Uma revisão sobre o problema de posicionamento no projeto circuitos integrados modernos

M. P. Fogaça, C. Meinhardt, P. F. Butzen

***Resumo*—O posicionamento é uma das principais etapas da síntese física de circuitos integrados. Ela é responsável por encontrar uma posição para cada um dos componentes do circuito enquanto otimiza uma função custo que avalia a solução. Embora seja alvo de pesquisas por mais de 50 anos, o constante crescimento dos circuitos cria novos desafios. Por isto, muitas soluções algorítmicas surgiram ao longo do tempo. Atualmente as técnicas analíticas se destacam por serem rápidas e encontrar bons resultados. Por muito tempo o objetivo do posicionamento foi reduzir o comprimento dos fios. No entanto, as atuais tecnologias de fabricação de circuitos integrados, com elevado número de componentes e transistores nanométricos, aumentou a complexidade para as ferramentas de posicionamento e destacou a importância de lidar com outras métricas, como análise de tempos de propagação e roteabilidade do circuito. Dada a vasta literatura na área, este trabalho tem como objetivo apresentar uma revisão bibliográfica sobre o assunto. Além disto, apresentar algumas experiências com alguns algoritmos.**

I. Introdução

Circuitos integrados fazem parte dos dias atuais. O constante crescimento da complexidade funcional destes circuitos se reflete no aumento do número de componentes integrados dentro de um único *chip*. Projetos atuais podem contar com mais de 1 bilhão de transistores, iniciando a era dos projetos ULSI ($Ultra\text{-}Large\text{-}Scale\ Integration$).

Devido a este elevado número de componentes, os projetos ULSI aumentam a complexidade do projeto de circuitos integrados (CIs). Em contrapartida, o mercado continua exigindo que o tempo de desenvolvimento de um novo projeto seja o menor possível. Estes fatores levaram a necessidade de automação dos processos envolvidos na geração de circuitos integrados, implicando no desenvolvimento de ferramentas de CAD ($Computer\ Aided\ Design$) a fim de tornar o desenvolvimento de projetos mais rápido, confiável e barato. Neste contexto, a industria e academia realizam pesquisas por melhores algoritmos e estruturas de dados capazes de lidar com os desafios destes projetos.

Este artigo focará no desenvolvimento de ferramentas de CAD voltadas para o fluxo de síntese da metodologia $Standard\ Cell$. Nesta metodologia, na fase de síntese física, a etapa de posicionamento é a responsável por determinar a posição de cada componente do circuito na área do *chip*. O problema do posicionamento é pesquisado desde a década de 60. Diversas soluções algorítmicas para lidar com o problema surgiram ao longo do tempo e hoje podem ser classificadas em 3 grupos: abordagens estocásticas [1], por particionamento [2], [3] e analíticas [4]–[8]. As abordagens estocásticas adotam algoritmos probabilísticos para encontrar uma solução viável mesmo reduzindo o domínio de busca. Abordagens por particionamento utilizam heurísticas de particionamento de grafos para dividir o problema em instâncias menores e resolvê-las individualmente. Abordagens analíticas modelam o problema através de uma função custo e utilizam métodos matemáticos para otimizá-la.

O principal objetivo deste trabalho é fornecer uma completa revisão do problema de posicionamento no projeto de CIs, apresentando conceitos, definições e os principais algoritmos e metodologias empregados atualmente. Dentre estes, se destacam a meta-heurística $Simulated\ Annealing$ e o Posicionamento Analítico. Por isso, este trabalho também apresenta os resultados obtidos com duas análises realizadas nestes algoritmos: um estudo sobre o alternativas para a função $schedule$ de temperatura na meta-heurística $Simulated\ Annealing$, e uma avaliação de quatro diferentes métodos de adição de $Spreading\ Forces$ em algoritmos analíticos.

O restante do texto está organizado da seguinte forma: a Seção II apresentará o fluxo de projeto de um circuito integrado. A Seção III definirá formalmente o problema do posicionamento destacando os principais conceitos e classificações. A Seção IV irá abordar sobre a etapa do posicionamento global, detalhando os principais algoritmos empregados na literatura e apresentará algumas análises realizadas sobre eles. Por fim, a Seção V irá falar sobre legalização e posicionamento detalhado.

II. Fluxo de projeto de circuitos integrados

Atualmente, o projeto de circuitos integrados para aplicações especificas, conhecidos como ASICs ($Application\text{-}Specific\ Integrated\ Circuits$) pode ser realizado através de duas principais metodologias: o projeto $Full\ Custom$ ou o projeto $Standard\ Cell$. Projetos $Standard\ Cell$ permitem maior rapidez de projeto e confiabilidade, por adotar células padrão já devidamente projetadas e caracterizadas. Este trabalho focará no desenvolvimento de ferramentas de CAD voltadas para o fluxo de síntese da metodologia $Standard\ Cell$ [9].

O fluxo de projeto é um conjunto de procedimentos que permite aos projetistas percorrerem um caminho livre de erros desde a especificação do sistema até a sua implementação no silício [10]. A Fig. 1a apresenta um fluxo de projeto de

M. P. Fogaça é graduando em Engenharia de Computação pela Universidade Federal do Rio Grande, Rio Grande - RS, Brasil. (e-mail: mateusfogaca@furg.br).

C. Meinhardt é professora do Centro de Ciências Computacionais da Universidade Federal do Rio Grande (C3/FURG). (e-mail: cristinameinhardt@furg.br).

P. F. Butzen é professor do Centro de Ciências Computacionais da Universidade Federal do Rio Grande (C3/FURG). (e-mail: paulobutzen@furg.br).

circuitos integrados em linhas gerais. O fluxo de síntese da metodologia *Standard Cell* pode ser dividido em 4 grandes fases: modelagem em alto-nível, síntese RTL, síntese lógica e síntese física. A primeira fase do projeto é a modelagem em alto nível. Nesta fase, uma descrição comportamental do sistema é construída através de uma linguagem HDL (*Hardware Description Language*) como SystemC, VHDL, Verilog ou até mesmo C/C++. O objetivo é descrever o comportamento do sistema de forma que ele possa ser simulado e validado. Entretanto, essa descrição é de alto nível dificultando a síntese de um *hardware* diretamente a partir dela. Portanto, a descrição comportamental é traduzida para uma descrição RTL.

Na descrição RTL o comportamento do circuito é representado através do fluxo de sinais entre registradores e as operações lógicas realizadas nestes sinais. Na síntese lógica, a descrição RTL é inicialmente transformada em um circuito com portas lógicas genéricas básicas. Assim é possível realizar otimizações independentes de tecnologia através da reestruturação das portas lógicas no circuito. Após isso, o circuito passa pelo mapeamento tecnológico. Esta etapa é responsável por substituir as portas lógicas genéricas por portas lógicas presentes na biblioteca de células adotada no projeto. Depois de mapeado, o circuito passa pela otimização dependente da tecnologia. Ela reestrutura o circuito baseada em informações elétricas das portas lógicas, como atrasos e consumo de potência.

