



**FACULDADE DE INFORMÁTICA**  
**PUCRS – Brasil**

<http://www.inf.pucrs.br>

# **ANATOMIA DE UMA FERRAMENTA DE SÍNTESE AUTOMÁTICA DE LAYOUT**

*Fernando Gehm Moraes*

**TECHNICAL REPORT SERIES**

---

August, 2000

Contact:

[moraes@inf.pucrs.br](mailto:moraes@inf.pucrs.br)

<http://www.inf.pucrs.br/~moraes>

F. G. Moraes is a senior lecturer (associate professor) at the PUCRS/Brazil since 1996. His main research topics are digital systems design and fast prototyping, digital systems physical synthesis CAD, telecommunication applications, hardware-software codesign. He is a member of the Hardware Design Support Group (GAPH) at the PUCRS.

F. G. Moraes é Professor Adjunto da Faculdade de Informática da PUCRS desde 1996, Doutor em Microeletrônica pela Universidade de Montpellier II (1994), Mestre em Ciência da Computação pela UFRGS (1990) , Engenheiro Eletricista pela UFRGS (1987). Principais tópicos de pesquisa: síntese física de circuitos integrados, ferramentas para o apoio ao projeto de circuitos integrados (CAD), prototipação rápida de sistemas digitais, projeto conjunto hardware/software.

Copyright © Faculdade de Informática – PUCRS

Published by the Campus Global – FACIN – PUCRS

Av. Ipiranga, 6681

90619-900 Porto Alegre – RS – Brazil

# SUMÁRIO

<b>1</b>	<b><i>Introdução</i></b>	<b>1</b>
<b>2</b>	<b><i>Síntese Automática de layout</i></b>	<b>2</b>
2.1	Biblioteca Virtual de Células	3
2.2	Síntese de Bibliotecas e Síntese de Macro-Células	4
2.3	A Ferramenta de Síntese Automática de Layout TROPIC3	5
2.4	Referências bibliográficas	8
<b>3</b>	<b><i>Estilo de Layout da Ferramenta TROPIC3</i></b>	<b>9</b>
3.1	Layout a nível de célula básica	9
3.2	Layout a nível de banda	12
3.3	Layout a nível de macro-célula	14
3.4	Codificação da tecnologia	15
3.5	Referências bibliográficas	17
<b>4</b>	<b><i>Síntese Física: do Netlist ao Layout</i></b>	<b>18</b>
4.1	Extração de células folha	19
4.2	Posicionamento das células folha	20
4.3	Geração de células folha	23
4.4	Assinalamento de pinos	25
4.5	Geração de banda	27
4.6	Fixação dos pontos de passagem entre bandas não adjacentes	28
4.7	Roteamento	29
4.8	Geração do arquivo de layout em formato CIF	31
4.9	Referências bibliográficas	31
<b>5</b>	<b><i>Extração de Elementos Parasitas</i></b>	<b>33</b>
5.1	Algoritmo de Extração de Conectividade	34
5.2	Modelo de Extração	36
5.3	Referências bibliográficas	39
<b>6</b>	<b><i>Integração da Síntese Física à Síntese Lógica</i></b>	<b>40</b>
6.1	Abordagem de biblioteca virtual no ambiente Virtual Synopsys	41
6.2	Referências bibliográficas	46
<b>7</b>	<b><i>Exercícios</i></b>	<b>47</b>



# ANATOMIA DE UMA FERRAMENTA DE SÍNTESE AUTOMÁTICA DE LAYOUT

## 1 INTRODUÇÃO

Este Capítulo apresenta os procedimentos que são necessários para implementar uma ferramenta de síntese automática de layout, assim como integrá-la a um fluxo de projeto que contemple níveis abstratos de descrição, como por exemplo, VHDL.

O Capítulo 2 apresenta os conceitos de biblioteca virtual de células e síntese de macro-células. Estes conceitos são importantes, pois permitem situar a abordagem de síntese automática frente a outras abordagens, como *standard-cells* ou *full-custom*. Ainda neste Capítulo, apresenta-se as motivações que conduziram ao desenvolvimento da ferramenta TROPIC3, que será a ferramenta utilizada como exemplo no decorrer do texto.

O Capítulo 3 apresenta o estilo de layout adotado na ferramenta TROPIC3. Cada detalhe de implementação, como posicionamento dos transistores, posição da alimentação, posição dos nós de saída, entre outros, são apresentados e justificados. O Capítulo 4 apresenta de forma simplificada os algoritmos necessários para implementar-se uma ferramenta de síntese de layout.

Considerando a importância das capacitâncias parasitas em tecnologias sub-micrônicas, o Capítulo 5 apresenta o procedimento adotado para extrair-se estas capacitâncias, de uma forma rápida e precisa.

O último Capítulo, 6, mostra o procedimento para integrar a síntese automática de layout a um fluxo de projeto com entrada VHDL, permitindo o uso de bibliotecas virtuais.

## 2 SÍNTESE AUTOMÁTICA DE LAYOUT

A redução acentuada do tempo de vida de um produto no mercado, a crescente complexidade dos circuitos integrados (CIs) e a pressão para que um produto esteja disponível o mais rapidamente possível ao consumidor são alguns dos problemas enfrentados pelos projetistas. Dois fatores tem contribuído significativamente para o aumento da complexidade dos CIs: utilização de *cores* (módulos pré-definidos de hardware) e tecnologias submicrônicas. Logo, um projetista de CIs deve concentrar-se em níveis de projeto mais abstratos, utilizando por exemplo linguagens de descrição de hardware como VHDL, deixando a síntese física para ferramentas que gerem o layout automaticamente.

Existem diversas soluções automatizadas para a síntese física, como: biblioteca de células pré-definidas (*standard-cells* e *gate-arrays*) e geradores de módulos (multiplicadores, memórias). A primeira classe, baseada em biblioteca de células, é utilizada para síntese de lógica irregular, ou randômica (não apresentam um padrão definido, como por exemplo máquinas de estado ou “lógica de cola”). A segunda classe, geradores de módulos, geram estruturas regulares a partir de blocos construtivos básicos, resultando em módulos compactos e com bom desempenho elétrico.

A nossa abordagem visa a síntese de blocos com lógica irregular. A diferença em relação aos métodos correntemente utilizados é a não utilização de biblioteca de células. Bibliotecas de células representam uma boa solução para tecnologias micrônicas, onde o desempenho elétrico é basicamente função do atraso das células. Logo, fica fácil caracterizar um bloco contendo células pré-definidas, pois basta somar os atrasos das células no caminho crítico do circuito. Em tecnologias submicrônicas esta característica não é mais verdadeira, pois o atraso depende em grande parte do comprimento das interconexões, e para tecnologias abaixo de 0,35 micrômetros o atraso das interconexões será superior ao atraso das próprias células que compõem o circuito.

As principais desvantagens dos métodos baseados em células pré-definidas são: (i) número limitado de funções lógicas, normalmente poucas funções complexas estão disponíveis nas bibliotecas; (ii) transistores de tamanho fixo, o que leva à necessidade de se implementar diversos padrões de células (*templates*) para cada função; (iii) dependências às regras de projeto, a cada mudança na tecnologia deve-se refazer toda a biblioteca; (iv) complexidade para as ferramentas de síntese gerenciarem bibliotecas complexas, contendo em torno de uma centena de funções diferentes, cada uma com diversos *templates*.

A abordagem que parte de uma lista de portas lógicas interconectadas (*netlist*) e gere o layout automaticamente a partir deste *netlist* e das regras de projeto resolve os 4 problemas acima: (i) o número de funções é ilimitado, em função apenas do *netlist*; (ii) cada célula pode ter o tamanho de seus transistores dimensionado em função da carga e das restrições de atraso; (iii) o layout é independente das regras de projeto, pois para uma nova tecnologia basta alterar o arquivo contendo as regras; (iv) as ferramentas de síntese não necessitam mais gerenciar complexas bibliotecas de células.

Esta abordagem, síntese automática de layout, tem entretanto algumas deficiências que limitam a sua utilização:

- As células não são pré-caracterizadas. Como visto acima, a pré-caracterização permite uma estimativa rápida de qual será o atraso do circuito. Quando parte-se de um *netlist* deve-se primeiro gerar o circuito, o que pode consumir um tempo grande de CPU, para depois estimar-

se o atraso. Como visto acima, pre-caracterização não é mais uma garantia de estimação precisa de atraso, pois nas atuais tecnologias deve-se considerar o atraso das interconexões. Logo, tanto para síntese automática quanto para biblioteca de células, deve-se estimar primeiro as capacitâncias de roteamento (parasitas) para depois estimar o atraso.

- A topologia das células é fixa. A síntese automática de layout segue padrões regulares de células, gerando apenas células complementares e portas de transmissão. Lógica não dual, como em circuitos dinâmicos, não são normalmente sintetizados. A ferramenta LAS (Cadence) sintetiza estas células irregulares, porém com uma densidade de integração muito baixa.
- O número de ferramentas no fluxo de projeto é maior que nas abordagens baseadas em células. Por exemplo, ferramentas como LAS ou TROPIC2, utilizam compactação de layout, a qual consome um tempo muito grande de CPU.

Apesar das vantagens das ferramentas de síntese automática, as 3 desvantagens acima fazem com que este tipo de ferramenta não seja muito utilizada pelos projetistas de CIs.

Além disto, devido ao uso de estilo regular de implementação do layout, a densidade de integração (em transistores por milímetro quadrado) obtida com ferramentas de síntese automática de layout tende a ser menor que as abordagens baseadas em bibliotecas de células (*standard cells*). Esta densidade de integração melhor para a bordagem *standard cells* deve-se ao fato que as células são projetadas manualmente (projeto *full-custom*), com uma ocupação mínima de área. Exemplo de células com layout extremamente otimizado nas bibliotecas de células: flip-flops e somadores. A forma encontrada pelas ferramentas de síntese de layout para compensar esta perda em densidade é a utilização de células complexas, o que reduz o número total de transistores.

Nosso **objetivo** é mostrar a implementação de uma ferramenta de síntese automática que resolva ao menos dois dos problemas acima, tornando o uso de síntese automatizada uma real alternativa aos métodos baseados em células. As deficiências que serão tratadas são: predição rápida de parasitas de roteamento e ausência de compactação. A predição de parasitas permite uma rápida estimativa do atraso do circuito. A ausência de compactação de layout permite uma síntese de layout extremamente rápida.

## 2.1 Biblioteca Virtual de Células

A principal diferença no fluxo de projeto entre as abordagens baseadas em bibliotecas e a síntese automática de células é o mapeamento tecnológico, que consiste na implementação de uma rede Booleana em termos de dispositivos providos pela tecnologia alvo.

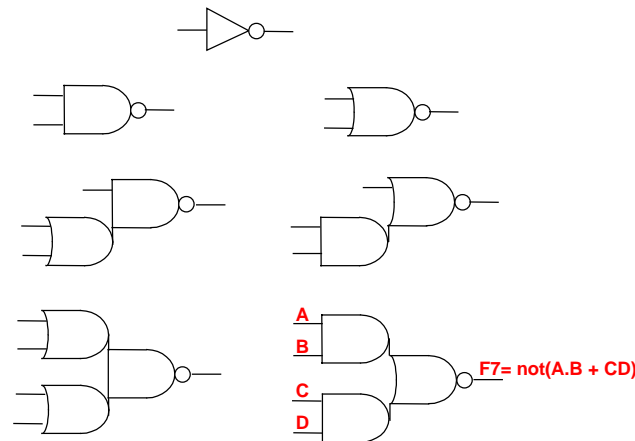
Em abordagens baseadas em biblioteca de células a ferramenta de mapeamento tecnológico é restrita a um conjunto limitado de funções pré-existentes.

Em ferramentas de síntese automática de layout o número de funções lógicas é normalmente função do número de transistores em série. Como mostrado na Tabela 1, se limitarmos o número de transistores em série em 4 (tanto no plano N como no plano P), teremos 3503 funções lógicas diferentes, o que é muito superior à qualquer biblioteca de células.

A Figura 1 ilustra as 7 diferentes funções lógicas que podem ser obtidas com a limitação de 2 transistores em série, tanto no plano N quanto no plano P. A função F7, que implementa uma soma de produtos negada, necessita 8 transistores para ser implementada. Caso fossem utilizadas células simples, como *nands* ou *nors*, seriam necessários ao menos 14 transistores. Se considerarmos células mais complexas, com 4 transistores em série, esta diferença é muito mais acentuada.

		Número de Transistores PMOS em Série				
		1	2	3	4	5
Número de Transistores NMOS em série	1	1	2	3	4	5
	2	2	7	18	42	90
	3	3	18	87	396	1677
	4	4	42	396	3503	28435
	5	5	90	1677	28435	425803

**Tabela 1 - Número de células versus número de transistores em série.**



**Figura 1 - Biblioteca (2,2), contendo 7 funções lógicas diferentes.**

Quando a ferramenta de mapeamento tecnológico trabalha sobre restrições topológicas (por exemplo, máximo número de transistores em série ou máximo fanout) ao invés de uma biblioteca pré-existente, dizemos que estamos trabalhando sobre uma **biblioteca virtual**.

A restrição topológica de número de transistores em série é função da tecnologia utilizada e do desempenho elétrico. Recomenda-se para tecnologias submicrônicas a utilização de bibliotecas (4,3), ou seja, até 4 transistores em série no plano N e até 3 transistores em série no plano P.

A vantagem de utilizarmos bibliotecas virtuais é o aumento do espaço de soluções para o mapeador tecnológico, o que conduz a um número sensivelmente menor de transistores (em média 35% menos) e consequentemente uma menor área de silício, melhor desempenho elétrico e menor consumo de potência.

## 2.2 Síntese de Bibliotecas e Síntese de Macro-Células

Duas alternativas para a síntese automática do layout utilizando bibliotecas virtuais podem ser consideradas:

- **Síntese de Bibliotecas.** Este método isola todas as células do *netlist* de entrada, gerando um layout único para cada célula. Todas as células devem apresentar a mesma altura e posição pré-definida para as entradas e saídas. Os passos seguintes da síntese física são praticamente os mesmos que na abordagem baseada em células, ou seja, posicionamento e roteamento. Este método resolve o problema de dependência à tecnologia e número limitado de funções, porém não permite um dimensionamento individual dos transistores.
- **Síntese de Macro-Células.** Neste método todas as células que compõem o circuito são sintetizadas, independentemente de terem ou não a mesma função. Desta forma, cada porta



lógica do circuito, mesmo tendo a mesma função lógica, poderá ter layouts diferentes. Esta abordagem exige ferramentas de posicionamento e roteamento dedicadas, porém permite o dimensionamento individual de cada transistor do circuito, de acordo com as restrições impostas pelo projetista.

## 2.3 A Ferramenta de Síntese Automática de Layout TROPIC3

TROPIC3 é uma ferramenta de síntese automática de layout de circuitos integrados. Utiliza bibliotecas virtuais, permitindo assim a síntese de qualquer célula complexa. O método de síntese empregado é o de síntese de macro-células, ou seja, todas as células do circuito são sintetizadas.

Apenas dois arquivos são necessários para a síntese do layout: um *netlist* em formato SPICE e um arquivo contendo as regras de projeto. Como saída da ferramenta TROPIC3 temos o layout do circuito, em formato CIF, e o arquivo contendo as capacitâncias parasitas de roteamento (inclusive, como será visto adiante, as de acoplamento entre condutores vizinhos).

As motivações para o desenvolvimento da ferramenta TROPIC3 foram:

- Reduzir o tempo de CPU necessário para a síntese de layout. A etapa de síntese responsável por consumir um grande tempo de CPU, como mencionado anteriormente, é a compactação de layout. Tanto as ferramentas LAS quanto TROPIC2 geram uma descrição simbólica do circuito, sendo posteriormente convertida por um programa de compactação baseado em um grafo de restrições. Esta etapa de compactação consome mais de 1 hora para um circuito com 5000 transistores, em uma Ultra-Sparc 10.
- Utilizar 3 níveis de metal e contatos empilhados para reduzir a área final do circuito.

A compactação de layout baseada em grafo de restrições foi substituída por uma compactação simplificada, baseada em grade virtual, adaptada ao estilo de layout implementado no TROPIC3. Esta compactação simplificada gera layouts menos densos, perda esta compensada largamente pela utilização dos 3 níveis de metal e contatos empilhados. A Tabela 2 compara as densidades obtidas com TROPIC2/LAS e TROPIC3. Observa-se que a densidade obtida foi sempre melhor com TROPIC3, justificando assim a nova abordagem.

Circuito	Tr#	Bandas	LAS dml	TROPIC2	TROPIC3
Adder	28	2	4780	5942	<b>10136</b>
Addergate	40	2	5598	6020	<b>9542</b>
Alu	260	4	5812	5294	<b>7606</b>
Alugate	432	4	6050	5405	<b>7494</b>
Rip	448	5	5961	5610	<b>8726</b>
Cla	528	5	5614	5498	<b>8133</b>
Hdb3	570	4	4903	6516	<b>7892</b>
Mult6	972	7	5950	5779	<b>7152</b>
Mult2	4512	16	4879	4080	<b>5924</b>

**Tabela 2 - Comparação de densidade de integração (tr/mm<sup>2</sup>) entre ferramentas de síntese de layout que utilizam compactação baseada em grafo de restrições (TROPIC2 e LAS) e TROPIC3 (grade virtual e 3 níveis de metal). Tecnologia 0.7  $\mu$ m. LAS dml significa LAS com dois níveis de metal.**

A consequência da simplificação da etapa de compactação fez com que o tempo de CPU para a síntese fosse reduzido drasticamente. A Tabela 3 mostra a densidade e o tempo de CPU para síntese e a cálculo de capacitâncias parasitas. Observa-se que o tempo total para síntese e cálculo de

capacitâncias parasitas foi de 337920 ms (5,6 minutos) para um circuito com 14376 transistores. Se fosse utilizada compactação, como no TROPIC2/LAS, certamente seriam necessárias várias horas de CPU na mesma máquina, e uma quantidade imensa de memória (mais que 200 Mbytes).

Circuito	Transistores	Redes	Bandas	X x Y ( $\mu\text{m}$ )	Area ( $\text{mm}^2$ )	Densidade ( $\text{tr}/\text{mm}^2$ )	CPU (ms) Geração	CPU (ms) Ext. Capac.
adder	28	13	1	20.45 x 13.60	0.00028	<b>100676</b>	50	170
addergate	40	15	2	16.80 x 25.20	0.00042	<b>94482</b>	180	310
alu	260	94	3	74.45 x 47.40	0.00353	<b>73677</b>	440	3970
alugate	432	117	4	90.00 x 59.40	0.00535	<b>80808</b>	500	6050
rip	448	163	4	88.70 x 56.40	0.00500	<b>89552</b>	720	8360
cla	528	215	4	111.45 x 62.80	0.00700	<b>75439</b>	660	9010
hdb3	570	191	6	87.00 x 85.60	0.00745	<b>76539</b>	420	9810
5xp1	798	308	7	107.45 x 101.20	0.01087	<b>73386</b>	790	15310
sao2	930	361	7	123.25 x 107.20	0.01321	<b>70388</b>	1030	17770
mult6	972	308	6	135.70 x 96.60	0.01311	<b>74147</b>	1230	15800
9sym	1092	420	8	126.00 x 122.80	0.01547	<b>70575</b>	1170	22790
c499	1556	511	7	194.45 x 117.20	0.02279	<b>68277</b>	1730	28820
c1355	2244	647	9	200.45 x 148.40	0.02975	<b>75437</b>	2390	41460
c1908	3146	990	14	199.00 x 239.40	0.04764	<b>66036</b>	3850	63680
mult2	4512	1239	13	284.70 x 260.80	0.07425	<b>60768</b>	5610	66220
c2670	4976	1762	15	302.45 x 278.00	0.08408	<b>59181</b>	6360	98230
c7552_3x3	6164	2101	15	364.45 x 364.80	0.13295	<b>46363</b>	14730	101100
c3540	7154	2359	15	435.25 x 309.00	0.13449	<b>53193</b>	9100	118810
mult12	8584	2455	16	432.00 x 341.60	0.14757	<b>58169</b>	15110	133850
c6288	10112	2706	18	442.00 x 337.80	0.14931	<b>67726</b>	11030	162970
c5315	10656	3429	15	636.45 x 390.00	0.24822	<b>42930</b>	24360	167510
c7552	14376	4764	17	764.00 x 433.20	0.33096	<b>43437</b>	109700	228220

**Tabela 3 – Densidade e tempo de CPU para a síntese de diversos benchmarks, utilizando-se a ferramenta TROPIC3, tecnologia 0.25  $\mu\text{m}$ , transistores dimensionados com  $w=2 \mu\text{m}$ ,  $l=0.25 \mu\text{m}$ , máquina Ultra-Sparc 10.**

Como consequência da velocidade da síntese automática de layout obtida com a ferramenta TROPIC3, iterações entre a síntese lógica e física podem agora serem feitas, pois o tempo de CPU para o cálculo preciso das capacitâncias parasitas não é mais o gargalo, como nas abordagens anteriores. Desta forma, o *netlist* de capacitâncias parasitas gerado por TROPIC3 pode ser utilizado por:

- Ferramentas de mapeamento tecnológico para selecionar quais portas serão utilizadas e onde *buffers* devem ser inseridos;
- Ferramentas de dimensionamento de transistores para dimensionar corretamente os transistores em função das capacitâncias obtidas;
- Ferramentas de análise de atraso (*timing analysers*) para estimativa do desempenho do circuito.
- Ferramentas de estimativa de consumo de potência.

