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

Geração de Circuitos Utilizando Matrizes de Células Pré-difundidas

por

José Luís Almada Güntzel

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

Prof. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, Maio de 1993.

AGRADECIMENTO

Agradeço a Ricardo Reis que, mais do que orientador, tornou-se um grande amigo, cujos conselhos, sempre otimistas, guiaram-me durante os muitos momentos de dúvida e angústia.

A Dante Barone, pela oportunidade de ter iniciado a trabalhar em Microeletrônica, descobrindo uma vocação (e por que não dizer uma paixão) até então adormecida.

A Tiaraju Wagner e Sergio Bampi, pela dedicação com a qual instruem as novas gerações de pesquisadores e profissionais da Microeletrônica e pela amizade alcançada ao longo dos vários anos de convívio.

A Luis Otavio Freire e Renato Ribas, com quem partilhei muitas e muitas horas de trabalho e aprendi o grande valor do trabalho em equipe.

Aos colegas André Reis e Luis Felipe Uebel, pelas inúmeras horas de estudo que dividimos, pelos inúmeros trabalhos que desenvolvemos juntos e pelos momentos de descontração que ajudaram a aliviar a pressão do cotidiano.

Aos meus auxiliares Anelise Hackbart, Marcus Kindel e Fernando Krüger, cuja dedicação, empenho e amizade foram decisivos para que os Projetos Marcela e Gama se tornassem realidade concreta.

A Marcelo Johann, que desenvolveu com brilhantismo a ferramenta de roteamento, tendo ainda contribuído com sugestões valiosas em muitas outras definições do Marcela.

Aos auxiliares Demétrio Freitas e Aline Flores, que tendo se juntando ao grupo no momento mais crítico, deram consideráveis contribuições.

A Daniel Fachin e Fábio Duarte, pelo projeto e testes do circuito TCHÊ,

respectivamente.

A César Crusius e André Hentz, por não medirem esforços em auxiliar usuários desesperados.

A Luigi Carro, a quem devo meus primeiros passos na Microeletrônica.

Aos colegas Marcos Dossa, Laerte Cleto, Luiz Claudio Santos e Luiz Fernando Ferreira, pelas infinitas dicas, trocas de idéias e amizade.

A Fernando Moraes, cujo apoio foi fundamental para que eu me integrasse ao Projeto TRANCA.

Aos colegas de mestrado Antônio Frainer, Yara Lemr, Sandra Freitas e Roberto Amboni, pela amizade vivenciada desde o nivelamento ou mesmo antes.

Aos colegas Javier Aprea, César Marcon, Gilberto Marchioro, Francisco Nascimento, Paulo Schermer, Carlos Eduardo Souza, Fernando Soto, Reginaldo Tavares, Gilson Wirth, André Aita e Alexandre Casacurta, pela amizade e companheirismo.

Aos amigos Alberto Dri e François Grimbert, vivendo no Velho Mundo, cuja convivência deixou saudades.

Agradeço também aos funcionários da secretaria do Curso de Pós-Graduação em Ciência da Computação, Joice, Elisiane, Claudia, Citamara e Júnior, às funcionárias da secretaria do Instituto de Informática, Maria do Carmo, Vera e Silvania e aos funcionários dos laboratórios, Luis Otavio, Camillo, Mauro e Antônio, pela presteza, atenção e carinho com que sempre me trataram. Às funcionárias da biblioteca, Margarida e Elisiane, pela atenção com que sempre me atenderam e às funcionárias Celina e Margarete, que pacientemente revisaram o formato deste volume.

À Margarida Richter Isoppo, por tudo que dividimos e pelo que passamos

juntos.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico e ao Povo Brasileiro, a quem devo o auxílio de custo recebido ao longo do desenvolvimento do meu trabalho.

A todos aqueles que, apesar de terem contribuído para que eu concretizasse esta fase das pesquisas, não foram citados por um lapso meu.

Por último, mas mais importante, agradeço ao Grande Arquiteto do Universo, que na sua infinita bondade concedeu-me mais esta oportunidade de progresso espiritual.

À minha família, Celula Mater *onde aprendi a prezar valores como o Caráter, o Respeito e a Fé.*

" O que será de nós
se estivermos cansados
da Verdade do Amor?"
Beto Guedes/Marcio Borges
in: "Contos da Lua Vaga"

SUMÁRIO

CONVENÇÕES DE NOTAÇÃO 11

LISTA DE ABREVIATURAS 12

LISTA DE FIGURAS 17

LISTA DE TABELAS 18

RESUMO 19

ABSTRACT 21

1 INTRODUÇÃO 23

1.1 Estilos de Projetos de ASICs 25

1.2 Estilo Programável por Algumas Máscaras 28

2 CIRCUITOS PRÉ-DIFUNDIDOS 31

2.1 Conceitos Fundamentais e Taxonomia para Pré-difundidos . . 34

2.1.1 Conceitos fundamentais para arquiteturas 35

2.1.1.1 Microarquiteturas 35

2.1.1.2 Macroarquiteturas 39

2.1.2 Conceitos fundamentais para estratégias de ocupação 41

2.2 Abordagens de Pré-difundidos 45

2.2.1 Abordagens convencionais 46

2.2.2 Abordagens avançadas 49

3 O AMBIENTE TRANCA 64

3.1 A Metodologia TRANCA 65

3.1.1 Estrutura de bandas 67

3.1.2 Maleabilidade 68

3.1.3 Transparência de células e blocos 68

3.1.4 Gerenciamento de trilhas 70

3.2 O Módulo TRAMO 72

3.2.1 Topologia das células 73

3.2.2 O subsistema POTRANCA 74

3.2.3 O subsistema RETRANCA 76

3.2.4 Considerações sobre os leiautes gerados pelo TRAMO 77

3.3 O Módulo TRAGO 78

3.3.1 Pré-processamento 80

3.3.2 Particionamento 81

3.3.3 Geração de bandas 81

3.3.4 Considerações sobre os leiautes gerados pelo TRAGO 83

4 A ABORDAGEM "MAR DE CÉLULAS" **85**

4.1 Definição de uma Nova Abordagem para Pré-difundidos 86

4.2 Arquitetura da Matriz de Uso Genérico 91

4.2.1 Características Topológicas 95

4.2.2 Características Elétricas 98

4.3 Outras Considerações Sobre a Abordagem Marcela 102

5 ALGORITMOS E FERRAMENTAS PARA A GERAÇÃO DE CIRCUITOS MARCELA **106**

5.1 Estratégias para o Assinalamento de Células 111

5.1.1 Alocação da matriz 113

5.1.2 Otimização da ocupação 116

5.2 O Assinalador MARLA 122

5.3 O Roteador MARTE 125

5.4 Análise de Desempenho das Ferramentas 131

5.5 Sinopse 133

6 AVALIAÇÃO DA ABORDAGEM MARCELA 136

6.1 O Circuito FFDSR 138

6.2 O Circuito FFJKSR 139

6.3 O Circuito Modem 142

6.4 O Circuito Alu LS181 144

6.5 Considerações Sobre os *Benchmarks* 147

7 CONCLUSÃO 149

ANEXO A-1 DESCRIÇÃO SPICE DO CIRCUITO MODEM . . 152

ANEXO A-2 ARQUIVO DE CONFIGURAÇÃO PARA O ASSINALADOR 158

ANEXO A-3 ARQUIVO DE DESCRIÇÃO DO POSICIONAMENTO 159

ANEXO A-4 DESCRIÇÃO DA TOPOLOGIA DA GRADE DE ROTEAMENTO PARA MATRIZ MAR1000 165

BIBLIOGRAFIA 167

CONVENÇÕES DE NOTAÇÃO

- `Arquivos exemplo escritos em VERBATIM`
- *Termos técnicos escritos em TIMES ITALIC*
- Texto escrito em TIMES ROMAN

LISTA DE ABREVIATURAS

ASIC	Circuito integrado de aplicação específica (*Application Specific Integrated Circuit*)
CI	Circuito integrado
CIF	Formato para descrição geométrica de circuitos integrados (*Caltech Intermediate Format*)
CMOS	*Complementary Metal Oxide Silicon*
CPGCC	Curso de Pós-Graduação em Ciência da Computação
dir.	direita
DOS	Sistema operacional em disco (*Disk Operator System*)
EDIF	Formato para a troca de projetos eletrônicos (*Electronic Design Interchange Format*)
esq.	esquerda
GME	Grupo de Microeletrônica
L	Comprimento do canal do transistor
PAC	Projeto auxiliado pelo computador
RS	Formato para descrição geométrica de circuitos integrados (Rio Grande do Sul)
VCC	*Voltage Continuous-Current*
W	Largura do canal do transistor

LISTA DE FIGURAS

Figura 1.1 Classificação para o projeto de ASICs segundo [REI 92a]. 26

Figura 2.1 Leiaute de uma CB típica. 36

Figura 2.2 Configurações de CBs derivadas da estrutura *dogbone*: com portas compartilhadas (a), com portas separadas (b), com pares de transistores pequenos extras (c), com número desigual de transistores **n** e **p** (d) e especializada (e). 40

Figura 2.3 Algumas macroarquiteturas de matrizes pré-difundidas. 42

Figura 2.4 Estratégias para a realização de conexões em matrizes pré-difundidas. 44

Figura 2.5 Macroarquitetura de um *gate array* típico, em tecnologia de um nível de metal. 47

Figura 2.6 Topologia de um *gate array* em dois níveis de metal, com isolamento geométrico. 48

Figura 2.7 Topologia de um *gate array* em dois níveis de metal, com isolamento por porta. 49

Figura 2.8 Topologia de um *compacted array* típico, apresentado em [NOI 85]. 50

Figura 2.9 Estratégia de roteamento do tipo **compartilhado**, com alocação de canais implícitos sobre áreas ativas das CBs. 52

Figura 2.10 Arquitetura (a) e estratégia (b) para *sea-of-gates* do tipo *compacted array* apresentada em [KUR 85]. 53

Figura 2.11 Arquitetura (a) e estratégia de ocupação (b) da abordagem *column macro-cell* [OKU 89]. 54

Figura 2.12 Uma CB para implementação eficiente de memória e lógica aleatória [TAK 85] (a) e esquema de um bit de memória SRAM com oito transistores (b). 55

Figura 2.13 Discretização de trilhas na alocação de canais de roteamento para a matriz de [TAK 85], quando da implementação de lógica aleatória. 56

Figura 2.14 CBs da abordagem Cipredi: em tecnologia 2.0μm (esq.) e segunda versão, em tecnologia 1.2μm (dir.). 57

Figura 2.15 CB utilizada na abordagem *gate forest* [BEU 88b]. 58

Figura 2.16 Unidade básica que compõe uma matriz da abordagem Marcela. 60

Figura 2.17 Macroarquitetura em blocos (*structured array*) para a implementação de sistemas microprogramados [MIY 86] 61

Figura 2.18 Leiaute de um *flip-flop* D com *set* e *reset* implementado na matriz *gate array* do CTI. 62

Figura 3.1 Conexão entre duas células vizinhas usando o canal (*standard cells*) (a) e por justaposição (TRANCA) (b). 66

Figura 3.2 Barramento de alimentação que define a estrutura de bandas na metodologia TRANCA. 67

Figura 3.3 Passo de metal 1. 67

Figura 3.4 Uso de células de interconexão. 69

Figura 3.5 Versões **transparente** (esq.) e **opaca** (dir.) para o leiaute de uma *nand* de três entradas. 70

Figura 3.6 Efeitos do uso de mais de uma trilha por conexão: aumento da área devido ao uso de célula de interconexão e redução da transparência da banda. 71

Figura 3.7 Esquema de prioridades na alocação de trilhas nas células da biblioteca do Projeto TRANCA. 72

Figura 3.8 Leiaute de uma *and* de 4 entradas. 73

Figura 3.9 Algoritmo de particionamento do Potranca. 75

Figura 3.10 Um possível leiaute TRAMO para o circuito Modem. 79

Figura 3.11 Fluxo da síntese com o módulo TRAGO. 80

Figura 3.12 Leiaute de uma célula gerada pelo TRAGO. 81

Figura 3.13 Um possível leiaute TRAGO para o circuito Modem. 84

Figura 4.1 UB da matriz de uso genérico. 92

Figura 4.2 Esquema de prioridades na alocação de trilhas em arquiteturas Marcela. 94

Figura 4.3 Leiautes das CBs da matriz de uso genérico sob a grade de roteamento. 96

Figura 4.4 Leiaute da matriz MAR1000. 97

Figura 4.5 Diagrama de ocorrências de fanouts de uma amostra de equivalentes Marcela. 99

Figura 4.6 Dois equivalentes Marcela para *xor*: com portas *nand* (esq.) e com *transmission gates* (dir.). 103

Figura 4.7 Leiautes para as duas versões da *xor*: com portas *nand* (esq.) e com *transmission gates* (dir.). 104

Figura 4.8 Dois equivalentes Marcela para o *flip-flop* D com *set* e *reset*. . . 105

Figura 4.9 Leiaute para as duas versões do *flip-flop* D SR: dinâmico, com *transmission gates* (esq.) e estático (dir.). 105

Figura 5.1 Fluxo de projeto para circuitos segundo a abordagem Marcela. . 107

Figura 5.2 Particionamento por quadratura. 118

Figura 5.3 Quadratura sobre um leiaute Marcela. 119

Figura 5.4 Lista de redes e lista de células. 123

Figura 5.5 Procedimento para alocação de matrizes Marcela. 124

Figura 5.6 Estrutura do ambiente de roteamento MARTE. 126

Figura 5.7 Matriz de restrições para a UB da matriz MAR1000. 128

Figura 5.8 Parâmentros para a grade de roteamento. 131

Figura 5.9 Roteamento simbólico do circuito Modem. 132

Figura 5.10 Leiautes para as três versões do Modem: versão manual (topo), versão semi-automática (centro) e versão automática (abaixo). . 134

Figura 6.1 Leiaute de um *flip-flop* D com *set* e *reset*, versões Marcela (dir.) e Cipredi (esq.). 138

Figura 6.2 Esquemático básico para o FFJKSR. 140

Figura 6.3 Leiaute do FFJKSR, versões Marcela (topo) e Cipredi (abaixo). 141

Figura 6.4 Leiautes das versões Marcela (topo) e Cipredi (abaixo) do circuito Modem. 143

Figura 6.5 Leiautes das versões Marcela (topo) e Cipredi (abaixo) do circuito Alu LS181. 146

LISTA DE TABELAS

Tabela 2.1 Resumo das características das abordagens avançadas apresentadas. As densidades referem-se a transistores efetivamente utilizados nas matrizes. 63

Tabela 4.1 Evolução das dimensões mínimas da tecnologia CMOS. 88

Tabela 4.2 Capacitâncias representadas pelas conexões (C) e capacitâncias representadas pelas portas dos transistores (C_t). 100

Tabela 4.3 Atrasos obtidos por simulação com *fanout* 12 (C_L=240fF). . . . 101

Tabela 5.1 Características das 3 versões do circuito Modem. 133

Tabela 6.1 Dados sobre as versões Marcela e Cipredi para o FFDSR. 139

Tabela 6.2 Dados sobre as versões Marcela e Cipredi para o FFJKSR. . . . 142

Tabela 6.3 Dados sobre as versões Marcela e Cipredi para o circuito Modem. 144

Tabela 6.4 Dados sobre as versões Marcela e Cipredi para o circuito AluLS181. 147

RESUMO

Este trabalho propõe e avalia uma nova abordagem para projeto de circuitos dedicados utilizando matrizes pré-difundidas. A principal vantagem desta abordagem, denominada Marcela, reside na decomposição lógica do circuito a ser implementado em termos de primitivas disponíveis na matriz escolhida. Aplicando-se tal procedimento, alcança-se grande flexibilidade em termos de posicionamento e roteamento, levando a uma melhor taxa de ocupação.

Primeiramente, é feito um levantamento das abordagens para pré-difundidos correntemente encontradas e uma taxonomia baseada nas características mais relevantes é definida.

As principais características da metodologia TRANCA são também mostradas. Leiautes gerados com os módulos TRAMO e TRAGO são analisados e algumas modificações na metodologia são sugeridas, visando uma exploração mais eficiente dos dois níveis de metal.

As bases para o desenvolvimento da abordagem Marcela são então descritas. A abordagem consiste de uma nova arquitetura para pré-difundidos e uma estratégia específica de ocupação. As principais características da matriz de propósito geral Marcela, primeira a ser definida, são a ausência de canais de roteamento, com as conexões sendo realizadas sobre as células, e a utilização de quatro tipos de células básicas, cada uma dedicada à implementação de uma função lógica primitiva. As células básicas estão organizadas em **unidades básicas**, as quais são repetidas regularmente para formar a matriz, numa abordagem denominada **mar de células**.

O problema do assinalamento de células e suas particularidades são solucionados utilizando-se uma combinação entre alocação seqüencial e técnicas de particionamento. Primeiro, é alocada a mínima superfície da matriz capaz de comportar o circuito em questão, numa fase chamada pré-assinalamento. Na fase de

otimização, partições são geradas respeitando a integridade das unidades básicas e trocas de células são realizadas entre os blocos de cada nova partição, em dois passos: trocas individuais, enquanto o bloco de destino não estiver cheio, e trocas de pares.

Para o roteamento, foi desenvolvida no CPGCC/UFRGS uma ferramenta específica para ser utilizada em leiautes gerados segundo a metodologia TRANCA. Esta ferramenta, denominada MARTE [JOH 92a][JOH 92b], emprega o algoritmo de Lee básico com algumas modificações, tal como a geração de *doglegs* para trocas entre trilhas adjacentes.

Com a finalidade de validar a abordagem, foram implementados alguns circuitos utilizando a abordagem Marcela e uma abordagem *sea-of-gates* tradicional. Para circuitos pequenos, tal como um *flip-flop* D, Marcela produziu uma melhor distribuição de conexões, a qual resulta em aumento da transparência. Porém, a taxa de ocupação encontrada foi menor do que a do circuito projetado com *sea-of-gates*. Por outro lado, para circuitos de complexidade maior, a área ocupada pode resultar bem menor do que no caso de se usar *sea-of-gates*, desde que sejam realizadas transformações lógicas apropriadas sobre a descrição equivalente Marcela ou uma matriz conveniente seja escolhida. Exemplos de leiautes desenvolvidos mostram que taxas de ocupação tão altas quanto 75% são atingidas.

Finalmente, da observação de circuitos gerados automaticamente, foram tiradas conclusões sobre modificações na arquitetura da matriz e nos algoritmos, de forma a melhorar as taxas de ocupação para qualquer tipo de circuito.

PALAVRAS-CHAVE: Microeletrônica, Pré-Difundidos, Projeto VLSI, Ferramentas de PAC.

TITLE: "Circuit Generation Using Prediffused Sea-of-Cells Masterslices"

ABSTRACT

This work proposes and evaluates a new approach for the design of ASICs using prediffused masterslices. The main advantage of this approach, called Marcela, relies on logic decomposition of the circuit to be implemented into the chosen masterslice available primitives. By applying this procedure, a great placement and routing flexibility is achieved, thus leading to a better transistor utilization rate.

First, a survey on current prediffused approaches is done and an specific taxonomy is defined based on the main important features encountered.

Also the main features of TRANCA methodology are shown. Layouts generated using TRAGO and TRAMO modules are analyzed and some modifications in the methodology are suggested, in order to better exploit both first and second metal layers.

Marcela approach development basis are described. The approach consists of a new prediffused architecture and an specific occupation strategy. The main architectural features of the general purpose Marcela masterslice are the absence of routing channels, with the connections running over the cells, and the utilization of four types of basic cells, each of them dedicated to perform one primitive logic function. Basic cells are organized into **basic units**, which are spread all over the masterslice, in a so called **sea-of-cells** approach.

The assignment problem and its peculiarities are solved by using a combination of sequential cell allocation and quadrature partition techniques. But first of all, a minimum masterslice area is allocated in a phase called preassignment. In the optimization phase, partitions are generated respecting basic units integrity and cell interchanges are applied to each new partition, following two steps: individual

changes, while the target block is not full, and pairwise interchange.

For the routing problem, an specific tool has been developed at CPGCC/ UFRGS for any module generator in which TRANCA methodology is applied. This tool, called MARTE [JOH 92a][JOH 92b], employs a basic Lee algorithm with some modifications as dogleg generation for changes between adjacent tracks.

In order to validate the approach, some circuits have been implemented using a traditional sea-of-gates and Marcela approaches. For small circuits, as a D flip-flop, Marcela approach has produced a better wiring distribution, which results in increase of transparency. But the occupation rate was found to be smaller than that of the sea-of-gates approach. On the other hand, for more complex circuits the amount of used area can be smaller than that of sea-of-gates case, since appropriate logic transformations are applied to the Marcela logic equivalent or a well suit masterslice is used. Implemented examples show that utilization rates as high as 0.75 are achieved.

Finally, from the observation of automatically generated layouts some modifications in masterslice architecture and in the algorithms are figured out.

KEYWORDS: Microelectronics, Prediffused Circuits, VLSI Design, CAD tools.

1 INTRODUÇÃO

A metodologia de projeto de circuitos integrados tem recebido grande atenção ao longo da última década e relevantes avanços foram alcançados no campo da automatização do projeto nos vários níveis. Os primeiros esforços foram concentrados na geração automática do leiaute, principalmente por ser esta a fase mais entediosa e sucetível a erros. Incontáveis algoritmos e heurísticas foram desenvolvidos para endereçar os problemas de **posicionamento**, **roteamento** e **planejamento topológico**, muitos deles oriundos do projeto de placas de circuito impresso devido à semelhança dos problemas. Numa etapa seguinte, os algoritmos passaram a ser mais elaborados, de forma a endereçar os problemas específicos da microeletrônica. Surge então o uso de células padrão como forma de acelerar o projeto de circuitos integrados e em seguida, a rápida evolução tecnológia força o desenvolvimento da geração automática de células visando tornar o projeto o mais independente possível das variações nas regras de desenho numa dada tecnologia.

A exploração dos avanços da tecnologia CMOS em conjunto com a utilização de ferramentas de PAC cada vez mais sofisticadas proporcionou o grande avanço tecnológico experimentado na década de 80, e que, ao que tudo indica, deverá continuar na década de 90. A popularização da eletrônica teve como conseqüência a criação de um mercado certo para a microeletrônica, com os chamados circuitos integrados para aplicações específicas (ASICs) sendo utilizados em profusão. O surgimento cíclico de novas áreas de aplicação para a eletrônica sacode o mercado de microeletrônica, criando novos nichos a serem explorados e permitindo que a demanda por ASICs continue crescendo.

Nos últimos anos as opções para implementação de ASICs têm aumentado bastante. As principais opções são os *full-customs*, os *standard cells*, os pré-difundidos (também referenciados por *gate arrays*) e mais recentemente, os circuitos programáveis pelo usuário, dentre os quais se enquadram os EPLDs (Erasable Pro-

grammable Logic Devices) e os FPGAs (Field Programmable Gate Arrays). Constantemente, os fabricantes têm lançado novos produtos dentro desta última opção, com características bastante variadas, mas sempre buscando aumentar o número de portas disponíveis aos usuários. Atualmente, tais circuitos apresentam capacidade limitada em cerca de 5000 *gates equivalentes* efetivamente utilizáveis.[1] Os principais fabricantes mundiais de FPGAs são a Xilinx Inc. e a Actel Co., enquanto que em EPLDs, o principal fabricante é a Altera Co.

Sem dúvida, este aumento no número de opções de implementação de ASICs reflete a disputa por um mercado que em 1990 movimentou 7,3 bilhões de dólares e cuja perspectiva é crescer ainda mais, devendo alcançar 12,4 bilhões de dólares em 1995 [TYL 91].

Porém, há de se ressaltar que o tempo de vida útil dos produtos eletrônicos é cada vez menor, em parte fruto da própria competição dos fabricantes. A necessidade de lançar produtos com inovações capazes de cativar o consumidor antes que a concorrência o faça tem impelido os fabricantes a reduzirem cada vez mais o tempo de projeto. O período que decorre entre a idéia de um novo produto e sua efetiva colocação no mercado é denominado **tempo para o mercado** e envolve especificação, projeto, prototipação, testes, reprojeto (se necessário), fabricação e finalmente lançamento do produto. Então, se o tempo de vida útil tem diminuído e o custo de desenvolvimento tem aumentado, a amortização do investimento é cada vez mais difícil, de forma que somente empresas com estratégias de desenvolvimento avançadas conseguirão se manter na concorrência. Além disso, mesmo com toda a concorrência forçando o preço final dos produtos para baixo, este alto custo de desenvolvimento deve ser repassado ao consumidor em algum momento. Neste caso, o percentual médio representado pelo custo dos CIs dentro dos sistemas eletrônicos tenderia a aumentar. Em 1989, este percentual era de 7% [CAR 92].

Então é bastante natural que sejam buscadas formas de implementação

[1] Um *gate equivalente* é definido como o número de transistores necessários para implementar uma porta *nand* de duas entradas em lógica complementar, ou seja, 2 pares de transistores **n-p**.

que reduzam o tempo de projeto ao máximo, sem comprometer a confiabilidade do produto.

1.1 Estilos de Projetos de ASICs

O surgimento de novas formas de implementação de CIs tem motivado novas tentativas de classificação. Em [REI 92a] é apresentada uma classificação que leva em conta o momento em que o circuito é configurado, agrupando as etapas da prototipação em **processo inicial**, **metalização** e **programação**.

O processo inicial diz respeito à definição dos dispositivos no silício, com a formação das regiões **n**, **p** e polissilício da porta. A metalização refere-se à etapa em que são implementadas as conexões físicas entre os elementos definidos no processo inicial e ao encapsulamento do circuito. A programação consiste na configuração do circuito através da introdução de dados ou pela alteração de sua estrutura física. A figura 1.1 mostra as etapas da prototipação de ASICs.

Então, os projetos podem ser classificados segundo a etapa na qual o circuito é diferenciado (personalizado), recebendo as denominações:

- programáveis por todas as máscaras;
- programáveis por algumas máscaras;
- programáveis após o encapsulamento.

Os ASICs **programáveis por todas as máscaras** tem seu processo de fabricação diferenciado (personalização) desde os processos iniciais. Esta situação leva à divisão dos custos fixos pelo volume de produção individual de cada ASIC, sendo portanto aplicáveis somente quando os volumes individuais de produção forem altos ou quando o desempenho dos circuitos tenha que ser elevado. O desempenho

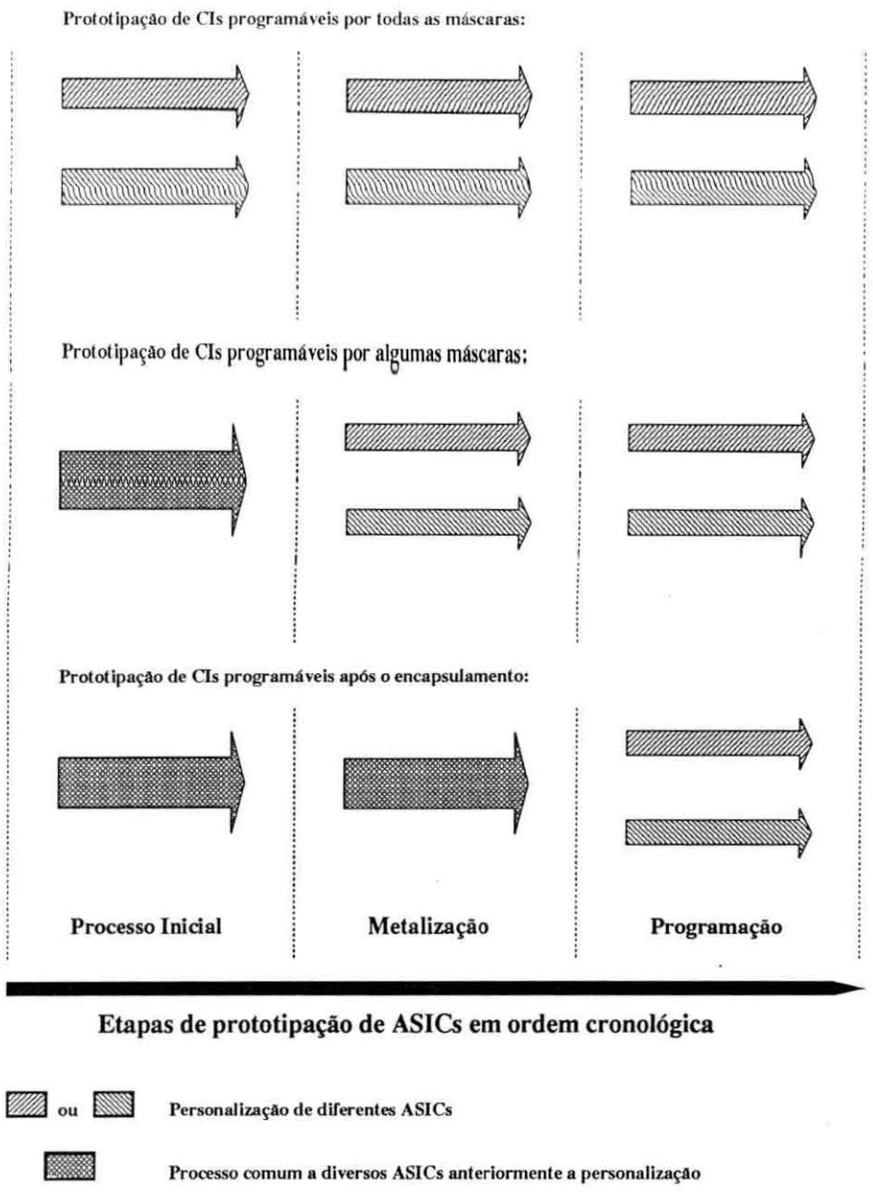

Figura 1.1: Classificação para o projeto de ASICs segundo [REI 92a].

elevado é garantido pelo fato de todas as etapas serem específicas. Esta classificação engloba os estilos *full custom*, a geração em módulos regulares e a geração em módulos de lógica aleatória.

Para os ASICs **programáveis por algumas máscaras**, a personalização inicia só a partir do processo de metalização, o que acaba amortizando os custos fixos do processo inicial e tornando a prototipação mais rápida (menos etapas de personalização). Enquadra-se aqui o estilo pré-difundido, com todas as abordagens existentes.

Já para os ASICs **programáveis após o encapsulamento**, a personalização se dá somente após a metalização o que, segundo esta taxonomia, inclui o encapsulamento. Aqui os custos fixos são reduzidos e a prototipação é extremamente rápida, consistindo apenas na programação de memórias internas (do tipo EPROM ou RAM) ou na ruptura de elementos de conexão, o que pode ser feito na própria bancada do projetista. Fazem parte desta classe os EPLDs e os FPGAs.

Estabelecida uma taxonomia abrangente para a implementação de ASICs, examinemos as características desejáveis tendo em mente as condições de competividade exigidas pelo mercado de eletrônica.

As principais características desejáveis nas formas de implementação podem ser enumeradas:

- Rapidez e segurança no projeto;
- Rapidez na prototipação;
- Funcionamento imediato garantido;

A rapidez e segurança é um reflexo direto do ambiente de projeto em si, no que concerne às ferramentas utilizadas e na integração das mesmas. Um ambiente completo deve prover total transparência de formatos para o usuário ou pelo menos o mínimo de formatos intermediários. Deve haver um conjunto de ferramentas que permitam fácil descrição do circuito, simulação, geração do leiaute, extração e ressimulação, no mínimo. Tudo isso aliado a um desempenho razoável das ferramentas e a interfaces amigáveis.

Mas sem dúvida, a rapidez de prototipação e o funcionamento imediato do projeto são os objetivos mais perseguidos atualmente, pois reduzem o tempo para mercado. A importância de ser o primeiro a lançar um produto é tal que por vezes são implementadas versões precursoras utilizando algum estilo de projeto com tempo de desenvolvimento reduzido mesmo que o desempenho do circuito não seja

muito bom. Caso os resultados de vendas sejam satisfatórios, novas versões com desempenho superior (em outro estilo de projeto) são desenvolvidas, já com a maior fatia do mercado assegurada.

No quesito rapidez de prototipação, os PLDs (EPLDs e FPGAs) são os que apresentam a maior vantagem por não necessitarem ser remetidos de volta à *foundry*. A prototipação pode ser feita pelo próprio projetista. Por outro lado, seu desempenho elétrico não é alto e a capacidade de integração ainda é baixa. Além disso, as ferramentas de alocação e roteamento a serem utilizadas nos FPGAs ainda demandam a interferência direta do usuário na realização das conexões faltantes, devido ao alto grau de complexidade da tarefa. Apesar da facilidade de prototipação, os FPGAs e EPLDs apresentam custo fixo independente do número de peças, o que determina o uso de outros estilos quando a demanda atinge a casa da dezena de milhares. Notadamente, nesta faixa acima, os pré-difundidos (programáveis por algumas máscaras) tendem a apresentar menor custo e maior benefício.

1.2 Estilo Programável por Algumas Máscaras

No estilo programável por algumas máscaras, o tempo para prototipação é bastante reduzido em relação ao programável por todas as máscaras, no caso de se considerar um processo com litografia por máscaras, pois a maior parte das etapas do processo já se encontram realizadas. Inclusive, este fator permite que o custo de produção seja dividido entre os vários usuários de uma mesma estrutura. Seu desempenho, no entanto, fica aquém daquele obtido pelos programáveis por todas as máscaras, sendo preferíveis quando não houver necessidade de circuitos muito velozes. Aliás, a maior parte dos ASICs desenvolvidos para aplicações na eletrônica de consumo visam principalmente a miniaturização, sem necessariamente demandar alto desempenho.

O número de variáveis envolvidas na escolha da opção de implementação

mais apropriada para cada caso é grande e a decisão deve ser tomada considerando-se a melhor relação custo/benefício resultante da interação entre tais variáveis. Certamente, a primeira providência é verificar se a opção candidata e a metodologia de projeto a ela associada viabilizam a implementação do projeto em questão. Satisfeita esta primeira premissa, parte-se para a análise da segunda, e por vezes a mais importante: a demanda.

O estilo programável por algumas máscaras é economicamente apropriado para baixas demandas, desde que o processo de fabricação efetivamente use máscaras, ao invés de escrita por feixe de elétrons. Porém, se a demanda for extremamente reduzida, o estilo mais apropriado pode ser o programável após o encapsulamento. A decisão então, deve levar em conta outros fatores, tais como o desempenho elétrico e o tempo para mercado. Com relação ao tempo para mercado, o estilo programável após o encapsulamento tem como grande vantagem o tempo de prototipação quase nulo.

Quanto ao número de dispositivos efetivamente disponíveis para a implementação de projeto, os circuitos programáveis por algumas máscaras são bem superiores aos programáveis após o encapsulamento, o que conduz a melhores taxas de aproveitamento do silício além da possibilidade de realização de circuitos mais complexos.

O critério de escolha pode ainda ser puramente circunstancial. Isto ocorre quando há necessidade de lançar um produto no mercado antes dos concorrentes. Geralmente, uma versão precursora é realizada com um estilo de projeto que proporcione baixo tempo para mercado e satisfaça aos requisitos de desempenho e capacidade de integração. Conquistada a parcela desejada do mercado, novas versões com melhor desempenho podem ser implementadas em outros estilos de projeto.

Por certo, as fronteiras que delimitam o uso de cada opção não são rígidas e podem variar conforme a tecnologia avança ou conforme a situação do mercado evolui. Segundo dados reais [TYL 91], o mercado mundial de circuitos dedicados

deverá continuar crescendo substancialmente. Quanto à divisão entre os estilos, a tendência é que os projetos *full custom* percam até 25% de sua fatia para os *standard cells* e pré-difundidos até 1995, enquanto que os programáveis após o encapsulamento cheguem a 6% do mercado global, no mesmo período. Porém, a tendência também revela um crescimento de mercado para todos os estilos de projeto. Estes dados por si só já justificam o esforço de desenvolvimento de todas as metodologias de projeto.

2 CIRCUITOS PRÉ-DIFUNDIDOS

Este capítulo discute os conceitos básicos relativos ao projeto de circuitos pré-difundidos, mostrando as principais estratégias e arquiteturas relacionadas, de modo a proporcionar subsídios para as análises que serão feitas em capítulos posteriores.

Com o objetivo de sistematizar o estudo de diversas arquiteturas, é apresentada também uma taxonomia, baseada nas diversas existentes na literatura, porém visando endereçar simultaneamente um maior número de abordagens. Com isso, se espera poder classificar de forma clara um grande número de arquiteturas, inclusive as mais recentes.

O estilo pré-difundido, também conhecido genericamente por *gate array*, entrou decididamente em cena no final dos anos 70, como uma forma de baratear os custos de produção de circuitos para aplicações específicas (ASICs), uma vez que vários usuários dividiriam um mesmo tipo de recurso que já se encontrava semi acabado. Os primeiros *gate arrays* utilizavam tecnologia TTL. Pouco mais tarde, foram lançados *gate arrays* CMOS, dada a maior capacidade de integração e menor consumo de potência de tal tecnologia.

A organização de uma estrutura pré-difundida obedece a forma de matriz de elementos regulares (ou pouco irregulares), cercadas de células de interface com o exterior (*pads*), configuráveis. Tais matrizes são pré-processadas em grandes quantidades, até o nível de metalização, para posterior utilização tanto em prototipação rápida, como em produção regular. Os elementos, por sua vez, podem ser transistores ou grupos de transistores, o que será discutido mais adiante.

Para a implementação de um projeto, os elementos devem ser convenientemente conectados, através de um ou mais níveis de metal, numa etapa de projeto denominada **personalização** da matriz. Após o assinalamento dos elementos e o

projeto das conexões, a descrição das máscaras é enviada à empresa que realiza o serviço de integração (denominada *foundry*), para a realização das etapas de metalização que irão efetivar a personalização das matrizes.

Geralmente, são as *foundries* que desenvolvem matrizes pré-difundidas, com o intuito de oferecer serviços de prototipação rápida. A proximidade com os avanços tecnológicos permite que estes sejam explorados de imediato no projeto de novas matrizes, o que não deixa de ser uma tentativa de compensar a falta de flexibilidade inerente ao estilo pré-difundido. Por outro lado, há todo um custo operacional embutido nas atualizações de tecnologia, justamente devido a esta alta dependência tecnológica. Resulta que, para oferecer ao cliente uma gama razoável de opções, as *foundries* que trabalham com *gate arrays* mantém famílias de matrizes, as quais se diferenciam pelas arquiteturas voltadas para aplicações diversas (às vezes, em tecnologias diversas). Cada família, por sua vez, é composta por matrizes de várias capacidades. Além disso, para facilitar a realização dos projetos com as matrizes, geralmente é fornecido um pacote de ferramentas, o qual inclui uma ou mais bibliotecas que contêm padrões de personalização para primitivas lógicas (inversor, *nand*, *nor*), blocos funcionais (*flip-flops*, multiplexadores, decodificadores) e por vezes até macrocélulas (registradores de deslocamento, somadores, multiplicadores). No caso de uma atualização de tecnologia, é preciso reprojetar todas as matrizes, bem como revalidar as bibliotecas pertinentes e possivelmente adaptar as ferramentas de PAC, o que demanda tempo e incorre em altos custos.

Apesar das várias desvantagens tais como desperdício de área e pouca flexibilidade de projeto, quando do seu surgimento, os *gate arrays* ofereciam algumas vantagens indiscutíveis sobre outros estilos *semicustom*, tais como reduzidos tempos para projeto e para fabricação, menor custo por peça para pequenas quantidades e necessidade de menor grau de especialização do projetista. Porém, com o avanço das ferramentas de PAC e dos geradores de módulos, o tempo de projeto já não representa uma vantagem tão clara. Além disso, com a sofisticação alcançada pelos geradores automáticos de leiaute, a figura do projetista de circuitos integrados

tende a ser substituída pela do engenheiro de sistemas. Na produção de pequenas quantidades, por sua vez, os *gate arrays* ganharam um forte concorrente, os circuitos programáveis pelo usuário, principalmente os FPGAs, para os quais o custo de produção para pequenas quantidades de peças é bastante reduzido. Então, os *gate arrays* freqüentemente têm sido utilizados para quantidades não muito pequenas de peças, na implementação de ASICs que representam versões precursoras, com o objetivo de acelerar o lançamento de novos produtos e desta forma conquistar nichos de mercados antes que os concorrentes o façam.

Por outro lado, o tempo de fabricação dos pré-difundidos ainda tende a permanecer menor do que os demais estilos programáveis por máscaras, devido ao fato da maior parte das etapas de processo encontrarem-se realizadas, apenas aguardando a personalização. Tipicamente, a personalização de um circuito pré-difundido pode representar desde 10% das etapas de processo, quando apenas um nível de metal é utilizado, até 50%, quando mais de dois níveis são utilizados [SIM 92] [YOR 88]. A tendência que tem se confirmado é a utilização de um número cada vez maior de níveis de personalização, à medida em que a tecnologia assim o permite. O fato é que, mesmo com o aumento do tempo e do custo da personalização, o ganho em flexibilidade (refletindo em melhor ocupação das matrizes) é mais significativo.

Dois aspectos devem ser considerados quando da análise das várias abordagens de pré-difundidos: a **arquitetura da matriz** e a **estratégia de sua ocupação**. A arquitetura tem sido uma preocupação constante nos pré-difundidos devido a pouca flexibilidade que estes apresentam no que se refere à realização das conexões. Com a grande evolução da tecnologia, que vem ocorrendo principalmente a partir da década de 80, houve uma certa flexibilização das arquiteturas. Porém, o mesmo avanço pôde ser experimentado pelos demais estilos de projeto, podendo-se concluir que sob este aspecto, não houve ganho com relação às demais formas de implementação de ASICs.

A estratégia de ocupação, por sua vez, diz respeito a **como** a matriz será utilizada para a implementação da lógica desejada e **como** serão realizadas

as conexões entre os elementos que compõem esta lógica. Em última instância, a estratégia de ocupação define o modo de utilizar a superfície de silício disponível na matriz. O aprimoramento das estratégias de ocupação tem se intensificado, a tal ponto deste ter igual ou maior influência na ocupação da matriz do que a própria arquitetura escolhida. Esta relação entre arquitetura e estratégia tende a se tornar cada vez mais íntima, contribuindo para uma constante melhora na ocupação das matrizes pré-difundidas.

2.1 Conceitos Fundamentais e Taxonomia para Pré-difundidos

Uma preocupação constante dos projetistas que se utilizam de matrizes pré-difundidas é escolher corretamente uma matriz que permita a implementação do circuito e minimize a área de silício. Para minimizar a área de silício é necessário que a matriz escolhida seja a menor possível.