A última fase antes de enviar o projeto para manufatura é chamada de síntese física. Ela é responsável por converter um circuito no domínio estrutural para o domínio físico. A Fig. 1b mostra um típico fluxo de síntese física para projetos com metodologia baseada em células padrões *Standard Cell* adotado para a maioria dos projetos de ASICs.

A primeira etapa da síntese física é chamada de *floorplaning*. Com o crescimento da complexidade dos circuitos integrados, o uso de abordagens hierárquicas passou a ser amplamente adotado [11]. Estas abordagens dividem os componentes do circuito em grande módulos de acordo com a tarefa que exercem. O objetivo do *floorplaning* é realizar uma divisão inicial no *chip*, reservando uma região para cada um desses blocos. As duas etapas seguintes da síntese física, posicionamento e roteamento, acontecem individualmente em cada um desses módulos.

Uma das principais etapas da síntese física é chamada de posicionamento. O posicionamento é a etapa responsável por encontrar uma posição válida para cada uma das células na área disponível para o projeto em um *chip*, obedecendo ainda outras restrições de projeto. Esta etapa tem uma grande importância para o projeto, pois afeta diretamente o desempenho, consumo de potência e atrasos do circuito. Posicionadores geralmente recebem como entrada uma relação de todas as células e suas conexões, conhecida como *netlist*. Além disso, ferramentas de posicionamento também recebem uma descrição da biblioteca de células, que representa o conjunto de células disponíveis para a composição dos circuitos integrados.

Em projetos ULSI é cada vez mais comum incluir grandes módulos proprietários no desenvolvimento do projeto, tais como memórias ou dispositivos analógicos. Neste caso, os algoritmos de posicionamento devem ser capazes de lidar com

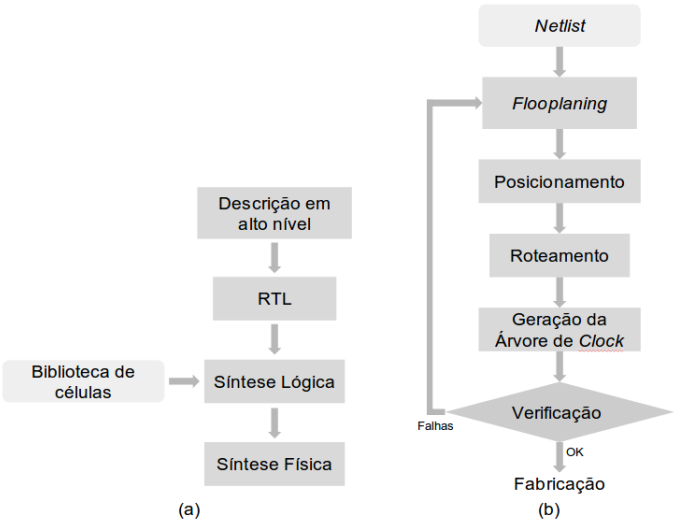

Figura 1. (a) Fluxo de projeto e (b) fluxo de síntese física.

estes módulos, conhecidos como obstáculos ou macro blocos, aumentando a complexidade da tarefa de posicionamento.

Outra etapa importante é o roteamento, responsável por determinar as rotas exatas pelas quais os fios que realizam a troca de sinais entre as células (interconexões) serão realizadas. Circuitos modernos podem conter milhões de conexões e as ferramentas de roteamento devem ser capazes de planejar todas elas. Para realizar esta tarefa, o roteamento é divido em dois passos: o roteamento global e o roteamento detalhado. O roteamento global divide a área do *chip* em canais adjacentes e cria as rotas entre estes canais. Já o posicionamento detalhado é responsável por determinar o caminho exato que os fios percorrerão.

Em circuitos síncronos, os dados são processados obedecendo um sinal de *clock*. Para garantir o funcionamento da lógica do circuito é necessário garantir que este sinal chegue em cada célula ao mesmo tempo. Pensando nisso, a etapa de geração da árvore de *clock* é responsável por distribuir este sinal garantindo que o atraso relativo entre cada célula seja o mínimo possível.

Depois que o circuito é posicionado, roteado e a árvore de *clock* é gerada, ele passa por ferramentas de teste e verificação. Uma delas é responsável pela extração de efeitos parasitas associados as conexões. Esses efeitos são devido as indutâncias, capacitâncias e resistências do circuito. Uma vez que essas informações são obtidas é possível simular a execução do *hardware*. Neste ponto é possível verificar com maior precisão se o circuito corresponde as especificações de projeto, verificando as características de atrasos, potência e área. Esta etapa é conhecida por ser um gargalo no projeto pois muitas vezes é necessário realizar novamente o posicionamento e o roteamento repetidamente até que se atinja as especificações de projeto desejadas.

Grande parte dos projetos de CIs tem como principal restrição os atrasos. Por isso, a maioria dos fluxos de síntese realiza uma análise dos tempos (atrasos) do circuito, conhe-

cida como Análise de $Timing$. A análise de $timing$ utiliza métodos para estimar os atrasos do circuito. Quanto mais informações sobre posicionamento, roteamento e efeitos parasitas forem considerada, mais precisa será a análise de $timing$. Por fim, são realizadas simulações para verificar ruídos, queda na tensão de alimentação e limites de eletromigração. Se o projeto atender as especificações, está pronto para fabricação.

III. Conceitos básicos sobre Posicionamento

O problema de encontrar uma posição válida para cada um dos componentes do circuito na área do $chip$ pode ser formalmente definido como segue.

O circuito é representado por um hipergrafo $N = (E, V)$, onde:

- o conjunto de arestas E representa os fios que conectam componentes do circuito;
- o conjunto de vértices V representa os componentes.

Objetivo: encontrar uma posição (X, Y) para cada componente, respeitando a restrição:

$$(\overrightarrow{X}, \overrightarrow{Y}) \in [X_{min}, X_{max}]^n \times [Y_{min}, Y_{max}]^n \quad (1)$$

O posicionamento é um problema de otimização e NP-completo [12]. Tradicionalmente, nesta etapa otimizar significa minimizar o valor de alguma variável do sistema. São exemplos de variáveis que se deseja minimizar: $wirelength$, atrasos, dissipação de potência e área. Por outro lado, algumas ferramentas modernas visam maximizar a roteabilidade do circuito deixando espaços vazios no $chip$ para facilitar a passagem de fios.

A. Conceitos Básicos

Algoritmos de posicionamento, em sua maioria, são modelados utilizando um conjunto de conceitos básicos. A Fig. 2a ilustra alguns destes conceitos que são descritos na sequencia.

- Célula: é uma porta lógica presente na biblioteca de células. Estas bibliotecas contêm todas as portas lógicas de um circuito, assim como suas características físicas (altura, largura, pinos de entrada e saída, tecnologia, dentre outros).
- Banda: um chip é divido em linhas de altura fixa. Estas linhas recebem o nome de bandas. As células estão limitadas a ocupar apenas o espaço de uma banda.
- Site: cada banda por sua vez é divida em fatias de largura fixa que recebem o nome de sites. O site é o menor espaço que uma célula pode ocupar. Enquanto portas lógicas mais simples ocupam 1 ou 2 sites, as mais complexas podem ocupar 30 ou mais.
- Net: é uma interconexão que liga duas ou mais células. Representa o fio que conectará os pinos de entrada ou saída de duas ou mais células, constituindo um circuito.
- $Half\text{-}Perimeter\ Wirelength$ (HPWL): é uma estimativa do comprimento das interconexões usadas para interligar as células utilizada para avaliar a qualidade do posicionamento. A Fig 2b mostra o calculo do HPWL para uma net com 3 células. Primeiramente encontra-se a mínima área retangular que englobe o centro de todas as $nets$. O HPWL será o meio perímetro dessa área.

