clasificados (GEC) genera, partiendo de los enlaces y de las funciones con premisas, una lista de enlaces clasificados por tipos. Por otro lado, el generador de

permutaciones del array (GPA) obtiene la lista de posibles permutaciones del array, a partir de las funciones con premisas. Paralelamente, el algoritmo A35 partiendo de las ecuaciones de las funciones a implementar, confeccionará una lista de caminos celulares óptimos en velocidad. Este algoritmo se llamará tantas veces como funciones contenga el array. El generador de permutaciones (ver diagrama B2) de caminos totales (GPCT) que tiene como entrada una permutación del array y el conjunto de listas de caminos óptimos en velocidad, genera todos los caminos totales posibles para dicha permutación del array. Por cada camino se llama al algoritmo ALG1 que con los enlaces clasificados, obtenidos por GEC, calcula la anchura total del array y los enlaces con sus tiempos correspondientes. Esta información es empleada por el algoritmo ALG2 para generar las holguras de los enlaces y el tiempo de propagación del array. Las prestaciones del array quedan definidas por su anchura y tiempo totales, por ello el siguiente paso será la optimización en anchura del array. El generador de funciones optimizables (GFO) genera una relación de funciones, que debido a su holgura de tiempo, son susceptibles de optimización en anchura (ver diagrama B3).

Los diagramas B4 y B5 abordan las posibles optimizaciones en anchura de las células. En B4 se muestran los pasos y algoritmos necesarios para generar todos los caminos óptimos en anchura a partir de la ecuación de cada función optimizable de la relación confeccionada por el GFO. El objetivo el generador de caminos válidos es seleccionar, entre todos los caminos óptimos en anchura de la función a optimizar, aquellos que mejor se ajusten a la holgura disponible para la misma, creando una lista de caminos para dicha función. El generador del camino total (GCT) para cada camino de esta lista, con el resto de caminos celulares que forman el array proporciona un nuevo

camino total para la permutación en análisis. Dicho camino total y los enlaces clasificados, calculados por GEC, son procesados por ALG1 para calcular la nueva anchura total del array el tiempo de cada enlace. Con los enlaces etiquetados a tiempo, el ALG2 calcula el nuevo tiempo total del array. Tanto la anchura como el tiempo total del array comparados con los valores obtenidos en la iteración anterior. Si el resultado supone una optimización en ambos parámetros o en alguno de ellos sin detrimento del otro, los nuevos valores se consideran como nueva referencia a comparar. Si el resultado es idéntico, se guarda el camino total correspondiente. En caso de obtener peores resultados se descarta el camino total que los generó.

Este proceso es necesario, puesto que una optimización en anchura de una función puede generar unas conexiones del metal entre células con una longitud tal, que su tiempo correspondiente afecte al camino crítico. El proceso de comparación se repetirá para toda la lista de caminos válidos de la función a optimizar, posteriormente, para todas las funciones optimizables según su holgura, y a continuación, para los caminos válidos de una permutación. Por último, se realiza el proceso de comparación para todas las permutaciones posibles dependiendo de las premisas especificadas. Finalmente, y como resultado, se obtendrá un conjunto de caminos totales y una relación de enlaces que definirán un array con el mejor tiempo posible y la mínima anchura para dicho tiempo.

4.8 SELECCION DE ALTURA ÓPTIMA

La altura de los arrays adyacentes delimita la del array en diseño. Por tanto, a veces será necesaria una nueva optimización, esta vez en altura.

El GAO obtiene distintos caminos totales, pero con tiempos y anchuras idénticos. Para seleccionar los arrays más óptimos se calcula la altura y se escojen aquellos caminos totales con altura mínima.

- **Cálculo de la altura del camino total**

En primer lugar el algoritmo ALG3 calcula la altura de cada camino total o ACT sin tener en cuenta los enlaces entre las células, mediante la siguiente fórmula:

$$ACT = HM_P + HM_N + 3$$

$$HM_P = Max\ [\ F1\ (H_P),\ F2\ (H_P) \ldots\ Fm\ (H_P)]$$

$$HM_N = Max\ [\ F1\ (H_N),\ F2\ (H_N) \ldots\ Fm\ (H_N)]$$

Los valores HM_P y HM_N corresponden a la altura máxima de las redes P y N respectivamente, para todas las células de un camino total y la constante tres para la altura de las regiones C, VDD y GND.

