UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXTRIBO:

UM EXTRATOR HIERÁRQUICO DE CIRCUITOS

por

Marcos Augusto Stemmer

Dissertação submetida como requisito parcial para a
obtenção do grau de Mestre em
Ciência da Computação

Ricardo Augusto da Luz Reis
orientador

Porto alegre, dezembro de 1989

CATALOGAÇÃO NA FONTE

Stemmer, Marcos Augusto
EXTRIBO: Um Extrator Hierárquico de Circuitos.
Porto Alegre, PGCC da UFRGS, 1989.
1v.
Diss. (mestr. ci. comp.) UFRGS - CPGCC, Porto Alegre, BR-RS, 1989.

Dissertação: Extração; Layout; Netlist; Células; Resistências; Capacitâncias; Transistores

AGRADECIMENTO

O meu agradecimento é endereçado a todos os professores e colegas do CPGCC, que deram o estímulo necessário para a realização deste trabalho. Quero agradecer especialmente ao meu orientador, Ricardo Reis, pelo apoio e pelas várias sugestões e revisões deste trabalho, e ao professor Tiaraju V Wagner, membro da comissão de leitura, pelo incentivo e amizade.

Também não posso deixar de agradecer aos colegas mestrandos que foram usuários das versões experimentais do extrator EXTRIBO e do editor EMA. A ajuda deles foi essencial para validar os programas em projetos reais. Eles além de encontrar um grande número de erros nos programas, deram várias sugestões para melhoramentos e lançaram desafios para a realização de novos projetos.

SUMARIO

LISTA DE FIGURAS

RESUMO

Este trabalho apresenta um programa que extrai uma descrição elétrica de um circuito integrado a partir da descrição geométrica de suas máscaras. O extrator, além de identificar os transistores e calcular as suas dimensões, também é capaz de avaliar o valor das resistências e capacitâncias parasíticas. A descrição do circuito pode ser hierárquica, com definições e chamadas de símbolos. O resultado da extração é um *netlist* hierárquico, usando definições e chamadas de subcircuitos. Os resultados da extração podem ser visualizados no *layout* do circuito, exibidos em um editor de máscaras, ou desenhados em papel com uma impressora gráfica.

ABSTRACT

This work presents a program that extracts an electrical description suitable for simulation from the layout of an integrated circuit. The extractor can identify and evaluate the dimensions of the transistors and can also calculate the parasitic resistances and capacitances. The program takes advantage of the hierarchy in the geometrical description of the circuit, generating an hierarchical netlist. A graphical layout editor allows the user to identify the components extracted. The layout with the extracted data may be hard-copied on paper using a graphic printer.

1. INTRODUÇÃO

No projeto de circuitos integrados é necessário que se tenha certeza da sua correção antes que sejam fabricados os primeiros exemplares. A confecção das máscaras e a fabricação dos circuitos integrados são processos demasiadamente demorados e dispendiosos para que possam serem usados apenas para testar um projeto. A grande complexidade dos circuitos integrados VLSI torna praticamente obrigatório o uso de computadores para a sua verificação. Por este motivo, grandes esforços têm sido dedicados ao desenvolvimento de ferramentas de projeto auxiliado por computador (PAC) voltadas à verificação e ao teste de projetos de circuitos integrados antes da sua fabricação. Atualmente existem programas que realizam verificações ou testes em vários níveis de abstração. A verificação é feita extraindo parâmetros característicos do projeto que está sendo executado e comparando-os com os valores desejados. O teste é feito através de simulações.

O nível mais baixo de abstração, a descrição geométrica das máscaras, é a última etapa do projeto antes da fabricação. A verificação das máscaras é feita com o uso de um programa verificador de regras de desenho (DRC) e de um extrator. O DRC verifica se o desenho das máscaras satisfaz a um conjunto de regras geométricas que precisam ser cumpridas para que, com as limitações da tecnologia, o circuito possa ser fabricado de maneira confiável. Tendo sido aprovado pelo DRC, o circuito pode ainda apresentar erros elétricos ou de lógica. Para a localização destes erros é necessário extraír uma descrição elétrica do circuito, formada pela lista dos seus componentes e a forma pela qual eles se interconectam. Isto é feito pelo extrator. Uma inspeção direta desta descrição do circuito permite identificar ligações erradas e componentes mal formados. Erros pertecentes a esta categoria podem ser detectados de forma automática através do uso de programas para a comparação de netlists. Estes programas comparam o netlist gerado pelo extrator com o netlist desejado para o circuito,

verificando se eles são equivalentes. Erros mais sutis, envolvendo a funcionalidade ou a lógica do circuito são detectados somente através de simulações. Os programas de simulação também usam como dados de entrada o netlist gerado pelo extrator. Na figura 1 se procurou ilustrar de forma gráfica como um extrator se enquadra dentro de um ambiente de projeto de circuitos integrados.

As simulações podem considerar vários níveis de detalhes, indo desde simulações elétricas detalhadas, incluindo efeitos parasíticos, passando por níveis de chaves, nível lógico, até transferências entre registradores. A forma pela qual o circuito deve ser descrito varia de acordo com o nível de abstração em que se quer fazer a simulação. As descrições adequadas para os níveis mais elevados, como nível de chaves, nível lógico ou de transferência entre registradores, podem ser obtidas a partir de uma descrição do circuito a nível de transistores gerada pelo extrator. Para que simulações a nível elétrico possam ser realizadas com precisão, é necessário que as resistências dos caminhos e as capacitâncias das áreas do circuito sejam levadas em consideração. Por ser uma operação muito complexa e dispendiosa em tempo de computação, o cálculo das resistências e capacitâncias parasíticas só é feito pelo extrator quando se quer fazer uma simulação elétrica detalhada.

O uso de extratores de circuitos é relativamente recente, tendo ganhado força com o uso intensivo das estações de trabalho para o projeto de circuitos integrados. Os primeiros extratores operavam sobre uma descrição plana do circuito, sem o cálculo de resistências e capacitâncias [Bar 77]. Pouco depois, no início dos anos 80, foram desenvolvidos vários métodos para calcular resistências e capacitâncias. Atualmente existem extratores de circuitos extremamente poderosos, capazes de analisarem circuitos VLSI inteiros. Porém tais programas requerem o uso de computadores grandes ou super-minis. Recentemente está surgindo um segmento de programas de projeto auxiliado por

computador (PAC. para microeletrônica baseados em microcomputadores, em especial os tipo PC. O constante aumento da capacidade de processamento dos computadores tipo PC está fazendo com que eles possam ser usados cada vez mais como estações de trabalho de baixo custo. Programas como um extrator de circuitos, porém, exigem cuidados especiais para aproveitarem com o máximo de eficiência, a capacidade limitada de memória dos PC's. Alguns extratores de circuitos baseados em PC já foram apresentados, como em [Cha 89].

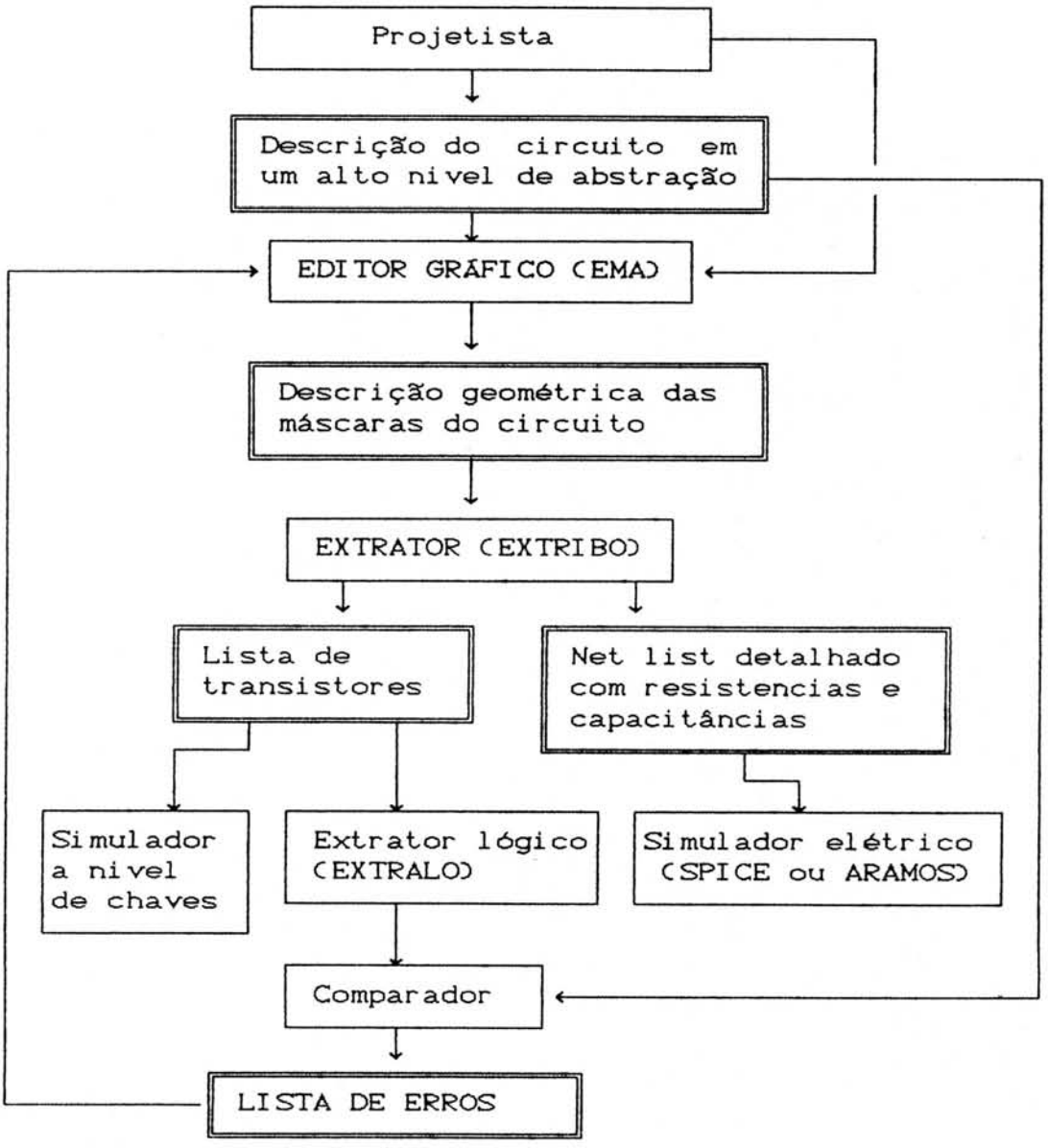


Figura 1.1. O extrator dentro de um ambiente de projeto de circuitos integrados full custom.

O extrator EXTRIBO foi desenvolvido em computadores tipo PC, como parte de um conjunto de ferramentas para o projeto de circuitos integrados. Tendo sido escrito em linguagem C, com os cuidados necessários para não comprometer a portabilidade, o programa EXTRIBO poderá ser recompilado em outros tipos de computadores, como, por exemplo, estações de trabalho UNIX.

O extrator EXTRIBO gera uma lista de transistores com as suas conexões e dimensões. Ele pode opcionalmente calcular também as resistências e capacitâncias parasíticas do circuito. A extração com o cálculo de resistências e capacitâncias é usada para a verificação detalhada de tempos de propagação e análise a nível elétrico de circuitos integrados de alto desempenho. O net-list completo deste tipo pode ser usado em simuladores a nível elétrico como o SPICE [VLA 81] ou ARAMOS [GRE87]. Por limitação dos simuladores, e das possibilidades de interpretação dos resultados, a simulação elétrica detalhada normalmente só é feita em pequenas partes do circuito.

O cálculo das resistências é feito usando um método de elementos finitos, descrito no capítulo 6. Este método se compara favoravelmente em matéria de velocidade e precisão com muitos dos melhores métodos apresentados recentemente [MIT86], [KEM88], [McC85]. O método usado consiste em formar uma rede de resistores sobre a região onde estão sendo calculadas as resistências. Esta malha é modelada através de uma matriz de admitâncias. A medida que a malha vai se formando através da criação de novos nós e resistores, os velhos vão sendo eliminados, mantendo na matriz apenas os nós recém criados. Uma idéia semelhante foi apresentada em [HAR 86]. A eliminação dos nós durante a formação da rede propicia uma vantagem muito grande em necessidade de armazenamento e tempo de computação. Os métodos usuais de elementos finitos armam uma matriz para toda a rede antes de fazer qualquer redução.

A extração sem os elementos parasíticos do circuito é muito mais rápida e econômica de memória. As operações envolvidas neste tipo de extração são a localização dos transistores, a avaliação das conexões internas e inter-níveis e a localização dos nós de alimentação: VDD e GND.

O extrator EXTRIBO pode verificar circuitos em qualquer tecnologia NMOS ou CMOS. A independência da tecnologia é obtida com o uso de um arquivo de entrada que descreve as características peculiares da tecnologia. A tecnologia é descrita de forma textual através de uma linguagem especial. No arquivo de tecnologia são definidos os níveis, com as suas características elétricas como resistividade e capacitância. Também são estabelecidas as regras de interconexões entre níveis e as regras de formação de transistores. A sintaxe da linguagem de descrição da tecnologia é apresentada em várias ocasiões ao longo de texto, e é exposta de forma sumária, porém completa no anexo A-1.

É na extração de circuitos grandes que se torna importante o uso de uma descrição hierárquica do circuito. Na descrição hierárquica, o circuito é projetado na forma de vários módulos. Estes módulos podem usar na sua definição instâncias de outros módulos definidos anteriormente. Como um determinado módulo pode ser chamado várias vezes na descrição de um circuito integrado, a descrição hierárquica se torna bem mais compacta que uma descrição plana equivalente. O extrator EXTRIBO é capaz de atuar sobre uma descrição hierárquica da geometria das máscaras, gerando uma lista de componentes também hierárquica. A extração hierárquica de circuitos é apresentada no capítulo 7.

O arquivo de saída do extrator não pode ser usado diretamente como entrada de dados por um simulador, porque o simulador exige informações adicionais que não podem ser extraídas do layout. Exemplos destas informações são as

requisições de relatórios de saída do simulador, especificações de sinais de entrada externos, ou outros componentes externos acrescentados ao circuito. A entrada destas informações exige a intervenção do usuário, com o auxílio de um editor de texto. Para isto é necessário que o usuário possa reconhecer no netlist do extrator os componentes do circuto. Os resultados obtidos pelo extrator em um formato puramente textual normalmente são de difícil interpretação. Os números de nós, transistores, resistores e capacitores na lista de componentes tem que ser associados aos seus correspondentes no desenho do circuito integrado. O extrator EXTRIBO dispõe de recursos gráficos que permitem associar visualmente o desenho do layout das máscaras com os números e nomes dos componentes encontrados. Estes recursos constam de um editor gráfico de máscaras (EMA), e do programa EMAPR, que desenha o circuito na impressora com a localização dos componentes mostrada no desenho. O EMA permite visualizar no vídeo gráfico o desenho do circuito com a identificação dos componentes encontrados pelo extrator. O programa EMA também é um editor hierárquico de máscaras, adequado para projetar manualmente circuitos integrados tipo full custom, standard cell ou gate array. Os programas EMA e EMAPR são descritos nos anexos A-2 e A-3.

O extrator completo é formado por um pacote de quatro programas e dois arquivos de dados. Os programas são:

a) EXPREP.EXE Pré processador hierárquico. Este programa deve ser rodado antes do extrator propriamente dito. Ele examina a hierarquia da descrição geométrica do circuito, gerando um arquivo que informa a localização das interfaces externas de cada célula. O preprocessador também simplifica a estrutura hierárquica do circuito. Esta simplificação é feita através da eliminação dos níveis hierárquicos desnecessários, gerando uma descrição equivalente do layout. Este programa é descrito em detalhes no capítulo 8.

b) EXTRIBO.EXE Programa principal do extrator. Este módulo usa os dados fornecidos pelo EXPREP para gerar o netlist do circuito. Ele também pode fornecer uma descrição do layout com informações a respeito da localização dos componentes encontrados. Esta descrição pode servir como dados de entrada para os programas EMA e EMAPR.

c) EMA.EXE Este programa é um editor e visualizador hierárquico de máscaras. Como editor ele tem todos os comandos necessários para projetar manualmente ciruitos integrados. Como visualizador, ele permite ver no video gráfico do computador a localização no layout dos componentes dos números dos nós encontrados pelo extrator. No capítulo 8 ele é descrito de forma resumida. Uma descrição detalhada pode ser encontrada em [STE89b].

d) EMAPR.EXE Impressor hierárquico de circuitos integrados. Este programa desenha o circuito integrado a partir da sua descrição hierárquica, usando uma impressora gráfica. O programa EMAPR pode mostrar no desenho a localização dos componentes encontrados pelo extrator, usando o mesmo arquivo de entrada que o editor EMA.

Estes 4 programas estão atualmente sendo acessados diretamente a partir do DOS, compartilhando informações através de arquivos. É um objetivo de trabalho tornar estes programas acessíveis através de um sistema integrado de gerenciamento de projeto.

2 EXTRAÇÃO DA CONECTIVIDADE

Uma das operações fundamentais que é realizada pelo extrator é reconhecer no layout das máscaras as partes que são eletricamente interligadas. O reconhecimento da conectividade pelo extrator consiste em classificar as regiões do circuito, atribuindo números de nós para cada região equipotencial. Partindo de uma representação do layout do circuito na forma de um conjunto de retângulos, a solução mais simples seria, para cada retângulo, pesquisar todos os outros a procura de algum que tenha contato com ele. Esta solução, porém é muito ineficiente em termos de tempo de computação. Os algoritmos de avaliação da conectividade mais eficientes levam em consideração a propriedade de localidade das operações sobre máscaras de circuitos integrados. A propriedade de localidade consiste no fato de que uma determinada região do circuito só se relaciona com partes na sua vizinhança. As partes distantes não precisam ser verificadas.

Enquanto uma comparação exaustiva sobre todos os retângulos tem tempo de computação de $O(N^2)$, onde N é o número de retângulos, os algoritmos mais eficientes, que aproveitam a propriedade de localidade, tem tempo de $O(N)$ ou $O(N.\log(N))$. Um dos métodos mais eficientes na operação sobre máscaras, é chamado *"corner-stitching"* [OUS84], [OUS85]. Este método consiste em usar uma geometria baseada em retângulos normalizados de uma forma especial, mantendo em cada retângulo, apontadores para os seus vizinhos. Este método, porém, só compensa realmente, se todas as operações sobre as máscaras do circuito forem realizadas sobre esta estrutura de dados, como é descrito em [OUS85], não sendo necessária nenhuma conversão.

Outro método bastante usado para analisar a geometria das máscaras aproveitando a localidade, consiste em dividir a área do circuito em linhas de varredura. Neste método, o tempo de computação é proporcional à área. O método das linhas de varredura também exige que seja feita uma conversão da estrutura de dados para um formato especial.

Um método baseado na estrutura original de retângulos da representação do circuito é o das árvores quádruplas '*quad trees*'. [PIT89], [BRO86]. Neste método, é criada sobre a estrutura geral de retângulos, uma estrutura auxiliar em forma de árvore binária bidimensional. A raíz desta árvore é um retângulo que engloba todo o circuito. Este retânglulo é partido ao meio horizontalmente e verticalmente, de modo a criar quatro partições. A estrutura de dados usada inclui apontadores para cada uma destas partições, além das coordenadas dos cantos. Cada partição destas tem uma estrutura de dados idêntica à da raiz. A altura desta árvore vai até que estes retângulos auxiliares englobem apenas um pequeno número de retângulos originais do circuito. Com o auxílio desta árvore, se pode chegar a qualquer lugar do circuito, percorrendo $\log_2(N)$ retângulos.

O método das *quad trees* tem a vantagem sobre o das linhas de varredura e o *corner-stitching* de que a estrutura de dados necessária é facilmente criada a partir de uma lista de retângulos. A eficiência do método em termos de velocidade é bastante boa. O tempo de computação para uma análise global do circuito é de $O(N.\log(N))$. Os algoritmos envolvidos são relativamente simples. O maior inconveniente deste método é a quantidade de memória necessária. A estrutura de dados auxiliar em árvore pode ocupar mais memória que o resto do circuito. Tendo em vista que o EXTRIBO foi criado para rodar inicialmente em um sistema MSDOS, com apenas 640Kb de memória disponível, se optou pelo uso de um algorítmo menos eficiente em termos de tempo de computação porém mais econômico de memória. De qualquer forma, o algoritmo das *quad trees* é uma opção interessante, que talvez seja implementada em alguma versão futura do EXTRIBO.

2.1 Ordenação dos retângulos

O método usado no extrator EXTRIBO, consiste em usar uma lista de retângulos obtida diretamente da descrição CIF ou RS do circuito. Nenhuma estrutura de dados auxiliar é acrescentada a esta lista, de modo a reduzir ao mínimo a quantidade de memória necessária. Os retângulos são definidos pelas coordenadas (X1, Y1) do canto inferior direito e (X2, Y2) do canto superior esquerdo. As coordenadas são números inteiros em unidades λ. A dimensão de λ é importante para a definição das áreas e perímetros usados para calcular as dimensões dos transistores e as capacitâncias. A unidade de distância λ é especificada no arquivo de descrição da tecnologia pela diretiva LAMBDA que tem a seguinte sintaxe:

LAMBDA < *dimensão em μm* >;

Os retângulos são ordenados segundo uma das coordenadas. Para obter o máximo de desempenho neste método, os retângulos são ordenados segundo a coordenada correspondente à maior dimensão do circuito. A ordenação pode ser feita usando um algoritmo de O(N.log(N)). As operações sobre as máscaras serão de $O(N^{3/2})$.

A extração da conectividade se divide em duas rotinas: a conectividade interna para cada nível e a conectividade externa, entre níveis. No EXTRIBO, os retângulos são classificados por níveis, formando listas encadeadas, ordenadas segundo uma das coordenadas.

Os retângulos são definidos pelas coordenadas dos cantos: (x_1, y_1) para o canto inferior esquerdo, e (x_2, y_2) para o canto superior direito, como mostra a figura 2.1.

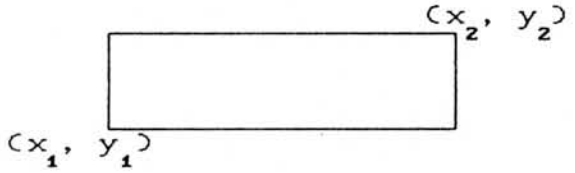


Figura 2.1. Retângulo definido pelas coordenadas dos cantos.

Se faz com que sempre $x_1 < x_2$ e $y_1 < y_2$. A ordenação é feita sempre segundo a coordenada x1. Se a maior dimensão do circuito corresponder à coordenada y, se faz uma troca de todas as coordenadas x e y, de modo que durante a extração, a ordenação seja segundo x_1.

A figura a seguir é um exemplo de conjunto de retângulos que será usado para demonstrar a vantagem de se avaliar a conectividade sobre uma lista ordenada.

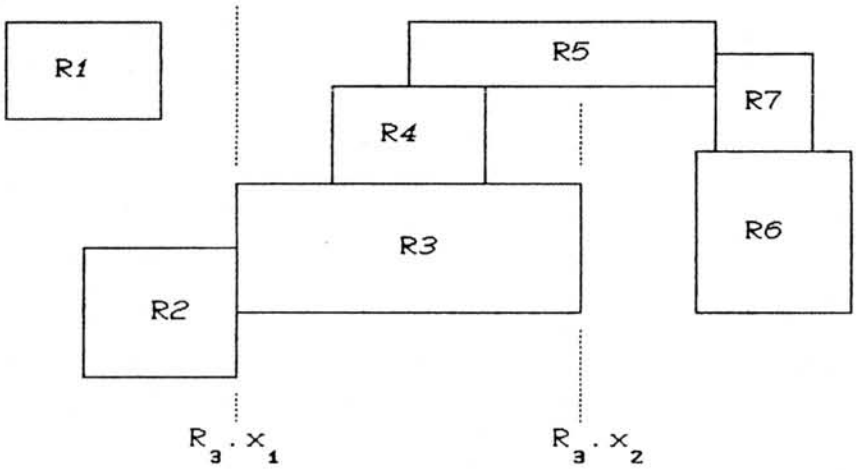


Figura 2.2 - Busca de contatos em uma lista ordenada de retângulos.

Na figura 2.2, os retângulos estão ordenados segundo a coordenada x_1. Durante a avaliação da conectividade interna, suponha que está se procurando os contatos com o retângulo R_3. As ligações dos retângulos Ri que estão à esquerda de $R_3.x_1$ ($R_i.x_1 < R_3.x_1$), já tiveram suas ligações com R_3 verificadas anteriormente. Portanto os retângulos R_1 e R_2 são eliminados implicitamente da verificação. A busca de contatos com o R_3 começa com R_4, R_5, ... Chegando ao R_6 a busca de ligações com R_3 pode ser encerrada, porque $R_6.x_1 > R_3.x_2$. Todos os outros retângulos tem a coordenada x_1 maior ou igual a $R_6.x_1$, sendo elimindos implicitamente. Desta forma a busca dos retângulos que tem ligação com R_3 fica restrita somente àqueles que tem a coordenada x1 dentro do intervalo entre $R_3.x_1$ e $R_3.x_2$, correspondente ao comprimento de R_3. No exemplo da figura apenas os retângulos R_4 e R_5 precisam ser verificados.

A verificação da conectividade interníveis é mais complicada, porque para cada nível, os retângulos são mantidos em listas encadeadas ordenadas diferentes. Da forma que foi implementada, a verificação da conectividade interníveis também exige que apenas os retângulos com a coordenada inicial compreendida dentro da largura do retângulo que está sendo verificado precisam ser pesquisados. Porém a conectividade interníveis precisa ser verificada em duas etapas. Para verificar as ligações entre os níveis A e B, é necessário verificar as ligações de A em relação a B e as de B em relação a A. Na primeira passagem se faz uma varredura ao longo de todos os retângulos Rb do nível B. Para cada retângulo Rb, se faz uma busca a partir do retângulo Ra do nível A em que:

$$Ra.x_1 \geq Rb.x_1$$

O programa segue testando os retângulos do nível A em relação a Rb até que

$$Ra.x_1 > Rb.x_2$$

Durante esta busca, se aproveita para encontrar também o retângulo Ra que satisfaz a condição

$$Ra.x_1 \geq Rb.proximo.x_1$$

Este retângulo será o ponto de partida na verificação das ligações com o próximo retângulo do nível B.

Então se pode passar para o próximo retângulo Rb. Ao final desta passagem, a verificação da conectividade não está completa, porque as ligações de retângulos em que $Ra.x_1 < Rb.x_1$ não são encontradas. É necessário que se faça esta mesma operação trocando o nível A com o nível B para encontrar estas ligações.

Não só a avaliação da conectividade, mas todas as operações sobre máscaras feitas no extrator EXTRIBO usam algoritmos que conseguem o máximo de eficiência com o uso da ordenação dos retângulos.

2.2 Indexação dos números dos nós

A conectividade do circuito é descrita atribuindo um número de nó a cada retângulo. Retângulos com o mesmo número são interligados. A verificação da conectividade consiste em encontrar todos os retângulos que tem contato, e fazer com que eles passem a ter o mesmo número de nó. Inicialmente todos os retângulos têm número de nó não definido. O estabelecimento do contato de um retângulo com número de nó definido com um com número indefinido é simples. O retângulo com número indefinido adquire o número do outro retângulo. Quando todos os dois retângulos têm números de nó definidos, é necessário mudar o número de um deles para que fiquem iguais. Então também é necessário mudar os números de todos os retângulos que tem o mesmo número do que foi trocado. Este procedimento exigiria que a cada contato encontrado seja feita uma revisão dos números de nó de todos os retângulos. A indexação dos números de nós permite que isto seja feito de uma maneira bem mais eficiente.

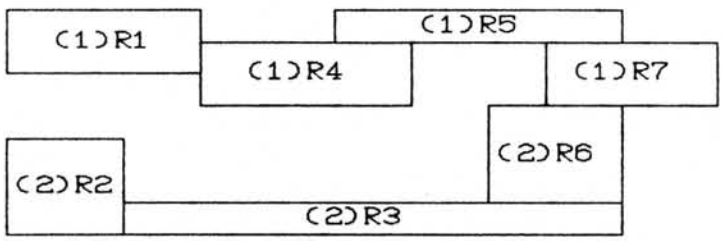


Figura 2.3 - Exemplo de avaliação da conectividade

Na figura 2.3 está sendo avaliada a conectividade da região. Os números de nós dos retângulos são mostrados entre parênteses. Suponhamos que a rotina de avaliação da conectividade acaba de constatar que existe contato entre os retângulos R6 e R7. Este contato significa que todos os retângulos que tem número de nó (1) e número (2) deverão passar a ter o mesmo número. Uma maneira de se fazer isto é percorrer todos os retângulos, e toda a vez que nó=2, fazer com que nó:=1. Porém existem maneiras muito mais eficientes.

No extrator EXTRIBO os números de nós foram indexados através de um arranjo de inteiros (lig[no]). O arranjo lig é inicializado de modo que lig[i]=i. Os números de nós, em vez de serem considerados diretamente, passam a ser lig[no]. Na situação inicial isto vem a dar no mesmo. Quando se quer estabelecer a conexão entre R6 e R7 basta fazer com que

lig[R6.no]:=lig[R7.no] , ou seja:

lig[2]:= lig[1], ou lig[2]:=1.

Com esta única operação, todos os retângulos que tinham nó=2 passam a ter nó=1

Em geral, a interligação entre um nó A e um nó B exige que se faça uma procura ao longo do array lig, substituindo todas as ocorrências de B por A. Esta busca e substituição ao logo de lig é muito menor do que se fosse feita diretamente sobre os retângulos.

Periodicamente é feita uma compactação do array lig, e uma desindexação dos números de nós. A compactação consiste em fazer com que os valores de lig passem a ser números consecutivos, sem saltos. Depois de compactado, o maior valor de lig é igual ao número de valores diferentes de lig. A desindexação é feita através da operação:

Ri.nó := lig[Ri.nó] para todos os retângulos.

Para manter a consistência, o array lig deve ser reinicializado de modo que:
lig[i] := i.

Após uma seqüência de operações como esta, o array lig terá a sua dimensão diminuída. Vários retângulos apontarão para um mesmo elemento de lig.

Operações periódicas de compactação e desindexação melhoram a eficiência do estabelecimento das conexões subseqüentes. Porém não se deve exagerar na freqüência destas operações porque elas por si são demoradas. Foram feitas várias experiências até encontrar as ocasiões em que as compactações e desindexações dão os melhores resultados. Na versão atual elas são feitas após a avaliação de cada uma das regras de conectividade.

2.3 Cálculo das áreas e perímetros

O cálculo das áreas e perímetros das regiões conexas permite estimar vários parâmetros importantes do circuito, inclusive as resistências e as capacitâncias. No relatório SPICE, a área e o perímetro das regiões de dreno e fonte são dados incluídos nas especificações dos transistores. As capacitâncias são estimadas usando um componente proporcional à área e uma proporcional ao perímetro da região.

$$C = C_p . \text{Perímetro} . \lambda + C_a . \text{Área} . \lambda^2$$

A resistência pode ser estimada com base na relação entre o perímetro e a área da região.

O cálculo da área e do perímetro de uma região conexa é feito ao mesmo tempo que é avaliada a conectividade interna. Cada retângulo tem espaço reservado para armazenar a informação a respeito da área e do perímetro da região à qual ele pertence. Estas informações são inicializadas com a área e o perímetro de cada retângulo. Quando, na avaliação da conectividade, são encontrados retângulos que se interceptam, a área e o perímetro da região de intersecção são descontadas. Ao final da avaliação da conectividade interna do nível, é feito o somatório das áreas e perímetros para cada número de nó.

2.4 Regras de conectividade

A maneira pela qual as camadas da tecnologia são interligadas é descrita no arquivo da tecnologia através de um conjunto de diretivas do tipo CON. A sintaxe é a seguinte:

```
CON <nivel 1> <nivel 2>
```

Esta diretiva indica que em todos os lugares onde houver contato entre retângulos do nível 1 e do nível 2, eles serão interligados. A conetividade dentro de um mesmo nível é implícita. O exemplo a seguir mostra como se descreve a conecctividade em uma tecnologia CMOS típica:

```
* W: Poço tipo N
* D: Difusão N
* B: Difusão P
* P: Silício policristalino
* C: Contato
* M: Nível 1 de metalização (Metal 1)
* H: Nível 2 de metalização (Metal 2)
* V: Via: (contato entre metal 1 e metal 2)
CON D W
CON D C
CON B C
CON P C
CON M C
CON M V
CON H V
```

As linhas precedidas de asterisco (*) são comentários. Observe que a ligação entre metal e difusão se dá indiretamente através dos contatos. Os níveis de metal 1 e metal 2 são ligados indiretamente através das vias. As difusões e os poços do mesmo tipo se ligam diretamente. No apêndice A1 é apresentado o conjunto completo das regras de descrição da tecnologia.