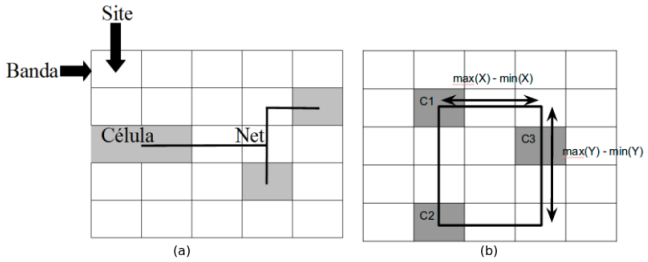

Figura 2. Representação gráfica de (a) banda, site, célula e net e (b) estimativa do comprimento das interconexões por HPWL.

B. Classificações

Os algoritmos de posicionamento podem ser classificados em dois grupos de acordo com seus parâmetros de entrada:

- Construtivos: produzem um posicionamento apenas a partir da $netlist$ do circuito. Posicionamentos quadráticos podem ser considerados como algoritmos construtivos.
- Iterativos: necessitam de um posicionamento inicial e realizam alterações iterativas em busca de uma solução melhor. Posicionadores que utilizam a meta-heurística $SimulatedAnnealing$ são exemplos de algoritmos iterativos.

Outra maneira de classificar os algoritmos é de acordo com a saída produzida. Os algoritmos que para uma determinada configuração de entrada sempre produzem o mesmo resultado são chamados determinísticos. O posicionamento quadrático é um exemplo. Os algoritmos que se baseiam em funções aleatórias e probabilidades e por isso podem produzir resultados diferentes para a mesma configuração de entrada são chamados de probabilísticos. É o caso dos algoritmos que adotam $Simulated\ Annealing$.

No entando, a maneira mais utilizada para classificar os algoritmos de posicionamento é de acordo com a sua natureza:

- Algoritmos de particionamento: utilizam técnicas de divisão e conquistas onde circuito é tratado como um hipergrafo e particionado recursivamente para facilitar a solução do problema.
- Algoritmos estocásticos: utilizam algoritmos probabilísticos de otimização. Aqui se destaca o $Simulated\ Annealing$.
- Algoritmos analíticos: representam o posicionamento através de uma função custo que se deseja otimizar.

C. Ferramentas de Posicionamento na Literatura

As primeiras ferramentas de posicionamento se caraterizavam por utilizar técnicas de divisão-e-conquista. Nelas, o circuito era tratado como um hipergrafo e particionado recursivamente para facilitar a solução do problema. A meta-heurística $Simulated\ Annealing$ (SA) [13] [14] foi aplicada com sucesso ao problema do posicionamento, dominando as primeiras ferramentas comercias e academicas de posicionamento. Entretanto, este algoritmo é conhecido por ser lento quando o número de componentes é elevado. Isto se tornou

um problema com o constante crescimento dos circuitos. Para lidar com isso, surgiram os posicionadores analíticos. As abordagens analíticas tratam o problema do posicionamento como um sistema de equações lineares originado de uma função custo que se deseja otimizar [14].

As ferramentas estado-da-arte não se baseiam apenas em um algoritmo mas em num fluxo tradicionalmente composto por três etapas: posicionamento global, legalização e posicionamento detalhado. No posicionamento global as células são espalhadas de forma uniforme na região do *chip*. Durante esta etapa, costuma-se gerar sobreposições entre os componentes. A etapa de legalização é responsável por retirar essas sobreposições alterando o posicionamento o mínimo possível. Por fim, o posicionamento detalhado divide o *chip* em regiões pequenas e tenta melhorar a solução localmente. Além disso, inicialmente as ferramentas de posicionamento minimizavam somente o comprimento total dos fios ($wirelength$) utilizados para realizar as interconexões entre as células. São os chamados $wirelength$-$driven\ placement$. De fato, ainda hoje a maioria dos algoritmos focam nessa métrica de avaliação. Recentemente outras métricas ganharam destaque, com o objetivo de melhorar a qualidade das soluções. Minimizar o comprimento dos fios pode gerar áreas muito congestionadas, dificultando a etapa de roteamento e afetando sua qualidade. Pensando nisso, os $routability$-$driven\ placement$ [15]–[17] reservam espaços vazios em suas soluções para a passagem de fios. Por muito tempo, considerou-se que minimizar o comprimento dos fios indiretamente melhorava o circuito com relação a tempos de propagação. No entanto, isto nem sempre é verdade. Por isto, as ferramentas de $timing$-$driven$-$placement$ [18]–[20] tentam reduzir os caminhos críticos do circuitos ou dar prioridade para reduzir conexões muito grandes.

IV. Posicionamento Global

O objetivo do posicionamento global é realizar uma distribuição das células pela área do *chip* enquanto otimiza a função custo. A etapa do posicionamento global tem grande impacto no resultado final em termos de qualidade e tempo de execução. Por este motivo é alvo da maioria das pesquisas em posicionamento [11].

A. Simulated Annealing

O $Simulated\ Annealing$ é inspirado no processo metalúrgico de fabricação dos metais. Neste processo a temperatura do metal é elevada até um valor no qual as moléculas estão tão agitadas que não obedecem a nenhuma organização. A próxima etapa consiste em resfriar o sistema lentamente. Isto permite que os átomos se arranjem de forma uniforme, diminuindo os defeitos do metal. Quando a temperatura assume valores baixos os átomos formam uma rede cristalina.

O Algoritmo 1 [12] apresenta uma implementação do $Simulated\ Annealing$ aplicado ao problema do posicionamento. É necessário conhecer três parâmetros: posicionamento inicial, temperatura inicial e número de perturbações. O algoritmo consiste basicamente em dois laços de repetição: o laço externo controlado pela variável $temperatura$ (linhas 4-13) e o laço interno controlado pela variável *número_de_perturbações* (linhas 5-11). Em cada iteração do laço interno a solução é perturbada (linha 6). Perturbar significa gerar um novo posicionamento através da troca de uma ou duas células de lugar. O próximo passo é avaliar essa troca através de uma função custo. As trocas que diminuem o custo sempre são aceitas enquanto as que aumentam estão sujeitas a expressão $e^{\frac{\Delta custo}{temperatura}}$ (linhas 8-14). Logo, a variável $temperatura$ é responsável pela aceitação das trocas. Tradicionalmente inicializa-se temperatura com um valor alto visando a aceitação de muitas trocas "ruins". Este valor é reduzido ao longo da execução do algoritmo de acordo a função $schedule$ (linha 12). Conforme a temperatura é reduzida, o algoritmo adquire um comportamento guloso. Desta maneira, o $Simulated\ Annealing$ consegue obter bons resultados, evitando mínimos locais.