Siguiendo con el ejemplo, la altura del Camino Total de la permutación PN, CT(PN) es:

HM_P = Max [F1(3), F2(3), F3(3), F4(1)] = 3

HM_N = Max [F1(2), F2(2), F3(2), F4(2)] = 2

Por lo tanto ACT ⟶ 3 + 2 + 3 = 8

- **Cálculo de la altura del array**

Las regiones U y L son empleadas para los enlaces o conexiones metálicas entre las células de un array. Calculando la altura de dichas regiones y sumándolas a ACT se obtiene la altura total del array (ATA) para una permutación. Sólamente los enlaces tipo B y C representan una conexión entre células. Por tanto, se tomarán sólo los enlaces pertenecientes a dichos tipos. Así, para el grafo de la figura 4-7 tenemos los enlaces siguientes:

TIPO B	TIPO C
eE2	h13→
fE4	i23→
	j34→

Como los enlaces tipo B establecen una unión entre la función asociada a la entrada y la función destino (indicada en el propio enlace), puede establecerse los siguientes ajustes:

eE2 ——→ F1(e) ——→ e12

fE4 ——→ F3(f) ——→ f34

TIPO B		TIPO C	
e12	F1(e), F2(e)	h13→	F1(s),F3(h)
		i23→	F2(s),F3(i)
f34	F3(f), F4(f)	j34→	F3(s), F4(j)

Partiendo de los enlaces resultantes se relacionan las entradas de las funciones y los enlaces de conexión. De esta forma, un enlace eXY supone una conexión entre las células FX y FY a través del enlace de nombre 'e'. Este enlace eXY se simboliza como FX(e), FY(e). Por tanto, se establece la conexión mediante dos puntos definidos por el número de función y el nombre del enlace.

Las expresiones F1(s), F2(s) y F3(s) indican las salidas de las funciones F1, F2 y F3.

Para establecer la región (U, L) más óptima en la que situar un enlace, se construye un gráfico en el que solamente se representan las entradas de las células referenciadas por los enlaces y las salidas de las células, todas ellas (entradas y salidas) ubicadas en el mismo orden en que se encuentran en el camino total.

Entradas referenciadas por los enlaces de tipo B y C:

F1(e,s)

F2(e,s)

F3(h, i, f, s)

F4(f, j)

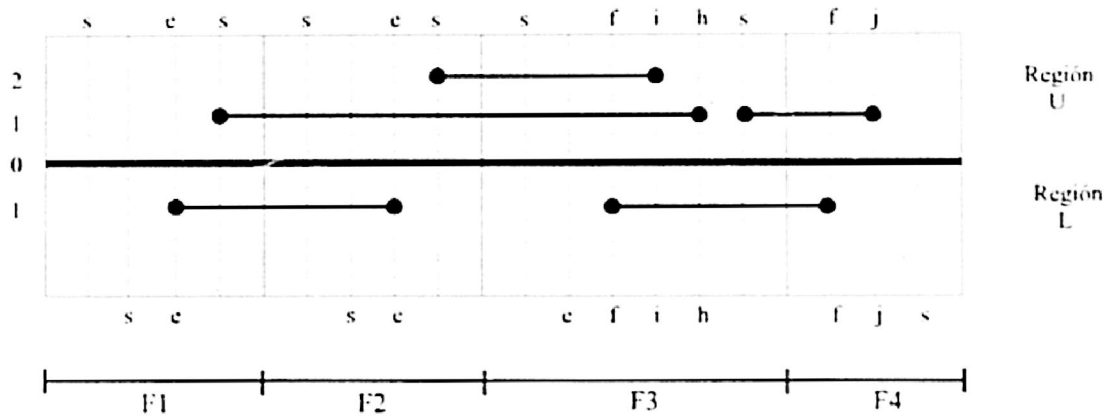


Figura 4-9 Gráfico para la conexión entre células.

