UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS GRADUAÇÃO EM MICROELETRÔNICA

ADRIEL MOTA ZIESEMER JUNIOR

Síntese Automática do Leiaute de Redes de Transistores

Tese apresentada como requisito parcial
para a obtenção do grau de Doutor em
Microeletrônica

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, maio de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Junior, Adriel Mota Ziesemer

Síntese Automática do Leiaute de Redes de Transistores / Adriel Mota Ziesemer Junior. – Porto Alegre: PGMICRO da UFRGS, 2014.

125 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Microeletrônica, Porto Alegre, BR–RS, 2014. Orientador: Ricardo Augusto da Luz Reis.

1. Síntese física. 2. Compactação do leiaute em 2D. 3. Células lógicas. 4. Redes de transistores. 5. SPICE. 6. EDA. 7. ASTRAN. 8. Microeletrônica. I. Reis, Ricardo Augusto da Luz. II. Título.

“C makes it easy to shoot yourself in the foot;
C++ makes it harder, but when you do, it blows away your whole leg”
— BJARNE STROUSTRUP

AGRADECIMENTOS

Gostaria agradecer primeiramente à minha família: minha esposa **Angelina** pelo apoio na decisão de fazer o doutorado e meu filho **Daniel** pelo incentivo que faltava para termina-lo. Sem o carinho e amor destes dois, tenho certeza que não teria chegado até aqui. Sou grato aos meus pais **Adriel** e **Nair** por todo apoio que sempre me deram, pela ótima educação, carinho, enfim, por tudo o que sou hoje. As minhas irmãs **Adriana** e **Luciana** cujos feitos alcançados com seus estudos serviram como referência durante boa parte da minha formação.

Sou grato ao meu orientador **Ricardo Reis** pela oportunidade que me foi dada. Ao **Cristiano Lazzari**, **Guilherme Flach** e **Renato Hentschke** por todas suas contribuições técnicas. Aos meus colegas de laboratório que estiveram literalmente sempre ao meu lado e que me deram todo o apoio desde o início do doutorado: **Glauco Santos**, **Cristina Meinhardt**, **Gracieli Posser** e **Felipe Pinto**. Ao **Matheus Moreira**, **Michel Arendt** e **Ney Calazans** cuja simbiose entre nossos trabalhos foi de extrema ajuda nesta reta final.

Por fim, agradeço aos bolsistas de IC **Felipe Nesello** e **Charles Leonardt**, que de uma forma ou de outra colaboraram com esta tese.

RESUMO

Fluxo de síntese física baseado em *standard cells* tem sido utilizado na indústria e academia já há um longo período de tempo. Esta técnica é conhecida por ser bastante confiável e previsível uma vez que a mesma biblioteca de células, que foi devidamente validada e caracterizada, pode ser utilizada em diferentes projetos. No entanto, há uma série de otimizações lógicas e elétricas para problemas como: redução do consumo estático, circuitos assíncronos, SEU, NBTI, DFM, etc. que demandam a existência de células inexistentes em bibliotecas tradicionais. O projeto do leiaute destas células é usualmente feito a mão, o que pode dificultar a adoção e desenvolvimento de novas técnicas. Neste trabalho foi desenvolvido uma ferramenta para síntese automática do leiaute de redes de transistores chamada ASTRAN. Esta ferramenta suporta geração de células irrestrita quanto ao tipo da rede de transistores, incluindo lógica não-complementar, auxiliando no desenvolvimento de circuitos otimizados com menor área, número de transistores, conexões, contatos e vias. Através da utilização de uma nova metodologia para compactação do leiaute com programação linear mista com inteiros (MILP), foi possível compactar eficientemente as geometrias das células simultaneamente em duas dimensões, além de lidar com regras de projeto condicionais existentes em tecnologias abaixo de $130nm$. ASTRAN conseguiu obter ganhos de produtividade uma ordem de grandeza superior ao do projeto exclusivamente manual, necessitando de apenas 12h para gerar células com até 44 transistores. Na comparação com *standard cells* comerciais - considerado o pior caso uma vez que o ganho estaria justamente em gerar células inexistentes nestas bibliotecas ou então utilizar a ferramenta para obter um leiaute inicial antes de otimizá-lo a mão - o resultado foi bastante próximo, sendo que 71% das células geradas com o ASTRAN apresentaram exatamente a mesma área.

Palavras-chave: Síntese física, compactação do leiaute em 2D, células lógicas, redes de transistores, SPICE, EDA, ASTRAN, microeletrônica.

Automatic Layout Synthesis of Transistor Networks

ABSTRACT

Cell library-based synthesis flows for ASICs is one of the most used methodologies in both industry and academia for design of VLSI. It is known to be very reliable and predictable since the same cell library can be characterised and used in several different designs. However, there is a number of logic and electric optimizations for problems like: leakage reduction, asynchronous circuits, SEU, NBTI, DFM, etc. that demands the development of new cells. These cell layouts are usually designed by hand, which can limit the adoption and development of promising techniques. This work presents the development of a tool for automatic synthesis of transistor networks called ASTRAN. It can generate cell layout with unrestricted cell structure, including non-complementary logic cells, supporting the developing of optimized circuits with smaller number of transistors, connections, contacts and vias. By using a new methodology for simultaneous two-dimensional (2D) layout compaction using mixed integer linear programming (MILP), we were able to support most of the conditional design rules that applies to technology nodes bellow $130nm$, while producing as result dense cell layouts. We demonstrate that ASTRAN can generate layouts with a very smal area overhead compared to commercial standard-cells and can improve productivity in one order of magnitude when compared to the manual design of the cells. Gates containing up to 44 transistors were generated in less than 12h of run-time.

Keywords: physical synthesis, 2D layout compaction, logic cells, transistor networks, SPICE, EDA, ASTRAN, microelectronics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC	Circuito Integrado de Aplicação Específica (*Application Specific Integrated Circuit*)
CAD	Projeto Auxiliado por Computador (*Computer-Aided Design*)
CI	Circuito Integrado
CIF	Formato Intermediário Caltech (*Caltech Intermediate Format*)
DFM	Projeto para Manufaturabilidade (*Design for Manufacturability*)
GDSII	Sistema de Base de Dados Gráficos II (*Graphic Database System II*)
CMOS	Semicondutor Metal-Óxido Complementar (*Complementary Metal-Oxide Semiconductor*)
E/S	Entrada e Saída
GND	Terra ou suprimento de energia negativo (*Ground*)
ILP	Programação Linear com Inteiros (*Integer Linear Programming*)
LEF	Formato para Compartilhamento de Biblioteca (*Library Exchange Format*)
MST	Menor Árvore de Expansão (*Minimum Spanning Tree*)
MRST	Menor Árvore Retilínea de Steiner (*Minimum Rectilinear Steiner Tree*)
NBTI	Instabilidade de Temperatura por Polarização Negativa (*Negative Bias Temperature Instability*)
OPC	Correção por Proximidade Ótica (*Optical Proximity Correction*)
PO	Parte Operativa
RTL	Nível de Transferência entre Registradores (*Register Transfer Level*)
SA	Resfriamento Simulado (*Simulated Annealing*)
SAT	Problema de satisfazibilidade booleana (*Satisfiability*)
SOI	Silício Sobre Isolante (*Silicon Over Insulator*)
SPICE	Programa de simulação com ênfase em Circuitos Integrados (*Simulation Program with Integrated Circuit Emphasis*)
SPT	Árvore de Menor Caminho (*Shortest Path Tree*)
TA	Aceitação por Limiar (*Threshold Accepting*)

VDD Suprimento de energia positivo

VHDL Linguagem de Descrição de Hardware VHSIC (*VHSIC Hardware Description Language*)

VLSI Integração em Muito Larga Escala (*Very Large Scale Integration*)

SUMÁRIO

1 INTRODUÇÃO

Ferramentas de CAD são utilizadas em um número cada vez maior de áreas da computação com o objetivo de aumentar a produtividade. No projeto de circuitos VLSI, seu uso tem crescido em importância devido ao aumento da complexidade dos dispositivos e à necessidade de obter produtos que atendam ao *time-to-market*.

Devido à busca do mercado pela produção de dispositivos com um desempenho cada vez maior e capaz de efetuar um maior número de operações, fábricas de semicondutores se esforçam na tentativa de produzir circuitos integrados com transistores de tamanho cada vez menor, mais rápidos, e com menor consumo de potência. Graças às melhorias feitas no processo de fabricação dos semicondutores, atualmente existem circuitos comerciais que contêm bilhões de transistores dentro de um único *chip*, como indica a Figura 1.1. Por outro lado, cada melhoria no processo de fabricação, cria um novo conjunto de regras para projeto de circuitos integrados e o uso de ferramentas automáticas de EDA pode diminuir o tempo para refazer todo o leiaute da biblioteca de células.

1.1 Motivação

O fluxo de síntese baseado em biblioteca de *standard cells* para circuitos integrados de aplicação específica (ASICs) (mostrado na Figura 1.2) é uma das metodologias mais utilizadas na indústria e academia para projetos de circuitos VLSI. Ele é conhecido por ser relativamente confiável e previsível uma vez que a biblioteca de células pode ser caracterizada e utilizada em diferentes projetos. No entanto, há uma série de otimizações lógicas e elétricas para problemas como: circuitos assíncronos (KARMAZIN; OTERO; MANOHAR, 2013), analógicos (VAZQUEZ et al., 2012), redução do consumo estático

Figura 1.1: Aumento do número de transistores nos processadores Intel ao longo dos anos (INTEL, 2014).

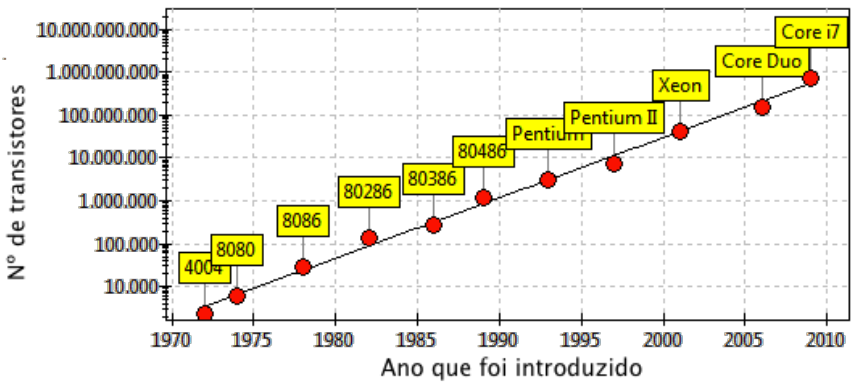
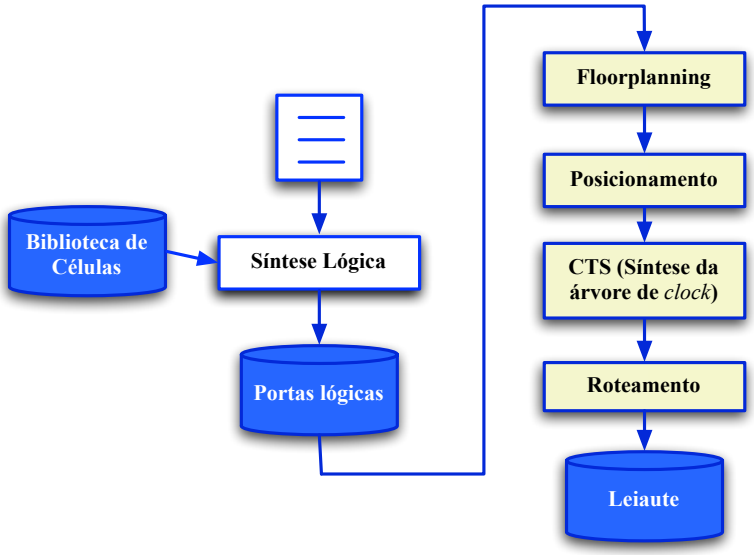


Figura 1.2: Fluxo de síntese física tradicional de circuitos (POSSER; REIS, 2011).



(LAZZARI; ZIESEMER; REIS, 2009), SEU (LAZZARI, 2007), NBTI (VATTIKONDA; WANG; CAO, 2006), DFM, etc. que demandam a existência de células inexistentes em bibliotecas tradicionais. O projeto do leiaute destas células é usualmente feito a mão, o que pode dificultar a adoção e desenvolvimento de novas técnicas.

Uma biblioteca de *standard cells* costuma ter entre entre 100 e 200 células com funções diferentes. Examinando a quantidade de funções lógicas diferentes que é possível criar utilizando até 4 transistores em série nos planos P e N, totalizariam 3503 células, como mostra a Tabela 1.1 (DETJENS et al., 1987). Considerando 3 dimensionamentos (que é um número bastante conservador) para cada uma destas células, resultaria em 10.509 células lógicas, número muito maior que o encontrado em qualquer biblioteca de *standard cells*.

Automação do projeto de leiaute é um desafio já ha muito tempo. Em nodos de tecnologia menores que $130nm$, o número de regras de projeto aumentou consideravelmente devido a problemas de litografia. A complexidade da geração de leiaute tem crescido como consequência, o que demanda novas técnicas para a geração de leiautes densos.

Trabalhos anteriores sobre migração de leiaute (FU et al., 2009; SAID et al., 2011) e síntese de células (ZIESEMER; LAZZARI; REIS, 2007; IIZUKA, 2007; GURUSWAMY et al., 1997) não suportam geração para nodos de tecnologia abaixo de $130nm$. CellTK (KARMAZIN; OTERO; MANOHAR, 2013) suporta até $90nm$, mas utiliza um fluxo próprio sem utilizar *standard cells*. Nangate (NANGATE LIBRARY CREATOR, 2014) e Prolific (PROGENESYS, 2011) (adquirida em 2011 pela ARM) possuem ferramentas para síntese de células capazes de produzir leiautes densos em tecnologias de até $28nm$ mas não disponibilizam informações sobre algoritmos e técnicas utilizadas.

1.2 Objetivos

Este trabalho tem como objetivo pesquisar técnicas de síntese automática do leiaute de redes de transistores, incluindo o desenvolvimento de uma ferramenta chamada ASTRAN. Atualmente ASTRAN é capaz de gerar leiautes para tecnologias de $65nm$ (comercial) e $45nm$ (acadêmica) a partir da descrição da redes de transistores das células em

Tabela 1.1: Número de funções possíveis usando um número máximo de transistores PMOS e NMOS em série (DETJENS et al., 1987).

		Número de transistores PMOS em série				
		1	2	3	4	5
Número de transistores NMOS em série	1	1	2	3	4	5
	2	2	7	18	42	90
	3	3	18	87	396	1677
	4	4	42	396	3503	28435
	5	5	90	1677	28435	125803

formato SPICE. Um compactador de leiaute em duas dimensões foi desenvolvido com o objetivo de gerar células com densidade próxima a de leiautes feitos a mão por projetistas experientes. Detalhes do fluxo de geração são apresentados e as melhorias com relação a implementação anterior para tecnologia de $350nm$ (ZIESEMER; LAZZARI; REIS, 2007) são destacadas.

1.3 Organização do Trabalho

Este trabalho está organizado como segue. O Capítulo 2 apresenta exemplos de problemas onde a síntese utilizando bibliotecas de *standard cells* pode limitar a qualidade dos circuitos devido à necessidade de células customizadas.

Um estudo sobre os algoritmos existentes na literatura para síntese de células, incluindo o estado-da-arte, é apresentado no Capítulo 3.

O Capítulo 4 apresenta o desenvolvimento do módulo para geração do leiaute de células CMOS da ferramenta ASTRAN. O restante da ferramenta, incluindo o fluxo para síntese de circuitos, é detalhado no Apêndice A.

Os resultados obtidos são apresentados no Capítulo 5 e por fim o Capítulo 6 apresenta as conclusões deste trabalho.

2 MOTIVAÇÃO PARA GERAÇÃO AUTOMÁTICA DE REDES DE TRANSISTORES

Este capítulo apresenta exemplos de técnicas e otimizações que demonstram a importância da geração de células customizadas sob demanda, frente a utilização de um fluxo tradicional de síntese utilizando somente as células existentes em bibliotecas de *standard cells*.

2.1 Introdução

A constante redução no tamanho dos transistores trazida por cada nova tecnologia de fabricação também traz novos desafios que devem ser mitigados de forma a aumentar o desempenho, qualidade e também reduzir o consumo de potência. Estes desafios enfatizam a necessidade de novas células além das existentes nas bibliotecas de *standard cells* (células padrão) de forma que as ferramentas de síntese tenham mais opções para lidar com estes problemas.

O estudo apresentado neste capítulo visa mostrar de que forma a utilização de células geradas automaticamente pode apoiar a utilização de técnicas existentes, mas que tem sua aplicação limitada devido a necessidade de fazer o projeto individual do leiaute das células, manualmente.

2.2 Mapeamento Utilizando Portas Complexas

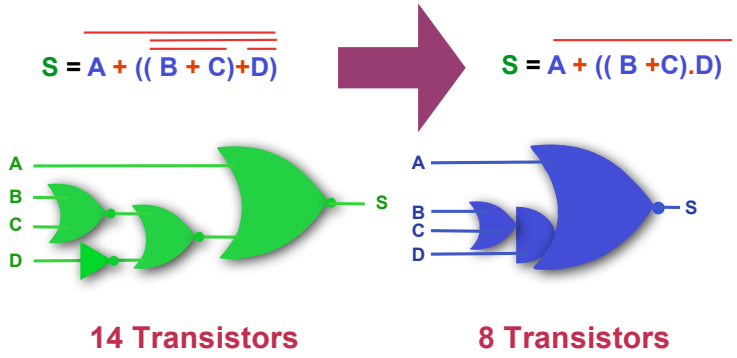
Uma forma de reduzir a quantidade de transistores em um circuito é utilizando *static CMOS complex gates* (SCCG) (POSSANI et al., 2013; REIS, 2011), desta forma funções lógicas de várias entradas podem ser implementadas utilizando apenas uma porta ao invés de várias como mostra a Figura 2.1.

A redução no número de transistores, além de permitir uma redução da área dos circuitos, pode ter impacto também na redução do consumo estático de energia uma vez que corrente de fuga costuma ter relação direta com o número de transistores.

A utilização de portas complexas colabora também na redução do congestionamento durante o roteamento detalhado do circuito. Isto porque as conexões necessárias já estão prontas dentro da célula (normalmente feitas utilizando *poly* e metal 1). Outro ganho é que conexões menores também tendem a ter menor atraso.

Devido à grande quantidade de combinações de células complexas que podem ser implementadas, bibliotecas comerciais de *standard cells* possuem usualmente apenas uma pequena quantidade destas células.

Figura 2.1: Mapeamento utilizando porta complexa (REIS, 2011).



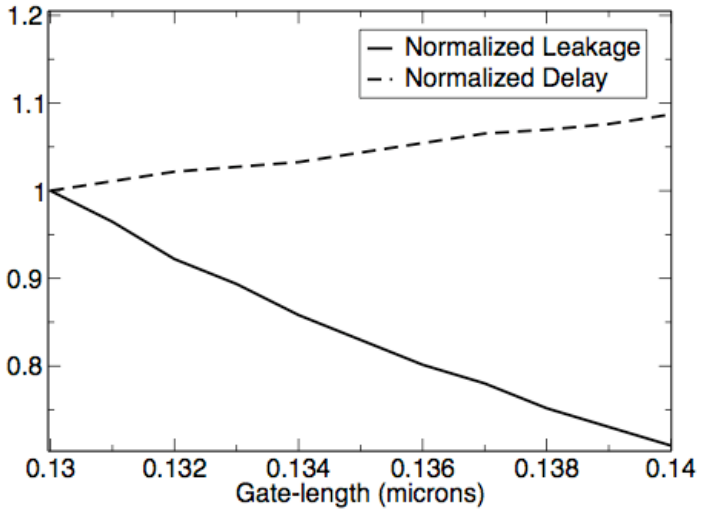
2.3 Circuitos Assíncronos

Ao mesmo tempo em que a tecnologia de fabricação de semicondutores evolui ao território dos nano-CMOS, técnicas de circuitos assíncronos ganham relevância devido a sua habilidade de cobrir problemas que são difíceis de resolver com o paradigma síncrono (BEEREL; OZDAG; FERRETTI, 2010; MARTIN; NYSTROM, 2006). Na última década houve um desenvolvimento substancial de técnicas e ferramentas para o projeto semi-customizado de assíncronos. Entre os considerados estado-da-arte, é possível citar: (BEEREL; DIMOU; LINES, 2011; BARDSLEY; TARAZONA; EDWARDS, 2009; REESE; SMITH; THORNTON, 2012; THONNART; BEIGNE; VIVET, 2012; CORTADELLA et al., 2006; NOWICK; SINGH, 2011).

Por outro lado, apesar destes trabalhos explorarem otimizações para projetos assíncronos, todos eles possuem como premissa a existência de um conjunto de portas no nível de célula. No entanto, este conjunto de portas possuem vários componentes específicos que não estão disponíveis em bibliotecas comerciais de *standard cells*, restringindo severamente a adoção destas técnicas e ferramentas, uma vez que projetistas precisam implementar a mão o conjunto de portas necessário. Entre os componentes mencionados que são tipicamente requeridos em projetos assíncronos estão portas: *Null Convention Logic* (NCL), *C-Elements*, *precharge half-buffers* e elementos de exclusão mútua (BEEREL; OZDAG; FERRETTI, 2010; MARTIN; NYSTROM, 2006; FANT; BRANDT, 1996). A maior parte destes componentes, incluindo portas NCL e C-Elements, podem ser construídos utilizando as células normalmente disponíveis nas bibliotecas de *standard cells*, no entanto, isto não é eficiente e pode introduzir problemas elétricos nos circuitos (MOREIRA; C. H. M. OLIVEIRA, 2013).

Projetar uma biblioteca de células é uma tarefa trabalhosa e que requer alto grau de experiência. Neste sentido, ter um fluxo que automatiza este processo é de grande importância. Alguns dos poucos trabalhos encontrados na literatura são: ASCEnD e cellTK. Enquanto o ASCEnD (MOREIRA et al., 2011) não possui suporte a automatização da geração do leiaute das células, cellTK (KARMAZIN; OTERO; MANOHAR, 2013) emprega um fluxo próprio de síntese não compatível com *standard cells* (o que o torna incompatível com ferramentas e técnicas existentes para a área de assíncronos, como por exemplo: dimensionamento de portas lógicas). Os autores reportam uma penalidade de 51% em área, 12% em energia e 9% em atraso quando comparado com circuitos *full-custom*.

Figura 2.2: Corrente de fuga e *delay* normalizados para um transistor NMOS (KIM et al., 2003).



2.4 Técnicas para Redução do Consumo Estático

Uma técnica que pode ser utilizada para redução do consumo estático de potência em circuitos VLSI é chamade de *gate length biasing* (GUPTA et al., 2004). Ela consiste em ajustar o comprimento do canal de alguns transistores com a finalidade de reduzir a corrente de fuga. Isto é possível pois a corrente de fuga de *sub-threshold*, ao contrário do *delay*, aumenta com a redução do comprimento do canal, conforme ilustra a Figura 2.2. Desta forma, é possível reduzir significativamente a corrente de fuga total do circuito, com pouco ou nenhum impacto ao atraso final, apenas redimensionando transistores pertencentes ao caminho crítico.

Em (LAZZARI; ZIESEMER; REIS, 2009), Lazzari desenvolveu uma metodologia para encontrar e redimensionar transistores fora do caminho crítico sem produzir qualquer penalidade ao atraso do circuito. Na sua metodologia, leiautes são produzidos após o comprimento do canal ter sido definido. Nos seus experimentos ele concluiu que uma aumento de 10% produz o melhor *trade-off* entre área e corrente de fuga. A ferramenta Synopsys Pathmill (SYNOPSYS, 2011) foi utilizada para calcular o atraso de cada caminho do circuito. A Figura 2.3 mostra o resultado obtido antes e depois do dimensionamento, onde foi obtido uma redução do consumo estático de 76% do valor original. É possível notar também um número maior de caminhos com atraso próximo do limiar.

2.5 Técnica para Redução do Atraso Devido a NBTI

Acredita-se que o efeito NBTI (*Negative Bias Temperature Instability*) é causado por ligações Si-H quebradas, induzidas por buracos positivos no canal dos transistores. De acordo com (VATTIKONDA; WANG; CAO, 2006), para um transistor PMOS, há duas fases para o NBTI, dependendo da sua condição de polarização, como mostra a Figura 2.4. Fase I: fase de estresse; *traps* positivos de interface acumulam durante o tempo de estresse com H difundindo para o *gate*. Fase II: fase de recuperação; buracos não estão presentes no canal e H difunde de volta, corrigindo as ligações Si-H que foram quebradas.

Uma vez que NBTI muda Vth com o tempo, diminuindo a velocidade dos transistores, violações de atraso podem ocorrer após um tempo contínuo de operação sob condições

Figura 2.3: Atraso dos caminhos antes e depois do dimensionamento do comprimento dos *gates* segundo técnica desenvolvida em (LAZZARI; ZIESEMER; REIS, 2009).

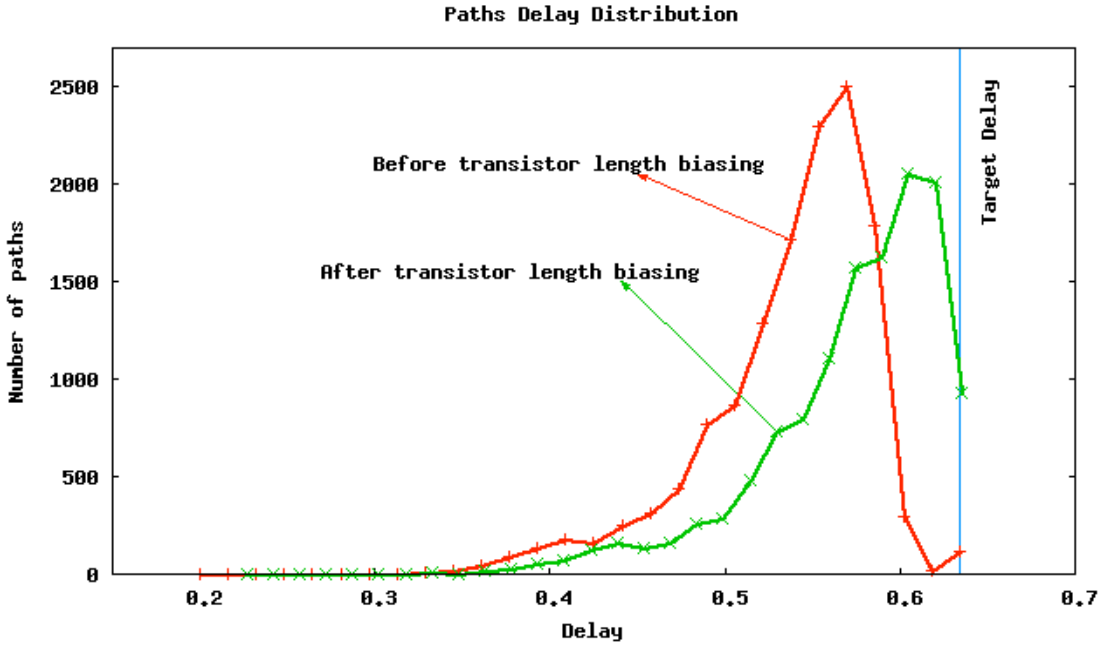


Figura 2.4: As duas fases do PMOS NBTI (VATTIKONDA; WANG; CAO, 2006).

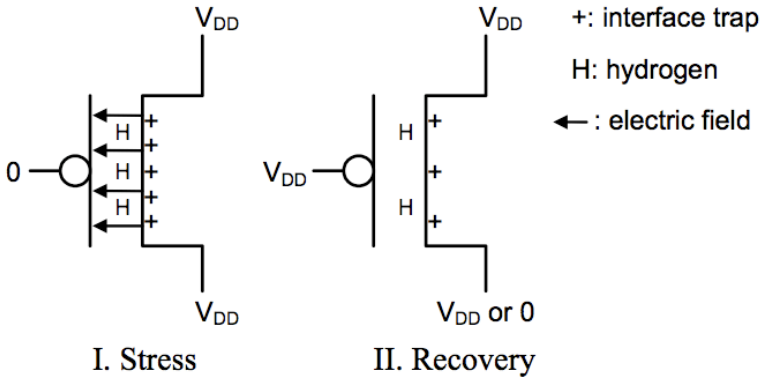


Figura 2.5: Sobre-dimensionamento da largura do *gate* dos transistores para compensar o efeito de NBTI ao longo dos anos (VATTIKONDA; WANG; CAO, 2006).

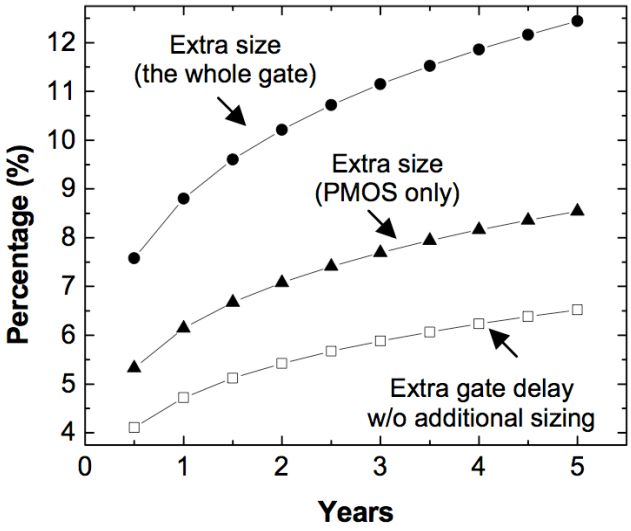


Tabela 2.1: Resultado da comparação entre dimensionamento usando células de uma biblioteca *standard cell* (SC) e dimensionamento usando programação geométrica (GP) em $45nm$ minimizando atraso e com restrição de área (POSSER et al., 2011).

	Power (μW)			**Timing** (ps)			**Area** (μm^2)		
	SC sizing	**GP sizing**	**R (%)**	**SC sizing**	**GP sizing**	**R (%)**	**SC sizing**	**GP sizing**	**R (%)**
C432	22.2	22.4	**-0.9**	718	666	**7.3**	210.4	210.4	**0.0**
C499	58.3	58.4	**-0.2**	750	651	**13.1**	536.4	536.4	**0.0**
C1908	33.6	33.7	**-0.3**	472	425	**10.0**	304.3	304.3	**0.0**
C880	31.4	31.1	**1.1**	451	330	**26.8**	281.0	277.4	**1.3**
apex1	239.8	239.5	**0.1**	673	504	**25.2**	2304	2296	**0.4**
apex2	527.1	523.6	**0.7**	863	650	**24.7**	5180	5145	**0.7**
apex3	254.3	251.9	**0.9**	687	507	**26.3**	2441	2413	**1.2**
apex5	264.6	258.3	**2.4**	662	431	**34.9**	2512	2446	**2.6**
Avg.	**178.9**	**177.3**	**0.5**	**660**	**521**	**21.0**	**1721**	**1704**	**0.8**

de estresse. Esta degradação precisa ser avaliada e tratada durante a fase de projeto para garantir a qualidade e longevidade do produto.

Dimensionamento de transistores tem sido largamente utilizado para resolver problemas de performance. Uma vez que o efeito NBTI degrada somente a performance dos transistores PMOS, sobre-dimensionar estes transistores ao longo do caminho crítico pode produzir o mesmo resultado de dimensionar todos os transistores, mas com menor área (VATTIKONDA; WANG; CAO, 2006). A Figura 2.5 mostra que pequenos incrementos de tamanho nos transistores PMOS, tão baixos quanto 8,5%, podem potencialmente compensar aumento no atraso por NBTI por até 5 anos de operação. Este acréscimo pode ser ainda menos significativo uma vez que somente os transistores pertencentes aos caminhos críticos sejam ajustados. A aplicação desta técnica exige o desenvolvimento de células otimizadas que normalmente não estão disponíveis em bibliotecas de *standard cells*.

2.6 Dimensionamento de Transistores ou Células Lógicas

Bibliotecas *standard cell* possuem normalmente poucos dimensionamentos disponíveis para cada célula lógica. Enquanto células muito utilizadas - tais como *buffers* e inversores - costumam ter maior quantidade de dimensionamentos (em geral até 10), células de maior tamanho e complexidade - como somadores e *flip-flops* - possuem poucas versões de si mesma (por volta de 3). Esta limitação pode restringir a quantidade de otimizações feitas no circuito com relação a uma geração onde as células/transistores podem ser dimensionados de forma contínua ou possuem maior granularidade.

Posser (POSSER; REIS, 2011) desenvolveu uma metodologia para dimensionamento de portas lógicas de circuitos integrados utilizando técnicas de otimização baseadas em Programação Geométrica (PG) (DUFFIN; PETERSON; ZENER, 1967). Os resultados obtidos na comparação com um fluxo *standard cell* padrão são mostrados na Tabela 2.1, onde se alcançou uma redução média de 21% no atraso do circuito, sem prejuízo significativo da potência e mantendo a área como restrição. Num outro experimento, mantendo o atraso como restrição, o dimensionamento permitiu uma redução média de 27% em potência e de 28% em área. Isto mostra o potencial da geração do leiaute de células otimizadas eletricamente.

3 GERAÇÃO AUTOMÁTICA DE LEIAUTE

Este capítulo aborda os principais métodos encontrados na literatura para geração de leiautes, com atenção especial à geração de células CMOS.

3.1 Introdução

A forma tradicional de projetar um circuito integrado requer que o projetista primeiramente defina uma biblioteca de células lógicas e um modelo comportamental da funcionalidade de um circuito integrado. Estas bibliotecas tipicamente incluem portas lógicas fundamentais como inversor, NOR, NAND, XOR, mas também têm elementos sequenciais como *latches* e *flip-flops*. Também é encontrado nestas bibliotecas diferentes versões de leiautes para uma mesma célula, diferenciando-se apenas no dimensionamento dos seus transistores. Bibliotecas de *standard cells* típicas costumam ter ao menos dois dimensionamentos de células para cada porta lógica: uma para menor área e consumo de energia, e uma para melhor desempenho. Isto é feito para que células diferentes, que implementam a mesma lógica, possam ser usadas para atender diferentes requisitos de área, potência e atraso.

Normalmente, bibliotecas de células são feitas de forma manual por um projetista experiente. Por este processo ser extremamente demorado e suscetível a erros, diversos trabalhos foram feitos na tentativa de automatizar parcialmente, ou totalmente, este procedimento.

3.2 Geração de Leiaute Segundo a Metodologia TRANCA / UFRGS

Diversas ferramentas para síntese física de CIs tem sido desenvolvidas no GME desde 1981 (SUSIN, 1981; CARRO, 1989; LUBASZEWSKI, 1990; MORAES, 1990), como mostra a linha do tempo da Figura 3.1 para a metodologia TRANCA. Por outro lado, muitas destas ferramentas não suportam mais as regras de projeto exigidas atualmente pelas *foundrys* para fabricação de circuitos integrados.

Em (WILKE et al., 2002) foi desenvolvida uma ferramenta para verificação temporal com base na simulação de vetores flutuantes e traço de caminhos para identificar o atraso crítico de circuitos combinacionais. Este trabalho, em conjunto com outros que foram desenvolvidos sobre este tema receberam o nome de TICTAC.

Um posicionador chamado MANGO PARROT foi apresentado em (HENTSCHKE, 2002). Esta ferramenta produz como resultado um posicionamento relativo das células ao longo das bandas do circuito com o objetivo de reduzir o comprimento total das conexões. Entre as características desta ferramenta está a capacidade de realizar o posicionamento

Figura 3.1: Linha do tempo das ferramentas de síntese física segundo a metodologia TRANCA.

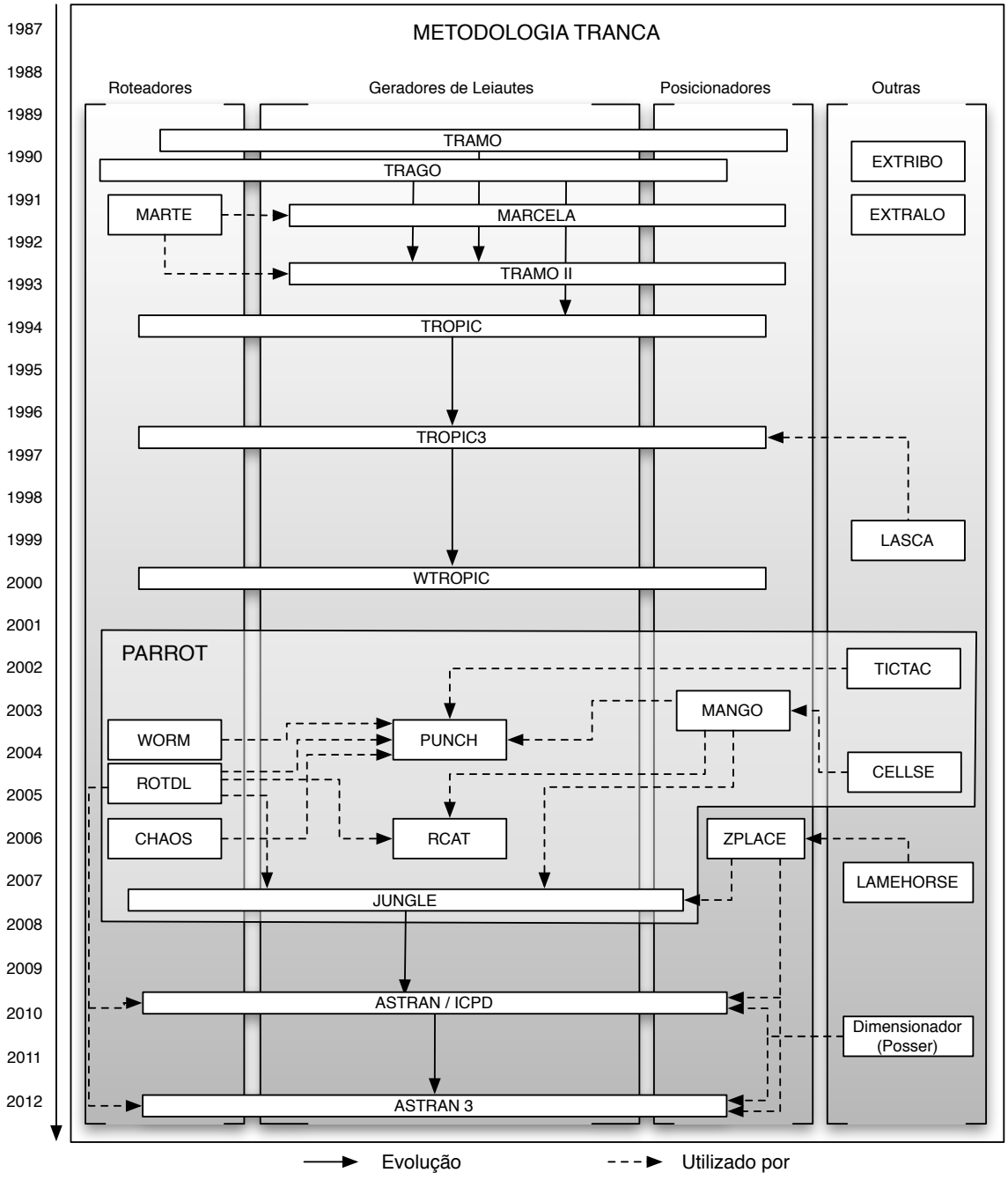
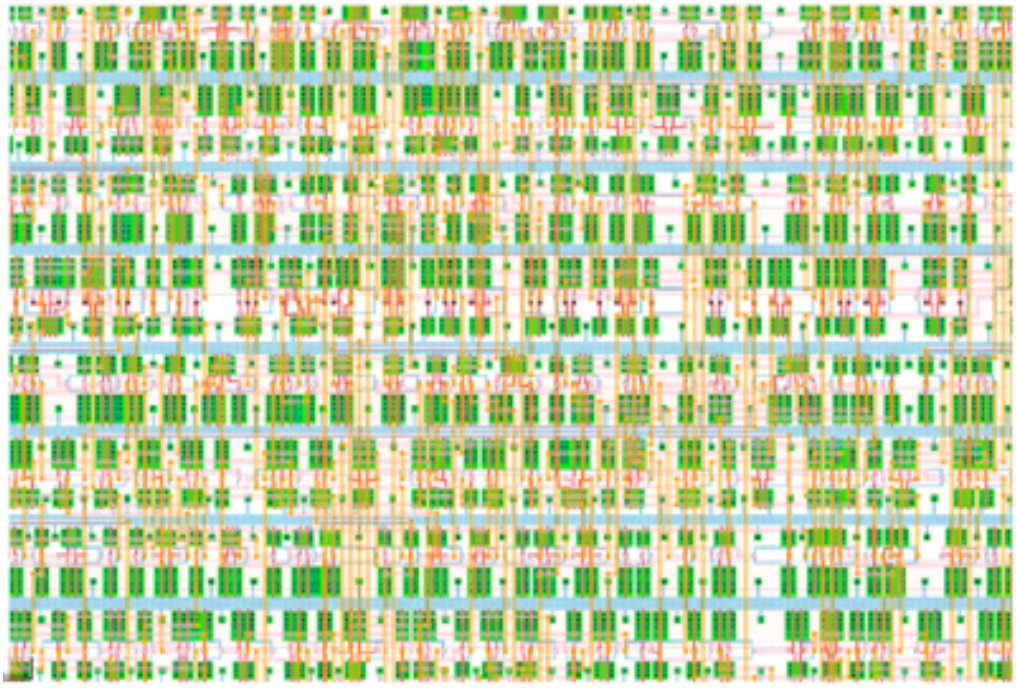


Figura 3.2: Exemplo de leiaute produzido pela ferramenta PUNCH (LAZZARI, 2003).



sem a existência prévia de uma biblioteca com o tamanho das células. A área ocupada por cada célula é estimada de acordo com a quantidade de transistores existentes e a tecnologia utilizada.

Em (LAZZARI, 2003), foi desenvolvida a ferramenta PARROT PUNCH com objetivo de aproveitar as ferramentas existentes na época e criar um fluxo de geração de leiaute com foco na redução de atraso e potência. A principal característica de PUNCH é a geração de leiautes *on-the-fly*, sem a necessidade de uma biblioteca de células. O leiaute é gerado como se o circuito inteiro fosse uma célula gigante, sem hierarquia, como mostra a Figura 3.2.

Em conjunto com a ferramenta PUNCH, foi desenvolvido um roteador em malha (*maze router*) chamado WORM para realizar as interconexões do circuito. As duas ferramentas possuem uma forte integração. Uma falha do roteador em completar o roteamento do circuito, faz com que o gerador insira automaticamente espaços nas regiões de conflito para a próxima tentativa do roteador.

O fluxo de geração de leiaute criado com o uso destas ferramentas recebeu o nome de PARROT. Diversas ferramentas foram desenvolvidas e integradas à este fluxo.

Um novo roteador mais robusto chamado ROTDL foi desenvolvido em (FLACH; HENTSCHKE; REIS, 2004). Apesar de não ter ainda uma integração tão forte com o gerador de leiautes, esta ferramenta é capaz de rotear em menos tempo que o WORM, circuitos de mesma dimensão.

Como a estimativa do tamanho das células feito pelo posicionador MANGO era muito simples e tinha uma grande margem de erro, em (ZIESEMER et al., 2006) foi apresentado um gerador de estimativas de tamanho de células chamado CELLSE. Com esta ferramenta foi possível estimar as dimensões das células de forma mais automatizada e precisa. Os leiautes produzidos por PUNCH apresentaram como resultado um leiaute com uma relação de aspecto mais previsível, com um menor número de redes não roteadas e comprimento médio das conexões.

