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

CRISTINA MEINHARDT

Geração de Leiautes Regulares Baseados em Matrizes de Células

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

Prof. Dr. Ricardo Reis
Orientador

Porto Alegre, setembro de 2006.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Meinhardt, Cristina

Geração de Leiautes Regulares Baseados em Matrizes de Células/ Cristina Meinhardt – Porto Alegre: Programa de Pós-Graduação em Computação, 2006.

131 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2006. Orientador: Ricardo Augusto da Luz Reis.

1. Microeletrônica. 2. Ferramentas de CAD. 3. Síntese Física. 4. Geração de Leiaute. 5. Regularidade 6. Previsibilidade. I. Reis, Ricardo Augusto da Luz. II. Título.

"As reticências são os três primeiros passos
do pensamento que continua
por conta própria o seu caminho.”
Mário Quintana

AGRADECIMENTOS

Agradeço ao meu orientador, Professor Ricardo Reis, por todo o incentivo e apoio ao longo destes anos, especialmente pela preocupação com meus próximos passos. Também agradeço ao professor Reginaldo Tavares, a orientação e colaboração na realização deste trabalho.

Aos professores do curso de Engenharia de Computação da FURG: Celso Luiz Lopes Rodrigues, Nelson Lopes Duarte Filho e Sílvia da Silva Botelho por todas as experiências pessoais e profissionais transmitidas. Especialmente à Sílvia, pelas oportunidades dadas e confiança depositada. Ao professor Tabajara Lucas de Almeida por despertar o gosto da pesquisa científica.

Aos meus colegas de trabalhos, de pesquisas e de laboratório agradeço por todas as conversas, discussões e, principalmente, por toda a ajuda que me deram, indispensável para a conclusão deste trabalho. Especialmente, agradeço à Ana, Arnaldo, Glauco, Digeorgia, Lucas, Guilherme, Felipe, Renato, Sandro, Vagner e Edgard por termos desenvolvido grandes amizades ao longo deste tempo, em que dividimos tarefas, problemas e conquistas.

Agradeço aos meus pais por todo o carinho dedicado e toda a compreensão nas ausências, nos momentos difíceis, nos meus silêncios. Agradeço a presença, mesmo distante, das minhas irmãs, fundamental para desenvolver o meu trabalho.

Finalmente, aos meus queridos amigos Si, Melissa, Marcelo, Odorico, Rodrigo e André por estarem sempre ao meu lado, me apoiando e vivendo momentos inesquecíveis.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ASIC	*Application Specific Integrated Circuit*
BDD	*Binary Decision Diagram*
CAD	*Computer-Aided Design*
CEITEC	Centro de Excelência em Tecnologia Eletrônica Avançada
CIF	*Caltech Intermediate Form*
CLB	*Configurable Logic Block*
CMOS	*Complementary MOS*
CMP	*Chemical-Mechanical Polishment*
CVD	*Chemical Vapor Deposition*
DFM	*Design For Manufacturability*
DFP	*Design For Performance*
DFV	*Design For Value*
DLL	*Delay-locked loops*
DRC	*Design Rule Checking*
DSP	*Digital Signal Processor*
FOTC	*Full Over The Cell*
FPGA	*Field Programmable Gate Array*
GME	Grupo de Microeletrônica da UFRGS
GND	*Ground* – Tensão de Referência
IP *cores*	*Intellectual Property cores*
ITRS	*International Technology Roadmap for Semiconductors*
LUT	*Look-up Table*
MDP	*Mask Data Preparation*
NAND	Célula lógica que representa a função booleana ~(A . B)
NINA	NOR, Inversor e NAND
NMOS	*N-channel Metal Oxide Semiconductor Field –effect transistor*
NOR	Célula lógica que representa a função booleana ~(A + B)

NRE	*Non-Recurring Engineering*
orBDD	*Or Binary Decision Diagram*
PLA	*Programmable Logic Array*
PLL	*Phase-locked loops*
PMOS	*P-channel Metal Oxide Semiconductor Field –effect transistor*
PSM	*Phase Shift Mask*
RET	*Resolution Enhance Technique*
UFRGS	Universidade Federal do Rio Grande do Sul
ULA	Unidade Lógica Aritmética
ULA	Unidade Lógica Aritmética
ULG	Unidade Lógica Programável
VDD	Tensão da Fonte de Alimentação

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho trata de pesquisa de soluções para a síntese física de circuitos integrados menos susceptíveis aos efeitos de variabilidade decorrentes do uso de tecnologias de fabricação com dimensões nanométricas. Também apresenta a pesquisa e o desenvolvimento de uma ferramenta para a geração de leiautes regulares denominada R-CAT. A regularidade geométrica é explorada pela repetição de padrões básicos de leiaute ao longo de uma matriz. A regularidade é apontada como uma das melhores alternativas para lidar com os atuais problemas de fabricação em tecnologias sub-micrônicas. Projetos regulares são menos suscetíveis aos problemas de litografia, aumentam o *yield* e diminuem o tempo gasto em re-projeto. Além disso, circuitos regulares apresentam maior previsibilidade de resultados de potência, atraso e *yield*, principalmente pelo fato das células estarem pré-caracterizadas.

A ferramenta desenvolvida visa o trabalho com dois tipos de síntese física para leiautes regulares, produzindo circuitos integrados personalizáveis por todas as máscaras ou circuitos personalizáveis por algumas máscaras. O principal objetivo deste gerador é a facilidade de conversão e adaptação dependendo da abordagem de matriz escolhida. Isso facilitará a comparação entre diferentes alternativas de matrizes, a adoção de blocos lógicos diversos e de novas tecnologias.

O gerador de leiautes R-CAT identifica células adjacentes com conexões em comum entre elas e realiza a conexão entre essas células em metal 1, reduzindo o número de conexões a ser realizado pelo roteador em até 10%.

A ferramenta R-CAT está inserida em um fluxo de projeto e depende do método de síntese lógica adotado. Duas ferramentas de síntese lógica foram utilizadas: SIS e OrBDDs, oferecendo duas linhas de projeto: a primeira priorizando a área e a segunda priorizando timing e interconexões curtas. Ambas respeitando a mesma regularidade geométrica imposta pela matriz.

Os resultados obtidos demonstram que as matrizes SIS ocupam 53% menos área do que a estratégia orBDD e reduzem o *wire length* em 30%. Uma área menor é obtida devido ao fato da ferramenta SIS gerar descrições com a metade de células lógicas e nets. Entretanto, as matrizes R-CAT OrBDD apresentam menor *wire length* médio, menor fan-out (redução de 15%), menor delay e maior roteabilidade. As sínteses OrBDD apresentam poucas nets não roteadas sem a inserção de trilhas extras.

Além disso, as matrizes R-CAT atingiram resultados até 40% menores em *wire length* e reduções de área de até 46% em relação às matrizes MARTELO.

Palavras-Chave: microeletrônica, ferramentas de CAD, síntese física, geração de leiaute, regularidade, previsibilidade.

Regular Layout Generation based on Cell Matrices

ABSTRACT

This work presents a research for physical synthesis of integrated circuits, which are less susceptible to the effects of variability observed in fabrication technologies using nanometers scale. Moreover, it presents a CAD tool developed to generate regular layouts, which is called R-CAT. The geometric regularity is achieved using basic patterns repeated along one matrix structure. Regularity is pointed like one of the best alternatives to deal with submicron technologies issues. Regular designs are less susceptible to lithographic problems, improve the yield and decrease the time to re-spin. Furthermore, regular circuits improve predictability of power consumption, timing and yield results, because the cells are pre-characterized.

The developed tool focuses on two types of physical synthesis for regular layouts, producing either integrated circuit customized using all masks or integrated circuits customized using some masks. The main goal is the facility of conversion and adaptation depending on the chosen matrix approach. This will make easier the comparison of different matrix approaches, besides the adoption of several logic blocks and new technologies.

R-CAT layout generator identifies adjacent cells that are placed in a same row and have common connections between them. In this case, the generator can make these connections in Metal 1. This technique reduces the number of connections to be done by the router. The experiments showed that this technique is able to reduce about 10% the number of connections to be done.

This tool is inserted into a design flow and it is dependent of the logic synthesis methodology adopted. Two logical syntheses tools were used in the flow: SIS and OrBDDs. R-CAT SIS and R-CAT orBDD Matrices were generated for a set of circuits. The use of R-CAT tool with SIS and orBDD logical synthesis offers two design lines: the first one highlights area and the second one emphasize timing and short connections. Both of them respect the same geometric regularity.

The results demonstrate that SIS matrices present 53% less area than orBDD approach and reduce the wire length by 30%. The area reduction is achieved because the SIS tool generates descriptions with the half of logic cells and nets. Nevertheless, the R-CAT orBDD matrices decreased the medium wire length, reduced the fan-out in 15%, reduced the delay and improved the routability. orBDD synthesis presents few non-routed nets without extra tracks insertion.

Moreover, the R-CAT matrices obtained about 40% better results in wire length and they reduced area in 46% when compared to MARTELO matrices.

Keywords: microelectronics, CAD tools, physical synthesis, leiaute generation, regularity, predictability.

1 INTRODUÇÃO

Atualmente, circuitos integrados VLSI usando nanotecnologia requisitam novas metodologias de projeto e ferramentas de CAD para lidar com os problemas decorrentes do processo de fabricação, tais como variabilidade, eletromigração, efeito antena e maus contatos (ZAHIRI, 2003). Tais problemas já eram percebidos nas tecnologias maiores, entretanto de modo amenizado. Os projetos de circuitos de aplicação específica (ASICs) podem acabar tendo custos elevados, principalmente devido às muitas etapas de re-projeto (*non-recurring engineering* – NRE), até atingir o objetivo esperado em funcionamento, potência e freqüência. O baixo *yield* obtido, causa direta do aumento do custo, é agravado pelo aumento de defeitos provocados por falhas de fabricação e por problemas de variabilidade de processo, tais como os decorrentes do processo de litografia *subwavelength* (BOLSEN, 2003).

Os projetistas de hardware para tecnologias submicrônicas devem atender demandas de performance, potência, custo, manufaturabilidade e *time-to-market*. Muitos projetos focam a otimização da manufaturabilidade utilizando técnicas de DFM (*design for manufaturability*). A utilização de leiautes regulares é uma técnica de DFM proposta para lidar com os problemas de projetos submicrônicos (GUPTA, 2003).

Circuitos integrados que exploram a regularidade através da adoção de padrões repetidos podem ser mapeados para as máscaras de fabricação com maior precisão e com menor probabilidade de gerar erros, facilitando o processo de fabricação além de aumentar o controle e a previsibilidade dos resultados. A regularidade auxilia na obtenção de melhores estimativas do atraso dos circuitos, especialmente dos atrasos devidos a interconexões. Leiautes regulares, com a utilização de repetição de padrões simples, diminuem o número de etapas de re-projeto e necessitam de um número menor de iterações entre as etapas de síntese (SHERLEKAR, 2004).

Historicamente, as metodologias de projeto *Standard Cell, Gate Array, Field Programmable Gate Array* (FPGA) e *Structured ASIC* apresentam diferentes modos de explorar a regularidade, sendo importante considerar que cada uma delas apresenta vantagens em relação à área, atraso, potência, previsibilidade ou *time-to-market*, quando aplicadas em projetos específicos. Deste modo, a utilização de circuitos regulares apresenta em uma metodologia de projeto que visa obter uma maior previsão de resultados sem sacrificar demasiadamente o desempenho e/ou a área, ainda concedendo ao projetista melhores estimativas de desempenho e consumo de energia dos circuitos integrados produzidos (ZAHIRI, 2003).

Nas tecnologias atuais torna-se cada vez mais importante realizar o projeto de ASICs considerando os problemas encontrados durante a fabricação e tentando adotar políticas de geração de leiautes que minimizem os efeitos danosos encontrados, assim como aumentem o *yield* dos projetos.

Na UFRGS existem projetos para geração de leiautes regulares anteriores a esta crescente demanda por manufaturabilidade decorrente da nano escala de fabricação. Tais projetos apresentaram idéias precursoras no modo de geração de leiaute, sendo muitas delas encontradas atualmente em metodologias comerciais para fabricação de circuitos mais previsíveis e mais competitivos quanto ao *time-to-market*. São projetos desenvolvidos neste propósito o Ágata (CARRO, 1996), o Maragata (LIMA, 1999), o Marcela (GÜNTZEL, 1993) e o Martelo (MENEZES, 2004). Através de suas experiências, alternativas exploradas e restrições adotadas, estes trabalhos contribuíram para a implementação de uma ferramenta capaz de reunir algumas características dos projetos anteriores com novas idéias e conceitos focando a geração de leiaute regular, onde a regularidade é o centro das escolhas realizadas. Explorando a regularidade nas formas de leiaute utilizadas, na repetição de padrões e na distribuição das *nets* pelo circuito, espera-se aumentar o *yield*, assim como, aumentar a previsibilidade dos projetos, uma vez que as células básicas estejam devidamente caracterizadas.

Neste trabalho são apresentadas alternativas de utilização de conceitos de regularidade na geração de leiautes e distribuição das conexões dentro de um fluxo de síntese de circuitos integrados ou blocos funcionais. A estratégia adotada para regularidade de leiaute é a utilização de uma matriz de células. O leiaute é construído com a repetição dos blocos básicos que implementam funções lógicas simples. Estes blocos básicos podem ser um único tipo de célula, um conjunto de células básicas ou um conjunto de células lógicas programáveis. Deste modo, será disponibilizada uma ferramenta, inserida em um fluxo de síntese regular, na qual serão preservados os aspectos de regularidade em todas as etapas e será possível utilizar diferentes tipos de bloco básicos, em diferentes tecnologias, dependendo das necessidades e características do projeto. Assim, busca-se aumentar a previsibilidade dos resultados, bem como o *yield*. Tamanha liberdade na concepção do bloco básico amplia o espaço para a pesquisa e avaliação das propriedades das células adotadas, aumentando as possibilidades de implementações de matrizes e, principalmente, permitindo resgatar pesquisas e metodologias anteriormente desenvolvidas para a geração de matrizes pré-difundidas.

Atualmente são exploradas duas alternativas para geração do leiaute: o projeto de circuitos usando matrizes de células pré-difundidas (estilo *Structured ASIC*) com personalização pelas camadas de projeto restantes e o projeto de matrizes totalmente personalizáveis, geradas dinamicamente. Ambas abordagens têm vantagens e desvantagens relacionadas às necessidades do projeto.

A utilização de matrizes pré-difundidas reduz o tempo de fabricação além de apresentar maior previsibilidade, considerando a verificação das camadas já difundidas. A personalização define a lógica do circuito e pode ser realizada por camadas superiores de metal, dependendo da disponibilidade de camadas fornecida pela tecnologia adotada. Tal modo de projeto pode ser utilizado para a fabricação de circuitos no CEITEC, onde pode auxiliar na redução dos custos de fabricação de outros circuitos utilizando espaços disponíveis nos *wafers* empregados em outros projetos, visto que as matrizes são pré-fabricadas. Outra possibilidade a ser estudada é pré-difundir matrizes em tecnologias menores que as disponibilizadas no CEITEC e realizar a personalização com os equipamentos disponíveis no CEITEC. Todavia, esta possibilidade engloba um grande número de fatores a serem futuramente pesquisados para realizar a compatibilidade entre as camadas de leiaute dos circuitos.

Entretanto, matrizes pré-difundidas implicam em limitações de número de células e diminuem a liberdade de utilização de diferentes células básicas. As matrizes totalmente

personalizáveis geradas dinamicamente atendem a demanda de células obtidas pelas etapas anteriores do fluxo de síntese e embora aumentem o tempo de produção, podem apresentar um grau compatível de previsibilidade se as células utilizadas já houverem sido fabricadas anteriormente.

1.1 Objetivo do trabalho

O objetivo desta pesquisa é fornecer uma ferramenta para geração de leiautes regulares compostos pela repetição de blocos básicos em formato matricial, inserida em um fluxo de síntese lógica e física voltado para atender às necessidades de regularidade. A ferramenta deve estar apta a operar tanto na geração de matrizes pré-difundidas configuráveis pelas camadas superiores de metal como na geração de matrizes totalmente personalizáveis dinamicamente.

O estudo se aprofunda em matrizes compostas pelas portas lógicas Inversor, NAND e NOR. Entretanto, é desenvolvido com o propósito de permitir a utilização de outras células básicas. Para isso, as células básicas compõem uma biblioteca de células. Assim, procura-se aumentar as possibilidades de matrizes que possam ser geradas, além de facilitar a utilização de diferentes tecnologias no leiaute das células básicas. Os objetivos principais deste trabalho são:

1- Construir uma ferramenta para a geração de matrizes compostas por células básicas lidas de bibliotecas de células;

2- Disponibilizar uma versão da ferramenta específica para a geração de matrizes compostas por células NAND, NOR e Inversores;

3- Observar a influência no leiaute e no roteamento do fornecimento de múltiplas opções de posição para os pinos de entrada e saída das células.

O terceiro objetivo decorre da observação dos resultados obtidos com a ferramenta Martelo. Na maioria dos circuitos observou-se um alto grau de nets não roteadas, quando não era permitida a inserção de trilhas extras verticais ou horizontais[1]. Uma das possíveis razões seria a disposição interna dos pinos de entrada e saída nas células. A célula adotada no projeto Martelo apresenta estes pinos muito próximos uns dos outros, podendo fornecer obstáculos para o roteador. Decidiu-se adotar diferentes posições dos pinos nas células básicas e observar se alguma das posições favorece o roteador.

1.2 Fluxo de Síntese

A ferramenta de geração do leiaute estruturado no formato de matriz está inserida num fluxo de síntese dedicado desde a etapa de síntese lógica até roteamento. O fluxo, mostrado na Figura 1.1 e melhor detalhado na Seção 5.3, inicia com uma etapa de síntese lógica específica para a síntese de matrizes compostas por portas lógicas simples. Em seguida, o pré-posicionamento posiciona as instâncias segundo a sua posição funcional no circuito. Este pré-posicionamento é refinado na etapa de posicionamento, onde as células são deslocadas mantendo características definidas na etapa de pré-posicionamento. A etapa de geração da matriz de células, foco deste trabalho, realiza a construção da matriz de acordo com o posicionamento fornecido e

[1] Trilha extra é uma área livre de tamanho igual a um passo de roteamento, inserida entre as células da matriz para auxiliar o roteador a finalizar o roteamento do circuito.

utilizando os leiautes das células básicas previamente geradas. Finalmente, as células são roteadas objetivando o menor custo de fiação possível. A ferramenta de roteamento suporta vários níveis de metal.

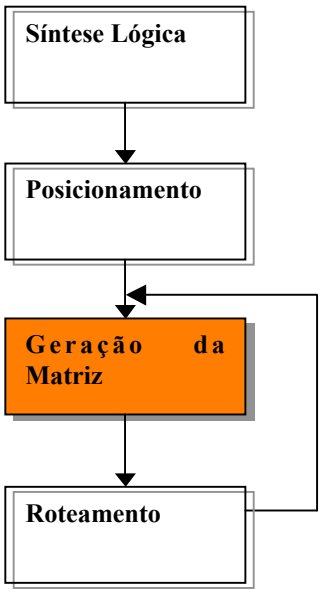


Figura 1.1: Fluxo de síntese adotado

1.3 Organização da Dissertação

Esta dissertação é composta de 8 capítulos. O segundo capítulo traça o panorama atual da fabricação de circuitos integrados, apresentando os problemas encontrados nas etapas de fabricação das tecnologicas sub-micrônicas. Além disso, mostra como a regularidade vem sendo utilizada como uma alternativa para minimizar estes problemas, principalmente, como meio para elevar o *yield.*

No Capítulo 3 são definidos conceitos sobre regularidade, assim como conceitos adotados para a caracterização de blocos básicos e de características de projetos regulares.

O Capitulo 4 reúne algumas das principais metodologias historicamente propostas focando a regularidade, procurando, mais do que apresentar as principais características de cada metodologia, destacar como elas exploram os conceitos de regularidade. Uma ênfase maior é dada a metodologia Structured ASIC e as ferramentas desenvolvidas na UFRGS.

No Capítulo 5 são descritas as ferramentas que compõem o fluxo de síntese adotado, inclusive a ferramenta para a geração de leiaute desenvolvida no presente trabalho. Esta ferramenta recebe o nome de R-CAT (*Regular – Cell Array synthesis Tool*). Neste capítulo são discutidas as opções escolhidas na implementação da ferramenta, bem como as características e restrições impostas na sua concepção. Ainda no capítulo 5, apresenta-se o fluxo de geração de leiautes regulares, detalhando as etapas internas da geração de leiaute, além de exemplificar as opções de geração disponibilizadas.

No Capitulo 6 são descritos os experimentos realizados para avaliar a influência no posicionamento interno dos pinos de entrada e saída das células básicas no roteamento do circuito.

O Capítulo 7 apresenta um conjunto de resultados. Inicialmente, são apresentadas as opções de síntese lógica estudadas para utilização em conjunto com a ferramenta de geração da matriz R-CAT. Em seguida, são mostrados os resultados de síntese física obtidos, comparando eles com a ferramenta Martelo e com as ferramentas de síntese da Cadence.

Finalmente, Capítulo 8 apresenta as conclusões e os trabalhos futuros.

2 PROBLEMAS EMERGENTES EM TECNOLOGIAS SUBMICRÔNICAS

A fabricação de circuitos em tecnologias sub-micrônicas altera o projeto físico dos circuitos, impõe regras de leiaute complexas e exige a adoção de uma ampla faixa de segurança na concepção dos blocos para lidar com a variabilidade do processo. A grande redução na escala de tamanho dos transistores introduz novas fontes de variação e torna o controle da variabilidade complicado. A variabilidade já afeta a geração de circuitos nas tecnologias de 180 nm e 130 nm. Estima-se que nas futuras tecnologias a variabilidade do processo aumente, causando a diminuição da previsibilidade de desempenho dos circuitos nanométricos, afetando diretamente a operação do circuito (GUPTA, 2003). Por exemplo, a corrente de fuga (*leakage*) tem dependência exponencial com o tamanho de *gate,* portanto a variância em L tem impacto exponencial na corrente de fuga. A Figura 2.1 mostra uma variação de 30% na freqüência de operação de circuitos de um mesmo projeto e de 20x a corrente de fuga, observadas após a fabricação de um projeto da Intel na tecnologia de 180 nm (BORKAR, 2003).

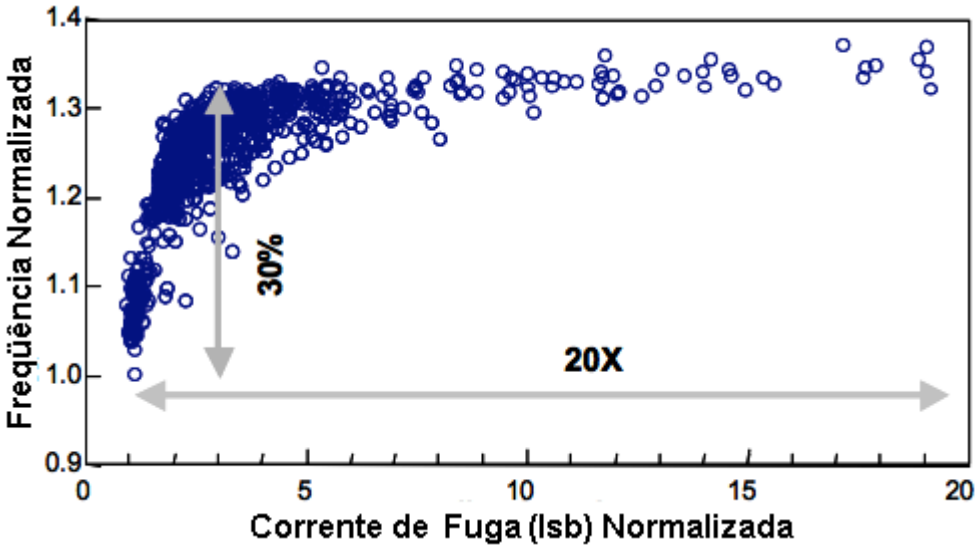


Figura 2.1: Variações na freqüência e na corrente de fuga observadas em um mesmo projeto (BORKAR, 2003)

Além disso, os projetos devem lidar com os efeitos parasitas, tais como eletromigração e ruído. Nos projetos em tecnologias nanométricas está mais difícil minimizar os efeitos parasitas (FERGUNSON, 2004). Estes problemas já existiam nas tecnologias de 0,35 μm e 0,25μm, entretanto nestas tecnologias, o projetista podia priorizar um dos problemas e minimizar os efeitos dos outros sem danificar o

funcionamento do projeto. Estudos afirmam que mais da metade dos projetos utilizando nanotecnologias não funciona diretamente devido às falhas funcionais relacionadas ao processo de fabricação (DIPERT, 2004). Este fato eleva o custo do projeto devido à necessidade de muitas etapas de re-projeto (NRE).

Segundo Sherlekar (2004), o número de vezes que um chip precisa ser alterado antes de atingir volume de produção é em torno de 4 ou 6 vezes, crescendo para projetos que empregam as novas tecnologias. São motivos para re-projeto: os erros na especificação funcional do ASIC, defeitos que não podem ser detectados antes de um processo de fabricação, revisões na especificação funcional para responder a características levantadas durante a prototipação, testes ou mudanças nos padrões ou protocolos usados no ASIC e problemas relacionados com o tamanho sub-micrônico, tais como ruído ou *crosstalk*.

Sobretudo, permanecem as demandas de desempenho, potência, custo, manufaturabilidade e *time-to-market*. Os atuais projetos focam a otimização do desempenho (*design for performance* - DFP) ou a otimização da manufaturabilidade (*design for manufacturability* – DFM). Em DFPs, as variações do processo interferem na distribuição de desempenho do circuito, implicando em metodologias de projeto para valor (*design for value* – DFV) onde o *yield* é maximizado. Alguns pesquisadores afirmam a necessidade de existir um caminho bidirecional de fabricação entre projetistas, ferramentas de CAD e a fabricação, onde o custo e o valor sejam os principais guias do processo em direção ao funcionamento do projeto, principalmente quando construído em tecnologias sub-micrônicas. (GUPTA, 2003).

Em DFMs, os projetos utilizam técnicas para elevar o *yield* do projeto, tais como análise de regras de DFM, testes físicos, projeto voltado à litografia (*Litho-friendly design*) e modelagem de silício nano métrico (*Nanometer silicon modeling*) (SAWICKI, 2005). Com a elevação do número de defeitos dos projetos, as indústrias começam a disponibilizar ferramentas que simulam os problemas ocorridos na litografia e sugerem modificações para tornar o leiaute mais manufaturável. As próprias indústrias estão disponibilizando regras de projeto recomendáveis para DFM (*DFM recommended*), que, quando adotadas, indicam quando um projeto torna-se fácil para a fabricação. Entretanto, essas regras exigem dos projetistas abandonar a busca pelos tamanhos mínimos, fornecidos pelas regras de DRC (*Design Rule Checking*) como os limites impostos pelo processo de fabricação. Ou seja, passar pelo DRC não é mais uma garantia de que o leiaute final saíra sem defeitos derivados do desenho do circuito.

Durante o processo de fabricação de circuitos podem ocorrer defeitos, sendo os três tipos principais de defeitos responsáveis pela redução do *yield* (ABERCROMBIE, 2005), descritos a seguir:

- Defeitos randômicos: Os defeitos randômicos são provocados por partículas de impurezas que se depositam durante o processo de fabricação no chip ou nas máscaras provocando curto-circuitos ou circuitos abertos. A Figura 2.2 apresenta exemplos destes defeitos provocados pela presença de partículas de impureza, ocasionando curto-circuitos, pontos de materiais indesejados e interrupções nas áreas projetadas. Esses tipos de defeitos eram as principais causas dos defeitos encontrados em tecnologias antigas. Com a redução de escala a taxa de defeitos randômicos permaneceu constante, mesmo com as melhorias nas qualificações das salas limpas.

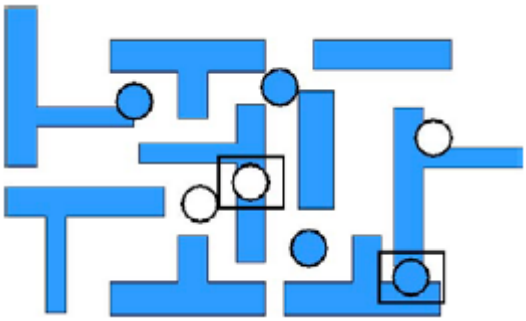


Figura 2.2: Exemplos de defeitos randômicos provocados por partículas de impurezas (ABERCROMBIE, 2005)

- Defeitos paramétricos: Os defeitos paramétricos ocorrem quando todos elementos de um chip estão funcionando corretamente, entretanto ele não corresponde aos requisitos da especificação do projeto, como *timing* ou potência. Este tipo de defeito é o resultado de interconexões parasitas e cresce com a redução de escala das tecnologias.
- Defeitos sistemáticos: o defeito sistemático está relacionado com o leiaute e as variações no processo de fabricação, tais como as variações mecânicas, litográficas e do plasma na corrosão. São exemplos de defeitos sistemáticos a diferença entre as alturas de um metal entre regiões do chip, efeito conhecido como *planarity* (SAWICKI, 2005) ou variações *in-die* (ROBERTSON, 2003), o acúmulo de carga entre componentes de interconexão durante a fabricação (efeito antena), vias abertas e a eletromigração. Os defeitos sistemáticos podem ser compensados com o uso de técnicas de RET (*Resolution Enhance Technique*), entretanto o leiaute projetado pode limitar a aplicação das principais técnicas de RET, principalmente quando são utilizadas as mínimas distâncias possíveis.

Além dos defeitos decorrentes do processo de fabricação, as variações no processo de fabricação das pastilhas podem ser divididas em dois grupos: a variação entre pastilhas de um mesmo projeto e as variações internas nas pastilhas. A variação entre pastilhas é a diferença no valor de um parâmetro entre pastilhas idênticas. Estas diferenças podem ser de chip para chip, de *wafer* para *wafer* ou de lote para lote. Está principalmente relacionada às propriedades dos equipamentos, ao posicionamento do *wafer* e à temperatura do processo.

A variação interna nas pastilhas pode ocorrer no *wafer* ou ser dependente dos padrões de leiaute. Por exemplo, as flutuações na dopagem do canal ou na espessura do óxido do *gate*. Este tipo de variação tem ligação com a posição da pastilha no *wafer*, sendo pastilhas vizinhas mais similares que pastilhas distantes. Esses dois tipos de variação apresentam componentes de defeitos randômicos e sistemáticos (GUPTA, 2003).

De acordo com Pileggi (2003), os avanços na correção ótica para as etapas de litografia, assim como os avanços nas técnicas de teste, não são suficientes para lidar com os desafios de printabilidade das tecnologias sub-micrônicas. Uma opção seria a utilização de projetos baseados em um conjunto de regras de leiaute onde todos os aspectos do leiaute são posicionados em um *grid*. Os pontos fora deste *grid* serão eliminados, sendo proibidos padrões de leiaute difíceis de serem desenhados nas

camadas de fabricação. Porém, isso implica em alto custo de re-projeto de todas as células de bibliotecas usadas em projetos *Standard Cell* e *IP cores* de acordo com as novas restrições impostas pelas regras de projeto recomendáveis para DFM fornecidas pelas *foundries*. Como alternativa para lidar com os desafios e atingir os objetivos de desempenho e *yield* pode-se explorar os conceitos de regularidade nos projetos. Projetos regulares têm conquistado espaço no mercado, sendo adotados por empresas tais como: NEC (2005), eASIC (2005b), ViASIC (2004a), Faraday (2005), Virage (SHERLEKAR, 2004) e LSI Logic RapidChip (2004), além de pesquisas desenvolvidas pela academia: VPGA (PILEGGI, 2003), River-PLA (MO, 2002), VCC (RAN 2004) e PPL (GU, 1989).

Nas próximas seções são caracterizados alguns dos pontos críticos do processo de fabricação em tecnologias nanométricas, salientando os problemas decorrentes das etapas do processo de fabricação, em especial, aqueles relacionados à litografia. Assim como, são expostas as dificuldades que afetam as interconexões do circuito. Muitas pesquisas indicam a necessidade de adotar formas de regularidade em todas as etapas da síntese para auxiliar na redução do impacto dos defeitos sistemáticos nos projetos. Existem divergências quanto à definição de regularidade em projeto de circuitos, portanto, o próximo capítulo fornece definições para os conceitos de regularidade adotados neste trabalho e utilizados na comparação entre os demais trabalhos na área.

2.1 Litografia e Processo de Fabricação

As melhorias nas técnicas de litografia ótica tornaram possível, ao longo dos anos, diminuir o tamanho dos componentes e características elétricas dos dispositivos, assim como, aumentar a integração e a densidade de transistores. A etapa de litografia abrange todos os passos envolvidos na transferência de um padrão especificado numa máscara para a superfície do *wafer*. Para produzir um circuito integrado, filmes finos de vários materiais são usados como barreiras para a difusão e implantação de átomos ou como isolantes entre os materiais condutores e o substrato. O processo litográfico remove padrões destes materiais respeitando as especificações projetadas nas máscaras de fabricação (JAEGER, 1993).

Entretanto, nas novas tecnologias os tamanhos usados nos desenhos das camadas do projeto, por exemplo, nas larguras das linhas, estão ficando menores que o tamanho da onda utilizada durante a litografia. A Figura 2.3 ilustra a redução da escala de processo de fabricação ao longo do tempo e o tamanho de onda empregado no processo de litografia. A utilização de ondas com tamanhos maiores que a escala de projeto dificulta o processo de litografia. Até a tecnologia de fabricação de 0.35 μm, o tamanho de onda utilizado na litografia era maior que o tamanho característico do silício na tecnologia empregada. Entretanto, nas tecnologias atuais o tamanho da onda utilizado no processo de litografia é maior que as dimensões mínimas das linhas a serem desenhadas. Por exemplo, o processo CMOS 130nm usa ferramentas de 193nm operando num regime de litografia inferior ao tamanho da onda, conhecido como litografia *subwavelength*. A litografia *subwavelength* agrava os problemas de variação (KARNIK, 2002). Os lasers de 193nm são usados para fabricar chips com dimensões de 90nm ou menores.

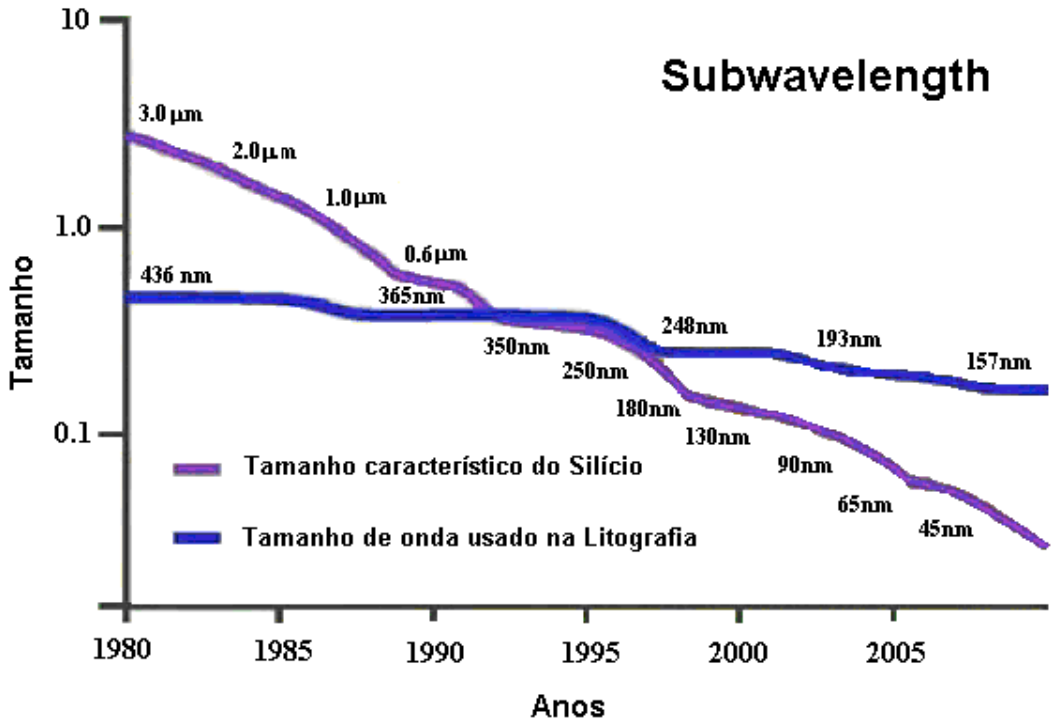


Figura 2.3: Relação entre o tamanho da onda usada na litografia com a redução de escala de processo (SYNOPSYS, 2005)

A Figura 2.4 demonstra os efeitos indesejáveis da variação ótica no processo de litografia. Note que as linhas totalmente preenchidas do leiaute desejado dificilmente são obtidas. Os resultados são linhas com cantos arredondados, imperfeições no traçado e, principalmente, pontos críticos onde foram aplicadas as larguras mínimas permitidas. Da mesma forma que no ponto crítico destacado na figura, onde quase não há deposição de material suficiente para manter a continuidade da conexão.

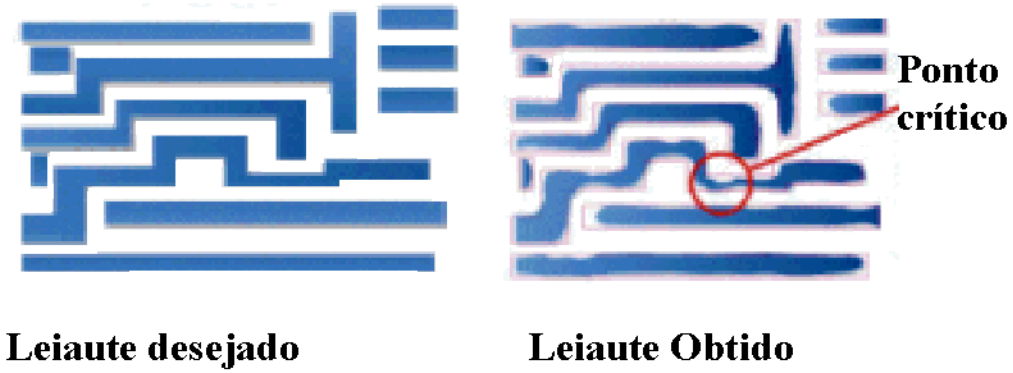


Figura 2.4: Diferenças entre o leiaute desejado e o obtido após a litografia (ROBERTSON, 2003)

Para auxiliar o processo de fabricação, as técnicas de RET são aplicadas nos três principais componentes da onda: direção, amplitude e fase. Os resultados obtidos através da aplicação de duas técnicas de RET em tecnologias atuais pode ser observado na Figura 2.5. Nesse exemplo são utilizadas duas técnicas: a *Optical Proximity Correction* (OPC) e a *Phase Shift Mask* (PSM) (REINHARDT, 2002). A técnica OPC insere retângulos em pontos estratégicos para forçar o resultado esperado. A técnica

PSM realiza um ajuste na angulação inicial da onda de litografia (OR BACH, 2004). Em tecnologias de 250 nm não eram necessárias técnicas de RET. Entretanto, pode-se observar a degeneração das formas projetadas na tecnologia de 180nm após a litografia sem a aplicação da técnica de OPC.

Na tecnologia de 90nm a imprecisão do processo de litografia pode resultar na ausência de áreas projetadas no *wafer* se nenhuma técnica de RET for adotada. Para esta tecnologia são aplicadas as técnicas de PSM e OPC para atingir resultados confiáveis. Além disso, no caso de um projeto onde nenhuma das técnicas é empregada, algumas linhas não são desenhadas. Entretanto, mesmo com estas técnicas é impossível otimizar o resultado da litografia para todos os pontos do leiaute, implicando em alguns pontos onde podem continuar ocorrendo grandes erros. Como exemplo, um dos problemas observados é que alguns pontos intermediários podem não ser desenhados, existindo então no circuito ausências de materiais ou regiões com tamanho inferior ao especificado no projeto (GUPTA, 2003).

