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

LUCAS DIOGO CAVALHEIRO

Posicionamento de Circuitos Integrados: Estudo e Implementação de Técnicas de Aprimoramento

Projeto de Diplomação

Prof. Dr. Ricardo Reis
Orientador

Felipe Pinto
Co-orientador

Porto Alegre, dezembro de 2010

"O insucesso é apenas uma oportunidade para começar de novo com mais inteligência."
— Henry Ford

AGRADECIMENTOS

Agradeço inicialmente aos meus pais, Neda e Antonio, por todo o apoio dado durante toda a vida estudantil, possibilitando que eu pudesse estudar em uma universidade pública sem precisar me preocupar com outras questões. Agradeço também as minhas irmãs, Aline e Patrícia, que, ao seu modo, contribuíram nesta caminhada. E não poderia esquecer da namorada (e futura esposa) Catarine, com quem convivo praticamente o tempo todo, com quem compartilho todos os sucessos e frustrações da vida acadêmica.

Na UFRGS, este trabalho não poderia ter sido feito sem a ajuda de diversas pessoas: Felipe Pinto, mestrando e co-orientador, principal incentivador deste trabalho e com quem muitas decisões foram discutidas e tomadas; Guilherme Flach, mestre em Microeletrônica, colega de GME, excelente programador e *workaholic* incondicional; e prof. Dr. Ricardo Reis, grande incentivador da pesquisa em Microeletrônica no Brasil, que, em meados de 2007, deu-me a possibilidade de iniciar o trabalho aqui apresentado, além de outros colegas de grupo e curso que, sempre que possível, contribuíram com as suas ideias mesmo nas conversas de bar.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CAD	Computer Aided Design
CPU	Central Processing Unit
DP	Detailed Placement
GP	Global Placement
GME	Grupo de Microeletrônica
HDL	Hardware Description Language
HP	Half-Perimeter
HPWL	Half-Perimeter Wirelenght
RAM	Random Acess Memory
SA	Simulated Annealing
VLSI	Very Large System Integration
WL	Wirelenght

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste trabalho, será tratado o problema de posicionamento de circuitos integrados. Desenvolveu-se um estudo dos diferentes algoritmos utilizados na resolução da questão e se implementou algumas soluções visando melhorar os resultados de posicionamento.

Além disso, procurou-se avaliar e investigar o reflexo das soluções propostas por diferentes métricas, abordando diversos conceitos de posicionamento de circuitos integrados relevantes atualmente.

Através das alterações propostas e executadas, foi possível obter um posicionamento 4 vezes mais rápido na média, com resultado em termos de comprimento de fio estimado 4% pior. Isto foi alcançado através do uso de processamento paralelo por meio da utilização da biblioteca OpenMP.

Também foi possível obter ganhos em tempo de execução através de técnicas de redução do tamanho do problema, sem, contudo, prejuízo em termos de comprimento de fio estimado. Inclusive, o algoritmo proposto se mostrou bastante eficiente na resolução de problemas de congestionamento.

Palavras-chave: Posicionamento, Circuitos Integrados, VLSI, CAD.

Integrated Circuits Placement: Study and Implementation of Improvement Technics

ABSTRACT

In this work, we treated the problem of placement of integrated circuits. A study of different algorithms used in the resolution of the issue has been developed and a few solutions were implemented to improve the results of placement.

In addition, this study sought to evaluate and investigate the effect of the proposed solutions by different metrics, addressing diverse placement concepts that are relevant today.

By the changes proposed and implemented, it was possible to obtain a placement four times faster on average, with the result in terms of half perimeter wire length 4 % worse. This was achieved through the use of parallel processing by using the OpenMP library.

We could also see gains in runtime using techniques to reduce the problem size, however without loss in terms of half perimeter wire length. Also, the algorithm was very efficient in solving congestion problems.

Keywords: Placement, Integrated Circuits, VLSI, CAD.

1 CONTEXTUALIZAÇÃO E INTRODUÇÃO

1.1 Contextualização

Ao longo dos últimos 50 anos, o número de transistores por circuito integrado tem dobrado a cada 2 anos, conforme descreve a Lei de Moore (MOORE, 1965). Para lidar com um volume tão elevado de elementos - de um processador Intel 4004, em 1974, com dois mil e trezentos transistores para, em 2006, um Intel dual-core Itanium com 1,7 bilhões de transistores(LIM, 2008), por exemplo -, desenvolveram-se técnicas para automatizar e otimizar esse processo. Pode-se desenvolver, por exemplo, um projeto *Full Custom*, ou seja, totalmente personalizado, em que se projeta cada transistor, assim como *Semi-Custom*, em que se utiliza bibliotecas de células pré-fabricadas (*Standard Cell*). Se no primeiro temos alto desempenho, no segundo temos desenvolvimento mais rápido e com menor custo. Como pontos negativos, no primeiro há um alto custo de projeto, ao passo que, no segundo, há perda de desempenho.

Este trabalho recai sobre uma das etapas da concepção de circuitos integrados quando utilizada a metologia *semi-custom* - o posicionamento. Na figura 1.1, podemos ver que a etapa de posicionamento se insere, resumidamente, após a descrição em linguagem de *hardware* (HDL) (descrição em linguagem de alto nível das funcionalidades do sistema) e síntese lógica (transformação da descrição HDL em portas lógicas padronizadas provenientes de uma biblioteca de células já previamente caracterizada). De posse das listas de redes em nível de portas lógicas (*gate-level netlist*), temos a etapa de posicionamento, alvo deste trabalho, em que se posicionam as portas lógicas, sob uma série de parâmetros de otimização, numa região que representa a área do circuito integrado. Após, na etapa de roteamento, o roteador recebe a descrição de conexões entre as células e decide qual a melhor forma de conectá-las, também sujeito a diversos parâmetros. Finalmente, realiza-se uma série de testes de estimativas de atraso, verificações de violações no projeto e, então, encaminha-se para a fabricação do circuito.

1.2 Introdução

Com o constante aumento de complexidade dos dispositivos eletrônicos, é cada vez mais necessária a melhora na eficiência das ferramentas de CAD para o desenvolvimento de sistemas VLSI de alta performance. Essa adequação torna-se necessária devido a uma mudança de paradigma: hoje, o atraso do circuito é determinado principalmente devido ao atraso das interconexões e não mais somente pelo atraso das portas lógicas. Dessa forma, obrigatoriamente as melhores soluções serão en-

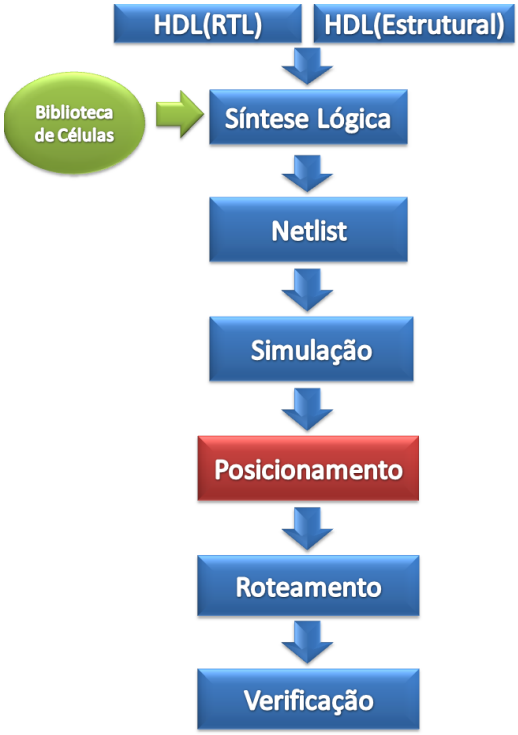

Figura 1.1: Fluxo de concepção de circuitos integrados (Top-Down)

contradas quando se tenta reduzir o atraso dos fios.

O posicionamento é uma etapa importante no processo de concepção de circuitos integrados, determinando fatores como comprimento das interconexões, área, roteabilidade, performance e outros. Nessa etapa, são definidas as posições dos componentes dentro do *chip*, de forma que posicionemos os elementos a fim de reduzir, principalmente, os comprimentos de fio que interconectam as células. Quanto mais células tivermos, mais complexo e demorado torna-se o problema de posicionamento, de modo que boas ferramentas devem escalar bem para um grande número de componentes.

Com o grande número de componentes que os projetos de circuitos atuais tem, deve-se pensar sempre em tempo de execução para as ferramentas de CAD. Logo, boas soluções para o problema podem ser encontradas através da melhor utilização do *hardware* disponível assim como do uso de técnicas que possibilitem diminuir o tamanho do problema.

Ao longo dos últimos anos, não se tem conseguido aumentar substancialmente a frequência de operação dos processadores com conjuntos de instruções semelhantes. Como alternativa, surgiram as arquiteturas *multi-core* como meio para se alcançar aumento de desempenho. Entretanto, de nada adianta termos várias unidades de processamento disponíveis se os programas escritos operam de forma serial e executam em apenas um núcleo. Dessa forma, torna-se necessário que as ferramentas de CAD, que requerem um alto grau de processamento, sigam a tendência e tornem-se paralelas (CATANZARO; KEUTZER; SU, 2008).

Em contraponto, sabe-se que não é elementar identificar perfis paralelos em algoritmos. Além disso, nem todos os problemas computacionais podem ser codificados paralelamente ou, até o presente momento, não há ferramental que proporcione

isso ser feito de forma rápida e eficiente. Ainda, como complicante, a programação paralela é fortemente dependente da arquitetura de *hardware*, dificultando a generalização da resolução de problemas utilizando essa metodologia.

Neste trabalho, serão apresentadas técnicas e soluções para os problemas atuais de posicionamento de circuitos integrados. No capítulo 2, serão apresentados os principais algoritmos para posicionamento e em quais posicionadores eles são empregados. A seguir, no capítulo 3, inicia-se a apresentação das técnicas, as implementações desenvolvidas e os resultados alcançados. Ao fim, no capítulo 4, concluiremos a análise do trabalho apresentado. Nos apêndices A e B, encontram-se em detalhes respectivamente as planilhas restantes de resultados e os *scripts* utilizados para os procedimentos de teste, a fim de que se possa verificar as variáveis utilizadas.

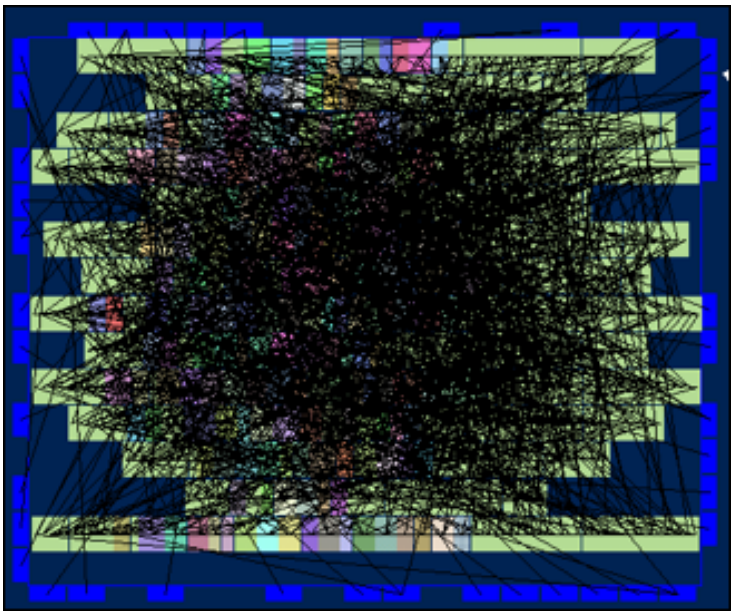

Figura 1.2: Posicionamento Randômico (HENTSCHKE, 2002)

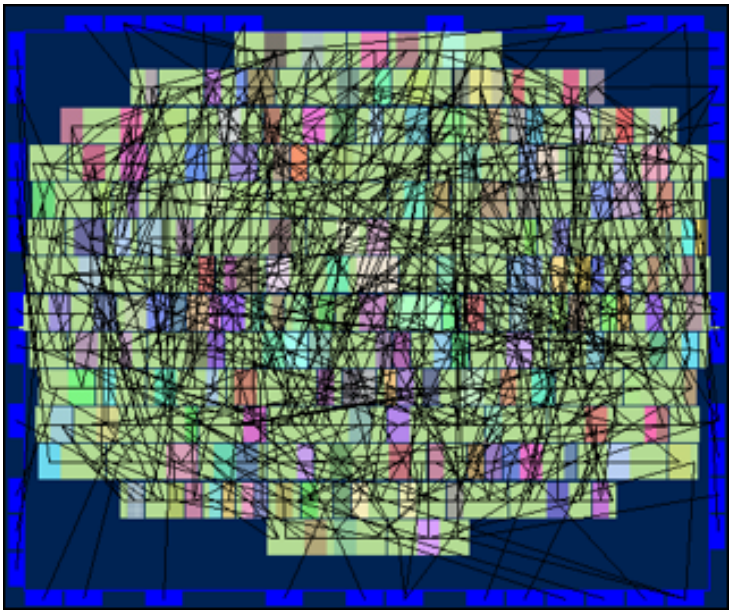

Figura 1.3: Posicionamento com Simulated Annealing (HENTSCHKE, 2002)

2 POSICIONAMENTO DE CIRCUITOS INTEGRADOS

O posicionamento é um problema NP-difícil - provado em (GAREY; JOHNSON, 1979), uma vez que o problema *Optimal Linear Arrangement* pode ser mapeado como um posicionamento -, ou seja, trata-se de uma versão otimizada de um problema NP-Completo. Além disso, o tamanho dos circuitos integrados atuais e a necessidade de se otimizar diversos objetivos simultaneamente torna difícil a tarefa de desenvolver soluções para esse problema. Por tudo isso, não é possível encontrar a melhor solução em tempo polinomial, sendo necessário o emprego de heurísticas de resolução. Na figura 1.2, temos um exemplo de um posicionamento aleatório e, na figura 1.3, um posicionamento utilizando *Simulated Annealing*. Percebe-se claramente a menor quantidade de fios interconectando as células na solução utilizando heurística de resolução, o que tem diversos reflexos no desempenho, área, *yield*, consumo, etc.

Os algoritmos de posicionamento podem ser divididos em 3 grandes grupos: Arrefecimento Simulado (***Simulated Annealing***), Particionamento Recursivo (***Recursive Partitioning***) e algoritmos dirigidos por força (***Force Directed methods***).

2.1 Simmulated Annealing

O ***Simulated Annealing*** é uma heurística que se originou de um processo térmico utilizado na metalurgia para obtenção de estados de baixa energia num sólido. O processo consiste de duas etapas: na primeira, a temperatura do sólido é aumentada para um valor máximo no qual ele se funde; na segunda, o resfriamento deve ser realizado lentamente até que o material se solidifique, sendo acompanhado e controlado esse arrefecimento. Nesta segunda fase, executada lentamente, os átomos que compõem o material organizam-se numa estrutura uniforme com energia mínima. Isto provoca que os átomos desse material ganhem energia para se movimentarem livremente e, ao arrefecer de forma controlada, dar-lhes uma melhor chance de se organizarem numa configuração com menor energia interna, para ter, como resultado prático, uma redução dos defeitos do material.

De forma análoga, o algoritmo de arrefecimento simulado no posicionamento substitui a solução atual por uma solução próxima (isto é, na sua vizinhança no espaço de soluções), escolhida de acordo com uma função objetivo e com uma variável T, dita Temperatura, por analogia. Quanto maior for T, maior a probabilidade de aceitação da nova solução, mesmo que a nova solução seja pior do que a atual. Essa metodologia faz com que se fuja de mínimos locais da função custo, tornando

possível alcançar um mínimo global ou a solução ótima (figura 2.1). Abaixo temos um exemplo de procedimento para o uso de SA para posicionamento.

