ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

ESPECIALIZACION DE ELECTRONICA
Y
TELECOMUNICACIONES

DISEÑO DE CIRCUITOS INTEGRADOS DE APLICACION ESPECIFICA (ASICs) DIGITALES CON TECNOLOGIA CMOS

VOLUMEN II
HERRAMIENTAS PARA DISEÑO DE ASICs

IVAN BERNAL CARRILLO
FREDY LEMUS CRIOLLO

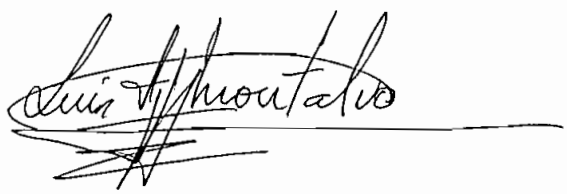
TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN LA ESPECIALIZACION DE INGENIERIA ELECTRONICA DE LA ESCUELA POLITECNICA NACIONAL

SEPTIEMBRE 1992

Certifico que bajo mi dirección la presente tesis fue realizada en su totalidad por los señores:

Iván Bernal Carrillo

Fredy Lemus Criollo



Ing. Luis Montalvo Ramírez

DIRECTOR

CONTENIDO GENERAL

VOLUMEN I. FUNDAMENTOS TEORICOS DE DISEÑO VLSI (*VERY LARGE SCALE INTEGRATION*)

VOLUMEN III. CODIFICADOR/DEDOCIFICADOR DE LINEA PROGRAMABLE *HDBn* COMO CASO DE ESTUDIO.

ANEXOS

INDICE DEL VOLUMEN II

CAPITULO 5

HERRAMIENTAS DE DISEÑO VLSI

La especificación y el diseño de un circuito integrado (CI) son procesos complejos, la verificación y validación de tales diseños constituyen retos aún mayores.

> "La incógnita básica de Moore acerca de qué puede ser hecho con VLSI se relaciona con un aspecto fudamental del diseño: cómo manejar la complejidad de los CIs VLSI"(1)

La clave del éxito de la tecnología VLSI descansa en el desarrollo de herramientas de diseño, simulación y verificación poderosas que ayudan al diseñador a producir un circuito integrado.

Todas las metodologías de diseño tratan de establecer caminos dentro y entre los diferentes niveles de diseño, incorporando todas las herramientas CAD posibles, con miras a alcanzar los objetivos siguientes:

---

(1) "Introduction to nMOS and CMOS VLSI Systems Design", A. Mukkherjee, pág. 278.

a) Reducir la complejidad y el tiempo de concepción de CIs, mejorando a la vez sus prestaciones.

b) Asegurar la implantación de diseños funcional y topológicamente correctos.

c) Garantizar la posibilidad de simular y verificar los diseños, considerando que el test de los circuitos crece en importancia a medida que estos incrementan su complejidad.

d) Optimizar la superficie ocupada por el diseño físico, a fin de aumentar el rendimiento de producción y reducir sus costos.

e) Desarrollar metodologías de diseño cuya dependencia del proceso tecnológico sólo afecte a los niveles más bajos de descripción de los CIs.

5.1 TIPOS DE HERRAMIENTAS DE DISEÑO

Al margen del estilo de diseño escogido, existe una cantidad de herramientas estándar necesarias para asegurar el éxito de una implantación VLSI. En esta sección se resumen las herramientas que típicamente se requieren para su realización, independientemente de la tecnología de diseño.

La ausencia de algún tipo particular de estas herramientas no conlleva necesariamente a un fracaso en el proceso de diseño, aunque sí complica su realización y aumenta los riesgos de error.

5.1.1 Herramientas de descripción del diseño

Los diseños se formulan mediante herramientas basadas en lenguages de descripción y/o captura esquemática.

a) Lenguages de descripción

Permiten la descripción e incluso el análisis, simulación y síntesis de los diseños. Las características básicas de estos lenguages las podemos resumir en los puntos siguientes:

i) Permiten descripciones mixtas de distintos niveles de abstracción: funcional, estructural y lógico.

ii) Cuentan con toda la riqueza sintáctica y semántica de los lenguages de programación de alto nivel, incorporan además tipos especiales de datos adecuados al tipo de descripciones que realizan.

iii) Algunos de ellos permiten la modelación parametrizable de los distintos elementos del diseño (incluso del tiempo).

iv) Dentro de la modelación, estos mismos lenguages permiten describir vectores de test y directivas de simulación y síntesis.

Al igual que los lenguages de programación, se requiere de un compilador que realice los análisis sintáctico/semánticos de cada descripción y cree las estructuras de datos correspondientes para procesar el sistema descrito de forma que pueda ser explotado por los procesos posteriores de simulación y síntesis.

b) Captura esquemática

Por otra parte los lenguages de descripción se complementan, para la formulación del diseño, con herramientas de captura gráfica de esquemas basadas en el uso de bibliotecas de componentes, celdas y generadores automáticos de módulos.

Además de toda una serie de facilidades para la edición gráfica, la captura esquemática debe ser capaz de:

i) Incorporar bibliotecas de componentes y de celdas que puedan introducirse en un diseño, y facilitar su actualización y mantenimiento.

ii) Verificar la conectividad correcta del esquema.

iii) Generar las netlists adecuadas para las siguientes etapas de diseño.

iv) Interactuar con procesos de simulación y síntesis, no sólo para pasarles información, sino también para registrar sobre el mismo esquema tales resultados.

5.1.2 Herramientas de síntesis

La síntesis es un proceso de refinamiento progresivo que traduce descripciones de alto nivel de abstracción en otras de bajo nivel. Se pueden distinguir tres niveles de síntesis:

a) **Síntesis estructural:** a partir de una descripción funcional se obtiene una descripción a nivel estructural en términos de bloques e interconecciones.

b) **Síntesis lógica:** a partir de los distintos bloques estructurales se obtiene una descripción a nivel de compuertas y registros.

c) **Síntesis física:** a partir de la descripción a nivel de compuertas y registros se obtiene la descripción geométrica del circuito o bloque funcional basándose en generadores, o bibliotecas de componentes (p. ej. celdas). A este nivel de síntesis también se lo llama mapeo tecnológico (*technologic mapping*) puesto que sus resultados se dirigen hacia una tecnología concreta.

Para evaluar los procesos de síntesis se deben considerar las siguientes características:

a) Lenguages de descripción que se utilizan en cada caso, y sus relaciones con los estándar.

b) Capacidad de simulación después de cada proceso de síntesis, y posibilidades de comparación de los resultados de síntesis/simulación con las descripciones y/o simulaciones de las etapas anteriores.

c) Grado de flexibilidad en cuanto a los tipos y características de las bibliotecas a emplear para la síntesis física.

d) Mecanismos de registro disponibles para reflejar las particularidades de una implantación física en las descripciones de nivel estructural/funcional superiores.

e) Posibilidades de documentación de cada una de las fases.

5.1.3 Editores de layout

El objetivo último de las herramientas de diseño VLSI es obtener una descripción de la disposición geométrica del layout del circuito diseñado, que permita generar los archivos PG (*Pattern Generators*) necesarios para la implan-

tación de las diferentes capas en el proceso de fundición de los CIs

Generalmente se usa para estos fines el formato CIF (*Caltech Intermediate Form*), adoptado por un grupo significativo de investigadores y casas fundidoras. Ello no implica que el diseñador deba usar directamente el formato CIF en sus descripciones, sino que existen los ***editores de layout*** consistentes en interfaces gráficas "CADs" que permiten manejar los diseños en forma gráfica encargándose el programa de la generación del formato CIF o similares.

Existen además, editores de layout capaces de vigilar el cumplimiento de las reglas de diseño y/o reglas de conectividad en el momento mismo que el diseñador trabaja.

Algunos editores de layout permiten también la preparación del plano de base; es decir, el posicionamiento de módulos funcionales y la formulación de estrategias de comunicación global y enrutamiento.

5.1.4 Extractores circuitales

El extractor circuital es un programa que, a partir de la descripción del layout de un CI (p. ej. el archivos CIF) examina las interrelaciones existentes entre sus máscaras e infiere la existencia de transistores y otros componentes,

junto con sus nodos de enlace, produciendo como salida la descripción de la red eléctrica equivalente ó *NETLIST*.

Se han formulado varios algoritmos para realizar la extracción del circuito equivalente de un layout, la metodología común a todos ellos es la siguiente:

a) Se identifican y caracterizan las zonas eléctricamente comunes (nodos) dentro y entre las diferentes máscaras que forman el layout.

b) Se identifican los transistores a través de la búsqueda de la superposición simultánea de capas de polysilicon, difusión y zona activa.

c) Se caracterizan los transistores mediante la estimación de parámetros tales como: relación largo/ancho y profundidad del canal, resistencia, capacitancia de los conductores de interconección, etc....

d) En aplicaciones en que se requiere alcanzar rendimientos exigentes del circuito, los extractores están preparados para la obtención de los elementos parásitos.

No es objetivo de los extractores de circuito, la verificación de reglas (geométricas o de conectividad), ni de funcionalidad del circuito eléctrico que generan.

La información de salida de estos extractores es usada por programas de verificación, sean estos de verificación funcional (simulación) o de reglas de conectividad.

5.1.5 Verificadores de layout

La verificación de layout consiste en el chequeo de un conjunto de reglas a las que el diseño debe ajustarse para ser geométrica y funcionalmente correcto. Existen básicamente dos tipos de verificación:

a) Verificación de reglas de conectividad

Las reglas de conectividad tienen que ver con la funcionalidad eléctrica del diseño (independientemente de su formulación geométrica) y dependen de la tecnología en que éste haya sido implantado.

Los verificadores de reglas de conectividad analizan características tales como: número máximo de transistores en serie, en paralelo, carga asociada a los nodos de salida, existencia de nodos sueltos, o de transistores colgados o sin polarización, etc...

b) Verificación de reglas geométricas (DRC)

Como se señaló anteriormente, existen un conjunto de reglas de diseño dentro de las que el fabricante garantiza

que aún considerando las variaciones inherentes al paso litográfico de generación de máscaras, el circuito obtenido será en alto porcentaje funcionalmente correcto. Por tanto, para que la fundición de un layout diseñado sea posible, este debe hallarse de acuerdo con las reglas del fabricante.

Los chequeadores de reglas de diseño ó DRCs (*Design Rule Checkers*) son piezas de hardware o software usadas para verificar que, independientemente de la funcionalidad lógica o eléctrica del circuito, todas las reglas geométricas de diseño de la tecnología seleccionada hayan sido cumplidas.

El DRC toma como entrada la descripción del layout del CI y basándose en un conjunto de reglas geométricas de diseño (p. ej. distancias mínimas entre capas, anchos mínimos, superposiciones y sobrelapamientos de capas permitidos, etc.) produce como salida una indicación de los errores cometidos.

5.1.6 Simuladores

Las herramientas de simulación son utilizadas para predecir el comportamiento y funcionalidad de los diseños de CIs. Este proceso es imprescindible antes de abordar cada etapa de síntesis entre los niveles de abstracción del diseño, por ello, los simuladores deben ser capaces de soportar los tipos de descripción, modelación y bibliotecas de cada nivel.

Como características importantes de los entornos de simulación se pueden citar:

a) El número de estados lógicos (0,1,X) y niveles de fuerza (forcing, HiZ, charge-storage) distintos, que pueden asociar a las señales en cada momento. Ello da una idea de la precisión con que pueden modelar la realidad.

b) Niveles de abstracción que son capaces de abordar simultáneamente.

c) Lenguages de descripción que aceptan los simuladores.

d) Posibilidades de simulación interactiva y modificación simultánea de esquemas/descripciones y estímulos durante la depuración de los diseños.

e) Facilidades para preparar la simulación y para analizar los resultados.

f) Capacidad para realizar análisis de tiempos.

Todas estas características deben traducirse en una interacción ágil y flexible entre las etapas de captura y simulación de un diseño, que permita una rápida pre-evaluación de las distintas alternativas de realización.

Dependiendo del grado de abstracción de las características del circuito diseñado y del tipo de información que se desee obtener de éste, se tienen distintos tipos de simuladores, así:

a) **Simuladores funcionales:** aquellos que se basan en sistemas especificados en términos de bloques funcionales con una función de transferencia entrada/salida definida.

b) **Simuladores lógicos o de nivel de compuerta:** aquellos que se refieren a módulos funcionales o modelos primitivos tales como compuertas NOT, AND, OR, NAND y NOR; consistentes en abstracciones del comportamiento lógico circuital de cada módulo obtenidas a partir de su comportamiento eléctrico.

c) **Simuladores circuitales:** aquellos que originan informaciones detalladas de los niveles de señal y retrasos en cada nodo del circuito, se basan en modelos matemáticos de sus elementos. Estos simuladores se caracterizan por su alta precisión, aunque requieren de gran cantidad de memoria y tiempo de procesamiento.

> "El tiempo de procesamiento es típicamente proporcional a n^m donde n es el número de dispositivos no lineales en el circuito y m es un valor entre el rango de 1 a 2. Este tipo de programas se emplean para verificar en detalle pequeños circuitos o para corroborar resultados de simulaciones más eficientes aunque menos precisas. *Es irrealista usar estos programas para la verificación de chips VLSI grandes*"(2)

---

(2) "Principles of CMOS VLSI Design, A Systems Perspective", N. Weste - K. Eshraghian, pág. 255.

d) **Simuladores de nivel de conmutación:** consisten en una simplificación de los simuladores circuitales en que el valor estático de cada nodo es interpretado como un valor lógico (1,0 ó X) que es renovado únicamente cuando las entradas a los transistores son perturbadas, lo que se traduce en un aumento de su velocidad de simulación y de su capacidad para simular circuitos grandes, aunque, en contraste no pueden predecir la temporización y velocidad característicos del circuito. Su ventaja sobre los simuladores lógicos radica en que existe una correspondencia directa entre el layout del diseño y su funcionalidad lógica sin necesidad de encasillar cada configuración dentro de una estructura (NAND, NOR, etc..) definida.

5.1.7 Verificadores de temporización

Los verificadores de temporización proveen al diseñador de información sobre los retrasos de las señales a través de los caminos del circuito. Realizan un análisis estadístico del sentido de flujo de las señales en todos los transistores que combinado con un conjunto de reglas dependientes de la tecnología de diseño, les permiten evaluar los retrasos que serán críticos en la operación real del circuito.

El uso de un verificador de temporización combinado con el de un simulador de conmutación provee al diseñador de información global sobre la funcionalidad del circuito, lo

que permite restringir la simulación circuital detallada únicamente a sectores críticos.

5.1.8 Herramientas de enrutamiento (interconexión)

Una de las etapas más importantes en la generación del layout de un circuito es la búsqueda del camino óptimo para realizar la interconexión que permita llevar las señales entre sus componentes, lo que se conoce como ***enrutamiento*** (*routing*).

El problema de enrutamiento de señales se formula mediante la especificación de:

a) un conjunto de módulos funcionales rectangulares,
b) un conjunto de pines asociado a estos módulos, y
c) un conjunto de redes de señal que especifica cuáles pines deben ser eléctricamente interconectados, pudiendo ser estas redes: punto a punto o multipuntos.

El objetivo de toda herramienta de enrutamiento consiste en realizar todos los enlaces indicados entre los bloques tendiendo a las siguientes características:

a) Número mínimo de líneas enlace,
b) Longitud mínima de las líneas de enlace, a fin de minimizar la distorsión de las señales que enrutan.
c) Incidencia mínima en el área global ocupada por el CI.

5.2 SECUENCIAS DE DISEÑO PARA LA CONCEPCION DE ASICs

5.2.1 SECUENCIA CLASICA DE DISEÑO

En el Capítulo 1 se indicó que el proceso de concepción y diseño de un circuito integrado se halla dividido en tres niveles descendentes de abstracción, a saber:

a) Nivel funcional.
b) Nivel estructural o circuital.
c) Nivel físico.

Las herramientas a utilizarse en el diseño de un ASIC, abordan todos estos niveles de abstracción. Estas herramientas cambian según se trate de un CI full-custom, semi-custom, o dispositivo programable; sin embargo, la filosofía a aplicarse dentro de cada nivel y en los interfaces entre niveles es similar y conlleva al uso estructurado de las diferentes herramientas de diseño disponibles.

El esquema de la Fig.5.1 describe la organización e interrelaciones entre las herramientas de diseño VLSI. Según este esquema la secuencia a seguirse para la concepción de un circuito integrado es la siguiente:

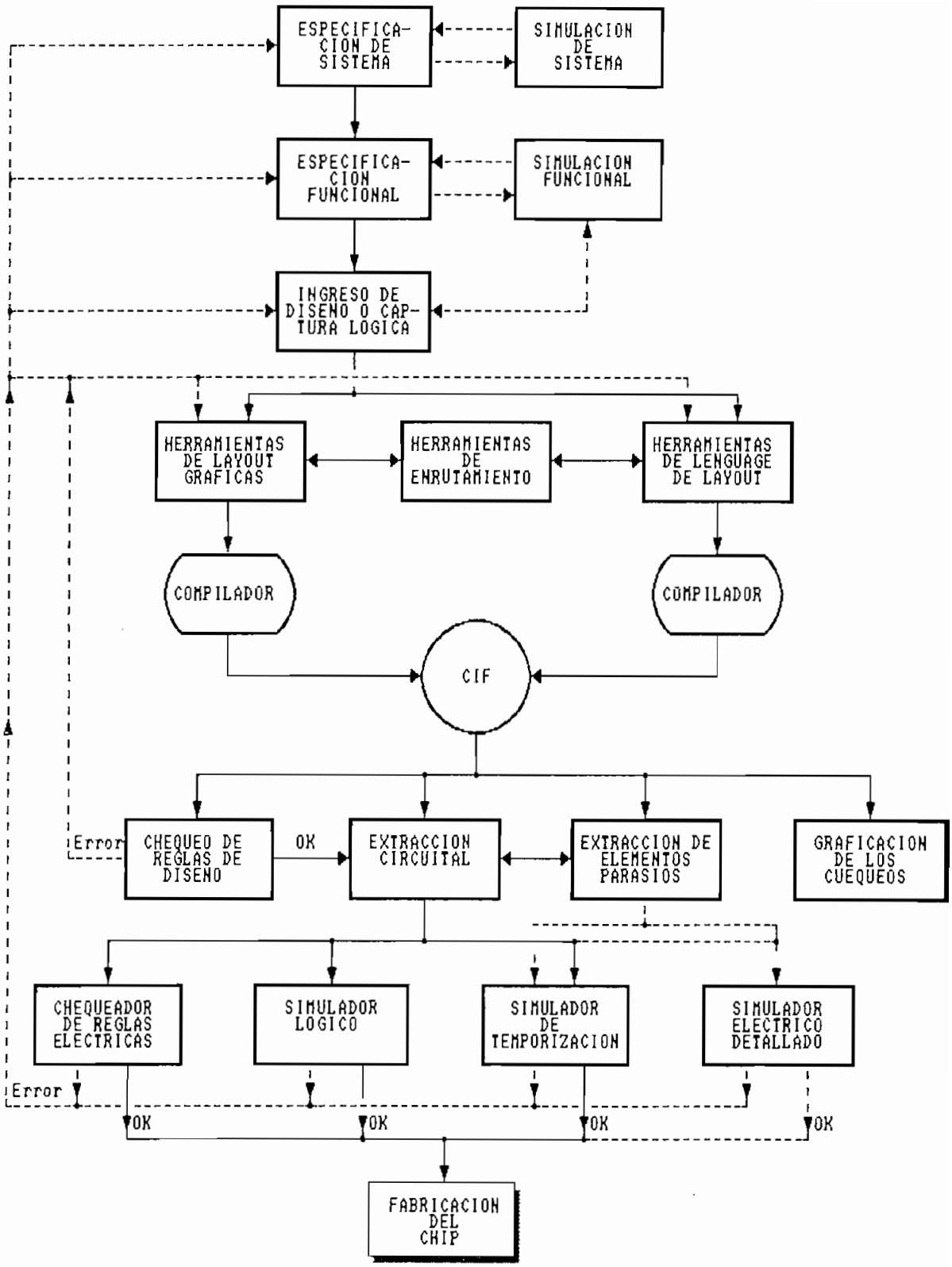


FIGURA 5.1 Secuencia de diseño de un CI.

a) El diseño se inicia mediante la especificación de sistema y su correspondiente simulación que mas bien tiene que ver con el ambiente global del cual el ASIC formará parte antes que con el CI mismo.

b) A continuación se realiza la formulación de alto nivel del circuito expresada mediante la especificación funcional del circuito, y se ingresa la descripción a nivel lógico del mismo a través de las herramientas de captura del diseño, que podemos considerar formada por:

   i) un esquema lógico en forma gráfica o de texto (*netlist*), y

   ii) un conjunto de vectores de test desarrollados con el doble objetivo de: preparar la simulación funcional, y realizar la cobertura de fallas para el test de los chips resultantes del proceso tecnológico.

c) Con los datos anteriores se aborda la simulación funcional y la simulación de fallas a fin de, paralelamente y mediante un proceso iterativo, depurar el diseño y completar los vectores de test.

d) Luego se realiza la distribución de bloques funcionales mediante herramientas de posicionamiento, encargadas de evaluar distintas opciones de layout, y que determinarán los pasos a seguirse en los procesos de ubicación y

conexionado (routing) de celdas y módulos. Son características importantes de estos procesos:

i) la optimización de superficie obtenida,

ii) el grado de parametrización (asignación de restricciones y/o prioridades, optimizaciones locales, etc..) y versatilidad que permita la herramienta al diseñador, y

iii) la capacidad de adaptación del diseño a distintas tecnologías de fabricación de CIs.

e) La información anterior es compilada mediante las herramientas de síntesis que, como resultado, generan el layout definitivo del circuito, consistente en su descripción geométrica (CIF o similares), con lo que se ha llegado al nivel físico.

f) Una vez obtenido el layout del CI, mediante las herramientas de verificación, se realiza su análisis geométrico-topológico a fin de verificar el cumplimiento de las reglas de diseño de la tecnología seleccionada. Del layout se evalúan además otras características tales como capacidades y resistencias de las líneas de conexión y retardos en las señales que se deriven del conexionado físico.

g) Luego, los extractores circuitales generan las descripciones circuitales (netlists) equivalentes del layout en

base a las que se verifica la coherencia del diseño respecto a las descripciones de las etapas anteriores.

h) A continuación se realiza la segunda simulación, mucho más precisa que la anterior puesto que incluye capacidades y retardos derivados del coneccionado y extraídos directamente del layout.

i) Un proceso de registro permitirá reubicar los datos sobre el layout, el esquema y/o la netlist, con el fin de reflejar las correcciones que se realizan y repetir las simulaciones y análisis de tiempos, tantas cuantas veces sea necesario, hasta llegar a cumplir con las especificaciones tanto funcionales como geométricas requeridas por el usuario y por el fabricante para el circuito diseñado.

j) Tras superar los análisis y simulaciones post-layout, antes de pasar a la fabricación, se debe escoger un encapsulado y realizar la asignación de *pines* (en el dado de silicio) y *pads* (en la cápsula). El *PAD* consiste en un cuadro de metal conectado hacia el interior del circuito y accesible desde el exterior, donde se soldará el hilo de conexión con el pin del encapsulado.

En muchos casos los usuarios externos a la firma fabricante finalizan su trabajo cuando los resultados de la primera simulación son correctos y los vectores de test dan una

buena cobertura de fallas (> 95% ó 98%), dejando el resto de etapas para el centro de diseño del fabricante.

5.2.2 Tendencias actuales de diseño

A pesar de que el esquema de la Fig.5.1 muestra de manera unificada las etapas de diseño para las distintas alternativas de ASIC, existen diferencias entre estas etapas que aumentan considerablemente conforme se desciende de nivel, y que introducen restricciones determinantes en el nivel físico. Estas restricciones de bajo nivel han dominado el diseño clásico forzando al uso de metodologías condicionadas por los niveles bajos de abstracción ***(metodologías bottom-up)***, que obligan al diseñador a decidir *a priori* las alternativas a utilizar en la implantación de su ASIC.

Actualmente la evolución de las metodologías de diseño de ASICs sigue dos caminos mutuamente complementarios.

a) Durante la segunda mitad de los años 80, junto con las herramientas CAD, están siendo desarrollados entornos integrados de diseño donde conviven de forma unificada diversas herramientas que comparten una base de datos común y un único interfaz con el usuario (Fig.5.2).

> "Actualmente están siendo desarrollados soportes de diseño estandarizados con el objetivo de que, al garantizarse la compatibilidad entre herramientas de diferentes aplicaciones, éstas puedan insertarse en soportes comunes. La iniciativa del soporte unificado es un

esfuerzo alentado por la industria a fin de estandarizar los interfaces entre herramientas CAD VLSI" (3)

En este campo, y a pesar de que ya existen algunos de estos entornos, se esperan importantes innovaciones durante los próximos años basadas, sobre todo, en los últimos estándares informáticos.

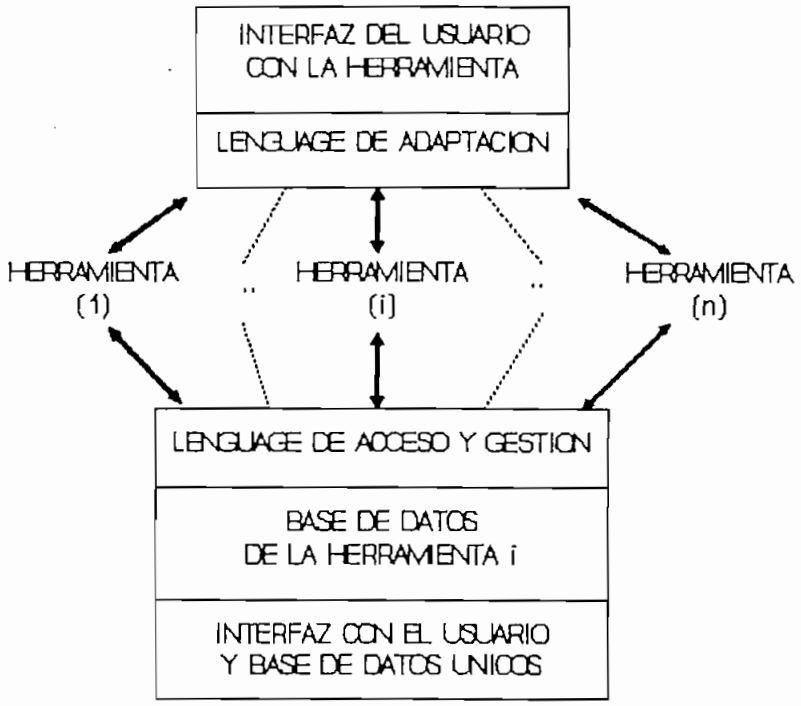


FIGURA 5.2 Entorno integrado de diseño de CIs.

b) Para poder abordar el nivel funcional, poco formalizado y utilizado hasta ahora, se están desarrollando lenguages y herramientas que permitan realizar y manipular descripciones de circuitos a ese nivel y enlazarlas con las etapas a los otros niveles (esquemático y físico).

---

(3) "Databases and Cell-Selection Algorithms for VLSI Cell Libraries", Computer Magazine IEEE, Febrero 1990, Simon Foo - Yoshiyasu Takefuji.

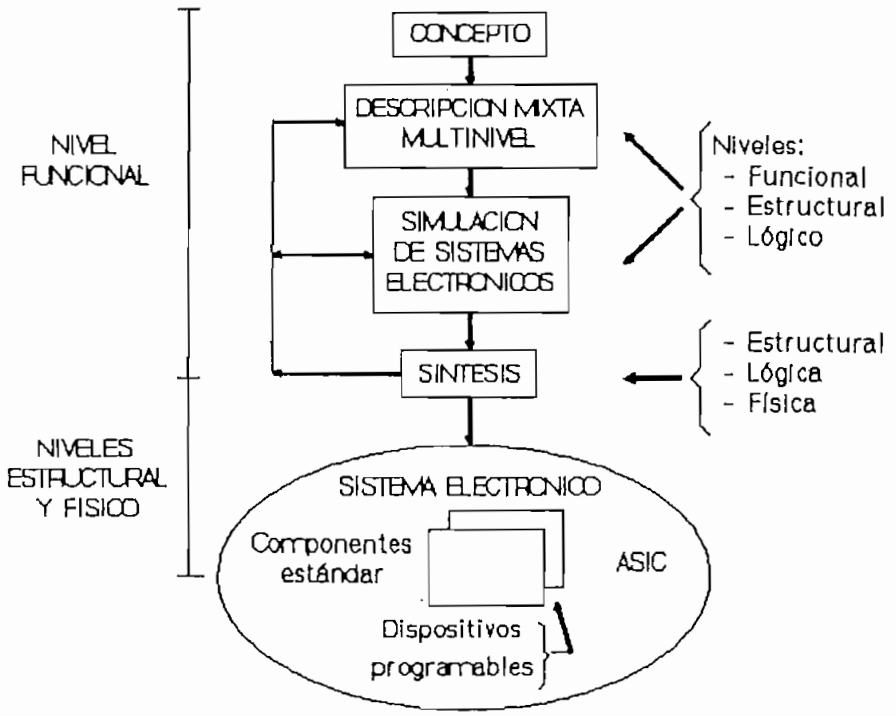


FIGURA 5.3 Esquema de las nuevas metodologías de diseño electrónico.

La Fig.5.3 muestra un esquema de estas nuevas etapas para el diseño de ASICs a nivel funcional. A partir de descripciones en lenguages de alto nivel se pueden realizar simulaciones mixtas (funcional, estructural, lógica) e implantaciones a estos niveles, evitando al máximo los condicionamientos que las descripciones de bajo nivel imponen.

"Los diseñadores de CIs VLSI necesitan de una herramienta computacional de diseño que encapsule diferentes niveles de las fases de diseño, desde arreglos geométricos de compuestos de bajo nivel, pasando por diseños esquemáticos y lógicos de nivel medio hasta manipulaciones simbólicas de alto nivel. Para diseñar rápidamente sistemas VLSI tal herramienta debe operar sobre un nivel de arreglo simbólico que evite los detalles de niveles menores"(4)

También se están desarrollando distintas alternativas de síntesis que, a partir de las descripciones funcionales, sean capaces de conducir, de forma automática o asistida por computador, hacia la implantación de distintos tipos de ASICs. De esta forma se facilitan metodologías de diseño *top-down* que, dirigidas desde niveles altos de abstracción, permiten evaluar distintas alternativas de diseño antes de abordar la implantación física de un circuito.

La confluencia de estas dos líneas de evolución debe permitir, en un futuro no lejano, la aparición de entornos de concepción de sistemas electrónicos que, no solo integren las distintas herramientas de CAD sino además contemplen las diversas alternativas y metodologías de diseño.

(4) "A structured Approach for VLSI Circuit Design", Kent Smith - Jun Gu.

5.3 PRINCIPIOS DE SIMULACION ELECTRICA Y LOGICA

Los programas de simulación de circuitos han sustituido paulatinamente a los procesos de test en base a implantaciones con componentes discretos, como forma de validar la corrección del diseño de un CI. De hecho, los efectos físicos, que son despreciables en los circuitos discretos comunes, se tornan importantes en los microcircuitos, al punto que tales diseños no pueden ser implantados a partir de componentes discretos en un laboratorio y dar resultados de test ajustados a la realidad, por tanto se debe, ya sea:

i) realizar una fabricación "de prueba" lo cual resulta extremadamente costoso y lento, peor aún si se sabe que su modificación u optimización resulta imposible, o

ii) simular cuidadosamente el circuito mediante un programa de computación.

En este acápite se delinean conceptos generales de simulación circuital tales como: la estructura de los archivos de simulación, la interpretación de los circuitos en forma de *netlists* y las directivas de simulación básicas.

Para el desarrollo de estos conceptos se ha escogido al simulador eléctrico SPICE (***Simulation Program with Integrated Circuit Emphasis***) que permite la simulación detallada de los circuitos eléctricos, y al simulador lógico NDL en el que se

hace una abstracción de nivel lógico del comportamiento eléctrico de los elementos del circuito.

5.3.1 Simulación Eléctrica "SPICE"

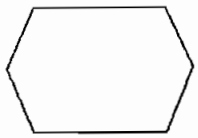
El programa SPICE desarrollado en la Universidad de California-Berkeley, es un simulador eléctrico en el que el comportamiento físico/eléctrico de los componentes del circuito se simula a partir de modelos matemáticos, lo que le permite proveer informaciones detalladas sobre el funcionamiento de los circuitos que simula.

Dada la variedad de las aplicaciones del simulador SPICE, en el desarrollo subsiguiente se han recopilado únicamente los elementos de la simulación SPICE aplicables al diseño *VLSI DIGITAL*, dejando de lado varias de sus opciones para otros popósitos.

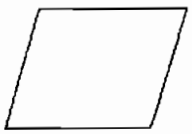
a) Simbología empleada

En la descripción estructural de los archivos en lenguage SPICE, se emplea la siguiente nomenclatura:

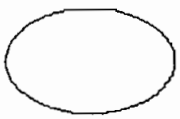
La estructura que contiene es de tipo general y es especificada en otra parte del análisis.



La estructura que contiene corresponde a un "**comando**" de simulación.



La estructura que contiene corresponde a la descripción de un "**dispositivo circuital**".



La secuencia de caracteres que contiene es propia del lenguage y debe ser empleada sin variación alguna. (Opcionalmente es factible cambiar los caracteres en mayúsculas que contiene por sus minúsculas)

Además, es conveniente notar que:

i) La sintaxis particular de cada "**dispositivo circuital**" en la simulación SPICE se detalla en el Anexo A.1.

ii) La sintaxis propia de cada "**comando**" de simulación SPICE se detalla en el Anexo A.2.

iii) La nomenclatura aquí descrita será utilizada también para la descripción de los lenguages empleados a lo largo del desarrollo del presente trabajo.

b) Observaciones generales

A continuación se indican estructuras sintácticas generales que deben ser consideradas en la elaboración de archivos para la simulación SPICE.

i) Para la descripción de los archivos se usan caracteres ASCII, por tanto éstos pueden ser generados en cualquier editor de texto.

ii) Los comandos y descripción de dispositivos se colocan uno por línea. De requerirse más de una línea para un mismo comando, estas se concatenan con un signo "+".

Ejemplo:

```
VPULSO 1 0 PULSE (-1mv 1mv 2ns 2ns 50ns 100ns)
```

Es equivalente a:

```
VPULSO 1 0 PULSE (-1mv 1mv 2ns 2ns
+ 50ns 100ns)
```

iii) Las líneas en blanco son ignoradas por el compilador.

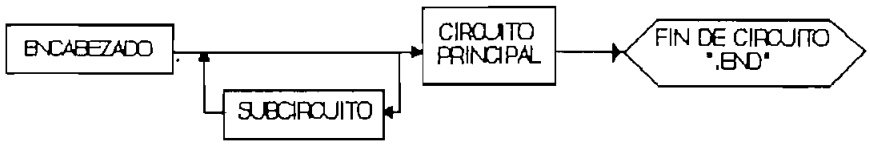
iv) Toda línea debe comenzar en la primera columna, es decir que no puede comenzar con espacios en blanco.

v) "*" Al inicio de una línea indica al compilador que esta línea es de comentario.

vi) ";" En cualquier lugar de la línea indica al compilador que el texto a su derecha es un comentario.

c) Estructura general

La estructura general del archivo de simulación SPICE es:

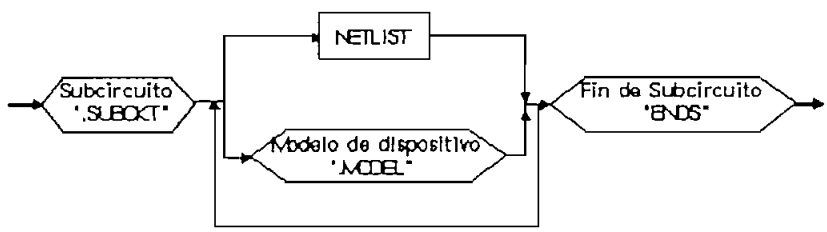


i) Encabezado

Corresponde a la primera línea del archivo, puede tener cualquier secuencia de caracteres ya que es ignorada por el compilador SPICE.

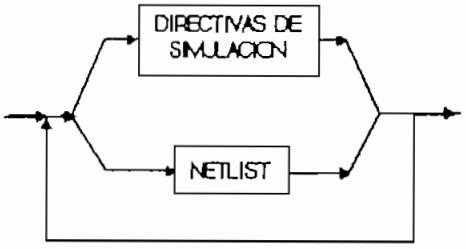
ii) Subcircuito

El lenguage SPICE tiene la facilidad de capturar una función circuital como "subred" de componentes que luego se adscriben a una red externa mayor. Se acostumbra emplear los subcircuitos para la definición de redes que son almacenadas en archivos de biblioteca. Su estructura es:



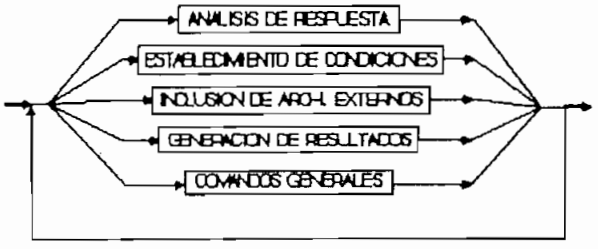
iii) Circuito principal

Posee la red eléctrica propia del circuito (NETLIST) y las directivas (comandos) para su simulación. Su estructura es:

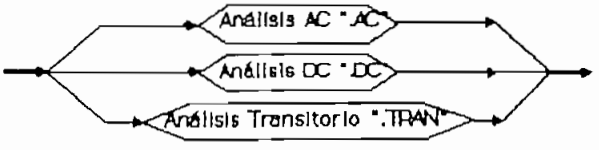


d) Directivas de simulación

Las directivas de simulación son comandos que indican al compilador SPICE los análisis circuitales que deberán ser realizados y cómo deberán ser ejecutados y reportados al usuario. La sintaxis y usos de cada directiva se detallan en el Anexo A.2. Las directivas empleadas en la simulación de circuitos VLSI digitales se agrupan en directivas de:



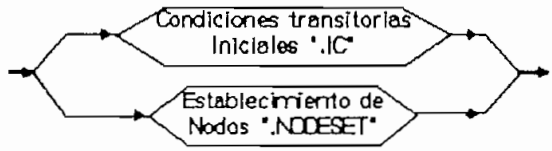
i) Análisis de respuesta



Permiten indicar al simulador la forma de variación de los parámetros de las fuentes de señal del circuito.

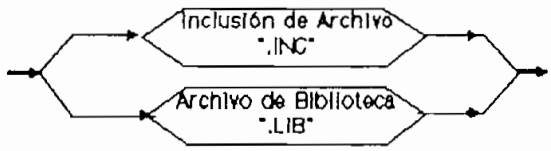
ii) Establecimiento de condiciones

Se emplean para indicar el estado eléctrico de los nodos al inicio, o durante todo el intervalo de tiempo de simulación.



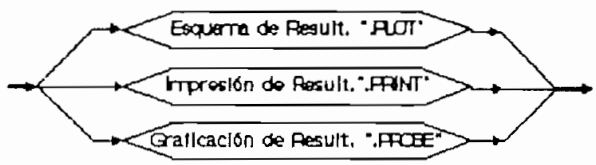
iii) Inclusión de archivos externos

Permiten incluir circuitos de biblioteca o secuencias de simulación preestablecidas.



iv) Generación de resultados

Se emplean para indicar el formato en que serán generados los resultados de la simulación.

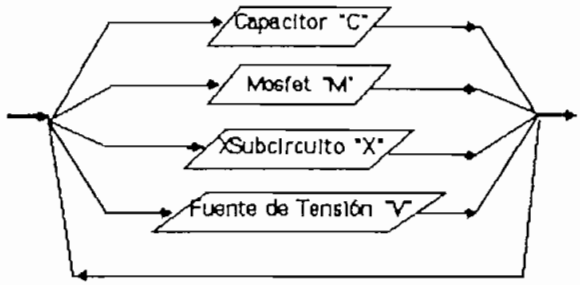


v) Comandos generales

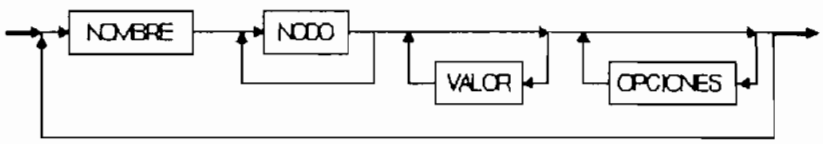
Permiten indicar los modelos físicos de los dispositivos y las opciones de control para los análisis que realiza el simulador.

e) Netlist

La NETLIST consiste en un conjunto de líneas empleadas para la descripción de la red circuital, cada línea contiene un elemento circuital ligado a la red. La estructura de la NETLIST empleada para la simulación de circuitos VLSI digitales es:

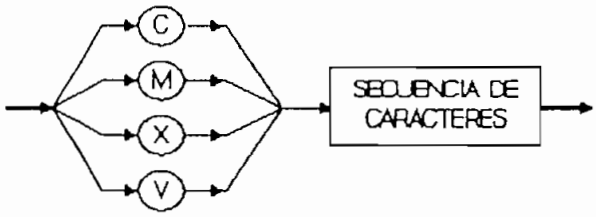


Cada elemento, según sus características tiene una estructura propia que se detalla en el Anexo A.1. No obstante, todos los elementos de la NETLIST siguen una estructura común cuyas implicaciones circuitales y eléctricas se detallan a continuación:



i) Nombre

El nombre es propio de cada elemento, su estructura es:



Del esquema anterior, se advierte que el primer caracter indica al compilador el tipo de disposivo circuital. La correspondencia entre este caracter y los tipos de elementos empleados en la simulación VLSI digital es:

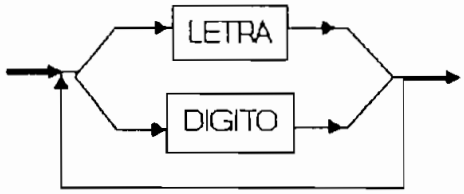
C :Condensador,

M :Mosfet,

X :Subcircuito, y

V :Fuente de tensión.

ii) Secuencia de caracteres

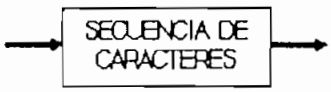


La secuencia de caracteres corresponde a un conjunto de letras y/o dígitos que puede ser tan larga como se desee.

SPICE no hace diferencia entre letras mayúsculas y minúsculas, es decir que "a" y "A" tienen el mismo valor sintáctico para el compilador.

iii) Nodo

Desde el punto de vista eléctrico, los nodos se pueden concebir como conductores comunes a dos o más elementos de un circuito. Su estructura sintáctica es:



La secuencia de caracteres tiene la misma estructura indicada en (ii). El primer caracter puede ser una letra o un dígito, dependiendo de la versión de SPICE que se emplee.

Además de su estructura sintáctica, existen ciertas reglas ELECTRICAS que deben ser respetadas en la descripción de los nodos:

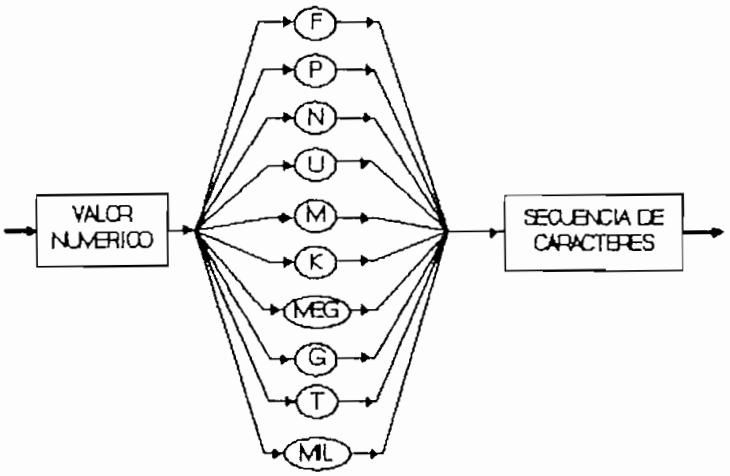
iii.1) Siempre debe existir un nodo referencial o de **tierra** que se recomienda identificarlo como "0".

iii.2) Todos los terminales de los elementos deben conectarse a otro terminal, por ende no se admiten "**nodos sueltos**".

iii.3) Todos los elementos circuitales tienen dos o más terminales o nodos, no pudiendo existir elementos de un solo terminal.

iv) Valor

Corresponde a la magnitud y unidad de medición que permiten cuantificar al dispositivo al que se asocia, su estructura es:



iv.1) El **valor numérico** es un número **decimal** o de **punto flotante.** Los valores de punto flotante escalan al número decimal por una potencia de diez, donde la letra "E" ó "D" separa al número decimal de su exponente de diez entero.

P.ej. 0.0045 puede ser escrito como 4.5E-3 ó 4.5D-3

iv.2) El caracter especial que sigue al valor numérico indica que este debe ser multiplicado por una potencia de diez (con una excepción), la equivalencia entre este caracter y los factores de multiplicación es:

F	femto	10^{-15}
P	pico	10^{-12}
N	nano	10^{-9}
U	micro	10^{-6}
M	milli	10^{-3}
K	kilo	10^{3}
MEG	mega	10^{6}
G	giga	10^{9}
T	tera	10^{12}
MIL	$25.4 \cdot 10^{-6}$	

iv.3) La secuencia de caracteres es ignorada por el compilador SPICE, no obstante, permite al usuario escribir los valores con sus unidades sin cambiar su significado. Es decir, se pueden escribir valores tales como: 10pF, 10picoseg, 10pamps, y todos significarán 10E-9 ya que las secuencias: F, icoseg y amps respectivamente, son ignoradas.

v) Opciones

Son características que permiten definir los parámetros físicos y eléctricos de cada dispositivo, son propias para cada elemento y se detallan en el Anexo A.1.

f) Ejemplo de aplicación

Construir la NETLIST, y realizar la simulación circuital de una celda AND de 2 entradas en base a las definiciones

subcircuitales de la celda NAND de 2 entradas "NAND2", y de un inversor "INV".

i) Construcción de la NETLIST

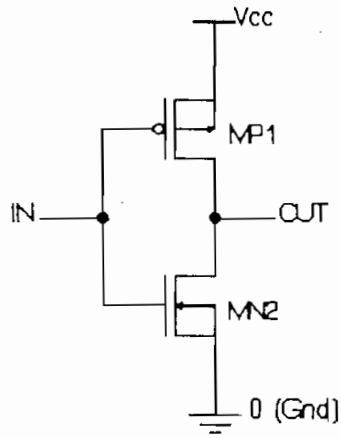


FIGURA 5.4 Esquema circuital del subcircuito "INV".

La Fig.5.4 indica el esquema circuital de la celda "INV" con su correspondiente estructura de nodos. La NETLIST para este circuito es la siguiente:

```
* CELDA: INVERSOR "INV"

.SUBCKT INV IN OUT VCC 0  ; Definición del subcircuito
MP1 OUT IN VCC VCC        ; MOSFET P1 (nodos: drenaje, compuerta, fuente, sustrato)
MN2 OUT IN  0   0         ; MOSFET N2 (nodos: drenaje, compuerta, fuente, sustrato)
.ENDS INV                 ; Fin de definición de subcircuito
```

El circuito correspondiente a la celda NAND de 2 entradas "NAND2", con su estructura de nodos se grafica en la Fig.5.5.

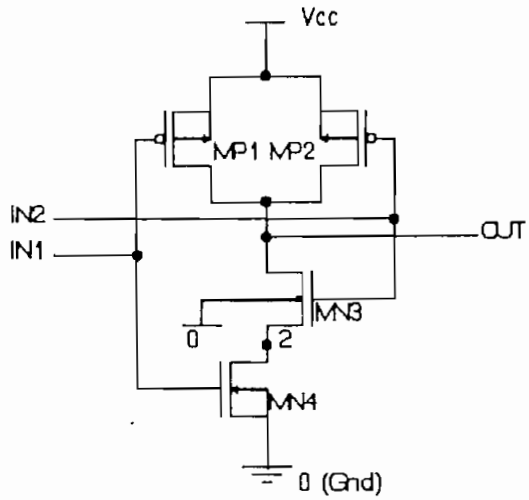


FIGURA 5.5 Esquema circuital del subcircuito "NAND2".

La NETLIST correspondiente al subcircuito "NAND2" de la Fig.5.5 es:

```
* CELDA NAND DE 2 ENTRADAS: "NAND2"

. SUBCKT NAND2 IN1 IN2 OUT VCC 0   ;Definición de subcircuito
MP1 OUT IN1 VCC VCC                ;MOSFET P1
MP2 OUT IN2 VCC VCC                ;MOSFET P2
MN3 OUT IN2  2   0                 ;MOSFET N3
MN4  2  IN1  0   0                 ;MOSFET N4
.ENDS NAND2
```

El circuito final correspondiente a la celda AND de 2 entradas "AND2" se lo constuye integrando los subcircuitos "NAND2" e "INV" anteriores, como se indica en la Fig.5.6. En este esquema se observa la naturaleza local de los nodos internos a cada subcircuito, se nota además la correspondencia entre los nodos de interfaz (declarados en la definición del subcircuito) y los nodos con los que cada bloque es integrado al circuito principal.

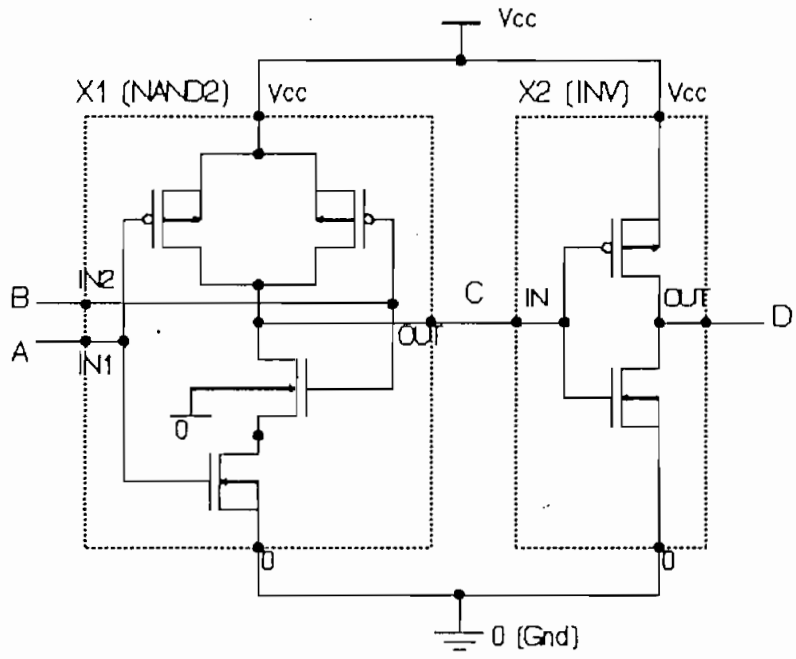


FIGURA 5.6 Esquema circuital del circuito "AND2" estructurado en base a las celdas "NAND2" e "INV".

La NETLIST de la celda "AND2" tiene por tanto dos macroelementos: X1, correspondiente al subcircuito "NAND2" y X2 correspondiente al subcircuito "INV" cuyos nodos de interconexión se grafican en la Fig.5.6. Así:

```
* CIRCUITO PRINCIPAL
* AND DE 2 ENTRADAS "AND2"

X1 A B C Vcc 0 NAND2      ;Llamada al subcircuito NAND2
X2 C D   Vcc 0 INV        ;Llamada al subcircuito INV
.END                      ;Fin de circuito principal
```

ii) Directivas de simulación

La NETLIST del circuito es suficiente para su caracterización si se desea realizar algún proceso de síntesis que permita obtener una descripción de este al nivel inferior (nivel físico). Sin embargo, es insuficiente para fines de

simulación ya que no se ha especificado la forma como esta se realizará.

Para la simulación de todo circuito, se deben intercalar en su NETLIST, los siguientes elementos y directivas de simulación.

ii.1) Fuentes de tensión.

- Fuente de polarización.
- Señales de entrada.

ii.2) Directivas de control.

- Análisis de respuesta.
- Generación de resultados.

En el caso del circuito AND2 del ejemplo de tiene:

ii.1) Fuentes de tensión

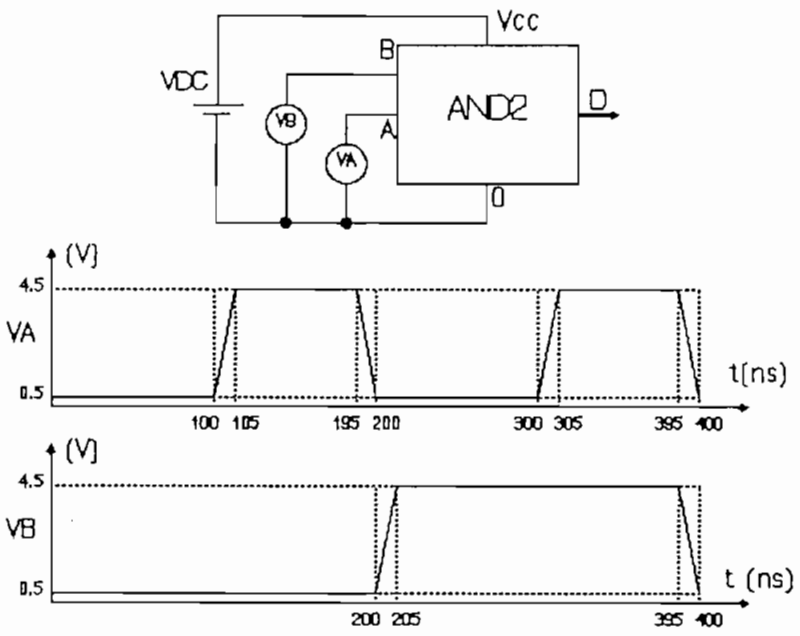


FIGURA 5.7 Fuentes de polarización y de señal para el circuito AND2.

Las fuentes de polarización y de señal que deben conectarse al circuito AND2, según su estructura de nodos se grafican en la Fig.5.7. Estas fuente en el lenguage SPICE, se especifican de la siguiente manera:

```
VDC Vcc 0  5.0 Voltios                                    ;Fuente de polarización

VA   A 0 PULSE(0.5V 4.5V 100NS 5NS 5NS 90NS 200NS)        ;Fuente de señal A
VB   B 0 PULSE(0.5V 4.5V 200NS 5NS 5NS 190NS 400NS)       ;Fuente de señal B
```

ii.2) Directivas de control:

En este tipo de simulación interesa analizar la respuesta "transitoria" del circuito frente a las variaciones de la señal de entrada, por lo que la directiva empleada es:

```
.TRAN 10ns 500ns ;Análisis de respuesta transitoria.
```

De la directiva planteada se tiene que se ha instruido al programa SPICE para que realice el análisis entre 0ns y 500ns con intervalos máximos de tiempo de 10ns.

Los resultados de la simulación pueden generarse de varias maneras, en el presente ejemplo se ha escogido la forma gráfica de exhibición de resultados por ser la que mejor permite visualizar la respuesta del circuito frente a sus señales de entrada, lo que se especifica de la forma:

```
.PROBE          ; Generación de resultados gráficos.
```

iii) Realización de la simulación

Reuniendo todos los elementos anteriores (con la adición de algunos comentarios), el archivo SPICE para la simulación del circuito AND de 2 entradas del ejemplo es el siguiente:

```
* CIRCUITO AND DE 2 ENTRADAS "AND2"
* ================================

* DEFINICION DE SUBCIRCUITOS

* CELDA: INVERSOR "INV"

.SUBCKT INV IN OUT VCC 0          ; Definición del subcircuito
MP1 OUT IN VCC VCC                ; MOSFET P1 (nodos: drenaje, compuerta, fuente, sustrato)
MN2 OUT IN  0   0                 ; MOSFET N2 (nodos: drenaje, compuerta, fuente, sustrato)
.ENDS INV                         ; Fin de definición de subcircuito

* CELDA NAND DE 2 ENTRADAS: "NAND2"

. SUBCKT NAND2 IN1 IN2 OUT VCC 0  ;Definición de subcircuito
MP1 OUT IN1 VCC VCC               ;MOSFET P1
MP2 OUT IN2 VCC VCC               ;MOSFET P2
MN3 OUT IN2  2   0                ;MOSFET N3
MN4  2  IN1  0   0                ;MOSFET N4
.ENDS NAND2

* CIRCUITO PRINCIPAL: AND DE 2 ENTRADAS "AND2"

X1 A B C Vcc 0 NAND2              ;Llamada al subcircuito AND2
X2 C D   Vcc 0 INV                ;Llamada al subcircuito INV

* FUENTES DE TENSION

VDC Vcc 0  5.0 Voltios                                  ;Fuente de polarización

VA   A 0 PULSE(0.5V 4.5V 100NS 5NS 5NS 90NS 200NS)      ;Fuente de señal A
VB   B 0 PULSE(0.5V 4.5V 200NS 5NS 5NS 190NS 400NS)     ;Fuente de señal B

* DIRECTIVAS DE SIMULACION

.TRAN 10ns 500ns                  ;Análisis de respuesta transitoria.
.PROBE                            ;Generación de resultados gráficos.

.END                              ;Fin de programa
```

Los resultados de esta simulación se generan en un archivo *PROBE.DAT* y requieren para su visualización (Fig.5.8) del programa *PROBE* (utilitario del SPICE).

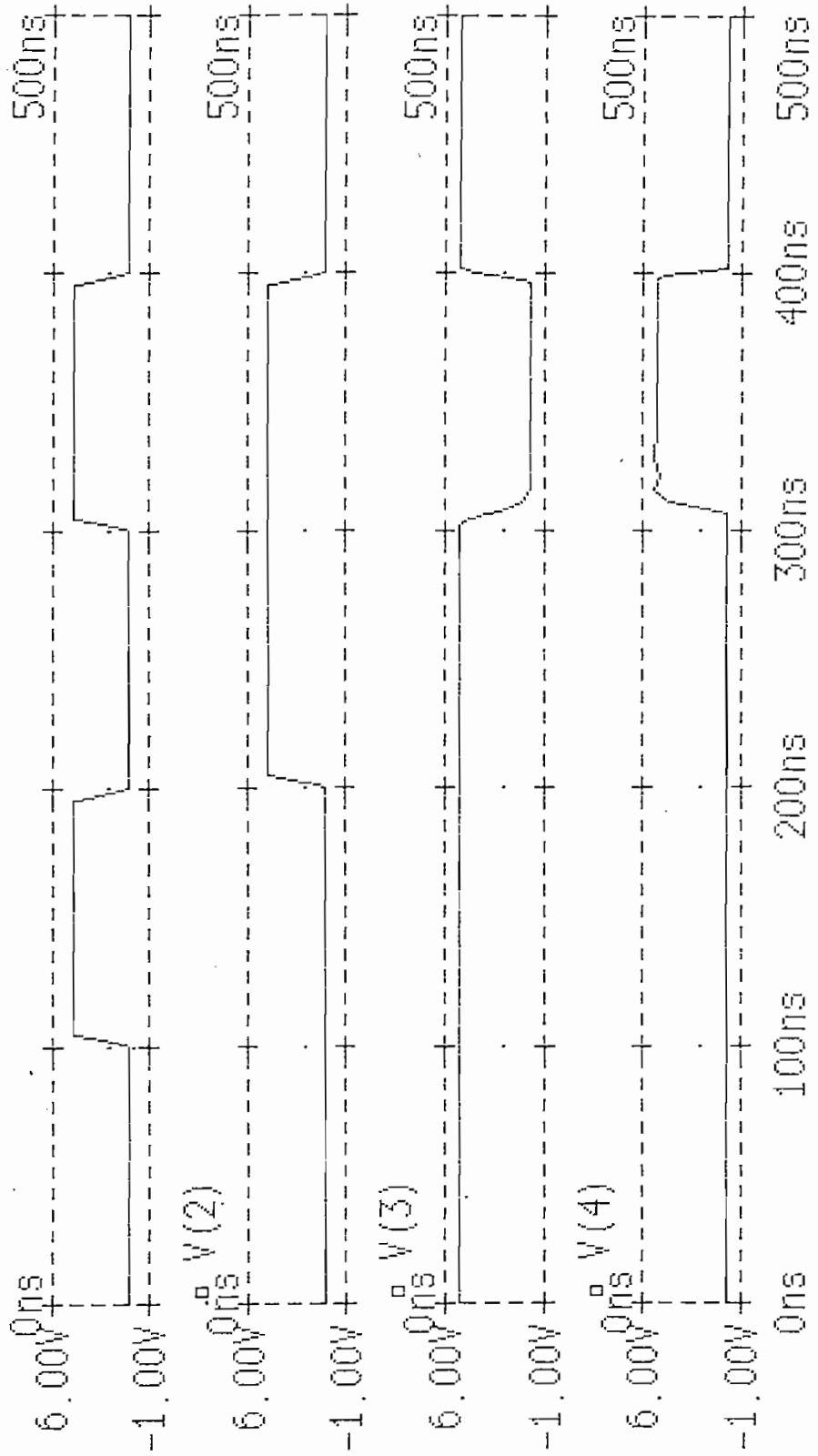


FIGURA 5.8 Visualización de los resultados de la simulación del circuito "AND2" generados en el programa PROBE.

5.3.2 Simulación lógica "NDL"(5)

La simulación eléctrica de los circuitos únicamente es factible para el caso de diseños pequeños debido al tiempo de simulación requerido. El diseño de sistemas mayores implica el uso de simuladores lógicos y/o de niveles de conmutación como es el caso del programa NDL. Estos programas son simplificaciones de simuladores circuitales como el SPICE, y por tanto, no proveen datos detallados del funcionamiento de los circuitos, no obstante, la información lógica que generan es valiosa para la depuración y análisis lógico/funcional de los circuitos.

a) Modelo de simulación

i) Algoritmo de simulación

El simulador NDL desarrollado en la UFRJ (Universidade Federal do Rio de Janeiro - Brazil), es del tipo "*unit-delay*" y se basa en el algoritmo desarrollado por R. Bryant. En este algoritmo se emplea para todos los transistores un retardo virtual igual a una unidad de tiempo. Las respuestas del simulador, en principio, son respuestas en estado estacionario, lo que implica que "**no es posible analizar la respuesta transitoria de las redes de transistores**"

(5) "TEDMOS IV: TURBO EDITOR PARA CIRCUITOS INTEGRADOS CMOS", E. Schmitz - J. Assis - J. Borges - ..., Cap 6.

El programa NDL únicamente permite la simulación de circuitos digitales en tecnologías:

i.1) **NMOS:** Tecnología de transistores MOS de canal N

i.2) **CMOS:** Tecnología de transistores MOS complementarios.

Para la simulación se emplea un modelo simplificado del transistor cuyo comportamiento puede corresponder a uno de los tres estados siguientes:

i.1) **Abierto:** impedancia infinita.

i.2) **Saturado:** impedancia cero.

i.3) **Indeterminado:** no se conoce si el transistor está abierto o saturado.

Los nodos, a su vez, pueden ser de tres tipos:

i.1) **De entrada:** cuando el nodo puede ser conectado a una fuente de tensión con capacidad de carga infinita (por ejemplo: compuerta de un transistor de entrada, drenaje de un transistor de paso).

i.2) **Pull-up:** cuando el nodo está conectado a un transistor de carga.

i.3) **Normal:** cuando el nodo está conectado a terminales de transistores tipo N o tipo P (NO transistores de carga)

Los nodos de *entrada*, independientemente de la tecnología de diseño son empleados por el diseñador para la conexión de fuentes de señal externas. En cambio los nodos de *pull-up*, y por ende los transistores de carga asociados a estos son propios de la tecnología NMOS como se indica en la Fig.5.9.

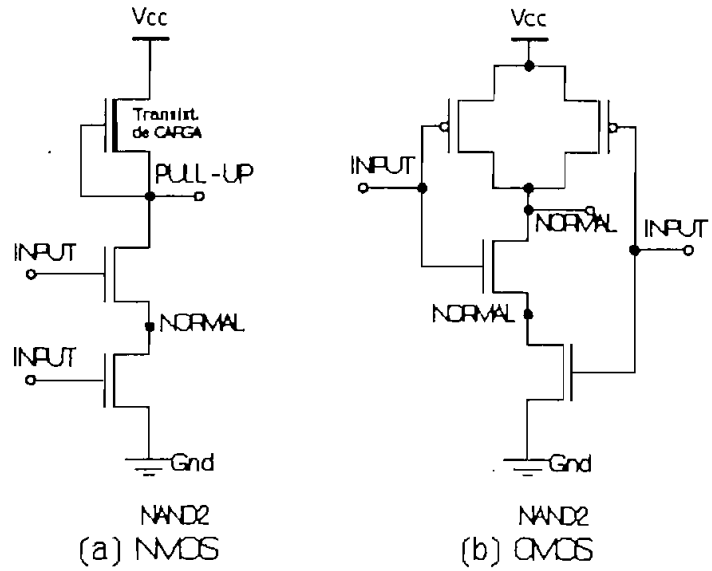


FIGURA 5.9 Caracterización de nodos en tecnologías (a) NMOS, (b) CMOS.

Independientemente de la tecnología de diseño, el programa asume que inicialmente todos los nodos del circuitos son *normales*, su caracterización como nodos de entrada o de pull-up es asignada por el diseñador en función de la tecnología seleccionada, en consecuencia, los nodos que éste omite conservan su caracterización original.

Para la simulación, antes de la aplicación de estímulos, el programa divide a los nodos circuitales cercanos en grupos. La simulación se procesa de grupo en grupo donde el estado de cada grupo es evaluado, y los grupos que son afectados por su cambio de estado se colocan en la lista de grupos para los que la simulación se ejecuta nuevamente, este proceso es iterativo hasta que no haya grupos con alteración lo que se conoce como **estado estable o de equilibrio** del circuito.

El algoritmo permite la simulación de circuitos con cientos (y hasta miles) de transistores en tiempos muy pequeños.

Existen circuitos con realimentaciones extrañas (aunque correctas) en que el simulador no encuentra el estado estable, de ser así, existe un número máximo de iteraciones luego de las que el simulador envia un mensaje de error, en este caso el diseñador debe proporcionar algún estado inicial o conectar algunos nodos del circuito a un estado fijo.

ii) **Pasos de simulación**

Son pasos diferenciales en los que se procesa la simulación, cada paso es una tentativa de llegar al estado estable del circuito y su realización es transparente para el usuario. La complejidad del circuito determina la cantidad de pasos necesarios para llegar al estado estable en el que se dice que el circuito **converge**.

El circuito **sincrónico** tiene su cadencia de ejecución comandada por un reloj de una o más *FASES*. Un conjunto completo de fases sin sobreposición es denominado *CICLO*. La secuencia de un ciclo de *n* fases es:

```
CICLO 1          ...          CICLO i

    FASE 1                        FASE 1
    FASE 2                        FASE 2
     ...                           ...
    FASE n                        FASE n
```

El estado del circuito puede ser exhibido al final de cada fase o de cada ciclo según las especificaciones del usuario.

En caso de que el circuito sea **asincrónico**, es decir que no sea sincronizado por un reloj externo, se requiere que las entradas sean definidas en forma de cadenas binarias. Se asume en este caso que el tiempo de duración de cada bit de la cadena puede ser de una FASE, o de un CICLO, dependiendo del comando dado al simulador.

b) Lenguage de simulación NDL

i) Observaciones generales.

A continuación se definen algunos aspectos generales a ser considerados en la elaboración de los archivos para la simulación lógica NDL

i.1) Los archivos se construyen en base a caracteres ASCII, por tanto, pueden ser generados en cualquier editor de texto.

i.2) El caracter "%" indica que el contenido a su derecha corresponde a un comentario, y por tanto es ignorado por el compilador.

i.3) En las descripciones subsiguientes se emplea la misma simbología definida para la simulación SPICE, en este

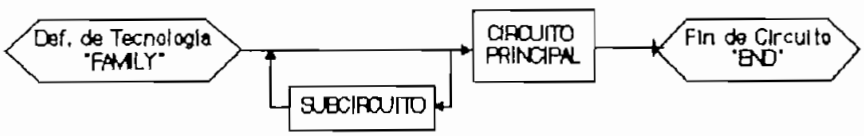
caso la sintaxis de los **elementos circuitales** se detalla en el Anexo B.1 y la de los **comandos de simulación** NDL en el Anexo B.2.

i.4) El programa NDL, como se indicó anteriormente, fue desarrollado en la UFRJ – Brasil, a ello se debe que su sintaxis emplee términos correspondientes a los idiomas **inglés** y **portugués.**

En el lenguage de simulación NDL, la descripción del circuito en base a los elementos circuitales se construye en un archivo *<circuito>.NDL* y los comandos de simulación se graban en otro archivo con extensión *.SIM* .

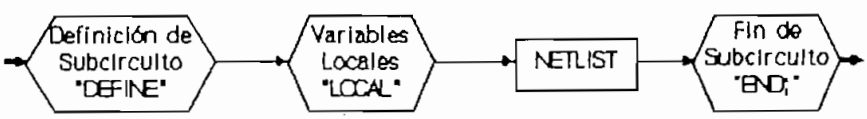
ii) Descripción del circuito (archivo .NDL)

La estructura general del archivo de descripción del circuito es:



ii.1) Subcircuito

Al igual que el SPICE, el lenguage NDL también permite definir subcircuitos (macros) que pueden ser conectados al circuito principal, su estructura es:

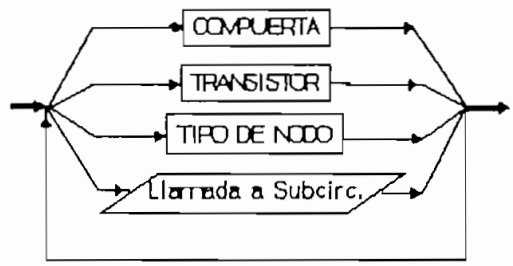


ii.2) Circuito principal

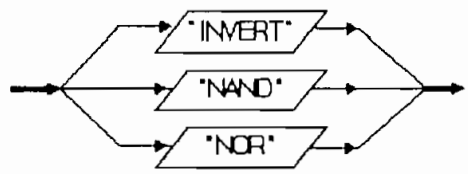
Posee la NETLIST del circuito y los comandos para su simulación. Su estructura es:

ii.3) Netlist

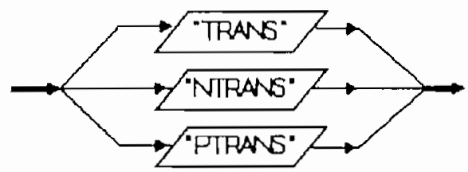
La NETLIST del lenguage NDL corresponde a la descripción de circuitos digitales basados en compuertas, transistores y subcircuitos (macros) cuya sintáxis se detalla en el Anexo B.1. La estructura general de la NETLIST es:



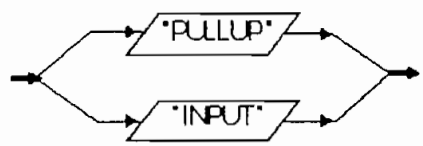
Compuertas



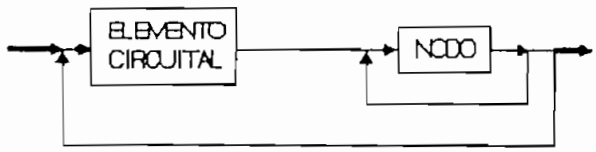
Transistores



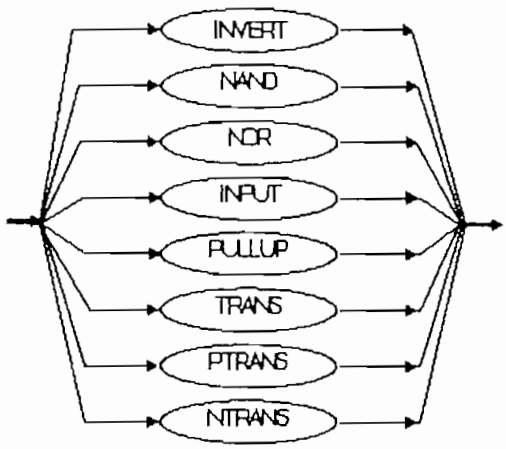
Tipos de nodos



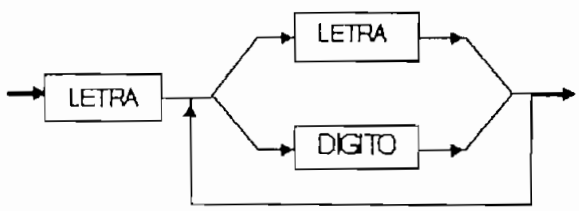
Al igual que en el lenguage SPICE, a pesar de que cada elemento circuital tiene su sintaxis propia, todos ellos tienen una estructura característica común que da lugar a la estructura de la NETLIST NDL siguiente:



Donde los elementos circuitales posibles se representan por las secuencias de caracteres:



Y los nodos tienen la estructura siguiente:

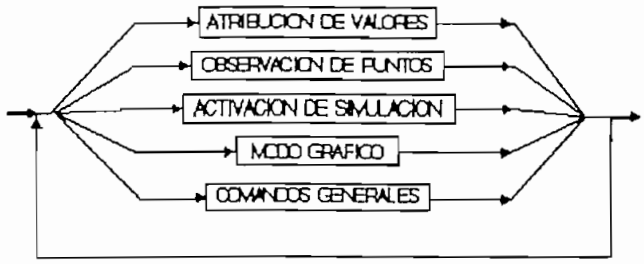


Las reglas eléctricas aplicables a los nodos son las mismas que se detallaron para el lenguage SPICE. De su estructura sintáctica se desprende que el primer caracter de

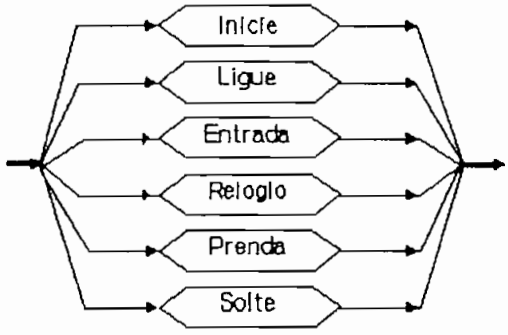
identificación del nodo debe ser una letra, pudiendo los caracteres restantes ser letras o dígitos.

iii) Comandos de simulación (archivo .SIM)

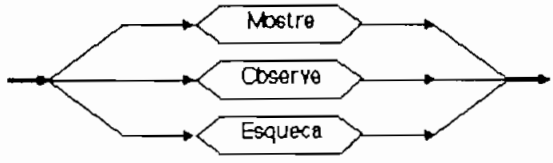
Permiten aplicar estímulos a los nodos del circuito, activar los procesos de simulación y exhibir al usuario los resultados de tales procesos. La sintaxis de cada comando se detalla en el Anexo B.2. Los comandos del lenguage NDL son:



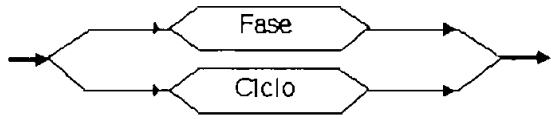
Comandos de atribución de valores



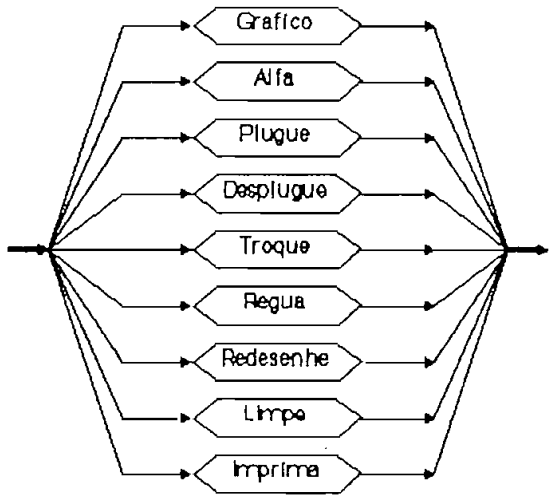
Comandos de observación de puntos del circuito



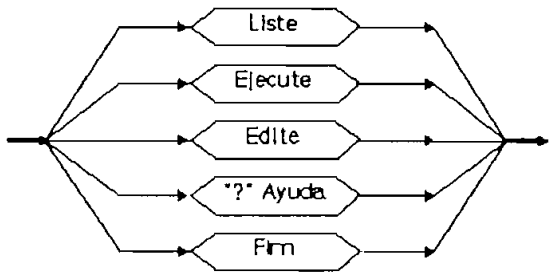
Comandos de activación del proceso de simulación



Comandos del modo gráfico



Comandos generales



c) Interfaz gráfico del simulador NDL

Los resultados de la simulación en el programa NDL pueden ser generados en forma de listados de los estados lógicos de los nodos (0, 1 ó X), sin embargo el simulador NDL, posee un interfaz en el que se exhiben sus resultados en forma gráfica, este interfaz (Fig.5.10) es de gran utilidad para el usuario, puesto que además es interactivo, ya que permite la generación de comandos de simulación conforme la simulación se realiza.

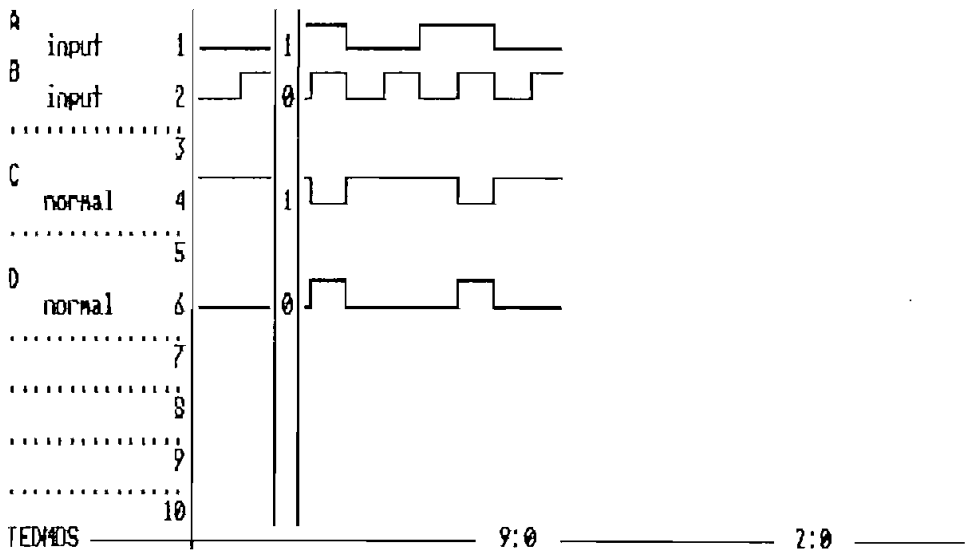
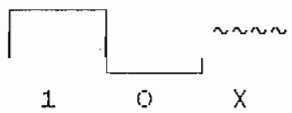


FIGURA 5.10 Interfaz gráfico del simulador NDL (Ejemplo de simulación).

A la izquierda de la ventana se indican los nombres de las señales a ser activadas en el osciloscopio que pueden ser del tipo: INPUT, PULLUP ó NORMAL.

A la derecha se muestra la secuencia de estados que las señales activas van tomando conforme la simulación avanza. Opcionalmente se puede generar una REGLA vertical (ver Anexo B.2 comando *Regua*) que puede ser movida sobre cada columna de puntos, la regla indica el estado lógico de la señal en aquellos puntos.

Los estados lógicos posibles de las señales son:



En la parte inferior de la ventana se tienen dos relojes de señalización cuyo formato es:

#ciclo:#fase

El primer reloj (al centro) indica el último ciclo/fase de simulación (con respecto al instante 0:0), el segundo reloj (a la derecha) únicamente aparece cuando se genera la regla vertical e indica el intervalo de tiempo de simulación (respecto al instante 0:0) sobre el que ésta se halla posicionada.

d) Ejemplo de aplicación

Para el circuito AND de 2 entradas "AND2" simulado eléctricamente con el programa SPICE, describir el circuito en formato NDL en base a las celdas NAND2 e INV y realizar su simulación lógica.

i) Descripción del circuito (archivo .NDL)

Para la descripción NDL de los subcircuitos NAND de 2 entradas "NAND2" e INVERSOR "INV" se han empleado los mismos esquemas circuitales (y su configuración de nodos) de las Figuras 5.4 y 5.5 empleados para la definición de la NETLIST SPICE. Fig.5.11.