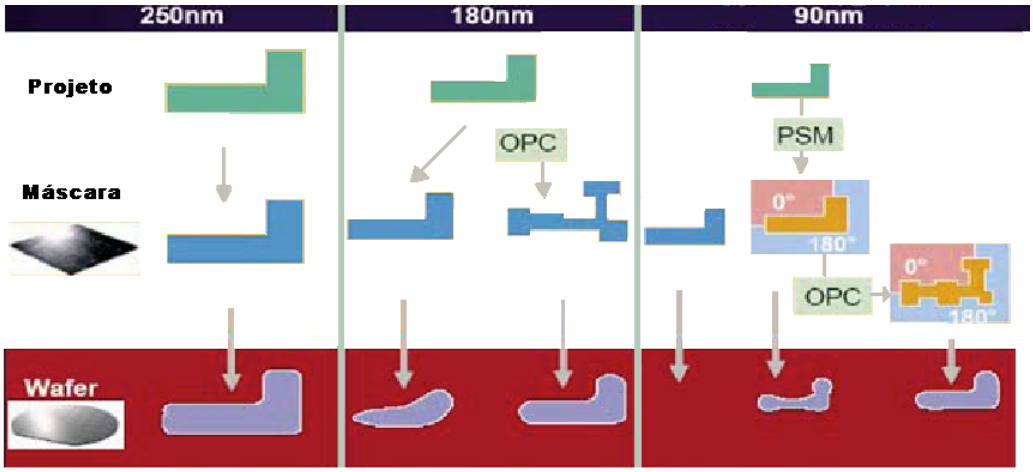


Figura 2.5: Resultados do processo de litografia nas tecnologias de 250, 180 e 90 nm e da aplicação de técnicas de RET para tecnologias de 180 e 90 nm (Or-BACH, 2004)

O uso das técnicas de RET está entre as principais causas de aumento do custo das máscaras e do tempo para fabricação. A Figura 2.6 ilustra o impacto no custo do processo de fabricação provocado pelas etapas do processo. A escrita ótica engloba a litografia e é a principal causa do aumento do custo devido às técnicas de RET e o tempo superior de escrita das máscaras. Somam aos custos, o crescente número de etapas de re-projeto, o baixo *yield* das máscaras, aumento do tempo de preparação dos dados e custo do equipamento (GUPTA, 2003).

As técnicas de OPC somente podem ser aplicadas quando o projeto inicial permite a inserção de regiões extras na preparação da máscara. Quando o projeto adota as regras mínimas na elaboração do leiaute, não há condições de aplicar OPC sem comprometer o DRC do projeto. Novamente, salienta-se a adoção das regras para manufaturabilidade, sempre que estas forem fornecidas para a tecnologia adotada, ou então o não uso das dimensões mínimas na concepção dos leiautes (SAWICKI, 2005).

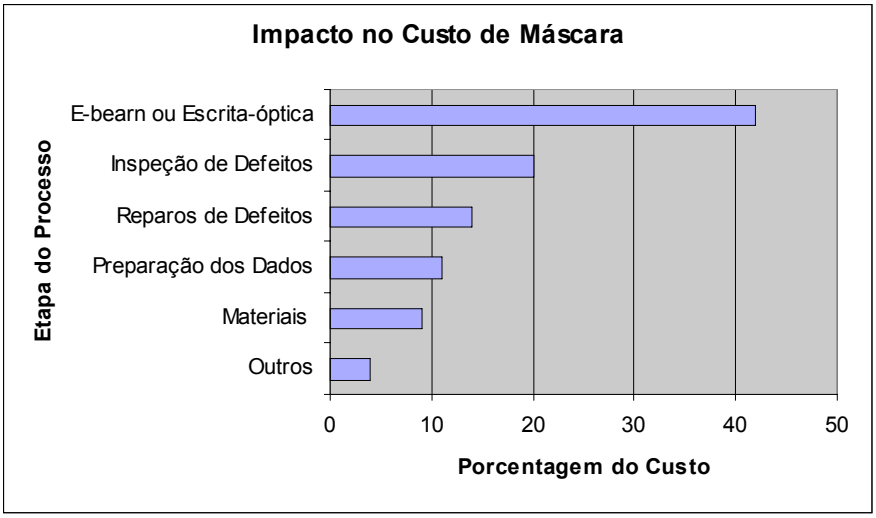


Figura 2.6: Impacto das etapas do processo no custo de fabricação (GUPTA, 2003)

A Figura 2.7 sugere como projetar um leiaute pensando nas inserções realizadas durante a etapa de OPC. Na Figura 2.7 a) são apresentadas as regras de projeto para as dimensões mínimas entre as camadas de poly e difusão. A Figura 2.7 b) exemplifica um detalhe de leiaute onde as dimensões críticas impedem a inserção de reforços nas extremidades e dobras da linha de *poly* sem uma violação de DRC. Na Figura 2.7 c) são expostas regras de distância entre as camadas para um projeto voltado para a manufaturabilidade. Nota-se que comparado com a Figura 2.7 a) as distâncias adotadas como mínimas são maiores. Já na Figura 2.7 d) podemos ver a aplicação de OPC no leiaute adotando as regras de manufaturabilidade ao invés das regras de mínima distância. Karnik (2002), Gupta (2003) e Or Bach (2004) destacam a necessidade de estilos de leiautes regulares, como *sea-of-gates* ou *sea-of-transistores* como alternativa para minimizar os problemas de litografia.

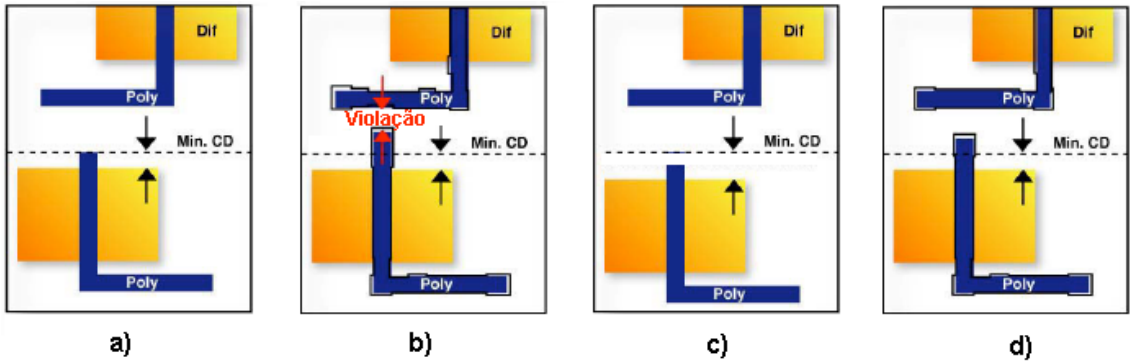


Figura 2.7: Exemplo de leiaute apropriado para a aplicação da técnica OPC (ABERCROMBIE, 2005)

2.2 Interconexões

Com o avanço das tecnologias, são disponibilizadas mais camadas de metal, aumentando o número possível de camadas aplicadas para a realização das interconexões. Atualmente é comum tecnologias com mais de oito camadas de metal, sendo em torno de quatro ou cinco camadas destinadas para a realização das interconexões. As camadas de metal normalmente são as últimas a serem processadas na fabricação de circuitos integrados. Durante o processo, a interconexão de uma *net* é compostas de vários segmentos desconexos de metal. Problemas na fabricação destes segmentos, assim como o fato de permanecerem desconectados durante os estágios intermediários da fabricação, podem fazer com que alguns destes fios atuem como antenas, captando cargas elétricas durante o funcionamento do circuito. Este efeito é conhecido como efeito antena e tem seu efeito agravado nos fios longos, pois estes podem receber maior quantidade de carga. Uma solução apresentada por Otten (2002) é evitar o efeito antena adotando roteadores globais que escolham a opção menos provável de sofrer o efeito antena para os fios mais longos do projeto, uma vez que os caminhos longos tipicamente apresentam opções de implementação na etapa de roteamento global.

No entanto, o efeito antena não é o único problema que surge relacionado às interconexões. Em muitos projetos utilizando tecnologias sub-micrônicas, o atraso devido às interconexões não pode mais ser ignorado por estar se tornando maior que o atraso das portas (CONG, 2001). Os fios mais longos influenciam diretamente o atraso do circuito, uma vez que o caminho crítico determinará o atraso e a freqüência de um circuito. Se o posicionador conhece estes caminhos críticos, ele pode forçar a adoção de um caminho com menor tamanho de fio. Entretanto, essa prioridade para algumas *nets* pode tornar *nets* não críticas em inesperadas *nets* críticas. Otten (2002) utiliza a Figura 2.8 para demonstrar a relação entre o atraso de uma conexão de 43µm e o atraso devido a um *gate*. Nela pode-se observar que o atraso devido às conexões vem se destacando como o principal fator no atraso do circuito para as tecnologias sub 180nm. Otten também salienta a importância de um fluxo de síntese onde o posicionamento e a síntese lógica trabalhem no planejamento das interconexões. Assim, as estimativas geradas nos estágios iniciais da síntese poderão ser mais precisas.

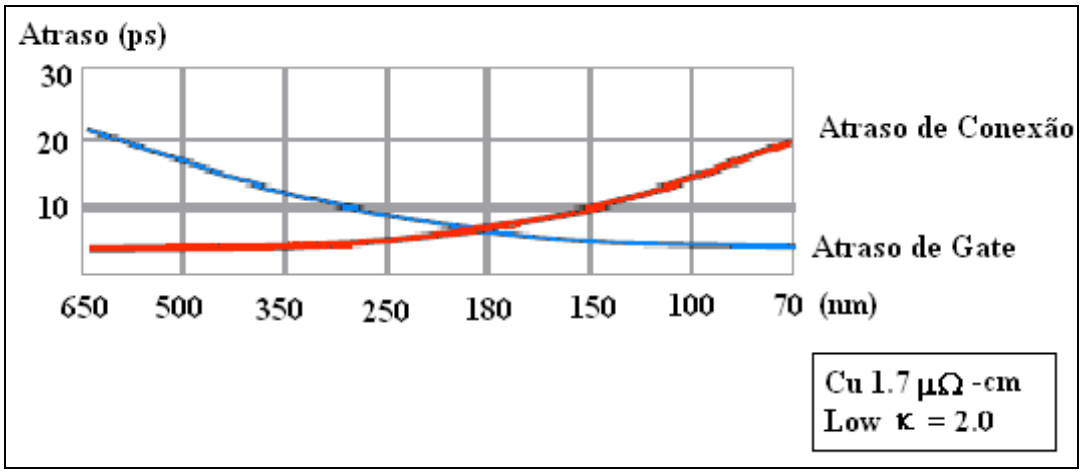


Figura 2.8: Atraso devido a conexões versus atraso devido aos *gates* (OTTEN, 2002)

Além de interconexões longas tornarem o circuito mais suscetível ao efeito antena e aumentarem o *timing* do circuito, o comprimento do fio aumenta os problemas relacionados à eletromigração. Experimentos demonstraram que a eletromigração torna-se um problema somente para fios maiores que um tamanho crítico, na ordem de 50μm para tecnologias de 180 nm (OTTEN, 2002).

Outro problema que afeta as conexões está relacionado com a densidade de metal depositado durante o processamento do circuito. As etapas de *Chemical Vapor Deposition* (CVD) e *Chemical-Mechanical Polishment* (CMP) produzem resultados diferentes dependendo da região do *wafer* (ROBERTSON, 2003). A Figura 2.9 mostra o impacto das variações *in-die* na tecnologia de 130nm. Nos processos anteriores, não existem variações na seção de metal (mostrado a esquerda). Em tecnologias nanométricas, a deposição de metal sofre variação influenciando diretamente a capacitância das conexões (figura central e à direita) (SAWICKI, 2005).

Figura 2.9: Exemplo dos efeitos das variações *in-die* (ABERCROMBIE, 2005)

3 CONCEITOS SOBRE ESTRUTURAS REGULARES

No capítulo anterior apontou-se a adoção de estruturas regulares como uma alternativa para lidar com os problemas de variabilidade nas etapas de processo, assim como com os problemas relacionados à litografia *subwavelength* e as interconexões. Contudo, regularidade é um conceito muito amplo e pode ser explorado de diversos modos dentro de projetos de circuitos integrados. Por esse motivo, as próximas subseções apresentarão algumas definições de conceitos adotados ao longo do trabalho. Esses conceitos serão fundamentais para a comparação entre metodologias de projeto voltadas para a regularidade e na definição das alternativas de regularidade exploradas no trabalho.

3.1 Regularidade Geométrica

A regularidade geométrica está relacionada à utilização de repetição de padrões geométricos na composição do leiaute. Além disso, busca-se minimizar o número de curvas dos fios, utilizando-se preferencialmente linhas retas. Esta regularidade é especialmente relevante para as máscaras de silício, polissilício e máscaras inferiores de metal que são caracterizadas pelas linhas mais finas dentre as utilizadas na geometria física do leiaute (PILEGGI, 2003).

3.2 Regularidade Lógica e de Roteamento

A regularidade lógica pode ser explorada na escolha dos componentes lógicos a serem utilizados durante o mapeamento tecnológico, no modo como é realizada a definição da função lógica no circuito ou na adoção de técnicas durante a síntese lógica que favoreçam a exploração de regularidade nas próximas etapas do fluxo de síntese (PILEGGI, 2003). Como exemplo, algumas implementações de síntese lógica adotam técnicas com o objetivo de garantir tamanhos de conexões menores durante o roteamento ou diminuir fan-out da nets. (TAVARES, 2005).

A regularidade de roteamento está relacionada ao modo como é construído o leiaute da rede de conexões. O roteamento pode explorar segmentos de conexões pré-definidas, sendo estes pré-fabricados ou não. Alguns projetos utilizam arquiteturas de roteamento pré-difundidas personalizadas pelas camadas de vias, como, por exemplo, VPGAs (PILEGGI, 2003) e VCGAs (RAN, 2004 a), apresentados na Seção 4.9.2. Nesses trabalhos, o roteamento é configurado por uma ou mais camadas de vias e os segmentos de metal que compõem as interconexões apresentam sempre o mesmo padrão de leiaute. O Strucured ASIC eASIC (eASIC, 2005a) restringe a configuração do roteamento a uma única via, garantindo um leiaute de conexões altamente regular, ou seja, toda a estrutura dos segmentos de metal das camadas 4 a 7 destes circuitos apresentara o mesmo padrão de leiaute.

3.3 Regularidade Estrutural

A regularidade estrutural significa a padronização de aspectos do projeto, tais como: o gerenciamento de restrições de projeto encontradas durante o processo de fabricação ou durante a síntese física. A estratégia de projeto para regularidade estrutural adquire inicialmente os requisitos físicos, restrições topológicas e regras de projeto e gera regras de regularidade para a construção dos circuitos. Estas regras definem os aspectos de regularidade geométrica, regularidade lógica e de roteamento a serem adotados nas etapas do fluxo de síntese (GU, 1989).

3.4 Ortogonalidade

A ortogonalidade é a independência funcional entre os menores componentes de lógica. A aplicação de máxima ortogonalidade entre os componentes garante que nenhum componente contém os outros, nem pode ser substituído por um outro único componente disponível (GU, 1989).

3.5 Granularidade

Conforme Gu (1989) a granularidade de um projeto é definida pela menor função lógica desenvolvida pelo menor componente lógico. A granularidade de um componente pode ser grossa, grande, média, pequena, ou fina, dependendo do tamanho da complexidade lógica contida no componente. A classificação do grau de granularidade pode ser definida como (ZAHIRI, 2003):

- Granularidade Fina: arquiteturas que adotam a granularidade fina utilizam como componentes elementos básicos como transistores e resistores, conforme ilustra a Figura 3.1. Este grau de granularidade é adotado no estilo de projeto Gate Matrix (Seção 4.2), Linear Matrix (Seção 4.3), Gate Array (Seção 4.4) e em FPGAs da empresa ATMEL (2005). As camadas de metal realizam as conexões destes componentes obedecendo a configurações pré-definidas.

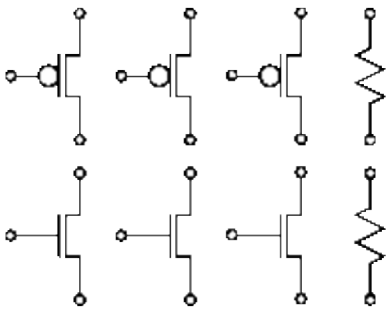


Figura 3.1: Exemplo de granularidade fina (ZAHIRI, 2003)

- Granularidade Pequena: projetos com granularidade pequena utilizam um pequeno conjunto de componentes responsáveis por funções lógicas básicas. A Figura 3.2 apresenta componentes comumente utilizados nestes projetos, tais como células inversores, NAND, NOR e *buffers*. A granularidade

pequena é explorada em estilos de projeto como Weinberger-Array (Seção 4.1) e Gate Array (Seção 4.4).

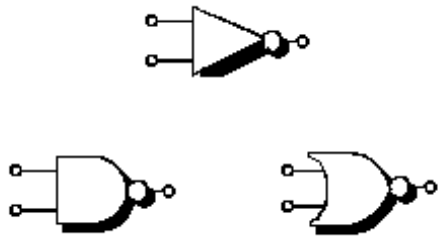


Figura 3.2: Exemplo de granularidade pequena (ZAHIRI, 2003)

- Granularidade Média: neste tipo de granularidade, os projetos podem adotar componentes compostos por células lógicas genéricas na forma de portas ou multiplexadores, junto com um ou dois flip-flops. A granularidade média é adotada em projetos conhecidos como Macrocell Arrays (Seção 4.4). Alternativamente às portas e multiplexadores, podem ser utilizadas *lookup tables* (LUTs) com flip-flops. Esta última alternativa é utilizada em blocos básicos de alguns FPGAs (Seção 4.7). A Figura 3.3 apresenta exemplos de portas que podem compor um projeto com granularidade média, podendo apresentar blocos compostos por multiplexadores ou por portas lógicas e multiplexadores ou ainda por LUTs, sempre em conjunto com flip-flops.

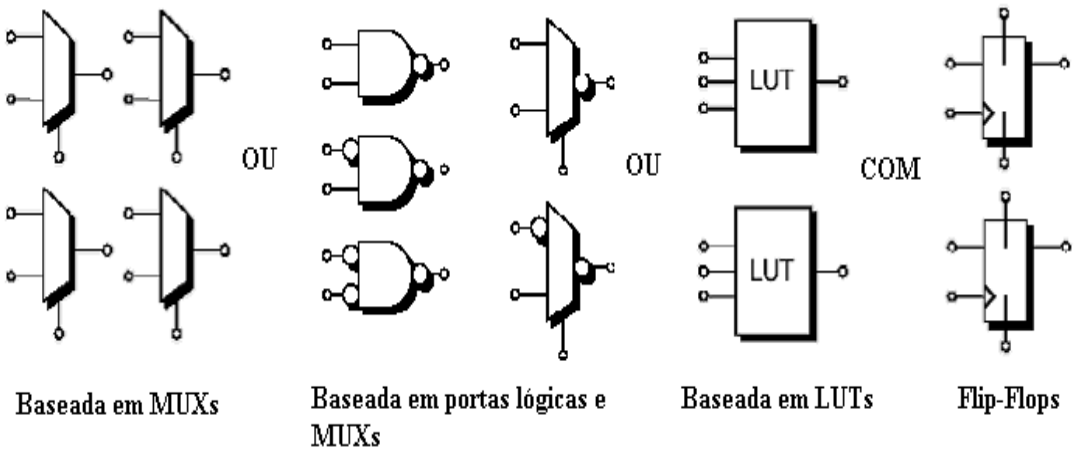


Figura 3.3: Exemplo de granularidade média

- Granularidade Grande e Grossa: projetos adotam granularidades maiores com o objetivo de minimizar a área. Projetos com granularidade grande ou grossa favorecem o roteamento ao disponibilizarem menor número de pinos de entrada e saída ao roteador. Entretanto nessas abordagens é necessário adotar boas técnicas de mapeamento tecnológico para tirar máxima vantagem das funcionalidades oferecidas em cada bloco básico e evitar o desperdício de área interna dos blocos.

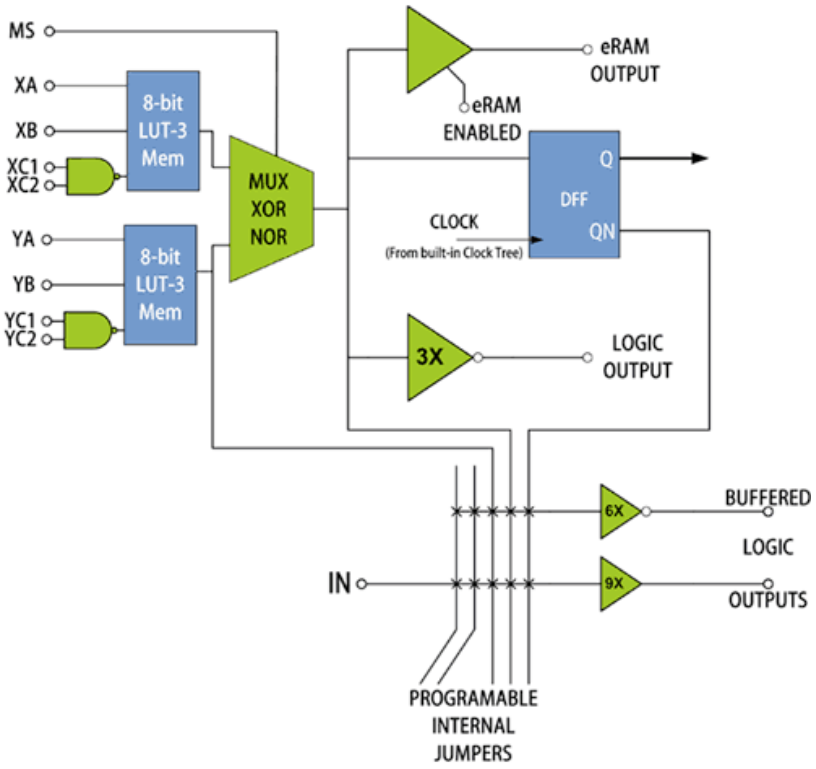


Figura 3.4: Exemplo de granularidade grande – Bloco interno de uma célula eASIC, composto por 2 LUTs de 3 entradas, bloco de memória RAM, um flip-flop D, *buffers* e portas básicas (eASIC, 2005a)

A escolha do tipo de componente que será utilizado no projeto afeta diretamente questões de roteamento e de área. Projetos com baixa granularidade necessitam maior número de componentes para realizar uma mesma função que projetos com granularidade alta. A baixa granularidade também implica em maior número de pinos a serem conectados pelo roteador. Entretanto, projetos com grande granularidade podem acarretar em desperdício de área interna no componente, ou seja, é necessário maior esforço lógico para utilizar todas as funções lógicas disponibilizadas por componentes em arquiteturas com grande granularidade (KOORAPATY, 2004). Procurou-se exemplificar metodologias de projeto regulares que adotam os diferentes níveis de granularidade. Estes estilos de projeto são caracterizados no próximo capítulo. Os projetos Structured ASIC (Seção 4.9) adotam diferentes graus de granularidade, dependendo do objetivo e fluxo de síntese priorizado pelo fabricante.

4 ESTILOS DE PROJETO PARA EXPLORAÇÃO DA REGULARIDADE

As opções de projeto de circuitos integrados possíveis atualmente são o projeto completo de um novo chip, o projeto utilizando um chip com partes pré-difundidas, ou seja, pré-fabricadas e o projeto adotando chips externamente programáveis. Independente da opção, ferramentas de CAD dedicadas aos diferentes modos de projeto tornam possível a finalização de um projeto. Dois fatores são adotados para a classificação: a forma de implementação (conhecida também como forma de fabricação) e os estilos de projeto (*design styles)* (DE MICHELI, 1994).

A forma de implementação refere-se às etapas de fabricação nas quais o projetista pode interferir, de modo a personalizar o circuito. O estilo de projeto refere-se à utilização de partes pré-projetadas, geradas automaticamente ou manualmente, pré-definidos ou não. De modo geral, estes fatores são considerados simultaneamente, dividindo as abordagens de implementação simplesmente em *custom* e *semicustom*. *Custom* refere-se a projetos onde todos os elementos são construídos especificamente de acordo com as definições e características do projeto. *Semicustom* caracteriza-se pela utilização de partes pré-projetadas ou pré-difundidas. Entretanto, cada forma de implementação possui seus próprios estilos de projeto, sendo que alguns estilos de projeto apenas têm significado em algumas formas de implementação.

Reis (1992) classifica as formas de implementação pelas etapas de fabricação de circuitos CMOS dividindo em difusão, metalização e encapsulamento. A difusão refere-se às etapas de configuração sobre silício dos transistores, podendo envolver o primeiro nível de metalização, quando utilizado para as interconexões locais (intracelulares). A etapa de metalização refere-se à criação das conexões entre células ou personalização de lógica pelas camadas superiores de metal. O encapsulamento é o momento em que o chip recebe o invólucro para permitir o seu manuseio.

A implementação de CIs pode ser classificada também quanto ao momento no qual o circuito integrado é personalizado. Güntzel (2000) apresenta a classificação a seguir para os circuitos quanto ao momento de personalização:

- Circuitos Integrados personalizáveis por todas as máscaras;
- Circuitos Integrados personalizáveis por algumas máscaras;
- Circuitos Integrados personalizáveis após o encapsulamento;

Esta classificação pode ser aplicada a projetos regulares. Assim, a primeira abordagem gera todas as máscaras da matriz, de acordo com o leiaute das células básicas, obedecendo ao posicionamento gerado durante o fluxo de síntese. Geralmente, os circuitos apresentam maior densidade lógica porque normalmente são aplicadas

regras para minimizar a área final ou são realizadas otimizações para atingir necessidades de desempenho ou atraso. Esta metodologia se assemelha aos estilos de leiaute Standard Cell, comentado na Seção 4.5, e *full-custom*.

Os circuitos regulares personalizáveis por algumas máscaras mantêm algumas máscaras do processo de fabricação pré-difundidas e permitem a personalização das células pelas camadas restantes, normalmente recebendo uma personalização pelas camadas de metal. Estes circuitos também podem ser denominados de pré-difundidos. O custo dos processos iniciais é amortizado e o tempo de produção reduzido. Exemplos deste estilo de leiaute são Gate Array, apresentado na Seção 4.4, e Structured ASIC, na Seção 4.9.

Na última abordagem, a matriz de células já está pré-difundida e será realizado o assinalamento das células às posições já difundidas. Deste modo, a diferenciação é feita pela inserção de uma lógica capaz de fazer a personalização interna, que pode ser através da ruptura ou fusão de elementos de conexão ou programação de conexões através de antifusíveis, transistores EPROM ou transistores de passagem. Esta personalização pode ser realizada no próprio local de projeto. O tempo de prototipação é considerado nulo, e são limitantes: o desempenho elétrico, o número de elementos lógicos implementados, e em função disto, o custo unitário alto. Fazem parte desta classe os PLDs (*Programmable Logic Devices*), PLAs (*Programmable Logic Arrays*) e FPGAs.

As próximas seções apresentam alguns dos principais estilos de projeto para geração automática de leiaute, relevantes pelas inovações inseridas historicamente ou pelo modo de explorar a regularidade fornecida no estilo de geração de leiaute.

4.1 Weinberger-array

As matrizes Weinberger foram o primeiro método a empregar leiaute fixo na síntese física de funções booleanas de vários níveis (RABAEY *apud* WEINBERGER, 1967). As descrições lógicas dos circuitos são compostas apenas por portas lógicas NOR, compondo uma matriz unidimensional de portas nMOS, com uma única porta por coluna. A Figura 4.1 apresenta um exemplo de circuito implementado no estilo de leiaute Weinberger-Array (MORAES, 2000). À esquerda está o diagrama lógico do circuito e à direita o leiaute simbólico onde se observa que cada coluna possui duas linhas verticais de metal, uma conectada com o transistor de *pull-up*, servindo de linha de saída da porta e outra conectada com a linha de alimentação GND. As portas consecutivas compartilham a mesma linha de alimentação de GND. Os sinais de entrada cruzam todas as portas através de linhas horizontais paralelas de polissilício. Deste modo, o tamanho da matriz é proporcional ao produto do número de portas (colunas) pelo número de sinais (linhas).

A regularidade é conseguida com todas as portas ocupando o mesmo espaço na matriz, uma coluna, sendo todas as portas iguais, e com cada sinal, seja de entrada, saída ou interno, ocupando completamente uma linha. Estas restrições implicam em matrizes esparsas com grande desperdício de área. Para reduzir a área desperdiçada, freqüentemente é utilizado o algoritmo *left-edge* (HASHIMOTO, 1971) para o assinalamento de mais de um sinal a mesma linha da matriz.

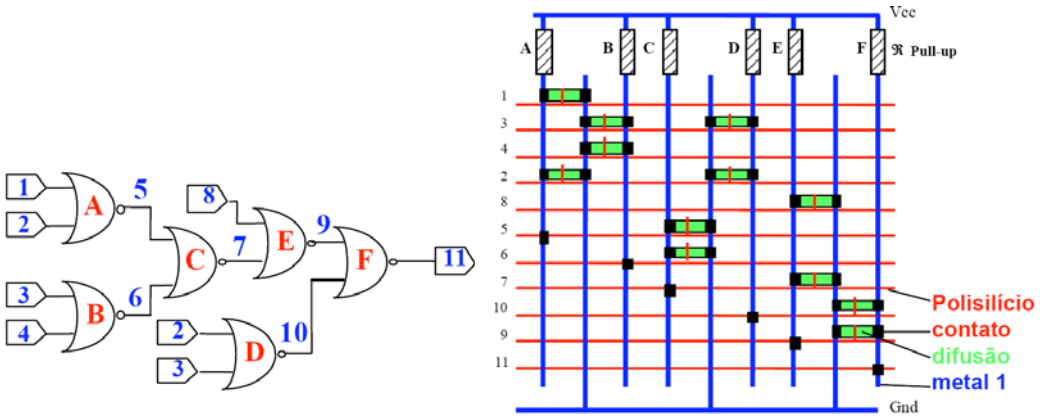


Figura 4.1: Circuito implementado no estilo Weingerger-array (MORAES, 2000)

4.2 Gate matrix

O estilo de projeto *gate matrix* foi proposto por Lopez e Law em 1980 (PREAS, 1988) para a síntese de lógica CMOS estática. Nele, o polissilício é distribuído regularmente em linhas verticais que são utilizadas para a conexão dos *gates* dos transistores. Normalmente, o leiaute consiste de linhas verticais de polissilício uniformemente espaçadas, linhas horizontais de difusão e segmentos de metal. A Figura 4.2 exemplifica esse estilo de leiaute. Cada sinal de entrada, saída ou interno é atribuído a uma linha de polissilício. O tamanho da matriz é definido em função do número de colunas de polissilício. Os transistores são conectados entre si por segmentos de metal ou difusão, em caso de conexões muito próximas (AGARWAL, 1994).

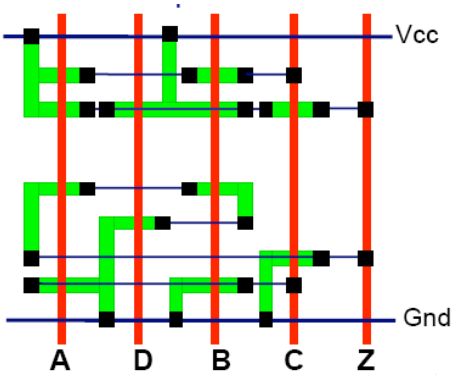


Figura 4.2: Exemplo do estilo Gate matrix (MORAES, 2000)

Este estilo apresenta um número elevado de quebras de difusão, o que implica no aumento das capacitâncias parasitas. Além disso, apresenta baixo desempenho para tecnologias sub-micrônicas e é limitado à implementação de funções simples (MORAES, 2000).

4.3 Linear matrix

O estilo de leiaute *linear matrix*, assim como o *gate matrix*, foi proposto para a síntese de lógica CMOS estática. Entretanto, foca a utilização de células complexas. Neste estilo, existem apenas 2 transistores (um 'N' e um 'P') por coluna de polissilício. As quebras das linhas de difusão encontradas no *gate matrix* são minimizadas gerando as células através de duas linhas horizontais de difusão, como mostra a Figura 4.3.

As principais características deste estilo são o uso de células estáticas, de conexões série/paralelas no modo dual apresentando planos N e P duais, duas linhas de difusão para a implementação dos transistores e de uma linha vertical de polissilício para os transistores de mesmo sinal de *gate*. (UEHARA, 1981)

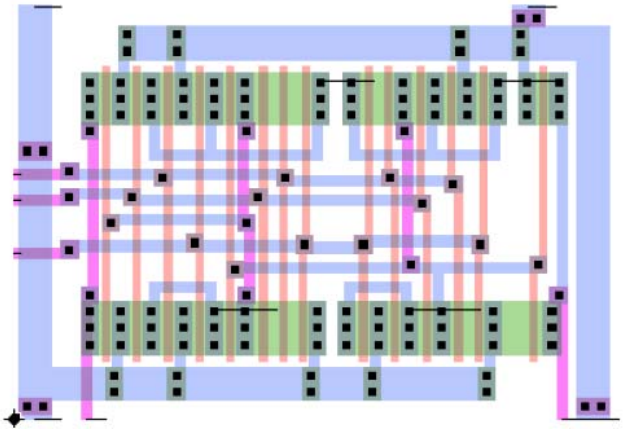


Figura 4.3: *Linear matrix* com duas linhas de difusão horizontais (MORAES, 2000)

4.4 Gate Array

Um circuito *Gate Array* é um circuito pré-fabricado com uma função particular na qual transistores, portas lógicas básicas e outros dispositivos ativos possuem posicionamentos pré-definidos e fabricados em um *wafer,* chamado de *master slice* (RABAEY, 2003). Outra abordagem, conhecida como *Macrocell Array*, somente difere no fato de ser uma matriz pré-fabricada de funções lógicas mais complexas, como flip-flops, ULA (Unidade Lógica Aritmética) e registradores. Em ambas metodologias, para a definição da função do circuito, conhecida como etapa de customização, são adicionadas as interconexões em metal ao *master slice*.

Estas técnicas reduzem o custo por máscara, já que poucas máscaras são produzidas com propósito específico. Além disso, reduzem o custo e tempo de teste, uma vez que o mesmo teste pode ser aplicado a todos os *gate arrays* fabricados em um mesmo *die*. Por essas razões, continuam sendo empregados comercialmente, como exemplo, os fabricados pela AMI (2005) e NEC (2005). No entanto, estes circuitos podem apresentar menor desempenho que outras metodologias de projeto de ASICs, sendo mais indicado para projetos de escala pequena de produção (AMI, 2005).

4.5 Standard Cell

A metodologia *Standard Cell* é o método de projeto de ASICs mais amplamente adotado atualmente. Nesta metodologia, células projetadas e testadas são armazenadas em bibliotecas de especificação de leiautes e disponibilizadas para uso nos projetos. É realizado o mapeamento da descrição lógica do circuito para as células disponibilizadas. Após, as células são posicionadas e roteadas. As células são projetadas seguindo regras especificas de projeto, de modo a serem facilmente integradas em um mesmo projeto (PREAS, 1988). Deste modo, todas as células devem: ter a mesma altura, permanecerem corretas quanto às regras de projeto quando inseridas próximas a outras células, obedecerem ao grid de roteamento e adotarem a mesma largura para as linhas de alimentação (REINHARDT, 2002).

A metodologia *Standard Cell* pode fornecer regularidade quanto aos *gates* utilizados, mas falha na regularidade de roteamento. Geralmente, as interconexões podem passar por qualquer região do leiaute. Isto pode diminuir a previsibilidade dos resultados de fabricação.

Um circuito com conexões longas e muitos componentes de leiaute, tais como, contatos e vias, pode apresentar piores resultados em dissipação de potência e *delay*. Muitas tecnologias *Standard Cell* adotavam roteamento de canal (JOHANN, 2001), o que tornava o leiaute mais imprevisível e mais sujeito às variações por este apresentar um canal de largura diferenciada entre as bandas de células. As novas tecnologias de *Standard Cell* adotam o roteamento sobre as células (*over-the-cell routing*), também conhecido como regime *fixed-die* (JOHANN, 2001). Com esta estratégia a área do circuito é fixa, determina pelas etapas de *floorplaning* e posicionamento, e as conexões deverão ser realizadas dentro do espaço de células.

No fluxo de projeto, os algoritmos de interconexão podem garantir alguma previsibilidade quanto ao tempo de desempenho nos casos onde é possível conectar todas as portas. Entretanto, é difícil prever o *wire length* antes do final do roteamento devido aos problemas de congestionamento. Áreas congestionadas são regiões onde existe grande quantidade de conexões realizadas, impedindo a passagem de novas conexões. Os algoritmos de roteamento necessitam encontrar modos de realizar as conexões em regiões congestionadas, criando dobras inesperadas nas conexões. Outro fator que dificulta a previsibilidade de conexões é o fato de algoritmos de roteamento serem, na maioria dos casos, baseados em decisões não-determinísticas.

4.6 Estilo baseado em células transparentes

Este estilo de leiaute é proposto para a geração automática de leiaute usando células transparentes e roteamento sobre as células (*full-over-the-cell routing*) dentro da Metodologia TRANCA (*Transparent Cell Approach*) (REIS, 1987). Esta metodologia, derivada da metodologia *full custom*, tem como principal objetivo a automação da geração do leiaute sem os canais de roteamento adotados nos circuitos *Standard Cells* (REIS, 1988). 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. A Metodologia TRANCA é caracterizada por (REIS, 1987):

- Uso de uma estrutura em bandas;
- Maleabilidade de blocos funcionais e de células;
- Transparência de células e blocos funcionais;
- Gerenciamento de trilhas.

As células transparentes são células projetadas para permitir a passagem de trilhas de metal verticais e horizontais sobre elas. A Figura 4.4 apresenta uma célula transparente e possibilidades de inserção de trilhas verticais e horizontais. Na Figura 4.4.a são mostradas as conexões internas de uma célula transparente. Estas conexões representam obstáculos para o roteador, assim como, são os pontos de conexão da célula com as trilhas de roteamento. Nas Figura 4.4.b e Figura 4.4.c são apresentadas conexões da célula com as trilhas horizontais (b) e com as verticais (c). As bandas de células são posicionadas uma do lado da outra, sem canais de roteamento e espelhadas em relação às outras de forma a compartilharem as linhas de alimentação. O desenho de todas as

células deve ter a mesma altura e obedecer ao *grid* de roteamento. Cada trilha tem uma prioridade específica de roteamento. O fluxo de síntese é diferente ao fluxo *Standard Cell*, pois o roteador deve considerar os obstáculos internos das células e controlar as trilhas de roteamento disponíveis (LUBASZEWSKI, 1990). A realização do roteamento é sobre os transistores, na abordagem FOTC (*full-over-the-cell routing*) (JOHANN, 1995).

Figura 4.4: a) Célula transparente com detalhe nas conexões internas. b) Transparência horizontal. c) Transparência vertical (REIS, 1987)

4.7 Programmable Logic Array - PLA

Um PLA (*Programmable Logic Array*) é um dispositivo programável usado para implementar lógica combinacional (PREAS, 1988). O circuito é baseado na representação booleana como um conjunto de termos de soma-de–produtos. Os termos produtos são mapeados para planos programáveis de portas AND. O plano de portas OR agrupa estes termos para produzir a saída desejada. A Figura 4.5 ilustra como são realizadas as conexões das entradas com a matriz de portas AND e como são geradas as saídas para as funções. PLAs podem ser mapeados diretamente em um leiaute regular com as linhas de polissilício e difusão distribuídas na mesma direção e as linhas de metal em direção ortogonal. A maioria dos PLAs é programável pelas máscaras, durante a fabricação (RABAEY, 2003).

Jayakumar e Khatri (2004) propõem uma matriz PLA dinâmica customizada apenas pelas máscaras de metal e de vias. Nesse trabalho, o PLA é construído utilizando somente portas NOR com pré-carga e nas saídas são posicionados flip-flops D. Os circuitos são gerados na tecnologia de 0.1 μm. Khatri (1999) propôs utilizar uma matriz PLA como parte de uma estrutura regular no estilo *Structured ASIC* (Seção 4.9).

Outra estrutura baseada em PLAs é a RiverPLA (RPLA) apresentada por Mo (2002a). O RiverPLA utiliza uma estrutura de múltiplos PLAs para atingir regularidade de leiaute e reduzir o tempo de projeto, quando comparado com a metodologia *Standard Cell*. Na síntese, são eliminadas as etapas de mapeamento tecnológico, posicionamento e roteamento. Os nodos de uma rede booleana representada por soma-de-produtos são clusterizados em uma pilha de PLAs, como mostrado na Figura 4.6. Em cada PLA, as saídas, e algumas das entradas, são ordenadas e passam para o próximo PLA através da estrutura de roteamento "*river routing*". Os sinais de saída podem ser tanto oriundos dos planos AND ou dos planos OR dos PLAs. Entre os planos AND e OR de cada PLA, são disponibilizados *buffers* dedicados. As entradas primárias são posicionadas no primeiro PLA (base da pilha) e as saída são posicionadas preferencialmente na direita ou no topo do circuito.