`Procedimento Simulated Annealing`

```
1: temp = temperaturaInicial;
2: posicionamento = posicionamentoInicial;
3: // Loop Externo
4: while ( condicao para continuar o loop externo )
5:     // Loop Interno
6:     while ( condicao para continuar o loop interno )
7:         novoPosicionamento = Perturbacao(posicionamento);
8:         delta = Custo(novoPosicionamento) - Custo(posicionamento);
9:         if (Aceita(delta, temp)) then
10:            posicionamento = novoPosicionamento;
11:        end if
12:    end while
13: temp = Historico(temp)
14: end while
15: return  posicionamento
```

As funções `Custo` e `Aceita` podem ser variadas. No posicionamento, a função `Custo` mais usada é a de HPWL, em que a perturbação do estado é feita através da troca de duas células de posição. Parte-se de uma solução inicial aleatória e, com o baixar da temperatura, começam a ser aceitas somente as trocas de células que diminuam o comprimento de fio. Também é comum haver outras ponderações na função `Custo`, como congestionamento, por exemplo. Já para a função `Aceita`, normalmente usa-se a função conhecida como Fator de Boltzmann, que é dada por $e^{(-delta/T)}$.

O *Simulated Annealing* foi muito utilizado nos anos 80 e 90, mas nos dias de hoje perdeu força. Embora conhecidamente produza excelentes resultados em termos de comprimento de fio e o algoritmo seja de fácil implementação, sofre de um grande problema de escalabilidade devido ao seu caráter aleatório. Esse problema de escalabilidade tem sido resolvido utilizando particionamento, que divide o circuito em problemas menores, seguido de SA ou através de paralelização(BANERJEE; JONES; SARGENT, 1990; CHANDY; BANERJEE, 1996). Ambas soluções levam a menores tempos de execução, porém tem como revés a piora do resultado. No primeiro caso, perde-se a noção global do circuito, levando a resultados bons localmente na partição mas não necessariamente bons globalmente. No segundo caso, uma vez que a paralelização é feita movendo-se duas ou mais células simultaneamente, ocorre o fenômeno de erro acumulado(BANERJEE; JONES; SARGENT, 1990), em que a função custo calculada não necessariamente representa o atual resultado. Hoje, o principal posicionador acadêmico que utiliza *Simulated Annealing* é o Dragon (WANG; YANG; SARRAFZADEH, 2000; TAGHAVI; YANG; CHOI, 2005; TAGHAVI et al., 2006).

2.2 Particionamento Recursivo

Utilizando o princípio de dividir para conquistar, o **Particionamento Recursivo** decompõe um posicionamento em instâncias menores, subdividindo a região de

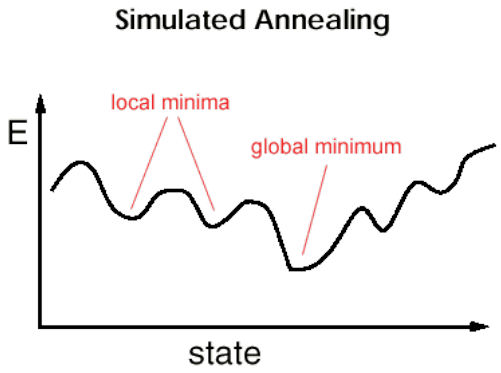

Figura 2.1: Função Custo, mínimos locais e mínimo global

posicionamento e assinalando módulos a essas sub-regiões, reformulando restrições, efetuando cortes na *netlist* para então obter boas soluções do problema original através da resolução de problemas menores, combinando-os. Essa decomposição é feita utilizando corte mínimo (min-cut), que subdivide a *netlist* onde há menos conexões entre células. Dito de outra forma, tenta-se deixar na mesma partição células mais fortemente conectadas (fig. 2.2).

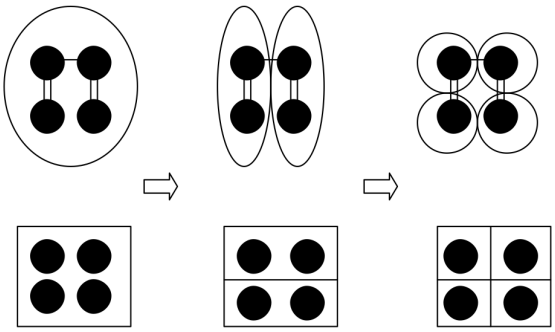

Figura 2.2: Exemplo de um particionamento recursivo

Uma partição é uma região do circuito, normalmente retangular, que é capaz de armazenar células e módulos e todas as redes que os conectam. O processo construtivo pode ser visto como uma sequência de ações em que, a cada passo, cada bloco ou partição é subdividido em pedaços menores para, ao final, conter apenas uma unidade (célula, macrobloco, etc.). Os dois principais algoritmos hoje utilizados são Fiduccia-Mattheyses (FIDUCCIA; MATTHEYSES, 1982), utilizado no Capo (CALDWELL; KAHNG; MARKOV, 2000; ROY et al., 2005) e h-Metis (KARYPIS et al., 1997), utilizado no Dragon (WANG; YANG; SARRAFZADEH, 2000; TAGHAVI; YANG; CHOI, 2005; TAGHAVI et al., 2006).

Em geral, os circuitos posicionados com particionamento recursivo apresentam menor congestionamento, o que representa maior facilidade para a etapa de roteamento, muitas vezes por subutilização da área do circuito - chamadas *bald areas*, ou áreas "carecas". Nos últimos anos, os dois representantes acadêmicos dessa técnica, Capo e Dragon, apresentaram resultados piores do que os outros posicionadores acadêmicos do estado da arte. Embora Dragon tenha sido o mais rápido entre todos os posicionadores, essa melhora no tempo de execução veio ao custo de resultados

ruins em termos de comprimento de fio (estimado)(NAM et al., 2005; NAM, 2006). Já o Capo apresentou o pior desempenho entre todos os posicionadores do *ISPD2006 Placement Contest* (NAM, 2006). Tais resultados podem ser explicados pela característica do particionamento, que é a primeira etapa do processo e que já promove definições contundentes sobre o resultado final do posicionamento. Uma vez que a célula é assinalada a uma partição, ela jamais mudará de partição. Logo, se algum erro ou imperfeição do algoritmo de particionamento se mostrar evidente, teremos grande comprometimento dos resultados. No *ISPD2006 Placement Contest*, os cinco primeiros colocados são posicionadores analíticos e todos utilizam alguma técnica de clusterização para melhorar o tempo de execução (NAM, 2006).

2.3 Algoritmos de Força Direta e Posicionamento Quadrático

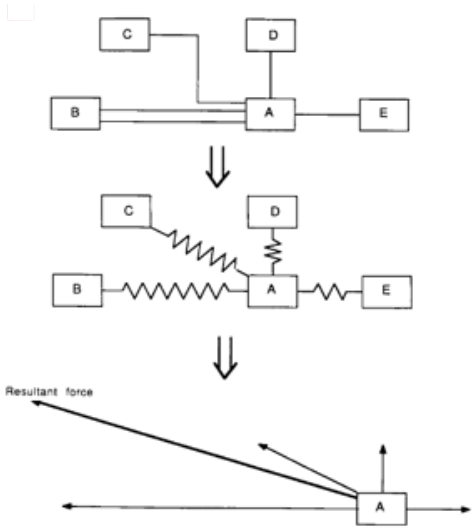

Figura 2.3: Modelando o problema de posicionamento como um sistema mola

2.3.1 Algoritmos de Força Direta

Nos **algoritmos de força direta**, introduzimos uma força de atração ou repulsão aos nodos ou conexões de acordo com alguma propriedade, normalmente referente a princípios físicos, como sistema mola - Lei de Hooke, e interação entre cargas elétricas - Lei de Coulomb. Um importante subconjunto desses algoritmos são os métodos analíticos, como posicionamento quadrático. No posicionamento quadrático, modelamos o problema de posicionamento como um sistema massa mola, em que as células do circuito são as massas e as interconexões são as molas (fig. 2.3). Duas massas ou, no caso, células são atraídas quando conectadas por uma interconexão. Essa força de atração depende do coeficiente da mola e da distância entre as duas massas, o que é um modelo interessante para posicionamento, uma vez que desejamos encurtar fios através da justaposição das células conectadas.

O comprimento de fio em posicionamento quadrático é medido pela distância euclidiana ao quadrado, portanto trata-se de um problema convexo e quadrático. Esse sistema pode ser resolvido por um sistema linear que indica o estado de equilíbrio, ou seja, o estado em que as molas compensam suas forças. Como resposta, teremos um posicionamento com as células agrupadas ao centro (fig. 2.4), com um

grande número de sobreposições (*cell overlapping*). Iterativamente, deve-se adicionar uma nova força ao sistema, de espalhamento (*spreading force*)- contrária à força de atração -, a fim de reduzir as sobreposições e melhorar o congestionamento, com as células migrando para regiões menos povoadas a cada iteração (fig. 2.5). Novamente, princípios físicos podem ser utilizados para prever o comportamento do algoritmo, como o da difusão de gases, que tendem a se espalhar para áreas menos densas, por exemplo.

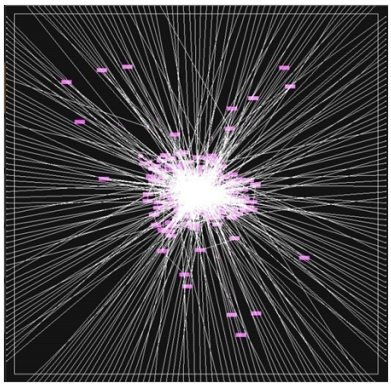

Figura 2.4: Resultado do posicionamento após primeira resolução do sistema linear

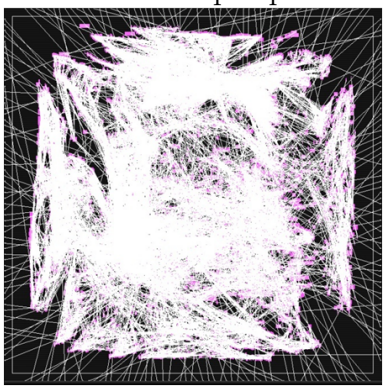

Figura 2.5: Resultado do posicionamento após algumas iterações

2.3.2 Posicionamento Quadrático

Comumente divide-se a etapa de posicionamento em duas fases: Posicionamento Global e Posicionamento Detalhado. A primeira etapa é a mais demorada e mais importante no processo, pois são estimadas em que regiões as células ficarão, o que praticamente define características como ocupação do circuito, comprimento de fio total, atraso, etc. Nessa etapa, as células podem se sobrepor ou estar em posições inválidas. No Posicionamento Detalhado, todas as células devem estar em posições válidas e sem sobreposições, sendo essa etapa encarregada de realizar pequenos movimentos de ajuste no leiaute, tornando-o legal. Essa abordagem para posicionamento foi popularizada pelos posicionadores quadráticos.

Inicialmente, aplica-se um modelo de redes às conexões do circuito, como o modelo híbrido, em que redes de tamanho 2 e 3 são modeladas como grafos completos, enquanto que toda rede com mais de três pinos é transformada em estrela. Após transformar as listas de redes, aplica-se a função quadrática de comprimento de fio $\phi(x, y)$, conforme equação 2.1. Pode-se observar que ambas coordenadas x e y po-

dem ser otimizadas independentemente com $\phi(x)+\phi(y)$, como se vê na equação 2.2. Aplicando a derivada na equação, a solução pode ser encontrada no ponto de mínimo (derivada igual a zero). Após alguns procedimentos algébricos, cada coordenada é resolvida pelo sistema de equações $Q \times x = dx$, em que Q representa a matriz de conectividade das células que podem ser movimentadas, x é o vetor desconhecido que representa a posição das células que podem ser movidas e dx é o vetor solução. Mais detalhes sobre como essas matrizes podem ser obtidas são encontrados em (ALPERT et al., 1997).

Nas equações 2.1 e 2.2, demonstramos o passo na equação de minimização do comprimento de fio global do circuito. Nota-se que ao elevar a primeira equação ao quadrado retiramos a raiz e facilitamos o cálculo, pois podemos calcular um eixo de cada vez e também temos uma operação a menos a resolver. No entanto, dessa forma, o resultado agora será o quadrado do comprimento de fio, um objetivo indireto. O sistema de equações pode ser resolvido utilizando um método Gradiente Conjugado Precondicionado com Fatorização Incompleta de Cholesky (KERSHAW, 1978) da matriz Q como precondicionadora.

$$\phi(x,y) = \frac{1}{2}\sum_{i,j=1}^{n}[\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}]^2 \tag{2.1}$$

$$\phi(x,y) = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2 + \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}(y_i - y_j)^2 \tag{2.2}$$

3 TÉCNICAS DE APRIMORAMENTO PARA POSICIONAMENTO DE CIRCUITOS INTEGRADOS

3.1 Paralelização do Posicionador Global PlaceDL

Neste trabalho, utilizou-se como ponto de partida o posicionador global PlaceDL (LIMA et al., 2009) e paralelizou-o a fim de reduzir seu tempo de execução. Para a paralelização, valeu-se da biblioteca OpenMP (OPENMP VERSION 3.0 COMPLETE SPECIFICATIONS, 2008), que dispõe de facilidades necessárias para uma boa conversão do algoritmo.

3.1.1 Posicionador Global PlaceDL

3.1.1.1 Classificação

O PlaceDL é um posicionador global que se utiliza de um algoritmo quadrático para resolver o problema de posicionamento. Nesse modelo, o objetivo é minimizar o somatório do quadrado da distância euclidiana das células conectadas, conforme visto em detalhes na subseção 2.3.2. Na figura 3.1, temos a descrição visual das etapas constituintes do posicionador PlaceDL.

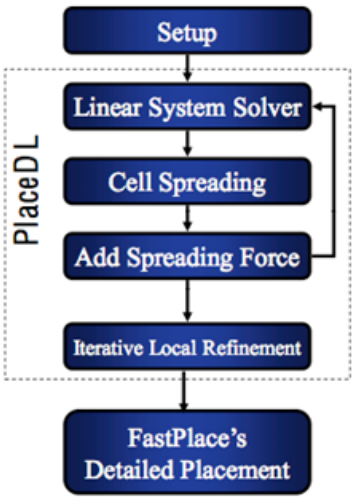

Figura 3.1: Fluxo da etapa de posicionamento utilizada no PlaceDL

3.1.1.2 Difusão Controlada e Refinamento Local Iterativo

Para tratar simultaneamente a minimização de comprimento de fio e a minimização da sobreposição entre as células, o PlaceDL intercala dois métodos com objetivos opostos: (i) minimização do comprimento de fios através da solução de um sistema linear e (ii) minimização da sobreposição através de um método que simula a difusão das células para regiões menos densas. A minimização do comprimento de fios tende a aglomerar as células, reduzindo o comprimento de fios, mas aumentando a sobreposição entre as células. Já a minimização da sobreposição espalha as células elevando significativamente o comprimento de fios. O processo de difusão é iterativo: a cada passo, a utilização das regiões é calculada e então as células são espalhadas baseadas no gradiente das densidades de cada região. As células são difundidas em pequenos passos até que haja uma diminuição de mais de 25% na utilização da região mais densa ou até que haja um aumento na utilização em relação à iteração anterior. Pequenos aumentos na utilização máxima podem ocorrer já que o processo de difusão é calculado de maneira discreta. Esse processo pode ser visto na figura 3.2.

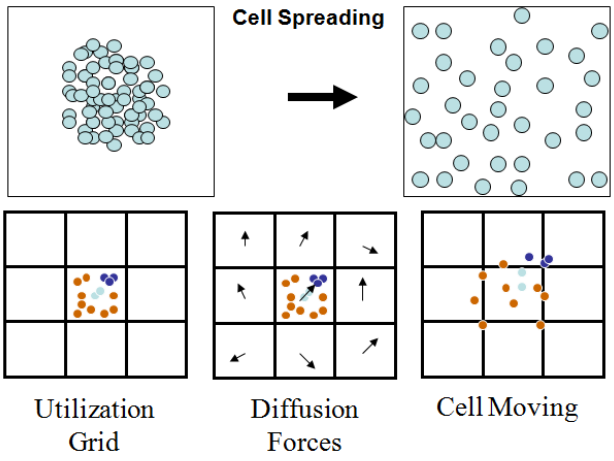

Figura 3.2: Ilustração do comportamento da difusão controlada de células

Para os cálculos de utilização, é utilizada uma grade de utilização que divide a área do circuito em regiões retangulares iguais de dimensão bin_{width} x bin_{height}. A utilização de uma região é definida como o somatório da área de sobreposição entre as células e a região. Note que uma célula pode contribuir para a utilização de mais de uma região. Ao final, a densidade de cada região é obtida dividindo-se a utilização pela área da região.