É muito importante ressaltar o fato que o posicionamento das células deve manter-se o mais próximo possível entre as diversas iterações de síntese de layout, afim de manterem-se coerentes as capacitâncias parasitas. Por exemplo, considere o dimensionamento de transistores. O dimensionamento de transistores é feito baseado na informação das capacitâncias parasitas. Se entre

uma iteração e a seguinte de síntese de layout o posicionamento for alterado, todas as capacitâncias parasitas serão também alteradas, tornando o dimensionamento feito sem valor.

Para concluir este Capítulo, mostramos que TROPIC3 é uma eficiente ferramenta de síntese de layout, tanto em termos de densidade de integração quanto de tempo de CPU. Falta apenas mostrar quão longe as densidades obtidas por TROPIC3 estão da densidade ótima. Para isto foi feito um estudo, considerando-se o estilo regular de implantação. A Tabela 4 mostra a máxima densidade que pode-se obter sem roteamento. Consideraremos a área de roteamento proporcional à área de difusão dos transistores (área ativa), 50% e 25% para tecnologias com dois e três níveis de metal respectivamente.

Process ( $\mu\text{m}$ )	Densidade Máxima (tr/mm <sup>2</sup> )
$\lambda=0.8$	8296
$\lambda=0.6$	14748
$\lambda=0.5$	31332
$\lambda=0.35$	62644
$\lambda=0.25$	125328
$\lambda=0.18$	250656

**Tabela 4 – Densidades máximas que podem ser obtidas com o estilo de implementação adotado por TROPIC3, sem considerar o roteamento.**

Assim, para a tecnologia 0.25  $\mu\text{m}$  teremos uma densidade com roteamento de 94000 transistores/mm<sup>2</sup> ( $125328 \cdot 0.75$ ). A média obtida através da Tabela 3 é de aproximadamente 60000 transistores/mm<sup>2</sup>. A principal causa desta diferença é devido à abordagem tradicional de roteamento adotada, baseada em canais de roteamento.

Um ponto importante a ser investigado em versões futuras da ferramenta TROPIC é uma metodologia de roteamento, com roteamento completo sobre a área ativa.

O próximo Capítulo detalha o estilo de layout adotado pela ferramenta TROPIC3.

## 2.4 Referências bibliográficas

- [DET87] E.DETJENS; G.GANNOT; R.RUDELL; S.VIN-CENTELLI; A.WANG.  
*Technology mapping in MIS.*  
ICCAD, 1987, pp. 116-119.
- [CAD91] Virtuoso layout synthesizer - LAS - user guide.  
*CADENCE™ Version 4.2, October 1991.*
- [IEN98] P.IENNE, A.GRIEBING.  
*Practical Experiences with Standard-Cell Based Datapath Design Tools - Do We Really Need Regular Layouts?*  
DAC'98.
- [GUR97] M.GURUSWAMY; R.L.MAZIASZ; D.DULITZ; S.RAMAN; V.CHILUVURI; A.FERNANDEZ;  
L.G.JONES.  
*CELLERITY: A fully automatic layout synthesis system for standard cell libraries.*  
DAC'97.
- [HWA93] C.HWANG, Y.HSIEH, Y.LIN, Y.HSU.  
*An efficient layout style for two-metal CMOS leaf cells and its automatic synthesis.*  
IEEE Transactions on CAD, Vol. 12, No.3, March 1993, pp. 410-423.
- [LEF97] M.LEFEBVRE, D.MARPLE, C.SECHEN.  
*The Future of Custom Cell Generation in Physical Synthesis.*  
DAC'97.
- [LOP80] A.D.LOPEZ, H.S.LAW.  
*A dense gate matrix layout for MOS VLSI.*  
IEEE Transactions on Electron Device, Vol. ED-27, No. 8, August 1980, pp. 1671-1675.
- [MOR99] F. MORAES, M. ROBERT and D. AUVERGNE.  
*A Virtual CMOS Library Approach for Fast Layout Synthesis.*  
VLSI, 1999.
- [SAR96] SARRAFZADEH,M.; WONG,C.K.  
*An Introduction to VLSI Physical Desig.*  
McGraw-Hill, 1996, 334p.
- [UEH81] T.UEHARA, W.VANCLEEMPOT.  
*Optimal layout of CMOS functional arrays.*  
IEEE Transactions on Computers, Vol. C-30, No. 5, May 1981, pp. 305-312.

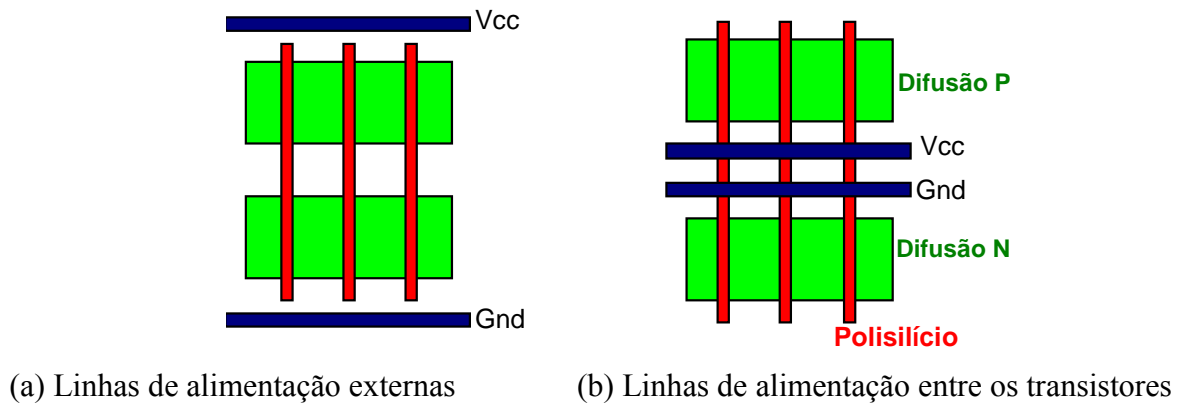
### 3 ESTILO DE LAYOUT DA FERRAMENTA TROPIC3

Este Capítulo apresenta o estilo de implantação adotado por TROPIC3, justificando as escolhas feitas. Ao final deste Capítulo apresenta-se a codificação das regras de projeto para uma dada tecnologia. É pressuposto para TROPIC3 a utilização de 3 níveis de metal e de contatos empilhados (conexão direta entre níveis não adjacentes, como polisilício e o segundo nível de metal).

Deve-se analisar o layout gerado pelo TROPIC3 em partes: a nível de célula, a nível de banda (linha horizontal de células) e a nível de layout completo (macro-célula).

#### 3.1 Layout a nível de célula básica

Os transistores são implementados horizontalmente, paralelos às linhas de alimentação. Esta técnica de posicionamento de transistores é denominada *linear-matrix*. A Figura 2 ilustra duas formas de implementação do estilo *linear-matrix*.



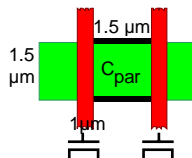
**Figura 2 - Alternativas para a implementação dos transistores segundo o estilo *linear-matrix*.**

A primeira alternativa, com as linhas de alimentação externas aos transistores sugere uma abordagem muito próxima à implementação *standard-cells*, com células de altura constante. A principal deficiência deste estilo de implementação ocorre quando o roteamento é feito entre os transistores, obrigando o afastamento dos planos N e P, e um conseqüente aumento no comprimento do polisilício vertical que une os transistores. Este aumento no comprimento do polisilício introduz capacitâncias e resistências parasitas que comprometem o desempenho elétrico do circuito. O segundo problema desta abordagem, independente se o roteamento é interno ou externo aos transistores, é o fato da altura de todas as células da mesma linha horizontal de células (banda) ser função da célula contendo os maiores transistores. Isto conduz a uma perda muito grande de área. Este estilo de implantação foi o adotado pelas ferramentas TROPIC2 e LAS.

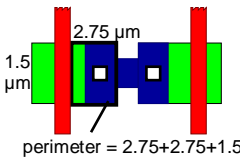
A segunda alternativa, linhas de alimentação entre os transistores, permite manter o comprimento mínimo nas linhas verticais de polisilício (proporcional ao afastamento entre os planos N e P), e permite que cada célula tenha um dimensionamento independente, sem afetar a área das células vizinhas. Esta topologia é adotada pela ferramenta TROPIC3.

Justifica-se o uso do estilo *linear-matrix* pelo fato de haver uma redução importante na capacitância lateral. A Figura 3 mostra 2 transistores conectados em série, de acordo com o estilo *linear-matrix*, sendo o primeiro par sem quebra na linha de difusão e o segundo par conectado por uma ponte em metal<sup>1</sup>. Muitos estilos de layout, como o *gate-matrix*, conectam todos os transistores

com pontes em metal1. A tecnologia para este exemplo tem os seguintes parâmetros:  $L_{\text{mim}}=1.0\mu\text{m}$ , capacitâncias de área e de perímetro para o nível difusão:  $C_{\text{area}}=0,31\text{ fF}/\mu\text{m}^2$  and  $C_{\text{side-wall}}=0,45\text{ fF}/\mu\text{m}$ . Observa-se que os transistores conectados por justaposição apresentam uma capacitância parasita 4 vezes menor que quando conectados por uma ponte em metal1. A partir desta observação, uma importante função custo que será utilizada pelo algoritmo de síntese de células será a minimização do número de quebras (*gaps*) nas linhas de difusão.

- Transistores justapostos:
 

$$C_{\text{par}} = C_{\text{AREA}} * 2.25\mu\text{m}^2 + C_{\text{side-wall}} * 3\mu\text{m}$$

$$C_{\text{par}} = 2.05\text{ fF}$$
- Conexão entre transistores via ponte de metal1 (gap de difusão)
 

$$C_{\text{par}} = 2 * (C_{\text{AREA}} * 4.12\mu\text{m}^2 + C_{\text{side-wall}} * 7\mu\text{m})$$

$$C_{\text{par}} = 8.86\text{ fF} (+ C_{\text{metal}} + C_{\text{contact}})$$

perimeter = 2.75+2.75+1.5

Figura 3 - Capacitâncias parasitas no estilo *linear-matrix*.

Uma vez definido o posicionamento dos transistores, deve-se decidir a direção dos níveis de metal. A escolha natural para as conexões de dreno e source dos transistores é **metal1** (primeiro nível de metal), pois caso um outro nível fosse utilizado seria necessário acrescentar contatos empilhados. A Figura 4 mostra a utilização de metal1 e metal2 a nível de célula, para uma porta *nand* de 3 entradas (3 transistores N em série e 3 transistores P em paralelo). O metal1 é utilizado para as conexões das regiões de dreno/source à alimentação, e para conectar o plano N ao plano P. Observar que a conexão dos nós de saída, entre o plano N e o plano P, é feita totalmente em metal1, sem contatos intermediários. O número elevado de contatos nas regiões de dreno/source objetivam a redução da resistividade nestas áreas. Caso a tecnologia empregada utilize silicetos nestas regiões, a inserção de apenas 1 contato é suficiente, o que reduziria o número de restrições topológicas impostas à ferramenta de roteamento. Esta característica não está atualmente implementada na ferramenta TROPIC3.

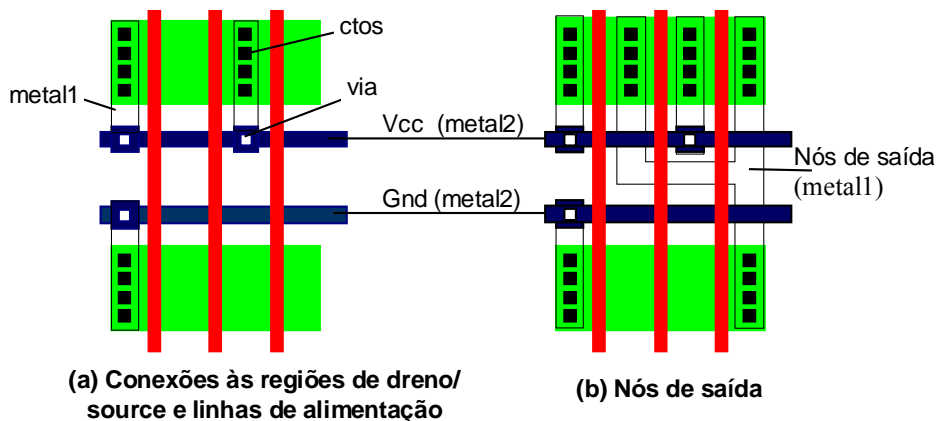


Figura 4 - Utilização de metal1 e metal2 para a síntese das células.

A escolha seguinte refere-se à implementação das linhas de alimentação (*gnd* e *vcc*). O nível metal1 não pode ser utilizado, devido aos obstáculos impostos pelos nós de saídas. O nível metal3 está muito distante do metal1, e reduziria a "porosidade" (transparência para o roteamento) da

célula. A escolha é o metal2. A largura das linhas de alimentação é definida no arquivo que descreve as regras de projeto. A cada conexão de um dreno/source à alimentação insere-se uma *via* (contato) entre os níveis metal1 e metal2.

Ainda, a nível de célula, temos as conexões ao substrato (*body-ties*). Estas podem ser facilmente realizadas, sobrepondo-se um retângulo de difusão e um contato sobre os nós conectados à alimentação. Desta forma, na atual versão da ferramenta TROPIC3, há sempre um *body-tie* para cada conexão à alimentação. Observar a existência de um contato empilhado (contato e via) em todas as conexões à alimentação.

A Figura 5 ilustra o layout completo de um conjunto de 4 células (o número de células pode ser deduzido do número de pontes de metal1 para conectar os transistores N aos transistores P). Observar neste layout a utilização dos níveis difusão, polisilício, metal1 e metal2. O polisilício utilizado para conectar os *gates* duais apresenta quebras (denominadas *jogs*), que são inseridas com o objetivo de reduzir a área das regiões de dreno/source. O mesmo ocorre com o metal1 utilizado para conectar os transistores N aos transistores P. Esta inserção de *jogs* é realizada pelo algoritmo de "síntese de banda", e vem de fato a substituir o algoritmo de compactação baseado em grafo de restrições.

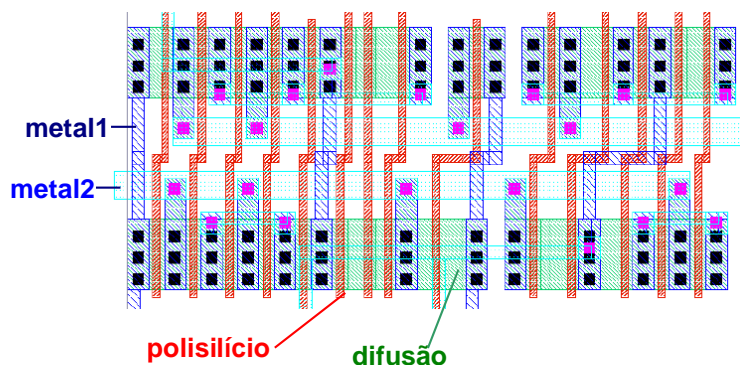


Figura 5 - Layout completo de um conjunto de células.

Uma otimização que pode tornar o layout mais denso é a utilização de múltiplos *jogs* nas linhas de metal1 e polisilício. Uma versão do TROPIC em desenvolvimento, denominada WTROPIC, contempla esta otimização. Um dos problemas ainda não resolvidos pela versão WTROPIC é onde posicionar os *body-ties*, pois as quebras de polisilício podem ocorrer sobre a *via* de alimentação.

Para finalizar o layout a nível de célula, mostra-se na Figura 6 a implementação de um flip-flop contendo portas de transmissão. Observa-se neste layout que os pares de portas de transmissão pertencentes a um mesmo estágio do flip-flop são agrupados, de forma a evitar a separação dos diversos componentes pertencentes à mesma célula. O pares de portas de transmissão são facilmente encontrados, bastando encontrar 2 portas de transmissão em série, com oposição de fase em relação ao controle de *clock*. Este par de portas de transmissão é denominado célula *macro*.

A estrutura do layout pode ser adaptada a portas não duais, modificando-se as restrições impostas ao algoritmo de síntese de células. Este exemplo de portas de transmissão mostra que pode-se implementar células sem necessariamente haver *gates* duais.



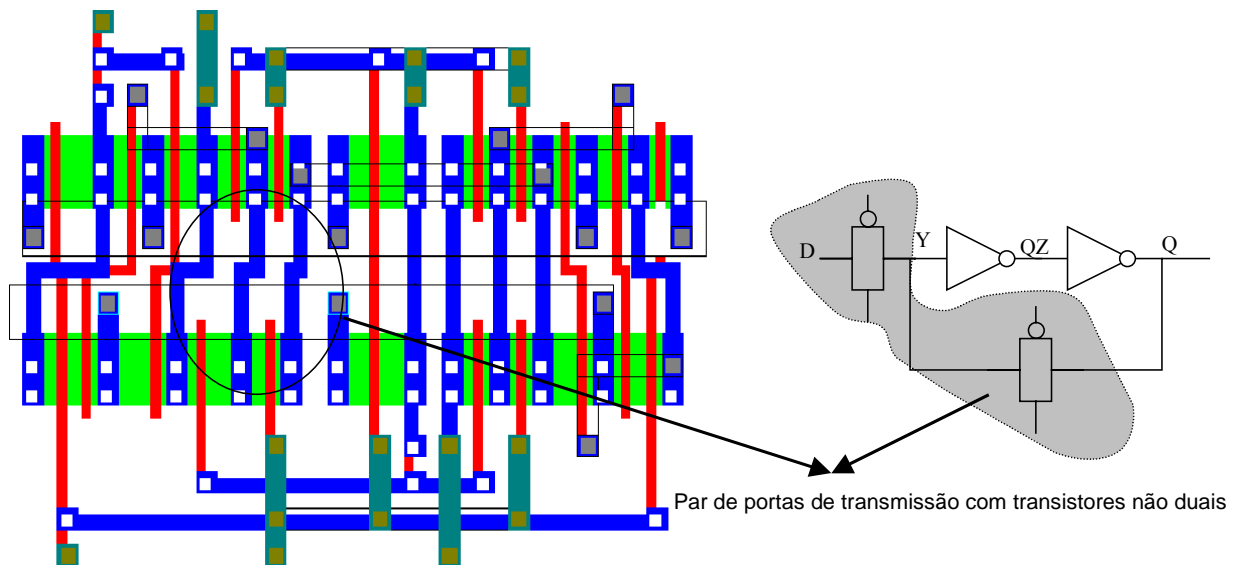


Figura 6 - Layout de um flip-flop mestre-escravo tipo D com reset, implementado com portas de transmissão.

### 3.2 Layout a nível de banda

A conexão entre uma determinada célula e suas vizinhas é feita através de contatos inseridos nos limites superiores e inferiores da célula. A região onde são inseridos os contatos é denominada *região de interface*, ilustrada na Figura 7.

Cada dreno/gate/source poderá ou não ser conectado às regiões de roteamento (regiões externa às células). Desta forma, os nós dos transistores classificam-se em:

- Nós externos: são aqueles que serão conectados às regiões de roteamento.
- Nós internos: são aqueles sem conexão às regiões de roteamento (é importante notar que existem nós internos que possuem conexões, porém sendo estas feitas sobre os transistores).