2.5 Avaliação do desempenho

O método mais usado para verificar a conectividade é o das linhas de varredura. A análise baseada diretamente em listas de retângulos não costuma ser muito recomendada na literatura. O motivo disto é que o método das linhas de varredura parece mais atraente do ponto de vista da escalabilidade. A escalabilidade significa que o esforço computacional não aumenta demais quando se passa a analisar circuitos maiores. Enquanto o método das linhas de varredura tem tempo de computação proporcional ao número de retângulos $O(N)$, o método da pesquisa sobre a lista ordenada de retângulos tem tempo de computação de $O(N^{3/2})$. Porém isto não significa que o método dos retângulos ordenados é mais lento. Pelo contrário, para circuitos não muito grandes, ele é bastante mais rápido que o método das linhas de varredura.

Para que se possa ter uma idéia comparativa do desempenho do método apresentado, se procurou na literatura uma referência a um extrator baseado em PC. A maioria dos extratores de que se tem notícia rodam em computadores mais poderosos, que tornam inválida uma comparação em termos de tempo de computação. Em [CHA89] é apresentado um extrator baseado em linhas de varredura para computadores tipo PC. Foram realizados testes comparativos no extrator EXTRIBO conforme se pode observar na figura 2.4. Os testes foram feitos sobre circuitos não hierárquicos. Por estes resultados se observa que mesmo no circuito com 100 transistores, o extrator EXTRIBO ainda é cerca de 4 vezes mais rápido. A medida que aumentam as dimensões do circuito, a vantagem do EXTRIBO tende a diminuir. Porém, como o EXTRIBO é hierárquico, normalmente não haverá necessidade de se extrair células muito grandes. Os circuitos realmente grandes serão extraídos como um conjunto de pequenas células.

Circuito	Número de retângulos	Número de transistores	Tempo em segundos
Inversor CMOS	57	2	0,4
Multiplexador de 2 entradas	153	6	1,2
Flip-Flop mestre-escravo	389	26	4,7
PLA em NMOS	984	153	51,2
Modem [REI 89]	6789	528	8:31,0

Figura 2.4 - Tabela que demonstra o desempenho do extrator EXTRIBO, atuando sobre um layout não hierárquico.

Circuito	Número de nós	Número de transistores	Tempo em segundos
Buffer de saída de um PLA	10	2	6
Somador serial NMOS	44	47	27
Módulo multiplicador em complementos de 2	93	104	1:30

Figura 2.5 - Desempenho de um extrator baseado em linhas de varredura [CHA 89]. Não foi mencionado especificamente o modelo de computador usado no teste.

Os tempos foram avaliados usando um computador tipo PC XT de 4,77MHz. Destes resultados conclui-se que os algoritmos usados tem um desempenho plenamente satisfatório. Os circuitos usados nos testes são células da biblioteca de células do projeto TRANCA [MOR 88]. O circuito modem foi apresentado em [REI 89]. Para circuitos não muito grandes, o método da pesquisa sobre uma lista ordenada de retângulos leva uma vantagem considerável sobre o método das linhas de varredura. Esta vantagem é ampliada pelo fato de o extrator EXTRIBO atuar sobre a descrição hierárquica do circuito, que descreve circuitos grandes através de várias células menores.

3 OPERAÇÕES LÓGICAS SOBRE MÁSCARAS

Depois da extração da conectividade, outra operação que o extrator deve realizar é criar novos níveis com base em operações lógicas sobere as máscaras de níveis já definidos. Tais operações, além de tornar mais versátil a lignuagem de descrição da tecnologia, são necessárias na extração dos transistores, capacitâncias e resistências.

Matematicamente, as máscaras dos circuitos integrados podem ser consideradas como conjuntos de pontos do plano. Desta forma, as operações sobre as márcaras são operações sobre conjuntos. Estas operações podem ser de união (A $\cup$ B), intersecção (A $\cap$ B) ou complementação ($\bar{A}$). Em lugar da complementação, nas operações sobre máscaras é mais conveniente usar a operação de exclusão, definida como A $\cap \bar{B}$. A complementação não é usada porque ela definiria uma região infinita do plano. Estando as máscaras representadas por conjuntos de retângulos, cada uma destas operações dá origem a um algoritmo que a implementa. Novos níveis de máscaras são definidos como sendo o resultado de operações entre níveis já existentes.

3.1 União

A geometria resultante da união da máscara A com a B consiste no conjunto de todos os pontos que pertencem a A ou a B. A operação de união é a mais fácil de ser implementada. Para realizar a operação

$$C := A \cup B \qquad (3.1)$$

basta que se coloque na lista de retângulos de C, todos os retangulos de A e todos os de B. Na versão atual do extrator EXTRIBO, não foi usada a operação de união.

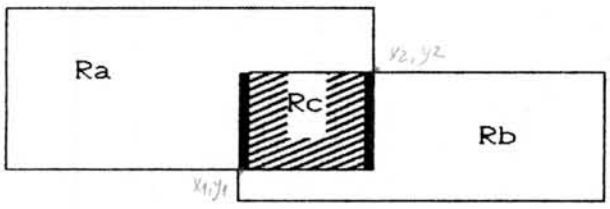
3.2 Intersecção

A operação de interseção exige que se faça uma busca dos retângulos das duas listas que tem contato entre si. Esta busca aproveita a ordenação dos retângulos da mesma forma que a avaliação da conectividade entre níveis diferentes. Suponha que se deseja realizar a operação

$C := A \cap B.$ (3.2)

O programa faz uma busca sobre as listas ordenadas de retângulos A e B, procurando retângulos que se interceptam. O procedimento desta busca é o mesmo descrito para a conectividade interniveis no ítem 2.2. Quando são encontrados um retângulo Ra do nível A e um retângulo Rb do nível B que tem contato, é criado um retângulo Rc do nível C, com as seguintes dimensões:

Rc.x1 := máximo (Ra.x1, Rb.x1);
Rc.y1 := máximo (Ra.y1, Rb.y1);
Rc.x2 := mínimo (Ra.x2, Rb.x2);
Rc.y2 := mínimo (Ra.y2, Rb.y2).



Fiura 3.1 - Intersecção de dois retângulos

Para obter o máximo de eficiência a partir da ordenação dos retângulos, a avaliação das intersecções entre dois níveis de máscaras é feita em duas etapas. Uma etapa é feita percorrendo todos os retângulos Ra do nível A. Para cada retângulo Ra é feita uma busca no nível B a partir do retângulo Rb em que:

Rb.x1 ≥ Ra.x1, indo até que Rb.x1 > Ra.x2.

Nesta primeira etapa não são encontradas as intersecções entre retângulos em que Rb.x1 < Ra.x1. Estes

contatos são encontrados repetindo a operação, invertendo os papeis dos níveis A e B.

Na descrição da tecnologia se pode definir um novo nível como sendo a intersecção de dois outros níveis definidos anteriormente. A sintaxe desta definição de novo nível na descrição da tecnologia é a seguinte:

LET C = A & B

A operação de intersecção também é usada pelo extrator para extrair os transistores e para calcular as capacitâncias verticais entre dois níveis.

3.3 Envolvente

Para manter uma consistência matemática nas definições a seguir, é necessário buscar na teoria dos conjuntos aplicada à topologia o conceito de envolvente. A envolvente de uma determinada região é esta região unida com o seu contorno.

$$envolvente(A) = A \cup contorno(A)$$

Portanto a envolvente de uma região do plano A, é a menor região que contem A completamente.

Uma propriedade interessante do contorno e da envolvente é que:

$$contorno(A) = envolvente(A) \cap \overline{A}$$

A maneira de representar a geometria que foi usada no extrator inclui obrigatoriamente os contornos das regiões. As regiões são representadas pelas suas envolventes.Isto faz com que a intersecção de regioes mutuamente exclusivas gera a intersecção dos contornos destas regiões, em vez de um conjunto vazio, como se iria esperar, pela teoria dos conjuntos.

3.4 Exclusão

A operação de exclusão implementada no extrator EXTRIBO pode ser definida em termos de conjuntos como sendo:

$C := envolvente(A) \cap \overline{B}$

Isto é: C é o conjunto de todos os pontos que pertencem a A e não pertencem a B, incluindo os contornos de A e de $\overline{B}$. Como na intersecção, esta operação é implementada como uma busca de todos os retângulos do nível A e do nível B que se interceptam. Quando uma intersecção é encontrada, vários casos são possíveis, dependendo das dimensões e das posições dos retângulos. Poderão resultar no nível C entre zero e quatro retângulos.

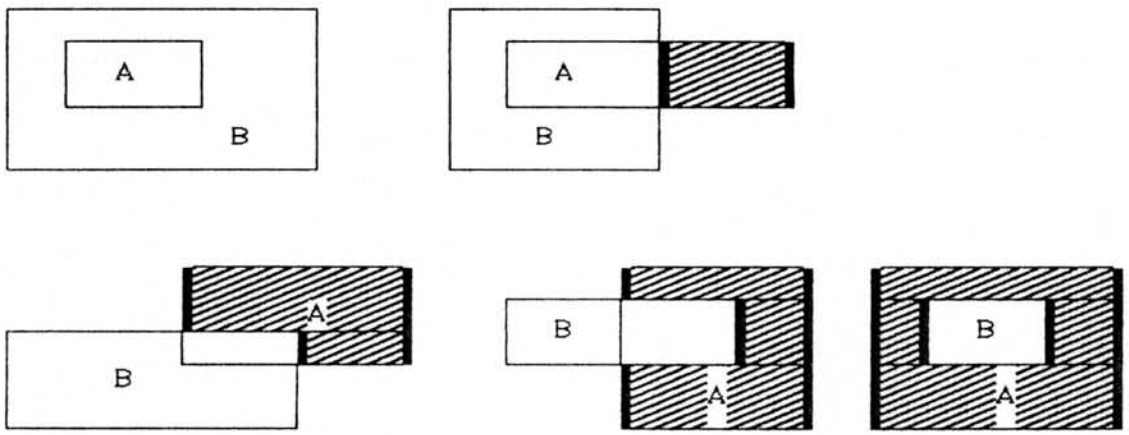


Figura 3.2 A operação $A := envolvente(A) \cap \overline{B}$, nos casos que resultam em 0, 1, 2, 3 ou 4 retângulos.

A operação de exclusão é usada na extração dos transistores. Para poupar memória, o nível do qual se faz a exclusão é o mesmo onde é armazenado o resultado. Isto também melhora a eficiência da operação porque nas regiões onde não há intersecção entre retângulos de A e de B, o nível A permanece inalterado. Portanto a operação realizada é: $A := envolvente(A) \cap \overline{B}$.

Como o resultado é armazenado junto com um dos operandos, o conjunto de retângulos do nível A se modifica durante a operação de exlcusão. Isto torna necessário que se tome uma série de cuidados para que ser mantenha a consistência da lista ordenada de retângulos, de modo que se possa realizar a operação em $O(N\sqrt{N})$. A operação precisa ser realizada em duas etapas: A primeira etapa tem uma passagem pelos retângulos do nível B no loop externo. Para cada retângulo Rb do nível B são verificados os retângulos Ra do nível A no intervalo em que:

$Ra.x1 \geq Rb.x1$ até que $Ra.x1 > Rb.x2$.

Nesta primeira passagem os retângulos do nível A estão sempre à direita dos de B. São identificados os casos mostrados na figura 3.3.

A segunda passagem é feita percorrrendo os retângulos do nível A. Para cada retângulo Ra, o programa procura retângulos do nível B, partindo do primeiro retângulo Rb em que $Rb.x1 > Ra.x1$, prosseguindo até chegar a um retângulo Rb para o qual $Rb.x1 > Ra.x2$.

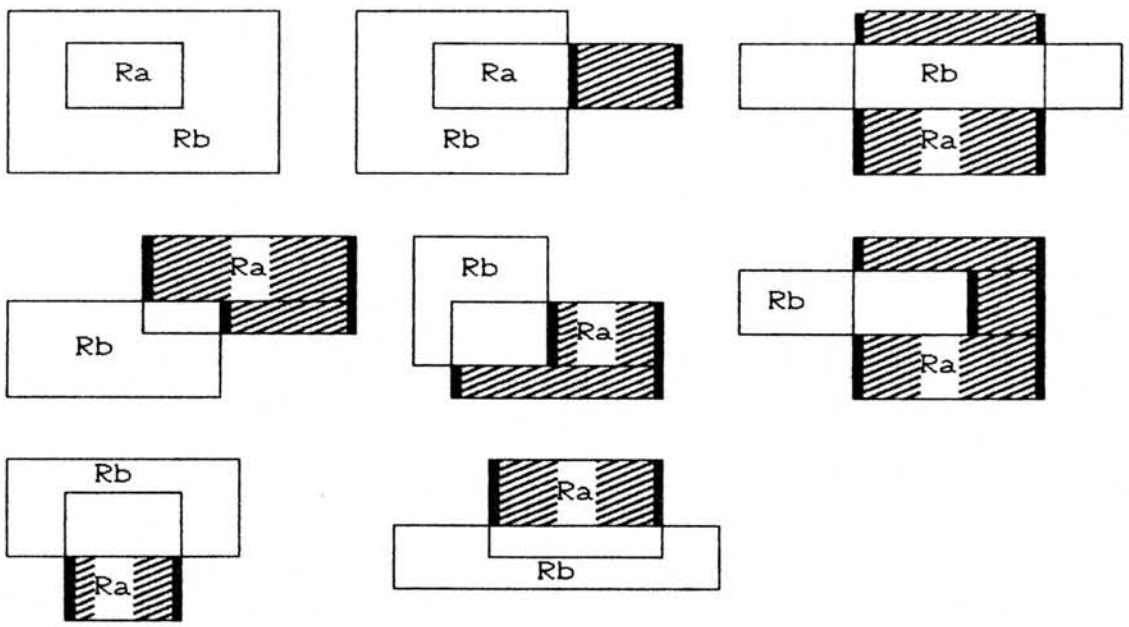


Figura 3.3 Casos de de contatos entre retângulos encontrados na primeira passagem da operação de exclusão. Nesta primeira

Na segunda passagem são reconhecidos os seguintes casos:

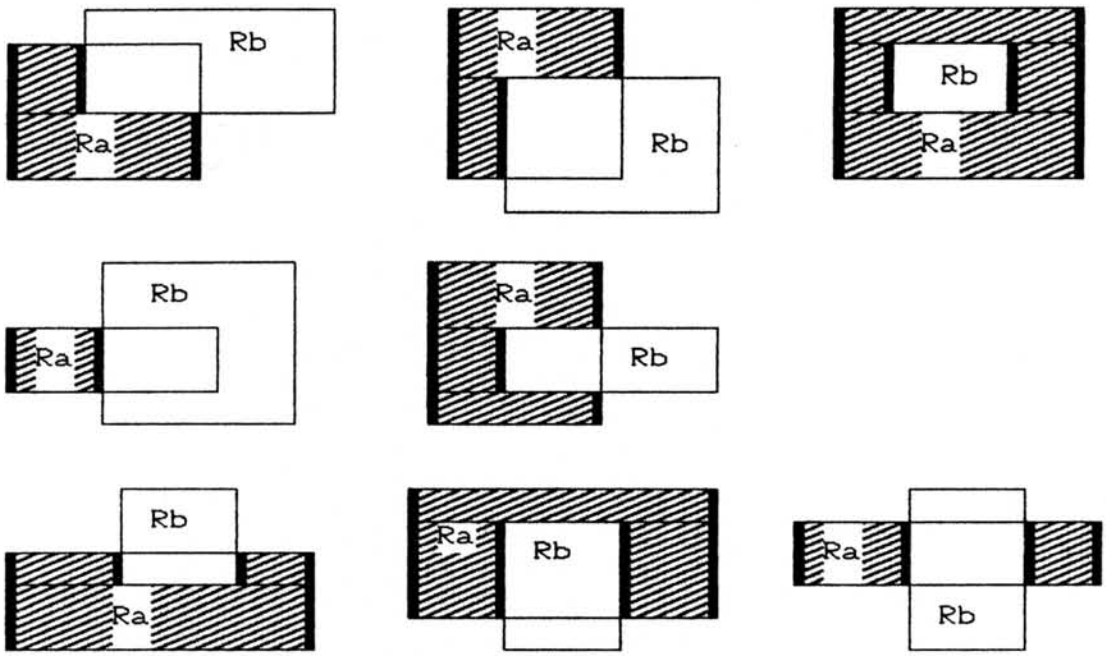


Figura 3.4 - Situaçãoes cobertas pela segunda passagem pela lista de retângulos durante a realização da operação de exclusão.

Os retângulos que são criados durante a operação de exclusão devem ser inseridos na lista com cuidado para mantê-la ordenada segundo a coordenada x1. Nos casos em que um retângulo do nível A tem a sua coordenada X1 alterada, ele tem que ser apagado e reinserido na lista em seu devido lugar.

Na versão atual do extrator, a operação de exclusão é usada na extração dos transistores e para preparar a lista de terminais dos resistores. Nestas operações, além da exclusão, também se quer obter a intersecção dos mesmos dois níveis. Para poupar uma pesquisa sobre todos os retângulos, a rotina de exclusão, também pode gerar uma intersecção ao mesmo tempo, com um acréscimo mínimo no tempo de computação.

4. EXTRAÇÃO DOS TRANSISTORES

O extrator EXTRIBO, na versão atual, é capaz de reconhecer os transistores tipo MOS que são usados nos vários tipos de tecnologias CMOS ou NMOS. Comparada com a extração das resistências, a extração dos transistores é uma tarefa fácil. Ela se divide em duas etapas. A primeira etapa é a localização geométrica dos transistores com o recorte da região ativa. A segunda etapa é a determinação dos terminais e o cálculo do comprimento e da largura do canal.

4.1 Localização dos transistores

Os transistores tipo MOS, de modo geral são formados a nível de máscaras pela sobreposição de tres camadas: o substrato, a região ativa e a porta. Para justificar o procedimento usado na extração dos transistores, vejamos qual é o significado destas camadas no processo de fabricação do circuito integrado.

A primeira camada é o substrato. O substrato é a camada básica sobre a qual o transistor será criado. Esta camada pode não ser descrita explicitamente ao nível de máscaras em certas tecnologias. Do ponto de vista do extrator, o substrato é usado para determinar de que tipo é o transistor que foi encontrado. Dependendo da tecnologia, o nível de implantação usado para criar transistores tipo enriquecimento pode ser definido para o extrator como substrato.

A segunda camada é a região ativa. A região ativa vai formar o dreno e a fonte dos transistores MOS. Esta é a região que, no processamento de um circuito integrado, fica recoberta por uma camada mais fina de óxido, sobre a qual se faz o implante e a difusão de dopantes que darão o caráter P ou N do transistor.

A terceira camada é que vai formar a porta (gate) dos transistores MOS. Normalmente ela é formada por metal ou silício policristalino, que é depositado sobre a camada de óxido. Fora das regiões ativas, esta camada se comporta simplesmente como um condutor. Nas regiões em que a camada de gate se cruzar com a região ativa, haverá a formação de um transistor. A difusão P ou N é feita sobre a região ativa depois que a camada de gate foi depositada. Desta forma o gate serve como máscara de auto-alinhamento para a difusão.

Ao encontrar um cruzamento das camadas que formam um transistor, o extrator deve eliminar da região ativa a parte que ficou sob o gate, de modo a separar o dreno da fonte. Em termos de operações sobre máscaras tem-se que:

$$D := A \cap \text{envolvente}(\overline{P}) \qquad (4.1)$$

onde D é a região de difusão do transistor, A é a região ativa e P é o nível de polisilício que vai formar o gate. Esta equação informa que o nível de difusão é formado por todos os pontos da região ativa que não estão sob o nível de polisilício. Esta equação expressa o fato de o nível de polisilício servir como máscara para o implante e subseqüente difusão de impurezas.

Ao encontrar um transistor, além de separar o dreno e a fonte, é necessário manter informações que permitam, em uma etapa posterior, reconhecer os transistores encontrados, e calcular as suas dimensões. Estas informações são mantidas através da criação de outros níveis resultantes de operações sobre as máscaras que formam os transistores. A primeira destas operações cria um nível de gate definido como:

$$G := A \cap P \qquad (4.2)$$

Os retângulos do nível G mantém informações sobre o tipo de transistor encontrado e referências aos retângulos dos níveis de gate e de difusão que lhes deram origem. Como as informações a respeito do transistor são mais do que se pode armazenar na estrutura de dados de um retângulo, é

criado também um segundo nível de gate com a mesma definição geométrica:

$$G2 := A \cap P \qquad (4.3)$$

O dreno e a fonte dos transistores são armazenaos em um novo nível DS, definido como:

$$DS = D \cap G \qquad (4.4)$$

Substituindo (4.1) e (4.2) em (4.4), e aproveitando a propriedade (3.3) vem que:

$$DS = A \cap \text{contorno}(P)$$

Nos casos em que o extrator calcula a resistência da região de difusão, os retângulos do nível DS servirão como terminais dos resistores.

Na figura 4.1 são mostrados os níveis que resultam das operações sobre máscaras (4.1), (4.2) e (4.4), que são realizadas durante a entração de um transistor.

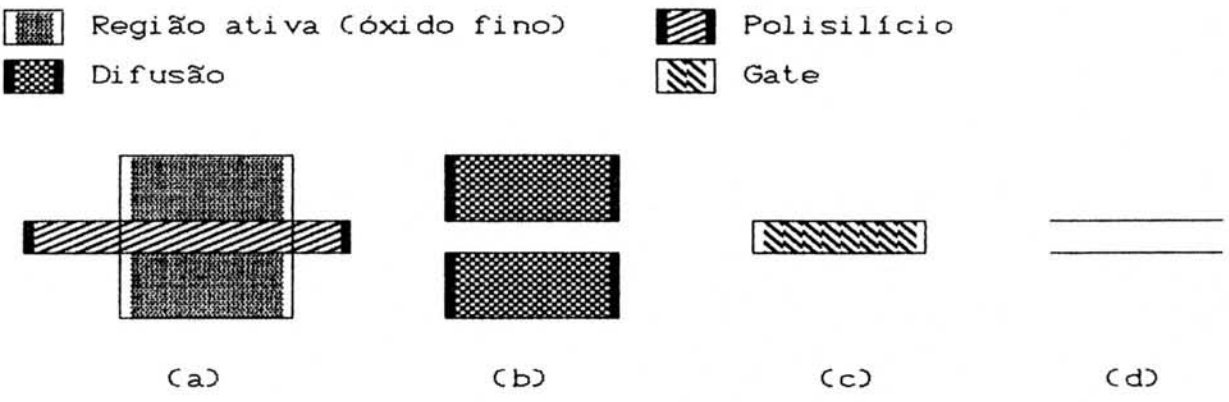
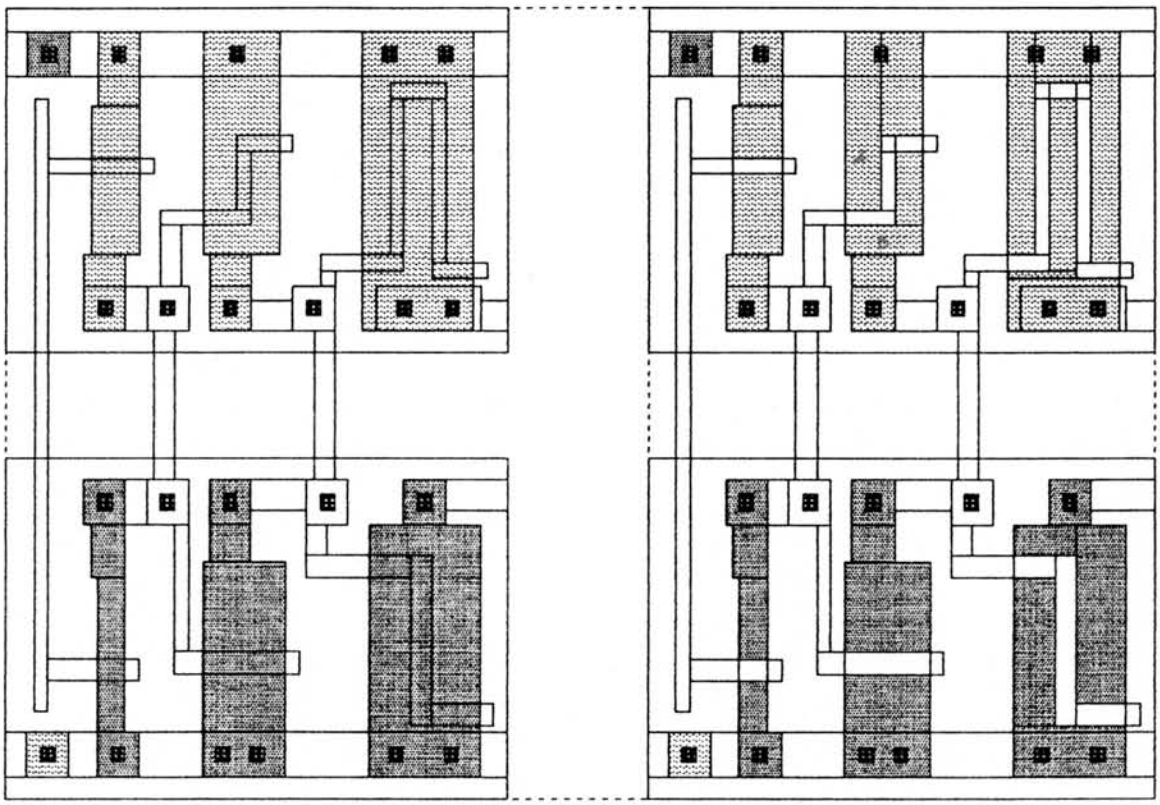


Figura 4.1 Operações sobre as máscaras realizadas durante a extração de um transistor. Em (a) vemos o cruzamento do retângulo de polisilício com a região ativa. Em (b) vemos o nível de difusão ($D = A \cap \text{envolvente}(\overline{P})$), criado pelo extrator. Em (c) aparece o nível de gate ($G = A \cap P$). Em (d) temos o nível de dreno e fonte: ($DS = D \cap G$).

A figura a seguir mostra como o extrator realiza estas operações sobre um inversor CMOS em que os transistores são formados por vários retângulos. Nela também aparecem os números de nós criados pelo extrator. Este desenho foi feito pelo módulo EMAPRINT, incluído no pacote do extrator.



(a) (b)

Figura 4.2 Um inversor CMOS: (a) O inversor na sua forma original. (b) O inversor com a difusão recortada para formar os transistores.

Para a avaliação da conectividade, considera-se implicitamente que o nível de gate tem ligação com o de polisilício, e que o nível de dreno e fonte tem ligação com a difusão.

Portanto as regras

```
CON DS D
CON G P
```

são implementadas internamente no extrator.

4.2. Cálculo das dimensões dos transistores

As operações descritas no ítem anterior são feitas antes do cálculo das resistências e capacitâncias e da extração da conectividade. Nesta ocasião ainda faltam informações para que as características do transistor sejam determinadas. Depois que a conectividade foi avaliada, os transistores encontrados são reanalisados, aproveitando as informações de números de nós, áreas e perímetros.

A conectividade interna do nível de gate G2 é avaliada. A conectividade dos níveis G e DS já foram avaliadas durante a verificação das regras de conectividade descritas na tecnologia. Vários retângulos interligados do nível G2 vão formar o gate de um único transistor. Os retângulos do nível G2 são ordenados segundo os números de nós, para que os que estiverem interligados fiquem em seqüência na lista. O dreno e a fonte são encontrados, procurando retângulos do nível DS correspondentes ao retângulo de G2 que está sendo verificado e que tenham números de nós diferentes. Se um transistor não tiver retângulos do nível DS com números de nós diferentes, correspondendo aos retângulos do nível G2, o dreno e a fonte deste transistor deverão estar em curto circuito. Neste caso o transistor será eliminado da lista de transistores. Opcionalmente este fato poderá gerar uma mensagem de erro.

As dimensões do canal dos transistores são calculadas com base no perímetro e na área das regiões do nível G2 e do nível DS. A largura do canal é determinada como sendo a metade do perímetro do dreno ou da fonte.

L = perímetro(DS) / 2 (4.5)

O comprimento do canal é calculado com base na área de gate:

W = area (G2) / L (4.6)

Para gerar um net-list em formato do SPICE [VLA81], é necessário também calcular a área e o perímetro das regiões de dreno e de fonte. Esta informação de área e perímetro é obtida durante a avaliação da conectividade e mantida na estrutura de dados dos retângulos, conforme foi visto no capítulo 3. A seguir é apresentado o net-list resultante do inversor da figura 4.2.

```
*
* UFRGS - PGCC - GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CIRCITO: buf33.cel
* TECNOLOGIA: gren2.tec TIPO CMOS
*
* CELULA BUF3
*
* Transistores tipo NMOS: 3
* Transistores tipo PMOS: 3
*
.SUBCKT BUF3 0 1 2 5
MP1   4 5 1 1 PMOS L=2.0U W=7.0U AD=137P AS=109P PD=58U PS=48U
MP2   3 4 1 1 PMOS L=2.0U W=21.0U AD=144P AS=204P PD=70U PS=70U
MP3   2 3 1 1 PMOS L=2.0U W=63.0U AD=214P AS=300P PD=94U PS=140U
MN4   3 4 0 0 NMOS L=3.0U W=12.0U AD=210P AS=168P PD=70U PS=52U
MN5   4 5 0 0 NMOS L=3.0U W=4.0U AD=115P AS=64P PD=60U PS=38U
MN6   2 3 0 0 NMOS L=3.0U W=36.0U AD=240P AS=232P PD=92U PS=86U
.ENDS BUF3
.END
```

4.3 Transistores no arquivo de descrição da tecnologia

No arquivo de descrição da tecnologia, os tipos de transistor são descritos através de uma diretiva MOS, com a seguinte sintaxe:

MOS *<nome> <nível de gate> <nível de dreno e fonte> <nível de substrato> <comprimento efetivo do canal ΔL> <largura efetiva do canal ΔW>*

O *<nome>* é o nome do modelo de transistor que aparecerá na lista de componentes. A seguir vem as letras que designam os níveis que formam gate, dreno e substrato do transistor. O nível de substrato é o nível de base que deve existir para caracterizar o tipo de transistor. Este nível de substrato pode ser o poço no caso das tecnologias CMOS, ou uma implantação, que distingue os transistores de enriquecimento dos de depleção nas tecnologias NMOS. Nos casos em que não deve haver nada por trás da região do canal do transistor, usa-se como substrato um nome de nível não definido na tecnologia. O comprimento e a largura efetiva do canal são valores que são somados ao comprimento e à largura do canal dos transistores no *net-list* em formato SPICE.

5 LOCALIZAÇÃO DA ALIMENTAÇÃO

Para gerar um *netlist* completo e correto é indispensável que o extrator identifique os nós de alimentação. Toda a interpretação do funcionamento do circuito vai depender disto. O substrato dos transistores vai estar ligado à terra ou a Vcc. As capacitâncias parasíticas são normalmente ligadas à terra. Além disto, o nó de terra deve ter um nome ou número especial nos *net-lists* gerados pelo extrator.

Foram implementados no extrator três métodos para descobrir quais são os nós da alimentação em uma célula do circuito. Um método é direto através de níveis da tecnologia que estarão necessariamente ligados a Vcc ou à terra. Os outros dois métodos são estatísticos, baseados no número de transistores ligados a Vcc ou à terra.

5.1 Método direto através de um nível especial

Este é o método mais simples e garantido de descobrir quais são os nós da alimentação. As diretivas VDD e GND definem no arquivo de tecnologia quais são os níveis que estão necessariamente ligados à alimentação. O extrator então passa a considerar todos os retângulos destes níveis como estando interligados, mesmo que não haja uma ligação geométrica entre eles. A sintaxe das diretivas VDD e GND é a seguinte:

VDD < *nível* >
GND < *nível* >

Nas tecnologias CMOS, os poços tipo P são normalmente ligados à terra e os poços tipo N são ligados a Vcc através dos *body-tyes*. Nestes casos basta usar as diretivas VDD e GND para os níveis dos poços. Quando os poços não são definidos na tecnologia, ou não são ligados explicitamente através de *body-tyes*, é necessário criar níveis especiais para localizar a alimentação. O inconveniente de se criar estes níveis especiais é que isto exige que se modifique a descrição original do circuito através de uma intervenção especial do usuário.

5.2 Método estatístico para CMOS

Quando não está disponível uma informação direta a respeito de qual é a região do circuito correspondente à terra e à alimentação, o extrator precisa recorrer a noções estatísticas para decidir quais provavelmente são os nós de alimentação. Não se pode dar uma resposta com certeza absoluta neste caso. Os métodos estatísticos usados iniciam por uma contagem dos terminais de transistores ligados a cada nó do circuito. O palpite de quais são os nós de alimentação é dado com base nas seguintes características dos circuitos CMOS:

a) VDD ou GND normalmente não são usados como gate de transistores, já que isto não tem utilidade lógica.

b) VDD normalmente é o nó que tem o maior número de ligações com o dreno ou fonte (neste ponto ainda não se pode distinguir o dreno da fonte) dos transistores tipo P.

c) GND normalmente é o nó que tem o maior número de ligações com o dreno ou fonte dos transistores tipo N.

