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

FELIPE DE ANDRADE PINTO

Posicionamento Visando Redução do Comprimento das Conexões

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

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, Agosto de 2011.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pinto, Felipe de Andrade.

Posicionamento Visando Redução do Comprimento das Conexões / Felipe de Andrade Pinto. – Porto Alegre: PGMICRO da UFRGS, 2011.

93 f.: II.

Dissertação de Mestrado – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em microeletrônica. Porto Alegre, BR – RS, 2011.

1. Posicionamento. 2. Desempenho 3. Ferramentas de CAD 4. Síntese Física 5. Microeletrônica. I. Reis, Ricardo. II. Johann, Marcelo. III. Posicionamento Visando Redução do Comprimento das Conexões

O pessimista vê dificuldade em cada oportunidade;
O otimista vê oportunidade em cada dificuldade.
"Winston Churchill"

AGRADECIMENTO

Primeiramente agradeço a Deus por iluminar o meu a caminho e me provar a cada dia que esta do meu lado.

Gostaria de agradecer aos meus amigos pelo companheirismo em vários momentos e que sem eles eu nada seria. Todos que estiveram perto de mim, me apoiando tanto na parte profissional quanto no pessoal. Pois sem eles com certeza não chegaria até aqui. Em especial gostaria de agradecer ao Bruno pela amizade de sempre. Gostaria de agradecer ao Dr. Gustavo Neuberger e pela amizade sempre demonstrada.

Outras pessoas que não podem ser esquecidas são todos os que me ajudaram na no meu estudo científico, apesar de alguns já terem se tornado meus amigos também. Primeiramente ao professor Dr. Ricardo Reis que me deu uma oportunidade de mostrar o meu valor mesmo eu nem sabendo C++. Ao Dr. Renato Hentschke, um dos melhores com quem já trabalhei e quem me ensinou muito do que eu sei. Ao professor Dr. Marcelo Johann pela amizade e pelos ensinamentos. Aos doutorandos Sandro Sawick, Gustavo Wilke, Glauco Borges, Cristiano Lazzari, Julio Balzano, Adriel Ziesemer, Digeórgia Silva, Lucas Brusamarello e Luciano Agostini pelo apoio sempre didático e paciente. Aos mestrandos do laboratório William Lauthenschager, Rafael Zago e Cristina Meinhart.

Agradecimento também muito merecido é do pessoal do TH, Leonardo Seiji Kuamoto, Sergio Mergen, Felipe Madruga, Marcio Trabuco, Thiago Ló e o Diego Pozzi.

Também gostaria de agradecer em especial ao Mestre Guilherme Flach, o Gui, pela amizade desde a infância, em todos os momentos de crescimento pessoal e profissional no qual sempre me apoiou. E a participação direta em momentos muito importantes da minha vida, por exemplo, a entrada na UFRGS.

E dos meus familiares eu agradeço a todos pelos momentos bons que passamos juntos, pois a família é o núcleo da sociedade e ela é o princípio da escultura do caráter humano e quanto a isso eu tive muita sorte. Gostaria de agradecer e muito minha avó Zulma que a tempo não esta mais entre nós, mas que ajudou a fazer o homem que eu sou.

Agradeço também a meu pai, em memória, pois você foi sempre o espelho por onde caminhei e até nunca tive motivos para me arrepender que onde quer que ele esteja, estaremos sempre juntos. Gostar de entender as pessoas como ele. Seus objetivos e posturas são qualidades que não tem preço.

E para finalizar, gostaria de agradecer por último às pessoas mais importantes na minha vida.

Ela é forte, esta sempre do meu lado, me sustenta e me faz crescer. Com seu amor exigente, me guia pelos rumos certos. Trabalhar para viver, ter iniciativa para empreender, perseverar para crescer, ser justo para dividir, amar para construir. Ela realmente imita Deus. Obrigado pelo esforço de ser como mãe o mesmo que Deus é como pai. Obrigado Dona Rosy. Amo-te Mãe.

Existe uma florzinha na natureza muito rara, que ao encontra-la você terá muita sorte, pois ninguém jamais havia encontrado. Ele dá força para continuar, garra para lutar, carinho para se reerguer paz para viver e amor, muito amor. E em troca ela só pede um pouco de atenção. Essa flor ao que dizem, infelizmente, existe apenas uma, e por sorte eu achei ela... O que eu fiz? Plantei lá no fundo do meu coração. Rego todos os dias com carinho e amor e ela tem crescido bem linda e bela. Hoje e sempre noi due. Amo-te Veinha.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLA

VLSI	Very Large Scale Integration (Integração em alta escala)
CWL	Critical Wirelength (Comprimento de Fio Crítico)
WL	Wirelength (Comprimento Total de Fio)
CS	Critical Star
FM	Fiduccia-Mattheyses
FD	Force Direct
SA	Simulated Anneling
BB	Bounding Box
P	Polinomial
NP	Não-Polinomial
KL	Kernighan-Lin
HPWL	Half-Perimeter Wirelength (Semiperímetro)
FPGA	Field Programmable Gate Array
	(Matriz com portas-lógicas programáveis)
STL	Standard Template Library
CAD	Computer Aided Design
LP	Programação Linear (Linear Programming)
GP	Programação Geométrica (Geometric Programming)
IP	Propriedade Intelectual (Intellectual Property)

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho será focado no problema de posicionamento de células lógicas em circuitos integrados. Neste problema necessitamos organizar as portas lógicas reduzindo o comprimento dos fios que as conectam da melhor forma possível. Para entender o problema e as soluções existentes é descrita uma explanação sobre técnicas e algoritmos que são utilizados atualmente ou que são historicamente importantes, de forma a conduzir o texto para as técnicas apresentadas neste trabalho. As técnicas que serão apresentadas neste trabalho têm objetivos individualmente diferentes, porém conduzem a novos resultados e perspectivas em posicionamento. Todas as técnicas são baseadas na modificação e análise do grafo do posicionamento.

Neste trabalho serão apresentadas quatro técnicas para melhorar a solução do problema de posicionamento. O primeiro trabalho a ser apresentado será a Critical Star que aplicado alguns nodos e arestas extras no grafo original para reduzir os caminhos críticos. A segunda técnica é a Logical Core I, ela traz um novo método de análise da dificuldade de posicionar um circuito VLSI. A terceira técnica, que tem forte conexão com a segunda, é a Logical Core II, mais focada em tempo de execução da técnica, ela gera um vetor com todas as dificuldades de posicionar cada célula no circuito. As duas técnicas aumentam o conhecimento do posicionador a respeito do problema e com isso vão de encontro a um ponto muito importante e ainda pouco abordado, a evolução da controlabilidade no posicionamento. A quarta técnica que será apresentada é a Logical Cluster. É uma técnica baseada na Logical Core II, e foi desenvolvida para otimizar os posicionadores já existentes no estado da arte. A técnica é muito eficiente e reduz o tempo de execução do posicionamento e muitas vezes reduz o comprimento de fio.

Palavras-Chave: Posicionamento, Desempenho, Eficiência, Ferramentas de CAD, Microeletrônica.

Placement to Improve Wirelength and Runtime

ABSTRACT

This work is focused on placement problem of VLSI circuits. The goal is organize the logic gates reducing the total wirelength that connect them. A non-effective placement assignment will not only affect the circuit performance but might also make it non-manufacturable by producing excessive wirelength. Then the next step in the assembly line, the routing problem could be insolvable.

In this work will be presents four differents techniques. The techniques are based on changing the graph to improve the placement results. The first one is the Critical Star that applies some extra nodes and edges to reduce the critical paths. The second algorithm is the Logical Core I which brings a new method to analyze the circuit hardness to place a circuit. The third algorithm is called Logical Core II and the focus is generate a vector with hardness to place each cell in the circuit, and increasing the placer information about the problem. The Logical Core I and II, both improving the possibility to compare the hardnesses, in different abstraction levels, and improve the placement controllability. The fourth algorithm is a fast algorithm, based on use the Logical Core II, it creates an effective clustering technique to improve the state-of-art placers results. Reducing the runtime and sometimes improving the wirelength results.

Keywords: Placement, Performance, Efficiency, CAD Tools, Microelectronics.

1 INTRODUÇÃO

A cada dia existe uma demanda maior de projetos de sistemas eletrônicos pelas necessidades de mercado. E com o crescente aumento dos requisitos, cresce também a complexidade dos circuitos desenvolvidos e mais componentes podem ser colocados numa mesma área. Essa complexidade gera um aumento da dificuldade de alcance das metas de velocidade do circuito, de consumo de potência, custo, *time-to-market* (que é o tempo que um projeto demora a ser desenvolvido e estar à venda no mercado), testabilidade entre outras.

É neste nicho que entram as ferramentas de CAD (Computer Aided Design), ferramentas que auxiliam o projetista de circuitos a ter um maior controle dessas variáveis. Muitas das metas de projeto são contraditórias. E sem o auxílio das ferramentas de CAD muitos dos projetos atuais, pela sua magnitude, seriam inviáveis.

O desenvolvimento do fluxo de CAD tem muitas etapas, contudo ele pode ser dividido em quatro grandes grupos: síntese em nível de sistema, síntese em alto nível, síntese lógica e síntese física. Síntese em nível de sistema está relacionada com a modelagem global do sistema. Dividindo o sistema em hardware e software, desenvolvendo o código de acordo com o tipo da aplicação objetivo. A síntese de alto nível esta relacionada com a síntese do hardware e sua descrição comportamental. O objetivo da síntese de alto nível é criar descrições de hardware eficientes, de forma a otimizar a arquitetura do projeto.

A síntese lógica é responsável pela otimização booleana e pelo mapeamento tecnológico, que consiste em descrever o circuito em componentes básicos chamados de portas lógicas ou células. Dentro das metodologias possíveis para o projeto de circuitos integrados digitais, ou ASICs (Application Specific Integrated Circuits), a mais utilizada é a denominada Standard Cells (STL, Stardard Template Library), que se utiliza de portas lógicas pré-projetadas e pré-caracterizadas. Outra metodologia importante é a denominada Full-Custom (onde todo o projeto pode ser customizado, ou seja, adaptado as necessidades específicas de cada projeto. Isso ocorre devido ao fato de não utilizar nenhuma biblioteca específica). Neste modelo temos uma grande capacidade de otimização em potência, área e desempenho. Porém se perde bastante em *time-to-market* e testabilidade.

Porquanto a síntese física faz a transformação do circuito em nível de portas e redes abstratas em algo físico, que pode ser implementado em silício, como na tecnologia atual, ou qualquer outro material que venha a ser desenvolvido. Entretanto as ferramentas que funcionam para o silício não necessariamente serão efetivas da mesma forma para outras tecnologias (nanotubos de carbono, por exemplo). Outro ponto importante na síntese física é obedecer todas as restrições elétricas de cada tecnologia e

com isso garantirem que o circuito funcionará corretamente de acordo com as especificações do projeto.

O leiaute de um circuito integrado define as geometrias das máscaras usadas no processo de fabricação as quais são utilizadas para transferir o processo de desenvolvimento das células para o silício. E a síntese física tem como principal tarefa à geração de leiaute. Mas para chegarmos ao leiaute temos de passar por algumas etapas. Entre elas, a etapa de posicionamento de circuitos onde as posições dos componentes são definidas dentro da área ativa do chip.

Posicionamento tem sido, a mais de 50 anos, extremamente importante no desenvolvimento de circuitos integrados (IC's), pois ele influência diretamente o comprimento dos fios das interconexões. E, à medida que avançamos em tecnologia, aumentamos o impacto do atraso das interconexões em determinar o desempenho dos circuitos VLSI já que o atraso dos fios não reduz tão rapidamente quanto o atraso das portas (Bakoglu 1990 e Sylvester 1998), ou seja, os fios estão se tornando cada vez mais problemáticos para os desenvolvedores como pode ser visto na Figura 1-1. O problema de posicionamento de componentes é NP - Completo (problemas não polinomiais são computacionalmente caros de se resolver de maneira exata), portanto não existe um algoritmo eficiente e exato para solucioná-lo. Então são usados algoritmos heurísticos que nos aproximam da solução ótima.

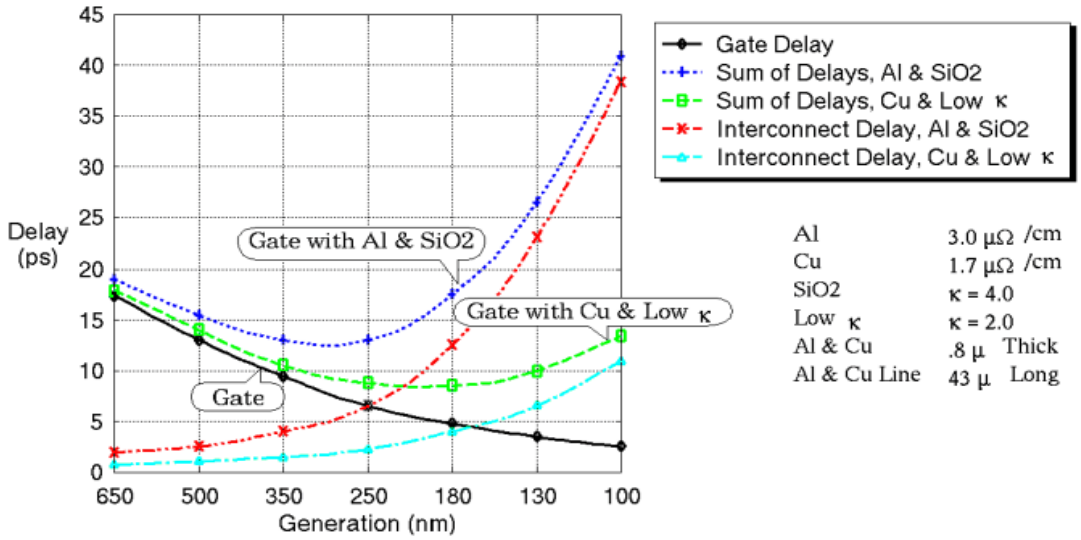


Figura 1-1: Gráfico (reprodução) que demonstra a diferença entre a evolução do atraso dos fios em relação ao atraso das portas lógicas. [ITRS, 09]

A nanotecnologia atualmente impõe muitos desafios para as ferramentas de síntese e muito destes problemas estão relacionados com o comprimento dos fios. Na Figura 1-1 vemos que a linha que representa as portas lógicas tem reduzido o impacto no tempo do circuito, porém as linhas que representam os fios formam uma parábola com inflexão perto de 180nm, o que mostra que atualmente estão influenciando de forma contundente no atraso dos circuitos. O desempenho dos circuitos é um dos objetivos mais importantes. É possível identificar facilmente a disputa de mercado entre grandes empresas para lançar o quanto antes o circuito com melhor desempenho. Empresas como AMD, Intel, entre outras gastam bilhões todos os anos para se posicionarem na

ponta do mercado que gira ainda mais bilhões em venda de processadores. No primeiro trimestre de 2010, a líder de mercado de processadores, Intel, anunciou um total de $10 bilhões em faturamento, $2,89 bilhões em lucros e desse valor a quantia que foi destinada a pesquisa foi mais de $1,5 bilhão só no quarto trimestre (demonstração de resultados do site da Intel na relação com investidores - http://www.intc.com/financials.cfm).

1.1 Posicionamento

O problema de posicionamento recebe como entrada o que é chamado de netlist, que é uma lista de todas as conexões do circuito, ou seja, todas as células do circuito e a forma que elas se conectam. Desta forma é possível ver a netlist como um grafo como nodos e arestas. Devido à evolução das netlists em termos de complexidade ao longo dos anos o posicionamento foi dividido em duas fases de forma que fosse possível criar soluções mais especializadas para cada etapa. Uma fase global que visa distribuir as células de forma uniforme no circuito, mas sem os cuidados de sobreposição de células (overlap) e legalização.

Nesta primeira fase temos um tratamento global das redes do circuito de forma a ganhar tempo. Esta parte é a mais importante para um bom posicionamento rápido, ou seja, boa distribuição de células com um comprimento de fio baixo, isto aumentou muito o seu interesse do ponto de vista científico à medida que não adianta uma solução ótima num prazo longo demais. Neste caso, seria melhor uma solução não tão boa, contudo eficiente. Para o cálculo do comprimento de fio, na etapa de posicionamento, precisaríamos rotear o circuito para ter a qualidade real da solução. Devido ao tempo de execução inviável utilizamos estimativas de comprimento de fio. A estimativa mais utilizada é chamada de Semiperímetro (Half-Perimeter Wirelength, HPWL). Essa estimativa gera um retângulo mínimo que contem todas as células da rede e faz a soma de um lado horizontal com um lado vertical do retângulo gerado.

A segunda fase se chama posicionamento detalhado, recebe-se um problema de posicionamento já com uma solução inicial, porém é preciso retirar todas as sobreposições e deixar todas as células em um local válido sem aumentar demasiadamente o comprimento de fio ou, inclusive, reduzi-lo. Enquanto a parte global precisa ser rápida, a parte detalhada precisa se focar em resolver com qualidade o posicionamento que recebeu, pois nesta parte se houver um número de células muito grande em uma determinada região da área ativa, mesmo que piorando o resultado, o posicionador será obrigado a reajustar as células para ter ao menos um posicionamento válido, obtendo uma solução para o problema.

O desenvolvimento de técnicas que melhorem os resultados em termos de desempenho dos circuitos é muito importante na área de microeletrônica. E diminuir o atraso dos circuitos integrados é reduzir, na verdade, o tempo que uma alteração na entrada de uma porta, ou de um subcircuito, leva para se manifestar na saída. Um dos estudos mais influentes sobre esta dissertação é o da relação entre a estrutura do circuito e a eficiência dos posicionadores desenvolvidos por Liu (2004). Dentro deste estudo, são analisados todos os três tipos de posicionadores existentes: os analíticos, os baseados em Simulated Annealing (SA) e os particionadores, que são testados e comparados frente aos benchmarks mais utilizados que são os (IBM, PEKO) que estão no formato Bookshelf. O trabalho de Liu traz consigo conclusões que foram ponto

inicial para as ideias desenvolvidas nesta dissertação, não necessariamente como verdades absolutas, tendo em vista que inclusive haverá discordâncias entre os dois trabalhos, mas como características importantes que foram retiradas de uma exaustiva pesquisa com muitos dados e discussões.

Esta parte da dissertação objetiva a melhora do tratamento do posicionador para cada circuito específico propondo uma evolução do conceito de controlabilidade. Assim sendo um dos pontos de interesse deste trabalho é saber quais grafos de entrada são mais complexos e quais são mais simples visando se utilizar disso para que se resolva mais eficientemente cada posicionamento. A ideia mais abstrata e inicial é que com uma maior controlabilidade do problema poderíamos, por exemplo, fazer uma pré-analise do posicionamento a partir das técnicas descritas neste trabalho e dependendo da variação do número e do tamanho das redes existentes, é possível ajustar o posicionador para ganhar eficiência no posicionamento. A forma de ganhar eficiência a partir de pré-análises será descrita nos capítulos posteriores.

Algumas conclusões importantes são desenvolvidas, entre elas que a qualidade dos posicionamentos está relacionada com a complexidade das interconexões do circuito. Esse estudo motivou grande parte das análises desenvolvidas neste trabalho.

Todos os posicionadores atuais procuram resolver eficientemente seus problemas e isso traz soluções com excelentes comprimentos de fio e eficientes como, por exemplo, no Fastplace 3 (Viswanathan, 2008), porém isso faz com que a análise empírica domine as soluções dos problemas de posicionamento atuais. A dificuldade implícita trazida com esta abordagem é a pouca compreensão científica do caminho correto para resolver o problema, nesta dissertação vamos expor todos os dados possíveis para aumentar a compreensão das soluções propostas.

Também iremos apresentar um novo posicionador acadêmico chamado PlaceDL (Lima, 2009), que tem como premissa apresentar todos os passos do desenvolvimento para que se consiga reproduzir o posicionador.

1.2 Contribuições

Esta dissertação apresenta uma forma de entender os padrões globais de cada netlist. Sob um ponto de vista mais abstrato, temos um grafo que representa um circuito, onde as arestas são as conexões e os nodos são as portas lógicas. Podemos alterar esta definição, mas sempre objetivando a melhora do tratamento dos posicionadores para cada caso específico mantendo a descrição do circuito.

A primeira parte da dissertação foca na compreensão e aplicação da técnica o atraso dos fios. A *Critical Star* [Pinto, 08] tem o objetivo de reduzir os caminhos críticos de forma transparente para o posicionador. É demonstrada a técnica aplicada em casos de uso, para exemplificar a aplicabilidade e demonstrar os ganhos de utilização da técnica. Dentro da técnica ainda será abordado um tópico a respeito do congestionamento gerado pela técnica, tendo em vista que para conseguirmos continuar com sucesso o processo de síntese física, congestionamento, é um fator fundamental. O problema de congestionamento é definido por uma determinada área onde temos mais células a serem posicionadas neste local do que área ativa disponível. Desta maneira não cabem todas as células nesta parte do circuito e teremos de redistribuir essas células para outras partes do circuito.

A segunda parte desta dissertação foca o desenvolvimento de dois algoritmos chamados Logical Core I [Pinto, 10] e II. Ambos distribuem pesos entre as células de forma que possamos identificar quais delas têm mais impacto no resultado final do posicionamento. No Logical Core I analisa-se a complexidade do grafo a ser posicionado através da soma dos valores probabilísticos gerados para cada célula. Também foi desenvolvida uma alteração na função custo do posicionador que incrementa o semiperímetro com informações globais do circuito de forma a melhor os circuitos posicionados.

A Logical Core II é uma técnica que evolui o algoritmo, mas mantêm o mesmo conceito de buscar as células mais importantes do circuito como a Logical Core I.

Também demonstraremos uma técnica de clusterização que utilizará esta análise para obter melhorias na solução do posicionamento, a técnica é chamada de Logical Cluster. É alcançada uma redução do tempo de execução de 15,1%, mantendo a qualidade do posicionamento.

2 CONCEITOS BÁSICOS PARA POSICIONAMENTO

Este é um capitulo aplicado aos conceitos básicos do problema de posicionamento que traz consigo uma breve visão de como são tratados e divididos. Ainda mostrará algumas das alternativas de solução, o que ajudará a compreender onde a dissertação se encaixa.

2.1 Classificação dos Problemas na Computação

Os problemas de computação são classificados pelo melhor algoritmo conhecido que o resolve como mostrou Cook (1971). Então temos subclasses de tipos de algoritmos. O conjunto de problemas P são aqueles para os quais existe um algoritmo determinístico e polinomial(P) capaz de resolver esse problema. Outro conjunto são os problemas NP, onde existe um algoritmo não determinístico capaz de resolvê-lo em tempo polinomial. Todo algoritmo determinístico é também não determinístico (ou seja, P ϵ NP). Ainda não foi encontrado nenhum problema P que não esteja contido em NP. A Figura 2-1 demonstra visualmente as intersecções e uniões entre os conjuntos de problemas.

Stephen Cook, um dos precursores da complexidade computacional, provou que se um determinado problema, por exemplo, SAT (o problema da satisfatibilidade booleana é o problema de determinar se existe um valor para as variáveis de uma fórmula booleana de tal forma que a solução a satisfaça) classificada como NP completo for resolvido por um algoritmo polinomial determinístico, haverá outros algoritmos P para todos os demais problemas NP (ou seja, seria P = NP). Um subconjunto importante é o chamado NP completo. Estes problemas têm um mapeamento, de tempo polinomial, para SAT.

Porém a realidade da informática é que não podemos fazer algoritmos não determinísticos, já que não podemos dividir o processo de resolução em infinitas máquinas teóricas. Então nossas soluções destes problemas são acima do polinomial (i.e. exponencial, fatorial, etc.).

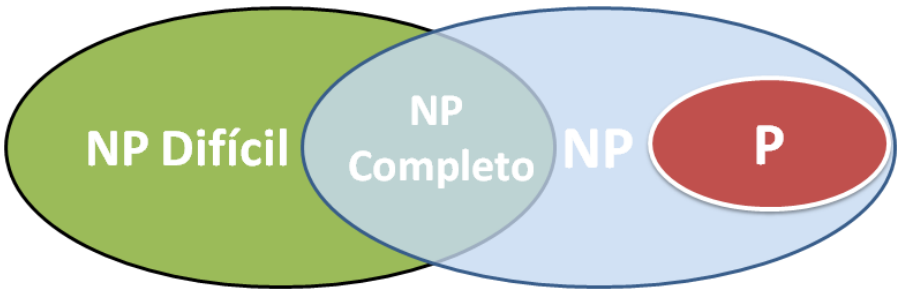


Figura 2-1: Relação entre os problemas computacionais. Problemas NP difícil, os problemas NP e na junção deles temos os problemas NP completos. Ainda temos os problemas P demonstrados como subgrupo dos problemas NP. (Cormen, 2001)

2.2 Problemas de Otimização

São definidos por problemas onde não apenas desejamos uma solução, mas desejamos a melhor solução possível. Este problema é equivalente a percorrer uma árvore em busca da folha que melhor satisfaz uma determinada função de avaliação. Um algoritmo não determinístico seria de simples implementação e fácil, simplesmente dispararíamos um processo para cada ramo da árvore e pediríamos para eles retornarem a melhor folha.

O ponto interessante é que o melhor algoritmo para resolver este tipo de problema é uma busca exaustiva, com tempo exponencial, em uma árvore n-ária de possibilidades. Então problemas de otimização com variáveis discretas são geralmente NP - Completos. Por isso muitas soluções do estado da arte utilizam técnicas de aproximação como programações lineares (LP), quadráticas, convexas, geométricas (GP), etc.

2.3 Busca Exaustiva

A busca exaustiva analisa todas as possibilidades de solucionar um determinado problema e seleciona a melhor. Porém desta forma a complexidade, por muitas vezes, ficará acima de polinomial. Em problemas pequenos este tempo pode ser aceitável. Por exemplo, o posicionador CAPO (Caldwell, 2000), após particionar o problema em pequenos grupos ele aperfeiçoa esses grupos com o algoritmo Branching Bound (Cormen, 2001), não exatamente uma busca exaustiva, mas próximo, que tem uma alta complexidade e é utilizado neste conhecido posicionador, mas para grupos com menos de 35 células, teoricamente pequeno.

