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

WTROPIC - um Gerador Automático de Macro Células CMOS acessível via WWW

por

JOÃO LEONARDO FRAGOSO

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

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Prof. Dr. Fernando Gehm Moraes
Co-orientador

Porto Alegre, julho de 2001

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Fragoso, João Leonardo

WTROPIC - um Gerador Automático de Macro Células CMOS acessível via WWW / por João Leonardo Fragoso. — Porto Alegre: PPGC da UFRGS, 2001.
89 f.: il.

Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2001. Orientador: Reis, Ricardo Augusto da Luz; Co-orientador: Moraes, Fernando Gehm.

I. Reis, Ricardo Augusto da Luz. II. Moraes, Fernando Gehm. III. Título.

”É preciso afastar todos os elos dos silogismos e entregarmo-nos totalmente à intuição como nosso derradeiro meio, visto que todas as teses traduzidas, de forma direta, uma da outra, desde que já esteja clara a dedução, reduzem-se a uma autêntica intuição.”

Descartes

”...e já está no terreno de quem pensa que tudo o que
não é fotografado é perdido, que é como se não tivesse
existido, e que então para viver de verdade é preciso
fotografar o mais que se possa, e para fotografar o mais que
se possa é preciso: ou viver de um modo o mais
fotografável possível, ou então considerar
fotografáveis todos os momentos da própria vida.
O primeiro caminho leva à estupidez, o segundo à loucura...”

Italo Calvino

in: ”A aventura de um fotógrafo”

A meus pais, Antonio e Vanda, que além de me proverem as condições para meus estudos, foram indispensáveis para a construção de meu caráter e de meus valores.

A Mariana Bohns onde sempre busco força e inspiração para continuar.

Agradecimentos

Diversas pessoas foram significativamente importantes na realização deste trabalho as quais não posso deixar de agradecer.

Agradeço principalmente ao meu professor orientador Ricardo Reis pela oportunidade apresentada de realizar este trabalho, pelo seu exemplo de entusiasmo e otimismo, pelos diversos ensinamentos transmitidos e pela enorme paciência de ter-me como seu aluno.

Agradeço ao meu co-orientador, professor Fernando Moraes, que não impôs dificuldade em juntar-se à tarefa de orientar-me e de contribuir significativamente com o trabalho desenvolvido.

Agradeço a todos os professores do Programa de Pós-Graduação que participaram da minha formação, especialmente os professores do Grupo de Microeletrônica, que mostraram-se presente nas inúmeras vezes em que foram solicitados.

Agradeço aos amigos José Güntzel, Jung Choi, Marcelo Johann que sempre foram fontes de inspiração e motivação para a realização deste trabalho e a todos os colegas do Programa de Pós-Graduação que de uma forma ou outra contribuirão na minha caminhada.

Agradeço às instituições de ensino e pesquisa brasileiras que permitem e estimulam o cultivo da ciência. Na realização deste trabalho, agradeço de forma especial a Universidade Federal do Rio Grande do Sul, ao Programa de Pós-Graduação em Ciência da Computação e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

E finalmente, agradeço a minha família que desde cedo me incentivou na apreciação e no desenvolvimento das virtudes e da ciência, compreendendo-me e permitindo-me crescer.

Sumário

Lista de Abreviaturas

ASIC	Circuito Integrado de Aplicação Específica
CI	Circuito Integrado
CPU	Unidade Central de Processamento (*Central Processing Unit*)
DRC	Verificador de regras de projeto(*Design Rule Checker*)
EPLD	Dispositivo Lógico Programável (*Erasable Programmable Logic Device*)
fig.	Figura
FOTC	Célula totalmente transparente(*Full Over-The-Cell*)
FPGA	Matriz de Portas Programáveis (*Field Programmable Gate Array*)
FUCAS	Síntese Automática *full-custom*
GME	Grupo de Microeletrônica
PPGCC	Programa de Pós-Graduação em Ciência da Computação
pg.	Página
SCCG	Porta Complexa Estática em CMOS(*Static CMOS Complex Gate*)
TRANCA	Abordagem de Células Transparentes (*Transparent-Cell Approach*)
UFRGS	Universidade Federal do Rio Grande do Sul

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta a pesquisa e o desenvolvimento da ferramenta para geração automática de leiautes WTROPIC. O WTROPIC é uma ferramenta para a geração remota, acessível via WWW, de leiautes para circuitos CMOS adequada ao projeto FUCAS e ao ambiente CAVE. O WTROPIC foi concebido a partir de otimizações realizadas na versão 3 da ferramenta TROPIC. É mostrado também, como as otimizações no leiaute do TROPIC foram implementadas e como essas otimizações permitem ao WTROPIC cerca de 10% de redução da largura dos circuitos gerados em comparação ao TROPIC. Como o TROPIC, o WTROPIC é um gerador de macro células CMOS independente de biblioteca.

Apresenta-se também, como a ferramenta WTROPIC foi integrada ao ambiente de concepção de circuitos CAVE, as mudanças propostas para metodologia de integração de ferramentas do CAVE que conduzem a uma melhora na qualidade de integração e a padronização das interfaces de usuário e como a síntese física de um leiaute pode ser então realizada remotamente.

Dessa maneira, obteve-se uma ferramenta para a concepção de leiautes disponível a qualquer usuário com acesso a internet, mesmo que esse usuário não disponha de uma máquina com elevada capacidade de processamento, normalmente exigido por ferramentas de CAD.

Palavras-chave: Internet, Microeletrônica, Síntese Automática, Projeto de CI, CAD, Leiaute.

TITLE: “WTROPIC - AN INTERNET BASED MACRO CELL GENERATOR”

Abstract

This work presents the research and the development of the WTROPIC automatic layout synthesizer tool. The WTROPIC is a remote CMOS layout generator that can be accessed through the WWW and adapted to FUCAS project and CAVE environment. WTROPIC was developed as a set of optimizations to version 3 of TROPIC . Also, it shows how the optimization in the layout of TROPIC was implemented and how those optimization enable the WTROPIC to get about 10% reduction in the width of its generated circuits when it is compared with the TROPIC. As the TROPIC, the WTROPIC is a free library CMOS macro cells generator.

Additionally, this report presents the integration of the WTROPIC tool in the CAVE framework and the modifications that were proposed by WTROPIC in the CAVE integration methodology that lead to improving the quality of integration and standardization of the interface of framework. The report also describes how the layout physical synthesis can be done remotely.

In this way , an automatic generator tool was developed, available to any user with internet access, even if that user does not have a computer with a powerful processor, as is usually required by those tools.

Keywords: Internet, Microelectronics, Automatic Synthesis, IC Design, CAD, Physical Design.

1 Introdução

Desde da introdução do primeiro circuito integrado comercial em 1961, a capacidade desses componentes tem crescido com passos largos. Os circuitos integrados comerciais no início da década de 60 possuíam em torno de algumas dezenas de transistores. Atualmente, com o grande avanço nas tecnologias de fabricação, circuitos de muito alto grau de integração (VLSI - *Very Large Scale Integration*) contendo mais dezenas de milhões de transistores estão sendo fabricados e o número de transistores continua dobrando a cada dezoito meses conforme a regra de Moore [MOO 79].

Logo, o projeto de circuitos integrados com esse largo número de transistores é uma tarefa árdua, penosa e com custos consideráveis. Nesse sentido, diversas ferramentas de CAD (*Computer-Aided Design*) foram desenvolvidas para ajudar os projetistas de circuitos integrados em suas atividades. Essas ferramentas de projeto tem como principais objetivos: (1) gerenciar a complexidade dos circuitos, (2) aumentar a produtividade dos projetistas e (3) permitir projeto de circuitos com bom desempenho e baixo consumo. Obviamente, a primeira exigência em um projeto é atender à especificação inicial. Entretanto, existem diversas características que um projetista gostaria de otimizar, as mais importantes são:

- **Área**: reduzir a área do circuito não é só importante por diminuir o silício utilizado, mas também por aumentar o rendimento e o desempenho. Nem todos os circuitos fabricados funcionam adequadamente, rendimento significa aqui a porcentagem de circuitos corretos. Defeitos no cristal, nas máscaras, devido a contatos com impurezas afetam menos quanto menores forem os circuitos.

- **Velocidade**: quanto mais rápido os circuitos executarem suas tarefas, mais atrativos esses circuitos serão. Aumentar a velocidade implica, em alguns casos, em aumento de área (como no caso de duplicação de componentes para permitir a paralelização de operações). Por outro lado, a redução de área através de melhor compactação do circuito, permite reduzir o comprimento médio das conexões, diminuindo os atrasos dos sinais. O processo de projeto deve sempre considerar o compromisso entre área e velocidade, haja vista que a velocidade de um circuito é parte de sua especificação e a área deve ser minimizada sem violar a especificação.

- **Dissipação de energia**: quando um circuito dissipa muita energia, esse pode aquecer em demasia e cessar seu funcionamento ou requerer dissipadores adicionais para o resfriamento. Em algumas aplicações o consumo de energia é de extrema importância ao projeto, por exemplo, em equipamentos portáteis alimentados por

baterias. Novamente, existe um compromisso entre essa característica e as outras: projetando-se para reduzir o consumo pode conduzir, por exemplo, ao aumento da área pela inclusão de hardware para controle de consumo.

- **Tempo de projeto**: normalmente não é um objetivo do próprio projeto, mas é uma requisição econômica. Então, um circuito satisfazendo a especificação deve ser disponibilizado o mais cedo possível, permitindo, na maioria das vezes, um retorno mais adequado dos investimentos. A diminuição do tempo de projeto implica em menor tempo para otimizações de outras características.

- **Testabilidade**: devido a possibilidade de um significante número de circuitos fabricados conter defeitos, todos devem ser passíveis de teste antes de seu uso. É importante, então, realizar o teste do circuito com facilidade já que o custo dos equipamentos de testes são elevados e deve-se minimizar o tempo de teste de um único ”chip”. Freqüentemente, aumentar a capacidade de teste de um circuito implica em aumento de área.

Pode-se combinar todas essas características em uma função de custo e a importância em otimizar a função custo pode ser observada na Tabela 1.1, onde estima-se o crescimento do custo de produção e a redução no número de peças produzidas do processador Pentium em relação ao aumento da área [MUR 96]. Porém, pela enorme complexidade dos circuitos VLSI, é impossível ao projetar um circuito VLSI tentar encontrar o ponto mínimo de sua função custo.

TABELA 1.1 – Estimativa do efeito de aumento de área no aumento de custo de produção do Pentium [MUR 96]

		Aumento de Área	
	Nominal	1%	15%
Área	160,2 mm^2	161,8 mm^2	184,2 mm^2
Custo	100%	101,5%	122%
Aumento do custo anual	-	U$ 63,5Milhões	U$961Milhões
Número de peças por semana	498,1Mil	97%	67,8%

Os conceitos de hierarquia e abstração ajudam a administrar tamanha complexidade. A hierarquia mostra a estrutura do projeto em diferentes níveis de descrição, enquanto a abstração mascara detalhes de níveis inferiores. A abstração torna possível a compreensão e domínio dos problemas porque, escondendo detalhes não relevantes a uma determinada etapa do projeto ou agrupando estes detalhes em uma visão simplificada, limita o número de elementos do projeto que interagem em um certo nível de hierarquia. Cada componente da abstração é decomposto em outras sub-partes interagindo em um nível

inferior de abstração e, essa decomposição, continua até que blocos básicos, passíveis de construção, sejam alcançados, como por exemplo a descrição dos transistores. O projeto de um circuito VLSI com milhões de transistores sem a utilização de hierarquia e abstração tornaria-se inviável.

Adicionalmente, o uso de uma única hierarquia não é suficiente para descrever o processo de projeto VLSI. Existe um consenso geral em distinguir três domínios de projeto:

- **Comportamental**: onde o projeto, ou partes do projeto, é visto como uma caixa preta, e a relação entre as entradas e saídas é dada sem nenhuma relação quanto a sua implementação.
- **Estrutural**: onde o projeto é visto como sub-partes interligadas. Nesse domínio preocupa-se com quais partes são usadas e como elas se relacionam.
- **Físico**: onde define-se a forma do projeto e/ou partes do projeto e qual será a geometria de cada bloco que constitui o projeto e em que posição, em relação às outras partes, esse bloco deve ser colocado.

Cada um desses domínios e a suas hierarquias podem ser vistos no gráfico Y [GAJ 83] mostrado na Figura 1.1. Cada eixo representa um domínio e o nível de abstração diminui em direção ao centro do gráfico.

Também, é possível identificar no gráfico uma metodologia de projeto indicado na Figura 1.1 por linhas pontilhadas unindo os diversos pontos de abstração. Claramente a metodologia mostrada não é única, diversas outras formas de percorrer os eixos podem ser viabilizadas e proporcionarem outras metodologias de projetos.

Em um projeto de VLSI, implementar e realizar uma metodologia implica em diversas ações e ferramentas. Independentemente da metodologia adotada, as ferramentas podem ser agrupadas por meio da função a que se destinam em: (1) ferramentas de síntese, (2) ferramentas de verificação, (3) ferramentas de análise, (4) ferramentas de otimização e (5) ferramentas de gerenciamento de projetos.

Claramente observa-se tarefas destinadas a executar uma transição entre dois pontos do gráfico Y, no mesmo domínio ou em domínio diferentes, essas tarefas são chamadas de síntese. A síntese adiciona detalhes a uma descrição de maneira a diminuir o seu grau de abstração ou mudar o domínio da descrição. Uma síntese pode ser realizada automaticamente pelo uso de **ferramentas de síntese** ou manualmente (algumas ferramentas permitem a intervenção do usuário para a tomada de certas decisões de projeto).

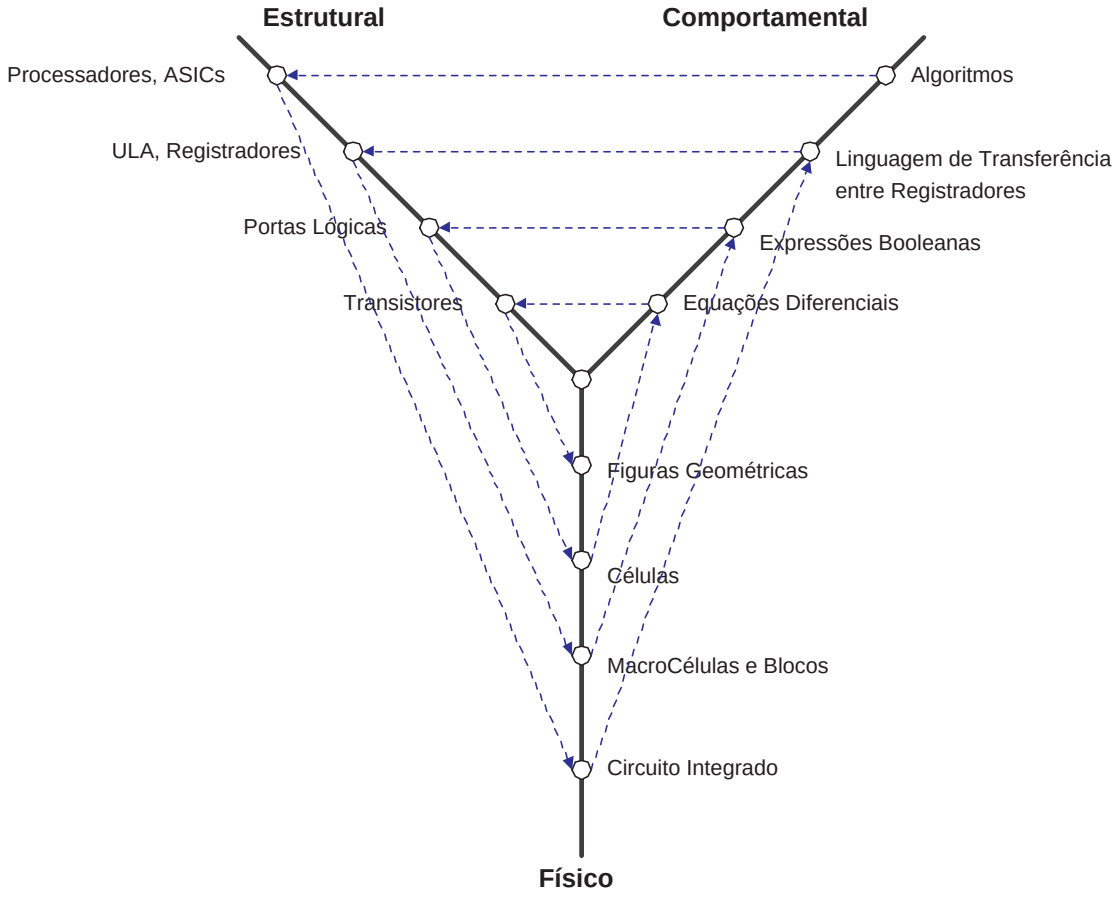

FIGURA 1.1 – Gráfico Y de Gajski ilustrando os três domínios de projeto [GAJ 83].

Idealmente, um passo de síntese é correto por construção, isto é,a transformação requerida para obter a nova descrição preserva o comportamento da descrição anterior. Quando a síntese não garante um resultado correto por construção, são necessárias **ferramentas de verificação**. Normalmente, passos de geração completamente automáticos podem ser aceitos como corretos por construção. Entretanto, a criatividade e a habilidade do projetista não podem ser substituídas por um programa e a intervenção do projetista é necessária em diversos estágios da síntese. Contrapondo-se, humanos são sujeitos a falhas de tempo em tempo e a maioria do seus erros podem ser detectados por ferramentas de verificação. Mesmo processos totalmente automatizadas devem ser verificados, pois as ferramentas de síntese podem conter falhas.

As **ferramentas de análise** destinam-se a extrair ou estimar, de um determinado nível de descrição, dados do projeto, tais como: a freqüência máxima de operação de um circuito, o consumo de energia, a área necessária para a sua construção, etc. e ajudam o projetista a identificar as alterações que podem causar maior benefício em otimizar o projeto.

Outro grupo de ferramentas importantes é o grupo das **ferramentas de otimização**. As ferramentas de otimização aumentam a qualidade do projeto sem, necessariamente, fazer uma transição no gráfico Y, simplesmente devem obter uma nova descrição otimizada no mesmo nível de abstração. Por exemplo, obter equações equivalentes ao conjunto de equações que descrevem um circuito, diminuindo o custo para realização e a complexidade dessas equações.

Existem também, ferramentas que não contribuem diretamente para a realização de um projeto, mas servem de auxílio a outras ferramentas: são as **ferramentas para gerenciamento do projeto**. Essas ferramentas servem para o controle do armazenamento de dados do projeto, comunicação entre ferramentas, a execução de ferramentas em momento adequado, etc.

Nesse contexto, diversas ferramentas têm sido propostas para atender a cada necessidade do projeto e adequadas a uma determinada metodologia de projeto. Também têm sido propostos, CAD *Frameworks* que são ambientes capazes de auxiliar e fornecer ferramentas para todas as etapas do projeto. Esses ambientes possuem a vantagem de fornecer uma interface única ao projetista, facilitando, assim, seu treinamento e uso das ferramentas.

1.1 A Metodologia TRANCA, o Projeto FUCAS e o Ambiente CAVE

O desenvolvimento de ferramentas de síntese de leiaute para blocos de lógica aleatória é um dos objetivos do GME/PPGC da UFRGS. Como resultado deste esforço, existem atualmente alguns módulos de geração automática disponíveis, alguns destes para ambiente WINDOWS e outros para ambiente UNIX. Para cada ambiente, existe interfaces para gerenciamento de ferramentas de síntese e das demais ferramentas de apoio, tais como extrator, planificador, conversores, exibidores de posicionamento e editores de leiautes. As ferramentas de síntese automática de leiaute adotam subconjuntos de características propostas pela metodologia TRANCA. O subconjunto, por sua vez, depende dos aspectos práticos de cada abordagem.

No esforço de automatização do processo de geração de leiautes de circuitos integrados, freqüentemente têm sido adotadas topologias simplificadas, de modo a facilitar a implementação de algoritmos e heurísticas capazes de tratar os problemas ligados ao posicionamento e roteamento das células. Porém, esse processo de simplificação implica em uma diferença muito grande de qualidade entre as soluções manuais (*full-custom*) e as automáticas. De um modo geral as soluções automáticas tendem a ser bastante piores do

que as manuais no que concerne à densidade de transistores.

A metodologia TRANCA [REI 87] propõe um conjunto básico de procedimentos, os quais são resultado da observação dos leiautes de blocos em lógica aleatória de diversas outras aplicações. A utilização de tais procedimentos numa ferramenta de síntese automática equivale a incorporar parte da experiência de um projetista humano, o que tende a melhorar a qualidade dos leiautes gerados.

A metodologia TRANCA foi desenvolvida a partir da observação de partes de controle de microprocessadores projetados em meados dos anos 70 [REI 83]. Estas partes de controle foram desenhadas manualmente e aliavam uma alta densidade de transistores com um alto desempenho elétrico considerando as tecnologias disponíveis na época.

As características fundamentais da metodologia TRANCA são a utilização de uma estrutura de bandas, onde são inseridas as células provenientes de uma biblioteca, ou geradas automaticamente, e a realização do roteamento sobre estas células [JOH 95] (abordagem *over-the-cell routing*), segundo um esquema de alocação ordenada de trilhas. As vantagens decorrentes destas características são perceptíveis sob dois aspectos: (1) economia de área e (2) melhora do desempenho elétrico.

A supressão dos canais de roteamento implica numa redução apreciável de área. Conseqüentemente, o comprimento médio das conexões fica reduzido, contribuindo para melhorar o desempenho elétrico. Por outro lado, a topologia das células deve suportar a passagem de um número conveniente de trilhas, a serem utilizadas nas conexões intercelulares. Com a realização do roteamento sobre as células, a distância média entre as trilhas e os terminais das células é menor, e particularmente, conexões entre células vizinhas podem ser feitas por justaposição, resultando em caminhos mais curtos. Esta característica tem duplo efeito: proporciona melhor desempenho elétrico ao circuito e reduz a área ocupada pelo roteamento.

Comparativamente aos *standard-cells*, onde mesmo conexões entre células vizinhas são realizadas utilizando os canais, os ganhos em termos de área e desempenho elétrico são apreciáveis.

A Metodologia TRANCA é caracterizada por [REI 87]:

- transparência de células e blocos funcionais;
- uso de uma estrutura em bandas;
- gerenciamento de trilhas.
- maleabilidade de blocos funcionais e de células;

O uso da metodologia TRANCA norteia o projeto de circuitos de modo a produzir uma utilização mais eficiente da superfície de silício, sendo ainda passíveis de automatização. A adoção dos procedimentos anteriormente descritos resulta em leiautes mais compactos. Mas a primeira vantagem é que tais procedimentos são perfeitamente passíveis de automação. Outra grande vantagem é que a flexibilidade na implantação da lógica, permite efetuar otimizações significativas quantos aos aspectos de desempenho e consumo.