O roteador ChAOS (SANTOS; JOHANN; REIS, 2006) surgiu como uma alternativa para roteamento convergente e ágil uma vez que é implementado com algoritmos de baixa complexidade. A convergência é garantida pela inserção de espaços entre as bandas. A agilidade é garantida por um pré-planejamento global baseado em árvores de expansão e um roteamento detalhado baseado no *Greedy Channel Router*. Os resultados mostraram que apesar de apresentar tempos de execução muito inferiores ao WORM e ROTDL, houve um aumento médio na área dos circuitos de 7,2% em comparação com este último.

Em (MEINHARDT, 2006) foi desenvolvido um gerador de leiautes regulares baseado em matrizes de células chamado RCAT. Esta ferramenta foi integrada ao fluxo PARROT e tinha como principal característica a previsibilidade das suas características elétricas.

Dando início às pesquisas na UFRGS na área de circuitos 3D, Hentschke (HENTSCHKE, 2007) desenvolveu um posicionador quadrático com refinamento local iterativo utilizando o método de *Threshold Accepting* (DUECK; SCHEUER, 1990) para redução de *wirelength* (comprimento das conexões) 3D com observância do caminho critico. Este posicionador, que recebeu o nome de ZPLACE, também realiza posicionamento em circuitos 2D e é capaz de suportar circuitos com centenas de vezes mais células do que o MANGO PARROT.

Em (SAWICKI et al., 2007), a ferramenta LAMEHORSE foi desenvolvida para realização do particionamento e posicionamento dos PADs e pinos de I/O em circuitos 3D. O posicionamento destes elementos assume fundamental importância para o correto funcionamento dos algoritmos de posicionamento quadráticos uma vez que são eles que servirão de suporte para as células ao longo das diferentes *tiers* (camada de *chips*). Além disto, um correto posicionamento pode contribuir significativamente para redução do wirelength e atraso das interconexões do circuito.

Em (ZIESEMER; REIS, 2007) foi desenvolvido um compilador de partes operativas e gerador automático de células lógicas chamado JUNGLE PARROT. Foi definido um formato de arquivo para descrição estrutural dos circuitos de parte operativa, o qual era interpretado para geração dos leiautes. O posicionamento dos módulos (banco de registradores, somador, multiplicador, etc.) era feito de forma regular, conforme um *template* previamente estabelecido. O roteamento era feito utilizando o ROTDL.

Nesta tese foi desenvolvida uma plataforma chamada ASTRAN para síntese automática de redes de transistores. Utilizou-se como ponto de partida a ferramenta JUNGLE PARROT a qual precisou ser adaptada para tornar-se compatível com o formato de *standard cells* e incluí-la num fluxo mais adequado para este tipo de circuito: com suporte a dimensionamento dos transistores, planejamento da planta baixa, posicionamento das interfaces do circuito, roteamento detalhado em metal 1, posicionamento incremental e interface gráfica com o usuário. A primeira versão do ASTRAN suportava compactação do leiaute em 1D e processos até $350nm$. Na sua terceira versão (a atual) foram inseridos suporte a compactação em 2D e geração para tecnologias até $65nm$ ($45nm$ se considerar o freePDK45 (FREEPDK45 DESIGN KIT, 2014)).

3.3 Métodos para Geração Automática de Leiautes de Células Lógicas

Lefebvre (LEFEBVRE; MARPLE; SECHEN, 1997), classificou os geradores de leiaute em 3 categorias distintas:

Geradores procedurais de leiaute podem ser baseados em alguma linguagem de descri-

ção ou simbólicos. O leiaute é descrito de uma forma independente de regras de desenho e o gerador transforma a descrição fornecida no leiaute da célula correspondente de acordo com as regras tecnológicas utilizadas. Cada célula necessita ter seu próprio código gerador e que precisa ser desenvolvido por um projetista experiente para uma melhor eficácia. Como desvantagem, geradores procedurais tendem a não serem amigáveis à mudanças drásticas na arquitetura da célula e tecnologias de interconexão.

Recompactação de leiautes pré-existentes provê uma solução mais elegante ao problema, uma vez que uma simples ferramenta genérica pode processar uma grande variedade de células. Entretanto, uma biblioteca pronta de células precisa ser previamente desenvolvida a mão e também possui a desvantagem de não tolerar muito bem mudanças grandes na arquitetura da célula bem como na sua tecnologia de fabricação. Por esta razão, esta técnica é utilizada mais frequentemente para alterações mais simples tais como: otimização de atraso em caminhos críticos, redução da área crítica (para aumentar o *yield*) e correção por proximidade ótica (OPC) (IIZUKA, 2007). Trabalhos que utilizam esta técnica incluem (FANG; ZHU, 2004; SAID; ABBAS; SHAHEIN, 2007).

Síntese de células é a geração do leiaute tendo como ponto de partida o *netlist* da rede de transistores de cada célula. Este método costuma ser completamente flexível quanto à arquitetura da célula e não requer nenhuma informação pré-existente do leiaute. Entretanto, o problema de criar leiautes competitivos em qualidade com os feitos-a-mão não é um problema trivial e a complexidade deste processo tem sido o maior empecilho para o seu sucesso comercial.

Dentre os três métodos, a síntese de células é a que oferece maior flexibilidade para suportar os avanços no estado-da-arte da síntese de circuitos digitais.

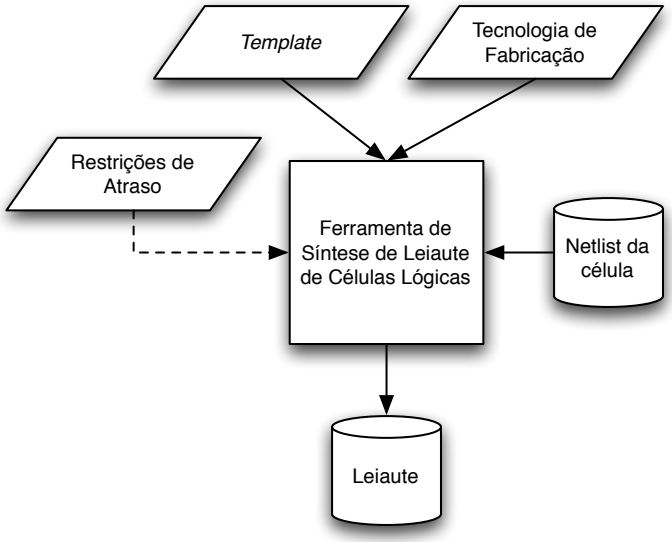
3.4 Síntese do Leiaute de Células

Síntese de células consiste em mapear uma rede de transistores dimensionados no leiaute de célula correspondente, de acordo com uma arquitetura de célula pretendida. Esta metodologia é ilustrada na Figura 3.3 e começa com o usuário provendo um *netlist* (especificação das redes), um *template* (modelo) para a biblioteca e a tecnologia de fabricação a ser utilizada. O *netlist* possui uma lista dos transistores que formam o circuito, com seus respectivos tamanhos, seus terminais de entrada/saída e interconexões. Opcionalmente, informações sobre atraso das conexões podem ser informados para fins de otimização do leiaute. O *template* descreve a forma física do leiaute das células tal como: altura, largura das linhas de alimentação, grade de roteamento, etc. De uma forma geral, o mesmo modelo costuma ser usado para construção de todas as células da mesma biblioteca.

A saída de um sistema para síntese de células é uma coleção de leiautes resultantes da solução sucessiva de três subproblemas: posicionamento dos transistores, roteamento e compactação.

O leiaute resultante costuma ser analisado por um projetista para verificar sua qualidade e, se necessário, fazer pequenos ajustes a mão para corrigir/melhorar suas características, antes da célula ser incluída em uma biblioteca.

Figura 3.3: Fluxo padrão para síntese de leiaute de células lógicas.



3.4.1 Especificação da Célula

O primeiro passo para a geração automática, é o projetista fornecer a especificação da célula. A especificação pode ser provida em 2 níveis de abstração diferentes: físico e lógico. O *netlist* lógico especifica a função lógica que a célula deve executar (Ex.: $S = A + B$, para uma porta OR) e não entra em detalhes da sua constituição interna (transistores, conexões, etc.). O *netlist* físico possui correspondência de um para um entre os componentes descritos e os implementados no leiaute final. Tanto o dimensionamento individual dos transistores quanto a arquitetura das conexões (*netlist*) são descritos, e por esta razão, é o preferido para ser utilizado como entrada em uma ferramenta de síntese de células lógicas. O formato de arquivo SPICE é um dos mais frequentemente utilizados para descrever redes de transistores em nível físico.

Para gerar circuitos a partir de um *netlist* lógico, é necessário primeiramente mapear a célula para um arranjo de transistores que realize a função especificada, obtendo-se assim o seu *netlist* físico. Há várias otimizações que podem ser feitas durante o mapeamento para obter células com diferentes características de consumo estático/dinâmico, *delay* e área. A ferramenta TABA (REIS; ROBERT; REIS, 1998) é um exemplo de ferramenta capaz de transformar o *netlist* lógico no físico automaticamente.

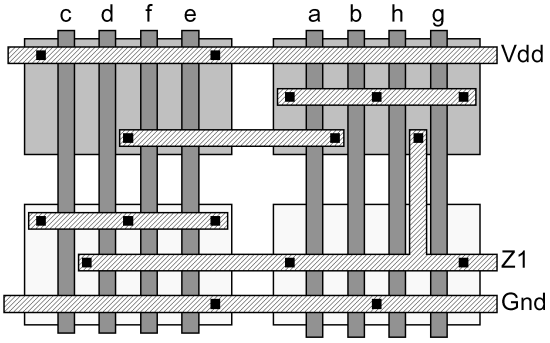
3.4.2 Estilos de Leiautes

Estilo de leiaute é a organização interna da célula e que normalmente serve como guia para todo o processo de síntese. Algumas das definições frequentemente encontradas no projeto de ferramentas de síntese de células são: regras para posicionamento de transistores P e N (tamanho, orientação, posição e quantidade de bandas), regras para roteamento entre transistores (camadas e regiões que podem ser utilizadas), posição das interfaces da célula, posição e forma das linhas de alimentação, etc.

Diversos estilos de leiautes visando a minimização da área e da complexidade dos algoritmos de posicionamento e roteamento através da definição de regiões específicas para alocação dos transistores P e N foram propostos na literatura. Alguns deles serão discutidos a seguir.

Uehara e vanCleemput (UEHARA; VANCLEEMPUT, 1981) propuseram o estilo de

Figura 3.4: Estilo de leiaute *linear-matrix* (1D) (RIEPE; SAKALLAH, 2003).



leiaute chamado *linear-matrix* para geração de circuitos de lógica CMOS complementar. Neste estilo, as células são formadas por duas linhas horizontais de transistores PMOS e NMOS paralelas à alimentação e com os polisilícios dos *gates* perpendiculares, como ilustrado na Figura 3.4. Agrupados desta forma, transistores com o mesmo sinal podem ser conectados diretamente com difusão ao invés de conexões com contatos e metais. Isto permite que as células tenham uma maior densidade de transistores e, ao mesmo tempo, melhorem suas características elétricas devido a uma menor capacitância de acoplamento entre os transistores.

Estilos baseados no *linear-matrix* são frequentemente referenciados como 1D (ou de uma dimensão) pois os transistores são posicionados sequencialmente, lado a lado, mudando apenas o seu ordenamento e orientação. As variações mais frequentes encontradas no estilo *linear-matrix* são quanto à posição e nível de metal utilizado nos canais de alimentação da célula, método de roteamento interno e posição dos pinos de entrada e saída.

O estilo 1-1/2D proposto em (REKHI; TROTTER; LINDER, 1995) define regiões para posicionamento dos transistores de forma similar ao *linear-matrix*, mas permite que transistores P possam ser posicionados na região de difusão N e vice-versa. Esta técnica permite um melhor aproveitamento de área principalmente nos casos de circuitos com grande discrepância no número de transistores PMOS e NMOS.

O estilo 2D permite arranjos vertical e horizontal de transistores. Enquanto em algumas abordagens são geradas múltiplas linhas de transistores 1D para formar o leiaute 2D como no exemplo da ferramenta CLIP (GUPTA; HAYES, 1997) mostrada na Figura 3.5 (a). Existem outras que não são baseadas em linhas de transistores e que permitem que os mesmos sejam posicionados de várias formas diferentes, como no exemplo da ferramenta TEMPO (RIEPE; SAKALLAH, 2003) mostrada na Figura 3.5 (b). Neste último caso, os transistores não obedecem nenhum estilo pré-determinado e o seu posicionamento é feito através de heurísticas que procuram por uma solução de menor custo. Este tipo de posicionador costuma ser mais frequentemente utilizado para circuitos analógicos e para famílias lógicas como *Cascade Voltage Switch Logic* (CVSL), *Pass Transistor Logic* (PTL) e *domino CMOS* onde não há muita regularidade entre as conexões dos transistores PMOS e NMOS, e nem equivalência entre seus *gates*.

3.4.2.1 Altura da Célula

Standard cells possuem usualmente uma banda de altura, sendo seus transistores todos posicionados entre as linhas de alimentação positiva e negativa. A grande maioria das ferramentas de síntese de leiaute de células lógicas, bem como ferramentas de síntese física para posicionamento e roteamento de circuitos, trabalham neste formato.

Figura 3.5: Estilos de leiautes 2D. (a) Posicionamento dos transistores em 1D com *double-height* (duas bandas de altura) (GUPTA; HAYES, 1997); (b) Estilo livre (RIEPE; SAKALLAH, 2003).

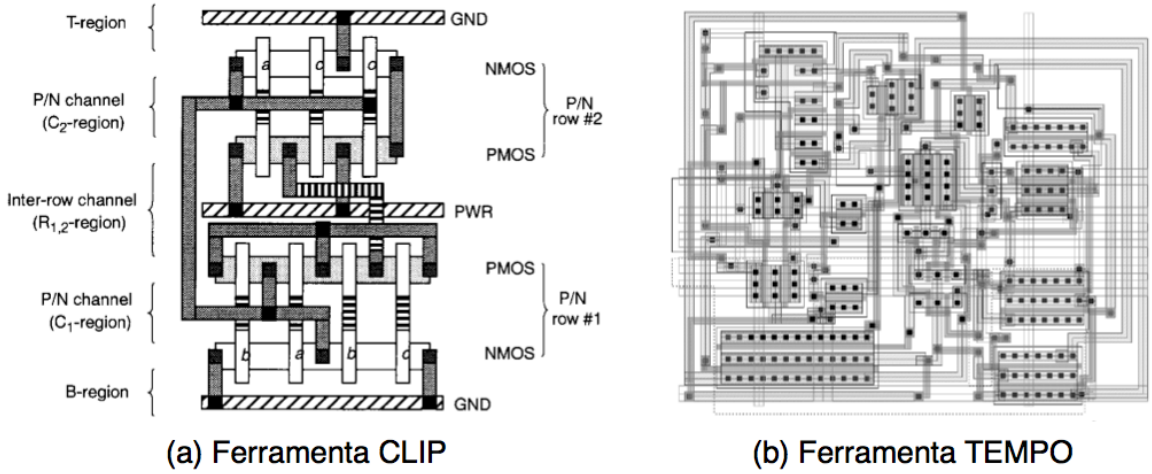
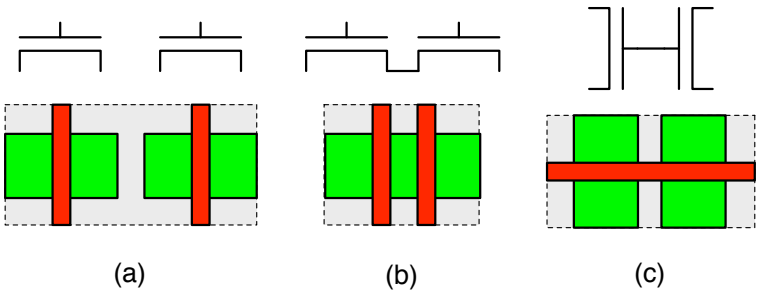


Figura 3.6: Exemplo de otimização elétrica e de área entre dois transistores: sem otimização (a), compartilhamento de difusão (b) e *gates* alinhados (c).



No entanto, em algumas aplicações específicas, como por exemplo partes operativas ou outros circuitos especializados, é comum a utilização de células com duas ou mais bandas de altura, como no exemplo mostrado na Figura 3.5 (a). Em alguns casos, a largura da célula é dada como restrição enquanto a sua altura é variável (IIZUKA, 2007). Este estilo de leiaute é chamado de *multi-row* (múltiplas bandas) e possui vantagem de possuir maior flexibilidade sobre o posicionamento e tamanho dos transistores. Ele oferece maior controle sobre a relação de aspecto do leiaute e também permite reduzir a densidade e comprimento das conexões internas.

3.4.3 Posicionamento dos Transistores

A área ocupada pela célula tem relação direta com o posicionamento de seus transistores porque existem diversas otimizações que podem ser feitas de acordo com o seu posicionamento. A Figura 3.6 mostra um exemplo de possíveis otimizações entre dois transistores. Caso estes transistores não possuam nenhum sinal em comum, eles precisam ser posicionados separadamente ocupando uma grande área como mostrado em (a). Caso eles possuam alguma rede em comum, otimizações visando minimizar a área ocupada e melhorar as características elétricas do circuito podem ser feitas. Este é o caso de (b), onde a fonte/dreno de um transistor pertence a mesma rede da fonte/dreno do outro transistor, e de (c), onde ambos compartilham o mesmo sinal de *gate*.

Efeitos como rotação, translação e espelhamento também podem ser explorados de forma a aproveitar melhor toda a área da célula. Entretanto, quanto maior o grau de liberdade sobre o posicionamento dos transistores, maior o espaço de soluções que o algoritmo de posicionamento deve pesquisar, o que reflete em um aumento da complexidade e do tempo de execução do algoritmo. Outro problema, é que uma distribuição arbitrária dos transistores ao longo do leiaute também costuma dificultar a etapa seguinte de roteamento da célula.

3.4.3.1 Algoritmos para Posicionamento de Transistores

A escolha do algoritmo de posicionamento de transistores é altamente dependente do estilo de leiaute adotado.

Quando Uehara (UEHARA; VANCLEEMPUT, 1981) propôs o estilo *linear-matrix*, ele também sugeriu o algoritmo do caminho de Euler para encontrar um posicionamento dos transistores que minimize o número de quebras na difusão. Entretanto, este método se limita apenas à geração de circuitos duais, onde é possível formar pares de transistores com o mesmo sinal nas duas difusões P e N.

Um método exato para minimizar altura e largura de células CMOS 1D foi proposto pela primeira vez por Maziasz e Hayes em (MAZIASZ; HAYES, 1991). Este método considera a ocupação dos canais de roteamento para tentar minimizar a altura da célula.

Gupta desenvolveu em (GUPTA; HAYES, 1996) um método de minimização da largura de células CMOS no estilo 2D (células 1D com altura de duas ou mais bandas) usando Programação Linear com Inteiros (ILP). A principal contribuição deste trabalho foi propor uma técnica para agrupar transistores em série para poder tratar de células grandes (com mais de 20 transistores) ao custo de uma pequena perda em otimalidade.

Dando prosseguimento a este trabalho, em (GUPTA; HAYES, 1997, 2000) foi apresentada a ferramenta CLIP que é capaz de realizar otimização na largura e na altura das células no estilo 2D. Com o uso de um resolvedor de ILP, a ferramenta é capaz de encontrar a solução ótima para a forma como o problema foi elaborado, considerando: altura, largura, *gaps*, alinhamento vertical dos transistores e densidade horizontal de conexões. Ainda neste trabalho, CLIP foi estendido para agrupar transistores e funcionar de forma hierárquica. A nova ferramenta foi chamada de HCLIP e é capaz de produzir resultados com tempo de execução até três ordens de grandeza menor que CLIP e produzindo os mesmos resultados em aproximadamente 80% dos casos.

Finalmente em (GUPTA; HAYES, 1998), Gupta apresentou a ferramenta FCLIP que diferencia-se de CLIP no fato de conseguir unir a técnica de *folding* (quebra de transistores) com posicionamento usando ILP.

Em (IIZUKA; IKEDA; ASADA, 2004), Iizuka desenvolveu um algoritmo também exato de posicionamento no estilo 1D utilizando a técnica de satisfazibilidade booleana (SAT). O posicionamento e roteamento dos transistores é primeiro transformado num problema de SAT e depois resolvido com um resolvedor de SAT. Os resultados obtidos foram similares aos de Gupta mas com garantia de convergência no roteamento da célula.

Todos estes métodos fazem pares de transistores P e N e não podem ser aplicados à células que possuem redes de transistores P e N não duais.

O problema destes métodos é que em uma biblioteca, não somente *flip-flops* ou *buffers three-state*, mas também circuitos originalmente duais acabam se tornando não duais como resultado da aplicação de outras técnicas, como por exemplo o *folding*. Além disto, a biblioteca também pode utilizar células com lógica de transistores de passagem (PTL), lógica dinâmica/dominó, etc. que podem ser altamente irregulares, com complexo com-

Tabela 3.1: Número de células não-duais em uma biblioteca *standard cell* comercial (IIZUKA; IKEDA; ASADA, 2005a).

Tecnologia	Número total de células	Número de células não-duais	%
130 nm A	527	274	52%
130 nm B	578	294	51%
90 nm A	462	90	19%
90 nm B	340	176	52%

partilhamento de difusão e roteamento pouco trivial. A Tabela 3.1 lista o número de células com redes P e N não duais encontradas em quatro bibliotecas de *standard cells*. Por esta razão, é desejável que ferramentas atuais de geração automática de células também sejam capazes de trabalhar com outros tipos de redes além das série/paralelo complementar.

Diversos métodos para posicionamento de células com redes arbitrárias e número diferente de transistores P e N foram propostos. Os principais trabalhos encontrados serão discutidos a seguir.

Lib (HSIEH et al., 1990) identifica clusters fortemente conectados, e forma pares de transistores P e N dentro do cluster para posicionar transistores no estilo 1D. Um algoritmo de *folding* também foi desenvolvido para tratar transistores maiores que a altura da banda de difusão.

Em (GUPTA; THE; HAYES, 1996), Gupta descreve uma ferramenta chamada XPRESS para produzir leiautes no estilo 1D com suporte a dimensionamento individual dos transistores com o uso de *folding*. Esta ferramenta identifica conjuntos de transistores fortemente conectados para formar subconjuntos. Um algoritmo exato é utilizado para encontrar a melhor cobertura de subconjuntos que minimize o número de quebras na difusão (*gaps*) e o número de trilhas de roteamento horizontais. Segundo o autor do trabalho, esta ferramenta foi utilizada ativamente dentro da Intel, na época, para produção de leiautes de bibliotecas de células para POs.

Em (GURUSWAMY et al., 1997) foi apresentada a ferramenta CELLERITY da Motorola para síntese automática de bibliotecas de *standard cell* no estilo 2D. O sistema desenvolvido suporta células de altura simples (uma banda) e de altura dupla (duas bandas). Um algoritmo de *folding* é aplicado nos transistores antes da realização do posicionamento, modificando o *netlist* inicialmente fornecido. O posicionamento é feito com *Simulated Annealing* (resfriamento simulado) onde a função de custo tem o objetivo de minimizar: o número de quebras de difusão, o comprimento total das conexões, a densidade de canal e o desalinhamento entre *gates*, fontes e drenos dos transistores. Os resultados obtidos com este trabalho deram origem a uma patente (MAZIASZ; GURUSWAMY; RAMAN, 2001) que apresenta detalhes da metodologia desenvolvida.

Um posicionador automático para células de POs é apresentado por Serdar em (SERDAR; SECHEN, 2001). Este algoritmo é capaz de posicionar os transistores sem um estilo previamente definido e em várias orientações diferentes. Para isto, o método proposto neste artigo sugere a representação dos blocos que representam os transistores a serem posicionados através de uma estrutura O-tree (utilizada originalmente em (GUO; CHENG; YOSHIMURA, 1999) para *floorplaning*). Nesta representação, a ordem dos elementos na árvore indica a posição relativa dos blocos, como pode ser observado na

Figura 3.7: Representação O-tree de um posicionamento (GUO; CHENG; YOSHIMURA, 1999).

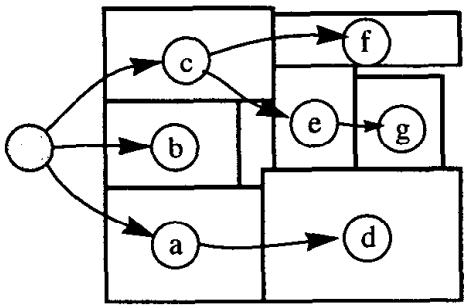
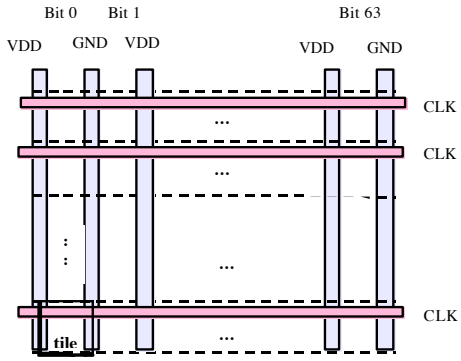
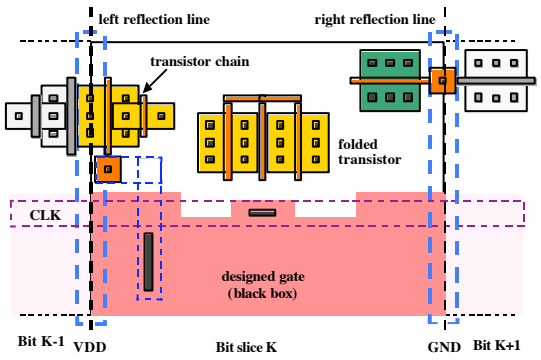


Figura 3.8: Posicionador automático de células de partes operativas (SERDAR; SECHEN, 2001).



(a) Vista global da estrutura da parte operativa



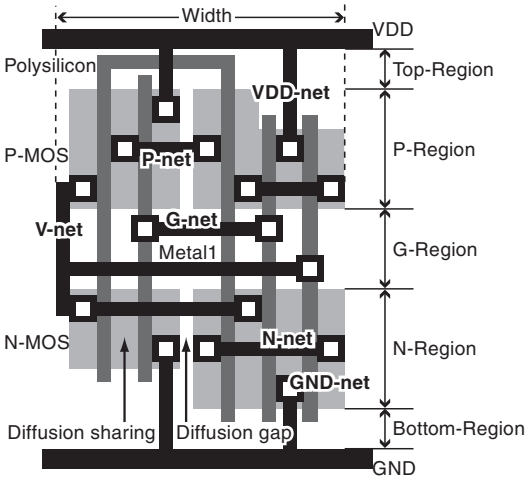
(b) Possibilidades para posicionamento de transistores

Figura 3.7. O autor divide o processo de posicionamento dos transistores em duas etapas: posicionamento global, usando *Simulated Annealing*, e posicionamento detalhado, efetuando transformações na estrutura da O-tree. Estas transformações no posicionamento detalhado visam permitir ao algoritmo lidar com cadeias de transistores em diferentes orientações e a adequação do leiaute às regras de desenho da tecnologia. As principais características do posicionador desenvolvido por Serdar são: capacidade de lidar com largura fixa de célula, posicionamento de transistores na linha de reflexão (para conectar com células adjacentes usando difusão) e formato de célula não retangular. O roteamento detalhado é realizado em 4 camadas de metal pela ferramenta comercial ITools (ITOOLS, 2007). A Figura 3.8 (a) mostra a estrutura regular da PO adotada por Serdar. Em (b), aparece em destaque um exemplo de célula para a estrutura mostrada e as alternativas de posicionamento dos transistores.

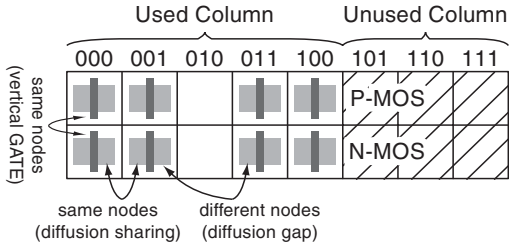
Em (RIEPE; SAKALLAH, 2003), Riepe desenvolveu para a Magma um gerador de leiaute de células 2D chamado TEMPO. A técnica utilizada por ele foi fazer agrupamentos de transistores que formam uma sequência ininterrupta de terminais de dreno e fonte e depois aplicar *Simulated Annealing* para posicionar os grupos de forma a minimizar a função de custo que utiliza uma combinação ponderada dos custos de roteamento e posicionamento (área, perímetro, violação da relação de aspecto, etc.). O restante do fluxo é realizado com a ajuda de um roteador detalhado chamado Anagram-II e o compactador Masterport, ambos providos por terceiros.

Iizuka propôs em (IIZUKA; IKEDA; ASADA, 2005b) modificações na sua ferramenta de síntese para suportar circuitos com redes não duais de transistores. Para reduzir

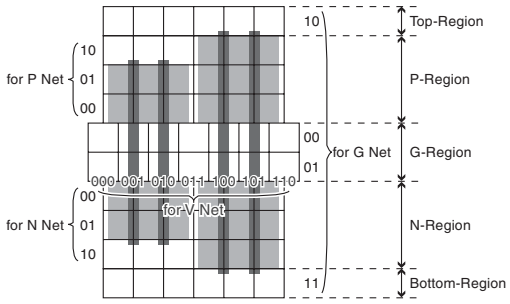
Figura 3.9: Posicionamento de transistores usando SAT (IIZUKA; IKEDA; ASADA, 2005b).



(a) Estilo de layout



(b) Formulação do posicionamento dos transistores



(c) Formulação do roteamento interno da célula

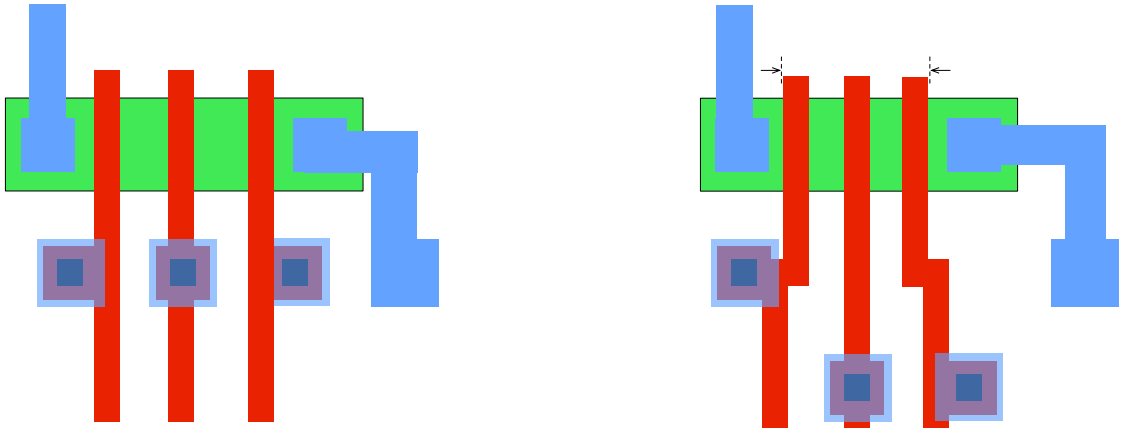
o problema do tempo de execução do algoritmo para células com número elevado de transistores, o autor fez uso de uma hierarquia similar à utilizada em (GUPTA; THE; HAYES, 1996). Como resultado, obteve-se uma melhora considerável no tempo de execução, reduzindo para menos de 2 segundos a geração de células que levavam mais de 1 hora para terminar. Isto permitiu aumentar de 43% para 81% o número de células cobertas pela geração automática na biblioteca utilizada. Em troca, houve uma pequena piora na qualidade do posicionamento, fazendo que algumas células apresentassem menos casamento dos sinais de *gate* entre os transistores das rede P e N. A Figura 3.9 mostra o estilo de leiaute (a), a formulação do posicionamento de transistores (b) e do roteamento interno da célula (c) utilizados por ele.

3.4.4 Posicionamento dos Contatos de Poço e Substrato (*TAPs*)

Para evitar o efeito *latch-up*, diversas tecnologias de fabricação exigem a colocação regular de contatos de poço/substrato (*well/body ties*, TAPs) ao longo do leiaute. Estes contatos são conectados à alimentação para polarizar o poço/substrato e evitar a ocorrência de curto-circuitos nos transistores. Normalmente, os contatos são posicionados em regiões específicas da célula, sob as linhas de alimentação, previamente à realização do posicionamento dos transistores. Entretanto, existem alguns trabalhos no sentido de encontrar um posicionamento dos contatos que traga um melhor aproveitamento da área da célula como em (MAZIASZ; GURUSWAMY; RAMAN, 2001). O método utilizado neste trabalho foi o de posicionar os contatos de substrato somente após a realização da compactação da célula. Um algoritmo varre o leiaute a procura de regiões disponíveis para a colocação dos contatos de substrato e por fim os conecta à alimentação.

Algumas bibliotecas de células não colocam estes contatos diretamente no leiaute das células lógicas. Ao invés disto, possuem células especiais que são colocadas em intervalos regulares, e também em espaços em brancos, maximizando assim a área útil

Figura 3.10: Influência do posicionamento das portas de E/S no leiaute da célula



(a) Posicionamento de pinos ineficiente

(b) Posicionamento de pinos otimizado

de cada célula lógica. Além disto, tecnologias baseadas em SOI também dispensam a necessidades destes contatos, podendo obter uma economia de área em alguns circuitos.

3.4.5 Portas de Entrada/Saída da Célula

As portas entrada/saída da célula são os meios responsáveis pela comunicação com as demais células e interfaces do circuito. O interfaceamento pode ser realizado através da colocação de vias para camadas superiores de metal - como no caso das bibliotecas de *standard cells* recentes para tecnologias com duas ou mais camadas de metal - ou também através de interfaces posicionadas no limite das células para conexões por justaposição - muito utilizado em tecnologias mais antigas, onde não existiam muitos níveis de metais e as conexões eram feitas através de canais de roteamento, e também em circuitos especiais onde o posicionamento das células obedecem regras pré-estabelecidas.

De uma forma geral, as bibliotecas de células possuem suas conexões com a alimentação feitas exclusivamente por justaposição. As demais interfaces da célula são posicionadas no seu interior para serem conectadas através de roteamento *over-the-cell* nas camadas superiores de metal. Estas regiões normalmente são alinhadas à uma grade de roteamento para facilitar a realização das interconexões do circuito e garantir a obediência às regras de desenho da tecnologia.

O posicionamento das interfaces também costuma ter grande influência na área final da célula e na facilidade de interconexão destas com as demais. Um posicionamento ineficiente dos pinos de entrada e saída da célula pode levar a um aumento da área do circuito e também inserir obstáculos para o roteamento externo conforme estudo feito em (LEFEBVRE; MARPLE; SECHEN, 1997). A Figura 3.10 (a) mostra uma situação onde o posicionamento das portas da célula levou a um aumento da área de difusão dos transistores, o que produz um aumento da resistência e das capacitâncias parasitas associadas. Contatos adjacentes posicionados próximos ao *gate* dos transistores dificultam a inserção de curvas no roteamento do polisilício, levando a uma maior distância entre os correspondentes transistores ligados em série. Em (b), a colocação dos pinos em trilhas alternadas possibilitou uma maior aproximação dos *gates* dos transistores em série.

3.4.6 Roteamento

Da mesma forma que o posicionamento, o roteamento interno da célula também é altamente dependente do estilo de leiaute escolhido.

Roteamento de canal é uma das técnicas mais utilizadas em leiautes 1D. Nesta metodologia é definido um canal entre as difusões P e N onde são feitas as conexões entre os transistores. Entretanto, esta solução falha em aproveitar os recursos existentes fora desta região como roteamento em metal sobre os transistores e em polisilício junto à alimentação. Alguns trabalhos como em (ONG; LI; LO, 1989), utilizam algoritmos específicos em combinação com o roteamento de canal para completar o roteamento nestas regiões. Outros reportam o uso de *maze routers* (roteadores em malha) com suporte a obstáculos. Este algoritmo tende a ser mais flexível em termos de estilo de leiaute e estrutura do circuito, mas apresenta maior dificuldade de ajuste (nos custos das arestas) para que possa produzir um leiaute com melhor qualidade. Em (POIRIER, 1989), Poirier descreve o uso de A* para melhorar o desempenho em relação a outros trabalhos que utilizam Lee (LEE, 1961).

De forma a tentar convergir para uma solução, alguns roteadores completam o roteamento pela adição automática de novas trilhas. Esta inserção pode ser disparada pela detecção de uma solução não factível ou *time-out* (tempo limite).

3.4.7 Compactação do Leiaute

Após os elementos estarem devidamente posicionados e o circuito roteado, o leiaute é finalmente compactado. Compactação é o processo de geração do leiaute da célula a partir de uma especificação inicial (leiaute abstrato/simbólico) de acordo com as regras de desenho especificadas (CHEN, 2009). O termo compactação também pode ser usado para a transformação de um leiaute (mais antigo) em um novo com diferentes regras de projeto (também chamado de migração ou recompactação). Trabalhos que não utilizam compactação, como o da ferramenta cellTK (KARMAZIN; OTERO; MANOHAR, 2013), por exemplo, se limitam a instanciar elementos regulares (ou macro blocos) um ao lado do outro e em geral apresentam leiautes com grande desperdício de área.

Uma vez que a altura da célula costuma ser fixa, a compactação no eixo X (largura) costuma ser priorizada pela maior parte dos compactadores para redução da área das células. Já a compactação no eixo Y (altura), permite uma maior aproximação dos transistores P e N (diminuindo a necessidade de aplicar a técnica de *folding*) e trilhas para roteamento horizontal (possibilitando a inserção de mais trilhas e com isto rotear células mais difíceis), o que pode também trazer um ganho indireto em área.

Grande parte dos compactadores são capazes de compactar o leiaute em apenas uma direção de cada vez (1D) (MARPLE; SMULDERS; HEGEN, 1988; GURUSWAMY et al., 1997; ZHU; FANG; TANG, 2005; LAZZARI, 2007; ZIESEMER; LAZZARI; REIS, 2007; IIZUKA, 2007; FU et al., 2009; SAID et al., 2011). Entretanto, existem alguns poucos trabalhos que reportam o uso de compactadores bi-dimensionais (2D) que realizam a compactação da altura e largura da célula simultaneamente (SHIGEHIRO et al., 1994; TANG; ZHU, 2005; MARPLE, 2012). Esta metodologia tem a vantagem de explorar melhor o espaço de soluções, diminuindo a chance de ficar presa em mínimos locais, como mostra a Figura 3.11. Eles são usualmente feitos utilizando heurísticas para a compactação 1D, uma vez que este problema já foi provado como sendo NP-completo (SCHLAG; LIAO; WONG, 1983; KEUTZER; NEWTON; ORSHANSKY, 2000). Em (MARPLE, 2012), por exemplo, a decisão sobre a posição relativa dos elementos é feita

Figura 3.11: Comparação entre a compactação 1D e 2D (KEUTZER; NEWTON; ORSHANSKY, 2000).

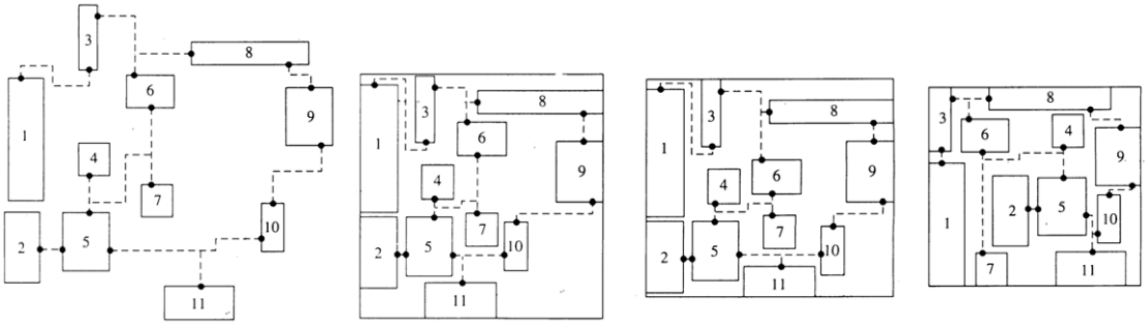
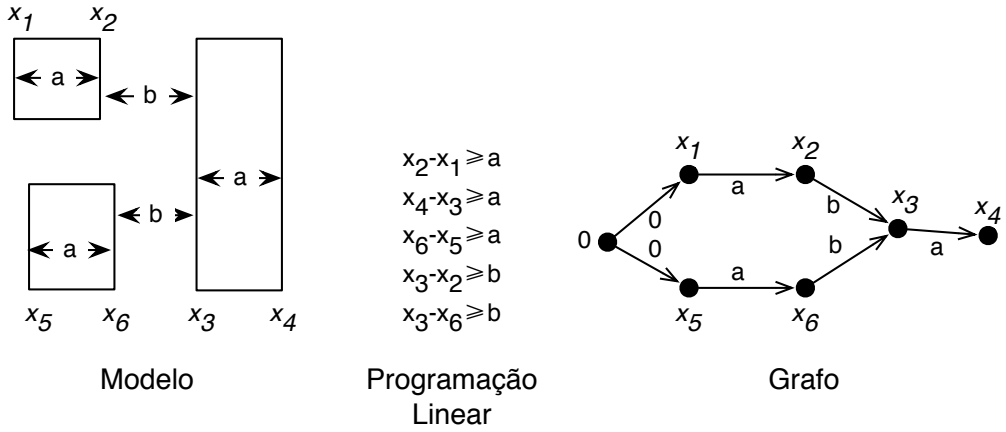


Figura 3.12: Compactação de leiaute baseada em Programação Linear e Grafos (CHANG, 2014).



antes de resolver o problema incrementalmente utilizando um método exato (ILP), com a intenção de diminuir a complexidade do método.

Os algoritmos encontrados para resolver este tipo de problema, em geral, envolvem técnicas de teoria de grafos e programação linear, ilustrados na Figura 3.12. Outros métodos incluem *simulated annealing*, *shadow propagation* e *scanline*. Uma revisão dos métodos de compactação pode ser encontrada em (BOYER, 1988).

Compactadores baseados em grafos (FU et al., 2009; SAID; ABBAS; SHAHEIN, 2007) são os mais comumente encontrados. Neste método, os nós representam os elementos do leiaute ou bordas dos polígonos, e os arcos representam regras de desenho ou restrições de conexão entre os elementos/bordas. Normalmente as regras de distância mínima da tecnologia são utilizadas para estabelecer valores para os arcos. O problema com compactadores baseados em grafos é que regras condicionais são de difícil representação. Isto os torna inadequados para tecnologias abaixo de $130nm$, que possuem número crescente de regras de projeto condicionais, como por exemplo: o espaçamento entre metais no final de linha e a extensão do *gate* dos transistores (ambas as regras serão explicadas em detalhes no Capítulo 4).