2.4 Algoritmos Heurísticos

São algoritmos que a partir de medidas experimentais ou aproximações objetivam conseguir, para a maioria dos casos, resultados próximos do ótimo, porém eficientemente. Eles geralmente exploram características específicas do problema ou resolvem o problema de forma simplificada para aceleram a geração do resultado. Quanto mais características se conseguir encontrar em um determinado problema, melhor e mais eficiente será o algoritmo heurístico.

Existem dois tipos gerais de algoritmos heurísticos: heurísticos subótimos e heurísticos ótimos. Como o próprio nome já diz a diferença entre eles é a qualidade do resultado gerado, ou seja, os ótimos conseguem alcançar a solução exata. Enquanto os subótimos tentam alcançar uma boa solução, porém essa não é ótima.

2.5 A Lei de Moore

A Lei de Moore (1965), diz que a complexidade para componentes com custos mínimos tem aumentado em uma taxa de aproximadamente um fator de dois por ano. E que se espera que esta taxa se mantenha, porém pode aumentar. E no longo prazo, a taxa de aumento é um pouco mais incerta, embora não haja razões para se acreditar que ela não se manterá quase constante por pelo menos 10 anos. Isso significa que em torno de 1975, o número de componentes por circuito integrado para um custo mínimo era 65.000. Após um pequeno ajuste da Lei para o número de transistores, foi dito e até hoje é também chamado de Lei de Moore que o número de transistores existentes em um circuito integrado dobra a cada 18 meses aproximadamente.

Já em 2005, o próprio Gordon Moore, disse que a famosa lei estava com os dias contados, pois estamos chegando ao limite físico para o tamanho de um transistor e para a concepção dos fios de conexão. O que traria a impossibilidade de um processador (na concepção do que conhecemos hoje) dobrasse a sua velocidade, porém é importante declarar que isso não impede a evolução no desempenho das máquinas que se alinham com a tendência de terem mais processadores em um mesmo circuito e ainda evolução na organização, a partir de novas modelagens e algoritmos de CAD melhores.

2.6 Estruturas Específicas para Compreensão do Posicionamento

Um ponto importante para um posicionador é ter uma boa heurística para medir os fios, pois o posicionador não pode tentar rotear todo o circuito a cada iteração, pois seria lento demais, apesar de extremamente preciso. A medição mais correta seria ainda utilizar uma fórmula física de análise do atraso. Como por exemplo, a ferramenta Spice, que modela o fio analisando as suas resistências, capacitâncias e indutâncias. Porém isso também seria muito lento. Então se utilizam aproximações que tentam calcular o valor real do custo do roteamento em termos de comprimento de fio.

2.6.1.1 SemiPerímetro (HPWL)

O cálculo do Semiperímetro (HPWL) é baseado no somatório de metade da menor Bounding Box (BB), que é a que contém todos os pinos da rede (por vezes todas as células da rede caso não utilizamos formatos com descrição dos pinos) para todas as redes que descrevem o circuito a ser posicionado. A Figura 2-2 descreve com um desenho a heurística.

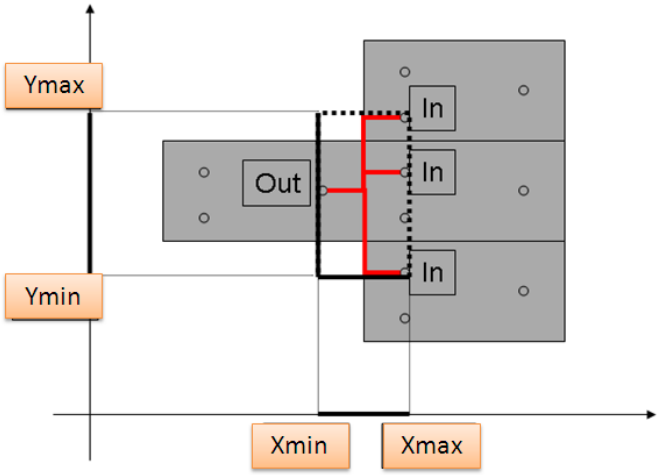


Figura 2-2: Mostra uma célula (OUT) conectada em outras 3 células (IN) de mesmo tamanho e como ficaria o roteamento se utilizássemos uma árvore de Steiner para conectar os quatro pontos indicados pela *netlist.*

$$\sum (X_{max} - X_{min}) + (Y_{max} - Y_{min}) \qquad 2.1$$

2.6.1.2 Grafo Completo

Calcula a distância dentre todos os pontos a conectar. Esta distância pode ser à distância Manhattan ou a distância Euclidiana. Na distância Manhattan só pode-se utilizar retas em 90 graus (sem nenhum tipo de diagonais). Este nome foi dado em homenagem à estrutura da cidade de Manhattan, onde os quarteirões estabelecidos no plano diretor da cidade mantêm todos em 90 graus. Na distância Euclidiana todos os ângulos necessários podem ser utilizados.

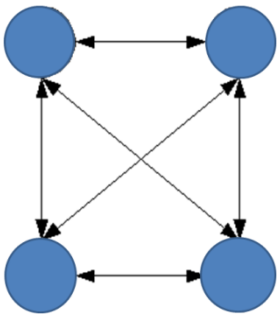


Figura 2-3: Mostra um grafo completo. Neste caso todas as arestas geradas devem ser somadas na função custo do problema.

2.6.1.3 Saída para Entradas

As redes da netlist tem sempre um pino de saída e vários de entrada, como pode ser visto na Figura 2-4. Esta estimativa calcula a distância da saída para todas as entradas. A estimativa consegue mostrar um pouco melhor o cálculo de timing, pois descreve o caminho que o sinal realmente irá fazer. Apesar de ignorar cálculos exatos como, por

exemplo, a relação real dos fios com as resistências e capacitâncias (quadrática), mas também por outro lado, ignora a colocação de buffers que dividem o fio em várias partes e assim reduzem o número de fios longos e linearizam o cálculo. Ou seja, ele comete erros de análise e resulta um valor maior que o real, mas como não desconta o efeito dos buffers reduz os erros.

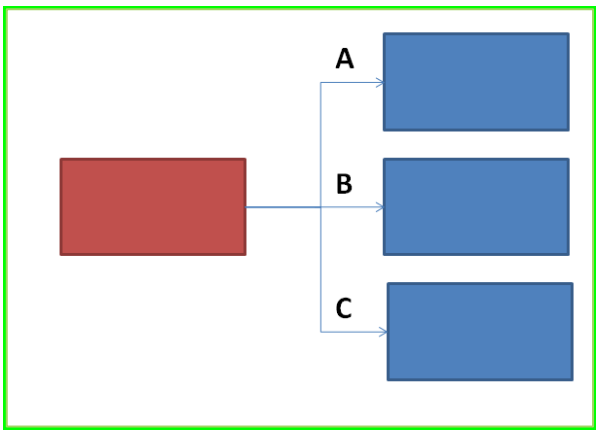


Figura 2-4: Os comprimentos das arestas A, B e C devem ser somados nessa função custo.

2.6.1.4 Menor Caminho

Um caminho é um conjunto de células interconectadas por onde passará um determinado sinal. Esta heurística traz informações globais do posicionamento que técnicas como o semiperímetro não conseguem captar. O problema deste método é que é pouco eficiente quando comparado ao Semiperímetro, pois calcular o menor caminho não tem uma solução trivial. Temos sempre de analisar todos os caminhos modificados e compará-los, além de que muitas vezes teremos um caminho menor, porém uma solução, na realidade, pior.

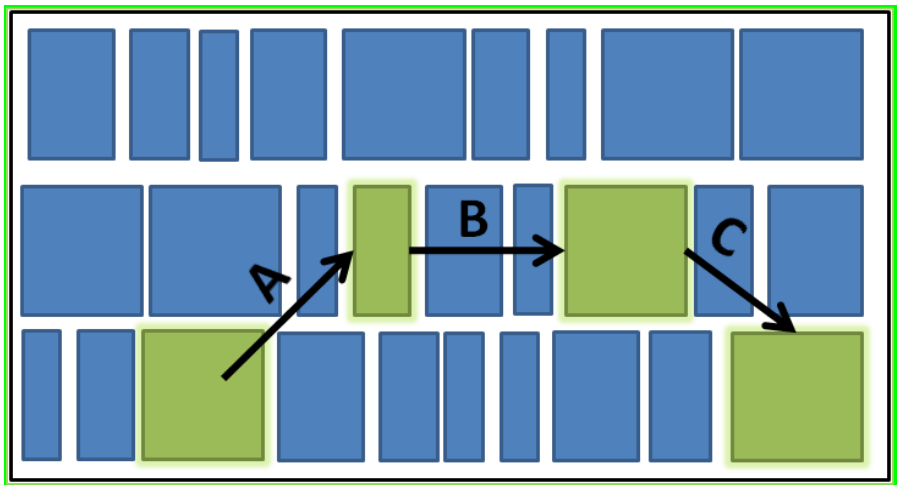


Figura 2-5: Demonstra um caminho, na hipótese dele ser o caminho mínimo entre as conexões.

2.6.1.5 Menor Árvore de Spanning (Cormen, 2001)

O objetivo do método é uma árvore que une todos os pinos da rede. É uma estimativa realista, pois alguns algoritmos de roteamento trabalham com este tipo de estrutura. Porém encontrar a menor Spanning Tree é outro problema de otimização, ou seja, bem demorado frente às outras opções.

2.6.1.6 Árvore de Steiner (Cormen, 2001)

Seria a heurística ideal para o posicionamento, pois é justamente o objetivo do algoritmo de roteamento, achar a menor Árvore de Steiner (Steiner Tree) para todas as redes e com isso as duas ferramentas, posicionador e roteador estariam com o mesmo objetivo final, o que é excelente para conseguir bons resultados. O problema desta estimativa é o tempo de processamento, pois é um problema NP Completo. Na Figura 2-2 o desenho dos fios (em vermelho, formando tridente) demonstra um exemplo de Árvore de Steiner.

Alguns posicionadores utilizam esta heurística para saberem se o posicionamento que finalizaram é um bom posicionamento, pois a heurística é bem real. Caso o posicionamento não fosse considerado bom, poderíamos repeti-lo com modificações baseadas neste primeiro resultado final.

2.6.2 Regra de Rent (Cormen, 2001)

A regra tem importância fundamental para este trabalho, pois aqui será proposto um novo método de calcular a complexidade de uma netlist de entrada do circuito, que se baseia em um valor relativo, assim como a lei de Rent, porém não utiliza nenhuma medição prévia, como por exemplo, o número médio de redes por bloco, apenas aplica o algoritmo desenvolvido com as probabilidades como será visto posteriormente na técnica Logical Core I desta dissertação.

A regra de Rent data de 1971 quando Landman (1971) fez uma observação empírica das células e suas conexões externas em uma partição do circuito. Esta medida tem uma forte relação com posicionamento global, que atualmente, é o grande problema em posicionamento, pois os circuitos estão cada vez maiores com elementos cada vez menores e mais conectados.

Em uma predeterminada região com G células e T interconexões cruzando o bloco, a lei relaciona G com T desta forma:

$$T = tG^{p} \qquad 2.2$$

t é o número médio de redes por bloco e p é o expoente de Rent. O expoente varia de 0,4 a 0,8 em circuitos reais.

Um expoente alto significa que em média o comprimento dos fios será maior e com isso o congestionamento também tende a ser maior.

2.6.3 Desempenho e Caminhos Críticos

No desenvolvimento de circuitos integrados estar atento a desempenho é sempre importante. E não raro este objetivo é a meta principal do desenvolvimento.

Em posicionamento, conseguir o circuito de melhor desempenho é exatamente identificar os caminhos críticos e diminuí-los tendo o cuidado de não aumentar os

outros caminhos a ponto de tornarem-se críticos. Um caminho crítico é formado por portas lógicas interligadas aonde existem os maiores atrasos de um determinado circuito integrado. Para diminuir o caminho critico é necessário aproximar as células que participam do caminho, para que os fios que as conectarão fiquem menores.

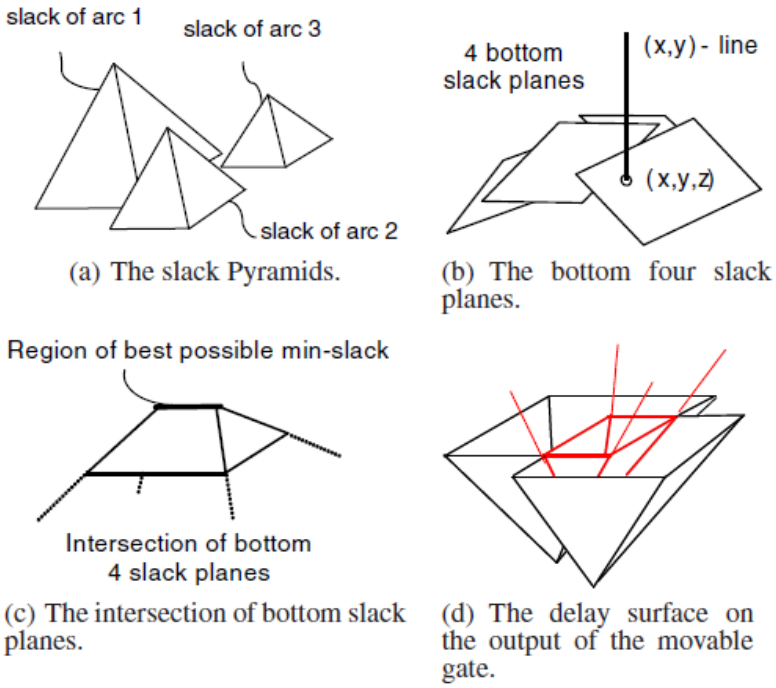
As técnicas de *Timing-driven* (dirigidas a desempenho) são classificadas como *net-based* ou *path-based* (RIESS, 1995; HUANG, 1997; EISENMANN, 1998). Os dois tipos têm suas vantagens e desvantagens.

Os algoritmos net-based são aplicados em todas as redes do circuito e com isso tornam-se mais escaláveis e por isso menos detalhistas nos ajustes. Os algoritmos net-based controlam o atraso impondo um pior caso possível ou então setando um peso para cada rede. E os algoritmos path-based recebem (ou geram) os caminhos críticos e trabalham neles para otimizar o atraso do circuito, com isso são menos escaláveis, mas têm, normalmente, a vantagem da complexidade do tempo de execução da técnica. As técnicas *path-based* modelam o problema corretamente, porém não conseguem ter escalabilidade, ou seja, têm de ser aplicados em pequenos circuitos, ou em uma parte do circuito, já que tratar todos se tornaria impossível. Em geral as técnicas existentes iteram entre uma etapa de ajuste de desempenho e uma de posicionamento. Contudo no estado da arte a mais eficiente técnica apresentada utiliza o que foi chamado pelo autor de *Geometric-based Approach* (Luo e Papa, 08). A ideia é transformar o problema de otimização de desempenho em um problema de otimização geométrica que utiliza um cálculo com intersecções de pirâmides de tempo (*delay*) pela distância da posição ótima para o posicionamento da célula.

A equação que traça a reta do tempo ótimo para duas portas é a seguinte:

$$X(m,n) = X_m - 0{,}5(X_p + X_q) \tag{2.3}$$

onde Xm é a distância horizontal entre m e o centro horizontal da rede de m, sem contar m neste cálculo. Assumindo que a rede n é delimitada pela porta p e pela porta q na dimensão x. Na Figura 2-6 vemos a explicação do próprio autor de como seria a análise geométrica.



(a) The slack Pyramids.

(b) The bottom four slack planes.

(c) The intersection of bottom slack planes.

(d) The delay surface on the output of the movable gate.

Figura 2-6: Retirado do artigo original, aqui temos um resumo de como funciona a técnica baseada na análise geométrica. Resumidamente temos que interseccionar as pirâmides para saber o conjunto validado de possibilidades de solução. [Luo e Papa, 08]

Com esta técnica o número de reajustes da análise de desempenho reduz bastante, principalmente para as primeiras iterações. Em geral a etapa de desempenho é responsável por encontrar os caminhos críticos e dizer as redes com maior peso. Enquanto o posicionamento ira resolver a posição de cada célula para que as questões de desempenho sejam levadas em consideração.

Existe outro ponto fundamental não citado ainda nesta dissertação é a questão dos *buffers* (reforçadores) que são nos circuitos desenvolvidos atualmente fator determinante para o objetivo de desempenho. Na sua forma mais simples são duas portas negadoras (*NOT*) inseridas no centro de um fio que se deseja reduzir o atraso (*delay*). Eles são utilizados como forma de redução do tempo necessário para que uma porta emita o seu sinal lógico até a porta conectada em sua saída, ou seja, a saída e a entrada fiquem eletricamente equivalentes. Isto ocorre, pois o fio que conecta as duas portas lógicas respeitam a seguinte fórmula de atraso:

$$T_{wire} = 0{,}7R_0 \sum_{i=1}^{N} (C_w)_i + \sum_{i=1}^{N} (R_W)_i \left(0{,}4(C_W)_i + 0{,}7(C_L)_i + 0{,}7\sum_{j=i+1}^{N} \left((C_L)_j + (C_W)_j\right)\right) \quad 2.4$$

Visto a fórmula podemos reparar que o valor do atraso cresce exponencialmente com relação ao comprimento do fio, então quando se insere um reforçador o atraso diminui. Um exemplo abstrato que explica bem a ideia é o seguinte: se tivéssemos um fio com 10 unidades de resistência, ao elevar esse valor ao quadrado teríamos o valor 100. Mas ao inserir um *buffer* exatamente no meio do fio, teremos duas metades de fio,

que são duas vezes 5. Então aplicando a mesma ideia elevamos cada metade ao quadrado, isso resulta em 2 vezes 25 que é 50. Ainda devemos levar em consideração o tempo de chaveamento do *buffer*, o que mostra que para o caso hipotético criado o tempo de atraso do *buffer* teria de ser maior ou igual a 50 para não valer a pena utilizá-lo. O atraso do *buffer* é geralmente bem menor nas tecnologias recentes.

O buffer são duas portas lógicas NOT (inversoras) uma conectada na outra, como mostra a **Figura 2-7**.

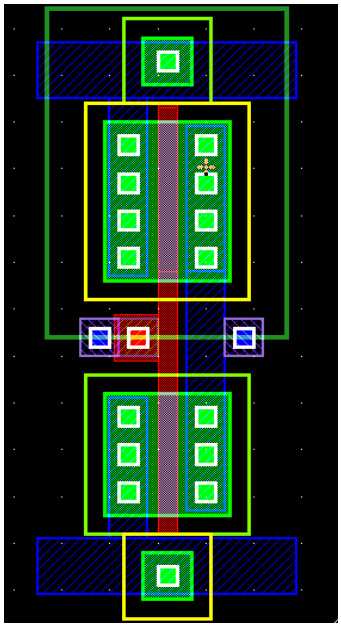


Figura 2-7: Mostra uma célula inversora que compõe um buffer.

Porém o que deve ser destacado aqui, como motivação, para as técnicas de desempenho ser tão importantes é que nos circuitos atuais, na média, 35% das células são *buffers*, o que demonstra a grande necessidade que se têm atualmente. Relembrando que a qualidade do posicionamento e do roteamento diminuem a necessidade de inserção de buffers. E segundo alguns estudos ao ultrapassarmos a tecnologia de 22nm (tecnologia atual) os *buffers* serão 70% da área ativa (Madden, 05) o que para muitos, entre esses o próprio Dr. Patrick Madden, seria o fim do desenvolvimento como conhecemos a menos que tenhamos soluções bem eficazes para desempenho sem utilizarmos tantos *buffers*.

3 POSICIONAMENTO

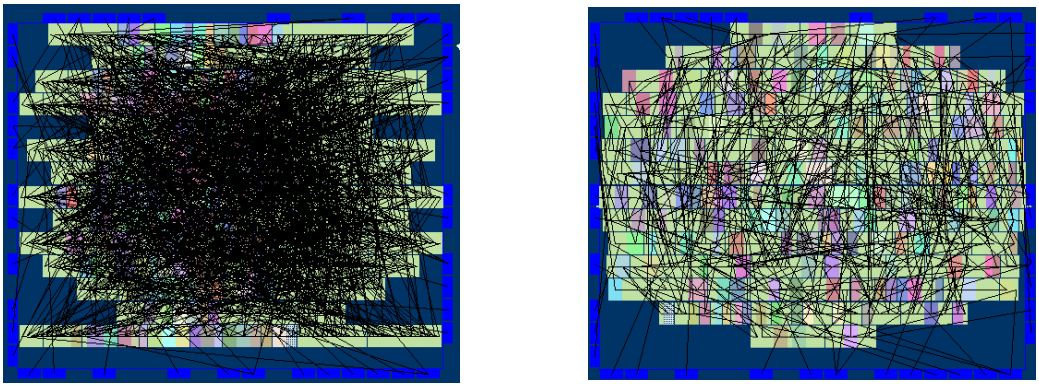
O posicionamento pode ser descrito com a etapa da síntese responsável por encontrar uma posição física para cada célula no circuito. As células do circuito podem ser componentes lógicos de vários tamanhos diferentes, podem ser transistores e até grandes sistemas empacotados (*IPs*).

Como entrada, um posicionador recebe a descrição do circuito (*netlist*) que contém informações sobre as dimensões e conectividade entre componentes. Uma *netlist* é um hipergrafo onde os nodos representam as células e as hiperarestas às redes do circuito. As redes, por sua vez, descrevem quais as células do circuito devem ser eletricamente equivalentes, ou seja, que devem estar conectadas por fios no leiaute final do circuito.

O principal objetivo do posicionamento é encontrar uma posição válida para cada componente de um circuito integrado enquanto avalia a melhora de um ou mais parâmetros específicos que podem ser comprimento do fio (WL), ou área do circuito, ou congestionamento, ou comprimento do fio dos caminhos críticos (CWL), entre outros. Além de deixar o circuito roteável, que significa, o resultado precisa ser bom o suficiente para que o algoritmo do roteador consiga traçar rotas e conectar todos os fios como descrito na *netlist* de entrada.

Dentre os parâmetros citados o mais importante é o comprimento dos fios, também porque o próximo passo após posicionamento (o passo de roteamento) necessita de uma boa distribuição das células. Se a distribuição for ruim poderemos ter uma inviabilidade de solução no próximo passo. O problema de roteamento também é NP completo e consiste em definir exatamente onde passarão os fios e quais materiais serão utilizados no leiaute final.

A Figura 3-1(a) apresenta uma solução randômica onde notamos o número excessivo de fios atravessando todo o circuito e a Figura 3-1(b) apresenta um posicionamento desenvolvido pelo algoritmo *Simulated Annealing*. Ao analisarmos as duas figuras, notamos uma grande diferença de fios atravessando o circuito. Esta analise visual verifica que o resultado do circuito Figura 3-1(a) é bastante pior em termos de comprimento de fio do que o resultado em Figura 3-1(b).



(a) Posicionamento Randômico (b) *Simulated Annealing*.

Figura 3-1: A importância do posicionamento para reduzir o comprimento dos fios. [Hentschke, 98]

Dentre os algoritmos existentes, nós podemos classificá-los em três grandes categorias:

a) *Simulated Annealing* (Sechen, 86; Su, 05);
b) Particionamento (Agnihotri, 05; Caldwell, 00; Chen, 05; Wang, 00);
c) Posicionamento Analítico (Chan, 05; Chen, 06; Eisenmann, 98; Hu, 05; Kahng, 05; Kleinhans, 91; Spindler, 06; Viswanathan, 05 ; Kim, 10).

Neste trabalho, estão descritos com maior profundidade um dos posicionadores de cada grupo. O Fastplace3 como analítico, o CAPO como particionador e o *Dragon* (Karypis, 1997) como baseado no algoritmo Simulated Anneling. Ainda existem algumas soluções híbridas que misturam as técnicas dos três grandes grupos já existentes (Taghavi, 2006) e (Jiang, 2006).

Os algoritmos baseados em *Simulated Annealing* (SA) têm boas chances de encontrar a solução ótima global do sistema, o problema que existe é que em geral os posicionadores que foram implementados nesta linha têm problemas de tempo de execução.

A segunda alternativa de solução adotada é o particionamento recursivo da *netlist* a partir de uma função custo que levam em consideração os fios que passariam por uma determinada região e o tamanho da área utilizada.

A terceira grande alternativa são os posicionadores analíticos que tem como principal objetivo minimizar a função custo utilizando métodos matemáticos. E dentro dos analíticos ainda é possível dividir em dois subgrupos, os que resolvem otimizações não lineares e os posicionadores quadráticos. O primeiro subgrupo é representado pelas construções do sistema como uma função não linear, por exemplo, pela resolução das funções log-soma-exponenciais e para isso é utilizado às otimizações com gradiente conjugado ou biconjugado.

O segundo grupo é formado pelos posicionadores quadráticos, pois tem como objetivo uma função quadrática, devido a uma modificação na equação geral do problema. Com esta modificação na equação, esses algoritmos conseguem resolver um sistema linear para alcançar os resultados. Isto lhes oferece uma vantagem em termos de complexidade do problema. É importante destacar que com esta abordagem o objetivo é o quadrado do comprimento de fio e não o próprio comprimento de fio. O que resulta em um objetivo indireto do que realmente se deseja. Existem na literatura alguns

trabalhos alteram a função custo e conseguem aproximar com o valor real da *Steiner Tree* mínima.

Os posicionadores quadráticos podem ainda ser divididos em dois grupos dependendo da forma com eles tratam os *overlaps* do circuito, isto porque ao resolver o sistema linear as células tendem a se agruparem no centro do circuito e criar muito *overlap*. Os baseados em *Constraints*, que são na verdade restrições a partir dos centros de massa da solução e os *Force-Direct* (FD), forças extras são adicionadas para espalhar as células.