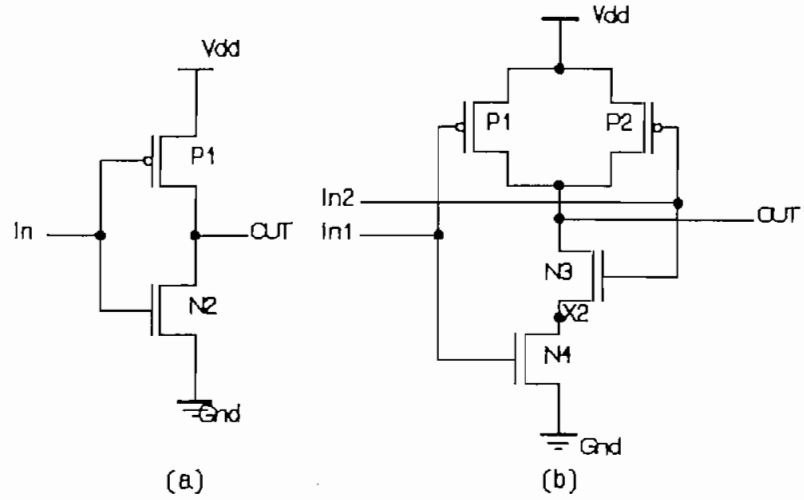


FIGURA 5.11 Diagrama circuital de los subcircuitos celdas INV(a) y NAND2(b).

La NETLIST correspondiente al subcircuito INV de la Fig.5.11(a) es:

```
% SUBCIRCUITO INVERSOR: "INV"

DEFINE INV In OUT;          % Definición del subcircuito INV
LOCAL;                      % No existen nodos locales internos

PTRANS  In   Vdd  Out;      % MOSFET P1 (nodos: compuerta, fuente, drenaje)
NTRANS  In   Gnd  Out;      % MOSFET N2 (nodos: compuerta, fuente, drenaje)
END;                        % Fin de definición de subcircuito
```

En este caso no existen nodos locales internos puesto que:

i.1) Los nodos In y Out son interpretados como nodos locales de interfaz por el compilador NDL.

i.2) Los nodos Vdd y Gnd son **nodos globales** para todos los subcircuitos.

La NETLIST para el subcircuito NAND2 de la Fig.5.11(b) es la siguiente:

```
% SUBCIRCUITO NAND 2 ENTRADAS: "NAND2"

DEFINE NAND2 In1 In2 OUT; % Definición de subcircuito "NAND2"
LOCAL x2;                 % Declaración de nodos globales

PTRANS  In1   Vdd  Out;   % MOSFET P1
PTRANS  In2   Vdd  Out;   % MOSFET P2
NTRANS  In2   x2   Out;   % MOSFET N3
NTRANS  In1   Gnd  x2 ;   % MOSFET N4
END;
```

El circuito AND2 se construye interconectando las celdas "INV" y "NAND2" como se indica en el esquema de la Fig.5.12.

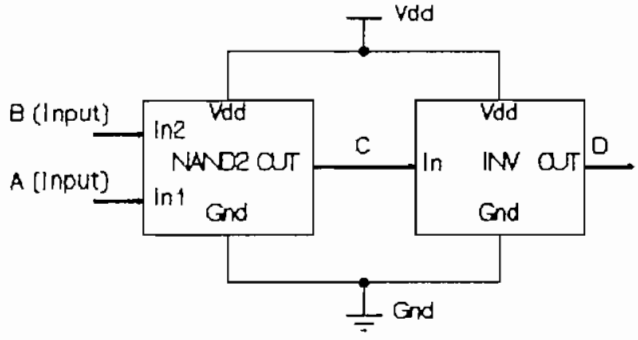


FIGURA 5.12 Interconexión de los subcircuitos NAND2 e INV para formar el circuito AND2.

Los nodos A y B deben declararse como nodos de entrada (INPUT) para el circuito global. La NETLIST NDL que describe el esquema de la Fig.5.12 es:

```
%    CIRCUITO AND DE 2 ENTRADAS: "AND2"
INPUT  A  B;              % Nodos de entrada
INV    C  D;              % Llamada a subcircuito INV
NAND2  A  B  C;           % Llamada a subcircuito NAND2
END                       % Fin de circuito
```

ii) Comandos de simulación (archivo .SIM)

Del mismo modo que en el programa de simulación SPICE, la NETLIST únicamente describe al circuito, pero es insuficiente para controlar el proceso de simulación y se requiere de:

ii.1) Señales de polarización y entrada de datos, y

ii.2) Directivas de control.

Que en el caso del simulador NDL no se incluyen junto con la descripción del circuito sino que se pueden digitar conforme la simulación se realiza o incluir en un archivo de comandos con extensión .SIM.

ii.1) Señales de polarización y entrada de datos:

Puesto que para la simulación lógica únicamente se disponen de 3 valores lógicos (1, 0 y X), las señales de polarización son:

```
LIGUE Vdd 1
LIGUE Gnd 0
```

que mantienen a los nodos Vdd y Gnd en un estado lógico permanente durante toda la simulación.

Las señales de entrada únicamente pueden especificarse para nodos declarados en la netlist como de INPUT (de entrada). En el ejemplo, a fin de cubrir todas las posibilidades de combinación lógica se ha definido;

```
ENTRADA A 0011
ENTRADA B 0101
```

ii.2) Directivas de control:

En este caso se ha escogido la visualización gráfica de los resultados mediante la generación del interfaz gráfico de la Fig.5.10 con el comando **GRAFICO**. Los nodos de interés deben asignarse a los canales del interfaz mediante el comando **PLUGUE**, así:

```
GRAFICO          % Modo gráfico de simulación
PLUGUE A 1       % Conexión del nodo A (INPUT) al canal 1
PLUGUE B 2       % Conexión del nodo B (INPUT) al canal 2
PLUGUE C 4       % Conexión del nodo C (NORMAL) al canal 4
PLUGUE D 6       % Conexión del nodo D (NORMAL) al canal 6
FASE 10          % Intervalo de simulación de 10 fases
```

Al final se ha definido el intervalo de simulación en número de fases (FASE) o de ciclos (CICLO), según el intervalo de tiempo que se desee que dure cada valor lógico.

iii) Realización de la simulación

Reuniendo los elementos enumerados anteriormente, el archivo de descripción del circuito AND2 es el siguiente (se han añadido la directiva de definición de tecnología, y algunos comentarios para documentación):

```
% SIMULACION LOGICA DEL CIRCUITO AND DE 2 ENTRADAS "AND2"
% ======================================================

FAMILY CMOS;              % Tecnología de diseño

% SUBCIRCUITO NAND 2 ENTRADAS: "NAND2"
DEFINE NAND2 I1 I2 OUT;  % Definición de subcircuito "NAND2"
LOCAL x2;                % Declaración de nodos globales
PTRANS  I1   Vdd  Out;   % MOSFET P1
PTRANS  I2   Vdd  Out;   % MOSFET P2
NTRANS  I2   x2   Out;   % MOSFET N3
NTRANS  I1   Gnd  x2 ;   % MOSFET N4
END;
```

```
% SUBCIRCUITO INVERSOR: "INV"
DEFINE INV In OUT;        % Definición del subcircuito INV
LOCAL;                    % No existen nodos locales internos
PTRANS  In   Vdd  Out;    % MOSFET P1 (nodos: compuerta, fuente, drenaje)
NTRANS  In   Gnd  Out;    % MOSFET N2 (nodos: compuerta, fuente, drenaje)
END;

% CIRCUITO AND DE 2 ENTRADAS: "AND2"
INPUT  A  B;              % Nodos de entrada
INV    C  D;              % Llamada a subcircuito INV
NAND2  A  B  C;           % Llamada a subcircuito NAND2
END                       % Fin de circuito
```

Y el archivo con los comandos de simulación es:

```
% SEÑALES LOGICAS DE POLARIZACION
LIGUE Vdd 1
LIGUE Gnd 0

% SEÑALES LOGICAS DE DATOS
ENTRADA A 0011
ENTRADA B 0101

% DIRECTIVAS DE CONTROL
GRAFICO                   % Modo gráfico de simulación
PLUGUE A 1                % Conexión del nodo A (INPUT) al canal 1
PLUGUE B 2                % Conexión del nodo B (INPUT) al canal 2
PLUGUE C 4                % Conexión del nodo C (NORMAL) al canal 4
PLUGUE D 6                % Conexión del nodo D (NORMAL) al canal 6
FASE 10                   % Intervalo de simulación de 10 FASES
```

Los resultados de esta simulación se indican en la Fig.5.13.

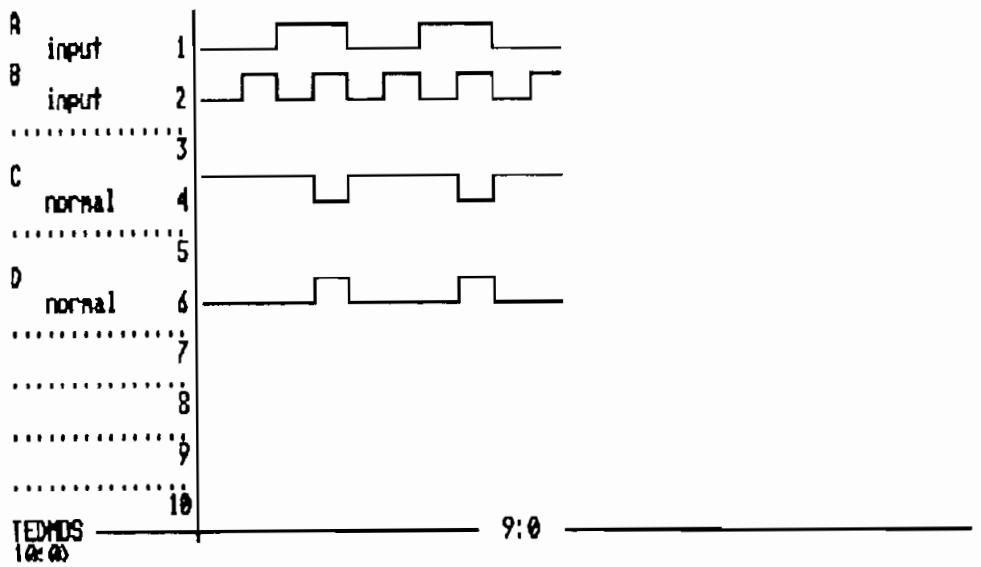


FIGURA 5.13 Exhibición gráfica de la simulación lógica del circuito AND2.

5.4 FORMATO CIF PARA DESCRIPCION GEOMETRICA DE LAYOUTS

El formato de descripción CIF, versión 2.0 (Caltech Intermediate Form), concebido por Sutherland y Ayers en Caltech, es un formato para la descripción de ítems gráficos. Esta forma de descripción geométrica es aceptada ampliamente en la actualidad como forma de intercambio estándar entre descripciones gráficas o simbólicas de alto nivel de un CI y las descripciones de bajo nivel necesarias para la generación de las máscaras empleadas en las fundidoras, así como para programas de visualización gráfica tanto impresa como de video.

La idea básica de este formato es especificar literalmente, con gran precisión, cada objeto geométrico en el diseño. Su uso provee a los grupos de diseño de un acceso fácil a dispositivos de salida, permite compartir diseños, combinar varios diseños en un solo diseño mayor, y así por el estilo.

El formato CIF sirve por tanto como denominador común en la descripción de varios proyectos de sistemas integrados, independientemente de los métodos de diseño empleados en su implantación, el layout es trasladado a formato CIF como intermediario, antes de ser transformado otra vez a cada formato particular, dependiendo de los dispositivos de salida usados por cada diseñador.

5.4.1 Lenguage de descripción CIF

La idea fundamental del formato CIF es describir sin ambigüedad la geometría de los layouts VLSI. En consecuencia, es importante que todos los diseñadores tengan exactamente el mismo entendimiento de esta forma de descripción, cuya estructura se indica a continuación.

a) Directivas de descripción

La descripción CIF consiste en un conjunto de directivas formadas por secuencias de caracteres ASCII, que por tanto pueden ser generadas en cualquier editor de texto.

Las directivas empleadas en el formato CIF y su formato son: (6)

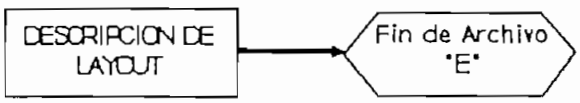
Directiva	Formato
Polígono (*Polygon*) con un camino	P camino
Caja (*Box*) con longitud, ancho, centro, y dirección (dirección por omisión: 1,0)	B entero entero punto punto
Rayo circular (*Round*) con diámetro y centro	R entero punto
Alambre (*Wire*) con ancho y camino	W entero camino
Especificación de capa (*Layer*)	L nombre-corto
Definición de símbolo inicial (*Definition Start*) con índice a,b (por omisión a=b=1)	DS entero entero entero
Definición de símbolo final (*Definition Finish*)	DF
Supresión de definiciones de símbolos (*Definitions Delete*)	DD entero
Símbolo de llamada (*Call*)	C entero transformación
Expansión de usuario	dígito Texto-de-usuario
Comentarios con texto arbitrario	(Texto de comentario)
Marcador de final (*End*)	E

(6) "Introduction to VLSI Systems", C. Mead - L. Conway, pág 116.

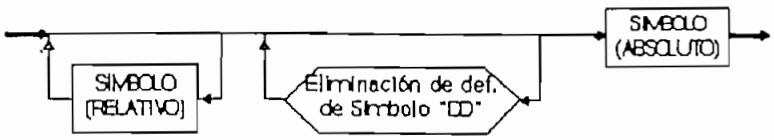
En las descripciones estructurales subsiguientes se emplea la misma simbología usada en la descripción de los lenguages SPICE y NDL, con la diferencia de que en este caso no existen elementos circuitales sino solo los "**comandos**" o "**directivas de descripción**" indicadas arriba y cuya sintaxis se detalla en el Anexo C.

b) **Estructura general del archivo CIF**

La estructura general del archivo CIF es la siguiente:



c) **Descripción del LAYOUT**



El LAYOUT tiene una estructura **jerárquica** basada en símbolos, que a manera de subcircuitos pueden ser invocados para ser integrados a los símbolos más grandes, y estos en símbolos aún mayores tantas cuantas veces sea necesario.

Los **símbolos "relativos"** se denominan así porque sus coordenadas de descripción son relativas al origen particular de la grilla del símbolo.

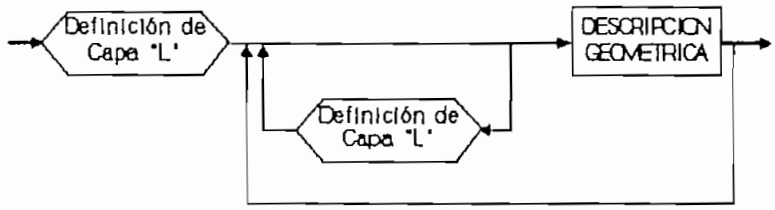
Unicamente existe un **símbolo "absoluto"** definido como aquel de mayor jerarquía en la descripción geométrica total,

por tanto sus coordenadas son absolutas, y su origen de coordenadas se convierte en el origen de coordenadas absoluto de la descripción del layout global del circuito.

d) Símbolo

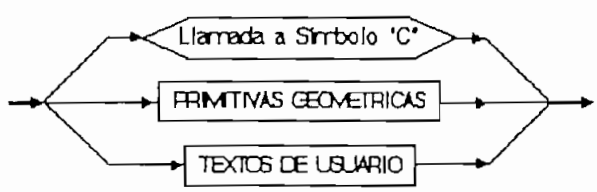
La estructura tanto de los **símbolos "relativos"** como del **símbolo "absoluto"** es la siguiente:

e) Descripción del símbolo



La **definición de capa "L"** es la directiva que indica las máscaras de fundición en que serán implantadas las estructuras contenidas en la **descripción geométrica** y se mantiene invariable hasta que una nueva definición de capa sea especificada.

f) Descripción geométrica

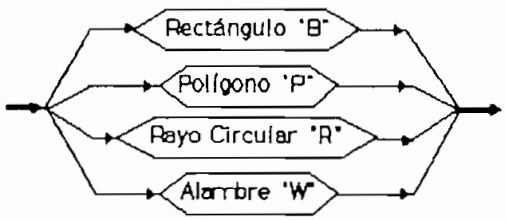


Las **llamadas a símbolos** son la escencia de la estructura jerárquica del formato CIF, y permiten insertar símbolos definidos anteriormente (símbolos de biblioteca) en sectores específicos de símbolos de mayor jerarquía.

Esta estructura provee de elasticidad al formato CIF y permite reducir el tamaño de sus archivos de descripción que de otra manera serían demasiado grandes para geometrías complejas.

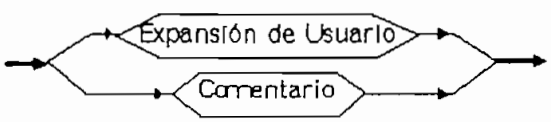
g) Primitivas **geométricas**

Para su descripción en formato CIF, todo layout debe ser expresado en términos de las estructuras geométricas elementales:



h) Textos de usuario

Los textos de usuario se definen bajo la modalidad de "**comentarios**" empleados para fines de documentación, y de "**expansiones de usuario**" que pueden ser concebidos como comandos propios de usuario para programas de post-procesamiento del layout.



5.4.2 Sistema de coordenadas para las descripciones CIF

El formato CIF usa el sistema referencial de coordenadas cartesianas de la Fig.5.14. Las direcciones y distancias son interpretadas siempre en términos de la vista frontal del CI terminado.

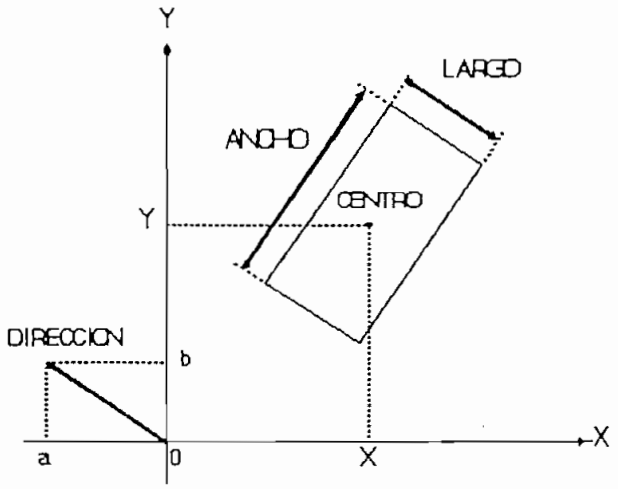


FIGURA 5.14 Representación de la primitiva *rectángulo* en el sistema referencial de coordenadas del formato CIF.

a) Unidades de medida

Las unidades de medida de distancias son centésimas de micras μm, no existiendo límite en el tamaño del número (a excepción de los límites propios del sistema de computación empleado).

b) Direcciones

En lugar de medir la rotación por ángulos, en el formato CIF se usa el sistema de "**vectores de dirección**" que se especifican mediante parejas de números enteros, lo que

elimina la necesidad de funciones trigonométricas en sus aplicaciones y evita el problema de selección de unidades de medidas angulares.

El par ordenado de enteros (a,b) (Fig.5.14) que caracterizan al vector de dirección corresponden a las componente del vector de dirección a lo largo del eje *x* y del eje *y*, respectivamente, así, p. ej. la dirección de 45° con respecto al eje x(+) podría ser representada por los enteros (1,1), (5,5), etc.., la dirección por omisión es aquella que coincide con el eje x(+), es decir (1,0) ó (n,0).

5.4.3 Ejemplo de aplicación

A continuación, a manera de ejemplo, se describe el layout de la celda AND de 2 entradas "AND2" construida en base a las celdas básicas INVERSOR "INV" y NAND de 2 entradas "NAND2", diseñadas en base a la tecnología ECPD de 1.5 μm, que fueron simuladas eléctrica y lógicamente con anterioridad.

a) Código de representación de capas

En la representación de las primitivas geométricas correspondientes a las diferentes máscaras del layout se ha empleado el siguiente código:

Pozo N (CNWI)

Implante N+ (CNPI)

Implante P+ (CPPI)

Area Activa (CTOX)

Polysilicon (CPOL)

Metal 1 (CME1)

Contacto (CCON)

b) Definición de símbolos (céldas básicas)

Inicialmente se ha definido el símbolo en formato CIF de la celda del inversor "INV" cuyo layout se exhibe en la Fig.5.15.

```
DS 10  10  1;
9 "CELDA INV";
L CNWI;
B 210 500 105 830;
L CNPI;
B 60 60 110 60;
B 70 60 135 400;
B 40 280 120 230;
L CPPI;
B 70 60 135 660;
B 40 70 120 725;
B 70 160 135 840;
B 60 110 130 975;
L CTOX;
B 60 60 110 60;
B 70 60 135 400;
B 40 280 120 230;
B 70 60 135 660;
B 40 70 120 725;
B 70 160 135 840;
B 60 110 130 975;
L CPOL;
B 20 820 50 530;
B 100 40 110 280;
B 130 20 125 850;
L CME1;
B 210 60 105 1000;
B 210 60 105 60;
B 60 60 140 400;
B 60 60 140 660;
B 40 40 190 650;
B 40 40 190 410;
L CCON;
B 20 20 110 60;
B 20 20 130 1000;
B 20 20 140 400;
B 20 20 140 660;
DF;
```

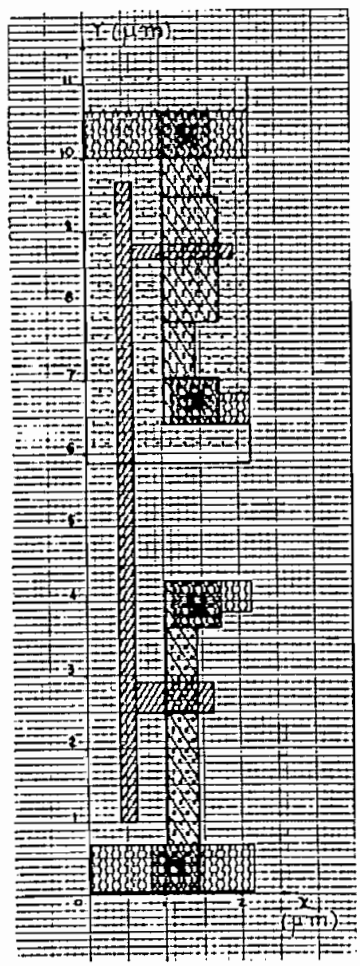


FIGURA 5.15 Layout de una celda correspondiente a un inversor en tecnología CMOS 1.5µ.

En la descripción CIF anterior cabe notar que:

i) Se ha asignado el #10 al símbolo de la celda INV implantada.

ii) Del factor de escala asociado al símbolo (DS 10 10 1;) se desprende que A=10 y B=10, de acuerdo a la Ec.[C.1] (ver Anexo C) todas las distancias son alteradas por un factor:

$$\textit{Distancia final} = \frac{\textit{Distancia en la definición} * 10}{1}$$

es decir que, por ejemplo, para la capa de polysilicon (CPOL), la descripción de las primitivas geométricas es equivalente a:

```
DS 10 1 1;
'
L CPOL;
B 200 8200 500 5300;
B 1000 400 1100 2800;
B 1300 200 1250 8500;
'
DF;
```

iii) Todos los parámetros de las primitivas geométricas de la celda se han definido con respecto a la esquina inferior izquierda que corresponde a su origen de coordenadas.

Bajo las mismas premisas se ha definido el símbolo #20 (DS 20) correspondiente al layout de la celda NAND de 2 entradas "NAND2" (Fig.5.16) cuya descripción CIF se detalla a continuación.

```
DS 20  10  1;
9 "CELDA NAND2";
L CNWI;
B 370 500 185 830;
L CNPI;
B 60 60 60 1000;
B 60 60 190 400;
B 60 60 200 60;
B 40 280 190 230;
L CPPI;
B 60 60 60 60;
B 40 120 130 750;
B 70 120 145 870;
B 80 60 150 660;
B 80 140 220 960;
B 70 140 225 700;
B 40 120 240 830;
L CPOL;
B 20 820 50 530;
B 140 20 130 860;
B 170 30 145 135;
B 160 30 230 195;
B 140 20 240 720;
B 20 820 320 530;
L CME1;
B 370 60 185 60;
B 370 60 185 1000;
B 150 60 185 660;
B 60 60 190 400;
B 150 40 295 410;
B 110 40 315 650;
L CCON;
B 20 20 60 1000;
B 20 20 60 60;
B 20 20 150 660;
B 20 20 190 400;
B 20 20 200 60;
B 20 20 220 1000;
B 20 20 220 660;
DF;
```

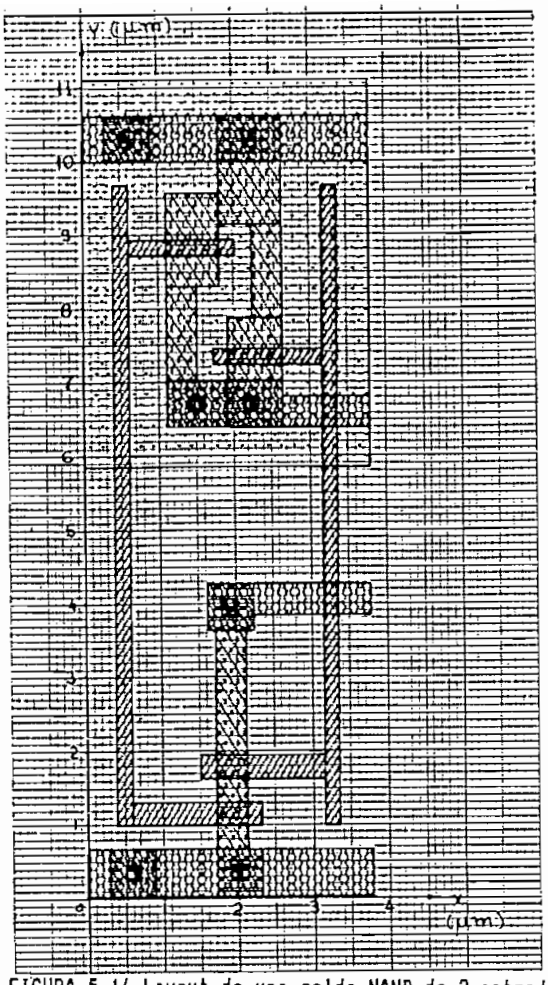


FIGURA 5.16 Layout de una celda NAND de 2 entradas diseñada en tecnología CMOS.

c) Definición de layout final

La celda final "AND2" se construye en base a la unión de los diseños de las dos celdas básicas anteriores, que dan lugar al layout de la Fig.5.17. cuya descripción CIF únicamente requiere de las llamadas a las símbolos definidos anteriormente:

```
DS 30 10 1;
9 "CELDA AND2";
C 20;
C 10 T -580 0 MX;
DF;
C 30;
E
```

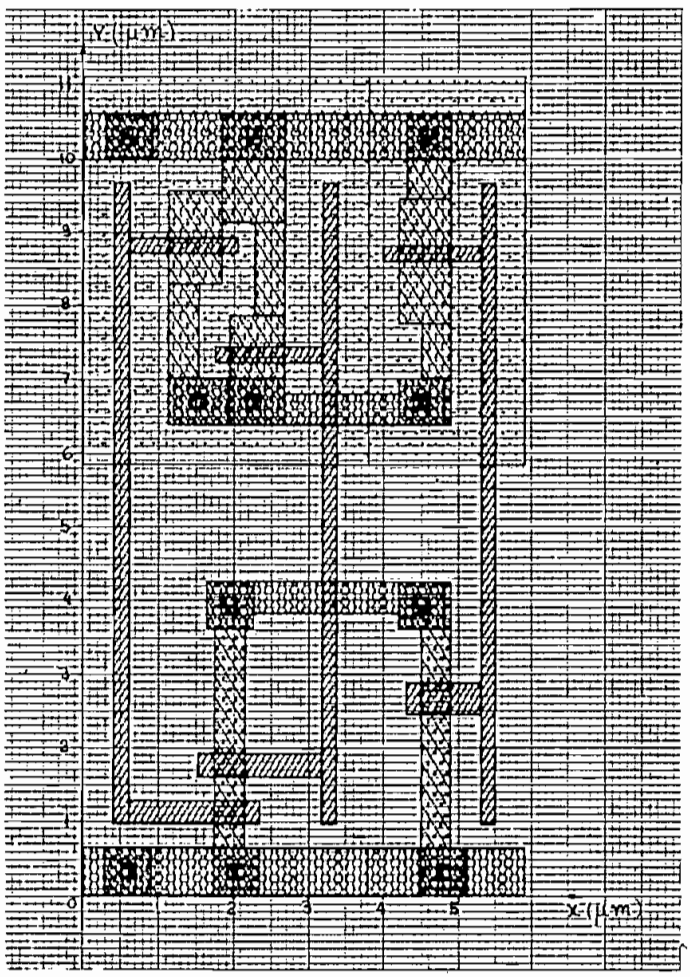


FIGURA 5.17 Layout de la celda de AND de 2 entradas.

Para la construcción de este layout ha sido necesario:

i) Llamar al símbolo #20 "C 20" correspondiente a la celda "NAND2", conservando el origen referencial de su grilla por lo que no se han requerido transformaciones en su posición en el layout.

ii) Llamar al símbolo #10 "C 10" correspondiente a la celda "INV"(Fig.5.18a), en este caso es necesario trasladar -5.8µm el origen de la grilla relativo a esta celda (Fig.5.18b) y reflejarla luego en la dirección del eje "y" (Fig.5.18c), nótese que la celda "INV" en el layout de la celda "NAND2" se halla invertida en "y" con respecto a su definición original Fig.5.15, de modo que su entrada se empalme con la salida de la celda NAND2. La transformación requerida será: "T -580 0 MX"

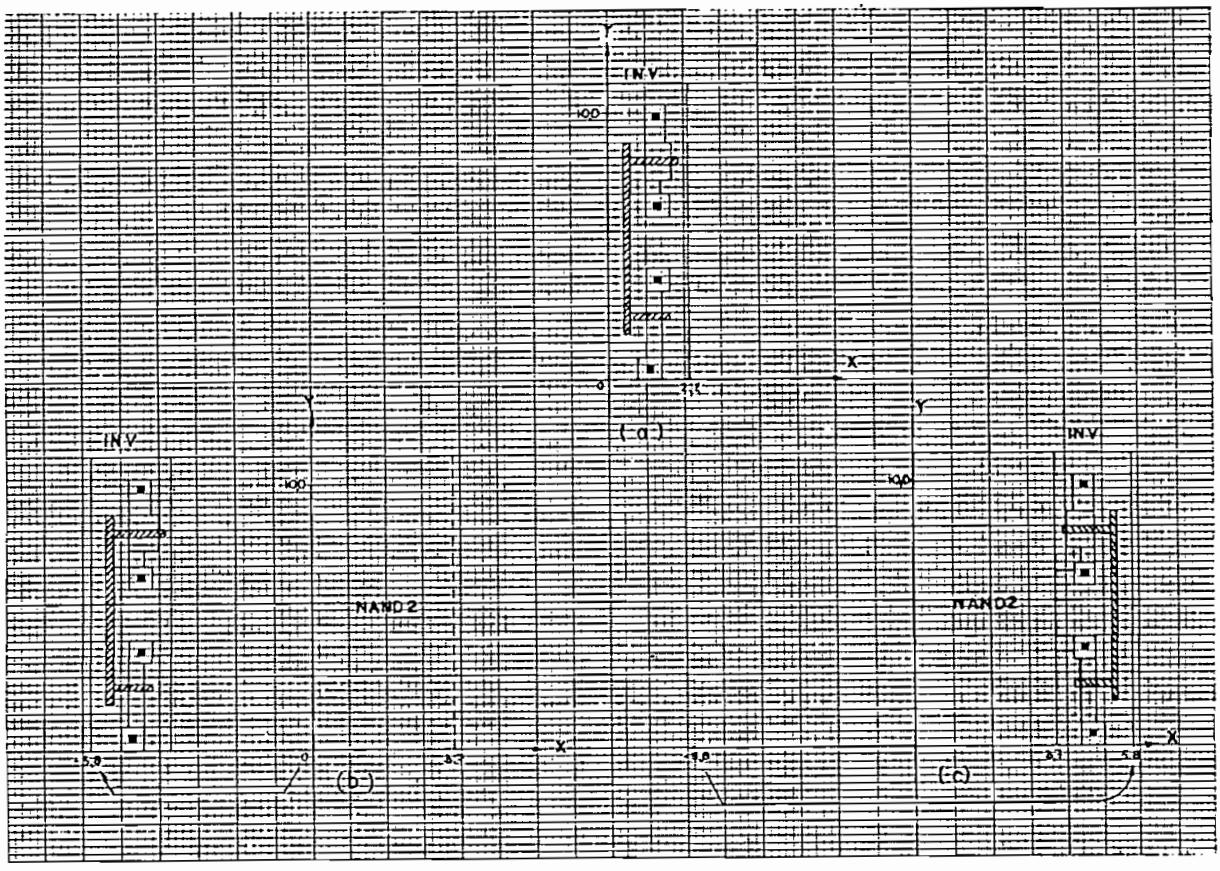


FIGURA 5.18 Movimientos de traslación de la celda "INV".

iii) Finalmente, cabe insistir en que la conversión de escalas indicada en la definición del símbolo #30 (DS 30 10 1) afecta tan solo a las dimensiones de las transformaciones en las llamadas (C 20.. y C 10..).

Es decir:

```
DS 30 10 1;              DS 30 1 1;
C 20;                    C 20;
C 10 T -580 0 MX;        C 10 T -5800 0 MX;
DF;                      DF;
```

son equivalentes.

iv) Por el contrario, las medidas de los símbolos internos a los símbolos llamados:

```
DS 10 10 1;              DS 20 10 1;
'                        '
'                        '
'                        '
DF;                      DF;
```

no son afectadas por la transformación de escalas y solo dependen de los argumentos propios de sus definiciones (DS 10 .. y DS 20..).

CAPITULO 6

DISEÑO DE ASICs BASADO EN CELDAS ESTANDAR

- METODOLOGIA CONVENCIONAL -

En los capítulos anteriores se ha delineado a grandes rasgos el marco teórico-práctico de diseño de circuitos integrados. Es de destacar que en el desarrollo de estos capítulos se ha realizado una revisión panorámica de las alternativas de diseño, sus ventajas, desventajas, aplicaciones y un análisis breve de los diferentes tipos de herramientas de diseño y su utilización.

En el capítulo presente se estudia en detalle la metodología clásica de diseño desde los puntos de vista de:

i) El estilo de diseño *semi-custom* basado en celdas estándar (*standard cells*).

ii) La concepción, diseño y simulación de ASICs mediante un conjunto de herramientas agrupadas en su mayor parte en torno al gerenciador *TENTOS* (desarrollado en la Universidad Federal de Río Grande del Sur - Brasil)

6.1 DISEÑO BASADO EN BANDAS Y CELDAS ESTANDAR

6.1.1 Definiciones generales

Conviene introducir algunas definiciones que permitan manejar de mejor manera los conceptos geométricos del diseño del layout de una celda estándar:

a) **Geometría de las celdas:** corresponde a la descripción detallada de todas las máscaras necesarias para el diseño de una celda.

b) **Topología de las celdas:** hace referencia a una visión externa, en la que se indica el tamaño, la forma y la situación de las entradas y salidas de cada celda.

6.1.2 Celdas estándar

Las celdas estándar (*standard cells*) son módulos lógicos (Fig.6.1) con una geometría prediseñada en base a ciertas restricciones que simplifican su ubicación topológica regular y su conexionado dentro del plano de base del diseño (Fig.6.2). Estas restricciones son:

a) Altura fija y conexionado lateral de las alimentaciones (polarización y tierra) por simple adyacencia con las celdas vecinas.

b) Capacidad de acceso a los terminales (pines) de entrada y salida de cada celda desde sus celdas adyacentes.

El ancho de las celdas no tiene restricción y depende de su funcionalidad.

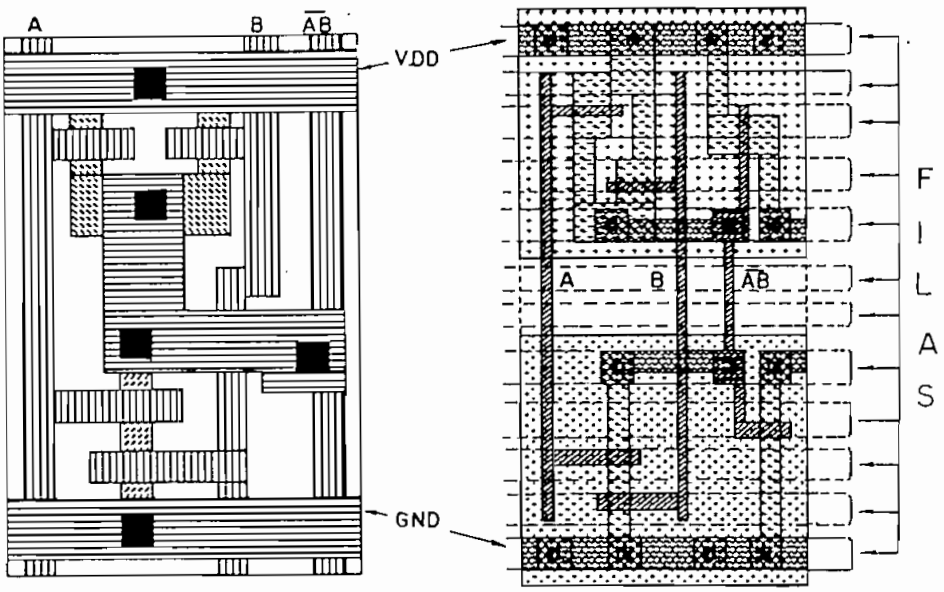


FIGURA 6.1 (a) Layout típico de una celda estándar.
(b) Layout típico basado en FILAS (*TRILLAS*).

En algunas implantaciones se busca la interconexión no sólo de las alimentaciones por adyacencia sino que se generaliza este criterio mediante la definición de Filas (*Trillas*)[1] (Fig.6.1(b)) consistentes en canales horizontales paralelos a las líneas de alimentación que permiten estan-

[1] "SCHAROP - UM ROTEADOR DETALHADO DE CANAL", C. Pereira - D. Couto.

darizar los diseños de las máscaras dentro de las celdas, garantizándose de esta manera uniformidad en las geometrías de las celdas y como se analizará posteriormente: mayores posibilidades de interconexión y por ende mejor aprovechamiento del área de silicio.

6.1.3 Plano de base del diseño basado en celdas estándar

El plano de base genérico de un circuito basado en celdas estándar se ilustra en la Fig.6.2 y está construido de la siguiente manera:

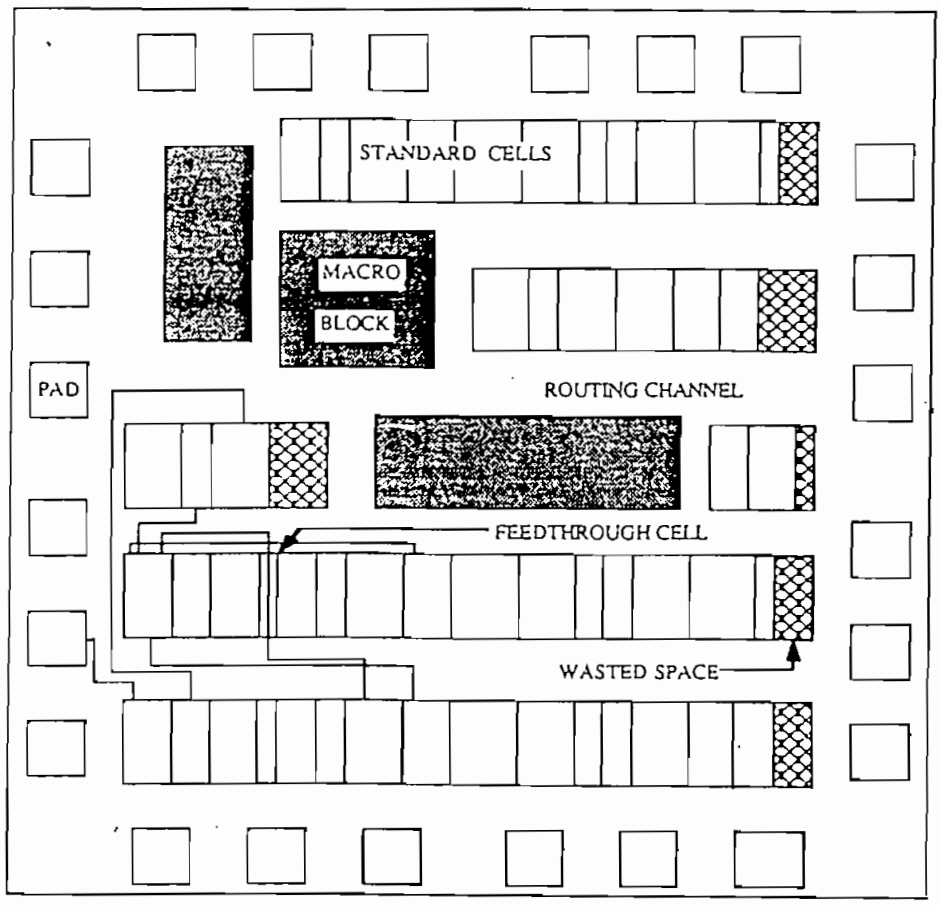


FIGURA 6.2 Plano de base genérico de un diseño en base a celdas estándar.

Las celdas son agrupadas en *Bandas* cuya altura está dada por la altura de la celda estándar. Entre las bandas existen *Canales de Enrutamiento* que son espacios reservados para el establecimiento de las interconexiones entre los componentes del diseño.

Como se indicó anteriormente, la geometría de las celdas es concebida de manera que las interconexiones de alimentación se orienten horizontalmente a través de los bordes superior e inferior de cada celda, éstas interconexiones delinean un trazo continuo en cada banda cuando las celdas se colocan en forma adyacente.

Las entradas y salidas lógicas de cada módulo se hallan en *Pines* consistentes en terminales accesibles lateralmente o desde los bordes superior o inferior de la celda.

Las *Interconexiones Horizontales* de los pines de las celdas se realizan mediante conductores que corren externa y/o internamente a las bandas de las celdas a que pertenecen:

a) Externamente, a través de los *canales de enrutamiento.*
b) Internamente, a través de los caminos horizontales delineados por las *filas* libres de las celdas.

Las *Interconexiones Verticales* entre bandas se realizan mediante *celdas estándar de interconexión* que contienen conductores dispuestos verticalmente que dan lugar a los

canales de enrutamiento vertical, estas celdas se ubican preferentemente en los bordes de las bandas (Fig.6.2).

Se acostumbra implantar los caminos de enrutamiento verticales en un solo tipo de material, y los caminos horizontales en otro material diferente. Así por ejemplo:

Enrutamiento vertical	Enrutamiento horizontal
Polysilicon	Metal 1
Metal 2	Metal 1

Finalmente, las bandas pueden dividirse en *Regiones*[2], con el criterio de agrupar en cada *región* aquellas celdas entre las que el número de interconexiones mutuas es mayor, con el fin de garantizar una distribución homogénea de las zonas de interconexión y de facilitar los algoritmos de enrutamiento.

6.1.4 Ensamblaje de celdas

El problema del ensamblaje de celdas para generar el layout de un circuito ASIC diseñado en base a celdas estándar puede enunciarse de la siguiente manera:

> "Dado un circuito electrónico consistente en módulos ortogonales con terminales de entrada y salida predefinidos e interconectados en forma predefinida, construir un layout indicando las posiciones de

[2] "Manual do Usuário do Projeto TRANCA V1.0", F. Moraes - M. Lubaszewski - R. Reis.

los módulos de manera tal que la longitud de las interconexiones y el área del layout sean mínimos.- "(3)

Del planteamiento anterior se concluye que los datos de entrada de este problema son:

a) La descripción geométrica y topológica de los módulos a interconectarse.

b) La descripción de las interconexiones entre los terminales de los módulos, que puede plantearse en forma de *NETLIST* o de esquema lógico-circuital.

Por otro lado, los datos de salida (resultados) son:

a) La lista de coordenadas x e y que determinan la posición y describen la geometría de todos los módulos.

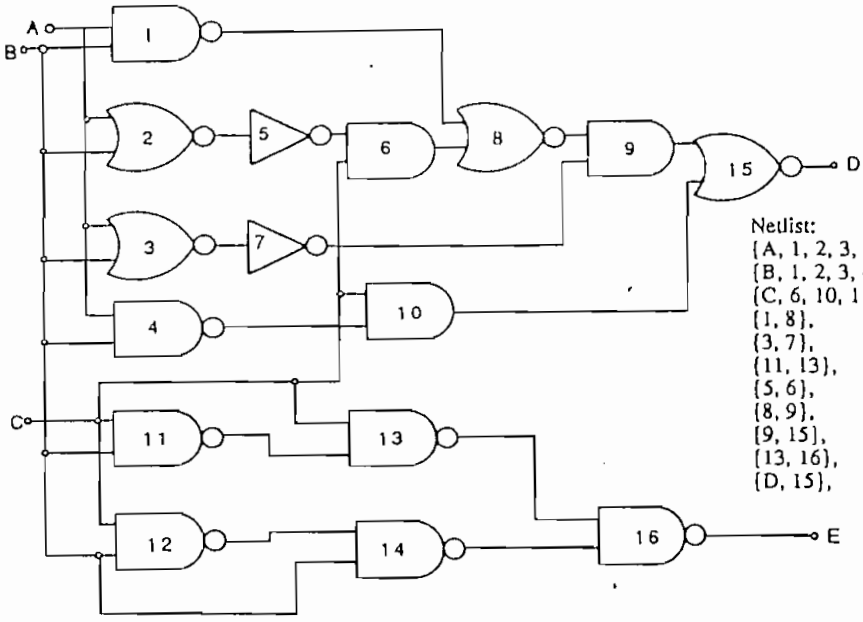
b) La geometría de las zonas de enrutamiento, esto es las capas correspondientes a las bandas y canales de interconexión que enlazan los pines de los módulos.

La Fig.6.3. representa un ejemplo del problema de ensamblaje de celdas donde los datos de entrada han sido planteados tanto en forma de NETLIST como esquemática (Fig.6.3 (a)), y como resultado se han obtenido los módulos ubicados e interconectados en el estilo de layout de celdas estándar (Fig.6.3 (b)).

---

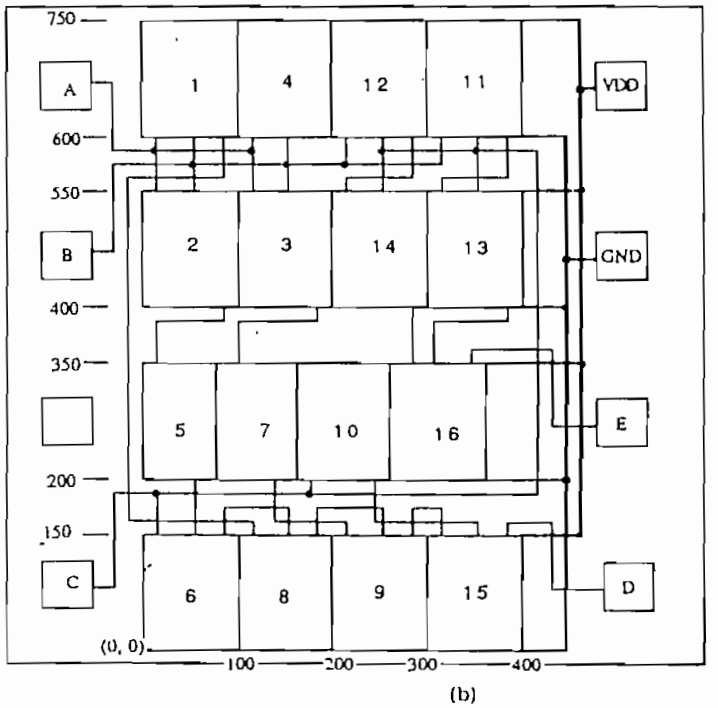
(3) "VLSI Cell Placement Tecnhiques", K. Shahookar - P. Mazumder, ACM Computing Surveys Vol. 23, No.2, Junio 1991, , pág. 144.

Para reducir la complejidad que supondría abordar la solución de este problema de forma global, se lo ha dividido en tres fases:



Netlist:
{A, 1, 2, 3, 4},
{B, 1, 2, 3, 4, 11, 12},
{C, 6, 10, 11, 12, 13},
{1, 8}, {2, 5},
{3, 7}, {4, 10},
{11, 13}, {12, 14},
{5, 6}, {6, 8},
{8, 9}, {7, 9},
{9, 15}, {10, 15},
{13, 16}, {14, 16},
{D, 15}, {E, 16}.

(a)



Placement
(cell, x, y):

(1, 0, 600)
(2, 0, 400)
(3, 100,400)
(4, 100,600)
(5, 0, 200)
(6, 0, 0)
(7, 75, 200)
(8, 100, 0)
(9, 200, 0)
(10, 150, 200)
(11, 300, 600)
(12, 200, 600)
(13, 300, 400)
(14, 200, 400)
(15, 300, 0)
(16, 250, 200)

(b)

FIGURA 6.3 Ejemplo de definición del problema de ensamblaje de celdas
(a) Planteamiento del problema (NETLIST) (b) Esquema resultante (coordenadas de los módulos).

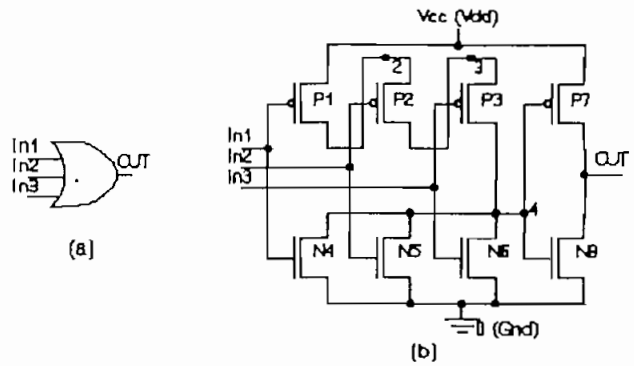
a) **Ubicación:** esta fase consiste en distribuir de manera óptima los módulos en las bandas, cuidando de proveer el espacio suficiente para el trazado de las conexiones entre ellos.

b) **Conexionado o enrutamiento:** en esta fase se procede a interconectar los módulos mediante el trazado del enrutamiento de las señales dentro y entre las bandas generadas en la fase anterior. Podría ocurrir que debido a la complejidad del circuito formulado o por limitaciones de los algoritmos empleados, no se encontrase ninguna solución válida, en cuyo caso se debería replantear la fase de ubicación.

c) **Compactación:** durante esta fase se trata de recuperar el espacio sobrante de aquel que se había previsto para conexiones y que no fue utilizado.

6.1.5 Biblioteca de celdas

La parte escencial de un entorno de diseño basado en celdas estándar constituye su *biblioteca de celdas*. La biblioteca de celdas es la fuente que contiene la información geométrico-topológica del conjunto de celdas disponibles para la implantación de un diseño. Es decir, la biblioteca de celdas almacena las distintas representaciones asociadas a cada celda, que permiten abordar las diferentes etapas de diseño.



(c)

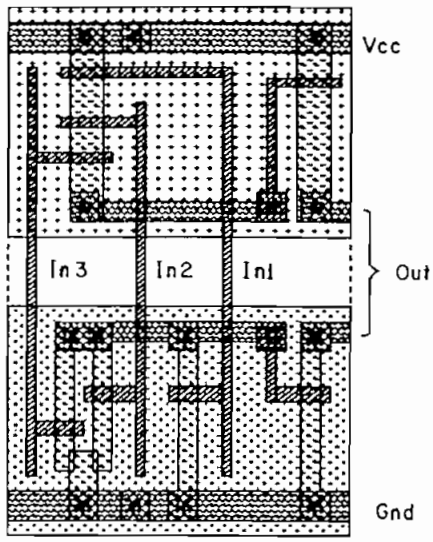


FIGURA 6.4 Ejemplo de una celda OR de 3 entradas con sus diferentes tipos de representaciones: (a) Simbolo, (b) Esquema y (c) Layout .

Los tipos de representaciones más habituales de cada celda pueden verse en la Fig.6.4 y son:

a) Símbolo

Corresponde a la forma gráfica que representa a cada celda en los procesos de captura gráfica de esquemas.

b) Esquema

Consistente en la representación lógica o eléctrica empleada para indicar la forma en que está implantada la celda (no siempre se halla a disposición de los usuarios externos a la firma fabricante).

c) Layout

Corresponde a la descripción de las máscaras necesarias para la implantación física de la celda.

En algunos casos la información geométrica de la celda está disponible en su totalidad para el usuario e incluso es susceptible de modificaciones y adaptaciones a cada diseño particular. En otros casos, en cambio, el usuario solo tiene acceso a una abstración o visión externa del layout consistente en una caja negra en la que se indican dimensiones, forma y situación de los pines de E/S de la celda; esta representación la utiliza el diseñador en sus procesos de ubicación y conexionado en el plano de base, y luego es reemplazada por el fabricante en el proceso de generación de las descripciones físicas definitivas del circuito integrado.

d) Modelos de simulación

Cada celda dispone de su modelo de simulación consistente en la NETLIST empleada para pasar al simulador o simu-

ladores la descripción circuital de la celda, su formato dependerá de las características propias de cada simulador.

Generalmente se acostumbra plantear los modelos de simulación en forma de definiciones de SUBCIRCUITOS que son invocados por los módulos funcionales que los contienen, o por el circuito principal, al estructurarse sus respectivos modelos de simulación.

A manera de ejemplo, se indican a continuación los modelos empleados para la *simulación circuital* "SPICE" y la *simulacion lógica* "NDL" de la celda básica de la Fig.6.4, los nodos circuitales mantienen correspondencia con aquellos de la representación esquemática de la celda (Fig.6.4 (b)).

```
*-------------------------------------------------*
*       MODELO DE SIMULACION CIRCUITAL SPICE(4)   *
*                   CELDA OR3                     *
*-------------------------------------------------*
*
* SUBCIRCUITO : OR 3 ENTRADAS
*

.SUBCKT OR3 In1 In2 In3 OUT vcc

MP1    2  In1 vcc vcc  PMOS L=2.0U W=7.0U AD=233P AS=180P PD=126U PS=114U
MP2    3  In2  2  vcc  PMOS L=2.0U W=7.0U AD=233P AS=180P PD=126U PS=114U
MP3    4  In3  3  vcc  PMOS L=2.0U W=7.0U AD=233P AS=180P PD=126U PS=114U
MN4    4  In1 Gnd Gnd  NMOS L=3.0U W=4.0U AD=12P  AS=100P PD=14U  PS=64U
MN5    4  In2 Gnd Gnd  NMOS L=3.0U W=4.0U AD=12P  AS=100P PD=14U  PS=64U
MN6    4  In3 Gnd Gnd  NMOS L=3.0U W=4.0U AD=12P  AS=48P  PD=14U  PS=38U
MP7   OUT  4  vcc vcc  PMOS L=2.0U W=7.0U AD=233P AS=180P PD=126U PS=114U
MN8   OUT  4  Gnd Gnd  NMOS L=3.0U W=4.0U AD=12P  AS=48P  PD=14U  PS=38U

.ENDS OR3
```

(4) Biblioteca de celdas del paquete TENTOS, Universidad Federal do Rio Grande do Sul - Brasil, archivo OR3.CIR.

```
%-------------------------------------------------%
%          MODELO DE SIMULACION LOGICA NDL        %
%                    CELDA OR3                    %
%-------------------------------------------------%
%
% CELULA: OR 3 ENTRADAS
%
DEFINE OR3 In1 In2 In3 OUT;
LOCAL x2 x3 x4;

PTRANS    In1   Vdd  x2   ;%Tr(P1)
PTRANS    In2   x2   x3   ;%Tr(P2)
PTRANS    In3   x3   x4   ;%Tr(P3)
NTRANS    In1   Gnd  x4   ;%Tr(N4)
NTRANS    In2   Gnd  x4   ;%Tr(N5)
NTRANS    In3   Gnd  x4   ;%Tr(N6)
PTRANS    x4    Vdd  Out  ;%Tr(P7)
NTRANS    x4    Gnd  Out  ;%Tr(N8)
END;
```

6.2 SISTEMA DE HERRAMIENTAS "TENTOS" PARA DISEÑO CONVENCIONAL EN BASE A CELDAS ESTANDAR

Como se estudió en el Capítulo 3 el problema de diseño de ASICs, al margen de la metodología de diseño adoptada, se concentra en el uso de herramientas a diferentes niveles de diseño, cuyo uso puede concatenarse de manera automática o semiautomática hasta llegar al objetivo último del diseño VLSI que es la generación final de las máscaras correspondientes a la definición funcional de un diseño.

En consecuencia, los ingenieros en la rama de VLSI encargados de proyectar sus circuitos integrados requieren disponer de un conjunto de programas de software para microelectrónica. A fin de abordar este problema, se ha adoptado el estudio del paquete experimental desarrollado por la UFRGS (Universidad Federal do Rio Grande do Sul - Brasil) consis-

tente en un conjunto de herramientas para diseño basado en CELDAS ESTANDAR integradas en torno al sistema TENTOS.

6.2.1 Generalidades del sistema TENTOS

El desarrollo de los módulos pertenecientes al sistema TENTOS fue realizado independientemente por un gran número de personas con áreas de interés y conocimientos diversos. La integración entre estos módulos ha dado lugar al sistema TENTOS que ha sido presentado para su utilización a diseñadores iberoamericanos de circuitos integrados que no hayan tenido participación en su fase de implantación y pruebas.

El sistema TENTOS constituye una herramienta de síntesis semiautomática de circuitos en lógica aleatoria, y constituye una opción intermedia entre la compactación de circuitos FULL-CUSTOM y las posibilidades de síntesis automática completa de circuitos en base a CELDAS ESTANDAR, sus principales características son:

a) Transparencia de celdas, basada en una biblioteca de celdas geométricamente compatibles.
b) Estructura de bandas, regiones y filas (que fueron analizadas anteriomente).

El sistema actúa en las diversas etapas de concepción del circuito, desde la generación de celdas hasta su posicionamiento para la obtención del layout final. El proyecto está

compuesto por diversas herramientas que realizan actividades específicas dentro del ambiente, cuya utilización integrada permite la generación final del circuito.

6.2.2 El interfaz gráfico

El sistema TENTOS es administrado por un interfaz gráfico de alto nivel (Fig.6.5), en cuya zona superior se indican los módulos del sistema, y en la inferior características generales (nombre, formato y tecnología) del circuito en proceso de diseño

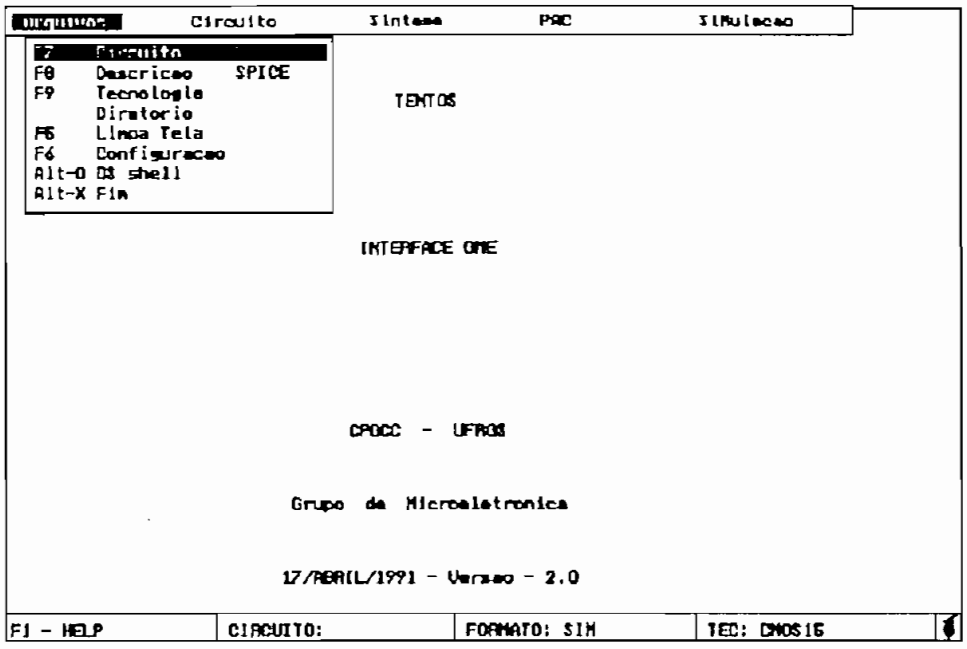


FIGURA 6.5 Interfaz principal del sistema TENTOS.

Las características básicas del interfaz son:

a) Organización jerárquica de los módulos del sistema.

b) Acceso directo a los programas del paquete.

a) Organización jerárquica de los módulos del sistema.

Los menúes del interfaz están organizados jerárquicamente tanto en la forma como en el fondo.

i) En la forma, ya que Tentos está constituído por un menú principal (dispuesto horizontalmente) cuyos submenúes se disponen verticalmnente, los submenúes de jerarquía inferior se generan parcialmente superpuestos con sus menúes padres dando lugar al árbol jerárquico cuya estructura se indica en la Fig.6.6.

ii) En el fondo, ya que las herramientas del sistema han sido agrupadas en el árbol de la Fig.6.6 siguiendo criterios de semejanza en cuanto a sus actividades. Además, el orden en que se hallan dispuestas, lejos de ser arbitrario, sugiere la secuencia que el diseñador debe seguir en el uso de las herramientas del sistema que se ilustra en la Fig.6.7.

Tanto los programas de la Fig.6.6 como su secuencia de uso (Fig.6.7), se detallan en los literales siguientes. En la realización de un diseño, los programas intercambian información por medio de archivos que se han denotado como *‹circuito›.**, además para su ejecución requieren de archivos especiales (por ejemplo: archivos.CFG), esta secuencia en la lectura y generación de archivos en cada programa es descrita en los literales siguientes como "**concatenación de archivos**"

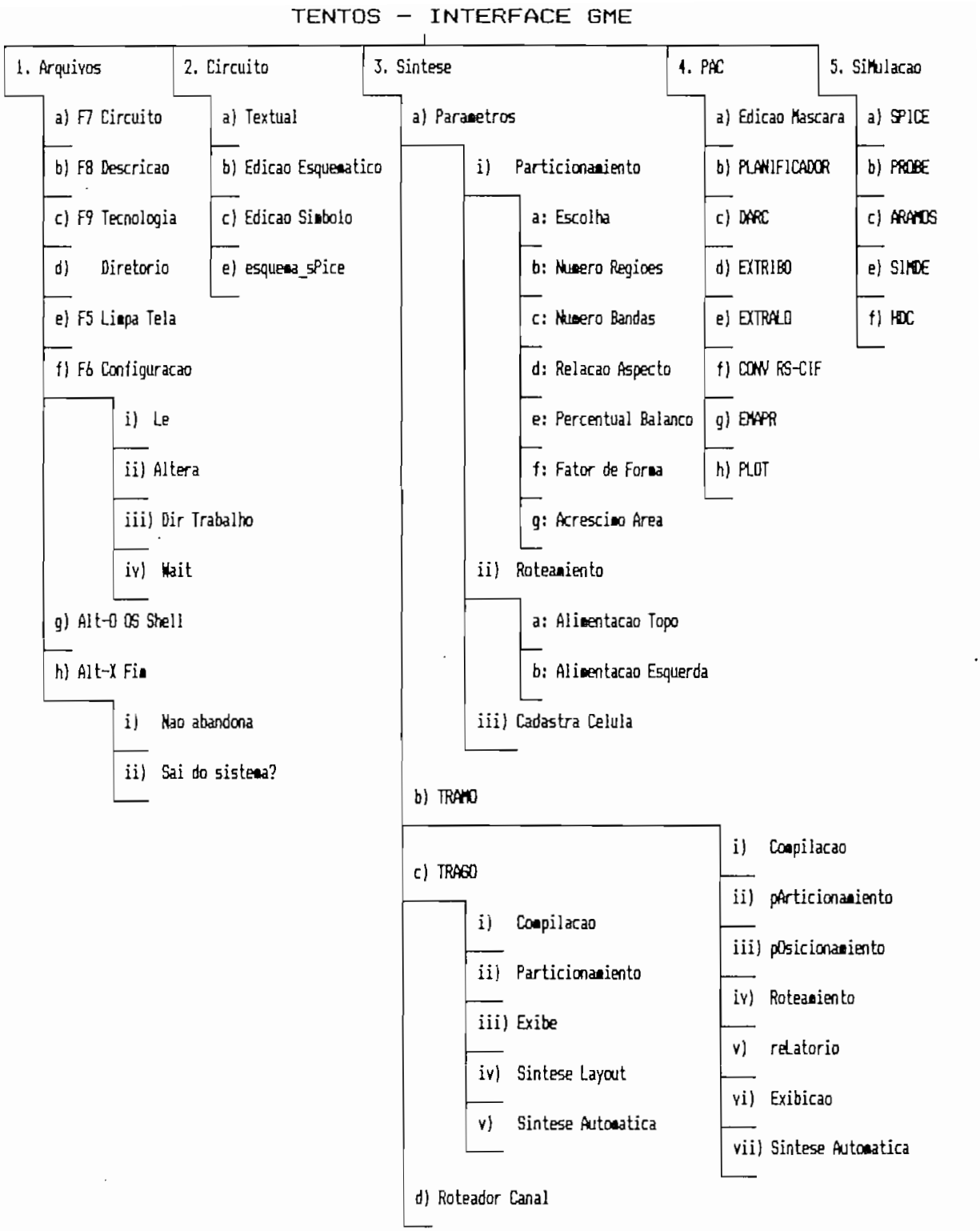


FIGURA 6.6 Distribución jerárquica de los módulos en el interfaz del sistema TENTOS.

NOTA: La terminología empleada en la Fig.6.6 corresponde a la empleada en el original en portugués del sistema TENTOS.

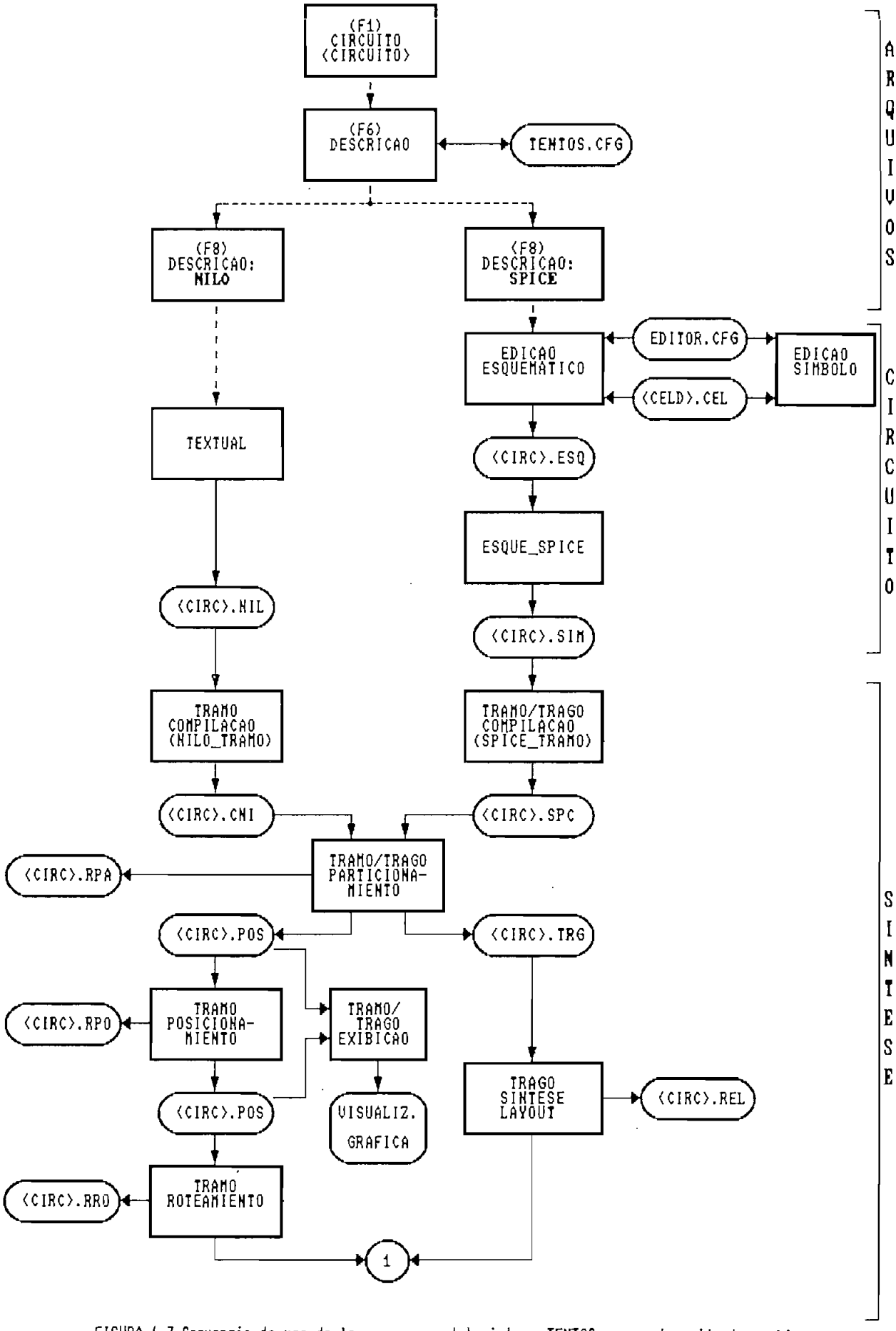


FIGURA 6.7 Secuencia de uso de los programas del sistema TENTOS y concatenación de archivos.

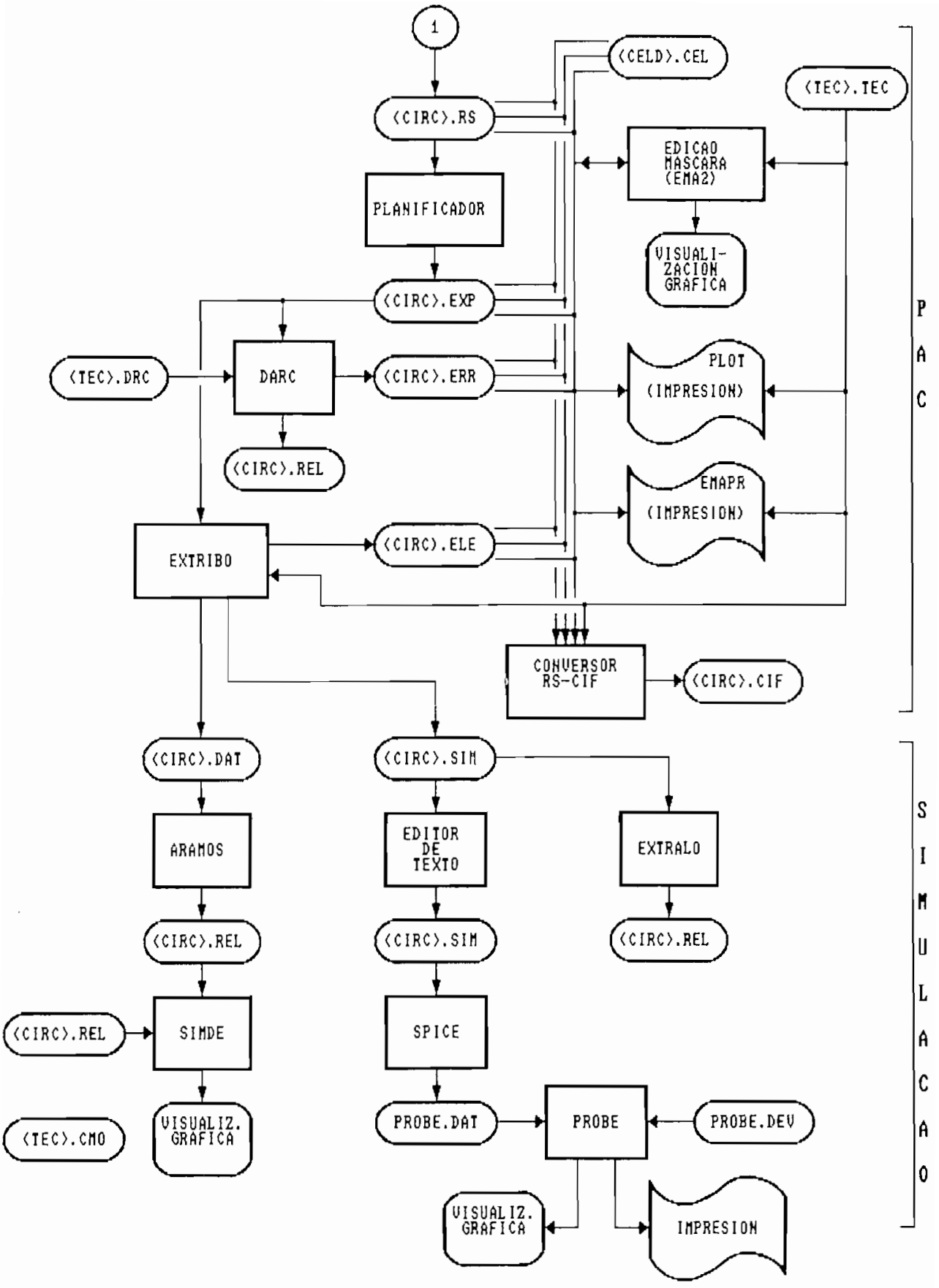


FIGURA 6.7(Cont.) Secuencia de uso de los programas del sistema TENTOS y concatenación de archivos.

b) Acceso directo a los programas del paquete

El interfaz se encarga del acceso a los módulos (herramientas) del paquete suministrando los parámetros que definen su comportamiento. Por tanto, una vez dentro del ambiente del sistema, el diseñador no requiere saber dónde se encuentran las herramientas, bibliotecas de celdas, archivos de tecnología; ni como llamar a los programas ya que todos estos parámetros se hallan previamente especificados en el archivo de configuración del interfaz (Anexo E.1), no sucede así si el diseñador desea invocar individualmente algún programa del paquete desde el sistema operativo (DOS).

Dentro del ambiente del sistema TENTOS la invocación de los programas se reduce a la selección de las opciones en los menúes, que se puede realizar:

i) Apuntando directamente a la opción del menú con el cursor, moviéndolo desde el teclado o mediante "ratón".

ii) Digitando el caracter de identificación asociado a cada ítem, que generalmente se destaca en mayúscula en la palabra de descripción del ítem (Fig.6.6).

iii) Presionando desde cualquier región del árbol de menúes alguna de las teclas de función:<F5> a <F9>, o la combinación <Alt + tecla>, donde <tecla> es el caracter en mayúscula (únicamente en el caso de los menúes principales del sistema).

El interfaz permite al usuario pedir en cualquier momento pequeñas informaciones adicionales de ayuda, mediante la tecla de función <F1>.

OBSERVACION:

En el Capítulo 7 se describe el proceso de diseño completo de un ASIC sencillo en cuyo desarrollo se han empleado las herramientas del sistema TENTOS. Este diseño se puede considerar como un tutorial del uso de los programas que se presentan a continuación.

6.2.3 Configuración del sistema (módulo: 1. Arquivos)

Contiene las operaciones relacionadas con la configuración del ambiente de trabajo, esto es: organización de los archivos de programas del paquete, de los archivos del usuario y opciones generales de funcionamiento del interfaz.

En caso de omitirse esta operación, se asume la configuración contenida en el archivo *<sistema>.CFG* (Anexo E.1)

De acuerdo a la Fig.6.6 los submódulos existentes en este menú son:

a) F7 Circuito

Permite seleccionar el nombre del circuito de trabajo que en adelante será indicado como *<circuito>*. De no indicar-

se nombre alguno se genera una ventana con los archivos de usuario existentes pudiendo seleccionarse uno de ellos (mediante el movimiento del cursor). De omitirse esta opción el interfaz nomina al diseño como: *sennome*.

b) F8 Descricao (Descripción)

Permite seleccionar el formato de descripción del diseño o el tipo de archivo a ser editado, ya sea en forma textual por el editor de texto del lenguage Turbo C, o en forma gráfica (de tratarse de una descripción geométrica) por el editor de máscaras EMA2. Los formatos de descripción posibles, las extensiones correspondientes en los archivos y su función se indican en la Tabla 6.1.

DESCRIPCION	EXTENSION	FUNCION
SPICE	.SIM	Descripción SPICE con extensión definida en *(sistema).CFG*
NILO	.NIL	Descripción en lenguage NILOTRANCA
ESQUEMA	.ESQ	Descripción esquemática
ARAMOS	.DAT	Descripción ARAMOS
CELULA	.CEL	Descripción de celda con extensión definida en *(sistema).CFG*
DRC	.ERR	Archivo de layout generado por el DARC
RS	.RS	Archivo de layout
CIF	.CIF	Archivo de layout en formato CIF
EXPANDIDO	.EXP	Archivo de layout no jerárquico, generado por el Expansor
RELATORIO	.REL	Archivo de documentación

TABLA 6.1 Tipos de descripciones en el sistema TENTOS

c) F9 Tecnología

Este módulo permite seleccionar la tecnología de diseño, cuya información se halla contenida en un archivo: *<tecnolo-*

gía>.TEC (ver Anexo E.3 para la tecnología CMOS15). Las tecnologías disponibles en el sistema TENTOS son:

CMOS12: Tecnología CMOS de un solo nivel de metal de 1.2µ con pozo N

CMOS15: Tecnología CMOS de un solo nivel de metal de 1.5µ con pozo N

CMOS20: Tecnología CMOS de un solo nivel de metal de 2.0µ con pozo N

ECPD12: Tecnología CMOS de dos niveles de metal de 1.2µ con pozo N

ECPD15: Tecnología CMOS de dos niveles de metal de 1.5µ con pozo N

BIPD: Tecnología Bipolar

ECDM: Tecnología CMOS de dos niveles de metal de Grenoble

NOTA: La versión del TENTOS de Abril de 1991 únicamente dispone de biblioteca de celdas definidas para la tecnología CMOS15.

d) Diretorio (Directorio)

Se emplea para seleccionar el directorio del ambiente DOS en el que se hallan contenidos los archivos de trabajo.

e) F5 Limpa Tela (Limpieza de pantalla)

En ciertos casos, luego de la ejecución de algún programa, el interfaz tiene problemas en la reconstrucción de su ventana gráfica, esta opción permite restaurar totalmente la presentación original de la ventana del interfaz.

f) F6 Configuracao (Configuración)

Como se mencionó anteriormente, la información de con-

figuración del ambiente del sistema TENTOS se halla contenida en el archivo: *<sistema>.CFG* (Anexo E.1), esta opción permite accesar a este archivo permitiendo su lectura y redefinición de ser necesario.

De acuerdo a la Fig.6.6 sus opciones son:

i) **Le (Lee):** permite indicar al sistema que se reconfigure según las instrucciones del archivo *<sistema>.CFG*.

ii) **Altera:** se emplea para editar y modificar el archivo de configuración.

iii) **Dir Trabalho (Dir Trabajo):** permite indicar el directorio de lectura y escritura de los archivos del diseño.

iv) **Wait (Espere):** se emplea para indicar el modo de exhibición de los mensajes intermedios del interfaz gráfico, existiendo dos modalidades:

WAIT 0: cuando el mensaje es retenido por pocos segundos en la pantalla y luego es retirado automáticamente.

WAIT 1: cuando el mensaje es retenido indefinidamente en la pantalla hasta que el diseñador oprima una tecla cualquiera.

g) **Alt-O OS Shell**

Permite al diseñador salir temporalmente al sistema operativo DOS, el retorno al interfaz del sistema TENTOS se

realiza digitando el comando EXIT.

h) Alt-X Fim (Fin)

Se emplea para salir definitivamente del sistema de diseño TENTOS pasando previamente por el submenú de confirmación de la instrucción como se indica en la Fig.6.6:

i) Nao abandona (No abandona)

ii) Sai do sistema? (Salir del sistema?).

6.2.4 Captura del diseño (módulo: 2. Circuito)

La descripción del diseño en el sistema TENTOS se realiza al **nivel estructural** (niveles eléctrico y/o lógico). Este módulo posee las herramientas de captura que permiten describir el diseño sea en forma de Texto (Netlist) o Gráfica, para ello cuenta con los siguientes submódulos (Fig.6.6).

a) Textual

Esta opción invoca al editor de textos del lenguage Turbo "C", que permite la elaboración de la NETLIST de descripción textual del circuito, según el formato adoptado en la opción *Arquivos/Descricao*, el diseño puede especificarse en los siguientes lenguages:

i) NILO TRANCA.

ii) SPICE.

i) Lenguage NILOTRANCA

El sistema TENTOS emplea el lenguage lógico de descripción NILO con ciertas adaptaciones al diseño VLSI consistentes en la adición de palabras reservadas empleadas para la asignación de restricciones de ubicación de celdas e interconexiones, que han dado lugar al lenguage de descripción textual: NILOTRANCA que se detalla en el Anexo D.1.

ii) Lenguage de descripción SPICE

La descripción del circuito puede realizarse en forma de NETLIST SPICE que aplicada a circuitos VLSI digitales ha sido detallada anteriormente en el numeral 5.3.