Em compactadores baseados em programação linear (FANG; ZHU, 2004), os lados dos retângulos que formam o leiaute são representados por variáveis enquanto que as regras são representadas por constantes. Desigualdades são utilizadas para modelar o espaçamento mínimo entre os lados dos retângulos. A utilização de resolvedores de equa-

Tabela 3.2: Comparação do ASTRAN com trabalhos relacionados e estado-da-arte

	ASTRAN	Nangate & Prolific	cellTK (2013)	Marple (2012)	Said (2011)	Fu (2009)	Iizuka (2007)	Lazzari (2007)	Zhu (2005)
Tipo	Síntese	Síntese	Síntese	Compact.	Analóg.	Migração	Síntese	Síntese	Migração
Folding	X	X	X				X	X	
Posicionamento de Transistores	X	X	X				X	X	
Roteamento *Intra-cell*	X	X	X				X	X	
Compactação	2D (exato)	X		2D (heuríst.)	1-1/2D	1-1/2D	1D	1D	1-1/2D
Regras de Leiaute Condicionais	X	?							

ções lineares que suportam variáveis inteiras (ILP) permite forçar que algumas variáveis assumam valores inteiros, o que é útil, por exemplo, para alinhar as geometrias à grade de roteamento ou da tecnologia.

As principais funções de minimização definidas durante a compactação normalmente se referem à largura das células e também ao comprimento das conexões (em especial as críticas). Algumas restrições também são frequentemente necessárias para adequar o resultado ao estilo de leiaute especificado, tais como: altura da célula, posição dos pinos, laterais alinhadas à grade de roteamento, etc.

3.5 Conclusão

Este capítulo fez uma breve revisão bibliográfica sobre geração automática de leiaute de células lógicas para tecnologia CMOS. As etapas básicas de um fluxo de geração e a forma como diversos trabalhos tratam cada um dos problemas associados foi apresentada.

Durante esta revisão, foi visto que células com redes não dual de transistores P e N representam até metade do total de células de uma biblioteca *standard cell*. Isto enfatiza a necessidade que geradores de células lógicas suportem qualquer tipo de rede de transistores.

Também foi visto que a compactação do leiaute pode ser feita em 1D ou 2D. Compactação em 2D, tem a vantagem de poder lidar com restrições tanto verticais quanto horizontais simultaneamente, podendo obter um melhor resultado em área, mas também apresentando maior complexidade de implementação e computacional.

A Tabela 3.2 mostra uma comparação dos trabalhos apresentados neste capítulo com as ferramentas comerciais da Nangate (NANGATE LIBRARY CREATOR, 2014) e Prolific (PROGENESYS, 2011) (baseados nos materiais de divulgação encontrados online), e a ferramenta ASTRAN desenvolvida neste trabalho, que será apresentada no próximo capítulo. Excetuando as ferramentas comerciais, cujos algoritmos não são divulgados, o ASTRAN é o primeiro trabalho a utilizar com sucesso um método exato para compactação em 2D com suporte a regras de leiaute condicionais.

4 DESENVOLVIMENTO DO GERADOR AUTOMÁTICO DO LEIAUTE DE REDES DE TRANSISTORES: ASTRAN

Este capítulo apresenta a ferramenta para síntese de redes de transistores MOS desenvolvida neste trabalho. Esta ferramenta foi elaborada a partir de uma seleção dos algoritmos e estratégias reportados no capítulo anterior e recebeu o nome de ASTRAN.

4.1 Introdução

Para automatizar o projeto do leiaute de células lógicas, foi desenvolvido uma ferramenta de síntese de células cujas características incluem o suporte a células com diferentes redes e tamanhos de transistores.

As células geradas são compatíveis com os principais modelos utilizados em bibliotecas comerciais de *standard cells*. Isto permite que estas possam ser adicionadas a uma biblioteca de células pré-existente.

A ferramenta foi desenvolvida objetivando não um processo de fabricação específico, mas um conjunto de regras de projeto comuns a diversos processos de fabricação. Com isto foi possível gerar células para diversos nodos de tecnologia entre $350nm$ e $45nm$.

4.2 Formulação do problema

A especificação da estrutura interna das células é feita no formato SPICE. Este formato possui, para cada célula informada:

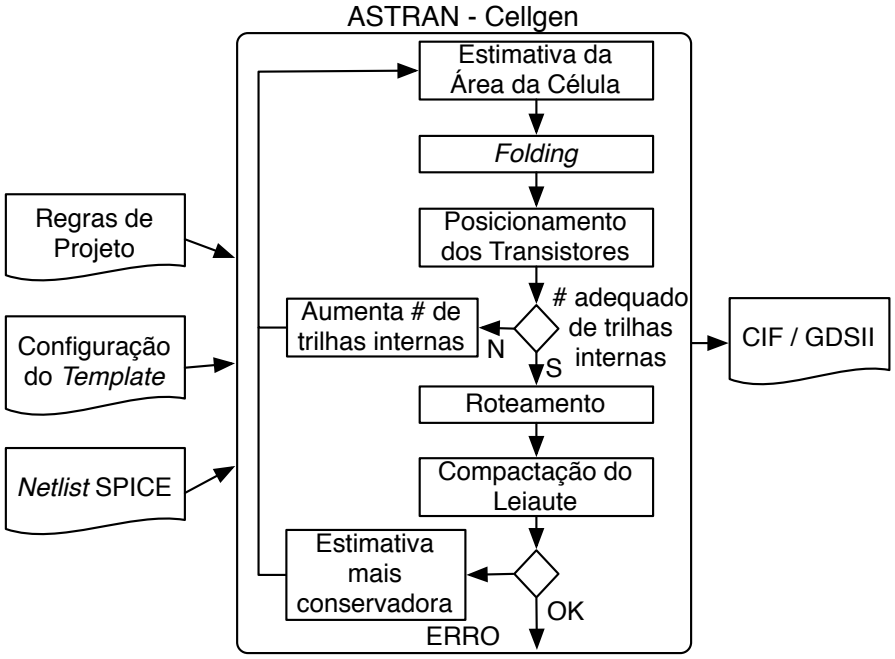
- a lista de seus transistores;
- as redes que os conectam;
- o comprimento e a largura dos *gates*;
- as interfaces de entrada/saída.

O gerador recebe este arquivo como entrada e gera o leiaute de cada uma das células de acordo com as regras da tecnologia especificada e modelo com os parâmetros de geração. O leiaute resultante é salvo em formato CIF, GDSII ou também pode ser exportado para LEF. A Figura 4.1 mostra o fluxo de projeto da ferramenta.

4.3 Estilo de Leiaute

O estilo de leiaute escolhido para ser utilizado no gerador de células é baseado fundamentalmente no estilo 1D proposto inicialmente por Uehara (UEHARA; VANCLEEM-

Figura 4.1: Fluxo de projeto do gerador de células (ZIESEMER et al., 2014a).



PUT, 1981). A Figura 4.2 ilustra o estilo utilizado neste trabalho e suas características são listadas na Tabela 4.1.

O alinhamento dos pinos de E/S da célula a uma grade de roteamento garante a conformidade de suas conexões às camadas superiores de metal, de acordo com a grade de roteamento detalhado e as regras de projeto da tecnologia utilizada. Também permite que algoritmos de menor complexidade possam ser utilizados para realização do roteamento com as demais células do circuito. Esta técnica tem sido o padrão para projeto de bibliotecas de células para fluxos automatizados de síntese, e por esta razão também foi definida no estilo de leiaute proposto.

Para garantir que os pinos fiquem alinhados à grade de roteamento é necessário fazer com que as células também sejam posicionadas alinhadas a uma grade. Existem duas formas de realizar este posicionamento. A Figura 4.3 (a) mostra a forma de posicionamento sem *offset* (deslocamento) no eixo X, onde as células são posicionadas sobre a grade de roteamento. Já em (b) é mostrado a forma de posicionamento com *offset* no eixo X, onde a célula é posicionada no meio entre duas posições adjacentes da grade. Isto permite que a célula tenha uma coluna a mais de espaço para posicionar seus pinos de E/S. A existência de *offset* em X ou Y faz parte do modelo da biblioteca de células escolhido pelo projetista.

4.4 Fluxo de Geração de Células do ASTRAN

Nesta seção é explicado cada um dos passos mostrados na Figura 4.1 para geração do leiaute das células com o ASTRAN.

4.4.1 Estimativa da Área da Célula

Neste passo é estimado a largura (W) máxima dos transistores (nas regiões P e N) e os recursos de roteamento que estarão disponíveis de forma a poder criar uma representação abstrata da célula que possa ser compactada posteriormente.

A Figura 4.4 mostra os elementos calculados durante este processo para o circuito de

Figura 4.2: Estilo de leiaute 1D utilizado no gerador

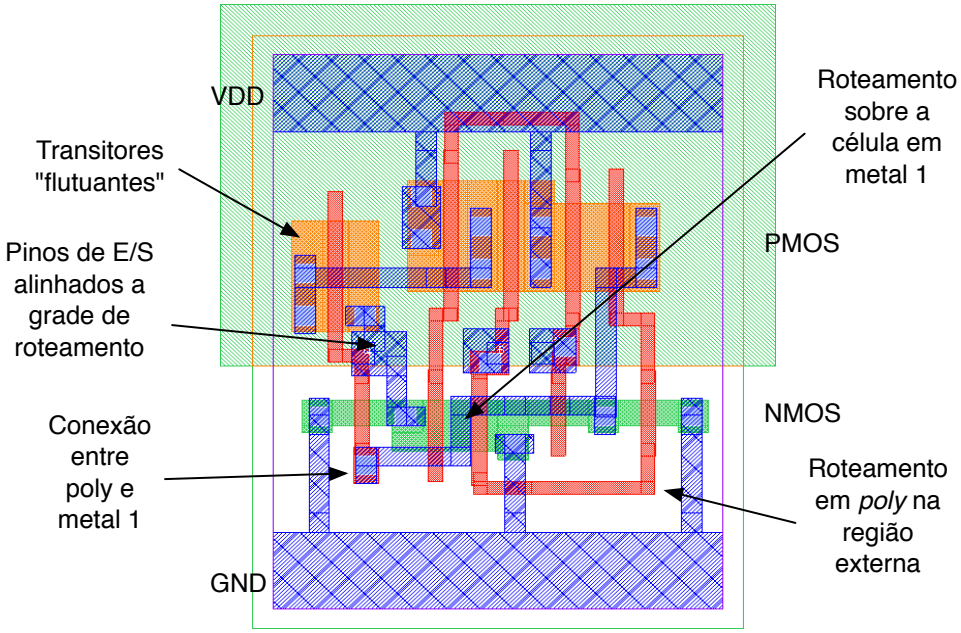
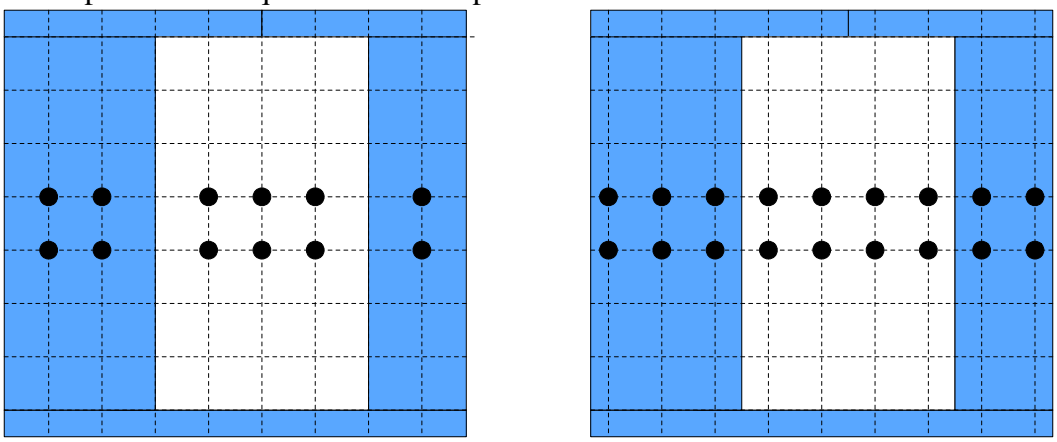


Tabela 4.1: Estilo de Leiaute

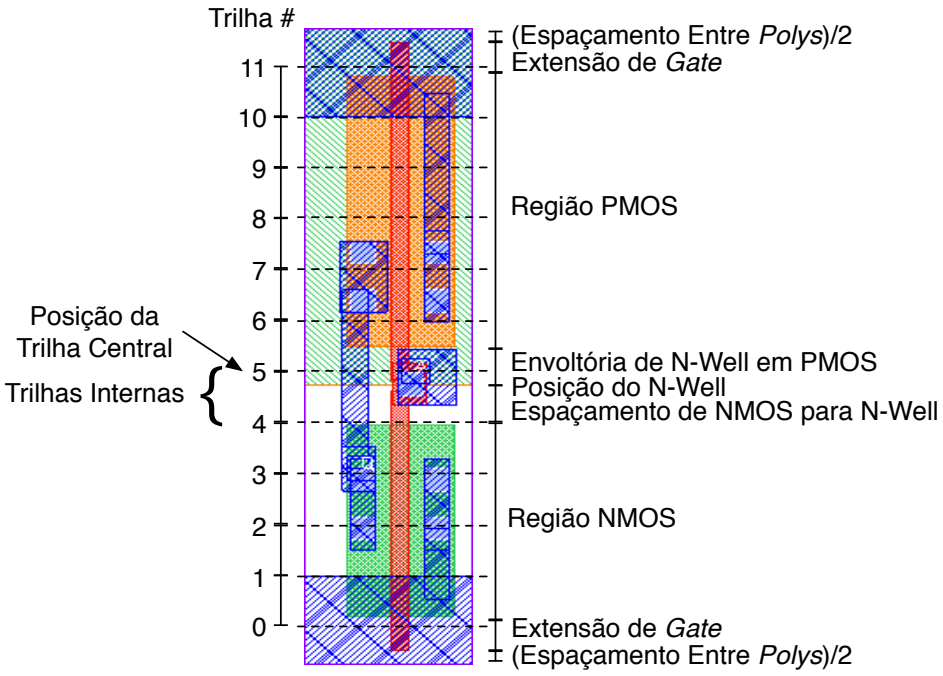
1. Suporte irrestrito quanto a estrutura da rede dos transistores (duais, não-duais, com transistor de passagem,...) e tamanho destes (largura e comprimento do canal). A estrutura é preservada (não ocorrendo nenhum re-ordenamento), exceto pela aplicação de *folding* quando necessário;
2. Transistores posicionados em duas bandas distintas para transistores PMOS e NMOS;
3. Roteamento interno da célula em polisilício, metal 1 e difusão;
4. Contactos entre polisilício e metal 1 em qualquer região da célula, exceto sobre os transistores;
5. Transistores podem ser posicionados em qualquer lugar dentro das regiões PMOS e NMOS. Eles podem se mover livremente tanto verticalmente quanto horizontalmente durante a compactação, desde que preservado o ordenamento relativo horizontal;
6. Trilhas sobre as difusões P e N para roteamento em metal;
7. Linhas de alimentação nas regiões superiores e inferiores da célula para conexão por justaposição com as células vizinhas em metal 1;
8. TAPs (para polarição do poço e substrato) posicionados em baixo das linhas de alimentação (opcional), conforme definido pelo modelo escolhido pelo projetista;
9. Portas de entrada e saída da célula alinhadas à grade de roteamento do circuito;
10. Inserção automática de *jogs* (conexões em L) para roteamento em metal 1 e *poly*, podendo este último ser desabilitado;
11. Altura da célula e posição do poço pré-determinados pelo modelo. Largura múltipla do *pitch* horizontal da grade de roteamento.

Figura 4.3: Opções de alinhamento das células à grade de roteamento. Destaque na quantidade de pinos de E/S que cada técnica permite.



(a) Sem offset (b) Com offset em X

Figura 4.4: Estimativa da área de um inversor.



um inversor. A altura das regiões de difusão PMOS e NMOS são calculadas de acordo com as regras de projeto e o *template* (modelo) de biblioteca de células escolhido: altura das células, posição do poço N (*N-Well*), número de trilhas internas, *offset*, *pitch* vertical/horizontal da grade de roteamento e inserção ou não de TAPs. A posição da trilha central é calculada de acordo com a posição do poço N e a grade de roteamento em metal 1 (dado pelo *pitch* vertical e levando em consideração o *offset*). Uma vez que esta posição é obtida (posição esta que é bastante utilizada para colocação dos pinos de E/S), o número de trilhas abaixo e acima desta trilha central é maximizada (de acordo com o *pitch* mínimo entre linhas de metal 1). Trilhas que se interseccionam ou violam a distância mínima com as linhas de alimentação são identificadas de forma que possam ser marcadas como pertencentes a estas redes durante a etapa de roteamento. O mesmo número de trilhas é utilizado para roteamento em polisilício. Isto é feito para facilitar a conexão entre as trilhas de metal 1 e polisilício que desta forma podem ocorrer em qualquer trilha desocupada da grade. No entanto, como as trilhas de polisilício possuem usualmente um *pitch* maior que as de metal 1, isto pode levar a uma representação irrealista da célula, impedindo que esta seja compactada (por não caber todas as trilhas na altura da célula).

As trilhas internas (trilhas horizontais de metal e *poly* entre as regiões PMOS e NMOS) permitem realizar a interconexão entre os transistores em ambas as difusões. Quanto maior a quantidade destas trilhas, maior a facilidade do roteador encontrar uma solução em células de difícil roteamento. Por outro lado, menor é o tamanho das regiões P e N, fazendo com que a célula tenha uma largura maior devido a aplicação da técnica de *folding* (explicada a seguir).

O número de trilhas internas é inicialmente calculado como o número de trilhas que cabem na região entre os transistores P e N (devido às regras de espaçamento mínimo da difusão N para o poço N e de envoltória mínima do poço N nos transistores P) de forma a maximizar estas regiões e evitar *folding*. Após a execução da etapa de posicionamento, uma estimativa da complexidade do roteamento é feita e então o número de trilhas é ajustado apropriadamente.

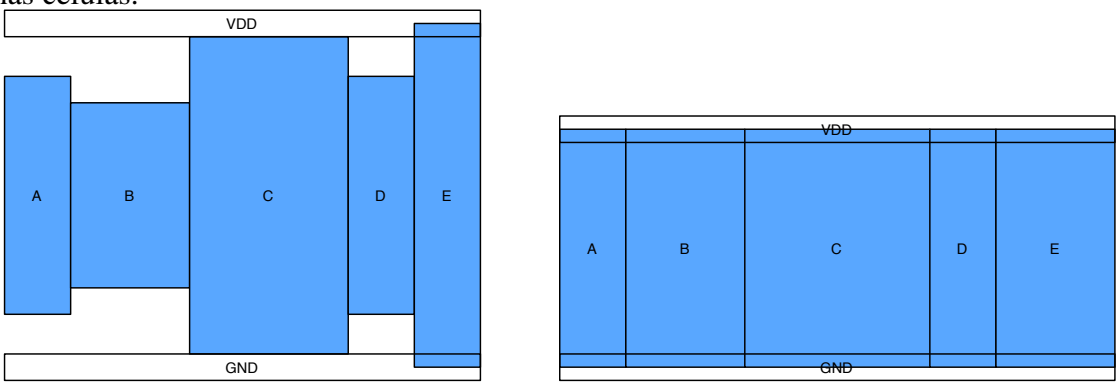
Uma vez que a ferramenta pode sub ou sobre-estimar o número de trilhas que cabem na altura da célula, o ASTRAN começa com uma abordagem otimista (maximizando as regiões PMOS/NMOS e o número de trilhas horizontais) e então, fica mais conservador (reduzindo o número de trilhas horizontais e o tamanho das regiões PMOS/NMOS) se o leiaute não puder ser compactado. A abordagem otimista ajuda na resolução de circuitos complexos mas pode não ser possível de compactar por estar alocando mais elementos do que cabem na altura da célula.

4.5 *Folding* dos Transistores

Dimensionamento de transistores é essencial para a produção de circuitos com alta performance. Várias ferramentas são capazes de realizar um dimensionamento individual dos transistores com o objetivo de reduzir o atraso e o consumo de energia (SANTOS et al., 2005). Outras realizam o dimensionamento de todos os transistores da célula simultaneamente (POSSER; REIS, 2011) com relação a uma célula de referência pré-dimensionada. Leiautes produzidos no estilo 1D com transistores de diferentes alturas tendem a desperdiçar área, uma vez que a altura de cada linha de difusão P e N é calculada de acordo com o seu transistor mais largo como pode ser visto na Figura 4.5.

Para resolver este problema, um dos métodos mais utilizados é o *folding* de transistores. Este método consiste em quebrar os transistores de maior tamanho em transistores

Figura 4.5: Diferença de altura na banda de acordo com a aplicação do método de *folding* nas células.



(a) Sem folding (b) Com folding

menores, conectados em paralelo, com o objetivo de manter a altura da célula reduzida e padronizada para toda a biblioteca, as custas de um pequeno acréscimo na largura. De acordo com Gupta (GUPTA; HAYES, 1998), o problema de *folding* pode ser classificado como posicionamento estático/dinâmico com *folding* estático/dinâmico:

1. Posicionamento e *folding* estáticos: Dado um posicionamento dos transistores e o limite nas suas larguras (limites para o *folding*), quebrar os transistores e determinar suas orientações de forma a preservar o posicionamento e minimizar a área.

2. Posicionamento estático com *folding* dinâmico: Dado um posicionamento dos transistores, determinar as quebras e orientação de cada transistor de forma a minimizar a área;

3. Posicionamento dinâmico com *folding* estático: Dado os limites para o *folding*, quebrar os transistores e determinar a sua posição e orientação de forma a minimizar a área da célula.

4. Posicionamento e *folding* dinâmicos: Para cada transistor, determinar o número de quebras, sua posição e orientação tal que a área total seja minimizada.

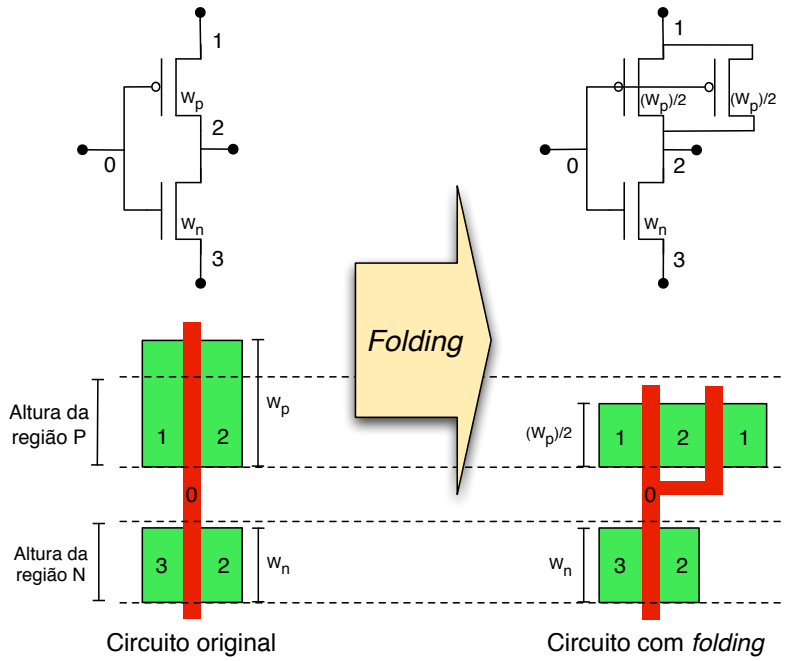
Apesar desta classificação originalmente ter sido proposta para posicionamento de leiautes no estilo 2D, o mesmo se aplica para 1D.

A metodologia escolhida neste trabalho foi a de número 3, com posicionamento dinâmico e *folding* estático. A razão para esta escolha é devido ao fato do *folding* potencialmente estragar o posicionamento dos transistores caso feito depois deste. Também, como as regiões de difusão P e N determinam limites para *folding* dos transistores, há pouca vantagem em realizar o *folding* dinamicamente.

4.5.1 Especificação do Problema

Segundo esta metodologia, o problema pode ser definido como: dada uma rede de transistores e limites de altura para as regiões de difusões P e N, substituir os transistores da rede com largura (W) maior do que o limite estabelecido por transistores menores, paralelos, de forma que a soma das suas larguras seja igual à largura do transistor original. Uma ilustração deste método é apresentada na Figura 4.6.

Figura 4.6: Método de *folding* aplicado à um transistor maior que a altura da linha de difusão.



A execução do algoritmo de *folding* garante que todos os transistores possuam largura menor ou igual ao estabelecido para suas respectivas regiões de difusão. Desta forma, nenhuma outra modificação na rede de transistores precisa ser feita nas etapas seguintes do fluxo.

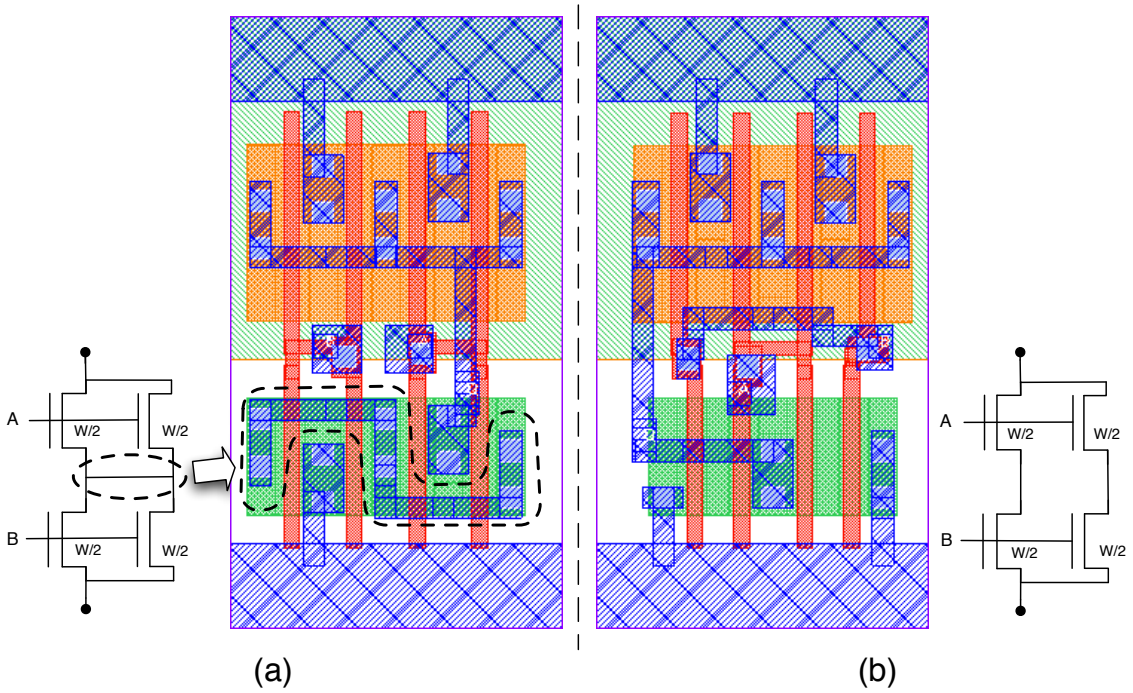
Uma questão que ainda pode ser melhorada é ilustrada na Figura 4.7. Ao invés de aplicar a técnica individualmente em cada transistor, melhores resultados podem ser obtidos identificando os transistores que estão em série e aplicando a técnica simultaneamente em todos eles, eliminando a necessidade de fazer as interconexões internas/intermediárias. Isto porque nós que estão no mesmo potencial não necessitam ser conectados, reduzindo o número de conexões entre os transistores e também o número de contatos entre os *gates*.

4.6 Posicionamento

A função da etapa de posicionamento é encontrar um arranjo de transistores dentro da área da célula que atenda a um determinado objetivo. O objetivo mais frequentemente encontrado nos algoritmos de posicionamento para o estilo 1D é o de encontrar o ordenamento e orientação dos transistores nas bandas P e N de forma que o número de quebras de difusões (*gaps*) seja mínimo e que haja uma correspondência entre os sinais de *gate* dos transistores verticalmente alinhados nas duas difusões.

Diversos métodos exatos são capazes de encontrar a solução ótima (MAZIASZ; HAYES, 1991; GUPTA; HAYES, 1998; IIZUKA; IKEDA; ASADA, 2004) ou quase-ótima (IIZUKA; IKEDA; ASADA, 2005a) para este problema em tempo aceitável. Entretanto, estes métodos não consideram o uso de outras métricas de qualidade que podem levar a soluções ainda melhores tais como: comprimento total das conexões e densidade do canal. Segundo (VAHIA; CIESIELSKI, 1999), o uso de métodos aleatórios tem tido grande aceitação entre pesquisadores devido a sua habilidade de lidar com funções de custos multidimensionais. Cellerity (GURUSWAMY et al., 1997) foi a primeira ferramenta a

Figura 4.7: Possibilidade de melhoria da técnica de *folding* no ASTRAN. (a) Implementação atual; (b) Modificação proposta diminuiu a quantidade de conexões e número de contatos entre os *gates* NMOS.



incorporar todas estas métricas num posicionador feito com *Simulated Annealing* (resfriamento simulado).

O posicionador desenvolvido neste trabalho utiliza como heurística tanto o algoritmo de caminho de Euler (que é capaz de encontrar rapidamente e de forma exata o posicionamento com o menor número de quebras de difusões) quanto um método heurístico que faz uso de *Threshold Accepting* (aceitação por limiar) - para posicionar transistores com uma estrutura de rede mais complexa e de forma a incluir outras métricas de qualidade durante o posicionamento.

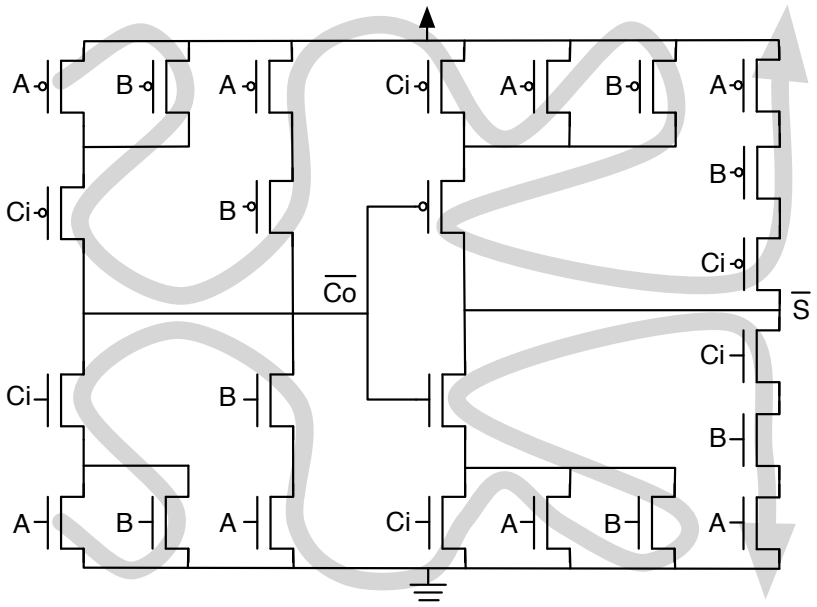
4.6.1 Especificação do Problema

O problema de posicionamento de transistores em 1D pode ser definido como: dada uma descrição da rede de transistores de uma célula, distribuir os transistores em duas listas distintas P e N, de acordo com o seu tipo, e encontrar um ordenamento e orientação dos transistores que minimize a área e melhores suas características elétricas. Para tanto, deve-se considerar que transistores vizinhos pertencentes à mesma lista e que compartilham o sinal da difusão podem ser posicionados sem a existência de *gaps*. Transistores com mesmo índice na lista oposta tendem a ser posicionados verticalmente alinhados durante a etapa de compactação do leiaute da célula.

Listas com diferentes quantidades de transistores P e N são completadas com transistores *dummies* para que ambas as listas fiquem com o mesmo tamanho e todos os transistores tenham o seu par na lista oposta.

Os detalhes da implementação dos dois métodos de posicionamento desenvolvidos são descritos a seguir.

Figura 4.8: Exemplo de caminho de Euler em um somador completo.



4.6.2 Caminho de Euler

Um método simples para encontrar um ordenamento ótimo para os transistores, em termos de número de *gaps* nas linhas de difusão, é através do algoritmo de caminho de Euler (UEHARA; VANCLEEMPUT, 1981). O método consiste em encontrar um caminho comum ininterrupto - com o mesmo ordenamento dos sinais de *gate* - para ambas as redes *pull-up*/*pull-down* e que passe por cada transistor apenas uma vez. A estrutura de dados utilizada para execução deste algoritmo foi um grafo não orientado onde as arestas representam os transistores e os vértices, as redes. Um exemplo de caminho de Euler na rede de transistores de um somador completo é mostrado na Figura 4.8.

Nem sempre é possível encontrar um único caminho de Euler para todo o grafo. Neste caso, a estratégia utilizada foi encontrar o menor número de sub-caminhos de Euler que juntos percorram todas as arestas do grafo apenas uma vez. Sempre que um sub-caminho termina, um *gap* é gerado e um novo caminho é percorrido, começando por qualquer outro par de arestas que ainda não tenham sido visitadas.

Uma limitação deste algoritmo é que apenas células que possuam equivalência de 1:1 entre os transistores das redes P e N são posicionadas com sucesso - como no caso da classe dos circuitos CMOS estático complementar. Alguns circuitos como multiplexadores e portas XNOR feitos com lógica de transistores de passagem também puderam ser posicionados utilizando caminho de Euler, mas somente quando possuíam esta característica na rede de seus transistores. No entanto, até mesmo circuitos CMOS estático complementar podem ter suas redes desbalanceadas pela aplicação da técnica de *folding*, impossibilitando a utilização deste algoritmo.

Devido à estas restrições, um novo posicionador de transistores precisou ser desenvolvido, sendo descrito a seguir.

4.6.3 Threshold Accepting

A heurística *Threshold Accepting* (DUECK; SCHEUER, 1990) foi proposta para ser uma evolução do algoritmo conhecido como *Simulated Annealing* (KIRKPATRICK; GELATT; VECCHI, 1983). A principal diferença é que enquanto neste a função de aceitação é probabilística, na heurística *Threshold Accepting* a função de aceitação é determinística, como será explicada a seguir. Segundo os autores, o método é mais simples e apresenta resultados aparentemente superiores ao *Simulated Annealing*.

O algoritmo é ilustrado da Figura 4.9. Ele começa com uma solução inicial qualquer (que pode ser aleatória) e executa perturbações com a finalidade de encontrar soluções próximas, de acordo com o valor do *threshold* (limiar). Este valor define o quanto soluções piores podem ser aceitas e diminui ao longo das iterações. Por fim, quando o *threshold* for igual a zero, o algoritmo torna-se guloso e somente aceita perturbações que levem à soluções de igual ou menor custo (melhores) que a atual. O algoritmo termina quando nenhuma melhora é detectada após várias iterações.

A Figura 4.10 mostra um exemplo de uma possível execução do algoritmo de *Threshold Accepting* (TA) sobre um circuito, e a melhoria do posicionamento conforme o *threshold* diminui.

A versão do algoritmo de TA utilizada neste trabalho é a mesma de (HENTSCHKE, 2007). Hentschke desenvolveu um *template* em C++ que implementa as funções essenciais do algoritmo, exigindo apenas o desenvolvimento das funções de avaliação de custo e perturbação, que variam para cada aplicação da técnica. Este *template* caracteriza-se por utilizar um escalonador de *threshold* adaptativo para acelerar o decréscimo do *threshold* nos casos onde a taxa de aceitação for muito alta - o que pode indicar que o algoritmo fica alternando entre soluções aleatórias sem convergir para um decréscimo no custo - e diminuir a velocidade, quando a taxa de aceitação for menor. Os dois métodos que precisaram ser implementados são descritos a seguir.

4.6.3.1 Função de Custo

A função de custo retorna uma medida qualitativa da solução ao longo das iterações do algoritmo de TA.

Foram definidas 6 métricas para determinar a qualidade do posicionamento em 1D (representadas na Figura 4.12):

- Largura da célula - Este índice tem o objetivo de produzir um posicionamento com um menor número possível de *gaps* (quebras) em cada difusão. Outro objetivo é alinhar verticalmente *gaps* que ocorram simultaneamente em ambas as difusões (pois o impacto em área costuma ser menor do que deixar eles esparsos). Para tanto, a largura da célula é dada pelo número de colunas utilizadas para criar sua representação abstrata. Cada transistor insere 3 colunas representando os seus 3 terminais (fonte, *gate* e dreno) na sua respectiva difusão. A inexistência de *gaps* entre transistores adjacentes em ambas as difusões faz com que a última coluna dos transistores anteriores sejam reaproveitadas para os transistores seguintes, sendo necessário inserir apenas 2 novas colunas para o *gate* e fonte/dreno.

- Número de *Gaps* - É o número total de quebras de difusão existentes nas difusões P e N somados. Esta medida faz-se necessária uma vez que *gaps* verticalmente alinhados contribuem apenas uma vez para o aumento da largura da célula (métrica esta explicada acima).

Figura 4.9: Fluxo de execução do algoritmo de *Threshold Accepting*.

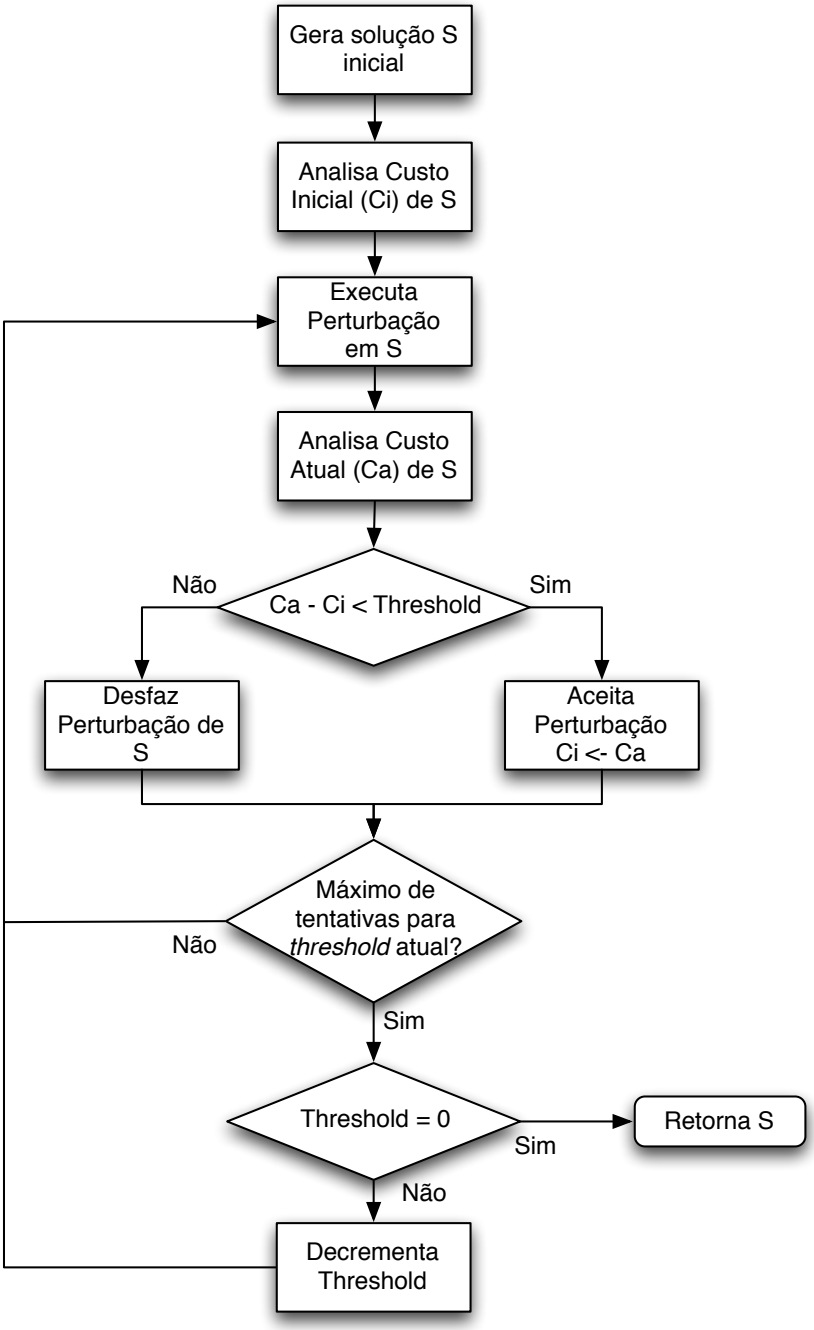
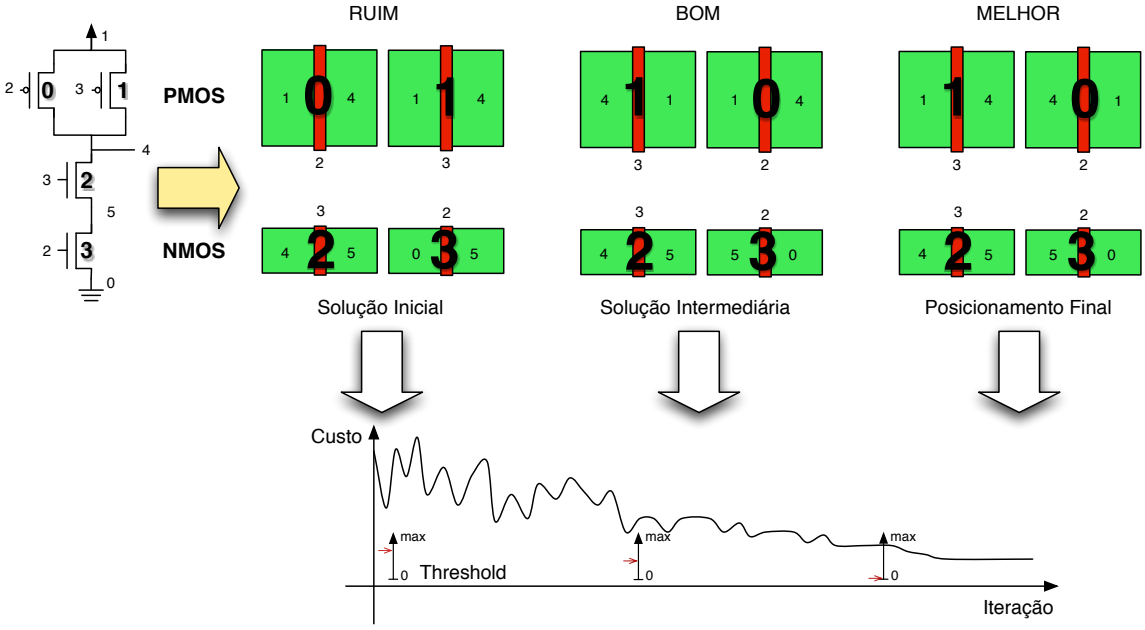


Figura 4.10: Exemplo de execução do algoritmo de posicionamento usando *Threshold Accepting*.



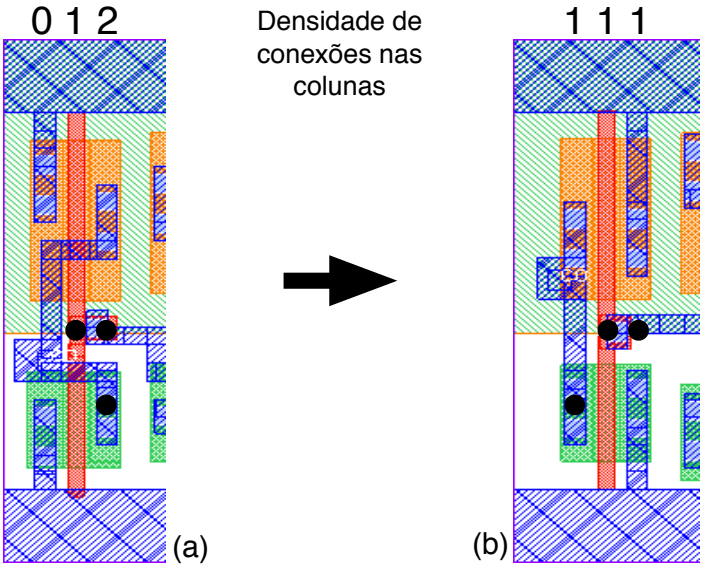
- *Gate Mismatches* - Quando os sinais de *gate* de dois transistores de difusões opostas são posicionados de forma alinhada, eles podem ser conectados de forma eficiente utilizando apenas polisilício. Por esta razão, esta métrica foi inserida para tentar minimizar o número de transistores "desalinhados". Ela é calculada pelo número de transistores com sinal de *gate* diferente do transistor que ocupa a mesma posição relativa na difusão oposta.

