

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

Estrutura de Roteamento em Circuitos *VLSI*

por

MARCELO DE OLIVEIRA JOHANN

Exame de Qualificação
EQ-15 CPGCC-UFRGS

Prof. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, 14 de Outubro de 1997.

Sumário

Lista de Abreviaturas	5
Lista de Figuras	7
Lista de Tabelas.....	8
Resumo.....	9
Abstract.....	10
1. Introdução	11
PARTE 1 - Projeto Físico de Circuitos VLSI	13
2. Projeto de Sistemas Eletrônicos Digitais.....	14
2.1 O diagrama de Gajski.....	14
2.2 Por que síntese?.....	15
2.3 Ciclo de Projeto de Circuitos VLSI	16
2.4 Ferramentas para <i>Electronic Design Automation</i>	18
2.5 Escopo da Síntese Física	19
3. Metodologias de Projeto e Estilos de Leiaute	20
3.1 Implementação de Sistemas Digitais	20
3.2 Formas de Implementação de ASICs.....	20
3.3 Metodologias de Projeto VLSI.....	21
3.4 Problemas de Síntese Física	23
3.5 Limitações das metodologias tradicionais	24
4. Tratabilidade de Problemas	26
4.1 Medidas de Complexidade.....	26
4.2 Classes de Problemas	27
4.3 Opções para Solução de Problemas Difíceis.....	28
4.3.1 Divisão e conquista	28
4.3.2 Exploração exponencial.....	28
4.3.3 Solução ótima em casos especiais	28
4.3.4 <i>Branch-and-Bound</i>	29

4.3.5 Algoritmos heurísticos.....	29
4.3.6 Algoritmos de aproximação.....	29
4.3.7 Algoritmos gulosos (<i>greedy</i>).....	29
4.3.8 Programação dinâmica.....	30
4.3.9 Programação Matemática.....	30
4.3.10 Simulação de têmpera.....	30
4.3.11 Simulação evolutiva.....	31
4.4 Problemas de Grafos.....	31
5. Algoritmos para Síntese Física.....	33
5.1 Algoritmos de Particionamento.....	33
5.2 Algoritmos para Planejamento Topológico.....	34
5.3 Algoritmos para Posicionamento.....	34
5.4 Algoritmos de Assinalamento.....	36
5.5 Algoritmos de Roteamento.....	37
6. Síntese Física em Tecnologias Sub-micrônicas.....	38
6.1 Performance Driven Synthesis.....	38
6.2 Mudança na importância das conexões.....	38
6.3 Mudanças na metodologia de síntese.....	39
PARTE 2 - Hierarquia de Problemas de Roteamento.....	41
7. Roteamento.....	42
7.1 Histórico e Abrangência.....	42
7.2 Classificação e Terminologia.....	43
7.2.1 Classificação de roteamento por objetivos.....	43
7.2.2 Classificações de roteamento quanto ao espaço.....	44
7.2.3 Classificações de roteamento quanto à modelagem do espaço.....	45
7.2.4 Classificação dos algoritmos de roteamento quanto ao processamento.....	46
7.3 Modelos que Condicionam o Roteamento.....	47
7.3.1 Posicionamento de terminais.....	47
7.3.2 Modelos de células.....	48
7.4 Problemas Clássicos de Roteamento.....	49
7.4.1 Canal de Roteamento.....	49

7.4.2 Caixa de conexões, ou <i>switch box</i>	52
7.4.3 Roteamento Planar	52
7.4.4 Roteamento em rio, ou <i>river routing</i>	53
7.4.5 Roteamento planar de uma caixa	53
7.4.6 <i>Single Row Routing Problem</i> , ou <i>SRRP</i>	53
7.4.7 Roteamento de subconjuntos planares tipo <i>SSPR</i> , <i>TSPR</i> , <i>ETSPR</i>	54
7.4.8 Problemas de assinalamento.....	55
7.4.8.1 Assinalamento de terminais às bordas (<i>BTA</i>).....	55
7.4.8.2 Assinalamento de pontos de cruzamento (<i>CPA</i>).....	55
7.4.9 Problemas de roteamento global	55
7.5 Algoritmos de Roteamento	57
7.5.1 Algoritmos de roteamento genéricos.....	57
7.5.1.1 Algoritmos genéricos baseados em caminhos em grafo	57
7.5.1.2 Algoritmos genéricos baseados em geometria	58
7.5.1.3 O algoritmo hierárquico.....	58
7.5.2 Algoritmos para roteamento de canal.....	59
7.5.3 Algoritmos para roteamento de caixas de conexão.....	60
7.5.4 Algoritmos para roteamento planar.....	60
7.5.5 Algoritmos para roteamento em rio	61
7.5.6 Algoritmos para roteamento de <i>SRRP</i>	61
7.5.7 Algoritmos para seleção de subconjuntos planares tipo <i>SSPR</i> , <i>TSPR</i> , e <i>ETSPR</i> ...	61
7.5.8 Algoritmos para assinalamento <i>BTA</i> e <i>CPA</i>	62
8. Hierarquia de Problemas em Roteamento.....	63
8.1 Dependência cíclica.....	63
8.2 Criação de problemas solúveis.....	64
8.3 Estudo de caso sobre roteamento de área	66
8.4 Estudo de caso sobre modelos para <i>QCL</i>	67
8.5 Convergência com desempenho.....	71
8.6 Relacionamento com outros temas	71
9. Conclusões.....	73
Bibliografia	74

Lista de Abreviaturas

<i>ASIC</i>	<i>Application Specific Integrated Circuit</i>
<i>BDD</i>	<i>Binary Decision Diagram</i>
<i>BIST</i>	<i>Built-In Self Test</i>
<i>BTA</i>	<i>Boundary Terminal Assignment</i>
<i>BTM</i>	<i>Boundary Terminal Model</i>
<i>CAD</i>	<i>Computer Aided Design</i>
<i>CI</i>	<i>Circuito Integrado</i>
<i>CPA</i>	<i>Cross Point Assignment</i>
<i>CTM</i>	<i>Center Terminal Model</i>
<i>DEC</i>	<i>Digital Equipment Corporation</i>
<i>DIP</i>	<i>Dual Inline Package</i>
<i>DRC</i>	<i>Design Rule Checking</i>
<i>EDA</i>	<i>Electronic Design Automation</i>
<i>EPLD</i>	<i>Electrically Programmable Logic Device</i>
<i>ETSPR</i>	<i>Equipotential Two Sided Planar Routing</i>
<i>fig.</i>	<i>figura</i>
<i>FPGA</i>	<i>Field Programmable Gate Array</i>
<i>FSM</i>	<i>Finite State Machine</i>
<i>GRC</i>	<i>Global Routing Cell</i>
<i>HCG</i>	<i>Horizontal Constraint Graph</i>
<i>HCVD</i>	<i>Horizontalmente Conectado Verticalmente Dividido</i>
<i>HCVC</i>	<i>Horizontalmente Conectado Verticalmente Conectado</i>
<i>HDVC</i>	<i>Horizontalmente Dividido Verticalmente Conectado</i>
<i>HVH</i>	<i>modelo de canal Horizontal-Vertical-Horizontal</i>
<i>IC</i>	<i>Integrated Circuit</i>
<i>IFIP</i>	<i>International Federation for Information Processing</i>
<i>ILP</i>	<i>Integer Linear Programming</i>
<i>LEA</i>	<i>Left-Edge Algorithm</i>
<i>LP</i>	<i>Linear Programming</i>
<i>LSI</i>	<i>Low Scale Integration</i>
<i>MBC</i>	<i>Mixed Block and Cell style</i>
<i>MCM</i>	<i>Multi-Chip Module</i>

<i>MFFC</i>	<i>Maximum Fanout-Free Cone</i>
<i>MIS</i>	<i>Maximum Independent Set</i>
<i>MKIS</i>	<i>Maximum k-Independent Set</i>
<i>MSO</i>	<i>Multiterminal Single-layer One-sided</i>
<i>MTM</i>	<i>Middle Terminal Model</i>
<i>NP</i>	<i>Não Polinomial</i>
<i>OTC</i>	<i>Over-The-Cell</i>
pág.	página
<i>PC</i>	<i>Parte de Controle</i>
<i>PCB</i>	<i>Printed Circuit Board</i>
<i>PLA</i>	<i>Programmable Logic Array</i>
<i>PO</i>	<i>Parte Operativa</i>
<i>PWB</i>	<i>Printed Wiring Board</i>
<i>QCL</i>	<i>Quickly Customized Logic</i>
<i>RAM</i>	<i>Random Access Memory</i>
<i>ROM</i>	<i>Read-Only Memory</i>
<i>RTL</i>	<i>Register Transfer Language</i>
<i>SMD</i>	<i>Surface Mounting Device</i>
<i>SRRP</i>	<i>Single Row Routing Problem</i>
<i>SSPR</i>	<i>Single Sided Planar Routing</i>
<i>TSPR</i>	<i>Two Sided Planar Routing</i>
<i>TTL</i>	<i>Transistor-Transistor Logic</i>
<i>UCLA</i>	<i>University of California at Los Angeles</i>
<i>ULA</i>	<i>Unidade Aritmética e Lógica</i>
<i>VCG</i>	<i>Vertical Constraint Graph</i>
<i>VHDL</i>	<i>Very High-speed integrated circuits Description Language</i>
<i>VHV</i>	<i>modelo de canal Vertical-Horizontal-Vertical</i>
<i>VLSI</i>	<i>Very Large Scale Integration</i>

Lista de Figuras

FIGURA 2.1 - Diagrama Y de Gajski (adaptado)	14
FIGURA 2.2 - Ciclo de projeto de um Circuito Integrado	16
FIGURA 3.1 - Classificação para a implementação de ASICs segundo [Rei92]	21
FIGURA 3.2 - Estilos de leiaute para um circuito e seus módulos.....	22
FIGURA 4.1 - Relações aceitas entre as classes de problemas	27
FIGURA 7.1 - Realização do roteamento segundo diferentes modelos	44
FIGURA 7.2 - Modelo físico para roteamento <i>Over-the-Cell</i>	45
FIGURA 7.3 - Modelos de células segundo [She95].....	48
FIGURA 7.4 - Modelo e terminologia de roteamento de canal	50
FIGURA 7.5 - Necessidades de roteamento em x, y, e mínimo global.....	50
FIGURA 7.6 - Roteamento de canal nem sempre é possível	51
FIGURA 7.7 - Grafos de restrições associados a um canal	51
FIGURA 7.8 - Roteamento em rio (<i>river routing</i>)	53
FIGURA 7.9 - Roteamento planar de uma caixa.....	53
FIGURA 7.10 - <i>Single Row Routing Problem (SRRP)</i>	54
FIGURA 7.11 - Grafos para roteamento global em <i>Standard Cell</i>	56
FIGURA 7.12 - O problema de árvores de Steiner.....	57
FIGURA 8.1 - Exemplo de decomposição de problemas	63
FIGURA 8.2 - Dependência cíclica entre dois problemas A e B.....	63
FIGURA 8.3 - Quebra de dependências cíclicas em uma passagem.	64
FIGURA 8.4 - Universo de problemas com sub-conjunto solúvel	65
FIGURA 8.5 - Abordagens para criação de problemas solúveis	65
FIGURA 8.6 - Roteamento <i>QCL</i> com estruturas de passagem.....	67
FIGURA 8.7 - Arquitetura dos canais de roteamento em [Sim92].	68

Lista de Tabelas

Tabela 4.1 - Comportamento de funções matemáticas complexas 26

Tabela 4.2 - Complexidade de tempo para problemas comuns em alguns grafos..... 32

Resumo

Este trabalho aborda a síntese física automática de circuitos integrados *VLSI*, e em especial a estrutura de roteamento dos mesmos. A síntese física tem como principal tarefa a geração de leiaute. A estrutura de roteamento é avaliada como um conjunto de problemas inter-relacionados, cuja solução deve completar todas as conexões do CI.

O projeto de um circuito integrado é realizado por um conjunto de etapas, conhecido como ciclo de projeto. A síntese física é a etapa responsável por produzir a estrutura geométrica do circuito e garantir o funcionamento elétrico de acordo com uma especificação estrutural e funcional mais abstrata. Há diferentes opções e metodologias para a implementação de um CI, resultando em diferentes estilos de leiaute. Na maioria dos casos, se tem um conjunto típico de problemas a serem solucionados, que são problemas de posicionamento e roteamento. Estes são em geral problemas tão complexos que exigem o uso das melhores técnicas e algoritmos conhecidos, e por esta razão faz-se necessário compreender as opções no tratamento de problemas complexos. O trabalho apresenta uma visão abrangente sobre todos estes aspectos, e discute também as necessidades advindas de novas tecnologias de fabricação.

A síntese das conexões do circuito, denominada roteamento, é a tarefa mais complexa na geração do leiaute do mesmo, e as demais etapas devem obrigatoriamente considerá-la. O roteamento possui diversas aplicações em diferentes instâncias de um projeto. Fatores como a forma das regiões para roteamento e a posição dos terminais a conectar condicionam fortemente a definição de cada problema. Os problemas de roteamento mais freqüentes são aqui caracterizados, e algoritmos desenvolvidos para sua solução são citados. Avalia-se as dependências que ocorrem quando se trabalha com uma hierarquia de problemas, e o modo pelo qual podem ser superadas. Este relacionamento é fundamental para o roteamento, e isto é exemplificado tomando-se as abordagens para roteamento de área e *QCL* como estudos de caso. O relacionamento do roteamento com outras etapas também precisa ser considerado, e estas podem ser etapas de síntese física, síntese lógica, ou mesmo de análise, como análise de temporização. Desta forma, busca-se um processo convergente, permitindo atender às restrições de desempenho com um tempo de projeto reduzido.

Palavras chave: *VLSI*, *EDA*, síntese física, roteamento

Abstract

This work addresses the automatic physical synthesis of VLSI integrated circuits, and in particular, its routing structure. The main task of the physical synthesis is the layout generation. The routing structure is considered as a set of related problems, whose solution builds all the interconnections of an IC.

The design of an integrated circuit is carried out by a set of steps, which represents the design cycle. Physical synthesis is the step that should provide circuit geometry and guarantee electrical operation against a more abstract specification, structural or behavioral. There are several options and methodologies to implement an IC, leading to different layout styles. In the most of cases, there is a typical set of problems to be solved, mainly placement and routing problems. These problems are in general so complex that they require the best techniques and algorithms known, and this is the motivation to understand the options for treating complex problems. This work provides a wide view about all these topics, discussing also the necessities coming from new fabrication technologies.

The synthesis of the interconnections in a circuit is called routing, and it is the most complex task for layout generation. Other tasks should consider it without exception. Routing has several applications on different instances of a design. Some parameters as the shape of routing regions and the terminal locations strongly affect the definition of each problem. Frequently used routing problems are characterized here, and the algorithms develop to solve them are mentioned. Dependencies that can occur when working with such a hierarchy of problems are evaluated, as well as the ways they can be overcome. This relationship is of fundamental importance for routing, and this is illustrated with case studies about area routing and QCL approaches. It must also be considered the relationship between the routing and other tasks. These tasks might be of physical synthesis, logic synthesis, as well as some behaviour analysis, like timing analysis. This way, a convergent process is targeted, in order to match performance constraints while preserving a short time-to-market.

Keywords: VLSI, EDA, physical synthesis, routing

1. Introdução

A crescente demanda no uso de sistemas computacionais cada vez mais complexos, e as diversas vantagens em embutir componentes eletrônicos nos produtos de consumo, tornam o projeto de circuitos integrados uma área crítica. Na verdade são estes circuitos que tornam possível a evolução da sociedade de informação. Um circuito integrado (*IC, Integrated Circuit*) consiste de um número de componentes eletrônicos construídos pela sobreposição bem definida de diferentes materiais sobre um substrato de silício, denominado de *wafer*.

Microprocessadores de uso genérico, contendo vários milhões de transistores em um único substrato são utilizados hoje em dia em computadores pessoais. O mercado aumenta espantosamente com a necessidade de projeto de circuitos integrados para aplicações específicas, os chamados *ASICs*, do inglês, *Application Specific Integrated Circuits*. Deve-se prover o projeto destes componentes sob fortes compromissos de desempenho, considerando velocidade, dissipação de potência, etc., e garantindo custo e tempo de projeto compatíveis com o mercado e com a obsolescência do produto.

Ferramentas de *CAD (Computer Aided Design)* tem sido utilizadas desde há muito para o projeto automatizado de sistemas eletrônicos, sendo aplicadas em diferentes etapas de sua concepção. O uso destas ferramentas permite que os projetistas trabalhem com informações mais significativas, evitando a necessidade de conhecimento e intervenção nas etapas de projeto mas próximas da tecnologia de fabricação. Estas etapas tratam com um número elevado de elementos, e são mais susceptíveis a erros. Quando é possível modelar e solucionar os problemas computacionalmente, realiza-se a tarefa de projeto automaticamente em tempo reduzido.

O crescimento na complexidade destes problemas provocou o surgimento de uma considerável indústria de *software* para automação de projetos eletrônicos, ou *EDA (Electronic Design Automation)*. Esta indústria faturou mais de US\$ 2 bilhões em 1996 [EDA97], sendo que a licença para rodar um pacote de ferramentas em um ponto de trabalho pode chegar ao preço de US\$ 1 milhão. *EDA* vem sendo considerada uma área importantíssima devido à diferença constatada entre o crescimento potencial na complexidade dos sistemas (58% ao ano), e o crescimento na produtividade dos projetistas (21% ao ano) [Bus97].

A área de *EDA* engloba uma grande variedade de ferramentas, para solucionar problemas os mais diversos. O resultado do projeto depende da forma como o sistema será implementado, considerando as tecnologias disponíveis. Em se tratando do projeto de circuitos integrados digitais *VLSI (Very Large Scale Integration)*, no qual este trabalho encontra seu foco, o objetivo é a obtenção da descrição das máscaras fotolitográficas utilizadas na sua fabricação.

Esta descrição é também chamada de *leiaute* do circuito, e representa a interface existente entre o projetista e a *foundry*, a qual detém a tecnologia de processos físico-químicos de produção. O *leiaute* é uma informação geométrica em duas dimensões, na qual são basicamente desenhados os transistores, elementos ativos do circuito, e suas interconexões. A síntese física é a parte do processo de projeto que trata de todos os aspectos envolvidos com a construção deste *leiaute*.

Este trabalho está dividido em duas partes principais. A primeira parte é denominada de “Projeto Físico de Circuitos *VLSI*”, e representa a área de abrangência do exame de qualificação do autor para o curso de doutorado. Nesta parte é discutido o ciclo de projeto de sistemas digitais, e em particular de circuitos *VLSI*, concentrando-se na síntese física e na automação deste processo. A segunda parte denomina-se “Hierarquia de Problemas de Roteamento”, e representa o tema de profundidade deste mesmo exame. Nesta parte, são discutidas as técnicas, modelos, algoritmos, e dependências que ocorrem para a síntese física das interconexões de um circuito.

A síntese física é de fundamental importância atualmente para o projeto de circuitos *VLSI*, já que estes são sistemas muito complexos que estão funcionando próximo a limites físicos das tecnologias de fabricação. As interconexões destes circuitos passaram a ter um peso muito maior nos últimos anos se comparadas com os elementos ativos nos mesmos. Assim, a escolha de estudar a hierarquia de problemas de roteamento, bem como suas relações, busca identificar e avaliar a otimização deste processo do qual depende o desempenho e o custo de projeto dos componentes.

PARTE 1 - Projeto Físico de Circuitos *VLSI*

2. Projeto de Sistemas Eletrônicos Digitais

O projeto de sistemas eletrônicos digitais em geral se dá através de um conjunto de transformações entre representações parciais. A compreensão tanto das representações quanto das transformações é fundamental neste processo. O ciclo de projeto envolve uma série de etapas, e um conjunto de tarefas típicas é desempenhado por ferramentas automáticas. Este capítulo provê uma visão geral destes assuntos, terminando por analisar o escopo da síntese física.

2.1 O diagrama de Gajski

O diagrama Y, originalmente proposto por Gajski e Kuhn [Gaj83], representa espacialmente a localização de descrições de sistemas em três domínios ou vistas: comportamental, estrutural e geométrico. Estes três domínios são representados pelos eixos que se afastam da descrição física exata, formando círculos concêntricos que representam os diversos níveis de abstração. Diversos autores apresentam mudanças significativas quanto aos níveis e conteúdo das descrições. A figura 2.1 apresenta a visão particular do autor, onde procura-se manter fidelidade com o diagrama original [Gaj88], ao mesmo tempo que se considera outras interpretações mais recentes, [Pre88], e [Wag94].

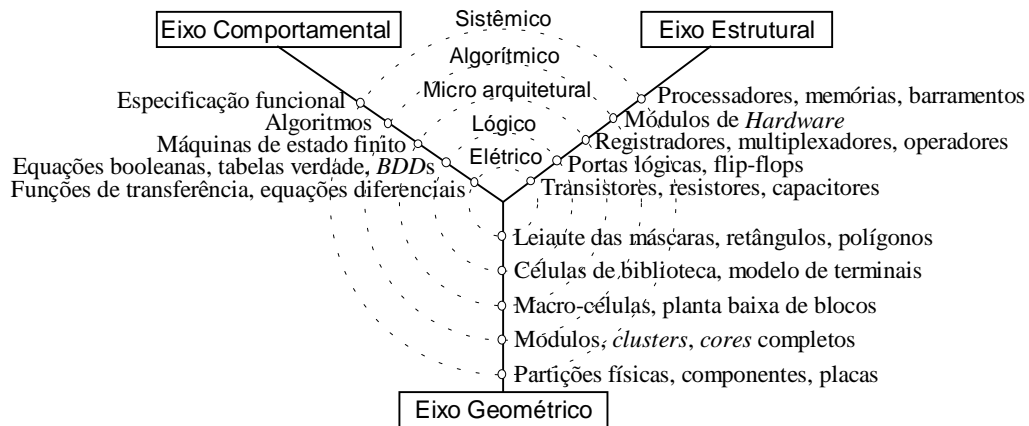


FIGURA 2.1 - Diagrama Y de Gajski (adaptado)

Apesar dos questionamentos formais de qualquer modelo, este diagrama permite enxergar separadamente as informações de natureza distinta, além de visualizar as transformações possíveis realizadas entre descrições de um sistema. O projeto de um sistema complexo é realizado através de um conjunto de transformações entre descrições em determinados níveis e domínios, até a obtenção da sua descrição detalhada em um ponto próximo do centro. Para circuitos digitais VLSI, este ponto é a interface com a *founary*: a informação geométrica do leiaute.

O uso sistemático de um conjunto determinado de transformações para a construção completa de um sistema é considerado uma metodologia de projeto. No diagrama Y, pode-se visualizar ou estabelecer uma metodologia de projeto específica, segundo as informações e transformações de que dispomos. Para cada metodologia,

existe um conjunto de transformações classificadas como síntese, pois agregam informação nova ao projeto. Outras transformações, ao contrario, analisam, extraem, comparam, ou simulam o funcionamento do sistema, tendo como principal objetivo a validação das etapas de síntese, e do projeto realizado [Wag94].

Uma metodologia de projeto é considerada *bottom-up* se ela inicia o projeto pela realização dos componentes mais primitivos, e vai compondo partes mais complexas até formar a descrição completa do sistema, tanto em detalhe, quanto em seu funcionamento global. Nesta classe estão as metodologias de projeto antigas, onde o projetista dispunha de um editor de esquemáticos e realizava o projeto usando componentes da biblioteca para compor macro elementos maiores.

Uma metodologia é considerada *top-down* se ela parte de uma descrição bastante abstrata e faz a decomposição do problema, solucionando-o em cada nível e gerando problemas em níveis mais detalhados, até a descrição final necessária. O uso de metodologias *top-down* está um pouco mais ligado à automação das etapas projeto. A expressão “compilação de silício” foi amplamente usada no meio científico para promover metodologias automáticas deste tipo, numa analogia com a engenharia de *software*. O ponto de partida é geralmente uma informação comportamental, já que a especificação de um sistema é dada pelo funcionamento adequado dentro de um ambiente próprio. Metodologias deste tipo são muito importantes para projetos portáteis, independentes de tecnologia.

Pode-se ver intuitivamente que nenhuma metodologia pode ser puramente *top-down* ou puramente *bottom-up*, mas será sempre um misto destas duas formas. Além disto, deve-se considerar tanto as etapas realizadas pelo projetista, quanto as informações já existentes, além daquelas geradas automaticamente por ferramentas de síntese. A classificação de ferramentas e metodologias torna-se facilmente ambígua, e esta é uma das razões pelas quais todos os conceitos e terminologias são imperfeitos.

2.2 Por que síntese?

Utiliza-se o termo síntese para designar uma transformação isolada ou um conjunto de transformações que sofre a descrição de um sistema, de forma a acrescentar detalhes sobre sua implementação, partindo-se de um nível inicial mais abstrato, e aproximando-se da descrição da sua realização física.

A principal questão é: a informação a ser agregada, vem de onde? Suponhamos que toda a informação venha de um conjunto de projetos humanos anteriores. Um projeto automático não pode satisfazer esta premissa. A principal razão é a diferença natural entre um projeto e outro. Mesmo que compartilhem de elementos comuns, a configuração de interconexão ou programação destes elementos é particular para cada projeto. Assim sendo, a automatização deste processo implica obrigatoriamente na criação de informação detalhada nova.