Neste sentido, a **taxa de ocupação** para um dado projeto mede a qualidade da utilização da matriz escolhida, sendo definida como a razão entre o número de transistores efetivamente utilizados na lógica do circuito e o número total de transistores disponíveis na matriz. A **densidade** de transistores disponíveis, por sua vez, tem sido uma medida bastante utilizada para expressar o grau de complexidade de projeto que uma determinada matriz é capaz de comportar. Porém, uma medida mais eficaz é a **densidade útil** de transistores, a qual corresponde a aplicação de uma taxa de ocupação média sobre a densidade da matriz. Os catálogos dos fabricantes contém necessariamente ou a densidade útil da matriz ou a taxa de ocupação média, de forma a permitir uma avaliação rápida da viabilidade de sua utilização com uma devida margem de segurança.

As diversas abordagens de pré-difundidos podem ser sistematicamente classificadas conforme as características apresentadas pela **arquitetura da matriz**

e pela **estratégia de ocupação**. Os diversos conceitos associados à arquitetura foram surgindo à medida que as tecnologias de fabricação evoluiam, permitindo novas configurações topológicas. Existe, portanto, um conjunto de conceitos já razoavelmente utilizados, cujas definições variam conforme o autor. Aqui não será diferente, uma vez que se tenta criar subsídios para englobar o maior número possível de configurações de matrizes.

Já os conceitos envolvidos nas estratégias de ocupação não são, via de regra, citados explicitamente pelos demais autores, aparecendo apenas como uma decorrência natural da arquitetura adotada. Porém, face ao surgimento de estratégias mais elaboradas, estes já são merecedores de certa atenção.

2.1.1 Conceitos fundamentais para arquiteturas

A arquitetura de matrizes pré-difundidas pode ser detalhada em dois níveis de abstração: **microarquitetura** e **macroarquitetura** [BEU 88b].

Microarquitetura diz respeito à menor unidade identificável na matriz, a qual recebe o nome de **célula de base** (CB). Na literatura, célula de base é freqüentemente referida por *logic cell* e *core cell.*

A macroarquitetura está relacionada com a forma pela qual a ou as células básicas são organizadas para formar as matrizes. Neste conceito estão contidas tanto a forma de realizarem-se as conexões entre células, como a existência ou não de uma estrutura de submatrizes ou blocos dedicados a implementações específicas.

2.1.1.1 Microarquiteturas

A figura 2.1 mostra o leiaute de uma CB tipicamente encontrada em matrizes pré-difundidas. As principais características topológicas das CBs são:

- número de transistores de cada tipo e a proporção entre estes;
- tamanho dos transistores;
- topologia das regiões de fonte/dreno e do polissilício da porta;
- técnica utilizada no isolamento das CBs;
- grau de especialização da CB.

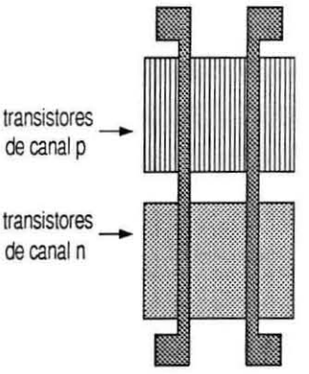

Figura 2.1: Leiaute de uma CB típica.

Cada CB pode ser composta simplesmente por um par **n-p** de transistores ou mesmo por vários pares. Na maior parte das matrizes, a CB é definida para comportar a implementação de uma porta **nand** ou uma porta **nor** de duas entradas [YOR 88]. A definição do número de transistores da CB é dependente da regularidade de sua topologia e pode determinar o grau de aproveitamento da matriz, conforme a estratégia de personalização adotada. Por exemplo, caso a personalização for realizada por meio de uma biblioteca de padrões de metalização contendo primitivas lógicas, cada primitiva não ocupa um número inteiro de CBs, necessariamente, podendo restar alguns transistores, os quais não poderão ser utilizados por outra primitiva.

O tamanho dos transistores é alvo de estudos detalhados quando do projeto de uma nova matriz, principalmente pelo caráter irrevogável da escolha. Pela própria característica de estarem pré-difundidos, é necessário que seja adotado um tamanho padrão, tal que a conectibilidade e o desempenho elétrico sejam razoáveis.

Porém, quanto ao desempenho elétrico, muito pouco há para fazer, uma vez que a minimização individual das primitivas é impossível. Existem CBs cujos transistores p possuem largura de canal W entre 1,5 a 2 vezes a largura de canal do transistor n, como tentativa de aproximar os tempos de subida e descida de um inversor em lógica complementar. Mas no que se refere à conectibilidade, as restrições das regras de desenho, juntamente com a necessidade de aumentar a flexibilidade fazem com que os transistores resultem bem mais largos do que o mínimo tecnológico. Um mínimo de três posições disponíveis para colocação de contatos para cada região de dreno e fonte deve ser provida para que ao menos as conexões internas às macrocélulas possam ser acomodadas sem a utilização de canais de roteamento. Um número maior de posições para contatos tende a aumentar a flexibilidade das conexões, podendo permitir inclusive o roteamento global sobre as CBs, utilizando trilhas disponíveis, o que caracteriza a abordagem de roteamento conhecida por *channelless*. Porém, a contrapartida é o aumento das cargas capacitivas representadas pelas regiões de dreno e fonte dos transistores, resultando em maiores correntes e maiores atrasos de propagação de sinais. Também podem ocorrer mais congestionamentos, dada a possibilidade de que conexões simples utilizem caminhos mais complexos. Logo, existe um tamanho ótimo para os transistores, o qual é ditado principalmente pela estratégia de roteamento a ser adotada na matriz.

As topologias de regiões de fonte e dreno e do polissilício da porta também dependem das estratégias de roteamento e procuram sempre aumentar a flexibilidade na conexão dos elementos, de maneira a facilitar a atuação das ferramentas automáticas de roteamento.

Quanto às técnicas de isolamento de CBs, existem duas:

- isolamento geométrico ou por óxido de campo;
- isolamento por porta (em inglês, *gate isolation*).

No isolamento geométrico as CBs estão separadas por regiões de óxido espesso, sendo a topologia, portanto, inalterável. Já no isolamento por porta [OKH 82], as CBs encontram-se dispostas de forma contínua, com os transistores limítrofes compartilhando regiões de fonte/dreno. Para efetuar o isolamento, um transistor ou um par de transistores devem ser **cortados**, mediante a conexão de sua porta à Vdd (canal **p**) ou à terra (canal **n**).

O grau de especialização de uma CB é o número de aplicações específicas, para as quais esta foi especialmente projetada. Na busca da universalização de aplicações de uma matriz, duas situações podem ocorrer:

- ou a CB permite uma grande gama de aplicações, apresentando pequeno ou nenhum grau de especialização;
- ou existe mais de uma CB na matriz, de modo que no conjunto, a gama de aplicações desejadas encontra cobertura.

A CB mostrada na figura 2.1 apresenta uma topologia comumente utilizada, conhecida por *dogbone* [BEU 88b]. A configuração básica da estrutura *dogbone* é constituída de dois pares de transistores **n-p** com drenos e fontes compartilhados. O isolamento entre as CBs é geométrico e os pares podem ou não compartilhar as portas.

A figura 2.2 mostra algumas variações possíveis em torno da estrutura *dogbone*. A utilização de portas compartilhadas e a inclusão de tiras de difusão para aumentar a conectividade (figura 2.2a) são recursos tipicamente encontrados nos primeiros *gate arrays*, onde a personalização era realizada com somente um nível de metal.

A configuração com portas separadas (figura 2.2b) permite o compartilhamento de drenos e fontes entre transistores adjacentes, o que aumenta o número de dispositivos disponíveis na matriz. Nesta topologia, o isolamento por porta é utilizado, permitindo uma melhor ocupação da matriz quando da implementação de

chaves **cmos** (*transmission gates*) e viabilizando a realização de circuitos em lógica dinâmica, conforme descrito em [NOI 85].

A inclusão de transistores de menor tamanho ou então uma proporção maior de transistores de canal **n** facilitam a implementação de bits de memória RAM em qualquer porção de uma matriz com tal configuração, o que é citado em [BEU 88a], [CAL 88a], [KUR 87] e [TAK 85]. As CBs das figuras 2.2c e 2.2d ilustram as topologias anteriormente citadas.

Também é possível modificar-se a estrutura *dogbone* de modo a aumentar seu grau de especialização. Neste sentido, a figura 2.2e mostra uma CB dedicada à implementação de estruturas PLA.

2.1.1.2 Macroarquiteturas

Algumas tentativas de classificar estruturas pré-difundidas tem sido feitas. Porém, ou a classificação não abrange arquiteturas recentes [YOR 88], ou não há diferenciação clara entre roteabilidade e distribuição de CBs na matriz [BEU 88b].

Desta forma, a taxonomia aqui adotada endereça separadamente as características supracitadas através de dois conceitos independentes: **roteabilidade** e **granularidade**.

A roteabilidade versa sobre a presença ou ausência de regiões dedicadas à realização das conexões, ou seja, o roteamento. O tipo de roteamento a ser realizado na região (intra ou intercelular) é indiferente para efeitos da presente classificação.[1] Então, quanto à roteabilidade, as macroarquiteturas são classificadas em:

- com canal (*channelled*);
- sem canal (*channelless*).

[1] Na realidade, o tipo de roteamento a ser realizado faz parte da estratégia de ocupação, a qual será detalhada no próximo item.

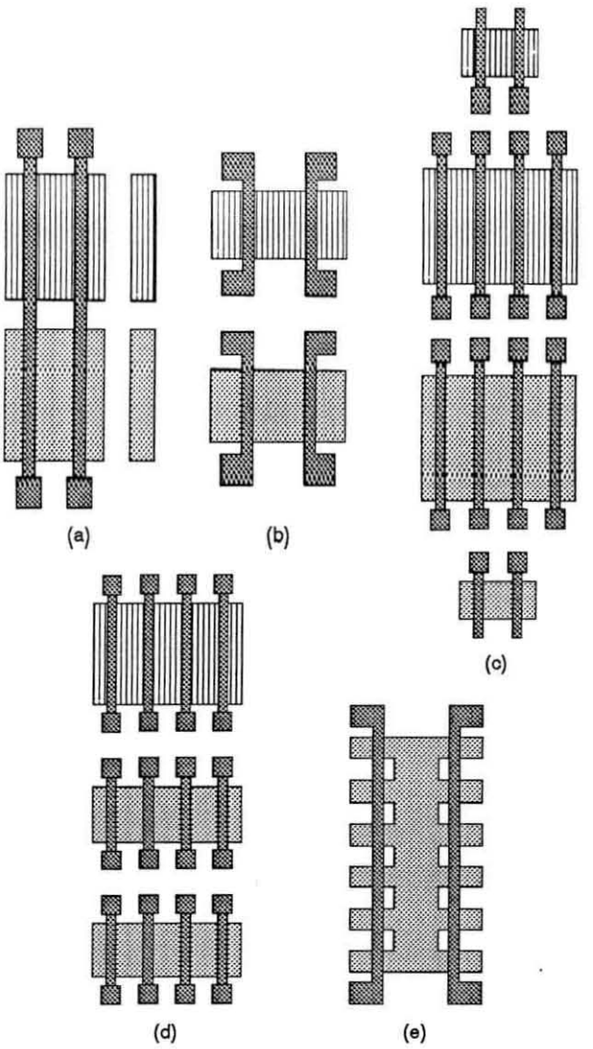

Figura 2.2: Configurações de CBs derivadas da estrutura *dogbone*: com portas compartilhadas (a), com portas separadas (b), com pares de transistores pequenos extras (c), com número desigual de transistores **n** e **p** (d) e especializada (e).

Aqui, canal é a região não ocupada por transistores. Quando utilizado fora do contexto da macroarquitetura, este canal fisicamente definido será referido por **canal explícito**. No presente trabalho, adotar-se-á a seguinte convenção: a existência de canais, ainda que restritos a porções da matriz, caracteriza-a como macroarquitetura com canal.

A granularidade da macroarquitetura visa definir a distribuição das CBs na composição da matriz. A classificação baseia-se no número de tipos de CBs diferentes que são utilizadas e como estas estão distribuídas. Logo, quanto à granularidade, uma macroarquitetura pode ser:

- uniforme;
- em blocos.

A macroarquitetura uniforme ou é constituída por somente um tipo de CB, ou as CBs estão distribuídas de modo uniforme. No último caso, apesar de haver mais de um tipo de CB, não é possível identificarem-se regiões diferenciadas dentro de uma mesma matriz.

A figura 2.3 mostra algumas macroarquiteturas. Quanto à roteabilidade, as matrizes da figura 2.3a, b e d são com canal, ao passo que a matriz de 2.3c é sem canal. Quanto à granularidade, a matriz de 2.3d apresenta estrutura em blocos, enquanto que as demais são uniformes.[2]

2.1.2 Conceitos fundamentais para estratégias de ocupação

O projeto de CIs com matrizes pré-difundidas pode ser simplistamente divido em duas etapas: **configuração da lógica** e **roteamento global**.

Ao conjunto de conexões que personalizam uma determinada região da matriz com uma lógica desejada, chamaremos **célula ou macrocélula lógica**.

Das duas etapas citadas, o roteamento é aquela que mais apresenta estratégias distintas. Estas estratégias, sem dúvida nenhuma, são o fruto, principalmente, do avanço acelerado da tecnologia CMOS. Aqui, o termo roteamento pode conduzir a uma idéia muito restrita, qual seja, conexões somente entre as células lógicas de qualquer complexidade. Então, consideremos genericamente o termo conexão como sendo a ligação entre quaisquer elementos, a ser realizada durante a etapa de personalização. Assim sendo, existem três estratégias para se realizarem

[2]Assumiu-se que a matriz da figura 2.3a possui um único tipo de CB devido a simetria das bandas, uma vez que o grau de detalhamento do desenho não permite uma identificação clara das CBs.

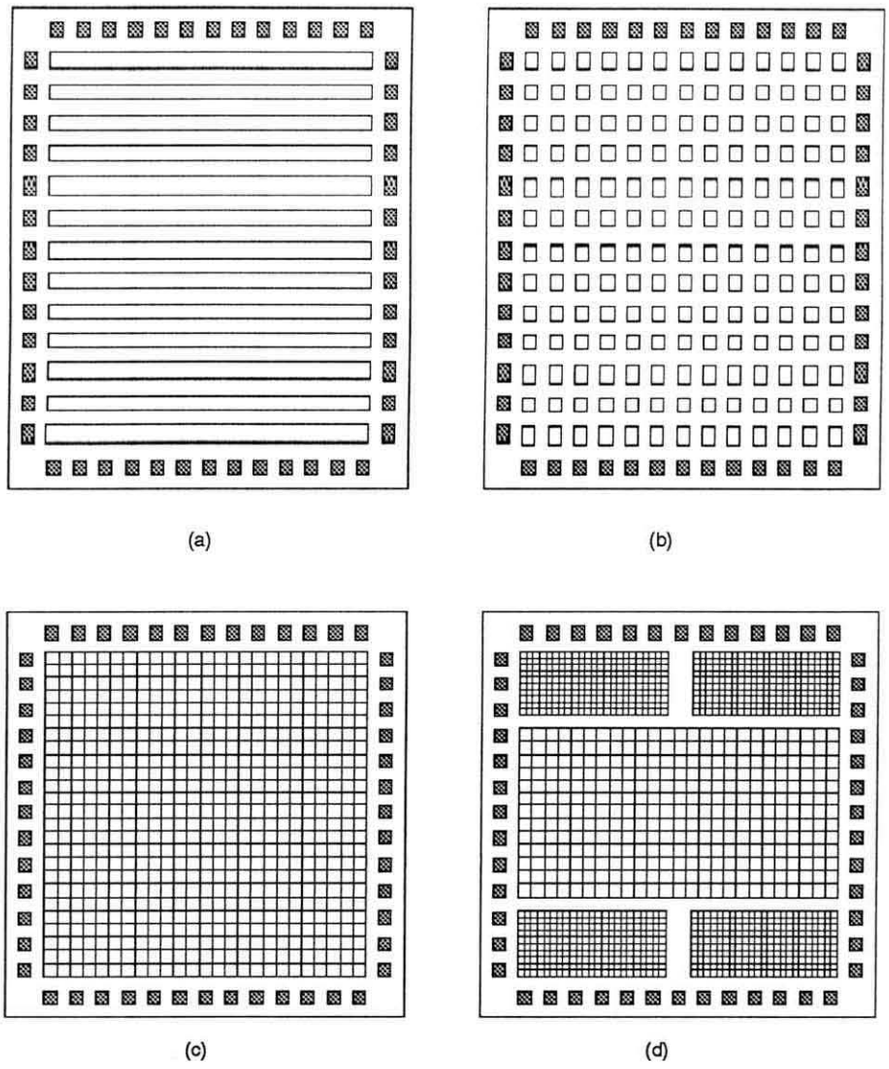

Figura 2.3: Algumas macroarquiteturas de matrizes pré-difundidas.

conexões numa matriz pré-difundida:

- em **canais explícitos**;
- em **canais dinâmicos**;
- e **compartilhado** com as células lógicas.

A figura 2.4 ilustra tais estratégias. No primeiro caso, a macroarquitetura deve apresentar canais explícitos (figura 2.4a), dentro dos quais podem existir segmentos pré-realizados em polissilício, difusão ou até mesmo em metal, a fim de facilitar o roteamento. Esta estratégia é encontrada nos primeiros pré-difundidos,

onde geralmente só havia um nível de metal para a realização de todo o roteamento

Na estratégia com canais dinâmicos, a macroarquitetura não apresenta canais explícitos. Estes são criados sobre bandas de CBs, inutilizando-as para a realização de lógica. Há duas variações desta estratégia. Na primeira, bandas de CBs são ocupadas alternadamente por lógica e por roteamente [NOI 85], confome mostra a figura 2.4b. A segunda variação é alocar bandas ou partes de bandas, conforme a quantidade de conexões a serem feitas [TAK 85][KUR 87](figura 2.4c). Obviamente, é necessário que a microarquitetura tenha sido projetada de modo que seja possível alocar apenas parte de uma banda de CBs. Independente da variação, esta foi a primeira estratégia de roteamento adotada nas arquiteturas sem canal, genericamente conhecidas por *sea-of-gates*. Talvez pelo fato de não existirem então ferramentas de PAC comercialmente disponíveis, capazes de explorar com sucesso o aumento de flexibilidade da macroarquitetura, o projeto era realizado com as mesmas ferramentas utilizadas nos projetos com *standard cells* tradicionais, tal como roteadores de canal, resultando em baixas taxas de ocupação.

A terceira estratégia não utiliza nem canais explícitos, nem canais dinâmicos. O roteamento é feito sobre as CBs, fazendo uso das trilhas restantes após o posicionamento das células lógicas, equivalendo ao *over-the-cell routing* dos pré-difundidos. Esta estratégia pode apresentar limitações graves, uma vez que o número de trilhas necessárias para o roteamento global aumenta com o aumento da complexidade do circuito. Para prover mais trilhas livres para conexões globais, a altura das bandas deve ser aumentada. Uma estratégia possível é aumentar a largura dos transistores, o que reflete num aumento da carga capacitiva, influenciando negativamente na freqüência de funcionamento do circuito. Em resumo, embora não haja um limite para a altura das bandas de CBs, neste caso existe um valor ótimo, acima do qual não só o desempenho elétrico fica comprometido, mas também a densidade de transistores disponíveis na matriz cai. Esta estratégia é ilustrada na figura 2.4d.

As três estratégias expostas são bastante características. No entanto,

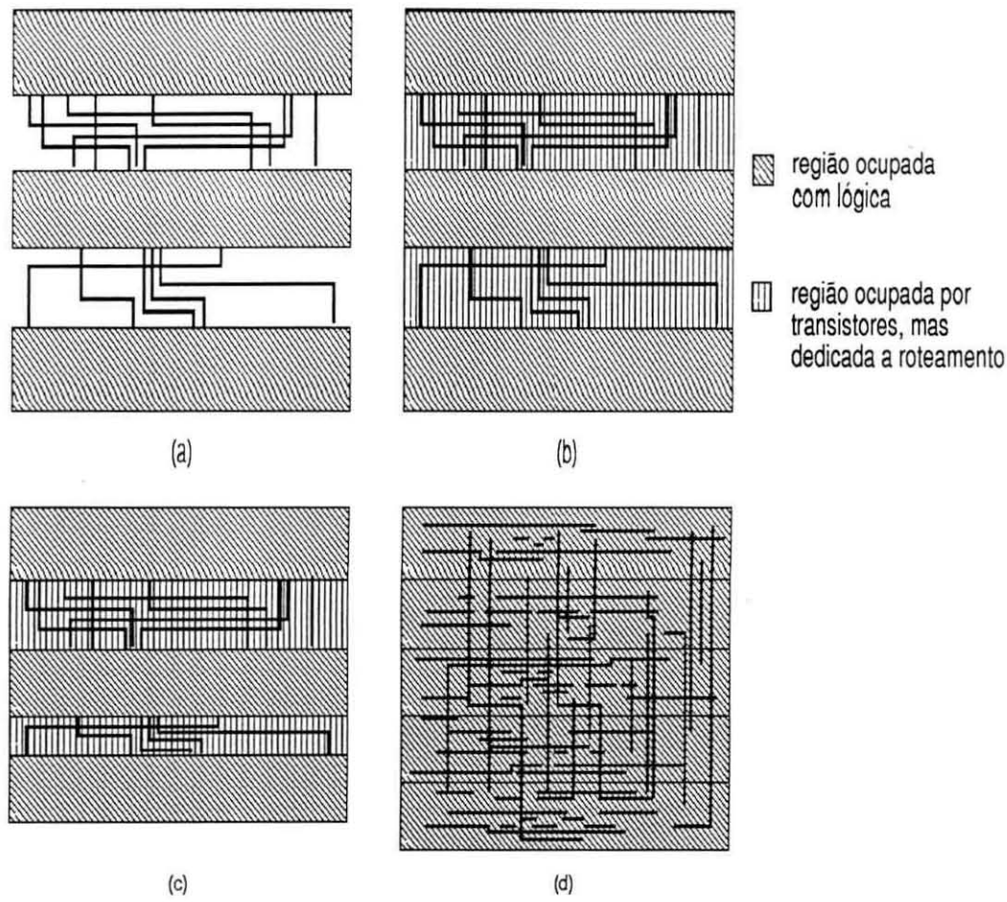

Figura 2.4: Estratégias para a realização de conexões em matrizes pré-difundidas.

muitas variações podem ser encontradas. Por exemplo, nos primeiros *gate arrays*, mesmo as ligações entre células adjacentes eram realizadas nos canais explícitos. Já nas arquiteturas sem canal, ocorre outra estratégia: conectar célula adjacentes através de justaposição dos terminais (*abutment*), ou seja, com conexões sobre os transistores. Neste caso, os canais dinâmicos alocados são utilizados somente para roteamento global.

A partir da disponibilidade de mais de um nível de metal, a forma como tais níveis são utilizados também passa a influenciar a estratégia de roteamento. Por exemplo, muitas abordagens de pré-difundidos, mesmo as sem canal, utilizam uma única camada (a mais próxima do dispositivo, *i.e.*, metal 1) para a realização das células lógicas. O roteamento entre estas poderá tanto ser realizado somente com as demais camadas como com todas as camadas [AND 88].

Além do roteamento, a forma de personalizar a lógica da matriz é o outro

item de suma importância na estratégia de ocupação de matrizes pré-difundidas. Por muito tempo, a personalização de lógica foi feita mediante o uso de bibliotecas de padrões de personalização, os quais definem fisicamente uma área da matriz. Porém, a inclusão de macrocélulas de grande complexidade nessas bibliotecas resulta numa significativa diminuição da flexibilidade, pois regiões cada vez maiores da matriz ficam irrevogavelmente configuradas. Com o objetivo de aumentar o grau de liberdade para a alocação da matriz, tem sido desenvolvidas técnicas de geração automática dos padrões de personalização para macroblocos, a partir de bibliotecas procedurais que contêm apenas algumas primitivas. *A priori*, a simples decomposição de partes de circuitos em termos de primitivas tende a aumentar o grau de liberdade para a geração dos macroblocos [DUC 91]. Neste caso, é óbvio que os geradores de módulos devem tirar máximo proveito da arquitetura da matriz.

2.2 Abordagens de Pré-difundidos

Esta seção traz alguns exemplos de abordagens de pré-difundidos, as quais são agrupadas segundo a cronologia dos avanços tecnológicos:

Desde seu surgimento, tem sido notório o aumento das densidades de transistores nas matrizes pré-difundidas. No entanto, há de se distinguir claramente duas componentes neste progresso:

- aumento da densidade de transistores disponíveis;
- aumento da taxa de ocupação das matrizes.

Durante os primeiros anos, a primeira componente foi a predominante, principalmente devido a sua íntima relação com o *scaling down* da tecnologia CMOS e devido também aos avanços ocorridos na arquitetura (*e.g.* surgimento do isolamento por porta).

A segunda componente não deixa de ser uma decorrência da primeira, mas acima disso, é o reflexo de projetos mais cuidadosos, nos quais as arquiteturas são elaboradas para as estratégias de ocupação, e não *vice-versa*.

Durante essa evolução, surgiu toda uma terminologia para designar arquiteturas e respectivas estratégias de ocupação, a qual varia bastante de autor para autor. Desta forma, definir-se-á uma terminologia própria.

Embora o termo *gate array* seja comumente usado para designar qualquer estilo pré-difundido, no escopo desta dissertação ele será utilizado para referirmo-nos àquelas arquiteturas que apresentam canais explícitos de roteamento.

Sea-of-gates, por sua vez, referir-se-á às arquiteturas que não apresentam canais explícitos de roteamento. Se o isolamento entre quaisquer transistores das CBs que constituem tais arquiteturas for realizado por porta, então será aplicado o termo *compacted arrays*.

Se a macroarquitetura apresentar granularidade do tipo em blocos, então esta será referenciada por *structured array*.

2.2.1 Abordagens convencionais

A arquitetura dos primeiros *gate arrays* apresentavam canais explícitos em uma ou até em duas direções. Considerando-se que na tecnologia então disponível só havia um nível de metalização, esses canais eram a solução mais eficiente para realizarem-se todas as conexões. Por serem regiões totalmente livres de transistores, pontes em polissilício ou em difusão (*underpasses*) eram estrategicamente providas de modo a facilitar as trocas de trilhas e de direção. Também havia necessidade de proverem-se pontes entre as CBs (*feedthroughs*), de modo a permitir que sinais trocassem de canais, cruzando bandas. A figura 2.5 ilustra a topologia de uma macroarquitetura *gate array* com canais horizontais.

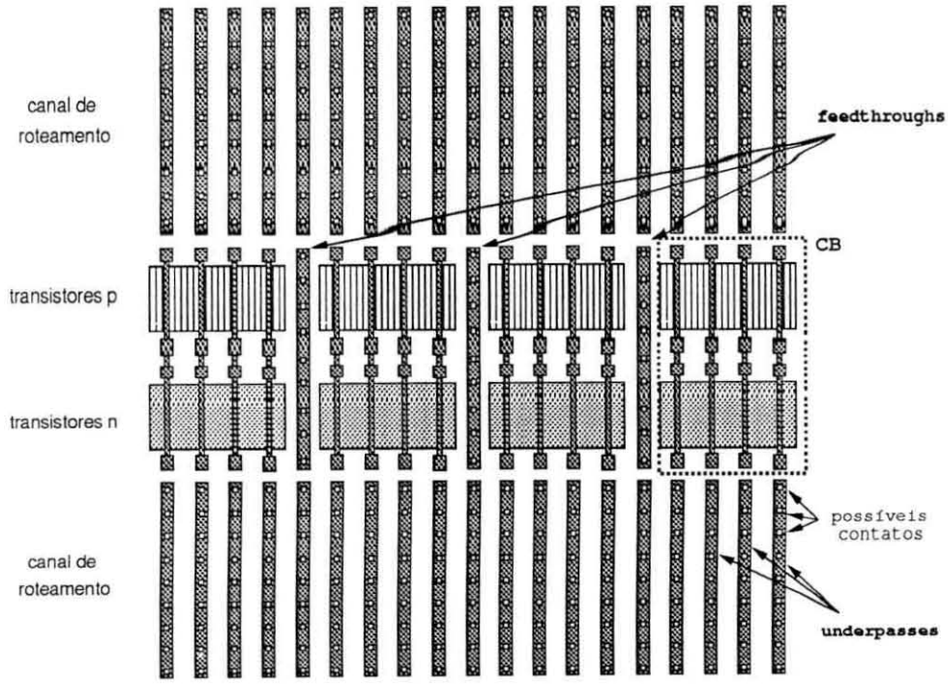

Figura 2.5: Macroarquitetura de um *gate array* típico, em tecnologia de um nível de metal.

Note-se que as CBs contendo quatro pares de transistores são isoladas geometricamente. Considerando-se a utilização de bibliotecas de células e macrocélulas lógicas implementadas somente na direção horizontal, há dois aspectos a serem considerados quanto à ocupação das CBs:

- para células lógicas que utilizem menos de quatro pares de transistores, uma CB é suficiente, resultando uma subutilização inerente à topologia;
- na implementação de macrocélulas cujo padrão é proveniente da biblioteca (fixo, portanto), a densidade de transistores ainda é pobre, pois há espaços sem transistores entre cada CB.

O posicionamento das células tendia a ser bastante delicado. A preocupação principal era não permitir conexões muito longas na direção vertical, uma vez que o atraso introduzido pelos segmentos em polissilício poderia comprometer o desempenho elétrico do circuito.

Com o advento de tecnologias com dois níveis de metal, não há mais a necessidade de *underpasses* e *feedthroughs*. Apesar do aumento da flexibilidade

proporcionada pelo avanço da tecnologia, num primeiro momento a estrutura de bandas de CBs intercaladas com canais explícitos de roteamento foi mantida. O motivo para tal conservadorismo era a possibilidade de utilização das mesmas ferramentas de PAC então disponíveis para projetos com *standard cells*, tais como posicionadores e roteadores de canal. Tais ferramentas começavam a surgir em grande profusão, sendo relativamente baratas e seguras (principalmente devido a sua simplicidade). Por outro lado, tal aproximação não permitia a exploração das características próprias dos pré-difundidos. Por exemplo, para a adequação dos leiautes *gate array* às ferramentas *standard cells*, todas entradas e saídas eram duplamente definidas nas bordas superiores e inferiores. Mesmo as conexões entre células adjacentes eram realizadas pelos canais, resultando num gasto desnecessário de trilhas. A figura 2.6 mostra a topologia de um *gate array* para dois níveis de metal, com isolamento geométrico. As CBs são constituídas por quatro pares **n-p**, com portas compartilhadas. O compartilhamento de portas reduz o número de conexões a serem feitas, caso só se implemente células em lógica complementar. Esta arquitetura apresenta as mesmas desvantagens que a anteriormente detalhada, no tocante à ocupação.

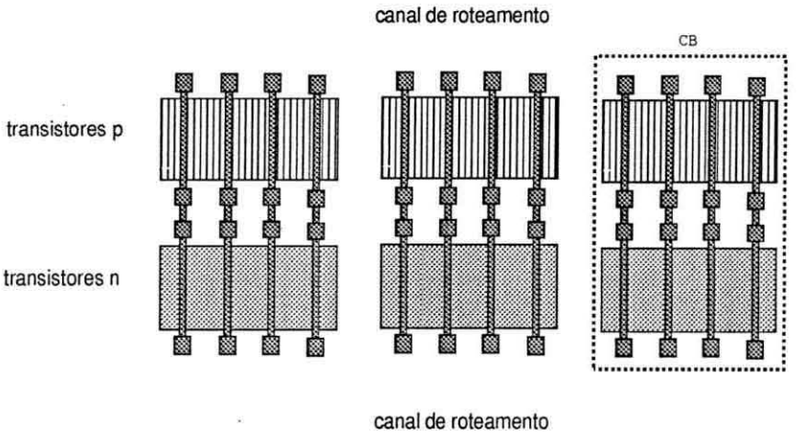

Figura 2.6: Topologia de um *gate array* em dois níveis de metal, com isolamento geométrico.

A aplicação da técnica de isolamento por porta permite que os transistores de um mesmo tipo compartilhem regiões de fonte e dreno, conforme mostra a figura 2.7. Com isto, o ajuste das células lógicas é mais fino, uma vez que a

CB é composta por apenas um par n-p. Isso reduz o desperdício de transistores. Por outro lado, a complexidade do roteamento interno às bandas aumenta, pois é necessário conectarem-se as portas dos pares complementares.

Como estratégia de utilização dos níveis de metal disponíveis, geralmente os padrões das células lógicas estão no primeiro nível de metal, por ser o mais próximo do polissilício e da difusão. Para roteamento nos canais, cada nível de metal é utilizado para a implementação de conexões numa direção, ficando o segundo nível reservado para cruzar as bandas de CBs. Outras estratégias também podem ser aplicadas, porém devendo estar estas em perfeita consonância com a capacidade das ferramentas de PAC disponíveis.

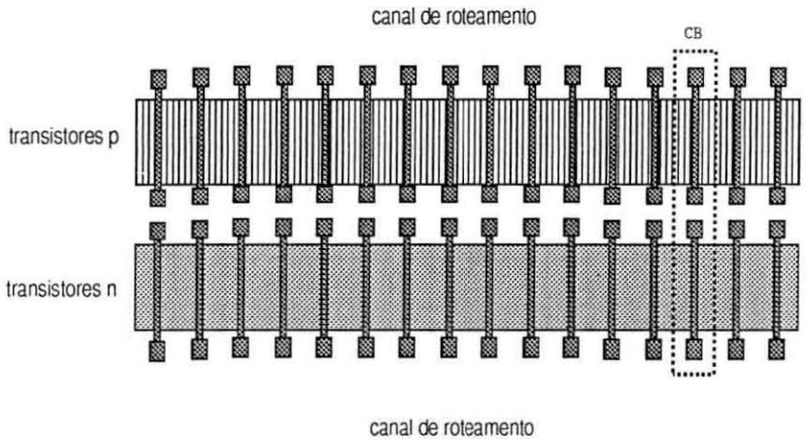

Figura 2.7: Topologia de um *gate array* em dois níveis de metal, com isolamento por porta.

2.2.2 Abordagens avançadas

Assim que as tecnologias de dois níveis de metal passaram a ser usadas nos pré-difundidos, profundas modificações ocorreram não só nas arquiteturas, mas também nas estratégias de ocupação.

A principal modificação arquitetural foi a supressão de canais explícitos de roteamento, surgindo o termo *sea-of-gates* para designar tais arquiteturas. Aí,

os pré-difundidos passam a apresentar características cada vez mais diferenciadas dos demais estilos de projeto, forçando o desenvolvimento de ferramentas de PAC dedicadas e estratégias direcionadas.

Em [NOI 85] é apresentada uma abordagem do tipo *compacted array*, com bandas de CBs alternadamente espelhadas. Na CB, os transistores **n** e **p** possuem mesmo comprimento de canal, comportando cada um três trilhas em metal 1, conforme ilustra a figura 2.8. As ligações intracelulares utilizam somente o primeiro nível de metal. O número de trilhas é suficiente para implementação das conexões internas, mesmo para macrocélulas. Porém, trilhas não utilizadas podem ser alocadas para conexões intercelulares. A estratégia de roteamento global consiste em alocar bandas de CBs para canais de roteamento, onde o número de bandas a serem alocadas dependerá da necessidade. O fato dos transistores terem o menor comprimento possível proporciona uma certa discretização na alocação dos canais, de modo a amenizar a subutilização destes.

O objetivo principal da arquitetura apresentada é permitir a implementação de lógica dinâmica em estruturas pré-difundidas. É relatada uma economia de área de até 50% em relação à lógica complementar, com o uso da técnica Nora [GON 83].

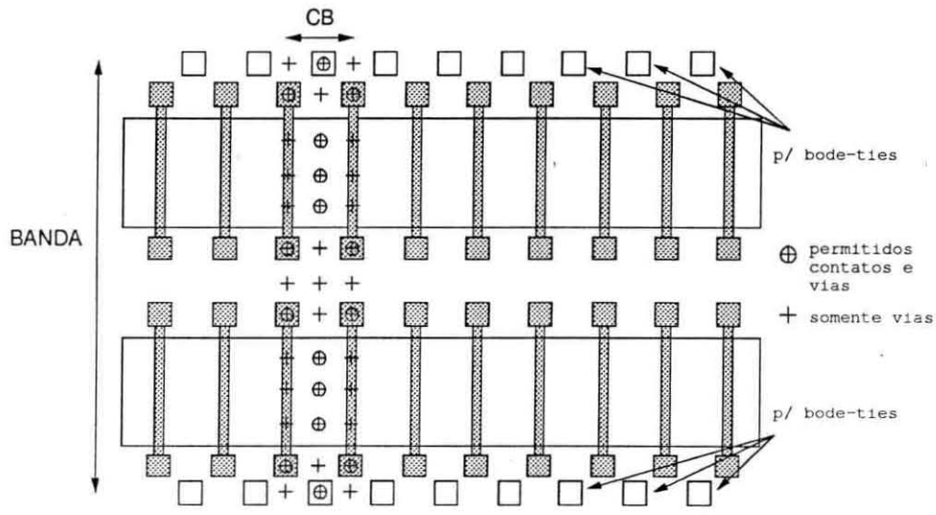

Figura 2.8: Topologia de um *compacted array* típico, apresentado em [NOI 85].

A alocação de canais de roteamento sobre as matrizes é um tópico que

tem sido exaustivamente pesquisado. Diversas estratégias tem sido propostas com o intuito de minimizar o desperdício de área, quando da alocação de canais.

As duas estratégias extremas de alocação são:

- alocar bandas inteiras para roteamento, assumindo o ônus de uma possível subutilização, ou;
- realizar todas as conexões sem alocar canais, misturando conexões intra e intercelulares, e correr o risco de não ser possível realizar todas as conexões.

A subutilização do canal alocado, referida na primeira estratégia, pode ser amenizada fazendo-se transistores pequenos. Porém, isso pode inviabilizar a realização de todas as conexões de macrocélulas sem a utilização do canal. Também pode ocorrer da capacidade de *drive* dos transistores não ser suficiente para carregar as conexões mais longas. Com relação ao outro extremo, o aumento do tamanho dos transistores provê mais trilhas livres a serem utilizadas para roteamento global. Mas o aumento inadvertido diminui a densidade da matriz e aumenta as cargas capacitivas a serem carregadas, conforme já descrito anteriormente. Então, a altura das bandas deve resultar de um compromisso entre espaço para o roteamento intracelular e fineza no ajuste do número de trilhas dos canais.

Em [SAI 85] é apresentada uma estratégia de roteamento a qual pode ser classificada como **compartilhada**. Utilizando-se de uma tecnologia com três níveis de metal, as conexões intracelulares são realizadas em metal 1, no centro da CB. Assim, as regiões ativas restantes e a zona entre duas colunas de CBs formam uma espécie de canal implícito, onde as conexões intercelulares são realizadas em metal 2 e metal 3 (figura 2.9).

Kuramitsu et alii relatam a não utilização de 38,5% das trilhas existentes nos canais dos *gate arrays*. Já para os *compacted arrays* com alocação flexível de canal, foi verificado que em média 49% das trilhas dos canais alocados não eram usadas.

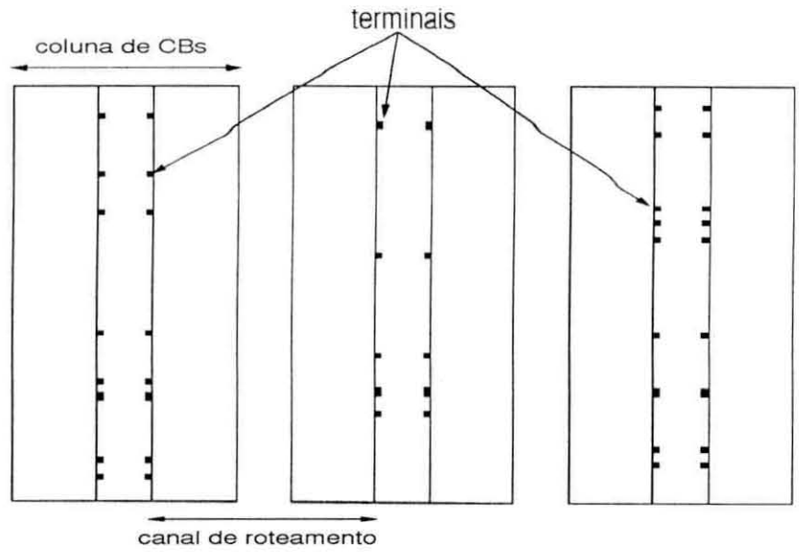

Figura 2.9: Estratégia de roteamento do tipo **compartilhado**, com alocação de canais implícitos sobre áreas ativas das CBs.

A fim de diminuir a subutilização dos canais e o desperdício de área associado, é proposta uma CB composta por um transistor **p** capaz de comportar quatro trilhas e dois transistores **n**, capazes de comportar três trilhas cada, conforme a figura 2.10a. Para células lógicas complexas, pares **p-n** são usados para lógica e transistores **n** isolados são usados como chaves. Sobre os transistores **p** ou **n** não utilizados são realizadas conexões internas. No caso de células de pequena complexidade, somente pares **n-p** são usados e a tira de transistores **n** restante pode ser alocada como canal de roteamento. Caso seja necessário, duas tiras adjacentes de transistores **n** podem ser alocadas, espelhando-se a célula lógica. Este ajuste mais fino na alocação de canais é mostrado na figura 2.10b.

