UMA IMPLEMENTAÇÃO DE *SIMULATED ANNEALING* PARA POSICIONAMENTO DE CÉLULAS EM CIRCUITOS VLSI

Mateus P. Fogaça, Luisa R. Cornetet
Acadêmico do curso de Engenharia de Computação – FURG
{mateus.p.fogaca, cornetet.luisa}@gmail.com
Cristina Meinhardt, Paulo F. Butzen
Professores do Centro de Ciências Computacionais – FURG
{cristinameinhardt, paulobutzen}@furg.br

Resumo. *No fluxo de síntese de circuitos integrados, o posicionamento é a etapa responsável por encontrar a melhor posição de cada célula em uma área que representa a superfície do chip. Neste artigo é apresentada a implementação de uma ferramenta de posicionamento de circuitos VLSI baseada no algoritmo Simulated Annealing.*

Palavras-chave: *Microeletrônica. Ferramentas de CAD. Posicionamento.*

1. INTRODUÇÃO

O projeto de circuitos integrados de larga escala de integração (*Very-large-scale integration*, VLSI) pode ser dividido nas seguintes etapas: descrição em alto nível, síntese lógica e síntese física. Na descrição em alto nível, o sistema é modelado usando uma linguagem de descrição de *hardware* (HDL). Na síntese lógica, essa descrição é transformada em um conjunto de portas lógicas (células) conectadas entre si. Após é realizada a síntese física, cujas principais etapas são o posicionamento e o roteamento. A etapa do posicionamento é responsável por atribuir posições válidas para todas as células do conjunto de células que constituem o CI, em uma área que representa o *chip*. Ferramentas de posicionamento focam principalmente na redução do comprimento de fios necessários para conectar as células. Entretanto, é na etapa de roteamento que é decidida a melhor maneira de realizar esta conexão. Circuitos atuais contam com milhares de células para serem posicionadas e roteadas (HENTSCHKE, 2002).

Este trabalho trata sobre a etapa do posicionamento, apresentando o estudo de uma implementação de posicionamento com a meta-heurística *Simulated Annealing (SA).*

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os conceitos básicos sobre posicionamento de circuitos VLSI. A Seção 3 discute a aplicação da meta heurística *SA* em um algoritmo de posicionamento. A Seção 4 descreve a implementação do algoritmo. Na seção 5 é analisado o comportamento do algoritmo. Por fim, são apresentadas as conclusões e trabalhos futuros.

2. CONCEITOS

Nesta seção serão apresentados alguns conceitos básicos para o entendimento do problema do posicionamento. A Fig. 1 ilustra alguns destes conceitos que são descritos na sequencia.

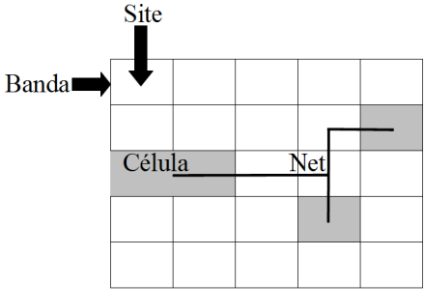

Figura 1. Representação gráfica de banda, *site*, célula e *net*.

Célula: É uma porta lógica presente na biblioteca chamada *Standard Cell Library*. 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.

Wirelength (wl): É uma estimativa que representa o comprimento das interconexões usadas para interligar as células.

Um aspecto importante do posicionador é a heurística utilizada para medir a qualidade do posicionamento. A abordagem mais correta seria, segundo Pinto (2010), rodar uma simulação elétrica para verificar os atrasos, capacitâncias e indutâncias dos fios a cada alteração na posição das células. Porém, isto deixaria o processo muito lento. Utiliza-se então a estimativa do comprimento dos fios (*wirelength*). Neste trabalho a estimativa do *wirelength* é feita considerando-se a distância *Manhattan* ponto-a-ponto, que é calculada usando apenas retas de 90º. Desta forma, a distância entre dois pontos é dada por:

$$d = \Delta x + \Delta y \qquad (1)$$

Para calcular o *wirelength* de uma *net*, calcula-se a distância *Manhattan* entre todos os pinos das células que compõe essa *net*. O *wirelength* total do circuito é dado pela soma dos *wirelength* de todas as *nets*. A Fig. 2 é utilizada para demonstrar cálculo do *wirelength* total para um circuito com quatro células (C1, C2, C3 e C4) e duas *nets* (C1-C2-C3 e C2-C4). Através do cálculo *Manhattan* ponto-a-ponto das *nets*, representadas pelas linhas tracejadas, é encontrado o valor de *wirelength* total de 10.