Como outro extremo, as informações agregadas poderiam ser totalmente criadas pela ferramenta, sem uso de elementos primitivos. A pureza neste modelo também é por si mesma um absurdo, seja porque a própria descrição final do projeto se vale de primitivas básicas, seja por que o conhecimento prévio de projetistas e de algoritmos está embutido na ferramenta de síntese. É valido, no entanto, buscar um modelo mais

próximo disto, pois que ele representa a generalização de problemas, e poderá prover a flexibilidade e coerência necessárias em projetos *top-down* mais complexos.

Há uma tendência natural em considerar-se o termo síntese com o sentido de geração mais pura ou formal, sem o uso de primitivas realizadas anteriormente de forma manual. Ao contrário, síntese é: método que reúne elementos simples para formar o composto, que parte do princípio e chega as conseqüências; generalização; demonstração de proposições por dedução das que já estão provadas; organização mental de um sistema; construção; resumo; [re]composição;

O oposto da síntese é a análise, o processo pelo qual se faz a decomposição do todo em suas partes, onde se vai do composto ao simples, ou dos efeitos às causas. A análise está mais presente no desenvolvimento das ferramentas e dos algoritmos que devem solucionar problemas. Da mesma forma, a análise faz parte do projeto automatizado dos componentes quando decompomos este processo em diversos subproblemas, assim como a síntese está presente na recomposição do todo utilizando as partes criadas e as primitivas.

Evitando o formalismo extremado, o termo síntese é aplicado para o conjunto de transformações que acrescentam detalhes construtivos, independentemente da origem das informações agregadas. Ele será válido tanto para projetos que se valem de subprojetos, quanto para os que geram detalhes dedicados, pois em ambos os casos informações primitivas vão sendo reunidas para a composição de todo o objetivo.

2.3 Ciclo de Projeto de Circuitos VLSI

Para caracterizar o ciclo de projeto de forma simplificada, adota-se um modelo seqüencial de projeto *top-down*, no qual cada etapa é completamente realizada e passa-se à etapa seguinte (fig. 2.2). A terminologia apresentada foi reunida de diversas fontes, e portanto não reflete exatamente nenhuma visão particular.

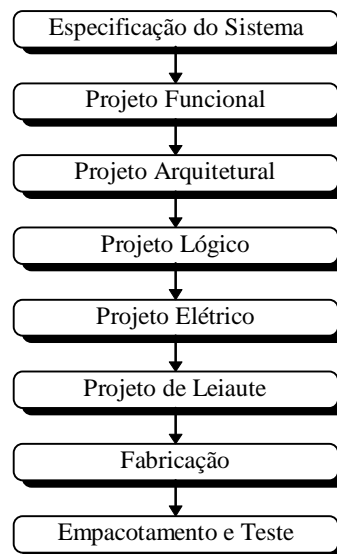


FIGURA 2.2 - Ciclo de projeto de um Circuito Integrado

Segundo [Pre88], cada uma destas etapas deve conter três passos: síntese, análise, e verificação. Já foi visto que a síntese agrega detalhes ao projeto. O passo de análise representa aqui a avaliação da descrição sintetizada em relação a seus requisitos ou à sua correção, e seria melhor chamado de avaliação. Finalmente, a verificação deve garantir que a descrição é equivalente a uma outra representação, sob todas as condições de interesse.

Especificação do Sistema - Esta etapa é de vital importância para todo o projeto, consome muito tempo, e exige importantes decisões técnicas e gerenciais. O sistema deve ser especificado em suas interfaces, protocolos, opções de arquitetura, desempenho, além de outros fatores mercadológicos e gerenciais.

Projeto Funcional - Visa obter a descrição comportamental abstrata de um sistema que funcione de acordo com as especificações. Esta etapa pode considerar alguns aspectos estruturais ou físicos que influenciem o comportamento, principalmente através de estimativas, mas deve preferencialmente preocupar-se com a funcionalidade. Normalmente a especificação do sistema em linguagem humana será traduzida para descrições comportamentais de *hardware*, como *VHDL*, *Verilog* ou *HardwareC*, e diagramas de tempo.

Projeto Arquitetural - Tendo o comportamento de todo o sistema, escolhe-se as opções arquiteturais para sua subdivisão, e também as de implementação interna de cada subdivisão. Aqui são consideradas estruturas como processadores, memórias, circuitos dedicados, e suas arquiteturas internas, como tipos de *cache*, uso de *pipeline*, etc. As mesmas linguagens já citadas são também empregadas. Um dos modelos mais gerais de arquitetura de um sistema é como um conjunto de máquinas que se comunicam. Cada uma é formada por uma parte operativa (PO ou *Data Path*) e uma parte de controle (PC), sendo a PC uma máquina de estados finitos (*FSM*) que gera sinais de controle para fluxo e processamento dos dados na PO.

Projeto Lógico - Nesta etapa o sistema é refinado estruturalmente. Módulos definidos na arquitetura são detalhados, e funções no domínio comportamental são traduzidas para o nível lógico estrutural. Usa-se de portas lógicas, registradores, multiplexadores, pequenos macro módulos de biblioteca (parametrizáveis ou não), barramentos, etc... Estes elementos devem ser selecionados conforme sua disponibilidade na tecnologia de implementação escolhida, pois não serão mais alterados.

Projeto Elétrico - Para os elementos não pertencentes a bibliotecas fixas, é realizado seu projeto elétrico a partir de elementos básicos, como transistores, resistores, diodos, capacitores, etc. E para o sistema como um todo, é considerado seu funcionamento e coerência elétrica. Transistores e conexões devem ser dimensionados para atender as necessidades de atraso e potência de cada circuito.

Projeto de Leiaute - Também denominado de projeto físico, é o conjunto de passos necessários para, a partir da descrição estrutural do circuito (composto por blocos, portas lógicas ou transistores), sintetizar a descrição geométrica final das máscaras, incluindo o leiaute daqueles que já o tem definido. Nesta etapa é praticamente obrigatório o uso de programas para realização automática das tarefas desejadas, em consequência do elevado número de elementos.

Fabricação - O circuito é fabricado por longos processos físicos e químicos a partir do seu leiaute completo, ou usando uma forma simplificada de implementação, dentre as descritas adiante. A fabricação de protótipos, ou sua emulação, é importante para que o componente possa ser testado fisicamente e em conjunto com seu ambiente de operação, antes da produção em massa.

Empacotamento e Teste - Inicialmente o que era uma simples escolha de um empacotamento físico para o substrato de silício, hoje engloba opções e tarefas bastante complexas, como o uso de *MCMs (Multi-Chip Modules)*, ou algoritmos para interconexão de pinos de um novo empacotamento. Finalmente, o teste para garantir o funcionamento de cada componente após sua fabricação é complexo em termos de hipóteses e cobertura de falhas, geração e aplicação de vetores de testes. Já é um padrão a existência de circuitos de auto teste dentro dos próprios componentes (*BIST*).

2.4 Ferramentas para *Electronic Design Automation*

Do ponto de vista de padrões de fato, há um consenso em enquadrar as ferramentas de *EDA* nas seguintes etapas:

Síntese de Alto Nível, ou Comportamental. Ferramentas necessárias para, a partir de uma descrição comportamental em nível algorítmico obter uma descrição estrutural em nível de macro componentes. São tarefas típicas da síntese de alto nível a alocação de recursos e o escalonamento de operações. A parte de controle resultante pode ser especificada tanto no domínio comportamental quanto no domínio estrutural em nível lógico, visto que a tradução entre um e outro pode ser feita com facilidade.

Síntese Lógica. Ferramentas de síntese lógica tem como objetivo tomar estas descrições estruturais ou comportamentais nos níveis *RTL* ou lógicos, e gerar a estrutura completa do circuito que será implementado, segundo as primitivas estruturais disponíveis na forma de implementação escolhida. São tarefas típicas a decomposição lógica, a otimização, e o mapeamento tecnológico.

Síntese de Leiaute, ou Síntese Física. Ferramentas necessárias para a geração do leiaute a partir da estrutura exata do circuito, formado tipicamente por portas lógicas e pela lista de interconexões. São tarefas típicas, detalhadas adiante, o posicionamento e o roteamento, razão pela qual são também chamadas de ferramentas de *Place & Route*.

É importante notar que estas três grandes classes tem sido usadas em seqüência nos projetos práticos realizados pela indústria, minimizando a interação existente entre uma etapa e outra. Apesar disto, face à maior complexidade e às novas características tecnológicas, o grau de interação necessário começa a ser bem maior.

É improvável que se mude os conceitos e termos usados para estas etapas de um momento para o outro, mas as considerações a respeito dos três domínios vem tornando-se obrigatórias em todos os níveis de abstração. As características novas, seus principais requisitos e problemas, serão discutidas brevemente ao final da primeira parte deste trabalho.

2.5 Escopo da Síntese Física

Conforme as definições dadas, a síntese física envolve-se principalmente com a geração do leiaute do circuito, tendo tipicamente uma descrição estrutural como ponto de partida. Isto pode nos levar a considerá-la como o conjunto de transformações referentes ao eixo geométrico apenas (as que o tem como origem ou destino). De fato, alguns autores consideram o eixo geométrico como eixo físico, já que a geometria é a abstração do circuito físico para um conjunto de máscaras em duas dimensões.

Por outro lado, em casos práticos há a tendência a considerar como síntese física aquelas transformações mais próximas do centro, abaixo de determinado nível. Isto ocorre porque elas se relacionam fortemente com a realização física do circuito, onde os três domínios estão em contato.

Novamente adota-se um modelo misto. Todas as tarefas que envolvem as informações geométricas serão consideradas como parte da síntese física, mesmo que estejam apenas sendo utilizadas numa transformação nos demais eixos, como estimativa para garantir que o comportamento ou a estrutura sejam válidos. São incluídas na síntese física também as transformações que ocorrem nos demais eixos quando estão muito próximas do centro, abaixo do nível lógico.

A razão para isto é simples: abstraindo-se questões metodológicas muito amplas, um circuito digital já tem seu comportamento e sua estrutura completamente definidos no nível lógico. Assim, faz parte da síntese física gerar o leiaute de forma a garantir este comportamento, o qual será expresso em termos de restrições de área, desempenho, etc... Assim, a síntese física inclui, por exemplo, o projeto elétrico, se este for necessário, que pertence ao domínio comportamental.

Cabe ainda duas observações. Em primeiro lugar, a comunicação entre a síntese física e as demais etapas se dá por estimativas, no sentido ascendente, e por restrições no sentido descendente, supondo que ela é a última etapa, como em metodologias tradicionais. Em segundo lugar, assume-se nos projetos mais complexos esta interação não é tão simples. Mesmo continuando válidas as definições feitas, busca-se uma generalização bem maior.

3. Metodologias de Projeto e Estilos de Leiaute

Dentre muitos fatores, as metodologias de projeto, e em especial de projeto físico, são em parte dependentes da tecnologia, ou forma de implementação dos circuitos. Assim, vejamos quais são as principais formas de implementação disponíveis, cada qual apropriada a uma faixa de produção e custo de projeto.

3.1 Implementação de Sistemas Digitais

A implementação de sistemas digitais já se tornara possível inicialmente com as tecnologias de Circuitos Impressos (*PCB - Printed Circuit Boards, ou PWB - Printed Wiring Boards*). A montagem de componentes simples, digitais ou analógicos, sobre uma placa onde as conexões estão “impressas” foi um primeiro passo em direção à integração. Este fato tem dupla importância neste escopo. Em primeiro lugar, muitos dos problemas de síntese física são equivalentes aos problemas estudados para automatizar o projeto de placas, e de fato, alguns algoritmos importantes foram desenvolvidos há décadas para estes problemas. Em segundo lugar, as placas de circuito impresso também evoluíram, e junto com a evolução na complexidade dos componentes que se montam sobre elas, são uma parte essencial nos projetos até os dias de hoje. Para reduzir o encapsulamento dos componentes e o tamanho dos furos nas placas, surgiram os dispositivos montados na superfície, ou *SMD - Surface-Mounting Devices*, e também o *Silicon on Board*, onde o substrato de silício é montado diretamente sobre a placa de circuito impresso, sem encapsulamento.

No sentido de eliminar ainda mais barreiras de desempenho impostas pelo tipo de encapsulamento, surgem os *MCMs (Multi Chip Modules)*. Os circuitos são montados sobre um outro substrato de silício, onde as camadas de interconexão, tipicamente em número de 6 a 60, são também integradas. Apesar dos diversos problemas de custo, dissipação de calor, entre outros, os quais ainda impedem a popularização destas tecnologias, espera-se um impacto na indústria equivalente ao uso dos *SMDs*, hoje comuns nos produtos eletrônicos. O próximo passo é a viabilização de *Wafer Scale Integration*, onde sistemas complexos podem ser integrados em um único *wafer*.

3.2 Formas de Implementação de ASICs

Restringindo o espectro para a implementação de circuitos dedicados em um único empacotamento, e considerando que a sua aplicação é voltada ao mercado de consumo, se tem a seguinte classificação para as formas de implementação disponíveis, segundo [Rei92], a qual é ilustrada na fig. 3.1. Esta classificação baseia-se no momento em que um circuito é diferenciado dos demais no processo de fabricação.

Os circuitos podem ser:

Circuitos personalizáveis por todas as máscaras - Têm sua fabricação diferenciada já nos processos iniciais de formação das regiões N, P, e polissilício de porta. São adequados a grandes volumes de produção ou a necessidades rigorosas de desempenho, dado que todos os detalhes podem ser definidos.

Circuitos personalizáveis por algumas máscaras - A diferenciação inicia somente a partir do processo de metalização, onde são realizadas as conexões. São também denominados de **pré-difundidos**, em razão de utilizarem matrizes de transistores previamente formados, estocadas em grande número. O custo dos processos iniciais é amortizado e o tempo de produção reduzido. Exemplos de estilos de leiaute desta categoria são *Gate Arrays* ou *Sea-of-Gates*.

Circuitos personalizáveis após o encapsulamento - A diferenciação é feita apenas através da programação de memórias internas, ou por ruptura ou fusão de elementos de conexão, podendo ser realizadas no próprio local de projeto. O tempo de prototipação é considerado nulo, e são limitantes o desempenho, o número de elementos implementados, e em função disto, o custo unitário alto. Fazem parte desta classe os *EPLDs (Electrically Programmable Logic Devices)* e *FPGAs (Field Programmable Gate Arrays)*.

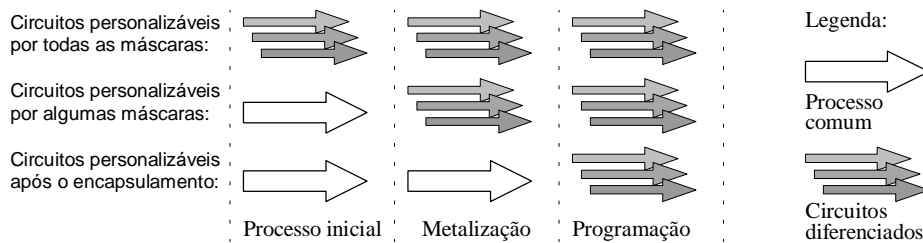


FIGURA 3.1 - Classificação para a implementação de ASICs segundo [Rei92]

Para as duas últimas formas de implementação, o projeto é feito com o uso de um conjunto de recursos previamente desenvolvidos. O desenvolvimento destes recursos recai no projeto desde o processo inicial, enquanto que a definição das últimas máscaras ou a programação dos dispositivos podem ser problemas semelhantes ou bem diferentes da programação das máscaras iniciais.

3.3 Metodologias de Projeto VLSI

Diversas metodologias surgiram para permitir a automatização do processo de projeto de circuitos VLSI programáveis por todas as máscaras. Estas metodologias estão ligadas principalmente a um conjunto de transformações e algoritmos que provêem um ciclo de projeto garantido, e também ao estilo de leiaute.

Por **Estilo de Leiaute** entendemos a organização geométrica que viabiliza ou provoca o uso de uma determinada metodologia. Inúmeros autores, no entanto, usam expressões como estilo de leiaute (*Layout Style*) [Sai95][Sap93] ou estilo de projeto (*Design Style*) [Bro91][She93] para algumas das formas de implementação. Na verdade, formas como pré-difundidos e programáveis em campo usam quase sempre estilo e metodologia particulares de leiaute, mas prefere-se adotar a classificação prévia por etapas de fabricação de [Rei92]. Apesar da dificuldade de classificação, os estilos e os conceitos comumente utilizados em metodologias de Projeto VLSI são os seguintes.

Em um projeto **Full-custom** se tem a liberdade para definir cada detalhe do leiaute dos transistores e conexões do circuito de maneira própria, sem o uso de padrões. Realizada por experientes projetistas elétricos e de leiaute, está técnica é empregada em circuitos ou partes de alto desempenho, como em microprocessadores de uso genérico.

Este estilo só pode ser implementado pela programação de todas as máscaras, razão pela qual aquela forma de programação também é denominada *Full-custom*. Como exemplo, a **DEC** faz a maior parte do projeto de seus processadores **Alpha** manualmente, em níveis elétrico e de leiaute, obtendo um desempenho bem maior que outras companhias com a mesma tecnologia de fabricação [Gru97]. Fatos como este demonstram a importância da síntese física e da pesquisa por melhores metodologias e ferramentas automáticas.

Em estilos *Cell Based*, o circuito é feito com o uso de blocos projetados anteriormente, que recebem o nome de células, macro células, módulos, etc. Este estilo surgiu como forma de reaproveitar o que já foi projetado. Usam-se os termos *General Cell*, *Macro-Cell* ou *Building Blocks* quando estes blocos tem tamanhos e formas arbitrárias. Apesar das dificuldades em se casar elétrica e geometricamente diferentes projetos, seu uso é ainda muito importante atualmente, sendo um meio-termo entre *full-custom* e *standard cell*. Como exemplo, o leiaute de um microprocessador inteiro pode ser embutido dentro de um outro projeto para fazer determinada parte do processamento. Os diversos problemas técnicos e de propriedade intelectual para reutilizar projetos anteriores inteiros têm sido muito discutidos atualmente.

O estilo *Standard Cell* usa uma estrutura de bandas de células de mesma altura, tomadas de uma biblioteca pré projetada e caracterizada em silício, e um conjunto de regiões para roteamento entre estas bandas. A metodologia associada permite um projeto rápido e seguro, onde a automação física é simplificada e o desempenho é garantido pelos elementos pré caracterizados. É um dos métodos preferidos para a implementação de **lógica aleatória**, aquela parte dos projetos que não goza de nenhuma estrutura regular. Em [She95] também é caracterizado um estilo denominado de *MBC*, onde o sistema de síntese física enxerga tanto os macro-blocos irregulares quanto as células individuais de uma região *standard cell*. A figura 3.2 ilustra de modo sintético estes estilos

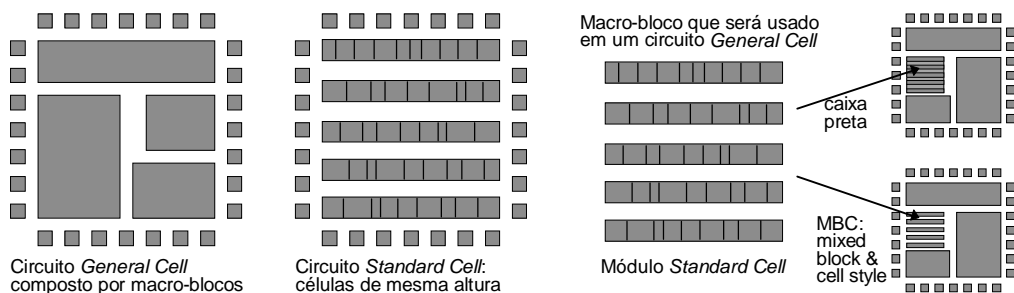


FIGURA 3.2 - Estilos de leiaute para um circuito e seus módulos

Por **Geração de Módulos** (ou compilação de módulos, segundo [Gaj88]), entende-se qualquer forma de gerar automaticamente o leiaute de um bloco com um razoável número de elementos (maior do que uma porta lógica). As vantagens são várias, como independência de tecnologia, otimização dedicada, flexibilidade, entre outras, principalmente se compararmos com módulos de biblioteca (*general cells*). Deve ser realizada por um sistema especialista que conhece uma determinada estrutura. Recebendo uma série de parâmetros, este pode gerar uma instância adequada de um universo de circuitos que diferem em termos de forma, tecnologia, desempenho, etc.

A **geração de módulos regulares** ocorre quando a estrutura lógica do circuito é regular, como em memórias *RAM*, *ROM*, bancos de registradores, *datapaths*, ou quando podemos gerar o leiaute de forma regular, como é o caso dos *PLAs*. Alguns módulos são considerados semi regulares, como somadores, multiplicadores, *ULAs*.

O uso de *PLAs* permite a geração de módulos em lógica aleatória com uma estrutura de leiaute regular. Entretanto, geração de módulos em lógica aleatória pode ser feita com estruturas de leiaute bem menos regulares, como nos estilos *gate-matrix* e *linear-matrix*. A própria geração de um circuito em *standard cells* não deixa de ser considerada como a geração de um módulo em lógica aleatória.

3.4 Problemas de Síntese Física

Existe um conjunto típico de tarefas que precisam ser resolvidas para a síntese física, sendo as mais importantes: particionamento, posicionamento, assinalamento de portas, terminais ou pontos de acesso, e roteamento.

O **roteamento** é a etapa responsável por realizar as conexões entre os terminais de entrada e saída dos diversos componentes do circuito, que são células folha ou blocos, por exemplo. A etapa de roteamento é bem caracterizada na segunda parte deste trabalho, que versa justamente sobre a hierarquia deste roteamento. As conexões tanto podem ser realizadas em espaços dedicados, quanto por sobre os elementos que se quer conectar, quando o número de camadas da tecnologia permitir. Em se tratando de um dos mais complexos problemas, o projeto das interconexões requer boas estimativas e planejamento prévio, para que, depois, cada instância sua seja solucionada exatamente nos níveis de menor abstração.

Assinalamento de portas, ou **alocação**, é a escolha de um componente já existente para a implementação de cada instância de porta do circuito. Ocorre principalmente quando a implementação é feita com circuitos já formados, como *FPGAs*. O **assinalamento de terminais** é necessário para encontrar o melhor terminal para cada conexão quando existem terminais equivalentes (duas entradas de uma porta **nand2**, por exemplo). Já o **assinalamento de pontos de acesso** é necessário quando um mesmo terminal de uma célula é acessível em diferentes posições. Outros problemas de assinalamento podem ocorrer caso as conexões já estejam implementadas, ou em decorrência de se ter dividido um problema em subproblemas.

Posicionamento é a tarefa de encontrar uma posição física para cada bloco ou célula que compõe o circuito, sem que nenhuma sobreposição ocorra, e com o objetivo de minimizar o tamanho das conexões. Em abordagens *general cell*, o posicionamento se torna muito complexo, e normalmente é executado de forma semi manual nos níveis mais altos de abstração geométrica. O planejamento de área neste nível é denominado **planejamento topológico** ou *floorplanning*.

O posicionamento é um pouco simplificado nas abordagens *standard cell*, e em todas abordagens nas quais uma das dimensões dos blocos a posicionar for padrão. Quando o número de camadas de roteamento não é grande, pode ser necessária a inclusão de células de passagem (*feedthroughs*) para as redes que cruzam de uma banda para outra. O posicionamento, ou assinalamento destas células pode consistir um

problema a parte. Diz-se que um posicionamento é **relativo** quando ele for dado como a ordem relativa em que as instâncias devem aparecer em cada banda. Ele será **absoluto** se informar, para cada instância, a sua posição definitiva exata. Normalmente estes dois problemas são realizados separadamente, devido à sua complexidade.

O **particionamento** é a divisão de um circuito em duas ou mais partes, segundo um determinado critério. Os critérios podem ser muitos, como mínimo número de conexões, número limitado de terminais, área, etc. O particionamento é necessário para subdividir o problema em problemas de menor complexidade, ou para acomodar um circuito em diferentes dispositivos, ou ainda como um dos métodos para se obter um posicionamento relativo bom.

Também é um problema de síntese física a **compactação** ([Sar96] cap. 7, [Pre88] seção 6.3), que tenta reduzir o tamanho do circuito ao máximo respeitando as regras de projeto. Como suporte e garantia para a síntese física, se tem diversos problemas de análise e verificação ([Pre88] cap. 8), como a **verificação de regras de projeto (DRC)**, e a **extração**, que, a partir de um leiaute, recupera o circuito elétrico ali implementado e suas características parasitas. Estes problemas não são abordados neste trabalho.

De acordo com cada metodologia, as tarefas básicas de síntese se configuram como problemas diferentes (a partir de dados diferentes), tendo restrições características, ou também diferentes funções objetivo. Estas diferenças são tantas que, em um caso prático, uma função objetivo de roteamento pode ser exatamente a oposta daquela necessária a outro caso. Por exemplo, rotear redes no canal menos denso para fins de balanceamento, ou rotear todas as redes que forem possíveis em um mesmo canal de tamanho fixo, maximizando sua utilização, para aproveitar os recursos locais.

3.5 Limitações das metodologias tradicionais

Um grande problema é o relacionamento entre as diversas etapas de cada metodologia, o que será enfatizado posteriormente. Ao estabelecer-se uma metodologia procura-se um caminho automático que permita chegar à solução, definindo-se a partir de então os subproblemas conseqüentes. Entretanto, a otimização do problema global não pode ser facilmente controlável através das interfaces definidas.

As metodologias de projeto tradicionais, como *Standard Cell* ou *Gate Array*, citadas acima, com seus estilos de leiaute característicos, são as mais utilizadas na prática para a concepção do leiaute, tanto em projetos genéricos (microprocessadores), quanto para circuitos dedicados. Contudo, partes ou projetos inteiros *full-custom* permanecem sendo essenciais para atingir os melhores resultados elétricos necessários em sistemas de alto desempenho.