O Projeto FUCAS (*FUll-Custom layout Automatic Synthesis* - síntese automática de leiautes de circuitos *full-custom*), dentro da metodologia TRANCA, objetiva o desenvolvimento de um conjunto de ferramentas, visando a obtenção de um sistema de síntese que permita a realização de circuitos *full-custom* de forma inteiramente automática, desde a sua especificação lógica até a geração do leiaute. O sistema deverá efetuar um mapeamento lógico visando minimizar o número de transistores (portanto diminuindo consumo de energia), e visando diminuir os caminhos críticos. A Figura 1.2 mostra um possível fluxo de concepção de circuitos integrados utilizando o projeto FUCAS.

A etapa de mapeamento tecnológico visa transformar a descrição lógica de um circuito em uma rede de portas lógicas que serão efetivamente realizadas à nível de silício [DEV 94]. O processo normalmente utilizado de mapeamento tecnológico está restrito ao uso das células disponíveis em uma biblioteca de células.

No projeto FUCAS, o projetista vai poder efetivamente explorar a etapa de otimização lógica ao máximo, pois não está limitado a um conjunto de células lógicas pré-definidas. Os estudos iniciais já permitem considerar a possibilidade de obter uma redução de cerca de 30% [REI 97] em termos de número de transistores.

A etapa de geração das células tenta obter uma topologia de leiaute otimizada e tornar possível ao projetista o dimensionamento automático de transistores, segundo as especificações do circuito em desenvolvimento. Esta ferramenta substituirá o uso da biblioteca de células utilizada anteriormente.

Aliado a esses esforços, o projeto CAVE [IND 98] propõe uma metodologia para o desenvolvimento e a integração de ferramentas em um ambiente único (*framework*) baseado em WWW. Os objetivos do projeto são: (1) acelerar o processo de concepção de circuitos, (2) diminuir o tempo de treinamento de projetista pelo uso de uma interface bem conhecida e (3) permitir o uso por uma equipe distribuída.

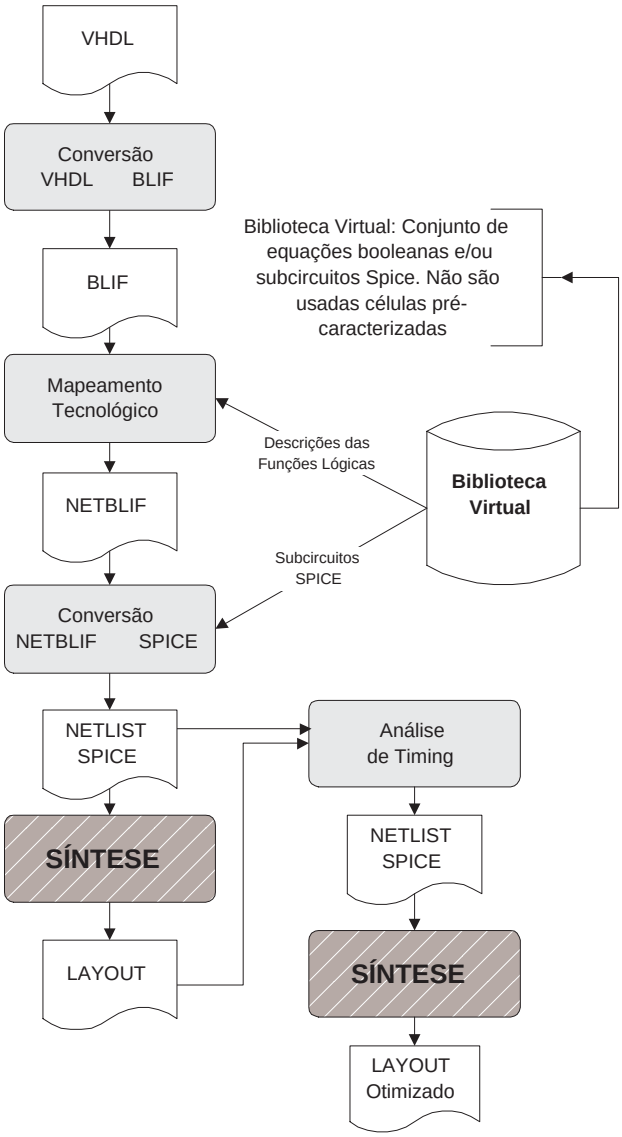

FIGURA 1.2 – Possível fluxo de concepção de CI no projeto FUCAS.

1.2 Objetivos e Plano de Apresentação

Dentro do contexto do projeto FUCAS e do ambiente CAVE , o presente trabalho apresenta a ferramenta WTROPIC, nova versão da ferramenta TROPIC [MOR 94].

O WTROPIC é uma ferramenta para a geração de leiautes de macro-células CMOS de lógica aleatória, independente de tecnologia e sem o uso de bibliotecas de células adequando-se, assim, às necessidades do projeto FUCAS. O WTROPIC apresenta otimizações na topologia de leiaute em relação a terceira versão do TROPIC, alcançando melhor resultados de densidade de transistores por milímetro quadrado.

Adicionalmente, visando a integração no ambiente CAVE e permitir o uso por pro-

jetistas menos habituados à ferramenta, foram desenvolvidos uma interface gráfica baseada em WWW utilizando a linguagem Java, e um servidor capaz de encapsular as tarefas de síntese.

O capítulo 2 mostra o estado da arte em geração de leiautes e síntese física, as características, classificações e as ferramentas desenvolvidas. O capítulo 3 descreve a ferramenta TROPIC, suas características, estilo de leiaute e os passos para geração do leiaute a partir de um netlist SPICE. O capítulo 4 mostra as otimizações realizadas na ferramenta TROPIC para a nova versão WTROPIC e os resultados da geração para a nova ferramenta. O capítulo 5 mostra o ambiente de projeto CAVE, a ferramenta WTROPIC e como o gerador de células foi integrado ao *framework* e o capítulo 6 apresenta as contribuições, conclusões do trabalho realizado e perspectiva de trabalhos futuros.

2 Sintetizadores de Leiautes

Desde a introdução dos primeiros circuitos integrados, diversos métodos foram propostos de maneira a automatizar a síntese dos leiautes desses circuitos. Essa tarefa corresponde a última etapa mostrada no gráfico-Y da Figura 1.1, logo, sintetizar o leiaute significa, a partir de uma descrição estrutural de um circuito, desenhar os polígonos necessários, em suas diferentes camadas, para a fabricação do componente.

2.1 Matriz de Weinberger

A matriz de Weinberger [WEI 67] é o primeiro método utilizado para a síntese física de funções booleanas de vários níveis aplicando um estilo de leiaute fixo. Um circuito contendo apenas portas lógicas *nor* (como o exemplo da Figura 2.1) é convertido em uma matriz unidimensional de portas nMOS, com uma única porta por coluna. Cada coluna consiste de duas linhas verticais de metal. Uma linha de metal conectada com o transistor de *pull-up* e que serve como a linha de saída da porta, enquanto a outra linha de metal está conectada com a linha de alimentação GND. Em aplicações reais duas portas consecutivas compartilham a mesma linha de alimentação GND. Os sinais de entradas para as portas são derivados de linhas paralelas de polissilício, desenhadas horizontalmente, cruzando por todas as portas.

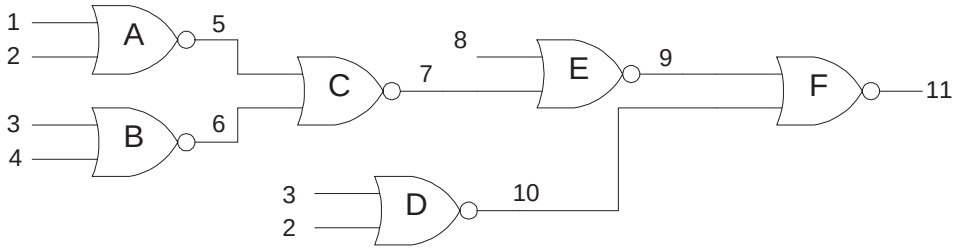

FIGURA 2.1 – Um circuito exemplo de lógica NOR.

O tamanho do leiaute da matriz de Weinberger é proporcional ao produto do número de portas(colunas) pelo número de sinais(linhas). Em um leiaute regular, cada porta ocupa toda uma coluna e cada sinal (entrada, saída ou interno) ocupa a totalidade de uma linha. Apesar dessa abordagem ser simples, provoca como resultados para o leiaute matrizes esparsas.

Pode-se observar, entretanto, no exemplo mostrado na Figura 2.2 que nenhum sinal realmente ocupa uma linha inteira. Na prática, diversos sinais podem compartilhar uma

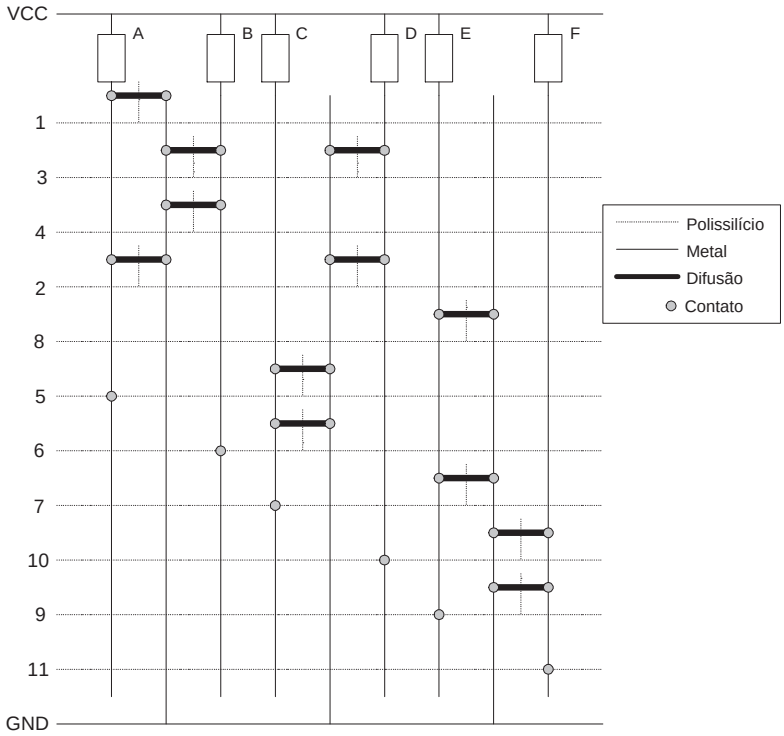

FIGURA 2.2 – Matriz de Weinberger para o circuito da Figura 2.1.

mesma linha permitindo assim que altura da matriz seja reduzida e otimizando o leiaute final. O problema de definir quais sinais ocupam cada linha de forma a minimizar o número de linhas é conhecido como problema de assinalamento de linha.

O algoritmo *left-edge* [HAS 71] comumente é utilizado para obter uma solução ótima para o problema de assinalamento das linhas. A Figura 2.3 mostra a matriz de Weinberger da Figura 2.2 após a utilização do algoritmo *left-edge* para o assinalamento das linhas.

Apesar do problema de assinalamento das linhas poder ser simples, é importante salientar que o resultado do assinalamento das linhas é dependente direto da ordem das portas nas colunas, isto é, diferentes colocações das portas resultam em diferentes resultados para o assinalamento.

2.2 Gate-Array

O processo de concepção utilizando *gate-array* basea-se no uso de um conjunto de células pré-difundidas. Todos os circuitos gerados por essa abordagem possuiram o mesma matriz de células. Entretanto, a função lógica do circuito é especificada pelas

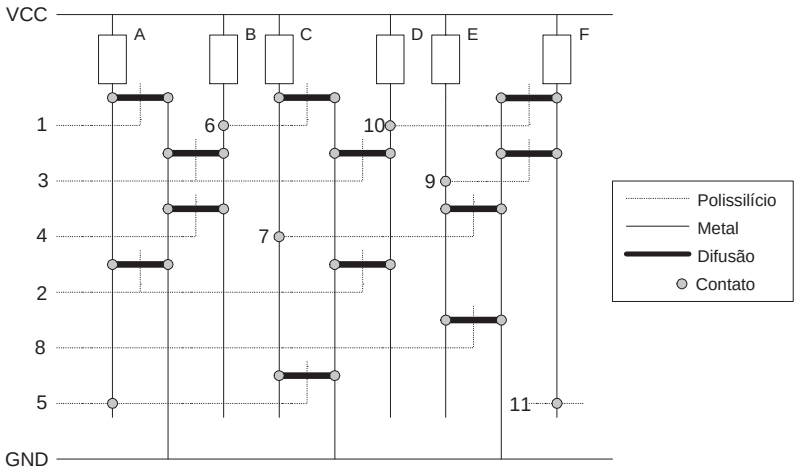

FIGURA 2.3 – Matriz de Weinberger após otimização usando o algoritmo de *left-edge* para o assinalamento das linhas.

camadas de metalização, que personalizam cada circuito e implementam a função lógica desejada.

Obviamente, as células que compõe a matriz, são elaboradas para o funcionamento sobre diversas circunstâncias, o que, normalmente, ocasiona o sobre-dimensionamento dos transistores que compõe as células. Dessa maneira, essa abordagem apresenta problemas de consumo e/ou desempenho. Em contra partida, o tempo de concepção e o baixo custo de fabricação são pontos favoráveis a esta metodologia.

Adicionalmente, a matriz prevê canais de roteamento para a realização desse, que também, normalmente, contém um número de linhas maior que a maioria dos casos exige. Assim, esta metodologia também apresenta problemas no roteamento e na densidade de transistores obtida. Como alternativa para esta abordagem, existe o estilo *sea-of-gates*, onde os canais de roteamento são suprimidos e o roteamento é realizado sobre as células, o que, obviamente, conduz a um aumento na densidade de integração e reduz a área necessária para concepção.

2.3 Standard-Cells

A arquitetura *standard-cells* divide a área do leiaute em linha paralelas, denominadas bandas, e entre cada par de linhas existe um canal de roteamento, como mostrado na Figura 2.4. Os sistemas baseados em *standard-cells* usam uma abordagem de posicionamento e roteamento (*place-and-route*), pois o problema de geração do leiaute é divido nessas duas etapas.

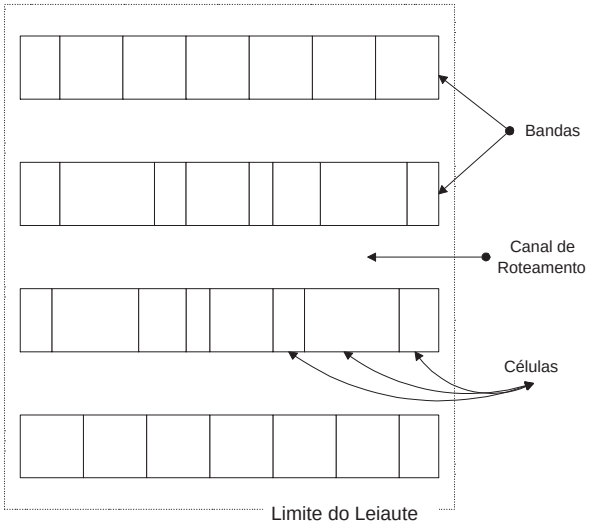

FIGURA 2.4 – Arquitetura de Standard Cell .

O sistema de projeto mantém uma biblioteca de células pré-projetadas, representando todas as possibilidades de unidades básicas a serem utilizadas por um circuito. Todas as células de uma biblioteca possuem a mesma altura fixa e, normalmente, com largura variável. Os pinos de entrada e saída das células são colocados na borda superior e inferior de seu limite para que possam ser unidos através do canal de roteamento.

O gerador do leiaute deve então, selecionar as células da biblioteca correspondentes às unidades básicas do circuito a ser concebido, decidir o posicionamento adequado de cada célula e realizar o roteamento das células de acordo com a descrição inicial. Quando há a necessidade de um sinal atravessar uma banda, células especiais, denominadas células de passagem, são utilizadas.

A vantagem da arquitetura *standard-cells* está na simplicidade do processo de geração. O fato de necessitar apenas do posicionamento e roteamento das células tornou este método popular, conduzindo ao desenvolvimento de diversos algoritmos. Outra vantagem desse método é sua confiabilidade, haja vista que todas as células são pré-caracterizadas antes de serem armazenadas na biblioteca. Logo, o projeto é livre de erros e o seu desempenho pode ser previsto.

Entretanto, a precisão das estimativas de desempenho e consumo desse método tem diminuído com o avanço das tecnologias submicrônicas, pois a contribuição das conexões no desempenho de circuitos de lógica aleatória e com algumas dezenas de milhares de transistores , nessas tecnologias, torna-se cada vez mais significativa, como mostra a Figura 2.5. Dessa maneira, é fácil notar que o desempenho do circuito será determinado pelas conexões realizadas pelo roteamento, reduzindo a importância das características individuais das células da biblioteca para a previsão do comportamento do circuito.

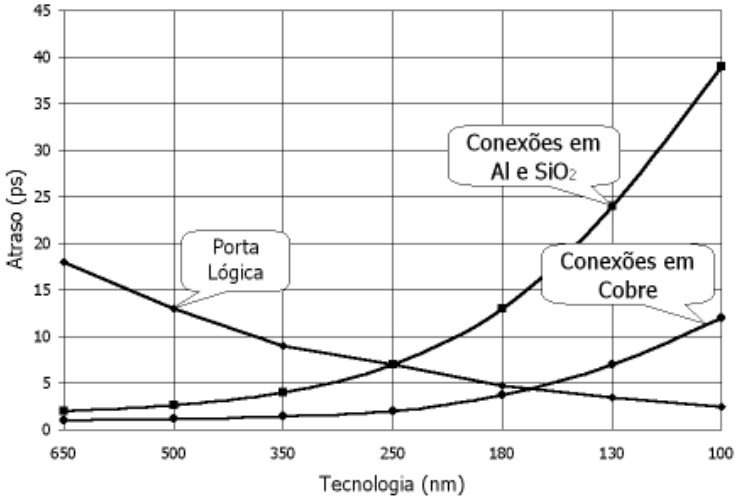

FIGURA 2.5 – Contribuição do atraso em tecnologias submicrônicas [SIA 97].

As principais desvantagens dessa abordagem são a baixa utilização de área e a manutenção da biblioteca. Devido a forma de disposição dos pinos, tamanhos fixos das células e uso de canais para o roteamento, 60% a 70% da área, normalmente em circuitos de lógica aleatória, é consumida para a realização das conexões entre as células. Adicionalmente, mudanças nas tecnologias de fabricação impõem redesenho das células da biblioteca, que normalmente são projetadas para uma determinada tecnologia.

2.4 Módulos Regulares

Diversos circuitos, tais como memórias, somadores, multiplicadores e lógica aleatória de dois níveis, obedecem a uma arquitetura definida e seu processo de concepção restringe-se simplesmente a aumentar ou diminuir o número de repetições de um pequeno conjunto de componentes para a sua geração de acordo com as especificações.

Dessa maneira, diversos geradores de módulos são utilizados para gerar esses circuitos, bastando ao projetista apenas definir alguns parâmetros para a personalização do circuito.

2.5 Gate Matrix

O método *gate matrix* foi introduzido por Lopez e Law [LOP 80] como uma abordagem sistemática para o problema de síntese de leiaute de larga escala. Nesse método as

células complexas CMOS estáticas são dispostas em linhas verticais de polissilício regularmente espaçadas usadas somente para conectar os gates dos transistores. Os transistores são formados pela intersecção de linhas de difusão dispostas horizontalmente através das linhas de polissilício. Dessa maneira, redes que conectam o dreno e a fonte de dois transistores vizinhos podem ser implementados em difusão, ou, no caso de transistores não vizinhos, implementadas através da primeira camada de metal também disposta em linhas horizontais. Tipicamente, o número de colunas de polissilício é igual ao número de entradas/saídas do circuito, haja vista, transistores que utilizam o mesmo sinal de gate são construídos sob a mesma coluna.

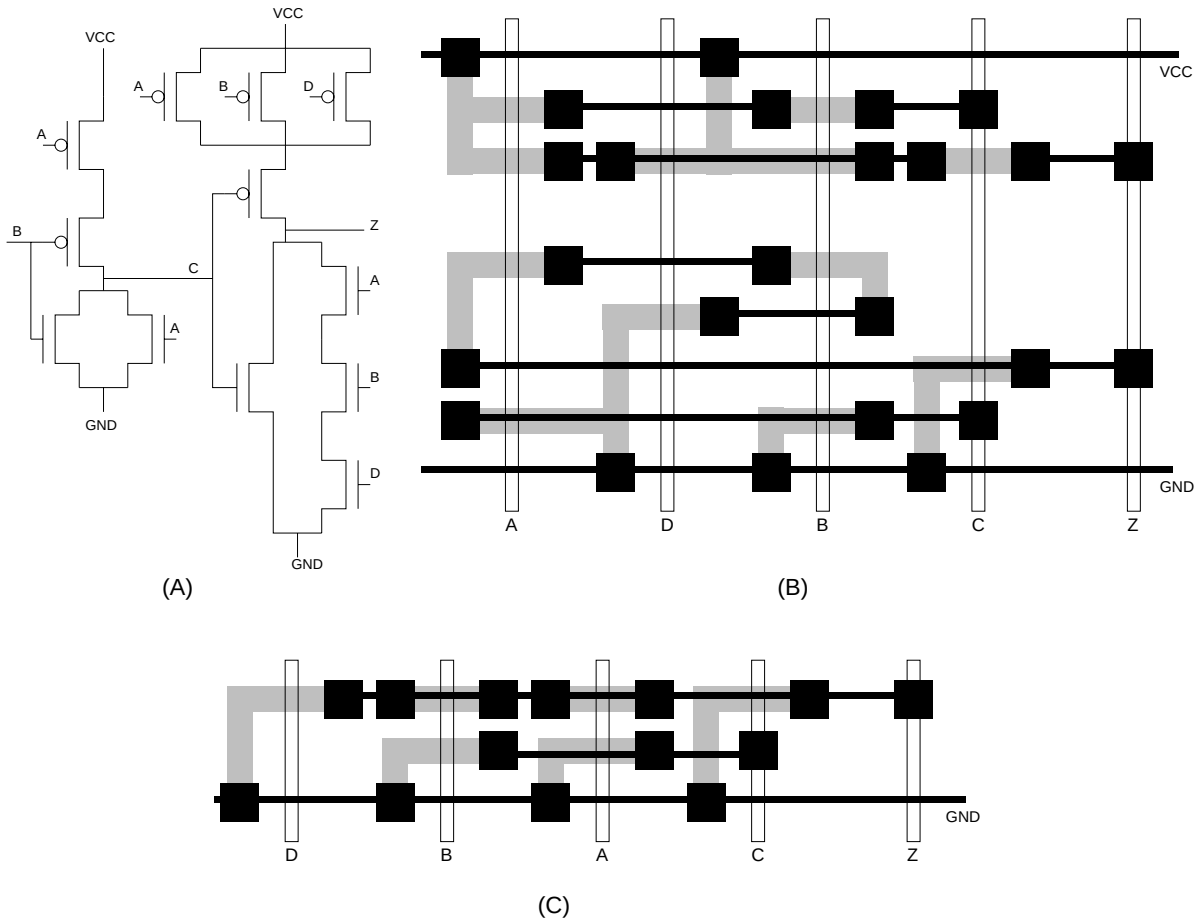

FIGURA 2.6 – Um exemplo de leiaute Gate Matrix (A) esquema do transistores (B) leiaute gate matrix e (C) Otimização do plano N pela permutação de colunas [PRE 85].