Na sequência da dissertação serão mostrados com detalhes os três tipos de solução possíveis. Primeiro os posicionadores analíticos que tem como representantes no trabalho apenas os quadráticos que compõem a grande maioria dos posicionadores analíticos atuais. Então serão apresentados os algoritmos particionadores, incluindo o posicionador CAPO. Por fim será apresentado a técnica *Simulated Annealing* e um posicionador que a utiliza chamado *Dragon* (Karypis, 1997).

3.1 Posicionamento Analítico (Quadrático)

Um tipo específico de solução é o posicionamento quadrático. No posicionamento quadrático, modela-se o problema como um sistema de molas (convexo e quadrático) e então se resolve diversos sistemas lineares que indicarão qual a posição das células que deixam o sistema em equilíbrio. Existem algumas técnicas de solução do sistema de equações, um exemplo comumente utilizado é o método do Gradiente Conjugado pré-condicionado com fatorização incompleta de Cholesky (Kershaw, 1978) da matriz Q como pré-condicionadora. Na fórmula vemos que quando elevamos a equação ao quadrado, conseguimos retirar a raiz da função. Essa retirada facilita a resolução do sistema, porém afeta o resultado que conseguiremos. Primeiramente o resultado será o quadrado do comprimento de fio, então se o valor de um fio for um de comprimento, teremos o resultado exato, mas quanto mais distante de um, pior será o resultado. Outro fato que se pode concluir é que os fios maiores serão ainda mais prejudicados na função, pois na equação vemos que o comprimento dos fios será elevado ao quadrado.

Função Quadratica de 2 pontos, identificados por x e y , onde

$$\Phi(x,y) = \frac{1}{2}\sum_{i,j=1}^{n}\sqrt{\left[(x_i - x_j)^2 + (y_i - y_j)^2\right]^2}$$

$$\Downarrow \qquad 3.1$$

$$\Phi(x,y) = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2 + \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(y_i - y_j)^2$$

Primeiramente, modela-se a *netlist* de entrada como um grafo através dos modelos de redes. Então, por último, o modelo de rede divide as hiperarestas do grafo.

O comprimento de fio será modelado com a distância Euclidiana quadrática entre os nodos e então o resultado tenderá a deixar as células todas na mesma posição para diminuir o atributo comprimento de fio, então precisamos de outro método para espalhar este resultado.

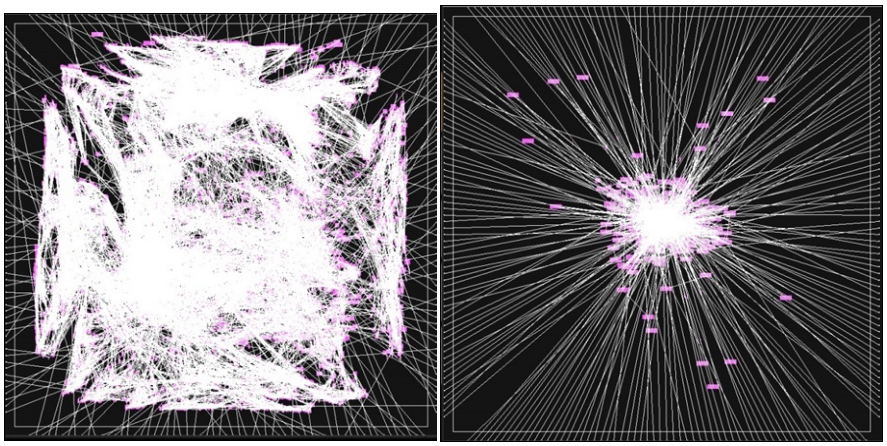


Figura 3-2: Um posicionamento quadrático global com a solução inicial e final. (imagem retirada do conjunto de *Benchmarks* IBM)

Um ponto importante para entender como o posicionamento quadrático funciona é entender como se constrói e como se utiliza a matriz de Admitância. A matriz de Admitância será a base para a solução do sistema linear. Aprofundar em excesso a Álgebra Linear não é necessário, porém uma ideia geral é importante.

A matriz de Admitância é descrita a partir das conexões existentes no grafo do posicionamento. Se houver uma conexão entre uma determinada célula A e uma célula B na matriz na posição A, em B haverá o valor -1, caso não haja a conexão o valor da mesma posição será zero. Assim será para toda a matriz menos a diagonal, onde cada campo da diagonal conterá o absoluto da soma dos valores de toda a linha á qual aquela diagonal pertence.

3.1.1 Posicionador Quadrático Fastplace

Nestes capítulos analisaremos alguns posicionadores quadráticos, entre eles o posicionador Fastplace (Viswanathan, 2005). Os posicionadores desenvolvidos dentro do grupo de CAD. São eles o Z-Place (Hentschke, 2008) e o PlaceDL (Lima, 2009). Será visto como eles resolvem o problema.

3.1.1.1Modelos de Redes

Uma rede é um conjunto de nodos, que devem ser eletricamente equivalentes. O modelo de *nets* define como conectaremos esses nodos. Os dois modelos de redes mais comuns são o modelo clique e o modelo estrela. O primeiro modelo conecta todos os nodos da *net* entre si. (**Figura 3-3**(a)). O segundo conecta cada nodo com um nodo virtual chamado do nodo estrela (**Figura 3-3**(b)).

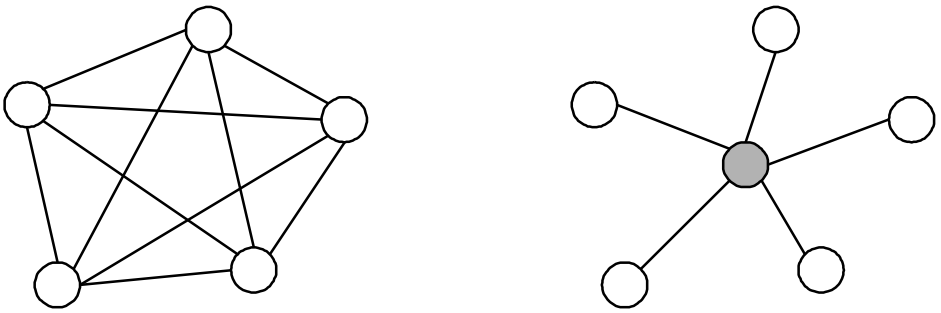


Figura 3-3: Modelos de Redes. Clique à esquerda e estrela à direita

Para posicionadores quadráticos, (Viswanathan, 2005) mostrou a equivalência do modelo de clique e do modelo estrela quando definimos o peso das conexões corretamente (prova no próximo subcapítulo). O modelo estrela tem a vantagem de diminuir o número de não zeros na matriz de admitância (matriz que utilizamos para solucionar o sistema linear) e com isso aumentar a velocidade da resolução do problema. Foi utilizado o modelo clique para as redes com dois e três nodos e para as maiores (quatro ou mais) utilizamos o modelo estrela.

3.1.1.2 Equivalência dos Modelos

A equivalência dos dois modelos foi provada por (Viswanathan, 05) e tem grande importância no desenvolvimento deste trabalho pela velocidade imposta para o posicionamento dos circuitos.

A primeira premissa é que para qualquer net do modelo estrela, o ponto de equilíbrio do pino estrela está posicionado no centro de gravidade de todos os nodos reais desta rede. Então nós sabemos que o nodo estrela estará sempre no baricentro de todas as células, de acordo com o peso de cada conexão. Consideremos uma rede de *k* nodos, onde Xs é a coordenada "x" e Ws seja o peso dessa conexão. Então, a força total sobre os nodos é dada por:

$$F = \sum_{j=1}^{k} W_s(x_j - x_s)$$

No ponto de equilíbrio, o total da força é zero (F=0). E com isso:

$$x_s = \frac{\sum_{j=1}^{k} x_j}{k}$$

E o primeiro teorema diz:

Para uma rede de *k* nodos, se o peso de uma rede de duas células é definido no modelo Clique e no modelo Estrela, então o modelo clique é equivalente ao modelo estrela no posicionamento quadrático. A força atuando numa célula de uma rede conectada através do modelo de clique é dada por:

$$F_i^{clique} = W_c \sum_{j=1, j!=i}^{k} (x_j - x_i)$$

Para o modelo Estrela, todos os nodos da rede estão conectados o pino (nodo) estrela. A força em um pino feita pelo nodo estrela é dada por:

$$F_i^{star} = k.W_c(x_s - x_i)$$
$$= k W_c \left(\frac{\sum_{j=1}^{k} x_j}{k} - x_i \right)$$
$$= W_c \left(\sum_{j=1}^{k} x_j - k x_i \right)$$
$$= W_c \sum_{j=1, j!=1}^{k} (x_j - x_i)$$
$$= F_i^{clique}$$

Como nós temos a mesma força atuando em ambos os modelos, é mostrado que o modelo clique é equivalente ao modelo estrela no posicionamento quadrático.

3.1.1.3 Legalizando

Nesta etapa é necessário modificar a matriz de conectividade para que a próxima solução do sistema linear resulte em novos pontos. De forma a evoluir a solução. Caso esta força "extra" que conecta a célula na borda do circuito não existisse, a solução do sistema linear seria sempre o mesmo. Então se cria forças que levem a célula que agora tem uma nova posição, fisicamente para esta nova posição ou pelo menos o mais próximo possível da posição de acordo com a solução gerada.

E para isso o FastPlace adiciona pinos virtuais, em uma técnica chamada de *Cell Shifting*, que se conectam na borda da área ativa do posicionamento até a célula que se deseja "puxar". Para calcular a força utilizamos a Lei de Hooke (que é uma lei física relacionada com a elasticidade dos corpos).

$$F = k.\sqrt{\Delta x^2 + \Delta y^2} \qquad 3.2$$

onde temos que F é a constante da mola, k é uma constante que no posicionamento significa o peso da conexão entre duas células e d é a distância entre as células.

E como um ajuste final ao posicionamento temos o chamado *Iterative Local Refinement.* Esta etapa consiste em dividir a área ativa de posicionamento em regiões (iguais a menos de divisão não inteira), como se fossem "caixas" de células. Então há uma iteração para cada célula do circuito, onde a técnica testa cada célula nas "caixas" adjacentes de forma gulosa, ou seja, se houver uma melhora no comprimento de fio sem aumentar demasiadamente a ocupação da área ou "caixa", a mudança é aceita. Após esta iteração é diminuído o tamanho das caixas e são refeitos todos os testes de células nas novas caixas adjacentes. Isso tem de ser feito até que não consigamos mais melhorias significativas com a técnica.

3.1.2 Posicionador Quadrático Z-Place

O Z-Place (Hentschke, 2008) tem as mesmas características e a base teórica do posicionador FastPlace, porém ele consegue posicionar circuitos 3D (tema que não será abordado neste trabalho) e foi implementado pelo grupo de CAD da UFRGS (GME-CAD). Neste trabalho ele foi utilizado para posicionar os circuitos 2D, portanto não se utiliza aqui todo o potencial do posicionador.

3.1.3 Posicionador Quadrático Global PlaceDL

O PlaceDL é um posicionador global e foi construído sobre a base teórica do posicionamento quadrático. Ele foi desenvolvido pelo grupo de CAD da UFRGS (Lima, 2009). Mais especificamente para implementação do PlaceDL tomamos como base o posicionador global do FastPlace. Nele implementamos uma nova técnica para o espalhamento de células além de descrever os requisitos necessários para que os resultados obtidos com o PlaceDL possam ser reproduzidos.

Os principais pontos positivos do PlaceDL são:

- Um posicionador rápido e com resultados relevantes perto de outros posicionadores tradicionais na literatura como CAPO (Caldwell, 2000) e FastPlace (Viswanathan, 05).

- Descreve todos os valores usados de forma que os resultados obtidos com o PlaceDL possam ser totalmente reproduzíveis. O FastPlace, o NTUPlace, entre outros apesar de ter resultados bons e obtidos muito rapidamente, não apresentam todos os detalhes para que uma implementação possa obter os mesmos resultados.

O fluxo completo é dividido em duas etapas maiores: difusão controlada e aprimoramento do comprimento de fios. Essas duas etapas serão mais bem explicadas nos capítulos seguintes.

3.1.3.1 Difusão Controlada

O PlaceDL, em uma análise da estrutura geral, intercala dois métodos com objetivos opostos: (1) minimização do comprimento de fios através da solução de um sistema linear e (2) minimização da sobreposição através de um método que simula a difusão das células das regiões mais densas para as menos densas. Tendo em vista que a minimização do comprimento de fios (WL) tende a aglomerar as células do circuito, diminuindo o comprimento de fios. Já a minimização da sobreposição espalha as células aumentando significativamente o comprimento de fios.

Inicialmente para a difusão das células, usamos a solução do sistema linear, a qual minimiza o comprimento de fios. O processo de difusão é iterativo e em cada iteração a utilização das regiões é recalculada. Então as células são espalhadas baseadas no gradiente das densidades de cada região do circuito. As células são difundidas de pouco em pouco até que haja uma diminuição de mais de 25% na utilização da região mais densa do circuito ou até que haja um aumento na utilização em relação à iteração anterior, piorando o resultado já obtido. Também é importante ressaltar que pequenos aumentos na utilização máxima podem ocorrer já que o processo de difusão é calculado de maneira discreta.

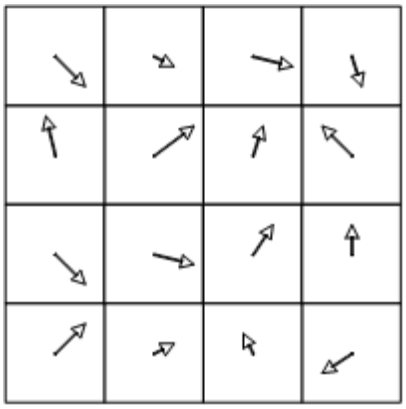


Figura 3-4: Representa a ideia da difusão controlada, onde para cada partição do circuito é calculado um vetor de ocupação.

3.1.3.2 Aprimoramento do Comprimento de Fios

Depois do processo de difusão controlada ter sido concluído, é necessário adicionar novas forças ao sistema linear para mantê-las nas novas posições calculadas. O sistema é modificado adicionando-se forças que puxam as células da sua posição original para a nova posição definida pelo processo de difusão.

Como podemos entender o problema de posicionamento quadrático como um sistema de molas. Desta forma, para atrair a célula para sua nova posição adicionamos uma mola conectada na célula em questão e na borda do circuito. A posição onde uma das extremidades da mola é anexada na borda do circuito, fica na intersecção da reta que passa pela posição antiga e nova da célula, na direção de movimento da célula com as bordas do circuito.

A seguir explicamos como a força de difusão, F = (F_x, F_y), sentida por uma célula que deve está difundindo de (x_{old}, y_{old}) para a posição (x_{new}, y_{new}) é calculada. Como F_x e F_y são computadas de maneira análoga, nos concentraremos somente em F_x. Desta forma, temos:

$$Fx = \mu * (Xnew - Xold) \tag{3.3}$$

em que μ é o coeficiente de difusão que é calculado através da seguinte equação:

$$\mu = 40 * (1 - e^{\frac{-10}{\max U}}) \tag{3.4}$$

Sendo maxU a densidade da região mais densa. Note que μ varia conforme maxU tendo valores menores quando maxU é grande e aumentando à medida que maxU vai diminuindo.

Essa variação no coeficiente de um nível mais baixo para um nível mais alto é para evitar que no início do processo, quando há grande quantidade de sobreposição, as células sejam difundidas para muito longe da sua posição original, causando assim um

acréscimo elevado no comprimento de fios de forma descontrolada. À medida que as células vão se espalhando, à distância na qual as células são movidas pela difusão é naturalmente diminuída, e então um coeficiente de difusão maior é usado.

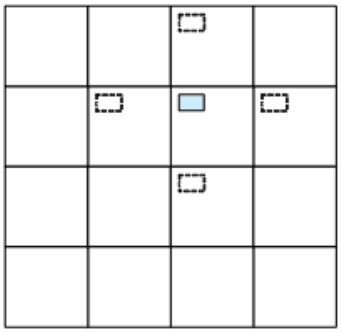


Figura 3-5: É demonstrado o aprimoramento da qualidade do comprimento de fio a partir da técnica de testar as células em caixas imaginárias vizinhas ao local da célula.

De posse da nova posição e da força de difusão sentida pela célula, podemos adicionar a nova força ao sistema. Como a nova força é modelada por uma mola conectando a célula e uma posição fixa na borda do circuito, apenas a diagonal e o vetor do lado direito precisam ser atualizados na posição correspondente a célula.

Na diagonal somamos o valor de β, onde:

$$\beta = \frac{\sqrt{Fx^2 + Fy^2}}{d} \qquad 3.5$$

sendo *d* a distância euclidiana entre a posição original e nova da célula. Ao elemento correspondente a célula no vetor do lado direito, adicionamentos βx_{pin} onde x_{pin} corresponde à posição x onde a mola foi conectada na borda do circuito.

Resultados apresentados pelo PlaceDL

Tabela 3-1: Estão descritos os tempos de execução e os resultados em comprimento de fio da comparação do PlaceDL com o CAPO e com o FastPlace. São utilizados os benchmarks IBM.

	Comprimento de Fios Semi-Perímetro			Comprimento de Fio (Relativo)		Tempo de Execução			Tempo de Execução (Relativo)	
	Capo (X1e6)	FastPlace (X1e6)	PlaceDL (X1e6)	Capo / PlaceDL	FastPlace / PlaceDL	Capo	FastPlace	PlaceDL	Capo / PlaceDL	FastPlace / PlaceDL
ibm01	1.86	1.84	1.94	0.96	0.95	1m 54s	8s	35s	3.26x	-4.35x
ibm02	4.10	3.82	3.88	1.06	0.98	3m 24s	17s	1m 32s	2.22x	-5.56x
ibm03	5.34	5.15	5.34	1.00	0.96	4m 01s	17s	1m 18s	3.09x	-4.55x
ibm04	6.26	6.04	6.46	0.97	0.93	4m 58s	21s	1m 38s	3.04x	-4.76x
ibm05	10.59	10.56	10.25	1.03	1.03	5m 17s	15s	2m 02s	2.60x	-8.13x
ibm06	5.87	5.30	5.35	1.10	0.99	5m 49s	27s	1m 35s	3.67x	-3.52x
ibm07	9.60	8.60	9.50	1.01	0.91	8m 49s	44s	2m 54s	2.76x	-3.95x
ibm08	10.28	9.54	9.69	1.06	0.98	9m 12s	47s	3m 53s	3.02x	-4.96x
ibm09	10.46	10.46	10.68	0.98	0.98	10m 39s	54s	4m 11s	2.55x	-4.65x
ibm10	19.71	18.47	18.89	1.04	0.98	13m 22s	1m 8s	5m 48s	2.30x	-5.12x
ibm11	15.60	15.11	15.37	1.01	0.98	14m 28s	1m 4s	4m 50s	2.99x	-4.53x
ibm12	25.41	24.06	23.85	1.07	1.01	13m 55s	1m 15s	6m 20s	2.20x	-4.27x
ibm13	19.13	18.72	16.68	1.15	1.12	17m 54s	1m 30s	6m 17s	2.85x	-4.19x
ibm14	36.01	34.91	35.73	1.01	0.98	31m 34s	2m 27s	16m 01s	1.97x	-6.54x
ibm15	44.27	43.00	43.03	1.03	1.00	39m 44s	3m 57s	26m 37s	1.49x	-6.74x
ibm16	49.69	45.43	46.36	1.07	0.98	41m 26s	4m 03s	31m 33s	1.31x	-8.53x
ibm17	69.73	66.38	64.15	1.09	1.03	44m 50s	4m 21s	42m 01s	1.07x	-10.46x
ibm18	47.10	45.40	46.40	1.02	0.98	46m 48s	4m 44s	45m 21s	1.03x	-9.58x
			Média	**1.04**	**0.99**			**Média**	**2.41x**	**-5.26x**

Analisando a Tabela 3-1 podemos observar que posicionador PlaceDL obtém resultados cerca de 4% melhor que o posicionador Capo e, em média, com tempo de execução menor. Contudo, à medida que o tamanho do circuito aumenta, o PlaceDL vai perdendo a vantagem de tempo em relação ao Capo, apesar de ir ganhando em melhora de comprimento de fio.

Em relação ao FastPlace, o PlaceDL apresenta resultados semelhantes em comprimento de fios, mas é, em média, 5x mais lento, o que compromete em muito a solução alcançada para comparação. No capítulo onde explicaremos a técnica de Clustering desenvolvido a partir da Logical Core, veremos um ganho em termos de tempo que poderá evoluir o posicionador neste ponto. O PlaceDL tem sido evoluído e os resultados parecem promissores.

3.2 Particionadores

Este capítulo merece atenção especial devido à importância dos particionadores para muitos dos tipos existentes de posicionadores desenvolvidos com sucesso em termos de comprimento de fio. O problema de particionamento é útil em várias outras áreas além da microeletrônica como *data-mining* (mineração em bases de dados), armazenamentos de dados em discos entre outras.

O problema consiste em dividir os vértices de um hipergrafo em *n* partes similares tais que o número de hiperarestas que conectam os vértices de cada uma das partes é minimizado. Como já dito anteriormente, um hipergrafo é a generalização de um grafo, onde o conjunto de arestas é substituído por um conjunto de hiperarestas. Uma hiperaresta pode conectar mais de dois vértices. Simplificando, o princípio básico de um particionador é que dado um circuito constituído de células (vértices) conectadas por fios (arestas), deve-se encontrar conjuntos constituídos por células, onde o número de conexões entre os conjuntos seja mínimo.

Não se conhece um algoritmo capaz de encontrar uma solução ótima em tempo polinomial, sendo este, outro problema NP completo. Dentre as heurísticas propostas para a solução do problema de particionamento temos (Motwani e Raghavan 1995) que resolveram o problema através da contração, onde dois vértices conectados por uma aresta são unidos a partir dos seus respectivos ganhos, que é o valor que os vértices recebem a partir de sua importância para a solução. Sendo que, este processo, é repetido até que o grafo tenha somente dois vértices. O conjunto das arestas finais representará um candidato à corte mínimo. Desta forma, essa solução possui custo O (n^3). Uma técnica bastante importante historicamente foi a de Kernighan-Lin (Schweikert e Kernighan, 1972), explicada em mais detalhes na próxima seção.

3.2.1.1Algoritmo Kernighan-Lin (KL)

Este algoritmo foi apresentado em 1970, onde a partir de uma solução inicial, efetua trocas entre pares de vértices localizados em conjuntos distintos, com complexidade de tempo (n^2logn). Em (Schweikert e Kernighan, 1972), o KL é aplicado na resolução do problema do particionamento de circuitos integrados.

Seja G um grafo não direcionado de n = 2p vértices, associado a uma matriz de custos M simétrica, onde a diagonal é zero para 1 <= i <= n. Desejamos dividir G em dois conjuntos, A e B, de tamanho p, tal que o custo de conexão entre os dois conjuntos seja mínimo. A heurística chamada KL (Kernighan e Lin 1970) recebe os dois conjuntos, A e B, como entrada e tenta encontrar a melhor solução efetuando trocas entre pares de elementos. Existem dois conceitos importantes para entender o algoritmo, o de custo externo e custo interno. Considerando a ϵ A, o custo externo E de um vértice a é definido como:

$$E(a) = X_j 2B_m(a, j)$$

e seu custo interno I como:

$$I(a) = X_j 2A_m(a, j)$$

Portanto, a diferença D entre os custos externos e internos são dados por:

$$D(a) = E(a) - I(a)$$

O mesmo é válido para um vértice b ϵ B. Outro conceito importante é o de um ganho g em se trocar os vértices a e b de conjunto:

$$g(a,b) = D(a) + D(b) - 2m(a,b)$$

Em um primeiro passo, o algoritmo calcula o valor de D para todos os vértices. No segundo passo, são escolhidos dois vértices a e b tal que g(a, b) seja máximo, o que representa o maior ganho possível para uma troca entre os elementos dos conjuntos. Chame a de a01 e b de b01 e retire-os dos conjuntos A e B. Recalcule então, os valores de D para os elementos i ϵ {A − (a)} e j ϵ {B − (b)}, da seguinte forma:

$$D_0(i) = D + 2m(i,a) - 2m(i,j)$$
$$D_0(j) = D + 2m(j,b) - 2m(j,a)$$

Repita o segundo passo do algoritmo para um novo par a02 ϵ {A − (a01)} e b02 ϵ {B − (b01)}.

Novamente separe-os de seus conjuntos originais e recalcule cada D dos elementos restantes dos conjuntos A−(a01, a02) e B −(b01, b02). Repita essa operação até que todos os pares (a01, b01), ... , (a0p, b0p) e seus respectivos ganhos tenham sido identificados.

Considere X = a01, . . . , a0k e Y = b01, . . . , b0k. A diminuição do custo ao trocar os conjuntos X e Y e Pk i=1 gi = C. Se C > 0, uma redução no custo pode ser feita e os conjuntos gerados pelas trocas efetuadas em A e B serão a entrada para uma nova execução do KL. Se C = 0, uma solução para a entrada inicial dada foi encontrada. O algoritmo pode então ser novamente executado para uma nova partição inicial, na tentativa de encontrar um candidato à corte mínimo menor que o já encontrado.

O KL possui complexidade de espaço $O(n^2)$ pois utiliza uma matriz de adjacências para armazenar o grafo G de n vértices. Por essa mesma razão, o passo inicial do algoritmo possui complexidade de tempo (n^2). A busca pelos melhores ganhos na troca de pares custa $(n^2 \log n)$, conforme demonstrado em (Kernighan e Lin 1970), sendo este o custo final da heurística KL.

Diversos trabalhos propuseram melhorias ao KL, por exemplo, Fiduccia e Mattheyses (FM) (Fiduccia e Mattheyses, 82) apresentaram o seu algoritmo que tem uma grande importância científica para este trabalho e será descrito no próximo capitulo.

3.2.1.2 Fiduccia-Mattheyses

O objetivo do algoritmo Fiduccia-Mattheys (FM), como um particionador, é reduzir o chamado corte mínimo (*Min-Cut),* ou seja, diminuir o máximo possível o número de arestas entre as partições como mostra a Figura 3-6.