Para células muito pequenas como uma porta OU CMOS, ou para circuitos com poucos transistores e muitas ligações, este método pode levar a conclusões erradas. Em circuitos hierárquicos compostos de várias células este problema é reduzido, verificando as ligações entre as células. Para cada célula, junto com a tentativa de identificação da alimentação, é fornecida uma estimativa do grau de certeza desta identificação. Quando se verifica a ligação entre duas células, caso a identificação da alimentação estiver de acordo, o grau de certeza é aumentado. Se for encontrada uma contradição na identificação da alimentação das duas células, a que tiver o menor grau de certeza será corrigida, e o grau de certeza é diminuido. Este método permite encontrar os nós de alimentação em circuitos hierárquicos grandes com uma probabilidade de erro muito pequena.

5.3 Método estatístico para NMOS

Nas tecnologias NMOS a localização da alimentação é feita de forma semelhante, com base nos transistores de carga. As regras usadas são as seguites:

a) Nos transistores de carga tipo depleção, existe ligação entre o gate e a fonte; o dreno então será ligado a VCC.

b) GND normalmente é o nó que tem o maior número de ligações com a fonte de transistores ativos, e não tem ligação com os transistores de carga.

Nos circuitos NMOS são usadas as mesmas técnicas para evitar erros na identificação do nó de terra na extração de circuitos hierárquicos que em CMOS. Em NMOS o nó de alimentação Vdd pode ser identificado com toda a certeza através dos transistores de carga.

5.4 Através da anotação no layout das máscaras

O editor de máscaras EMA permite escrever trechos de texto sobre o layout. Estes textos são usados pelo extrator para associar locais do layout com nós do netlist. Desta forma o extrator pode dar nomes aos nós do netlist, de acordo com os nomes dados aos locais correspondentes do layout do circuito. Os nomes de nós VDD e GND tem significado especial para o extrator. As regiões conexas do circuito que tem ligação com os locais onde foram colocadas as anotações VDD e GND serão consideradas pelo extrator como sendo os nós de alimentação correspondentes.

6 EXTRAÇÃO DAS RESISTÊNCIAS E CAPACITÂNCIAS

6.1 introdução

Quando se quer calcular as formas de onda detalhadas e os tempos de propagação de sinais com precisão, nas partes críticas de circuitos digitais de alto desempenho, ou em circuitos integrados analógicos, é indispensável que as capacitâncias e as resistências das linhas de roteamento sejam calculadas.

As capacitâncias parasíticas podem ser calculadas com boa precisão através do modelo de placas paralelas, compensando o efeito das bordas, usando o perímetro e a área da região de sobreposição [BAR 88]. A resistência é o parâmetro mais difícil de ser calculado. Ela depende não apenas da geometria da região, mas também do caminho seguido pela corrente. No caso de uma resistividade superficial constante, o problema recai na solução da equação de Laplace em duas dimensões.

Os algoritmos para o cálculo de resistências, em geral, são um compromisso entre a precisão dos resultados e a rapidez da solução. Os mais simples e rápidos se baseiam na relação comprimento / largura do caminho da corrente. O caminho da corrente é subdividido em retângulos, que são convertidos para resistências pela relação comprimento / largura dos retângulos. Estas resistências são então associadas para formar a malha final desejada. A precisão deste método pode ser melhorada com o uso de uma biblioteca de formas padrão, com resistências equivalentes conhecidas [McCOR 85]. Outros métodos de cálculo de resistências usam formas mais gerais, como transformações conformes, diferenças finitas ou elementos finitos.

O algoritmo usado no extrator EXTRIBO se baseia no método de elementos finitos. O método dos elementos finitos é um método de análise poderoso e geral, que consiste na substituição dos parâmetros distribuidos por elementos discretos finitos de comportamento aproximadamente equivalente. No caso presente, o parâmetro distribuido é a resistividade do condutor, que é modelada por uma rede de resistores discretos. O método de elementos finitos tem sido um dos mais usados atualmente [MIT 87]. Este método consiste em criar uma malha de resistores sobre a area do condutor. Esta malha é modelada através de uma matriz de admitâncias cuja dimensão é igual ao número de nós da rede. Esta matriz é então reduzida a uma equivalente envolvendo apenas os terminais dos resistores. Como esta matriz de admitâncias pode atingir dimensões muito grandes, este método exige grande capacidade computacional e é limitado a pequenas regiões do circuito. No estudo aqui apresentado se procurou adaptar o método dos elementos finitos para que ele se torne suficientemente econômico para poder ser usado até em microcomputadores como o PC, em que o algoritmo foi implementado.

No método apresentado aqui foi criada uma forma simples, sistemática e eficiente de formar uma rede de resistores que modela a condutividade superficial da região em estudo. Esta malha resistiva vai sendo simplificada a medida em que vai se formando, de modo que os requisitos de armazenamento e tempo de computação são drásticamente reduzidos. O resultado é um modelo de resistências multiportas na forma de uma matriz de admitâncias. Esta forma de atacar o problema permite obter as resistências de forma completamente geral, sem precisar considerar condições de contorno complicadas.

As resistências em um circuito integrado têm basicamente dois efeitos a serem considerados. Em regime permanente, quando as variações de tensão são bastante lentas para que se possa desprezar os efeitos das capacitâncias, o efeito das resistências consiste na degradação dos níveis de

tensão nos caminhos por onde passa corrente e na limitação das correntes de curto circuito. Estes efeitos são desprezíveis em circuitos em tecnologia CMOS, devido às baixíssimas correntes em regime permanente envolvidas. O outro efeito das resistências no comportamento do circuito se relaciona com o tempo de carga das capacitâncias. Este é o efeito predominante em circuitos MOS. A extração das capacitâncias parasitas envolve o problema de se modelar parâmetros distribuídos por elementos discretos. Não existe uma representação exata, por um número finito de elementos discretos, para a resistência e a capacitância de uma região do circuito. Neste ponto novamente se faz necessário assumir um compromisso entre a precisão e a complexidade do modelo. No programa extrator de circuitos EXTRIBO, em que este método foi implementado, optou-se pela utilização de um modelo pi de capacitâncias distribuidas, em que a capacitância de uma determinada região é modelada por capacitores ligados aos nós terminais das resistencias. Nos locais onde a capacitância ultrapassa determinado valor, um novo nó com um capacitor é criado. Este tipo de modelo deve apresentar resultados bastante precisos até para a avaliação de circuitos de alto desempenho.

6.2 Matriz de admitâncias

O primeiro passo na obtenção de um modelo das resistências de um circuito consiste em determinar quais serão as regiões envolvidas e quais serão os terminais dos resistores. A região a ser analizada para o cálculo de resistências deve ser uma região conexa de um mesmo nível. Regiões não conexas entre si devem ter suas resistências calculadas independentemente, já que elas não se relacionam. As regiões devem pertencer a um mesmo nível, pois os contatos de um nível para outro exigem um tratamento especial. Os terminais serão todos os pontos da região considerada por onde pode entrar ou sair corrente elétrica. Tipicamente estes pontos são os contatos com outros níveis e

como transistores ou capacitores. Portanto a descrição da geometria da região onde será avaliada a resistência e a lista dos pontos terminais dos resistores devem ser obtidas como uma etapa preliminar do programa.

De maneira geral, a resistência da região com N terminais será modelada através de resistores conectados em todas as N.(N-1)/2 combinações 2 a 2 de terminais. As relações entre correntes e tensões nos terminais também podem ser modeladas como uma matriz de admitâncias, pela equação matricial:

$$\mathbf{i}_t = \mathbf{Y}_a . \phi_t \qquad (6.1)$$

onde:

$\mathbf{i}_t$ = vetor das correntes que entram pelos nós terminais,
$\mathbf{Y}_a$ = matriz de admitâncias,
ϕ_t = vetor das tensões nos nós terminais.

Para uma região com tres terminais, a matriz $\mathbf{Y}_a$ se relaciona diretamente com as resistências entre pares de terminais por:

$$\mathbf{Y}_a = \begin{bmatrix} G_{12}+G_{13} & -G_{12} & -G_{13} \\ -G_{21} & G_{21}+G_{23} & -G_{23} \\ -G_{31} & -G_{32} & G_{31}+G_{32} \end{bmatrix} \qquad (6.2)$$

onde G_{ij} são as condutâncias entre os nós terminais.

O método aqui apresentado, desta forma, obtém a matriz $\mathbf{Y}_a$ a partir da geometria da região de resistência superficial e da lista de nós terminais.

6.3 Modelo de resistência

É necessário inicialmente obter um modelo para uma resistência laminar de forma poligonal arbitrária. Assumem-se as seguintes hipóteses simplificativas:

i) Sendo E o campo elétrico, σ a condutividade superficial e J a densidade siperficial de corrente, então dentro do condutor:

$$J = \sigma E \qquad (6.3)$$

onde J e E são funções bidimensionais de x e y, com componentes apenas ao longo de x e y. Considera-se que a condutividade é a mesma em qualquer direção.

ii) Não há fluxo de corrente para fora dos contornos do condutor a não ser nos pontos de contato.

iii) A região de resistência superficial considerada tem contornos retangulares, horizontais ou verticais, podendo ser representada por um conjunto de retângulos (geometria tipo Manhattan).

iv) As posições e dimensões dos retângulos usados para descrever a geometria da resistência laminar podem ser representadas por números inteiros, variando portanto a incrementos discretos de comprimento λ.

Estas hipóteses não chegam a comprometer a precisão do método nem a limitar seriamente a sua generalidade. O processo para o cálculo de resistências foi criado com o objetivo de ser incorporado a um extrator com geometria tipo Manhattan com coordenadas discretas.

A rede proposta aqui, mostrada na figura (6.1) é formada por resistores em posição diagonal em relação aos contornos das máscaras unindo pontos de rede localizados dentro do condutor nos pontos em que a soma das coordenadas x e y é par (ou ímpar). Cada um dos resistores desta rede tem uma resistência $R=1/\sigma$. Em uma região retangular de condutividade superficial σ, comprimento L e largura W onde a

corrente é paralela à borda de comprimento L, a resistência equivalente é dada por:

$$R = \frac{L}{\sigma . W} \qquad (6.4)$$

A malha de resistores usada satisfaz esta relação desde que as dimensões do retângulo se ajustem exatamente na malha. Isto acontece quando as dimensões do retângulo são múltiplas da distância entre os nós da malha.

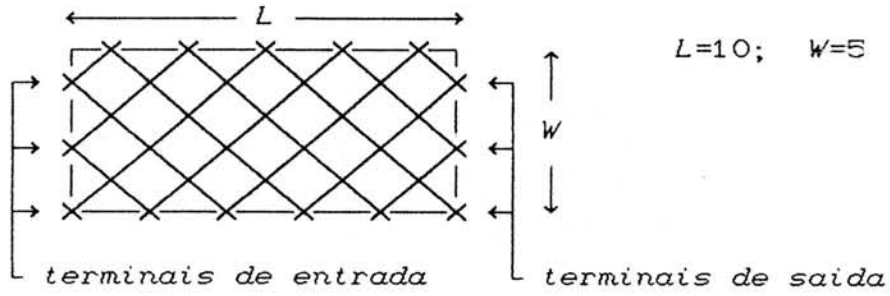
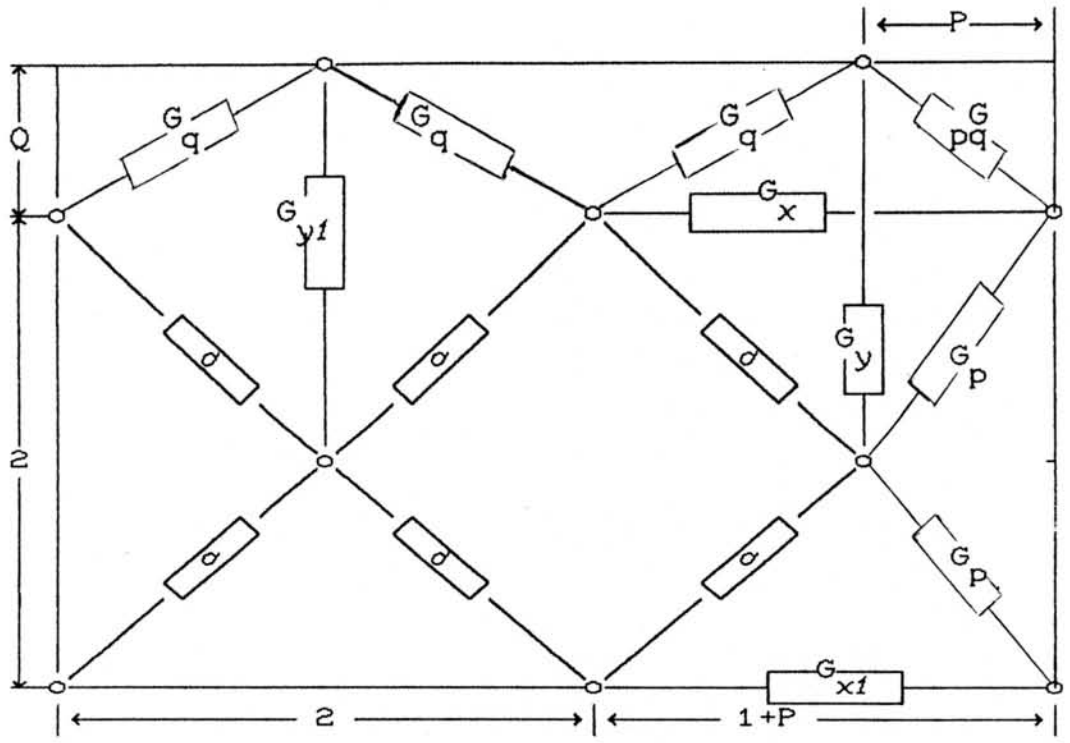


Figura 6.1 A malha de resistores sobre um retângulo.

Na figura 6.1, as linhas diagonais representam os resistores de resistência $1/\sigma$ da malha. Pode-se observar que entre os terminais de entrada e os de saída existem 10 resistores em série (número igual ao comprimento do retângulo), e 5 resistores em paralelo (número igual à largura). Isto significa que a resistência equivalente da malha será a mesma que a obtida pela equação (6.4). Em situações em que a corrente não é paralela a uma das bordas do retângulo, ou em configurações complexas de retângulos, este tipo de rede fornece resultados aproximados. Quanto mais fina a malha melhor a aproximação.

Nos casos em que a condição iv não pode ser satisfeita, i. e., as dimensões dos retângulos não sao múltiplos exatos do passo da malha, se pode usar uma configuração de resistências especial nas bordas destes retângulos de modo a preservar a regra $R = L/\sigma W$, para correntes paralelas a qualquer uma das bordas. As configurações a serem usadas são mostradas na figura (6.2).



Fiura 6.2 *Malha de resistências ao longo da borda de um retângulo de dimensões fracionárias.*

O retângulo da figura tem dimensões 3+P na direção x e 2+Q na direção Y. Neste retângulo é ajustada uma rede de passo 1. As dimensões P e Q são as frações de passo de rede excedentes em X e em Y. As frações de passo de rede P e Q nas dimensões do retângulo requerem uma modificação da rede junto à borda, para que as relações $R = L / \sigma.W$ continuem sendo válidas tanto na direção x, como em y. A rede da figura 6.2 satisfaz estas condições com os seguintes valores de condutâncias:

$$G_p = P.\sigma \quad (6.5a) \qquad G_q = Q.\sigma \quad (6.5b) \qquad G_{pq} = P.Q.\sigma \quad (6.5c)$$

$$G_x = \frac{(1 + Q)(1 - P)}{(1 + P)}.\sigma \quad (6.5d) \qquad G_y = \frac{(1 + P)(1 - Q)}{(1 + Q)}.\sigma \quad (6.5e)$$

$$G_{x1} = \frac{2.(1 - P)}{(1 + P)}.\sigma \quad (6.5f) \qquad G_{y1} = \frac{2.(1 - Q)}{(1 + Q)}.\sigma \quad (6.5g)$$

Esta configuração permite que se adote malhas que não se adaptam exatamente às dimensões do retângulo, compensando a parte fracionária que sobra com resistores especiais. Desta forma as dimensões dos retângulos podem também ser fracionárias e representadas em ponto flutuante.

O tamanho ideal para a malha deve ser determinado em função da complexidade da região em estudo. Uma fórmula heurística adotada no extrator EXTRIBO consiste em fazer com que o passo da malha seja proporcional à relação entre a área e o perímetro da região. Desta forma a malha se torna mais fina para regiões mais complexas. Aumentando a escala, o passo da malha aumenta da mesma maneira, mantendo a precisão e o tempo de computação.

$$\lambda = \frac{\text{área}}{4 * \text{perímetro}} \qquad (6.6)$$

Quando se adota esta fórmula, normalmente a malha não se ajusta perfeitamente nos contornos da região. Isto exige que as fórmulas especiais para as bordas dos retângulos de dimensões fracionárias sejam necessárias.

A malha de resistores é formada por um conjunto de nós, ligados aos seus vizinhos pelos resistores de resistência $1/\sigma$. O efeito de cada um destes nós é incluido na matriz Y como um acréscimo de um em sua dimensão. Um resistor entre os nós i e j terá os seguintes efeitos na matriz Y:

$$y_{ij} := y_{ij} - \sigma_{ij} \; ; \qquad (6.7a) \qquad y_{ji} := y_{ji} - \sigma_{ij} \; ; \qquad (6.7b)$$

$$y_{ii} := y_{ii} + \sigma_{ij} \; ; \qquad (6.7c) \qquad y_{jj} := y_{jj} + \sigma_{ij} \; ; \qquad (6.7d)$$

A formação da malha de resistores, portanto provocará um crescimento muito grande da matriz Y. Uma matriz Y grande exige grandes quantidades de memória para ser armazenada e grandes esforços computacionais para ser processada. É necessário, portanto limitar o seu crescimento. Como a matriz **Y** é simétrica, pode-se armazenar apenas a triangular superior ou inferior.

6.4 Redução da matriz Y

Ao final dos cálculos apenas os nós terminais devem participar da matriz Y. Os nós intermediários, criados com a malha de resistores devem ser eliminados, reduzindo a matriz Y a uma equivalente. A eliminação dos nós intermediários deve ser feita sempre que for possível, para manter limitada a dimensão da matriz.

A rede resistiva completa, incluindo os nós intermediários é dada por uma equação matricial do tipo:

$$\mathbf{i} = \mathbf{Y}.\ \phi \qquad (6.8)$$

Esta equação pode ser decomposta, por partição de matrizes, separando os nós terminais dos nós intermediários. Os nós intermediários terão tensões ϕ_i. A soma das correntes que entram nos nós intermediários é zero. Com uma partição deste tipo a equação (6.8) poderá ser reescrita como:

$$\begin{bmatrix} \mathbf{i}_t \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{21} & \mathbf{Y}_{22} \end{bmatrix} \begin{bmatrix} \phi_t \\ \phi_i \end{bmatrix} \qquad (6.9)$$

Através da eliminação dos nós intermediários se pretende obter a equação (6.1), que envolve apenas os nós terminais. Por partição de matrizes aplicada sobre a equação (6.9) vem que:

$$\mathbf{i}_t = (\ \mathbf{Y}_{11} - \mathbf{Y}_{12}.\mathbf{Y}_{22}^{-1}.\mathbf{Y}_{21}).\phi_t \qquad (6.10)$$

Portanto:

$$\mathbf{Y}_a = \mathbf{Y}_{11} - \mathbf{Y}_{12}.\mathbf{Y}_{22}^{-1}.\mathbf{Y}_{21} \qquad (6.11)$$

A fórmula (6.11) permite obter uma matriz equivalente $\mathbf{Y}_a$, eliminando os nós intermediários ϕ_i. Um caso particularmente interessante para o nosso estudo é aquele em que se quer eliminar apenas um nó de cada vez. Neste caso, a equação matricial (6.11) pode ser interpretada como uma operação sobre os elementos que elimina uma linha e uma

quando se quer eliminar a linha e a coluna k, ve-se que:

$$y_{aij} = y_{ij} - \frac{y_{ik} \cdot y_{kj}}{y_{kk}} \qquad (6.12)$$

Depois de feita a operação (6.12) sobre todos os elementos da matriz **Y**, a linha k e a coluna k deixam de existir. Para que isto possa ser interpretado como uma redução da dimensão da **Y**, é necessário que a última linha e coluna da matriz sejam eliminadas. Se a linha e coluna k não forem as últimas, se faz necessária uma renumeração dos elementos da matriz, trocando a última linha e coluna com a linha e coluna k. Na matriz **Y**, isto é feito copiando os elementos da última linha e coluna n sobre a linha e coluna k, e colocando y_{nn} no lugar do y_{kk}. Feito isto, a linha n e a coluna n estarão liberadas, permitindo que a dimensão da matriz seja reduzida. A aplicação da fórmula (6.12) sobre a matriz pode ser mais eficiente com o aproveitamento da simetria da **Y** e não operando sobre as linhas em que $y_{ik} = 0$.

Como a matriz **Y** normalmente é bastante esparsa, apesar das constantes reduções de dimensão obtidas através das eliminações de nós, técnicas de armazenamento apenas dos elementos não nulos podem resultar em uma considerável economia de memória e aumento de desempenho.

6.5 Propagação da malha de resistores

Uma parte essencial deste algoritmo para o cálculo de resistências por elementos finitos consiste em gerar a malha descrita no ítem (6.3) de forma simples e sistemática. A rede deve crescer de modo a facilitar a eliminação de nós com o máximo de freqüência, para evitar que a matriz se torne muito grande. Um nó pode ser eliminado quando todas as ligações deste nó com os seus vizinhos já foram incluidas na matriz **Y**. Outro aspecto que deve ser considerado para a formação da rede é que ela deve ficar limitada à região do condutor onde está sendo calculada a resistência, não ultra-

passando os seus contornos, mas preenchendo todo o condutor.

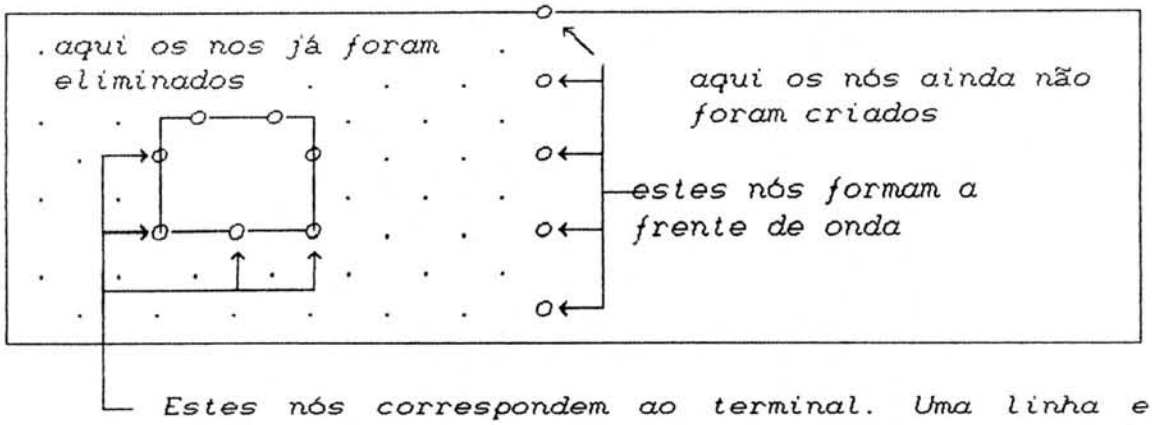


Figura 6.3 - A propagação dos nós, formando uma frente de onda em uma região retangular com um contato. Os nós ativos, mantidos na memória são mostrados como "o". Os nós que já foram eliminados aparecem como ".". Os nós que ainda não foram criados não aparecem na figura.

A estratégia usada consiste em manter uma estrutura de dados com as seguites informações para cada nó da rede:

NUMERO = Número do nó
TIPO = Indica se o nó tem contato com um terminal.
X, Y = Coordenadas do nó
V00 = número do vizinho à esquerda abaixo
V01 = numero do vizinho à direita abaixo
V10 = número do vizinho à esquerda acima
V11 = número do vizinho à direita acima
PRÓXIMO = Elo na lista encadeada

Os nós são mantidos em uma lista encadeada ou duplamente encadeada, para facilitar a criação de novos nós e a liberação dos nós eliminados. A estrutura informa o número do nó, que corresponde ao índice na matriz **Y**, as coordenadas x e y correspondentes à localização do nó no circuito e os números dos nós dos quatro vizinhos possíveis. A informação dos números dos vizinhos tem dupla finalidade: Informa qual é o nó vizinho, se ele existir e tiver sido determinado, permitindo assim identificar os terminais do

não existe vizinho naquela direção ou que aquele vizinho ainda não foi criado. Na versão implementada deste algoritmo, números de nós de vizinhos conhecidos e válidos são indicados por números positivos. Se o número do nó for zero, não existe vizinho naquela direção. Se o número for -1, o vizinho na direção correspondente ainda não foi criado.

A rotina de propagação da rede verifica cada nó a procura de vizinhos não determinados. Encontrando uma referência a vizinho com número -1, no nó que vamos chamar de nó de base, o programa verifica primeiro se as coordenadas correspondentes a este vizinho ficam dentro dos limites do condutor. No caso em que ele ficar fora do condutor, é colocada a indicação de nó inexistente (zero) na tabela de vizinhos do nó que está sendo verificado, e nenhum nó é criado. Se as coordenadas do vizinho do nó de base que está sendo verificado estão dentro dos contornos do condutor, um novo nó deve ser criado nestas coordenadas. A criação do nó implica na formação de mais uma linha e coluna na matriz **Y**. A tabela de nós vizinhos deste nó recém criado é preenchida através de uma pesquisa ao longo de todos os nós da lista, à procura de nós cuja distância tanto em x como em y até o novo nó seja de 1λ. A cada vizinhança encontrada desta maneira, deve ser incluido um resistor da rede na matriz **Y**, de acordo com as fórmulas de inclusão de resistores (6.7a), (6.7b), (6.7c), e (6.7d).

Quando todas as vizinhanças do nó de base tiverem sido verificadas não haverá mais nenhuma ligação que ainda possa ser feita a este nó. Então o nó poderá ser eliminado da estrutura de dados que especifica as coordenadas e vizinhanças da lista de nós. Quando um nó é eliminado, a matriz **Y** pode ser reduzida, aplicando a equação (6.12) sobre ela.

A propagação da rede de resistores, feita desta maneira faz com que sejam mantidos na memória apenas aqueles nós que formam a frente de onda. O resultado disto é que o número de nós, e portanto a dimensão da matriz **Y**, pára de crescer depois que a largura dos caminhos de corrente é

preenchida. A rede então vai crescendo através da criação de novos nós, e apagamento dos velhos. O processo estará terminado quando todos os nós criados pelo módulo de propagação da rede tiverem sido eliminados. Quando isto acontecer, a matriz **Y** resultante será a matriz $\mathbf{Y}_{\alpha}$, que relaciona as tensões nos terminais com as suas correntes. Esta matriz pode por sua vez ser interpretada como um conjunto de resistores equivalentes de dois terminais, conforme foi mostrado na equação (6.2).

Quando um nó criado durante a propagação da rede atinge um terminal, este nó apresenta características especiais. O número deste nó será o número do terminal. Este número corresponde a um dos índices iniciais da matriz **Y**, que permanecerá até o fim da propagação da malha. Quando este nó for eliminado, a matriz **Y** não será reduzida. Outros nós que tiverem contato com o mesmo terminal, terão o mesmo índice na matriz.

6.6 Preparação da lista de terminais e da lista de retângulos que formam o condutor

O módulo de cálculo de resistências recebe como dado de entrada a geometria da região onde será calculada a resistência na forma de uma lista de retângulos, e a localização dos terminais, também como uma lista de retângulos. Estas listas são obtidas por um levantamento prévio de conectividade pelo extrator.

Para que o algoritmo seja aplicado de forma eficiente, convém que a região onde está sendo avaliada a resistência seja conexa. O cálculo das resistências em regiões não conexas é feito de forma muito mais eficiente em etapas independentes, segundo a estratégia de dividir para conquistar. Portanto o extrator, para calcular as resistências de modo eficiente deve seguir as etapas abaixo:

a) Avaliar a conectividade para os níveis em que serão calculadas as resistências, criando uma lista de retângulos descrevendo a geometria de cada uma das regiões conexas. Quando dois ou mais níveis com resistividades definidas são interligados, eles podem ser incluídos na mesma região para o cálculo de resistências.

b) Encontrar os pontos em que serão formados os terminais da resistência multiportas de cada região conexa. Estes pontos poderão ser contatos com outros níveis que não tem resistividade definida, terminais de componentes ativos do circuito ou terminais de capacitores. No caso da extração herárquica, os retângulos que formam interfaces externas também devem ser usados como terminais. Os transistores exigem um tratamento especial, para que os seus terminais possam ser tratados como terminais de resistores, conforme foi visto no capítulo 3.

c) Calcular uma matriz $\mathbf{Y}_{\alpha}$, pelo método de elementos finitos descrito aqui, para cada região conexa, obtendo a representação equivalente por resistores de dois terminais. Os terminais dos resistores são identificados através de referências aos retângulos usados para formar a lista dos terminais da resistência multiportas.

d) Avaliar a conectividade inter níveis incluindo nesta etapa a resistência de contato. As regiões onde foram calculadas as resistências não devem mais serem consideradas interligadas. Isto foi feito marcando os retângulo das regiões com resistências, e excluindo estes retângulos marcados na avaliação da conectividade.

6.7 Simplificação da malha de resistores resultante

O algoritmo descrito aqui gera, para uma região com N terminais, resistores entre cada uma das N*(N-1)/2 combinações 2 a 2 de terminais. Muitas vezes, nem todas estas combinações devem ser ligadas por resistores. Um caso típico é o que está mostrado na figura 6.5 a seguir:

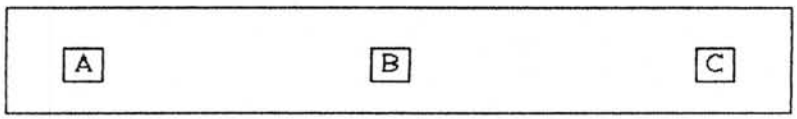


Figura 6.4 (a) Padrão de teste para demonstrar que nem sempre existe um resistor entre todas as combinações 2 a 2 de terminais

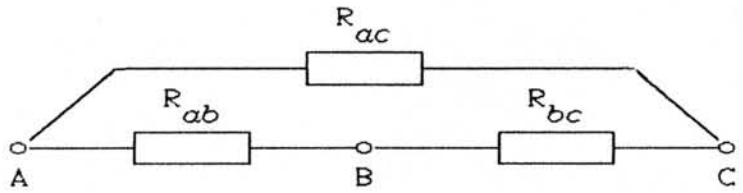


Figura 6.4 (b) Malha de resistores resultante. A ligação entre os nós A e C através do resistor R_{ac} na verdade não existe.

No caso da figura 6.5, por causa da simetria, a admitância entre os nós A e C é zero. Simplesmente não existe ligação direta entre A e C. Em outros casos também ocorre que alguns elementos da matriz Y resultante do cálculo das resistências não são exatamente zero, mas são muito menores que os demais elementos da Y. Estes elementos também não são considerados. O critério usado no extrator EXTRIBO foi de não considerar os elementos que sejam menores que um centésimo da média dos elementos da matriz Y. O valor de um centésimo se justifica pelo fato de que a margem de erro nos valores das resistências é da ordem de 1%. No caso da figura 6.5, o extrator incluiria no *netlist* resultante apenas os resistores R_{ab} e R_{bc}.

Ao final da extração do circuito com resistências, depois de avaliada a conectividade global do circuito, pode resultar que algumas resistências fiquem ligadas em paralelo, entre o mesmo par de terminais. Existe para isto um trecho no final da extração que reduz estas resistências em paralelo a uma equivalente.

6.8 Resultados dos testes

Foram feitos alguns testes para verificar a precisão e a eficiência do método de calculo de resistências. O primeiro padrão usado foi um único retângulo com dois contatos, conforme aparece na figura 6.5. Por causa da simplicidade deste padrão de teste, pode-se usar uma malha extremamente fina, aproveitando a simetria, para obter um valor de resistência considerado "exato" para fins de avaliação da precisão do método.

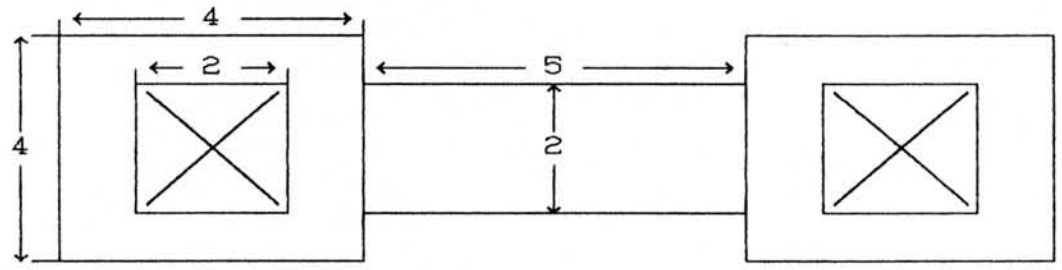


Figura 6.5: Padrão de teste usado para avaliar a precisão do método.