Existem diversas automatizações para a abordagem gate matrix e entre elas citam-se [WIN 82] e [CHA 87]. A principal dificuldade dessa abordagem é determinar a ordem das colunas de transistores, pois o número de linhas de difusão necessárias para implementar um circuito é diretamente dependente da ordem das colunas. Em [UEH 81] mostrou-se ser possível implementar uma célula utilizando uma única linha de difusão, se, e somente se, existirem caminhos de Euler através dos planos P e N da célula e esses caminhos passarem pela mesma seqüência de sinais de entrada.

2.6 Linear Matrix

O estilo de leiaute *linear matrix* foi proposto por [UEH 81] para a síntese do leiaute de células CMOS. A principal diferença desse estilo para o estilo gate matrix está na colocação dos transistores. Nesse estilo apenas um transistor P e um transistor N são dispostos em uma coluna de polissilício e as células são geradas em duas linhas horizontais de difusão. Os transistores vizinhos que compartilham o mesmo sinal são unidos na linha de difusão e a geração tenta diminuir as descontinuidades no intuito de melhorar o desempenho elétrico das células.

De acordo com [UEH 81], as principais características do estilo linear matrix são: (1) células estáticas, (2) conexões série/paralelo, (3) planos N e P duais, (4) duas linhas de difusão para a implementação dos transistores e (5) transistores com o mesmo sinal de gate alinhados. A topologia das células linear, como mostrado na Figura 2.7 matrix decompõem-se em cinco regiões horizontais e foi comumente usada em [CHE 89] [MAZ 91] [HSI 91] [LIN 92]:

- duas linhas de difusão para a disposição dos transistores verticalmente, compartilhando a linha de difusão;
- duas áreas para a realização do roteamento de nós internos das células (canais de roteamento P e N), dipostas entre as linhas de alimentação e as linhas de difusão ;
- uma região no centro das células para a realização do roteamento entre as células.

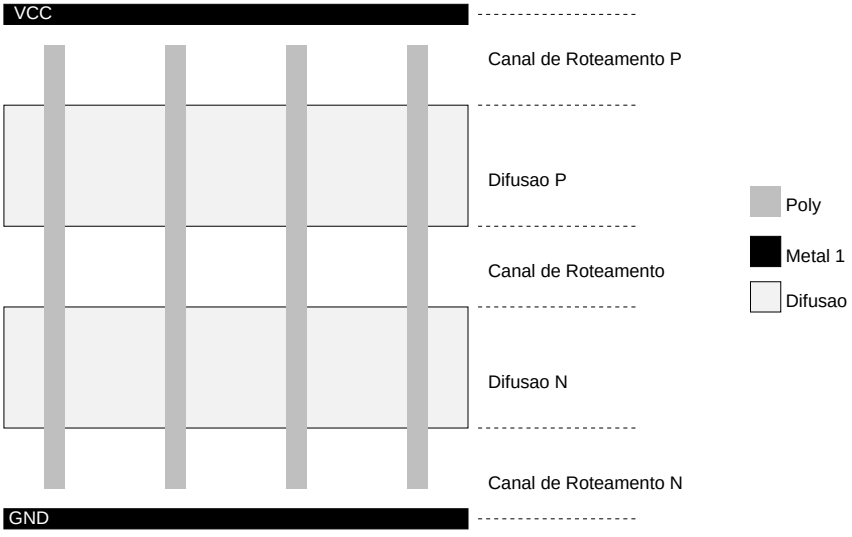

FIGURA 2.7 – Topologia do leiaute de uma célula linear matrix [MAZ 91].

Ainda, é possível minimizar a largura de uma célula conectando os transistores adjacentes pela sobreposição das áreas de dreno e fonte desses transistores. Em [UEH 81], é apresentado um algoritmo heurístico, para a pesquisa de caminhos de Euler nos grafos que representam as conexões de cada células. Como mostrado por [UEH 81], caso seja possível determinar um caminho de Euler para o plano P e outro caminho de Euler para o plano N para uma determinada célula, então é possível gerar esta célula utilizando apenas uma única linha de difusão para cada plano e essa linha de difusão não apresentará *gaps* (espaços vazios), sendo que todos os transistores da célula compartilharão suas áreas de dreno e fonte com os transistores vizinhos.

Diversas otimizações foram propostas para o algoritmo apresentado em [UEH 81]. Maziasz em [MAZ 87], propõe um algoritmo recursivo para a pesquisa de caminhos nos grafos das células. Esse algoritmo permite encontrar a solução ótima em largura para as células que obedecem as restrições de [UEH 81]. Outros algoritmos que permitem alcançar a solução ótima em largura também são propostos em [WIM 87] [WAN 89] [WAN 90].

Quando uma célula não possui o caminho de Euler nos dois grafos da célula, ainda é possível realizar algumas modificações na topologia das células. Estudos da influência da mudança da ordem dos transistores são apresentados em [CHE 88] [LEF 89] [MAD 89]. O algoritmo de [MAD 89] aceita a determinação de restrições para a posição dos pinos de entrada e saídas das células, permitindo assim, que as células geradas possam ser utilizadas em ambientes *standard-cells*.

Aumentando a abrangência do estilo linear matrix, [CHE 88a] e [CHE 89] apresentam algoritmos capazes de realizar a síntese de lógica dinâmica. [CHE 88a] apresenta uma técnica de síntese para circuitos que possuam transistores N em um número bem maior do que transistores P. A técnica consiste em colocar a metade dos transistores N junto a massa e a outra metade junto a linha de alimentação. A abordagem de [CHE 88a] faz uso das ferramentas de disposição das células e roteamento de sistemas *standard-cells* e apresenta problemas durante a fase de roteamento, resultado em espaços vazios nas células. As bandas geradas possuem altura variável e igual a altura da célula mais complexa existente na banda e os espaços vazios decorrentes da implementação de células mais simples não são utilizados.

No estilo linear matrix também existe o problema de realizar a minimização da altura necessária para a concepção de uma célula. Alguns algoritmos tentam reduzir a altura das células pela diminuição na densidade de conexões existentes no canal de roteamento, como o algoritmo apresentado em [ONG 89]. Em [MAZ 91] e [NAK 92] algoritmos ótimo para resolver, conjuntamente, os problemas de minimização da largura e da altura

com reordenamento do transistores são apresentados.

Todos os algoritmos apenas tratavam células complementares e portas de transmissão. Entretanto, em [CAR 92] generaliza o problema da síntese para as células planares (Figura 2.8), onde não é possível a decomposição da célula em redes série/paralelo.

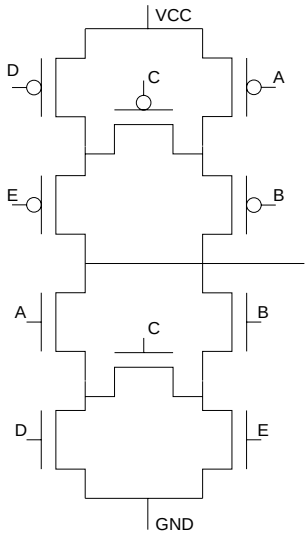

FIGURA 2.8 – Células complexa sem conexões série/paralelo [MOR 94].

Para o estilo linear matrix, também foram proposto algoritmos que tratam a minimização de altura, largura e canal de roteamento em células contendo mais de uma banda. Em [APT 87] é proposto um sistema em diversas bandas, onde as principais características são: (1) altura da banda igual a altura da célula mais complexa, (2) canal de roteamento entre bandas, (3) as células geradas são transparentes a o nível de metal-2, onde são realizados os roteamentos verticais e (4) os planos P e N das células são armazenados em uma biblioteca.

Outro sistema de várias bandas apresentado em [LAK 90], apresenta uma topologia com 3 canais de roteamento e o algoritmo tenta realizar o alinhamento de transistores que possuam o mesmo sinal de *gate* e estão dispostos em banda diferente, para minimizar o roteamento através do uso do polissilício para o roteamento vertical. Entretanto, como em outros métodos para diversas bandas o autor admite que sua abordagem não permite a geração de circuitos contendo mais do que uma centena de transistores.

Em [HEE 92] uma outra topologia para diversas bandas foi desenvolvida, sem canais de roteamento entre as bandas, com bandas de alturas variáveis e o uso do segundo nível de metal para o roteamento vertical. Nesse sistema também o autor admite o limite de cada módulo poder conter até 200 transistores e, como em outros métodos, ferramentas de posicionamento e roteamento baseados nos modelos *standard-cells* são utilizadas. Também em [GUP 2000] é apresentada uma técnica para geração de células 2D (em ape-

nas duas bandas) utilizando uma algoritmo ótimo para minimização da altura e largura das bandas, entretanto, de acordo com o próprio autor, a sua abordagem é prática para circuitos com até 20 transistores.

Em [WAN 93] é apresenta uma otimização para a topologia do leiaute linear matrix. As principais mudanças na topologia são: (1) o desenho das linhas de alimentação no centro das células, (2) as regiões de roteamento da células são dispostas na parte superior e inferior da banda e (3) a conexão entre o plano P e N das saídas da células é realizada verticalmente por linhas em metal-2. A Figura 2.9 mostra a nova topologia proposta por [WAN 93].

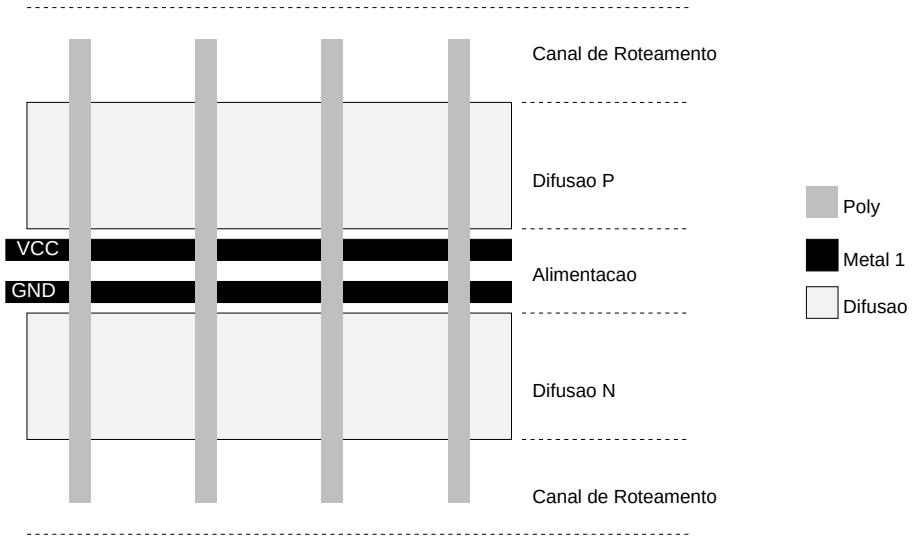

FIGURA 2.9 – Topologia linear matrix proposta por [WAN 93].

Nessa nova topologia, a disposição das linhas de alimentação no centro da célula, faz com que todas as células de uma mesma banda possam ser justapostas e permite que essas células tenham alturas diferentes, pois o alinhamento das células é realizado pelo centro horizontal da banda. Assim sendo, pode-se inclusive gerar transistores com larguras diferentes para atender as requisições elétricas impostas ao projeto.

O uso da topologia proposta em [WAN 93], também conduz ao aparecimento de um canal de roteamento com forma não retangular como mostrado na Figura 2.10. Porém, algoritmos de roteamento que realizam o roteamento sobre as células podem ser usados para permitir a redução na altura do canal e melhorar o desempenho elétrico do circuito.

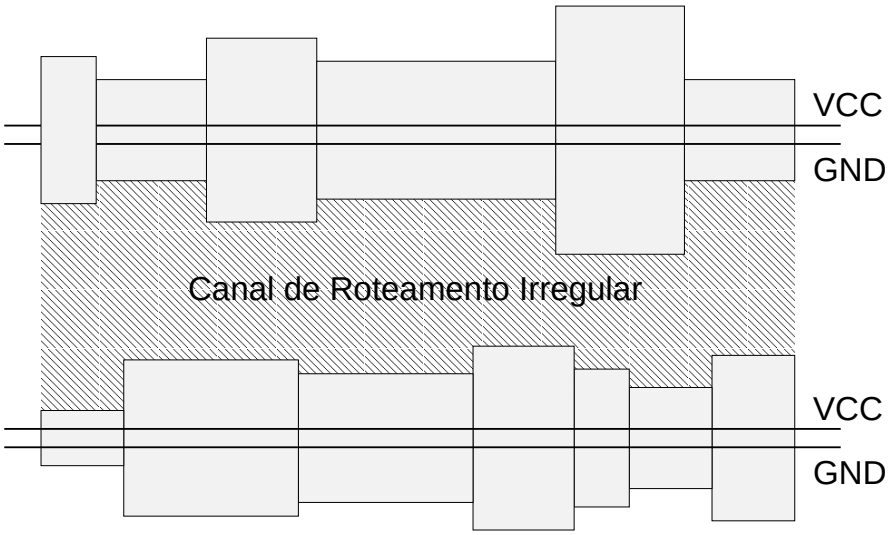

FIGURA 2.10 – Canal de roteamento para as células linear matrix com alimentação no centro da banda.

3 A Ferramenta TROPIC

Este capítulo apresenta a ferramenta TROPIC3 [MOR 2000] (TROPIC na sua versão 3), mostrando a topologia dos leiautes gerados e os passos da geração da síntese. Este estudo detalhado foi realizado, pois o capítulo 4 discute as otimizações realizadas no leiaute e como essas alterações foram inseridas no sistema. Para tanto, um largo conhecimento da ferramenta, suas estruturas de dados e algoritmos envolvidos foi exigido para viabilizar as alterações. Este capítulo é uma compilação dos textos apresentados em [REI 2000] [REI 2000a] [MOR 94].

O TROPIC3 é uma ferramenta para a síntese automática de leiaute de circuitos integrados. Como o TROPIC3 faz uso de bibliotecas virtuais para a concepção dos circuitos, permite a síntese de qualquer célula complexa. O método de síntese empregado é o de síntese de macro-células, ou seja, todas as células do circuito são sintetizadas como um único bloco.

O TROPIC3 usa apenas dois arquivos de entrada para a síntese do leiaute. Um arquivo com a descrição do circuito a ser gerado (transistores e conexões em formato SPICE, podendo ser este arquivo hierárquico ou não. O segundo arquivo é uma descrição das regras de projeto, contendo as restrições e tamanhos mínimos para as camadas e as constantes elétricas que são utilizadas para a extração elétrica do circuito.

O resultado da síntese pode fornecer diversos arquivos, mas sempre será produzido um arquivo em formato CIF com o leiaute do circuito gerado. Caso seja solicita a extração do circuito, um arquivo adicional é produzido contendo as capacitâncias parasitas de roteamento (inclusive as de acoplamento entre condutores vizinhos [KLE 2000] [KLE 2000a]).

As versões anteriores do TROPIC utilizavam grafos de restrições para realizar a compactação do leiaute, o que necessitava de um considerável tempo de CPU para realização. Essa compactação foi removida na versão 3 do TROPIC e a geração apenas é baseada em uma grade virtual. Esta simplificação resulta em leiautes menos denso, porém o uso de 3 níveis de roteamento conduz a uma redução no espaço de roteamento, compensando a perda apresentada pela nova técnica. A Tabela 3.1 compara a densidade do TROPIC3 com as versões anteriores e o LAS [CAD 91], mostrando o ganho em densidade de integração.

Logo, a vantagem obtida na simplificação no processo de compactação é a diminuição significativa no tempo CPU necessário para a síntese. Desta forma, o

TABELA 3.1 – Comparação de densidade de integração entre ferramentas de síntese de leiaute que utilizam compactação baseada em grafos de restrições e o TROPIC3. Tecnologia 0,7μm [REI 2000a].

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

TROPIC3 permite que diversas interações no processo de síntese possam ser realizadas. A Tabela 3.2 mostra o resultado desta nova abordagem. Por exemplo, o circuito C7552 (circuito *benchmark*) com cerca de 14000 transistores pode ser sintetizado em menos de 6 minutos, o que não ocorria em versões anteriores.

Assim sendo, as etapas de síntese lógica - etapa anterior do processo de síntese automática, como proposto na Figura 1.2 - e a etapa de síntese física podem ser repetidas para melhorar a qualidade dos circuitos gerados, pois as novas interações podem ser realizadas levando em conta os dados obtidos de uma síntese preliminar. Desta forma, pode-se, a partir de um *netlist* contendo elementos parasitas extraído com a ferramenta TROPIC3, fazer a inserção de *buffers* em linhas de roteamento longas, dimensionar os transistores adequadamente e obter uma estimativa de desempenho e consumo mais precisa com ferramentas de análise de *timing* e de consumo.

A Tabela 3.3 mostra as máximas densidades - densidades ótimas - que se podem obter sem roteamento, utilizando roteamento com 2 níveis de metal e com 3 níveis de metal. Considera-se a área de roteamento proporcional a área de difusão do circuito, provocando uma redução de 50% na densidade dos circuitos quando se utiliza roteamento em 2 níveis de metal e redução de 25% quando se utiliza roteamento em 3 níveis de metal.

A média obtida através da Tabela 3.2 é de aproximadamente 60000 transistores/mm^2, ficando 64% da densidade máxima esperada para a tecnologia 0.25μm. Em [REI 2000a] este percentual é atribuído a utilização de canais de roteamento para a geração do leiaute pelo TROPIC3.

Frente a esses resultados, a ferramenta TROPIC3 mostrou-se adequada e eficiente para a sua inserção ao projeto FUCAS. Assim sendo, partiu-se dessa ferramenta para a

TABELA 3.2 – Densidade e tempo de CPU para a síntese de diversos benchmarks, utilizando-se a ferramenta TROPIC3, tecnologia 0.25μm, transistores dimensionados com w=2μm, l=0.25vμm, máquina Ultra-Sparc 10 [REI 2000a].

Circuito	xtors	Redes	Bandas	Área (mm^2)	Densidade (tr/mm^2)	CPU(ms) Geração	CPU (ms) Extração
adder	28	13	1	0.00028	100676	50	170
addergate	40	15	2	0.00042	94482	180	310
alu	260	94	3	0.00353	73677	440	3970
alugate	432	117	4	0.00535	80808	500	6050
rip	448	163	4	0.00500	89552	720	8360
cla	528	215	4	0.00700	75439	660	9010
hdb3	570	191	6	0.00745	76539	420	9810
5xp1	798	308	7	0.01087	73386	790	15310
sao2	930	361	7	0.01321	70388	1030	17770
mult6	972	308	6	0.01311	74147	1230	15800
9sym	1092	420	8	0.01547	70575	1170	22790
c499	1556	511	7	0.02279	68277	1730	28820
c1355	2244	647	9	0.02975	75437	2390	41460
c1908	3146	990	14	0.04764	66036	3850	63680
mult2	4512	1239	13	0.07425	60768	5610	66220
c2670	4976	1762	15	0.08408	59181	6360	98230
c7552 3x3	6164	2101	15	0.13295	46363	14730	101100
c3540	7154	2359	15	0.13449	53193	9100	118810
mult12	8584	2455	16	0.14757	58169	15110	133850
c6288	10112	2706	18	0.14931	67726	11030	162970
c5315	10656	3429	15	0.24822	42930	24360	167510
c7552	14376	4764	17	0.33096	43437	109700	228220

TABELA 3.3 – Densidades máximas que podem ser obtidas com o estilo de implementação adotado por TROPIC3.

Processo (μm)	Densidade Máxima (tr/mm^2)		
		com roteamento	
	sem roteamento	2 níveis de metal	3 níveis de metal
l=0.8	8296	4148	6222
l=0.6	14748	7374	11061
l=0.5	31332	15666	23499
l=0.35	62644	31322	46983
l=0.25	125328	62664	93996
l=0.18	250656	125328	187992

realização de otimizações que pudessem conduzir a uma melhor densidade de integração, reduzindo a distância entre o TROPIC3 e a densidade máxima esperada, bem como a sua

inserção em um ambiente integrado de projeto.

3.1 A Topologia do Leiaute

A ferramenta TROPIC3 utiliza o estilo de leiaute linear matrix, similar a versão proposta por [HWA 91]. As principais características da topologia são:

- estilo linear matrix com duas linhas de difusão e a direção da largura dos transistores e ortogonal as linhas de difusão;
- o roteamento é implementado em três níveis de metal e com a utilização de contatos empilhados, diminuindo a área de roteamento;
- sem compactação de leiaute. Essa característica é muito importante, pois permite a geração rápida do leiaute. Ferramentas, como o LAS [CAD 91] por exemplo, criam descrições simbólicas intermediárias do leiaute e requerem um passo adicional de compactação, que, normalmente, exigem uma quantidade significativa de tempo de CPU;
- completo cálculo das capacitâncias e resistências parasitas do circuito;
- arquivo para a descrição de tecnologia simples, contendo apenas 28 regras de desenho e 38 regras elétricas.

Como características principais do leiaute da células, pode-se citar:

- conexões entre os planos P e N (linhas de difusão) para compor as saídas das células realizada diretamente em metal1;
- linhas de difusão P e N possuem separação mínima;
- linhas de alimentação (VCC e GND) no centro da banda e realizadas em metal2;
- roteamento sobre as células (OTC - *over-the-cell routing*) para conectar as nós internos de portas CMOS complexas e redes que pertençam a um único canal;
- apenas gates com conexão no canal de roteamento ficam alinhados pela grade de roteamento, demais transistores não possuem restrição de alinhamento e são conectados com *jogs* em polissilício ao seu dual.

A Figura 3.1 mostra o leiaute de um conjunto de células gerados utilizando o TROPIC3 onde é possível a identificação das principais características da síntese.

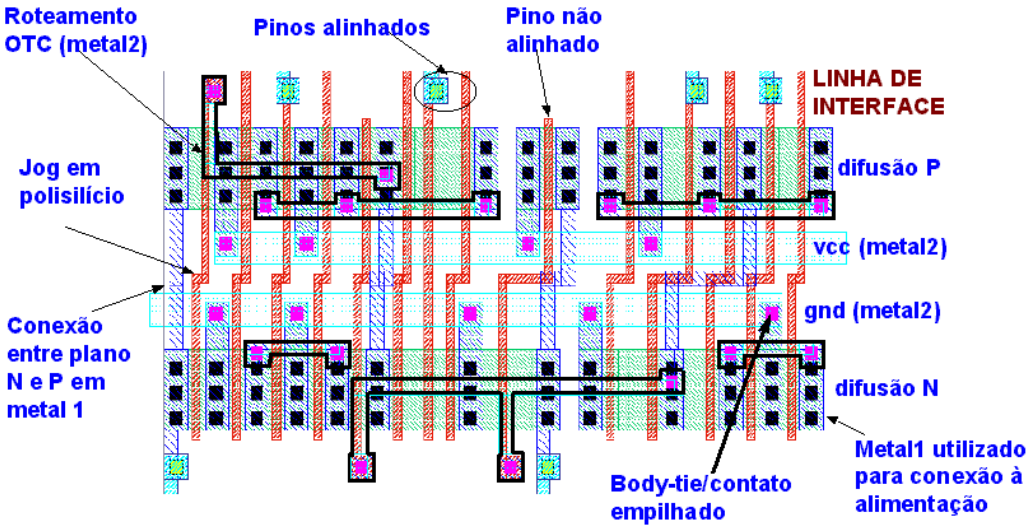

FIGURA 3.1 – Exemplo de leiaute TROPIC3 [REI 2000a].