- Comprimento das conexões - Esta métrica realiza uma estimativa do tamanho final de todas as conexões dentro da célula. O tamanho de cada conexão é definido como a largura - em número de colunas (*gates*, difusões, *gaps*) - de cada rede da célula. Desta forma, um valor menor para esta métrica tende a produzir como resultado um leiaute com conexões de menor tamanho. As conexões pertencentes as redes de alimentação não entram no cálculo por não adicionarem conexões horizontais dentro da célula. Na Figura 4.10 o posicionamento final teve um custo menor que o intermediário graças a este fator, que tende a produzir posicionamentos com maior quantidade de conexões com a alimentação e com melhores características elétricas.

- Densidade máxima de conexões - Mede a quantidade de conexões que cruzam horizontalmente determinada coluna do circuito. Serve para diminuir a complexidade do roteamento: uma vez que há uma quantidade limitada de recursos de roteamento horizontal entre os transistores, uma extrapolação deste valor é fator suficiente para impedir sua geração. É calculado com base na coluna inicial e final de cada rede da célula (exceto redes de alimentação já que estas, em geral, não utilizam recursos de roteamento horizontais). Todas as colunas entre estas duas posições (incluindo elas próprias) têm sua densidade incrementada em uma unidade. Ao final, a densidade máxima de conexões é tomada com base na coluna de maior densidade.

- Densidade local das conexões - Esta métrica foi a última a ser adicionada e é calculada pelo somatório do quadrado das densidades locais das colunas (conforme

Figura 4.11: Melhoria no roteamento causada pelo uso da métrica de densidade local de conexões no leiaute de um inversor dentro de um somador total (ZIESEMER et al., 2014b). Os leiautes mostrados correspondem a antes (a) e depois (b) da aplicação da métrica. Redes que ocupam as colunas estão destacadas com um círculo preenchido (redes de alimentação e pinos não são contabilizados).



explicado no item anterior). A motivação para a sua utilização é exemplificada pela Figura 4.11. Enquanto a densidade máxima de conexões é uma importante métrica para permitir que mais células possam ser roteadas, ela não colabora para reduzir o número de conexões nas demais colunas da célula, como podemos ver em (a). Um simples espelhamento dos transistores P e N do inversor, como mostrado em (b), permitiu reduzir a densidade de conexões na coluna da direita de 2 para 1, o que permitiu melhorar a qualidade final do roteamento.

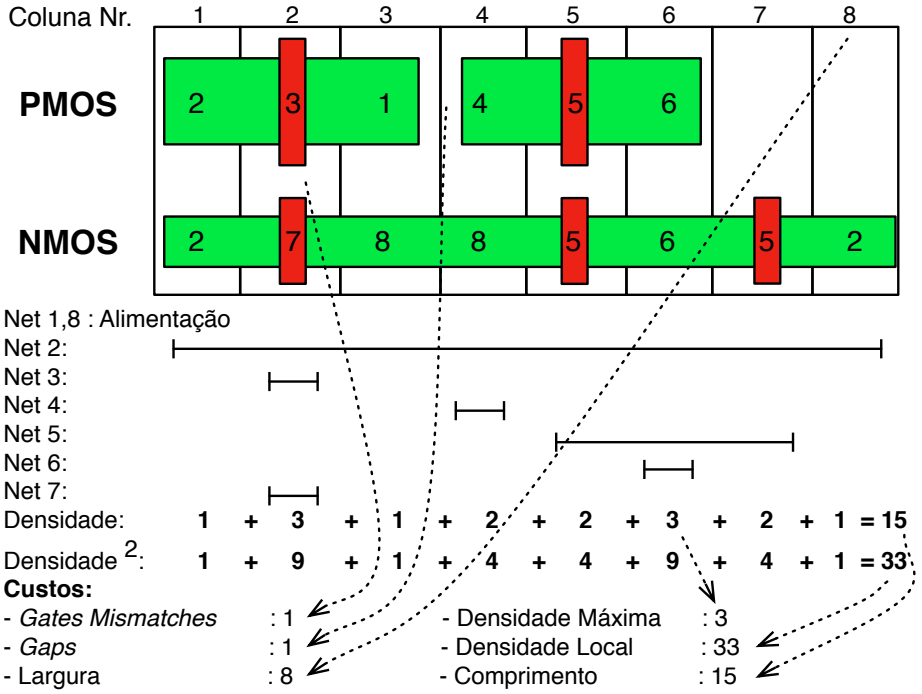
O modelo de representação do posicionamento da célula é mostrado na Figura 4.12. Este modelo é utilizado para calcular o custo de posicionamento e também serve como entrada para a etapa seguinte de roteamento interno da célula. Após o algoritmo encontrar um posicionamento, duas colunas extras são adicionadas às extremidades da célula para facilitar o roteamento na etapa seguinte do fluxo.

Algumas métricas possuem uma contribuição maior do que outras para a qualidade geral do posicionamento. Isto foi solucionado ponderando a contribuição de cada métrica para o cálculo do valor final da função de custo. Na implementação atual, o peso de cada métrica usada para este cálculo foi parametrizado. Empiricamente, obtivemos um resultado satisfatório utilizando os seguintes pesos:

$$W = 100(4w_{gm} + 4w_{md} + 3w_w + 2w_g + w_{wl}) + w_{ld}$$

onde w_{gm} é o número de *gate mismatches*, w_{md} é a densidade máxima de conexões, w_w é a largura da célula (em número de colunas), w_g é o número de *gaps*, w_{wl} é o comprimento total das conexões, w_{ld} é o somatório da densidade local de conexões e W é o custo total a ser minimizado. Os pesos maiores para w_{gm} e w_{md} se devem por eles estarem diretamente relacionados com a dificuldade de roteamento, podendo impedir os circuitos de serem roteados com sucesso. Em modelos de biblioteca com menor altura de célula, e por consequência menor número de trilhas, pode ser necessário aumentar o peso de w_{md}.

Figura 4.12: Exemplo de conversão de um posicionamento na estrutura de dados de roteamento com os custos associados (ZIESEMER et al., 2014b).



4.6.3.2 Função de Perturbação

A função de perturbação tem o objetivo de expandir o espaço de busca de forma que novas opções sejam experimentadas a cada iteração do algoritmo de TA.

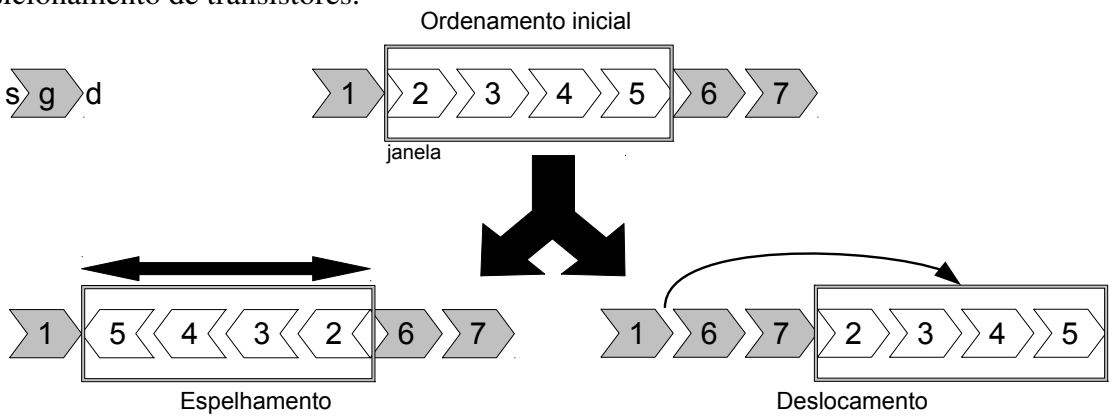
Para encontrar incrementalmente novas soluções, o algoritmo desenvolvido modifica o posicionamento fornecido, movendo um conjunto contínuo de transistores das listas P ou N (ou ambas juntas) a cada iteração, conforme exemplo da Figura 4.13. Para cada lista selecionada, uma janela de tamanho menor ou igual ao número de transistores existentes na difusão é selecionada. Dentro desta janela, dois tipos de movimentos com iguais chances de execução podem ser escolhidos. O primeiro movimento realiza a operação de deslocamento movendo todos os transistores da janela em direção ao início ou ao final da lista de transistores. O segundo movimento realiza a operação de espelhamento onde todos os transistores dentro da janela têm sua ordem e orientação invertida. Todos os parâmetros são aleatoriamente selecionados para uma melhor exploração do espaço de soluções.

4.6.4 Notas

O método de caminho de Euler, apesar de extremamente rápido na maneira como foi implementado, possui grande dificuldade na adoção de métricas além das de número de *gaps* e número de *gates* desalinhado. Outro problema que surge é como modificar o algoritmo para tratar de transistores *dummies* (sem par) sem aumentar significativamente a complexidade do posicionador. Foram feitas algumas tentativas ao longo deste trabalho, mas elas faziam com que células com redes não complementares com cerca de 20 transistores demorassem horas para executar. Por isto esta estratégia foi descontinuada.

O algoritmo de TA possui a facilidade de poder agrupar diversas medidas de qualidade e também de suportar virtualmente qualquer estrutura de célula. Apesar do tempo de execução do posicionamento ser, de uma forma geral, maior do que utilizando o algoritmo

Figura 4.13: Movimentos implementados na função de perturbação do algoritmo de posicionamento de transistores.



de caminho de Euler, esta situação se inverte quando a célula necessita mais de 2 *gaps*. Isto se deve à complexidade elevada do algoritmo que tem de testar todas as possibilidades de quebras de difusão para encontrar o ordenamento ótimo dos transistores.

Os melhores resultados do algoritmo de TA foram obtidos quando o algoritmo era executado com um *threshold* elevado (para forçar a criação de uma solução inicial aleatória). Uma tentativa de utilizar o resultado do algoritmo de caminho de Euler como solução inicial e em seguida executar o TA com um *threshold* inicial reduzido foi realizada, mas os resultados logo mostraram que o algoritmo tende a convergir rapidamente para mínimos locais e os resultados não eram tão bons quando comparados com a solução inicial aleatória e *threshold* elevado.

Em (ZIESEMER; LAZZARI; REIS, 2007) foi feita uma comparação do número de *gaps* gerado entre o algoritmo de posicionamento de transistores do ASTRAN e o de Iizuka (IIZUKA; IKEDA; ASADA, 2004) para um conjunto de células. Os resultados, reproduzidos na Tabela 4.2, mostram que a implementação do ASTRAN obteve o mesmo número de quebras de difusão para todos os circuitos em lógica complementar e ainda conseguiu posicionar células em lógica não complementar, o que não é suportado pela outra ferramenta. Adicionalmente, foi possível detectar redução no comprimento das conexões de até 15% uma vez que a implementação de Iizuka não possui este objetivo na função de custo.

4.7 Roteamento

Para realizar o roteamento, primeiramente é necessário converter o posicionamento fornecido em uma estrutura de dados que contenha: uma representação das partes do circuito que precisam ser conectadas e os lugares por onde podem passar conexões - de acordo com o estilo de leiaute definido e os dados obtidos na etapa de estimativa da área da célula. A estrutura escolhida neste trabalho foi um grafo onde os nós representam os terminais dos transistores, pinos de entrada/saída e pontos de articulação das redes de roteamento; e as arestas representam as conexões entre os nós durante a etapa de compactação do leiaute (normalmente mapeados para linhas de polisilício/metal1, contatos e vias).

A Figura 4.14 mostra o modelo de grafo de roteamento utilizado e o que cada conexão representa na célula. Este grafo é construído instanciando os nós de metal 1 e *poly* de acordo com o número de trilhas calculado. Para cada coluna de difusão obtida pelo po-

Tabela 4.2: Comparação do ASTRAN com Iizuka (IIZUKA; IKEDA; ASADA, 2004) no número de *gaps*.

	#Trans.	#Gaps		Width (μm)	Run Time (s)
		Our tool	Iizuka		
INV	2	0	0	2.8	0.08
BUFFER	4	0	0	5.6	0.32
MUX2:1 (PTL)	4	1	-	7	0.23
NAND2	4	0	0	4.2	0.28
MUXI2:1 (PTL)	6	1	-	5.6	0.66
NOR3	6	0	0	5.6	0.71
MUX4:2 (PTL)	8	1	-	11	13.01
NOR4	8	0	0	8.4	1.64
OAI211	8	0	0	8.4	2.80
XOR2	10	0	-	8.4	0.94
MUX2	12	0	-	11.2	2.96
OA33	14	0	-	12.6	14.96
XNR30	20	3	-	16.8	13.82
AOI444	24	1	-	21	71.44
OAI444	24	1	-	21	59.00
MUX41	26	2	-	21	55.32
FAD1	28	1	1	21	85.61
JK1	34	3	-	30.8	229.20

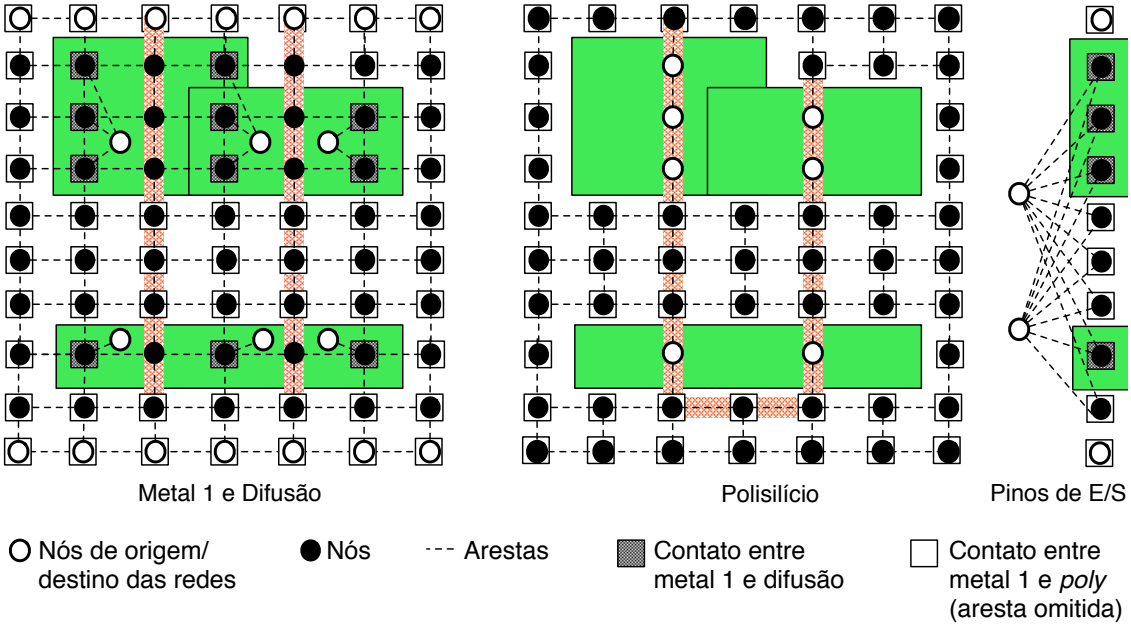
sicionador, dois nós extras são criados para representar as difusões P e N dos transistores (que também são marcados como pertencentes a estas respectivas redes). Arestas são adicionadas para conecta-los aos nós das trilhas em metal 1 sobre os transistores, de acordo com a largura da difusão. Para as colunas de *gate*, os nós de *poly* que ocupam o lugar dos transistores são marcados como pertencendo à rede do *gate*. Nós em metal 1 cuja posição estimada se interseccione ou viole a distância mínima para as redes de alimentação, são marcados como pertencentes a respectiva rede de alimentação. Isto é feito para que outras redes não utilizem estes nós para realizar suas interconexões, uma vez que isto violaria regras de espaçamento mínimo. O resultado do roteamento deste grafo representa a forma como as interconexões da célula devem ser realizadas.

Pesos diferenciados são dados a determinadas arestas para priorizar o uso de conexões com menor resistividade e/ou que tenham menor impacto de área. De uma forma geral, arestas que representam vias/contatos possuem peso maior que as de polisilício, que por sua vez possuem peso maior que as de metal 1.

Uma das otimizações implementadas foi o aumento do custo de realizar conexões verticais em metal 1 na região das trilhas internas das colunas de difusão, ao mesmo tempo em que diminui o custo de realizar conexões horizontais em metal 1 sobre os transistores. A razão para esta alteração é que em muitas situações a largura da célula é determinada pelo número de conexões verticais nesta região, como mostra a Figura 4.15 (a). Com esta otimização, o roteador dá preferência por conectar horizontalmente sinais da mesma rede, mesmo percorrendo uma distância maior, como mostrado em (b), resultando em uma célula com menor largura.

Convém notar que apesar desta modificação ter reduzido a largura de algumas células em até 26% (como no caso de uma AND2), este comportamento pode não ser desejável em células de maior complexidade de roteamento (onde o número excessivo de conexões horizontais pode impedir o leiaute de ser gerado). Neste caso, foi empiricamente definido

Figura 4.14: Modelo de grafo utilizado para roteamento interno da célula.



que esta otimização não seja utilizada em células que possuem densidade máxima de conexões horizontais maior que 50% do número de trilhas horizontais disponíveis. A razão para este número conservador é que conexões verticais (como por exemplo: conexões com a alimentação ou entre transistores P e N) podem ocupar várias trilhas, inserindo obstáculos as conexões horizontais, e contam somente uma vez ao computar a densidade durante a etapa de posicionamento dos transistores.

Para efetuar o roteamento da célula, foi utilizado uma versão modificada do algoritmo de roteamento baseado em negociação de congestionamento chamado PathFinder, detalhado a seguir.

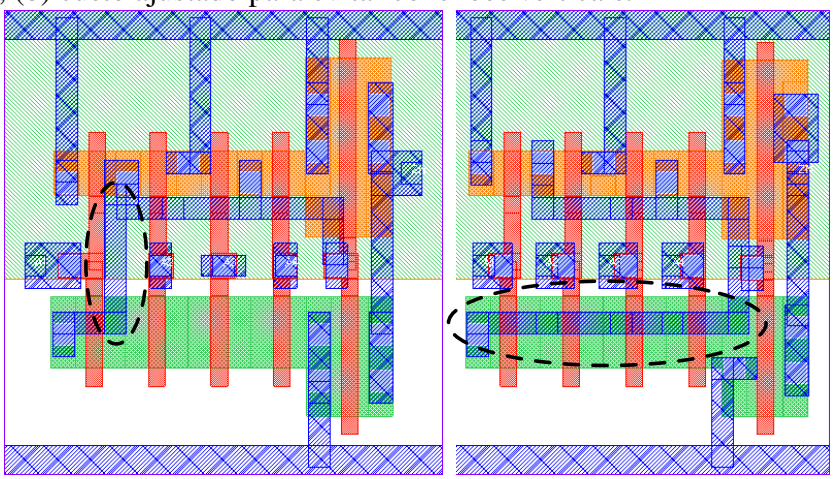
4.7.1 O algoritmo para roteamento PathFinder

O algoritmo PathFinder (MCMURCHIE; EBELING, 1995) foi originalmente concebido para a realização de roteamento em FPGAs mas esta técnica também pode ser aplicada para solução de outros problemas de roteamento. Por ser um algoritmo baseado em negociação de congestionamento, ele possui como característica a capacidade das redes competirem por nós que estejam congestionados. Redes com maior dificuldade em traçar caminhos alternativos para realizar suas conexões, vencem a disputa com as demais, produzindo segundo o autor, um resultado melhor globalmente - quando comparado à outros algoritmos estilo *rip-up and re-route* - e próximo ao ótimo (em média 4,5% de acréscimo em *delay* com relação ao mínimo teórico - ignorando-se os conflitos - segundo o autor).

O roteador é composto de duas partes semi-independentes: um roteador de sinal e um roteador global. O roteador de sinal realiza a conexão entre os nós pertencentes à mesma rede. Já o roteador global chama o roteador de sinal diversas vezes para conectar todas as redes do circuito e ajusta o custo de compartilhamento dos nós baseado na demanda dos sinais àquele recurso.

O roteamento de sinal é feito utilizando o algoritmo de Dijkstra (DIJKSTRA, 1959), recebendo como entrada uma lista com os nós origem e o grafo de roteamento. Este algoritmo foi escolhido pois funciona melhor com grafos valorados do que por exemplo

Figura 4.15: Leiaute de uma célula NAND4. (a) custo normal das arestas no grafo de roteamento; (b) custo ajustado para evitar conexões verticais.



o algoritmo de Lee (LEE, 1961). Também foi preterido com relação ao A* (HART; NILSSON; RAPHAEL, 1968) pela dificuldade de se estimar o custo entre os nós no modelo de grafo de roteamento utilizado. O algoritmo de Dijkstra executa uma busca em largura, expandindo primeiro os nós de menor custo acumulado em relação à origem, até encontrar o primeiro nó pertencente à mesma rede. O roteamento de redes com grau maior que dois é feito conectando um nó de origem ao seu mais próximo na primeira iteração. Nas iterações seguintes é utilizado como origem a lista de todos os nós da árvore parcial de roteamento. Estes então são expandidos simultaneamente a procura do que estiver mais próximo (em termos de custo), até que não haja mais nenhum nó pertencente à rede fora da árvore de roteamento.

O custo de usar um determinado nó n durante uma iteração do roteador de sinal é dado pela fórmula:

$$Cn = (Bn + Hn) * Pn$$

onde Bn é o custo base da aresta que chega ao nó n, Hn é o histórico de congestionamento de n em iterações passadas, Pn é a quantidade de sinais atualmente utilizando n e Cn é o custo final de utilizar n.

Durante a primeira iteração, Pn é inicializado com 1, assim nenhuma penalidade é imposta pelo uso de n independente de quantos sinais ocupem o nó. Nas iterações seguintes, Pn é incrementado gradualmente, dependendo de quantos sinais ocupam n, fazendo com que algumas redes desistam e encontrem uma outra rota de menor custo.

A chave do algoritmo é o fator Hn. A cada iteração que n é compartilhado, Hn é incrementado lentamente. Seu efeito é de permanentemente aumentar o custo da utilização dos nós congestionados de forma que rotas alternativas sejam tentadas.

A métrica Pn é importante para acelerar a execução do algoritmo. Ela insere um fator de ordenamento na realização das conexões - servindo como um critério de desempate entre redes que disputam o mesmo nó e também diminuindo as chances de duas ou mais conexões desistirem do nó ao mesmo tempo. Entretanto, um acréscimo muito abrupto de Pn pode fazer com que as redes desistam muito rapidamente de nós congestionados, eliminando a competição e tornando o algoritmo demasiadamente sensível à ordem de realização das conexões, tal como o esquema de *rip-up and re-route* padrão.

O roteamento termina quando todas as redes forem roteadas sem que ocorra nenhum conflito entre elas ou, em caso de falha, quando atingir um número limite de tentativas sem sucesso.

Detalhes do algoritmo são mostrados abaixo:

```
01:Enquanto haver recursos compartilhados
02:  Repete para para cada rede Ri (rot. global)
03:    Desfaz a árvore de roteamento RTi (rot. de sinal)
04:    Inicializa RTi com o nó fonte de Ri
05:    Repete até que todos os nós de Ri sejam encontrados
06:      Inicializa priority queue PQ com RTi a custo 0
07:      Repete até que um novo nó de Ri seja encontrado
08:        Remove o nó de menor custo m da PQ
09:        Repete para todos os fanouts n de m
10:          Insere n em PQ com custo Cn + Pim
11:        Fim
12:      Fim
13:      Repete para cada nó n no caminho de volta para RTi
14:        Atualiza Cn
15:        Insere n em RTi
16:      Fim
17:    Fim
18:  Fim
19:Fim
```

onde Pim é o custo acumulado do caminho da RTi até o nó m.

4.7.2 Otimizações do roteador

Após a implementação da primeira versão do roteador em (ZIESEMER; LAZZARI; REIS, 2007), os resultados - apesar de satisfatórios - mostraram que o método podia ser melhorado. A qualidade do roteamento interno da célula é muito importante para gerar uma célula com menor área e melhores características elétricas. Por esta razão, otimizações foram feitas tanto para acelerar o algoritmo quanto para melhorar a qualidade do resultado.

4.7.2.1 Aceleração do PathFinder

Uma técnica sugerida pelo autor da ferramenta PathFinder é a aceleração do algoritmo, roteando somente redes que apresentaram conflitos na iteração anterior. Isto pode ser feito pois, como não há modificação no custo dos nós, o roteador tende a rotear a rede novamente pelo mesmo caminho da iteração passada. A diferença em relação a implementação original é que a manutenção da rede faz com que o fator Pn, sensível à ordem, seja diferente, podendo modificar o resultado final. No entanto, a qualidade do resultado tende a ser a mesma.

A alteração proposta pelo autor com relação ao algoritmo original, é descrita abaixo:

```
02:  Repete para cada rede Ri não roteada ou congestionada
```

Com esta modificação, foi possível perceber um ganho significativo de velocidade, principalmente nas últimas iterações, onde a quantidade de redes congestionadas (as únicas que precisam ser roteadas novamente) é menor.

Tabela 4.3: Resultados obtidos para roteamento *intra-cell* entre o roteador antigo do ASTRAN (ZIESEMER; LAZZARI; REIS, 2007) e o novo, com as modificações utilizadas para aceleração.

	Custo			Tempo		
Célula	Antigo	Novo	Dif. (%)	Antigo (s)	Novo (s)	Dif. (%)
MUX2x1	2956	2945	-0,37	0,16	0,12	-28,83
NAND2	1739	1739	0	0,21	0,21	-0,95
ADD22	3643	3656	0,35	0,42	0,12	-71,94
XNR30	3776	3769	-0,18	0,32	0,15	-53,25
NOR3	2649	2609	-1,51	0,46	0,13	-72,30
OAI444	10352	10309	-0,41	49,95	1,27	-97,46
OAI422	5276	5288	0,23	0,20	0,20	-2,99
XCFF	5543	5538	-0,09	2,68	0,46	-82,72
CCMOS	1729	1729	0	0,19	0,19	-1,58
OAI311	3847	3847	0	0,15	0,08	-50,65
NOR33	4246	4236	-0,24	1,04	0,21	-79,42
TOTAL			-0,20			-46,27

Outra modificação feita em relação ao algoritmo original foi a troca da função de custo conforme sugestão proposta por Markov em (ROY; MARKOV, 2007):

$$Cn = Bn + (Hn * Pn)$$

Durante os testes esta função apresentou resultados ligeiramente melhores que a original do algoritmo PathFinder principalmente em tempo de execução.

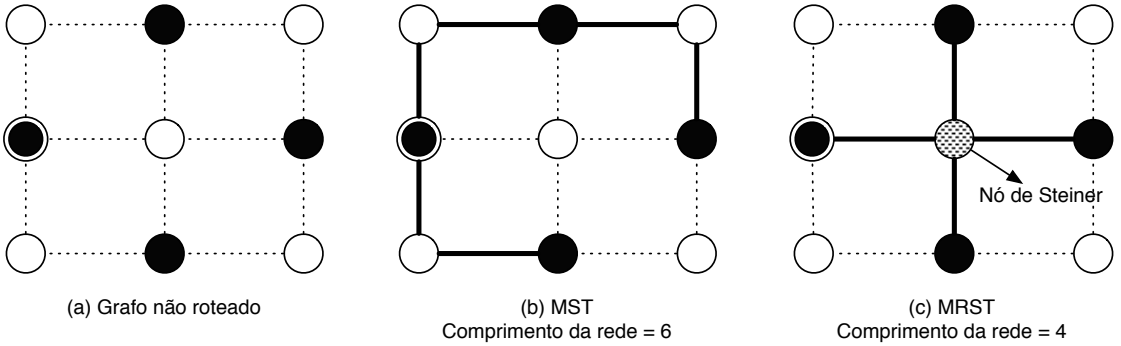
A Tabela 4.3 apresenta os resultados obtidos com a comparação do novo algoritmo de roteamento acelerado, e utilizando a nova função de custo, com a sua implementação original em (ZIESEMER; LAZZARI; REIS, 2007), para um conjunto de portas lógicas. Foi possível observar um aumento consistente na velocidade de execução em todos os circuitos, com uma média de aproximadamente 46%. Houve também uma redução média no custo do roteamento de aproximadamente 0,2%, embora nem todos os circuitos tenham apresentado melhora.

4.7.2.2 Otimização com Iterated 1-Steiner

O roteador de sinal, apesar de extremamente rápido e capaz de produzir resultados satisfatórios, pode obter resultados ainda melhores com a utilização de nós de *Steiner*. Esses nós, quando adicionados à rede original, têm a propriedade de reduzir o custo de interconexão levando o roteador de sinal a maximizar o número de caminhos reaproveitados na busca de nós terminais da mesma rede. Esta redução é ilustrada na Figura 4.16 onde para o mesmo grafo (a) é mostrado um exemplo de árvore mínima de expansão (MST) (b) e também de uma árvore mínima retilínea de *Steiner* (MRST) (c), que apresenta menor comprimento das conexões devido a inserção do nó de *Steiner* à árvore de roteamento.

O problema de encontrar a árvore mínima de Steiner tem sido extensivamente estudado e já foi provado como sendo NP-completo em (GAREY; JOHNSON, 1977). Heurísticas precisam ser usadas para resolver o problema em tempo polinomial.

Figura 4.16: Comparação entre roteamento antes (a), utilizando MST (b) e com a inserção de nós de *Steiner* (c) (KAHNG; ROBINS, 1990).



A abordagem utilizada para adicionar nós de Steiner no roteador intracell foi a utilização do algoritmo *Iterated 1-Steiner Node* (KAHNG; ROBINS, 1990), que calcula iterativamente o melhor nó de *Steiner*, adicionando-o à rede. A definição do algoritmo é mostrada abaixo:

Dado um conjunto de nós $N = \{n_1, ..., n_n\}$ pertencentes a um grafo e uma rede R, o *1-Steiner point* é o nó n do grafo tal que o custo da árvore mínima de expansão $MST(R \cup n)$ é minimizado, e $MST(R \cup n) < MST(R)$. A técnica consiste em iterativamente calcular os *1-Steiner points*, e adicioná-los no conjunto de nós de Steiner S. Cada nó adicionado, faz com que o custo de $MST(R \cup S)$ diminua. O algoritmo termina quando não houver nenhum nó n tal que $MST(R \cup S \cup n) < MST(R \cup S)$.

Por ter sido utilizado um grafo valorado, o custo da MST foi calculado pela soma dos custo de todos os arcos utilizados pela arvore de roteamento da rede.

O autor propõe, para acelerar o algoritmo, que sejam utilizados como candidatos a Steiner somente os nós cujo grau seja maior que 2. Isto pode ser feito pois nós com grau menor que 3 não podem gerar derivações para conectar a outros nós, como demonstrado por Hanan (HANAN, 1966).

Devido à complexidade elevada, não foi possível utilizar o algoritmo de *Iterated 1-Steiner Node* em conjunto com o roteador de sinal. A solução encontrada foi executar ele somente após o roteador global ter encontrado uma solução factível para o roteamento *intra-cell*, como pós-processamento. A partir desta solução, cada rede do circuito roteado é desfeita, uma de cada vez, mantendo-se todas as demais como obstáculos. Para cada rede, o algoritmo de roteamento de sinal (modificado para utilizar a técnica de *Iterated 1-Steiner*) é executado. Ao final de cada iteração, o algoritmo verifica se alguma rede sofreu alteração. Caso isto tenha ocorrido, uma nova iteração é executada percorrendo todas as redes novamente. Isto é feito pois uma modificação em uma rede pode liberar recursos de roteamento e eliminar obstáculos para as demais. O algoritmo termina quando todas as redes são percorridas sem sofrer qualquer modificação. O algoritmo implementado neste trabalho é ilustrado na Figura 4.17. A Figura 4.18 mostra um exemplo de execução.

Os resultados obtidos com a otimização do roteamento com *Iterated 1-Steiner* são mostrados na Tabela 4.4. Como esperado, todos os circuitos melhoraram ou mantiveram seu custo de interconexões inalterado, sendo que a redução média obtida foi de 0,12%. Esta redução é significativa uma vez que, no modelo de grafo utilizado, os arcos que representam os pinos de E/S possuem custo relativamente elevado com relação aos demais (aproximadamente 2 ordens de grandeza maior), para evitar inserção de mais de um pino para uma mesma rede. Desta forma, melhoras no roteamento em arcos de menor custo

Figura 4.17: Fluxograma do algoritmo de *Iterated 1-Steiner Node*.

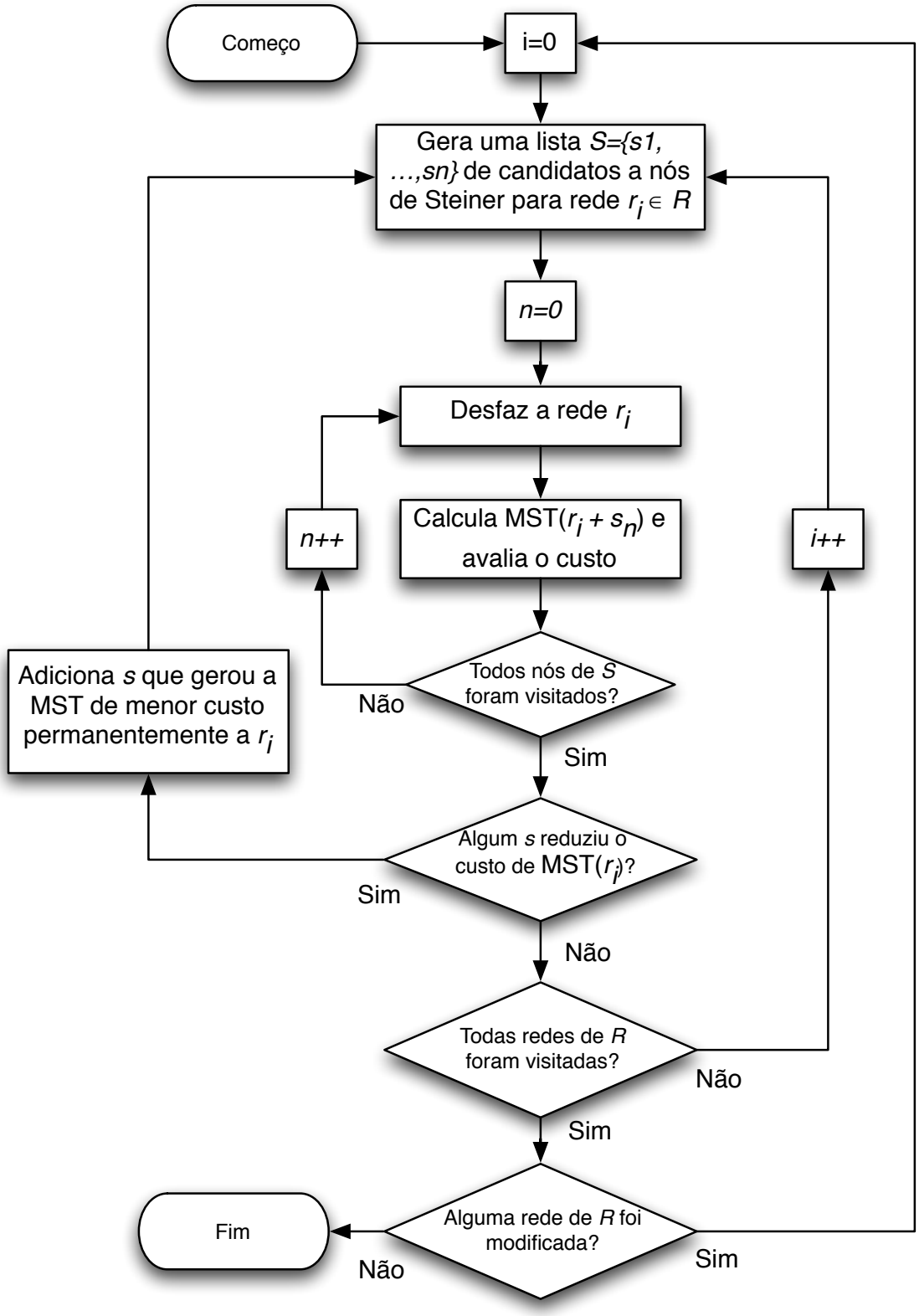


Figura 4.18: Exemplo de execução do algoritmo de otimização do roteamento com *Iterated 1-Steiner Node*.

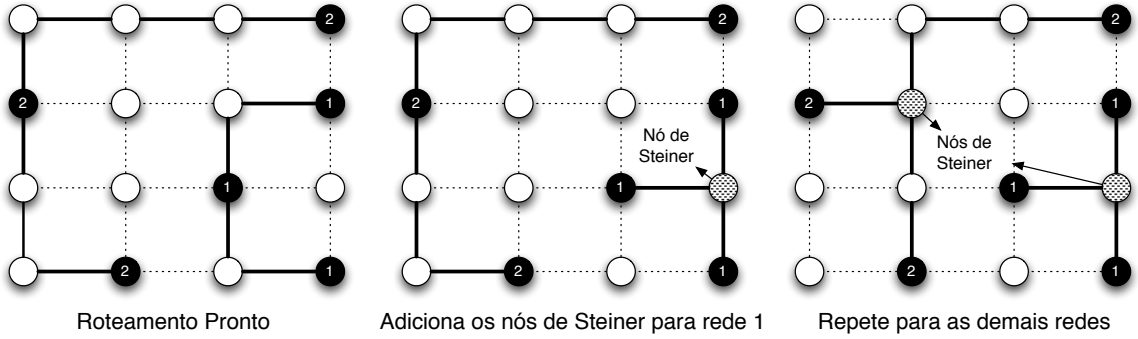


Tabela 4.4: Resultados obtidos para roteamento *intra-cell* com e sem otimização por *Iterated 1-Steiner*.

	Custo			Tempo		
Célula	Original	Otim.	Dif. (%)	Original	Otim.	Dif.(%)
MUX2x1	2945	2926	-0,65	0,12	0,24	105
NAND2	1739	1739	0	0,21	0,45	118
ADD22	3656	3651	-0,14	0,12	0,47	303
XNR30	3769	3763	-0,16	0,15	0,52	246
NOR3	2609	2609	0	0,13	0,27	117
OAI444	10309	10304	-0,05	1,27	3,39	167
OAI422	5288	5288	0	0,20	0,58	195
XCFF	5538	5532	-0,11	0,46	1,01	117
CCMOS	1729	1729	0	0,19	0,37	99
OAI311	3847	3842	-0,13	0,08	0,44	483
NOR33	4236	4226	-0,23	0,21	2,12	892
TOTAL			-0,12			195

são amortizados. Exemplo desta melhora é mostrado no leiaute da Figura 4.19 onde a adição de um nó de steiner permitiu a redução da largura da célula.

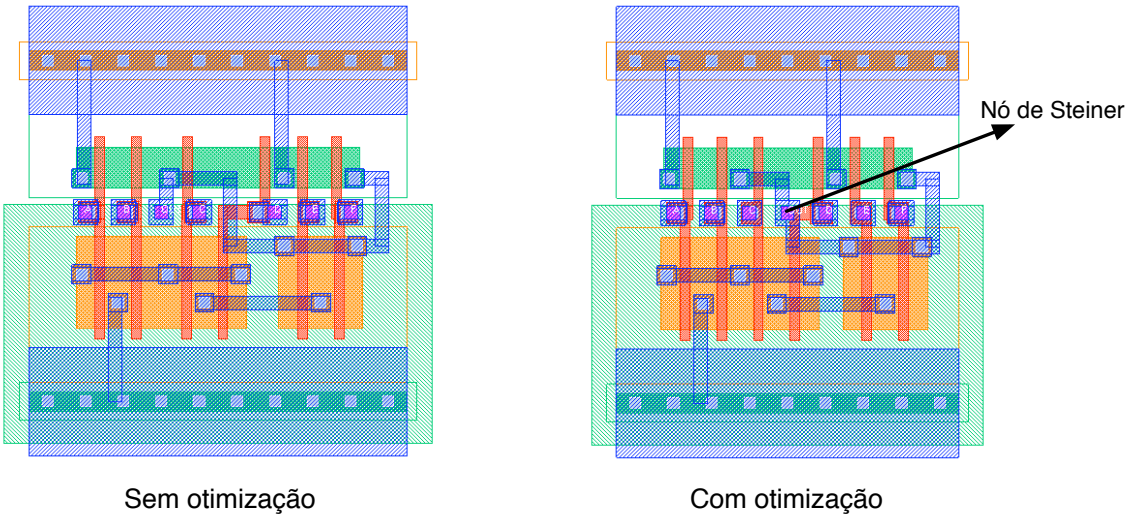
O tempo de execução, mesmo tendo aumentado em até 900% no pior caso, não levou mais do que 4 segundos para terminar, não sendo portanto um problema para os circuitos testados.

4.7.2.3 Arborescência

Uma possível melhoria que ainda não foi implementada no ASTRAN, é a priorização de arborescência ao invés do comprimento das conexões. Arborescência permite minimizar a distância máxima entre a origem do sinal (pinos de entrada, difusão e redes de alimentação) e cada um dos destinos das redes (*gate* dos transistores e pinos de saída). Esta técnica pode diminuir a resistência e atraso dos sinais, mas por outro lado também pode aumentar a densidade de conexões, capacitância e largura da célula. Um possível *trade-off* poderia ser priorizar arborescência somente em redes críticas e de alimentação.

Esta técnica é detalhada no Apêndice A onde foi implementada para roteamento detalhado (ao invés de *intra-cell*).

Figura 4.19: Leiautes de uma célula com e sem otimização do roteamento interno pelo algoritmo de *Iterated 1-Steiner*.



4.8 Compactação

A compactação é o último passo para geração do leiaute do circuito. As geometrias que compõem o desenho da célula são instanciadas e posicionadas de acordo com o resultado obtido nas etapas de posicionamento e roteamento. Além disto, restrições são adicionadas para tornar o leiaute compatível com o estilo de leiaute definido e algoritmos de minimização são utilizados para diminuir a largura da célula e o tamanho das conexões.

O resultado das etapas anteriores de geração é suficiente para prover uma informação relativa da posição de cada polígono que forma o leiaute da célula. Isto permite que este problema possa ser modelado como um conjunto de inequações que pode ser resolvido através de um resolvedor de equações lineares. A técnica utilizada neste trabalho para resolver o problema de compactação de leiaute com programação linear com inteiros (ILP) ou programação linear mista com inteiros (MILP), é explicada a seguir.