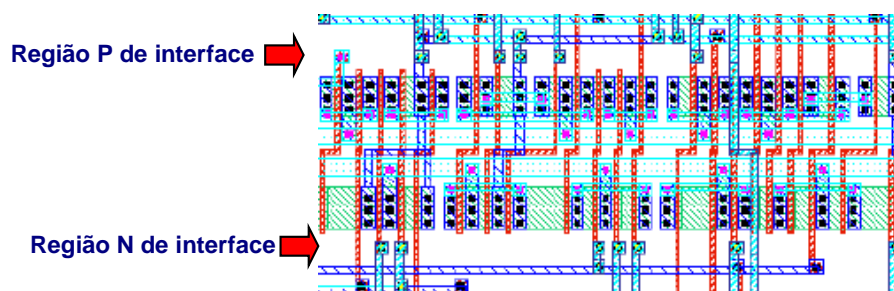


Figura 7 - Regiões N e P de interface entre as células e as regiões de roteamento.

Os nós externos serão alinhados a uma grade virtual, cujo passo é definido pelo tamanho da cabeça de contato da tecnologia. Os nós internos não possuem restrições de alinhamento, podendo utilizar as distâncias mínimas das regras de projeto. Esta característica resulta em redução de área (reduz a largura do circuito) e reduz as capacitâncias de dreno/source (melhorando o desempenho elétrico do circuito). Observar ainda na Figura 7 a existência de áreas de dreno/source superiores à área mínima. Este problema ocorre devido ao alinhamento de certos nós às regiões de roteamento. A inserção de múltiplos *jogs* nas linhas de polisilício e metal1 pode reduzir estas áreas (mas não eliminá-las).

Utilizando esta técnica de implementação, uma mesma função lógica poderá ter layouts diferentes, pois uma dada instância pode ter conexões apenas com a parte superior da banda, e outra



instância ter conexões apenas com a parte inferior. Neste caso, a primeira instância (conexões com o lado superior) terá a área dos drenos/sources P superior as áreas N, devido ao alinhamento dos nós P, e vice-versa para a segunda instância.

O algoritmo que determina de qual lado das células são feitas as conexões denomina-se assinalamento de pinos (do inglês *pin-assignment*). Este algoritmo é o que determina de forma indireta qual será a largura da banda, pois impõe quais pinos serão alinhados à grade.

Antes do roteamento ser realizado, um conjunto de redes que atende a certos critérios (como ser nó interno de porta complexa ou pertencer a apenas um banda) é excluído da lista de redes a serem roteadas. O roteamento deste conjunto de redes é feito sobre a área ativa do circuito, sem haver alinhamento à grade virtual de roteamento. Esta técnica de roteamento denomina-se OTC - *over the cell routing*. Os pinos que serão roteados sobre os transistores deixam de ser externos e passam a ser internos.

O roteamento OTC é feito em metal2, inserindo-se um via sobre os drenos/source. Se uma rede conecta apenas drenos/sources a sua implementação OTC é simples, sendo apenas uma linha horizontal de metal2 com vias nos pontos de contato. Caso uma linha OTC conecte também gates, faz-se necessário implementar um *jog* em metal2, até a região de interface, e lá inserir um contato que una o polisilício ao metal2. Estas conexões OTC acabam gerando obstáculos para outras redes OTC, devido à existência de segmentos verticais em metal2. Logo, há uma sequência na escolha das redes OTC a ser seguida: primeiro seleciona-se todas as redes que conectam apenas drenos/sources (nós internos de portas complexas) e depois as redes que possuem conexão a gates de transistores (redes presentes em apenas uma banda).

Todas as redes internas das portas complexas são roteadas sobre os transistores. Pode-se argumentar que isto irá aumentar em demasia a área reservada aos transistores, pois poderia ocorrer um número excessivo de trilhas em determinadas células. A literatura reporta um número máximo de 4 trilhas. Logo, esta decisão de projeto não irá penalizar a área final do circuito.

A Figura 8 resume as características vistas até agora, tanto a nível de célula quanto a nível de banda. No canto superior esquerdo da figura observa-se as duas topologias de rede OTC mencionadas no parágrafo anterior.

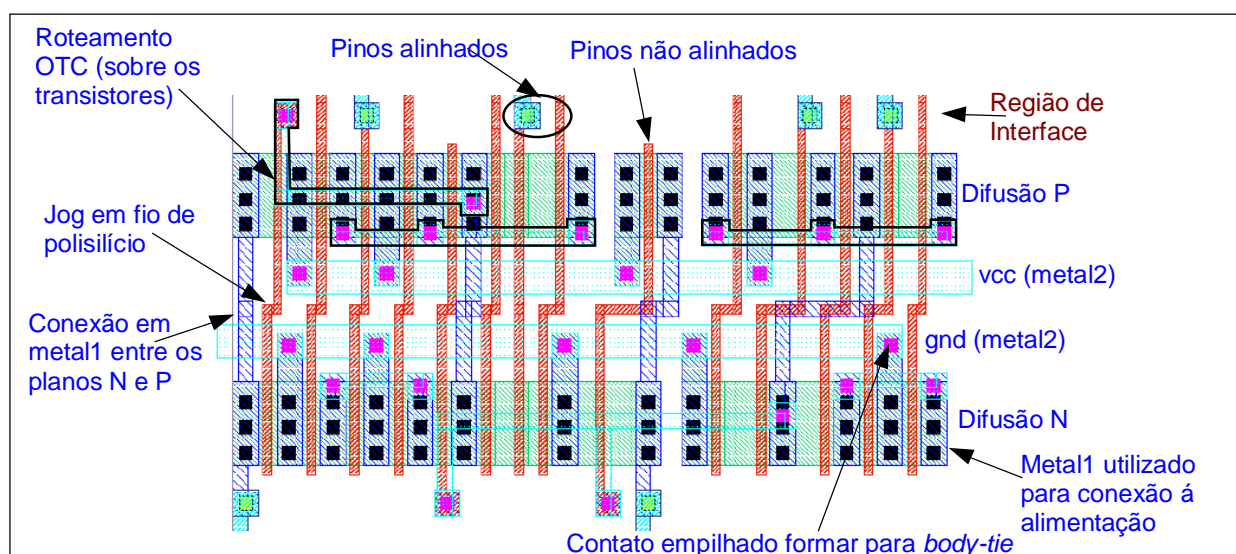


Figura 8 - Layout completo a nível de banda.

### 3.3 Layout a nível de macro-célula

A Figura 9 ilustra as regiões de roteamento assim como a utilização do terceiro nível de metal, o **metal3**. Tanto o layout a nível de célula quanto a nível de banda não utilizam o metal3. Logo, as bandas são transparentes ao metal3, permitindo o seu uso para a conexão vertical entre bandas não adjacentes.

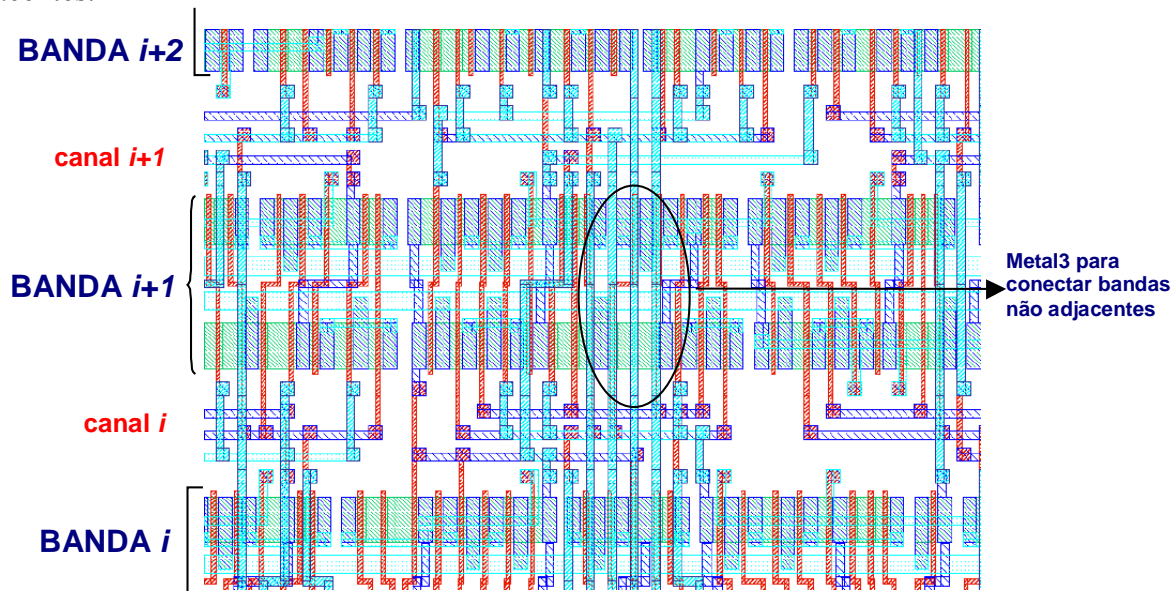


Figura 9 - Regiões de roteamento e utilização do metal3.

A primeira observação importante no que concerne a região de roteamento é a existência de dois canais superpostos para cada canal de roteamento. Em cada região de roteamento existe um canal implementado em polisilício/metal1 e outro superposto a este em metal2/metal3. A vantagem deste procedimento é a redução da área de silício ocupada para a implementação das conexões. A dificuldade na implementação deste procedimento está na resolução de ciclos verticais, como será explicado posteriormente na Seção relativa ao roteamento vertical. A escolha dos níveis de roteamento é detalhado na Tabela 5.

Região	Horizontal	Vertical
Banda (célula)	Difusão Metal2	Polisilício Metal1 Metal3
Roteamento	Metal1 Metal2	Polisilício Metal3

Tabela 5 - Níveis de roteamento.

As regiões de interface entre as bandas e os canais de roteamento tem por função básica "ajustar" os níveis. Por exemplo, se tivermos um linha vertical em polisilício no canal de roteamento com conexão a um gate de transistor, nenhum contato será necessário. Porém, se houver um metal3 vertical com conexão a um gate é necessário inserir um contato empilhado, compreendendo via2-via-contato. As regiões de interface pode ser facilmente observadas na Figura 9.

Uma crítica que pode ser feita à atual escolha dos níveis das regiões de roteamento é o fato do metal1 estar em paralelo ao metal2, o que pode causar grandes capacitâncias de acoplamento no caso em que a área comum entre dois fios for grande. A direção dos fios pode ser facilmente

modificada no algoritmo de roteamento. A escolha das direções foi arbitrária, tendo por objetivo apenas reduzir o número de contatos nas regiões de interface. A explicação é simples, há muito mais gates (polissilício) e redes entre bandas não adjacentes (metal3) que saídas de células (metal1). Por esta razão adotou-se a direção vertical para metal3 e polissilício, deixando a direção horizontal para metal1 e metal2.

Para concluir este estudo de estilo de layout a nível de macro-célula, a Figura 10 mostra os planos difusão/polissilício e metal1/metal2/metal3 de um circuito exemplo. Os objetivos em se mostrar esta figura são: (i) percebe-se pelo plano difusão/polissilício que aproximadamente 50% da área do circuito é dedicada à implementação dos transistores e 50% dedicada ao roteamento; (ii) há uma distribuição homogênea do roteamento vertical/horizontal (devido ao algoritmo de posicionamento); (iii) não há áreas de congestão no roteamento (também devido ao posicionamento); (iv) as entradas/saídas do circuito estão posicionados nas extremidades superior e inferior da macro-célula. O usuário pode definir no *netlist* que descreve a macro-célula o lado em que as entradas/saídas serão posicionadas (norte, sul, leste, oeste).

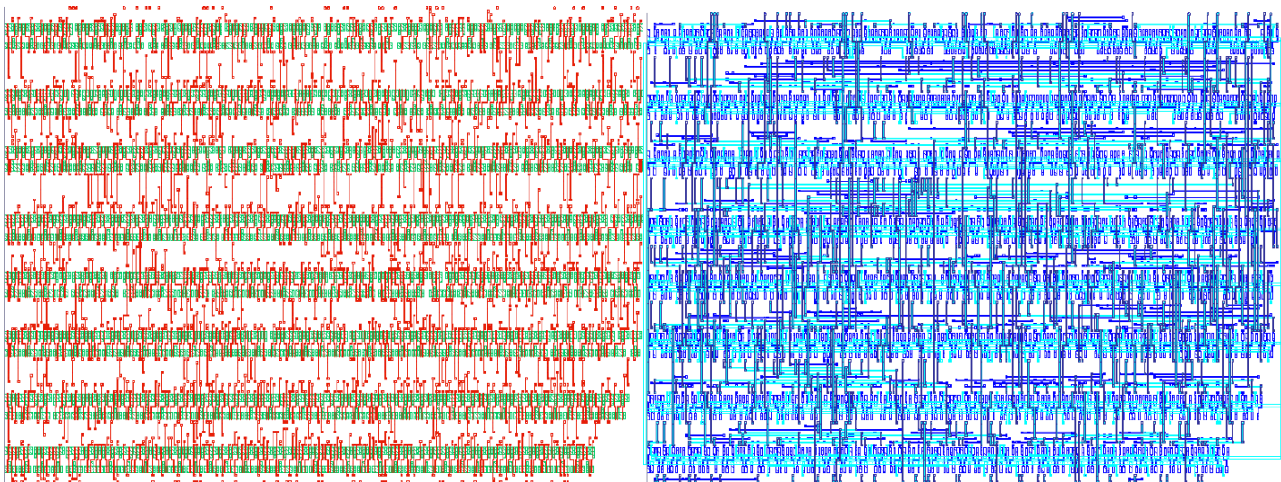


Figura 10 - Planos difusão/polissilício e metal1/metal2/metal3 de um circuito exemplo.

### 3.4 Codificação da tecnologia

Esta Seção mostra como definir as regras de projeto para uma dada tecnologia. A Figura 11 mostra o arquivo de regras para uma tecnologia 0.25  $\mu\text{m}$ . Este arquivo contém 4 partes principais: (i) diretórios de entrada e saída; (ii) regras geométricas; (iii) regras para geração do arquivo CIF, ou seja, camadas para o layout final; (iv) regras elétricas para o cálculo das capacitâncias parasitas (neste exemplo esta parte está incompleta).

As regras geométricas são descritas em termos de  $\mu\text{m}/1000$  (*nm*). Por exemplo, a distância entre dois polígonos de polissilício - DPOLI, é de 450, ou seja 0,45  $\mu\text{m}$ .

As regras referentes aos condutores (polissilício, difusão, metal1, metal2, metal3) compreendem apenas as regras de distância mínima (D) e largura mínima (L). Para cada contato (contato, via1 e via2) temos 3 regras: largura do contato (L), distância entre dois contatos (D) e margem (M) que deve ser respeitada. As regras dos contatos definem a largura da grade virtual que será utilizada no momento do roteamento. O passo da grade é definido por:

$$\text{grid} = \text{head} + \max(\text{DM1}, \text{DM2}, \text{DM3}) \quad (1)$$

onde





A seção seguinte às regras de desenho compreende as regras de tradução de camadas para o arquivo CIF. Define-se o comando que a ferramenta de edição de layout utiliza para os textos (*LABELCMD*) e uma tabela que relaciona o nome da camada utilizada pela ferramenta TROPIC3 e o nome da camada utilizada pela ferramenta de edição de layout.

O último conjunto de regras referem-se às capacitâncias parasitas. As capacitâncias de área são denominadas por  $C_a$  e as de perímetro por  $C_p$ . As capacitâncias  $C_j$  referem-se às capacitâncias de difusão. Estas regras são utilizadas pela ferramenta que determina as capacitâncias de roteamento.

*Este pequeno conjunto de regras de projeto, simples de modificar e de fácil compreensão, permite ao projetista rapidamente sintetizar o mesmo circuito para diferentes processos de fabricação. Este arquivo de regras permite uma rápida migração de uma tecnologia para outra. Esta é uma importante vantagem das ferramentas de síntese automática de layout sobre as abordagens baseadas em bibliotecas de células.*

### 3.5 Referências bibliográficas

- [BUR98] J.L.BURNS; J.A.FELDMAN.  
*C5M – A Control-Logic Layout Synthesis System for High-Performance Microprocessors.*  
IEEE Transactions on CAD, Vol. 17, no. 1, January 1998, pp. 14-23.
- [DUF90] J.C.DUFOURD, J.F.NAVINER, F.JUTAND.  
*Preform : A Process Independent Symbolic Layout System.*  
ICCAD 90, pp. 248-251.
- [FUK95] M.FUKUI, N.SHINOMUYA, T.AKINI.  
*A New Layout Synthesis for Leaf Cell Design.*  
Asia South Pacific DAC 95, pp. 259-264.
- [KIM94] J.KIM, S.M.KANG, S.SAPATNEKAR.  
*High Performance CMOS Macromodule Layout Synthesis.*  
ISCAS'94, pp.179-182.
- [LEE92] LEE, J.; WONG, C. K.  
*A Performance-Aimed Cell Compactor with Automatic Jogs.*  
IEEE Transactions on CAD, v. 11, n.12, p. 1495-1507, Dec. 1992.
- [MAZ92] R.L.MAZIASZ, J.P.HAYES.  
*Layout minimization of CMOS Cells.*  
Kluwer Academic Publishers, 1992.
- [MOR90] F.MORAES, R.REIS.  
*EXTRALO - Extrator Lógico.*  
5th SBCCI, 1990, pp. 167-176.
- [LEF89] M.LEFEBVRE, C.CHAN.  
*Optimal ordering of gate signals in CMOS complex gates.*  
CICC 89, pp. 17.5.1-17.5.4.
- [LIN91] M.LIN, H.PERNG, C.HWANG, Y.LIN.  
*Channel density reduction by routing over the cells.*  
28th DAC, 1991, pp. 120-125.
- [PIG88] C.PIGUET et al.  
*Alladin: a CMOS gate matrix layout system.*  
ISCAS 88, pp. 2427-2430.
- [WIN82] O.WING.  
*Automated gate matrix layout.*  
ISCAS 82, pp. 681-685.

## 4 SÍNTESE FÍSICA: DO NETLIST AO LAYOUT

Este Capítulo apresenta de forma sucinta as diversas etapas da síntese física. As etapas que compõe a ferramenta TROPIC3 são descritas abaixo.

1. Leitura dos arquivos Spice e de regras de projeto.
2. Planificação do netlist Spice. Esta etapa transforma o netlist Spice em uma lista plana de transistores, independentemente do formato da descrição de entrada ser hierárquico ou não.
3. Extração das células folha. Esta etapa encontra todas as células folha (aquelas compostas por um conjunto de  $n$  entradas e somente 1 saída) e portas de transmissão na lista plana de transistores. A largura de cada célula é uma função de seu número de entradas, dado o estilo de implantação regular.
4. Posicionamento das células folha. Esta etapa posiciona todas as células do circuito conforme o número de bandas especificado pelo usuário. Caso o usuário não especifique o número de bandas, este é calculado pelo programa de forma a gerar um circuito com altura igual à largura. A principal função custo de algoritmo de posicionamento é a redução da área de roteamento, distribuindo de forma homogênea o roteamento horizontal e vertical.
5. Geração da células folha. O algoritmo de síntese de células define a ordem dos transistores N e P para cada célula do circuito de forma a minimizar o número de quebras nas linhas de difusão e o número de trilhas internas de roteamento.
6. Assinalamento de pinos. Uma vez as células posicionadas nas bandas e a ordem de seus transistores definida, pode-se determinar a posição dos pinos de entrada e saída. Os pinos poderão ser conectados na extremidade superior ou inferior da célula. Nesta fase também são selecionadas as redes que serão roteadas sobre os transistores (roteamento OTC).
7. Geração de banda. É um algoritmo “guloso”, que gera cada banda da extremidade esquerda até a extremidade direita. Nesta etapa são posicionados os drenos/gates/source de acordo com o estado de cada nó. Se os nós devem ser conectados às regiões de roteamento eles são alinhados à grade virtual, caso não possuam conexões ou são nós OTC não serão alinhados à grade virtual. Ao final da geração de todas as bandas permite-se mover as células que estão nas extremidades das bandas para equilibrar as larguras entre as bandas e assim economizar área. **Não** há descrição simbólica, gera-se diretamente o layout em função das regras de desenho.
8. Fixação dos pontos de passagem entre bandas não adjacentes. Esta etapa fixa a posição das linhas verticais em metal3 que serão utilizadas para conectar bandas que não são vizinhas. Apesar de as bandas serem transparentes ao nível metal3, existem um série de restrições ao seu posicionamento, como será visto posteriormente.
9. Roteamento de canal. Esta etapa conecta os nós de cada canal, utilizando a técnica de dois canais superpostos, utilizando 4 níveis para o roteamento.
10. Geração do arquivo de layout em formato CIF.