Outra técnica para ajuste fino de trilhas para o roteamento em matrizes *sea-of-gates* é proposta em [OKU 89]. Uma CB contendo nove pares **n-p** é definida (figura 2.11a). Pares com portas compartilhadas são alternados com pares de portas separadas e cada transistor oferece três posições possíveis para colocação de contatos. O roteamento intracelular é realizado com primeiro nível de metal. Para a acomodação das células lógicas, as CBs vão sendo empilhadas, à medida que vão sendo necessárias, formando uma estrutura em forma de coluna (daí, a estratégia ser denominada *column macro-cell*). O alinhamento das colunas permite que canais de roteamento sejam discretizados no sentido do comprimento dos canais dos tran-

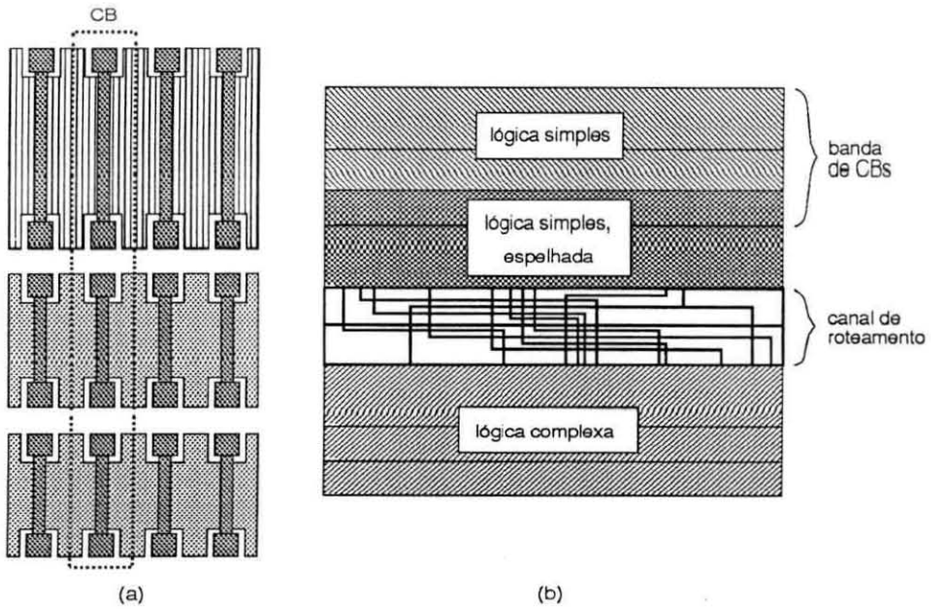

Figura 2.10: Arquitetura (a) e estratégia (b) para *sea-of-gates* do tipo *compacted array* apresentada em [KUR 85].

sistores, e não mais no sentido da largura. Como o leiaute da CB é projetado de modo a acomodar somente uma trilha no sentido da largura do canal do transistor, o ajuste do canal de roteamento alocado é exato, sem desperdício de trilhas. A figura 2.11b elucida esta estratégia de ocupação. Em [FRE 91] também é apresentada uma estratégia de ocupação em forma de coluna, porém a topologia da CB é um pouco diferente.

A impossibilidade de realização de lógica aleatória e blocos de memória de forma eficiente numa mesma matriz é um fator que restringia a utilização do estilo pré-difundido para a implementação de ASICs. Por isso, desde cedo muitos esforços foram focalizados na pesquisa de arquiteturas nas quais a implementação de blocos RAM e ROM não resultassem numa baixa taxa de ocupação da matriz. Notadamente, duas tendências surgiram:

- projetar uma CB que suportasse tanto lógica aleatória como memórias RAM e ROM e formar a matriz a partir desta única CB, ou
- projetar mais de uma CB, otimizando cada qual para uma aplicação específica. A matriz seria formada por uma arquitetura em blocos, cada bloco sendo composto por um único tipo de CB.

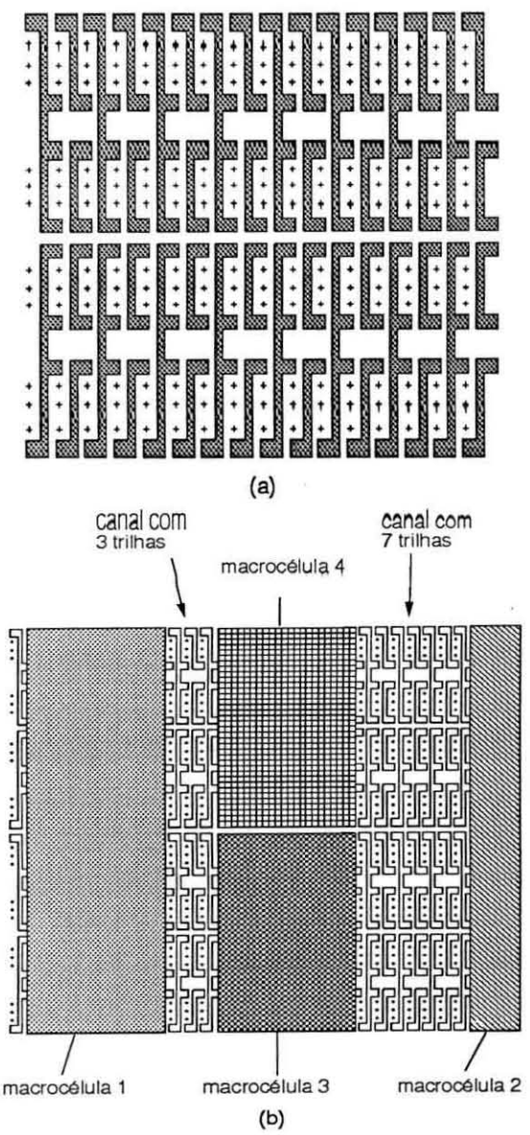

Figura 2.11: Arquitetura (a) e estratégia de ocupação (b) da abordagem *column macro-cell* [OKU 89].

Para a implementação de CBs dedicadas a memórias, transistores com dimensões distintas são necessários para prover bom desempenho elétrico [TAK 85]. Especificamente na primeira tendência, onde esta diferenciação se dá no nível da microarquitetura, pode haver certa perda de área associada à não utilização de todos os transistores. Porém, a assimetria da topologia pode ser aproveitada na discretização da alocação de trilhas para o roteamento. Isto significa que o suposto desperdício de área pode transformar-se numa forma eficaz de ajustar o tamanho dos canais de roteamento. Apenas a granularidade deste ajuste não vai ser tão pequena quanto àquela da estrutura em coluna.

A estratégia descrita anteriormente é amplamente referenciada na bibliografia. Um exemplo clássico é provido por Takahashi et alii [TAK 85]. A topologia da CB foi desenvolvida para acomodar um bit de memória SRAM com oito transistores e ainda ser apropriada para implementação de lógica. Conforme é visto na figura 2.12a, a CB é composta por dois pares **n-p** de transistores grandes isolados geometricamente, com portas separadas e cuja largura dos canais possui orientação horizontal. Há também dois pares **n-p** de transistores pequenos com fontes e drenos compartilhados aos pares e portas separadas. Tal configuração permite que blocos RAM e ROM sejam criados em qualquer parte da matriz, misturando-se com lógica aleatória. No caso de SRAM com oito transistores, os dois inversores são realizados com os transistores pequenos, ao passo que as chaves que conectam o bit aos barramentos de escrita e leitura são feitas com transistores grandes, conforme mostra a figura 2.12b.

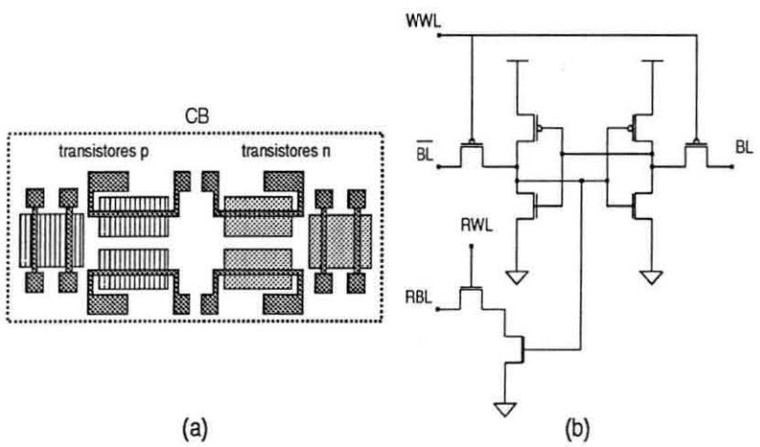

Figura 2.12: Uma CB para implementação eficiente de memória e lógica aleatória [TAK 85] (a) e esquema de um bit de memória SRAM com oito transistores (b).

Na implementação de lógica aleatória, são utilizados os transistores grandes e a região dos transistores pequenos pode ser usada para roteamento. Porém, no caso de células lógicas de grande complexidade, tais como somadores e registradores de deslocamento, normalmente são usados ambos tamanhos de transistores, de modo a aumentar a taxa de ocupação. Na alocação de canais de roteamento, o ajuste no número de trilhas se dá em capacidades fixas de 10, 24, 34, 48 etc, conforme forem alocadas somente regiões sobre transistores pequenos, coluna de CBs

ou ainda combinações de ambos. Isto é mostrado na figura 2.13.

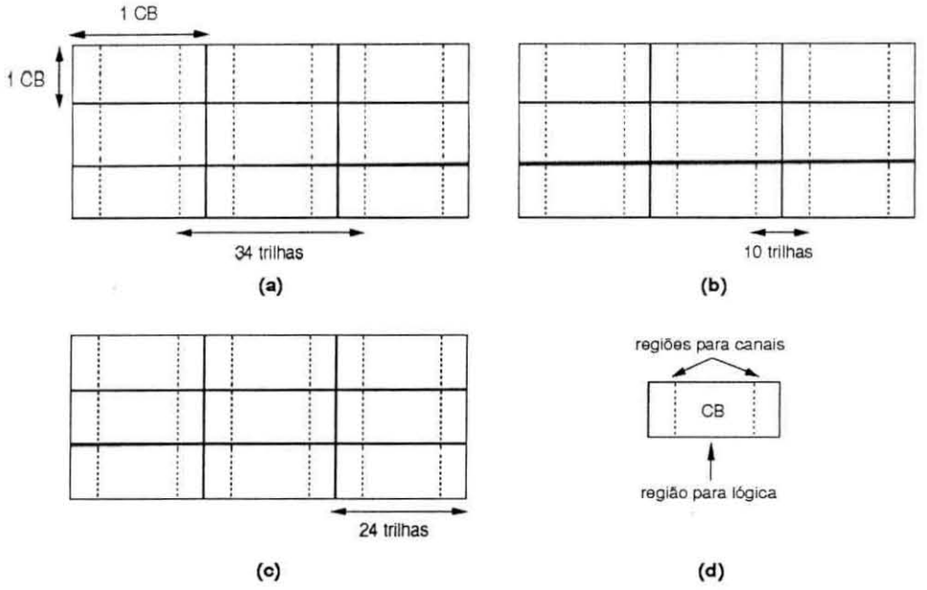

Figura 2.13: Discretização de trilhas na alocação de canais de roteamento para a matriz de [TAK 85], quando da implementação de lógica aleatória.

A preocupação de permitir a alocação de memória RAM em qualquer parte da matriz também está presente na CB **GME** da abordagem Cipredi [CAL 88a], que utiliza tecnologia 2.0μm. Porém, sua topologia apresenta quatro pares de transistores grandes com portas separadas e dois pares de transistores pequenos, com portas compartilhadas, todos com a mesma orientação. Os transistores pequenos utilizam isolamento geométrico e estão posicionados entre cada banda de transistores grandes, conforme é mostrado na figura 2.14 (à esquerda). O posicionamento e a orientação destes transistores pequenos, no entanto, ocasiona um mau aproveitamento da superfície de silício, o qual torna-se mais acentuado quando estes não são utilizados. Na implementação de lógica aleatória ou pelo simples uso dos padrões de personalização contidos na biblioteca [CAL 88b], a taxa de ocupação pode atingir valores tão baixos quanto 0,48 [FRE 89].

Uma segunda versão da CB **GME**, em tecnologia 1.2μm, apresenta apenas transistores grandes [PAI 91] (figura 2.14, à direita). Estes ainda foram redimensionados de modo a acomodar maior número de trilhas, permitindo roteamento global compartilhado com as células lógicas, evitando ao máximo a alocação de bandas para a realização de roteamento. Os transistores **p** possuem canal mais longo do

que os n, numa tentativa de aproximar os tempos de subida e descida do inversor.

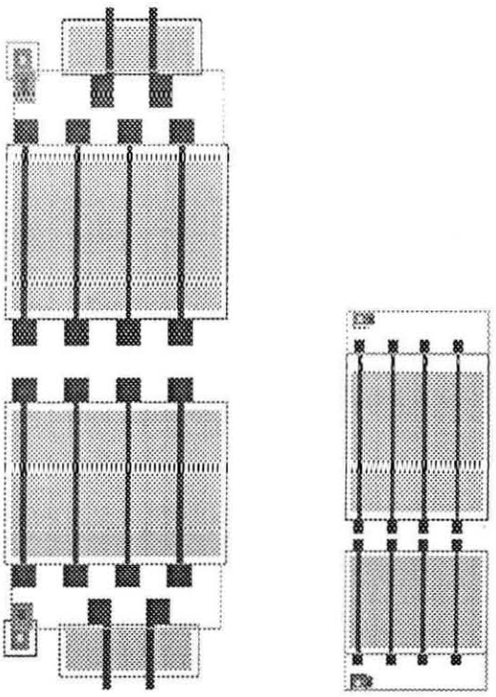

Figura 2.14: CBs da abordagem Cipredi: em tecnologia 2.0μm (esq.) e segunda versão, em tecnologia 1.2μm (dir.).

A abordagem **Gate Forest** destaca-se das anteriormente mostradas por priorizar não a arquitetura, mas a estratégia de projeto, dando grande ênfase à hierarquia e ao planejamento topológico [BEU 88a]. No entanto, grande parte da personalização é baseada no uso de bibliotecas de padrões. A diferença básica é que são providas mais opções:

- biblioteca de padrões em lógica complementar;
- compiladores de módulos RAM, ROM, multiplicadores etc;
- biblioteca de padrões em lógica dinâmica, segundo várias técnicas;
- e implementação manual de padrões, caso o usuário desejar.

Para permitir tantas opções, foi projetada uma CB com boa flexibilidade tanto na topologia como no tipo e proporção dos transistores. A figura 2.15 mostra a topologia da CB *gate forest*, a qual é composta por 2 transistores **p**, 4 transistores

n grandes e 2 transistores **n** pequenos, onde a relação de tamanho é 2:4:1. Esta assimetria e proporção entre tipos diferentes visa facilitar tanto a implementação de lógica dinâmica, quanto de memória. No projeto da CB, foi maximizada a conectibilidade das células, ao invés de apenas minimizar a área de silício ocupada, como geralmente ocorre nos demais *sea-of-gates*. Como reflexo desta preocupação, transistores com orientação a 45 graus são providos, de modo a aumentar o número de opções de terminais em trilhas diferentes, sem que seja necessário realizar alguma conexão em metal para isso [BEU 88b]. Também há uma certa **transparência** nas células, permitindo que sinais sejam roteados juntamente com as conexões intracelulares. Quando da não utilização dos transistores **n** pequenos, a região por estes ocupada pode ser usada para roteamento.

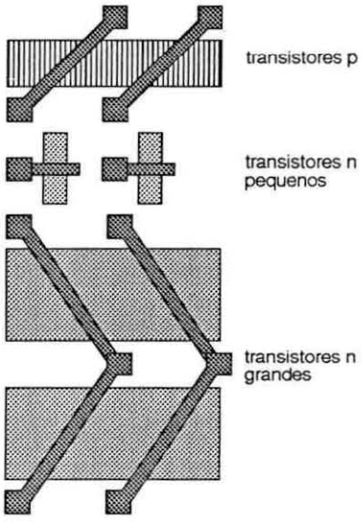

Figura 2.15: CB utilizada na abordagem *gate forest* [BEU 88b].

A macroarquitetura pode ser classificada como uniforme, sem canais, uma vez que a matriz é composta por somente um tipo de CB, numa configuração tipicamente *sea-of-gates*.

Apesar do grande número de arquiteturas avançadas que tem sido desenvolvidas, na prática, as estratégias de projeto não parecem ter evoluído: na maioria das abordagens, a personalização ainda é realizada com o uso de bibliotecas de padrões e o roteamento global é feito com o uso de canais, alocados sobre transis-

tores das matrizes. Duchene et alii apresentam uma estratégia de projeto com pré-difundidos que visa abolir o uso das bibliotecas de padrões e ainda explorar melhor a estrutura *compacted array*, realizando o roteamento sobre as células, sem alocar canais [DUC 91]. A técnica trata o circuito de forma global, a partir de um nível superior de abstração, particionando-o em blocos, cujos tamanhos, formatos e localizações são determinadas por uma ferramenta de planejamento topológico. Após, cada bloco será gerado automaticamente, conforme as seguintes características:

- as células são geradas no nível de transistor;
- o roteamento entre células vizinhas é realizado por justaposição dos terminais;
- as conexões restantes são realizadas sobre as células.

Somente após a geração de todos os blocos, é que serão alocados canais verticais e horizontais sobre a matriz, para o roteamento entre estes. Quanto à arquitetura, é utilizada uma matriz *compacted array* com transistores pequenos, sem qualquer alteração especial. O segredo do bom aproveitamento da área reside na estratégia propriamente dita. Os blocos, ou subcircuitos, podem conter tipicamente de 50 a 800 transistores e taxas de ocupação entre 0,60 e 0,85 são relatadas.

A alocação de canais é adiada para o final do projeto, de modo que somente as conexões realmente difíceis serão feitas em trilhas especiais, o que tende a reduzir o número de trilhas necessárias. Também o planejamento topológico inicial permite uma avaliação das posições e ordem dos terminais de cada bloco, podendo-se controlar o congestionamento nos canais entre os blocos.

A abordagem Marcela também dá maior ênfase à estratégia de ocupação [GUN 91a][GUN 91c]. A arquitetura da matriz é composta por tipos de CBs diferentes, sendo cada uma específica para a implementação de uma função lógica simples. Um projeto a ser implementado nesta abordagem deve ser decomposto em termos de funções disponíveis, o que resulta num aumento da flexibilidade de

posicionamento.[3]. Com isso, é a ferramenta de síntese que vai decidir qual a melhor posição para as células que se originaram de um mesmo bloco funcional, evitando assim os congestionamentos locais decorrentes do uso de bibliotecas geométricas. Também no que diz respeito à arquitetura, há ganhos significativos, principalmente pela filosofia de projeto adotada, a qual prioriza a realização completa do roteamento. A abordagem Marcela será discutida com maior detalhamento no capítulo 4. A figura 2.16 apresenta uma unidade básica de uma matriz Marcela.

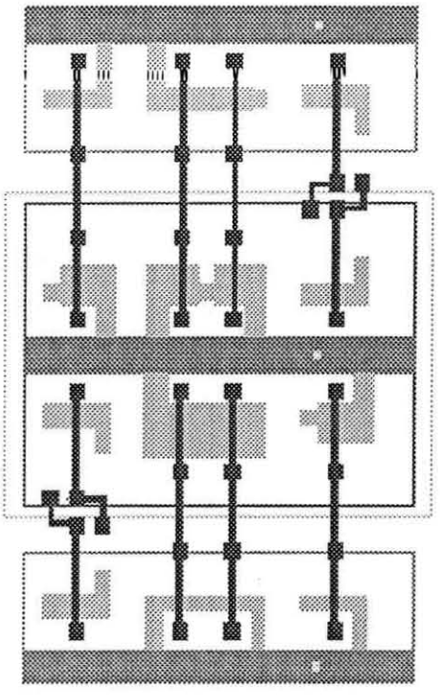

Figura 2.16: Unidade básica que compõe uma matriz da abordagem Marcela.

As abordagens que utilizam arquiteturas em blocos, referenciadas na bibliografia por *structured arrays*, não são muito freqüentes nem nos meios acadêmicos, nem nos meios comerciais. Miyahara et alii desenvolveram uma arquitetura orientada para aplicações em sistemas baseados em microprogramação [MIY 86], a qual constitui um exemplo clássico. Nos sistemas microprogramados, geralmente são necessários blocos RAM e ROM de alta capacidade, o que é difícil de ser obtido em matrizes pré-difundidas. Para que fosse possível implementar ROM e RAM numa matriz pré-difundida, alcançando maior densidade de bits, foi desenvolvida uma ar-

[3]Neste caso, assinalamento seria o termo mais apropriado

quitetura específica, composta por quatro blocos RAM de 1k cada e oito blocos ROM de 16k cada, posicionados ao redor de um bloco de 6000 CBs, organizadas numa estrutura com canais, para implementação de lógica aleatória. Estas CBs são formadas por dois pares n-p cada, isoladas geometricamente e com topologia do tipo *dogbone*. Entre os diversos blocos, existem canais explícitos de roteamento, uma vez que não é possível passarem-se trilhas de roteamento através dos blocos RAM e ROM. Os blocos RAM e ROM são constituídos por CBs específicas para cada caso. Se necessário, composições de blocos podem ser feitas para modificar o tamanho da palavra de memória. A figura 2.17 mostra a macroarquitetura.

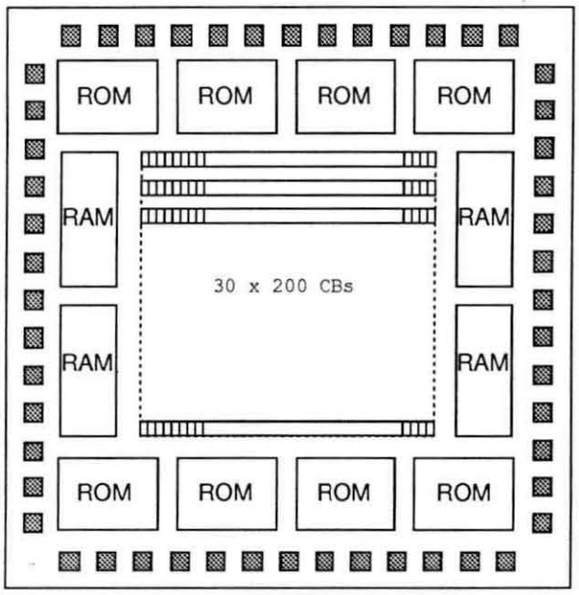

Figura 2.17: Macroarquitetura em blocos (*structured array*) para a implementação de sistemas microprogramados [MIY 86]

Há ainda *structured arrays* no mercado que oferecem blocos dedicados à implementação de multiplicadores, ULAs e até CPUs completas [MEY 89]. Altas taxas de ocupação de tais arquiteturas só ocorrem em alguns casos muito específicos, o que acaba restringindo sua utilização. Por isso, a viabilidade comercial desta abordagem depende da identificação de usuários dispostos a compartilhar as matrizes.

A última estratégia a ser comentada é a utilizada na matriz pré-difundida do Centro Tecnológico para Informática (CTI). Foi desenvolvida uma matriz em

tecnologia 1.5μm de dois níveis de metal, porém a personalização é realizada somente com o segundo nível, estando o primeiro já configurado [SIM 92]. O objetivo desta estratégia é permitir a personalização das matrizes com o equipamento que o CTI dispõe, aumentando a velocidade de prototipação e diminuindo seu custo. Para a personalização da matriz, devem ser realizadas poucas etapas de processo (cerca de 10% do total), as quais tendem a ser bem menos críticas que as anteriores.

A matriz é composta por 6260 pares de transistores, organizados em bandas, as quais são intercaladas com canais de roteamento. Os canais tem capacidade para 10 trilhas horizontais de conexão cada, e possuem pontes em metal 1 com vias posicionadas em pontos estratégicos, de modo a evitar os cruzamentos. A personalização da matriz é feita com o auxílio de uma biblioteca de padões, a qual possui 16 primitivas. A técnica de isolamento usada é a por porta. Para a realização automática do roteamento, é utilizado um roteador de canal comercial e após, é feita uma adaptação do resultado para o canal real, considerando as restrições impostas pelas pontes em metal1. A figura 2.18 mostra o leiaute de um *flip-flop* D com *set* e *reset* pertencente à biblioteca de células *gate array* do CTI.

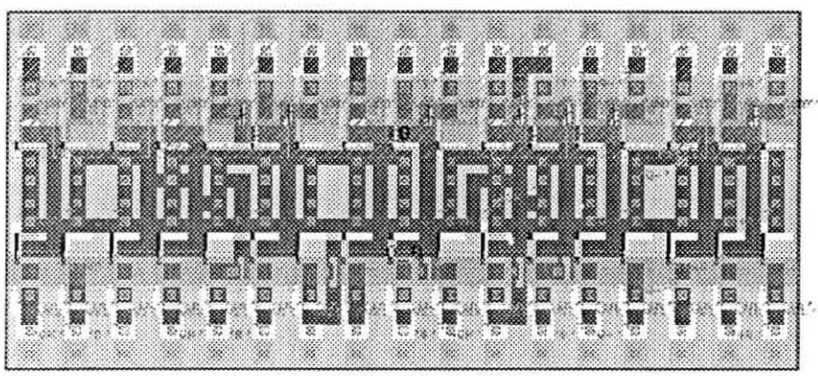

Figura 2.18: Leiaute de um *flip-flop* D com *set* e *reset* implementado na matriz *gate array* do CTI.

Apesar da baixa densidade de transistores disponíveis na matriz, a estratégia de utilizar apenas um nível de metal para personalização da matriz, confeccionada em tecnologia de dois níveis, permite a redução do custo do protótipo e permite que este seja implementado sem a necessidade de envio ao exterior. É uma

solução prática voltada para a realidade do mercado nacional de microeletrônica.

A tabela 2.1 resume características de algumas das abordagens avançadas discutidas nessa seção, dando uma visão mais ampla das diversas formas de melhorar o aproveitamento das matrizes pré-difundidas.[4]

abordagem	tecnologia	macroarq.	microarq.	densidade lóg. aleat. (trans./mm^2)	densidade memória (bits/mm^2)
van Noije (Leuven)	3.0μm 2 níveis de metal	*compacted array* 12k transistores	1 par n-p trans. iguais	880	-
Saigo (Toshiba)	2.0μm 3 níveis de metal	*sea-of-gates* 106k transistores	2 pares n-p Wp=2.1Wn	570	-
Kuramitsu (Mitsubishi)	1.3μm 2 níveis de metal	*compacted array* 540k transistores	2 trans. n 1 trans. p	1850	-
Okuno (Mitsubishi)	0.8μm 2 níveis de metal	*sea-of-gates* 1.4M transistores	9 pares n-p	6000	RAM=1900 ROM=6300
Takahashi (Fujitsu)	1.8μm 2 níveis de metal	*sea-of-gates* 240k transistores	2 pares n-p gr. 2 pares n-p peq.	920	RAM=230 ROM=1900
Cipredi (UFRGS)	2.0μm 2 níveis de metal	*sea-of-gates*	4 pares n-p gr. 2 pares n-p peq.	400	RAM=78
Gate Forest (IMS - Alemanha)	-	*sea-of-gates*	2 trans. p 4 trans. n gr. 2 trans. n peq.	-	-
Duchene (Lausanne)	2.0μm 2 níveis de metal	*compacted array*	1 par n-p	530	-
Marcela (UFRGS)	1.2μm 2 níveis de metal	*sea-of-cells* 4K transistores	4 tipos de CBs	1590	-
Miyahara (NEC)	1.6μm 2 níveis de metal	*structured array* 200k transistores	CBs específicas	700	RAM=650 ROM=12 000
IM/CTI (Campinas)	1.5μm 2 níveis de metal	*gate array* 12.5k transistores	1 par n-p	600	-

Tabela 2.1: Resumo das características das abordagens avançadas apresentadas. As densidades referem-se a transistores efetivamente utilizados nas matrizes.

[4]Com relação à abordagem desenvolvida no IM/CTI, a densidade refere-se à matriz não personalizada.

3 O AMBIENTE TRANCA

O desenvolvimento de ferramentas de síntese de leiaute para blocos em lógica aleatória é o objetivo principal do Projeto TRANCA, em andamento no GME/CPGCC da UFRGS. Como resultado deste esforço, há atualmente dois módulos de geração automática disponíveis, TRAMO [LUB 90] e TRAGO [MOR 90a], ambos rodando em ambiente DOS e em ambiente UNIX. Para cada ambiente, existe uma interface para gerenciamento destas ferramentas de síntese e das demais ferramentas de apoio, tais como extrator, planificador, conversores, exibidores de posicionamentos e editores de leiautes.

No ambiente DOS, as ferramentas TRANCA encontram-se integradas pelo gerenciador TENTOS [MOR 91]. Trata-se de uma interface orientada a menus, formada por uma janela principal e várias opções que abrem submenus do tipo *pop-up*. A interface gerencia a execução das ferramentas e dos diretórios. O diálogo entre as ferramentas é feito por intermédio de arquivos e os formatos de entrada e saída podem ser controlados pela interface.

No UNIX [SOU 91], a interface faz uso das facilidades providas pelo ambiente de janelas OPENWINDOWS, as quais provêem grande eficácia na interação com o usuário. O diálogo entre ferramentas também ocorre via arquivos, de forma transparente ao usuário.

As ferramentas de síntese automática de leiaute do Projeto TRANCA adotam subconjuntos de características propostas pela metodologia TRANCA. O subconjunto, por sua vez, depende dos aspectos práticos de cada abordagem. A seguir, serão abordadas as características de leiaute propostas pela metodologia TRANCA.

3.1 A Metodologia TRANCA

No esforço de automatização do processo de geração de leiautes de circuitos integrados, freqüentemente têm sido adotadas topologias simplificadas, de modo a facilitar a implementação de algoritmos e heurísticas capazes de tratar os problemas ligados ao posicionamento e roteamento das células. Porém, deste processo de simplificação decorre uma distância muito grande, em termos de qualidade, entre as soluções manuais (totalmente *full custom*) e as automáticas. De um modo geral, as soluções automáticas tendem a ser bastante piores do que as manuais, no que concerne à densidade de transistores.

A metodologia TRANCA (TRANsparent-Cell Approach) [REI 87] propõe um conjunto básico de procedimentos, os quais são o resultado da observação dos leiautes de blocos em lógica aleatória de diversos microprocessadores comerciais desenhados a mão [REI 83]. A utilização de tais procedimentos numa ferramenta de síntese automática equivale a incorporar parte da experiência de um projetista humano, o que tende a melhorar a qualidade dos leiautes gerados.

As características fundamentais da metodologia TRANCA são a utilização de uma estrutura de bandas, onde são inseridas as células provenientes de uma biblioteca, ou geradas automaticamente, e a realização do roteamento sobre os transistores (abordagem *over-the-cell routing*), segundo um esquema de alocação ordenada de trilhas. As vantagens decorrentes destas características são perceptíveis sob dois aspectos:

- Economia de área e
- Melhora do desempenho elétrico.

A supressão dos canais de roteamento implica numa redução apreciável de área. Conseqüentemente, o comprimento médio das conexões fica reduzido, contribuindo para melhorar o desempenho elétrico. Por outro lado, a topologia das

células deve suportar a passagem de um número conveniente de trilhas, a serem utilizadas nas conexões intercelulares. Com efeito, isto é considerado na metodologia TRANCA, conforme será visto adiante. Com a realização do roteamento sobre as células, a distância média entre as trilhas e os terminais das células é menor, e particularmente, conexões entre células vizinhas podem ser feitas por justaposição (*abutment*), resultando em caminhos mais curtos. Esta característica tem duplo efeito: proporciona melhor desempenho elétrico ao circuito e reduz a área ocupada pelo roteamento. Comparativamente aos *standard cells*, onde mesmo conexões entre células vizinhas são realizadas nos canais, os ganhos em termos de área e desempenho elétrico são apreciáveis. A figura 3.1 mostra uma conexão entre células vizinhas em *standard cells* e na metodologia TRANCA.

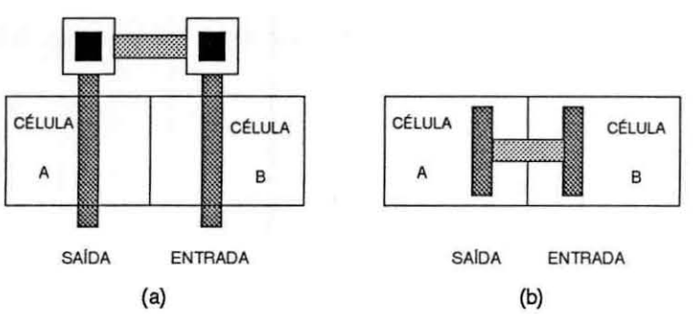

Figura 3.1: Conexão entre duas células vizinhas usando o canal (*standard cells*) (a) e por justaposição (TRANCA) (b).

Os procedimentos da metodologia TRANCA baseiam-se em quatro pontos [REI 87]:

- Estrutura de bandas;
- Maleabilidade;
- Transparência de células e blocos;
- Gerenciamento de trilhas.

Estes pontos objetivam nortear o projeto de circuitos integrados com o uso de células padrão (*cell-based*) de modo a produzir uma utilização mais eficiente da superfície de silício, sendo ainda passíveis de automatização.

3.1.1 Estrutura de bandas

Os barramentos de alimentação são definidos como dois pentes em metal 1, cujos dentes estão intercalados, conforme mostra figura 3.2. Cada banda é delimitada por um par de linhas de alimentação Vdd/Gnd, correndo na direção horizontal, sendo que a distância entre estas é constante a fim de permitir a inserção de células de altura fixa. As células, por sua vez, podem ser projetadas com altura correspondente a uma ou mais bandas, conforme sua complexidade ou restrições topológicas. Internamente a cada banda, é definido um conjunto de trilhas em metal 1, as quais serão utilizadas para realização das conexões intra e intercelulares. Tais trilhas possuem largura mínima e a distância entre duas trilhas adjacentes é igual ao passo de metal 1, o que permite apenas a colocação de contatos alternados (figura 3.3).

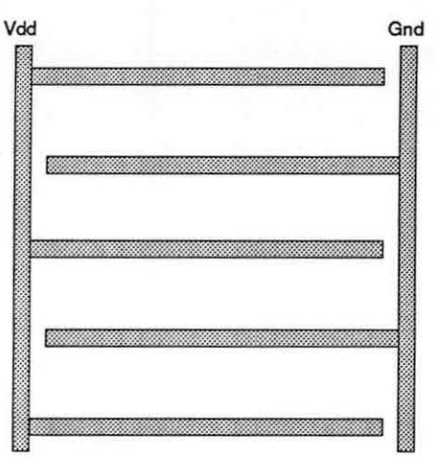

Figura 3.2: Barramento de alimentação que define a estrutura de bandas na metodologia TRANCA.

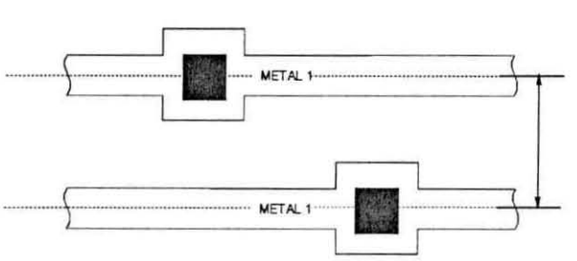

Figura 3.3: Passo de metal 1.

Para a definição do número de trilhas necessárias, foram considerados

estudos estatísticos sobre número de trilhas existentes sobre células de circuitos NMOS [REI 83]. Tais estudos apontaram o número médio de trilhas por banda como sendo igual a 7. Em [REI 87] são sugeridas 10 trilhas como sendo o número ótimo para circuitos CMOS, desde que para essa tecnologia há um acréscimo de conexões internas decorrente da ligação entre as redes **p** e **n**.

3.1.2 Maleabilidade

A maleabilidade diz respeito à capacidade que um bloco de células ou mesmo uma única célula tem de permitir a modificação de sua forma, mediante alteração topológica dos elementos internos. Blocos podem ser classificados como **duros** ou **moles**, de acordo com o grau de maleabilidade apresentado. Particularmente, blocos em lógica aleatória são **moles**, e por esse motivo tendem a ser tratados por último no planejamento topológico, na expectativa de que seus formatos se adeqúem às áreas ainda disponíveis [REI 85].

3.1.3 Transparência de células e blocos

Transparência de um bloco numa direção é definida como sendo a razão entre o número de trilhas que podem cruzar o bloco e o número total de trilhas na referida direção. A mesma definição é aplicável a células.

A implementação do primeiro protótipo baseado na metodologia TRANCA considerou o uso de um único nível de metal, o qual é associado às conexões horizontais. Então, a transparência horizontal refere-se ao número de trilhas disponíveis ao roteamento intercelular. Desde que conexões verticais devem ser feitas com o uso de difusão ou polissilício, a transparência vertical é praticamente nula, o que obriga a adoção de **células de interconexão** em polissilício toda a vez que for necessário atravessar uma banda ou simplesmente trocar de trilha. A figura 3.4

ilustra o uso de células de interconexão.

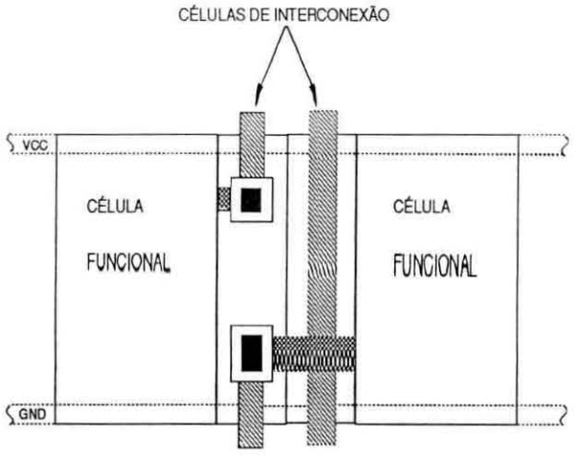

Figura 3.4: Uso de células de interconexão.

O uso de células de interconexão soluciona o problema da baixa transparência vertical, porém, provoca um aumento na área do circuito. Para tentar amenizar este problema, é necessário que as células sejam projetadas com dimensão horizontal mínima.

O desenvolvimento de células **transparentes** resulta num acréscimo de área destas células comparativamente às células **opacas** dos **standard cells**. Porém, a avaliação de consumo de área deve ser feita sobre o soma da área ocupada pelos transistores, mais a área ocupada pelo roteamento. Neste caso, desde que o número de trilhas por banda tenha sido convenientemente escolhido, o resultado é compensador. Os levantamentos estatísticos relatados em [REI 83] indicam que a transparência em circuitos *full custom* oscila entre 63% e 77%, independente do número de trilhas da banda. Na figura 3.5(esq.) vê-se uma versão para o leiaute de uma *nand* de três entradas, onde a transparência é 8. Na figura 3.5(dir.), a mesma célula foi projetada com altura mínima. Considerando-se a área ocupada por 8 trilhas de metal 1 num canal de roteamento adjacente a célula **opaca**, o acréscimo de área na solução **opaca** é de 21%.

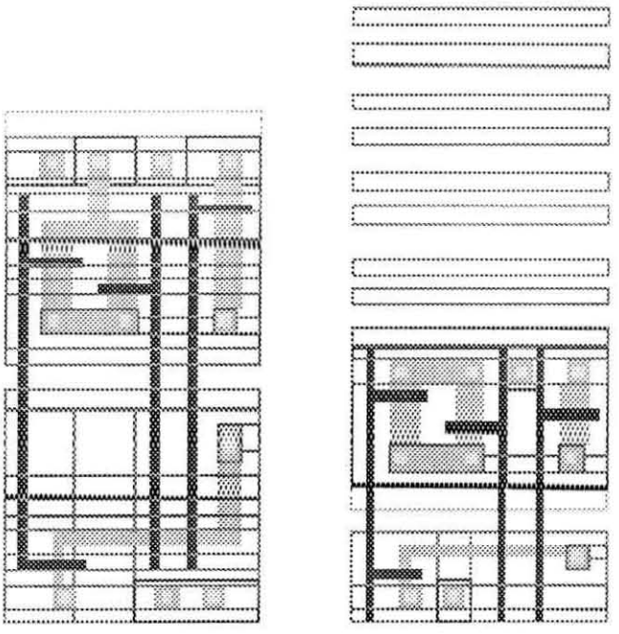

Figura 3.5: Versões **transparente** (esq.) e **opaca** (dir.) para o leiaute de uma *nand* de três entradas.

3.1.4 Gerenciamento de trilhas

De forma adversa dos *standard cells*, onde os canais vão crescendo à medida em que mais trilhas de roteamento são necessárias, na metodologia TRANCA, as trilhas constituem um recurso limitado, o qual deve ser coerentemente administrado. Para isso, TRANCA lança mão de um esquema de prioridades o qual reserva as trilhas de mais alta prioridade para as conexões internas e as de menor prioridade para as conexões externas. A atribuição do grau de prioridade às trilhas depende das restrições da tecnologia adotada, da topologia individual das células e sobretudo da política de realização do roteamento global. A adoção de um esquema de prioridades deficiente poderá forçar trocas de trilhas desnecessárias e o conseqüente aumento de área gerado pela inserção de uma célula de interconexão ou ainda conduzir à saturação das bandas, o que seria bem mais desastroso. Mesmo com o uso de um segundo nível de metal na direção vertical, há a necessidade de gerenciar o uso das trilhas. Neste caso, não haveria o acréscimo de área referente às células de interconexão, mas haveria desperdício de trilhas, o que também é grave. O fato é que sempre é preferível utilizar-se uma única trilha por sinal, independente do número de células que devam ser atravessadas. O uso de mais de uma trilha implica em

redução da transparência da banda como um todo. Isto é ilustrado na figura 3.6.

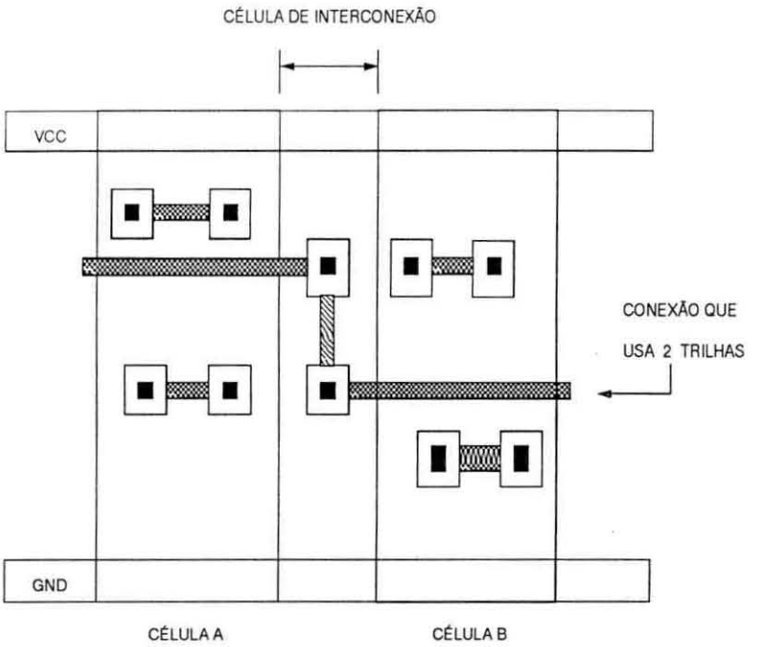

Figura 3.6: Efeitos do uso de mais de uma trilha por conexão: aumento da área devido ao uso de célula de interconexão e redução da transparência da banda.

