UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

HENRIQUE PLÁCIDO

Posicionamento Global de Células em Circuitos VLSI

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação

Orientador: Prof. Dr. Ricardo Augusto da Luz Reis

Porto Alegre
junho de 2016

“Aqueles que se sentem satisfeitos sentam-se e nada fazem.
Os insatisfeitos são os únicos benfeitores do mundo.”

— WALTER S. LANDOR

AGRADECIMENTOS

Aos meus pais, pelo amor, dedicação, investimento e apoio que foram fundamentais para que eu chegasse até aqui.

Ao meu orientador, Ricardo Reis, pela orientação e por ter me proporcionado a oportunidade de trabalhar como bolsista IC no Grupo de Microeletrônica da UFRGS, o que me permitiu muitas aprendizagens na área de EDA (*Electronic Design Automation*).

Aos colegas do Grupo de Microeletrônica da UFRGS, Guilherme Flach, Jucemar Monteiro e Mateus Fogaça pelo auxílio durante o desenvolvimento desse projeto.

RESUMO

O posicionamento de células é a etapa da síntese automática de circuitos integrados responsável por determinar a localização das células na área do circuito. Como o posicionamento de células pertence a classe de problemas NP-Completo, é impossível obter a solução ótima em um tempo computacional razoável mesmo para instâncias com algumas poucas dezenas de células. Atualmente, os circuitos têm centenas de milhares a milhões de células, portanto, a medida que a tecnologia avança, torna-se imprescindível o desenvolvimento de novas heurísticas que sejam capazes de gerar soluções melhores, visto que a qualidade do posicionamento afeta fortemente o desempenho do *chip*, a área, consumo de energia e distribuição de calor. O problema de posicionamento é tradicionalmente dividido em três estágios: posicionamento global, legalização e posicionamento detalhado. O objetivo do posicionamento global é espalhar as células pela área do circuito enquanto otimiza o comprimento das conexões, congestionamento, entre outras métricas. Ainda, são permitidas sobreposições entre as células e não há a tentativa de alinhá-las nas bandas de posicionamento. Nesse trabalho é feito um estudo sobre o problema de posicionamento global e a implementação de uma ferramenta de posicionamento global. Para a implementação, foi proposto um algoritmo analítico quadrático que utiliza o modelo híbrido de conexões para construir o sistema linear, e para fazer o espalhamento das células pela área do circuito, foi utilizada a técnica *Look-Ahead Legalization* do algoritmo SimPL.

Palavras-chave: Microeletrônica. ferramentas de EDA. síntese física. posicionamento.

ABSTRACT

Placement is the step of the automatic synthesis of integrated circuits responsible for determining the location of cells in the circuit area. Since placement belongs to the class of NP-Complete problems, it is impossible to obtain the optimal solution in a reasonable computational time even for instances with a few dozen of cells. Nowadays, the circuits have hundreds of thousands to millions of cells, thus as the technology advances, it becomes necessary to develop new heuristics that are able to produce better solutions, since the quality of placement strongly affects the chip performance, area, power consumption and heat distribution. The placement problem is traditionally divided into three stages: global placement, legalization and detailed placement. The goal of global placement is to spread the cells of the chip area while optimizes the wirelength, congestion, among other metrics. Yet, overlaping among cells are allowed and there is not any attempt to align them on the placement rows. In this work, it is done a study about the global placement and the development of a global placement tool. It has been proposed an analytical quadratic algorithm that uses the hybrid net model to build the linear system and the Look-Ahead Legalization technique of SimPL to spread the cells on the chip area.

Keywords: Microelectronics. EDA tools. physical synthesis. placement.

LISTA DE ABREVIATURAS E SIGLAS

BB	Bounding Box
B2B	Bound to Bound
CAD	Computer-Aided-Design
CISC	Complex Instruction Set Computer
DRC	Design Rule Checking
E-LAL	Extended Look-Ahead Legalization
EBB	Explicit Bin-Blocking
EDA	Electronic Design Automation
FM	Fiduccia-Mattheyses
HDL	Hardware Description Language
HPWL	Half-Perimeter WireLength
ICCG	Incomplete Cholesky Conjugate Gradient
ILR	Iterative Local Refinement
KL	Kernighan-Lin
LAL	Look-Ahead Legalization
LVS	Layout Versus Schematics
ProLR	Progressive Local Refinement
RISC	Reducted Instruction Set Computer
RMST	Rectilinear Minimum Spanning Tree
RSMT	Rectilinear Steiner Minimum Tree
RTL	Register-Transfer-Level
SA	Simulated Annealing
STA	Static Timing Analysis
STST	Single-Trunk Steiner Tree

TDP Timing-Driven Placement

ULA Unidade de Lógica e Aritmética

VLSI Very-Large Scale Integration

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Em 1958, surgiu o primeiro circuito integrado graças ao trabalho de Jack Kilby, que, na época, trabalhava na Texas Instruments. Essa tecnologia permitiu a integração dos componentes de um circuito eletrônico em um mesmo substrato de material semicondutor (RABAEY; CHANDRAKASAN; NIKOLIC, 2003). Desde então, a indústria eletrônica avançou enormemente, tanto que, no fim da década de 80, já haviam *chips* com mais de um milhão de transistores, como, por exemplo, o processador Intel 486 DX (INTEL, 2008). Devido a essa evolução, os projetos de circuitos integrados ficaram complexos. Somada à grande complexidade, o *time-to-market*, isto é, o tempo desde a concepção de um produto até a sua venda, é outro fator crucial, pois, para o produto ter sucesso comercial, a empresa deve lançá-lo no menor tempo possível. Por essas razões, os engenheiros aderiram ao uso de ferramentas de CAD (*computer-aided-design*) (RABAEY; CHANDRAKASAN; NIKOLIC, 2003). Essa ferramentas, também conhecidas como ferramentas de EDA (*Electronic Design Automation*), automatizam as etapas de desenvolvimento de um *chip*.

Para a criação de circuitos integrados, surgiu inicialmente o estilo de projeto chamado de *full-custom*. Nesse estilo, os projetistas desenham o leiaute completo do *chip* transistor a transistor respeitando as regras da tecnologia a ser utilizada, nas quais estão previstas restrições elétricas e topológicas. Apesar desse estilo permitir ao projetista uma melhor exploração do espaço de soluções, de modo que os requisitos de projeto sejam satisfeitos, o aumento da complexidade dos projetos fez surgir a necessidade de explorar o uso de uma hierarquia de projeto, de modo que era empregada uma composição ascendente (*bottom-up*) para o desenvolvimento do *chip*. Assim, ao invés de fazer um desenho de todo o leiaute do circuito integrado, desenham-se leiautes de blocos lógicos básicos, como, por exemplo, as funções lógicas *or, and, xor, not*, entre outras. Os leiautes desse blocos, conhecidos como células-folha, são usados para compor blocos mais complexos, dessa forma, os blocos gerados vão sendo agregados para formar componentes ainda mais complexos, até chegar no circuito final, que representa o topo na composição ascendente. Devido a maior complexidade, o estilo *full-custom* ficou restrito a projetos nos quais o circuito deve apresentar o máximo desempenho (REIS et al., 2002).

Então percebeu-se a importância de utilizar, em novos projetos, as células-folhas previamente geradas em projetos anteriores, pois isso implicava uma redução do esforço de geração de leiaute. Além disso, por serem células já usadas em outros projetos, elas

estavam devidamente validadas e documentadas. No entanto, apena a reutilização de células ainda não era o suficiente, ou seja, era preciso automatizar a geração do leiaute. Para isso, algumas estratégias passaram a ser observadas a fim de permitir o desenvolvimento de ferramentas de EDA capazes de criar um leiaute a partir das células. As estratégias adotadas incluíam fixar uma das dimensões das células da biblioteca, normalmente a altura. Assim, originou-se o estilo de projeto baseado em células-padrão, conhecido como *standard-cells*. Nesse estilo, há uma biblioteca que disponibiliza uma variedade de células para a geração do leiaute do circuito integrado. Assim, diminui-se o tempo de desenvolvimento do *chip*, visto que, ao utilizar uma biblioteca de células comercial, não há a necessidade de empregar esforço no desenho de leiautes. Além da rapidez do projeto, esse estilo permite também uma maior confiabilidade, pois, como citado anteriormente, as células disponibilizadas são validadas e documentadas (REIS et al., 2002). No contexto de *standard-cells*, as ferramentas de EDA são basicamente responsáveis por posicionar as células do circuito e realizar as conexões entre elas. A etapa de posicionamento visa encontrar uma posição para cada célula de modo que alguma métrica seja otimizada, como, por exemplo, comprimento de fios. O posicionamento é dividido em três estágios: posicionamento global, legalização e posicionamento detalhado (KAHNG et al., 2011). Nesse trabalho, que está focado no estilo *standard-cells*, foi desenvolvida uma ferramenta de EDA para automatizar o estágio de posicionamento global.

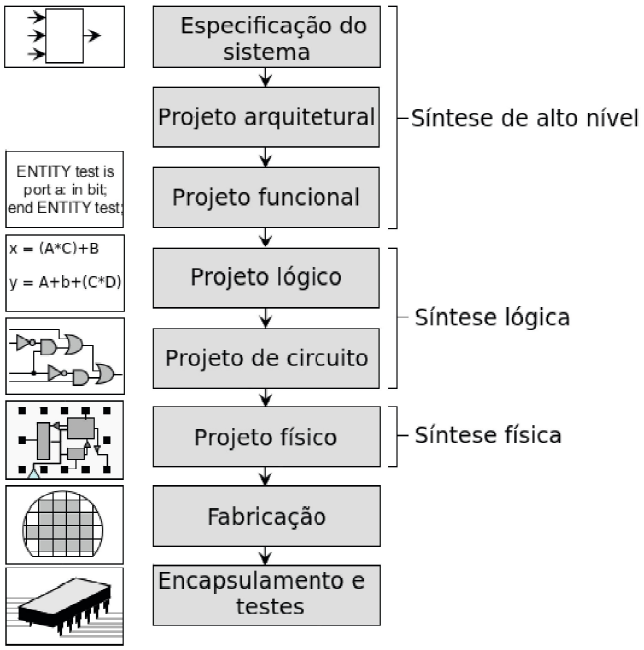
1.1 Fluxo de Projeto Baseado em Biblioteca de Células Padrão

O fluxo de projeto de um circuito integrado é uma sequência de passos que permite o desenvolvimento do *chip*, desde a sua especificação até a fabricação. Segundo (SHERWANI, 1999), um fluxo de projeto típico *standard-cell* é formado pelas etapas de especificação do sistema, projeto arquitetural, projeto funcional, projeto lógico, projeto de circuito, projeto físico (síntese física), fabricação, encapsulamento e testes. Os passos de um fluxo típico estão ilustrados na Figura 1.1.

1.1.1 Síntese de Alto Nível

A primeira etapa é a especificação do sistema, que é uma representação em alto nível do sistema a ser desenvolvido. Nessa etapa, são especificados os objetivos e requeri-

Figura 1.1: Fluxo de projeto típico



Fonte: adaptado de (KAHNG et al., 2011).

mentos do sistema, como performance, funcionalidade e dimensões físicas do *chip*. Além disso, também é definida qual será a tecnologia de fabricação que será utilizada.

No projeto arquitetural, é estabelecida uma arquitetura básica do sistema para que os requerimentos estabelecidos na etapa anterior sejam satisfeitos. Nessa etapa, os projetistas decidem, por exemplo, caso um processador esteja sendo desenvolvido, se será RISC (*Reduced Instruction Set Computer* - Processador com conjunto de instruções reduzido) ou CISC (*Complex Instruction Set Computer* - Processador com conjunto de instruções complexo), o número de ULAs (unidade de lógica e aritmética) e de unidades de ponto flutuante, a estrutura e o número de *pipelines*, níveis e quantidade de memória *cache*, entro outros. O resultado dessa etapa é uma especificação micro-arquitetural, que é uma descrição textual das decisões tomadas.

Durante o projeto funcional, também conhecido como projeto comportamental, é criada uma descrição de alto nível do comportamento do sistema a fim de analisá-lo em termos de sua funcionalidade e performance. Para criar a descrição, são utilizadas HDLs (*Hardware Description Language* - Linguagem de descrição de hardware) (PALNITKAR, 1996), tais como VHDL e Verilog. O ponto principal dessa etapa é definir o comportamento do sistema sem levar em consideração detalhes de implementação. Por exemplo, durante essa etapa pode ser especificado que é necessário realizar uma multiplicação, mas não será detalhado como ela será executada. Ou seja, a ideia é especificar o comportamento em termos das entradas, saídas e o comportamento temporal de cada

módulo (SHERWANI, 1999).

1.1.2 Síntese Lógica

Como a descrição comportamental criada na etapa de projeto funcional é de alto nível, pode não ser possível sintetizar um circuito com portas lógicas diretamente a partir dela. Por isso, durante a síntese lógica, que é formada pelas etapas de projeto lógico e projeto de circuito, a descrição é convertida em um *netlist*, que é uma estrutura que contém os sinais (interconexões) e os elementos do circuito que estão conectados a esses sinais (KAHNG et al., 2011). Na etapa de projeto lógico, a primeira de nível lógico, a descrição comportamental é traduzida em uma descrição RTL (*Register-Transfer-Level*), que descreve o sistema em termos dos registradores, fluxo de sinais entre eles e as operações lógicas realizadas nos sinais (CHURIWALA; GARG, 2011). Ainda nessa etapa, ferramentas de síntese lógica convertem a descrição RTL em um circuito com portas lógicas genéricas, de modo que as expressões booleanas derivadas são minimizadas a fim de se obter um circuito otimizado.

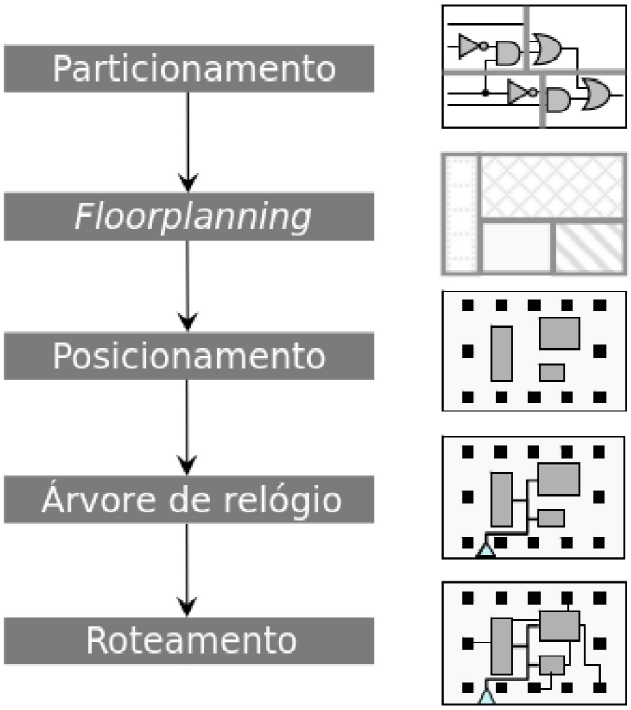
Na etapa de projeto de circuito, a última de nível lógico, as portas lógicas do circuito otimizado obtido na etapa de projeto lógico são mapeadas em portas lógicas presentes na biblioteca de células adotada para o projeto. Esse processo de conversão é feito por uma ferramenta de mapeamento tecnológico (GEREZ, 1999). O resultado final é um *netlist* mapeado e implementável na tecnologia estabelecida na especificação do projeto.

1.1.3 Síntese Física

O projeto físico, ou síntese física, é responsável por gerar um leiaute do *chip* a partir do *netlist* do circuito. Como a síntese física é um processo complexo, ela é, segundo (KAHNG et al., 2011), subdividida nas etapas de particionamento, *floorplanning*, posicionamento, síntese da árvore de relógio e roteamento. A Figura 1.2 apresenta o fluxo de síntese física.

Um *chip* pode ter milhões de portas lógicas. Então, devido a limitações de poder de processamento e memória, poderia não ser possível gerar o leiaute do circuito considerando todas as portas lógicas durante a execução da síntese física. Por isso, na etapa de particionamento, o *chip* é particionado em sub-circuitos (blocos), de modo que o problema

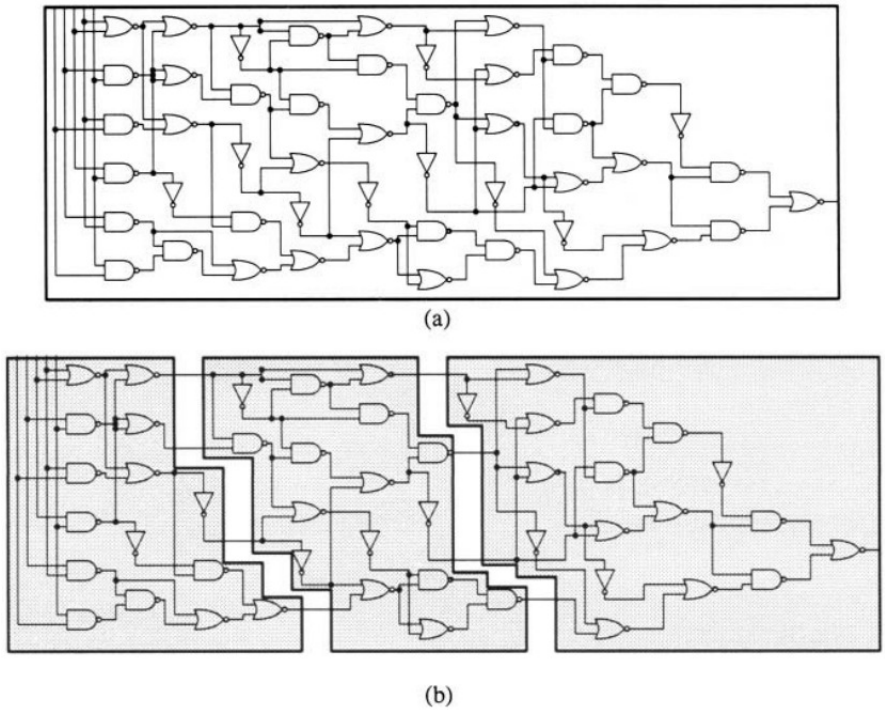
Figura 1.2: Fluxo típico de síntese física



Fonte: adaptado de (KAHNG et al., 2011).

seja reduzido em problemas menores e independentes. Além disso, particionar o circuito também ajuda a obter resultados melhores seguindo a filosofia dividir-para-conquistar. Um exemplo de particionamento pode ser visualizado na Figura 1.3, que ilustra um circuito que foi dividido em três partições. Segundo (KAHNG et al., 2011), uma forma bem comum de particionar o circuito é criar sub-circuitos utilizando como critério a minimização do número de conexões entre as partições. Um exemplo de algoritmo baseado nessa ideia é o Kernighan-Lin (KL) (KERNIGHAN; LIN, 1970), que particiona o *netlist* em dois sub-circuitos.

Figura 1.3: Exemplo de particionamento

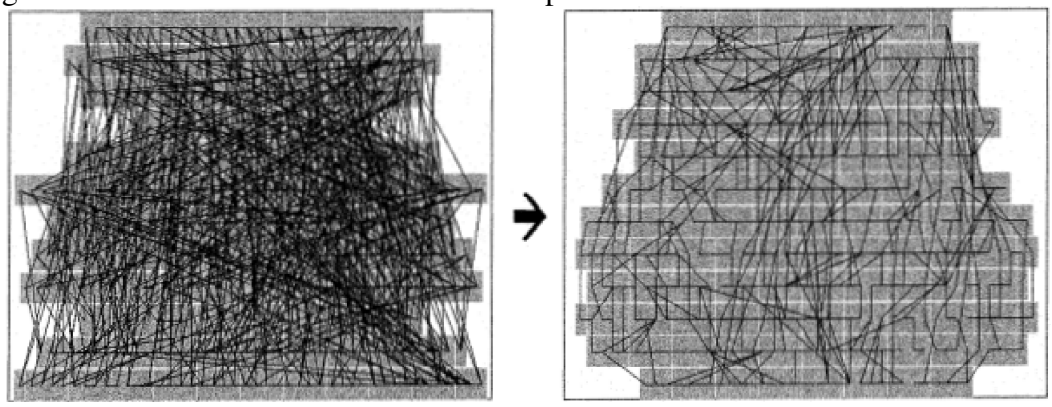


Fonte: (SHERWANI, 1999).

A etapa seguinte ao particionamento, o *floorplanning*, é responsável por determinar a localização e as dimensões dos blocos criados anteriormente, além de determinar as posições dos *pads* do *chip* e de módulos maiores, como memórias, por exemplo.

Após particionar o circuito em módulos menores e determinar a localização de cada um deles durante o *floorplanning*, é executado o posicionamento, que é responsável por determinar a localização de cada uma das células pertencentes a cada uma das partições. A etapa de posicionamento é crucial para a síntese física, pois um posicionamento de baixa qualidade pode impossibilitar o roteamento e, além disso, gerar uma solução que não satisfaça os critérios de desempenho estabelecidos para o projeto (SHERWANI, 1999). A Figura 1.4 ilustra um exemplo de um posicionamento aleatório inicial e o resultado obtido ao posicionar o circuito utilizando o método *simulated annealing* (SA). No primeiro posicionamento, é possível observar que há um grande número de conexões congestionadas, o que torna o roteamento muito difícil. Já no posicionamento feito pelo algoritmo de SA, o congestionamento foi drasticamente reduzindo.

Figura 1.4: Posicionamento aleatório versus posicionamento com *simulated-annealing*



Fonte: adaptado de (HENTSCHKE; JOHANN; REIS, 2005).

Depois do posicionamento, a etapa de roteamento determina as rotas dos fios que conectam os elementos do circuito. Os caminhos gerados devem respeitar as regras de projeto a fim de garantir que o *chip* possa ser fabricado. O objetivo mais importante dessa etapa é rotear 100% dos fios, pois, caso contrario, o circuito não irá funcionar. Outro importante objetivo é a minimização do comprimento dos fios, embora o posicionamento seja o maior definidor do comprimento de fios do circuito, pois os atrasos dos fios não podem comprometer os requisitos de desempenho. O roteamento é usualmente dividido em dois estágios: o roteamento global e o roteamento detalhado. O roteamento global particiona a área do circuito em regiões de roteamento e então atribui a cada uma delas as conexões, de modo que não seja ultrapassado o limite da capacidade de roteamento de cada região. O roteamento detalhado é responsável por determinar os caminhos exatos das conexões (KAHNG et al., 2011).

Em circuitos com elementos síncronos, como *flip-flops*, é necessária a etapa de síntese de árvore de relógio, que é responsável por garantir que o sinal de *clock* chegue simultaneamente nesses elementos dentro do tempo que satisfaça os requisitos de performance do circuito (KAHNG et al., 2011).

O leiaute criado durante a síntese física precisa passar por um estágio de verificação a fim de validar a solução desenvolvida. Nesse estágio, um dos processos executados é o DRC (*Design Rule Checking*), que é responsável por verificar se o leiaute obedece as regras impostas pelo processo de fabricação que será utilizado. Um exemplo desse tipo de verificação seria a análise das distâncias entre as interconexões. Nesse caso, o DRC precisa verificar se os fios estão separados por uma distância mínima que é determinada por uma regra de projeto. Outro processo executado durante a verificação é o LVS (*Layout Versus Schematics*), que, a partir do leiaute, gera um *netlist* e o compara com o *netlist* original (gerado pela síntese lógica), averiguando, assim, a equivalência entre eles.

Ainda, é feito um processo de extração de efeitos parasitas, que obtém as resistências, capacitâncias e indutâncias do leiaute.

É importante destacar que, atualmente, as etapas de síntese lógica e síntese física não são mais distintas como o fluxo apresentado, que separa didaticamente as etapas do projeto. Antigamente, os atrasos dos circuitos eram determinados pelos atrasos das portas lógicas. Mas hoje, devido a redução das dimensões dos transistores, as capacitâncias de carga e resistências foram drasticamente reduzidas, diminuindo os atrasos das portas lógicas. Por outro lado, o encolhimento das interconexões aumenta a resistência delas devido à redução da seção transversal. Além disso, a capacitância dos fios também aumenta devido a redução do espaçamento entre eles. Assim, os atrasos das conexões se tornaram um fator limitante no desempenho do circuito (ROSSUM, 2009). Por essa razão, a síntese lógica passou a utilizar informações físicas e a síntese física passou a chamar métodos de otimização que antes eram utilizados somente durante a síntese lógica (FLACH, 2015). Um exemplo dessa interação entre as sínteses lógica e física é o algoritmo de posicionamento proposto em (STENZ et al., 2000). O algoritmo parte do princípio de que a síntese lógica não tem informações do leiaute do circuito, o que torna as estimativas de atrasos dos fios imprecisas. Então, a partir de um circuito com as células previamente posicionadas, o algoritmo determina os tempos de propagação dos sinais nos fios do circuito e utiliza esses dados para aplicar transformações no *netlist* original a fim de reduzir os atrasos nos caminhos. Depois que o *netlist* é alterado, as células são reposicionadas, visto que, durante o processo de transformação, podem ser removidas ou inseridas algumas portas lógicas.

1.2 Objetivos do Trabalho

Nesse trabalho, é implementado um algoritmo de posicionamento global de células em circuitos VLSI. Devido às diversas abordagens que podem ser encontradas na literatura, os objetivos são:

- Entender o problema de posicionamento de células em circuitos integrados e os impactos dessa etapa no fluxo do projeto.
- Fazer uma revisão bibliográfica dos algoritmos estado-da-arte para posicionamento global.
- Implementar um algoritmo de posicionamento global analítico, quadrático baseado

em técnicas propostas em (VISWANATHAN; CHU, 2004) e (KIM; LEE; MARKOV, 2012).

1.3 Organização do Texto

Esse trabalho está organizado em 6 capítulos. No próximo capítulo, é introduzido o problema de posicionamento de células em circuitos integrados, apresentando os conceitos básicos, objetivos de otimização, técnicas de estimativa de comprimento de fio e etapas do posicionamento. O capítulo 3 discute com detalhes a etapa de posicionamento global, apresentando as classes de algoritmos que podem ser encontradas na literatura. Ainda neste capítulo, são apresentados os modelos de redes utilizados para modelar as conexões em posicionadores analíticos quadráticos. No capítulo 4, é feita uma revisão bibliográfica sobre algoritmos de posicionamento global, apresentando o estado-da-arte dos posicionadores analíticos quadráticos. O capítulo 5 apresenta a ferramenta de posicionamento global desenvolvida nesse trabalho e os resultados obtidos com ela. Por fim, no capítulo 6, é feita a conclusão do trabalho.

2 POSICIONAMENTO DE CÉLULAS EM CIRCUITOS INTEGRADOS

O posicionamento de células é considerado um dos problemas mais importantes da etapa de síntese física de circuitos integrados. Embora esteja sendo estudado há décadas, ele ainda é um tema amplamente explorado devido às novas restrições e desafios que o avanço tecnológico impõe (LIN et al., 2013). O posicionamento de células é um problema NP-Completo (SHERWANI, 1999), então, os algoritmos propostos são heurísticas que visam obter boas soluções em um tempo computacional limitado.

O objetivo do posicionamento é determinar onde cada célula do circuito será posicionada dentro dos limites do leiaute do *chip*. O posicionamento é guiado por objetivos a serem otimizados, como, por exemplo, minimização do comprimento total de fios. Além disso, está sujeito às restrições de projeto, como não permitir sobreposição entre as células e alinhá-las nas bandas de posicionamento (KAHNG et al., 2011).