Após a etapa de difusão, é utilizado um refinamento local iterativo semelhante ao encontrado no posicionador FastPlace (VISWANATHAN; PAN; CHU, 2007) para reduzir o comprimento de fio total. Embora o processo de difusão controlada seja eficiente no espalhamento das células, acaba por aumentar significativamente os comprimentos de fios, fazendo ser necessária uma etapa de otimização desse objetivo. Então, sequencialmente, cada célula é deslocada para a mesma posição relativa nas regiões vizinhas ao norte, sul, leste e oeste. Para cada nova posição, um nova pontuação é calculado. Essa pontuação leva em conta a variação do comprimento de fio baseado no HPWL e a variação na utilização. A alteração do comprimento de fio é ponderada com 0,5 e as mudanças na utilização com 0,25, ou seja, variações em comprimento de fio tem prioridade na pontuação sobre as variações de utilização.

A célula será movida então para o vizinho com maior pontuação positiva. Se todos as pontuações forem negativas, a célula ficará na região atual. O processo de

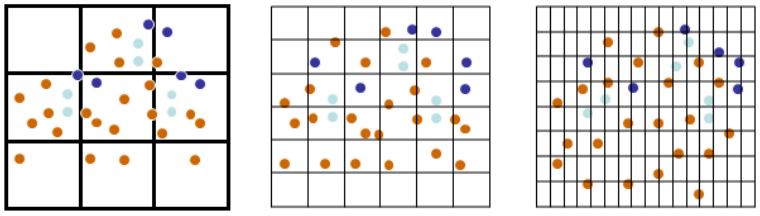

Figura 3.3: Variação do tamanho da grade durante o refinamento iterativo local

aprimoramento de fio é iterativo e varia o tamanho das regiões da grade regular de 10x o tamanho usado na etapa de difusão até 2x aquele tamanho (figura 3.3). Esses tamanhos de grades permitem que as células sejam movidas por distâncias maiores no começo do processo e, ao final, ocorram apenas ajustes finos. Essa fase de melhoramento do resultado é a que mais consome tempo em toda a etapa de posicionamento global no PlaceDL, tornando-se um bom alvo para paralelização. Além disso, ao observar a equação 2.2, nota-se que $\phi(x)$ e $\phi(y)$ podem ser calculados independentemente. Logo, temos aqui também um perfil para paralelização.

3.1.2 OpenMP

OpenMP (Open Multi-Processing) é uma API que possui diretivas de compilação, sub-rotinas e variáveis de ambiente para implementação de programas paralelos. Criada em 1997 inicialmente com suporte a Fortran, hoje está na versão 3.0 e possui suporte a Fortran, C e C++. Como características, seu uso se dá de forma explícita através de diretivas de programação no código; baseada em *threads*, seu uso é feito em sistemas com memória compartilhada utilizando o modelo *Fork-Join* de paralelismo, em que uma tarefa inicial gera outras n tarefas filhas. O motivo principal da escolha da OpenMP para este trabalho deve-se a grande facilidade para a paralelização do algoritmo, além é claro das garantias de desempenho que os desenvolvedores da biblioteca informam. Além disso, as características do problema direcionaram a escolha da ferramenta. Para paralelizar um laço for, por exemplo, basta que adicionemos uma diretiva de compilação antes do trecho de código da seguinte maneira:

```
#pragma omp parallel for
for (i = 0; i < N; i++)
        a[i] = large_computing();
```

Cada thread i irá executar a ação `a[i] = large_computing();`. Se tivermos N *threads* e N unidades de processamento no exemplo acima, o laço será executado totalmente em paralelo. Embora pareça que o tempo de execução do laço será N vezes menor, na prática, isso não ocorre. Uma vez que temos um sistema de memória compartilhada, o desempenho é limitado pela arquitetura de memória. Além disso, há também problemas relacionados a balanceamento das tarefas, que somente são descobertos durante a execução e, portanto, difíceis de prever em tempo de compilação.

3.1.3 Implementação

O posicionamento no PlaceDL pode ser dividido em 3 etapas: preparação dos dados (listas de redes, matriz de adjacências, etc.), ciclo de resolução do sistema linear e difusão controlada e, por fim, refinamento local iterativo. Dentre essas 3 etapas, as duas últimas são as mais custosas em termos computacionais, sendo a terceira fase a mais determinante no tempo total de posicionamento, principalmente para circuitos de grande porte. Dessa forma, o mais inteligente a ser feito é concentrar os esforços na paralelização do refinamento iterativo local. Iniciou-se a paralelização do algoritmo de resolução do sistema linear que, como dito anteriormente, pode ter as posições x e y resolvidas independentemente. Aqui, apenas se precisou fazer as chamadas do resolvedor linear de forma paralela para obter simultaneamente as matrizes Q_x e Q_y. Abaixo, código da alteração:

`Paralelização do cálculo do Gradiente Conjugado`

```
#pragma omp parallel sections {

#pragma omp parallel section {
SolveByConjugateGradientWithPCond(A,Qx,xold,maxIterations,error,U,L);
        }
#pragma omp parallel section {
SolveByConjugateGradientWithPCond(A,Qy,xold,maxIterations,error,U,L);
        }
}
```

O Refinamento Iterativo Local é, no PlaceDL serial, um laço `for` que possui como variável de controle o multiplicador do tamanho do `bin`. Assim, a cada iteração, os *bins*, assim com os movimentos de células, tornam-se cada vez menores. Na paralelização dessa parte do código, desenrolou-se o laço a fim de realizar chamadas em paralelo para o mesmo multiplicador - em vez de fazer chamadas em paralelo para diferentes multiplicadores, caso se paralelizasse puramente o laço -, ou seja, para cada valor de multiplicador são feitas 16 chamadas em paralelo da função de Refinamento Iterativo Local. Abaixo, o trecho de código com as alterações feitas:

`Paralelização do Refinamento Iterativo Local`

```
#pragma omp parallel sections {
#pragma omp parallel section {
        irl( multiplier_1, step, false, xold, yold ); /* x 16 */
   ...
  }
}
 #pragma omp parallel sections {
 #pragma omp parallel section {
        irl( multiplier_2, step, false, xold, yold ); /* x 16 */
        ...
  }
}
...
...
#pragma omp parallel sections {
#pragma omp parallel section {
```

```
            irl( multiplier_10, step, false, xold, yold ); /* x 16 */
            ...
    }
}
```

Dentro da função `irl()`, alterou-se o processo de seleção e movimentação de células, de forma a que se impossibilitasse que duas (ou mais) *threads* escolhessem a mesma célula e a movimentassem, fazendo com que o movimento final fosse a soma das ações. Para essa alteração, utilizou-se as primitivas `omp_set_lock()` e `omp_unset_lock()`.

Foram testados diferentes arranjos de número de *threads* e de número de *threads* por multiplicador, e a abordagem que obteve melhores resultados foi a descrita acima, uma vez que se utilizou de um computador com 8 *cores* com capacidade de *hyperthreading*. Essa abordagem possibilita maior aproveitamento da capacidade de processamento disponível, uma vez que há um melhor balanceamento de carga.

Devido a essa paralelização, ocorreram problemas de cálculo de ocupação dos *bins*. No funcionamento original do algoritmo de otimização, uma célula verifica uma baixa ocupação em determinada área do circuito e decide migrar. No entanto, esse movimento, como ocorre serialmente não precisa levar em conta a movimentação das demais células. Já na versão paralela, o cálculo não sabe que outras células também farão ou já fizeram a migração para a zona com baixa ocupação. Uma vez que todas as *threads* verificam simultaneamente a ocupação de um *bin*, temos o problema de migrações em massa células para regiões pouco ocupadas (FLACH et al., 2007), melhor visto nas figuras 3.4(b) e 3.4(c). Esse problema ocorre ciclicamente, levando a péssimos resultados tanto em termos de comprimento de fio quanto em tempo de execução. Para solucionar esse problema, duas soluções podem ser implementadas: (a) fazer a variável de utilização do *bin* levar em consideração o histórico das `n` últimas utilizações ou (b) tornar a migração de células estocástica. Em (a), a migração em massa de células para uma região pouco densa não torna a região bastante povoada anteriormente suscetível a receber novas migrações no ciclo seguinte, já que a variável de controle dirá que a região tem baixa utilização. Em (b), não há migração em massa, uma vez que se adiciona uma probabilidade de aceitação de movimento. Como contraponto, em ambos os casos podemos impedir com que ocorram movimentos que possivelmente fossem bons para o resultado final. Neste trabalho, foi implementada a solução (a), acompanhada de alterações em variáveis de controle de parada, como mudança da prioridade de comprimento de fio frente à utilização. O impacto dessas alterações no funcionamento do algoritmo pode ser melhor visto na figura 3.5.

3.1.4 Resultados

A fim de verificar o funcionamento do novo posicionador - agora paralelo - e seus resultados, o seguinte ambiente de teste foi montado: foram executados os posicionadores PlaceDL e PlaceDL Paralelo para posicionar globalmente os circuitos-teste IBM (IBM01 ao IBM13) (VISWANATHAN; CHU, 2004). Posteriormente, utilizou-se o posicionador detalhado FastPlace 3.0 (VISWANATHAN; PAN; CHU, 2007) para ter um resultado em termos de comprimento de fio sem sobreposições e sem células em posições inválidas.

O *hardware* do ambiente de teste foi um Apple Mac Pro® com dois processadores Intel® Xeon Nehalem 2.26GHz série 5500 (8 *cores* com capacidades de *hyperthrea-*

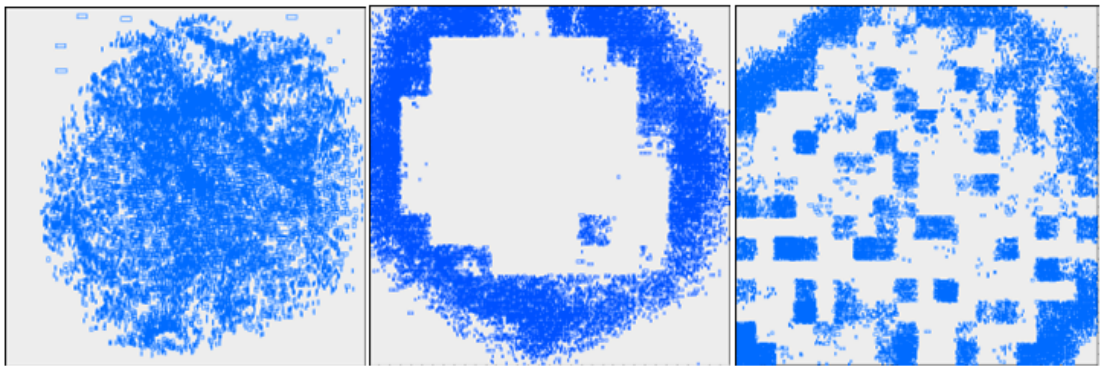

Figura 3.4: Efeito da migração em massa de células: (a) saída do resolvedor linear (b) saída após primeira iteração do refinamento iterativo local e (c) saída após quarta iteração do refinamento iterativo local

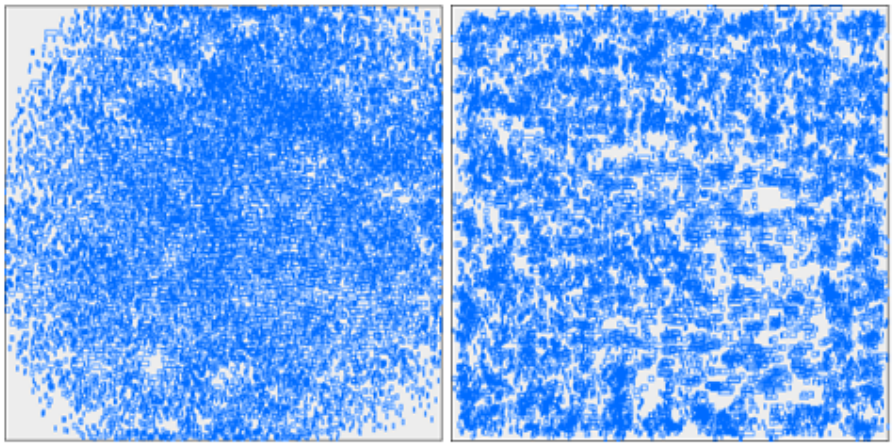

Figura 3.5: (a) nova saída do resolvedor linear e (b) saída após décima etapa do refinamento iterativo local

ding). A versão paralela foi executada com 16 *threads* simultâneas. O compilador utilizado foi o LLVM-GCC 4.2.

Conforme se vê na tabela 3.1, os ganhos em tempo de execução indicam que, de fato, há espaço para o desenvolvimento de soluções paralelas no campo da microeletrônica. Com as alterações, conseguiu-se redução no tempo execução de 80% do posicionamento global e de 74% no tempo total de execução da combinação posicionamento global e posicionamento detalhado. Dito de outra forma, todo o processo de posicionamento executa 4x mais rápido.

Por outro lado, a piora da qualidade do resultado nos mostra que há muito o que se aprender na paralelização dos algoritmos de CAD. Para os circuitos testados, o resultado teve média 4% pior em termos de HPWL. Percebe-se também, devido aos problemas de sobreposição de células expostos na subseção 3.1.3, que o tempo para o posicionamento detalhado executar aumentou em 18%. Embora esse tempo não seja dominante sobre o tempo total de posicionamento, é importante salientar essa diferença. Para efetuar a paralelização de uma ferramenta tão especializada como um posicionador, não basta apenas cuidar das regiões críticas para não haver *deadlocks*, por exemplo, mas também é necessário estudar as alterações nos padrões de comportamento de cada algoritmo quando paralelizado e suas condições de corrida. Essas alterações exigem um cuidado especial às variáveis e constantes que

Tabela 3.1: Comparativo entre PlaceDL Serial e PlaceDL Paralelo

ckt	GP Time	DP Time	Total Time	HPWL	SpeedUp
ibm01	0,107	1,098	0,178	1,039	5,625
ibm02	0,125	1,189	0,198	1,031	5,055
ibm03	0,154	1,333	0,238	1,054	4,203
ibm04	0,188	1,310	0,268	1,056	3,726
ibm05	0,170	1,302	0,230	1,051	4,347
ibm06	0,156	0,947	0,236	1,051	4,235
ibm07	0,216	1,135	0,273	1,043	3,662
ibm08	0,193	1,161	0,263	1,037	3,801
ibm09	0,179	1,037	0,229	1,031	4,371
ibm10	0,221	1,449	0,280	1,043	3,568
ibm11	0,329	1,016	0,385	1,032	2,596
ibm12	0,222	1,118	0,283	1,037	3,529
ibm13	0,246	1,231	0,307	1,057	3,255
Avg	**0,193**	**1,179**	**0,259**	**1,043**	**3,998**

dão normalmente o ajuste fino das decisões de parada dos algoritmos, devendo ser estudadas caso a caso.

3.2 Logical Core

Durante o posicionamento global, são definidas muitas características do circuito que dificilmente se alterarão profundamente durante o restante do processo. Um equívoco neste momento pode representar um circuito que não atende às especificações. Fazendo uma analogia, cada circuito pode ser visto como um novelo de lã embolado. No posicionamento, devemos tentar ao máximo desembolar esses fios. A Logical Core (PINTO et al., 2010) tenta prever, antes do posicionamento, o quão embolado está esse "novelo". Desse modo, o estudo desta técnica e a sua utilização no posicionamento de circuitos poderá ser benéfica para a melhora dos resultados.