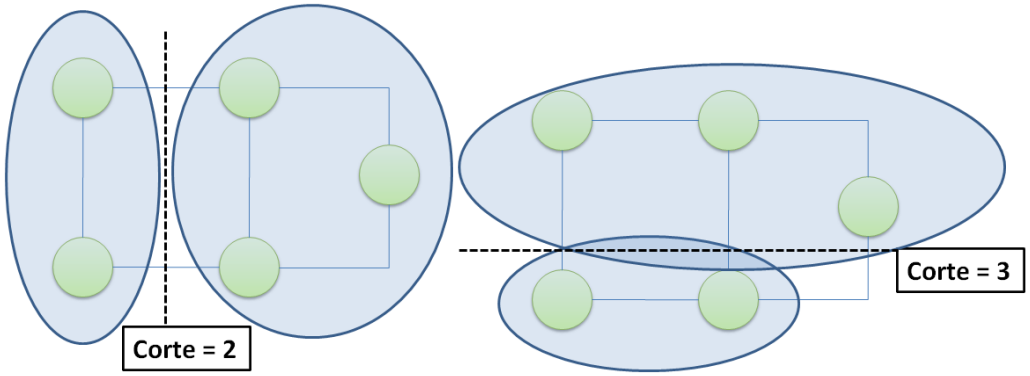


Figura 3-6 Aqui pode ser visto duas formas diferentes de particionar um grafo. Cada uma delas gera um corte (número de aresta que sai de conjunto e chega ao outro).

O algoritmo pode ser brevemente explicado da seguinte forma: Primeiramente se computa o nodo com maior ganho (conceito explicado na sessão sobre a Kernighan-Lin) de todos os vértices do grafo, então se procura um vértice para a troca pela função também dos ganhos de diminuição de arestas. Consequentemente se atualiza o valor de todos os seus nodos vizinhos. Faz-se isso enquanto houver trocas possíveis, após se tentar mover todos os nodos, percorre-se a sequencia de trocas e fixamos a troca que resultou em menor corte. Então, fazemos essa troca permanente e apagamos todas as tentativas posteriores. Portanto, este processo é repetido até que não se obtenha mais ganhos com as trocas.

3.2.1.3 Particionador hMetis

O hMetis (Karypis, 1997) é baseado no que chamamos de particionamento de grafos multinível (*multilevel graph partitioning*). Ele utiliza vários níveis para partição de hipergrafos. Esses níveis seriam divisões em partições de tamanhos diferentes. E existem três fases: a fase de engrossamento, a fase de partição e a fase de refinamento. A fase de engrossamento é repetida várias vezes. A cada iteração um hipergrafo menor é obtido utilizando uma técnica de fusão de vértices. O número de vezes que a fase é executada depende de características de conectividade do grafo, mas pode ser controlado através de um parâmetro de entrada.

Na fase de partição, é permitida a escolha de várias técnicas de bisseção (bisseção espectral, Kernighan-Lin, ou algoritmo de crescimento, entre outros). O objetivo de todas estas técnicas é o de gerar uma bisseção no hipergrafo de tal modo a minimizar o corte de hiperarestas.

A fase de refinamento é repetida o mesmo número de vezes que a fase de engrossamento. A cada vez são desfeitas as junções de vértices feitas durante a fase de engrossamento e são retomadas as hiperarestas que haviam desaparecido, reconstituindo assim, gradualmente, o grafo original. Com a inserção destes vértices e hiperarestas a bisseção deve ser adaptada para que o menor número de hiperarestas seja cortado. Feita esta adaptação, uma nova repetição do refinamento é iniciada. O processo termina quando for obtido um hipergrafo com o mesmo número de vértices e arestas do hipergrafo original.

3.2.2 Posicionador CAPO (baseado em Particionamento)

Neste capítulo será analisado como funciona o posicionador CAPO, também será compreendido como ele resolve o problema de posicionamento. O CAPO é um posicionador que utiliza particionamento para resolver o posicionamento.

A ideia do CAPO é ser um algoritmo *Top-Down,* ou seja, procura decompor o problema de posicionamento em instâncias menores, subdividindo o problema. Dividir para conquistar. Porém esta divisão é eficiente em tempo de execução apenas para pequenos problemas, com o aumento do tamanho do problema de posicionamento temos que ter um acelerador e no caso do CAPO é o particionador de hipergrafo baseado na heurística *Fiduccia-Mattheyses* (FM).

No posicionador CAPO, a divisão nos blocos se faz da seguinte maneira: primeiramente aplica-se a heurística FM para dividir o problema em blocos com mais de 200 células; uma heurística mais simples para dividir em blocos com 35-200 células; e então *branch-and-bound* (Cormen, 2001) para menos de 35 células, ou seja, só quando o problema é realmente pequeno.

No caso, cada parte do hipergrafo particionado vai para um bloco que representa uma parte da área ativa de posicionamento destinada a este bloco. As células no bloco representam os nodos do sub-hipergrafo, as conexões das células são as hiperarestas o peso das células é representado, pela largura total da célula, assim a soma do peso de todas as células deve ser menor ou igual ao peso comportado pelo bloco, para que exista uma solução legalizada para este bloco, caso contrário terá sobreposição, ou como é mais conhecido *Overlap*.

O processo de *Top-Down* pode ser visto como uma sequência de passos, onde é necessário visitar cada bloco de células e decidir se partimos aquele bloco ao meio ou não, e após essa decisão, analisar como vamos distribuir as células no bloco. E decidindo colocar uma determinada célula em um bloco, ela não sairá mais dele. Esse processo de trocas de células difere do posicionador *Dragon*, como veremos melhor no capítulo específico.

3.3 Algoritmo Simulated Anneling

O *Simulated Anneling* (SA), também chamado de arrefecimento simulado, é um algoritmo baseado no processo de cristalização dos metais. Na termodinâmica o processo consiste em aquecer um metal a uma temperatura elevada (temperatura máxima na qual o metal se funde), de tal forma que as moléculas se agitem bastante. Em seguida é diminuída a temperatura do sistema vagarosamente, e com isso as moléculas se organizam no ponto de menor energia e aos poucos vão ordenadamente se ajustando. Ao final deste processo as moléculas estão bem organizadas e ajustadas e como resultado tem-se uma redução dos defeitos do material. Existem muitos *applets* Java disponíveis na Web para melhor compreensão de como o algoritmo soluciona os problemas, por exemplo, no site do *Heaton Research* (*Heaton Research*, 2011).

A principal vantagem deste processo é que ele tende ao resultado ótimo da solução. Isto ocorre, pois o algoritmo faz escolhas aleatórias de qual próxima célula irá mover e principalmente por que ele aceita algumas modificações que a princípio pioram a solução. Essas modificações garantem que no infinito, não ficaremos presos em nenhum mínimo local.

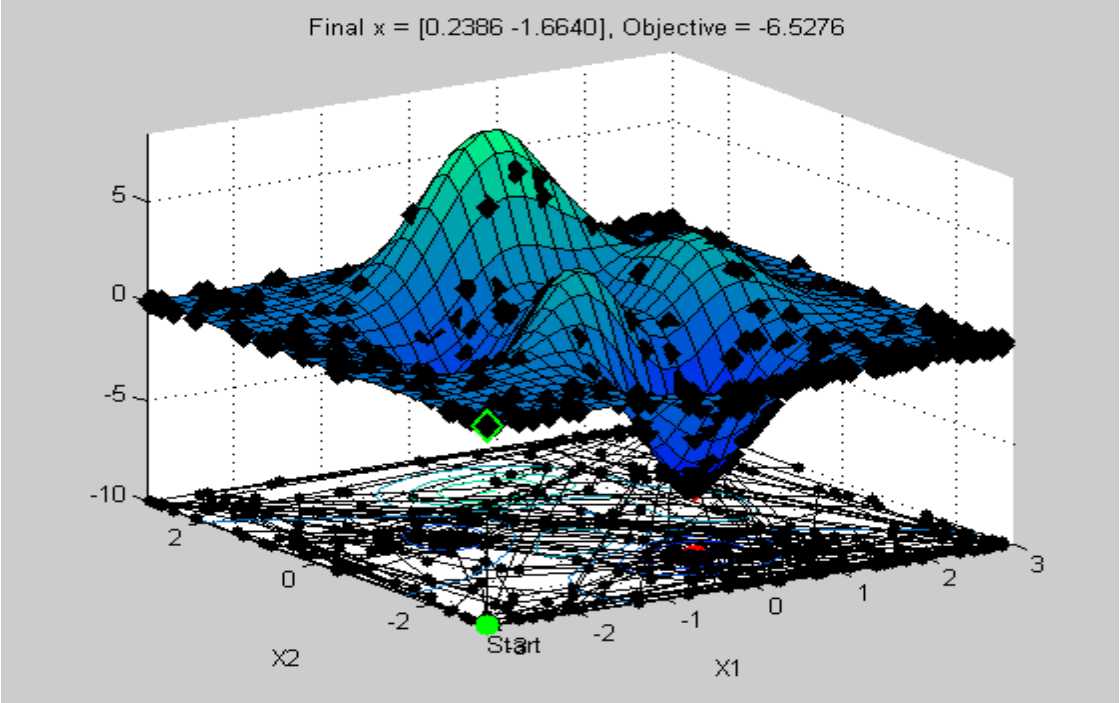


Figura 3-7: Apresenta uma curva de exemplo de um conjunto solução. Sem ligação com nenhum problema específico, apenas para fins didáticos. Gráfico gerado no Matlab.

Na figura 3.7 vemos um gráfico 3D que exemplifica um conjunto solução, onde vemos uma curva mapeada de soluções. Supondo que o gráfico representasse uma curva de soluções de posicionamento, nós desejaríamos reduzir o comprimento de fio, portanto o nosso objetivo seria o ponto da curva mais próximo da base (o ponto que está em vermelho). No gráfico temos um ponto de mínimo local (em verde), que significa um valor que localmente é o menor, porém analisando todo o conjunto solução, vê-se que existe outro ponto que seria uma opção melhor.

Haja vista que é impossível fazer esse mapeamento visual para um problema de posicionamento devido ao tamanho que geraria o conjunto solução e os vários eixos que seriam necessários.

O algoritmo é demonstrado no quadro:

```
1: temp = temperaturaInicial;
2: place = posicionamentoInicial;
3: // Loop Externo
4: while ( condição para continuar o loop externo )
5:     // Loop Interno
6:     while ( condição para continuar o loop interno )
7:         novoPosicionamento = Perturbacao(posicionamento);
```

8: delta = *Custo*(*novoPosicionamento*) − *Custo*(*posicionamento*);
9: **if** (*Aceita*(delta, *temp*)) **then**
10: *posicionamento* = *novoPosicionamento*;
11: **end if**
12: **end while**
13: *temp* = *Historico*(*temp*)
14: **end while**
15: **return *posicionamento***

Um ponto muito importante é a variação do custo, aqui chamado de delta(Δ) e reflete como o algoritmo varia com o tempo.

Se $\Delta < 0$: Aconteceu uma diminuição da energia, o que significa que a nova solução é melhor que a anterior. O método aceita a nova solução.

Se $\Delta \geq 0$: Aconteceu um aumento de energia. A aceitação ou não depende das temperaturas. Quanto maior a temperatura maior será a chance de esta solução ser aceita. Para calcular a probabilidade de se aceitar a nova solução, é utilizada uma função conhecida por fator de Boltzmann, que é dada por:

$e^{\frac{-\nabla}{T}}$, onde T é um parâmetro do método, chamado de temperatura.

O procedimento é finalizado quando a temperatura chega a um valor baixo e próximo de zero. Então nenhuma solução que piore o valor da solução será mais aceita.

A utilização do SA já foi um grande tópico de pesquisa pois a estrutura do algoritmo é bem simples já que é baseado em randomização, que significa movimentar os elementos aleatoriamente. Tem a seu favor também o potencial de fornecer soluções bastante próximas da ótima. O principal problema é que, em geral, para problemas grandes o SA torna-se lento e hoje em dia o tempo de solução do posicionamento é cada vez mais importante devido ao crescimento do tamanho dos circuitos e da complexidade deles. Se a análise de tempo de resolução do problema não for levada em consideração, uma solução pode levar mais tempo do que o previsto para conseguir a solução do posicionamento, inclusive um algoritmo lento facilmente pode levar anos (literalmente) para solucionar um grande problema.

3.3.1 Posicionador *Dragon* (baseado em S.A.)

A base teórica do posicionador *Dragon* (Karypis, 1997) é o mais simples dos três analisados. A estrutura do *Dragon* é baseada em particionamento, como o posicionador CAPO, porém o *Dragon* utiliza o particionador hMetis (Karypis, 1997), que também já foi citado anteriormente, como à parte de posicionamento global.

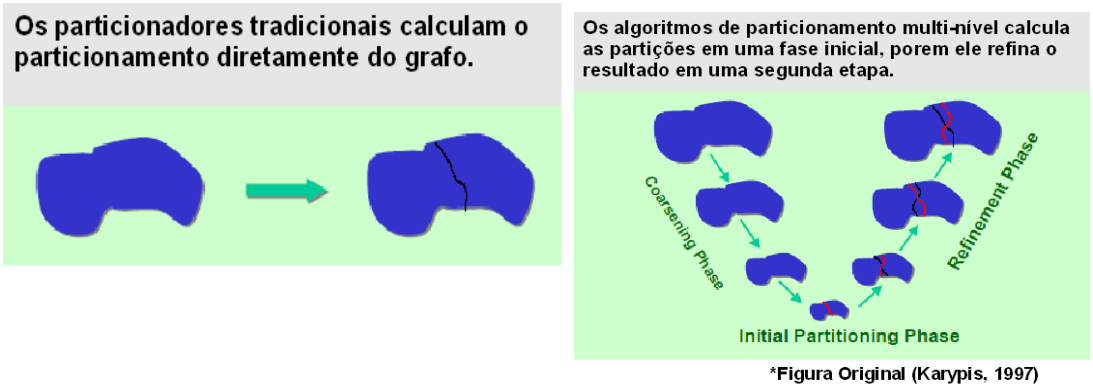


Figura 3-8: Nesta figura retirada do artigo do hMetis, vemos sob o ponto de vista de Karypis a diferença de um posicionador tradicional e um particionamento multinível (Karypis, 1997).

É nesta parte que as células vão definir qual partição, ou seja, qual parte do circuito eles estarão até o final do posicionamento. E na parte do posicionamento detalhado, ele visita cada partição de células e resolve, para cada uma, um Simulated Annealing (ver Simulated Anneling) em cada bloco de células. Porém no *Dragon* a troca de células entre duas partições vizinhas é permitida, ou seja, o que o hMetis estipula não é tratado como lei, mas sim uma boa indicação de qual seria o local mais adequando para aquela célula. Segundo o artigo do CAPO (Caldwell 2000), quando se utiliza um particionador melhor (no caso, o baseado na heurística Fiduccia-Mattheyses) é menos necessário trocar células entre as partições vizinhas em busca de uma melhor qualidade de posicionamento, que como visto anteriormente, o CAPO realmente não troca mais a células de suas partições iniciais.

Como o *Dragon* utiliza particionamento também, alguns autores preferem dizer que ele se encaixa no grupo dos particionadores e outros dizem que ele é baseado em Simulated Annealing. Pressuponho que o mais correto seria dizer que ele é um posicionador híbrido, mas em termos de resolução do problema, dentro das questões especificas deste trabalho, ele tem a serventia de demonstrar a transparência da técnica *Critical Star* e que o algoritmo SA também gera uma boa qualidade de posicionamento.

4 TÉCNICA DA CRITICAL STAR

A *Critical Star* é uma técnica de aumento do desempenho médio do circuito, que tem por objetivo fazer com que o posicionador fique atento a desempenho e reduza os caminhos críticos de forma a conseguir um circuito de melhor qualidade sem retrabalho.

Para isso a ideia que guia a técnica é a de aproximarmos as células de um caminho critico. Diminuindo a distância média entre elas e com isso o roteamento também terá uma proximidade maior dos pinos que participam do caminho crítico a serem interligados. Então haverá uma provável diminuição no tamanho das interconexões críticas. Tendo em vista a importância do tamanho das interconexões no atraso dos circuitos atuais (Bakoglu, 1990 e Sylvester, 1998).

A técnica *Critical Star* melhora o desempenho de circuitos VLSI de forma rápida, fácil de programar e eficiente. A técnica tem como princípio a alteração da estrutura da rede de células (*netlist*) e com isso funciona para qualquer posicionador, pois ela não é um algoritmo e sim um modo diferente de o posicionador "ver" o problema.

No estado da arte existem algoritmos com outras abordagens. Por exemplo, o *Geometric-Based Analysis* (Tao, 2008) e com programação linear (Andrew, 2008), para isso utilizam uma solução inteira de posicionamento e iterações enquanto a *Critical Star* cria uma facilidade na implementação. Não é necessário fazer um programa específico para resolver o problema de desempenho, o próprio posicionador já o resolve. Além da possibilidade de qualquer técnica dirigida a desempenho pode ser utilizada em conjunto com a *Critical Star*, visto que é apenas uma alteração no grafo do circuito. Ainda outra vantagem da técnica, devido a sua natureza, é a possibilidade de utilização para outros circuitos com blocos de tamanhos diferentes (*mixed-size*), pois para a *Critical Star* o tratamento é transparente durante a resolução do posicionamento, apesar da inserção da técnica recorrer ao sistema de ajuste da *netlist* para a modificação, no posicionador não muda nada.

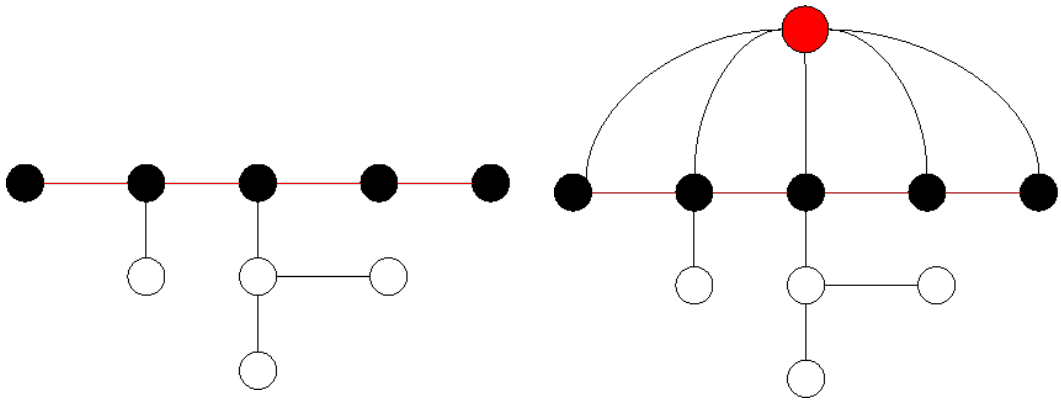
Ao pensar a maneira diferente de vermos o problema, temos de levar em consideração o seguinte: existem muitas estruturas de hipergrafos que representam o mesmo problema em posicionamento. Por exemplo, quando temos duas células (por exemplo, *Standard Cells*) do mesmo tamanho ou quando temos pesos diferentes conectando duas células iguais, em termos elétricos, temos a mesma entrada, porém abstratamente em termos matemáticos e físicos, são abordagens completamente diferentes. Ou seja, o problema real que resolvemos é o mesmo, porém abstratamente, dentro do posicionador, as coisas são bem diferentes. Estas diferenças é que trazem à possibilidade de ajustes na estrutura e possibilidades de melhoria na comparação de

como um dado posicionador reage a cada tipo de entrada. A modelagem do circuito aplicada para cada posicionador pode modificar tanto a forma como o posicionador resolve o problema, assim como, a solução final.

4.1 O Conceito da Critical Star

Esta técnica depende da *netlist* de entrada do circuito e do conhecimento prévio dos caminhos críticos, ou seja, ela não calcula os caminhos críticos (utiliza a ferramenta comercial Leonardo Spectrum, da Mentor). Sendo assim é necessária uma prévia análise do grafo do circuito dizendo quais são os caminhos que ela deve cuidar. Em geral, as ferramentas de analise de desempenho (*timing*) necessitam de um atraso máximo que o projetista pretende que aquele circuito funcione, este valor tem influência direta nos fatores de área e consumo do circuito. Caso no circuito exista apenas uma rede de clock, que é a rede de fios que alimenta os flip-flops e com isso gera a capacidade de memória nos circuitos, ela deverá ter o atraso máximo indicado no analisador de desempenho, mas ainda dentro deste ponto teríamos que levar em consideração algumas variáveis que não são o escopo deste trabalho, como por exemplo, o atraso da própria rede de clock.

O conceito consiste em adicionar novas redes globais e artificiais na *netlist* original, formando uma *netlist* nova e assim agregando conhecimento para o posicionador. Esse conhecimento é adicionado em forma de uma *Critical Star*, que é um nodo virtual (não ocupa área ativa) e é encaixado no grafo de forma a agrupar as células que fazem parte de um caminho crítico que desejamos reduzir. A inserção acontece como apresentado na Figura 4-1:



(a) Um grafo de um posicionamento, onde em preto está um caminho crítico.

(b) A *Critical Star* atuando sobre este grafo. A técnica simplesmente insere um nodo virtual e conecta todos os nodos do grafo.

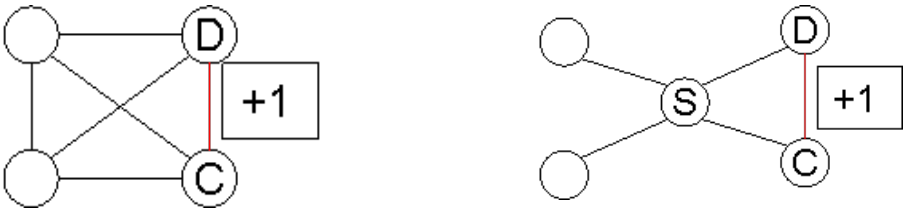
Figura 4-1: Inserção da *Critical Star* em uma sequência de nodos do grafo

Para esta conexão colocamos o dobro do peso (a maioria das conexões entre dois nodos tem peso um) e esta modificação no valor foi obtida empiricamente, ou seja, efetuamos centenas de experimentos utilizando posicionadores que utilizam várias estratégias diferentes e utilizamos os *benchmarks* padrões para o estudo de posicionamento.

A ideia principal da técnica é inserir um nodo virtual que segura todos os nodos do grafo que participam do caminho critico. Uma ideia parecida já foi relatada (Kahng. 2001), onde o trabalho existente aplica para um posicionador global e quadrático uma força que "amarra" os flip-flops de um caminho crítico. Desta forma ele modifica apenas o sistema linear na solução de um posicionador analítico enquanto no nosso trabalho propomos uma modificação que influi em todo o processo do posicionamento (fase global e detalhada). Outra questão é que ao focarem-se apenas nos flip-flops as outras células do caminho crítico estão "soltas" e podem levar a um resultado sem ganho de desempenho. Isto pode ocorrer tanto pelas células estarem "soltas" como também é possível que o posicionamento detalhado não se importe com a aproximação ocorrida no posicionamento global e simplesmente desfaça a melhoria. Esse problema não ocorre na *Critical Star*, pois a técnica influência tanto o posicionamento global quanto o detalhado e ainda mantém todas as células do caminho que desejamos reduzir conectadas.

As forças de conexão com a *Critical Star* indicam para o posicionador a existência dos caminhos críticos. Com isso o posicionador tem uma visão mais completa e global do problema. E assim podemos adicionar a *Critical Star* para todos os caminhos críticos desejados, sempre tendo em vista que se adicionarmos para um número excessivo de caminhos poderemos ter um impacto de desempenho significativo.

A *Critical Star* consiste em também adicionar pesos nas arestas, pois o nodo virtual sozinho não cria resultados tão bons. Na Figura 4-2 analisaremos como fica a adição de forças no grafo de cada um dos 2 modelos que estamos trabalhando aqui, Clique e Estrela.



(a) Adição de força no modelo Clique. Entre o *driver* e o nodo que participa do caminho crítico.

(b) Adição de força no modelo Estrela. Entre o *driver* e o nodo participante do caminho critico

Figura 4-2: Como funciona a adição de forças nos sistemas de nets.

4.2 Metodologia de Teste da Critical Star

As técnicas deste trabalho foram testadas utilizando os *benchmarks* (ISCAS`99). As informações de atraso foram geradas em uma ferramenta comercial chamada *Leonardo* da *Mentor Graphics* a partir do arquivos VHDL e mapeada para o *netlist* em nível lógico. Desta forma análise de *timing* é independente de qualquer posicionamento existente.

O teste foi realizado com números variados de caminhos críticos: 5, 10, 15, 20, 25, 50, 75 e 100. Desta forma foi possível traçar um gráfico de comportamento partindo do pressuposto que os posicionadores têm certa estabilidade de resultados.

4.3 Inserção da Critical Star no FastPlace3 e no Z-Place

Para utilizar a técnica nos posicionadores quadráticos, como o FastPlace3 (Viswanathan, 06), o Z-Place ou o PlaceDL, o tratamento das modificações na *netlist* é igual e transparente para o posicionador. As modificações foram desenvolvidas com base nas alterações da matriz de conectividade que é montada pelos posicionadores quadráticos.

Neste ponto explanaremos quais alterações devem ser efetuadas na *netlist*. A técnica é bem simples e funciona em duas etapas de alterações:

1. A primeira é a inserção de uma nova célula virtual que conectará todo o caminho crítico. Dentro da matriz de conectividade ficará uma rede exatamente igual ao caminho crítico, desde que o caminho tenha mais de 3 células, pois como o modelo de redes do FastPlace, Z-Place e PlaceDL transforma todas as redes acima de 3 nodos em modelo Estrela. Assim ficará um nodo virtual conectado a todas as células do caminho, mantendo-as com uma tendência de proximidade mutua.
2. A segunda é a adição de pesos nas conexões das células que fazem parte do caminho crítico. Então adicionamos a *netlist* n-1 redes de duas células, tal que n é igual a todas as células do caminho critico, desta forma estamos aumentando o peso das conexões do caminho crítico, em geral, para o dobro. As *n-1* redes são o número de arestas necessárias para conectar os n nodos.