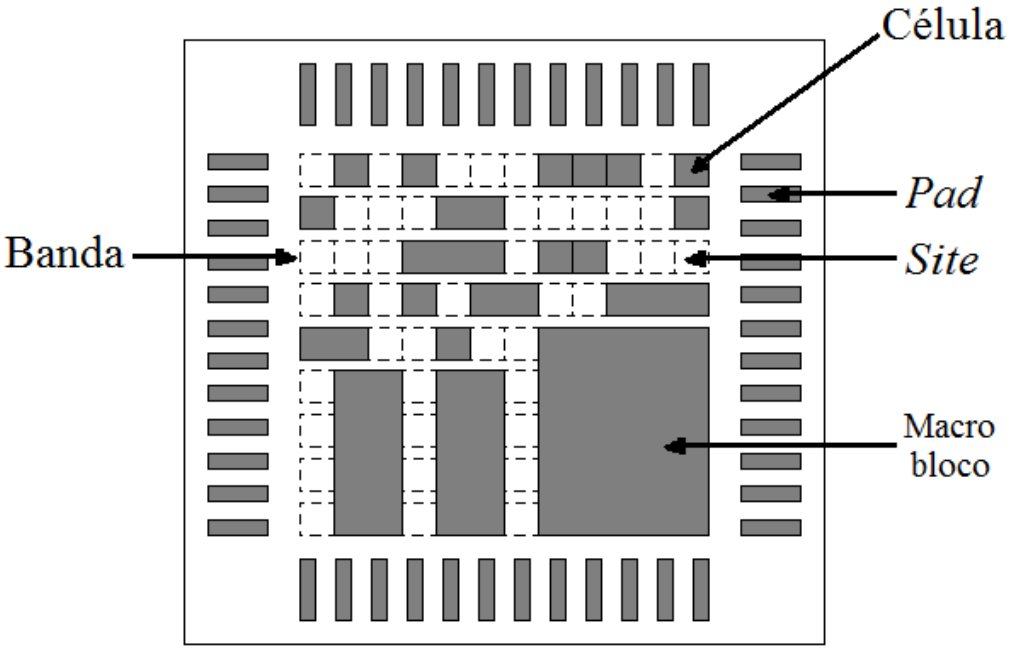
O leiaute de um *chip*, como exemplificado na Figura 2.1, é formado por bandas de posicionamento (*rows*) de altura fixa, e, entre elas, pode haver ou não um espaço. Cada banda de posicionamento consiste em um número de *sites*, que são os menores espaços que os componentes do circuito podem ocupar. Os componentes do circuito são as células, os macro blocos e os *pads* de entrada e saída. As células apresentam uma altura fixa, que corresponde à altura das bandas de posicionamento, mas têm largura variável, que é múltipla da largura de um *site*. Já os macro blocos são tipicamente muito maiores que as células e apresentam altura e largura variáveis, por isso, podem ocupar mais de uma banda (LAVAGNO; SCHEFFER; MARTIN, 2006). A Figura 2.1 ilustra esses conceitos.

O posicionamento deve ser roteável (KAHNG et al., 2011), isto é, o leiaute resultante deve permitir que todas as redes de interconexões possam ser roteadas, caso contrário, seria impossível fabricar o *chip*. Cabe ressaltar ainda que a qualidade de um circuito integrado está fortemente relacionada com a qualidade do resultado do posicionamento (SPINDLER; JOHANNES, 2007).

O posicionamento de células começou a ser pesquisado nos anos de 1960, quando, na indústria, foram propostos os primeiros métodos de particionamento do *netlist*, que acabaram motivando melhorias nas heurísticas de particionamento de grafos. No início dos anos 1980, surgiram os posicionadores analíticos, mas foram substituídos pelos algoritmos baseados em *Simulated Annealing* (SA), pois esses apresentavam resultados melhores. Os algoritmos baseados em *Simulated Annealing* dominaram a indústria por uma década, porém, em meados dos anos 1990, eles deixaram de ser escaláveis devido

ao aumento da complexidade dos novos projetos. Então, a partir de 2005, ocorreu o retorno das técnicas de posicionamento analítico, pois elas maturaram a ponto de gerarem melhores resultados em relação às técnicas anteriores, e, também, serem mais escaláveis (MARKOV; HU; KIM, 2012).

Figura 2.1: Representação de banda, *site*, célula e *pad*



Fonte: figura elaborada pelo autor.

2.1 Métricas de Otimização

O posicionamento das células é guiado pela otimização de uma métrica. Algumas métricas normalmente adotadas, que podem ser encontradas em (KAHNG et al., 2011) e (WANG; CHANG; CHENG, 2009), são o comprimento total dos fios, congestionamento de interconexões, consumo de energia, distribuição de calor e análise de atrasos (*timing*).

2.1.1 Comprimento Total de Fios

O comprimento total dos fios é a métrica a ser minimizada que normalmente é a mais utilizada. A minimização do comprimento total de fios indiretamente minimiza os atrasos do circuito, pode tornar o roteamento dos fios mais fácil e pode reduzir o consumo de energia do *chip* (WANG; CHANG; CHENG, 2009).

Pesos podem ser utilizados para priorizar algumas redes dentre outras. Segundo

(MONTEIRO, 2014), os critérios para priorizar uma rede podem ser em função delas estarem em uma região do circuito em que há congestionamento, violações no tempo de propagação dos sinais, pontos de aquecimento (hot spot), etc. Para um posicionamento P, uma estimativa de comprimento total de fios com redes ponderadas pode ser expressa pela Equação 2.1:

$$L(P) = \sum_{rede \in P} W(rede) \times L(rede) \tag{2.1}$$

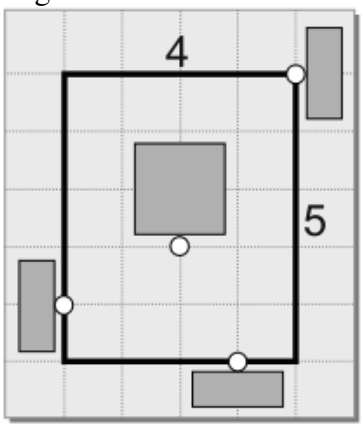
onde $W(rede)$ é o peso da rede e $L(rede)$ é uma estimativa do comprimento de fio da mesma (KAHNG et al., 2011).

É custoso fazer, durante o posicionamento, uma predição do comprimento total de fios que o *chip* teria após a etapa de roteamento (WANG; CHANG; CHENG, 2009). Dessa forma, algumas abordagens são utilizadas para fazer uma estimativa dessa métrica. Uma consideração importante para fazer a estimativa do comprimento total de fios durante o posicionamento é a rapidez com a qual uma estimativa pode ser computada (KAHNG et al., 2011). A seguir, serão discutidos alguns dos modelos utilizados para estimar o comprimento total de fios.

2.1.1.1 Estimativa de Comprimento de Fio com Half-Perimeter WireLength (HPWL)

O modelo *Half-Perimeter Wirelength* (HPWL) é normalmente utilizado porque é capaz de estimar o comprimento de fio de uma rede com uma precisão razoável e é rápido e fácil de ser computado. O comprimento de fio de uma rede é estimado computando o semi-perímetro do menor retângulo que envolve todos os pinos da rede, tal que os limites do retângulo correspondam às posições dos pinos mais externos da rede (KAHNG et al., 2011). Como pode ser visto na Figura 2.2, as dimensões do menor retângulo que envolve os pinos da rede são 4 e 5. Portanto, a estimativa HPWL, nesse caso, é 9, que é a metade do perímetro do retângulo.

Figura 2.2: Modelo HPWL



Fonte: (KAHNG et al., 2011).

2.1.1.2 Estimativa de Comprimento de Fio com Clique

No modelo Clique, cada um dos pinos de uma rede está conectado com todos os outros pinos da mesma rede, ou seja, para uma rede com p pinos, cada pino tem $p-1$ conexões (KAHNG et al., 2011). Para esse modelo, comprimento de fio de uma rede é expresso pela Equação 2.2:

$$L(rede) = \frac{2}{p} \sum_{e \in clique} d_M(e) \tag{2.2}$$

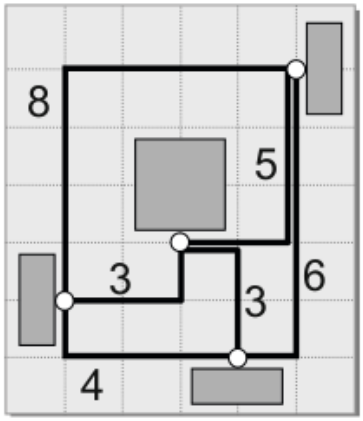
onde e é uma aresta no modelo clique e $d_M(e)$ é a distância Manhattan entre os pinos da aresta e. A distância Manhattan d_M entre dois pontos p_1 e p_2 é expressa pela Equação 2.3:

$$d_M(p_1, p_2) = |x_2 - x_1| + |y_2 - y_1| \tag{2.3}$$

onde (x_1, y_1) e (x_2, y_2) correspondem à localização dos pontos p_1 e p_2, respectivamente (KAHNG et al., 2011).

No exemplo ilustrado na Figura 2.3, há uma rede com 4 pinos e a distância Manhattan entre cada um deles está indicada na aresta que conecta cada um dos pares. Pela Equação 2.2, o comprimento total de fio para essa rede é 14,5.

Figura 2.3: Modelo Clique

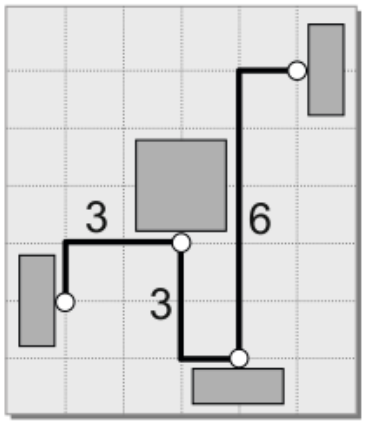


Fonte: (KAHNG et al., 2011).

2.1.1.3 Estimativa de Comprimento de Fio com Cadeia Monotônica

O modelo de cadeia monotônica conecta os pinos de uma rede usando uma topologia de cadeia, ou seja, eles são conectados sequencialmente. Cada pino que está em um dos extremos da cadeia tem grau um e os intermediários têm grau dois. Encontrar o caminho de menor comprimento corresponde ao problema de busca de caminho Hamiltoniano, que é NP-Difícil. Por isso, nesse modelo, os pinos são ordenados usando as suas coordenadas x ou y e, então, são conectados sequencialmente respeitando a ordenação obtida. Apesar de simples, esse método superestima o comprimento de fio real (KAHNG et al., 2011), isto é, o comprimento de fios após a etapa de roteamento. No exemplo da Figura 2.4, o comprimento de fio estimado é 12, visto que esse resultado corresponde ao somatório dos valores (distância Manhattan) das arestas que conectam os pinos da topologia.

Figura 2.4: Modelo Cadeia Monotônica

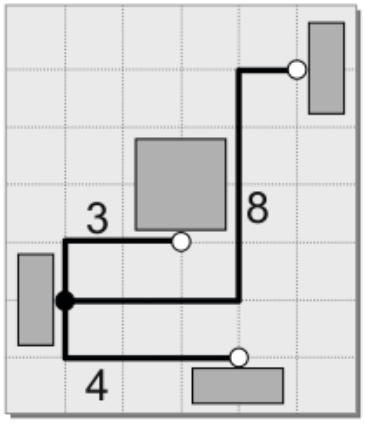


Fonte: (KAHNG et al., 2011).

2.1.1.4 Estimativa de Comprimento de Fio com Modelo Estrela

O modelo estrela considera um dos pinos da rede como o nodo fonte e os outros como destino. Então, é criada uma conexão entre o pino fonte e cada pino destino. Esse modelo é especialmente útil para otimização de *timing*, visto que é possível capturar a direção de propagação de um sinal a partir de um pino de saída para um ou mais pinos de entrada. Para esse modelo são necessárias somente $p - 1$ arestas, sendo p o número de pinos da rede. Isso pode ser vantajoso para modelar redes com um número elevado de pinos. No entanto, o modelo estrela superestima o comprimento de fio real. Na Figura 2.5, o comprimento de fio estimado para a rede é 15, que é o valor do somatório das distâncias Manhattan entre o pino fonte e cada um dos pinos destino.

Figura 2.5: Modelo Estrela

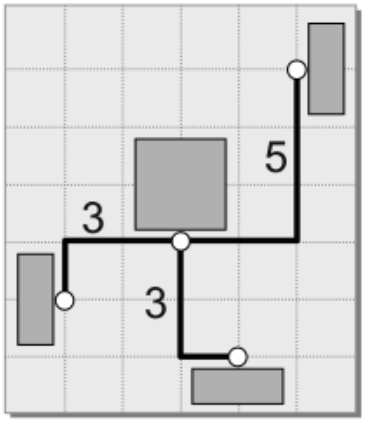


Fonte: (KAHNG et al., 2011).

2.1.1.5 Estimativa de Comprimento de Fio com Rectilinear Minimum Spanning Tree (RMST)

O modelo rectilinear minimum spanning tree (RMST) decompõe uma rede com p pinos em conexões entre dois pinos e os conecta com $p - 1$ conexões. Algoritmos de RMST podem explorar a geometria de Manhattan para atingir um tempo de execução de complexidade $O(p\ log\ p)$ (KAHNG et al., 2011). A Figura 2.6 apresenta uma rede em que foi aplicado um algoritmo de RMST para estimar o comprimento de fio. Somando as distâncias Manhattan indicadas nas arestas que conectam os pinos, o comprimento de fio estimado para essa rede é de 11.

Figura 2.6: Modelo RMST

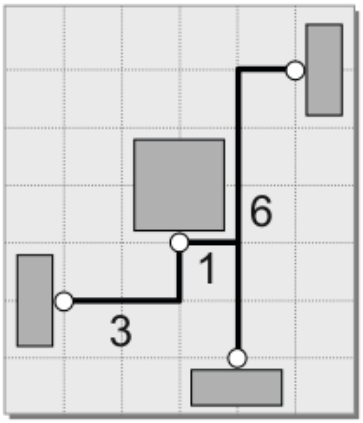


Fonte: (KAHNG et al., 2011).

2.1.1.6 Estimativa de Comprimento de Fio com Rectilinear Steiner Minimum Tree (RSMT)

No modelo rectilinear Steiner minimum tree (RSMT), Figura 2.7, todos os p pinos de uma rede estão conectados e ainda podem estar ligados a $p - 2$ pontos adicionais de Steiner. Encontrar um conjunto ótimo de pontos de Steiner para um conjunto arbitrário de pontos é um problema NP-Dífícil. Se os pontos de Steiner são conhecidos, uma RSMT pode ser encontrada construindo uma RMST sobre a união do conjunto original de pontos com o conjunto de pontos de Steiner adicionados (KAHNG et al., 2011).

Figura 2.7: Modelo RSMT



Fonte: (KAHNG et al., 2011).

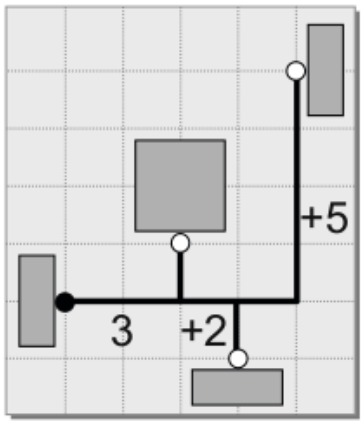
2.1.1.7 Estimativa de Comprimento de Fio com Rectilinear Steiner Arborescence (RSA)

O modelo rectilinear Steiner arborescence (RSA),Figura 2.8, de uma rede com p pinos é uma árvore onde um nodo fonte s_0 está conectado a $p-1$ nodos destino. No RSA, o comprimento do caminho a partir de s_0 até qualquer um dos nodos destino s_i, $1 \leq i < p-1$, deve ser igual a distância Manhattan entre s_0 e s_i (KAHNG et al., 2011). Ou seja, para todos os destinos s_i

$$L(s_0, s_i) = d_M(s_0, s_i) \tag{2.4}$$

onde $L(s_0, s_i)$ é o comprimento do caminho a partir de s_0 até s_i na árvore. Computar o caminho mínimo em RSA é um problema NP-Difícil (KAHNG et al., 2011).

Figura 2.8: Modelo RSA

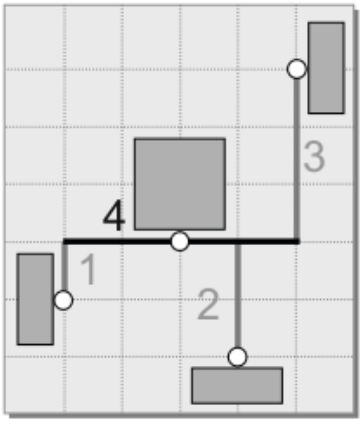


Fonte: (KAHNG et al., 2011).

2.1.1.8 Estimativa de Comprimento de Fio com Single-Trunk Steiner Tree (STST)

O modelo single-trunk Steiner tree (STST), Figura 2.9, consiste de um segmento vertical ou horizontal (*trunk*) e da conexão de todos os pinos a esse segmento utilizando segmentos horizontais ou verticais. STSTs são usualmente utilizadas para estimação devido a facilidade de serem construídas (KAHNG et al., 2011).

Figura 2.9: Modelo STST



Fonte: (KAHNG et al., 2011).

2.1.2 Congestionamento de Interconexões

O congestionamento pode ser definido pela relação de demanda de conexões com a oferta de espaço para elas (HENTSCHKE, 2002). Uma região com alto congestionamento pode levar a desvios de roteamento excessivos, ou, até mesmo, impossibilitar o roteamento. Os desvios de roteamento podem implicar um aumento excessivo do comprimento de fios , aumentado os atrasos e consumo de energia (LAVAGNO; SCHEFFER; MARTIN, 2006).

Fazer uma estimativa do congestionamento e minimizá-lo durante a etapa de posicionamento facilitaria a tarefa do roteador, visto que ele poderia dedicar mais tempo ao objetivo de fazer as conexões dos pinos. Além disso, o resultado do roteamento tende a ser melhor (MONTEIRO, 2014). Métodos de minimização de congestionamento tipicamente começam dividindo a área do leiaute em regiões retangulares. Então, é utilizado um método de roteamento global rápido para fazer uma estimativa do número de fios que cruzam cada fronteira das regiões. Essa estimativa é comparada com a oferta de recursos de roteamento disponível. Se a demanda excede a quantidade de recursos, então a região é expandida, ou as células são movidas para outra região a fim de diminuir a demanda de recursos de roteamento (LAVAGNO; SCHEFFER; MARTIN, 2006).

2.1.3 Consumo de Energia

Dispositivos móveis são alimentados por uma bateria, então a minimização do consumo de energia é um importante objetivo a ser considerado (LAVAGNO; SCHEF-

FER; MARTIN, 2006).

A capacitância de uma rede de interconexão é proporcional ao seu comprimento. Então, o problema de minimização de consumo de energia pode ser formulado como um problema de minimização de comprimento de fio. Visto que o consumo de energia também está relacionado ao chaveamento dos transistores, as redes devem ser ponderadas segundo o grau de atividade dos componentes dessa rede (WANG; CHANG; CHENG, 2009).

2.1.4 Distribuição de Calor

Uma distribuição não homogênea de temperatura na área do *chip* pode afetar as características de circuitos sensíveis à temperatura. Pode levar, também, a problemas de confiabilidade. Então, é desejável fazer um posicionamento adequado de elementos que geram calor para que não existam pontos com uma temperatura muito mais elevada que as demais partes do *chip* (WANG; CHANG; CHENG, 2009).

Fazer um posicionamento orientado à temperatura é difícil por algumas razões. Primeiro, a geração de calor em cada um dos elementos muda com o tempo, pois depende das operações executadas no circuito. Segundo, além dos transistores, os fios também geram calor. Durante o posicionamento, é difícil fazer uma predição do calor que será gerado pelos fios. Terceiro, o perfil de temperatura do *chip* é determinado pela geração e transferência de calor e é difícil fazer uma aproximação sem o uso de de uma simulação demorada (WANG; CHANG; CHENG, 2009).

Uma solução prática para um posicionamento orientado a temperatura é distribuir a geração média de calor dos módulos de forma uniforme na área do *chip* (assumindo que a natureza dinâmica da geração de calor, geração de calor nos fios e transferência de calor são efeitos secundários) (WANG; CHANG; CHENG, 2009).

2.1.5 Análise de Atrasos (*Timing*)

O comprimento total de fios afeta a frequência máxima que o *clock* do sistema pode atingir, visto que ele depende dos atrasos das portas lógicas e dos fios. Em tecnologias mais antigas, os atrasos eram determinados majoritariamente pelas portas lógicas, mas, atualmente, os atrasos dos fios contribuem significativamente com o atraso total de

um determinado caminho (KAHNG et al., 2011).

O posicionamento orientado a performance (*Timing-driven placement* (TDP)) otimiza os atrasos do circuito tanto para satisfazer as restrições temporais quanto para que a frequência do clock seja a maior possível. Os atrasos do circuito são usualmente verificados utilizando a análise estática do tempo de propagação (*Static Timing Analysis* (STA)), que se baseia em estimativas dos atrasos dos fios e portas lógicas (KAHNG et al., 2011).

Na prática, alguns fluxos de projeto industriais não incorporam métodos orientados a performance durante o posicionamento inicial porque as informações de temporização podem ser muito imprecisas até que as posições das células tenham sido determinadas. Em vez disso, o TDP é incorporado em iterações subsequentes do posicionamento, especialmente no posicionamento detalhado, que será discutido na seção 2.2.3 (KAHNG et al., 2011).

2.2 Etapas do Posicionamento

Para lidar com a complexidade do posicionamento de células, as ferramentas de EDA tradicionalmente dividem essa etapa em três estágios: posicionamento global, legalização e posicionamento detalhado, Figura 2.10. Esses estágios serão discutidos a seguir.

Figura 2.10: Etapas do posicionamento

Fonte: adaptado de (FLACH, 2015).

2.2.1 Posicionamento Global

Seja o *netlist* $\mathcal{N} = (E, V)$, onde o conjunto de arestas E representa os fios que fazem as ligações entre as células e o conjunto V representa as células do circuito. O objetivo do posicionamento global é encontrar uma posição para cada uma das células dentro dos limites da área do circuito de modo que os objetivos adotados para o projeto

sejam minimizados. Nesse estágio, não interessa a posição final de cada célula, mas sim a distribuição delas ao longo do espaço de posicionamento (HENTSCHKE, 2002). Assim, o posicionamento global irá tentar reduzir a sobreposição de células sem preocupação em remover pequenas sobreposições entre elas. Além disso, não há a tentativa de alinhá-las nas bandas de posicionamento.

O estágio de posicionamento global é o mais importante dos três, pois ele é o que tem o maior impacto na qualidade da solução (WANG; CHANG; CHENG, 2009).

O escopo desse trabalho é o estágio de posicionamento global, por isso ele será discutido com mais detalhes no próximo capítulo.

2.2.2 Legalização

O resultado do posicionamento global tipicamente apresenta um grande número de sobreposições entre as células, além de elas poderem estar posicionadas fora das bandas de posicionamento. Por isso, o posicionamento precisa ser legalizado. A legalização é o processo de remover as sobreposições entre as células e alinhá-las nas bandas de posicionamento (WANG; CHANG; CHENG, 2009). Durante esse estágio, as células devem sofrer o menor deslocamento possível, de modo que a solução não seja drasticamente alterada (KAHNG et al., 2011).

2.2.3 Posicionamento Detalhado

Após a legalização, pode haver espaço para melhorar a solução, pois as perturbações provocadas para legalizar as células geralmente aumentam o comprimento de fio (WANG; CHANG; CHENG, 2009). Então, no estágio de posicionamento detalhado, o posicionamento resultante da legalização é otimizado através de trocas locais, de modo que a solução continue sendo legal. Nesse estágio também podem ser usados modelos de roteamento, atraso, entre outros mais precisos visto que são tratados menos elementos.

3 POSICIONAMENTO GLOBAL

No capítulo anterior, foram apresentados os fundamentos e os estágios da etapa de posicionamento de células em circuitos integrados. Nesse capítulo, o estágio de posicionamento global, por ser o escopo desse trabalho, será discutido com mais detalhes, de modo que serão mostradas as categorias de algoritmos que podem ser encontradas na literatura.

Segundo (KAHNG et al., 2011), existem três categorias de algoritmos de posicionamento global: algoritmos baseados em particionamento, algoritmos estocásticos e técnicas analíticas.

Nos algoritmos baseados em particionamento, o *netlist* e o leiaute são divididos recursivamente, de modo que uma função custo seja minimizada, como, por exemplo, o numero de conexões entre partições (SPINDLER; SCHLICHTMANN; JOHANNES, 2008).

Os algoritmos estocásticos utilizam movimentos randômicos para otimizar uma função custo (KAHNG et al., 2011). Geralmente, é empregado o *Simulated Annealing*, que teoricamente é capaz de encontrar uma solução ótima global. No entanto, possui um elevado tempo de processamento, o que limita a sua aplicação (SPINDLER; SCHLICHTMANN; JOHANNES, 2008).

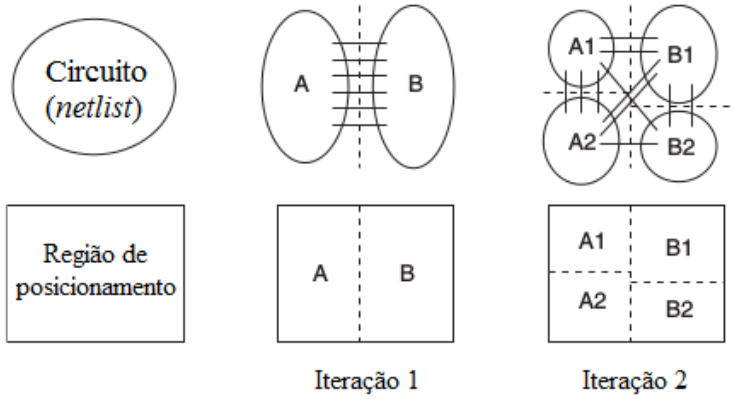
Os algoritmos baseados em técnicas analíticas modelam o posicionamento utilizando uma função objetivo que é minimizada por métodos de otimização numérica. Dependendo da função objetivo, os posicionadores analíticos podem ser classificados em duas categorias: não-lineares e quadráticos (SPINDLER; SCHLICHTMANN; JOHANNES, 2008). Os posicionadores analíticos não-lineares são baseados em uma função custo não-linear, como, por exemplo, expressar o comprimento de fio como uma função log-sum-exp. Já os posicionadores analíticos quadráticos formulam o comprimento de fio como uma função custo quadrática, que pode ser minimizada eficientemente ao resolver um sistema de equações lineares.

A seguir, as três categorias de algoritmos de posicionamento global introduzidas acima serão discutidas com uma maior profundidade.