3.2 As Etapas de Geração

Essa seção apresenta os passos realizados pela ferramenta TROPIC3 para a geração do leiaute. É importante conhecer as etapas de geração, pois apenas algumas dessas etapas foram alteradas na ferramenta WTROPIC. A Figura 3.2 mostra um fluxograma das etapas do TROPIC3.

A primeira etapa é a **leitura dos arquivos** de entrada necessários para síntese: o arquivo do circuito descrito em um *netlist* SPICE e o arquivo de regras de projeto. Realizada a leitura do circuito, uma **planificação** do circuito é feita para obter-se uma lista de transistores sem hierarquia. A etapa de planificação é realizada sempre, sendo o circuito descrito hierarquicamente ou não.

Tendo-se uma descrição plana do circuito, então realiza-se a **extração das células folhas**. Esta etapa consiste em obter uma lista células que possuam *n* entradas e uma única saída. A saída é obtida a partir de um nó comum entre um transistor P e N. Partindo-se desse nó, faz-se a redução de todos os transistores P e N, até obter um caminho para VCC e GND, respectivamente, e os pares de transistores.

Se após a extração das células folhas ainda sobrarem transistores não reduzidos, e caso estes transistores não sejam identificados como portas de transmissão (transistores P e N em paralelo), então é porque existem células não complementares. Nesse caso, o processo de geração é abortado, pois o TROPIC3 só sintetiza células duais. Obviamente, se o circuito conter, por exemplo, portas *ands*, *or* ou *flip-flops*, estas portas são quebradas e extraídas como um conjunto de outras células folhas.

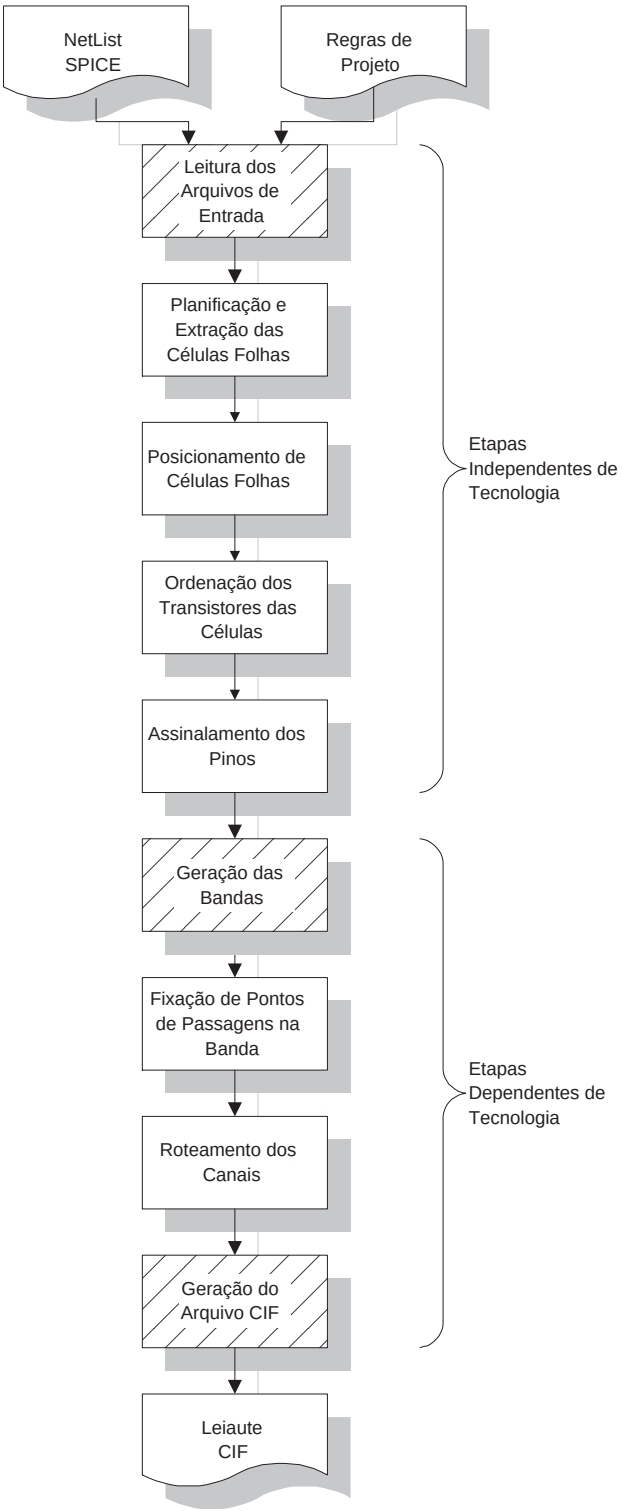

FIGURA 3.2 – Etapas de Geração do TROPIC3.

Com o término da extração de células, pode-se começar duas etapas: o posicionamento das células e a ordenação dos transistores dentro de cada célula. Para o **posicionamento das células**, usá-se um algoritmo determinístico baseado em *min-cut* - corte mínimo. O algoritmo divide a área de um circuito em quadrantes, realizando divisões

verticais e horizontais, sucessivamente. No caso do TROPIC3, as divisões horizontais respeitam o número de bandas definido pelo usuário e caso o usuário não especifique o número de bandas, o TROPIC3 define um número para obter uma relação de aspecto - altura por largura do circuito - mais próximo de 1 possível.

Em cada quadrante, utiliza-se um algoritmo baseado na conectividade das células para definir a ordem das células dentro do quadrante. Para o posicionamento dos quadrantes, adicionalmente ao algoritmo de quadratura, utiliza-se propagação de pinos para melhorar a solução de roteamento. Para propagação de pinos, o particionamento de um quadrante leva em consideração os outros quadrantes já posicionados definindo para cada quadrante os sinais de interface como forças de atração de células. Dessa maneira, uma determinada célula é *atraída* para um determinado quadrante, onde sua posição minimiza o comprimento total das conexões.

Paralelamente ao posicionamento, realiza-se a **ordenação dos transistores** em cada células folha. A partir da lista dos transistores de cada células e suas conexões, faz-se a busca de caminhos de Euler nos planos P e N da célula. A existência de caminhos de Euler, como mostrado em [CAR 92], indica que o célula pode ser sintetizada utilizando uma única linha de difusão para cada plano e, obviamente, sem a inserção de *gaps*.

Após obtido a lista de caminhos nos planos P e N, realiza-se a intersecção desses dois conjuntos para obter uma solução de caminho que possua a mesma ordem no transistores. Caso mais de uma solução seja encontrada, a escolha é feita tentando-se minimizar o comprimento da célula, o roteamento interno e tentando-se maximizar o número de contatos.

Caso não exista uma solução comum para os planos P e N, mas existam caminhos de Euler nos planos, o TROPIC3 escolhe entre todas as soluções nos dois planos, uma solução em um plano que inserirá o menor número de *gaps* no outro plano.

Também pode não ser encontrado nenhum caminho de Euler em nenhum dos planos. Nesse caso o reordenamento dos transistores - mudança nas conexões entre os transistores sem alterar a função lógica implementada pela célula - poderia conduzir a uma solução com caminho de Euler. Entretanto, o TROPIC3 não possui esta otimização e a solução é encontrada pelo concatenamento de soluções parciais. A ferramenta notifica ao usuário todas as células em que não foi possível obter o caminho de Euler, permitindo ao usuário a realização de um reordenamento manual, caso ele julgue necessário.

O **assinalamento dos pinos** só começa após completado o posicionamento das células nas bandas do circuito e dos transistores dentro de cada célula. Agora já é possível saber a ordem dos transistores dentro de cada banda. A tarefa do assinalamento consiste

em definir que nós da célula terão que possuir contatos na região de interface da banda para serem roteados no canal. Nós internos a uma célula ou internos a uma determinada banda são marcados como *mortos*, já que seu roteamento pode ser realizado sobre a própria banda (OTC - *Over-The-Cell*) e não necessita de canais de roteamento. Como cada conexão pode ser realizada pelo canal de roteamento superior ou inferior de uma banda, pois cada conexão pode acessar os dois canais, a escolha de canal será utilizado é feita minimizando-se o comprimento total da rede por todo o circuito, levando-se em consideração todos os nós que deverão ser conectados a essa rede.

Todas as etapas da geração até esse ponto são independentes de tecnologia, e, diferentemente das versões anteriores do TROPIC, não utilizam descrições simbólicas do circuito, simplesmente uma estrutura de dados interna a ferramenta. A partir desse ponto, as regras de projeto obtidas inicialmente serão usadas para a geração do leiaute.

Tendo-se concluído o posicionamento das células e o assinalamento dos pinos, inicia-se a geração de cada banda. Cada banda é percorrida da esquerda para a direita, passando por todas as células da banda e para cada célula da banda, por todos os pares de transistores e para cada par de transistores pelo transistor N e P. O algoritmo básico implementado pelo TROPIC3, obtido em [REI 2000a], é mostrado na Figura 3.3.

para todas bandas **r** *do circuito*
para todas as celulas **c** *da banda r*
para todos os transistores **t** *da celula c*
para os planos **n** *e* **p** *de t* {

1. *determinar a coordenada em microns do dreno t referente ao plano n ou p, levando-se em conta se o dreno deve ser alinhado ou não a grade virtual.Se for alinhado, fixar a coordenada geométrica com a primeira posição livre da grade disponível. Se não for alinhado, utilizar regra de distancia mínima em relação ao gate anterior. Caso a coordenada sobreponha-se a outra rede, avançar a coordenada do dreno (caso de colisão em meta1 com saída de célula anterior);*
2. *determinar a coordenada em microns do gate t referente ao plano n ou p. Mesmas restrições que o dreno, tomado-se o cuidado de não se colocar linhas de polissilício em curto-circuito;*
3. *determinar a coordenada em microns do fonte t referente ao plano n ou p, segundo as mesmas restrições.*

}

FIGURA 3.3 – Algoritmo para geração das bandas [REI 2000a].

A Figura 3.4 apresenta uma porção de um leiaute de uma banda gerada pela ferramenta TROPIC3. Observa-se neste leiaute, que a inserção de *gaps* na linhas de difusão ou

o alinhamento de pinos pela grade para serem conectados a linha de interface, produzem o aumento da área de difusão. Esta etapa não produz o arquivo CIF, mas determina a coordenada real de cada dreno, fonte e gate dentro de sua banda

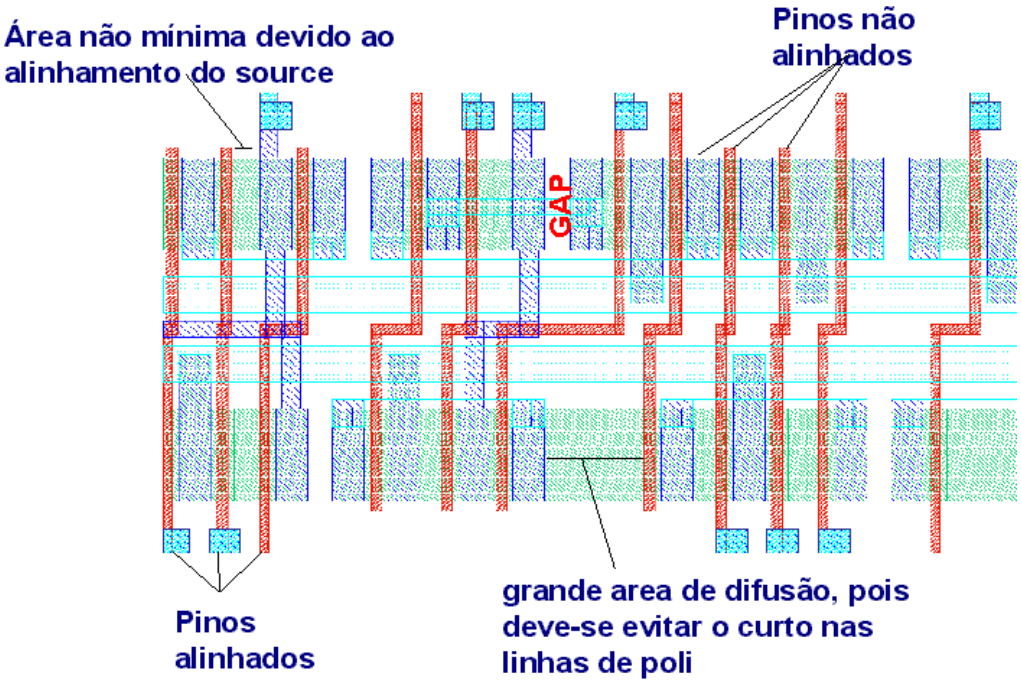

FIGURA 3.4 – Exemplo de leiaute de uma banda [REI 2000a].

A etapa de **fixação de pontos de passagem na banda** consiste em determinar pontos sobre a banda onde *feedthroughs* devem ser inseridos para conectar bandas não adjacentes.

Determinado-se a posição de todas as redes na região de interface, parte-se para o roteamento dos canais. O **roteamento** do TROPIC3 utiliza um algoritmo *left-edge* [HAS 71], onde uma linha de roteamento é inserida e todas as redes possíveis, da esquerda para a direita, são roteadas sobre esta linha. Caso ainda restem redes não roteadas, uma nova linha de roteamento é inserida e, novamente tomadas da esquerda para a direita, as redes restantes são roteadas. Esse passo é repetido até que não restem mais redes no canal a serem conectadas.

O TROPIC3 utiliza dois canais de roteamento sobrepostos para reduzir a área de conexões. O canal inferior utiliza as linhas de polissilício na vertical e linhas de metal-1 na horizontal. O canal superior utiliza linhas de metal-2 na horizontal e linhas de metal-3 na vertical. A alocação pode ser facilmente alterada no roteador, para evitar o efeito das capacitâncias de acomplamento entre as linhas de metal-1 e metal-2 realizadas em paralelo, entretanto a mudança ocasionaria a inserção de vias em todos os *feedthroughs*, já que a posição desses na banda é vertical.

A última etapa, consiste em realmente **gerar o arquivo CIF**. Esse passo percorre as estruturas internas de armazenamento da ferramenta TROPIC3, percorrendo o circuito da parte inferior a superior (canal de roteamento - banda - canal de roteamento) e da esquerda para a direita em cada banda, e a com base nas informações geradas nas etapas anteriores desenha-se os polígonos do circuito. Ao final, um pente de alimentação é desenhado.

As etapas de leitura dos arquivos de entrada, geração das bandas e geração do arquivo CIF sofreram alterações para obter um aumento na densidade dos transistores gerados. Essas alterações, bem como os resultados obtidos com essas alterações são mostrados no Capítulo 4.

4 Topologia para Alta Densidade de Leiaute

Como visto no capitulo 3, a técnica de geração do leiaute pela ferramenta TROPIC3 não faz uso de descrições simbólicas, e não necessita de um passo de compactação. Tal característica, aliada ao processo de geração das bandas, conduz a ferramenta a obter resultados satisfatórios em termos de tempo de CPU, mas à priori resultados não tão evidentes quanto a densidade de integração. Porém, a utilização de três níveis de metais e contatos empilhados entre níveis não adjacentes para a realização do roteamento, torna os resultados obtidos pela ferramenta também satisfatórios.

Apesar desses resultados, o estudo da topologia dos leiautes gerados para diversos circuitos utilizando o TROPIC3 mostra que a forma de utilização das linhas de difusão, dependendo da estrutura dos circuitos, apresenta significativa área de dreno e/ou fonte para os transistores sintetizados. Como o TROPIC3 utiliza o estilo *linear-matrix*, canais de roteamento interbanda, o aumento da área de dreno dos transistores implica em aumento da largura das bandas geradas.

Este capítulo apresenta as otimizações realizadas nos algoritmos de geração de leiaute para permitir um melhor uso das linhas de difusão, minimizando assim, as capacitâncias parasitas do circuito e aumentando a densidade dos leiautes sintetizados através da ferramenta e o resultado dessas otimizações.

4.1 Otimização 1 - Jogs com Alturas Diferentes

Esta otimização tem como objetivo a diminuição das áreas de difusão pela implementação de diversos *jogs* com alturas diferentes. Normalmente, em ferramentas para a geração de leiautes, a utilização de cada uma das camadas de polissilício, metal-1, metal-2 e metal-3 é sempre realizada em uma mesma direção para simplificar o algoritmo roteamento. A união de duas linhas paralelas na mesma camada é realizada através de uma linha ortogonal em outra camada, evitando a criação de restrições para o roteamento em uma determinada camada. Entretanto, esta técnica adiciona um número considerável de contatos e vias, pois em diversos casos duas linhas paralelas poderiam ser unidas por uma linha ortogonal na mesma camada. Um *jog* é a inserção de uma linha ortogonal para unir duas linhas paralelas, evitando o uso de outra camada.

Na topologia TROPIC3, todos os *jogs* de polissilício, causados pelo desalinhamento dos transistores P e N, são inseridos no centro das células como mostrado nas Figuras 3.1

e 3.4. Como a banda é gerada da esquerda para a direita, assume-se como coordenada (0,0) da banda o canto inferior esquerdo e os valores crescem no sentido direita/topo. Logo, para impedir curtos-circuitos em polissilício, o processo de geração da banda impõe a seguinte restrição: as coordenadas $X_N^{xtor(I)}$ e $X_P^{xtor(I)}$ devem ser maiores que a maior das coordenadas $X_N^{xtor(I-1)}$ e $X_P^{xtor(I-1)}$ acrescida da largura das linhas de polissilício e da distância mínima entre linhas de polissilício (regra de desenho). X_N^{xtorI} significa a coordenada esquerda do transistor I do plano N da banda que está sendo gerada.

O cumprimento dessa restrição no TROPIC3 é essencial, pois desta forma, fica garantido que nunca haverá a sobreposição de linhas de polissilício, já que os transistores do plano P e N só começam a serem desenhados após os dois transistores N e P anteriores serem completados. A Figura 4.1 mostra o resultado para a geração de uma porta NAND de três entradas com a restrição para a colocação dos transistores. Os transistores P estão em paralelo e exigem a colocação de contatos na região de difusão, fazendo os transistores P serem gerados com um espaçamento maior do que os transistores N. Desta maneira, o segundo e o terceiro par de transistores ficam desalinhados e são gerados com *jogs*, mas a restrição imposta, faz com que o último transistor N seja gerado com um espaçamento maior que o mínimo, criando áreas de difusão não mínimas.

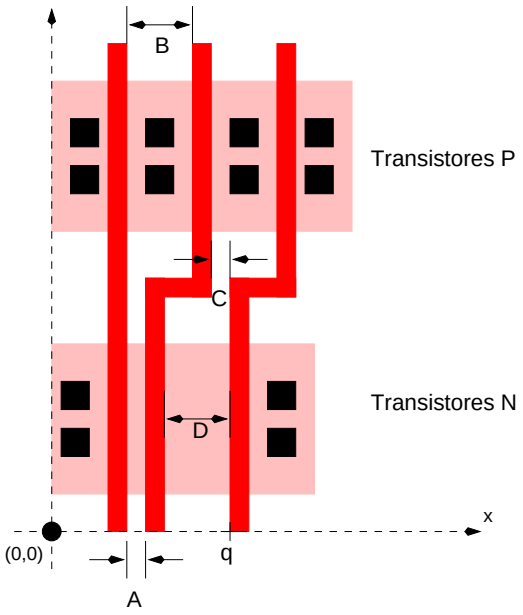

FIGURA 4.1 – Regra de geração dos transistores no TROPIC3: (A) distância mínima entre transistores, (B) distância mínima entre transistores com contanto de dreno/fonte, (C) distância mínima entre transistores imposta pelo TROPIC3, (D) área de difusão não mínima.

É possível perceber na Figura 4.1 a possibilidade de manter a distância mínima entre os transistores, caso os *jogs* não sejam gerados todos no centro da célula. A Figura 4.2 mostra o leiaute da mesma porta NAND de três entradas usando os *jogs* em polissilício com alturas diferentes. O desalinhamento dos *jogs* permitiu o desenho de todos os

transistores com distância mínima.

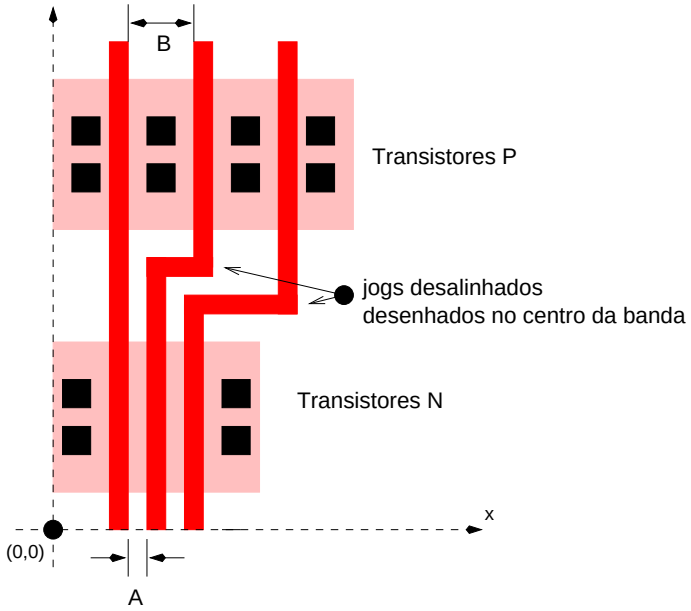

FIGURA 4.2 – Transistores desenhados com *jogs* desalinhados.(A) distância mínima entre transistores, (B) distância mínima entre transistores com contanto de dreno/fonte.

Adicionalmente, a utilização de espaçamento mínimo entre as linhas de difusão P e N permite realizar a inserção de diversas linhas horizontais - ortogonalmente à posição dos transistores - de polissilício entre os dois planos. Essas linhas permitem a implementação de diversos *jogs* em polissilício. Para implementar a inserção de diversas quebras em polissilício o processo de síntese do TROPIC3 sofreu as seguintes alterações:

1. após a leitura do arquivo de regras de desenho, no início da síntese, calcula-se o numero máximo de linhas horizontais possíveis entre as linhas de difusão, determinando o número máximo de quebras a serem inseridas. As linhas são numeradas da inferior a superior como mostrado na Figura 4.3;

2. a estrutura de dados que armazena os sinais dos gates dos transistores P e N foi alterada para armazenar a informação da sua quebra. Isto é, cada sinal sabe em qual linha será implantada a sua quebra e qual o sentido da quebra em relação ao transistor N - esquerda ou direita;

3. na fase de geração das células folha, após a geração de cada par de transistores I determina-se o $X_N^{xtor(I+1)}$ e $X_P^{xtor(I+1)}$ mínimos para o próximo par de transistores e o número da linha da próxima quebra.