La única modificación consiste en que la determinación de los pines del interfaz y su orientación se especifican al final de la netlist, mediante las palabras reservadas:

```
* interface: <nombre del pin> * orientacao = <orie> *
```

donde <orie> puede ser N (Norte), S (Sur), L (Este *LESTE*) u O (Oeste). Por ejemplo:

```
* CIRCUITO: NAND 2 ENTRADAS

MP1   Out   I1 vcc vcc PMOS L=2.0U W=7.0U
MP2   Out   I2 vcc vcc PMOS L=2.0U W=7.0U
MN3   Out   I1 2   0   NMOS L=3.0U W=4.0U
MN4    2    I2 0   0   NMOS L=3.0U W=4.0U

* interface: I1    *  orientacao=S  *
* interface: I2    *  orientacao=S  *
* interface: Out   *  orientacao=S  *

.END
```

Concatenación de archivos (Fig.6.7)

El diseño descrito mediante el módulo *Circuito/Textual* es grabado por el interfaz del sistema de acuerdo al formato de descripción seleccionado en la opción *Arquivos/Descricao* (ver Tabla 6.1) de la siguiente manera:

i) De seleccionarse el formato NILO el archivo generado es: *<circuito>.NIL*.

ii) De seleccionarse el formato SPICE el archivo generado es: *<circuito>.SIM*

b) Edicao Esquematico (Edición esquemática)[5]

El editor de esquemas (Fig.6.8(a)) permite generar esquemas eléctricos para documentación, simulación eléctrica, simulación logica y síntesis de circuitos integrados. Las funciones basicas del editor son:

i) Carga, movimiento, rotación, reflexión y dimensionamiento de los símbolos contenidos en las bibliotecas.

ii) Operaciones con áreas del esquema: movimiento, copia y remoción.

iii) Inserción de pines de esquema.

iv) Inserción de conexiones: buses y señales simples.

v) Diversas escalas de visualización.

vi) Inserción de textos.

---

[5] "ESQUELETO - Editor de Esquemas Eletricos",D. Sachet - E. Pereira - R. Petry. ANAIS VI Seminário Interno de Microeletrónica, UFRGS-Brásil , pág. 37.

En la definición de textos en el editor de esquemas se deben especificar dos características:

i) **Punto de efecto:** consistente en el punto al que se va a asociar el texto, no coincide necesariamente con el lugar en el que se ha creado el texto.

ii) **Atributo:** indica el tipo de elemento al que se le asigna el texto, es decir: señal de entrada y/o salida, pin de entrada y/o salida, conexión, título de esquema, etc. Existen atributos especiales que asociados a los símbolos de celda y conexiones de red, realizan funciones similares a las **palabras reservadas** en el lenguage NILOTRANCA ya que permiten asignar restricciones para el proceso de síntesis del layout. Estos atributos son:

ii.1) **REGI** : Región del símbolo

ii.2) **BAND** : Banda del símbolo

ii.3) **ESPE** : Reflexión (*espelhamento*) del símbolo

ii.4) **ORDE** : Orden de conexión.

ii.5) **PESO** : Peso de conexión

ii.6) **TRIL** : Fila de conexión.

ii.7) **ORIE** : Orientación de conexión.

ii.8) **PESQ** : Pin de esquema.

ii.9) **SIG:** Nombre de nodo.

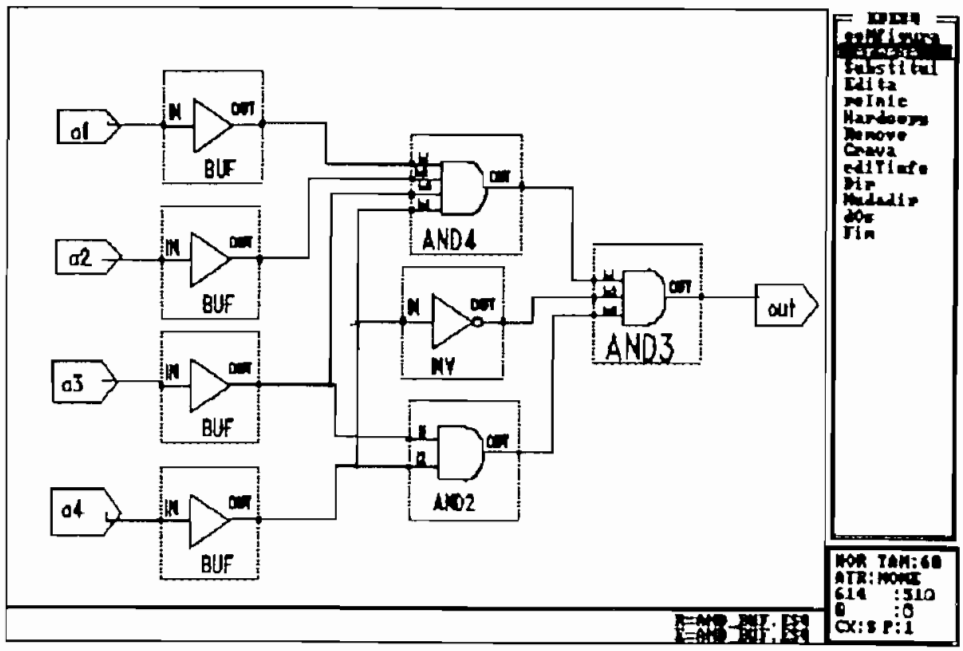
ii.7) **TITL:** Título del esquema.

Concatenación de archivos (Fig.6.7)

El editor esquemático emplea para su configuración el archivo *EDITOR.CFG* (Anexo E.2), Los símbolos empleados en la

descripción de los esquemas se leen de los archivos *<celda>.SMB* de la biblioteca del TENTOS. Y el esquema del circuito es grabado en el archivo *<circuito>.ESQ*

(a)



(b)

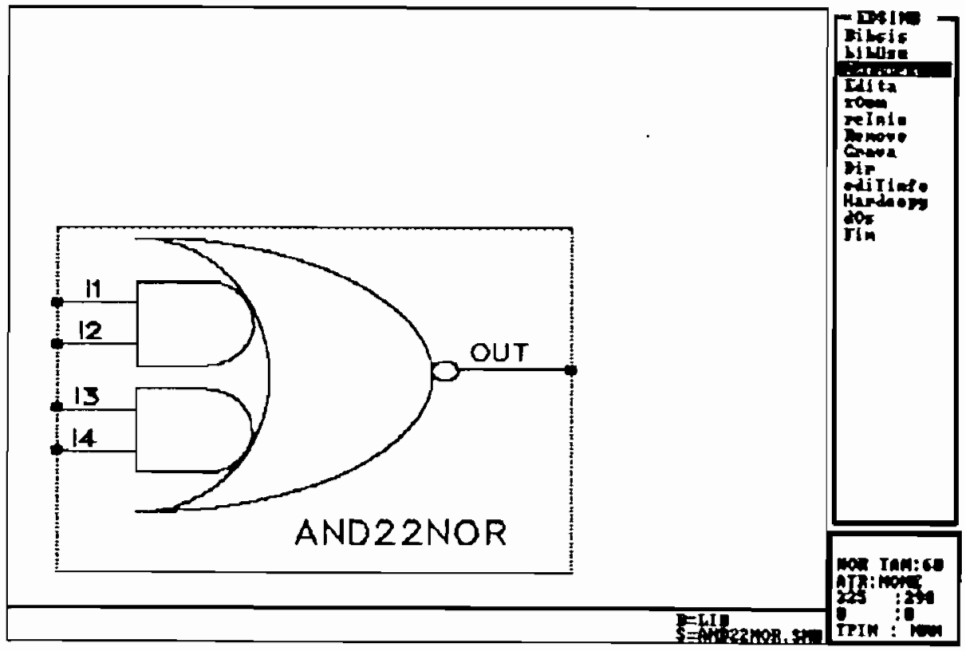


FIGURA 6.8 (a) Interfaz del Editor de Esquemas.
(b) Interfaz del Editor de Simbolos.

c) Edicao Simbolo (Edición de simbolos)

El editor de símbolos (Fig.6.8(b)) permite la generación de los símbolos gráficos que utiliza el editor de esquemas. Estos símbolos representan elementos eléctricos, electrónicos y compuertas lógicas, que permiten al diseñador describir su proyecto en varios niveles de abstracción.

Todo símbolo es identificado por un nombre y es caracterizado por un conjunto de elementos definidos en su creación. Estos elementos son:

i) **Nombre:** texto que identifica al símbolo.
ii) **Marco:** rectángulo que define la frontera del símbolo
iii) **Pines:** define puntos de entrada/salida del simbolo.
iv) **Textos:** son campos alfanuméricos asociados a atributos que contienen informaciones para documentación y post-procesamiento.
v) **Descripción gráfica:** lista de primitivas gráficas que describen al símbolo.

Concatenación de archivos (Fig.6.7)

Para su ejecución requiere del archivo *EDITOR.CFG* (Anexo E.2). Los símbolos diseñados se graban en el archivo *<celda>.SMB* y pasan a formar parte de la biblioteca de símbolos.

e) Esquema_sPice

Al margen del método empleado en la generación del diseño (textual o esquemático), los algoritmos de síntesis del layout y simulación del diseño; requieren una descripción unificada de éste. Dentro de este contexto, el postprocesador Esquema_sPice transforma automáticamente la descripción esquemática de un circuito generada en el *Editor de Esquemas* en una descripción textual con el formato de *NETLIST SPICE*.

Concatenación de archivos (Fig.6.7)

El submódulo Esquema_sPice lee la descripción esquemática del diseño contenida en *<circuito>.ESQ* y la transforma en un NETLIST SPICE que graba en el archivo *<circuito>.SIM*

6.2.5 Síntesis del layout (módulo: 3. Sintese)

Dentro de este módulo se agrupan los programas que sintetizan el circuito, es decir, partiendo de la descripción a nivel estructural generan su descripción a nivel físico. Existen dos métodos de síntesis: TRAMO (*Tranca Automatic Module Generator*) y TRAGO (*Tranca Automatic Generator*) en que los layouts resultantes, cuyos planos de base se indican en las Figuras.6.10 y 6.12 respectivamente, tienen características propias como se verá posteriormente.

De acuerdo a la Fig.6.6, los submódulos de sintetización del layout son:

a) Parámetros

Permite el establecimiento de los parámetros para los procesos de ubicación (particionamiento y enrutamiento) de celdas de los sintetizadores (b)TRAMO y (c)TRAGO, y la caracterización de las celdas existentes así como de las que se van añadiendo a la biblioteca de celdas.

De acuerdo a la (Fig.6.6), sus opciones son:

i) Particionamiento

ii) Roteamiento (Enrutamiento)

iii) Cadastra Celula (Caracterización de las celdas)

i) Particionamiento

El proceso de particionamiento y por tanto sus parámetros, que se analizarán posteriormente, son comunes a los sintetizadores TRAMO y TRAGO, estos parámetros consisten en una serie de variables empleadas por los algoritmos de distribución de celdas en el plano de base (Figuras 6.10 y 6.12), y además definen los criterios prioritarios a adoptarse (optimización de área o de enrutamiento) para optar por una solución particular al problema del particionamiento. Estos parámetros son los que en última instancia determinan la configuración global del layout del diseño.

En el Anexo D.2 se realiza un estudio detallado de los parámetros de particionamiento, a continuación se indican los parámetros que el sistema adopta por omisión:

a: Escolha: Individual

b: Número Regioes: 1

c: Número Bandas: Calculado

d: Relacao Aspecto: 1.000

e: Percentual Balanco: 50%

f: Fator de forma: 1.000

g: Acrescimo Area: 0.000

ii) Roteamiento (Enrutamiento)

La alimentación de las bandas se realiza a través de sus bordes izquierdo y derecho, y de sus filas primera y última como se indica en las Figuras 6.10 y 6.12. La forma en que se distribuyen las señales de alimentación (Vcc y Gnd) en las bandas del circuito se define mediante la selección de los parámetros:

a: Alimentacao Topo (Alimentación Superior): que puede ser Vcc o Gnd

b: Alimentacao Esquerda (Alimentación Izquierda): que puede ser Vcc o Gnd

iii) Cadastra celula (Caracterización de celdas)

El módulo de caracterización de las celdas se emplea para mantener actualizado el archivo de biblioteca de celdas. Presenta una ventana de interfaz propia (Fig.6.9) que permite incluir, alterar, consultar y excluir celdas de biblioteca.

Cada celda existente en la biblioteca es caracterizada mediante las siguientes especificaciones:

iii.1) **Nombre:** Es único para cada celda, y permite invocarla en las descripciones a diferentes niveles

iii.2) Número de entradas.

iii.3) Número de salidas.

iii.4) Largo.

iii.5) Tipo y posición de cada pin.

iii.6) **Nombre externo:** Que corresponde al nombre del archivo que contiene la descripción geométrica de la celda.

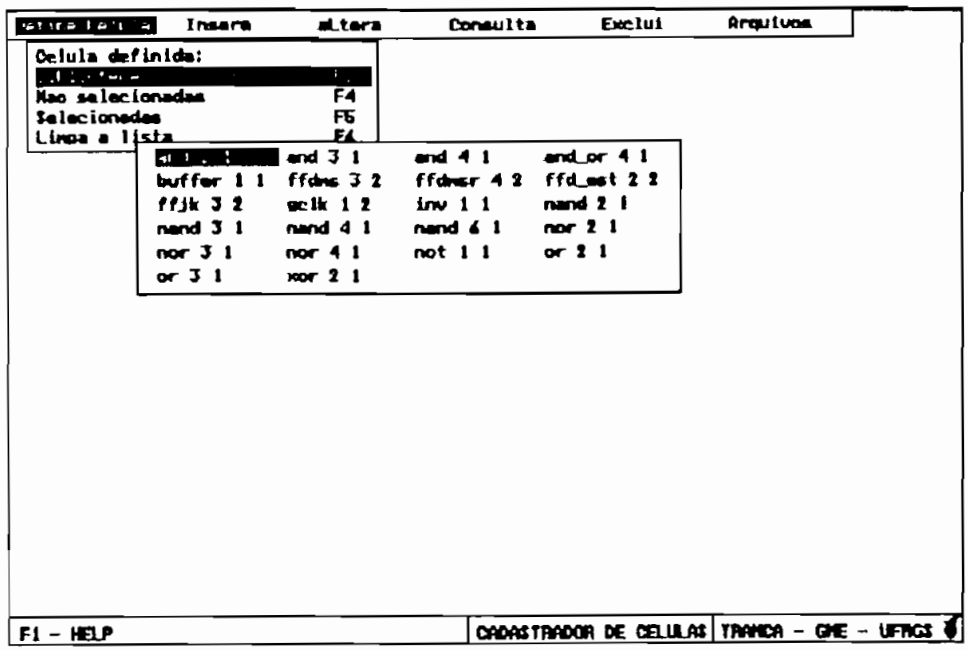


FIGURA 6.9 Interfaz gráfico del caracterizador de celdas.

b) TRAMO (*Tranca Automatic Module Generator*)

Permite la síntesis automática del layout para circuitos en lógica aleatoria (compuertas lógicas) de acuerdo con el plano de base de la Fig.6.10.

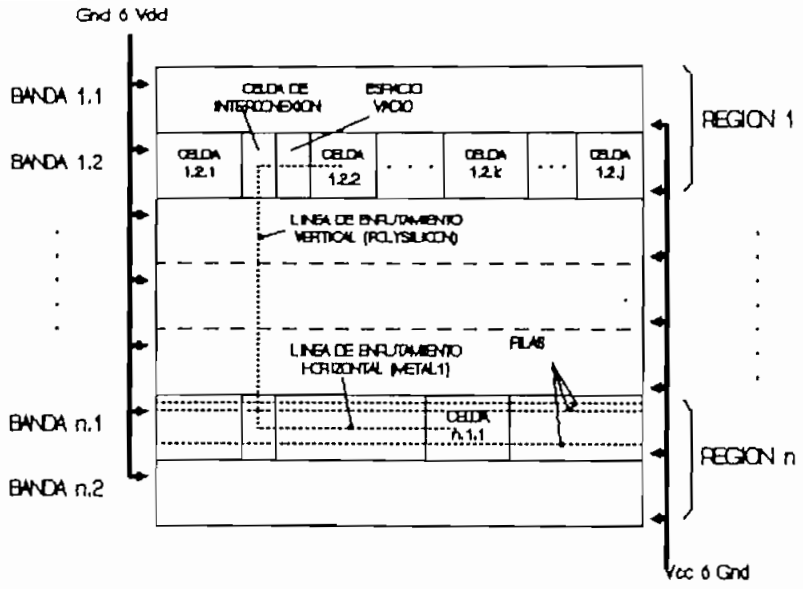


FIGURA 6.10 Plano de Base del layout característico de la síntesis TRAMO.

Las características propias del layout generado por el TRAMO son:

i) Las celdas que se emplean tienen altura constante y su layout se halla predefinido en una biblioteca.

ii) La geometría de las celdas de biblioteca es definida en base a *filas* por lo que se dice que tienen *transparencia horizontal*.

iii) El enrutamiento lateral se realiza por las filas libres y el vertical se realiza por las celdas de interconexión como se verá posteriormente.

Las opciones que presenta el sintetizador TRAMO (Fig.6.6) son:

i) Compilacao (Compilación)

ii) pArticionamiento.

iii) pOsicionamiento.

iv) Roteamiento (Enrutamiento)

v) reLatorio (Documentación)

vi) Exibicao (Exhibición)

vii) Sintese Automatica (Síntesis Automática)

i) Compilacao (Compilación)

Es el primer paso dentro del proceso de síntesis de un circuito, consiste en la extracción y codificación de las informaciones descritas en lenguage de alto nivel, en archivos intermedios a través de los que se comunica con los procesos posteriores.

Los lenguages de alto nivel para la descripción de la NETLIST del circuito son NILOTRANCA y SPICE (según el formato seleccionado en la opción *Arquivos/Descricao*), dependiendo del lenguage seleccionado, el interfaz invocará automáticamente al compilador NILO-TRAMO o al compilador SPICE-TRAMO, según el caso.

Concatenación de archivos (Fig.6.7)

i) De emplearse el compilador NILO-TRAMO, el archivo de descripción del circuito es *<circuito>.NIL* y el archivo que el compilador genera es *<circuito>.CNI*.

ii) De emplearse el compilador SPICE-TRAMO, la descripción del circuito se lee del archivo *⟨circuito⟩.SIM* y el compilador genera el archivo *⟨circuito⟩.SPC*.

ii) Particionamiento

El particionamiento (ó *posicionamiento relativo*) se encarga de la distribución de las celdas de un circuito en *bandas* y regiones (Fig.6.10). Este particionamiento se realiza en base a los siguientes criterios generales:

ii.1) La configuración global de la distribución de celdas en bandas y regiones es regida por los parámetros de particionamiento.

ii.2) La configuración puntual de la distribución de celdas en bandas y regiones, es regida por las restricciones consignadas en la descripción del circuito.

ii.3) Las celdas que son enlazadas a través de una red se ubican preferentemente en la misma banda.

ii.4) Las bandas son subdivididas en ***regiones*** que tienden a aglutinar las celdas de conectividad afín, con lo que se evita la sobrecarga de las redes de enrutamiento en cada banda.

El algoritmo desarrollado para la solución del problema

se basa en la asignación de *pesos* diferenciados a cada celda, los pesos son evaluados a partir de los criterios descritos anteriormente, luego según los pesos asignados se distribuyen las celdas en una banda y un bloque residual que es nuevamente particionado, el proceso es iterativo hasta la generación de la última banda.

Concatenación de archivos (Fig.6.7)

Los archivos de entrada al particionador son:

i) *<circuito>.CNI* si la compilación se hizo con el NILO-TRAMO

ii) *<circuito>.SPC* si la compilación se hizo con el SPICE-TRAMO

Los archivos que el particionador genera son:

i) *<circuito>.POS*: archivo intermediario para el posicionamiento TRAMO, posee además la información que es leída por el submenú de exhibición gráfica: *Exibicao*

ii) *<circuito>.TRG*: archivo intermediario para la síntesis TRAGO.

iii) *<circuito>.RPA*: archivo de documentación para el usuario, contiene los resultados del particionamiento.

ii) Posicionamiento

El posicionamiento *intra-banda* consiste en la ubicación absoluta de las celdas en sus respectivas bandas (Fig.6.10) a partir de la distribución parcial realizada por el particionador.

El posicionamiento es realizado según el siguiente algoritmo:

ii.1) El posicionamiento de las bandas se realiza en orden ascendente según el largo de cada banda.

ii.2) Cada celda recibe un *peso* cuyo valor dependerá de la orientación de los pines de interfaz, de las restricciones asignadas a cada celda y de las celdas vecinas con las que se interconecta.

ii.3) Las celdas se van posicionando dentro de sus bandas, en orden descendente según el valor de su peso resultante.

ii.4) Finalmente se determina la orientación de las celdas, buscando minimizar el largo de las interconexiones entre celdas vecinas.

En la ejecución de su algoritmo el posicionador siempre da prioridad a la atención de las restricciones de interfaz, minimización de área, y minimización de las interconexiones globales del circuito.

Concatenación de archivos (Fig.6.7)

El archivo de entrada al posicionador es *‹circuito›.POS* generado por el particionador. Los archivos de salida son:

i) *‹circuito›.POS*: archivo intermediario para el proceso

posterior de enrutamiento TRAMO, posee además la información para el submenú de exhibición gráfica: *Exibicao*

ii) *<circuito>.RPO*: archivo de documentación para el usuario, contiene los resultados del posicionamiento.

iv) Roteamiento (Enrutamiento)

El enrutamiento entre celdas es la fase de síntesis del layout, responsable de la realización física de las redes de conexión dentro y entre bandas. Según su orientación existen dos tipos de redes de enrutamiento como se indica en la Fig.6.10:

iv.1) **Horizontales:** cuando recorren a través de las **filas** no ocupadas por el layout propio de la celda, estas redes se implementan en la capa de METAL 1.

iv.2) **Verticales:** cuando recorren a través de ***celdas de interconexión*** que para el efecto se intercalan entre las celdas funcionales, estas redes se realizan en la capa de POLYSILICON.

Los algoritmos de enrutamiento implican dos limitaciones:

iv.1) La apertura de los espacios para intercalar las celdas de interconexión determina un desperdicio de área inevitable, que surge como consecuencia directa de la falta de ***transparencia vertical*** y la no utilización de la máscara de METAL 2 en la definición de los layouts de las celdas de biblioteca.

iv.2) El enrutamiento horizontal a través de las *filas* libres es limitado puesto que de no encontrarse estas, las líneas de enrutamiento no se forman por lo que la eficiencia del enrutamiento en layouts complejos es *menor al 100%* (como se evidenciará en el Capítulo 10)

Estas limitaciones en el sistema TRAMO están siendo objeto de estudio, existen al momento propuestas concretas para la depuración de este sistema que buscan superar precisamente las deficiencias aquí indicadas (6).

Concatenación de archivos (Fig.6.7)

Su archivo de entrada es *<circuito>.POS* generado por el posicionador.

El enrutador genera los archivos:

i) *<circuito>.RS*: descripción geométrica del layout jerárquico del circuito.

ii) *<circuito>.RRO*:archivo de documentación para el usuario con los resultados del enrutamiento.

v) Relatorio (Documentación)

En esta opción se invoca al editor de textos del sistema TENTOS, en el que se carga automáticamente los archivos de

(6) "TRAMO II: PROPOSTA PARA UM GERADOR DE LEIAUTE BASEADO EN CÉLULAS PARA CIRCUITOS CMOS DIGITAIS COM DOIS NIVEIS DE METAL", A. Reis - R. Reis, VI Simposio Brasileiro de Concepção de Circuitos Integrados - Brasil, pág.90.

documentación *<circuito>.RPA*, *<circuito>.RPO* ó *<circuito>.RRO* ,que contienen en formato de texto los resultados de los procesos de particionamiento, posicionamiento o enrutamiento. respectivamente. El archivo leído por el editor de textos corresponde a la documentación del último proceso ejecutado.

vi) Exibicao (Exhibición)

La opción de exhibición gráfica invoca al programa EXTRAMO que, mediante una ventana propia (Fig.6.11), grafica la distribución de las celdas resultante de los procesos de particionamiento y/o posicionamiento del circuito. Esta información se halla contenida en el archivo *<circuito>.POS*, y en base a ella el usuario puede decidir si realizar o no una nueva síntesis del circuito variando los parámetros.

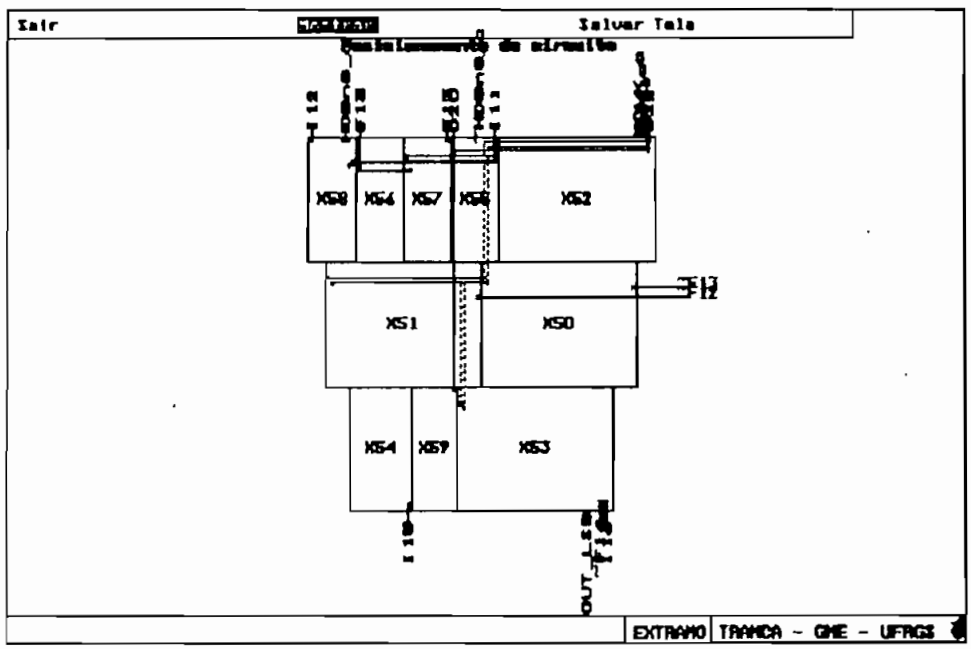


FIGURA 6.11 Ventana de exhibición gráfica del programa EXTRAMO.

vii) Sintese Automatica (Síntesis Automática)

Cada una de las etapas de la síntesis del circuito, puede ejecutarse individualmente (paso a paso). No obstante, la opción *Síntesis Automática* permite realizar el proceso completo de síntesis del diseño, iniciando con la descripción lógica del circuito y terminando con la generación de su layout, sin detenerse al pasar de un proceso al otro.

c) TRAGO (*Tranca Automatic Generator*)[7]

Es un sistema de síntesis automática de layout para circuitos en lógica aleatoria de acuerdo con el plano de base de la Fig.6.12.

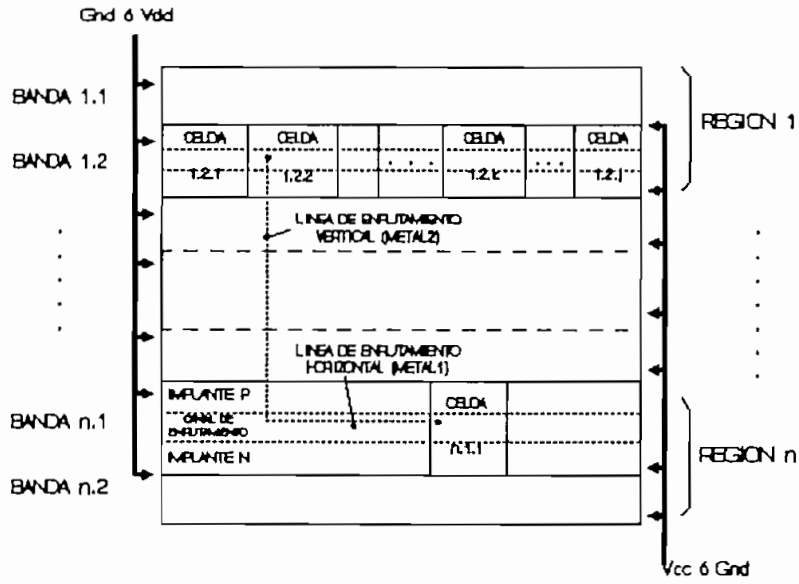


FIGURA 6.12 Plano de Base del layout característico de la síntesis TRAGO.

[7] "MANUAL DO USUARIO DO PROJETO TRANCA V1.0", F. Moraes - M. Lubaszewski - R. Reis.

Las características particulares del layout generado por el TRAGO son:

i) Las celdas del diseño no provienen de una biblioteca predefinida, sino que son ***pregeneradas*** durante el proceso de ***compilación.***

ii) Las bandas se dividen en tres sectores:

ii.1) zona de IMPLANTE P de altura fija sobre la que se forman los transistores PMOS durante el proceso de ***síntesis de layout.***

ii.2) zona de IMPLANTE N de altura fija sobre la que se forman los transistores NMOS,

ii.3) canal de enrutamiento horizontal, dentro del que se realiza el enrutamiento horizontal entre celdas en base a METAL1, su altura es variable dependiendo de la cantidad de líneas de enrutamiento que posea,

debido a la altura variable de la zona de enrutamiento horizontal, la altura final de las bandas es también variable.

iii) El enrutamiento vertical se realiza en METAL2 por sobre la celdas por lo que no requiere intercalar celdas de interconexión.

iv) Las celdas dentro de una misma banda tienen la misma altura.

De acuerdo a la Fig.6.6 las opciones que presenta el sintetizador TRAGO son:

i) Compilacao (Compilación).

ii) Particionamiento.

iii) Exhibe (Exhibición).

iv) Sintese Layout (Síntesis del Layout).

v) Sintese Automatica (Síntesis Automática).

i) Compilación

Puesto que la síntesis TRAGO no depende de la biblioteca de celdas, en la compilación se realiza un proceso de ***pregeneración*** en el que de acuerdo a la netlist que describe a la celda, esta es descompuesta en celdas elementales (NAND, NOR, INV), especificándose los enlaces entre estas y los terminales de entrada/salida de la celda final.

A diferencia de la síntesis TRAMO, el único lenguage posible para la descripción de la NETLIST del circuito en la síntesis TRAGO es el lenguage SPICE (no es factible la descripción en lenguage NILOTANCA) puesto que únicamente el compilador SPICE-TRAGO ha sido desarrollado para realizar el proceso de pregeneración de celdas.

Concatenación de archivos (Fig.6.7)

El archivo de entrada al compilador TRAGO corresponde a la descripción a nivel lógico del circuito contenida en el archivo *<circuito>.SIM*. Los resultados de la compilación se graban en el archivo *<circuito>.SPC*

ii) Particionamiento

El proceso de particionamiento de celdas es el mismo que el de la síntesis TRAMO, La única diferencia radica en que en este caso se posicionan las celdas pregeneradas en la compilación, en la síntesis TRAMO en cambio se posicionan las celdas predefinidas en biblioteca.

Concatenación de archivos (Fig.6.7)

En este caso la información de entrada al particionador únicamente proviene del archivo *<circuito>.SPC* ya que la compilación SPICE-TRAMO es la única factible.

Los archivos generados por el particionador son los mismos que se indicaron anteriormente en la síntesis TRAMO.

iii) Exibe (Exhibición)

El interfaz para la exhibición gráfica de los resultados de particionamiento/posicionamiento del sistema TRAMO (Fig.6.10), también es usado para la exhibición de los resultados del particionamiento en el sistema TRAGO.

iv) Sintese Layout (Síntesis del Layout)

El proceso de síntesis del layout del sistema TRAGO se realiza en función del plano de base de la Fig.6.12 y tiene dos etapas:

i) Las celdas pregeneradas en la compilación y distribuídas en el posicionamiento son interconectadas horizontalmen-

te en METAL 1 y verticalmente en METAL2, como se indicó anteriormente.

ii) Los transistores de las celdas básicas son formados generando ZONAS ACTIVAS sobre los sectores de IMPLANTE N y P de la banda, luego se forman las compuertas en POLYSILICON con la particularidad de que todos los transistores son posicionados horizontalmente. La densidad media de transistores resultante de este proceso es de 650 transistores/mm^2.

La capacidad de reajuste de la altura de las bandas para dar cabida a las interconexiones verticales permite una eficiencia de enrutamiento del 100%. Además la ausencia de celdas de interconexión permite un mejor aprovechamiento del área de diseño con respecto al TRAMO.

Concatenación de archivos (Fig.6.7)

El archivo de entrada al proceso de síntesis TRAGO es *⟨circuito⟩.TRG*.

Los archivos de salida son:

i) *⟨circuito⟩.RS*: archivo que contiene la descripción gráfica del layout generado.

ii) *⟨circuito⟩.REL*: archivo de documentación de los resultados de la síntesis TRAGO.

v) Sintese Automatica (Síntesis Automática)

Al igual que en el TRAMO, los pasos de síntesis del circuito pueden ser ejecutados individualmente o se puede realizar automáticamente todo el proceso mediante la ejecución de esta opción.

d) Roteador Canal (Enrutamiento de Canal)[8]

Es una herramienta en proceso de desarrollo. Considerando que el enrutamiento manual de módulos es un proceso largo y sujeto a errores, el objetivo de esta herramienta es realizar simbólicamente la conexión automática de pines en una región rectangular (pines interiores, exteriores y laterales) sin preocuparse de las reglas tecnológicas. Al momento han sido concluídos los programas de generación de máscaras de enrutamiento, faltando los programas de posicionamiento de los módulos a ser interconectados.

6.2.6 Diseño a nivel físico (módulo: 4. PAC)

Este módulo realiza el procesamiento y postprocesamiento del layout generado en el módulo de Síntesis, esto es: de la edición gráfica del layout (en forma de video y/o impresa), y de la generación, y verificación de las *netlists* equivalentes para los procesos de simulación posteriores.

---

[8] "SCHAROP - Um Roteador Detalhado de Canal", C. Pereira - D Couto, pág 849.

Los programas de procesamiento gráfico del layout del circuito del sistema TENTOS se caracterizan por emplear un formato de descripción geométrica propio denominado lenguage RS, similar al formato CIF, que se estudia en el Anexo D.3

De acuerdo a la Fig.6.6 los submódulos del módulo FAC son:

a) Edicao Mascara (Edición de máscaras)(9)

Este submódulo permite invocar al editor de máscaras jerárquico EMA2 cuyo interfaz gráfico se exhibe en la Fig.6.13. Este programa realiza la exhibición y modificación de layouts de circuitos integrados descritos en formato RS.

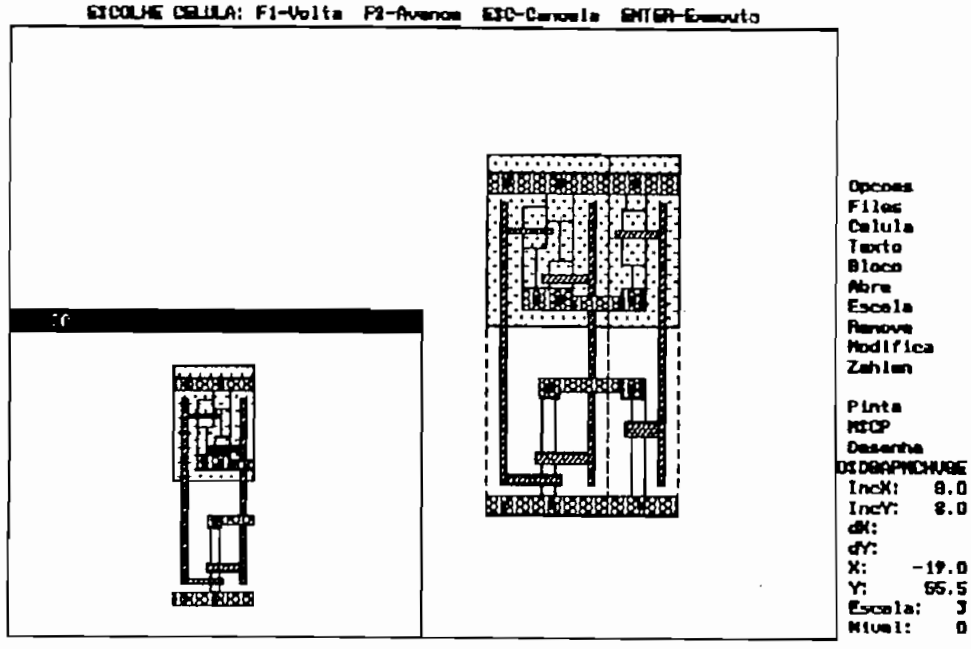


FIGURA 6.13 Interfaz gráfico del Editor de máscaras.

(9) "EDITOR DE MÁSCARAS EMA2 INTERFACE GRAFICA DO EXTRATOR", M. Steiner.

El editor de máscaras EMA2 es una herramienta de CAD que permite realizar operaciones gráficas de alto nivel. Sus características básicas son las siguientes:

i) Su única primitiva gráfica es el rectángulo, mismo que puede ser generado, modificado o removido de cualquier sector del layout.

ii) Permite la expansión y reducción de espacio físico que puede ser empleado para la inclusión o remoción de primitivas geométricas.

iii) Permite realizar operaciones a nivel de bloque, esto es: copiar, mover, eliminar o definir nuevas celdas a partir de las primitivas encerradas en el bloque marcado.

iv) Permite cargar en memoria varias celdas desde la biblioteca, o definir nuevas celdas.

v) Su poderío radica en que permite trabajar simultáneamente con todas las celdas presentes en RAM del computador, realizando operaciones de traslación, reflexión, y copia sobre layouts de celdas mayores. Este proceso puede ser recursivo, es decir, a partir de celdas simples se pueden generar celdas mayores, tantas veces cuantas la memoria del equipo de computación lo permita.

vi) Permite diseñar en todas las máscaras del layout.

vii) Permite exhibir, pintar y numerar selectivamente las diferentes máscaras que forman el layout del circuito. La opción de numeración es de especial utilidad para:

vii.1) exhibir los nodos que en el layout corresponden a la *netlist* de su circuito equivalente generado por el EXTRIBO, y

vii.2) editar las primitivas que poseen violaciones de reglas geométricas en el layout generado por el DARC (verificador de reglas de diseño).

viii) Es factible editar el layout en todas las escalas posibles.

Concatenación de archivos (Fig.6.7)

Para su ejecución requiere los archivos:

i) *⟨tecnología⟩.TEC*: que contiene detalles de edición gráfica tales como: colores asignados a cada máscara y escalas de las primitivas geométricas del archivo RS (ver Anexo E.3 para el caso de la tecnología CMOS 1.5μ)

ii) *⟨circuito⟩.RS, .EXP, .ERR, .ELE* y *⟨celdas⟩.CEL*, y cualquier archivo de descripción de layout (jerárquico o plano) que se halle en formato RS.

"El editor de máscaras no permite editar layouts en formato CIF debiendo estos archivos previamente transformarse a formato RS".

Su dispositivo de salida es la pantalla del computador. El archivo de salida contiene las modificaciones realizadas en el layout, su nombre es requerido por el programa al usuario, de ser omitido, el programa asume el mismo nombre del archivo entrante. Su dispositivo de salida

b) PLANIFICADOR

Algunos programas del módulo PAC únicamente admiten primitivas geométricas descritas en coordenadas absolutas (con respecto a un solo origen) ya que sus algoritmos no son capaces de trabajar con definiciones de celdas cuyas coordenadas son relativas al origen de la grilla de cada celda.

El planificador o expansor, es un preprocesador para estos programas puesto que convierte un layout jerárquico (definido en base a celdas) en plano. Para ello transforma las coordenadas relativas de todas las primitivas geométricas de las celdas en coordenadas absolutas (con respecto a un origen único coincidente con el origen de la celda de mayor jerarquía del layout).

Concatenación de archivos (Fig.6.7)

El expansor emplea como entrada la descripción del layout jerárquico contenido en el archivo *<circuito>.RS* y genera la descripción del layout plano en el archivo *<circuito>.EXP*.

c) DARC

Como se indicó anteriormente, todo diseño de un layout debe cumnplir con un conjunto de reglas de diseño propias de cada fabricante.

El programa DARC (*Design Automatic Rule Checker*) verifica sobre el layout del circuito el cumplimiento de las reglas geométricas de diseño en base a un conjunto de normas predefinidas. De detectar violaciones de reglas las marca sobre el layout y las lista en un archivo de documentación.

Concatenación de archivos (Fig.6.7)

El programa DARC requiere para su ejecución de los archivos:

i) *<tecnologia>.DRC*, que contiene las reglas a ser verificadas (en el Anexo E.4 se indica el archivo CMOS15.DRC definido según las reglas de diseño de la empresa ES2)

ii) *<circuito>.EXP*, que contiene la descripción del layout plano ó expandido del circuito diseñado.

Los resultados del DARC se generan en los archivos:

i) *<circuito>.REL*, en el que se describen las violaciones de reglas detectadas y las coordenadas de las primitivas geométricas involucradas en tales violaciones.

ii) *⟨circuito⟩.ERR*, consistente en un layout en formato RS en el que se numeran las primitivas geométricas con violaciones de error.

d) EXTRIBO[10]

Recordando lo indicado anteriormente: además de la verificación de reglas geométricas, todo layout debe ser sometido a una verificación de su estructura circuital y de su funcionalidad lógica (simulación).

Estas verificaciones requieren del circuito eléctrico equivalente del layout del diseño. Para ello el programa EXTRIBO detecta y numera los transistores y nodos en el layout y en base a ellos genera la descripción textual del circuito equivalente en forma de *NETLIST*.

Concatenación de archivos (Fig.6.7)

Los archivos de entrada al EXTRIBO son:

i) *⟨circuito⟩.EXP*: contiene la descripción expandida del layout del circuito.

ii) *⟨tecnologia⟩.TEC*: contiene los parámetros que definen los modelos físicos de transistores, características

---

[10] "EXTRIBO: UMA VERSAO CORRIGIDA E MELHORADA DO EXTRATOR HIERARQUICO DE CIRCUITOS". M. Steeer - R. Reis. ANAIS VI Seminario Interno de Microelectrónica. UFRGS - Brasil.

resistivas, capacitivas y reglas de conectividad de las capas del circuito.

Los arhivos de salida de este programa son:

i) *⟨circuito⟩.DAT* ó *⟨circuito⟩.SIM*: en este archivo se graba la *NETLIST* del circuito equivalente del layout, que puede hallarse en formato ARAMOS o SPICE respectivamente, según lo indique el archivo de configuración del sistema *TENTOS.CFG* (Anexo E.1).

ii) *⟨circuito⟩.ELE*: contiene el layout del circuito en formato RS en cuyas primitivas geométricas se consigna la numeración de nodos y de transistores correspondiente a la *netlist* del circuito equivalente.

e) EXTRALO

Así como el DARC verifica sobre el layout el cumplimiento de las reglas geométricas de diseño, independientemente de su estructura circuital, el EXTRALO en cambio analiza la estructura circuital del layout, sin ocuparse de las reglas geométricas, esto es:

i) Detecta en el circuito equivalente del layout del circuito todas las celdas básicas contenidas.

ii) Analiza las características topológicas de cada celda detectada, esto es: identifica los nodos de entrada y salida de las celdas, y las otras celdas con las que ésta se halla interconectada.

iii) Organiza los transistores en pares complementarios (en la tecnología CMOS), y alerta al diseñador en caso de detectar transistores "sueltos" es decir: transistores sin sus respectivos complementarios.

Concatenación de archivos (Fig.6.7)

Para su ejecución requiere del archivo *<circuito>.SIM* en el que se halla la netlist SPICE del circuito equivalente del layout, y genera sus resultados en el archivo *<circuito>.REL*.

f) CONV RS-CIF

Puesto que el formato de descripción geométrica CIF es uno de los formatos adoptados por la mayor parte de fundidoras de Silicio para el intercambio de diseños, es indispensable transformar las descripciones de los layouts generados en formato RS al formato CIF.

La opción CONV RS-CIF se encarga de transformar automáticamente cualquier descripción en formato RS a su equivalente en formato CIF.

Concatenación de archivos (Fig.6.7)

Los archivos de entrada a este programa son:

i) *⟨tecnologia⟩.TEC*,en el que se hallan definidas las equivalencias de capas entre los formatos RS y CIF.

ii) *⟨circuito⟩.RS*, *.EXP*, *.ERR*, *.ELE* y *⟨celdas⟩.CEL*, y todo archivo que contenga un layout descrito en formato RS.

El programa CONV RS-CIF genera la descripción CIF, equivalente a la descripción RS del archivo entrante en un archivo del mismo nombre con la extensión *.CIF*

g) EMAPR y h) PLOT

Estos dos programas se emplean para imprimir el layout del circuito integrado descrito en formato RS.

La impresión y sombreado de las capas de diseño del layout pueden ser realizadas en forma selectiva

Las diferencias básicas entre estos dos programas son las siguientes:

i) El PLOT permite la impresión de layouts tanto jerárquicos como planos, en cambio el programa EMAPR únicamente permite la impresión de layouts planos ó expandidos.

ii) El PLOT permite la impresión de los layouts a cualquier escala dada por el usuario, de no tener la impresora capacidad para imprimir layouts demasiado grandes estos se generan por partes. El EMAPR en cambio nunca excede la capacidad de tamaño de la impresora, por ello pide al usuario los márgenes superior, inferior y laterales de la hoja y adapta internamente las escalas al espacio disponible.

Concatenación de archivos (Fig.6.7)

Los archivos de entrada a estos programas son:

i) *<tecnologia>.TEC* en el que se hallan los patrones de sombreado de cada capa.

ii) *<circuito>.RS, .EXP, .ERR, .ELE y <celdas>.CEL*, y todo archivo que contenga un layout descrito en formato RS.

El dispositivo de salida es el impresor.

6.2.7 Simulación circuital (submódulo: 5. Simulacao)

Como se indicó en el numeral 5.2, en todo diseño es necesaria la simulación del circuito tanto a nivel lógico como a nivel físico antes de ser enviado para su fabricación.

En el sistema TENTOS la *netlist* del diseño a nivel lógico se genera en el módulo *Circuito*, y la *netlist* del

circuito equivalente del diseño a nivel físico se genera en el módulo PAC. Estas descripciones circuitales permiten la simulación del diseño tanto antes como después de la síntesis de su layout.

El módulo *Simulacao* (Simulación) posee las herramientas necesarias para la simulación circuital del diseño y para el procesamiento gráfico de sus resultados. De acuerdo a la Fig.6.7 estas herramientas son:

a) SPICE

El programa de simulación eléctrica SPICE basa su funcionamiento en la representación del comportamiento físico de cada elemento circuital mediante modelos matemáticos, lo que le permite analizar en detalle la respuesta de cada elemento de un diseño eléctrónico. Sin embargo debido a los elevados tiempos de procesamiento y memoria requeridos, únicamente se lo emplea para la simulación de diseños pequeños o de celdas básicas.

El estudio de la estructura y sintáxis de la *NETLIST* SPICE, así como de las directivas de simulación empleadas en la simulación de circuitos VLSI digitales ha sido realizado en el numeral 5.3 del presente trabajo.

Concatenación de archivos (Fig.6.7)

Este programa requiere para su ejecución:

i) La netlist de descripción del circuito contenida en el archivo *<circuito>.SIM* generado por el EXTRIBO y modificado por la adición de directivas de simulación mediante un editor de texto.

ii) Los resultados de la simulación contenidos en el archivo *PROBE.DAT*.

b) PROBE

El programa PROBE es un utilitario del SPICE que realiza el procesamiento y exhibición gráfica de los resultados de la simulación SPICE. Sus principales opciones son:

i) Grafico de los resultados en diferentes rangos y modos de escalas (decimal, sexagecimal y logarítmica)

ii) Operaciones matemáticas elementales entre las tablas resultantes correspondientes a cada nodo.

iii) Impresión de gráficos.

Concatenación de archivos (Fig.6.7)

Requiere del archivo de configuración *PROBE.DEV* y sus datos de entrada los lee del archivo *PROBE.DAT* generado por el programa SPICE.

Los dispositivos de generación de gráficos que emplea son la pantalla del computador, y el impresor.

c) ARAMOS y d) SIMDE

Con respecto al simulador eléctrico ARAMOS, no ha sido factible realizar su estudio en detalle debido a la falta de documentación respecto a la sintáxis de su *NETLIST* y de sus directivas de simulación, lo que tampoco ha permitido el empleo del programa SIMDE para la exhibición de los resultados de este tipo de simulación.

e) HDC

Corresponde a un simulador funcional cuya *NETLIST* es especificada en lenguage C, sin embargo, este programa no ha sido todavía integrado a la versión del paquete de herramientas TENTOS de Abril de 1991.

CAPITULO 7

CASO DE ESTUDIO SENCILLO UTILIZANDO TENTOS: DISEÑO DE UN MEDIO SUMADOR

En los capítulos anteriores, se ha configurado el marco teórico requerido por un diseñador para el desarrollo de ASICs tanto en lo referente a los fundamentos físicos, cuanto a las metodologías de diseño. En el presente capítulo se reunen los elementos anteriores y se los aplica en un caso de estudio concreto, en el que se sigue el proceso completo para el desarrollo de un ASIC, desde su concepción al nivel más alto (funcional), pasando por los pasos de diseño, simulación y síntesis a diferentes niveles, hasta la obtención de los prototipos y su caracterización.

El diseño se lo realiza a manera de tutorial en base a las herramientas del sistema TENTOS. Es necesario tener en cuenta el diagrama de flujo indicado en la Fig.6.7 a fin de seguir el orden de uso de las herramientas así como la secuencia de generación de los archivos y su nominación en el transcurso del diseño.

El ASIC diseñado fue propuesto para su integración en el Chip Multiproyecto (MPC 91-3) del Centro Nacional de Microelectrónica (CNM) de Barcelona-España y fundido en Diciembre de 1991 por la empresa European Silicon Structures ES2 de Francia. Por tanto a lo largo del diseño cumple con las normas propuestas por el CNM para los diseños participantes en el MPC.

7.1 DISEÑO A NIVEL FUNCIONAL

En el diseño de un ASIC a nivel funcional únicamente se plantea la manera como este debe responder a un conjunto dado de señales de entrada, al margen de la tecnología de diseño y de las características eléctricas del circuito que resuelva el problema.

El circuito a diseñarse corresponde a un "Medio Sumador" (o sumador de números de un bit). Esquemáticamente el circuito está formado (Fig. 7.1) por dos entradas A y B correspondientes a los bits a ser sumados y por las salidas S(suma) y C(carry).

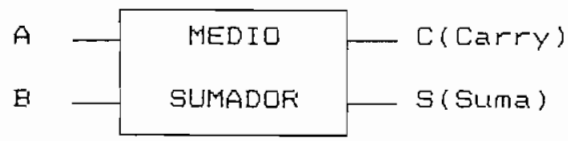


FIGURA 7.1 Esquema del circuito integrado Medio Sumador.

El comportamiento funcional de los terminales de salida

S y C frente a las combinaciones lógicas en los terminales de entrada A y B está dado por la Tabla 7.1:

BITS DE ENTRADA		BITS DE SALIDA	
A	B	S(Suma)	C(Carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLA 7.1 Formulación funcional del circuito integrado Medio sumador.

De donde se tiene que:

$$C = A \cdot B$$

$$S = \overline{A}B + A\overline{B}$$

Sumando el elemento "0" dos veces ($A\overline{A}$ y $B\overline{B}$), se tiene:

$$S = \overline{A}B + A\overline{B} + (A\overline{A} + B\overline{B})$$

$$S = (\overline{A}+\overline{B}) \cdot (A+B)$$

$$S = (\overline{A \cdot B}) \cdot (A+B)$$

Si: $X = \overline{A \cdot B}$ y $Y = A+B$

Entonces: $C = \overline{X}$ $S = X \cdot Y$

Que son las ecuaciones en las que se basará el diseño en los niveles inferiores del circuito integrado propuesto.

7.2 DISEÑO A NIVEL ESTRUCTURAL

7.2.1 Inicialización del sistema

Al iniciar el diseño de un circuito integrado, se deben identificar sus parámetros generales de configuración, esto es:

a) Nombre del diseño.

b) Tecnología de diseño

c) Directorios de trabajo.

a) Nombre del diseño

En las descripciones del Capítulo 6 y en particular de la Fig.6.7, el conjunto de archivos que contienen el diseño han recibido la nominación general de *<circuito>.**, en el caso concreto de circuito "Medio Sumador" el diseño ha sido llamado "*MED_SUM*". Este nombre cumple con las restricciones de nomenclatura dadas por el CNM, que son:

> "*** Nombres de celdas y circuitos:**
> Normas:
> – El primer carácter ha de ser una letra
> – Los otros caracteres pueden ser letras o números y de los caracteres especiales **unicamente** podrá usarse el "_"
> – La longitud de los nombres no podrá superar 10 caracteres"(1)

En el sistema TENTOS el diseño se nomina mediante la opción *Arquivos/Circuito* (ó directamente mediante *<F7>*), y se digita la secuencia MED_SUM, como resultado este nombre aparecerá en la esquina inferior izquierda del interfaz del TENTOS durante todo el desarrollo del diseño.

---

(1) "Servicio MPC - Especificaciones de Participación -. Restricciones Generales", CNM Barcelona - España, pág 1.

b) Tecnología de diseño

Con respecto a la tecnología de diseño la regulación del CNM es:

"* Tecnología de diseño:
Las tecnologías escogidas para el MPC91 son:
ECPD15 (ES2 1.5µm) MPC91-1 Y MPC91-3
BCE (AMS, 2µm doble Poly doble Metal) MPC 91-2"(2)

Sin embargo, puesto que la biblioteca de celdas del sistema TENTOS está definida para la tecnología CMOS15 y considerando que la tecnología CMOS15 es un caso particular de la tecnología ECPD15 (DUAL LAYER METAL 1.5µm CMOS) en el que no se hace uso de la capa de METAL2, se ha seleccionado la tecnología *CMOS15* en la opción *Arquivos/Tecnologia*, (ó *<F9>*), aun cuando ello implique un desperdicio de recursos desde el punto de vista de diseño.

Cabe notar que la biblioteca de PADs empleados (PADLIB2), se halla definida para la tecnología ECPD15 y tiene sus terminales definidos en METAL2, por lo que necesariamente se debe acceder a esta capa para la interconexión de los PADs lo que se ha hecho, como se verá posteriormente, en forma manual mediante el editor de máscaras EMA2.

De omitirse esta definición el interfaz asume la tecnología de diseño indicada en el archivo de configuración *TENTOS.CFG* (Anexo E), en que se tiene la directiva:

$arquivo_regras : CMOS15

(2) "Servicio MPC - Presentación -". CNM Barcelona - España, pág. 7.

c) Directorios de trabajo

Mediante la opción *Arquivos/Diretorio* se ha escogido para la generación y lectura de los archivos de diseño al directorio \MED_SUM. De omitirse esta definición el sistema TENTOS asumiría los directorios de trabajo indicados en el archivo *TENTOS.CFG* (Anexo E), esto es:

```
DIRETORIOS DO USUARIO
     $dir_entrada              \TENTOS
     $dir_saida                \TENTOS
     $dir_tmp                  \TENTOS
```

7.2.2 DESCRIPCION DEL DISEÑO

El diseño a nivel estructural corresponde a la descripción del circuito que ejecuta el conjunto de ecuaciones lógicas planteadas en su descripción a nivel funcional.

La descripción del diseño a nivel estructural puede realizarse en forma textual o gráfica, es necesario destacar que tan solo una de ellas es suficiente; no obstante, para fines de ilustración se han realizado ambas descripciones.

a) Descripción gráfica

Para la descripción gráfica del diseño al nivel lógico se invoca la opción *Circuito/Edicao Esquematico*.

El esquema del circuito lógico que realiza las ecuaciones funcionales deducidas anteriormente:

$$X = \overline{A \cdot B} \qquad Y = A+B$$

$$C = \overline{X} \qquad S = X \cdot Y$$

ha sido generado en el archivo *MED_SUM.ESQ* y se grafica en la Figura 7.2, en su construcción han sido considerados los siguientes aspectos:

i) Los símbolos de las celdas estándar se han leído de los archivos *NAND2.SMB*, *INV.SMB*, *OR2.SMB* y *AND2.SMB* pertenecientes a la biblioteca del sistema TENTOS.

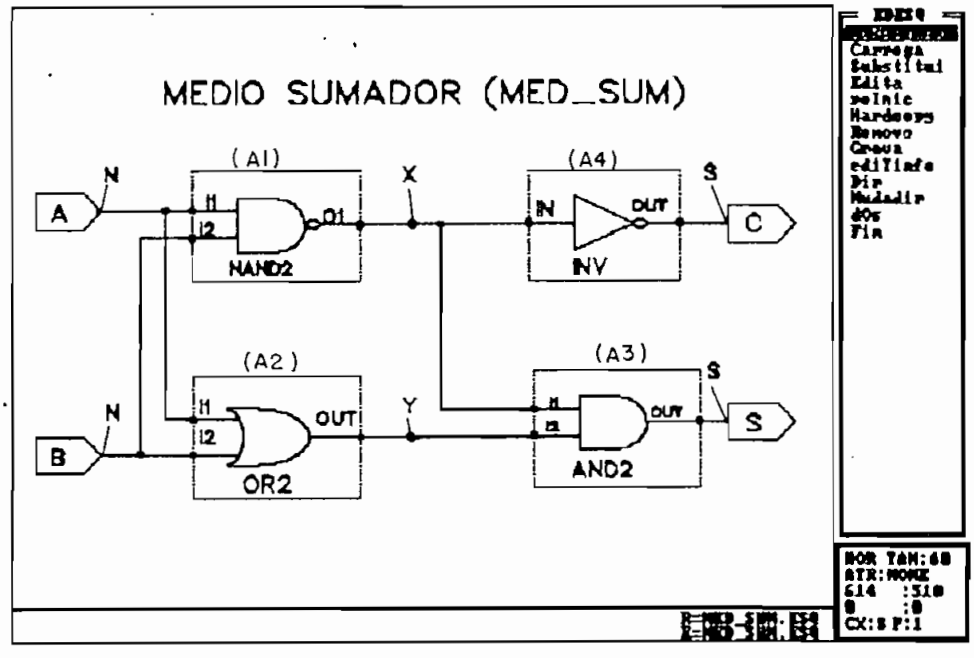


FIGURA 7.2 Diseño del circuito MED_SUM generado en el programa EDESQ.

ii) Los terminales del diseño, en este caso A/B (de entrada) y S/C (de salida) deben ser adscritos cada uno a un pin de esquema.

iii) Los textos empleados para rotular: terminales, señales, pines, etc., tienen cada uno un **"atributo"** característico (ver numeral 6.2.4(b)), los atributos asignados a los textos empleados en la Fig.7.2 son:

TEXTO	ATRIBUTO
A	PESQ
B	PESQ
C	PESQ
S	PESQ
X	SIG
Y	SIG
MEDIO SUMADOR (MED_SUM)	TITL
N	ORIE
S	ORIE

Los dos últimos textos con atributo "ORIE" se han introducidos para asignar restricciones de orientación a la ubicación de los terminales de interfaz de modo que la estructura del circuito es la indicada en la Fig.7.3

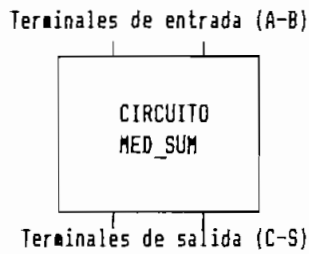


FIGURA 7.3 Orientación de terminales en el circuito MED_SUM.

iv) Los textos tienen además un **"punto de efecto"** que indica el elemento del esquema sobre el que inciden, el punto

de efecto y el elemento del esquema pueden ser coincidentes como en el caso de los textos A, B, C y S, o pueden hallarse distantes como sucede con los textos N, X, Y y S en que las líneas ligadas a ellos indican su punto de efecto.

b) Descripción textual

La descripción del diseño en forma textual puede hacerse en uno de los dos lenguages siguientes:

i) NILOTRANCA

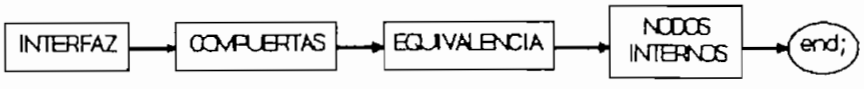
ii) SPICE

También en este caso una de las dos descripciones es suficiente para caracterizar a un diseño particular.

i) Descripción en lenguage NILOTRANCA

Antes de iniciar la descripción de la NETLIST, se debe informar al sistema TENTOS el lenguage que se empleará, para ello en el submenú *Arquivos/Descricao* (ó *<F8>*) se selecciona la opción "NILO".

El lenguage NILOTRANCA a emplearse en esta descripción se estudia en detalle en el Anexo D.1, la estructura de su NETLIST es:



NOTA:

En la descripción en lenguage NILOTRANCA se construirá una red similar en configuración, y restricciones, a aquella de la Fig.7.2, ello no implica que las dos descripciones sean interdependientes o que la una de lugar a la otra, sino que se emplea este gráfico para facilidad de exposición de la construcción de la NETLIST NILOTRANCA.

i.1) Sección de INTERFAZ

Los terminales de interfaz del circuito MED_SUM (Fig.7.2) son A/B de entrada (*in*) y S/C de salida (*out*). Las restricciones de orientación asignadas a estos terminales (Fig.7.3) en la formulación gráfica del circuito, tienen sus palabras reservadas equivalentes *North* y *South* en el lenguage NILOTRANCA. La sección de interfaz para el circuito MED_SUM es entonces:

```
Interface
          North in  A,B  : Terminal;
          South out S,C  : Terminal;
Behavior
```

i.2) Sección de COMPUERTAS

Las compuertas empleadas en el circuito de la Fig.7.2 son:

CELDA DE BIBLIOTECA	NOMBRE INTERNO
Nand (2 entradas)	A1
Or (2 entradas)	A2
And (2 entradas)	A3
Not (1 entrada)	A4

que en la NETLIST NILOTRANCA se declaran de la siguiente manera:

```
Gate
        A1(in=2)   : Nand;
        A2(in=2)   : Or;
        A3(in=2)   : And;
        A4(in=1)   : Not;
```

i.3) Sección EQUIVALENCIA

En esta sección se formula el enlace entre los terminales del diseño de la sección de INTERFAZ y las celdas estándar de la sección de COMPUERTAS. En la Fig.7.2 se tiene que:

A = entrada a Nand2 (A1), entrada a Or2 (A2)

B = entrada a Nand2 (A1), entrada a Or2 (A2)

C = salida de Inv (A4)

S = salida de and2 (A3)

por tanto la sección de EQUIVALENCIA del circuito MED_SUM es:

```
Equivalence
        A = A1.1,A2.1
        B = A1.2,A2.2
        C = A4.2
        S = A3.3
```

i.4) Sección de NODOS INTERNOS

Los nodos internos en el esquema de la Fig.7.2 en forma descriptiva son:

X = salida de Nand2 (A1), entrada a Inv (A4),
entrada a And2 (A3)

Y = salida de Or2 (A2), entrada a And (A3).

al interpretarse esta descripción en lenguage NILOTRANCA no se debe olvidar que primero se declaran los terminales de salida de las celdas y luego los de entrada, así:

```
Node
            X = A1.3,A4.1, A3.1;
            Y = A2.3,A3.2;
```

Luego de agrupar los elementos anteriores y añadir las estructuras de delimitación (encabezamiento y final) de la NETLIST, se obtiene la descripción del circuito MED_SUM en lenguage NILOTRANCA contenida en el archivo *MED_SUM.NIL*:

```
Agency Sumador de 1_bit MED_SUM;
Level = NILO;
Interface
            North in  A,B   : Terminal;
            South out S,C   : Terminal;
Behavior
Gate
            A1(in=2)   : Nand;
            A2(in=2)   : Or;
            A3(in=2)   : And;
            A4(in=1)   : Not;
Equivalence
            A = A1.1,A2.1
            B = A1.2,A2.2
            C = A4.2
            S = A3.3
Node
            X = A1.3,A4.1, A3.1;
            Y = A2.3,A3.2;
End;
```

ii) Descripción en lenguage SPICE

En este caso, para informar al sistema TENTOS el tipo de descripción a realizar, se selecciona en el submenú *Arquivo/Descricao* ó (*<F8>*) la opción "SPICE".

A diferencia de la NETLIST NILOTRANCA, la NETLIST SPICE guarda una relación directa con la descripción gráfica de la Fig.7.2 contenida en el archivo *MED_SUM.ESQ*, ya que puede ser generada automáticamente a partir de ésta, mediante la opción

Circuito/esquema_sPice en el archivo ***MED_SUM.SIM***, esta NETLIST está estructurada de la siguiente manera:

ii.1) Definiciones subcircuitales, en las que las compuertas de la descripción gráfica se consideran como subcircuitos que se leen del archivo *<celda>.CIR* correspondiente de la biblioteca del sistema TENTOS.

ii.2) Llamadas a subcircuitos, en que se describen los enlaces entre las celdas, la nominación asignada a los nodos es la misma que se emplea en el diagrama de la Fig.7.2.

ii.3) Terminales de interfaz, en que se indican los terminales del circuito y su restricción de orientación (Fig.7.3)

NOTA:

"Para propósitos de síntesis para la generación del layout del compilador solo se requieren las secciones de "llamadas a subcircuitos" y de "terminales de interfaz" pudiendo omitirse las "definiciones subcircuitales".

A continuación se indica la NETLIST SPICE del archivo ***MED_SUM.SIM*** que describe al esquema de la Fig.7.2, en ella se ha omitido la estructura de cada subcircuito que puede ser estudiada en detalle en el Anexo F.1.

```
*|----------------------------------------------|*
*|  RELATORIO SPICE     MEDIO SUMADOR (MED_SUM) |*
*|  VERSAO  1.0                                 |*
*|  PROJETO SID-UFRGS                           |*
*|----------------------------------------------|*

*|----------------------------------------------|*
*|  DEFINICAO DOS SUBCIRCUITOS                  |*
*|----------------------------------------------|*
* CELULA: NAND 2 ENTRADAS
.SUBCKT NAND2 I1 I2 O1 vcc
  '
.ENDS NAND2

* CELULA: OR 2 ENTRADAS
.SUBCKT OR2 I1 I2 OUT vcc
  '
.ENDS OR2

* CELULA: AND 2 ENTRADAS
.SUBCKT AND2 I1 I2 OUT vcc
  '
.ENDS AND2

* CELULA: INVERSOR
.SUBCKT INV IN OUT vcc
  '
.ENDS INV

*|----------------------------------------------|*
*|_CHAMADA_DOS_SUBCIRCUITOS                     |*
*|----------------------------------------------|*
X1  A  B  X  vcc  NAND2
X2  A  B  Y  vcc  OR2
X3  X  Y  S  vcc  AND2
X4  X  C     vcc  INV

*|_SINAIS_DE_INTERFACE_DO_CIRCUITO |*
*|---------------------------------|*
*  interface: A   *   orientacao=N  *
*  interface: B   *   orientacao=N  *
*  interface: C   *   orientacao=S  *
*  interface: S   *   orientacao=S  *
.END
```

c) Simulación lógica

Con el propósito de verificar la corrección de la formulación del diseño al nivel lógico, esta debe ser simulada, a

fin de confrontar los resultados obtenidos con la formulación del circuito al nivel funcional de la Tabla 7.1 El sistema TENTOS empleado (versión de Abril/91) carece de simulador lógico por lo que se ha empleado el simulador NDL del sistema TEDMOS, estudiado en el Capítulo 5.3.

La NETLIST NDL del circuito MED_SUM que se indica a continuación ha sido generada a partir de la NETLIST SPICE del archivo *MED_SUM.SIM* y grabada en un archivo *MED_SUM.NDL*, de la misma manera que en la NETLIST SPICE anterior, se ha omitido el detalle de los subcircuitos que se indica en el Anexo F.1.

```
%-----------------------------------------------
% DESCRIPCION NDL     MEDIO SUMADOR (MED_SUM)   ¦
% FREDY LEMUS C./IVAN BERNAL C.                 ¦
%-----------------------------------------------
FAMILY CMOS;

% DEFINICION DE SUBCIRCUITOS
% ==========================
% SUBCIRCUITO NAND 2 ENTRADAS
DEFINE NAND2 I1 I2 OUT;
 '
END;

% SUBCIRCUITO OR 2 ENTRADAS
DEFINE OR2 I1 I2 OUT;
 '
END;

%    SUBCIRCUITO AND 2 ENTRADAS
DEFINE AND2 I1 I2 OUT;
 '
END;

%    SUBCIRCUITO INVERSOR
DEFINE INV In OUT;
 '
END;
```

```
% CIRCUITO PRINCIPAL MED_SUM
% ==========================

INPUT   A  B;
NAND2  A  B  X;
OR2    A  B  Y;
AND2   X  Y  S;
INV    X  C;
END
```

Los comandos de simulación lógica han sido generados en el archivo *COMANDOS.SIM* que a continuación se describe:

```
% POLARIZACION

LIGUE Vdd 1      ;% Nodo Vdd se liga a "1L" permanentemente.
LIGUE Gnd 0      ;% Nodo Gnd se liga a "0L" permanentemente.

% DATOS DE ENTRADA

ENTRADA A 0011   ;% Se forman todas las combinaciones lógicas
ENTRADA B 0101   ;% posibles entre A y B.

% DIRECTIVAS DE CONTROL

GRAFICO          ;% Se invoca al interfaz gráfico del simulador.
PLUGUE A 1       ;% Nodo A se conecta al canal 1 del osciloscopio.
PLUGUE B 2       ;% Nodo B se conecta al canal 2.
PLUGUE C 4       ;% Nodo C se conecta al canal 4.
PLUGUE S 6       ;% Nodo S se conecta al canal 6.
FASE   16        ;% Se ejecuta la simulación durante 16 fases de reloj.
```

Los resultados de la simulación se exhiben en la Fig.7.4, en este gráfico se observa que el comportamiento de las señales lógicas de los terminales de salida C y S frente a las combinaciones de las señales de entrada A y B coincide con la formulación al nivel funcional de la Tabla 7.1, en consecuencia los diseños a nivel funcional y lógico son equivalentes.

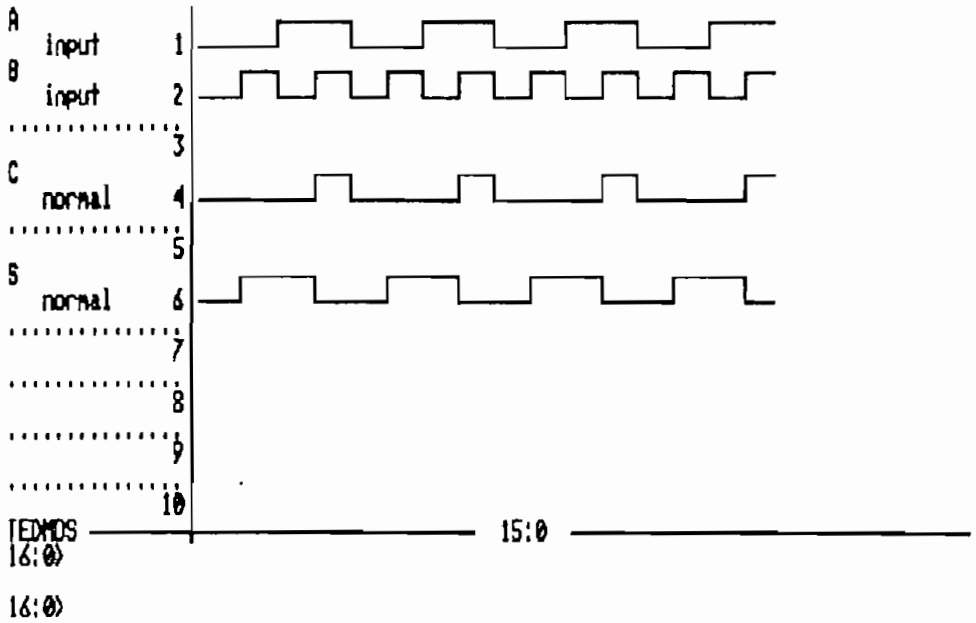


FIGURA 7.4 Resultado de la simulación HDL del diseño lógico del circuito MED_SUM.

7.3 DISEÑO A NIVEL FISICO

El diseño al nivel lógico descrito anteriormente debe ser sintetizado, este proceso de "síntesis" permite, a partir de la descripción lógica, generar la descripción de las máscaras que permitan a los fabricantes fundir el ASIC.

En el diseño del circuito MED_SUM se ha usado el método de síntesis TRAMO en base a celdas estándar del sistema TENTOS.

7.3.1 Síntesis del circuito

Aun cuando el circuito MED_SUM diseñado ha sido descrito en dos lenguages diferentes:

1. NILOTRANCA (archivo *MED_SUM.NIL*), y

2. SPICE (archivo *MED_SUM.SIM*).

Ello no implica que ambas descripciones den lugar a layouts distintos, sino que mas bien podrían considerarse como dos opciones equivalentes para describir un mismo circuito. La única diferencia radica en el programa de compilación que el interfaz emplea, salvo esta diferencia, el proceso de síntesis del circuito es idéntico en procedimientos y resultados para ambos casos.

a) Selección de Parámetros

Mediante la opción *Sintese/Parametros* se ingresa al menú de selección de parámetros de ubicación y enlace de las celdas, los parámetros a seleccionarse son:

i) Parámetros de particionamiento.

ii) Parámetros de enrutamiento.

i) Parámetros de particionamiento

Los parámetros para el particionamiento del diseño se estudian en el Anexo D. Dada la simplicidad de el circuito MED_SUM se han asumido los parámetros propuestos por el sistema que son:

a: Escolha: Individual

De modo que el compilador genere una sola solución.

b: Numero regioes: 1

Puesto que el circuito es pequeño, no existen riesgos de concentración de interconexiones en algun sector del layout.

c: Numero Bandas: Calculado

No se consideran restricciones en el número de bandas.

d: Relacao Aspecto: 1

Es deseable que el circuito tienda a ser cuadrado por lo que la relación de aspecto (largo/ancho) se ha fijado en 1.

e: Percentual Balanco: 50%

f: Fator de forma: 1.000

El Porcentaje de Balanceo define el tamaño máximo de una banda en relación al promedio, y el Factor de Forma permite aumentar el tamaño de las bandas extremas en proporción a la reducción del tamaño de las bandas centrales en circuitos grandes, en este caso se espera tener pocas bandas (1 ó 2) por lo que estos parámetros no afectan a la forma del layout.

g: Acrescimo Area: 0.000

Se ha despreciado el incremento de área que implican las celdas de interconexión.

ii) Parámetros de enrutamiento

Para el posicionamiento de las líneas de polarización se han empleado los parámetros:

a: *Alimentacao Topo:* Vcc
b: *Alimentacao Esquerda:* Vcc

Que darán lugar a la configuración del circuito indicada en la Fig. 7.5.

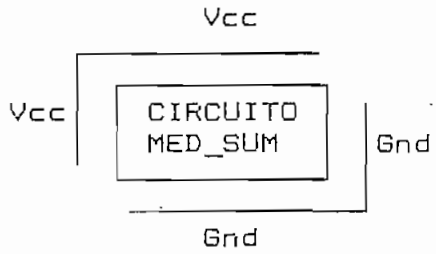


FIGURA 7.5 Líneas de polarización en el circuito MED_SUM.

b) Compilación

Independientemente del tipo de descripción empleada (NILO ó SPICE) para el diseño al nivel estructural, la compilación se realiza al invocarse la opción *Sintese/TRAMO/Compilacao*.

i) De haberse empleado la descripción NILO (NILOTRANCA) el interfaz emplea el compilador CN (Compilador NILOTRANCA), que genera el archivo *MED_SUM.CNI* a partir de la descripción contenida en el archivo *MED_SUM.NIL*.

ii) De haberse empleado la descripción SPICE, el interfaz emplea el compilador CS (Compilador SPICE), que genera el archivo *MED_SUM.SPC* a partir de la descripción contenida en el archivo *MED_SUM.SIM*.

c) Particionamiento

El proceso de particionamiento (distribución en bandas) de las celdas se realiza con la opción *Sintese/TRAMO/pArticionamiento*, y es el mismo independientemente del archivo generado en la compilación (*MED_SUM.NIL* ó *MED_SUM.SPC*); en consecuencia, en adelante, el diseño es único sin importar la forma en que se haya descrito el circuito al inicio.

Los archivos generados por el particionador son: *MED_SUM.TRG* (intermediario para la síntesis TRAGO), el archivo de documentación *MED_SUM.RPA* y el archivo *MED_SUM.POS* que contiene los esquemas de posicionamiento y es empleado luego por el posicionador, la información gráfica contenida en este último archivo puede ser visualizada mediante la opción *Sintese/TRAMO/Exibicao* en que se genera el esquema que se indica en la Fig.7.6.

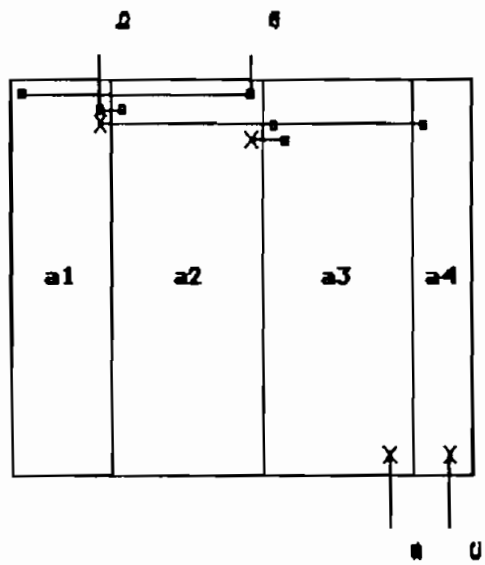


FIGURA 7.6 Exhibición del Particionamiento del circuito MED_SUM (Archivo MED_SUM.POS).

En el diagrama de la Fig. 7.6 conviene destacar que:

i) Las celdas de biblioteca son indicadas por su nombre interno asignado en la netlist SPICE ó NILOTRANCA, en este caso, el esquema ha sido generado a partir de la NETLIST NILOTRANCA, es decir:

CELDA DE BIBLIOTECA	NOMBRE INTERNO
Nand (2 entradas)	A1
Or (2 entradas)	A2
And (2 entradas)	A3
Not (1 entrada)	A4

ii) Las celdas se han distribuido únicamente en una banda dado que se trata de un circuito pequeño.

iii) Las líneas de enlace tienen en sus extremos las marcas "O" que indica la celda a la que entra la señal y "X" que indica la celda de la que la señal sale.

iv) La disposición de las celdas no es definitiva puesto que éstas aún no han sido "posicionadas" sobre la banda.

d) Posicionamiento

La distribución de las celdas dentro de la banda se la realiza por medio de la opción *Sintese/TRAMO/pOsicionamiento*. El programa de posicionamiento emplea la información entregada luego del particionamiento en el archivo ***MED_SUM.POS***.

La información resultante de este proceso se genera en los archivos ***MED_SUM.RPO*** de documentación, y ***MED_SUM.POS*** que se emplea como entrada para el programa de enrutamiento.

Al igual que en el particionamiento, la distribución de celdas generada por el posicionador en el archivo *MED_SUM.POS* puede ser exhibida mediante la opción *Sintese/TRAMO/Exibicao* como se indica en la Fig.7.7.

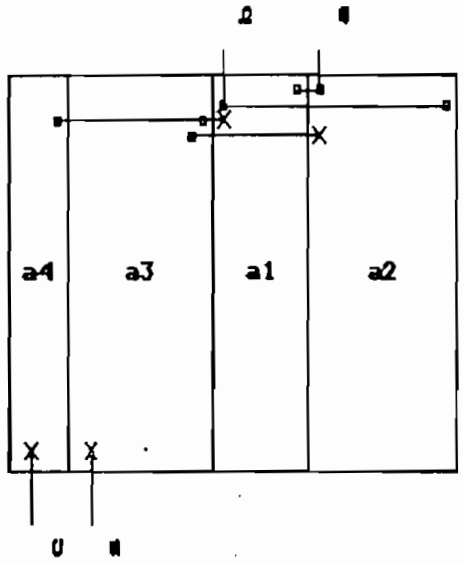


FIGURA 7.7 Exhibición del Posicionamiento del circuito MED_SUM (Archivo MED_SUM.POS).

La notación empleada en el diagrama de la Fig.7.7 es la misma de la Fig.7.6, la única diferencia entre ambas distribuciones radica en que la distribución de celdas del posicionador es definitiva y en base a esta se generará el layout del circuito.