3.1 Algoritmos Baseados em Particionamento

Os posicionadores baseados em técnicas de particionamento recursivamente dividem o *netlist* de forma descendente (*top-down*) em 2^k partições, sendo $k \geq 1$. Quando $k > 1$, o particionamento é comumente chamado de particionamento *multiway*. Nos casos particulares em que em que $k = 1$, o particionamento é chamado de *bisection*, e, quando $k = 2$, *quadrisection*. Além de particionar o *netlist*, o posicionador ainda divide recursivamente a área do leiaute em regiões retangulares (*bins*) e associa cada partição do *netlist* a um dos *bins*. O particionamento é um processo de refinamento *top-down*, onde cada *bin* é subdividido em *bins* com um número menor de células. A divisão dos *bins* é feita através de um corte horizontal ou vertical no caso em que são criadas duas partições (*bisection*), ou por cortes horizontal e vertical simultâneos quando são criadas quatro partições (*quadrisection*). A Figura 3.1 ilustra um exemplo de processo de divisão recursiva do *netlist* e do leiaute, de modo que, a cada iteração, são criadas duas partições (*bisection*). O particionamento do *netlist* e dos *bins* continua recursivamente até que cada *bin* tenha um pequeno número de células (LAVAGNO; SCHEFFER; MARTIN, 2006).

Figura 3.1: Exemplo de particionamento do *netlist* e do leiaute



Fonte: adaptado de (WANG; CHANG; CHENG, 2009).

Durante o processo de particionamento, cada corte feito miniminiza de forma heurística o número de conexões entre cada partição (KAHNG et al., 2011). Alguns algoritmos tradicionais utilizados para minimizar o número de conexões entre as partições são o Kernighan-Lin (KERNIGHAN; LIN, 1970) e o Fiduccia-Mattheyses (FM) (FIDUCCIA; MATTHEYSES, 1982). Dois exemplos de algoritmos baseados em particionamento são o Capo (ROY et al., 2005), que recursivamente divide o *netlist* e a área do leiaute em dois (*bisection*), o Dragon (TAGHAVI; YANG; CHOI, 2005), que utiliza uma abordagem híbrida que combina o particionamento com o *simulated annealing*.

3.2 Algoritmos Estocásticos - *Simulated Annealing*

O *Simulated Annealing* (SA) é uma técnica de otimização combinatória que tem as suas origens na mecânica estatística. Esse algoritmo simula o processo de recozimento utilizado no processo de têmpera de metais.

Dado um problema de otimização combinatória, o SA parte de uma configuração inicial e iterativamente muda para uma nova configuração até que algum critério de parada seja satisfeito. Em cada iteração, é feito, randomicamente, uma transição da atual configuração para a nova. Se a nova configuração leva a uma melhoria da função custo ou objetivo sendo otimizado, ela é aceita. Por outro lado, se o custo aumenta, a nova solução é probabilisticamente rejeitada, de modo que a probabilidade de rejeição aumenta a medida que são executadas as iterações. A habilidade de fazer uma transição para uma nova configuração com valores de custo piores permite que o SA escape de mínimos locais, o que é um ingrediente essencial para o seu sucesso (LAVAGNO; SCHEFFER; MARTIN, 2006).

Em posicionamento, uma nova configuração pode ser gerada deslocando uma célula ou trocando as posições de duas células. Para uma piora ΔC na função custo, devido a uma transição da configuração atual para uma nova, a probabilidade de rejeição da transição é dada por $1 - e^{\frac{-\Delta C}{T}}$, onde T, chamado de temperatura, é o parâmetro que controla a probabilidade de rejeição. Para variar T, é utilizado um processo de resfriamento, no qual inicialmente é atribuído a T um valor alto, que é gradualmente reduzido, de modo que transições para configurações que pioram a função custo fiquem menos prováveis ao passo que a temperatura diminui (LAVAGNO; SCHEFFER; MARTIN, 2006).

Para o algoritmo obter boas soluções, o processo de resfriamento precisa ser lento. Por essa razão, o *simulated annealing* tem um tempo de execução alto (LAVAGNO; SCHEFFER; MARTIN, 2006).

Uma implementação do *simulated annealing* aplicada ao problema de posicionamento de células é apresentada no Algoritmo 1, disponível em (KAHNG et al., 2011). A partir de um posicionamento inicial (linha 3), a função PERTURBAR gera um novo posicionamento $P^{'}$ perturbando o posicionamento atual (linha 6). A variável $\Delta custo$ armazena a diferença entre o custo do novo posicionamento e do anterior (linha 7). Se o custo melhora, isto é, se $\delta custo < 0$, então o novo posicionamento $P^{'}$ é aceito (linhas 8-9). Caso contrário $P^{'}$ é probabilisticamente aceito (linhas 10-13). Esse processo continua enquanto a temperatura T for maior que uma temperatura mínima T_{min}.

Algoritmo 1: *Simulated Annealing*

Entrada: conjunto de todas as células V

Saída: posicionamento P

```
1  início
2      T = T_0
3      P = POSICIONAR(V)
4      enquanto T > T_0 faça
5          enquanto !PARAR() faça
6              P' = PERTURBAR(P)
7              Δcusto = CUSTO(P') − CUSTO(P)
8              se Δcusto < 0 então
9                  P = P'
10             senão
11                 r = RANDOM(0, 1)
12                 se r < e^(−Δcusto/T) então
13                     P = P'
14                 fim
15             fim
16         fim
17         T = α × T
18     fim
19 fim
```

3.3 Algoritmos Analíticos

O posicionamento analítico expressa a função objetivo a ser otimizada como uma função matemática analítica das posições das células do circuito (WANG; CHANG; CHENG, 2009). Assim, o problema de posicionamento é transformado em um problema matemático. As técnicas de posicionamento analítico são classificadas como quadráticas ou não-lineares (SPINDLER; SCHLICHTMANN; JOHANNES, 2008). Os posicionadores analíticos quadráticos, Seção 3.3.1, formulam o comprimento de fio como uma função objetivo quadrática, que pode ser minimizada através da resolução de um sistema de equações lineares. Os posicionadores analíticos não-lineares são baseados em uma função

objetivo não-linear, como, por exemplo, formular o comprimento de fio através de uma função log-sum-exp. A ferramenta desenvolvida nesse trabalho usa a abordagem analítica quadrática, portanto, técnicas analíticas não-lineares não serão discutidas nesse texto.

3.3.1 Técnicas Analíticas Quadráticas

Nas técnicas analíticas quadráticas, os objetivos são modelados como funções quadráticas convexas (WANG; CHANG; CHENG, 2009). Para expressar a função objetivo como uma função quadrática, será considerado, no momento, que todas as redes de conexões possuem apenas dois pinos, isto é, conectam apenas duas células (redes com mais de dois pinos serão discutidas adiante). Para uma rede $\{i, j\}$ (rede que conecta as células i e j), o seu comprimento de fio (função objetivo) pode ser dado pela distância Manhattan entre as células:

$$L_{\{i,j\}} = |x_i - x_j| + |y_i - y_j|. \tag{3.1}$$

No entanto, $L_{\{i,j\}}$ não é diferenciável. Por essa razão, é adotada a distância Euclidiana quadrática:

$$L'_{\{i,j\}} = (x_i - x_j)^2 + (y_i - y_j)^2. \tag{3.2}$$

A Equação 3.2 é usualmente chamada de comprimento quadrático de fio. Então, para todas as células do circuito, define-se a função objetivo quadrática $\Phi(x, y)$ como:

$$\Phi(x, y) = \frac{1}{2} \sum_{1 \leq i < j \leq n} C_{\{i,j\}} \times ((x_i - x_j)^2 + (y_i - y_j)^2) \tag{3.3}$$

onde $C_{\{i,j\}}$ é o peso da conexão entre as células i e j. Visto que cada fio é representado duas vezes, isto é, uma para cada nó do circuito, foi adicionado o fator $\frac{1}{2}$. A Equação 3.3 pode ser representada matricialmente como:

$$\Phi(x, y) = \frac{1}{2} x^T Q_x + d_x^T x + \frac{1}{2} y^T Q_y + d_y^T y + constante \tag{3.4}$$

onde Q_x e Q_y são matrizes $n \times n$ simétricas positivas definidas, e d_x e d_y são vetores com n elementos, sendo n o números de células móveis do circuito (VISWANATHAN; CHU, 2004). O termo constante representa as ligações entre células fixas. A Equação 3.4 pode ser separada em $\Phi(x, y) = \Phi(x) + \Phi(y)$, de modo que as dimensões x e y podem ser

tratadas individualmente. Assim, o restante da discussão irá considerar apenas a dimensão x, isto é:

$$\Phi(x) = \frac{1}{2}x^T Q_x + d_x^T x + constante. \tag{3.5}$$

Os elementos da matriz hessiana Q_x representam as conexões entre os pares de células móveis, enquanto os elementos do vetor d_x representam as conexões entre uma célula móvel e um outro componente fixo, como uma célula fixa, um terminal ou um macro bloco. Se duas células i e j são móveis, o peso da conexão entre elas w é adicionado nas entradas q_{ii} e q_{jj} e subtraído das entradas q_{ij} e q_{ji} da matrix hessiana Q. Se uma das células do par é fixa, por exemplo a j, então o peso da conexão entre elas é adicionado na entrada q_{ii} da matriz Q e é subtraído $w \cdot x_j$ do vetor d_x, sendo x_j a posição x da célula fixa j. Se as duas células são fixas, Q_x e d_x não são modificados. A matriz Q_x é positiva semi-definida se não existem elementos fixos no circuito e é positiva definida se há algum elemento fixo. Em ambos os casos, a função $\Phi(x)$ é convexa e o seu valor mínimo é encontrado igualando a sua derivada a zero (SPINDLER; SCHLICHTMANN; JOHANNES, 2008). A derivada de $\Phi(x)$ é dada por:

$$\frac{\partial \Phi(x)}{\partial x} = Q_x + d_x. \tag{3.6}$$

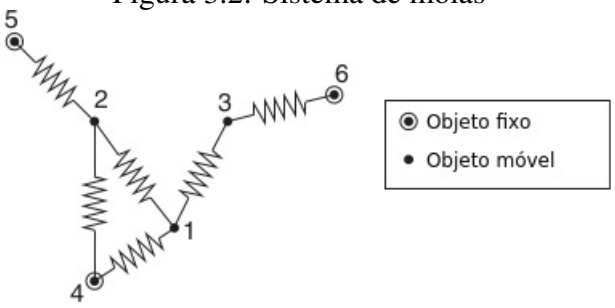
Ao igualar a equação 3.6 a zero, isto é,

$$Q_x + d_x = 0 \tag{3.7}$$

é obtido um sistema de equações lineares. Através da resolução desse sistema, são obtidas as posições das células na dimensão x com comprimento total de fios mínimo. O mesmo procedimento é aplicado para encontrar as posições das células na dimensão y.

No posicionamento quadrático, cada conexão entre dois pinos pode ser interpretada como uma mola elástica, Figura 3.2, então, a Equação 3.3 representa a energia potencial total do sistema de molas. Isso significa que encontrar a configuração de menor energia do sistema de molas é equivalente a minimizar o comprimento total de fios no posicionamento quadrático (WANG; CHANG; CHENG, 2009).

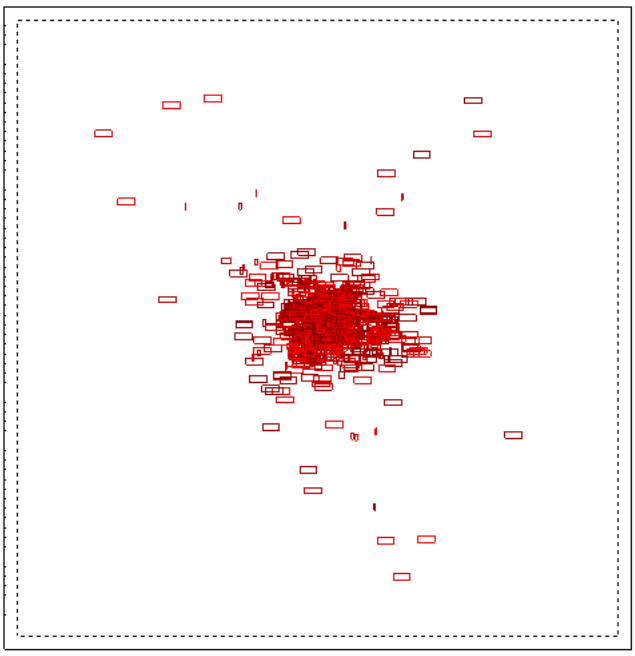
Figura 3.2: Sistema de molas



Fonte: adaptado de (WANG; CHANG; CHENG, 2009).

Para um sistema de molas, a configuração de menor energia é também a de equilíbrio estático, ou seja, a força resultante atuando em cada nó do sistema é zero. Assim, se a Equação 3.7 for satisfeita, os objetos do sistema estarão em equilíbrio estático (WANG; CHANG; CHENG, 2009). No entanto, a solução do sistema de equações lineares 3.7 pode resultar em uma configuração em que as células estejam concentradas em alguma parte da área de posicionamento do *chip*, ou seja, pode haver uma grande quantidade de células sobrepostas, conforme ilustrado na Figura 3.3, o que impossibilita a fabricação do circuito integrado. Assim, forças de espalhamento são adicionadas ao sistema para que as células sejam gradualmente distribuídas pela área do *chip*, reduzindo, assim, as sobreposições. A principal diferença entre os algoritmos quadráticos são como as conexões entre as células são modeladas e como as forças de espalhamento são introduzidas no sistema (FLACH; JOHANN; REIS, 2011).

Figura 3.3: Posicionamento quadrático inicial



Fonte: figura elaborada pelo autor.

3.3.2 Modelos de Conexão entre Pinos

Os circuitos tipicamente contêm redes de conexão com dois ou mais pinos. Como os posicionadores analíticos quadráticos constroem o sistema linear tomando pares de conexões, redes com mais de dois pinos precisam ser decompostas em grupos de pares de pinos. Para isso, diversos modelos foram propostos (WANG; CHANG; CHENG, 2009). Nessa seção serão apresentados os modelos clique, estrela, híbrido e *Bound to Bound*.

3.3.2.1 Modelo Clique

O clique é o modelo tradicionalmente usado em algoritmos analíticos. Nesse modelo, uma rede com k pinos é decomposta em conexões de dois pinos formando um clique (VISWANATHAN; CHU, 2004), ou seja, cada um dos pinos está conectado com os demais pinos da rede.

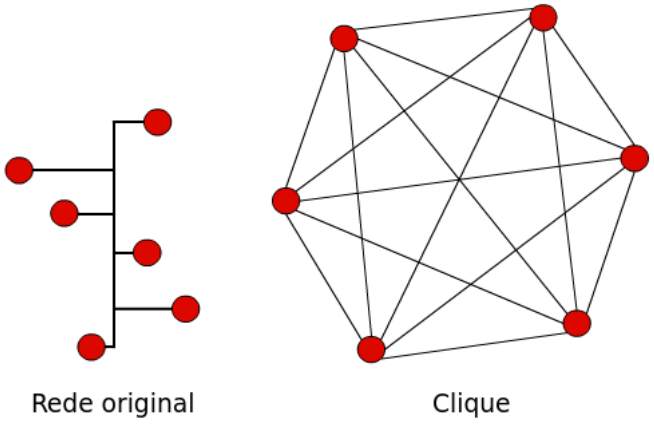
Uma rede transformada em um clique com k pinos possui

$$\frac{k(k-1)}{2} \tag{3.8}$$

conexões entre pares de pinos (VISWANATHAN; CHU, 2004).

Uma forma de determinar o peso das conexões no modelo clique é $\frac{w}{k-1}$ (VYGEN, 1997), em que w é o peso da rede original e k é o número de pinos da rede. A Figura 3.4 ilustra a transformação de uma rede com 6 pinos em um clique.

Figura 3.4: Modelo de conexão entre pinos clique

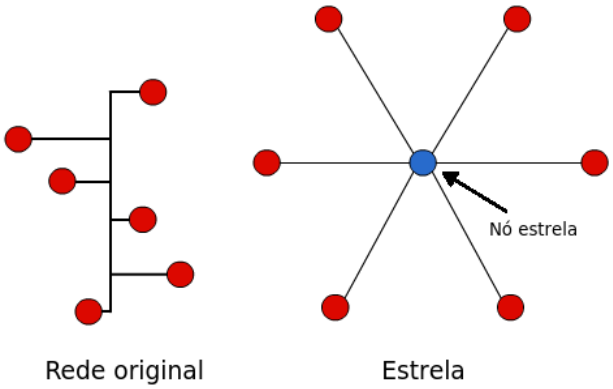


Fonte: figura elaborada pelo autor.

3.3.2.2 Modelo Estrela

No modelo estrela, é criado um nó adicional ao qual todos os pinos da rede são conectados. Assim, uma rede com k pinos é decomposta em k conexões (VISWANATHAN; CHU, 2004). No posicionamento analítico quadrático, o nó estrela é adicionado no sistema linear como se fosse uma célula do circuito. A Figura 3.5 ilustra a transformação de uma rede de 6 pinos em estrela.

Figura 3.5: Modelo de conexão entre pinos estrela



Fonte: figura elaborada pelo autor.

O trabalho de (MO; TABBARA; BRAYTON, 2000) mostra que, em média, o modelo estrela gera 30% a menos de conexões entre pares de pinos do que o modelo clique.

3.3.2.3 Modelo Híbrido

No modelo híbrido, proposto por (VISWANATHAN; CHU, 2004), redes com até 3 pinos são decompostas utilizando o modelo clique, enquanto que, para redes com mais pinos, é utilizado o modelo estrela.

No trabalho de (VISWANATHAN; CHU, 2004), foi provado que para uma rede com k pinos, se o peso atribuído para as conexões entre pares de pinos é w_c no modelo clique e kw_c no modelo estrela, então o modelo clique é equivalente ao modelo estrela no posicionamento quadrático.

3.3.2.4 Modelo Bound to Bound

No modelo *Bound to Bound* (B2B) (SPINDLER; JOHANNES, 2007), os pinos de menor coordenada x e o de maior coordenada x, chamados de pinos de borda, são conectados entre si. Os pinos restantes, também chamados de pinos internos por estarem

em posições intermediárias, são conectados somente aos pinos de borda. A Figura 3.6 ilustra a decomposição de uma rede utilizando o modelo B2B. Uma rede com k pinos terá

$$1 + 2(k - 2) \tag{3.9}$$

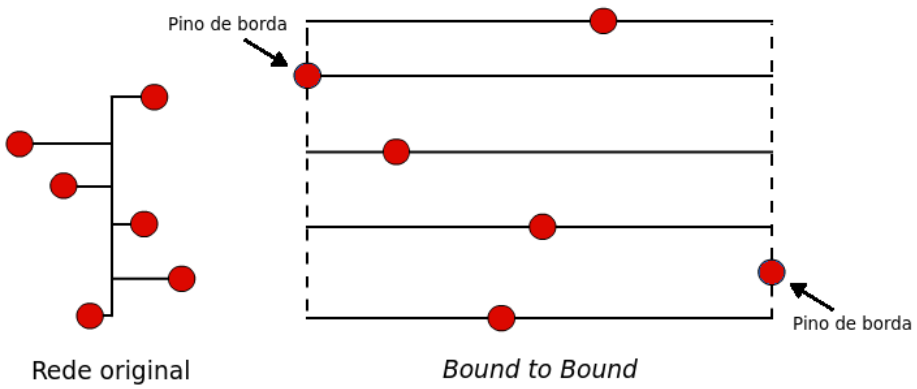
conexões entre dois pinos.

O peso para uma conexão entre um par de pinos é dada por:

$$w = \frac{2}{P - 1}\frac{1}{l_x} \tag{3.10}$$

onde k é o número de pinos da rede e l_x é a distância entre os pinos em relação ao eixo x.

Figura 3.6: Modelo de conexão entre pinos *Bound to Bound*



Fonte: figura elaborada pelo autor.

Uma vantagem do modelo B2B é que ele permite que as ferramentas de posicionamento utilizem o HPWL como métrica de otimização.

4 TRABALHOS RELACIONADOS

Esse capítulo apresenta uma revisão bibliográfica sobre algoritmos quadráticos, descrevendo desde o FastPlace (VISWANATHAN; CHU, 2004), que marcou o retorno dos algoritmos com objetivos quadráticos e baseados em sistemas de força (MONTEIRO, 2014), até os mais recentes.

4.1 FastPlace, FastPlace 2.0 e FastPlace 3.0

No FastPlace (VISWANATHAN; CHU, 2004) as redes de interconexão são modeladas utilizando o modelo híbrido, apresentado na Seção 3.3.2.3. Para realizar o posicionamento, o FastPlace é dividido em três estágios. No primeiro, é executada iterativamente a intercalação da resolução do sistema linear, construído a partir do modelo híbrido, com uma técnica de deslocamento de células até que a distribuição de células seja razoavelmente homogênea. No segundo estágio, o posicionamento obtido no primeiro é refinado. Para isso, são intercaladas a resolução do sistema linear com uma técnica de refinamento local e com técnica de deslocamento de células utilizada no estágio anterior. Esse processo é repetido até que as células estejam bem distribuídas. Por fim, no terceiro estágio, o posicionamento é legalizado. As etapas do FastPlace são apresentadas no Algoritmo 2.

Algoritmo 2: *FastPlace*

```
1  início
2      Estágio 1: Posicionamento Global Aproximado (CGP)
3      repita
4          Otimização Global (resolução do sistema linear).
5          Deslocamento de células e adição de forças de espalhamento.
6      até o posicionamento seja aproximadamente homogêneo;
7      Estágio 2: Posicionamento Global com comprimento de fio otimizado
8                 (WIGP)
9      repita
10         Otimização Global (resolução do sistema linear).
11         Refinamento Local Iterativo.
12         Deslocamento de células e adição de forças de espalhamento.
13     até as células estejam bem distribuídas;
14     Estágio 3: Posicionamento Detalhado (DP)
15     repita
16         Reduzir o comprimento de fio usando uma heurística gulosa.
17         Legalizar o posicionamento.
18     até não há otimização significativa comprimento de fio;
19 fim
```

No subestágio de deslocamento de células, linhas 5 e 12 do Algoritmo 2, a região de posicionamento é dividida em uma grade de *bins*, de modo que cada *bin* acomode em média quatro células. A partir do posicionamento obtido através da resolução do sistema linear (Otimização Global), é computada a utilização de cada *bin*, que é a área total de células dentro do *bin*. Para calcular a utilização, são somadas as áreas das células que estão totalmente dentro do *bin* com a área de interseção das células que estão parcialmente cobertas pelo *bin*. Após o cálculo da utilização, as células são deslocadas. Considerando a dimensão x, o algoritmo trata cada linha de *bins* individualmente, deslocando horizontalmente as células cobertas pelos *bins* dessa linha. Para isso, é criada uma estrutura auxiliar de *bins* que reflete a utilização dos *bins* da grade original. Se um *bin* da grade original tem uma alta utilização, o *bin* correspondente na estrutura auxiliar tende a sua largura maior que a do original. Da mesma forma, se um *bin* na grade original tem uma baixa utilização, o seu *bin* correspondente na estrutura auxiliar tende a ter a largura menor que a do *bin* original. A Figura 4.1(a) ilustra um gráfico que reflete a largura dos *bins* de uma linha da grade original e a utilização de cada deles. A Figura 4.1(b) mostra a largura dos *bins* auxiliares criados. Considerando a dimensão x, o cálculo da coordenada x (NB_i) da borda do bin auxiliar i é dado por:

$$NB_i = \frac{OB_{i-1}(U_{i+1} + \delta) + OB_{i+1}(U_i + \delta)}{U_i + U_{i+1} + 2\delta} \tag{4.1}$$

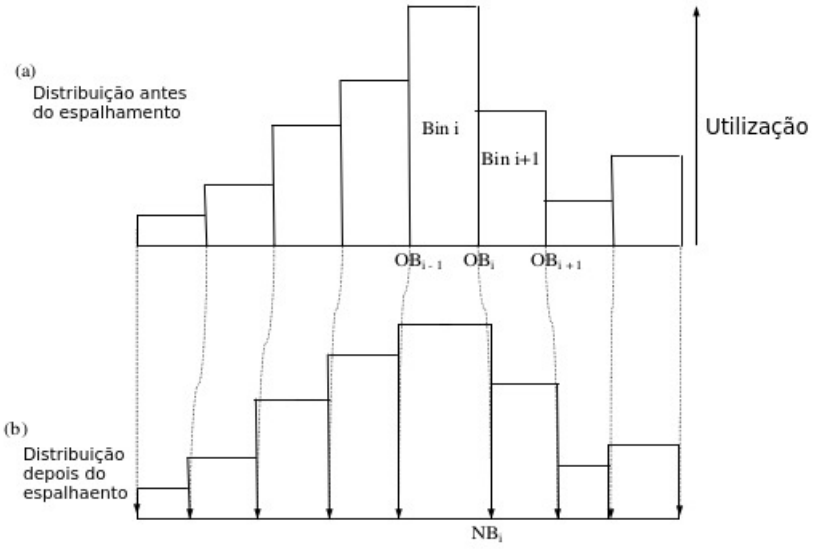
onde OB_i é a coordenada x da borda do *bin* i na grade original e δ é uma constante de valor 1.5. Depois que a estrutura auxiliar foi criada, as células do *bin* i da grade original são mapeadas para o *bin* auxiliar correspondente. O espalhamento das células dentro do *bin* auxiliar i é realizado calculando uma nova coordenada x para cada uma das células mapeadas. O cálculo da nova coordenada x, ($x_j^{'}$), da célula j é dado por:

$$x_j^{'} = \frac{NB_i(x_j - OB_{i-1}) + NB_{i-1}(OB_i - x_j)}{OB_i - OB_{i-1}} \tag{4.2}$$

onde x_j é a coordenada x da célula j antes de fazer o deslocamento dela.

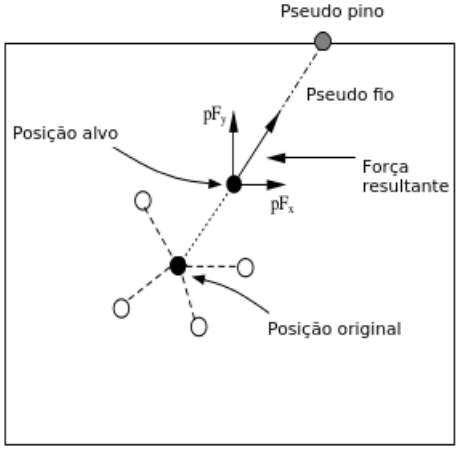
Após as novas posições de cada uma das células terem sido determinadas, é necessário adicionar uma força a cada uma delas para evitar que elas voltem para a posição original na próxima vez em que o sistema linear for resolvido. Isso é feito conectando cada uma das células posicionadas na posição alvo a um pseudo pino localizado na borda do *chip*, conforme ilustrado Figura 4.2.

Figura 4.1: Utilização e dimensão dos *bins*



Fonte: adaptado de (VISWANATHAN; CHU, 2004).

Figura 4.2: Adição de uma força através da conexão com um pseudo pino



Fonte: adaptado de (VISWANATHAN; CHU, 2004).

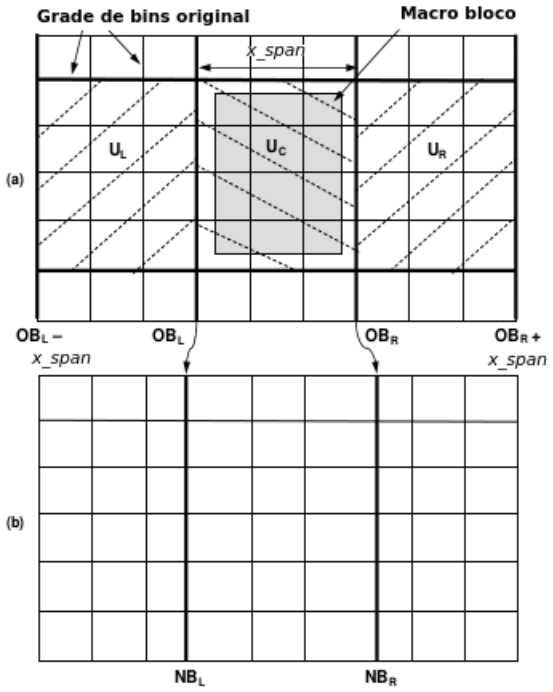
Como o pseudo pino adicionado na borda do *chip* é fixo, apenas a diagonal principal da matriz Q e os vetores d_x e d_y são atualizados.