Nas metodologias de síntese de leiaute mais utilizadas, existem alguns grandes inconvenientes, que são:

- dependência das regras de tecnologia, pois as bibliotecas de células precisam ser reprojatadas para cada nova tecnologia, o que resulta em um custo elevado;
- desempenho elétrico fixo, não se adequando às necessidades de cada projeto ou instância;
- impossibilidade de utilizar portas complexas, os quais podem melhorar o

desempenho, reduzir a potência dissipada e a área consumida, mas possuem tantas variantes que não podem estar disponíveis nas bibliotecas;

- aproveitamento da área do silício inferior aos projetos manuais;

Assim, a alternativa de geração de módulos em lógica aleatória tem grande importância, pois permite a geração automática completa do leiaute, incluindo os transistores e as suas interconexões. Com o domínio desta alternativa pode-se gerar leiautes muito mais eficientes, livrando-se ainda das limitações citadas, presentes nos métodos tradicionais, permitindo alcançar melhor desempenho [Mor94].

No domínio geométrico, é interessante o conceito de *Zero Routing Footprint*, apresentado em [She95], que se pode traduzir para área de roteamento nula. Segundo esta visão, o conjunto formado pela tecnologia de fabricação, a qual torna disponíveis as camadas metálicas para roteamento, somada à metodologia ou estrutura de roteamento, deve ser tal que favoreça a construção de um leiaute final onde não haja nenhum espaço dedicado somente às conexões. Este limite surge pela formação dos transistores no substrato, a qual não permite nenhuma sobreposição entre eles.

O roteamento, em contrapartida, pode ser sobreposto aos transistores, e portanto o objetivo de ter a sobreposição total pode e deve ser obtido para reduzir a área de silício, reduzindo tamanho de conexões, atraso, e por conseguinte também a potência dissipada, de grande importância para projetos de alta frequência, móveis, e pequenos. Este conceito é bastante inovador, pois foi pouco considerado nas metodologias tradicionais. Incursões neste campo são feitas utilizando as técnicas de roteamento sobre células, como *OTC*, transparência, roteamento de área, etc., citadas adiante.

4. Tratabilidade de Problemas

Para que ferramentas de *CAD* possam realizar tarefas de síntese, estes problemas, após modelados, precisam ser tratáveis computacionalmente. Mesmo com todas as técnicas de metodologia e subdivisão que são empregadas, os problemas de síntese física em geral, e em particular de roteamento, permanecem muito difíceis. Por esta razão, além de estruturas de dados e engenharia de *software* apropriadas, é necessário dominar um conjunto de técnicas teóricas e práticas para os tratar.

4.1 Medidas de Complexidade

Um algoritmo é uma seqüência de passos que pode ser executada, segundo um modelo de computação, para solucionar determinado problema. Para um mesmo problema existem diferentes algoritmos possíveis. Um algoritmo possui uma complexidade espacial e uma complexidade temporal. A sua complexidade espacial é a quantidade de memória necessária para sua execução. A complexidade temporal é a quantidade de tempo necessária para que ele complete a tarefa.

O tempo ou a quantidade de memória necessários são funções do número de entradas que o algoritmo recebe, e de constantes. Deve-se fornecer as entradas com o mínimo de tamanho necessário para definição completa do problema. Observando o comportamento das funções matemáticas na tabela 4.1, percebe-se que para problemas muito pequenos elas se cruzam de maneiras distintas. Dependendo do valor das constantes, a que apresenta menor valor pode ser uma ou outra. Entretanto, normalmente deve-se tratar com problemas maiores e com constantes pouco significativas. As células cinza representam problemas completamente intratáveis.

função	n = 2	n = 6	n = 10	n = 20	n = 10 ²	n = 10 ³	n = 10 ⁶
n	2	6	10	20	10 ²	10 ³	10 ⁶
3n	6	18	30	60	3 * 10 ²	3 * 10 ³	3 * 10 ⁶
n log ₁₀ n	0.6	4.7	10	26	2 * 10 ²	3 * 10 ³	6 * 10 ⁶
n ²	4	36	10 ²	4 * 10 ²	10 ⁴	10 ⁶	10 ¹²
n ³	8	216	10 ³	8 * 10 ³	10 ⁶	10 ⁹	10 ¹⁸
2 ⁿ	4	64	10 ³	10 ⁶	10 ³⁰	10 ³⁰¹	> 10 ⁵⁰⁰
n!	2	720	3 * 10 ⁶	2 * 10 ¹⁸	9 * 10 ¹⁵⁷	> 10 ⁵⁰⁰	> 10 ⁵⁰⁰

Tabela 4.1 - Comportamento de funções matemáticas complexas

Diz-se que $f(n) = \Omega(g(n))$ se existem constantes n_0 e c tal que $f(n) \geq c * g(n)$ para todo $n > n_0$. Assim, $f(n)$ é limitada abaixo por $g(n)$. Por exemplo, sabe-se que ao menos $n \log_2 n$ comparações são necessárias para ordenar n chaves genéricas. Assim, qualquer algoritmo para ordenação baseado em comparações requer tempo $\Omega(n \log_2 n)$. Diz-se que $f(n) = O(g(n))$ se existem constantes n_0 e c tal que $f(n) \leq c * g(n)$ para todo $n > n_0$. Assim, $f(n)$ é limitada acima por $g(n)$. Por fim, diz-se que $f(n) = \Theta(g(n))$ se $f(n) = \Omega(g(n))$ e $f(n) = O(g(n))$. A função Ω é usada como limite inferior teórico. A medida ‘O’ é usada para expressar que um algoritmo requer no máximo determinada função de tempo ou memória.

4.2 Classes de Problemas

Um problema é considerado fácil, ou tratável, se existe um algoritmo determinístico que o resolve em tempo polinomial. Um algoritmo determinístico é aquele que pode ser executado em um modelo real de computação, como uma máquina de Turing. Assim, temos a classe **P** de problemas que tem complexidade $\Theta(p(n))$, onde n é o tamanho da entrada e $p(n)$ é uma função polinomial de n , geralmente um polinômio de pequeno grau. Problemas que pertencem a classe **P** são: árvore de menor custo, menor caminho a partir de uma origem, casamento de grafos, ordenação, etc.

Na segunda classe temos os problemas **NP**, ou não-determinísticos polinomiais. São aqueles para os quais é possível apresentar um algoritmo não-determinístico que seria executado em um modelo irreal de computação, em tempo polinomial. O algoritmo funciona escolhendo a opção certa sempre que uma é necessária, até encontrar a solução. De outra forma somente é possível encontrar a solução ótima nesta classe, deterministicamente, se todas as combinações de entrada forem experimentadas, em tempo exponencial.

Os problemas **NP-completos** são aqueles problemas **NP** que estão intimamente relacionados, e para os quais, se fosse encontrado um algoritmo determinístico polinomial que o resolvesse, todos seriam resolvidos. Diz-se que um problema P_1 se reduz a outro problema P_2 se e somente se, dado uma instância I_1 de P_1 , é possível construir uma instância I_2 de P_2 em tempo polinomial tal que solucionando I_2 se obtém a solução de I_1 . Exemplos de problemas **NP-completos**, já pertencentes aos problemas considerados difíceis ou intratáveis, são: caixeiro-viajante, partição de grafos, árvore de Steiner, assinalamento quadrático, e grande parte dos problemas de síntese física.

Um **problema de decisão** é aquele para o qual a solução é um valor binário, verdadeiro ou falso, e um **problema de otimização** é aquele que deve dar como resultado a melhor solução dentre um conjunto possível. Por exemplo, saber se um determinado grafo pode ser colorido com k cores é um problema de decisão, enquanto que encontrar o menor k tal que um grafo possa ser colorido com k cores é um problema de otimização. Se a versão de decisão de um problema P é **NP-completa**, então a versão de otimização do mesmo problema é **NP-difícil**.

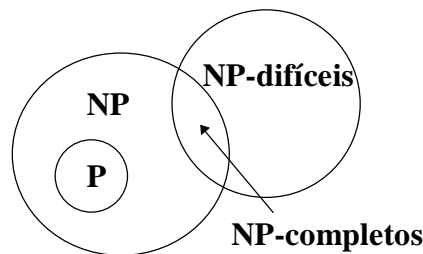


FIGURA 4.1 - Relações aceitas entre as classes de problemas

Um problema **NP-difícil** clássico que não é **NP-completo** é, dado um algoritmo qualquer e uma seqüência de valores de entrada, saber se o algoritmo termina ou entra em um laço infinito. Não existe algoritmo não-determinístico para solucioná-lo. Um problema **NP-difícil** clássico que é **NP-completo** é o de satisfabilidade. Dado um circuito

de n entradas e uma saída, o que corresponde a uma fórmula cujos literais são as entradas diretas ou negadas, deve-se responder se é possível uma combinação de valores para as variáveis de modo que a saída seja verdadeira? O problema equivale à justificação, necessária para testabilidade ou identificação de caminhos falsos.

A figura 4.1 apresenta as relações que se acredita existir entre estas classes de problemas. Para melhor compreensão, podemos encontrar literatura especializada em algoritmos [Hor97], e em tratabilidade de problemas [Gar79]. Pequenos resumos que podem ser muito úteis são também encontrados em diversos livros de projeto físico para circuito *VLSI*, dado sua importância, como em [Len90] [She93] [Sai95] [She95] e [Mic94].

4.3 Opções para Solução de Problemas Difíceis

Infelizmente, quase todos os problemas de síntese física são NP-difíceis, e não há algoritmos de tempo polinomial conhecidos para solução. Acredita-se também que não possa jamais haver, apesar de não provado matematicamente. Ainda, com o elevado número de elementos em problemas práticos, mesmo algoritmos com complexidade polinomial quadrática podem ser intoleráveis. Para tratar problemas deste tipo, dispõe-se de uma série de técnicas oriundas da matemática, filosofia, e da própria informática, as quais são resumidas a seguir.

4.3.1 Divisão e conquista

Esta técnica consiste em dividir sucessivamente o problema em problemas menores até que uma solução simples exista, de forma que a combinação simples destas soluções parciais forme a solução completa do problema. É o que se faz com todos os problemas complexos em geral. Como algoritmo, entretanto, cabe uma definição mais precisa. Dado um problema a ser resolvido para n entradas, o algoritmo divide as entradas em k grupos distintos, gerando k novos problemas. Frequentemente estes problemas são do mesmo tipo do original, e então aplica-se recursivamente o método até que o tamanho dos subproblemas seja tratável, ou mesmo trivial, como quando têm 0, 1 ou 2 entradas. A solução de cada grupo de k subproblemas divididos é combinada adequadamente até se ter a solução do problema original.

4.3.2 Exploração exponencial

Dependendo do tamanho do problema, pode ser válido testar todas as soluções possíveis. Em alguns casos temos um problema pequeno, ou podemos limitá-lo a certo tamanho, como acontece por divisão e conquista, para garantir que seja viável a pesquisa exponencial.

4.3.3 Solução ótima em casos especiais

Felizmente, como veremos em problemas de grafos, por exemplo, os problemas não ocorrem em sua generalidade, sendo passíveis de restrições naturais ou artificialmente impostas. Assim, pode-se restringir um problema a casos especiais para os quais existam algoritmos de menor complexidade para sua solução.

4.3.4 *Branch-and-Bound*

A técnica de *branch-and-bound* procede a exploração do espaço (ou conjunto de soluções) como a pesquisa em uma árvore que representa as decisões possíveis, e tem como folhas todas as soluções. Esta árvore não existe fisicamente, mas é gerada em tempo de execução. A vantagem reside no fato de que, tendo um meio de pré computar a qualidade da solução em um nível intermediário da árvore, ramos inteiros que são ruins podem ser podados, e passa-se a pesquisar outras alternativas. No pior caso, se tem a exploração exponencial completa, e se houver uma solução ela será encontrada.

4.3.5 Algoritmos heurísticos

Mesmo quando todo o espaço de combinações é possível, as instâncias que ocorrem na prática apresentam uma distribuição característica. Algoritmos heurísticos em geral se valem desta “normal” e de regras pragmáticas para apresentar soluções próximas das ótimas com baixa complexidade. Eles possuem a desvantagem de que sempre será possível inviabilizar a solução apresentando um caso especial, considerado ruim, de entradas.

4.3.6 Algoritmos de aproximação

Algoritmos de aproximação são aqueles que apresentam uma solução correta com a garantia de que esta solução está dentro de uma determinada percentagem da solução ótima. Fazendo uma análise do pior caso e do melhor caso que o algoritmo pode produzir, podemos avaliar sua complexidade e a proximidade das suas soluções em relação àquela que é ótima.

4.3.7 Algoritmos gulosos (*greedy*)

Os algoritmos gulosos são geralmente empregados para selecionar um conjunto de entradas que satisfaz determinadas restrições, ao mesmo tempo em que maximiza ou minimiza determinada função objetivo. O algoritmo trabalha por etapas, e a cada etapa julga a inclusão de uma entrada separadamente, em função de sua qualidade para formar uma solução ótima. Somente são aceitas entradas que mantêm uma solução válida, e, dentre elas, uma que melhor satisfaça a função objetivo para a solução parcial.

Os algoritmos gulosos também utilizam-se de heurísticas para tomar as decisões que têm grande probabilidade de levar à solução ótima no início, já que não desfazem nenhuma operação. A ordem de decisões pode ser auxiliada pelo critério de prioridades, ou ainda estar implícita em uma abordagem hierárquica, como divisão e conquista. Já as heurísticas precisam capturar restrições e objetivos características de cada problema, principalmente através de estimativas. Do ponto de vista teórico, um algoritmo guloso corresponde a um processo que tenta se aproximar do mínimo desde o princípio, nunca corrigindo suas decisões, sendo que é incapaz de encontrar um mínimo global se estiver convergindo para um mínimo local.

4.3.8 Programação dinâmica

Da mesma forma que um algoritmo guloso, a programação dinâmica se aplica a problemas cuja solução pode ser vista como um conjunto de decisões, ou entradas, a serem selecionadas. Para alguns destes problemas, podemos selecionar as decisões independentemente de sua ordenação, e todos estes podem ser solucionados otimamente por um algoritmo guloso. Mas isto não vale para quaisquer problemas. Para os que são dependentes de ordenação, seria preciso enumerar todas as seqüências possíveis de decisões e selecionar a melhor, o que resulta em complexidade de tempo exponencial.

A programação dinâmica reduz drasticamente a quantidade de seqüências pesquisadas evitando aquelas que não podem fazer parte da solução ótima. Isto pode ser explorado para aqueles problemas que têm o princípio da optimalidade. Este princípio vale quando uma seqüência ótima de decisões tem a propriedade de que, seja qual for o estado inicial e a decisão atual, as decisões restantes devem constituir também uma seqüência ótima de decisões. É o que ocorre, por exemplo, na pesquisa pelo caminho mais curto em um grafo. Então, enquanto um algoritmo guloso gera apenas uma única seqüência de decisões, a programação dinâmica gera várias, evitando aquelas que certamente não participam da solução ótima.

4.3.9 Programação Matemática

As técnicas de programação matemática surgiram no campo de pesquisa operacional, para solucionar problemas onde deve-se encontrar valores para um conjunto de variáveis, de forma a satisfazer um conjunto de inequações como restrições, e ainda maximizar ou minimizar uma outra equação em função destas variáveis. Diversos problemas práticos de produção, transporte, bem como problemas científicos, são naturalmente bem expressos desta forma. A programação linear (*LP*) é usada quando as equações de restrições e de função objetivo são funções lineares. Quando alguma delas for não linear (quadrática, por exemplo), trata-se de programação não linear [Las70]. A programação linear inteira (*ILP*) restringe o valor das variáveis para números inteiros. Isto torna o problema mais difícil, já que pode ocorrer que nem o mínimo real, nem seu arredondamento sejam o mínimo inteiro [Hu70]. A programação linear 0-1 trabalha com variáveis binárias que representam a ausência ou presença, e não quantificação. O fluxo em redes (*network flows*) é um problema particular de programação linear que consiste em maximizar o fluxo entre dois nodos de um grafo conectado com capacidades específicas em suas arestas. Diversos métodos e teorias podem ser encontradas na literatura para resolver estes problemas, e também há ferramentas já desenvolvidas para tal. Os problemas de programação não linear são obviamente os mais difíceis.

4.3.10 Simulação de têmpera

A **técnica de simulação de têmpera** (*simulated annealing*) foi introduzida por Kirkpatrick em 1983, e é amplamente utilizada na solução de problemas de otimização combinatória, principalmente quando o espaço de soluções não é bem compreendido. Esta técnica já foi aplicada a diversos problemas de *CAD*, e se vale da simulação de um processo físico de resfriamento lento usado para cristalização de metais, sendo iterativo e aleatório. Neste processo, tenta-se evitar a parada em mínimos locais, que ocorre em outros algoritmos, através da busca por uma configuração de menor energia em um

sistema fechado de moléculas. Partindo de uma solução inicial qualquer e de uma temperatura inicial alta, gera-se uma alteração para a qual calcula-se a diferença de qualidade, ou escore, Δs . Se $\Delta s < 0$ a alteração é aceita, e em caso contrário, ela é aceita com probabilidade $e^{-\Delta s/t}$. A medida que fazemos a temperatura t cair, a probabilidade de aceitar escores piores decresce. O processo tem que ser ajustado através da escolha certa da temperatura inicial, da função de decréscimo de temperatura, e da condição de equilíbrio em cada temperatura. São incluídas heurísticas para a seleção de alterações e avaliação do escore quando deve-se modelar características restritivas e de desempenho particulares de um problema, as quais o algoritmo não modela naturalmente.

4.3.11 Simulação evolutiva

Outra classe de algoritmos combinatórios ou aleatórios (*randomized algorithms*) são os algoritmos baseados em simulação evolutiva, ou algoritmos **genéticos**. Em processos biológicos, as espécies tornam-se melhores na medida em que evoluem de uma geração para outra. O conceito de conservar genes bons e eliminar os ruins tem sido explorado para alguns problemas de otimização combinatórios. Um algoritmo genético inicia com um conjunto de configurações a que chamamos população. Cada indivíduo desta população representa uma solução possível, caracterizada pela presença dos genes em determinada posição. A cada geração, os genes dos indivíduos com melhores características são combinados para formar novos indivíduos. Então, os piores indivíduos são eliminados, de forma a manter a população constante. Operações de combinação dos genes podem ser cruzamento, mutação ou inversão.

Os algoritmos de simulação de têmpera e evolutivos são ambos computacionalmente intensivos, sendo que estes últimos tendem a ser mais eficientes e mais rápidos. Entretanto, uma vez que tratam com uma série de soluções ao mesmo tempo, necessitam de mais espaço.

4.4 Problemas de Grafos

Muitos dos problemas de síntese física podem ser modelados como grafos, e a vantagem reside em que, além da natural abstração, estes problemas foram muito bem estudados em teoria de grafos, sub-área da matemática. Um grafo é composto por um conjunto de vértices e um conjunto de arestas, cada uma entre dois vértices. Um resumo da terminologia básica de grafos pode ser encontrado em [She93] e [She95], além de problemas, algoritmos e classes particulares de grafos.

As seguintes classes de grafos são importantes para circuitos *VLSI*: grafos perfeitos em geral; árvores; grafos planares; grafos bipartidos; grafos de intervalos; grafos de permutações; grafos de arcos circulares; grafos circulares; grafos policirculares; grafos de comparabilidade e co-comparabilidade; grafos cordais; grafos de segmentos; grafos de planos convexos; grafos de cordas; grafos de direções;

Alguns problemas básicos de grafos são: máximo conjunto independente de vértices (*MIS*); máximo conjunto k -independente (*MKIS*); máximo *clique* (poderíamos traduzir por facção ou grupo); mínima coloração; máximo conjunto bipartido; conjunto dominante; etc. Apesar da maioria destes problemas serem NP-difíceis para grafos em geral, eles são frequentemente polinomiais para algumas classes particulares de grafos.

Grafos de intervalo representam interseções de segmentos paralelos, e são importantes para roteamento. Devido à sua estrutura característica de interconexão, relacionada aos grafos perfeitos cordais e de co-comparabilidade, seus problemas podem ser solucionados em tempo polinomial.

Grafos de permutação representam cruzamentos de conexões entre terminais dispostos entre duas retas paralelas. São, portanto, úteis na solução de roteamento por conjuntos planares. Grafos de permutação também estão relacionados aos grafos perfeitos de comparabilidade e co-comparabilidade, e tem seus problemas solúveis em tempo polinomial.

Grafos circulares representam cruzamentos de cordas entre pontos dispostos num círculo. São equivalentes a **grafos de sobreposição**, um tipo particular de grafo de intervalos onde nenhum intervalo está completamente contido em outro. Grafos circulares, assim como os de permutação, são importantes para avaliar a planaridade de um problema. Podemos executar o roteamento planar entre terminais dispostos no limite de uma região delimitada por qualquer polígono, se o grafo circular correspondente não contiver nenhuma aresta, desde que haja espaço para acomodar a largura das conexões.

Não se deve confundir a planaridade de um problema modelado como grafo com a planaridade do grafo, que é a possibilidade de desenhá-lo em um plano sem cruzamento de arestas. Um grafo só não é planar se contiver os grafos completos K_5 ou $K_{3,3}$. Isto não ocorre normalmente com circuitos se modelarmos os vértices como terminais e as arestas como conexões, pois não temos múltiplas conexões diferentes para um mesmo terminal, mas redes equipotenciais. Daí a necessidade de, modelando as conexões como vértices, e suas interseções no espaço como arestas, representando intervalos, sobreposições ou permutações, selecionar os conjuntos planares nos tipos de grafo descritos acima. Como contra exemplo, se pode citar a modelagem dos cruzamentos de conexões entre células do tipo *general cell*. Pode-se modelar as células como vértices (não seus terminais), e então avaliar a planaridade das conexões pela planaridade do próprio grafo,

Em resumo, para os problemas mais frequentes em tipos de grafos que modelam condições restritas ou topológicas de roteamento, existem algoritmos de complexidade polinomial, conforme a tabela 4.2.

Tipo de Grafo	<i>MIS</i>	<i>Clique</i>	<i>MKIS</i>
Intervalo	$O(n \log n)$	$O(n \log n)$	$O(n^2)$
Permutação	$O(n \log n)$	$O(n \log n)$	$O(k n^2)$
Circular	$O(n^2)$	$O(n^2 \log n)$	NP-difícil

Tabela 4.2 - Complexidade de tempo para problemas comuns em alguns grafos

5. Algoritmos para Síntese Física

Já que muitos dos problemas de síntese física são NP-difíceis, são utilizadas diversas das técnicas gerais vistas anteriormente, além de heurísticas particulares desenvolvidas para cada caso. Alguns dos algoritmos mais usados para estes problemas são comentados a seguir.

5.1 Algoritmos de Particionamento

Os algoritmos de particionamento podem ser classificados primeiramente em relação à existência ou não de uma partição inicial. Na ausência de uma partição inicial, os algoritmos são ditos **construtivos**, pois fornecem uma partição a partir da estrutura de interconexão do circuito (*net-list*). Já os algoritmos **iterativos** recebem um conjunto de partições já realizadas e alteram as partições do mesmo *net-list* para melhorar algum dos critérios de particionamento. Algoritmos construtivos são normalmente mais rápidos e são usados para prover partições iniciais razoáveis, enquanto que os iterativos são usados para refiná-las até que sejam satisfatórias.

O critério usado para avaliar um particionamento pode ser uma combinação dos seguintes: corte mínimo, mínimo número de conexões entre partições; atraso em redes críticas, que pode ser muito maior nas conexões entre partições; número de terminais, quando estes estão limitados pelos tipos de componentes; área de cada partição, também restrita por unidade ou custo; e número de partições, um compromisso entre a complexidade de projeto de cada uma e a complexidade de composição de todas.

Os algoritmos de particionamento também podem ser determinísticos ou não. São **determinísticos** quando apresentam sempre a mesma solução para um mesmo conjunto de entradas. Eles serão **aleatórios**, ou **combinatórios**, quando se utilizarem de opções aleatórias como técnica de variação para a otimização.

Os algoritmos baseados em **migração de grupos** são algoritmos iterativos que trocam elementos entre partições para refiná-las. Entre eles se destacam os algoritmos de: Kernighan e Lin (K-L), trocando células entre partições; Schweikert e Kernighan, onde a mesma técnica opera com troca de redes; Fiduccia e Mattheyses, que reduz a complexidade de K-L para $O(t)$, onde t é o número de terminais; entre outros [She93].

O algoritmo de **têmpera simulada**, iterativo e probabilístico, tem sido muito usado para particionamento e posicionamento. As alterações provocadas no sistema correspondem a movimentos de células entre partições, posições ou *slots*. Este algoritmo está disponível no pacote de ferramentas de posicionamento **TimberWolf**, desenvolvido na universidade de **Stanford**.

Algoritmos evolutivos, ou genéticos, são também muito utilizados. Neste caso, o fenótipo dos indivíduos representam as posições das células nas partições ou no espaço, e as operações para a criação de cada nova geração representam os movimentos das células.

Dentre outros métodos e heurísticas para particionamento, é importante o conceito

de *MFFC, Maximum Fanout-Free Cones* [Con95a]. Esta técnica permite formar grupos de componentes interconectados de modo que o número de conexões internas seja maior do que o número de conexões externas. Além de capturar o critério de localidade das conexões, esta técnica permite otimizar medidas como o atraso, contemplando o sentido das conexões (de fonte e carga).