As etapas 1 a 6 são independentes da tecnologia, sendo as etapas 7 a 10 dependentes das regras de projeto. Segue-se neste Capítulo a descrição das etapas 3 a 7. As etapas 1 e 2, por sua simplicidade de implementação, não são descritas.

## 4.1 Extração de células folha

Esta é a terceira etapa no fluxo de síntese. Até o início desta etapa a base de dados da ferramenta possui apenas duas listas de transistores, uma N e outra P, cada uma podendo conter, por exemplo, 20000 transistores.

Inicia-se o procedimento de extração de células folha pela redução do *netlist*. O procedimento de redução é mostrado na Figura 13. Passa-se para o algoritmo uma dada lista de transistores (N ou P), e o algoritmo busca nesta lista, enquanto houverem, transistores em paralelo e depois em série.

```

função reduz_netlist( lista de transistores )
{
    enquanto (   busca_transistores_em_paralelo(lista de transistores) ∨
                busca_transistores_em_série (lista de transistores)
            );
}

```

Figura 13 - Algoritmo de redução do netlist SPICE.

A Figura 14 mostra a execução da etapa de redução sobre a rede N de uma dada célula complexa. Ao final da execução da redução do netlist, teremos os conjuntos (N,P) de cadeias que representam **todas** as células folha do circuito. Os transistores que não foram reduzidos pelo algoritmo da Figura 13 podem ainda ser classificados como portas de transmissão. Uma porta de transmissão é uma ocorrência de um transistor P em paralelo com um transistor N. Todos os demais transistores não reduzidos correspondem a casos de lógica não dual, ocasionando um interrupção no programa, dado o fato que TROPIC3 não trata lógica que não for complementar, ou porta de transmissão.

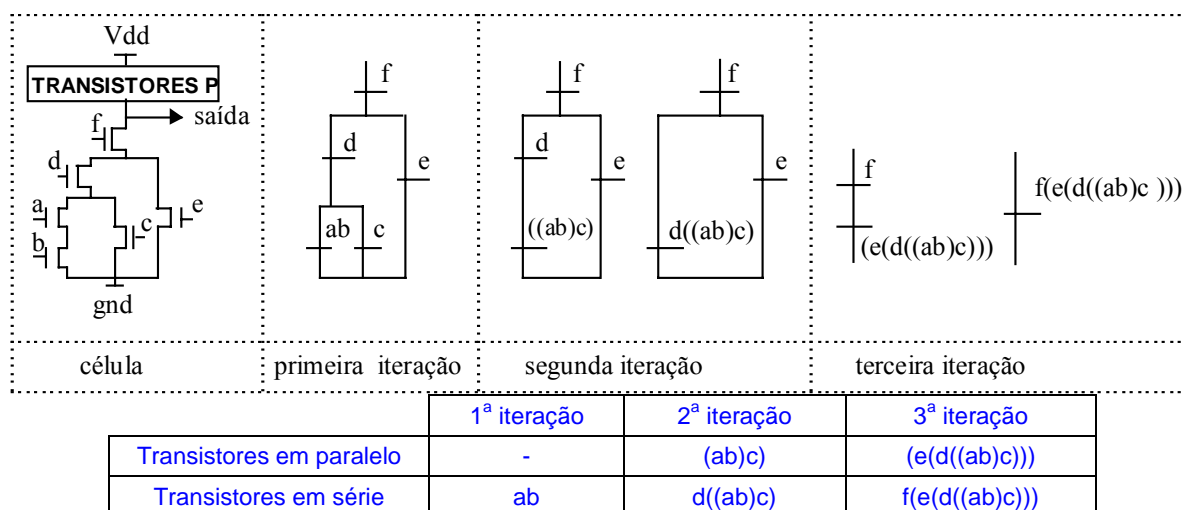


Figura 14 - Redução de uma rede N.

Observar que, se no *netlist* original tivermos portas *and* ou *flip-flops*, estas portas serão quebradas em mais de uma célula folha. As cadeias encontradas são representadas por um grafo, tendo por extremidade um nó de alimentação (*gnd* para a cadeia N, e *vcc* para a cadeia P) e na outra extremidade um nó qualquer. Logo, para formar uma célula folha basta encontrar uma cadeia N que possua um nó comum à outra cadeia P. Este nó comum é a saída da célula. Esta etapa é denominada **pareamento**. No final da etapa de pareamento teremos um vetor contendo as seguintes informações cada célula:

- cadeias N e P: permitem recuperar os transistores N e P da célula, e desta forma os nós dreno/gate/source, assim como as dimensões W/L;
- rede de saída;
- número de entradas, o que define a largura da célula;
- se é porta dual ou porta de transmissão.

A dimensão deste vetor é igual ao número de células folha mais o de portas de transmissão.

Este algoritmo é o utilizado pela ferramenta *EXTRALO* (extração lógica), pois permite recuperar a partir de um *netlist* SPICE todas as funções lógicas que compõe o circuito. Se na notação de parênteses utilizarmos '[' para designar transistores em paralelo e '(' para designar transistores em série, torna-se simples descrever a equação booleana da cadeia. A Figura 15 mostra a obtenção da equação booleana a partir da cadeia N que forma a célula. Há entre as ferramentas de TROPIC a ferramenta *SPI2VHDL*, que converte um *netlist* SPICE em VHDL estrutural.

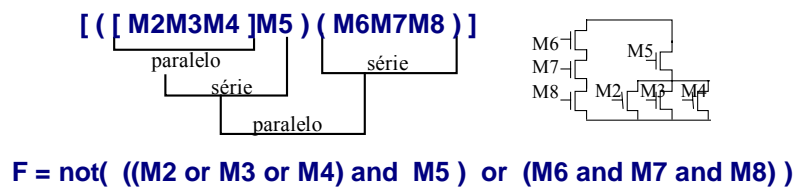


Figura 15 - Utilização da redução do netlist para determinar a equação booleana.

## 4.2 Posicionamento das células folha

Esta é a quarta etapa no fluxo de síntese. A ferramenta de posicionamento é externa, comunicando-se com TROPIC3 através de arquivos. A Figura 16 ilustra esta comunicação. TROPIC3 recebe o *netlist* SPICE e as regras de projeto. Envia para a ferramenta de posicionamento o arquivo reduzido (com as células folha) e um arquivo de biblioteca, contendo para cada célula folha o seu número de entradas e a largura. O posicionamento envia para o TROPIC3 as células posicionadas. Uma otimização feita no *netlist*, objetivando evitar que flip-flops tenham seus componentes internos posicionados separadamente, é a formação de células *macro*. As células *macro* agrupam as portas de transmissão internas à flip-flops, garantido assim que sejam posicionadas de forma justaposta.

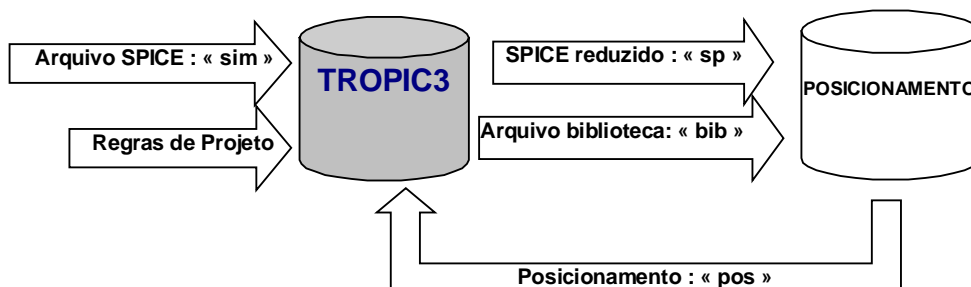


Figura 16 - Comunicação entre TROPIC3 e a ferramenta de posicionamento.

O algoritmo de posicionamento implementado pertence à classe dos algoritmos determinísticos. A escolha do algoritmo baseou-se em dois critérios: redução do comprimento total de roteamento e distribuição homogênea entre roteamento horizontal e vertical, evitando assim áreas de congestão nos centros das bandas.



O algoritmo de base é o *min-cut* (corte mínimo). Este algoritmo é utilizado para dividir o circuito iterativamente em quadrantes. O circuito é inicialmente dividido verticalmente, de tal forma que a área total da esquerda seja igual à área da direita, e haja um número mínimo de redes cruzando a fronteira entre os dois quadrantes (Figura 17, passo 1). A divisão seguinte, horizontal, deve respeitar o número de bandas. Por exemplo, se o número de bandas for 5, haverá uma relação de 3 para 2 entre as áreas superior e inferior. Depois, cada quadrante é sucessivamente dividido verticalmente e horizontalmente, até que não for mais possível realizar partições horizontais.

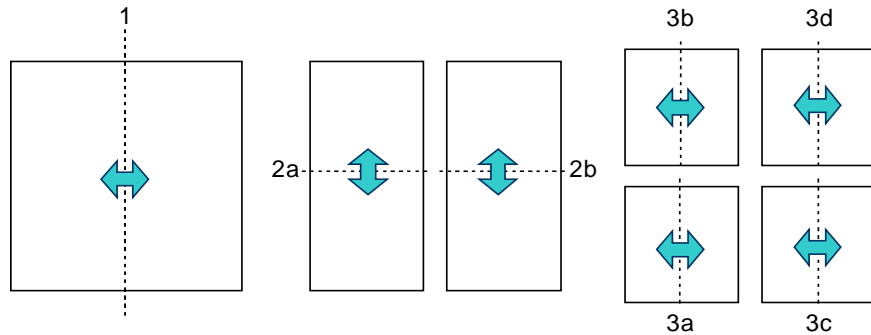


Figura 17 - Algoritmo de partição utilizando quadratura.

Ao final da quadratura haverá um grande conjunto de quadrantes, cada um possuindo em média 2 a 8 células. As células são posicionadas internamente aos quadrantes utilizando-se um algoritmo muito simples, baseado na conectividade entre as células.

Somente a utilização do algoritmo de quadratura não irá otimizar o roteamento. Considere a Figura 18. Se realizarmos a partição horizontal dos dois quadrantes de forma independente, acontecerá situações em que uma rede comum entre os dois quadrantes seja posicionada no quadrante esquerdo na parte superior e no quadrante direito na parte inferior, resultando numa conexão muito longa, apesar de haver corte mínimo nos quadrantes (Figura 18a). Se considerarmos a existência de milhares de rede, o roteamento terá uma qualidade medíocre. A solução para otimizar o roteamento é a **propagação de pinos**. Isto significa que cada quadrante a ser particionado deve levar em consideração todos os demais quadrantes já posicionados. O resultado da propagação de pinos é mostrada na Figura 18b. A propagação de pinos reduz o comprimento total do roteamento, melhorando assim o desempenho elétrico do circuito.

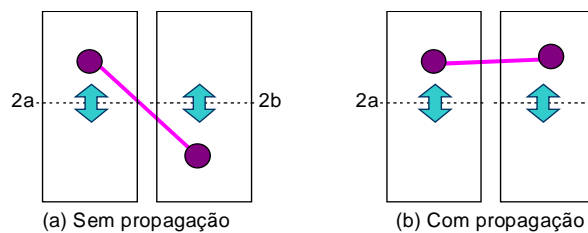


Figura 18 - Propagação de pinos.

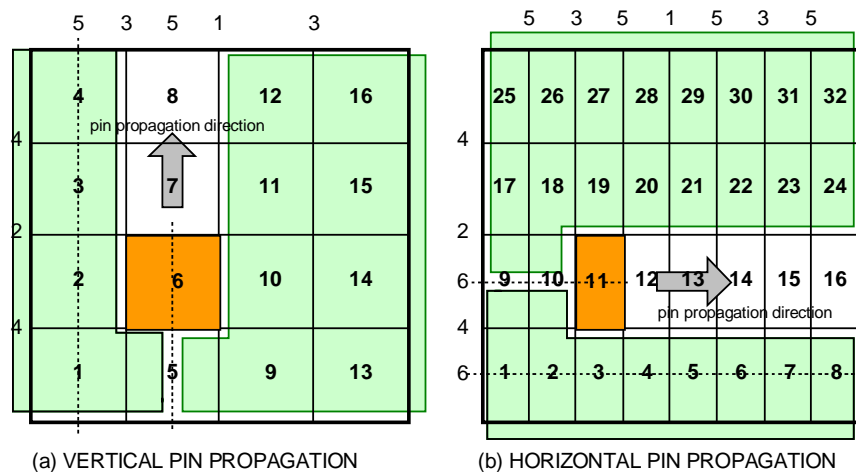
Para cada quadrante devemos determinar os sinais de interface. Estes sinais de interface são utilizados como *forças*, que atraem uma determinada célula para um dado quadrante. No exemplo da Figura 18 a célula do quadrante da direita seria *puxada* para cima, pois há uma vizinha à esquerda impondo uma interface. Para a partição horizontal, os quadrantes são processados linha a linha, da esquerda para a direita. Para a partição vertical, os quadrantes são processados coluna por coluna, de baixo para cima.

### Exemplo de propagação horizontal de pinos

Suponha que estejamos realizando a partição horizontal do quadrante de número 11 (Figura 19b). Os sinais que serão considerados interface para este quadrante são:

- Pinos sul:
  - entrada/saídas sul da macro-célula presentes no quadrante 11;
  - sinais comuns entre os quadrantes 11 com os quadrantes 1 a 8;
  - sinais comuns entre os quadrantes 11 e os sinais presentes na partição inferior dos quadrantes 9 e 10 (já particionados).
- Pinos norte:
  - entrada/saídas norte da macro-célula presentes no quadrante 11;
  - sinais comuns entre os quadrantes 11 com os quadrantes 17 a 32;
  - sinais comuns entre os quadrantes 11 e os sinais presentes na partição superior dos quadrantes 9 e 10 (já particionados).

A partição vertical de pinos é similar, utilizando-se pinos leste/oeste ao invés de pinos norte/sul.



**Figura 19 - Propagação horizontal e vertical de pinos.**

Observar que não há distinção entre partição e posicionamento. A saída do posicionamento de células é um arquivo texto, contendo o número de bandas, e para cada banda o número de células e a posição relativa de cada célula, da esquerda para a direita. Um exemplo de arquivo de saída do posicionamento é mostrado na Figura 20. Neste exemplo temos 3 bandas. A primeira banda contém 8 células, posicionadas na ordem X21, X13, X9, X15, X4, X17, X27, X2 (X indica célula, notação herdada do netlist SPICE para designar sub-circuito). As bandas 2 e 3 contém respectivamente 12 e 7 células, com a posição indicada na Figura. A banda 1 corresponde à banda inferior (sul) e a última banda, a banda superior (norte).

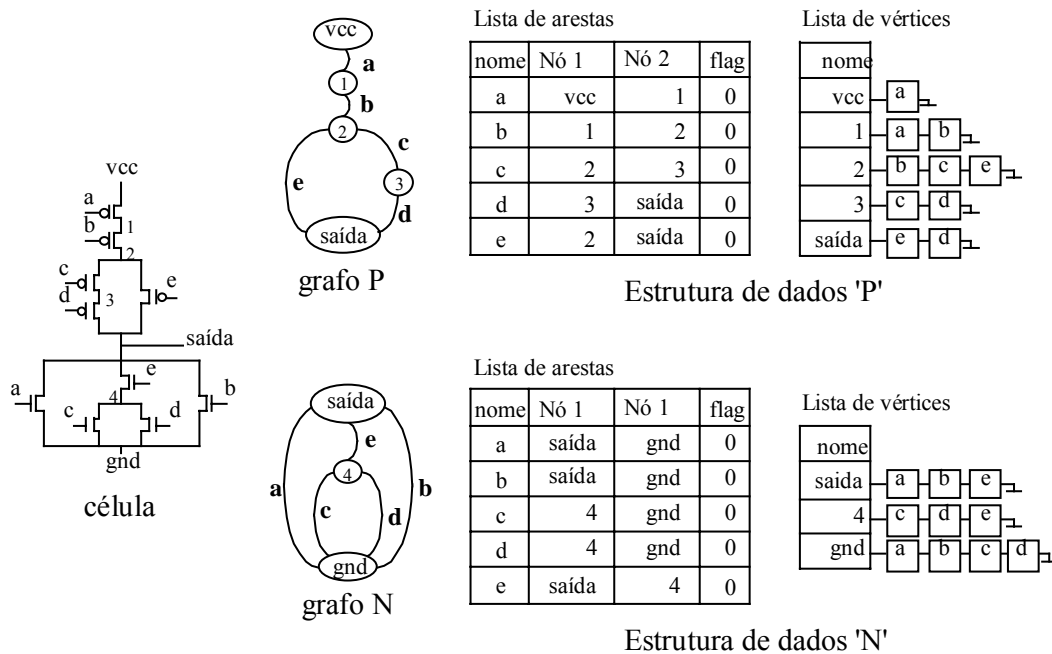
3	
8	X21 X13 X9 X15 X4 X17 X27 X2
12	X22 X6 X18 X26 X25 X12 X3 X11 X19 X1 X7 X14
7	X10 X5 X24 X20 X8 X16 X30

**Figura 20 - Arquivo exemplo de posicionamento.**

### 4.3 Geração de células folha

Esta é a quinta etapa no fluxo de síntese. Esta etapa poderia ser realizada em paralelo ao posicionamento, pois são etapas independentes. O objetivo desta etapa **não** é gerar o layout de cada célula, como poderia indicar o título desta seção. O objetivo é determinar a ordem dos transistores N e P no interior de cada célula.

A primeira etapa na geração é a construção da estrutura de dados necessária para a geração de cada célula. A Figura 21 ilustra a estrutura de dados para uma porta complexa. Os vértices correspondem aos nós de dreno/source e as arestas aos nós de gate. Cada grafo, N e P, são representados por duas listas, uma de arestas e outra de vértices. A lista de arestas contém para cada aresta (gate) suas duas extremidades (dreno/source), e a lista de vértices as arestas conectadas a cada vértice.



**Figura 21 - Grafo para representar os planos N e P de uma dada célula.**

Assim como na etapa de pareamento, nós iremos agora construir uma cadeia para cada grafo, que represente um caminho que passe por todas as arestas do grafo, passando uma única vez por cada aresta. Este caminho é denominado de caminho de Euler. A existência do caminho de Euler no grafo implica na síntese da linha de difusão sem quebras (*gaps*), o que minimiza as capacitâncias parasitas, conforme mencionado no Capítulo de estilo do layout. O algoritmo simplificado para encontrar o caminho de Euler é apresentado na Figura 22. É passado para este algoritmo uma aresta inicial  $a_i$  e um vértice a ela conectado  $v_k$ .

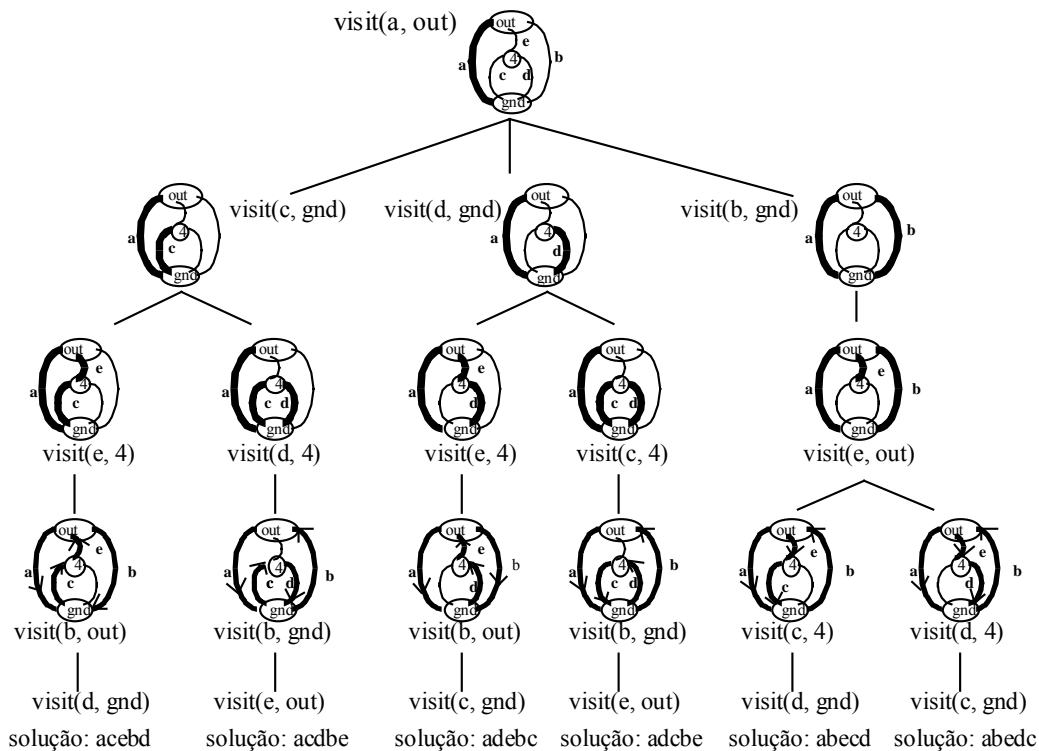
```

procedimento visitar_grafo( $a_i, v_k$ )
{
  • colocar  $a_i$  em um vetor temporário
  • sinalizar  $a_i$  como visitado ( $a_i.flag=1$ )
  • se todos os arcos do grafo já foram visitados
    então : armazenar o vetor temporário como uma solução temporária
    senão : para todos os arcos  $a_s$  conectados ao vértice vizinho de  $s_k$  : visitar_grafo( $a_s, v_{vizinho}$ )
  • retirar  $a_i$  do vetor temporário
  • liberar  $a_i$  ( $a_i.flag=0$ )
}

```

**Figura 22 - Algoritmo para o caminho de Euler.**

A Figura 23 mostra a execução do algoritmo acima, considerando como ponto de partida a aresta 'a' e o vértice “saída” (*out* na Figura). O número de vezes em que o algoritmo é executado em cada plano é proporcional ao número de arestas\*2 (pois cada aresta conecta 2 vértices).