A influência da técnica ocorre distintamente nas duas fases de posicionamento. Na fase do posicionamento global temos a alteração da matriz de conectividade que é resolvida no sistema linear e assim já na solução deste, temos uma alteração de como ficará a disposição das células. A outra mudança ocorre na fase local com a alteração da função custo que calcula o comprimento do fio total, no caso, temos uma alteração no calculo do Semiperímetro. Por exemplo, ao calcular o comprimento do fio de duas células, *A* e *B* que fazem parte do caminho crítico, ao invés de contar no custo o tamanho do fio de distância *d1*, o valor será, geralmente, à distância *d1* vezes 2 (adição do peso) mais uma distância *d2* da célula *A* até o nodo Virtual *V* mais a distância *d3* da célula *B* até o nodo Virtual *V*.

$$CS = 2 * d1 + d2 + d3 \qquad 4.1$$

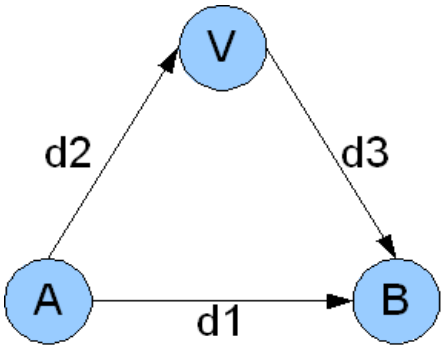


Figura 4-3: Auxilia a explicação da técnica da *Critical Star*

4.4 Resultados no Z-Place

A tabela 4.1 mostra os resultados da técnica *Critical Star* sobre o Z-Place na otimização de caminhos críticos comparada com os resultados do Z-Place sem cuidados com os caminhos críticos. Também a tabela de adição de forças extras no caminho critico, sem a *Critical Star*. A *Critical Star* é melhor do que as duas técnicas anteriores aplicadas individualmente para a redução do comprimento de fio e comprimento de fio crítico, sob a análise dos resultados obtidos.

Tabela 4-1: Atuação do *Critical Star* no Z-Place. O comprimento dos fios e os respectivos ganhos de comprimento de fio e comprimento de fio críticos.

Benchs	Baseline		Adição de Forças		Critical Star 2D	
	WL	CWL	WL	CWL	WL	CWL
b01	1801	638	1895	578	1818	606
b02	6485	1202	5983	849	6103	886
b03	2300	880	2323	927	2447	854
b04	2365	965	2244	806	2349	749
b05	2884	1463	2944	1218	2822	1380
b06	10249	1320	10422	1144	10195	1260
b07	20025	1557	19342	1491	17693	906
b08	3576	742	3536	820	3693	655
b09	97253	3168	97412	2893	90268	2384
b10	236910	3661	241478	3593	234539	3339
b11	380949	4996	374604	4911	372894	4667
b12	99728	3035	96105	2833	96123	2858
Total WL	864525	23627	858288	22063	840944	20544
%	100.00%	100.00%	99.28%	93.38%	97.27%	86.95%

4.5 Resultados no FastPlace3

Na tabela 4.2 observamos os resultados da técnica frente ao posicionador FastPlace3 (Viswanathan, 2007). No gráfico da Figura 4-4 podemos observar que há uma estabilidade no número de caminhos críticos necessários para que alcancemos o melhor valor entre comprimento de fio e comprimento de fio critico. Esta estabilidade só é afetada gravemente no *benchmark* b13, onde provavelmente temos um grau de liberdade maior no conjunto solução do posicionamento.

Tabela 4-2: Resultado da técnica do posicionamento com e sem o *Critical Star* 2D sobre o posicionador FastPlace 3 (Viswanathan 2007). Da esquerda para a direita temos as medições de comprimento de fio (WL) e de comprimento de fio crítico (CWL) do próprio FastPlace. À direita temos em resultados com a utilização da técnica em números absolutos e em percentuais. E finalmente o número de caminhos testado que representou a melhor configuração.

FastPlace 3	Origi						
Circuit	WL	CWL	WL	ΔWL %	CWL	ΔCWL %	N° of Pat
b04a	7761	1642	7844	1.07	1195	-27.22	10
b09a	2048	767	2057	0.44	474	-38.20	10
b10a	2435	1095	2440	0.21	867	-20.82	20
b11a	9402	1125	9669	2.84	918	-18.40	5
b13a	3394	696	3434	1.18	388	-44.25	75
b14a	89425	2769	87355	-2.31	2258	-18.45	5
b14_1a	88128	2327	89927	2.04	1641	-29.48	25
b15a	215433	1706	218392	1.37	1688	-1.06	15
b15_1a	209740	1312	214346	2.20	1044	-20.43	10
b17a	666066	3014	671428	0.81	2054	-31.85	15
b17_1a	647882	3499	653670	0.89	1856	-46.96	15
			Avg	0.98	Avg	-27.01	

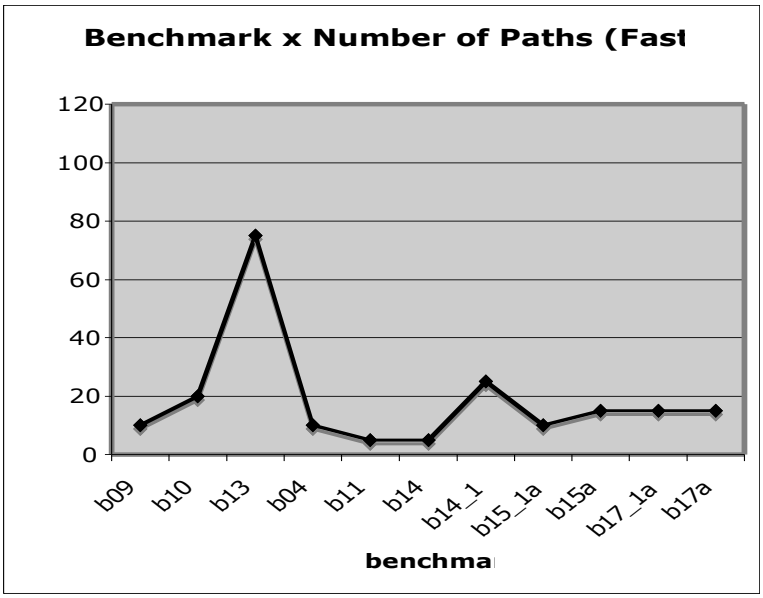


Figura 4-4: Mostra a distribuição dos melhores *trade-off* (comprimento de fio total e comprimento de fio crítico) na relação do número de células (tamanho do benchmark) e o melhor número de caminhos críticos a serem utilizados.

4.5.1 Conclusões sobre Congestionamento no Posicionador FastPlace3 com a Critical Star

Este subtópico visa apresentar um breve estudo sobre o congestionamento dos posicionamentos gerados pelas ferramentas FastPlace 3 com a inclusão da técnica *Critical Star*. Incorporamos este estudo, pois um dos pontos mais discutidos atualmente com relação ao posicionamento é que não adiantaria o posicionador gerar um resultado com menor comprimento de fio se aquele resultado é impossível de rotear. E isso é

coerente, tendo em vista que posicionamento é uma parte da engrenagem que gera um circuito digital. Porém qualquer técnica que diminua a liberdade de posicionamento tende a aumentar o congestionamento, pois o posicionador terá menos opções de organização e a tendência de aumento do congestionamento máximo se comprova na tabelas 4.3.

Mas o congestionamento de um circuito digital tem outras influências críticas, como por exemplo, os problemas térmicos dos circuitos. Que significa que onde há muitos fios há muito calor e isto também pode inviabilizar a produção do circuito.

Então utilizamos a ferramenta *CongestionMaps Plotter* (Adya, 2005) para comparar os níveis de congestionamento no circuito resultante com a aplicação ou não da *Critical Star*. Na tabela 4.3 temos o resultado que na média das áreas de posicionamento o congestionamento aumentou 0.91% e o valor de pico aumentou 6.74%, esses valores indicam um aumento não muito significativo. Mas deve ser considerado que no aumento do valor do congestionamento de pico dificultaria o roteamento e em alguns casos poderia só funcionar sem a utilização da *Critical Star*.

Tabela 4-3: Mostra uma comparação do congestionamento dos circuitos apresentados, na utilização do posicionador FastPlace 3 com e sem a técnica proposta. As tabelas são divididas em normal (FastPlace original) e Alterado (FastPlace com a *Critical Star*). E subdividido em mediana, média e pico. E à direita o número de caminho críticos que utilizamos.

FP	Normal			Alterado						
benchs	**median**	**average**	**peak**	**median**	**average**	**peak**	**%median**	**%average**	**%peak**	No CPs
b14_1a	19,3263	19,648	66,4844	17,5108	20,1577	73,2508	-9,39393	2,594157	10,17743	25
b14a	19,3167	20,7917	68,5911	17,5537	20,3711	70,6916	-9,12682	-2,02292	3,062351	5
b15_1a	21,2392	21,6703	73,1772	21,364	21,9726	74,1437	0,587593	1,394997	1,320767	10
b15a	20,3225	21,6525	65,6679	21,0357	21,8801	73,495	3,509411	1,051149	11,91922	15
b17_1a	19,4417	21,8827	84,3222	18,9594	22,0859	102,326	-2,48075	0,928587	21,3512	15
b17a	19,8733	22,4669	101,874	20,1187	22,817	94,3657	1,234823	1,558292	-7,37018	15
						avg	-2,61161	0,917377	6,743463	

4.6 Reflexão sobre a Complexidade Aproximada da Critical Star nos Posicionadores Quadráticos

Neste capitulo faremos uma análise sobre o tempo de acréscimo ao se utilizar a técnica *Critical Star*. O valor não é exato, pois modifica a forma como é construída a estrutura da matriz que será utilizada para resolver o posicionamento. A técnica atua na resolução do sistema linear e como a complexidade é regida pelo número de não zeros da matriz é difícil de medir a complexidade real tendo em vista o ajuste ao modelo de redes (Clique e Estrela) e mais, o pior caso seria muito pessimista, porém vamos tentar mostrar os pontos para uma possível análise e junto com os resultados mostraremos o tempo na análise empírica.

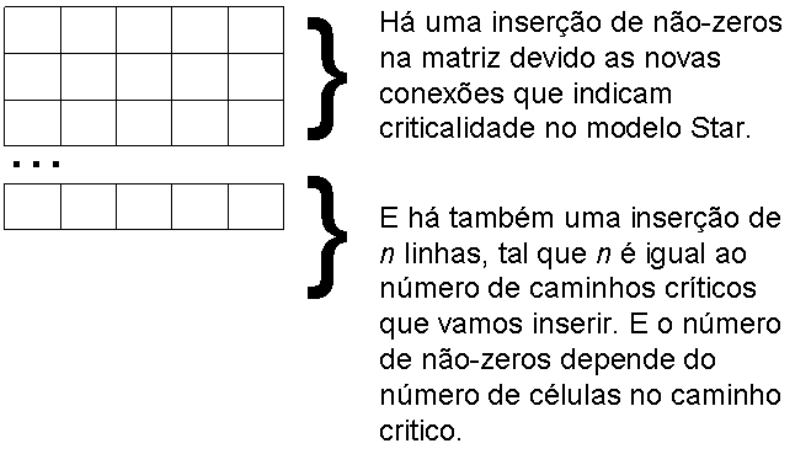


Figura 4-5: As inserções na matriz de admitância do sistema linear.

Tendo em vista que a maioria das conexões é de 2 e 3 nodos, temos mais modelos Clique do que Estrela. O modelo Clique não tem aumento de complexidade no posicionamento, pois a aresta já existe e ela é apenas valorada diferentemente.

Outro fator que aumenta o tempo é que: com o aumento de conexões existentes o recalculo do comprimento de fio será mais demorado, mas como o número de caminhos críticos não é tão grande e o valor é proporcional ao aumento do número de fios esse valor não influenciou muito nos resultados.

4.7 Inserção da Critical Star no Posicionador CAPO

Apesar de termos o código-fonte do CAPO e podermos fazer a inserção da *Critical Star* desta forma, preferimos não adicionar a técnica no CAPO pelo código-fonte. Adicionamos pela mudança na *netlist*, ou seja, mudamos o hipergrafo da entrada sem modificar o solucionador para manter o padrão de inserção da técnica da mesma forma transparente dos outros posicionadores. A capacidade de inserção de forma transparente é um ponto positivo da técnica e por outra análise a comparação entre os resultados será justa.

Então foi inserido um nodo virtual de tamanho zero, ou seja, um nodo que não ocupa área ativa no posicionamento. Um nodo virtual para cada caminho crítico. Para isso programamos um *parser* de ajuste da *netlist* original para a nova *netlist*. Aumento do peso das conexões já existentes se dá com a simples criação e uma rede nova, com as duas células que queremos aumentar o peso.

O CAPO entende estas mudanças como alterações no modo que ele calcula o comprimento total dos fios na solução do posicionamento, alterando assim as divisões das células dentro dos blocos e gerando assim prioridade para as células que participam dos caminhos críticos.

4.8 Resultados no Posicionador CAPO

No CAPO obtivemos excelentes resultados com um aumento de apenas 0.65% no comprimento total do fio e uma diminuição de 29.16% de comprimento de fio

crítico, como podem observar na tabela 6.1. Outro ponto a ser notado é que para termos o comprimento de fio real devemos utilizar a *netlist* original do circuito.

Tabela 4-4: Resultados da técnica inserida no CAPO. Comprimento de fio total e Comprimento de fio crítico. Da esquerda para a direita temos as medições de comprimento de fio (WL) e de comprimento de fio crítico (CWL) do próprio CAPO. À direita temos em resultados com a utilização da técnica em números absolutos e em percentuais. E finalmente o número de caminhos testado que representou a melhor configuração.

CAPO	Origina						
Circuit	WL	CWL	WL	ΔWL %	CWL	ΔCWL %	N° of Pat
b04a	7013	1258	7278	3.78	1185	-5.80	10
b09a	1804	694	1842	2.11	585	-15.71	5
b10a	2478	1204	2592	4.60	1148	-4.65	5
b11a	9590	1241	9615	0.26	862	-30.54	10
b13a	3452	778	3189	-7.62	610	-21.59	5
b14a	85364	2331	83963	-1.64	1957	-16.04	5
b14_1a	74853	1933	78277	4.57	1657	-14.28	5
b15a	203445	1915	204265	0.40	1351	-29.45	25
b15_1a	208116	1485	205758	-1.13	395	-73.40	100
b17a	654809	2838	659973	0.79	1622	-42.85	50
b17_1a	652470	3758	659282	1.04	1262	-66.42	50
			Avg	0.65	Avg	-29.16	

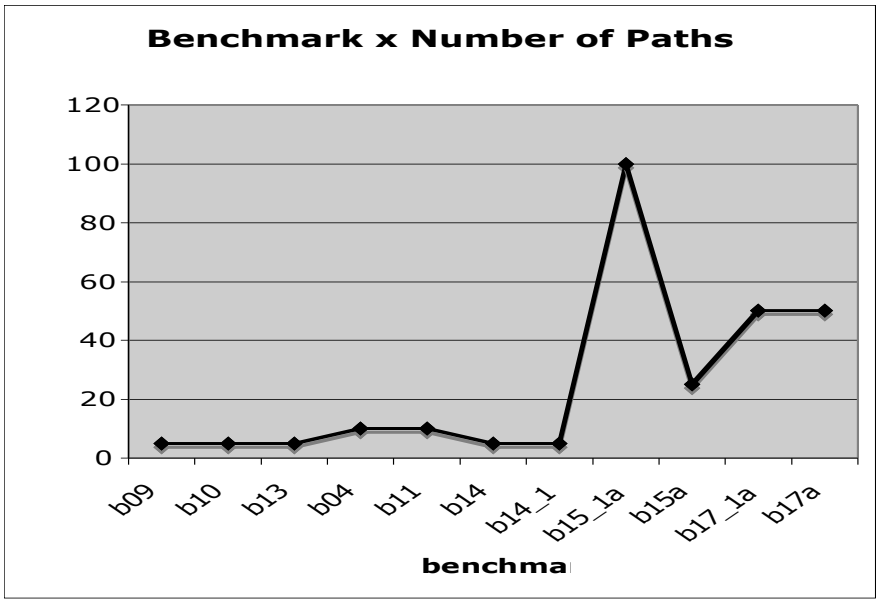


Figura 4-6: Mostra a distribuição dos melhores *trade-off* (comprimento de fio total e comprimento de fio crítico) na relação do número de células (tamanho do Benchmark) e o melhor número de caminhos críticos a serem utilizados.

Uma ideia para utilizar a técnica da melhor forma possível seria saber de antemão qual o número de caminhos críticos que são mais adequados para um determinado circuito. Esta informação é complicada de se definir, mesmo sabendo-se que estamos utilizando um posicionador previamente escolhido. Mas um ponto positivo que se retira

da tabela 4.4 é que este valor no caso do CAPO tende a crescer com o aumento do número de células no circuito, ou seja, à medida que aumenta o número de células do circuito, aumenta também a capacidade do CAPO com a *Critical Star* de tratá-los. Isto demonstra um ponto positivo do posicionador.

Vemos que o conjunto *Critical Star* acoplado no CAPO foi muito bom e isso significa que o CAPO tem uma grande capacidade de lidar com pequenas modificações na *netlist*, não sendo tão sensível. Ele conseguiu efetuar as melhorias sem perder seu foco de comprimento de fio global. Na **Tabela 4-5** vemos uma comparação entre os comprimentos de fio totais e os comprimentos de fio críticos dos posicionadores CAPO e FastPlace 3. Aqui podemos notar que o comprimento de fio crítico foi sempre menor no FastPlace 3, o que denota uma melhor adaptação com a *Critical Star*.

Tabela 4-5: Mostra uma comparação entre o posicionador CAPO e o FastPlace3 onde temos o melhor *trade-off* entre comprimento de fio total e comprimento de fio crítico.

	CAPO	FastPlace 3		CAPO	FastPlace 3	
Circuit	WL	WL	ΔWL %	CWL	CWL	ΔCWL %
b04a	7278	7844	7.78	1258	1195	-5.01
b09a	1842	2057	11.67	694	474	-31.70
b10a	2592	2440	-5.86	1204	867	-27.99
b11a	9615	9669	0.56	1241	918	-26.03
b13a	3189	3434	7.68	778	388	-50.13
b14a	83963	87355	4.04	2331	2258	-3.13
b14_1a	78277	89927	14.88	1933	1641	-15.11
b15a	204265	218392	6.92	1915	1688	-11.85
b15_1a	205758	214346	4.17	1485	1044	-29.70
b17a	659973	671428	1.74	2838	2054	-27.63
b17_1a	657866	653670	-0.64	3758	1856	-50.61

4.8.1 Conclusões em Congestionamento no CAPO com a Critical Star

Este subtópico tem uma grande relação com o capítulo da análise do congestionamento no FastPlace. Neste capitulo será exposto os resultados do posicionador CAPO com a técnica da *Critical Star.*

Tabela 4-6: Mostra uma comparação do congestionamento dos circuitos apresentados, na utilização do posicionador CAPO com e sem a técnica proposta.

CAPO	Normal			Alterado						
benchs	median	average	peak	median	average	peak	%median	%average	%peak	No CPs
b14_1a	17,5765	17,1797	51,7891	17,7878	17,8962	54,6218	1,202173	4,17062	5,469684	5
b14a	19,3069	20,3037	68,3135	20,0916	20,1292	59,3884	4,06435	-0,85945	-13,0649	5
b15_1a	22,8825	21,6159	65,1206	22,3288	21,319	63,9471	-2,41975	-1,37353	-1,80204	100
b15a	20,8961	20,6311	66,5693	22,5223	20,7626	56,4493	7,782313	0,637387	-15,2022	25
b17_1a	22,6614	22,4856	86,6866	23,7097	22,6437	81,478	4,625928	0,703117	-6,00854	50
b17a	22,4385	22,5384	84,1032	23,4214	22,6737	90,8098	4,380418	0,600309	7,974251	50
						avg	3,272572	0,64641	-3,77229	

No CAPO tivemos um resultado um pouco diferente com relação ao FastPlace3, tivemos um aumento de 0.64% no congestionamento médio, o que demonstra que aumentará um pouco a dificuldade de roteamento, mas tendo em vista que o valor é baixo, a influência é baixa. Já no congestionamento máximo, ou seja, na área do circuito que teríamos uma maior dificuldade de roteamento, tivemos uma diminuição de -3.77%, o que significa que teríamos uma facilidade maior de resolver a parte mais crítica.

Analisando o resultado do CAPO com o do FastPlace3, notamos que a técnica se adequou melhor ao CAPO, pois inclusive melhorou a distribuição das células de forma a facilitar na maioria dos casos o roteamento. Sob este ponto de vista a inserção da técnica no CAPO daria vantagens ao posicionador, pois além de estar atento a desempenho ainda facilitaria o trabalho do roteador.

4.9 Inserção da Critical Star no *Dragon*

A inserção da Critical Star no *Dragon* foi feita da seguinte forma: foi adicionado uma célula virtual com altura da banda e largura 0, ou seja, a área da célula é igual à zero, para que o valor do nodo virtual não afete no posicionamento. E assim conseguimos modificar a *netlist* do *Dragon* internamente. Com isso implementamos, pela *netlist*, a *Critical Star*, e a técnica de aumentar o peso das conexões já existentes é igual ao caso do CAPO, onde apenas criamos e uma rede nova, com as duas células que queremos aumentar o peso (adição de peso). O *Dragon* altera o modo que calcula o comprimento total dos fios na solução do posicionamento, alterando assim as divisões das células dentro dos blocos e gerando assim prioridade para as células que participam dos caminhos críticos.

4.10 Resultados no *Dragon*

Tabela 4-7: Mostra uma comparação entre o posicionador *Dragon* e o FastPlace3 onde temos o melhor trade-off entre comprimento de fio total e comprimento de fio crítico.

	Dragon Original		Dragon Alterado 100			
Circuit	**WL Dragon**	**CWL Dragon**	**WL**	**CWL**	**ΔWL %**	**ΔCWL %**
b01/fd	531	465	655	500	23,352	7,527
b02/fast	182	144	214	123	17,582	-14,583
b04	7595	1435	8730	851	14,944	-40,697
b07	5143	755	6533	470	27,027	-37,748
b08	2268	937	2922	549	28,836	-41,409
b09	1998	747	2741	419	37,187	-43,909
b10/fast	2413	1195	3434	716	42,312	-40,084
b11	9366	1286	11416	542	21,888	-57,854
b13	3374	824	3924	449	16,301	-45,510
b14	87494	2237	96479	902	10,269	-59,678
b14_1	76590	2357	86135	739	12,462	-68,647
b15	191794	1706	208020	502	8,460	-70,574
b15_1	191144	1001	211957	410	10,889	-59,041
b17	647290	3270	654830	969	1,165	-70,367
b17_1	653179	3882	672605	973	2,974	-74,936
				média	18,377	-47,834

Como visto na **Tabela 4-7**, no *Dragon* tiveram os piores resultados dentre todos os posicionadores testados, o que mais contou para este resultado foi os *benchmarks* pequenos que mostram um aumento muito grande de comprimento de fio o que afeta demais a média. Esse resultado demonstra que o *Dragon* não conseguiu lidar tão bem quanto os outros posicionadores com a *Critical Sta*r. Mas observamos que houve uma redução média de quase 48% dos caminhos críticos, que significa metade do comprimento de fio. Neste caso o mais interessante seria trabalhar com o meio termo, um algoritmo que reduzisse menos o comprimento dos fios nos caminhos críticos e não piorasse tanto o comprimento de fio, algo que a *Critical Star* não conseguiu fazer com o *Dragon*.

Um ponto que merece reflexão é que com estes resultados conclui-se que o *Dragon* seja o posicionador mais sensível à alteração na *netlist*, o que significa que é o

posicionador que mais leva em consideração pequenas alterações no grafo do posicionamento, alterando assim de forma contundente o resultado final.

5 INTRODUÇÃO A LOGICAL CORE

Este capítulo analisará como a *netlist* pode influenciar o posicionamento. Também será discutida uma análise de como o Semiperímetro visualiza a *netlist* e um problema que foi importante para a evolução que obtivemos na técnica da *Logical Core I*.

Primeiramente uma análise sobre o trabalho de Liu (2004) que fez importantes conclusões que foram ponto inicial para as ideias desenvolvidas nesta dissertação.

A primeira conclusão é que posicionadores com características estruturais diferentes reagem diferentemente frente a características semelhantes dos benchmarks. Isso significa que se encontrarmos uma *netlist* com muitas redes globais, redes que afetam uma porcentagem grande das células do circuito. Se o posicionamento for fácil para um posicionador que é baseado em um algoritmo específico, não necessariamente será fácil para todos os posicionadores. A eficiência do problema de posicionamento esta, intrinsecamente, correlacionado com a complexidade das interconexões dos circuitos. Ou seja, quanto mais complexo for à organização da *netlist*, maior será a dificuldade de termos eficiência.

Redes globais aumentam o esforço para eliminar os overlaps das células e pioram a qualidade do posicionamento dos posicionadores analíticos. Como trabalho futuro eles propunham fazer análises da eficiência de posicionamentos de *netlists* com a distribuição da complexidade dos fios de forma desigual, ou seja, em cada parte da *netlist* teríamos diferentes complexidades. A segunda proposta seria descobrir uma maneira de reestruturando a *netlist* melhorar a eficiência do posicionamento. Para esta linha, esta dissertação desenvolveu a *Logical Core*. Ainda que não exatamente da forma como foi proposta.

Na etapa de posicionamento não é possível, em termos de tempo, ter exatamente o comprimento de fios necessário para conectar as *n* células do circuito, pois para isso seria necessário rotear todo o circuito a cada ajuste do posicionador. Isto seria muito lento e demoraria muito para saber se o novo posicionamento é bom o suficiente. Então se usa uma heurística de aproximação do resultado. Na grande maioria dos casos aplica-se o chamado Semiperímetro (HPWL) a qual define que o comprimento de fio para conectar uma rede é igual ao Semiperímetro (HPWL) da *bounding box* mínima que engloba todas as células da rede. A heurística se aproxima bastante da realidade e valida assim os resultados, o que em outras palavras quer dizer que esta heurística é uma boa aproximação ao modelo Steiner.