5.2 Algoritmos para Planejamento Topológico

O problema de posicionar blocos com tamanhos e formas arbitrárias ocorre quase que inevitavelmente na montagem do leiaute final de um circuito complexo. Apesar de algoritmos de particionamento e posicionamento serem utilizáveis, há diversas particularidades e tarefas adicionais necessárias.

Os blocos a posicionar podem ser **blocos duros** ou **blocos flexíveis**. Para os primeiros há uma definição prévia do seu tamanho, o que ocorre, por exemplo, ao incluir o leiaute inteiro de um projeto anterior. Já os blocos flexíveis podem ser gerados com diferentes relações de aspecto, sendo que existe uma função custo associada à sua forma.

Usando somente blocos flexíveis, por uma questão de simplificação, temos que o planejamento pode ser feito por sucessivas bipartições com tamanhos e sentido arbitrários. Também podemos ter partições múltiplas com estrutura topológica definida. A estimativa de área para interconexão deve ser considerada, e no caso em que são utilizados canais para roteamento, deve-se fazer a ordenação dos canais. Esta ordem é necessária em virtude do tamanho de um canal (que é variável) influenciar na posição dos terminais e no roteamento dos canais perpendiculares que com ele fazem fronteira.

5.3 Algoritmos para Posicionamento

A despeito da existência dos casos mais gerais, limita-se normalmente a discussão sobre posicionamento para o posicionamento baseado em bandas (*row-based layouts*). Neste contexto, deve-se encontrar uma posição adequada, em determinada banda, para cada um de um conjunto de n objetos que tenham uma das dimensões padronizadas. Nenhuma sobreposição pode ocorrer e procura-se viabilizar o roteamento necessário. O circuito final deve ainda respeitar uma certa forma pré estabelecida.

A etapa de posicionamento poderá reduzir o problema de roteamento necessário, ou, se mal feita, torná-lo insolúvel. Para tanto, guia-se primordialmente pela estimativa do tamanho das conexões. As medidas de estimativa de comprimento de conexões para redes com mais de dois pinos podem ser baseadas nos seguintes critérios ([Sai95] cap. 4): semi-perímetro do retângulo envolvente (*bounding-box*); grafo completo dos terminais; cadeia mínima de terminais; múltiplas conexões da fonte para a carga; aproximação de árvore de Steiner; ou ainda árvore mínima. Dentre inúmeras funções objetivo para posicionamento temos: minimizar tamanho total de conexões; minimizar o número máximo de conexões que cortam linhas imaginárias; minimizar a densidade máxima; ou minimizar o atraso de fonte para carga;

Diversos algoritmos de posicionamento são baseados em particionamento, e neste caso, usam-no como uma etapa prévia, ou mediante alterações nos mesmos algoritmos já apresentados. Em geral, podemos realizar os passos na seguinte ordem: particionamento

inicial, particionamento iterativo; geração do posicionamento relativo, e finalmente posicionamento absoluto.

Nos métodos **baseados em particionamento**, são executadas sucessivas partições, desde dividir o circuito completo em duas ou mais partes, até o momento em que cada parte tenha o tamanho aproximado de uma célula. É evidente a dificuldade desta medida aproximada, que ocorre pelo número arbitrário de células. Ao final, o erro de aproximação discreto pode conter grandes distorções, mesmo por que a maior célula pode ter até 10 vezes o tamanho da menor. Daí a necessidade de adaptação do particionamento para minimizar tais distorções, e também de uma etapa separada para a geração do posicionamento relativo final, também chamado de posicionamento linear.

Para que, a cada partição, se possa considerar a posição de terminais que já estão atribuídos à partições vizinhas em um determinado nível, usa-se propagação de terminais ([Sai95] pág.164). Estes são considerados como pinos externos fixos da partição sendo repartida, assim como as conexões com os *pads* do circuito o são para o circuito inteiro.

Nos algoritmos baseados em **crescimento de aglomerados**, são utilizadas sementes, escolhidas ao acaso ou com um bom critério como número de conexões, por exemplo, em torno das quais as demais células vão sendo acrescentadas. Para cada célula procura-se acrescentá-la ao conjunto com o qual tem mais conexões. Há diferentes abordagens para a seleção das sementes e para a inclusão em um dos aglomerados.

Nos algoritmos de **posicionamento dirigido por força**, toma-se as células uma a uma e, para cada qual calcula-se a posição de força resultante nula. Tenta-se colocar a célula nesta posição, e quando estiver ocupada, a célula que a estava ocupando é selecionada como candidata para um novo movimento. Este tipo de algoritmo é considerado com algoritmo numérico. Para evitar a sobreposição de células, pode-se modelar forças repulsivas entre as células não conectadas ([Sar96] seção 2.3.2).

Diversos algoritmos de otimização combinatória tem sido empregados para posicionamento, dentre eles: simulação de têmpera, simulação evolutiva, programação linear, assinalamento matemático, resolução de redes resistivas, redes neurais, entre outros. O leitor pode encontrar descrições e referências detalhadas de cada uma destas técnicas, bem como de seus resultados em [Pre88] cap. 4, [She93], [Sai95] cap. 4, [Sar96] seção 2.3, entre outros.

A simulação de têmpera, enquanto técnica apropriada a problemas de otimização, foi considerada como uma das melhores opções, juntamente com algoritmos genéticos. Atualmente, estes sistemas tem sido fortemente questionados pela quantidade de tempo que requerem, e por não capturarem determinados aspectos importantes dos circuitos, principalmente sua estrutura, e objetivos exatos de desempenho. Assim, uma série de novos algoritmos tem se mostrado mais eficientes para posicionamento. Dentre estas encontra-se: combinar outras técnicas com simulação de têmpera para melhorar seu desempenho [Sun95]; basear-se na estrutura do circuito [Tsa95]; ou usar técnicas de particionamento de forma mais inteligente, guiando-se pelo desempenho do circuito [Hua97].

5.4 Algoritmos de Assinalamento

Um assinalamento é um problema teórico de otimização, e consiste em encontrar um mapeamento bi-unívoco entre dois conjuntos, como por exemplo, conjunto de recursos R e conjunto de tarefas T . Um assinalamento linear ocorre quando busca-se minimizar o somatório de custos C_{ij} para todo par de elementos (i,j) , $i \in R$, $j \in T$. O assinalamento quadrático ocorre quando há um custo adicional $A_{k,l}$ associado a pares (k,l) de elementos de um mesmo conjunto (k e $l \in R$ ou k e $l \in T$). Assinalamentos linear e quadrático são bem estudados em teoria. Os problemas de assinalamento de portas e pinos em circuitos *VLSI* são generalizações onde o custo adicional está associado a grupos de elementos, e não somente a pares.

O assinalamento de portas surgiu com circuitos *LSI* como *TTLs*, onde um componente de uma placa continha apenas algumas portas lógicas equivalentes. Além do posicionamento do componente, era necessário o assinalamento das portas lógicas do circuito para aquelas disponíveis nos primeiros. Além da utilização de técnicas de posicionamento para colocar as portas nos componentes, também a troca de instâncias de portas do circuito nas portas físicas dos componentes era necessária.

O mesmo problema toma nova forma e importância nos *EPLDs* e *FPGAs*. Neste caso temos blocos lógicos equivalentes já posicionados dentro de um componente, mas em uma escala muito maior de integração. Assim, todo o posicionamento do circuito toma a forma de assinalamento de portas e de conexões. A vantagem está em que este assinalamento é um caso restrito e mais simples de posicionamento, para o qual os mesmo algoritmos são usados.

Par a solução de problemas de assinalamentos de pinos equivalentes em um ambiente *general cell*, foram propostos os algoritmos por mapeamento em círculos concêntricos, por assinalamento topológico, e pelo método de nove zonas ([Pre88] e [She93]). Entretanto, no caso de assinalamento de pinos para roteamento de um canal, temos dois problemas distintos: assinalamento de pinos equivalentes, e assinalamento de pinos com deslocamento de posição, ou pinos móveis.

Para o assinalamento ou troca de terminais equivalentes em roteamento de canais, o problema é localizado, e não genérico. Supondo que as conexões das redes do circuito já tenham sido atribuídas a determinados canais, a troca de terminais pode ser avaliada apenas considerando as conexões destes terminais em cada um dos canais adjacentes. Um algoritmo que trata deste caso é apresentado em [Joh97] págs. 39-43. Outro trabalho que aborda este assinalamento localizado, considerando também desempenho, é encontrado em [Her95].

O problema dos pinos móveis é interessante para realização do posicionamento absoluto, embora não seja suficiente. Após um posicionamento relativo, ou quando ainda se tem células com posições de terminais flexíveis, há liberdade para movimentá-los um pouco em ambos os sentidos, desde que respeitada sua ordem. Assim, estes movimentos podem reduzir bastante a altura do canal. O assinalamento de terminais móveis é abordado em [Gop83] e [Yan91]. Se o movimento possível é causado pela flexibilidade das próprias instâncias de células, deve-se considerar ainda as restrições existentes entre os pinos da mesma instância, o que não é modelado nos trabalhos citados.

5.5 Algoritmos de Roteamento

Os problemas de particionamento, posicionamento e assinalamento, já vistos, são problemas bem mais genéricos do que os de roteamento, tendo uma quantidade menor de variantes, versões, ou restrições. Mesmo naqueles casos, uma razoável quantidade de abordagens diferentes são utilizadas. O roteamento envolve então um universo a parte, composto por diferentes problemas, modelos, e algoritmos para cada problema.

Em vários casos pode-se ter algoritmos de roteamento mais genéricos. Este é o caso de algoritmos de procura de caminhos em grafos, de formação de árvores de menor custo, ou modelagens geométricas, como traçar um conjunto de linhas em um plano, de um ponto origem até um ponto destino, sem que elas interseccionem objetos ou outras linhas já posicionadas neste plano.

Os problemas característicos de roteamento, e os algoritmos usados para solucionar tanto estes problemas específicos, quanto os genéricos, são abordados na segunda parte deste trabalho.

6. Síntese Física em Tecnologias Sub-micrônicas

Observa-se que a incorporação de etapas de síntese de alto nível para projetos *VLSI* é um processo efetivo em sistemas de *CAD* práticos atuais. Entretanto, esta “migração” para etapas em níveis de sistema não significa que as etapas inferiores, de síntese lógica ou física, estejam solucionadas da melhor forma. Ao contrário, em projetos cada vez mais complexos, e principalmente com o surgimento das tecnologias de fabricação submicrônicas, há grande necessidade de se estimar, garantir ou otimizar diversos aspectos da síntese física de forma mais completa.

6.1 *Performance Driven Synthesis*

As características das tecnologias de fabricação evoluem e apresentam sensibilidade diferente a determinados aspectos. Em projetos de grande complexidade percebe-se atualmente que é necessário estimar e tratar devidamente alguns aspectos nas estruturas físicas, no leiaute, os quais não estavam sendo considerados pelas metodologias já conhecidas. A comunidade científica tem explorado recentemente o tema *Performance Driven Layout* ou *Performance Driven Physical Synthesis* justamente buscando a geração do leiaute dos circuitos de formas mais eficiente.

Normalmente parte dos esforços nesta área ainda é feito dentro dos limites das mesmas metodologias tradicionais, considerando novos critérios ou possibilidades de otimização, o que certamente já tem contribuído de modo fundamental para melhorar os resultados. Alguns dos aspectos da síntese física estão em foco segundo o prisma de *performance-driven*, quando são geralmente associados a um caso particular, considerando-se, por exemplo, a modelagem de problemas de interferência (ou *crosstalk*) [Gao94] [Kir94].

Entretanto, tornou-se evidente que os sistemas de síntese física (estilo de leiaute, posicionamento, roteamento) precisam evoluir muito para suportar as projetos atuais e futuros. Circuitos muito mais complexos estão sendo projetados a partir de níveis altos de abstração (sistemas), para os quais se tem fortes restrições de desempenho, como pequeno atraso e baixo consumo de potência, fatores que demandam uma estrutura de geração de leiaute e roteamento bem mais otimizada do que a disponível em sistemas convencionais do tipo *Standard Cell*.

6.2 Mudança na importância das conexões

A maior preocupação atual nesta área é provocada pela mudança na importância do roteamento para o atraso de um circuito em comparação com a importância do atraso das células [Pap95]. Em tecnologias mais antigas, o atraso do circuito era praticamente definido pelo atraso de suas células, ou componentes ativos, mesmo porque a complexidade dos circuitos era menor. Desta forma, uma metodologia do tipo *Standard Cell* era bastante adequada, pois utilizava células pré-caracterizadas, capazes de garantir um *fanout* (carga) adequado ao tamanho daqueles projetos, permitindo uma boa estimativa de desempenho. Otimizações no roteamento tinham então como objetivo fundamental baratear o custo através da redução da área do circuito.

Entretanto, para as tecnologias e projetos atuais, temos que a influência do atraso das portas no atraso total do circuito passa a ser menor do que a influência do roteamento. Além disto, o próprio projeto dos elementos ativos (células) precisa considerar impreterivelmente as diferenças dos elementos passivos, que são os caminhos de roteamento que devem conduzir os sinais. Além disto, com o aumento da frequência de operação e redução no tamanho dos transistores, temos conexões maiores que o comprimento de onda do sinal que transmitem.

Desta forma, vemos que é necessário não somente novos algoritmos e estruturas de roteamento muito eficientes para otimização flexível (dirigida aos problemas certos, como caminhos críticos no circuito), mas também viabilizar a geração adequada dos elementos ativos do circuito, não limitando-se a bibliotecas de células pré-caracterizadas. O uso de bibliotecas com várias versões de células adequadas a diferentes *fanouts* pode cobrir parcialmente a solução do problema [Sap95], mas provavelmente isto tenderá com o tempo a ser mais caro (contabilizando projeto e manutenção) do que a tecnologia embutida em um gerador de células.

6.3 Mudanças na metodologia de síntese

O professor Jason Cong, co-diretor do Laboratório de CAD para VLSI da *University of Califórnia at Los Angeles (UCLA)*, por ocasião do tutorial sobre “*Timing Driven Design for VLSI Layout*” no IFIP VLSI’95, ressaltou a mudança na importância do roteamento, a partir do que se busca uma metodologia na qual as conexões devam ser planejadas antes das células e transistores do circuito, o que também é colocado em [Pap95]. Esta tendência foi comparada com a revolução provocada pela orientação a objetos na parte de *software*.

Os primeiros programas eram planejados segundo um paradigma totalmente procedural, sendo depois montadas as estruturas de dados que fossem necessárias. A partir do paradigma de orientação à objetos, primeiro considera-se a modelagem dos dados, suas características, forma de armazenamento e acesso, para somente então escrever o programa em si. Esta mesma mudança é necessária para o leiaute de circuitos VLSI. Até agora, a maioria dos circuitos é projetada a partir de bibliotecas com células pré-caracterizadas, e a montagem do circuito é feita com um roteamento posterior em uma estrutura fixa para simplicidade, não permitindo grande flexibilidade ou otimização. Mas devido à importância que tem o roteamento, contribuindo com a maior parcela do atraso em um projeto atual, deve-se passar a modelá-lo ou sintetizá-lo prioritariamente, antes de considerar-se as células, tamanho dos transistores, etc...

A síntese de alto nível, ou chamada de síntese comportamental, também desempenha um papel essencial para *performance driven layouts*, visto que pode-se através de opções controladas desde etapas iniciais, estimar e verificar diferentes alternativas do espaço de projeto, optando-se pelas que mais atendem as necessidades existentes. Mas mesmo neste ponto o assunto “roteamento” não se torna menos presente. Pelo contrário, renova-se a necessidade de estimar e posteriormente projetar corretamente as características de roteamento. Como defende o professor Michael J. S. Smith (Compass e Universidade do Hawaii), a síntese de alto nível precisa realmente considerar todos os aspectos de “alto nível”, não somente aqueles comportamentais, mas também os estruturais e geométricos. Assim, os três eixos do diagrama Y de Gajski

[Gaj83] devem ser considerados desde o nível mais alto de abstração para permitir que as estimativas e decisões iniciais do processo tenham uma base mais exata com o resultado físico completo.

O prof. Smith propunha, de forma mais prática, o relacionamento entre a síntese de alto nível e a etapa de *floorplanning* do circuito, de tal forma a considerar, estimar e projetar gradualmente o roteamento do circuito, como também é salientado em [Pap95], o que tem sido enfatizado em trabalhos de outros autores e instituições. Na prática, nas metodologias tradicionais, um processo pouco flexível de divisão e conquista é normalmente empregado. Se o circuito foi particionado em blocos e estes blocos separados e colocados a cargo de diferentes equipes para a síntese em diferentes ferramentas, dificilmente as otimizações entre blocos são atingidas. Percebe-se a necessidade de modelos melhores para tratar com todo o processo de uma forma mais flexível. A indústria e a comunidade acadêmica estão desenvolvendo atualmente diversos trabalhos neste sentido. Já há ferramentas comerciais para planejamento topológico que consideram as características de interconectividade do circuito, seus limites, análise de temporização, dissipação de potência, e assim por diante, desde os níveis mais altos de abstração.

PARTE 2 - Hierarquia de Problemas de Roteamento

7. Roteamento

O roteamento é a etapa de síntese física responsável pela definição das rotas e dos materiais para conexão de pinos de elementos que devem ter o mesmo potencial elétrico. Os elementos podem ser desde transistores isolados, pequenas células lógicas, macro células funcionais, grandes blocos já projetados, até componentes inteiros já empacotados. Os materiais disponíveis para roteamento são geralmente um conjunto de camadas metálicas, e furos nas camadas de isolamento para pôr em contato trechos de camadas metálicas verticalmente adjacentes. As rotas serão estabelecidas com estas camadas nos espaços permitidos. As conexões não podem se tocar quando não pertencem ao mesmo conjunto equipotencial de terminais, e portanto, são obstáculos naturais entre si. Obstáculos externos são impostos também pela forma de implementação ou estilo de leiaute, tanto como limitação na área, como com a presença de objetos dentro dela.

7.1 Histórico e Abrangência

O roteamento surgiu inicialmente aplicado à confecção de placas (*PCB* ou *PWB*), quando a escala de integração de componentes ainda era pequena. Neste âmbito temos desde as primeiras placas de face simples, até tecnologias atuais com mais de uma dezena de camadas de interconexão.

Com o aumento na escala de integração, o roteamento de circuitos integrados passou a ter maior atenção. Neste caso, as formas de implementação e estilos de leiaute condicionam fortemente o tipo de roteamento, e novos problemas foram caracterizados e desenvolvidos.

Considerando sistemas atuais que envolvem dezenas de milhões de transistores, há claramente uma grande hierarquia tanto em nível lógico como em nível físico, com diferentes tecnologias de fabricação das conexões. Para cada parte desta hierarquia há um conjunto de problemas característicos de roteamento, alguns equivalentes, outros diferenciados. Em geral busca-se trabalhar não somente restrito a um destes níveis, de forma a obter um melhor resultado global. Entretanto, isto é dificilmente obtido, dada a existência de barreiras tecnológicas, ou mesmo devido a incapacidade de tratar problemas de otimização muito grandes.

Um tipo de componente que tem importância crescente são os programáveis em campo (*FPGAs*), seja pela facilidade de prototipação, seja pela possibilidade de fazer sistemas reconfiguráveis ou tolerantes a falhas. Neste caso, as conexões já estão fabricadas, e a tarefa de roteamento se traduz na sua programação, ou escolha de quais conexões podem e serão usadas para as redes lógicas do circuito. Este problema tende mais a ser modelado como assinalamento, como *network flows*, por exemplo. É semelhante ao que ocorre nos demais circuitos *VLSI* do ponto de vista global, mas bastante diferente do ponto de vista detalhado. Já no projeto do componente, além da análise de roteabilidade, como em outros circuitos pré fabricados, devemos projetar o número e o tamanho das conexões, bem como sua programabilidade, sendo um problema característico.

Exclui-se do espectro outros problemas de roteamento usados fora da microeletrônica, embora alguns algoritmos possam ser compartilhados. Um exemplo deste tipo de problema é o roteamento de mensagens em uma rede de comunicação. A grande diferença é a dimensão temporal, já que as rotas são compartilhadas no tempo.

7.2 Classificação e Terminologia

Os problemas e os algoritmos de roteamento podem ser classificados por diversos critérios. Apresenta-se algumas das classificações importantes, bem como termos notáveis empregados historicamente.

7.2.1 Classificação de roteamento por objetivos

Em primeiro lugar, visando distinguir diferentes aplicações do termo roteamento, se propõe a seguinte extensão à classificação de [Pre88]. O roteamento pode ser: detalhado, global, funcionalmente especializado, ou tecnologicamente especializado.

O **roteamento detalhado**, cuja definição é intuitiva, é a tarefa de especificar completa e detalhadamente as rotas de cada conexão, seus materiais, furos, posições e dimensões exatas. Sendo um problema muito complexo para o circuito inteiro, é normalmente tratado por partes. Assim, o roteamento detalhado completo de um circuito é formado pela simples união das soluções de um grande conjunto de roteamentos detalhados restritos à pequenas porções do circuito.

O **roteamento global** é a etapa responsável por dividir o problema de roteamento de todo o circuito para um conjunto completamente especificado de roteamentos detalhados suficientemente pequenos para que sejam tratáveis. Esta divisão certamente tira proveito da localidade de conexões e espaços. Entretanto, em todos os casos, o roteamento global necessita planejar as conexões mais longas decompondo-as em pequenas conexões locais a cada roteamento detalhado. São tarefas também características do roteamento global a definição de espaços para que as conexões sejam acomodadas, por exemplo, pela inclusão de estruturas de passagem, e o gerenciamento da forma de cada rede de múltiplos terminais, que será uma árvore dentro do grafo de possibilidades de conexão entre terminais.

O **roteamento funcionalmente especializado** é aquele necessário para conexões com características especiais, como alimentação, relógio, barramento, ou sinais de entrada e saída. A alimentação requer larguras apropriadas à corrente gerada pelo consumo do circuito. O sinal de relógio deve ser provido com a mínima defasagem à todas as partes do circuito, e para tanto seu roteamento deve ter uma estrutura topológica especial, além de larguras ajustadas de acordo com a carga. O síntese das conexões de barramentos e de entrada e saída podem ter também características próprias, sendo estas na maioria das vezes exploradas pela regularidade estrutural ou física.

O **roteamento tecnologicamente especializado** ocorre em situações específicas da tecnologia de fabricação de equipamentos eletrônicos. Este é o caso do roteamento de pinos para diferentes empacotamentos. Este tipo de roteamento, por exemplo, além da geometria característica, não faz parte do processo de projeto de um circuito particular,

mas sim do projeto do próprio empacotamento. Ele ainda pode incluir outras necessidades avançadas, modelagem de indutância, acoplamento eletromagnético, etc.

7.2.2 Classificações de roteamento quanto ao espaço

O número de conexões em um circuito cresce aproximadamente de acordo com o número de componentes a conectar. Entretanto, com o maior número de componentes, se tem maior distância entre eles, e conseqüentemente, maior tamanho das conexões. Assim, a área necessária para o roteamento cresce mais se comparada à área necessária para as células. Para reduzir este problema, com o passar do tempo, as tecnologias de fabricação passam a oferecer mais camadas de interconexão, para que o roteamento possa ser realizado por sobre a área dos elementos primitivos do circuito.

Já que os transistores do circuito são formados no substrato, eles não podem ser sobrepostos de nenhuma forma, e representam o limite mínimo de área para um dado circuito (definido em nível elétrico). Assim, temos que a tecnologia de fabricação, junto ao suporte de sistemas de *EDA*, devem viabilizar que todas as conexões sejam realizadas sem espaços adicionais, por sobre os próprios transistores, gerando circuitos com a mínima área, [She95].

Quando há espaços dedicados somente para as conexões entre as células, estes espaços são denominados normalmente de canais de roteamento, ou *switch boxes*, como caracterizado adiante. Se não há canais reservados, diz-se que o roteamento é sem canais (*channelless routing*). A ausência de canais pode ser só aparente. Isto se dá quando, feitas as conexões por sobre uma área, os transistores desta área são inutilizados.

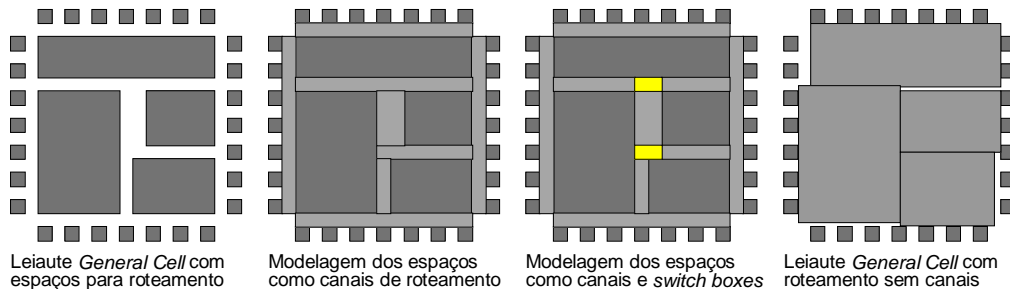


FIGURA 7.1 - Realização do roteamento segundo diferentes modelos

O termo roteamento sobre células, ou *OTC (over-the-cell)*, aparece pela primeira vez em [Deu80], e designa a realização de algumas conexões por sobre a área das células, quando há espaço e camadas disponíveis para isto (fig. 7.2). Diversos trabalhos abordam diferentes modelos para *OTC*, em especial publicações de Cong, Preas, Sherwani, entre outros. Referências à estes trabalhos podem ser encontradas em [Joh95].