A Logical Core analisa o grafo de descrição do circuito, verificando um conjunto de caminhos que compartilham nodos. O objetivo do algoritmo é determinar como se dá o relacionamento entre cada célula do circuito e suas redes, obtendo uma métrica de complexidade de posicionamento do circuito. Esse conhecimento pode ajudar o comportamento do posicionador no controle dos resultados.

O algoritmo propõe que o circuito pode ser visto como uma coleção de caminhos da entrada para a saída. Esses caminhos podem ou não ter células em comum. Na prática, temos que uma célula participa de vários caminhos, assim como alguns caminhos diferentes possuem várias células em comum. Conforme pode ser visto no gráfico 3.6, calculado para o *benchmark* ibm01 (VISWANATHAN; CHU, 2004), selecionando apenas 9% das células do circuito, estamos manipulando 42% dos caminhos do circuito.

A técnica proposta é baseada no algoritmo do Google Page Rank® . Explicando superficialmente, o Google Page Rank® pondera o número de links para um domínio vezes o peso de cada domínio que fez o link. Esse cálculo determina a relevância de um site na Internet. Para o caso de posicionamento de circuitos integrados, calcula-se a importância de cada célula para o posicionamento considerando o número de redes que chegam a uma determinada célula vezes a importância de cada célula que

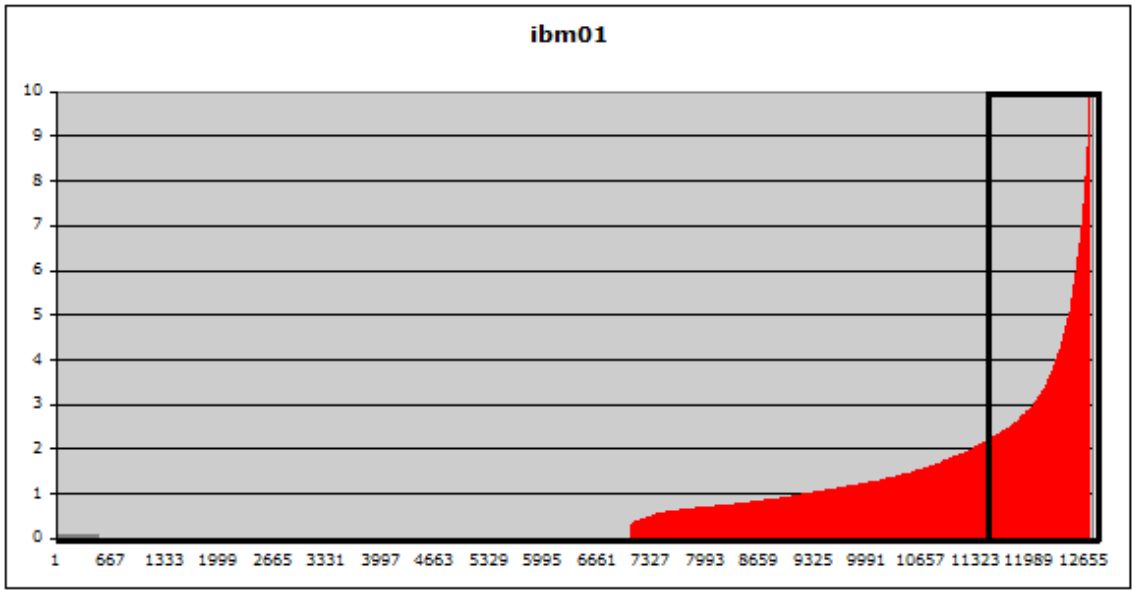

Figura 3.6: Distribuição Probabilística do ibm01

é a origem dessa conexão. Abaixo, na equação 3.1, temos como é calculada a Logical Core e o Google Page Rank® :

$$P(C) = (1 - \delta) + \frac{\delta(P(\tau 1)}{X(\tau 1)} + ... + \frac{\delta(P(\tau \nu)}{X(\tau \nu)} \tag{3.1}$$

em que a função `P` significa o valor da célula, que por sua vez é definido pela soma dos pesos de todas as células que apontam para `C`. A função `X` retorna o número de células que apontam para `C`. Já δ é a variável de *damping*. *Damping*, ou amortecimento, é a variável que tem por objetivo reduzir a amplitude da oscilação de um sistema oscilatório, que é o caso da Logical Core. Quanto menor é o valor de δ, mais rápido o sistema converge, porém menos preciso é o resultado e menor é o alcance em profundidade no grafo da rede. Imaginando o circuito como um sistema de baldes (células) e encanamentos (fios), δ representa o quanto de água é repassado ao próximo recipiente. Assim sendo, pode ocorrer de "faltar água"para encher os baldes que estão para o fim do sistema pois os baldes que estão no começo ficaram muito cheios. Caso seja usado um valor baixo de *damping*, teremos muitos valores em 0 nas células, o que mostra o baixo alcance que o algoritmo teve, refletindo em imprecisão para a utilização da técnica. Conforme pode ser visto em (PINTO et al., 2010), o valor de δ utilizado foi 0.99 para o cálculo das probabilidades de cada célula.

Na figura 3.7, temos um circuito hipotético com os valores da Logical Core em cada nodo.

3.3 Clusterização

A tradução literal para o substantivo *cluster* é grupo e, para o verbo, agrupar ou aglomerar. Em posicionamento, formar um *cluster* significa reunir duas ou mais células do circuito. O principal objetivo da etapa de clusterização é reduzir o tamanho do problema a ser resolvido, já que, através de agrupamentos, reduzimos o número de elementos a serem tratados, assim como o número de redes (geralmente

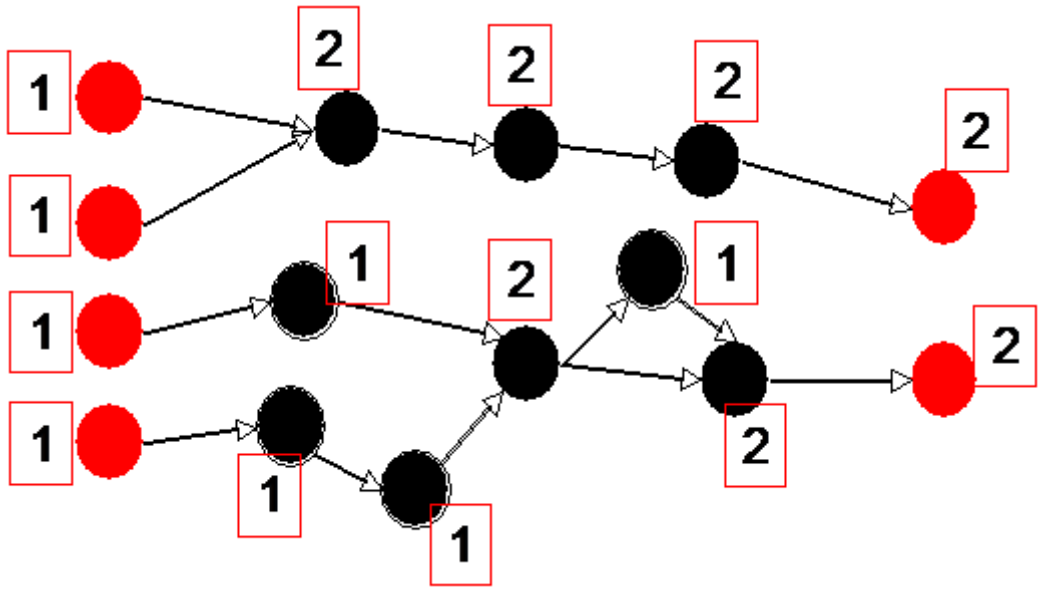

Figura 3.7: Exemplo de caminhos com pesos calculados pela Logical Core

ocorre, mas não é obrigatoriamente necessário). Isso implica reduzir a matriz de conectividade (ou estrutura representativa semelhante), uso de memória, tempo para percorrer todas as células do circuito, entre outros fatores. Com um circuito menor e mais simplificado, abre-se espaço para tratar melhor questões de comprimento de fio, ocupação, requisitos temporais, etc.

Por outro lado, toda simplificação leva a perda de informação. O cerne da clusterização, assim como de todo o princípio de compactação, é descartar informação redundante e preservar informação relevante. Em geral, no posicionamento, devemos agrupar células conectadas fortemente e manter separadas as desconexas ou com poucas conexões. Isso é feito a fim de reduzir o número de conexões entre *clusters*, o que é semelhante ao princípio do processo de particionamento, mas ao contrário - que divide ao invés de agrupar.

Além disso, há questões de granularidade dos *clusters*, em que se deve ponderar a área de cada agrupamento a fim de não formar muitos macroblocos - um complicante do processo de posicionamento - ou grupos de células muito largos, que levam a aumentos de comprimento de fio, pois limita-se o espectro possível de movimentos. Devido a todas essas variáveis, é difícil encontrar uma solução generalista e que funcione para todos os casos, que seja eficiente tanto na resolução do tamanho do problema quanto na otimização do HPWL, por exemplo.

Há basicamente duas maneiras de se efetuar a clusterização: *cluster* transiente e *cluster* persistente. No primeiro, a clusterização faz parte do núcleo do algoritmo, como uma etapa que é feita e desfeita iterativamente e em várias partes do circuito durante o posicionamento, de acordo com um determinado objetivo (ALPERT et al., 2005). No *cluster* persistente, transforma-se a *netlist* de entrada antes de entregá-la ao posicionador e, após o posicionamento global, desfazem-se os *clusters* para se retornar ao circuito original. A técnica empregada neste trabalho é a segunda, pois pode-se tratar a clusterização como um pré-processamento independente do posicionador, em que não se depende da disponibilidade do código fonte do posicionador e do trabalho despendido para a sua compreensão e alteração. Por outro lado, não

sabemos como cada posicionador irá se comportar perante a rede clusterizada e como se dará a interação entre a clusterização gerada pela ferramenta proposta e a implementada dentro do posicionador.

3.3.1 Logical Cluster

3.3.1.1 Proposta

Baseada na técnica Logical Core (PINTO et al., 2010), uma metodologia de clusterização foi implementada. Aproveitou-se as informações globais do grafo do problema de posicionamento que o método fornece para tentar acelerar o tempo de execução de posicionadores e, se possível, não haver prejuízos em termos de qualidade do resultado.

Conforme descrito anteriormente no capítulo 3.2, a Logical Core nos fornece uma relação de prioridade ou importância que as células tem no circuito, chamada aqui de probabilidade. Resumidamente, células com alto valor de probabilidade participam de mais caminhos, ou seja, são mais fortemente conectadas, enquanto que as com baixa probabilidade possuem poucas conexões e interferem pouco no resultado final de HPWL. Logo, parece nítido que devemos clusterizar as células com baixo valor de probabilidade, uma vez que, segundo a técnica, elas tem pouca relevância no circuito e podem ser ignoradas sem grandes prejuízos para se reduzir o tamanho do problema.

3.3.1.2 Algoritmo

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

`Algoritmo de Clusterização da LogicalCluster`

```
ordena (crescente) vetor de celulas de acordo com probabilidade
para cada celula ate LIM
        pega celula C1
        area_cluster = 0
        faca para cada celula conectada a C1
                pega celula C2
                se probabilidade(C2) < probabilidade(LIM)
                                clusteriza (C1, C2)
        enquanto area_cluster < 4,5*area_media_celulas
fim
```

No pseudo-código acima, `LIM` refere-se à célula limite no vetor ordenado crescentemente pelo valor de probabilidade. Essa célula limite é calculada da seguinte forma: `0.1, 0.2 ou 0.3 x número total de células` do circuito original. São visitadas uma porcentagem do total de células, da célula de menor probabilidade à célula limite. Essa abordagem faz com que tenhamos um padrão de clusterização entre os diferentes *benchmarks*. Além disso, como é percorrido um subconjunto do total de células, a clusterização é instantânea: o maior circuito testado, ibm18, que possui mais de 200 mil células, levou zero segundos para ser clusterizado na máquina de testes.

No algoritmo, utilizar um valor menor de porcentagem (0.1, 0.2 ou 0.3) representa, nesse caso, ser mais conservador na clusterização, pois agruparemos menos células e o impacto no circuito será menor. A escolha por limitar a área do *cluster* em

4,5 vezes a área média de uma célula foi baseada nos resultados de (HU; MAREK-SADOWSKA, 2003). A escolha pela área média garante também o padrão desejado para diferentes circuitos-teste. Foram realizados testes sem a limitação por área, que resultaram em péssimos resultados, tanto em termos de tempo de execução quanto de HPWL, demonstrando que os posicionadores atuais tem grande dificuldade em lidar com os circuitos mais recentes, que possuem diversos macroblocos (IPs).

Os circuitos clusterizados obtiveram reduções no número de células variando conforme o limite escolhido. Os detalhes podem ser vistos na tabela 3.2.

Tabela 3.2: Reduções no tamanho de cada circuito-teste, em %

CKT / %	0,1	0,2	0,3
ibm01	-1,24	-4,29	-9,35
ibm02	-2,66	-8,37	-15,90
ibm03	-0,06	-3,54	-8,30
ibm04	-0,51	-4,24	-10,21
ibm05	-3,68	-9,72	-15,46
ibm06	-1,77	-5,40	-11,39
ibm07	-0,74	-3,40	-8,15
ibm08	-1,20	-7,17	-14,58
ibm09	-0,65	-4,27	-9,78
ibm10	-1,97	-6,19	-12,15
ibm11	-1,63	-5,66	-11,11
ibm12	-1,55	-4,93	-9,75
ibm13	-1,96	-5,18	-11,50
ibm14	-1,24	-4,97	-10,37
ibm15	-2,15	-7,08	-14,18
ibm16	-2,11	-6,33	-13,00
ibm17	-2,10	-7,30	-13,56
ibm18	-2,57	-8,32	-15,35

3.3.1.3 Implementação

A fim de poder verificar o funcionamento da clusterização proposta, implementou-se, em C++, um gerador de descrições de *cluster*. O programa lê o arquivo de probabilidades gerado pela Logical Core, *ckt.prob*, e gera um novo arquivo com as descrições das uniões de células a serem feitas. O novo arquivo gerado tem a seguinte forma:

ckt.cluster:

```
cluster c1 c2
cluster c3 c4 c5
```

em que cada linha representa um *cluster* entre as células presentes na linha. Nesse exemplo, o circuito original tem 5 células e o clusterizado apenas duas. Os passos seguintes do processo estão indicados na figura 3.8.

3.3.1.4 Metodologia de Testes

Uma vez que não temos acesso aos códigos-fonte da maioria dos posicionadores, a abordagem para os testes teve de proceder de forma externa aos programas, ou seja, deu-se ao posicionador um circuito previamente alterado através da clusterização e, ao final do posicionamento, fez-se a conversão para o circuito original. Foram selecionados 3 posicionadores: FastPlace3 (VISWANATHAN; PAN; CHU, 2007),

NTUplace3-LE (CHEN et al., 2006) e PlaceDL. Para este último posicionador, a versão testada não é a mesma presente no capítulo 3.1.1; trata-se de uma versão nova, parcialmente reformulada e ainda sem resultados publicados, que utiliza particionamento recursivo para gerar um posicionamento inicial e, após, posicionamento quadrático. Este novo PlaceDL utiliza processamento paralelo e também só executa posicionamento global.

Uma vez que os 3 posicionadores selecionados permitem uma interrupção entre o processo de posicionamento global e detalhado, a desclusterização foi feita entre essas duas etapas. Essa abordagem foi a que apresentou melhores resultados. Todos os posicionadores do *ISPD2006 Contest* procedem de forma semelhante (LIM, 2008), ou seja, desfazem a clusterização antes de final do posicionamento para efetuar ajustes localizados no circuito.

Para cada posicionador, procedeu-se como mostrado na figura 3.8. Os tempos totais de execução presentes nas tabelas consideram os tempos de todos os programas necessários para se fazer a clusterização.