**Figura 23 - Execução do algoritmo visitar grafo (*visit*) sobre a célula (AB(E(CD))) - Figura 21, plano N, à partir da aresta 'a' e do vértice 'out'.**

A execução do algoritmo no caso apresentado, gerou 6 ordenamentos de transistores que satisfazem a condição do caminho de Euler: ACEBD, ACDBE, ADEBC, ADCBE, ABEC e ABEDC. Ao executarmos o algoritmo sobre todas as arestas e vértices deste exemplo, obteremos 32 ordenamentos para o plano N e 4 ordenamentos para o plano P.

Uma vez criado os conjuntos de soluções N e P, deve-se realizar a interseção entre os dois conjuntos, de forma a encontrar o **mesmo** ordenamento N e P. Para o exemplo acima, teríamos as soluções CDEBA e ABEDC, simétricas entre si.

São adotados diversos critérios de desempate, quando existem soluções comuns: redução do número de linhas de roteamento interno (minimiza a altura da célula), redução do comprimento total do roteamento interno e por último o maior número possível de pontos de conexão à alimentação.

Caso não seja encontrada uma solução comum, mas existe ao menos uma solução em um dos planos, seleciona-se primeiro aquela que minimiza o número total de quebras nas linhas de difusão. Apesar de extremamente raro, pode acontecer ausência de caminho de Euler em ambos os grafos. Neste caso, o ordenamento acaba sendo feito pela concatenação de soluções parciais. Este caso específico não está otimizado na ferramenta TROPIC3, por representar uma situação particular, que pode ser resolvida alterando-se a ordem dos transistores no interior da célula, sem alterar a função lógica. Este procedimento, denominado *reordering*, pode ser automático, porém não está implementado. Então, caso aconteça no netlist uma célula sem caminho de Euler (o programa envia mensagem), o projetista pode alterar manualmente esta célula afim de obter um layout final de

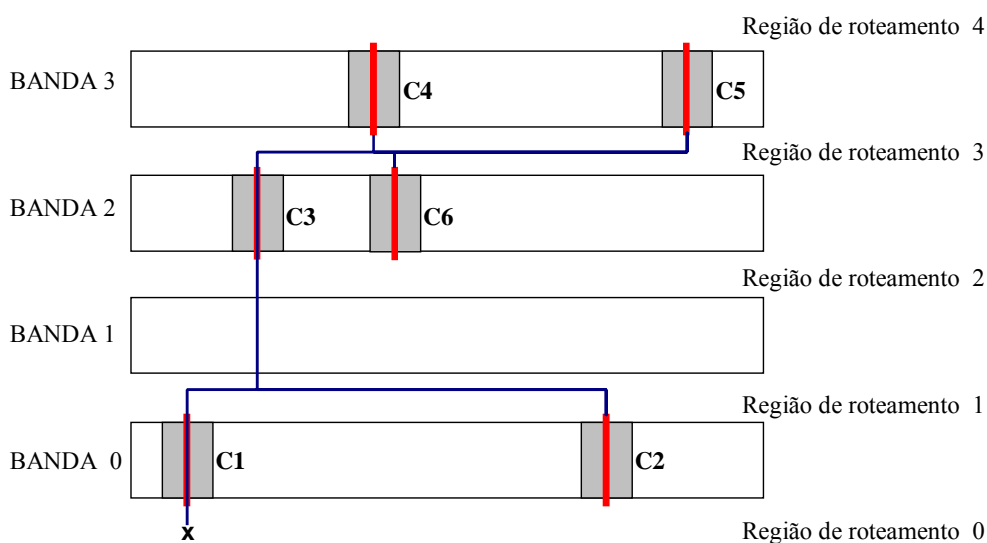
melhor qualidade.

*Ao final desta etapa de síntese, teremos em cada célula uma ordenação de transistores que implicará em um número mínimo de quebras de difusão.*

## 4.4 Assinalamento de pinos

Ao chegarmos neste ponto da síntese já temos a ordem relativa dos transistores no interior de cada célula, e a ordem relativa da posição de cada célula, em cada banda. Logo, é possível construir uma representação simbólica de todo o circuito, utilizando distâncias unitárias. Esta representação simbólica é interna à base de dados, não existindo em nenhum momento da síntese geração de descrição simbólica para ferramenta de compactação, como TROPIC2 ou LAS.

As células de cada banda que compõe o circuito podem ter conexões tanto pelo lado superior (*top*) quanto pelo lado inferior (*bottom*). A Figura 24 ilustra uma rede "x", que é interface sul<sup>1</sup>, conectada à 6 células, C1 a C6. O roteamento ótimo implica que o algoritmo utilize os seguintes pinos: *top/bottom* da célula C1, *top* C2, *top/bottom* C3, *bottom* C4, *bottom* C5, *top* C6. Observar que será necessária a passagem de uma linha em metal3 sobre a segunda banda (BANDA 1).



**Figura 24 - Exemplo de assinalamento (correto) de pinos.**

Há uma otimização realizada **antes** de iniciar o assinalamento de pinos. Esta otimização é a seleção dos pinos que terão roteamento sobre os transistores, ou pinos OTC. Estes pinos conectam redes internas às portas complexas ou são redes que pertencem apenas a uma única banda. Os nós OTC são pré- marcados como *mortos*, e não são considerados durante o assinalamento de pinos. Eles serão roteados separadamente das demais redes.

O algoritmo de assinalamento inicialmente marca todos os pinos da rede como **livres** e determina em quais regiões de roteamento (*canais*) a rede encontra-se. Depois, executa-se o algoritmo da Figura 25.

<sup>1</sup> restrição de posição de entrada ou saída (interface) na borda do circuito, imposta pelo usuário, podendo ser norte, sul, leste e oeste.

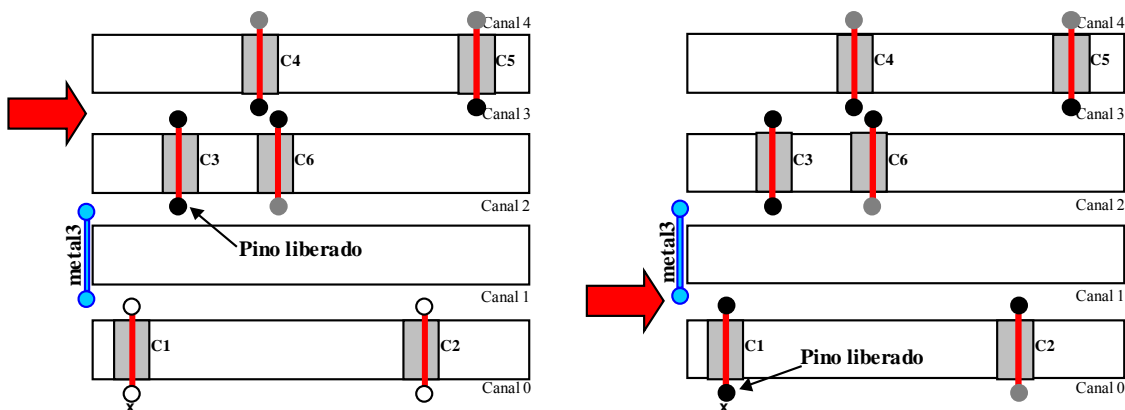
faça{

1. escolher região de roteamento (canal  $i$ ) com o maior número de pinos **livres**;
  2. se não houverem pinos livres no canal  $i$  ir para o passo 7;
  3. para as células com pinos inferiores livres (top do canal), marcar os pinos como **usados**, e **matar** os pinos superiores destas células;
  4. para as células com pinos superiores livres (bottom do canal), marcar os pinos como **usados**, e **matar** os pinos inferiores destas células;
  5. caso as células com pinos inferiores tenham conexão nos canais superiores, **liberar** um dos pinos superiores que foram mortos no passo 3. O pino liberado é escolhido de forma a ter sua coordenada dentro, ou o mais próximo possível, do intervalo da rede nos canais superiores ao canal  $i$ .
  6. caso as células com pinos superiores tenham conexão nos canais inferiores, **liberar** um dos pinos inferiores que foram mortos no passo 4. O pino liberado é escolhido de forma a ter sua coordenada dentro, ou o mais próximo possível, do intervalo da rede nos canais inferiores ao canal  $i$ .
  7. marcar canal  $i$  como assinalado;
- } enquanto houverem regiões sem assinalamento de pinos efetuado

**Figura 25 - Algoritmo de assinalamento de pinos.**

A Figura 26 ilustra o assinalamento de pinos para o exemplo da Figura 24. A rede encontra-se presente em todos os canais. A execução do algoritmos seria:

- escolhe-se o canal 3 (4 pinos livres), marca-se estes 4 pinos como *usados* (em preto), e os demais (pinos superiores do canal  $i+1$  e inferiores do canal  $i-1$ ) como *mortos* (em cinza). Como nos canais superiores ao canal  $i+1$  não há pinos livres, não há necessidade de liberar nenhum pino. Porém como nos canais  $i-1$  há pinos livres, *libera-se* um pino cuja coordenada esteja dentro do intervalo das coordenadas da rede nos canais  $i-1$ ;
- escolhe-se agora o canal 1, pois ele tem **3 pinos livres** (células C1 e C2 e ponte em metal3 a ser inserida sobre a banda 1). Novamente temos 2 pinos marcados como *usados*, e um liberado no canal 0. O pino escolhido é aquele conectado na interface sul.
- como não há mais nenhum canal com pino livre, o algoritmo termina.



**Figura 26 - Execução do assinalamento de pinos. O ponto de passagem em metal3 é determinado posteriormente, em função das restrições geométricas impostas na banda. Ele é considerado como pino livre na contagem de pinos nos canais.**

O algoritmo de assinalamento de pinos é o que define de forma indireta a largura de cada banda, pois todos os pinos marcados como *usados* serão alinhados à grade virtual de roteamento. Todos os demais nodos são posicionados livremente, em função das distâncias mínimas especificadas no arquivo de regras da tecnologia.

## 4.5 Geração de banda

Esta sétima etapa é a responsável por determinar a coordenada **real** (geométrica) de cada dreno/gate/source que compõe o circuito. O algoritmo implementado varre cada banda, da primeira à última, da esquerda para a direita, coluna a coluna. O algoritmo básico é mostrado na Figura 27. Ao final da geração de todas as bandas, caso haja diferença na largura destas, permite-se mover as células que estão nas extremidades das bandas para equilibrar as larguras entre as bandas e assim economizar área.

*para todas bandas  $r$  do circuito*

*para todas as células  $c$  da banda  $r$*

*para todos os transistores  $t$  da célula  $c$*

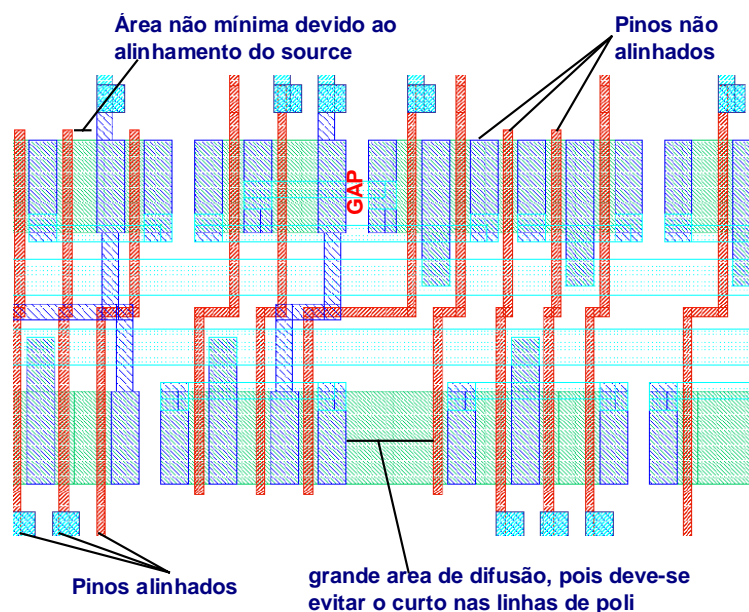
*para os planos  $n$  e  $p$  de  $t$  {*

- *determinar a coordenada em microns do dreno  $t$  referente ao plano  $n$  ou  $p$ , levando-se em conta se o dreno deve ser alinhado ou não à grade virtual. Se for **alinhado**, fixar a coordenada geométrica com a primeira posição livre da grade disponível. Se **não** for alinhado, utilizar regra de distância mínima em relação ao gate anterior. Caso a coordenada sobreponha-se a outra rede, avançar a coordenada do dreno (caso de colisão em *metal1* com saída de célula anterior);*
- *determinar a coordenada em microns do gate  $t$  referente ao plano  $n$  ou  $p$ . Mesmas restrições que o dreno, tomando-se o cuidado de não se colocar linhas de polisilício em curto-circuito;*
- *determinar a coordenada em microns do source  $t$  referente ao plano  $n$  ou  $p$ , segundo as mesmas restrições.*

*}*

**Figura 27 - Algoritmo de geração de banda.**

A Figura 28 ilustra o layout de uma banda, considerando-se o procedimento acima. Nesta Figura pode-se observar claramente como o layout é montado. Há uma tendência, neste exemplo, das linhas de polisilício apresentarem jogs para a direita, pois nesta banda a maior parte das conexões está sendo feita pela lado superior, e por consequência os alinhamentos serão feitos neste lado. Observa-se também uma grande área de difusão N. Esta deve-se a um quebra (*gap*) no plano P.



**Figura 28 - Geração de Banda.**



Apesar de acontecerem algumas áreas de difusão que possam introduzir capacitâncias parasitas, o algoritmo gera de forma eficiente as bandas, com poucas áreas não mínimas. Há duas formas de otimizar este procedimento de geração. A primeira é permitir a inserção de múltiplos *jogs* nas linhas de polisilício. O perigo desta solução é a geração de topologias do tipo "escada", como ocorrem nos compactadores baseados em grafo de restrições. A segunda solução é realizar ao menos duas passagens por banda, a primeira da esquerda para a direita (como está-se fazendo) e uma segunda, da direita para a esquerda, uniformizando as áreas de difusão.

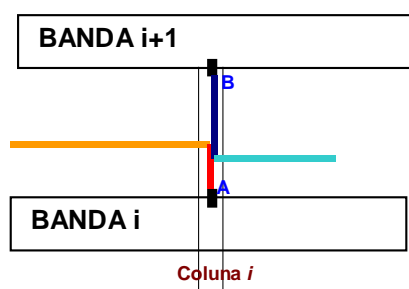
*Esta etapa ainda não gerou o layout, porém já temos as coordenadas em microns de todos os drenos/gates/source que compõem o circuito.*

## 4.6 Fixação dos pontos de passagem entre bandas não adjacentes

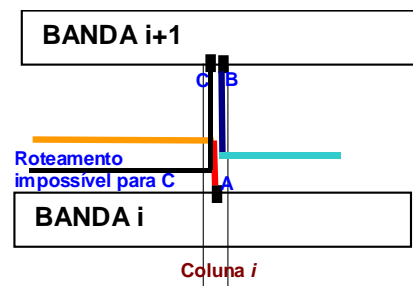
Esta etapa fixa a posição das linhas verticais em metal3 que serão utilizadas para conectar bandas que não são vizinhas (ver Figura 24). O termo utilizado para designar estas redes de metal3 que são posicionadas sobre as bandas é *feedthrough*, termo herdado da metodologia standard-cells.

Apesar das bandas serem totalmente transparentes ao metal3, um conjunto de 4 regras restringe o uso deste nível. Estas regras são aplicadas para que o roteamento de canal possa depois ser realizado. As regras são:

1. Os *feedthroughs* são alinhados à grade vertical de roteamento.
2. O número de redes diferentes por coluna no canal de roteamento é no máximo 2 (Figura 29). Como utilizamos 4 níveis de roteamento, poderiam haver em cada coluna até 4 redes diferentes. Porém com temos apenas 2 níveis verticais, uma terceira rede na mesma coluna pode tornar o roteamento impossível. Por outro lado, podemos ter mais que 2 pinos por coluna, desde que seja mantida a restrição de duas redes, exemplo com 3 pinos, no lado superior redes A e B e no lado inferior rede A.



(a) duas redes na mesma colunas são sempre roteáveis com 4 níveis de roteamento.



(b) Caso tenhamos 3 redes na mesma coluna, o roteamento pode até ser realizável em alguns caso, mas como o apresentado é impossível.

**Figura 29 - Restrição de duas redes por coluna.**

3. Não são permitidos *feedthroughs* sobre gates de transistores OTC. A razão para isto é muito simples, necessita-se de uma via para conectar o metal2 do roteamento OTC ao polisilício de gate. Se colocarmos um *feedthrough*, provavelmente será necessário uma via2 para conectar este *feedthrough* ao canal de roteamento, o que geraria um curto circuito com o nó OTC.
4. Não são permitidos *feedthroughs* pertencentes a diferentes redes em bandas adjacentes.



Uma vez determinado o conjunto de restrições, ordena-se as redes conforme o número de *feedthroughs* que devem ser inseridos. Inicia-se pelas redes com o maior número de *feedthroughs*, pois estas são as redes mais longas.

O procedimento para determinar a coordenada dos *feedthroughs* é baseado no cálculo dos intervalos das coordenadas da rede nas bandas superiores e inferiores, como o procedimento de assinalamento de pinos. Uma vez determinada a posição do *feedthrough* na banda, verifica-se se não há restrições na coordenada. Havendo restrições, passa-se a procurar posições livres à esquerda e à direita. A inserção do *feedthrough* é feita de forma independente no lado superior e inferior, pois nem sempre há a mesma coluna livre. O resultado é uma linha de metal3 com *jog*, assim como no polisilício dos gates e metal1 dos nós de saída. A Figura 30 mostra as linhas de metal3 sobre uma determinada banda. Observar no centro da Figura uma linha de metal3 com *jog* inserido.

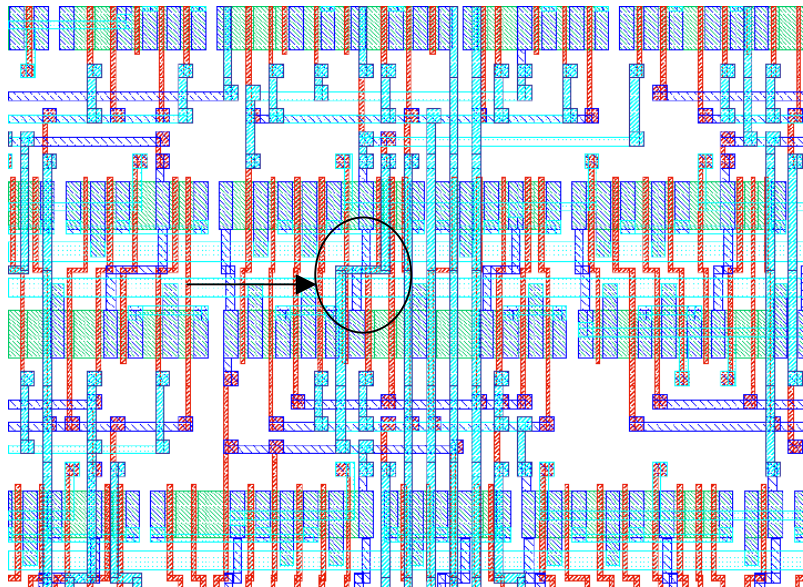


Figura 30 - Inserção de feedthrough.