Algoritmo 1: Pseudocódigo do $Simulated\ Annealing$.

```
   Entrada: posicionamento_inicial, tempertatura_inicial,
   número_de_perturbações
   Saída: pos
1  begin
2      temperatura ← temperatura_inicial;
3      pos ← posicionamento_inicial;
4      repeat
5          for i ← 0 to número_de_perturbações do
6              novo_pos ← perturbação(pos);
7              Δcusto = custo(novo_pos) - custo(pos);
8              if Δcusto < 0 then
9                  pos = novo_pos;
10             else if random(0,1) > e^(Δcusto/temperatura) then
11                 pos = novo_pos;
12         temperatura ← schedule(temperatura);
13     until temperatura = 0;
```

As função de perturbação (linha 6), função de custo (linha 7) e $schedule$ de temperatura (linha 12) são três funções importantes na implementação do $Simulated\ Annealing$ para posicionamento e costumam ser explorados para a obtenção de melhores resultados. Características destas três funções são detalhadas a seguir.

1) Função de perturbação: a função de perturbação tem por objetivo alterar um posicionamento prévio. Estas alterações podem ser realizadas através de trocas aleatórias de posição. Entretanto, alguns trabalhos exploram trocas gulosas. Trocas gulosas mudam as células de posição visando uma melhora no posicionamento. Um estudo comparando as técnicas de trocas aleatórias e trocas gulosas quanto a redução do comprimento dos fios é apresentado em [21], explorando dois tipos de trocas aleatórias: a troca simples e a troca dupla. Na troca simples, sorteia-se uma célula e uma posição ainda não ocupada. Desta forma, a célula passa a ocupar uma nova posição. Já na troca dupla, sorteiam-se duas células e uma é colocada na posição da outra. A Fig. 3. exemplifica estes dois tipos de trocas aleatórias.

2) Função custo: a função custo calcula um valor numérico a fim de avaliar um posicionamento. As abordagens tradicio-

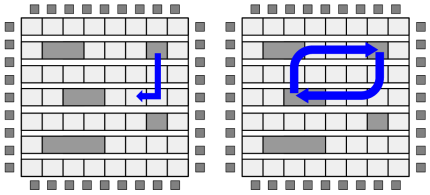

Figura 3. Exemplo de troca simples (esquerda) e troca dupla (direita).

nais estão relacionadas com a estimativa do comprimento total dos fios nas interconexões entre as células, atrasos, dissipação de potência e congestionamento. A métrica mais utilizada é a estimativa do comprimento dos fios. Neste trabalho, o valor da função custo é dada pelo HPWL.

3) Schedule de temperatura: o *schedule* de temperatura pode ser definido como uma função matemática decrescente que determina o comportamento da temperatura conforme a execução do algoritmo. Como pode ser observado em [22], escolher um bom *schedule* de temperatura é de extrema importância para o SA obter bons resultados. Neste trabalho, duas abordagens para função de temperatura serão estudadas e são ilustradas pela Fig. 4:

1) *Schedule* Linear. É a curva de resfriamento mais simples que pode ser adotada. Partindo de uma temperatura inicial, a cada iteração é subtraído um coeficiente de resfriamento fixo do valor atual.
2) *Schedule* de terceiro grau. Implementa uma curva com um ponto de inflexão. Antes do ponto, a segunda derivada da curva é negativa e após é positiva. Por ser mais complexa, permite maior flexibidade. É possível escolher onde se deseja colocar o ponto de inflexão. Através dele é possível determinar se o SA executará mais tempo com o comportamento de um algoritmo guloso ou aceitando trocas "ruins".

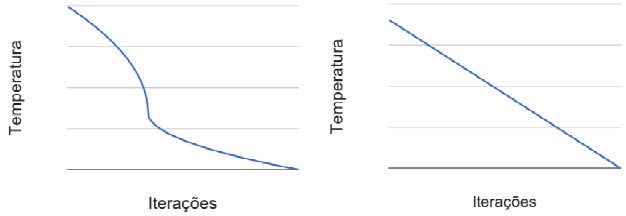

Figura 4. *Schedules* linear (esquerda) e de terceiro grau (direita).

Ambos *schedules* foram aplicados a 4 *benchmarks* [23] [24]. A Tabela I mostra características individuais de cada circuito.

Tabela I. BENCHMARKS IBM

Circuito	# Células	# *Pads*	# *Nets*	# linhas	# *sites* por linha
IBM01	12506	246	14111	96	1550
IBM02	19342	259	19584	109	1757
IBM03	22853	283	27401	121	1938
IBM04	27220	287	31970	136	2198

Inicialmente foi calculada a temperatura inicial para cada circuito. A Equação 2 mostra o cálculo da temperatura inicial para o SA [12], onde $temp\_inicial$ é a $temperatura\ inicial$ do SA, Δp_medio é a variação média da função custo no circuito e $prob\_inicial$ é a probabilidade inicial do algoritmo aceitar uma troca "ruim". Calcula-se a $prob\_inicial$ realizando perturbações aleatórias. Para estes resultados, foi utilizado $prob\_inicial$ = 0,4.

$$temp\_inicial = -\Delta p\_medio/log(prob\_inicial) \quad (2)$$

Ambos os *schedules* de temperatura foram configuradas para que a temperatura atinja o valor 0 na milésima iteração. Os experimentos realizados neste trabalho consideram que o ponto de inflexão do *schedule* de terceiro grau acontece na iteração 350 e assume um valor igual a 35% da temperatura inicial [25].

As células foram distribuídas de forma aleatória na área do circuito, antes da execução do SA. A Fig. 5 mostra a redução do $wirelength$ ao longo da execução do algoritmo para o circuito IBM01. Em um primeiro momento, ambas as curvas tem uma queda brusca. Isto é atribuído ao posicionamento inicial aleatório, pois inicialmente é muito fácil realizar trocas boas. Entre 50 e 200 iterações, o SA está aceitado muitas trocas ruins e o $wirelength$ eventualmente assume um valor maior que o da iteração anterior, causando oscilações na curva. Entre 200 e 400 iterações o comportamento muda. Enquanto na curva de $wirelength$ do *schedule* linear apenas se observa um redução no número de oscilações, na curva do *schedule* de terceiro grau verifica-se uma queda acentuada. Isto ocorre porque a temperatura também teve este mesmo comportamento, ou seja, a temperatura é reduzida bruscamente e o SA passa a assumir um comportamento guloso. A partir de 600 iterações ambas as curvas passam a convergir para seu resultado. O tempo de execução é o mesmo para os dois *schedules*, sendo função apenas do número de componentes que compõem o circuito. O comportamento foi o mesmo para todos os *benchmarks*.

A Tabela II apresenta os resultados individuais de cada *schedule* e a melhora apresentada pela abordagem de terceiro grau. Nota-se que embora a diferença decresça de acordo com o tamanho do circuito o *schedule* de terceiro grau obteve os melhores resultados.

Tabela II. TABELA COMPARATIVA COM *wirelengths* FINAIS PARA AS DUAS ABORDAGENS DE *schedule* DE TEMPERATURAS.