A tabela a seguir mostra os resultados dos testes feitos sobre esta figura. Por extrapolação dos resultados obtidos concluiu-se que a resistência "exata" deve ser de 3,269. Na tabela é mostrado o valor da resistência obtida, o erro relativo (em relação ao valor extrapolado), o tempo de computação em segundos e o número de nós da rede usada. A cada linha da tabela o passo da rede foi reduzido para a metade.

Passo da rede	Nº de nós	Resistência	Erro	Tempo
1	34	2,8333	13,33%	0,2s
1/2	118	3,1618	3,28%	0,8s
1/4	381	3,2239	1,38%	5,9s
1/8	1315	3,2506	0,563%	42,3s
1/16	1227	3,2615	0,23%	36,4s

O teste foi feito com um computador tipo PC XT rodando a 4,77 MHz, com coprocessador 8087. A última linha da tabela foi feita sobre apenas um quarto da figura de teste, aproveitando a simetria. A isto se deve o fato de para o passo de 1/8 ter dado tempo de computação e número de nós próximo ao que deu para o passo de 1/16. Em relação aos tempos de computação, a tabela demonstra que o tempo é aproximadamente proporcional ao quadrado do número de nós da rede $O(N^2)$, conforme foi previsto teoricamente. Nos métodos de elementos finitos convencionais, o tempo de computação é de $O(N^3)$.

Os resultados obtidos nos testes se comparam de maneira muito favorável com outros métodos para calcular resistências, tanto em precisão como em tempo de computação. [KEMP 87]

6.9 Cálculo das capacitâncias

As capacitâncias são calculadas com base no perímetro e na área de cada região. São calculadas apenas as capacitâncias verticais na versão atual do extrator EXTRIBO. As capacitâncias que acoplam horizontalmente linhas paralelas próximas, apesar de também terem efeitos consideráveis, não são calculadas, porque este tipo de cálculo seria muito complexo.

O extrator EXTRIBO calcula as capacitâncias nos seguintes casos:

6.9.1 Capacitância contra o substrato de regiões sem resistência:

-Certas regiões com baixa resistividade, desde que não tenham caminhos muito estreitos e longos, são consideradas pelo EXTRIBO como sendo diretamente interligadas. A resistência abaixo da qual o extrator considera como ligação direta é especificada no arquivo de descrição da tecnologia. Nestas regiões sem resistência, o cálculo das capacitâncias contra o substrato é muito simples:

$$C = C_p.\text{perímetro}.\lambda + C_a.\text{área}.\lambda^2$$

O perímetro e a área são calculados em unidades de λ. Para convertê-los para unidades métricas existem as multiplicações por λ e por λ^2. C_p é a capacitância por unidade de perímetro e C_α é a capacitância por unidade de área. Esta fórmula é usada para calcular a capacitância correspondente a cada número de nó. Caso o valor calculado fique abaixo de um mínimo especificado na descrição da tecnologia, ela não será considerada.

6.9.2 Capacitâncias internodais entre dois níveis

As capacitâncias internodais entre um nível A e um nível B são calculadas desde que sejam satisfeitas as seguintes condições. A capacitância internodal por unidade de área deve ter sido definida no arquivo de descrição da tecnologia, as regiões de ambos os níveis não devem ter resistência e deve haver uma área mínima de sobreposição entre estas regiões. A capacitância internodal será proporcional à área da sobreposição.

6.9.3 Capacitâncias nas regiões com resistência

Nas regiões com resistência, a capacitância distribuída faz com que o caminho de corrente se comporte como uma linha de transmissão. Os formatos dos *netlists* usados não permitem descrever com precisão este comportamento. A solução usada é modelar esta situação de forma aproximada, usando um modelo π, como mostra a figura 6.6.

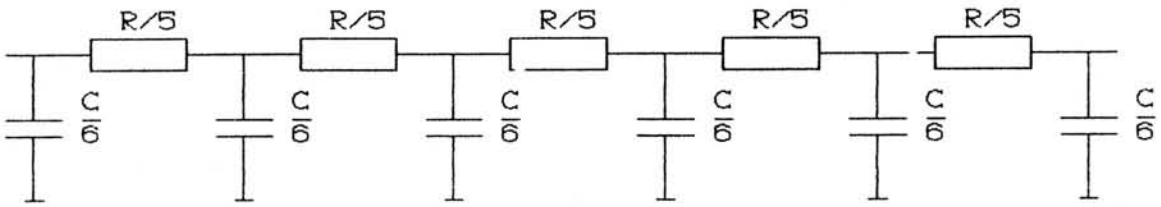


Figura 6.6 *a* Uma linha com resistências e capacitâncias

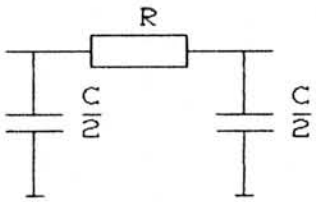


Figura 6.6 *b* Aproximação usando um modelo π.

A capacitância da região com resistência é calculada normalmente, com base no perímetro e na área da região. A capacitância resultante é então dividida pelo número de terminais e acrescentada a cada um deles.

Para cada retângulo com resistência o programa verifica antes de calcular as resistências se a capacitância deste retângulo fica acima de um valor mínimo especificado no arquivo de descrição da tecnologia. Em caso positivo, este retângulo passa a ser tratado como um termimal. Desta forma podem ser criados nós especiais para acomodar as capacitâncias que forem consideradas relevantes (ver figuras 6.7 e 6.8).

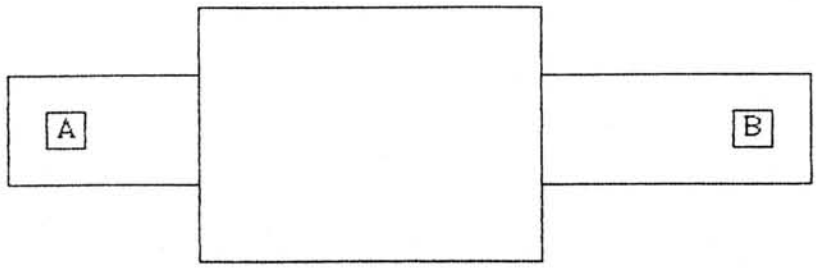


Figura 6.7 Padrão de retângulos que faz com que o extrator crie um nó especial para acomodar a capacitância.

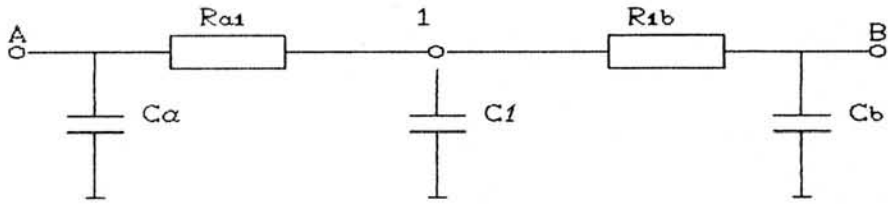


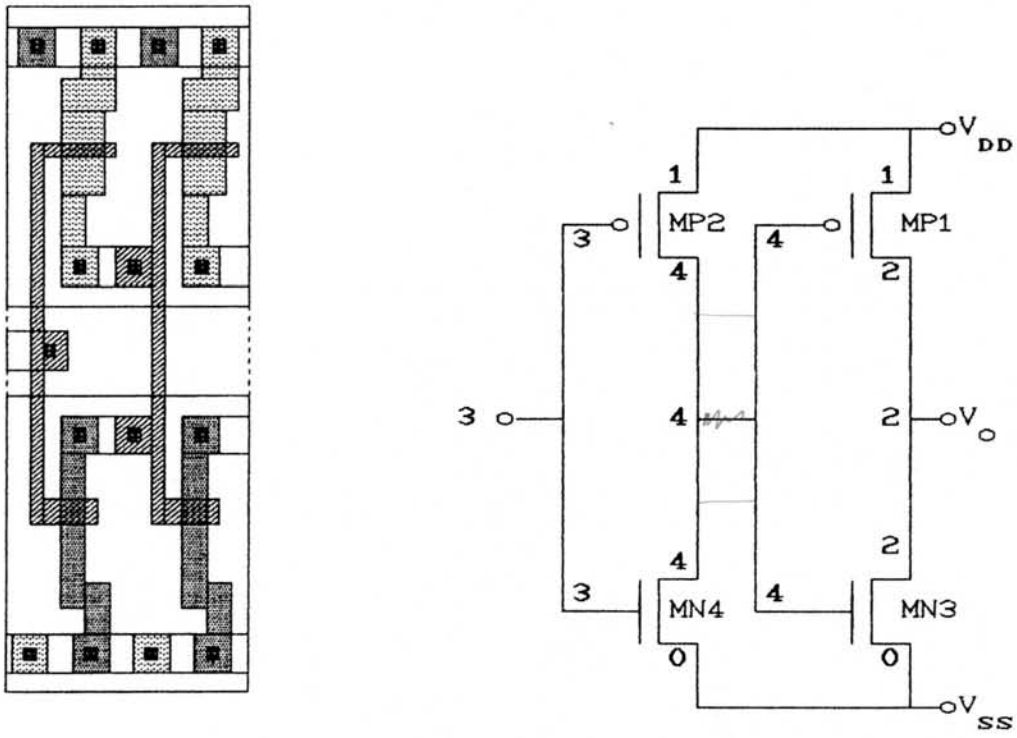
Figura 6.9 Circuito resultante, com o nó criado especialmente para incluir o capacitor C_1.

6.10 Exemplo de simulação com resistências e capacitâncias

O exemplo a seguir mostra os efeitos das resistências e capacitâncias no resultado da simulação. O circuito usado neste teste é uma seqüência de dois inversores em tecnologia CMOS. O algoritmo de cálculo de resistências, implementado no extrator EXTRIBO foi usado para calcular as resistências nos caminhos de difusão e de polisilício. As partes de metal, por terem resistências muito baixas, foram consideradas como condutores ideais.

Para verificar qual é o efeito das resistências e capacitâncias nas formas de onda, foram feitas duas simulações do circuito. Uma permitindo que o próprio SPICE faça uma estimativa das resistências e capacitâncias com base nas informações de perímetro e área das regiões de dreno e fonte dos transistores, e a outra usando o extrator para calcular

A figura 6.10a mostra o layout do circuito usado no teste. O netlist gerado pelo extrator sem calcular resistencias e capacitâncias é mostrado na figura 6.10b. O esquema do circuito correspondente a este netlist é a figura 6.11. O netlist gerado pelo extrator foi usado no simulador SPICE para obter a resposta na saída do inversor duplo a uma onda quadrada aplicada na entrada. O resultado desta simulação, usando o netlist sem resistências e capacitâncias é mostrado na figura 6.12.



(a) (b)

Figura 6.9 (a) Layout do duplo inversor CMOS usado no teste do cálculo das resistências e capacitâncias.

Figura 6.9 (b) Desenho esquemático correspondente ao netlist sem resistências e capacitâncias.

```
* UFRGS PGCC GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CIRCITO: inv2.cel
* TECNOLOGIA: gren2.tec  TIPO CMOS
*
* CIRCUITO PRINCIPAL
*
* Transistores tipo NMOS: 2
* Transistores tipo PMOS: 2
*
* Nomes dos nos:
*
MP1   2   4   1   1 PMOS L=2.1U W=7.5U AD=120P AS=119P PD=60U PS=52U
MP2   4   3   1   1 PMOS L=2.1U W=7.5U AD=120P AS=119P PD=60U PS=52U
MN3   2   4   0   0 NMOS L=4.2U W=4.3U AD=64P AS=120P PD=38U PS=66U
MN4   4   3   0   0 NMOS L=4.2U W=4.3U AD=64P AS=120P PD=38U PS=66U
.END
```

Figura 6.10 Netlist do inversor duplo gerado pelo extrator sem o cálculo de resistências e capacitâncias.

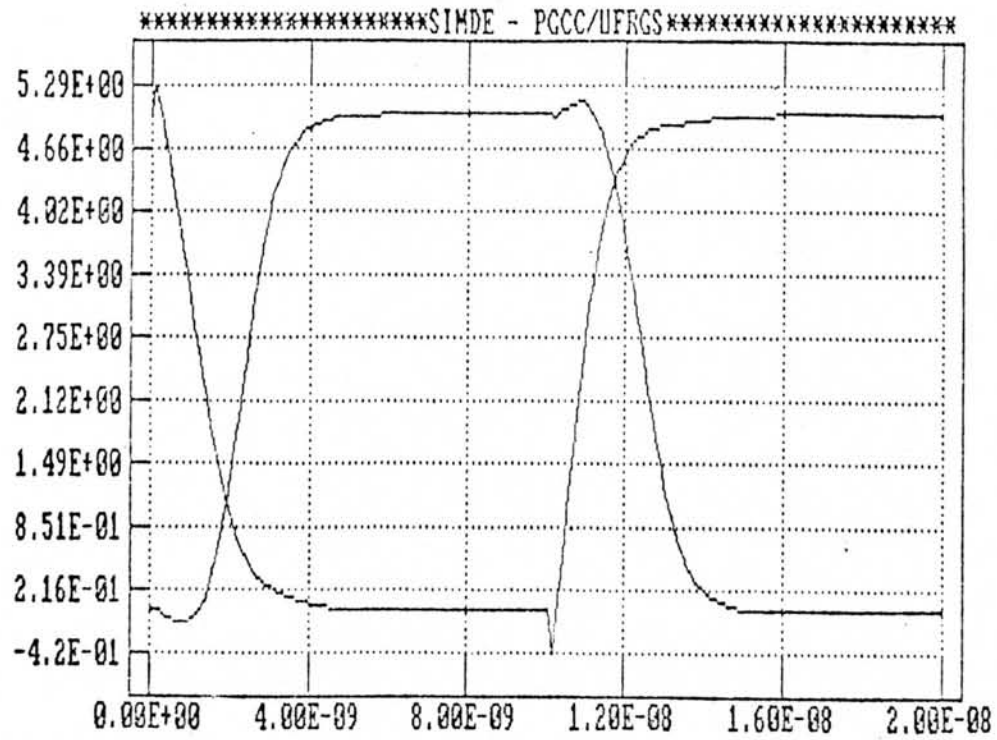


Figura 6.11 Resultado da simulação do circuito sem o cálculo de resistências. As resistências e capacitâncias são estimadas com base no perímetro e área das regiões de dreno e fonte dos transistores.

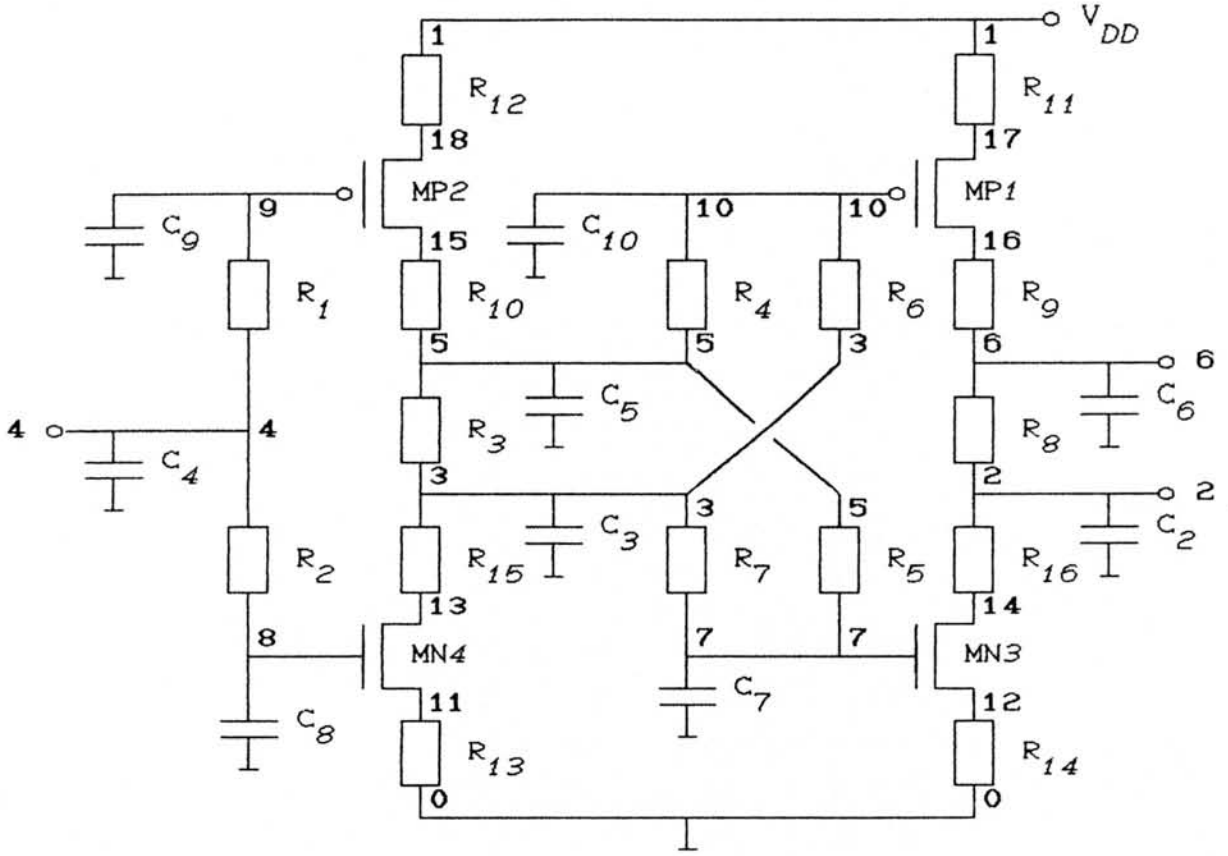


Figura 6.13 Desenho esquemático do circuito resultante da extração com resistências e capacitâncias.

Quando o extrator não calcula as resistências do nível de difusão, as áreas e perímetros da região de dreno e fonte dos transistores são calculadas. Isto permite ao simulador fazer uma estimativa dos valores das resistências e capacitâncias destas regiões. O objetivo do estudo feito com o inversor duplo CMOS foi verificar se existem diferenças consideráveis entre os resultados da simulação usando os valores precisos de resistências e capacitâncias calculados pelo extrator e os resultados da simulação quando o simulador estima os valores das resistências e capacitâncias com base somente no perímetro e na área das regiões de dreno e fonte dos transistores.

```
* UFRGS PGCC GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CELULA: inv2.cel
* TECNOLOGIA: gren2.tec  TIPO CMOS
*
* Modelos dos transistores
*
.MODEL NMOS NMOS LEVEL=2 LD=0.15U TOX=440E-10 NSUB=0.95E16
VTO=1.15 UO=693 UEXP=0.111 UCRIT=10K DELTA=1.68 XJ=1U VMAX=41K
+NEFF=1.16 RSH=45 NFS=1E11 JS=100U CJ=105U CJSW=240P MJ=0.48
+MJSW=0.27 PB=0.45V CGDO=270P CGSO=270P
.MODEL PMOS PMOS LEVEL=2 LD=0.25U TOX=440E-10 NSUB=3.24E16
+VTO=-0.8 UO=271 UEXP=0.131 UCRIT=10K DELTA=0.89 XJ=2U VMAX=32K
+NEFF=0.77 RSH=80 NFS=1E11 JS=100U CJ=330U CJSW=430P MJ=0.48
+MJSW=0.4 PB=1.04V CGDO=350P CGSO=350P
*
VCC 1 0 DC 5
MP1    16 10 17  1 PMOS L=2.5U W=7.1U
MP2    15  9 18  1 PMOS L=2.5U W=7.1U
MN3    12  7 14  0 NMOS L=4.3U W=4.2U
MN4    11  8 13  0 NMOS L=4.3U W=4.2U
R1      4   9     384.1945
R2      4   8     282.7746
R3      5   3     316.1861
R4      5  10     248.3995
R5      5   7    5169.8872
R6      3  10    9679.2207
R7      3   7     144.6652
R8      6   2     280.2668
R9      6  16     193.9380
R10     5  15     193.9380
R11     1  17     125.1184
R12     1  18     125.1184
R13     0  11     225.5308
R14     0  12     225.5308
R15     3  13     122.7246
R16     2  14     122.7246
C2      0   2   2.84F
C3      0   3   8.76F
C4      0   4   5.92F
C5      0   5   9.48F
C6      0   6   4.12F
C7      0   7   18.6F
C8      0   8   16.2F
C9      0   9   14.4F
C10     0  10   16.8F
*Estas linhas foram acrescentadas usando um editor de texto para
*especificar a fonte de sinal e requisitar o relatorio de saída.
VIN 4 0 PULSE(0 5 0 0 0 10NS 20NS)
.TRAN/OP 0.1NS 20NS
.PRINT TRAN V(3) V(2)
.END
```

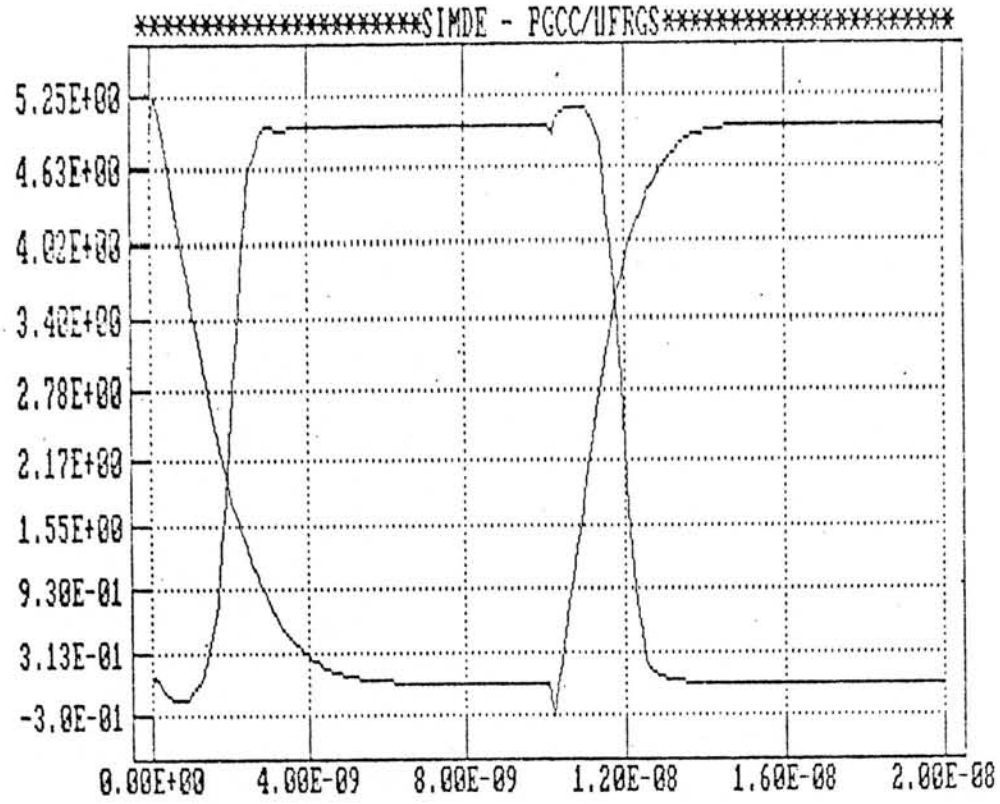


Figura 6.14 Resultado da simulação do circuito com resistências e capacitâncias.

O mesmo inversor duplo CMOS, extraído com resistências e capacitâncias, resultou no netlist da figura 6.12, que corresponde ao circuito mostrado na figura 6.13. Este netlistfoi simulado no SPICE, obtendo como resultado as curvas apresentadas na figura 6.14. Foi aplicada uma onda quadrada na entrada. As curvas correspondem às tensões na saída de cada um dos inversores.

Pelos resultados mostrados aqui se observa uma semelhança muito grande entre os resultados do cálculo com resistências e o cálculo em que o SPICE usa as áreas e perímetros para estimar os valores das resistências e capacitâncias. Outros testes mostraram que esta semelhança foi em grande parte coincidência. Através das informações de áreas e perímetros, porém se verificou que o SPICE consegue quase sempre resultados pelo menos semelhantes aos obtidos com o cálculo preciso das resistências e capacitâncias. O simulador ARAMOS, porém não faz qualquer estimativa dos

definidos no *net-list*. Isto faz com que os resultados obtidos com o cálculo das resistências e capacitâncias fique muito diferente dos obtidos só com os transistores. Por este motivo recomenda-se que quando se quer fazer estimativas da velocidade de operação de circuitos usando o simulador ARAMOS, se faça a extração do circuito com as resistências e capacitâncias.

7 EXTRAÇÃO HIERÁRQUICA

As máscaras dos circuitos integrados em geral, e principalmente os VLSI, apresentam freqüentemente regiões com estruturas repetidas, correspondentes a blocos lógicos iguais que são necessários em partes diferentes do circuito. Estes blocos lógicos podem serem descritos no layout como definições de células, e podem serem chamados em todos os locais do circuito onde forem necessários, como instâncias de células definidas anteriormente. Várias linguagens de descrição de circuitos aproveitam este conceito de descrição hierárquica para descrever tanto a geometria das máscaras como a lista de componentes. Como exemplo de linguagens de descrição hierárquica do layout temos EDIF, CIF [MEA 80], Lucie [JER 81] ou RS [JAC 86c], e para a lista de componentes temos SPICE [VLA 81] e ARAMOS [UCL 86].

O projeto de um circuito integrado VLSI pode ser feito de forma modular, definindo as funções básicas como células básicas, os blocos mais complexos como células definidas usando chamadas das células básicas, e o circuito inteiro como chamadas das células que formam os blocos. Do ponto de vista do projeto, a descrição hierárquica apresenta pelo menos duas grandes vantagens em relação a uma descrição plana. A divisão do circuito em blocos menores torna muito mais fácil o seu entendimento e projeto. Os blocos fundamentais se tornam fáceis de entender e de modificar por serem pequenos. O circuito inteiro ou os blocos maiores também podem ser analisados com facilidade, por estarem em um nível de abstração elevado. A outra vantagem do uso da hierarquia na descrição do circuito está no fato de que células repetidas só precisam serem definidas uma vez. Isto, além de reduzir o tamanho da descrição, poupando memória, também poupa trabalho no desenvolvimento e nas modificações do projeto e reduz as possíveis fontes de erro.

7.1 Descrição hierárquica do layout das máscaras

Para entender melhor como é feita a descrição hierárquica das máscaras, veremos como uma descrição é feita nas linguagens de descrição de layout CIF e RS. Descrições de circuitos nestas linguagens servem como dados de entrada para o extrator. O circuito é descrito nestas linguagens como um conjunto de definições de células. As definições de células podem serem feitas em termos das primitivas geométricas (retângulos) ou de chamadas de células.

As células são definidas como conjuntos de retângulos, através dos comandos de especificação de nível e definição de bloco. A definição da célula deve iniciar por um cabeçalho, (comando "DS" tanto em CIF como em RS), que é acompanhado de um número que serve como identificação á célula. Opcionalmente pode-se atribuir um nome à célula usando o comando "9". O final da definição de célula é marcado pelo comando "DF".

A sintaxe da definição da célula em CIF é a seguinte:

DS <número da célula>;
[9 <nome>;]
<especificações de níveis (LM), retângulos (B) ou chamadas de células>
DF;

A sintaxe da definição de célula em RS é idêntica, exceto pelo fato de o nome aparecer entre aspas:

DS <número da célula>;
[9 "*<nome>*";]
<especificações de níveis (LM), retângulos (B) ou chamadas de células>
DF;

As células definidas desta forma podem serem usadas nas definições subseqüentes através de chamadas de células. Nas chamadas podem serem feitas transformações de translação, rotação ou espelhamento. Na linguagem RS, existe também uma diretiva de repetição, que permite em uma chamada especificar arranjos unidimensionais ou bidimensionais de instâncias da célula. A linguagem CIF não permite definir repetições em apenas uma chamada, de modo que arranjos de instâncias de uma célula tem que serem expandidos como várias chamadas.

Tanto em CIF como em RS, a chamada de uma célula tem a seguinte sintaxe:

C <numero da célula> [<transformações>];

As transformações podem serem as seguintes:

T <int: dx> <int: dy> - Translação: o inteiro dx é somado a todas as coordenadas x da célula que está sendo chamada, e dy é somado às coordenadas y.

Em RS temos as transformações de espelhamento MX e MY:

MX - Espelhamento no eixo X: as coordenadas X tem o sinal trocado.

MY - Espelhamento no eixo Y.

Em CIF as transformações de espelhamento são: M X e M Y (com um espaço no meio).

R <vetor> -Rotação (mesma sintaxe para CIF e RS). A rotação é especificada por um par de inteiros com o seguinte significado:

R 0 1 -Rotação de 90° em sentido antihorário.
R -1 0 -Rotação de 180°. O mesmo que MX MY.
R 0 -1 -Rotação de -90°.

As diretivas de repetição na chamada de célula são exclusivas do RS. As diretivas PX e PY podem serem combinadas para formar arranjos bidimensionais.

PX *<int: NumRep> <int: deslocamento>* -Repete a instância da célula NumRep vezes com espaçamento em X de *<deslocamento>*.

PY *<int: NumRep> <int: deslocamento>* -Repete a instância da célula NumRep vezes com espaçamento em Y de *<deslocamento>*.
Exemplo de chamada de célula:

```
C 45 MX R 0 1 T 200 678 PX 3 130;
```

A célula 45 é espelhada em X, girada 90° em sentido antihorário, transladada de 200λ em X e 678λ em Y, e são feitas 3 copias desta instância da célula 45, com deslocamentos de 130λ na coordenada X entre as copias. Em CIF são necessárias 3 chamadas de célula para obter o mesmo resultado:

```
C 45 M X R 0 1 T 200 678;
C 45 M X R 0 1 T 330 678;
C 45 M X R 0 1 T 460 678;
```

No formato interno usado para descrever o layout do circuito no extrator não são permitidas diretivas de repetição de chamadas de células. Isto é feito porque eletricamente cada chamada de um arranjo de chamadas de células aparece ligada a partes diferentes do circuito, não podendo, portanto serem tratadas da mesma maneira.

A descrição hierárquica das máscaras propicia várias vantagens na extração de um circuito. A subdivisão do circuito em blocos menores proporciona uma melhora em termos de tempo de computação, porque, como já foi visto no ítem 2.2, o tempo de computação é proporcional ao número de retângulos na potência 3/2. Isto significa que tem-se uma redução no tempo de computação proporcional à raiz quadrada do número de divisões. No final, o tempo de computação da

extração hierárquica sobre o de uma extração plana é muito menor. A extração só precisa ser feita uma vez nas células que aparecem repetidas na descrição hierárquica do circuito. Porém é na economia de memória que está a vantagem mais importante do uso de um extrator hierárquico. A extração de um circuito pode ser feita analisando uma célula de cada vez, mantendo as outras células em arquivos baseados em disco. Desta forma o tamanho máximo do circuito a ser extraido deixa de ser limitado pela capacidade de memória disponível no computador, o que nos computadores tipo PC era uma limitação séria.

7.2 Expansão local da hierarquia das máscaras

Em uma descrição hierárquica podem haver chamadas de células que chamam outras células, que por sua vez também podem chamar outras células, e assim por diante, até chegar a uma célula descrita em termos de primitivas geométricas (retângulos no caso atual). A cada chamada está associado um conjunto de transformações (translações, rotações e espelhamentos). A existência de uma seqüência de chamadas, uma dentro da outra, até chegar na primitiva geométrica, significa que esta primitiva deverá sofrer uma seqüência de transformações, correspondente a cada chamada de célula, indo da mais interna para a mais externa. Um programa destinado a expandir a hierarquia, deve ser capaz de realizar as transformações correspondentes a uma instância, sobre os retângulos da célula chamada, e também sobre as subcélulas que foram chamadas de dentro dela. Qualquer seqüência de transformações pode ser expressa por um conjunto de cinco transformações equivalentes, representadas por dois inteiros e tres indicadores. Estas transformações são: translação em X, translação em Y, espelhamento em X, espelhamento em Y e troca das coordenadas X e Y. Quando uma célula deve sofrer uma série de transformações sucessivas, é conveniente converter esta seqüência em uma única transformação equivalente. Esta transformação pode ser usada então para todos os

retângulos que compõem a célula. Para obter esta transformação equivalente são fetas as seguites operações:

a) A translação equivalente a uma seqüência de duas transformações é a soma das translações, se a segunda transformação não tiver espelhamento. Ela é a primeira translação menos a segunda, se a segunda tiver espelhamento.

b) Se houver troca de coorenadas X e Y, trocar o inicador de espelhamento em X com o indicador de espelhamento em Y.

c) Realizar uma operação OU exclusivo com os indicadores de espelhamento e troca de coordenadas X e Y das duas transformações, para obter os indicadores da transformação equivalente.