   Para a determinação desses valores utilizou-se uma heurística. Observando-se as diversas topologias mostradas em [FRA 98], pode-se perceber que as quebras ocorrem todas num mesmo sentido, até que a possibilidade de inserção de quebras seja

esgotada ou ocorra uma quebra no sentido contrário. A ocorrência de qualquer uma das duas situações implica no reinicio do controle das linhas de quebra e a restrição inicial imposta pelo TROPIC3, como mostrado na Figura 4.1(C) é atendida.

Caso a restrição do TROPIC3 seja atendida antes por um determinado par de transistores N e P que está sendo gerado, por exemplo pela ocorrência de um jog no sentido oposto, também força o reinicio do controle de quebras. Na realização das quebras para a esquerda a alocação das linhas de polissilício é realizada no sentido decrescente das linhas e no caso das quebras para a direita a alocação é realizada no sentido crescente das linhas.

4. a fase de geração do arquivo de leiaute foi alterada para permitir a leitura das quebras armazenadas nos transistores e o desenho de diversas linhas de polissilício.

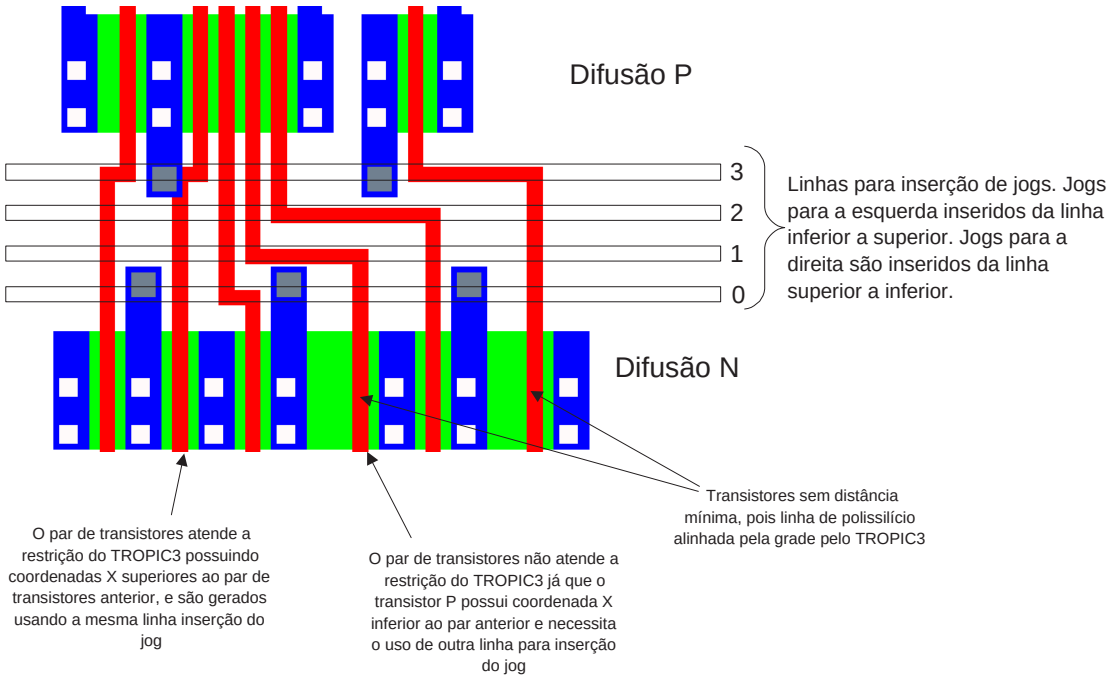

FIGURA 4.3 – Ocupação da diversas linhas de polissilício para a inserção de jogs.

A Figura 4.3 mostra o resultado da implementação das quebras nos dois sentidos. Os dois primeiros pares de transistores estão desalinhados para a direita e os *jogs* de polissilício são implementados utilizando a mesma linha horizontal, pois a existência de contatos de dreno e fonte permite que a restrição imposta pelo TROPIC3, item C da Figura 4.1, seja atendida e não sendo necessário o uso de outra linha. Sempre que um par de transistores gerados estiver todo a direita do par de transistores anterior, atendendo a restrição, é permitida a inserção de um *jog* na linha superior(linha 3 da Figura 4.3) ou inferior(linha 0 da Figura 4.3).Caso o transistor P esteja a esquerda do transistor N,

a linha superior será usada, caso contrário a linha inferior será usada. Assim, enquanto houver linhas é possível a inserção de quebras em polissilício.

Na Figura 4.3, os quatro outros pares de transistores estão desalinhados para a esquerda e os *jogs* foram inseridos da linha inferior a superior. Como o primeiro deles atende a restrição do TROPIC3, isto é, o par de transistores a ser desenhado possui coordenadas horizontais superiores ao par anterior, permite que o controle recomeçasse pela linha inferior.

Após as alterações terem sido implementadas no TROPIC3, diversos circuitos de *benchmarks* foram sintetizados, como por exemplo o somador da Figura 4.4, para permitir a avaliação da nova topologia. Os resultados dessa avaliação mostraram que não houve melhora, mas sim decréscimo. A Tabela 4.1 apresenta os resultados da comparação contra o TROPIC3 e o aumento da largura do circuito, em média, 4% em relação a versão original. Com o acréscimo da largura do circuito houve uma diminuição na densidade de integração na mesma proporção, pois a altura dos circuitos não se modificou.

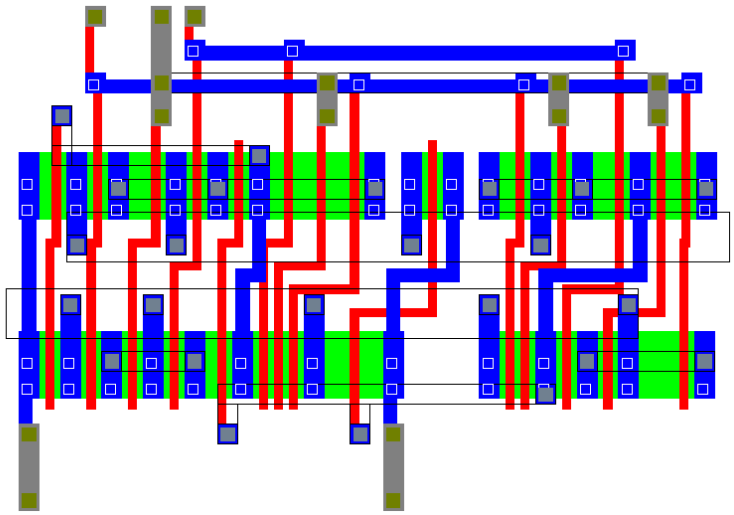

FIGURA 4.4 – Somador sintetizado utilizando a otimização 1.

O problema desta topologia que ocasiona o decréscimo na densidade de transistores ocorre quando o limite de quebras é alcançado e a restrição do TROPIC3 é imposta, ou seja, o par de transistores não pode ser desenhado a menos que os transistores sejam deslocados para a direita para evitar sobreposição das linhas de polissilício. A Figura 4.5 mostra um situação desse tipo. Nestes casos um série de transistores é desenhada com tamanho mínimo e o transistor que não pode ser alocado sofre um grande deslocamento, fazendo que as capacitâncias parasitas que antes estavam distribuídas pelos diversos transistores, agora concentrem-se em um único.

Esse tipo de distribuição faz com que o desempenho elétrico deste último transistor

TABELA 4.1 – Comparação da largura dos circuitos sintetizados com o TROPIC3 e a primeira otimização. Tecnologia 0,25μm.

Circuitos	TROPIC3	Otimização 1	Melhoria
adder	20,45	20,45	0%
addergate	16,80	16,80	0%
alu	74,45	77,45	-4%
alugate	90,00	98,45	-9%
rip	88,70	87,70	1%
cla	111,45	112,45	-1%
mult6	135,70	138,45	-2%
c1355	200,45	223,45	-11%
andre	355,45	355,45	0%
mult2	284,70	296,45	-4%
c5315	636,45	675,25	-6%
c7552	764,00	828,25	-8%
		Média	-4%

fique degradado e, na maioria dos casos, acarreta em maior largura da banda, pois alguns transistores são desenhados alinhados pela grade e não utilizando o tamanho mínimo. Assim, o deslocamento para direita do par de transistores, na maioria dos casos, é maior do que o necessário e maior do que os deslocamentos impostos pelo TROPIC3.

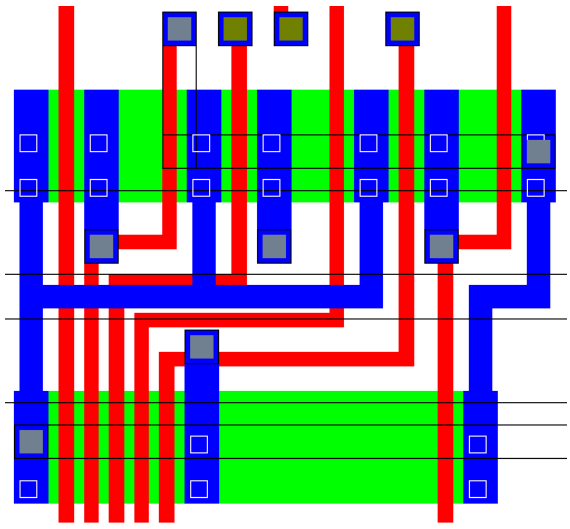

FIGURA 4.5 – Limite de *jogs* alcançado.

4.2 Otimização 2 - Diversos Jogs com Alturas Diferentes em um mesmo Par de Transistores

Essa segunda otimização objetiva sanar o problema causado pela inserção de múltiplos *jogs* para um par de transistores. O caminho natural é continuar otimizando o uso do polissilício entre as linhas de difusão. Dessa maneira, a intenção é permitir que o polissilício de conexão entre o par de transistores possa fazer quantas quebras forem necessárias. A idéia do múltiplos *jogs* é permitir a maior aproximação possível entre transistores adjacentes em um mesmo plano.

Para tanto, as alterações realizadas na otimização 1 foram acrescidas em:

1. adicionou-se na estrutura de gate outras informações, capacitando a estrutura a armazenar os múltiplos *jogs*. Utilizou-se uma lista dinâmica para realizar tal tarefa.
2. na determinação dos $X_N^{xtor(I+1)}$ e $X_P^{xtor(I+1)}$ mínimos para o próximo par de transistores e o número da linha da próxima quebra, adicionou-se o tratamento de procura da primeira coordenada X capaz de permitir a conexão entre o plano N e P em polissilício. Na verdade, o resultado é um algoritmo de roteamento bem especializado e com condições de contorno bem definidas, permitindo assim que esse algoritmo fosse implementado durante a própria geração da banda.
3. alterar novamente a etapa de geração do arquivo CIF para realizar o desenho dos múltiplos *jogs* no mesmo par de transistores P e N.

Novamente para avaliação, os mesmos circuitos *benchmarks* foram sintetizados e os resultados de comparação com o TROPIC3 são mostrados nas Tabelas 4.2 e 4.3.

Como resultado da segunda otimização teve-se uma pequena melhora média na largura dos circuitos. Mesmo utilizando múltiplas quebras em polissilício, não é possível garantir a área mínima de dreno/fonte, e como a largura do circuito representa a largura da maior banda, não significa que houve, nos casos de aumento da largura do circuito, aumento de todas as bandas.

Salienta-se também, pela avaliação da Tabela 4.3 que houve acréscimo na altura do circuito em quase todos os casos. Havendo a diminuição da largura do circuito, diminui-se o espaço de roteamento e aumenta o congestionamento de pinos. Esses fatores levam a uma necessidade de um roteador mais eficiente ao implementado na ferramenta TROPIC3 e utilizados nas otimizações, pois o roteamento ineficiente acaba introduzindo um maior número de linhas no canal de roteamento, ocasionando assim um aumento na altura do circuito.

TABELA 4.2 – Comparação da largura dos circuitos sintetizados com o TROPIC3 e a segunda otimização. Tecnologia 0,25μm.

Circuito	TROPIC3	Otimização 2	Melhoria
adder	20,45	18,45	10%
addergate	16,80	15,00	11%
alu	74,45	72,95	2%
alugate	90,00	93,00	-3%
rip	88,70	82,45	7%
cla	111,45	105,45	5%
mult6	135,70	126,45	7%
c1355	200,45	209,25	-4%
andre	355,45	339,00	5%
mult2	284,70	275,75	3%
c5315	636,45	641,35	-1%
c7552	764,00	793,45	-4%
		Média	3%

TABELA 4.3 – Comparação da altura dos circuitos sintetizados com o TROPIC3 e a segunda otimização. Tecnologia 0,25μm.

Circuito	TROPIC3	Otimização 2	Melhoria
adder	13,60	15,00	-10%
addergate	25,20	28,00	-11%
alu	47,40	50,20	-6%
alugate	59,40	66,00	-11%
rip	56,40	64,00	-13%
cla	62,80	66,30	-6%
mult6	96,60	106,00	-10%
c1355	148,40	159,00	-7%
andre	270,00	265,00	2%
mult2	260,80	279,00	-7%
c5315	390,00	410,00	-5%
c7552	433,20	453,00	-5%
		Média	-7%

4.3 Otimização 3 - WTROPIC: Desalinhamento de Todos os Transistores

Apesar da inserção de múltiplos *jogs* em uma mesma linha de polissilício ter apresentado resultados positivos, ainda não é possível garantir todas as áreas de dreno e fonte mínimas.

Como visto na seção 3.1, o TROPIC3 impõe a restrição de que todo sinal da célula que deva conectar-se ao canal de roteamento esteja alinhado pela grade virtual que será utilizada pela ferramenta de roteamento. Essa imposição, força o deslocamento para a direita, pois a banda é gerada da esquerda para a direita, de sinais que não atendam a restrição. Esses sinais são deslocados para a direita até alinhar-se a grade.

A mesma restrição é imposta aos transistores. Se um transistor a ser conectado ao canal, não está sob a grade, o transistor é deslocado para a direita até ajustar-se. Isso provoca a geração de áreas não mínimas de dreno e fonte. A Figura 4.6(A) mostra como três transistores P seriam gerados pelo TROPIC3. Nesse caso, somente o segundo transistor teria áreas de dreno e fonte mínimas.

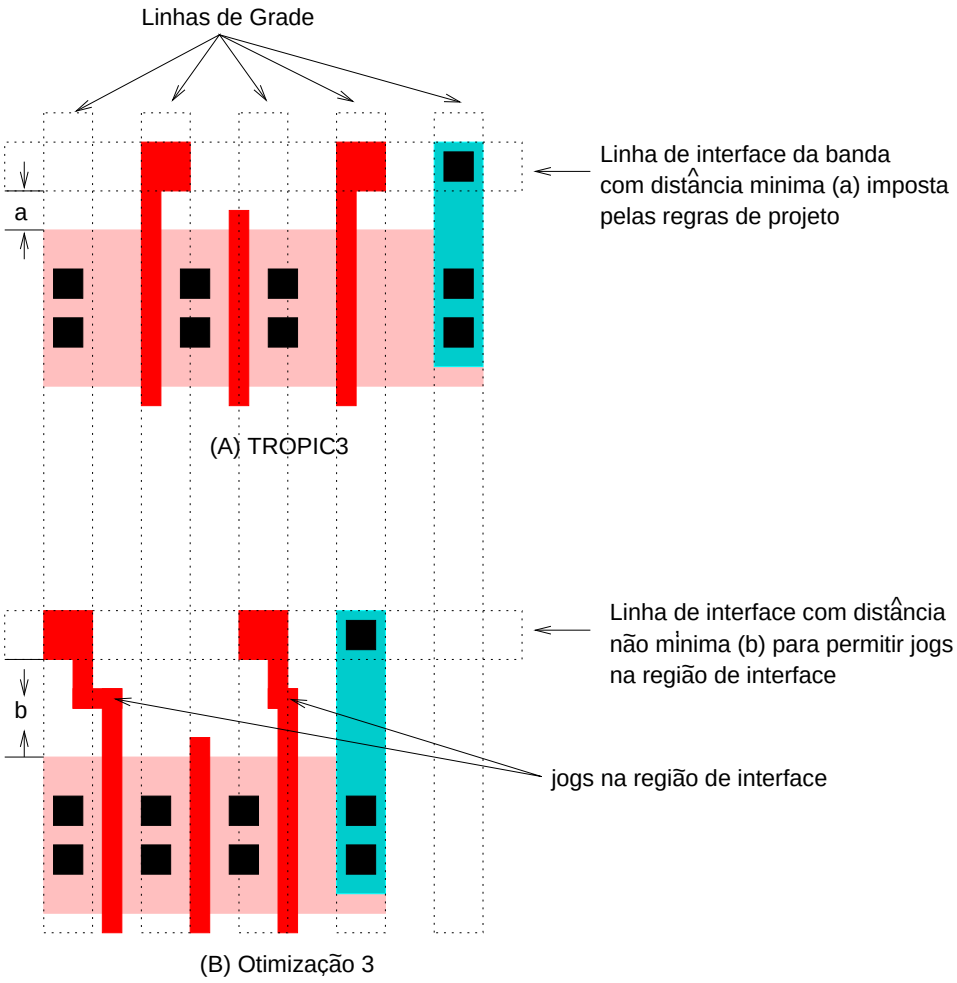

FIGURA 4.6 – Linhas de interface da banda para o (A)TROPIC3 e para a (B) terceira otimização.

Contudo, é possível inserir *jogs* na região de interface da célula para conectar o transistor com a posição da cabeça de contato. E mais, esses *jogs* só ocorreram no sentido do transistor para a esquerda, pois enquanto o transistor não alcançar a próxima coluna da grade, este poderá ser conectado utilizando a coluna anterior. Caso a coluna anterior já

estiver ocupada, a próxima coluna então, será utilizada e o transistor será deslocado para a direita.

O resultado dessa abordagem é que, caso o transistor possa ser desenhado com tamanho mínimo de dreno e este estiver desalinhado da grade, o transistor pode ser construído nesta posição e a sua conexão ao canal de roteamento é realizada por um *jog* na região de interface.

A Figura 4.6(B) mostra a alteração realizada no leiaute da banda para implementar os *jogs* na região de interface da banda. A linha de interface da banda foi distanciada da banda para permitir que *jogs* pudessem ser inseridos nessa região.

A implementação desse método forçou a alteração da rotina de assinalamento da posição das redes na estrutura de armazenamento do canal, para permitir que um sinal desalinhado ou ocupasse a coluna anterior da grade ou alinhasse-se a próxima coluna na grade. Essa simples mudança permitiu a geração de quase a totalidade dos transistores sem grade. Os transistores só não são desenhados com tamanho mínimo de dreno e fonte ou quando não é possível a inserção de mais *jogs* em polissilício ou quando o transistor não estiver alinhado à grade e a coluna anterior da grade estiver sendo utilizada por outro sinal.

Para o caso do transistor não estar alinhado à grade e a coluna anterior da grade estar ocupada, descartou-se a hipótese de criar *jogs* para a direita, que apesar de não deslocar o transistor para a direita, imporia uma restrição a mais para o próximo transistor, o que poderia ocasionar em um deslocamento maior para o próximo transistor.

Para o desenho do *jog* na região de interface foi preciso alterar a etapa de geração do arquivo CIF. Também foi preciso aumentar a distância entre a linha de interface e a banda, para permitir que as linhas de polissilício pudessem ser desenhadas sem violar as regras de desenho.

Essas alterações foram realizadas a partir da otimização 2, isto é, o estilo de leiaute gerado contém a inserção de quebras diversas em polissilício. A Figura 4.7 mostra o circuito somador sintetizado com os transistores desalinhados.

A Tabela 4.4 mostra o resultado da redução da largura dos circuitos *benchmarks* gerados em comparação com o TROPIC3. A diminuição média da largura fica em torno de 11%.

Novamente nessa situação, como na otimização 2, a diminuição da largura dos circuitos fez com que os leiautes produzidos tivessem um aumento de altura, devido ao congestionamento do roteamento e pela pouca eficiência do algoritmo de roteamento im-

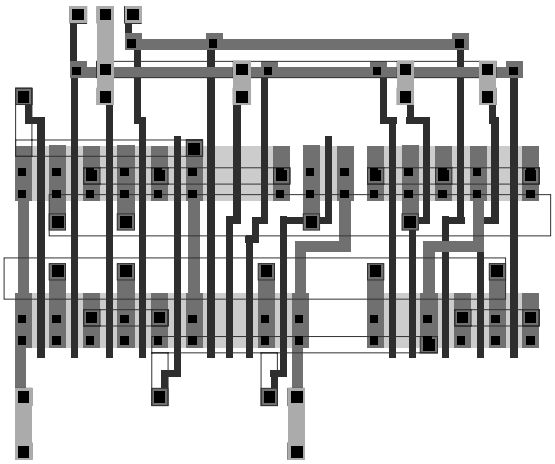

FIGURA 4.7 – Circuito somador sintetizado através da otimização 3.

TABELA 4.4 – Comparação da largura dos circuitos sintetizados com o TROPIC3 e a terceira otimização - WTROPIC. Tecnologia 0,25μm.

Circuitos	TROPIC3	Otimização 3	Melhoria
adder	20,45	18,60	9%
addergate	16,80	14,80	12%
alu	74,45	66,05	11%
alugate	90,00	80,00	11%
rip	88,70	79,15	11%
cla	111,45	101,65	9%
mult6	135,70	120,65	11%
c1355	200,45	175,00	13%
andre	355,45	317,15	11%
mult2	284,70	252,75	11%
c5315	636,45	571,00	10%
c7552	764,00	679,10	11%
		Média	11%

plementado no TROPIC. Somado-se a isso, o aumento da altura também foi devido ao aumento da distância das linhas de interfaces em todas as bandas. O aumento da altura para a terceira otimização ficou, para os circuitos sintetizados, na média de 9%. As Figuras 4.8 e 4.9 mostram os resultados de ganho de largura e altura das três otimizações em relação a versão 3 do TROPIC.