El estado final de las celdas en el circuito (Fig.7.7) se indica en el archivo *MED_SUM.RPO* y es el siguiente:

```
Posicionamento do circuito C:\TENTOS\MED_SUM.POS

Banda 1
celula=    a4 coordenada_simbolica=   0 status rotacional=espelhada
celula=    a3 coordenada_simbolica=  21 status rotacional=espelhada
celula=    a1 coordenada_simbolica=  74 status rotacional=espelhada
celula=    a2 coordenada_simbolica= 109 status rotacional=espelhada
```

e) Enrutamiento

En el proceso de enrutamiento se enlazan las celdas y se genera el layout del circuito.

El enrutamiento se ejecuta por medio de la opción *Sintese/TRAMO/Roteamiento* para la entrada a este proceso el interfaz emplea el archivo *MED_SUM.POS*, y sus resultados se generan en el archivo de documentación *MED_SUM.RRO* y en el archivo *MED_SUM.RS* que contiene el layout del circuito, en formato RS.

El archivo de documentación *MED_SUM.RRO* posee información sobre la ubicación y coordenadas de las celdas y la distribución de las redes en el layout del circuito. Así:

```
Roteamento da Banda 1 do Circuito C:\TENTOS\MED_SUM

Topo VCC. Ordem Trilhas VCC P7 P5 P3 P9 P1 P0 P8 P2 P4 P6 GND.
Largura Banda=199

Celula  Tipo  Largura Coord Nodos
        INT       12    24 c
a4      not       21    36 &c &c !x
        INT       12    57 s
a3      and       53    69 &s &s !y !x
a1      nand      35   122 &x &x !b !a
        INT       12   157 y
a2      or        54   169 &y &y !a !b
```

```
Acesso a Interface Oeste-Leste:
Oeste  Leste
VCC    VCC


GND    GND

Redes Nao Roteadas:


Legenda

INT - celula de interconexao intrabanda
EXT - celula de interconexao interbanda
VAZ - espaco vazio
 !  - entrada de celula funcional
 &  - saida de celula funcional

Informacoes do Modulo Gerado

Area do Circuito   =    26182
Relacao de Aspecto =    0.429

Interface Norte/Sul

Rede   Banda Celula CoordX Pino
     a    1      a1   122    2
     b    1      a2   169    2
     s    1  INTERC    57
     c    1  INTERC    24

Interface Oeste/Leste

Rede   Banda CoordX Trilha
```

7.3.2 Layout del circuito

El proceso de síntesis del circuito MED_SUM entrega como resultado principal el archivo *MED_SUM.RS* que posee la descripción del layout del circuito en formato RS). La visualización y edición de los layouts del diseño, jerárquicos o planos, al nivel físico descritos en formato RS se realiza por medio de la opción *PAC/Edicao Mascara* que invoca al editor de máscaras EMA2 del sistema TENTOS.

La generación del layout mediante el proceso de síntesis TRAMO es consecuencia de un proceso de ubicación de celdas estándar en una estructura de bandas de acuerdo a la distribución asignada a éstas (Fig.7.7). En consecuencia la descripción geométrica contenida en el archivo *MED_SUM.RS* tiene una estructura jerárquica basada en símbolos parciales que se indica a continuación:

ARCHIVO *MED_SUM.RS*

```
DS 1 1 10;
9 "AND2.CEL";
'
'
DF;
DS 2 1 10;
9 "NAND2.CEL";
'
'
DF;
DS 3 1 10;
9 "INV.CEL";
'
'
DF;
DS 4 1 10;
9 "OR2.CEL";
'
'
DF;
```

```
DS 23 1 10;
9 "LAYOUT GLOBAL";
C 3 T -570 0 MX;
C 1 T -1221 0 MX;
C 2 T -1571 0 MX;
C 4 T -2232 0 MX;
9 "ENLACES ENTRE CELDAS";
'
'
DF;

C 23;
     }
E
```

En esta descripción se tiene:

i) Los símbolos de menor jerarquía: *(1) AND2, (2) NAND2, (3) INV* y *(4) OR2*, son tomados de los archivos *<celda>.CEL* de la biblioteca del sistema TENTOS.

ii) El símbolo de mayor jerarquía formado por las llamadas a las celdas básicas y la descripción de las redes que las enlazan y por los caminos de polarización.

Esta estructura jerárquica y el posicionamiento de las celdas sobre las máscaras que las enlazan se esquematiza en la Fig.7.8, el layout del circuito MED_SUM así generado se exhibe en detalle en la Fig.7.9.

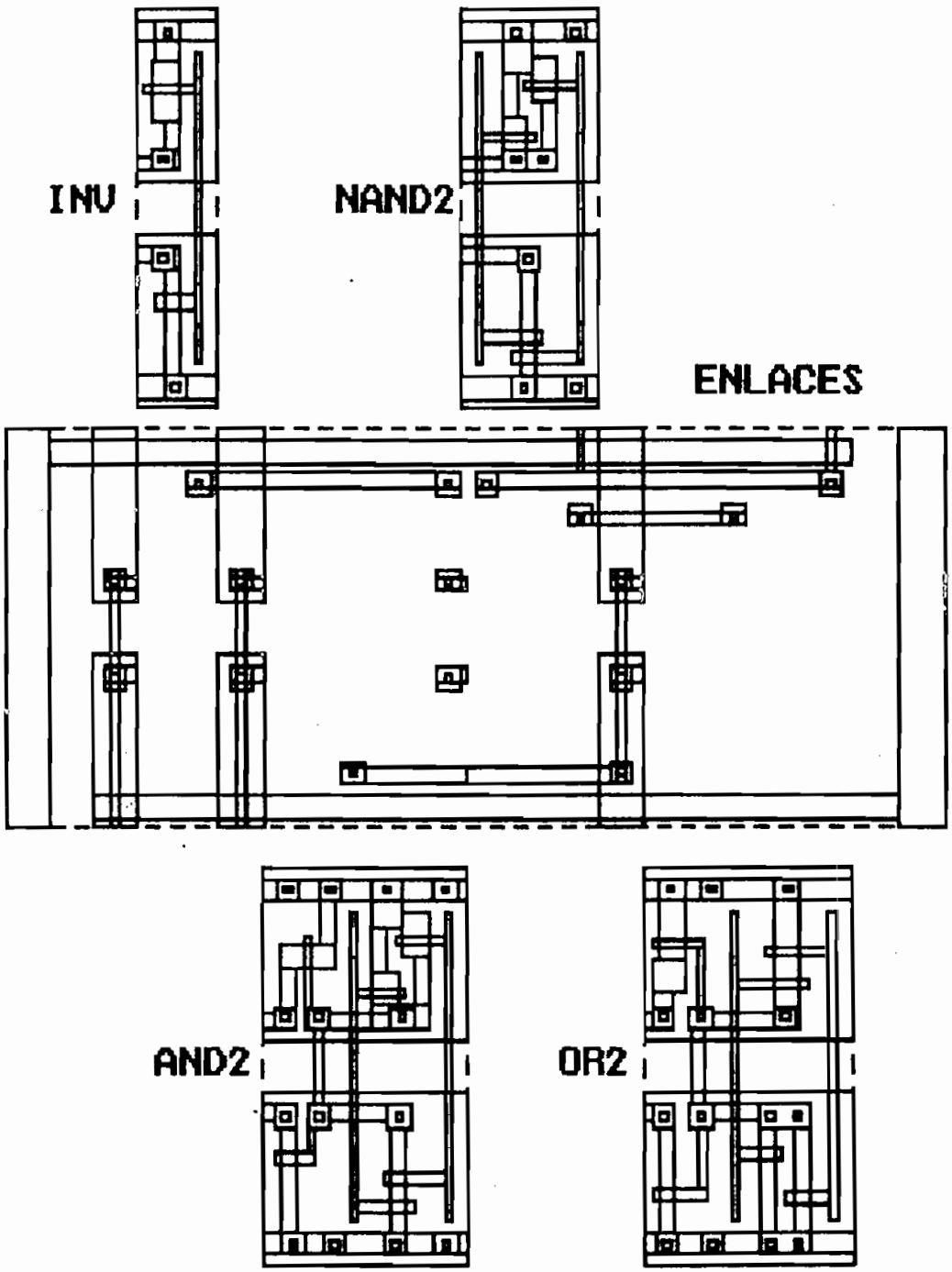


FIGURA 7.8 Ubicación de las celdas básicas sobre las máscaras de interconexiones en el circuito MED_SUM.

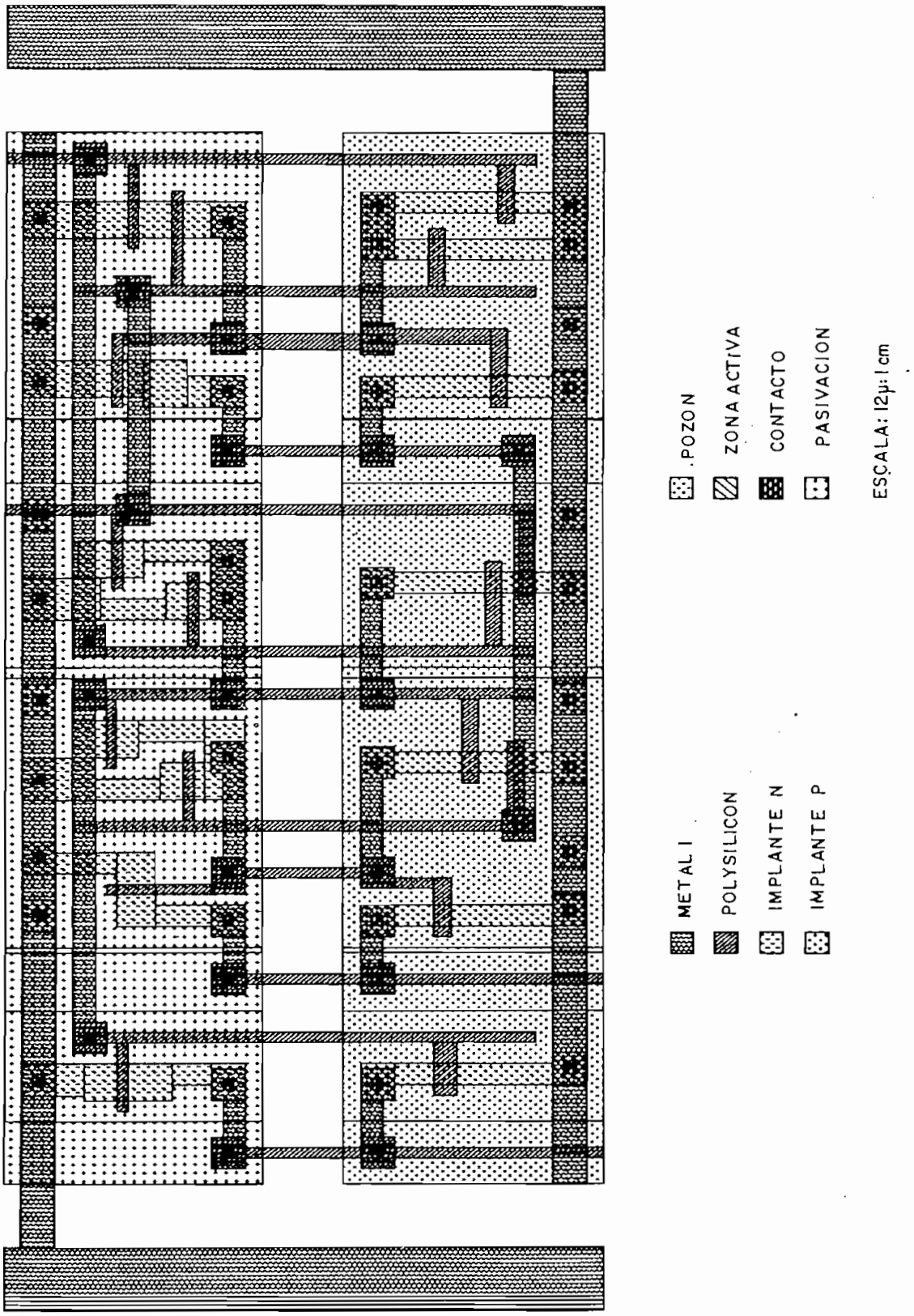


FIGURA 7.9 Layout del circuito MED_SUM generado automáticamente en el sistema TENTOS.

a) Expansión del layout

Como ya se ha indicado, el layout contenido en el archivo *MED_SUM.RS* tiene estructura "jerárquica" que es transformada en "plana" ó "expandida" mediante la opción *PAC/PLANIFICADOR*, el layout expandido es grabado en el archivo *MED_SUM.EXP*.

La estructura de la descripción jerárquica contenida en el archivo *MED_SUM.RS* con respecto a la descripción expandida del archivo *MED_SUM.EXP* es en resumen la siguiente:

ARCHIVO *MED_SUM.RS*

```
DS 1 1 10;
9 "AND2.CEL";
'
'
DF;
DS 2 1 10;
9 "NAND2.CEL";
'
'
DF;
DS 3 1 10;
9 "INV.CEL";
'
'
DF;
DS 4 1 10;
9 "OR2.CEL";
'
'
DF;
```

```
DS 23 1 10;
9 "LAYOUT GLOBAL";
C 3 T -570 0 MX;
C 1 T -1221 0 MX;
C 2 T -1571 0 MX;
C 4 T -2232 0 MX;
9 "ENLACES ENTRE CELDAS"
DF;

C 23;

E
```

ARCHIVO *MED_SUM.EXP*

```
DS 1 1 10;
9 "MED_SUM.EXP"
'
'
'
'
'
'
'
'
'
'
'
'
DF;

E
```

La descripción jerárquica del archivo *MED_SUM.RS* tiene varios símbolos en los que las coordenadas de la descripción del layout de cada símbolo son relativas a su origen de coordenadas particular. El archivo *MED_SUM.EXP* en cambio está

formado por un solo símbolo que contiene las coordenadas de todas las primitivas geométricas del archivo *MED_SUM.RS*, recalculadas por el programa de expansión y definidas con respecto al origen de su símbolo "DS 23" que es el de mayor jerarquía.

El layout descrito en ambos casos *MED_SUM.RS* y *MED_SUM.EXP* es el mismo y se grafica en la Fig.7.9.

7.3.3 Verificación del diseño

En el Capítulo 4 se vió que todo layout de un ASIC antes de considerarse como funcional y topológicamente correcto, debe pasar por un conjunto de verificaciones, a saber:

a) Verificación geométrica.

b) Verificación eléctrica de interconexiones.

c) Verificación funcional (simulación)

a) Verificación geométrica

A pesar de que el layout del circuito MED_SUM de la Fig.7.9 ha sido generado automáticamente, ello no implica que este no posea violaciones de reglas geométricas de diseño debido al caracter experimental del sistema TENTOS. La opción *PAC/DARC* permite identificar automáticamente las violaciones de reglas de diseño en el layout expandido del archivo *MED_SUM.EXP* en base al conjunto de normas establecidas por la fundidora ES2 a través del CNM.

"Reglas de Diseño:
ES2 1.5µm : ES2-ECPD15 rules (rev.A, 30 Dec.1988)
ES2 1.2µm : ES2-ECPD12 rules (rev.A, 30 Oct.1989)" (3)

Estas reglas se hallan contenidas en el archivo *CMOS15.DRC* (Anexo E). Las violaciones detectadas en el layout son indicadas en el archivo *MED_SUM.REL* y registradas sobre el layout en el archivo *MED_SUM.ERR*.

Las violaciones geométricas que se pueden cometer abarcan una gran variedad de posibilidades, no obstante, los errores de definición de celdas estándar y de síntesis de layout detectados en los circuitos generados por el sistema TENTOS son:

i) Superposición de máscaras de Metal1.

ii) Superposición de máscaras de Polysilicon.

iii) Máscaras de Pasivación y Pozo N en posiciones incorrectas.

iv) Ausencia de Zonas Activas en la definición de transistores.

v) Formación de transistores indeseados.

Las correcciones que a continuación se indican han sido realizadas "manualmente" en el editor de máscaras EMA2 (opción *PAC/Edicao Mascara*).

---

(3) "Servicio MPC - Especificaciones de Participación -. Restricciones Generales", CNM Barcelona-España, pág 1.

i) Superposición (solapamiento) de máscaras de Metal1

Los algoritmos de síntesis TRAMO se caracterizan por generar los caminos de enrutamiento horizontales en base a Metal1, estos caminos son implantados sobre las máscaras de las celdas por lo que las capas de Metal1 de éstas se superponen a las de las líneas de enrutamiento horizontal (Fig.7.10) originando espacios de Metal1 con doble definición (solapados) que violan las reglas del MPC. Específicamente la regla violada dice:

> "* Polígonos y Paths
> Están prohibidos
> - Los lados de longitud cero
> - Los polígonos y paths que se solapan consigo mismo"(4)

La opción *PAC/DARC* del sistema TENTOS detecta estos sobrelapamientos y genera en el archivo *MED_SUM.REL* el mensaje:

T 802 - distancia metal/metal < 2.4;

puesto que el programa DARC considera a cada rectángulo como una primitiva geométrica individual, por lo que interpreta la superposición de rectangulos de un mismo material (en este caso Metal1) como violaciones de las distancias mínimas permitidas para éste.

La máscara de Metal1 con los errores de solapamientos corregidos se exhibe en la Fig.7.11.

---

(4) "Servicio MPC - Especificaciones de Participación -, Restricciones generales", CNM Barcelóna-España, pág 2.

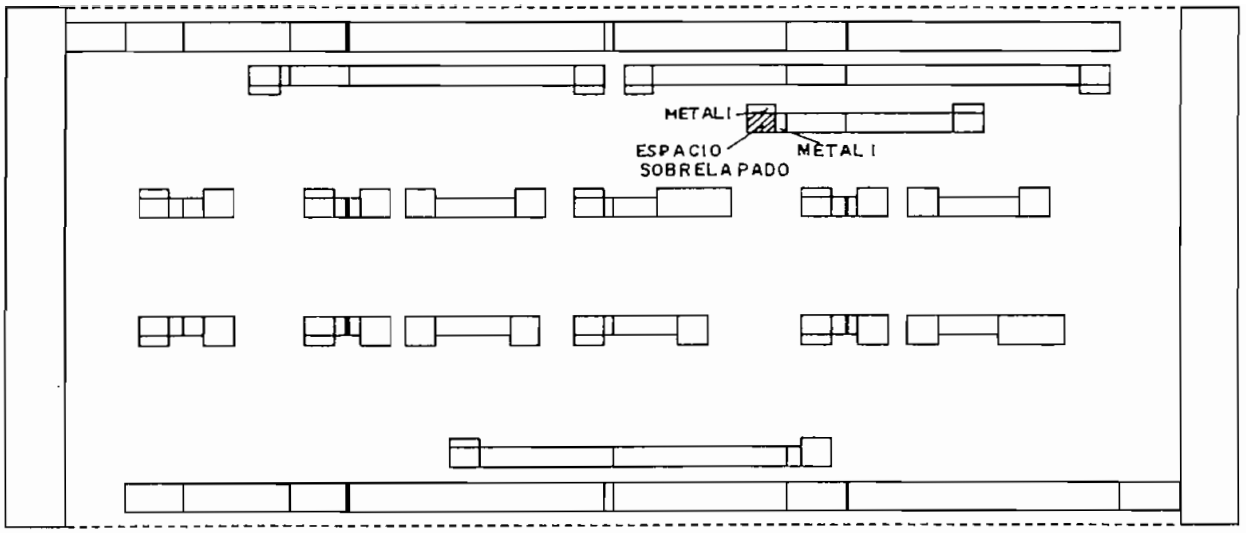


FIGURA 7.10 Máscara de Metal1 del circuito MED_SUM generada automáticamente.

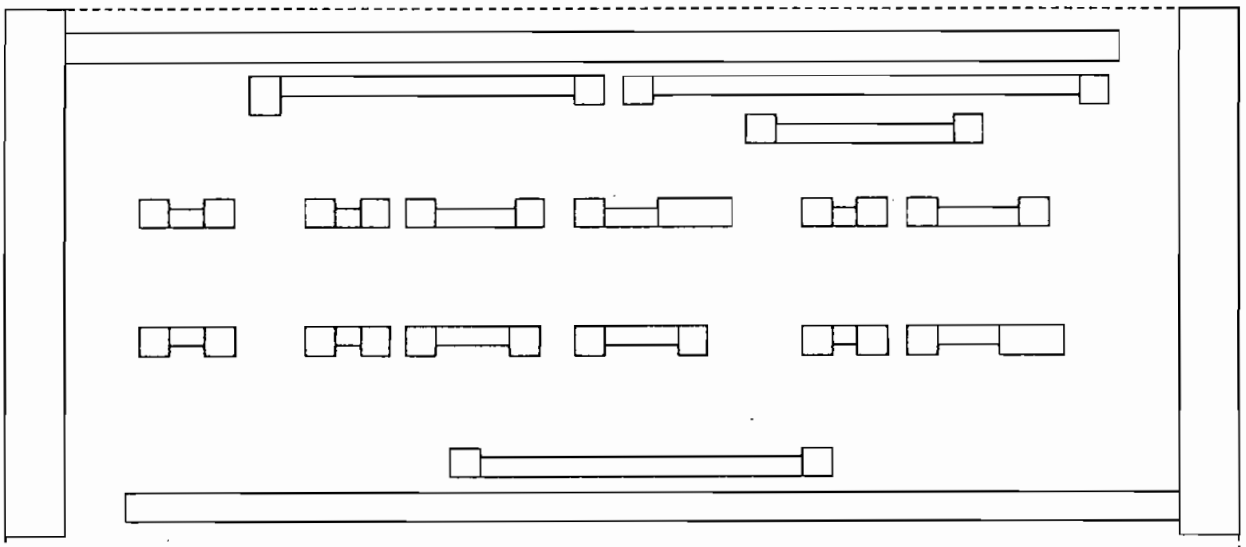


FIGURA 7.11 Máscara de Metal1 del circuito MED_SUM corregida.

ii) Superposición (solapamiento) de máscaras de polysilicon

De manera análoga a los caminos de enrutamiento horizontales, los caminos verticales se generan automáticamente en base a Polysilicon y son sobrepuestos sobre las máscaras de

este material definidas para las celdas estándar generando solapamientos de estas capas (Fig.7.12). El mensaje generado en el archivo *MED_SUM.REL* es:

T 504 - distancia poli/poli (2.4;

por las mismas razones indicadas para la máscara de Metal1.

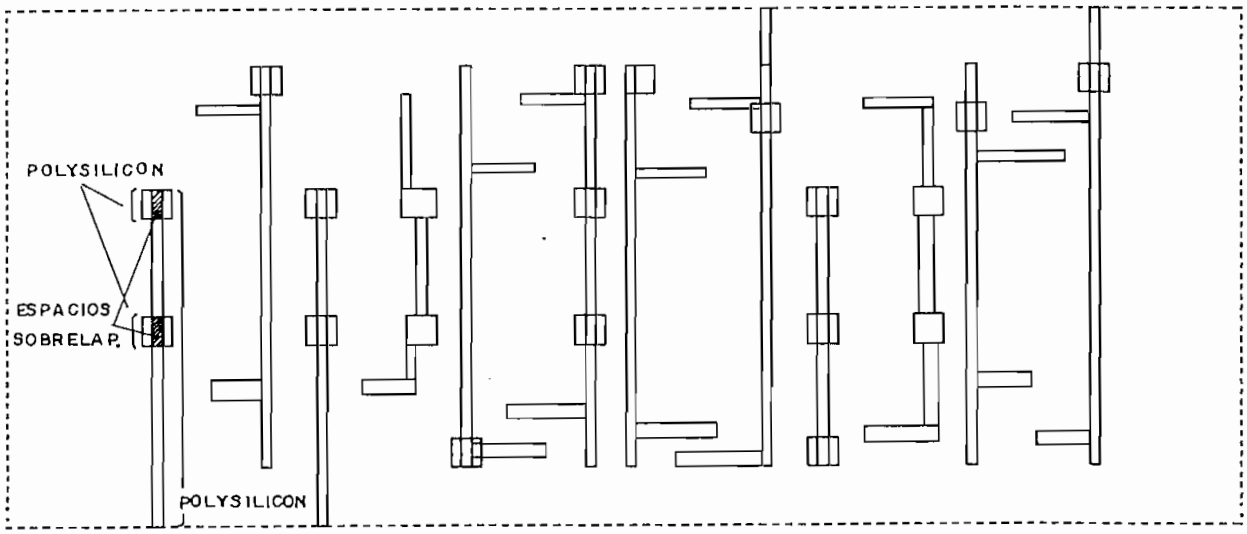


FIGURA 7.12 Máscara de Polysilicon del circuito MED_SUM generada automáticamente.

El layout de la máscara de Polysilicon corregido se indica en la Fig.7.13.

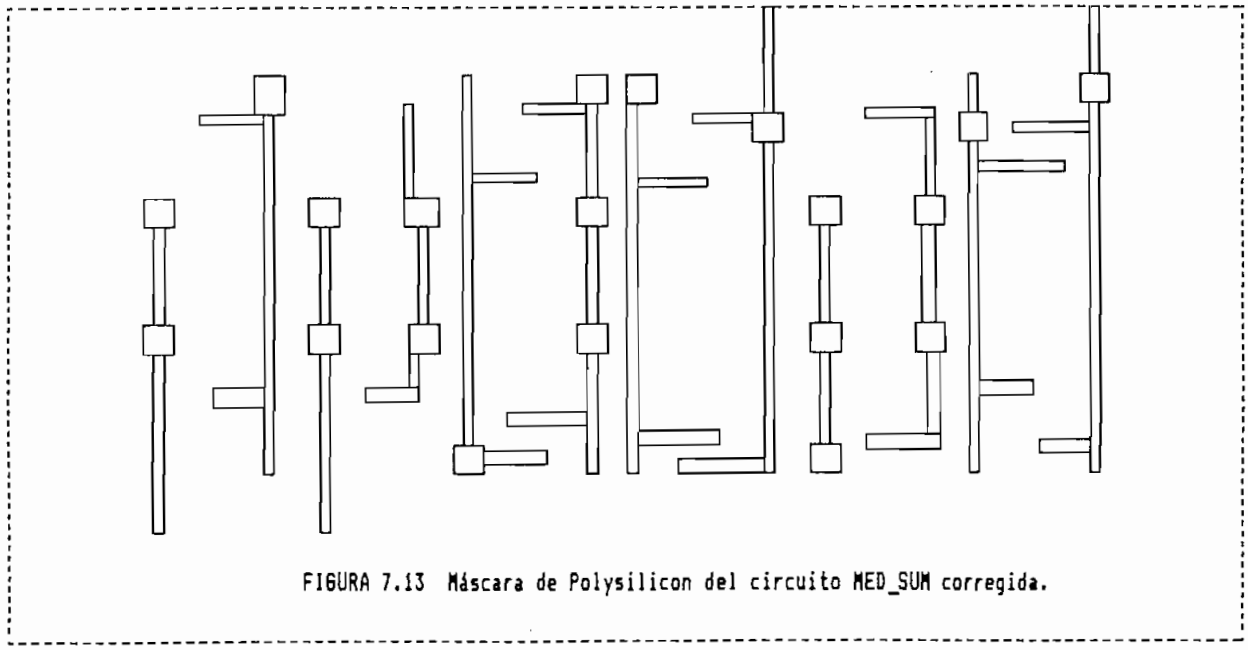


FIGURA 7.13 Máscara de Polysilicon del circuito MED_SUM corregida.

iii) Máscaras de Pasivación y Pozo N en posiciones incorrectas

Como se indicó en el Capítulo 3, la tecnología CMOS consiste en transistores del tipo PMOS sobre una zona N y del tipo NMOS sobre una zona P. Sin embargo, las celdas estándar del sistema TENTOS y por tanto el layout que generan al ser incluídas en un circuito, poseen los transistores de tipo PMOS sobre una zona de Pasivación y los transistores del tipo NMOS sobre una zona N (Fig.7.14).

El programa DARC del sistema TENTOS al identificar este error genera en el archivo *MED_SUM.REL* los mensajes:

iii.1) T transistor n dentro de poco n;
T transistor p fora do poco n;

iii.2) T 901 - dimensao da depassivacao < 100;

Los mensajes (iii.1) indican la posición incorrecta del Pozo N con respecto a los transistores PMOS y NMOS. El mensaje (iii.2) indica que la zona de Pasivación es más pequeña de lo esperado ya que en su posición correcta debería sobreponerse sobre toda la oblea de silicio, a excepción de las almohadillas de contacto de los PADs.

Para corregir estos errores, en el layout del circuito MED_SUM se ha removido la zona de Pasivación y se ha trasladado el Pozo N a su posición correcta (bajo los transistores PMOS) como se indica en la Fig.7.15.

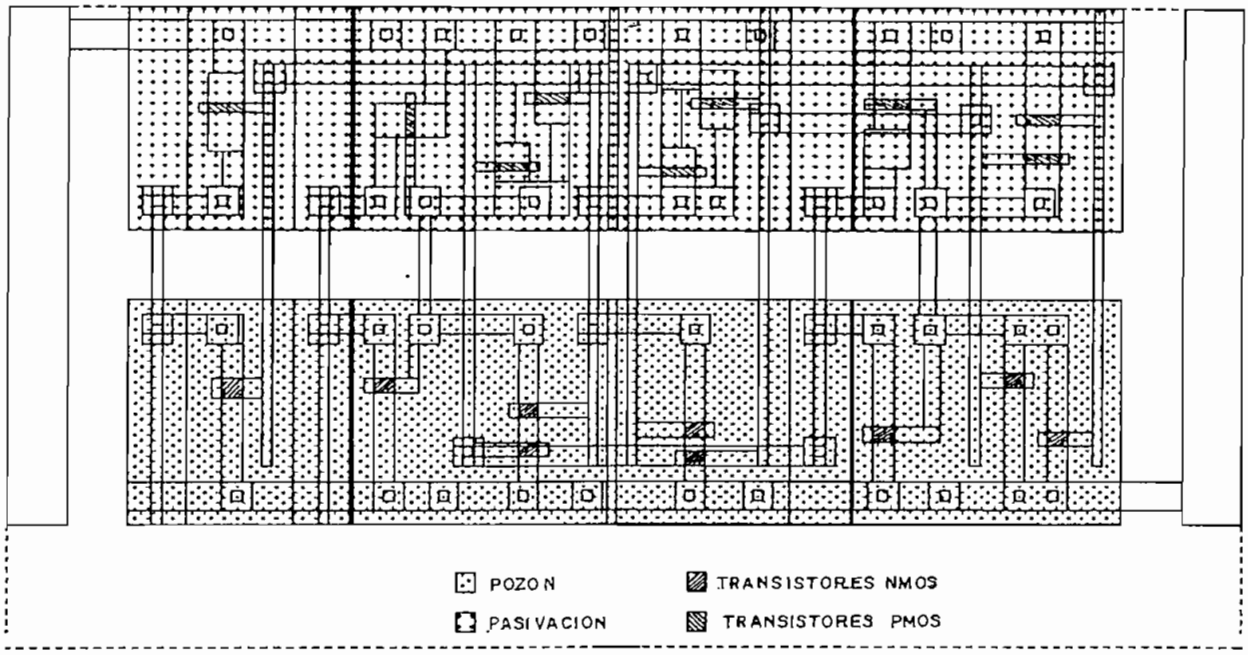


FIGURA 7.14 Definición de Transistores sobre las capas de Pozo N y de Pasivación en el circuito MED_SUM generado automáticamente.

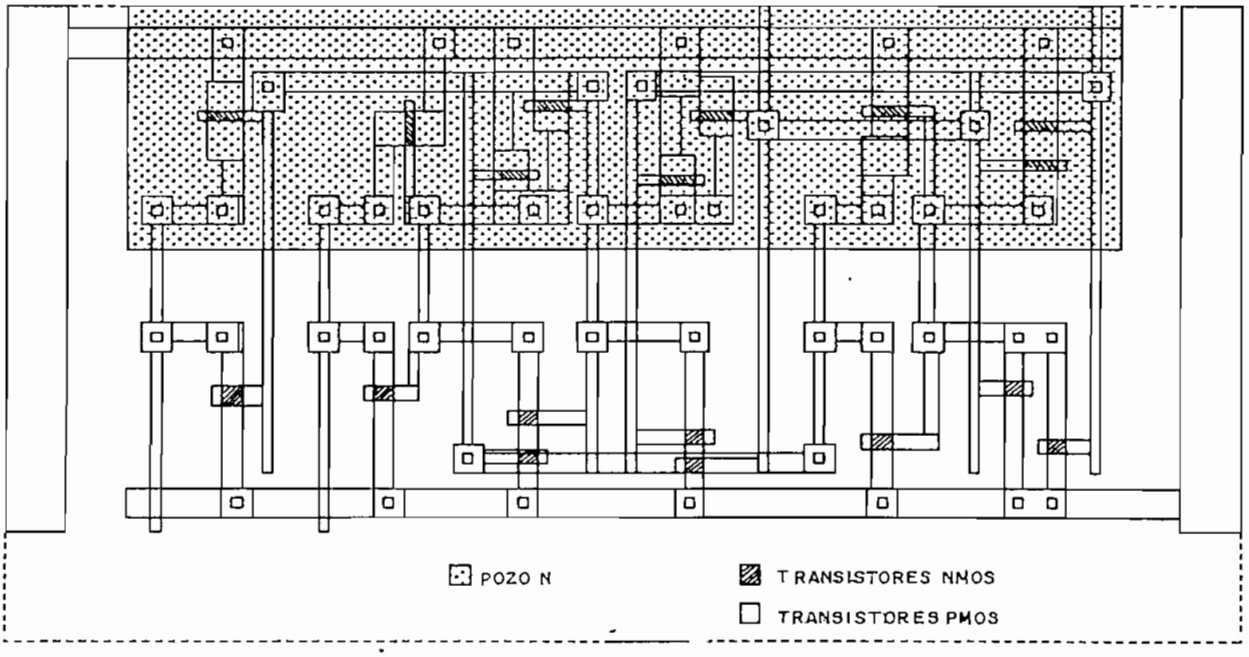


FIGURA 7.15 Layout del circuito MED_SUM con el Pozo N trasladado y la zona de Pasivación removida con respecto al layout de la Figura 7.14.

iv) Ausencia de Zonas Activas en la definición de transistores.

Al analizar el proceso tecnológico de fabricación de los CIs (Capítulo 3) se ha indicado que además de las máscaras de implantación N y P se debe construir una máscara de "Zonas Activas" para definir los transistores, sin embargo en los layouts de las celdas de biblioteca del sistema TENTOS "no existen Zonas Activas", en estas condiciones los programas de verificación circuital no detectan transistores, y peor aun, en caso de fundirse estos circuitos, los transistores no se formarían y el chip se inutilizaría totalmente.

Este error no puede ser detectado por el programa DARC ya que no se trata de un error geométrico, sino de un error de concepción del diseño.

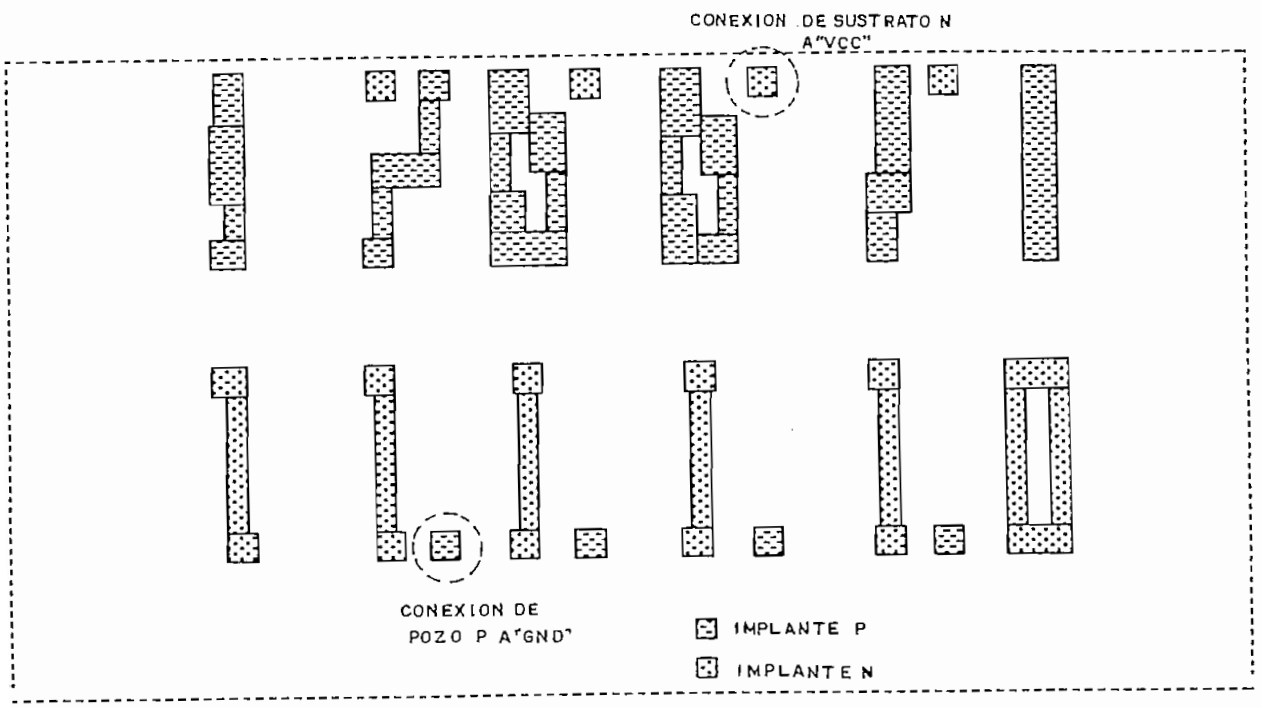


FIGURA 7.16 Layout del circuito MED_SUM generado automáticamente (no existen zonas activas).

En la Fig.7.16 se exhibe el layout generado automáticamente para el circuito MED_SUM en que se observan zonas de implante N+ y P+, pero no existen zonas activas.

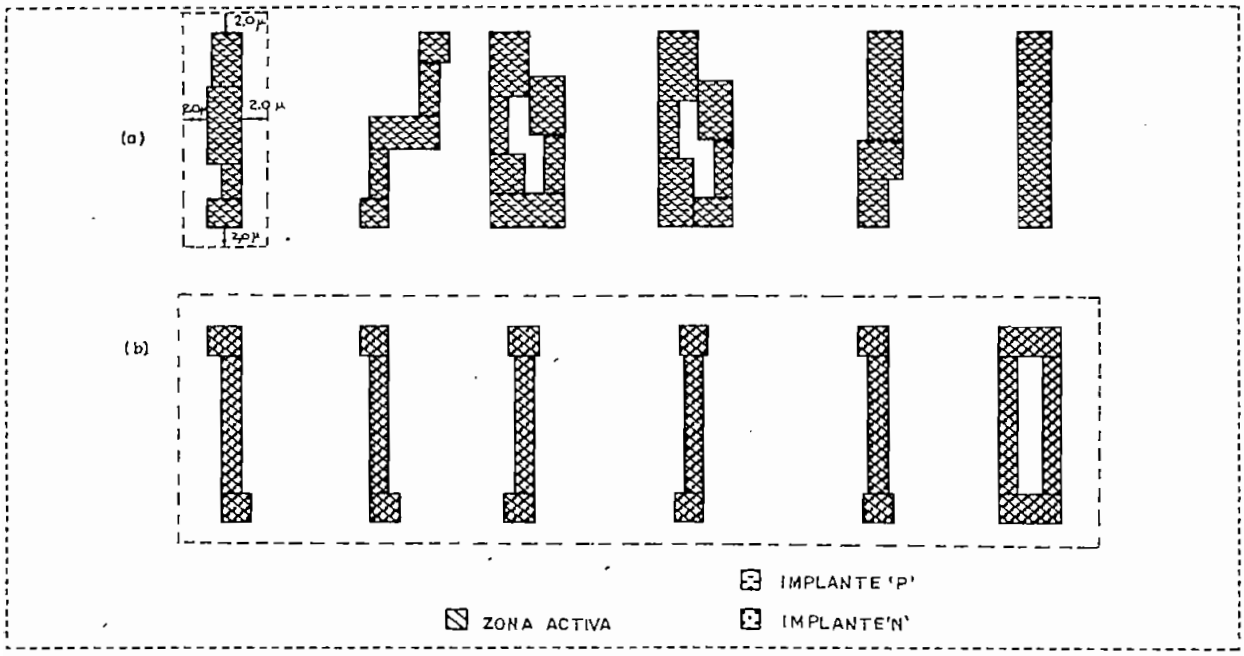


FIGURA 7.17 Layout del circuito MED_SUM en que se indican las posibilidades de definición de zonas activas.

Para la corrección del layout debe definirse una máscara de zonas activa, para ello se tienen las siguientes posibilidades:

iv.1) De acuerdo a las reglas de diseño 603 y 604 del ES2[B] (ver Anexo G), las zonas de difusión N+ y P+ deben sobrelapar a las zonas activas con un margen mínimo de 2µm, por lo que se podría definir las zonas activas en las posiciones originales de

[B] "Dual Layer Metal 1.5µm CMOS Design Rules (ECPD15/1)", European Silicon Structures ES2, pág 9.

los implantes, y ensanchar las zonas de implante cubriendo a las zonas activas como se indica en la Fig.7.17 (a).

iv.2) También se podría definir una sola zona de implante N+ como se indica en la Fig.7.17 (b), una zona de implante P+ sobre el Pozo N, y las zonas activas sobre las posiciones originales de los implantes N+ y P+.

iv.3) Una tercera opción sería considerando la regla 607 del ES2 (Anexo G) en que se impone una zona de coincidencia mínima de 2.0µm entre las zonas activas y los implantes en el caso de que no haya sobrelapamiento de los implantes sobre las zonas activas. En este caso se podrían superponer las zonas activas a los implantes siempre que no haya zonas comunes con longitud inferior a 2.0µm como en efecto sucede.

La opción (iv.1) implicaría una redefinición de todos los implantes que sería posible en el caso del circuito MED_SUM por ser pequeño, pero sería demasiado complicada en el caso de un diseño de mayores proporciones. La opción (iv.2) es la que adopta el sistema TRAGO en que se definen sub-bandas de implantes y sobre estas las zonas activas como se indicó en el Numeral 6.2. Finalmente la última opción es la más sencilla puesto que únicamente se requiere copiar las

zonas de implante sobre las zonas activas, es esta la opción que se ha adoptado para la corrección de este error.

v) Formación de transistores indeseados

Finalmente, al definirse de manera automática las líneas de enrutamiento vertical se pueden producir superposiciones de capas de Polysilicon con las zonas de Implantación N+ y/o P+ de conexión a Vcc y Gnd para protección del "Latch-up" que originarían la formación de "posibles transistores" que no están planificados en el diseño (Fig.7.18).

Estos errores pueden o no ser detectados por el programa DARC del sistema TENTOS, ya que dependiendo de la disposición geométrica de la superposición de las máscaras pueden o no violarse reglas de distancias límites.

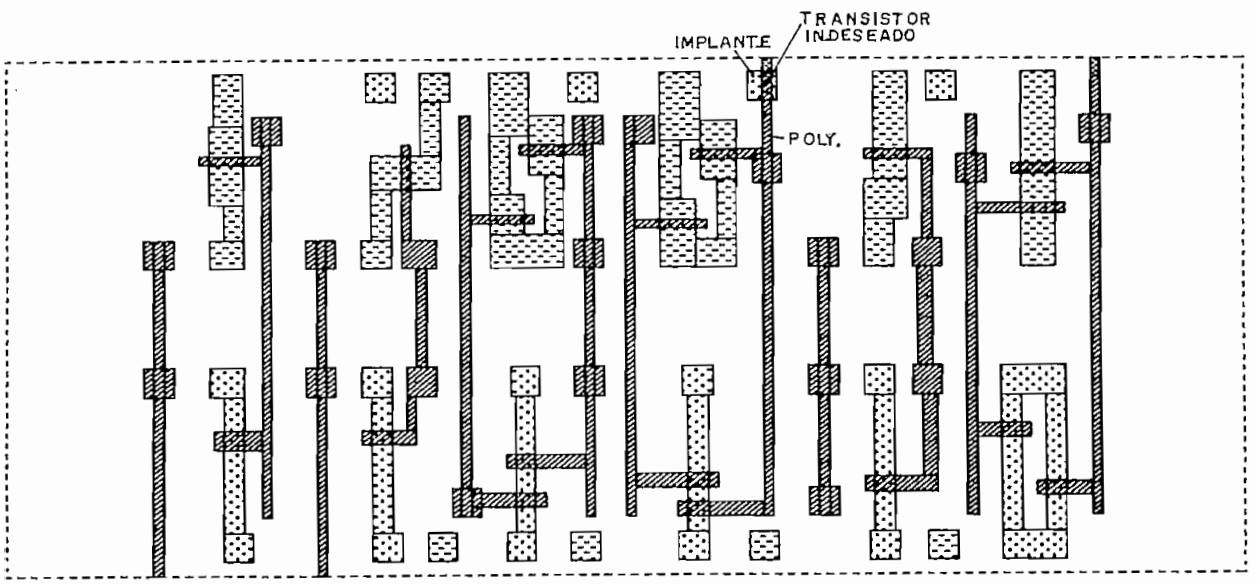


FIGURA 7.18 Formación de transistores erróneos en el layut del circuito MED_SUM.

Los transistores indeseados son detectados al extraerse el circuito equivalente del layout mediante la opción *PAC/EXTRIBO* que genera mensajes de alerta con respecto a transistores anormales encontrados en el layout.

Para corregir este error se traslada el rectángulo de una de las dos capas involucradas (Difusión o Polysilicon) a una posición en la que la superposición no se produzca, sin que varíe la estructura circuital del diseño.

Una vez corregidos los errores indicados, se obtiene el layout del circuito "Medio Sumador" que se indica en detalle en la Fig.7.19, esta descripción ha sido grabada en el archivo *MED_SUM.EXP*, compárese este layout con aquel de la Fig.7.9 generada automáticamente por el sistema TENTOS.

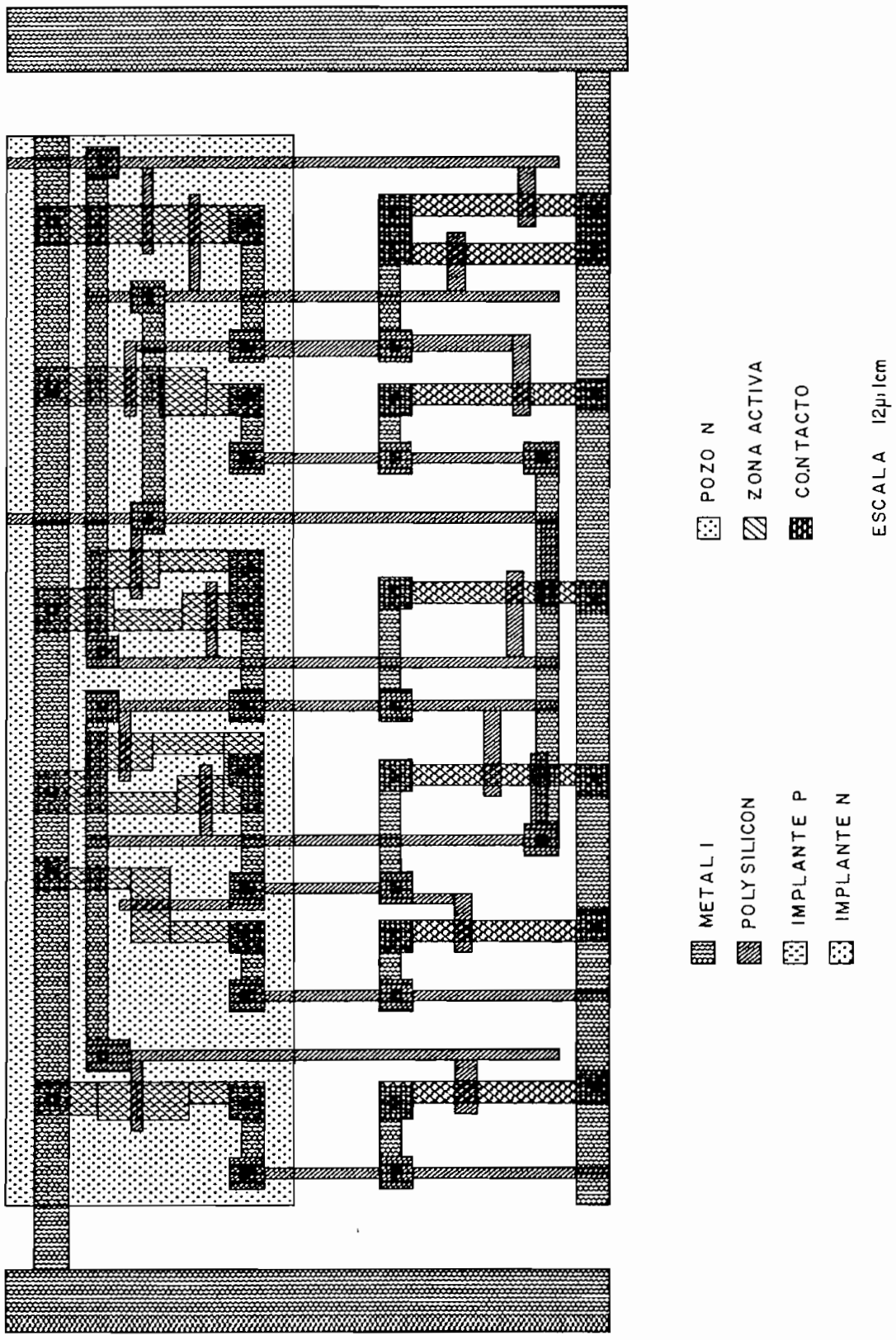


FIGURA 7.19 Layout del circuito MED_SUM en que se han corregido los errores de diseño.

b) Verificación eléctrica de las conexiones del layout

La estructura geométrica del layout del circuito MED_SUM debe interpretarse en forma circuital, a fin de analizar sus estructura de nodos y transistores, al margen de la funcionalidad lógica para la que el circuito fue concebido.

i) Extracción del circuito equivalente

El circuito eléctrico equivalente al layout descrito en el archivo *MED_SUM.EXP* (Fig.7.19) es extraído automáticamente en el sistema TENTOS mediante la opción *PAC/EXTRIBO*. El esquema circuital resultante es descrito en forma de NETLIST SPICE que es grabada en el archivo *MED_SUM.SIM*.

A fin de obtener una descripción circuital más ajustada a la realidad, el programa EXTRIBO define en la NETLIST SPICE, mediante el comando .MODEL (Anexo A), los modelos de los transistores según los parámetros dados por la fundidora (ES2) para la tecnología CMOS de 1.5µm, estos modelos se indican en el archivo de tecnología *CMOS15.TEC* (Anexo E).

La NETLIST SPICE generada por el programa EXTRIBO para el circuito de la Fig.7.9 es la siguiente:

```
* EXTRIBO: EXTRATOR HIERARQUICO DE CIRCUITOS VERSAO 3.2
* CPGCC - UFRGS - GRUPO DE MICROELETRONICA - MAS
*
* CIRCUITO: MED_SUM.CEL
* TECNOLOGIA: \gae\tec\cmos15.tec  TIPO CMOS
```

```
* Modelos dos transistores
.MODEL NMOS NMOS LEVEL=2 LD=0.325U TOX=250E-10 NSUB=2E16 VTO=0.7 UO=510
+UEXP=0.22 UCRIT=24.3K DELTA=0.4 XJ=0.4U VMAX=54K NEFF=4 RSH=55 NFS=0 JS=2U
+CJ=130U CJSW=620P MJ=0.53 MJSW=0.53 PB=0.68V CGDO=320P CGSO=320P
*
.MODEL PMOS PMOS LEVEL=2 LD=0.3U TOX=250E-10 NSUB=5E16 VTO=-1.1 UO=210
+UEXP=0.33 UCRIT=51K DELTA=0.4 XJ=0.5U VMAX=47K NEFF=0.88 RSH=75 NFS=0 JS=10U
+CJ=490U CJSW=590P MJ=0.46 MJSW=0.46 PB=0.78V CGDO=320P CGSO=320P
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  MED_SUM
*
* Transistores tipo NMOS: 9
* Transistores tipo PMOS: 9
MN1    11  10   0   0 NMOS L=30.2U W=40.3U AD=17400P AS=18200P PD=860U PS=900U
MP2     2  10   1   1 PMOS L=20.1U W=70.5U AD=4200P AS=13300P PD=260U PS=520U
MP3    11  12   2   1 PMOS L=20.1U W=70.5U AD=7700P AS=4200P PD=360U PS=260U
MN4    11  12   0   0 NMOS L=30.2U W=40.3U AD=17400P AS=18200P PD=860U PS=900U
MN5     9  11   0   0 NMOS L=30.2U W=40.3U AD=10400P AS=6800P PD=580U PS=400U
MP6     9  11   1   1 PMOS L=20.1U W=70.5U AD=16000P AS=11200P PD=620U PS=460U
MP7     7  12   1   1 PMOS L=20.1U W=70.5U AD=18000P AS=23000P PD=840U PS=1000U
MN8     5  12   0   0 NMOS L=30.2U W=40.3U AD=1200P AS=4800P PD=140U PS=300U
MN9     7  10   5   0 NMOS L=30.2U W=40.3U AD=1200P AS=10000P PD=140U PS=560U
MP10    7  10   1   1 PMOS L=20.1U W=70.5U AD=18000P AS=23000P PD=840U PS=1000U
MP11    8   7   1   1 PMOS L=20.1U W=70.5U AD=19500P AS=22200P PD=860U PS=980U
MN12    8   7   6   0 NMOS L=30.2U W=40.3U AD=2000P AS=8400P PD=180U PS=480U
MN13    6   9   0   0 NMOS L=30.2U W=40.3U AD=2000P AS=5600P PD=180U PS=340U
MP14    8   9   1   1 PMOS L=20.1U W=70.5U AD=19500P AS=22200P PD=860U PS=980U
MP15    4   8   1   1 PMOS L=20.1U W=70.5U AD=11800P AS=12200P PD=620U PS=640U
MN16    4   8   0   0 NMOS L=30.2U W=40.3U AD=6400P AS=10800P PD=380U PS=600U
MN17    3   7   0   0 NMOS L=40.2U W=40.3U AD=7000P AS=10400P PD=400U PS=580U
MP18    3   7   1   1 PMOS L=20.1U W=70.5U AD=12600P AS=10800P PD=620U PS=480U
.END
```

Cabe destacar que a diferencia de la NETLIST SPICE generada a partir de la descripción esquemática (numeral 7.2.2) esta NETLIST no se basa en archivos de biblioteca predefinidos, sino que es obtenida directamente a partir del layout plano del archivo *MED_SUM.EXP*, debido a ello tiene únicamente un circuito principal sin subcircuitos.

Además, únicemente fuera del ambiente TENTOS se puede ejecutar el programa EXTRIBO, con la opción "-E" (comando desde el DOS: EX7 -E) para generar un layout que se graba en un archivo *MED_SUM.ELE* con la numeración de transistores y

nodos asignada a cada capa de acuerdo con la numeración de la NETLIST SPICE del archivo *MED_SUM.SIM* también generada por el EXTRIBO. El layout contenido en el archivo *MED_SUM.ELE* se exhibe en las Fig.7.20.

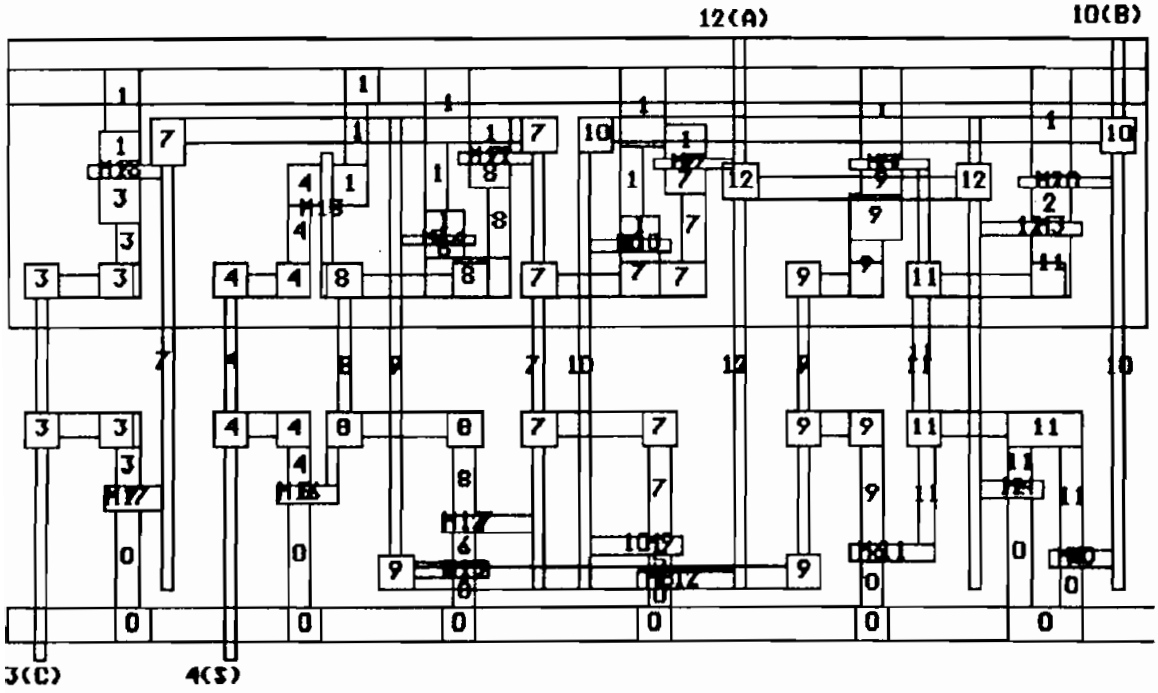


FIGURA 7.20 Layout del circuito MED_SUM con su estructura de nodos generado por el programa EXTRIBO.

ii) Verificación de interconexiones

En base al circuito equivalente del layout del archivo *MED_SUM.EXP* se pueden identificar los elementos lógicos (celdas básicas) que contiene y analizar las relaciones de conexión entre éstos, además de identificar los transistores que las forman. Para ello se ejecuta la opción *PAC/EXTRALO*, sus resultados se generan en el archivo *MED_SUM.REL* que se detalla a continuación:

```
ARQUIVO FONTE   MED_SUM.SIM
ARQUIVO DESTINO MED_SUM.REL
Numero de celulas 6

CEL  0 : INV_1
MN17  (-)  MP18
entrada        7
Saida    3

CEL  1 : INV_1
MN16  (-)  MP15
entrada        8
Saida    4

CEL  4 : INV_1
MN5   (-)  MP6
entrada       11
Saida    9
celulas conectados:   2

CEL  2 : NAND_2
MN13 MN12   (-)   ( MP14 MP11 )
entrada        9       7
Saida    8
celulas conectados:   1

CEL  3 : NAND_2
MN9 MN8   (-)   ( MP10 MP7 )
entrada       10       12
Saida    7
celulas conectados:   2   0

CEL  5 : NOR_2
( MN4 MN1 )   (-)   MP2 MP3
entrada       12       10
Saida   11
celulas conectados:   4

TRANSISTORES SOLTOS
```

La información de este archivo ha sido interpretada en forma esquemática como se indica en la Fig.7.21.

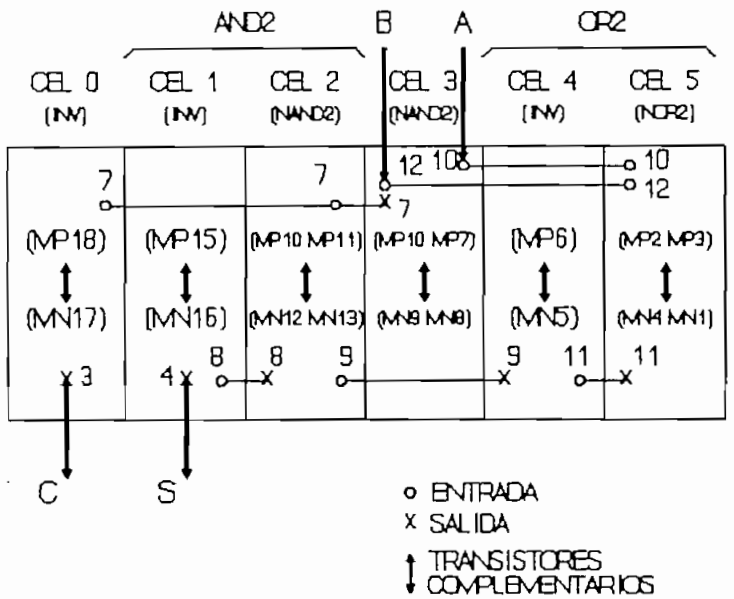


FIGURA 7.21 Interpretación de la información generada por la opción EXTRALO.

La información obtenida en el archivo *MED_SUM.REL* e interpretada en la Fig.7.21 permite:

ii.1) Identificar los nodos de prueba para realizar las verificaciones funcionales posteriores.

ii.2) Verificar la corrección en los enlaces ya que en el proceso de síntesis o de corrección de errores pudieron haberse omitido o eliminado conexiones.

ii.3) Detectar, en el caso de la tecnología CMOS, la presencia de transistores que pudieron haberse formado accidentalmente en el proceso de generación del layout, ya que estos transistores no formarían parte de celda básica alguna o no tendrían su complementario. Nótese que al final del archivo existe el mensaje "TRANSISTORES SOLTOS" en que se identifican los transistores con algún nodo anormal o sin su correspondiente complementario.

c) Verificación funcional del layout (simulación)

La simulación del circuito equivalente del layout del circuito MED_SUM permitirá evaluar su respuesta frente a las exitaciones en sus terminales de entrada, y confrontar su comportamiento lógico con aquel planteado en el diseño a nivel funcional.

La simulación del layout del circuito MED_SUM será realizada desde dos puntos de vista diferentes:

i) Simulación eléctrica SPICE.

ii) Simulación lógica NDL.

i) Simulación eléctrica SPICE

La simulación eléctrica SPICE es realizada mediante la opción *SiMulacao/SPICE* del interfaz del sistema TENTOS.

En este caso ha sido posible la simulación SPICE del layout por tratarse de un circuito pequeño, sin embargo, en circuitos correspondientes a layouts de mayor proporción este tipo de simulación ya no será posible debido a la cantidad de memoria y tiempo de procesamiento que se requiere.

La NETLIST SPICE del archivo *MED_SUM.SIM*, que describe al circuito equivalente del layout no es suficiente para la realización de la simulación SPICE puesto que como se vió en el Numeral 5.3, se deben incluir en la NETLIST:

i.1) Fuentes de señales y de polarización.
i.2) Directivas de simulación.

Los terminales de polarización en el diagrama de la Fig. 7.20 tienen la numeración "1" y "0". Para las fuentes de señal se tiene que los nodos de entrada al circuito han sido numerados como "12" y "10" que corresponden a los terminales "A" y "B" del diseño esquemático y los nodos de salida han sido numerados como "3" y "4" que corresponden a los terminales "C" y "S", respectivamente.

En consecuencia, las fuentes de señal se conectarán al circuito como se indica en la Fig.7.22.

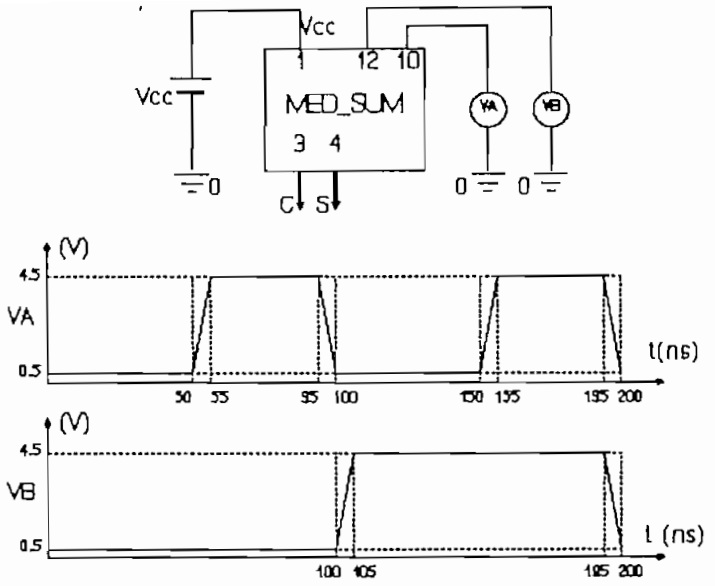


FIGURA 7.22 Conexión de las fuentes en el layout del circuito MED_SUM.

Por tanto, en la NETLIST SPICE anterior se deberán añadir las líneas:

```
* POLARIZACION
VCC 1 0 DC 5V
* SEÑALES DE ENTRADA
VA 12 0 PULSE(0.5V 4.5V 50NS 5NS 5NS 40NS 100NS)
VB 10 0 PULSE(0.5V 4.5V 100NS 5NS 5NS 90NS 200NS)
```

De acuerdo a las fuentes conectadas, para la simulación del circuito se requiere el análisis de su respuesta transitoria entre 0 y 400ns (comando .TRAN), también se ha instruído la generación gráfica de los resultados (comando .PROBE).

```
* DIRECTIVAS DE SIMULACION
.TRAN 10NS 400NS
.PROBE
.END
```

Los resultados de esta simulación se generan en el archivo *PROBE.DAT* y pueden ser visualizados mediante la opción *SIMULACION/PROBE*. El gráfico así obtenido se exhibe en la Fig.7.23.

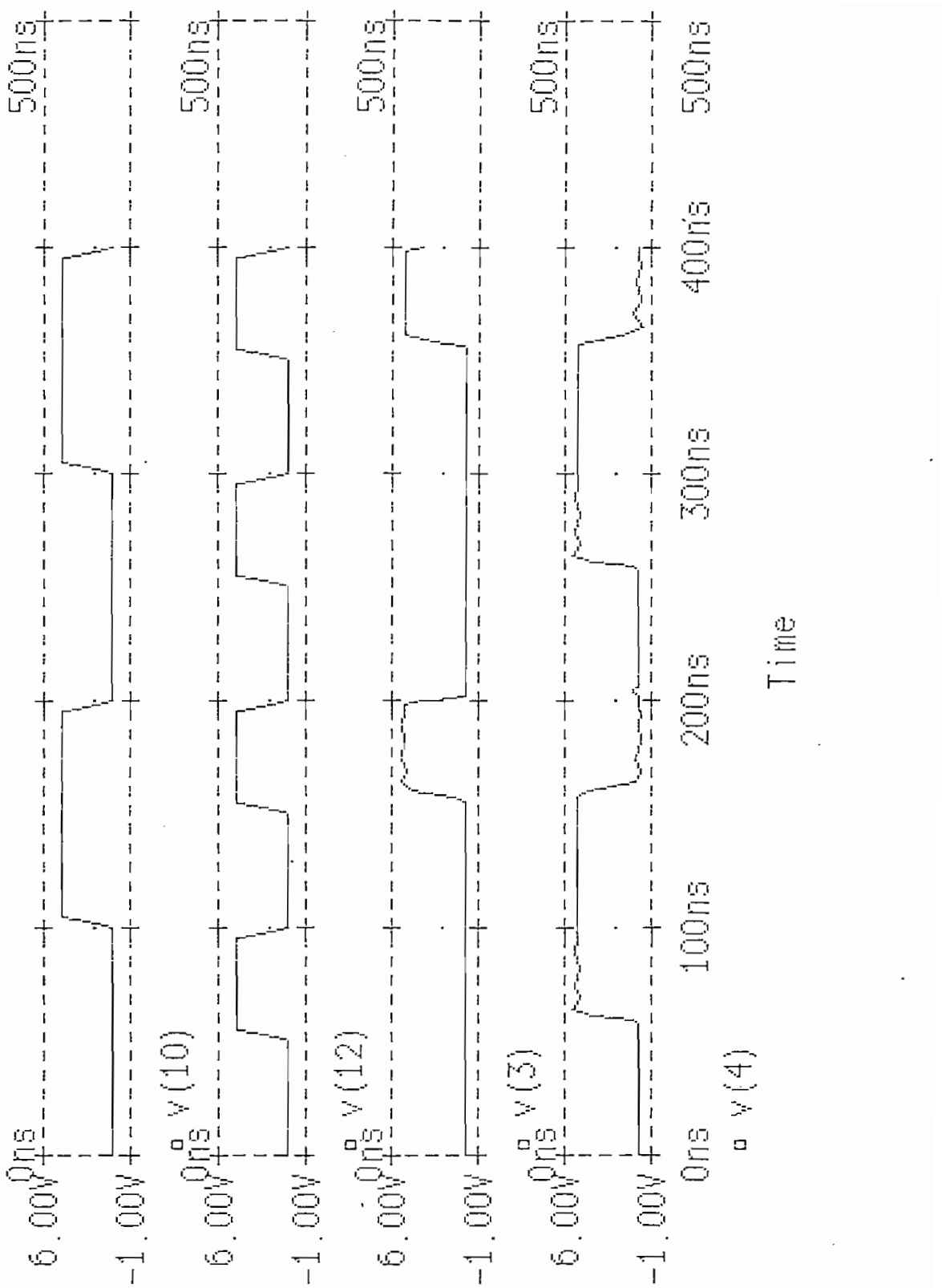


FIGURA 7.23 Edición gráfica de la simulación SPICE del circuito MED_SUM.

i) Simulación lógica NDL

Puesto que la simulación lógica del circuito MED_SUM no puede ser realizada con el sistema TENTOS. Se ha empleado el simulador lógico NDL.

La NETLIST NDL para la simulación del circuito MED_SUM ha sido obtenida a partir de la NETLIST SPICE del archivo *MED_SUM.SIM* indicado anteriormente en forma manual mediante un editor de texto.

El esquema circuital con la configuración de nodos modificada según las restricciones del lenguage NDL para el layout del circuito MED_SUM se exhibe en la Fig.7.24. (Recuérdese que la nominación de los nodos en el lenguage NDL debe iniciarse en un caracter).

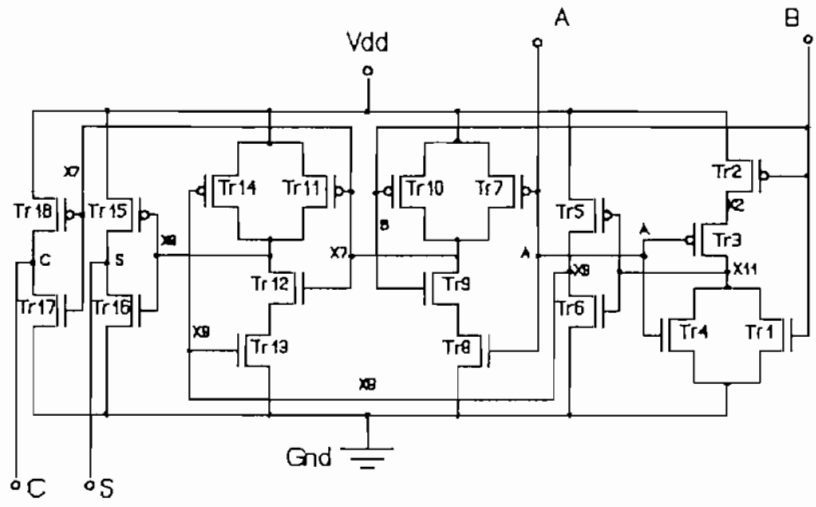


FIGURA 7.24 Esquema circuital del layout del circuito MED_SUM con la nominación de nodos empleada en la NETLIST NDL.

La NETLIST NDL correespondiente al circuito de la Fig.7.24 es la siguiente:

```
%-------------------------------------------------------------
%  NETLIST NDL EQUIVALENTE DEL LAYOUT DEL CIRCUITO (MED_SUM)   ¦
%              FREDY LEMUS C./IVAN BERNAL C.                   ¦
%-------------------------------------------------------------
%  TRANSISTORES NMOS: 9
%  TRANSISTORES PMOS: 9
FAMILY CMOS;
INPUT A B;
       NTRANS    A   gnd x11;% Tr_1
       PTRANS    A   vdd x2 ;% Tr_2
       PTRANS    B   x2  x11;% Tr_3
       NTRANS    B   gnd x11;% Tr_4
       NTRANS    x11 gnd x9 ;% Tr_5
       PTRANS    x11 vdd x9 ;% Tr_6
       PTRANS    B   vdd x7 ;% Tr_7
       NTRANS    B   gnd x5 ;% Tr_8
       NTRANS    A   x5  x7 ;% Tr_9
       PTRANS    A   vdd x7 ;% Tr_10
       PTRANS    x7  vdd x8 ;% Tr_11
       NTRANS    x7  x6  x8 ;% Tr_12
       NTRANS    x9  gnd x6 ;% Tr_13
       PTRANS    x9  vdd x8 ;% Tr_14
       PTRANS    x8  vdd x4 ;% Tr_15
       NTRANS    x8  gnd x4 ;% Tr_16
       NTRANS    x7  gnd x3 ;% Tr_17
       PTRANS    x7  vdd x3 ;% Tr_18
END
```

Los comandos para la simulación lógica de este circuito son los mismos del archivo *COMANDOS.SIM* que fueron empleados en la simulación del diseño esquemático, y se reproducen a continuación:

```
%  POLARIZACION
LIGUE Vdd 1
LIGUE Gnd 0

%  DATOS DE ENTRADA
ENTRADA A 0011
ENTRADA B 0101

%  DIRECTIVAS DE CONTROL
GRAFICO
PLUGUE A 1
PLUGUE B 2
PLUGUE C 4
PLUGUE S 6
FASE  16
```

Los resultados de la simulación lógica del layout del circuito MED_SUM se exhiben en la Fig.7.25.

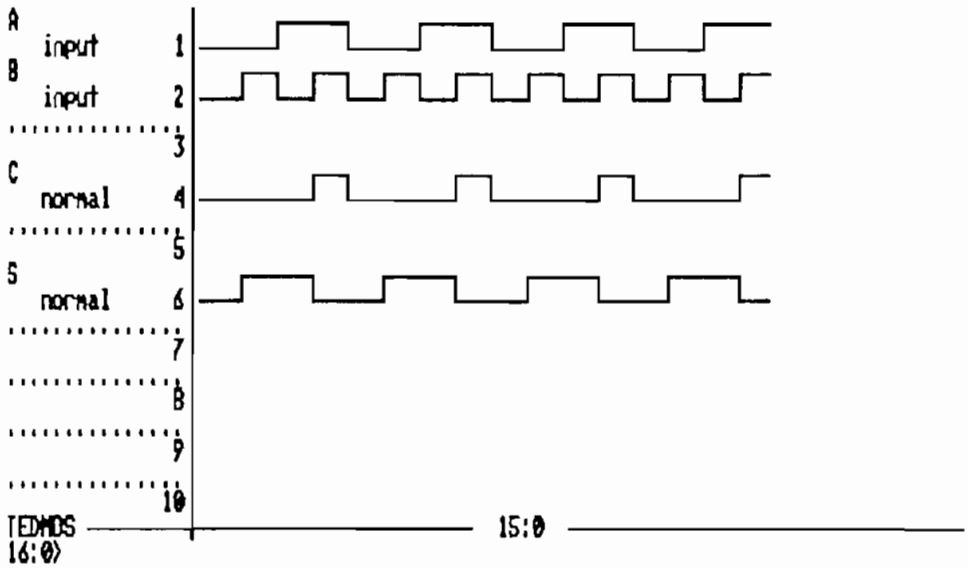


FIGURA 7.25 Resultados de la simulación NDL del layout del circuito MED_SUM.

Del análisis de los resultados de la simulación se concluye que:

i) Los resultados de la simulación eléctrica SPICE (Fig.7.23) expresan el mismo contenido lógico que los resultados de la simulación lógica NDL (Fig.7.25).

ii) Los resultados de la simulación del diseño al nivel físico concuerdan con aquellos de la simulación del diseño al nivel esquemático (Fig.7.4).

iii) Los resultados de la simulación del diseño al nivel físico y estructural cumplen con los requerimientos planteados en el diseño al nivel funcional en la Tabla 7.1.

En consecuencia:

"El layout diseñado para el circuito MED_SUM (Fig.7.29) es correcto y cumple con los requerimientos planteados para el ASIC del circuito Medio Sumador ó Sumador de un Bit".

7.3.4 Integración de PADs al layout del circuito MED SUM

El layout completo de un circuito integrado está formado por el núcleo y un anillo de PADs como se indica en la Fig.7.26.

FIGURA 7.26 Estructura general del layout de un CI.

El núcleo del circuito MED_SUM corresponde al layout de la Fig.7.19. El anillo de PADsS constituye la zona que abarca los PADs y sus interconexiones que rodean al núcleo del CI.

En el presente literal se describe la forma como se han seleccionado los PADs, y las interconexiones necesarias para integrarlos al layout global del circuito integrado MED_SUM.

a) Selección de PADs

El layout del circuito MED_SUM tiene la estructura de terminales de la Fig.7.27, cada uno de estos terminales debe ser accedido externamente a través de un PAD diferente, diseñado específicamente para cada propósito (entrada, salida, entrada/salida, polarización, etc..).

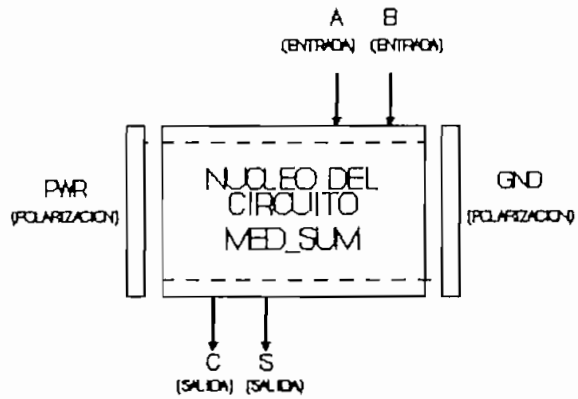


FIGURA 7.27 Estructura de terminales del circuito MED_SUM.

Puesto que el sistema TENTOS carece de biblioteca de PADs, se ha empleado la biblioteca: **PADLIB4 (CIF) Versión: PADLIB2-ECPD15-1.4**[6] cuya estructura se halla de acuerdo con las reglas de la fundidora ES2 para el proceso CMOS de 1.5µm (ECPD15/1).

---

[6] "ES2 ECPD15 & ECPD12 Library Databook, Standard Cell Libraries", European Silicon Structures (ES2), pág 1-17.

Para el circuito MED_SUM han sido seleccionados los siguientes PADs cuya estructura y características eléctricas y geométricas se detallan en el Anexo F.2.

NOTA:

Los layouts de los PADs que se indican en el Anexo F.2, únicamente corresponden a las máscaras de los terminales de acceso a los PADs y no a su layout completo, esta información es reservada por el fabricante. Además su descripción original se halla en formato CIF por lo que han debido ser traducidos al formato RS para su uso en el Editor de Máscaras EMA2.

i) PADs de entrada y salida

A fin de dar al Chip diseñado una característica de trabajo acorde con la tecnología TTL se ha optado por el uso de:

PADs de entrada

IPS8G: TTL Input Buffer (REV.1.4/JUNE 90)

PADs de salida

OPS1U: 4mA Output Buffer (REV.1.4/JUNE 90)

ii) PADs de polarización

Las celdas de alimentación y tierra de la biblioteca PADLIB4 empleada son:

PWRPY: Alimentación para etapas de alimentación solo de celdas de salida.

GNDPY: Tierra para etapas de alimentación solo de celdas de salida.

PWRCO: Alimentación para el núcleo y prebuffers de celdas de E/S y decodificadores.

GNDCO: Tierra para el núcleo, bloques y prebuffers de celdas de E/S y decodificadores.

PWRBK: Celda de alimentación opcional para bloques.

Para la formulación de los PADs de polarización se deben seguir los siguientes principios básicos:

> "i) Todas las partes del dispositivo deben ser polarizadas por lo que debe haber al menos un par de celdas PWR.. y GND..
>
> ii) Debe haber un par de celdas PWRCO y GNDCO por cada 40 mm² de area del núcleo, o 64 celdas de E/S.
>
> Esta configuración tiene el beneficio de aislar el núcleo del layout del ruido de conmutación en el buffer de salida, ya que los buffers de entrada salida son desacoplados. Los rebotes en alimentación y tierra debido a la conmutación en la salida y la inductancia en el chip y en sus enlaces tienen efectos mínimos en los niveles de entrada y en el núcleo."(7)

El núcleo del circuito MED_SUM tiene una área de 0.026mm² (0.247mm x 0.105mm), 2 celdas de entrada y 2 celdas de salida, es decir 4 celdas de E/S por lo que será suficiente emplear las celdas PWRPY y GNDPY para polarizar al anillo de PADs y al núcleo del CI simultáneamente.