A maioria das análises feitas sobre as descrição hierárquica das máscaras requer uma expansão da hierarquia restrita a um determinado local do circuito. Por exemplo, para visualizar uma parte do circuito com um editor gráfico, se deve expandir a hierarquia do layout apenas na região que fica dentro da janela de visualização. Expandir a hierarquia no circuito inteiro seria uma perda de tempo.

Para agilizar a expansão local da hierarquia, se criou o conceito de envelope. O envelope de uma célula é o menor retângulo que engloba todos os retângulos e todos os envelopes das células que a compõem. Sendo um retângulo, o envelope sofre as mesmas transformações que os demais retângulos, quando a célula é instanciada. Uma chamada de célula não tem efeito fora do retângulo resultante do envelope submetido às transformações da chamada. Isto permite inferir que se o envelope de uma célula que foi instanciada estiver fora da região de interesse, todos os retângulos que seriam gerados na expansão desta chamada ficarão fora desta região. Esta chamda de célula, portanto, poderá ser decartada como um todo, sem que seja necessário verificar os seus retângulos individualmente. Apenas as células cujo envelope está total ou parcialmente dentro da região de interesse precisam ser expandidas.

A expansão da descrição hierárquica da geometria das máscaras é feita através de chamadas recursivas de uma função. O algoritmo para expandir uma célula é apresentado a seguir usando uma linguagem de programação informal. CLI é uma chamada de célula à qual corresponde um conjunto de transformações e uma célula que está sendo chamada CLI.CELULA.

```
subprograma EXPANDE(CELULA):
Para cada instância CLI que aparece em CELULA
  {
  Acrescenta CLI à seqüência de transformações;
  Obtém a transformação TRANSEQ, equivalente à seqüência de
  transformações;
  Transforma o envelope de CLI.CELULA segundo TRANSEQ;
  Se o envelope estiver dentro da região considerada
    {
    Expande a célula CLI.CELULA,
    usando recursivamente o subprograma EXPANDE(CLI.CELULA);
    }
  Para cada retângulo RET da CELULA
    {
    Transforma RET segundo TRANSEQ;
    Se RET está dentro da região considerada
       { Usa o RET transformado (desenha, por exemplo); }
    Passa para o próximo retângulo RET;
    }
  Retira CLI da seqüência de transformações;
  Passa para a próxima instância CLI;
  }
Fim do subprograma EXPANDE;
```

O algoritmo para a expansão local da hierarquia permite expandir um número ilimitado de células aninhadas através de chamadas recursivas dele mesmo. Ele também depende de outro subprograma que acha a transformação equivalente a uma série de transformações.

7.3 Estrutura de dados da descrição hierárquica

A estrutura de dados usada para descrever o layout se baseia em tres tipos de estruturas: os cabeçalhos das definições de células, os retângulos e as chamadas de células. O circuito como um todo é formado por um conjunto de definições de células. As células são definidas usando um cabeçalho de definição de célula, uma lista de retângulos para cada camada, e uma lista de chamadas de células.

O cabeçalho da definição de uma célula apresenta informações de caráter geral da célula. Estas informações são:

a) Nome da célula;

b) Número da célula;

c) Dois pares de coordenadas que definem os cantos opostos do envelope;

d) Apontador para o início da lista encadeada dos retângulos que compõem a célula;

e) Apontador para o início da lista de chamadas de células;

f) Apontador para a definição da próxima célula.

Os retângulos são os elementos fundamentais na descrição da geometria das máscaras. Além dos dois pares de coordenadas que determinam a localização e as dimensões dos retângulos, esta estrutura de dados mantém algumas informações específicas para o extrator. Estas informações são:

a) Camada a que o retângulo pertence (indica se o retângulo faz parte da máscara de difusão, de polisilício, ou de metal etc.)

b) Número de nó elétrico do retângulo. (Ver extração da conectividade no capítulo 2);

c) Número de níveis da hierarquia segundo os quais o retângulo deve ser promovido durante a extração

d) Perímetro e área da região onde está o retângulo;

e) Elo na lista de retângulos para o cálculo de resistências;

f) Elo na lista de retângulos promovidos em uma chamada de célula.

g) Apontador para o próximo retângulo da lista.

As chamadas de células informam que célula está sendo chamada, e que transformações esta célula deve sofrer. As transformações podem ser de translação, rotação ou espelhamento. Os campos da estrutura de dados das chamadas de células são os seguintes:

a) Apontador para o cabeçalho da definição da célula que está sendo chamada. Através deste apontador se tem acesso a todas as informações a respeito da célula chamada.

b) Valores das translações em X e em Y. As translações consistem em somar estes valores às coordenadas correspondentes de todos os componentes da célula chamada.

c) Indicadores de espelhamento em X e espelhamento em Y. O espelhamento consiste em trocar o sinal da coordenada correspondente.

d) Indicador de troca das coordenadas X com as coordenadas Y. Esta troca de coordenadas, junto com os espelhamentos, permite realizar rotações de qualquer ângulo múltiplo de 90°. Por exemplo, uma rotação de 90° pode ser obtida através das troca das coordenadas X e Y, seguida de um espelhamento em X.

e) Apontador para a lista de retângulos que foram promovidos de modo a formar a lista de parâmetros da chamada da célula. Esta lista de parâmetros é o conjunto de retângulos através dos quais a célula onde está a chamada tem ligação elétrica com a célula que está sendo chamada. O significado desta lista de parâmetros será explicado com mais detalhes mais adiante.

f) Apontador para a próxima chamada de célula. Este apontador é o elo da lista encadeada das instâncias de células usadas em uma definição de célula.

Estas tres estruturas permitem armar o grafo em árvore correspondente à descrição hierárquica da geometria das máscaras de um circuito integrado. Uma definição formal

da estrutura de dados usada pode ser vista pelo cabeçalho em linguagem C do programa. Além das estruturas usadas para representar a geometria das máscaras, que são o retângulo, a definição de célula e a instância de célula, aparece aqui a estrutura transistor, usada para armazenar o resultado da extração, e o texto, que o extrator usa para atribuir nomes aos nós.

```
/*-------------------------------------------------------------*/
/* EXTRATOR HIERÁRQUICO DE CIRCUITOS  * EXTRIBO VERSÃO 3.0  */
/*                                                             */
/*                  ARQUIVO DE CABEÇALHO                       */
/*-------------------------------------------------------------*/

/*-------------------------------------------------------------*/
/* A estrutura retang define a primitiva geométrica básica   */
/* do layout para o extrator, que é o retângulo. No retângulo*/
/* estão incluídas informações a respeito da região equipo-  */
/* tencial a que ele pertence.                                */
/*-------------------------------------------------------------*/
typedef struct retang
   {
   int  x1, y1, x2, y2; /* Coordenadas dos cantos            */
   float  area;         /* Área da região equipotencial     */
   int  per,            /* Perímetro da região               */
        tag;            /* Número de nó da equipotencial     */
   char niv,     /* nível a que o retângulo pertence         */
        rank;    /* rank é o número de  níveis  herárquicos  */
                 /* em que o retângulo deverá ser promovido  */
                 /* durante a extração do circuito           */
   struct retang
      *prox,     /* Elo por níveis                           */
      *pcli,     /* Elo na lista de parâmetros da chamada    */
      *pno;      /* Elo na lista de terminais de resistores  */
   } *PRET;

typedef struct texto   /* O texto tem o tamanho de 2 retângulos */
  {
  int x, y, tamanho, tag;
  char texto[18];
  struct texto *prox;
  } *TEXTO;

/*-------------------------------------------------------------*/
/* A estrutura celdef é o cabealho da célula no extrator     */
/* hierárquico. Ela tem os apontadores para os retângulos    */
/* que tem ligações externas.                                 */
/*-------------------------------------------------------------*/
```

```
typedef struct celdef
  {
  PRET  liret;        /* lista de retângulos          */
  void  *linst;       /* Lista de instâncias de células */
  char  nomedef[20]; /* Nome da célula               */
  int
     numero,     /* Número da célula             */
     mbb[4],     /* Mínimo retângulo envolvente  */
     nret,       /* Número de retângulos         */
     ntr[4],     /* Número de transistores       */
     suprimido; /* Indica que a célula não consta do netlist */
                 /* Porque não tem nenhum componente próprio. */
  struct celdef *ant, *proxdef; /* Elo na lista encadeada    */
  } *CELDEF;

/*-------------------------------------------------------------*/
/* A estrutura celinst descreve uma chamada de célula. Nesta */
/* estrutura está uma referência à célula que está sendo     */
/* chamada e as informações que especificam as transforma-   */
/* ções que esta célula deve sofrer.                         */
/*-------------------------------------------------------------*/
typedef struct celinst
  {
  int mr, /* Indicador de espelhamento e rotação: txy my mx */
          /* Os 3 bits menos significativos são indicadores:*/
          /* bit 0: mx = indicador de espelhamento em X     */
          /* bit 1: my = indicador de espelhamento em Y     */
          /* bit 2: txy= indicador de troca de X com Y      */

  tr[2];  /* Translação em X: tr[0] e em y: tr[1]           */

  numero; /* Número da célula que está sendo chamada        */

  CELDEF celula;  /* Apontador para a célula chamada */
  struct celinst *proxinst;
  PRET param;      /* Apontador para a lista encadeada de  */
                   /* retângulos que foram promovidos      */
  } *CELINST;

/*-------------------------------------------------------------*/
/* A estrutura transistor informa as características,        */
/* dimensões e terminais dos transistores.                   */
/*-------------------------------------------------------------*/
struct transistor
   {
   char tipo;            /* Tipo de transistor          */
   float wg, lg;         /* Largura e comprimento do canal */
   struct retang
        *pns,            /* Apontador para source       */
        *pnd,            /* Apontador para drain        */
        *png,            /* Apontador para gate         */
        *pnb,            /* Apontador para bulk         */
        *pnc;            /* Apontador para canal        */
   struct transistor *proxt;     /* Elo                  */
   };
```

7.4 Netlist hierárquico: isomorfismo da hierarquia

Do ponto de vista da teoria dos grafos, a descrição hierárquica da geometria das máscaras apresenta uma estrutura em árvore, onde são permitidas reconvergências, mas ciclos ou laços não são permitidos. Esta restrição permite que as várias definições de células que compõem o circuito sejam ordenadas de modo que uma determinada definição de célula dependa apenas de células que já foram definidas anteriormente. Neste caso, diremos que as células estão definidas em ordem ascendente na hierarquia. No nivel mais baixo, temos as células que não apresentam chamadas de células em sua definição, devendo serem as primeiras a aparecerem na descrição do circuito. A medida que sobe-se na hierarquia, começam a aparecer células que chamam instâncias das células hierarquicamente mais baixas.

O extrator gera uma lista de componentes com a mesma topologia hierárquica que a descrição geométrica das máscaras. Portanto a hierarquia da lista de componentes gerada pelo extrator e a da geometria das máscaras devem ser isomórficas. Na lista de componentes a definição de célula corresponde a uma definição de subcircuito. Em formato SPICE esta definição de subcircuito tem a seguinte sintaxe:

.SUBCKT *<nome do subcircuito> <no1> [<no2> <no3>...]*
<lista dos componentes internos do subcircuito>
.ENDS [*<nome do subcircuito>*]

A linha .SUBCKT marca o início da definição do subcircuito. Depois do nome do subcircuito vem uma lista de números dos nós que tem ligações externas (parâmetros do subcircuito). Os numeros de nós usados na descrição dos componentes internos tem validade local, exceto o nó 0 (terra). A linha .ENDS marca o final da definição do subcircuito.

```
*
* UFRGS - PGCC - GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CIRCITO: dois
* TECNOLOGIA: gren2.tec  TIPO CMOS
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  INVERSOR
*
* Transistores tipo NMOS: 1
* Transistores tipo PMOS: 1
*
.SUBCKT INVERSOR   0   1   2   3   4
*
MP1   2   4   1   1 PMOS L=2.1U W=8.5U AD=72P AS=72P PD=34U PS=34U
MN2   3   4   0   0 NMOS L=3.2U W=6.3U AD=48P AS=54P PD=28U PS=30U
.ENDS INVERSOR
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  BUFFER
*
* Transistores tipo NMOS: 2
* Transistores tipo PMOS: 2
*
.SUBCKT BUFFER   0   1   2   3   4   5
*
X1      0   1   2   4   3 INVERSOR
X2      0   1   3   3   5 INVERSOR
.ENDS BUFFER
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  BARRAMENTO
*
* Transistores tipo NMOS: 8
* Transistores tipo PMOS: 8
*
X1      0   1   6   7   6  12 BUFFER
X2      0   1   3   2   3  10 BUFFER
X3      0   1   4   5   4  11 BUFFER
X4      0   1   8  13   8   9 BUFFER
.END
```

Figura 7.2 - Netlist resultante da extração hierárquica do circuito descrito em RS na figura 7.1.

As chamadas de subcircuitos, correspondendo às chamadas de células na descrição hierárquica da geometria das máscaras, tem a seguinte forma geral:

X<*índice da chamada*> <*no1*> [<*no2*> <*no3*>...] <*nome do subcircuito*>

A chamada é feita usando um pseudo-componente cujo nome deve começar com a letra X. A lista de números de nós que vem a seguir corresponde, na mesma ordem, aos nós da lista de parâmetros na definição do subcircuito.

A correspondência entre a descrição do layout do circuito e o netlist hierárquico gerado pelo extrator, pode ser observada no exemplo a seguir. Inicialmente foi definido um inversor CMOS. Na célula "BUFFER", foram feitas duas chamadas consecutivas do inversor de modo a inverter o sinal duas vezes. Na célula "BARRAMENTO", foram feitas duas chamadas duplas do "BUFFER", formando um barramento de 4 bits. A descrição RS da geometria deste circuito é mostrada na figura 7.1.

```
DS 1;
9 "INVERSOR";
(Aqui vai a lista de retângulos que descreve o inversor);
DF;

DS 2;
9 "BUFFER"; (Buffer formado por 2 chamadas do inersor);
C 1 T 20 2;
C 1 T 0 2;
DF;

DS 3;
9 "BARRA"; (4 Chamadas do buffer, formando o barramento);
C 2 T 0 -100 MY PY 2 96;
C 2 PY 2 96;
DF;
```

Figura 7.1 - Descrição RS do inversor duplo. A lista de retângulos na definição da célula INVERSOR foi omitida.

```
*
* UFRGS - PGCC - GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CIRCITO: dois
* TECNOLOGIA: gren2.tec  TIPO CMOS
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  INVERSOR
*
* Transistores tipo NMOS: 1
* Transistores tipo PMOS: 1
*
.SUBCKT INVERSOR   0   1   2   3   4
*
MP1   2   4   1   1 PMOS L=2.1U W=8.5U AD=72P AS=72P PD=34U PS=34U
MN2   3   4   0   0 NMOS L=3.2U W=6.3U AD=48P AS=54P PD=28U PS=30U
.ENDS INVERSOR
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  BUFFER
*
* Transistores tipo NMOS: 2
* Transistores tipo PMOS: 2
*
.SUBCKT BUFFER   0   1   2   3   4   5
*
X1      0   1   2   4   3 INVERSOR
X2      0   1   3   3   5 INVERSOR
.ENDS BUFFER
*
* SUBCIRCUITO CORRESPONDENTE A CELULA  BARRAMENTO
*
* Transistores tipo NMOS: 8
* Transistores tipo PMOS: 8
*
X1      0   1   6   7   6  12 BUFFER
X2      0   1   3   2   3  10 BUFFER
X3      0   1   4   5   4  11 BUFFER
X4      0   1   8  13   8   9 BUFFER
.END
```

Figura 7.2 - Netlist resultante da extração hierárquica do circuito descrito em RS na figura 7.1.

7.5. Extração hierárquica

A técnica de extração hierárquica de circuitos adotada aqui consiste em analisar em detalhes uma célula de cada vez, não se preocupando com as demais. Isto traz uma série de vantagens, dentre as quais a principal é a economia de memória. O relacionamento de uma determinada célula com as outras que compõem um circuito se dá de duas maneiras. A célula pode se relacionar com outras abaixo dela na hierarquia através das chamadas de células, que se traduzem no net-list, como chamadas de subcircuitos. A célula também se relaciona com células hierárquicamente superiores através do cabeçalho do subcircuito, com a sua lista de parâmetros. Portanto para que se possa fazer uma análise ascendente deste tipo é necessário que se tenha um conhecimento prévio de que partes de cada célula podem ter contato com outras células do circuito. Estas partes que podem ter contato com outras células é que darão origem aos parâmetros na declaração .SUBCKT do cabeçalho do subcircuito.

A solução encontrada para descobrir onde se localizam as ligações externas de cada célula foi escrever um programa independente do programa principal, especialmente para esta finalidade. Este programa serve como um preprocessador, que gera um arquivo intermediário que informa, juntamente com a descrição RS das máscaras, que retângulos de cada célula têm ligações externas. O arquivo intermediário também diz em quantos níveis hierárquicos cada retângulo deve ser promovido para que a sua ligação externa seja encontrada.

O programa principal do extrator analisa as células em ordem ascendente, uma de cada vez, mantendo na memória apenas a célula que está sendo analisada no momento e a lista dos retângulos que tem ligações externas de cada uma das células que já foram extraídas. Quando o programa encontra uma chamada de célula, os retângulos com ligações

ativa no momento. A promoção de um retângulo consiste em buscar um retângulo na célula que está sendo chamada, transformá-lo segundo a transformação da chamada, e incluir este retângulo transformado na lista de retângulos da célula que está sendo extraída.

A extração da célula então se dá da mesma forma que a extração plana de um circuito, porém incluindo os retângulos promovidos como se eles fizessem parte da célula. As chamadas de células incluem uma lista dos retângulos promovidos que elas geraram. Isto permite que o extrator obtenha, em uma etapa posterior, uma lista de parâmetros de cada chamada de célula.

A lista de parâmetros da declaração .SUBCKT é obtida usando as informações a respeito das interfaces externas de cada célula fornecidas pelo pré-processador. A cada retângulo é associado um número (RANK) que indica quantos níveis hierárquicos um retângulo deve ser promovido. Todos os retângulos que tiverem este número RANK diferente de zero terão seus números de nós incluídos na lista de parâmetros do cabeçalho .SUBCKT.

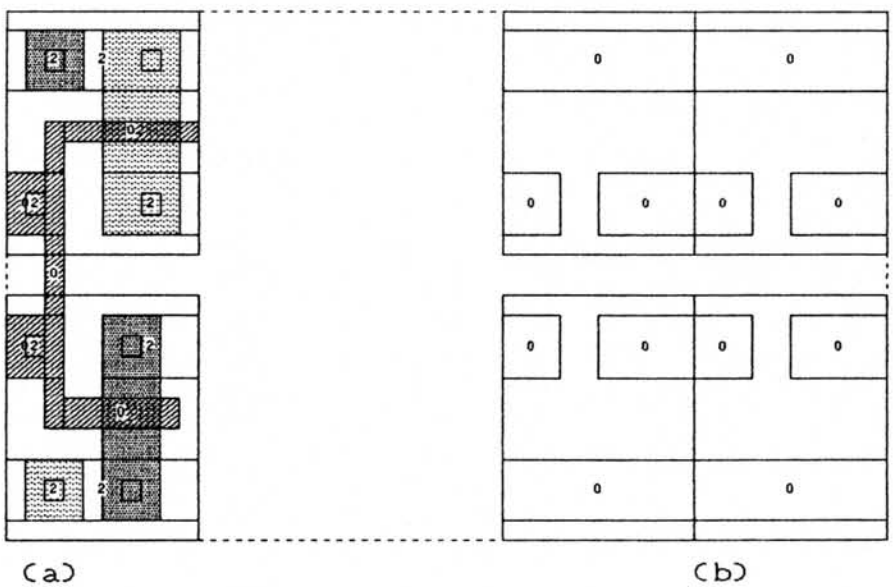
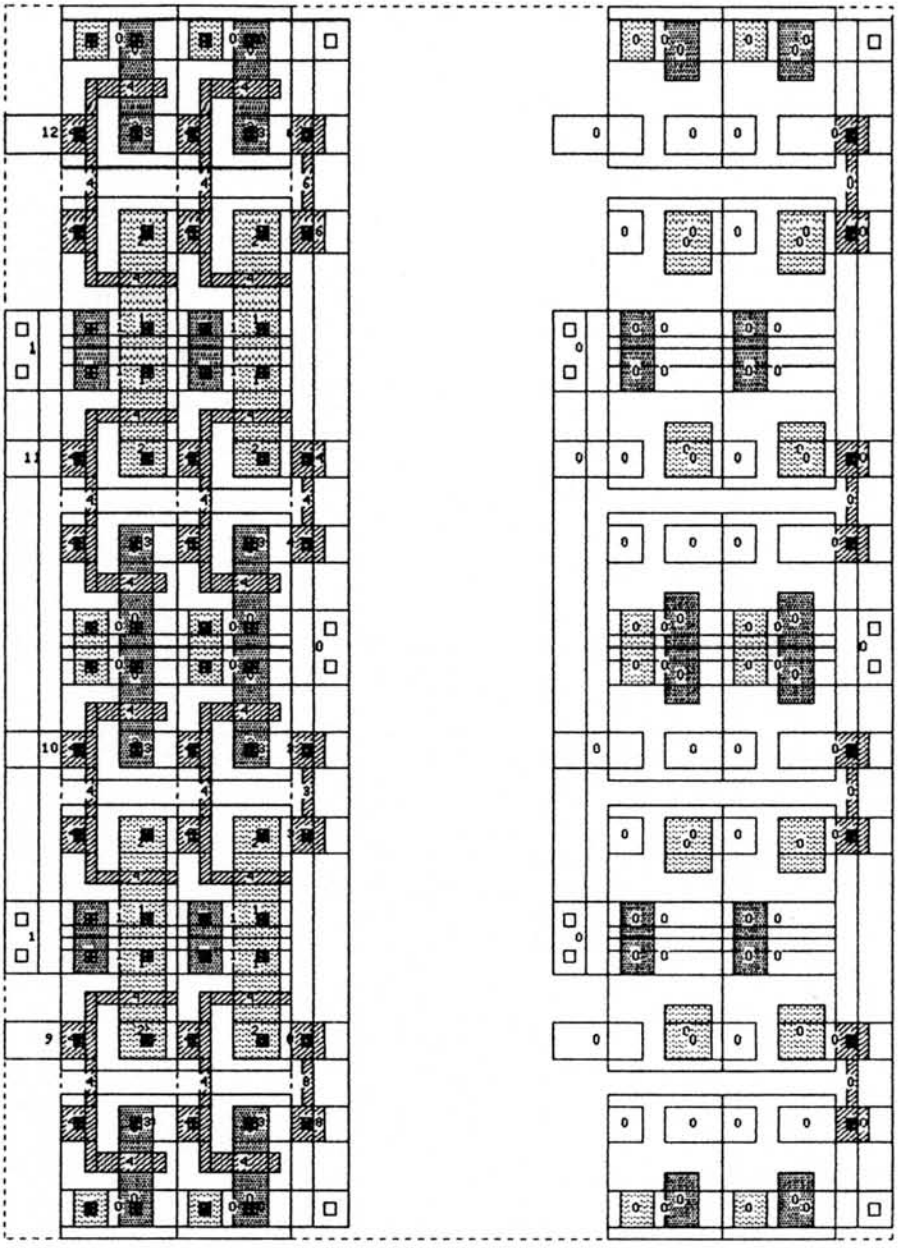


Figura 7.3 - (a) O inversor com o número de promoções (RANK)



(a) (b)

Fiura 7.4 - (a) Um barramento formado por 4 chamadas do buffer. (b) O barramento com apenas os retângulos promovidos

7.6 Preprocessador para identificar as interfaces externas das células

Enquanto o programa principal do extrator analisa uma célula de cada vez, seguindo de baixo para cima na hierarquia, o módulo para encontrar as interfaces externas das células exige uma verificação ascendente e descendente de várias células simultaneamente. Para isto, o programa mantém a descrição completa do circuito na memória, usando as tres estruturas de dados básicas, as definições de células, os retângulos e as chamadas de células.

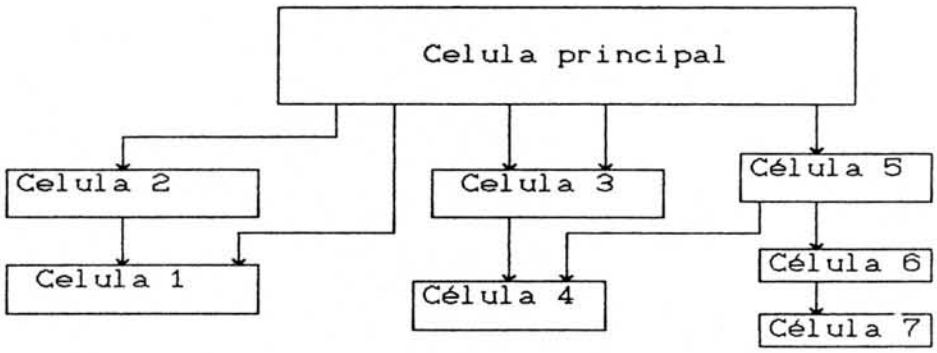


Figura 7.5- *Esquema de uma estrutura hierárquica de células. Se a célula 7 tiver uma ligação com a célula 2 através da célula principal, as interfaces da célula 7 terão RANK=3, e as da célula 2 terão RANK=2.*

O algoritmo de localização das interfaces de uma célula (que chamaremos de CELAT) inicia pela busca de chamadas da celula CELAT em todo o circuito. Quando uma chamada é encontrada, o programa procura todos os retângulos que tem contato com o envelope da chamada da CELAT. Para achar estes retângulos, além de verificar os retângulos próprios da célula que chamou CELAT, todas as células vizinhas ou sobrepostas à chamada de CELDEF são expandidas. Os retângulos encontrados são então transformados para as coordenadas próprias da CELAT. Então o programa verifica se estes retângulos tem contato com algum dos retângulos próprios da CELAT, segundo as regras de conectividade da

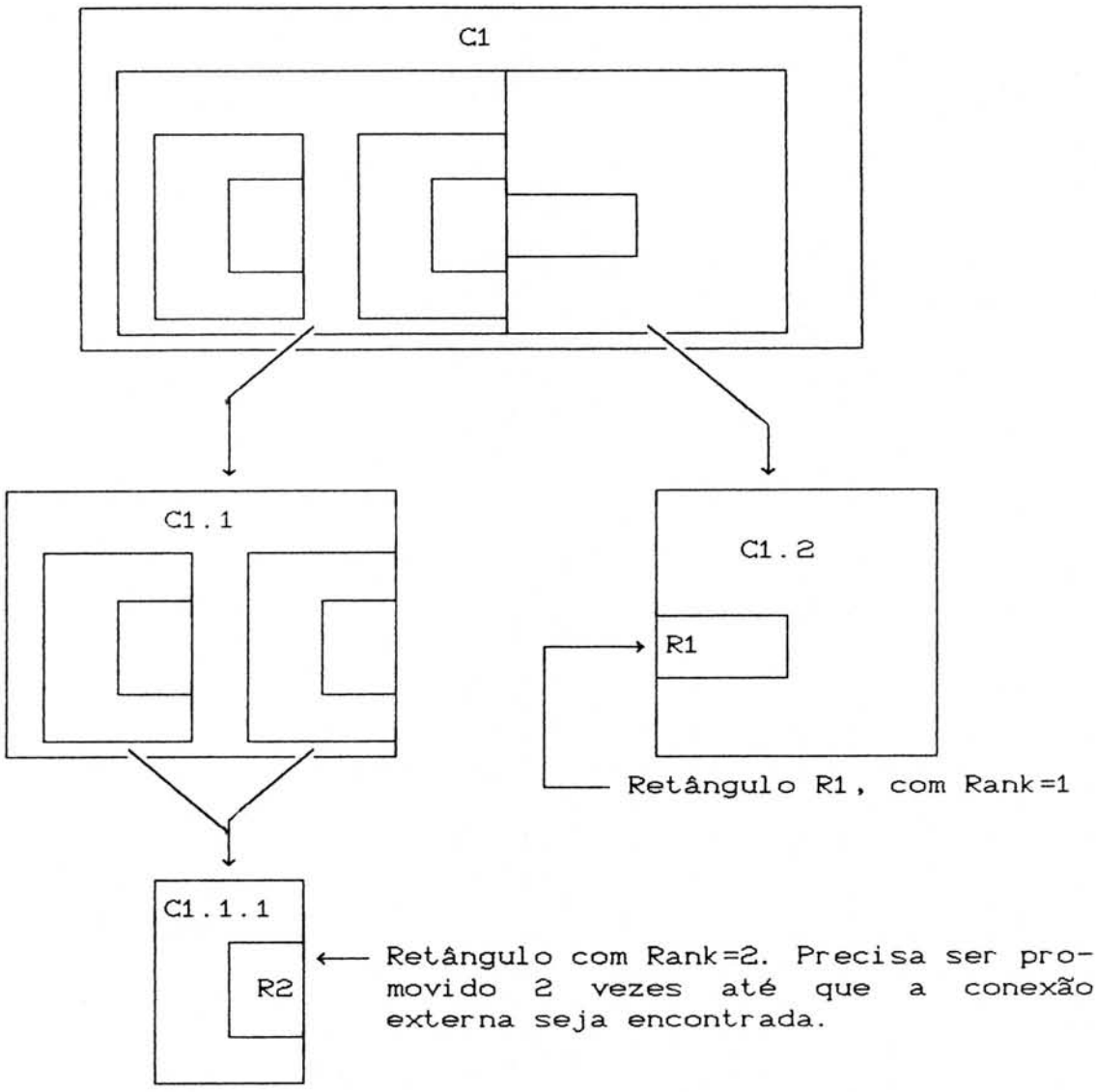
contatos externos nesta etapa são marcados com RANK=1. Isto significa que eles devem serem promovidos uma vez na análise ascendente da estrutura hierárquica do circuito pelo segundo módulo do extrator.

As ligações externas, porém, não aparecem somente nas células que chamam a célula CELAT. Também podem existir ligações que só vão se revelar em alguma das células que chamam uma das células que chamam a célula considerada CELAT. Os retângulos da CELAT que tem ligações deste tipo são marcados com RANK=2. Eles deverão ser promovidos duas vezes na análise de baixo para cima da hierarquia. A localização de interfaces externas da célula, que se revelam depois de um número qualquer de chamadas de profundidade, é feita através da chamada recursiva de um subprograma. O subprograma INTERFACE descrito a seguir serve para encontrar os retângulos da célula CELPROC que tem contato externo.

```
Subprograma INTERFACE(CELPROC):
1. RANK := RANK + 1;
2. Para cada célula CLD do circuito:
2.1. Para cada chamada de célula CLI que aparece em CLD:
2.1.1. Se CLI é uma chamada da célula CELPROC:
2.1.1.1. Acrescenta CLI à seqüência de transformações;
2.1.1.2. Encontra a transformação equivalente TRANSEQ;
2.1.1.3. Transforma o envelope de CELPROC segundo TRANSEQ;
2.1.1.4. EXPANDE(CLD) dentro do envelope de CELPROC;
2.1.1.5. Para todos os retângulos RET da célula CLD expandida:
2.1.1.5.1. Se RET foi gerado pela chamada CLI, passa para o
          próximo retângulo RET;
2.1.1.5.2. Se estiver dentro do envelope transformado de CELPROC:
2.1.1.5.2.1. Se RET tem contato com algum retângulo de CELPROC:
2.1.1.5.2.1.1. Marca o retângulo de CELPROC com RANK, se ele
              ainda não estiver marcado com um valor maior
              de RANK;
2.1.1.5.3. Passa para o próximo retângulo RET;
2.1.1.6. Chama recursivamente INTERFACE(CLD);
2.1.1.7. Retira CLI da seqüência de transformações;
2.1.2 Passa para a próxima chamada de célula CLI;
2.2. Passa para a próxima célula CLD;
3. RANK := RANK - 1;
4. Fim do subprograma INTERFACE.
```

O subprograma INTERFACE é chamado para cada célula do circuito com RANK inicializado em 0 (zero). Ao final da operação, o campo RANK dos retângulos estará marcado com a profundidade de chamadas de células que foi necessária verificar para encontrar a interface externa.

Para agilizar o funcionamento do programa, os retângulos das células já analisadas, que não tem ligação externa são eliminados cada vez que o programa termina de verificar as ligações externas de uma célula.



A figura 7.6 ilustra o significado do número RANK em uma descrição hierárquica do layout. A célula C1.1.1 se liga com a célula C1.2 indiretamente através da célula C1. Para que esta conexão seja encontrada, R1 precisa ser promovido uma vez, e R2 precisa ser promovido duas vezes.