Fazendo a análise de uma célula de uma net, a estimativa do Semiperímetro (HPWL) leva em conta três parâmetros:

- A quantidade de pinos de saída da célula. (cada saída é um novo retângulo Semiperímetro para somar)

- A quantidade de pinos de entrada da célula. (O retângulo do HPWL pode ficar maior com mais nodos, um caso onde não aumentaria, por exemplo seria quando uma nova pequena célula é adicionada entre as duas já conectadas, não influenciando na Bounding Box (BB))

- A célula com maior área. (O HPWL fica maior, pois a área da célula participa do cálculo, levando em consideração que para isso, ou aumenta na horizontal, ou aumenta na vertical ou aumentam os dois).

Figura 5-1: Características que o Semiperímetro considera em seu cálculo de importância. Didaticamente pode ser descrito como os posicionadores “veem” uma célula.

Porém o circuito combinacional é um conjunto de caminhos direcionado das entradas para as saídas, assim sendo, não podemos desprezar as células que fazem parte de um mesmo caminho apenas porque elas têm poucas conexões.

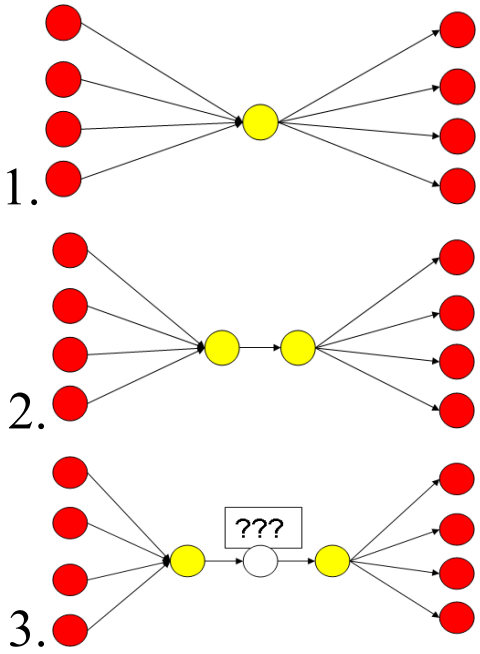


Figura 5-2: Exemplos de grafos: Em vermelhos (cinza escuro) os nodos de I/O. (em amarelo, cinza claro) os nodos mais importantes e o branco mostra o nodo que será erradamente calculado pelo Semiperímetro, pois deveria ter o mesmo peso dos amarelos.

Na Figura 5-2 temos três exemplos de um tipo de evolução da conectividade do grafo, onde mostra que à medida que a profundidade do grafo aumenta com relação aos pinos de entrada e saída o cálculo do Semiperímetro sofre problemas para constatar quais são os nodos mais importantes globalmente. Isso ocorre, pois o cálculo do Semiperímetro necessita ser rápido e ele não tem informações da relação entre redes e nem entre caminhos, a única informação que ele tem é local. Com a soma das informações locais é que conseguirias alcançar o fator global. Porém com isso demonstra-se que em termos de melhoria do comprimento de fio, a exploração deste indício melhoraria a qualidade do posicionamento global. O problema que temos aqui é: Como descobriremos as células mais significativas e como iremos explorar durante o posicionamento essas células.

Uma das técnicas que será apresentada visa atender esta deficiência da heurística. Para encontrar as células mais significativas necessitamos de um algoritmo que analise todos os caminhos e descreva neles uma forma de distribuir importância para todas as células, então assim terão um valor para conseguir o relacionamento global das células. Para esse primeiro problema foi desenvolvido o conceito de *Logical Core* que será apresentado no capitulo seguinte.

6 LOGICAL CORE I E II

Neste capítulo demonstraremos os dois principais algoritmos desenvolvidos neste trabalho. Os algoritmos *Logical Core I* e *II* para analise do grafo do posicionamento. Primeiramente iremos introduzir o conceito e a linha de raciocínio. Cada técnica será descrita e depois será proposta uma discussão para aumentar a controlabilidade no posicionamento.

O conceito da *Logical Core* baseia-se na compreensão de que um circuito pode ser entendido de forma diferente de um conjunto de células conectadas como redes. Ele também pode ser visto como um conjunto de caminhos direcionado das entradas para as saídas. Com a visão de caminhos temos o conceito de sobreposição de caminhos para que tenhamos variáveis diferentes atacando as portas (ou células) e criando lógica.

Porém tendo em vista que algumas células fazem parte de mais caminhos do que outras podemos dizer que essas células, do ponto de vista lógico, terão mais importância para a organização do posicionamento do que aquelas que fazem parte de um número menor de caminhos.

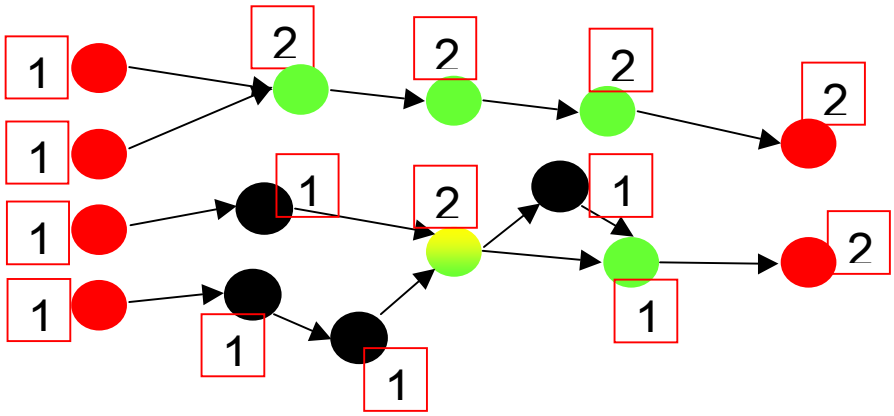


Figura 6-1: Exemplo de distribuição das probabilidades nas células do posicionamento.

Foram desenvolvidas duas versões de algoritmos, *Logical Core I* e *II*. O *Logical Core I* explora o grafo como uma busca em profundidade e vai "carregando" os valores e com isso acaba sendo mais lenta, porém consegue ter uma expectativa da dificuldade do posicionamento como um todo que inclusive compararemos com a Rent's Rule.

A segunda alternativa explora a Álgebra Linear intrínseca ao problema e utiliza a matriz de conectividade do grafo do posicionamento para otimizar o processo, com isso tem valores para cada célula, mas não temos as informações necessárias para dizermos se o circuito como um todo será difícil de posicionar. Ainda dentro da segunda implementação, temos duas versões. Utilizando o grafo de forma unidirecional ou de forma bidirecional, ou seja, no primeiro algoritmo os pesos se deslocam em apenas um

sentido (dos pinos de entradas para os pinos de saída), já na segunda eles podem ir e voltar livremente até se equilibrarem, esta segunda recria mais fielmente a teoria do algoritmo PageRank® (Page, 1998) do Google.

6.1 Logical Core I (Técnica de análise do circuito que utiliza Busca em Profundidade)

No primeiro algoritmo repassa iterativamente os valores setados nos terminais de entrada para todo o circuito, valorando a cada célula de acordo com o peso das que apontam para ela e da sua conectividade. Portanto temos uma distribuição probabilística de pesos que varia de acordo com a conectividade do grafo de entrada do problema de posicionamento.

Com essa organização temos um vetor de probabilidades de todas as células de participarem de um caminho crítico. A ideia teórica seria como se tivéssemos nos terminais nascentes de água limitada por uma constante. Essa água irá seguir pelas conexões alcançando as células, porém cada célula retém um pouco de água. Isso faz com que ao longo do tempo o algoritmo convirja.

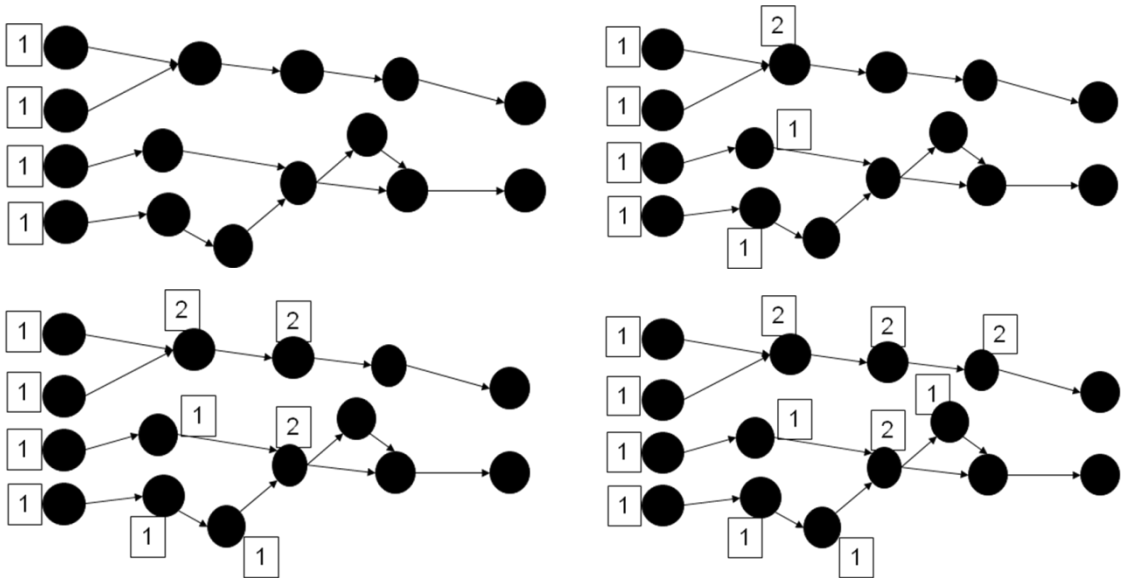


Figura 6-2: A sequência acima demonstra como o algoritmo da Logical Core I repassa os valores em cada uma das suas iterações. Os valores apresentados começam na figura superior esquerda, onde todos os pinos de entrada do circuito recebem o valor um.

Os dois algoritmos são baseados na ideia do algoritmo PageRank do Google (Page, 1998), agindo como um algoritmo de fluxo, no caso ele determina a importância das páginas de Internet, que é fortemente relacionado com o objetivo do *Logical Core I*, encontrar as células do grafo que são as mais importantes. Porém há algumas diferenças, por exemplo, a inicialização dos valores, onde temos valores iguais para todos os terminais de entrada, diferentemente do Google que inicializa com os seus próprios padrões de relevância.

Os padrões do Google dizem respeito a informações existentes antes do inicio do algoritmo. Por exemplo, todas as páginas que forem citadas pelo site da Universidade de

Stanford são “bons” sites. No caso da Logical Core não temos este tipo de informação pré-estabelecida o que aumenta o desafio da solução.

A organização algorítmica é baseada nos links de entrada na célula, ou seja, quais células apontam para a célula que queremos calcular. Quantas são e quais os pesos delas. As equações utilizadas pelo PageRank e pela *Logical Core I* são respectivamente:

$$PR(A) = (1-d) + d * \frac{PR(t1)}{C(t1)} + ... + \frac{PR(tn)}{C(tn)} \quad 6.1$$

$$LC(A) = (1-d) + d * \frac{LC(t1)}{C(t1)} + ... + \frac{LC(tn)}{C(tn)} \quad 6.2$$

O valor de *d,* chamado de *damping*. Pois o valor utilizado pelo Google não é expresso em lugar algum, sendo um dos segredos de mercado da máquina de busca.

A questão é a seguinte: *d* deve estar entre 0 e 1. Quanto mais próximo de 0, menos peso será passado ao seu nodo vizinho, assim sendo ele converge mais rápido mas perde em profundidade de difusão do peso e também em exatidão ao modelo teórico do *Logical Core I*. Quanto mais próximo de 1 ele demora mais a convergir mas tem bastante difusão e se aproxima mais do modelo teórico. Neste trabalho é utilizado o valor de 0.8, este valor foi selecionado de forma empírica.

O problema deste algoritmo é que o tempo de execução é grande, porém demonstrou bons resultados de melhoria da qualidade do posicionamento com a técnica proposta de alteração do Semiperímetro (HPWL). O tempo de execução nos obrigou a evoluir a técnica e desenvolver um segundo algoritmo que será explicada na sequencia.

6.2 Logical Core II (Técnica de análise do circuito que utiliza Multiplicação de Matrizes)

O segundo algoritmo utiliza um método mais eficiente. E ela foi desenvolvida em duas subversões. As diferenças são com relação a como percorremos o grafo. Em uma primeira versão de forma unidirecional (ou seja, a partir dos pinos de entrada) ou bidirecional (vindos a partir dos pinos de entrada e paralelamente vindo dos pinos de saída).

Este algoritmo tem o foco na mesma solução utilizada pelo Google. Tem como resultado a construção de uma matriz de pesos relativos a cada célula de acordo única e exclusivamente com a sua conectividade no grafo. O algoritmo se apresenta com a montagem de uma matriz de conectividade do grafo.

O algoritmo funciona a partir de uma multiplicação das matrizes de conectividade e a matriz com os valores de cada nodo. Este cálculo gera uma nova distribuição de pesos entre todas as células, ou seja a importância de cada célula vai modificando de acordo com a sua conectividade. Essa multiplicação das matrizes é iterativa e sendo assim segue até que os valores de todos os nodos do grafo se estabilizem, ou seja, permaneçam constantes frente ao valor calculado previamente.

Sob o ponto de vista matemático, a segunda implementação do algoritmo do *Logical Core* está calculando um autovetor da matriz de conectividade, A, das células. O algoritmo é baseado no algoritmo de Power (Demmel e Kågström., 88) para calcular o

autovalor dominante (em módulo) de uma matriz real. Abaixo tem um pseudocódigo da *Logical Core II*, onde x é o vetor dos pesos das células e b um vetor auxiliar do tamanho de x. O vetor b é inicializado com todas as suas posições com o valor 1.

```
Inicializa b = (1,1,...1)
então faça:
    x ← b
    b ← Ax
while norm2(b-x) > erro
```

Se a variável *erro* for um valor pequeno (na implementação estamos usando $1e^{-5}$), "*While* norm2(b-x) > *error*", pode ser lido como "enquanto a diferença entre b e x seja grande", ou melhor, "pare quando b for aproximadamente igual a x". Então se b for aproximadamente x, podemos escrever:

$$Ax = b \Rightarrow Ax = x \tag{6.3}$$

e concluir que o *Logical Core* encontrou um autovetor x de A cujo autovalor associado é um (1).

6.2.1 Análise Visual da Logical Core II

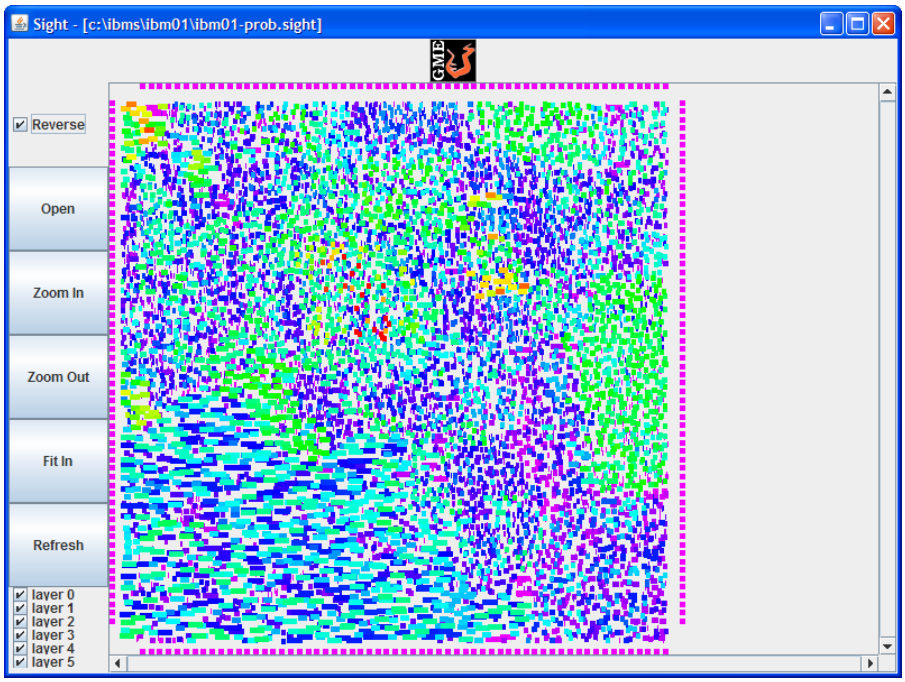
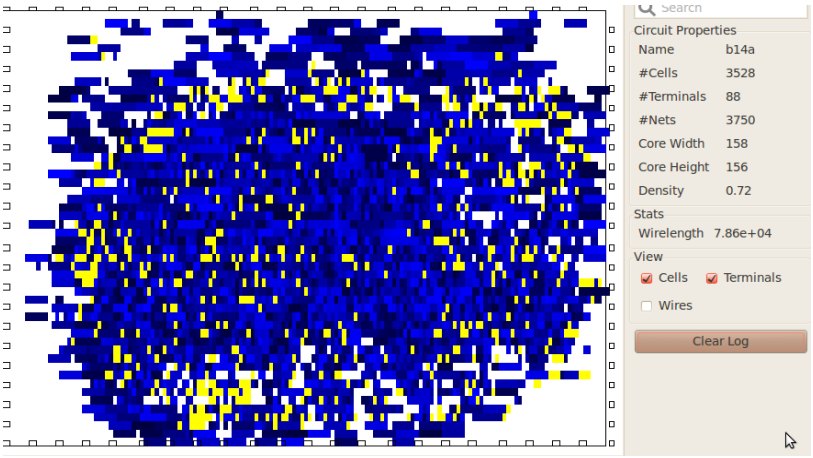
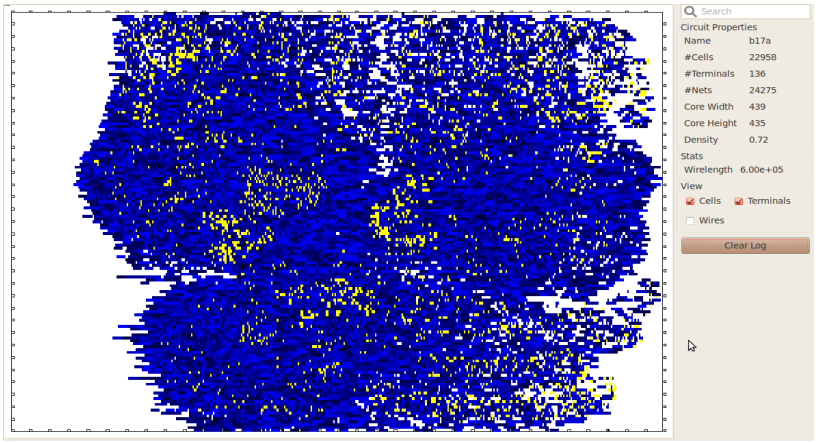


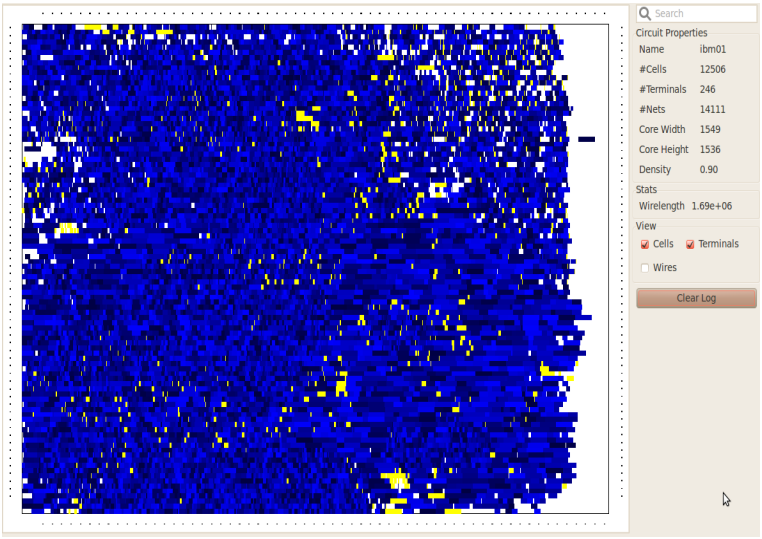
Figura 6-3: Posicionamento Global do *benchmark* ibm01 mostrado em um visualizador. Onde as cores representam a importância da célula, segundo a Logical Core, para posicionamento.

A partir da análise de vários benchmarks vimos que as células com maior peso não tem tendência de necessariamente serem posicionadas em algum lugar específico do circuito, ou seja, não existe uma lógica definida para onde estarão as células mais complexas de um circuito em um bom posicionamento.

Na Figura 6-4 vemos respectivamente os benchmarks b14, b17, ibm01, ibm15. As células do circuito foram coloridas de acordo com a sua importância para o posicionamento, segundo a Logical Core II. A distribuição de cores foi convencionada da seguinte maneira: as cores mais claras são as células de menor importância e à medida que a complexidade aumenta, as cores escurecem.


Search
Circuit Properties
Name b14a
#Cells 3528
#Terminals 88
#Nets 3750
Core Width 158
Core Height 156
Density 0.72
Stats
Wirelength 7.86e+04
View
Cells
Terminals
Wires
Clear Log


Search
Circuit Properties
Name b17a
#Cells 22958
#Terminals 136
#Nets 24275
Core Width 439
Core Height 435
Density 0.72
Stats
Wirelength 6.00e+05
View
Cells
Terminals
Wires
Clear Log


Search
Circuit Properties
Name ibm01
#Cells 12506
#Terminals 246
#Nets 14111
Core Width 1549
Core Height 1536
Density 0.90
Stats
Wirelength 1.69e+06
View
Cells
Terminals
Wires
Clear Log

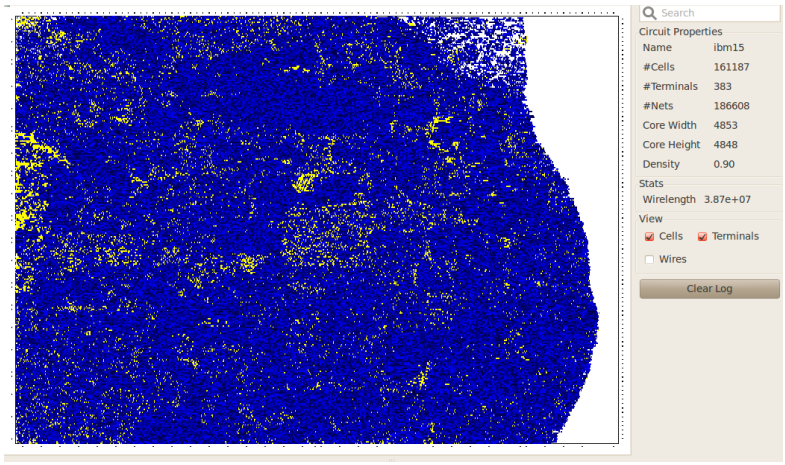


Figura 6-4: Nestas figuras vemos respectivamente os benchmarks b14, b17, ibm01, ibm15 com a distribuição da importância de cada uma das células calculada. Exibida da mais clara (menos importante) para a mais escura (mais importante).

6.3 Resultados do Logical Core I

Este subcapítulo apresenta os resultados da Logical Core I. A ideia foi demonstrar que a partir do *Logical Core I* conseguimos resultados próximos a Regra de Rent. Primeiramente calculamos a regra de Rent por sete vezes com uma ferramenta largamente utilizada chamada Rentc para um benchmark IBM e os resultados foram, respectivamente:

```
Rent exponent(p) = 0.534
Rent exponent(p) = 0.508
Rent exponent(p) = 0.528
Rent exponent(p) = 0.533
Rent exponent(p) = 0.563
Rent exponent(p) = 0.557
Rent exponent(p) = 0.551
```

Um problema para a utilização da regra para o posicionamento é que a regra de Rent não se mantém estável, devido às modificações que o particionador faz no circuito, o que demonstra que é um indicador ruim para um posicionador que necessita de eficiência no cálculo, pois precisa de vários cálculos seguidos para termos uma definição a partir da média. Uma alternativa seria utilizar a média que no caso do exemplo destes valores é 0,539, e o desvio padrão é de 0,019. O que significa 3,6% da média.

Mas o cálculo das distribuições de pesos proposto neste trabalho é estável, pois esta baseada na distribuição lógica do circuito. O cálculo vai responder quão "denso" é o circuito, e como isso implica no posicionamento, porém o calculo de Rent leva em consideração que à medida que o problema cresce a complexidade do circuito não

cresce linearmente. Porém nosso calculo relativiza a complexidade em um padrão que vai do número de células do circuito até o número de células ao quadrado do circuito.

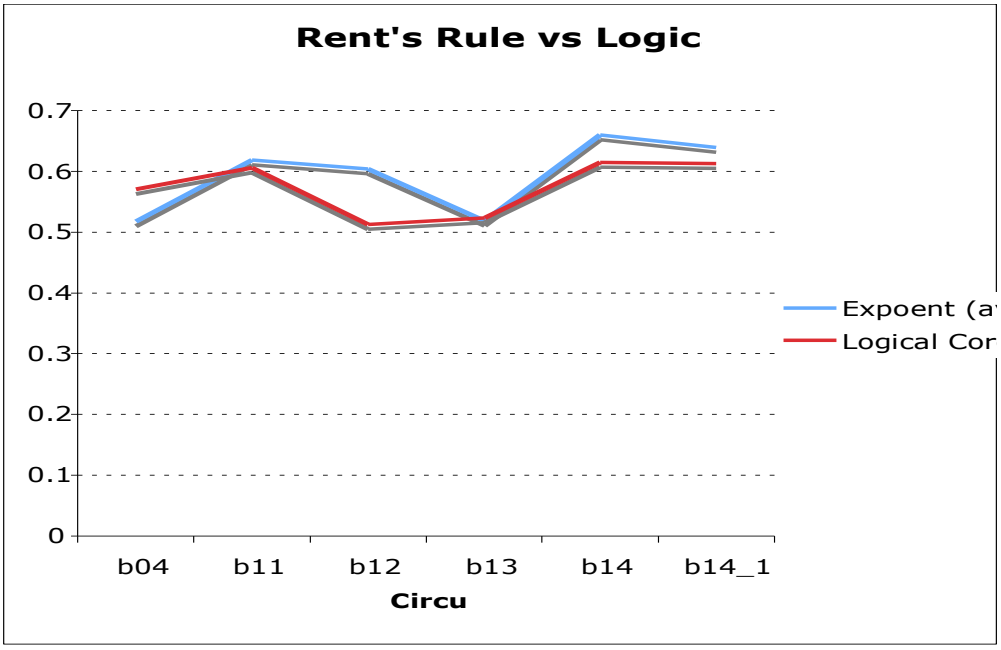


Figura 6-5: Nesta tabela temos uma comparação da Logical Core I com a Regra de Rent

6.4 Resultados da Logical Core II

Neste capitulo será demonstrado os resultados da técnica de modificação da função custo do Semiperímetro (HPWL). Na tabela 7.1 vemos o semiperímetro do circuito original calculado com o posicionador PlaceDL, na próxima coluna temos os resultados com a Logical Core e a diferença percentual da melhoria do comprimento de fio. Os tempos das duas técnicas são bem próximos por isso foram retirados.