O *Iterative Local Refinement* (ILR), ou refinamento local iterativo, linha 11 do Algoritmo 2, é executado para reduzir o comprimento de fio durante as iterações do estágio 2 do algoritmo. Ele faz uso de uma grade de *bins* para calcular a utilização em uma determinada região de posicionamento. Para cada célula presente em um determinado *bin*, são computados quatro *scores* que correspondem às quatro possíveis direções de movimento. Para calcular o *score*, é assumido que a célula está movendo a partir de sua posição atual em um *bin* origem para a mesma posição em um *bin* destino adjacente. Cada *score* é uma soma ponderada de duas componentes: a primeira seria a redução no comprimento de fio de todas as redes conectadas à célula ao realizar o movimento. Para estimar o compri-

mento de fio nessa situação, é utilizada a métrica HPWL. A segunda componente é uma função da utilização dos *bins* origem e destino. Se todos os quatro *scores* são negativos, a célula não será movida para outro *bin* caso contrário, ela será movida para o *bin* destino com o maior *score*.

O FastPlace 2.0 (VISWANATHAN; PAN; CHU, 2006) é uma extensão do FastPlace para tratar de circuitos que têm macro blocos. Para deslocar os macro blocos, é empregada a mesma técnica de deslocamento de células, linhas 5 e 12 do Algoritmo 2, discutida anteriormente. A única diferença entre o deslocamento de células e o deslocamento de macro blocos é como a estrutura auxiliar de *bins* é construída. No caso da dimensão x, ao invés de tratar individualmente os *bins* de uma determinada linha, é considerado o conjunto de todos os *bins* que fazem interseção com o macro bloco. A Figura 4.3 ilustra como a estrutura auxiliar é construída.

Figura 4.3: Construção da estrutura auxiliar de *bins*



Fonte: adaptado de (VISWANATHAN; PAN; CHU, 2006).

A Figura 4.3 mostra que ao invés de alterar a coordenada x da borda de um *bin* como acontece no caso do deslocamento de células, são alteradas as coordenadas x das bordas dos *bins* que delimitam a região formada pelos *bins* que fazem interseção com o macro bloco. A borda esquerda (NB_L) e a borda direita (NB_R), são calculadas, respectivamente por

$$NB_L = \frac{(OB_L - x\_span)(U_C + \delta) + OB_R(U_L + \delta)}{U_L + U_C + 2\delta} \tag{4.3}$$

e

$$NB_R = \frac{OB_L(U_R + \delta) + (OB_R + x\_span)(U_C + \delta)}{U_R + U_C + 2\delta}. \tag{4.4}$$

O cálculo da nova posição x do macro bloco é dado por:

$$x^{'} = \frac{NB_R(x - OB_L) + NB_L(OB_R - x)}{OB_R - OB_L} \tag{4.5}$$

onde x a posição original do macrobloco.

A legalização no FastPlace 2.0 é feito em duas etapas: na primeira, as células são ignoradas e são removidas as sobreposições entre macro blocos. Na segunda, os macro blocos são mantidos fixos e e as células são legalizadas. Durante o processo de legalização dos macro blocos, é desejável mantê-los o máximo possível nas posições obtidas durante a etapa de posicionamento global, denotadas pelo algoritmo como posições alvo. Assim, o problema de legalização dos macro blocos torna-se um problema de minimização da perturbação, ou seja, dos deslocamentos dos macro blocos em relação às posições alvo. A legalização dos macro blocos emprega o *simulated annealing*, que utiliza como função custo uma soma ponderada da perturbação total com a penalidade por posicionar o macro bloco for dos limites da área de posicionamento. Para não perturbar demasiadamente a solução, são permitidos somente trocas aleatória entre dois macro blocos adjacentes. Após as sobreposições entre os macro blocos terem sido removidas, os macro blocos são fixados, tornando-se obstáculos durante a legalização das células.

O FastPlace 3.0, (VISWANATHAN; PAN; CHU, 2007), introduz algumas melhorias em relação às versões anteriores do FastPlace. A etapa de posicionamento global é dividida em uma estrutura multinível, que utiliza dois níveis de clusterização. A clusterização é empregada porque reduz o problema de posicionamento para circuitos muito grandes. No primeiro nível de clusterização do FastPlace 3.0, são criados clusters com dois ou três objetos baseando-se somente na conectividade entre eles. Então é feito um posicionamento inicial rápido utilizando como objetos a serem posicionados os clusters formados. A solução obtida é repetidamente refinada utilizando o processo de refinamento local iterativo regular. Após o refinamento, é feita uma nova clusterização, de modo que nesse nível são consideradas a conectividade entre os objetos sendo clusterizados e a posição deles. Os novos clusters são posicionados através da resolução do sistema linear. Em cada cluster, são executados o deslocamento de células e a adição de forças de espalhamento discutidos anteriormente. Antes de legalizar a solução, o posicionamento

obtido passa por uma série de refinamentos.

Da mesma forma que o FastPlace 2.0, a legalização no FastPlace 3.0 também é dividida em duas etapas: legalização de macro blocos e legalização de células. A diferença é que a legalização das células utiliza uma técnica de controle de congestionamento.

4.2 Kraftwerk 2

O Kraftwerk 2 (SPINDLER; SCHLICHTMANN; JOHANNES, 2008) utiliza o modelo de redes *bound-to-bound*, apresentado na Seção 3.3.2.4 para modelar as conexões do circuito. As forças de espalhamento que são aplicas nas células são separadas em duas forças: a força de movimento e a força de retenção. A força de movimento é responsável por espalhar as células pela área do circuito. Para determinar a força de movimento, o posicionamento é representado como um sistema de demanda e fonte D, Equação 4.6.

$$D(x,y) = D_{mod}^{demanda}(x,y) - D_{mod}^{fonte}(x,y) \tag{4.6}$$

O termo $D_{mod}^{demanda}(x,y)$ representa os módulos enquanto que $D_{mod}^{oferta}(x,y)$ é a área de posicionamento do *chip*. A demanda e a fonte devem estar equilibrados, isto é a integral da demanda deve ser igual a integral da fonte.

Para formular a demanda dos módulos, é utilizada uma função retangular R. Para pontos dentro do retângulo, o valor da função é um. Já para pontos fora do retângulo, o valor da função é zero. O retângulo é definido pelo seu canto inferior esquerdo $(x_{ll,y_{ll}})$, sua largura w e sua altura h, ou seja, $R(x,y;x_{ll},y_{ll},w,h)$. A demanda de um módulo com largura w_i e altura h_i localizado na posição $(x_i^{'},y_i^{'})$, é, então, definida pela Fórmula 4.7 abaixo.

$$D_{mod,i}^{demanda}(x,y) = d_{mod,i}.R(x,y;x_i^{'} - \frac{w_i}{2}, y_i^{'} - \frac{h_i}{2}, w_i, h_i) \tag{4.7}$$

A demanda para todos os módulos do circuito é definida como a soma das demandas de todos os módulos móveis M e todos os módulos fixos F, Equação 4.8.

$$D_{mod}^{demanda} = \sum_{i=1}^{M+f} D_{mod,i}^{demanda}(x,y) \tag{4.8}$$

Então, para um ponto qualquer (x,y), o valor de $D_{mod}^{demanda}$ reflete o número de módulos sobrepostos nesse ponto.

A fonte D_{mod}^{fonte} é dada pela área de posicionamento do *chip*, como pode ser visto abaixo na Fórmula 4.9.

$$D_{mod}^{fonte} = d_{sup}.R(x, y; x_{Chip}, y_{Chip}, w_{Chip}, h_{Chip}) \tag{4.9}$$

Após definir $D_{mod}^{demanda}$ e D_{mod}^{fonte}, a Fórmula 4.6 pode ser interpretada como uma distribuição de cargas elétricas que cria um potencial elétrico Φ, dado pela equação de Poisson, Equação 4.10.

$$(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})\Phi(x, y) = -D(x, y) \tag{4.10}$$

As novas posições dos módulos são determinadas através da solução da Equação 4.10. A força de movimento para o módulo i é criada por uma mola que conecta o módulo ao ponto calculado pela equação de Poisson.

Para espalhar as células pela área do *chip*, é utilizada a força de movimento. Além dessa força, há também a força das conexões entre os módulos atuando, que é responsável pela minimização do comprimento de fio. Assim, a força das conexões precisa ser compensada em cada iteração do algoritmo, caso contrário, os módulos voltariam para as posições iniciais, na qual o comprimento de fios é mínimo. Então, a compensação é feita pela força de retenção, que é igual ao negativo da força das conexões.

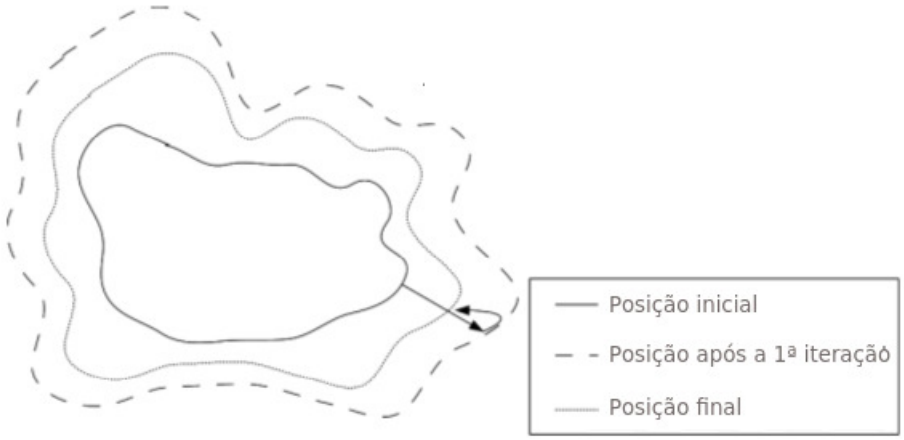
No Kraftwerk 2 o sistema de equações lineares é construído somando as forças de movimento com as de retenção e de conexões e igualando a soma a zero. Ao resolver o sistema com respeito a x, por exemplo, é obtido o deslocamento Δx para cada um dos módulos.

4.3 Quadratic Placement with Single-iteration Linear System Solver

O algoritmo, proposto por (FLACH; JOHANN; REIS, 2011), explora um fenômeno que ocorre ao resolver um sistema linear utilizando o método *Incomplete Cholesky Conjugate Gradient* (ICCG). Inicialmente, as células expandem e depois tendem a ser puxadas, estabilizando em suas respectivas posições finais nas iterações finais do ICCG. A expansão é causada pelas forças de espalhamento adicionadas no sistema enquanto que a contração ocorre devido a conectividade entre as células. Nas iterações iniciais do resolvedor, as forças de espalhamento fazem com que as posições das células sejam superestimadas em relação as suas posições finais. À medida que o sistema converge, as

células sofrem uma contração, sendo puxadas até atingirem as suas respectivas posições finais, como pode ser visto na Figura 4.4.

Figura 4.4: Expansão e contração das posições das células



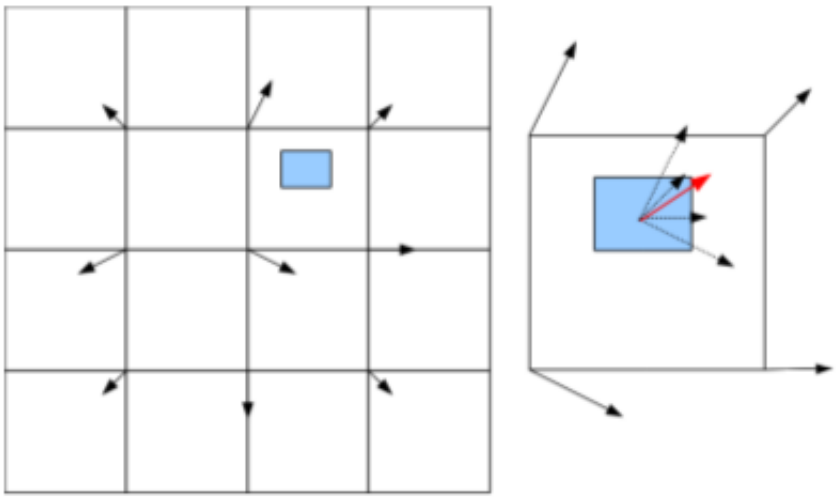
Fonte: adaptado de (FLACH; JOHANN; REIS, 2011).

Baseado no fenômeno que ocorre com as células à medida que a resolução do sistema linear converge, foi proposto um algoritmo de posicionamento que executa apenas a primeira iteração do resolvedor e, então, adiciona forças de espalhamento ao sistema.

O algoritmo, inicialmente, particiona as células em quatro grupos. Então, cada grupo será posicionado em seu respectivo quadrante na área de posicionamento.

Para adicionar forças de espalhamento, a área de posicionamento é dividida hierarquicamente em regiões retangulares. No primeiro nível, a área é dividida em quatro regiões retangulares iguais. No próximo nível, cada uma das regiões criadas no nível anterior é dividida em quatro regiões retangulares iguais também. O processo de divisão recursiva termina quando a área da região do último nível é aproximadamente igual à área média de todas as células. Sempre que a área é dividida, são computadas forças nos cantos das regiões criadas. Essas forças apontam a partir de regiões com alta densidade para regiões de baixa densidade. Então, a força que irá atuar na célula é o resultado da interpolação das forças dos quatro cantos da região, Figura 4.5. Esse processo é feito em cada nível da divisão da área do circuito.

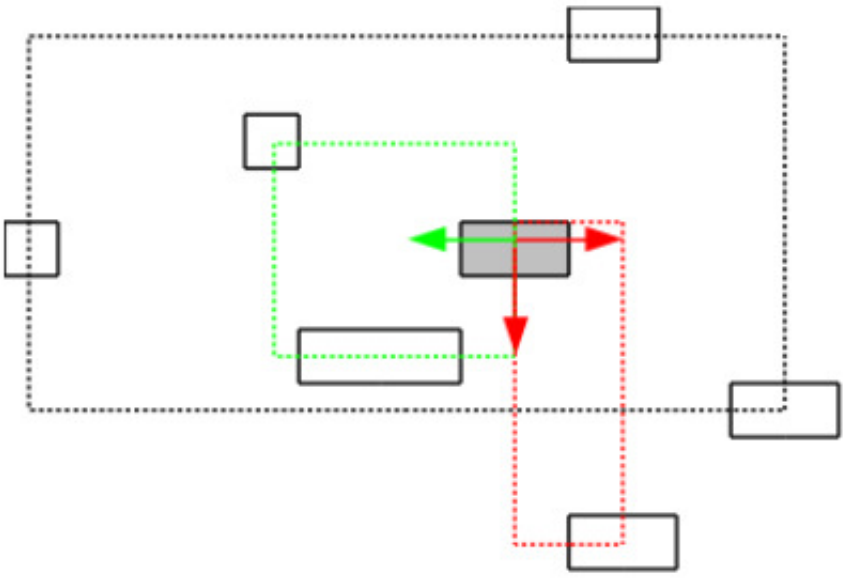
Figura 4.5: Adição de forças de espalhamento



Fonte: (FLACH; JOHANN; REIS, 2011).

O algoritmo também adiciona forças de espalhamento em células que estão na borda da *Bounding Box* (BB) da rede de interconexão, isto é, do menor retângulo que engloba todos os pinos de uma rede. Para essas células, uma força é adicionada a fim de movê-la em direção à borda oposta da *Bounding Box*. No exemplo da Figura 4.6, há uma célula que está na borda de duas *Bounding Boxes*. Considerando a BB verde, é adicionada, na célula, somente uma força que aponta para a borda oposta, pois a célula está em apenas uma das bordas dessa BB. Já no caso da BB vermelha, são computadas duas forças que apontam para bordas opostas, pois a célula está em duas bordas dessa BB. Visto que essas forças tendem a reduzir a *Bounding Box*, elas estão reduzindo diretamente o HPWL de uma determinada rede. Por isso, elas são chamadas de forças HPWL.

Figura 4.6: Adição de forças HPWL

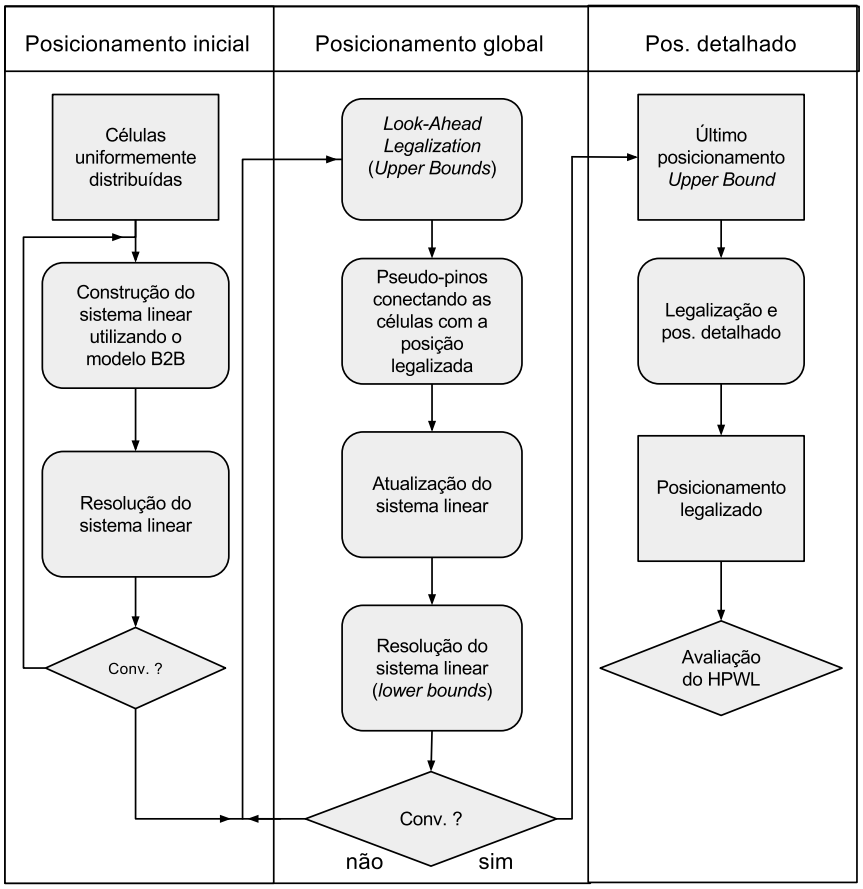


Fonte: (FLACH; JOHANN; REIS, 2011).

4.4 SimPL

O algoritmo SimPL (KIM; LEE; MARKOV, 2012) consiste de três fases: posicionamento inicial, posicionamento global e posicionamento pós-global. A Figura 4.7 apresenta o fluxo de execução do SimPL. No posicionamento inicial, o circuito é modelado utilizando o modelo de conexões *Bound to Bound*, Seção 3.3.2.4. O modelo B2B é dependente do posicionamento, então, durante a primeira fase, o sistema linear é reconstruído e resolvido até que o HPWL do circuito convirja.

Figura 4.7: Fluxo de execução do SimPL



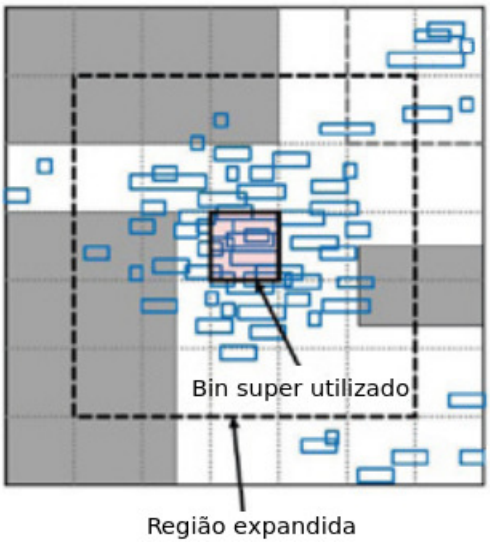
Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

A fase de posicionamento global utiliza dois posicionamentos, o *upper-bound* (posicionamento superior) e o *lower-bound* (posicionamento inferior). O *upper-bound* é gerado pela algoritmo de *Look-Ahead Legalization* (LAL) e o *lower-bound* é obtido através da resolução do sistema linear.

O *Look-Ahead Legalization* (LAL), ou legalização aproximada, divide a área de posicionamento em uma grade de *bins*. Para cada *bin*, é calculada a sua densidade, que é definida por $\frac{A_c}{A_a}$, onde A_c é a área total de células contida no *bin* e A_a a área total disponível

do *bin*. Um *bin* é considerado super utilizado se a sua densidade excede um dado valor γ, tal que $0 < \gamma < 1$. O algoritmo identifica *bins* super utilizados e os agrupa com os vizinhos. Em seguida, a região definida pelo agrupamento de *bins* é expandida até que ela seja uma região retangular com densidade $\leq \gamma$. Essa região também é chamada de *cluster*. A Figura 4.8 ilustra um *bin* com alta densidade e o *cluster* formado a partir dele.

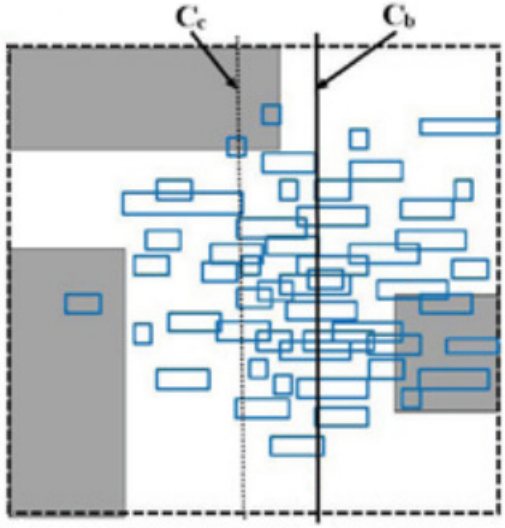
Figura 4.8: Região expandida de *bins* super utilizados



Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

O LAL executa um particionamento geométrico recursivo da região expandida. No primeiro nível, o particionamento é feito por uma linha de corte vertical. Nos demais, a direção da linha de corte alterna entre horizontal e vertical. O processo continua até que o *cluster* tenha atingido dez níveis de particionamento ou enquanto as sub-regiões criadas tenham área maior que quatro vezes a área de um *site*. A linha de corte, chamada de C_b, divide a região em duas sub-regiões, a B_0 e a B_1, de modo que ambas tenham a mesma área livre para posicionar as células. Além da C_b, é utilizada uma outra linha de corte paralela à C_b, a C_c, que cria as sub-regiões S_0 e S_1, de maneira que seja distribuída igualmente entre elas a área total de células contida na região original. A Figura 4.9 ilustra como as linhas C_b e C_c dividem a região.

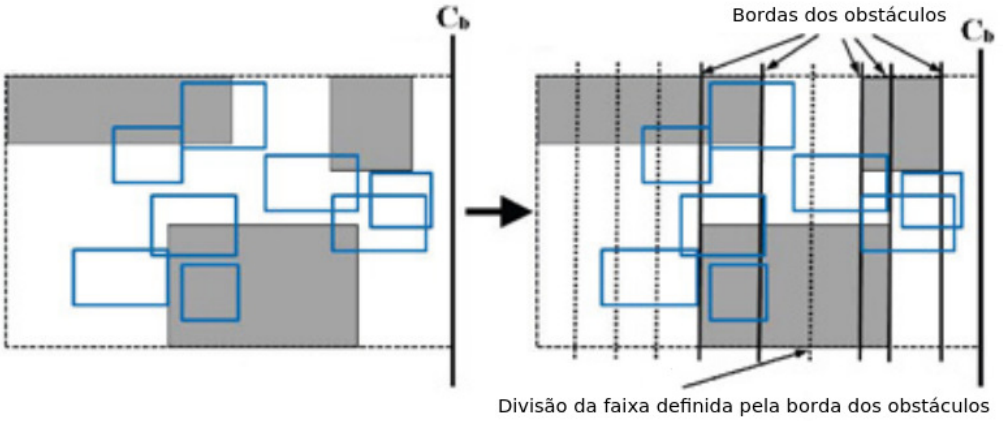
Figura 4.9: Cortes C_b e C_c



Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

Após dividir a região expandida, são adicionadas, nas duas sub-regiões, linhas de corte paralelas a C_b nas bordas dos obstáculos, definindo faixas. As faixas são ainda subdivididas se a área livre delas for superior a 10% da área livre da respectiva sub-região. A Figura 4.10 ilustra a criação das faixas na sub-região esquerda formada a partir da divisão da região expandida pela linha de corte C_b vertical.

Figura 4.10: Divisão da sub-região esquerda em faixas

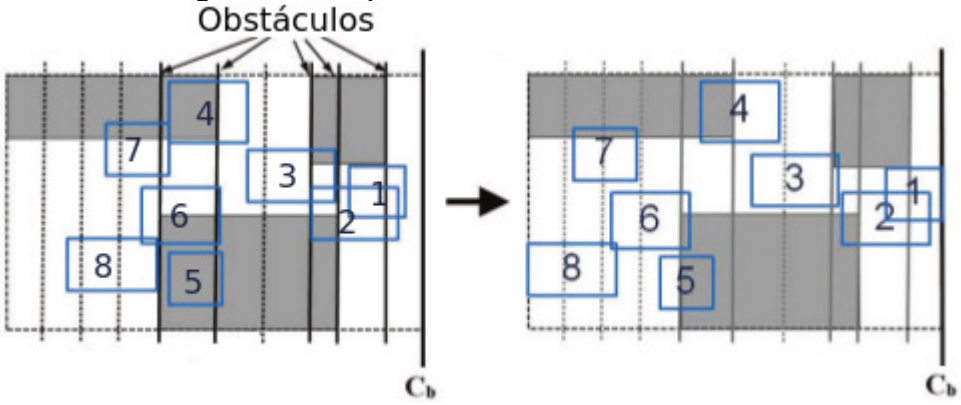


Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

As células móveis presentes nas sub-regiões criadas pela linha de corte C_c são ordenadas pela distância em relação à C_b. Então, aquelas que pertencem a S_0 são posicionadas nas faixas da sub-região B_0, enquanto que as que pertencem a S_1 são posicionadas nas faixas da sub-região B_1. Para realizar o posicionamento das células dentro das faixas, o LAL posiciona a célula mais afastada em relação à C_b na faixa livre mais afastada em relação à C_b também. Uma faixa é considerada livre enquanto enquanto a área total de células contidas nessa faixa for menor que γA_a, sendo A_a a área disponível para posicio-

namento da respectiva faixa. A Figura 4.11 ilustra o processo de espalhamento das células nas faixas.