7.7 Simplificação da estrutura hierárquica

Quando um número grande de níveis hierárquicos é usado para descrever um circuito, o netlist gerado pelo extrator tende a se tornar confuso. Todos os sinais de interface das células básicas tem que ser passados como parâmetros de chamadas de subcircuitos através de todos os níveis intermediários da hierarquia até chegar na célula onde as ligações se realizam. Isto faz com que apareçam no net-list definições de subcircuitos com um número muito grande de parâmetros que muitas vezes não contribuem com nada de novo no circuito. O resultado é um netlist confuso, e desnecessariamente grande.

A solução encontrada para resolver este problema foi simplificar a estrutura hierárquica da descrição do layout, eliminando os níveis desnecessários. Esta operação de simplificação da hierarquia é feita no módulo pré-processador, antes da localização das interfaces externas das células. As definições de células que formam níveis intermediários na hierarquia, isto é, que chamam outras subcélulas e que são chamadas apenas uma vez em todo o circuito, são eliminadas. A eliminação é feita promovendo os retângulos e as chamadas de subcélulas da célula que está sendo eluminada para a célula onde ela é chamada, com as transformações necessárias. Feito isto, a definição da célula pode ser eliminada da estrutura de dados em memória.

Com esta simplificação, todas as definições de subcircuitos apresentam componentes novos ou definem ligações entre várias células. O número de interfaces externas das células se torna muito menor quando a hierarquia é simplificada, permitindo que um netlist claro e compacto seja gerado pelo extrator.

7.8 Parâmetros das chamadas de subcircuitos

O cabeçalho do subcircuito, correspondente ao cartão .SUBCKT do relatório SPICE deve apresentar como lista de parâmetros os números de nós dos retângulos que tem interfaces externas. Não devem haver parâmetros repetidos. Para evitar a repetição de parâmetros, os retângulos com interfaces externas são ordenados segundo o número de nó, e os retângulos com números consecutivos são omitidos da lista de parâmetros do cartão .SUBCKT.

No início da operação de extração de uma célula, as chamadas de subcélulas são analisadas. Os retângulos com ligações externas, de cada célula que é chamada, são promovidos para a célula que está sendo extraída. Os retângulos resultantes desta promoção são colocados em uma lista encadeada especial, associada à respectiva chamada de célula. Quando o programa escreve o net-list, os parâmetros da chamada de subcircuito são obtidos através desta lista. É importante garantir a correspondência entre os parâmetros da definição da célula e os parâmetros usados nas chamadas desta célula. Para garantir esta correspondência, a lista dos retângulos com interface externa da célula que está sendo chamada e a lista de retângulos promovidos correspondentes à instância são percorridas simultaneamente. Uma chamada de célula pode ter na lista de retângulos que devem ser promovidos vários retângulos com o mesmo número de nó. Isto dá origem a uma série de conexões no circuito que são feitas através da célula chamada. Estas conexões são levadas em consideração depois que a conectividade interna do circuito foi

Quando a extração de uma célula é concluída, os componentes obtidos a partir desta célula são anexados ao arquivo de saída. Feito isto, a maior parte das informações referentes a esta célula não será mais necessária, e será apagada da memória. Devem ser mantidos na memória o cabeçalho da célula e os retângulos que tem interfaces externas. Mais adiante, quando esta célula for chamada, estas informações serão necessárias. Os retângulos com interfaces externas serão promovidos, e servirão como parâmetros para as chamadas. Células que são usadas apenas como conexão ou roteamento, não dando origem a nenhum componente como transistor, resistor ou capacitor, não aparecerão no *netlist*. Estas células, porém são mantidas normalmente na estrutura de dados em memória. Elas poderão influenciar nas interligações das células que as chamarem.

O procedimento completo do extrator hierárquico fica mais claro na lista de etapas a seguir:

PROGRAMA PRINCIPAL

1. Ler o arquivo de descrição da tenologia
2. Abrir o arquivo RS ou CIF da descrição geométrica da célula e o arquivo de saída.
3. Ler o arquivo RS ou CIF até encontrar um fim de célula DF
4. Chamar o subprograma EXTRATOR DE CÉLULA
5. Repetir as operações a partir do passo 3 até chegar ao fim do arquivo RS ou CIF
6. Fechar os arquivos.
7. Fim

Subprograma EXTRATOR DE CÉLULA

1. Ordenar os retângulos da célula atual
2. Para cada chamada de célula CLI feita na célula atual
   2.1. Promover os retângulos de CLI, realizando as transfor-

2.2. Os retângulos promovidos terão o campo RANK reduzido de um.

3. Realizar as operações sobre as máscaras especificadas no arquivo de descrição da tecnologia.

4. Realizar as operações de máscaras para a primeira etapa da extração dos transistores. Promover a separação entre o dreno e a fonte.

5. Se for requisitada a extração detalhada com resistências e capacitâncias:

   5.1. Preparar as listas de retângulos que formam as regiões com resistência. Isto é feito através de uma avaliação de conectividade interna do nível em questão, gerando uma lista de retângulos para cada número de nó.

   5.2. Preparar a lista de terminais de cada uma destas regiões. Estes terminais podem ser retângulos de outros níveis que têm contato ou terminais dos transistores.

   5.3. Calcular as resistências para cada região, usando o algoritmo descrito no capítulo 6.

   5.4. Indicar que os retângulos usados no cálculo de resistências não serão mais considerados na avaliação da conectividade.

6. Avaliar a conectividade interna e inter níveis de acordo com as regras de interconexões especificadas no arquivo de descrição da tecnologia.

7. Verificar as possíveis conexões feitas através de chamadas de células.

8. Segunda etapa da extração dos transistores: Reconhecer os retângulos que formam os terminais e calcular as dimensões do canal e as áreas e perímetros de dreno e fonte.

9. Se tiver sido solicitado o cálculo de resistências, realizar as possíveis ligações das resistências em paralelo.

10. Localizar a alimentação pelos métodos descritos no capítulo 5.

11. Determinar os parâmetros das chamadas de subcircuitos. Usar a lista de retângulos promovidos gerada por cada chamada de célula.

12. Anexar ao arquivo de saída os resultados correspondentes à célula extraída.

13. Criar um cabeçalho para as informações resumidas da célula.

14. Passar os retângulos que são interfaces externas (RANK≠0) para o conjunto de informações resumidas da célula. Estes retângulos serão os parâmetros para as próximas chamadas.

15. Desalocar a memória usada para a extração desta célula.

16. Fim do subprograma EXTRATOR DE CÉLULA.

Estas etapas apresentam uma visão global do funcionamento do segundo módulo do extrator, que é o módulo que realiza a extração propriamente dita. A cada rodada do subprograma extrator de células, um trecho do *netlist* correspondente a um subcircuito é escrito e a memória usada é desalocada. Desta forma o circuito extraído pode ser maior do que o maior circuito que cabe inteiro na memória.

7.9 O pós-processador

O extrator gera um *netlist* correto do ponto de vista das ligações, a não ser pelo fato de que pode ter ocorrido um erro na identificação da alimentação, em algumas células, conforme foi visto no capítulo 5. Para corrigir estes erros de identificação da alimentação e simplificar o *netlist*, eliminando a passagem de parâmetros desnecessários ou repetidos para os subcircuitos é necessário que o *netlist* seja analizado globalmente. Esta enálise é feita como uma terceira etapa da extração, com o uso de um pós-processador. Nesta etapa os dados geométricos do layout do circuito não são mais necessários.

O pós-processador realiza as seguintes tarefas:

a) Verifica, e corrige se necessário, a identificação da alimentação das células.

b) Elimina os parâmetros desnecessários ou repetidos das chamadas de subcircuitos.

c) Verifica se existem no circuito configurações anômalas de transistores, como portas CMOS não complementares, entradas em aberto, transistores de carga NMOS não ligados a VDD etc. Encontrando algum destes casos o programa emite um aviso, denunciando um provável erro no layout do circuito.

d) Atualiza o arquivo de visualização do layout com os novos números de nós obtidos pelo pós-processador.

8 RELATÓRIOS DE SAÍDA

Atualmente o extrator EXTRIBO pode fornecer quatro tipos de relatório de saída. Dois deles são listas de componentes para simuladores: o SPICE e o ARAMOS. Um é uma descrição do netlist em formato binário, que serve para passar informações para o pós processador. O outro relatório é uma escrição geométrica do *layout* do circuito à qual são acrescentadas informações a respeito da localização dos componentes encontrados pelo extrator. Todos estes relatórios são fornecidos em formato ASCII, de modo que eles podem ser lidos ou editados com um editor de texto. Os relatórios SPICE e ARAMOS usam apenas um subconjunto das diretivas admissíveis.

8.1 Relatório SPICE

Os relatórios SPICE e ARAMOS não estão prontos para serem usados em simuladores, da forma que são gerados pelo extrator. É necessário acrescentar as requisições de apresentação dos resultados da simulação e as descrições dos sinais de entrada usados como excitação.

No caso em que não é solicitado o cálculo das resistências e capacitâncias, o relatório SPICE será simplesmente uma lista de transistores com as especificações das áreas e perímetros das regiões de dreno e de fonte. O próprio simulador fará uma estimativa das resistências e capacitâncias com base nas informações de perímetro e área.

Quando é incluído o cálculo de resistências e capacitâncias, as áreas e perímetros das regiões de dreno e fonte poderão ser incluídas ou não, dependendo de especificações encontradas no arquivo de descrição da tecnologia. As áreas e perímetros são omitidos quando a capacitância e a resistência da região de difusão que forma o dreno e a fonte

capacitância da região de difusão, com base na área de dreno e de fonte. Desta forma, o simulador pode levar em consideração a variação da capacitância da junção da difusão com o substrato, em função da tensão.

8.2 Relatório ARAMOS

O simulador ARAMOS usa um formato de descrição do cir- cuito similar ao SPICE, apenas com regras de sintaxe diferentes. A opção de fornecer relatórios também neste formato foi feita por que este simulador tem sido muito usado no CPGCC da UFRGS.

O simulador ARAMOS não estima automaticamente as resistências e capacitâncias das interconexões. Por este motivo, para obter resultados precisos é indispensável que se faça a extração do circuito com resistências e capacitâncias.

8.3 Relatório gráfico

A lista de componentes obtida pelo extrator tem pouco ou nenhum valor se não poder ser interpretada pelo usuário. O arquivo gerado pelo extrator não pode ser usado diretamente como entrada para um simulador porque faltam algumas informações, que não podem ser obtidas a partir do layout do circuito. As informações que faltam normalmente são as requisições dos relatórios de saída do simulador e a especificação dos sinais de entrada e componentes externos ao circuito. O usuário deve acrescentar estas informações à lista de componentes gerada pelo extrator, com o uso de um editor de textos. Isto exige que o usuário seja capaz de identificar os componentes e os números dos nós obtidos pelo extrator. A maneira mais adequada de se fazer isto é associando os componentes obtidos pelo extrator com os padrões geométricos que os geraram, através de uma identificação visual. Esta necessidade de identificação visual deu origem às interfaces gráficas EMA e EMAPR, que são programas acessórios do extrator.

Levando em consideração a grande importância da visualização dos resultados para que o extrator seja uma ferramenta eficaz na verificação de circuitos integrados, foi dedicado um grande esforço na elaboração das interfaces gráficas. Foram escritas mais de 4000 linhas de programa em linguagem C (Quase metade de todo o pacote do extrator) dedicadas à visualização dos resultados.

Para aproveitar ao máximo os 640Kb do MSDOS, as interfaces gráficas foram implementadas como programas independentes, que transmitem informações entre si através de arquivos de texto. Estas ferramentas de projeto e verificação de circuitos integrados deverão ser integradas entre si e com outras ferramentas que estão sendo desenvolvidas no PGCC da UFRGS através de uma interface gerenciadora de menus (o sistema GIM [BAG89]). O objetivo é criar um sistema integrado para o projeto de circuitos integrados, de onde se possa acessar ferramentas de síntese automática [LUB89], [MOR89a], o editor de máscaras EMA, um DRC [GOM88], um banco de dados [MOR89b] e o extrator EXTRIBO.

O extrator EXTRIBO, além de fornecer as listas de componentes do circuito em formatos compatíveis com os simuladores SPICE e ARAMOS, também pode gerar uma descrição do layout com a localização dos componentes encontrados. A linguagem usada para esta descrição do layout com a localização dos componentes e dos números de nós é um superconjunto da linguagem RS. A instrução que define um retângulo, nesta linguagem, além das dimensões e das coordenadas do canto inferior esquerdo, passa a ter um quinto campo inteiro opcional. Este campo define o número de nó da região equipotencial a que o retângulo pertence. Ele é opcional para manter a compatibilidade com a linguagem RS. Desta forma a sintaxe da definição de um retângulo é a seguinte:

Os transistores podem ser localizados no layout através dos retângulos do nível de GATE. Os números de nós destes retângulos servem como identificação para os transistores.

A localização dos resistores é feita através de um comando especial. A declaração R define o índice que identifica o resistor, as coordenadas dos terminais e o valor da resistência. A sua sintaxe é a seguinte:

R<*índice*> <x_1> <y_1> <x_2> <y_2> <*nó1*> <*nó2*> <*resistência*>;

Esta diretiva permite que o editor EMA leia os dados necessários para desenhar os resistores obtidos pelo extrator sobre o layout do circuito.

Um recurso muito importante para a documentação do funcionamento de um circuito integrado é a possibilidade de incluir textos no desenho. Os textos são definidos através da diretiva 4N.:

4N < *x* > < *y* > "< *string* >";

Este comando associa um texto a um par de coordenadas. O editor EMA permite criar, editar e visualizar textos deste tipo. Para o extrator, textos incluídos desta forma no layout do circuito tem um significado maior do que uma simples documentação. O texto lido pelo extrator passa a ser considerado como o nome da região equipotencial para onde apontam as coordenadas. Se no local das coordenadas houverem níveis sobrepostos com números de nós diferentes, será considerado o nível mais alto no processamento do circuito integrado. Desta forma, o extrator pode usar nos Netlists gerados nomes de nós em vez de apenas números de nós. Com o uso abundante de nomes adequados para os sinais que são usados em um circuito integrado, o net-list gerado pelo extrator pode tornar-se claro a ponto de ficar auto

A visualização de um circuito com os componentes encontrados pelo extrator e nomes e números de nós, além de facilitar a interpretação dos resultados das simulações, permite detectar vários tipos de erro de layout, como conexões erradas ou faltando.

9 EMA - O EDITOR DE MÁSCARAS E INTERFACE GRÁFICA DO EXTRATOR

O programa EMA é um editor gráfico de máscaras para microeletrônica que permite visualizar e modificar a descrição hierárquica do *layout* circuitos integrados. Um conjunto de comandos de edição com visualização direta dos resultados na tela (tipo WYSIWYG), junto com uma estrutura de menus, tornam o programa fácil e agradável de usar. Além da possibilidade de criar, modificar e apagar retângulos e das facilidades de copiar e mover blocos no âmbito da edição de células, o editor permite atuar na estrutura hierárquica da descrição do circuito. Com o uso do EMA se pode criar novas definições de células, percorrer livremente as diversas células que compõem a descrição do circuito, e criar instâncias de células, editando interativamente as transformações que a célula deve sofrer ao ser chamada.

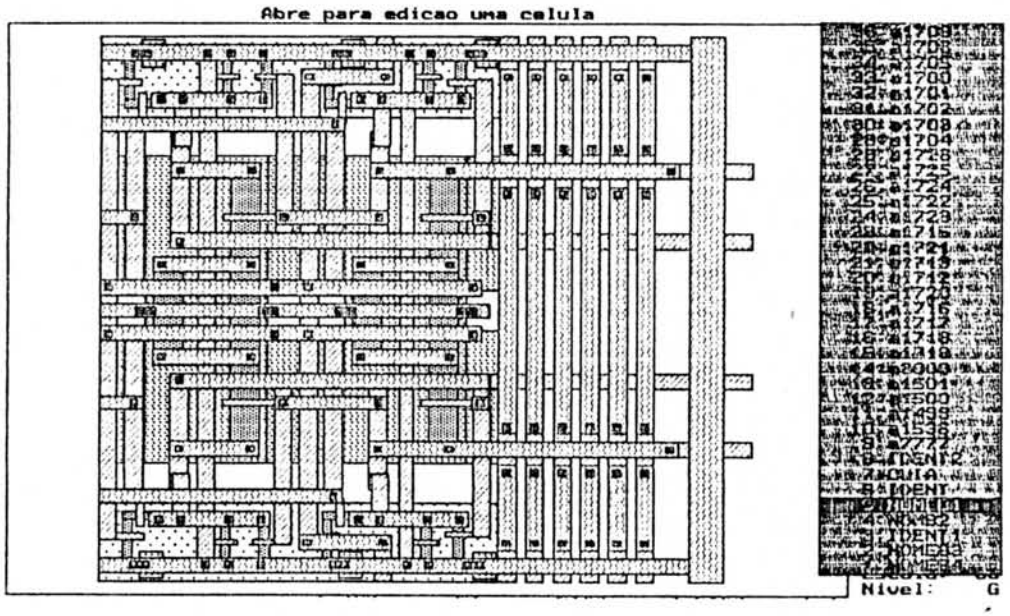


Figura 9.1 - Tela típica do editor EMA

O editor EMA também pode ser usado como interface gráfica para o extrator. A comunicação é feita passando um arquivo gerado pelo extrator para o EMA. Este arquivo contém informações sobre números de nós, localização dos transistores e as descrições dos componentes extraídos. Estes componentes podem ser localizados ou visualizados no layout usando comandos especiais para a visualização do programa.

O editor EMA pode ser usado com vários tipos de placas gráficas para computadores tipo PC, usando em todos eles os modos de maior resolução.

No apêndice A2 é apresentado um manual de instruções com uma descrição de alguns dos comandos do editor.

9.1 Algoritmos

Para ter-se o máximo de flexibilidade na edição, inclusão e remoção de elementos, o editor usa listas encadeadas para localizar os 3 tipos de objetos com que ele trabalha: definições de símbolos, instâncias de símbolos e retângulos. Os retângulos são ordenados segundo uma das coordenadas dos seus cantos. A ordenação dos retângulos torna muito mais rápida a localização de contatos ou sobreposições de retângulos.

A expansão da hierarquia para desenhar o circuito é feita através da chamada recursiva de uma função para desenhar células. Esta função mantém a informação da seqüência de chamadas de células ao longo da hierarquia e determina qual será a transformação equivalente à seqüência de transformações que cada retângulo deverá sofrer. Desta forma os retângulos só precisarão ser transformados uma vez, mesmo que a célula a que eles pertencem tenha sido transformada várias vezes. Isto faz com que o circuito possa ser redesenhado muito mais rapidamente.

As informações do extrator em relação aos componentes são passadas para o editor/visulaizador através de um arquivo de texto que contém o resultado da extração em formato SPICE, junto com uma descrição tipo RS do layout do circuito que associa um número de nó a cada retângulo. Os transistores e resistores podem ser localizados usando níveis especiais acrescentados pelo extrator.

9.2 EMAPRINT - programa para desenhar um circuito na impressora

O programa EMAPRINT serve para desenhar o *layout* de um circuito integrado na impressora a partir da sua descrição hierárquica em linguagem RS extendida. O programa EMAPRINT também reconhece todas as informações adicionais fornecidas pelo extrator no relatório gráfico. O programa EMAPRINT inclui no desenho os números dos nós, identificações dos transistores e textos, se estas informações estiverem disponíveis na descrição do circuito. O programa usa as técnicas de expansão local da hierarquia descritas no capítulo 7.

O EMAPRINT permite imprimir o circuito de várias maneiras, selecionáveis pelo usuário. A escala em que o desenho é feito é calculada automaticamente, de acordo com o tamanho do circuito e o espaço disponível no papel. As opções de impressão e dados sobre o espaço no papel são especificados na linha de comando. Uma lista completa das opções do EMAPRINT é apresentada no apêndice A2.

10 CONCLUSÃO

Foi apresentado um programa extrator de circuitos hierárquico para computadores tipo PC. Alguns algoritmos novos ou soluções pouco usuais foram necessárias para tirar o máximo proveito dos recursos limitados do computador. Para acelerar as pesquisas sobre as listas de retângulos, foi adotado o método da ordenação dos retângulos segundo uma das coordenadas. Este método, mesmo não sendo o mais eficiente em tempo de computação para circuitos grandes, é bastante rápido para células pequenas e médias, e não necessita nenhuma estrutura de dados auxiliar.

O método para calcular resistências tem como principais vantagens a generalidade e a precisão dos resultados. Também no cálculo das resistências se tomou bastante cuidado para usar a memória com o máximo de eficiência. O padrão usado para formar a rede de resistores mostrado no capítulo 6 foi criado especialmente para ser usado no extrator EXTRIBO.

A maior economia de memória, porém, se conseguiu a partir da extração hierárquica. O extrator hierárquico analisa apenas uma célula de cada vez, mantendo na memória apenas as interfaces externas das células já analisadas. Desta forma é possível realizar a extração de circuitos que não caberiam interamente na memória. Todos estes cuidados para poupar memória permitiram extraír com sucesso um circuito com mais de 1000 transistores com apenas 256Kbytes de memória livre. Deve ser possível realizar a extração de circuitos com mais de 2000 transistores em um PC com 640K. Os tempos de computação também foram bastante bons: O circuito de teste modem, com 524 transistores, levou 60 segundos na extração hierárquica em um PC AT (ver apêndice A4).

O conjunto de programas do extrator evoluiu através de várias versões mais simples, que foram usadas em projetos de circuitos integrados no CPGCC da UFRGS. Muitos dos

alunos do CPGCC. Um exemplo é o editor de máscaras EMA. O plano inicial era de apenas criar uma interface gráfica para o extrator. Porém um editor de máscaras hierárquico capaz de tirar proveito das placas gráficas EGA ou VGA estava sendo muito necessitado. O fato de as ferramentas estarem sendo utilizadas à medida que o programa foi sendo desenvolvido foi muito produtivo para o aperfeiçoamento do extrator.

O plano inicial era de criar o extrator como uma única ferramenta integrada. Foi necessário separar o extrator em 4 programas para permitir um aproveitamento eficaz da memória. Os programas que compõem o pacote ainda deverão ser integrados através de um sistema gráfico de menus [BAG 89], incluindo nesta integração várias outras ferramentas de CAD para microeletrônica desenvolvidas no CPGCC da UFRGS, formando uma estação de trabalho gráfica para projeto de circuitos integrados.

Apesar dos esforços para tirar o máximo de proveito dos recursos do PC, o programa também foi feito com vistas a poder ser portado para sistemas maiores. Todo o extrator foi escrito em linguagem C, evitando usar funções específicas do MSDOS. Versões futuras do extrator deverão ser portadas para estações de trabalho em sistema UNIX (*).

A possibilidade de se extraír um *netlist* de um circuito integrado inteiro abre caminho para a utilização de novas ferramentas para verificação de circuitos integrados. Programas para a comparação de *netlists* e simulação capazes de lidar com milhares de transistores deverão ser adquiridos ou escritos para aproveitar o potencial desta ferramenta.

(*) O extrator EXTRIBO já foi portado com sucesso para estações de trabalho SUN.

ANEXOS

ANEXO A1 - ARQUIVO DE DESCRIÇÃO DA TECNOLOGIA

O arquivo de descrição da tecnologia mantêm ainda certas características de versões antigas do extrator. Estas características permanecem apenas para garantir a compatibilidade com as versões anteriores.

Basicamente as referências a nomes de níveis são feitas no formato RS. Os nomes dos níveis na descrição CIF aparecem apenas em uma tabela de conversão.

A parte inicial do arquivo ainda é quase idêntica à definida para uso no antigo editor de máscaras de circuitos integrados usado no CPGCC da UFRGS, o MicroEditor [JAC85]. A primeira linha do arquivo de descrição da tecnologia informa qual é o tipo da tecnologia. Atualmente o tipo pode ser apenas CMOS ou NMOS. Segue o número de níveis básicos. Outros níveis poderão ser criados mais adiante usando a diretiva DEF. A seguir aparece a designação de cada nível (normalmente apenas uma letra no formato RS), seguida de um número que indica a cor a ser usada pelo editor EMA2. Nas oito linhas seguintes aparecem oito números binários de 8 bits que definem o padrão de pontos a ser usado para pintar os retângulos de cada nível quando o circuito for desenhado no vídeo pelo editor EMA2 ou na impressora pelo EMAPRINT.

A segunda parte do arquivo de tecnologia fornece as informações específicas para o extrator. Esta parte tem um formato bem mais flexível, sendo formada por um conjunto de diretivas. Estas diretivas são interpretadas pelo EXTRIBO seqüencialmente como uma linguagem de programação. Todas as diretivas iniciam com uma palavra chave que identifica o seu tipo. As diretivas disponíveis são as seguintes.

CIF <*nome* RS *do nível* > < *nome* CIF *do nível* >

A diretiva CIF serve para estabelecer a correspondência dos nomes dos níveis na descrição RS com os nomes em CIF. É necessária uma diretiva deste tipo para cada nível da tenologia. Com esta informação o editor EMA pode ler ou gravar circuitos descritos em CIF. As diretivas CIF não são obrigatórias, porém a sua falta faz com que os programas possam ler ou gravar apenas arquivos RS.

VDD *<nivel>*

Esta regra indica que o nível referenciado está necessariamente ligado a VDD.

GND *<nivel>*

Esta regra indica que o nível referenciado está necessariamente ligado a GND.

DEF *<novo nivel>* = *<nivel 1>* & *< nivel 2 >*

A regra DEF permite definir um novo nível como sendo a intersecção de dois níveis definidos anteriormente. Depois da diretiva DEF, o novo nível pode ser usado em qualquer outra regra (inclusive outra regra tipo DEF).

EXPANDE *<nivel>* *<expanção>*

A regra expande diz que todos os retângulos do nível especificado devem ser aumentados em todas as direções pelo valor da *<expanção>*.

APAGA *<nivel>*

Certos níveis de máscaras apenas prejudicam os resultados da extração. Estes níveis podem ser apagados usando esta diretiva.

CON *<nivel 1>* *<nivel 2>*

A palavra chave CON indica que os níveis 1 e 2 são mutuamente interligados. Normalmente são necessárias diversas regras do tipo CON para descrever a conectividade em uma determinada tecnologia.

MOS *<nome> <nível do gate> <nível do dreno> <nível do substrato> <acréscimo no comprimento efetivo do canal > < acréscimo na largura efetiva do canal>*

A regra MOS descreve como são formados os transistores. Logo após a palavra chave MOS vem o nome do modelo de transistor que está sendo descrito. Então vem o nível que forma o gate, o nível que forma dreno e fonte do transistor, e o nível do substrato, que é o que deve estar por trás da região de canal para que fique caracterizado o tipo de transistor. Em certos tipos de transistores, não deve aparecer nenhum retângulo por baixo como nível de substrato. Estes tipos de transistores devem ser os primeiros a serem especificados. Como nível de substrato deve aparecer uma letra não definida na tecnologia. Os acrés cimos na largura e comprimento efetivo do canal, são levados em consideração na geração do relatório em formato SPICE.

RES *<nível> <resistência superficial> <resistência de contato>*

A declaração RES define a resistência da superície do nível e a resistência de contato com outros níveis. Quando o extrator calcula as resistências de uma região, a resistência de contato é somada a cada resistor equivalente calculado. Ambos os valores são números reais em Ohms.

RTH *<resistência mínima a ser considerada>*

RTH (Threshold Resistance) é a resistência mínima que ainda deve ser considerada pelo extrator. Resistências menores que a definida em RTH são consideradas como ligações diretas. O valor de RTH é dado em Ohms e é válido para todos os níveis.

XCAP <*nível*> <*capacitância* ($\mu F/m^2$)>

XCAP especifica a capacitância por área do nível contra o substrato.

MCAP <*nível 1*> <*nível 2*> <*capacitância* ($\mu F/m^2$)>

MCAP especifica a capacitância por área da intersecção entre o nível 1 e o nível 2.

XCMIN <*capacitância em* FF (Femto Farad) >

XCMIN especifica qual é a menor capacitância que ainda deve ser considerada pelo extrator. Capacitâncias menores que XCMIN são eliminadas da lista de componentes.

LAMBDA < *dimensão em* μm >

As coordenadas do *layout* do circuito são especificadas em unidades λ parametrizadas. Para calcular corretamente as capacitâncias, dimensões dos transistores e áreas e perímetros, o extrator precisa converter as dimensões em unidades de λ para μm. Portanto as dimensões lineares são multiplicadas por λ e as áreas por λ^2.

FIM

A palavra chave FIM sem parâmetros indica o fim das regras de conectividade e formação dos transistores. A seguir vem dois textos englobados entre chaves { }. O primeiro deles é copiado no início do arquivo de saída do extrator, e o segundo no fim. Estes textos podem servir para definir os modelos dos transistores para o SPICE.

A1.1 Exemplo completo de um arquivo de tecnologia

```
CMOS
10
W 6
00000000
00000000
00000000
00010000
00000000
00000000
00000000
00000000
S 5
00000000
00000000
10000000
00000000
00000000
00000000
00001000
00000000
A 2
10001000
00100010
10001000
00100010
10001000
00100010
10001000
00100010
N 3
01000100
00000000
00100010
00000000
01000100
00000000
00100010
00000000
I 2
01000100
00000000
00100010
00000000
01000100
00000000
00100010
```

```
P 4
00100010
01000100
10001000
00010001
00100010
01000100
10001000
00010001
M 1
00100100
01000010
01000010
00100100
00100100
01000010
01000010
00100100
C 7
11111111
10011001
10011001
11111111
11111111
10011001
10011001
11111111
H 14
11110000
00001111
11110000
00001111
11110000
00001111
11110000
00001111
V 15
10001000
00100010
10001000
00000000
10001000
00100010
10001000
00000000
G 8
11001100
00110011
11001100
00110011
11001100
00110011
11001100
00110011
```

```
*-------------------------------------------------------------*
* DESCRICAO DA TECNOLOGIA CMOS COM 2 NIVEIS DE METAL DE GRENOBLE *
* PARA SER USADA PELO EXTRATOR EXTRIBO V 3.0                  *
*-------------------------------------------------------------*
* GDSII  CIF     RS      Descrição da camada                  *
*-------------------------------------------------------------*
*  01    CNWI    W       Poço tipo N
*  02    CTOX    A       Área ativa (óxido fino)
*  11    CPOL    P       Polisilício
*  12    CNPI    N       Implantação N+
*  14    CPPI    I       Implantação P+
*  16    CCON    C       Contato
*  17    CME1    M       Metal 1
*  18    CVIA    V       Via
*  19    CME2    H       Metal 2
*  20    CPAS    G       Passivação
*-------------------------------------------------------*
*          Tabela de tradução entre CIF e RS            *
*-------------------------------------------------------*
CIF W CNWI
CIF A CTOX
CIF P CPOL
CIF N CNPI
CIF I CPPI
CIF C CCON
CIF M CME1
CIF V CVIA
CIF H CME2
CIF G CPAS
*-------------------------------------------------------*
*        Definição do fator de escala λ em μm           *
*-------------------------------------------------------*
LAMBDA 1.0
*-------------------------------------------------------*
*            Localização da Alimentação                 *
*-------------------------------------------------------*
VCC S
GND W
*-------------------------------------------------------*
*            Definição de novos níveis                  *
*-------------------------------------------------------*
* Difusão N+: intersecção entre área ativa (A) e implante N+
LET DN = A & N
* Difusão P+: intersecção entre área ativa (A) e implante P+
LET DP = A & I
* Os níveis I, N e A não são mais necessários.
* Devem ser apagados.
APAGA I
APAGA N
APAGA A
```

```
*-------------------------------------------------------*
*              Regras de Conectividade                  *
*-------------------------------------------------------*
CON DN W
CON DN C
CON DP C
CON P  C
CON M  C
CON M  V
CON H  V
*-------------------------------------------------------*
*      Regras de Formação dos Transistores:             *
*MOSFET <nome> <gate> <drain> <substr> <eff. len> <eff. widt>*
*-------------------------------------------------------*
MOSFET NMOS P DN W 0.3 0.3
MOSFET PMOS P DP X 0.5 0.3
*-------------------------------------------------------*
*      Resistência Mínima a ser Considerada (Ohm)       *
*-------------------------------------------------------*
RTH 100
*-------------------------------------------------------*
*      Resistência Laminar e Resistência de Contato (Ohm)*
*-------------------------------------------------------*
RES DN  40.0   35.0
RES DP  50.0   25.0
RES P  22.0   20.0
RES M   0.1    0.0
RES H   0.05   0.2
*-------------------------------------------------------*
*      Capacitância mínima a ser considerada (fF)       *
*-------------------------------------------------------*
XCMIN 1.0
*-------------------------------------------------------*
*         Capacitância em uF/m2 de cada nível           *
*-------------------------------------------------------*
XCAP DN  110
XCAP DP  350
XCAP P  36.0
XCAP M  18.6
XCAP H  12.5
XCAP & 860.0
FIM
```

ANEXO A2 - INSTRUÇÕES DE USO DOS PROGRAMAS

O pacote do extrator deve incluir os seguintes arquivos:

*** Programas ***

EXPREP.EXE - Pré-processador para encontrar as interfaces externas das células.

EXTRIBO.EXE - Extrator.