Em quase todos os casos houve aumento da altura dos circuitos gerados, devido à necessidade de um número maior de linhas de roteamento. Mesmo na primeira otimização onde não houve redução na largura dos circuitos devido ao problema mostrado na Figura 4.5, em alguns casos houve aumento da altura, pois os pinos foram assinalados mais próximos em decorrência da otimização e também propiciaram um congestionamento

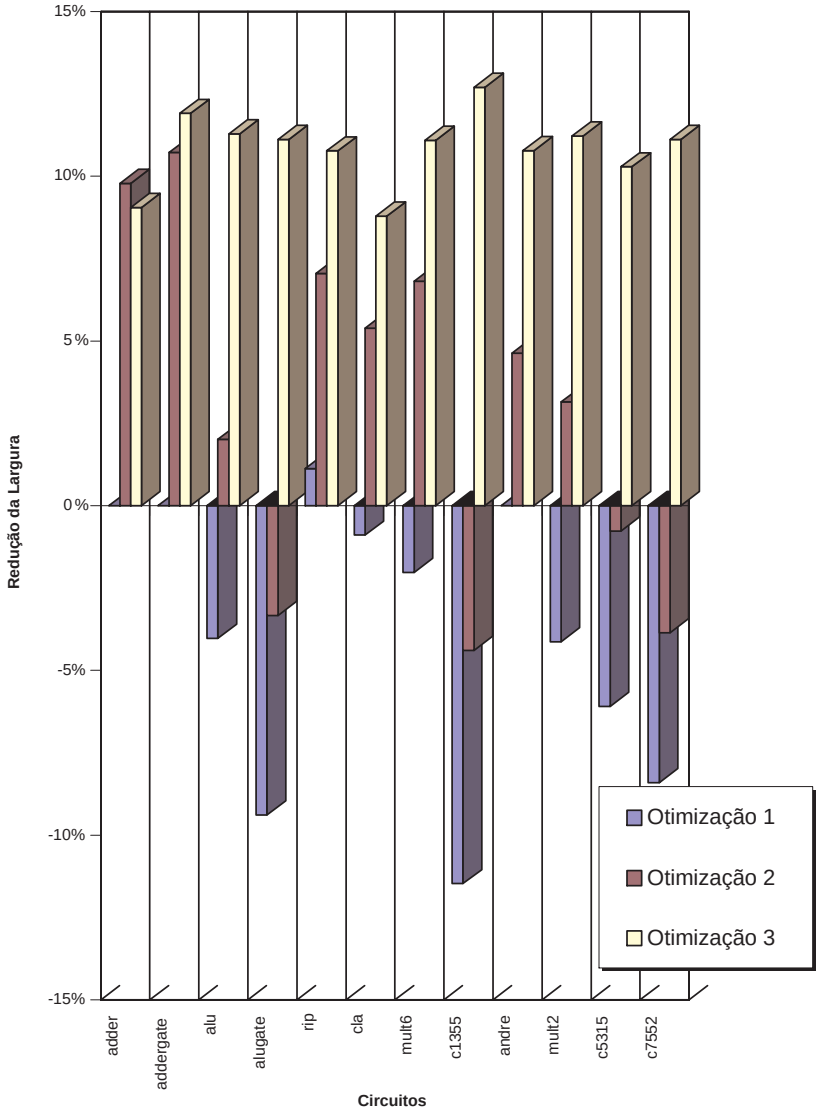

FIGURA 4.8 – Gráfico da redução de largura dos circuitos gerados com as três otimizações em comparação ao TROPIC3.

para o roteamento.

Na otimização 3, que resultou em redução de largura em todos os circuitos sintetizados, o aumento de altura foi mais significativo, já que além dos problemas com o roteamento, também foi preciso aumentar a altura da área de interface de cada banda para implementar os *jogs* no polissilício para o desalinhamento das cabeças de contato. Apesar do aumento em média de 9%, ainda assim foi possível alcançar um ganho na densidade de integração em torno dos 4%.

A Figura 4.10 mostra o leiaute completo gerado pela terceira otimização do circuito C1355. Nesse circuito quase a totalidade das áreas de dreno e fonte são mínimas e também

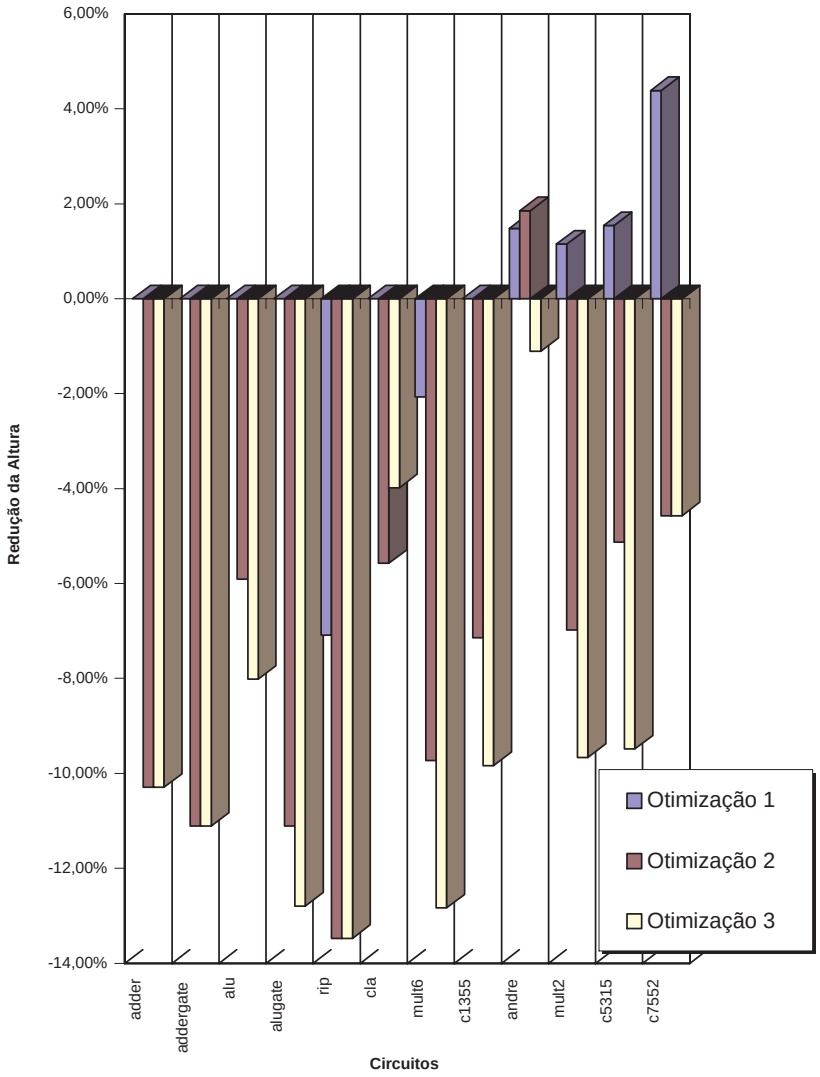

FIGURA 4.9 – Gráfico da redução de altura dos circuitos gerados com as três otimizações em comparação ao TROPIC3.

pode-se notar que a largura do circuito é determinada pela banda mais larga, sendo que em algumas bandas a largura do circuito não é utilizada totalmente. No circuito da Figura 4.10 pode-se perceber que as duas bandas mais inferiores não consomem toda a largura, mostrando também a eficiência da otimização empregada.

Um ponto importante a ser investigado em versões futuras da ferramenta, não só a melhora do algoritmo do roteamento, mas sim uma metodologia de roteamento, com roteamento completo sobre a área ativa e que permita então suprimir, ou pelo menos, reduzir os canais de roteamento.

Também é importante salientar, que apesar dos resultados de densidade de

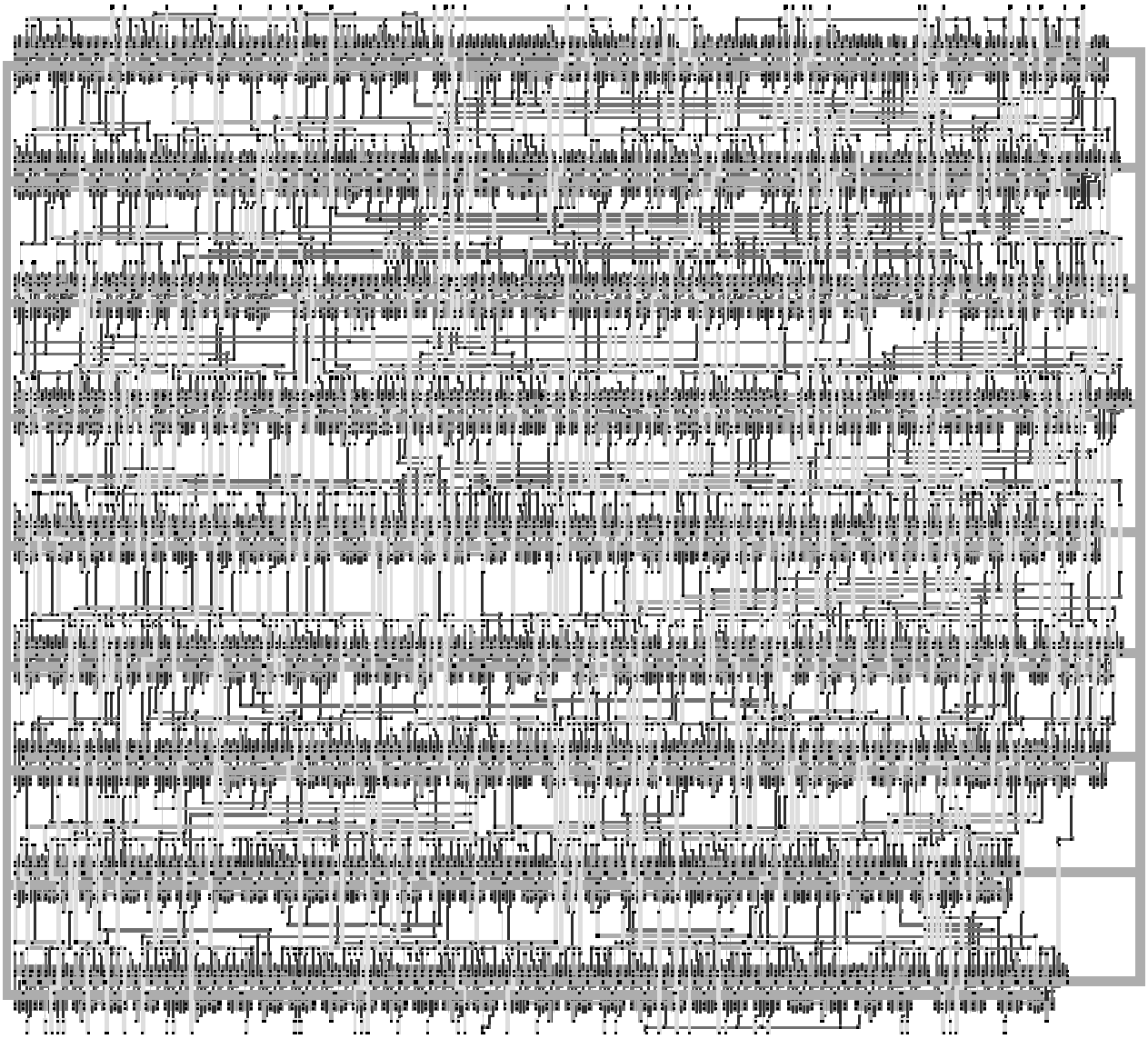

FIGURA 4.10 – Leiaute completo para o circuito C1355.

integração dos transistores não terem sido significativos, as otimizações apresentadas aqui são relevantes para uma melhora de densidade da ferramenta, pois os espaços deixados por um ineficiente uso das áreas de difusão não poderia ser recuperado nas outras etapas do processo de síntese do TROPIC3. Desta forma, o trabalho apresentado aqui, juntamente com estudos futuros em roteamento e uma metodologia para geração o mais livre possível de uma grade virtual, podem conduzir o TROPIC e WTROPIC a obterem resultados mais satisfatórios ainda para a densidade de integração do transistores.

Na atual implementação do WTROPIC houve a supressão da colocação dos *body-ties*. Essa supressão foi necessário, pois os contatos do substrato eram colocados sob todos os pontos onde a linha de alimentação é conectada a linha de metal-1. Já com a otimização das linhas de polissilício não é possível a inserção de *body-ties* em todos os

pontos, mas deve-se adicionar uma condição para a colocação. Observando-se os leiautes, pode-se perceber inicialmente que, sempre que um contato de alimentação for colocado e o par de transistores anterior não possuir coordenadas superiores ao ponto onde está sendo inserido o contato, é possível inserir um *body-tie*. Está condição ainda não está implementada no WTROPIC e precisa ser validada.

Todas as otimizações apresentadas neste capítulo fazem parte da ferramenta WTROPIC. O capitulo 5 apresenta como essa ferramenta foi desenvolvida com o intuito de refletir todas as alterações realizadas e como esta ferramenta se adequa ao ambiente CAVE.

5 O Ambiente CAVE e a Integração do WTROPIC

Um ambiente de apoio ao projeto - *CAD framework* - engloba os recursos disponibilizados aos desenvolvedores de ferramentasde CAD, ao usuário final dessas ferramentas e ao integrador ou administrador do ambiente de CAD em si [BAR 92]. Estes recursos visam facilitar os processos efetuados pelas três classes de usuários e reduzir o tempo gasto por eles nas suas atividades. Assim um CAD *framework* deve conter mecanismos para integrar ferramentas, para auxiliar no desenvolvimento de ferramentas a serem integradas e para permitir ao usuário final uso dessas ferramentas de forma simples e flexível.

Nesse sentido, [IND 98] propõe um modelo para o desenvolvimento de *frameworks* baseados em World Wide Web. As principais características do modelo são:

- uso de interface largamente conhecida - navegadores WWW;
- possibilidade de acesso remoto;
- independente de plataforma de hardware;
- sem grandes exigências para as máquinas clientes.

O ambiente CAVE é um protótipo e o resultado do esforço de alguns dos pesquisadores do Grupo de Microeltrônica da UFRGS em desenvolver ferramentas de apoio ao projeto. Assim sendo, a ferramenta WTROPIC apresentada neste trabalho integra-se a este ambiente como a possibilidade de realização de síntese física através da internet.

5.1 A Arquitetura do CAVE

O ambiente CAVE está baseado em uma arquitetura de interface de três níveis: (1) WebBrowser, (2) Hiperdocumento e (3) Ferramenta. Como o ambiente de projeto é baseado em WWW, o uso do navegador WWW, principal interface do usuário com o WWW e disponível comercialmente, permite a redução no esforço de criar a base para o restante da interface. Adicionalmente, tem-se a redução da sobrecarga cognitiva - a dificuldade em se obter familiaridade - que algum projetista teria ao deparar-se pela primeira vez com a interface.

O segundo nível, permite acesso a todas as partes do ambiente de projeto. Esta camada é implementada através de hiperdocumentos. Os hiperdocumentos informam ao

nível 1, como os objetos do framework estão relacionados e tem como principal objetivo facilitar a navegação do usuário pelo diversos aspectos do framework.

O terceiro nível é a aplicação propriamente dita e ela definirá a interface com o usuário. Essas interfaces são inevitavelmente diversas, por motivos relacionados à sua funcionalidade e à sua arquitetura de integração com o ambiente de projeto.

Para a integração das ferramentas, o modelo proposto por [IND 98] prevê dois grupos de ferramentas:

1. **Alto grau de interação**: engloba ferramentas que requerem do projetista o intenso uso de interface gráfica, como editores de esquemáticos, editores de leiautes e ferramentas gráficas de síntese de alto nível. Essas ferramentas precisam ser implementadas ou re-implementadas usando a linguagem Java e anexadas a um hiperdocumento.

2. **Baixo grau de interação**: engloba ferramentas que não exijam uma interação constante do usuário. Nesse grupo o usuário age como um provedor de dados e analisador de resultados, enquanto o processamento desses dados é feito por uma máquina servidora.

A Figura 5.1 mostra o modelo de comunicação para estes dois grupos de ferramentas.

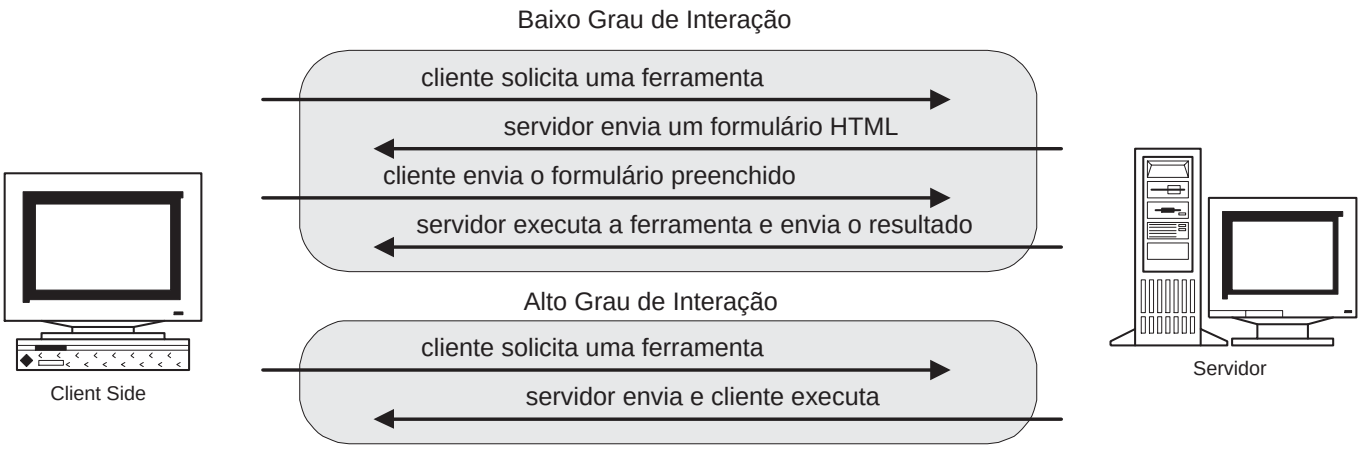

FIGURA 5.1 – Modelo de comunicação para os grupos de ferramentas.

5.2 O WTROPIC

O WTROPIC [FRA 2000] é uma ferramenta de síntese física e integra-se ao ambiente CAVE. Por ser uma ferramenta de baixo grau de interação, esta deveria utilizar o modelo de comunicação para este grupo.

Entretanto, a realização da síntese através de formulários HTML pode tornar-se tediosa e com as seguintes desvantagens:

- a interface com o usuário (formulário) fica desprovido de habilidade. Neste modelo validações de campos e auxilio ao usuário deveriam ser implementadas em linguagem script - como Javascript - porém a implementação do interpretador dessas linguagens é definida pelo interpretador do navegador, podendo variar dependendo do fabricante de cada navegador;
- a aparência do formulário pode variar dependendo do navegador e das configurações pessoais de cada usuário;
- o tempo da geração não é previsível, pois depende da ocupação da máquina servidor. Após o usuário enviar o formulário preenchido, caso a elaboração do resultado consuma um tempo elevado, a sessão do usuário poderia ser invalidada e o usuário poderia ficar sem os seus resultados.

Por essas razões a ferramenta WTROPIC não foi elaborada através do uso de formulários e apresenta um modelo diferente para a comunicação do usuário com o servidor. O modelo, mostrado na Figura 5.2, utiliza-se de uma interface de usuário (cliente) escrita em Java Applet e de um servidor desenvolvido para a ferramenta para receber as requisições dos usuários através da interface.

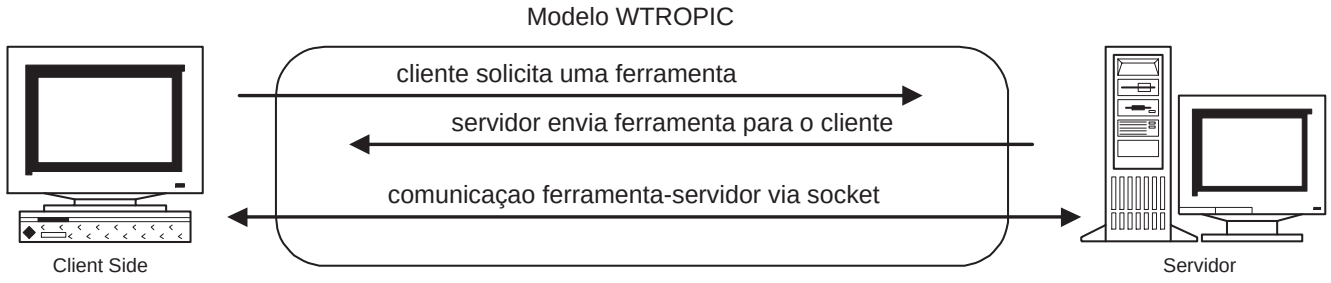

FIGURA 5.2 – Modelo de comunicação para o WTROPIC.

Esta mudança no modelo de comunicação não implica num afastamento do modelo proposto para o ambiente CAVE, mas sim uma evolução. Note-se, que a hierarquia de interfaces foi mantida, já que este tipo de interface é utilizado para ferramentas de alto grau

de interação e o usuário continua sendo o provedor de informações, só que as informações enviadas por ele podem chegar diretamente ao servidor sem a necessidade de requisições HTTP.

O modelo do WTROPIC apresenta vantagens quanto ao modelo anterior, e o próprio projeto CAVE está sendo alterado para uma nova versão neste novo modelo como mostrado em [IND 2000]. Este novo modelo, permite que todas as ferramentas possuam interfaces escritas em Java, fazendo com que estas ferramentas possam ter aparência similar. Essa similaridade é muito interessante para um *framework*. Também, o uso de Applets Java para a interface do usuário, permite um maior controle sobre os processos realizados no servidor para as ferramentas de baixo grau de integração.

5.2.1 O Servidor

Para aceitar e processar as requisições do cliente, um servidor, chamado WTSERVER, foi desenvolvido para a ferramenta WTROPIC. Esse servidor é um programa escrito em linguagem C e sistema operacional UNIX. A principal função do servidor é a execução do processo de síntese física. A escolha da linguagem C para o desenvolvimento foi motivada, especialmente, pelo fato dos algoritmos da síntese física estarem codificados em C. Assim sendo, a tarefa de encapsular estes algoritmos dentro de um servidor tornou-se bem menos penosa e garantiu o desempenho do servidor para atender a diversas requisições simultâneas de diferentes usuários.

A comunicação com o cliente é estabelecida via *socket*. *Socket* é um fluxo de conexão de rede e é o método utilizado para realizar a comunicação na Internet. Quando o usuário solicita uma URL, um socket provê um fluxo de dados para o usuário, e esse fluxo de dados se desfaz assim que a requisição for atendida. Isto provoca um protocolo sem memória associado com formulários HTML e programas CGI, já que o fluxo de dados e a comunicação só permanecem enquanto uma requisição estiver sendo atendida. Assim sendo, manter informações sobre o usuário, como por exemplo seu nome e o nome de arquivos de tecnologia utilizados em sínteses anteriores, em ferramentas elaboradas através de formulários HTML ou deve ser feito pelo servidor e enviado ao usuário toda vez que uma nova página é enviada, ou mantido em formulários nas páginas HTML e enviado ao servidor em cada requisição para que o servidor utilize estas informações e as devolva em uma nova página para a próxima requisição. Qualquer uma das soluções acarreta num acréscimo no volume de dados em cada requisição do cliente e em cada resposta do servidor.

Entretanto, é possível estabelecer um socket que permanece válido durante todo o

tempo que o cliente e o servidor estiverem trocando mensagens. A implementação de sockets existem em diversas linguagens, não importando qual delas será usada em cada extremidade de uma comunicação. Cada novo socket criado pertence a uma sessão de um usuário. Caso um mesmo usuário crie diversas sessões, por exemplo executando o browser várias vezes em sua máquina e para cada um deles acessar uma mesma URL, cada sessão terá seu próprio fluxo de comunicação. Essa identificação de cada fluxo fica a cargo da implementação de socket de cada linguagem, não precisando ser uma preocupação do desenvolvedor da aplicação.