---

(7) "ES2 ECPD15 & ECPD12 Library Databook, Standard Cell Libraries", European Silicon Structures (ES2), pág 1-8.

En consecuencia la configuración de PADs del circuito MED_SUM es la que se indica en la Fig. 7.28.

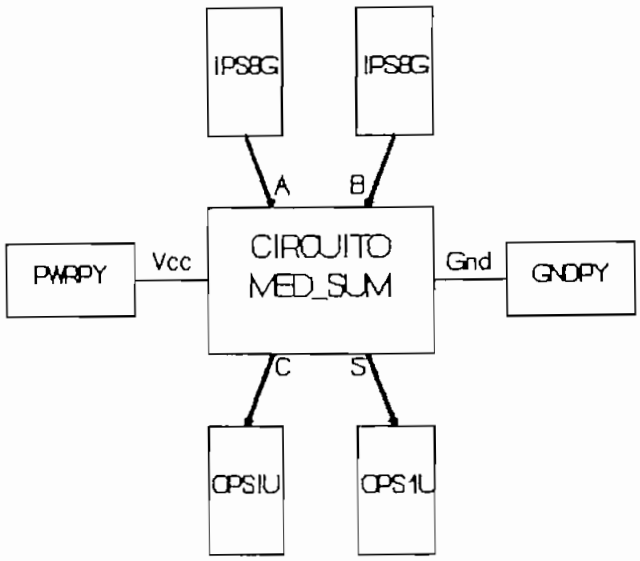


FIGURA 7.28 Estructura de PADs del circuito MED_SUM.

b) Interconexión de PADs

La integración de los PADs al layout del CI se realiza al nivel de diseño físico en forma manual mediante el editor de máscaras EMA2, ya que el sistema TENTOS no posee herramienta que realice esta tarea automáticamente.

Como ya se indicó, las celdas de los PADs únicamente indican sus terminales de acceso, no obstante, esta información es suficiente para interconectarlos al CI. Como se verá posteriormente, la integración total de los PADs la realiza el fabricante en base a los terminales indicados en el diseño.

Para la ubicación y conexión de PADs se deben seguir los siguientes lineamientos básicos:

"Ubicación de PADS

i) Verificar la correcta conexión entre sí de aquellos PADs que se ubiquen de forma adyacente.

ii) Cuando un PAD y su vecino estén separados una cierta distancia, realizar todas las conexiones laterales entre ambos PADs, respetando las dimensiones de los pines a conectar y el tipo de layer (normalmente Metal2)

iii) Verificar que en cada uno de los cuadro lados del circuito, los PADs y/o líneas de metal estén bien alineadas" (8)

De acuerdo a ello, el proceso empleado para la interconexión de PADs es el siguiente:

i) Inicialmente, se colocan los PADs perfectamente alineados y distribuídos simétricamente en torno al núcleo del circuito, es de especial importancia la simetría en la distribución de los PADs de polarización ya que estos serán empleados para polarizar tanto al anillo como al núcleo.

ii) Se realizan las conexiones entre los terminales del núcleo y los terminales de acceso a los PADs de E/S, estas conexiones pueden hacerse en Metal1 ó en Metal2 ya que los terminales de acceso a los PADs son hechos en base a estos dos materiales. La distribución de PADs en

(8) "Servicio MPC - Especificaciones de participación -. Restricciones Generales", CNM Barcelona - España, pág 2.

torno al núcleo del circuito MED_SUM y sus interconexiones con los terminales de E/S se indican en la Fig.7.29.

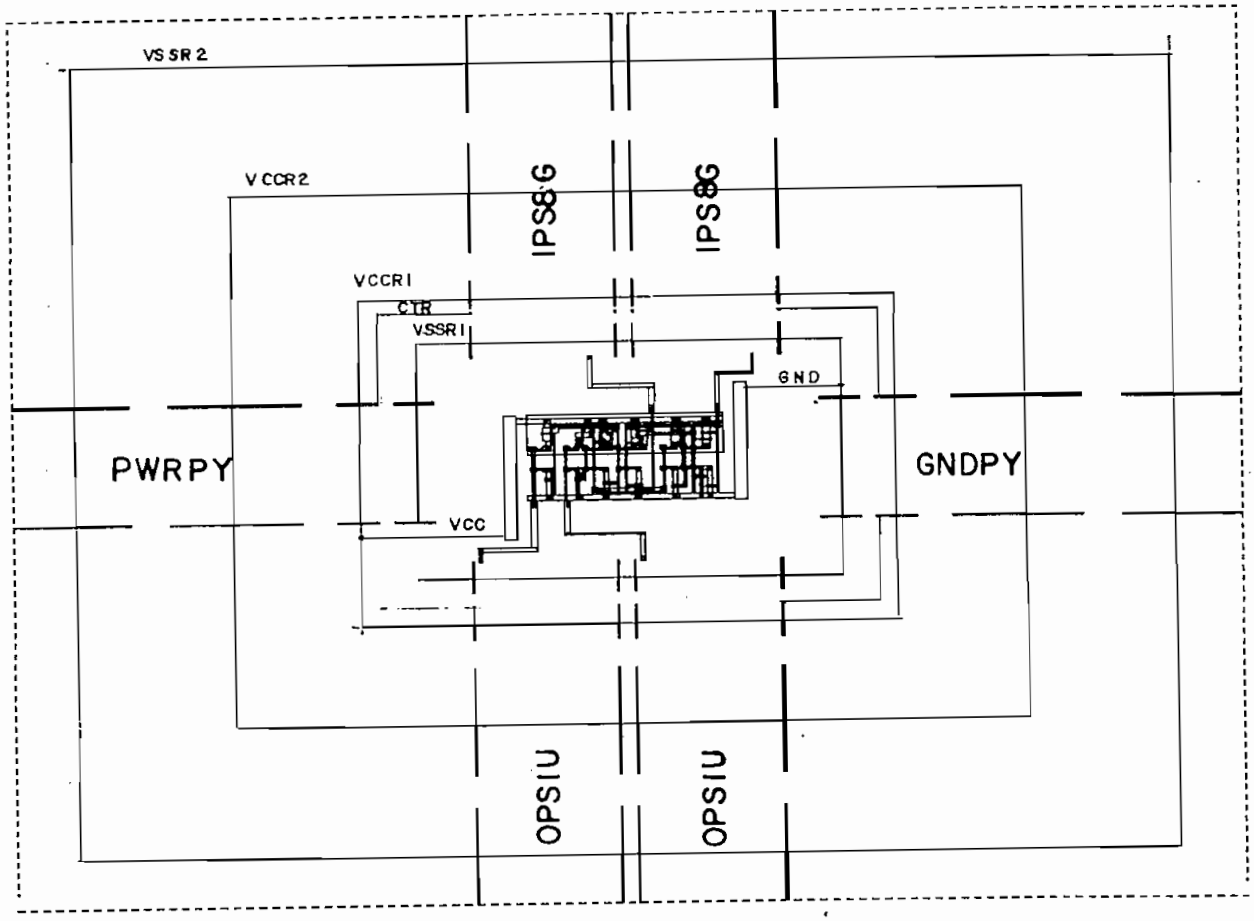


FIGURA 7.29 Distribución de PADs, anillo de interconexiones y enlaces entre PADs y núcleo.

iii) Se realiza la interconexión lateral de los PADs como se indica en la Fig.7.29 . Esta interconexión consiste en cuatro anillos (VSSR2, VCCR2, VCCR1, VSSR1) que deben

alinearse con los terminales de los PADs, además en las esquinas se deben realizar las uniones de los terminales CTR. Los dos anillos inferiores VCCR1 y VSSR1 se emplean para llevar las señales Vcc y Gnd, respectivamente al núcleo, nótese que el anillo de VSSR1 no se cierra totalmente a fin de permitir el acceso desde el núcleo al anillo VCCR1.

Los anillos de interconexión de PADs se construyen en Metal2 que es el material de acceso lateral a los PADs, para la realización de estos anillos diseñaron localmente las celdas que se exhiben en la Fig.7.30 tanto para la interconexión en las esquinas (Fig.7.30a) como para las interconexiones laterales (Fig.7.30b).

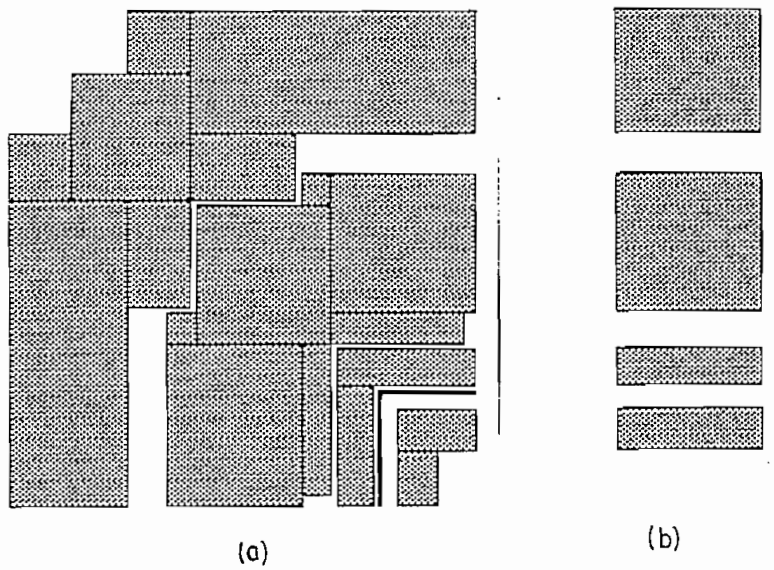


FIGURA 7.30 Celdas en Metal 2 para la interconexión en las esquinas (a) y lateral (b) de los PADs.

iv) Finalmente se generan las máscaras de polarización según el esquema de la Fig.7.29. Nótese que los terminales de

polarización del núcleo fueron implantados en Metal1 por lo que se requiere el uso de VIAS para el acceso de estos terminales a los anillos en Metal2.

El layout completo del circuito MED_SUM se exhibe en la Fig.7.31 y el detalle del núcleo con sus conexiones en base a Metal1, Metal2, VIAS y CONTACTOS se exhibe en la Fig.7.32.

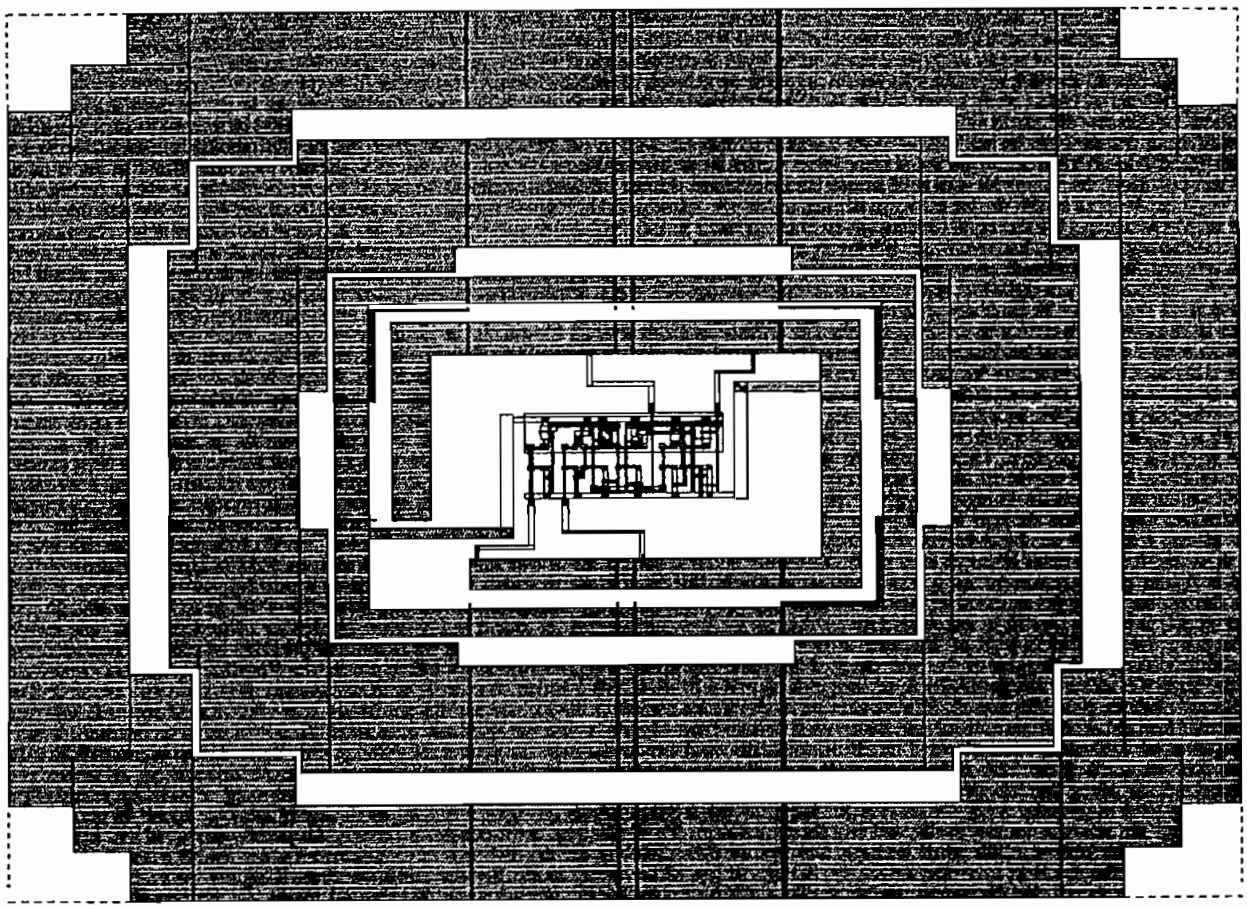


FIGURA 7.31 Layout completo del circuito MED_SUM.

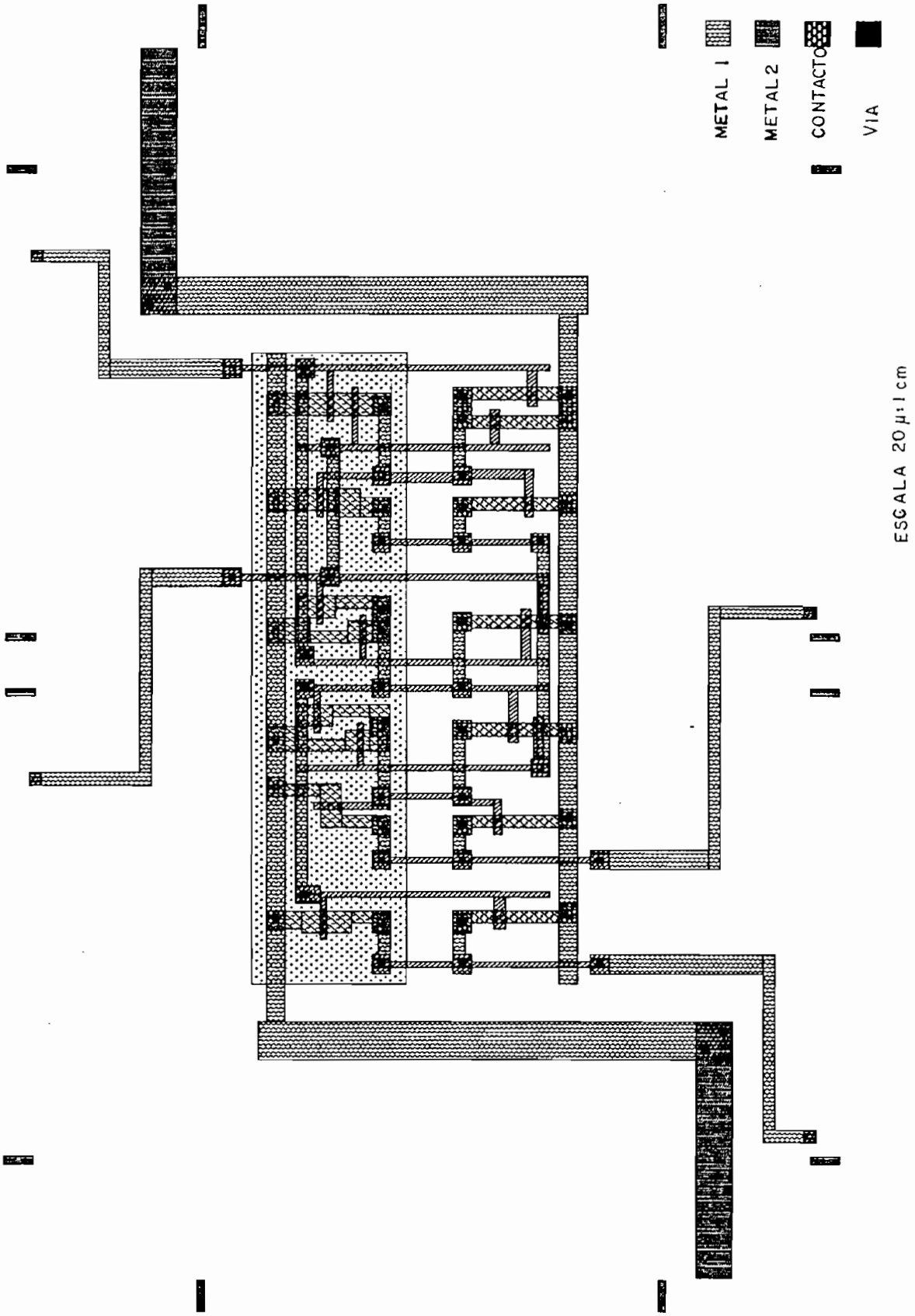


FIGURA 7.32 Detalle del núcleo del circuito MED_SUM con sus conexiones hacia los PADs.

7.3.5 Generación del archivo CIF

Puesto que se ha trabajado con el editor de máscaras EMA2, para la descripción del layout se ha empleado el lenguage RS, sin embargo de acuerdo con las restricción del CNM:

> "*Formatos de Descripción
> Los formatos más usuales para la descripción de los circuitos que se envían al MPC son GDSII y CIF"(*)

el diseño deberá transformarse al formato CIF, para ello se emplea la opción *PAC/CONV RS-CIF* que emplea como entrada el archivo *MED_SUM.EXP* y a partir de este, genera automáticamente al archivo *MED_SUM.CIF* que contiene la descripción del layout de la Fig.7.31 en formato CIF, su estructura jerárquica es la siguiente:

```
DS 102 1 1;
9 PRSGNDPY ;
'
'
DF;
DS 104 1 1;
9 PRSIPS8G ;
'
'
DF;
DS 103 1 1;
9 PRSOPS1U ;
'
'
DF;
DS 101 1 1;
9 PRSPWRPY ;
'
'
DF;
```

```
DS 105 1 1;
9 MED_SUM ;
C 101 T -2500 6800 R 0 1;
C 102 T -2500 31700 R 0 1 MX;
C 103 T 13200 7000 MY;
C 103 T -3300 7000 MY;
C 104 T 13200 17600;
C 104 T -3300 17600;
L CNWI;

9 NUCLEO DEL CIRCUITO;
'
'
'
'
DF;
C 105;
E
```

(*) "Servicio MPC - Especificaciones de Participación -. Restricciones Generales", CNM Barcelona-España, pág. 1.

La descripción anterior cumple con las normas de organización a que debe atenerse el archivo CIF enviado para su integración al UMP - Iberoamericano que son:

"i) Para permitir la portabilidad del diseño:
USAR SIEMPRE "RELATIVE PATHS".

ii) El directorio de trabajo contiene todas las celdas del circuito. Todas ellas son "hermanas" independientemente de la jerarquía interna del circuito.

iii) Debajo de cada celda o block solo pueden haber REFRESENTACIONES de esta celda y NO OTRAS CELDAS.

iv) Generar un fichero ordenado, es decir que ninguna celda debe ser utilizada (Call) antes de haber sido definida (DS)

v) La descripción CIF debe expresarse en las mismas unidades que las descripciones CIF de los PADs (normalmente en centésimas de micra.

vi) Las descripciones CIF del diseño completo sería conveniente que finalizacen de la siguiente forma:

```
.Nombre del diseño y del fichero: Top_Cell.CIF
.Parte final de la descripción CIF del diseño Top_Cell
.........
Ds n;
9 Top_Cell;
.........
DF:
C n;
E      "(10)
```

(10) "Servicio MPC - Especificaciones de Participación -. Restricciones Generales", CNM Barcelona - España, págs 3 y 8.

7.4 RESULTADOS DEL FUNCIONAMIENTO DE LOS PROTOTIPOS FABRICADOS EN LA FUNDIDORA ES2 (Francia).

El circuito MED_SUM desarrollado y probado en los literales anteriores fue enviado al Centro Nacional de Microelectrónica de Barcelona-España (CNM) para su revisión e integración al Chip Multi-Proyecto (MPC) a ser fundido en el Run de Diciembre de 1991 en la empresa European Silicon Structures (ES2) de Francia.

7.4.1 Integración del circuito MED_SUM al MPC

El circuito MED_SUM de la Fig.7.31 contiene:

a) Las interconexiones entre PADs

b) Las conexiones de acceso entre el núcleo del circuito y los PADs.

c) La descripción del núcleo del layout del circuito.

Pero no contiene la descripción misma de los PADs empleados.

En el CNM el circuito MED_SUM es modificado por la inclusión de los PADs en el layout del diseño de la Figura 7.31, dando lugar al "layout completo" del circuito MED_SUM de la Fig.7.33.

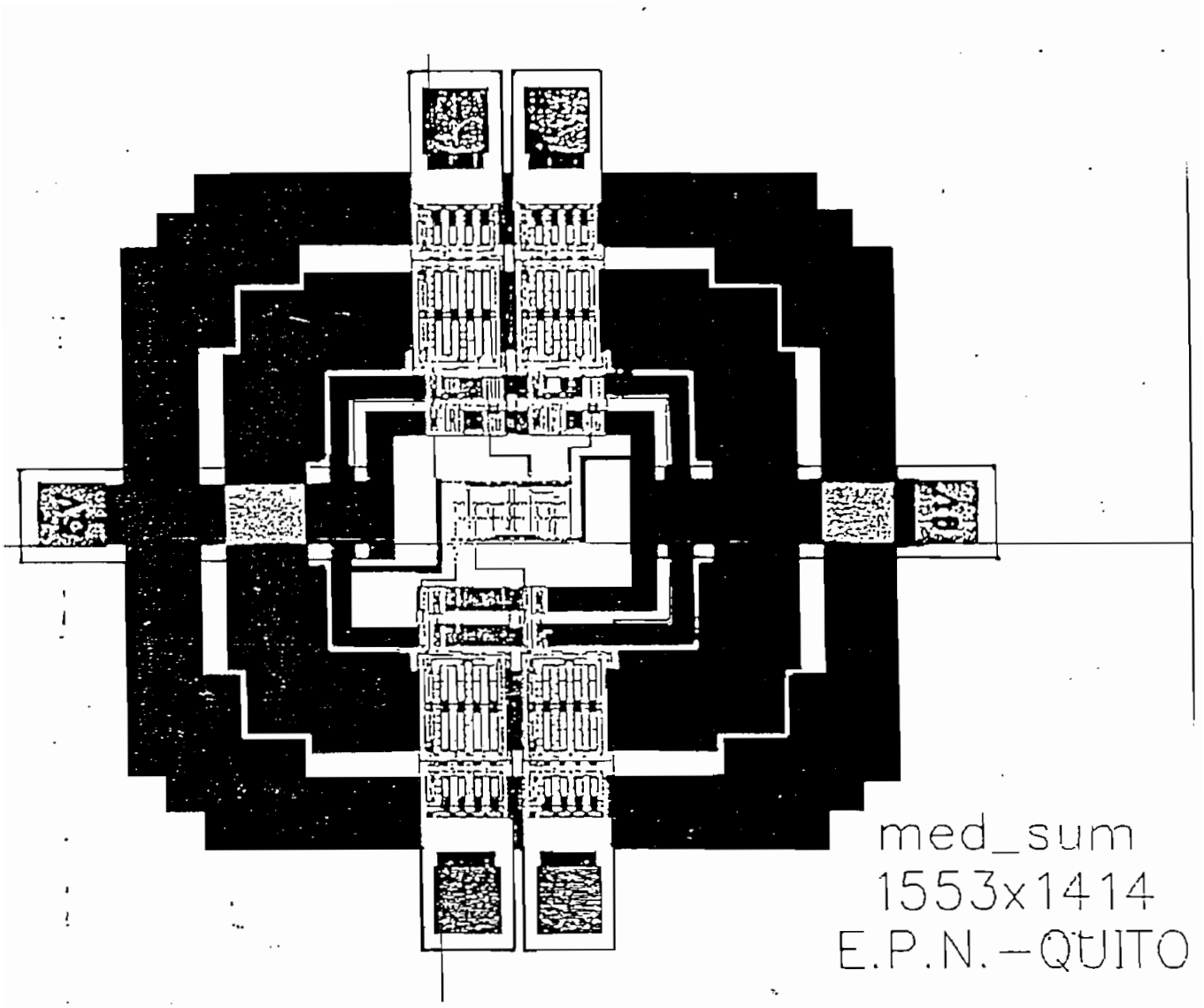


FIGURA 7.33 Layout completo del circuito MED_SUM (incluído PADs).

Puesto que el circuito MED_SUM forma parte de un Chip Multi-Proyecto (MPC), este es incluído para su fundición junto con los otros circuitos participantes en el proyecto. Además a cada PAD de los circuitos de la oblea se les asigna uno de los 24 pines externos del encapsulado.

La posición del circuito MED_SUM dentro de su Chip Multi-proyecto correspondiente y su asignación de pines en el encapsulado asignada por el CNM se grafica en la Fig. 7.34.

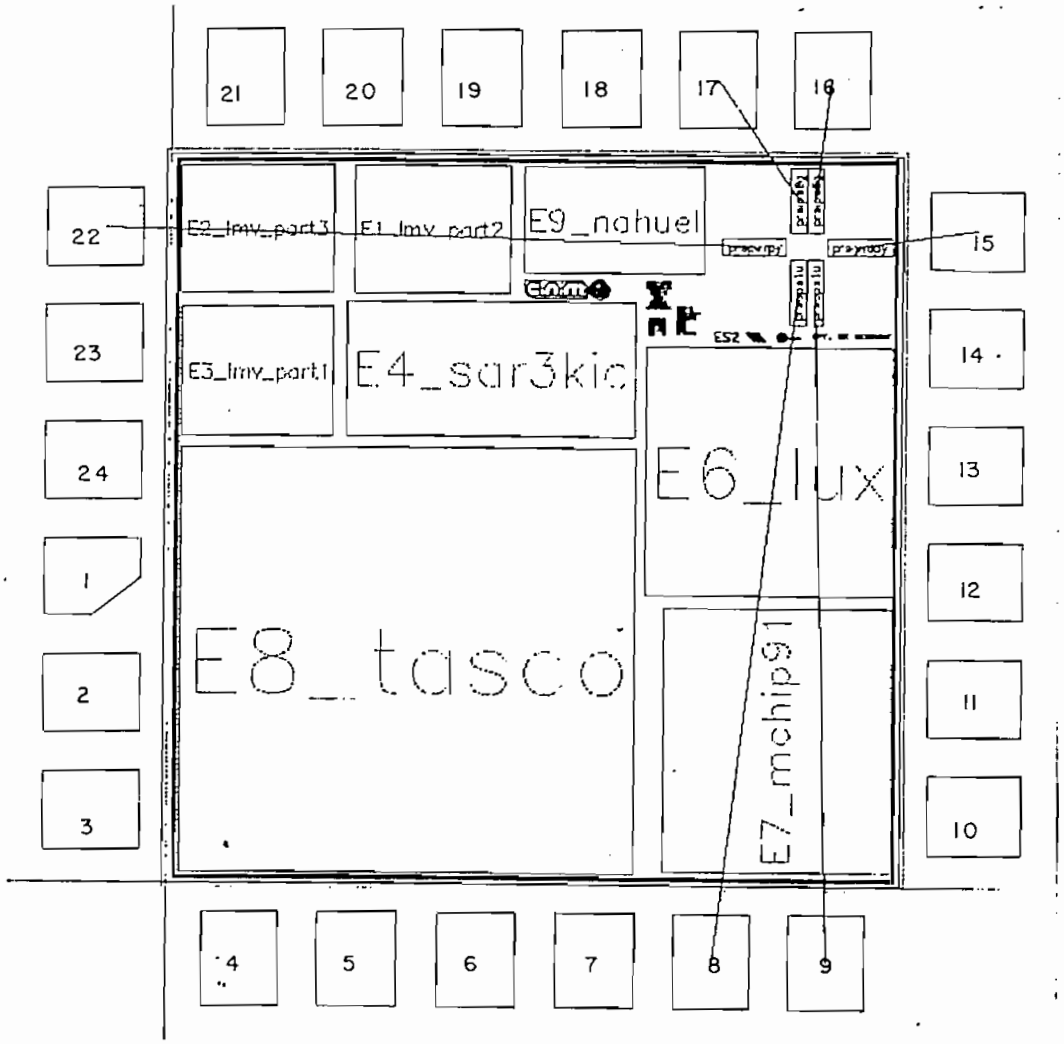


FIGURA 7.34 Posición del circuito MED_SUM dentro del MPC y su estructura de pines en el encapsulado.

En esta asignación de pines la correspondencia entre la numeración de pines y los terminales del circuito MED_SUM es la que se indica en la Tabla 7.2.

PINES DE ENCAPSULADO (NUMERACION)	TERMINALES DEL C.I. (NOMINACION)
8	C (Carry)
9	S (Suma)
15	GND
16	A
17	B
22	Vcc

TABLA 7.2 Numeración/Nominación de los pines del C.I. MED_SUM.

7.4.2 Prototipos del circuito MED SUM

El CNM entrega a cada uno de los diseñadores participantes en el proyecto cinco prototipos del chip multiproyecto. Los prototipos correspondientes al circuito "Medio Sumador ó Sumador de un Bit (MED_SUM)" fundidos en el Run CMOS de Diciembre de 1991 en la empresa Europepan Silicon Structures (ES2) en Francia, fueron recibidos por la EPN en el mes de Mayo de 1992.

La fotografía de la Fig.7.35 indica uno de los prototipos del chip multiproyecto que incluye el circuito MED_SUM.

La fotografía de la Fig.7.36 muestra el "dado" que comparten los CIs agrupados en el Chip Multiproyecto, compárese esta fotografía con el esquema de la Figura 7.34

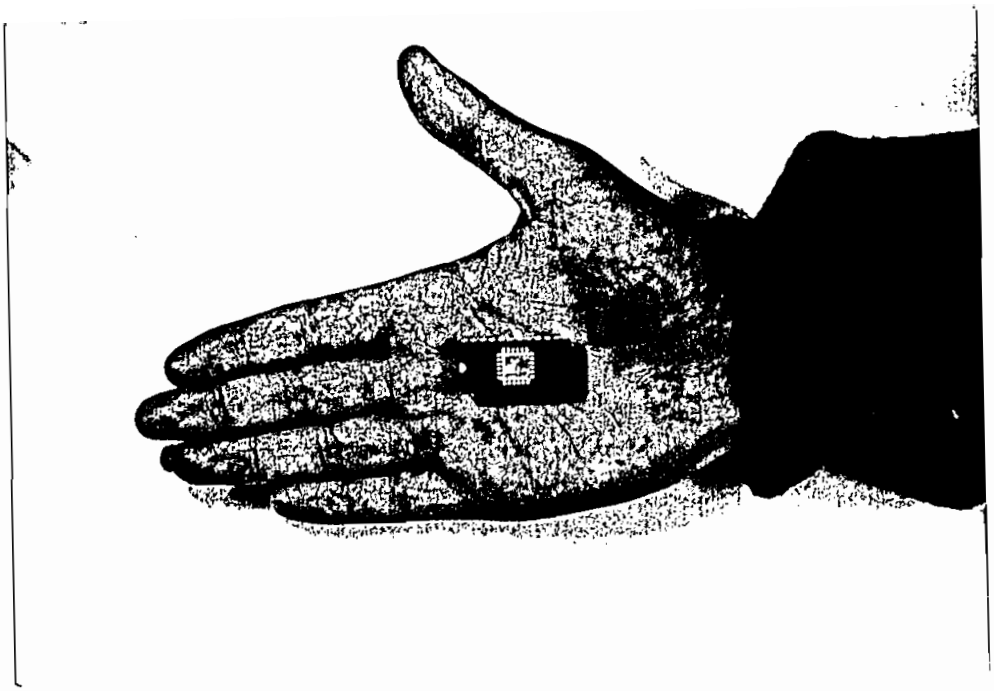


FIGURA 7.35 Prototipo del MPC que contiene al circuito MED_SUM.

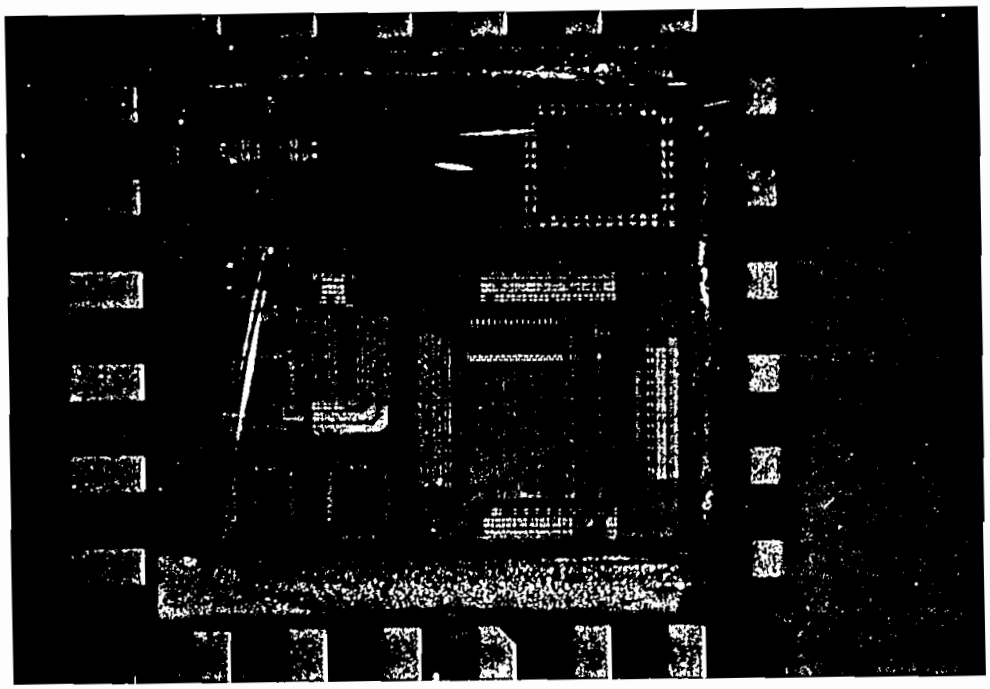


FIGURA 7.36 Oblea con los CIs agrupados en el Chip Multi-proyecto.

La Figura 7.37 muestra una aproximación al circuito MED_SUM integrado al Chip Multi-Proyecto y sus interconexiones con los pines del encapsulado.

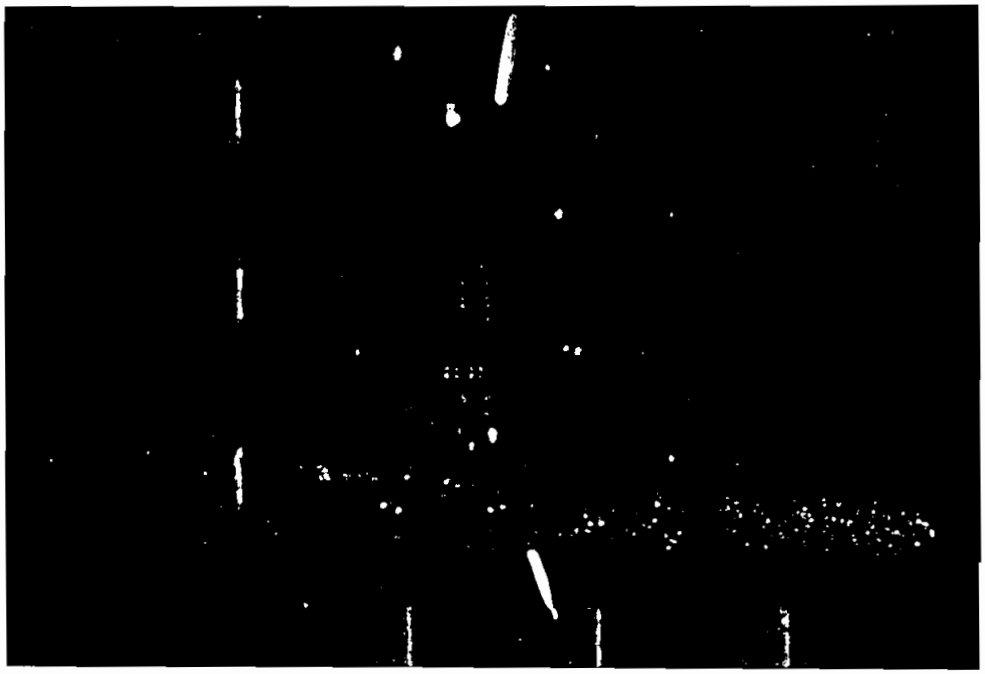


FIGURA 7.37 Detalle del CI MED_SUM.

La fotografía de la Fig.7.38 muestra en detalle el layout completo del circuito integrado MED_SUM trasladado a la oblea de silicio, esta fotografía permite apreciar la estructura del layout formado por el núcleo, PADs y los anillos de interconexión y polarización, compárese con los planos de las Figuras 7.31 y 7.33 en que se muestra el layout del CI propuesto para su fundición.

Finalmente, la Fig.7.39 muestra el núcleo del layout del CI MED_SUM en que se pueden observar en detalle las diferentes capas y construcciones que forman el CI, así como las líneas de interconexión en Metal1 y Metal2 desde sus terminales hacia los PADs, compárese esta fotografía con la Fig.7.19 que indica el layout del CI MED_SUM corregido y con la

Fig.7.32 en que se indica el plano del núcleo del layout del CI MED_SUM propuesto.

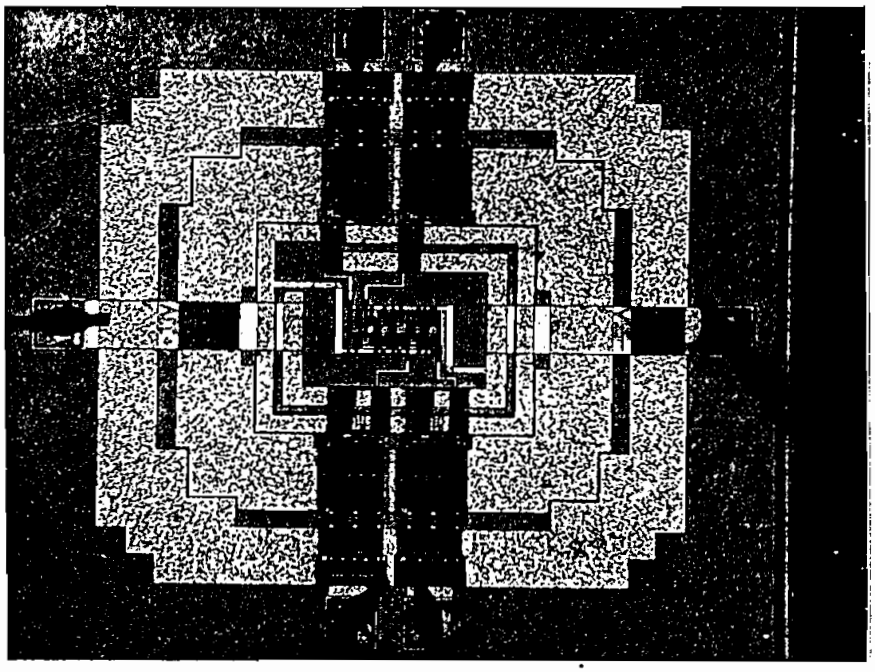


FIGURA 7.38 Layout del CI MED_SUM.

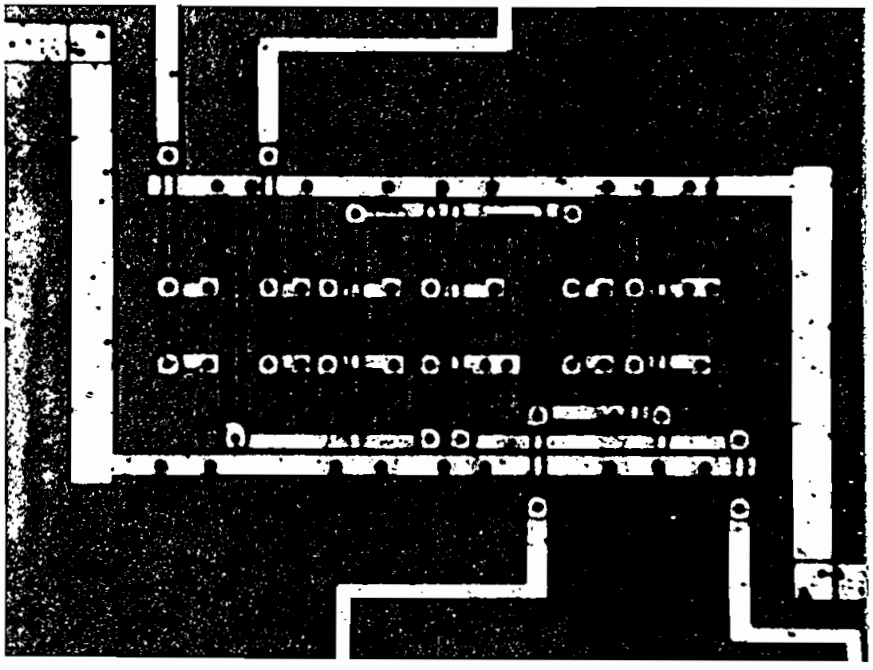


FIGURA 7.39 Detalle del núcleo del layout del CI MED_SUM.

7.4.3 Caracterización del circuito integrado

A fin de caracterizar el circuito Integrado "Medio Sumador", se transcribe a continuación informe (traducido) de su evaluación realizada en el Centro Tecnológico para Informática - CTI de Campinas - Brasil en donde tuvo lugar en Julio de 1992 la reunión de evaluación técnica del PMU.(11)

INFORME DE TEST

Circuito: MED_SUM

CAD: TENTOS

Proyectista: GRUPO DE MICROELECTRONICA/EPN

Responsable de las pruebas: LUIS MONTALVO

Descripción Funcional: MEDIO SUMADOR

Ai	Bi	Si	Ci
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TABLA 7.3 Descripción funcional del C.I. Medio Sumador.

Objetivo de las pruebas:

1) Verificación funcional

2) Medición de los tiempos de propagación

3) Análisis de PADs

---

(11) REUNIAO TÉCNICA DEL PROYECTO PMU e ILA9200
CYTED
PROGRAMA IBEROAMETICANO DE CIÉNCIA Y TECNOLOGIA PARA EL DESARROLLO
FUNDAÇAO CENTRO TECNOLÓGICO PARA INFORMÁTICA - CTI
CAMPINAS - SAO PAULO - BRASIL
27 a 31 de JULHO de 1992

Características del Circuito:

- Analógico/Digital: Digital
- Tecnología: ECPD_15
- Run: MPC91_1/MPC91_3/ILA92 (MPC91_3)
- PINES: Cantidad/Numeración/Identificación

NUMERACION	IDENTIFICACION
8	Ci (S)
9	Si (S)
15	GND
16	Ai
17	B1
22	Vcc
	TOTAL: 6

TABLA 7.4 Identificación de Pines del C.I. Medio Sumador.

- Entrada: 2
- Salida: 2
- Bidireccionales:
- Alimentación: 2

Planteamiento de Pruebas/Tiempo:

1) Verificación funcional/1 hora
2) Medición de tiempos de propagación/1 hora
3) Análisis de PADs / (30 min)

Equipamiento para las Pruebas:

1) 8182A DATA ANALYZER (8182A - HP)
2) 8082 PULSE GENERATOR (8082A - HP)

Resultados:

1) Verificación funcional:

Se obtiene la siguiente tabla de verdad:

Ai	Bi	Si	Ci
0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	0

TABLA 7.5 Verificación funcional del C.I. Medio Sumador.

Se observa que Si y Ci tienen una inversión.

Causa:

En el diseño se utilizaron los PADs de salida de ES2 OPSIU asumiendo que eren buffers no inversores, cuando en realidad son buffers inversores.

2) Medición de tiempos de propagación:

tp = 20 nseg

Observaciones y Mejoras:

El objetivo del Grupo de Microelectrónica de la EPN en esta corrida fue recorrer completamente el proceso de diseño, fabricación y pruebas de un CI y el miamo ha sido satisfecho completamente. El error lógico existente en el circuito por la inversión de los PADs no tiene trascendencia.

CAPITULO 8

DISEÑO DE ASICs BASADO EN CELDAS ESTANDAR - METODOLOGIA ESTRUCTURADA PPL (*Physical Placement of Logic*)

PPL (*Path Programmable Logic ó Physical Placement of Logic*) es un método de diseño de circuitos integrados estructurado, basado en celdas, simbólico e independiente de la tecnología, desarrollado por el grupo de VLSI de la Universidad de Utah (USA).

Cada una de las características mencionadas anteriormente y otras adicionales que presenta PPL, se justifican a lo largo de este capítulo.

8.1 DISEÑO CONVENCIONAL VERSUS DISEÑO PPL

Para tener una idea clara del tipo de metodología utilizada por PPL, es necesario establecer las posibles semejanzas y diferencias con otros métodos de diseño de CIs.

FPL es un método estructurado, basado en celdas, al igual que lo son: el arreglo de compuertas (*gate-array*) y el de celdas estándar (*standard-cells*). El enfoque diferente que adopta FPL para el desarrollo del proceso de diseño, permite que al referirnos a otros métodos estructurados basados en celdas, como los dos mencionados anteriormente, se lo haga denominándolos como *convencionales*.

La característica de estructurado de FPL proporciona una mejor solución a los complejos problemas que involucra el diseño de circuitos VLSI y permite la automatización del proceso de diseño de CIs asistido por computador.

De manera general, el diseño convencional de circuitos integrados incluye cinco fases:

a) especificación,
b) diseño funcional,
c) diseño lógico,
d) diseño circuital, y
e) diseño físico.

Cada una de las fases está caracterizada por síntesis, análisis y verificación. Dentro del proceso de diseño, se utilizan comunmente las descripciones: funcional, estructural y física, para indicar diferentes aspectos del sistema en desarrollo. Todos estos aspectos, mencionados de manera resumida, se tratan en detalle en el Cap. 1.

Los diseñadores han llegado a la conclusión que para facilitar el diseño de CIs, se requiere de una ayuda computacional que integre diferentes niveles de las fases de diseño, iniciando desde el nivel más bajo de distribución geométrica (*layout*), pasando por el diseño de nivel medio esquemático y lógico, hasta una manipulación simbólica de alto nivel.

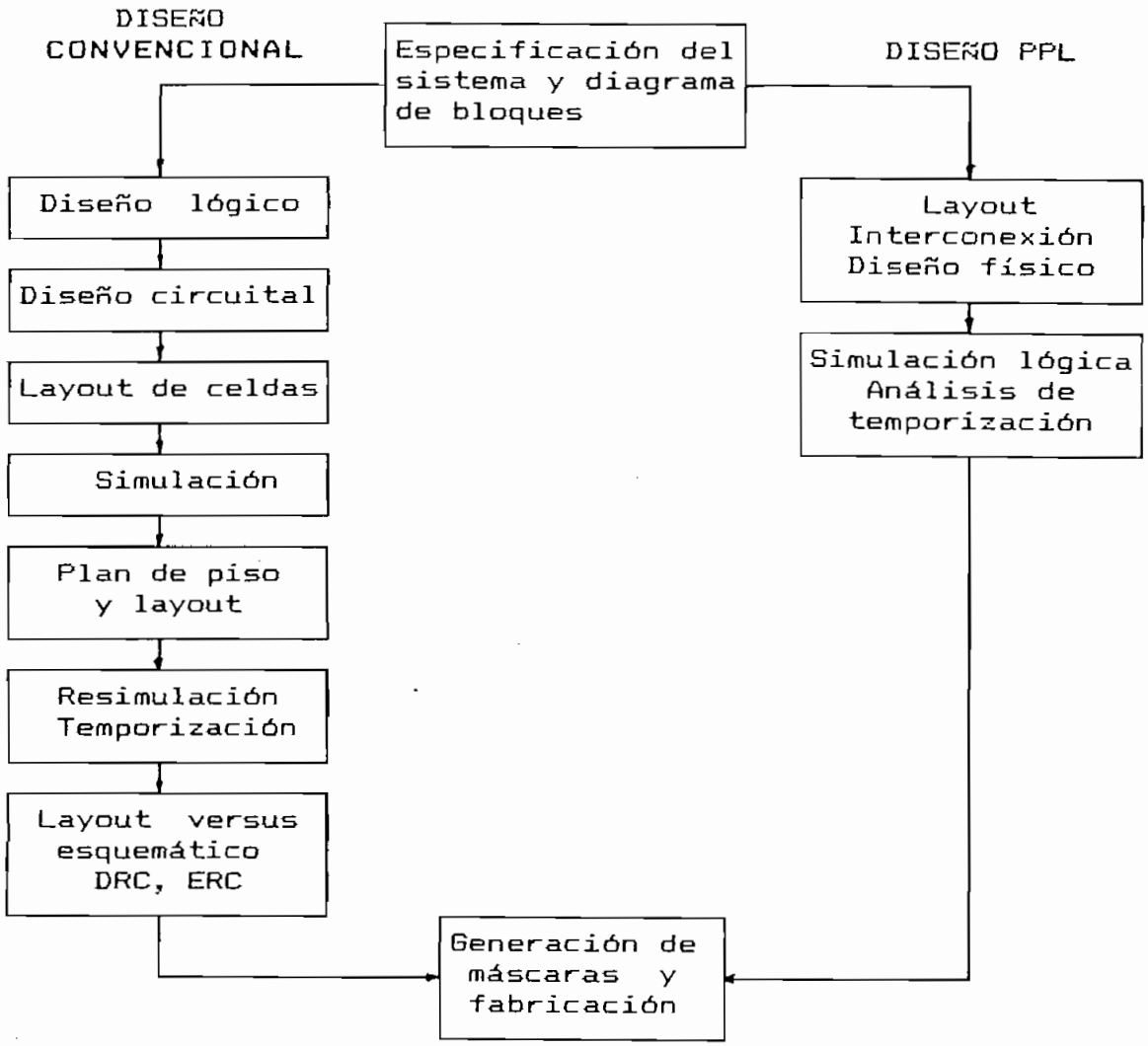


FIGURA 8.1 Métodos de diseño convencional versus el método de diseño PPL.

En la Fig. 8.1 se esquematiza el proceso de diseño seguido de manera general por los métodos convencionales, *custom y semi-custom*, y el proceso seguido por el método PPL. El diseño con PPL provee descripciones consistentes en las tres representaciones del diseño (funcional, estructural y física), en todos sus niveles de abstracción. A diferencia de los métodos convencionales, PPL reune muchas de las caracteristicas deseables en una herramienta computacional. PPL encapsula diferentes niveles de las fases de diseño, evitando de esta manera un gran número de pasos muchas veces repetitivos y opera completamente a un nivel simbólico, eliminando los detalles de los niveles bajos del diseño.

La Fig. 8.2 contrasta el diseño de un CI utilizando celdas estándar y el diseño con PPL. En el diseño con celdas estándar se observa la separación entre el diseño lógico y el diseño físico; los pasos de captura esquemática, simulación lógica y análisis aproximado de temporización son completamente independientes de los pasos de diseño físico, verificación del *layout* y análisis de temporización exacto.

Generalmente, el diseñador realiza varias iteraciones en su diseño a nivel lógico; unicamente cuando ha conseguido los resultados esperados a nivel lógico pasa al diseño físico. Si el diseño físico no cumple con las especificaciones establecidas, reitera en el nivel físico e incluso, si es necesario, lo hace en el nivel lógico.

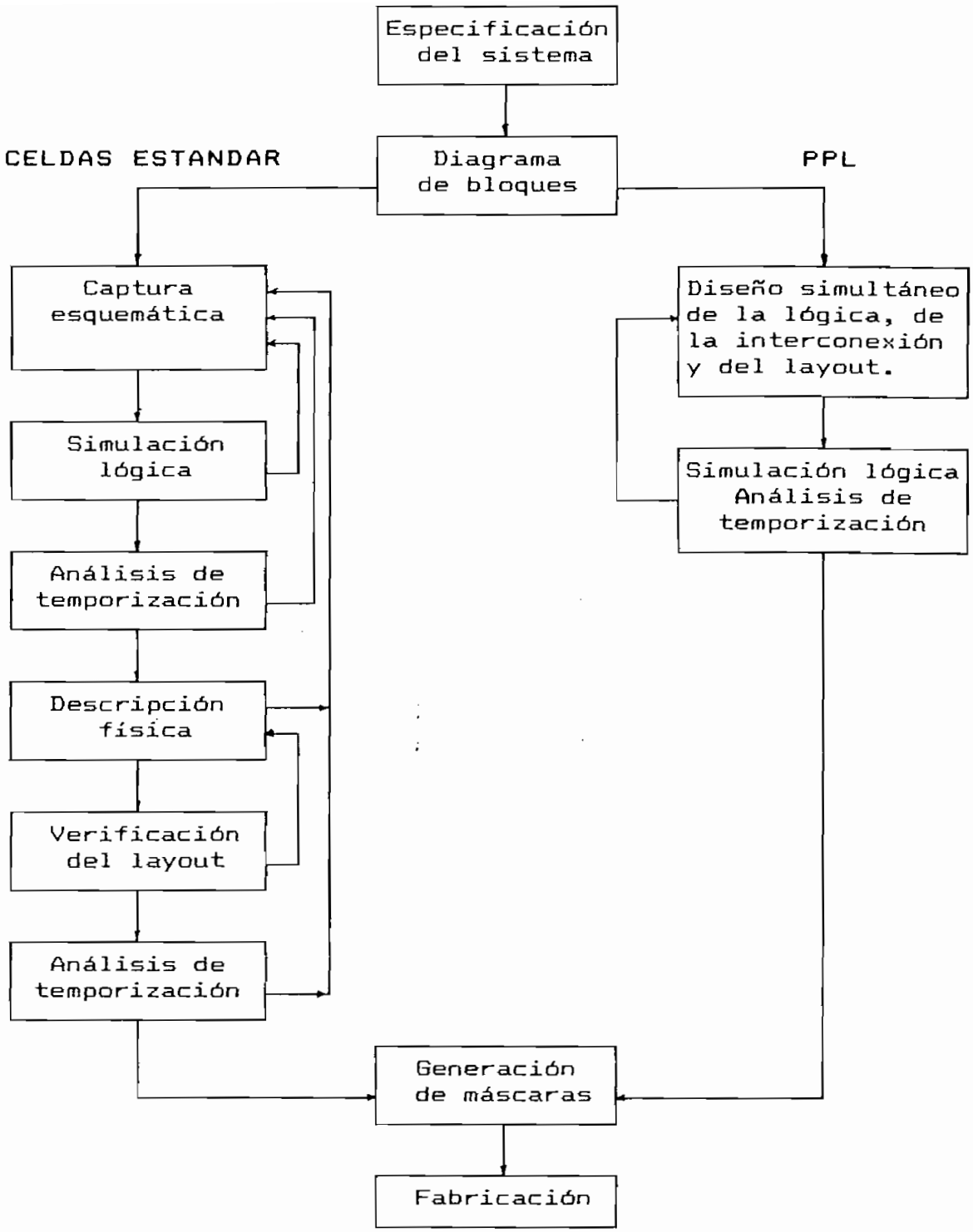


FIGURA 8.2 Método de diseño de celdas estándar versus el método de diseño PPL.

Causa frecuente, que obliga al diseñador a realizar varias iteraciones dentro del nivel físico e incluso a regresar al nivel lógico, es el tiempo de retardo de las señales demasiado grande, debido a la ubicación automática de los caminos de interconexión.

En la metodología PPL se combinan, en un solo paso, los niveles lógico y físico, reduciendo de esta manera el número total de pasos de diseño e incrementando la productividad del diseñador.

La metodología PPL permite que el diseñador sea quien determine, con planificación previa mínima, la ubicación relativa de las celdas para cumplir determinada función y gracias al nivel de abstracción que proporciona la representación simbólica de las celdas PPL y a la estructura de las mismas (las cuales permiten el enrutamiento de las señales y la interconexión con otras celdas unicamente con colocarlas unas junto a otras), se puede omitir la necesidad de que sea un programa el cual realice la ubicación y enrutamiento de las celdas de manera automática.

Las características mencionadas hacen que el diseñador controle la topología del *layout* al momento de realizar el diseño lógico, controlando también los retardos del circuito diseñado. Es precisamente el instante de la descripción lógica utilizando las celdas PPL cuando se hace presente la combinación de las dos fases de diseño lógico y físico, pues

el diseñador obtiene una realimentación automática del circuito diseñado, lo que le permite realizar un análisis de su comportamiento en su diseño inicial y no requiere recorrer gran parte del proceso para evaluar los resultados de su concepción del sistema.

De la característica de definición simultánea de los niveles lógico y físico del diseño, justamente se desprende el nombre del método PPL: *Path Programmable Logic ó Physical Placement of Logic*, que da la idea de una "ubicación física de la lógica".

Frente a métodos de diseño *full-custom*, que se caracterizan por una alta densidad de transistores en sus circuitos, tiempos de diseño grandes y la necesidad de diseñadores altamente especializados en cada una de las fases del proceso de diseño, PPL presenta las ventajas de lograr densidades en sus circuitos similares e incluso superiores a las alcanzadas por los métodos *full-custom*, tiempos de diseño significativamente más cortos y sin necesidad de diseñadores altamente especializados.

Comparado con técnicas de diseño *semi-custom*, tales como las celdas estándar y arreglos de compuertas (*standard cells, gate-arrays*), PPL reduce el tiempo de diseño en un factor de tres y al mismo tiempo mejora la densidad en factores de dos y tres, presentando la ventaja de no requerir diseñadores especializados en cada una de las fases del proceso.

Las Figs. 8.3 y 8.4 permiten apreciar graficamente lo anteriormente mencionado; se presenta una comparación en términos de la densidad de los circuitos diseñados y del tiempo de diseño, utilizando PPL y otras metodologías de diseño.

En la Fig. 8.3 la densidad se expresa en función del área requerida por transistor, mientras menor sea este valor se tiene un mayor número de transistores por unidad de área, PPL alcanza una densidad muy cercana a la ofrecida por los métodos *full-custom*.

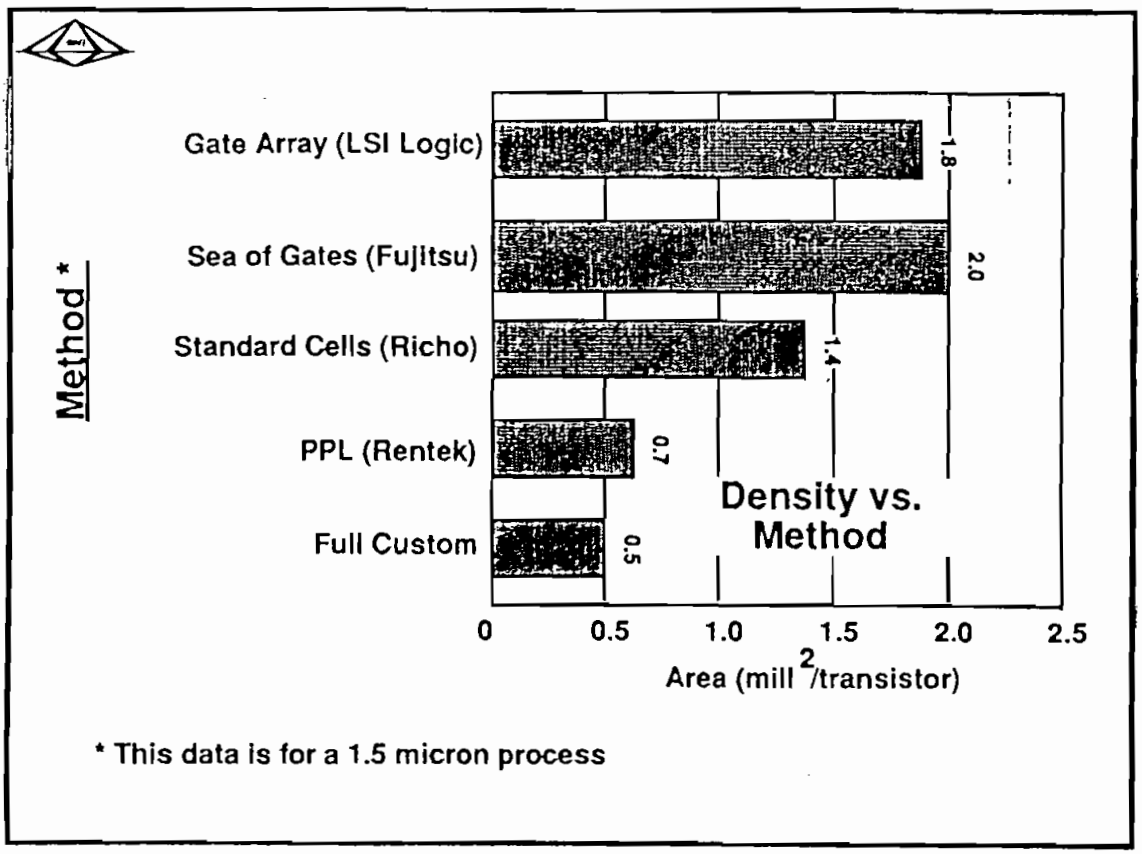


FIGURA 8.3 Comparación de la densidad de transistores alcanzada por PPL y otros métodos de diseño.

La Fig. 8.4 presenta, de manera muy clara, el menor tiempo requerido por PPL frente a otros métodos de diseño; para este caso el eje horizontal se expresa en función de días requeridos por cada 100 compuertas incorporadas definitivamente al diseño.

Los gráficos de comparación que se presentan fueron realizados por la compañía *Rentek Inc.*, en base a información recopilada de manera independiente.

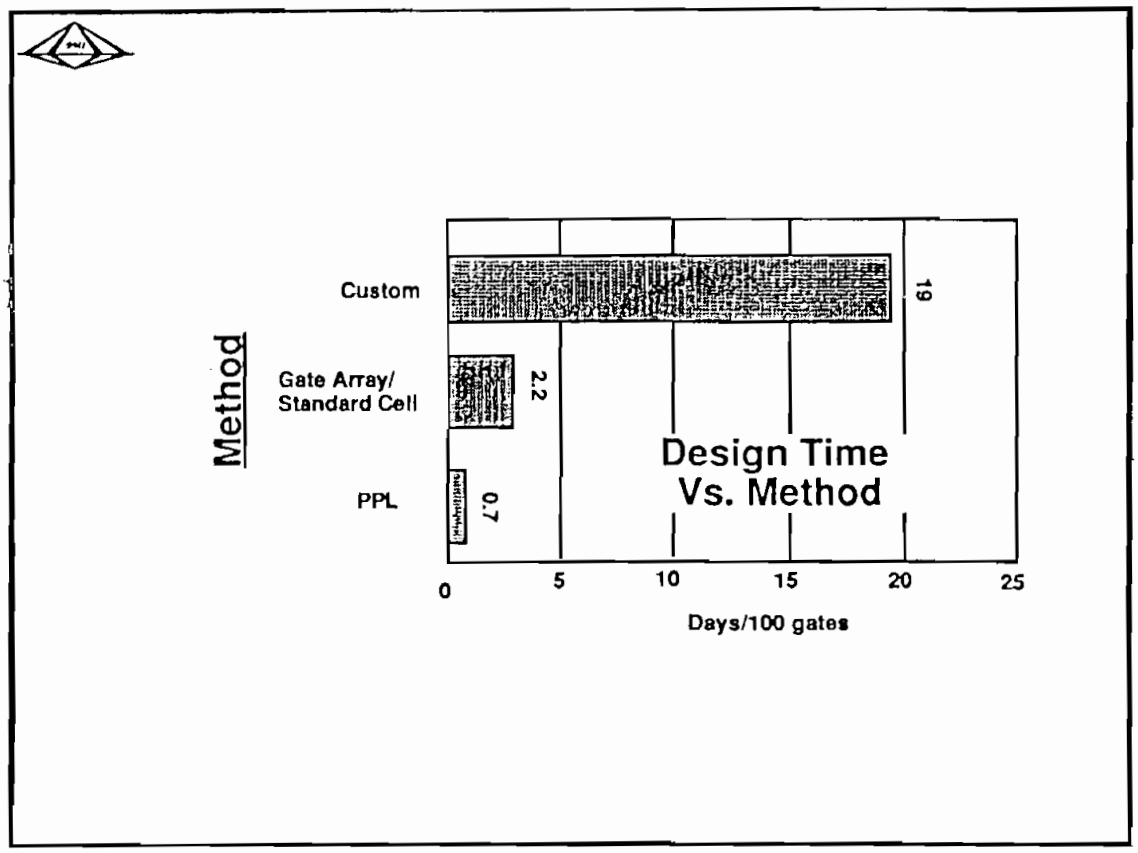


FIGURA 8.4 Comparación del tiempo de diseño con diferentes métodos de diseño.

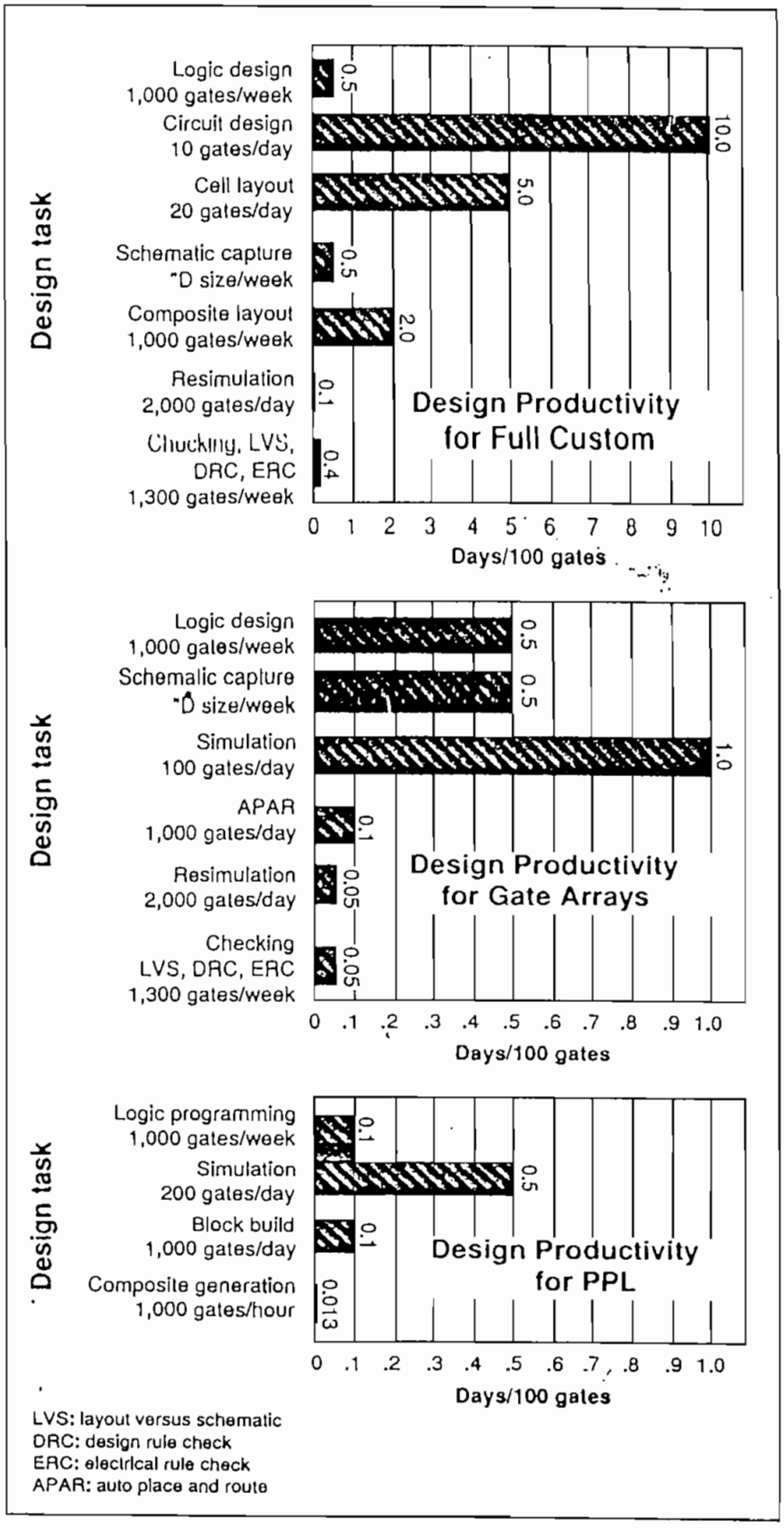


FIGURA 8.5 Comparación detallada del tiempo de diseño utilizando diferentes métodos.

La Fig. 8.5 presenta una evaluación del tiempo requerido para la realización de ciertas tareas de diseño utilizando métodos *full-custom, gate-arrays y PPL*. Nuevamente se pone de manifiesto la integración de algunas fases de diseño, la reducción de tiempo y el incremento de la productividad conseguidos por la metodología PPL. Para los métodos de diseño *convencionales*, el número de tareas a evaluar es mayor.

8.2 METODOLOGIA DE DISEÑO PPL

Para introducir las ideas principales de la metodología PPL, basada en celdas, conviene revisar brevemente los principios utilizados por los métodos de diseño de celdas estándar mas comunes, para luego realizar una comparación con los utilizados por PPL.

Entre las características de las familias de celdas estándar más comunes se tiene:

a) Las celdas tienen una altura fija y ancho variable, el cual depende de la complejidad de la celda.

b) Las celdas se agrupan formando filas y luego estas filas se colocan en columna, una sobre otra.

c) Para el enrutamiento de las señales entre celdas, generalmente se tienen filas de enrutamiento entre ellas ya sea bajo o sobre las filas que contienen las celdas y cuando es posible corriendo directamente caminos entre las celdas.

d) Lo mencionado anteriormente lleva a tener V_{DD} y Gnd entre las celdas.

e) Muchas veces, las celdas y los caminos de interconexión deben colocarse en localidades de columnas específicas, de esta manera todas las celdas ocupan un número discreto de columnas en la dirección horizontal pero tienen una altura fija.

f) Esto da como resultado que se tenga una estructura unidimensional ya que la única variable es el número de columnas que una celda puede ocupar.

PPL, por el contrario, presenta las siguientes características:

a) Las celdas no tienen ni altura ni ancho fijos. Las celdas PPL pueden ocupar un número múltiple de localidades de filas y columnas. Por ejemplo algunas celdas, llamadas celdas "unitarias", ocupan solamente una localidad de una fila y una columna, mientras otras celdas, llamadas "múltiples", pueden ocupar varias localidades.

Una celda flip-flop podría ocupar dos columnas y tres filas, mientras un inversor podría ocupar solamente dos filas y una única columna.

b) Como consecuencia de lo anterior, las celdas no se agrupan formando filas de una altura fija.

c) Las celdas están diseñadas para permitir su interconexión, por las cuatro direcciones, simplemente colocándolas unas junto a otras, gracias a los caminos de metal que corren hacia los extremos en cada una de las celdas. Los caminos de la celda se utilizan también para el enrutamiento de las señales, pueden permanecer intactos o removerlos de acuerdo a las necesidades del diseñador.

d) El método de interconexión desarrollado a través de las celdas PPL, permite un ahorro sustancial del área necesaria para la interconexión.

e) V_{DD} y Gnd corren verticalmente entre las columnas y forman parte del *layout* de cada una de las celdas. En el proceso de diseño su presencia y enrutamiento son transparentes al diseñador.

f) Todas las características mencionadas, en conjunto, llevan a que la metodología PPL de como resultado final una estructura de dos dimensiones.

En el método PPL, un circuito se diseña ubicando símbolos, que representan a las celdas, en una matriz de celdas formada por filas y columnas, desde este punto de vista la ubicación puede ser arbitraria. Más adelante se demuestra que para cumplir una función determinada las celdas deben colocarse junto a otras que sean compatibles, en otras palabras las celdas no trabajan solas, sino que son útiles cuando se las coloca con otras para formar estructuras más complejas, para esto se debe realizar una planificación previa de la ubicación relativa de las celdas, esto se debe principalmente al *layout* propio de cada una de ellas.

8.2.1 La grilla de caminos PPL

El *layout* de las celdas PPL se visualiza mejor asumiendo que inicialmente el *chip* está formado totalmente por caminos de interconexión de metal (metal 1 y metal 2) formando lo que se podría llamar un "mar de caminos" en el que se colocan los transistores o compuertas, en lugar de imaginarlo como un "mar de transistores o compuertas" que luego son interconectados por caminos. Los caminos de la grilla PPL están regularmente espaciados y corren horizontal y verticalmente. En la Fig. 8.6 se esquematiza el mar de caminos, puede observarse también que V_{DD} y Gnd, son parte de los caminos verticales. Los caminos verticales no se conectan a ninguno de los caminos que corren horizontalmente, cuando se desee que esto ocurra se debe utilizar celdas diseñadas para este objetivo.

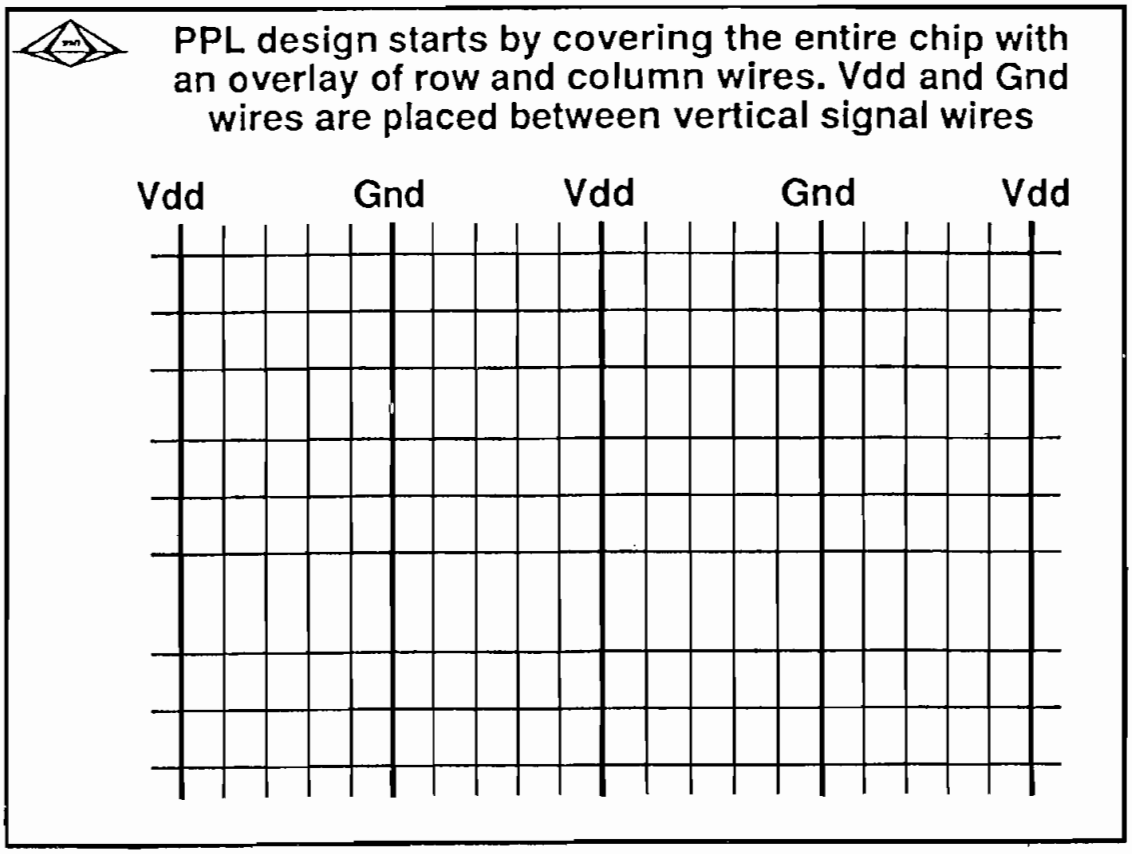


FIGURA 8.6 Caminos horizontales y verticales de la grilla de metal PPL, incluidos V_{DD} y Gnd.

Los caminos horizontales y verticales de PPL se dividen en conjuntos de caminos, de tal manera que cada fila y columna pueden representarse como constituídos por N caminos horizontales y M caminos verticales. El número máximo de N y M fue determinado, por los diseñadores de PPL, en base a información empírica. Se determinó que para cualquier columna M sea máximo cuatro y que para las filas N sea máximo tres. En general, N y M son al menos 1 y 2, respectivamente. Por ejemplo, para la tecnología nMOS, hay dos caminos verticales y un horizontal por cada localidad fila/columnna de la grilla.

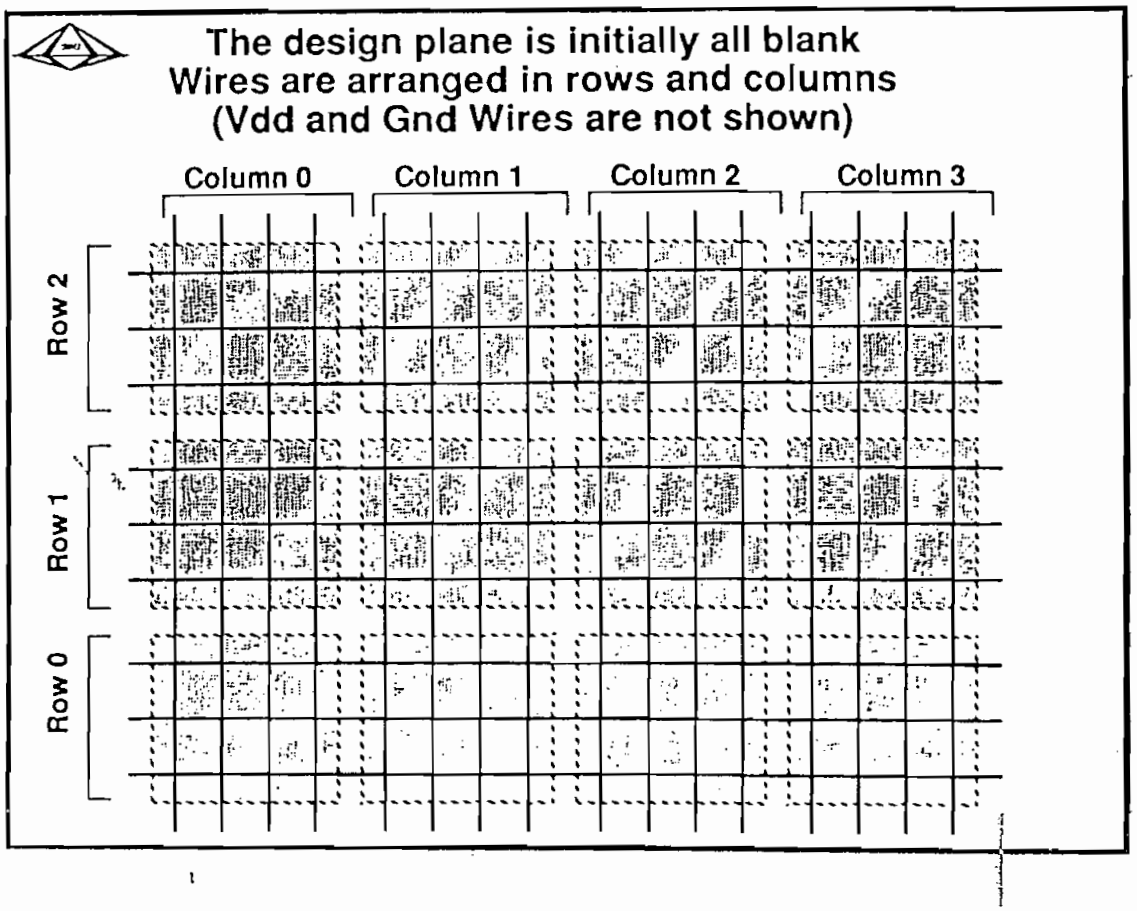


FIGURA 8.7 Particionamiento de los caminos de la grilla de metal PPL en filas y columnas.

Para el caso CMOS, tecnología con la que se trabajará durante la realización del presente trabajo, cada celda unitaria tiene cuatro caminos verticales y tres caminos horizontales en cada localidad fila/columna (Fig. 8.7). Los caminos de polarización (V_{DD} y Gnd) corren, entre pares de columnas, por un nivel bajo de interconexión, hacia todas las localidades de la grilla (Fig. 8.6), pero son invisibles al diseñador, no se incluyen en los M por N caminos y son compartidos por el par de columnas.