Circuito	Linear ($\times 10^6$)	3º grau ($\times 10^6$)	Melhora
IBM01	5,0	4,2	16,0%
IBM02	8,1	6,8	16,0%
IBM03	12,1	10,8	10,7%
IBM04	14,9	13,6	8,7%

Concluiu-se que ao adotar o *schedule* de terceiro grau obtem-se melhores resultados. Isto acontece porque através do ponto de inflexão da curva pode-se ajustar o algoritmo para executar mais tempo de forma gulosa mesmo sem perder a característica do SA de escapar de mínimos locais.

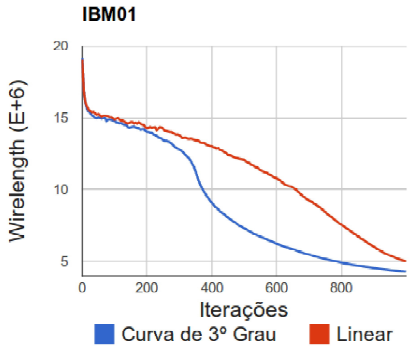

Figura 5. Curvas de $wirelength$ para as duas abordagens no $Benchmark$ IBM01

B. Posicionamento Analítico

O posicionamento analítico formula o custo do posicionamento como uma função matemática das posições das células do circuito e tenta otimizá-la através de técnicas analíticas. Dependendo da função modelada as técnicas são classificadas como quadráticas ou não-quadráticas. Técnicas quadráticas utilizam funções convexas que podem ser otimizadas resolvendo um conjunto de equações lineares. Em contrapartida, as não-quadráticas utilizam funções não lineares, por exemplo o modelo $log\text{-}sum\text{-}exp$ de $wirelength$ e a função de densidade $bell\text{-}shaped$ [26]. Elas obtém resultados tão bons quanto as quadráticas, porém são difíceis de implementar e pouco eficientes computacionalmente.

1) Modelo de $wirelength$ quadrático: o HPWL é uma função convexa que não pode ser derivada, impedindo seu uso no posicionamento analítico. A solução é utilizar outros modelos que captem o objetivo de redução dos fios. A técnica mais consolidada na literatura, o modelo de $wirelength$ quadrático. O wirelength quadrático de um fio que liga dois componentes é dado por:

$$L_{i,j} = (x_i - x_j)^2 + (y_i - y_j)^2 \tag{3}$$

Na prática, costuma-se adicionar o fator $\frac{1}{2}$ a expressão para facilitar a derivação. Logo o $wirelength$ quadrático total do circuito é dado por:

$$L_{i,j} = \frac{1}{2} \times \sum_{1 \leq i < j \leq n} c_{\{i,j\}} \times (x_i - x_j)^2 + (y_i - y_j)^2 \tag{4}$$

Onde $c_{\{i,j\}}$ é um peso associado a cada conexão. Tradicionalmente o valor de $c_{\{i,j\}}$ para um fio que liga dois componentes é 1. Caso não haja conexão entre eles $c_{\{i,j\}}$ assume o valor 0. Este modelo é implementado em sua forma matricial, dada por:

$$[L = \frac{1}{2} x^T Q x + d_x^T x + \frac{1}{2} y^T Q y + d_y^T y] \tag{5}$$

Nesta notação, x e y são os vetores que armazenam as posições das células, ou seja, são variáveis. Q é uma matriz obtida a partir da fórmula $Q = D - C$, onde D é uma matriz diagonal tal que $D_{ii} = \sum_{j \epsilon V} c_{\{ij\}}$ e C é a matriz de conectividade do hipergrafo do circuito. Por fim, d_x e d_y são vetores que armazenam as conexões com componentes fixos na forma $d_{x_i} = -\sum_{j \epsilon V_{fixos}} c_{ij} x_j$.

Para encontrar os valores de x e y que gerem o $wirelength$ mínimo, L é derivado parcialmente com relação a x e y e o resultado é igualado a 0, gerando um conjunto de equações lineares para o eixo x e outro para o eixo y:

$$\frac{\partial L}{\partial x} = Qx + dx = 0 \tag{6}$$

$$\frac{\partial L}{\partial y} = Qy + dy = 0 \tag{7}$$

2) Conexões entre múltiplos pinos: a abordagem apresentada só trata conexões entre 2 elementos. Na prática, os fios que realizam as conexões do circuito podem conectar múltiplos pinos, como ilustrado na Fig. 6a. Para se adequar ao modelo estas conexões precisam ser decompostas. Existem diversas metodologias na literatura para realizar esta tarefa.

A Fig. 6b mostra o modelo clique. Neste modelo a conexão original é substituída por um grafo completo, ou seja, todos são conectados com todos. Para manter o sistema balanceado, o peso das conexões no modelo clique é $\frac{c}{k-1}$ [6], onde c representa o peso da conexão original e k é o número de pinos. Uma outra alternativa é o modelo estrela, apresentado na Fig.6(c). Neste modelo todos os pinos são conectados a um nodo adicional, chamado de estrela [6]. O nodo estrela é adicionado ao sistema como se fosse uma célula do circuito.

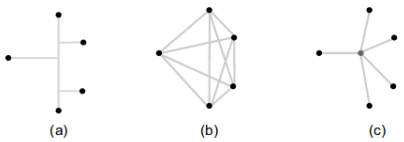

Figura 6. Representação para uma conexão com multiplos pinos (a) através do modelo clique (b) e modelo estrela (c).

Viswanathan provou que os dois modelos são equivalentes desde que os pesos das conexões tenham sido atribuídos corretamente [5]. Partindo deste pensamento, ele propôs um modelo híbrido [5]. O modelo adota cliques para conexões com 3 pinos ou menos e estrela para as demais. Devido ao uso de nodos estrela, o número de conexões para $benchmarks$ industriais diminui em torno de 10 vezes em comparação ao modelo clique puro. Além disso, o uso do modelo clique para conexões com 2 ou 3 pinos evita que muitas variáveis adicionais sejam criadas pela inserção de nodos estrela. Estes dois fatores contribuem para otimizar a execução do posicionamento quadrático.

Recentemente o modelo $Bound2Bound$ foi proposto e obteve um grande sucesso [7]. Ele se destaca dentre os demais por representar precisamente o HPWL através do $wirelength$ quadrático. Neste modelo as conexões resultantes são realizadas apenas com os nodos extremos. Todas as conexões possuem peso $w = \frac{2}{(p-1)l}$, onde l é a distância entre os pinos. O $Bound2Bound$ possui muitas vantagens: permite que as ferramentas utilizem o HPWL como métrica de otimização;

Gera um número de conexões menor que os demais, exceto para conexões com 4 pinos ou mais; Não introduz variáveis adicionais.

3) $SpreadingForces$: o posicionamento quadrático pode ser fisicamente interpretado como um sistema de molas onde os componentes do circuito são pontos adimensionais, as conexões entre eles são molas e os *pads* são pontos fixos.