Nos circuitos em tecnologia CMOS, a conexão entre as redes **p** e **n** são realizadas nas trilhas de maior prioridade. As trilhas mais internas às células tendem a ser as utilizadas para roteamento dos sinais globais, tais como relógio, *reset* etc [REI 83]. O esquema de prioridades apresentado na figura 3.7 é utilizado nas bibliotecas de células do Projeto TRANCA [MOR 90b][REI 90][CRU 91]. Para roteamento global, são alocadas as trilhas a partir de P0, enquanto que para as conexões internas são alocadas trilhas a partir de P9.

A adoção dos procedimentos anteriormente descritos, ainda que parcialmente, resulta em leiautes mais compactos. Mas sem dúvida, a maior vantagem é que tais procedimentos são perfeitamente passíveis de automatização. O subconjunto a ser adotado irá depender basicamente do estilo de projeto ao qual a ferramenta se destina. Resultados bastante promissores são relatados em [LUB 90] e [MOR 90a], onde são descritos e utilizados módulos geradores de leiautes para lógica aleatória. Tais módulos serão comentados brevemente nas próximas seções.

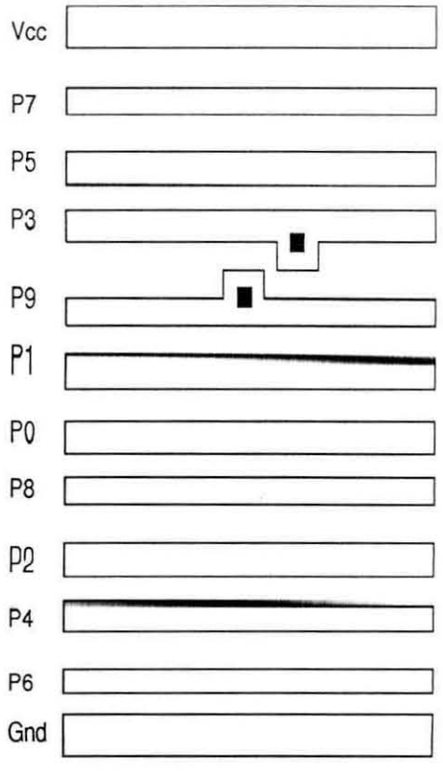

Figura 3.7: Esquema de prioridades na alocação de trilhas nas células da biblioteca do Projeto TRANCA.

3.2 O Módulo TRAMO

O módulo TRAMO (TRAnca MOdule generator) [LUB 90][SOT 91] realiza a geração automática de blocos em lógica aleatória a partir de células previamente desenhadas segundo a metodologia TRANCA. As células poderão ter sido selecionadas dentre as primitivas disponíveis nas bibliotecas ou projetadas conforme a necessidade.

O TRAMO é composto fundamentalmente por dois subsistemas:

- POTRANCA, o qual realiza o particionamento do circuito e o assinalamento das células às bandas e
- RETRANCA, responsável pelo posicionamento final (absoluto) e roteamento.

Há ainda um módulo exibidor (EXTRAMO), o qual permite ao usuário visualizar os resultados parciais do processo de síntese, antes mesmo da geração

das máscaras. Essa facilidade permite a realização de sucessivos particionamentos e posicionamentos para a escolha do conjunto de parâmetros mais conveniente para a síntese definitiva das máscaras.

O diálogo entre os subsistemas é feito por meio de arquivos, e gerenciado pela interface TENTOS, conforme já mencionado no início do capítulo.

3.2.1 Topologia das células

O desenho dos leiautes das células segue a estrutura de bandas e faz uso do esquema de prioridades na alocação das trilhas, já descritos na seção anterior. A figura 3.8 mostra o leiaute de uma célula *and* de 4 entradas.

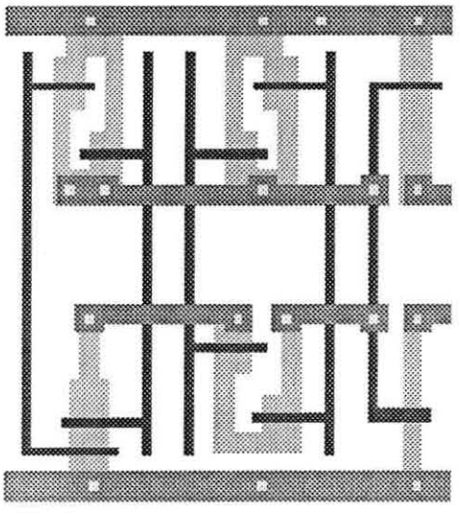

Figura 3.8: Leiaute de uma *and* de 4 entradas.

As principais características topológicas seguidas são:

- As entradas são realizadas com linhas de polissilício na vertical, as quais cruzam todas as trilhas não utilizadas para conexões locais;
- As saídas estão disponíveis em metal na borda direita da célula.

A disposição das linhas de polissilício permite a programação da entrada na trilha desejada mediante a colocação de um contato, juntamente com a devida margem de metal e polissilício. Quanto às saídas, pelo menos uma delas, no caso de haver mais de uma, apresenta a conexão entre as redes **p** e **n** incompleta e usando as trilhas P8 e P9. Esta medida visa possibilitar a conexão por justaposição sempre que possível, o que economiza a área de implementação da ponte em polissilício. Caso a justaposição da saída com alguma entrada da célula adjacente à direita não seja possível, existe uma célula de interconexão definida para isso.

Atualmente, existem duas bibliotecas, sendo uma em tecnologia 2.0μm [MOR 90b][REI 90] e outra em tecnologia 1.5μm [CRU 91]. As células implementam primitivas dos níveis lógico e funcional, envolvendo desde inversores, portas *nand* e *nor* até *flip-flops* de diversos tipos e bits para contadores síncronos. No total, são 46 células por biblioteca.

3.2.2 O subsistema POTRANCA

O subsistema POTRANCA recebe uma descrição do circuito a ser gerado em linguagem NILOTRANCA, a qual é uma extensão da linguagem NILO [WAG 87]. A descrição é constituída de portas lógicas e chaves bidirecionais, juntamente com todas as redes de interconexão, o que vem a caracterizar o circuito no nível lógico estrutural. O usuário pode impor restrições à síntese através de diretivas da linguagem, as quais versam sobre o posicionamento dos conectores da interface, a fixação de posições e orientações de células etc.

O problema do posicionamento de circuitos é dividido em duas etapas:

- particionamento do circuito em bandas e
- posicionamento intrabanda das células.

No particionamento, é utilizada a técnica de fatiamento [BRE 77] em conjunção com a heurística de Fiduccia-Matheyses [FID 82] para a transferência de células. As células são intercambiadas aos pares entre o novo bloco e o bloco residual, na expectativa de que o número de redes em comum entre os dois blocos considerados decresça. O algoritmo para o posicionamento relativo divide o módulo em regiões (partição horizontal) e depois, cada região em bandas (partição vertical), o que é mostrado na figura 3.9.

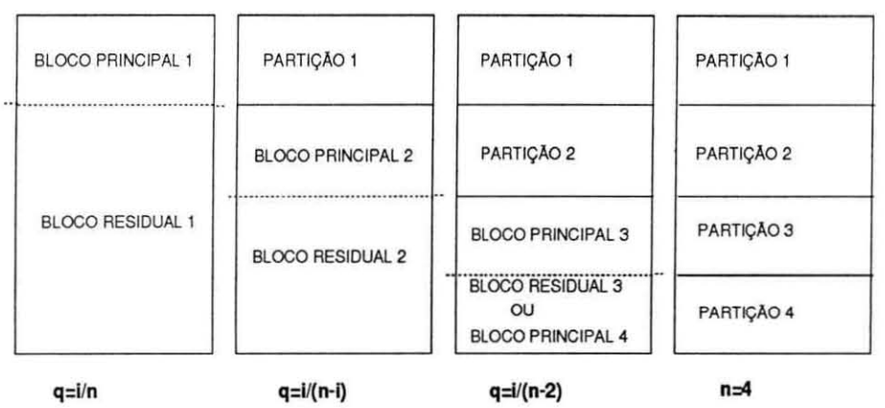

Figura 3.9: Algoritmo de particionamento do Potranca.

A partição horizontal objetiva evitar a saturação de trilhas dentro das bandas, o que acaba favorecendo o uso de células de interconexão. Já a partição vertical explora as ligações em metal da estrutura de banda. A cada nova região submetida à partição vertical, a localização das redes compartilhadas com a região vizinha anteriormente particionada é considerada.

A estratégia para o posicionamento intrabanda é um misto das técnicas de crescimento de aglomerados e das técnicas de partição: o movimento das células é feito de forma linear e a escolha baseia-se no corte mínimo. O primeiro passo é centralizar todas as bandas com relação à banda mais longa. Após, as células vão sendo selecionadas conforme seus escores com relação às já posicionadas dentro de sua banda. A seleção de células ocorre de forma concorrente entre as bandas e a ordenação das redes de interface norte e sul são consideradas.

Dentre os parâmetros que o usuário pode ajustar para interferir no posicionamento, encontram-se o fator de forma do bloco, o número de regiões e o número

de bandas.

3.2.3 O subsistema RETRANCA

O roteamento do circuito é iniciado pelo assinalamento das saídas das células, buscando primeiramente atender às imposições de trilhas que o usuário porventura tenha feito. A escolha das trilhas para entradas, por sua vez, é feito de modo a incentivar a justaposição e as redes mais longas recebem prioridade nesta alocação. Para as redes que devem trocar de bandas ou mesmo atravessar bandas, é feita a alocação de células de interconexão em pontos mais propícios, sempre considerando as bandas envolvidas aos pares de vizinhas.

Feito esse roteamento global, todos os pinos resultam alocados e o problema do roteamento intrabanda pode ser considerado caso a caso, para cada banda. Para o roteamento intrabanda, é realizada a composição das matrizes de roteamento das células constituintes, devidamente ordenadas. Nestas matrizes, estão mapeadas as possibilidades de acesso às entradas das células pela direita e pela esquerda e as trilhas ocupadas pela ou pelas saídas. Com essas informações é realizada a alocação de trilhas, seguindo da esquerda para a direita na banda, e considerando para cada pino, o próximo conector e as trilhas livres entre estes. Este processo é realizado à luz do esquema de prioridades na alocação das trilhas.

Concluído o roteamento, as bandas são compostas de modo que as linhas de alimentação componham dois pentes com dentes intercalados. Para isso, as bandas são alternadamente espelhadas em relação a y, juntamente com suas matrizes de roteamento. Só então, a representação simbólica do roteamento é expandida para a geométrica.

3.2.4 Considerações sobre os leiautes gerados pelo TRAMO

Os leiautes gerados pelo TRAMO são efetivamente mais compactos do que leiautes *standard cells*, conforme é relatado em [LUB 90]. Isto demonstra as vantagens do uso da metodologia TRANCA. Porém, considerando-se as restrições adotadas, quais sejam, uso de somente um nível de metal e máxima compactação horizontal das células, parece claro que as vantagens poderiam ser muito maiores.

O uso de somente um nível de metal força o surgimento de uma **direção preferencial** para roteamento, o que inevitavelmente conduz a congestionamentos de conexões na referida direção. O uso de células de interconexão, por sua vez, parece uma opção mais interessante do que criar pontes de polissilício no meio das células. Ocorre que no primeiro caso, apenas as pontes necessárias serão criadas e seu posicionamento é mais flexível. Já no segundo caso, o assinalamento de pontes dependeria do posicionamento das células e invariavelmente haveria mais pontes do que o necessário.

Há ainda um efeito de congestionamento local causado pelo uso de primitivas demasiadamente grandes. Como todas as células tem a altura de uma banda obrigatoriamente, quanto mais complexa for a primitiva, maior deve ser sua dimensão horizontal e mais trilhas tendem a ser usadas na sua implementação, o que baixa a transparência da banda onde esta estiver posicionada. Isto também inflexibiliza o posicionamento de células de interconexão, uma vez que os pontos de inserção ficam mais distantes. Soluções para esse efeito de *clusterização* horizontal seriam:

- Evitar o uso de primitivas muito complexas, decompondo a lógica;
- Implementar primitivas complexas em mais de uma banda, o que já está previsto na metodologia TRANCA.

Obviamente, tais soluções devem ser previstas na implementação de uma ferramenta de PAC.

A figura 3.10 mostra um possível leiaute do circuito Modem [REI 89], gerado pelo TRAMO (tecnologia 2.0μm).

O uso da metodologia TRANCA com tecnologia de dois níveis de metal e o desenvolvimento de um módulo de geração automática é proposto em [REI 91]. O módulo TRAMO II, lá apresentado, prevê o uso de células geradas automaticamente conforme a metodologia TRANCA. O conceito de decomposição de lógica é aplicado, de modo a aumentar a flexibilidade para a ferramenta de posicionamento.

3.3 O Módulo TRAGO

O TRAGO (TRanca Automatic GeneratOr) [MOR 90a] é um sistema que realiza a síntese baseado na geração automática de células. Porém, sua proposta é bastante inovadora. Ao invés de realizar o roteamento sobre as células geradas e devidamente posicionadas, o processo é invertido. A partir de uma previsão topológica para as células, é feito todo o roteamento da melhor forma possível, sem restrições. Após, os transistores são inseridos no roteamento e acomodados conforme o resultado do roteamento. Esta abordagem equivale a inserir a lógica sob os canais de roteamento.

A figura 3.11 mostra o fluxograma da síntese com o módulo TRAGO.

As células geradas pelo sistema têm sua topologia baseada na estratégia *gate matrix*, porém, apresentam somente um par de transistores por coluna. Cada célula é representada por duas tiras horizontais de difusão, as quais podem ser posicionadas sob as trilhas de roteamento de diferentes modos, conforme for mais conveniente. A figura 3.12 mostra o leiaute de uma célula gerada pelo TRAGO.

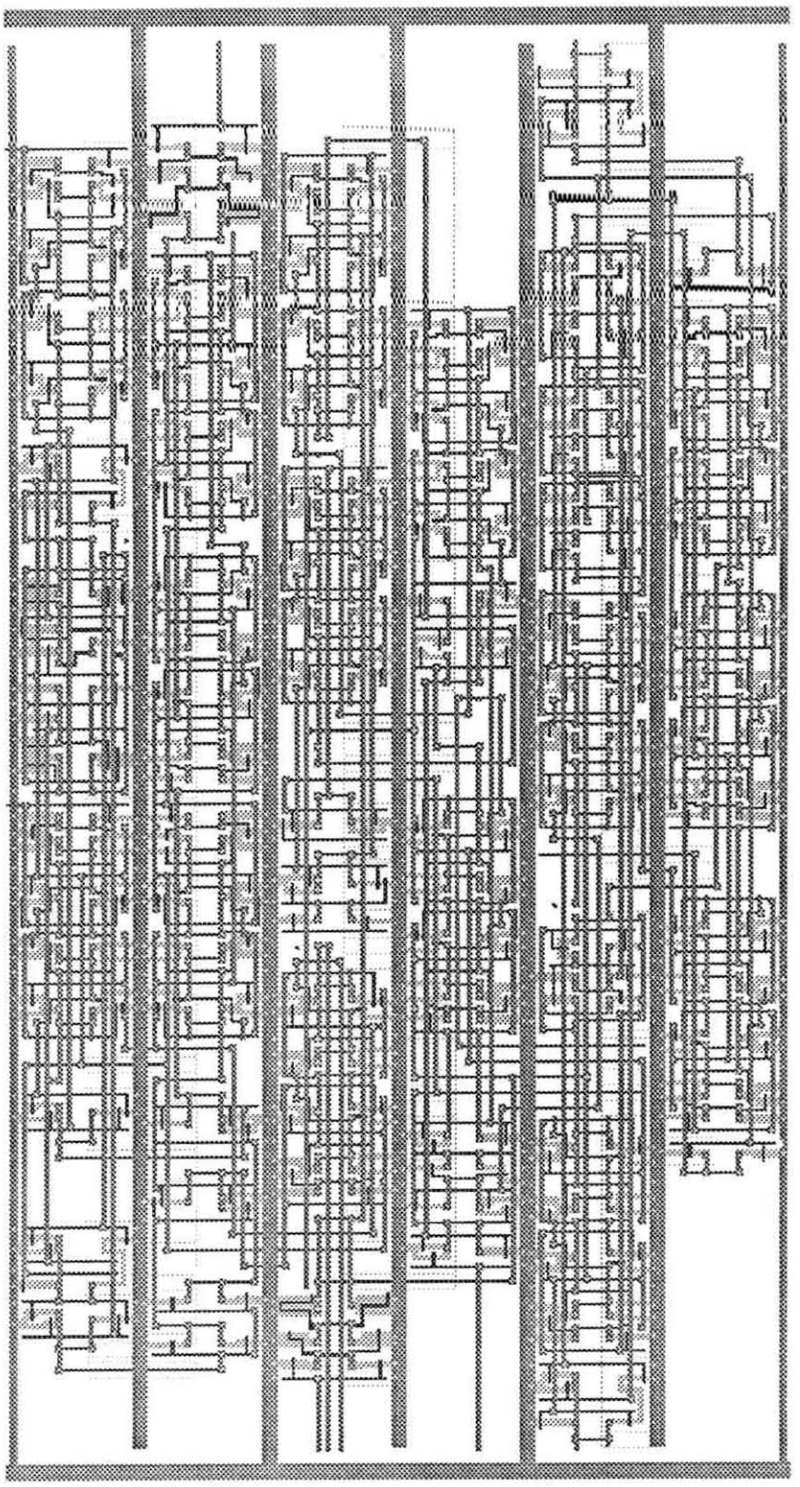

Figura 3.10: Um possível leiaute TRAMO para o circuito Modem.

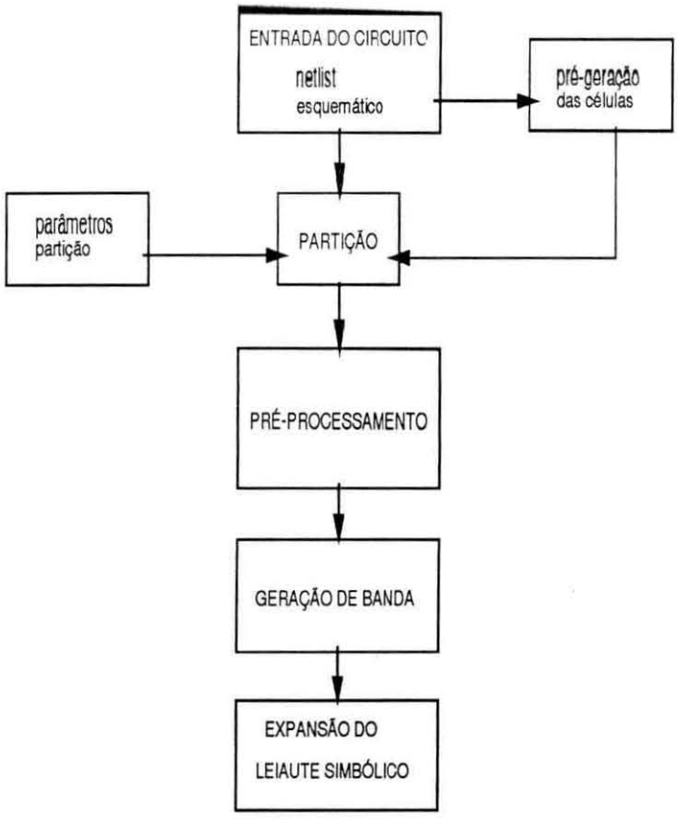

Figura 3.11: Fluxo da síntese com o módulo TRAGO.

3.3.1 Pré-processamento

A entrada do módulo TRAGO é uma *netlist* Spice hierárquica, a qual descreve células e as redes associadas. A descrição pode ser confeccionada manualmente ou através do editor de esquemáticos ESQUELETO [MOR 90c], o qual roda em ambiente DOS e permite a geração de relatório Spice.

O pré-processamento é a etapa na qual a descrição Spice é planificada e, a seguir, re-hierarquizada, de forma a agrupar transistores em **células básicas**, as quais serão sintetizadas. As células básicas são formadas por conjuntos de transistores conectados em série e/ou em paralelo entre a alimentação e a saída, tais como portas *nand*, *nor* e inversor.

Após a re-hierarquização, são ordenados os pares de transistores complementares com mesma porta, de modo que estes compartilhem a mesma coluna de polissilício. Este procedimente é denominado **pareamento**.

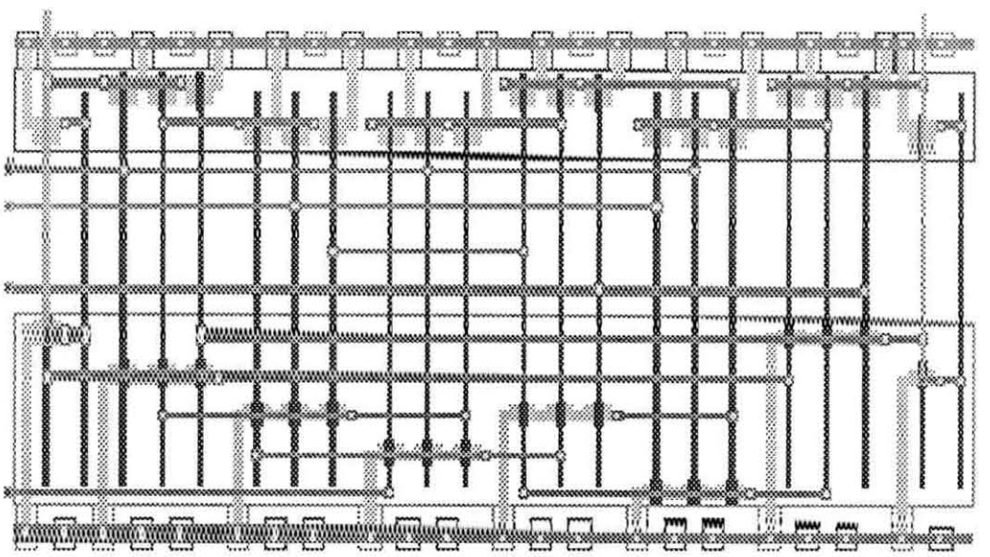

Figura 3.12: Leiaute de uma célula gerada pelo TRAGO.

3.3.2 Particionamento

Antes de iniciar o particionamento, é realizada uma pré-geração das células básicas, unicamente para informar ao particionador o número total de células do circuito, e para cada célula, o número e posição de entradas e saídas e sua largura.

O particionamento propriamente dito segue o mesmo algoritmo do TRAMO, tanto assim que o subsistema POTRANCA é utilizado.

3.3.3 Geração de bandas

A geração dos leiautes das bandas inicia pelas bandas que possuem maior número de redes divididas, ou seja, redes compartilhadas com outras bandas. As células serão geradas com topologia obedecendo às restrições do local onde devem ser inseridas. Porém, a ordem horizontal dos transistores dentro das células básicas já foi definida pelo pareamento, restando definir a ordenada das tiras de área ativa.

O posicionamento das células básicas utiliza técnicas de crescimento por aglomerados [PRE 88], de modo a maximizar as ligações por justaposição. Quando a justaposição não for possível, a ligação entre redes **p** e **n** é realizada por uma ponte

de polissilício, resultando em aumento de área.

Para o roteamento intrabanda, é utilizado o algoritmo *left-edge* devido à inexistência de restrições verticais. Porém, são definidos 5 subcanais, conforme a natureza da ligação:

- nós **N**: conectam drenos e fontes tipo **n** dentro de uma mesma célula;
- nós **NS**: conectam drenos e fontes de uma saída às portas das outras células;
- nós **P**: conectam drenos e fontes tipo **p** dentro de uma mesma célula;
- nós **PS**: conectam drenos e fontes de uma saída às portas das outras células;
- nós **MI**: mistos; conexões apenas entre portas.

O roteamento intrabanda é feito seguindo regras de ocupação de trilhas. Por exemplo, trilhas tipo NS podem ser usadas para conectar nós NS, N e MI. Trilhas N podem ser usadas para conectar nós N e MI. Trilhas MI só podem ser usadas na conexão de nós MI. O procedimento de roteamento horizontal é aplicado a cada ordenação de células. A ordenação eleita é aquela que resultar no menor número de trilhas. Este procedimento gera bandas de alturas diferentes, decorrência das necessidades de conexões de cada banda.

O roteamento vertical é feito em metal 2, sem restrições topológicas. O posicionamento e roteamento das bandas leva em consideração a topologia das bandas já geradas, de modo que as redes divididas já posicionadas não são mais alteradas.

Feito o roteamento simbólico, os transistores são inseridos sob as trilhas na seguinte ordem: trilhas NS e PS, N e P e MI. Com isto, fica determinada a posição vertical dos transistores.

3.3.4 Considerações sobre os leiautes gerados pelo TRAGO

Apesar de utilizar dois níveis de metal, o TRAGO segue a mesma estrutura rígida de bandas no que concerne ao roteamento. Daí haver grande concentração de conexões na direção horizontal, enquanto que conexões na vertical são minimizadas. Isto é bastante óbvio, na medida em que o TRAGO utiliza o mesmo particionador do TRAMO, o que caracteriza uma má exploração do segundo nível de metal. Essa má exploração ainda pode ser observada no modo pelo qual as conexões de saídas não justapostas são realizadas: com uma ponte em polissilício.

Contudo, a contribuição positiva do TRAGO reside na estratégia de somente gerar o leiaute das células após a realização do roteamento. Esta é uma forma eficaz de priorizar a minimização da área das conexões, uma vez que cada vez mais, esta tende a ser dominante em relação à área ocupada pelos transistores.

Por outro lado, a total flexibilidade na alocação de trilhas de roteamento por banda apresenta duas facetas: uma positiva e outra negativa. A positiva refere-se ao fato de nunca ocorrer saturação de bandas, pois sempre é possível a alocação de outra trilha. A negativa é que a partir de um certo número de trilhas, pode haver desperdício de área e piora do desempenho elétrico devido ao aumento dos parasitas. Além disso, ocorre o surgimento de uma espécie de canal de roteamento no centro das bandas. Estes efeitos negativos são detalhados em [REI 92b].

Na figura 3.13, é possível ver-se um leiaute TRAGO para o circuito Modem.

Encontra-se em desenvolvimento um sistema de síntese automática de leiautes denominado TROPIC, o qual resulta da integração do TRAGO com o sistema PRINT [ROB 88]. O PRINT é uma ferramenta de dimensionamento automático de transistores que utiliza uma formulação explícita para tratamento do atraso em redes CMOS.

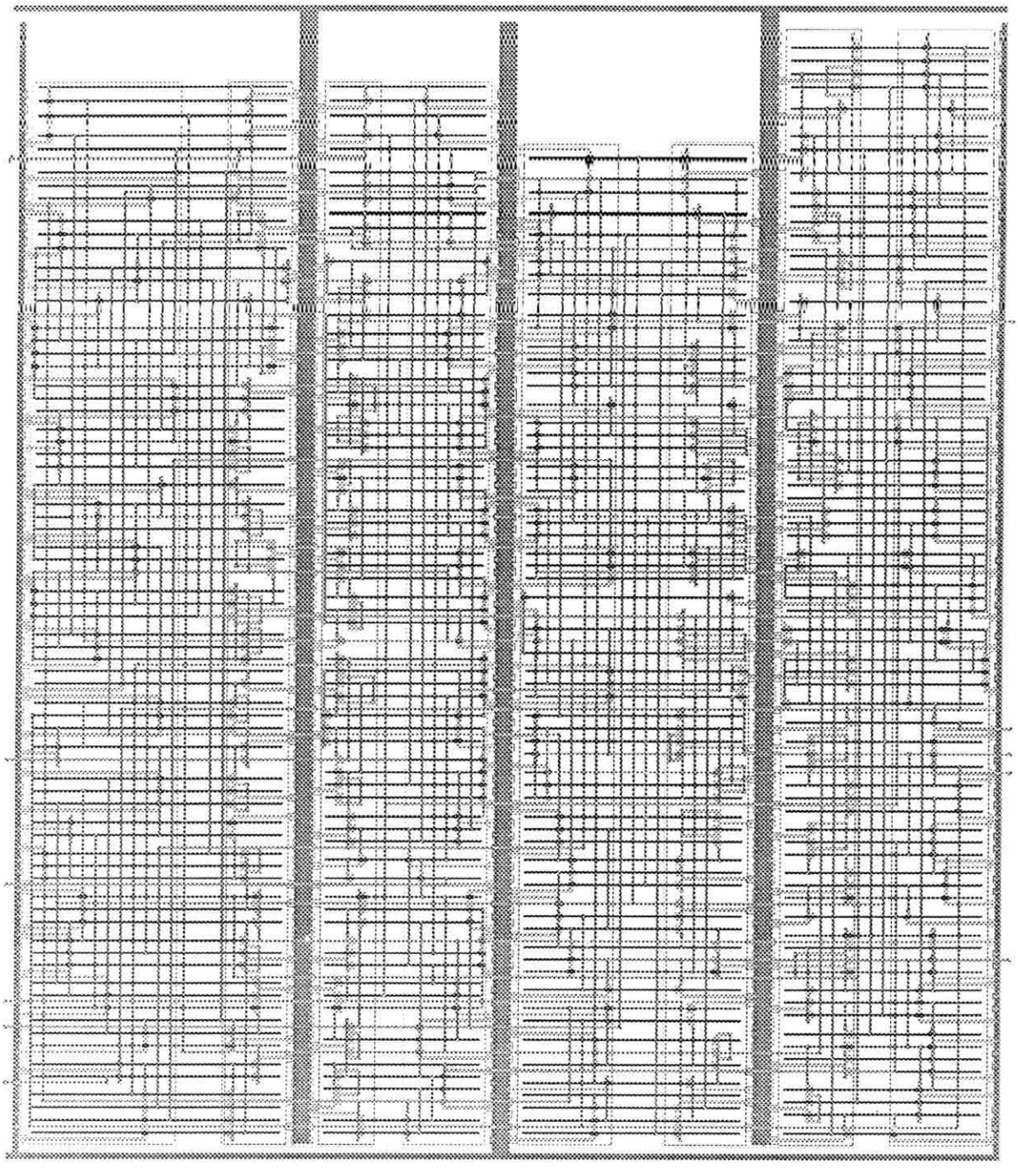

Figura 3.13: Um possível leiaute TRAGO para o circuito Modem.

4 A ABORDAGEM "MAR DE CÉLULAS"

Conforme visto no capítulo anterior, o sistema TRANCA oferece ao usuário duas opções de geração automática de leiaute: uma baseada em células TRANCA e outra baseada na geração de células segundo a abordagem *gate matrix*.

Estas opções oferecem inúmeras vantagens em relação aos métodos *full custom*, tanto em termos de tempo de geração quanto em termos de correção. Porém, no caso de se precisar prototipar um projeto com rapidez, a menos que se use um processo com escrita por feixe de elétrons, estas opções não parecem muito adequadas. Sendo assim, seria interessante incorporar ao TRANCA um módulo que permitisse a prototipação rápida de circuitos aliada ao pequeno tempo para mercado, na faixa de demanda imediatamente inferior aos estilos programáveis por todas as máscaras.

Se por um lado o mercado de Microeletrônica no Brasil encontra-se retraído, por outro lado, o mercado nacional de eletrônica de consumo não é desprezível e uma solução envolvendo o uso de ASICs poderia representar uma demanda de alguns milhares de peças/ano, o que justificaria a utilização de pré-difundidos como uma solução de baixo custo. Além disso, a realização da personalização aqui mesmo no Brasil é uma possibilidade menos improvável do que a implantação da *foundry* nacional. Esta suposição é confirmada pelo fato do Centro Tecnológico para a Informática (CTI) estar desenvolvendo seu próprio serviço de prototipação rápida baseado na personalização de matrizes pré-difundidas no exterior.

Pelo exposto anteriormente, o estilo **programável por algumas máscaras** (ou pré-difundido) é bastante conveniente para ser incorporado ao sistema TRANCA.

A definição e o desenvolvimento de uma abordagem versátil para pré-difundidos, que permita taxas de ocupação superiores àquelas atualmente atingidas

e que seja passível de integração com o TRANCA são os principais objetivos deste trabalho.

4.1 Definição de uma Nova Abordagem para Pré-difundidos

Na incessante busca de melhores taxas de ocupação, a maioria das abordagens de pré-difundidos ou procura aumentar ao máximo o número de transistores disponíveis nas matrizes, ou provê transistores com tamanhos diferentes para permitir o ajuste na alocação de canais. Isto revela maior preocupação com a arquitetura do que com a estratégia de ocupação propriamente dita.

Além disso, a aplicação dos resultados da constante evolução da tecnologia CMOS na implementação de novas arquiteturas tem sido o maior responsável pela viabilidade e expansão do uso comercial dos pré-difundidos. Porém, estes mesmos avanços tecnológicos também foram responsáveis pela ascenção dos dispositivos programáveis pelos usuários (PLDs), principalmente os FPGAs, que concorrem com os pré-difundidos no que se refere à prototipação rápida e implementação de ASICs.[1]

Então, o aprimoramento dos pré-difundidos não deve basear-se tão somente em fatores arquiteturais, tal como o aumento da capacidade de integração da tecnologia CMOS. É necessário elaborar estratégias de ocupação com vistas a explorar de forma mais eficaz a área de silício envolvida numa matriz. Para que isso seja possível, impõe-se uma nova visão para **abordagem**, a qual trate de forma eqüalitária a **estratégia de ocupação** e a **arquitetura**, desenvolvendo-as em total consonância. Sem dúvida, esta nova filosofia representa um diferencial a favor do aprimoramento dos pré-difundidos, já que os avanços tecnológicos são igualmente explorados por todos os estilos de projeto, inclusive pelos programáveis após o encapsulamento.

[1] Na realidade, a faixa de utilização dos FPGAs concentra-se um pouco abaixo da dos pré-difundidos. Isto ocorre porque o custo unitário dos FPGAs é invariante com o número de peças produzidas.

Para se propor um método que proporcione melhor aproveitamento da superfície de silício, é necessário avaliar-se o impacto da evolução da tecnologia CMOS sobre as abordagens existentes. Conforme exposto no primeiro parágrafo desta seção, e já detalhado no capítulo 2, muitas abordagens tentam aumentar a densidade útil simplesmente aumentado a densidade disponível, enquanto que outras provêem transistores menores para que menos espaço seja perdido quando da alocação de canais para roteamento. Estas estratégias parecem equivocadas se examinarmos mais a miúde as seguintes tendências decorrentes do *scaling down* da tecnologia:

- A dimensão dos transistores tem diminuído mais do que a dimensão da cabeça de contato;
- À medida que aumenta a possibilidade de integração, mais complexos tornam-se os roteamentos.

A tabela 4.1 mostra as áreas ocupadas pelo canal do transistor mínimo e pela cabeça de contato para várias tecnologias CMOS. A razão entre estas áreas constitui uma aproximação para a razão entre área ativa e área de roteamento nos circuitos integrados, pois todo transistor conecta-se ao resto do circuito por meio de pelo menos dois contatos.

A segunda tendência é bastante óbvia e acaba por agravar o problema criado pela primeira: se um circuito puder ser reprojetado numa tecnologia cujo fator de *scaling* é K, então a área ocupada resultará K^2 vezes menor [HAY 80]. Por outro lado, desde que as dimensões mínimas foram reduzidas, torna-se possível integrarem-se mais dispositivos utilizando a mesma área K, o que aumenta o número de conexões.

Associando-se as duas tendências, conclui-se que cada vez mais, a área ocupada pelas conexões torna-se dominante com relação à área dos transistores propriamente ditos.

tecnologia empresa	data	1- área da cabeça contato	2- área do transistor min (WxL)	1/2
3.0μm AMI 1 nível metal	1986	6 x 6	4 x 3	3
2.0μm ES2 2 níveis metal	1988	4 x 4	3 x 2	2.7
1.5μm ES2 2 níveis metal	1988	4 x 4	2 x 1.6	5
1.2μm ES2 2 níveis metal	1989	3 x 3	1.5 x 1.2	5
1.0μm ES2 2 níveis metal	1990	2.5 x 2.5	1.25 x 1.0	5

Tabela 4.1: Evolução das dimensões mínimas da tecnologia CMOS.

Além do problema do aumento da complexidade dos roteamentos, há de se considerar um outro problema presente nos estilos *cell based.* Nas abordagens que fazem uso de primitivas geométricas definidas ao longo de uma banda, nota-se que ocorre uma *clusterização* de transistores no mesmo sentido. Esta *clusterização* gera um congestionamento local de ligações na mesma direção, cuja densidade é diretamente proporcional à complexidade da primitiva. Portanto, do uso de primitivas muito complexas decorrem dois problemas:

- O traçado de conexões globais fica dificultado;
- Aumenta a granularidade do posicionamento, restringindo o espaço de soluções capazes de conduzir à ocupação ótima da superfície de silício.

Nos pré-difundidos, estes problemas não podem ser menosprezados, uma vez que há menos flexibilidade de alocação de rotas para as conexões.

A decomposição lógica reduz sensivelmente o efeito de *clusterização.* A idéia básica é prover primitivas geométricas de pequena complexidade e forçar a decomposição dos circuitos a serem implementados. Desta forma, obtém-se uma menor granularidade para o posicionamento e deixa-se ao encargo da ferramenta

de posicionamento achar a melhor localização de todas as células, inclusive das originadas pela decomposição.

Na definição de uma abordagem de pré-difundidos para o sistema TRANCA, foi dada total prioridade para o roteamento, justamente para minimizar todos os problemas descritos anteriormente. Esta priorização reflete-se de forma mais incisiva na elaboração da arquitetura das matrizes e no modo pelo qual as trilhas de roteamento serão alocadas. Isto será detalhado na seção 4.2.1.

Com relação ao problema da *clusterização*, considere-se uma matriz composta de alguns poucos tipos de CBs, onde cada tipo só permite a implementação de uma única função booleana. As funções, por sua vez, são de menor complexidade possível, de modo a garantir a decomposição de qualquer função lógica. Então, o projeto de um circuito obedece a seguinte seqüência de passos:

1. Escolha de uma matriz conveniente;
2. Geração de uma descrição equivalente através da decomposição lógica do circuito em termos de primitivas disponíveis na matriz escolhida;
3. Assinalamento das portas da descrição equivalente às CBs da matriz;
4. Roteamento do circuito.

Adotando-se tal estratégia, o conceito de biblioteca de primitivas perde o sentido (pelo menos no nível geométrico), pois todas as primitivas disponíveis já estão posicionadas na matriz sob a forma de CBs de uso específico. Deve-se apenas assinalar tais CBs, ou seja, para cada função lógica da descrição equivalente, escolher uma CB da matriz segundo critérios que visem a otimização do roteamento e do desempenho elétrico. Esta estratégia foi escolhida por amenizar os congestionamentos locais gerados pela *clusterização*.

O fato da arquitetura apresentar CBs bem definidas e de uso direcionado espalhadas uniformemente permite imaginar-se que as matrizes são compostas de

células pré-difundidas ao invés de transistores. Daí a abordagem ter sido denominada **MARCELA** (Mar de Células).

A característica que diferencia a abordagem Marcela dos demais pré-difundidos é a decomposição lógica. Examinemos, pois, os efeitos deste procedimento sobre a ocupação das matrizes. Chamemos, o conjunto de funções booleanas disponíveis numa matriz de **primitivas lógicas** ou simplesmente **primitivas**. Como cada CB existente na matriz foi projetada para implementar uma primitiva e como a quantidade de elementos (CBs) da matriz é fixa, a proporção entre tipos de CBs também é fixa. Por outro lado, os projetos podem apresentar as mais variadas proporções entre primitivas após a decomposição, sendo de se esperar que a taxa de ocupação das matrizes também varie muito. Ou seja, a taxa de ocupação de uma matriz depende da proporção entre primitivas do projeto e da proporção entre os tipos de CBs disponíveis na matriz utilizada.

Supondo que uma mesma arquitetura seja utilizada, a otimização da ocupação de matrizes pode ser realizada segundo três procedimentos independentes:

- Escolhendo uma matriz com proporção conveniente entre as CBs;
- Utilizando degeneração lógica;
- Aplicando simplificação de lógica multinível sobre a descrição equivalente;

O primeiro procedimento só é válido se estiver disponível mais de um tipo de matriz, o que pode ser bastante oneroso do ponto de vista comercial. Já a degeneração lógica consiste em utilizar uma ou mais CBs para implementar uma primitiva para a qual estas não foram projetadas. Por exemplo, pode-se implementar inversores em CBs projetadas para *nands*. Isto seria vantajoso caso houvesse escassez de CBs para inversores. Por outro lado, este procedimento não é aplicável para qualquer tipo de primitiva. Por fim, a simplificação multinível é a otimização de maior complexidade, mas que, a princípio, tenderia a trazer melhores resultados.

No escopo deste trabalho não será abordada a simplificação multinível, enquanto que o uso de degeneração lógica será explicitado quando for o caso.

4.2 Arquitetura da Matriz de Uso Genérico

O primeiro passo no desenvolvimento da abordagem Marcela foi a especificação de uma matriz que comportasse os mais variados tipos de projetos, com um grau de complexidade médio. A especificação de tal matriz de uso genérico envolveu dois tópicos:

- Definição das primitivas para as quais seriam implementadas CBs;
- Determinação da proporção ideal entre estas CBs.

O conjunto de primitivas escolhido foi inversor, *nand* e *nor*, por serem simples e permitirem a decomposição de qualquer outra lógica. Adicionalmente, a experiência com projetos digitais aponta a necessidade de chaves bidirecionais para facilitar o acesso a barramentos e a implementação de multiplexadores e de *flip-flops* dinâmicos. Então, foi incorporado ao conjunto de primitivas a chave complementar CMOS, também conhecida por *transmission gate*.

A determinação da proporção entre as primitivas, bem como a escolha do número de entradas para *nand* e *nor*, foram feitas mediante levantamento estatístico sobre uma amostra contendo circuitos projetados no âmbito do GME e as partes de controle de dois microprocessadores comerciais [GUN 91a] [GUN 91c]. Os resultados apontaram o inversor como sendo de longe a primitiva mais utilizada, indicando que sua quantidade deveria ser o dobro das primitivas *nand* e *nor*. Quanto à chave CMOS, parece razoável provê-la na mesma quantidade em que os inversores ocorrem, de modo a facilitar a implementação de bits de memória dinâmica, a serem usados em *flip-flops*.

Também, o posicionamento relativo destas CBs foi estudado, e a organização fundamental, denominada **unidade básica (UB)**, foi definida. A unidade básica, por sua vez, é repetida de modo a formar a matriz. O número de CBs disponíveis na matriz é múltiplo do número de UBs que a compõe, enquanto que a proporção entre as CBs é determinada pela UB adotada. A figura 4.1 mostra a UB da matriz de uso genérico.