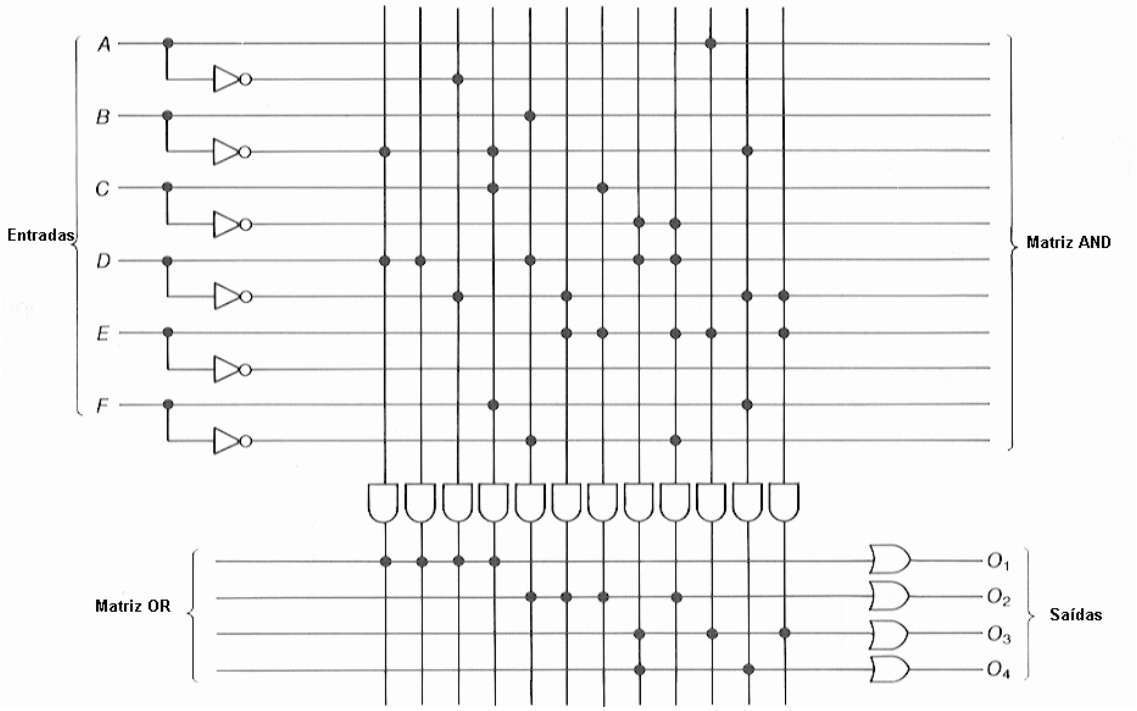


Figura 4.5: Esquema lógico de um PLA (MANO, 1984)

O *river routing* apresenta grande regularidade por utilizar formatos fixos e conexões curtas. Nele são utilizadas apenas duas camadas de metal. O metal 1 é dedicado às conexões dos planos AND e o metal 2 é reservado para as conexões dos planos OR. A estrutura de roteamento, junto com os *buffers* dedicados, é utilizada como alternativa para tratamento dos de buferização.

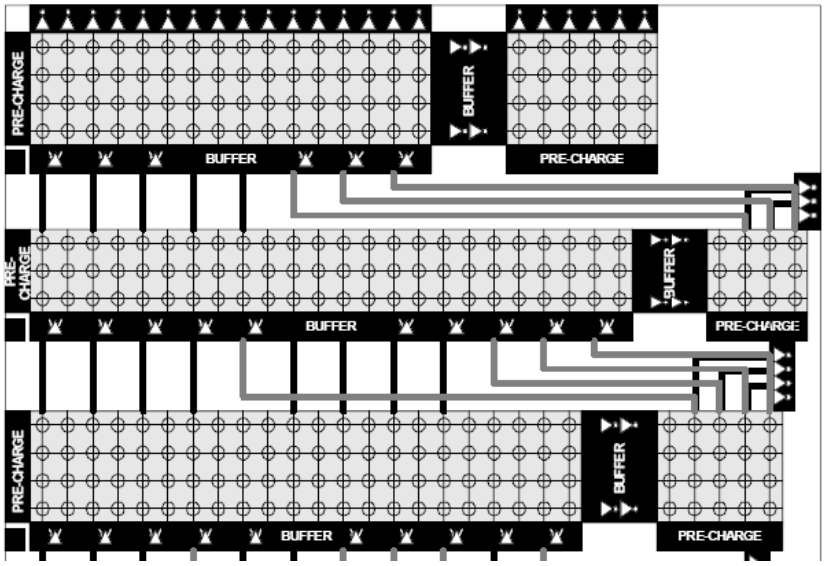


Figura 4.6: Arquitetura adotada no projeto RiverPLA (MO, 2002a)

4.8 Field Programmable Gate Array - FPGA

Um FPGA é um dispositivo composto de componentes lógicos e interconexões programáveis pelo projetista após o processo de fabricação. Os componentes lógicos podem ser programados para as funcionalidades de portas básicas como AND, OR, XOR, NOT ou funções combinacionais mais complexas como decodificadores ou

funções matemáticas. Na maioria dos FPGAs, também estão disponíveis elementos de memória, podendo ser simples flip-flops ou blocos mais completos de memória. A conexão entre os blocos é realizada pelas interconexões programáveis (RABAEY, 2003).

A arquitetura básica consiste de uma matriz de blocos lógicos configuráveis (CLBs) e canais de roteamento. Geralmente, os canais de roteamento têm a mesma largura, ou seja, o mesmo número de fios. Um bloco lógico típico consiste de uma LUT de 4 entradas e de um flip-flop, como mostrado na Figura 4.7. O bloco possui quatro entradas para a LUT, uma entrada para o sinal de clock e uma única saída selecionada entre a saída da LUT ou do flip-flop. Cada saída de bloco lógico pode ser conectada a qualquer segmento de fiação dos canais adjacentes ao CLB.

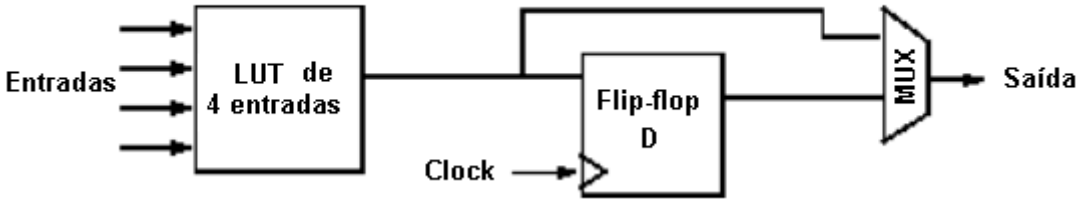


Figura 4.7: Típico bloco básico de FPGAs

Em cada intersecção de canais verticais e horizontais existe uma *switch box*. Em cada *switch box* podem existir chaves programáveis que permitem a conexão com outros fios de canais adjacentes. A Figura 4.8 representa um dos nodos componentes de um FPGA, ilustrando como as células e os *switch box* se comunicam e permitem a programação.

Alguns FPGAs adotam uma arquitetura de granularidade grossa, incluindo os tradicionais blocos lógicos e interconexões, com processadores embarcados. São exemplos os dispositivos Virtex-II PRO (XILINX, 2005), Virtex-4 (XILINX, 2006). Alguns FPGAs da ATMEL (2005) adotam estruturas com granularidade fina, como matrizes de transistores, em suas implementações.

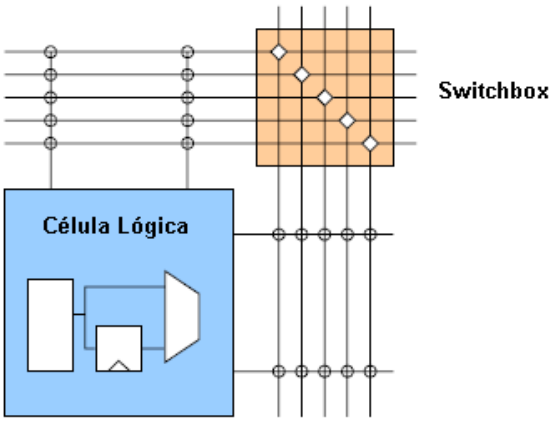


Figura 4.8: Célula lógica e *switch box* de um FPGA (PILLEGI, 2003)

FPGAs são mais lentos que ASICs, considerando uma mesma tecnologia de fabricação. Além disso, apresentam limitação física de componentes e consomem mais energia. Entretanto, oferecem vantagens quanto ao *time-to-market* , facilidade de reprogramação e baixo custo de NRE.

4.9 Structured ASIC

Nesta metodologia, os circuitos são construídos por uma matriz de blocos lógicos básicos e fios, pré-fabricados ou feitos com máscaras fixas. Os projetos são completamente customizados pelas camadas restantes de metal ou são programáveis através de uma ou mais máscaras de fabricação de modo a implementar um ASIC (PILLEGI, 2003). A Figura 4.9 diferencia, quanto ao processo de fabricação, as tecnologias *Standard Cell*, onde todas as máscaras são customizadas, e *Structured* ASIC, onde algumas camadas são utilizadas para definição da camada base, muitas vezes pré-fabricadas, e as camadas restantes são reservadas para a customização.

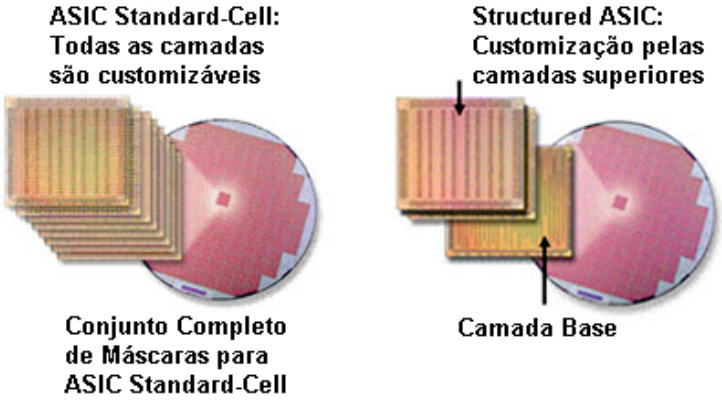


Figura 4.9: Comparação entre o processo de fabricação Standard Cell e Structured ASIC (ALTERA, 2005a)

O estilo *Structured* ASIC surgiu para preencher um espaço de projeto não explorado pelas as metodologias FPGA e *Standard Cell*. A Figura 4.10 apresenta uma comparação entre as metodologias FPGA e *Standard Cell*, destacando o espaço de projeto onde projetos *Structured* ASIC podem ser empregados (ZAHIRI, 2003). Essa metodologia é recomendada para projetos de volume médio, apresenta melhor *yield* e *time-to-market* quando comparado a metodologia *Standard Cell*, além de melhor desempenho e menor área que projetos em FPGAs (RAN, 2004 b).

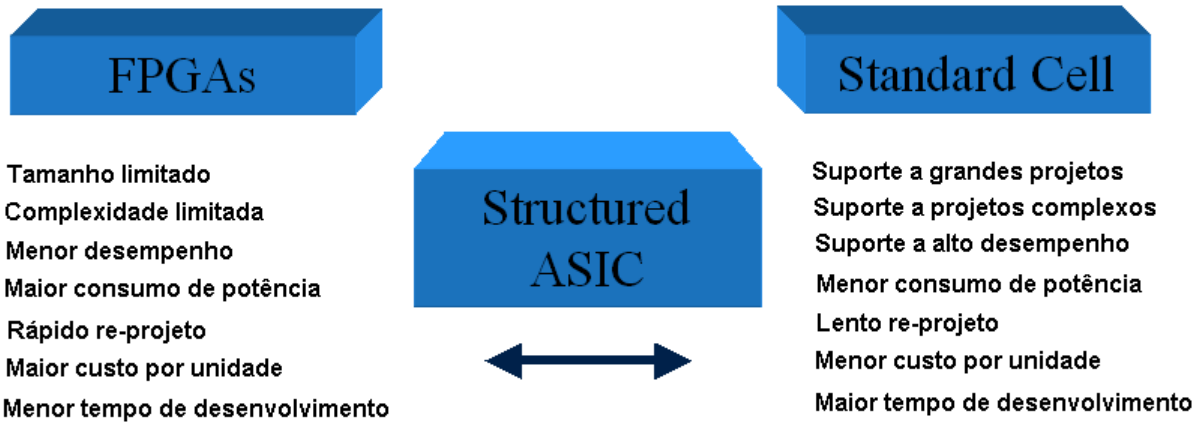


Figura 4.10: Comparação entre a metodologia FPGA e a metodologia *Standard Cell* (ZAHIRI, 2003)

Structured ASICs são menos complexos de projetar porque apenas é necessário o projeto de algumas poucas camadas para completar o projeto. Também, apresentam

menor tempo de projeto por adotarem o posicionamento de blocos de células pré-fabricados (KHANG, 2003).

4.9.1 A arquitetura de Structured ASICs

A arquitetura básica de *Structured* ASICs é composta por duas partes: o elemento fundamental, chamado também de célula básica ou bloco básico, e a matriz de elementos básicos (WU, 2004). Esta matriz é pré-fabricada (também conhecida como *array of elements* ou *sea of components*).

O elemento fundamental contém uma pequena quantidade de lógica, implementando elementos como portas básicas, multiplexadores e LUTs. Dependendo da granularidade da arquitetura, estes blocos lógicos podem conter registradores e memórias RAM (ZAHIRI, 2003). Em um mesmo chip, podem existir um ou mais tipos de elementos fundamentais formando o *Structured* ASIC. Cada elemento é projetado para diferentes propósitos, por exemplo, podem ser adotados um tipo de elemento para lógica combinacional e outro tipo para lógica seqüencial. A Figura 4.11 ilustra este tipo de arquitetura, onde existem elementos lógicos e de armazenamento.

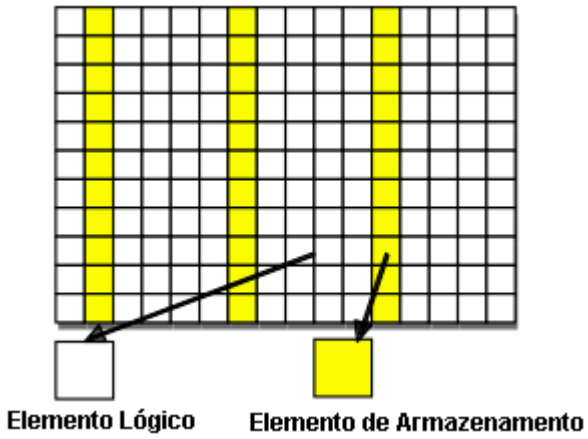


Figura 4.11: Arquitetura Structured ASIC composta de elementos lógicos e de armazenamento (WU, 2004)

Existem dois modelos de matriz (WU, 2004):

- Matriz uniforme: o modelo de matriz uniforme é projetado como uma matriz grade por todo o chip. Normalmente, adota somente um elemento fundamental. Esse elemento pode ser configurado como elemento lógico ou seqüencial. Todos os elementos ocupam o mesmo espaço na matriz e estão alinhados com os elementos adjacentes, verticalmente e horizontalmente. A Figura 4.11 mostra uma matriz neste estilo.
- Matriz não uniforme: neste modelo existem dois ou mais tipos de elementos. Cada tipo de elemento tem que ser alocado em posições específicas do chip, mantendo o alinhamento vertical entre elementos do mesmo tipo. A Figura 4.12 ilustra o modelo de matriz não uniforme. Nesse exemplo, existem três tipos de elementos: um elemento lógico, um buffer para otimização de timing e um elemento seqüencial.

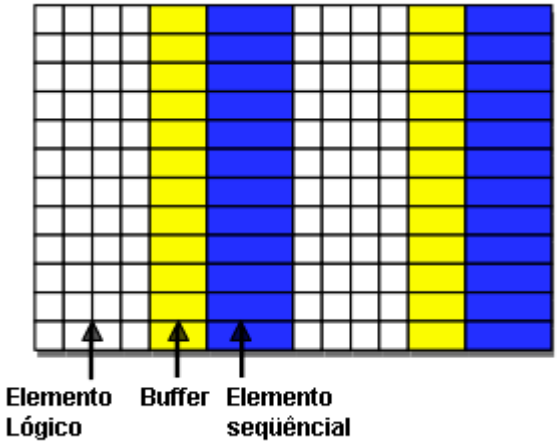


Figura 4.12: Arquitetura Structured ASIC composta por blocos de diferentes tamanhos (WU, 2004)

As funções adotadas nos elementos fundamentais são funções genéricas utilizadas na maioria das aplicações. Estas funções são principalmente pré-fabricadas pelas seguintes razões (BURSKY, 2004):

- Desempenho: as funções pré-fabricadas podem adquirir melhor desempenho e ocupar menor área;
- Previsibilidade: uma vez pré-fabricadas as funções, serão conhecidas todas as características de desempenho, tais como, atrasos e dissipação de potência. Além disso, podem ser pré-verificadas.

As funções são configuradas, geralmente, por um dos três modos apresentados a seguir ou por uma combinação deles:

- Máscaras de metal: normalmente são adotadas as camadas superiores às camadas de metal empregadas no leiaute do bloco básico para definição da lógica.
- Máscaras de vias: a configuração da lógica e das características do roteamento podem ser configuradas através de uma ou mais máscaras de vias.
- Células de memória: assim como realizado em FPGAs, a definição da lógica é configurada através de *bit streams* recebidos na inicialização do sistema. A configuração do sistema é volátil.

A maioria dos fabricantes define *Structured* ASICs como ASICs compostos por uma matriz configurável de elementos genéricos, blocos de memória, PLLs (*phase-locked loops*), DLLs (*delay-locked loops*), interfaces de I/O. (BURSKY, 2004). A Figura 4.13 ilustra estas características da arquitetura típica de *Structured* ASICs comerciais. Todas as funções necessárias para a aplicação, e não disponibilizadas em blocos fixos, são implementadas na matriz, utilizando as camadas de metal ou vias para a customização da função. Algumas abordagens de matrizes assemelham-se aos *Gate Arrays*, diferenciando essencialmente no fato da maioria das camadas de metal estarem já pré-fabricadas, com as conexões entre os transistores já definidas e grande parte das interconexões locais e globais implementadas (ZAHIRI, 2003). Os elementos de alguns

Structured ASICs também são similares a elementos de FPGAs, entretanto sem o *overhead* necessário nos FPGAs para a programação (WU, 2004).

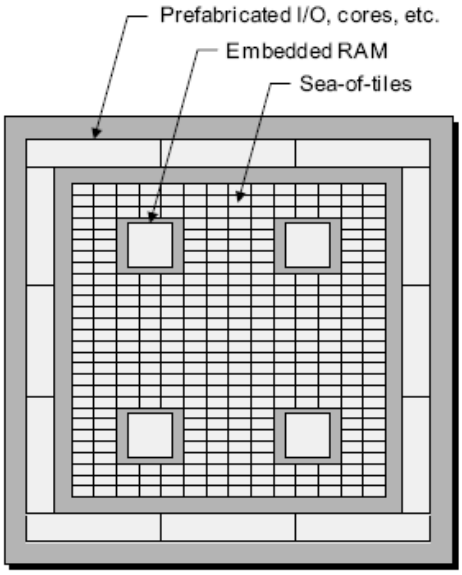


Figura 4.13: Arquitetura típica de *Structured ASICs* comerciais (ZAHIRI, 2003)

Freqüentemente o termo Plataforma (*Plataform*) é utilizado em conjunto com o termo *Structured* ASIC. Plataformas e *Structured* ASICs têm muitas similaridades, entretanto representam metodologias de projeto diferentes (LIPMAN, 2004). Ambas soluções partem do princípio de utilizar partes pré-difundidas que contenham uma coleção de recursos pré-selecionados inseridos em um chip. Cada solução determina o conjunto de funções que comporá o circuito pré-fabricado e como será realizada a personalização deste chip. É neste ponto que as soluções Plataforma e *Structured* ASIC diferem. Diferentemente de *Structured* ASICs, as Plataformas ASICs adicionam, em suas arquiteturas, blocos de maior complexidade funcional, tais como, controladores Ethernet, processadores, controladores de memória, interfaces SERDES (*multi-gigabit serializer-deserializer*), dentre outros (BURSKY, 2004). São exemplos de Plataforma ASICs, o ISSP fabricado pela NEC (2005), o AccelArray da Fujitsu(2005), o RapidChip da LSI (2004), o MPC da Faraday (2005) e o XpressArray da AMI Semiconductor (AMI, 2005b).

4.9.2 Exemplos de Strucured ASICs

A seguir serão apresentados alguns trabalhos acadêmicos e algumas soluções comerciais que abordam as tecnologias *Structured* ASIC e Plataformas para o desenvolvimento de circuitos integrados.

Um fluxo de síntese para uma matriz de *gates* configuráveis pelas vias, o VCGA (*via-configurable gate array*) é proposto por Ran e Marek-Sadowska (2004 a). Esta matriz é composta por blocos lógicos pré-fabricados e por máscaras de metal fixas. As funções lógicas são configuradas pelas máscaras das vias 1 e via 2, assim como as interconexões entre os blocos básicos adjacentes. As interconexões restantes serão realizadas pelas vias 3 ou pelas vias 4.

Um VCGA é composto por blocos básicos chamados de VCCs (RAN, 2004). A arquitetura interna de um VCC é formada por transistores alinhados verticalmente e regiões de difusão N e P. O Metal 1 é distribuído verticalmente e conecta os terminais

de fonte/dreno dos transistores. O Metal 2 é colocado horizontalmente e realiza as conexões entre segmentos de Metal 1. As vias configuram a função do VCC. O número de transistores em um VCC é parametrizável, ou seja, um *n*-VCC contém *n* pares de transistores. Dado o *n*, o número de linhas de Metal 2 é determinado e o leiaute do VCC é gerado automaticamente. A Figura 4.14 mostra um VCC configurado para a função $f = \overline{(x1 + x2)(x3 + x4)}$.

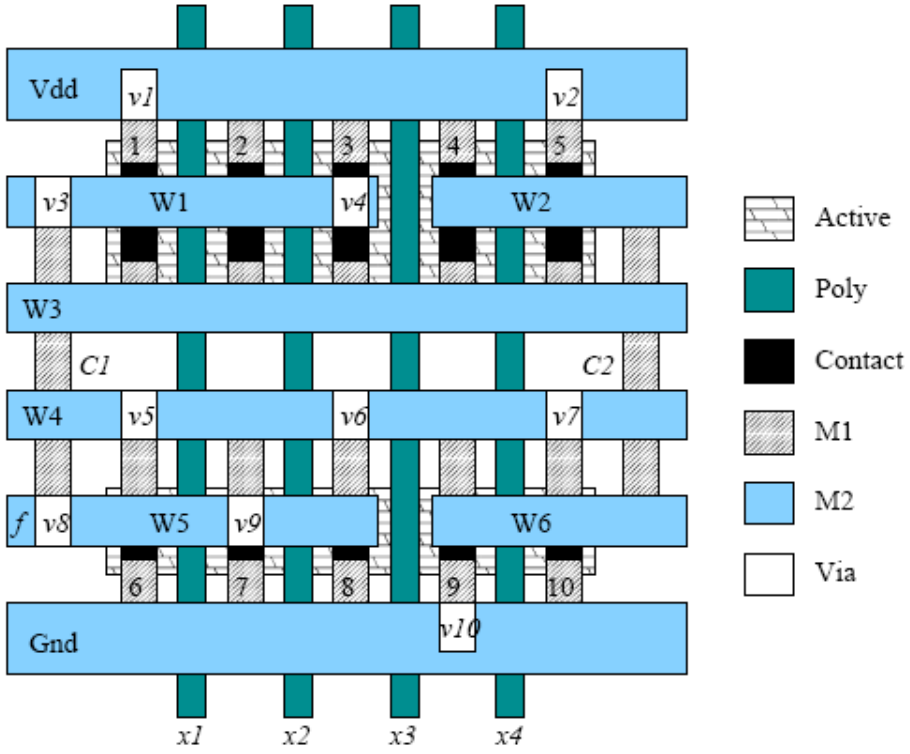


Figura 4.14: Bloco VCC configurado para a função $f = \overline{(x1 + x2)(x3 + x4)}$ (RAN, 2004)

Os *gates* dos VCCs podem apresentar tamanhos diferentes. Um VCGA possui também elementos de memória, formados por flip-flops D definidos pelas vias. Uma versão de VCGA consiste de quatro 5-VCCs e um flip-flop D (RAN, 2005). Os quatros 5-VCCs são dispostos em círculo, rotacionados 90 graus em relação ao anterior. Entre VCCs vizinhos existem matrizes de inversores.

Assim como no VCGA, no *Via Patterned Gate-Array*, VPGA (TONG, 2003), são utilizadas as máscaras de vias para customizar as funções lógicas e as interconexões, ao invés de chaves e elementos de memória SRAM adotados nos FPGAs. Dentre os blocos básicos adotados, o VPGA propõem o uso de uma LUT que, comparada com LUTs utilizadas em FPGAs, é mais rápida por apresentar um nível a menos de lógica e as conexões serem através de vias ao invés de serem por células SRAM (mais resistivas). A Figura 4.15 apresenta o diagrama lógico de uma LUT de 3 entradas em um FPGA e no VPGA (PILEGGI, 2003), No diagrama do FPGA, a cada entrada dos multiplexadores do terceiro nível, conectado ao sinal C, são utilizadas células SRAM para armazenar a configuração. Este nível é substituído no VPGA por conexões, configuradas através da máscara de vias, com o sinal C, ou com o sinal complemento de C. Cada círculo indica um potencial local de posicionamento de via.

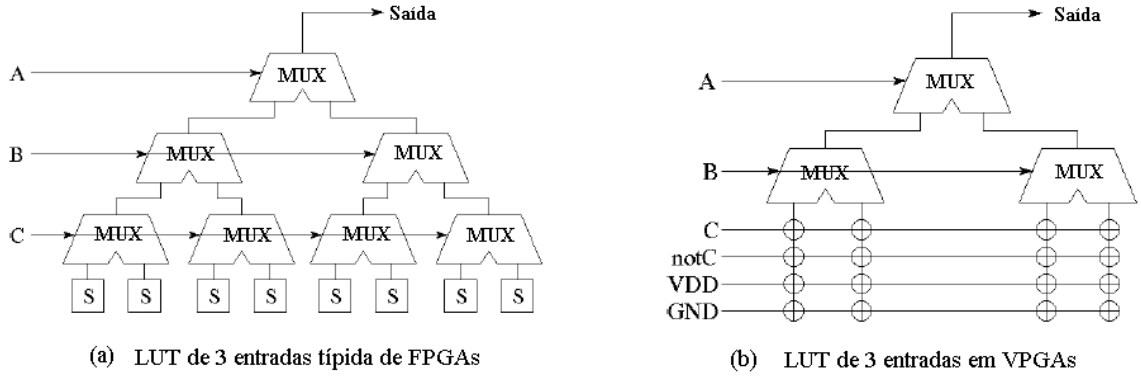


Figura 4.15: Diferença entre LUTs de 3 entradas típicas de FPGAs e de VPGAs (PILLEGI, 2003)

A Figura 4.16 (a) apresenta um exemplo de um bloco básico VPGA composto por uma LUT de 3 entradas e duas portas NAND de três entradas com inversão do sinal de entrada configurada pela camada de vias, 7 inversores e um flip-flop. No leiaute as linhas de *poly* são colocadas em espaços regulares e projetadas para evitar conflitos de projeto, caso seja adotada a técnica PSM na etapa de litografia. Para reduzir a perda de *yield* devido às falhas nos contatos ou vias, é implementada a redundância de contatos e os tamanhos dos segmentos de metal são maiores que os indicados como dimensões mínimas. O roteamento é realizado através de uma matriz de interconexões, construída sobre as células (PATEL, 2003). Um exemplo de uma matriz de conexões 6x3 é mostrado na Figura 4.16 (b).

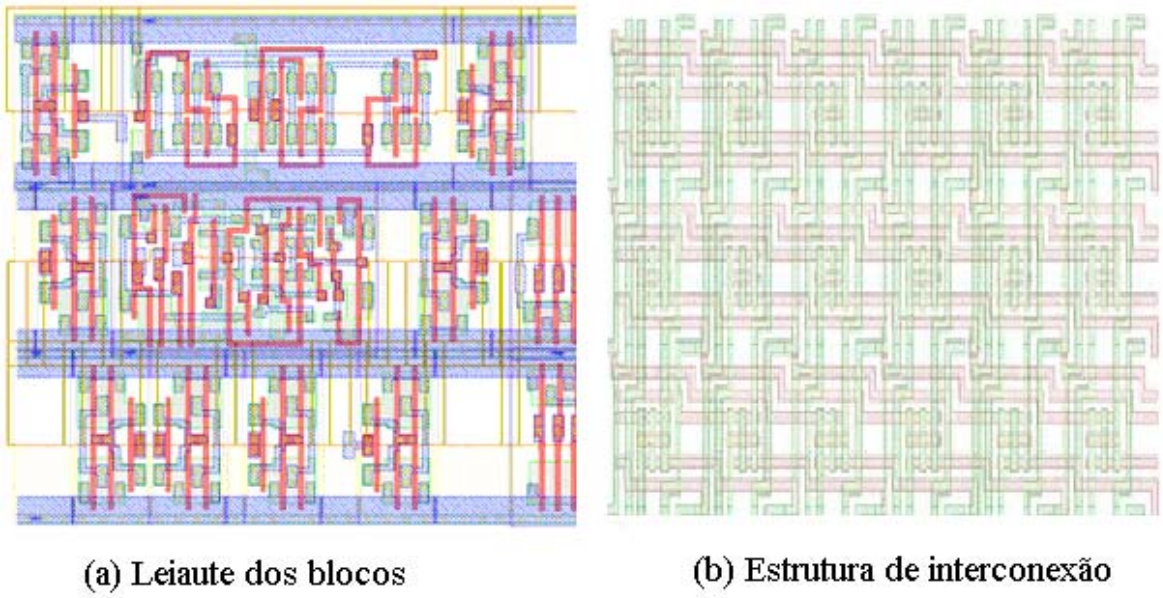


Figura 4.16: Exemplo de leiaute e da estrutura de interconexões gerados pela ferramenta VPGA (PATEL, 2003)

Uma solução comercial que explora a configuração somente pela máscara de vias é a ViASIC (LIOYD, 2005) onde é utilizado um leiaute muito denso, composto de células de memória e células lógicas intercaladas, obtendo um bit de RAM por *gate* sem acrescentar muita área (ViASIC, 2005a). As funções lógicas das células, as interconexões e a memória podem ser configuradas pelas vias (ViASIC, 2005 b).

Ao invés de pensar em um circuito como inicialmente coberto com um mar de transistores ou de *gates*, o PPL, *Path Programmable Logic*, vê o circuito como um conjunto de fios horizontais e verticais, chamado de *sea-of-wires*, no qual podem ser inseridos transistores ou *gates* (GU, 1989). Para gerar um leiaute, sobre o *sea-of-wires* mostrado na Figura 4.17 (a), é somente necessário remover os fios existentes e substituir pelo circuito pré-definido, como mostrado na Figura 4.17(b). As células menores ocupam exatamente uma unidade de área, chamada de *unit cell*. As células que ocupam mais de uma unidade de área são chamadas de *multiple cells*. Um leiaute é composto pela combinação destas células sobre a matriz de conexões, como na Figura 4.18. As células elementares podem ser programadas para realizarem as funções AND e OR, além disso, são disponibilizadas células de flip-flops, de inversores e de transistores de passagem. As conexões podem ser realizadas pelos quatro lados das células, através dos fios do *sea-of-wires*.

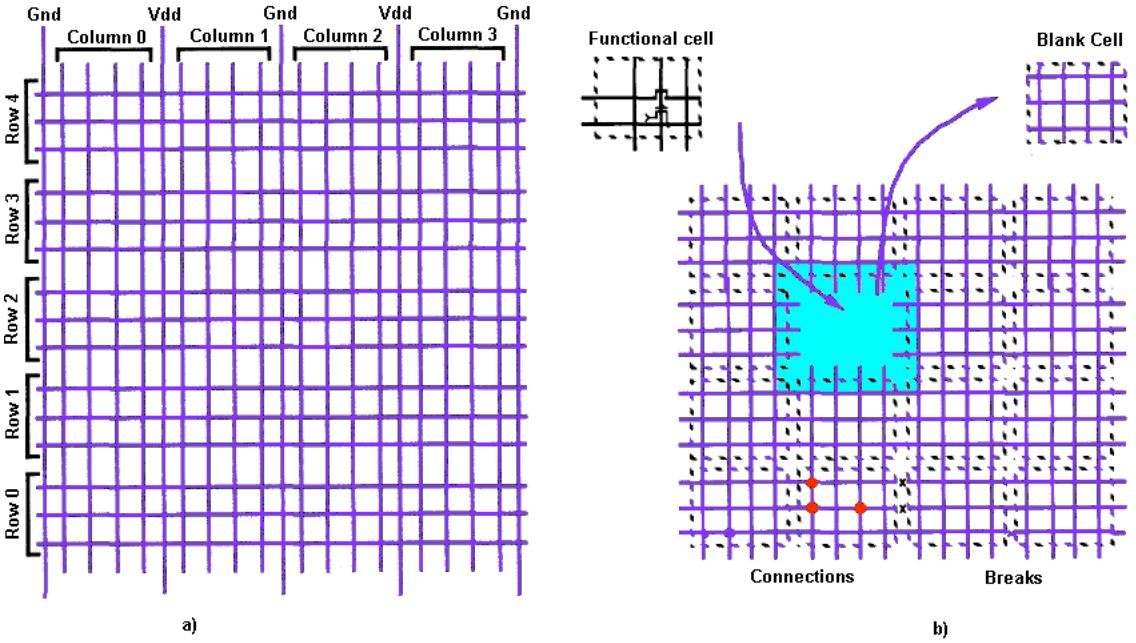


Figura 4.17: Construção de leiaute PPL: a) Aspecto da matriz inicial de conexões – *sea-of-wires*. b) Inserção de célula dentro de um *unit cell* do *sea-of-wires*, com a remoção da fiação pré-existente (GU, 1989).

A Virage Logic's propõem o ASAP MP *Design* (SHERLEKAR, 2004). Este projeto consiste na adoção de um único tipo de célula básica de quatro transistores, sendo dois transistores P e dois transistores N. Deste modo, é possível implementar funções como NAND2, NOR2, Buffers e Inversores. Todas as células da biblioteca ASAP MP são construídos com uma ou mais células básicas adjacentes em uma banda. Essas células básicas compõem uma matriz com as bandas alternadamente espelhadas sobre o eixo X, onde cada célula somente pode ser alocada em posições pré-definidas, ou seja, respeitando um alinhamento das células sobre um *grid* de posicionamento.

A Figura 4.19 mostra a célula básica na tecnologia TSMC de 90 nm, incluindo as conexões em metal 1 que implementam uma célula NAND2. No leiaute são adotados os tamanhos mínimos para os segmentos de metal. O poço N e o substrato são colocados de modo que possam ser compartilhados com as células adjacentes.

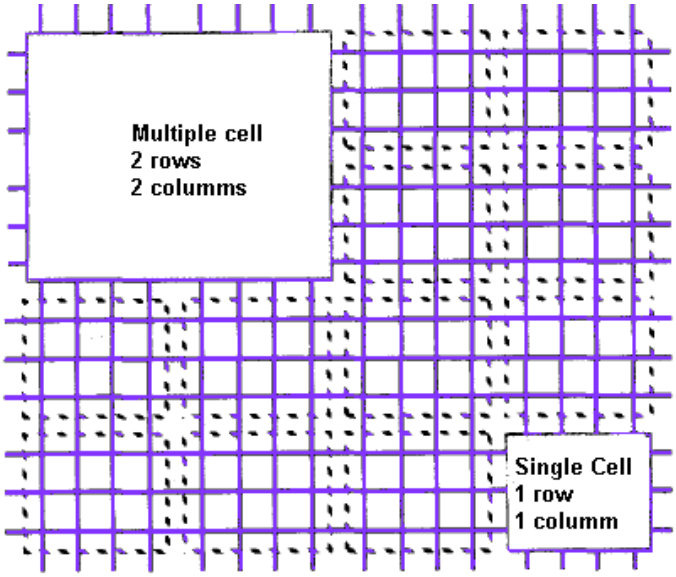


Figura 4.18: Leiaute PPL: composto por *multiple cells* e *single cells* inseridas no *sea-of-wires* (GU, 1989).

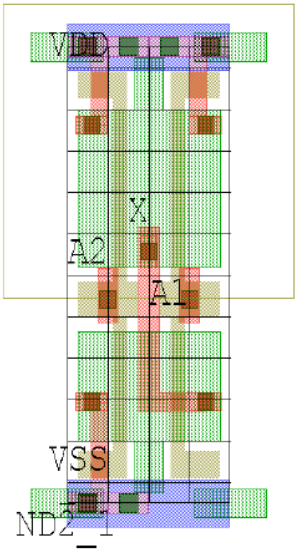


Figura 4.19: Célula NAND gerada pelo ASAP MP Design (SHERLEKAR, 2004)

O HardCopy (ALTERA, 2005a) é um *Structured* ASIC fabricado na tecnologia de 90nm, onde a customização das células é realizada pelas duas camadas superiores de metal. A granularidade da arquitetura é fina, sendo pré - fabricada uma matriz composta de transistores e preservando a arquitetura utilizada nos FPGAs da Altera. As células desta matriz recebem o nome de HCells (CHAWLA, 2005). As células podem implementar módulos do FPGA tais como: LUTs, registradores ou blocos aritméticos. Estas células são agrupadas em macros para implementar módulos do FPGA Stratix (ALTERA, 2005b) ou partes de um processador de sinal digital (DSP). Existem oito macros HCell para funções DSP sendo elas versões de multiplicadores. Uma célula é mostrada na Figura 4.20a) e como é realizado o mapeamento de uma célula na matriz do HardCopy, na Figura 4.20 b). A vantagem deste *Structured* ASIC é a garantia fornecida pela Altera de que após a simulação e teste do projeto no FPGA Stratix, o chip HardCopy produzido funcionará.

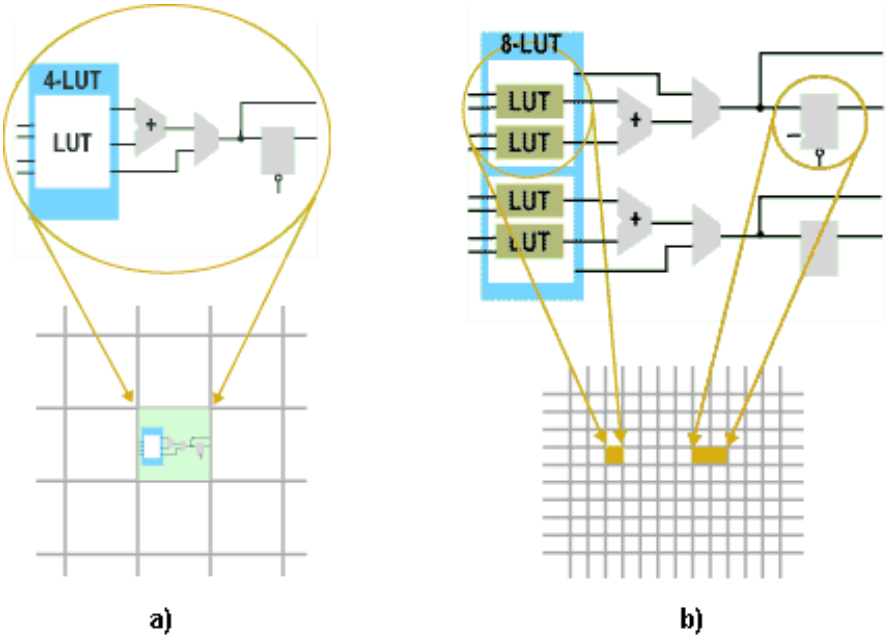


Figura 4.20: Hcell. a) Uma célula Hcell. b) Mapeamento de uma célula Hcell na matriz HardCopy (ALTERA, 2005a)

O eASIC une em um único projeto estratégias adotadas em FPGAs e na geração de ASICs (eASIC, 2005). A Figura 4.21 mostra os aspectos explorados de cada tecnologia para a composição de um eASIC. A lógica é implementada usando LUTs, de modo semelhante aos FPGAs, programáveis por células SRAM e customizadas através de um *bit-stream* carregado na inicialização do sistema. Entretanto, o roteamento é pré-fabricado, sendo customizado por uma camada de via. A customização pode ser feita praticamente sem custo de produção, quando o volume de produção for médio a grande ou utilizando a técnica de *Direct-Write eBeam* (HINTERMAIER, 1991).