En el gráfico se sitúan primero los enlaces de tipo C ($i23^{\rightarrow}$, $h13^{\rightarrow}$, $j34^{\rightarrow}$), puesto que tienen una ubicación definida.

De ésta forma, el enlace $h13^{\rightarrow}$ indica que la conexión óptima se establece por la región U, tomando la salida más a la derecha de la función F1 y uniéndola con la entrada h de la función F3. Una vez indicados dichos enlaces se colocan los de tipo B (e12, f34), que podrán estar situados indistintamente en la región U o L (la longitud de la conexión

siempre será la misma), para ello se localiza aquella región en la que su implementación no suponga un incremento de la altura de la región donde se ubique. El enlace f34 no incrementa la altura en la región L, sin embargo, sí lo hará en la región U. Si la ubicación de los enlaces de tipo B no supone ninguna diferencia en las regiones U o L entonces se elegirá una de ellas arbitrariamente.

La permutación en análisis del ejemplo desarrollado, genera una altura de magnitud dos en la región U y de uno en la región L, lo que supone una altura añadida de tres al valor ACT, resultando la Altura Total del Array (ATA) igual a once. El cálculo del parámetro ATA se debe realizar para todos los caminos totales óptimos obtenidos a partir del algoritmo GAO, seleccionando posteriormente aquel o aquellos que den como resultado la menor altura total del array.

CCT(1, 1) = 1647.211	- Tiempo total del array
ATA(1, 1) = 11	- Altura total del array
ACA(1, 1) = 29	- Anchura total del array

En consecuencia, con los algoritmos desarrollados conseguimos arrays lineales con máxima velocidad, menor anchura para dicha velocidad y menor altura para dicha anchura.

5. CONCLUSIONES

En la exposición de este trabajo de investigación ha quedado claramente demostrado que con los algoritmos aportados (GAO y ALG3) se obtienen los caminos totales más óptimos en velocidad y entre éstos, los de menor área posible para la realización de un array lineal. A diferencia de investigaciones anteriores en este campo, nuestro método no es heurístico, es decir, obtenemos siempre la mejor optimización, incluso para arrays complejos y/o con un número alto de variables por célula. Además, debido al procedimiento empleado en el análisis de los grafos, el tiempo de proceso de los algoritmos desarrollados es significativamente menor que los obtenidos con anterioridad. El algoritmo GAO, con el cual logramos la optimización de velocidad y anchura de arrays lineales, más el ALG3 para la selección de altura óptima estan implementados en lenguaje "C", por mayor transportabilidad (En el apéndice C se incluyen las rutinas relevantes de dichos algoritmos). Una vez demostrada teóricamente la optimización marcada como objetivo de esta tésis se ha procedido a la ratificación empírica. Para ello, se han evaluado nuestros algoritmos frente a arrays de especificaciones muy diversas. La contrastación de estos resultados con los mismos generados a partir de algoritmos de otros autores (Tabla 1-2) permite observar que sólo cuatro de ellos (Maziasz [MAZI91], Mailhot y DeMicheli [MAIL88], Ong Li y Lo [ONGL89], Tu [TU93]) consiguen unos resultados aproximados a los nuestros, desde el punto de vista de la optimización del área/velocidad. En el apéndice A se presenta un conjunto de arrays y los resultados obtenidos para ellos, con los algoritmos propuestos

por dichos autores y los expuestos en esta tésis. Al analizar los resultados obtenidos, se comprueba que respecto a la velocidad, los algoritmos de Maziasz, Mailhot y Ong, optimizan de forma deficiente y sólo el algoritmo diseñado por Tu [TU93] se puede considerar eficaz, aunque en presencia de ciertas células irregulares (por ejemplo F1 en apendice A-A3) la optimización que realiza no es total. En la optimización de la anchura se aprecia bastante similitud en los resultados. Sin embargo sólo Mailhot consigue, una total optimización de anchura pero no la velocidad lograda por nuestro método (Apéndice A-A4).

Cabe destacar que las comparaciones planteadas son empíricas, ya que al ser heurísticos los métodos desarrollados por los diversos autores, puede darse el caso de que para un array dado, los resultados generados por dichos métodos coincidan casualmente con los correspondientes a una optimización total de la velocidad, el área o ambas.