En base a lo descrito hasta el momento, el ubicar una celda en la grilla PPL es equivalente a retirar una pequeña sección de caminos y reemplazarla por un circuito predefinido, como se indica en la Fig. 8.8. Para el ejemplo presentado, la celda introducida está formada por un par de transistores y puede apreciarse que algunos caminos de la grilla original no están presentes, la presencia y/o ausencia de los caminos depende de la distribución geométrica de cada celda.

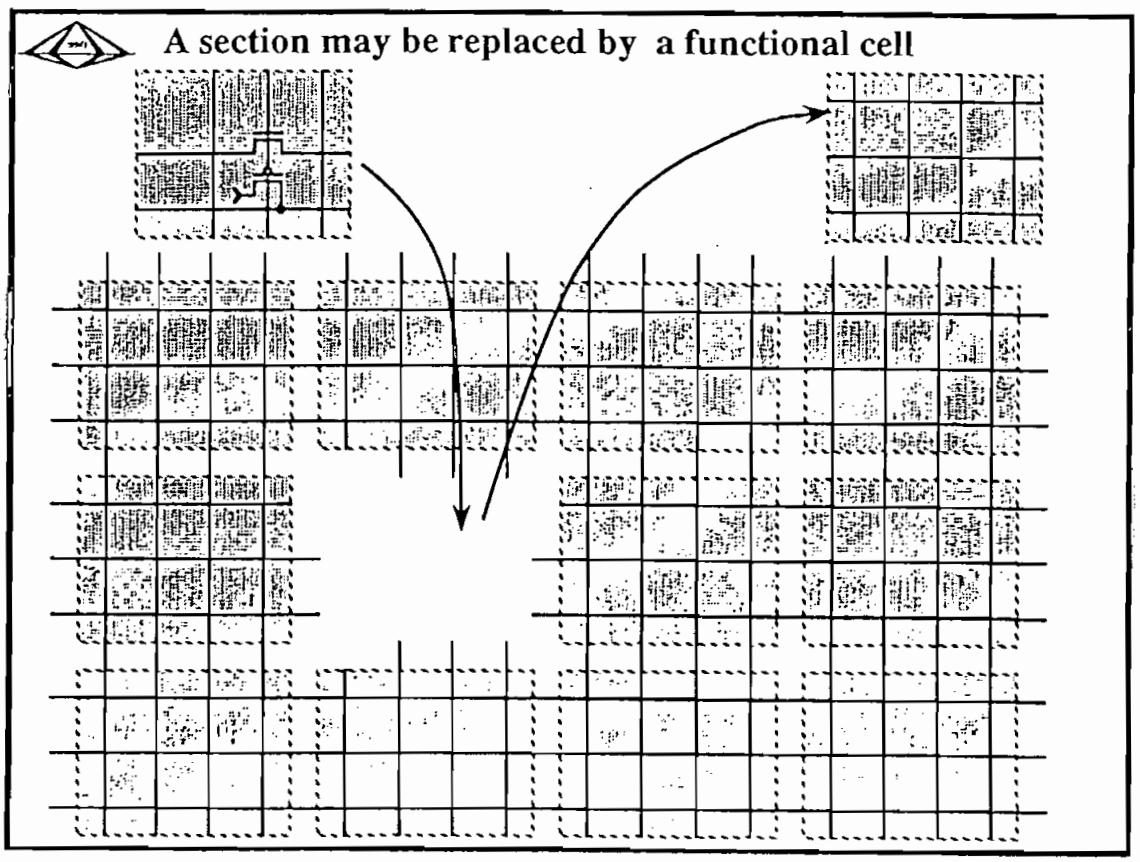


FIGURA 8.8 Colocación de una celda funcional en la grilla PPL.

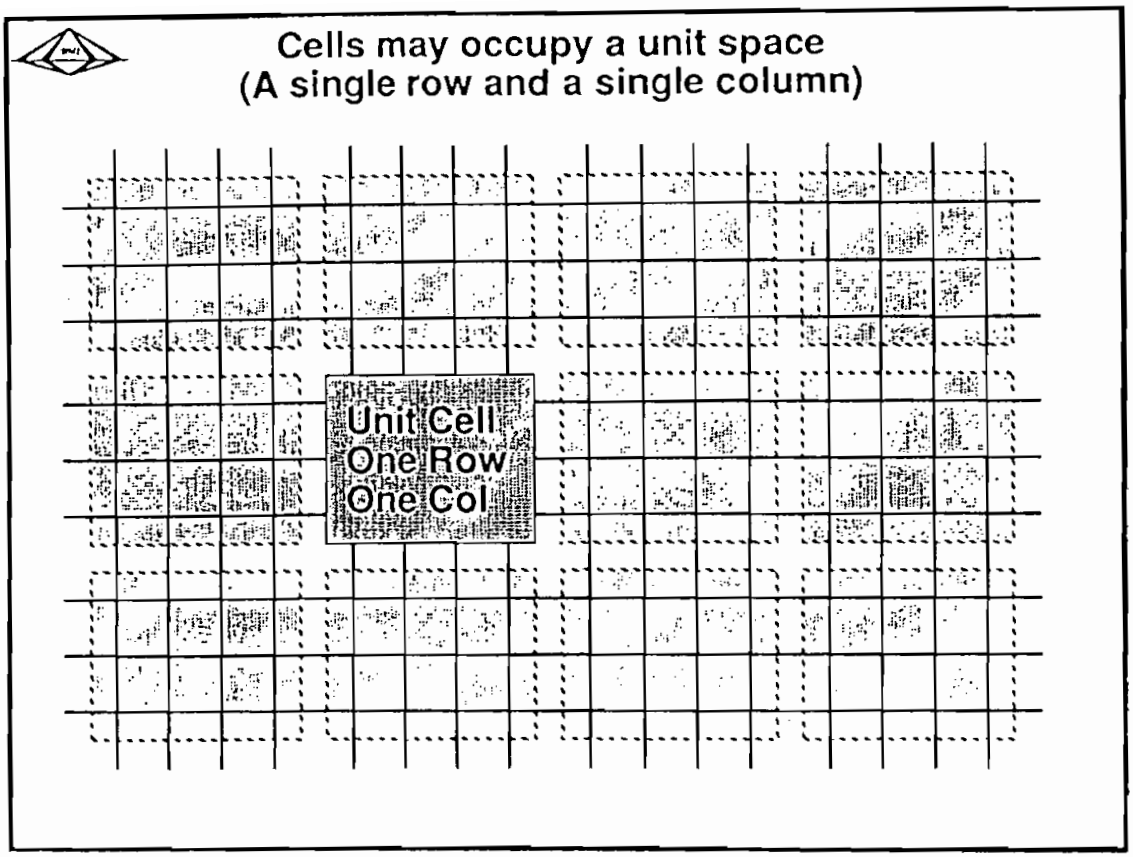


FIGURA 8.9 Celda unitaria: una fila por una columna.

Las Figs. 8.9 y 8.10 ilustran la colocación de una celda unitaria y de una celda múltiple en el arreglo PPL, respectivamente. La celda unitaria cubre una localidad de la grilla constituída por una única fila y columna. La celda múltiple puede cubrir varias localidades de la grilla, abarcando varias filas y columnas completas. La representación para la celda múltiple en el caso de la Fig. 8.10 cubre 2 filas y 2 columnas. En las Figs. 8.9 y 8.10, los caminos de interconexión de los cuatro lados de las celdas se dejan aún intactos, permitiendo su conexión con las celdas circundantes.

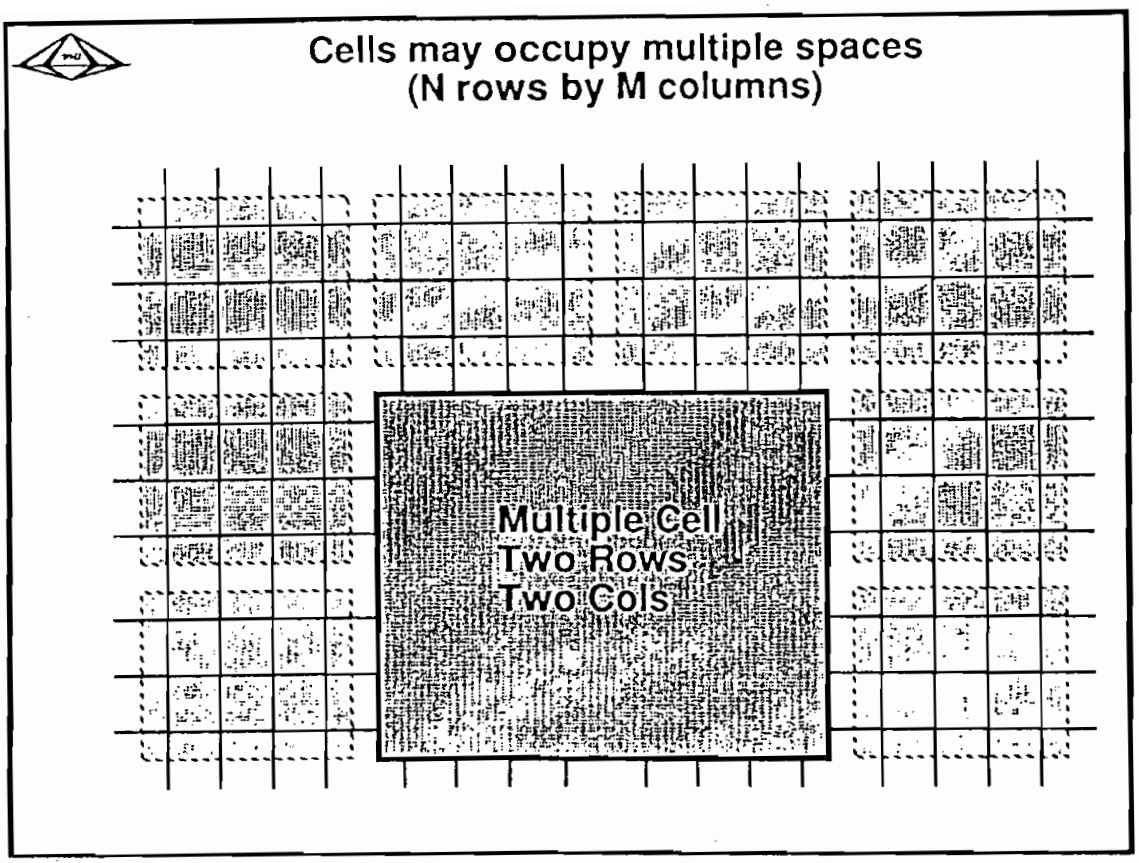


FIGURA 8.10 Celda múltiple: 2 filas por 2 columnas.

Las celdas de acuerdo a su *layout* propio, permiten que los caminos de las filas y columnas crucen por ellas o pueden bloquearlos. De los caminos que atraviezan las celdas, el diseñador decide si dejarlos intactos o interrumpirlos. La interrupción se realiza en los limites de las celdas, forzando "interrupciones" (*breaks*) como se ilustra en la Fig. 8.11. Las interrupciones se tratan en detalle más adelante.

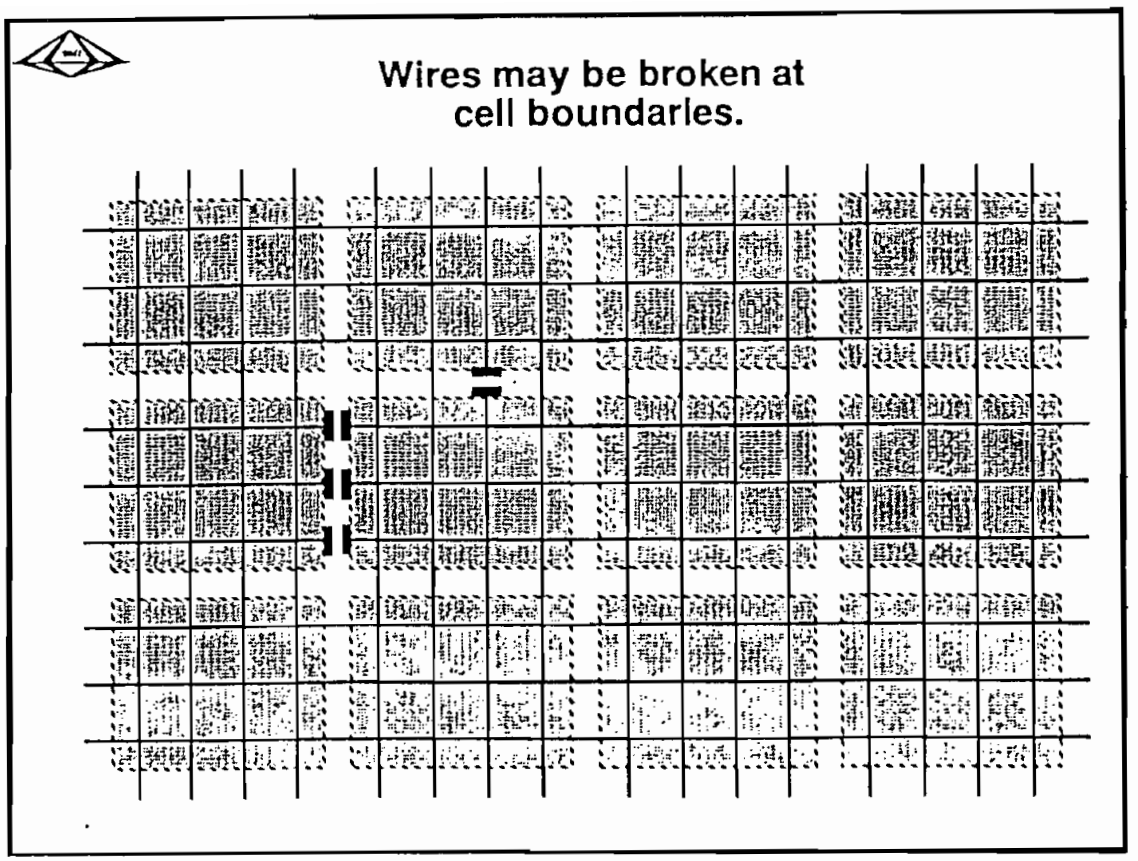


FIGURA 8.11 Interrupción de los caminos en los límites de las celdas.

8.2.2 Las celdas PPL

Las celdas PPL son la clave para el funcionamiento de los circuitos VLSI diseñados utilizando la metodología PPL. Al nivel más bajo del *layout*, dichas celdas deben cumplir las especificaciones eléctricas y geométricas, además de las conexiones topológicas múltiples. Al nivel del diseño lógico simbólico más alto, la representación simbólica de las celdas debe ofrecer un método de diseño general, jerárquico, independiente de la tecnología y basado en reglas deducidas de la experiencia proporcionada por la manipulación de las celdas.

a) Representaciones gráficas de las celdas PPL para CMOS.

Se han desarrollado conjuntos de celdas PPL para algunas tecnologías, dos para tecnología nMOS, cuatro para tecnología CMOS y dos para tecnologías de Arsenuro de Galio (GaAs).

Se describen a continuación las representaciones gráficas del conjunto de celdas para la tecnología CMOS, de la cual se dispone actualmente en la EPN.

Los nombres y estructuras de las celdas varían de un conjunto de celdas a otro. Las celdas desarrolladas para CMOS por ejemplo, pueden ser totalmente diferentes de las celdas desarrolladas para nMOS. Sin embargo, el traducir un diseño de un tipo de tecnología a otro no es un proceso complejo.

Se disponen de múltiples maneras gráficas de representar las celdas PPL. En la Fig. 8.12 se muestran la representación simbólica, lógica, de transistores y a nivel de *layout* de una de las celdas unitarias elementales.

El diseño de un CI utilizando PPL se realiza utilizando la representación simbólica; se recurre a la representación a nivel de transistores cuando se requiere tener una visión ampliada de las celdas. La representación lógica se utiliza para evaluar o trabajar sobre un diseño ya realizado. Por lo general, el diseñador no requiere descender al nivel del *layout* de las celdas.

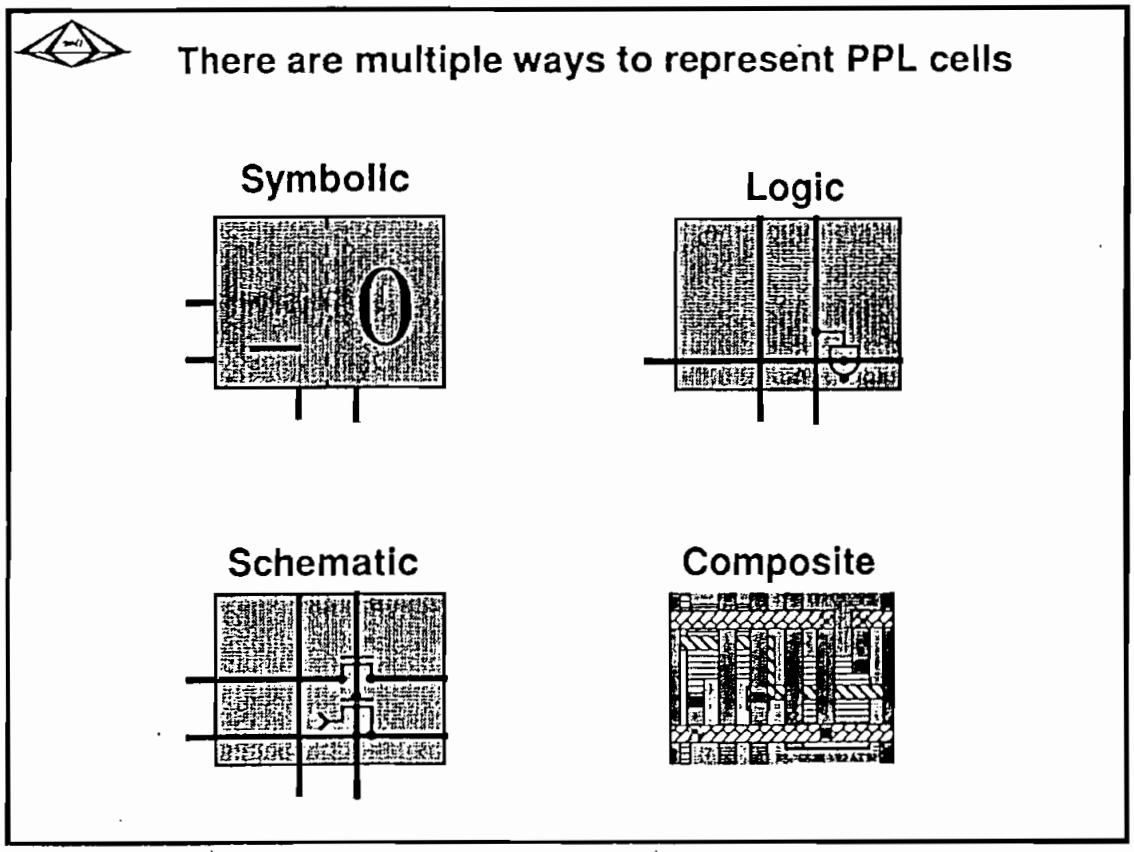


FIGURA 8.12 Múltiples representaciones gráficas de las celdas PPL.

Puede notarse la Fig. 8.12 la diferencia en el grado de abstracción que nos ofrece cada representación. El nivel más bajo lo tiene la representación de la distribución geométrica compuesta, que incluye las diferentes capas de metal y semiconductores que forman la celda (*composite*), un grado más arriba está la de transistores, seguidamente, la representación lógica (con la que el diseñador de circuitos digitales está más familiarizado); por último, la representación simbólica, que ofrece el nivel más alto de abstracción y que es la base de la metodología PPL, pues permite el diseño de CIs desde un punto de vista "simbólico" antes que "lógico".

i) Representación simbólica

La representación simbólica permite realizar una planificación de la ubicación relativa de las celdas para cumplir una función deseada, misma que puede ser un módulo de un sistema más complejo. Esta planificación puede hacerse, aún sin disponer de un computador, unicamente entendiendo el fundamento de la metodología PPL y la información disponible en los símbolos de las celdas y haciendo uso, de disponerse, de la experiencia previa en el diseño de otros circuitos con PPL y de posibles reglas deducidas de dicha experiencia.

Para procesar la información de la planificación previa mencionada anteriormente, es indispensable la utilización de un computador. Entre los programas con los que se desarrolla el diseño PPL se encuentra el editor TILER. Este editor está orientado al diseño de CIs, está basado estrictamente en la representación de cada una de las celdas por medio de caracteres (no es un editor gráfico). Los caracteres corresponden a los nombres de las celdas, los cuales son ingresados por el diseñador y TILER es el encargado de reservar las localidades de la grilla PPL necesarias, de acuerdo al caracter de la celda ingresada. En la versión de PPL disponible actualmente, con TILER se representa una grilla de máximo 79 filas X 36 columnas. Algunas de las características de la representación simbólica que se mencionan en la presente explicación , no son presentadas en la pantalla del editor TILER, otras son presentadas indirectamente en una línea de *status* del editor

y otras en modos de edición especiales de TILER, que se describen al estudiar las diferentes herramientas computacionales que incluye el paquete de diseño PPL.

ii) Representación lógica

La representación lógica se la puede obtener con la ayuda del paquete CAPFAST que trabaja junto con el paquete PPL.

Una vez descrito el circuito con la ayuda de TILER, se procesa esta información para posteriormente realizar la captura esquemática y obtener la representación del circuito a nivel lógico. Debe señalarse que la captura esquemática no es una acción indispensable dentro del proceso de diseño PPL, pero si es recomendable. Se realiza un etudio más detallado de CAPFAST en el numeral de descripción de las herramientas del paquete PPL.

iii) Representación a nivel de transistores

La representación a nivel de transistores nos permite tener mayor detalle de las celdas que el ofrecido por la representación simbólica y lógica. Solamente está disponible en la descripción detallada de cada una de las celdas que se incluye en el manual de PPL, y no se la puede visualizar en el computador. Sin embargo, debe quedar claro que la representación simbólica ofrece la información necesaria para

desarrollar el diseño, las representaciones restantes son útiles durante el periodo de aprendizaje de las herramientas PPL, luego de lo cual la manipulación de la representación simbólica es suficiente.

iv) Representación geométrica

La representación geométrica, puede ser visualizada utilizando el programa LEDIT. LEDIT es un editor de máscaras y acepta como entrada archivos en formato CIF, para su posterior visualizacion e impresión. Actualmente no se dispone de LEDIT por lo que no se incluyen resultados obtenidos con este programa, ni se describe su manipulación.

b) Descripción de la representación simbólica de las celdas

La representación simbólica de una celda PPL se ilustra en la Fig 8.13. Las líneas grises y anchas, horizontales y verticales, que cruzan al símbolo, denotan el límite entre localidades unitarias adyacentes de la grilla, 4 localidades para la celda de la Fig. 8.13. Estas líneas de límite de filas y columnas son de información y solamente se utilizan en la descripción simbólica, no se las visualiza en la pantalla del computador al utilizar TILER.

Las líneas verticales entrecortadas, que tampoco se presentan en TILER, indican la división de la localidad de una celda unitaria en dos regiones, la ubicada a la derecha

de dicha línea se utiliza para representar, mediante un caracter, la función de la celda, y la otra región, ubicada a la izquierda, se utiliza para colocar un caracter que indica el uso de una interrupción (*break*) impuesta por el diseñador.

Los caracteres utilizados para representar las celdas PPL se escojen, por lo general, para reflejar la función lógica de las celdas. El caracter "F", por ejemplo, se utiliza para representar un *flip-flop*, los caracteres "r" y "s" representan a las celdas que ejecutan las funciones de *reset* y *set* de los *flip-flops*.

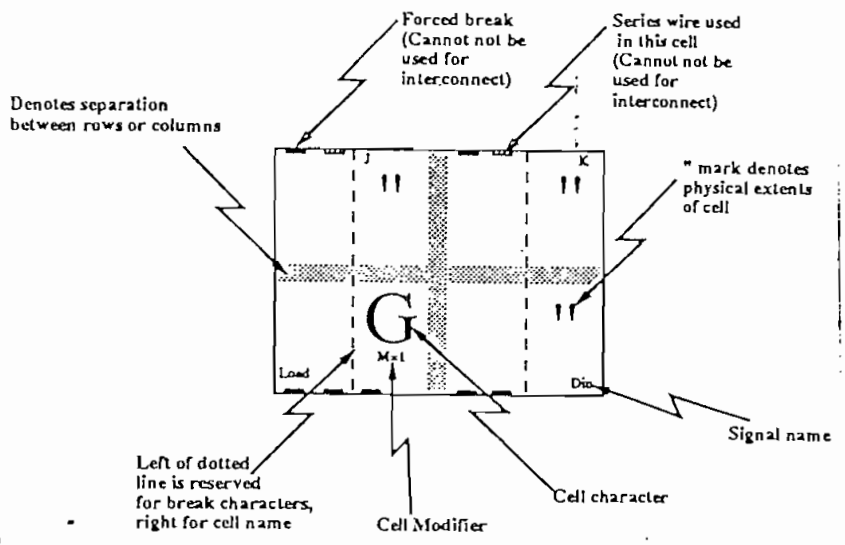


FIGURA 8.13 Información disponible en la representación simbólica de una celda PPL.

Puede haber ocasiones frecuentes, especialmente en celdas múltiples, en las que no se utiliza una o incluso ninguna de las dos regiones en que se divide una localidad de

celda unitaria. Por ejemplo en la Fig. 8.13 la letra G, en la localidad de la esquina inferior izquierda, representa el nombre de la celda, pero en las tres localidades unitarias adicionales que ocupa la celda, se aprecia que en la región correspondiente al nombre de la celda se ha colocado un par de comillas y se ha dejado vacía la región correspondiente al caracter de la interrupción. Las comillas se utilizan para representar la extensión de las celdas múltiples.

Ejemplos de celdas en las que se utilizan interrupciones se presentan más adelante, luego de realizar una explicación más detallada de las mismas y de ciertas restricciones en su uso.

En la parte inferior inmediata al nombre de la celda, se pueden apreciar los caracteres M = 1, esto indica que la celda utilizada ha sido alterada con un valor numérico denominado modificador, en este caso igual a uno. Este concepto de los modificadores permite tener un caracter para una celda base y unicamente añadiendo un modificador a dicha representación, acceder a una celda cuya descripción en la base de datos de PPL es totalmente independiente. Las celdas se agrupan bajo un mismo nombre considerando sus similitudes, en muchas de las celdas, un modificador introduce una alteración más de forma que de función. La información sobre los modificadores se presenta en la línea de status de TILER. En el siguiente numeral se analizan con más detalle los modificadores.

Es importante mencionar que a cada uno de los cuatro caminos verticales y tres caminos horizontales que constituyen una localidad unitaria de la grilla PPL, se les ha asignado un nombre determinado asociado a su función y/o posición dentro de la grilla. Los nombres de los caminos se presentan en la Fig. 8.14. Para las filas se tiene: TROW (de *TOP ROW*), el camino superior de las filas, el SROW (de *SERIES ROW*) y ROW. Para las columnas se tiene: RCOL (de *RIGHT COLUMN*) o columna de la derecha, LCOL (de *LEFT COLUMN*) o columna de la izquierda, SCOL (de *SERIES COLUMN*) y el TCOL (de *TOP COLUMN*).

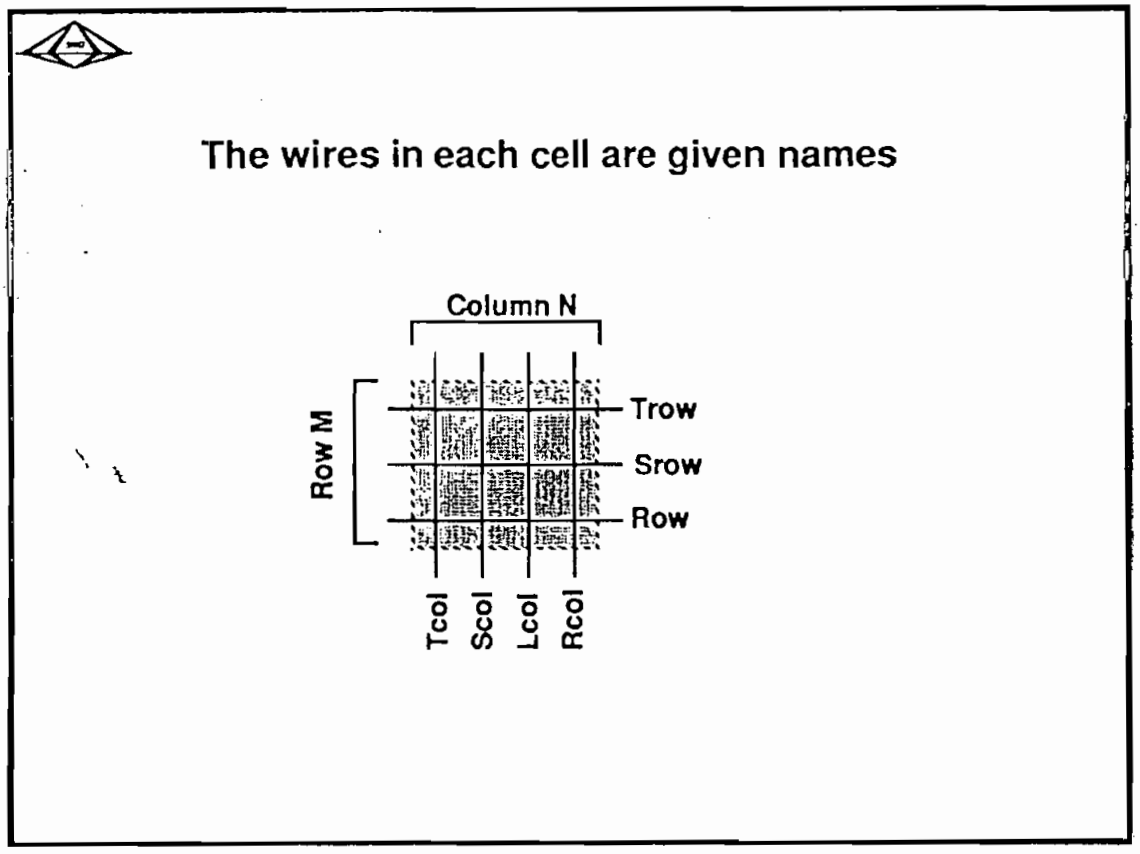


FIGURA 8.14 Nombres de los caminos horizontales y verticales de la grilla PPL.

Para una celda unitaria, se puede decir que en su límite horizontal, superior e inferior, la celda tiene cuatro puertos (RCOL, LCOL, SCOL, TCOL), y en su límite lateral, derecho e izquierdo, tiene tres puertos (TROW, SROW, ROW). De acuerdo a la estructura propia de cada celda, los puertos pueden ser de entrada o salida, no siempre están disponibles y de los utilizados, algunos son nombrados de acuerdo generalmente a su función dentro de la celda.

Regresando a la Fig. 8.13 puede notarse que para el caso de celdas múltiples, las señales de los puertos de entrada y salida son nombradas unicamente en el límite más externo de las mismas, para la celda G se han nombrado algunos de los puertos como: *K, J, Din, Load*. Estos nombres *no* pueden ser visualizados cuando se trabaja con TILER.

También, en los límites exteriores de la celda de la Fig. 8.13, pueden apreciarse pequeños rectángulos, unos negros y otros rayados, que están centrados respecto a alguno de los caminos que corresponden a los puertos. Estos rectángulos se utilizan para indicar interrupciones, (*breaks*), propios de la celda (*series port y forced breaks*) debido a su *layout* particular. Las interrupciones a las que se hizo mención al inicio de esta descripción, aquellas que se representan en la parte izquierda de la línea entrecortada divisoria, son para las interrupciones que el diseñador desea establecer y se detallan más adelante.

Los rectángulos negros indican que un determinado puerto no está disponible, en otras palabras que ningún camino puede conectarse a ese punto, a estas marcas se las conoce como interrupciones obligadas o forzadas (*forced breaks*). Los rectángulos rayados indican que esos puertos son parte de un grupo de transistores conectados en serie (*stack*) para implantar una compuerta y no pueden ser utilizados para otro fin que no sea el mencionado y en caso contrario deberían ser interrumpidos. A este tipo de rectángulo se le denomina puerto serie (*series port*). El concepto de lo que es un puerto serie se lo entiende mejor al utilizar las celdas PPL, especialmente las celdas elementales, y al analizar las representaciones gráficas de las diferentes celdas.

La información de las interrupciones de cualquiera de los caminos, sean éstas propios de la celda o impuestas por el diseñador, se representa en la línea de *status* de TILER, y se visualizan en la pantalla en un modo especial de operación del editor (MODE ZOOM IN) que se explica oportunamente.

Para la celda G de la Fig. 8.13, puede apreciarse que:

i) Para el caso de la columna de la izquierda:

   i.1 En la parte inferior unicamente RCOL no tiene un *forced break*, si una señal ingresa desde la parte superior (o inferior) en RCOL pasa directamente a través de la celda.

i.2 Ninguna señal puede ingresar por TCOL en la parte superior (o inferior), y obviamente no se la puede obtener en TCOL en la parte inferior (o superior), esto se representa con los rectángulos negros.

i.3 SCOL de la parte superior debe conectarse a un *stack* de transistores, externo a esta celda, en el cual uno de los extremos del *stack* esta dentro de esta celda, y es justamente la razón por la cual en la parte inferior no se presenta otro rectángulo rayado, sino uno negro, que nos indica que este puerto no está disponible. El otro terminal del stack se consigue colocando las celdas adecuadas en la parte superior de la "G".

i.4 En LCOL de la parte superior, se tiene una de las salidas proporcionada por la celda (J), esta señal no está disponible en la parte inferior en donde se tiene un rectángulo negro.

ii) Para la columna de la derecha se tiene una configuración muy similar, adicionalmente se puede señalar que:

ii.1 Es LCOL quien permite el paso directo de una señal a través de la celda.

ii.2 Por RCOL, en la parte inferior se ingresa un dato de entrada (Din) necesario para la opera-

ción de la celda. Por RCOL de la parte superior se tiene la otra salida que proporciona la celda (K). Para esta celda la señal Din es diferente de la señal K, a pesar de estar las dos en RCOL.

iii) Para las dos filas de está celda se tiene que:

iii.1 Todos los caminos pasan directamente a través de la celda. No existe ninguna interrupción propia del *layout* de la celda.

iii.2 El camino ROW de la fila inferior, nombrado como *Load*, es una señal de control de la celda y en este caso puede ingresar a la celda por la derecha o por la izquierda y atraviesa la celda sin sufrir alteración .

En algunas celdas, como en el caso de la Fig. 8.15, las señales de salida están disponibles en los dos extremos de los caminos, así Φ_{in} y Φ_{out}, están presentes en RCOL tanto en la parte superior como en la inferior. La celda "4" también nos demuestra que las señales de entrada y/o salida de una celda pueden estar presentes ya sea en las filas y/o columnas y las interrupciones de los caminos, pueden estar también presentes en cualquier de los caminos de las filas: ROW, SROW, o TROW.

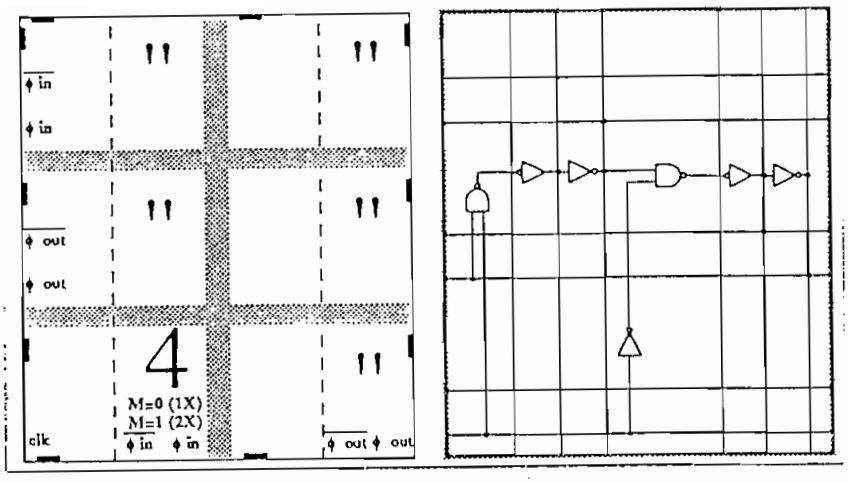


FIGURA 8.15 Celda "4"

c) Modificadores

Las celdas pueden modificarse para cumplir requerimientos diferentes. Los modificadores pueden afectar a las celdas de dos maneras, la primera es cambiar el tamaño de los transistores que están constituyendo la celda y la segunda es cambiar la función de la celda.

La representación simbólica de una celda modificada para variar el tamaño de los transistores se presenta en la Fig. 8.16. Esta figura contiene dos celdas "I" (inversor), la celda I de la izquierda está formada por una fila y una columna para valores de los modificadores de 0 y 1, la diferencia radica que con M=0 el tamaño de los transistores es 1X, siendo X simplemente una unidad de referencia, y con M=1 el tamaño de los transistores es 2X, es decir el doble que para el caso M=1.

Nótese que las dos celdas ocupan cada una un área unitaria. La celda I de la derecha, está formada por dos filas y una columna para M=2 y M=3. En el caso de M=2, los transistores de la celda tienen un tamaño de 4X y para M=3 el tamaño de los transistores es 8X.

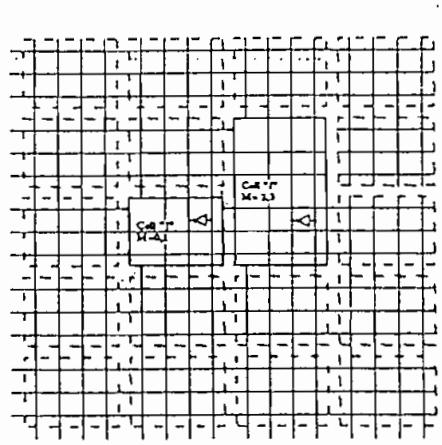


FIGURA 8.16 Celdas "I" con diferentes tamaños de transistores

Un ejemplo en el cual se modifica la función de una celda se muestra en la Fig. 8.17. Esta ilustración contiene la representación simbólica de dos celdas "r", una con M = 0 y otra con M = 1. Las dos celdas son muy similares, su diferencia radica en el hecho que con M = 0 tiene su entrada en el camino ROW y con M = 1 tiene su entrada en el camino SROW. Cuando se trabaja con TILER, el modificador por predefinición tiene un valor de 0; para el caso de M=0, por lo tanto, la celda "r" se inserta en la grilla sin especificar el valor del modificador.

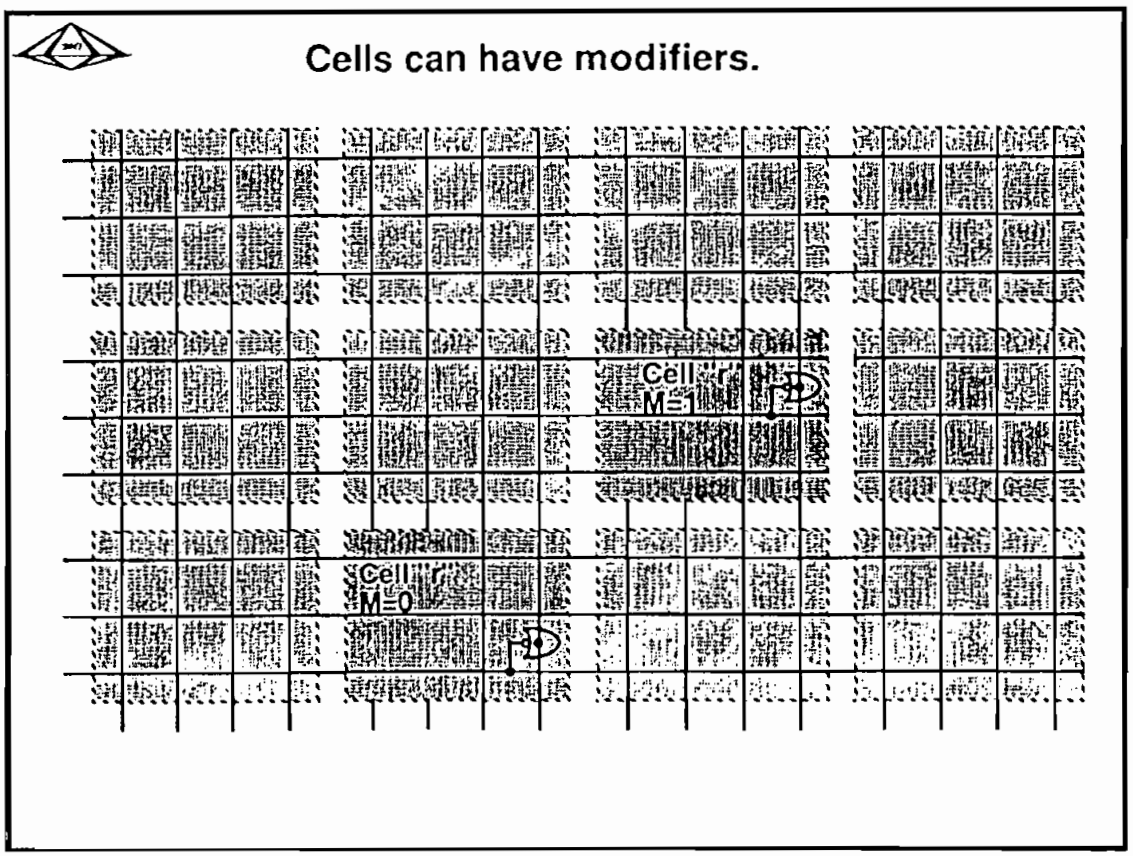


FIGURA 8.17 Celdas "r" con diferentes funciones

En realidad cada versión modificada de una celda puede ser considerada como una celda diferente, con su descripción propia, lo que demandaría de una lista demasiado extensa para nombrar a todas las celdas. En la mayoría de los casos, las celdas modificadas son en muchos aspectos equivalentes a las celdas no-modificadas y las otras modificadas. En el manual del paquete FPL se presenta la descripción detallada de cada una de las celdas, en esta descripción se presenta la información que es común a todas las celdas, sean modificadas o no, y luego se especifica el resultado de incluir un modificador diferente de cero.

d) Interrupciones (*breaks*)

Haciendo un pequeño resumen de la explicación dada sobre las interrupciones, se las puede clasificar en:

i) Interrupciones propias de las celdas: debidas a su *layout* particular, sobre las cuales el diseñador no puede actuar para modificarlas o retirarlas. Son de dos tipos:

i.1 Interrupciones forzadas (*Forced breaks*)

i.2 Interrupciones del stack serie (*Series Port*)

ii) Interrupciones añadidas por el diseñador de acuerdo a sus necesidades.

Las interrupciones que el diseñador desea utilizar, con el objetivo de eliminar el paso de una señal no deseada hacia cierta región de la grilla o parte del circuito, o como parte del proceso de enrutamiento de señales, lo hace colocando las interrupciones en la parte inferior y en el lado izquierdo de la representación simbólica de las celdas. La ubicación posible de los caracteres que representan las interrupciones, de filas y columnas (incluso de filas y columnas en la misma región), para el caso general de una celda múltiple se indica en la Fig. 8.18.

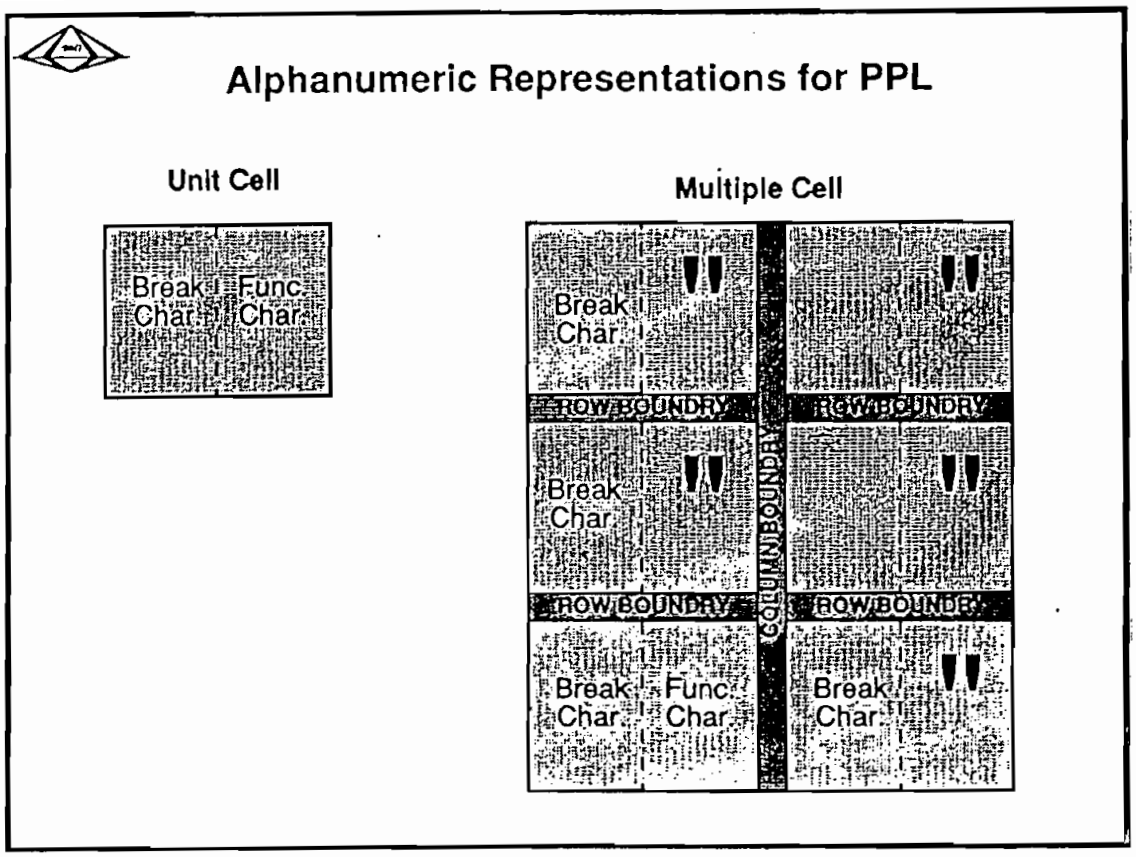


FIGURA 8.18 Ubicación de los caracteres de interrupción de filas y columnas

Cuando sea necesario interrumpir algún camino al lado derecho o en la parte superior de una celda, se lo debe hacer en la localidad de la grilla o en la celda que se encuentra inmediatamente a la derecha o en la parte superior de la celda en question, respectivamente.

Cuando un diseño está concluído (o un módulo del diseño total) se lo debe aislar de la grilla PPL, dejando intactos unicamente los caminos que llevan las señales de entrada y salida e interrumpiendo todos los caminos restantes. Si los caminos que se desean interrumpir están en celdas ubicadas en

el extremo de un módulo determinado, la interrupción se debe realizar en las localidades de la grilla localizadas más a la derecha o arriba de las celdas del límite del módulo. Por lo tanto, los símbolos de las interrupciones están ocupando una localidad de la grilla fuera del módulo y la región de la misma reservada para el nombre de la celda no es utilizada.

La Fig. 8.19 presenta un pequeño módulo diseñado en base a las celdas PPL. Este módulo permite aclarar las ideas expuestas anteriormente. Para aislar el módulo de la grilla, se interrumpen todos los caminos horizontales y verticales (arriba, abajo, a la derecha, y a la izquierda) a excepción de las señales de entrada y salida del módulo. Las interrupciones necesarias en el lado izquierdo y parte inferior del módulo se colocan dentro de la celdas del módulo.

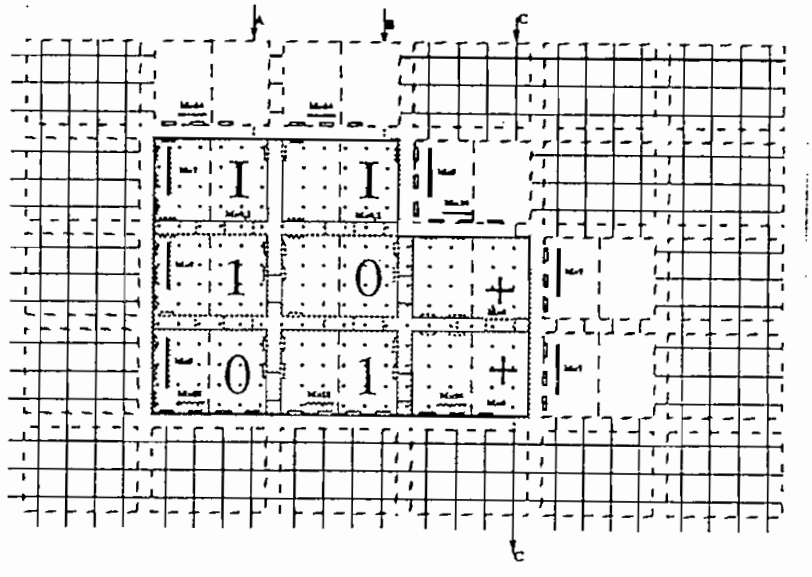


FIGURA 8.19 Ubicación de los caracteres de interrupción para aislar un módulo

En el extremo izquierdo se lo hace en las celdas I, 1, O, con el caracter "¦". Para la parte inferior del módulo, las interrupciones de los caminos de las columnas se representan con el caracter "-".

En el ejemplo también se demuestra la manera de representar el caso en que se interrumpen los caminos hacia/desde el módulo en la parte derecha y superior del mismo. En este caso se colocan las interrupciones a continuación del módulo, utilizando localidades unitarias de la grilla PPL, a la derecha y sobre el módulo.

Al igual que las interrupciones propias de las celdas, las interrupciones que el diseñador desea utilizar, pueden representarse por pequeños rectángulos centrados sobre el camino que están bloqueando. (Fig. 8.19).

Los dos caracteres que se utilizan, por lo general, para representar las interrupciones, son: "_" para las interrupciones de los caminos de las columnas, y "¦" para las interrupciones de los caminos de las filas; sin embargo, debe señalarse que en TILER no son los únicos caracteres utilizados, se utilizan también un conjunto de caracteres de acuerdo a los caminos interrumpidos.

El conjunto de interrupciones disponible (*predefined*) se presenta en la Fig. 8.20, el pequeño rectángulo indica el camino que está siendo bloqueado y debe notarse que el carac-

ter que representa a la interrupción está en la región izquierda de la línea punteada, lo que indica que puede combinarse con una celda PPL, cuyo nombre estaría en la región derecha de la línea punteada.

Las interrupciones, como las celdas, pueden también modificarse. Podría suponerse que a cada uno de los caminos se le ha asignado un peso de ponderación de acuerdo a su ubicación, muy similar a lo utilizado en el sistema binario de numeración.

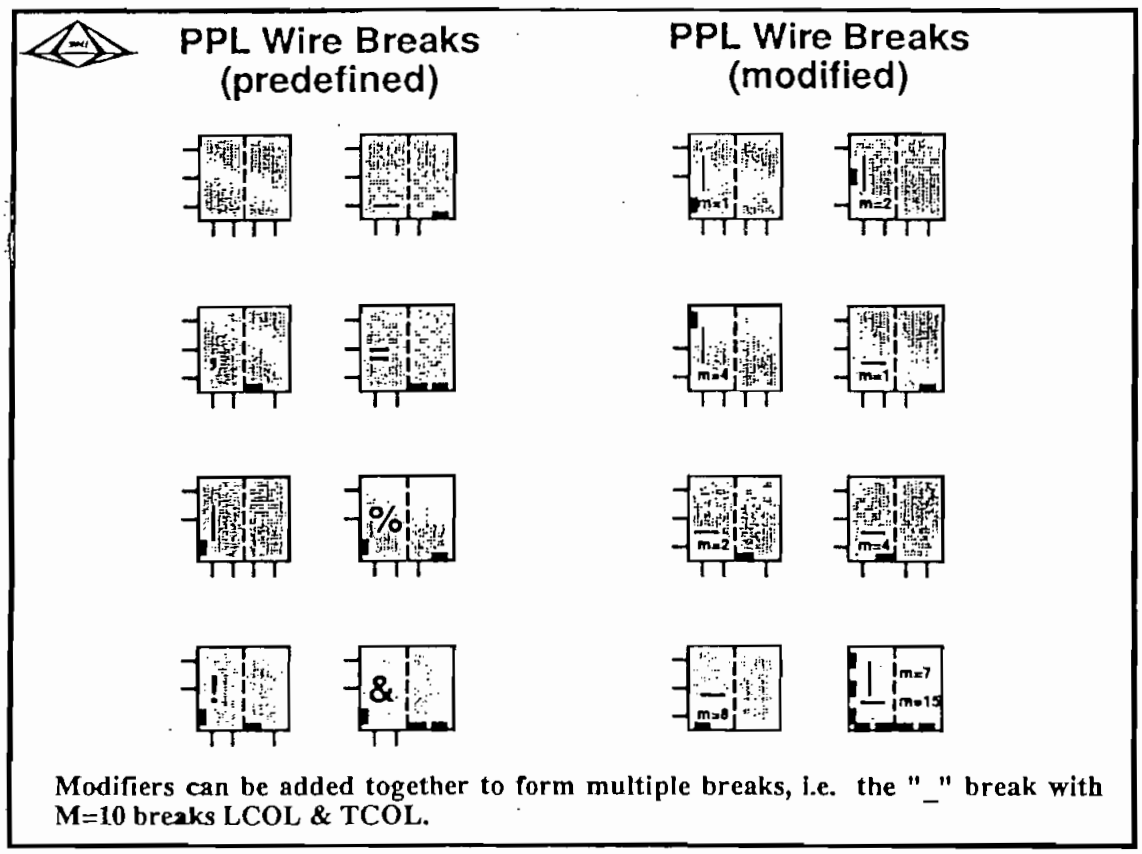


FIGURA 8.20 Conjunto de interrupciones predefinidas y modificadas

Para las columnas:

TCOL	SCOL	LCOL	RCOL
8	4	2	1

Para las filas:

TROW	SROW	ROW
4	2	1

El conjunto de interrupciones modificadas (*modified*) se presenta también en la Fig. 8.20. Los valores de las interrupciones modificadas pueden sumarse entre sí para conseguir cualquier combinación de interrupciones, (en la Fig. 8.20 se presenta, en la última celda, el caso para M=7 en las filas y M=15 en las columnas).

Las interrupciones pueden hacerse tanto para filas como para columnas simultaneamente, desde este punto de vista es más práctico realizar esta operación con las interrupciones modificadas y sumando los modificadores. Además se debe considerar que el valor del modificador para filas es independiente de aquel para las columnnas.

Por ejemplo, un valor de interrupción modificado para columna de 10 (1010); interrumpe los caminos TCOL y LCOL. Un valor modificado para fila de 5 (101) interrumpe los caminos TROW y ROW.

Un ejemplo de lo mencionado anteriormente, puede apreciarse revisando los valores de los modificadores asociados a las celdas de interrupción de la Fig. 8.19, en la cual se representan las interrupciones utilizando los caracteres "¦" y "_" . La celda " 0 " de la esquina inferior izquierdo indica la manera de representar interrupciones tanto de filas como de columnas. En aquellas celdas que permiten el paso de señales debe notarse el valor del modificador, escogido de tal manera de no interrumpir los caminos que llevan dichas señales.

e) Celdas básicas

El conjunto de celdas PPL incluye las celdas básicas necesarias para realizar las operaciones AND y OR, lo que determina una estructura equivalente a un arreglo lógico programable (PLA); se incluyen también dentro del conjunto de celdas: flip-flops, inversores, celdas de interconexión de filas con columnas, cargas y transistores de paso y muchas estructuras adicionales que facilitan el desarrollo de circuitos combinacionales y secuenciales.

Las celdas básicas PPL: '0', '1', 's', 'r/+', 'I' son todas celdas unitarias, son celdas individuales que pueden utilizarse como los bloques constituyentes de casi todo circuito. En la Fig. 8.21 se muestran las celdas básicas PPL con su representación nivel simbólico, lógico y de transistores.

Las celdas básicas '0', '1', 's' y 'r/+' están constituídas por un par de transistores, un "p" y un "n", y funcionan como parte de las compuertas NAND, que se estructuran para ser utilizadas como elemento primitivo de diseño. La elección de la celda a utilizar depende de la posición relativa de dichas celdas y de las señales que deban ser manejadas hacia las compuertas.

Las celdas 'I' y 'V', que no se presenta en la Fig. 8.21, son inversores que están orientados en diferentes direcciones, con sus señales de entrada y salida en los caminos verticales.

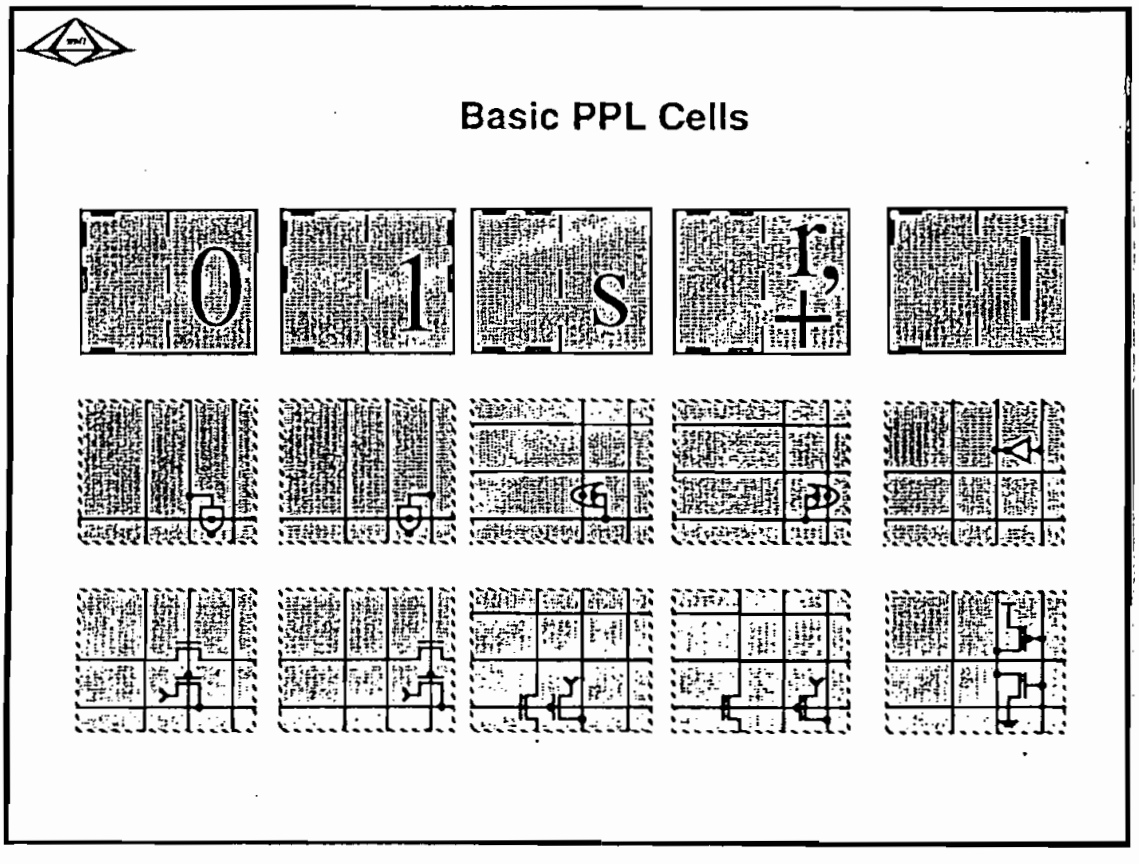


FIGURA 8.21 Celdas básicas PPL

8.2.3 Estructuras AND y OR utilizando celdas básicas PPL

A continuación se presentan algunos ejemplos sencillos utilizando las celdas básicas unitarias para realizar las funciones AND y OR. Se introducen también las ideas referentes a lógica positiva y negativa que simplifican el diseño y que se utilizan para la descripción de las funciones que realiza cada una de las celdas PPL.

a) Lógica "mixta" PPL.

El tipo de lógica que se utiliza en un sistema está determinado por el significado lógico asociado con los voltajes físicos en cualquiera de los nodos de un sistema.

Si los nodos utilizan un voltaje ALTO para representar un valor lógico VERDADERO (*high-true*) y un voltaje BAJO para un valor lógico FALSO (*low-false*), se dice que utilizan **logica positiva.**

Voltaje alto = Verdadero V_{ALTO}= V
Voltaje bajo = Falso V_{BAJO}= F

Si los nodos utilizan un voltaje BAJO para representar un valor lógico VERDADERO (*low-true*) y un voltaje ALTO para un valor lógico FALSO (*high-false*), se dice que utilizan **logica negativa.**

Voltaje alto = Falso	V_{ALTO}= F
Voltaje bajo = Verdadero	V_{BAJO}= V

Cuando se utilizan los términos NAND y NOR para referirse a dichas compuertas, implicitamente se asocia la lógica positiva, de tal manera que NAND y NOR se refieren más bien a la función eléctrica de la compuerta.

La "COMPUERTA NAND" es un circuito que tiene un voltaje bajo en su salida siempre que todas sus entradas tengan un voltaje alto, y una "COMPUERTA NOR" es un circuito que tiene un voltaje bajo en su salida siempre que cualquiera de sus entradas tenga un voltaje alto.

Para la compuerta NAND puede resumirse su comportamiento eléctrico en la siguiente tabla:

A	B	$\overline{A.B}$
V_{BAJO}	V_{BAJO}	V_{ALTO}
V_{BAJO}	V_{ALTO}	V_{ALTO}
V_{ALTO}	V_{BAJO}	V_{ALTO}
V_{ALTO}	V_{ALTO}	V_{BAJO}

Para lógica positiva, la tabla anterior expresada en términos de sus valores lógicos sería:

A	B	$\overline{A.B}$
F	F	V
F	V	V
V	F	V
V	V	F

Por lo tanto, hablar de lógica positiva en definitiva es referirse de manera directa a los voltajes que físicamente existen, esto es con lo que el ingeniero está acostumbrado a trabajar.

Las celdas PPL están basadas en sistemas de lógica mixta. Esto permite que las estructuras AND-OR puedan implantarse físicamente utilizando o solo compuertas NAND o solo compuertas NOR. El tipo particular de compuertas utilizado depende del proceso de fabricación del circuito (nMOS, CMOS, GaAs, etc) y en base a consideraciones para su operación.

En la mayoría de los procesos CMOS, se prefiere utilizar las compuertas NAND, de tal manera que los transistores canal "n", más rápidos, puedan ubicarse en serie y los transistores canal "p", más lentos, se conecten en paralelo. Esta configuración usualmente puede operar a velocidades más altas que su correspondiente configuración utilizando compuertas NOR, en la cual se ponen transistores canal "p" en serie y transistores canal "n" en paralelo. La metodología PPL, para CMOS, está desarrollada de tal manera que la mayoría de las funciones lógicas se realicen con compuertas NAND.

El conjunto de celdas basado en compuertas NOR, utilizan lógica positiva para los caminos de las filas y negativa para las columnas. El conjunto de celdas que se basan en compuertas NAND utilizan lógica positiva en los caminos de las columnas y lógica negativa en los caminos de las filas.

b) Estructura típica de una compuerta NAND para CMOS

Realizando un pequeño resumen de la estructura típica de una compuerta NAND CMOS (Fig. 8.22), se puede señalar las siguientes características:

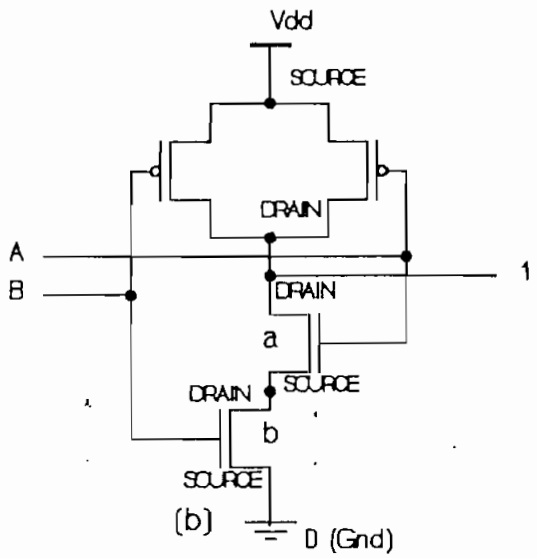


FIGURA 8.22 Estructura típica a nivel de transistores de una compuerta NAND

i) Se tienen dos señales de entrada (A,B) cada una controlando un transistor "n" y un transistor "p",de acuerdo a la tecnología CMOS, a través de sus compuertas.

ii) Los dos transistores "p" están en paralelo, tienen conectadas sus fuentes a V_{DD}, y tienen sus drenajes conectados a un mismo punto (1).

iii) Los dos transistores "n" están conectados uno a continuación de otro, la fuente del transistor (a) está unido electricamente al drenaje del transistor (b), el drenaje del transistor (a) está conectado al punto (1), y la fuente del transistor (b) está conectado a tierra (Gnd).

iv) Para el caso más general, en el que se tenga una compuerta de "n" entradas, se tiene un grupo de "n" transistores nMOS conectados uno a continuación de otro, formando un stack de "n" transistores en SERIE que llevan a la salida a un voltaje bajo (pull-down), sin que el orden de los transistores dentro del stack sea relevante. El un extremo del stack estaría conectado a tierra (Gnd) y el otro extremo estaría conectado a un grupo de "n" transistores pMOS conectados en paralelo, que actúan para llevar a la salida a un voltaje alto (pull-up).

La representación a nivel de transistores de la Fig. 8.22 nos proporciona una visión clara de la estructura de la compuerta NAND.

A continuación se analiza como con PPL facilmente puede obtenerse la estructura de una compuerta NAND típica, para luego pasar a analizar el uso de esta compuerta en las funciones AND y OR, considerando para ello la lógica mixta utilizada por PPL.

i) Caso 1 : Compuerta NAND con las entradas en RCOL y salida en ROW: $(\overline{A.B})$

En la Fig 8.22 se muestra la representación convencional de una compuerta NAND y la manera en la que se distribuye dicha compuerta al utilizar las celdas PPL. Se requieren dos celdas "1" para estructurar la compuerta NAND. En la figura se incluyen también las representaciones simbólica, lógica y a nivel de transistores de las celdas empleadas.

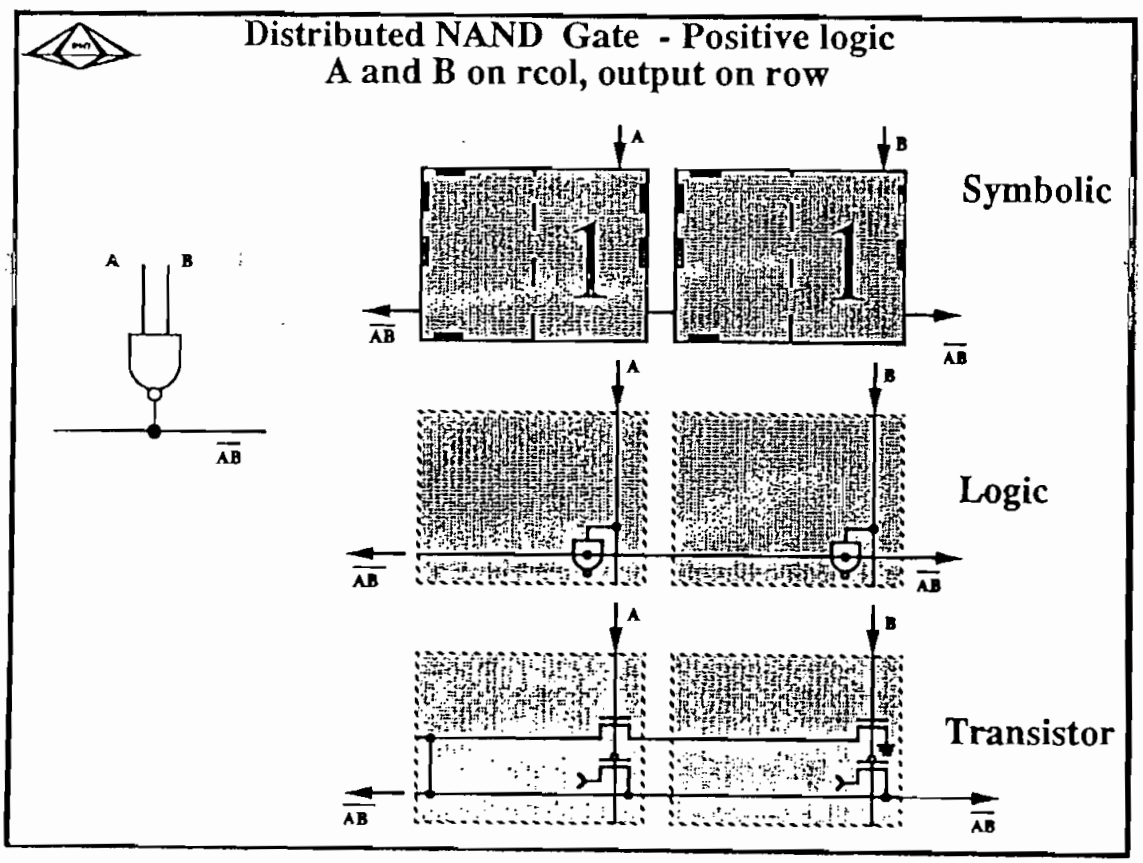


FIGURA 8.23 Compuerta NAND utilizando dos celdas "1" en la misma fila

La compuerta se forma en base a dos celdas "1" colocadas una junto a otra, en la misma fila, las señales de entrada a estas celda están en los caminos RCOL (de cada celda) y la salida está disponible en el camino horizontal ROW.

Si se analiza en detalle la estructura de la celda "1" a nivel de transistores, puede notarse que está constituída por un transistor "n" y un "p" controlados por medio de su compuerta (*gate*) con la señal que ingrese por RCOL.

El transistor "n" tiene sus otros dos terminales en SROW, uno al lado derecho y el otro al lado izquierdo. Asignar a uno de los extremos como fuente/drenaje y al otro como drenaje/fuente, no es relevante debido a que se trabaja a nivel simbólico y se lo puede hacer a voluntad. Con motivos de explicación, se asume que la fuente del transistor "n" es el terminal de la derecha y drenaje el de la izquierda. El drenaje del transistor "p" está conectado al camino ROW, y su fuente a V_{DD}.

Cuando se colocan dos celdas "1" en la misma fila, automaticamente quedan conectados la fuente del transistor "n" de la señal de entrada A con el drenaje del transistor "n" de la señal de entrada B; lo mismo ocurre con los dos drenajes de los transistores "p". De esta manera se ha formado el conjunto de transistores "p" en paralelo y el *stack* de transistores "n" conectados en serie.

Para tener una estructura NAND, como la típica presentada en la Fig. 8.22, se debería conectar un extremo del *stack* al conjunto de transistores "p" y el otro extremo a Gnd. Esta tarea no la debe realizar el diseñador sino que es PPL quien de manera automática realiza esta función.

La conexión a tierra se la conoce como un "*ground*" y la conexión al grupo de transistores en paralelo se conoce como un "*short*". El resultado de colocar un "*short*" y un "*Gnd*" puede apreciarse en la representación a nivel de transistores de la Fig. 8.23. Considerando lo asumido en cuanto a cual de los terminales es fuente y cual es drenaje, se llega a demostrar la equivalencia entre la estructura típica y la logrado con las celdas PPL.

ii) Caso 2 : Compuerta NAND con entradas en RCOL y LCOL y salida en ROW: $(\overline{A.B})$

Como regla general o convención, no una condición inviolable, pero que luego permite obtener información directa de caracter lógico de la distribución de las celdas dentro de un diseño, por simple observación, se asume que el camino RCOL se utiliza para enrutar las señales VERDADERAS (originales) y el camino LCOL es el encargado de llevar las señales FALSAS (invertidas). Considerando lo anterior, para realizar la operación $\overline{A.B}$ se utiliza la estructura de la Fig. 8.24.

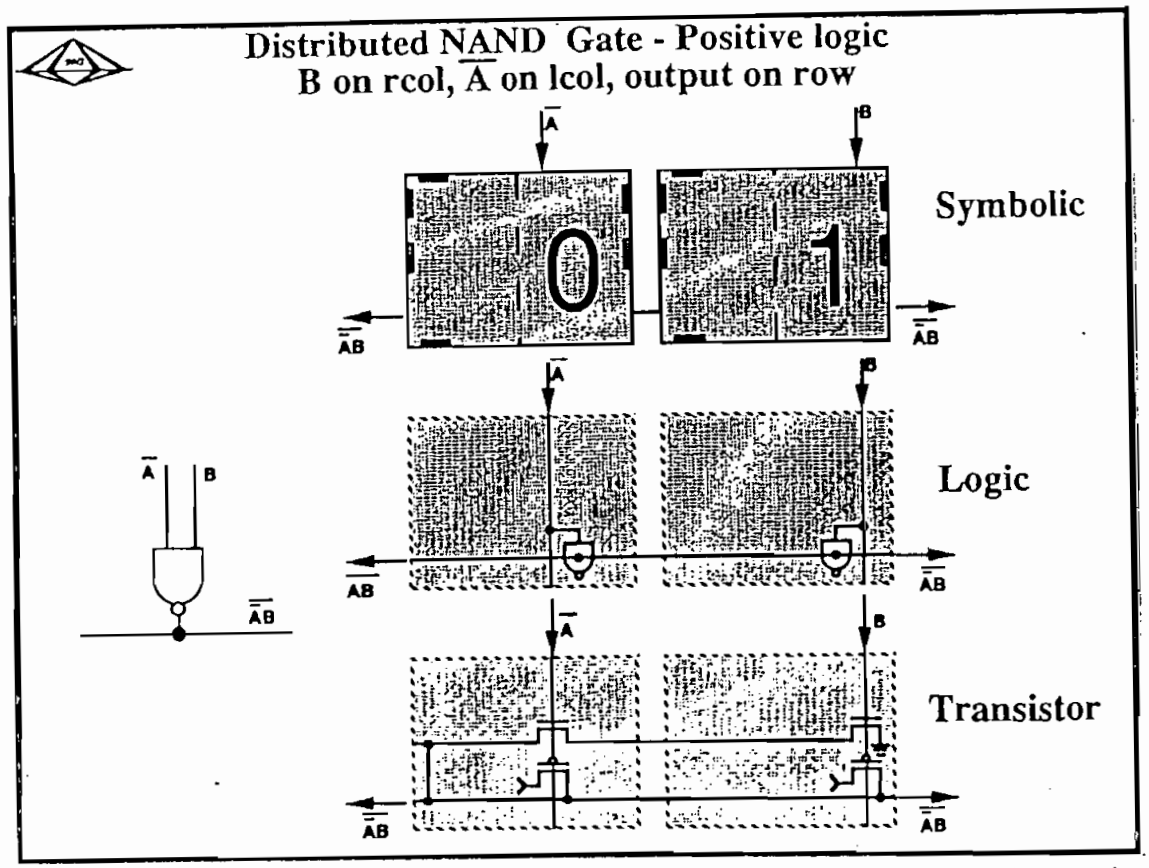


FIGURA 8.24 Compuerta NAND con entradas en LCOL y RCOL.

La compuerta es de igual manera implantada en base a dos celdas, en este caso: "0" y "1". Observando especialmente la representación a nivel de transistores, se puede ver que la estructura de la compuerta es idéntica a la del caso 1. La diferencia que se encuentra es precisamente que la señal de entrada A complementada ingresa a la compuerta por el camino LCOL y es la celda "0" la que permite este tipo de enrutamiento. La estructura de la celda "0" es muy similar a la celda "1", con la única diferencia que la señal de control de las compuertas está ahora en el camino LCOL, y no en RCOL.

iii) Caso 3 : Compuerta NAND con entradas en ROW y salida en RCOL: $(\overline{A.B})$

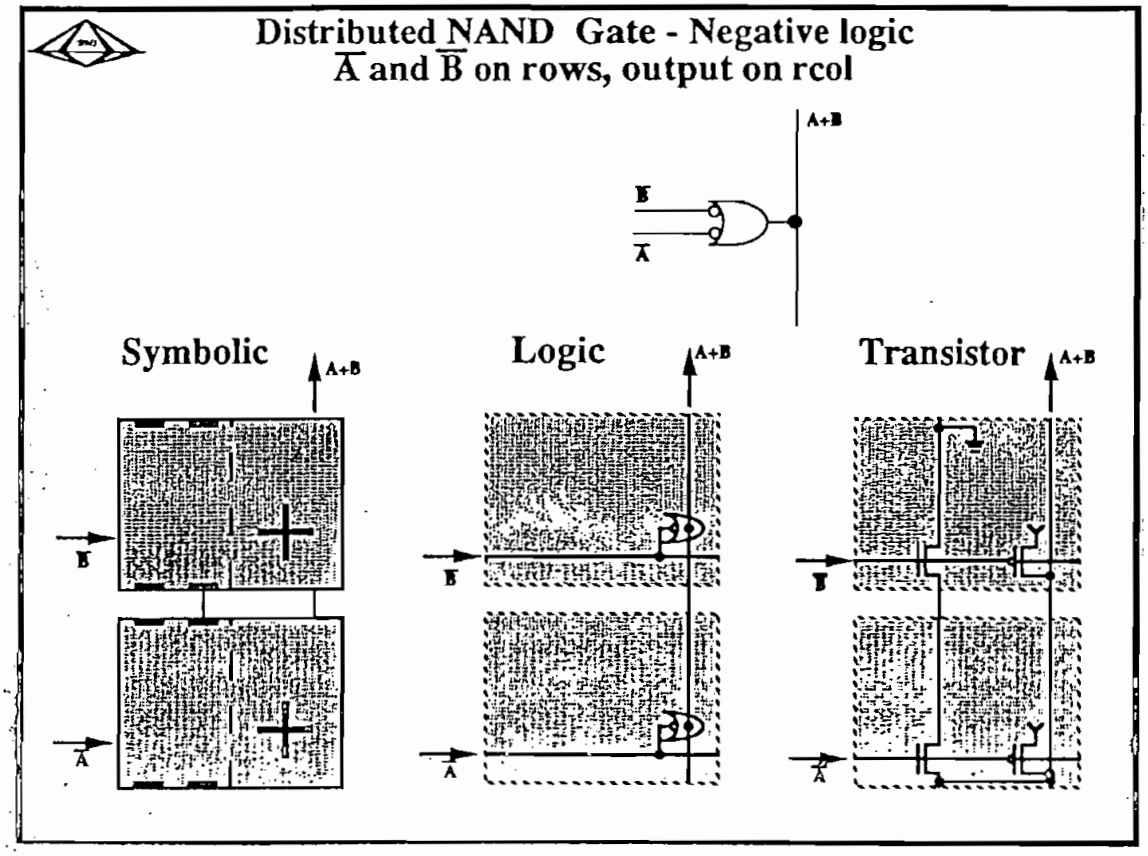


FIGURA 8.25 Compuerta NAND utilizando celdas "+" dentro de una columna.

En la Fig. 8.25 se presenta la implantación de una compuerta NAND utilizando dos celdas "+" dentro de una misma columna. El resultado es el de la estructura de la compuerta NAND que se obtuvo para los casos 1 y 2; esto puede comprobarse analizando el esquema a nivel de transistores de la Fig. 8.25. La compuerta tiene sus 2 entradas en las filas y su salida en el camino vertical RCOL.

Una compuerta NAND puede ser utilizada para realizar la función OR. Al trabajar con lógica positiva, desde el punto de vista eléctrico de niveles de voltaje, las señales de entrada A y B deben ser negadas como paso previo a ingresar las señales a la compuerta NAND. Por lo tanto la compuerta NAND realizaría la operación $\overline{\overline{A}.\overline{B}}$, que aplicando las leyes del Algebra de Boole nos proporciona finalmente la función A+B. Por la explicación dada más adelante, para la obtención de las funciones AND y OR, utilizando lógica mixta, la compuerta NAND formada en las "columnas" ha sido elegida para presentar la formación de la función OR.

iv) Caso 4 : Compuerta NAND con entradas en ROW y salida en LCOL: $(\overline{A.B})$

La Fig. 8.26 presenta también la estructuración de una compuerta NAND, con la estrutura típica que se ha utilizado en todos los casos anteriores. La diferencia con respecto al caso 3 es que la señal de salida de la compuerta se presenta en LCOL en lugar de hacerlo en RCOL. En este caso la compuerta NAND se la obtiene colocando dos celdas "s" en una columna. La celda "s" es similar a la "+" con la diferencia que su salida está en LCOL en lugar de RCOL. El nombre "s" se lo ha asignado debido a que se la utiliza como la entrada SET de las celdas de flip-flops, en la presente aplicación se la utiliza para obtener una compuerta NAND. Al igual que en el caso anterior las señales deberían ser previamente negadas para conseguir la función OR.