Os posicionadores FastPlace3 e NTUplace3-LE possuem implementações de clusterização que não podem ser desabilitadas. Logo, os resultados atingidos tratam-se de uma interação entre a metodologia de clusterização proposta e a clusterização com BestChoice (ALPERT et al., 2005).

O sistema operacional do ambiente de testes foi o Ubuntu 10.04. O *hardware* está formado com dois processadores Intel® Xeon Nehalem 2.26GHz série 5500 (8 *cores* com capacidades de *hyperthreading*) e 8GB de RAM. Embora apenas PlaceDL obtenha explícita vantagem em tempo de execução com a disponibilidade de várias unidades de processamento, o *hardware* proporciona aos outros dois posicionadores menor variabilidade nos tempos de execução, uma vez que, com 8 *cores* disponíveis, são pequenas as chances de o programa em execução "perder"a CPU.

3.3.2 Resultados

Todos os resultados obtidos foram realizados seguindo roteiro de testes visto na figura 3.8.

Conforme se vê na tabela 3.3, para os posicionadores FastPlace3 e NTUplace3-LE os resultados foram satisfatórios. Obtiveram-se melhorias em tempo de execução sem prejuízos em termos de HPWL, na média. Os resultados demonstrados nessa tabela são para as execuções com limite de 0.1 no algoritmo de clusterização, ou seja, a abordagem mais conservadora das propostas. Os dados de caso a caso serão mostrados e discutidos a seguir. Para fins de melhor leitura e diagramação do texto, algumas tabelas estão no apêndice A (Tabelas Extras).

As tabelas A.1, A.2, A.3 e A.4 mostram respectivamente os resultados para HPWL, tempo global, tempo detalhado e tempo total dos testes com **FastPlace3**. Analisando a tabela A.1 (HPWL, página 44), observamos que as 3 propostas não alteraram significativamente o resultado considerando a média. Mas é curioso notar que, para alguns casos, a clusterização proposta conseguiu melhorias de até 4% (ibm11) em termos de comprimento de fio estimado, mesmo com uma redução de apenas 1,63% no número de células do circuito (tabela 3.2, página 31). Na tabela A.2 (tempo global, página 45), vemos que a solução proposta é capaz de reduzir em até 20% o tempo de execução. Em compensação, analisando a tabela de tempos de posicionamento detalhado (A.3), temos uma piora de até 5% no tempo de execução do posicionador detalhado. Isso se deve ao fato de que a desclusterização produz

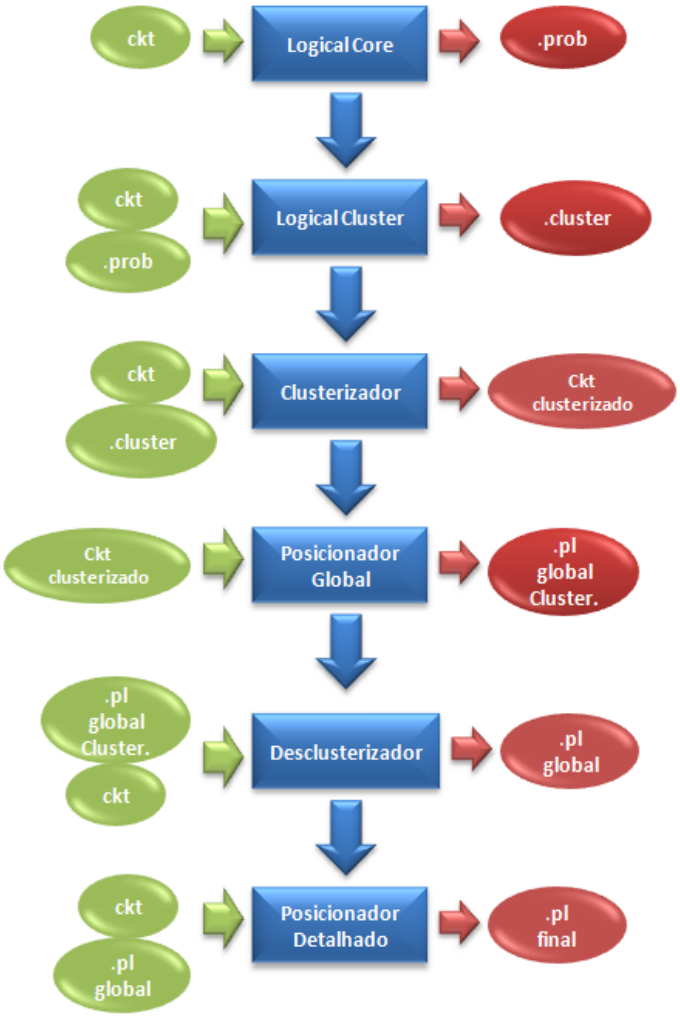

Figura 3.8: Roteiro do experimento e entradas e saídas de arquivo

sobreposições de células: durante o posicionamento global, ocorre uma legalização superficial do circuito, a fim de entregar uma solução com menos sobreposições para o posicionador detalhado. Ocorre que essa legalização, para esse experimento, é feita sobre o circuito clusterizado, ou seja, não representa de fato a legalização do circuito real. Para alguns casos (ibm17), o tempo de execução se compromete em até 78%, fazendo com que o tempo total seja dominado pelo tempo do posicionamento detalhado. Caso fosse possível integrar o algoritmo ao posicionador e se desfazer a clusterização proposta antes do fim do posicionamento global, certamente se teria um posicionamento detalhado equivalente ou com menos comprometimento em termos de tempo de execução.

As tabelas A.5, A.6, A.7 e A.8 mostram respectivamente os resultados para HPWL (página 46), tempo global (página 47), tempo detalhado (página 47) e tempo total (página 48) dos testes com **NTUplace3-LE**. Também para os 3 casos de teste, não houve significativas diferenças, na média, para os resultados em termos de HPWL, embora isoladamente tenha havido ganhos (ibm18) ou perdas (ibm16) de até 6%. Ao que parece, pelos resultados obtidos no ibm18, o posicionador sofre de problemas de escalabilidade, uma vez que, para o caso em questão, quanto maior

Tabela 3.3: Resultados comparativos dos circuitos posicionados com e sem clusterização (LogicalCluster) no FastPlace3, NTUplace3-LE e PlaceDL

Placer		FastPlace3		NTUplace3*		PlaceDL**	
ckt	#cells	HPWL	Runtime	HPWL	Runtime	HPWL	Runtime
ibm01	12506	1,000	1,139	1,006	0,777	1,011	1,087
ibm02	19342	0,992	1,030	1,013	0,900	0,987	0,854
ibm03	22853	1,000	1,051	0,987	0,894	0,997	1,137
ibm04	27220	0,985	0,734	1,017	0,818	0,994	1,080
ibm05	28146	1,067	0,893	1,002	0,753	1,096	1,151
ibm06	32332	1,000	0,828	0,987	0,930	1,006	0,942
ibm07	45639	1,003	1,107	1,010	0,769	1,001	0,983
ibm08	51023	1,006	0,797	0,990	0,863	1,005	1,037
ibm09	53110	0,976	1,057	1,004	0,899	1,009	1,065
ibm10	68685	1,028	0,869	0,999	0,858	0,984	1,051
ibm11	70152	0,957	0,707	1,015	0,985	0,978	1,066
ibm12	70439	0,980	0,949	0,980	0,847	1,003	1,080
ibm13	83709	0,997	0,925	1,001	0,772	1,012	1,044
ibm14	147088	0,980	0,870	0,991	0,970	0,998	1,029
ibm15	161187	1,027	0,861	0,996	0,785	1,012	0,916
ibm16	182980	0,980	0,796	0,959	0,989	1,028	1,011
ibm17	184752	1,025	1,135	0,986	1,000	0,979	0,962
ibm18	210341	0,995	0,858	0,974	1,239	1,074	1,088
	Avg	1,000	0,923	0,996	0,892	1,010	1,032

a redução do tamanho do circuito, maiores as vantagens em termos de resultado. A LogicalCluster também se mostrou eficiente na redução do tempo de execução do posicionador global, levando em média entre 18 e 28% menos tempo para executar. Já no posicionamento detalhado, os tempos de execução praticamente dobraram para todos os casos de limite. Na figura 3.9, podemos ver a saída do posicionamento global não alterado do NTUplace3-LE para o circuito ibm01. Nota-se que o posicionador global entrega um resultado com pouquíssimas sobreposições, facilitando em muito o trabalho do posicionador detalhado, o que não acontece no caso do clusterizado (figura 3.10) devido ao procedimento de desclusterização realizado para os testes e já explicado anteriormente.

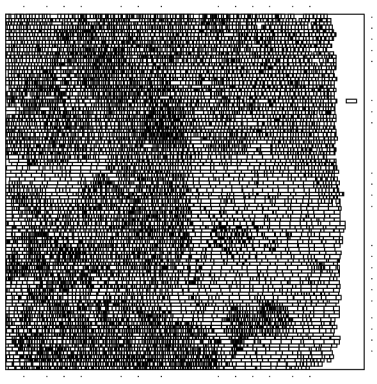

Figura 3.9: Saída do posicionamento global com NTUplace3-LE sem alterações

As tabelas A.9, A.10, A.11 e A.12 mostram respectivamente os resultados para HPWL (página 48), tempo global (página 49), tempo detalhado (página 49) e tempo total (página 50) dos testes com **PlaceDL**. Nesse caso, os resultados foram na

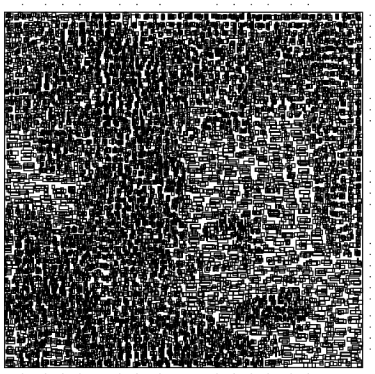

Figura 3.10: Saída do posicionamento global com NTUplace3-LE com LogicalCluster (limite 0.1) após desclusterização

média entre 1 e 2% piores em termos de HPWL e 3 e 6% mais lentos em termos de tempo de execução. Embora o posicionamento global tenha sido acelerado, houve comprometimento no tempo de execução do posicionamento detalhado, muito em função dos problemas de legalização que a abordagem escolhida gera.

3.3.3 Verificando a validade da Logical Cluster

A fim de verificar se realmente faz sentido aplicar a técnica Logical Core à clusterização, foram atribuídas probabilidades aleatórias (entre 1 e 10) a cada célula do circuito. Após, foram efetuadas as mesmas metodologias anteriores de clusterização e posicionamento (figura 3.8 e o algoritmo proposto na seção 3.3.1.2). Pode-se afirmar que, se a Logical Core não faz sentido ou não tem relação com o problema de posicionamento, não haverá diferença significativa nos resultados se agruparmos células utilizando como base os valores calculados pela Logical Core ou calculados aleatoriamente. Além disso, pelos resultados, poderemos provar se faz sentido ou não a clusterização de células com baixa probabilidade.

Nas tabelas 3.4 e 3.5 temos os resultados para posicionamentos no FastPlace3 utilizando as duas clusterizações. Como era de se esperar, os resultados da clusterização randômica foram piores tanto em tempo de execução quanto em resultado. Conforme o valor limite aumenta, ou seja, há maior intervenção no circuito, percebe-se a piora gradativa do resultado.

3.3.4 Avaliação de Congestionamento

Outra preocupação importante para o posicionamento de circuitos integrados, que tem ganhado foco nos últimos anos (ISPD - ROUTABILITY-DRIVEN PLACEMENT CONTEST, 2011), é a predição da roteabilidade do circuito. Para que um circuito tenha seu roteamento facilitado, é preciso que haja uma distribuição equilibrada das conexões ao longo do circuito. Reduzir comprimento de fio indiretamente reduz congestionamento (figura 1.3, página 14); no entanto, poderemos ter um mesmo circuito posicionado por duas ferramentas diferentes e com o mesmo comprimento de fio estimado, porém com níveis de congestionamento diferentes e, logo, comprimentos de fio reais (roteados) diferentes.

Por outro lado, durante o posicionamento, não se tem a informação de qual algoritmo de roteamento será usado. Sem conhecer exatamente por onde o fio irá passar, qualquer métrica utilizada será uma aproximação ao comportamento real.

Tabela 3.4: Resuldado em HPWL comparativo entre circuito clusterizado com LogicalCluster e randomicamente no FastPlace3

CKT / %	0,1	0,2	0,3
ibm01	1,005	1,027	1,003
ibm02	1,022	1,023	1,038
ibm03	1,061	1,015	0,997
ibm04	1,013	0,997	1,031
ibm05	0,935	0,983	1,019
ibm06	1,022	1,035	1,038
ibm07	1,013	1,050	1,087
ibm08	1,006	1,032	1,034
ibm09	1,026	1,000	1,016
ibm10	0,994	1,011	1,026
ibm11	1,036	1,013	1,015
ibm12	1,022	1,002	1,034
ibm13	1,021	1,009	1,016
ibm14	1,041	1,051	1,015
ibm15	0,998	1,087	1,039
ibm16	1,009	1,035	0,991
ibm17	1,014	1,000	0,974
ibm18	1,021	1,005	1,039
Avg	1,014	1,021	1,023

Para as estimativas apresentadas neste trabalho, utilizou-se da ferramenta *CongestionMaps Plotter* (CONGESTIONMAPS PLOTTER, 2004). Ela implementa os conceitos propostos em (WESTRA; BARTELS; GROENEVELD, 2004), que define uma estimativa probabilística de uso dos recursos de roteamento para cada *bin*. Nesse caso, congestionamento representa a razão entre o uso do *bin* e a sua capacidade. O tamanho do *bin* varia conforme o tamanho do circuito, mas trata-se de uma pequena porção da área total. Ao final da execução do programa, são relatados, por faixas de nível de congestionamento, um percentual das áreas com maior e menor congestionamento. Áreas com nível de congestionamentos acima de 50% representam regiões com utilização de recursos acima da capacidade do *bin*. Logo, nessas áreas, há menos espaço e liberdade para a passagem de fios durante o roteamento. Desse modo, circuitos com muitas áreas com congestionamento elevado devem ser mais difíceis efetuar o roteamento. Além disso, tais circuitos terão um comprimento de fio roteado maior, uma vez que o algoritmo de roteamento terá de contornar a região congestionada para chegar à célula destino e efetuar a conexão.

Os arquivos de posicionamentos utilizados para esta análise foram provenientes da ferramenta PlaceDL. Uma vez que a versão utilizada do posicionador não apresenta nenhum algoritmo de clusterização embutido, pode-se avaliar melhor o comportamento da solução proposta. Para fins de comparação, utilizou-se a saída convencional, sem alteração, do PlaceDL, e as saídas do posicionamentos com o circuito clusterizado através da LogicalCluster e do algoritmo BestChoice (ALPERT et al., 2005). Ambos testes, LogicalCluster e BestChoice, procederam da mesma forma exposta anteriormente em 3.3.1.4: clusterização, posicionamento global (PlaceDL), desclusterização e posicionamento detalhado (FastPlace3).

Tabela 3.5: Resuldado em tempo de execução comparativo entre circuito clusterizado com LogicalCluster e randomicamente

CKT / %	0,1	0,2	0,3
ibm01	0,954	1,053	1,134
ibm02	0,944	1,042	1,168
ibm03	0,903	1,114	1,107
ibm04	1,211	1,121	1,089
ibm05	1,254	0,962	1,279
ibm06	1,081	0,961	0,934
ibm07	0,931	0,982	1,174
ibm08	1,117	1,079	0,953
ibm09	0,927	1,057	1,126
ibm10	0,965	1,135	1,198
ibm11	1,207	1,183	1,010
ibm12	1,043	1,052	1,026
ibm13	0,918	1,055	0,999
ibm14	1,178	1,039	1,025
ibm15	0,993	0,935	0,841
ibm16	1,541	0,809	1,027
ibm17	1,158	1,013	1,035
ibm18	1,548	1,014	0,913
Avg	1,104	1,034	1,058

3.3.4.1 Resultados

A tabela 3.6 representa o relatório do *CongestionMaps Plotter* para a saída original do PlaceDL. Na primeira coluna temos as faixas de nível de congestionamento, divididas em bandas de 10%. Nas demais colunas temos o **percentual em área do circuito** que apresenta o congestionamento descrito na linha da tabela. A última linha da tabela representa o somatório das porções em área do circuito contendo congestionamento elevado, ou seja, uso maior do que a capacidade.