4.8.1 Programação Linear Mista com Inteiros

A área de MILP tem apresentado avanços significativos recentemente, tais como: pré-resolução, planos de corte, heurísticas, detecção de simetria, paralelismo, entre outros. Gurobi (GUROBI OPTIMIZER, 2014) alega ser 24x mais rápido que resolvedores de código aberto (gratuitos). Isto permitiu a utilização de modelos mais complexos (ZIESEMER; REIS, 2014) do que os utilizados anteriormente em (ZIESEMER; LAZZARI; REIS, 2007). Neste trabalho foi possível modelar a maior parte das regras condicionais, existentes nos processos recentes de fabricação de semicondutores, e obter o resultado ótimo para estes modelos em um tempo razoável (menos de 24h para células com até 40 transistores). Além da síntese de células lógicas (KARMAZIN; OTERO; MANOHAR, 2013), esta metodologia pode ser aplicada a outros problemas recentes tais como migração de células digitais (FU et al., 2009) e analógicas (SAID et al., 2011).

Tendo o posicionamento dos transistores e o roteamento interno da célula prontos, é possível instanciar as estruturas (retângulos) que formam o leiaute do circuito. Inicialmente, estas estruturas são desprovidas de posição e tamanho. Em seguida, uma variável é associada a cada borda dos elementos do leiaute que necessitam ser compactados (borda esquerda e direita para compactação horizontal e borda superior e inferior para compac-

tação vertical). Uma vez que a posição relativa dos retângulos é conhecida, é possível descrever a relação entre as variáveis na forma de inequações lineares. As regras que definem os espaçamentos entre estas variáveis são descritas no arquivo de tecnologia.

4.8.1.1 Formulação do Problema

Compactação de leiaute baseado em bordas pode ser formulado como um problema de programação linear (LP) criando restrições para cada elemento do leiaute. Um modelo de LP possui um objetivo que é maximizar ou minimizar o resultado de uma função linear seguido por um número de desigualdades:

$$\begin{array}{ll} \text{Minimizar:} & c^T x \\ \text{Sujeito a:} & Ax \leq b \\ & x \geq 0 \end{array} \tag{4.1}$$

onde x representa o vetor de variáveis (a ser determinado os valores), c representa o peso de cada variável, b é um vetor de coeficientes para cada desigualdade e A é uma matriz de constantes.

Regras de projetos podem ser modeladas no formato $x_i - x_j \geq d_{ij}$, onde x_j representa a posição de determinada borda de um elemento, x_i é a posição de outra borda um elemento e d_{ij} é a distância mínima entre as bordas (que podem ser de um mesmo elemento, como a largura mínima, ou de elementos diferentes, como o espaçamento mínimo).

MILP permite forçar algumas das variáveis a terem um valor inteiro, o que é útil para alinhar os pinos à grade de roteamento, como em (ZIESEMER; LAZZARI; REIS, 2007). Variáveis binárias (variáveis inteiras que podem assumir valores entre 0-1) podem ser utilizadas para modelar situações onde determinada restrição pode ser aplicada ou não. Isto permite resolver a classe de problemas de otimização combinatória tais como satisfatibilidade booleana (SAT).

O método desenvolvido neste trabalho consiste em modelar cada uma das regras condicionais utilizando variáveis booleanas. Regras de projeto mutualmente exclusivas podem ser modeladas como:

$$\begin{array}{rcl} x_2 - x_1 & \geq & b_1 r_1 \\ x_2 - x_1 & \geq & b_2 r_2 \\ x_2 - x_1 & \geq & b_n r_m \\ b_1 + b_2 + ... + b_n & = & 1 \end{array} \tag{4.2}$$

onde $r_i (1 \leq i \leq m)$ é uma constante que representa o valor da regra i e $b_j (1 \leq j \leq n)$ é uma variável booleana associada à aplicação condicional da respectiva regra r_i.

Adicionalmente, uma constante de relaxação R pode ser utilizada de forma a permitir que regras inativas sejam "relaxadas" de forma a não interferirem em outras regras:

$$x_2 - x_1 + R \geq b_j (r_i + R) \tag{4.3}$$

Desta forma, quando a regra está ativa ($b_j = 1$), obtém-se:

$$\begin{array}{rcl} x_2 - x_1 + R & \geq & 1(r_i + R) \\ x_2 - x_1 & \geq & r_i \end{array} \tag{4.4}$$

e quando a regra está inativa ($b_j = 0$):

$$\begin{array}{rcl} x_2 - x_1 + R & \geq & 0(r_i + R) \\ x_2 - x_1 & \geq & -R \end{array} \tag{4.5}$$

de outra forma, se r_i não fosse utilizado, isto resultaria em $x_2 - x_1 \geq 0$, sendo x_1 um limitante inferior para x_2.

Estas formulações são a base para a maior parte dos modelos apresentados a seguir.

4.8.2 Compactação do Leiaute em 2D utilizando MILP

Compactação de leiaute em duas dimensões tem a vantagem de lidar com todas as decisões de leiaute simultaneamente, o que pode potencialmente melhorar a qualidade do leiaute. Metodologias em 1D decidem apenas uma dimensão por vez. O problema com esta abordagem é que vários elementos do leiaute podem possuir diferentes fatores de forma (tais como a envoltória de metal sobre contatos e a área mínima) e não podem ser resolvidos de forma ótima pela possibilidade de cair em mínimos locais. Por outro lado, dada a liberdade de movimento adicional quando compactando em 2D, o número de restrições usadas para modelar o espaçamento mínimo cresce significantemente. Isto ocorre porque não apenas elementos adjacentes em todas as direções precisam ser verificados, mas também elementos distantes.

O resolvedor de MILP encontra a solução ótima para os modelos, retornando os valores para cada variável. Se um tempo-limite de execução é definido, o resolvedor retorna a melhor solução válida encontrada até o momento (caso haja) - o que pode ser útil para modelos grandes, onde muitas horas/dias são necessários para encontrar ou provar a otimalidade.

Foi denotado x_{i_a} e x_{i_b} variáveis inteiras representando as bordas da esquerda e direita do elemento i e y_{i_a} e y_{i_b} as bordas inferiores e superiores de i. Variáveis binárias auxiliares começam com b. Os valores das regras de projeto são definidos como constantes e são representadas em maiúsculo como em: $E1M1CT$. A Figura 4.20 mostra as principais regras de projeto suportadas pelo ASTRAN. Mais informações sobre estas e outras regras podem ser encontradas no Apêndice B.

4.8.2.1 Posicionamento dos pinos de entrada/saída

O posicionamento das vias para a camada de metal 2 deve estar alinhado à grade de roteamento, como ilustra a Figura 4.21. Isto pode ser feito forçando que a posição central delas seja múltipla do *pitch* horizontal e vertical da grade de roteamento, levando em consideração o *offset*. A formulação encontrada para resolver este problema foi a seguinte:

$$\begin{aligned} x_{via} - HOFFSET &= x_{via_{int}} HPITCH \\ y_{via} - VOFFSET &= y_{via_{int}} VPITCH \\ int &: x_{via_{int}}, y_{via_{int}} \end{aligned} \tag{4.6}$$

onde $HPITCH$ e $VPITCH$ definem a largura do *pitch* horizontal e vertical da grade de roteamento; $HOFFSET$ e $VOFFSET$ definem o *offset*; x_{via} e y_{via} a posição central da via; e $x_{via_{int}}$ e $y_{via_{int}}$, variáveis auxiliares inteiras.

4.8.2.2 Envoltória de Difusão, Polisilício e Metal nos Contatos

Tecnologias antigas costumavam permitir apenas o estilo de envoltória quadrado nos contatos, que era facilmente modelado devido a sua simetria horizontal e vertical. No entanto, tecnologias recentes passaram a permitir também o estilo retangular, que pode ser desenhado na horizontal ou vertical, aumentando para três o número de possibilidades para cada envoltória de contato.

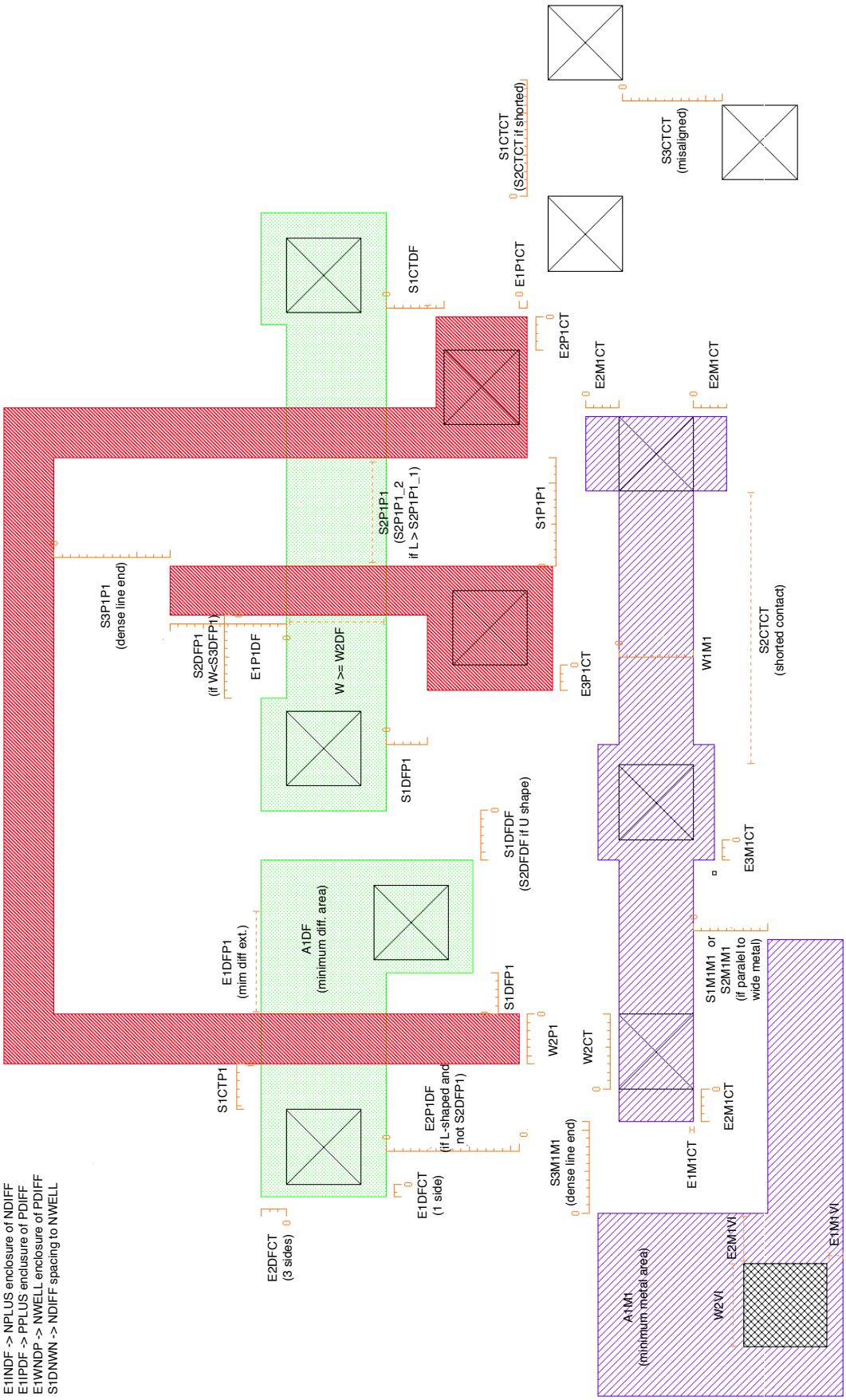


Figura 4.20: Principais regras de projeto suportadas pelo ASTRAN.

Figura 4.21: Alinhamento dos pinos de entrada/saída das células à grade de roteamento.

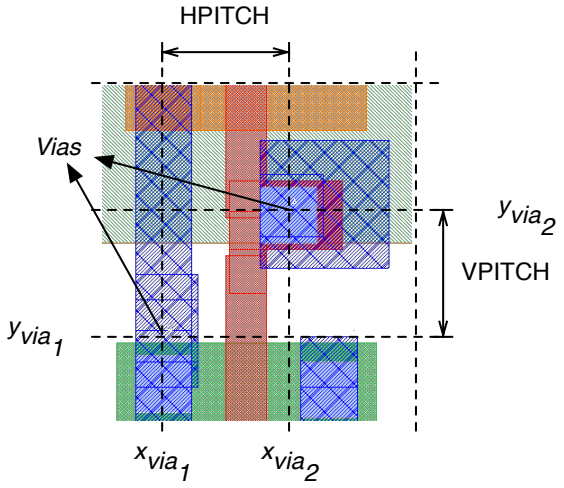


Figura 4.22: Envoltória de metal em um contato.

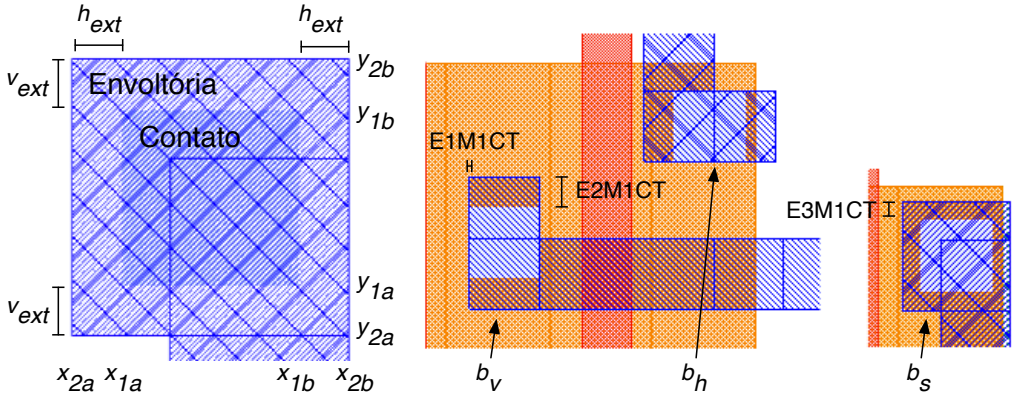


Figura 4.23: Possibilidades de envoltórias de difusão em um contato: (a) estilo horizontal, (b) estilo vertical e (c) estilo vertical com esticamento da difusão.

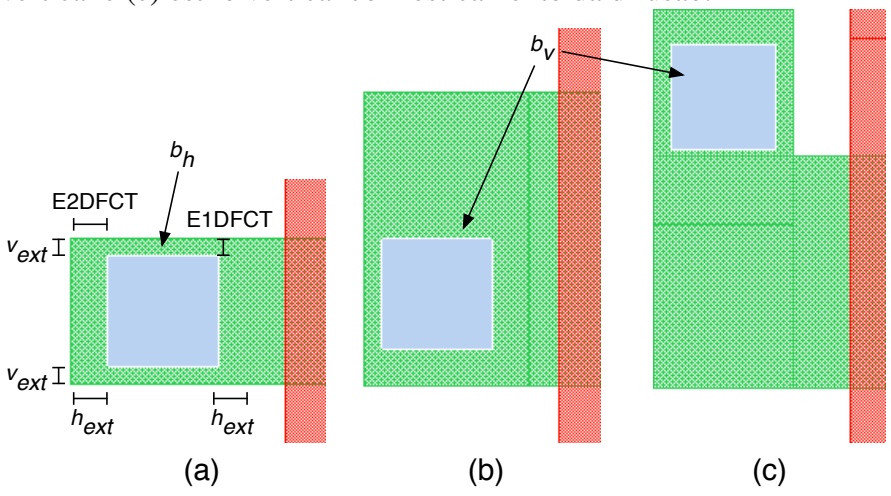
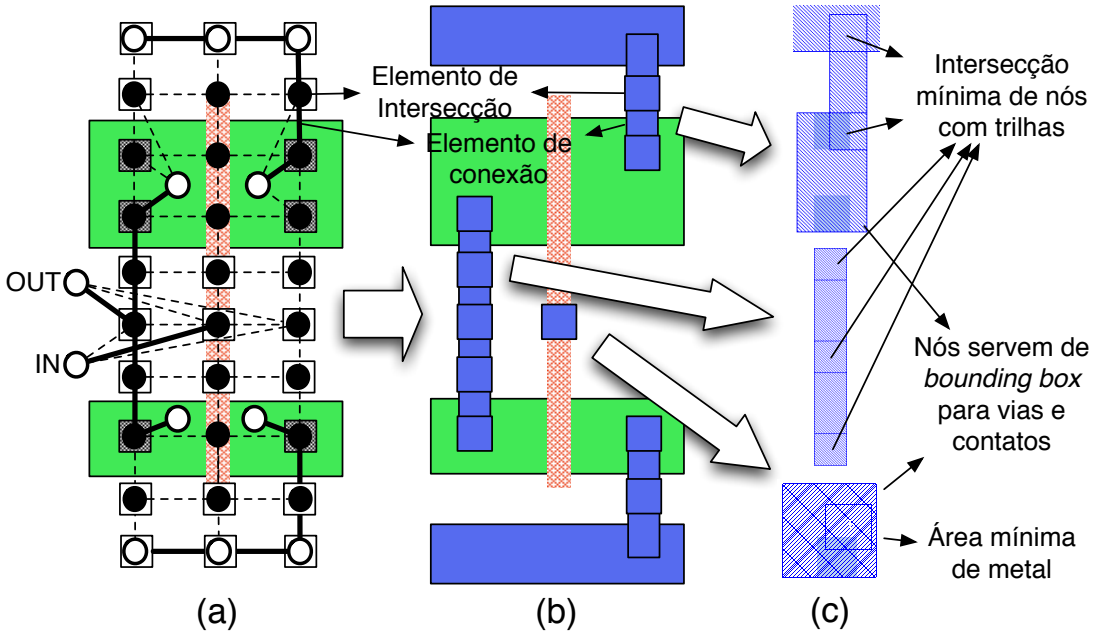


Figura 4.24: Modelo de roteamento. (a) grafo roteado para a camada de metal 1; (b) representação abstrata dos elementos; (c) leiaute final dos elementos em destaque.



Este problema de otimização combinatorial foi modelado no ASTRAN utilizando a fórmula para regras de projeto mutualmente exclusivas citada anteriormente:

$$\begin{aligned}
h_{ext} &\geq b_h E2M1CT \\
h_{ext} &\geq b_v E1M1CT \\
h_{ext} &\geq b_s E3M1CT \\
v_{ext} &\geq b_h E1M1CT \\
v_{ext} &\geq b_v E2M1CT \\
v_{ext} &\geq b_s E3M1CT \\
x_{2a} - x_{1a} &\geq h_{ext} \\
x_{1b} - x_{2b} &\geq h_{ext} \\
y_{2a} - y_{1a} &\geq v_{ext} \\
y_{1b} - y_{2b} &\geq v_{ext} \\
b_h + b_v + b_s &= 1
\end{aligned} \tag{4.7}$$

onde b são variáveis binárias indicando se o estilo retangular horizontal (b_h), vertical (b_v) ou quadrado (b_s) deve ser aplicado; h_{ext} e v_{ext} são variáveis auxiliares que definem, respectivamente, a extensão horizontal e vertical mínima com relação ao contato, de acordo com o estilo selecionado; x e y definem as posições finais de cada borda dos elementos. A Figura 4.22 ilustra o significado de cada uma destas variáveis.

Esta fórmula é flexível o suficiente para permitir que o resolvedor retorne o estilo ótimo para cada situação (de acordo com o modelo utilizado). A Figura 4.23 mostra esta mesma regra aplicada para uma envoltória de difusão em um contato.

4.8.2.3 Espaçamento Mínimo de Polys e Metais

O modelo de roteamento em metal 1 e polisilício utilizado pelo ASTRAN é ilustrado na Figura 4.24. Após o roteamento ter terminado, cada vértice e aresta do grafo pertencentes a uma árvore de roteamento (a) dão origem a um elemento de intersecção e de conexão, respectivamente (b). Cada elemento de intersecção é utilizado como uma

Figura 4.25: Espaçamento entre metais.

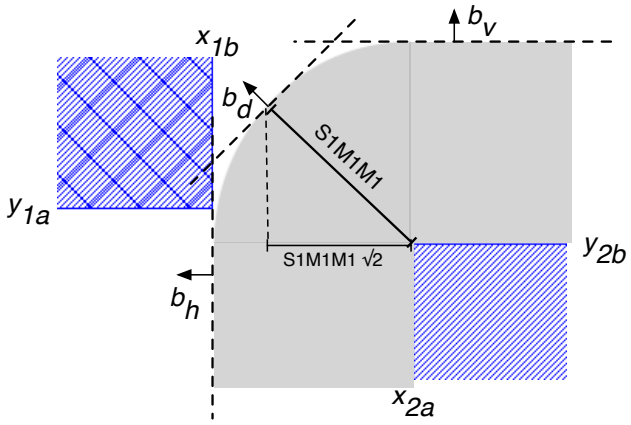


Figura 4.26: Espaçamento mínimo para o nó da rede número 4. (a) Regras de espaçamento 1D e 2D; (b) detalhe do espaçamento 2D sobre um nó; (c) possível problema na utilização do espaçamento 2D e solução.

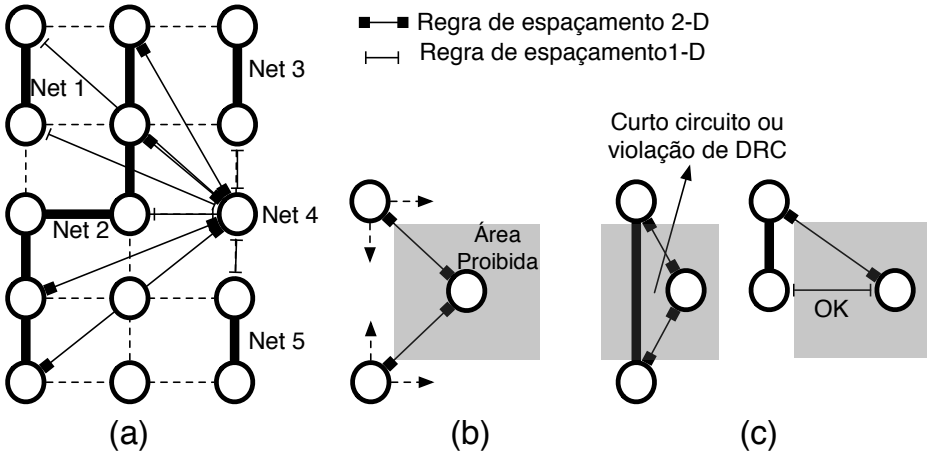
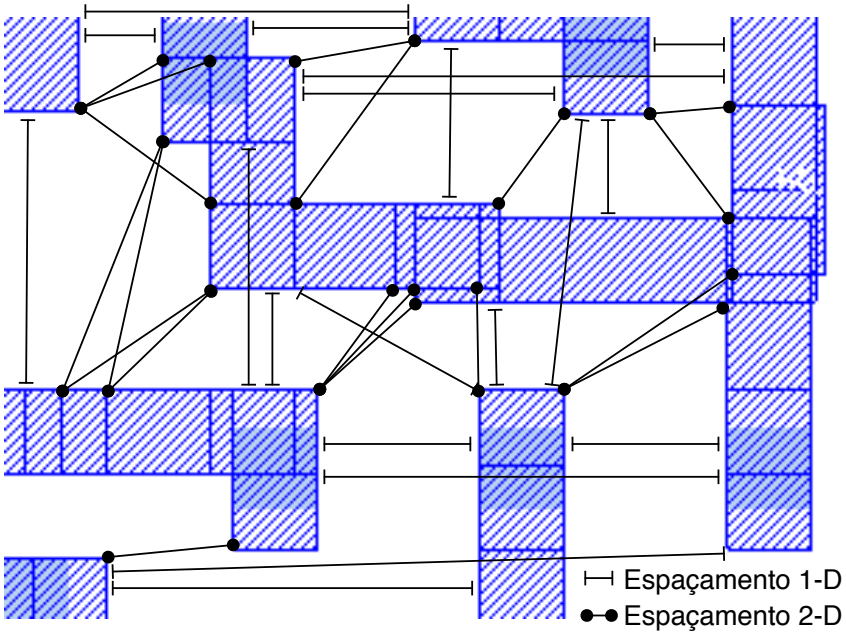


Figura 4.27: Regras de espaçamento 1D e 2D.



bounding box para vias, contatos e também extremidades dos elementos de conexão com vértices adjacentes. Cada final de linha de conexão deve se interseccionar em pelo menos a distância da regra de largura mínima (para materiais daquela camada) com os elementos de conexão (c). Isto permite simplificar o modelo para que somente regras de espaçamento mínimo entre os elementos de intersecção precisem ser inseridas (e não também entre elementos de conexão).

O problema com a modelagem 2D para as regras de espaçamento mínimo é criar um modelo que dê liberdade suficiente para os elementos se movimentarem, ao mesmo tempo em que respeita as regras de projeto e não excessivamente a complexidade. Isto foi conseguido utilizando a variável de relaxação R de forma que regras inativas não imponham restrições às ativas. Isto permite que o resolvedor de MILP escolha a melhor orientação para cada par de elementos: à esquerda ou acima. Adicionalmente, uma terceira orientação foi criada para permitir que os elementos possam ser posicionados na diagonal, dando mais liberdade ao modelo, ilustrado na Figura 4.25. Isto foi feito aproximando a distância mínima, que é uma função quadrática, de uma função linear: $\Delta x + \Delta y \geq 2(S1M1M1/\sqrt{2})$. A formulação utilizada para estas regras é:

$$\begin{array}{rcl} x_{2a} - x_{1b} + R & \geq & b_h(S1M1M1 + R) \\ y_{2a} - y_{1b} + R & \geq & b_v(S1M1M1 + R) \\ (x_{1a} - x_{2b}) + (y_{2a} - y_{1b}) + R & \geq & b_d(2\frac{S1M1M1}{\sqrt{2}} + R) \\ b_h + b_v + b_d & = & 1 \end{array} \tag{4.8}$$

onde b_h, b_v e b_d são variáveis binárias que indicam qual das regras deve ser aplicada.

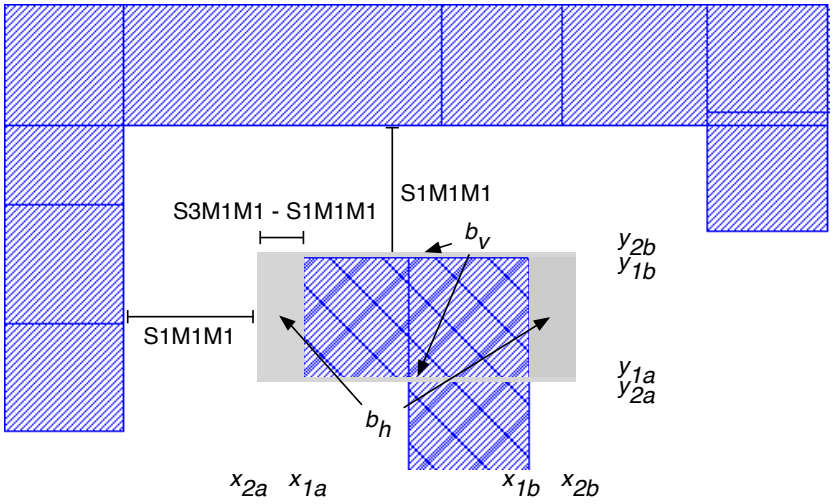
Esta formulação pode ser estendida a todas as laterais dos elementos, para uma máxima otimização. No entanto, para reduzir a complexidade do modelo, foi optado por aplicar esta regra somente sobre elementos que estejam próximos, como ilustra a Figura 4.26 (a). Isto dá liberdade para que estes elementos sejam posicionados em diferentes regiões (b) enquanto elementos distantes (com outros elementos no meio, e que possuem menor probabilidade de obter qualquer tipo de ganho com compactação 2D) utilizam as regras de espaçamento 1D: $x_{2a} - x_{1b} \geq S1M1M1$ para espaçamento horizontal ou $y_{2a} - y_{1b} \geq S1M1M1$ para espaçamento vertical. De forma a evitar que dois vértices adjacentes se movam, produzindo a violação mostrada em (c), a regra 1D também é aplicada para vértices vizinhos na vertical e horizontal. Nenhuma destas regras é aplicada para nós pertencentes a uma mesma rede, como mostrado a Figura 4.27. Isto inseriria restrições desnecessárias, piorando o resultado final da compactação. Por outro lado, a não inserção destas regras pode gerar violações de regras de DRC, mas que pela nossa experiência tem sido simples de serem resolvidas a mão. A mesma metodologia é aplicada tanto para metal 1 quanto para polisilício, difusão e contatos.

4.8.2.4 Espaçamento em Final de Linhas Densas de Metal 1 e Poly

Algumas tecnologias proíbem o uso da regra de espaçamento mínimo em final de linhas densas (com outros elementos da mesma camada próximos) pois curto-circuitos podem ocorrer devido a problemas de litografia. Neste caso, um dos elementos adjacentes deve ter sua distância mínima aumentada.

Este problema foi resolvido virtualmente aumentando o tamanho do elemento de referência (para fins de aplicação de restrições de espaçamento com outros elementos), em dois lados opostos, na medida da diferença entre a regra de espaçamento mínimo e a regra para regiões densas. Desta forma, as modelagens para as regras de espaçamento mínimo, explicadas anteriormente, podem ser utilizadas sem qualquer tipo de modificação,

Figura 4.28: Espaçamento em final de linhas densos.



ao mesmo tempo que garante que nunca dois lados adjacentes utilizarão a regra mínima de espaçamento simultaneamente. O modelo MILP para esta restrição é:

$$\begin{aligned} \delta &= S3M1M1 - S1M1M1 \\ x_{1a} - x_{2a} &\geq b_h\delta \\ x_{2b} - x_{1b} &\geq b_h\delta \\ y_{1a} - y_{2a} &\geq b_v\delta \\ y_{2b} - y_{1b} &\geq b_v\delta \\ b_h + b_v &= 1 \end{aligned} \tag{4.9}$$

onde δ é uma constante definida como a diferença entre a regra de espaçamento em final de linhas densas $S3M1M1$ e a regra de espaçamento mínimo $S1M1M1$; b_h define se o elemento deve ser virtualmente aumentado na horizontal e b_v, na vertical. A Figura 4.28 ilustra este modelo sendo aplicado para a camada de metal 1 de uma célula. A região sombreada representa o espaço extra adicionado pelo modelo.

4.8.2.5 Relaxação da Regra de Espaçamento Mínimo

A restrição de espaçamento mínimo pode ser relaxada para aumentar a confiabilidade do leiaute (para fins de DFM). Este problema poderia ser resolvido aumentando a regra de espaçamento mínimo para todos os elementos. No entanto, esta modificação seria muito restritiva potencialmente aumentando a área da célula. Para que haja um compromisso entre área e DFM, este problema foi resolvido definindo uma borda virtual (a mesma da regra de espaçamento em finais de linhas densas) maior ou igual (em todos os lados) que um valor a ser maximizado até um determinado limite (por exemplo, 10% da regra de espaçamento mínimo). Assim, este espaçamento extra só é inserido caso haja espaço suficiente, sem sacrificar outros quesitos mais importantes.

No entanto, isto não impede de na existência de 3 linhas paralelas, o resolvedor deixar um dos espaçamentos entre os elementos com 10% de espaço extra enquanto o outro fica com 0% (no caso de inexistência de espaço suficiente no leiaute para a adição de ambos espaçamentos extras). Neste caso, o ideal seria deixar 5% de acréscimo em ambos os lados do elemento central, uma vez que a melhora no DFM não é linear. Isto foi feito definindo o mesmo tamanho de borda para ambos lados opostos dos elementos. Desta forma, o resolvedor procura manter este centralizado caso haja elementos próximos nos

Figura 4.29: Relaxação do espeaçamento mínimo para DFM.

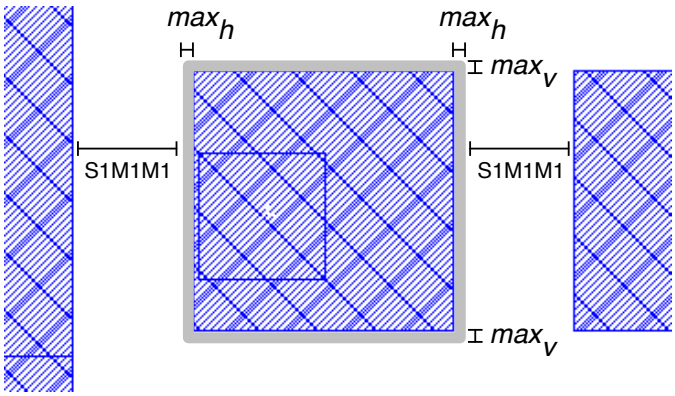
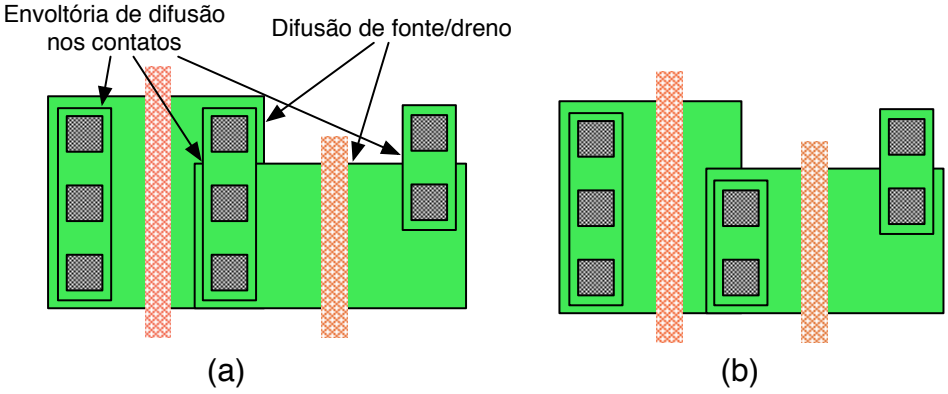


Figura 4.30: Modelo de transistor para compactação. (a) ASTRAN; (b) Proposta de melhoria.



dois lados, como mostra a Figura 4.29. Abaixo é apresentada a formulação para este problema onde as variáveis max_h e max_v representam as bordas a serem maximizadas:

$$\begin{aligned} x_{1a} - x_{2a} &\geq max_h \\ x_{2b} - x_{1b} &\geq max_h \\ y_{1a} - y_{2a} &\geq max_v \\ y_{2b} - y_{1b} &\geq max_v \\ max_h &\leq S1M1M1 * 0.1 \\ max_v &\leq S1M1M1 * 0.1 \\ Maximize &: max_h + max_v \end{aligned} \tag{4.10}$$

4.8.2.6 Modelagem dos transistores

O modelo de transistor utilizado no ASTRAN é mostrado na Figura 4.30 (a). Cada *gate* de transistor é instanciado com sua respectiva difusão de fonte e dreno. Os contatos possuem sua própria envoltória de difusão que é utilizada para mantê-los conectados verticalmente com o transistor. Além disto, a difusão dos contatos também define limites horizontais mínimos para a difusão de fonte e dreno do transistor anterior e/ou do próximo (caso existam). Em (b) é apresentada uma possível melhoria sobre a implementação atual. Durante os testes foi possível perceber que, em raras situações, o modelo atual pode levar a um aumento desnecessário da área da célula. Isto ocorre devido a uma simplificação que não permite que a difusão de fonte e dreno comece/termine no meio da difusão dos

contatos. Esta modificação exige um modelo mais complexo de transistor e de regras de espaçamento para os demais elementos da célula e por isto ainda não foi implementado.

4.8.2.7 Regra Condicional de Extensão do Gate para Difusões em L

Esta é uma regra bastante complexa de modelar uma vez que a extensão do *gate* depende da existência ou não de uma difusão em formato de L (que não pode ser previamente calculado sem impor restrições para a funcionalidade de esticamento vertical da difusão) e do espaçamento desta difusão para o *gate* do transistor. Adicionalmente, a regra de espaçamento entre difusão e *gate* também depende da largura do transistor e não deve ser aplicada caso a difusão não tenha formato de L. Este problema foi modelado conforme mostrado na Figura 4.31 e a formulação mostrada abaixo:

$$
\begin{aligned}
b_{smallW} &= \{1,0\} \\
y_{diffEnc_b} - y_{newDiff_b} &\leq y_{Ldist} \\
y_{lastDiff_b} - y_{newDiff_b} &\leq y_{Ldist} \\
y_{Ldist} &\leq b_{Lshape} R \\
\\
b_{smallW} + b_{Lshape} &\leq b_{applyS2} + 1 \\
b_{Lshape} - b_{applyS2} &\leq b_{applyExtraExt} \\
\\
x_{gate_a} - x_{diffEnc_b} &\geq b_{Lshape} S1DFP1 \\
x_{gate_a} - x_{lastDiff_b} &\geq b_{Lshape} S1DFP1 \\
x_{gate_a} - x_{diffEnc_b} &\geq b_{applyS2} S2DFP1 \\
x_{gate_a} - x_{lastDiff_b} &\geq b_{applyS2} S2DFP1 \\
y_{gate_b} - y_{newDiff_b} &\geq E1P1DF \\
y_{gate_b} - y_{newDiff_b} &\geq b_{applyExtraExt} E2P1DF
\end{aligned}
\tag{4.11}
$$

onde b_{smallW} é uma constante que é pré-definida como 1, caso a largura do transistor for menor que um limiar (definido pelas regras de projeto), ou 0, caso contrário; $y_{lastDiff_b}$ é o tamanho do esticamento vertical da difusão. b_{Lshape} é calculado como 1 sempre que $y_{lastDiff_b}$ for maior do que 0. Após, é feito o uso de lógica booleana com inequações para decidir quando cada regra deve ser aplicada: o espaçamento extra da difusão (representado por $b_{applyS2}$) é aplicado sempre que b_{smallW} e b_{Lshape} forem 1; a extensão extra do *gate* (representada por $b_{applyExtraExt}$) é aplicada sempre que b_{Lshape} valer 1 e $b_{applyS2}$ valer 0. Finalmente, estas regras são aplicadas nos elementos de acordo com o valor definido pelas variáveis binárias: $b_{applyS2}$ e $b_{applyExtraExt}$. Esta formulação é repetida para cada canto de cada transistor da célula.

4.8.2.8 Área Mínima das Conexões em Metal 1 e Difusão

O problema com esta regra de projeto é que área é uma função quadrática. Por ter sido utilizado um resolvedor de MILP, foi necessário aproximar esta regra de uma função linear, como mostra a Figura 4.32, onde foi utilizado o semi-perímetro. Isto permite que diferentes relações de aspecto sejam exploradas para os elementos, reduzindo o número de restrições que podem gerar um leiaute de pior qualidade.

No entanto, ao comparar a curva da função linear com a curva da função quadrática (ideal) para a área mínima, é possível perceber que a primeira se afasta de forma significativa desta última para formas próximas a de um quadrado. Este problema foi contornado adicionando uma outra formulação condicional para geração de elementos quadrados, e

Figura 4.31: Regra condicional de extensão do *gate*.

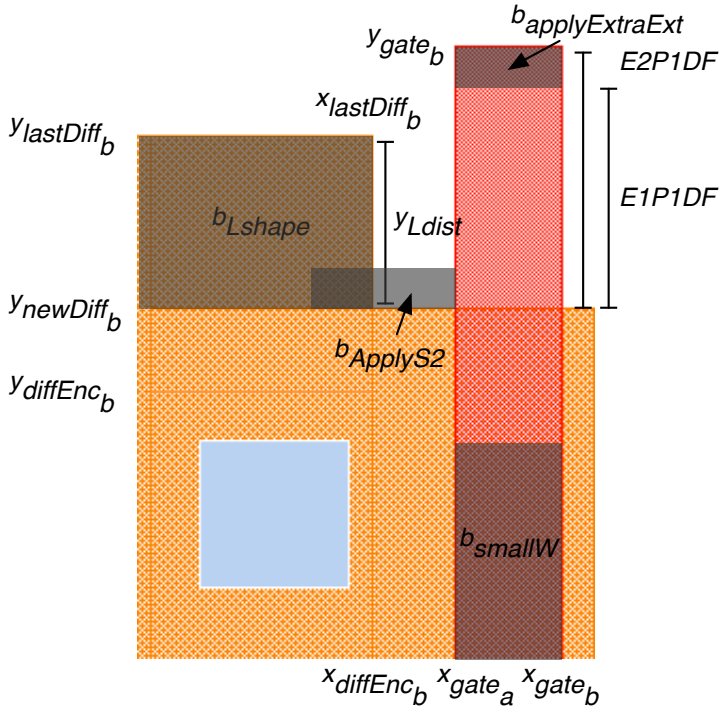
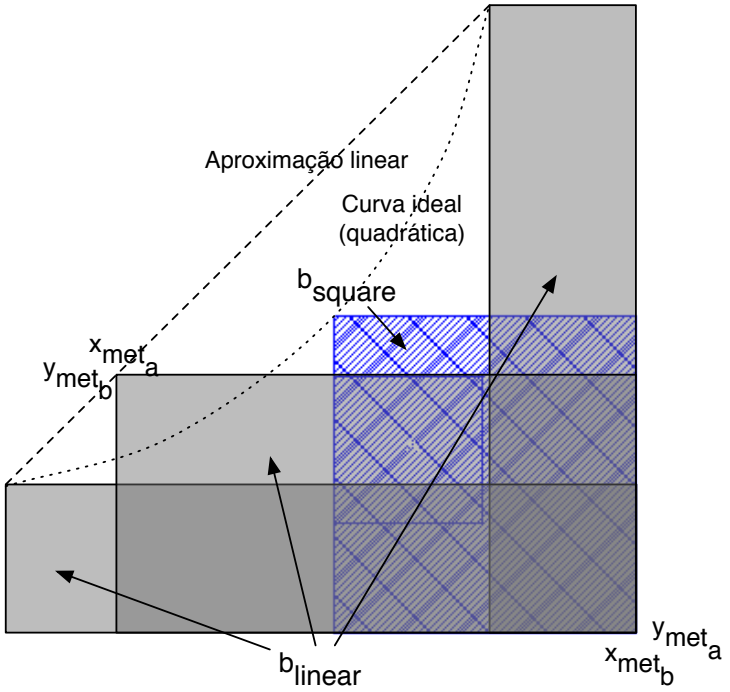


Figura 4.32: Área mínima dos elementos.



deixando o resolvedor decidir, através de variáveis booleanas, qual a formulação mais adequada:

$$\begin{array}{rcl} x_{met_b} - x_{met_a} & = & w_{met} \\ y_{met_b} - y_{met_a} & = & h_{met} \\ w_{met} + h_{met} & \geq & b_{linear}(\frac{A1M1}{W2VI} + W2VI) \\ w_{met} & \geq & b_{square}\sqrt{A1M1} \\ h_{met} & \geq & b_{square}\sqrt{A1M1} \\ b_{square} + b_{linear} & = & 1 \end{array} \tag{4.12}$$

onde b_{square} e b_{linear} são variáveis binárias mutualmente exclusivas que permitem selecionar entre os dois estilos de área mínima: quadrado ou linear, respectivamente.

4.8.2.9 Contatos Redundantes

O objetivo de inserir contatos extras é reduzir a resistência das conexões e melhorar a manufaturabilidade do leiaute da célula. Este problema foi solucionado instanciando uma quantidade fixa de contatos extras e deixar o resolvedor de MILP decidir quais deles devem ser "materializados" no leiaute final. Os demais contatos ficam com todos os valores igual a zero para suas geometrias. A formulação abaixo mostra como é feita a adição de cada contato extra:

$$\begin{array}{rcl} y_{newCnt_a} - y_{lastCnt_b} & = & b_{newCnt}S2CTCT \\ x_{newCnt_b} - x_{newCnt_a} & = & b_{newCnt}W2CT \\ y_{newCnt_b} - y_{newCnt_a} & = & b_{newCnt}W2CT \\ Maximize & : & \delta b_{newCnt} \end{array} \tag{4.13}$$

onde b_{newCnt} indica se o referido contato deve existir ou não no leiaute final da célula. Uma função de maximização foi utilizada para que o resolvedor procure colocar o maior número possível de contatos. Um peso de ajuste δ, é utilizado para compensar o aumento dos outros custos, como os de interconexão.