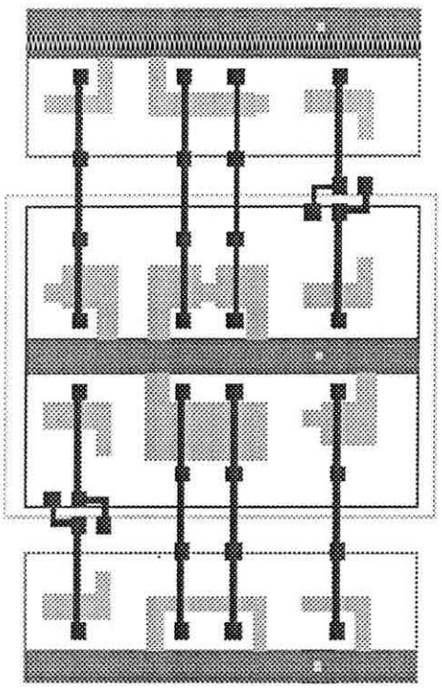

Figura 4.1: UB da matriz de uso genérico.

Na arquitetura das matrizes, foi aplicado um subconjunto de características da metodologia TRANCA, adaptado à tecnologia com dois níveis de metal e às restrições impostas pelo estilo pré-difundido.

A estrutura de banda foi mantida e o barramento de alimentação segue a estrutura de pente, nos mesmos moldes do TRANCA. Cada UB é delimitada por duas linhas de GND, sendo cruzada por uma linha de VCC ao centro. As CBs que ocupam a banda superior encontram-se espelhadas.

Com relação à arquitetura, a inovação da abordagem Marcela reside no fato de utilizarem-se conceitos já abandonados nos pré-difundidos, juntamente com

conceitos avançados. Dentre os avançados, suprimiram-se os canais de roteamento (explícitos ou alocados), adotando-se o conceito de roteamento compartilhado (*over-the-cell routing*). Para tanto, as CBs foram projetadas de modo a acomodar trilhas de roteamento tanto na horizontal como na vertical, dando-lhes um caráter de **transparência**.

Sabendo-se da necessidade de priorizar o roteamento, a estratégia de preencher toda a matriz com transistores não conduz a melhores taxas de ocupação. Então, pode-se agrupá-los em CBs, isolando-os geometricamente, e tirar máximo proveito desta topologia aparentemente obsoleta. Por exemplo, desde que as CBs são totalmente independentes umas das outras, pode-se projetar cada tipo de CB de modo a obter máximo desempenho elétrico para a primitiva a ser implementada. Adotando-se lógica complementar, as portas dos pares complementares podem ser conectadas em polissilício, o que economiza muitas ligações em metal. Especificamente na matriz de uso genérico, onde as portas dos transistores estão na direção vertical, o efeito sobre a transparência vertical é extremamente positivo, permitindo praticamente o mesmo grau de roteabilidade existente na direção horizontal. Esta é, sem dúvida, a característica arquitetural mais relevante da abordagem Marcela.

A região de óxido espesso existente entre as CBs é utilizada para roteamento, comportando uma trilha vertical que, devido a sua posição estratégica, vem facilitar as ligações entre células de bandas diferentes.

O uso de dois níveis de metal proporciona a existência de transparência nas duas direções. O metal 1 é utilizado na horizontal para facilitar a conexão das fontes dos transistores. Como qualquer entrada ou saída de célula é realizada em metal 1, este nível deve merecer um tratamento um pouco privilegiado.

Para a definição do número de trilhas horizontais, assumiu-se o valor 10, proposto em [REI 87], como uma boa aproximação. O número de trilhas verticais é função da largura de cada CB. Na realidade, as larguras das CBs foram discretizadas em termos de trilhas verticais.

Diferentemente do TRAMO, não foi adotado o passo de metal para determinar as dimensões das trilhas. O motivo é a necessidade de se aumentar o número de pontos de contatos e vias, o que seria dificultado se os contatos e/ou vias fossem alternados. Para que contatos e vias adjacentes fossem permitidos, as trilhas de metal possuem largura tal que comportam um contato ou via, estando as trilhas separadas entre si da distância mínima. Mas a exemplo do TRAMO, há necessidade de administrar-se a alocação de trilhas para evitar a saturação decorrente da má utilização. A figura 4.2 mostra o esquema de prioridades para as trilhas horizontais. No sentido vertical não há esquema semelhante.

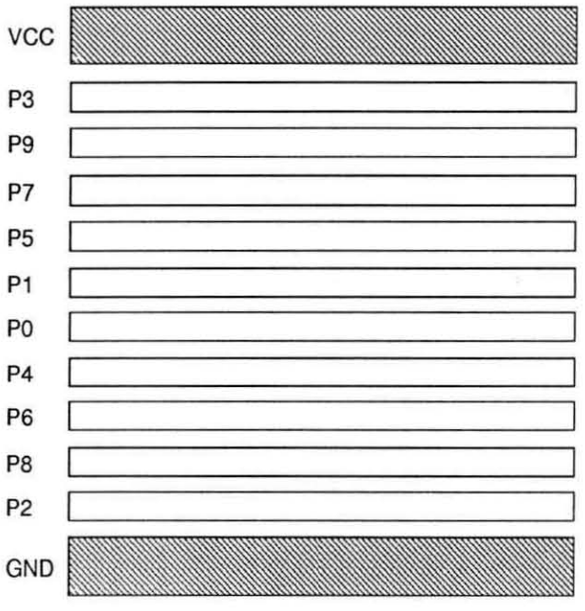

Figura 4.2: Esquema de prioridades na alocação de trilhas em arquiteturas Marcela.

Com a decomposição lógica, a distinção entre roteamento local e global perde um pouco o sentido. Mas se assumirmos roteamento **intracelular** como sendo as conexões entre redes **p** e **n** e roteamento **intercelular** como sendo qualquer outro tipo de conexão, pode-se adotar o mesmo esquema de prioridades do TRAMO, onde trilhas para roteamento intracelular são alocadas a partir de P9 e P8, enquanto trilhas para roteamento intercelular são alocadas a partir de P0 e P1. Note-se ainda que a ordem em que os graus de prioridades aparecem é diferente daquela adotada no TRAMO: isto deve-se à topologia das células, a ser detalhada na próxima seção.

4.2.1 Características Topológicas

Para a implementação do leiaute da matriz de uso genérico, adotou-se a tecnologia CMOS de 1.2μm de canal e dois níveis de metal da empresa francesa ES2, cujo acesso é possível por meio dos projetos multiusuário dos quais a UFRGS participa. Nesta tecnologia, as regras referentes às dimensões mínimas e espaçamento dos dois níveis de metal são as mesmas, o mesmo ocorrendo com relação a contatos e vias. Isto significa que o passo horizontal é igual ao vertical.

O desenho das CBs teve como referência uma grade virtual de metal 1 na horizontal e metal 2 na vertical, representando todas as trilhas existentes sob a superfície a ser ocupada pela célula. Esta grade serve como guia na previsão do roteamento. Por conveniência, contatos ou vias só podem ser colocados nos pontos de interseção da grade, de modo que as possíveis entradas e saídas das células foram localizadas sobre tais pontos. Também, a troca de direção nos roteamentos só pode ocorrer nestes pontos de interseção. Por motivo de economia de área, optou-se por restringir a colocação de vias sobre polissilício. Fora esta restrição, a área referente a CBs não alocadas para lógica pode ser completamente utilizada para roteamento.

Para permitir o acesso às entradas das células, foram providas **esperas** em polissilício sobre as entradas para acomodar um contato. Cada entrada possui 3 ou 4 esperas, conforme o tipo de CB. A personalização das fontes dos transistores é feita simplesmente mediante a colocação de um contato.

As áreas ativas dos transistores estão localizadas sobre as trilhas P9 e P7 no caso dos tipo **p** e sobre as trilhas P8 e P6 no caso dos **n**. Estas trilhas são reservadas para realização de conexões intracelulares, só sendo liberadas quando a CB não for utilizada. Ainda assim, sua utilização para roteamento intercelular fica restrita às regiões onde há CBs que não serão utilizadas.

As esperas estão localizadas em P2, P3, P4 e P5, sendo também possível o acesso por qualquer outra trilha, desde que se use *dogleg*. Inclusive, o *dogleg* é

um recurso previsto no roteamento global, só havendo restrição quanto ao seu uso sucessivo numa mesma conexão.

A chaves CMOS constituem o caso crítico em termos de conectividade, pois cada UB deve ser acessada por quatro sinais. Para aumentar a conectividade destas CBs, cada transistor recebeu 3 esperas para conexão de entradas. A localização de 2 das 3 entradas sobre as trilhas P0 e P1 visa a facilitar a conexão dos sinais globais de relógio, os quais normalmente controlam as chaves CMOS.

As topologias das CBs, juntamente com a grade podem ser visualizadas na figura 4.3.

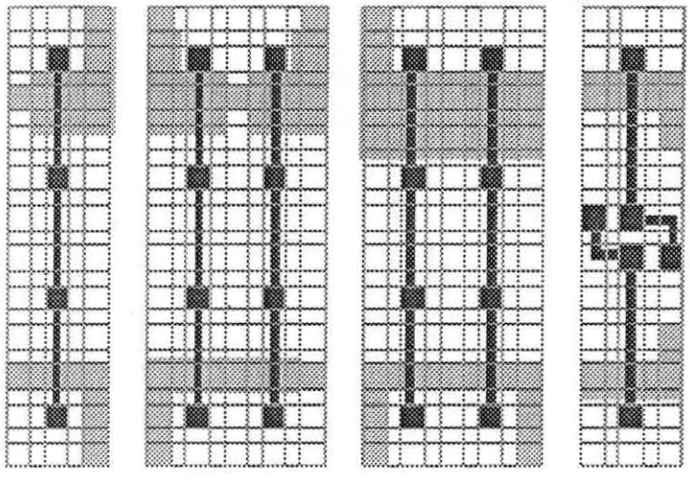

Figura 4.3: Leiautes das CBs da matriz de uso genérico sob a grade de roteamento.

A capacidade alvo da matriz foi fixada inicialmente em 1000 *gates equivalentes*, de modo a comportar projetos de pequena e média complexidades, tipicamente encontrados nas **lógicas de cola**.

O leiaute da matriz, então denominada **MAR1000**, é mostrado na figura 4.4.

A matriz MAR1000 é composta por 512 UBs, o que equivale a 1512 CBs, assim distribuídas:

- 252 CBs para *nand* de duas entradas;

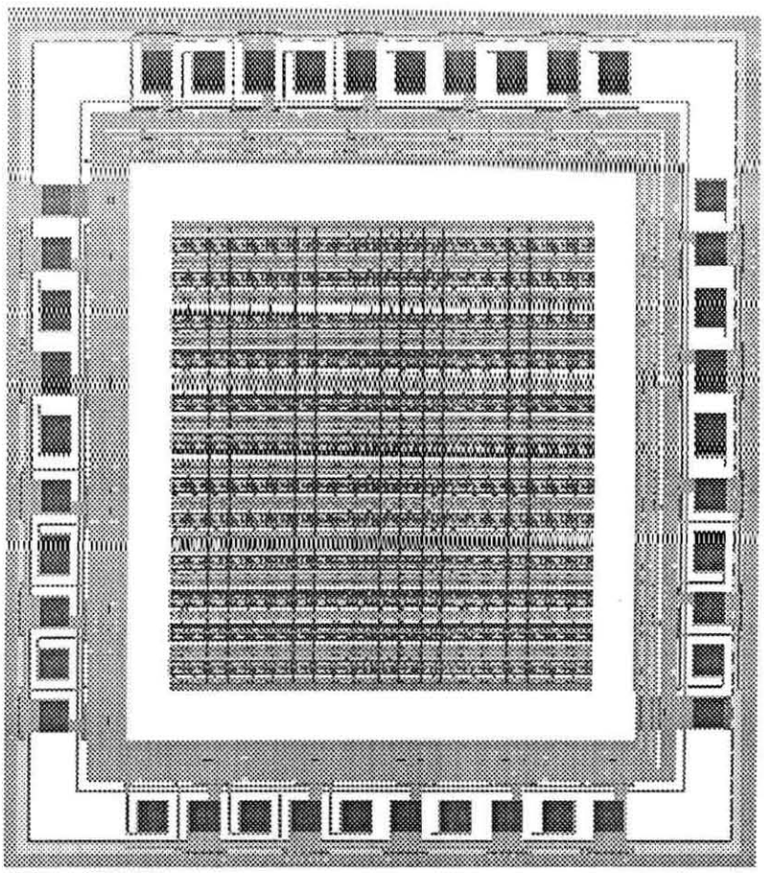

Figura 4.4: Leiaute da matriz MAR1000.

- 252 CBs para *nor* de duas entradas;
- 504 CBs para inversores;
- 504 CBs para chaves CMOS.

A contagem de *gates* atinge 1008 e a densidade da matriz não personalizada é de 2133 transistores/mm^2. São também providos 40 *pads*:

- 2 *pads* de alimentação (1 Vdd e 1 Gnd);
- 19 *pads* de entrada;
- 19 *pads* de saída;

Os *pads* de entrada e saída encontram-se alternadamente distribuídos. A área total da matriz é 2,52x2,52mm^2.

4.2.2 Características Elétricas

O projeto das CBs levou em consideração tanto aspectos topológicos quanto aspectos elétricos [GUN 91a]. Quanto aos topológicos, buscou-se facilitar o roteamento, flexibilizando a realização das conexões entre os elementos. No que concerne ao desempenho elétrico, procurou-se minimizar a diferença entre os tempos de subida e descida de cada célula, com a preocupação de dar aos transistores o dimensionamento ótimo. A ausência de qualquer leiaute implementado com esta abordagem até então levou à adoção de um *fanout* estimado com base na experiência do projetista, cujo valor equivale à quatro vezes a maior carga capacitiva encontrada entre as CBs.[2] É interessante ressaltar que variações do desempenho elétrico devido a diferentes topologias não puderam ser constatadas pelas simulações elétricas com o Spice 2G [VLA 81]. Assim sendo, dentre as várias versões de leiaute que proporcionavam bom desempenho elétrico para cada CB, foram escolhidas aquelas que maximizavam a conectibilidade.

Na reavaliação do projeto elétrico das CBs, procurou-se elucidar dois pontos principais:

- Quais são os números típico e máximo de portas conectadas a uma mesma saída, na abordagem e
- Dada uma rede qualquer, qual a pior relação entre a carga capacitiva representada pelas entradas das células a ela conectadas e a carga representada pela própria rede.

Para determinar-se o primeiro ponto, foi feito um levantamento estatístico utilizando os equivalentes Marcela para 4 circuitos em lógica aleatória [GUN 92]: Codificador e Decodificador do Modem [REI 89], Gamatual [APR 92] e ALULS181. O resultado é apresentado no diagrama 4.5. As maiores ocorrências são, sem dúvida

[2] No caso, a CB para *nor* sempre possuia a maior capacitância de entrada.

nenhuma, 1 e 2. Porém, adotando-se 6 como o caso tipicamente severo de *fanout*, diminui-se sensivelmente a necessidade de avaliação de caminhos críticos para inclusão de reforçadores de sinal (*buffers*). Como a avaliação de caminhos críticos não será abordada neste trabalho, adotou-se 6 como sendo o *fanout* típico para efeitos de avaliação do desempenho elétrico das CBs.

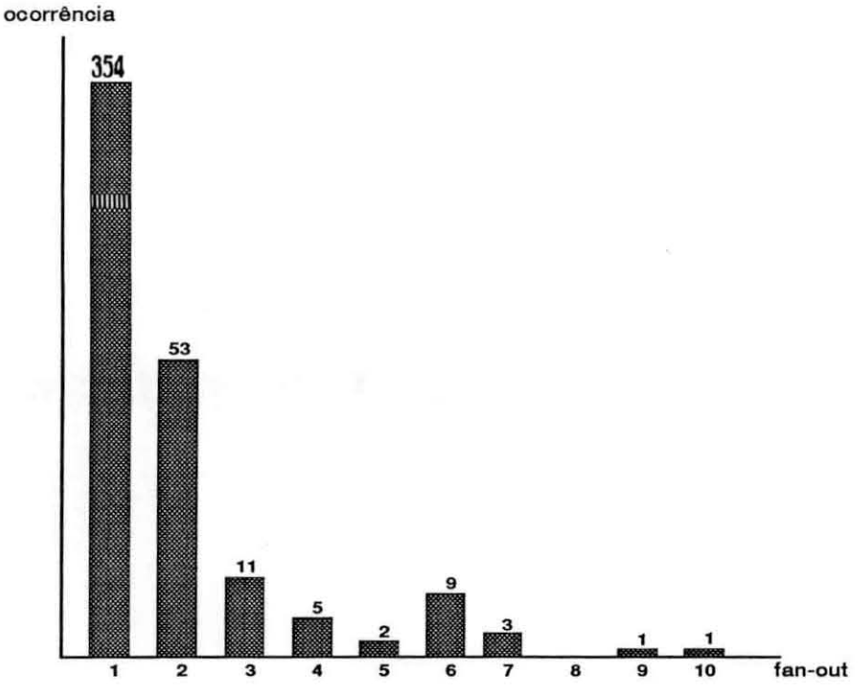

Figura 4.5: Diagrama de ocorrências de fanouts de uma amostra de equivalentes Marcela.

A elucidação do segundo ponto dependeria de levantamento estatístico sobre um leiaute implementado, de modo a obterem-se dados reais. Utilizando-se o leiaute do circuito teste TCHÊ [GUN 91b], foi possível avaliarem-se as conexões de forma mais realista, inclusive levando em conta o efeito da decomposição lógica. O circuito TCHÊ implementa o Modem e mais alguns blocos funcionais tais como *flip-flops*, multiplexadores e somador completo, todos assinalados e conectados manualmente. Foram então levantados os 10 supostos piores casos de cargas capacitivas deste circuito, entre conexões mais longas e maior número de portas conectadas numa mesma saída [GUN 92]. A tabela 4.2 revela os valores encontrados, onde C_t é o somatório das capacitâncias das portas conectadas a uma saída e C é a capacitância do roteamento. Nota-se que o valor médio para a relação C_t/C é 1,18, indicando

que pode ocorrer com certa facilidade valores iguais para C_t e C.

rede	C_t (fF)	C (fF)	C_t/C
I38	146,1	71,5	2,04
I21	59	88,3	0,69
I39	20,5	61,1	0,34
I20	107,6	36,5	2,95
A5	20,5	32,7	0,63
I11	53,1	87,5	0,61
O8	110,5	106,6	1,04
I24	48,6	57,2	0,85
I95	169,1	125,3	1,35
A22	136,5	102,3	1,33

Tabela 4.2: Capacitâncias representadas pelas conexões (C) e capacitâncias representadas pelas portas dos transistores (C_t).

Da conjunção destes dois resultados estatísticos, conclui-se que o caso severo de carga seria C = C_t = 6, ou seja, uma célula em tal situação conectar-se-ia a 12 outras células. Neste ponto, assumiu-se que o *fanout* unitário corresponderia a carga capacitiva de uma das entradas da *nor*, cujo valor (20fF) corresponde ao pior caso.

Então, as CBs que implementam funções lógicas foram ressimuladas com o Spice 3D2 em ambiente SUN, fazendo uso da condição de carga definida anteriormente, ou seja, C_L=12 x 20fF. As medidas dos atrasos foram feitas segundo dois critérios:

- Critério dos 50%-50%: tempo decorrido entre o sinal de entrada atingir 50% de seu valor final até a saída atingir 50% do valor final resultante da variação aplicada;
- Critério dos 10%-90%: tempo decorrido entre o sinal de entrada atingir 10% de seu valor final até a saída atingir 90% do valor final resultante da variação aplicada.

A tabela 4.3 mostra os valores obtidos nas simulações. No caso das CBs para *nand* e *nor*, onde há duas entradas, é mostrado somente o maior dos valores encontrados.

Critério	Atraso(ns)	inversor	*nand*	*nor*
50%-50%	tc_{LH}	0,67	0,80	1,12
	tc_{HL}	0,69	0,88	0,95
10%-90%	tc_{LH}	1,83	2,10	2,82
	tc_{HL}	1,71	2,20	2,17

Tabela 4.3: Atrasos obtidos por simulação com *fanout* 12 (C_L=240fF).

Note-se que para o critério dos 50%-50%, que é o utilizado comercialmente, o maior atraso encontrado foi 1,12ns. Mas para o critério dos 10%-90%, o maior atraso é de 2,82ns. O pior caso sempre corresponde à *nor*, cuja otimização de desempenho acaba esbarrando nas características da tecnologia CMOS. O desempenho do inversor e da *nand* foram praticamente equivalentes. Esta constatação induz ao uso de CBs *nands* (e não *nors*) para a implementação de inversores, quando da adoção de lógica degenerada.

Com a finalidade de melhor caracterizar a matriz MAR1000, foi feita simulação de *toggle* do *flip-flop* D simples (sem *set* ou *reset*), implementado com o uso de *transmission gates*. O resultado da simulação apontou para uma freqüência máxima de funcionamento igual a 250MHz.

A julgar pelos resultados das simulações Spice, e levando em conta o fator de segurança adotado em todas as aproximações assumidas, o desempenho elétrico da matriz MAR1000 parece estar assegurado.

Teoricamente, a avaliação de desempenho poderia ser feita de dois modos:

- Medindo-se os atrasos nas estruturas de silício, ou
- Simulando-se eletricamente o circuito ou suas partes fundamentais, fazendo uso

de aproximações realistas para as cargas das saídas.

A não adoção da primeira opção deve-se à imprecisão de qualquer medida realizada com o equipamento atualmente disponível no GME. Além disso, infelizmente as estruturas em anel implementadas no circuito teste TCHÊ para as referidas medidas de precisão se mostraram inoperantes em todos os protótipos, não tendo sido possível até o presente momento obterem-se aproximações razoáveis para os atrasos das células Marcela. Este fator justificou todo o procedimento de ressimulação descrito anteriormente.

Não obstante o fracasso das medidas com os osciladores, outras estruturas do circuito teste funcionaram. Porém, como as demais estruturas estão conectadas ao exterior somente por *pads*, seus desempenhos provavelmente ficam mascarados pelo máximo desempenho dos próprios *pads*. Mesmo assim, foi detectado o funcionamento correto do *flip-flop* D até freqüências pouco acima de 10MHz.

4.3 Outras Considerações Sobre a Abordagem Marcela

É importante ressaltar mais uma vez que a decomposição lógica é unicamente funcional e nunca posicional, ou seja, a escolha de um equivalente Marcela para uma dada função não define em absoluto a posição para o assinalamento das primitivas resultantes. Apenas obtém-se uma nova descrição lógica no nível estrutural, composta pelas primitivas disponíveis na matriz alvo. A definição de posições ficará ao encargo da ferramenta de assinalamento.

A decomposição de um determinado projeto deve ser feita com base na biblioteca de equivalentes Marcela para a matriz escolhida, a qual possui descrições no nível lógico estrutural em termos de primitivas Marcela para funções lógicas e blocos funcionais (tais como *flip-flops* e decodificadores). Algumas funções e blocos podem ter mais de uma versão, de modo a fornecer ao projetista um recurso a mais

quando a ocupação da matriz estiver próxima do limite.

Para ilustrarem-se tanto as opções de decomposição como os leiautes para a matriz MAR1000, serão apresentados dois exemplos bastante típicos: a função lógica *xor* e o bloco funcional *flip-flop* D com *set* e *reset*.

A figura 4.6 mostra duas versões para a função *xor*: uma utilizando somente portas *nand* e inversores e outra utilizando *transmission gates*. A versão que utiliza somente portas *nand* e inversores é apropriada para circuitos em que haja barramentos ou registradores que fazem uso de *transmission gates*. Já a versão da *xor* que usa *transmission gates* é mais apropriada para circuitos onde há sobras dessas primitivas, como por exemplo, quando houver somente *flip-flops* estáticos. Poder-se-ia ainda imaginar outras duas versões, originadas destas duas, as quais usassem lógica degenerada. Na falta de CBs para inversores, o que é algo bastante comum, CBs para *nands* seriam utilizadas para implementar tal primitiva. Com isso, temos quatro versões para a função *xor* de duas entradas.

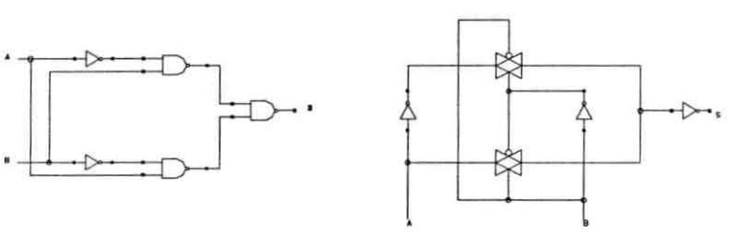

Figura 4.6: Dois equivalentes Marcela para *xor*: com portas *nand* (esq.) e com *transmission gates* (dir.).

A figura 4.7 mostra um possível leiaute Marcela para cada uma das versões da *xor*. Conforme dito anteriormente, é impossível precisar qual será o leiaute final para a função, uma vez que o assinalamento depende também do resto do circuito.

A figura 4.8 mostra duas versões para um *flip-flop* D com *set* e *reset*. A utilização da versão com *transmission gates* é recomendada sempre que possível,

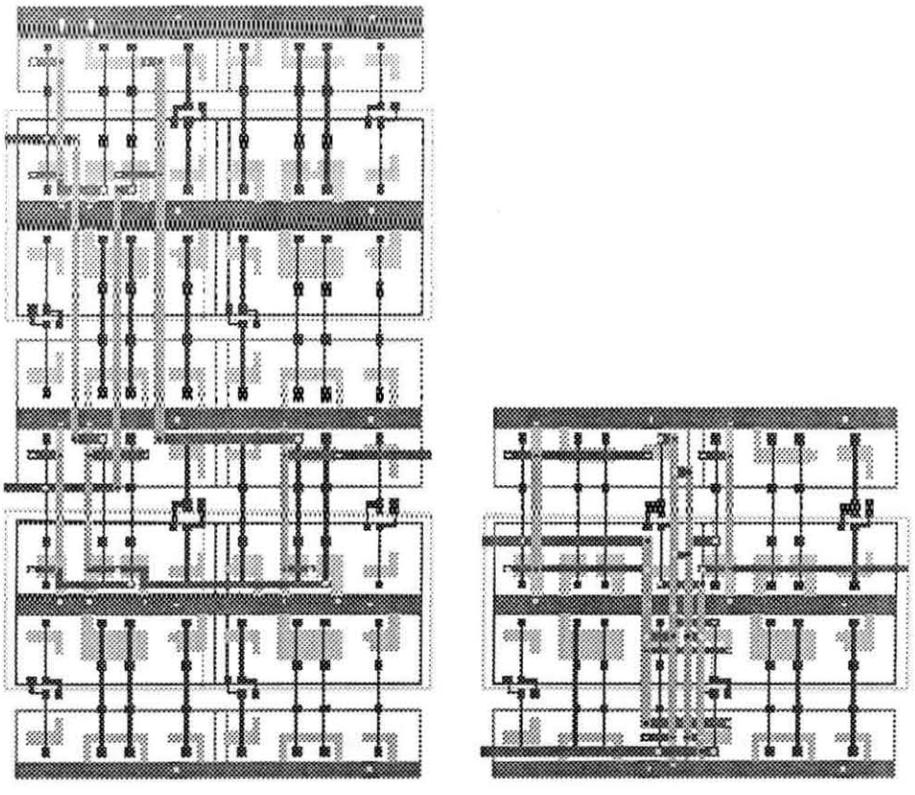

Figura 4.7: Leiautes para as duas versões da *xor*: com portas *nand* (esq.) e com *transmission gates* (dir.).

no caso de se usar a matriz MAR1000 para implementação do circuito, pois além de ser mais rápida, a proporção entre as CBs da UB desta matriz é extremamente favorável. O uso da versão estática deve restringir-se para os casos onde há escassez de *transmission gates*.

A figura 4.9 mostra uma possível implementação para cada versão do *flip-flop* D.

Obviamente, nada impede que, escolhidos os equivalentes, o projetista tente realizar simplificações ou até mesmo transformações lógicas com o intuito de diminuir a quantidade de alguma função que esteja escassa. Este esforço é válido quando o projeto explorar o limite da matriz, o que é bastante interessante sob o ponto de vista econômico.

Os algoritmos e ferramentas para automação do projeto são descritos no próximo capítulo.

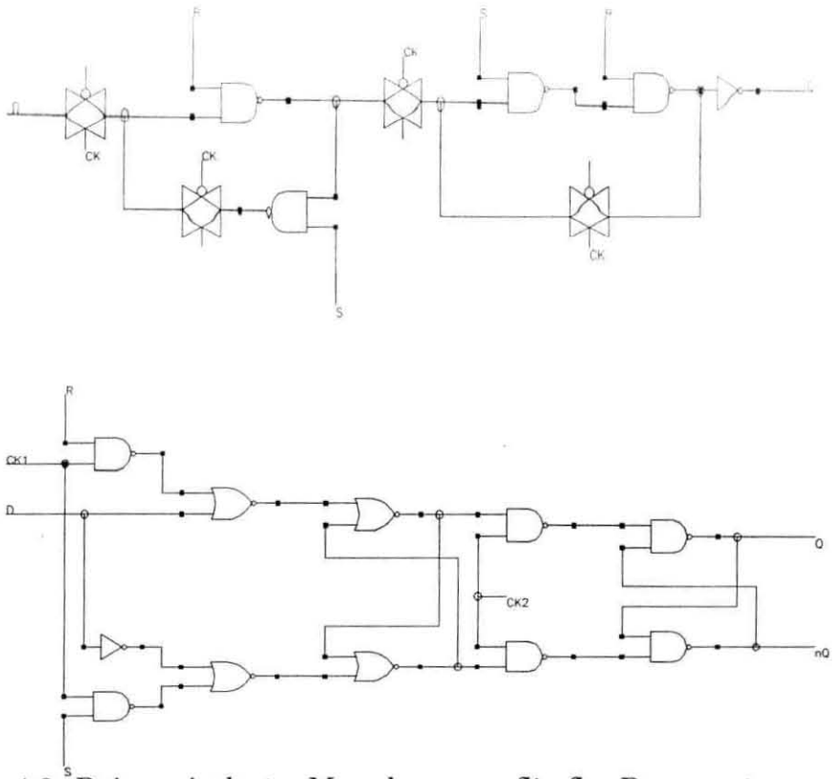

Figura 4.8: Dois equivalentes Marcela para o *flip-flop* D com *set* e *reset*.

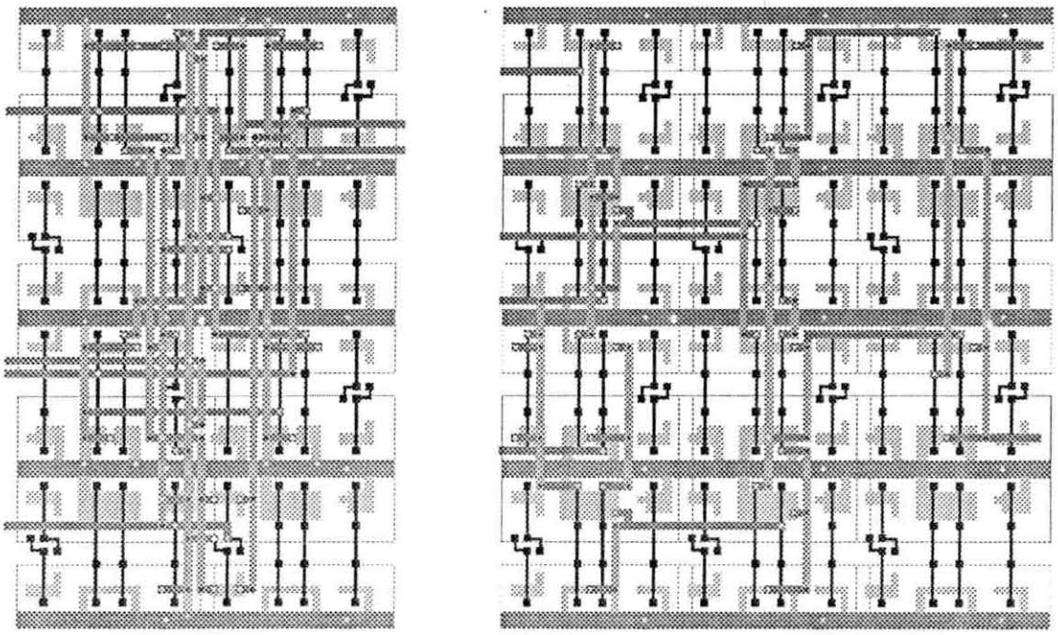

Figura 4.9: Leiaute para as duas versões do *flip-flop* D SR: dinâmico, com *transmission gates* (esq.) e estático (dir.).

5 ALGORITMOS E FERRAMENTAS PARA A GERAÇÃO DE CIRCUITOS MARCELA

A síntese automática de circuitos envolve uma série de procedimentos, os quais, em sua maioria, baseiam-se em heurísticas. Isto ocorre porque invariavelmente a solução ótima para tais problemas é de complexidade não polinomial (np), inviabilizando a computação. Muito embora os resultados alcançados aplicando-se heurísticas geralmente fiquem longe do ótimo, ou mesmo longe das soluções encontradas pelo projetista humano, ainda assim vale a pena automatizar o processo, pois o ganho em termos de tempo e precisão são enormes. E se for permitido ao usuário acessar alguns parâmetros que guiem a síntese, então é possível gerarem-se várias soluções num curto espaço de tempo e após, escolher-se a mais conveniente.

O processo de geração de leiaute com a abordagem Marcela segue o fluxo mostrado na figura 5.1, correspondendo aos seguintes passos:

1. Adaptação da lógica;
2. Assinalamento das CBs;
3. Roteamento da matriz.

Inicialmente, assumamos que o circuito a ser gerado encontra-se descrito no nível lógico estrutural. Um módulo denominado **adaptador de lógica** tem a função de realizar a geração do equivalente Marcela para a matriz alvo. Esta geração é feita mediante a substituição das funções lógicas e blocos funcionais da descrição original pelos equivalentes existentes na biblioteca. Otimizações e remanejos de lógica serão necessários caso uma ou mais primitivas excedam as respectivas capacidades na matriz a ser utilizada. Isto será acusado quando da alocação da porção da matriz para o projeto, a ser realizada como primeiro passo do assinalamento, e por isso denominado pré-assinalamento. Isto sugere um laço de realimentação no fluxo. Em caso de haver mais de um tipo de matriz disponível, ao invés de

iniciar otimizações, o usuário pode simplesmente recomeçar a geração escolhendo outra matriz e avaliando o resultado do pré-assinalamento. Da comparação entre os pré-assinalamentos, pode ser escolhida a matriz que oferece melhores chances de acomodar o circuito.

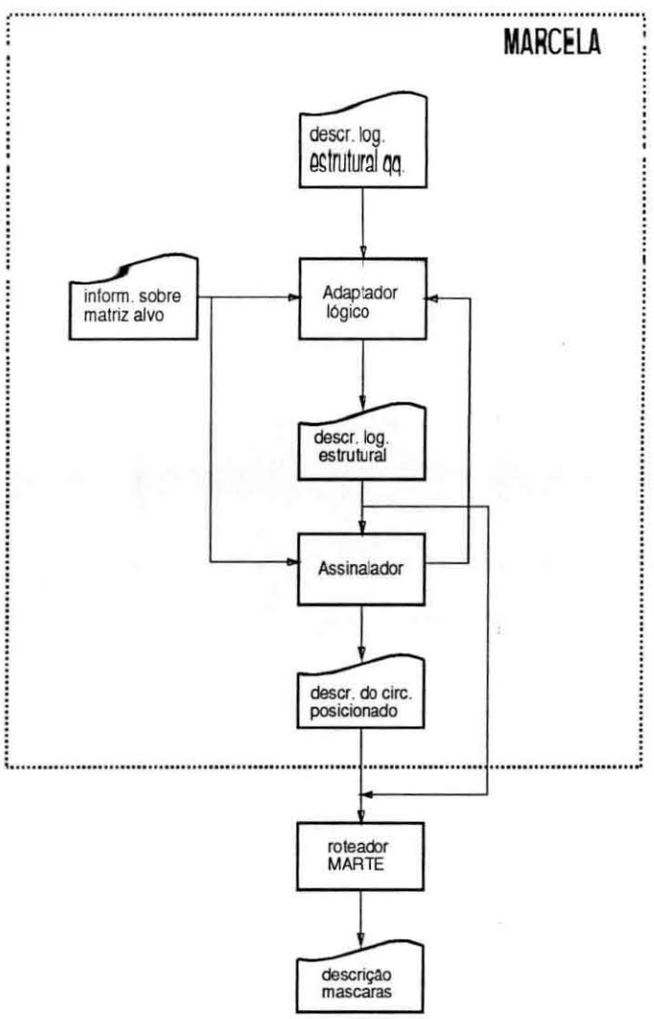

Figura 5.1: Fluxo de projeto para circuitos segundo a abordagem Marcela.

O assinalamento é realizado pelo módulo **assinalador** e consiste em alocar uma região da matriz, mapeando sobre esta as primitivas existentes na descrição equivalente Marcela. As informações necessárias para a realização desta tarefa são basicamente topológicas e descrevem cada matriz disponível. Os procedimentos de assinalamento são detalhados na próxima seção.

O roteamento do circuito é realizado pelo roteador MARTE [JOH 92a] [JOH 92b], o qual foi desenvolvido para ser utilizado tanto no Marcela como no

TRAMO II, ou ainda em qualquer abordagem *cell based* na qual as células e as conexões possam ser mapeadas por uma grade de restrições. Trata-se de um roteador tipo *maze* com algumas modificações que visam otimizar a ocupação das trilhas e o tempo de computação. Mais detalhes do MARTE serão apresentados na seção 5.3.

Na figura 5.1, os módulos de uso específico da abordagem Marcela encontram-se envolvidos pelo retângulo pontilhado. Tais módulos levam em conta as características singulares da abordagem. Porém, como o diálogo entre eles se dá por troca de arquivos, e sendo o Marcela parte do sistema TRANCA, houve um esforço no sentido de padronizar os formatos. Esta padronização pôde ser flexibilizada no caso dos arquivos envolvidos serem de uso exclusivo do Marcela. Mas no caso de arquivos para geração do roteamento e sua posterior expansão, a padronização foi imprescindível.

O esforço de padronização foi estendido às interfaces, com vistas a facilitar a integração de um novo sistema TRANCA ou mesmo de partes deste com sistemas comerciais. Por hora, as providências tomadas neste sentido foram abolir a linguagem RS para a descrição de leiautes de circuitos integrados [TOD 86], substituindo-a por CIF, e adotar o formato Spice para descrição das redes. Como RS implementa um subconjunto de comandos CIF, não há nenhum prejuízo em realizar tal troca. Além disso, CIF é um formato bastante difundido internacionalmente, o que favorece a integração das ferramentas de síntese localmente desenvolvidas com ferramentas desenvolvidas em outros centros de pesquisa.

Já a opção pelo formato Spice não foi tão natural. Embora a linguagem NILOTRANCA fosse bastante conveniente para a geração de leiaute, o fato de ser desenvolvida localmente dificulta a integração de uma ferramenta que a utilize. Além disso, seu uso implica em dois problemas associados à simulação. Primeiramente, é desconhecida a influência do subconjunto de comandos NILO e das diretivas acrescentadas para a especificação de NILOTRANCA quando do uso do simulador lógico do ambiente AMPLO. E o segundo problema diz respeito à necessidade de um conversor NILOTRANCA-Spice para permitir que sejam realizadas simulações

no nível elétrico.

Cogitou-se então, adotar algum formato que fosse um padrão internacional e que ao mesmo tempo permitisse a integração imediata com o maior número de ferramentas comerciais dentre as disponíveis no GME. Embora VHDL esteja se consolidando como padrão, ainda são poucas as ferramentas comerciais que a utilizam, principalmente no nível estrutural. Como segunda opção, surgiu EDIF, para a qual há conversores em muitos pacotes comerciais. Porém, esbarrou-se no problema de haver muitas versões deste formato. A adoção de um subconjunto que satisfizesse às condições propostas anteriormente exigiria uma decisão no nível do Grupo de Microeletrônica, de forma a estabelecer-se um padrão verdadeiro.

Em razão destas dificuldades, optou-se por assumir o formato Spice como padrão interno para descrição de redes no novo sistema TRANCA. Tal opção não inviabiliza a adoção de um padrão externo, tal como o próprio EDIF, pois neste caso a descrição seria convertida uma única vez para o formato Spice, caso fosse desejável permitir simulação elétrica do circuito antes ou após a geração. Nesta definição de formato, considerou-se ainda a iminente descontinuidade do módulo TRAMO, o qual utiliza NILOTRANCA, e sua substituição pelo TRAMO II. Como o módulo TRAGO já utiliza o formato Spice, a padronização interna do TRANCA estaria assegurada.

A utilização do formato Spice ainda traz consigo algumas vantagens adicionais. Para o sistema TRANCA rodando em equipamento compatível com IBM/PC, a edição do esquemático pode ser realizada com o uso do editor ESQUELETO, para o qual existe um gerador de formato Spice. No caso do TRANCA rodando em ambiente SUN, pode-se editar o esquemático com o editor do sistema SOLO2000 (CADENCE) e após, utilizar um conversor de formato SILOS-Spice, já existente no âmbito do GME. Seja qual for o caso, o conforto do projetista estará assegurado.

Os problemas de simplificação e uso de lógica degenerada ou remanejo de

funções são consideravelmente difíceis de serem automatizados. Embora seja possível preverem-se benefícios decorrentes das simplificações, optou-se por concentrarem-se esforços na geração do leiaute propriamente dito, contingenciando o desenvolvimento de ferramentas para tratar a lógica aos resultados obtidos nesta primeira etapa. Porém, o usuário possui o recurso de optar entre as versões disponíveis na biblioteca de equivalentes, podendo gerar vários pré-assinalamentos somente com a troca destes.

Porém, caso ainda necessite reduzir área, o projetista poderá aplicar as transformações mais convenientes, dentre aquelas apresentadas no capítulo anterior.

A automatização da geração de circuitos segundo a abordagem Marcela foi feita tomando-se a matriz MAR1000 como exemplo. Porém todos os procedimentos levam em consideração a elaboração de futuras matrizes. É interessante notar que a própria ferramenta de assinalamento permite simularem-se novas organizações de matrizes, sem contudo projetá-las. Basta que os arquivos de descrição sejam confeccionados.

Obviamente, existe um conjunto mínimo de características *sine qua non* a serem observadas para implementação de novas matrizes. Dentre estas, citam-se o uso de uma grade de roteamento, que forneça grande flexibilidade e uso de UBs com formato mais retangular possível (ou pelo menos, simétrico). Há ainda um conjunto de recomendações, cujo não cumprimento não inviabiliza a automatização, mas prejudica a ocupação de matrizes. Fazem parte deste conjunto, primitivas muito complexas, UBs compostas por um número grande de CBs e desproporção entre transparência vertical e horizontal.