5.1 FUTURAS LÍNEAS DE INVESTIGACIÓN

Nuestro algoritmo ha logrado unos resultados totalmente satisfactorios, en muchos casos mejores incluso que los esperados de las pruebas realizadas sobre el diseño del mismo. Sin embargo, somos conscientes de las limitadciones que hemos impuesto para desarrollar y materializar el algoritmo con cierta facilidad. A partir de este punto el siguiente objetivo se puede centrar en eliminar dichas limitaciones, y por otro lado, extender esta metodología de diseño de arrays, al de un C.I. completo (Floorplanning).

Con nuestro algoritmo modelizaremos tanto los transistores como sus conexiones, no considerando ni distintos tamaños ni tipos de material en ninguno de los casos. Eliminar esta limitación no supondría una gran complejidad, ya que en síntesis implicaría la inclusión de clasificaciones adicionales para los enlaces en función de estas variables y la modificación del cálculo de las anchuras y tiempos para las células y conexiones.

Otra consideración interesante es la creación de arrays a la medida. Si tenemos en cuenta que los arrays configuran finalmente un C.I., podemos encontrar casos en los cuales el array óptimo no es el de mejor Velocidad-Anchura-Altura (orden de optimización en nuestro algoritmo), sino que el orden de optimización debería ajustarse a otros parámetros.

Una posible ampliación del método consistiría en incluir la optimización del consumo de corriente por ser éste el tercer parámetro junto con el área y velocidad que mejor define las prestaciones de un C.I.

Por otro lado, de entre los diversos tipos de células existentes (serie-paralelo, dinámicas, etc.) para la construcción de arrays, nuestro método trata las de tipo serie-paralelo por tener unas características medias que favorecen un mayor rendimiento. Nuestros algoritmos se podrían generalizar fácilmente, para la optimización de otros tipos de células, puesto que las abordadas son las de mas compleja optimización debido al empleo de dos redes duales, mientras que en otros casos (por ejemplo dinámicas), se usa una sola red o implican una menor rigidez en el diseño.

Finalmente también sería interesante extender nuestra teoría para el desarrollo optimizado de arrays no lineales (matriciales) en los que las células están dispuestas tanto en horizontal como en vertical.

5.2 APLICACIONES PRACTICAS

Los compiladores de silicio actuales carecen en gran medida de "inteligencia" para la síntesis y optimización del diseño del layouts en la fabricación de C.I. necesitando en muchos casos la intervención del diseñador para lograr resultados óptimos. La implementación de los algoritmos expuestos en esta tesis permitirá una menor intervención y por consiguiente un menor tiempo de diseño, obteniendo además unos resultados con un grado de optimización difícilmente alcanzable de forma manual por el diseñador. Otro aspecto importante en el diseño de C.I. es la simulación previa del layout; una buena simulación da como resultado una importante reducción del tiempo dedicado a la posterior corrección de errores así como un aumento de la fiabilidad del circuito. El estudio de las respuestas que realizan nuestros algoritmos es perfectamente aplicable a la simulación, con sólo sustituir las constantes modelizadas por los valores reales de capacidad, resistencia y dimensionado utilizados en el diseño de cada C.I. en particular .

BIBLIOGRAFÍA

[AT&T90] AT&T Corporation, ***CMOS Cell Library: Standard Cells and Functional Blocks***, 1990.

[BARY89] R. Bar-Yehuda y otros, **"Depth-first-search and dynamic programming algorithms for efficient CMOS cell generation"**, *IEEE Transactions on Computer Aided Design*, vol. CAD-8, Julio 1989, pp. 737-743.

[BRAD90] M.L. Brady y M.Sarrafzadeh, **"Stretching a knock-knee layout for multilayer wiring"**, *IEEE Transactions on Computers*, vol. C-39, Enero 1990, pp. 148-151.

[BRAY84] R.K. Brayton, G.D. Hachtel, C. McMullen y A. Sangiovanni-Vincentelli, ***Logic Minimization Algorithms for VLSI Synthesis***, The Kluwer Academic Publishers Series in Engineering and Computer Science 2, Norwell, MA, 1984.