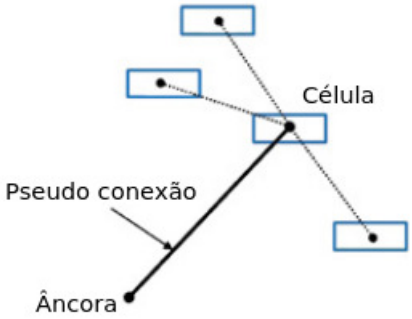
Figura 4.11: Espalhamento das células nas faixas



Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

As posições das células geradas pelo algoritmo de LAL são utilizadas como âncoras, ou seja, como pontos fixos que serão conectados às respectivas células do posicionamento *lower-bound* através de pseudo conexões, como pode ser visto na Figura 4.12.

Figura 4.12: Conexão da âncora à célula



Fonte: adaptado de (KIM; LEE; MARKOV, 2012).

Como as âncoras são pontos fixos, elas não implicam em um aumento nas dimensões da matriz de conectividade Q, visto que o peso da pseudo conexão é adicionado somente na diagonal principal da matriz Q e nos vetores d_x e d_y. O peso da pseudo conexão é multiplicado por um fator α, dado por:

$$\alpha = 0.01 * (1 + \text{Número da iteração}) \tag{4.11}$$

onde Número da iteração indica a iteração em execução do posicionamento global.

A fim de capturar com mais precisão a quantidade de sobreposição entre as células, os *bins* são refinados em cada iteração, de modo que as suas dimensões seja reduzidas por um fator $\beta = 1.06$ até que a área deles seja aproximadamente igual a quatro vezes a

área média das células. Esse processo de refinamento afeta o agrupamento de *bins* super utilizados durante o LAL, limitando o movimento das células e induzindo a convergência do algoritmo em iterações posteriores.

Depois que o posicionamento *lower-bound* convergiu, é executado mais uma vez o LAL e a solução é enviada para a etapa de posicionamento pós-global.

No posicionamento pós-global, a solução é legalizada e é executado o posicionamento detalhado. No entanto, o SimPL não propõe técnicas para essa etapa, apenas sugere o uso de algoritmos estado-da-arte.

4.5 MAPLE

Os algoritmos de posicionamento estado-da-arte integram múltiplos estágios de otimização, os quais podem utilizar diferentes objetivos. Uma péssima coordenação entre estágios consecutivos pode causar mudanças radicais nos resultados de posicionamentos intermediários. Essas mudanças são destrutivas quando as melhorias obtidas em um estágio anterior são desfeitas, aumentando o tempo de execução e debilitando a qualidade da solução final.

Em algoritmos de posicionamento multinível, iterações antes e após o processo de desagrupamento de células (*unclustering*) e antes do posicionamento detalhado podem provocar mudanças abruptas nos objetivos.

O MAPLE (KIM et al., 2012) é um *framework* multinível que foi proposto para resolver ou contornar os problemas apresentados acima. Ele introduz um novo estágio intermediário que otimiza a combinação linear dos objetivos utilizados no estágio precedente e no estágio sucessivo. Esse algoritmo consiste de três etapas: agrupamento de células (*clustering*), posicionamento *top-level* e *Progressive Local Refinement* (ProLR) ou refinamento local progressivo. Para fazer o agrupamento das células, é aplicado o algoritmo *Best-choice clustering* (ALPERT et al., 2005) de modo que o número de grupos (*clusters*) seja igual à metade do número de objetos que serão posicionados.

Na etapa de posicionamento *top-level*, é adotado o *Look-Ahead Legalization* do SimPL (KIM; LEE; MARKOV, 2012). O algoritmo gera, em cada iteração, uma solução *upper-bound* e outra *lower-bound*. Para o caso de circuitos com macro blocos fixos, o posicionamento ocorre da mesma forma que no SimPL. Para posicionar macro blocos móveis, foi proposta uma variação do *Look-Ahead Legalization*, chamada de *Extended Look-Ahead Legalization* (E-LAL). O E-LAL gera a solução *upper-bound* em dois pas-

sos: primeiro, são determinadas as posições dos macro blocos. Depois, as células são posicionadas. Para posicionar os macro blocos, o E-LAL utiliza uma variante da técnica de deslocamento de células do FastPlace (VISWANATHAN; CHU, 2004).

O *Progressive Local Refinement* (ProLR) é utilizado para reduzir o comprimento de fio e a densidade de módulos. Essa técnica adota como base o ILR (VISWANATHAN; CHU, 2004) para modificar o posicionamento. Enquanto o ILR tende a ser destrutivo, o ProLR promove uma transição gradual. Ambos utilizam uma grade de *bins* e gulosamente movem células entre *bins* adjacentes. Ao contrário do ILR, os *bins* no ProLR são pequenos e permanecem inalterados sempre que o método é executado. Para restringir as movimentações de células, a área dos *bins* é limitada a cinco vezes a área média das células.

Para tornar o refinamento local menos destrutivo, é adotada a técnica *Explicit Bin-Blocking* (EBB). Ela consiste de duas componentes: EBB^+ e EBB^-. O EBB^+ restringe a entrada de células em alguns *bins* quando há expectativa do movimento ser prejudicial. Ele é aplicado em alguns *bins* para limitar a densidade. O EBB^- restringe a saída de células de alguns *bins* e encoraja a entrada de células neles. O seu objetivo é atrair células de *bins* com densidade mais alta.

Ao movimentar as células, o ProLR busca otimizar a Fórmula 4.12, que é uma combinação linear do comprimento de fio com a densidade de utilização. Para isso, é feito uma inspeção dos melhores movimentos para cada um dos dois objetivos e, então, é selecionado aquele que não prejudica o outro objetivo. O ProLR executa duas otimizações: o ProLR-w e o ProLR-d, os quais otimizam o comprimento de fio e a densidade de utilização, respectivamente.

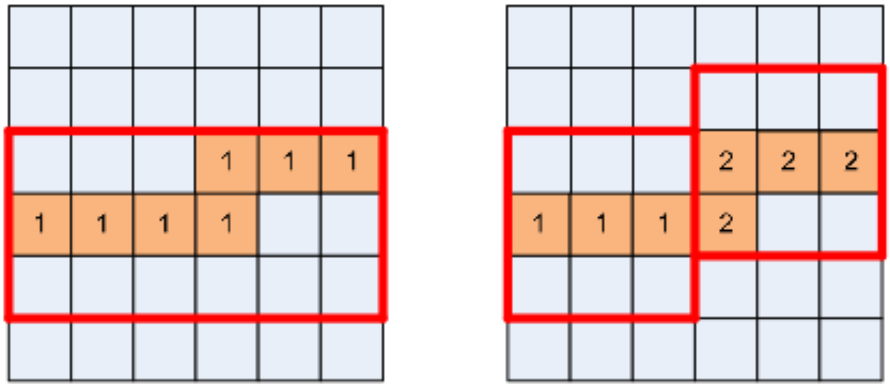
$$Escore(m) = \alpha \cdot \Delta_{HPWL} + \beta \cdot \theta \cdot \Delta_{densidade} \tag{4.12}$$

4.6 Polar, Polar 2.0 e Polar 3.0

O POLAR (LIN et al., 2013) é dividido em três estágios: posicionamento inicial, posicionamento orientado a densidade e legalização local. No posicionamento inicial, o comprimento de fios é minimizado desconsiderando as sobreposições entre as células. Nesse estágio, o circuito é inicialmente modelado utilizando o modelo de conexões híbrido para obter uma solução inicial. Em seguida o comprimento de fio é otimizado iterativamente utilizando o modelo de conexões *Bound to Bound*.

Após o posicionamento inicial, a região de posicionamento é dividida em uma grade uniforme de *bins*. Em cada iteração da etapa de posicionamento orientado a densidade, são identificados os *hotspots*, que são regiões com *bins* espacialmente contíguos e com alta utilização. As regiões identificadas são expandidas através da agregação de *bins* vizinhos, resultando em regiões retangulares que tenham espaço suficiente para acomodar todas as células pertencentes à região não expandida. Podem ocorrer sobreposições entre as regiões. Extensas regiões expandidas são divididas para evitar que as células sofram um deslocamento muito grande dentro da região expandida original. Um exemplo de divisão pode ser visualizada na Figura 4.13, na qual os *bins* com alta utilização estão pintados com a cor laranja e a região expandida é delimitada por um retângulo vermelho. Nesse exemplo, a região expandida retangular é dividida em duas regiões retangulares menores. As novas regiões devem ser capazes de acomodar todas as suas células, então, se necessário, é executada a expansão dessas regiões.

Figura 4.13: Divisão da região expandida



Fonte: (LIN et al., 2013).

Uma vez que as regiões expandias foram determinadas, é aplicado um algoritmo de legalização aproximada inspirado no SimPL. No caso do SimPL, como dito anteriormente na Seção 4.4, a região de posicionamento é alternadamente cortada por linhas verticais ou horizontais e também é segmentada por *stripes* paralelos à linha de corte. Então, as células são espalhadas em cada um dos *stripes*. No POLAR, a região de posicionamento também é cortada por uma linha vertical ou horizontal, a diferença é que a linha corta a área de posicionamento passando pelas bordas dos *bins*. Além disso, a região de posicionamento não é segmentada em *stripes*. Então, durante o processo de particionamento do POLAR, as células não são efetivamente movimentadas como no SimPL, mas apenas atribuídas a uma sub-região. O processo de particionamento acaba quando toda a região de posicionamento foi particionada em seus *bins*. Após esse processo, as células são posicionadas no centro do *bin* ao qual foram atribuídas durante o particionamento.

A técnica de particionamento não mantem as posições relativas entre as células, então, para espalhar as células, foi proposta uma técnica de refinamento global. Nessa técnica, uma determinada célula v_i é movida para a região ótima, enquanto as demais permanecem fixas. Seja N_i o conjunto de redes de conexões associadas à célula v_i. Para cada rede p pertencente à N_i, a sua *Bounding Box* (BB), isto é, o menor retângulo que envolve os pinos da rede, é dada pelas coordenadas $(x_l[p], y_l[p], x_r[p], y_r[p])$. As coordenadas da região ótima são definidas como as medianas de $(x_l[1], x_r[1], x_r[2], x_l[2], \ldots)$ e $(y_l[1], y_r[1], y_r[2], y_l[2], \ldots)$.

Depois que as posições das células foram determinadas, o modelo B2B é atualizado. Para cada célula, é adicionada na matriz de conectividade uma conexão entre a posição original e a posição obtida no processo de legalização aproximada.

O POLAR 2.0 (LIN; CHU, 2014) é uma extensão do POLAR (LIN et al., 2013) proposta para tratar congestionamento de conexões. Esse algoritmo é dividido em três estágios. No primeiro, ele implementa as etapas de posicionamento global do POLAR para gerar uma solução inicial orientada à minimização do comprimento total de fios. Na segunda etapa, são executadas iterativamente em sequência uma técnica de análise de roteamento e uma extensão da legalização aproximada orientada a roteamento proposta no POLAR, que além de detectar regiões com alta concentração de células, também detecta regiões congestionadas. Na última etapa, a solução é legalizada.

Durante a análise de roteamento, é feito uma estimativa da demanda de roteamento utilizando as posições dos pinos obtidas durante o processo de legalização aproximada e o FastRoute (XU; ZHANG; CHU, 2009). Após a análise, serão conhecidas para cada célula as demandas de recursos de roteamento na horizontal (atributo demanda-H) e de recursos na vertical (atributo demanda-V).

A legalização aproximada orientada a roteamento do POLAR 2.0 utiliza, além da informação de demanda de área das células, os atributos demanda-H e demanda-V obtidos durante a análise de roteamento. Então, basicamente, a legalização aproximada distribui essas três demandas baseando-se na oferta de recursos, isto é, área disponível e recursos de roteamento. Na legalização aproximada do POLAR, regiões com alta concentração de células eram expandidas resultando em regiões retangulares com área suficiente para acomodar as células. Na legalização aproximada orientada a roteamento do POLAR 2.0, além de detectar *hotspots* de alta concentração de células, são detectados também *hotspots* de congestionamento de conexões, isto é, pontos congestionados. Então, a região deve ser expandida de modo que a nova região tenha recursos suficientes para o roteamento dos

fios. Essa técnica de legalização aproximada distribui as demandas de roteamento para regiões que têm recursos de roteamento não sendo utilizados.

O POLAR 3.0 (LIN; CHU; WU, 2015) foi proposto para explorar o paralelismo dos algoritmos analíticos quadráticos estado-da-arte. Para isso, foi utilizado como base o POLAR (LIN et al., 2013). A paralelização do posicionamento é feita dividindo as células em partições, de modo que cada partição é tratada individualmente por um núcleo.

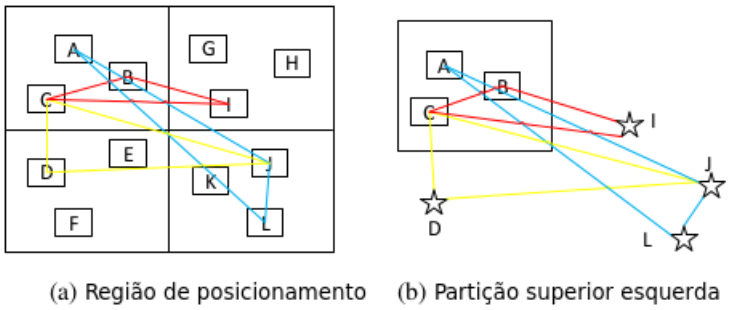
No primeiro estágio do algoritmo, é feito um posicionamento inicial através da resolução do sistema linear. Da mesma forma que o POLAR, o POLAR 3.0 utiliza o modelo de conexões B2B para a construção do sistema linear. Em seguida, é executada a legalização aproximada do POLAR, finalizando o primeiro estágio.

No segundo estágio, as iterações do algoritmo são divididas em uma série de *frames*, que norteiam como será feita a paralelização do posicionamento. Um *frame* é definido como a tupla (xx, yy, dd, n), onde xx e yy indicam quantas partições serão criadas e n o número de iterações consecutivas que irão adotar essa configuração de particionamento. Por exemplo, o *frame* $(4, 4, 0, 5)$ define que serão utilizadas 4×4 partições, totalizando 16, nas próximas 5 iterações. Se dd é 0, as partições são criadas dividindo, primeiro, a região de posicionamento através de linhas horizontais e depois verticais. Caso contrário, se dd é igual a 1, a região é inicialmente cortada por linhas verticais e depois por linhas horizontais. Cada partição é tratada independentemente por uma *thread* diferente, de modo que a execução de cada uma é dividida em quatro passos: construção do sistema linear, adição de forças de espalhamento, resolução do sistema linear e legalização aproximada. O POLAR 3.0 espera todas as *threads* finalizarem para, então, atualizar a posição de cada célula. A execução continua enquanto o HPWL do circuito não convergir.

Cada partição é composta por uma região, um conjunto de células e um conjunto de conexões. Dada uma partição, se uma célula contida nela está conectada com uma outra célula pertencente a outra partição, a célula da outra partição será considerada fixa. Por exemplo, na Figura 4.14, a célula A da partição superior esquerda está conectada ás células J e L da partição inferior direita. Então, para a célula A, as células J e L são consideradas fixas.

Para maximizar o paralelismo, é esperado que as partições tenham um número similar de células, pois, caso elas não estejam equilibradas, a partição com mais células será um gargalo no tempo de execução.

Figura 4.14: Particionamento da região de posicionamento



(a) Região de posicionamento (b) Partição superior esquerda

Fonte: adaptado de (LIN; CHU; WU, 2015).

Se o particionamento é aplicado apenas uma vez, a qualidade do posicionamento não seria satisfatória, pois cada célula estaria restrita à mesma partição, não podendo, assim, migrar para outras partições, o que limita o espaço de soluções. Assim, o POLAR 3.0 introduz o conceito de esquema de particionamento variado. Para prevenir as células de ficarem restritas à mesma partição, são utilizadas mais de uma configuraçãoes de *frames*. Por exemplo, poderia-se adotar, inicialmente, o *frame* $(4, 4, 0, 5)$, que define que durante cinco iterações serão utilizadas 16 partições e depois o *frame* $(1, 1, 0, 1)$, que indica que na próxima iteração haverá apenas uma partição, ou seja, toda a área de posicionamento. Assim, na última iteração, as células poderiam migrar para qualquer região.

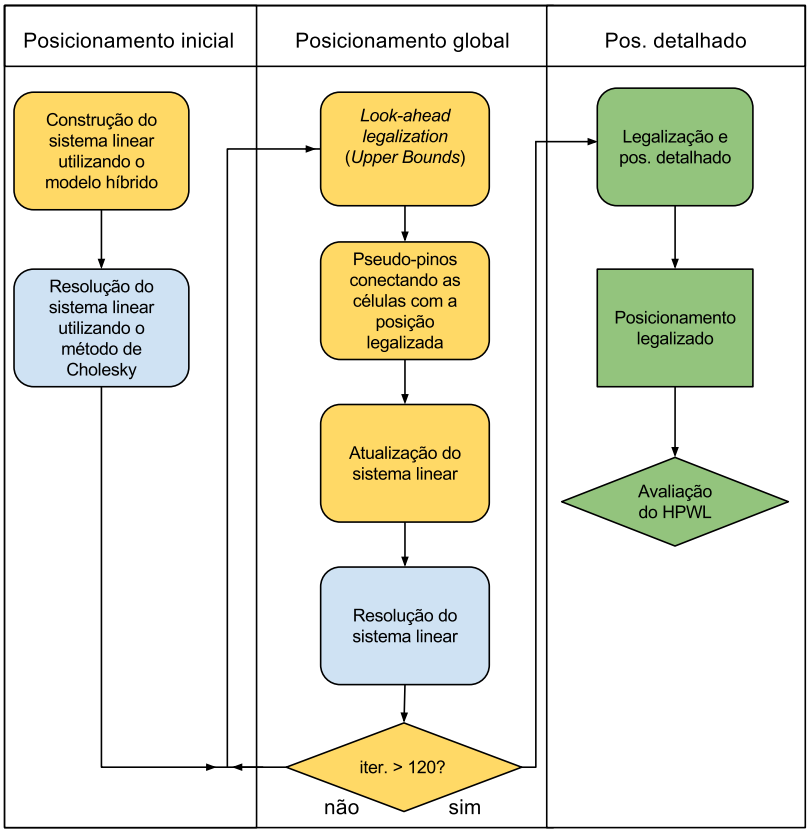
5 FERRAMENTA QUADRÁTICA PARA POSICIONAMENTO GLOBAL

Este capítulo apresenta o desenvolvimento de um algoritmo quadrático para o posicionamento global de células em circuitos integrados. A implementação foi baseada no algoritmo SimPL (KIM; LEE; MARKOV, 2012), que é caracterizado por ser de implementação mais simples que os demais algoritmos no estado-da-arte, porém que apresenta resultados competitivos. A seção 5.1 irá explicar os detalhes de implementação, mostrando as estratégias e recursos utilizados, enquanto que a Seção 5.2 irá apresentar os experimentos realizados e os resultados obtidos.

5.1 Detalhes da Implementação Proposta

A solução proposta nesse trabalho é baseada no algoritmo descrito na Seção 4.4, apresentando algumas diferenças. Enquanto que o SimPL aplica o modelo *Bound to Bound* (SPINDLER; JOHANNES, 2007), a implementação proposta emprega, para simplificar o desenvolvimento, o modelo híbrido (VISWANATHAN; CHU, 2004), descritos nas Seções 3.3.2.4 e 3.3.2.3, respectivamente. A segunda é que a ferramenta não altera as dimensões dos *bins*, também para simplificar a solução. Conforme descrito na Seção 4.4, em cada iteração do posicionamento global, as dimensões dos *bins* são reduzidas por um fator $\beta = 1.06$ para capturar com mais precisão a sobreposição entre as células e para induzir a convergência do algoritmo em iterações posteriores. A última é o resultado do posicionamento que é enviado para a etapa de posicionamento pós-global. No SimPL, após a solução da etapa de posicionamento global convergir, é executado uma última vez o *Look-Ahead Legalization*, de modo que a etapa pós-global recebe um posicionamento *upper-bound*. Na implementação proposta, o *Look-Ahead Legalization* não é executado após a convergência do posicionamento global. Portanto, a etapa pós-global recebe um posicionamento *lower-bound*, isto é, o resultado da solução do sistema linear, pois esse posicionamento é o resultado de um processo de otimização. A Figura 5.1 apresenta o fluxo proposto. Os blocos de cor amarela foram implementados nesse trabalho. Os blocos de cor azul correspondem a recursos disponíveis na ferramenta UPlace e os de cor verde representam as tarefas executadas pelo PlaceUtils (PLACEUTILS, 2016).

Figura 5.1: Fluxo de execução da ferramenta



Fonte: figura elaborada pelo autor.

Conforme apresentado na Figura 5.1, primeiramente é gerada uma solução inicial através do posicionamento quadrático, isto é, através da resolução do sistema linear construído utilizando o modelo híbrido de conexões. Em seguida, as células são espalhadas pela área do *chip*. O espalhamento é feito intercalando a técnica *Look-Ahead Legalization* (LAL) com a atualização e resolução do sistema linear. Conforme descrito na Seção 4.4, o sistema linear é atualizado com as posições geradas pelo LAL, que são utilizadas como âncoras, ou seja, como pontos fixos que serão conectados às respectivas células do posicionamento *lower bound* através de pseudo conexões. Nesse trabalho, não foram implementados algoritmos para as etapas de legalização e posicionamento detalhado, assim, foi empregada a ferramenta PlaceUtils (PLACEUTILS, 2016) para a realização dessas tarefas.

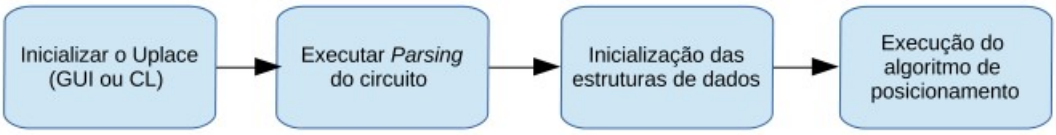
Inicialmente, o critério de parada adotado para a etapa de posicionamento global foi o mesmo proposto no artigo do SimPL, que consiste em monitorar a diferença entre as soluções *lower bound* e *upper bound*. No entanto, para alguns *benchmarks* a convergência nunca ocorria. Por essa razão, o número de iterações foi fixado em 120, que, conforme observado em vários experimentos, é o suficiente para produzir um bom espalhamento de

células, visto que o HPWL não variava significativamente nas observações realizadas.

O fluxo proposto foi implementado em C++11 e inserido na ferramenta UPlace (FLACH et al., 2008–2016), que oferece diversas funcionalidades já implementadas fundamentais para esse projeto, como, por exemplo, um *parser* para *benchmarks* de circuitos no formato *Bookshelf* (BOOKSHELF, 2016), uma estrutura de dados eficiente para armazenar o circuito, um resolvedor de sistemas lineares, interface gráfica para visualizar o resultado do posicionamento e, também, diversos algoritmos.

O fluxo de execução do UPlace é apresentado na Figura 5.2. Basicamente, ele é composto pela inicialização da ferramenta com a interface gráfica ou linha de comando. Em seguida, é feito o *parsing* do circuito e a inicialização das estruturas de dados. Por fim, é executado o algoritmo de posicionamento definido pelo usuário.

Figura 5.2: Fluxo de execução do UPlace



Fonte: figura elaborada pelo autor.

5.2 Experimentos e Resultados

Essa seção apresenta os experimentos realizados com o algoritmo desenvolvido e os resultados obtidos. Os experimentos foram realizados utilizados os *Benchmarks* IBM do ISPD 2004 (BENCHMARKS, 2004) e os *Benchmarks* adaptec1 e bigblue1 do ISPD 2005 (BENCHMARKS, 2005), cujas informações estão disponíveis nas Tabelas 5.1 e 5.2, respectivamente. Para fazer a estimação do HPWL, foi utilizada a ferramenta *PlaceUtils*.

A ferramenta foi compilada com o g++ (GCC) versão 4.9.2. Os experimentos foram realizados em uma máquina com a seguinte configuração:

- Sistema operacional Ubuntu 15.04
- kernel Linux 3.19.0-59-generic
- Processador Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
- 32 GB de memória RAM

Tabela 5.1: Informações dos benchmarks IBM do ISPD 2004 (BENCHMARKS, 2004)

Circuito	Células	*Pads*	Redes	Pinos	Bandas
IBM01	12506	246	14111	50566	96
IBM02	19342	259	19584	1199	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 5.2: Informações dos benchmarks do ISPD 2005 (BENCHMARKS, 2005)

Circuito	Objetos	Móveis	Fixos	Redes	I/Os periféricos
adaptec1	211447	210904	543	221142	480
adaptec2	255023	254457	566	266009	407
adaptec3	451650	450927	723	466758	0
adaptec4	496045	494716	1329	515951	0
bigblue1	278164	277604	560	284479	528
bigblue2	557866	534782	23084	577235	0
bigblue3	1096812	1095519	1293	1123170	0
bigblue4	2177353	2169183	8170	2229886	0

Inicialmente, foram realizados dois experimentos. No primeiro, foi gerado uma solução inicial através do posicionamento quadrático. Em seguida, a ferramenta PlaceUtils foi utilizada para legalizar o posicionamento gerado. No segundo experimento, foi aplicado o algoritmo desenvolvido, que gera o mesmo posicionamento quadrático inicial e em seguida espalha as células. Após o espalhamento, a solução foi legalizada utilizando o PlaceUtils. Para avaliar os resultados, foram feitas duas análises. A primeira compara, para cada *benchmark*, o HPWL obtido na etapa de legalização nos dois experimentos e a segunda compara os tempos de execução do legalizador. A Tabela 5.3 apresenta o HPWL das soluções legalizadas nos dois experimentos.

Tabela 5.3: Resultados em HPWL ($\times 10^6$) para os *Benchmarks* IBM, adaptec1 e bigblue1 nos dois experimentos

Circuito	HPWL experimento 1	HPWL experimento 2
IBM01	11,28	3,77
IBM10	140,23	30,89
IBM17	740,47	111,44
adaptec1	462,63	156,99

A Tabela 5.4 apresenta os tempos de execução do legalizador para cada *benchmark* nos dois experimentos.

Tabela 5.4: Tempo de execução em segundos do legalizador nos dois experimentos

Circuito	t. exec. experimento 1	t. exec. experimento 2
IBM01	16,12	0,53
IBM10	584,08	4,32
IBM17	5011	10,76
adaptec1	29774	659,13

É possível observar, a partir da Tabela 5.3, que as estimativas de comprimento de fio obtidas no experimento 2 são significativamente menores que as obtidas no experimento 1. Além disso, a Tabela 5.4 mostra que, no experimento 2, a execução do legalizador foi muito mais rápida. Esses resultados evidenciam a importância do posicionamento global, pois uma boa distribuição de células na área do *chip* resulta em bons resultados em termos de HPWL, além de facilitar a tarefa do legalizador.

A seguir, serão apresentados os resultados do posicionamento detalhado obtidos pela ferramenta implementada, comparando-os com os do algoritmo FastPlace (VISWANATHAN; CHU, 2004) e SimPL (KIM; LEE; MARKOV, 2012). A Tabela 5.5 faz uma comparação dos resultados do posicionamento detalhado obtidos pela ferramenta implementada com os resultados obtidos pelo FastPlace. Já a Tabela 5.6 faz a mesma comparação com o algoritmo SimPL para os *benchmarks* adaptec1 e bigblue1. Para facilitar a visualização dos resultados nas tabelas, a ferramenta implementada será chamada de HPlace.