5.1 Estratégias para o Assinalamento de Células

A descrição Spice de um circuito a ser gerado contém informações que não serão utilizadas para a síntese, mas que interessam à simulação elétrica. Ao assinalador e ao roteador interessa saber:

- os nomes das células que compõem o circuito;
- o tipo de cada célula;
- a lista das redes existentes.

A biblioteca de equivalentes contém descrições Spice a partir de subcircuitos, os quais correspondem às primitivas disponíveis na matriz utilizada. Assim, a descrição de cada subcircuito, em termos de transistores, os quais passaremos a chamar **circuitos folha**, não é levada em conta pela geração do leiaute, pois as primitivas já possuem estruturas definidas pelas CBs. Além disso, os dados sobre tamanhos dos transistores nas CBs estão contidos nas descrições de subcircuitos.

Os **nomes das células** correspondem aos nomes dados às instâncias dos circuitos folha, os quais, pela sintaxe Spice, devem iniciar pela letra **X** (maiúscula), seguida de um número. No caso da descrição ser plana, só haverá um nível hierárquico, e cada célula corresponde diretamente à instância de uma primitiva. Porém, se houver mais níveis hierárquicos, será necessário planificar a descrição, batizando as células com seus **nomes absolutos**, formados pela concatenação dos nomes de todos os circuitos hierarquicamente superiores. Assim, se na descrição de um circuito houver um registrador de nome **reg1**, formado por *flip-flops* **ffdrs**, uma instância **X1** pertencente a **reg1** chamar-se-ia **X1.ffdsr.reg1**.

O **tipo de cada célula** é o nome da célula folha de sua instância, o qual sempre corresponde ao nome da primitiva por ela implementada.[1] No caso da

[1] Por simplicidade e consistência.

matriz MAR1000, os tipos existentes são inversor (inv), *nand*, *nor* e *transmission gate* (tg).

Como o formato Spice descreve redes, a **lista das redes** é facilmente obtenível, pois após o nome da instância, segue a lista ordenada de nós aos quais seus pinos estão conectados.

O problema do assinalamento assemelha-se muito ao posicionamento de células. Se encararmos a própria UB como sendo uma célula, o problema parece ficar extremamente simplificado, pois todas as células teriam mesmo tamanho, bastando arranjá-las sobre a superfície da matriz, de modo a ocupar a menor área possível. Por outro lado, a dificuldade fica transferida para o assinalamento interno à cada UB. A pergunta que surge é: qual a heurística mais apropriada para alocar as CBs, de modo a reduzir desperdício em cada UB e ainda observar a natureza de agrupamento de primitivas? A idéia de gerar *clusters* com capacidade menor ou igual à de uma UB para posterior particionamento da superfície não parece ser a melhor solução. Provavelmente, os resultados não justificam a complexidade do algoritmo de gerência de ocupação das UBs.

Refletindo melhor sobre a natureza da abordagem Marcela e sobre os aspectos práticos dos pré-difundidos, conclui-se que há três situações possíveis de ocorrer na geração de circuitos:

- Ou a matriz escolhida comporta plenamente o circuito a ser projetado, não havendo nenhuma preocupação quanto à taxa de ocupação;
- Ou o circuito encontra-se no limite da capacidade da matriz e é necessária certa cautela quanto à sua ocupação;
- Ou a matriz escolhida não tem capacidade para acomodar o circuito.

A menos que a discrepância entre complexidade do circuito e capacidade da matriz escolhida seja muito grande, a última situação só será constatada após

várias tentativas de geração com opções diferentes e transformações lógicas.

Na primeira situação, a única preocupação seria assegurar um bom desempenho elétrico, já que não há nenhuma restrição quanto à porção de matriz a ser alocada.

Voltemo-nos então à segunda opção, a qual deve guiar o procedimento de assinalamento. Cada matriz possui uma capacidade total, a qual pode ser decomposta em capacidade individual por primitiva. Se as capacidades individuais forem suplantadas, então é necessário apelar para transformações. Se os números de primitivas do circuito estiverem próximos das capacidades individuais, é necessário otimizar a alocação a fim de torná-la possível. O procedimento de assinalamento é então decomposto em duas etapas: **alocação da matriz** e **otimização da ocupação**. Esta forma de abordar o problema é bastante hortodoxa, encontrando correspondentes no posicionamentos, a saber: geração de um posicionamento inicial mediante o uso de algum **algoritmo construtivo** e otimização de tal posicionamento por meio de um **algoritmo iterativo** [PRE 88].

5.1.1 Alocação da matriz

A alocação da matriz é a denominação dada à geração de um assinalamento inicial e arbitrário cujas funções são prover uma solução inicial a ser otimizada e determinar a menor porção de matriz capaz de comportar o circuito. Esta última função envolve um cálculo denominado **pré-assinalamento**, já referido anteriormente, e que vai servir como métrica para a possibilidade de geração.

O princípio do assinalamento é bastante simples: dado um circuito a ser gerado e escolhida uma matriz, haverá pelo menos uma primitiva que limita o processo de alocação. A esta primitiva denominamos **elemento limitante** (ou

limitante, somente).[2]

Para calcular o limitante, basta realizar a razão entre a quantidade de cada primitiva existente na descrição equivalente pela quantidade de tal primitiva na UB, ao qual chamaremos **número limitante**. O limitante será aquela primitiva que apresentar o maior número limitante. Ocupando a menor quantidade de UBs de forma a comportar o limitante, assegura-se a otimização do leiaute como um todo. Como exemplo, tomemos o equivalente Marcela (matriz MAR1000) para o circuito Modem [REI 89], cuja composição, em termos de primitivas, é:

- 24 *nand*;
- 21 *nor*;
- 71 inversores;
- 59 *transmission gates*.

A UB da matriz MAR1000, por sua vez, apresenta a seguinte quantidade de primitivas:

- 1 *nand*;
- 1 *nor*;
- 2 inversores;
- 2 *transmission gates*.

Calculando-se os números limitantes obtém-se 35,5 para o inversor como sendo o maior número limitante. O **máximo limitante** B será o maior número limitante, caso este seja inteiro, ou o primeiro inteiro maior. Então, para este equivalente do circuito, o limitante é o inversor e B vale 36.

[2] A probabilidade de haver mais de um limitante é muito reduzida, mas se houver, poderemos escolher um de forma arbitrária, sem perda de precisão.

A seguir, são calculados os números de UBs nas direções **x** e **y**, de modo a atingir-se a relação de aspecto mais próxima de 1 possível. A relação de aspecto é a razão entre a dimensão **y** e a dimensão **x** da envoltória do leiaute. As relações a serem satisfeitas são:

$$n_x \cdot n_y = B \tag{5.1}$$

$$n_x \cdot L_x = n_y \cdot L_y \tag{5.2}$$

onde n_x e n_y são os números de CBs na horizontal e na vertical, respectivamente, e L_y/L_x é a relação de aspecto da UB, a qual vale 1,67 para a matriz MAR1000.

Voltando ao caso do circuito Modem, os valores calculados são $n_x = 4,7$ e $n_y = 7,7$. Então é realizado um arredondamento tal que o produto de UBs $N_x.N_y$ seja mínimo. Há quatro possibilidades de arredondamento:

- N_x é primeiro inteiro menor que n_x e N_y é o primeiro inteiro maior do que o resultado de B/N_x;
- N_x é primeiro inteiro maior que n_x e N_y é o primeiro inteiro maior do que o resultado de B/N_x;
- N_y é primeiro inteiro menor que n_y e N_x é o primeiro inteiro maior do que o resultado de B/N_y;
- N_y é primeiro inteiro maior que n_y e N_x é o primeiro inteiro maior do que o resultado de B/N_y.

Para o Modem, os valores encontrados são N_x=9, N_y=4 e o número de UBs alocadas será igual a 36. Com estes dados já é possível calcular a área ocupada pelo circuito: 604,8 x 446,4μm^2 (ou 0,275 mm^2).

Feito o pré-assinalamento, resta completar o assinalamento inicial, provendo uma posição para cada primitiva. A alocação pode ser feita segundo várias heurísticas [PRE 88]. A mais simples é a seqüencial, a qual aloca para cada célula

lida da descrição, a próxima CB livre que corresponda ao tipo desejado. Embora seja de fácil programação, esta técnica geralmente produz resultados pobres, dificultando o refinamento posterior. Outra técnica de alocação pode ser a por geração de aglutinados (*clusters*). Neste caso, é escolhida uma célula para servir de semente e após, células são selecionadas de acordo com o grau de conectibilidade com a semente que já se encontra assinalada. Esta técnica pode ainda apresentar muitas variantes, pois há muitas formas possíveis de geração de sementes e de escolha da próxima CB a ser ocupada.

5.1.2 Otimização da ocupação

O método de otimização da ocupação da matriz utiliza um algoritmo de **trocas** [HAN 76] aplicado a blocos gerados mediante particionamento por **quadratura** [BRE 77].

Um método de trocas consiste basicamente em selecionar um elemento por vez como sendo o primário, o qual terá sua posição trocada com cada um dos demais. Se alguma das tentativas resultar em melhora do posicionamento, então as novas posições são mantidas, senão as posições anteriores são restauradas. Como em cada ciclo todos os $n(n-1)/2$ pares são testados, a complexidade do algoritmo é proporcional a n^2, o que torna sua aplicação direta impossível para circuitos muito grandes. A solução é restringir o conjunto de tentativas de trocas, restringindo-se a vizinhança do elemento [IOS 83]. A partir de um tamanho de vizinhança fornecido pelo usuário, o programa escolhe os elementos candidatos e realiza a mesma seqüência de operações de troca de pares. Um ciclo é completado quando cada elemento tiver sido escolhido como primário. O tempo de computação é proporcional a $mn/2$, onde n é o número de elementos primários e m é o tamanho da vizinhança.

Os métodos de partição de circuitos não são por si só algoritmos de otimização. Eles constituem uma sistematização do traçado de linhas de corte que

objetiva definir blocos aos quais serão aplicados os métodos de trocas. Sua freqüente utilização deve-se ao fato de possibilitarem uma visão global dos elementos a serem posicionados e das respectivas conexões. Os elementos a serem posicionados são divididos em dois ou mais grupos, denominados partições, de forma sistemática até que cada partição possua um único elemento ou um número pequeno de elementos. Esta abordagem descendente permite levarem-se em consideração primeiro porções maiores do circuito, chegando ao final a grupos de elementos. O uso associado de um algoritmo de trocas promove a permanência dos elementos fortemente conectados num mesmo bloco da partição, o que corresponde a formar *clusters* de células fortemente conectadas.

No método da quadratura (figura 5.2), a superfície de leiaute contendo todos os elementos é bisseccionada por uma linha de corte vertical c_1, produzindo os blocos B_1 e B_2. Após serem realizadas as trocas de elementos, os novos blocos são redivididos por uma linha de corte horizontal c_2, gerando os blocos B_3, B_4 e novos blocos B_1 e B_2, diferentes dos anteriores. O procedimento é aplicado recursivamente até que cada elemento esteja posicionado ou até que cada bloco atinja o tamanho mínimo desejado.

Segundo Corrigan, o efeito da imposição de linhas de corte numa dada direção é diminuir o comprimento das conexões na direção perpendicular à linha de corte considerada [COR 79]. Mas mais importante do que isso, a diminuição do número de redes cortadas implica na diminuição do congestionamento de conexões no sentido perpendicular à linha de corte. Então, alternando-se linhas de corte horizontais com verticais estaremos homogeneizando a distribuição de conexões entre os dois sentidos, o que está em perfeita concordância com a estratégia de ocupação Marcela.

A aplicação da quadratura a uma superfície Marcela respeita a integridade das UBs, isto é, as posições possíveis para linhas de corte são as fronteiras entre UBs. Uma linha de corte nunca secciona uma UB. Utilizando novamente o exemplo do equivalente Marcela do circuito Modem, nota-se que o número de UBs

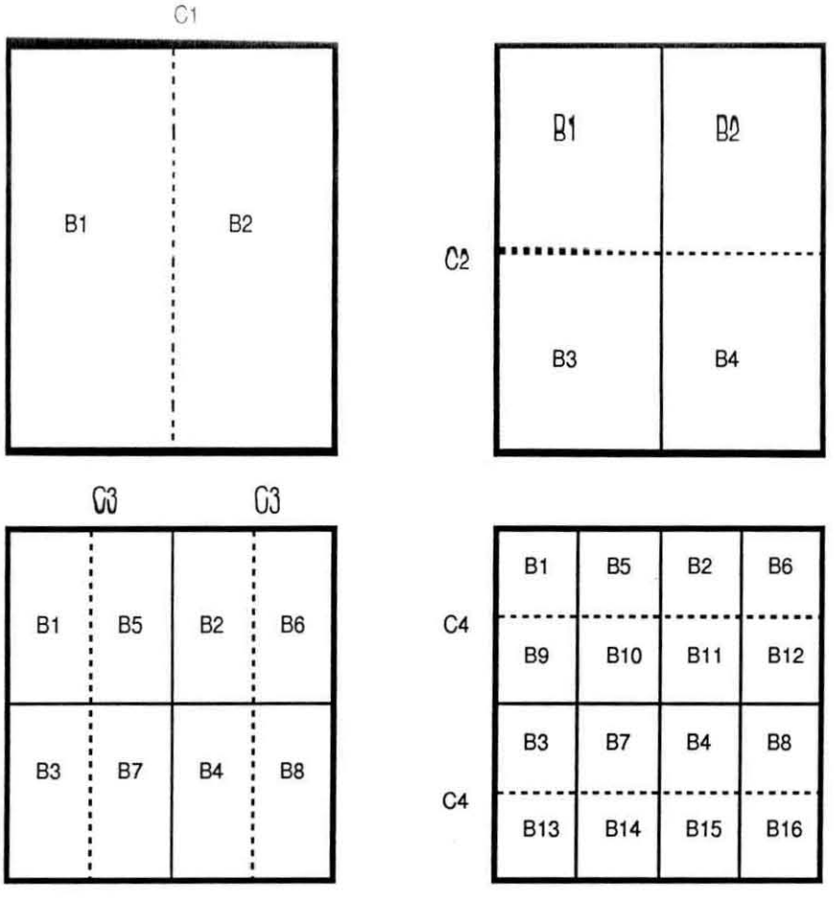

Figura 5.2: Particionamento por quadratura.

na horizontal é ímpar. Então, o seccionamento se dará de modo a dividir os blocos de forma desigual, mas mais próxima da metade. Ao resíduo final do particionamento, é aplicada uma última linha de corte. A figura 5.3 mostra a seqüência das linhas de corte no particionamento do exemplo em questão.

A seqüência das linhas de corte é padrão. Ao iniciar o processo, a **profundidade** máxima do particionamento, que vem a ser o número de níveis de particionamento possíveis, é selecionada pelo usuário, ou lida do arquivo de configuração. No caso da figura 5.3, a profundidade máxima é 6. Então, o usuário pode interferir na granularidade das otimizações definindo a profundidade desejada. Caso contrário, o programa assume a profundidade definida no arquivo de configuração.

A otimização por meio de trocas é realizada sobre cada uma das partições seccionadas por linhas de corte e de forma seqüencial, uma partição após a outra. As trocas de elementos se dão somente entre pares de blocos de uma mesma partição.

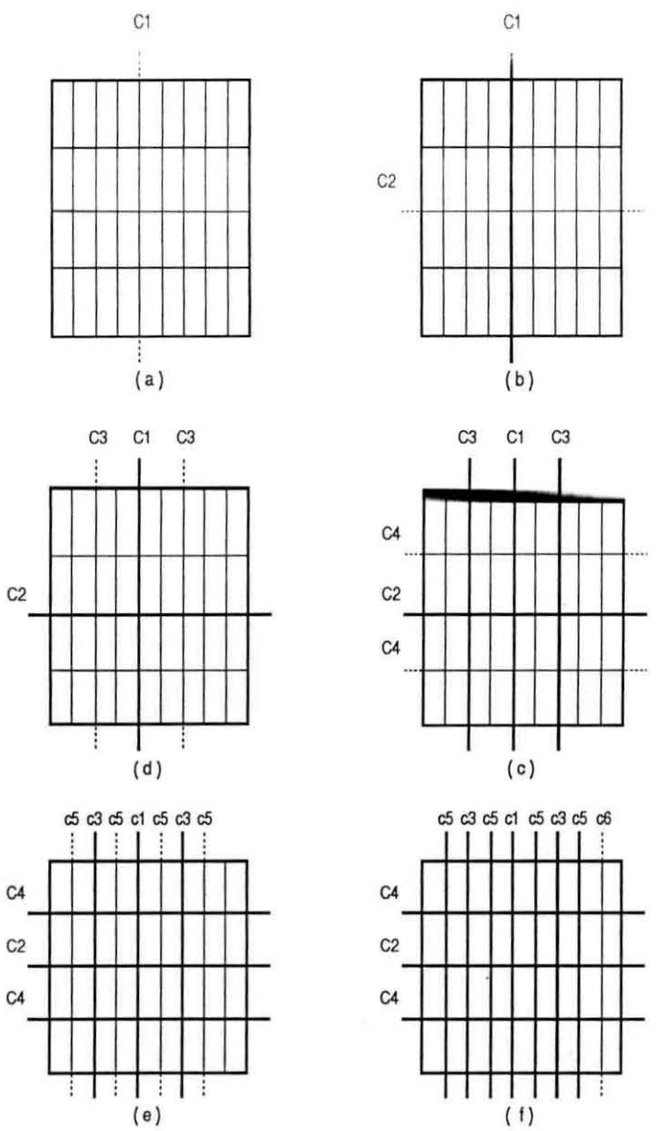

Figura 5.3: Quadratura sobre um leiaute Marcela.

Depois que todas as partições estejam otimizadas, a superfície é submetida a mais um particionamento. Para tanto, os blocos anteriormente criados são promovidos a partições, as quais serão bisseccionadas pelo novo conjunto de linhas de corte, formando novos blocos.

O objetivo das trocas é aglutinar os elementos mais fortemente conectados, passando-os para um mesmo bloco. Considerando-se que tal objetivo seja alcançado, a complexidade do roteamento diminui sensivelmente, pois um número grande de conexões tem seus comprimentos diminuídos devido à proximidade das células a serem conectadas. Esta diminuição dos comprimentos também é favorável

sob o ponto de vista da ocupação de trilhas, já que o número de conexões longas é minimizado.

Considerando-se uma partição cortada pela linha c, elementos são trocados entre os dois blocos resultantes com o intuito de diminuir o número de redes seccionadas por c. Somente os elementos conectados às redes seccionadas são candidatos a trocas. No caso Marcela, há três etapas a serem cumpridas na otimização de uma partição: uma etapa de tratamento de interfaces e duas etapas de trocas, baseadas na heurística de Fiduccia-Matheyses [FID 82].

No tratamento de interfaces, os elementos conectados a redes de interface são identificados e fixados no bloco conveniente. Por exemplo, se a linha de corte for vertical, serão tratados os elementos que se conectam a redes de interface leste e oeste: os elementos da rede oeste devem ser passados para o bloco da esquerda e os da rede leste para o bloco da direita. Por ordem, são tentadas trocas simples (caso haja excedente das CBs desejadas no bloco destino) e trocas de pares, tipo a tipo. Caso o bloco de destino não comporte mais aquele tipo de célula que se conecta à rede de interface, então ela permanecerá no bloco onde está. Terminado este tratamento de interfaces, os elementos de interfaces são tornados fixos. Mesmo que possuam redes seccionadas, eles não serão mais candidatos às trocas.

Na segunda etapa, são realizadas as trocas simples de elementos. Para cada bloco, é feito o balanço de CBs disponíveis. Então, para cada bloco são calculados os ganhos individuais das trocas para todas as candidatas, as quais são ordenadas. O processo de trocas propriamente dito é iniciado escolhendo-se a célula de maior ganho dentre os dois blocos e passando-a para o outro bloco, caso haja CB do seu tipo disponível. Caso contrário, é tentada a próxima célula. Se, efetuada a troca, o número de redes seccionadas diminuir, então a célula trocada é fixada no novo bloco e os ganhos são recalculados para que nova célula seja selecionada. Caso contrário, a célula trocada é temporariamente fixada, os ganhos são recalculados e nova troca é realizada. Se o número de redes diminuir, tais trocas são transformadas em definitivas. Porém, se o número de redes seccionadas não diminuir, a

seqüência de trocas continua até que o valor máximo de trocas seja atingido ou o número de redes cortadas diminua. Se ainda assim, não houver melhora, então a primeira situação é reestabelecida, com a primeira célula escolhida sendo fixada no bloco de origem e a próxima célula sendo escolhida para troca. As tentativas de trocas páram quando não houver mais células candidatas. O procedimento de **relaxamento** da solução visa escapar dos mínimos locais para tentar alcançar mínimos globais. Note-se ainda que as trocas simples permitem um espalhamento no assinalamento de CBs, já que estas haviam sido concentradas na porção inferior da matriz pelo assinalamento inicial.

Esgotadas as trocas simples de elementos, serão tentadas trocas de pares, considerando que os pares devam ser do mesmo tipo. Esta etapa objetiva realizar alguma otimização nas posições dos elementos pertencentes ao tipo limitante, pois não é garantido que tenha ocorrido alguma otimização destes na etapa anterior e o motivo é simples: caso o pré-assinalamento alocar um número de UBs capaz de comportar o número exato de células do tipo limitante (nem mais nem menos), então nenhuma troca simples será possível a elas. O procedimento de trocas é o mesmo da etapa anterior. A diferença é que para cada célula de um bloco, são identificadas as possíveis companheiras no outro bloco. Os ganhos dos pares são calculados e então é escolhido o par de maior ganho. O processo pára quando todos os pares tiverem sido tentados.

O ganho individual é calculado pela seguinte expressão, proposta em [LUB 90]:

$$C_i = CE - CI \tag{5.3}$$

onde CE é o custo externo da célula, igual ao somatório dos pesos das redes que a conectam exclusivamente a células do outro bloco e CI é o custo interno, igual ao somatório dos pesos das redes que a conectam exclusivamente a células do mesmo bloco.

As trocas de pares são implementadas como duas trocas simples suces-

sivas. Tal implementação permite grande simplificação na estrutura de dados do programa e constitui-se numa aproximação razoável.

Os pesos *default* adotados no cálculo dos ganhos das trocas são 1. Porém, está previsto que as redes internas de qualquer equivalente da biblioteca possuam pesos maiores. O motivo é incentivar a aglutinação das primitivas que se originaram da decomposição de uma função lógica mais complexa, facilitando o roteamento automático e favorecendo o desempenho elétrico do circuito. Por exemplo, as células que compõem um *flip-flop* tenderão a permanecer próximas, pois as redes que as conectam possuem peso maior.

Para permitir a identificação das redes de interface e das redes com peso diferente do *default*, são inseridas diretivas na descrição Spice, as quais o simulador encara como simples comentários. Para interface, cada rede é identificada pelo nome com que aparece na *netlist* após a diretiva "*interface:". Para alterar o peso de redes basta acrescentar a diretiva "*netweight:", seguida do valor, e ao seu lado, a diretiva "*list:", com a lista dos nomes das redes que possuem tal peso. No Anexo A-1, vê-se a descrição Spice para o circuito Modem.

5.2 O Assinalador MARLA

A fim de avaliar as estratégias de assinalamento descritas anteriormente, foi implementado um protótipo inicial de assinalador denominado MARLA (MARcela Layout Assigner). Esta versão inicial realiza as etapas de assinalamento inicial e refinamento por meio de trocas de células, porém não trata os pinos de interface do circuito.

Inicialmente, a descrição do circuito em formato Spice plano é lida e duas listas são criadas: uma lista de células e uma de redes. Cada célula da lista de células possui ponteiros para as redes que a conectam com as demais células. De

modo semelhante, cada rede da lista de redes possui ponteiros para as células por ela conectadas. Esta estrutura agiliza as pesquisas quando da formação das partições e trocas de células entre estas. A figura 5.4 mostra esta estrutura duplamente relacionada.

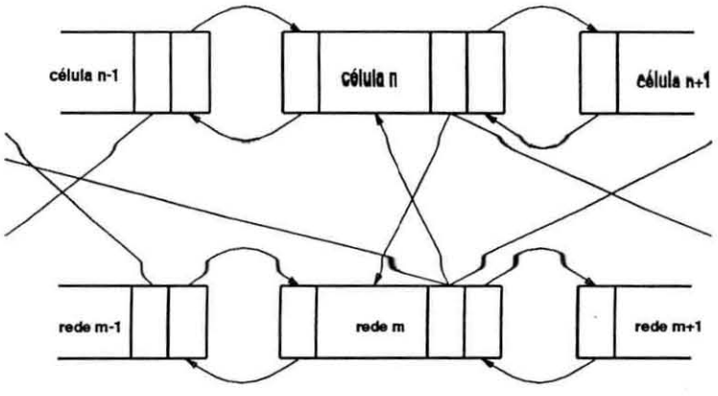

Figura 5.4: Lista de redes e lista de células.

Outra estrutura de grande importância é a matriz de ponteiros, a qual é responsável pela representação interna da matriz de células. Cada elemento da matriz é um ponteiro para uma célula da lista de células, respeitando o tipo de CB que ela representa. Quando nenhuma célula é assinalada a uma dada CB, o ponteiro correspondente possui valor nulo (*null*). Note-se que a matriz de ponteiros e a matriz de células (física) estão intimamente relacionadas.

Uma terceira estrutura foi criada para facilitar as trocas de células. Quando o circuito é particionado em dois blocos, os ponteiros de células dos blocos são copiados para um vetor de ponteiros. Este vetor é mantido ordenado de acordo com os ganhos das células. Cada troca de célula é efetuada mediante a troca de um *flag* no primeiro elemento livre do vetor. Quando todas as trocas terminaram, as posições das células são atualizadas na matriz e o vetor é apagado.

Para a alocação da matriz (assinalamento inicial), foi implementado um algoritmo baseado em crescimento de aglomerados. A primeira posição da matriz (x=y=0) é preenchida com a primeira célula da lista de células. Após, a próxima célula a ser assinalada é escolhida por uma função recursiva a qual seleciona, dentre as células ainda não assinaladas, aquela que possui o maior número de conexões em

comum com as já assinaladas. Feito isto, a função invoca a si própria para assinalar a célula escolhida e para escolher a próxima célula a ser assinalada. Cabe ressaltar que existe um tamanho máximo para os aglomerados gerados. Quando este tamanho é atingido, uma nova semente é selecionada e o processo é reiniciado. O uso de mais de uma semente evita que o processo de *clusterização* de células se degenere, pois à medida em que o número de células já assinaladas cresce, também cresce o número de células não assinaladas que possuem muitas conexões com as já assinaladas. A figura 5.5 ilustra o procedimento de alocação da matriz.

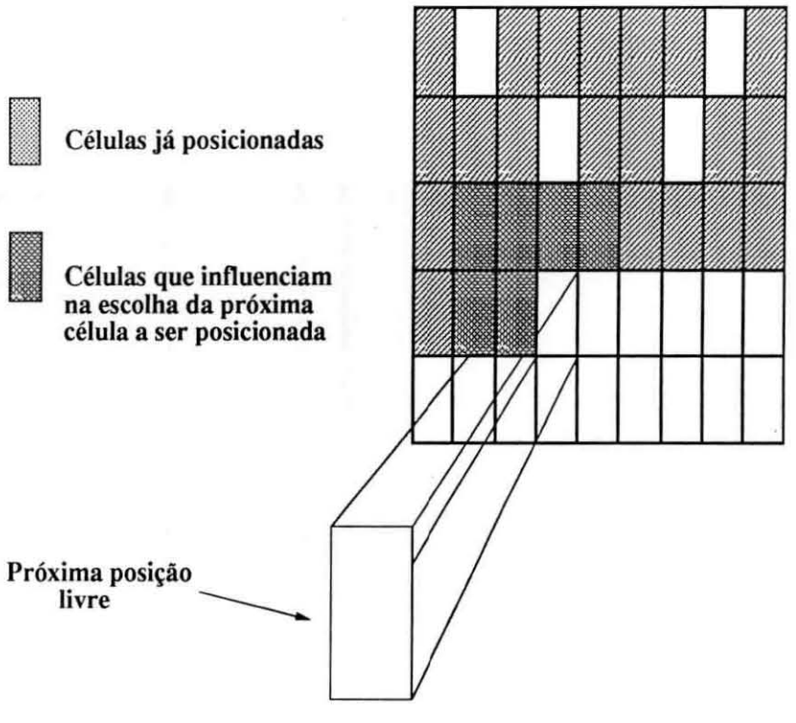

Figura 5.5: Procedimento para alocação de matrizes Marcela.

Os procedimentos de particionamento e trocas de células entre partições implementados são exatamente os descritos na seção 5.1.2. Nesta fase, o vetor de ponteiros descrito anteriormente é utilizado. As trocas individuais de células utilizam a primeira célula livre deste vetor, ou seja, a de maior ganho. Para as trocas de pares, o procedimento é essencialmente o mesmo, exceto que para cada tentativa de troca, é escolhido um par de células livres do mesmo tipo, mas pertencentes a blocos diferentes.

Após todas as trocas terem sido realizadas, o vetor de ponteiros para células é utilizado para atualizar a matriz, e após, é apagado.

O usuário pode interferir no processo de assinalamento por meio dos seguintes parâmetros:

- Tamanho da matriz;
- Número máximo de trocas;
- Profundidade do particionamento.

O tamanho da matriz é definido em termos de UBs, nas direções horizontal e vertical. O número máximo de trocas diz respeito à quantidade de tentativas de trocas realizadas antes que o algoritmo abandone uma determinada seqüência. A profundidade do particionamento equivale ao tamanho da menor partição, o que é dado em número de UBs nas direções x e y. Caso o usuário não informe estes dados ao programa, eles serão lidos do arquivo de configuração, o qual possui ainda outras informações que devem ser transparentes. Dentre estas estão os tipos de CBs (nomes dos arquivos) que compõem a UB da matriz utilizada e sua topologia, os nomes dos arquivos que contém as restrições de roteamento e o número de trilhas horizontais e verticais da UB. Estas informações só podem ser alteradas mediante edição do arquivo de configuração e estão relacionadas com a geração do arquivo que descreve posicionamento para o roteador MARTE. O Anexo A-2 mostra um arquivo de configuração para a matriz MAR1000.

5.3 O Roteador MARTE

O roteamento dos circuitos Marcela é realizado com o uso do ambiente MARTE [JOH 92b], o qual foi desenvolvido para ser utilizado no sistema TRANCA, embora possa ser utilizado em qualquer leiaute representável por matriz de res-

trições. Suas principais características são a **realimentação**, a **controlabilidade** e a previsão para **diversas opções de algoritmos**. Isto é possível graças à compatibilidade sintática e semântica dos arquivos de entrada e saída do roteamento simbólico. A estrutura do ambiente, com os principais módulos e os arquivos internos são mostrados na figura 5.6.

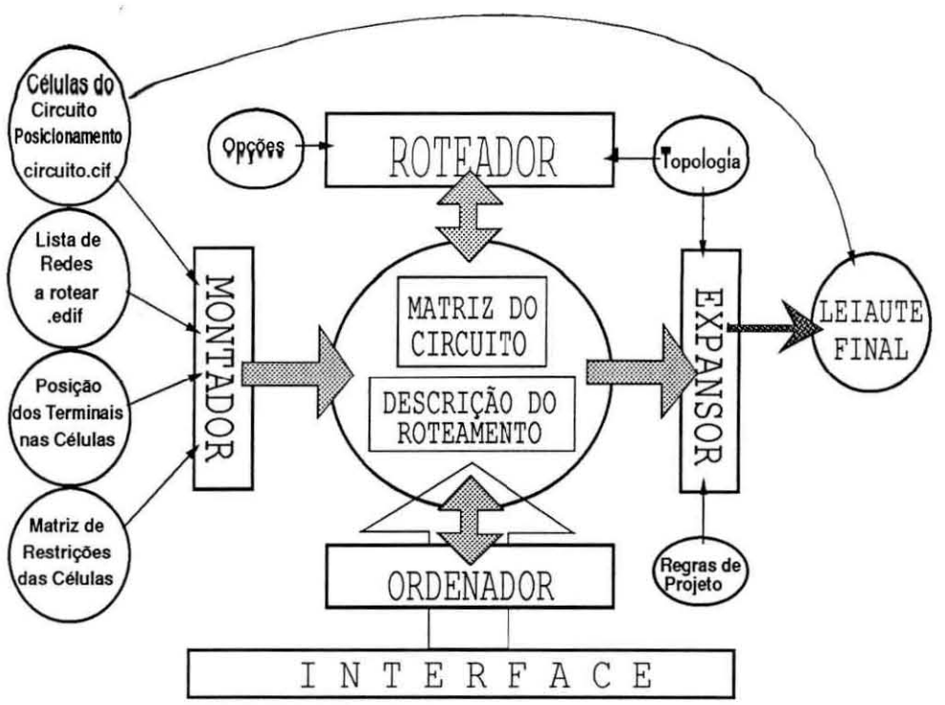

Figura 5.6: Estrutura do ambiente de roteamento MARTE.

A realização do roteamento sobre as células, como prevê a metodologia TRANCA, determina que um número muito grande de restrições tenham que ser levadas em consideração. Diferentemente dos leiautes TRAMO e TRAGO, onde os roteamentos vertical e horizontal são tratados em separado, os leiautes Marcela e TRAMO II oferecem maior liberdade, tratando de forma eqüalitária todas as direções.

Por esses motivos, o MARTE (**MAze RouTing Environment**) [JOH 92b] usa o algoritmo *maze* básico com algumas modificações. Os roteadores *maze* se caracterizam por trabalharem sobre uma grade e utilizar a filosofia do caminho mais curto. A área a ser roteada é dividida em quadrados, denominados **células**, e a

cada um é atribuído um código que permite ou não a passagem ou acesso de conexões. O algoritmo básico, ou de Lee [PRE 88], consiste em marcar as células não bloqueadas a partir do ponto de origem até encontrar o destino, ou cobrir toda a área. A marcação é feita com um número de uma seqüência, permitindo que, uma vez encontrado o destino, o algoritmo possa identificar o caminho mais curto até a origem. Feito isto, as células não alocadas para o caminho da conexão são desmarcadas, enquanto que as utilizadas acrescentam restrições sobre a área por elas ocupada.

No caso das abordagens Marcela e TRAMO II, o roteamento deve ser feito segundo uma grade de metal 1 na horizontal e metal 2 na vertical, e o alinhamento de entradas e saídas é garantido pela topologia das células. Trocas de trilhas só podem ser feitas nas interseções da grade, caso não haja restrição para via. Para que fosse possível abstrair o mundo geométrico para o simbólico, foi criada a matriz de restrições, cuja função é armazenar as restrições para cada ponto da grade. Basicamente, há três tipos de restrições: para metal 1, para metal2 e para via. A granularidade da descrição das restrições está diretamente associada à relação topológica entre a grade de roteamento e o leiaute das camadas inferiores. A figura 5.7 mostra a matriz de restrições para a UB da matriz MAR1000.

A matriz de restrições de um circuito é montada a partir das matrizes de restrições das células que a compõem. Durante o roteamento, as novas conexões vão sendo acrescentadas às restrições presentes na matriz, de acordo com os caminhos encontrados. A matriz, portanto, deve estar presente na memória, sendo gravada posteriormente no arquivo final. Isto implica numa grande área de memória ou *swapping* que o computador deve ter para processar circuitos grandes, pois o acesso à matriz não é seqüencial. A matriz resultante de um roteamento deve permanecer porque o circuito pode ser pós-processado.

Um problema grave dos roteadores *maze* é o bloqueio de entradas por outras conexões. Para evitar isso, o MARTE prevê um mecanismo de reserva de terminais de células, o qual ainda não se encontra implementado. Segundo este

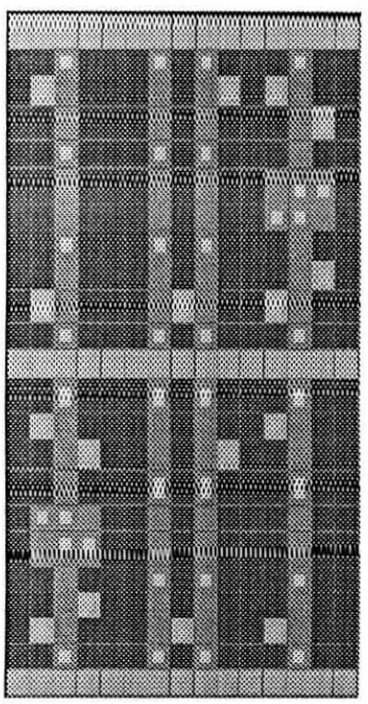

Figura 5.7: Matriz de restrições para a UB da matriz MAR1000.

mecanismo, existem dois tipos de reservas possíveis: uma em que a posição reservada é única para um sinal e outra em que existe um conjunto de terminais possíveis, onde um dos quais deve ser reservado. No primeiro caso, basta colocar uma restrição para o nível de acesso. Para o segundo caso, foi criada uma nova informação armazenada na matriz, que indica a reserva destes pontos. O algoritmo verifica se há restrição para determinado nível e em caso afirmativo, verifica se esta restrição é reserva ou não. Se não for reserva, ele não pode passar, pois a restrição é absoluta. Porém, se a restrição for reserva, então é verificado se os demais pontos do terminal ainda estão livres ou já foram ocupados. Se existir ao menos um livre, o sinal de reserva é transferido para lá, liberando este ponto para uma conexão passar.

Cada ponto da matriz de restrições utiliza 8 bits e armazena restrições para metal 1, metal 2, via, indicação de reserva alternativa, senha (dois bits que indicam célula testada mais os bits de restrição de metal 1 e metal 2 para a senha de 1 bit), indicação de destino e de borda da área do circuito (ou da região a ser pesquisada).

Além do roteador propriamente dito, fazem parte do ambiente uma interface de edição e visualização, o montador da matriz de restrições e um expansor

do roteamento para leiaute de máscaras.

A interface, denominada MEXEM (Matrix EXhibition and Edition Module), tem o propósito de editar matrizes de restrições, inserir redes a rotear e visualizar o roteamento simbólico. Dentre as operações de entrada e saída, citam-se criar, ler e gravar arquivos ou ainda inserir matrizes menores sobre a matriz de trabalho (operação *merge*).

A utilização do roteador demanda que todas as células a serem utilizadas tenham suas matrizes de restrições confeccionadas, o que atualmente é feito manualmente com o uso da interface MEXEM. Juntamente com as restrições, são marcados os terminais de entrada e saída. Está previsto o desenvolvimento de uma ferramenta capaz de mapear automaticamente tais restrições a partir de um arquivo que descreva a grade de roteamento.

O montador é a ferramenta responsável pela formação da matriz de restrições do circuito a ser roteado. Para esta montagem, é criada uma matriz de restrições para o circuito a partir dos arquivos que contêm as restrições das células envolvidas e seus respectivos terminais. A posição simbólica obtida do arquivo de descrição do posicionamento. O arquivo que descreve o posicionamento, por sua vez, é fornecido pelo módulo posicionador (assinalador, no caso Marcela) e deve conter as seguintes informações:

- nome da célula (nome da instância);
- coordenadas simbólicas da célula;
- status em relação a **x** e a **y**;
- tipo da célula (nome da primitiva da biblioteca);
- lista das redes a ela conectadas, em ordem consistente.

O Anexo A-3 apresenta um exemplo de arquivo de descrição de posicionamento.

O expansor tem a finalidade de traduzir a descrição do roteamento simbólico para as coordenadas geométricas. Estra tradução consiste na definição da posição e do tamanho real de cada elemento. Na abstração simbólica, os pontos da grade são formados pela interseção entre as trilhas verticais e horizontais da grade de roteamento. Para o cálculo das coordenadas geométricas, o expansor consulta um arquivo que define a topologia da grade utilizada. Os parâmetros listados neste arquivo possuem o micron como unidade e podem ser identificados na figura 5.8. Para a matriz MAR1000, os valores são os seguintes:

- y1 = 3.0;
- y2 = 3.0;
- y3 = 3.0;
- y4 = 1.8;
- y5 = 3.0.

Os mesmos parâmetros definidos para a direção vertical são aplicados à direção horizontal, bastando substituir **y** por **x** na lista. O Anexo A-4 apresenta o arquivo de descrição de topologia da grade para a matriz MAR1000.

Na fase de expansão são selecionados os níveis a serem expandidos, pois alguns níveis servem apenas de referência interna, tal como terminais em difusão. Ao final do procedimento, é gerado um arquivo formato CIF, contendo a descrição das máscaras do roteamento.

Uma particularidade dos algoritmos tipo *maze* é a sensibilidade à ordem de realização das conexões. Neste sentido, pode-se antever a necessidade de encontrar heurísticas apropriadas às abordagens Marcela e TRAMO II a fim de se implementar um módulo ordenador que gerencie com eficiência os recursos disponíveis em termos de trilhas. Isto corresponde aproximadamente à adoção de novos esquemas de prioridades para roteamentos.

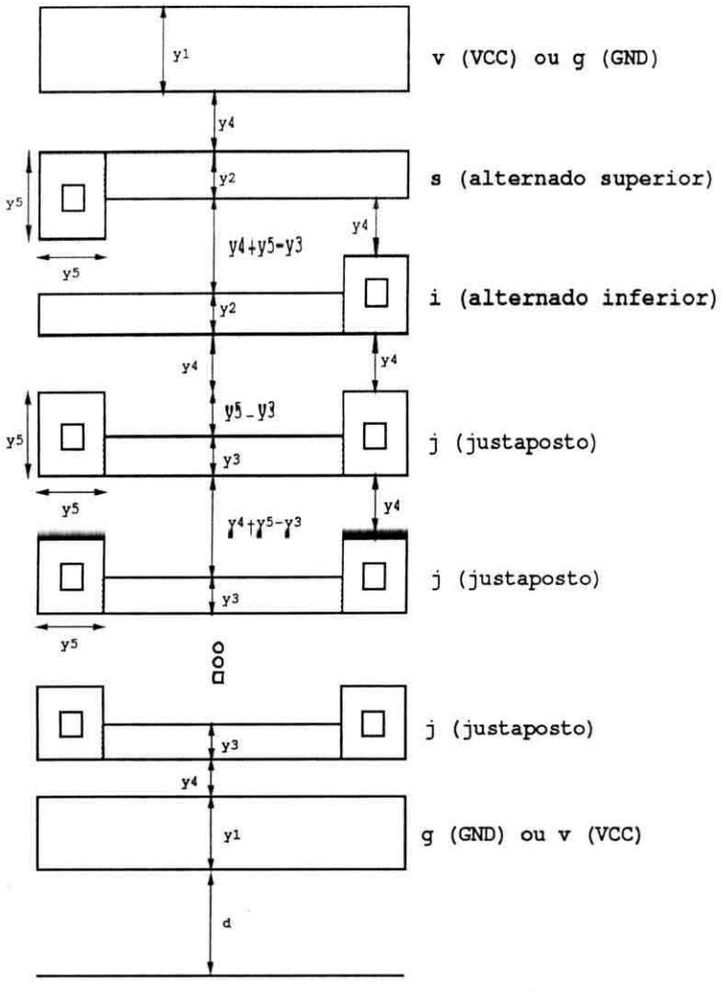

Figura 5.8: Parâmentros para a grade de roteamento.