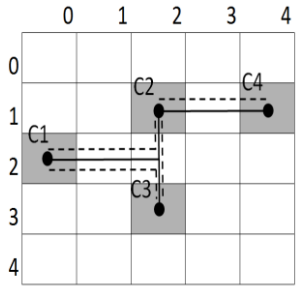

Figura 2. Estimativa do wirelength (linhas tracejadas) com Manhattan ponto-aponto de duas nets (linhas contínuas).

3. ALGORITMOS DE POSICIONAMENTO

Existem muitas soluções de posicionamento proprietárias e acadêmicas (KIM et al., 2002, WANG et al., 2000, VISWANATHAN et al., 2007). A maioria destas ferramentas realiza o posicionamento em várias etapas. Inicialmente é definida uma posição inicial para todas as células. Esta etapa é chamada de posicionamento construtivo. Nas etapas consecutivas, as ferramentas focam na otimização do *wirelength*, explorando algoritmos aleatórios ou baseados em meta-heurísticas, tais como *Simulated Annealing*, busca tabu ou algoritmos genéticos. As próximas subseções apresentam um algoritmo construtivo, responsável por criar um posicionamento inicial utilizado como base para as soluções de otimização e a aplicação da meta-heurística SA em um algoritmo de posicionamento de circuitos.

3.1 Algoritmo Aleatório

O algoritmo aleatório é bem simples, como apresentado na Fig. 3. Para cada célula é atribuída uma posição que não esteja ocupada. Por essa simplicidade, costuma ser muito rápido. Entretanto, costuma apresentar posicionamento com elevado *wirelength*.

```
Algoritmo 1: Posiciona_aleatório ()
1: para cada (Célula c na netlist)
2:     Coloque c em uma posição
       aleatória (x,y) não ocupada;
```

Figura 3. Pseudocódigo do algoritmo aleatório.

3.2 *Simulated Annealing*

O *Simulated Annealing (SA)* faz uma analogia a um processo utilizado na metalurgia, onde os metais são elevados a altas temperaturas, e depois resfriados lentamente de forma controlada. Assim é proporcionada uma melhor distribuição dos átomos do metal, formando uma estrutura cristalina (SU et al., 2001).

Uma versão de algoritmo *SA* é apresentada na Fig. 4. No problema do posicionamento, é necessário que as células estejam previamente dispostas. Então são realizadas trocadas de posições entre duas células selecionadas aleatoriamente e é analisado se o *wirelength (wl)* diminuiu. Quem controla a aceitação das trocas de posições é a temperatura do sistema, *temp*. Quanto maior a temperatura do sistema, maior a chance de ser aceita uma troca que aumente o *wirelength*. Essas trocas, aparentemente ruins, são responsáveis por permitirem que o algoritmo escape de mínimos locais. Com o avanço nas interações, a temperatura diminui de acordo com o coeficiente de resfriamento, *c_resfriamento*, gerando trocas mais criteriosas, ou seja, diminuindo as ocorrências de trocas "ruins".

4. PROJETO E IMPLEMENTAÇÃO

O foco deste trabalho é a implementação de uma ferramenta que, dado um conjuto de células, *nets* e a área de um *chip*, realiza o posicionamento das células visando minimizar o *wirelength* total das *nets*. Esta ferramenta foi implementada utilizando a linguagem de programação Java por ser multi-plataforma.

```
Algoritmo 2: Posiciona_SA ()
1: Inicialização
2:    temp = temperatura_inicial;
3: enquanto (wl estiver melhorando)
4:    Sorteie duas posições
      aleatórias C1 e C2;
5:    Troque C1 e C2 de lugar;
6:    Δwl = wl_antigo – wl_novo;
7:    se ( Δwl > 0 )
8:       Manter a troca;
9:    senão
10:      Coloque c em uma posição
       aleatória (x,y) não ocupada;
11:   se ( e^(Δwl/temp) > random (0,1))
12:      Manter a troca;
13:   senão
14:      Desfaça a troca;
15:  temp = temp/c_resfriamento;
```

Figura 4. Pseudocódigo do algoritmo *Simulated Annealing*.