Percebe-se que, para os circuitos ibm04, ibm08, ibm12, ibm14, ibm15 e ibm16, o congestionamento não é um problema relevante, uma vez que para quase toda a área do circuito o congestionamento é baixo. Para fins de correta avaliação dos resultados, os experimentos seguintes serão feitas somente sobre os demais circuitos.

Ao se formar um *cluster*, fazemos com que duas células sejam posicionadas próximas uma da outra. Se agruparmos duas células com probabilidade alta, formamos um ponto de grande demanda por conexão no circuito. Se agruparmos as células com baixa probabilidade, aumentamos a demanda dessa zona e, por outro lado, diminuímos a diferença entre os centros de alta demanda por fio e os de baixa. Essa equalização entre as zonas de alta e baixa demanda por fio faz com que os recursos para o roteamento sejam melhor administrados durante o posicionamento, levando a um congestionamento menor do circuito.

Na tabela 3.7, temos o comparativo resumido entre as saídas dos 5 testes efetuados - circuito original, circuito clusterizado com BestChoice e circuito clusterizado com 3 valores limite diferentes para a LogicalCluster. Os valores estão padronizados para os do circuito não clusterizado. Os números comparados são referentes aos somatórios das áreas com congestionamento acima de 50% (última linha da tabela 3.6).

Novamente, é curioso notar como pequenas alterações do circuito provocam mu-

Tabela 3.6: Congestionamento: valores calculados para o PlaceDL sem alterações

ckt	ibm01	ibm02	ibm03	ibm04	ibm05	ibm06	ibm07	ibm08	ibm09
0-10	13,54	16,42	19,43	35,56	15,44	13,63	16,16	80,47	11,10
10 – 20	5,82	8,71	18,45	55,31	5,41	7,00	25,73	19,06	16,12
20 – 30	15,33	16,30	29,13	8,72	7,79	14,46	28,23	0,25	30,41
30 – 40	22,65	16,98	20,43	0,24	14,44	25,09	16,58	0,16	25,20
40 – 50	21,30	13,40	8,77	0,07	17,55	22,76	7,73	0,03	13,36
50 – 60	11,39	12,09	3,00	0,04	18,37	11,27	4,09	0,00	3,34
60 – 70	5,54	8,53	0,69	0,04	12,93	4,44	1,26	0,00	0,38
70 – 80	3,23	4,97	0,08	0,00	5,90	0,98	0,17	0,01	0,05
80 – 90	0,83	2,09	0,00	0,00	1,84	0,32	0,03	0,01	0,03
90 – 100	0,37	0,50	0,02	0,01	0,34	0,06	0,02	0,01	0,01
Σ > 50% (%)	21,36	28,19	3,79	0,10	39,37	17,07	5,57	0,02	3,81

ckt	ibm10	ibm11	ibm12	ibm13	ibm14	ibm15	ibm16	ibm17	ibm18
0-10	12,30	13,03	46,67	9,62	27,63	18,25	13,09	8,50	10,41
10 – 20	25,26	16,29	51,05	12,63	54,10	35,65	38,94	11,36	23,77
20 – 30	32,78	27,21	2,08	25,18	17,13	26,47	37,39	19,59	28,47
30 – 40	20,32	24,99	0,12	25,61	1,02	14,39	8,96	25,21	19,75
40 – 50	6,83	13,09	0,05	16,68	0,08	4,50	1,28	19,97	9,71
50 – 60	1,74	4,02	0,01	7,23	0,02	0,61	0,22	9,90	4,65
60 – 70	0,49	0,98	0,02	2,26	0,01	0,09	0,07	3,88	2,21
70 – 80	0,16	0,25	0,01	0,65	0,00	0,02	0,03	1,16	0,87
80 – 90	0,07	0,12	0,00	0,13	0,00	0,00	0,01	0,40	0,14
90 – 100	0,03	0,02	0,01	0,01	0,01	0,00	0,01	0,05	0,02
Σ > 50% (%)	2,50	5,39	0,03	10,29	0,05	0,73	0,34	15,38	7,89

danças tão drásticas no comportamento do congestionamento, como pode-se notar nas linhas referentes aos circuitos ibm07 e ibm09 da tabela 3.7. Na média, duas das três versões da LogicalCluster se comportaram melhor para o congestionamento do que a solução proposta pelo BestChoice.

Para os circuitos com maiores problemas de congestionamento, ibm01, **ibm02** e **ibm05** (tabela 3.6), todos os testes com a LogicalCluster apresentaram um comportamento semelhante ou vantajoso em relação ao circuito original ou clusterizado com BestChoice. As tabelas com os valores absolutos de todos os testes encontram-se no anexo A: BestChoice (tabela A.13, página 50), LogicalCluster com limite 0.1 (tabela A.14, página 51), LogicalCluser com limite 0.2 (tabela A.15, página 51) e LogicalCluster com limite 0.3 (tabela A.16, página 52)

Na figura 3.11, temos graficamente dois posicionamentos do circuito ibm02, sem e com clusterização, respectivamente. Ambos posicionamentos tem HPWL semelhante (clusterizado é 2,3% menor), porém a relação entre o somatório das áreas congestionadas difere em 33%. Nota-se claramente que a versão clusterizada reduziu as áreas congestionadas, representadas em amarelo e vermelho na imagem.

Outras técnicas propostas, como a adição de células falsas (*filler-cell*) para o preenchimento do espaço em branco (ADYA; MARKOV; VILLARRUBIA, 2003) tem resultado também satisfatórios, porém ao custo de acréscimo no tamanho do problema, ou seja, maior tempo de execução e possíveis problemas de escalabilidade do algoritmo, o que não ocorre no caso da clusterização.

Tabela 3.7: Congestionamento: comparativo entre o circuito original e com clusterizações diferentes

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

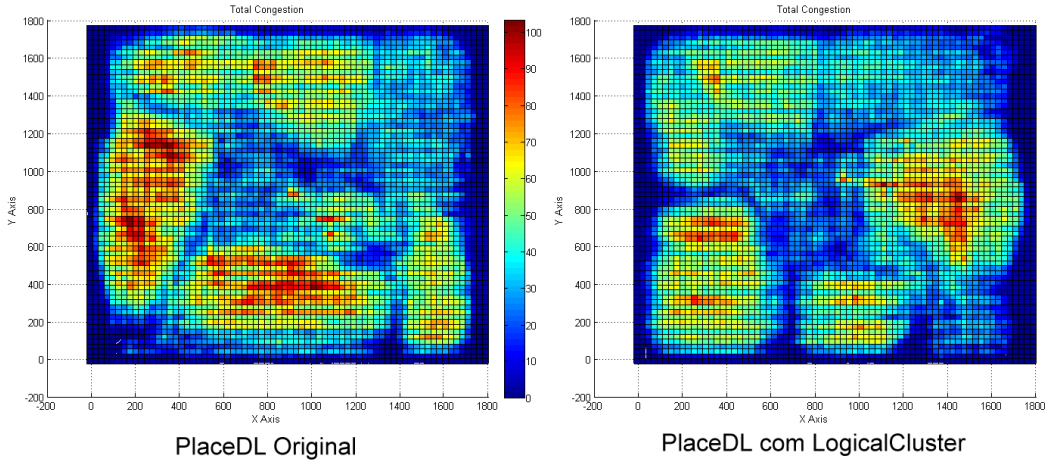

Figura 3.11: Mapa de congestionamento do ibm02 posicionado sem e com LogicalCluster (0.2)

3.3.5 Dificuldades Encontradas

Ferramentas acadêmicas nem sempre tem funcionamento padrão. E prever comportamentos de programas escritos por outros é bastante difícil, ainda mais em se tratando de ferramentas de CAD. Durante os testes, alguns fatos curiosos ocorreram.

No FastPlace3, se o circuito de entrada não possuir as células com nomes no estilo `letraNumero` (padrão de todos os *benchmarks*), a leitura do circuito pelo programa e a montagem das estruturas de dados é extremamente demorada (de milissegundos para dezenas de segundos). Esse comportamento estranho demorou a ser descoberto e quase fez com que a abordagem escolhida fosse abandonada. No entanto, bastou renomear os *clusters* de `cluster_1` para `c1` que o problema se resolveu.

Tentou-se realizar os mesmos testes em um posicionador com *Simulated Annealing*. Porém, as duas versões do Dragon (Dragon e Dragonmix) não aceitaram os circuitos clusterizados. No primeiro caso, a versão só aceita circuitos com altura padrão dos *benchmarks* ibm, enquanto que o Dragonmix, apesar de aceitar células de diferentes alturas e macroblocos, não completou nenhuma execução (*segmentation*

fault).

4 CONCLUSÃO

O objetivo deste trabalho foi implementar melhorias para o problema de posicionamento. Para se alcançar esse objetivo, na primeira parte do presente trabalho foi estudado o fluxo de posicionamento de circuitos integrados, investigando os algoritmos de posicionamento assim como detalhes específicos do problema. Também nesta etapa foi possível a familiarização com as metodologias de testes adotadas em trabalhos relacionados, assim como o estudo de linguagem de *script* (*shell script*) para a execução dos testes.

Após, iniciou-se a alteração dos códigos do posicionador PlaceDL para realizar a paralelização do algoritmo, conseguindo ganhos em tempo de execução de 4 vezes, na média.

Estudou-se também a técnica Logical Core a fim de utilizá-la como base para uma otimização de posicionamento. Decidiu-se por implementar um algoritmo de clusterização de circuitos que, além de reduzir o tamanho do problema e por consequência melhorar o tempo de execução, não provocasse perdas em termos de comprimentos de fio estimado. O algoritmo apresentou desempenho satisfatório para essa questão.

Por fim, procurou-se avaliar como as soluções propostas alteram as características de congestionamento de um circuito, apresentando um estudo comparativo das influências de diferentes metodologias de clusterização. Os resultados aqui também foram satisfatórios.

REFERÊNCIAS

ADYA, S. N.; MARKOV, I. L.; VILLARRUBIA, P. G. On Whitespace and Stability in Mixed-Size Placement and Physical Synthesis. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2003., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2003. p.311–. (ICCAD '03).

ALPERT, C. et al. A semi-persistent clustering technique for VLSI circuit placement. In: PHYSICAL DESIGN, 2005., New York, NY, USA. **Proceedings...** ACM, 2005. p.200–207. (ISPD '05).

ALPERT, C. J. et al. Quadratic placement revisited. In: DESIGN AUTOMATION CONFERENCE, 34., New York, NY, USA. **Proceedings...** ACM, 1997. p.752–757. (DAC '97).

BANERJEE, P.; JONES, M. H.; SARGENT, J. S. Parallel Simulated Annealing Algorithms for Cell Placement on Hypercube Multiprocessors. **IEEE Trans. Parallel Distrib. Syst.**, Piscataway, NJ, USA, v.1, n.1, p.91–106, 1990.

CALDWELL, A. E.; KAHNG, A. B.; MARKOV, I. L. Can recursive bisection alone produce routable placements? In: DAC '00: PROCEEDINGS OF THE 37TH ANNUAL DESIGN AUTOMATION CONFERENCE, New York, NY, USA. **Anais...** ACM, 2000. p.477–482.

CATANZARO, B.; KEUTZER, K.; SU, B.-Y. Parallelizing CAD: a timely research agenda for eda. In: DAC '08: PROCEEDINGS OF THE 45TH ANNUAL DESIGN AUTOMATION CONFERENCE, New York, NY, USA. **Anais...** ACM, 2008. p.12–17.

CHANDY, J. A.; BANERJEE, P. Parallel simulated annealing strategies for VLSI cell placement. In: VLSID '96: PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON VLSI DESIGN: VLSI IN MOBILE COMMUNICATION, Washington, DC, USA. **Anais...** IEEE Computer Society, 1996. p.37.

CHEN, T.-C. et al. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2006., New York, NY, USA. **Proceedings...** ACM, 2006. p.187–192. (ICCAD '06).

CONGESTIONMAPS Plotter. [Online; acessado em 13 de Novembro de 2010], `http://vlsicad.eecs.umich.edu/BK/PlaceUtils/`.

FIDUCCIA, C. M.; MATTHEYSES, R. M. A linear-time heuristic for improving network partitions. In: DAC '82: PROCEEDINGS OF THE 19TH DESIGN AUTOMATION CONFERENCE, Piscataway, NJ, USA. **Anais...** IEEE Press, 1982. p.175–181.

FLACH, G. et al. Cell placement on graphics processing units. In: SBCCI '07: PROCEEDINGS OF THE 20TH ANNUAL CONFERENCE ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, New York, NY, USA. **Anais...** ACM, 2007. p.87–92.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**: a guide to the theory of np-completeness. New York, NY, USA: W. H. Freeman & Co., 1979.

HENTSCHKE, R. F. **Algoritmos para Posicionamento de Celulas em Circuitos VLSI**. [Online; acessado em 16 de Novembro de 2010], `http://hdl.handle.net/10183/2598`.

HU, B.; MAREK-SADOWSKA, M. Fine granularity clustering for large scale placement problems. In: PHYSICAL DESIGN, 2003., New York, NY, USA. **Proceedings...** ACM, 2003. p.67–74. (ISPD '03).

ISPD - Routability-driven Placement Contest. [Online; acessado em 13 de Novembro de 2010], `http://www.ispd.cc/contest.html`.

KARYPIS, G. et al. Multilevel hypergraph partitioning: application in vlsi domain. In: DAC '97: PROCEEDINGS OF THE 34TH ANNUAL DESIGN AUTOMATION CONFERENCE, New York, NY, USA. **Anais...** ACM, 1997. p.526–529.

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

LIM, S. K. **Practical Problems in VLSI Physical Design Automation**. [S.l.]: Springer Publishing Company, Incorporated, 2008.

LIMA, C. et al. PlaceDL: a new technique to spread the cells on quadratic placement. In: XV WORKSHOP IBERCIHP VOL.1, Buenos Aires, AR. **Anais...** [S.l.: s.n.], 2009.

MOORE, G. E. Cramming More Components onto Integrated Circuits. **Electronics**, [S.l.], v.38, n.8, p.114–117, April 1965.

NAM, G.-J. ISPD 2006 Placement Contest: benchmark suite and results. In: ISPD '06: PROCEEDINGS OF THE 2006 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2006. p.167–167.

NAM, G.-J. et al. The ISPD2005 placement contest and benchmark suite. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2005. p.216–220.

OPENMP Version 3.0 Complete Specifications. [Online; acessado em 10 de Novembro de 2010], `http://www.openmp.org/mp-documents/spec30.pdf`.

PINTO, F. et al. Logical Core Algorithm: improving global placement. In: IEEE ANNUAL SYMPOSIUM ON VLSI, 2010., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2010. p.69–73. (ISVLSI '10).

ROY, J. A. et al. Capo: robust and scalable open-source min-cut floorplacer. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2005. p.224–226.

TAGHAVI, T. et al. Dragon2006: blockage-aware congestion-controlling mixed-size placer. In: ISPD '06: PROCEEDINGS OF THE 2006 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2006. p.209–211.

TAGHAVI, T.; YANG, X.; CHOI, B.-K. Dragon2005: large-scale mixed-size placement tool. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, New York, NY, USA. **Anais...** ACM, 2005. p.245–247.

VISWANATHAN, N.; CHU, C. ISPD04 IBM Standard Cell Benchmarks with Pads. In: Washington, DC, USA. **Anais...** IEEE Computer Society, 2004. p.135–140.