EMA2.EXE - Editor de Máscaras e interface gráfica do extrator.

EMAPRINT.EXE - Programa para desenhar o circuito na impressora gráfica.

*** Arquivos auxiliares ***

*.TEC - Arquivos de descrição da tecnologia.
*.CEL - Descrição RS do circuito.
*.BGI - Drivers para os dispositivos gráficos (CGA.BGI, EGAVGA.BGI...). Estes drivers são fornecidos pela Borland International junto com as bibliotecas gráficas dos compiladores Turbo C. Eles são usados no programa EMA2.

A2.1 Chamada do extrator

A extração hierárquica de um circuito é feita inicialmente executando o programa EXPREP.EXE. Suponhamos que o nome do circuito é CIRCUITO.RS, e o arquivo de descrição da tecnologia é TECNOL.TEC. Então a chamada do programa é feita a partir do DOS através da linha de comando:

EXPREP CIRCUITO.RS TECNOL.TEC

As extensões .RS e .TEC são usadas como *default*.

O programa mostra o nome da célula do circuito que está sendo processada no momento. Ao final da execução é gerado o arquivo CIRCUITO.RNK. Este arquivo contém as informações de interfaces entre células necessárias para a segunda parte da extração. O arquivo CIRCUITO.RNK pode ser visualizado usando o EMA2 ou o EMAPRINT. A extração do circuito então é feita chamando o programa EXTRIBO:

EXTRIBO CIRCUITO.RNK TECNOL.TEC <*opções de linha de comando*>

As extensões .RNK e .TEC são default, e podem ser omitidas. As opções de linha de comando podem estar presentes ou não. Elas são as seguintes (as letras podem ser maiúsculas ou minúsculas):

-R A presença desta opção indica que deve ser feita a extração com resistências e capacitâncias.

-S<*nome*> O programa deverá gerar um relatório em formato SPICE. Será criado um arquivo com o nome especificado, usando a extensão .CIR como default. Caso o nome não seja especificado, será criado um arquivo com o mesmo nome do circuito original, com a extensão .CIR (no exemplo usado: CIRCUITO.CIR). Se a opção -S for omitida, o arquivo SPICE não será gerado.

-A<*nome*> A opção -A é similar à opção -S, porém requisitando a geração de um arquivo em formato ARAMOS. A extensão *default* é .ARA.

-E<*nome*> A opção -E requisita a geração de um arquivo de descrição do *layout*, com os números dos nós e a localização dos componentes encontrados pelo extrator. Este arquivo é para ser lido pelo programa EMA2 ou EMAPRINT. A extensão *default* é .EMA.

-B<*nome*> Com a opção -B, o extrator gera um netlist em formato binário, (<*nome*>.EX7) para ser lido pelo pós-processador. A extração de circuitos hierárquicos é feita normalmente usando os comandos:

EXPREP <*nome do circuito*> <*tecnologia*>
EXTRIBO <*nome do circuito*> <*tecnologia*> -b
EX7POS <*nome do circuito*>

Exemplo:

```
EXPREP CIRCUITO.RS TECNOL
EXTRIBO CIRCUITO TECNOL -B -E
EX7POS CIRCUITO
```

O resultado deste exemplo será um arquivo CIRCUITO.RNK, produzido pelo pré-processador, indicando o número de promoções que os retângulos com inteface externa devem ter; um arquivo CIRCUITO.EMA, que é uma descrição do layout anotada pelo extrator com os números dos nós e os nomes dos transistores; um arquivo CIRCUITO.EX7, que é uma descrição do netlist em formato binário, gerada pelo EXTRIBO para ser lida pelo pós-processador EX7POS; e um arquivo CIRCUITO.CIR, que é o netlist em formato SPICE.

A seleção das opções é feita completamente pela linha de comando. Isto torna fácil automatizar a seqüência de chamadas de programas para extraír um circuito usando um arquivo BATCH.

Emitindo o comando EXTRIBO, a partir do DOS sem nenhum argumento, o programa vai gerar na tela, uma página de ajuda listando as opções de linha de comando e descrevendo o seu significado.

A2.2 Uso do emaprint

O programa EMAPRINT pode receber todos os dados necessários na linha de comando, ou perguntar por eles durante a execução do programa. A forma geral da chamada do programa é:

EMAPRINT <*nome do circuito* [.CEL]> <*arquivo tecnologia* [.TEC]> [<*opções*>]

O nome do circuito tem a extensão *default* .CEL, mas pode ser usado o CIRCUITO.EMA como dado de entrada, quando se quer ver no papel os números de nós e os componentes encontrados pelo extrator. As opções são as seguintes:

-C<*número de colunas da impressora*>

Este comando especifica qual é a largura do papel. O número de colunas deve ser 80 ou 132. Se a opção -C for omitida ou inválida, o programa assumirá que a largura do papel é 80 colunas.

-R<*resolução*>

Esta opção seleciona o modo de resolução da impressora. A resolução pode ser simples (-R1) ou dupla (-R2). Outros valores de densidade não são considerados. A resolução dupla é assumida em caso omisso.

A resolução, combinada com a largura do papel determina o número máximo de pontos da impressora que cabem em uma linha.

480 na impressora de 80 colunas em resolução simples.
960 na impressora de 80 colunas em resolução dupla.
816 na impressora de 132 colunas em resolução simples.
1632 na impressora de 132 colunas em resolução dupla.

-L<*largura*>

Esta opção especifica que largura, em número de caracteres da impressora deve ser aproveitada no desenho. Em caso de omissão, a largura do desenho será igual à largura do papel. Se a largura especificada para o desenho for maior que a largura do papel, o desenho será feito em partes, que poderão ser recortadas e coladas.

-M<*margem esquerda*>

A margem esquerda é especificada em número de caracteres da impressora. Esta opção permite que o desenho fique deslocado para a direita no papel. Em caso de omissão,é assumido o valor zero para a margem esquerda.

-H Posição horizontal. Esta opção diz que o circuito deve ser desenhado na mesma posição em que ele foi criado no editor.

-V Posição vertical. Faz com que o circuito seja desenhado com uma rotação de 90° em relação à sua posição original. A opção -H é usada nos casos omissos.

-U<*profundidade*>

A profundidade é um número inteiro que especifica quantos níveis hierárquicos de profundidade devem ser expandidos. A profundidade é o número de células aninhadas que são expandidas. Quando a profundidade especificada é atingida, o programa apenas desenha os envelopes das chamadas de células mais internas. Portanto profundidades menores que o número máximo de chamadas de células aninhadas do circuito faz com que o circuito não seja completamente expandido. Isto muitas vezes é conveniente, quando se quer ver o circuito em um nível de abstração mais elevado.

-D<*lista de níveis*>

Esta opção especifica uma lista de níveis que serão desenhados. Desenhar um nível neste caso é marcar o contorno dos retângulos deste nível. A lista de níveis é especificada usando os nomes das camadas usados na descrição RS, como uma seqüência de caracteres. A opção -DT faz com que todos os níveis sejam desenhados. Se a opção -D for omitida, o programa fará a pergunta:

Desenhar níveis (<*lista dos níveis da tecnologia*>)?

A lista de níveis da tecnologia aparece como um lembrete.

-P<*lista de níveis*>

A opção -P (Pintar) funciona de forma similar à opção -D. Pintar um nível para o EMAPRINT significa preencher o interior dos retângulos deste nível com o padrão de bits correspondente, definido no arquivo de tecnologia.

-N<*lista de níveis*>

Numerar os níveis: O funcionamento da opção -N é similar ao da -D e da -P. Através desta opção são selecionados os níveis a serem numerados. Numerar um nível significa mostrar dentro do desenho de cada retângulo o seu número de nó.

ANEXO A3 USO DO EDITOR DE MÁSCARAS EMA

O editor EMA exige para ser usado os seguintes arquivos, além da desrição RS do circuito a ser editado:

*.TEC - Arquivo de descrição da tecnologia.

*.BGI - Interfaces gráficas: Dpendendo do tipo de vídeo do computador é necessária uma das seguintes interfaces gráficas: CGA.BGI, EGAVGA.BGI, HERCULES.BGI, IBM8514.BGI, ATT400.BGI ou PC3270.BGI. Estes drivers gráficos são fornecidos pela Borland junto com o compilador TURBOC 2.0.

A3.1 Chamada do programa

Ao chamar o EMA podem ser postos dois nomes de arquivo na linha de comando da seguinte forma:

C>EMA <*nome da célula a ser editada*[.CEL]>
<*arquivo tecnologia* [.TEC]>

As extensões .CEL e .TEC são default. O nome da célula a ser editada pode ser omitido. Neste caso o editor começará com o buffer de edição vazio. Se o nome do arquivo tecnologia for omitido, ele será solicitado explicitamente pelo programa.

A3.2 Comandos básicos de edição

No modo de edição se pode visualizar uma janela do circuito na maior parte da tela, tendo à direita um menu e informações sobre os níveis a serem desenhados ou pintados e a posição do cursor. O cursor pode ser movimentado usando as teclas de flexas. O tamanho de cada passo, indicado como INC na janela de status pode ser duplicado usando a tecla "+" (sinal de mais) ou reduzido à metade usando a tecla "-" (sinal de menos). Sempre que o cursor bater em uma das bordas da janela de visualização do circuito, a janela será

mudada de lugar de modo que o cursor fique no meio da janela, e o circuito será redesenhado. Pode-se também forçar o programa a atualizar a janela de visualização usando a tecla de espaço " ".

As opções do menu que aparecem à direita da tela são selecionadas acionando a tecla correspondente à letra inicial de cada palavra. A seguir são apresentados algus dos principais comandos do editor:

<ESC> - Cancela qualquer comando pendente.

flexas, home, pgup, end, pgdn - Movimentam o cursor.

<ENTER> - Esta tecla, quando acionada no modo normal de movimentação do cursor, inicia a criação de um retângulo. A tecla <ENTER> também é usada para concluir outros comandos pendentes.

A - "Abre" - Permite abrir (ou fechar) espaços no circuito. Quando o comando Abre for concluído (teclando <ENTER>) toda a porção do circuito que estiver à direita do cursor será movida tanto quanto o cursor tiver sido movido enquanto o comando estava ativo. Se o cursor tiver sido movido para a direita, será aberto um espaço no circuito, que poderá ser usado para incluir novos componentes. Se o cursor tiver sido movido para a esquerda, será fechado o espaço percorrido. Este comando também funciona na vertical, abrindo espaço quando o cursor é movido para cima e fechando o espaço quando o cursor é movido para baixo.

B - "Bloco" - Permite definir uma área retangular do circuito. Esta área poderá então ser apagada, copiada ou movida para outro lugar. Ao concluir a definição do bloco com <ENTER>, se poderá selecionar uma das 4 operações de blocos disponíveis: Remover, Mover, Copiar ou Definir nova célula; através das letras iniciais (R, M, C ou D).

C - Célula - Chama o menu de operações com células.

D - "Desenhar" - Seleciona os níveis que serão desenhados.

E - "Escala" - A escala determina qual é o tamanho da janela.

F - File - Chama o menu de controle de arquivos.

I - Incremento - Ajusta o tamanho dos incrementos em x e y.

M - "Modifica" - Modifica o retângulo do nível corrente em que o cursor está dentro. Este comando se subdivide em 2 opções: Dimensão - Altera apenas a posição do vértice mais próximo do cursor;
Posição - Move todo o retângulo de acordo com o cursor.
O comando M fica pendente até que seja apertada a tecla <ENTER>, que tornará as alterações definitivas.

N - "Nível" - Permite alterar o nível corrente.

O - Opções - Chama o submenu de seleção de opções.

P - "Pintar" - Permite especificar que níveis devem ser pintados.

R - "Remove" - Remove o retângulo sob o cursor.

X - Este comando permite especificar uma nova coordenada X do cursor como um dado numérico. Respondendo à pergunta "Nova coordenada X=" com um valor numérico, o cursor será movido para a nova coordenada, e a tela será atualizada, se necessário. Se o valor numérico especificado for precedido pelo sinal "+", o valor dado será somado com a coordenada X atual para dar a nova coordenada X do cursor. Veja o exemplo a seguir:

Coordenadas atuais: X=200 Y=100
Dados fornecidos ao programa com o comando "X":
Nova coordenada X=+40
coordenadas resultantes: X=240 Y=100

Y - O comando Y é similar ao comando X, porém atuando sobre a coordenada y do cursor.

Z - Zahlen - Seleciona quais níveis deverão ser numerados de acordo com os números de nós usados pelo extrator. Este comando só é ativo com os arquivos tipo *.ELE gerados pelo extrator.

T - Texto - Ativa o submenu de texto. Se pode criar um texto de até 16 caracteres em qualquer lugar do circuito. Os textos também podem ser eliminados, ou modificados. Uma das opções do menu de texto permite procurar um texto no circuito, localizando o cursor sobre o texto encontrado.

A3.3 Submenus

A3.3.1 Submenu de CÉLULA - Este submenu é acessado usando a tecla "C" a partir do ambiente normal de edição. São apresentadas as opções descritas a seguir:

Abre - Abre uma célula para edição. Selecionando a opção ABRE, aparece um menu com toda a lista dos nomes e números de todas as células presentes na memória. Se o espaço não for suficiente para mostrar todas elas, serão mostradas aquelas que couberem, havendo a possibilidade de rolar a janela do menu (*scroll*) quando o cursor tocar em uma das extremidades do menu. Uma das células poderá ser selecionada posicionando o cursor com as teclas de flexas e pressionando <ENTER> na opção desejada. A célula selecionada será a nova célula aberta para edição.

Numera - Esta função permite selecionar uma das células para receber um novo número usando o mesmo tipo de menu que a função ABRE. A célula selecionada poderá ser designada pelo novo número especificado neste comando.

Instancia - Chama uma instância de célula. As células são selecionadas em um menu com o nome e o número ou em uma janela com o desenho da célula. A transformação é determinada através de outro menu, com a visualização imediata do efeito da transformação.(F3 é equivalente).

Compacta - O comando C procura encontrar uma forma equivalente de representar o layout com um número menor de retângulos, evitando assim que os retângulos fiquem fragmentados. Retângulos completamente sobrepostos em um mesmo nível são eliminados e retângulos adjacentes com a mesma largura são transformados em um só.

Cria célula - O usuário informa o nome e o número da nova célula, e ela é criada e aberta para edição.

Profundidade - Este comando permite especificar até que profundidade a descrição hierárquica deve ser expandida quando o programa desenha o circuito. O programa que desenha o circuito expande células que tem chamadas de células, que também podem ter chamadas de células, etc, até que a profundidade especificada é atingida. Quando isto acontece, apenas os retângulos envolventes das células chamadas naquele nível hierárquico são desenhados.

A3.3.2 Submenu de ARQUIVOS (FILES) - Selecionado usando a letra F a partir do modo normal de edição. As opções são as seguintes:

Grava - Grava em disco uma descrição do circuito inteiro usando o formato RS - EMHIR. Se não for fornecido um novo nome para o arquivo, será gerado um arquivo ????.BAK correspondente à versão anterior, e a nova versão será gravada com o mesmo nome pelo qual foi lida. Os comandos GRAVA, CARREGA, LE CELULA e SALVA CELULA chamam um menu para selecionar o formato de leitura ou gravação, com as opções CIF e RS.

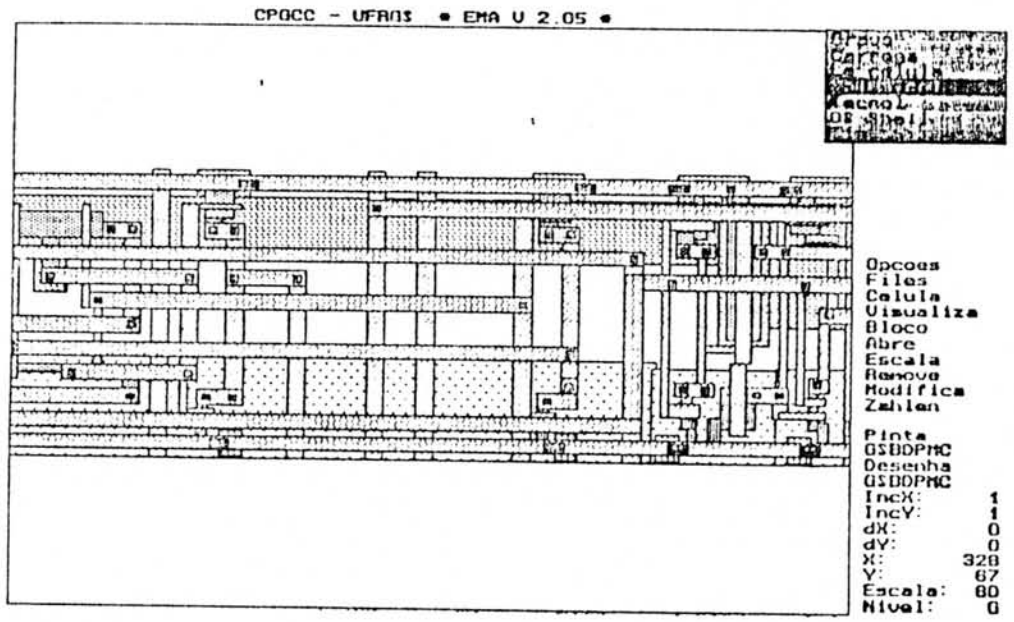


Figura A3.2 - Tela do editor com o menu de arquivos

Carrega - Carrega na memória a partir do disco um novo circuito para edição. Tudo o que estava na memória antes de carregar o novo circuito é apagado. O nome do arquivo a ser carregado pode ser fornecido de três maneiras: Se for fornecido um nome completo de arquivo, com nome e extensão, este arquivo será lido, se for encontrado (são admitidas especificações de drives e caminhos para diretórios). Se o nome do arquivo for fornecido sem extensão, será assumida a extensão .CEL como default. Se o nome do arquivo for fornecido usando *coringas* (* ou ?) do DOS, (e.g.: *.CEL ou *.*) será apresentado um menu na direita da tela contendo os nomes dos arquivos do diretório corrente que satisfazem a esta especificação. O arquivo a ser lido poderá então ser escolhido movendo a barra seletora com as flexas ↑ ↓, e consumando a escolha com a tecla <ENTER>. Se apenas um arquivo satisfizer a especificação com coringas, ele será lido automaticamente sem que apareça o menu do diretório.

Le célula - Lê uma célula a partir do disco, acrescentando-a às que já estão na memória. Este comando é útil para criar um circuito a partir de uma bibioteca de células em disco. A forma de selecionar o arquivo em disco a partir do qual a célula será lida é feita da mesma maneira que no comando CARREGA.

Salva célula - Este comando permite gravar um subconjunto das células presentes na memória do computador como um arquivo separado. Selecionando esta opção obtém-se um menu listando todas as células pelo número e nome, semelhante ao das funções do menu de Célula. A diferença é que este menu é de múltipla escolha. Se pode correr o cursor sobre a lista de células usando as flexas. Usa-se <ENTER> para marcar as células a serem incluídas no arquivo. <ESC> sai do menu sem fazer nada. <ESPAÇO> faz com que o programa pegunta qual deve ser o nome do arquivo onde as células selecionadas serão gravadas. Esta função é útil para quebrar um circuito

Tecnologia - Esta função serve para ler um outro arquivo de descrição da tecnologia de dentro do editor. A seleção do arquivo a ser lido como descrição da tecnologia pode ser feita usando um menu de diretório que funciona da mesma maneira que nos comandos para leitura de um circuito ou células de um circuito, porém usando a extensão .TEC como default.

OS Shell - Chama o interpretador de comandos do MSDOS. Esta opção permite realizar comandos do MSDOS ou executar outros programas sem perder o trabalho que está sendo feito no editor. Se pode voltar para o editor usando o comando EXIT.
Fim - Abandona o programa.

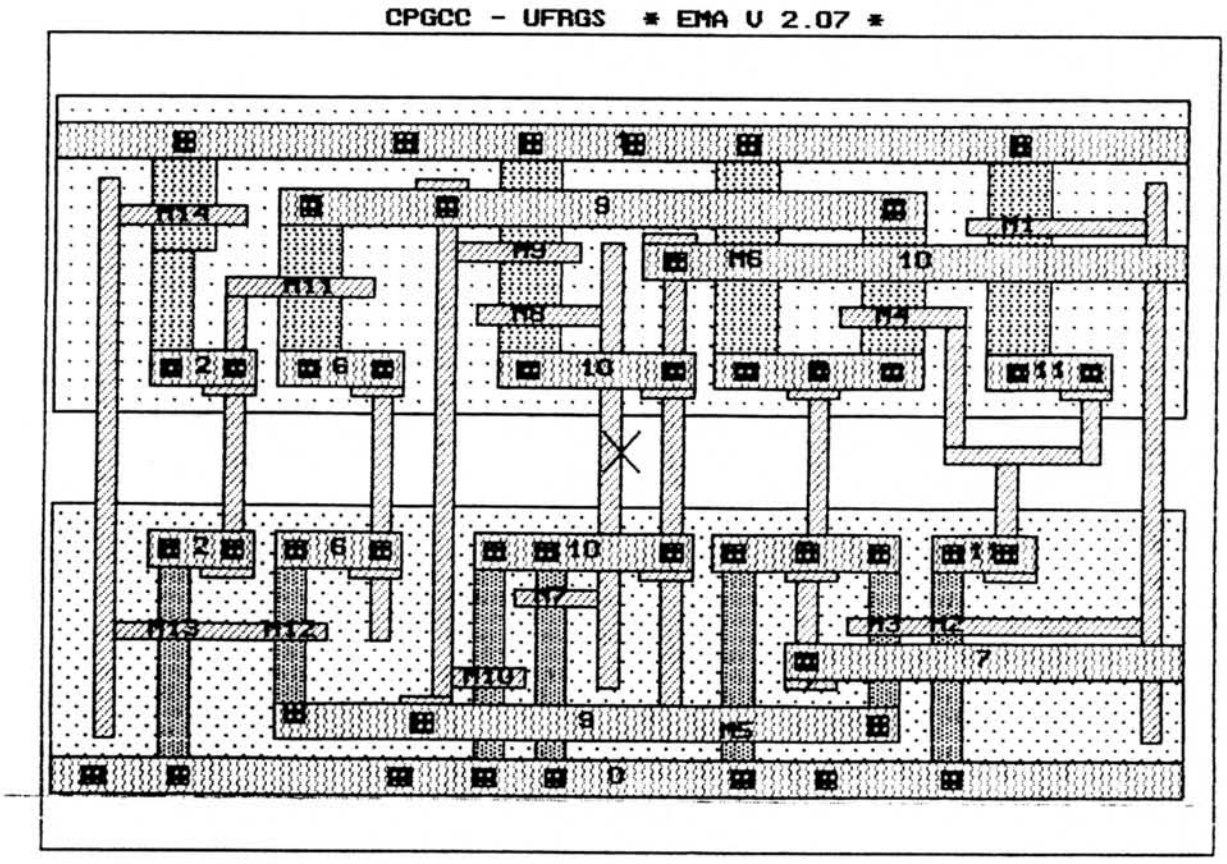


Figura A3.3 - Desenho do circuito com números de nós e identificação de transistores.

A3.3.3 Submenu de VISUALIZAÇÃO

No submenu de visualização estão as opções que fazem com que o EMA seja realmente uma interface gráfica do extrator, além de ser um editor de máscaras. As opções disponíveis são:

Numerar niveis - Permite fornecer uma lista dos níveis a serem numerados com os números de nós usados pelo extrator. Também são mostrados os nomes dos transistores.

Ver Resistências - As resistências criadas pelo extrator são desenhadas no circuito.

Identificar R - A resistência mais próxima do cursor é identifiada na linha de texto. Nesta linha aparecem o nome da resistencia, os números dos nós terminais e o seu valor.

A3.3.4 Submenu de TEXTO

No submenu de texto estão as opções que permitem associar nomes a locais do circuito. As opções disponíveis permitem criar um texto na posição atual do cursor, apagar ou modificar o texto mais próximo do cursor, ou procurar um texto, localizando o cursor sobre o texto encontrado. O extrator é capaz de ler estes textos e associá-los a números de nós.

Através do menu de textos, se pode também auxiliar o extrator na identificação da alimentação, colocando os nomes VDD e GND nos devidos lugares do layout.

A3.4 Comandos hierárquicos

F1 - Passa para a célula anterior na lista de células presentes na memória. O nome e número da célula será mostrado na parte superior da tela. Esta será a nova célula corrente em edição.

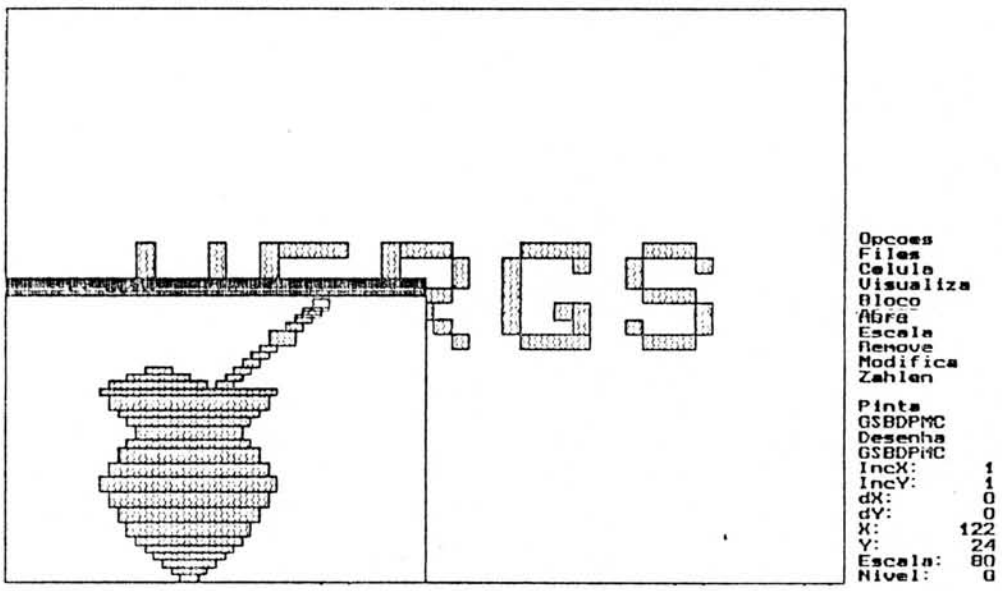


Fig A3.4 - Janela de seleção de células para instanciação.

F2 - Passa para a célula seguinte na lista de células na memória. Este é o comando inverso de F1.

F3 - Cria instância: Este comando permite chamar uma instância de qualquer uma das células cuja definição esteja presente na memória, desde que isto não implique em uma chamada recursiva. Ao chamar o comando F3, aparecerá na parte inferior esquerda do vídeo uma janela com a imagem da célula a ser escolhida, com um cabeçalho mostrando o número

poderá ser selecionada usando as teclas F1 e F2 para retroceder ou avançar na lista de células. Pressionando <ESC> a janela será retirada e a execução do comando será suspensa. Pressionando <ENTER>, será criada uma instância da célula selecionada na posição atual do cursor (o cursor ficará no canto superior direito da célula), a janela de seleção será removida, e aparecerá o menu de transformações de instâncias. Quando uma célula é selecionada desta maneira, o programa procura reordenar as definições de células em ordem ascendente de modo que cada célula tenha apenas instâncias de células definidas anteriormente. Se não for possível colocar as células em ordem ascendente na hierarquia, será dada a mensagem de erro "Definição recursiva", e o comando será cancelado.

Quando uma célula foi selecionada desta forma, ela aparece no circuito em edição, de modo que o seu canto inferior esquerdo fique na posição do cursor. Neste momento é ativado o menu de seleção de transformação, com as seguintes opções:

(R)otação espelhamento (X ou Y) (T)ranslada (P) repete X (Q) repete Y

R - Faz a célula girar 90 graus em sentido anti-horário, mantendo o canto inferior direito na posição atual do cursor.

X ou Y - Espelha a célula segundo as coordenadas X ou Y.

T - Translada - Permite mudar o lugar onde a célula está sendo instanciada. Ao emitir este comando aparece o envelope da célula, que poderá ser movimentado usando os comandos normais de movimentação do cursor (inclusive os comandos X e Y). Quando o envelope estiver posicionado no lugar desejado, se deve concluir com <ENTER>, voltando para o menu de seleção de transformação.

P - Repetição em X; Q - Repetição em Y - Os comando P e Q servem para criar um array instâncias de células. Depois de fornecer o número de repetições, a distância entre os elementos do array é definida através de movimentos do cursor, que são concluídos com <ENTER>, de forma similar ao comando "T" de translação de células.

<ENTER> - Emitido de dentro do menu de definição de transformação, conclui a criação da instância, de modo que o editor volta ao modo normal de operação, e a nova instância da célula é criada.

<ESC> - A tecla ESCAPE usada de dentro do menu de seleção de transformação cancela a criação da instância. O editor volta ao modo normal e nenhuma instância é criada.

F4 - Modifica chamada de célula - Este comando permite alterar a transformação aplicada sobre uma instância de célula já presente no circuito, ou eliminar uma chamada de célula. A chamada a ser modificada é selecionada posicionando o cursor sobre ela e chamando F4. O envelope da célula selecionada será posto em destaque e será feita a pergunta:

Modificar instância da célula: <número>: <nome> (S/N)?

Respondendo 'S' o cursor passará para o canto inferior esquerdo da instância selecionada e o editor passará ao menu de seleção da transformação, que já foi descrito para a função F3. O menu de transformação vai funcionar como se a instância selecinada tivesse sido recém criada.

Se houverem diversas instâncias sobrepostas no lugar onde o cursor estava quando se pressionou F4, a instância desejada poderá ser selecionada respondendo não [illegible] até que o nome, número ou envelope da

também ser cancelado sumariamente respondendo a pergunta com <ESC>. O comando F4 pode ser usado para apagar uma chamada de célula, pressionando <ESC> quando o programa estiver no menu de seleção de transformação.

F5 - Cria nova definição de célula - Para criar uma nova definição de célula usa-se a função F5, especificando o nome e o número da nova célula. Feito isto o editor continuará na célula que estava sendo editada. Para editar a nova célula, é necessário procurá-la usando as teclas F1 ou F2, depois de te-la criado.

F6 - Modifica ou apaga definição de célula - Esta função permite mudar o nome e o número de uma célula. A célula a ser modificada é sempre a célula em edição no momento. Se a célula em edição estiver completamente vazia (se ela não tiver nenhum retângulo ou instância de célula), apertando a tecla <ESC> enquanto está sendo pedido o seu nome ou número em conseqüência do comando F6 fará com que esta célula seja apagada da lista de células na memória. Quando se tenta alterar o número da célula, o programa verifica se este número já foi usado em outra célula. Em caso afirmativo, o programa dará a mensagem de erro "Número já usado em outra célula", e manterá o número inalterado. Quando o número de uma célula é alterado, a consistência do resto do circuito permanece inalterada, porque em todas as referências a esta célula o número irá automaticamente mudar de acordo.

ANEXO 4 EXEMPLO DE EXTRACAO HIERARQUICA

O circuito usado como exemplo é o um ASIC para modem [REI 89], desenvolvido por André Reis, Fernando Moraes e Ricardo Reis. O circuito foi feito usando a biblioteca de células do projeto TRANCA [LUB 89], [REI 87], em tecnologia CMOS no CPGCC da UFRGS. O circuito é de tamanho médio, com 524 transistores.

O circuito foi projetado usando vários níveis hierárquicos intermediários. Ao ser submetido ao pré processador, com a simplificação da estrutura hierárquica, o circuito passou a ter apenas dois níveis: as definições das células e a célula que forma o circuito inteiro através de chamadas de células. A chamada do pré processador foi feita com o comando:

```
EXPREP MODEM.RS GREN2
```

O pré processador levou 23 segundos em um computador tipo PC AT de 12 MHz para gerar o arquivo MODEM.RNK, que informa quais são as interfaces externas das células e tem a estrutura hierárquica simplificada.

A extração, usando o módulo EXTRIBO, é feita depois de obtido o resultado do pré processador através do comando:

```
EXTRIBO MODEM GREN2 -B -E
```

O programa assumiu como default as extenções .RNK para o circuito e .TEC para a tecnologia. As opções -S e -B significam que o extrator deve gerar um arquivo de saída MODEM.EX7, em formato binário, e um arquivo MODEM.ELE, para ser lido pelo editor EMA para a visualização dos componentes do circuito no layout. Durante a extração, o EXTRIBO fornece um grande número de informações, que servem para deixar o usuário ao par do andamento da extração. O extrator levou 40 segundos para realizar a extração.

O arquivo de simulação MODEM.CIR, em formato SPICE é obtido com o uso do pós-processador EX7POS, através do comando:

EX7POS MODEM

A seguir é apresentdo o desenho do layout do modem e o netlist hierárquico em formato SPICE gerado pelo extrator.