Tabela 5.5: Comparação dos resultados obtidos pela ferramenta implementada com o algoritmo FastPlace

Circuito	HPWL ($\times 10^6$) FastPlace	HPWL ($\times 10^6$) HPlace
IBM01	1,91	3,43
IBM02	4,02	6,42
IBM03	5,45	7,89
IBM04	6,63	10,23
IBM05	10,96	14,92
IBM06	5,55	8,54
IBM07	9,56	15,56
IBM08	10,01	19,78
IBM09	11,26	18,78
IBM10	19,31	28,49
IBM11	16,03	27,53
IBM12	25,04	35,66
IBM13	19,46	30,42
IBM14	36,09	61,72
IBM15	45,21	72,01
IBM16	48,43	73,59
IBM17	68,09	104,67
IBM18	46,89	113,78

Tabela 5.6: Comparação dos resultados obtidos pela ferramenta implementada com o algoritmo SimPL

Circuito	HPWL ($\times 10^6$) SimPL	HPWL ($\times 10^6$) HPlace
adaptec1	77,73	148,12
bigblue1	97,42	147,62

Para todos os *bechmarks* utilizados nos experimentos, os resultados obtidos foram piores em comparação com o FastPlace e SimPL. Algumas conjeturas que podem ser feitas para explicar os resultados seriam que o FastPlace apresenta um estágio que miniminiza o HPWL diretamente, o *Iterative Local Refinement* e que o SimPL adota o modelo de conexões B2B, apresentado na seção 3.3.2.4, que permite que as ferramentas utilizem o HPWL como métrica de otimização. Portanto, a investigação dessas conjeturas será feita em trabalhos futuros.

A Tabela 5.7 apresenta o impacto que a legalização teve no posicionamento global e o impacto que o posicionamento detalhado teve na solução legalizada.

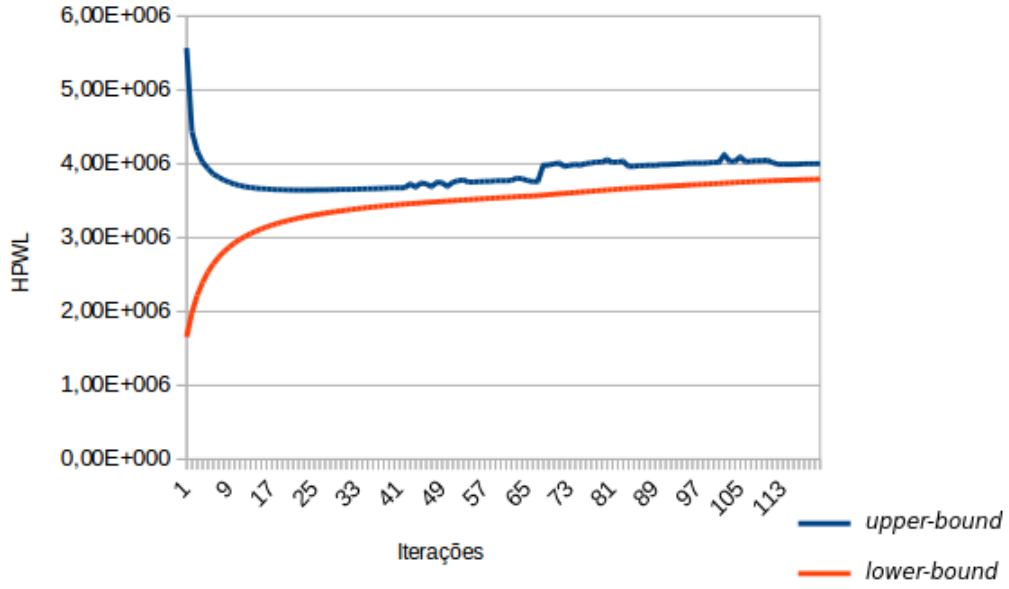
Tabela 5.7: Impactos da legalização e posicionamento detalhado

Circuito	HPWL ($\times 10^6$) pos. global	HPWL ($\times 10^6$) legalização	Impacto legalização	HPWL ($\times 10^6$) pos. detalhado	Impacto pos. det.
IBM01	3,79	3,77	0,99	3,43	0,90
IBM02	6,43	7,02	1,09	6,42	0,91
IBM03	7,96	8,70	1,09	7,89	0,90
IBM04	9,73	11,22	1,15	10,23	0,91
IBM05	15,02	15,85	1,05	14,92	0,94
IBM06	8,32	9,58	1,15	8,54	0,89
IBM07	17,27	17,22	0,99	15,56	0,90
IBM08	19,35	21,60	1,11	19,78	0,91
IBM09	18,56	20,41	1,09	18,78	0,92
IBM10	29,29	31,03	1,05	28,54	0,91
IBM11	27,51	30,09	1,09	27,71	0,92
IBM12	37,30	38,99	1,04	36,23	0,92
IBM13	30,03	34,06	1,13	31,18	0,91
IBM14	63,33	66,25	1,04	61,19	0,92
IBM15	69,63	79,56	1,14	74,02	0,93
IBM16	74,46	83,00	1,11	76,47	0,92
IBM17	110,74	114,95	1,03	107,43	0,93
IBM18	113,05	123,55	1,09	115,30	0,93
adaptec1	124,28	156,99	1,26	148,97	0,94
bigblue1	146,67	156,00	1,06	147,62	0,94

Pela Tabela 5.7, é possível observar que para os *benchmarks* IBM01 e IBM07 o HPWL da legalização foi menor que o do posicionamento global. Uma possível explicação para esse fenômeno é que a convergência do posicionamento global foi atingida antes de 120 iterações, ou seja, as células passaram a oscilar em torno das posições onde elas deveriam estar, piorando o resultado. Além disso, é possível observar que para o *benchmark* adaptec1 o impacto da legalização foi mais expressivo em relação aos demais. Visto que o adaptec1 possui macro blocos espalhados em toda área do *chip*, o legalizador tem menor liberdade para posicionar as células, portanto, o HPWL após a legalização tende a ser maior. O *benchmark* bigblue1 também possui macro blocos, porém eles estão concentrados na borda do circuito, portanto, o impacto da legalização não foi tão grande como no adaptec1.

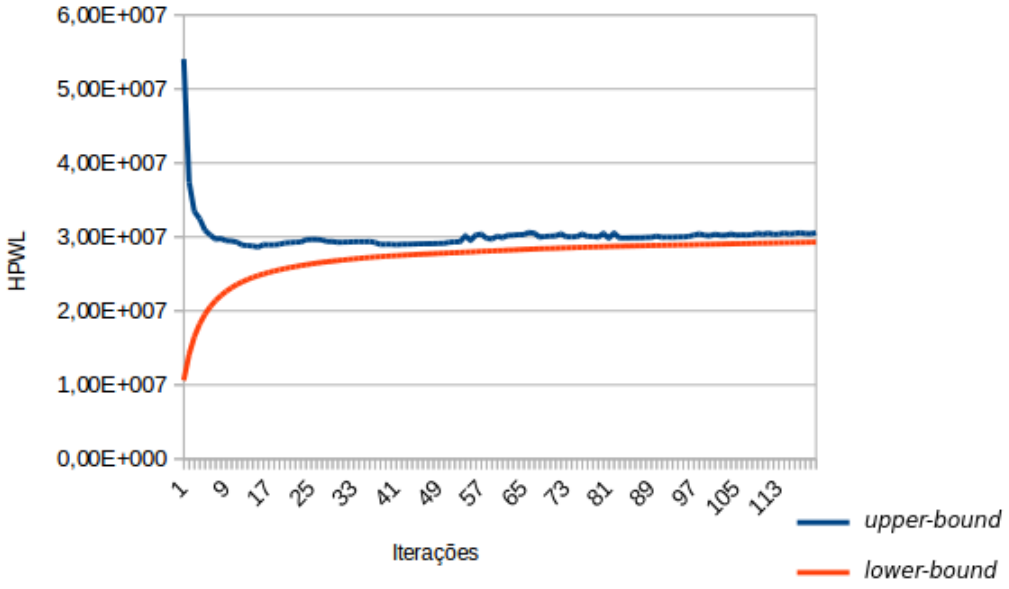
Além dos resultados apresentados na Tabela 5.7, também foram obtidas as curvas de convergência dos posicionamentos *upper-bound* e *lower-bound* da etapa de posicionamento global. Para isso, foram escolhidos os *benchmarks* IBM01, IBM10, IBM17 e adaptec1. As curvas podem ser observadas nas figuras 5.3, 5.4, 5.5 e 5.6. Como esperado, é possível observar que a medida que são executadas as iterações do algoritmo de *Look-Ahead Legalization*, os dois posicionamentos tendem a convergir.

Figura 5.3: Gráfico de convergência das curvas de *lower-bound* e *upper-bound* do circuito IBM01



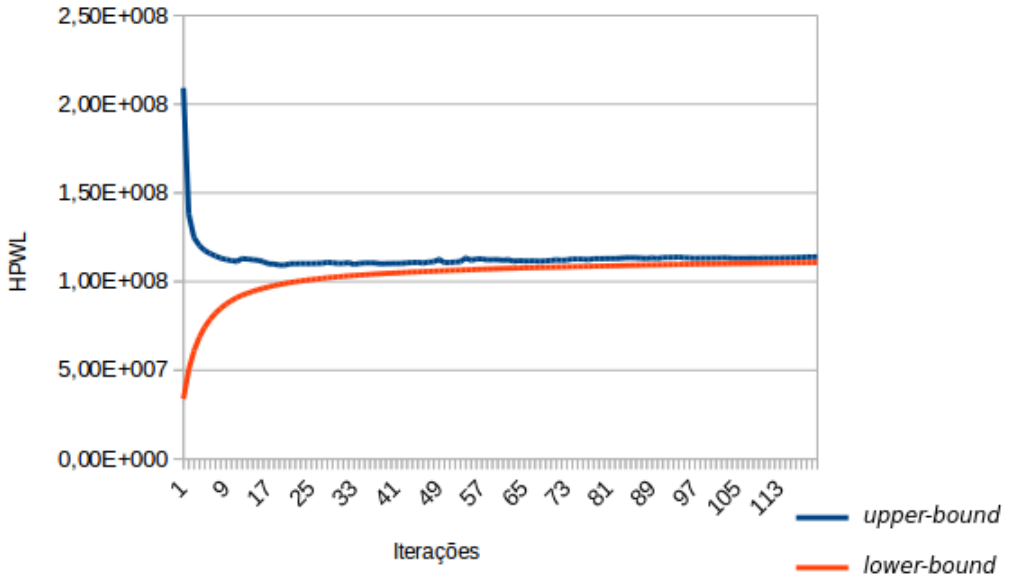
Fonte: figura elaborada pelo autor.

Figura 5.4: Gráfico de convergência das curvas de *lower-bound* e *upper-bound* do circuito IBM10



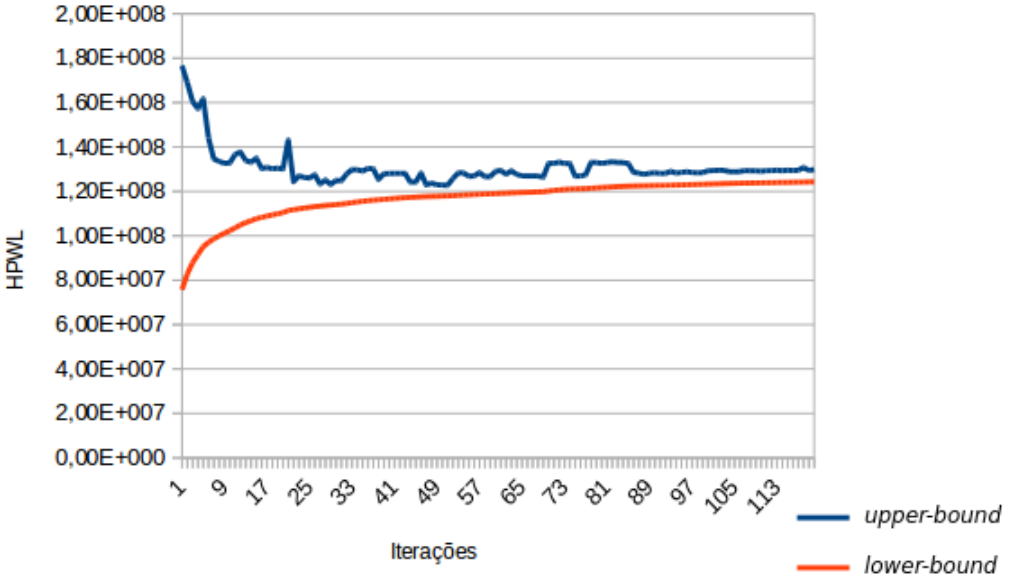
Fonte: figura elaborada pelo autor.

Figura 5.5: Gráfico de convergência das curvas de *lower-bound* e *upper-bound* do circuito IBM17



Fonte: figura elaborada pelo autor.

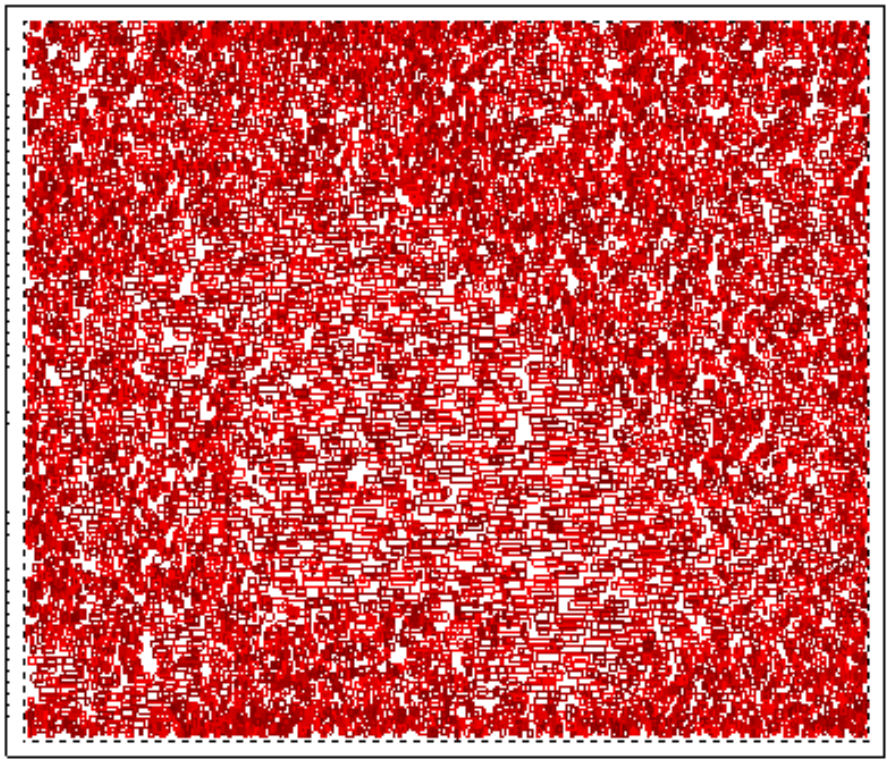
Figura 5.6: Gráfico de convergência das curvas de *lower-bound* e *upper-bound* do circuito adaptec1



Fonte: figura elaborada pelo autor.

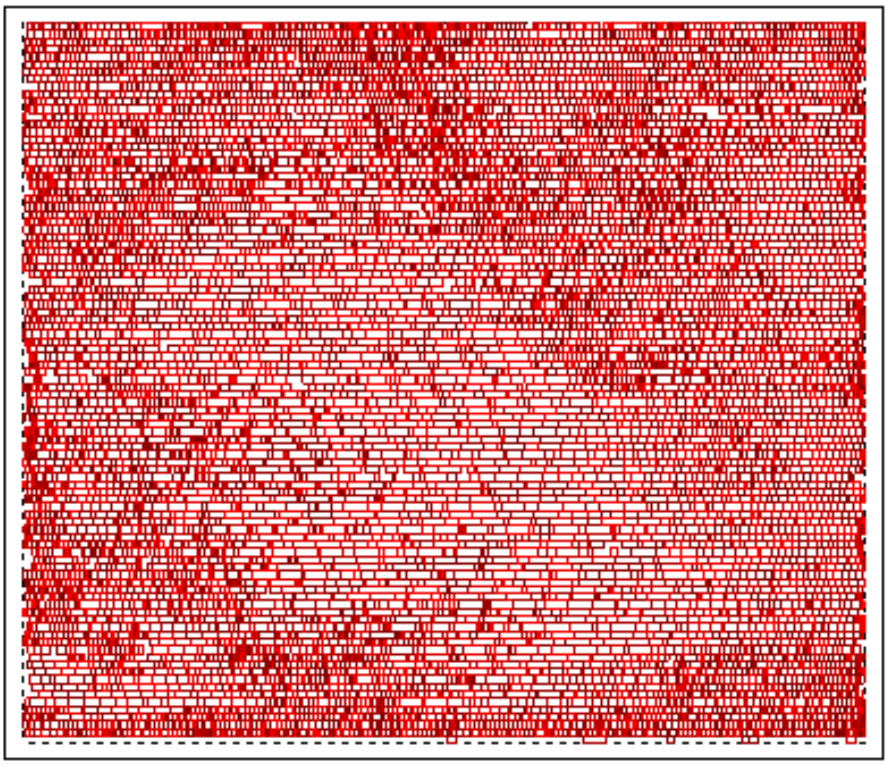
Foram geradas imagens que ilustram o resultados do posicionamento global feito pelo algoritmo desenvolvida nesse trabalho e da solução legalizada gerada pela ferramenta *PlaceUtils*. As próximas imagens ilustram os posicionamentos obtidos para os *benchmarks* IBM01, IBM10, IBM17 e adatpec1.

Figura 5.7: Resultado do posicionamento global para o circuito IBM01 utilizando a ferramenta desenvolvida



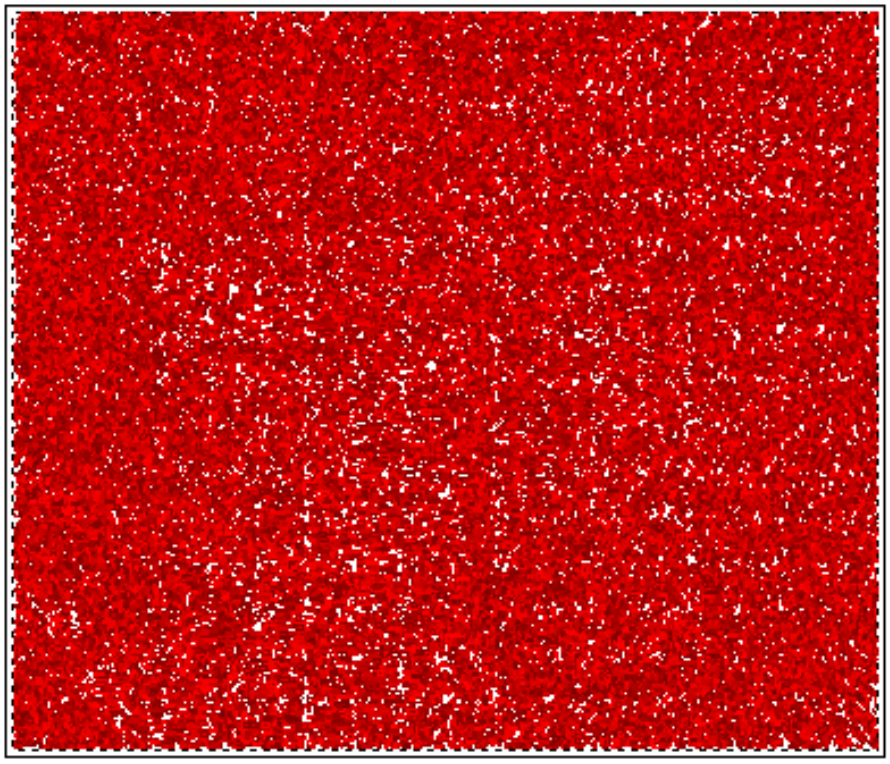
Fonte: figura elaborada pelo autor.

Figura 5.8: Resultado da legalização do *PlaceUtils* para o circuito IBM01



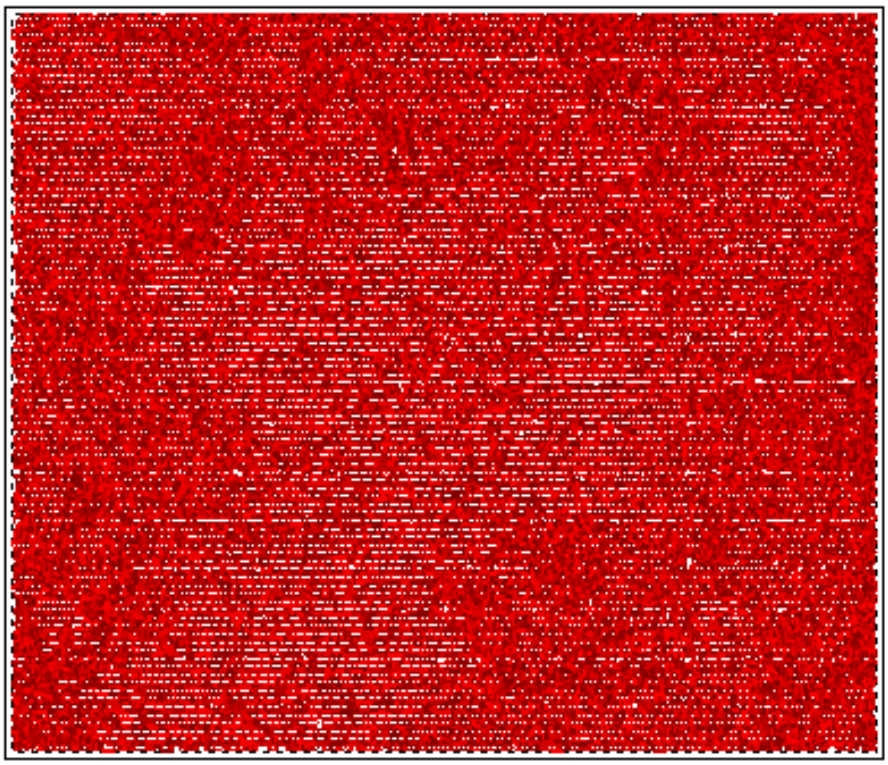
Fonte: figura elaborada pelo autor.

Figura 5.9: Resultado do posicionamento global para o circuito IBM10 utilizando a ferramenta desenvolvida



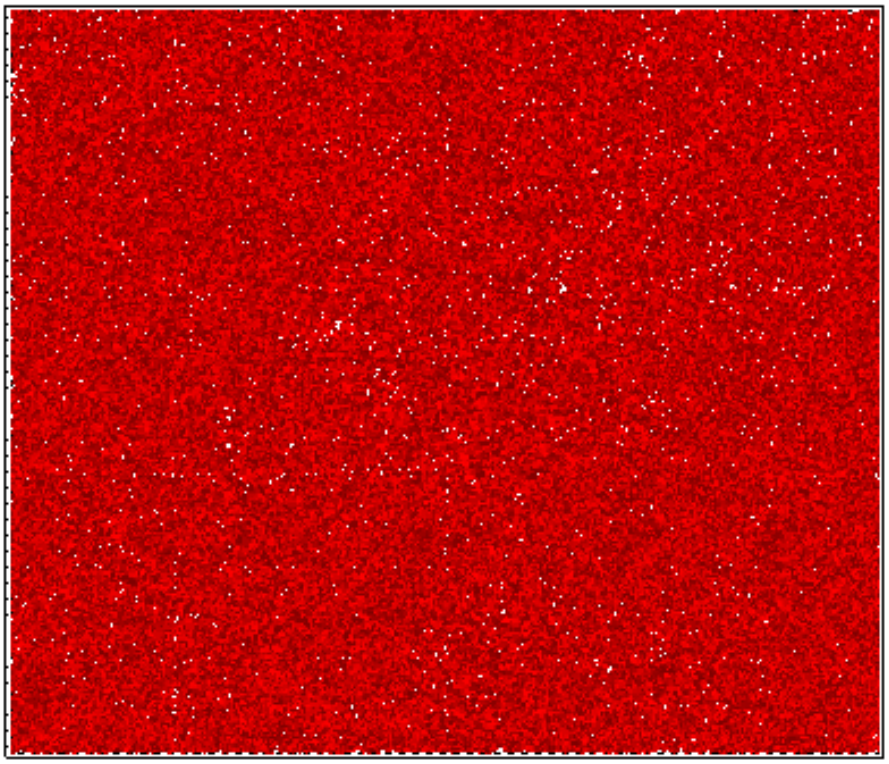
Fonte: figura elaborada pelo autor.

Figura 5.10: Resultado da legalização do *PlaceUtils* para o circuito IBM10



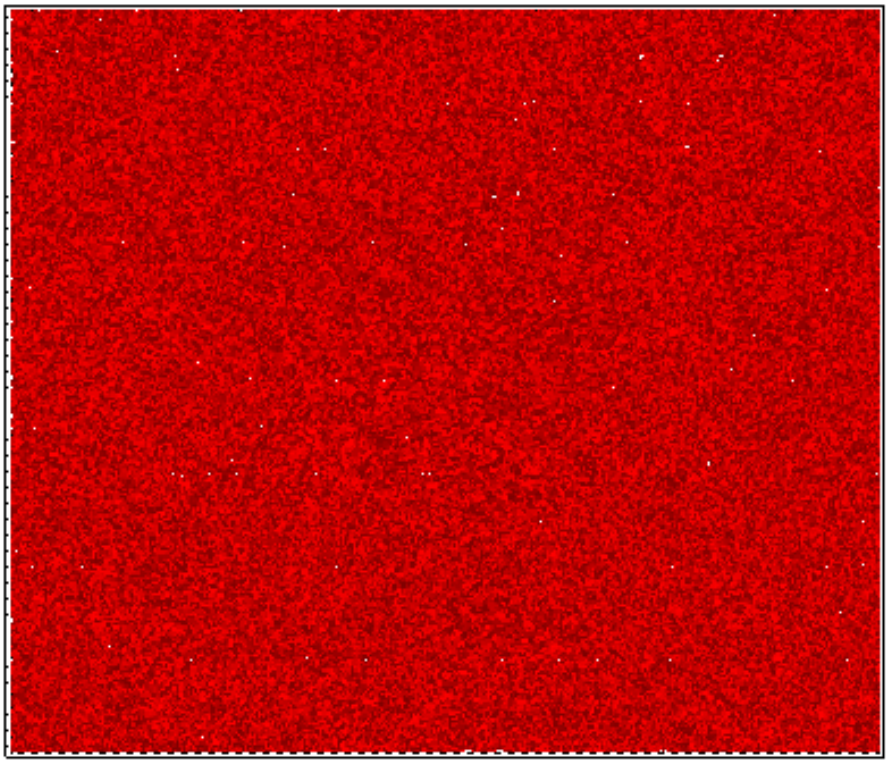
Fonte: figura elaborada pelo autor.