O programa recebe dois parâmetros: as dimensões do *chip* e um arquivo com os dados do circuito. O circuito é descrito no formato de *netlist*, com as informações de nome da célula, conexões de entrada e saída e tipo de porta lógica. Nesta implementação, considerou-se todas as células possuindo tamanho 1.

Os dados são então armazenados em 4 classes: *Cell*, *Net*, *Netlist* e *Placement*. A classe *Cell* modela a célula, através de um identificador, sua posição e sua função lógica. A classe *Net* possui um identificador e a lista de *Cells* que a compõem. A classe *Netlist* armazena em uma lista todas as *Nets* do circuito e implementa o algoritmo para estimação do *wirelength*. Por fim, a classe *Placement* armazena uma matriz que representa a área do *chip* e implementa os algoritmos de posicionamento.

Uma vez que o posicionamento tenha sido executado, os dados de saída são armazenados em um arquivo texto no formato de posicionamento referencial. A primeira linha do arquivo define quantas bandas o circuito possui. Cada linha subsequente representa uma banda. O número inicial mostra quantos *sites* a banda possui. Este número será seguido de

identificadores, representado a célula que está posicionada em cada um dos *sites*. Um caractere *D* representa que o *site* não foi ocupado.

5. RESULTADOS

A Fig. 5 apresenta um gráfico com a evolução do *wirelength* ao longo da execução do algoritmo proposto. O *benchmark* adotado neste exemplo foi um circuito com 360 células em um *chip* de 20 bandas, cada uma com 20 *sites*. Inicialmente observa-se que a curva possui uma queda suave. Isto acontece porque em altas temperaturas o *SA* aceita muitas trocas ruins. Conforme a temperatura do sistema diminui a curva fica mais acentuada, pois a quantidade de trocas ruins também diminui. Após aproximadamente 600 iterações, a curva se estabiliza, e as trocas realizadas contribuem pouco para a redução do *wirelength*.

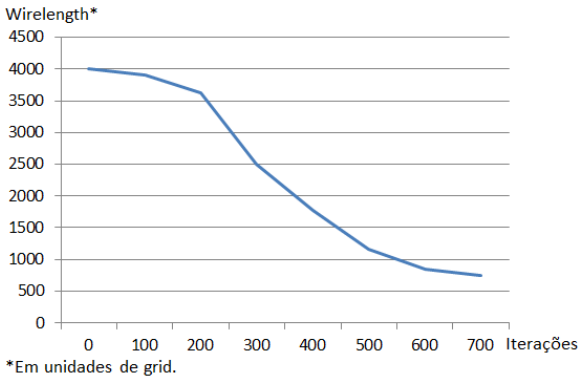

Figura 5. Gráfico *wirelength* x iterações com *Simulated Annealing*.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi apresentado um posicionador de células em circuitos integrados simples que faz uso do algoritmo *Simulated Annealing*. Os próximos passos incluem o estudo de outros métodos de estimativa do *wirelength*, de outros algoritmos de posicionamento, além da extensão para formatos de arquivos de entrada e saída utilizados nas ferramentas de CAD estado da arte.

7. REFERÊNCIAS

HENTSCHKE, R. F. **Algoritmos para posicionamento de células em circuitos VLSI**. 2002. 137 f. Dissertação (Mestrado em Microeletrônica) – Universidade Federal do Rio Grande do Sul, Porto Alegre. 2002.

KIM, Myung-Chul; LEE, Dong-Jin; MARKOV, Igor L. SimPL: An effective placement algorithm. **IEEE Transactions on CAD of Integrated Circuits and Systems,** v. 31, n. 1, p. 50-60, 2012.

PINTO, F. de A. **Posicionamento Visando Redução do Comprimento das Conexões**. 2010. 94 f. Dissertação (Mestrado em Microeletrônica) – Universidade Federal do Rio Grande do Sul, Porto Alegre. 2010.

SU, Lixin et al. Learning as applied to stochastic optimization for standard-cell placement. **IEEE Transactions on CAD of Integrated Circuits and Systems,** v. 20, n. 4, p. 516-527, 2001.

VISWANATHAN, Natarajan; PAN, Min; CHU, Chris. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In: **Proc. of the 2007 Asia and South Pacific Design Automation Conference**. IEEE Computer Society, 2007. p. 135-140.

WANG, Maogang; YANG, Xiaojian; SARRAFZADEH, Majid. Dragon2000: standard-cell placement tool for large industry circuits. In: **Proc. of International Conference on Computer-Aided Design**. IEEE Press, 2000. p. 260-263.