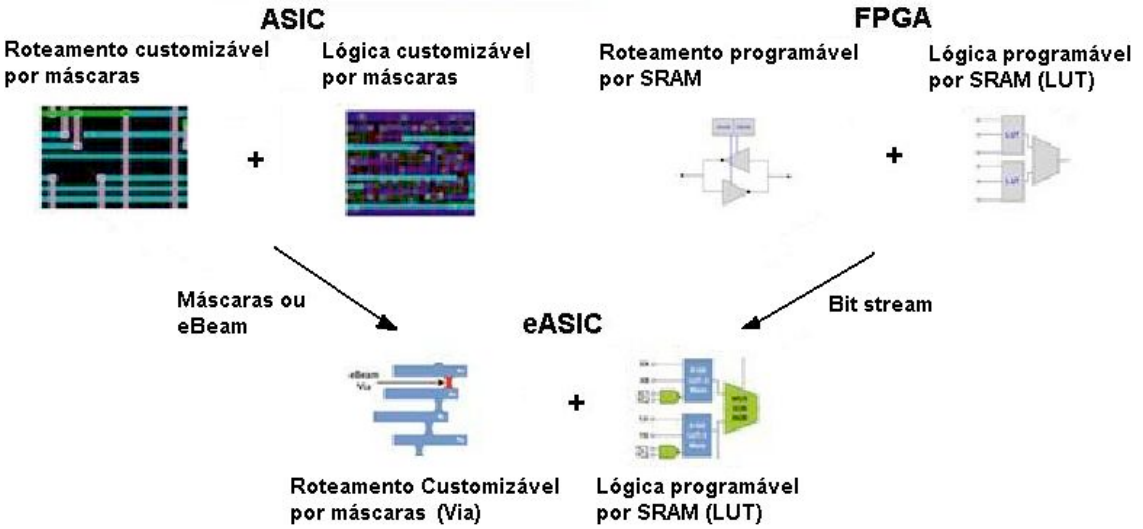


Figura 4.21: Comparação entre as metodologias ASIC, FPGA e eASIC (eASIC, 2005a).

Os *wafers* são pré-processados e armazenados em um banco de *wafers* até o momento da customização da camada de vias. Todas as camadas até o metal 6 são iguais para todos os projetos e são pré-processadas. Apenas a via 6 é utilizada para customizar o roteamento, as interfaces de I/O e os tipos de células, como mostra a

Figura 4.22. As células são chamadas *eCell* e podem ser de três tipos: lógicas, SRAMs ou PLDs. A programação lógica de cada *eCell* é realizada pelo *download* de um *bit-stream*, diferentemente da maioria dos Structured ASICs onde a definição da lógica das células é realizada ou pelas camadas de metal ou pelas camadas de vias.

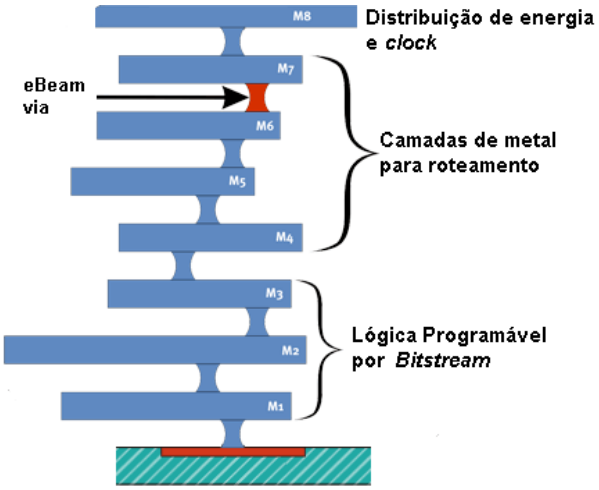


Figura 4.22: Vista lateral das camadas de metal de um eASIC. As três camadas inferiores de metal são utilizadas para programação da lógica. As camadas M4 a M7 são utilizadas para o roteamento das células. A customização é realizada apenas pela máscara de via 6 (eASIC, 2005b).

A Faraday (FARADAY, 2005) atualmente disponibiliza quatro produtos Structured ASIC:

- *Metal Programmable Cell Array Library* (MPCA): disponibiliza células pré-fabricadas, personalizáveis por 3 camadas de metal nas tecnologias de 0.35 μm, 0.25 μm, 0.18 μm, 0.15 μm, 0.13 μm e 90 nm CMOS. A Figura 4.23 apresenta uma vista lateral das camadas de uma célula programável, onde as camadas inferiores estão pré-fabricadas e as camadas de metal M4 a M6, e suas vias, são reservadas para a configuração do circuito (YU-WEN, 2005).

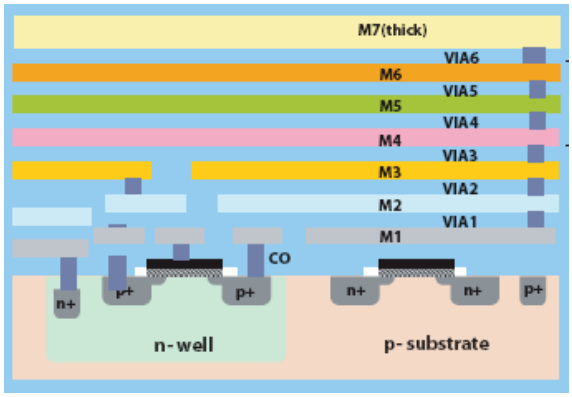


Figura 4.23: Camadas de programação de uma célula MPCA da Faraday. Vista lateral das camadas de fabricação: as camadas inferiores estão pré-fabricadas e as camadas de metal M3 – M6 e suas vias são reservadas para a configuração do circuito (FARADAY, 2005).

- *Metal Programmable I/O Library* (MPI/O): disponibiliza interfaces de entrada e saída onde é necessário modificar apenas uma camada de metal para atender diferentes aplicações. Disponível nas mesmas tecnologias do MPCA.
- *Template Family – General Purpose Platform*: é composto por uma combinação de células programáveis por metal, MPI/Os, memórias, IP *blocks,* disponibilizados em uma plataforma pré-fabricada em *master-slice* na tecnologia de 0.13 μm, como mostrado na Figura 4.24.

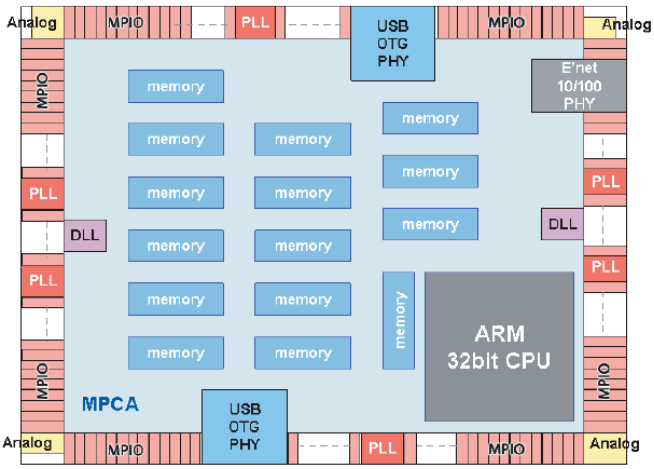


Figura 4.24: Exemplo de *Template master-slice* da Faraday (FARADAY, 2005)

Composer Platform Family – Application – Oriented Platform: inclui processadores e memórias embarcadas, interfaces para memória externa, circuitos analógicos e PLLs. Além disso, reserva área para inserção de *user-defined logic,* como MPCAs, facilmente modificáveis através do re-projeto das camadas de metal utilizadas na personalização.

A NEC (2005) foi uma das primeiras empresas a disponibilizar Structured ASICs. Atualmente, ela fabrica o ISSP (*Instant Silicon Solution Platform*) nas tecnologias CMOS 0.15 μm e 90nm. O ISSP é uma plataforma composta por estruturas de clock, memórias SRAM, APLLs (*Analog phase-locked loops*), DLLs e uma matriz multi-*gates*, chamada de CMG (*Complex multi-gate*) *array*. A Figura 4.25 mostra a arquitetura de um ISSP na tecnologia de 90nm. A arquitetura do ISSP geralmente apresenta as regiões não ocupadas por blocos complexos preenchidas por CMGs. Os CMGs são formados por multiplexadores e flip-flops, sendo os multiplexadores implementados por portas NAND de duas entradas, como destacado na Figura 4.25. O circuito é customizado pelas camadas 4 e 5 de metal e vias. Nos ISSPs fabricados em 90nm, as camadas de metal 6 e 7 são reservadas para alimentação de toda a plataforma.

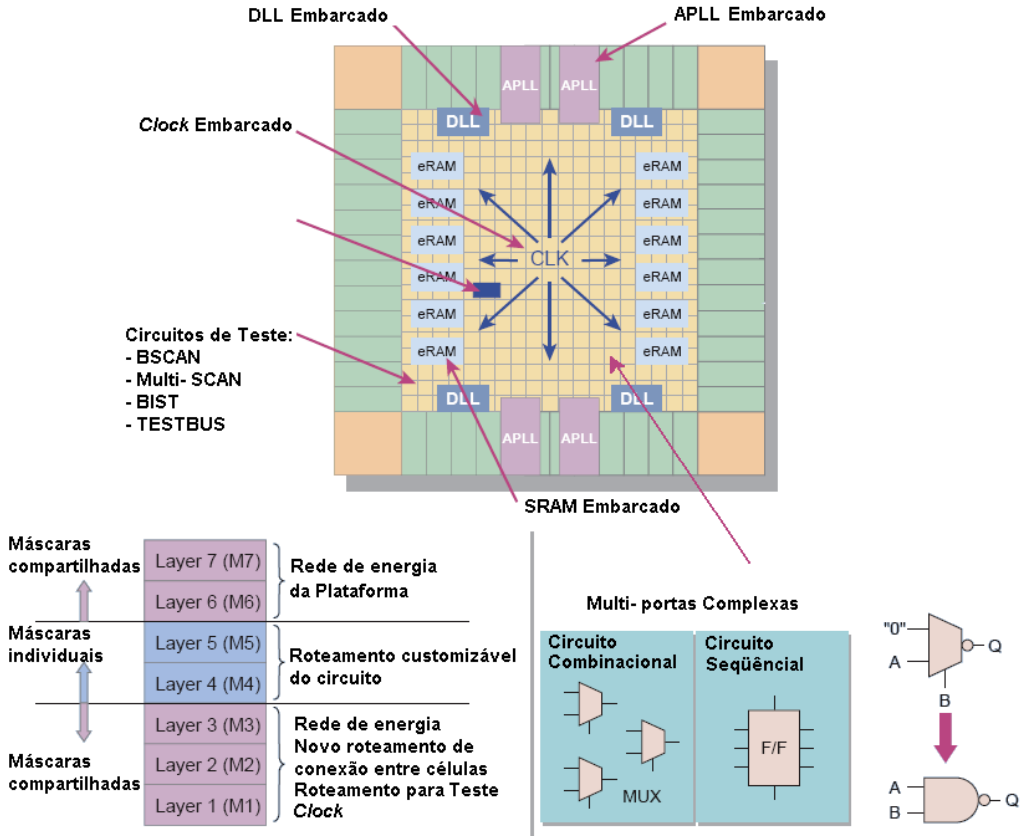


Figura 4.25: Arquitetura do ISSP: vários blocos complexos e a matriz de *Complex multi-gates*. Cada célula da matriz possui flip-flops e multiplexadores. Os multiplexadores são construídos com portas NAND. As camadas de metal 4 e 5 são utilizadas para a customização do roteamento do circuito (NEC, 2005).

AMI Semiconductor XpressArray (AMI, 2005b) é composto por uma matriz de entrada e saídas padrão, blocos de memória configuráveis, blocos lógicos de granularidade grande. A Figura 4.26 mostra a organização de um XpressArray onde a região central é preenchida com um *sea-of-macros* com memórias RAM, DLLs, PLLs e suporte para os módulos de entrada e saída.na tecnologia de 0.18 micra.

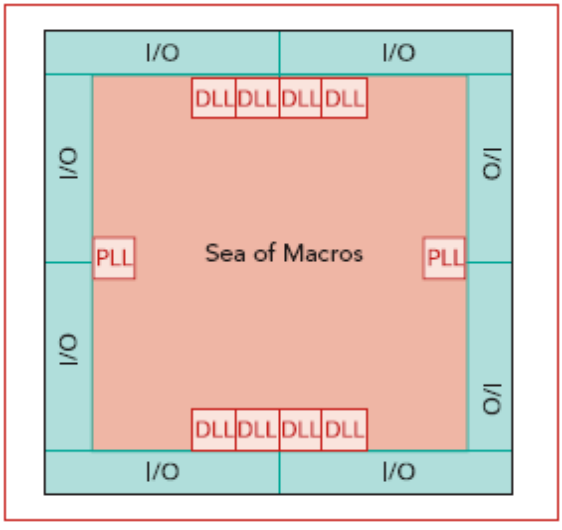


Figura 4.26: Organização do XpressArray (AMI, 2005b)

A Accel Array (FUJITSU, 2005) é uma plataforma disponibilizada na tecnologia de 0.11 micro e customizada pelas quatro camadas de metal superiores. São disponibilizadas macros pré-difundidas de células lógicas, memória SRAM, flip-flops e módulos de interface, além de módulos de entrada e saída programáveis (Metal-programmable I/O cells). A Figura 4.27 mostra a arquitetura de uma célula de um AccelArray, mostrando a disposição interna das macros.

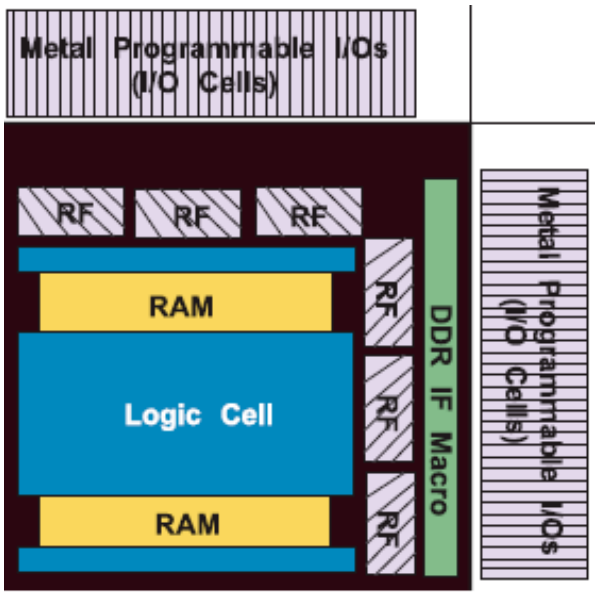


Figura 4.27: Arquitetura de uma célula de uma Plataforma AccelArray (FUJITSU, 2005)

A LSI Logic oferece comercialmente uma solução de Structured ASIC, conhecida como RapidChip Platform ASIC (LSI, 2004), voltada principalmente para projetos de System-on-Chip (SoC). Os circuitos estão disponíveis nos processos de 0.18µ e 0.11µ. Existem três pontos principais sobre os quais RapidChip ASICs são construídos: RapidChip Slices, RapidReady CoreWare® IP e RapidWorx™ Design Kit. Os dois primeiros estão mais relacionados à arquitetura adotada, enquanto o último é um fluxo de síntese de RTL as portas lógicas posicionadas.

Os circuitos podem apresentar memórias e IP cores, além de matrizes de granularidade fina, chamadas de R-Cells. A Figura 4.28 apresenta a arquitetura adotada em um RapidChip Slice. Cada *slice* incorpora, além das matrizes R-Cells (na figura chamada de *Transistor Fabric*), uma combinação particular de blocos de memória, PLLs e IP *blocks*, fornecidos pela biblioteca LSI Logic CoreWare, a mesma utilizada em produtos da LSI para geração Standard Cell. Cada retângulo na figura corresponde a uma região *diffused*, ou seja, pré-fabricada. As R-Cells são compostas por transistores P e N difundidos em áreas livres do chip, nos quais a biblioteca de células será mapeada para implementar a lógica desejada. O anel de entrada e saída é composto por I/Os configuráveis ou I/Os dedicados. Dependendo das especificações do projeto, os blocos de IP oriundos da LSI Logic CoreWare podem ser de quatro formas:

- *Diffused* IP: pode conter Standard Cells, mixed signal ou logic full custom. Estes projetos fornecem o posicionamento, roteamento e são verificados. Eles devem ser fornecidos através de *netlists*, não podem ser realocados e são fabricados em regiões *diffused*.

- *Hard* IP: são posicionados e roteados nas R-Cells e podem ser realocados como um único bloco.
- *Firm* IP: são especificados através de *netlists* com as restrições de *timing*. Já posicionados, mas não roteados e implementados em R-Cells. O roteamento é realizado durante a etapa de roteamento.
- *Soft* IP: são descrições RTL com restrições de *timing*, implementados em R-Cells e podem ser livremente realocados no *slice*. Oferecem maior flexibilidade porque são sintetizados durante a síntese física usando o RapidWorx™ Design Kit.

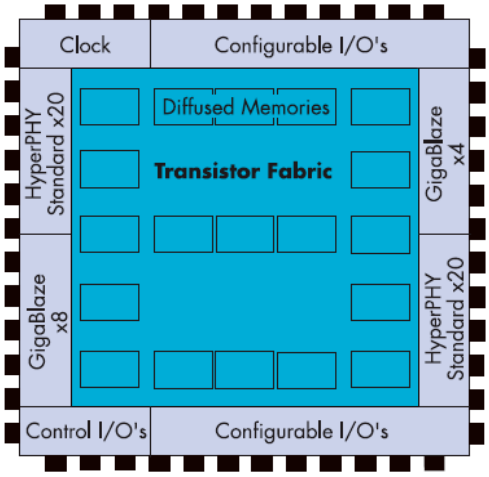


Figura 4.28: Arquitetura básica de um RapidChip Slice: composto por blocos complexos e a matriz R-Cell, (Transistor Fabric) (LSI, 2004)

4.10 Trabalhos desenvolvidos na UFRGS

4.10.1 ÁGATA

O ambiente de projeto ÁGATA (JOHANN, 2000, CARRO, 1996 e 1997) foi o primeiro projeto de geração de circuitos ASICs regulares na UFRGS a ter a matriz projetada fabricada. O projeto foi desenvolvido voltado para a geração de *Gate Arrays*, trabalhando com uma matriz pré-fabricada de transistores.

A matriz, mostrada na Figura 4.29, possui canais de roteamento distribuídos horizontalmente entre as bandas de transistores. Este canal possui trilhas verticais pré-fabricadas em metal 1. As conexões são realizadas em metal 2. Apenas a camada de metal 2 é especificada pelo usuário e é posteriormente processada, reduzindo o custo de prototipação e aumentando a velocidade de produção.

O ambiente ÁGATA foi composto por duas ferramentas principais: o simulador e o gerador de leiautes. Os projetos eram simulados e, quando estavam de acordo com as especificações, passavam para a realização física do sistema, utilizando as ferramentas de particionamento (CARRO, 1996) e roteamento (JOHANN, 1996) desenvolvidas no ambiente ÁGATA.

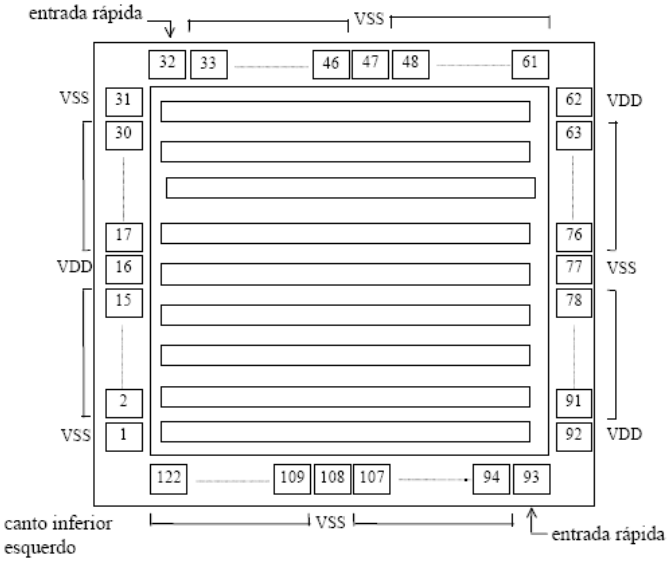


Figura 4.29: Matriz GA2500: disposição dos PADs e das bandas (SIMÕES, 1992)

4.10.2 MARCELA

No projeto MARCELA (GÜNTZEL, 1993), a geração do leiaute constrói uma matriz de uso genérico composta por células básicas dispostas dentro de unidades básicas repetidas ao longo da matriz.

A arquitetura apresentada possui células básicas bem definidas e de uso direcionado espalhadas uniformemente. O conjunto de células básicas implementadas na versão desenvolvida foi Inversor, NAND e NOR devido à simplicidade e facilidade de decomposição de qualquer outra lógica nestas funções. Adicionalmente foi inserida ao conjunto uma chave complementar, *transmission gate*, para facilitar a implementação de multiplexadores e *flip-flops* dinâmicos. A proporção entre essas primitivas, bem como o número de entradas para as portas NAND e NOR foi determinado estatisticamente, considerando uma amostra com circuitos projetados no GME e partes de controle de microprocessadores comerciais. Este estudo apontou que o número de inversores deveria ser o dobro das outras portas e que a chave poderia ser apresentada na mesma proporção que os inversores, facilitando a implementação de bits de memória dinâmica. As células básicas são dispostas formando uma unidade básica da matriz de uso genérico. Para a implementação do leiaute da matriz de uso genérico, adotou-se a tecnologia CMOS 1.2 _m de canal e dois níveis de metal. O barramento de alimentação segue a estrutura de pente. Cada unidade básica é delimitada por duas linhas de GND, sendo cruzada por uma linha de VCC ao centro. As células que ocupam a banda superior da unidade são espelhadas.

O fluxo de síntese adotado parte da escolha de uma matriz pré-difundida conveniente, abordando os aspectos de número de portas. Após inicia-se a geração de uma descrição equivalente do circuito através da decomposição lógica em termos das primitivas disponíveis na matriz escolhida, no caso, primitivas NAND, NOR e inversor. Na próxima etapa é realizado o assinalamento das portas da descrição equivalente às células básicas da matriz e finalmente, é realizado o roteamento.

As células básicas foram projetadas de modo a acomodar trilhas de roteamento tanto na vertical como na horizontal visando o roteamento compartilhado (*over-the-cells routing*).

4.10.3 MARAGATA

O projeto Maragata (LIMA, 1999) constrói o leiaute utilizando matrizes de ULGs (Universal Logic Gates) customizadas pelas camadas superiores de metal. ULGs são células lógicas programáveis que representam uma função booleana de *m* entradas que pode ser configurada para implementar qualquer função booleana de *n* entradas, com *m* maior *n*. Diferente dos MPGAs (*Masked Programmable Gate Arrays*) convencionais que utilizam bandas de transistores e compõem um circuito utilizando componentes de uma biblioteca de padrões de conexões pré-definidas para customizar os transistores.

O Maragata propôs ULGs que podem implementar tanto lógica combinacional e seqüencial. conforme mostra as Figura 4.30 e Figura 4.31. A Figura 4.30 mostra duas ULGs utilizadas, formadas por multiplexadores e *buffers* de saída. Para implementar lógica seqüencial é necessário que exista um flip-flop por bloco lógico. Entretanto, quando os blocos lógicos são utilizados para lógica combinacional existe perda de área correspondente ao flip-flop. Os multiplexadores foram implementados com *transmission-gates* ao invés de portas CMOS com o objetivo de minimizar o número de transistores e a dissipação de potência. Os transistores são dimensionados para trabalhar como *buffers*.

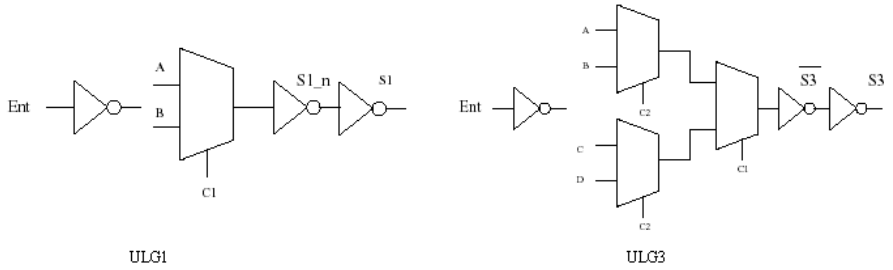


Figura 4.30: ULG1 e ULG3 propostos no MARAGATA (LIMA, 1999)

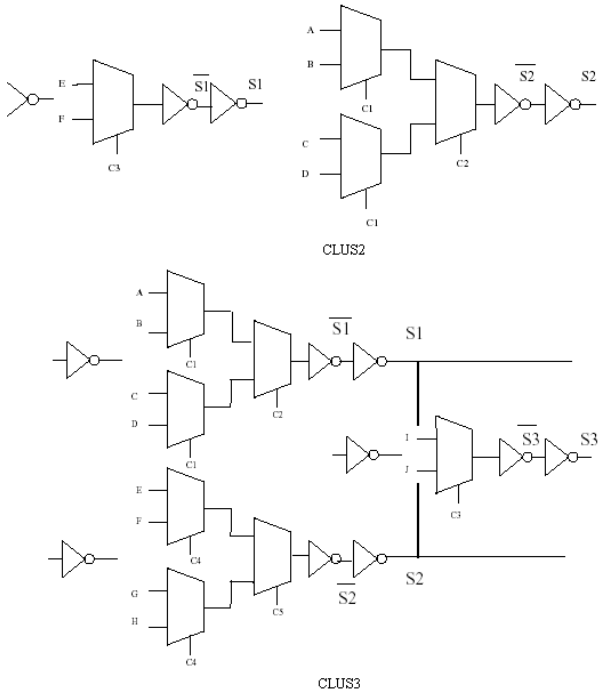


Figura 4.31: Clusters implementados no MARAGATA (LIMA, 1999)

A Figura 4.32 mostra o leiaute interno de uma ULG utilizando tecnologia 0.8 μm, com duas camadas de metal e Metal 2 nas linhas verticais. Este leiaute é o leiaute correspondente à ULG3 da Figura 4.30.

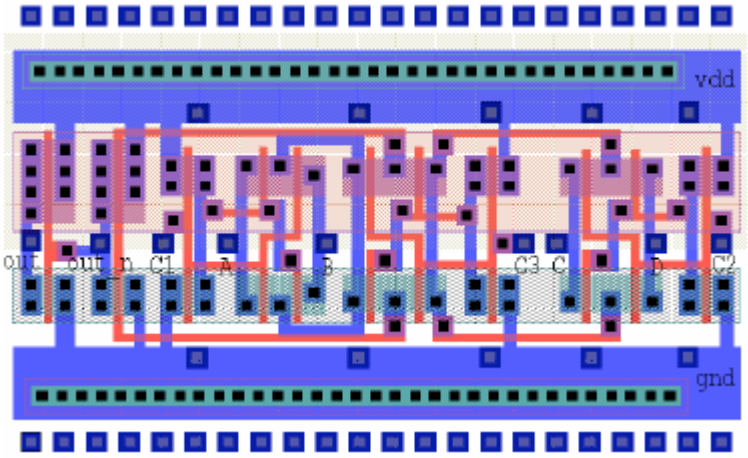


Figura 4.32: Leiaute interno de uma ULG (LIMA, 1999)

4.10.4 MARTELO

O Martelo (MENEZES, 2004) é um gerador de leiaute de matrizes regulares. Neste projeto foi analisado qual o tipo de célula seria mais adequado para sintetizar em tecnologia CMOS, concentrando o estudando em células NAND e NOR. Estas células diminuem o número de transistores por unidade lógica da matriz e também apresentam fácil implementação de funções booleanas. A célula implementada foi uma célula NAND por possuir os transistores PMOS em paralelo, diminuindo o tempo de transição *low-high* comparado à célula NOR.

Além disso, as células estudadas visam o uso da ferramenta de síntese lógica desenvolvida por Tavares (2005), responsável pelo mapeamento de funções booleanas através de células NAND e inversores, sendo capaz também de regular o *fan-out* máximo do circuito.

O Martelo tem como principais características:

- Gerar a lógica dedicada À matriz de NANDs;
- Explorar a possibilidade de formação de conexões curtas através da regulagem do *fan-out* (podendo resultar em duplicação da lógica);
- Geração de conexões mais curtas pela lógica

Os protótipos desenvolvidos do Martelo utilizam a célula NAND implementada nas regras da AMS 0,35 _m, sendo a célula básica formada por duas portas NAND conectadas que pode ser espelhada em todos os sentidos, como mostra a Figura 4.33. Quando uma banda possuir número ímpar de transistores, a última NAND estará sem o par. Para a geração de inversores existem duas opções: a primeira consiste em estabelecer uma linha de *poly* que conecte as duas entradas da NAND; a segunda consiste em conectar a entrada que fica mais ao centro da célula à VDD. A opção utilizada na implementação da matriz foi a segunda porque ela diminui o tempo de transição *high-low* visto que as capacitâncias do transistor próximo ao GND já estão descarregadas. Além disso, como apenas o *gate* de um transistor necessita ser descarregado, é menor o *fan-out* da célula que antecede o inversor, permitindo um

chaveamento mais rápido. A Figura 4.34 apresenta o leiaute de uma matriz 3 x 10 de NANDs gerada pelo Martelo.

O gerador de leiaute permite ao usuário dimensionar a largura dos transistores PMOS e NMOS.

O Martelo utiliza roteamento sobre as células. Se a ferramenta responsável pelo roteamento não consegue realizar o roteamento de todas a *nets*, a geração de leiaute insere trilhas verticais ou horizontais nos locais de congestionamento, conhecidas como *dummies*.

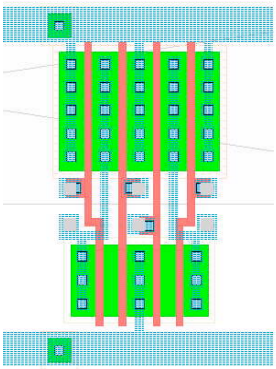


Figura 4.33: Leiaute de um par de NANDs utilizado como *template* para a geração da matriz Martelo (MENEZES, 2004)

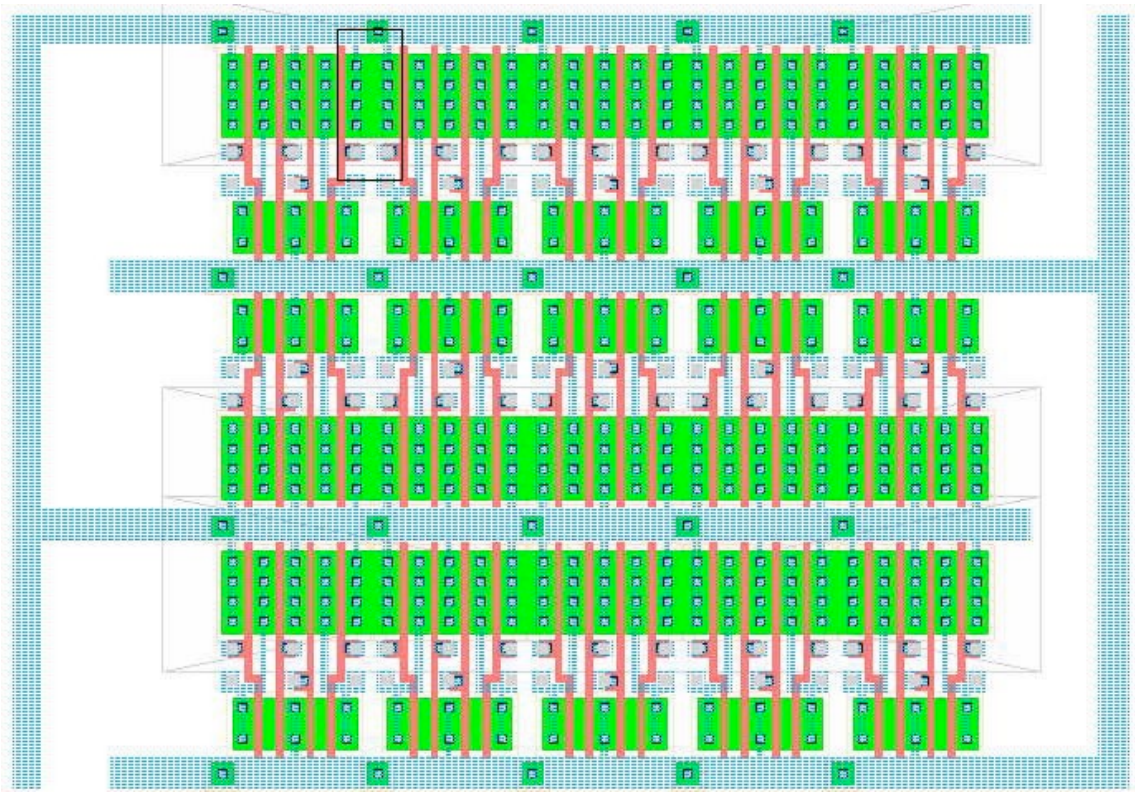


Figura 4.34: Leiaute de uma matriz 3 x 10 de NANDs gerada pelo MARTELO (MENEZES, 2004)

O projeto MARTELO apresenta algumas restrições, dentre elas a dificuldade de roteamento das células devido à concentração das entradas e saídas numa pequena região, limitação imposta pela busca da redução de área. Outra característica é o fato da célula básica ser um par de NANDs. Um dos problemas encontrados na ferramenta MARTELO foi a dificuldade de roteamento com poucas camadas de metal. Uma das causas deste problema pode ser atribuída ao posicionamento dos pinos de entrada e saída. Acredita-se que um modo de minimizar este problema seria oferecer para a ferramenta responsável pelo roteamento mais de uma opção de posição para cada pino de entrada e saída constituinte da célula.

4.11 Comparação entre os trabalhos apresentados

Os trabalhos e metodologias apresentados neste capítulo diferem entre si em vários aspectos, entretanto, todos eles exploram estruturas regulares para a geração de leiautes. Uma classificação proposta por Pileggi (2003) organiza os trabalhos de acordo com como é realizada a definição da lógica e das interconexões no processo de fabricação. São destacados quatro modos principais de customização dos projetos: através de todas as camadas de fabricação, somente pelas camadas de metal e de vias, somente pelas camadas de vias, através de OTP ou através de dados armazenados em memórias SRAM. A Figura 4.35 expande a classificação de Pileggi, classificando os trabalhos e algumas das metodologias apresentadas. Nela é possível observar que a maioria dos trabalhos apresentada na Seção sobre Structured ASICs pertence à mesma classificação, sendo a lógica e as interconexões customizadas pelas camadas de metal e vias. A Tabela 4.1 apresenta um resumo das características dos trabalhos apresentados na seção sobre Structured ASICs e compara estes trabalhos com os desenvolvidos na UFRGS.

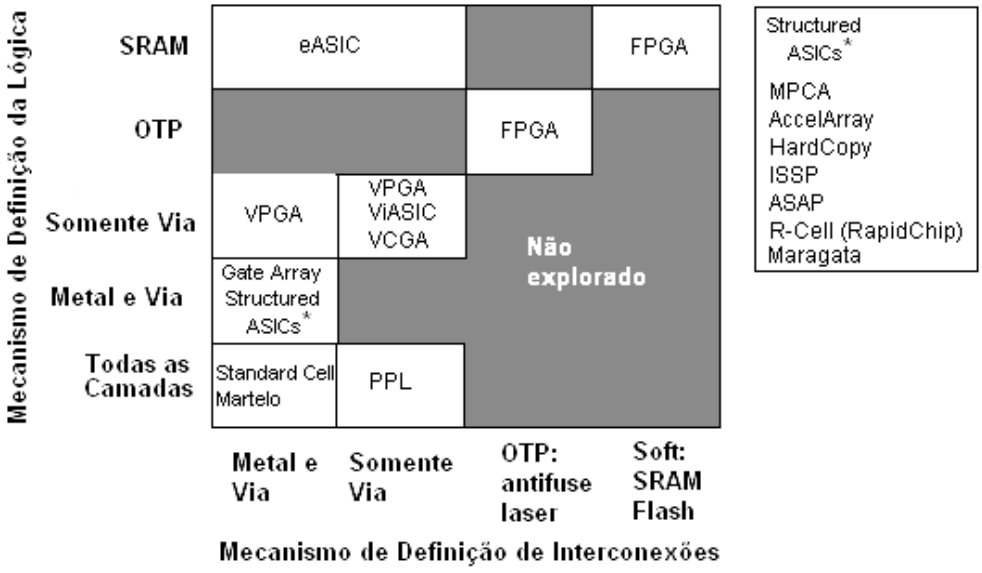


Figura 4.35: Classificação dos trabalhos quanto ao modo de definição da lógica e das interconexões (PILLEGI, 2003)

Tabela 4.1: Resumo das características de trabalhos apresentados

Trabalho	Classificação	Granularidade	Definição da Lógica	Definição das Interconexões	Tecnologia
VCGA	Structured ASIC	Fina	Vias 1 e 2	Vias 3 e 4	180 nm
VPGA	Structured ASIC	Média	Via	Via	130 nm
ViASIC	Structured ASIC	Média	Via	Via	350 nm e 130 nm
PPL	Structured ASIC	Fina ou Pequena	Metais inferiores	Metais superiores	3 μm
ASAP	Structured ASIC	Pequena	Metal 1	Metais Superiores	90 nm
HardCopy	Structured ASIC	Fina	2 camadas superiores de metal		90 nm
eASIC	Structured ASIC	Média	Bitstream em células SRAM	Via 6	90 nm
ISSP (NEC)	Plataform ASIC	Média à grossa	Pré-fabricado	Metal 4 e 5	150nm e 90 nm
MPCA (Faraday)	Plataform ASIC	Média	Pré-fabricado	Metal 4 a 6	350 nm à 90 nm
XpressArray	Plataform ASIC	Grande	Pré-fabricado	Metal superiores	180nm
Accel Array	Plataform ASIC	Grande	Metal 4 e Metal 5	Metal 4 e Metal 5	110nm
RapidChip (R-Cells)	Plataform ASIC	Fina	Pré-fabricado	Depende do projeto	180nm e 110 nm
Marcela	Sea-of-gates	Pequena	Pré-fabricado	Metais superiores	1.2 μm
Àgata	Gate Array	Fina	Pré-fabricado	Metais superiores	1.5 μm
Maragata	Structured ASIC	Média	Metal	Metal	0.8 μm
Martelo	Matriz de células	Pequena	Baseado em *templates*	Metais superiores	0.35 μm

5 GERADOR DE MATRIZES REGULARES - A FERRAMENTA R-CAT

Este capítulo apresenta um gerador de matrizes regulares compostas por células básicas lidas de bibliotecas de células. A ferramenta desenvolvida visa o trabalho com dois tipos de síntese física para leiautes regulares, produzindo circuitos integrados personalizáveis por todas as máscaras ou circuitos personalizáveis por algumas máscaras. O principal objetivo deste gerador é a facilidade de conversão e adaptação à abordagem de matriz escolhida. Isso facilitará a comparação entre diferentes alternativas de matrizes, a adoção de blocos lógicos diversos e de novas tecnologias.

O gerador de matrizes regulares recebeu o nome R-CAT. O gerador explora a regularidade geométrica. Além disso, pode-se obter regularidade lógica ao adotar sínteses lógicas dedicadas ao modelo do R-CAT.

A regularidade geométrica é atingida pela repetição de padrões no leiaute. A Figura 5.1 apresenta a arquitetura genérica de matrizes R-CAT. O modelo de matriz adotado é o modelo uniforme. Cada célula ocupa uma posição na matriz. Todas as células são alinhadas verticalmente e horizontalmente. As células empregadas adotam leiautes simples, com conexões curtas e priorizando a utilização de linhas retas de polissilício.