A Fig. 7 mostra o resultado do posicionamento quadrático para um *benchmark* industrial com 12506 células e 246 *pads*. Devido ao fato desta formulação não considerar as dimensões dos componentes, o resultado geralmente possui muitas sobreposições principalmente na região central do *chip*. Por consequência o resultado do posicionamento quadrático não é uma solução viável. Uma alternativa para lidar com este problema é a adição de forças extras ao sistema chamadas de *Spreading* ou *Diffusion Forces*. Estas forças tem como objetivo iterativamente deslocar as células de regiões mais densas para regiões menos densas. Neste caso, não se resolve apenas um sistema de equações lineares, mas quantos forem necessários até as células ficarem uniformemente distribuídas.

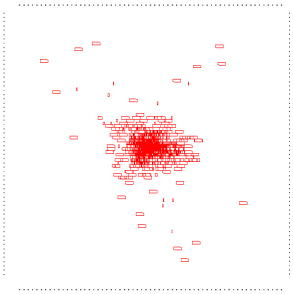

Figura 7. Resultado do posicionamento quadrático.

4) Avaliação de metodologias de aplicação de $Spreading\ Forces$: este trabalho apresenta uma avaliação de quatro modelos de *spreading forces*. Em todos, as forças foram implementadas como pinos virtuais ligados as células. A Fig. 8a apresenta os dois primeiros modelos avaliados. Primeiramente, quatro pinos virtuais são colocados nos cantos do *chip*. A seguir, as células são dividas em quatro grupos, cada um deles conectado a um pino. A diferença entre os modelos se da na forma de dividir os grupos. O modelo 1, representado por linhas contínuas, divide o circuito no centro físico do *chip*. O modelo 2, representado por linhas tracejadas, divide o circuito através do centro de massa do posicionamento.

Os modelos 3 e 4 são ilustrados na Fig. 8b. Nestes modelos, a posição dos pinos virtuais varia para cada célula. A posição do pino virtual é calculada com base em um ponto de origem e a posição da célula. No modelo 3 (linhas contínuas) o ponto de origem é o centro físico do *chip* enquanto que no modelo 4 (linhas tracejadas) é o centro de massa do posicionamento. A posição do pino virtual é determinada através de uma reta que parte do ponto de origem e se prolonga até a borda do *chip*. O pino é colocado no ponto de intersecção da reta com a borda.

Dois fluxos de execução foram aplicados nos quatro primeiros *benchmarks* da Tabela I. A Fig. 9 mostra os passos

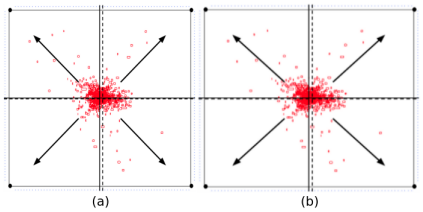

Figura 8. (a) Modelo 1 (linhas contínuas) e modelo 2 (linhas tracejadas) e (b) modelo 3 (linhas contínuas) e modelo 4 (linhas tracejadas)

executados neste trabalho. A primeira etapa, chamada de inicialização, gera uma solução inicial através do posicionamento quadrático. Este posicionamento serve como entrada para os dois fluxos de execução. No fluxo 1 a solução é legalizada gerando o *Posicionamento Final A*. No fluxo 2, as *spreading forces* são adicionadas e o posicionamento quadrático é refeito. Por fim, o posicionamento intermediário gerado nesta etapa é legalizado gerando o *Posicionamento Final B*. Em ambos os fluxos, a etapa de legalização foi realizada com a ferramenta PlaceUtil [27].

Um ponto importante é saber definir o valor ideal para as *spreading forces*. O uso do valor tradicional para conexões entre dois pinos, que é 1, pode causar muitas sobreposições nas bordas do circuito. Para avaliar encontrar o valor ideal cada modelo foi aplicado 10 vezes variando a intensidade de 0,1 até 1.

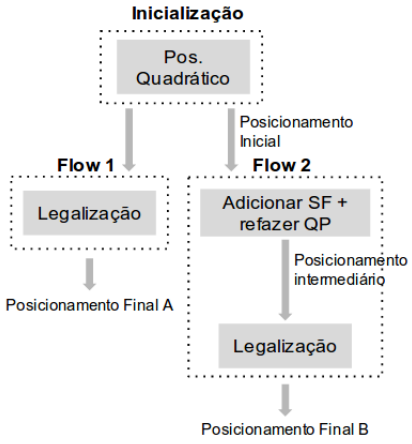

Figura 9. Etapa de inicialização seguida por dois fluxos de execução. No primeiro a solução é legalizada. O segundo explora o uso de *spreading forces* para melhorar a solução inicial.

Os resultados foram divididos em duas análises: com relação ao *Wirelength* final e tempo de execução. Todos os experimentos foram implementados utilizando a linguagem C++ e executados em um Intel Core i7-2670QM 2.20GHz com 6GB de memória.

A Tabela III mostra uma comparação entre os dois fluxos de execução em termos de *wirelengh*. O fluxo 2 foi executado com todos os modelos de *spreading forces* apresentados. Os resultados mostram que o fluxo 2 gera resultados em média

50% melhores. No entanto nenhum modelo se destacou dentre os demais.

A outra análise compara os tempos de execução. O resultados mostram que o tempo de execução do posicionamento quadrático dobra com o uso de *spreading forces*. A única exceção é o IBM02, no qual o tempo triplica. No entanto, o tempo de legalização reduz em 80%. A Tabela V compara o tempo de execução total para cada fluxo. É possível notar que o tempo de execução total é determinado pelo posicionamento quadrático, ou seja, o ganho no tempo de legalização é desprezível.

Tabela III. HPWL FINAL PARA CADA FLUXO

Circuito	Fluxo 1 - HPWL (E+06)	Fluxo 2 - HPWL (E+06)			
		Modelo 1	Modelo 2	Modelo 3	Modelo 4
IBM01	12,0	5,1	5,3	6,0	5,9
IBM02	23,0	12,6	12,5	16,8	16,0
IBM03	31,0	14,5	14,0	14,4	12,8
IBM04	41,3	26,2	17,8	24,2	18,3

Tabela IV. TEMPO DE EXECUÇÃO DE CADA PASSO

Circuito	Pos. Quadrático Inicialização (s)	Adicionar SF e refazer PQ (s)	Legalização Fluxo 1 (s)	Legalização Fluxo 2 (s)
IBM01	427,19	393,72	20,07	2,94
IBM02	1263,94	2716,02	43,33	5,79
IBM03	2069,17	2205,46	53,28	2,60
IBM04	3172,80	4735,24	64,71	8,24

Tabela V. TEMPO DE EXECUÇÃO TOTAL PARA CADA FLUXO

Circuito	Fluxo 1 (s)	Fluxo 2 (s)
IBM01	447,26	823,85
IBM02	1307,27	3985,75
IBM03	2122,45	4277,23
IBM04	3237,51	7916,28

Por fim, a Fig. 10 apresenta o tempo de execução do posicionamento quadrático como uma função da intensidade das *spreading forces*. É possível concluir que o tempo de execução decai exponencialmente conforme a intensidade cresce.