Tabela 6-1: Demonstra os resultados da inserção da técnica de modificação da função custo do posicionamento.

Circuito (ISCAS)	Original	Logical Core	Comprimento de fio
b03	1674	1529	-8.66%
b04	7625	7323	-3.96%
b08	2189	2044	-6.62%
b09	1880	1851	-1.54%
b10	2709	2418	-10.74%
b11	9249	9190	-0.64%
b14	91350	87859	-3.82%
b14_1	83821	81432	-2.85%
b15	202641	199106	-1.74%
		TOTAL	4,51%

6.5 Controlabilidade e Complexidade do Grafo

Este capítulo propõe, a partir da *Logical Core I*, uma evolução na controlabilidade dos posicionadores existentes. A evolução consiste em que com a *Logical Core* sabemos quais *netlists* (grafos de entrada) são mais complexas e quais são mais simples, visando assim explorar estas informações de modo a auxiliar o posicionador a utilizar uma abordagem que resolva mais eficientemente cada problema.

A proposta é que antes de começar a solucionar o problema, crie-se um analisador do problema. Este pode ser a *Logical Core* ou outro que se ajuste ao posicionador. Algumas atitudes dos posicionadores são focadas demasiadamente em algum caso específico. Isto cria, por muitas vezes, etapas fixas que só serão realmente úteis em alguns poucos casos e que na maioria das vezes será pouco útil.

No capitulo introdutório citamos o cálculo da regra de Rent, que poderia ser um possível candidato a ser utilizado pelos posicionadores como controlador de dificuldade do posicionamento, porém a regra de Rent leva em consideração o posicionamento (a variável T representa o número de conexões que cruza o bloco, ao menos um particionamento é necessário) e a proposta é de uma medida de complexidade que analise apenas a *netlist*, ou seja, fazendo a análise antes do posicionamento e ficando livre da dependência pela qualidade do posicionamento. Prevendo a complexidade do problema é possível criar técnicas que evoluam a forma de utilização dos posicionadores. Por exemplo, caso a técnica identifique que um determinado grafo terá uma complexidade muito alta de ser posicionado, podemos criar soluções que o tratem mais especificamente, levando em consideração sua complexidade e assim poupando tempo de solução. Criando assim diferentes estratégias de soluções dos posicionadores para cada problema individualmente.

Ao tentar solucionar um problema complexo utilizando um posicionador analítico sabemos de antemão que teremos de utilizar muitas iterações para resolver o problema. Valendo-se desta informação podemos solucionar o sistema linear uma vez e gerarmos pinos virtuais para a nova solução do sistema e em vez de medirmos o novo comprimento de fio para saber o próximo passo, cortar esse passo das iterações e ir diretamente para o novo calculo do sistema linear, pois como o problema é complexo, não vamos ter conseguido grandes melhorias com as primeiras iterações. Com isso iremos reduzir o tempo da solução e probabilisticamente teremos o mesmo resultado em comprimento de fio.

A base da controlabilidade vem juntamente com a técnica de *Logical Core I*, em uma verificação simples, que é a soma de todas as complexidades de cada célula do circuito. Esta soma resultará um número que deve ser interpretado como a complexidade relativa do circuito da seguinte maneira:

A escala começa no número de células do grafo até número de células ao quadrado, porém quanto mais próximo de número de células maior será a complexidade do grafo. E quanto mais próximo de número de células ao quadrado, menor será a complexidade do grafo.

$$\sum P(todas\_celulas) \qquad 6.4$$

Figura 6-6: Ilustração da leitura do método de análise da complexidade do grafo do posicionamento.

A ideia teórica por trás da *Logical Core I* é que quanto menor o valor, mais as células estão conectadas e com isso repassam o valor mais vezes até conseguir se estabilizar. E como há uma redução natural do valor a cada repasse isso demonstra que temos mais conectividade no grafo para o mesmo número de células. É importante lembrar que a cada repasse o algoritmo recomeça, pois não sabe se este novo valor não esta faltando para o cálculo final de uma célula.

Também é importante citar que caso tivéssemos de posicionar um grafo completo (onde todos os nodos são conectados com todos os outros nodos), qualquer solução de posicionamento seria ótima. Para a *Logical Core I* teria valores bem altos na distribuição para todos os nodos. Já na *Logical Core II* todos os nodos ficariam com o valor constante e igual a um. Ou seja, neste caso a técnica não precisaria ser utilizada, mas este caso está longe do real e serve apenas para um analise teórica.

Uma análise sobre a capacidade da controlabilidade no *benchmark* do ISCAS99, b07, notamos que modificando apenas 8.5% das células mais significativas no posicionamento, estamos afetando 42% dos caminhos e fazendo isso de forma capaz, podemos melhorá-los, ou seja, com quase metade dos caminhos podem ser melhorados com pouco esforço se soubermos quais são as células, mas significativas do posicionamento. A proposta é que isso seja utilizado nos posicionadores futuros.

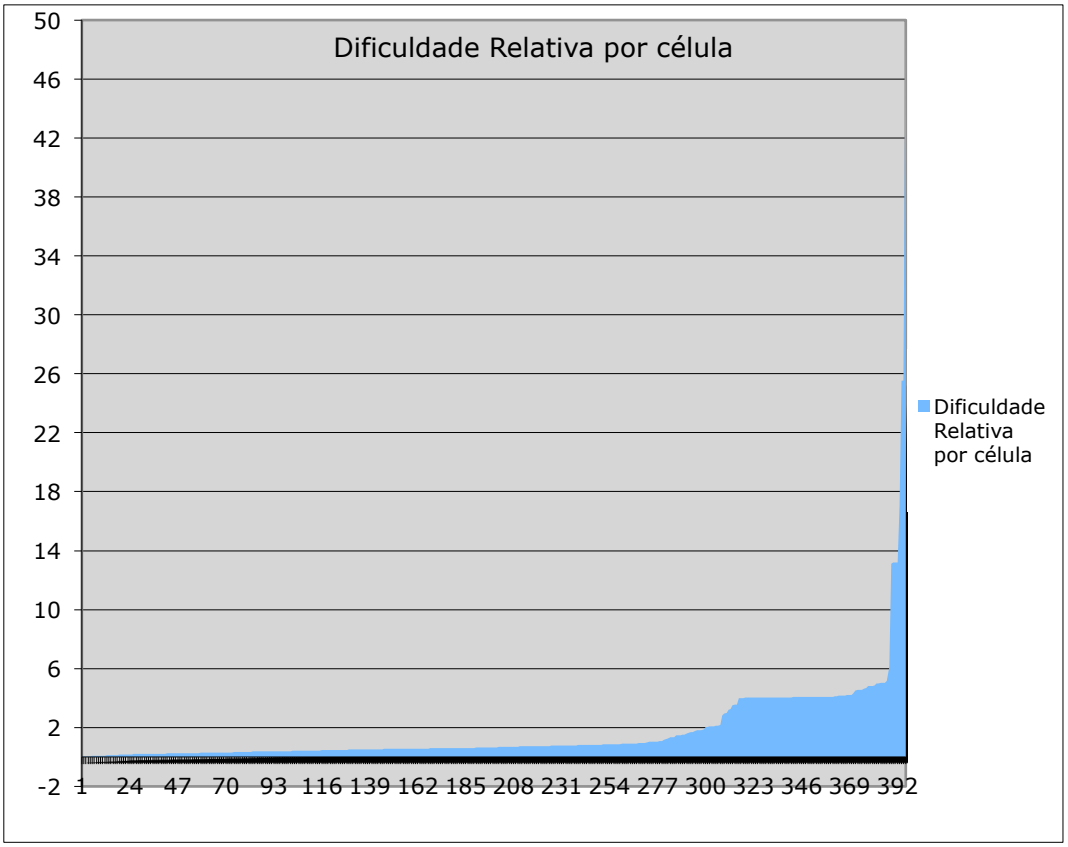


Figura 6-7: Distribuição das dificuldades (da menor para a maior) de um *benchmark* chamado b04 do ISCAS 99.

Isto traz um controle maior do problema de posicionamento, conhecimento sobre o grafo do circuito e um aumento da controlabilidade da solução do problema. Controlabilidade é um conceito que não costuma aparecer em problemas de posicionamento, mas se eficiência é necessária, então a controlabilidade faz com que tenhamos mais um conceito para alcançar os nossos objetivos.

Em FastPlace 3 (Viswanathan, 2007), fica claro que é utilizado um tipo de controlabilidade no processo, que necessita de um *tunning* bem apurado de como a ferramenta trata o problema. Quando no artigo Viswanathan conclui o capítulo com a seguinte frase (traduzida de forma livre): "até que fique bom o suficiente", ele na verdade está dizendo que existe um tipo de um controlabilidade, mas ela é empírica.

Entende-se que é necessário um controle maior do sistema do que simplesmente uma contagem de células para enfrentar todos os desafios de circuitos VLSI. Utilizando a *Logical Core* ou outra ferramenta de análise, qualquer posicionador poderá ser ajustado para ter como entrada a importância de cada célula e com isso fazer um ajuste para melhorar os resultados.

7 CLUSTERIZAÇÃO E A LOGICAL CLUSTER

Este capítulo explica a técnica de clusterização como parte introdutória ao algoritmo desenvolvido chamado *Logical Cluster,* que também será apresentado neste capítulo.

7.1 Clusterização

Na clusterização trabalha-se em um nível de abstração maior do que células. Trabalha-se no grafo que representa o posicionamento, a *netlist*, portanto temos nodos e arestas. Em posicionamento, formar um *cluster* significa reunir dois ou mais nodos do circuito em um grupo, ou um conjunto de forma a reduzir o tamanho do problema o máximo possível ou para aproveitar características dos nodos do mesmo grupo. Haja vista que desta forma reduzimos o número de nodos que precisam ser manipulados diretamente. É importante que depois que se resolva o problema com os nodos agrupados, também se precisa resolver o problema do posicionamento interno de cada grupo, também chamado de *declustering* (desagrupar). Esta redução impacta também o número de redes que conectam elas. Portanto temos uma redução das estruturas que manipulamos para resolver o problema.

Em um exemplo mais próximo das soluções atuais poderíamos utilizar como exemplo: o número de células a serem particionadas; o tamanho da entrada para o algoritmo Simulated Annealing; ou a matriz de conectividade. Ainda temos redução do impacto em termos computacionais de forma mais geral, como, por exemplo, a redução do uso de memória.

Esta técnica tem uma grande ligação com a técnica de particionamento. Em geral as técnicas de *clustering* objetivam otimizar o tempo de execução da solução de um posicionamento, pois primeiramente resolvem um problema menor que é o posicionamento do subgrafo como um macrobloco de células, e depois se resolve os posicionamentos dentro de cada bloco. A ideia de diminuir a entrada de um determinado problema descartando informações menos importantes é uma estratégia utilizada em muitos outros problemas da computação. É importante ressaltar que posicionamento ainda hoje é uma etapa bastante significante em termos de tempo da síntese física. Por outro lado, toda simplificação leva à perda de informação. E a clusterização, sendo ela um tipo de compactação de informação, pretende descartar informação redundante e preservar informação relevante.

Outra vantagem que se torna evidente levando em consideração a atual direção das arquiteturas de computadores é que uma técnica de clustering facilita a paralelização do sistema, onde é possível com um bom *clustering* resolver o sistema macro (blocos de células) em um processador e ao mesmo tempo solucionar os posicionamentos internos de cada bloco em outros processadores. Desta forma a utilização de *clustering* cresce de

importância para os posicionadores futuros, onde explorar programação paralela é visto como uma tendência bastante forte.

O primeiro a propor uma ideia de clusterizar os nodos adjacentes na história do posicionamento foi Schuler (1970), desde lá muitas técnicas de otimização foram criadas. Revisaremos alguns destes importantes trabalhos para a história da clusterização de circuitos integrados.

Uma técnica de *Clustering* bastante importante historicamente data de 1979, quando (Stabler e Eurichik, 1979) desenvolveram uma ideia de clusterizar a cada linha da *grid mesh*. Isto significa que ao desenvolver este trabalho utilizava um tipo de *Standard Cells*, pois todas as células tinham a mesma altura, desta forma, ele agrupava todas as células que um particionador "vertical" havia designado para aquela determinada "linha" do circuito. O objetivo era minimizar a seguinte função:

$$k = \sum_{i=1\ldots n} \sum_{j=1\ldots n} kij \qquad 7.1$$

onde Kij é o número (ou peso, dependendo do autor) das conexões que cruzam do grupo i para o grupo j. No algoritmo é fixa a utilização de um grupo, um para cada linha real do circuito. E dado que exista uma heurística que troca de forma inteligente as células para melhorar o resultado final do agrupamento das células. Algumas evoluções a este algoritmo utilizavam a técnica de *Branch-and-Bound* para melhorar ainda mais o resultado.

Para explicar a valoração que calcula se uma célula deve ou não ser clusterizada com outra assumiremos que dois objetos v1 ϵ h e ϵ v2 h foram clusterizados, onde h é uma hiperaresta de k-pinos do grafo.

No trabalho de (Karypys e Kumar, 1999) o cálculo do valor é 1/ (k-1-1). Enquanto em (Cong e Lim, 2004) a equação se modifica para 1/ (k – 1). Esta pequena diferença de uma aresta se dá porque a transformação de uma hiperaresta para um clique de arestas acontece apenas uma vez antes da clusterização.

O outro passo foi desenvolvido por Chan (2000) que modificou o algoritmo utilizando uma técnica chamada *connectivity-base*. Porém, sob o ponto de vista de alguns autores, a grande diferença foi à inserção da área utilizada pelos vértices no cálculo da clusterização, em Schuler e Ulrich (1972). Com isso ele conseguiu uma solução mais balanceada em termos de distribuição das células entre os grupos, ponto esse que segue sendo bastante significativo caso não haja uma pré-analise da área ativa que determinado grupo de células irá utilizar.

Entre os trabalhos do estado da arte se destacam a técnica chamada *Best-Choice* de (Alpert e Khang, 2005). Esse trabalho alcança uma melhora no desempenho do posicionador de 2.2 vezes. E ainda uma melhora na qualidade do posicionamento de 4,2% em média. O teste foi efetuado sobre o posicionador CPLACE (Alpert, 2003), do seu próprio grupo.

7.1 Logical Cluster (Técnica de Clusterização)

Conforme descrito em (Pinto, 2010) a *Logical Core* nos fornece uma relação de prioridade ou importância que as células têm no circuito integrado a partir da estrutura lógica do grafo. Portanto ela necessita apenas da *netlist* de entrada do circuito.

A *Logical Core II* analisa a conectividade de cada célula do grafo e realiza uma ponderação de quanto conectada esta a célula com relação às suas vizinhas diretas e indiretas, porém à medida que a distância lógica aumenta, a influência diminui. A técnica da *Logical Core II* é o que fez a *Logical Cluster* ser possível, sem ela não teríamos a ponderação entre as células. Antes deste trabalho tínhamos um valor para o circuito como um todo a partir da Regra de Rent, porém não uma relação individual para cada célula do circuito independente como a *Logical Core II*.

Resumidamente, células com alto valor de probabilidade de serem importantes (valor do resultado da *Logical Core*) participam de mais caminhos, ou seja, são mais fortemente conectadas. Logo o raciocínio da técnica é clusterizar as células com baixo valor de probabilidade de serem importantes, uma vez que, segundo a técnica, elas têm pouca relevância no circuito. Em outras palavras a técnica retira da entrada do problema (grafo) as células que não tem muita importância para o resultado final e se foca nas células com maior importância.

A seguinte metodologia de clusterização foi testada, conforme descreve o algoritmo abaixo:

```
ordena (crescente) vetor de células de acordo com a Logical Core
para cada célula até LIM
      pega célula C
      área_cluster = 0
      faça para cada célula conectada a C
            pega célula C'
            se probabilidade(C') < probabilidade(LIM)
                  clusteriza (C, C')
      enquanto área_cluster < 4*area_média_célula
fim
```

No pseudocódigo acima, LIM refere-se à célula limite no vetor ordenado crescentemente pelo valor de probabilidade. Esse valor foi fixado em 0.3 vezes o número de células, ou seja, 30% das células do circuito.

Então, é visitada uma percentagem do total de células, da célula de menor probabilidade à célula limite. A escolha por limitar a área do cluster em 4,5 vezes a área média de uma célula foi baseada nos resultados de Alpert (2005), com o algoritmo utiliza-se 4 vezes.

A técnica da Logical Cluster é extremamente rápida. O tempo de execução para todos os benchmarks testados não chega a 1 segundo, isto se deve a simplicidade do algoritmo que executa apenas para as células com menor probabilidade. Então as *n* células dessa parte do circuito são testadas em uma operação de menor (que significa uma subtração e um teste se é igual à zero) com as *m* células que as conectam. Nos circuitos em geral, a maior parte das redes têm 2 e 3 células, o que demonstra que serão

feitos poucos testes. Então é calculada a área média do cluster. Feita essa análise, a partir dos conceitos de complexidade de algoritmos, vemos que será rápida a solução que a Logical Cluster propõe. Para os *benchmarks* IBM temos que o tempo de execução foi quase insignificante para todos os circuitos disponíveis.

A parte mais demorada é a execução da Logical Core II, pois o algoritmo precisa conseguir encontrar a convergência. Mas por outro lado a Logical Core II é bastante paralelizável o que significa um ganho significativo no tempo de execução. Na vemos os tempos de geração do vetor com as importâncias de cada célula no circuito para todos os *benchmarks*.

7.2 Resultados da Técnica Logical Cluster

Neste capítulo apresentaremos os resultados da *Logical Cluster*. Em geral, os posicionadores não tem código aberto, ou seja, apenas os executáveis são disponibilizados para pesquisa. Esta linha de difere de raciocínio difere da ideia que se tem no grupo de microeletrônica, onde tentamos disponibilizar os códigos.

Nesta parte utiliza-se um método de teste que modifica o circuito, ainda enquanto *netlist*. Fazem-se *clusters* a partir das analises da Logical Core e então se resolve o novo problema, com a nova *netlist* que representa o mesmo circuito.

Nas técnicas de clustering, ao final de uma etapa de posicionamento, se faz necessário reajustar a posição das células que por opção da técnica foram clusterizadas, agrupadas. Para que agora disjuntas, continuem mantendo uma boa qualidade no comprimento de fio. Portanto todos os resultados obtidos foram realizados seguindo o seguinte roteiro de testes: (1) clusterização do circuito, (2) posicionamento global, (3) desclusterização do circuito e (4) posicionamento detalhado.

Nós utilizamos 3 posicionadores diferentes para embasar e demonstrar o funcionamento da técnica. O posicionador NTUPlace (CHEN, 2006), o FastPlace3 e o PlaceDL. Aqui cabem duas observações importantes. o PlaceDL apenas executa posicionamento global e aqui temos uma versão um pouco modificada da explicada no artigo original do PlaceDL, apenas baseado na versão do capítulo do PlaceDL, é a última versão disponível.

Neste ambiente de testes o sistema operacional foi o Ubuntu 10.04, e o hardware foram dois processadores Intel RXeon Nehalem 2.26GHz serie 5500 (8 cores com capacidades de hyperthreading) e 8GB de RAM.

Na Tabela 7-1 podemos ver o tempo do cálculo da Logical Core II para os benchmarks IBM, visto que os tempos são relativamente pequenos perto do tempo de resolução do problema de posicionamento, isso demonstra a viabilidade da utilização da técnica para circuitos do estado da arte.

Tabela 7-1: Tempo do cálculo da Logical Core II para os benchmarks IBM em segundos.

Logical Core

Ckt	Tempo (s)
ibm01	0,32
ibm02	0,22
ibm03	0,13
ibm04	0,69
ibm05	0,32
ibm06	0,19
ibm07	0,42
ibm08	0,87
ibm09	0,71
ibm10	1,2
ibm11	0,86
ibm12	0,77
ibm13	1,06
ibm14	3,23
ibm15	2,43
ibm16	2,26
ibm17	3,28
ibm18	7,36

Na tabela 7.3 notamos que a redução do tamanho do circuito é bastante significativa. E também conseguimos notar que à medida que os circuitos crescem há uma tendência da técnica de conseguir reduzir ainda mais o circuito. A Logical Cluster esta partindo de um conjunto de 30% das células menos importantes para a resolução do problema de posicionamento, ou seja, a técnica analisa as células e a medida que vai calculando as 30% menos importantes ela agrupa essas células em outras próximas de forma a equalizar o valor das importâncias.

Na tabela 7.5 nós temos a descrição completa dos circuitos IBM, que nos informa a estrutura do circuito, facilitando a compreensão dos resultados da Logical Cluster. Temos, para cada *benchmark*, o número de células, o número de pinos de entrada e saída, o número de redes, o número de pinos das células e por último o número de linhas S*tandard Cell* do circuito, pois um circuito baseado na estrutura Standard Cell se caracteriza por ter uma altura fixa para todas as linhas ou bandas. Essa padronização das células leva a uma diminuição dos problemas enfrentados para se projetar um circuito, mas também traz algumas impossibilidades de ganhos em otimizações lógicas entre células ou até em redução de área.

Tabela 7-2: Redução do tamanho dos circuitos IBM com a clusterização de 30% das células do circuito.

Redução de Tamanho do Circuito (%)

CKT / %	0,3
ibm01	-9,35
ibm02	-15,90
ibm03	-8,30
ibm04	-10,21
ibm05	-15,46
ibm06	-11,39
ibm07	-8,15
ibm08	-14,58
ibm09	-9,78
ibm10	-12,15
ibm11	-11,11
ibm12	-9,75
ibm13	-11,50
ibm14	-10,37
ibm15	-14,18
ibm16	-13,00
ibm17	-13,56
ibm18	-15,35

Na tabela 7.4 faz-se uma comparação entre o tempo de resolução do FastPlace3 (*baseline*) e o tempo com a utilização do *Logical Cluster*. Aqui se pode notar que a redução do tempo de execução para o mesmo comprimento de fio foi de 15,1%, uma porcentagem expressiva.

Outro ponto interessante na utilização da Logical Core é que além de diminuirmos o tempo de execução do processo de solução a partir da Logical Cluster, ainda existe uma possibilidade de otimizar o resultado a partir das informações já obtidas.

Em estudos futuros pode-se utilizar essa vantagem para refinar o resultado do posicionamento. Há uma tendência de que as células mais importantes a partir do cálculo da Logical Core influenciem mais na solução final. Então, por exemplo, poderia se utilizar em melhorias como ponderações de trocas em um refino via Simulated Annealing, em uma ponderação das forças via algum refino proveniente de uma analise física, entre outras possibilidades.

Tabela 7-3: Compara o tempo de resolução do FastPlace3 original com o tempo do caso em que o grafo é preparado antes, através da utilização da Logical Cluster. Vemos que a redução do tempo de execução para o mesmo comprimento de fio foi em média 15,1%. A partir do conjunto de *benchmarks* IBM nota-se que o tamanho dos circuitos não influência a qualidade da clusterização e sim características próprias da estrutura do circuito.

T_GP+T_DP	em segundos		
CKT / %	Logical Cluster	FastPlace3_Original	Comp. Fio
ibm01	6,61	6,62	1,00
ibm02	13,07	14,11	0,93
ibm03	12,61	14,92	0,85
ibm04	14,50	20,88	0,69
ibm05	18,65	20,93	0,89
ibm06	21,00	23,99	0,88
ibm07	26,28	28,99	0,91
ibm08	40,53	49,85	0,81
ibm09	31,84	38,83	0,82
ibm10	43,05	58,04	0,74
ibm11	44,10	59,50	0,74
ibm12	52,91	62,02	0,85
ibm13	62,95	85,92	0,73
ibm14	84,55	100,81	0,84
ibm15	147,03	154,48	0,95
ibm16	138,16	174,16	0,79
ibm17	160,45	145,00	1,11
ibm18	168,76	168,44	1,00
		Média	0,849

Na Tabela 7-3 podemos notar a redução do tempo de execução foi de 15,1% em média.

A partir do conjunto de *benchmarks* IBM nota-se que o tamanho dos circuitos não influência a qualidade da clusterização e sim características próprias da estrutura do circuito. Estas características individuais também podem servir para estudos futuros, visto que é um dos caminhos para alcançarmos soluções próximas do posicionamento ótimo. Já que até hoje ainda não temos circuitos reais com o seu conjunto de soluções ótimas, só temos os circuitos PEKO, que foram artificialmente construídos.

Tabela 7-4: Descrição dos benchmarks IBM que foram utilizados nos testes. Aqui temos o número de células, pinos de entrada e saída, redes, pinos das células e por último o número de linhas standard cell do circuito.