Embora alguns modelos tenham sido associados fortemente à existência de um determinado número de camadas e a determinadas limitações de espaço, muitos deles representam abstrações ou problemas formais básicos. Assim, o modelo de canal, ou um roteamento planar, continuam sendo de grande importância mesmo em abordagens onde todo o roteamento é realizado sobre as células, e onde há várias camadas disponíveis para tal. Problemas deste tipo são em geral os chamados **problemas de roteamento restritos**, os quais são bem característicos. Os problemas mais importantes desta classe

serão mencionados adiante.

FIGURA 7.2 - Modelo físico para roteamento *Over-the-Cell*

No caso mais geral em que o roteamento é realizado sem canais por sobre células ou blocos utilizados do circuito, diz-se que é um roteamento de área (*area routing*). Dois fatores caracterizam este tipo de problema: a inexistência de limites parciais nesta área; e a presença de obstáculos, ou restrições, nas camadas que precisam ser usadas para o roteamento. Estes casos podem exigir, ao menos em determinadas fases, a utilização dos chamados **algoritmos de roteamento genéricos**.

O roteamento também pode ser feito por **justaposição** simples, como a alimentação em *standard cells*, ou vários sinais em partes operativas do tipo *bit-slice*, nas linhas de acesso aos *bits* de grandes blocos de memória, ou ainda em estruturas de processamento regulares, como *arrays* sistólicos.

7.2.3 Classificações de roteamento quanto à modelagem do espaço

Na maioria dos casos, para os sinais ordinários do circuito, todas as conexões tem a mesma largura e os terminais estão alinhados. Neste caso o roteamento pode ser baseado em uma **grade**, onde abstrai-se o tamanho físico exato, e considera-se apenas as unidades de passo da grade. O roteamento será também simbólico, sobre uma grade virtual, se o restante do leiaute também estiver sendo projetado simbolicamente, em diagramas de barras. Usa-se o termo roteamento topológico para expressar uma abordagem que captura um pouco mais a semântica do problema, independente do leiaute exato, de grade, ou mesmo de um número específico de camadas de interconexão. Em uma análise topológica, procura-se, por exemplo, avaliar o número de cruzamentos que são necessários para a solução, ou como minimizá-los.

Para redes críticas, e em face à crescente importância em otimizar as conexões em seu atraso e dissipação de potência, há a necessidade de modelos e algoritmos para roteamento com larguras de conexões variáveis. Maiores larguras de trilhas devem ser associadas a determinados trechos de conexões para reduzir a resistência, permitindo maior corrente, de acordo com o exigido pelo restante da rede a partir do ponto em questão [Con95b]. Este problema exige algoritmos específicos para otimizar e definir as

larguras adequadas de cada trecho, e algoritmos de roteamento detalhado capazes de acomodar tais conexões na área disponível. O casamento entre conexões em diferentes camadas já é um problema em função de diferentes tamanhos e espaçamentos mínimos para cada camada, nas tecnologias atuais. Se consideramos a realização de conexões com larguras arbitrárias, mesmo um planejamento prévio deste casamento torna-se impraticável.

O uso de conexões em ângulos diferentes de 90 graus também pode ser muito importante. Somente no roteamento de canal, por exemplo, o uso de conexões inclinadas em 45 graus pode reduzir bastante a altura deste. Dentre as dificuldades e razões para que isto não seja comumente utilizado, encontramos questões tecnológicas, de descrição e verificação, e mesmo algorítmicas, já que surgem problemas que, se não são mais difíceis, ao menos são novos. Conexões inclinadas foram bastante usadas em leiautes de células desenhadas a mão, como podemos comprovar em fotografias de microprocessadores, e ainda o são em placas de circuito impresso. Uma proposta semelhante, bastante interessante, é a de fazer leiautes somente com ângulos retos, mas cuja forma global seja em diamante (rombo, ou losango) [Nai86]. Isto reduz a distância máxima dentro do circuito de 2λ para $\sqrt{2}\lambda$, onde λ é o lado do mesmo circuito, com área aproximada λ^2 .

Já para o roteamento global, pode-se modelar o problema como um grafo ou como uma grade regular. No caso de modelagem por grafo, os vértices podem representar tanto os canais e *switchboxes*, quanto as intersecções entre estes. Além destas formas existem outros tipos de grafos associados aos problemas, como é mostrado em [She93]. Em ambos os modelos, os objetivos principais são: encontrar o caminho mais curto para cada rede; reduzir o tamanho do circuito como um todo; melhorar a roteabilidade [Che94]; balancear o congestionamento das regiões.

7.2.4 Classificação dos algoritmos de roteamento quanto ao processamento

Podemos classificar os algoritmos de roteamento pelo modo com que tratam a solução no que diz respeito ao relacionamento entre as diversas redes que precisam ser conectadas no mesmo espaço. Assim, os algoritmos podem ser:

- incrementais ou seqüenciais - aqueles que realizam as conexões uma a uma até completar todas ou não ser mais possível efetuá-las;
- integrais ou paralelos - são os que consideram ao mesmo tempo todas as conexões necessárias e buscam uma solução global ao problema;
- refinadores ou iterativos - partem de uma solução inicial simples ou parcial e modificam as soluções encontradas até ser obtida a melhor ou alguma satisfatória;

Os primeiros dois são mutuamente exclusivos, enquanto que o terceiro pode ser utilizado em conjunto com os demais, pois se caracteriza por operações como “desfaz e refaz”, o que consiste em retornos a etapas já realizadas. A classe de algoritmos seqüenciais apresenta entre outros o problema da ordenação. Isto ocorre porque tomando cada conexão sem considerar as demais, sua realização pode impedir outras se esta bloquear trechos do único caminho possível para conexões posteriores. Isto significa que mesmo que haja uma combinação de caminhos que permita realizar todas as conexões de um circuito, num algoritmo seqüencial esta solução pode não ser

encontrada em consequência da ordem em que foram realizadas as conexões. Como resumo a solução de cada rede não deve visar somente o mínimo local, pois isto acaba prejudicando o mínimo global, e neste sentido os algoritmos paralelos e iterativos são mais evoluídos.

7.3 Modelos que Condicionam o Roteamento

O roteamento de um circuito depende principalmente de dois fatores: do modelo de roteamento, e dos algoritmos de roteamento. Em uma área restrita, o modelo de roteamento definirá exatamente um problema de roteamento, o qual pode ser resolvido por diferentes algoritmos. O modelo de roteamento em nível de circuito, entretanto, representa todas as restrições e características dos recursos de roteamento disponíveis no circuito. Está, portanto, intimamente relacionado com os modelos de blocos, células e de terminais usados no circuito, de acordo com o estilo de leiaute. Adota-se aqui o termo estrutura de roteamento para designar este modelo em todo o circuito e os fatores que nele influem.

Um modelo de roteamento define o seguinte:

- A forma das regiões de roteamento em cada camada;
- O número de camadas de metal para conexões;
- A posição dos terminais;
- Presença de terminais equipotenciais, ou estruturas de passagem;
- Restrições para a colocação de furos de contato (vias);

Em [She95], no cap. 4, é salientada a importância da escolha dos modelos de roteamento para minimizar a área total do circuito, e então são analisados os parâmetros para o roteamento, principalmente quanto à posição dos terminais e quanto aos modelos de células para *standard cells*. A classificação apresentada para a localização dos terminais, entretanto, é bastante redundante em relação àquela apresentada para os modelos de células. Assim, este trabalho tece considerações mais genéricas quanto os terminais.

7.3.1 Posicionamento de terminais

O posicionamento dos terminais em um modelo de roteamento é definido pelas seguintes características:

- Camada em que estão disponíveis: polissilício, metal 1, ou metal 2;
- Localização na área de roteamento: bordas, centro, ou regiões;
- Alinhamento: em uma linha, em múltiplas linhas, ou aleatoriamente dispostos;
- Forma dos terminais: ponto, barra, ou conjunto arbitrário de pontos;

Esta caracterização procura ser bastante abrangente. A posição dos terminais é o fator mais importante para que se possa definir um problema de roteamento e desenvolver um algoritmo que o solucione eficientemente. Existem problemas e algoritmos específicos que trabalham com as disposições mais conhecidas. Mas o número de possibilidades diferentes é muito grande, e, na prática, algoritmos dedicados podem ser necessários para solucionar um problema particular que ocorre em

determinada tecnologia e estilo de leiaute.

Evidentemente é possível formalizar um problema completamente genérico, mas certamente nenhum algoritmo para resolvê-lo poderá encontrar uma solução tão eficiente como quando conhecemos previamente a disposição dos terminais. É semelhante ao que ocorre com os problemas de grafos. Quando os grafos com os quais trabalhamos pertencem a alguma das categorias de grafos perfeitos, podemos ter a solução ótima em tempo polinomial, mesmo quando o problema é NP-difícil para grafos em geral.

Na prática, a disposição dos terminais respeita algumas regras e limites impostas diretamente pelo modelo de células usado. Este modelo é em sua quase totalidade um modelo semelhante a *standard cells*, bastante adequado a qualquer estilo de leiaute baseado em bandas (*row-based*).

7.3.2 Modelos de células

A figura 7.3 mostra a classificação empregada para o modelo de células, de acordo com a posição dos terminais, conforme [She95], pág. 110, cuja nomenclatura é bastante aceita. As células são classificadas em primeiro lugar quanto à forma de seus terminais, entre células com terminais em forma de ponto, ou de barra, pelo que exclui-se demais formas ou diferentes orientações possíveis. Os terminais em forma de barra no entanto, são assim considerados apenas quando os intervalos de início ou fim em que estão presentes são variáveis. Quando todos terminais tem forma de barra vertical mas estão presentes no mesmo intervalo vertical, então o modelo usado é o mesmo que para terminais em ponto. Por esta razão a forma dos terminais é considerada como um fator de alinhamento. A localização dos terminais pode ser na borda, no centro ou no meio. Eles são considerados “ho centro” quando há apenas uma linha determinais, e “ho meio” quando estão dispostos em duas linhas.

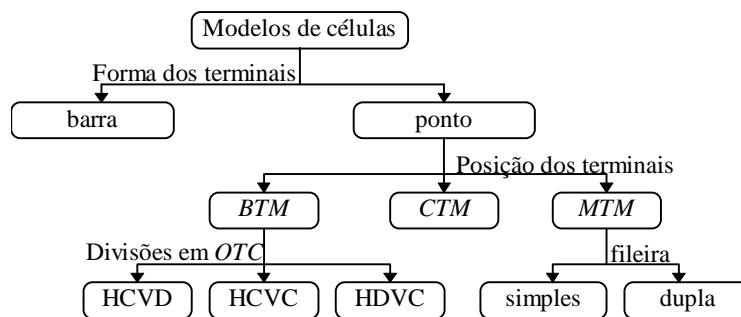


FIGURA 7.3 - Modelos de células segundo [She95]

Assim, os modelos básicos de células seguem a seguinte nomenclatura:

- *Target Based*: Terminais em forma de barras de altura variável;
- *Terminal Based*: Terminais em forma de ponto;
- *BTM*: *Boundary Terminal Model* - terminais nas bordas da célula;
- *CTM*: *Center Terminal Model* - terminais em uma linha central;
- *MTM*: *Middle Terminal Model* - terminais dispostos em duas linhas internas;

O modelo *BTM* é considerado como o modelo tradicional, por ter sua origem desde os tempos em que a tecnologia só oferecia uma camada metálica. Os terminais estão disponíveis nas bordas inferior e superior da célula, de forma equipotencial. O uso deste modelo em tecnologias com mais camadas metálicas foi explorado com as técnicas de *OTC* que selecionam um conjunto de conexões para serem feitas através de um roteamento planar na área disponível. De acordo com a presença de divisões para a camada usada para *OTC* sobre a célula, Cong et al. apresentaram três subclasses de células *BTM* [Con90]:

- HCVD: Horizontalmente Conectada, Verticalmente Dividida;
- HCVC: Horizontalmente Conectada, Verticalmente Conectada;
- HDVC: Horizontalmente Dividida, Verticalmente Conectada;

Cada um destes modelos apresenta as versões para tecnologias com duas ou três camadas metálicas, e para a possibilidade ou não de colocação de vias sobre a área da célula. Ainda, de acordo com cada modelo, posições e camadas típicas são usadas para a alimentação, e mesmo para os terminais. Assim, há algumas restrições ou alternativas adicionais, como pode ser visto em [She95], e em mais detalhes nos artigos sobre cada modelo. Em cada modelo possível, há um conjunto diferentes de problemas de roteamento para solucionar. Geralmente eles serão compostos pela seleção de conjuntos de redes planares, assinalamento de posições com roteamento planar, e roteamento de canal, problemas estes que são abordados a seguir.

7.4 Problemas Clássicos de Roteamento

Esta seção apresenta os problemas de roteamento que são usados com mais frequência, ocorrendo em estilos *standard cell*, *gate arrays*, em placas de circuito, roteamento sobre células, *MCMs*, etc...

7.4.1 Canal de Roteamento

Um dos modelos mais empregados no roteamento de circuitos *VLSI* é o modelo de canal. Um canal é uma região retangular, tendo uma das dimensões geralmente bem maior do que a outra, sendo a maior chamada de comprimento, e a menor de altura. Neste problema temos um conjunto de terminais dispostos nas duas margens, ao longo do comprimento do canal, os quais devem ser conectados na sua área interna, tipicamente por duas camadas. A figura 7.4 mostra este modelo e sua nomenclatura. Este problema surgiu com células *BTM* em tecnologias de uma ou duas camadas metálicas, quando a área por sobre a célula já estava ocupada em ambas as camadas.

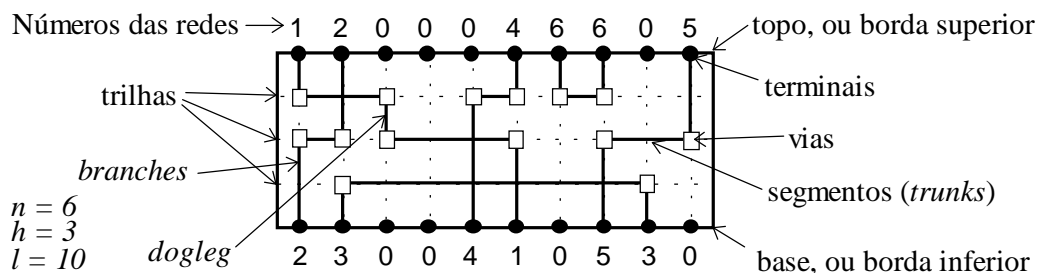


FIGURA 7.4 - Modelo e terminologia de roteamento de canal

Já que um canal tem altura típica de algumas unidades até algumas poucas dezenas de trilhas, e comprimento de algumas dezenas até milhares de trilhas, ele representa um problema de roteamento horizontal. Dado um conjunto de n redes. Se, em média, estas redes abrangem a metade do comprimento do canal, temos que a necessidade de roteamento horizontal é de $n * \lambda / 2$. Se assumimos que a metade das redes tem terminais em apenas um lado, a necessidade de roteamento vertical é de $n * h / 2$. Observe que se h é muito menor que λ , então a necessidade de roteamento vertical é muito menor do que a horizontal, embora a área disponível para conexões verticais, $h * \lambda$, seja exatamente igual àquela disponível para conexões horizontais quando associamos uma camada à direção horizontal e outra à vertical, como normalmente é feito. Observe que esta análise não é perfeita, pois se avaliarmos a necessidade horizontal e vertical da rede da fig. 7.5, vemos que um roteamento mínimo exige mais do que a mínima necessidade horizontal somada à mínima vertical. Esta diferença é justamente o espaço de trabalho que existe para aliviar o congestionamento em um sentido, através do uso de maiores conexões no outro. De fato, os algoritmos de roteamento de canal procuram analisar e gerenciar da melhor forma possível as conexões laterais, apenas modelando as restrições possíveis nas conexões verticais.

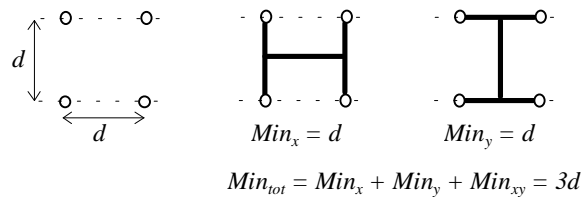


FIGURA 7.5 - Necessidades de roteamento em x, y, e mínimo global

A definição formal de um problema de canal é dada normalmente em uma de duas formas: simbólica, ou espacial. Na forma simbólica, temos dois conjuntos de terminais, T e B , respectivamente do topo e da base do canal, e funções que têm estes conjuntos como domínio e suas posições x e seus números de rede como imagens. Na forma espacial, temos dois vetores V_t e V_b de comprimento λ , onde cada posição tem um zero se nenhum terminal está presente, ou o número da rede que tem um terminal nesta posição. A primeira forma facilita a especificação formal do problema e de suas restrições, como por exemplo, definir a posição no n -ésimo terminal anterior. Já a forma espacial é mais apropriada para o armazenamento das informações e para o processamento de um algoritmo. Os demais problemas de roteamento restrito também podem ser especificados em uma destas duas formas. Cabe lembrar também que em muitos trabalhos considera-se uma rede apenas como um par de terminais, e redes multi-terminal devem ter sido previamente decompostas em conexões de dois terminais.

O modelo de roteamento no canal também pode ser classificado conforme o número de camadas e o uso que fazemos delas. Existem algoritmos para roteamento de canais em duas e três camadas. A atribuição de uma direção obrigatória para cada camada acontece no modelo de camadas reservadas. Neste caso, conforme a direção que atribuímos para as camadas, contando a partir das inferiores, temos os modelos VH ou HV para canais com duas camadas, ou VHV e HVH para canais com três camadas. Os modelos com camadas não reservadas são superiores em termos de altura mínima e

também em menor utilização de vias, mas requerem algoritmos mais complexos.

É freqüente a necessidade de se fazer conexões nas laterais do canal, mas poucas vezes isto é considerado nos algoritmos. Algumas outras extensões incluem canais com quebras, ou canais circulares (com 4 quebras), também chamados de anéis, canais com topo e base não retilíneas, entre outras. A altura do canal é fixa quando se trabalha com formas de implementação programáveis por algumas máscaras, onde o espaço existente para o roteamento já foi definido. Para projetos *full-custom*, no entanto, esta altura é variável, e então assume-se que sempre será possível resolver o problema, sendo desejada a solução que apresente a menor altura em trilhas. A existência de uma solução não é genericamente uma verdade nos modelos com camadas reservadas (como em quase todos os algoritmos). Pode-se ver na fig. 7.6 um exemplo trivial onde somente existe solução de esta obrigatoriedade for suprimida. Na prática, porém, assume-se que a distribuição de terminais e vazios nas bordas do canal é bem mais amena, e sempre existirá uma solução.

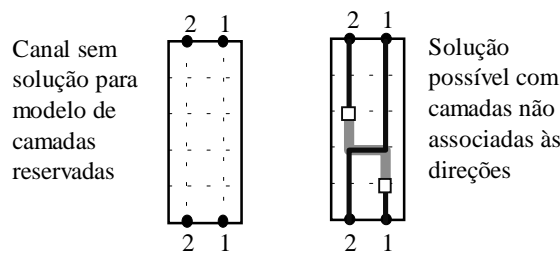


FIGURA 7.6 - Roteamento de canal nem sempre é possível

Em função da dificuldade de roteamento horizontal, o roteamento do canal é basicamente o problema de assinalar segmentos horizontais de redes à determinadas trilhas do canal. Segmentos verticais são usados para conectar segmentos horizontais da mesma rede em diferentes trilhas, ou com os terminais. Há dois tipos de restrições que devem ser satisfeitas em tal tarefa, as restrições horizontais e as restrições verticais.

Chama-se *HCG* o grafo de restrições horizontais, que é na verdade um grafo de intervalos dos segmentos de rede. Este grafo é importantíssimo para a análise e roteamento do canal. Em um modelo de duas camadas, segmentos diferentes não podem ocupar a mesma trilha, e então, o máximo *clique* em *HCG* define o limite inferior em número de trilhas para o canal. O grafo de restrições verticais *VCG* possui uma aresta entre dois segmentos se eles possuem conexão com terminal na mesma coluna, ou seja, se começam ou terminam no mesmo ponto.

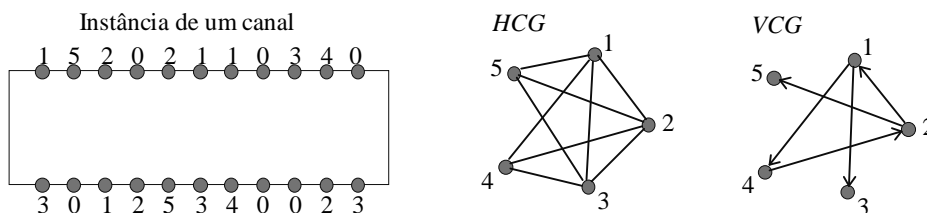


FIGURA 7.7 - Grafos de restrições associados a um canal

VCG é um grafo dirigido, onde o sentido das arestas representa a ordem em que as conexões devem estar para que os segmentos verticais não se interseccionem. Se houver

ciclos em *VCG* então alguns segmentos devem ser divididos através de *doglegs*, para que uma ordenação seja possível. As restrições verticais também impõem limitação na altura do canal. Se não forem usados *doglegs*, então o maior caminho em *VCG* limita a altura mínima do canal.

7.4.2 Caixa de conexões, ou *switch box*

Uma caixa de conexões, ou *switch box*, é um retângulo com largura e altura aproximadamente iguais, com terminais nos quatro lados. Este tipo de problema ocorre para conectar canais de roteamento perpendiculares, ou uma região entre quatro módulos, entre outras situações. Embora se possa pensar em um *switch box* como uma generalização de um canal, ele não apresenta a mesma característica horizontal, e também não possui altura variável. Enquanto se procura minimizar a altura de um canal, para um *switch box* procuramos avaliar sua roteabilidade, e encontrar apenas uma solução possível. Assim, ele é melhor caracterizado por um grafo circular.

7.4.3 Roteamento Planar

Roteamento planar é aquele possível de ser realizado em apenas uma camada, sem cruzamento de conexões. Uma série de modelos e algoritmos surgiram para este tipo de problema, principalmente em situações onde somente dispomos de uma única camada de roteamento. Os problemas de roteamento planar, entretanto, são de grande importância mesmo nas situações onde várias camadas estão disponíveis para o roteamento, por várias razões.

Considerando um roteamento em múltiplas camadas, se tem duas principais abordagens para tratá-lo. A primeira seria considerar um roteamento em três dimensões. Este problema, além de complexo, não ocorre com tanta frequência, seja pela limitação no número de camadas, seja pelas suas diferentes características tecnológicas. Há ainda um outro fator que é a hierarquia estrutural e posicional conferida ao circuito em todo o seu projeto. Desta forma, a classificação das conexões em locais, intra-blocos, ou inter-blocos, leva geralmente a reservar determinadas camadas para determinadas classes de conexão. A segunda abordagem, sobre onde justamente estas divisões recaem, é a divisão de um roteamento de várias camadas em um conjunto de roteamentos em apenas uma ou duas camadas.

Uma característica importante de todo roteamento planar é que ele não possui vias, pois não precisa de troca de camadas, exceto nos seus terminais. Um grande esforço é empregado na redução do número de vias no leiaute de um circuito. As vias possuem várias desvantagens:

- menor confiabilidade no processo de fabricação;
- maior atraso dos sinais pelo aumento das resistências;
- menor imunidade a ruído pela introdução de resistências em série;
- maior área devido à margem dos metais sobre os furos;

Assim, uma das melhores abordagens para um roteamento em múltiplas camadas é selecionar os subconjuntos planares de conexões para serem realizados cada um em uma única camada, reduzindo o número de furos. De fato, esta abordagem tem sido utilizada

para tratar roteamentos com várias camadas em *PCBs* [Dor81] [Tsu81], leiautes *VLSI*, e *MCMs* [She95] págs. 323-329. Alguns problemas particulares de roteamento planar que são notáveis por sua importância são vistos a seguir. Após, são também relacionados problemas de seleção de subconjuntos planares, aplicados a diferentes casos, conforme consta na bibliografia.

7.4.4 Roteamento em rio, ou *river routing*

A definição de um roteamento em rio é semelhante à de um roteamento de canal, mas com várias restrições. Cada rede deve ter apenas dois terminais, um no topo e outro na base, e a ordem em que as redes aparecem deve ser a mesma no topo e na base. Este problema pode ser resolvido com apenas uma camada se houver espaço suficiente separando o topo da base. Ele ocorre principalmente para conectar vários sinais entre dois módulos (um barramento, por exemplo), ou ainda para conectar terminais de células de uma parte central para as bordas, em modelos de células *CTM* ou *MTM*, por exemplo.

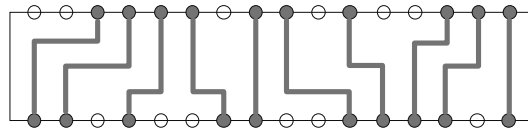


FIGURA 7.8 - Roteamento em rio (*river routing*)

7.4.5 Roteamento planar de uma caixa

Um problema um pouco mais genérico é o roteamento planar de uma caixa de conexões. Para avaliar a possibilidade de conexão planar, basta verificar o não cruzamento em um grafo circular. Resta o problema de verificar se há espaço suficiente, semelhante ao problema de roteamento em rio. Este problema pode ocorrer em uma região não retangular, por exemplo, entre vários módulos. Neste caso, a avaliação dos cruzamentos é a mesma, mas a do espaço é mais complexa.