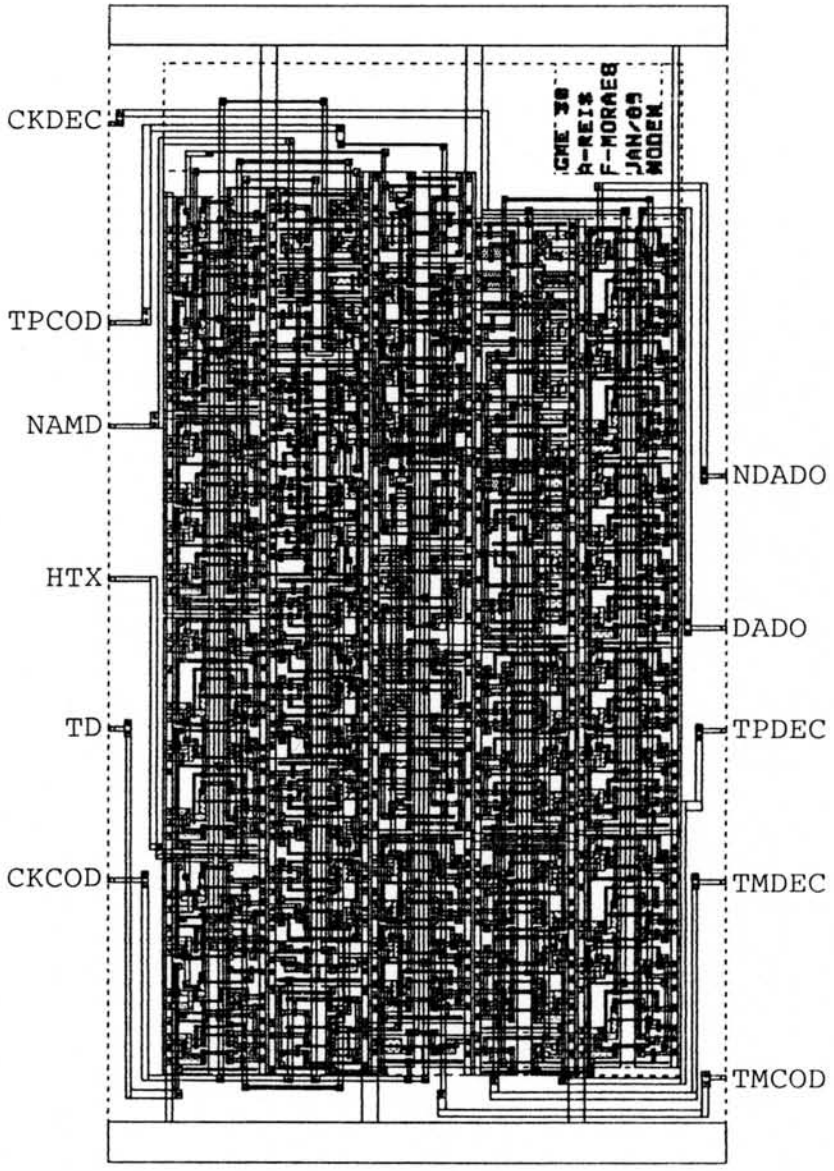


Figura A4.1 - Layout do circuito integrado usado como

```
*
* UFRGS - PGCC - GRUPO DE MICROELETRONICA
* EXTRATOR DE ARQUIVO DE SIMULACAO SPICE VERSAO 3.0
*
* CIRCITO: modem\modem
* TECNOLOGIA: gren2.tec TIPO CMOS
*
* Modelos dos transistores
*
.MODEL NMOS NMOS LEVEL=2 LD=0.15U TOX=440E-10 NSUB=0.95E16 VTO=1.15
+UO=693 UEXP=0.111 UCRIT=10K DELTA=1.68 XJ=1U VMAX=41K NEFF=1.16 RSH=45
+NFS=1E11 JS=100U CJ=105U CJSW=240P MJ=0.48 MJSW=0.27 PB=0.45V
+CGDO=270P CGSO=270P
*
.MODEL PMOS PMOS LEVEL=2 LD=0.25U TOX=440E-10 NSUB=3.24E16 VTO=-0.8
+UO=271 UEXP=0.131 UCRIT=10K DELTA=0.89 XJ=2U VMAX=32K NEFF=0.77 RSH=80
+NFS=1E11 JS=100U CJ=330U CJSW=430P MJ=0.48 MJSW=0.4 PB=1.04V
+CGDO=350P CGSO=350P
*
* CELULA FFD_MS
*
* Transistores tipo NMOS: 13
* Transistores tipo PMOS: 13
*
.SUBCKT FFD_MS 0 1 9 11 13 16 17
MP1    3 14  1   1 PMOS L=2.0U W=7.0U AD=169P AS=63P PD=82U PS=32U
MP2    9 17  3   1 PMOS L=2.0U W=7.0U AD=63P AS=169P PD=32U PS=82U
MN3    7 16  0   0 NMOS L=3.0U W=4.0U AD=12P AS=60P PD=14U PS=36U
MN4    7 14  9   0 NMOS L=3.0U W=4.0U AD=12P AS=114P PD=14U PS=56U
MN5    9 17  0   0 NMOS L=3.0U W=4.0U AD=114P AS=144P PD=56U PS=78U
MP6    3 16  1   1 PMOS L=2.0U W=7.0U AD=169P AS=112P PD=82U PS=46U
MN7   17  9  0   0 NMOS L=3.0U W=4.0U AD=160P AS=160P PD=80U PS=76U
MP8    2 16  1   1 PMOS L=2.0U W=7.0U AD=191P AS=63P PD=78U PS=32U
MP9   17  9  2   1 PMOS L=2.0U W=7.0U AD=63P AS=191P PD=32U PS=78U
MN10   6 15 17   0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=80U
MN11   6 16  0   0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=76U
MP12   2 15  1   1 PMOS L=2.0U W=7.0U AD=191P AS=112P PD=78U PS=46U
MP13   8 11  1   1 PMOS L=2.0U W=7.0U AD=180P AS=56P PD=86U PS=30U
MN14   5 13  0   0 NMOS L=3.0U W=4.0U AD=12P AS=60P PD=14U PS=36U
MN15   5 11 15   0 NMOS L=3.0U W=4.0U AD=12P AS=114P PD=14U PS=56U
MP16 15 14  8   1 PMOS L=2.0U W=7.0U AD=63P AS=180P PD=32U PS=86U
MN17 15 14  0   0 NMOS L=3.0U W=4.0U AD=114P AS=144P PD=56U PS=78U
MP18   8 13  1   1 PMOS L=2.0U W=7.0U AD=180P AS=105P PD=86U PS=44U
MN19 14 15  0   0 NMOS L=3.0U W=4.0U AD=160P AS=160P PD=80U PS=76U
MP20 10 13  1   1 PMOS L=2.0U W=7.0U AD=191P AS=63P PD=78U PS=32U
MP21 14 15 10   1 PMOS L=2.0U W=7.0U AD=63P AS=191P PD=32U PS=78U
MN22   4 12 14   0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=80U
MN23   4 13  0   0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=76U
MP24 10 12  1   1 PMOS L=2.0U W=7.0U AD=191P AS=112P PD=78U PS=46U
MN25 12 11  0   0 NMOS L=3.0U W=4.0U AD=72P AS=100P PD=42U PS=56U
MP26 12 11  1   1 PMOS L=2.0U W=7.0U AD=203P AS=63P PD=72U PS=32U
.ENDS FFD_MS
```

```
*
* CELULA NOR2
*
* Transistores tipo NMOS: 2
* Transistores tipo PMOS: 2
.SUBCKT NOR2 0 1 2 4 5
MP1    3  5  1  1 PMOS L=2.0U W=7.0U AD=35P AS=63P PD=24U PS=32U
MP2    2  4  3  1 PMOS L=2.0U W=7.0U AD=154P AS=35P PD=58U PS=24U
MN3    2  5  0  0 NMOS L=3.0U W=4.0U AD=152P AS=168P PD=76U PS=78U
MN4    2  4  0  0 NMOS L=3.0U W=4.0U AD=152P AS=168P PD=76U PS=78U
.ENDS NOR2
*
* CELULA NAND4
*
* Transistores tipo NMOS: 4
* Transistores tipo PMOS: 4
*
.SUBCKT NAND4 0 1 4 6 7 8 9
MP1    4  9  1  1 PMOS L=2.0U W=7.0U AD=195P AS=243P PD=86U PS=104U
MP2    4  8  1  1 PMOS L=2.0U W=7.0U AD=195P AS=243P PD=86U PS=104U
MP3    4  7  1  1 PMOS L=2.0U W=7.0U AD=195P AS=243P PD=86U PS=104U
MP4    4  6  1  1 PMOS L=2.0U W=7.0U AD=195P AS=243P PD=86U PS=104U
MN5    2  7  3  0 NMOS L=2.0U W=8.0U AD=168P AS=64P PD=88U PS=32U
MN6    2  9  4  0 NMOS L=2.0U W=8.0U AD=168P AS=116P PD=88U PS=64U
MN7    5  6  3  0 NMOS L=2.0U W=8.0U AD=32P AS=136P PD=24U PS=66U
MN8    5  8  0  0 NMOS L=2.0U W=8.0U AD=32P AS=72P PD=24U PS=34U
.ENDS NAND4
*
* CELULA AND2
*
* Transistores tipo NMOS: 3
* Transistores tipo PMOS: 3
*
.SUBCKT AND2 0 1 3 4 5 6
MP1    4  6  1  1 PMOS L=2.0U W=7.0U AD=195P AS=222P PD=86U PS=98U
MP2    3  4  1  1 PMOS L=2.0U W=7.0U AD=118P AS=122P PD=62U PS=64U
MP3    4  5  1  1 PMOS L=2.0U W=7.0U AD=195P AS=222P PD=86U PS=98U
MN4    3  4  0  0 NMOS L=3.0U W=4.0U AD=64P AS=108P PD=38U PS=60U
MN5    2  6  4  0 NMOS L=3.0U W=4.0U AD=20P AS=84P PD=18U PS=48U
MN6    2  5  0  0 NMOS L=3.0U W=4.0U AD=20P AS=56P PD=18U PS=34U
.ENDS AND2
*
* CELULA XOR2
*
* Transistores tipo NMOS: 3
* Transistores tipo PMOS: 3
*
.SUBCKT XOR2 0 1 2 4 5
MP1    2  5  4  1 PMOS L=2.0U W=7.0U AD=133P AS=70P PD=52U PS=34U
MP2    3  5  1  1 PMOS L=2.0U W=7.0U AD=133P AS=133P PD=52U PS=52U
MP3    2  4  5  1 PMOS L=2.0U W=7.0U AD=105P AS=105P PD=50U PS=50U
MN4    2  3  4  0 NMOS L=3.0U W=4.0U AD=84P AS=52P PD=48U PS=32U
MN5    2  4  3  0 NMOS L=3.0U W=4.0U AD=72P AS=64P PD=42U PS=38U
MN6    3  5  0  0 NMOS L=3.0U W=4.0U AD=76P AS=96P PD=44U PS=54U
.ENDS XOR2
```

```
*
* CELULA NAND3
*
* Transistores tipo NMOS: 3
* Transistores tipo PMOS: 3
*
.SUBCKT NAND3 0 1 3 5 6 7
MP1    3  7  1  1 PMOS L=2.0U W=7.0U AD=203P AS=63P PD=72U PS=32U
MP2    3  6  1  1 PMOS L=2.0U W=7.0U AD=205P AS=254P PD=86U PS=100U
MP3    3  5  1  1 PMOS L=2.0U W=7.0U AD=205P AS=254P PD=86U PS=100U
MN4    2  7  3  0 NMOS L=3.0U W=4.0U AD=152P AS=80P PD=84U PS=46U
MN5    2  5  4  0 NMOS L=3.0U W=4.0U AD=152P AS=120P PD=84U PS=68U
MN6    4  6  0  0 NMOS L=3.0U W=4.0U AD=120P AS=48P PD=68U PS=30U
.ENDS NAND3
*
* CELULA INV2
*
* Transistores tipo NMOS: 1
* Transistores tipo PMOS: 1
*
.SUBCKT INV 0 1 2 3
MP1    2  3  1  1 PMOS L=2.0U W=7.0U AD=126P AS=108P PD=62U PS=48U
MN2    0  3  2  0 NMOS L=4.0U W=4.0U AD=104P AS=70P PD=58U PS=40U
.ENDS INV
*
* CELULA NAND2
*
* Transistores tipo NMOS: 2
* Transistores tipo PMOS: 2
*
.SUBCKT NAND2 0 1 2 4 5
MP1    2  5  1  1 PMOS L=2.0U W=7.0U AD=180P AS=233P PD=84U PS=102U
MP2    2  4  1  1 PMOS L=2.0U W=7.0U AD=180P AS=233P PD=84U PS=102U
MN3    3  4  2  0 NMOS L=3.0U W=4.0U AD=12P AS=100P PD=14U PS=56U
MN4    3  5  0  0 NMOS L=3.0U W=4.0U AD=12P AS=48P PD=14U PS=30U
.ENDS NAND2
*
* CELULA FFJK
*
* Transistores tipo NMOS: 10
* Transistores tipo PMOS: 10
*
.SUBCKT FFJK 0 1 9 10 11 12 14
MP1   10 14  1  1 PMOS L=2.0U W=19.0U AD=250P AS=170P PD=78U PS=76U
MN2    5 14  0  0 NMOS L=3.0U W=11.0U AD=99P AS=132P PD=40U PS=46U
MN3    5 13 10  0 NMOS L=3.0U W=11.0U AD=99P AS=113P PD=40U PS=48U
MP4   10 13  1  1 PMOS L=2.0U W=19.0U AD=250P AS=170P PD=78U PS=76U
MP5   13 12  1  1 PMOS L=2.0U W=7.0U AD=231P AS=56P PD=80U PS=30U
MP6   13 14  1  1 PMOS L=2.0U W=7.0U AD=231P AS=231P PD=80U PS=80U
MN7    4 11 13  0 NMOS L=3.0U W=4.0U AD=20P AS=52P PD=18U PS=32U
MN8    3 14  4  0 NMOS L=3.0U W=4.0U AD=20P AS=20P PD=18U PS=18U
MN9    3 12  0  0 NMOS L=3.0U W=4.0U AD=20P AS=56P PD=18U PS=34U
MP10  13 11  1  1 PMOS L=2.0U W=7.0U AD=70P AS=231P PD=34U PS=80U
MN11   2 11  8  0 NMOS L=3.0U W=4.0U AD=20P AS=56P PD=18U PS=34U
MN12   7 10  2  0 NMOS L=3.0U W=4.0U AD=20P AS=20P PD=18U PS=18U
MN13   7  9  0  0 NMOS L=3.0U W=4.0U AD=20P AS=52P PD=18U PS=32U
MP14   8 11  1  1 PMOS L=2.0U W=7.0U AD=252P AS=70P PD=86U PS=34U
```

```
MP15  8 10  1  1 PMOS L=2.0U W=7.0U AD=252P AS=238P PD=86U PS=82U
MP16  8  9  1  1 PMOS L=2.0U W=7.0U AD=70P AS=238P PD=34U PS=82U
MP17 14  8  1  1 PMOS L=2.0U W=19.0U AD=250P AS=170P PD=78U PS=76U
MP18 14 10  1  1 PMOS L=2.0U W=19.0U AD=250P AS=170P PD=78U PS=76U
MN19  6  8 14  0 NMOS L=3.0U W=11.0U AD=55P AS=113P PD=32U PS=48U
MN20  6 10  0  0 NMOS L=3.0U W=11.0U AD=55P AS=176P PD=32U PS=54U
.ENDS FFJK
*
* CELULA BUF1
*
* Transistores tipo NMOS: 1
* Transistores tipo PMOS: 1
*
.SUBCKT BUF1 0 1 2 3
MP1   2  3  1  1 PMOS L=2.0U W=19.0U AD=166P AS=170P PD=74U PS=76U
MN2   2  3  0  0 NMOS L=3.0U W=11.0U AD=134P AS=138P PD=66U PS=68U
.ENDS BUF1
*
* CELULA BUF3
*
* Transistores tipo NMOS: 3
* Transistores tipo PMOS: 3
*
.SUBCKT BUF3 0 1 2 5
MP1   4  5  1  1 PMOS L=2.0U W=7.0U AD=137P AS=109P PD=58U PS=48U
MP2   3  4  1  1 PMOS L=2.0U W=21.0U AD=144P AS=204P PD=70U PS=70U
MP3   2  3  1  1 PMOS L=2.0U W=63.0U AD=214P AS=300P PD=94U PS=140U
MN4   3  4  0  0 NMOS L=3.0U W=12.0U AD=210P AS=168P PD=70U PS=52U
MN5   4  5  0  0 NMOS L=3.0U W=4.0U AD=115P AS=64P PD=60U PS=38U
MN6   2  3  0  0 NMOS L=3.0U W=36.0U AD=240P AS=232P PD=92U PS=86U
.ENDS BUF3
*
* CELULA FFDC_MS
*
* Transistores tipo NMOS: 15
* Transistores tipo PMOS: 15
*
.SUBCKT FFDC_MS 0 1 12 14 15 18 19 20
MP1   5 16  1  1 PMOS L=2.0U W=7.0U AD=169P AS=63P PD=82U PS=32U
MP2  18 20  5  1 PMOS L=2.0U W=7.0U AD=63P AS=169P PD=32U PS=82U
MN3   9 16 18  0 NMOS L=3.0U W=4.0U AD=12P AS=114P PD=14U PS=56U
MN4   9 19  0  0 NMOS L=3.0U W=4.0U AD=12P AS=60P PD=14U PS=36U
MN5  18 20  0  0 NMOS L=3.0U W=4.0U AD=114P AS=144P PD=56U PS=78U
MP6   5 19  1  1 PMOS L=2.0U W=7.0U AD=169P AS=112P PD=82U PS=46U
MN7  20 18  0  0 NMOS L=3.0U W=4.0U AD=84P AS=88P PD=48U PS=50U
MP8   4 14 20  1 PMOS L=2.0U W=7.0U AD=168P AS=84P PD=62U PS=38U
MN9  20 14  0  0 NMOS L=3.0U W=4.0U AD=172P AS=148P PD=86U PS=70U
MP10  3 19  1  1 PMOS L=2.0U W=7.0U AD=170P AS=63P PD=72U PS=32U
MP11  4 18  3  1 PMOS L=2.0U W=7.0U AD=168P AS=170P PD=62U PS=72U
MN12  8 17 20  0 NMOS L=3.0U W=4.0U AD=28P AS=172P PD=22U PS=86U
MN13  8 19  0  0 NMOS L=3.0U W=4.0U AD=28P AS=148P PD=22U PS=70U
MP14  3 17  1  1 PMOS L=2.0U W=7.0U AD=170P AS=105P PD=72U PS=44U
MP15  2 12  1  1 PMOS L=2.0U W=7.0U AD=180P AS=56P PD=86U PS=30U
MN16  7 15  0  0 NMOS L=3.0U W=4.0U AD=12P AS=60P PD=14U PS=36U
MN17  7 12 17  0 NMOS L=3.0U W=4.0U AD=12P AS=114P PD=14U PS=56U
MP18 17 16  2  1 PMOS L=2.0U W=7.0U AD=63P AS=180P PD=32U PS=86U
MN19 17 16  0  0 NMOS L=3.0U W=4.0U AD=114P AS=144P PD=56U PS=78U
```

```
MP20  2 15  1  1 PMOS L=2.0U W=7.0U AD=180P AS=105P PD=86U PS=44U
MN21 16 17  0  0 NMOS L=3.0U W=4.0U AD=84P AS=88P PD=48U PS=50U
MP22 10 14 16  1 PMOS L=2.0U W=7.0U AD=168P AS=84P PD=62U PS=38U
MN23 16 14  0  0 NMOS L=3.0U W=4.0U AD=168P AS=152P PD=84U PS=72U
MP24 11 15  1  1 PMOS L=2.0U W=7.0U AD=170P AS=63P PD=72U PS=32U
MP25 10 17 11  1 PMOS L=2.0U W=7.0U AD=168P AS=170P PD=62U PS=72U
MN26  6 13 16  0 NMOS L=3.0U W=4.0U AD=28P AS=168P PD=22U PS=84U
MN27  6 15  0  0 NMOS L=3.0U W=4.0U AD=28P AS=152P PD=22U PS=72U
MP28 11 13  1  1 PMOS L=2.0U W=7.0U AD=170P AS=105P PD=72U PS=44U
MN29 13 12  0  0 NMOS L=3.0U W=4.0U AD=72P AS=100P PD=42U PS=56U
MP30 13 12  1  1 PMOS L=2.0U W=7.0U AD=203P AS=63P PD=72U PS=32U
.ENDS FFDC_MS
*
* CELULA FFDS_MS
* Transistores tipo NMOS: 15
* Transistores tipo PMOS: 15
.SUBCKT FFDS_MS 0 1 12 14 15 16 18 20 16
MN1  12 16  0  0 NMOS L=3.0U W=4.0U AD=124P AS=48P PD=68U PS=30U
MP2   5 16 12  1 PMOS L=2.0U W=7.0U AD=119P AS=91P PD=48U PS=40U
MP3   4 19  1  1 PMOS L=2.0U W=7.0U AD=245P AS=63P PD=100U PS=32U
MP4   5 20  4  1 PMOS L=2.0U W=7.0U AD=119P AS=245P PD=48U PS=100U
MN5  12 20  0  0 NMOS L=3.0U W=4.0U AD=122P AS=76P PD=60U PS=44U
MN6   9 19 12  0 NMOS L=3.0U W=4.0U AD=84P AS=122P PD=50U PS=60U
MP7   4 18  1  1 PMOS L=2.0U W=7.0U AD=245P AS=63P PD=100U PS=32U
MN8   9 18  0  0 NMOS L=3.0U W=4.0U AD=84P AS=48P PD=50U PS=30U
MN9  20 12  0  0 NMOS L=3.0U W=4.0U AD=160P AS=160P PD=80U PS=76U
MP10  3 18  1  1 PMOS L=2.0U W=7.0U AD=191P AS=63P PD=78U PS=32U
MP11 20 12  3  1 PMOS L=2.0U W=7.0U AD=63P AS=191P PD=32U PS=78U
MN12  8 17 20  0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=80U
MN13  8 18  0  0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=76U
MP14  3 17  1  1 PMOS L=2.0U W=7.0U AD=191P AS=112P PD=78U PS=46U
MN15 17 16  0  0 NMOS L=3.0U W=4.0U AD=108P AS=64P PD=60U PS=38U
MP16  2 16 17  1 PMOS L=2.0U W=7.0U AD=119P AS=56P PD=48U PS=30U
MP17 10 15  1  1 PMOS L=2.0U W=7.0U AD=242P AS=56P PD=102U PS=30U
MP18  2 19 10  1 PMOS L=2.0U W=7.0U AD=119P AS=242P PD=48U PS=102U
MN19 17 19  0  0 NMOS L=3.0U W=4.0U AD=118P AS=80P PD=58U PS=46U
MN20  7 15 17  0 NMOS L=3.0U W=4.0U AD=84P AS=118P PD=50U PS=58U
MP21 10 14  1  1 PMOS L=2.0U W=7.0U AD=242P AS=63P PD=102U PS=32U
MN22  7 14  0  0 NMOS L=3.0U W=4.0U AD=84P AS=48P PD=50U PS=30U
MN23 19 17  0  0 NMOS L=3.0U W=4.0U AD=160P AS=160P PD=80U PS=76U
MP24 11 14  1  1 PMOS L=2.0U W=7.0U AD=191P AS=63P PD=78U PS=32U
MP25 19 17 11  1 PMOS L=2.0U W=7.0U AD=63P AS=191P PD=32U PS=78U
MN26  6 14  0  0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=76U
MN27  6 13 19  0 NMOS L=3.0U W=4.0U AD=28P AS=160P PD=22U PS=80U
MP28 11 13  1  1 PMOS L=2.0U W=7.0U AD=191P AS=112P PD=78U PS=46U
MN29 13 15  0  0 NMOS L=3.0U W=4.0U AD=72P AS=100P PD=42U PS=56U
MP30 13 15  1  1 PMOS L=2.0U W=7.0U AD=203P AS=63P PD=72U PS=32U
.ENDS FFDS_MS
```

```
*
* CIRCUITO PRINCIPAL
*
* Transistores tipo NMOS: 262
* Transistores tipo PMOS: 262
*
* Nomes dos nos
* No 18:  TMCOD
* No 19:  TPDEC
* No 21:  TMDEC
* No 22:  CKCOD
* No 31:  TD
* No 38:  TPCOD
* No 42:  HTX
* No 46:  NDADO
* No 50:  NAMD
* No 59:  DADO
* No 61:  CKDEC
X1   0 1 14 32 23 NAND2
X2   0 1 23 13 60 56 24 FFD_MS
X3   0 1 32 20 60 56 34 FFD_MS
X4   0 1 34 11 60 24 46 FFJK
X5   0 1 50 60 11 24 NAND3
X6   0 1 53 60 46 34 NAND3
X7   0 1 54 50 53 NAND2
X8   0 1 56 60 BUF3
X9   0 1 14 54 60 28 56 27 FFDC_MS
X10  0 1 33 27 60 56 35 FFD_MS
X11  0 1 43 35 60 56 44 FFD_MS
X12  0 1 44 54 60 58 56 45 FFDC_MS
X13  0 1 5 21 30 55 18 22 55 FFDS_MS
X14  0 1 22 41 21 39 18 38 FFDC_MS
X15  0 1 48 38 21 18 9 FFD_MS
X16  0 1 10 21 9 51 18 59 51 FFDS_MS
X17  0 1 51 2 3 55 AND2
X18  0 1 55 49 57 NOR2
X19  0 1 57 39 5 10 48 NAND4
X20  0 1 7 21 0 55 18 40 55 FFDS_MS
X21  0 1 29 40 21 18 26 FFD_MS
X22  0 1 19 26 21 18 4 FFD_MS
X23  0 1 16 4 6 XOR2
X24  0 1 18 21 BUF1
X25  0 1 17 21 BUF1
X26  0 1 25 16 21 17 6 FFD_MS
X27  0 1 15 6 52 XOR2
X28  0 1 31 15 INV
X29  0 1 12 21 31 8 NAND3
X30  0 1 36 12 INV
X31  0 1 37 21 42 15 NAND3
X32  0 1 42 8 INV
X33  0 1 8 47 6 XOR2
X34  0 1 3 52 21 17 47 FFD_MS
X35  0 1 52 59 47 XOR2
.END
```

BIBLIOGRAFIA

[BRO 86] BROWN, Randy Lee. Multiple Storage quad trees: A simpler faster alternative to bisector list quad trees. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.5, n.3, p. 413-418; July 1986.

[CAR 87] CARLSON, E.; RUTENBAR, R. A. A Scanline Data Structure Processor for VLSI Geometry Checking. **IEEE Transactions on Computer-Aided Design of Integrated Circits and Systems**. New York, V.6, n.5, p.780-797, Sept. 1987.

[CHA 89] CHANDRAMOULI, V. A PC-based CAD Tool For Circuit Extraction. **IEEE Circuits And Devices Magazine**. New York, v.5, n.5, p.44-47; Sept. 1989.

[GOM 88] GOMES, R. F. DARC: Um Verificador de Regras de Projeto de CI's Utilizando Programação em Lógica. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 3, Gramado, 13-15 abr., 1988. **Anais**. Porto Alegre, S. B. C., 1988. p. 85-94.

[HAR 86] HARBOUR, M. G. & DRAKE, J. M. Calculation of Multiterminal Resistances in Integrated Circuits. In: **IEEE transactions on Circuits and Systems**. v. 33, n.9, p. 462-463. Apr. 1986.

[HEN 87] HENKEL, Volker. **Zur Hierarchichen Parameter-Extraction Beim Entwurf Integrierter Schaltkreise.** Braunschweig, Technische Universität, 1987. Tese de doutorado.

[JAC 86a] JACOBI, R. P. **Microeditor: Especificação, Estrutura de dados e Principais Algoritmos.** Porto Alegre, PGCC da UFRGS, 1988. RP 90.

[JAC 86b] JACOBI, R. P. MicroEditor: editor gráfico para microeletrônica em PC-IBM. In CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, Campinas, 13-17 jul, 1986. **Anais**. Campinas, CPqD/Telebrás, 1986. p. 408-418.

[JAC 86c] JACOBI, R. P. **Sistema RS** Porto Alegre, PGCC-UFRGS, 1986. Edição interna.

[KEM 88] KEMP, A. J.; PRETORIUS, J. A.; SMIT, W. The Generation of a Mesh for Resistance Calculation in Integrated Circuits. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**. New York, v.7, n.10, p. 1029-1037, Oct 1988.

[LUB 89] LUBASZEWSKY, M.; REIS, R. A. L.; BAGGIO, A. A Random Logic Generator Using TRANCA Methodology. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 6., Porto Alegre, 12-14 jul. 1989. **Anais**. Porto Alegre, SBC, 1989. p. 67-78.

[McC 84] McCORMICK, s. p. EXCL: A Circuit Extractor for IC Designs In: DESIGN AUTOMATION CONFERENCE, 21., Albuquerque, June 25-27, 1984. **Proceedings**. New York, IEEE, 1984. p. 616-623.

[MEA 80] MEAD, C.; CONWAY, L. **Introduction to VLSI systems** Reading, Addison-Wesley, 1980

[MIT 87] MITSUHASHI, T.; YOSHIDA, K. A Resistance Calculation Algorithm and Its Aplication to Circuit Extraction. **IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems**, New York, v.6, n.3, p. 337-345, May 1987.

[MOR 88] MORAES, F. G.; REIS, A. I.; LUBASZEWSKI, M.; REIS, R.A.L. **Biblioteca de Standard Cells do Projeto Tranca**. Porto Alegre, PGCC da UFRGS, 1988. RP 92.

[MOR 89b] MORSCHEL, Ivan; BARONE, Dante A. C. Experiência de integração de ferramentas: uso do GERME. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 6., Porto Alegre, 12-14 jul. 1989. **Anais**. Porto Alegre, SBC, 1989. p. 685-695.

[OUS 84] OUSTERHOUT, J. K. Corner Stitching: A data structuring technique for VLSI layout tools. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.CAD 3, n.1, p. 100-111, Jan 1984.

[OUS 85] OUSTERHOUT, J. K.; HAMACHI, G. I.; MAYO, R. N.; SCOTT, W. S.; TAYLOR, G. S. The MAGIC VLSI layout system. **IEEE Design and Test of Computers**, Los Alamitos, v. 2, n. 2, p.19-30, 1985.

[PIT 89] PITAKSANOKUL, A.; THANAWASTIEN, S.; LURSINAP, C. Comparisons of Quad Trees and 4D Trees: New Results. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 8, n. 11, p. 1157-1163, Nov 1989.

[REI 88] REIS, R. A. L.; GOMES, R. F.; LUBASZEWSKI, M. An Efficient Design Methodology for Standard Cell Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS. Helsinki, June 7-9, 1988. **Proceedings**. Pitscaway, IEEE, 1988. p. 1213-1216.

[REI 89] REIS, A. I.; MORAES, F. G.; REIS, R. A. L. MODEM: Desenvolvimento de um ASIC para Modems de Banda Base. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 6., Porto Alegre, 12-14 jul. 1989. **Anais**. Porto Alegre, SBC, 1989. p. 617-625.

[STE 89a] STEMMER, M. A.; REIS, R.A.L. EXTRIBO: Um Extrator de Circuitos In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 4. Rio de Janeiro, 12-14 abr. 1989. **Anais**. Rio de Janeiro, SBC, 1989. p. 1-9.

[STE 89b] STEMMER, M. A.; REIS, R. A. L. Um Algoritmo Para Calcular Resisténcias em Circuitos Integrados. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICRO-ELETRÔNICA, 6. Porto Alegre, 12-14 jul. 1989. **Anais**. Porto Alegre, SBC, 1989. p. 575-586.

[SAK 83] SAKURAI, Takayasu. Approximation of wiring delay in MOSFET LSI. **IEEE Journal of Solid State Circuits**, New York, v. SC-18, n. 4, p. 418-426, Aug. 1983.

[TRU 87] TRULLEMANS, C. **ARAMOS: Another Relaxation Analyser of MOS Circuits** Louvain-la-Neuve, UCL, Laboratoire de Microeletronique, 1987. Relatório interno.

[VLA 81] VLADIMIRESCU, A.; ZHANG, K.; NEWTON, A. C.; PEDERSON, D. O.; SANGIOVANNI-VICENTELLI, A. **SPICE Version 2G User's Guide** Berckley, University of California, 1981.

SERVIÇO PÚBLICO FEDERAL

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

EXTRIBO: um Extrator Hierárquico de Circuitos

Dissertação apresentada aos Srs.

Prof. Dr. Sergio Bampi

Prof. Tiaraju Vasconcellos Wagner

Prof. Dr. Ricardo A. da L. Reis

Visto e permitida a impressão

Porto Alegre, 18 / 07 / 90

Prof. Dr. Ricardo A. da L. Reis
Orientador

Prof. Dr. Ricardo Augusto da L. Reis
Coordenador do Curso de Pós-Graduação
em Ciência da Computação