Na figura 5.9 vê-se o resultado do roteamento simbólico do circuito Modem, versão mínima (9 x 4 UBs), exibido pela interface MEXEM.

5.4 Análise de Desempenho das Ferramentas

Para analisar-se o desempenho das ferramentas utilizadas na geração de circuitos Marcela, foram implementadas três versões do circuito Modem [REI 89], todas partindo do mesmo esquemático lógico: a vesão 1 é totalmente manual, com assinalamento e roteamento manuais, a versão 2 é semi-automática, com o mesmo assinalamento manual, porém com roteamento automático e a versão 3 é automática, com ambos assinalamento e roteamento automáticos. Todas as três versões utilizam

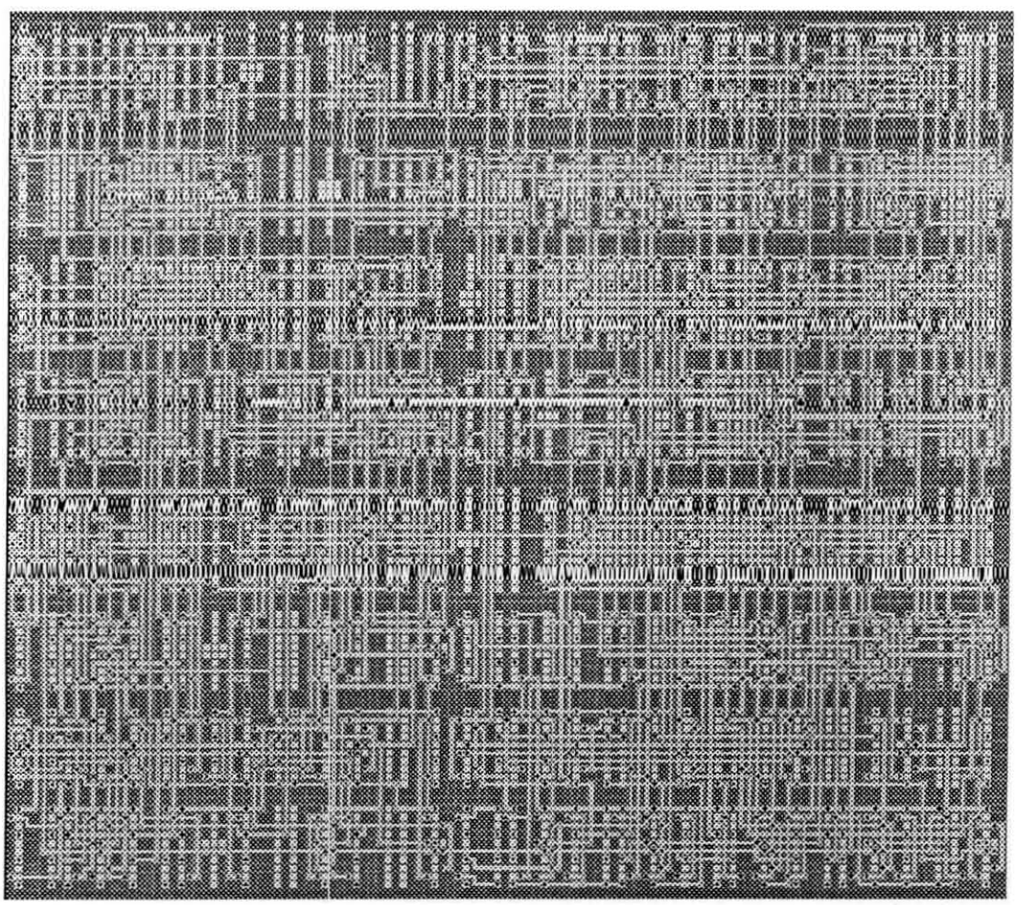

Figura 5.9: Roteamento simbólico do circuito Modem.

o tamanho mínimo para o circuito, ou seja, 9 x 4 UBs. A partir da análise dos leiautes e dos resultados dos roteamentos, em termos de redes roteadas, é possível avaliarem-se as ferramentas individualmente e até mesmo em conjunto.

O circuito Modem mostra-se particularmente interessante por apresentar uma proporção entre tipos de BCs (inv:tg:*nand*:*nor*) igual a 2 : 1.69 : 0.68 : 0.59, muito próxima da proporção existente na matriz MAR1000, a qual é 2 : 2 : 1 : 1.

A tabela 5.1 mostra os resultados obtidos.

A versão 1 provou ser possível alcançar-se 100% de conexões realizadas. Já a versão 2 alcançou 98.7% de conexões completadas, indicando um excelente desempenho do roteador MARTE. À luz deste resultado, pode-se avaliar os 95.8% obtidos pela versão 3. Nota-se que o assinalamento gerado pelo MARLA piora um pouco o desempenho final do processo de síntese. Apesar do resultado obtido poder

versão	1	2	3
área resultante [mm^2]	0.275	0.275	0.275
taxa de ocupação	76.7%	76.7%	76.7%
densidade de transistores [$/mm^2$]	1607	1607	1607
conexões realizadas	100%	98.7%	95.8%

Tabela 5.1: Características das 3 versões do circuito Modem.

ser considerado bom, ele também revela uma certa distância entre o procedimento humano e as heurísticas adotadas para o assinalamento.

Observando-se os leiautes gerados (figura 5.10), nota-se que a versão manual apresenta uma distribuição mais homogênea das conexões. Nas versão 3, existe um certo congestionamento de conexões, provavelmente devido ao aumento do comprimento médio das ligações, o que, por sua vez, é o efeito de um assinalamento nem tão otimizado.

5.5 Sinopse

Este capítulo tem como principais contribuições ao trabalho, o estudo dos algoritmos mais adequados para a síntese de circuitos Marcela, juntamente com as adaptações necessárias, a implementação de um protótipo de assinalador e os testes de desempenho da referida ferramenta. O estudo e previsão das estratégias que darão condições à criação do ambiente Marcela também fizeram parte do escopo do trabalho.

Os resultados de desempenho da ferramenta indicam que ainda existe a necessidade de continuar o estudo de heurísticas que se aproximem mais do procedimento humano, principalmente no que diz respeito à geração de aglomerados (*clusterização*).

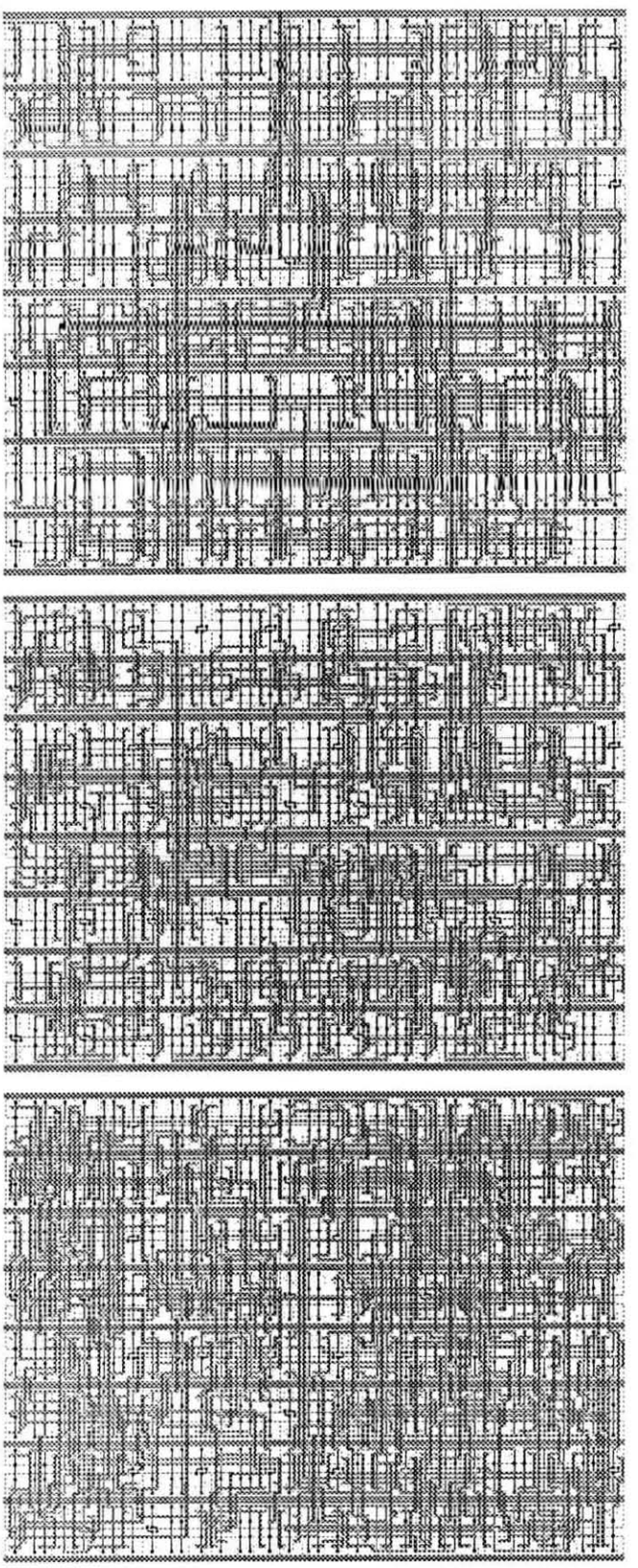

Figura 5.10: Leiautes para as três versões do Modem: versão manual (topo), versão semi-automática (centro) e versão automática (abaixo).

Outra contribuição importante foi a possibilidade de avaliar o desempenho do roteador MARTE, operando em conjunto com uma ferramenta de assinalamento. Cabe ressaltar que os circuitos mais complexos já roteados pelo MARTE até o presente momento foram implementados com a abordagem Marcela.

6 AVALIAÇÃO DA ABORDAGEM MARCELA

Com a finalidade de avaliar o desempenho da abordagem Marcela no que diz respeito à taxa de ocupação, densidade de transistores, transparência dos circuitos, área efetivamente ocupada e percentagem de conexões realizadas automaticamente, foram implementados alguns *benchmarks* de diversas complexidades. As comparações ficaram restritas ao próprio estilo pré-difundido, uma vez que antes de mais nada, é necessário verificar-se a viabilidade dos circuitos Marcela perante outros tipos de pré-difundidos. Além disso, comparações entre circuitos realizados segundo estilos de projeto diferentes não exprimem o real desempenho das abordagens, pois além dos fatores físicos, há ainda os de mercado, os quais geralmente não são considerados nas comparações.

A abordagem Cipredi [CAL 88a] foi eleita concorrente virtual da abordagem Marcela por estar disponível no âmbito do GME/CPGCC. Sua macroarquitetura é do tipo *compacted arrays* e a estratégia de ocupação da matriz evita ao máximo a alocação de bandas para roteamento. Isto geralmente é possível devido ao grande número de trilhas horizontais: 7 sobre os transistores **p**, 5 sobre os **n** e 2 no centro da banda. A CB Cipredi, juntamente com outras características, encontram-se descritas na seção 2.2.2.

A implementação das versões Cipredi utilizou a matriz e a biblioteca de personalização em tecnologia 1.2μm [PAI 91], a mesma tecnologia usada na definição da matriz MAR1000. Por outro lado, a inexistência de ferramentas de PAC para a geração automática com o Cipredi obrigou a utilização de ferramentas desenvolvidas para outras abordagens e estilos. Para amenizar os problemas decorrentes deste fato, escolheram-se ferramentas cujos algoritmos fossem adequados à topologia dos circuitos Cipredi (na medida do possível). Para o posicionamento das células Cipredi, foi utilizada a ferramenta POTRANCA, já descrita na seção 3.2.2. Tal escolha deveu-se

ao fato das células Cipredi apresentarem características bastante similares àquelas encontradas nas células da biblioteca TRANCA, tais como a realizacão das conexões preferencialmente na direção horizontal e a baixa transparência vertical. O roteamento, por sua vez, foi realizado com o sistema MARTE, por ser o único roteador do tipo *maze* disponível no GME até o presente momento. Neste caso, houve a necessidade de se realizarem algumas adaptações para viabilizar o uso da ferramenta, pois a arquitetura Cipredi não foi desenvolvida em consonância com a filosofia de roteamento do MARTE. Por exemplo, o Cipredi usa metal 1 na direção vertical e metal 2 na horizontal, o que corresponde justamente ao contrário do adotado pelo Marcela (e pela metodologia TRANCA).

A implementação dos leiautes Marcela utilizou o protótipo inicial da ferramenta MARLA, descrita na seção 5.2. O roteamento foi realizado com o roteador MARTE, e somente as conexões geradas automaticamente foram expandidas para leiaute. Todas as versões Marcela ocupam a menor área possível da matriz MAR1000.

Os circuitos implementados foram um *flip-flop* D com *set* e *reset* (FFDSR), um *flip-flop* JK com *set* e *reset* (FFJKSR), o circuito Modem [REI 89] e uma unidade aritmética de quatro bits, pertencente à família LS181 (Alu LS181). Os circuitos Modem e Alu LS181 seguem os fluxos de projeto descritos nos dois parágrafos anteriores. Para os circuitos FFDSR e FFJKSR, o fluxo não foi seguido para o caso Cipredi porque tais versões já se encontram disponíveis na biblioteca de primitivas.

Os dados referentes a transistores utilizados, densidade de transistores e taxa de ocupação consideram somente transistores efetivamente usados na implementação de lógica. Isto significa que no caso Cipredi, os transistores usados no isolamento elétrico (*gate isolation*) são considerados como sendo não utilizados. As percentagens de conexões realizadas, por sua vez, só tem sentido de serem avaliadas nos casos em que o MARTE foi utilizado. A seguir, são detalhadas as características dos leiautes dos *benchmarks*.

6.1 O Circuito FFDSR

O circuito FFDSR é realizado com *transmission gates*, conforme o esquema mostrado pela figura 4.8(alto).

Observando-se o leiaute *sea-of-gates* (figura 6.1, esq.), pode-se notar que a transparência vertical é baixa devido à necessidade de conectarem-se as portas dos transistores complementares. Além disso, são necessárias cinco trilhas para a implementação das conexões internas. Para que fosse viável a realização do roteamento sem a alocação de canais, a arquitetura Cipredi provê 14 trilhas horizontais por banda, levando à utilização de transistores **n** com 22.6 μm de largura de canal e transistores **p** com 32.8 μm de largura de canal. Estas dimensões determinam uma capacitância de entrada do par complementar como sendo da ordem de 94fF.

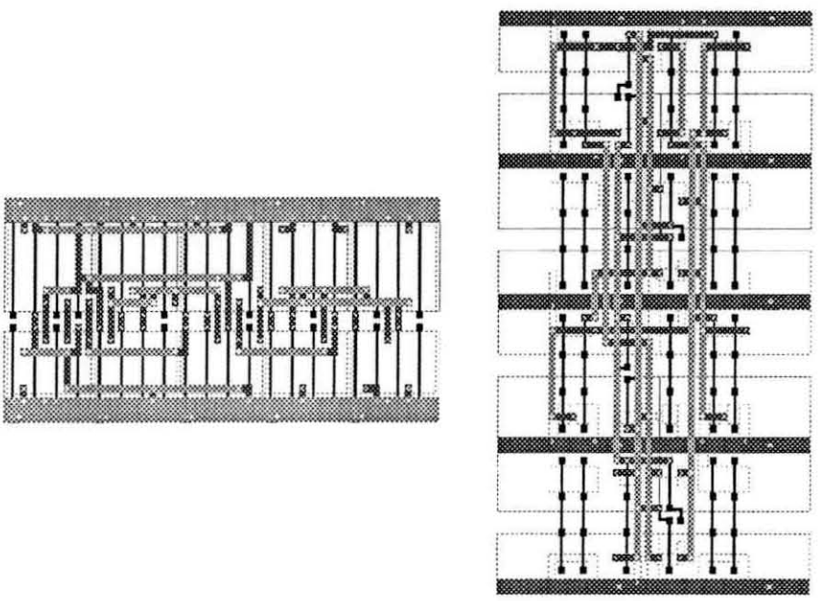

Figura 6.1: Leiaute de um *flip-flop* D com *set* e *reset*, versões Marcela (dir.) e Cipredi (esq.).

A figura 6.1 (à dir.) mostra um possível leiaute para a versão Marcela do mesmo *flip-flop*, com assinalamento e roteamento realizados manualmente sobre uma porção da matriz MAR1000. Apesar da versão Marcela ser 24% maior, pode-se observar uma distribuição mais equilibrada das conexões entre as direções horizontal

e vertical e um signficativo aumento da transparência na direção vertical, decorrentes da cuidadosa alocação das trilhas para roteamento e também do fato das portas dos pares complementares já se encontrarem ligadas através do poli. Além disso, a transparência horizontal da versão Marcela é um pouco inferior à da versão *sea-of-gates*, o que parece bastante natural, na medida em que nesta direção corre metal 1, o qual é obrigatoriamente utilizado para a conexão dos elementos pré-difundidos.

Também deve-se levar em conta que a distribuição de sinais de relógio, *set* e *reset* apresenta maior complexidade em função das células estarem aglutinadas em ambas direções **x** e **y**. Para um circuito onde o relógio e demais sinais de inicialização são globais, é possível uma racionalização no uso da transparência horizontal.

A tabela 6.1 resume as características dos leiautes anteriores. A capacitância de entrada refere-se ao pior valor, por pares complementares.

versão	**Cipredi**	**Marcela**
área [μm^2]	16934	21025
transistores utilizados	28	28
densidade [transistores/mm^2]	1653	1333
taxa de ocupação	0,7	0,58
transparência horiz.	9/14	22/40
transparência vert.	2/40	9/20

Tabela 6.1: Dados sobre as versões Marcela e Cipredi para o FFDSR.

6.2 O Circuito FFJKSR

O esquemático básico do cicuito FFJKSR é apresentado na figura 6.2. Trata-se de um FFDSR com uma lógica associada. Neste caso, o pequeno aumento na complexidade já permite que examinemos outro aspecto da abordagem Marcela: a exploração conveniente de transformações lógicas mediante a escolha de versões da biblioteca de equivalentes.

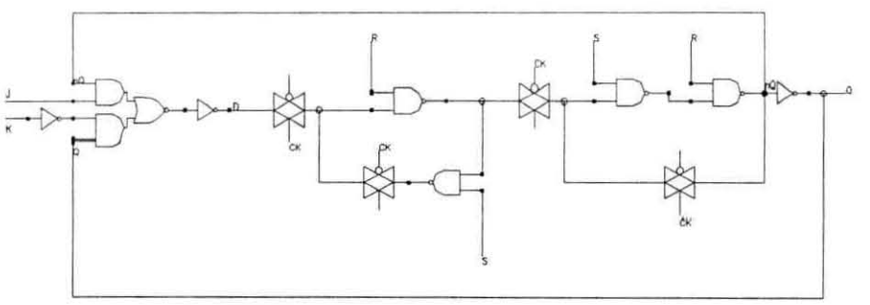

Figura 6.2: Esquemático básico para o FFJKSR.

Tanto a versão Marcela quanto a Cipredi buscaram otimizar o número de transistores mediante transformações, porém cada qual dentro de suas possibilidades. No caso da versão Cipredi, a lógica de entrada foi implementada com uma superporta do tipo A2OI. No caso da Marcela, duas possibilidades de decomposição são possíveis. A primeira é decompor a função A2OI em termos de *nors*, implementando o FFDSR com *nands*, e a segunda é decompor a A2OI em termos de *nands*, usando *nors* no FFDSR. Neste último caso, há a vantagem de eliminar-se um inversor.

A escolha depende do desempenho elétrico desejado[1] e da lógica de ativação do *set* e do *reset*. O uso de quantidades semelhantes de *nor* e *nand* visa alterar o mínimo possível o equilíbrio da distribuição de primitivas num circuito maior. Para esta comparação, adotou-se implementar o FFDSR com *nor* por resultar em menor número de transistores. A figura 6.3 mostra os leiautes do FFJKSR para Cipredi e Marcela.

O roteamento da versão Marcela foi realizado automaticamente. Nota-se claramente a *clusterização* horizontal do *sea-of-gates* cujas primitivas são implementadas numa única banda. A transparência vertical praticamente inexiste na versão Cipredi e tal circuito constitui-se numa barreira para sinais verticais, caso seja parte de um circuito maior. A tabela 6.2 mostra os dados levantados dos leiautes.

[1]Na matriz MAR1000, o atraso da célula *nor* é cerca de 20% maior do que o da célula *nand*.

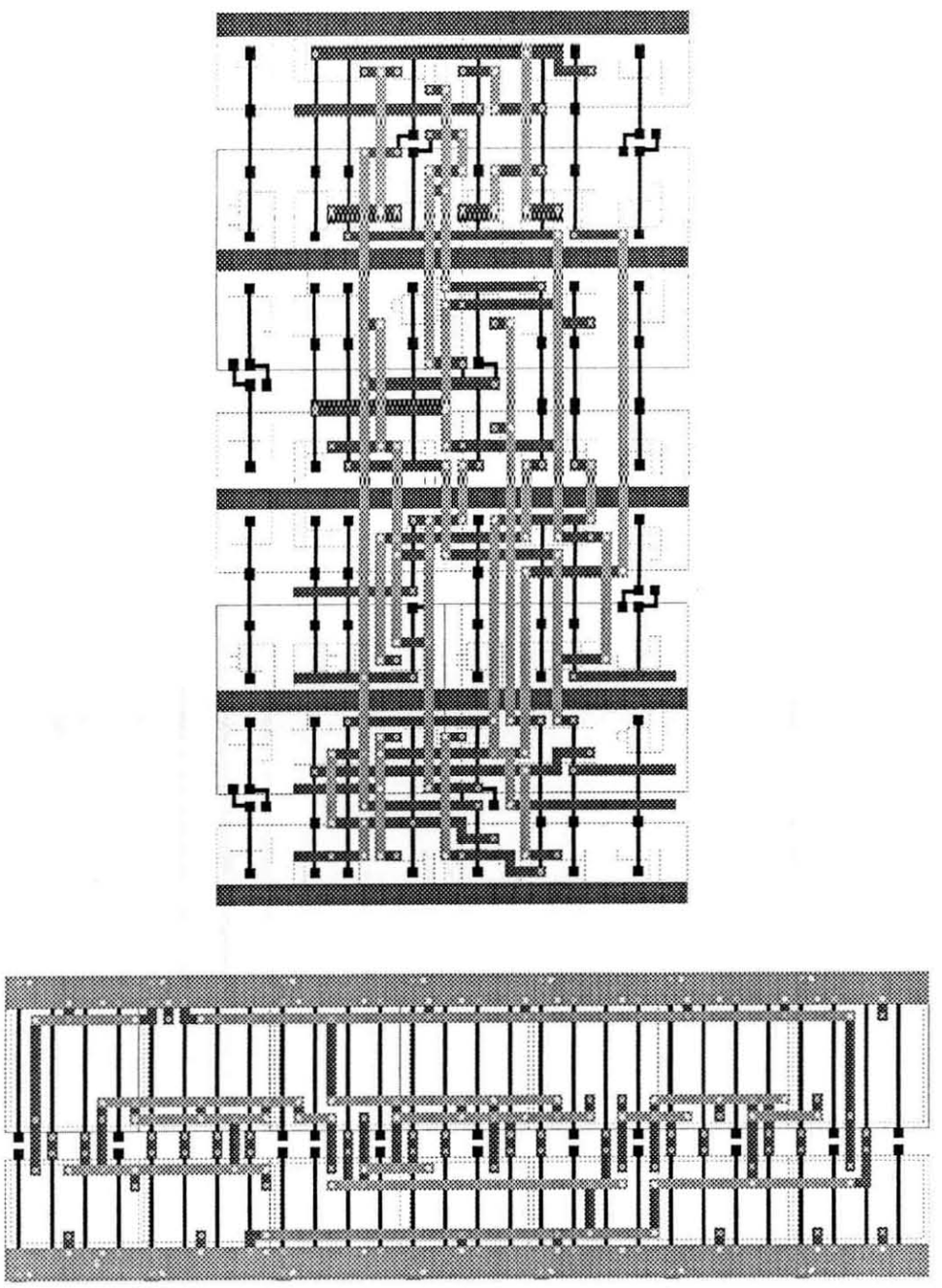

Figura 6.3: Leiaute do FFJKSR, versões Marcela (topo) e Cipredi (abaixo).

Apesar de necessitar mais transistores para implementar a mesma lógica, a versão Marcela resultou menor. Considerando-se a região da matriz MAR1000 sob a qual foram realizadas as conexões, a taxa de ocupação para Marcela é 0,71, o que é um bom resultado.

Quanto às transparências, a versão Cipredi é quase opaca na vertical, como já se esperava. Porém, percebe-se que a transparência horizontal da versão

versão	Cipredi	Marcela
área [μm^2]	25600	24200
transistores utilizados	34	40
densidade [transistores/mm^2]	1327	1650
taxa de ocupação	0,61	0,71
transparência horiz.	8/14	3/40
transparência vert.	3/56	9/23

Tabela 6.2: Dados sobre as versões Marcela e Cipredi para o FFJKSR.

Marcela está muito pequena. A razão disto é o fato do roteador MARTE não utilizar um esquema explícito de prioridades de alocação de trilhas. Como as únicas redes fornecidas ao MARTE pertencem ao próprio FFJKSR, não houve influência das redes externas e este teve maior grau de liberdade na escolha dos caminhos.

6.3 O Circuito Modem

O circuito Modem [REI 89] foi escolhido para a realização de *benchmark* pelo motivo já exposto na seção 5.4.

Ambas versões implementam a mesmas funções do ponto de vista dos pinos externos. Porém, como a matriz MAR1000 foi projetada para facilitar a implementação de *flip-flops* com *transmission gates*, a versão Marcela procurou valer-se de tal propriedade quando da escolha dos equivalentes lógicos.

A figura 6.4 mostra o leiaute Marcela (topo) e o leiaute *sea-of-gates* (abaixo), na mesma escala.

A tabela 6.3 sumariza as características das versões Marcela e Cipredi.

Conforme os dados levantados, o leiaute Marcela resultou 41% mais compacto que o Cipredi. Também a taxa de ocupação e a densidade de transistores da

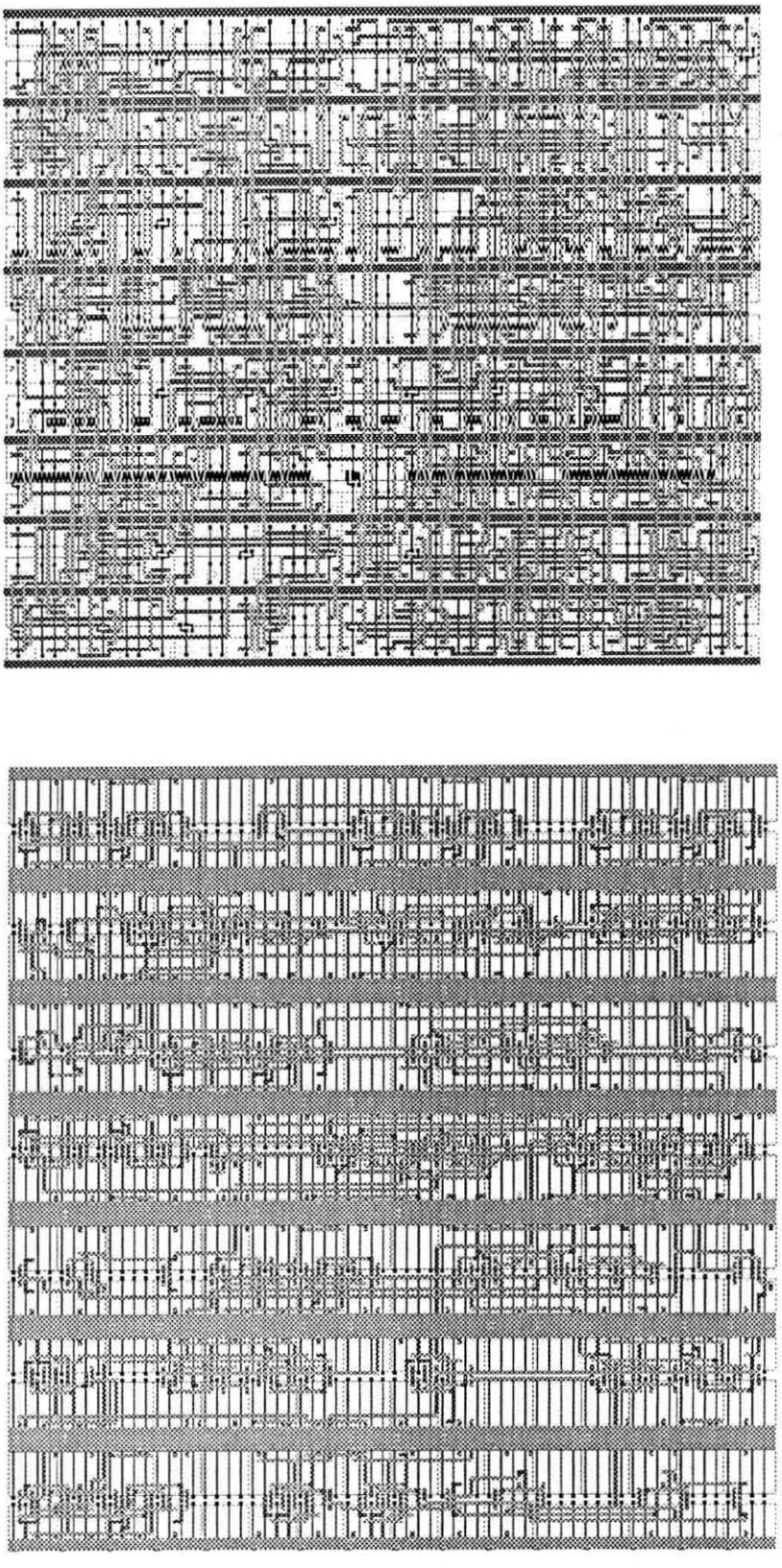

Figura 6.4: Leiautes das versões Marcela (topo) e Cipredi (abaixo) do circuito Modem.

versão	Cipredi	Marcela
área [mm^2]	0,388	0,275
transistores utilizados	506	442
densidade [transistores/mm^2]	1304	1607
taxa de ocupação	0,56	0,77
conexões realizadas	98,6%	95,8%

Tabela 6.3: Dados sobre as versões Marcela e Cipredi para o circuito Modem.

versão Marcela são significativamente superiores. O único dado negativo é a percentagem de conexões completas pelo MARTE: no caso Marcela, esta é ligeiramente inferior. Esta desvantagem é conseqüência do menor número de conexões a serem realizadas no caso Cipredi, uma vez que as primitivas utilizadas são de complexidade razoável (em sua maioria, *flip-flops*).

6.4 O Circuito Alu LS181

Este circuito é puramente combinacional, e seu esquema lógico original utiliza portas de até cinco entradas, além de oito portas *xor*.

A adaptação de tal esquema para a versão Cipredi foi praticamente direta, pois a maior parte das portas existem na biblioteca geométrica.

Porém, no caso Marcela, a decomposição lógica abriu uma ampla perspectiva de transformações. Este exemplo mostrou que a simples substituição das portas existentes pelos seus equivalentes mais simples é uma estratégia que nem sempre conduz a bons resultados. Em outras palavras, é necessário usar o método iterativo para determinar qual é o melhor conjunto de equivalentes a ser utilizado.

Numa primeira aproximação, foi adotado como equivalente da *xor* aquele que utiliza três *nands* e dois inversores, como mostrado na figura 4.6(esq.). Neste

caso, o balanço de primitivas foi o seguinte:

- 73 *nands*;
- 33 *nors*;
- 31 inversores;
- 0 *transmission gates*

Como se pode notar, o elemento limitante é a *nand*, cuja quantidade é maior do que o dobro do segundo elemento mais freqüente. Para tal distribuição, a matriz MAR1000 não é muito apropriada, pois h um grande desperdício de transistores. São necessárias 77 UBs, 11 em **x** e 7 em **y**, perfazendo uma área de $0{,}611\mu m^2$.

Substituindo-se o equivalente da *xor* pelo outro que utiliza *transmission gates*, a desproporção fica bastante amenizada. A nova distribuição passa a ser:

- 49 *nands*;
- 33 *nors*;
- 65 inversores;
- 16 *transmission gates*

O número mínimo de UBs foi reduzido para 50, sendo 10 em **x** e 5 em **y**. A área ocupada, por sua vez, passa para $0{,}396\mu m^2$, representando uma redução de 54% em relação ao equivalente anterior.

A figura 6.5 mostra os leiautes para a versão Marcela, segundo este último equivalente discutido, e para a versão Cipredi.

A tabela 6.4 mostra as principais características apresentadas pelos leiautes.

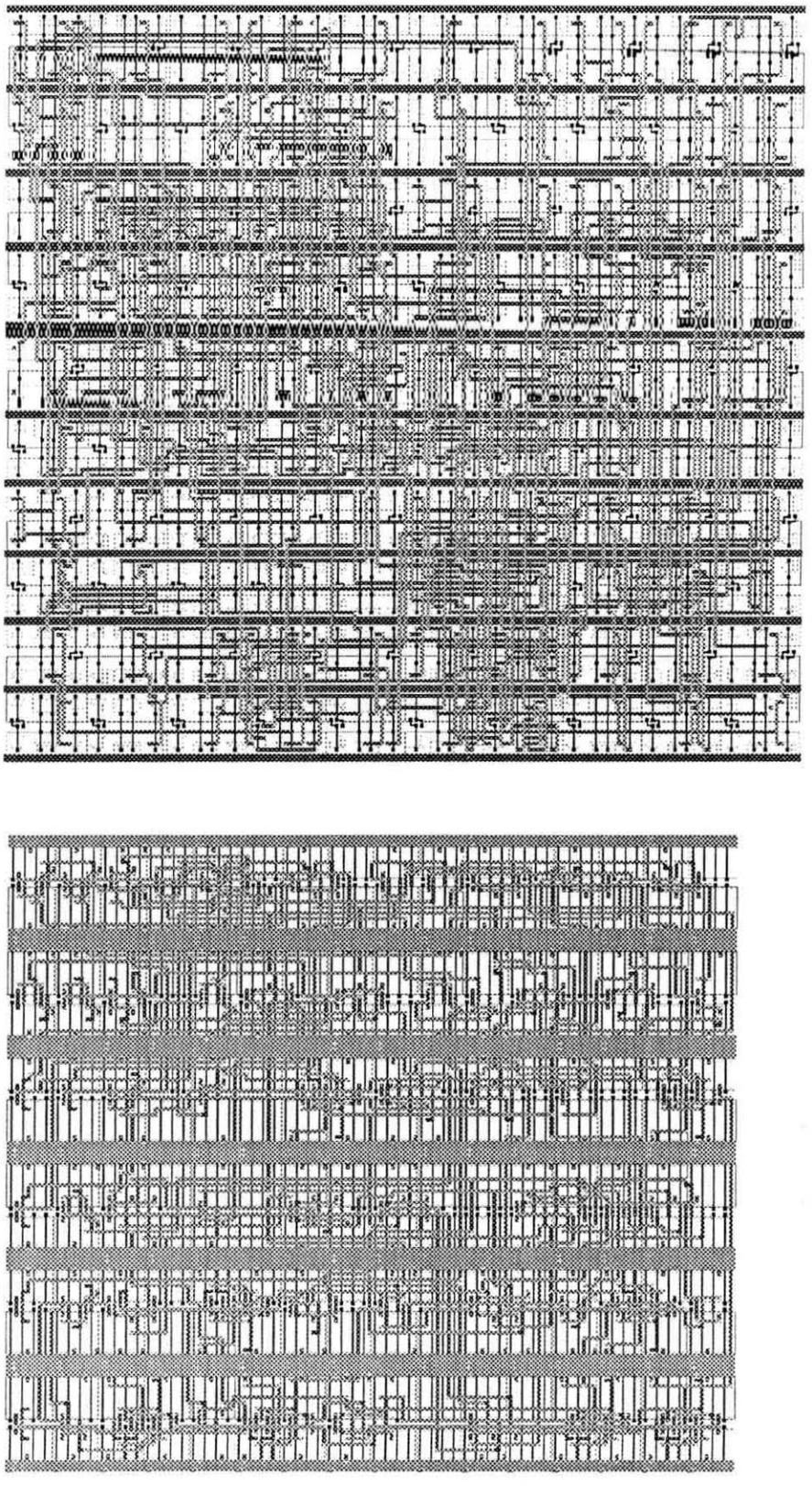

Figura 6.5: Leiautes das versões Marcela (topo) e Cipredi (abaixo) do circuito Alu LS181.

versão	Cipredi	Marcela
área [mm^2]	0,331	0,396
transistores utilizados	412	490
densidade [transistores/mm^2]	1244	1237
taxa de ocupação	0,54	0,61
conexões realizadas	86,6%	94%

Tabela 6.4: Dados sobre as versões Marcela e Cipredi para o circuito AluLS181.

A escolha cuidadosa do equivalente para *xor* representou considerável ganho em termos de área, porém isto não foi suficiente para gerar um leiaute mais compacto do que o da versão *sea-of-gates*. A versão Marcela resultou 19,6% maior do que a Cipredi. Porém, a taxa de ocupação obtida é um pouco superior, indicando que o problema reside principalmente na decomposição lógica, e não no leiaute propriamente dito.

Além disso, é interessante ressaltar que, embora a complexidade do roteamento na versão Cipredi não seja alta, o MARTE conseguiu completar somente 86,6% das conexões, enquanto que para a versão Marcela, este índice chegou a 94%. Isto é o indício de que a ocupação da matriz MAR1000 tende a ser mais racional do que a da matriz GME.

6.5 Considerações Sobre os *Benchmarks*

Pelos *benchmarks* desenvolvidos, nota-se que há uma tendência da abordagem Marcela fornecer leiautes mais compactos à medida em que aumenta o grau de complexidade dos circuitos. Parece que o efeito de distribuir melhor as conexões causa uma aparente perda para circuitos pequenos, quando considerados isoladamente. Mas num todo, o aumento da transparência é benéfico a ponto de permitir boa compactação.

O circuito Alu LS181 é um exemplo do impacto da escolha dos equivalentes convenientes a fim de melhorar a ocupação da matriz. Além disso, há de se considerar o efeito de manipulações da lógica, o qual foi explorado no FFJKSR e no Modem. Nestes dois casos, os resultados foram positivos. Porém, a intervenção humana foi intensa e um algoritmo ou heurística para desempenhar tal papel é de elevada complexidade. Por outro lado, qualquer que seja o sistema de geração de leiautes baseado em biblioteca de células, o problema da manipulação da lógica estará presente, pois bibliotecas com muitos elementos são de difícil manutenção.

No caso dos leiautes Marcela, notou-se a não ocupação de trilhas sobre as entradas das células, as quais são em polissilício. Apesar de se constituir numa ótima forma de cruzar bandas inteiras de UBs na vertical, a falta de pontos de acesso impediu que o roteador as utilizasse. Para que estas trilhas fossem aproveitadas, seria necessário que o algoritmo de roteamento realizasse a expansão tentando aplicar *doglegs*, o que ainda não está implementado. Desta forma, poder-se-ia desalinhar tais entradas, de modo a permitir uma exploração destas trilhas, independentemente da entropia gerada na organização da matriz. Com efeito, se o deslocamento for de apenas uma trilha vertical, o modelo topológico da UB não necessita ser alterado para utilização no assinalador.

Observou-se também que o aumento do número de pontos para contatos de dreno aumentaria a flexibilidade para conexões. Entretanto há um temor que o roteador explore tal liberdade de forma indisciplinada, bloqueando indevidamente as conexões mais longas. Bem ou mal, a existência de um único ponto para colocação do contato de dreno por transistor cria uma restrição equivalente ao esquema de prioridades do TRAMO. Deve-se ressaltar que o algoritmo *maze* não permite a implementação de um tal esquema.

7 CONCLUSÃO

Este trabalho buscou estudar, implementar e testar alguns procedimentos voltados ao desenvolvimento de um módulo de geração automática de circuitos integrados no estilo pré-difundido, unindo características arquiteturais inspiradas na metodologia TRANCA e uma estratégia de ocupação que obriga a decomposição do circuito a ser implementado em primitivas lógicas simples. Foi definida, avaliada e testada uma matriz de propósito geral, denominada MAR1000.

Da decomposição, resultaram circuitos com conexões mais bem distribuídas entre as direções horizontal e vertical, levando a boa transparência em ambas direções. Da arquitetura, resultou que o número de conexões internas fica proporcionalmente reduzido, uma vez que portas de transistores complementares já se encontram conectadas em polissilício. A estratégia de priorizar a realização das conexões conduziu a uma arquitetura bastante propícia à automatização do roteamento.

A aglutinação das células procedentes de um mesmo bloco funcional ou função lógica complexa é deixada ao encargo da ferramenta de assinalamento, a qual decidirá o posicionamento relativo conforme as posições das células vizinhas, o que também contribui para a melhor distribuição das conexões.

Com boas distribuições de roteamento, o problema da saturação de trilhas é atenuado e a mesma arquitetura pode ser expandida para capacidades maiores. Além disso, para as complexidades de circuitos até agora testadas, a definição de 10 trilhas de roteamento por banda, sugerida pela metodologia TRANCA, se mostrou suficiente.

Os resultados dos *benchmarks* com *sea-of-gates* demonstram que, à medida em que aumenta a complexidade dos circuitos, maior tende a ser o ganho da abordagem Marcela com relação aos *sea-of-gates*, em termos de área.

O caso do circuito Alu LS181 demonstra que é necessário dar-se atenção

à decomposição lógica nos casos em que há grande desproporção entre o elemento limitante e os demais elementos. Para estes casos, três soluções podem ser vislumbradas:

- Prover formas automáticas e eficientes de se realizarem transformações lógicas sobre o equivalente Marcela do circuito;
- Dotar o Adaptador de Lógica de um algoritmo inteligente para escolha de versões de equivalentes da biblioteca;
- Criar pelo menos mais uma matriz, onde a UB apresente proporção mais eqüalitária entre as quatro primitivas já usadas (por exemplo, 1:1:1:1).