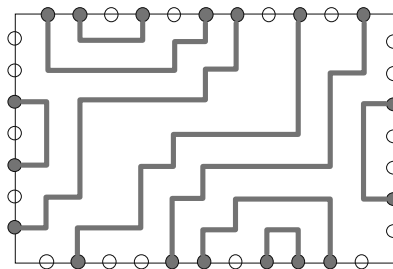


FIGURA 7.9 - Roteamento planar de uma caixa

7.4.6 *Single Row Routing Problem*, ou *SRRP*

O problema de *SRRP* é um dos mais estudados na literatura, mas cuja definição formal é apenas parcial em relação aos problemas que ocorrem na realidade. O problema é: dada uma linha com terminais igualmente espaçados sobre ela, cada terminal pertencente a uma rede de um conjunto de redes, encontrar um roteamento planar, em apenas uma camada. A rota de cada conexão deve ser tal que não contenha nenhum

movimento duplicado horizontalmente, mas pode conter múltiplos movimentos verticais. Entre cada dois terminais, assume-se que um número arbitrário de conexões pode ser passado. Com esta liberdade, está provado que para quaisquer terminais e redes, é sempre possível encontrar um roteamento válido. O problema passa a ser encontrar o roteamento com a mínima ocupação de trilhas nas regiões superior e inferior, com menor comprimento de conexões, com menor número de cruzamentos verticais, ou com melhor distribuição de cruzamentos verticais.

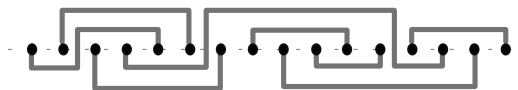


FIGURA 7.10 - *Single Row Routing Problem (SRRP)*

Este problema originou-se em placas *PCB* ou *PWB* com componentes em empacotamento *DIP*. Naqueles casos, os terminais formavam grandes linhas, e havia espaço para algumas conexões passarem entre dois terminais adjacentes. Genericamente falando, no entanto, o número de conexões que passam entre dois terminais é limitado, como acontece em um modelo baseado em grade para circuitos *VLSI*. Assim, a maioria dos trabalhos referentes a *SRRP* não apresenta uma solução sempre viável, e não modela as condições desta viabilidade.

7.4.7 Roteamento de subconjuntos planares tipo *SSPR*, *TSPR*, *ETSPR*

Há um conjunto de problemas de roteamento planar bem característicos, mas cuja aplicação envolve geralmente a realização de um determinado conjunto de redes de um universo total necessário. Estes problemas ocorrem em roteamento *OTC* e em *MCMs*. Assim, o objetivo não é encontrar um roteamento para um conjunto completo de redes, nem minimizar sua área, mas sim selecionar o máximo conjunto de redes de um universo, de forma que este possa ser completado no modelo dado. Problemas básicos deste tipo, explorados em [She95] págs. 146-151, são os seguintes:

SSPR: Single-Sided Planar Routing - O modelo corresponde a um problema de *SRRP* com a metade da área, e, portanto, não teria solução garantida para um conjunto completo de redes. A seleção de um conjunto de redes possível, no entanto, não é um problema difícil. Obter o máximo conjunto possível é a questão central.

TSPR: Two-Sided Planar Routing - Este problema é semelhante a dois problemas anteriores espelhados, compartilhando a área na qual as conexões são feitas.

ETSPR: Equipotential Two-Sided Planar Routing - Este problema é semelhante ao anterior, mas definido sobre conjuntos de barras equipotenciais que unem o topo à base. Assim, temos a redução no número de terminais possíveis, mas a liberdade de ter os terminais em ambos os extremos, além de estruturas de passagem naquelas colunas onde nenhum terminal está presente. Esta liberdade significa maior complexidade para solucionar o problema otimamente.

7.4.8 Problemas de assinalamento

Problemas de assinalamento ocorrem quando elementos de um conjunto estão competindo por elementos de outro conjunto, e deve-se achar uma função que associe cada para cada elemento do domínio, um elemento da imagem, univocamente. Alguns problemas restritos de roteamento se confundem com problemas de assinalamento, principalmente quando a identificação dos conjuntos é mais significativa do que as tarefas intermediária para associação de um par de elementos. Problemas desta classe são geralmente aplicados a casos particulares, mas existem alguns que ocorrem com mais frequência.

7.4.8.1 Assinalamento de terminais às bordas (*BTA*)

Um destes problemas é o assinalamento de terminais à borda das células, ou *BTA*, que ocorre em modelos de células *CTM* ou *MTM*. Como definido em [She95], pags. 96-99 e 148-149, é um problema semelhante a *river routing*, com a exceção de que os terminais em uma das bordas não têm sua posição definida. Podem ser então colocados em qualquer posição possível de um intervalo definido em função do número de trilhas e das posições de terminais vizinhos. O problema é encontrar um assinalamento que favoreça as conexões no canal adjacente. Basicamente cada terminal tem um sentido preferencial para o qual deve ser assinalado. Visto que dois terminais adjacentes podem ter sentidos preferenciais opostos, este problema é de otimização.

7.4.8.2 Assinalamento de pontos de cruzamento (*CPA*)

Este é um problema com menor número de restrições, e portanto, difícil, o qual é muito importante para roteamento de área. A abordagem de divisão e conquista pela separação em roteamento global e roteamento detalhado encontra uma certa dificuldade de ser aplicada a roteamento de área, já que neste caso não há uma divisão natural desta área. A aplicação de algoritmos genéricos é também ineficiente, porque geralmente estes são sequenciais, incapazes de contemplar a melhor solução global.

A abordagem de divisão e conquista empregada em roteamento de área, divide esta artificialmente em células globais, chamadas de *GRCs*. O roteamento global faz a especificação de quais *GRCs* serão usadas por cada conexão do circuito, de modo idêntico a outros modelos. Os terminais internos de cada *GRC* tem sua posição bem definida, mas aquelas conexões que devem cruzar uma *GRC* não têm uma posição definida na sua borda. O problema de *CPA* (*Cross Point Assignment*) é justamente assinalar estas posições para cada rede, de forma que os problemas de roteamento detalhados locais (uma generalização de *switch box*) sejam bem definidos.

7.4.9 Problemas de roteamento global

O Roteamento global, caracterizado anteriormente, encontra duas aplicações principais, com modelos um pouco distintos. A primeira é o roteamento global em um estilo de leiaute *general cell*, também referido como *building blocks*. A segunda é própria da síntese de um destes blocos em um estilo baseado em bandas, *standard cell*, por exemplo.

Em uma metodologia *general cell*, a etapa de planejamento topológico deverá definir o tamanho, forma e posicionamento dos blocos, além de quais são as regiões usadas para roteamento. O tamanho destas regiões pode ou não já ter sido definido, mas de qualquer forma uma ordenação dos canais e *switch boxes* deve ser providenciada de forma a minimizar a perda de área após o roteamento detalhado. Com uma ordenação sem dependência cíclica, o roteamento detalhado de cada região pode ser feito a partir dos que já estão, e definir as alturas mínimas dos canais.

O roteamento global deve modelar as áreas para roteamento, suas interseções, e encontrar os caminhos de todas as redes nestas áreas. A modelagem é feita com um grafo, em uma de duas formas possíveis. Pode-se modelar os canais como vértices e suas interseções como arestas, ou as interseções como vértices e os canais como arestas. Em ambos os casos se tem um grafo regular ou irregular sobre o qual deve-se encontrar os caminhos para as redes.

Já em circuitos baseados em bandas, normalmente não há canais verticais, e portanto não é possível construir um grafo desta forma. As conexões verticais devem ser feitas em uma de três formas: usando pontos específicos de passagem; inserindo espaços adicionais para conexões verticais; ou então livremente sobre as células. Este último modelo se assemelha mais ao roteamento de área, onde podemos dividir a área artificialmente para obter um grafo regular.

Nas duas primeiras formas de implementar conexões verticais, mais freqüentes, a modelagem mais natural é formar grafos específicos de cada rede, a partir das posições de seus terminais. A figura 7.11 mostra os terminais de uma rede em um circuito *standard cell*, o grafo correspondente, e duas árvores deste mesmo grafo que podem ser selecionadas para conectar esta rede. A seleção de árvores melhores para cada rede deve ser provida para melhorar o desempenho do circuito. Isto pode ser feito com um algoritmo que remove as arestas de maior custo que não desconectam o grafo. Em contrapartida, o congestionamento dos canais deve ser modelado, e evitadas as conexões nas regiões de maior densidade, encontrando uma solução global para o conjunto de redes.

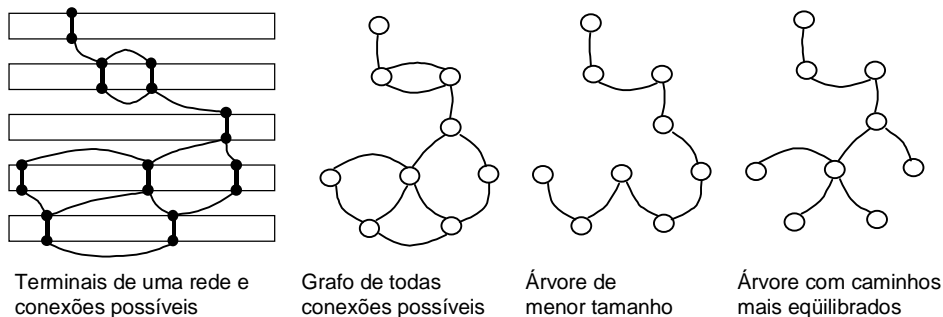


FIGURA 7.11 - Grafos para roteamento global em *Standard Cell*

Para as redes que necessitam a seleção ou inclusão de estruturas de passagem, o grafo precisa modelar estas estruturas em adição aos terminais da própria rede. Mesmo assim, os grafos ainda são particulares para cada rede. Neste caso, se tem um problema diferenciado de encontrar a árvore mínima. Este problema consiste em encontrar a árvore

mínima que cubra determinados nodos (aqueles que representam os terminais), mas não todos os nodos do grafo (fig. 7.12). Estas árvores são conhecidas como árvores de Steiner. Há diversos algoritmos usados para encontrar árvores de Steiner segundo diferentes critérios [Alp95] [Coy95], assim com trabalhos que endereçam o problema completo de roteamento global [Kao95] [Sad95],

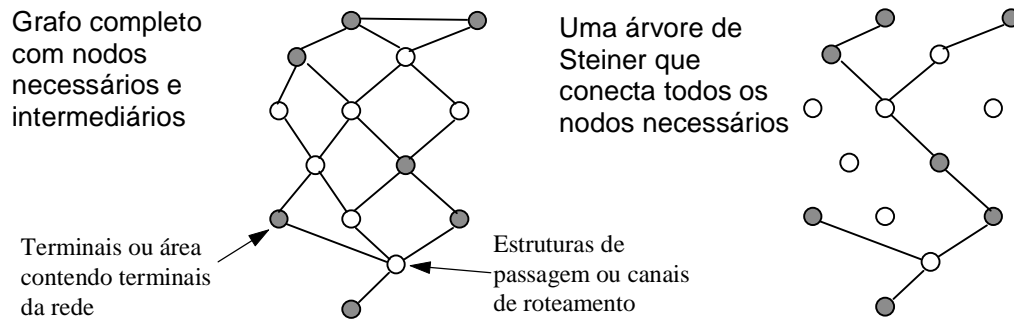


FIGURA 7.12 - O problema de árvores de Steiner

7.5 Algoritmos de Roteamento

Alguns algoritmos de roteamento representam uma abordagem característica, a qual pode ser aplicada a diversos dos problemas restritos. Há também os algoritmos ditos genéricos, os quais em princípio se pode aplicar a qualquer problema de roteamento. Mais comum, no entanto é usar algoritmos específicos para cada problema. Os algoritmos mais usados são citados a seguir, segundo cada classe de problemas de que se dispõe. Eles não são descritos completamente, mas apenas citados e acompanhados de comentários gerais sobre seu comportamento. O leitor deve se referir à bibliografia adequada para compreender ou comprovar as características de cada algoritmo.

7.5.1 Algoritmos de roteamento genéricos

Há dois tipos principais de algoritmos de roteamento genéricos, os baseados em grafos e os baseados em geometria. Cada classe apresenta um grande conjunto de variantes. Os algoritmos baseados em grafos são empregados para roteamento global ou para roteamento detalhado. Neste último caso, a área a ser roteada é modelada como um grafo regular, onde cada nodo representa uma interseção das trilhas da grade de roteamento. No caso de não haver uma grade natural, o algoritmo força a modelagem discreta como se ela existisse. Os algoritmos geométricos, ao contrário, podem modelar a área com coordenadas reais, e trabalhar com objetos e linhas de qualquer inclinação. É comum, no entanto, limitar o universo a linhas ortogonais em 90° , com posições também discretas, visto que a tecnologia de fabricação sempre possui uma resolução mínima.

7.5.1.1 Algoritmos genéricos baseados em caminhos em grafo

Os *maze routers* são algoritmos sequenciais que fazem a pesquisa pelo caminho mais curto entre dois pontos em uma grade, a qual poderia ser um grafo qualquer, pesquisa feita em largura. Cada célula desta grade pode estar livre ou bloqueada. O algoritmo básico é conhecido como algoritmo de Lee (1961), e consiste em marcar as

células não bloqueadas a partir da origem até o destino, com números de uma seqüência, de tal forma que seja possível identificar o menor caminho a partir do destino.

As principais vantagens deste algoritmo são:

- se existe um caminho entre a origem e o destino, ele é encontrado;
- o caminho encontrado é o mais curto;

As principais desvantagens são:

- tempo de processamento, quadrático em relação ao tamanho do circuito;
- memória requerida para armazenar o espaço em grade;
- ordem das conexões, a qual pode inviabilizar o roteamento, perdendo a garantia de encontrar uma solução para um conjunto de redes, mesmo se ela existe;

Para solucionar estes problemas uma série de otimizações foi proposta na literatura, alterando a característica de largura da pesquisa, associando custos às células, alterando a seqüência de números armazenada, ou alterando a ordem com que as conexões são feitas.

7.5.1.2 Algoritmos genéricos baseados em geometria

Dentre os baseados em geometria, destacam-se os algoritmos *line-probe* e *line-expansion* (ver em [Pre88] cap. 5). Ambos traçam linhas da origem para o destino. Se estas linhas interseccionam algum obstáculo, são escolhidos pontos de escape de onde novas linhas são traçadas. Diversas outras abordagens geométricas são também propostas. Em geral, elas apresentam menor consumo de memória e menor tempo de execução, mas pecam igualmente por serem seqüenciais. O uso de algoritmos seqüenciais encontra sua utilidade em avaliação de roteabilidade, geração de uma solução inicial de roteamento global, e também no roteamento das ultimas redes que não puderam ser roteadas por um algoritmo mais simples. Para problemas grandes de roteamento detalhado de todas as conexões, entretanto, deve-se evitar tanto quanto possível o seu uso.

7.5.1.3 O algoritmo hierárquico

Embora seja classificado com um algoritmo de roteamento de canal em [Pre88] e [She93], na verdade este é um algoritmo de roteamento genérico, tendo sido desenvolvido para o roteamento de *gate arrays* e utilizado com eficiência também em roteamento de canais e *switch box* como podemos verificar em [Bur83b]. A classificação como algoritmo de roteamento de canais se deve principalmente ao primeiro artigo publicado, [Bur83a], intitulado “*Hierarchical Channel Routing*” e que trata exclusivamente do roteamento de canais por este método.

Sua abordagem hierárquica na forma de divisão e conquista aliado ao modelo da área com capacidades de roteamento horizontal, vertical e possibilidades de mudança de camada o tornam especialmente adequado ao roteamento de *gate arrays* e de outros modelos uniformes, como roteamento de área. O roteamento de canais é na realidade um problema particular que também é melhor solucionado pelo algoritmo. Em [Bur83b] é

esclarecido que, embora o roteamento de um canal fixo possa ser feito diretamente pelo mesmo algoritmo, a possibilidade de variação da altura do canal (incremento conforme a necessidade) levou a significativas mudanças, produzindo a versão especializada apresentada no primeiro artigo, [Bur83a].

O algoritmo faz o roteamento sobre uma grade de duas camadas restringindo de forma absoluta a orientação destas, como em roteadores de canais tradicionais. Sua técnica se baseia na redução do problema de roteamento de uma grade ($m \times n$) para o roteamento em grade ($2 \times n$). Isto é feito dividindo-se a altura m em dois subgrupos de tamanho $m/2$ sucessivamente, solucionando o problema entre cada par de subgrupos a cada vez.

Para o problema de roteamento em grade $2 \times n$ são apresentadas duas abordagens. A primeira é baseada na solução exata do problema de roteamento em uma grade 2×2 usando programação linear e estendendo hierarquicamente estas soluções. A segunda baseia-se no roteamento mais tradicional sobre grade $2 \times n$ rede a rede de forma seqüencial. Isto é realizado porque o problema de árvores de Steiner com custos arbitrários pode ser solucionado em tempo linear quando a altura da grade é somente 2.

7.5.2 Algoritmos para roteamento de canal

Sendo um dos problemas mais freqüentes em leiaute *VLSI*, existem dezenas de algoritmos diferentes empregados para roteamento de canal. Os algoritmos básicos mais importantes são: *Left-Edge*, *Dogleg*, *YK*, *Greedy*, *YACR2*, Hierárquico.

O *Left-Edge (LEA)* é um algoritmo que roda em tempo linear em relação ao tamanho do canal, e gera sempre a solução ótima em alocação de trilhas, em ausência de restrições verticais. Caso hajam restrições verticais, pode-se combinar os nodos do *VCG* para as conexões assinaladas às mesmas trilhas. Se o grafo resultante não contiver nenhum ciclo, então é possível uma ordenação das trilhas para produzir um roteamento válido. Se, no entanto, *VCG* combinado tiver um ou mais ciclos, não é possível realizar o roteamento sem a inserção de *doglegs*. Os algoritmos conhecidos como *Dogleg* consistem em considerar a introdução de *doglegs* nas conexões, removendo a restrição de que cada segmento de rede esteja em apenas uma trilha por toda a sua extensão. Isto quebra o *VCG*, de forma que os nodos sobre os quais se aplicam as arestas (restrições) são replicados, e os ciclos podem ser removidos. Foi demonstrado que encontrar o melhor número e posição dos *doglegs* é NP-completo ([Pre88] pág. 174). O algoritmo Y-K (Yoshimura e Kuh, 1982, [She93] pág. 286) foi o primeiro a considerar a estrutura teórica dos grafos de restrições. Ele considerava ambos *HCG* e *VCG*, mas, sem o uso de *doglegs*, não contemplava a presença de ciclos em *VCG*.

O algoritmo *Greedy* é um algoritmo heurístico que também roda em tempo linear, apresentando um bom desempenho. Ele processa o roteamento da esquerda para a direita, conectando os terminais às trilhas, e estas entre si, em cinco passos, de acordo com a prioridade natural destas conexões. Ele busca conectar redes presentes em mais de uma trilha o tanto quanto possível, já que as conexões horizontais são limitantes em um canal. Como outras heurísticas, ele é sensível a alguns parâmetros de que necessita em sua implementação, e não garante completar o roteamento antes de chegar ao outro extremo do canal.

O algoritmo *YACR2* também se baseia no *LEA*, e faz um roteamento por este método, permitindo a violação dos conflitos verticais. Em uma etapa posterior, ele soluciona estes conflitos através do uso de padrões de conexões (*branches*) não apenas verticais. O uso de padrões simples com curvas em apenas uma camada minimiza os impasses, mas uma série de outros passos especialistas são usados para garantir uma solução, como a inclusão de trilhas e colunas adicionais. Um quadro comparativo destes algoritmos básicos pode ser observado em [Pre88] pág. 180, e [She93] pág. 301

Há uma série de algoritmos para o roteamento de canais com três camadas metálicas, em sua maioria, adaptações dos algoritmos anteriores. O canal pode ser construído nos modelos VHV ou HVH. No primeiro, temos duas camadas para roteamento vertical, o que elimina os conflitos verticais existentes. Assim, o roteamento sempre será possível com altura mínima correspondente ao *clique* de *HCG*, com um algoritmo *Left-Edge*. Entretanto, como já foi visto o canal é um problema de roteamento horizontal, e no modelo HVH pode-se encontrar um roteamento com altura mínima dada pela metade do *clique* de *HCG*, ou o máximo caminho em *VCG*, mas nem sempre com a garantia de encontrar esta solução.

O algoritmo Glitter de Chen e Kuh (1986) ([She93] pág. 291) permite o roteamento de canais não baseados em grade, com conexões de larguras variáveis. Este algoritmo foi estendido para o roteamento em três camadas por Chen em 1986 ([She95] pág. 88). Existem outros algoritmos de roteamento de canal específicos para canais com bordas não retilíneas, ou para conexões em 45° , para duas e três camadas de roteamento.

Normalmente emprega-se uma fase final de minimização de vias no roteamento de canal, onde se pode eliminar vias adjacentes. Outra etapa de pós processamento comum é a compactação do canal. A compactação é possível principalmente pela não presença de vias em todos os pontos possíveis da grade. Esta compactação é vertical, e pode ser feita de modo mais simples se comparada com compactação de leiautes em geral.

7.5.3 Algoritmos para roteamento de caixas de conexão

Os problemas de *switch box*, apesar de mais complexos do que o roteamento de um canal, são normalmente solucionados por algoritmos semelhantes, adaptados. Encontra-se na literatura o uso do algoritmos hierárquico [Bur83b], uma variação do algoritmo *Greedy* [Ham84] para o sistema de síntese de leiaute **Magic**, sistemas baseados em inteligência artificial, como o algoritmo *WEAVER* [Joo86], dentre outros. Em roteamento de área, o roteamento de cada *GRC* corresponderá a um *switch box* com terminais internos também.

7.5.4 Algoritmos para roteamento planar

Os algoritmos para roteamento planar de uma região fechada, retangular ou não, não são geralmente muito complexos. Para verificar a planaridade basta construir o grafo circular e verificar a ausência de arestas. Isto pode ser feito percorrendo os terminais pela ordem com que se encontram na borda, e usando uma pilha para colocá-los ou retirá-los. Este método é evidente para redes de dois terminais, mas pode ser generalizados para redes multi-terminal. Para verificar se há espaço suficiente para as conexões, basta traçá-las uma a uma o mais próximo da margem interna possível. Se

todas podem ser desenhadas, então o roteamento é possível, e foi feito. Deve-se posteriormente minimizar conexões que estão grandes por acompanharem a margem, quando há espaço lateral sobrando. Tão importante quanto o roteamento planar é a seleção de um conjunto planar que pode ser roteado no canal. Este problema é o mesmo de encontrar o máximo conjunto independente de vértices (*MIS*) no grafo circular, e pode ser resolvido em tempo $O(n^2)$ ([She95] págs. 49-51).

7.5.5 Algoritmos para roteamento em rio

O problema de roteamento em rio é bastante simples. A partir de uma das bordas, há conexões que devem ser feitas para a direita e outras para a esquerda. Podemos definir assim grupos de conexões adjacentes que precisam ser feitas no mesmo sentido. Visto que as conexões não se cruzam, não há qualquer conflito entre estes grupos, e o problema é decomposto na solução de cada grupo. Para cada grupo pode-se encontrar um roteamento válido com um algoritmo guloso que faz a conexão o mais perto da borda origem possível para a direita, ou o mais afastado possível para a esquerda, tomando todos os terminais na mesma borda. Verificar qual a mínima distância entre bordas para que o roteamento seja possível também é um problema simples, e pode ser avaliado através do congestionamento em linhas de corte verticais ([She95] pág. 74).

7.5.6 Algoritmos para roteamento de *SRRP*

Como já citado, este é um dos problemas mais estudados de roteamento detalhado, e para o qual há dezenas de algoritmos e avaliações heurísticas diferentes. Sua origem no roteamento de *PCBs* com componentes *DIP* abrangia o assinalamento de vias, posicionamento das colunas, assinalamento das camadas, o *SRRP* propriamente dito, e etapa posterior de remoção de vias. O diagrama de intervalos é uma representação do problema proposta por Kuh, Kashiwabara e Fujisawa em 1979 que teve um papel muito importante no desenvolvimento de vários algoritmos para *SRRP*. Outro conceito fundamental proposto pelos mesmos pesquisadores é o número de corte (*cut number*) de terminais e redes. O número de corte de um terminal é o número de conexões que passam sobre ele, e o número de corte da rede é o máximo dentre os números de corte de seus terminais. Este número permite avaliar se uma realização é ótima ou não, mas não permite a existência de um algoritmo ótimo. De fato, Arnold (Harvard-1982) provou que encontrar uma realização com mínimo congestionamento é NP-difícil. Trang, Marek-Sadowska e Kuh em 1984 apresentam uma heurística mostrando que as redes com maiores números de corte devem ser postas no centro da permutação. Posteriormente, Du e Liu (1987) avaliam quando esta heurística produz resultados ótimos e quando não produz, e propõem um algoritmo que considera números de corte locais a "grupos" de redes, atingindo resultados melhores. Swerwani e Deogun desenvolveram vários trabalhos nos quais empreendem abordagens baseadas em grafos para a solução de *SRRP*. Estes algoritmos podem ser encontrados em [She93] págs 271-274. Estranhamente em nenhum dos trabalhos citados é modelada a restrição no número de conexões que pode passar entre dois terminais, mas apenas a minimização destes *doglegs*.