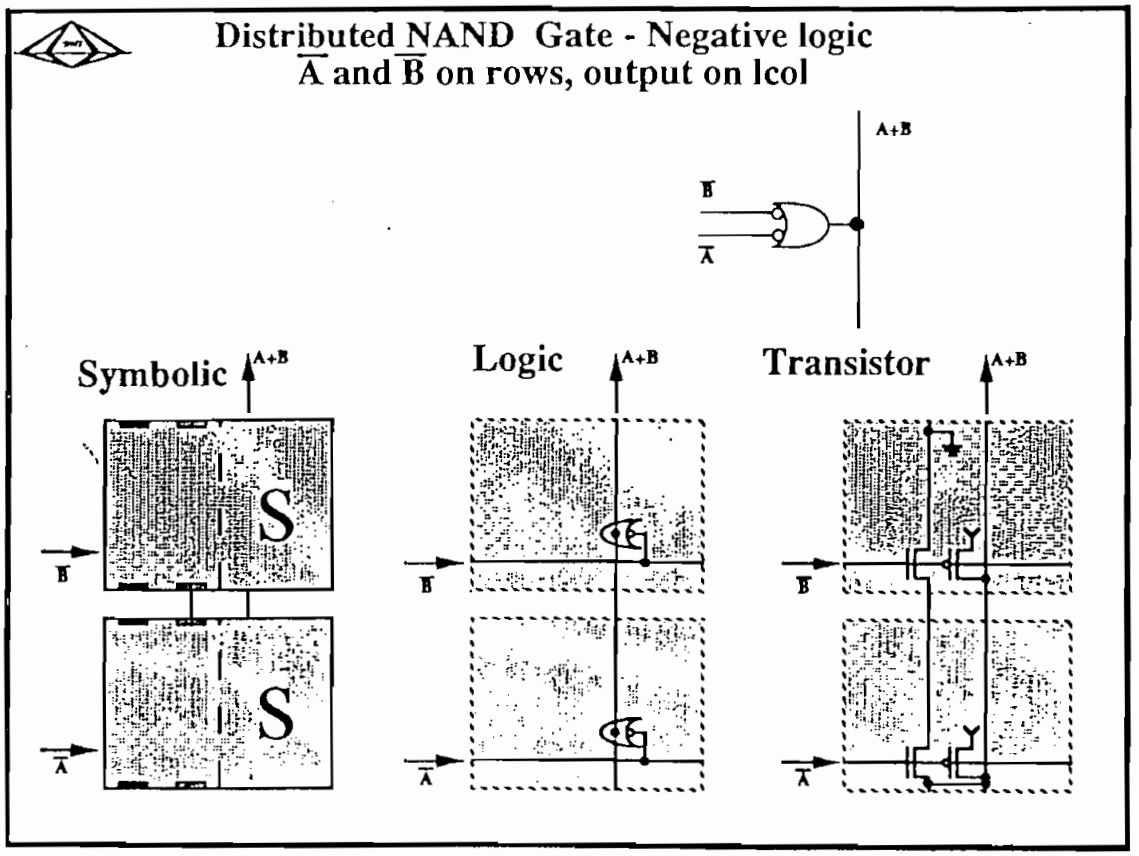


FIGURA 8.26 Compuerta NAND utilizando celdas 's' dentro de una columna

c) La función AND utilizando lógica mixta PPL

Considerando la estructura de la compuerta NAND, conseguida en las filas en los casos 1 y 2 anteriores, se analiza dicha estructura conjuntamente con la lógica mixta adoptada por PPL para obtener la función AND.

En los casos 1 y 2 analizados, las entradas de las compuertas NAND están en las columnas (lógica positiva) y las salidas se obtuvieron en las filas (lógica negativa).

Se demuestra a continuación que la distribución de celdas básicas: "0s" y/o "1s" , dentro de una misma fila, realiza la funcion AND si se considera la lógica mixta utilizada por PPL. (Fig. 8.27).

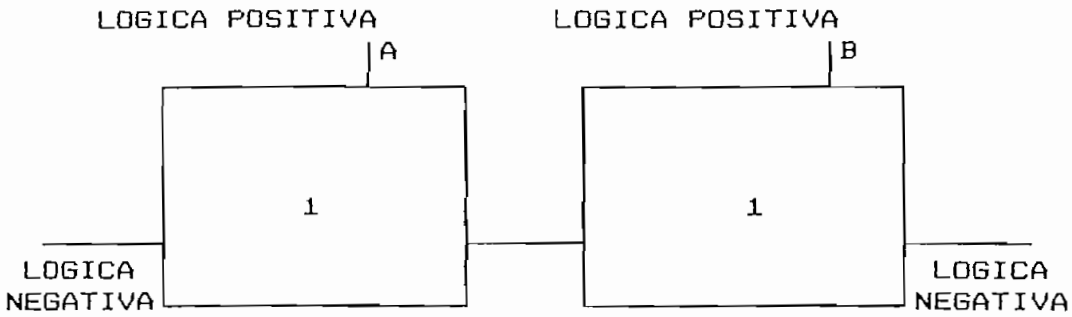


FIGURA 8.27 Función AND en base a una compuerta NAND, utilizando la lógica mixta PPL.

La tabla de verdad de la función AND es:

A	B	A.B
F	F	F
F	V	F
V	F	F
V	V	V
COLUMNAS LOGICA "+"		FILA LOGICA "-"

Asociando a los valores de verdad (V, F) de las entradas y salida, los niveles de voltaje, de acuerdo a la lógica utilizada por filas y columnas se tiene:

Las columnas trabajan con lógica positiva:

$V_{ALTO} = V$
$V_{BAJO} = F$

Las filas trabajan con lógica negativa:

$V_{ALTO} = F$
$V_{BAJO} = V$

El resultado es:

A	B	A.B
V_{BAJO}	V_{BAJO}	V_{ALTO}
V_{BAJO}	V_{ALTO}	V_{ALTO}
V_{ALTO}	V_{BAJO}	V_{ALTO}
V_{ALTO}	V_{ALTO}	V_{BAJO}

que corresponde al comportamiento eléctrico de la compuerta NAND que se estructuró con las celdas PPL.

Si por un momento se asume que la finalidad del diseño era obtener un CI que realice la función AND, el análisis realizado utilizando lógica mixta conduce a un resultado erróneo, el diseño realizado con las celdas "0", "1", en realidad entrega como resultado la operación NAND. La explicación en base a valores lógicos (lógica positiva y negativa) tiene validez pero no desde el punto de vista eléctrico y es de gran utilidad en el diseño de circuitos más complejos ya que permite simplificar la implantación de un circuito y también obtener facilmente información de un circuito ya diseñado, planteando directamente ecuaciones en función de términos AND.

d) La función OR utilizando lógica mixta PPL

Utilizando nuevamente el concepto de lógica mixta, basados en la estructura de la compuerta NAND de los casos 3 y 4, se demuestra que se puede implantar la función OR.

Las señales de las cuales se desea obtener la función A+B no son negadas previamente como se indicó en el análisis previo del caso 3, la metodología consiste simplemente en considerar que las dos señales de entrada trabajan con lógica negativa (filas) y que la señal de salida trabaja con lógica positiva (columnas).

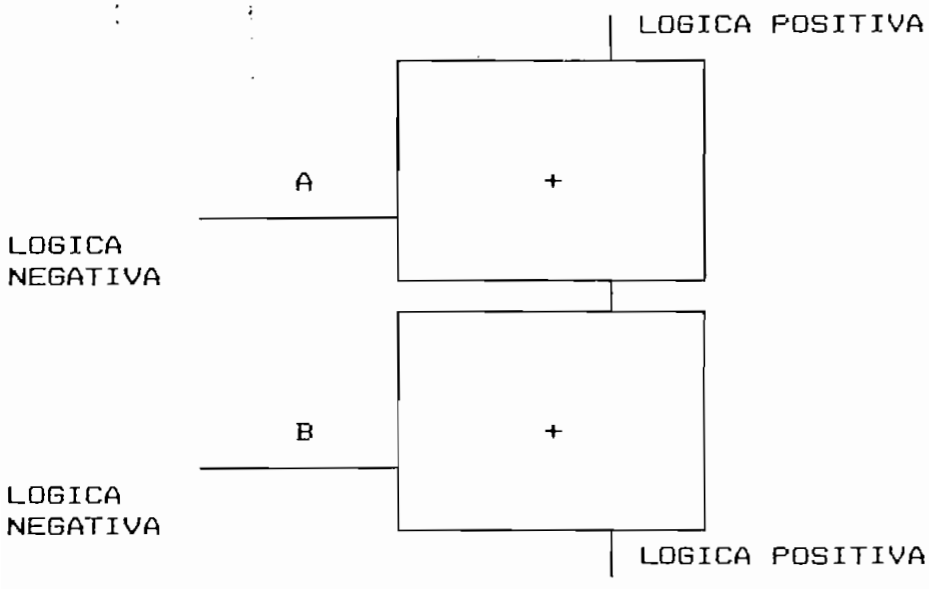


FIGURA 8.28 Función OR en base a una compuerta NAND, utilizando lógica mixta.

La tabla de verdad para la función OR es:

A	B	A+B
F	F	F
F	V	V
V	F	V
V	V	V

FILAS LOGICA "−"

COLUMNA LOGICA "+"

Asociando los niveles de voltaje a las entradas y salida, de acuerdo a la lógica utilizada por cada una de ellas, el resultado es:

A	B	A+B
V_{ALTO}	V_{ALTO}	V_{BAJO}
V_{ALTO}	V_{BAJO}	V_{ALTO}
V_{BAJO}	V_{ALTO}	V_{ALTO}
V_{BAJO}	V_{BAJO}	V_{ALTO}

Reordenando las filas de la tabla anterior:

A	B	A+B
V_{BAJO}	V_{BAJO}	V_{ALTO}
V_{BAJO}	V_{ALTO}	V_{ALTO}
V_{ALTO}	V_{BAJO}	V_{ALTO}
V_{ALTO}	V_{ALTO}	V_{BAJO}

Como se deseaba demostrar, esta tabla corresponde al comportamiento eléctrico de la compuerta NAND que se está utilizando para implantar la función OR.

De igual manera que cuando se obtuvo la función AND, si el objetivo final hubiese sido el disponer de un CI que realice la función OR, el análisis que se acaba de realizar nos conduce a un resultado equivocado, pues lo que se obtiene es el resultado de la operación NAND. Para obtener la función OR deseada, se debería haber invertido previamente las señales, para luego ingresar a la compuerta NAND, como se explicó anteriormente.

Cuando se diseñan circuitos más complejos, las ideas expuestas anteriormente para realizar las funciones AND y OR se utilizan conjuntamente y la facilidad que proporcionan queda de manifiesto en el ejemplo de la función EXOR que se presenta para aclarar los conceptos explicados.

Como es claro, de toda la explicación anterior, el diseñador puede hacer caso omiso del tipo de lógica utilizada, ya sea en filas o columnas, y guiarse por valores de voltajes. Pero como se desprende de la práctica con PPL, el utilizar los conceptos de lógica mixta facilita la tarea de diseño, que es el objetivo de este paquete, y por otro lado en la descripción individual de las celdas se menciona la lógica mixta con mucha frecuencia y para la comprensión de la finalidad de la existencia de cada celda es necesario entender dichos conceptos.

La explicación realizada se centró en los casos de compuertas con dos entradas. Para conseguir una compuerta AND

de "n" entradas, lo único que debe hacerse es colocar "n" celdas ya sean "1" y/o "0" en total, en la misma fila. Para el caso de la función OR de "n" entradas, se deben colocar "n" celdas "r" y/o "s" en total, en la misma columna.

Las celdas individuales básicas no son las únicas que permiten estructurar compuertas NAND, así:

i) pueden combinarse dentro de un fila las celdas: 0, 1, j, w, a, o, X, E.

ii) en las columnas pueden combinarse las celdas: +, r, v para proporcionar salida en RCOL, y las celdas: s, i con salidas en LCOL.

8.2.4 Diseño de la compuerta OR EXCLUSIVA (EXOR)

Con la ayuda de este ejemplo se demuestra la manera de combinar las estructuras PPL de compuertas NAND formadas tanto en filas como en columnas y especialmente la ventaja de utilizar las funciones AND y OR conseguidas en base a la lógica mixta. Se debe mencionar también que el conjunto de celdas PPL incluye una celda que cumple la función or-exclusiva (EXOR), el presente desarrollo es tan sólo con fines didácticos.

En la Fig. 8.29 se presenta un diagrama esquemático de la compuerta EXOR, utilizando la representación convencional de las compuertas, se utilizan inversores y compuertas NAND para formar la función $A.\bar{B} + \bar{A}.B$, que es la forma expandida de $A \oplus B$.

Para desarrollar este ejercicio con PPL se utilizan las celdas básicas: "0", "1", "+" ,"I".

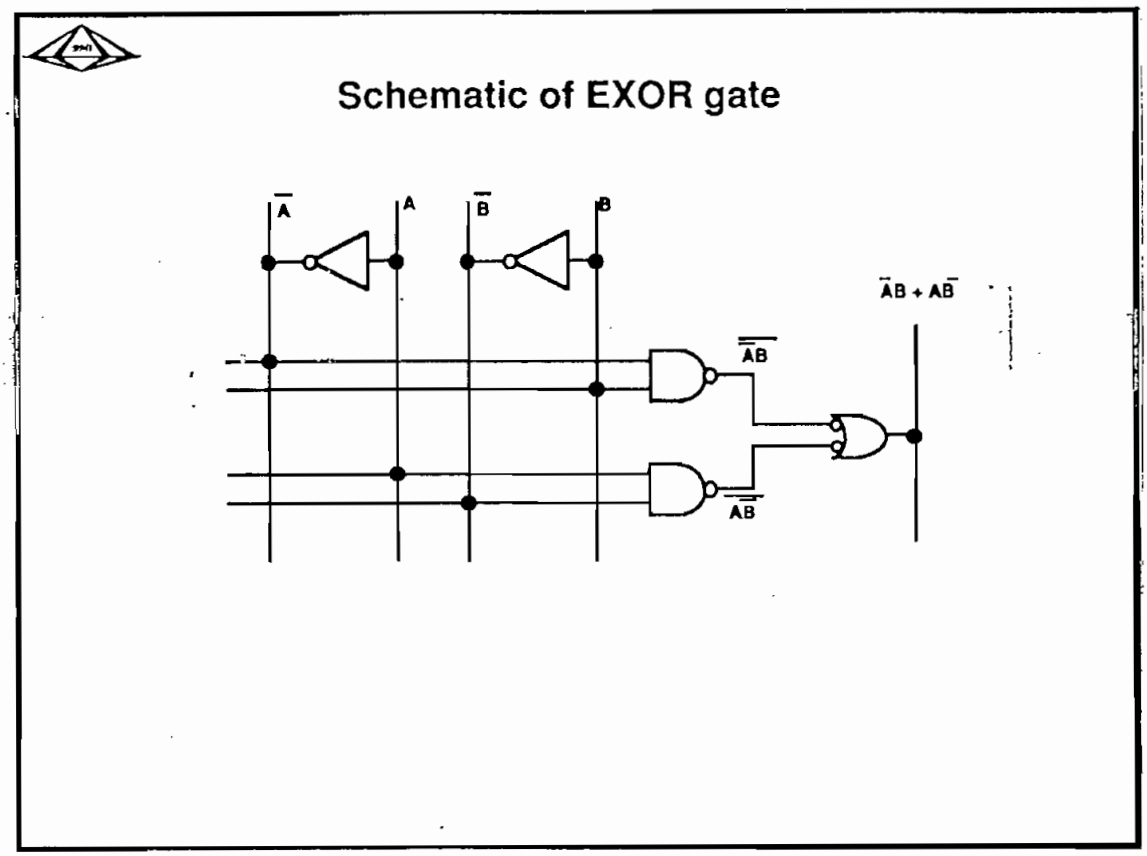


FIGURA 8.29 Representación de la función EXOR con simbología tradicional en base a compuertas NAND.

La representación simbólica de la función EXOR utilizando las celdas PPL se presenta en la Fig. 8.30.

Con el objeto de comprobar que la estructura presentada en la Fig. 8.30 corresponde efectivamente a la función EXOR puede realizarse un análisis, sin considerar la lógica mixta PPL, planteando las ecuaciones para las compuertas NAND que se forman y aplicando el Algebra de Boole.

La primera fila del arreglo contiene dos celdas "I" que permiten obtener el complemento de las señales de entrada. La segunda fila realiza la función NAND de las señales A complementada y B. La tercera fila realiza la función NAND de A y B complementada. Para los dos casos, las señales ingresan por las columnas y se obtiene el resultado en las filas.

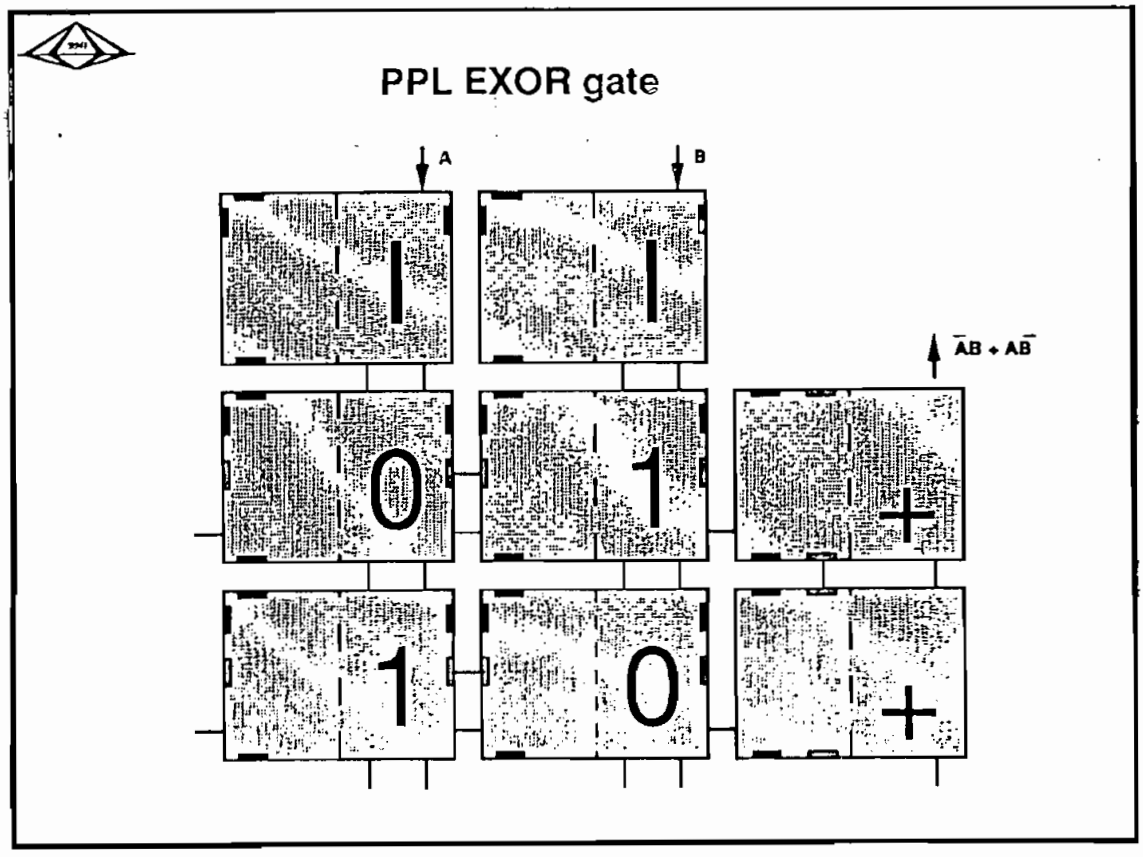


FIGURA 8.30 Representación simbólica de la función EXOR implantada con celdas PPL .

Las dos celdas "+" forman una nueva compuerta NAND, la cual tiene como entradas los términos obtenidos anteriormente en las filas. El resultado de la función NAND de las señales de las filas es: $\overline{\bar{A}.B}$ y $\overline{A.\bar{B}}$, lo que aplicando el Algebra de Boole se reduce a: $\bar{A}.B + A.\bar{B}$.

Este análisis demuestra que la configuración de celdas PPL de la Fig. 8.30 cumple con la función planteada.

El realizar todo el análisis en base a compuertas NAND, que involucra el aplicar el algebra de Boole, puede convertirse en un proceso tedioso y con una gran probabilidad de incurrir en errores, sobre todo en circuitos extensos y más complejos. A continuación se demuestra como facilita el haber desarrollado las funciones AND y OR con el uso de la lógica mixta de PPL en filas y columnas.

La función a realizar incluye dos términos AND y un OR. Se debe recordar que si se desea implantar la función AND (utilizando los conceptos de lógica mixta) de algunas señales, éstas deben ingresar por las columnas a celdas "0" y/o "1" y el resultado se obtiene en las filas. Como los términos AND incluyen las señales complementadas, se utilizan celdas "I". Cada término AND se forman con una fila de celdas colocando directamente celdas "0" y "1", por cada término del producto. El término $\bar{A}.B$ se obtiene en la segunda fila de celdas utilizando una celda "1" para la señal B y una celda "0" para enrutar el complemento de A.

El segundo término AND se obtiene en una fila diferente (la tercera), con una estructura muy similar a la descrita para el otro producto, utilizando esta ocasión una celda "1" para la señal A y una celda "0" para la señal B complementada. En las filas se tienen por lo tanto los términos AND: $\bar{A}.B$ y $A.\bar{B}$.

En el caso 3 analizado, dejando de lado la lógica mixta, al implantar la función OR se vio la necesidad de invertir las señales antes de ingresar a la compuerta NAND. Para el caso de la compuerta EXOR, el realizar la función OR no requiere de la inversión de las señales que ingresan a la compuerta NAND formada por las celdas "+". Se debe recordar que las funciones resultantes en las filas son: $\overline{\bar{A}.B}$ y $\overline{A.\bar{B}}$, es decir que implicitamente las señales de entradas a la compuerta NAND que ejecuta la función OR, están ya invertidas. Por lo tanto, para realizar la suma lógica de los dos términos AND, lo único que debe hacerse es colocar dos celdas "+" en una misma columna.

En resumen puede decirse que para realizar una función que está expresada como suma de productos lógicos, se deben crear primeramente los productos, con las variables entrando por las columnas, y con las señales resultantes en las filas se crean las sumas lógicas, el resultado final se dispone en las columnas. Enfocado de esta manera el diseño, los conceptos de lógica mixta ya no son relevantes, sino unicamente las ideas de como se forman las funciones AND y OR y muchos menos

se considera el realizar la obtención de los resultados de cada una de las compuertas NAND que se formaron, ni utilizar el Algebra de Boole.

En las Figs. 8.31 y 8.32 se muestran las representaciones lógica y a nivel de transistores utilizadas para la compuerta EXOR, en éstas se han suprimido los caminos no utilizados para dejar claras las interconexiones entre los diferentes elementos.

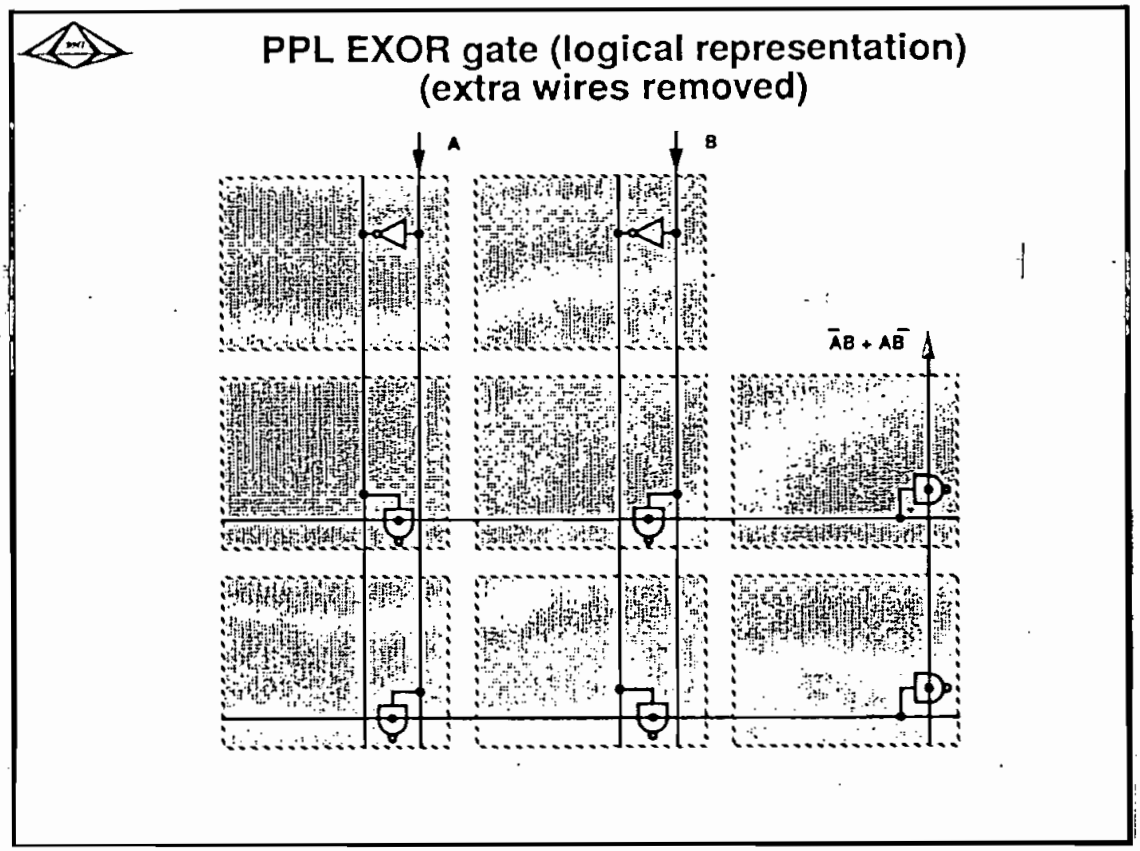


FIGURA 8.31 Representación lógica de la función EXOR .

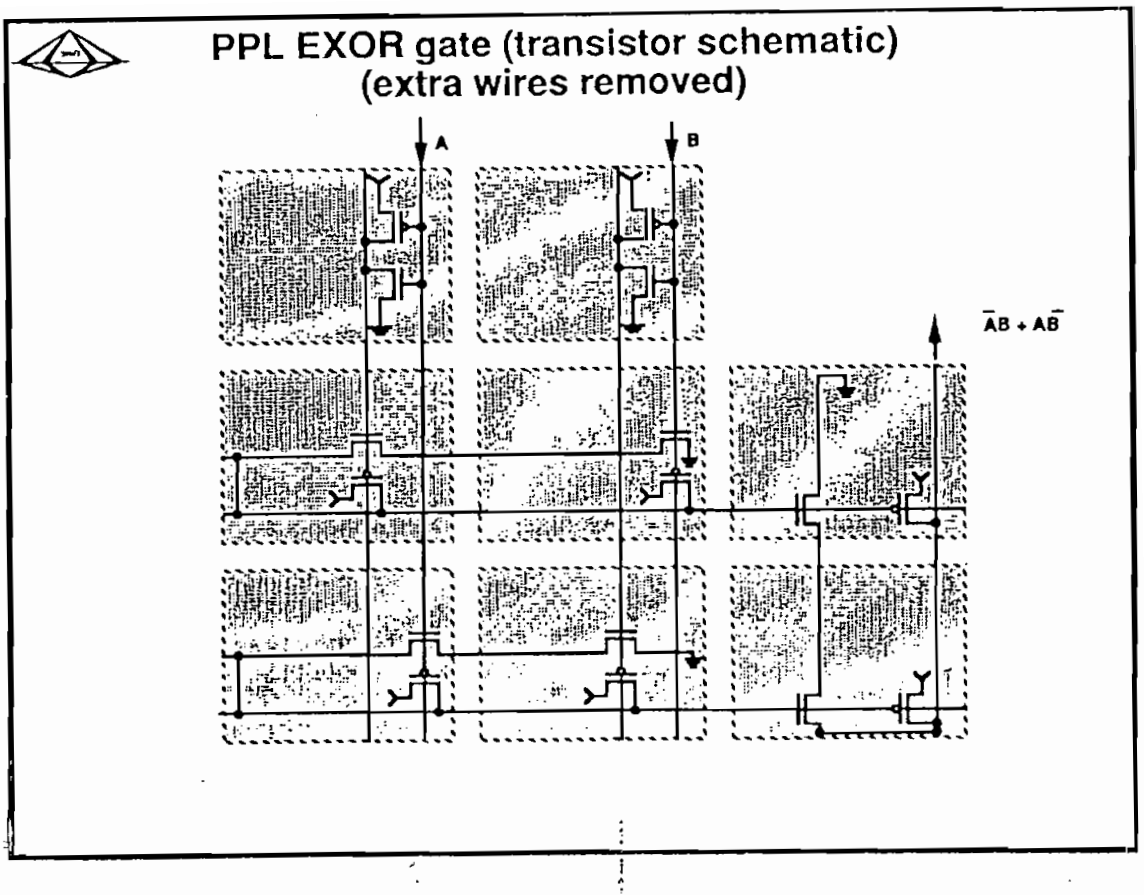


FIGURA 8.32 Representación a nivel de transistores de la función EXOR.

Algunas características propias del diseño PPL que son aplicables y de mucha utilidad en alguna aplicación específica, son por ejemplo:

a) En la Fig. 8.33 se presenta la misma compuerta EXOR con celdas "BLANK" (que permite el paso de los caminos de la grilla sin alterar a ninguno de ellos). Estas celdas añaden, en este caso, una fila y una columna al diseño original. Se presenta también el diagrama lógico del nuevo arreglo.

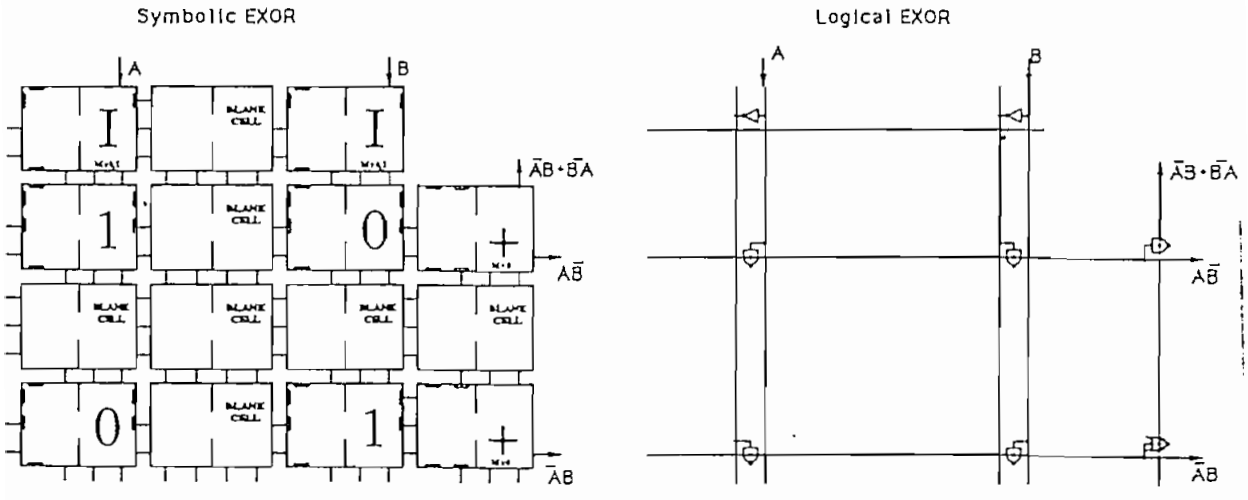


FIGURA 8.33 Compuerta EXOR con una fila y una columna añadidas.

La utilización de las celdas BLANK puede ser de mucha utilidad en la construcción de algunos circuitos, pues permiten obtener una adecuada distribución física y proveer los canales de enrutamiento que fuesen necesarios, en caso de ser insuficientes los ofrecidos por la ubicación de las celdas de función. Esta facilidad permite el utilizar el espacio disponible de una manera más eficiente y en ocasiones puede mejorar la densidad del circuito.

2. En PPL se tiene la facilidad de intercambiar filas y columnas completas y aún, a pesar del cambio introducido en la configuración, continuar obteniendo la misma función, esto gracias a la estructura del *layout* de las celdas.

Esta característica de intercambio de filas y columnas se pone de manifiesto en la Fig. 8.34 en la que se presentan cuatro posibles configuraciones para la compuerta EXOR. Se pueden intercambiar solo filas, solo columnas o filas y columnas a la vez. Una de las cuatro configuraciones puede ser, para determinada aplicación, la que ofrezca las mejores características, dependiendo de su forma y tamaño.

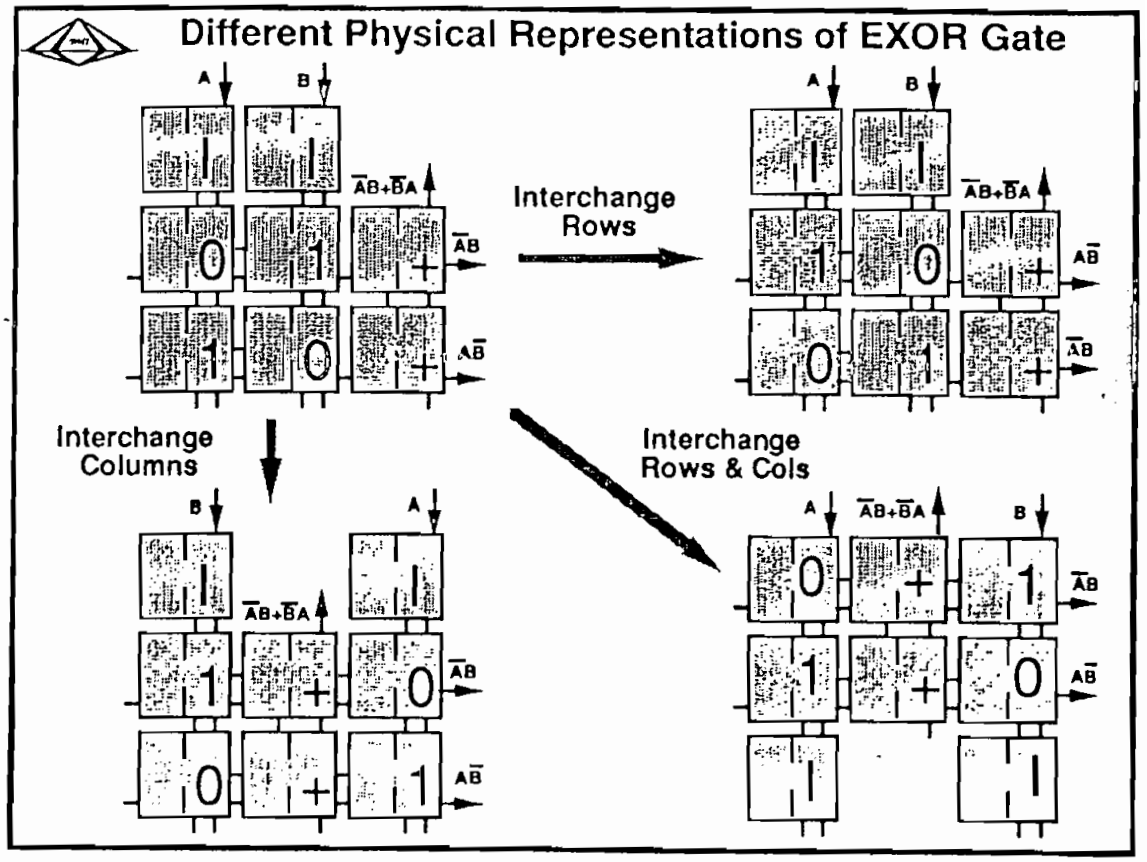


FIGURA 8.34 Versiones de la compuerta EXOR variando la ubicación de filas y columnas.

8.2.5 Celdas de interconexión

Son aquellas celdas que permiten realizar la interconexión de los caminos de la grilla PPL, tienen gran aplicación principalmente en el enrutamiento de señales entre módulos y hacia los *PADs*. Este tipo de celdas ocupan el espacio de una localidad unitaria y no pueden ni deben ser tratadas de manera similar a las interrupciones. Al trabajar con interrupciones, éstas se pueden representar dentro del área de una celda sin necesidad de ocupar una nueva localidad de la grilla. En resumen, no puede ubicarse una celda de interconexión en la misma localidad que una celda PPL, pero si pueden estar acompañadas de interrupciones.

En la Fig. 8.35 se presenta el conjunto de celdas de interconexión con sus posibles valores de modificadores. En las celdas "?" y "~" se pueden sumar los modificadores, con esto se tiene la opción de obtener cualquier combinación de conexiones que se desee. En las celdas, "#", "*", "$", "@", "3", no existe la posibilidad de sumar los valores de los modificadores, unicamente puede utilizarse uno de los valores predeterminados, de acuerdo a las necesidades del diseñador.

Por ejemplo, si se utiliza la celda "#" con un modificador M=1, los caminos RCOL y SROW se conectan. Si no se especifica el valor de un modificador, se tiene por predefinición un valor de M=0, que conecta los caminos RCOL y ROW.

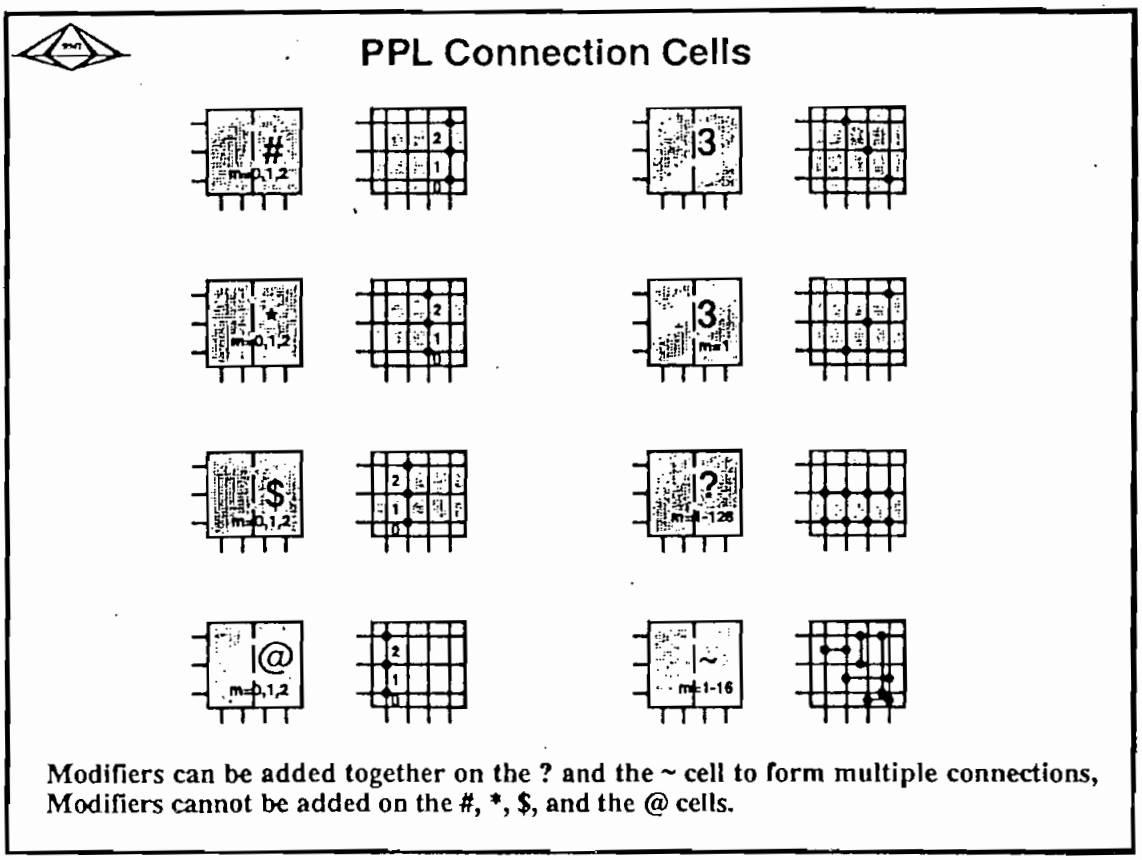


FIGURA 8.35 Celdas de interconexión.

Entre las celdas de interconexión, la celda "?" es la única que no permite conexiones en el camino TROW, pero si permite hacer las conexiones con los otros caminos. Esta restricción es causa directa del layout de la celda.

Las celdas de conexión incluyen también las celdas "g" y "^", no presentadas en la Fig. 8.35, que permiten la conexión de los caminos de la grilla a Gnd y V_{DD} respectivamente. Estas celdas permiten también la suma de sus modificadores.

Algunas de las celdas de conexión pueden parecer un poco extrañas, pero luego de cierta práctica con su manipulación, su utilidad es evidente. De la experiencia obtenida, puede señalarse que las celdas de interconexión más utilizadas son: "#", "*", "3" y "?".

a) Interconexión entre módulos

La Fig. 8.36 presenta un ejemplo sencillo, en el cual se desea enrutar la señal que ingresa a la grilla en la parte superior derecha (A) a la entrada de un inversor, y la salida de dicho inversor a la parte inferior de la grilla.

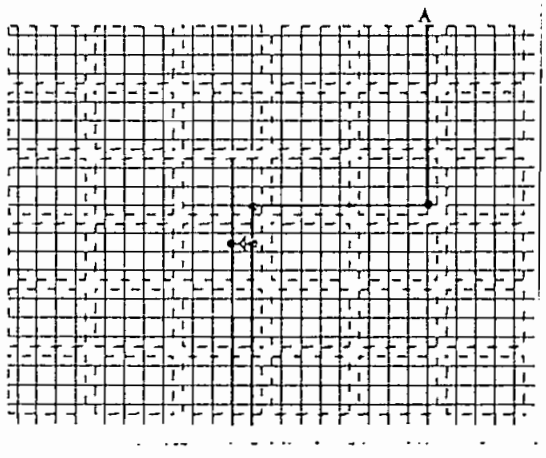


FIGURA 8.36 Esquema simple de interconexión

La representación simbólica del inversor y de las celdas de interconexión necesarias para realizar el enrutamiento solicitado, se presentan en la Fig. 8.37.

La celda de interconexión utilizada es la "#", que interconecta filas con columnas. Nótese también la aplicación de las interrupciones para bloquear el paso de la señal que está siendo enrutada hacia otros sectores de la grilla en los cuales no se la requiere.

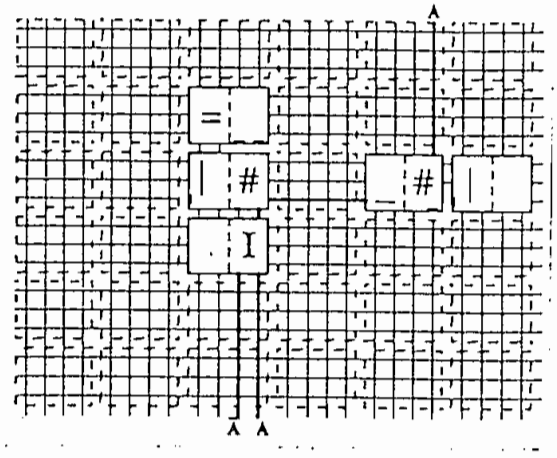


FIGURA 8.37 Esquema simbólico de interconexión simple

En la Fig. 8.38 se presenta un posible esquema de interconexiones necesarias para enrutar señales de entrada y/o salida alrededor de un módulo representativo PPL. La figura incluye la planificación realizada para lograr el enrutamiento de las señales, dicha planificación se debe ejecutar con las celdas de interconexión. En la Fig. 8.39 se presenta el módulo y la representación simbólica del conjunto de celdas de interconexión que cumplen con lo indicado en la Fig. 8.38. Se incluyen también las interrupciones necesarias para aislar las señales que se están enrutando, del resto de la grilla.

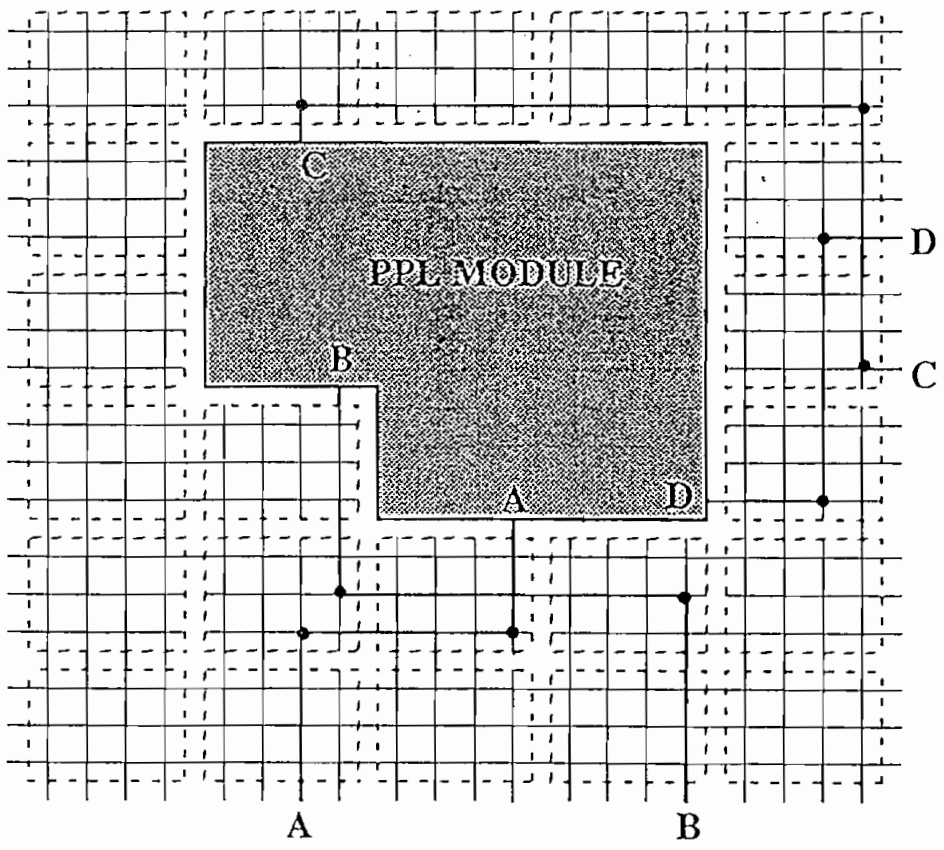


FIGURA 8.38 Interconexiones requeridas para enrutar las señales de entrada/salida de un módulo PPL.

Las celdas de conexión utilizadas son: "#", "?", "*".El ejemplo presenta también la manera de combinar celdas de interconexión con sus respectivos modificadores y las interrupciones, también con sus respectivos modificadores.

Es relevante la aplicación de la celda "?".

i) Con un modificador de M=18, se logra en la misma localidad de la grilla realizar dos interconexiones: RCOL con SROW y LCOL con ROW. Esto permite enrutar adecuadamente las señales A y B.

ii) En combinación con esta celda se utilizan interrupciones para las filas y columnas. Con M=1 como modificador de la interrupción, se permite a la señal A ingresar a la celda LCOL, e impide que la señal B pase por RCOL. (M=1). Se asume que los otros caminos no llevan señales ya que fueron interrumpidos dentro del módulo.

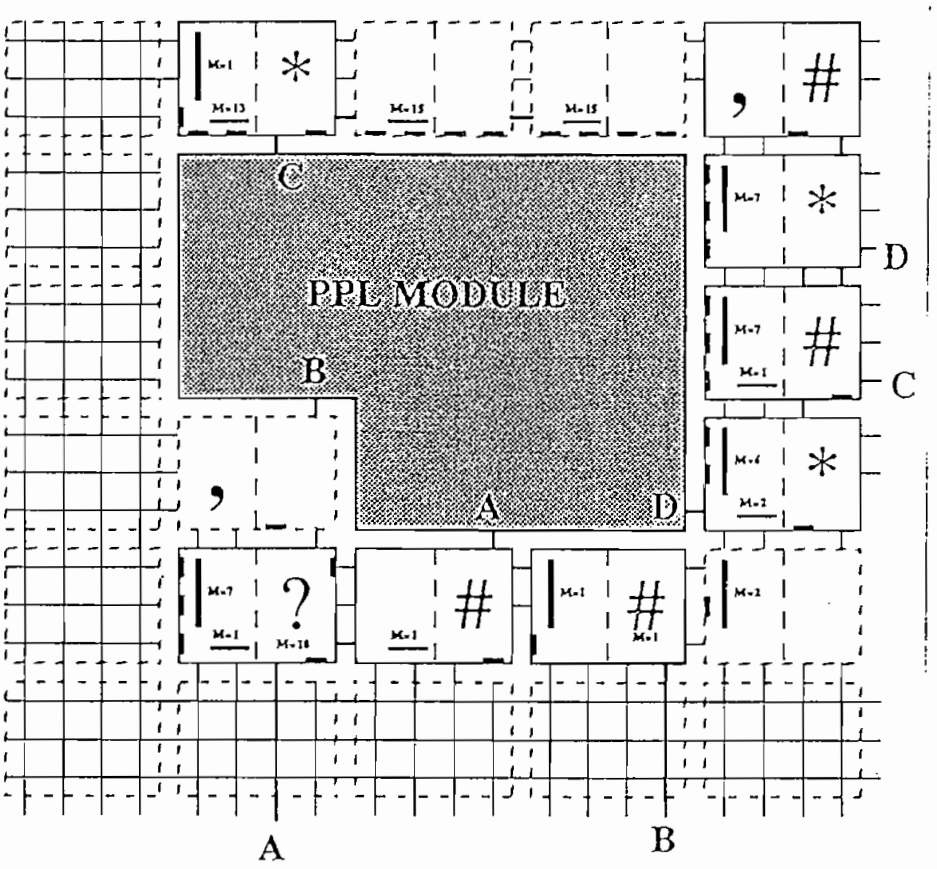


FIGURA 8.39 Esquema de las celdas de interconexiones requeridas para enrutar las señales de entrada/salida de un módulo PPL.

iii) La interrupción de las filas, evita que las señales A,B se enruten hacia la parte izquierda del arreglo, o que alguna señal extraña ingrese desde otro sector de la grilla produciendo conflicto con las señales A,B. Se utiliza un M=7, para bloquear todos los caminos de la fila.

De los dos ejemplos planteados se concluye que el diseñador al enrutar una señal debe preocuparse no solo de hacerlo hasta su destino final sino que también debe interrumpirla en cada lugar en el que posiblemente cause problemas con otras señales que corran por los mismos caminos.

Como conclusión a todo lo analizado hasta el momento, en especial gracias a la estructura física de las celdas, a su representación simbólica, al conjunto total de celdas disponibles y a que la metodología PPL permite implantar directamente cualquier función expresada en suma de productos sin preocuparnos de la estructura interna de las celdas, el diseño se transforma en un diseño simbólico antes que lógico.

8.3 MANIPULACION DE LAS HERRAMIENTAS PPL. CASO DE ESTUDIO SENCILLO: DISEÑO DE UNA COMPUERTA "EXOR"

El sistema de diseño PPL permite generar archivos para utilizarlos con otros paquetes de *software* totalmente independientes, pero que complementan y ayudan a la depuración de

los diseños realizados con PPL. Así, se puede obtener la información necesaria para trabajar con los simuladores SPICE e HILO, y con el programa de captura esquemática CAPFAST.

La versión de PPL disponible actualmente en la EPN, incluye el paquete CAPFAST y los utilitarios necesarios para generar las *netlists* a nivel de transistores para el simulador eléctrico SPICE.

Lastimosamente, las limitaciones de las versiones de SPICE disponibles en la EPN durante el desarrollo de este trabajo, no permitieron su uso con circuitos de mas de 100 transistores, que para diseño VLSI es un número demasiado pequeño. Sin embargo, con la certeza que la EPN dispondrá en poco tiempo de una versión actualizada del simulador SPICE, se describe la manera de obtener las *netlists* de los circuitos diseñados.

Los paquetes PPL y CAPFAST son sistemas completos de herramientas, cada uno dispone de su documentación respectiva, a la cual se hace referencia continuamente en caso que se deseen mayores detalles.

Durante la instalación del *software*, los paquetes deben ser configurados correctamente para que puedan interactuar.

8.3.1 Instalación del software

a) PPL

El hardware requerido para la instalación de PPL es mínimo, se requiere un computador AT compatible con: 640K de memoria, tarjeta de video EGA, disco duro, disquetera 5 ¼ de alta densidad.

La EPN adquirió el paquete PPL a la empresa norteamericana Bonneville Microelectronics, Inc. (BMI), encargada de su distribución. La limitación encontrada con este paquete es la protección de *sotware* que tiene incluída.

De los tres *diskettes* originales en los que se incluye todo el *software*, dos tienen protección. La protección se presenta en el número máximo de instalaciones permitidas, unicamente 2. Para controlar esto, los dikettes originales tiene un contador que se decrementa o incrementa cada vez que el paquete se instala o desinstala, respectivamente.

Un problema que retardó el inicio del trabajo con este paquete, se encontró al intentar instalarlo en una máquina con las características solicitadas. Más tarde, se encontró que el problema se debía al *drive* utilizado. Se procedió entonces a su reemplazo, consiguiendo instalar los programas. Debido a que la protección de *software* obliga a escribir sobre los *diskettes* para alterar el valor del contador, se

supone que la causa del problema con el *drive* original fue que para dicha escritura posiblemente se debe acceder a una región de los *diskettes* que el *drive* original no permitía.

El contenido de los 3 *diskettes* es el siguiente: el primero contiene los programas ejecutables, el segundo las librerías (bases de datos) necesarias y el tercero contiene el programa que permite la generación de archivos CIF. El *diskette* de programas permite 2 instalaciones, el segundo no tiene protección de ninguna clase y el tercero permite generar solamente 4 archivos CIF.

Los *diskettes* de programas y librerías poseen un programa de instalación. El de programas tiene un programa para desinstalación. El programa de conversión del tercer *diskette* puede ser ejecutado en el mismo *diskette*.

Por ejemplo, para instalar el *software* en el drive C, desde el drive A, para los dos *diskettes*, se ejecuta:

install C:

Además de crear los directorios necesarios y copiar los archivos respectivos en cada directorio, se crea un archivo con el nombre de *ppl.bat*. La ejecución de este archivo crea los *paths* y variables ambientales necesarias para la ejecución de los programas de PPL.

Para desinstalar el paquete, desde el drive A, con el *diskette* de programas se ejecuta:

recall C:

Mayores detalles de los procesos de instalación y desinstalación pueden encontrarse en los manuales de FPL.

Durante la instalación, como parte de la estrategia de protección de FPL, se crean dos archivos escondidos: EV31.SYS y EV32.SYS. Estos archivos no deben ser borrados ni alterados, cualquiera de las dos acciones impide la desinstalación; además, los programas ya no pueden pueden ejecutarse en el disco duro.

b) CAPFAST

La utilización de este paquete causó algunos inconvenientes:

i) La primera limitación que impone este paquete es la protección de *hardware* que posee, que debe ser colocada en el puerto paralelo del computador para que se ejecuten la mayoría de programas.

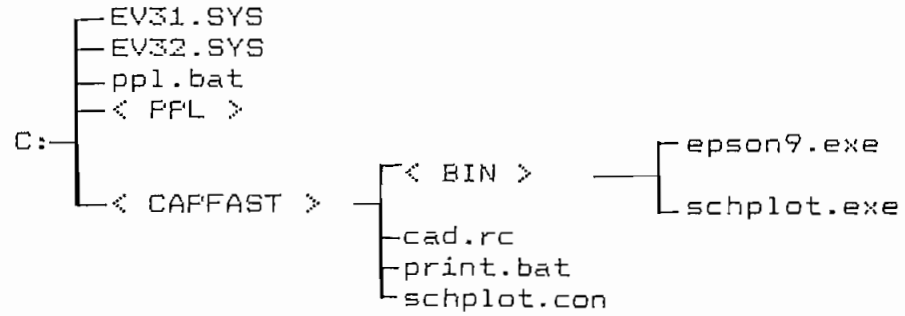
ii) Para circuitos de considerable número de transistores, que se traduce en un número también considerable de representaciones lógicas, se requiere instalar al menos 1M de memoria extendida.

iii) Al instalar en el disco duro, aún con las herramientas mínimas indispensables, requirió de al menos 4M del disco, y al menos 8M libre durante la instalación.

De CAPFAST se requieren solamente el editor de esquemáticos SCHEDIT y los diversos programas de impresión. La función de SCHEDIT se limita solamente a la presentación en la pantalla del esquema lógico proporcionado por PPL. No tiene sentido realizar ninguna edición ni corrección del diagrama lógico, lo único que se puede editar es el rotulado para la impresión. Con circuitos grandes, en la pantalla se presentan solo pequeñas secciones, lo que reduce aún más la utilidad de SCHEDIT.

Por las razones expuestas se decidió utilizar unicamente los programas de impresión. Las ventajas de esta decisión es que no se requiere de memoria extra y no se requiere de la protección de *hardware*.

Se presenta una posible estructura de los directorios y el nombre de los archivos contenidos en ellos:



Se crea un directorio con el nombre de CAPFAST, dentro de éste un nuevo directorio con el nombre BIN. El directorio BIN contiene el programa de impresión SCHPLOT y el driver para la impresora elegida. En este caso se ha elegido el driver para la impresora epson de 9 pines y compatibles.

Dentro del directorio CAPFAST, se deben crear los archivos *cad.rc* y *print.bat*. El archivo *cad.rc* no puede tener otro nombre, el archivo *print.bat* puede tener cualquier nombre. El contenido de los archivos es:

print.bat:

```
set ~p3=c:\ppl
set ~=c:\capfast
path %path%;c:\capfast\bin
PROMPT $p$g
```

cad.rc:

```
schplot -p.+~p3/sym
```

Para proceder a la impresión se debe correr siempre el archivo (.bat). Una vez generado el archivo (.sch) con las herramientas PPL, para la impresión se requiere una librería para la definición de los símbolos de cada celda, esta información se tiene en el directorio C:\PPL\SYM, lo que se le indica al programa SCHPLOT en el archivo *cad.rc*.

SCHPLOT siempre busca el archivo *schplot.con*, si no se encuentra se genera un mensaje de error, pero todo el proceso puede realizarse sin ningún problema. Este archivo contiene opciones especiales de impresión. Mayores detalles pueden encontrarse los capítulos 6 y 13 del manual de CAPFAST.

8.3.2 Herramientas PPL

El paquete de diseño PPL incluye un conjunto de 9 programas ejecutables, los archivos de descripción de las celdas (bases de datos) requeridas por los diferentes programas, archivos de texto que se utilizan para las ayudas en línea de algunos programas y un conjunto de ejemplos cuya descripción se encuentra en los manuales de PPL.

Las 9 herramientas de PPL que se dispone actualmente en la EPN son:

1. TILER
2. SIMPPLEX
3. SIMPPL
4. PPLPR
5. SPPLICE
6. PPLACE
7. EXPPLODE
8. PPLOW
9. PPL2CIF

Se debe indicar que cada uno de los programas tiene varias opciones que el usuario selecciona de acuerdo a sus necesidades. El detalle de los comandos que tienen algunos programas y la manera de invocar los programas con las diferentes opciones que estos ofrecen se explican en detalle en el manual de PPL.

La Fig. 8.40 presenta un esquema que indica la secuencia de utilización de los programas. Como se observa en la figura, PPL permite generar archivos para CAPFAST y SPICE.

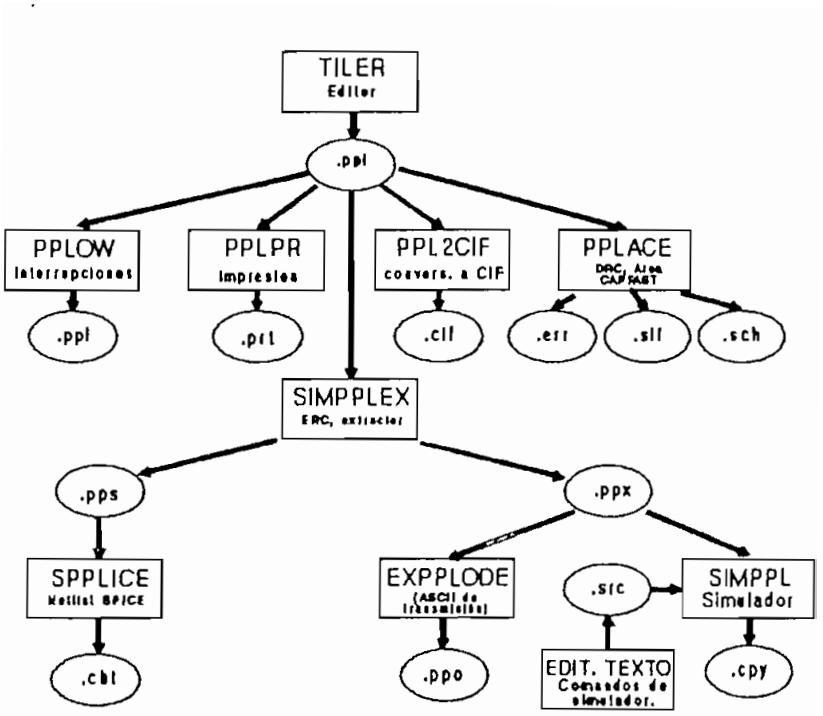


FIGURA 8.40 Herramientas de diseño PPL

Para demostrar la utilidad de todas las herramientas de PPL, se ha escogiodo el diseño de la la compuerta EXOR como caso de estudio sencillo. Las celdas necesarias y su disposición relativa fue desarrollada en el numeral 8.2, ahora se seguirá con este ejemplo todo el proceso de diseño hasta disponer del archivo CIF que se envía a las fundidoras.

Del conjunto de programas, tres constituyen las herramientas básicas del proceso de diseño, las demás prestan ayudas para obtener resultados del proceso de diseño y para generar archivos para CAPFAST y SPICE.

Las herramientas básicas de diseño PPL son: TILER, SIMPPLEX y SIMPPL. En la Fig. 8.41 se presenta la secuencia de utilización de estos tres programas.

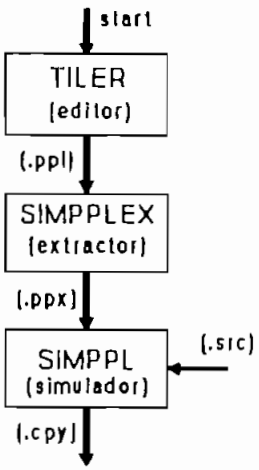


FIGURA 8.41 Herramientas básicas PPL.

A continuación se describe brevemente la interacción entre las herramientas básicas expuesta en la Fig. 8.41, para luego realizar una descripción más detallada de cada una.

A diferencia del paquete TENTOS, en donde el proceso de diseño demanda mucho trabajo a nivel físico, con FPL el diseño se limita unicamente a describir un circuito en base a los caracteres que representan las celdas, para lo cual se utiliza el editor TILER. Al salvar el circuito editado en TILER, se genera un archivo con extensión (.ppl).

Una vez descrito el circuito con TILER, SIMPPLEX utiliza el archivo (.ppl) para generar una descripción lógica, un *netlist* a nivel de compuertas e interconexiones entre las mismas, y entrega un archivo con extensión (.ppx) que será utilizado por el simulador lógico SIMPPL. Si SIMPPLEX no advierte la presencia de errores eléctricos graves, se pasa a la simulación, caso contrario los errores deben ser corregidos con TILER y se debe repetir el proceso.

El conjunto de señales de estímulo y puntos en los que se desea obtener los resultados de la simulación, deben ser generados por el usuario en cualquier editor de texto y almacenados en un archivo con extensión (.src). Los archivos (.src) y (.ppx) son utilizados por SIMPPL. Los resultados de la simulación se almacenan en un archivo con extensión (.cpy).

Si la simulación proporciona los resultados esperados, el diseño está concluido; caso contrario, se debe iniciar nuevamente el proceso de diseño, retornando a la edición con TILER para realizar las correcciones en el diseño lógico.

8.3.3 TILER

En el numeral 8.2 ya se mencionaron algunas de las características del editor TILER y la información de las celdas que puede obtenerse a partir de su línea de *status*. De manera resumida se presentan las características más importantes de TILER:

1. TILER es un editor interactivo especial, ideado para el diseño de CI. Se basa en la representación de las celdas por medio de caracteres (no es un editor gráfico).

2. TILER permite ubicar las celdas en la grilla PPL. Para esto, se debe posicionar el cursor en la localidad deseada e ingresar el caracter de la celda corespondiente.

3. TILER es el encargado de reservar las localidades de la grilla PPL necesarias, de acuerdo al caracter de la celda ingresado.

4. En la versión de PPL disponible actualmente, TILER puede manipular una grilla de máximo 79 filas X 36 columnas.

5. Información sobre las interrupciones y los modificadores de las celdas se presentan en la línea de *status*, en la parte inferior de la pantalla de TILER.

6. TILER ofrece mucha de la información del manual PPL respecto a sus comandos, modos de operación y celdas PPL.

a) **La línea de status**

En la Fig. 8.42 se presenta la línea de status de TILER, la información disponible en esta línea es:

TILER es el nombre del editor.

(SCMOS2OT) identifica la tecnología que ha sido cargada.

main:foo.ppl identifica el nombre del archivo sobre el cual se está trabajando.

B:111.1111 identifica el *status* de las interrupciones de los caminos en la ubicación actual del cursor. Las tres primeras barras verticales ("1"), a la izquierda del punto (".") representan los caminos de las filas (TROW, SROW y ROW), y las cuatro a la derecha del punto, repre-

sentan los caminos de las columnas (TCOL, SCOL, LCOL y RCOL). Una interrupción se representa con una línea horizontal (-); por ejemplo, si los caminos de RCOL y TROW son interrumpidos, las interrupciones se representan como __11.111_;

M:0(0) indica el valor del modificador de la celda en forma decimal y entre paréntesis en forma hexadecimal.

R2:C2 da la ubicación del cursor: (FILA:COLUMNA).

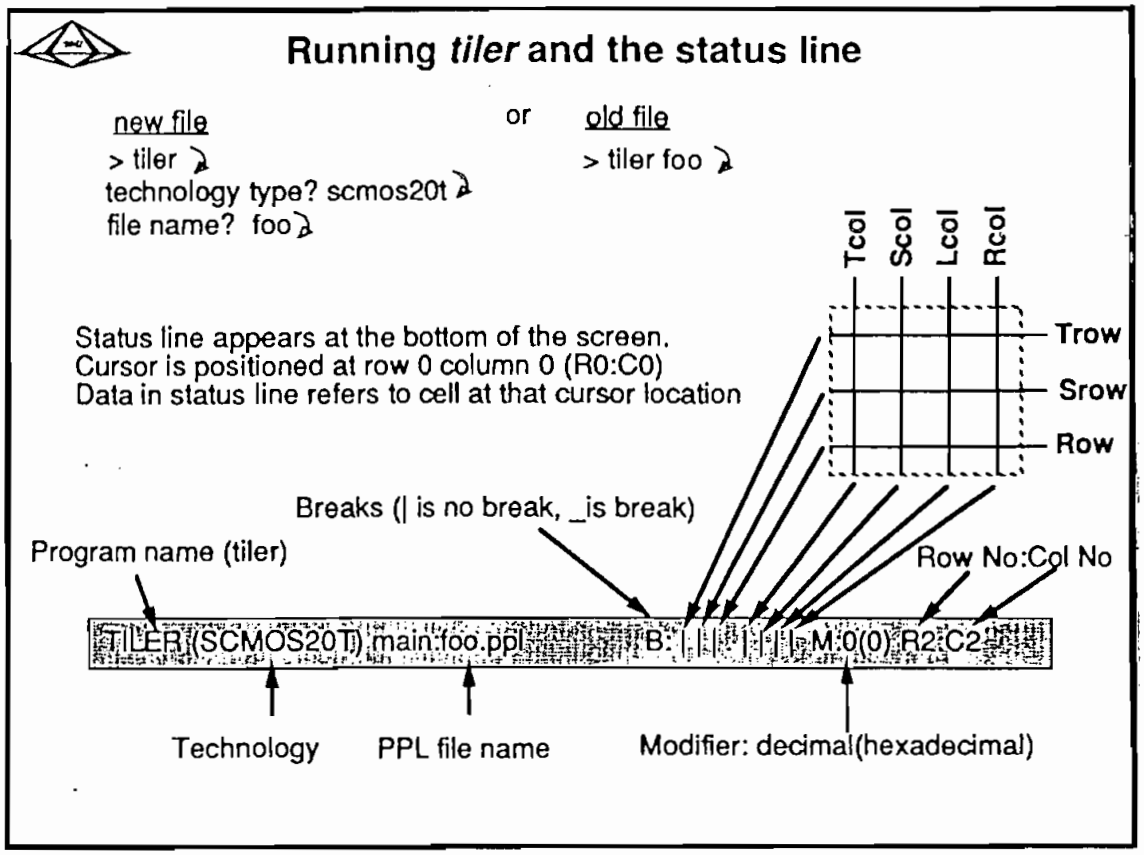


FIGURA 8.42 Ejecución de TILER y su línea de *status*

Para ejecutar el programa basta con ingresar:

tiler nombre

En donde "nombre" es el nombre asignado al archivo que se va a crear. Cuando se inicia el trabajo sobre un nuevo archivo el programa solicita la tecnología a utilizarce.

Una vez ingresado el nombre de la tecnología se carga el archivo de tecnología respectivo. La información contenida en este archivo es referente al proceso utilizado para definir las celdas PPL (nMOS, CMOS, GaAs, etc), el número de caminos por fila y columna de la grilla PPL, el espaciamiento entre celdas en la grilla e información de cada celda utilizada en la tecnología especificada.

Cuando se trabaja sobre un archivo con un diseño creado anteriormente, al invocar a TILER, acompañado del nombre del archivo de trabajo, se carga también automaticamente el archivo de tecnología con que se creo.

La tecnología CMOS que se dispone actualmente en la EPN es la de 2.0 µm, para un proceso de fabricación de doble pozo (*twin tub*) y con dos niveles de metal. Las celdas están desarrolladas en base al conjunto de reglas de diseño especificado por MOSIS (MOSIS es una fundidora de gran reconocimiento, trabaja con programas de investigación y universidades en los EEUU). El nombre de esta tecnología se ingresa como: SCMOS2OT.

Una vez concluida la sesión de trabajo, se salva el diseño realizado y se abandona el programa. Por predefinición, al archivo creado se le asigna una extensión (.ppl), que puede ser alterada por el usuario.

b) Comandos de TILER

TILER presenta un extenso conjunto de comandos, por lo que solamente se exponen ideas generales de los tipos de comandos y algunos ejemplos.

TILER presenta inicialmente su pantalla de trabajo unicamente con la línea de *status* y el resto de la pantalla completamente limpia. Cuando se ingresa un comando, éste se observa en la parte inferior izquierda de la pantalla, justo bajo la línea de *status*.

La mayoría de los comandos de TILER se ejecutan con la ayuda de **caracteres de control.** Los caracteres de control generalmente son caracteres no imprimibles y se presentan en la pantalla como un caracter precedido por el símbolo "^". Así por ejemplo, son caracteres de control: ^A, ^B, ^C, etc.

Para ingresar un caracter de control se presiona la tecla "CONTROL" y luego la tecla del caracter correspondiente. Muchos de los comandos de TILER son simples caracteres de control, en otros comandos los caracteres de control sirven como prefijos de otros caracteres.

Los caracteres de control utilizados como prefijos por TILER son: <esc> (tecla "*escape*"), ^X y ^Z. Por ejemplo: ^XC, ^XH, ^ZB, ^ZL, etc.

La mayoría de los comandos tienen efecto inmediato y su resultado se observa directamente en la pantalla. Si se ingresa un comando de manera equivocada o cuando se ingresa un caracter al que no se le ha asignado una celda PPL, TILER presenta una indicación de error.

Cuando se ingresa un prefijo, este se presenta en la pantalla, bajo la línea de status, y de ser necesario, el cursor permanece junto al prefijo esperando por los caracteres restantes del comando. Cuando se ingresan los comandos las letras mayúsculas y minúsculas desencadenan la misma acción, así por ejemplo ^a y ^A son comandos equivalentes.

i) Comandos con caracteres de control

Estos comandos son los que se ingresan como simples caracteres de control. Así por ejemplo: ^A, ^B, ^N, ^V.

ii) Comandos extendidos

Los comandos eXtendidos utilizan el prefijo ^X, que se presenta en la pantalla como ^X. Muchos de estos comandos se usan para leer y escribir archivos u otras operaciones que requieran información adicional del usuario.

A continuación del prefijo pueden estar caracteres imprimibles ordinarios (letras, números u otros caracteres imprimibles) o caracteres de control. Ejemplos de este tipo de comandos son: ^XA, ^XI, ^XN, ^X=, ^X(, ^X^A, ^X^C.

iii) "Meta" comandos

Los comandos que utilizan el prefijo <esc> se denominan meta comandos (*meta commands*). Existen dos clases de meta comandos:

1. **Ordinarios**, que se forman presionando la tecla <esc>, seguida de cualquier caracter imprimible ordinario. Estos comandos representan su prefijo en la pantalla como "M-". Ejemplos de este tipo de comandos son: M-D, M-C, M-A, M-U, M-<, M-?.

2. **Meta comandos de control** (*meta-control commands*), que se forman presionando <esc> seguido por un **caracter de control**. Este tipo de comandos también pueden ejecutarse utilizando el caracter de control ^Z como prefijo, seguido por un caracter ordinario. Estos comandos representan su prefijo en la pantalla como "M-C-" ó como "^Z". Ejemplos de este tipo de comandos son: ^ZC, ^ZD, ^ZZ, ^Z?.

Si un prefijo se ingresa por error o si es necesario no ejecutar un comando parcialmente ingresado, el comando puede abortarse con "^G". Si un comando ha iniciado su ejecución, nada puede hacerse para detenerlo.

Para introducir el manejo de los comandos de uso más frecuente, se expone una secuencia detallada del proceso para definir con TILER el circuito de la compuerta EXOR.

c) La compuerta EXOR

i) Para ingresar al programa se escribe *tiler exor*, esto ejecuta TILER y asigna al archivo con el nombre "*exor.ppl*". A continuación el programa solicita la tecnología a utilizar, se debe ingresar *SCMOS2OT*. El programa inicia su operación con el cursor en la esquina inferior izquierda de la grilla, en la posición (R0:C0).

ii) Las celdas necesarias para construir la compuerta EXOR y la posición relativa de las mismas, se presentaron en la Fig. 8.30. Las celdas que se utilizaron para el diseño son: "O", "1", "I" y "+".

Para ingresar las celdas se debe posicionar el cursor en la localidad deseada y escribir el caracter de la celda. Para el posicionamiento del cursor se utilizan las teclas de las flechas, aunque alternativamente existe un conjunto de comandos para el mismo objetivo.

El módulo EXOR en la grilla PPL, debe quedar en la pantalla de TILER similar a la ilustración de la Fig. 8.43. Las celdas se ingresan desde la localidad (R0:C0).

FIGURA 8.43 Módulo EXOR en la pantalla de TILER

En la Fig. 8.44 se esquematiza una pequeña sección de la grilla PPL, suficiente para el módulo EXOR. En esta grilla se ha numerado las filas y columnas y servirá como referencia para la explicación.

R3				
R2	I	I		
R1	1	0	+	
R0	0	1	+	
	C0	C1	C2	C3

FIGURA 8.44 Módulo EXOR en la grilla PPL

Como primer paso, se posiciona el cursor en (R2:C0) y se ingresa una celda "I". El cursor queda en la posición (R2:C1), en donde se requiere otra celda "I", por lo cual se ingresa nuevamente "I".

Las demás celdas se ubican de la siguiente manera: en (R1:C0) una celda "1", en (R1:C1) una celda "0", en (R1:C2) una celda "+", en (R0:C0) una celda "0", en (R0:C1) una celda "1", en (R0:C2) una celda "+".

iii) Para la simulación se deben especificar los nodos del circuito en los cuales se aplicarán señales de estímulo y los nodos en los cuales se desea obtener los resultados de la simulación. Los nombres de las señales (etiquetas) de entrada y salida de la compuerta EXOR (a, b, c) se deben asignar a los caminos de la grilla que correspondan. Para esto se ubica el cursor en (R2:C0) y se ejecuta el comando ^X *N*, luego se debe especificar el camino y el nombre, en este caso *rcol a*. Esto asigna al camino RCOL el nombre "a"

Para los dos nombres restantes se ubica el cursor en (R2:C1) y se ingresa el comando ^X *N* y luego *rcol b* y en (R2:C2) se ingresa el comando ^X *N* y luego *rcol c*.

Los pasos 2 y 3 pueden intercambiarse. Para observar la lista de los nombres asignados a los diferentes caminos se utiliza el comando ^X *L*.

iv) Para definir el módulo se han utilizado 3 filas y 3 columnas de la grilla. Al iniciar TILER, éste maneja una grilla de 22 filas y 36 columnas, que con los comandos adecuados pueden extenderse a 79 filas y 36 columnas.

En este punto del diseño es innecesario utilizar toda la grilla disponible.

Para retirar la grilla no utilizada y definir claramente la extensión del módulo EXOR se ubica el cursor en (R2:C2) y ejecuta el comando: ^X }. En respuesta TILER pregunta si se desea eliminar toda la grilla sobre y bajo la posición del cursor (*Destroy all ppl above/right of cursor?*); se responde afimativamente con *y* de "*yes*".

v) Una vez que se ha definido completamente el módulo se debe grabar todo lo realizado. Para grabar un archivo se utiliza el comando: ^X ^S. El archivo generado tiene el nombre **exor.ppl**.

vi) Para abandonar el programa se utiliza el comando: ^X ^C

d) **Operaciones comunes con TILER.**

i) **Celdas con modificadores**

Para ingresar celdas con modificadores se utiliza la siguiente secuencia:

1. <esc>
2. valor numérico del modificador
3. nombre de la celda

Si se desea acompañar de un modificador una celda cuyo nombre sea uno de los siguientes caracteres: (0, 1,.., 9, -), se procede de la siguiente manera:

1. <esc>
2. valor numérico del modificador
3. ^Q
4. nombre de la celda

ii) Especificación de interrupciones.

Cuando se desea colocar interrupciones, de acuerdo a las necesidades y a lo explicado en el numeral 8.2, se procede de la siguiente manera:

1. Ubicar el cursor en la localidad deseada
2. <esc>
3. valor numérico
4. "¦" si se interrumpen caminos de las filas
   "-" si se interrumpen caminos de las columnas

En la pantalla la representación de las interrupciones no es muy clara, se pueden identificar claramente aquellas que representan a las interrupciones de los caminos ROW, LCOL y RCOL, denominados primarios; así como las diferentes combinaciones entre ellos. Los carateres utilizados por TILER para las diferentes combinaciones se presentan en la Tabla 8.1.

Símbolo	Significado
'	Interrumpido ROW
,	Interrumpido LCOL
.	Interrumpido RCOL
=	Interrumpido LCOL y RCOL
!	Interrumpido ROW y LCOL
%	Interrumpido ROW y RCOL
&	Interrumpido ROW, LCOL, RCOL

TABLA 8.1 Caracteres utilizados por TILER para indicar interrupciones de los caminos primarios

La información sobre las interrupciones de los otros caminos (SROW, TROW, SCOL, TCOL), se encuentran indicadas en la línea de *status*, junto con las interrupciones de los caminos primarios, pero para la ubicación actual del cursor. Para solucionar este problema se tiene un modo especial de operación de TILER, que se explica más adelante.

iii) Borrado de celdas e interrupciones

Con el comando ^*D* se borra la celda y las interrupciones asociadas a la posición actual del cursor. Igual efecto tiene utilizar la tecla <*delete*>.

Con el comando ^*Z D* se borra una celda pero no las interrupciones. Con la tecla <*backspace*> se borran solamente las interrupciones de una localidad, si está presente una celda no es borrada.

iv) Inserción de filas y columnas

Cuando sea necesario incrementar el número de filas y/o columnas se procede de la siguiente manera:

1. Ubicar el cursor en el límite de la grilla.
2. <esc>
3. Número de filas o columnas que se desea incrementar
4. ^X^I
5. dirección hacia donde se extiende la grilla, el número especificado en 3. Las direcciones posibles son cuatro:

   ^P hacia arriba (*previous*)

   ^N hacia abajo (*next*)

   ^B hacia la izquierda (*back*)

   ^F hacia la derecha (*forward*)

El especificar un número a un comando para que se ejecute (como en 3.) se denomina "paso de un argumento". Muchas veces la acción de un comando se modifica ligeramente con el paso de un argumento o realiza una acción el número de veces especificado.