Esta é a única etapa que pode abortar a síntese de layout, devido à falta de espaço para inserir *feedthroughs*. A solução indicada quando isto acontece é ou utilizar um número menor de bandas (reduz o roteamento vertical) ou substituir as interfaces norte/sul por interfaces leste/oeste (reduz o número de redes que cruzam verticalmente o circuito).

## 4.7 Roteamento

Ao chegarmos nesta etapa, temos todos os canais de roteamento definidos, com os nós pertencentes às bandas (etapa de geração de banda) e com os nós que passam sobre as bandas (etapa de inserção *feedthroughs*). O algoritmo empregado é muito simples, baseado no procedimento *left-edge*. Todo o roteamento é simbólico, sobre a grade virtual. Os polígonos referentes ao roteamento são gerados na última etapa de síntese, geração do arquivo de layout (formato CIF).

A principal otimização realizada é a utilização de dois canais superpostos, utilizando 4 níveis de roteamento. Com isto consegue-se reduzir à metade a área de conexões. A Figura 31 ilustra esta idéia. Há um canal inferior, implementado em polisilício/metal1 conectado por um contato; e um canal superior, implementado em metal2/metal3 conectado por uma via2. Estes canais são independentes. Raras situações de ciclos verticais ocasionam a utilização de dois condutores verticais para o mesmo condutor horizontal.

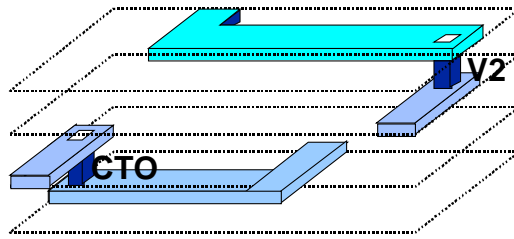


Figura 31 - Roteamento com 4 níveis.

O algoritmo de roteamento, de forma simplificada, é mostrado na Figura 32.

Para todos os canais de roteamento:

(a) Trilha = 0;

(b) Repetir enquanto houverem redes a serem roteadas no canal:

(b.1) Inserir na trilha corrente o maior número possível de redes, iniciando na coordenada esquerda do canal indo até a extremidade direita, utilizando o nível de roteamento **horizontal1**. Para que uma rede seja inserida, não devem haver restrições verticais (ver Figura 33).

(b.2) Repetir 'b.1', para o nível **horizontal2**. Estamos neste caso sobrepondo 2 canais de roteamento.

(b.3) Se 'b.1' e 'b.2' falharem (restrições verticais em todos os fios), insere-se redes utilizando ambos níveis verticais para o mesmo nível horizontal. Neste caso não é permitido sobreposição de roteamento.

(b.4) track = track+1;

Figura 32 - Algoritmo de roteamento.

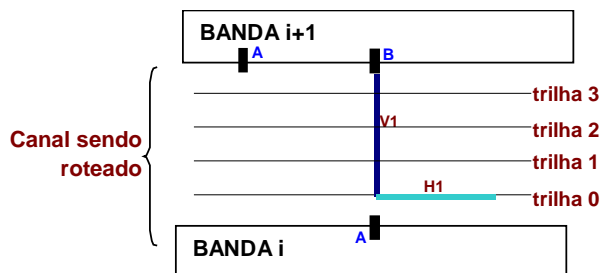


Figura 33 - Exemplo de restrição vertical. Considere que a trilha 0 já foi roteada, e que a rede B utilizou nesta trilha os níveis H1 (horizontal1) e V1 (vertical1). Ao tentarmos rotear na trilha 1 a rede A com o nível H1, haverá colisão no nível V1, impossibilitando o uso de H1 para a rede A.

A superposição dos níveis de roteamento e o caso de uma rede roteada com os 2 níveis horizontais é mostrado Figura 34

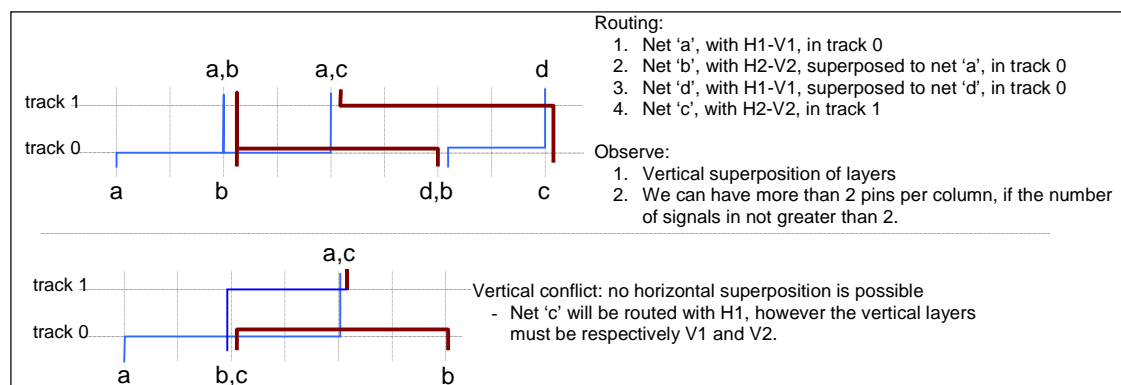


Figura 34 - Procedimento de roteamento.

A escolha dos níveis de roteamento H1/V1/H2/V2 foi arbitrária, tendo por objetivo apenas

reduzir o número de contatos nas regiões de interface. Foram escolhidos os níveis: H1 - metal1, V1 - polisilício, H2 - metal2 e V2 - metal3. O fato do metal1 estar em paralelo ao metal2, pode causar grandes capacitâncias de acoplamento caso a área comum entre dois fios for grande. Pode-se alterar facilmente o roteador para utilizar metal3 horizontalmente. O único efeito negativo desta escolha será a inserção de vias2 em todos os *feedthroughs* de metal3, pois a direção destes é vertical, e no canal de roteamento a condutor vertical será o metal2. Solução mais eficiente em termos de área seria limitar a sobreposição entre metal1/metal2, conforme um parâmetro inserido no arquivo de regras de projeto.

## 4.8 Geração do arquivo de layout em formato CIF

Esta é a última etapa do procedimento de síntese. O procedimento básico é varrer a estrutura de dados dos canais e das bandas, gerando os polígonos conforme as regras de projeto. As coordenadas horizontais já estão definidas: nos canais pela grade virtual e nas bandas pela etapa de geração de banda. A altura do circuito é até o momento desconhecida. Logo, o procedimento de expansão (geração de polígonos) procede na seguinte ordem: *canal 0, banda 1, canal 1, banda 2, canal 2, ..., banda n-1, canal n*, onde *n* designa o número de bandas do circuito. Insere-se um pente de alimentação, em metal2, conforme a largura da alimentação definida no arquivo de regras.

## 4.9 Referências bibliográficas

- [BHI93] S.BHINDARDE, A.PANYAM, N.A.SHERWANI.  
*On optimum cell models for over-the-cell routing.*  
6th International Conference on VLSI Design, 1993, pp. 94-99.
- [CAR92] B.S.CARLSON, C.Y.R.CHEN, D.S.MELIKSETIAN.  
*An efficient algorithm for the identification of dual eulerian graphs and its application to the cell layout.*  
ISCAS 92, 1992, pp. 2248-2251.
- [CON90] J.CONG, B.T.PREAS, C.L.LIU.  
*General Models and algorithms for the over-the-cell routing in standard-cell design.*  
27th DAC, 1990, pp. 709-715.
- [DON90] DONATH, W.E. et al.  
*Timing Driven Placement Using Complete Path Delays.*  
27th DAC, 1990, pp. 84-90.
- [KIN96] J.KIM, S.M.KANG.  
*A New Triple-Layer OTC Channel Router.*  
IEEE Transactions on CAD, v. 15, no. 9, September 1996, pp. 1059-1070.
- [FID82] C.M.FIDUCCIA, R.M.MATTHEYSES.  
*A linear time heuristic for improving network partitions.*  
19th DAC, 1982, pp. 175-181.
- [KAH95] KAHNG, A.B., ROBINS, G.  
*On Optimal Interconnections for VLSI.*  
Kluwer Academic Publishers, 1995, 286p.
- [KER70] B.W.KERNIGHAN, S.LIN.  
*An efficient heuristic procedure for partitioning graphs.*  
Bell System Technical Journal, No. 49, February 1970, pp. 291-308.
- [LIN92] H.LIN, H.PERNG, Y.HSU.  
*Cell height reduction by routing over the cells.*  
ISCAS 1992, pp. 2244-2247.
- [MOR94] MORAES, FERNANDO.  
*Synthèse de Macro-Cellules en Technologie CMOS.*  
Montpellier, França, Université de Montpellier II - Sciences et Techniques du Languedoc, 1994. 180p.  
Thèse Doctoral.

- [PRE88] B.T.PREAS, M.J.LORENZETTI.  
*Physical Design Automation of VLSI Systems.*  
Benjamin/Cummings Publishing, 1988. 510p.
- [REK95] S.REKHI; J.D.TROTTER; D.H.LINDER.  
*Automatic layout synthesis of leaf cells.*  
DAC'95.
- [SHE95] SHERWANI, Naveed.  
*Algorithms for VLSI Physical Design Automation.*  
Kluwer Academic Publishers, Boston, 1995. 538p.
- [TER94] M.TERAI, K.NAKAJIMA, K.TAKAHASHI, K.SATO.  
*A New Approach to Over-the-Cell Channel Routing with Three Metal Layers.*  
IEEE Transactions on CAD, Vol. 13, No 2, February 94, pp. 187-200.
- [TSU95] M.TSUCHIYA, T.KOIDE, S.WAKABAYASHI, N.YOSHIDA.  
*A Three-Layer Over-the-Cell Multi-Channel Routing Method for a New Cell Model.*  
Asia South Pacific DAC 95, pp. 195-202.
- [VEL96] A.J.VELASCO; X.MARÍN; R.PESET; J.CARRABINA.  
*Performance driven layout synthesis: optimal pairing and chaining.*  
Fifth ACM/SIGDA Physical Design Workshop, April 1996, pp. 176-182.

## 5 EXTRAÇÃO DE ELEMENTOS PARASITAS

Com a diminuição do tamanho dos transistores para dimensões sub-micrônicas, o tamanho das interconexões passou a contribuir de maneira significativa para o atraso do circuito. Isto pode ser visto na Figura 35, onde nas tecnologias de 0,25  $\mu\text{m}$ , as capacitâncias de interconexão, formadas por metalização de alumínio e dielétrico de dióxido de silício, contribuem em 50% do atraso total. Já para as tecnologias 0,18  $\mu\text{m}$ , a contribuição é de 70%, e para tecnologias 0,15  $\mu\text{m}$ , espera-se que a capacitância das interconexões contribuam 80% do atraso total.

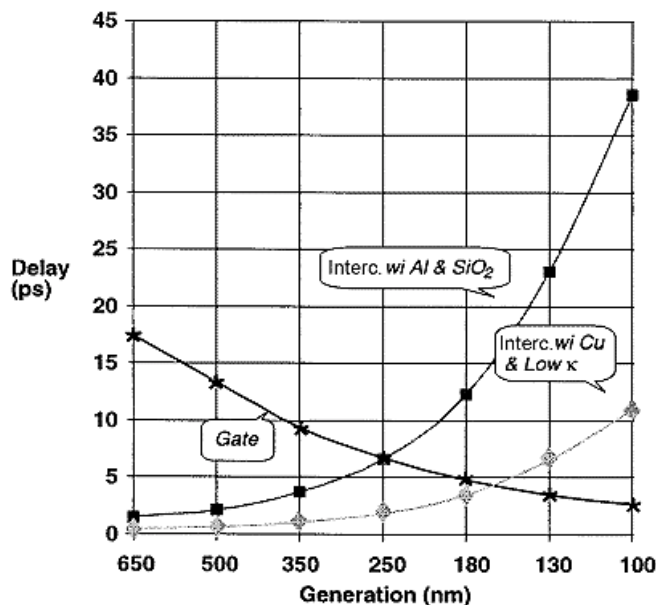


Figura 35 - Contribuição do atraso devido as interconexões.

Com isto, tem sido estudado a utilização de outros materiais para as interconexões que representem uma menor contribuição no atraso do circuito. Por exemplo, melhores resultados são obtidos com a utilização do cobre juntamente com um isolante de baixa constante dielétrica (Figura 35). Para tecnologias 0,25 $\mu\text{m}$  utilizando cobre, a contribuição das interconexões é de apenas 25% do atraso total, chegando a 45% nas tecnologias 0,18 $\mu\text{m}$ , e 55% nas tecnologias 0,15 $\mu\text{m}$ .

Além disto, a crescente diminuição na largura das interconexões fez com que a espessura se mantivesse constante ou fosse diminuída de um fator muito menor, para limitar a resistência. Então, a maior dimensão das conexões passa a ser a altura, como mostrado na Figura 36. Para tecnologias 0,25  $\mu\text{m}$ , a razão entre a altura e a largura é de 1,8, enquanto que espera-se que atinja 2,7 nas tecnologias 0,07  $\mu\text{m}$ .



Figura 36 - Vista de uma conexão submicrônica.

Juntamente com o aumento do número de níveis de interconexões (5-6), a capacitância de acoplamento entre conexões próximas tornou-se tão significativa quanto as capacitâncias ao plano terra (substrato). Isto provoca o efeito *crosstalk*, que é a interferência no nível de um sinal que está trafegando em um conexão devido a uma outra conexão vizinha, degradando a integridade dos

sinais e podendo levar ao mal funcionamento do circuito.

Com tudo isto, o problema de avaliação das capacitâncias de interconexões passa a ser uma função complexa com vários parâmetros do layout.

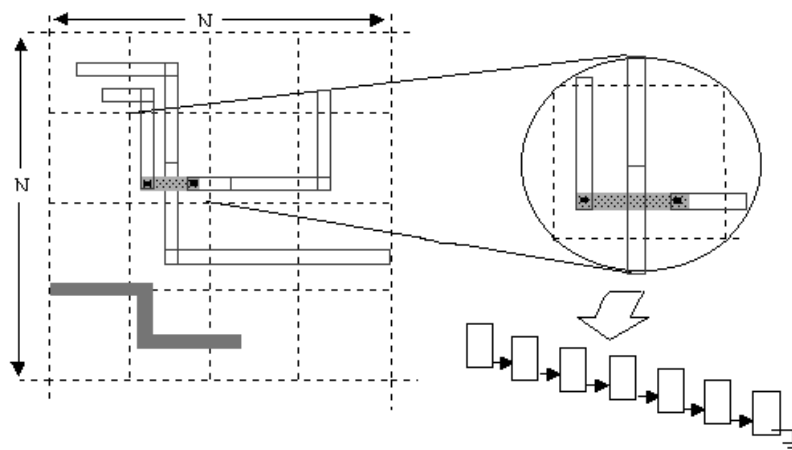
Logo, uma análise pós-layout é necessária para considerar os efeitos das interconexões, principalmente em processos submicrônicos. Nestes processos as limitações de desempenho, como atraso e integridade dos sinais, tem sido determinadas muito mais pelos efeitos das interconexões do que pelas características dos dispositivos ativos.

Com isto, foi integrado ao ambiente de síntese automática de layout TROPIC3 uma ferramenta chamada *LASCA*, que avalia as interações entre as diversas camadas de um circuito integrado, resultando na extração das capacitâncias de acoplamento, capacitâncias ao plano terra (substrato) e resistências.

## 5.1 Algoritmo de Extração de Conectividade

A extração das conectividades é uma das primeiras tarefas realizadas por um extrator elétrico. Ela tem a função de identificar as diferentes camadas dos polígonos e consequentemente, estabelecer a forma como os dispositivos estão interligados, atribuindo-lhes números de nodos para regiões equipotenciais.

A extração das conectividades realizada pelo extrator *LASCA* é feita por um algoritmo *bin-based*. Ele sobrepõe uma grade virtual sobre a área do layout, como mostrado na Figura 37. Esta grade divide a área total do circuito em uma série de quadrados, chamados "*bins*". A complexidade de espaço desta estrutura de dados é dada por  $O(b.P)$ , onde  $b$  é o número total de bins, e  $P$  é o número total de polígonos.



**Figura 37 - Algoritmo Bin-Based.**

Cada elemento desta matriz irá conter todos os polígonos que estão interseccionando este quadrado, sendo armazenados através de uma lista encadeada. Poderá ocorrer que um polígono esteja contido em mais de um quadrado, e consequentemente estar incluído em cada lista encadeada destes quadrados. Pode parecer um grande desperdício de memória, mas somente ponteiros para os polígonos são armazenados. Isto também evita redundância de informações, já que cada polígono deverá ter informações da sua posição e do nível a que pertence. Na Figura 37 temos o *bin* em destaque sendo interseccionado por sete polígonos que representam vias, metal1 e metal2.

O algoritmo *bin-based* apresenta um melhor desempenho do que uma estrutura de listas

encadeadas, desde que cada *bin* possua um pequeno número de polígonos, e consequentemente, os procedimentos de pesquisa serão realizados em listas curtas. O desempenho deste algoritmo é função do tamanho dos *bins*. Se os *bins* forem muito grandes, as listas encadeadas poderão conter centenas de polígonos, aumentando o tempo de CPU. Com o tamanho dos *bins* pequenos, haverá um aumento da memória utilizada, já que cada polígono interseccionará diversos *bins*.

Para uma tecnologia 0,25  $\mu\text{m}$ , foi fixado o tamanho dos *bins* em 2  $\mu\text{m}$  x 2  $\mu\text{m}$ . Desta forma, as listas encadeadas apresentaram, em média, até 20 polígonos. Na Figura 38 temos o histograma de distribuição dos polígonos para o circuito C7552 (benchmark ISCAS). Este circuito apresenta 265563 polígonos. Analisando-se o tamanho das lista encadeadas, temos que 86,8% dos *bins* apresentaram até 16 polígonos. Outra análise realizada, foi referente ao número de *bins* ocupados por um polígono, onde 97% dos polígonos ocupam de 1-5 *bins*. Com isto, para encontrar todos os polígonos conectados a um dado polígono, devemos pesquisar 80 polígonos (5x16), ao invés de pesquisar todos os polígonos. Observe que este é o caso típico.

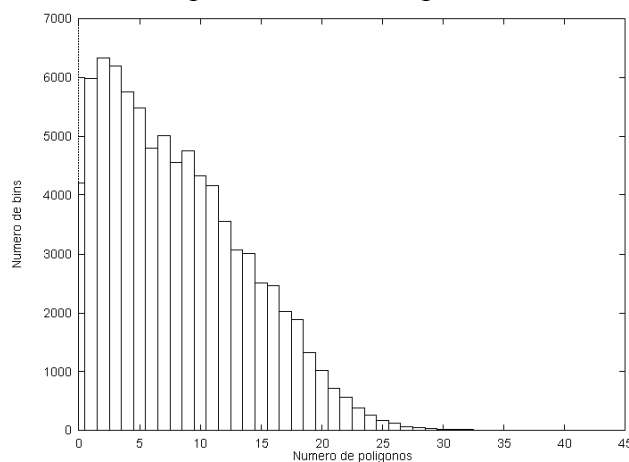


Figura 38 - Histograma da distribuição dos polígonos.

Outra vantagem deste algoritmo é o reduzido tempo de CPU para a extração da conectividade de circuitos complexo. A Figura 39 relaciona o tempo de CPU para a extração dos elementos parasitas do circuito *versus* o número de polígonos do circuito.