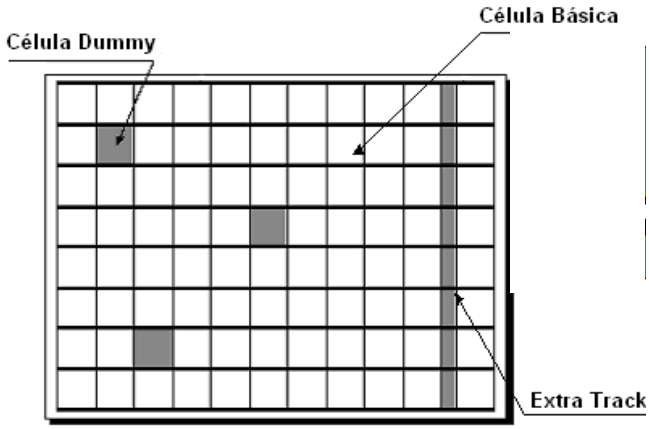


Figura 5.1: Arquitetura genérica R-CAT

Em sínteses de circuitos personalizáveis por todas as camadas, algumas posições da matriz podem ficar vagas. Estas posições recebem o nome de células *dummies* e são inseridas na etapa de posicionamento para facilitar o roteamento. Neste estilo de projeto é permitida a inserção de extra *tracks*, ou seja, trilhas inseridas em regiões congestionadas pela fiação de modo a facilitar a conclusão do roteamento.

Nas sínteses de circuitos personalizáveis por algumas camadas, todas as posições da matriz são previamente determinadas. Deste modo, algumas células podem não ser aproveitadas.

A matriz pode trabalhar com um único bloco básico, um conjunto de blocos básicos ou blocos básicos configuráveis. O tipo de bloco básico utilizado dependerá do projeto lógico do circuito. Os blocos básicos utilizados são armazenados em bibliotecas de células. A adoção de biblioteca de células permite a rápida alteração do leiaute dos blocos e migração de tecnologia. As restrições da matriz R-CAT quanto às células adotadas são:

1. Todas as células devem possuir a mesma altura.
2. Todas as células devem possuir os pinos de entrada e saída alinhados com o *grid* de roteamento.
3. A largura adotada para cada posição na matriz será a maior largura dentre as células utilizadas.

A granularidade do projeto é definida pelo número de células que compõem o circuito. Circuitos com baixa granularidade tendem a apresentar maior regularidade geométrica graças à repetição em maior freqüência dos mesmos padrões básicos. Projetos com granularidade média, onde, por exemplo, são adotadas LUTs, podem reduzir a área necessária de projeto. Entretanto, diminuem a regularidade geométrica.

5.1 Matrizes Personalizáveis por todas as máscaras

O projeto de matrizes personalizáveis por todas as máscaras gera circuitos regulares, onde cada célula básica ocupa uma posição alinhada horizontal e verticalmente com as células vizinhas, compondo uma matriz de células, como mostrado na Figura 5.2. A posição de cada célula é determinada na etapa de posicionamento. Além disso, são inseridos espaços livres para facilitar a etapa de roteamento. Estes espaços podem ser de dois tipos: células *dummies* ou extra *tracks*. Células *dummies* são vazios na matriz, com tamanho igual ao ocupado pelas células. Extra *tracks* são trilhas verticais inseridas em regiões onde há congestionamento de roteamento. Estas trilhas têm a largura correspondente a um passo no *grid* de roteamento, de modo a manter o alinhamento das células e dos pinos com o *grid* de roteamento.

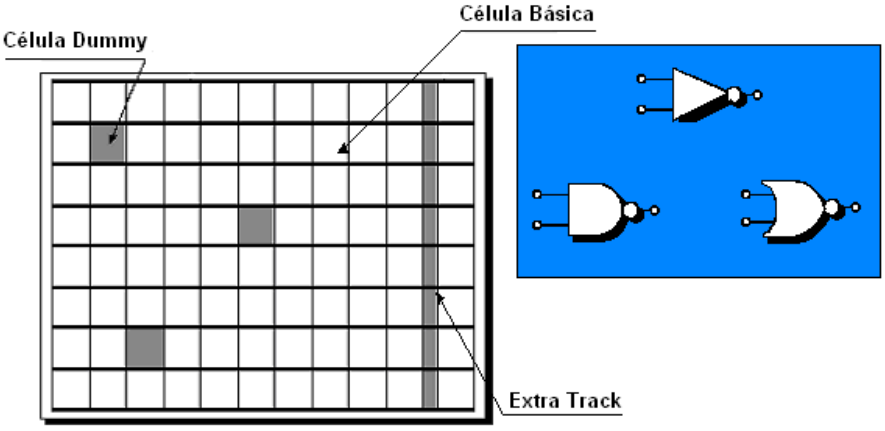


Figura 5.2: Arquitetura da matriz R-CAT personalizável por todas as máscaras

As células básicas adotadas podem variar quanto ao tipo lógico. Na Figura 5.2 são mostrados três tipos de células básicas: inversores, NANDs e NORs. Estas células foram escolhidas para serem utilizadas na geração de uma versão de matriz R-CAT, chamada de matriz NINA (NOR, Inversores e NANDs). Entretanto, outras células podem ser inseridas na matriz.

As matrizes personalizáveis por todas as máscaras são geradas respeitando as características de posicionamento, do *netlist* de entrada e das células básicas adotadas. A geração de circuitos utilizando matrizes personalizáveis por todas as camadas é composta por todos os passos de geração, descritos na Seção 5.4.

5.2 Matrizes Personalizáveis por algumas máscaras

Existem dois tipos de matrizes personalizáveis por algumas máscaras possíveis de serem desenvolvidas com a ferramenta R-CAT:

- **Matrizes com roteamento configurável**: O primeiro tipo é uma matriz pré-fabricada onde cada posição da matriz é ocupada com uma célula lógica. Esse tipo de matriz torna possível realizar projetos semelhantes ao projeto Marcela (GÜNTZEL, 1993). Nele, a customização do circuito é realizada pelas interconexões entre as células pré-difundidas. Para isso é necessário fazer a associação entre as descrições de interconexões, portas lógicas e posicionamento com as posições pré-definidas da matriz. As camadas responsáveis pela customização são as camadas necessárias para o roteamento do circuito.
- **Matrizes com lógica e roteamento configuráveis**: No segundo tipo de matriz personalizável, cada posição da matriz é ocupada com um bloco básico customizável. A principal diferença desta abordagem é que as camadas restantes para a customização são utilizadas para definir a lógica do bloco básico, além de definirem as interconexões entre as células básicas. Este segundo estilo assemelha-se a metodologia Structured ASIC, principalmente ao trabalho apresentado em Sherlekar (2004).

A Figura 5.3 apresenta a arquitetura de matrizes R-CAT personalizáveis por algumas máscaras. Ela ilustra o segundo tipo de matriz personalizável por algumas máscaras descrito anteriormente, ou seja, as matrizes onde a lógica e o roteamento são configuráveis. No exemplo da Figura 5.3, cada posição da matriz é ocupada por um bloco básico, mostrado em destaque fora da matriz. Nem todos os blocos pré-fabricados são necessários para a geração do circuito. Os blocos não utilizados são representados em escuros na figura. Estes blocos podem ser representados na descrição de posicionamento das células como células dummies, utilizadas para facilitar o roteamento ou podem representar posições da matriz que não foram associadas às células descritas na síntese. A mesma representação pode ser estendida para o primeiro tipo de matriz apresentado: as matrizes configuráveis pelo roteamento. Para isso, é necessário considerar que cada bloco básico é uma célula básica completamente pré-definida. O problema de associar corretamente as células disponíveis numa matriz pré-difundida, visando obter o maior aproveitamento de área possível recebe o nome de Assinalamento em alguns fluxos de síntese (GÜNTZEL, 1993). No fluxo de síntese adotado neste trabalho, considera-se que a descrição gerada pelo posicionador será válida quanto à associação de células em matrizes pré-difundidas do primeiro tipo. No caso de utilização da ferramenta R-CAT para este propósito específico é interessante

considerar a inserção de uma ferramenta de assinalamento de portas lógicas no fluxo de projeto da matriz pré-difundida. Uma ferramenta semelhante foi desenvolvida para a utilização no projeto Marcela .

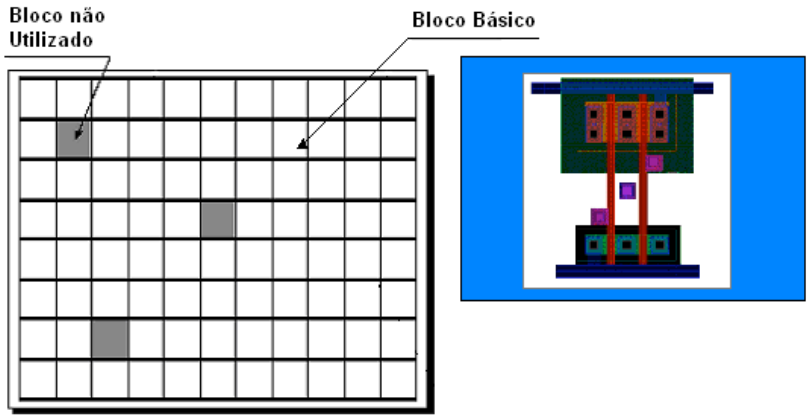
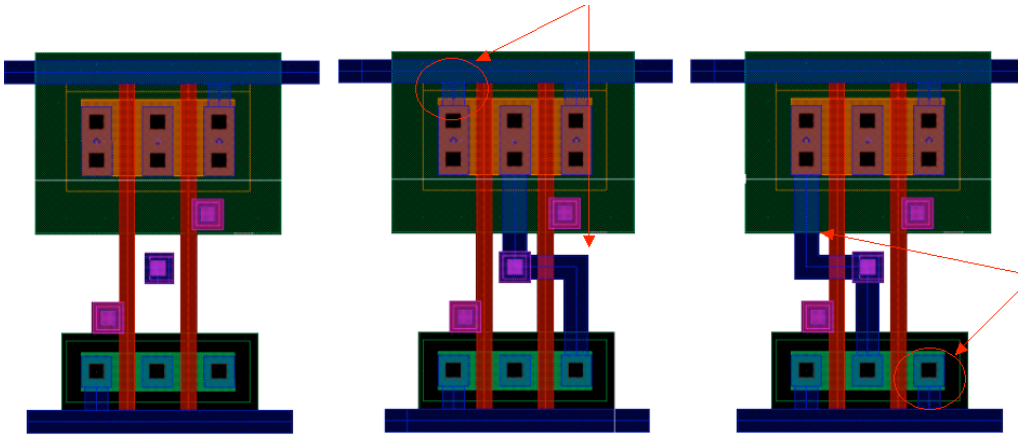


Figura 5.3: Arquitetura da matriz R-CAT personalizável por algumas as máscaras (pré-fabricada)

Uma proposta de matriz com lógica e roteamento configurável por algumas máscaras utiliza um bloco básico composto por dois transistores P e dois transistores N configurável em portas lógicas dos tipos NAND, NOR, Inversor ou Buffer, por um nível de metal. Essa proposta é semelhante à apresenta por Sherlekar (2004), entretanto, o leiaute do bloco básico difere entre as duas abordagens. Na Figura 5.4 é apresentado o bloco básico em (a). A configuração desse como porta NAND é apresentada em (b) e como porta NOR em (c). Para configurar a célula em uma célula NAND são adicionadas duas conexões de metal, neste caso em metal 1, podendo ser realizada em um nível de metal superior. A primeira conexão liga um dos terminais do transistor P à alimentação. A segunda conexão realiza a conexão entre a região P e a região N. São necessárias três alterações para a transformação da célula NAND em célula NOR: o deslocamento da conexão central em metal 1, a remoção da ligação do primeiro transistor P em Vdd e a ligação do segundo transistor N em GND. Essa abordagem fortalece a regularidade geométrica ao explorar os mesmos padrões geométricos na customização das duas portas. Repare na conexão central, responsável pela ligação entre as regiões P e N das portas. O padrão geométrico é apenas deslocado de uma configuração para a outra, acarretando na repetição do mesmo padrão ao longo do circuito com pequena variação no intervalo de repetição.



(a) (b) (c)

Figura 5.4: Leiaute de um bloco básico sendo configurado como uma célula NAND e depois uma célula NOR

Basicamente, a geração de circuitos utilizando matrizes personalizáveis por algumas camadas é composta por duas etapas:

1- **Criação da Matriz**: gera a descrição da matriz composta de blocos básicos ou células básicas. As máscaras utilizadas nessa descrição podem ser pré-fabricadas e armazenadas para uma futura customização ou fabricadas juntamente com as máscaras de customização.

   A etapa de criação da matriz é composta pelos passos de geração de matriz R-CAT mostrados na Figura 5.5 e detalhados na Seção 5.4:

Passos da Criação da Matriz Personalizável por algumas máscaras

Entrada: posicionamento, descrição do bloco básico

Saída: descrição CIF da matriz

1. Definição da Biblioteca de Células
2. Aquisição de Dados
3. Composição do Leiaute

Figura 5.5: Passos da Criação da Matriz Personalizável por algumas máscaras

2- **Customização**: gera a descrição das máscaras restantes necessárias para a customização do circuito. Pode compreender apenas a geração das máscaras necessárias para a realização das interconexões entre as células ou, ainda, a geração de máscaras para a definição da lógica de blocos básicos.

A etapa de customização dos circuitos considera a existência de uma matriz pré-definida. Ela é composta pelos passos mostrados na Figura 5.6 e detalhados a seguir na Seção 5.4:

Passos da Customização da Matriz Pré-fabricada

Entrada: posicionamento, netlist,

Saída: descrição CIF da matriz

1. Aquisição de Dados
2. Composição do Leiaute
3. Roteamento
4. Finalização do Leiaute

Figura 5.6: Passos da Customização da Matriz Pré-fabricada

Dependendo do tipo de matriz a ser construída, os passos possuem rotinas desnecessárias para a geração do tipo de matriz escolhido, por exemplo, quando já existe uma matriz pré-difundida, não são necessários os passos responsáveis pela descrição das células já fabricadas. Esses passos são omitidos, assim como, as informações não necessárias para a síntese escolhida. Por exemplo, durante a etapa de customização, o passo de composição do leiaute fica basicamente restrito à geração da matriz de pontos para o roteamento e à geração das máscaras de customização de lógica, caso seja necessário.

5.3 Fluxo de Síntese

A geração do leiaute estruturado no formato de matriz está inserida num fluxo de síntese dedicado desde as etapas de síntese lógica até o roteamento. As ferramentas desenvolvidas para este fluxo estão voltadas para a regularidade e previsibilidade desejadas da síntese destes circuitos. O fluxo, mostrado na Figura 1.1, inicia com uma etapa de síntese lógica específica para a síntese regular, por exemplo, a sínteses lógica de matrizes compostas por portas lógicas simples, NANDs e NORs. A posição funcional das células na representação do circuito dentro da etapa de síntese lógica pode produzir um pré-posicionamento. Estes pré-posicionamento pode ser utilizado como configuração inicial, assim como, pode impor restrições na etapa de posicionamento. Após, as instâncias geradas são posicionadas, considerando, ou não, a posição funcional no circuito. Se for considerado o pré-posicionamento, ele será refinado na etapa de posicionamento, onde as células são deslocadas mantendo a banda pré-definida na etapa de pré-posicionamento. A etapa de geração da matriz de células, foco deste trabalho, realiza a construção da matriz de acordo com o posicionamento fornecido e utilizando os leiautes das células básicas previamente geradas. Finalmente, as células são roteadas, objetivando o menor custo de fiação possível. A ferramenta de roteamento suporta vários níveis de metal.

As ferramentas de síntese lógica (TAVARES, 2005 e 2006) são baseadas na modelagem do circuito em estruturas de dados semelhantes a BDDs (*Binary Decision Diagrams*), chamadas de orBDD (*or Binary Decision Diagrams*). Estas estruturas são utilizadas na síntese lógica para representar somas de produtos sem redundância. Cada caminho da raiz ao terminal 1 representa um produto da função e cada literal é representado por um nodo (TAVARES, 2000). Estas estruturas podem ser vistas como uma combinação de um multiplexador de 2 entradas e uma porta OR, como mostrado na Figura 5.7.

Os multiplexadores podem ser implementados com diferentes tipos de circuitos, por exemplo, com transistores de passagem, portas complexas, associações de NAND, NOR e inversores ou com somente portas NANDs. As estruturas orBDD são diretamentente mapeadas para as células lógicas adotadas. Para auxiliar na busca pelo conjunto de células capaz de implementar a rede de multiplexadores, as funções são agrupadas em sub-redes. Cada sub-rede tem *n* entradas e uma saída. Deste modo, cada nodo orBDD é representado como um *cluster*. A Figura 5.8 apresenta o circuito gerado com portas NAND, NOR e inversores para o nodo mostrado na Figura 5.7, com *clusters* de até 4 entradas.

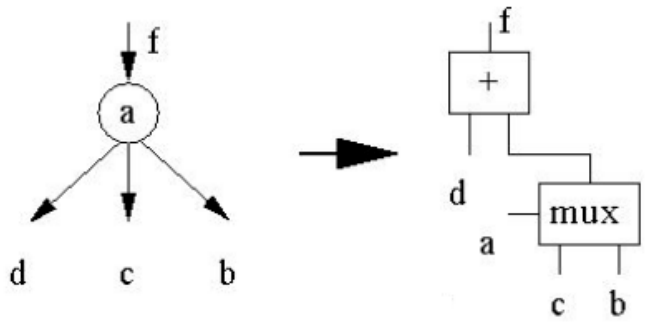


Figura 5.7: Exemplo de um nodo orBDD mapeado para um multiplexador de 2 entradas e uma porta OR

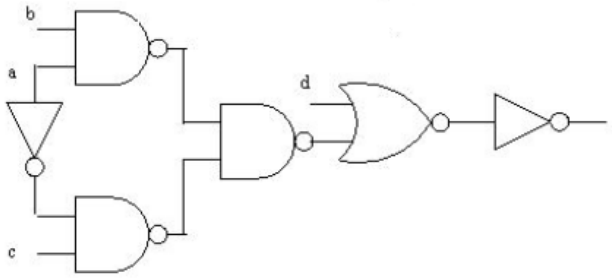


Figura 5.8: Circuito gerado com portas NAND, NOR e Inversores para um exemplo de nodo orBDD

Desse modo, as células internas de um cluster apresentam fan-out 1, ou seja, cada célula interna de um cluster está conectada em apenas uma outra porta. As células de saída do cluster possuem fan-out maior que 1.

Essas ferramentas de síntese lógica podem explorar modos de geração de descrições lógicas de circuitos. Uma ferramenta pode priorizar a geração de circuitos com baixo *fan-out* e mapeamento somente para NANDs, enquanto outras podem adotar o mapeamento das estruturas de dados para multiplexadores implementados com o uso de portas NAND, NOR e Inversores. O pré-posicionamento gerado pode auxiliar na redução do tempo de processamento do posicionador e na redução de *wire length,* dependendo da situação. Embora este fluxo priorize o uso de ferramentas de síntese lógica dedicadas, os módulos componentes deste fluxo são aptos a trabalhar com descrições lógicas geradas por outras ferramentas.

O posicionamento das células adota a ferramenta Mangoparrot (HENTSCHKE, 2004). Esta ferramenta aplica um algoritmo *Simulated Annealing* (KIRKPATRICK, 1983) onde o fator a ser minimizado durante o posicionamento é o *wire length* final do circuito sem permitir sobreposição de células. A saída descreve o posicionamento

relativo das células, descrita em bandas, onde cada banda se transformará em uma linha da matriz e cada célula ocupará uma coluna.

A etapa de geração do leiaute e as técnicas de otimização empregadas são descritas na Seção 5.4.5. A etapa de roteamento será abordada na Seção 5.4.6. O roteamento é realizado por uma ferramenta auxiliar. O roteador adotado não foi desenvolvido neste trabalho e sim, faz parte das pesquisas realizadas no grupo de pesquisa de ferramentas para a síntese física do GME.

Algumas ferramentas adicionais a este fluxo básico de síntese podem ser inseridas para melhorar os resultados obtidos ou para auxiliar na análise e compreensão dos resultados. Ferramentas de análise de timing e *cell sizing* podem contribuir na concepção de circuitos menores em área e com melhores resultados de *delay*, entretanto podem aumentar o tempo necessário para a geração dos mesmos. Foram desenvolvidas ferramentas de verificação de descrições SPICE e uma ferramenta de visualização de posicionamento, VCELL (MEINHARDT, 2005).

5.4 Passos da geração de matrizes R-CAT

A etapa de geração da matriz de células proposta neste trabalho difere da geração de matriz defendida no projeto Martelo. O projeto R-CAT utiliza células previamente definidas em bibliotecas de células ao invés dos *templates* empregados no Martelo. O uso de bibliotecas de células facilita a incorporação de novos blocos lógicos com diferentes funções lógicas, assim como, a adaptação a novas tecnologias com regras de projetos diferentes das definidas nos *templates.* A geração do leiaute engloba os passos de definição das células básicas, caso estas não estejam pré-definidas na biblioteca de células, geração do leiaute da matriz através da utilização das células básicas e respeitando as informações geradas na síntese lógica e posicionamento. Ao mesmo tempo, é durante a composição do leiaute que são realizadas otimizações nas descrições de forma a favorecer o roteamento do circuito. Conexões entre células adjacentes podem ser realizadas em níveis inferiores de metal, respeitando as restrições impostas no leiaute das células básicas. Estas conexões reduzem o número de conexões e o número de obstáculos fornecidos para roteador. Entretanto, muitas vezes para aumentar o número destas conexões é necessário realizar a troca das *nets* assinaladas aos pinos das portas básicas. Esta função somente pode ser realizada em portas básicas onde a ordem dos sinais não interfere no resultado da função.

A Figura 5.9 apresenta as etapas constituintes do processo de geração da matriz regular. Alguns passos do processo são realizados apenas durante a fase de caracterização de células, dependendo do estilo de síntese regular escolhido.

A geração do leiaute regular da matriz oferece dois pontos de início de processamento diferentes: a geração da biblioteca de células básicas e a composição do leiaute final. O leiaute da matriz é determinado pela descrição das *nets* e portas do circuito no formato *netlist* SPICE e pelo posicionamento das células determinado nas etapas de pré-posicionamento e posicionamento do fluxo de síntese, mostradas na Figura 1.1.

A geração da biblioteca envolve a definição das funções lógicas que serão projetadas e disponibilizadas para a geração da matriz. Através de bibliotecas de células, novos blocos básicos em diferentes tecnologias podem ser inseridos de acordo com o projeto objetivo. As bibliotecas de células básicas armazenam descrições dos leiautes das células básicas no formato de descrição de leiaute CIF geradas manualmente. Na versão

inicial, a biblioteca de células será composta de células NAND, NOR e Inversores, projetados na tecnologia AMS 0.35 μm.

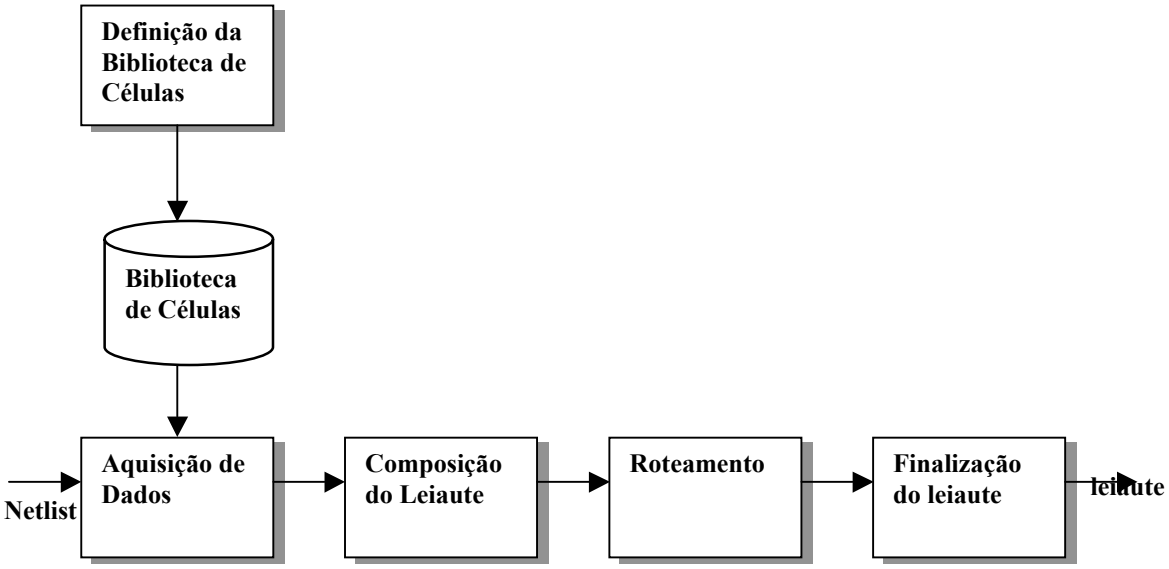


Figura 5.9: Etapas do processo de geração da matriz

A composição do leiaute envolve a instanciação das células básicas descritas no arquivo de *netlist* e dispostas de acordo com o resultado do posicionamento. Também são determinadas as posições dos pinos das células sobre a matriz de roteamento. Finalmente são integrados os leiautes resultantes da composição da matriz e do roteamento.

Foram desenvolvidas cinco opções de utilização da ferramenta:

- *default*: gera matrizes customizáveis por todas as camadas. Nesta opção todos os passos de geração são realizados e o circuito é totalmente gerado a partir da descrição das interconexões e da descrição de posicionamento. São instanciadas as descrições das células básicas utilizadas na síntese lógica, previamente projetadas e armazenadas na biblioteca de células.
- *-matrix*: gera uma matriz *n* por *m* composta por um único bloco básico. Os parâmetros *n* e *m* definem o número de células nas bandas e o número de bandas do circuito, respectivamente. Esta matriz poderá ser pré-fabricada ou não. Nesta matriz, todas as posições estão ocupadas, mesmo que elas não venham a ser utilizadas. A customização do bloco básico é definida posteriormente.
- *–prediff*: gera uma matriz de células pré-posicionadas que serão customizadas apenas pelo roteamento. Neste caso, é necessária a passagem de um arquivo de descrição do posicionamento desejado para as células. As células de uma mesma coluna devem possuir a mesma largura.
- *–customize*: realiza a customização de matrizes onde a lógica dos blocos básicos também será customizada. Para isso, é necessária a passagem das descrições das máscaras de customização, no formato CIF para cada célula empregada na síntese lógica. A customização define as conexões complementares ao bloco básico previamente gerado com a opção –matrix.

- *–rot*: é utilizada para customizar matrizes de células pré-posicionadas. Neste estilo, a lógica das células está totalmente definida na matriz gerada com a opção –prediff. A ferramenta gera a rede de pontos para o roteamento e realiza a integração com a matriz previamente gerada, se a descrição dessa for fornecida.

5.4.1 Biblioteca de Células

Diferente de outros trabalhos que adotavam leiautes fixos ou *templates* para a geração das matrizes. Esse trabalho optou pela utilização de bibliotecas de células para armazenar as células básicas empregadas na concepção das matrizes regulares. Essa escolha tem como principais objetivos permitir:

- a adoção de diferentes tipos de células básicas, capazes de gerar matrizes regulares de acordo com as características do projeto;
- a fácil adaptação a diferentes tecnologias.

Deste modo, a ferramenta de geração de leiaute está apta a trabalhar com conjuntos pequenos, médios ou grandes de células básicas de acordo com os requisitos do projeto. A granularidade do projeto será definida pelo conjunto de células básicas adotado na síntese lógica dos circuitos.

Para cada tipo de célula lógica adotado na síntese lógica deverá haver uma descrição do leiaute desta célula armazenada na biblioteca de células.

5.4.2 Definição da Biblioteca de Células

A primeira etapa para a geração de uma matriz regular nesse sistema é a geração da biblioteca de células básicas. A biblioteca de células básicas armazenará a descrição do leiaute interno das células básicas empregadas na construção da matriz. Com a utilização de uma biblioteca de células é possível facilmente estender a utilização da ferramenta de geração de matrizes regulares para as novas tecnologias de projeto de circuitos integrados, assim como, ampliar ou modificar as funções lógicas empregadas. Estas duas características contribuem para a constante atualização do sistema, sendo possível gerar células básicas nas diferentes tecnologias disponíveis e, principalmente, permitindo a utilização da ferramenta no estudo de matrizes compostas por diferentes células básicas.

Inicialmente, apresentaremos as considerações observadas na utilização da ferramenta de geração de leiaute para compor matrizes personalizáveis por todas as máscaras. A regularidade de leiaute é explorada através da utilização de células básicas do tipo NAND, NOR e Inversor. Adotaremos esse projeto como exemplo dos procedimentos e considerações adotados para a composição da biblioteca de células. A escolha do leiaute a ser utilizado nas células básicas para este trabalho deve considerar o dimensionamento mínimo das regiões N e P dos transistores, as formas geométricas regulares, o número de trilhas verticais e horizontais possíveis, o posicionamento dos pinos e a fácil conversão do leiaute de portas NAND em NOR. As configurações apresentadas a seguir considerarão o leiaute de células NAND. O diagrama elétrico de portas NOR de 2 entradas é muito semelhante à composição de portas NAND de 2 entradas, como mostra na Figura 5.10. Embora a diferença entre elas se restrinja ao modo de conexão dos transistores, os resultados elétricos de atrasos de subida e descida dos sinais de saída podem ser bem diferentes (RABAEY, 2003). A escolha do leiaute

deverá levar em conta um leiaute que apresente bons resultados para as portas NAND e NOR em relação aos atrasos dos sinais, determinado pelo dimensionamento dos transistores.

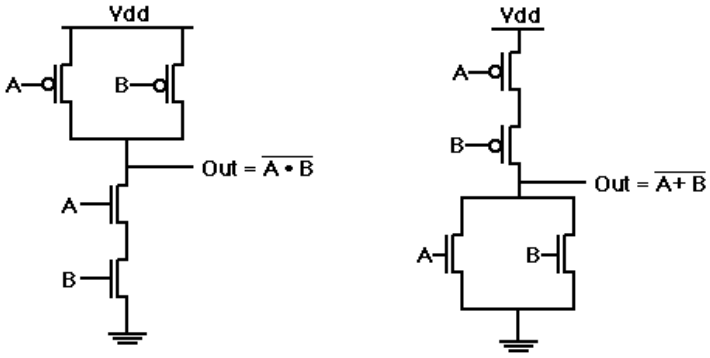


Figura 5.10: Esquemático de portas NAND e NOR

A célula básica utilizada no MARTELO, mostrada na Figura 5.11, permite o estudo de algumas possibilidades de alterações no leiaute visando obter um leiaute mais regular. Algumas configurações são propostas sobre este modelo. Uma das possíveis alterações foca a utilização de estruturas geométricas simples e retas na composição do leiaute, sugerindo a utilização de linhas de polissilício retas, sem dobras. A adoção de leiautes com linhas retas de polissilício pretende fornecer um leiaute mais regular, com formatos geométricos mais fáceis de serem corretamente fabricados, ou seja, com menor taxa de erros ocorridos na etapa de litografia. Esta modificação altera a posição do pino de saída, deslocando–o para outra trilha vertical paralela às trilhas verticais que conectam os pinos de entrada. O pino de saída poderia, assim, ficar localizado entre os pinos de entrada, sendo na mesma trilha horizontal ou em outra, ou ele também poderia ser deslocado à esquerda. Tais alternativas são apresentadas na Figura 5.13. O deslocamento do pino para trilhas muito à esquerda ou direita da célula pode levar ao aumento da largura da célula ou dificultar o encaixe de células vizinhas, devido às regras de distância mínima entre metais. Uma alternativa é alternar a trilha horizontal para os pinos de entrada. Deste modo, o pino de saída estará alinhado horizontalmente com um dos pinos de entrada. Essas alternativas de células são comparadas quanto à influência da adoção de uma delas no roteamento dos circuitos em um estudo de caso apresentado na Seção 6.

Outro aspecto estudado é a utilização de uma célula básica Inversor, com menor largura devido à utilização de apenas dois transistores. A solução adotada no Martelo é a utilização de células NAND para a criação de inversores. Para tal efeito, uma das entradas da porta NAND é conectada à alimentação. A adoção de uma célula básica inversor com metade da largura das células NAND e NOR pode provocar o desalinhamento das colunas da matriz de células. Uma solução para não afetar o alinhamento das colunas é inserir uma célula básica Inversor com menor largura em uma posição da matriz, deixando livre a região onde não existe leiaute na célula.

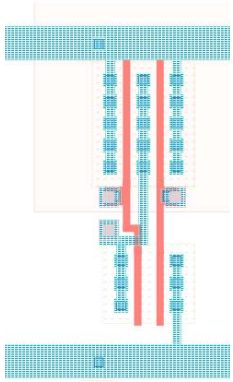


Figura 5.11: Leiaute da célula NAND utilizado no MARTELO

No projeto de matrizes personalizáveis por algumas camadas, compostas por células NAND, NOR e Inversores, uma célula programável por um ou dois níveis de metal pode ser utilizada, como apresentado na Seção 5.2. É relevante estudar os resultados elétricos obtidos para a programação desta célula como porta NAND e como porta NOR. A Figura 5.12 apresentada o bloco básico e as possibilidades de configuração da lógica como porta NAND e como porta NOR. Por exemplo, esse leiaute adota uma relação de Wp = 2Wn com dimensões mínimas para as ligações em metal. A célula permite a passagem de cinco trilhas verticais e seis trilhas horizontais.

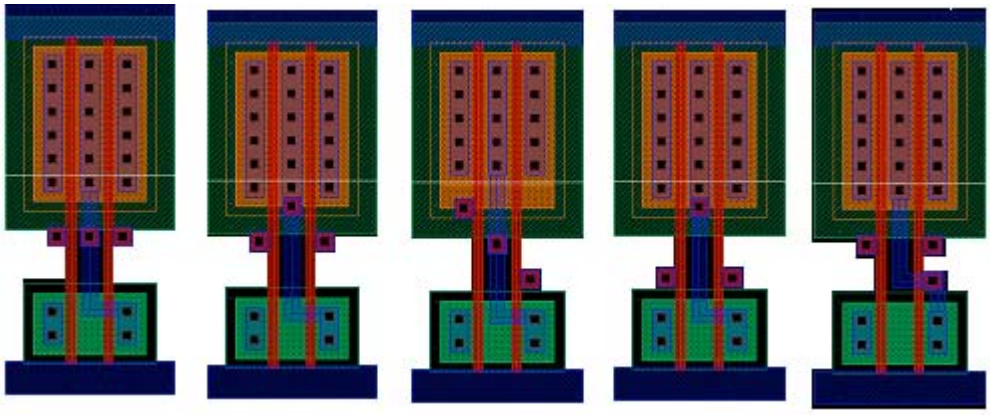


Figura 5.13: Alternativas de leiaute de uma célula NAND com poly reto

A descrição do leiaute interno da célula básica é armazenada na biblioteca de células no formato CIF. Uma descrição do formato CIF pode ser encontrada em (RUBIN, 2005). No Apêndice A são disponibilizadas algumas das descrições de células básicas utilizados nos estudos de caso deste trabalho.

5.4.3 Aquisição de Dados

O leiaute será definido pelas informações armazenadas nos arquivos de *netlist*, posicionamento e de descrição das células básicas. Primeiramente, a leitura do arquivo de *netlist* fornecerá a relação das células e das suas conexões, assim como dos pinos do circuito. O arquivo de posicionamento informará ao sistema a posição de cada célula lida no *netlist*.

A aquisição das descrições das células utilizadas serão processadas e utilizadas para a composição do leiaute final, de forma hierárquica.

Na geração de matrizes configuráveis por algumas camadas, durante a geração da customização é opcional a passagem da matriz previamente gerada. Dependendo do projeto, somente é necessária a geração das máscaras de roteamento.

5.4.4 Arquivo de configuração

A utilização de um arquivo de configuração tornou-se necessária para a geração do leiaute final, para identificação das células e passagem de parâmetros de configuração do roteador. Este arquivo fornece a altura e largura das células na matriz, considerando que todas as células devem manter a mesma relação de aspecto para alinhamento vertical e horizontal das colunas na matriz.

Após, são indicados o eixo de referência para o posicionamento das células, através das coordenadas eixo_X e eixo_Y. Também são informados os parâmetros a serem passados ao roteador Rotdl (FLACH, 2005). Os parâmetros para o roteador são descritos na seguinte ordem: posição inicial do grid de roteamento, o passo do grid de roteamento, ou seja, o intervalo entre as linhas do grid de roteamento, o número, nome e tamanho das camadas de metal disponíveis para roteamento.

Por último, as células básicas disponíveis na biblioteca de células são definidas, ou apenas o nome das células envolvidas no processo de síntese. Para cada célula básica são fornecidos o nome, a identificação utilizada na descrição CIF para esta célula, a posição dos pinos de entrada fornecida em função das trilhas horizontais e verticais que passam pela célula e a posição dos pinos de saída. A Figura 5.14 mostra um exemplo de arquivo de configuração para uma matriz composta por células básicas NAND, NOR e Inversores, onde as células possuem tamanho de 6*9 μm, com três camadas de metal para roteamento. No exemplo, a célula Inversor recebeu o nome *inv* na descrição SPICE e a identificação 3 na descrição CIF armazenada na biblioteca de células, com a entrada posicionada na trilha horizontal 1 e vertical 4.

```
altura 900
largura 600
eixo_X 1600
eixo_Y 450
rot_inicio_X 0
rot_inicio_Y 0
grid_step 150
supply_size 80
metal 3
metal_nomes CMS CVS CMT CVT CMQ
metal_tamanho 90 50 70 50 70
nand2  7 1 4 3 4 2 3
inv     3 1 4 2 3
nor2   4 1 4 3 4 2 3
```

Figura 5.14: Arquivo de Configuração

5.4.5 Composição do Leiaute

O passo de composição de leiaute produz a descrição do circuito respeitando a opção utilizada na chamada do programa, dentro dos cinco tipos de utilização da ferramenta descritos na Seção 5.4. Cada tipo de utilização necessita de um conjunto diferente de informações.

A geração de matrizes personalizáveis por algumas máscaras pode ser feita com blocos básicos sem lógica previamente definida ou com células lógicas arranjadas em uma ordem pré-definida. Na geração com blocos básicos, a etapa de geração recebe a descrição do bloco lógico e o tamanho da matriz, ou seja, as dimensões da matriz em função do número desejado de células e bandas. O resultado é uma matriz de blocos básicos, sem a existência de células *dummies*. Na geração com células pré-posicionadas, a etapa de geração realiza a instanciação das células básicas de acordo com uma descrição do posicionamento desejado para cada célula básica. Comumente, nesse estilo de projeto, células de um mesmo tipo compõem uma coluna de células. A descrição da posição de células segue o padrão adotado no arquivo de posicionamento, entretanto, ao invés de referências às células, no arquivo devem ser passadas referências ao tipo de célula desejado. É permitido o uso de células *dummies*. Como padrão, adotou-se como identificador o número de identificação na descrição CIF da célula básica.

Nas demais opções de utilização da ferramenta, o gerador de leiaute utiliza as informações sobre as conexões entre as células e sobre o posicionamento. Estas informações são lidas na etapa de aquisição de dados. Os parâmetros passados para a ferramenta incluem o caminho destas descrições na árvore de diretórios, o caminho da biblioteca de células e opcionalmente, o caminho para o arquivo de configuração da ferramenta. O caminho do arquivo de configuração é pré-definido em um caminho *default,* o mesmo diretório corrente de execução da ferramenta. As células encontradas no *netlist* são pesquisadas na biblioteca de células e o leiaute destas células é inserido na descrição do leiaute final do circuito, no formato CIF de descrição de leiaute. Cada descrição de células possui um número identificador de células na linguagem CIF que será utilizado na instanciação das células no leiaute final.

A grade de alimentação é realizada em metal 1 e tem sua largura definida no arquivo de configuração. Normalmente, em ferramentas de geração de leiaute, é adotado como padrão a utilização de linhas de alimentação com no mínimo o dobro do tamanho mínimo definido na regra de projeto da tecnologia para a camada de metal empregada (REINHARDT, 2002 b). No projeto de circuitos customizados por todas as camadas, os locais preenchidos por células *dummies* e por trilhas são completados durante a geração da grade alimentação, garantindo a continuidade do sinal.

Os fatores determinantes na composição da matriz são o número de células básicas utilizadas, o posicionamento destas células, o número de células *dummies* projetadas pelo posicionamento para facilitar o roteamento final do circuito, o número de bandas e o tamanho das células básica definidas na biblioteca de células.

Durante a geração da matriz de células, as células são instanciadas respeitando a posição definida durante o posicionamento. Todas as células são dispostas mantendo o alinhamento vertical e horizontal. Nesta implementação, será considerada largura igual para todas as células básicas, sendo utilizada a maior largura como distância entre células adjacentes. Todas as células possuem a mesma altura de banda e as células são espelhadas para compartilhar as linhas de alimentação, formando o pente de alimentação.