7.5.7 Algoritmos para seleção de subconjuntos planares tipo *SSPR*, *TSPR*, e *ETSPR*

Algoritmos para seleção de subconjuntos de redes que podem ser roteadas em

regiões planares baseiam-se quase que exclusivamente na identificação de conjuntos independentes de vértices (*MIS*) em um grafo circular. Este é o caso de *SSPR*, que aparece também com os nomes de *MSO* (*Multiterminal Single-layer One-sided*) e *SPO* (?) em [She95] pág. 168. O problema *TSPR* não tem sua complexidade conhecida, e um algoritmo de aproximação foi apresentado por Cong, Preas, e Liu em 1990, citados por em [She95] pág. 161. O problema *ETSPR* é ainda considerado um problema aberto, mas para o qual um algoritmo de aproximação também foi proposto por Bhingarde, Panyam e Sherwani em 1993, citados por [She95] pág. 161. Em geral podemos encontrar algoritmos para estes problemas ou para algumas variantes suas em [She95] caps. 6 e 7.

7.5.8 Algoritmos para assinalamento *BTA* e *CPA*

Para o assinalamento *BTA*, a definição dos intervalos possíveis apresentada em [Joh97] pág. 45 é computada diretamente se temos os terminais armazenados em forma simbólica (não espacial). Esta formulação é semelhante àquela apresentada em [She95] pág. 98, sendo que ambas são muito mais simples do que aquela apresentada em [Don93] para o mesmo problema. Com base nestes intervalos, Bhingarde et al., citados por [She95] pág. 96-97, apresentam um algoritmo ótimo com programação dinâmica e *backtracking* para solucionar os problemas de *BTA* em ambas as bordas de um canal. É importante ressaltar que o modelo apresentado para *BTA* não é realista. Na prática, algumas das redes terão terminais vizinhos, e suas conexões podem ser feitas na região entre os terminais e a borda, como definido nos outros problemas. Então, o que ocorre é um misto destes dois tipos de problemas. Há regiões nas quais roteamentos planares serão possíveis e ocuparão a área *OTC*. Restando ou não espaço, as conexões de terminais, ou já parcialmente feitas, poderão ser assinaladas (ou não) a um determinado intervalo na borda das células. Assim, a modelagem dos intervalos precisa ser refinada ([Joh97] pág. 45) para que depois o *BTA* restante seja exato, e o roteamento correspondente seja também encontrado sem colidir com as conexões planares completas em *OTC*.

Em um ambiente de roteamento de área, o problema de *CPA* é especificamente abordado por [Kao95]. A definição dos pontos de cruzamento em cada *GRC* é feita analisando os cruzamentos entra as conexões ao longo de uma série de *GRCs*, horizontal e verticalmente. O algoritmo proposto roda em tempo linear em virtude de sua complexidade ser limitada pelo tamanho das *GRCs*, o que não cresce com o tamanho do circuito. São modelados os pontos possíveis de cruzamento, os terminais e obstáculos internos a cada *GRC*. O algoritmo usa uma etapa específica de re-roteamento global, para solucionar redes que não tenham sido completadas. A principal vantagem deste algoritmo, além de sua originalidade, está na avaliação dos cruzamentos, o que é topologicamente um avanço. Entretanto, apesar de várias redes serem re-roteadas simultaneamente, com o aumento do tamanho do circuito se terá um aumento no número de vezes que o processo se repete, e isto não se enquadra na linearidade citada. É importante ressaltar que o problema de área routing tem seu universo próprio, e *CPA* é apenas uma abordagem existente, mas não exatamente a caracterização natural do problema.

8. Hierarquia de Problemas em Roteamento

Há duas razões principais para que um problema a ser solucionado tenha que ser dividido em vários subproblemas. A primeira é ser um problema complexo, de difícil modelagem. A segunda é, mesmo para um problema mais simples, ter um número de entradas muito grande. Problemas genéricos são em geral mais complexos, e pelas características de um problema completo, pode-se identificar claramente algumas partes que são independentes. Deste modo se tem pontos específicos para a divisão necessária. Para o roteamento de um CI, esta divisão envolve tanto subproblemas puramente de roteamento como outros de posicionamento, lógica, etc... Em outras palavras, a decomposição dos problemas e composição das soluções são necessárias e implícitas nas metodologias ou estilos de leiaute usados. Por exemplo, o roteamento de área em [Kao95] é decomposto como mostra a fig. 8.1. Embora se possa ter problemas completamente independentes, isto nem sempre ocorre. Assim, nesta hierarquia de problemas, existe duas possibilidades de dependência, que são a dependência cíclica, e a criação de problemas solúveis.

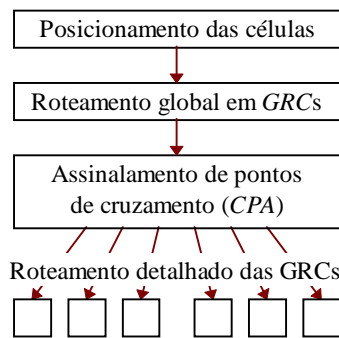


FIGURA 8.1 - Exemplo de decomposição de problemas

8.1 Dependência cíclica

A dependência cíclica ocorre quando a definição de um problema A depende da solução de um problema B, cuja definição depende da própria solução de A. Esta dependência é normalmente apenas parcial, sendo que ambos os problemas possuem dependência de outros dados que podem ser fixos ou oriundos de outros problemas, como mostra a fig. 8.2. Quando eles são fixos em todos os casos, talvez eles possam ser explorados incorporando seu conhecimento nos algoritmos usados para resolver os problemas.

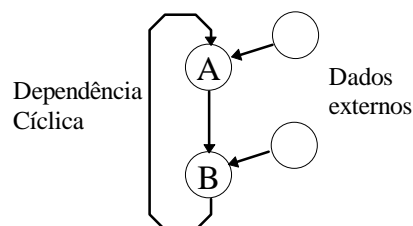


FIGURA 8.2 - Dependência cíclica entre dois problemas A e B

É fácil de encontrar exemplos deste tipo de dependência. O algoritmo de troca de terminais equivalentes por banda apresentado em [Joh97], págs. 39-43, depende completamente da informação de extremidade destes terminais em relação às conexões nos canais adjacentes. Ora, esta informação somente pode ser exata se o roteamento global já tiver definido quais são os segmentos de conexão usados para cada rede. Para canais com capacidades fixas, como *gate arrays*, este mesmo roteamento global necessita avaliar exatamente a capacidade dos canais, e portanto, precisa saber em que posição os terminais de cada rede estão. Esta informação é obviamente alterada pelo algoritmo de troca de terminais.

Existe duas considerações para a redução deste tipo de conflito sem o uso de iterações, e devem estas ser usadas em conjunto. A primeira é comparar a quantidade de informações dependentes e independentes de cada um dos problemas, e fazer em primeiro lugar aquele problema que apresenta menor sensibilidade à dependência cíclica. A segunda é usar algum método de aproximação das informações que deveriam ser providenciadas pelo problema que será feito em segundo lugar. Nem sempre é possível que os erros desta aproximação mantenham a primeira solução válida. Se os erros inviabilizam a solução, a iteração é necessária. Quando no entanto, os erros são aceitáveis, apenas nos afastamos da solução ótima, mas permanecemos com um processo que termina em um passo. A aproximação será geralmente uma estimativa de alguma solução para o segundo problema, e podemos acrescentar um refinamento do primeiro problema ainda em uma passagem, como mostra a fig. 8.3.

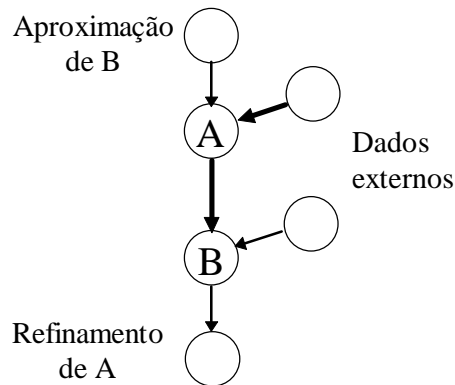


FIGURA 8.3 - Quebra de dependências cíclicas em uma passagem.

8.2 Criação de problemas solúveis

O segundo tipo de dependências é aquele que afeta a garantia de se encontrar uma solução para um dos problemas, podendo ocorrer mesmo em dependências não cíclicas. Seja dois problemas A e B, B dependendo de A, sendo que B tem solução para um conjunto do universo de instâncias possíveis, mas não para todas elas, como mostra a fig. 8.4. O problema é: como fazer com que a solução de A sempre gere dados que se encontrem dentro do conjunto solúvel de B? Esta relação ocorre com frequência entre problemas de leiaute de circuitos. As metodologias que mais tiveram sucesso foram justamente aquelas que venceram este tipo de dificuldade, algumas pela sua inexistência.

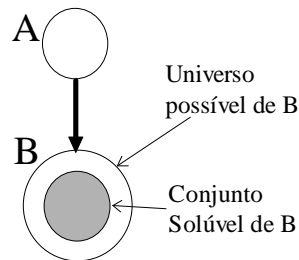


FIGURA 8.4 - Universo de problemas com sub-conjunto solúvel

Para um dado problema que não apresenta solução para todas as entradas, somente o fato de analisar quando uma instância sua tem ou não solução, é geralmente mais complexo do que realizar esta solução. Isto mostra a dificuldade de identificar cada elemento do conjunto solúvel de B. O problema de gerar soluções possíveis é o problema de definir completamente o conjunto solúvel de B.

Para solucionar estas situações pode-se trabalhar tanto em B quanto em A. Se B é um problema muito complexo, cuja solução não é exata, pode-se procurar melhores algoritmos para solucioná-lo, de forma que o conjunto solúvel de B seja ampliado. Esta é a principal ocupação da pesquisa por algoritmos de otimização combinatória em geral, e mesmo de algoritmos heurísticos dedicados a problemas particulares.

Pode-se trabalhar no problema A, ou nos dados que dele provêm, de modo a limitar o conjunto de suas soluções dentro do universo possível. Limitar este conjunto pode ser bastante fácil, mas nem sempre é fácil fazer com que ele case com o conjunto solúvel de B. Mesmo quando se consegue caracterizar bem este conjunto, como se mostrou ser difícil, esta caracterização pode não ser fácil de usar dentro do problema A, porque isto praticamente significa solucionar A e B juntos. Ora, foi visto que a divisão em subproblemas separados é necessária, e portanto, solucioná-los juntos pode ser impraticável. Apesar destas dificuldades, que devem ser avaliadas, há muitas situações onde a criação de problemas sempre solúveis pode ser encontrada. A figura 8.5 ilustra o aumento do conjunto solúvel, o uso da caracterização exata, a limitação a um subconjunto menor, e a caracterização de um conjunto inexato, mas cuja probabilidade de gerar problemas insolúveis é aceitável. Neste último caso, pode-se gerar alguma perturbação em A para gerar outra solução, ou usar algum método dedicado para remover os conflitos que tornam B insolúvel, se estes são casos restritos locais.

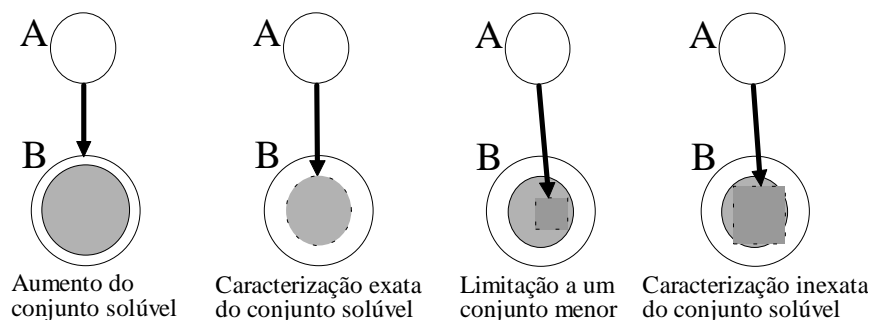


FIGURA 8.5 - Abordagens para criação de problemas solúveis

8.3 Estudo de caso sobre roteamento de área

O roteamento de área (*area routing*) aplicado a macro-blocos, a circuitos *MBC*, a *gate arrays* sem canais, ou mesmo para *MCMs*, é de grande importância atualmente, devido à disponibilidade de várias camadas para roteamento. Estas camadas permitem a busca de circuitos com “*zero routing footprint*”, através da eliminação dos espaços dedicados para roteamento. Segundo [She95] (pág. 15), são usados algoritmos *greedy* e *maze* para endereçar o problema de roteamento de área. Já para o roteamento de *MCMs* com um número grande de camadas, Sherwani utiliza uma abordagem bem mais objetiva, através da seleção de subconjuntos planares para o roteamento de caixas de conexão em forma de torre (devido às várias camadas).

A inexistência dos espaços dedicados para roteamento restrito dificulta muito a subdivisão do problema. As abordagens que decompõem o roteamento em *GRCs*, como em [Kao95], fazem uma divisão artificial do área, e portanto, a definição dos subproblemas se torna bastante arbitrária. Daí provém a necessidade de resolver o problema de *CPA* coerentemente entre os diversos *switch boxes* que devem ser posteriormente roteados. Esta abordagem é praticável. Entretanto, sua otimização é muito difícil. Esta dificuldade advém de dois principais fatores. Em primeiro lugar, há forte dependência entre *CPA* e roteamento dos *switch boxes*, o que se traduzirá em dependências fortes entre todos os problemas de roteamento detalhado. Em segundo lugar, há grande dificuldade em otimizar o uso de cada *switch box*, pois se for limitado muito o número de conexões em cada um, eles serão subutilizados, e se o roteamento global especificar um número maior de conexões, pode facilmente gerar problemas de *switch box* insolúveis.

Em contraposição a esta abordagem, pode-se buscar uma metodologia onde o roteamento detalhado seja realizado integralmente, e não em partes restritas. Esta é a visão particular do autor. A localidade das decisões pode ser explorada através da localidade do próprio algoritmo, mas sem as divisões artificiais e rígidas das *GRCs*. Este é justamente o modo pelo qual operam os algoritmos *greedy*. Assim, pode-se realizar o roteamento global que atribui as conexões às *GRCs*, e utilizar esta informação como guia para um roteamento detalhado de toda a área. Este roteamento é detalhado porque não deve mudar as formas e a distribuição das redes globalmente. Ele é integral porque irá considerar as dependências entre as *GRCs*, e sua localidade é processar o roteamento linha a linha, ou coluna a coluna. Um algoritmo *greedy* peca por não saber se sua convergência está sendo para um mínimo local ou global, sendo isto baseado nas decisões que tomou inicialmente. Então, avaliando a solução global e restringindo as decisões do algoritmo *greedy* a questões locais evita-se quase totalmente esta deficiência. Um algoritmo desta natureza foi proposto em [Joh94].

O roteamento de área através da seleção de múltiplos roteamentos planares também é uma abordagem integral muito interessante. A seleção de conjuntos planares a partir de todas as conexões da área, no entanto, tem maior complexidade, assim como o roteamento de cada camada será mais complexo, porque o número de camadas não é tão grande, e a área do circuito é. Explorar a posição dos terminais para possíveis subdivisões deste problema, como era feito em *PCBs* e *PWBs* com componentes *DIP* é uma ótima alternativa. É também semelhante às técnicas de *OTC* que vem sendo utilizadas, podendo haver uma transição suave entre metodologias e estilos de leiaute. Destaca-se finalmente abordagens semelhantes ao algoritmo hierárquico, que começam

por fazer a distribuição global de conexões, e refinam esta solução até a definição das rotas detalhadas de cada rede. Em todos estes casos, as mesmas dependências previamente vistas devem ser cuidadosamente avaliadas, pois há grandes chances de especificação de problemas insolúveis. Acredita-se, porém, que isto ocorra com muito menos frequência que na abordagem de *CPA* e *switch boxes*. O uso de algoritmos do tipo *maze*, embora ainda muito difundido, não representa uma boa abordagem integral, pois introduz dependências entre o roteamento de cada rede, além de requerer tempo proporcional ao quadrado da área para execução.

8.4 Estudo de caso sobre modelos para *QCL*

QCL corresponde a uma forma de implementação de *ASICs* na qual são usadas matrizes do tipo *gate array* cuja personalização se dá através de uma única camada metálica, embora o processo no qual a matriz foi projetada possa conter várias. Tanto para as conexões internas das células, a partir das bandas de transistores, quanto para o roteamento nos canais, são utilizados conjuntos de estruturas fixas de roteamento, pré fabricadas em um nível inferior de metal, não programável (fig. 8.6). As vantagens desta forma estão na redução ainda maior do número de etapas de processo para a diferenciação dos circuitos, e na conseqüente redução de custo e tempo para o mercado. Ao invés de requerer quatro máscaras, como em um *gate array* de dois níveis, a personalização por um único nível requer apenas uma, porque até mesmo os furos para por em contato as duas camadas metálicas já podem estar implementados. Basta corroer a segunda camada metálica para programação, ou usar algum processo de corte de conexões com laser, por exemplo, para programar o dispositivo.

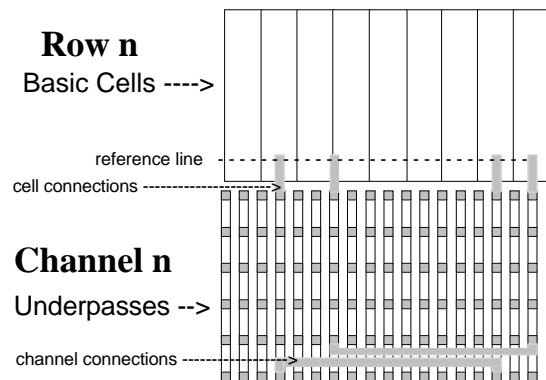


FIGURA 8.6 - Roteamento *QCL* com estruturas de passagem

Há diferentes modelos para *QCL*. Em geral, o roteamento de canal é realizado com uma camada metálica sobre um conjunto de estruturas de passagem verticais, denominadas de *underpasses*. Estes *underpasses* já estão fabricados na primeira camada metálica, e o isolamento já possui furos em posições determinadas. A matriz apresentada em [Sim92] (fig. 8.7), cujo roteamento é abordado em [Joh97], possui uma via depois de cada duas trilhas horizontais que passam sobre os *underpasses*. Em [Sun91] e [Don93] são apresentadas algoritmos para roteamento de duas arquiteturas diferentes. Em ambos os casos os *underpasses* apresentam apenas duas vias, uma em cada extremidade. Estes dois trabalhos referem-se a arquiteturas desenvolvidas e utilizadas no Japão.

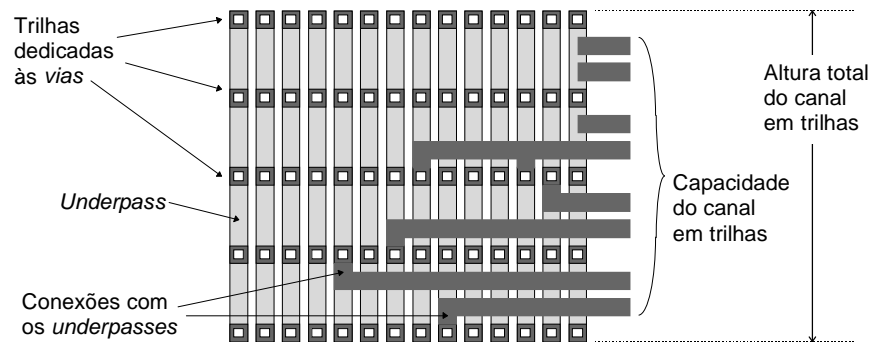


FIGURA 8.7 - Arquitetura dos canais de roteamento em [Sim92].

Trabalhos anteriores também consideram a estratégia de roteamento de uma camada e meia (*one and half layer routing*). Em [Ser91] é apresentado um modelo mais genérico para o roteamento de regiões com *underpasses* em diversas posições e orientações. A técnica empregada é de roteamento rede por rede, não muito adequada para roteamento de canais de tamanho fixo. O trabalho apresentado em [Son85] já endereçava um problema semelhante a *QCL*, mas com tecnologia de uma camada metálica, onde as estruturas de passagem, denominadas de *crossunders*, eram implementadas em polissilício. As posições das *vias* eram livres, e neste caso se tem a mesma ausência de restrições verticais que na arquitetura de [Sim92]. Entretanto, o passo de polissilício considerado na época foi de aproximadamente o dobro do passo de metal. Assim, todo o problema consistia no assinalamento dos terminais aos *crossunders*, utilizando um roteamento que interfere nas conexões do canal. É mostrado um algoritmo guloso para solucionar o problema.

Como nos demais *gate arrays*, a quantidade de recursos restrita impõe várias dificuldades ao roteamento, e exercita as dependências já vistas. Entretanto, neste caso particular de *QCL*, a existência das estruturas fixas de roteamento amplia bastante o conjunto de problemas e as dependências detalhadas entre eles.

A arquitetura proposta em [Sim92] utiliza um modelo de roteamento entre os *underpasses* de canal que permite a solução ótima em tempo linear pelo algoritmo *LEA*, já que não há pode haver arestas em *VCG* quando as conexões nunca começam ou terminam na mesma coluna. Mesmo neste caso, o assinalamento dos terminais aos *underpasses*, dado por um roteamento em rio, e a prévia definição da presença e posição destes terminais nas bordas de célula, apresentam a dificuldade de geração de problemas solúveis. Esta situação é bem explicada em [Joh97] págs. 27-29. É empregada uma bordagem pragmática que limita o número de terminais de células, representando a limitação do conjunto de problemas criados a um sub-conjunto pequeno, que na prática está sempre dentro do conjunto solúvel.

Os modelos de canal com apenas duas vias em cada *underpass* acrescentam uma dependência a mais em série. Enquanto que no modelo com vias intercaladas qualquer assinalamento de *underpasses* gera um problema de canal solúvel, no modelo com apenas duas vias, apenas um pequeno conjunto de assinalamentos poderá ter solução, e esta pode também não ser encontrada. O problema resultante é semelhante a um *SRRP*. Na verdade, qualquer solução de um *SRRP* pode ser mapeada para o modelo de *underpasses* com duas vias. Entretanto, há conexões que podem ser feitas neste modelo

e que não podem ser feitas em *SRRP*, pois corresponderiam a ir até o infinito nas regiões superior e inferior para fazer uma conexão. Além disto, a avaliação do congestionamento é alterada, pois a capacidade não está dividida entre as regiões superior e inferior do *SRRP*, mas é compartilhada. O problema assemelha-se bem mais ao *ETSPR*, exceto pelo fato de que Sherwani define como *ETSPR* apenas a seleção de um conjunto de redes realizável no modelo. No caso de *QCL*, é necessário realizar todas as redes neste modelo.

Os trabalhos desenvolvidos em [Sun91] e [Don93] apresentam boas estratégias para endereçar este problema, mas devemos guardar algumas reservas. Em primeiro lugar, as formulações, bem como algumas partes dos algoritmos, não são úteis na prática. É assumido, por exemplo, que cada rede é formada por apenas dois terminais, um na base e outro no topo do canal. Ora, embora acredite-se que a implementação dos autores tenha contemplado casos mais genéricos, a formulação apresentada não o faz, e portanto, é irreal. Em segundo lugar, os algoritmos exploram com eficiência o espaço de soluções, mas em nenhum momento é discutido o procedimento que deve ser adotado no caso em que a solução não existe, o que pode acontecer em diversos momentos. Acredita-se portanto, que além do uso de técnicas semelhantes para a solução dos problemas detalhados, faz-se necessário um estudo técnico bem objetivo para melhor caracterizar a solubilidade destes problemas.

Por fim, apresenta-se a seguinte questão: tendo-se a necessidade de desenvolver uma matriz *QCL* e seu correspondente suporte de *CAD*, qual o melhor modelo de canal a ser adotado, e quais são os tamanhos necessários para as áreas de roteamento em rio, de canal, e espaçamento entre terminais, de forma a ter a melhor garantia de solução sem comprometer o custo em área? Há diversos modelos possíveis, que se diferenciam pelos seguintes fatores:

- orientação dos *underpasses*;
- conexão dos *underpasses* com os terminais;
- número de filas com *underpasses*;
- espaçamento para roteamento planar;
- número de cabeças de via em cada *underpass*;

Os trabalhos citados consideram apenas *underpasses* verticais, com exceção de [Ser91], que considera estruturas em posições arbitrárias. O uso de *underpasses* horizontais regularmente espaçados, entretanto, pode ser um modelo interessante na ausência de restrições a muitas vias em cada conexão. Um algoritmo para roteamento neste modelo deve basicamente cuidar da como fazer as conexões dos terminais com alguma trilha, sendo sua propagação trivial. A vantagem é usar melhor a altura do canal para as conexões horizontais, o que é perdido nos demais modelos.

Em [Sun91] os *underpasses* estão pré conectados com os terminais de células em uma das bordas do canal. Este é um modelo interessante, o qual também pode ser usado para aproveitar melhor a altura do canal, já que as linhas de conexão dos terminais e a dos *underpasses* são compartilhadas. Uma variação deste modelo seria usar mais de uma fila de *underpasses*, ou seja, dois por coluna. Então, cada ‘meio *underpass*’ pode estar pré conectado com um terminal possível de célula. Isto elimina a etapa de assinalamento de *underpasses*. De forma genérica, entretanto, podemos imaginar que ela estará

igualmente presente na tarefa de conectar as duas metades do canal. Esta conexão pode ser feita com espaçamento zero ou com um outro espaçamento maior, através de roteamento em rio.