[CHEN89] C.C. Chen y S. Chow, **"The layout synthesizer: an automatic netlist-to-layout system"**, *Proceedings 26th Design Automation Conference*, 1989, pp. 232-238.

[CHEN89] H.Y. Chen y S.M. Kang, **"Performance driven cell generator for dynamic CMOS circuit"**, *Proceedings, IEEE International Symposium on Circuits and Systems*, 1989, pp. 1883-1886.

[CHEN92] I.Y. Chen, ***Synthesis of VLSI Sea-of-Wires Arrays***, Tesis Doctoral, Universidad de Arizona, 1992.

[CHUE72] L. Y. Chuen-Tao, ***Aplicaciones Prácticas del P.E.R.T. y C.P.M.*** , Ediciones Deusto, 1972.

[DOMI89] A. Domic y otros, **"CLEO: a CMOS layout generator"**, *Proceedings, IEEE International Conference on Computer Aided Design*, Noviembre 1989, pp. 340-343.

[GAJS85] D.D. Gajski, **"Silicon compilation"**, *VLSI Systems Design*, Noviembre 1985, pp. 48-64.

[HILL88] Dwight Hill, Don Shugard, John Fisburn y Kurt Keutzer, ***Algorithms and Techniques for VLSI Layout Synthesis***, The Kluwer Academic Publishers Series in Engineering and Computer Science 65, Norwell, MA, 1988.

[HO92] P.J. Ho, ***Aspects of CMOS VLSI Complex Multiplier Design***, Licenciatura, Universidad de California, 1992.

[HSIE90] Y. Hsieh, C. Hwang, Y. Li y Y. Hsu, **"LIB: a cell layout generator"**, *Proceedings 27th. Design Automation Conference,*, 1990, pp. 474-479.

[HUAN89a] S. Huang y O. Wing, **"Gate matrix partitioning"**, *IEEE Transactions on CAD*, vol. 8, nº 7, Julio 1989, pp. 756-767.

[HUAN89b] S. Huang y O. Wing, **"Improved gate matrix layout"**, *IEEE Transactions on CAD*, vol. 8, nº 8, Agosto 1989, pp. 875-889.

[HUCH90] Y.H. Hu y S.J. Chen, **"GM Plan: A gate matrix layout algorithms based on artificial intelligence planning techniques"**, *IEEE Transactions on CAD*, vol. 9, nº 8, Agosto 1990, pp. 836-845.

[HWAN89] C. Hwang, Y. Hsieh, Y. Lin y Y. Hsu, **"An optimal transistor-chaining algorithm for CMOS cell layout"**, *Proceedings, International Conference on Computer Aided Design*, Noviembre 1989, pp. 344-347.

[JUST90] K. Just y otros, **"Palace: a layout generator for SCVS logic blocks"**, *Proceedings 27th Design Automation Conference*, 1990, pp. 468-473.

[KIM92] S. Kim, ***CMOS VLSI Layout Synthesis for Circuit Performance***, Tesis Doctoral, Universidad de Pensilvania, 1992.

[KWON88] Y. Kwon y C. Kyung, **"A fast heuristic for optimal CMOS functional cell layout generation"**, Proceedings, IEEE International Symposium of Circuits and Systems, 1988, pp. 2423-2426.

[LEFE89] M. Lefebvre y C. Chan, **"Optimal ordering of gate signals in CMOS complex gates"**, *IEEE Custom Integrated Circuits Conference*, 1989, pp. 17.5.1-17.5.4.

[LEFE90] M. Lefebvre, C. Chan y G. Martin, **"Transistor placement and interconnect algorithms for leaf cell synthesis"**, *IEEE European Design Automation Conference*, 1990, pp. 119-123.

[LENG90] T. Lengaüer, ***Combinatioral Algorithms for Integrated Circuit Layout***, Chichester, Wiley, 1990.

[LIND89] I. Lin, D. Du y S. Yen, **"Gate matrix layout synthesis with two-dimensional folding"**, *Proceedings 26th. Design Automation Conference*, 1989, pp. 37-42.