5.4.5.1 Conexões entre células Adjacentes

O gerador de leiautes R-CAT identifica células adjacentes que possuem conexões em comum entre elas, posicionadas na mesma banda. Quando isso acontece, o gerador pode realizar a conexão entre essas células em metal 1. Esse procedimento remove

algumas *nets* ou reduz trechos de *nets*, reduzindo o número de conexões a ser realizado pelo roteador. Através de experimentos, verificou-se que esta estratégia pode reduzir em 10% o número de conexões a ser realizado.

Optou-se por utilizar metal 1 porque, deste modo, não são inseridos obstáculos para o roteamento. Entretanto, dependendo do ordenamento das *nets* nas portas adjacentes, não é possível realizar a conexão em metal 1 sem desrespeitar as distâncias mínimas e sem alterar o projeto da célula básica. Nestes casos, a ferramenta de geração de leiaute realiza o reordenamento das *nets* nos pinos de entrada.

O reordenamento das *nets* somente é realizado em projetos onde a troca dos sinais atribuídos aos pinos de entrada não altera o resultado lógico esperado do circuito. Chang et al. (2005) realizam o reordenamento dos pinos em uma etapa particular do fluxo de síntese, após o posicionamento. No gerador R-CAT, o reordenamento das *nets* é realizado durante a geração do leiaute.

O processamento do reordenamento é realizado através da varredura das bandas, célula por célula, verificando se a célula possui alguma *net* em comum com a próxima célula. Se houver uma *net* comum, é verificado à quais pinos estão conectadas as *nets*. As *Nets* associadas a pinos de saída não podem ser trocadas, sendo marcadas como fixas. Para cada célula apenas uma troca de pinos é permitida. Se a *net* compartilhada não estiver conectada no pino de entrada mais próximo da célula vizinha, o assinalamento das *nets* de entrada é trocado e a conexão em metal 1 é realizada. Na célula vizinha também é verificada a posição do pino associado à *net* compartilhada. Caso pino não seja o pino mais próximo da célula anterior, é realizado o reordenamento entre as *nets* da célula vizinha e esta célula é marcada como modificada, para evitar novas trocas de sinais.

A Figura 5.15 apresenta o algoritmo para encontrar conexões entre células adjacentes. Considere o circuito composto por *m* bandas representadas por componentes de um vetor B = (b_1, ..., b_m), onde cada banda é composta por *j* células representadas por componentes de um vetor C= (c_1, ..., c_j). A cada célula é atribuído um vetor de interconexões, de tamanho *p* correspondente ao número de pinos de entrada e saída existentes na célula. Cada conexão atribuída a um pino é representada por um componente em um vetor N = (n_1, ..., n_p). As interconexões são identificadas pelos seus nomes. A função conecta_células avalia se a conexão pode ser realizada e se necessário, efetua o reordenamento das *nets*. Somente é permitido o reordenamento das *nets* associadas aos pinos de entrada nos casos onde a ordem dos sinais não afeta a saída lógica da célula.

As possibilidades de conexões são apresentadas na Figura 5.16. Nela, cada célula é representada por um retângulo. O leiaute interno das células é omitido. Os pinos de entrada e saída são representados por pontos, sendo os pinos de entrada em azul e os pinos de saída em vermelho. As alternativas a) a d) referem-se à células com pinos posicionados na mesma altura, as alternativas e) a g) referem-se de modo geral à células com pinos não alinhados, a alternativa h) ilustra reordenamentos não permitidos e a alternativa i) às conexões em metal 1 não permitidas entre células adjacentes. As alternativas a) e e) mostram os casos mais simples de conexão entre células adjacentes, no qual as *nets* comuns entre as células estão posicionadas nos pinos mais próximos às bordas das células. Neste caso, somente é necessário gerar a descrição CIF deste segmento e remover esta interconexão da lista de conexões passadas ao roteador. Em b) a célula adjacente possui a *net* n3 em comum, entretanto, para realizar a conexão em

metal 1, é necessário reordenar as nets primeiro. O reordenamento é a troca das nets associadas aos pinos de entrada, neste caso, troca-se n5 conectada na primeira entrada com n3 conectada na segunda entrada. Em c) e g) é necessário o reordenamento nas duas células para poder realizar a conexão em metal 1 entre os pinos associados à *net* n3. As conexões podem ser realizadas entre células adjacentes mesmo existindo células *dummies* posicionadas entre elas, como mostrado na figura d). Se os pinos de entrada não são alinhados, a conexão é realizada como mostrado na figura f).

As nets associadas aos pinos de saída não podem ser reordenadas, como mostrado em h). Isso porque os pinos de saída possuem posição fixa no leiaute. As conexões mostradas em metal1 na figura i) não são permitidas porque atravessam o leiaute da célula básica, podendo causar sobreposição de conexões em metal1 ou desrespeitar as regras de projeto.

5.4.6 Roteamento

A etapa de roteamento pode ser vista como um passo externo à geração do leiaute. Entretanto, juntamente com a instanciação das células básicas é gerada uma matriz de pontos para o roteador. Nesta matriz são fornecidas as informações sobre as conexões entre as células, obtidas do *netlist*. A posição dos pinos de entrada e saída na matriz de roteamento é obtida observando o passo de roteamento adotado no *grid*, ou seja, o deslocamento das linhas de roteamento na matriz e a posição interna das trilhas nas células.

Foi utilizado o roteador Rotdl (FLACH, 2005) para realizar o roteamento totalmente sobre as células. Este roteador é baseado em um algoritmo *maze router* (JOHANN, 2001). O resultado do roteamento fornece uma descrição CIF das camadas envolvidas no roteamento.

5.4.7 Finalização do Leiaute

A finalização do leiaute realiza a integração do leiaute gerado pela matriz de células e o leiaute gerado pelo roteador. A Figura 5.17 mostra o leiaute gerado para uma matriz de células NAND formada por 13 células em cada uma das 8 bandas, com três posições livres, uma da banda superior e duas na banda inferior. A estrutura de interconexões gerada pelo roteador é mostrada no centro. Após a etapa de finalização do leiaute, as duas partes componentes do leiaute formam o leiaute final do circuito.

Algoritmo para encontrar conexão entre células adjacentes

Entrada: B = (b_1, ..., b_m),

Saída: O número de conexões entre células adjacentes realizado, armazenado na variável n_con.

1. n_con ⇐ 0; nets_adjacentes⇐ false;
2. Para todo *i* tal que 1≤ *i* ≤ m faça
    a. Para todo *j* tal que 1≤ *i* ≤ *j -1 faça*
        i. N_{origem} ⇐ Nc_j
        ii. deslocamento ⇐ 1
        iii. Se tipo ($c_{j+deslocamento}$) = Dummy então:
            1. deslocamento ⇐ deslocamento +1
        iv. $N_{destino}$ ⇐ N($c_{j+delocamento}$)
        v. Para todo *r* tal que 1≤ *r* ≤ p_{origem}
            1. Para todo *s* tal que 1≤ *s* ≤ $p_{destino}$
                a. Se nome (n_r) = nome (n_s) então:
                    i. Se conecta_células() = true então:
                        1. nets_adjacentes ⇐ true
                        2. break
                    ii. Fim se
                b. Fim se
            2. Fim para
            3. Se nets_adjacentes = true
                a. j= j+1
                b. break
            4. Fim se
        vi. Fim para
    b. Fim Para
3. Fim para

Figura 5.15: Algoritmo para encontrar células adjacentes

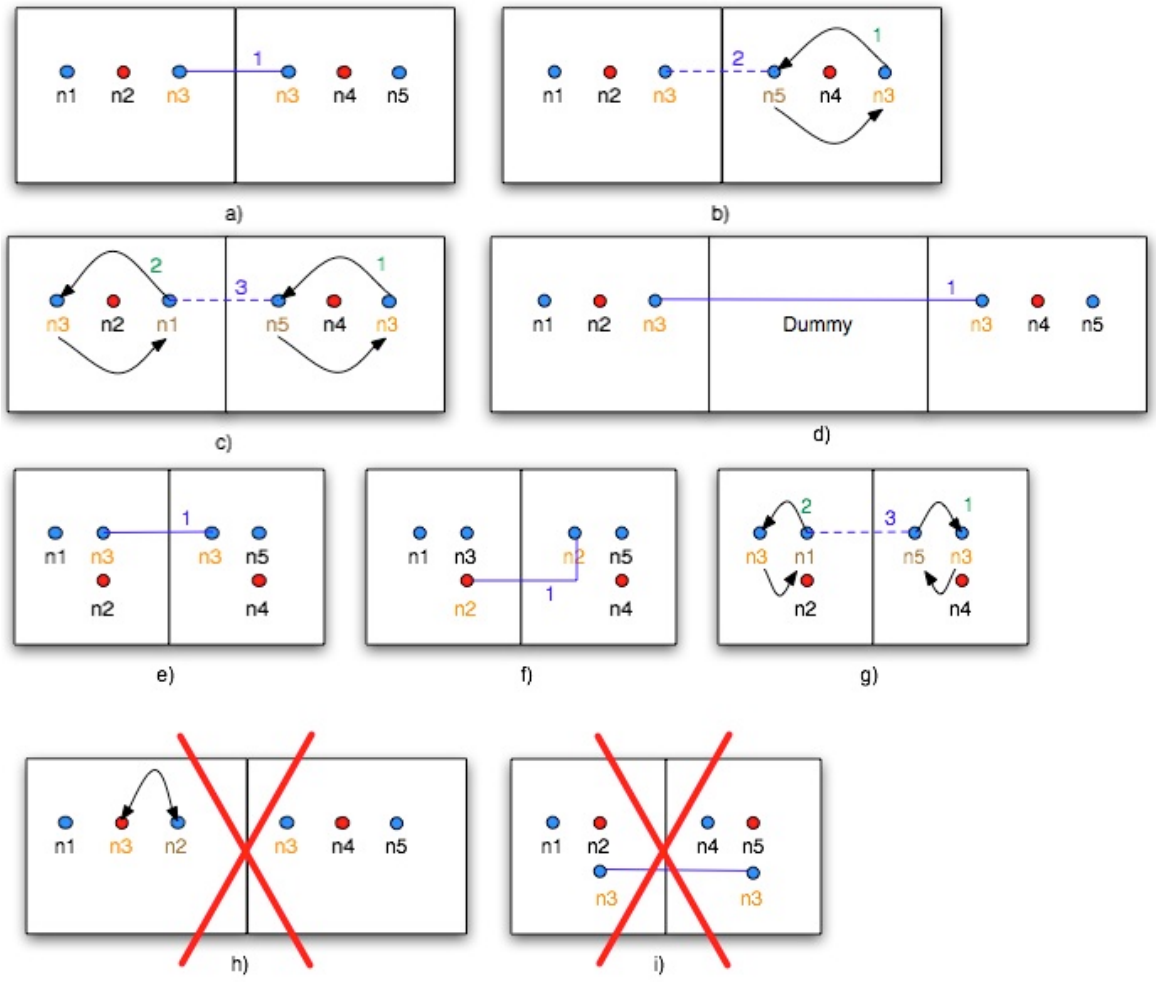


Figura 5.16: Possibilidades de conexões entre células adjacentes e reordenamento de *nets*. Cada célula possui dois pinos de entrada e um de saída. Os pinos de entrada são representados em azul e o pino de saída em vermelho. As alternativas h) e i) não são permitidas.

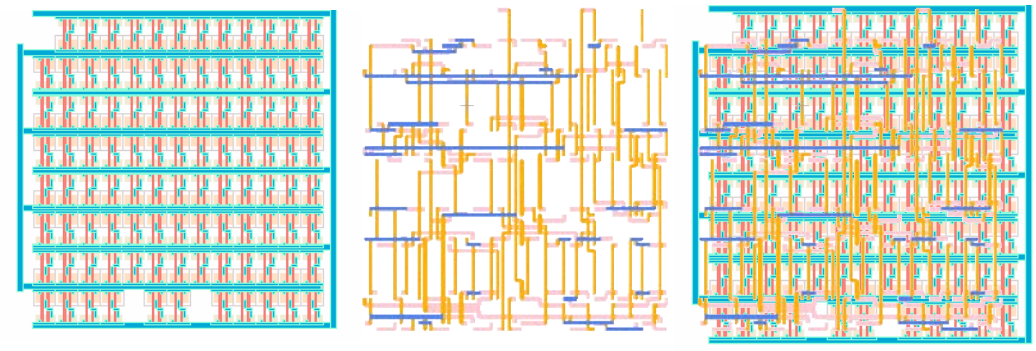


Figura 5.17: Matriz R-CAT de células NAND 13x8, a estrutura de conexões para customização do circuito e o leiaute final do circuito.

5.5 Caracterização das Células

A caracterização das células utilizadas nos projetos visa manter uma forma de documentação padrão entre todas as bibliotecas de células, de forma que novos projetos possam utilizar as bibliotecas disponibilizadas e novas células possam ser facilmente adicionadas às bibliotecas existentes (CARRO, 1987). As informações das células conterão dados de identificação, topológicos e elétricos. O arquivo de caracterização deverá apresentar os seguintes itens:

a) Nome: Nome da célula pelo qual ela será identificada no projeto, tanto na síntese lógica como na síntese física.

b) Função: descrição da função lógica da célula.

c) Tecnologia: fornece a tecnologia utilizada no projeto da célula.

d) Regras: indica o fabricante ou fornecedor das regras.

e) Largura: dimensão da largura final da célula básica.

f) Área: área final da célula básica dada pela largura vezes a altura.

g) Consumo: análise de consumo extraída da simulação elétrica da célula.

h) Atraso intrínseco: extraído da simulação elétrica conforme caracterização.

i) Capacitância de entrada: A capacitância de entrada é proporcional à capacitância da porta (*gate*) dos transistores de entrada. A capacitância dos transistores de entrada é proporcional à capacitância do óxido que separa o polisílicio do substrato, à largura do canal e ao comprimento do canal do transistor, como mostrada abaixo. Pode ser estimada conforme caracterização e simulação.

$$C_g = C_{ox} \times W \times L$$

Onde:

C_g: capacitância da porta

C_{ox}: capacitância do óxido

W: largura do canal do transistor

L: comprimento do canal do transistor

j) Resistência de saída: resistência de *pull-up* e *pull-down* estimadas. A resistência de *pull-up* é calculada no momento de subida (carga do capacitor de saída) de acordo com o atraso na subida ($T_{\rho LH}$). A resistência de *pull-down* é calculada na descarga do capacitor de acordo com o atraso na descida ($T_{\rho HL}$). A fórmula de cálculo para a resistência de *pull-up* e a resistência de *pull-down* é fornecida abaixo (RABAEY, 2003).

$$R_{pull-up}(s) \cong \frac{t_{pLH}}{C_{load} \times \ln(0,5)} \qquad R_{pull-down}(s) \cong \frac{t_{pHL}}{C_{load} \times \ln(0,5)}$$

Onde:

$R_{pull\text{-}up}$ (s) : Resistência de *pull-up* da saída s

$R_{pull\text{-}down}$(s): Resistência de *pull=down* da saída s

t_{pLH} : atraso na subida do sinal s

t_{pHL} : atraso na descida do sinal s

C_{load} : capacitância na saída s

ln (0,5) é uma constante de aproximação

k) *Fan-out*: o *fan-out* de uma porta *p* é definido pela carga associada às entradas das portas conectadas à porta em questão. Quando as cargas das portas a serem conectadas à saída da porta *p* são semelhantes, o fan-out é referenciado, muitas vezes, ao número de portas conectadas na saída da porta *p*.

l) Nomes das E/S: nomes das entradas e saídas das células.

m) Coordenadas de E/S: posição dos pinos de entrada e saída relativa as trilhas horizontais e verticais internas da célula.

n) Localização de armazenamento: identificação da estrutura de armazenamento onde está armazenado arquivo de descrição SPICE da célula.

o) Autor: identificação do responsável pelo projeto da célula.

p) Data: data de inserção da célula na biblioteca.

q) Comentários Gerais: reservado para descrição de características adicionais da célula descrita, tais como, condições especiais de funcionamento, tipo de lógica, etc.

A caracterização das células é um processo adicional e optativo. Caberá ao projetista da biblioteca de células identificar a importância futura da caracterização e realizar os procedimentos necessários para a obtenção dos valores.

As ferramentas de estimativas, tais como de *timing* e *cell sizing*, poderão fazer uso destas informações, principalmente para fornecer estimativas do comportamento do circuito.

5.6 Utilização da Ferramenta

A Figura 5.18 mostra as opções disponibilizadas na ferramenta R-CAT. São disponibilizadas as seguintes cinco opções de utilização da ferramenta. A opção *default* é utilizada para a geração de matrizes customizáveis por todas as camadas. As demais opções são para a geração de etapas das matrizes customizáveis por algumas camadas. A ferramenta foi desenvolvida em C++.

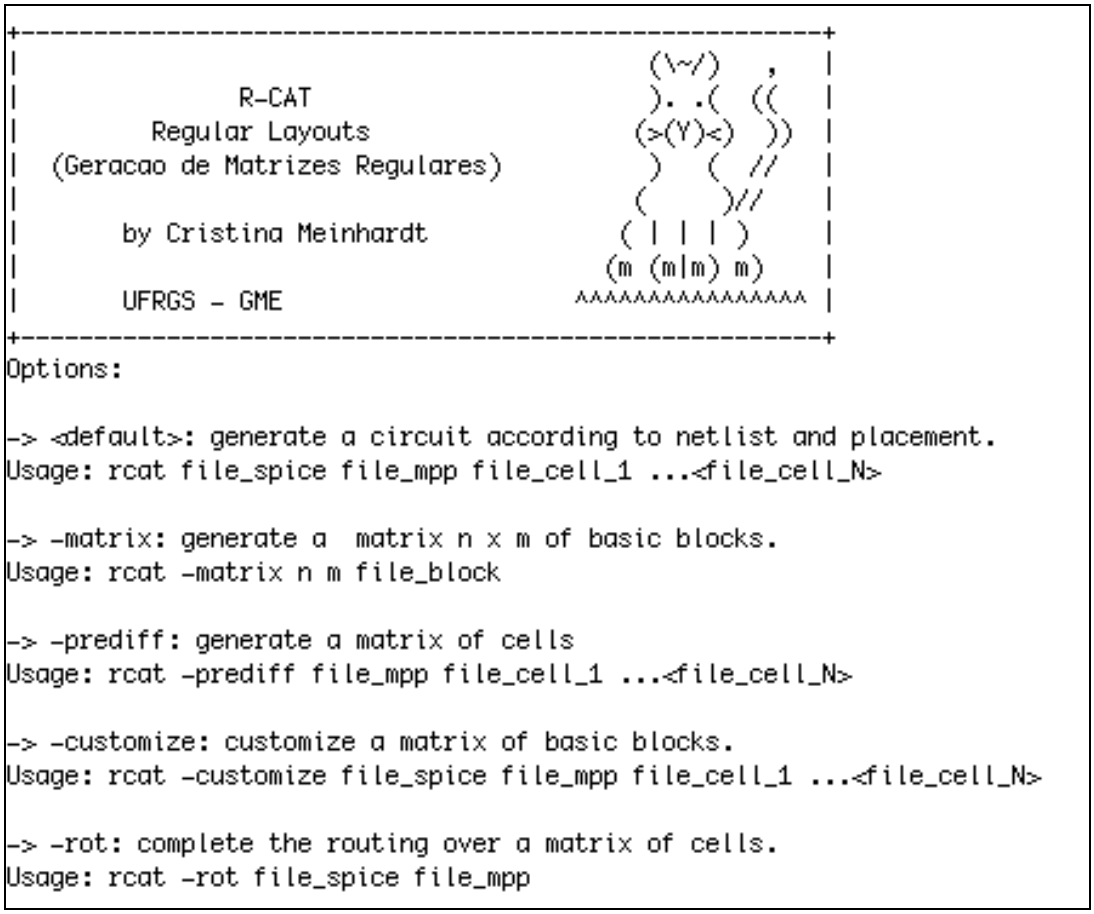


Figura 5.18: Opções da ferramenta R-CAT

A seguir são descritos os parâmetros passados em cada uma das opções:

- <default>: Nessa opção, são necessários os arquivos de descrição do *netlist,* do posicionamento e de descrição das células básicas adotadas.
- *-matrix*: Nessa opção, o número de colunas da matriz é representado pelo número inteiro *n* e corresponde ao número de células em cada linha. Cada linha da matriz, é uma banda do circuito e é representada pelo número inteiro *m*. O arquivo de descrição do bloco básico utilizado apresenta a descrição CIF do bloco básico.
- *-prediff*: Nessa opção é necessária a descrição da posição das células básicas componentes da matriz e a descrição CIF de cada célula lógica utilizada.
- *-customize*: Nessa opção, são necessários os arquivos de descrição do *netlist,* do posicionamento e de descrição das células básicas adotadas.
- *-rot*: Nessa opção, além da descrição das interconexões entre as células, é necessário o arquivo de posicionamento das células para geração da matriz de pontos a ser passada para o roteador.

A Figura 5.19 mostra o leiaute gerado após a utilização do programa R-CAT para a geração de uma matriz de blocos de dimensões 30 * 30. Esta matriz foi gerada com a seguinte chamada do programa: rcat -matrix 30 30 block.cif

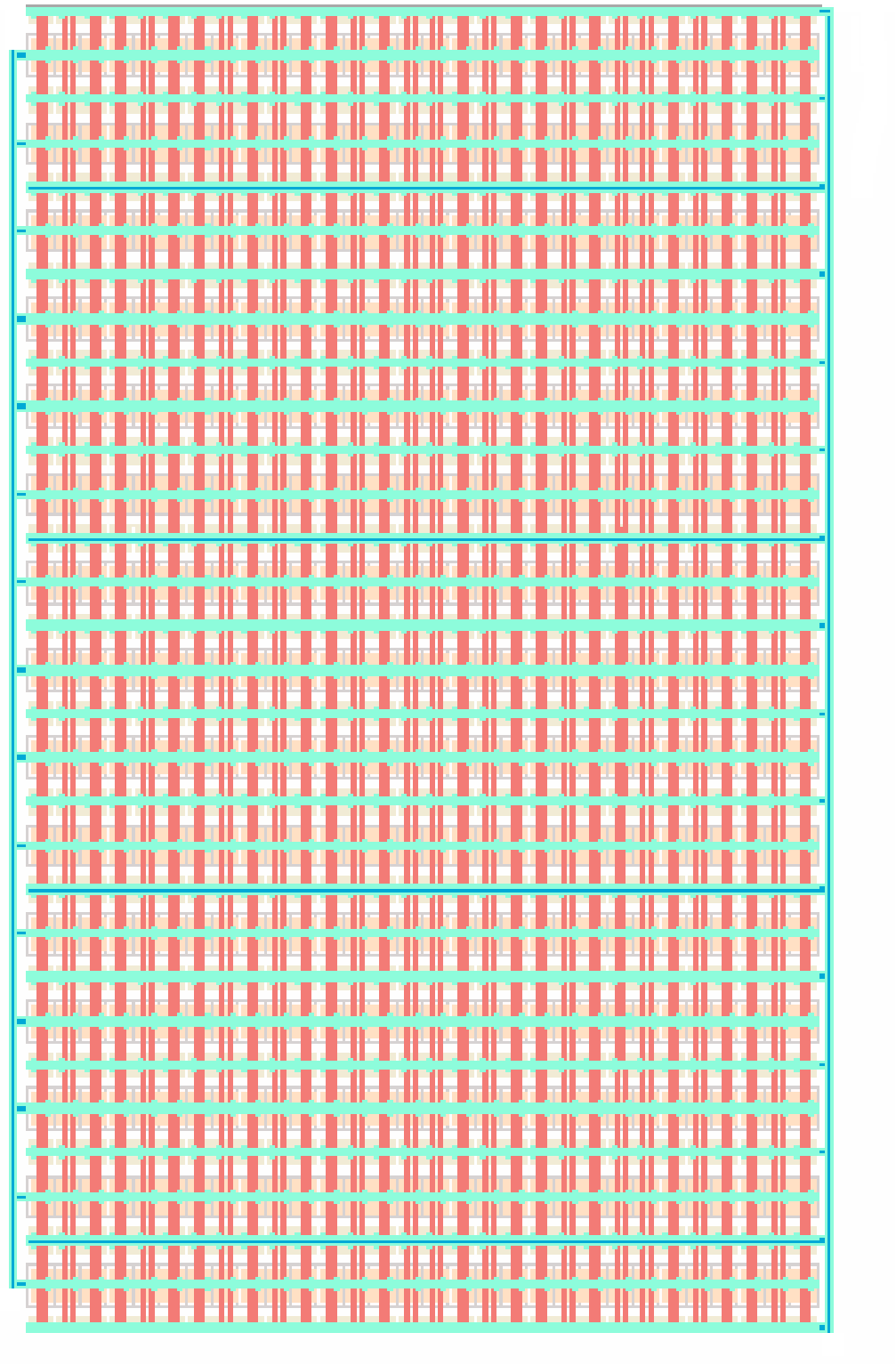


Figura 5.19: Matriz de blocos básicos 30 células por 30 bandas

6 INTERFERÊNCIA DO POSICIONAMENTO DOS PINOS DE ENTRADA E SAÍDA DAS CÉLULAS BÁSICAS NO ROTEAMENTO

A ferramenta de geração de matrizes regulares foi utilizada para a síntese de leiautes regulares NINA, compostos pelos blocos básicos NAND, NOR e Inversor. Estas células básicas são células de leiaute simples, com máximo de duas entradas, e com leiaute similar, como exposto anteriormente.

Um estudo de caso foi proposto para a avaliação da influência no roteamento da posição interna dos pinos de entrada e saída das células básicas. Esta questão foi inferida por experiências anteriores utilizando a síntese com a ferramenta Martelo, onde o roteador empregado possuía uma alta taxa de *nets* não roteadas, sendo necessária a inserção de trilhas adicionais entre as células para concluir o roteamento. Esta solução representa um aumento na área final da matriz, evidenciado pela necessidade de manter as células alinhadas com as células vizinhas na direção da trilha inserida. A dúvida remanescente era se a dificuldade de roteabilidade era devido à proximidade dos pinos de entrada e saída da célula NAND utilizada na ferramenta Martelo (ver Figura 5.11).

Uma experiência inicial consistiu na exploração de possibilidades de posicionamento para os pinos de entrada e saída em uma célula básica NAND. Foram realizadas experiências com duas configurações de tamanho de célula NAND, sendo a configuração 1 com Wp aproximadamente igual à Wn e a configuração 2 com uma célula NAND com Wp aproximadamente igual três vezes Wn. Para a configuração 2, cinco experimentos variando a posição dos pinos foram efetuados, enquanto que para a configuração 1 foram realizados somente três experimentos devido ao seu tamanho não possibilitar outras composições de pinagem diferentes. Nas duas configurações, as células foram projetadas com tecnologia 0.35 μm, com três níveis de metal para roteamento. A largura da célula NAND da configuração1 é de 6μm por 9μm de altura. Enquanto na configuração2, a célula NAND é projetada com 16,5 μm de altura e 6μm de largura.

A Figura 6.1 exibe os cinco experimentos determinados para a célula NAND da configuração 2. A primeira configuração alinha os três pinos da interface da célula na mesma trilha horizontal. A segunda e quarta configurações mantêm os dois pinos de entrada na mesma trilha vertical, deslocando a posição do pino de saída para outra trilha horizontal. A terceira configuração coloca cada pino em uma trilha horizontal. Todas as configurações anteriores adotam uma trilha vertical para cada interface da célula. A quinta configuração adota o posicionamento da pinagem semelhante ao da ferramenta Martelo, com uma trilha vertical compartilhada entre um dos pinos de entrada e o pino

de saída, enquanto os dois pinos de entrada estão posicionados na mesma trilha horizontal.

É possível observar na Figura 6.1 que o leiaute da célula é o mesmo para todas as configurações, deste modo não serão observados impactos na área final do circuito gerado. A modificação de uma configuração para a outra neste estudo de caso foi realizada somente através da alteração do arquivo de configuração da matriz, onde é determinada a posição dos pinos das células básicas para a geração da matriz e geração da matriz de pontos para o roteador.

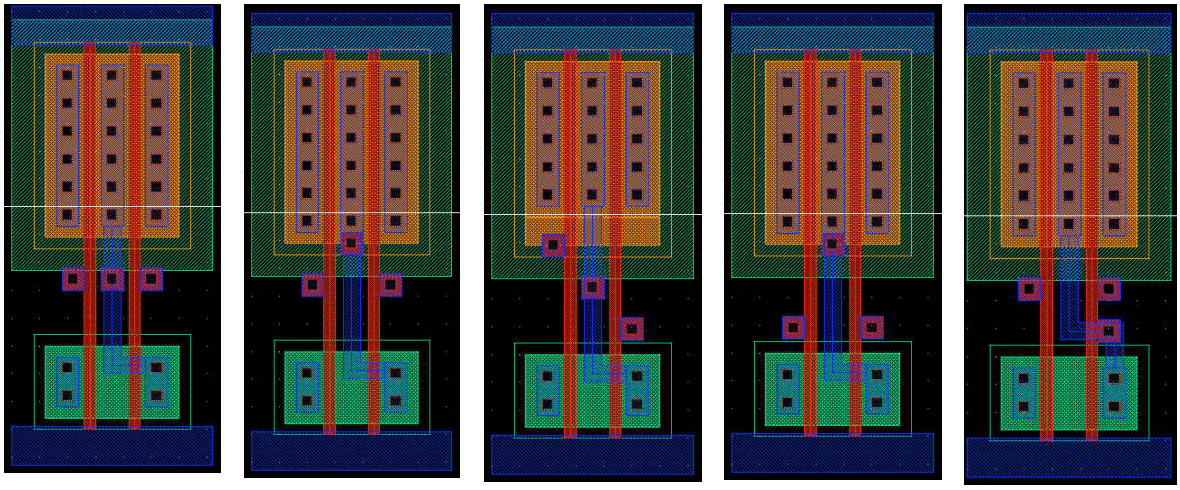


Figura 6.1: Alternativas exploradas para o posicionamento dos pinos nas Configurações 1 e 2

A metodologia adotada no estudo de caso inicia com a geração da descrição do leiaute para um conjunto de circuitos utilizando as cinco configurações de pinagem determinadas para os dois tipos de células NAND (menor e maior) e a posterior observação da dificuldade de roteamento encontrada mantendo constante o número de camadas de metal disponível e não permitindo a inserção de trilhas. Os circuitos escolhidos para o estudo de caso são circuitos onde era necessária a inserção de trilhas verticais ou horizontais para roteamento total das *nets* com a ferramenta Martelo. Os resultados observados e comparados são a porcentagem de *nets* não roteadas e a porcentagem de grupos não roteados. Todos os circuitos foram submetidos às mesmas ferramentas de síntese lógica, posicionamento, geração do leiaute e roteamento. Não foram realizadas modificações no netlist dos circuitos para diferentes configurações de pinagem. Em cada configuração foram levantados os dados para comporem tabelas semelhantes à Tabela 6.1. Nelas são armazenados os dados dos resultados obtidos para os seis circuitos utilizados em cada experimento de cada configuração. As três colunas iniciais descrevem o circuito, fornecendo o nome, o número de células lógicas e o número de *nets*. A quarta coluna indica o número de *nets* não roteadas sem a inserção de trilhas extras para roteamento. A quinta coluna representa a porcentagem das *nets* totais do circuito que não foram roteadas. A última coluna informa a fiação obtida. É importante salientar que este resultado de fiação informa somente a quantidade de fiação necessária para realizar as conexões realizadas pelo roteador.

Os resultados obtidos em cada experimento permitem a composição de tabelas comparativas da percentagem de *nets* não roteadas em cada experimento de cada configuração. Com base nestas informações, é possível determinar se algum dos

experimentos de posicionamento da pinagem interna das células influenciou positivamente na capacidade de roteamento do roteador. Uma informação adicional foi inserida nestas tabelas para indicar a média de *nets* não roteados em cada experimento.

Tabela 6.1: Resultados obtidos para o Experimento 1 da Configuração 1

Circuito	Células	Nets	Não roteadas	%	Fiação
alu4	3230	3248	462	14,2241	1,90E+07
x1	5338	5393	1156	21,4352	3,25E+07
x3	3301	3440	423	12,2965	1,95E+07
apex3	5563	5621	1171	20,8326	3,38E+07
apex6	3293	3432	406	11,8298	1,95E+07
apex7	2209	2262	237	10,4775	1,29E+07

Observando os resultados obtidos para a configuração 1 fornecidos na Tabela 6.2 podemos observar uma alta taxa de *nets* não roteadas, sendo aproximadamente 15 % das *nets* com três camadas de metal e sem a inserção de trilhas. Além disso, observa-se que não existe uma diferença significativa de roteabilidade entre os experimentos.

Tabela 6.2: Porcentagem de *nets* não roteadas com a Configuração1

Circuito	exp1	exp2	exp3
Alu4	14,2241	16,441	14,501
x1	21,4352	19,5995	20,712
x3	12,2965	14,4186	12,0349
apex3	20,8326	20,3167	21,651
apex6	11,8298	14,3357	12,2669
apex7	10,4775	12,29	10,8753
média	15,1826	16,234	15,340

Os resultados dos experimentos realizados com a configuração 2, mostrados na Tabela 6.3 apresenta menor taxa média de *nets* não roteadas em comparação com os resultados obtidos para a configuração 1 devido ao maior número de trilhas horizontais disponibilizados na área dessa célula. Entretanto, também não é possível identificar um experimento que favoreça o roteamento com a sua configuração interna dos pinos,

sendo alguns experimentos melhores para alguns circuitos, mas não melhores de um modo geral.

O experimento foi repetido para um segundo conjunto de circuitos gerados por uma ferramenta de síntese lógica que gera circuitos compostos por células NAND, NOR e Inversores. O tamanho de célula adotado neste experimento é igual para todas as células, considerando que a célula inversora ocupará menos espaço em largura. Cada posição da matriz é capaz de receber uma célula de 10,5 μm de altura e 6 μm de largura. Nesse experimento são adotadas as mesmas condições do primeiro experimento: os circuitos utilizados em todas as configurações passaram pelas mesmas ferramentas de síntese física, usando as mesmas configurações das ferramentas. Somente foi modificado o arquivo de configuração para representar corretamente a posição dos pinos de entrada e saída.

Tabela 6.3: Porcentagem de *nets* não roteadas com a Configuração 2

Circuito	Exp1	Exp2	Exp3	Exp4	Exp5
Alu4	6,6503	6,6503	6,6503	6,6503	7,6663
X1	5,2475	5,2475	6,1747	5,3403	6,5084
X3	2,4128	2,4128	3,6337	2,9942	3,5174
Apex3	7,1873	7,1873	8,0947	7,3830	8,8241
Apex6	2,6224	2,6224	3,2925	2,7098	3,1760
Apex7	1,9010	1,9010	2,2989	1,6357	2,2104
Média	*4,3369*	*4,337*	*5,024*	*4,4522*	*5,3171*

Nesta configuração são adotadas cinco alternativas de posicionamento interno dos pinos de entrada e saída das células, como mostra a Figura 6.2. A primeira alternativa explorada no experimento 1 é o posicionamento de todos dos pinos na mesma trilha horizontal, mostrada na alternativa mais à esquerda na figura. O experimento 2 adota o posicionamento dos pinos semelhante ao adotado no gerador Martelo. O terceiro experimento posiciona os pinos intercalando trilhas livres entre eles, como mostrado na alternativa central da figura. O quarto experimento adota pinos distribuídos em uma linha diagonal, com o pino de saída no centro da célula. O experimento 5 explora um posicionamento dos pinos onde não há compartilhamento de trilhas nem vertical nem horizontal, mostrado na alternativa mais à direita da figura. Os pinos apresentam a mesma configuração nas células NAND e NOR. Nas células inversoras, os pinos correspondem às posições determinadas para cada um dos pinos de entrada e para o pino de saída, respectivamente, considerando que este deve estar compreendido dentro da área da célula inversora.

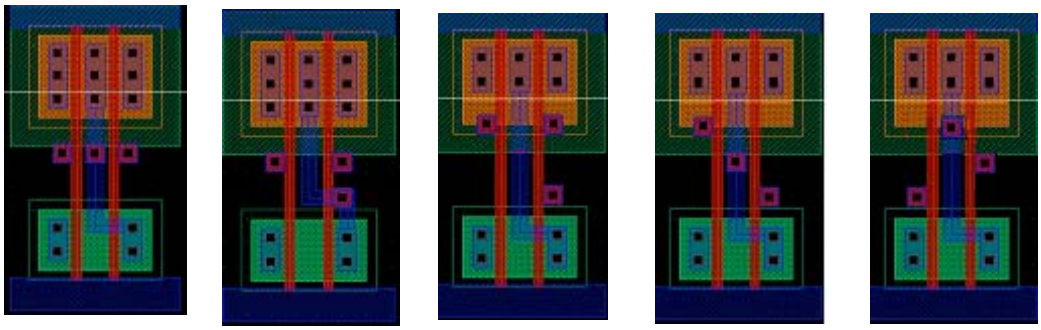


Figura 6.2: Alternativas exploradas para o posicionamento dos pinos na configuração 3

As características dos circuitos utilizados são apresentadas na Tabela 6.4, onde são detalhados o número de células, o número de pinos do circuito e o número de *nets* gerados pela síntese lógica.

Tabela 6.4: Característica dos circuitos utilizados na Configuração 3

Circuito	Células	Pinos	Nets
9symml	461	18	469
Alu2	705	16	706
Alu4	2344	22	2309
Apex1	3653	90	3602
Apex2	1207	42	1228
Apex3	3851	104	3828
Apex7	848	86	893
C8	219	46	247
Clipe	470	14	472
Cordic	259	25	279
Count	256	51	275
Duke2	1002	51	1015
E64	317	15	382
Example2	599	151	682
Frg1	291	31	310
Lal	217	45	242
Pcle	109	28	128
Pcler8	224	44	251
T481	189	17	202

Term1	631	44	657
Ttt2	373	45	395
Unreg	182	52	218
X1	951	86	994
X2	114	17	133
X3	1520	234	1637
X4	849	165	935
Z4ml	102	11	109

Os resultados de fiação obtidos para os circuitos em cada um dos experimentos são mostrados na Figura 6.3. Assim como observado na Configuração 1 e 2, as alternativas de posicionamento dos pinos exercem pouca influência no *wire length* final do circuito, sendo que o experimento com melhor resultado depende do circuito avaliado. É importante salientar que a medida de *wire length* gerada para os circuitos não roteados considera apenas as *nets* totalmente conectadas, não tendo uma boa precisão.

Os circuitos onde não foi possível completar o roteamento sem a inserção de trilhas adicionais são apresentados na Tabela 6.5. Para cada experimento é detalhado o percentual de nets não roteadas. Novamente observa-se a semelhança entre os resultados obtidos. Tal semelhança é melhor destacada na Figura 6.4, onde são expostos os percentuais de nets não roteadas para experimento de cada circuito da Tabela 6.5.