Algumas tecnologias sugerem dois contatos como a quantidade ideal, enquanto outras pedem que sejam colocados o máximo que cabe na difusão. A Figura 4.33 apresenta o significado de cada variável e mostra o resultado da aplicação deste recurso no leiaute de uma célula.

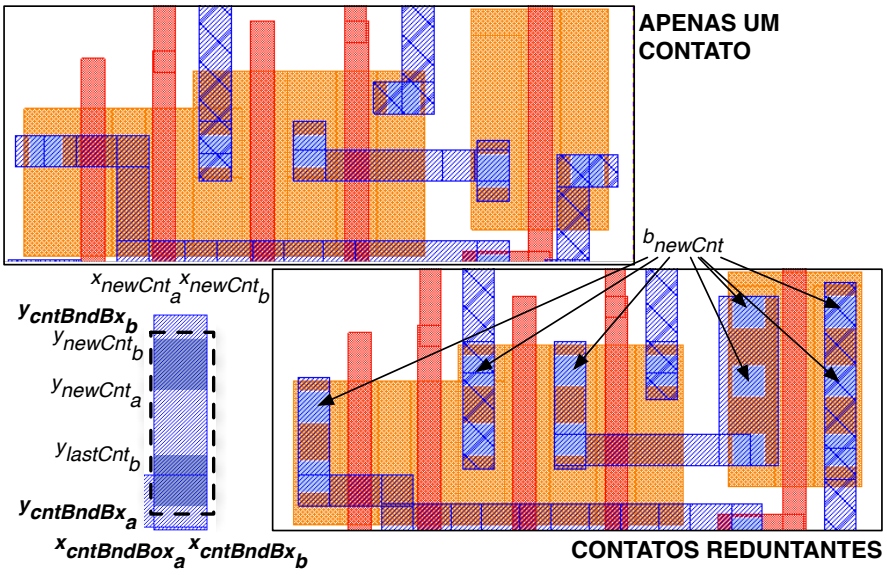
4.8.2.10 Função Objetivo

Um projetista, ao desenhar um circuito, possui uma lista de objetivos para otimização. Alguns objetivos costumam ter maior prioridade de otimização do que outros, como é o caso da largura da célula com relação ao aumento do espaçamento para DFM.

A maneira encontrada para priorizar a otimização de determinados objetivos em detrimento de outros (de maneira similar à forma que um projetista faria), foi através da atribuição de pesos diferenciados às variáveis na função de minimização. Isto é feito fazendo uma soma de produtos de cada variável pelo seu peso. Desta forma, para dar maior prioridade a minimização de conexões em polisilício sobre as de metal, ao mesmo tempo em que prioriza acima de tudo a diminuição da largura da célula, bastaria fazer como no exemplo:

$$min: \quad 5000WIDTH + 5POLYWL + MET1WL; \tag{4.14}$$

Figura 4.33: Contatos redundantes para difusão.



onde $WIDTH$ é a largura da célula, $POLYWL$ é o comprimento das conexões em *poly*, $MET1WL$ é o comprimento das conexões em metal 1 e as constantes são os pesos.

No ASTRAN, uma série de medidas foram utilizadas para serem minimizadas na função objetivo. São elas:

- *Largura* : Em uma biblioteca de células, normalmente todas as células possuem a mesma altura. Desta forma, a minimização da largura é a principal minimização que pode ser aplicada e possui o maior peso entre todas as demais regras.

- *Aproximação dos Contatos de Fonte e Dreno dos Transistores* : A difusão possui, normalmente, uma resistência mais elevada que a do metal. Por esta razão, uma otimização importante é diminuir a distância entre os contatos de fonte e dreno. Isto é feito calculando uma *bounding box* para ambos contatos e também o *gate* dos transistores. Por fim, a largura e altura dos lados desta *bounding box* são inseridas na função de minimização.

- *Contatos Redundantes* : O somatório do número de variáveis binárias que definem se cada contato extra deve ser materializado ou não é contabilizado na função de minimização. O peso destas variáveis é negativo, uma vez que o objetivo é maximizar a quantidade destes contatos.

- *Roteamento em Polisilício e Metal* : O metal possui uma resistência consideravelmente inferior à resistência do polisilício. Por esta razão, as conexões feitas em polisilício possuem um peso (prioridade de minimização) maior que as de metal.

- *Conexões em L* : Uma conexão de metal ou *poly* com muitas dobras ("escadinha") pode ser ruim para manufaturabilidade do leiaute. Isto foi tratado criando uma variável booleana que possui o valor de 1 sempre que houver um joelho na conexão e 0, caso contrário. Por fim, o somatório de todas estas variáveis é inserido na função de minimização.

- *Tamanho dos Elementos* : Tanto a altura quanto a largura dos elementos precisam ser inseridas na função de minimização sob risco de terminem com tamanho maior do que o mínimo necessário.

4.8.3 Polarização do substrato

A polarização do substrato é necessária em tecnologias não SOI para evitar que cargas livres interfiram no funcionamento do transistor, efeito este conhecido como *latch up*. Isto é feito com a inserção de contatos para difusão (TAPs) nos poços e substratos. Há uma regra de projeto que define a distância máxima que os transistores devem estar dos TAPs, variando entre uma tecnologia e outra.

Em bilbiotecas de *standard cells*, usualmente cada célula possui seus próprios TAPs ou então células especiais são colocadas em intervalos regulares para realizar esta tarefa.

A abordagem utilizada neste trabalho foi de implementar diferentes modelos (*templates*) de biblioteca de células, e permitir que o projetista possa escolher qual deseja utilizar. Os modelos implementados no ASTRAN foram: sem TAPs, com TAPS contínuas (entre uma célula e outra) e com TAPs descontínuas, o que foi suficiente para gerar células compatíveis com bibliotecas de *standard cells* para tecnologias comerciais de 350, 130 e $65nm$.

5 RESULTADOS

Este capítulo apresenta as comparações feitas com a finalidade de analisar o desempenho da ferramenta ASTRAN.

5.1 Geração de Circuitos Analógicos em 65nm

Circuitos analógicos impõe alguns desafios extras ao projetista devido a existência de componentes como resistores e capacitores que não são utilizados em circuitos digitais. O ASTRAN não suporta geração de resistores, no entanto, capacitores podem ser modelados como transistores onde o dreno/fonte é um terminal e o *gate* é outro.

As células utilizadas neste teste fazem parte de um sensor de envelhecimento programável de circuitos digitais. O sensor em si é construindo utilizando capacitores que, ao serem descarregados lentamente através de uma resistência, consegue detectar a ocorrência de transições de sinais próximas ao fim do período de relógio. Com isto, ele consegue prever se os *chips* estão próximos de falhar por envelhecimento pois, conforme envelhecem, estes se tornam mais lentos.

Este sensor foi desenvolvido primeiramente em tecnologia de $350nm$ (VAZQUEZ et al., 2009). Devido a existência de um resistor neste projeto, o sensor acabou não sendo gerado completamente no ASTRAN, parte do circuito precisou ser gerada manualmente.

No entanto, para geração em $65nm$ (VAZQUEZ et al., 2010, 2012), o sensor foi largamente otimizado ficando com apenas transistores e capacitores. Isto permitiu a geração de todo o circuito utilizando apenas o ASTRAN (utilizando regras de projeto conservadoras, uma vez que não suportava completamente esta tecnologia), que na época compactava o leiaute apenas em 1D. O leiaute obtido apresentou largura de $9{,}12um$, como mostra a Figura 5.1.

Em seguida, este circuito foi otimizado a mão para redução de área e correção de erros de DRC. O leiaute final apresentou largura de $8{,}4um$, representando uma diferença de 8,6% em relação ao circuito gerado automaticamente.

Após a implementação do compactador de leiaute em 2D ter sido finalizada, este circuito foi gerado novamente. O novo leiaute apresentou largura de $8{,}2um$, ou seja, uma redução de 10,1% com relação a implementação anterior do ASTRAN e de 2,4% com relação ao mesmo leiaute otimizado a mão. A razão para este resultado foi que a nova implementação permitiu que os transistores P e N ficassem mais próximos, eliminando a necessidade de aplicar a técnica de *folding* em 3 transistores.

O mesmo processo foi repetido para o *flipflop* que se conecta à saída do sensor. A largura do leiaute obtida pelo gerador antigo, pela otimização manual e pelo novo gerador com compactação em 2D, foi de $5{,}6um$. Isto ocorreu porque o leiaute produzido pelo gerador antigo já tinha muito pouco desperdício de espaço.

Figura 5.1: Leiaute do sensor de envelhecimento em tecnologia de $65nm$ (VAZQUEZ et al., 2012).

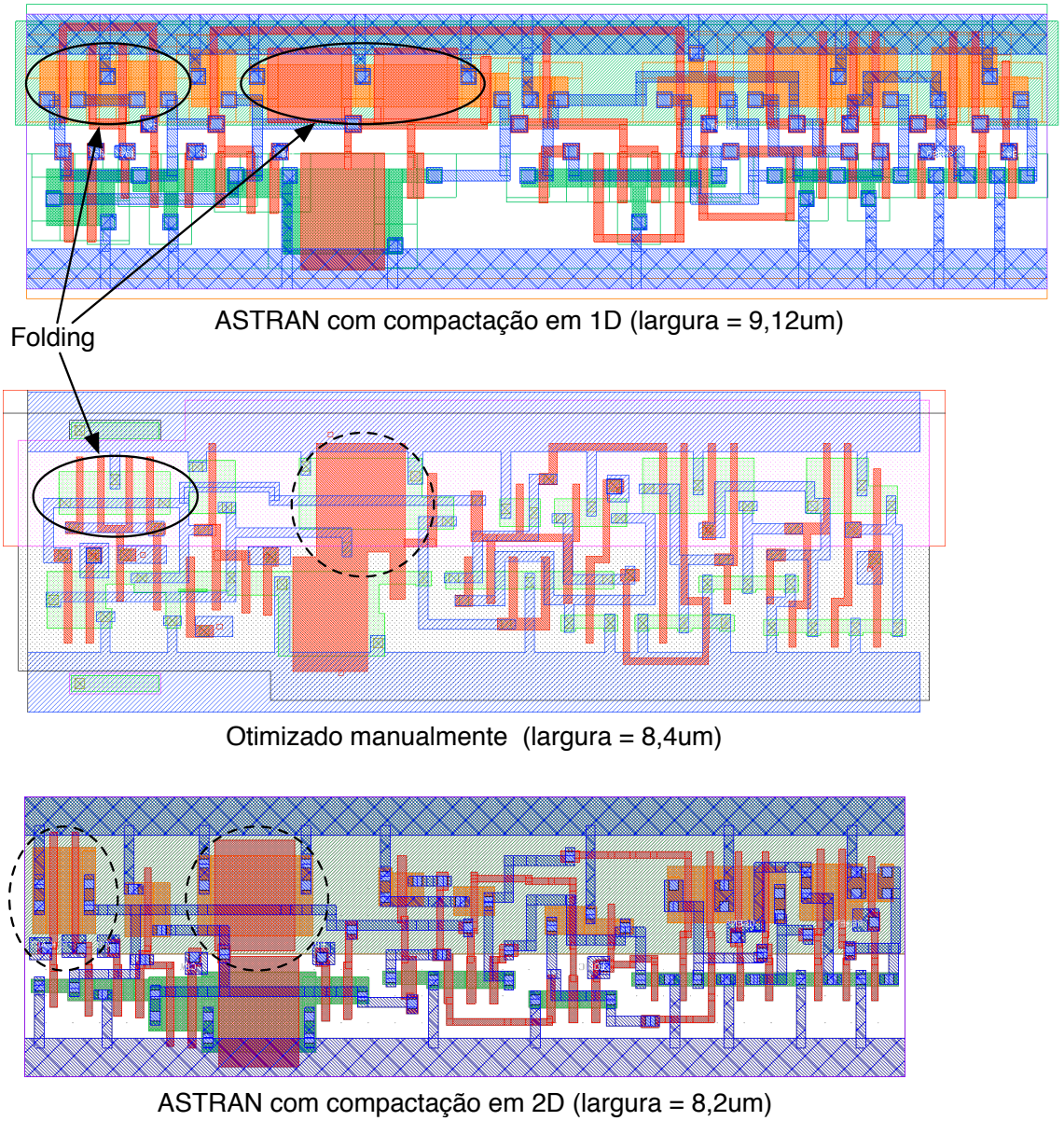
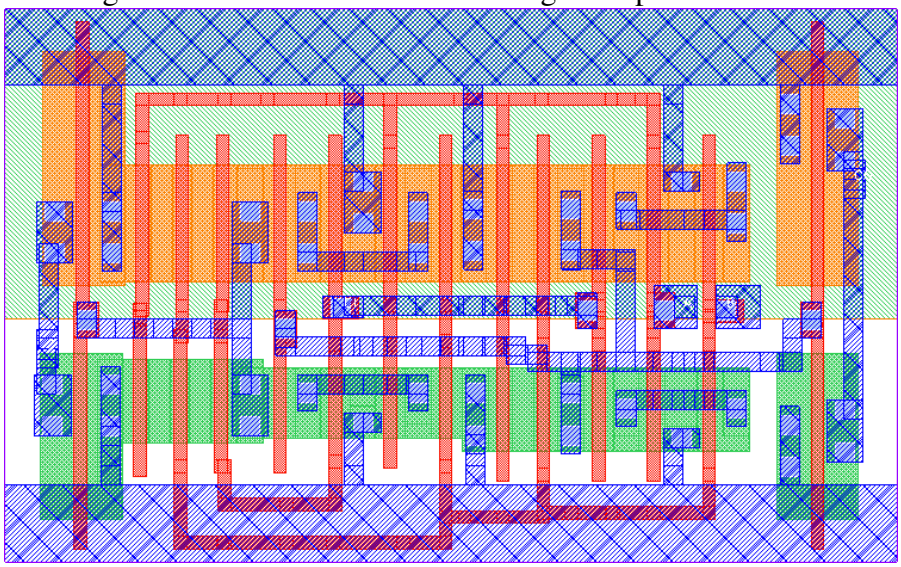


Figura 5.2: Leiaute de um FAD1X9 gerado pelo ASTRAN.



5.2 Comparação com *Standard Cells* em Tecnologia de 65nm

Um conjunto de *standard cells* disponíveis em uma biblioteca do processo CMOS ST $65nm$ foi utilizado como estudo de caso para avaliar a densidade dos leiautes produzidos pelo ASTRAN. Estas células são usualmente desenhadas a mão por projetistas experientes, o que permite uma comparação justa da ferramenta quanto ao leiaute das redes de transistores.

Este estudo utilizou um conjunto de diferentes células lógicas, com diferentes fatores de potência, permitindo comparar tanto células com transistores pequenos, quanto células com transistores que precisaram ser divididos pela técnica de *folding*.

O netlist SPICE de cada uma das células foi obtido através dos esquemáticos que acompanham as *standard cells*. O ASTRAN foi configurado para produzir as células com o mesmo modelo (altura da célula, posição do poço, etc.) da biblioteca da ST, sendo portanto 100% compatíveis entre si. O leiaute de cada uma das células foi gerado e importado para o Cadence Virtuoso. Cada célula foi validada com DRC e LVS, e comparada com a sua correspondente *standard cell*. Os erros de DRC encontrados (devido a regras ainda não modeladas, tais como o espeçamento mínimo entre elementos de uma mesma rede) foram todos corrigidos localmente, com pequena intervenção manual.

Os resultados obtidos para a largura das células são mostrados na Tabela 5.1. Adicionalmente, o tempo de execução para gerar cada célula com o ASTRAN também foi mensurado. Para tanto foi utilizado uma estação Intel Xeon E5520 2.27GHz W3540 de 16 cores. O aumento médio da largura das células sintetizadas automaticamente foi de apenas 3,6%, sendo que 71% delas apresentou a mesma largura das *standard cells*. Este resultado representa um grande avanço com relação a versão anterior do ASTRAN (com suporte a compactação do leiaute apenas em 1D) o qual apresentava acréscimo de área de 16% em média (POSSER et al., 2010). O leiaute da célula AND2X4 corresponde ao pior caso, com um aumento de 20% na largura da célula. Um pequeno aumento da distância entre dois transistores adjacentes - causada pelo modelo simplificado de transistores mostrado na Figura 4.30 - fez com que a célula ficasse com o comprimento de apenas um *pitch* horizontal maior (de 0,2um). Por se tratar de uma célula pequena, isto causou um grande aumento relativo na largura desta. Como os leiautes das células não estão disponíveis,

Tabela 5.1: Comparação entre leiautes de células gerados usando ASTRAN e equivalentes, em uma biblioteca de *standard cells* de $65nm$.

Célula	**# Trans.**	**Largura (μm)**			**Tempo Exec. (s)**
		Std. Cell	ASTRAN	%	ASTRAN
AND2X4	6	1	1,2	20	10
FAD1X4	28	3,6	4	12	750
FAD1X9	28	4,2	4,2	0	1800
HAD1X9	14	2,4	2,4	0	30
HAD1X18	18	2,8	2,8	0	205
INVX0	2	0,6	0,6	0	1
MUX21X9	12	1,8	2	11	80
NAND2X11	8	1,4	1,4	0	5
NAND2X21	12	2	2	0	8
NAND3X3	6	1,2	1,2	0	6
OAI211X3	8	1,4	1,4	0	7
OAI211X11	16	2,4	2,6	8	26
XOR2X3	10	1,6	1,6	0	16
XOR2X6	10	1,6	1,6	0	15
TOTAL	-	28	29	3,6	-

especulamos que o mesmo tenha ocorrido com as células MUX21X9 e OAI211X11. A célula FAD1X4 apresentou o segundo pior resultado. Por ser uma célula que apresenta transistores próximos ao tamanho mínimo, acreditamos que esta *standard cell* tenha sido projetada com transistores empilhados nas difusões P e N (o que não é suportado pelo ASTRAN). A célula FAD1X9 (Figura 5.2) não apresentou qualquer diferença na largura, provavelmente por possuir transistores maiores.

O tempo de execução se manteve curto mesmo para células de tamanho maior, permitindo ao projetista reduzir significativamente o tempo para produzir o leiaute de cada célula de horas/dias para apenas alguns minutos. A Figura 5.3 apresenta a variação do tempo de execução em função do número de transistores existentes nas células geradas. Foi notado um crescimento predominantemente exponencial na complexidade dos algoritmos utilizados.

5.3 Geração de Células para Circuitos Assíncronos

Encontra-se em desenvolvimento um fluxo totalmente automatizado para projeto de assíncronos chamado ASCEnD2 (MOREIRA et al., 2014) (segunda versão do fluxo ASCEnD (MOREIRA et al., 2011, 2013)), mostrado na Figura 5.4. Este novo fluxo utiliza a ferramenta ASTRAN desenvolvida neste trabalho para automatizar a etapa de geração da biblioteca de células para assíncronos. Os resultados obtidos são apresentados a seguir.

5.3.1 Comparação de Células

Com a finalidade de poder comparar os leiautes produzidos pelo ASTRAN com leiautes de células assíncronas, um subconjunto de células disponíveis na biblioteca ASCEnD serviu como estudo de caso. Esta biblioteca contém portas especiais utilizadas no projeto de circuitos assíncronos tais como C-elements e NCL gates, os quais complementam

Figura 5.3: Análise do tempo de geração das células no ASTRAN em função do número de transistores.

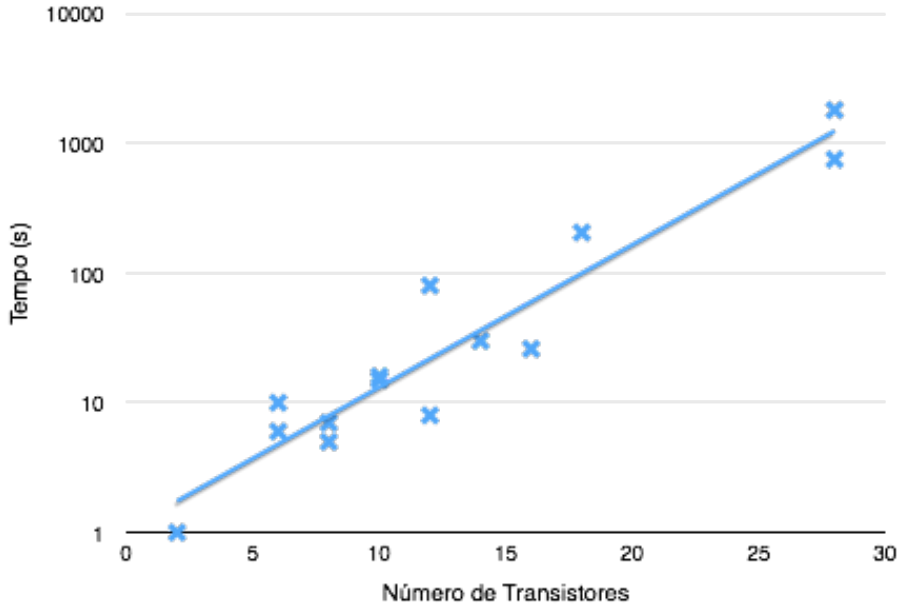
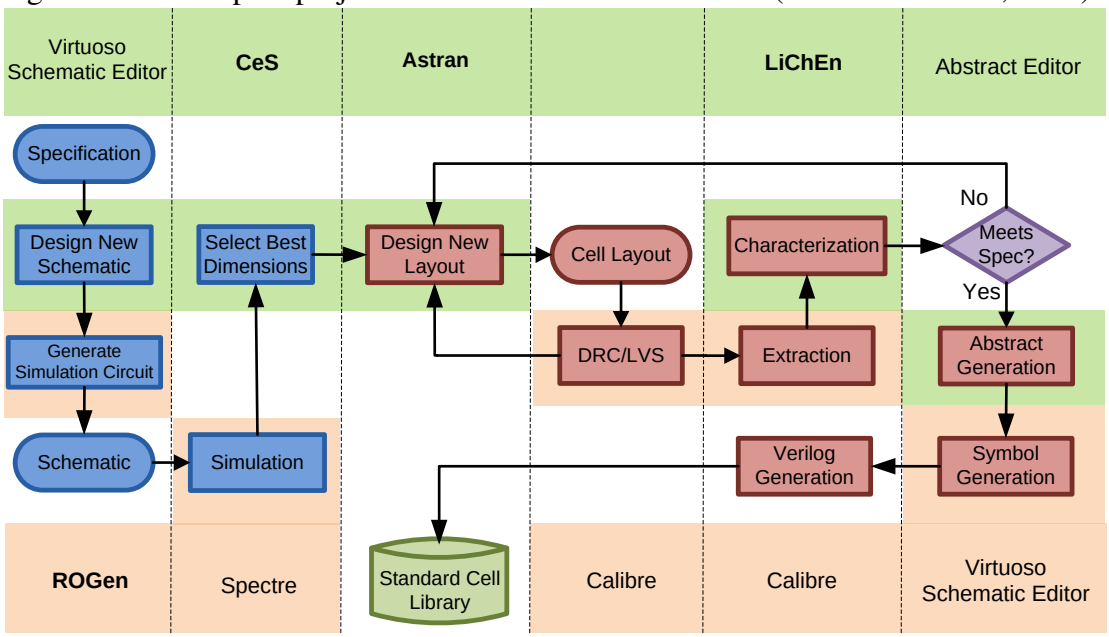


Figura 5.4: Fluxo para projeto de assíncronos do ASCEnD2 (MOREIRA et al., 2014).



as bibliotecas de portas lógicas tradicionais. ASCEnD foi desenvolvida para o processo STMicroelectronics CMOS de $65nm$. Ela contém mais de 500 portas feitas à mão. Isto permite uma comparação com leiautes feitos por projetista de média experiência.

Foi selecionado um conjunto destas portas como estudo de caso. Estas foram geradas com ASTRAN (que foi incorporado ao fluxo do ASCEnD2) para o mesmo modelo de biblioteca (altura de célula, posição do poço, etc.) e processo de fabricação. Não foi feita nenhuma tentativa de ajustar parâmetros de geração para melhorar o resultado, tendo sido considerado nos testes o primeiro leiaute obtido. O conjunto de 53 células lógicas escolhido inclui células com diferentes topologias e funcionalidades de C-Elements (SHAMS; EBERGEN; ELMASRY, 1996) e portas NCL (FANT; BRANDT, 1996):

- Resettable Sutherland C-Elements (RSUC2) de 2 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- Resettable Weak Feedback C-Elements (RWFC2) de 2 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- Resettable van Berkel C-Elements (RVBC2) de 2 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- Sutherland C-Elements (SUC2) de 2 entradas em 5 tamanhos (X2, X4, X7, X9, X13).
- Weak Feedback C-Elements (WFC2) de 2 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- van Berkel C-Elements (VBC2) de 3 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- Sutherland C-Elements (SUC2) de 3 entradas em 6 tamanhos (X2, X4, X7, X9, X13, X18).
- Weak Feedback C-Elements (WFC2) de 3 entradas em 5 tamanhos (X2, X4, X7, X9, X13).
- Portas NCL: 1-de-2, 1-de-3, 1-de-4, 2-de-3, 2-de-4, 3-de-4 e 3-de-5 (NCL12, NCL13, NCL14, NCL23, NCL24, NCL34 and NCL35) com tamanho X4.

As diferentes topologias de C-Elements permitem que diferentes compromissos sejam explorados e são típicos para modelos assíncronos. Cada topologia possui uma organização diferente da rede de transistores apresentando variados níveis de complexidade de roteamento, como mostram os esquemáticos da Figura 5.5. Além disto, foram utilizados vários tamanhos de transistores diferentes para a mesma célula. Para fins de comparação, a Figura 5.3.1 (a), por exemplo, mostra o leiaute de uma NCL24 gerada pelo ASTRAN, enquanto (b) mostra a mesma célula feita a mão, presente na biblioteca ASCEnD.

Devido à grande quantidade de dados gerados, os resultados obtidos são apresentados na Figura 5.8 de forma comparativa, em porcentagem. A área das portas neste estudo foi obtida diretamente a partir do leiaute. Os gráficos da primeira linha mostram que o ASTRAN obteve uma redução no tamanho para a grande maioria das células. A redução média foi de 19,2%, sendo 45% no melhor caso. Somente as duas versões da célula WFC3 apresentaram aumento de área, que foi de apenas 8%. A célula NCL35X4 (mostrada na Figura 5.7) corresponde à célula com maior quantidade de transistores e foi gerada

Figura 5.5: Exemplo de células assíncronas da bilbioteca ASCEnD: (a) SUC2, (b) VBC2, (c) NCL23 and (d) NCL35.

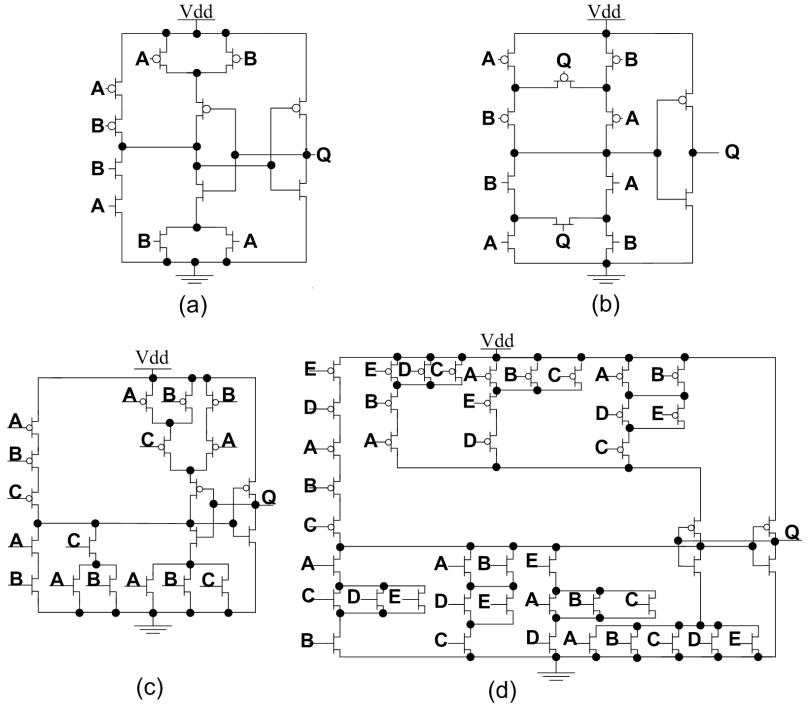
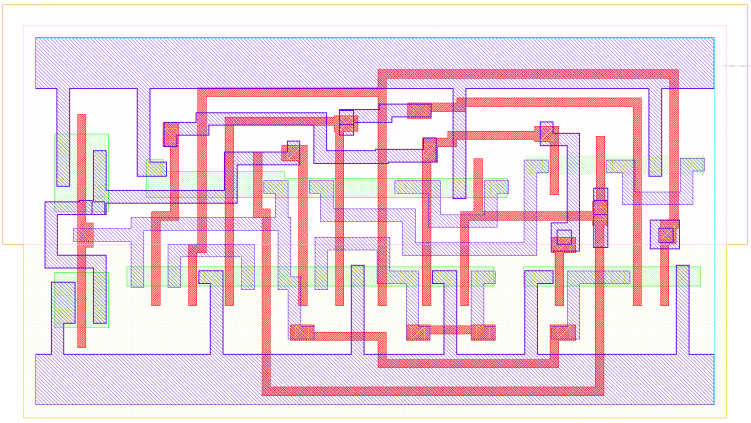
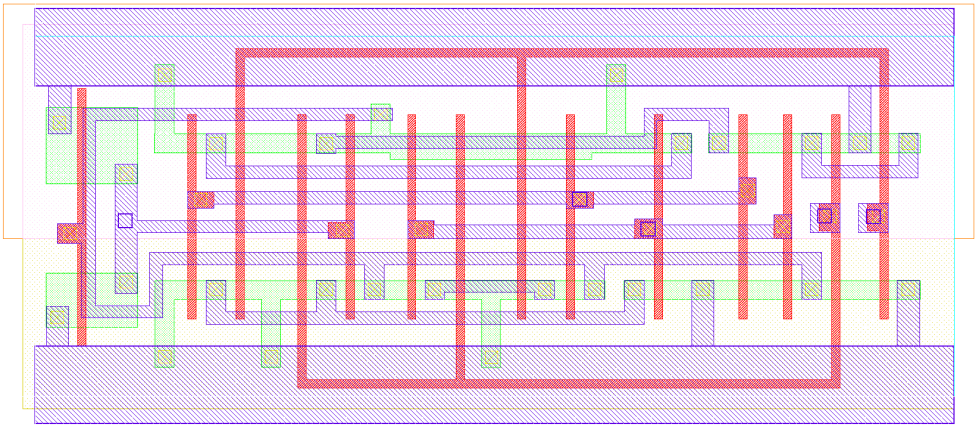


Figura 5.6: Leiaute da porta NCL24 com tamanho X4 presente na biblioteca ASCEnD.

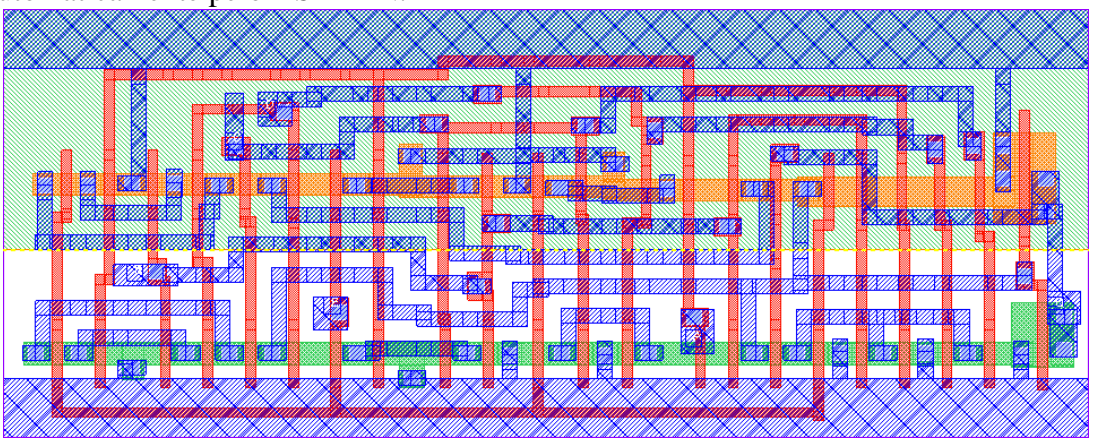


(a) ASTRAN



(b) Feita a mão

Figura 5.7: Leiaute da porta NCL35 com tamanho X4, contendo 44 transistores, gerada automaticamente pelo ASTRAN.



em 12h de tempo de execução (ZIESEMER et al., 2014b). As capacitâncias parasitas foram extraídas utilizando o Calibre PEX assumindo pior caso para a extração de RC. A segunda linha da figura mostra os resultados observados para a capacitância parasita interna total. O fluxo permitiu uma redução em várias das portas, sendo 60% no melhor caso mas também apresentou aumento de até 65% em outras. Na média o resultado foi bastante balanceado, com melhora de apenas 1%. A capacitância de entrada para o pior caso também foi avaliada. Os resultados apresentados na terceira linha, mostram que as células geradas pelo ASTRAN possuem menor capacitância para a grande maioria das células. Neste teste houve uma redução de 17% em média e de 59% no melhor caso. Os resultados piores ficaram limitados a um pequeno conjunto de células e foi de no máximo 21%.

O *netlist* extraído de cada porta foi simulado usando o Cadence Spectre, com todos os arcos de transição e estados estáticos sendo exercitados. Durante a simulação, a energia por transição e atraso de propagação para cada arco além da corrente de fuga para estados estáticos foram medidos. A simulação assumiu um processo típico com as portas operando com tensão e temperatura padrões (1 V e 25 C). Através dos valores medidos, foram calculados a média de energia por transição, atraso médio e média da corrente de fuga para cada porta. Como mostram a quarta e quinta coluna da figura, o consumo de energia e o atraso apresentaram pouca diferença, com um pequeno ganho para as células geradas pelo ASTRAN. As portas NCL, que tipicamente possuem maior complexidade que as demais, apresentaram os melhores resultados, com até 46% de redução. De uma forma geral, as células que apresentaram os piores resultados neste teste foram as mesmas que apresentaram resultado ruim na capacitância parasita. Uma hipótese para este problema é que neste teste todas as células foram geradas com apenas 2 trilhas de roteamento *intracell*, fazendo com que mais conexões fossem realizadas em *poly* abaixo das linhas de alimentação, criando capacitâncias extras. Finalmente, as células geradas automaticamente apresentaram redução na corrente de fuga para a grande maioria das células. Mesmo que modesta, de apenas 3,6% em média e 9% no melhor caso, esta redução pode ser significativa, principalmente para dispositivos móveis (onde o consumo estático costuma ser responsável por boa parte do consumo total de energia).

Por fim, é estimado que o projeto manual das 53 portas tenha consumido em torno de 27 dias para serem projetadas. As mesmas células foram geradas com o ASTRAN e corrigidas pontualmente, para aprovação no DRC, em apenas 2 dias. Isto representa ganho

Figura 5.8: Comparação de área, parasitas, capacitância de entrada, energia por transição, atraso e corrente de fuga entre células geradas pelo ASTRAN e equivalentes feitas à mão na biblioteca ASCEnd.

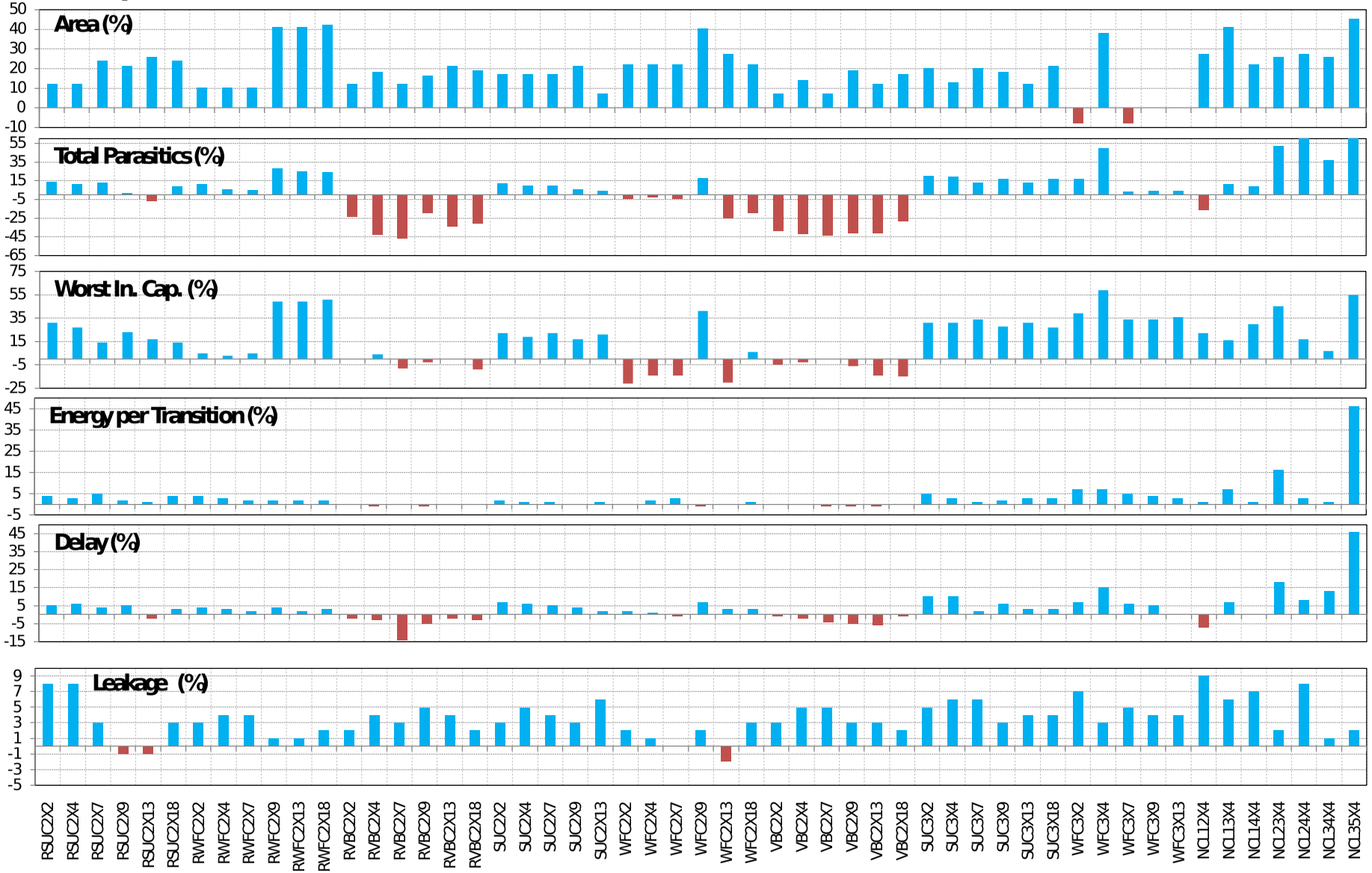


Tabela 5.2: Resultados da simulação para circuitos utilizando células da biblioteca ASCEnD e geradas com o ASTRAN.

		Área (mm^2)	**Atraso** (ns)	**Consumo Dinâmico** (mW)	**Consumo Estático** (μW)
KS0	**ASCEnD**	0,001815	1,929	1,008	0,144
	ASTRAN	0,001311	1,869	1,005	0,140
	Melhora	*28%*	*3%*	*0%*	*3%*
KS1	**ASCEnD**	0,002177	2,031	0,769	0,125
	ASTRAN	0,001673	1,910	0,730	0,118
	Melhora	*23%*	*6%*	*5%*	*6%*
KS2	**ASCEnD**	0,002419	1,649	0,877	0,121
	ASTRAN	0,001914	1,616	0,861	0,114
	Melhora	*21%*	*2%*	*2%*	*6%*
SK0	**ASCEnD**	0,010365	3,414	2,568	0,642
	ASTRAN	0,007437	3,326	2,501	0,624
	Melhora	*28%*	*3%*	*3%*	*3%*
SK1	**ASCEnD**	0,012456	3,419	1,936	0,554
	ASTRAN	0,009527	3,105	1,889	0,525
	Melhora	*24%*	*9%*	*2%*	*5%*
SK2	**ASCEnD**	0,013849	2,662	2,326	0,537
	ASTRAN	0,010921	2,623	2,265	0,508
	Melhora	*21%*	*1%*	*3%*	*5%*

de produtividade de uma ordem de grandeza com relação ao projeto puramente manual. Também é importante ressaltar que o ASTRAN foi configurado com seus parâmetros padrões, o que abre margem para que resultados ainda melhores possam ser obtidos com ajuste específico para cada célula tais como: número de trilhas internas, alinhamento dos transistores, geração mais conservadora, ajuste nos pesos de roteamento/posicionamento e número de iterações do posicionador.

5.3.2 Comparação entre Circuitos

Usando as portas do estudo de caso apresentado na seção anterior, um novo estudo foi feito desta vez com o projeto de 6 diferentes circuitos: 3 versões de um somador de 8 bits Kogge-Stone KS0, KS1 e KS2, utilizando as portas Martin, Sutherland e van Berkel C-Elements, respectivamente; e 3 versões de um somador de 32 bits Sklansky SK0, SK1 e SK2, também utilizando Martin, Sutherland e van Berkel C-Elements, respectivamente. Todos os estudos de casos empregam codificação dual-rail e utilizam estilo de lógica *delay-insensitive minterm-synthesis*, descrito em (BEEREL; OZDAG; FERRETTI, 2010). Somadores Kogge-Stone possuem um total de 306 portas, enquanto somadores Sklansky possuem 1738. Todos circuitos foram manualmente projetados usando netlists SPICE extraídos após o leiaute e duas versões foram feitas: uma utilizando as células feitas a mão da biblioteca ASCEnD e outra com células sintetizadas automaticamente utilizando o ASTRAN. A Tabela 5.2 mostra a área total dos circuitos. Nesta comparação, ganhos entre 21% e 28% foram obtidos pela utilização do ASTRAN.

Para analisar o consumo dinâmico/estático e também o desempenho dos circuitos, foi criado um ambiente de *mixed-signal* que permitiu simular o netlist SPICE usando um

testbench do SystemC. Os simuladores analógico e digital empregados foram o Cadence Spectre e Incisive, respectivamente. O *testbench* mediu o atraso de propagação dos circuitos inserindo dados aleatórios durante 1 ms. As medidas de consumo de energia foram realizadas utilizando o comando ".measure" da linguagem SPICE. A Tabela 5.2 apresenta o resultado obtido para o consumo de energia dinâmico e estático. Todos os resultados assumem o uso de extração RC no pior caso operando em condições típicas (processo típico, 1 V e 25 C). Os resultados mostraram que o ASTRAN é capaz de gerar células que proveem similar compromissos de desempenho e energia quando comparado à células projetadas a mão. De fato, houve um pequeno ganho nestes quesitos em todos os circuitos testados.

5.4 Outros Resultados

Uma biblioteca de *standard cells*, em formato SPICE, contendo 126 células para o FreePDK45 (FREEPDK45 DESIGN KIT, 2014) de $45nm$, foi utilizada como teste para o ASTRAN. Esta biblioteca contém células usualmente encontradas na maior parte das bibliotecas comerciais, incluindo: FFs, multiplexadores, somadores e AOIs. A maior parte destas, com cerca de 3 tamanhos diferentes.