A utilização de uma das soluções não inviabiliza o uso simultâneo de outra, muito antes pelo contrário. Porém, o custo de implementá-las deve ser analisado.

Quanto à máxima capacidade implementável pela arquitetura da matriz MAR1000, ainda é necessário estudar-se o comportamento dos roteamentos de circuitos complexos implementados em matrizes com capacidade em torno de 10000 a 20000 *gates equivalentes*.

O desenvolvimento de ferramentas de PAC vem auxiliar não somente o projeto de circuitos em si, mas a própria avaliação da abordagem. A avaliação anteriormente citada pode ser parcialmente realizada sem necessariamente implementar uma tal matriz. O teste da capacidade de roteamento poderia ser feito em ambiente totalmente simbólico, apenas definindo uma porção de matriz maior e um conjunto de redes a rotear bastante complexo. Da mesma forma, novas matrizes, constituídas de novas UBs, podem ser testadas, sem que se dispenda esforço na implementação de leiautes.

Da análise do desempenho do roteador sobre a MAR1000, resultaram algumas observações interessantes. Por exemplo, a falta de pontos para acessar as trilhas verticais existentes sobre o polissilício das portas dos transistores determina

sua não utilização pelo roteador. Embora estejam livres de ponta a ponta, o fato das entradas estarem alinhadas na vertical dificulta o acesso, principalmente porque a única maneira de acessá-las é por meio de *doglegs*. Como a versão inicial do MARTE não realiza *doglegs*, é impossível o acesso a tais trilhas no roteamento automático. Há duas soluções para tal problema:

- Pesquisar novas rotinas capazes de realizar *doglegs* durante a expansão;
- Desalinhar as entradas das células.

Outra sugestão de mudança da topologia da MAR1000 refere-se ao aumento de pontos para colocação de contatos de dreno para as CBs *nand*, *nor* e inversor, de modo a aumentar sua conectibilidade horizontal com a *transmission gate*.

Além das modificações arquiteturais já comentadas, as próximas etapas para o Projeto Marcela podem ser:

- Inclusão de uma biblioteca de primitivas MAR1000 para o SOLO2000, de modo que seja possível a utilização do editor de esquemáticos e simulador lógico SILOS;
- Integração de uma biblioteca de primitivas no ambiente TENTOS, permitindo o uso do ESQUELETO para a edição do esquemático;
- Desenvolvimento de um analisador de caminho crítico para inserção de reforçadores de sinal (*buffers*), os quais seriam implementados com inversores e/ou portas *nand* disponíveis na matriz;
- Implementação de um novo circuito de teste, aproveitando a experiência obtida no projeto e teste do circuito TCHÊ;
- Projeto, fabricação e testes de *pads* configuráveis.

ANEXO A-1 DESCRIÇÃO SPICE DO CIRCUITO MODEM

```
*************************************************************************
*                                                                       *
*                        PROJETO MARCELA                                *
*                                                                       *
*              DESCRICAO DO CIRCUITO MODEM / MARCELA                    *
*                                                                       *
*                                                                       *
*************************************************************************

* CIRCUITO: modemM.cir
* TECNOLOGIA: cmos12.tec  TIPO CMOS
*
* Modelos dos transistores
*
.MODEL NMOS NMOS LEVEL=2 LD=0.125U TOX=250E-10 NSUB=2E16 VTO=0.7
+UO=510 UEXP=0.22 UCRIT=24.3K DELTA=0.4 XJ=0.4U VMAX=54K NEFF=4
+RSH=55 NFS=0 JS=2U CJ=130U CJSW=620P MJ=0.53 MJSW=0.53 PB=0.68V
+CGDO=320P CGSO=320P
*
.MODEL PMOS PMOS LEVEL=2 LD=0.10U TOX=250E-10 NSUB=5E16 VTO=-1.1
+UO=210 UEXP=0.33 UCRIT=51K DELTA=0.4 XJ=0.5U VMAX=47K NEFF=0.88
+RSH=75 NFS=0 JS=10U CJ=490U CJSW=590P MJ=0.46 MJSW=0.46 PB=0.78V
+CGDO=320P CGSO=320P
*
*
VCC 1 0 5
*
*CIRCUITO PRINCIPAL
*COD   Td HTX NAMI  X1  X2  X3  X4  X5  X6  X7  X8  X9 X10 X11 X12  T+  T-
*     500 514 540  509 517 523 532 533 506 527 544 534 535 550 551 560 566
XT1  502 503 500 501 TG
XT2  503 502 501 504 TG
XT3  503 502 505 507 TG
XT4  502 503 507 509 TG
XT5  502 503 510 511 TG
XT6  503 502 511 513 TG
```

```
XT7  502 503 512 515 TG
XT8  502 503 515 517 TG
XT9  502 503 518 519 TG
XT10 503 502 521 519 TG
XT11 503 502 520 522 TG
XT12 502 503 522 523 TG
XT13 502 503 525 526 TG
XT14 503 502 529 526 TG
XT15 503 502 528 530 TG
XT16 502 503 530 532 TG
XT17 502 503 544 545 TG
XT18 503 502 547 545 TG
XT19 503 502 546 548 TG
XT20 502 503 548 535 TG
XT21 502 503 0   567 TG
XT22 503 502 567 569 TG
XT23 503 502 568 570 TG
XT24 502 503 570 572 TG
XT25 502 503 573 574 TG
XT26 503 502 576 574 TG
XT27 503 502 575 577 TG
XT28 502 503 577 579 TG
XT29 502 503 580 581 TG
XT30 503 502 583 581 TG
XT31 503 502 582 584 TG
XT32 502 503 584 586 TG
XT33 502 503 590 591 TG
XT34 503 502 593 591 TG
XT35 503 502 592 594 TG
XT36 502 503 594 551 TG
XI1  505 504 INV
XI2  507 508 INV
XI3  509 510 INV
XI4  511 512 INV
XI5  516 517 INV
XI6  517 518 INV
XI7  519 520 INV
XI8  520 521 INV
XI9  522 524 INV
XI10 524 523 INV
XI11 523 525 INV
XI12 528 529 INV
XI13 530 531 INV
XI14 532 533 INV
XI15 538 539 INV
```

```
XI16 541 527 INV
XI17 545 546 INV
XI18 546 547 INV
XI19 548 549 INV
XI20 549 535 INV
XI21 535 534 INV
XI23 557 554 INV
XI24 544 555 INV
XI25 563 558 INV
XI26 503 502 INV
XI27 568 569 INV
XI28 570 571 INV
XI29 572 573 INV
XI30 574 575 INV
XI31 575 576 INV
XI32 577 578 INV
XI33 578 579 INV
XI34 579 580 INV
XI35 581 582 INV
XI36 582 583 INV
XI37 584 585 INV
XI38 585 586 INV
XI39 586 587 INV
XI40 591 592 INV
XI41 592 593 INV
XI42 594 595 INV
XI43 595 551 INV
XI44 551 550 INV
XR1  501 506 505 NOR
XR2  506 508 509 NOR
XR3  512 514 513 NOR
XR4  514 515 516 NOR
XR5  526 527 528 NOR
XR6  527 531 532 NOR
XR7  536 537 538 NOR
XR8  539 540 506 NOR
XR9  559 565 560 NOR
XR10 565 564 566 NOR
XR11 506 567 568 NOR
XR12 506 571 572 NOR
XA1  509 517 536 NAND
XA2  523 532 537 NAND
XA3  535 506 541 NAND
XA4  532 534 542 NAND
XA5  533 535 543 NAND
```

```
XA6  542 543 544 NAND
XA7  534 551 552 NAND
XA8  550 535 556 NAND
XA9  552 556 557 NAND
XA10 550 555 561 NAND
XA11 544 551 562 NAND
XA12 561 562 563 NAND
XA13 557 558 559 NAND
XA14 554 563 564 NAND
XA15 587 551 588 NAND
XA16 586 550 589 NAND
XA17 588 589 590 NAND


*DEC    Td+ Td-  Z1  Z2  Z3  D   ~D
*       600 623 608 629 622 659 658
XT37 602 601 600 603 TG
XT38 601 602 605 603 TG
XT39 601 602 604 606 TG
XT40 602 601 606 608 TG
XT41 602 601 623 624 TG
XT42 601 602 626 624 TG
XT43 601 602 625 627 TG
XT44 602 601 629 627 TG
XT45 602 601 631 632 TG
XT46 601 602 634 632 TG
XT47 601 602 633 635 TG
XT48 602 601 636 365 TG
XT49 602 601 638 639 TG
XT50 601 602 641 639 TG
XT51 601 602 640 642 TG
XT52 602 601 642 644 TG
XT53 602 601 645 646 TG
XT54 601 602 648 646 TG
XT55 601 602 647 649 TG
XT56 602 601 649 651 TG
XT57 602 601 652 563 TG
XT58 601 602 655 653 TG
XT59 601 602 654 656 TG
XT60 602 601 656 658 TG
XI45 603 604 INV
XI46 604 605 INV
XI47 606 607 INV
XI48 607 608 INV
XI49 608 609 INV
```

```
XI50 624 625 INV
XI51 625 626 INV
XI52 627 628 INV
XI53 628 629 INV
XI54 629 630 INV
XI55 602 601 INV
XI56 621 622 INV
XI57 632 633 INV
XI58 637 636 INV
XI59 636 638 INV
XI60 639 640 INV
XI61 640 641 INV
XI62 642 643 INV
XI63 643 644 INV
XI64 644 645 INV
XI65 646 647 INV
XI66 647 648 INV
XI67 649 650 INV
XI68 650 651 INV
XI69 651 652 INV
XI70 653 654 INV
XI71 657 658 INV
XI72 658 659 INV
XR13 608 602 610 NOR
XR14 602 629 614 NOR
XR15 602 613 619 NOR
XR16 602 617 620 NOR
XR17 619 620 621 NOR
XR18 622 633 634 NOR
XR19 622 635 637 NOR
XR20 622 654 655 NOR
XR21 622 656 657 NOR
XA18 610 616 611 NAND
XA19 611 616 612 NAND
XA20 614 612 615 NAND
XA21 615 612 616 NAND
XA22 609 612 613 NAND
XA23 616 630 617 NAND
XA24 608 629 631 NAND


*INVERSOR      E   S
.SUBCKT INV   100 101 VCC
MN1   101 100   0   0 NMOS L=1.2U W=3.0U AD=31P AS=45P PD=27U PS=36U
MP2   101 100   1   1 PMOS L=1.2U W=7.6U AD=43P AS=79P PD=29U PS=44U
```

```
.ENDS INV


*TG         GP  GN  E/S S/E
.SUBCKT TG 202 201 200 203 VCC
MN3    200  201  203  0 NMOS L=1.2U W=4.5U AD=25P AS=39P PD=20U PS=29U
MP4    200  202  203  1 PMOS L=1.2U W=4.6U AD=25P AS=39P PD=20U PS=29U
.ENDS TG


*NOR2        E1  E2  S
.SUBCKT NOR 300 301 302 VCC
MN5    302  301   0   0 NMOS L=1.2U W=3.0U AD=40P AS=45P PD=33U PS=36U
MN6    302  300   0   0 NMOS L=1.2U W=3.0U AD=40P AS=97P PD=33U PS=68U
MP7    303  300   1   1 PMOS L=1.2U W=10.3U AD=88P AS=174P PD=37U PS=86U
MP8    303  301  302  1 PMOS L=1.2U W=10.3U AD=88P AS=58P PD=37U PS=31U
.ENDS NOR


*NAND2        E1  E2  S
.SUBCKT NAND 400 401 402 VCC
MN9    403  401  402  0 NMOS L=1.2U W=4.5U AD=37P AS=97P PD=25U PS=68U
MN10   403  400   0   0 NMOS L=1.2U W=4.5U AD=37P AS=97P PD=25U PS=68U
MP11   402  400   1   1 PMOS L=1.2U W=7.6U AD=63P AS=75P PD=37U PS=43U
MP12   402  401   1   1 PMOS L=1.2U W=7.6U AD=63P AS=73P PD=37U PS=42U
.ENDS NAND


*netweight: 5   *list: 587,586,550,551


*netweight: 10  *list: 589,588,590,591,592,593,594,595,581,582,583,584,585
```

ANEXO A-2 ARQUIVO DE CONFIGURAÇÃO PARA O ASSINALADOR

```
rlasp 1
tipos                          * topologia da UB
 tg  2 0 0 0 0 10 12 0 1
 nor 1 4 0 0 0
 nand 1 4 12 0 1
 inv 2 10 0 0 0 0 12 0 1
.endt
corte 1 1            * tamanho da menor particao nas direcoes x e y, em UBs

traslacao 0 0

nx 14                * tamanho do circuito (valores default)
ny 9

incx 14              * numero de trilhas por UB
incy 24

max_trocas 7         * numero maximo de trocas por iteracao

fundo ub1            * nome do arquivo de restricoes da UB usada
```

ANEXO A-3 ARQUIVO DE DESCRIÇÃO DO POSICIONAMENTO

SINTAXE:

```
instancia posx posy espx espy tipo rede [ rede ... ] ;
```

SEMANTICA:

```
instancia = nome da instancia da celula posicionada no circuito
posx e posy = coordenadas da posicao da celula
espx e espy = espelhamento da celula em relacao a X e Y ( 0 ou 1 )
tipo = nome da celula da biblioteca ( arquivos que a descrevem )
rede = label de um no' do circuito, que forma uma rede
```

EXEMPLO:

Circuito Modem, versao minima (9 x 4 UBs)

N.B.:

ub1 = restricoes da matriz;
inv,nand,nor,tg = restricoes das celulas.

```
ub0    0    0 0 0    ub1 ;
ub1    0   24 0 0    ub1 ;
ub2    0   48 0 0    ub1 ;
ub3    0   72 0 0    ub1 ;
ub4   14    0 0 0    ub1 ;
ub5   14   24 0 0    ub1 ;
ub6   14   48 0 0    ub1 ;
ub7   14   72 0 0    ub1 ;
ub8   28    0 0 0    ub1 ;
ub9   28   24 0 0    ub1 ;
ub10  28   48 0 0    ub1 ;
ub11  28   72 0 0    ub1 ;
ub12  42    0 0 0    ub1 ;
ub13  42   24 0 0    ub1 ;
ub14  42   48 0 0    ub1 ;
```

```
ub15  42  72 0 0   ub1 ;
ub16  56   0 0 0   ub1 ;
ub17  56  24 0 0   ub1 ;
ub18  56  48 0 0   ub1 ;
ub19  56  72 0 0   ub1 ;
ub20  70   0 0 0   ub1 ;
ub21  70  24 0 0   ub1 ;
ub22  70  48 0 0   ub1 ;
ub23  70  72 0 0   ub1 ;
ub24  84   0 0 0   ub1 ;
ub25  84  24 0 0   ub1 ;
ub26  84  48 0 0   ub1 ;
ub27  84  72 0 0   ub1 ;
ub28  98   0 0 0   ub1 ;
ub29  98  24 0 0   ub1 ;
ub30  98  48 0 0   ub1 ;
ub31  98  72 0 0   ub1 ;
ub32 112   0 0 0   ub1 ;
ub33 112  24 0 0   ub1 ;
ub34 112  48 0 0   ub1 ;
ub35 112  72 0 0   ub1 ;
XI69    10   0 0 0  inv  651  652 ;
XI56     0  12 0 1  inv  621  622 ;
XI58    10  24 0 0  inv  637  636 ;
XI44     0  36 0 1  inv  551  550 ;
XI48    10  48 0 0  inv  607  608 ;
XI47     0  60 0 1  inv  606  607 ;
XI52    10  72 0 0  inv  627  628 ;
XI53     0  84 0 1  inv  628  629 ;
XI68    24   0 0 0  inv  650  651 ;
XI67    14  12 0 1  inv  649  650 ;
XI64    24  24 0 0  inv  644  645 ;
XI46    14  36 0 1  inv  604  605 ;
XI49    24  48 0 0  inv  608  609 ;
XI54    14  60 0 1  inv  629  630 ;
XI72    24  72 0 0  inv  658  659 ;
XI59    14  84 0 1  inv  636  638 ;
XI55    38   0 0 0  inv  602  601 ;
XI45    28  12 0 1  inv  603  604 ;
XI1     38  24 0 0  inv  505  504 ;
XI26    28  36 0 1  inv  503  502 ;
XI50    38  48 0 0  inv  624  625 ;
XI71    28  60 0 1  inv  657  658 ;
XI70    38  72 0 0  inv  653  654 ;
XI51    28  84 0 1  inv  625  626 ;
```

```
XI65    52   0 0 0  inv  646  647 ;
XI66    42  12 0 1  inv  647  648 ;
XI57    52  24 0 0  inv  632  633 ;
XI62    52  48 0 0  inv  642  643 ;
XI63    42  60 0 1  inv  643  644 ;
XI61    52  72 0 0  inv  640  641 ;
XI60    42  84 0 1  inv  639  640 ;
XI16    66   0 0 0  inv  541  527 ;
XI29    56  12 0 1  inv  572  573 ;
XI32    66  24 0 0  inv  577  578 ;
XI21    56  36 0 1  inv  535  534 ;
XI37    66  48 0 0  inv  584  585 ;
XI6     56  60 0 1  inv  517  518 ;
XI5     66  72 0 0  inv  516  517 ;
XI14    56  84 0 1  inv  532  533 ;
XI27    80   0 0 0  inv  568  569 ;
XI13    70  12 0 1  inv  530  531 ;
XI28    80  24 0 0  inv  570  571 ;
XI39    70  36 0 1  inv  586  587 ;
XI3     80  48 0 0  inv  509  510 ;
XI24    70  60 0 1  inv  544  555 ;
XI19    80  72 0 0  inv  548  549 ;
XI20    70  84 0 1  inv  549  535 ;
XI12    94   0 0 0  inv  528  529 ;
XI42    84  12 0 1  inv  594  595 ;
XI43    94  24 0 0  inv  595  551 ;
XI11    84  36 0 1  inv  523  525 ;
XI4     94  48 0 0  inv  511  512 ;
XI35    84  60 0 1  inv  581  582 ;
XI36    94  72 0 0  inv  582  583 ;
XI25    84  84 0 1  inv  563  558 ;
XI30   108   0 0 0  inv  574  575 ;
XI31    98  12 0 1  inv  575  576 ;
XI9    108  24 0 0  inv  522  524 ;
XI10    98  36 0 1  inv  524  523 ;
XI18   108  48 0 0  inv  546  547 ;
XI23    98  60 0 1  inv  557  554 ;
XI34   108  72 0 0  inv  579  580 ;
XI33    98  84 0 1  inv  578  579 ;
XI2    122   0 0 0  inv  507  508 ;
XI15   112  12 0 1  inv  538  539 ;
XI41   122  24 0 0  inv  592  593 ;
XI40   112  36 0 1  inv  591  592 ;
XI38   122  48 0 0  inv  585  586 ;
XI17   112  60 0 1  inv  545  546 ;
```

```
XI7    122  72 0 0  inv  519  520 ;
XI8    112  84 0 1  inv  520  521 ;
XA20     4  36 0 1 nand  614  612  615 ;
XA23     4  60 0 1 nand  616  630  617 ;
XA19     4  84 0 1 nand  611  616  612 ;
XA21    18  36 0 1 nand  615  612  616 ;
XA22    18  60 0 1 nand  609  612  613 ;
XA18    18  84 0 1 nand  610  616  611 ;
XA24    46  36 0 1 nand  608  629  631 ;
XA3     60  12 0 1 nand  535  506  541 ;
XA7     60  36 0 1 nand  534  551  552 ;
XA10    60  60 0 1 nand  550  555  561 ;
XA5     60  84 0 1 nand  533  535  543 ;
XA4     74  12 0 1 nand  532  534  542 ;
XA15    74  36 0 1 nand  587  551  588 ;
XA6     74  60 0 1 nand  542  543  544 ;
XA8     74  84 0 1 nand  550  535  556 ;
XA9     88  60 0 1 nand  552  556  557 ;
XA13    88  84 0 1 nand  557  558  559 ;
XA17   102  12 0 1 nand  588  589  590 ;
XA2    102  36 0 1 nand  523  532  537 ;
XA14   102  60 0 1 nand  554  563  564 ;
XA11   102  84 0 1 nand  544  551  562 ;
XA1    116  36 0 1 nand  509  517  536 ;
XA16   116  60 0 1 nand  586  550  589 ;
XA12   116  84 0 1 nand  561  562  563 ;
XR19     4   0 0 0  nor  622  635  637 ;
XR16     4  48 0 0  nor  602  617  620 ;
XR14     4  72 0 0  nor  602  629  614 ;
XR17    18  48 0 0  nor  619  620  621 ;
XR15    18  72 0 0  nor  602  613  619 ;
XR21    32  48 0 0  nor  622  656  657 ;
XR13    32  72 0 0  nor  608  602  610 ;
XR18    46  24 0 0  nor  622  633  634 ;
XR20    46  48 0 0  nor  622  654  655 ;
XR11    60   0 0 0  nor  506  567  568 ;
XR4     60  72 0 0  nor  514  515  516 ;
XR6     74   0 0 0  nor  527  531  532 ;
XR12    74  24 0 0  nor  506  571  572 ;
XR3     74  72 0 0  nor  512  514  513 ;
XR5     88   0 0 0  nor  526  527  528 ;
XR9     88  72 0 0  nor  559  565  560 ;
XR8    102   0 0 0  nor  539  540  506 ;
XR7    102  24 0 0  nor  536  537  538 ;
XR10   102  48 0 0  nor  565  564  566 ;
```

```
XR1    116   0 0 0  nor  501  506  505 ;
XR2    116  24 0 0  nor  506  508  509 ;
XT39     0   0 0 0   tg  601  602  604  606 ;
XT47    10  12 0 1   tg  601  602  633  635 ;
XT37     0  24 0 0   tg  602  601  600  603 ;
XT38    10  36 0 1   tg  601  602  605  603 ;
XT57     0  48 0 0   tg  602  601  652  563 ;
XT40    10  60 0 1   tg  602  601  606  608 ;
XT43     0  72 0 0   tg  601  602  625  627 ;
XT44    10  84 0 1   tg  602  601  629  627 ;
XT48    14   0 0 0   tg  602  601  636  365 ;
XT56    24  12 0 1   tg  602  601  649  651 ;
XT53    14  24 0 0   tg  602  601  645  646 ;
XT49    14  72 0 0   tg  602  601  638  639 ;
XT1     28  24 0 0   tg  502  503  500  501 ;
XT2     38  36 0 1   tg  503  502  501  504 ;
XT60    28  48 0 0   tg  602  601  656  658 ;
XT59    38  60 0 1   tg  601  602  654  656 ;
XT42    28  72 0 0   tg  601  602  626  624 ;
XT41    38  84 0 1   tg  602  601  623  624 ;
XT55    42   0 0 0   tg  601  602  647  649 ;
XT54    52  12 0 1   tg  601  602  648  646 ;
XT46    42  24 0 0   tg  601  602  634  632 ;
XT45    52  36 0 1   tg  602  601  631  632 ;
XT51    42  48 0 0   tg  601  602  640  642 ;
XT52    52  60 0 1   tg  602  601  642  644 ;
XT58    42  72 0 0   tg  601  602  655  653 ;
XT50    52  84 0 1   tg  601  602  641  639 ;
XT21    56   0 0 0   tg  502  503    0  567 ;
XT25    66  12 0 1   tg  502  503  573  574 ;
XT15    56  72 0 0   tg  503  502  528  530 ;
XT16    66  84 0 1   tg  502  503  530  532 ;
XT23    70   0 0 0   tg  503  502  568  570 ;
XT22    80  12 0 1   tg  503  502  567  569 ;
XT24    70  24 0 0   tg  502  503  570  572 ;
XT5     70  48 0 0   tg  502  503  510  511 ;
XT6     70  72 0 0   tg  503  502  511  513 ;
XT7     80  84 0 1   tg  502  503  512  515 ;
XT4     84   0 0 0   tg  502  503  507  509 ;
XT14    94  12 0 1   tg  503  502  529  526 ;
XT13    84  24 0 0   tg  502  503  525  526 ;
XT36    94  36 0 1   tg  502  503  594  551 ;
XT32    84  48 0 0   tg  502  503  584  586 ;
XT31    94  60 0 1   tg  503  502  582  584 ;
XT11    84  72 0 0   tg  503  502  520  522 ;
```

```
XT30    94  84 0 1   tg  503  502  583  581 ;
XT26    98   0 0 0   tg  503  502  576  574 ;
XT33   108  12 0 1   tg  502  503  590  591 ;
XT12    98  24 0 0   tg  502  503  522  523 ;
XT8    108  36 0 1   tg  502  503  515  517 ;
XT19    98  48 0 0   tg  503  502  546  548 ;
XT20   108  60 0 1   tg  502  503  548  535 ;
XT28    98  72 0 0   tg  502  503  577  579 ;
XT29   108  84 0 1   tg  502  503  580  581 ;
XT27   112   0 0 0   tg  503  502  575  577 ;
XT3    122  12 0 1   tg  503  502  505  507 ;
XT35   112  24 0 0   tg  503  502  592  594 ;
XT34   122  36 0 1   tg  503  502  593  591 ;
XT17   112  48 0 0   tg  502  503  544  545 ;
XT18   122  60 0 1   tg  503  502  547  545 ;
XT9    112  72 0 0   tg  502  503  518  519 ;
XT10   122  84 0 1   tg  503  502  521  519 ;
```

ANEXO A-4 DESCRIÇÃO DA TOPOLOGIA DA GRADE DE ROTEAMENTO PARA MATRIZ MAR1000

SINTAXE:

```
variavel = inteiro ;
...
parametro = real ;
...
bands;
d = espacamento ( tipo prioridade , tipo prioridade ... ) trilhas ;
...
sections;
d = espacamento ( tipo prioridade , tipo prioridade ... ) trilhas ;
...
```

SEMANTICA:

```
variavel = nb - numero de bandas de celulas;
           ns - numero de secoes;
           nl - numero de linhas;
           nc - numero de colunas.
parametro = x1 a x5 - parametros para as linhas de metal 1;
            y1 a y5 - parametros para as linhas de metal 2;
            z1 a z4 - parametros para os elementos de contato.
espacamento = numero real para espacamento entre bandas (ou secoes).
tipo = tipo da trilha, definido no modulo de grade ( v,g,a,i,s ).
prioridade = numero inteiro, prioridade da trilha para roteamento.
trilhas = inteiro para conferir numero de trilhas da banda ou secao.
```

EXEMPLO:

Circuito Modem, versao minima (9 x 4 UBs)

```
nb = 8;
ns = 1;
nl = 96;
nc = 126;

x1 = 3.0;
```

```
x2 = 3.0;
x3 = 3.0;
x4 = 1.8;
x5 = 3.0;

y1 = 3.0;
y2 = 3.0;
y3 = 3.0;
y4 = 1.8;
y5 = 3.0;

z1 = 1.5;
z2 = 0.75;
z3 = 1.5;
z4 = 0.75;

bands;

d = 0 (g0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,v0) 12; * uma banda
d = 0 (v0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,g0) 12;
d = 0 (g0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,v0) 12;
d = 0 (v0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,g0) 12;
d = 0 (g0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,v0) 12;
d = 0 (v0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,g0) 12;
d = 0 (g0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,v0) 12;
d = 0 (v0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,g0) 12;


sections;

d =  0 (
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,  * uma coluna de CBs
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,
j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0,j0
) 126;
```

BIBLIOGRAFIA

[AND 88] ANDERSON, Floyd E.; FORD, Jenny M. A 150k channelless gate array design in 0.5μm CMOS technology. **IEEE Journal of Solid-State Circuits**, New York, v. 23, n. 2, p. 520-522, Apr. 1988.

[APR 92] APREA, Javier F. et al. GAMA: uma interface de aquisição para cintilografias. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 7, 29 set.-02 out. 1992, Rio de Janeiro, RJ. **Anais**...Rio de Janeiro: SBC/SBMICRO/UFRJ, 1992. 284 p. p. 148-163.

[BEU 88a] BEUNDER, M.A.; HOEFFLINGER, B.; KERNHOF, J.P. New directions in semicustom arrays. **IEEE Journal of Solid-State Circuits**, New York, v. 23, n. 3, p. 728-735, June 1988.

[BEU 88b] BEUNDER, M.A.; KERNHOF, J.P.; HOEFFLINGER, B. The cmos gate forest: an efficient and flexible high-performance ASIC design environment. **IEEE Journal of Solid-State Circuits**, New York, v. 23, n. 2, p. 387-399, Apr. 1988.

[BRE 77] BREUER, M. A. A class of min-cut placement algorithms. In: DESIGN AUTOMATION CONFERENCE, 14, 20-22 June 1977, New Orleans. **Proceedings**... New York: ACM/IEEE, 1977. 507 p. p. 284-290.

[CAL 88a] CALAZANS, Ney L.V. **CIPREDI: contribuição inicial para um método de concepção de circuitos integrados pré-difundidos.** Porto Alegre: PGCC da UFRGS, 1988. 233 p. Dissertação de Mestrado.

[CAL 88b] CALAZANS, Ney L.V. et al. **Biblioteca de células para circuitos pré-difundidos.** Porto Alegre: PGCC da UFRGS, 1988. 81 p. (Relatório de Pesquisa, 101)

[CAR 92] CARRO, Luigi. Perspectivas para a Microeletrônica: a nova geração. In: SEMINÁRIO INTERNO DE MICROELETRÔNICA, 7, 21-22 nov. 1991, Capão da Canoa, RS. **Anais**... Porto Alegre: CPGCC da UFRGS, 1992. 156 p. p. 84-89.

[COR 79] CORRIGAN, L. I. A placement capacity based on partitioning. In: DESIGN AUTOMATION CONFERENCE, 16, 25-27 June 1979, San Diego. **Proceedings**... New York: ACM/IEEE, 1979. 567 p. p. 406-413.

[CRU 91] CRUSIUS, César et al. **Biblioteca de células TRANCA: regras ECPD15/1.** Porto Alegre: CPGCC da UFRGS, 1991. 71 p. (Relatório de Pesquisa, 146)

[DUC 91] DUCHENE, P.; DECLERCQ, M.; KANG, S.M. An integrated layout system for sea-of-gates module generation. In: EUROPEAN DESIGN AUTOMATION CONFERENCE, 25-28 Feb. 1991, Amsterdam, NL. **Proceedings**... Los Alamitos: IEEE, 1991. 601 p. p. 237-241.

[FID 82] FIDUCCIA, C. M.; MATHEYSES, R. M. A linear-time heuristic for improving network partitions. In: DESIGN AUTOMATION CONFERENCE, 19, 14-16 June 1982, Las Vegas. **Proceedings**... New York: IEEE, 1982. 919 p. p. 175-181.

[FRE 89] FREIRE, Luis Otavio de S. **Contribuição à concepção de circuitos integrados pré-difundidos utilizando o método CIPREDI.** Porto Alegre: PGCC da UFRGS, 1989. 155 p. Dissertação de Mestrado.

[FRE 91] FREIRE, Luis Otavio de S.; D' ALMEIDA, M. Column macro-cell structure for gate arrays. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 6, 23-25 out. 1991, Jaguariúna, SP. **Anais**... Campinas: SBMICRO/SBC/CTI, 1991. 221 p. p. 209-210.

[GON 83] GONÇALVES, Nelson F.; DE MAN, H. J. NORA: A racefree dynamic CMOS technique for pipelined logic structures. **IEEE Journal of Solid-State Circuits**, New York, v. 18, n. 13, p. 261-266, June 1983.

[GUN 91a] GÜNTZEL, José L. **Desenvolvimento de uma matriz de células pré-difundidas.** Porto Alegre: CPGCC da UFRGS, 1991. 105p. (Trabalho Individual, 226)

[GUN 91b] GÜNTZEL, José L.; RIBAS, Renato; FACHIN, Daniel; REIS, Ricardo A. L. TCHÊ: o circuito teste do projeto Marcela. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 6, 23-25 out. 1991, Jaguariúna, SP. **Anais**... Campinas: SBMICRO/SBC/CTI, 1991. 221 p. p. 21-30.

[GUN 91c] GÜNTZEL, José L.; RIBAS, Renato; REIS, Ricardo A. L. Marcela: uma nova abordagem para pré-difundidos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 6, 15-19 jul. 1991, Belo Horizonte, MG. **Anais**... Belo Horizonte: SBMICRO/UFMG, 1991. 648 p. p.534-543.

[GUN 92] GÜNTZEL, José L.; HACKBART, Anelise; KRÜGER, Fernando; REIS, Ricardo A. L. Comparação entre as abordagens Marcela e *sea-of-gates*: um estudo de caso. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 7, 29 set.-02 out. 1992, Rio de Janeiro, RJ. **Anais**...Rio de Janeiro: SBC/SBMICRO/UFRJ, 1992. 284 p. p. 164-178.

[HAN 76] HANNAN, M. P. K. et al. Some experimental results on placement techniques. In: DESIGN AUTOMATION CONFERENCE, 13, 28-30 June 1976, San Francisco. **Proceedings**... New York: IEEE, 1976. 501 p. p. 214-224.

[HAY 80] HAYES, Jim. MOS Scaling. **IEEE Computer**, Los Alamitos, v. 13, n. 1, p. 8-13, Jan 1980.

[IOS 83] IOSUPOVICI, A. et al. A module interchange placement technique. In: DESIGN AUTOMATION CONFERENCE, 20, 27-28 June 1983, Miami Beach. **Proceedings...** New York: ACM/IEEE, 1983. 815 p. p. 171-174.

[JOH 90] JOHNSON, B.; QUARLES, T. **SPICE3 version 3D2: user's manual.** Berkeley, CA: University of California, 1990. 120 p.

[JOH 92a] JOHANN, Marcelo de Oliveira. **MARTE: ambiente de roteamento simbólico.** Porto Alegre: Instituto de Informática da UFRGS, 1992. 106p. Projeto de Diplomação.

[JOH 92b] JOHANN, Marcelo de Oliveira. **Desenvolvimento do sistema de roteamento simbólico MARTE.** Porto Alegre, Instituto de Informática da UFRGS, 1992. 74p. Projeto de Diplomação.

[KUR 87] KURAMITSU, Y et al. A 540k-transistor cmos variable-track masterslice. **IEEE Journal of Solid-State Circuits**, New York, v. 22, n. 2, p. 198-201, Apr. 1987.

[LUB 90] LUBASZEWSKI, Marcelo S. **Geração automática de lógica aleatória utilizando a metodologia TRANCA**. Porto Alegre: CPGCC da UFRGS, 1990. 232p. Dissertação de Mestrado.

[MEY 89] MEYER, Ernest. Structured arrays reenter semicustom arena. **Computer Design**, Tulsa, Oklahoma, v. 28, n. 1, p. 25-26, Jan. 1989.

[MIY 86] MIYAHARA, N et al. A composite cmos array with 4k RAM and 128k ROM. **IEEE Journal of Solid-State Circuits**, New York, v. 21, n. 2, p. 228-233, Apr. 1986.

[MOR 90a] MORAES, Fernando G. **TRAGO: síntese automática de leiaute para circuitos em lógica aleatória**. Porto Alegre: CPGCC da UFRGS, 1990. 199p. Dissertação de Mestrado.

[MOR 90b] MORAES, Fernando G. et al. **Edição revisada da biblioteca de células TRANCA.** Porto Alegre: CPGCC da UFRGS, 1990. 61p. (Relatório de Pesquisa, 125)

[MOR 90c] MORAES, F. G.; PEREIRA, C. E.; MARCHIORO, G. F. ESQUELETO: editor de esquemas elétricos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 5, 11-13 jul. 1990, Campinas, SP. **Anais**... Campinas: SBMICRO, 1990. 374 p. p. 3-11.

[MOR 91] MORAES, Fernando G.; REIS, Ricardo A. L. **TENTOS: gerenciador de software para Microeletrônica.** Porto Alegre: CPGCC da UFRGS, 1991, 36p. (Relatório de Pesquisa, 155)

[NOI 85] NOIJE, W. A. M. van; DECLERCK, G.J. Advanced cmos gate array architecture combining "gate isolation" and programmable routing channels. **IEEE Journal of Solid-State Circuits**,New York, v. 20, n. 2, p. 469-479, Apr. 1985.

[OKH 82] OKHURA, I et al. Gate Isolation: a novel basic cell configuration for cmos gate arrays. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 17-19 May 1982, Rochester. **Proceedings**... New York: IEEE, 1982. p. 307-310.

[OKU 89] OKUNO, Y. et al. 0.8μm 1.4M-transistor cmos SOG based on column macro-cell. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 15-18 May 1989, San Diego, CA. **Proceedings**... New York: IEEE, 1989. p. 821-824

[PAI 91] PAIXÃO, Gustavo et al. Uma nova célula de base gate array em 1.2μm. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 6, 23-25 out. 1991, Jaguariúna, SP. **Anais**... Campinas: SBMICRO/SBC/CTI, 1991. 221 p. p. 11-20.

[PRE 88] PREAS, Bryan T.; KARGER, Patrick G. Placement, assignment and floorplanning. In: PREAS, Bryan T. **Physical design automation of VLSI systems.** Menlo Park: Benjamin/Cummings, 1988. 510 p. p. 87-155.

[REI 83] REIS, Ricardo A. L. **TESS: evaluateur topologique predictif pour la génération automatique des plans de masse de circuits VLSI.** Grenoble: Institut Polytechnique de Grenoble, 1983. 353 p.

[REI 85] REIS, Ricardo A. L. Estratégias básicas para a concepção automática do layout de circuitos em lógica aleatória. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 2, 20-27 jul. 1985, Porto Alegre, RS. **Anais**... Porto Alegre: SBC/UFRGS, 1985. p. 223-36.

[REI 87] REIS, Ricardo A. L. A new standard cell CAD methodology. In: CUSTOM INTEGRATED CIRCUITS CONFERENCE, 04-07 May 1987, Portland. **Proceedings**... New York: IEEE, 1987. 732 p. p. 385-388.

[REI 88] REIS, Ricardo A. L.; GOMES, Rogério; LUBASZEWSKI, Marcelo S. An efficient design methodology for standard cell circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 07-09 June 1988, Helsinki. **Proceedings**... Piscataway: IEEE, 1988. v. 2, p. 1213-1216.

[REI 89] REIS, André I. et al. MODEM: desenvolvimento de um ASIC para modens de banda base. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 4, 12-14 jul. 1989, Porto Alegre, RS. **Anais**... Porto Alegre: SBMICRO/UFRGS, 1989. 2 v. v. 2 p. 617-626.

[REI 90] REIS, André I. et al. **Biblioteca de células TRANCA, regras CMP: Parte II.** Porto Alegre: CPGCC da UFRGS, 1990. 39 p. (Relatório de Pesquisa, 129)

[REI 91] REIS, André Inácio; REIS, Ricardo A. L. TRAMO II: proposta para um gerador de leiaute baseado em células para circuitos cmos digitais com dois níveis de metal. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 6, 23-25 out. 1991, Jaguariúna, SP. **Anais**... Campinas: SBMICRO/SBC/CTI, 1991. 221 p. p. 90-99.

[REI 92a] REIS, André I.; GÜNTZEL, José L.; RIBAS, Renato R. Algumas formas de implementação de ASICs. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 7, 29 set.-02 out. 1992, Rio de Janeiro, RJ. **Anais**...Rio de Janeiro: SBC/SBMICRO/UFRJ, 1992. 284 p. p. 15-34.

[REI 92b] REIS, André I.; REIS, Ricardo A. L. Algumas considerações sobre altura de bandas em uma abordagem do tipo *channelless routing*. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, 7, 29 set.-02 out. 1992, Rio de Janeiro, RJ. **Anais**...Rio de Janeiro: SBC/SBMICRO/UFRJ, 1992. 284 p. p. 35-48.

[ROB 88] ROBERT, Michel et al. PRINT methodology: a compilation approach for cell library generation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 07-09 June 1988, Helsinki. **Proceedings**... Piscataway: IEEE, 1988. v. 1, p. 965-968.

[SAI 85] SAIGO, Takashi et al. A triple-level wired 24k-gate CMOS gate array. **IEEE Journal of Solid-State Circuits**, New York, v. 20, n. 5, p. 1005-1011, Oct. 1985.

[SIM 92] SIMÕES, S.A. et al. Matriz gate array cmos avançada, configurável por um único nível de metal. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 7, 08-10 jul. 1992. São Paulo, SP. **Anais**... São Paulo: SBMICRO/USP, 1992. 698 p. p. 281-291.

[SOT 91] SOTILLE, Mauro A. et al. **Manual do sistema TRAMO: Projeto TRANCA - Versão 1.0.** Porto Alegre: CPGCC da UFRGS, 1991. 142 p. (Relatório de Pesquisa, 144)

[SOU 91] SOUZA, Carlos E. S. **Uma interface gráfica para o projeto TRANCA em ambiente UNIX.** Porto Alegre: Instituto de Informática da UFRGS, 1991. 83 p. Projeto de Diplomação.

[TAK 85] TAKAHASHI, H. et al. A 240k transistor cmos array with flexible allocation of memory and channels. **IEEE Journal of Solid-State Circuits**, New York, v. 20, n. 5, p 1012-1017, Oct. 1985.

[TOD 86] TODESCO, A. **Manual do sistema RS.** Porto Alegre, PGCC da UFRGS, 1986. 10 p.

[TYL 91] TYLE, Sherrie van Sales of ASICs and programmable chips stay strong. **Electronic Design**, Cleveland, v. 39, n. 4, p. 75, Feb. 28, 1991.

[VLA 81] VLADIMIRESCU, A. et al. **SPICE user's guide: version 2G.** Berkeley, CA: University of California, 1981. 55 p.

[WAG 87] WAGNER, Flávio R.; FREITAS, Carla M. D. S. **Nilo: uma linguagem para a descrição de hardware no nível de portas lógicas.** Porto Alegre: PGCC da UFRGS, 1987. 18 p. (Relatório de Pesquisa, 66)

[YOR 88] YORK, Trevor. Gate array architectures. **Microprocessors and Microsystems,** Survey, v. 12, n. 6, p. 323-330, July 1988.

Informática
UFRGS

Dissertação apresentada aos Srs.:

Prof. Dr. José Monteiro da Mata (DCC/UFMG)

Prof. Dr. Ricardo A. da L. Reis

Prof. Dr. Sergio Bampi

Prof. Tiaraju Vasconcellos Wagner

Vista e permitida a impressão.
Porto Alegre, 02 / 06 / 93 .

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

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