Tabela 6.5: Taxa de *nets* não roteadas em cada experimento

Circuito	exp1	exp2	exp3	exp4	exp5
apex1	0,11%	0,11%	0,11%	0,11%	0,11%
apex3	0,13%	0,13%	0,13%	0,13%	0,16%
apex7	0,22%	0,23%	0,23%	0,23%	0,11%
c8	2,02%	2,43%	2,43%	2,43%	2,43%
cordic	0%	0%	0%	0%	0,36%
count	3,27%	3,27%	3,64%	2,91%	2,91%
e64	6,28%	5,50%	5,50%	5,50%	5,24%
example2	5,43%	5,87%	6,07%	5,62%	4,77%
frg1	0,32%	0,32%	0,65%	0,65%	0,65%
lal	1,65%	1,29%	1,29%	1,29%	0,83%
pcle	1,56%	0%	0%	0%	0%
pcler8	0,40%	0,40%	0,40%	0,40%	0,40%
term1	0,46%	0,46%	0,46%	0,46%	0,30%
ttt2	0,25%	0,25%	0,26%	0,26%	0,25%
unreg	3,67%	4,59%	5,05%	4,59%	4,13%

x1	0,30%	0,30%	0,30%	0,30%	0,20%
x3	3,05%	3,24%	3,07%	3,07%	2,99%
x4	2,57%	2,89%	2,70%	2,70%	2,35%

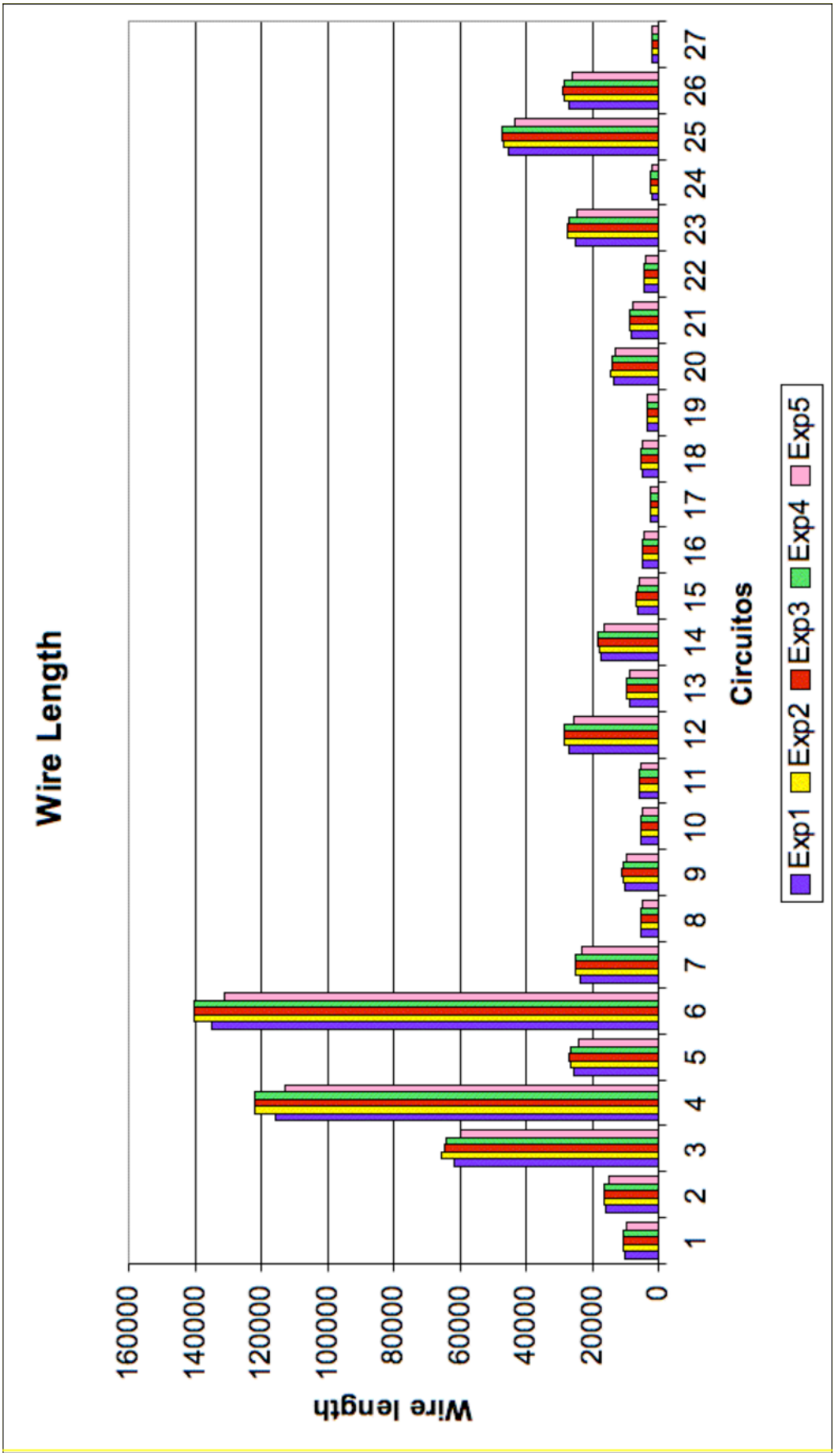


Figura 6.3: Resultado de *Wire length* para os circuitos da Configuração 3

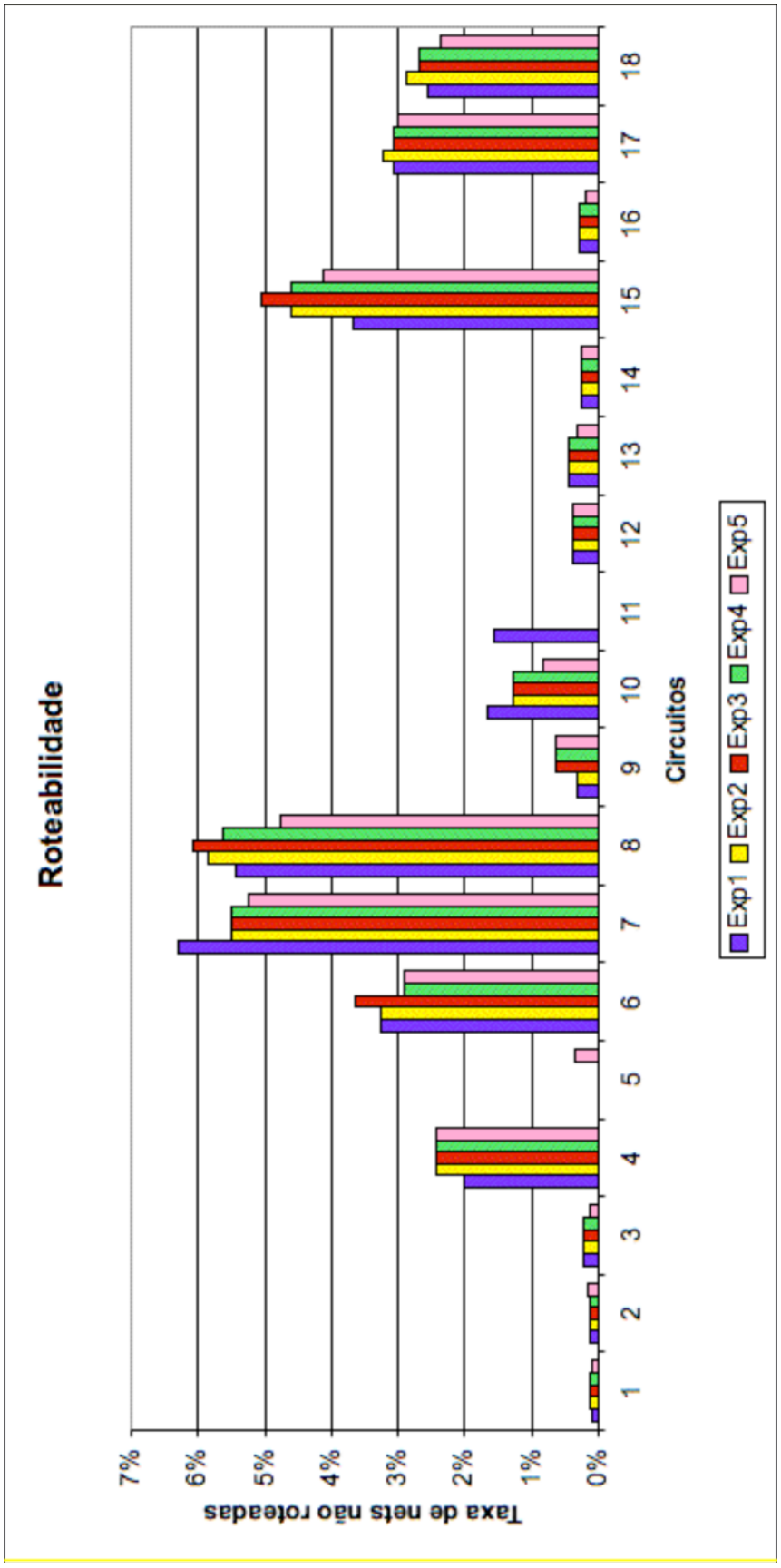


Figura 6.4: Resultados de roteabilidade para a Configuração 3

Estes resultados indicam que a posição interna dos pinos em células básicas de funções lógicas pequenas (envolvendo até duas variáveis) não tem grande influência na capacidade de roteamento total de circuitos em matrizes regulares, pelo menos considerando os leiautes das células utilizadas. Tais resultados consideram a utilização de um algoritmo *maze router* para roteamento.

7 RESULTADOS

A geração de leiautes no fluxo de síntese proposto está altamente relacionada com as escolhas realizadas nas etapas anteriores do fluxo. Decisões tomadas na síntese lógica afetam diretamente o tipo de lógica adotado, o número de portas, o número de interconexões, o tamanho das interconexões e o fan-out das redes. A área e a potência estão, principalmente, relacionadas às decisões sobre tipo de lógica e ao número de portas gerado, enquanto os três últimos fatores influenciam nos resultados de *wire length* e *timing* do circuito. Por outro lado, muitas das alternativas exploradas para aumentar a regularidade e previsibilidade dos circuitos podem aumentar a área e a potência, quando comparados a leiautes *full-custom* ou *Standard Cell*, onde cada célula ou bloco de circuito é projetado de forma a ocupar a menor área possível, de modo a atender às restrições de *timing* e potência. Dentre as alternativas de regularidade exploradas, destacam-se, neste projeto, a opção de priorizar a utilização de linhas retas de poly e de não adotar tamanhos mínimos no projeto das células básicas. Estas alternativas aumentam a área do circuito, entretanto, espera-se que o aumento no yield compense o custo de aumento de área.

A estreita relação entre a síntese lógica e o leiaute final motivaram o estudo do comportamento de diferentes modelos de sínteses lógica aplicadas no gerador R-CAT. As comparações das descrições geradas pelas ferramentas de sínteses lógicas têm como objetivo entender qual síntese lógica é mais apropriada para atender diferentes restrições de projeto. Por exemplo, alguns projetos podem priorizar resultados de *timing* ao invés de área, enquanto outros podem preferir a adoção de sínteses com menor *fan-out* ou que resultem interconexões mais curtas.

Foram analisados três métodos diferentes de síntese lógica, possíveis de serem adotados no projeto de circuito usando matrizes R-CAT. A primeira síntese é obtida com a ferramenta SIS, adotando o script *rugged* para minimização de número de portas lógicas. O mapeamento da descrição é restrito a uma biblioteca mínima, compatível com as portas lógicas adotadas na geração do R-CAT e composta pelas células básicas: células NAND, NOR, Inversores e Buffers. A segunda e a terceira sínteses lógica são geradas pelo mapeamento direto de estruturas orBDD. A segunda síntese, orBDD_Nand, restringe as células utilizadas somente para células NAND e inversores. Esta síntese é apropriada para implementação de circuitos com a ferramenta Martelo, onde somente são utilizadas células NAND, inclusive para a geração de inversores. A terceira síntese realiza o mapeamento das estruturas para células NOR, NAND e Inversores. Esta síntese recebe o nome orBDD_NINA (orBDD mapeado para NORs, Inversores e NANDs).

A comparação entre os métodos de síntese lógica aborda os seguintes aspectos: número de células lógicas, número de nets e análise de timing. A análise de timing foi

realizada pela ferramenta Prime Time da Synopsys. Todos os métodos foram mapeados para as mesmas células da biblioteca Standard Cell AMS 0.35 μm, com os parâmetros fornecidos pela Cadence. Na comparação foram utilizadas mais duas ferramentas de síntese lógica: SIS- Cadence e orBDD- Cadence. Cada uma destas ferramentas foi gerada a partir das descrições iniciais SIS e orBDD, entretanto, foi utilizada a ferramenta Pks_Shell para otimizar a lógica, permitindo livre mapeameanto para células complexas, disponíveis na bliblioteca do Kit Cadence AMS 0.35. As comparações com resultados de circuitos *Standard Cell* são apenas para fornecer uma referência de valores encontrados com outra metodologia de projeto. Essa referência é adotada para situar os valores encontrados nas gerações com as ferramentas Martelo e R-CAT.

A Tabela 7.1 apresenta a comparação do número de células gerado pelas ferramentas de síntese lógica. Pode-se observar que a liberdade de mapeamento reduz consideravelmente o número de células independente da síntese lógica inicial. A síntese lógica efeituada pelo SIS resulta 48% menos células que a síntesea baseada em orBDD. O mesmo comportamento acontece quanto ao número de nets gerados pelas ferramentas. Os resultados de números de nets podem ser observados na Tabela 7.2. As sínteses orBDD apresentam o dobro de nets em média.

Tabela 7.1: Números de Células

Circuito	SIS		orBDD		
	Rugged	Cadence	NINA	NAND	Cadence
Alu2	367	163	705	742	220
Alu4	1066	470	2344	2440	718
Apex2	472	152	1207	1204	218
Clip	205	94	470	487	101
Cordic	200	27	259	287	40
Count	153	95	256	306	116
Duke2	490	240	1002	1030	240
Example2	348	188	599	682	199
Frg1	223	76	291	310	80
X1	400	177	951	1045	195

No entanto, a síntese lógica baseada em orBDD é efetuada com controle do fan-out. As células que compõem um cluster possuem fan-out igual a 1, e somente as nets de saída dos clusters atingem várias portas (TAVARES, 2006). Deste modo, o fan-out médio das sínteses baseadas em orBDD é 15% menor que o gerado pelo SIS e 47% menor que o gerado pelas sínteses livremente mapeadas para células complexas. A diferença entre os fan-outs gerados pelas sínteses orBDD_Nand e orBDD_NINA é de 2%. O controle do fan-out também afeta a capacitância das portas lógicas. A ferramenta PrimeTime fornece as capacitâncias das portas utilizadas, o que é mostrado na Figura

7.2. Em média, as capacitâncias das portas lógicas em sínteses usando orBDD são 45% menores que as capacitâncias das portas na síntese SIS e 130% menores que as capacitâncias das portas complexas mapeadas com o Cadence.

Tabela 7.2: Número de Nets

Circuito	SIS		OrBDD		
	Rugged	Cadence	NINA	NAND	Cadence
Alu2	377	172	706	748	237
Alu4	1080	477	2309	2448	739
Apex2	510	190	1228	1211	264
Clip	214	104	472	490	117
Cordic	223	53	279	292	70
Count	188	131	275	308	158
Duke2	512	259	1015	1037	269
Example2	433	272	682	687	291
Frg1	251	106	310	322	115
X1	451	227	994	1057	260

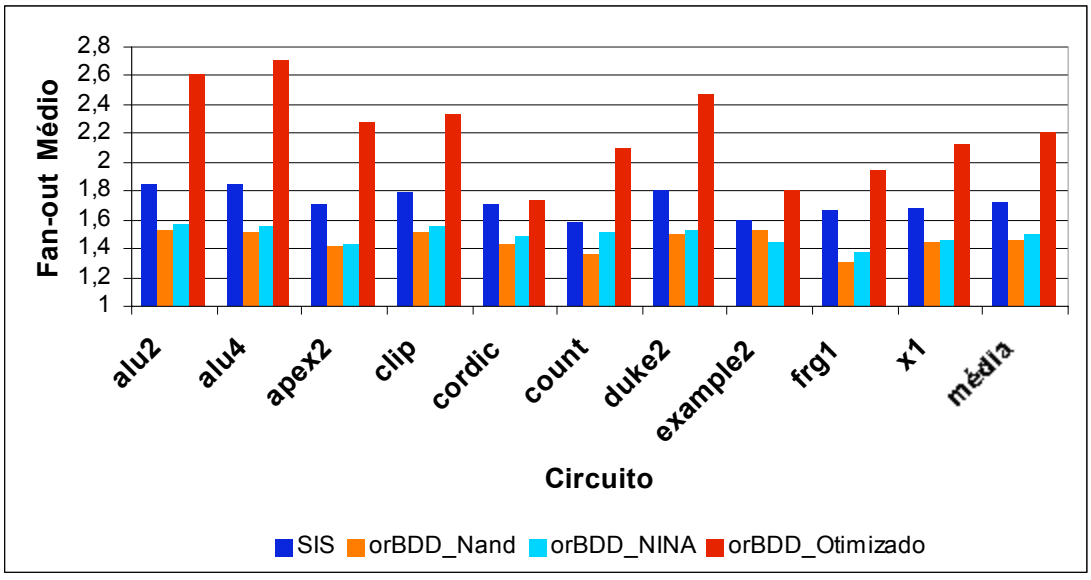


Figura 7.1: Fan-out médio dos circuitos nas ferramentas de síntese lógica avaliadas

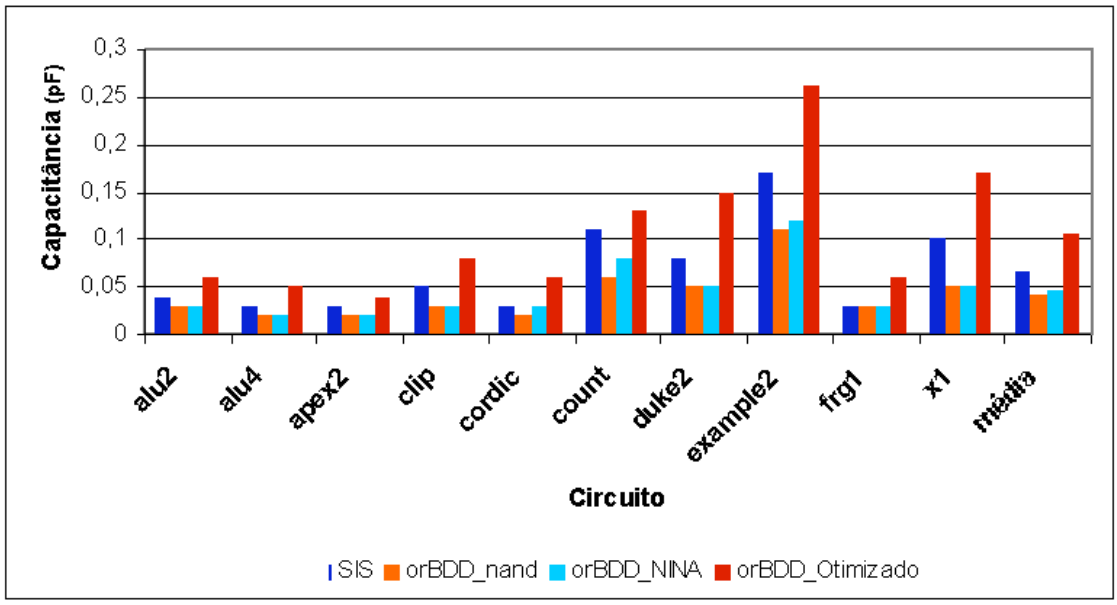


Figura 7.2: Capacitância média das portas nas ferramentas de síntese lógica avaliadas

A análise de timing dos circuitos gerados demonstrou que as sínteses lógicas baseadas em orBDDs apresentam os mesmos resultados médios de atraso dos circuitos. Esses resultados são 6% superiores aos encontrados para os circuitos livremente mapeados para células complexas pelo Cadence e 58% menores que os encontrados para os circuitos gerados pela síntese lógica da ferramenta SIS otimizados para área. Complementarmente, observa-se a importância de avaliar como seriam os resultados obtidos com a ferramenta SIS se a minimização adotada fosse de *delay*, por exemplo, obtida com o *script delay* disponibilizado junto com a ferramenta SIS.

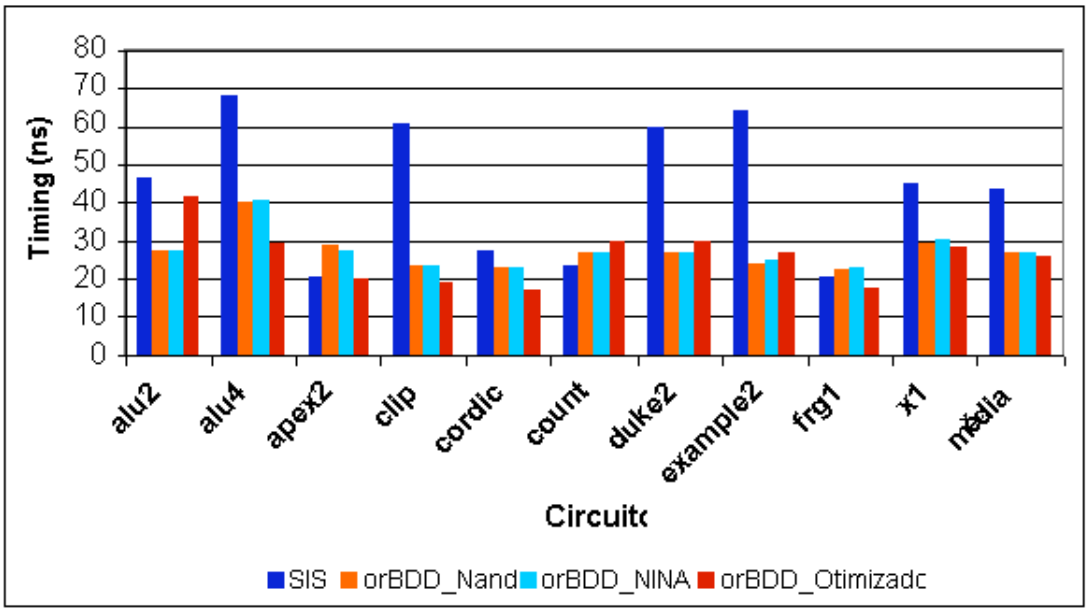


Figura 7.3: Análise de Timing dos circuitos gerados pelas sínteses lógicas avaliadas.

As ferramentas SIS e orBDD NINA foram utilizadas como entrada para geração de matrizes R-CAT. Todas as ferramentas respeitam as mesmas restrições para todas as

etapas restantes do fluxo de síntese. Os circuitos foram posicionados com o posicionador Mangoparrot configurado para livre alteração de posição das células e minimização de *wire length*. Após, as descrições de posicionamento e de interconexões foram utilizadas no gerador de matriz R-CAT, utilizando as células básicas NAND, NOR, Inversor e Buffer. Os circuitos foram roteados com três níveis de metal para roteamento e não foi permitida a inserção de trilhas extras pela ferramenta de geração da matriz. Os blocos básicos da matriz apresentam os pinos de entrada e saída dispostos conforme o experimento 5 da configuração 3, apresentados no Capítulo 6. As células foram dimensionadas para apresentarem tamanhos compatíveis com células da biblioteca *Standard Cell* do Cadence. Deste modo, o bloco básico tem área de 63 μm^2, com largura de 6 μm e altura de 10,5 μm. O tamanho dos transistores projetado é de Wp sendo o dobro de Wn. São permitidos três níveis de metal para o roteamento. O nível de metal 1 é reservado para uso nas células e nas linhas de alimentação.

A síntese orBDD Nand foi implementada na matriz Martelo. As etapas de posicionamento e roteamento foram realizadas seguindo os mesmos procedimentos adotados para a geração na matriz R-CAT. A ferramenta Martelo foi utilizada nestes experimentos com os seguintes parâmetros: três camadas de metal para roteamento, sem inserção de trilhas, com o tamanho dos transistores P setado para quatro vezes o tamanho mínimo e o tamanho dos transistores N setado para duas vezes o tamanho mínimo. Estes parâmetros foram escolhidos visando a utilização do mesmo tamanho das células NAND utilizadas no R-CAT e no Cadence.

Os resultados de leiaute obtidos com o R-CAT, nas versões de síntese lógica orBDD_NINA e SIS, serão comparados com os resultados de leiaute dos mesmos circuitos aplicados no Martelo e em quatro versões *Standard Cell*: orBDD-SC, SIS-SC e SIS-SC-O e orBDD-SC-O. Esses circuitos *Standard Cell* utilizam as descrições lógicas iniciais SIS e orBDD e as sínteses otimizadas para área e mapeadas permitindo a utilização de células complexas.

A versão *Standard Cell* orBDD-SC adota a síntese lógica orBDD_NINA e restringe a ferramenta pks_shell da Cadence a realizar o mapeamento lógico para células equivalentes na biblioteca *Standard Cell* da tecnologia AMS 0.35 μm. O mapeamento das células foi realizado do mesmo modo para a versão *Standard Cell* SIS-SC.

A versão *Standard Cell* SC-O implementa o máximo de otimização lógica e o mapeamento livre para células complexas na ferramenta pks-shell da Cadence para a tecnologia AMS 0.35 μm. O posicionamento e roteamento dos circuitos *Standard Cell* foram realizados na ferramenta Seultra da Cadence com a tecnologia AMS 0.35 μm. No posicionamento foi permitida a inserção de espaços livres até atingir o total roteamento dos circuitos.

Os resultados de área obtidos com as sete alternativas de síntese são apresentados na Tabela 7.3. A comparação destes resultados será realizada comparando os resultados da ferramenta R-CAT contra as outras ferramentas. Observa-se que a síntese SIS resulta em uma matriz R-CAT 47% menor em área que a matriz R-CAT NINA, por possuir 48% menos células lógicas na síntese lógica. A matriz R-CAT SIS representa também uma redução de 83% de área comparada a área necessária para as matrizes Martelo e apenas 8% maior que o circuito *Standard Cell* gerado para a mesma descrição lógica. Os circuitos *Standard Cell* livremente mapeados para células complexas implicam na metade da área, sem características regulares e sem previsibilidade sobre os resultados de fabricação. A comparação entre a matriz R-CAT NINA e a matriz Martelo indica que

a matriz R-CAT é 14% maior que a matriz Martelo. Este resultado é justificado principalmente por dois fatores: primeiro, a síntese lógica orBDD NINA apresenta apenas 5% de células a menos que a síntese lógica orBDD Nand; segundo, a matriz Martelo compacta o leiaute de duas células NAND adjacentes. O segundo fator reduz a área das células básicas NAND , entretanto, reduz também a regularidade geométrica da matriz.

Tabela 7.3: Comparação dos resultados de área entre R-CAT, Martelo, Cadence e Cadence Otimizado (μm^2)

	OrBDD				SIS		
Circuito	Martelo	R-CAT	SC	SC-O	RCAT	SC	SC-O
Alu2	43671.2	48300	38986.3	17968.9	25500	24226.9	12961.8
Alu4	119735.7	161280	124220.5	55955.7	70560	64541.3	35853.4
Apex2	71016.4	85560	63176.7	17622.1	31920	29052.7	12354.3
Clip	29312	37920	25936.9	8271.7	13680	12961.8	7318.74
Cordic	18020	16380	15862.5	4269.9	13680	12961.8	3593.58
Count	18020	16380	15537.5	9535.1	10200	10069.8	7318.74
Duke2	60500.6	66420	53430.2	18700.2	33060	29980.6	19446.1
Example2	43136	40320	32851.5	15189.4	24480	21387.7	14220.5
Frg1	20370	18480	15862.5	7318.74	14400	14871.7	6424.02
X1	62418.4	64800	50963.06	15537.5	28080	24226.9	12961.8
Média	*48620.03*	*55584*	*43682.7*	*17036.9*	*26556*	*24428.1*	*13245.3*

Os resultados de área podem ser melhor visualizados na Figura 7.4. Nela, para cada circuito utilizado são apresentados os resultados de área para cada uma das sete diferentes alternativas avaliadas na Tabela 7.3. Cada modo de implementação recebeu uma representação colorida, onde foram atribuídos tons azulados para os resultados gerados a partir de orBDDs, e tons avermelhados os circuitos originados com síntese lógica fornecida pela ferramenta SIS. A ordem das barras do gráfico corresponde a ordem das colunas na Tabela 7.3. Tal notação será empregada nas próximas figuras deste capítulo. É possível observar que a geração da matriz R-CAT resulta em aumento de área em praticamente todos os circuitos. Este aumento é particularmente agravado nos circuitos Alu4 e Apex2.

A Tabela 7.4 apresenta os resultados de *wire length*. Os circuitos das matrizes R-CAT e Martelo foram roteados com o mesmo roteador. Os circuitos R-CAT SIS apresentam resultados de *wire length* 30% menores em relação ao R-CAT NINA e 20% maiores que a versão *Standard Cell* da mesma síntese lógica. As comparações com *Standard Cell* fornecem apenas uma referência, uma vez que circuitos Structured ASICs possuem maior área e, conseqüentemente, maior *wire length*. Porém, eles apresentam a vantagem de serem regulares com melhor *Design-For-Manufaturability*. Na Figura 7.5 nota-se que os resultados de *wire length* para as matrizes Martelo são maiores em todos os circuitos.

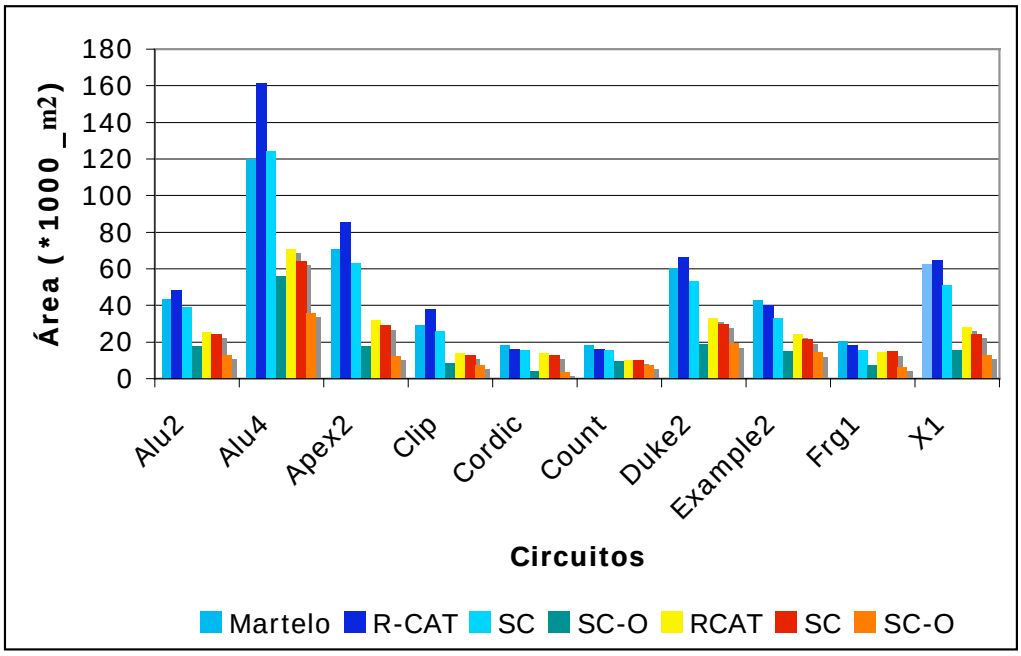


Figura 7.4: Resultados de área (μm^2)

Tabela 7.4: Comparação dos resultados de *wire length* obtidos com R-CAT, Martelo, Cadence e Cadence Otimizado (μm)

Circuito	orBDD				SIS		
	Martelo	RCAT	SC	SC-O	RCAT	SC	SC-O
Alu2	25794	22678.5	20008	16532.8	17437.5	15380.1	9977
Alu4	98926.5	89304	75867.2	71181.5	68080.5	60843.8	43029.7
Apex2	40966.5	36202.5	30731.2	12933.6	23335.5	18683.9	7932.2
Clip	15133.5	15121.5	11155.3	4795.1	7840.5	6950.2	4305.9
Cordic	8295	7660.5	5446.7	1652.7	7398	6151.3	1133.1
Count	9972	8262	6226.6	4777.7	5620.5	4181.9	2948.4
Duke2	52780.5	39811.5	27708.9	14639.25	27640.5	20882.2	15512.4
Example2	36975	24963	17687.4	9688.6	16588.5	13567.8	9851.95
Frg1	9426	9378	6163.8	3938.5	9262.5	7441	3306.5
X1	42742.5	36864	28623.3	12496.15	21244.5	15929.5	10444.5
Média	*34101.1*	*29024.5*	*22961.8*	*15263.5*	*20444.8*	*17001.1*	*10844.1*

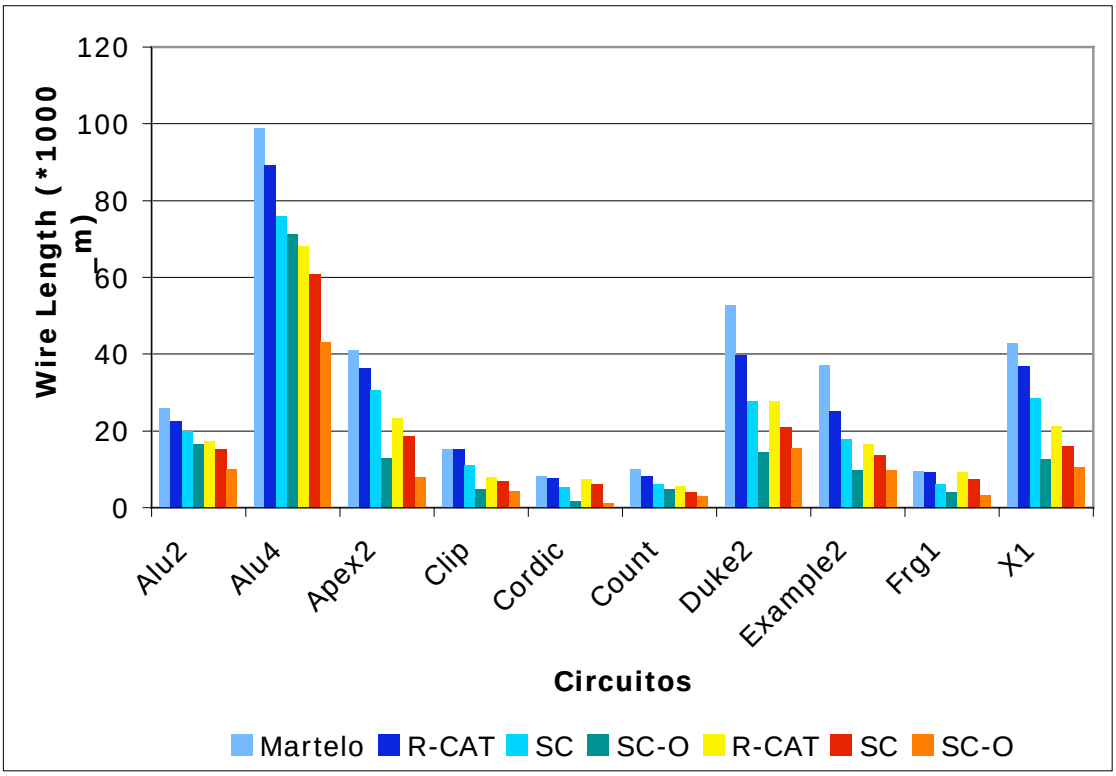


Figura 7.5: Resultados de *wire length* (µm)

É importante observar que, embora as implementações dos circuitos orBDD nas matrizes R-CAT e Martelo apresentem o dobro de nets que os circuitos SIS implementados na matriz R-CAT, o *wire length* das duas primeiras implementações é cerca de 54% maior que o da terceira. Esse fato deve-se à clusterização adotada na síntese orBDD. Com a clusterização, células componentes de um mesmo cluster possuem fan-out 1, atacando apenas uma outra entrada de células provavelmente posicionadas próximas. Essa técnica é responsável também pelo menor fan-out médio, visto anteriormente na Figura 7.1. A característica de clusterização dos circuitos orBDD resulta em conexões mais curtas, aspecto desejável em projetos em tecnologias sub-micrônicas, principalmente para reduzir o atraso devido às interconexões. A Tabela 7.5 apresenta a comparação dos resultados de *wire length* médio dos circuitos. O *wire length* médio é obtido pela divisão dos resultados de *wire length* pelo número de nets em cada configuração de circuito. Essa medida fornece um indicativo do tamanho médio das conexões no circuito. Os resultados da tabela, junto com a ilustração destes na Figura 7.6, mostram que o tamanho médio das interconexões na matriz R-CAT orBDD é 20% menor que os obtidos na matriz R-CAT SIS e 11% menor que os obtidos na matriz Martelo. As versões Standard Cell Otimizadas apresentam poucas interconexões, entretanto, em sua maioria, são compostas por conexões longas. As versões Standard Cell sem otimização apresentam conexões médias mais curtas em relação às suas matrizes equivalentes. Acredita-se que estes resultados sejam devido à característica das ferramentas de posicionamento e roteamento adotas no fluxo de síntese da Cadence. Um ponto interessante a ser notado é o fato da matriz R-CAT orBDD possuir conexões 4% menores que a versão Standard Cell originada na descrição SIS e sem Otimização.

Tabela 7.5: Comparação dos resultados de *wire length* médio obtidos com R-CAT, Martelo, Cadence e Cadence Otimizado (μm)

Circuito	orBDD				SIS		
	Martelo	RCAT	SC	SC-O	R-CAT	SC	SC-O
Alu2	34.43	32.12	28.06	69.75	46.25	40.05	58
Alu4	40.41	38.67	32.75	96.32	63.03	55.97	90.2
Apex2	33.82	29.48	24.86	48.99	45.75	36.06	41.74
Clip	30.88	32.03	23.28	40.98	36.63	31.44	41.4
Cordic	28.4	27.45	19.04	23.61	33.17	26.74	21.37
Count	32.37	30.04	22.08	30.23	29.89	21.44	22.5
Duke2	50.89	39.22	27.11	54.42	53.98	40.23	59.89
Example2	53.82	36.6	25.67	33.24	38.31	30.83	36.22
Frg1	29.27	30.25	19.44	34.24	36.9	28.84	31.19
X1	40.43	37.08	28.33	48.06	31.4	34.78	46.01
Média	*37.472*	*33.294*	*25.062*	*47.984*	*41.531*	*34.638*	*44.852*

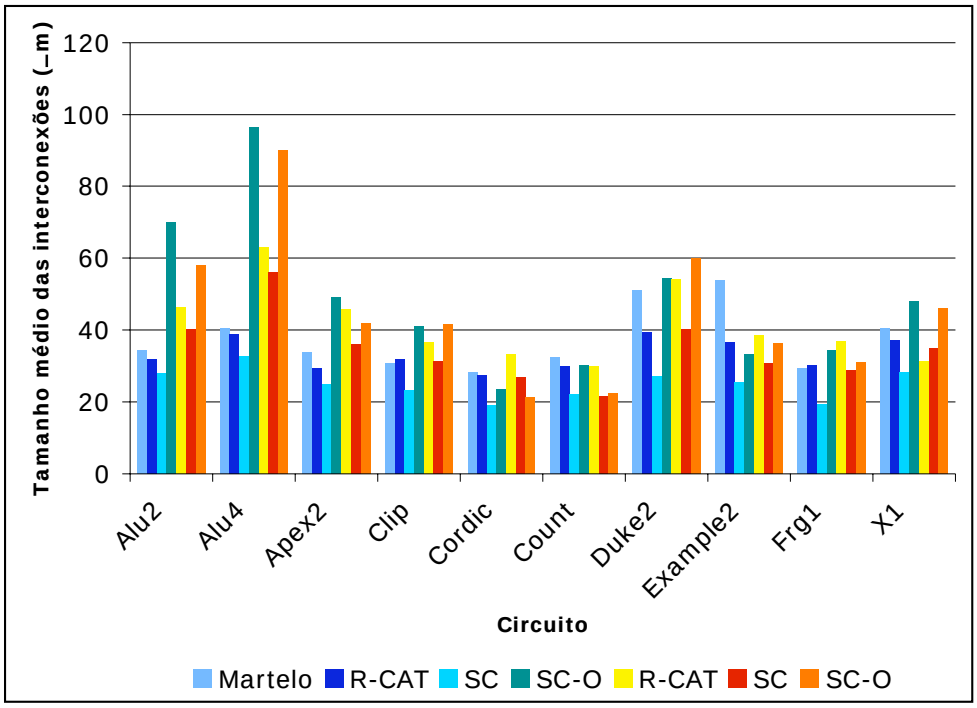


Figura 7.6: Comparação dos resultados obtidos quanto ao tamanho médio das interconexões (μm)

Os resultados de *wire length* nas matrizes R-CAT e Martelo de alguns circuitos são resultados parciais devido ao fato do roteador não haver conseguido realizar o completo roteamento das nets sem a inserção de espaços. A Tabela 7.6 apresenta a taxa de roteablidade dos circuitos nas matrizes R-CAT e Martelo. A maioria dos circuitos foi

100% roteado. Nos circuitos derivados de ORBDDs a taxa de circuitos não roteados é menor que a dos circuitos derivados da síntese SIS. Embora em alguns casos a taxa de não roteabilidade de alguns circuitos atinja quase 10% (no caso do circuito Example2 na opção SIS R-CAT) a maioria dos circuitos não roteados apresentou índice médio de 1 *net* não roteada, o que pode ser considerado um baixo índice respeitando a restrição de não inserir trilhas extras para completar o roteamento. Os resultados são ilustrados na

Figura 7.7.