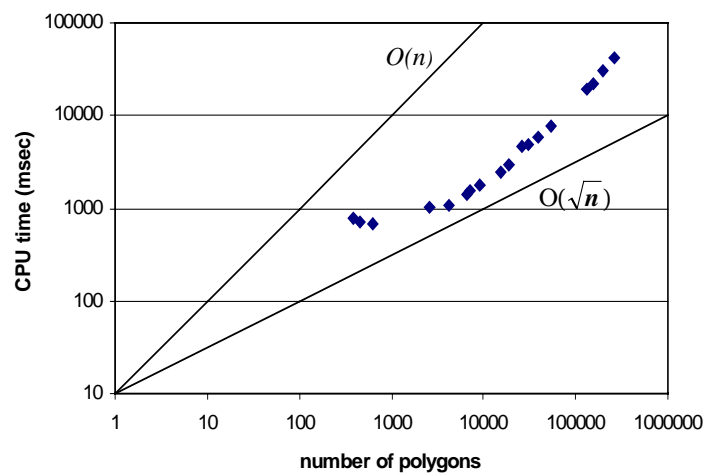


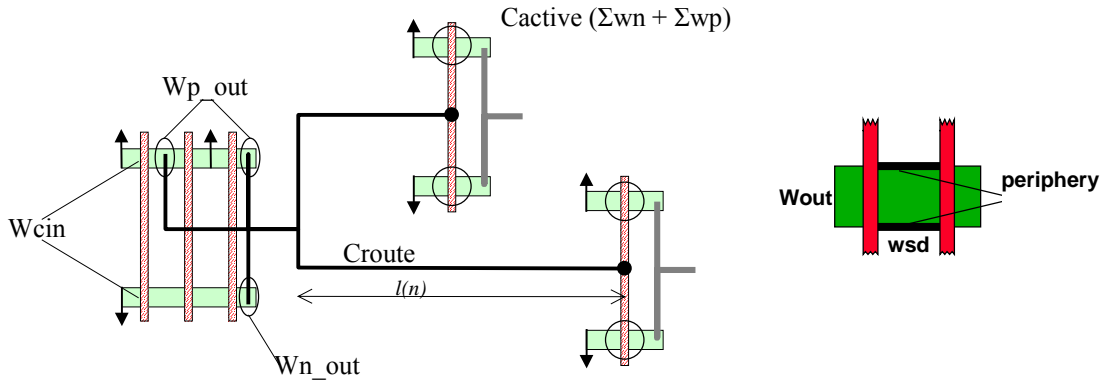
Figura 39 – Tempo para realizar a extração *versus* o número de polígonos no arquivo de layout.



## 5.2 Modelo de Extração

A capacitância parasita de cada rede,  $C_{load}$  é dada pela fórmula abaixo. Estes três elementos são mostrados na Figura 40.

$$C_{load} = C_{diff} + C_{active} + C_{route} \quad (1)$$



**Figura 40 – Componentes que determinam a capacitância parasita de uma determinada rede.**

A capacitância de difusão é dada pela equação abaixo:

$$C_{diffusion} = \frac{(W_{nout} \cdot C_{jn} + 2 \cdot C_{jnsw}) \cdot wsd \cdot drainN\# + (W_{pout} \cdot C_{jp} + 2 \cdot C_{jp\sw}) \cdot wsd \cdot drainP\# + W_{nout} \cdot C_{jnsw} + W_{pout} \cdot C_{jp\sw}}{FDIF}$$

onde:

- $drainN\#$  e  $drainP\#$  correspondem ao número de regiões de dreno/source que devem ser consideradas para o cálculo a capacitância parasita;
- $wsd$  corresponde à largura média das áreas de dreno/source no layout;
- $C_{jn}$ ,  $C_{jp}$ ,  $C_{jnsw}$ ,  $C_{jp\sw}$  correspondem as capacitâncias de difusão, fornecidas pelo arquivo de tecnologia;
- $FDIF$  é um constante que divide o valor total da capacitância de difusão, valor fornecido no arquivo de regras.

A parcela de capacitância ativa é função do *fanout* da rede, ou seja, do número de gates conectados na rede. Esta capacitância é dada por:

$$C_{active} = (\sum W_n + \sum W_p) \cdot l_{min} \cdot Cox$$

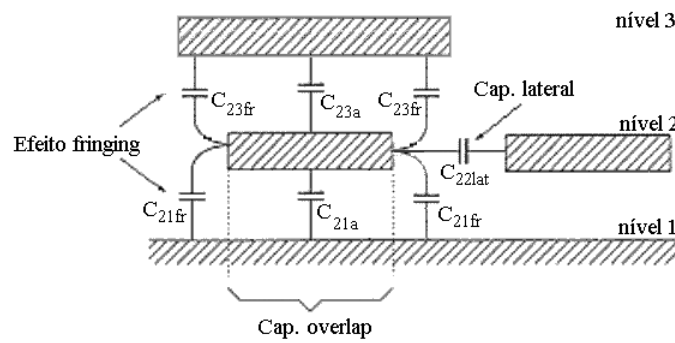
Como as capacitâncias de difusão e de área ativa dependem do layout gerado, ela são calculadas pela ferramenta de síntese de layout e fornecidas ao extrator de conectividade.

### Modelo para capacitância de roteamento

Algumas simplificações foram tomadas para a avaliação desta capacitância, tendo em vista que o tratamento de estruturas multiníveis é uma tarefa complexa e difícil. Para cada conexão em análise num plano  $i$ , os planos  $i \pm 2$  foram tomados como terra, e somente foram calculadas as capacitâncias entre conexões vizinhas ou conexões nos planos  $i \pm 1$ .

As capacitâncias de interconexão de cada nodo do circuito foram calculadas usando o modelo da Figura 41. Ele consiste de dois níveis condutores sobre um plano (substrato), considerado como terra.





**Figura 41 - Modelo de capacitâncias.**

Três componentes foram considerados:

- Capacitância de Overlap ( $C_{over}$ ) – é devido a sobreposição de dois condutores em planos diferentes. Eles são  $C_{21a}$  e  $C_{23a}$  na Figura 41. São calculados considerando a área de sobreposição, dada pela seguinte fórmula:

$$C_{over} = C_{area} \cdot W \cdot L$$

Onde  $C_{area}$  é a capacitância de área (fF/ $\mu m^2$ ),  $W \cdot L$  é a área de sobreposição

- Capacitância lateral ( $C_{lat}$ ) – é a capacitância entre dois condutores no mesmo plano. Na Figura 41 é a parcela  $C_{22lat}$ . É calculada utilizando formulação empírica, onde deve-se considerar a distância entre os dois condutores.
- Capacitância Fringing ( $C_{fr}$ ) – é devido ao acoplamento entre dois condutores de planos diferentes. São representados por  $C_{21fr}$  e  $C_{23fr}$  na Figura 41. É calculada pela fórmula abaixo, sendo  $C_{length}$  é a capacitância de perímetro (fF/ $\mu m$ ), e  $L$  é o perímetro da conexão.

$$C_{fr} = 2 \cdot C_{length} \cdot L$$

### **Modelo para resistência de roteamento**

O cálculo das resistências foram realizados utilizando a seguinte fórmula:

$$R = R_{\square} \frac{L}{W}$$

Onde  $R_{\square}$  é a resistividade do material, dada em  $\Omega/\square$ ,  $L$  é o comprimento da conexão e  $W$  é a largura. Para obter a resistência de uma conexão, simplesmente multiplicamos a resistividade do material pelo comprimento da conexão, divida pela largura. Uma rede completa, geralmente tem vários polígonos, com segmentos verticais e horizontais. A resistência total da rede é obtida somando-se as resistências de todos os polígonos que compõem a rede.

Após a extração das capacitâncias e resistências, devemos representar a conexão de forma apropriada. No extrator LASCA, três opções estão disponíveis ao usuário, que deverá considerar a precisão requerida numa análise do funcionamento elétrico do circuito, tempo de simulação e a complexidade do circuito extraído. Os modelos disponíveis são:  $L$ ,  $\pi$  e  $T$  *lumped*, mostrados na Figura 42. Na Figura 43 temos um exemplo de um cruzamento de duas conexões e o seu respectivo modelo de extração. Note-se o elevado número de componentes utilizados para representar as duas conexões.

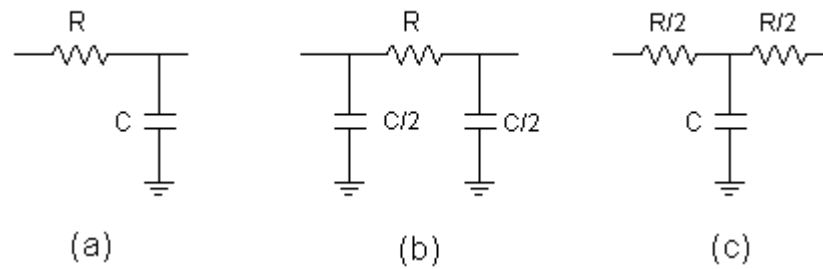


Figura 42 - Modelos RC (a) L lumped (b)  $\pi$  lumped (c) T lumped.

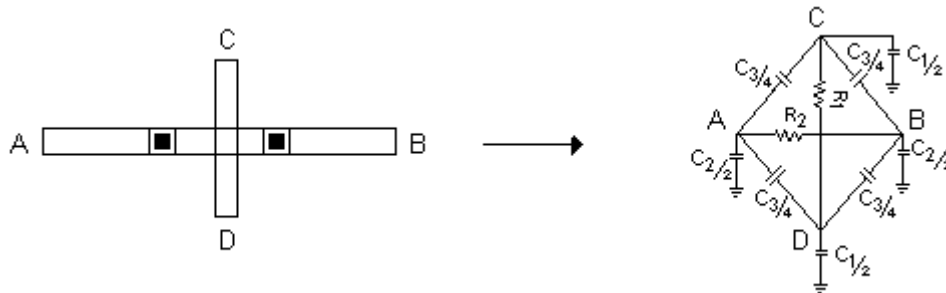


Figura 43 - Cruzamento de duas conexões e a respectiva extração.

### Análise do desempenho da extração de capacitâncias parasitas

A Tabela 6 mostra resultados de simulação elétrica. Se extrairmos apenas as capacitâncias relativas ao substrato, teremos uma avaliação otimista do atraso do circuito, com um erro médio de 8,25 % (pior caso: 15%). Quando consideremos todas as capacitâncias de roteamento o erro médio passa a ser 3,63%, com pior caso inferior a 10%. Estes dados validam o extrator de parasitas *LASCA*, o qual é muito mais rápido que as ferramentas de extração convencionais (exemplo para um circuito com 265563 polígonos: a ferramenta *lasca* consumiu 42,5 segundos, enquanto que o extrator *DIVA* 19,2 minutos).

Circuito	Transistores	Redes	Número de polígonos	Atraso do caminho crítico (ns)		
				$C_{ground}$	$C_{ground} + C_{coupl}$	Diva
Adder	28	13	449	0.3624	0.3847	0.36783
Addergate	40	15	635	0.4042	0.4304	0.4247
Alu	260	94	4291	0.985	1.1167	1.1056
Alugate	432	117	6503	1.0984	1.2401	1.2208
Rip 16bits	448	163	7270	3.3713	3.7872	3.8571
Cla 16bits	528	215	9066	2.5539	2.8737	2.6234
Rip 32 bits	896	323	15503	7.0158	8.2248	8.2730
Cla 32bits	1056	427	19051	5.3084	6.2971	5.7966

Tecnologia: 0.25 $\mu$ m, 3 níveis de metal com contatos empilhados. Dimensão dos transistores  $w=2\mu$ m,  $l=0.25\mu$ m.

**Tabela 6 – Atraso obtido por simulação elétrica, considerando: (1) apenas as capacitâncias relativas ao substrato, extraídas com a ferramenta *LASCA*; (2) capacitâncias de acoplamento mais as relativas ao substrato, extraídas com a ferramenta *LASCA*; (3) capacitâncias obtidas com o extrator de referência, *Diva*, do conjunto de ferramentas *CADENCE*.**

## 5.3 Referências bibliográficas

- [ARO96] N.D. ARORA, K.V. RAOL, R. SCHUMANN AND L.M. RICHARDSON.  
*Modeling and Extraction of Interconnect Capacitances for Multilayer VLSI Circuits.*  
IEEE Transactions on CAD, 15(1):58-66, Jan. 1996.
- [CAD98] Diva Interactive Verification Reference  
*CADENCE™. 1998*
- [CHE92] J.CHERN, J.HUANG, L.ARLIDGE, P.LI, P.YANG.  
*Multilevel Metal Capacitances Models for CAD Design Synthesis Systems.*  
IEEE Electron Devices Letters, v.13, n.1, p.32-34, Feb. 1992.
- [CON96] J. Cong, A.B. Kahng, D. Noice, N. Shiralli and S.H. Yen.  
*Analysis and Justification of a Simple, Practical 2 1/2D Capacitance Extraction Methodology.*  
UCLA Computer Science Technical Report 970013, 1996.
- [FER00] **Fábio Klein Ferreira**  
*Extração de Elementos Parasitas em Circuitos CMOS Submicronicos*  
**Dissertação de Mestrado, PPGC-UFRGS, 2000**  
<http://www.inf.pucrs.br/~moraes/FMpapers.html>
- [NAB91] K. NABORS and J. WHITE.  
*FastCap: A Multipole Accelerated 3D Capacitance Extraction Program.*  
IEEE Transactions on CAD, 10(11):1447-1459, Nov. 1991.
- [SAK92] T. SAKURAI.  
*Approximation of wiring delay in Mosfet LSI.*  
IEEE Journal Electron Devices Letter, 13(1):32-34, Feb. 1992.

## 6 INTEGRAÇÃO DA SÍNTESE FÍSICA À SÍNTESE LÓGICA

Este Capítulo apresenta o procedimento para integrar a síntese automática de layout a um fluxo de projeto com entrada VHDL, permitindo o uso de bibliotecas virtuais. Como mencionado, bibliotecas virtuais baseiam-se no uso de células geradas automaticamente por uma ferramenta de síntese de layout, ao invés da utilização de bibliotecas com células pré-caracterizadas. Desta forma, pode haver na biblioteca virtual um conjunto muito rico de portas complexas.

Nossa primeira abordagem para integrar síntese de layout à síntese lógica foi baseada em utilizar ferramentas dedicadas para mapeamento tecnológico, tais como TABA (UFRGS-LIRMM) ou SYNTHETIC (UAB). O procedimento consiste em gerar uma descrição contendo apenas portas básicas (*and*, *or*, *inv* em formato *netblif*), e depois utilizando estas ferramentas obtêm-se o circuito com portas complexas. Esta abordagem foi abandonada por três razões muito fortes:

1. Ao especificar-se restrições temporal e/ou de área para a ferramenta de síntese lógica, ela deve gerar uma solução que atenda às restrições utilizando as portas lógicas que serão utilizadas na síntese física. Se a ferramenta de síntese lógica gera uma solução com portas básicas, o atraso/área será calculado sobre estas portas básicas. Logicamente, após o mapeamento com portas complexas estas estimativas perderam todo o sentido.
2. As ferramentas para mapeamento com portas complexas não consideram lógica síncrona, o que impede de gerar circuitos reais.
3. As ferramentas de síntese lógica, como Synopsys, podem tratar a especificação de uma biblioteca virtual contendo centenas de portas complexas, modeladas através de equações de atraso, e assim realizar a síntese lógica e o mapeamento tecnológico como uma única operação.

Substituiu-se esta primeira abordagem utilizando-se uma ferramenta que realize ambas tarefas, síntese lógica e mapeamento tecnológico, conjuntamente. A Figura 44 mostra o fluxo de projeto utilizando esta abordagem. Caso o circuito seja especificado de forma comportamental, uma ferramenta de síntese lógica, por exemplo Synopsys, deve ser utilizada para gerar uma descrição estrutural, mapeada sobre a biblioteca virtual. Quando o projetista define seu circuito desta forma, ele não precisa *a priori* ter nenhum conhecimento da biblioteca virtual que está sendo utilizada. Caso o projetista deseje especificar o seu circuito utilizando níveis mais baixos de abstração, ele pode descrever seu circuito diretamente em VHDL estrutural, mas neste caso ele deve ter conhecimento das portas existentes na biblioteca, pois ele as instanciará.

Uma vez obtida a descrição estrutural, ela é automaticamente convertida em formato SPICE, pois para a síntese automática de layout necessita-se uma descrição a nível de transistores. A biblioteca virtual é modelada para um dimensionamento regular de transistores, isto significa, todos os transistores têm a mesma largura. Esta dimensão regular é modificada após a otimização elétrica, caso necessária, em função das capacitâncias parasitas inseridas pelo roteamento.

Uma primeira síntese de layout é feita, objetivando-se obter as capacitâncias parasitas. A descrição SPICE original, em conjunto com as capacitâncias calculadas, são utilizadas pelas ferramentas de análise de desempenho elétrico (*timing analysis*) e de otimização elétrica. Caso o desempenho elétrico obtido não atenda às especificações impostas no início da síntese, algumas estratégias de otimização podem ser empregadas: dimensionamento de transistores, inserção de *buffers* ou estágios repetidores.

Caso esta otimização elétrica seja necessária, é feita uma segunda síntese de layout, mantendo-se o posicionamento da primeira. Deve-se manter o mesmo posicionamento entre as iterações de síntese de layout, pois as otimizações elétricas são feitas considerando-se o posicionamento/roteamento feitos.

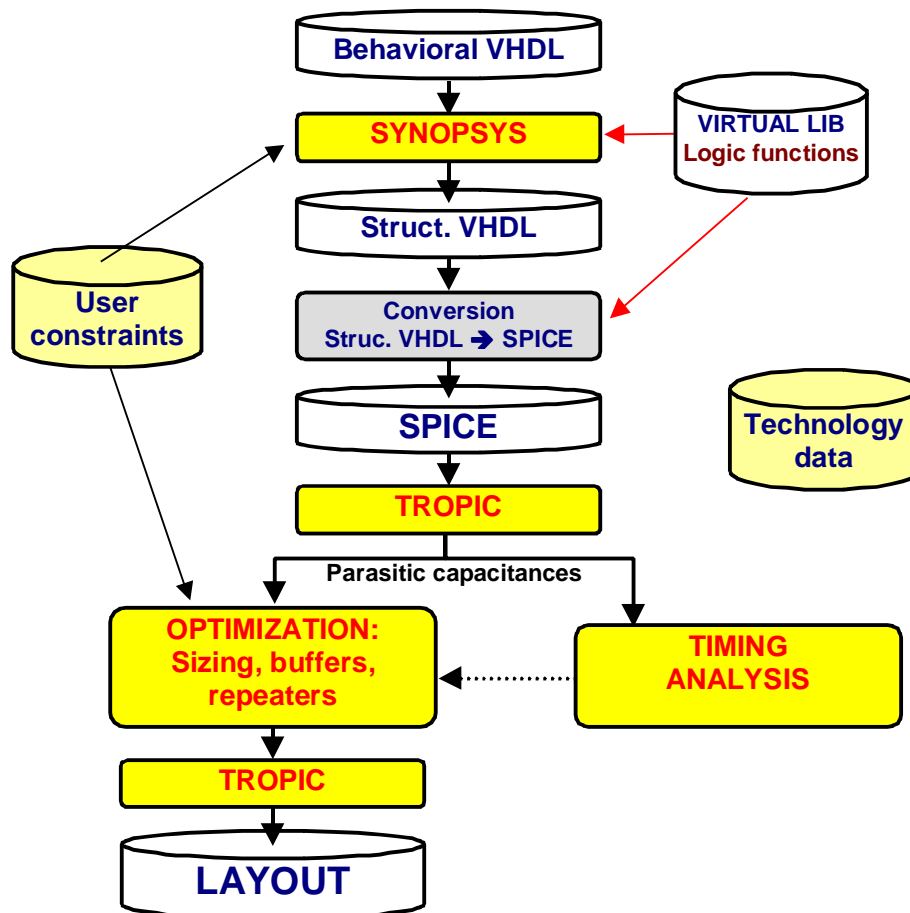


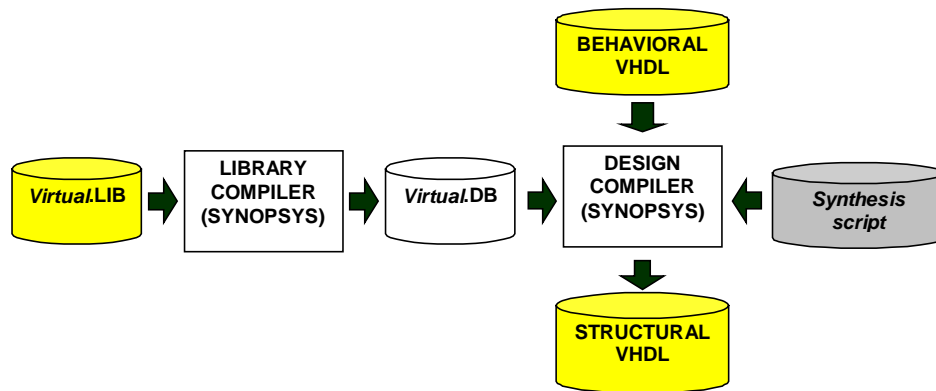
Figura 44 - Fluxo de projeto utilizando biblioteca virtual e síntese automática de layout.