VISWANATHAN, N.; PAN, M.; CHU, C. FastPlace 3.0: a fast multilevel quadratic placement algorithm with placement congestion control. In: ASP-DAC '07: PROCEEDINGS OF THE 2007 ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, Washington, DC, USA. **Anais...** IEEE Computer Society, 2007. p.135–140.

WANG, M.; YANG, X.; SARRAFZADEH, M. Dragon2000: standard-cell placement tool for large industry circuits. In: ICCAD '00: PROCEEDINGS OF THE 2000 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, Piscataway, NJ, USA. **Anais...** IEEE Press, 2000. p.260–263.

WESTRA, J.; BARTELS, C.; GROENEVELD, P. Probabilistic congestion prediction. In: PHYSICAL DESIGN, 2004., New York, NY, USA. **Proceedings...** ACM, 2004. p.204–209. (ISPD '04).

ANEXO A TABELAS EXTRAS

Tabela A.1: HPWL - FastPlace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	FastPlace3_Original	Comp.	Comp.	Comp.
ibm01	1726949	1694436	1739968	1727729	1,00	0,98	1,01
ibm02	3567545	3572508	3598260	3598083	0,99	0,99	1,00
ibm03	4648088	4797531	4823746	4646751	1,00	1,03	1,04
ibm04	5648961	5673784	5717478	5732461	0,99	0,99	1,00
ibm05	10544016	10083068	9876540	9884011	1,07	1,02	1,00
ibm06	4937631	4952971	4972950	4936872	1,00	1,00	1,01
ibm07	8249225	8238413	8130475	8227714	1,00	1,00	0,99
ibm08	9104892	8997777	8953440	9048117	1,01	0,99	0,99
ibm09	9438459	9679914	9458114	9667507	0,98	1,00	0,98
ibm10	17757396	17381900	17417888	17268496	1,03	1,01	1,01
ibm11	13999328	13997840	14275271	14635573	0,96	0,96	0,98
ibm12	22391624	23037076	22499084	22838560	0,98	1,01	0,99
ibm13	16834828	16898164	17049844	16882824	1,00	1,00	1,01
ibm14	31258024	31298822	31635772	31896196	0,98	0,98	0,99
ibm15	39565748	38039968	38439764	38524492	1,03	0,99	1,00
ibm16	43355836	43036540	44473792	44243820	0,98	0,97	1,01
ibm17	61413428	61383128	63549392	59920616	1,02	1,02	1,06
ibm18	40459008	41666604	41314384	40676484	0,99	1,02	1,02
				Avg	1,000	0,999	1,003

Tabela A.2: Tempo Global - FastPlace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	FastPlace3_Original	Comp.	Comp.	Comp.
ibm01	5,18	5,03	4,66	4,79	1,08	1,05	0,97
ibm02	10,37	9,28	9,24	10,30	1,01	0,90	0,90
ibm03	12,50	8,96	9,18	11,38	1,10	0,79	0,81
ibm04	10,66	11,61	9,61	16,70	0,64	0,70	0,58
ibm05	13,26	16,63	12,21	15,58	0,85	1,07	0,78
ibm06	14,00	15,99	15,68	19,39	0,72	0,82	0,81
ibm07	21,83	21,80	18,26	20,37	1,07	1,07	0,90
ibm08	25,01	26,14	28,15	36,76	0,68	0,71	0,77
ibm09	29,60	24,82	22,95	27,01	1,10	0,92	0,85
ibm10	31,08	30,09	28,23	43,13	0,72	0,70	0,65
ibm11	29,61	28,13	27,09	43,17	0,69	0,65	0,63
ibm12	36,56	33,46	32,88	40,28	0,91	0,83	0,82
ibm13	61,72	39,95	45,65	64,10	0,96	0,62	0,71
ibm14	53,86	54,07	49,25	67,60	0,80	0,80	0,73
ibm15	69,89	76,39	85,67	106,22	0,66	0,72	0,81
ibm16	93,92	78,34	66,55	109,73	0,86	0,71	0,61
ibm17	84,02	78,69	74,81	96,83	0,87	0,81	0,77
ibm18	83,77	82,10	96,14	110,50	0,76	0,74	0,87
				Avg	0,854	0,794	0,795

Tabela A.3: Tempo Detalhado - FastPlace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	FastPlace3_Original	Comp.	Comp.	Comp.
ibm01	2,04	1,60	1,95	1,83	1,11	0,87	1,07
ibm02	3,94	4,32	3,83	3,81	1,03	1,13	1,01
ibm03	3,05	4,04	3,43	3,54	0,86	1,14	0,97
ibm04	3,98	3,94	4,89	4,18	0,95	0,94	1,17
ibm05	5,12	6,51	6,44	5,35	0,96	1,22	1,20
ibm06	5,67	5,40	5,32	4,60	1,23	1,17	1,16
ibm07	9,85	8,75	8,02	8,62	1,14	1,02	0,93
ibm08	13,83	11,83	12,38	13,09	1,06	0,90	0,95
ibm09	10,73	8,89	8,89	11,82	0,91	0,75	0,75
ibm10	18,13	15,07	14,82	14,91	1,22	1,01	0,99
ibm11	11,58	10,57	17,01	16,33	0,71	0,65	1,04
ibm12	21,51	17,90	20,03	21,74	0,99	0,82	0,92
ibm13	16,70	19,06	17,30	21,82	0,77	0,87	0,79
ibm14	30,66	32,48	35,30	33,21	0,92	0,98	1,06
ibm15	60,68	60,18	61,36	48,26	1,26	1,25	1,27
ibm16	42,38	70,40	71,61	64,43	0,66	1,09	1,11
ibm17	77,31	77,21	85,64	48,17	1,60	1,60	1,78
ibm18	53,46	67,94	72,62	57,94	0,92	1,17	1,25
				Avg	0,973	1,013	1,052

Tabela A.4: Tempo Total - FastPlace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	FastPlace3_Original	Comp.	Comp.	Comp.
ibm01	7,54	6,95	6,93	6,62	1,14	1,05	1,05
ibm02	14,53	13,82	13,29	14,11	1,03	0,98	0,94
ibm03	15,68	13,13	12,74	14,92	1,05	0,88	0,85
ibm04	15,33	16,24	15,19	20,88	0,73	0,78	0,73
ibm05	18,70	23,46	18,97	20,93	0,89	1,12	0,91
ibm06	19,86	21,58	21,19	23,99	0,83	0,90	0,88
ibm07	32,10	30,97	26,70	28,99	1,11	1,07	0,92
ibm08	39,71	38,84	41,40	49,85	0,80	0,78	0,83
ibm09	41,04	34,42	32,55	38,83	1,06	0,89	0,84
ibm10	50,41	46,36	44,25	58,04	0,87	0,80	0,76
ibm11	42,05	39,56	44,96	59,50	0,71	0,66	0,76
ibm12	58,84	52,13	53,68	62,02	0,95	0,84	0,87
ibm13	79,48	60,07	64,01	85,92	0,93	0,70	0,74
ibm14	87,75	89,78	87,78	100,81	0,87	0,89	0,87
ibm15	133,00	139,00	149,46	154,48	0,86	0,90	0,97
ibm16	138,56	151,00	140,42	174,16	0,80	0,87	0,81
ibm17	164,61	159,18	163,73	145,00	1,14	1,10	1,13
ibm18	144,59	157,40	176,12	168,44	0,86	0,93	1,05
				Avg	0,923	0,896	0,883

Tabela A.5: HPWL - NTUplace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	NTUplace3_Original	Comp.	Comp.	Comp.
ibm01	1694796	1713023	1726668	1685143	1,01	1,02	1,02
ibm02	3572900	3528452	3596297	3526805	1,01	1,00	1,02
ibm03	4671101	4727691	4745987	4733598	0,99	1,00	1,00
ibm04	5876718	5796466	5838221	5775934	1,02	1,00	1,01
ibm05	9962724	10053264	9764116	9944742	1,00	1,01	0,98
ibm06	4944536	4969520	5029230	5008886	0,99	0,99	1,00
ibm07	8439768	8191474	8309610	8358811	1,01	0,98	0,99
ibm08	9102122	9074438	9094857	9189443	0,99	0,99	0,99
ibm09	9461713	9651777	9475370	9421964	1,00	1,02	1,01
ibm10	17369397	17282821	17693412	17381894	1,00	0,99	1,02
ibm11	14160855	14106991	14279729	13946979	1,02	1,01	1,02
ibm12	21792276	21959736	22526387	22228242	0,98	0,99	1,01
ibm13	16792977	17040077	16788707	16772447	1,00	1,02	1,00
ibm14	31287303	31509206	31514439	31571929	0,99	1,00	1,00
ibm15	38425494	38383275	39483535	38579492	1,00	0,99	1,02
ibm16	41484025	45321138	45679230	43237931	0,96	1,05	1,06
ibm17	62138784	64263396	63348885	63003317	0,99	1,02	1,01
ibm18	42169308	41352916	40619453	43298326	0,97	0,96	0,94
				Avg	0,996	1,002	1,006

Tabela A.6: Tempo Global - NTUplace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	NTUplace3_Original	Comp.	Comp.	Comp.
ibm01	19,00	22,00	19,00	28,00	0,68	0,79	0,68
ibm02	36,00	38,00	40,00	44,00	0,82	0,86	0,91
ibm03	58,00	58,00	55,00	70,00	0,83	0,83	0,79
ibm04	77,00	71,00	65,00	101,00	0,76	0,70	0,64
ibm05	82,00	74,00	69,00	118,00	0,69	0,63	0,58
ibm06	93	80,00	65,00	106,00	0,88	0,75	0,61
ibm07	157,00	131,00	155,00	221,00	0,71	0,59	0,70
ibm08	156,00	147,00	136,00	194,00	0,80	0,76	0,70
ibm09	207,00	236,00	202,00	240,00	0,86	0,98	0,84
ibm10	317,00	281,00	281,00	386,00	0,82	0,73	0,73
ibm11	314,00	331,00	292,00	331,00	0,95	1,00	0,88
ibm12	373,00	343,00	289,00	457,00	0,82	0,75	0,63
ibm13	360,00	416,00	419,00	493,00	0,73	0,84	0,85
ibm14	1061,00	1049,00	793,00	1125,00	0,94	0,93	0,70
ibm15	1458,00	1311,00	1456,00	1929,00	0,76	0,68	0,75
ibm16	2421,00	2183,00	1795,00	2491,00	0,97	0,88	0,72
ibm17	2557,00	2215,00	2013,00	2605,00	0,98	0,85	0,77
ibm18	2593,00	2680,00	2169,00	2144,00	1,21	1,25	1,01
				Avg	0,820	0,807	0,724

Tabela A.7: Tempo Detalhado - NTUplace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	NTUplace3_Original	Comp.	Comp.	Comp.
ibm01	4,00	4,00	4,00	2,00	2,00	2,00	2,00
ibm02	7,00	7,00	7,00	4,00	1,75	1,75	1,75
ibm03	8,00	8,00	8,00	4,00	2,00	2,00	2,00
ibm04	9,00	9,00	9,00	5,00	1,80	1,80	1,80
ibm05	11,00	11,00	11,00	6,00	1,83	1,83	1,83
ibm06	11,00	11,00	11,00	6,00	1,83	1,83	1,83
ibm07	18,00	15,00	18,00	7,00	2,57	2,14	2,57
ibm08	20,00	18,00	17,00	11,00	1,82	1,64	1,55
ibm09	18,00	19,00	21,00	11,00	1,64	1,73	1,91
ibm10	25,00	26,00	27,00	14,00	1,79	1,86	1,93
ibm11	25,00	26,00	27,00	14,00	1,79	1,86	1,93
ibm12	25,00	28,00	28,00	14,00	1,79	2,00	2,00
ibm13	31,00	29,00	32,00	15,00	2,07	1,93	2,13
ibm14	52,00	57,00	53,00	26,00	2,00	2,19	2,04
ibm15	79,00	76,00	83,00	31,00	2,55	2,45	2,68
ibm16	85,00	100,00	96,00	44,00	1,93	2,27	2,18
ibm17	88,00	110,00	102,00	44,00	2,00	2,50	2,32
ibm18	114,00	115,00	93,00	47,00	2,43	2,45	1,98
				Avg	1,883	1,967	1,989

Tabela A.8: Tempo Total - NTUplace3: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	NTUplace3_Original	Comp.	Comp.	Comp.
ibm01	23,32	26,32	23,32	30,00	0,78	0,88	0,78
ibm02	43,22	45,22	47,22	48,00	0,90	0,94	0,98
ibm03	66,13	66,13	63,13	74,00	0,89	0,89	0,85
ibm04	86,69	80,69	74,69	106,00	0,82	0,76	0,70
ibm05	93,32	85,32	80,32	124,00	0,75	0,69	0,65
ibm06	104,19	91,19	76,19	112,00	0,93	0,81	0,68
ibm07	175,42	146,42	173,42	228,00	0,77	0,64	0,76
ibm08	176,87	165,87	153,87	205,00	0,86	0,81	0,75
ibm09	225,71	255,71	223,71	251,00	0,90	1,02	0,89
ibm10	343,20	308,20	309,20	400,00	0,86	0,77	0,77
ibm11	339,86	357,86	319,86	345,00	0,99	1,04	0,93
ibm12	398,77	371,77	317,77	471,00	0,85	0,79	0,67
ibm13	392,06	446,06	452,06	508,00	0,77	0,88	0,89
ibm14	1116,23	1109,23	849,23	1151,00	0,97	0,96	0,74
ibm15	1539,43	1389,43	1541,43	1960,00	0,79	0,71	0,79
ibm16	2508,26	2285,26	1893,26	2535,00	0,99	0,90	0,75
ibm17	2648,28	2328,28	2118,28	2649,00	1,00	0,88	0,80
ibm18	2714,36	2802,36	2269,36	2191,00	1,24	1,28	1,04
				Avg	0,892	0,870	0,801

Tabela A.9: HPWL - PlaceDL: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	PlaceDL_Original	Comp.	Comp.	Comp.
ibm01	1764753	1734173	1749059	1746255	1,011	0,993	1,002
ibm02	3862712	3808365	3980247	3913983	0,987	0,973	1,017
ibm03	5034337	5077645	5201356	5050648	0,997	1,005	1,030
ibm04	6020084	6051629	6479531	6054543	0,994	1,000	1,070
ibm05	11242046	12210236	11898600	10254738	1,096	1,191	1,160
ibm06	5133639	5155088	5056685	5103165	1,006	1,010	0,991
ibm07	8804515	9141956	9031761	8798971	1,001	1,039	1,026
ibm08	10173209	10091783	9337715	10125293	1,005	0,997	0,922
ibm09	10677050	10181235	10991480	10577640	1,009	0,963	1,039
ibm10	18012060	18206112	18418604	18307868	0,984	0,994	1,006
ibm11	15227280	15451028	15431112	15571247	0,978	0,992	0,991
ibm12	24503648	24950332	25567860	24440860	1,003	1,021	1,046
ibm13	18826520	19033096	19303648	18609872	1,012	1,023	1,037
ibm14	35642476	35706136	34140224	35715840	0,998	1,000	0,956
ibm15	43256504	43996372	43860160	42748476	1,012	1,029	1,026
ibm16	46247984	46629400	46620472	45005876	1,028	1,036	1,036
ibm17	66975216	70698544	70010544	68380320	0,979	1,034	1,024
ibm18	49102124	48144260	45428460	45699712	1,074	1,053	0,994
				Avg	1,010	1,020	1,021