[MADS89] J. Madsen, **"A new approach to optimal cell sysnthesis"**, *Proceedings, International Conference on Computer Aided Design*, 1989, pp. 336-339.

[MAIL88] F. Mailhot y G. DeMicheli, **"Automatic layout and optimization of static CMOS cells"**, *Proceedings, International Conference on Computer Design*, Octubre 1988, pp. 180-185.

[MAZI90] R.L. Maziasz y J.P. Hayes, **"Layout optimization of static CMOS functional cells"**, *IEEE Transactions on Computer Aided Design*, vol. CAD-9, Julio 1990, pp. 708-719.

[MAZI91] R.L. Maziasz, ***Exact layout area minimization of static CMOS cells***, Tesis Doctoral, Universidad de Michigan, 1991.

[MAZI92] R.L. Maziasz y J.P. Hayes, ***Layout minimization of CMOS cells***, The Kluwer Academic Publishers Series in Engineering and Computer Science 160, Norwell, MA, 1992.

[NAIR85] R. Nair, A. Bruss y J. Reif, **"Linear time algorithms for optimal CMOS layout"**, *VLSI: algorithms and architectures. Proceedings of the international workshop on parallel computing and VLSI*, (P. Bertolazzi y F. Lucas, eds.), 1985, North Holland Publication, pp. 327-340.

[ONGL89] C. Ong, J. Li y C. Lo, **"GENAC: an automatic cell synthesis tool"**, *Design automation. Proceedings of the 1989, 26th ACM/IEEE Conference*, Las Vegas, NV, Junio 25-29, 1989, pp. 239-244.

[POGG76] P. Poggioli, ***Aplicación Práctica del Método PERT*** , Editores Técnicos Asociados S.A., Barcelona, 1976.

[POIR89] C.J. Poirier, **"Excellerator: custom CMOS leaf cell layout generator"**, *IEEE Transactions on Computer Aided Design*, vol. CAD-8, Julio 1989, pp. 744-755.

[ROME79] C. Romero, ***Técnicas de Programación y Control de Proyectos*** , Ediciones Pirámide S.A., Madrid, 1979.

[SAUC89] G. Saucier y otros, **"A channelless layout for multinivel synthesis with compiled cells"**, *Proceedings, International Conference on Computer Design*, Octubre 1989, pp. 35-38.

[SUN89] P.K. Sun, **"CETUS: a versatile custom cell synthesizer"**, 1989, pp. 348-351.

[TU93] H.T. Tu, ***New Approaches for VLSI Layout Compaction***, Tesis Doctoral, Universidad de Purdue, 1993.

[UEHA81] T. Uehara y W.N. VanCleemput, **"Optimal layout of CMOS functional arrays"**, *IEEE Transactions on Computers*, C-30, 5 (Mayo 1981), pp. 305-312.

[UYEM92] J.P. Uyemura, ***Circuit design for CMOS VLSI***, The Kluwer Academic Publishers, Norwell, MA, 1992.

[WEIN67] A. Weinberger, ***"Large scale integration of MOS complex logic"***, *IEEE Journal of Solid-State Circuits*, vol. SC-2, pp. 182-190, Diciembre, 1967.

[WEST92] N.H.E. Weste y K. Eshraghian, ***Principles of CMOS VLSI Design: A Systems Perspective (2ª edición)***, Addison-Wesley VLSI Systems Series, 1992.

[WIME87] S. Wimer y otros, **"Optimal chaining of CMOS transistor in a functional cell"**, *IEEE Transactions on Computer Aided Design*, vol. CAD-6, Septiembre 1987, pp. 795-801.

[WING82] O. Wing, **"Automated gate-matrix layout"**, *Proceedings, IEEE International Symposium on Circuits and Systems*, 1982, Roma, Italia, pp. 681-685.

[WING83] O. Wing, **"Interval-graph based circuit layout"**, *Proceedings, IEEE International Conference on Computer Aided Design*, Santa Clara, California, 1983, pp. 84-85.

[WOLF94] W. Wolf, ***Moderm ULSI Design: a Systems Approach***, Prentice Hall International Editions, 1994.

[YANG78] E.S. Yang, ***Fundamentals of Semiconductor Devices***, McGraw-Hill, 1978.