Cuando no se desea especificar el argumento, pero se desea la acción modificada del comando, se utiliza el caracter de control ^U como argumento. Un comando de esta naturaleza es por ejemplo: ^U ^X ^C.

v) Manipulación de archivos

Para insertar un módulo dentro de la grilla PPL, se ubica el cursor en la posición deseada y se utiliza el comando: ^X I, como respuesta a este comando TILER pregunta el nombre del archivo que contiene el módulo a insertar. El usuario debe proporcionar esta información.

Para salvar un circuito modificado, sin alterar el archivo original que lo contiene, se lo debe grabar con otro nombre. El comando a utilizarse es: ^X ^W, TILER pregunta entonces el nuevo nombre, el cual debe ser ingresado por el usuario.

Para iniciar la edición de un nuevo circuito o de uno al que se le desea modificar, sin tener que abandonar TILER y utilizando la misma tecnología, se utiliza el comando ^X ^R. TILER ofrece la oportunidad de grabar el circuito actual antes de iniciar la nueva sesión. El usuario debe especificar el archivo a cargarse o el nombre del nuevo archivo.

Para observar el contenido de un archivo de texto en la pantalla, sin tener que salir de TILER, se utiliza el comando ^XF. El usuario debe especificar el nombre del archivo.

Para salvar periodicamente el trabajo que se va realizando sobre un archivo, se utiliza el comando ^X ^S.

vi) Edición de nombre de los caminos

Como se indicó anteriormente para obtener una lista de los nombres asignados (etiquetas) a los caminos de la grilla PPL, elegidos por el diseñador, se utiliza el comando *^X L*. Para remover una etiqueta se utiliza el comando *^X R*. TILER pregunta el nombre de la etiqueta a borrarse de la lista.

vii) Interacción directa entre TILER, SIMPPLEX Y SIMPPL

Si al terminar una sesión de TILER se abandona el programa con el comando *^U ^X ^C*, SIMPPLEX se ejecuta directamente. Si SIMPPLEX no encuentra errores fatales, automaticamente se carga SIMPPL, quedando listo para el ingreso de comandos. Si al final de la sesión de simulación se requiere regresar a TILER, se lo pude hacer sin necesidad de abandonar SIMPPL. Para ello se ejecuta el comando *TILER* (de SIMPPL).

viii) Ayuda en línea

Se puede obtener ayuda e información sobre las celdas, sobre los comandos y algunas características avanzadas de TILER.

Para acceder a la ayuda se utiliza el comando: "*^X ?*". En respuesta a este comando TILER presenta el siguiente mensaje:

"DOC (? for help):"

Si se ingresa "?", se despliega una lista de las diferentes categorías de ayuda disponibles. Si se conoce la categoría de ayuda que se requiere, se ingresa simplemente el caracter asociado a cada categoría.

Con los comandos "*M-* ?" y "*M-* /" se puede acceder a la descripción de cualquier celda o comando especificados.

Dentro de las opciones de TILER, se tiene también la posibilidad de que usuarios con cierta experiencia en el manejo de este programa puedan definir "macros" y asignar, si se desea, cada "macro" a una tecla determinada elegida por el usuario.

e) Modos de *display* de TILER

TILER permite visualizar los circuitos editados en tres modos diferentes de presentación en la pantalla. Estos modos son: *NORMAL*, *ZOOM_IN* y *ZOOM-OUT*. Los modos difieren entre sí por el formato utilizado para presentar las celdas en el arreglo PPL. En la línea de *status* se indica si se está trabajando en un modo diferente del normal.

Para intercambiarse entre los modos NORMAL y ZOOM-IN se utiliza el comando *<esc> Z*. Para pasar al modo ZOOM-OUT se ejecuta el comando *^U <esc> Z*. Para retornar al modo normal desde ZOOM-OUT solamente se ejecuta el comando *<esc> Z*.

i) Modo NORMAL

El modo NORMAL, es el que se ha utilizado hasta el momento, permite visualizar el caracter de la celda junto con algunas de las interrupciones. El modo NORMAL es el modo de operación presente por predefinición y es adecuado cuando no se está editando secciones del circuito que utilicen complejos enrutamientos de las señales.

ii) Modo ZOOM-OUT

El modo ZOOM-OUT presenta unicamente los caracteres de las celdas, sin ninguna información de las interrupciones, por lo que permite apreciar grandes secciones de un circuito extenso.

iii) Modo ZOOM-IN

El modo ZOOM-IN permite visualizar los caracteres de las celdas, sus modificadores y a todos los caminos de la grilla y las interrupciones, sean éstas propias de la celda o impuestas por el diseñador.

En la Fig. 8. 45 se presenta una celda "s", con un modificador M=1 y con los caminos SROW y TROW interrumpidos. En el lado derecho de la figura puede observarse la representación de la celda en modo ZOOM-IN.

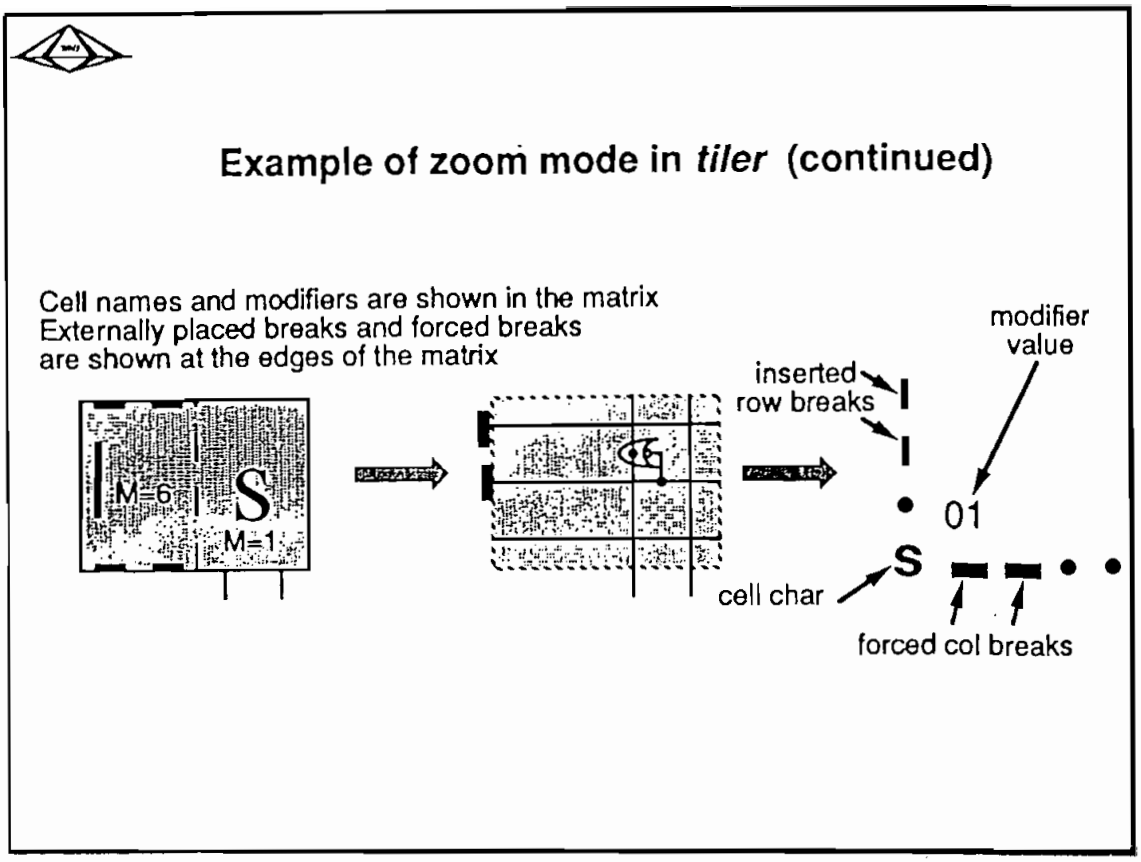


FIGURA 8.45 Modo ZOOM-IN de edición para una celda "s"

ZOOM-IN utiliza para cada celda unitaria una matriz de 5 columnas por 4 filas. En la esquina inferior izquierda se ubica el nombre de la celda, muy cercano a éste se ubica el valor del modificador (en hexadecimal). Cada uno de los tres caminos horizontales y cuatro verticales de una localidad PPL, está representado en la parte izquierda e inferior de la matriz. Un punto en la matriz indica que el camino correspondiente no está interrumpido, mientras que una línea vertical u horizontal indica que los caminos asociados están interrumpidos, ya sea por que lo determina el *layout* de la celda o por que el usuario lo ha determinado.

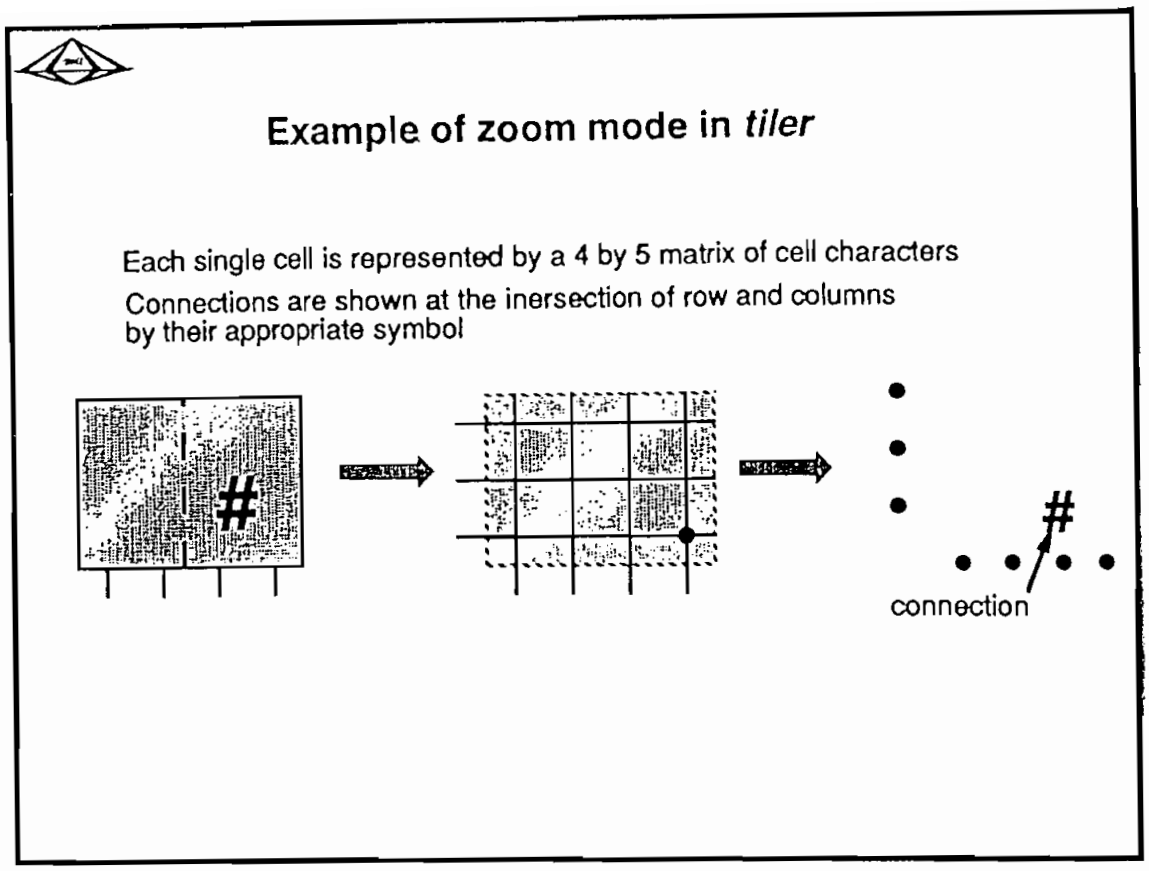


FIGURA 8.46 Modo ZOOM-IN de edición para una celda de interconexión

Para el caso de las celdas de interconexión, el modo ZOOM-IN presenta el caracter correspondiente al nombre de la celda en cada intersección en que se conecte un camino horizontal con uno vertical, por lo que los modificadores de estas celdas ya no son necesarios y no se presentan.

Este modo de trabajo es particularmente útil cuando se desea realizar enrutamientos complejos, en donde se requiere permitir el paso de las señales hacia los lugares en donde sean necesarias e interrumpirlas en donde no lo son. En la Fig. 8.46 se presenta la representación para la celda "#".

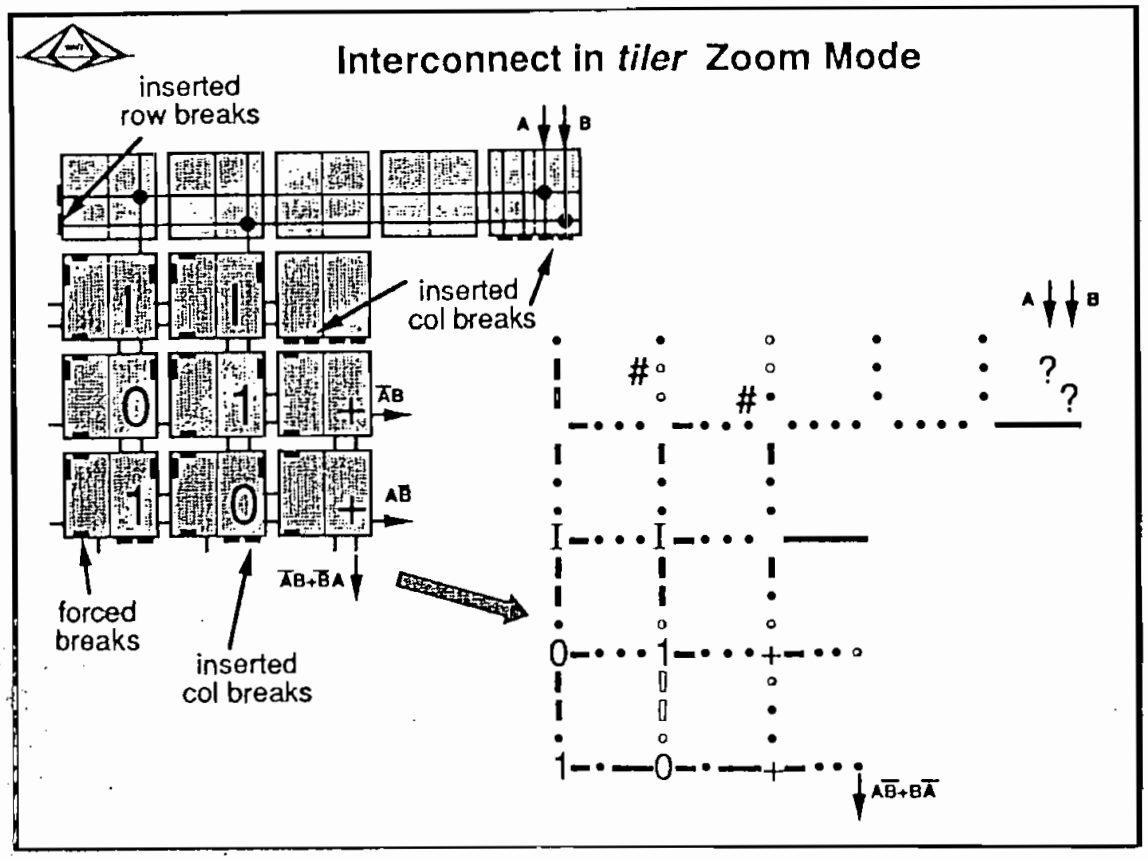


FIGURA 8.47 Modo ZOOM-IN de edición para el módulo de la compuerta EXOR

En la Fig. 8.47 se presenta la compuerta EXOR diseñada, y la estructura que se observaría en la pantalla en el modo ZOOM-IN. Al diseño realizado se le ha añadido tres celdas de interconexión para demostrar la facilidad con que puede seguirse el paso de una señal. La celda "?" presenta el caso en el que se utilizan dos símbolos en los puntos en donde se unen los caminos verticales con los horizontales.

8.3.4 SIMPPLEX

Como se indicó anteriormente este programa "extrae" un *netlist* que contiene elementos lógicos convencionales (compuertas NAND, NOR, flip/flops, etc) y sus interconexiones. Esta tarea se basa principalmente en determinar cuales celdas se agrupan para formar compuertas. Este programa puede ser invocado con diferentes opciones, la función de cada opción puede encontrarse en el manual del paquete PPL.

SIMPPLEX cumple también otras tareas muy importantes:

a) La primera es realizar un chequeo inicial de los archivos (.src) que se utilizan con el simulador. Esto se realiza si el usuario ha preparado el archivo (.src) antes de correr SIMPPLEX. Si se encuentran errores SIMPPLEX los presenta durante su ejecución, indicando el nombre del error y la ubicación del mismo dentro del archivo. Posibles errores encontrados pueden ser un comando mal especificado o que los nombres de los nodos utilizados en el archivo (.src) no correspondan con los utilizados al editar el circuito con TILER. En el manual del paquete PPL se puede encontrar una lista con los errores que pueden cometerse y su posible origen.

b) SIMPPLEX realiza un chequeo de reglas eléctricas y genera los mensajes correspondientes cuando se detectan errores. Existen cuatro categorías de errores:

i) *FATAL*, errores severos o fatales que el simulador no puede ignorar y deben ser corregidos antes de simular el circuito. Cuando se encuentran estos errores SIMPPLEX entrega el mensaje: "*Fatal to SIMPPL*", si no existen errores fatales pero existen errores de las otras categorías, se indica que la simulación puede realizarce ("*simulation allowed*"). Un ejemplo de este tipo de errores es haber conectado una salida a Gnd.

ii) *ERROR*, son errores que pueden ser ignorados por el simulador, pero que deben ser corregidos tarde o temprano, pues posiblemente harán que el circuito falle si se los pasa por alto. Un ejemplo típico es el no haber conectado las señales de entrada a los PADs correspondientes, error que se corrije realizando las conexiones necesarias, pero puede hacerse después de simular el circuito.

iii) *WARNING*, corresponden a problemas menores que pueden necesitar corrección, pero que no hacen fallar al circuito. Por ejemplo, se genera un error de este tipo si se utiliza un nombre para un camino SROW o SCOL que es parte de un *stack*.

iv) *NITPICK*, son errores debido a problemas menores que generalmente no causan problemas. Por ejemplo, una salida desconectada genera este tipo de error.

v) *TIMING*, estos errores de "temporización" tienen la misma categoría que los *ERROR* y proporcionan mensajes de advertencia para aquellos nodos que tienen cargas capacitivas muy grandes o tiempos subida/bajada muy grandes. Sin embargo, la base de datos utilizada por SIMPPLEX no contiene toda la información necesaria para generar información exacta de la temporización y estos mensajes pueden ser irrelevantes. En los casos que se considere la posibilidad que estos mensajes tengan importancia debe procederse a realizar la simulación eléctrica con el programa SPICE.

En el manual del paquete PPL se puede encontrar una lista de errores de cada tipo y una descripción detallada de las posibles causas de los mismos.

Para ejecutar el programa y generar el archivo necesario para el simulador, se debe escribir simplemente ***simpplex***, acompañado por el nombre del archivo. No es necesario especificar la extensión del archivo, que se asume es (.ppl). Si el archivo no tiene la extensión (.ppl), entonces debe especificarse la extensión utilizada.

Haciendo uso de las opciones de SIMPPLEX, que se detallan en el manual del paquete PPL, se pueden generar también archivos que son parte del proceso de obtención de *netlist* para el simulador SPICE.

Para el caso de la compuerta EXOR que se está diseñando, se invoca a SIMPPLEX de la siguiente manera:

simpplex exor

El resultado obtenido en la pantalla se muestra en la Fig. 8.48. Como se observa en la figura, se presenta una lista de las acciones que ejecuta el programa, una evaluación del número de errores de algunas de las categorías y un resumen de la estadística de los dispositivos utilizados en el circuito.

```
SIMPPLEX Version 4.8. Copyright 1986-91. Bonneville Microelectronics Inc.
Loading SCMOS20T.SDB database...
Loading circuit from file exor.ppl...
Finding circuit context...
Tracing circuits connections...
Creating extract file exor.ppx...

FATAL ERRORS:    0 (simulation allowed)
ERRORS:          2 (2 ignored)
WARNINGS:        0

Device statistics
PPL rows:        3
PPL columns:     3
Transistors:     16
Internal nodes:  9
Input pointers:  2/8
Output pointers: 2/8
Total pads:      0
```

FIGURA 8.48 Pantalla obtenida al ejecutar SIMPPLEX

El resultado de SIMPPLEX indica que no hay errores fatales, ni *WARNINGS*, pero si dos del tipo *ERROR*. Estos dos errores se deben a que las dos señales de entrada (a,b) no están conectadas a ningún PAD y son solo nodos sueltos.

La interconexión de las señales, tanto de entrada como de salida, hacia los PADs se realiza más adelante y en este momento los dos errores pueden ser ignorados. Por lo tanto, se permite realizar la simulación.

La estadística revela que el circuito:

a) Utiliza 3 filas y 3 columnas.

b) Utilizan 16 transistores, que se justifican por las 8 celdas básicas empleadas, cada una formada por dos transistores, lo que da un total de 16.

c) Tiene 9 nodos internos (7 en la compuerta EXOR y 2 correspondientes a V_{dd} y Gnd).

d) Los denominados "*Input pointers*" indican el peor caso de "*fan in*", para el ejemplo 2, y el número total de entradas a **compuertas**, para el ejemplo 8.

e) Los "*Output pointers*" indican el peor caso de "*fan out*", para el ejemplo 2, y el número total de salidas de **compuertas**, para el ejemplo 8.

Un método para conocer el número de transistores que utiliza cada celda, es crear con TILER un archivo que contenga solamente la celda seleccionada, se corre SIMPPLEX y de los resultados se toma el número de transistores.

8.3.5 SIMPPL

SIMPPL es un simulador lógico de una unidad de retardo (*unit delay simulator*) y asume que cada transición en el circuito requiere una sola unidad de retardo.

La simulación puede realizarse de manera interactiva con SIMPPL o en modo de "*batch*", para lo cual se crea el archivo (.src), que es lo más práctico. Este archivo (.src) se crea con ayuda de cualquier editor de texto y contiene una serie de comandos utilizados por SIMPPL.

SIMPPL ofrece un conjunto extenso de comandos. Información sobre los comandos se obtiene en la ayuda que ofrece SIMPPL. Para una lista de comandos se ejecuta el comando HELP, para información sobre un comando se ingresa HELP acompañado del nombre del comando que se requiere.

A continuación se exponen los comandos incluídos en el archivo (.src) que se utilizará para simular la compuerta EXOR:

```
1.   copy
2.   vector in a b
3.   watch  in c
4.   set in=00
5.   cycle
6.   set in=01
7.   cycle
8.   set in=10
9.   cycle
10.  set in=11
11.  cycle
```

El archivo debe crearse de manera idéntica a la presentada, sin inculir los números, que se utilizan solo para referencia en esta explicación.

En la primera línea, el comando *copy*, instruye a SIMPPL para que almacene los resultados de la simulación en un archivo que se denominará: *exor.cpy*. Si no se incluye este comando los resultados de la simulación se presentan solo en la pantalla y muchas veces por la rapidez con que se despliegan, es imposible analizar los resultados.

En la segunda línea, el comando *vector* define un vector con el nombre elegido por el usuario, en este caso "*in*". Para el ejemplo presentado, el vector está compuesto por las dos entradas (a,b).

En la tercera línea, el comando *watch* define la lista de nodos en los que se desea saber los cambios producidos por las señales de estímulo. Para el ejemplo presentado se incluye en esta línea el vector "*in*" y el nodo "c". Esto instruye al simulador a entregar los valores lógicos de los nodos (a,b,c).

La línea 4 en realidad es el inicio de las pruebas. Con el comando *set* se define que el vector "*in*" tenga el valor "00"; es decir, la señal "a=0" y la señal "b=0". Se deben definir los valores en el mismo orden que se definieron las componentes del vector.

La línea 5, con el comando "*cycle*" permite la propagación de las señales de entrada definidas en la línea 4 hasta la salida.

Las líneas 6, 8 y 10 setean las entradas a "01", "10" y "11" respectivamente, con lo que se simula el circuito para todas las posibilidades de combinaciones de las entradas. Las líneas 7, 9 y 11 propagan las señales de entrada hacia la salida.

Para documentar adecuadamente el archivo (.src), se inicia una línea con el caracter ";". Esto le indica al simulador que no debe ejecutar lo que está a continuación y que se trata solamente de un comentario.

Para ejecutar el programa se digita: *simppl exor.*

En la Fig. 8.49 se presenta los mensajes que despliega SIMPPL. Al final SIMPPL está listo a recibir los comandos para la simulación.

```
Registered to: University STUDENTS
(C) Copyright 1990 Bonneville Micro.
All rights reserved
SIMPPL Version 4.3 Copyright 1986-91 Bonneville Microelectronics Inc.
Loading network from file exor.ppx...
Finding initial circuit state...
Ready for commands. For assistance, type HELP
sim>
```

FIGURA 8.49 Pantalla obtenida al ejecutar SIMPPL

Para ejecutar los comandos contenidos en el archivo *exor.src*, se debe ingresar el comando: *source*. El simulador asume que el archivo (.src) tiene el mismo nombre del archivo con que se invocó a SIMPPL, caso contrario se debe especificar claramente el nombre y/o extensión del archivo de comandos.

Los resultados de la simulación se despliegan en la pantalla y se graban en el archivo (.cpy). Para abandonar SIMPPL, se debe ingresar el comando: *quit*, lo que permite retornar el control del computador al sistema operativo.

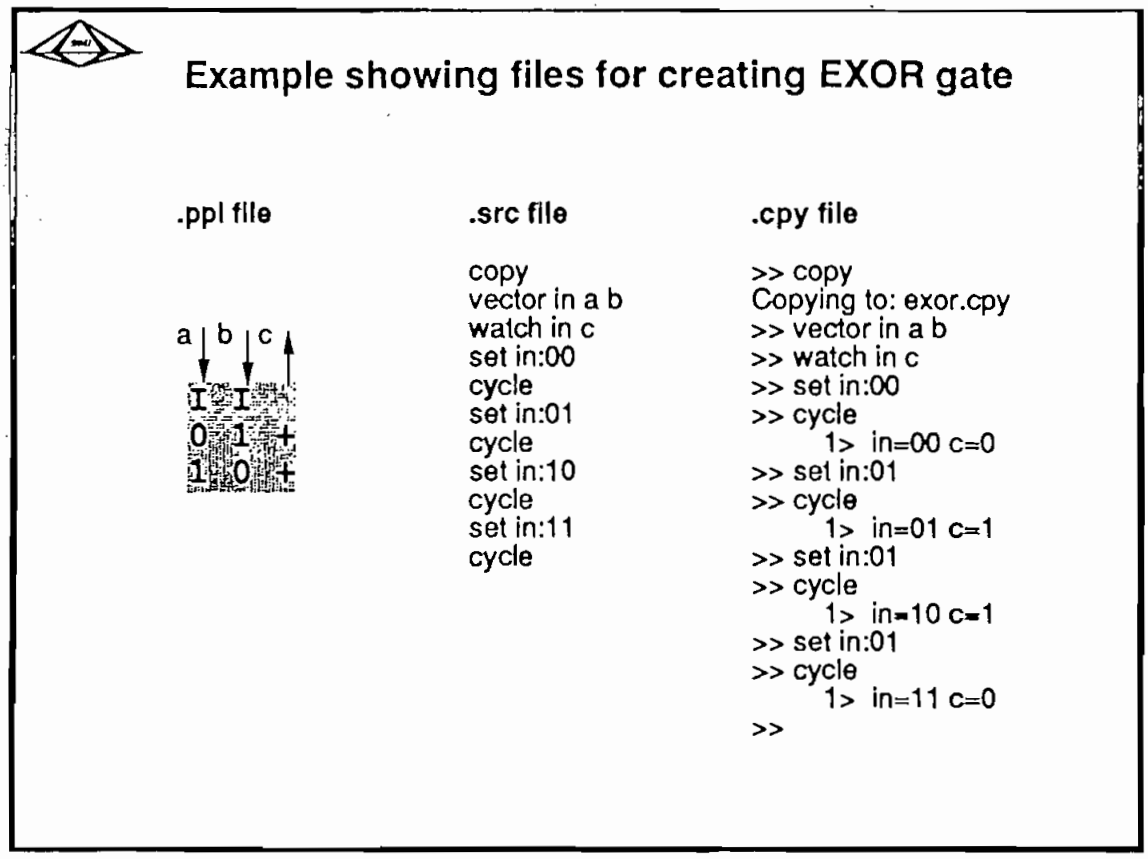


FIGURA 8.50 Resultados de la simulación de la compuerta EXOR.

En la Fig. 8.50 se presentan el arreglo de las celdas, el archivo (.src) y el resultado de la simulación que estará contenido en el archivo *exor.cpy*. Según lo especificado se entregan como resultado los valores del vector "*in*" y la señal de salida "c". Los resultados de la simulación son satisfactorios ya que el circuito simulado cumple con la función EXOR.

Como se explica con mayor detalle en el numeral 8.3.5 y 8.3.6, para completar el diseño se deben realizar las conexiones del circuito diseñado a los PADs, generando un nuevo archivo (.ppl) y repetir nuevamente el proceso seguido hasta el momento. Una vez hecho esto, se procede a la conversión del nuevo archivo (.ppl) al formato CIF.

A continuación se describen brevemente los programas utilitarios del paquete PPL.

8.3.6 Programas utilitarios

a) PPLPR

Este programa utiliza archivos (.ppl) y crea archivos ASCII con una extensión (.prt). que puede ser utilizado en cualquier impresora convencional o archivos *post script* para impresoras laser. El archivo generado describe el diseño realizado.

Siguiendo con el ejemplo de la compuerta EXOR, para crear un archivo *exor.prt* para una impresora convencional se debe ejecutar PPLPR de la siguiente manera: ***pplpr exor.***

En la Fig. 8.51 se presenta el contenido del archivo (.prt) generado, éste presenta las celdas utilizadas con numeración para filas y columnas. Por último, el archivo incluye una descripción de los nodos que fueron nombrados con TILER; por ejemplo, la línea que describe el nodo "a" tiene el siguiente formato:

a [2,0,0] = nombre [fila, columna, camino]

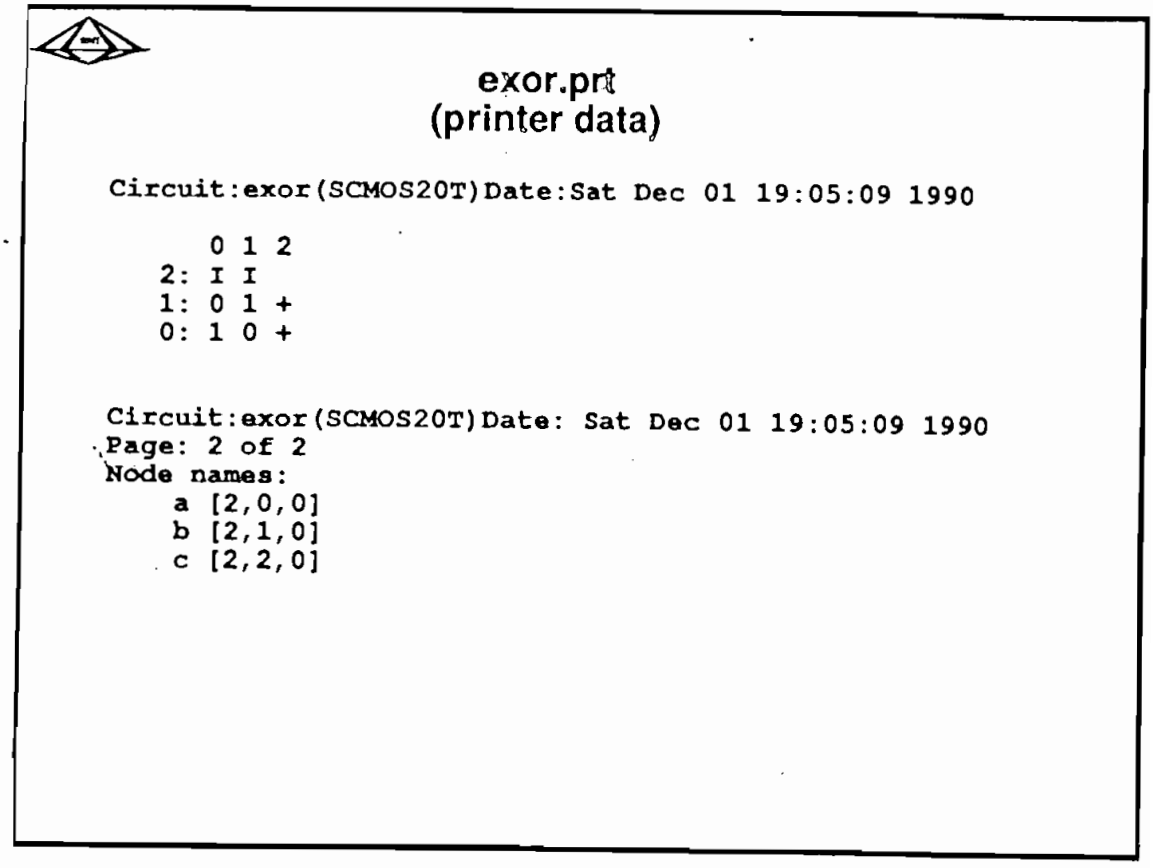


FIGURA 8.51 Contenido del archivo *exor.prt* generado

La línea presentada significa que el nombre "a" está ubicado en la fila 2, columna 0 (R2:C0) y está asignado al camino RCOL. El número asignado a cada camino es:

0	rcol
1	lcol
2	scol
3	tcol
4	row
5	srow
6	trow

El programa permite también generar archivos sin la numeración de filas y columnas, eliminar la lista de los nombres de los nodos, imprimir en formato para modo ZOOM-IN, ZOOM-OUT, y otras opciones adicionales que pueden encontrarse detalladas en el manual del paquete PPL.

Para obtener un archivo (.prt) con el formato ZOOM-IN se invoca a PPLPR de la siguiente manera:

pplpr -Z

El contenido del archivo (.prt) generado con formato ZOOM-IN, se presenta en la Fig. 8.52.

b) SPPLICE

Con las herramientas PPL se pueden generar *netlists* para el simulador SPICE. Por ejemplo, para la compuerta EXOR se utiliza el programa SIMPPLEX con el siguiente formato:

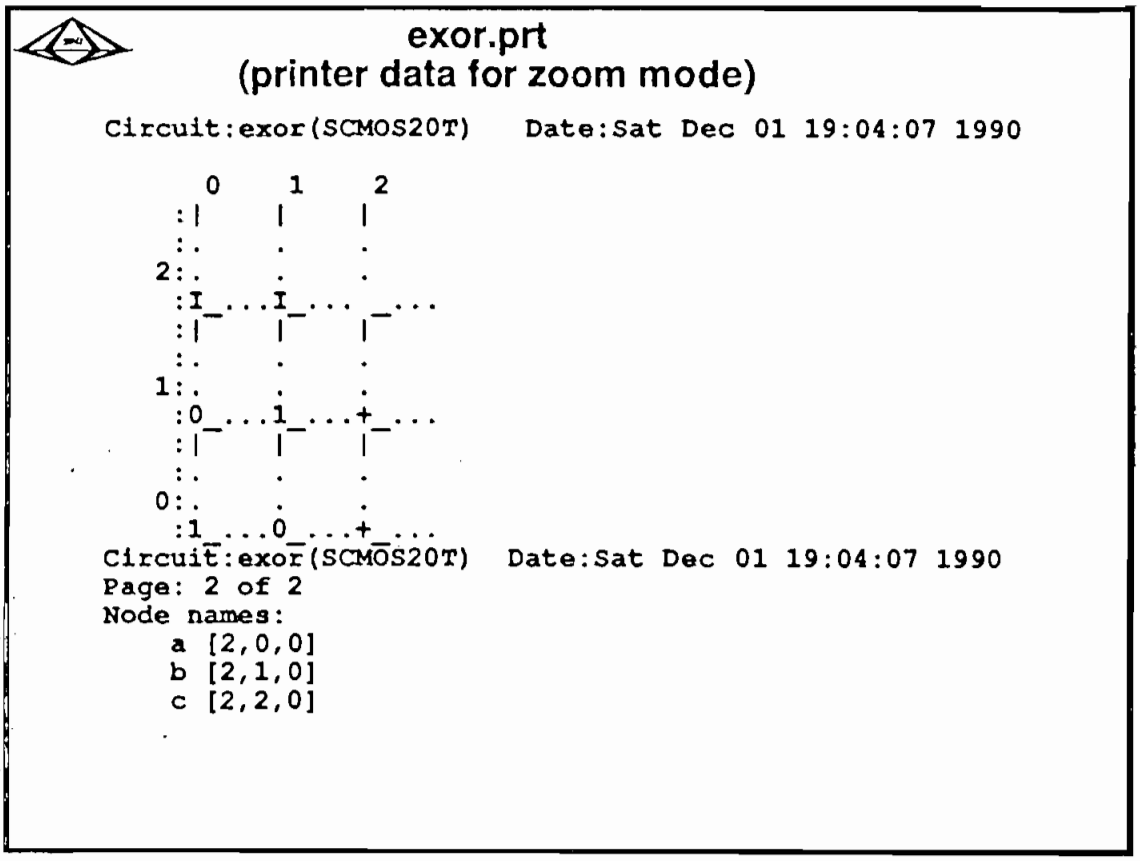


FIGURA 8.52 Archivo exor.prt generado con formato ZOOM-IN

simpplex -sx.ckt exor

Con las opciones con que se invoca a SIMMPLEX, se obtiene un archivo con extensión (.pps), para el presente caso: *exor.pps*.

Finalmente, para obtener el *netlist* SPICE se utiliza el programa SPPLICE, que utiliza el archivo (.pps) generado anteriormente:

spplice exor

La ejecución de SPPLICE genera el archivo *exor.ckt*, que define la interconexión de los transistores utilizados y el valor de las capacitancias asociadas a cada uno de los nodos del circuito. Los valores de capacitancias presentados corresponden al *layout* del diseño y no son solamente estimaciones, sino cálculos bastante exactos.

En la Fig. 8.53 se presenta el archivo *exor.ckt* listo para la simulación SPICE. Las líneas enmarcadas deben ser incluídas por el usuario. Los modelos para los transistores "n" y "p" se obtuvieron de los manuales del paquete PPL.

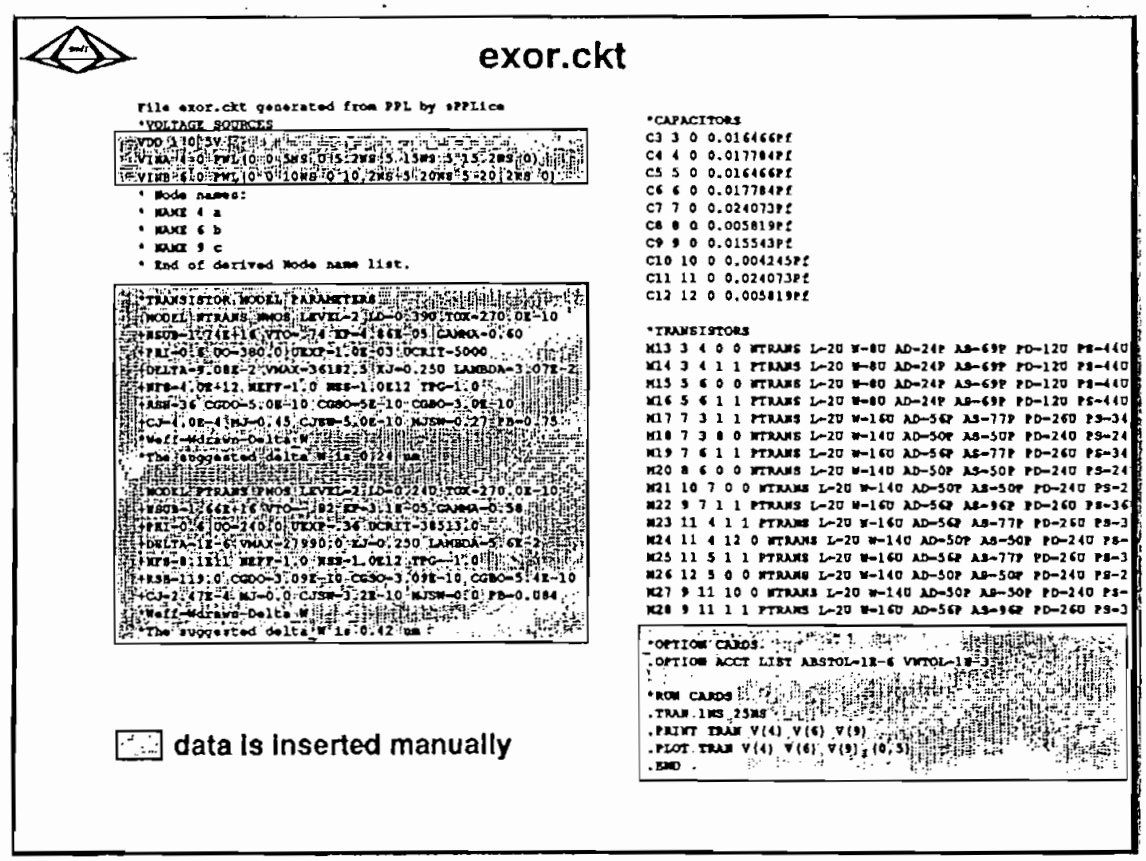


FIGURA 8.53 Archivo *exor.ckt* con los comandos necesarios adicionados para el simulador SPICE

b) PPLACE

Este programa permite generar archivos (.sch) a partir de los archivos (.ppl). Un archivo (.sch) contiene la descripción del circuito para obtener un esquemático con CAPFAST. Para el caso de la compuerta EXOR se genera el archivo (.sch) invocando a PPLACE de la siguiente manera:

pplace -3 exor

En la Fig. 8. 54 se presenta el esquema impreso para el caso de la compuerta EXOR. Como puede verse, esta representación a nivel lógico define símbolos para cada celda.

Para realizar la impresión del archivo (.sch) se debe generar primeramente un archivo intermedio utilizando el programa SCHPLOT. Este archivo intermedio es una descripción del esquemático que es luego utilizado por el driver de la impresora. El formato para la ejecución de SCHPLOT es:

schplot [opciones] nombre.sch > nombre.gen

De las opciones se resumen las más importantes:

i) -c número de columnas y -r número de filas.

Se especifica en cuantas partes se divide el esquema a ser impreso.

Luego de impresas, las partes pueden pegarse para obtener el esquema total. Estas opciones son útiles especialmente en circuitos grandes en los cuales el imprimir en una sola hoja no proporciona ningún detalle.

ii) -overlap porcentaje

Cuando se imprime en varias partes, se especifica un argumento entero que hace que cada hoja impresa tenga un sobrelapamiento de la siguiente para poder unirlas de mejor manera.

iii) -outline

Cuando se imprime en varias partes, cada hoja impresa tendrá una línea entrecortada por la cual se cortan las hojas y unirlas de mejor manera.

Una vez generado el archivo intermedio, se procede a la impresión, para esto se usa el driver de la impresora. Para la explicación se ha escogido el EPSON9, que tiene el siguiente formato:

epson 9 [opciones] nombre.gen

Las principales opciones de este programa se refieren a la calidad de la impresión, más especificamente a la densidad de los puntos impresos.

La densidad de puntos debe especificarse para el sentido horizontal y vertical, mientras mayor, de mejor calidad es el resultado. El formato es:

-x puntos por pulgada

-y puntos por pulgada

Para el driver epson9, se dispone de los valores:

En x: 60, 72 (predefinición), 80, 90, 120, 240.

En y: 72 (predefinición), 216.

Por ejemplo, si se desea especificar que el esquema exor.sch sea impreso en 9 partes, con la línea de puntos alrededor de cada una, se ejecuta:

schplot -c3 -r3 -outline exor.sch > exor.gen

Si se desea la impresión de la más alta calidad, se ejecuta "*epson9*" de la siguiente manera

epson9 -x240 -y216 exor.gen

Mayores detalles de los programas de impresión se encuentran en el manual de CAPFAST, Cap. 6.(*Plotting Schematic Hardcopies*).

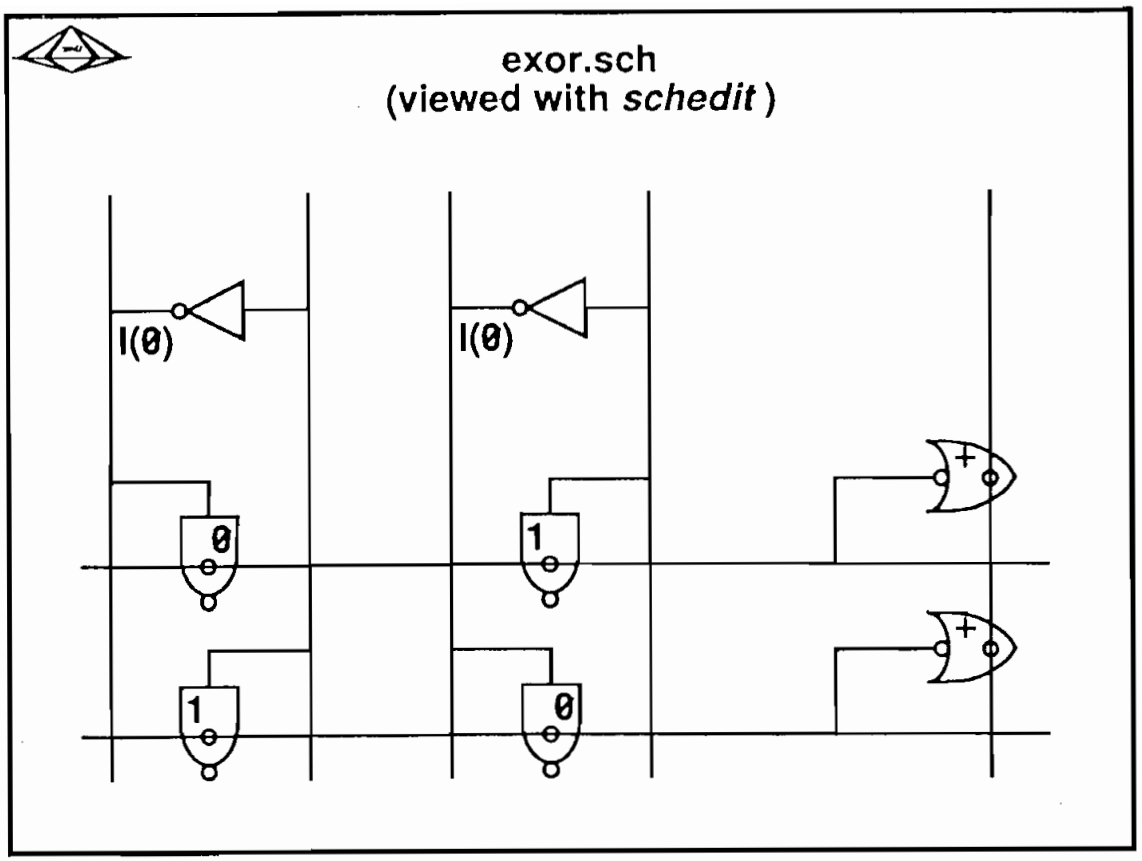


FIGURA 8.54 Archivo *exor.sch* impreso con SCHPLOT

c) EXPPLODE

PPL fue concebido en la Universidad de Utah, en ella se suelen tener computadores pequeños conectados a un computador central para compartir recursos. El convertir el archivo (.ppx) a formato ASCII permite enviar los archivos de los diseños para la simulación. Esta es la función del programa EXPPLODE, generar archivos ASCII. Los archivos generados tienen una extensión (.ppo). Sin embargo, se debe mencionar que la descripción del circuito contenida en este archivo para el desarrollo del presente trabajo no aportó con nada.

8.3.7 Inserción de un módulo en el *PAD-FRAME*

En el Cap. 3, numeral 3.4.1, se expuso las ideas generales sobre lo que es un *PAD-FRAME*, y se dijo que simplemente es una estructura que tiene ubicaciones predeterminadas para los *PADs* y en cuyo interior se debe introducir el diseño realizado.

PPL ha diseñado dos *PAD-FRAMEs*, que se incluyen en el software disponible, y que son aceptados por la fundidora MOSIS. El disponer de esta estructura simplifica mucho el trabajo de diseño y se demuestra el ahorro sustancial de tiempo y esfuerzo si se comparan con los invertidos en el paquete TENTOS.

En la Fig. 8.55 se presenta el esquema del *PAD-FRAME* y las interconexiones desde los *PADs* del núcleo hacia los "*bonding PADs*". La estructura de la figura es la requerida por MOSIS. El empaquetado que entrega MOSIS para estos diseños es del tipo DIP de 40 pines. En los cuadros que representan los "*bonding PADs*", se incluye el número de pin al cual se conecta cada "*bonding PAD*". En la parte izquierda de la Fig. 8.55 se presenta las coordenadas en las que deben estar ubicados los *PADs*.

Las dimensiones de la cavidad de este tipo de proyectos es de 2220 X 2250 µm, y a los CIs se les denomina "*tiny chips*", por su reducido tamaño.

MOSIS 2 MICRON (40pin) TINY CHIP FRAME
SUBMITTED PROJECT SIZE = 2220 BY 2250 MICRONS

PINi	X	Y
1	2170	1325
2	2170	1525
4	2170	1925
5	2170	2200
6	1910	2200
7	1710	2200
8	1510	2200
9	1310	2200
10	1110	2200
11	910	2200
12	710	2200
13	510	2200
14	310	2200
15	50	2200
16	50	1925
17	50	1725
18	50	1525
19	50	1325
20	50	1125
21	50	925
22	50	725
23	50	525
24	50	325
25	50	50
26	310	50
27	510	50
28	710	50
29	910	50
30	1110	50
31	1310	50
32	1510	50
33	1710	50
34	1910	50
35	2170	50
36	2170	325
37	2170	525
38	2170	725
39	2170	925
40	2170	1125

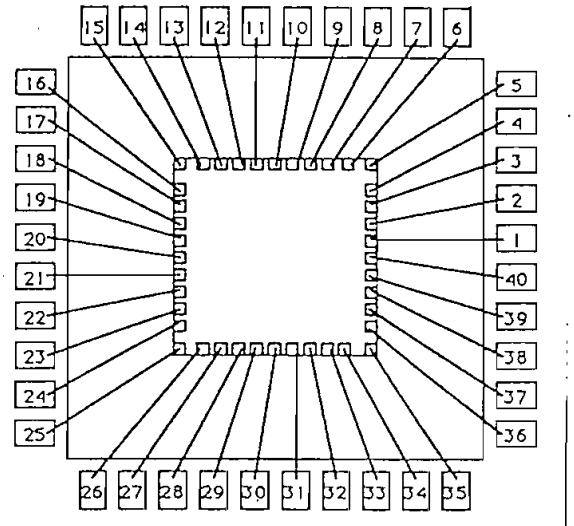


FIGURA 8.55 *Tiny chip* de MOSIS

Los *PAD-FRAMEs* creados por PPL para diseñar "*tiny chips*" están contenidos en dos archivos nombrados como: tiny36.ppl y tiny36.ppl. Estas estructuras se han construído como cualquier otro módulo PPL de extensión (.ppl), que puede editarse con TILER. Estos archivos consideran ya la ubicación permitida para los *PADs*.

En la Fig. 8.56 se presenta el *PAD-FRAME* "tiny34". Como puede verse, inicialmente las localidades destinadas a los *PADs* están utilizadas por los *PADs* denominados "*blank*" (numeral 3.4.1), que se representan por la celda "z". Se tiene un total de 34 celdas "z", de ahí el nombre de tiny34.

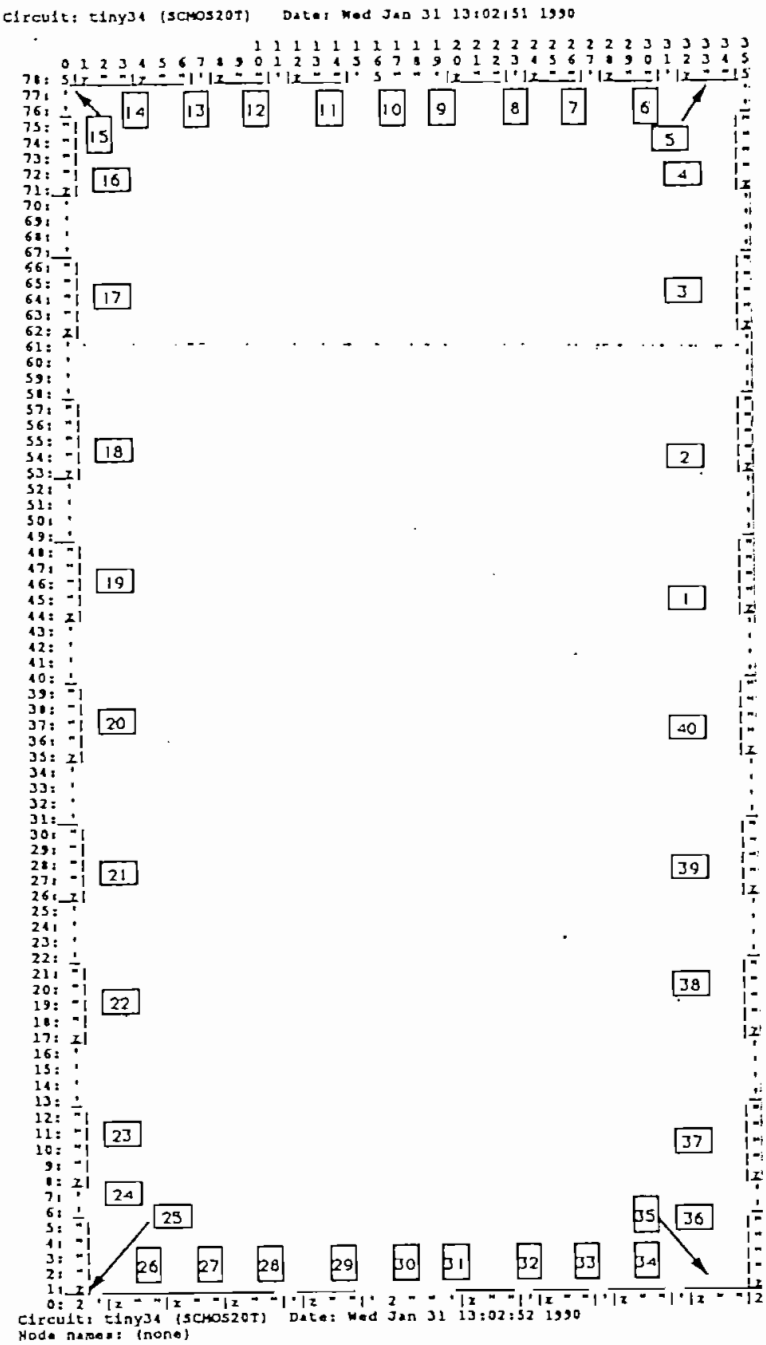


FIGURA 8.56 *Tiny34 chip* de PPL

Cada celda "z" tiene una dimensión de 5 columnas y tres filas, de igual manera que todas las celdas correspondientes a los *PADs*.

Se puede observar en la parte superior de la estructura tres celdas "5", que corresponden a los pines 5, 10 y 15. Las celdas "5" son los *PADs* para V_{DD}.

En la parte inferior se tienen tres celdas "2", que corresponden a los pines 25, 30 y 35. Las celdas "2" son los *PADs* para Gnd.

Cuando se desea insertar un *PAD* de entrada (celda "K"), salida (celda "Q") o bidireccional (celda "Z"), se debe borrar las celda "z" e ingresar el caracter correspondiente al *PAD* deseado.

En el tiny34 se tienen 6 *PADs* para polarización, de estos se destinan 4 (2 para V_{DD} y 2 para Gnd) para polarizar los *PADs*. El par restante se utiliza para polarizar el núcleo.

Esta estrategia de polarización soluciona problemas debido a la circulación de grandes corrientes en los caminos que distribuyen la polarización y que pueden causar caídas de voltaje considerables. Además, existe la posibilidad que se desarrolle ruido excesivo si muchos *PADs* conmutan simultaneamente.

El *PAD-FRAME tiny36* tiene originalmente 36 celdas "z" y destina dos *PADs* para Vdd y dos para Gnd.

Tanto el tiny34 como el tiny36, están construídas en una grilla de 79 filas y 36 columnas. Todo circuito PPL diseñado está sujeto a los límites impuestos por el *PAD-FRAME*.

a) Inserción de la compuerta EXOR en el *PAD-FRAME*

i) Cargar el *PAD-FRAME tiny34* con TILER, para esto se debe digitar *tiler tiny34*. El archivo *tiny34.ppl* debe estar en el mismo directorio de trabajo en donde se encuentra el archivo *exor.ppl*. La pantalla del computador debe presentar la estructura de la Fig. 8.56, sin los rectángulos y los números dentro de ellos.

ii) Ahora debe insertarse el módulo correspondiente a la compuerta EXOR. Para esto se elije la ubicación en la cual se desea insertar el módulo, se ingresa el comando ^X *I* y TILER pregunta el nombre del archivo. Para el ejemplo se elije la posición (R10:C15) y el archivo *exor*, se asume la extensión (.ppl).

iii) Se debe aislar el módulo EXOR del resto de la grilla. Para esto, se insertan interrupciones para filas y columnas, excepto para los caminos que llevan las señales de entrada y salida. Para el ejemplo, se asume que todas las señales serán enrutadas por la parte inferior del módulo; por lo tanto, se deben interrumpir todos los caminos de la parte superior y de los dos lados del módulo.

En la Fig. 8.57 se presenta la parte inferior del *PAD-FRAME* con el módulo de la compuerta EXOR insertado. Se representa la división de la grilla para fines de explicación.

Para aislar el módulo en el lado izquierdo, hay que ubicarse en (R12:C15) e ingresar el comando *<esc>* 7¦. Este comando se ingresa también en (R11:C15) y (R10:C15). Con esto se bloquean todos los caminos desde y hacia el módulo en el lado izquierdo.

Para aislar el lado derecho se posiciona el cursor en (R12:C18) y se ingresa nuevamente el comando *<esc>* 7¦. Se repite este comando en las posiciones (R11:C18) y (R10:C18).

Para bloquear los caminos en la parte superior se ubica el cursor en (R13:C15) y se ingresa el comando *<esc>15_*. Se repite este comando en las posiciones (R13:C16) y (R13:C17).

iv) Se decide conectar la señal "a" al pin 23, la señal "b" al pin 26 y la señal "c" al pin 31. Para esto se deben borrar las celdas "x" que se encuentran en las ubicaciones de los *PADs* correspondientes y reemplazarlas con celdas "K" para las entradas y celdas "Q" para la salida.

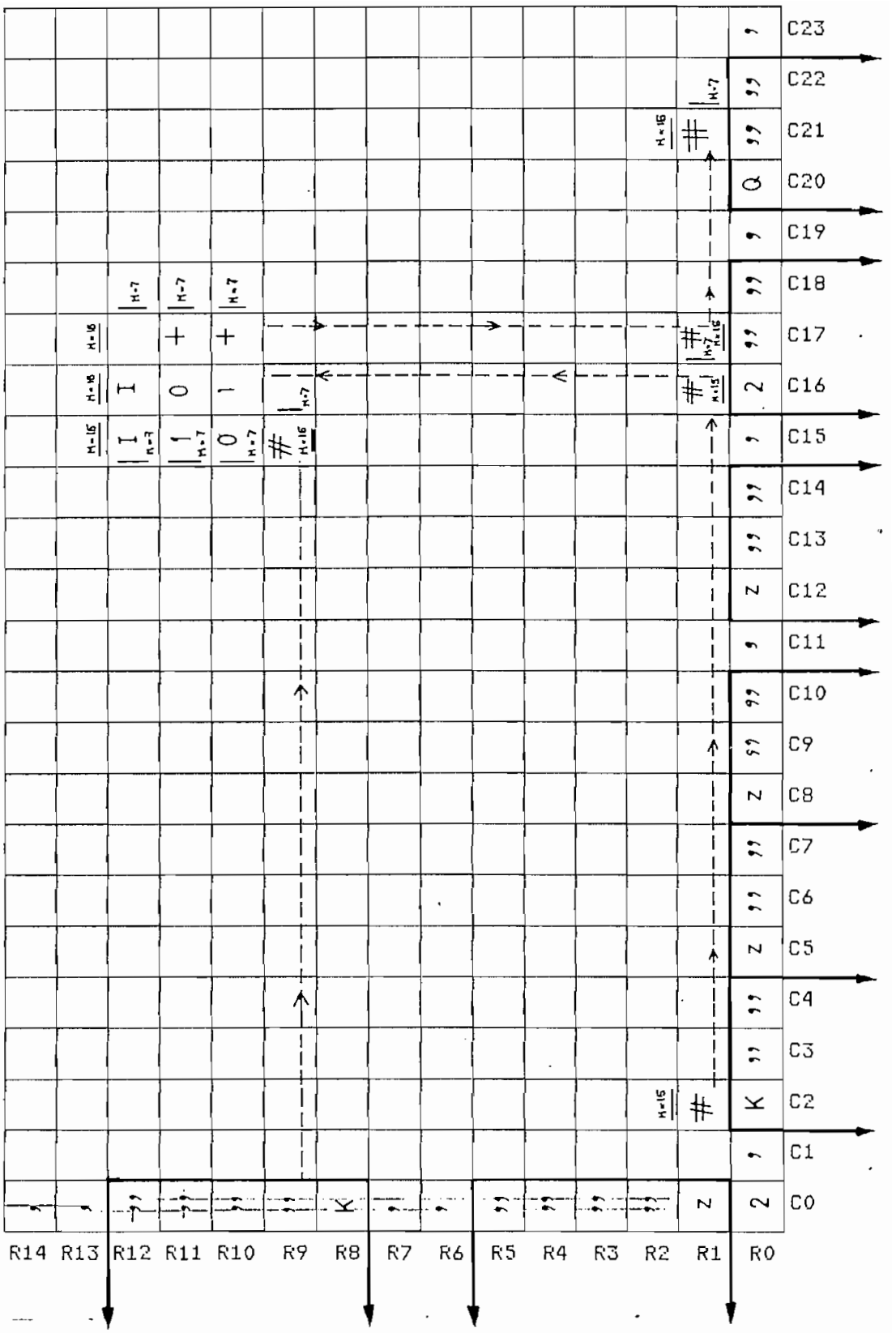


FIGURA 8.57 Inserción de la compuerta EXOR en el *PAD-FRAME*

Se ubica el cursor en (R8:C0), se borra la celda "z" con el comando ^*D* y se inserta una celda "K". En (R0:C2), se borra la celda "z" y se inserta una celda "K". Finalmente, en (R0:C20), se borra la celda "z" y se inserta una celda "Q".

Revisando la estructura de las celdas "K" y "Q" se determina, que de acuerdo a las posiciones de las celdas, los puntos de acceso a los *PADs* para este caso serán: para la señal "a" en el camino ROW de la posición (R9:C0), para la señal "b" en el camino RCOL de la posición (R0:C2) y para la señal "c" en el camino RCOL de la posición (R0:C21).

v) Ahora se debe enrutar las señales desde el módulo EXOR hasta los puntos de acceso de los *PADs*. Para la señal "a" el enrutamiento se realiza colocando una celda "#" en la posición (R9:C15). La celda "#" conecta el camino RCOL de C15 con el camino ROW de R9, lo que permite el ingreso de la señal desde el *PAD* hasta el módulo.

El camino RCOL que lleva la señal "a" debe ser bloqueado para que no vaya hacia abajo o hacia la derecha de la posición (R9:C15). Para bloquear el paso de la señal "a" hacia abajo, en (R9:C15) se digita *<esc> 15_*. Para interrumpir el paso de la señal hacia la derecha, se ubica el cursor en (R9:C16) y se digita *<esc>* 7¦.

La señal de entrada "b" se enruta colocando celdas "#" en (R1:C16) y en (R1:C2). Para interrumpir el paso hacia lugares no deseados, se interrumpe la señal "b" con *⟨esc⟩ 15_* en (R1:C16) y (R2:C2) y *⟨esc⟩* 7¦ en (R1:C17). La señal de salida "c" se enruta colocando celdas "#" en las posiciones (R1:C17) y (R1:C21) e ingresando las siguientes interrupciones: *⟨esc⟩ 15_* en (R1:C17) y (R2:C21) y *⟨esc⟩* 7¦ en (R1:C22).

vi) Cuando el *módulo* EXOR se insertó en el *PAD-FRAME*, también se insertaron los nombres de los nodos: (a, b, c). La lista de los nodos puede revisarse con el comando *^XL*. Ahora es necesario remover los nombres del *módulo* y asignarlos a los *PADs*. Esto permite utilizar el archivo (.src) que se utilizó para simular el circuito *exor.ppl*.

Para el caso de la señal "a", se ubica el cursor en la posición (R9:C0) y se ingresa el comando *^X P*. TILER pregunta por un nombre de prueba (*probe*), se responde con "a". Se formula una nueva pregunta para saber si se desea trasladar el nombre del *módulo* al *PAD*, se responde afirmativamente. El resultado final es que el nombre "a" queda asignado al *PAD* y se remueve del camino del *módulo*. Sin embargo, desde el punto de vista eléctrico, es el mismo punto. De manera similar se debe nombrar los *PADs* de las posiciones (R0:C2) y (R0:C21) con "b" y "c", respectivamente. En realidad, los nombres se asignan a las localidades en donde se ubican el nombre de las

celdas.TILER debe presentar una pantalla similar a la que se presenta en la Fig. 8.58.

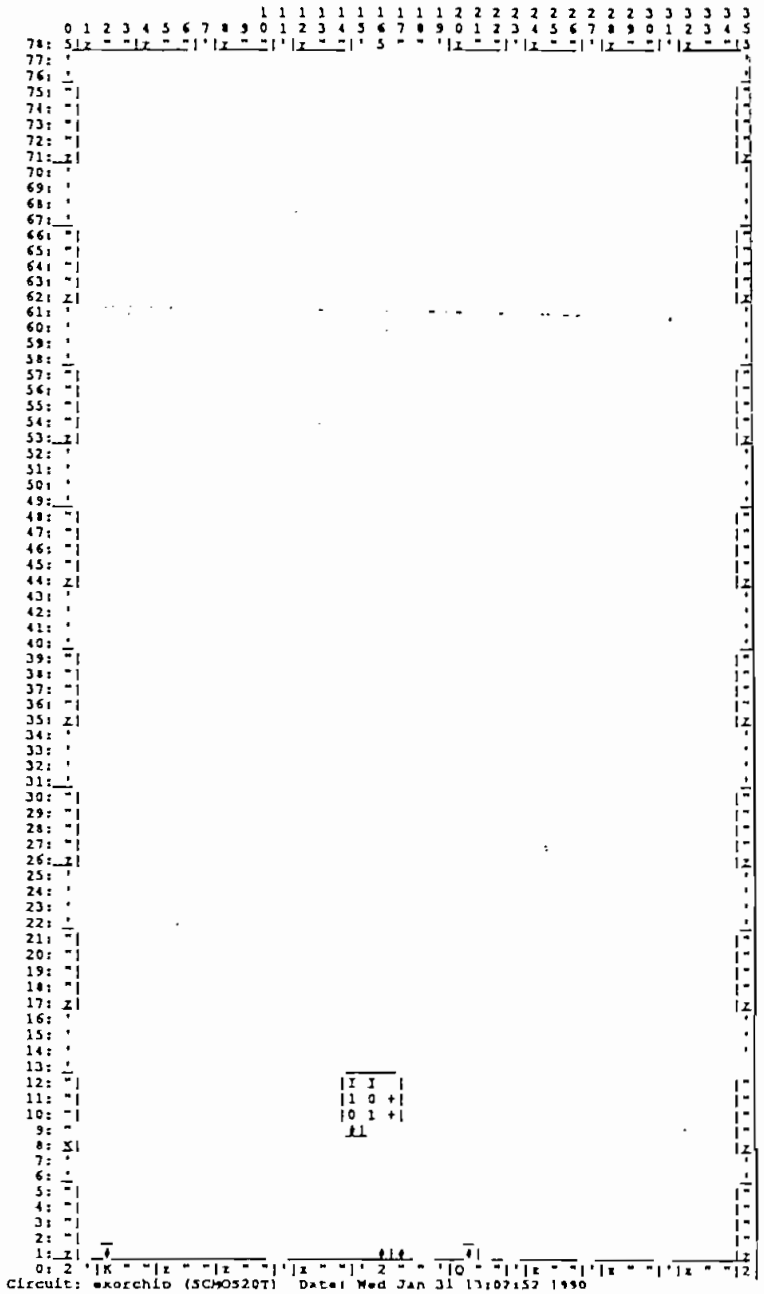


FIGURA 8.58 Diseño concluido de la compuerta EXOR

vii) El diseño se guarda con un nombre diferente, para no sobreescribir el archivo tiny34, para esto se utiliza el comando ^X ^W. TILER pregunta por un nuevo nombre. Se ha escogido el nombre *exorchip*.Finalmente se abandona TILER ejecutando el comando ^X ^C.

Una vez concluída la edición, nuevamente se debe correr SIMPPLEX sobre el archivo *exorchip.ppl*, el resultado se presenta en la Fig. 8.59. En este caso ya no existen errores de ninguna de las categorías y se puede proceder con la simulación.

```
SIMPPLEX Version 4.8. Copyright 1986-91. Bonneville Microelectronics Inc.
Loading SCMOS20T.SDB database...
Loading circuit from file exorchip.ppl...
Finding circuit context...
Tracing circuits connections...
Creating extract file exorchip.ppx...

FATAL ERRORS:    0
ERRORS:          0
WARNINGS:        0

Device statistics
PPL rows:        79
PPL columns:     36
Transistors:     58
Internal nodes:  12
Input pointers:  2/11
Output pointers: 2/11
Signal pads:     34
VDD pads:        3
GND pads:        3
Total pads:      40
```

FIGURA 8.59 Resultados de la ejecución de SIMPPLEX sobre el archivo *exorchip*

La estadística del circuito revela los siguientes datos:

i) Se han utilizado 79 filas y 36 columnas.

ii) El número de transistores se ha incrementado a 58, de los 16 que tenía originalmente el módulo EXOR. Esto se explica debido a los tres *PADs* utilizados para las señales de entrada y salida.

Utilizando SIMPPLEX se puede determinar que cada PAD, de cualquier tipo (K, Q, Z), contiene en su estructura 14 transistores. Por lo tanto, solo en los *PADS* se tienen 42 transistores, que sumados a los 16 del módulo dan el total de 58.

iii) El número de *PADs* de señal se reporta como 34, para V_{DD} hay tres, de igual manera para Gnd. El número total de *PADs* es 40.

El siguiente paso es simular el diseño completo, que ahora incluye los PADs. Para la simulación se puede utilizar el mismo archivo *exor.src* preparado anteriormente, esto gracias a que para los *PADs* se han utilizado los mismos nombres que los del módulo EXOR.

Luego de ingresar al simulador, se le indica el archivo de comandos con: ***source exor.src***. El resultado de la simulación debe ser idéntico al obtenido para el módulo EXOR.

b) PPLOW

Observando la Fig. 8.58, se concluye que el enrutamiento de las señales desde el módulo hasta los *PADs* no son visibles en la pantalla y es difícil saber exactamente por donde van las señales enrutadas. Para facilitar la visualización del enrutamiento se utiliza el utilitario PPLOW. Para el caso del ejemplo, este programa se ejecuta de la siguiente manera:

pplow exorchip

PPLOW utiliza el archivo (.ppl) y genera uno nuevo con la misma extensión. Al utilizar TILER, se presenta el circuito como en la Fig. 8.60. PPLOW utiliza los símbolos de las interrupciones para "resaltar" (*highlight*) los caminos de las señales que van hacia los *PADs*.

Sin embargo, se debe mencionar que PPLOW no añade las interrupciones solo en los caminos hacia los PADs. Al correr este programa se hace un chequeo de todos los caminos de la grilla que corren sin tener una señal asociada a ellos, y se los interrumpe, en cada celda en donde existan. Por ejemplo, en la Fig. 8.61 se presenta la impresión del modo ZOOM-IN de la compuerta EXOR, antes y después de haber corrido PPLOW.

Para entender los efectos de PPLOW se analiza la Fig. 8.61, para las interrupciones añadidas en las columnas. Se han colocado interrupciones en SCOL para la primera y segunda

columna, y en LCOL de la tercera columna. Ninguno de estos caminos se utiliza para llevar señales.

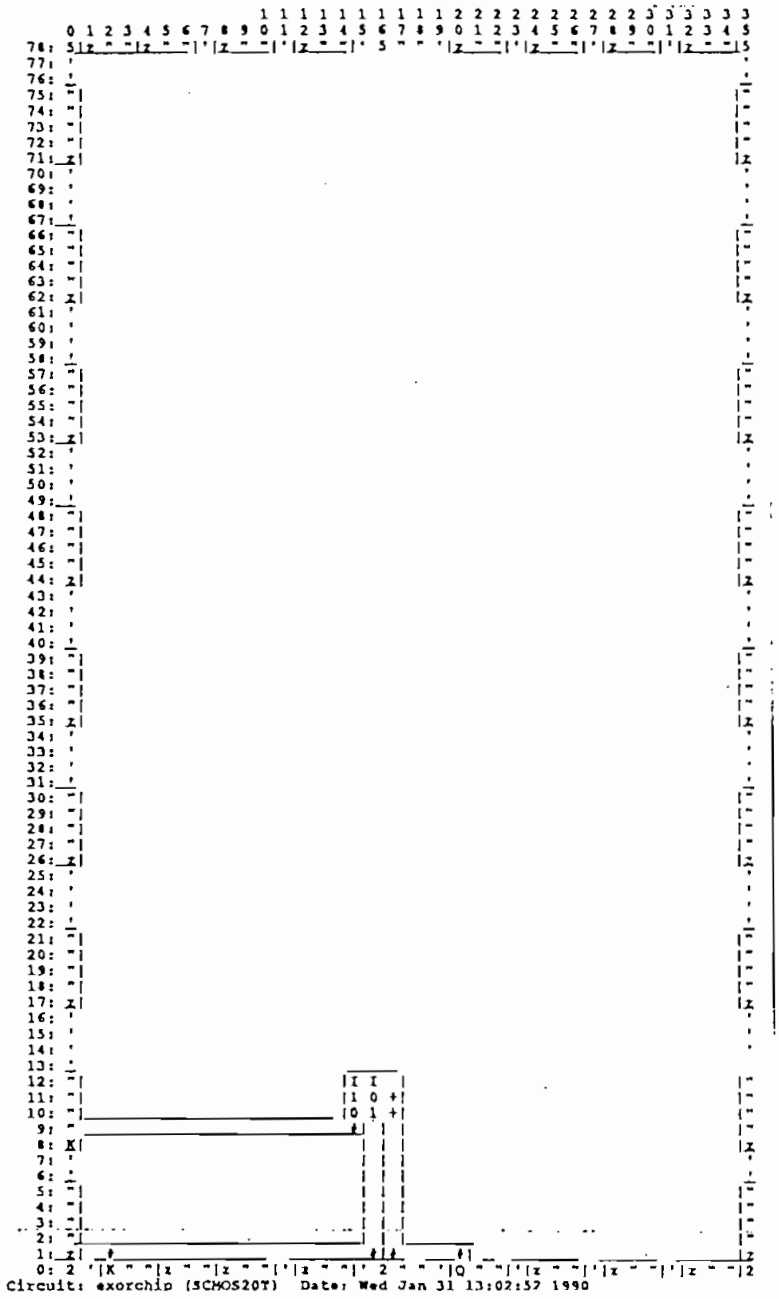


FIGURA 8.60 Diseño concluido de la compuerta EXOR luego de PPLOW

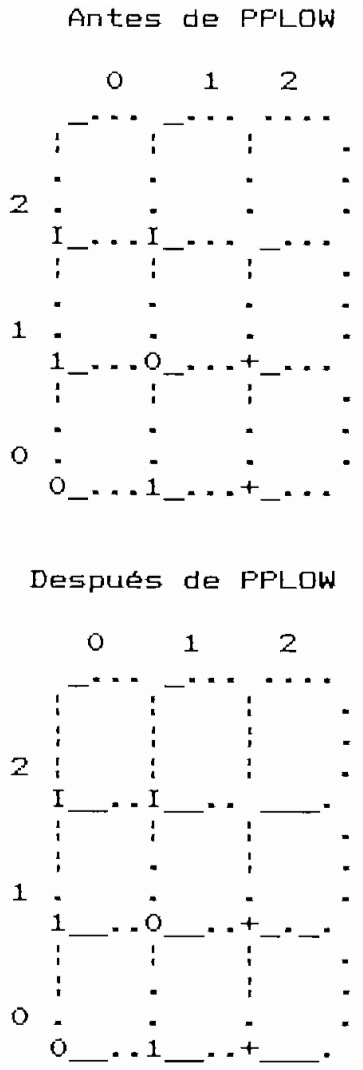


FIGURA 8.61 Compuerta EXOR antes y después de PPLOW

8.3.8 PPL2CIF: Conversión de archivos .ppl a formato CIF

Cuando se tiene la seguridad que el circuito diseñado cumple todas las especificaciones planteadas, se debe proceder a generar los archivos en formato CIF.

Esta tarea la realiza el programa PPL2CIF, toma los archivos (.ppl) y genera un archivo (.cif). Para el ejemplo de la compuerta EXOR diseñada, el programa se corre de la siguiente manera:

ppl2cif exorchip

La pantalla desplegada al ejecutar el programa se presenta en la Fig. 8.62.

```
This program will expire after 3 more Use(s).
PPL conversion Disk, Limited To 4 Conversions
(C) Copyright 1990 Bonneville Microelectronics Inc.


PPL2CIF Version 4.1a, Copyright 1986-91 University of Utah
Reading file "C:\PPL\TESIS\EXORCHIP.PPL"...
Loading cell set SCMOS20T...
Performing DRC verification...
Removing unused cell conections...
Processing file "C:\PPL\TESIS\EXORCHIP.PPL"...
        size: 220 X 2250 (microns)
        pads: 40
Extracting cells from "SMOStile.sif" library...
Extracting cells from "TINY20pd.sif" library...
Extracting cells from "CMOSbits.sif" library...
CIF-CHECKSUM: 4547434 90684
```

FIGURA 8.62 Pantalla y resultados obtenidos al ejecutar PPL2CIF

Cuando se corre este programa sobre un archivo (.ppl), pueden generarse mensajes de error. Estos mensajes indican violación de las reglas de diseño, que probablemente harán que el circuito no funcione adecuadamente. A pesar que PPL2CIF realiza el control de las reglas de diseño (DRC), ningún error puede corregirse con este programa.

PPL2CIF es también remueve las celdas de conexión no utilizadas (celdas "*blank*" que llenan la grilla), lo que reduce también el tamaño de los archivos (.cif). Con las opciones que ofrece se pueden generar también archivos pseudo-CIF (.sif) y se puede convertir archivos (.sif) a (.cif). Un archivo (.sif) tiene un formato que se basa en llamadas a las celdas utilizando el formato CIF.

Finalmente se da una estimación del área del proyecto. En este caso el circuito utiliza el *PAD-FRAME* con dimensiones especificadas por MOSIS. Las dimensiones entregadas coinciden con las solicitadas por MOSIS (2220 x 2250 µm).

También PPLACE puede utilizarse para realizar un DRC y proporcionar datos similares a los entregados por PPL2CIF. Para esto se lo invoca de la siguiente manera:

pplace exorchip

Para el caso de la compuerta EXOR, la ejecución del programa entregó los resultados presentados en la Fig. 8.63.

```
Registered to: University STUDENTS
(C) Copyright 1990 Bonneville Micro.
All Rights Reserved
PPLACE Version 1.0e, Copyright 1986-91 Bonneville Microelectronics Inc.
Reading file "exorchip.ppl"...
Loading cell set SCMOS20T...
Performing DRC verification...
Removing unused cell conections...
Processing file "exorchip.ppl"...
        size: 220 X 2250 (microns)
```

FIGURA 8.63 Pantalla y resultados obtenidos al ejecutar PPLACE

Este programa realiza un DRC y entrega un reporte de los errores CIF en un archivo con extensión (.err).

El DRC puede realizarce también dentro de TILER, para esto se utilizan los comandos:

a) *<esc> C*, este comando posiciona el cursor en donde existe un error e indica la naturaleza del problema, por lo que es útil para corregir error por error.

   Si se utiliza el comando precedido por ^U (*^U <esc> C*), TILER trata de corregir el problema automaticamente antes que entregar un mensaje. El DRC se realiza de izquierda a derecha y de abajo hacia arriba, desde la posición original del cursor.

b) *^X C*, con este comando se realiza el DRC de toda la grilla PPL y se crea un archivo (.err) en el que se presentan los errores encontrados.

Una lista de los mensajes de error que pueden presentarse y los posibles causas de los mismos, se pueden encontrar en el manual del paquete PPL.