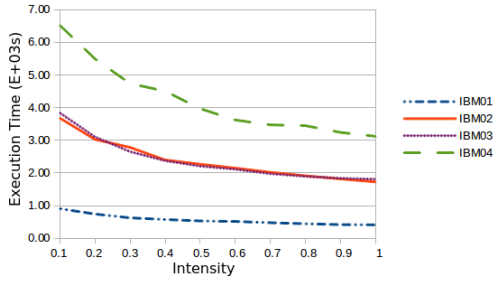

Figura 10. Tempo de execução como uma função da intensidade das *spreading forces*.

C. Particionamento

O posicionamento por particionamento pode ser definido da seguinte maneira: dado um circuito e uma área de *chip*, cada célula é atribuída a um região específica no espaço disponível para posicionamento. A Fig. 11 ilustra o funcionamento de uma ferramenta de posicionamento por particionamento. Primeiramente, o circuito é divido em dois subcircuitos. Da mesma maneira, a área física do *chip* é separada em 2 partes, chamadas de regiões, uma para cada subcircuito (Fig. 11b). A divisão é realizada pensando que cada região deve possuir um espaço disponível maior que a área total das células do seu subcircuito. Esta operação é repetida recursivamente até que a quantidade de células em cada região seja suficientemente pequena (Fig. 11c). A tarefa de determinar uma posição específica para cada célula dentro de uma região é feita pelas etapas de legalização e posicionamento detalhado, descritas na seção V.

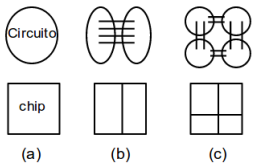

Figura 11. Ilustração do funcionamento de uma ferramenta de particionamento.

A divisão do circuito em subcircuitos é tratada como um problema de particionamento de hipergrafos. Neste caso, as células são tratadas como vértices e as conexões como arestas. O problema do particionamento de hipergrafos consiste em agrupar os vértices em n grupos minimizando o número de conexões entre eles. O *Fiduccia-Mattheyses* [28] é uma heurística clássica para particionamento.

O Algoritmo 2 [11] apresenta o pseudocódigo do *Fiduccia-Mattheyses* para o particionamento em 2 grupos. O algoritmo recebe como entrada um particionamento inicial, geralmente aleatório. Os passos 3-10 são chamados de "passe". Dentro de cada passe, o algoritmo computa o vértice de maior ganho (linha 5).

O ganho de um vértice é dado pelo número de arestas que cruzam os grupos menos o número de arestas que cruzarão os grupos caso o vértice troque de grupo. A Fig. 12a ilustra um grafo particionado em 2 grupos. O número de conexões que cruzam os grupos é igual a 3. Caso o vértice e mude de grupo, o número de conexões que cruzam os grupos será 2 (Fig. 12b). Neste caso, diz-se que o ganho de $g(v_e) = 3 - 2 = 1$.

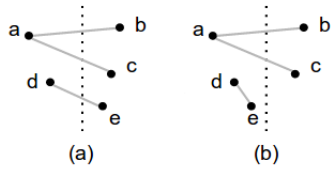

Figura 12. Grafo particionado em 2 grupos. O ganho do vértice e é igual a 1 pois caso troque de grupo, o número de conexões que cruzam os grupos cai de 3 para 2.

Uma vez que o vértice de maior ganho tenha sido encon-

trado, ele troca de grupo e é travado, ou seja, não é considerado até o próximo "passe". O ganho desta troca é armazenado (linhas 6-8). Esta operação é repetida n vezes, onde n é o número de vértices. No fim de cada passe, é encontrada a sequencia de k trocas que maximizem o custo total (linha 9). Os k movimentos são mantidos e os demais desfeitos (linha 10).

Algoritmo 2: Pseudocódigo do $Fiduccia\text{-}Mattheyses$.

```
   Entrada: Particionamento inicial
1  begin
2      repeat
3          Destravar todos os vértices;
4          for i ← 0 to |V| do
5              Selecionar o vértice v_i ainda não travado com
               o ganho máximo;
6              g_i ← g(v_i);
7              Mover o vértice v_i para o outro grupo;
8              Travar o vértice v_i;
9          Encontrar k tal que G = Σ_{i=i}^{k} g_i
10         Tornar os movimentos v_1, ..., v_k permanentes e
           desfazer os movimentos v_{k+1}, ..., v_n;
11     until G = 0;
```

O $Fiduccia\text{-}Mattheyses$ se torna ineficiente para hipergrafos muito grandes. Uma solução é utilizar abordagens hierárquicas. Estas abordagens são compostas por três etapas: etapa de engrossamento, particionamento inicial e etapa de refinamento. Na etapa de brutalização, hipergrafos menores são gerados através do agrupamento de vértices com muitas ligações entre si. Na etapa de particionamento inicial, uma heurística é utilizada para criar grupos iniciais com base no hipergrafo agrupado. Por fim, na etapa de refinamento os agrupamentos são desfeitos e a heurísticas de particionamento são utilizadas para melhorar a solução. O $hMetis$ [29] é um algoritmo rápido e que gera bons resultados utilizando a estratégia descrita.

V. Legalização e posicionamento detalhado

A etapa da legalização recebe como entrada o posicionamento global e tem como objetivo retirar todas as sobreposições alterando solução inicial o mínimo possível. O Tetris [30] é um algoritmo clássico de legalização. Primeiramente ele ordena as células de acordo com sua coordenada x em ordem crescente. A seguir, as células são posicionadas individualmente respeitando essa ordenação. Para cada célula, o algoritmo percorre todos os $sites$ de todas a bandas procurando a posição livre no qual o deslocamento da célula é mínimo. O Tetris é um algoritmo heurístico guloso e por isso seu resultado impacta consideravelmente o posicionamento global.

Pensando nisso, o Abacus [31] pode ser visto como uma evolução do Tetris. De forma semelhante, ele ordena todas as células de acordo com a coordenada x. A seguir, busca a posição de deslocamento mínimo da célula em cada banda utilizando programação dinâmica. Neste caso, ele modifica a posição de todas as células já colocadas naquela banda, respeitando apenas a ordenação. O Abacus reduz em média 30% o movimento das células e aumenta em apenas 7% o tempo de execução. Após a legalizada, a solução passa para o posicionamento detalhado.

O posicionamento detalhado é a última etapa do fluxo. Esta etapa divide o $chip$ em regiões pequenas visando melhorar a solução através de trocas locais. Estas trocas podem ter vários objetivos, por exemplo:

- Diminuir o impacto da legalização, em termos de $wirelength$;
- Compensar por imprecisões da função custo adotada no posicionamento global;
- Adicionar novas métricas a função custo, melhorando a solução em termos de roteabilidade, $timing$ e/ou dissipação de potência.

O $Simulated\ Annealing$ é um algoritmo que pode ser adotado para o posicionamento detalhado desde que a função de perturbação não viole a legalidade do circuito. Embora produza bons resultados, ele não costuma ser adotado devido ao tempo de execução elevado. Uma alternativa é o algoritmo $Brand\text{-}and\text{-}Bound$ [32] [33] que define janelas de células vizinhas e as reordena para achar a melhor combinação. Devido a complexidade computacional as janelas possuem no máximo 8 células tornando o algoritmo ineficiente.