Até a data de hoje, agosto de 2000, a etapa de otimização elétrica não está implementada no fluxo de projeto de TROPIC3, sendo alvo de investigações futuras. Um protótipo do sistema está disponível à comunidade científica em:

<http://www.inf.pucrs.br/~moraes/tropic2000.tar.gz>

## 6.1 Abordagem de biblioteca virtual no ambiente Virtual Synopsys

A Figura 45 mostra a abordagem de biblioteca virtual no ambiente Synopsys de síntese lógica. Três arquivos alimentam a ferramenta de síntese lógica "*design compiler*": o arquivo contendo a descrição da biblioteca virtual, o arquivo contendo a descrição VHDL (ou Verilog) do circuito de entrada e o *script* de síntese contendo os parâmetros de síntese.



**Figura 45 - Abordagem de biblioteca virtual no ambiente Synopsys de síntese lógica.**

### ***Biblioteca Virtual***

O primeiro passo no fluxo de projeto é descrever a biblioteca virtual, no formato da ferramenta "*library compiler*". Esta biblioteca após compilada gera um arquivo binário, aceito pela ferramenta de síntese lógica. Para exemplificar o processo, temos na Figura 46 a descrição de uma célula complexa. A sintaxe para descrever a funcionalidade de uma célula é muito simples, bastando indicar sua função booleana, através do comando *function*. A descrição de *timing* porém é muito complexa, havendo diversos modelos possíveis para a descrição do atraso da célula. O modelo apresentado na Figura considera tempo constante para subida e descida, assim como a mesma capacitância para todas as entradas. Outros modelos, como os baseados em tabelas (*look-up tables*) estão hoje sendo investigados.

```

cell(AOI1125) {
  area : 18;
  pin(z) {
    direction : output;
    function : "!(A*(B+(C*(D+E*(F+G*(H+I))))))";
    max_capacitance : 0.250000;
    timing() {
      intrinsic_rise : 0.25 ;
      intrinsic_fall : 0.30 ;
      rise_resistance : 0.15 ;
      fall_resistance : 0.10 ;
      related_pin : "A B C D E F G H I" ;
    }
  }
  pin(A B C D E F G H I) {
    direction : input;
    capacitance : 0.250000;
  }
}
  
```

**Figura 46 - Descrição de uma célula complexa no ambiente de síntese Synopsys (a declaração de *timing* está simplificada)**

Não há restrições quanto ao número de células complexas que podem ser descritas na biblioteca virtual. Deve-se entretanto observar a restrição do número de transistores em série, aconselhando-se utilizar 4 no plano N e 3 no plano P (396 funções diferentes, conforme a Tabela 1).

Os elementos seqüenciais (flip-flops, latches) e tri-states estão presentes também na biblioteca virtual. Estes elementos não podem ser automaticamente inseridos na biblioteca, devido à complexidade de sua descrição. Existe um arquivo padrão, que é configurado para cada tecnologia, através de simulações elétricas, de acordo com as regras de projeto. Há 9 elementos nesta

biblioteca: 4 FFDs (somente D, set, reset, set/reset), 4 LATCHs (somente D, set, reset, set/reset) e 1 buffer tri-state. A documentação Synopsys indica como conjunto mínimo 3 elementos: FFD set/reset, LATCH D set/reset e buffer tri-state

### Descrição VHDL circuito de entrada

A descrição VHDL de entrada pode ser comportamental, estrutural ou um misto de ambas. A Figura 47 mostra uma descrição VHDL parcial no nível comportamental. Neste tipo de descrição o projetista não necessita saber quais células estão presentes na biblioteca virtual. A descrição é feita utilizando-se comandos como atribuições concorrentes (equivalentes a multiplexadores) ou processos (equivalentes a registradores). Descrições neste nível apresentam a vantagem de serem portáteis, permitindo uma fácil migração de uma biblioteca virtual a outra.

```
...
input_adder2 <= (not busB) when uins.ula=AsubB else
                (others=>'0') when uins.ula=negA or uins.ula=incB else
                (others=>'1') when uins.ula=decA else
                busB;
...
process(ck,reset,uins)
begin
if (reset = '1') then
    c <= '0';      n <= '0';  z <= '0';
elsif ck'event and ck = '0' then
    if uins.c = '1' then c <= cout; end if;
    if uins.nz = '1' then
        n <= out_ula(15);
        if out_ula="0000000000000000"
            then z <= '1';
            else z <= '0';
        end if;
    end if;
end if;
end process;
...
```

**Figura 47 - Exemplo parcial de descrição VHDL comportamental.**

Caso uma descrição VHDL estrutural ou mista é utilizada, como a apresentada na Figura 48, o usuário deve conhecer o nome das células na biblioteca virtual.

```
library IEEE;
use IEEE.Std_Logic_1164.all;

entity my_circuit is
    port( pg: out reg2; PGij, PGjk, PGkl : in reg2 );
end my_circuit;

architecture A1 of my_circuit is
    signal s1, s2 : std_logic;
    begin
        U0 : nor3 port map ( PGij(0), PGjk(0), PGkl(0), s1);
        U1 : inv port map ( s1, pg(0));
        U3 : AOI_BK3 port map (PGij(1), PGij(0), PGjk(1), PGjk(0), PGkl(1), s2);
        U4 : inv port map ( s2, pg(1));
    end A1;
```

**Figura 48 - Exemplo parcial de descrição VHDL estrutural.**

Neste exemplo, as células *nor3*, *inv* e *AOI\_BK3* devem estar presentes na biblioteca, pois caso não estejam, a síntese lógica e o mapeamento tecnológico não serão feitos. Este tipo de descrição não é recomendado, pois não é portátil entre uma biblioteca e outra. Seu uso é recomendado



apenas quando o projetista deseje utilizar especificamente uma dada porta complexa. Por exemplo, suponha que o projetista esteja implementando uma arquitetura de soma muito rápida, e na sua cadeia de propagação de *carry* ele necessite um porta complexa específica. Neste caso ele referencia a porta complexa da biblioteca com a função desejada.

### Script de síntese

O terceiro arquivo que alimenta a ferramenta de síntese lógica é o *script* de síntese. Este *script* guia a ferramenta de síntese lógica, indicando qual biblioteca utilizar, esforço de síntese, qual a máxima área que o circuito deve ter, qual o atraso, qual o fanout máximo, etc. O conjunto de parâmetros da ferramenta é muito rico em opções, permitindo explorar um vasto espaço de soluções. A Figura 49 apresenta um *script* de síntese contendo restrições de área (*set\_max\_area*), de atraso (*set\_max\_delay*) e de fanout (*set\_max\_fanout*). É muito importante que o VHDL resultante da síntese não possua hierarquia, pois o atual tradutor de VHDL para SPICE trata apenas descrições planas. Para isto, no *script* são inseridos os comandos *uniquify* e *ungroup* que realizam a tarefa de "planificar" o arquivo de saída.

```
link_library = {sccg33.db libdff.db}           // Bibliotecas de portas complexas e de elementos
target_library = {sccg33.db libdff.db} '       // sequenciais

cell_list = { sklansky }

foreach (cell, cell_list) {

    read -f vhdl cell + ".vhdl"
    current_design cell

    set_input_delay 0.0 all_inputs()
    set_max_delay 3.5 to all_outputs()
    out_load = 0.25
    set_load out_load all_outputs()
    set_max_area 300
    set_max_fanout 6 cell

    compile -map_effort medium

    /* Removes multiply-instantiated hierarchy */
    uniquify -force
    ungroup -flatten -all

    vhdlout_write_components = TRUE
    vhdlout_single_bit = "TRUE"
    vhdlout_dont_create_dummy_nets = "TRUE"

    write -format vhdl -hierarchy -output cell + "_S.vhdl"
}
quit
```

**Figura 49 - Exemplo de *script* de síntese, onde área e atraso são especificados.**

### Resultado da síntese

O arquivo resultante da síntese lógica é um arquivo VHDL estrutural, mapeado sobre a biblioteca virtual, o qual é automaticamente convertido em descrição SPICE.

A descrição a nível de transistor para cada célula é obtida através de sua equação booleana (comando *function* na Figura 46). A Figura 50 mostra a equação de uma dada célula, e o respectivo sub-circuito em formato SPICE. A ferramenta que converte VHDL estrutural para SPICE utiliza o mesmo arquivo de descrição utilizado pelo compilador de biblioteca (*library compiler*).

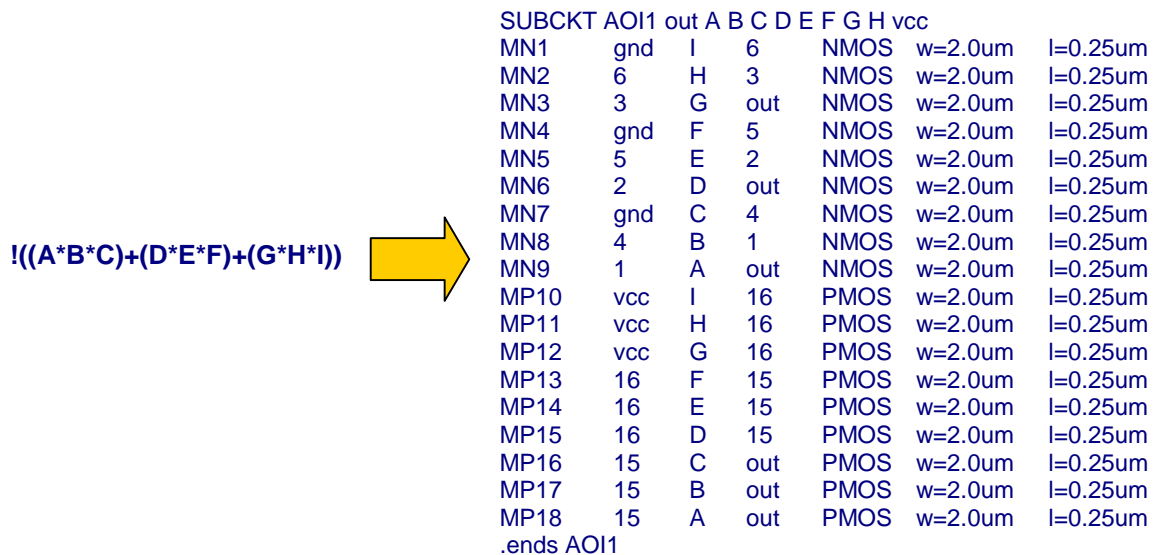


Figura 50 - Equação booleana de uma porta complexa e sua descrição em formato SPICE.

Cada célula (*component*, em VHDL) utilizado no VHDL estrutural é convertido em um sub-circuito SPICE, a partir do comando *function* na biblioteca virtual. Esta tradução é feita por uma rotina recursiva, que considera a operação ‘\*’ (and) como transistores conectados em série e a operação ‘+’ (or) como transistores conectados em paralelo, para o plano N (o plano P é dual ao plano N). Observar que todas as funções na biblioteca virtual são negativas. Os elementos sequenciais e o buffer tri-state são gerados à partir de um arquivo modelo (assim como para a biblioteca virtual). Todas as referências às células (*port map* em VHDL) são traduzidas como chamadas a sub-circuitos. Finalmente, os pinos de entrada/saída do circuito são obtidos através das portas declaradas na *entity* (comando que inicia um novo circuito em VHDL). O arquivo SPICE resultante é utilizado como descrição de entrada para a ferramenta de síntese física. Exemplo de descrição VHDL de origem e descrição de SPICE final é mostrada na Figura 51.

```

Entity div4 is
  Port(
    clk,rst: in std_logic;
    div, div4, div8: out std_logic );
end div4;
Architecture arch of div4 is
  component DIV2
    Port( clk,rst: in std_logic; d2_clk: out std_logic );
  end component;
  signal carry : std_logic_vector(2 downto 0);
begin
  st0 : DIV2 Port Map ( clk => clk, rst=> rst, d2_clk => carry(0) );
  st1 : DIV2 Port Map ( clk => carry(0),rst=> rst, d2_clk => carry(1) );
  st2 : DIV2 Port Map ( clk => carry(1),rst=> rst, d2_clk => carry(2) );
  div <= carry(0);
  div4 <= carry(1);
  div8 <= carry(2);
end;

Entity DIV2 is
  Port(clk,rst: in std_logic; d2_clk: out std_logic );
end DIV2;
Architecture arch of DIV2 is
  signal temp : std_logic;
begin
  d2_clk <= temp;
  process (clk,rst)
  begin
    if (rst='0') then
      temp <= '0';
    elsif (clk'event and clk='1') then
      temp <= not temp;
    end if;
  end process;
end;

```

```

**
** SUBCIRCUITS (generated from the lib file)
**
.SUBCKT DFFR D rst clk QN Q vcc
MN1    H clk gnd gnd NMOS  l=0.25u  w=2u
MP2    H clk vcc vcc PMOS  l=0.25u  w=2u
MN3    NH H gnd gnd NMOS  l=0.25u  w=2u
MP4    NH H vcc vcc PMOS  l=0.25u  w=2u
...
.ends DFFR

.SUBCKT GAT1_1 out A vcc
MN1    gnd A out gnd NMOS  l=0.25u  w=2u
MP2    vcc A out vcc PMOS  l=0.25u  w=2u
.ends GAT1_1

**
** COMPONENTS (generated from the VHDL file)
**
X1 net21 n3 clk net21 div vcc DFFR
X2 net20 n3 n1 net20 div8 vcc DFFR
X3 net19 n3 n2 net19 div4 vcc DFFR
X4 n3 rst vcc GAT1_1
X5 n2 net21 vcc GAT1_1
X6 n1 net19 vcc GAT1_1

**
** INPUTS AND OUTPUTS
** (generated from the ENTITY in the VHDL file)
**
*interface: div8 orient n output
*interface: div4 orient n output
*interface: div orient n output
*interface: rst orient s input
*interface: clk orient s input

```

(a) descrição VHDL de origem

(b) descrição SPICE final obtida da descrição SPICE

Figura 51 - Exemplo de descrição VHDL de origem e de descrição SPICE final.

## 6.2 Referências bibliográficas

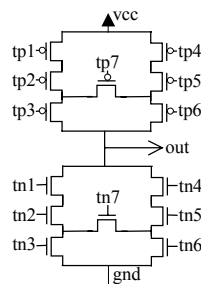
- [BEN91] BENKOSKI, J.; STEWART, R. TATOO:  
*An industrial Timing Analyzer with False Path Elimination and Test Pattern Generation.*  
EDAC, 1991, p.256-260.
- [CHA93] CHANG, Hoon, ABRAHAM, Jacob A., CHAN:  
*An Efficient Critical Path Analysis Algorithm.*  
EDAC, 1993, pp.444-448.
- [CRE98] S.CREMOUX; N.AZEMARD; D.AUVERGNE.  
*Path resizing based on incremental technique.*  
ISCAS98.
- [DAG99] J.DAGA, D.AUVERGNE.  
*A Comprehensive Delay Macro-Model of Submicrometer CMOS Logics.*  
IEEE Journal of Solid States Circuits, vol 34, n°1, pp.42-55, January 1999.
- [FAN95] FANG, C.-L.; JONE, W.-B.  
*Timing Optimization by Gate Resizing and Critical Path Identification.*  
IEEE Transactions on CAD, v.14, n.2, p.201-217., February 1995.
- [JOU87] JOUPPI, Norman P.  
*Timing Analysis and Performance Improvement of MOS VLSI Designs.*  
IEEE Transactions on CAD, v.CAD-6, n.4, pp. 650-665, July 1987.
- [OBE88] OBERMEIER, F.W.; KATZ, R.  
*An Electrical Optimizer that Consider Physical Layout.*  
17th DAC, 1988. p. 453-459.
- [REI95] A.REIS, M.ROBERT, D.AUVERGNE, R.REIS.  
*Associating CMOS Transistors with BDD Arcs for Technology Mapping.*  
Electronic Letters, Vol. 31, No 14, July 1995.
- [SAI95] SAIT,S.; YOUSSEF,H.  
*VLSI Physical Design Automation - Theory and Practice.*  
IEEE Press, 1995, 436p.
- [SIA97] SEMICONDUCTOR INDUSTRY ASSOCIATION.  
*The National Technology Roadmap for Semiconductor.*  
Available by WWW in <http://notes.sematech.org/ntrs/PublNTRS.nsf>, 1997
- [SYN98] SYNOPSYS.  
*Library Compiler User Guide, Volume 1.*  
Synopsys Documentation, v1998.08.

## 7 EXERCÍCIOS

Supõe-se para estes exercícios a instalação do TROPIC3. O nome dos circuitos nas questões refere-se aos circuitos no diretório “tropicD/benchs”. Aconselha-se a ler modo de utilização do programa e como instalá-lo lendo o arquivo “*README\_TROPIC2000*”, no diretório “tropicD”. Buscar em: <http://www.inf.pucrs.br/~moraes/tropic2000.tar.gz>

Para visualizar o layout pode-se utilizar o editor de layout *edllex* com o arquivo de configuração *ed.cfg*, ambos presentes no diretório *tropicD*. Atenção: deve-se remover os *labels* do arquivo de layout, pois o *edllex* não aceita texto com caracteres especiais, para isto usar o programa *label*, também disponível na instalação.

1. Faça o diagrama de transistores para a função  $F = \text{not}((A \text{ and } B) \text{ or } C)$ , descreva esta função no formato *SPICE*, colocando as entradas no lado superior (interface norte) e a saída no lado direito (interface leste). Utilize as regras  $0.25 \mu\text{m}$  (parâmetro  $-t25$ ), com dimensionamento  $W=2\mu\text{m}$  e  $L=0.25\mu\text{m}$ . Sintetize este circuito com uma banda e observe o layout. Altere a dimensão dos transistores, gerando o layout novamente.
2. Agora, para o layout acima gerado modifique alguns parâmetros do arquivo de tecnologia e observe o impacto no layout. Sugestão, altere a largura do polisilício, largura do metal, regras relacionadas aos contatos. Altere uma regra por vez para analisar o impacto no layout.
3. É possível o circuito da questão 1 ( $F = \text{not}((A \text{ and } B) \text{ or } C)$ ) ser gerado em mais de uma banda? Por quê?
4. O diagrama de transistores abaixo é sintetizável pela ferramenta TROPIC3? Argumente a resposta.



5. Determine uma função lógica que não tenha caminho de Euler no plano N, e outra função lógica sem caminho em ambos planos.
6. Implemente um circuito com portas de transmissão, por exemplo, uma *latch*, e identifique no layout o posicionamento destas portas de transmissão.
7. A ferramenta LASCA (ou *ext* na instalação) permite exibir graficamente a distribuição do comprimento das conexões. Sintetize um grande circuito (com mais de 10000 transistores, use a ferramenta *extralo* para localizá-lo), e com a ferramenta LASCA exiba o histograma de distribuição do comprimento das conexões. Qual a porcentagem do número de redes com comprimento igual ou inferior a  $200 \mu\text{m}$ ? Qual a conclusão que pode-se tirar, e se ela é ou não positiva? O algoritmo de posicionamento de células pode ser considerado eficiente? Por quê?
8. Use a ferramenta EXTRALO (extração lógica) sobre o circuito “*adder.sim*”, e analise as funções lógicas obtidas. Exprima na forma de equação booleana as equações obtidas e verifique se correspondem à função de soma.

9. O arquivo “*rip.sim*” e “*cla.sim*” são somadores de 16 bits. Pede-se (o modelo do arquivo de simulação encontra-se em *tropicD/simul*, assim como o arquivo contendo o modelo dos transistores):
- a) Utilize as regras 0.25  $\mu\text{m}$ , com dimensionamento  $W=2\mu\text{m}$  e  $L=0,25\mu\text{m}$ , e sintetize ambos circuitos de tal forma que tenham um formato o mais próximo de um quadrado (altura igual a largura).
  - b) Anote em tabela o número de transistores, número de bandas, área e densidade de transistores por milímetro quadrado para cada um dos circuitos.
  - c) Extraia as capacitâncias parasitas de ambos circuitos, utilizando a ferramenta LASCA, utilizando os 3 modelos de capacitâncias. O resultado será um conjunto de 6 arquivos de capacitâncias parasistas.
  - d) Simule agora as 3 versões de cada somador, anotando o atraso do caminho crítico em uma tabela. Analise o desempenho elétrico de ambos somadores.
  - e) Qual dos dois somadores você escolheria, e por qual razão?
-