O ASTRAN foi capaz de gerar todas elas em menos de 2 dias de trabalho, demonstrando a capacidade da ferramenta em lidar com diferentes tipos e tamanhos de células.

Por outro lado, se formos analisar a dificuldade para a compactação do leiaute, este processo (que é acadêmico) apresenta quantidade muito menor de regras de projeto que o kit de $65nm$ utilizado nas demais comparações, sendo portanto de menor relevância para analisar as capacidades do compactador desenvolvido.

6 CONCLUSÃO

Neste trabalho foi apresentado um fluxo para síntese do leiaute de redes de transistores. Uma ferramenta chamada ASTRAN foi desenvolvida para permitir o rápido projeto de células CMOS, com o menor desperdício de área possível, a partir de uma descrição em formato SPICE e sem restrições quanto a arquitetura da rede de transistores. Foram feitas uma série de melhorias com relação a versão anterior, apresentada em (ZIESEMER; LAZZARI; REIS, 2007), com o intuito de permitir a geração de leiautes de células para tecnologias de fabricação comerciais de até $65nm$. Tecnologias abaixo de $130nm$ possuem um conjunto grande de regras de projeto, incluindo regras condicionais, que exigiram o desenvolvimento de uma nova técnica para compactação dos leiautes. A técnica desenvolvia utilizando MILP, além de conseguir lidar com as principais regras de projeto condicionais, consegue também compactar o leiaute em duas dimensões simultaneamente, o que pelo nosso conhecimento é o primeiro trabalho publicado a atingir estes objetivos de forma eficiente, utilizando um método exato.

Os resultados da comparação com uma biblioteca comercial de *standard cells* (estado da arte) em $65nm$ mostraram que é possível gerar portas lógicas de forma automática com apenas 3,6% de acréscimo de área, em média. Este seria o pior caso de comparação uma vez que o ganho do ASTRAN estaria justamente em gerar células que ainda não existem em bibliotecas, possibilitando uma melhor otimização lógica e elétrica do circuito. Em outro teste, das 126 células existentes em uma biblioteca de $45nm$, todas foram geradas com sucesso.

Na comparação com células feitas a mão da biblioteca ASCEnd para circuitos assíncronos, foi possível obter uma redução média na área de 19%, enquanto os circuitos testados com estas células apresentaram redução de 24%. Também foi obtido pequenos ganhos para atraso, consumo estático e dinâmico em todos os circuitos testados. O aumento de produtividade estimado (em comparação ao projeto manual das células) foi de uma ordem de grandeza, com uma média de cerca de 26 células por dia (considerando também importação e validação com LVS e DRC). Células contendo até 44 transistores foram todas geradas em 12h ou menos de tempo de execução, demonstrando a capacidade do ASTRAN em lidar com células grandes e de terminar a geração em um tempo aceitável.

Estes resultados demonstram que o ASTRAN pode ser de grande ajuda tanto para o rápido desenvolvimento de novas células lógicas (em adição às existentes nas bibliotecas de *standard cells*), quanto para a geração de circuitos com células *on-demand* (com mapeamento sem uso de uma biblioteca), permitindo que mais métodos de otimização sejam utilizados.

6.1 Trabalhos Futuros

Com a conclusão deste trabalho, identificou-se alguns pontos que podem ser melhorados ou que precisam ser melhor estudados:

- Permitir intervenção manual do projetista no posicionamento e roteamento das células, antes de compactar o leiaute destas;
- Modificar o algoritmos de *folding* dos transistores para que ele não duplique individualmente transistores em série, diminuindo o número de conexões entre eles, e com isto, melhorar a qualidade do leiaute;
- Modificar o algoritmo de roteamento *intra-cell* para fazer arborescência em redes críticas;
- Aprimorar o algoritmo de compactação para utilizar a regra de espaçamento mínimo 2D para elementos adjacentes na vertical e horizontal e também em elementos distantes;
- Eliminar os erros de DRC remanescentes: espaçamento mínimo de elementos de uma mesma rede e camada (em curto-circuito entre si);
- Implementar heurísticas para geração especulativa: testar mais de um posicionamento/roteamento por célula; compactar diversos leiautes de uma mesma célula e selecionar o melhor (de menor custo na função de minimização);
- Suportar células com duas bandas de altura, necessárias para geração de portas com estrutura complexa e grande número de transistores;
- Fazer um novo fluxo de geração de células compatível com processos estado-da-arte ($22nm$ ou menos), onde os transistores precisam ser agrupados em ilhas de difusão com o mesmo tamanho, exigindo um novo posicionador de transistores (como o proposto por (CORTADELLA, 2013)).

REFERÊNCIAS

BARDSLEY, A.; TARAZONA, L.; EDWARDS, D. Teak: a token-flow implementation for the balsa language. In: INTERNATIONAL CONFERENCE ON APPLICATION OF CONCURRENCY TO SYSTEM DESIGN, ACSD, 9., 2009, Augsburg, Germany. **Proceedings...** Washington: IEEE Computer Society, 2009. p.23–31.

BEEREL, P. A.; DIMOU, G. D.; LINES, A. M. Proteus: an asic flow for ghz asynchronous designs. **IEEE Design & Test of Computers**, [S.l.], v.28, n.5, p.36–51, 2011.

BEEREL, P. A.; OZDAG, R. O.; FERRETTI, M. **A Designer's Guide to Asynchronous VLSI**. [S.l.]: Cambridge University Press, 2010.

BOYER, D. G. Symbolic layout compaction review. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 25., 1988, Atlantic City, New Jersey, United States. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1988. p.383–389.

BRUSAMARELLO, L.; HENTSCHCKE, R.; MORELLI, C.; REIS, R. Um Sistema DistribuÌdo Para Posicionamento de Células em Circuitos VLSI. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 5., 2004, Pelotas, RS. **Proceedings...** [S.l.: s.n.], 2004.

CADENCE Design Systems. Disponível em: <http://www.cadence.com/>. Acesso em: 19 jan. 2012.

CALDWELL, A.; KAHNG, A. B.; MARKOV, I. **Placement Formats**. Disponível em: <http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/plFormats.html>. Acesso em: 19 jan. 2012.

CARRO, L. **Gerador Parametrizável de Partes Operativas CMOS**. 1989. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

CHANG, Y. W. **Compactação de Leiaute**. Notas de Aula. Disponível em: <http://cc.ee.ntu.edu.tw/ ywchang/Courses/EDA/lec4.pdf>. Acesso em: 21 mai. 2014.

CHEN, W.-K. **Computer Aided Design and Design Automation**. 3rd.ed. Boca Raton, FL, USA: CRC Press, Inc., 2009.

CONG, J. et al. Provably good performance-driven global routing. **IEEE Trans. on CAD**, [S.l.], v.11, p.739–752, 1992.

CORTADELLA, J. Area-Optimal Transistor Folding for 1-D Gridded Cell Design. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, [S.l.], v.32, n.11, p.1708–1721, 2013.

CORTADELLA, J. et al. Desynchronization: synthesis of asynchronous circuits from synchronous specifications. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.25, n.10, p.1904–1921, 2006.

DETJENS, E. et al. Technology mapping in MIS. In: ACM/IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1987, Los Alamitos, CA, USA. **Proceedings...** New York: ACM Press, 1987. p.116–119.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische Mathematik**, [S.l.], v.1, p.269–271, 1959.

DUECK, G.; SCHEUER, T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. **J. Comput. Phys.**, San Diego, CA, USA, v.90, n.1, p.161–175, 1990.

DUFFIN, R.; PETERSON, E.; ZENER, C. **Geometric programming**: theory and application. [S.l.]: Wiley, 1967.

ELMORE, W. C. The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers. **Journal of Applied Physics**, [S.l.], v.19, n.1, p.55–63, Jan 1948.

FANG, F.; ZHU, J. Calligrapher: a new layout migration engine based on geometric closeness. In: INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN, ISQED, 5., 2004. **Proceedings...** Los Alamitos: IEEE Computer Society, 2004. p.25–30.

FANT, K.; BRANDT, S. NULL Convention Logic^{TM}: a complete and consistent logic for asynchronous digital circuit synthesis. In: INTERNATIONAL CONFERENCE ON APPLICATION SPECIFIC SYSTEMS, ARCHITECTURES AND PROCESSORS, 1996. **Proceedings...** Los Alamitos: IEEE Computer Society, 1996. p.261–273.

FLACH, G.; HENTSCHKE, R.; REIS, R. Algorithms for improvement of RotDL router. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 2004, Ijuí: Unijuí. **Proceedings...** [S.l.: s.n.], 2004.

FLACH, G.; JOHANN, M.; REIS, R. Quadratic placement with single-iteration linear system solver. In: INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 24., 2011, João Pessoa, Brazil. **Proceedings...** New York: ACM Press, 2011. p.109–112.

FREEPDK45 design kit. Disponível em: <http://www.eda.ncsu.edu/wiki/FreePDK>. Acesso em: 19 jan. 2014.

FU, D.-S. et al. Topology-driven cell layout migration with collinear constraints. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 2009. **Proceedings...** Los Alamitos: IEEE Computer Society, 2009. p.439–444.

GAREY, M.; JOHNSON, D. The Rectilinear Steiner Tree Problem is NP-Complete. **SIAM Journal on Applied Mathematics**, [S.l.], v.32, n.4, p.826–834, 1977.

GEREZ, S. H. **Algorithms for VLSI Design Automation**. 1st.ed. New York, NY, USA: John Wiley & Sons, Inc., 1999.

GUO, P.-N.; CHENG, C.-K.; YOSHIMURA, T. An O-tree representation of non-slicing floorplan and its applications. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 36., 1999, New Orleans, Louisiana, United States. **Proceedings...** New York: ACM Press, 1999. p.268–273.

GUPTA, A.; HAYES, J. P. Width minimization of two-dimensional CMOS cells using integer programming. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1996, San Jose, California, United States. **Proceedings...** Washington: IEEE Computer Society, 1996. p.660–667.

GUPTA, A.; HAYES, J. P. CLIP: an optimizing layout generator for two-dimensional cmos cells. In: ANNUAL CONFERENCE ON DESIGN AUTOMATION, DAC, 34., 1997, Anaheim, California, United States. **Proceedings...** New York: ACM Press, 1997. p.452–455.

GUPTA, A.; HAYES, J. P. Optimal 2-D cell layout with integrated transistor folding. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1998, San Jose, California, United States. **Proceedings...** New York: ACM Press, 1998. p.128–135.

GUPTA, A.; HAYES, J. P. CLIP: integer-programming-based optimal layout synthesis of 2d cmos cells. **ACM Transactions on Design Automation of Electronic Systems, TODAES**, [S.l.], v.5, n.3, p.510–547, 2000.

GUPTA, A.; THE, S.-C.; HAYES, J. P. XPRESS: a cell layout generator with integrated transistor folding. In: EUROPEAN CONFERENCE ON DESIGN AND TEST, EDTC, 1996. **Proceedings...** Washington: IEEE Computer Society, 1996. p.393.

GUPTA, P. et al. Selective gate-length biasing for cost-effective runtime leakage control. In: ANNUAL CONFERENCE ON DESIGN AUTOMATION, DAC, 41., 2004, San Diego, CA, USA. **Proceedings...** New York: ACM Press, 2004. p.327–330.

GUROBI Optimizer. Disponível em: <http://www.gurobi.com/>. Acesso em: 19 jan. 2014.

GURUSWAMY, M. et al. CELLERITY: a fully automatic layout synthesis system for standard cell libraries. In: DESIGN AUTOMATION CONFERENCE, DAC, 34., 1997, Anaheim, California, United States. **Proceedings...** New York: ACM Press, 1997. p.327–332.

HANAN, M. On Steiner's problem with rectilinear distance. **SIAM J. Appl. Math**, [S.l.], n.14, p.255–265, 1966.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems, Science, and Cybernetics**, [S.l.], v.SSC-4, n.2, p.100–107, 1968.

HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

HENTSCHKE, R. **Algorithms for Wire Length Improvement of VLSI Circuits With Concern to Critical Paths**. 2007. Tese (Doutorado em Ciência da Computação) — PPGC, UFRGS, Porto Alegre.

HENTSCHKE, R. F.; FLACH, G.; PINTO, F.; REIS, R. 3D-Vias Aware Quadratic Placement for 3D VLSI Circuits. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, ISVLSI, 2007. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.67–72.

HSIEH, Y.-C. et al. LiB: a cell layout generator. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 27., 1990, Orlando, Florida, United States. **Proceedings...** New York: ACM Press, 1990. p.474–479.

HUANG-YU, C.; YAO-WEN, C. **Global and detailed routing**. 1st.ed. Taipei: Springer, 2009. 687-749p.

IIZUKA, T. **Optimal Layout Synthesis of Standard Cells in Large Scale Integration**. 2007. Tese (Doutorado) — Department of Electronic Engineering, Graduate School of Engineering, The University of Tokyo, Tokyo, Japan.

IIZUKA, T.; IKEDA, M.; ASADA, K. High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability. In: CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2004, Yokohama, Japan. **Proceedings...** Piscataway: IEEE Press, 2004. p.149–154.

IIZUKA, T.; IKEDA, M.; ASADA, K. Exact Minimum-Width Transistor Placement for Dual and Non-dual CMOS Cells. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, [S.l.], v.E88-A, n.12, p.3485–3491, 2005.

IIZUKA, T.; IKEDA, M.; ASADA, K. Exact minimum-width transistor placement without dual constraint for CMOS cells. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, GLSVSLI, 15., 2005, Chicago, Illinois, USA. **Proceedings...** New York: ACM Press, 2005. p.74–77.

INTEL. Quick Reference Guide: <http://www.intel.com/pressroom/kits/quickreffam.htm> . Acesso em: 19 jan. 2014.

ITOOLS. Disponível em: <http://www.internetcad.com/>. Acesso em: 22 mar. 2007.

KAHNG, A. B.; ROBINS, G. A new class of Steiner tree heuristics with good performance: the iterated 1-steiner approach. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1990, Santa Clara, CA, USA. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1990. p.428–431.

KARMAZIN, R.; OTERO, C.; MANOHAR, R. cellTK: automated layout for asynchronous circuits with nonstandard cells. In: IEEE INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS, ASYNC, 19., 2013. **Proceedings...** Los Alamitos: IEEE Computer Society, 2013. p.58–66.

KEUTZER, K.; NEWTON, A. R.; ORSHANSKY, M. **Layout Compaction**. Disponível em: <http://www.eecs.berkeley.edu/ keutzer/classes/244fa2004/pdf/5-1-compaction.pdf>. Acesso em: 19 jan. 2014.

KIM, N. S. et al. Leakage Current: moore's law meets static power. **Computer**, Los Alamitos, CA, USA, v.36, n.12, p.68–75, Dec. 2003.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, [S.l.], v.220, n.4598, p.671–680, May 1983.

LAZZARI, C. **Automatic Layout Generation of Static CMOS Circuits Targeting Delay and Power Reduction**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

LAZZARI, C. **Transistor level automatic generation of radiation-hardened circuits**. 2007. Tese (Doutorado em Microeletrônica) — PGMICRO, UFRGS, Porto Alegre.

LAZZARI, C.; ZIESEMER, A.; REIS, R. An automated design methodology for layout generation targeting power leakage minimization. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS, AND SYSTEMS, ICECS, 16., 2009. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2009. p.81–84.

LEE, C. Y. An Algorithm for Path Connections and Its Applications. **IRE Transactions on Electronic Computers**, New York, v.EC-10, p.346–365, Sept. 1961.

LEFEBVRE, M.; MARPLE, D.; SECHEN, C. The future of custom cell generation in physical synthesis. In: DESIGN AUTOMATION, DAC, 34., 1997, Anaheim, California, United States. **Proceedings...** New York: ACM Press, 1997. p.446–451.

LEONHARDT, C. C.; ZIESEMER, A.; REIS, R. Improved Detailed Routing Using Pathfinder and A*. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 24., 2009. **Proceedings...** Porto Alegre: SBC, 2009.

LEONHARDT, C. C.; ZIESEMER, A.; REIS, R. **Roteamento de Circuitos VLSI**. 2010. Trabalho de Conclusão de Curso — Instituto de Informática, UFRGS, Porto Alegre.

LIKHACHEV, M. **A* and Weighted A* Search**. Disponível em: <http://www.cs.cmu.edu/ motionplanning/lecture/Asearch_v8.pdf>. Acesso em: 19 jan. 2012.

LUBASZEWSKI, M. **Geração Automática de Lógica Aleatória Utilizando a Metodologia TRANCA**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MARPLE, D. **Two dimensional compaction system and method**. EP Patent 1,405,228.

MARPLE, D.; SMULDERS, M.; HEGEN, H. An efficient compactor for 45° layout. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 25., 1988, Atlantic City, New Jersey, United States. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1988. p.396–402.

MARTIN, A.; NYSTROM, M. Asynchronous Techniques for System-on-Chip Design. **Proceedings of the IEEE**, [S.l.], v.94, n.6, p.1089–1120, June 2006.

MAZIASZ, R. L.; GURUSWAMY, M.; RAMAN, S. **US Patent 6209123**: methods of placing transistors in a circuit layout and semiconductor device with automatically placed transistors. [S.l.]: IEEE, 2001.

MAZIASZ, R. L.; HAYES, J. P. Exact width and height minimization of CMOS cells. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, DAC, 28., 1991, San Francisco, California, United States. **Proceedings...** New York: ACM Press, 1991. p.487–493.

MCMURCHIE, L.; EBELING, C. PathFinder: a negotiation-based performance-driven router for fpgas. In: ACM INTERNATIONAL SYMPOSIUM ON FIELD-PROGRAMMABLE GATE ARRAYS, FPGA, 3., 1995, Monterey, California, United States. **Proceedings...** New York: ACM Press, 1995. p.111–117.

MEINHARDT, C. **Geração de Leiautes Regulares Baseados em Matrizes de Células**. 2006. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MOFFITT, M. D.; ROY, J. A.; MARKOV, I. L. The Coming of Age of (Academic) Global Routing. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2008. **Proceedings...** New York: ACM, 2008. p.148–155.

MORAES, F. G. **TRAGO - Síntese Automática de Leiaute para Circuitos em Lógica Aleatória**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

MOREIRA, M. T.; ARENDT, M. E.; ZIESEMER, A.; REIS, R.; CALAZANS, N. L. V. Automated Synthesis of Cell Libraries for Semi-Custom Asynchronous Design. In: IEEE 20TH INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS - FRESH IDEAS (DEMO), ASYNC, 2014, Potsdam, Germany. **Proceedings...** [S.l.: s.n.], 2014.

MOREIRA, M. T.; C. H. M. OLIVEIRA, N. L. V. C. ASCEnD: a standard cell library for semi-custom asynchronous design. In: SIMPÓSIO SUL DE MICROELETRÔNICA, SIM, 28., 2013, Porto Alegre. **Proceedings...** Porto Alegre: SBC, 2013.

MOREIRA, M. T.; OLIVEIRA, B.; PONTES, J.; CALAZANS, N. A 65nm standard cell set and flow dedicated to automated asynchronous circuits design. In: IEEE INTERNATIONAL SOC CONFERENCE, SOCC, 2011. **Proceedings...** USA: IEEE, 2011. p.99–103.

MOREIRA, M. T.; OLIVEIRA, C.; PORTO, R.; CALAZANS, N. Design of NCL gates with the ASCEnD flow. In: IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS, LASCAS, 4., 2013. **Proceedings...** USA: IEEE, 2013. p.1–4.

NANGATE Library Creator. Disponível em: <http://www.nangate.com/>. Acesso em: 19 jan. 2014.

NOWICK, S.; SINGH, M. High-performance asynchronous pipelines: an overview. **Design & Test of Computers, IEEE**, [S.l.], 2011.

ONG, C.-L.; LI, J.-T.; LO, C.-Y. GENAC: an automatic cell synthesis tool. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 26., 1989, Las Vegas, Nevada, United States. **Proceedings...** New York: ACM Press, 1989. p.239–244.

POIRIER, C. J. Excellerator: custom cmos leaf cell layout generator. **IEEE Transactions in Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.8, n.7, p.744–755, July 1989.

POSSANI, V.; MARQUES, F.; ROSA JUNIOR, L. da; CALLEGARO, V.; REIS, A.; RIBAS, R. Transistor-level optimization of CMOS complex gates. In: IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS, LASCAS, 4., 2013. **Proceedings...** USA: IEEE, 2013. p.1–4.

POSSER, G.; FLACH, G.; WILKE, G.; REIS, R. Gate Sizing Minimizing Delay and Area. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, ISVLSI, 2011, Chennai, India. **Proceedings...** Los Alamitos: IEEE Computer Society, 2011. p.315–316.

POSSER, G.; REIS, R. **Dimensionamento de Portas Lógicas Usando Programação Geométrica**. 2011. Dissertação (Mestrado em Ciência da Computação) — UFRGS, Porto Alegre.

POSSER, G.; ZIESEMER, A.; GUIMARES, D.; WILKE, G.; REIS, R. A study on layout quality of automatic generated cells. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, ICECS, 17., 2010. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2010. p.651–654.

PROGENESYS. Disponível em: <http://www.arm.com/about/newsroom/arm-acquires-prolific-inc.php>. Acesso em: 12 fev. 2014.

REESE, R.; SMITH, S.; THORNTON, M. Uncle - An RTL Approach to Asynchronous Design. In: IEEE INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS, ASYNC, 18., 2012, Lyngby, Denmark. **Proceedings...** Los Alamitos: IEEE Computer Society, 2012. p.65–72.

REIS, A.; ROBERT, M.; REIS, R. Topological parameters for library free technology mapping. In: BRAZILIAN SYMPOSIUM ON INTEGRATED CIRCUIT DESIGN, SBCCI, 11., 1998, Armação de Buzios, RJ, Brazil. **Proceedings...** Los Alamitos: CA: IEEE Computer Society, 1998. p.213–216.

REIS, R. Design automation of transistor networks, a new challenge. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2011, Rio de Janeiro, Brasil. **Proceedings...** USA: IEEE, 2011. p.2485–2488.

REKHI, S.; TROTTER, J. D.; LINDER, D. H. Automatic layout synthesis of leaf cells. In: ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, DAC, 32., 1995, San Francisco, California, United States. **Proceedings...** New York: ACM, 1995. p.267–272.

RIEPE, M. A.; SAKALLAH, K. A. Transistor placement for noncomplementary digital VLSI cell synthesis. **ACM Transaction on Design Automation of Electronic Systems**, New York, NY, USA, v.8, n.1, p.81–107, 2003.

ROY, J. A.; MARKOV, I. L. High-performance routing at the nanometer scale. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2007, Piscataway, NJ, USA. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2007. p.496–502.

SAID, H.; ABBAS, H.; SHAHEIN, H. A Novel General Graph-Based Simplex Algorithm Applied to IC Layout Compaction and Migration. In: INTERNATIONAL DESIGN AND TEST WORKSHOP, IDT, 2., 2007. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.241–246.

SAID, H. et al. Analog Layout Retargeting. In: GRAEB, H. E. (Ed.). **Analog Layout Synthesis**. [S.l.]: Springer US, 2011. p.205–242.

SANTOS, C. L. dos et al. Incremental Timing Optimization for Automatic Layout Generation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005, Kobe, Japão. **Proceedings...** USA: IEEE, 2005. v.4, p.3567–3571.

SANTOS, G. B. V. dos; REIS, R. **Area Routing in Digital Integrated Circuits**. 2006. Dissertação (Mestrado em Ciência da Computação) — UFRGS, Porto Alegre.

SANTOS, G.; JOHANN, M.; REIS, R. Channel Based Routing in Channel-less Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2006. **Proceedings...** USA: IEEE, 2006. p.4 pp.–340.

SAWICKI, S.; HENTSCHKE, R.; FLACH, G.; JOHANN, M.; REIS, R. Studying the influence of I/O pads placement on wirelength and 3D-Vias of VLSI 3D integrated circuits. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 2007, Porto Alegre, RS, Brazil. **Proceedings...** Porto Alegre: SBC, 2007. p.89–92.

SCHLAG, M.; LIAO, Y.-Z.; WONG, C. An algorithm for optimal two-dimensional compaction of VLSI layouts. **Integration, the VLSI Journal**, [S.l.], v.1, n.2–3, p.179 – 209, 1983.

SERDAR, T.; SECHEN, C. Automatic datapath tile placement and routing. In: CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2001, Munich, Germany. **Proceedings...** Piscataway: IEEE Press, 2001. p.552–559.

SHAMS, M.; EBERGEN, J.; ELMASRY, M. A comparison of CMOS implementations of an asynchronous circuits primitive: the c-element. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, ISLPED, 1996, Monterey, CA. **Proceedings...** USA: IEEE, 1996. p.93–96.

SHIGEHIRO, Y. et al. Optimal layout recycling based on graph theoretic linear programmig approach. In: IFIP TC10/WG 10.5 INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, VLSI, 1994. **Proceedings...** Amsterdam: North-Holland Publishing, 1994. p.25–34.

SUSIN, A. A. **Etude des Parties Operatives a Elements Modulaires pour Processeurs Monolithiques**. 1981. Tese (Doutorado em Engenharia) — Université Grenoble, Grenoble, France.

SYNOPSYS. Disponível em: <http://www.synopsys.com/>. Acesso em: 19 jan. 2012.

TANG, Q.; ZHU, J. Two-dimensional layout migration by soft constraint satisfaction. In: INTERNATIONAL SYMPOSIUM ON QUALITY OF ELECTRONIC DESIGN, ISQED, 6., 2005. **Proceedings...** USA: IEEE, 2005. p.35–39.

THONNART, Y.; BEIGNE, E.; VIVET, P. A Pseudo-Synchronous Implementation Flow for WCHB QDI Asynchronous Circuits. In: INTERNATIONAL SYMPOSIUM ON ASYNCHRONOUS CIRCUITS AND SYSTEMS, ASYNC, 2012, Lyngby, Denmark. **Proceedings...** Los Alamitos: IEEE Computer Society, 2012. p.73–80.

UEHARA, T.; VANCLEEMPUT, W. M. Optimal layout of CMOS functional arrays. **IEEE Transactions on Computers**, New York, v.30, n.5, p.305–312, 1981.

VAHIA, D.; CIESIELSKI, M. Transistor level placement for full custom datapath cell design. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 1999, Monterey, California, United States. **Proceedings...** New York: ACM Press, 1999. p.158–163.

VATTIKONDA, R.; WANG, W.; CAO, Y. Modeling and minimization of PMOS NBTI effect for robust nanometer design. In: DESIGN AUTOMATION CONFERENCE, DAC, 43., 2006, New York, NY, USA. **Proceedings...** New York: ACM Press, 2006. p.1047–1052.

VAZQUEZ, J. C.; CHAMPAC, V.; ZIESEMER, A. M.; REIS, R.; TEIXEIRA, I. C.; SANTOS, M. B.; TEIXEIRA, J. P. Delay sensing for long-term variations and defects monitoring in safety–critical applications. **Analog Integrated Circuits and Signal Processing**, [S.l.], v.70, p.249–263, February 2012.

VAZQUEZ, J. C.; CHAMPAC, V.; ZIESEMER, A. M.; REIS, R.; TEIXEIRA, I.; SANTOS, M. B.; TEIXEIRA, J. P. Low-sensitivity to process variations aging sensor for automotive safety-critical applications. In: VLSI TEST SYMPOSIUM, VTS, 28., 2010, Santa Cruz, CA. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2010. p.238–243.

VAZQUEZ, J.; CHAMPAC, V.; ZIESEMER, A.; REIS, R.; TEIXEIRA, I.; SANTOS, M.; TEIXEIRA, J. Built-in aging monitoring for safety-critical applications. In: IEEE INTERNATIONAL ON-LINE TESTING SYMPOSIUM, IOLTS, 15., 2009, Sesimbra, Lisbon. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 2009. p.9 –14.

WILKE, G. et al. Finding the Critical Delay of Combinational Blocks by Floating Vector Simulation and Path Tracing. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 15., 2002, Porto Alegre. **Proceedings...** Washington: IEEE Computer Society, 2002. p.277–282.

ZHU, J.; FANG, F.; TANG, Q. Calligrapher: a new layout-migration engine for hard intellectual property libraries. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.24, n.9, p.1347–1361, Sep 2005.

ZIESEMER, A. et al. Cell Size Estimative in an Automatic Layout Generation Flow. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 21., 2006, Porto Alegre. **Proceedings...** Porto Alegre: Instituto de Informática da UFRGS, 2006. p.257–260.

ZIESEMER, A.; LAZZARI, C.; REIS, R. Transistor level automatic layout generator for non-complementary CMOS cells. In: IFIP INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, VLSI-SOC, 2007, Atlanta, GA, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.116 –121.

ZIESEMER, A.; REIS, R. **Geração Automática de Partes Operativas de Circuitos VLSI**. 2007. Dissertação (Mestrado em Ciência da Computação) — UFRGS, Porto Alegre.

ZIESEMER, A.; REIS, R. Simultaneous Two-Dimensional Cell Layout Compaction Using MILP with ASTRAN. In: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, ISVLSI, 2014, Tampa, FL, USA. **Proceedings...** [S.l.]: IEEE, 2014. (aceito para publicação).

ZIESEMER, A.; REIS, R.; MOREIRA, M. T.; ARENDT, M. E.; CALAZANS, N. L. V. A Design Flow for Physical Synthesis of Digital Cells with ASTRAN. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 24., 2014, Houston, TX. **Proceedings...** USA: ACM, 2014.

ZIESEMER, A.; REIS, R.; MOREIRA, M. T.; ARENDT, M. E.; CALAZANS, N. L. V. Automatic Layout Synthesis with ASTRAN Applied to Asynchronous Cells. In: IEEE LATIN AMERICAN SYMPOSIUM ON CIRCUITS AND SYSTEMS, LASCAS, 5., 2014, Santiago, Chile. **Proceedings...** USA: IEEE, 2014.

APÊNDICE A ASTRAN: SÍNTESE FÍSICA DE CIRCUITOS COM ASTRAN

Este capítulo apresenta o fluxo do ASTRAN para síntese física de circuitos VLSI, a partir de uma descrição SPICE ou Verilog. Este fluxo ainda se encontra em fase de desenvolvimento e cada uma das etapas atualmente implementadas é descrita em detalhes.

A.1 Introdução

Dentro do grupo de pesquisa no qual este trabalho se inclui, foram desenvolvidas várias ferramentas para tratar diferentes problemas do processo de síntese física. No entanto, um problema que havia é que estas ferramentas ainda não se comunicavam entre si, impedindo sua utilização dentro de um fluxo de síntese física. Dentre as ferramentas existentes estavam: o posicionador de células ZPlace (HENTSCHKE et al., 2007), o roteador RotDL (FLACH; HENTSCHKE; REIS, 2004) e também o gerador de leiautes de células lógicas CellGen (ZIESEMER; LAZZARI; REIS, 2007).

Este é ainda um trabalho em andamento e este capítulo apresenta os módulos que já foram implementados.

A.2 Fluxo

A ferramenta CellGen foi originalmente desenvolvida para a síntese exclusiva de leiautes de células lógicas. As células geradas eram inseridas em uma biblioteca para em seguida serem utilizadas por um compilador de parte operativa (ZIESEMER; REIS, 2007)

O fluxo deste compilador foi o ponto de partida para o desenvolvimento da ferramenta ASTRAN. No entanto, como o posicionamento e roteamento dos módulos de parte operativa (*bit-cells*) era feito de forma especializada (pré-definido), esta ferramenta precisou ser modificada para fazer síntese de lógica aleatória.

Um novo fluxo foi então desenvolvido para permitir síntese de circuitos digitais a partir de uma descrição da rede de transistores, ao mesmo tempo que permite que células sejam geradas por demanda, após o dimensionamento dos transistores. Este fluxo é mostrado na Figura A.1.

A.3 Descrição da rede de transistores

A descrição SPICE permite que ferramentas de síntese lógica recebam descrições de circuito em linguagens de alto nível como VHDL, Verilog, etc. e entreguem um circuito com transistores dimensionados para serem gerados pelo ASTRAN. Um problema que

Figura A.1: Fluxo da ferramenta de síntese física ASTRAN.

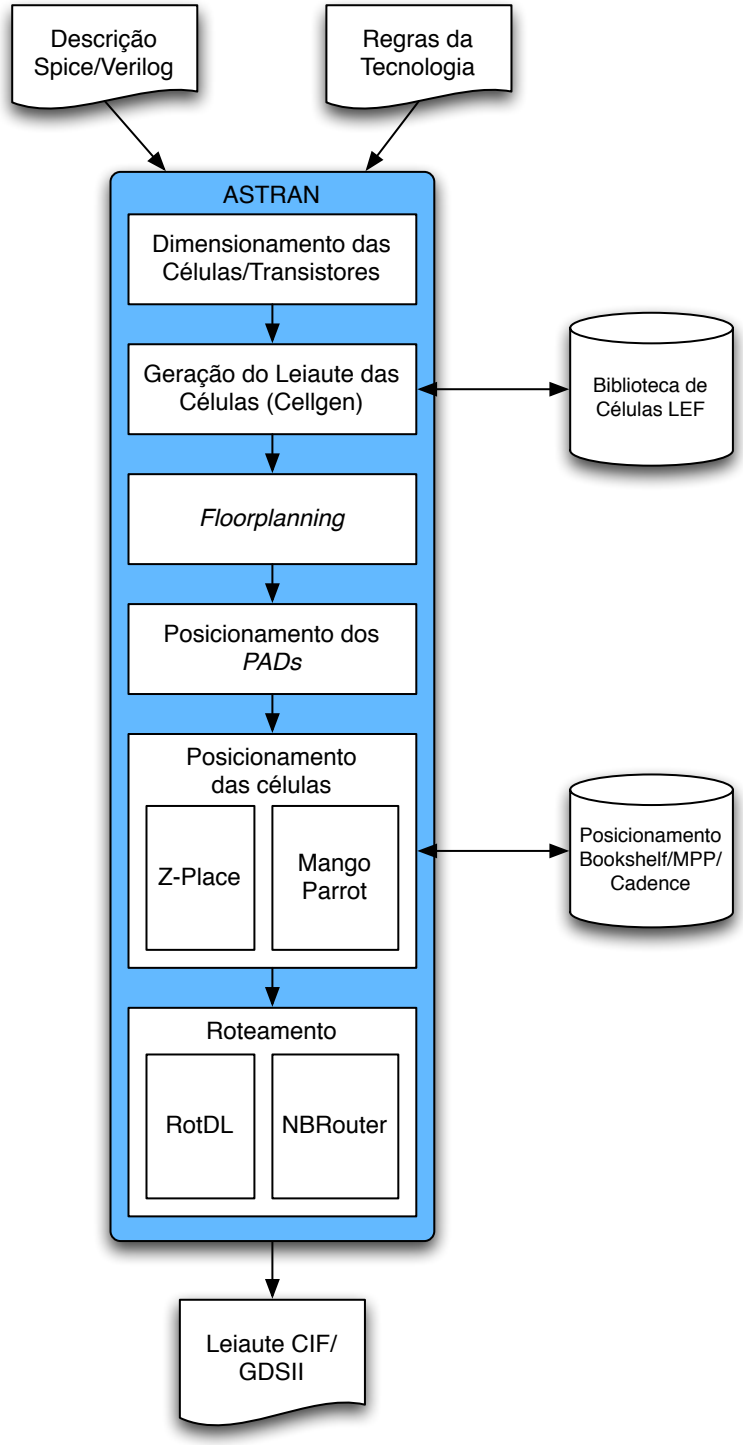
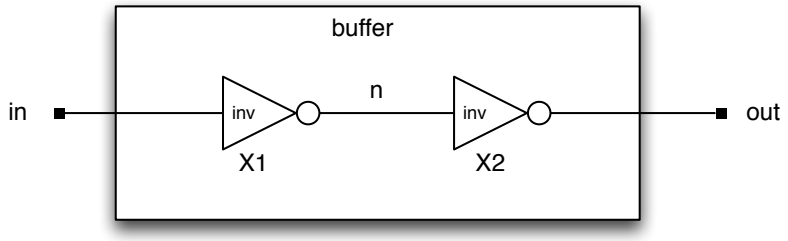


Figura A.2: Diagrama lógico de um buffer.



existe é que o formato SPICE não estabelece uma hierarquia rígida que faça distinção entre células e o restante do circuito.

Normalmente, ao extrair-se o leiaute de uma célula em ferramentas comerciais (CADENCE DESIGN SYSTEMS, 2011; SYNOPSYS, 2011), as células são descritas dentro de sub-circuitos (.SUBCKT), provendo tanto uma forma de indicar suas conexões de entrada e a saída, quanto uma forma de poder reutilizar ela múltiplas vezes através de instâncias. No ASTRAN foi adotado esta mesma padronização, considerando cada sub-circuito como uma célula que pode ter seu leiaute gerado pelo módulo CellGen.

O topo do projeto (TOP), é um sub-circuito especial com as instâncias das células e conexões entre elas. Caso o topo esteja fora de qualquer hierarquia na descrição SPICE, o ASTRAN cria automaticamente um sub-circuito com o nome do projeto. A importância de colocar a hierarquia topo dentro de um sub-circuito é permitir descrever os terminais de entrada e saída..

No entanto, a descrição SPICE não contempla informações sobre o posicionamento dos terminais no leiaute. Isto impede que o ASTRAN utilize esta informação para fazer um posicionamento das células mais eficiente com o posicionador zPlace (HENTSCHKE et al., 2007).

A forma encontrada para contornar esta deficiência foi adicionar esta informação como comentário (linha começando com *) no arquivo SPICE. Desta forma o arquivo mantém a compatibilidade com o formato original. Isto foi feito com o comando **interface*, seguido do nome da rede que o pino pertence, da orientação (E, W, N ou S) e da direção (I ou O).

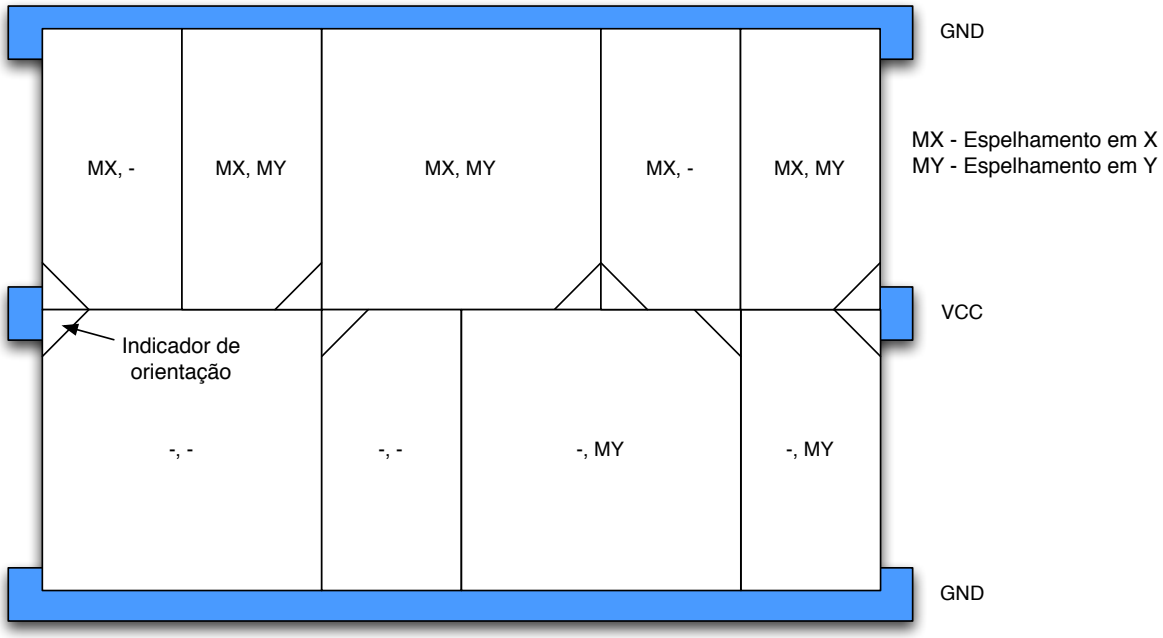
O código abaixo mostra um exemplo de arquivo SPICE suportado pelo ASTRAN para o circuito do *buffer* mostrado na Figura A.2:

```
*interface in W I
*interface out E O

.subckt inv in out vcc gnd
MN1 out in gnd gnd NMOS L=0.35U W=2.0U
MP2 out in vcc vcc PMOS L=0.35U W=2.0U
.ends inv

X1 in n vcc gnd inv
X2 n out vcc gnd inv
```

Figura A.3: Estrutura das células em bandas. O triângulo no canto indica o espelhamento das instâncias.



A.4 Regras de Projeto

As regras de projeto são utilizadas tanto para geração das células pelo CellGen, quanto para realizar o roteamento detalhado do circuito. Elas são fornecidas pela *foundry* e devem ser respeitadas pelo projetista para que não ocorra problemas durante a fabricação do circuito. As regras suportadas atualmente pelo ASTRAN são listadas no Apêndice B. Com este conjunto de regras foi possível suportar diversos *design kits* de até $45nm$ (FREEPDK45 DESIGN KIT, 2014).

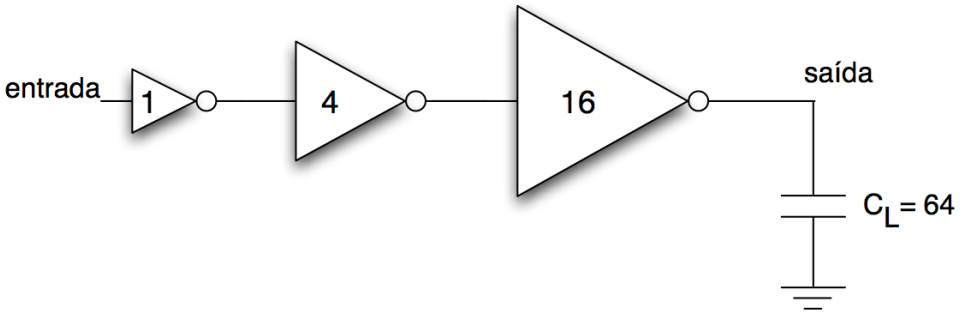
A.5 Configuração do Circuito

O posicionamento de circuitos baseados em células é usualmente estruturado em bandas, como mostrado na Figura A.3. Cada célula obedece uma estrutura regular que facilita a conexão com as demais células do circuito, sem que ocorra violação de regras de leiaute.

Dentro de uma mesma banda as células podem ser espelhadas sobre o eixo Y, enquanto o espelhamento em X ocorre usualmente em bandas alternadas para compartilhar a linha de alimentação com as bandas adjacentes.

No entanto, as regras para a criação de *standard cells* não são fornecidas pelas regras de projeto, mas sim definida pelo projetista da biblioteca, podendo mudar de um projeto para outro. O ASTRAN suporta atualmente 3 modelos de bibliotecas, onde a principal diferença entre eles é quanto à forma de inserção dos TAPs (contatos para polarização do poço e substrato): sem TAPS, com TAPS contínuos (entre as células) e descontínuos. As demais características seguem o padrão adotado pela maioria das bibliotecas de *standard cells*: com células de altura fixa, posicionadas em bandas, com alimentação nas extremidades superiores e inferiores, e pinos alinhados a uma grade de roteamento. Alguns parâmetros foram flexibilizados para permitir a utilização de bibliotecas de células com caraterísticas diferentes. A lista abaixo mostra os parâmetros suportados:

Figura A.4: Dimensionamento pela metodologia de *Fanout* de 4.