Tabela A.10: Tempo Global - PlaceDL: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	PlaceDL_Original	Comp.	Comp.	Comp.
ibm01	1,61	1,67	1,62	1,69	0,95	0,99	0,96
ibm02	3,86	4,38	3,39	5,20	0,74	0,84	0,65
ibm03	2,67	2,54	2,13	2,67	1,00	0,95	0,80
ibm04	4,91	4,35	5,20	4,32	1,14	1,01	1,21
ibm05	2,71	1,67	1,91	2,85	0,95	0,59	0,67
ibm06	3,80	3,90	3,76	3,78	1,01	1,03	0,99
ibm07	6,10	6,27	5,49	6,34	0,96	0,99	0,87
ibm08	5,59	5,66	5,98	5,85	0,95	0,97	1,02
ibm09	9,67	10,62	9,99	10,20	0,95	1,04	0,98
ibm10	14,18	15,15	13,26	13,21	1,07	1,15	1,00
ibm11	12,68	12,24	11,90	12,99	0,98	0,94	0,92
ibm12	11,11	11,55	11,00	11,36	0,98	1,02	0,97
ibm13	22,47	22,40	20,18	21,37	1,05	1,05	0,94
ibm14	29,01	27,51	27,09	28,87	1,00	0,95	0,94
ibm15	30,23	34,05	29,85	34,52	0,88	0,99	0,86
ibm16	45,03	46,34	42,52	48,03	0,94	0,97	0,89
ibm17	54,48	47,66	47,53	65,21	0,84	0,73	0,73
ibm18	39,51	44,52	50,40	49,02	0,81	0,91	1,03
				Avg	0,955	0,950	0,913

Tabela A.11: Tempo Detalhado - PlaceDL: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	PlaceDL_Original	Comp.	Comp.	Comp.
ibm01	1,8	2,03	1,74	1,78	1,03	1,14	0,98
ibm02	5,90	5,80	6,14	6,49	0,91	0,89	0,95
ibm03	4,32	3,94	4,06	3,59	1,20	1,10	1,13
ibm04	4,31	4,71	4,85	4,86	0,89	0,97	1,00
ibm05	7,99	11,01	11,64	6,73	1,19	1,64	1,73
ibm06	5,41	5,59	5,31	6,20	0,87	0,90	0,86
ibm07	7,32	7,93	7,89	7,75	0,94	1,02	1,02
ibm08	11,50	12,1	12,69	11,46	1,00	1,05	1,11
ibm09	10,80	10,43	10,62	9,68	1,12	1,08	1,10
ibm10	14,06	15,35	16,54	14,81	0,95	1,04	1,12
ibm11	14,03	11,6	15,44	12,88	1,09	0,90	1,20
ibm12	19,88	18,28	21,44	18,04	1,10	1,01	1,19
ibm13	18,78	19,75	20,40	19,14	0,98	1,03	1,07
ibm14	30,05	30,59	34,48	31,68	0,95	0,97	1,09
ibm15	39,75	42,25	46,15	44,54	0,89	0,95	1,04
ibm16	45,05	43,47	49,90	43,29	1,04	1,00	1,15
ibm17	56,66	59,06	62,81	53,70	1,06	1,10	1,17
ibm18	55,37	57,72	51,23	44,98	1,23	1,28	1,14
				Avg	1,025	1,060	1,112

Tabela A.12: Tempo Total - PlaceDL: valores absolutos e comparativos para cada valor de limite

CKT / %	0,1	0,2	0,3	PlaceDL_Original	Comp.	Comp.	Comp.
ibm01	3,77	4,02	3,68	3,47	1,087	1,157	1,061
ibm02	9,98	10,40	9,75	11,69	0,854	0,889	0,834
ibm03	7,12	6,61	6,32	6,26	1,137	1,056	1,009
ibm04	9,91	9,75	10,74	9,18	1,080	1,062	1,171
ibm05	11,02	13,00	13,87	9,58	1,151	1,358	1,448
ibm06	9,40	9,68	9,26	9,98	0,942	0,970	0,928
ibm07	13,84	14,62	13,80	14,09	0,983	1,038	0,980
ibm08	17,96	18,59	19,54	17,31	1,037	1,073	1,129
ibm09	21,18	21,76	21,32	19,88	1,065	1,095	1,073
ibm10	29,44	31,70	31,00	28,02	1,051	1,132	1,106
ibm11	27,57	24,71	28,20	25,87	1,066	0,955	1,090
ibm12	31,76	30,60	33,21	29,40	1,080	1,041	1,129
ibm13	42,31	43,21	41,64	40,51	1,044	1,067	1,028
ibm14	62,29	61,33	64,80	60,55	1,029	1,013	1,070
ibm15	72,41	78,73	78,43	79,06	0,916	0,996	0,992
ibm16	92,34	92,07	94,68	91,32	1,011	1,008	1,037
ibm17	114,42	110,00	113,62	118,91	0,962	0,925	0,956
ibm18	102,24	109,60	108,99	94,00	1,088	1,166	1,159
				Avg	1,032	1,056	1,067

Tabela A.13: BestChoice - Congestionamento: valores

ckt	ibm01	ibm02	ibm03	ibm05	ibm06	ibm07
0-10	13,9735	17,5453	19,1095	14,5214	14,1514	13,6293
10 – 20	6,89443	11,4286	18,2324	6,66577	9,1453	16,6234
20 – 30	16,0665	15,3521	29,7521	9,07247	16,7521	24,5327
30 – 40	22,3146	15,2113	18,165	15,5489	27,4237	19,6262
40 – 50	19,5445	14,326	9,98482	16,6577	21,1966	12,8418
50 – 60	10,9264	12,4748	4,11537	15,0081	8,62027	6,65455
60 – 70	6,83287	8,61167	0,522854	12,8718	2,22222	3,64313
70 – 80	2,49307	3,8833	0,033733	7,12547	0,37851	1,59225
80 – 90	0,831025	0,985915	0,067465	2,2715	0,07326	0,597092
90 – 100	0,123115	0,181087	0,016866	0,256896	0,03663	0,259605
Σ > 50% (%)	21,21	26,14	4,76	37,53	11,33	12,75

ckt	ibm09	ibm10	ibm11	ibm13	ibm17	ibm18
0-10	13,2878	17,2291	13,9378	10,94	8,56481	10,3669
10 – 20	20,058	39,5546	22,5339	15,9471	8,9399	22,3653
20 – 30	34,5051	30,3776	30,3838	29,5749	17,8048	32,0838
30 – 40	21,9605	10,2904	20,5645	26,0132	24,5285	20,662
40 – 50	7,86266	1,91116	8,57367	12,5508	22,1536	8,5219
50 – 60	1,90993	0,418962	3,06363	4,01039	12,0563	3,72408
60 – 70	0,371581	0,137741	0,813601	0,859707	4,6232	1,80344
70 – 80	0,029727	0,057392	0,084166	0,08975	1,1274	0,406435
80 – 90	0,007432	0,011478	0,022444	0,004724	0,158608	0,054822
90 – 100	0,007432	0,011478	0,022444	0,009447	0,042867	0,011342
Σ > 50% (%)	2,33	0,64	4,01	4,97	18,01	6,00

Tabela A.14: LogicalCluster Limite 0.1- Congestionamento: valores

ckt	ibm01	ibm02	ibm03	ibm05	ibm06	ibm07
0-10	13,60	16,18	22,18	14,47	13,76	16,14
10 – 20	6,83	9,88	27,90	5,39	10,12	27,95
20 – 30	19,67	17,93	31,66	6,84	21,17	28,86
30 – 40	25,82	17,48	13,61	14,08	28,30	15,54
40 – 50	17,24	13,46	3,76	20,29	19,28	7,17
50 – 60	9,23	11,45	0,76	18,54	6,03	3,48
60 – 70	4,46	8,27	0,08	11,76	1,21	0,68
70 – 80	2,15	3,70	0,03	6,22	0,10	0,15
80 – 90	0,77	1,43	0,00	2,14	0,01	0,02
90 – 100	0,22	0,22	0,02	0,27	0,01	0,02
Σ > 50% (%)	16,84	25,07	0,89	38,93	7,36	4,34

ckt	ibm09	ibm10	ibm11	ibm13	ibm17	ibm18
0-10	14,14	14,12	12,99	10,25	8,09	12,53
10 – 20	22,64	31,36	18,92	12,64	9,53	26,65
20 – 30	39,47	32,52	33,09	27,09	17,36	27,75
30 – 40	19,26	15,74	23,56	27,30	23,04	18,22
40 – 50	3,86	5,03	8,89	14,51	21,65	8,47
50 – 60	0,57	0,95	2,16	5,72	12,67	3,81
60 – 70	0,05	0,20	0,29	1,93	5,23	1,67
70 – 80	0,01	0,06	0,07	0,47	1,83	0,60
80 – 90	0,00	0,01	0,01	0,09	0,55	0,25
90 – 100	0,01	0,01	0,01	0,02	0,06	0,06
Σ > 50% (%)	0,64	1,23	2,54	8,22	20,34	6,38

Tabela A.15: LogicalCluster Limite 0.2- Congestionamento: valores

ckt	ibm01	ibm02	ibm03	ibm05	ibm06	ibm07
0-10	13,54	18,09	16,87	15,89	13,58	15,04
10 – 20	6,71	11,43	16,16	7,60	8,30	21,28
20 – 30	15,60	17,89	26,60	10,71	15,26	23,90
30 – 40	23,42	18,63	21,69	15,60	25,71	18,60
40 – 50	19,58	14,95	11,64	16,35	22,64	11,18
50 – 60	11,39	10,87	5,18	14,49	10,83	6,10
60 – 70	5,79	5,05	1,25	11,36	2,86	2,73
70 – 80	2,86	2,43	0,40	6,02	0,74	0,98
80 – 90	0,98	0,60	0,19	1,84	0,06	0,16
90 – 100	0,12	0,06	0,03	0,15	0,01	0,03
Σ > 50% (%)	21,14	19,01	7,05	33,86	14,51	10,00

ckt	ibm09	ibm10	ibm11	ibm13	ibm17	ibm18
0-10	12,54	13,27	13,63	10,36	8,65	11,48
10 – 20	14,92	28,98	25,56	13,07	12,80	20,68
20 – 30	29,96	33,05	33,27	28,04	24,54	25,28
30 – 40	25,54	17,17	19,57	26,78	26,74	21,54
40 – 50	12,34	5,72	5,95	14,57	16,41	11,82
50 – 60	3,72	1,46	1,62	5,60	7,32	5,62
60 – 70	0,86	0,24	0,21	1,39	2,43	2,47
70 – 80	0,08	0,07	0,08	0,15	0,77	0,85
80 – 90	0,02	0,02	0,06	0,03	0,29	0,22
90 – 100	0,03	0,02	0,05	0,01	0,06	0,03
Σ > 50% (%)	4,72	1,81	2,01	7,18	10,86	9,19

Tabela A.16: LogicalCluster Limite 0.3- Congestionamento: valores

ckt	ibm01	ibm02	ibm03	ibm05	ibm06	ibm07
0-10	13,54	18,43	19,43	15,22	13,76	15,78
10 – 20	6,46	12,92	21,29	7,57	8,08	23,95
20 – 30	16,53	17,73	31,07	11,80	15,56	26,59
30 – 40	23,45	16,50	17,44	17,44	26,01	19,20
40 – 50	20,68	14,97	7,99	14,01	22,71	9,28
50 – 60	11,85	10,46	2,36	14,32	10,42	3,47
60 – 70	4,68	5,79	0,37	11,53	2,69	1,25
70 – 80	2,15	2,70	0,03	5,60	0,68	0,37
80 – 90	0,52	0,48	0,00	2,26	0,07	0,09
90 – 100	0,12	0,02	0,02	0,24	0,02	0,02
Σ > 50% (%)	19,33	19,46	2,78	33,95	13,88	5,20

ckt	ibm09	ibm10	ibm11	ibm13	ibm17	ibm18
0-10	11,07	14,47	15,10	9,68	8,25	12,34
10 – 20	13,51	32,71	24,08	10,58	9,80	29,00
20 – 30	28,27	31,91	32,22	24,27	19,75	27,51
30 – 40	27,10	15,25	19,00	27,90	27,17	17,53
40 – 50	15,29	4,60	7,61	17,76	20,94	8,13
50 – 60	4,03	0,84	1,73	7,25	9,48	3,76
60 – 70	0,60	0,13	0,16	2,15	3,48	1,41
70 – 80	0,07	0,06	0,04	0,32	0,95	0,25
80 – 90	0,03	0,01	0,02	0,08	0,17	0,05
90 – 100	0,03	0,01	0,03	0,01	0,01	0,02
Σ > 50% (%)	4,76	1,05	1,98	9,81	14,09	5,49

ANEXO B SCRIPTS

Neste apêndice, para fins de consulta, estarão listados alguns dos *shell scripts* utilizados para execução dos testes apresentados neste trabalho.

Exemplo de clusterização e execução do posicionamento no FastPlace3

```
./logicalcore_paralell ibm01.aux 16 10e-05
calcula a Logical Core com 16 threads e erro 10e-05 e gera o arquivo ibm01.prob

./logicalcluster ibm01 -prob ibm01.prob -perc 0.1
gera o arquivo de cluster: ibm01_0.1_low.cluster

./khcluster ibm01.aux ibm01_0.1_low.cluster testeIBM01_01
gera um circuito (clusterizado) de nome testeIBM01_05 e testeIBM01_01.uncluster
(arquivo de referência para desclusterizar após posicionamento global)

./FastPlace3.0_Linux32_GP . testeIBM01_01.aux .
gera um arquivo de posicionamento global chamado testeIBM01_01_FP_gp.pl

./khcluster testeIBM01_05.aux testeIBM01_05.uncluster ibm01_low_01_FP_GP -uncluster -pl testeIBM01_01_FP_gp.pl
desclusteriza o circuito posicionado globalmente:
gera um arquivo ibm01_low_01_FP_GP.pl com o posicionamento global e nomes reais das células

./FastPlace3.0_Linux32_DP -legalize . ibm01.aux . ibm01_low_01_FP_GP.pl
gera um arquivo ibm01_low_01_FP_dp.pl com o posicionamento final
```

Procedimento de testes do NTUplace3-LE

```
NTUplace3
for ckt in ibm01 ibm02 ibm03 ibm04 ibm05 ibm06 ibm07 ibm08 ibm09 ibm10 ibm11 ibm12 ibm13 ibm14 ibm15 ibm16 ibm17 ibm18;
do
cp NTUplace3-Lpnorm ${ckt}
cd ${ckt}
for perc in 0.1 0.2 0.3;
do
./NTUplace3-Lpnorm -aux c${ckt}_${perc}.aux -nolegal -nodetail >> ../NTU_gp.txt
mv c${ckt}_${perc}.gp.pl c${ckt}_${perc}_NTU.gp.pl
./khcluster c${ckt}_${perc}.aux c${ckt}_${perc}.uncluster uc${ckt}_${perc}_NTU_GP  -uncluster -pl ...
c${ckt}_${perc}_NTU.gp.pl
./NTUplace3-Lpnorm -aux ${ckt}.aux -loadpl uc${ckt}_${perc}_NTU_GP.pl -noglobal >> ../NTU_dp.txt
mv uc${ckt}_${perc}_NTU_GP.ntup.pl ${ckt}_uc_${perc}_NTU.dp.pl
  done;
cd ../
done;
```

Procedimento de testes do PlaceDL

```
for ckt in ibm01 ibm02 ibm03 ibm04 ibm05 ibm06 ibm07 ibm08 ibm09 ibm10 ibm11 ibm12 ibm13 ibm14 ibm15 ibm16 ibm17 ibm18;
do
cd ${ckt}
for perc in 0.1 0.2 0.3;
do
./PlaceDL-32 c${ckt}_${perc}.aux >> ../PlaceDL_gp.txt
./khcluster c${ckt}_${perc}.aux c${ckt}_${perc}.uncluster uc${ckt}_${perc}_PlaceDL_gp -uncluster -pl ...
c${ckt}_${perc}_PlaceDL_gp.pl
```

```
./FastPlace3.0_Linux32_DP -legalize . ${ckt}.aux . uc${ckt}_${perc}_PlaceDL_gp.pl >> ../PlaceDL_FP_dp.txt
mv ${ckt}_FP_dp.pl ${ckt}_uc_${perc}_PlaceDL_FP_dp.pl
  done;
cd ../
done;
```