O servidor deve fornecer acesso a diversos clientes individualmente e simultaneamente e poder processar todas as requisições. Isto é alcançado pela divisão do processo do servidor (*fork*) em um novo processo toda a vez que uma nova conexão é estabelecida. Dessa maneira, existirão na máquina servidora um número de cópias do WTSERVER igual ao número de clientes que estiverem conectados num determinado instante.

Cada instância do servidor cria um *socket* de comunicação com o cliente que requisitou a conexão e possibilita a troca de mensagens com a aplicação. A troca de mensagens para esta conexão criada com o servidor é exclusiva para este cliente e a conexão e a instância do servidor persistirão enquanto o usuário mantiver sua sessão aberta. A sessão é encerrada quando o usuário termina a execução de seu browser ou solicita através da ferramenta para que a conexão seja encerrada.

Para a troca de mensagens entre o servidor e o cliente através do *socket* foi criado um pequeno protocolo para o envio e recebimento de dados. Do lado do cliente, foi criada uma classe para encapsular uma conexão e que contém métodos para a solicitação de tarefas ao servidor. Por simplicidade, as requisições ao servidor foram numeradas para a sua identificação. Então, cada requisição do usuário possui os seguintes campos:

- **tamanho do buffer enviado**: este campo indica quantos bytes estão sendo enviados ao servidor;
- **identificador da requisição**: um número indicando ao servidor que tarefa está sendo requisitada pelo cliente;
- **parâmetros**: uma lista de parâmetros, separados por um caracter nulo, com os parâmetros necessários a cada tarefa. Este campo não é obrigatório para todas as requisições.

A Figura 5.3 mostra o formato do buffer de dados enviado pelo cliente e um suposto buffer utilizado para o *login* de um usuário chamado **fragoso** com a senha **12345**.

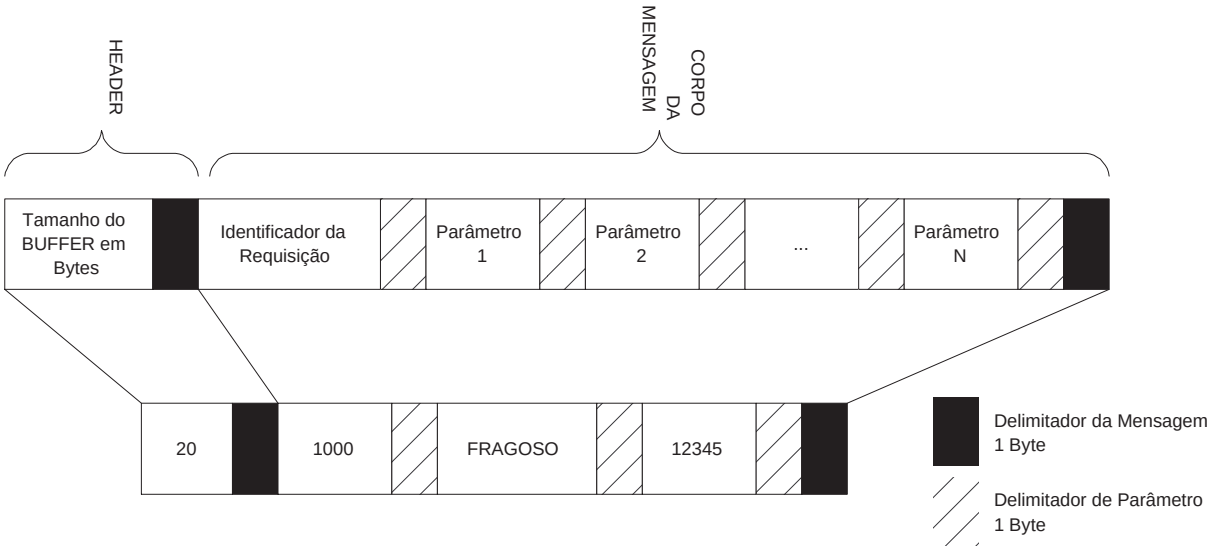

FIGURA 5.3 – Formato do buffer para requisições de tarefas ao servidor.

Similarmente, um protocolo é utilizado pelo servidor para resposta às solicitações de um cliente conectado. O buffer de resposta do servidor é montado da mesma maneira, contudo o campo de identificação da requisição não é enviado pelo servidor, apenas os parâmetros de retorno. A Figura 5.4 mostra o buffer montado pelo servidor e enviado pela ferramenta. A Figura também mostra o retorno para uma requisição de login com sucesso, retornando o código de sucesso igual a **1000** e a mensagem **Login Accepted**.

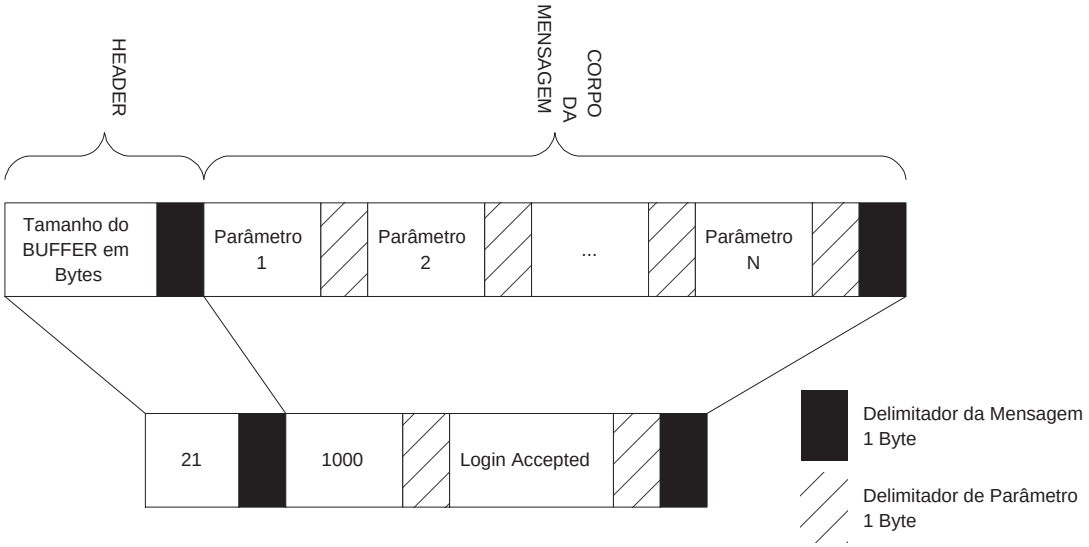

FIGURA 5.4 – Formato do buffer de retorno de mensagens do servidor.

Através desse simples protocolo, o usuário faz todas as requisições para o servidor

e as tarefas possíveis do WTSERVER são:

1. aceitar conexões de usuários;
2. validar o acesso aos dados;
3. controlar os usuários do WTROPIC, permitindo que os usuários manipulem apenas seus arquivos;
4. receber, guardar e enviar arquivos de configuração das regras de projeto da síntese física;
5. receber, armazenar e enviar arquivos de netlist SPICE;
6. receber os parâmetros para uma determinada geração;
7. executar a geração do leiaute;
8. enviar para o usuário os arquivos CIF gerados.

Para realizar essas tarefas foi criado uma hierarquia de diretórios onde o servidor deve ser instalado. Por questões de segurança o servidor deve ser instalado em uma conta de usuário comum da rede UNIX, e apenas o administrador da ferramenta deve ter acesso a esta conta. Os arquivos do servidor, bem como os arquivos do usuários que estiverem armazenados no servidor devem possuir acesso restrito, também, ao administrador da ferramenta. O servidor, em ambientes UNIX, deve ser disparado pelo administrador, dessa forma somente o servidor WTSERVER terá acesso à árvore de diretórios no qual foi instalado e aos arquivos dos usuários da ferramenta.

Para a instalação do servidor WTSERVER uma árvore com três sub-diretórios, mostrada na Figura 5.5, foi criada:

- *bin*: possui os arquivos executáveis da ferramenta, bem como os executáveis para realizar a síntese física;
- *users*: diretório utilizado para armazenar as informações dos usuários, arquivos de tecnologia, arquivo de entrada SPICE e arquivos CIF. Cada usuário da ferramenta possui um sub-diretório onde seus arquivos são armazenados;
- *log*: diretório onde são armazenados os arquivos de histórico de acesso e uso.

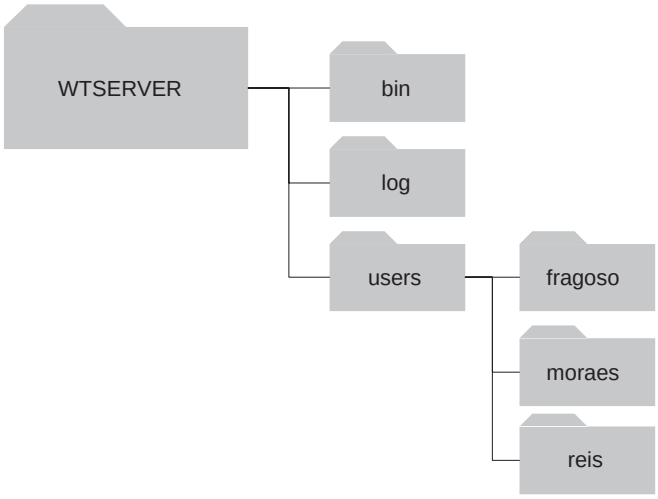

FIGURA 5.5 – Estrutura de diretório do servidor WTSERVER.

Muitas das funções implementadas pelo servidor, como a validação de usuários e o controle de arquivos, são funções de controle e deveriam estar implementadas no nível inferior do *framework*. Porém, o ambiente CAVE ainda não provê esse tipo de serviço e essas tarefas necessitaram ser desenvolvidas no servidor do WTROPIC.

Quando um cliente solicita a realização de uma síntese pelo servidor WTSERVER, o processo do servidor que está atendendo o cliente cria uma nova *thread*. A thread significa que uma determinada função do servidor está sendo executada em paralelo com as demais. Assim, é possível ao processo servidor continuar atendendo a outras solicitações de um mesmo usuário, como por exemplo enviar ao usuário o arquivo gerado em sínteses anteriores, mesmo quando este servidor estiver realizando uma síntese física. Por razões de desempenho e não arquiteturais, não é permitido pelo servidor que um determinado usuário execute mais de uma síntese simultaneamente no servidor.

Como resultado principal da síntese física no servidor, dois arquivos são produzidos: (1) o arquivo CIF com o leiaute realizado e (2) um arquivo de resultado da síntese, que pode ser consultado pelo usuário para verificar a qualidade de sua síntese. Os demais arquivos gerados durante a síntese física também ficam armazenados no servidor a disposição do usuário.

5.2.2 O Cliente

A interface com o usuário é realizada por uma Applet desenvolvida em Java versão 2 [MIC 2000] com suporte a Swing como mostrado na Figura 5.6. O Swing [MIC 2000a] é um pacote fornecido pela Sun na versão 2 do Java e possui melhor tratamento e

apresentação para a criação de interfaces gráficas. As applets, por definição, são pequenas aplicações, normalmente desenvolvidas em JAVA, e que são inseridas com objetos de hiperdocumentos da WEB. Desta forma, as applets são desenvolvidas para serem acessadas e utilizadas através de um navegador.

O usuário acessa a aplicação WTROPIC através dessa Applet encapsulada em hiperdocumento disponível em uma URL. Para o usuário, apesar da síntese estar sendo realizada em uma máquina remota, ele tem a impressão de possuir uma ferramenta de síntese instalada e disponível na sua própria máquina. Esse efeito é produzido no usuário, pois toda a troca de informações entre a aplicação cliente e o servidor não é pode ser percebida pelo usuário.

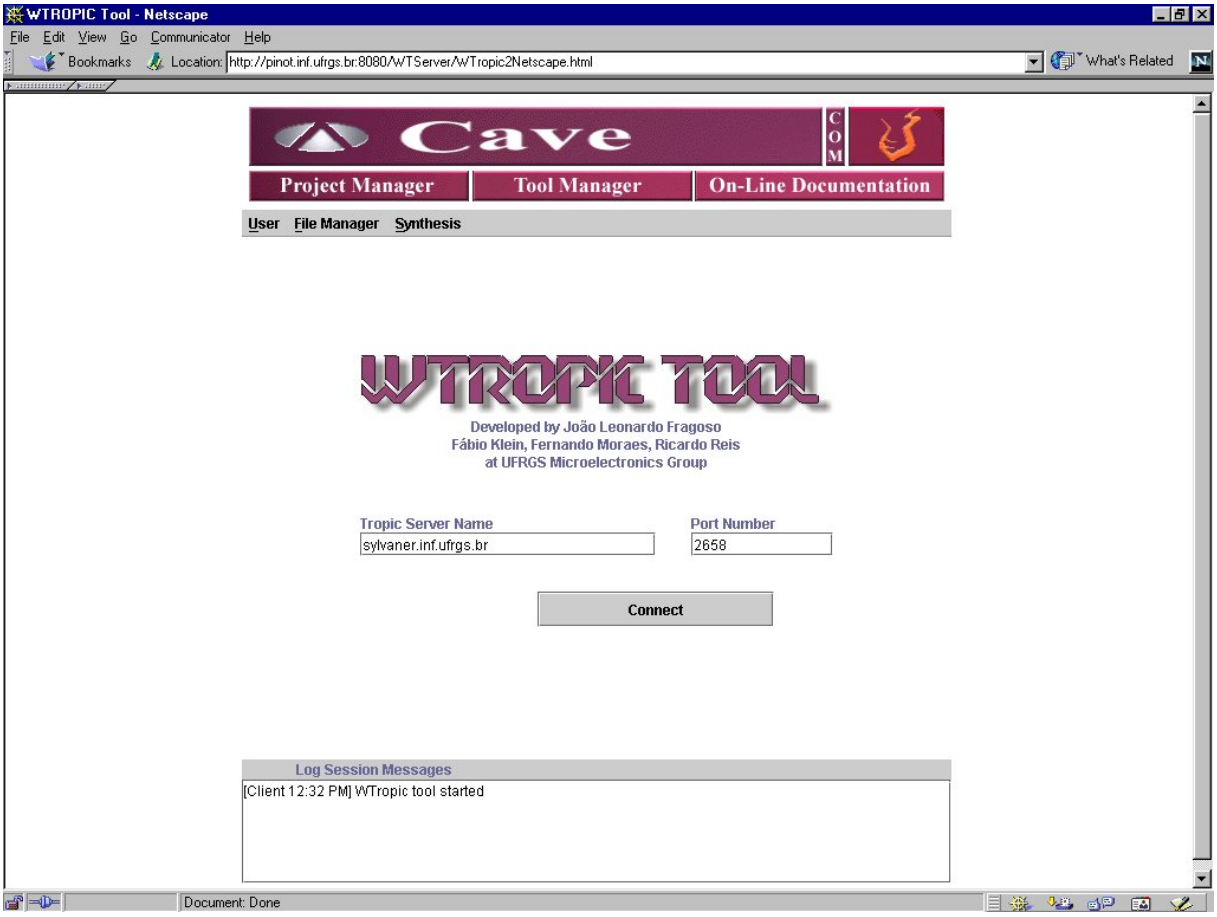

FIGURA 5.6 – Interface do WTROPIC - conexão com o WTSERVER.

Outrossim, WTROPIC é nome de todo o sistema, mas ao longo deste capítulo usar-se-á como o nome da aplicação cliente. Já que o usuário só tem contato com a interface desenvolvida em Java e este possui a impressão de que a interface é toda a aplicação, é natural que use-se aqui o nome WTROPIC somente para o lado do cliente.

As principais funções realizadas pelo WTROPIC são:

1. conectar-se a um servidor WTSERVER indicado pelo usuário;
2. criar contas de usuários no servidor no qual estiver conectado;
3. verificar, para cada conexão no servidor, o nome a senha de acesso do usuário;
4. encerrar a sessão do usuário;
5. auxiliar o usuário na criação de arquivos de configuração para síntese física;
6. enviar arquivos contendo netlist SPICE para o servidor;
7. receber arquivos gerados durante a síntese física;
8. permitir a leitura do arquivo contendo o resultado da última síntese;
9. solicitar a execução de uma síntese física, que será realizada no servidor.

A execução de todas as tarefas listadas acima, implica em comunicação com o servidor, isto é, troca de informações entre a aplicação cliente e servidora. Por exemplo, o *login* de um usuário, implica no envio do nome do usuário e a senha ao servidor. Caso o *login* seja autorizado o servidor envia à aplicação cliente os arquivos do usuários disponíveis no servidor.

Obviamente, a disponibilização da ferramenta é feita por um hiperdocumento que contém a inserção da Applet. Esse hiperdocumento deve estar disponível em um servidor WWW na rede internet. Entretanto, a máquina que contém o servidor WTSERVER pode ser qualquer máquina também disponível na rede, não precisando conter um servidor WWW, pois a conexão da ferramenta com o WTSERVER é feita diretamente. Esta característica permite que o processamento possa ser distribuído em diversas máquinas e, adicionalmente, vários servidores WTSERVER podem ser disponibilizados.

A ferramenta permite ao usuário definir em qual servidor WTSERVER ele deseja conectar-se como mostrado na Figura 5.6. Após a conexão com o servidor, o usuário poderá efetuar o *login* no servidor WTSERVER escolhido na tela mostrada na Figura 5.7, para começar o trabalho de síntese.

Na Figura 5.6 é possivel identificar os três níveis da arquitetura CAVE, o nível do navegador - janela do browser e parte externa da aplicação , o nível de hiperdocumentos - integrado com a barra do CAVE e o nível da aplicação - menus, caixas de texto e botões ao centro. A interface do WTROPIC fornece um menu de funções na parte superior onde estão listadas todas as tarefas que podem ser executadas.

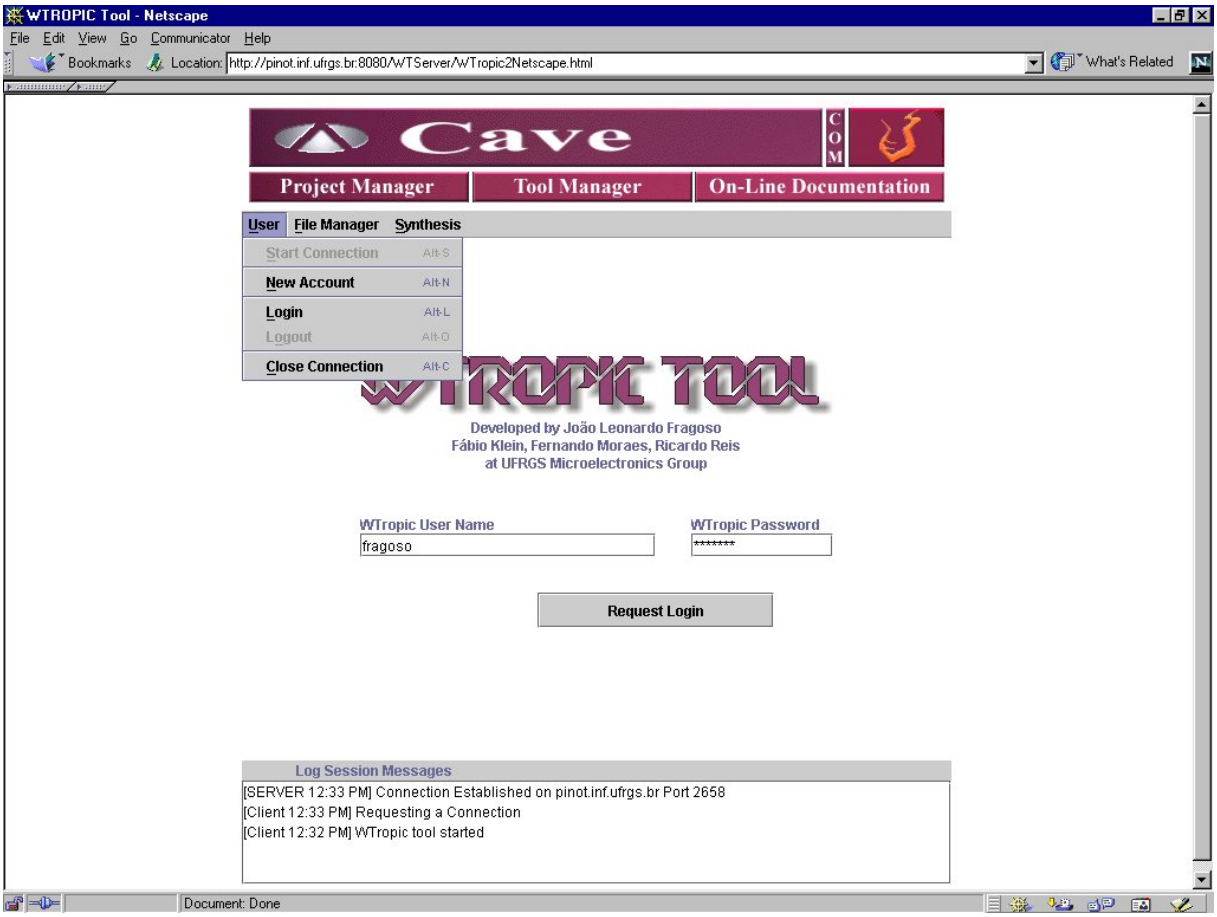

FIGURA 5.7 – Tela para *login* no WTROPIC.

Outra facilidade da ferramenta está na possibilidade do arquivo de configuração de regras de projeto ser produzido através da interface (ver Figura 5.8). Os parâmetros de geração são definidos na interface que se encarrega de enviar estes dados ao servidor para serem gravados como um arquivo de configuração do usuário. Dessa maneira, o usuário não precisa conhecer o formato do arquivo de configuração, apenas precisa enviar as informações ao servidor, diminuindo consideravelmente o tempo de treinamento.

Além disso, a ferramenta oferece uma tela para requisição de síntese(ver Figura 5.9), onde também os parâmetros são informados através da interface. O usuário indica, entre os arquivos já enviados ao servidor, qual deseja sintetizar, o arquivo de regras de projetos a largura e comprimento dos transistores a serem gerados, caso não esteja especificado na descrição SPICE e se deseja ou não fazer a extração das capacitâncias parasitas.