- Largura do *pitch* horizontal e vertical da grade de roteamento;
- Nome das redes de alimentação. Ex.: VCC e GND;
- Largura das redes de alimentação em metal 1;
- Altura da banda em passos da grade de roteamento (a altura precisa ser múltipla do *pitch* vertical da grade de roteamento);
- Posição do poço N dentro das células.

O nome das redes de alimentação costuma variar de uma tecnologia para outra e por esta razão precisou ser parametrizado. Estas redes diferem das demais pois sua conectividade no leiaute é feita por justaposição com as células adjacentes, enquanto a conexão das demais redes é feita por algoritmos específicos para roteamento *intra-cell* e detalhado. Sem esta informação a ferramenta não conseguiria funcionar adequadamente.

A.6 Dimensionamento de Transistores

O dimensionamento dos transistores é feito com o uso de um multiplicador, dado pelo atributo m nas instâncias das células, para representar a sua escala com relação à célula original. Este fator m é multiplicado por todos os W dos transistores para obtenção do novo dimensionamento da célula. A Figura A.4 mostra um exemplo de dimensionamento de buffers utilizando a metodologia de *Fanout* de 4.

A etapa de dimensionamento dos transistores foi desenvolvida por Posser (POSSER; REIS, 2011) dentro do *framework* do ASTRAN. Neste trabalho foi desenvolvido uma ferramenta de dimensionamento de portas lógicas para circuitos integrados, utilizando técnicas de otimização de problemas baseadas em Programação Geométrica (PG) (DUFFIN; PETERSON; ZENER, 1967).

Primeiramente as portas são modeladas usando o modelo de chaves RC e o atraso é calculado usando o modelo de Elmore - que produz funções posinomiais possibilitando a resolução do problema por programação geométrica. Para cada porta, é utilizado um fator de escala que multiplica a largura dos seus transistores. O valor deste fator de escala são as variáveis de otimização do problema.

A.7 Geração das Células Lógicas

Após o dimensionamento das células, é possível gerar o leiaute de cada uma das células do circuito. Isto é feito através do módulo CellGen (ZIESEMER; LAZZARI; REIS,

2007; ZIESEMER; REIS, 2007) do ASTRAN. Opcionalmente, as células podem ser carregadas já prontas de uma biblioteca salva no formato LEF (Library Exchange Format).

A geração do leiaute é feita uma célula de cada vez. Foi dado ao projetista a opção de configurar parâmetros como: número mínimo de trilhas internas, esforço e peso das métricas para posicionamento dos transistores, alinhamento dos transistores, otimização do roteamento *intra-cell* (com nós de Steiner) e custos do roteamento/compactador.

Ao final, o circuito é compactado e o leiaute é inserido na biblioteca de células da ferramenta ASTRAN. Esta biblioteca também pode ser salva em formato LEF para uso posterior.

Os leiautes gerados pelo ASTRAN também podem ser exportados para outras ferramentas, incluindo comerciais, em formato CIF e GDSII. Um script foi desenvolvido para automatizar esta tarefa na ferramenta ICFB da Cadence (CADENCE DESIGN SYSTEMS, 2011), configurando todos os parâmetros de importação sem precisar da intervenção do usuário.

A.8 Detecção da Posição dos Pinos das Células

O formato utilizado para o armazenamento da biblioteca de células lógicas no ASTRAN é o LEF. Este formato possui informações sobre a geometria das redes de entrada e saída das células e obstruções, mas não possui informações sobre as coordenadas onde pode ser inserido pinos para as camadas superiores de metal, pinos estes que devem estar alinhados à grade de roteamento do circuito.

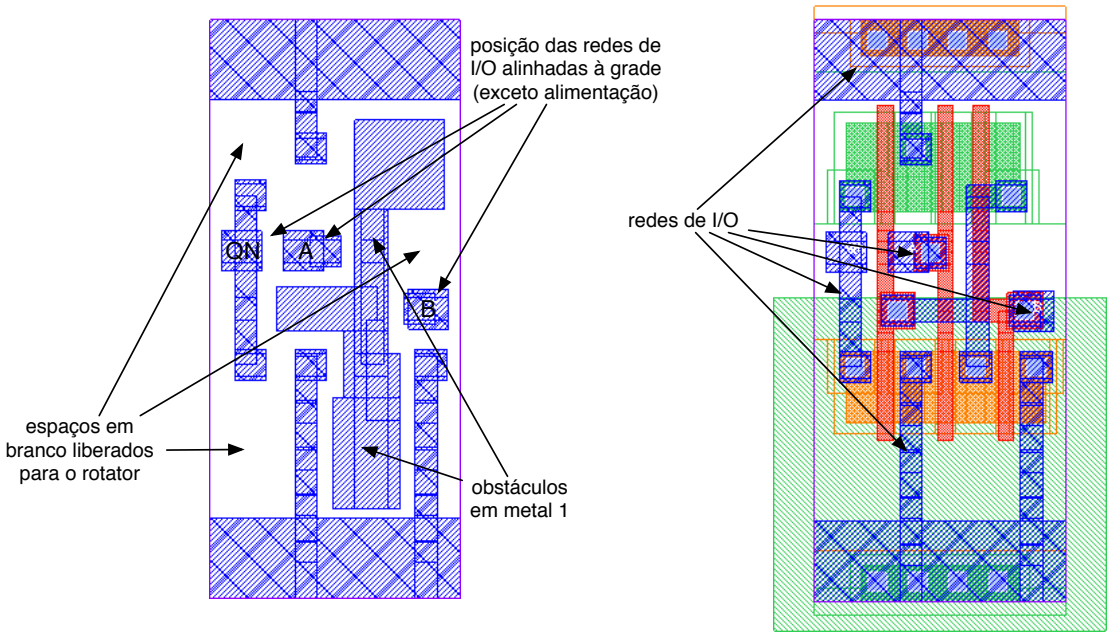
A abordagem utilizada no ASTRAN foi tentar identificar estas coordenadas (onde podem ser inserido pinos de E/S) em cada intersecção da grade de roteamento que passa sobre cada célula. Isto é feito verificando se é possível inserir vias nessas posições, conectando com alguma rede de E/S da camada de Metal 1, sem violar qualquer regra de desenho. Atualmente o ASTRAN somente é capaz de identificar a posição dos pinos se o estilo de envoltória de metal 1 na via for quadrado, o que é mais comum em processos acima de $130nm$. O estilo retangular ainda não é suportado.

As obstruções são camadas especiais utilizadas pelos projetista de leiaute de células para esconder o seu desenho por razões de propriedade intelectual. As geometrias das camadas de polisilício, difusão, contato, etc. são omitidas, deixando apenas a informação mínima necessária para a utilização da célula num fluxo de síntese automatizado. A Figura A.5 mostra um exemplo de célula lógica onde foi escondido todo o leiaute das camadas abaixo de metal 1. Além disto, as redes de metal 1 também foram "desfiguradas" com obstruções para que o roteador não tente fazer conexões por estas regiões.

As posições da grade de roteamento sobre a célula onde é possível a inserção de pinos são marcados com o nome da rede a qual pertencem. É possível que mais de uma posição da grade seja marcada com a mesma rede. Neste caso o roteador considerará todos os pinos como eletricamente equivalentes, podendo fazer a conexão por qualquer um deles.

Os lugares onde é possível a inserção de pinos mas não há redes de metal 1 abaixo e nem obstruções, são liberados para uso durante o roteamento detalhado. Desta forma, o roteador poderá utilizar estes espaços para fazer conexões utilizando metal 1, como por exemplo, com células adjacentes.

Figura A.5: Exemplo de célula encontrada em um arquivo LEF (a) e o seu correspondente leiaute (b) gerado pelo CellGen (ZIESEMER; LAZZARI; REIS, 2007). Redes de I/O estão marcadas com textura em X.



(a) Layout LEF (b) Layout completo

A.9 Planta Baixa (*floorplanning*)

Antes de realizar a etapa de posicionamento e roteamento, é necessário primeiro definir o formato da área e as margens do leiaute do circuito. A altura é definida pelo projetista em número de bandas. Já a largura é calculada automaticamente pelo ASTRAN de acordo com a área total das células e a percentagem de espaços em branco inseridos.

As margens definem o espaçamento dos pinos de entrada e saída do circuito até a área onde as células serão posicionadas.

Uma vez realizada a planta baixa, o leiaute do topo do circuito é criado e todos os elementos são instanciados (ainda que sem posição definida): células e pinos de entrada e saída (interface) do circuito. A Figura A.6 mostra o *template* do leiaute da planta baixa dos circuitos utilizado pelo ASTRAN.

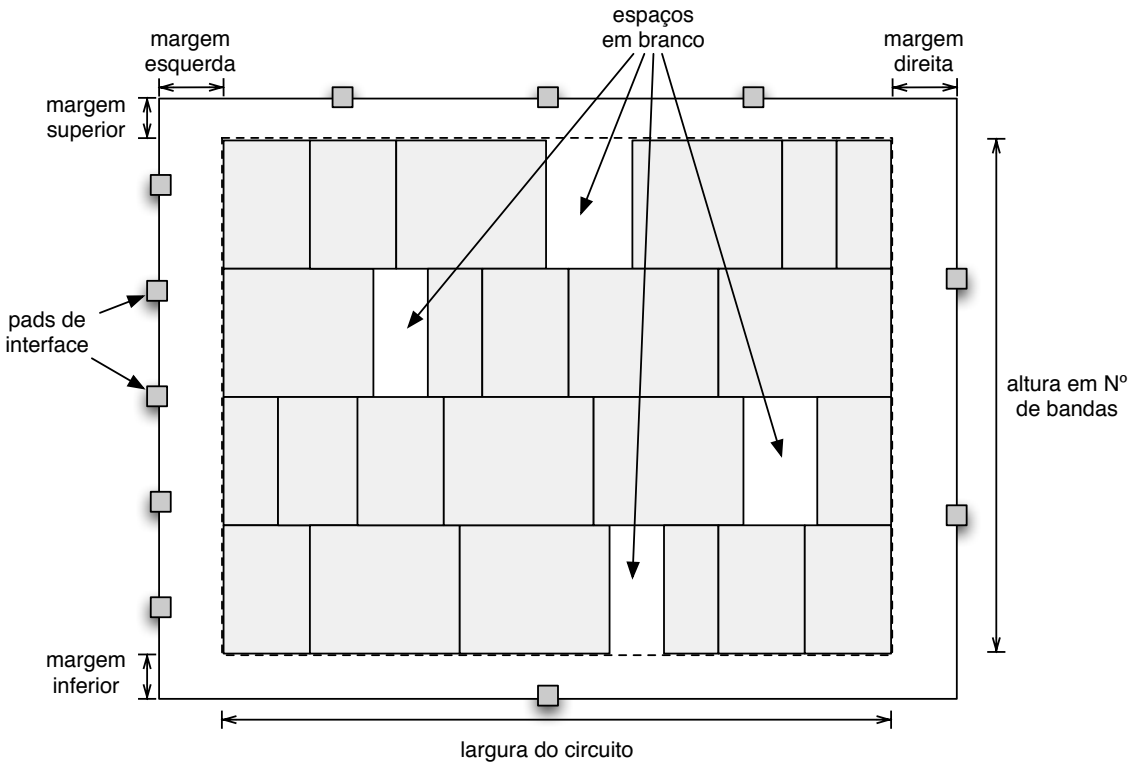
A.10 Posicionamento dos Pinos de Entrada e Saída do Circuito

O posicionamento dos pinos é feito respeitando a direção das interfaces determinada no arquivo SPICE. Esta etapa é importante para a etapa seguinte de posicionamento, uma vez que tendem a guiar o posicionamento das células nesta etapa, reduzindo o comprimento das conexões e/ou atraso do circuito.

Neste trabalho os pinos foram distribuídos igualmente espaçados, de acordo com o lado do circuito a qual pertencem. Esta técnica é ilustrada na Figura A.6.

Esta é uma abordagem que ainda pode ser otimizada em trabalhos futuros. Uma possibilidade é agrupar terminais cuja distância lógica é pequena e separar outros cuja distância é maior.

Figura A.6: Modelo da planta baixa do leiaute de um circuito no ASTRAN.



A.11 Posicionamento das Células

Após o *floorplanning*, todas as células encontram-se sobrepostas umas as outras. O objetivo do posicionamento é: distribuir as células ao longo da área do circuito, de forma que não haja sobreposições, e otimizando uma função de custo (que normalmente leva em consideração: o tamanho total das conexões, os congestionamentos, o atraso dos caminhos críticos, etc.).

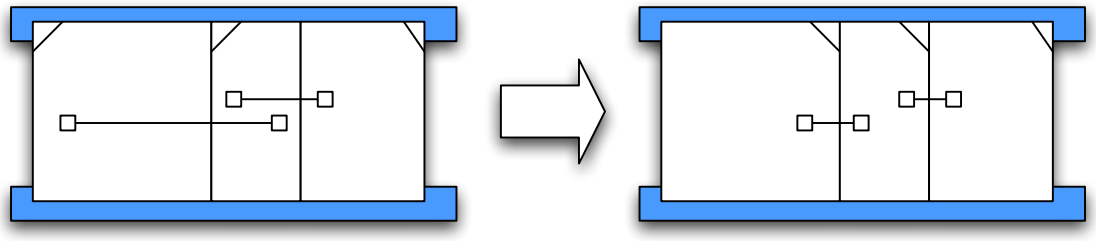
O posicionador que foi integrado ao fluxo do ASTRAN é o zPlace, desenvolvido por Hentschke (HENTSCHKE et al., 2007) e posteriormente mantido por Flach (FLACH; JOHANN; REIS, 2011). Trata-se de um posicionador quadrático, que modela o problema como um sistema de molas. A cada iteração, conforme for ocorrendo sobreposições, forças são adicionadas no sistema com o objetivo de espalhar as células. A partir de determinado ponto, um legalizador é chamado para posicionar as células em posições próximas de onde estavam, mas válidas, eliminando as sobreposições.

O formato de entrada e saída suportado pelo zPlace é o Bookshelf (CALDWELL; KAHNG; MARKOV, 2011) que também foi adotado como o formado padrão do ASTRAN para salvar e ler o posicionamento das células. Além deste, o ASTRAN também importa $.mpp$ do Mango Parrot (BRUSAMARELLO et al., 2004) e exporta $.place$ que é formato utilizado pela Cadence (CADENCE DESIGN SYSTEMS, 2011).

A.12 Espelhamento das Células em Y

Uma característica dos posicionadores utilizados no fluxo do ASTRAN é que as células são posicionadas sem informação da localização dos pinos de E/S das células. O zPlace, por exemplo, considera o centro das células como ponto de origem e destino de todas as conexões. Por esta razão, o espelhamento da célula em Y (MY) não é aprovei-

Figura A.7: Exemplo de uso do espelhamento das células para redução do comprimento das conexões.



(a) Sem espelhamento (b) Com otimização por espelhamento

tado pela ferramenta em benefício do roteamento. Apesar do formato Bookshelf suportar espelhamento das células, este recurso não é utilizado pelo posicionador.

O problema desta abordagem é ilustrado na Figura A.7 (a). Muitas vezes ocorre da célula possuir pinos cujas conexões são feitas com células que estão no lado oposto de onde foram instanciados. Nestes casos, um simples espelhamento da célula em Y contribui para reduzir o comprimento das conexões, facilitando o roteamento do circuito como mostra a Figura A.7 (b).

Por esta razão, foi desenvolvido um algoritmo para otimizar o espelhamento das células chamado "Auto Flip". A ideia por trás deste algoritmo é: para cada instância de célula no circuito, descobrir qual espelhamento contribui mais para aumentar o tamanho das redes as quais se conectam.

Uma vez que as instâncias de células que usualmente mais contribuem para o tamanho da rede são justamente aquelas que ficam sobre a *bounding box* de alguma rede, apenas estas foram consideradas para otimização. Além disto, como o espelhamento em Y contribui apenas para a redução dos limites da *bounding box* sobre o eixo X, apenas estes limites foram utilizados para o algoritmo de otimização mostrado a seguir:

1 - Para cada rede do circuito (excetuando-se as de alimentação) são calculados os limites mínimos X_{min} e máximos X_{max} da sua coordenada no eixo X.

2 - Percorre-se novamente cada rede, a procura de instâncias de células i que tenham pinos posicionados sobre X_{min} ou X_{max}. Para cada instância i encontrada, soma-se a $MYgain[i]$ a contribuição para o aumento ou redução da *bounding box* da rede gerado pelo espelhamento de i.

3 - Para cada instância i cujo $MYgain[i]$ é positivo, espelha-se i sobre o eixo Y, caso contrário, retira-se o espelhamento.

Um exemplo da execução deste algoritmo é ilustrado na Figura A.8. Todas as redes tiveram uma redução na sua largura ou mantiveram o mesmo tamanho inicial.

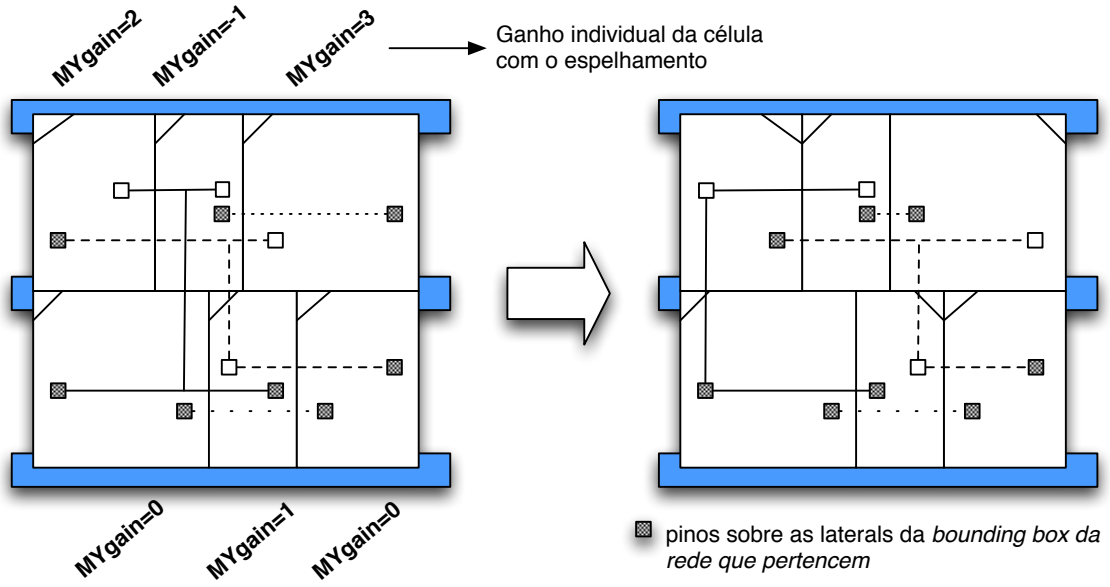
Apesar deste algoritmo não ser ótimo, sua execução é rápida e todos os circuitos testados apresentaram redução de até 2% no comprimento total das conexões.

A.13 Posicionamento Incremental

A função do posicionador incremental é permitir que o fluxo convirja para uma solução válida. Isto normalmente é feito obtendo um *feedback* do roteador, como por exemplo áreas de congestionamento, para que o posicionamento produza um resultado melhor.

O posicionador incremental desenvolvido no ASTRAN examina cada instância de célula do circuito, perguntando para o roteador NBRouter a quantidade de conflitos (nós ocupados por duas ou mais redes) existentes sobre ela. Ao ser detectado algum con-

Figura A.8: Exemplo de execução do algoritmo autoFlip.



(a) Sem espelhamento (b) Com otimização por espelhamento

flito, o posicionador insere espaços em branco junto à célula, virtualmente aumentando sua largura em uma unidade. O ASTRAN modela o posicionamento utilizando equações lineares, onde a posição relativa das células é mantida e uma função de custo para minimização do *wirelength* (apenas no eixo X) e largura do circuito é inserida. Um resolvedor de ILP é então executado para encontrar um novo posicionamento. A cada nova iteração do roteador e do posicionador incremental, se o conflito persistir, a largura dos espaços em branco é aumentada até convergir para uma solução, como mostra a Figura A.9. Neste exemplo o congestionamento foi resolvido deslocando as células para o lado até abrir espaço suficiente para passar a conexão que apresentava conflito.

No ASTRAN, este algoritmo também permite legalizar posicionamentos onde ainda existam células sobrepostas. Caso a largura total das células em uma banda exceda a largura do circuito, esta é automaticamente redimensionada para aquele valor.

A.14 Roteamento

O roteamento é a etapa do fluxo de síntese física responsável por realizar as conexões entre os elementos do circuito, de acordo com o posicionamento obtido e obedecendo as regras de projeto. A especificação de um problema de roteamento consiste da posição dos terminais, do *netlist* - que indica quais elementos devem ser conectados - e a área disponível para roteamento (GEREZ, 1999).

O roteamento assume grande importância devido à crescente influência das interconexões no atraso total do circuito e o acréscimo de área que um circuito de difícil roteabilidade pode gerar. Além disto, o tempo de execução costuma ser um fator limitante, impedindo que algoritmos de maior complexidade computacional sejam usados (LEONHARDT; ZIESEMER; REIS, 2010). Por ser um problema combinatorial de alta complexidade computacional, para fazê-lo tratável é usualmente resolvido por uma abordagem de dois estágios: roteamento global seguido de roteamento detalhado.

A Figura A.10 ilustra este conceito. Inicialmente tem-se o posicionamento inicial

Figura A.9: Execução do posicionador incremental para remoção das sobreposições, otimização do *wirelength* e remoção de congestionamentos.

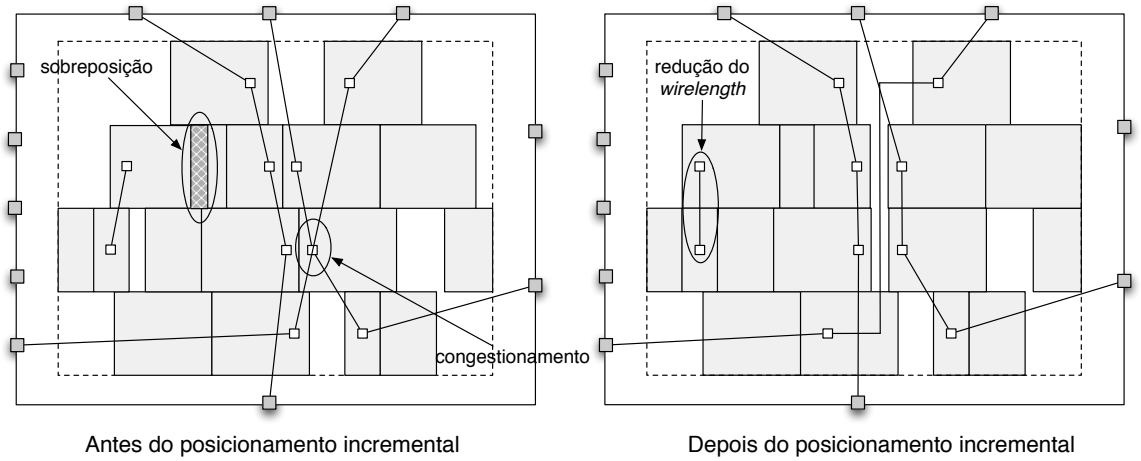
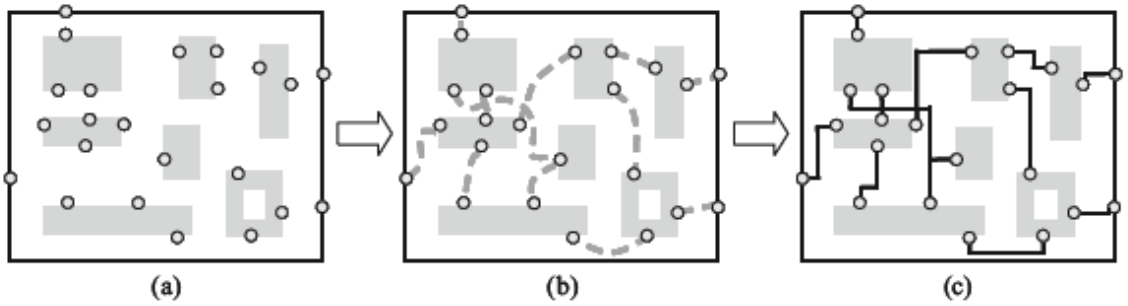


Figura A.10: Visão geral do problema de roteamento : (a) Circuito posicionado; (b) Roteamento Global; (c) Roteamento Detalhado (HUANG-YU; YAO-WEN, 2009).



e a posição dos terminais das células como mostrado em (a). O roteador global então particiona o circuito em regiões e define os caminhos entre as regiões para todas as redes, de forma a evitar congestionamentos, como mostrado em (b). Depois, de acordo com os caminhos obtidos no roteamento global, o roteador detalhado atribui canais e vias para as redes (c).

Dois roteadores são suportados atualmente pelo ASTRAN: o RotDL e o NBRouter. Eles são detalhados a seguir.

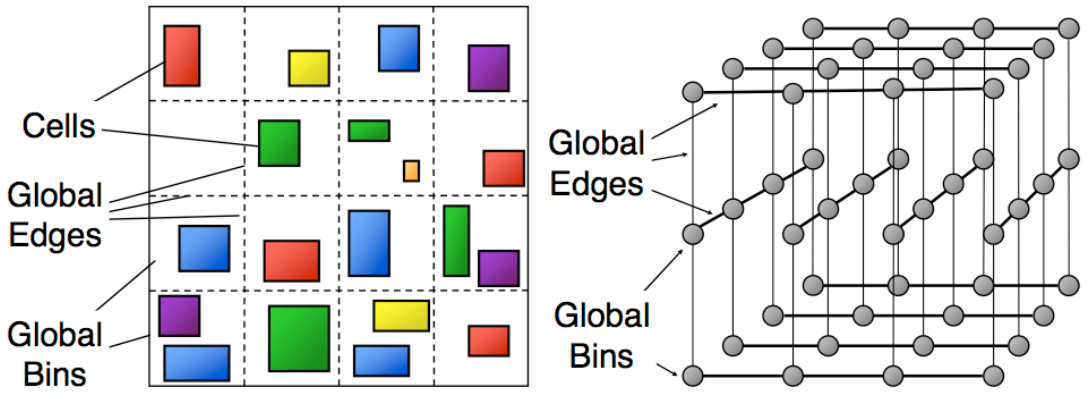
A.14.1 RotDL

O RotDL (FLACH; HENTSCHKE; REIS, 2004) é um roteador baseado em *maze routing* com técnicas de *rip-up* e *reroute* para lidar com bloqueios. Sua principal característica é velocidade e pouco uso de memória. Esta técnica procura solucionar conflitos desfazendo o roteamento das redes conflitantes e fazendo-as novamente. O RotDL não faz roteamento global, somente roteamento detalhado. Por esta razão, circuitos muito grandes (com mais de 1000 redes) e de difícil roteabilidade tendem a demorar tempo excessivo para terminar, além de ocuparem muita memória.

A.14.2 NBRouter

O NBRouter é um roteador global e detalhado desenvolvido em conjunto com o ASTRAN. Ele utiliza técnica de negociação de congestionamento similar ao Pathfinder (MC-

Figura A.11: Circuito dividido em global bins e o respectivo grid (MOFFITT; ROY; MARKOV, 2008).



MURCHIE; EBELING, 1995) e A* para aceleração da busca de nodos pertencentes à mesma rede.

A.14.2.1 Roteamento Global

A etapa de roteamento global dentro do ASTRAN foi desenvolvida em conjunto com Leonhardt (LEONHARDT; ZIESEMER; REIS, 2010).

Enquanto o roteador detalhado é responsável pela definição do leiaute final de interconexão, trabalhando sobre um *grid* detalhado, o roteador global é aplicado sobre uma região com maior granularidade e nível de abstração.

O *grid* de roteamento global é modelado como um grafo 2D, onde os vértices correspondem a um conjunto de regiões retangulares nas quais é dividido o circuito. Essas sub-áreas são conhecidas como *Global Bins* (caixas globais) (MOFFITT; ROY; MARKOV, 2008). Ao contrário do algoritmo de roteamento detalhado, onde vértices e arestas possuem capacidade unitária, no roteamento global eles possuem capacidade proporcional ao número de conexões suportadas pela *Global Bin*. A execução do algoritmo de roteamento global do NBRouter retorna um grafo onde os nós possuem informação do caminho que as redes podem explorar durante o roteamento detalhado.

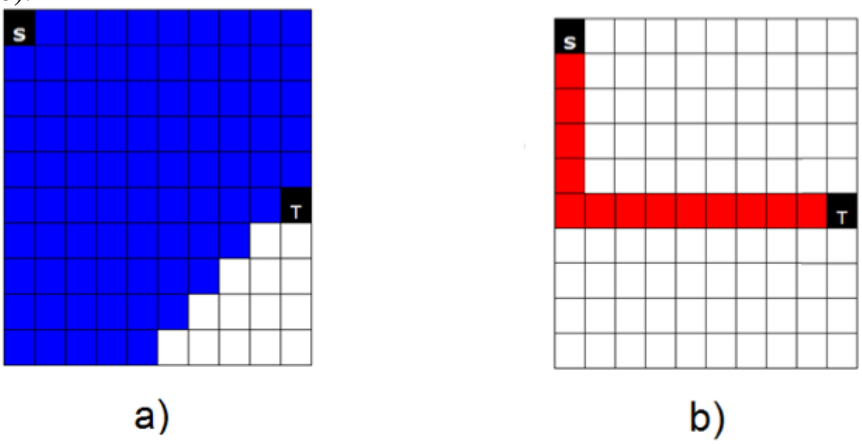
A.14.2.2 Roteamento Detalhado

O roteamento detalhado é a etapa do fluxo de síntese física responsável por realizar as conexões entre os elementos do circuito. Ela pode ser executada diretamente, ou utilizando informações do roteamento global.

Na implementação do NBRouter, caso haja uma grade de roteamento global, o algoritmo de busca de caminho aumenta o custo para expandir nós diferentes dos indicados pelo grafo de roteamento global.

O NBRouter foi desenvolvido como uma adaptação do algoritmo utilizado para roteamento *intracell*, explicado anteriormente. A principal diferença entre eles é o fato deste funcionar sobre uma grade (ao invés de um grafo, sem informação de posição). Isto permite o uso do algoritmo A* (HART; NILSSON; RAPHAEL, 1968) para pesquisa pelo menor caminho, que é mais eficiente que o algoritmo de Dijkstra (DIJKSTRA, 1959). Este algoritmo acelera a busca pois tenta expandir primeiro os nós que levam ao destino, antes de tentar outros caminhos. Cada nó vizinho ao nó atual é adicionado a uma fila de

Figura A.12: Pesquisa utilizando (a) Dijkstra e (b) A* (LEONHARDT; ZIESEMER; REIS, 2010).



prioridade segundo a seguinte fórmula:

$$f(x) = g(x) + h(x)$$

onde $f(x)$ é o valor da função heurística sobre o nó x, $g(x)$ é o custo do percurso do nó inicial até o nó x e $h(x)$ é o custo estimado entre o nó x e o nó final.

A fórmula utilizada para calcular $h(x)$ é a sua distância rectilinear (Manhattan) tridimensional até o nó destino:

$$h(x) = \Delta X + \Delta Y + \Delta Z$$

onde ΔX é o custo horizontal estimado entre o nó de destino e o nó atual (que é usualmente proporcional a distância), ΔY é o custo vertical e ΔZ o custo da diferença entre as camadas de metal.

Como as redes do roteamento detalhado podem ter grau maior que dois (três ou mais nós que precisam ser conectados), a função $h(x)$ precisou ser adaptada pois a busca nem sempre é feita considerando apenas um nó destino, podendo haver vários. A abordagem utilizada foi calcular $h(x)$ para todos os nós de destino e utilizar o menor valor encontrado (a distância até o nó destino que estiver mais próximo).

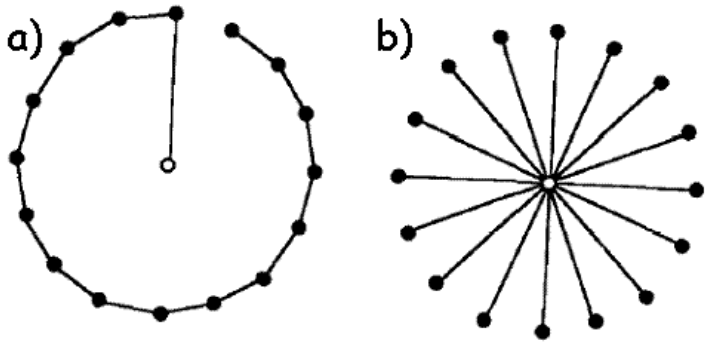
Na hipótese de dois ou mais nós possuírem o mesmo $f(x)$ ao serem inseridos na fila de prioridade (situação comum pois todos os nós dentro da *bounding box* entre os nós de origem e destino tendem a ter o mesmo $f(x)$), a prioridade para remoção da fila é dada para o nó que estiver mais próximo do destino, ou seja, com menor $h(x)$. Assim, menos nós precisam ser expandidos, acelerando o algoritmo.

A Figura A.12 ilustra a diferença na quantidade de nós expandidos entre os algoritmos de Dijkstra e A*.

A.14.2.3 Arborescência

Em roteamento detalhado, arborescência de caminhos mínimos, ou SPT (*Shortest Path Tree*), é uma árvore em que o caminho da origem do sinal da rede a todos os destinos (*sinks*) é minimizado (SANTOS; REIS, 2006). Ela difere da MST (*minimum spanning tree*) por esta não otimizar a distância da origem do sinal até os terminais de destino. A desvantagem é que mais recursos de roteamento são utilizados, dificultando a conexão das outras redes do circuito. A Figura A.13 (a) ilustra o caso extremo onde o roteador

Figura A.13: MST (a) e Arborescência de Caminhos Mínimos (SPT) (b) para o mesmo conjunto de terminais (CONG et al., 1992).



faz um grande trajeto até chegar no último nó, podendo gerar um atraso significativo para esta conexão, enquanto (b) mostra uma SPT com o menor caminho da origem até todos os terminais, mas com maior tamanho da rede.

O NBRouter faz arborescência começando o roteamento pelo nó de origem de sinal da rede. Em seguida, este nó é expandido até encontrar o nó de destino mais próximo. Até este ponto não há diferença para o MST. No entanto, para cada nova iteração do algoritmo, os nós pertencentes à árvore parcial de roteamento (já utilizados nas iterações anteriores) são inicializados com custo $g(x)$ equivalente a sua distância Manhattan até a origem (ao invés de 0, como seria no roteamento com MST).

Um meio termo entre MST e SPT é obtido inicializando estes nós utilizando a seguinte fórmula:

$$g(x) = arbFactor * rectDist(origin, x)$$

onde $arbFactor$ é um fator de ajuste que pode assumir valores entre 0 e 1. Assim, com $arbFactor = 0$ tem-se que todos os nós pertencentes à árvore parcial de roteamento são inicializados com zero, obtendo-se uma MST. Com $arbFactor = 1$, os nós são inicializados com sua distância até a origem, obtendo-se uma SPT. Valores intermediários permitem um ajuste entre ambos os formatos de árvore.

Uma opção que foi considerada é utilizar arborescência somente em redes pertencentes a um caminho crítico, ou melhor, em terminais que são percorridos por um caminho crítico. Esta opção, no entanto, não chegou a ser testada pelo fluxo do ASTRAN não possuir ainda esta informação.

A.14.2.4 Sobre-estimando $h(x)$

A otimalidade do A* é garantida quando a função $h(x)$ é admissível, ou seja, não é sobre-estimada. Com $h(x) = 0$, o algoritmo se comporta de forma similar ao do Dijkstra, expandindo primeiro os nós mais próximos da origem. Por outro lado, ao se sobre-estimar $h(x)$, os nós mais afastados dos destinos terão custo artificialmente inflado. Com isto, o algoritmo insistirá um pouco mais antes de abandonar um percurso que esteja próximo do destino e começar novamente um outro próximo da origem. Isto faz com que o algoritmo acelere, em troca de uma possível perda na otimalidade. Esta técnica é chamada *Weighted A** (LIKHACHEV, 2011).

De fato, sobrestimando a distância até o destino em 20% produziu como resultado uma redução de até 50% no tempo de execução do roteador com perda pouco significativa (até 1% de acréscimo, com média próxima de 0%) no comprimento das conexões.

A.14.2.5 Otimização do roteamento

Ao final, uma etapa de otimização com o algoritmo de *iterated 1-steiner* (KAHNG; ROBINS, 1990) é opcionalmente executada. Nós de steiner são nós especiais que são adicionados ao roteador de sinal para forçar o compartilhamento de um mesmo caminho para dois ou mais nós de destino (*sinks*) de uma mesma rede, diminuindo o comprimento da conexão. Este algoritmo foi explicado em detalhes no Capítulo 4.

De acordo com testes realizados por Leonhardt (LEONHARDT; ZIESEMER; REIS, 2009), a otimização do roteamento diminuiu o comprimento das conexões em 1,3% em média ao custo de um aumento de cerca de 200% no tempo de execução do roteador para circuitos pequenos. No entanto, devido à elevada complexidade computacional do algoritmo, o tempo de execução aumenta exponencialmente se tornando inviável para circuitos com poucas centenas de células.

A.14.2.6 Trabalhos Futuros

O problema em utilizar a distância como custo, é que o atraso das conexões não necessariamente é linear. Camadas diferentes de metal possuem diferentes características elétricas (capacitância e resistência) e o compartilhamento de conexões entre uma origem de sinal e múltiplos destinos também influencia negativamente no sinal. Uma métrica mais adequada, por exemplo, é a utilização do Elmore delay (ELMORE, 1948). Hentschke (HENTSCHKE, 2007) fez um estudo aprofundado sobre diferentes topologias de conexões, e como isto afeta a qualidade do roteamento, em seu roteador Amaze.

A.15 Conclusão

Este capítulo apresentou o fluxo da ferramenta ASTRAN em conjunto com os algoritmos utilizados por esta. A ferramenta desenvolvida foi capaz de gerar com sucesso tanto leiautes de células como também de circuitos completos, dada uma descrição em SPICE de entrada. No entanto, este fluxo é um trabalho ainda em andamento e atualmente suporta geração de circuitos somente para tecnologia maiores que $350nm$.

APÊNDICE B REGRAS DE PROJETO USADAS PELA FERRAMENTA ASTRAN

```
S1P1P1   Minimum POLY1 spacing
S2P1P1   Minimum GATE to GATE spacing
S2P1P1_1 Gate length for rule S2P1P1.2
S2P1P1_2 GATE to GATE spacing if L>S2P1P1.1
S3P1P1   Minimum POLY1 spacing in a dense line end
S1DFP1   Minimum POLY1 spacing to DIFF
S2DFP1   Minimum POLY1 spacing to DIFF if W<S3DFP1
S3DFP1   Transistor Width for rule S2DFP1
E1P1DF   Minimum POLY1 extension of GATE
E2P1DF   Minimum POLY1 extension of GATE in L shape diff
E1DFP1   Minimum DIFF extension of GATE
E1DNP1   Minimum NDIFF extension of GATE when butted to PDIFF
E1DDP1   Minimum PDIFF extension of GATE when butted to NDIFF
R1P1     Maximum ratio of POLY area to touched GATE area
S1DFP2   Minimum POLY2 spacing to DIFF
S1P1P2   Minimum POLY1 spacing to POLY2
W2CT     Fixed CONT size
E1M1CT   Minimum MET1 enclosure of CONT
E2M1CT   Minimum MET1 extension on CONT on at least 2 opposite
sides
E3M1CT   Minimum MET1 enclosure of CONT on all sides if E2M1CT
is not fulfilled
E1DFCT   Minimum DIFF enclosure of CONT
E2DFCT   Minimum DIFF enclosure of CONT (min. 2 oposite sides)
E1P1CT   Minimum POLY1 enclosure of CONT
E2P1CT   Minimum POLY1 extension on CONT on all sides at least
2 opposite sides
E3P1CT   Minimum POLY1 enclosure of CONT if E2P1CT is not
fulfilled
E1P2CT   Minimum POLY2 enclosure of CONT
S1CTP1   Minimum DIFFCON spacing of GATE
S1CTDP   Minimum NDIFFCON spacing of PDIFF
S1CTDN   Minimum PDIFFCON spacing of NDIFF
S1CTDF   Minimum POLY1CON spacing of DIFF
S1CTP2   Minimum POLY1CON spacing of POLY2
```

```
R1M1     Minimum ratio of MET1 area to die area
R2M1     Maximum ratio of MET1 area to connected GATE and
CPOLY area
W2VI     Fixed VIA size
E1M1VI   Minimum MET1 enclosure of VIA
E2M1VI   Minimum MET1 enclosure of VIA on at least 2 opposite
sides
E1M2VI   Minimum MET2 enclosure of VIA
S1M2M2   Minimum MET2 spacing to MET2
S1M2M1   Minimum MET2 spacing to MET1 over CPOLY
S1M1M2   Minimum MET1 spacing to MET2 over CPOLY
R2M2     Maximum ratio of MET2 area to connected GATE and
CPOLY area
W2P1     Minimum GATE length
W2DF     Minimum DIFF width
W2V2     Fixed VIA2 size
S1M1M1   Minimum MET1 spacing to MET1
S2M1M1   Minimum Wide MET1 spacing to MET1
S3M1M1   Minimum MET1 spacing in a dense line end
E1M2V2   Minimum MET2 enclose of VIA2
S1CTCT   Minumum aligned CONT spacing
S2CTCT   Minumum shorted CONT spacing
S3CTCT   Minumum misaligned CONT spacing
E1INDF   Minimum NPLUS extension of DIFF
E1IPDF   Minimum PPLUS extension of DIFF
E1WNDP   Minimum NTUB enclosure of PDIFF
S1DNWN   Minimum NDIFF spacing to NTUB
S2M2M2   Minimum MET2 spacing to WIDE_MET2
A1M1     Minimum MET1 area
W1M1     Minimum MET1 width
W1M2     Minimum MET2 width
W1M3     Minimum MET3 width
W1M4     Minimum MET4 width
W1M5     Minimum MET5 width
W1M6     Minimum MET6 width
W1M7     Minimum MET7 width
W1M8     Minimum MET8 width
W1M9     Minimum MET9 width
W1M10    Minimum MET10 width
A1DF     Minimum DIFF area
S1DFDF   Minimum DIFF spacing to DIFF
S2DFDF   Minimum U shape DIFF spacing
S1VIVI   Minimum VIA spacing to VIA
S1V2V2   Minimum VIA2 spacing to VIA2
S1M3M3   Minimum MET3 spacing to MET3
E1M3V2   Minimum MET3 enclosure of VIA2
E1M3V3   Minimum MET3 enclosure of VIA3
W2V3     Fixed VIA3 size
```

```
E1M4V3   Minimum MET4 enclosure of VIA3
S1M4M4   Minimum MET4 spacing to MET4
W2V4     Fixed VIA4 size
E1M4V4   Minimum MET4 enclosure of VIA4
E1M5V4   Minimum MET5 enclosure of VIA4
S1M5M5   Minimum MET5 spacing to MET5
W2V5     Fixed VIA5 size
S1IPIP   PPLUS spacing to PPLUS
S1ININ   NPLUS spacing to NPLUS
W2V6     Fixed VIA6 size
W2V7     Fixed VIA7 size
W2V8     Fixed VIA8 size
W2V9     Fixed VIA9 size
W2V10    Fixed VIA10 size
```