Figura 5.11: Resultado do posicionamento global para o circuito IBM17 utilizando a ferramenta desenvolvida



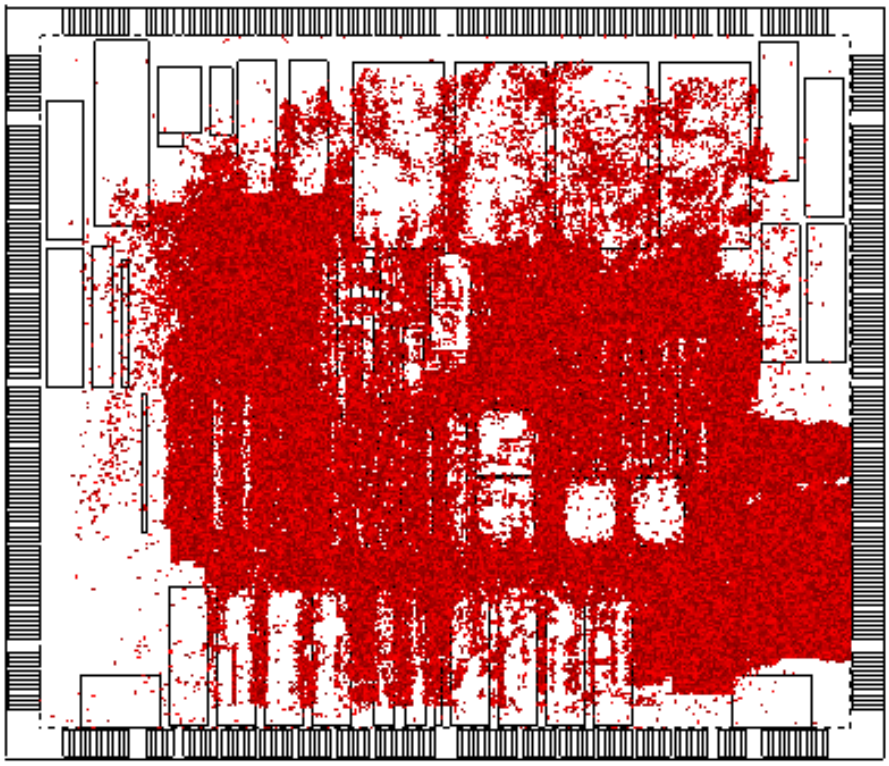
Fonte: figura elaborada pelo autor.

Figura 5.12: Resultado da legalização do *PlaceUtils* para o circuito IBM17



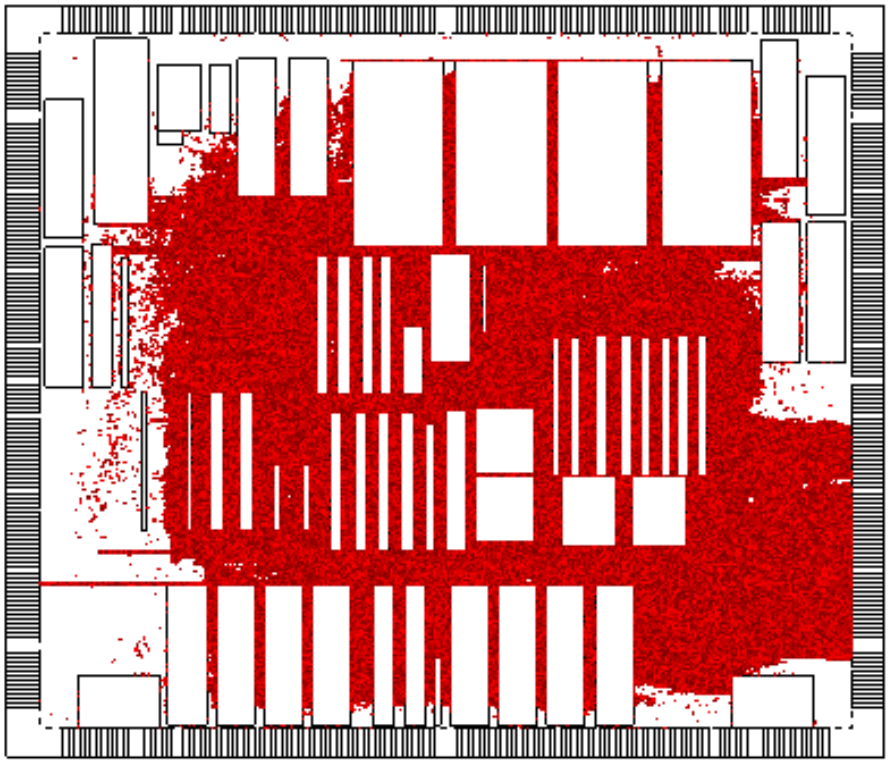
Fonte: figura elaborada pelo autor.

Figura 5.13: (Resultado do posicionamento global para o circuito adaptec1 utilizando a ferramenta desenvolvida



Fonte: figura elaborada pelo autor.

Figura 5.14: (Resultado da legalização do *PlaceUtils* para o circuito adaptec1

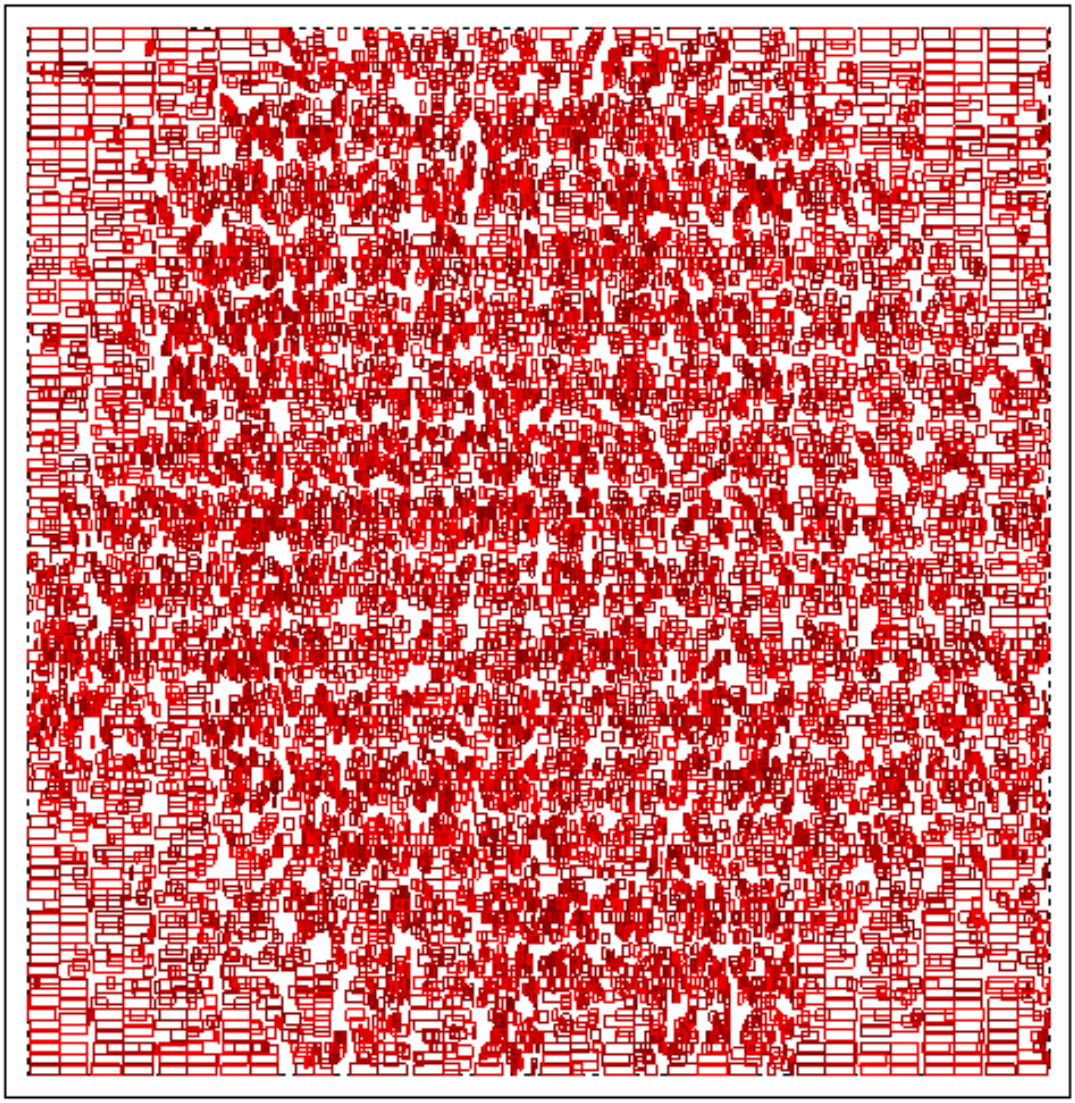


Fonte: figura elaborada pelo autor.

As imagens anteriores mostram que para os circuitos IBM01, IBM10 e IBM17 a ferramenta desenvolvida foi capaz de realizar um bom espalhamento, ocupando bem a área disponível para posicionamento. Já para o circuito adaptec1, apesar do bom espalhamento obtido, ainda restaram regiões livres para acomodar células. Também é possível observar que, visualmente, os resultados do posicionamento global e legalização são muito próximos, o que evidencia que a legalização não perturbou de forma expressiva a solução obtida pela ferramenta desenvolvida, conforme os dados apresentados na Tabela 5.7. Visto que a legalização deve modificar o mínimo possível o posicionamento global, é possível concluir que o trabalho desenvolvido realiza um bom espalhamento de células.

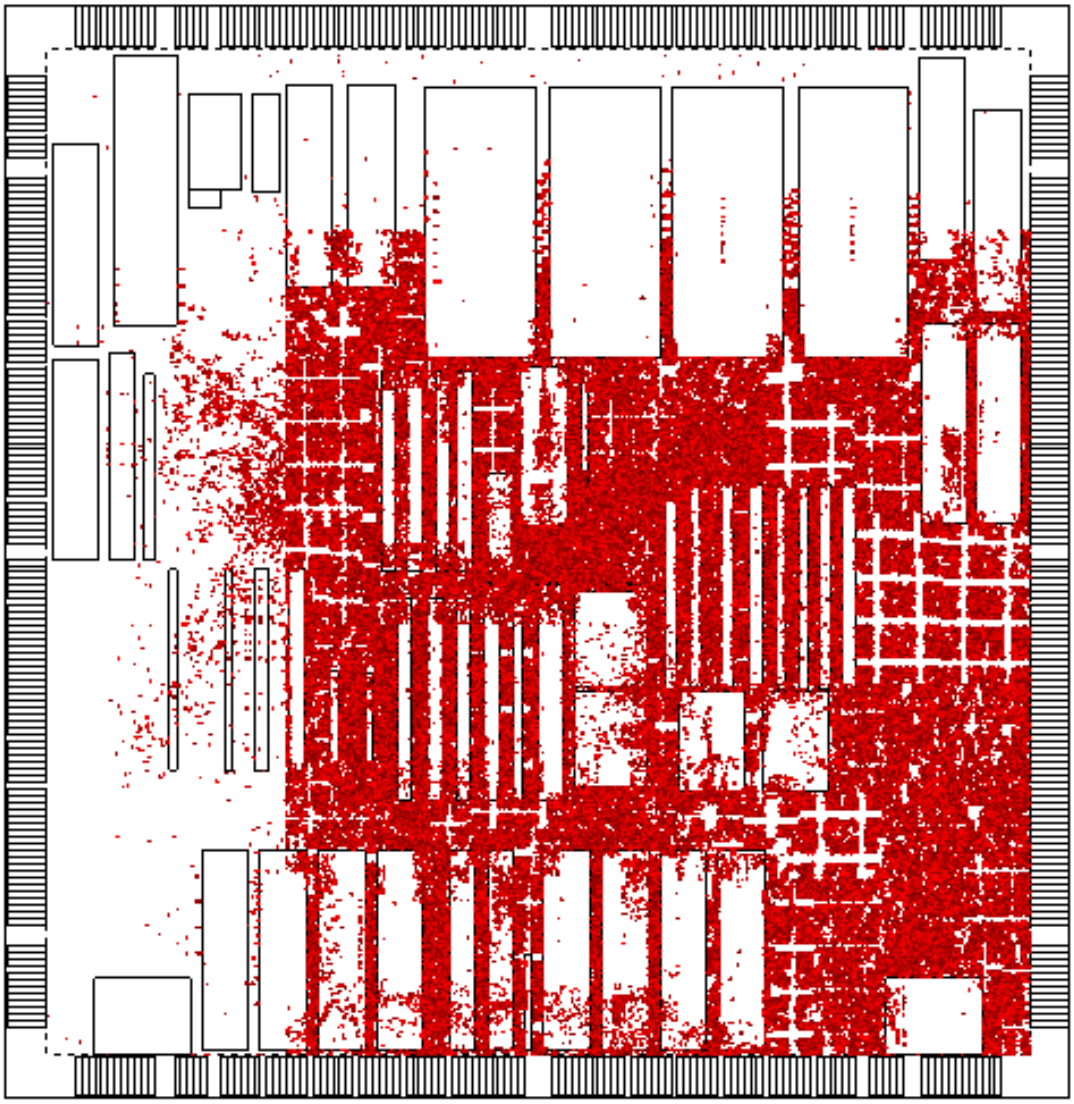
Para fins de visualização do comportamento do *Look-Ahead Legalization* implementado, as Figuras 5.15 e 5.16 abaixo ilustram o resultado do LAL na primeira iteração do posicionamento global para os circuitos IBM01 e adaptec1, respectivamente.

Figura 5.15: (Resultado do *Look-Ahead Legalization* na primeira iteração do posicionamento global para o circuito IBM01



Fonte: figura elaborada pelo autor.

Figura 5.16: (Resultado do *Look-Ahead Legalization* na primeira iteração do posicionamento global para o circuito adaptec1



Fonte: figura elaborada pelo autor.

A Figura 5.15 mostra que nas regiões onde foram posicionadas as maiores células, foi obtido um resultado bem próximo de uma legalização, visto que as células foram posicionadas lado a lado, como acontece quando o legalizador determina a localização das células nas bandas de posicionamento. Para as células menores, ocorreram sobreposições.

A Figura 5.16 revela que, para o circuito adaptec1, o LAL produziu retângulos de aglomerados de células. Um trabalho futuro será investigar as causas desse efeito e verificar como ele impacta na qualidade da solução.

Por fim, a Tabela 5.8 mostra o tempo de execução da ferramenta para os *benchmarks* utilizados nos experimentos.

Tabela 5.8: Tempo de execução do posicionamento global em segundos

Circuito	**tempo exec. pos. global (s)**
IBM01	20,9484
IBM02	33,7458
IBM03	34,9032
IBM04	39,3943
IBM05	42,4931
IBM06	56,9568
IBM07	94,2919
IBM08	92,6574
IBM09	96,8681
IBM10	119,281
IBM11	105,255
IBM12	119,401
IBM13	141,019
IBM14	291,019
IBM15	307,861
IBM16	344,293
IBM17	395,439
IBM18	419,202
adaptec1	544,762
bigblue1	933,088

6 CONCLUSÕES

Neste trabalho, foi realizado o estudo do problema de posicionamento de células em circuitos integrados e a implementação de uma ferramenta de posicionamento global analítica quadrática.

Os algoritmos analíticos quadráticos modelam o circuito como um sistema de molas, onde as células móveis são pontos adimensionais móveis e as células fixas e pinos de entrada/saída são pontos adimensionais fixos. Utilizando um determinado modelo de conexões entre células, é construído um sistema linear, cuja solução representa o ponto de equilíbrio dos pontos adimensionais (células), isto é, a força resultante em cada um deles é zero. Para essa configuração, o comprimento de fios do circuito é mínimo, no entanto, há um grande número de sobreposições entre as células e elas não estão alinhadas nas bandas de posicionamento, o que torna a solução inviável. Por essa razão, os algoritmos analíticos quadráticos aplicam iterativamente forças de espalhamento, de modo que as sobreposições sejam gradativamente reduzidas. O que diferencia os algoritmos analíticos quadráticos é a forma como são computadas as forças de espalhamento.

A ferramenta desenvolvida modela as conexões utilizando o modelo híbrido, proposto em (VISWANATHAN; CHU, 2004), e, para espalhar as células pela área do *chip*, foi adotada a técnica *Look-Ahead Legalization* (LAL) do SimPL (KIM; LEE; MARKOV, 2012), que inspirou o surgimento de outros algoritmos, como o MAPLE (KIM et al., 2012), o POLAR (LIN et al., 2013), POLAR 2.0 (LIN; CHU, 2014) e POLAR 3.0 (LIN; CHU; WU, 2015).

O algoritmo de posicionamento global implementado neste trabalho foi codificado utilizando a linguagem de programação C++ 11 e foi integrado ao UPlace. Para verificar o seu funcionamento, foram fornecidas à ferramenta os *benchmarks* IBM do ISPD 2004 e o *benchmark* adaptec1 do ISPD 2005. Apesar do tempo de execução não ter sido alto e de as células terem sido bem espalhadas para todos os circuitos de teste, as soluções obtidas tiveram o comprimento total de fios, estimado utilizando a métrica *half-perimeter wirelength*, maior que as estimativas obtidas nos algoritmos FastPlace e SimPL. As conjeturas feitas para explicar os resultados piores foram que o FastPlace possui uma etapa de refinamento que miniminiza diretamente o HPWL e que o SimPL utiliza como modelo de redes o *Bound to bound*, que permite que as ferramentas de posicionamento utilizem o HPWL como métrica de otimização. Portanto, em trabalhos futuros, essas hipóteses serão investigadas. Outro ponto que será explorado futuramente é um critério de parada do po-

sicionamento global, visto que nesse trabalho foi utilizado um número fixo de iterações, o que não é uma solução interessante.

É possível observar nas Figuras 5.7 a 5.14 que, visualmente, os resultados do posicionamento global e legalização são próximos, o que mostra que o legalizador utilizado nos experimentos não modificou demasiadamente a solução obtida pela ferramenta implementada. Além disso, a Tabela 5.7 mostra que a legalização não teve um impacto expressivo no posicionamento global. Portanto, é possível concluir que o espalhamento de células realizado pela etapa de posicionamento global foi bom, visto que o legalizador deve perturbar o mínimo possível o posicionamento global. Ainda na Tabela 5.7, é possível notar que o circuito de teste que teve o maior impacto da legalização foi o adaptec1, pois ele possui macro blocos distribuídos por toda a área do *chip*. Portanto, o legalizador tem menos liberdade para posicionar as células, o que explica o porquê de o aumento no HPWL ser mais expressivo em relação aos outros circuitos de teste.

A Figura 5.16 mostra que para o circuito de teste adaptec1, o *Look-Ahead Legalization* formou retângulos de aglomerados de células. Portanto, futuramente, esse fenômeno será investigado para avaliar o seu impacto no resultado final.

Por fim, o problema de posicionamento de células continua sendo explorado pela academia e indústria, pois a cada dia surgem novos desafios devido ao avanço tecnológico na indústria de microeletrônica. Assim, busca-se desenvolver novas heurísticas que sejam capazes de lidar com as restrições impostas e gerar resultados melhores.

REFERÊNCIAS

ALPERT, C. et al. A semi-persistent clustering technique for vlsi circuit placement. In: **Proceedings of the 2005 International Symposium on Physical Design**. New York, NY, USA: ACM, 2005. (ISPD '05), p. 200–207.

BENCHMARKS. **ISPD04 IBM Standard Cell Benchmarks with Pads**. 2004. Available from Internet: <http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.public.iastate.edu/~nataraj/ISPD04_Bench.html>. Accessed in: 22 de maio de 2016.

BENCHMARKS. **ISPD 2005 Placement Benchmarks**. 2005. Available from Internet: <http://archive.sigda.org/ispd2005/contest.htm>. Accessed in: 27 de maio de 2016.

BOOKSHELF. **GSRC Bookshelf Format for placement instances**. 2016. Available from Internet: <http://vlsicad.eecs.umich.edu/BK/ISPD06bench/BookshelfFormat.txt>. Accessed in: 21 de maio de 2016.

CHURIWALA, S.; GARG, S. **Principles of VLSI RTL Design: A Practical Guide**. [S.l.]: Springer New York, 2011. (SpringerLink : Bücher).

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: **Proceedings of the 19th Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 1982. (DAC '82), p. 175–181.

FLACH, G.; JOHANN, M.; REIS, R. Quadratic placement with single-iteration linear system solver. In: **Proceedings of the 24th Symposium on Integrated Circuits and Systems Design**. New York, NY, USA: ACM, 2011. (SBCCI '11), p. 109–112.

FLACH, G. et al. **UPlace**. 2008–2016.

FLACH, G. A. **Discrete Gate Sizing and Timing-Driven Detailed Placement for the Design of Digital Circuits**. Thesis (PhD) — UFRGS, Porto Alegre, Brasil, 2015.

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

HENTSCHKE, R.; JOHANN, M.; REIS, R. New trends and technologies in computer-aided learning for computer-aided design: Ifip tc10 working conference: Edutech 2005, october 20–21, perth, australia. In: ____. Boston, MA: Springer US, 2005. chp. Blue Macaw: A Didactic Placement Tool Using Simulated Annealing, p. 37–47.

HENTSCHKE, R. F. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. Dissertation (Master) — UFRGS, Porto Alegre, 2002.

INTEL. **Intel Microprocessor Quick Reference Guide**. 2008. Available from Internet: <http://www.intel.com/pressroom/kits/quickreffam.htm#i486>. Accessed in: 19 de abril de 2016.

KAHNG, A. et al. **VLSI Physical Design: From Graph Partitioning to Timing Closure**. 1. ed. [S.l.]: Springer Netherlands, 2011.

KERNIGHAN, B.; LIN, S. An Efficient Heuristic Procedure for Partitioning Graphs. **The Bell Systems Technical Journal**, v. 49, n. 2, 1970. Disponível em: https://www.cs.utexas.edu/ pingali/CS395T/2009fa/papers/kl.pdf. Acessado em: 4 de abril de 2016.

KIM, M.-C.; LEE, D.-J.; MARKOV, I. L. Simpl: An effective placement algorithm. **Transactions on Computer-Aided Design of Integrated Circuits and Systems**, IEEE Press, Piscataway, NJ, USA, v. 31, n. 1, p. 50–60, jan. 2012.

KIM, M.-C. et al. Maple: Multilevel adaptive placement for mixed-size designs. In: **Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design**. New York, NY, USA: ACM, 2012. (ISPD '12), p. 193–200.

LAVAGNO, L.; SCHEFFER, L.; MARTIN, G. **EDA for IC Implementation, Circuit Design, and Process Technology**. [S.l.]: CRC Press, 2006. (Electronic Design Automation for Integrated Circuits Hdbk).

LIN, T.; CHU, C. Polar 2.0: An effective routability-driven placer. In: **Proceedings of the 51st Annual Design Automation Conference**. New York, NY, USA: ACM, 2014. (DAC '14), p. 123:1–123:6.

LIN, T. et al. Polar: Placement based on novel rough legalization and refinement. In: **Proceedings of the International Conference on Computer-Aided Design**. Piscataway, NJ, USA: IEEE Press, 2013. (ICCAD '13), p. 357–362.

LIN, T.; CHU, C.; WU, G. Polar 3.0: An ultrafast global placement engine. In: **Proceedings of the IEEE/ACM International Conference on Computer-Aided Design**. Piscataway, NJ, USA: IEEE Press, 2015. (ICCAD '15), p. 520–527.

MARKOV, I. L.; HU, J.; KIM, M.-C. Progress and challenges in vlsi placement research. In: **Proceedings of the International Conference on Computer-Aided Design**. New York, NY, USA: ACM, 2012. (ICCAD '12), p. 275–282.

MO, F.; TABBARA, A.; BRAYTON, R. K. A force-directed macro-cell placer. In: **Proceedings of the 2000 IEEE/ACM International Conference on Computer-aided Design**. Piscataway, NJ, USA: IEEE Press, 2000. (ICCAD '00), p. 177–181.

MONTEIRO, J. L. **Algoritmo de Posicionamento Analítico Detalhado Guiado a Caminhos Críticos**. Dissertation (Master) — UFRGS, Porto Alegre, Brasil, 2014.

PALNITKAR, S. **Verilog HDL: A Guide to Digital Design and Synthesis**. [S.l.]: SunSoft Press, 1996. (Verilog HDL: A Guide to Digital Design and Synthesis, v. 1).

PLACEUTILS. **Executable Placement Utilities**. 2016. Available from Internet: <http://http://vlsicad.eecs.umich.edu/BK/PlaceUtils/>. Accessed in: 22 de maio de 2016.

RABAEY, J.; CHANDRAKASAN, A.; NIKOLIC, B. **Digital integrated circuits: a design perspective**. 2. ed. Upper Saddle River, New Jersey: Pearson Education, 2003. (Prentice Hall electronics and VLSI series).

REIS, R. A. d. L. et al. **Concepção de circuitos integrados**. Porto Alegre, Brasil: Editora Sagra Luzzatto, 2002. (Instituto de Informática da UFRGS).

ROSSUM, M. V. Advanced nanoscale ulsi interconnects: Fundamentals and applications. In: ____. New York, NY: Springer New York, 2009. chp. MOS Device and Interconnects Scaling Physics, p. 15–38.

ROY, J. A. et al. Capo: Robust and scalable open-source min-cut floorplacer. In: **Proceedings of the 2005 International Symposium on Physical Design**. New York, NY, USA: ACM, 2005. (ISPD '05), p. 224–226.

SHERWANI, N. **Algorithms for VLSI Physical Design Automation**. 3. ed. Norwell, Massachusetts, USA: Kluwer Academic Publishers, 1999.

SPINDLER, P.; JOHANNES, F. M. Kraftwerk: a fast and robust quadratic placer using an exact linear net model. In: **Modern Circuit Placement**. [S.l.]: Springer, 2007. p. 59–93.

SPINDLER, P.; SCHLICHTMANN, U.; JOHANNES, F. M. Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model. **Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on**, IEEE, v. 27, n. 8, p. 1398–1411, 2008.

STENZ, G. et al. Performance optimization by interacting netlist transformations and placement. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, v. 19, n. 3, p. 350–358, 7 2000.

TAGHAVI, T.; YANG, X.; CHOI, B.-K. Dragon2005: Large-scale mixed-size placement tool. In: **Proceedings of the 2005 International Symposium on Physical Design**. New York, NY, USA: ACM, 2005. (ISPD '05), p. 245–247.

VISWANATHAN, N.; CHU, C. C.-N. Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In: **Proceedings of the 2004 International Symposium on Physical Design**. New York, NY, USA: ACM, 2004. (ISPD '04), p. 26–33.

VISWANATHAN, N.; PAN, M.; CHU, C. Fastplace 2.0: An efficient analytical placer for mixed-mode designs. In: **Proceedings of the 2006 Asia and South Pacific Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 2006. (ASP-DAC '06), p. 195–200.

VISWANATHAN, N.; PAN, M.; CHU, C. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In: **Proceedings of the 2007 Asia and South Pacific Design Automation Conference**. Washington, DC, USA: IEEE Computer Society, 2007. (ASP-DAC '07), p. 135–140.

VYGEN, J. Algorithms for large-scale flat placement. In: **Proceedings of the 34th Annual Design Automation Conference**. New York, NY, USA: ACM, 1997. (DAC '97), p. 746–751.

WANG, L.-T.; CHANG, Y.-W.; CHENG, K.-T. T. (Ed.). **Electronic Design Automation: Synthesis, Verification, and Test**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.

XU, Y.; ZHANG, Y.; CHU, C. Fastroute 4.0: Global router with efficient via minimization. In: **Proceedings of the 2009 Asia and South Pacific Design Automation Conference**. Piscataway, NJ, USA: IEEE Press, 2009. (ASP-DAC '09), p. 576–581.