O WTROPIC oferece, então, uma maneira simples e eficiente de geração de leiautes através da internet, permitindo até que usuários inexperientes possam realizá-la. Também, não é exigido do usuário equipamentos de hardware mais sofisticados para a execução,

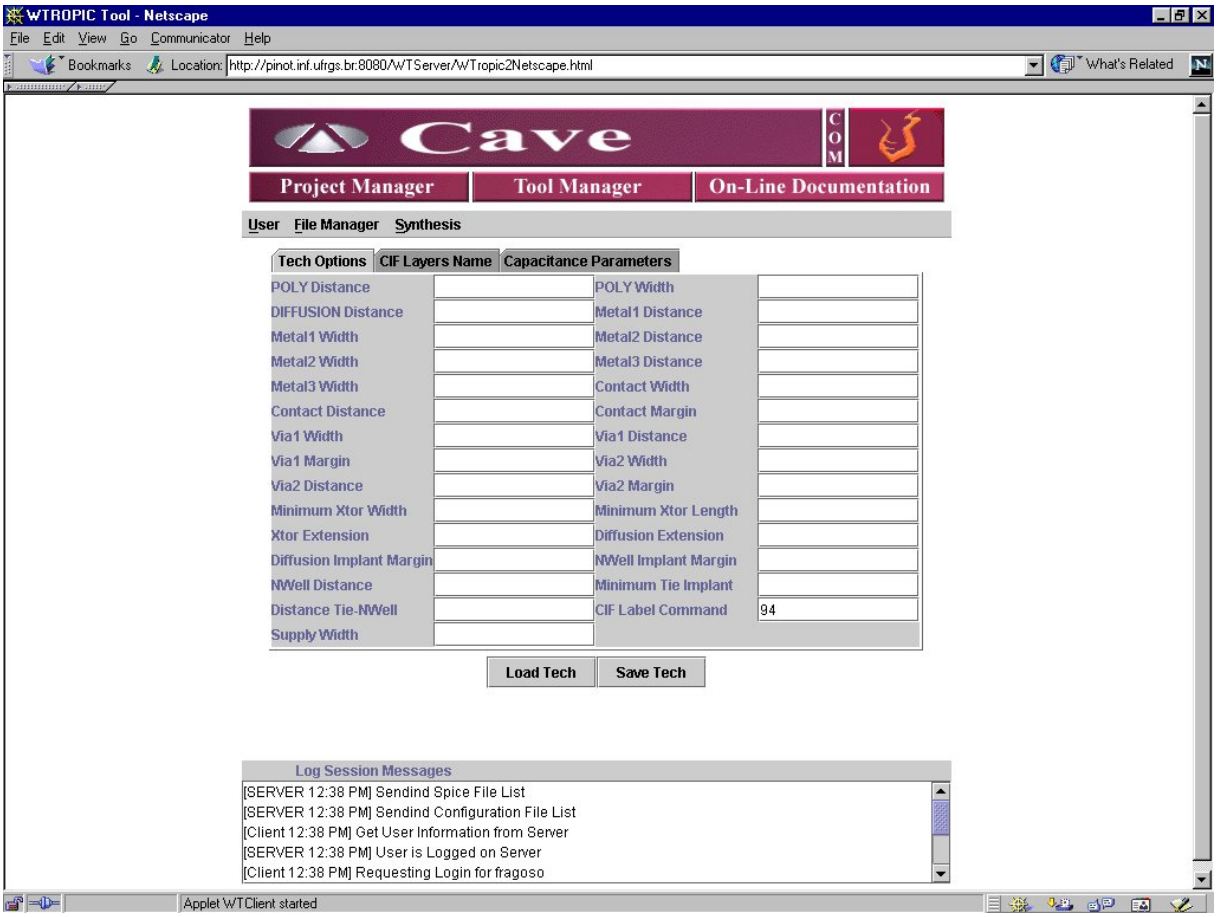

FIGURA 5.8 – Tela de configuração dos parâmetros tecnológicos no WTROPIC.

apenas um computador com acesso a internet e um navegador.

O uso da ferramenta também não impõe nenhuma exigência severa de tráfego na rede, pois a comunicação é feita diretamente com o WTSERVER. Dessa maneira, toda a troca de informações é feita através do *socket*, não sendo necessário nenhum protocolo adicional, como por exemplo a troca de dados através de formulários HTML e o envio de páginas HTML para o usuário. No caso do WTROPIC a situação mais exigente à condição de rede é a transferência de arquivos de entrada e saída de síntese.

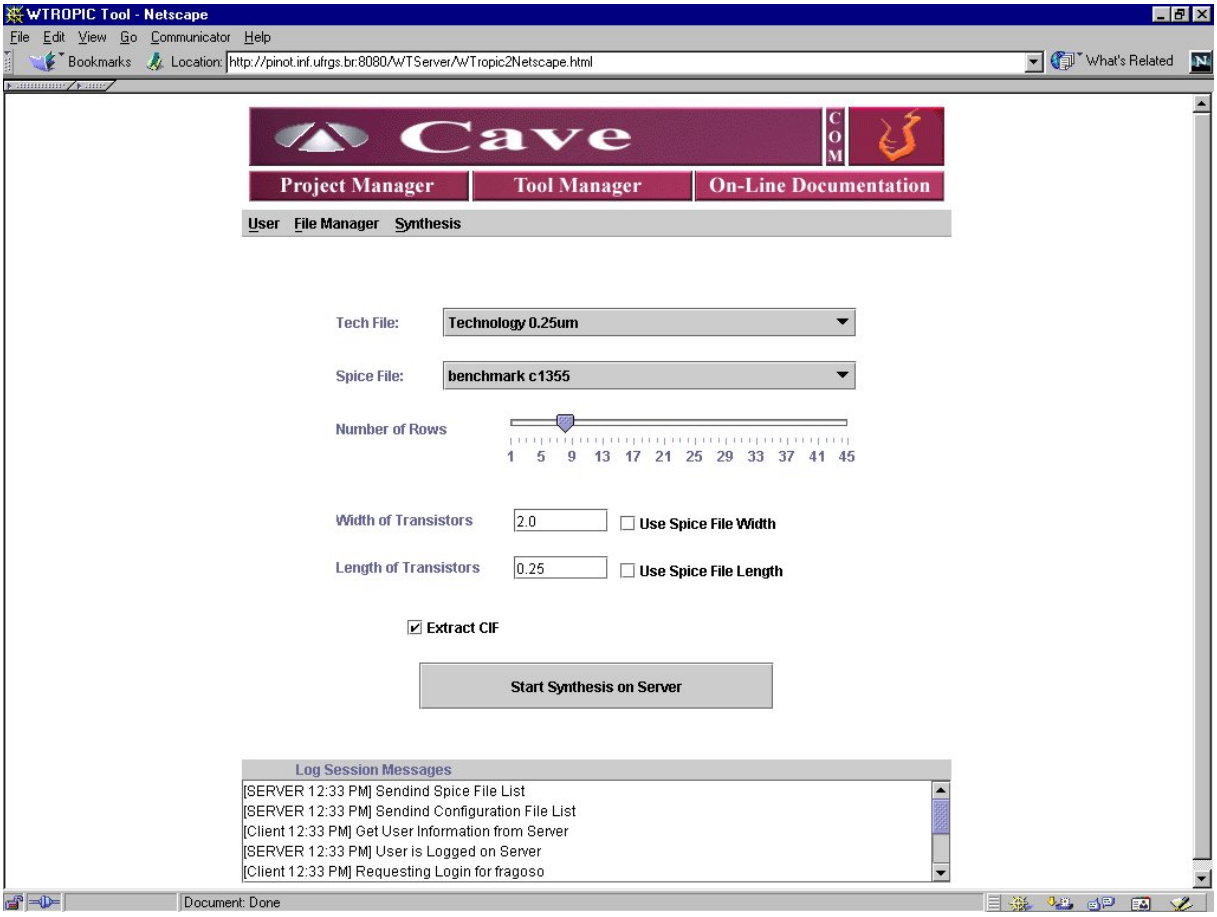

FIGURA 5.9 – Tela de síntese no WTROPIC.

6 Conclusões e Trabalhos Futuros

Este trabalho apresentou a realização da ferramenta WTROPIC - um gerador automático de macro células acessível via WWW. O WTROPIC é um sintetizador de leiautes concebido a partir da ferramenta TROPIC, com otimizações na topologia do leiaute e com uma interface desenvolvida para permitir o acesso remoto dos usuários.

As alterações na topologia do leiaute do polissilício apresentaram resultados positivos, conduzindo a um redução média da largura dos circuitos em 11% quando comparado com o TROPIC3. Entretanto, a diminuição da largura dos circuitos aumenta a utilização da grade de roteamento e mostrou a ineficiência do algoritmo de roteamento. Esta ineficiência provoca nos circuitos gerados utilizando o WTROPIC um aumento de altura quando comparado ao TROPIC3 com o mesmo número de bandas, pois o algoritmo de roteamento é o mesmo e no WTROPIC, esse algoritmo acaba por inserir um número maior de linhas de roteamento.

Contudo, apesar de uma altura maior, as otimizações conseguiram alcançar um aumento na densidade dos circuitos em cerca de 4%, além de provocarem sensível diminuição nas áreas de dreno e fonte. Acredita-se que com otimizações no algoritmo de roteamento, possa-se diminuir o número de linhas necessárias para a realização do roteamento e assim, junto com as otimizações apresentadas aqui, alcançar um significante aumento na densidade de integração.

Também foram apresentados neste trabalhos, os esforços realizados para a integração do WTROPIC no ambiente CAVE e a sua utilização através da internet. Após esses esforços, conseguiu-se uma ferramenta com as seguintes características:

Acesso remoto: a ferramenta pode ser acessada a partir de qualquer máquina conectada à rede internet.

Independência de plataforma: apesar dos algoritmos da síntese física estarem escritos em C para sistema operacional UNIX, a interface do cliente, desenvolvida em Java, pode ser executada em qualquer máquina independente da plataforma utilizada.

Requisitos simples de hardware e software: as exigência para a utilização da ferramenta são apenas estar conectado a internet e possuir uma máquina que possua um navegador internet e capaz de executar uma máquina virtual Java.

Padronização da interface: independentemente da plataforma utilizada a interface apresentar-se-á e comportar-se-á igualmente, além de ser uma interface conheci-

da, pois utiliza menus e botões, similarmente ao navegador internet, característica não alcançada com o uso CGI e formulários HTML como proposto em [IND 98].

Adicionalmente, o WTROPIC propôs uma melhoria em relação ao modelo de integração apresentado originalmente no ambiente CAVE. Este novo modelo de integração, permite a todas aplicações integradas ao *framework*, mesmo que tenham um pequeno grau de interação, fazer uso de *Applets* Java para permitir a comunicação remota e disponibilizar ao usuário uma interface mais poderosa e robusta para suas aplicações. Permite também uma padronização de aparência e comportamento entre todas as ferramentas do ambiente, já que todas as aplicações, no ponto de vista do usuário final, possuirão uma interface através de *Applets*.

Apesar da mudança no paradigma de integração das ferramentas, a ferramenta WTROPIC pode ser facilmente integrada ao ambiente CAVE, pois contém os três níveis da arquitetura de interface proposta e porque possui a interface semelhante a ferramentas de alto grau de interação.

O servidor WTSERVER foi desenvolvido especificamente para atender aos requerimentos da aplicação WTROPIC. No entanto, esse servidor é facilmente extensivo e permite que o trabalho desenvolvido possa ser utilizado para outras ferramentas de projeto que já possuam implementação e que possuam uma baixo grau de interação - tipicamente, ferramentas executadas através de linha de comando. Logo, este trabalho contribuiu para o desenvolvimento de ferramentas do Grupo de Microeletrônica da UFRGS, possibilitando que uma série de outras ferramentas possam ser integradas ao ambiente simplesmente pela disponibilização de um *Applet* Java.

Como perspectivas de futuros trabalhos, existe a possibilidade de realização de otimizações no algoritmo de roteamento do WTROPIC permitindo uma maior densidade de integração.

Somando a isso, as atividades de autenticação do usuário e manipulação de seus arquivos deveriam estar implementadas no ambiente do *framework*. Existe também a possibilidade da existência de vários servidores para a aplicação WTROPIC, porém todos comportam-se individualmente sem compartilhamento de dados e sem compartilhamento das informações dos usuários.

Dessa maneira, nenhum modelo de autenticação está proposto no ambiente CAVE, podendo assim ser objeto de estudo um modelo capaz de autenticar e manter os dados do usuário em um *framework* com ferramentas distribuídas e mesmo possuindo diversos servidores do *framework* distribuídos.

Bibliografia

[APT 87] APTE, J.; G.KEDEM. Strip layout: a new layout methodology for standard circuits modules. In: DESIGN AUTOMATION CONFERENCE, 24., 1987. **Proceedings...** New York:ACM/IEEE, 1987. p.363–369.

[BAR 92] BARNES, Timothy et al. **Electronic CAD frameworks.** Norwell, Massachusetts: Kluwer Academic Publishers, 1992. 195p.

[CAD 91] CADENCE. **Virtuoso layout synthesizer - LAS - user guide version 4.2**. Italy: CADENCE, 1991.

[CAR 92] CARLSON, B.; CHEN, C. Y.; MELIKSETIAN, D. S. An efficient algorithm for the identification of dual eulerian graphs and its application to the cell layout. In: ISCAS, 1992, San Diego. **Proceedings...** New York:ACM/IEEE, 1992. p.2248–2251.

[CHA 87] CHANG, Y.; CHANG, S. C.; HSU, L. H. Automated layout generation using gate matrix approach. In: DESIGN AUTOMATION CONFERENCE, 24., 1987. **Proceedings...** New York:ACM/IEEE, 1987. p.552–558.

[CHE 89] CHEN, C. C.; CHOW, S. L. The layout synthetizer: an automatic netlist-to-layout system. In: DESIGN AUTOMATION CONFERENCE, 26., 1989. **Proceedings...** New York:ACM/IEEE, 1989. p.232–238.

[CHE 88] CHEN, C. Y.; HOU, C. Y. A new layout optimization methodology for CMOS complex gates. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1988, Santa Clara, CA. **Proceedings...** New York:ACM Press, 1988. p.368–371.

[CHE 88a] CHEN, H. Y.; KANG, S. M. iCOACH: a circuit optimization aid for CMOS high-performance circuits. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1988, Santa Clara, CA. **Proceedings...** New York:ACM Press, 1988. p.372–375.

[DEV 94] DEVADAS, Srinivas; GHOSH, Abhijit; KEUTZER, Kurt. **Logic synthesis.** New York: McGraw-Hill, 1994. 406p.

[FRA 2000] FRAGOSO, João Leonardo; MORAES, Fernando; REIS, Ricardo. WTROPIC: a www-based macro-cell generator. In: SBCCI BRAZILIAN SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 2000, Manaus,Brasil. **Proceedings...** Los Alamitos:IEEE Computer Society, 2000. p.133–138.

[FRA 98] FRAGOSO, Joao; REIS, Ricardo. **Comparação entre estilos de leiautes para definição de metodologia de geração atomática de células**. Porto Alegre: CPGCC da UFRGS, 1998. (RP-185).

[GAJ 83] GAJSKI, D. D.; KUHN, R. H. New VLSI Tools. **IEEE Computer**, New York, v.16, n.12, p.11–14, Dec. 1983.

[GUP 2000] GUPTA, A.; HAYES, J. P. CLIP: integer-programming-based optimal layout synthesis of 2D CMOS cells. **IEEE Transactions on Design Automation of Electronic Systems**, New York, v.5, n.3, p.510–547, July 2000.

[HAS 71] HASHIMOTO, A.; STEVENS, J. Wire routing by optimizing channel assignment within large apertures. In: DESIGN AUTOMATION CONFERENCE, 8., 1971. **Proceedings...** New York:ACM/IEEE, 1971. p.155–159.

[HEE 92] HEEB, H.; FICHTNER, W. A module generator based on the pq-tree algorithm. **IEEE Transactions on Computer-Aided Design**, New York, v.11, n.7, p.876–884, July. 1992.

[HSI 91] HSIEH, Y. et al. LiB: a CMOS cell compiler. **IEEE Transactions on Computer-Aided Design**, New York, v.10, n.8, p.994–1005, Aug. 1991.

[HWA 91] HWANG, C. Y. et al. An Efficient Layout Style for 2-Metal CMOS Leaf Cells and Their Automatic Generation. In: DESIGN AUTOMATION CONFERENCE, 28., 1991. **Proceedings...** New York:ACM/IEEE, 1991. p.481–486.

[IND 2000] INDRUSIAK, Leandro; REIS, Ricardo. From a hyperdocument-centric to an object-oriented approach for cave project. In: SBCCI BRAZILIAN SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 2000, Manaus,Brasil. **Proceedings...** Los Alamitos:IEEE Computer Society, 2000. p.125–130.

[IND 98] INDRUSIAK, Leandro Soares. **Ambiente de Apoio ao Projeto de Circuitos Integrados baseado no WWW**. Porto Alegre: CPGCC da UFRGS, 1998. Dissertação de Mestrado.

[JOH 95] JOHANN, Marcelo; REIS, Ricardo. A Full Over-the-Cell Routing Model. In: IFIP VERY LARGE SCALE INTEGRATION CONFERENCE, 1995, Tokyo, Japan. **Proceedings...** New York:IFIP/IEEE/ACM, 1995. p.845–850.

[KLE 2000] KLEIN, Fabio Ferreira. **Extração de Capacitâncias e Resistências Parasitas em Circuitos CMOS Submicrônicos**. Porto Alegre: CPGCC da UFRGS, 2000. Dissertação de Mestrado.

[KLE 2000a] KLEIN, Fabio Ferreira; MORAES, Fernando; REIS, Ricardo. LASCA - interconnect parasitic extraction tool for deep-submicron ic design. In: SBCCI BRAZILIAN SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, 2000, Manaus,Brasil. **Proceedings...** Los Alamitos:IEEE Computer Society, 2000. p.327–332.

[LAK 90] LAKHANI, G.; RAO, S. A multiple row-based layout generator for CMOS cell. In: ISCAS, 1990, New Orleans. **Proceedings...** New York:ACM/IEEE, 1990. p.1697–1700.

[LEF 89] LEFEBVRE, M.; CHAN, C. Optimal ordering of gate signals in CMOS complex gates. In: CICC, 1989. **Proceedings...** New York:ACM/IEEE, 1989. p.17.5.1–17.5.4.

[LIN 92] LIN, H.; PERNG, H.; HSU, Y. Cell height reduction by routing over the cells. In: ISCAS, 1992, San Diego. **Proceedings...** New York:ACM/IEEE, 1992. p.2244–2247.

[LOP 80] LOPEZ, A.; LAW, H. A dense gatematrix layout for MOS VLSI. **IEEE Transactions on Electronic Devices**, New York, v.ED-27, n.8, p.1671–1675, Aug. 1980.

[MAD 89] MADSEN, J. A new approach to optimal cell synthesis. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1989, Santa Clara, CA. **Proceedings...** New York:ACM Press, 1989. p.336–339.

[MAZ 91] MAZIASZ, R. L.; HAYES, J. P. Exact width and height minimization of CMOS cells. In: DESIGN AUTOMATION CONFERENCE, 28., 1991. **Proceedings...** New York:ACM/IEEE, 1991. p.487–493.

[MAZ 87] MAZIASZ, R. L.; HAYES, J. P. Layout optimization of CMOS functional cells. In: DESIGN AUTOMATION CONFERENCE, 24., 1987. **Proceedings...** New York:ACM/IEEE, 1987. p.551–554.

[MIC 2000] MICROSYSTEM, Sun. **Java tutorial**. Disponível em: <http://java.sun.com/tutorial/>. Acesso em:ago. 2000.

[MIC 2000a] MICROSYSTEM, Sun. **About jfc and swing**. Disponível em: <http://java.sun.com/docs/books/tutorial/uiswing/start/swingIntro.html>. Acesso em:ago. 2000.

[MOO 79] MOORE, Gordon. VLSI: some fundamental challenges. **IEEE Spectrum**, New York, v.16, n.2, p.30–36, Apr. 1979.

[MOR 2000] MORAES, Fernando Gehm. **Anatomia de uma ferramenta de síntese**. Disponível em:

<http://www.inf.pucrs.br/moraes/papers/emicro_doc.pdf>. Acesso em:dez. 2000.

[MOR 94] MORAES, Fernando Gehm. **Synthese Topologique de Macro-Celluls en Technologie CMOS**. Montpellier, FR: Université Montpellier, 1994. Tese de Doutorado.

[MUR 96] MURRAY, B.; HAYES, John P. Testing ICs:getting to the core of the problem. **IEEE Computer**, New York, v.29, n.11, p.32–38, Nov. 1996.

[NAK 92] NAKAGAKI, T.; YAMADA, S.; FUKUNAGA, K. Fast optimal algorithm for CMOS functional cell layout based on transistor reordering. In: ISCAS, 1992, San Diego. **Proceedings...** New York:ACM/IEEE, 1992. p.1697–1700.

[ONG 89] ONG, C.; LI, J.; LO, C. GENAC: an automatic cell synthesis tool. In: DESIGN AUTOMATION CONFERENCE, 26., 1989. **Proceedings...** New York:ACM/IEEE, 1989. p.239–244.

[PRE 85] PREAS, Bryan; LORENZENTTI, Michael. **Physical design automation of VLSI systems.** Melo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1985. 510p.

[REI 2000] REIS, Ricardo; et. al. **Sistemas digitales - síntese física de circuitos integrados**. Bogota, Colombia: CYTED, 2000. 373p.

[REI 2000a] REIS, Ricardo. **Concepção de circuitos integrados.** Porto Alegre, Brasil: Porto Alegre: Sagra Luzzatto, 2000. 252p.

[REI 97] REIS, André Inácio. **Assignation Technologique sur Bibliotheques Virtuelles de Portes Complexes CMOS**. Montpellier, FR: Université Montpellier, 1997. Tese de Doutorado.

[REI 87] REIS, Ricardo Augusto da Luz. A new Standard Cell CAD Methodology. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1987, New York, NY. **Proceedings...** New York:IEEE, 1987. p.385–388.

[REI 83] REIS, Ricardo. **Tess evaluater automatique des plans de masse de circuits vlsi**. Grenoble, FR: Institute National Polytechnique, 1983. Tese de Doutorado.

[SIA 97] SIA Semiconductor Industry Association. **The national technology roadmap for semiconductor**. Disponível em: <http://notes.sematech.org/ntrs/PublNTRS.nsf>. Acesso em:mar. 1997.

[UEH 81] UEHARA, Takao; VANCLEEMPUT, William M. Optimal Layout of CMOS Functional Arrays. **IEEE Transactions on Computers**, New York, v.30, n.5, p.305–312, May 1981.

[WAN 89] WANG, C. H. et al. An optimal transistor-chaining algorithm for CMOS cell layout. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 1989, Santa Clara, CA. **Proceedings...** New York:ACM Press, 1989. p.344–347.

[WAN 90] WANG, C. H. et al. An optimal transistor-chaining algorithm for CMOS cell layout. **IEEE Transactions on Computer-Aided Design**, New York, v.9, n.7, p.781–786, July 1990.

[WAN 93] WANG, C. H.; et. al. An efficient layout style for two-metal CMOS leaf cells and its automatic synthesis. **IEEE Transactions on Computer-Aided Design**, New York, v.12, n.3, p.410–423, Mar. 1993.

[WEI 67] WEINBERGER, A. Large scale integration of MOS complex logic: a layout method. **IEEE Journal of Solid-State Circuits**, New York, v.SC-2, n.4, p.182–190, Dec. 1967.

[WIM 87] WIMMER, S.; PINTER, R. Y.; FELDMAN, J. A. Optimal chaining of CMOS transistors in a functional cell. **IEEE Transactions on Computer-Aided Design**, New York, v.CAD-6, n.5, p.795–801, Sept. 1987.

[WIN 82] WING, O. Automated gate matrix layout. In: ISCAS, 1982, Rome. **Proceedings...** New York:ACM/IEEE, 1982. p.681–685.