O algoritmo Domino [34] produz bons resultados e é computacionalmente eficiente. Assim como o $Branch\text{-}and\text{-}Bound$, ele adota a estratégia de montar janelas de células vizinhas. A diferença é que divide as célula, cujo tamanho é representado por w_i, em w_i partes unitárias. Em seguida, move estas partes unitárias simultaneamente utilizando técnicas de fluxo máximo de custo mínimo em redes. O problema do fluxo máximo de custo mínimo em redes possui uma complexidade computacional menor que o $Branch\text{-}and\text{-}Bound$, permitindo janelas maiores.

Por fim, o ECO-System [35] é um $framework$ de particionamento robusto baseado na ferramenta CAPO [2]. Quando aplicado no posicionamento detalhado, é capaz de encontrar áreas onde a solução pode ser melhorada. Nestas áreas ele aplica legalização e posicionamento detalhado simultaneamente.

VI. Conclusão

O posicionamento é uma etapa de extrema importância para o fluxo de projeto de circuitos integrados. O assunto é alvo de pesquisas há mais de 50 anos pois o constante crescimentos dos circuitos proporciona o surgimento de novos desafios. Por isso, diversas abordagens algorítmicas surgiram ao longo do tempo. Atualmente as técnicas analíticas dominam as ferramentas estado da arte por terem a capacidade de tratar grandes circuitos de forma eficiente. Até pouco tempo atrás, o objetivo de otimização em posicionadores era apenas a redução do comprimento dos fios. No entanto, outras métricas ganharam destaque recentemente. Surge a demanda por trabalhos que busquem melhorar as soluções em termos de $timing$ e roteabilidade.

Por fim, este trabalho apresentou experiências com duas grandes técnicas de posicionamento: $SimulatedAnnealing$ e posicionamento quadrático. No $SimulatedAnnealing$ se mostrou a importância da escolha de um bom $schedule$ de temperatura para controlar o comportamento do algoritmo. No posicionamento quadrático se mostrou como o uso de $spreading\ forces$ é importante para otimizar os resultados em termos de $wirelength$ e tempo de execução.

REFERÊNCIAS

[1] C. Sechen and A. Sangiovanni-Vincentelli, "The timberwolf placement and routing package," *IEEE Journal of Solid-State Circuits*, vol. 20, no. 2, pp. 510–522, 1985.

[2] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov, "Capo: robust and scalable open-source min-cut floorplacer," in *Proceedings of the 2005 international symposium on Physical design*. ACM, 2005, pp. 224–226.

[3] A. R. Agnihotri, S. Ono, and P. H. Madden, "Recursive bisection placement: feng shui 5.0 implementation details," in *Proceedings of the 2005 international symposium on Physical design*. ACM, 2005, pp. 230–232.

[4] M.-C. Kim, D.-J. Lee, and I. L. Markov, "Simpl: An effective placement algorithm," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 1, pp. 50–60, 2012.

[5] N. Viswanathan, M. Pan, and C. Chu, "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*. IEEE Computer Society, 2007, pp. 135–140.

[6] J. Vygen, "Algorithms for large-scale flat placement," in *Proceedings of the 34th annual Design Automation Conference*. ACM, 1997, pp. 746–751.

[7] P. Spindler and F. M. Johannes, "Kraftwerk: a fast and robust quadratic placer using an exact linear net model," in *Modern Circuit Placement*. Springer, 2007, pp. 59–93.

[8] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, no. 7, pp. 1228–1240, 2008.

[9] R. A. d. L. Reis *et al.*, "Concepção de circuitos integrados," *Instituto de Informática da UFRGS. Editora Sagra Luzzatto*, 2000.

[10] N. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*. Addison-Wesley Publishing Company, 2010.

[11] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, *Electronic design automation: synthesis, verification, and test*. Morgan Kaufmann, 2009.

[12] R. F. Hentschke, "Algoritmos para o Posicionamento de células em circuitos VLSI," Master Thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre - RS, Brasil, 2002.

[13] S. Brooks and B. Morgan, "Optimization using simulated annealing," *The Statistician*, pp. 241–257, 1995.

[14] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in VLSI placement research," in *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*. IEEE, 2012, pp. 275–282.

[15] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 5, pp. 858–871, 2007.

[16] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *Proceedings of the conference on Design, automation and test in Europe*. EDA Consortium, 2007, pp. 1226–1231.

[17] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability-driven white space allocation for fixed-die standard-cell placement," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 4, pp. 410–419, 2003.

[18] T. Hamada, C.-K. Cheng, and P. M. Chau, "Prime: a timing-driven placement tool using a piecewise linear resistive network approach," in *Proceedings of the 30th international Design Automation Conference*. ACM, 1993, pp. 531–536.

[19] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for fpgas," in *Proceedings of the 2000 ACM/SIGDA eighth international symposium on Field programmable gate arrays*. ACM, 2000, pp. 203–213.

[20] A. B. Kahng and Q. Wang, "An analytic placer for mixed-size placement and timing-driven placement," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. IEEE Computer Society, 2004, pp. 565–572.

[21] R. F. Hentschke and R. Reis, "Improving simulated annealing placement by applying random and greedy mixed perturbations [ic layout]," in *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*. IEEE, 2003, pp. 267–272.

[22] J. Lam and J.-M. Delosme, "Performance of a new annealing schedule," in *Proceedings of the 25th ACM/IEEE Design Automation Conference*. IEEE Computer Society Press, 1988, pp. 306–311.

[23] ISPD04 IBM Standard Cell Benchmarks with Pads. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.public.iastate.edu/~nataraj/ISPD04_Bench.html

[24] N. Viswanathan and C.-N. Chu, "FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 5, pp. 722–733, 2005.

[25] R. HENTSCHKE, D. FIORENTIN, and R. REIS, "Uma comparação do processo de low annealing com high annealing aplicado ao posicionamento de células," in *Proceedings of the IX Workshop IBERCHIP, Havana, Cuba*, 2003.

[26] W. C. Naylor, R. Donelly, and L. Sha, "Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer," Oct. 9 2001, uS Patent 6,301,693.

[27] Executable Placement Utilities. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/PlaceUtils/

[28] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Design Automation, 1982. 19th Conference on*. IEEE, 1982, pp. 175–181.

[29] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in vlsi domain," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 7, no. 1, pp. 69–79, 1999.

[30] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," Apr. 9 2002, uS Patent 6,370,673.

[31] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," in *Proceedings of the 2008 international symposium on Physical design*. ACM, 2008, pp. 47–53.

[32] M. A. Breuer, "Min-cut placement," *J. Design Autmation and Fault Tolerant Computing*, vol. 1, no. 4, pp. 343–362, 1977.

[33] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Optimal partitioners and end-case placers for standard-cell layout," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 1304–1313, 2000.

[34] K. Doll, F. M. Johannes, and K. J. Antreich, "Iterative placement improvement by network flow methods," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 13, no. 10, pp. 1189–1200, 1994.

[35] J. A. Roy and I. L. Markov, "Eco-system: Embracing the change in placement," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 12, pp. 2173–2185, 2007.