APÊNDICE A — TRABALHO DE GRADUAÇÃO I
Posicionador Global Analítico Quadrático de Células Lógicas

Henrique Plácido[1], **Orientador Prof. Dr. Ricardo Augusto da Luz Reis**[1]

[1]Instituto de Informtica – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{hplacido,reis}@inf.ufrgs.br

Abstract. *Placement is the stage of the physical synthesis of digital integrated circuits that determines the location of logic cells or other logic elements on the chip's surface. This is an important stage, because it affects the circuit performance, heat distribution, power consumption, and, also, the next stage of the chip development flow: routing. In this work, will be studied and implemented the global placement stages of the placement algorithm FastPlace: a placer belonging to the class of quadratic analytical algorithms.*

Resumo. *O posicionamento é a etapa da síntese física de circuitos integrados digitais que determina a localização de células lógicas ou outros elementos lógicos na superfície do chip. Essa etapa é importante, pois, ela afeta a performance do circuito, distribuição de calor, consumo de energia, e, também, a etapa seguinte do fluxo de desenvolvimento do chip: o roteamento. Nesse trabalho, serão estudados e implementados os estágios de posicionamento global do algoritmo de posicionamento FastPlace: um posicionador pertencente à classe dos algoritmos analíticos quadráticos.*

1. Introdução

O progresso na indústria de semicondutores, seguindo a lei de Moore, levou ao desenvolvimento de circuitos integrados que contêm milhões, até mesmo, bilhões de transistores, milhares de pinos e dezenas de camadas de interconexão [Kahng et al. 2011]. O processo de concepção desses circuitos é altamente complexo e depende fortemente de ferramentas de síntese automática [Kahng et al. 2011], também conhecidas como softwares de EDA (*Electronic Design Automation*). Com o uso desse recurso, em vez de os projetistas desenharem manualmente o leiaute do *chip*, é fornecida à ferramenta de EDA uma descrição do circuito feita com uma linguagem de descrição de hardware como, por exemplo, Verilog. Então, o software de síntese automática automatizará o desenvolvimento do *chip* dividindo esse processo em síntese lógica e síntese física.

A síntese lógica automatiza o processo de converter a descrição do circuito em uma estrutura com elementos de circuito de baixo nível disponíveis em uma biblioteca, que é uma coleção de células lógicas e outros elementos mais complexos, como, por exemplo, blocos de memória RAM. Assim, para uma dada descrição de entrada e uma biblioteca com uma determinada tecnologia fornecidas, a etapa de síntese lógica mapeia a descrição do circuito em uma estrutura chamada de *netlist*. Um *netlist* é uma lista de sinais (pinos ou terminais que estão conectados para estarem no mesmo potencial elétrico) e elementos de circuito que se conectam a esses sinais. Essa estrutura é utilizada como entrada para o estágio de síntese física [Kahng et al. 2011].

Durante a síntese física, todos os elementos presentes no *netlist* são instanciados com suas representações geométricas. Dessa forma, são atribuídas às células lógicas, transistores e outros elementos as suas localizações dentro do *chip* e, também, são roteadas as trilhas de metal que fazem a interconexão desses elementos [Kahng et al. 2011]. Além disso, a síntese física ainda executa a distribuição das redes de alimentação pelo *chip*, roteamento da rede de *clock* e a etapa de *Timing Closure*, que otimiza o circuito para que ele alcance uma determinada performance.

A etapa de encontrar um local para cada célula lógica e outros blocos mais complexos é conhecida como posicionamento e é o foco desse trabalho. Essa etapa é dividida em três subetapas: posicionamento global, legalização e posicionamento detalhado [Kahng et al. 2011].

A proposta desse trabalho é a implementação de um posicionador global. Para isso, será realizado o estudo e a codificação do posicionador FastPlace 1.0 [Viswanathan and Chu 2004], um algoritmo de posicionamento analítico quadrático.

O restante desse trabalho é organizado da seguinte forma: na seção 2 serão apresentados os fundamentos teóricos relacionados ao posicionamento global. Na seção 3 serão apresentados trabalhos relacionados, expondo uma visão geral de outros algoritmos estado-da-arte. Na seção 4 serão mostradas a proposta e a metodologia que serão utilizadas. Na seção 5, será apresentada a conclusão desse trabalho, e por fim, na seção 6, é exibido o cronograma de atividades que serão desenvolvidas ao longo do projeto.

2. Fundamentação Teórica

Nessa seção, serão apresentados os fundamentos teóricos sobre a etapa de posicionamento, enfatizando seus objetivos e suas subetapas. Na Subseção 2.1.2, será dado um enfoque para o posicionamento global, pois esse é o tema desse trabalho. Por fim, na Seção 2.1.3, será explicado o funcionamento do posicionador FastPlace, o algoritmo que foi adotado para o desenvolvimento desse projeto.

2.1. Posicionamento

Na etapa de posicionamento é determinada a localização e a orientação de cada elemento de circuito (células lógicas, blocos mais complexos) no leiaute do *chip* de forma que sejam respeitadas algumas restrições impostas para o projeto e, também, que as minimizações desejadas sejam atendidas, como, por exemplo, minimizar o comprimento total das interconexões entre as células [Kahng et al. 2011].

Alguns elementos podem ter localizações fixas enquanto que outros podem ser movidos (posicionáveis). A qualidade do posicionamento dos elementos que podem ser movidos influenciará fortemente a etapa de roteamento das interconexões [Kahng et al. 2011], que é a etapa seguinte no fluxo do desenvolvimento do *chip*. Isso significa que se a etapa de posicionamento não gerar um bom resultado, a etapa de roteamento poderá encontrar dificuldades na execução do roteamento, ou, até mesmo, não conseguir rotear todas as interconexões.

O processo de posicionamento é dividido nas etapas de posicionamento global, legalização e posicionamento detalhado. O posicionamento global frequentemente negligencia as formas e tamanhos dos elementos que podem ser posicionados e não se preocupa em alinhá-los em posições válidas no *chip*, ou seja, eles não serão posicionados

de forma que sejam respeitadas as bandas de posicionamento [Kahng et al. 2011], isto é, os espaços entre as linhas de alimentação. Sobreposições entre os elementos que estão sendo posicionados são permitidas, visto que a ênfase da etapa de posicionamento global está em espalhar os objetos pelo *chip*. A legalização é executada antes e depois da etapa de posicionamento detalhado e seu objetivo é alinhar os objetos posicionados na etapa anterior de forma que eles estejam posicionados nas bandas. Além disso, a legalização também remove as sobreposições entre as células, de modo que sempre é levado em consideração os impactos que essas alterações causarão no comprimento das interconexões e, consequentemente, no *delay* do circuito. O posicionamento detalhado altera de forma incremental a posição de cada célula lógica efetuando trocas locais, como, por exemplo, efetuando a troca de dois objetos ou deslocamentos para dar lugar a uma outra célula [Kahng et al. 2011].

2.1.1. Objetivos do Posicionamento

2.1.1.1 Comprimento Total dos Fios

A minimização do comprimento total de fios (interconexões entre as células lógicas) é o objetivo mais comum adotado nos posicionadores. Minimizar o comprimento total de fios indiretamente minimiza outros objetivos, como, por exemplo, os atrasos pelo circuito, pois um fio menor implica um atraso de propagação de sinais menor. Outros objetivos que também são indiretamente minimizados são o consumo de energia do *chip*, porque fios menores introduzem capacitâncias de carga menores e, também, o roteamento das trilhas de interconexão [Wang et al. 2009].

Não é fácil realizar durante a etapa de posicionamento uma predição do comprimento total de fios que o chip terá após a etapa de roteamento, pois essa informação depende muito do roteamento que será executado. Por essa razão, existem algumas abordagens que visam estimar durante a etapa de posicionamento o comprimento total dos fios. A mais utilizada é a HPWL (*half-perimeter wirelength*). A HPWL de uma rede (pinos que estão conectados) é igual à metade do perímetro do menor retângulo delimitador que inclui todos os pinos da rede. Esse método de estimação é muito utilizado porque ele pode ser computado em um tempo linear. Além disso, ele permite calcular o comprimento exato para redes que contêm dois ou três pinos [Wang et al. 2009].

Outra abordagem que é comumente utilizada para fazer uma predição do comprimento total de fios é a RMST (*rectilinear minimum spanning tree*). A RMST é uma árvore cujo comprimento total é igual ao comprimento mínimo dos fios calculado utilizando a distância de Manhattan. Essa técnica é exata para redes de dois pinos, porém ela pode superestimar o comprimento total para redes com mais pinos [Wang et al. 2009].

2.1.1.2 Roteamento

Um dos requisitos mais básicos de uma solução de posicionamento é que ela permita que a etapa seguinte do fluxo, o roteamento, seja capaz de executar o roteamento de todas as interconexões do circuito. Se a etapa de roteamento não conseguir executar o roteamento de todos os fios, a solução retornada pelo posicionador não é útil [Wang et al. 2009].

2.1.1.3 Performance

O posicionamento tem um grande impacto no atraso de cada interconexão e, consequentemente, na performance do *chip*. Visto que cada vez mais os componentes do *chip* se tornam menores devido à evolução nos processos de fabricação, os atrasos das interconexões tornam-se cada vez mais relevantes no desempenho do circuito. No entanto, o atraso de uma interconexão não depende somente do tamanho dela, mas sim também de outros fatores como, por exemplo, largura do fio, espaçamento entre fios, etc. Levar em consideração todas as variáveis que impactam no desempenho do circuito tem um custo computacional alto, por isso, na prática, os atrasos nos fios são controlados durante a etapa de posicionamento colocando restrições no tamanho total deles [Wang et al. 2009].

2.1.1.4 Consumo de Energia

A capacitância em uma rede é proporcional ao seu comprimento. Dessa forma, o problema de minimizar o consumo de energia pode ser formulado como um problema de minimização do tamanho total dos fios, visto que o consumo do circuito está relacionado com essas capacitâncias [Wang et al. 2009].

2.1.1.5 Distribuição de Calor

Uma distribuição não homogênea do calor pelo *chip* poderia afetar as características de circuitos que são sensíveis à temperatura e poderia, também, levar a problemas de confiabilidade. Assim, é desejável que os elementos que geram calor sejam bem distribuídos pelo *chip* de tal forma que não existam nele pontos com uma temperatura muito mais alta [Wang et al. 2009].

Executar um posicionamento levando em consideração esse critério não é um procedimento simples, pois, por exemplo, o calor gerado em cada elemento varia conforme a sua atividade [Wang et al. 2009].

2.1.2. Posicionamento Global

A etapa de posicionamento global é a mais importante das três, pois ela tem um alto impacto na qualidade da solução de posicionamento produzida [Wang et al. 2009]. O propósito dessa etapa é espalhar as células lógicas pela superfície do *chip* de tal forma que os objetivos, como por exemplo, comprimento total de fios, sejam minimizados sem violar as restrições impostas para o projeto. Nesse estágio de posicionamento, podem ocorrer sobreposições entre as células lógicas e, também, elas podem ser posicionadas fora das bandas (espaços entre as linhas de alimentação) permitidas [Kahng et al. 2011]. Além disso, o tamanho e a forma de cada célula não tem muita relevância, exceto os macro blocos, pois eles geralmente são mantidos em posições fixas e, também, as células lógicas não podem ser posicionadas em seus espaços [Chen et al. 2007][Monteiro 2014].

Algumas células lógicas podem ter posições pré-fixadas antes da etapa de posicionamento. Já as outras podem ser posicionadas em qualquer região da superfície do *chip*,

exceto nos locais onde os macro blocos já foram posicionados, visto que esses espaços são reservados. Dessa forma, o algoritmo de posicionamento global espalhará somente as células lógicas que podem ser movimentadas, não alterando a posição de cada célula préfixada. A posição encontrada pelo algoritmo para cada célula é determinada em função dos objetivos a serem atingidos [Monteiro 2014].

2.1.2.1 Algoritmos de Posicionamento Global

Os algoritmos de posicionamento global podem ser classificados em: analíticos, estocásticos e por particionamento [Spindler and Johannes 2007][Monteiro 2014].

Os algoritmos analíticos são baseados em uma função objetivo que é minimizada através de métodos matemáticos. Dependendo do tipo de objetivo a ser alcançado no posicionamento, os algoritmos analíticos podem ser quadráticos ou não lineares. Nos quadráticos, a função objetivo é quadrática e pode ser minimizada eficientemente através da resolução de um sistema de equações lineares derivado. Já nos algoritmos não lineares, a função objetivo não é linear e é minimizada utilizando métodos de otimização não lineares [Monteiro 2014].

Os posicionadores estocásticos são aqueles que utilizam movimentações aleatórias para minimizar a função custo. Esses posicionadores apresentam um problema de escalabilidade que é proporcional ao aumento no número de células lógicas. Como exemplo desse tipo de algoritmo, é possível citar o *Simulated Annealing* [Kahng et al. 2011].

No posicionamento por particionamento, o *netlist* e o leiaute são divididos recursivamente em *sub-netlists* e sub-regiões menores, de modo que o número de conexões entre as partições seja minimizado. Um exemplo de posicionador por particionamento é o *Min-Cut* [Kahng et al. 2011].

2.1.3. FastPlace 1.0

Nessa sessão, será apresentado o algoritmo de Posicionamento FastPlace 1.0. Esse algoritmo é baseado na abordagem de posicionamento quadrático, a qual formula o problema de minimização como uma função objetivo quadrática [Kahng et al. 2011].

O estudo desse algoritmo é fundamental para a execução desse trabalho, pois a proposta de solução é fortemente baseada no FastPlace.

2.1.3.1 Visão Geral do Algoritmo

O algoritmo FastPlace 1.0 [Viswanathan and Chu 2004] pode, basicamente, ser dividido em três estágios. O objetivo do primeiro é minimizar o comprimento total de fios e espalhar as células pela superfície do *chip* (nos locais permitidos) a fim de obter um posicionamento global inicial que será aperfeiçoado. Esse estágio é composto por um procedimento iterativo no qual são executadas a Otimização Global e o Deslocamento de Células. A otimização global é utilizada para minimizar a função objetivo quadrática (comprimento de fios). No deslocamento de células, a região onde as células estão sendo posicionadas é

dividida em retângulos de mesmo tamanho, de modo que cada um conterá uma determinada quantidade de células lógicas. Assim, as células são deslocadas sem ultrapassar os limites do retângulo. Por fim, forças de espalhamento são adicionadas em cada célula para prevenir que elas não se sobreponham.

No segundo estágio do FastPlace, o posicionamento global executado no primeiro estágio é refinado através da intercalação da técnica de Refinamento Local Iterativo com a Otimização Global e com o Deslocamento de Células. O Refinamento Local Iterativo é uma técnica que é utilizada para reduzir o comprimento de fios baseado na medida HPWL e, também, acelerar a convergência do algoritmo. No final desse estágio, as células lógicas estarão bem distribuídas pela superfície do chip com um bom valor de comprimento total de fios.

Finalmente, no terceiro estágio, é executado o Posicionamento Detalhado. Nessa etapa, o posicionamento executado anteriormente é legalizado de modo que cada célula seja posicionada em alguma banda válida. Além disso, todas as sobreposições entre as células são removidas.

As etapas de posicionamento global do algoritmo FastPlace estão resumidas abaixo na Figura 1.

Figura 1. Posicionamento Global do Algoritmo FastPlace

Etapas do Posicionamento Global do Algoritmo FastPlace

Estágio 1: Posicionamento GlobalInicial

1. **Repetir**
2. Executar a Otimização Global.
3. Executar o Deslocamento de Células e Adiconar Forças de Espalhamento.
4. **Até o posicionamento ser razoavelmente uniforme.**

Estágio 2: Posicionamento Global com Comprimento de Fios reduzido

1. **Repetir**
2. Executar a Otimização Global.
3. Executar o Refinamento Local Iterativo.
4. Executar o Deslocamento de Células e Adiconar Forças de Espalhamento.
5. **Até as células estarem uniformemente distribuídas.**

Fonte: adaptado de [Viswanathan and Chu 2004].

2.1.3.2 Otimização Global

O posicionamento quadrático utiliza molas para modelar a conectividade do circuito. A energia potencial total das molas, a qual é uma função quadrática do comprimento delas, é

minimizada para produzir uma solução de posicionamento [Viswanathan and Chu 2004].

Seja n o número de células que serão posicionadas e (x_i, y_i) as coordenadas do centro da célula i. O posicionamento das células na superfície do *chip* é dado pelos dois vetores de dimensão n $x = (x_1, x_2, x_3, \ldots, x_n)$ e $y = (y_1, y_2, y_3, \ldots, y_n)$. Seja W_{ij} o peso da rede de interconexão entre as células i e j. O custo dessa rede é dado abaixo na Equação 1.

$$\frac{1}{2}W_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{1}$$

Se a célula i está conectada a uma célula física f com coordenadas (x_f, y_f), o custo da rede é dado por:

$$\frac{1}{2}W_{if}[(x_i - x_f)^2 + (y_i - y_f)^2]. \tag{2}$$

Assim, a função objetivo que soma o custo de todas as redes pode ser escrita usando notação matricial, como é mostrado em seguida na Equação 3.

$$\Phi(x, y) = \frac{1}{2}x^T Q_x + d_x^T x + \frac{1}{2}y^T Q_y + d_y^T y + constante \tag{3}$$

Na função da Equação 3, Q é uma matriz $n \times n$ e d_x e d_y são vetores de dimensão n. Considerando apenas a direção x, a função objetivo é minimizada resolvendo o sistema de equações lineares da Equação 4.

$$Q_x + d_x = 0 \tag{4}$$

O sistema de equações lineares da Equação 4 fornece a solução para o problema de minimizar a função quadrática da Equação 3. Para esse problema, não são adicionadas restrições.

2.1.3.3 Deslocamento de Células

O resultado da Otimização Global é um posicionamento que minimiza a função objetivo quadrática. No entanto, essa etapa não leva em consideração a sobreposição entre as células. Dessa forma, o posicionamento resultante tem muitas células sobrepostas e, além disso, não é muito uniforme. A etapa de Deslocamento de Células tenta distribuir de forma mais uniforme as células pela superfície do circuito enquanto tenta manter as posições relativas obtidas na etapa de Otimização Global.

Inicialmente, a região onde as células estão sendo posicionadas é dividida em retângulos de mesmas dimensões. O interior de cada retângulo pode acomodar uma média de 4 células lógicas. Então, baseada no posicionamento feito na etapa de Otimização Global, a utilização de cada retângulo (U_i) é calculada. U_i é definido como a área total de todas as células presentes no interior do retângulo i. No cálculo de U_i, são somadas as áreas de todas as células que estão completamente cobertas pela área do retângulo i

e a área de sobreposição entre a célula e o retângulo para células que estão parcialmente sobrepostas com o retângulo. Então, as células são deslocadas em torno da região de posicionamento de acordo com posição atual de cada uma e, também, com base no retângulo onde elas estão inseridas.

O deslocamento de células é um processo de duas etapas. Primeiro, baseado na atual utilização de todos os retângulos de uma determinada fileira de retângulos, uma estrutura refletindo a utilização de cada retângulo é construída. Depois, cada célula pertencente a um determinado retângulo é linearmente mapeada para o retângulo correspondente da estrutura construída. Como resultado, células localizadas em retângulos com alta utilização serão deslocadas de modo que a utilização e a sobreposição entre as células sejam reduzidas.

2.1.3.4 Adição de Forças de espalhamento

Depois que as células foram deslocadas, forças adicionais precisam ser adicionadas a elas para que não retornem para as posições anteriores durante a próxima etapa de Otimização Global.

2.1.3.5 Refinamento Local Iterativo

Segundo [Viswanathan and Chu 2004], a função objetivo quadrática por si só não fornece o melhor resultado possível em termos de comprimento de fios. Dessa forma, é utilizado o Refinamento Local Iterativo para reduzir o comprimento de fios.

O procedimento de Refinamento Local Iterativo é intercalado com o Deslocamento de Células e com a Otimização Global durante o estágio 2 (Posicionamento Global com Comprimento de Fios Reduzido).

3. Trabalhos Relacionados

Muitas abordagens foram propostas para solucionar o problema de posicionamento de células lógicas. Dessa forma, nessa sessão, será apresentada a visão geral de alguns algoritmos estado-da-arte que fornecem resultados próximos, em termos de comprimento de fios, aos do FastPlace.

3.1. ePlace

O ePlace [Lu et al. 2014] é um algoritmo de posicionamento global analítico não-linear que pode ser utilizado em grandes projetos. Ele utiliza uma função densidade baseada em eletrostática para remover a sobreposição entre as células e o método de Nesterov para minimizar o custo não linear.

A abordagem do ePlace é modelar as portas lógicas como cargas elétricas. Dessa forma, a força elétrica é utilizada para espalhar as cargas (células lógicas) pela superfície até que o ponto de equilíbrio seja atingido. Quando as cargas atingem o ponto de equilíbrio, a menor taxa de sobreposição de células é atingida.

3.2. MAPLE

O Maple [Kim et al. 2012b] é um algoritmo de posicionamento multinível de circuitos de larga escala. O Maple trata restrições de utilização e os macro blocos são considerados elementos móveis durante o posicionamento.

A técnica utilizada para reduzir o comprimento de fios e a densidade de utilização da área de um *bin* (retângulo que contém os elementos sendo posicionados) é a *Progressive Local Refinement* (ProLR). Nessa técnica, é adotada uma iteração do *Iterative Local Refinement* (ILR) para modificar o posicionamento.

3.3. SimPL

O SimPL [Kim et al. 2012a] é um algoritmo de posicionamento quadrático. Nessa técnica, são mantidos dois posicionamentos: um superior e outro inferior que juntos convergem para uma solução final. O limite superior do posicionamento é obtido através de um algoritmo de legalização aproximada. Já o limite inferior é gerado ao minimizar o comprimento de fio nas iterações em que o sistema de equações lineares é resolvido.

4. Proposta e Metodologia

A proposta desse trabalho é a implementação de um posicionador global analítico quadrático de células lógicas que produza resultados, em termos de comprimento de fio, próximos aos dos algoritmos estado da arte mais poderosos. Além dos resultados próximos, é esperado que o posicionador seja mais rápido que os demais, de modo que o ganho em velocidade de processamento compense o resultado pior. Para tanto, serão utilizados os estágios 1 e 2 do algoritmo FastPlace 1.0 [Viswanathan and Chu 2004] descrito na Seção 2.1.3. O estágio 3 do FastPlace não será implementado, pois nele são realizadas as etapas de legalização e posicionamento detalhado, que não são o foco desse trabalho.

A abordagem adotada foi a de posicionamento analítico quadrático, pois, segundo [Viswanathan and Chu 2004], ela permite que o posicionamento em circuitos muito grandes possa ser feito de forma rápida e eficiente. Como citado anteriormente na Seção 2.1.1.3, o comprimento de fios impacta no desempenho do chip, portanto, ele será o objetivo que será minimizado.

A solução será implementada utilizando a linguagem de programação C++. Para facilitar o desenvolvimento desse projeto, será utilizado um *framework* desenvolvido pelo Grupo de Microeletrônica da UFRGS, o Uplace [Flach et al. 2015]. O UPlace é um ambiente utilizado para o desenvolvimento de algoritmos de posicionamento global, legalização e posicionamento detalhado [Monteiro 2014]. Esse framework implementa diversas funcionalidades, como, por exemplo, uma estrutura de dados para armazenar o circuito de entrada que será posicionado, interface gráfica, *parsers* para os formatos de descrição de circuitos DEF/LEF e Bookshelf, um resolvedor de sistemas de equações lineares, entre outras funcionalidades. Assim, o uso dessa infraestrutura agilizará o desenvolvimento do projeto, mantendo o foco no problema de posicionamento global em si.

Para testar a solução implementada, serão fornecidos como entradas arquivos de *benchmark* em formato Bookshelf. Os arquivos Bookshelf que serão utilizados são da conferência ISPD 2004 e contêm *benchmarks* baseados em circuitos reais da IBM.

Visto que o objetivo a ser minimizado é o comprimento de fio, o resultado do algoritmo será avaliado e comparado com outros algoritmos utilizando a métrica HPWL (*Half-Perimiter Wirelength*).

5. Conclusão

Nesse trabalho foi apresentada a proposta de implementação de um algoritmo de posicionamento global analítico quadrático. A solução apresentada é baseada no algoritmo FastPlace 1.0 [Viswanathan and Chu 2004], fazendo o uso de duas das suas três etapas, ou seja, descartando a etapa de posicionamento detalhado. Devido a abordagem adotada, é esperado que a solução que será implementada forneça bons resultados em termos da métrica HPWL.

Visto que esse trabalho trata do estudo e implementação de um algoritmo já proposto, espera-se que ele proporcione uma grande experiência na área de EDA, de modo que futuramente seja possível propor outras soluções que forneçam bons resultados.

6. Cronograma

Essa sessão apresenta as tarefas que serão desenvolvidas ao longo da execução do projeto. Inicialmente, será feito um estudo muito mais aprofundado sobre o FastPlace 1.0 [Viswanathan and Chu 2004]. Em seguida, será feito um estudo do *framework* UPlace [Flach et al. 2015] a fim de entender as suas funcionalidades e, dessa forma, permitir a integração dele no projeto. Então, no restante do projeto serão, basicamente, feitos a implementação do algoritmo e seus testes. Abaixo segue uma estimativa da distribuição das atividades ao longo dos próximos meses.

Tabela 1. Cronograma de atividades

	Atividade	Período de execução
1	Estudo aprofundado do FastPlace 1.0	Julho de 2015
2	Estudo do framework UPlace	Julho de 2015
3	Implementação e testes do algoritmo de posicionamento global	Agosto a novembro de 2015
4	Escrita do relatório do Trabalho de Graduação 2	Novembro a dezembro de 2015
5	Apresentação do Trabalho de Graduação 2	Dezembro de 2015

7. Referências

Referências

Chen, T.-C., Jiang, Z.-W., Hsu, T.-C., Chen, H.-C., and Chang, Y.-W. (2007). Ntuplace3: An analytical placer for large-scale mixed-size designs. In *Modern Circuit Placement*, pages 289–309. Springer.

Flach, G. et al. (2008–2015). Uplace.

Kahng, A., Lienig, J., Markov, I., and Hu, J. (2011). *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Netherlands, 1 edition.

Kim, M.-C., Lee, D.-J., and Markov, I. L. (2012a). Simpl: An effective placement algorithm. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(1):50–60.

Kim, M.-C., Viswanathan, N., Alpert, C. J., Markov, I. L., and Ramji, S. (2012b). Maple: Multilevel adaptive placement for mixed-size designs. In *Proceedings of the 2012 ACM International Symposium on International Symposium on Physical Design*, ISPD '12, pages 193–200, New York, NY, USA. ACM.

Lu, J., Chen, P., Chang, C.-C., Sha, L., Huang, D. J.-H., Teng, C.-C., and Cheng, C.-K. (2014). eplace: Electrostatics based placement using nesterov's method. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 121:1–121:6, New York, NY, USA. ACM.

Monteiro, J. L. (2014). Algoritmo de posicionamento analítico detalhado guiado a caminhos críticos. Master's thesis, UFRGS, Porto Alegre, Brasil.

Spindler, P. and Johannes, F. M. (2007). Kraftwerk: a fast and robust quadratic placer using an exact linear net model. In *Modern Circuit Placement*, pages 59–93. Springer.

Viswanathan, N. and Chu, C. C.-N. (2004). Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In *Proceedings of the 2004 International Symposium on Physical Design*, ISPD '04, pages 26–33, New York, NY, USA. ACM.

Wang, L.-T., Chang, Y.-W., and Cheng, K.-T. T., editors (2009). *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.