Ckt	#Cells	#Pads	#Nets	#Pins	#Rows
ibm01	12506	246	14111	50566	96
ibm02	19342	259	19584	81199	109
ibm03	22853	283	27401	93573	121
ibm04	27220	287	31970	105859	136
ibm05	28146	1201	28446	126308	139
ibm06	32332	166	34826	128182	126
ibm07	45639	287	48117	175639	166
ibm08	51023	286	50513	204890	170
ibm09	53110	285	60902	222088	183
ibm10	68685	744	75196	297567	234
ibm11	70152	406	81454	280786	208
ibm12	70439	637	77240	317760	242
ibm13	83709	490	99666	357075	224
ibm14	147088	517	152772	546816	305
ibm15	161187	383	186608	715823	303
ibm16	182980	504	190048	778823	347
ibm17	184752	743	189581	860036	379
ibm18	210341	272	201920	819697	361

Tabela 7-5: Compara os resultados da técnica de Clustering chamada Best-Choice com o Logical Clustering com a porcentagem de células do circuito clusterizadas sob parâmetro. Demonstrando a redução do tamanho final do circuito.

HPWL			
CKT / %	0,3	FastPlace3_Original	Comp.
ibm01	1739968	1727729	1,01
ibm02	3598260	3598083	1,00
ibm03	4823746	4646751	1,04
ibm04	5717478	5732461	1,00
ibm05	9876540	9884011	1,00
ibm06	4972950	4936872	1,01
ibm07	8130475	8227714	0,99
ibm08	8953440	9048117	0,99
ibm09	9458114	9667507	0,98
ibm10	17417888	17268496	1,01
ibm11	14275271	14635573	0,98
ibm12	22499084	22838560	0,99
ibm13	17049844	16882824	1,01
ibm14	31635772	31896196	0,99
ibm15	38439764	38524492	1,00
ibm16	44473792	44243820	1,01
ibm17	63549392	59920616	1,06
ibm18	41314384	40676484	1,02
		Média	1,000

ckt	Original	BestChoice	LC_0.1	LC_0.2	LC_0.3
ibm01	1,00	0,99	0,79	0,99	0,90
ibm02	1,00	0,93	0,89	0,67	0,69
ibm03	1,00	1,25	0,24	1,86	0,73
ibm05	1,00	0,95	0,99	0,86	0,86
ibm06	1,00	0,66	0,43	0,85	0,81
ibm07	1,00	2,29	0,78	1,80	0,93
ibm09	1,00	0,61	0,17	1,24	1,25
ibm10	1,00	0,25	0,49	0,72	0,42
ibm11	1,00	0,74	0,47	0,37	0,37
ibm13	1,00	0,48	0,80	0,70	0,95
ibm17	1,00	1,17	1,32	0,71	0,92
ibm18	1,00	0,76	0,81	1,16	0,70
Avg.	1,00	0,93	0,68	0,99	0,79

Também temos uma comparação com outra técnica de clustering já citada neste trabalho chamada Best-Choice. Na tabela 7.6 demonstra a diferença do tamanho do grafo gerado após a utilização da técnica de *clustering*. Pode-se notar que a redução com o parâmetro 0.3 (que significa 30% das células com menor valor no resultado da Logical Core), que foi justamente o valor utilizado em nossos testes reduz cerca de 14% mais o tamanho do circuito do que um algoritmo do estado da arte. Focando-se nos algoritmos da Logical Cluster que utilizaram parâmetros de 0.1 e 0.2, notamos que a redução variou bastante o que dificulta as conclusões.

Os resultados da NTU foram bem interessantes como pode ser conferido na **Tabela 7-6**.

Tabela 7-6: Resultados de tempo da Logical Cluster na NTU contra o NTU original.

Tempo Total

Ckt	NTU3_original	LC	variação
ibm01	30	23,32	0,777
ibm02	48	47,22	0,984
ibm03	74	63,13	0,853
ibm04	106	74,69	0,705
ibm05	124	80,32	0,648
ibm06	112	76,19	0,680
ibm07	228	173,42	0,761
ibm08	205	153,87	0,751
ibm09	251	223,71	0,891
ibm10	400	309,2	0,773
ibm11	345	319,86	0,927
ibm12	471	317,77	0,675
ibm13	508	452,06	0,890
ibm14	1151	849,23	0,738
ibm15	1960	1541,43	0,786
ibm16	2535	1893,26	0,747
ibm17	2649	2118,28	0,800
ibm18	2191	2269,36	1,036
			0,801

8 CONCLUSÕES E TRABALHO FUTUROS

Além das conclusões sobre este trabalho, este capítulo conclusivo trará visões importantes de trabalhos já existentes, assim como reflexões de perspectivas futuras para a pesquisa e evolução do posicionamento.

A respeito da *Critical Star*, um ponto importante é que em Liu (2004), uma das conclusões do trabalho diz: quanto mais *global nets* em um posicionamento analítico, pior será a eficiência do posicionamento e pior será a qualidade do resultado. Pelas experiências com a *Critical Star* vemos que a conclusão deveria ser modificada, pois inserimos *global nets* na estrutura da *netlist* do *FastPlace3* e com isso obtivemos melhorias na qualidade do posicionamento em muitos casos.

Para tanto tivemos que colocar uma *global net* exata, e com isso ajudar a identificar os caminhos pode ajudar a resolução analítica e melhorar a qualidade da solução, porém do ponto de vista de tempo de execução ela complica ainda mais a solução do problema. Como trabalho futuro se pode analisar outras características dos *benchmarks*.Para entender melhor onde o algoritmo tem mais ganhos e quais configurações têm de ser modificadas para obtermos bons resultados em outros posicionadores e em outros *benchmarks*.

Uma curiosidade é que em Liu (2004) diz que iriam trabalhar sobre a estrutura da *netlist* para aumentar a eficiência do posicionamento e este trabalho muda a *netlist* para inserir melhorias de qualidade no posicionamento. Porém também é adicionado como trabalhos futuros modificar a *netlist* para obter eficiência no posicionamento, ou seja, retirar ou substituir arestas no grafo de forma que a resolução analítica não perca qualidade na solução. É visto que o *Logical Cluster* faz isso com "compressão" de nodos, porém entendemos que isso pode ser evoluído para arestas também. Um trabalho efetuado sobre arestas necessita de maior conhecimento específico sobre o modo que um posicionador resolve o problema, por exemplo, para sabermos que uma determinada aresta é "dispensável" ou não para certo posicionador, estaremos estabelecendo um padrão de comportamento que poderá ser replicado muitas vezes. Quanto mais vezes um determinado padrão se repete, maiores serão os ganhos ao identifica-lo e utiliza-lo.

Outro ponto que merece reflexão é que o *Dragon* é o posicionador mais sensível à alteração na *netlist*, o que significa que é o posicionador que mais leva em consideração pequenas alterações no grafo do posicionamento, alterando assim de forma contundente o resultado final.

No CAPO notamos que quanto mais caminhos críticos adicionamos melhor o posicionador aperfeiçoa o comprimento de fio crítico sem comprometer o comprimento e fio total, já no caso do *FastPlace3* independente do tamanho do *benchmark*, temos um número médio de caminhos críticos bem definidos, entre 10 e 20 caminhos, o que

apresenta uma dificuldade maior do método analítico para administrar uma grande quantidade de redes em seus posicionamentos do que do método utilizado pelo CAPO.

Uma conclusão importante é que o número de caminhos críticos ideal para o CAPO tende a crescer com o aumento do número de células no circuito, como mostra gráfico 4.7. Como a *Critical Star* não tem a controlabilidade necessária para saber o quanto afeta o circuito, temos um ruído que não conseguimos retirar empiricamente. E no caso do *FastPlace3* temos uma estabilidade no número de caminhos críticos que fica entre 10 e 20 caminhos, um número relativamente baixo para o número de células e redes dos *benchmarks* estudados. Isto demonstra que os posicionadores analíticos têm grande vantagem para a utilização desta técnica. Destaca-se também a possibilidade de no futuro encaixar a técnica da *Critical Star* com a técnica da *Logical Core*, de forma a conseguir escolher melhor os caminhos, talvez até de forma iterativa, para alcançar uma solução melhor.

Outra abordagem interessante é que no momento que colocamos um número excessivo de caminhos críticos (sempre lembrando que "excessivo" é relativo ao *benchmark* e ao posicionador juntos, ou seja, é uma análise difícil de prever) temos uma piora significativa no comprimento de fio global, pois o posicionador se foca demais nos caminhos críticos e perde o ajuste para o comprimento total dos fios. Isto segue a proposição descrita em (Liu, 04), ou seja, em alguns casos é possível modificar o grafo com aumento de qualidade de solução. Mas se aumentamos em excesso perdemos o ajuste global e isso afeta demais o comprimento de fio.

Na segunda parte temos o desenvolvimento de um algoritmo chamado *Logical Core* com uma propriedade de convergência relativamente rápida. Algoritmo que cria a possibilidade de entendermos o quanto cada nodo do circuito influência na dificuldade de posicionar o grafo. Também temos uma evolução no conceito de controlabilidade inserido com o algoritmo do *Logical Core*. Este conceito permite, por exemplo, a análise de que reposicionando, ao menos globalmente, 9% das células do circuito estamos afetando 42% dos caminhos, que nos traz uma probabilidade alta de afetarmos um caminho critico junto com a melhoria do comprimento de fio total.

Como o algoritmo trabalha em um nível de abstração relacionado com a conectividade do grafo, temos potencialmente, uma probabilidade de ganhos em vários objetivos diferentes como desempenho, comprimento médio dos caminhos, entre outros, porém a utilização da técnica para estes objetivos experimentalmente fica como proposta para trabalhos futuros. Trabalhando com este nível de abstração criamos inúmeras possibilidades de analises que ainda não foram pesquisadas, como os impactos do grafo em potência, em área, entre outros.

Outra analise é que à medida que a *netlist* aumenta (por exemplo, linearmente, 1, 2, 3, 4, 5,...) a probabilidade de caminhos afetados aumenta ainda mais (a sequencia é a 1, 3, 6, 10, 15,...), fazendo com que os problemas maiores tenham relações percentuais ainda maiores do que os nove para 42 descritos no exemplo. Com isso quanto maior o problema maior o número de caminhos que uma modificação no posicionamento irá acarretar probabilisticamente.

Outro ponto importante é a utilização a partir da Logical Core I para a solução parcial de uma característica do Semiperímetro (HPWL). Que traz a possibilidade de um uso em conjunto das duas heurísticas para que tenhamos em uma a visão mais global e na outra uma visão mais realística, sempre levando em consideração que o fator

de ajuste das duas técnicas é muito importante em qualquer tipo de solução que seja implementada, aqui demonstramos claramente a soma utilizada.

Com o desenvolvimento da *Logical Core* se consegue definir quais as células do circuito são as mais importantes para o posicionamento e quais são as menos importantes com um algoritmo relativamente rápido, frente aos tempos atuais de resolução do posicionamento dos circuitos utilizados na indústria. Conseguimos excelentes resultados que evoluíram o desempenho dos posicionadores que são hoje em dia, o estado da arte.

A técnica cria grandes possibilidades de estudo, pois ainda não existia algoritmo que conseguisse ponderar a dificuldade de posicionar uma célula. Nem de forma relativa, nem de forma absoluta. Os estudos deverão evoluir ainda para técnicas de particionamento e divisão do circuito. Outra área que deve evoluir é a criação de uma maior parametrização para o projetista do circuito a ser posicionado. Tendo em vista que as mudanças vão possibilitar a criação de ajustes ainda menores entre qualidade e tempo de execução.

Os resultados de diminuição de 20% do tempo para resolver o problema utilizando um posicionador do estado da arte como o *NTUplace*, demonstram a grande valia do algoritmo para auxiliar a resolução de problemas de posicionamento. Também na comparação com outras técnicas de clustering a *Logical Cluster* obteve resultado superior, na comparação com o algoritmo chamado *Best-Choice* conseguimos reduzir em 14% a mais o tamanho do problema.

Um trabalho futuro interessante trata de seguir alguns resultados não apresentados neste trabalho que demonstram que a técnica consegue aumentar a roteabilidade (capacidade de se rotear o circuito) de uma solução, pois ele junta partes menos densas do circuito e com isso deixa mais livre as partes mais densas.

9 REFERÊNCIAS

ADYA, S. N., YILDIZ M. C., MARKOV, I. L., VILLARRUBIA, P. G., PARAKH P. N. AND MADDEN P. H., "Benchmarking For Large-scale Placement and Beyond", International Symposium on Physical Design (ISPD), pp. 95-103, Monterey, CA, April 2003

ADYA, S., MARKOV, I.-CongestionMaps-Win32.exe - CongestionMaps Plotter http://vlsicad.eecs.umich.edu/BK/PlaceUtils/ > Acesso em set/2010

ADYA, S.N., YILDIZ, M.C., MARKOV, I.L., Villarrubia P.G., Parakh P.N. e Madden P.H., "Benchmarking for Large-scale Placement and Beyond", em Proc. International. Symposium on Physical Design, pp.95-103, 2003, Monterey, CA, USA.

ALLAN, A. EDENFELD, D. JOYNER, W.H., JR. KAHNG, A.B.RODGERS, M. ZORIAN, Y., International Technology Roadmap for Semiconductors, http://www.itrs.net/home.html-2001 technology roadmap for semiconductors - > Acesso em set/2010

ALPERT, C. J., NAM, G.-J., VILLARRUBIA, P. G., "Effective Free Space Management for Cut-based Placement", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22(10), pp. 1343-1353, 2003, [S.l.].

ALPERT, C. J.; CHAN, T.; HUANG, D. J.-H.; MARKOV, I.; YAN, K. "Quadratic placement revisited". In: DAC '97: Proceedings. ACM Press, 1997. p.752-757. Anaheim, CA, USA.

ALPERT, C., KAHNG, A., NAM, G-J. , REDA, S. , VILLARRUBIA, P., A semi-persistent clustering technique for VLSI circuit placement, Proceedings of the 2005 international symposium on Physical design, April 03-06, 2005, San Francisco, California, USA

BAKOGLU, H. B., Circuits, "Interconnects and Packaging for VLSI". Addison-Wesley Publishing Company, 1990.

CALDWELL, A. E.; KHANG A. B.; MARKOV, I. L. "Can Recursive Alone Produce Routable Placements?" In Proceedings Design Automation Conference, 2000. Proceedings 2000. 37th, Los Angeles, California, USA.

CHAN, T., CONG, J., KONG, T., AND SHINNERL, J., "Multilevel Optimization for Large-Scale Circuit Placement," in Proc. IEEE International Conference on Computer-Aided Design, 2000, pp. 171-176, San Jose, CA, USA

CHANG, C-C, XIE, M, CONG, J. PEKO-Benchmarks, 2002 - http://cadlab.cs.ucla.edu/~pubbench/placement/ > Acesso em set/2010

CHEN, T.-C., JIANG, Z.-W., HSU, T.-C., CHEN, H.-C., CHANG Y.-W., NTUplace3 (http://eda.ee.ntu.edu.tw/w04/download/ntuplace.php) > Acesso em set/2010

CHOU, Y-C, LIN, Y-L: "A performance-driven standard-cell placer based on a modified force-directed algorithm". International Symposium on Physical Design 2001: p. 24-29, Sonoma County, CA, USA

CONG J. AND S. K. LIM, "Edge Separability-Based Circuit Clustering with Application to Multilevel Circuit Partitioning," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23(3), pp. 346-357, 2004, [S.l.].

COOK, S.A., “The Complexity of Theorem Proving Procedures”. Proceedings Third Annual ACM Symposium on Theory of Computing, May 1971, pp 151-158. , New York, NY, USA, 1971. ACM.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. (2001). “Introduction to Algorithms”. The Mit Press, 2nd edition. ISBN 0-262-03293-7.

DEMMEL J., KAGSTROM B. “Accurate solutions of ill-posed problems in control theory”. SIAM J. Matrix Anal. Appl., 9(1):126-145, 1988.

Demonstração de resultados da Intel - http://www.intc.com/financials.cfm - "Intel Reports Best Quarter Ever" > Acesso em nov/2010

EISENMANN, H., JOHANNES, F. "Generic global placement and floorplanning". In: DAC '98: Proceedings. ACM Press, 1998. p.269-274. San Francisco, California, USA

EISENMANN, H., JOHANNES, F. M., "Generic Global Placement and Floorplanning," In 35th Proceedings of the ACM/IEEE Design Automation Conference, pages 269-274, 1998. San Francisco, California, USA.

FIDUCCIA, C. M. AND MATTHEYSES, R. M. (1982). A linear-time heuristic for improving network partitions. Conference on Design Automation, 19th:175-181. Washington, U.S.A.

FLACH, G., PINTO, F., HENTSCHKE, R, REIS, R. "A novel cell placer based on Quadratic Placement and Simulated Annealing", 2006, Porto Alegre, RS, Brazil. Proceedings of 21 South Symposium on Microeletronics (SIM) 2006. Porto Alegre, RS, Brazil: UFRGS, 2006. p. 249-255. (ISBN: 85-7669-063-2)

HALPIN, B., CHEN, C.Y., AND SEHGAL, N., "Timing driven placement using physical net constraints," Proc. Design Automation Conference, pp.780-783, 2001. Las Vegas, Nevada, USA

HEATON REASEARCH - Exemplo visual de aplicação do algoritmo Simulated Annealing - http://www.heatonresearch.com/articles/64/page1.html > acesso em fev/2011

HENTSCHKE, R, PINTO, F., FLACH, G., REIS, R.; "Quadratic Placement for 3D Circuits Using Z-Cell Shifting, 3D Iterative Refinement and Simulated Annealing" ISVLSI '07. IEEE Computer Society Annual Symposium on March 2007 Page(s):67 – 72, Porto Alegre, RS, BRA.

HUANG, D. J., KAHNG, A. B.,. "Partitioning-Based Standard-Cell Global Placement with an Exact Objective," In ACM/IEEE ISPD, page 18-24, 1997. Napa Valley, CA, USA.

IBM - ISPD04, International Symposium of Physical Design - IBM Standard Cell Benckmarks with Pads. 2006. Phoenix, Arizona, USA

ISCAS `99- Benchmarks, 1999 - http://www.fm.vslib.cz/~kes/asic/iscas/ > Acesso em set/2010

JIANG, Z.-W., CHEN, T.-C., HSU, T.-C., CHEN, H.-C., AND CHANG Y.-W.. NTUplace2: A hybrid placer using partitioning and analytical techniques. In Proc. of ISPD, pages 215-217, 2006, San Jose, California, USA.

KAHNG, A. B., MANTIK S. AND MARKOV, I. L., ``Min-max Placement For Large-scale Timing Optimization" , in Proc. ACM/IEEE Intl. Symp. on Physical Design (ISPD), pp. 143-148, April 2002. Del Mar, CA, USA.

KAHNG, A., WANG, Q.. "Implementation and Extensibility of an Analytic Placer". Proc. ACM/IEEE Intl. Symp. on Physical Design, April, 2004. Phoenix, Arizona, USA

KARYPIS G. AND KUMAR V., "Multilevel k-way Hypergraph Partitioning," in Proc. ACM/IEEE Design Automation Conference, 1999, pp. 343-348, New Orleans, LA, USA.

KARYPIS G., AGGARWAL R., KUMAR V., AND SHEKHAR S.. “Multilevel Hypergraph Partitioning: Applications in VLSI Domain”. 34th Design and Automation Conference, pp. 526 - 529, 1997, Anaheim, California, USA.

KERNIGHAN, B. W. AND LIN, S. (1970). “An efficient heuristic procedure for partitioning graphs”. Bell System Technical Journal, 49:291-307.

KERSHAW, D. S. “The incomplete Cholesky conjugate gradient method for the iterative solution of systems of linear equations”. Journal of Computational Physics, Elsevier [S.l.], v.26, n.1, p.43..65, 1978.

LANDMAN, B. S., RUSSO, R. L., “On a Pin Versus Block Relationship For Partitions of Logic Graphs”, IEEE Trans. on Comput., col. C-20, pp. 1469-1479, 1971.[S. l.]

LIMA, C. P., FLACH, G. ª, PINTO, F., REIS, R. PlaceDL : uma nova técnica de espalhamento de células para posicionamento quadrático. In: Workshop Iberchip (15. : 2009 mar. : Buenos Aires, ARG). XV Workshop Iberchip. La Plata : Ediciones Científicas Americanas, c2009. p. 116-121

LUO, T., PAPA, D. A., LI , Z. , SZE, C. N. , ALPERT, C. J. , PAN, D. Z., Pyramids: an efficient computational geometry-based approach for timing-driven placement, Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, November 10-13, 2008, San Jose, CA, USA.

M.-C. KIM, D.-J. LEE, I. L. MARKOV, "SimPL: An Effective Placement Algorithm" in Proc. Int'l. Conf. on Computer-Aided Design (ICCAD), pp. 649-656, November 2010, San Jose, CA, USA.

MADDEN, H. P. "SuperSized VLSI: A Recipe for Disaster", EDP, April 05.

MAKHORIN A., GNU Linear Programming Kit, "http://www.gnu.org/software/glpk/", 2008 > Acesso em set/2010

MOORE G., "Cramming More Components onto Integrated Circuits" pp.4. Electronics Magazine, 1965. [S.l.]

MOTWANI, R. AND RAGHAVAN, P. (1995). Randomized Algorithms. Cambridge University Press, 1st edition.

OBERMEIER, B. AND JOHANNES, F. M.. Quadratic placement using an improved timing model. In Proc. Design Automation Conf., pages 705-710, 2004. San Diego, CA, USA.

OU, S-L, PEDRAM, M "Timing-driven Placement Based on Partitioning with Dynamic Cut-net Control", In ACM/IEEE Design Automation Conference, 2000, pp 472-476 Los Angeles, CA, USA.

PAGE, L., BRIN, S.: The anatomy of a large-scale hypertextual web search engine. Proceedings of the Seventh International Web Conference, pp. 107 - 117 (1998), Karlsruhe, ALE.

PINTO, F., FLACH, G., HENTSCHKE, R, REIS, R., "Z-Place Timing-Driven Approach"), 2007, Porto Alegre, RS, Brazil. Proceedings of 22 South Symposium on Microeletronics (SIM) 2007.

PINTO. F, CAVALHEIRO. L, REIS. R, JOHANN. M., "Logical Core Algorithm: Improving Global Placement," ISVLSI, pp.69-73, 2010 IEEE Annual Symposium on VLSI, 2010. Lixouri Kefalonia, GRE.

PINTO. F, CAVALHEIRO. L, REIS. R,A Global Critical Path Aware Placement Technique In: SBCCI - Symposium on Integrated Circuits and Systems Design, 2008, Gramado, RS - Brazil. Procedings of 21th annual conference on integrated circuits and systems design. , 2008.

QINGHUA L , MALGORZATA M-S, A study of netlist structure and placement efficiency, Proceedings of the 2004 international symposium on Physical design, April 18-21, 2004, Phoenix, Arizona, USA.

RIESS, B. M., ETTELT, G. G., "SPEED: Fast and Efficient Timing Driven Placement," In Proceedings of the IEEE International Symposium on Circus and System, pages 377-380, 1995. Seattle, WA , USA.

SARRAFZADEH, M, KNOL, D, TELLEZ, G. "A Delay Budgeting Algorithm Ensuring Maximum Flexibiligy in Placement", IEEE transactions on Computer-Aided Design of Intergrated Circuits and Systems. Vol 16 No11 Nov 1997 pages1332-1341. Evanston, IL

SCHULER, D. M. AND ULRICH, E. G., "Clustering and Linear Placement," in Proc. ACM/IEEE Design Automation Conference, 1972, pp. 50-56, San Jose, CA, USA.

SCHULER, D. M., "The Clustering of Interconnected Nodes," GTE Laboratories Technical Memorandum 70-468.1, December 1970, [S.l.].

SCHWEIKERT, D. G. and KERNIGHAN, B. W., A Proper Model for the Partitioning of Electrical Circuits, Proc. 9th Design Automation Workshop, pp. 57-62, 1972. Dallas, TX, USA.

SWARTZ, W., SECHEN C. "Timing Driven Placement for Large Standard Cell Circuits". In: DAC '95: Proceeding of the 32nd ACM/IEEE Conference on Design Automation, 1995, New York, NY, USA. ACM Press, 1995. p.211-215.

SYLVESTER, D., KEUTZER, K., "Getting to the Bottom of Deep Submicron", pp. 203-211, ICCAD 1998. Email: amit.chowdhary@intel.com

TAGHAVI, T, YANG, X., CHOI, B.-K., WANG, M., AND SARRAFZADEH. M., Dragon2006: Blockage-aware congestion-controlling mixed-size placer. In Proc. of ISPD, pages 209-211, 2006.

TAGHAVI, T.;YANG X.;CHOI B-K., WANG W., AND SARRAFZADEH M., "Dragon2005: Large Scale Mixed-Sized Placement Tool". International Symposium on Physical Design, ISPD Design Contest, pages. 42-47, IEEE, 2005.

VISWANATHAN, N, CHU, C. C.-N. "FastPlace: Efficient Analytical Placement using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model" IEEE Transactions Computer-Aided Design of Integrated Circuits and Systems, Vol 24, Issue 5, May 2005, [S. l.].

VISWANATHAN, N.; PAN, M.; CHU, C. C.-N. FastPlace: an analytical placer for mixed-mode designs. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. ACM Press, 2005. p.221-223.

WANG, M.;YANG, X.; SARRAFZADEH M.; "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits". International Conference on Computer-Aided Design. IEEE, November 2000.

WESTE, N. H. E. AND ESHRAGHIAN, K., Principles of CMOS VLSI Design: A System Perspective, Addison-Wesley, 2nd edition, 1993

YU, (KEVIN) C., Arizona State University- http://www.eas.asu.edu/~ptm/cgi-bin/test/nanocmos.cgi > Acesso em set/2010

"Posicionamento Visando Redução do Comprimento das Conexões"

por

Felipe de Andrade Pinto

Defesa de Dissertação apresentada aos Senhores:

Prof. Dr. Antônio Carlos Schneider Beck Filho (UFRGS)

Profª. Drª. Fernanda Gusmão de Lima Kastensmidt (UFRGS)

Prof. Dr. Sandro Sawicki(UNIJUI)

Vista e permitida a impressão.

Porto Alegre, 26/08/2011

Prof. Dr. Ricardo Augusto da Luz Reis – Orientador