O espaçamento é em geral necessário nestes modelos para permitir flexibilidade nos assinalamentos (conexões entre terminais e *underpasses*). Se não há espaço, a conexão somente pode ser feita com o *underpass* alinhado, e isto praticamente inviabiliza qualquer compartilhamento de recursos. Se não há um *underpass* para cada terminal em ambas as bordas, e isto é normal, eles competem pelos mesmos recursos, e o espaçamento permitirá a conexão com *underpasses* vizinhos. O aumento deste espaço pode melhorar um pouco esta flexibilidade, mas provavelmente deve ficar restrito a uma ou duas trilhas. Um espaçamento maior tende a não ser bem utilizado, já que, em qualquer roteamento em rio, há grupos de conexões que se dirigem a um sentido e grupos que se dirigem a outro. Se dois grupos se dirigem a direções opostas, então não adianta aumentar a área de roteamento planar, pois as conexões não podem se cruzar nesta área.

No caso do usar este modelo com roteamento em rio central, o problema é basicamente o seguinte. Dadas duas ordens de terminais distintas, encontrar uma terceira ordem tal que a troca de ordem a partir das originais para esta seja possível de rotar nas áreas internas dos *underpasses* inferiores e superiores, e tenha uma defasagem central que possa ser acomodada na área de roteamento em rio existente. Isto é semelhante ao problema de assinalamento de terminais (*TA*) citado em [She95] págs. 150-151, mas com maior flexibilidade interna, acompanhado da estimativa de solubilidade do roteamento de cada região (*ETSPR* completo). A avaliação de grafos de permutação, ou dos diagramas de permutação de *SRRP*, também são bons indicadores neste processo.

Finalmente, é necessário comentar o fator mais evidente, qual seja, o número de vias existentes por *underpass*. É interessante lembrar que se pode permitir a programação das vias, e neste caso volta a ser possível conectar os *underpasses* otimamente com um algoritmo *LEA*, como em [Joh97], mas sem desperdício de área na altura do canal. Isto não é feito normalmente porque a complexidade do processo de personalização cresce bastante apenas com esta alteração. Um determinado conjunto de conexões ponto a ponto pode garantidamente ser realizado no modelo com apenas duas vias se o grafo correspondente de sobreposição de intervalos puder ser colorido com duas cores [Don93]. Assim, pode-se avaliar em um conjunto grande de casos práticos qual a coloração dos grafos correspondentes, e então utilizar um número de vias $v = r + 1$, onde r é o número de regiões que acomodam, cada uma, um conjunto de conexões colorido com duas cores. A ordem em que os conjuntos são selecionados pode alterar o resultado. Poucas redes muito complexas levam cada uma a um aumento na mínima coloração do grafo, e portanto, se roteadas em regiões com grande espaçamento, podem desperdiçar esta área. Isto ocorre pela sua natureza, que é crítica em termos de conexões, mas não em área. Para acomodar tais conexões pode ser útil ter *underpasses* mistos, com algumas regiões maiores, e com vias intercaladas em duas trilhas, onde as conexões mais complexas serão feitas.

A seleção destes modelos e parâmetros, bem como dos algoritmos necessários, somente poderá ser definida com experimentos práticos, já que é fortemente dependente da quantidade e posição dos terminais, e das redes que os unem. De qualquer forma, temos um exemplo característico em que o problema é decomposto em um conjunto de

subproblemas nem sempre solúveis. Os problemas que devem ser resolvidos, são, em resumo: *ETSPR* completo; avaliação da solubilidade do mesmo; permutação de terminais; assinalamento semelhante a *BTA* duplo, de terminais a *underpasses*; etc...

8.5 Convergência com desempenho

Segundo [Pap95], o problema geral de uma metodologia é a sua característica de não convergência, a qual acentuou-se bastante com as tecnologias submicrônicas. Os processos de síntese de alto nível dominados são mais ou menos “caóticos”, onde uma perturbação na entrada gera diferentes e incontroláveis resultados na estrutura de saída. Esta, por sua vez, é passada a um tipo de sistema de *place and route* para o qual mudanças na entrada também não possuem resultados previsíveis no leiaute. Assim, a tarefa de resintetizar o circuito, quando os requisitos de desempenho não são atendidos, torna-se um processo de tentativa e erro, duvidoso, e não convergente. Um boa metodologia deve prover um processo observável (onde se saiba o grau de otimização atual e possível), controlável, e, é claro, convergente.

A convergência das metodologias de leiaute sempre foi explorada ao preço da perda de flexibilidade, e em consequência, de otimizações em geral. Quase todos os modelos de leiaute, e em especial de roteamento, adotam fortes simplificações em relação a sua generalidade, e isto torna sua solução muito eficiente e garantida, isto é, convergente, dentro dos limites estabelecidos. Exemplos são o estilo de leiaute *standard cell* e seu roteamento, ou então a própria ordem de etapas separadas empregada no ciclo de projeto.

Atualmente, porém, não se pode pagar o preço destas simplificações, pois elas representam principalmente o desempenho das conexões do circuito, que domina os problemas críticos de um projeto. Assim, quando uma determinada parte do leiaute sintetizado não atende às restrições de funcionamento necessárias, torna-se necessário re-sintetizá-la. Nas metodologias antigas, isto significava repetir o processo com algum novo parâmetro ou dado, mas se o processo é “caótico”, e não facilita esta iteração, todo o resultado é alterado, e torna-se difícil convergir.

A mesma característica de gerar apenas problemas solúveis, vista sob o prisma estrito de área, deve ser buscada para o funcionamento do circuito, ou seja, para o desempenho de seus elementos e conexões. Assim, um processo convergente é aquele que, de acordo com uma determinada estratégia, converge para a solução do problema maior, que é o projeto do circuito que apresente o desempenho necessário. Esta característica deve ser buscada, em nível de leiaute, pelo estudo cuidadoso da estrutura topológica do seu roteamento, e pelo conjunto de subproblemas globais ou detalhados que serão necessários para solucioná-lo, sem deixar de levar em conta os modelos comportamentais que regem seu funcionamento.

8.6 Relacionamento com outros temas

Já foi visto que existe uma íntima relação entre os modelos de posicionamento, estilo de leiaute, de células e de terminais, e os problemas de roteamento. Geometricamente falando, é necessário que todos estes modelos sejam coerentes para a geração eficiente do leiaute. Entretanto, embora se tenha avaliações puramente

geométricas que modelam naturalmente outros aspectos do circuito, existe a necessidade de agregar valor a todo este processo, seja nas etapas de síntese, seja nas de análise. Uma vez que as conexões tornaram-se praticamente a questão principal de um projeto, deve-se considerar o relacionamento do roteamento com todas as demais etapas de projeto, pertençam elas à síntese física ou não.

Um exemplo claro deste relacionamento é a otimização simultânea de *drivers* e de larguras de conexão para uma rede crítica. Podemos ter neste mesmo problema o dimensionamento de transistores e a inserção de reforçadores [Con94]. Este é um caso evidente. De forma geral, podemos ter alterações na estrutura (no *netlist*) do circuito durante etapas de síntese física, alterações estas necessárias justamente em função da síntese física. Os critérios para efetuar tais alterações geralmente estão relacionados a análise de temporização, dissipação de potência, ou dissipação de calor, entre outros. Há diversos trabalhos de pesquisa e desenvolvimento endereçando re-síntese e re-temporização [Aok95]. A integração destes diferentes problemas em um processo convergente parece ser o desafio central. Alguns trabalhos que abordam estas questões são [Ste97] [Hol97] [Cam97] [Keu97].

As mudanças na síntese física para garantir o desempenho adequado nos projetos do próximo século serão fatalmente no sentido de planejar todo o circuito desde o princípio tendo como foco central as suas conexões, não somente em sua geometria, mas com análises elétricas complexas e precisas, de forma a controlar o comportamento do que está sendo gerado. Isto é em resumo o que é apresentado neste conjunto de trabalhos, colocado aqui com as palavras tomadas de [Lap97].

9. Conclusões

Este trabalho procurou situar a síntese física dentro do contexto do projeto de circuitos integrados *VLSI*, caracterizando seus principais problemas, e as soluções mais comuns que tem sido adotadas. Em face às novas condições tecnológicas, os modelos, metodologias e estilos de leiaute precisam ser reavaliados. A síntese física continua tendo um papel fundamental no projeto dos circuitos, e sua otimização tem sido de crescente importância. A pesquisa diretamente em síntese física, ou no seu relacionamento com as demais etapas de projeto, é indispensável para que se possa vencer a defasagem que existe entre o aumento potencial da complexidade dos projetos e a produtividade dos projetistas.

De todo o conjunto de problemas de síntese física, o roteamento é a questão mais delicada, pois são as conexões do circuito que dominam atualmente o atraso dos sinais. O roteamento envolve uma estrutura ou modelo geral para todo o circuito, e uma hierarquia de outros problemas, cada qual com suas características e restrições próprias. Este complexo deve estar em sintonia com a metodologia de projeto, e deve ter a característica indispensável da convergência. Esta convergência foi abordada neste trabalho de três principais formas: na avaliação de toda a síntese física, no estudo teórico da definição de problemas e algoritmos para solucioná-los, e em diversos exemplos práticos e detalhados, os quais ilustram e justificam estes pontos.

Os problemas básicos foram enriquecidos com alguns questionamentos críticos, e foram também relacionados entre si. Há problemas bem compreendidos, e outros que se encontram em aberto. Espera-se que o texto possa servir como fonte básica de informação e referência sobre síntese física e roteamento, e também motivar novos experimentos, propostas e modelos para otimização destes processos.

Bibliografia

- [Alp95] ALPERT, C. J. et al. Prim-Dijkstra Tradeoffs for Improved Performance-Driven Routing Tree Design. **Transactions o CAD of ICs and Systems**. v.14 n.7, p. 890, 1995.
- [Aok95] AOKI, T.; MURAKATA, M.; MITSUHASHI, T.; GOTO, N. Fanout-tree Restructuring Algorithm for Post-Placement Timing Optimization. In: IFIP VLSI'95, Chiba, Japan, 29 ago.-1 set., 1995. **Proceedings...**
- [Bro91] BROWN, Andrew. **VLSI Circuits and Systems in Silicon**. London: McGraw-Hill, 1991. 483p.
- [Bur83a] BURSTEIN, M.; PELAVIN, R. Hierarchical channel routing. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 20., June 1983. **Proceedings...** New York: IEEE, 1983. p.591-597.
- [Bur83b] BURSTEIN, M.; PELAVIN, R. Hierarchical wire routing. **IEEE Transactions on Computer-Aided Design of ICs and Systems**, New York, v. CAD-2, n. 4, p.223-234, Oct. 1983.
- [Bus97] BUSHROE, R. G., DASGUPTA, S., DENGI, A. et al. Chip Hierarchical Design Systems (CHDS): A Foundation for Timing-Driven Physical Design into the 21st Century. In: International Symposium on Physical Design, 1997 (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.212-217.
- [Cam97] CAMPOSANO, Raul. The Quarter Micron Challenge: Integrating Physical and Logic Design. In: International Symposium on Physical Design, 1997 (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.211.
- [Che94] CHENG, Chih-liang Eric. RISA: Accurate and Efficient Placement Routability Modeling. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.690.
- [Chi97] **ChipExpress** - The ASIC Time to Market Solution. '<http://www.chipexpress.com>'. Visitada em 31/07/1997.
- [Con90] CONG, Jason; PREAS, B.; LIU, C. L. General Models and Algorithms for Over-the-Cell Routing in Standard Cell Design. In: IEEE DESIGN AUTOMATION CONFERENCE, 27. **Proceedings...** New York: IEEE, 1990. p.709-715.
- [Con94] CONG, Jason; KOH, Cheng-Kok. Simultaneous Driver and Wire Sizing for Performance and Power Optimization. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.206.
- [Con95a] CONG, Jason; XU, Dongmin. Exploiting Signal Flow and Logic Dependency

- in Standard Cell Placement. In: IFIP VLSI' 95, Chiba, Japan, 29 ago1 set., 1995. **Proceedings...**
- [Con95b] CONG, Jason; LEUNG, K.-S. Optimal Wiresizing Under Elmore Delay Model. **Transactions o CAD of ICs and Systems**. v.14 n.7, 1995. p.321.
- [Con97] CONG, Jason; MADDEN, Patrick. Performance Driven Global Routing for Standard Cells. In: International Symposium on Physical Design, 1997 (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.73-80.
- [Coy95] McCOY, Bernard A.; ROBINS, Gabriel. Non-Tree Routing. **Transactions on CAD of ICs and Systems**. v.14 n.6, 1995. p.780.
- [Deu80] DEUTSCH, David N., GLICK, Paul. An Over-the-Cell Router. ACM/IEEE DAC, 19., **Proceedings...** New York: IEEE, 1980. p.32-39.
- [Don93] DONG, Sai-keung; SUN, Yachyang; LIU, C. L. et al. Two Channel Routing Algoritms for Quickly Customized Logic. In: EDAC, 1993, Paris, FR. **Proceedings...** Los Alamitos: IEEE, 1993. p.122-126.
- [Dor81] DOREAU, Michel T.; KOZIOL, Piotr. T W I G Y : A Topological Algorithm Based Routing System. In: DESIGN AUTOMATION CONFERENCE, 18., 1981. **Proceedings...** New York: IEE, 1981. p.746-755.
- [EDA97] Electronic Design Automation Consortium. **1996 EDA Revenues Top 2.3 Billion**. 'http://www.edac.org/EDAC/News/EDAC17.html'. Visitada em 20/09/1997.
- [Gaj83] GAJSKI, D.; KUHN, R. New VLSI Tools. **IEEE Computer**. New York, v. 16, n. 12, p.11-14, Dec. 1983.
- [Gaj88] GAJSKI, D. **Silicon Compilation**. 1988.
- [Gao94] GAO, Tong; LIU, C. L. Minimum Crosstalk Switchbox Routing. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.610.
- [Gar79] GAREY, M. R., JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. New York: Freeman, 1979.
- [Gop83] GOPAL, I. S.; COPPERSMITH, D.; WONG, C. K. Optimal Wiring of Moveable Terminals. **IEEE Transactions on Computers**, C-32 p. 845-858, Sep 1983.
- [Gru97] GRUNDMANN, W. J. Physical Design Realities for Digital's StrongARM and Alpha Microprocessors. In: International Symposium on Physical Design, 1997 (ISPD'97). Napa Valley, CA. April 14-16. **Proceedings...** (Invited Talk).
- [Gun95] GÜNTZEL, J.; REIS, R.; FLORES, A.; JOHANN, M. A Novel Approach for ASIC Layout Generation. In: 38th Mid-West Symposium on Circuits and

- Systems (MWSCAS' 95), Rio de Janeiro, Brazil, August 13-16, 1995. **Proceedings...**
- [Ham84] HAMACHI, G. T.; OUSTERHOUT, J. K. A Switchbox Router with Obstacle Avoidance. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 21., June 1984. **Proceedings...** New York: IEEE, 1984. p.173-179.
- [Her95] HER, T. W.; WANG, T.-C.; WONG, D. F. Performance-Driven Channel Pin Assignment Algorithm. **Transactions o CAD of ICs and Systems**. v.14 n.7, p.849. 1995.
- [Hol97] HOLT, G.; TYAGI, A. Minimizing Interconnect Energy Through Integrated Low-Power Placement and Combinatorial Logic Synthesis. In: International Symposium on Physical Design, 1997, (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.48-53.
- [Hor97] HOROWITZ, Ellis; SAHNI, Sartaj; RAJASEKARAN, Sanguthevar. **Computer Algorithms/C++**. New York: W. H. Freeman, 1997. 769p.
- [Hu70] HU, Te Chiang. **Integer Programming and Network Flows**. Massachusetts: Addison-Wesley, 1970. 452p.
- [Hua97] HUANG, Dennis J.-H.; KAHNG, Andrew. Partitioning-based Standard-cell Global Placement with an Exact Objective. In: International Symposium on Physical Design, 1997, Napa Valley, CA. **Proceedings...** New York: ACM, 1997. p.18-25.
- [Joh94] JOHANN, Marcelo O. **Roteamento sobre Células Transparentes**. Porto Alegre, Pós-Graduação em Ciências de Computação/UFRGS, 1994. (Dissertação de Mestrado).
- [Joh95] JOHANN, M.; REIS, R. A Full Over-the-Cell Routing Model. In: IFIP VLSI' 95, Chiba, Japan, 29 ago.-1 set., 1995. **Proceedings...**
- [Joh97] JOHANN, M.; **Projeto Funcional do Roteador GAROTA**. Porto Alegre: CPGCC da UFRGS, 1997. TI-649 (Trabalho Individual). 54p.
- [Joo97] JOOBANI, R.; SIEWIOREK, D. WEAVER: A Knowledge-based Routing Expert. **IEEE Design & Test of Computers**, New York, v. 3, n. 1, p.12-23, Feb 1986.
- [Kah94] KAHNG, Andrew B.; TSAO, Chung-Wen Albert. Low-Cost Single-Layer Clock Trees With Exact Zero Elmore Delay Skew. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.213.
- [Kah95] KAHNG, Andrew B.; ROBINS, Gabriel. **On Optimal Interconnections for VLSI**. Norwell: Kluwer, 1995. 286p.
- [Kao95] KAO, Wen-Chung, PARNG, Tai-Ming. Cross Point Assignment With Global Rerouting for General Architecture Designs. **Transactions on CAD of ICs**

and Systems. v.14 n.3, 1995. p.337.

- [Keu97] KEUTZER, Kurt, NEWTON, Richard, SHENOY, Narendra. The Future of Logic Synthesis and Physical Design in Deep-Submicron Process Geometries. In: International Symposium on Physical Design, 1997 (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.218-223.
- [Kir94] Kirkpatrick, Desmond A.; Sangiovanni-Vicentelli, Alberto L. Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.616.
- [Koi95] KOIDE, Tetsushi et al. A New Performance Driven Placement Method with the Elmore Delay Model for Row Based VLSIs. In: IFIP VLSI' 95, Chiba, Japan, 29 ago.-1 set., 1995. **Proceedings...**
- [Lap97] LAPOTIN, David, P., GHOSHAL, Uttam et al. Physical Design Challenges for Performance. In: International Symposium on Physical Design, 1997 (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997.
- [Las70] LASDON, Leon S. **Optimization Theory for Large Systems**. New York: Macmillan, 1970. 523p.
- [Len90] LENGAUER, Thomas. **Combinatorial Algorithms for Integrated Circuit Layout**. Chichester: John Wiley & Sons, 1990. 697p.
- [Mic94] Giovanni De Micheli. **Synthesis and Optimization of Digital Circuits**. New York: McGraw-Hill, 1994. 579p.
- [Mor93] MORAES, F.; AZEMARD, N.; ROBERT, M.; AUVERGNE, D. Flexible Macrocell Layout Generator. In: 4th ACM/SIGDA Physical Design Workshop, **Proceedings...** p. 105-116 Apr. 19-21, 1993, U.S.A.
- [Mor94] MORAES, F.; ROBERT, M.; AUVERGNE, D.; REIS, R. An Efficient Layout Synthesis Approach for CMOS Random Logic Circuits In: Congresso da SBMICRO, 9, Rio de Janeiro, 8-12 ago. **Proceedings...** SBMICRO, 1994. pg. 48-57.
- [Nai86] NAIR, Ravi, HONG, Se June. The 'Diamond' Chip – An Alternative Rectilinear Gate-Array. In: ICCD, 1986. **Proceedings...** New York: IEEE, 1986. p.123.
- [Pap95] PAPROCKI, Wieslaw; POIROT, Franck. Deep Submicron Design. In: X Congresso da SBMICRO, Canela, 31 Jul.-4 Ago. 1995. **Anais...** SBMICRO/II-UFRGS, Porto Alegre, 1995. p.33.
- [Pre88] LORENZETTI, Michael J., BAEDER, d. Scott, Routing. In: PREAS, B. T., **Physical Design Automation of VLSI Systems**, Menlo Park: Benjamin/Cummings Publishing Company Inc., 1988, chap. 5, p.157-210.
- [Rei88] 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.
- [Rei92] REIS, André I.; GÜNTZEL, José L.; RIBAS, Renato R. Algumas formas de Implementação de ASICs. In SBCCI, 7., 1992, Rio de Janeiro. **Anais...** Rio de Janeiro: UFRJ/SBC/SBMICRO, 1992, 284p. p.35-48.
- [Rei93] REIS, André Inácio. **Geração de Células Transparentes**. Porto Alegre: CPGCC da UFRGS, 1993. 128p. (Dissertação de Mestrado).
- [Rei95] A. Reis, M. Robert, D. Auvergne, R. Reis. Associating CMOS transistors with BDD arcs for technology mapping. **Electronics Letters**, 6th, v. 31, n. 14, p.1118-20. July 1995
- [Sad95] SADOWSKA, Margozata M.; SARRAFZADEH, Majid. The Crossing Distribution Problem. **Transactions o CAD of ICs and Systems**. v.14, n.4, p.423. 1995.
- [Sai95] SAIT, Sadiq M., YOUSSEF, Habib. **VLSI Physical Design Automation - Theory and Practice**. New York: IEEE, 1995. 426p.
- [Sap93] SAPATNEKAR, Sanchin S.; KANG, S. M. Steve. **Design Automation for Timing-Driven Layout Synthesis**. Massachusetts: Kluwer, 1993. 269p.
- [Sap95] SAPATNEKAR, Sanchin S.; HAJJ, Ibrahim N. Timing and Area Optimization for Standard Cell VLSI Circuits Design. **Transactions o CAD of ICs and Systems**. v.14 n.3, p.308. 1995.
- [Sar96] SARRAFZADEH, Majid, WONG, C. K. **An Introduction to VLSI Physical Design**. New York: McGraw-Hill, 1996. 334p.
- [Sek94] SEKI, Mitsuho; INOUE, Kenji; KATO, Kazuo; TSURUSAKY, Kouki. A Specified Delay Accomplishing Clock Router Using Multiple Layers. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.289.
- [Ser91] SERVÍT, Michal, SCHIMIDT, Jan. Strategy of One and Half Layer Routing. **Microprocessing and Microprogramming**. v.32, n.1-5, Aug. 1991. p.417-423.
- [She93] SHERWANI, Naveed; **Algorithms for VLSI Physical Design Automation**. Massachusetts: Kluwer, 1993. 538p.
- [She95] SHERWANI, Naveed; BHINGARDE, Siddharth; PANYAM, Anand. **Routing in the Third Dimension: From VLSI Chips to MCMs**. IEEE, New York, 1995.
- [Sim92] SIMÕES, S. A. et al. Matriz gate array cmos avançada, configurável por um único nível de metal. In: Congresso da SBMICRO, 7, 1992. São Paulo, SP.

Anais... São Paulo: SBMICRO/USP, 1992. p.281-291.

- [Son85] SONG, J. N., CHEN, Y. K. An Algorithm for One and Half Layer Channel Routing. In: IEEE Design Automation Conference, 22., **Proceedings...** New York: IEEE, 1985. p.131-136.
- [Ste97] STENZ, G.; RIESS, B. R.; ROHFLEISCH, B; JOHANNES F. M. Timing Driven Placement in Interaction with Netlist Transformations. In: International Symposium on Physical Design, 1997, (ISPD'97). **Proceedings...** New York: ACM/IEEE, 1997. p.36-41.
- [Sun91] SUN, Y., DONG, S., SATO, S., LIU, C. S. A Channel Router for Single Layer Customization Technology. In: IEEE International Conference on Computer-Aided Design (ICCAD), 1991. **Proceedings...** p.436-439.
- [Sun95] SUN, W.-J.; SECHEN, Carl. Efficient and Effective Placement for Very Large Circuits. **Transactions on CAD of ICs and Systems**. v.14 n.3, p.349. 1995.
- [Tak95] TAKAHASHI, K.; NAKAJIMA, K.; TERAJ, M.; SATO, K. Min-Cut Placement with Global Objective Functions for Large Scale Sea-of-Gate Arrays. **Transactions o CAD of ICs and Systems**. v.14 n.4, p.434. 1995.
- [Tel94] TÉLLEZ, Gustavo E.; SARRAFZADEH, Majid. Clock Period Constrained Minimal Buffer Insertion in Clock Trees. In: IEEE/ACM International Conference on CAD, San Jose, CA, Nov. 6-10, 1994. **Proceedings...** Los Alamitos: IEEE Press, 1994. p.219.
- [Tsa95] TSAY, Yu-Wen; LIN, Youn-Long. A Row-Based Cell Placement Method That Utilizes Circuit Structural Properties. **Transactions on CAD of ICs and Systems**. v.14. n.3, p.393. 1995.
- [Tsui81] TSUI, Raymond Y.; SMITH, Robert, J. A High-Density Multilayer PCB Router Based on Necessary and Sufficient Conditions for Single Row Routing. In: DESIGN AUTOMATION CONFERENCE, 18., 1981. **Proceedings...** New York: IEE, 1981. p.372-381.
- [Tsuk81] TSUKIYAMA, Shuji; KUH, Ernest S.; SHIRAKAWA, Isao. On The Layering Problem of Multilayer PWB Wiring. In: DESIGN AUTOMATION CONFERENCE, 18., 1981. **Proceedings...** New York: IEE, 1981. p.738-745.
- [Yan91] YAN, C.; WONG, D.F. Optimal Channel Pin Assignment. **IEEE Transactions on Computer-Aided Design**, v. CAD-10, n. 11, p.1413-1423, Nov 1991.
- [Yeh95] YEY, C.-W.; CHENG, C.-K.; LIN, T.-T. Y. Optimization by Iterative Improvement: An Experimental Evaluation on Two-Way Partitioning. **Transactions o CAD of ICs and Systems**. v.14, n.2, p.145. 1995.
- [Wag94] WAGNER, Flávio Rech. **Ambientes de Projeto de Sistemas Eletrônicos**. UFPE, 1994. (Livro-texto da IX Escola de Computação).