Tabela 7.6: Taxa de roteabilidade dos circuitos nas matrizes R-CAT e Martelo

Circuito	orBDD		SIS
	Martelo	NINA R-CAT	R-CAT
Alu2	100	100	100
Alu4	100	100	99.54
Apex2	100	99.79	100
Clip	100	100	100
Cordic	99.7	100	100
Count	99.7	98.18	96.81
Duke2	99.9	100	99.8
Example2	100.0	95.66	91.22
Frg1	99.7	98.46	99.2
X1	99.9	99.02	98.67
Média	*99.9*	*99.1*	*98.5*

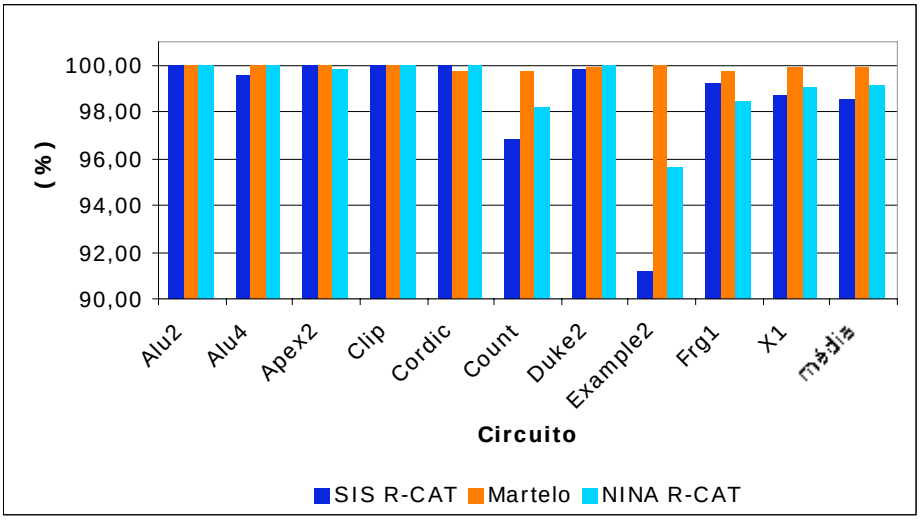


Figura 7.7: Porcentagem de *nets* roteadas após a síntese para as descrições lógicas obtidas com o SIS e com OrBDDs

Os resultados obtidos mostram que as ferramentas de síntese lógica baseadas em OrBDDs resultam em um maior número de portas lógicas, fato que acarreta em uma

maior área. Entretanto, estas sínteses resultam em interconexões com tamanhos mais curtos para a maior parte dos circuitos, sendo em média 20% menor o tamanho médio das interconexões. A comparação entre as matrizes R-CAT SIS e Or BDD mostra que as matrizes geradas usando o SIS ocupam 53% menos área e reduzem o *wire length* em 30%. A menor área é devido ao fato de que o SIS gera descrições com a metade de células lógicas e nets. Entretanto, as matrizes R-CAT orBDD apresentam menor *wire length* médio, menor *fan-out* (redução de 15%), menor delay (redução de 58%) e maior roteabilidade. As sínteses orBDD apresentam poucas nets não roteadas sem a inserção de trilhas extras.

Tabela 7.7: Vantagens e Desvantagens das matrizes R-CAT

Matriz	Vantagens	Desvantagens
SIS+ R-CAT	Regularidade geométrica Menor área Menor *wire length*	Maior fan-out médio Interconexões mais longas
OrBDD+R-CAT	Regularidade geométrica Regularidade lógica Interconexões mais curtas Menor fan-out médio Maior roteabilidade	Maior área Maior número de *nets* Maior *wire length*

Comparando as matrizes R-CAT NINA e as matrizes MARTELO, as matrizes NINA reduzem o *wire length* em 15%, embora aumentem a área necessária em 14%. Ao permitir a utilização de um maior conjunto de células básicas, assim como, a adoção de outras ferramentas de síntese lógica, as matrizes R-CAT atingiram resultados até 40% menores em *wire length* e reduções de área de até 46% em relação as matrizes MARTELO.

Os circuitos Standard Cell, principalmente os otimizados, apresentaram menor área e menor *wire length*. Entretanto, tais circuitos não apresentam o mesmo grau de regularidade explorado nas matrizes. Esses circuitos também apresentam resultados desfavoráveis quanto ao tamanho médio das interconexões e em relação ao fan-out médio gerado pelas otimizações.

8 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma ferramenta para a geração de leiautes regulares. O gerador de matrizes regulares recebeu o nome R-CAT. A regularidade é explorada pela repetição de padrões básicos de leiaute, explorando a regularidade geométrica. A regularidade é apontada como uma das melhores alternativas para lidar com os atuais problemas de fabricação em tecnologias sub-micrônicas. Projetos regulares são menos suscetíveis aos problemas de litografia, aumentam o *yield* e diminuem o tempo de gasto em re-projeto. Além disso, circuitos regulares apresentam maior previsibilidade de resultados de potência, atraso e *yield*, principalmente pelo fato das células estarem pré-caracterizadas.

A ferramenta de geração de leiautes R-CAT está inserida em um fluxo de projeto e depende do método de síntese lógica adotado. Os experimentos realizados demonstram que ao efetuar a síntese lógica usando orBDD reduz-se o tamanho médio das interconexões, diminuindo o *delay* dos circuitos. Embora os circuitos gerados nesta abordagem necessitem maior área e apresentem maior *wire length*, o preço pago pelo aumento da área pode ser amenizado com o aumento do *yield* e pela previsibilidade dos resultados.

A ferramenta desenvolvida visa o trabalho com dois tipos de síntese física para leiautes regulares, produzindo circuitos integrados personalizáveis por todas as máscaras ou circuitos personalizáveis por algumas máscaras. O principal objetivo deste gerador é a facilidade de conversão e adaptação a abordagem de matriz escolhida. Isso facilitará a comparação entre diferentes alternativas de matrizes, a adoção de blocos lógicos diversos e de novas tecnologias.

O gerador de leiautes R-CAT identifica células adjacentes, posicionadas na mesma banda, com conexões em comum entre elas. Quando isso acontece, o gerador pode realizar a conexão entre essas células em metal 1. Esse procedimento remove algumas nets ou reduz trechos de *nets*, reduzindo o número de conexões a ser realizado pelo roteador. Através de experimentos, verificou-se que esta estratégia pode reduzir em 10% o número de conexões a ser realizado.

A avaliação da influência no roteamento da posição interna dos pinos de entrada e saída das células básicas indica que a posição interna dos pinos em células básicas de funções lógicas pequenas (envolvendo até duas variáveis) não influi na capacidade de roteamento total de circuitos em matrizes regulares. A experiência consistiu na exploração de possibilidades de posicionamento para os pinos de entrada e saída das células básicas, em diferentes configurações de área e tipos de matrizes. O algoritmo de roteamento adotado em todas as síntese foi o Rotdl.

A avaliação das sínteses lógicas baseadas em OrBDDs resultam em maior número de portas lógicas, fato que acarreta em maior área. Entretanto, estas sínteses resultam

em interconexões com tamanhos mais curtos para a maior parte dos circuitos, sendo em média 20% menor o tamanho médio das interconexões. A comparação entre as matrizes R-CAT SIS e orBDD mostra que as matrizes geradas usando o SIS ocupam 53% menos área e reduzem o *wire length* em 30%. A menor área é devido à síntese lógica efetuada pelo SIS gerar descrições com a metade de células lógicas e nets. Entretanto, as matrizes R-CAT orBDD apresentam menor *wire length* médio, menor fan-out (redução de 15%), menor delay (redução de 58% na síntese lógica) e maior roteabilidade. As sínteses orBDD apresentam poucas nets não roteadas sem a inserção de trilhas extras.

Comparando as matrizes R-CAT NINA e as matrizes MARTELO, as matrizes NINA reduzem o *wire length* em 15%, embora aumentem a área necessária em 14%. Ao permitir a utilização de um maior conjunto de células básicas, assim como, a adoção de outras sínteses lógicas, as matrizes R-CAT atingiram resultados até 40% menores em *wire length* e reduções de área de até 46% em relação às matrizes MARTELO.

Os circuitos Standard Cell, principalmente os otimizados, apresentaram menor área e menor *wire length*. Entretanto, tais circuitos não apresentam o mesmo grau de regularidade explorado nas matrizes. Esses circuitos também apresentam resultados desfavoráveis quanto ao tamanho médio das interconexões e em relação ao fan-out médio gerado pelas otimizações. Os circuitos R-CAT ganham em regularidade, apresentando melhor DFM, ou seja, apresentam menor probabilidade de falhar, pois, após uma fabricação, os resultados de printabilidade das matrizes serão previsíveis.

A utilização da ferramenta R-CAT com as sínteses lógicas SIS e orBDD oferece duas linhas de projeto: a primeira priorizando à área e a segunda priorizando timing e interconexões curtas. Ambas respeitando a mesma regularidade geométrica imposta pela matriz.

O atual fluxo de projeto não garante a convergência do sistema, ou seja, não fornece garantias de finalizar o circuito totalmente roteado. Uma solução para tornar o sistema convergente com o roteador Rotdl, é a implementação no gerador de leiautes de um rotina iterativa com o roteador. Essa rotina seria responsável por inserir trilhas verticais ou células *dummies* nos pontos de congestionamento do roteamento.

Uma alternativa para rápida prototipação envolve a utilização do roteador ChAOS (SANTOS, 2006). Este roteador trabalha sobre um modelo de pinos alinhados, fazendo uso de rotinas de baixa complexidade e garantindo a convergência com pequena penalidade de área. O roteador ChAOS pode ser utilizado quando as células adotarem um leiaute com todos os pinos alinhados na mesma altura. O ChAOS garante a convergência do sistema, ou seja, garante o roteamento total do circuito, com a inserção de espaço. Os espaços podem ser células dummies ou extra *tracks*. A posição destes espaços é fornecida e a matriz pode ser reconstruída para atender às novas condições de roteamento. Uma outra opção é alterar o fluxo de síntese, de modo que, em um momento inicial, o gerador de matrizes forneça somente a grade de roteamento. O roteador realiza o roteamento e indica os espaços necessários. Então, com base nessas novas condições é gerada a matriz. Além disso, a ferramenta permite a fácil adaptação às tecnologias com diversas camadas de metal para roteamento, o que aumentará a convergência do fluxo de geração de circuitos.

A ferramenta de geração de matrizes pode ser utilizada para geração de matrizes no estilo MARTELO, MARCELA e MARAGATA, sendo necessário, para isso, a geração das células utilizadas nestes projetos. A ferramenta possibilitará o estudo de matrizes com células programáveis e de outros blocos básicos com maior granularidade.

Com o objetivo de gerar circuitos fabricáveis, ainda são necessárias ao fluxo de síntese à criação de ferramentas capazes de realizar o anel de *pads* do circuito e o roteamento entre os *pads* e o *core* do circuito. Tem-se como plano futuro, a avaliação da possibilidade de fabricação de um Structured ASIC no CEITEC, assim como um Structured ASIC a ser processados em tecnologia de 90 nm ou inferior.

REFERÊNCIAS

ABERCROMBIE, D.; FERGUNSON, J. **Design for Manufacturing:** What Designers Need to Know About the Changes in Yield Management. 2005. Disponível em: <www.techonline.com/community/ed_resource/tech_paper/37674>. Acesso em: fev. 2005

AGARWAL, R.; GUPTA, I. S. Design for Manufacturing: On the Synthesis of Gate Matrix Layout. In: INTERNATIONAL CONFERENCE ON VLSI DESIGN, 7., 1994. **Proceedings…** [S.l.: s.n.], 1994. p.203-206.

ALTERA. **Altera – HardCopy II Architecture.** Disponível em: <www.altera.com/products/devices/hardcopyii/features/architecture/hr2-architecture.html> Acesso em: jun. 2005.

ALTERA. **Altera – Stratix Devices:** New Levels of System Integration. Disponível em: <www.altera.com/products/devices/stratix/stx-index.jsp>. Acesso em: out. 2005.

AMI. **AMI Semiconductor. Gate Array**. Disponível em: <www.amis.com/asics/gate_array.html>. Acesso em: dez. 2005.

AMI. **AMI Semiconductor.** XPressArray. Disponível em: <www.amis.com/asics/structured_asics/XPressArray.html>. Acessado em: jun. 2005 b.

ATMEL. Disponível em: <www.atmel.com>. Acesso em: dez. 2005.

BOLSENS, I. et al. Fast, Cheap and Under Control: The Next Implementation Fabric**.** In: DESIGN AUTOMATION CONFERENCE, DAC , 40., 2003. **Proceedings…** New York: ACM, 2003. p. 355-356.

BORKAR, S. et al. Parameter Variations and Impact on Circuits and Microarchitecture. In: DESIGN AUTOMATION CONFERENCE, DAC, 40., 2003. **Proceedings…** New York: ACM, 2003. p. 338-342.

BURSKY, D. Design Systems with Platform ASICs. Basics of Design – Platform ASIC. **Electronic Design Online**, n. 7599, Mar. 2004. Disponível em: <www.elecdesign.com/Articles/ArticleID/7599/7599.html>. Acesso em: ago. 2004.

CALDWELL, A.; KAHNG, A.; MARKOV, I. Can Recursive Bisection Alone Produce Routable Placements?. In: CONFERENCE ON DESIGN AUTOMATION, DAC, 37., 2000. **Proceedings…** New York: ACM, 2000. p.477-482.

CARRO, L. **Desenvolvimento e caracterização de uma biblioteca de standard cells.** Porto Alegre: PGCC da UFRGS, 1987. 29 p. (RP-81).

CARRO, L. et al. An Environment to Design Digital Circuits Based on the Brazilian Gate- Array. In: WORKSHOP IBERCHIP, 2., 1996, São Paulo. **Anais…**São Paulo: LSI/USP, 1996. p. 198-205.

CARRO, L. et al. Ambiente ÁGATA de Projeto, Versão Beta 2.0. In: WORKSHOP IBERCHIP, 3., 1997. **Anais …** México: Departamento de Ingineria Electrica, 1997. p. 494-503.

CHANG, K.; MARKOV, L.; BERTACCO, V. Post-Placement Rewiring and Rebuffering by Exhaustive Search for Functional Symmetries. In: IEEE/ ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2005. **Digest of Technical Papers.** New York: ACM, 2005. p. 56-63.

CHAWLA, R. **FPGAs and Strucutred ASICs**: New Solutions for Changing Market Dynamics. Disponível em:

<http://www.chipdesignmag.com/display.php?articleId=255>. Acesso em: jan. 2006.

CHRZANOWSKA-JESKE, M.; MISHCHENKO, A. Synthesis for Regularity using Decisions Diagrams. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005. **Proceedings…** [S.l.]: IEEE, 2005. v. 5, p. 4721-4724.

CONG, J. An Interconnect-Centric Design Flow for Nanometer Technologies. **Proceedings of the IEEE,** Los Alamitos, v.89, n. 4, p.505-528, Apr. 2001.

DALLY, W. J.; CHANG, A. The Role of Custom Design in ASIC Chips. In: CONFERENCE ON DESIGN AUTOMATION, DAC, 37., 2000. **Proceedings…** New York: ACM, 2000. p.643-647.

DE MICHELI, G. **Synthesis and Optimization of Digital Circuits**. New York: McGraw–Hill, 1994.

DIPERT, B. Silicon Segmentation: ASIC Spin-offs Seek Success Amid Established Alternatives. **EDN**, [S.l.], p. 57 – 66, Sept. 2003.. Disponível em: <www.edn.com/article/CA321801.html?text=silicon+segmentation>. Acesso em: jun. 2004.

eASIC. eASIC Technology Overview. eASIC – The configurable logic company. Disponível em: <www.easic.com/index.php?p=technology>. Acesso em: jun.2005.

eASIC eASIC Fabric. eASIC – The configurable logic company. Disponível em: <www.easic.com/index.php?p=easic_fabric>. Acesso em: jun.2005b.

FARADAY. **Structured ASIC**. 2004. Disponível em:

<http://www.faraday-tech.com/html/products/structuredASIC.html>. Acesso em: jun.

2005.

FERGUSON, J. **The Necessary Link for Design Closure**: LVS-Parasitic Extraction. [S.l.]: Mentor Graphics, Apr. 2004. Disponível em: <www.techonline.com/community/ed_resource/tech_paper/36294>. Acesso em: abr. 2005.

FINCO, S. et al. A New Concept for Cost Effective Smart Power Ics Based on a Unique Cell Type. In: IEEE INDUSTRY APPLICATIONS CONFERENCE; IAS ANNUAL MEETING, 33., 1998. **Proceedings…** [S.l.]: IEEE, 1998. v. 2, p. 1111 – 1118.

FLACH, G.; HENTSCHKE, R.; REIS, R. Improving Maze Routers Routability by a New Rip-up and Reroute Approach. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 2005, Santa Cruz do Sul. **Proceedings…** Santa Cruz do Sul: UNISC, 2005. p.83-86.

FUJITSU. Documentation: FUJITSU United States. AccelArray. Disponível em: <www.fujitsu.com/us/services/edevices/microelectronics/accelarray/docaccelarray> Acesso em: dez. 2005.

GU, J.; SMITH, K. F. A Structured Approach for VLSI Circuit Design**. Computer**, New York, v. 22, n.11, p. 9 – 22, Nov. 1989.

GÜNTZEL, J. L. A. **Geração de Circuitos Utilizando Matrizes de Células Pré-Difundidas.** 1993. 174 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

GÜNTZEL, J. L. A. **Sistemas Digitales – Síntese física de circuitos integrados**. Bogotá: CYTED - Ediciones Uniandes, 2000.

GUPTA, P.; KAHNG, A. B. Manufacturing-Aware Physical Design In: IEEE/ ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2003. **Digest of Technical Papers.** New York:ACM, 2003**.** p. 681-687.

HASHIMOTO, A.; STEVENS, S. Wire Routing By Optimizing Channel Assignment whitin Large Apertures. In: DESIGN AUTOMATION WORKSHOP, DAC, 8., Atlantic City. **Proceedings…** New York: ACM, 1971. p 155-169.

HENTSCHKE, R. F.; TAVARES, R.; REIS, R. A Cell placer for Matrix Regular Layouts. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, 19., 2004. **Proceedings…** Ijui: UNIJUI, 2004. p.59-64.

HINTERMAIER, M. et al. A Process Latitude for Direct Write eBeam Lithography. **Microelectronic Engineering**, [S. l.], v. 13, n. 1-4, p. 105-108, Mar. 1991.

JAEGER, R. C. **Introduction to microelectronic fabrication.** [S.l.]: Addison-Wesley, 1993. (Modular series on solid state devices, v. 5)

JAYAKUMAR, N.; KHATRI, S. P. A METAL and VIA Maskset Programmable VLSI Design Methodology using PLAs. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2004, San Jose, California. **Digest of Techical Paper.** New York: ACM, 2004. p. 590- 594.

JOHANN, M. O.; REIS, R. A Full Over-the-Cell Routing Model. In: IFIP INTERNATIONAL SYMPOSIUM ON VERY LARGE SCALE INTEGRATION, 1995, Chiba, Japan. **Proceedings…** [S.l.: s.n.], 1995. p. 845-850.

JOHANN, M. O.; CARRO, L.; REIS, R. GAROTA: Gate Array Router of ÁGATA System. In: UFRGS MICROELECTRONICS SEMINAR, 11., 1996, Porto Alegre. **Proceedings…** Porto Alegre: CPGCC da UFRGS, 1996. p.97-100.

JOHANN, M.; CARRO, L.; REIS, R. A. L. Automatic MasterSlice Generation with GAROTA. In: WORKSHOP IBERCHIP, 2000, São Paulo. **Proceedings...** Campinas: CTI, 2000. p. 355-360.

JOHANN, M. O. **Novos algoritmos para roteamento de circuitos VLSI**. 2001. 158 f. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

KARNIK, T.; BORKAR, S.; DE, V. Sub 90-nm Technologies-Challenges and Opportunities for CAD. In: IEEE/ ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2002. **Proceedings…** New York: IEEE, 2002**.** p.203-206.

KHANG, A. et al. What is the Next Implementation Fabric. **IEEE Design & Test of Computers,** New York, v. 20, n. 6, p. 86-95, Nov. 2003.

KHATRI, S.; BRAYTON, R.; SANGIOVANNI-VINCENTELLI, A. **A VLSI Design Methodology using a Network of PLAs Embedded in a Regular Leiaute Fabric**. Berkeley: Electronics Research Laboratory, University of California, 1999. (Technical Report – UCB/ERL M99/50)

KHETERPAL, V.; STROJWAS, A. J.; PILEGGI, L. Routing Architecture Exploration for Regular Fabrics. In: Design Automation Conference, DAC, 2004. **Proceedings…** [S.l.:s.n.], 2004. p. 204-207.

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

KOORAPATY, A. et al. Exploring Logic Block Granularity for Regular Fabrics. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, DATE, 2004. **Proceedings…** Los Alamitos: IEEE Computer Society, 2004**.** p. 468-473.

LIAO, Y.; CHOW, S. Routing Considerations in Symbolic Leiaute Synthesis**.** In: ACM/ IEEE DESIGN AUTOMATION CONFERENCE, DAC, 29., 1992. **Proceedings…** [S. l.]: IEEE, 1992. p. 682-686.

LIOYD, M. **Standard-Metal:** The Ultimate Structured ASIC Fabric. SOC central. Disponível em: <www.soccentral.com/results.asp?CategoryID=488&EntryID=15888>. Acesso em: jun. 2005.

LIMA, F. G. **Projeto com Matrizes de Células Lógicas Programáveis.** 1999. 126 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

LIPMAN, J. The "Missing Link" of SoC Design- Platform and Structured ASICs.

Techonline, Jun. 2004. Disponível em:

<www.techonline.com/community/member_company/non_member/feature_article/312 > Acesso em: set. 2004.

LOPEZ, A. D.; LAW, H. S. A. Dense Gate Matrix Leiaute Method for MOS VLSI. **IEEE Transations on Electron Devices**, New York, v.27, n. 8, p. 1671-1675, Aug. 1980.

LSI LOGIC. **RapidChip Tecnhology Fast Custom Silicon Trough Platform-Based Design**. [S.l.], 2004. Disponível em: <www.lsilogic.com> Acesso em: set. 2004.

MACMILLEN, D. et al. W. An Industrial View of Electronic Design Automation. **IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems**, New York, v.19, n.12. Dec. 2000.

MANO, M. **Digital Logic and Computer Design.** London: Prentice-Hall International, 1984. 492p.

MEINHARDT, C.; TAVARES, R.; REIS, R**.** A Matrix Cell Viewer**.** In: WORKSHOP IBERCHIP, 11., 2005, Salvador. **Anais…** [S.l.:s.n.], 2005. p.293-294.

MENEZES, C. C. **Geração Automática de Leiaute através de uma Matriz de Células NAND – MARTELO.** 2004. 53 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

MO, F.; BRAYTON, R. K. River PLAs: A Regular Circuit Structure. In: DESIGN AUTOMATION CONFERENCE, DAC, 39., 2002. **Proceedings…** [S.l.:s.n.], 2002. p. 201-206.

MO, F.; BRAYTON, R. K. Whirlpool PLAs: A Regular Logic Structure and Their Synthesis**.** In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2002 **Proceedings…** New York: ACM SIGDA, 2002**.** p. 543-550.

MO, F.; BRAYTON, R. K. PLA-Based Regular Structures and their Synthesis. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v. 22, n. 6, p. 723 – 729, June 2003.

MORAES, F. G. **Anatomia de uma Ferramenta de Síntese Automática de Leiaute - Estilos de layout**. Palestra: EMICRO 2000. Disponível em: <www.inf.pucrs.br/~moraes/papers/emicro_sld.pdf> Acesso em: dez. 2005.

NEC Electronics/ Gate Array and Embedded Array ASIC/ Documentation. Disponível em: <www.necel.com/gatearray/english/document/doclist.html> Acesso em: set. 2005.

OR BACH, Z. **Paradigm Shift in ASIC Technology In- Standard Metal Out – Standard Cell. eASIC.** Disponível em: <www.easic.com/pdf/WP_0001.pdf>. Acesso em: fev. 2006.

OTTEN, R. H. J. M.; CAMPOSANO, R.; GROENEVELD, P. R. Design Automation for Deep submicron: present and future. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, DATE, 2002, Paris. **Designers' Forum.** Los Alamitos: IEEE Computer Society, 2002. p. 650 -657.

PATEL, C. et al. An Architectural Exploration of Via Patterned Gate Arrays. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2003. **Proceedings…** New York: ACM, 2003**.** p. 184-189.

PILEGGI, L. et al. Exploring Regular Fabrics to Optimize the Performance-Cost Trade-Off. In: DESIGN AUTOMATION CONFERENCE, DAC, 2003. **Proceedings…** New York: ACM, 2003. p.782 – 787.

PREAS, B.; LORENZETTI, M. **Physical Design Automation of VLSI Systems.** Menlo Park: Benjamin/ Cummings, 1988. 510 p.

RABAEY, J. M. **Digital integrated circuits:** a design perspective**.** 2nd ed. Upper Saddle River: Prentice Hall, 2003. 761 p.

RAN Y.; MAREK-SADOWSKA, M. An Integrated Design Flow for a Via-Configurable Gate Array. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2004. **Digest of Technical Papers.** New York: ACM, 2004. p. 582-589.

RAN, Y.; MAREK-SADOWSKA, M. On Designing Via-configurable Cell Blocks for Regular Fabrics. In: DESIGN AUTOMATION CONFERENCE, DAC, 41., 2004. **Proceedings…** New York: ACM, 2004. p.198-203.

RAN Y.; MAREK-SADOWSKA, M. Via-Configurable Routing Architectures and Fast Design Mappability Estimation for Regular Fabrics. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2005. **Proceedings…** New York: ACM, 2005. p. 25-32.

REINHARDT, M. **Automatic Layout Modification including design reuse of the Alpha CPU in 0.13 micron SOI technology**. Boston: Kluwer Academic, 2002. p. 15-50.

REINHARDT, M. **Automatic Leiaute Modification including design reuse of the Alpha CPU in 0.13 micron SOI technology**. Boston: Kluwer Academic, 2002a. p. 51-76.

REIS, A, I.; GÜNTZEL, J. L.; RIBAS, R. Algumas Formas de Implementação de ASICs. In: SIMPÓSIO BRASILEIRO DE CONCEPÇÃO DE CIRCUITOS INTEGRADOS, SBCCI, 7., 1992. **Anais...** Rio de Janeiro: SBC, 1992. p.15-34.

REIS, R. **TESS Evaluater Automatique des Plans de Masse de Circuits VLSI**. 1983. Tese – (Doutorado) Institute National Polytechnique, Grenoble, França.

REIS, R. A new Standard Cell CAD Methodology. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, 1988, Portland, Oregon. **Proceedings…** New York: IEEE, 1987. p. 385-388.

REIS, R.; GOMES, R.; LUBASZEWSKI, M. S. An Efficient Design Methodology for Standard Cell Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS. **Proceedings…** New York: CIRCUITS AND SYSTEMS, 1988. p. 1213-1216.

REIS, R.; MORAES, F. **Sistemas Digitale**s: Metodologías de diseño VLSI. Bogota: CYTED, 2000. p. 67-100.

REIS, R.; KASTENSMIDT, F. L.; GÜNTZEL, J. L. Physical Design Methodologies for Perfomance Predictability and Manufacturability. In: COMPUTING FRONTIERS, 1., 2004. **Proceedings…**New York: ACM, 2004. p. 390-397.

ROBERTSON, C. **Challenges of Silicon Modeling in Nanometer Designs**. [S.l.]: Mentor Graphics Corp. Technical Publication, 2003. Disponível em: <www.mentor.com/techpapers/abstracts/mentorpaper_18773.cfm> Acesso em: abr. 2004.

RUBIN, S. M. **Computer Aids for VLSI Design**. Appendix B: Caltech Intermediate Format. Disponível em: <www.rulabinsky.com/cavd/text/chapb.html>. Acesso em: abr. 2005.

SANTOS, G. B. V.; JOHANN, M.; REIS, R. Channel Based Routing in Channel-less Circuits. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2006, Ilha de Kós, Grécia. **Proceedings…**Los Alamitos: IEEE, 2006.

SAWICKI, J. Achieving Better DFM: EDA Tools Pave the Way to Improved Yield. **EDA Tech Fórum**, [S.l.], v. 2, n. 2, p.28-32, June 2005.

SHENOY, N. V.; KAWA, J.; CAMPOSANO, R. Design Automation for Mask Programmable Fabrics. In: DESIGN AUTOMATION CONFERENCE, DAC, 41., 2004. **Proceedings…** New York: ACM, 2004. p.192-197.

SHERLEKAR, D. D. Design Considerations for Regular Fabrics. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2004. **Proceedings…** New York: ACM, 2004. p. 97-102.

SIMÕES, S. A. et al. Matriz Gate Array CMOS Avançada, Configurável por um Único Nível de Metal. In: SBMICRO, 7., 1992. São Paulo, SP. **Anais…** São Paulo : SBMICRO, 1992. p. 281-291.

SYNOPSYS. Products – Distributed Processing. Disponível em: <www.synopsys.com/products/ntimrg/dist_proc_ds.html> Acesso em: nov.2005

TAVARES, R.; BERKELAAR, M. Logic Circuits Based on or Binary Decision Diagrams. In: INTERNATIONAL WORKSHOP ON LOGIC SYNTHESIS, IWLS, 2000. **Proceedings…** [S.l.:s.n.], 2000. p. 91-95.

TAVARES, R.; MEINHARDT, C.; REIS, R. Logic Synthesis for a Configurable Cell Matrix. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 20., 2005. **Proceedings…** Santa Cruz do Sul: UNISC, 2005. p.83-86.

TAVARES, R.; MEINHARDT, C.; REIS, R. **orBDDs Direct Mapping for Structured Logic Circuits.** Trabalho submetido e aprovado para a 13th IEEE International Conference on Electronics, Circuits and Systems, 2006, Nice. Ainda não publicado.

TONG, K. Y. et al. Regular Logic Fabrics for a Via Patterned Gate Array (VPGA). In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, 2003. **Proceedings…** New York: IEEE, 2003. p. 53-56.

UEHARA, T.; VANCLEEMPUT, W. Optimal Layout of CMOS Functional Arrays. **IEEE Transactions on Computers**, New York, v. C-30, n.5, p.195-202, 1981.

ViASIC. Structured ASICs and Configurable SOCs using ViASIC Standard Metal Solutions. Via Mask Structured ASIC. 2004. Disponível em: <www.viasic.com/Solutions/Structured_ASICs/ViaMask_Datasheet_2004.2.04.pdf> Acesso em: dez. 2004.

WEINBERGER, A. Large Scale Integration of MOS Complex Logic. **IEEE Solid State Circuits**, New York: v. SC-2, p. 182-190, Dec. 1967.

WESTE, N. H. E.; KAMRAN, E. **Principles of CMOS VLSI Design.** 2nd ed. Reading, Massachusetts : Addison - Wesley, 1992. 713 p.

WU, K.; TSAI, Y. Structured ASIC, Evolution or Revolution? In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2004. **Proceedings**… New York: ACM, 2004. p. 103-106

YU-WEN, T.; CHUANG, S. **The platform based SOC design that utilizes Structured ASIC Technology.** Disponível em: <www.us.design-reuse.com/articles/article9566.html>. Acesso em: jun. 2005.

XILINX. VIRTEX II PLATAFORM FPGA S. Disponível em: <www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_platform_fpgas/index.htm>. Acesso em: dez. 2005.

XILINX. VIRTEX-4 MULTI-PLATAFORM FPGA S. Disponível em: <www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/index.htm> e <www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_platform_fpgas/index.htm>. Acesso em: jan. 2006.

ZAHIRI, B. Structured ASICs: Opportunities and Challenges. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN, ICCD, 21., 2003. **Proceedings…** Los Alamitos: IEEE Computer Society, 2003. p. 404- 409.

Anexo Tabelas de Dados

Tabela A.8.1: Fan- out médio das portas lógicas gerados pelas sínteses lógicas analisadas.

Circuito	SIS	Nand	NINA	Cadence
Alu2	1,84	1,53	1,57	2,6
Alu4	1,84	1,51	1,56	2,7
Apex2	1,71	1,41	1,43	2,28
Clip	1,79	1,51	1,56	2,33
Cordic	1,7	1,43	1,48	1,73
Count	1,58	1,36	1,51	2,1
Duke2	1,8	1,5	1,53	2,47
Example2	1,59	1,52	1,45	1,8
Frg1	1,66	1,31	1,37	1,94
X1	1,68	1,44	1,46	2,12
Média	*1,719*	*1,452*	*1,492*	*2,207*

Observação: Estes valores foram utilizados na geração do gráfico apresentado na Figura 7.1.

Tabela A.8.2: Capacitância média (pF) observada nas sínteses lógicas avaliadas

Circuito	SIS	Nand	NINA	Cadence
Alu2	0,04	0,03	0,03	0,06
Alu4	0,03	0,02	0,02	0,05
Apex2	0,03	0,02	0,02	0,04
Clip	0,05	0,03	0,03	0,08
Cordic	0,03	0,02	0,03	0,06
Count	0,11	0,06	0,08	0,13
Duke2	0,08	0,05	0,05	0,15
Example2	0,17	0,11	0,12	0,26
Frg1	0,03	0,03	0,03	0,06
X1	0,1	0,05	0,05	0,17
Média	*0,067*	*0,042*	*0,046*	*0,106*

Observação: . Estes resultados foram utilizados na geração do gráfico apresentado na Figura 7.2.

Tabela A.8.3: Resultados da análise de timing (ns) realizada para as sínteses lógicas..

Circuito	SIS	Nand	NINA	Cadence
Alu2	46,33	27,96	27,88	41,92
Alu4	67,86	40,2	40,32	29,25
Apex2	20,5	28,57	27,68	20,15
Clip	60,42	23,27	23,37	18,58
Cordic	27,93	23,18	22,89	16,89
Count	23,48	27,45	27,05	30,06
Duke2	59,95	27,23	27	29,95
Example2	64,09	24,25	24,89	26,89
Frg1	20,83	22,7	22,74	17,63
X1	45,1	29,64	30,74	28,5
Média	*43,64*	*27,44*	*27,45*	*25,98*

Observação: Estes valores foram utilizados na geração do gráfico apresentado na Figura 7.3

Apêndice Descrições CIFs

Descrição CIF do bloco básico:

```
(CIF file written on 23-Mar-2005  18:41:26 by CADENCE);
(Modified on 4-April-2005 by Cristina Meinhardt);

DS 1 1 1;
9 metal2;
DF;
DS 2 1 1;
9 metal1;
DF;
DS 3 1 1;
9 M2_M1;
L CVA;
B 40 40 0 0;
L CMS;
B 80 80 0 0;
L CMF;
B 80 80 0 0;
DF;
DS 4 1 1;
9 M1_POLY;
L CCC;
B 40 40 0 0;
L CMF;
B 80 80 0 0;
L CPG;
B 80 80 0 0;
DF;
DS 5 1 1;
9 M1_N;
L CCC;
B 40 40 0 0;
L CSN;
B 160 160 0 0;
L CMF;
B 80 80 0 0;
L CAA;
B 80 80 0 0;
DF;
DS 6 1 1;
9 M1_P;
L CCC;
B 40 40 0 50;
B 40 40 0 -50;
L CSP;
B 160 260 0 0;
```

```
L CMF;
B 80 180 0 0;
L CAA;
B 80 180 0 0;
DF;

DS 8 1 1;
9 block;
L CMF;
B  640 60 -1260 490;
B  640 60 -1260 -420;
B 60 60 -1090 430;
B 60 60 -1410 -360;
L CPG;
B 40 810 -1330 35;
B 40 810 -1170 35;
L CSP;
B 480 280 -1250 320;
L CSN;
B 510 200 -1245 -290;
L CAA;
B 400 200 -1250 320;
L CAA;
B 400 100 -1250 -290;
L CWN;
B 630 470 -1255 305;
C 6 T -1090 310;
C 6 T -1250 310;
C 6 T -1410 310;
C 5 T -1090 -290;
C 5 T -1250 -290;
C 5 T -1410 -290;
C 4 T -1120 120;
C 4 T -1380 -150;
C 3 T -1250 -20;
C 3 T -1120 120;
C 3 T -1380 -150;
C 2 T -1560 -420;
C 2 T -1570 490;
C 1 T -1120 120;
C 1 T -1380 -150;
C 1 T -1250 -20;
DF;
C 8;
E
```

Descrição CIF da célula NAND de 2 entradas

```
(CIF file written on 23-Mar-2005  18:41:26 by CADENCE);
(Modified on 4-April-2005 by Cristina Meinhardt);

DS 1 1 1;
9 metal2;
DF;
DS 2 1 1;
9 metal1;
DF;
DS 3 1 1;
9 M2_M1;
L CVA;
B 40 40 0 0;
L CMS;
B 80 80 0 0;
L CMF;
B 80 80 0 0;
DF;
DS 4 1 1;
9 M1_POLY;
L CCC;
B 40 40 0 0;
L CMF;
B 80 80 0 0;
L CPG;
B 80 80 0 0;
DF;
DS 5 1 1;
9 M1_N;
L CCC;
B 40 40 0 0;
L CSN;
B 160 160 0 0;
L CMF;
B 80 80 0 0;
L CAA;
B 80 80 0 0;
DF;
DS 6 1 1;
9 M1_P;
L CCC;
B 40 40 0 50;
B 40 40 0 -50;
L CSP;
B 160 260 0 0;
L CMF;
B 80 180 0 0;
```

```
L CAA;
B 80 180 0 0;
DF;

DS 7 1 1;
9 nand2;
L CMF;
B 640 60 -1260 490;

B  640 60 -1260 -420;
B 60 60 -1090 430;
B 60 60 -1410 430;
B 60 60 -1410 -360;
B 60 220 -1250 110;
B 160 60 -1140 -20;
B 60 200 -1090 -150;
L CPG;
B 40 810 -1330 35;
B 40 810 -1170 35;
L CSP;
B 480 280 -1250 320;
L CSN;
B 510 200 -1245 -290;
L CAA;
B 400 200 -1250 320;
L CAA;
B 400 100 -1250 -290;
L CWN;
B 630 470 -1255 305;
C 6 T -1090 310;
C 6 T -1250 310;
C 6 T -1410 310;
C 5 T -1090 -290;
C 5 T -1250 -290;
C 5 T -1410 -290;
C 4 T -1120 120;
C 4 T -1380 -150;
C 3 T -1250 -20;
C 3 T -1120 120;
C 3 T -1380 -150;
C 2 T -1560 -420;
C 2 T -1570 490;
C 1 T -1120 120;
C 1 T -1380 -150;
C 1 T -1250 -20;
DF;
C 7;
E
```

Descrição CIF da célula NOR de 2 entradas:

```
(CIF file written on 23-Mar-2005  18:41:26 by CADENCE);
(modified on 4-April-2005 by Cristina Meinhardt);

DS 1 1 1;
9 metal2;
DF;
DS 2 1 1;
9 metal1;
DF;
DS 3 1 1;
9 M2_M1;
L CVA;
B 40 40 0 0;
L CMS;
B 80 80 0 0;
L CMF;
B 80 80 0 0;
DF;
DS 4 1 1;
9 M1_POLY;
L CCC;
B 40 40 0 0;
L CMF;
B 80 80 0 0;
L CPG;
B 80 80 0 0;
DF;
DS 5 1 1;
9 M1_N;
L CCC;
B 40 40 0 0;
L CSN;
B 160 160 0 0;
L CMF;
B 80 80 0 0;
L CAA;
B 80 80 0 0;
DF;
DS 6 1 1;
9 M1_P;
L CCC;
B 40 40 0 50;
B 40 40 0 -50;
L CSP;
B 160 260 0 0;
L CMF;
B 80 180 0 0;
L CAA;
```

```
B 80 180 0 0;
DF;

DS 12 1 1;
9 nor2;
L CMF;
B  640 60 -1260 490;
B  640 60 -1260 -420;
B 60 60 -1090 430;
B 60 60 -1090 -360;
B 60 60 -1410 -360;
B 60 220 -1250 -150;
B 160 60 -1360 -20;
B 60 200 -1410 110;
L CPG;
B 40 810 -1330 35;
B 40 810 -1170 35;
L CSP;
B 480 280 -1250 320;
L CSN;
B 510 200 -1245 -290;
L CAA;
B 400 200 -1250 320;
L CAA;
B 400 100 -1250 -290;
L CWN;
B 630 470 -1255 305;
C 6 T -1090 310;
C 6 T -1250 310;
C 6 T -1410 310;
C 5 T -1090 -290;
C 5 T -1250 -290;
C 5 T -1410 -290;
C 4 T -1120 120;
C 4 T -1380 -150;
C 3 T -1250 -20;
C 3 T -1120 120;
C 3 T -1380 -150;
C 2 T -1560 -420;
C 2 T -1570 490;
C 1 T -1120 120;
C 1 T -1380 -150;
C 1 T -1250 -20;
DF;
C 12;
E
```