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

SANDRO SAWICKI

Particionamento de Células e Pads de I/O em Circuitos VLSI 3D

Tese de Doutorado

Prof. Dr. Ricardo Augusto da Luz Reis
Orientador

Porto Alegre, dezembro de 2009.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Sawicki, Sandro

Particionamento de Células e Pads de I/O em Circuitos VLSI 3D/ Sandro Sawicki – Porto Alegre: Programa de Pós-Graduação em Computação, 2009.

150 f.:il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2009. Orientador: Ricardo Augusto da Luz Reis.

1.Circuitos VLSI 3D. 2. Particionamento 3.Pads de I/O. I. Reis, Ricardo Augusto da Luz. II. Particionamento de Células e Pads de I/O em Circuitos VLSI 3D.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. Carlos Alexandre Netto
Vice-Reitor: Prof. Rui Vicente Oppermann
Pró-Reitor de Pós-Graduação: Prof. Aldo Bolten Lucion
Diretor do Instituto de Informática: Prof. Flávio Rech Wagner
Coordenador do PPGC: Prof. Álvaro Freitas Moreira
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Alguém já me dizia que a vida é muito parecida com uma estação rodoviária, uns chegam, outros se vão. Que estamos aqui somente de passagem. Analogamente falando, durante o período desta tese, muitas pessoas queridas se foram, por outro lado, outras chegaram. Muitas coisas aconteceram, muitas descobertas, muitos amigos, muitas dúvidas, muitos momentos de solidão, muita reflexão.

Um agradecimento imenso a Deus, o grande arquiteto, por me dar a possibilidade de estar aqui nessa "rodoviária"; de me destinar uma família tão querida como é a minha. Ao meu pai Mário, minha mãe Nelda, que me ensinaram muitas coisas, dentre elas, de ver sempre o lado bom das pessoas, de ter caráter, honestidade e falar sempre a verdade. Aos meus avós, Olinda (*in memoriam*) e Reinholdo (*in memoriam*) que, apesar da baixa escolaridade, foram fundamentais na minha formação pessoal. Ao meu irmão Jackson que, apesar da distância, sempre torceu pela minha felicidade.

Um agradecimento especial à minha esposa Bianca, uma pessoa maravilhosa que encontrei neste período. Sempre ao meu lado, fundamental, superando as distâncias e a saudade. Obrigado Bibica! Ah, e também a minha filha Manuela que está chegando!

A minha nova família: Vera Lúcia (sogra) e o Jorginho, Elmário Prediger (sogro), Odete e Victória, ao meu grande amigo Luiz Alberto Fior, minha cunhada Priscila (Piti) e minha afiliada Luiza.

Ao meu orientador Professor Ricardo Reis um grande amigo que admiro muito. Atual compadre. Um grande líder, uma pessoa a frente de seu tempo. Obrigado por me proporcionar a oportunidade de conviver com o GME da UFRGS ao qual conheci grandes amigos e obtive inúmeros aprendizados. Um imenso abraço para toda a sua família.

Um agradecimento especial também ao meu co-orientador, Prof. Marcelo Johann, um amigo que admiro bastante, muito competente, sempre cheio de idéias e, além de todas essas virtudes, um excelente músico.

Agradeço ao meu ex-vizinho, colega, amigo e conselheiro Renato Hentschke, uma pessoa simplesmente brilhante. Parceiro dos jogos de vôlei, das discussões técnicas e não-técnicas, dos jogos de futebol, das caronas, dos churrascos e carreteiros de "quase" todas as noites. Obrigado Dr. Mr. Hentschke.

Ao amigo Gustavo Wilke, um agradecimento especial pelas ponderações acerca da vida e do trabalho. Sempre pronto para ajudar, sereno, camarada, atencioso, uma pessoa muito agradável para se conviver. Obrigadão Wilkão!

Um imenso agradecimento ao meu grande amigo e irmão Odaylson Eder. Não tenho palavras para descrever o quanto lhe sou grato. Se estou aqui hoje, reconheço que foi graças ao teu apoio. Muito obrigado de coração.

Aos colegas e professores: Gustavo Neuberger, Guilherme Flach, Felipe Pinto, Adriel, Lisane Brisolara, Jane, Caco, Julius, Glauco, ao casal de afiliados Lincoln e Duda, Gerson Battisti e Iara, Lucas Brusamarelo, José Almada Güntzel, Luciano Agostini, Cesar Zeferino, Renato Ribas, André Reis, Luciana Buriol, Leandro Indrusiak, Luigi Carro, Érika Cota, Altamiro Susin, Marcelo Lubaszewski, Gilson Wirth, Flávio Wagner, Fernanda Lima Kastensmidt , Sérgio Bambi.

Um agradecimento as seguintes pessoas que acompanharam a trajetória deste trabalho: Adroaldo Azambuja, Miroslava, Eduardo e Amanda, Adalberto Hommerding e Cleide Maria Krein, Luiz Francisco Franco, Márcia, Frederico e Vitória, Bruno, José Flávio de Souza Inajá, Flavinha e Ricardo, Auri Copetti Júnior, Adriane e Álvaro, Oda, Naira e Isa, Bira, Kéka e Pedro Augusto, Jackson Quanaco De Ley e Branca, Adil Baú e Lili, Roberto Laux, Janor L. Duarte, Melissa Marchi Juchen, Jucelito, Pati e Arthur, Lúcio Roseira Braga, Ronald Reisdorfer, Roberto Zanoni e Leonardo, Rafael Zanoni e Thaize, Fábio Queruz, Valquíria, Daniel e Álvaro, Paulo Engler, Altair, Arara (Gilmar), Talvane Engroff, Cassíus Frosi Lenzi, Dimas, Andressa, Elias e Romilda, Jores Luis Gnatta e Sonéli, Marcos Fachinello e Gisa, Alexandre Schüller Xilão e Michele, Giancalo Schuh e Melissa, Tiago Denardin, Cícero (vermeio), Vivian e Lorenzo, Charles Carazzo e Natasha, Romeu – o motivador, Sávio, Ronaldo Arbo, Bruno, seu Chico, Roberto Laux, Mari e Clóbis.

Aos meus amigos de Hannover: Stefan, Darius, Ole, Peter, Thomas, Amir, Philipp, Marcus, Min, Daniel, Neele, Frau Stoll e Prof. Barke, obrigado pelo convívio agradável junto ao IMS – *Institute of Microelectronics and Systems*. Vielen Danke!

Sumário

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

EDA	Electronic Design Automation
CAD	Computer Aided Design
SOI	Silicon On Insulator
VLSI	Very Large Scale Integration
TSV	Through-Silicon Via

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A etapa de particionamento em circuitos VLSI 3D é fundamental na distribuição de células e blocos para as camadas do circuito, além de auxiliar na redução da complexidade dos posicionadores. Estes, quando o particionamento é bem realizado, permitem que se atinjam soluções com menor comprimento total de fios, o que reduz a dissipação de potência e aumenta o desempenho dos circuitos. Atualmente, os algoritmos utilizados para resolver o problema de particionamento em circuitos 3D são adaptações daqueles aplicados em circuitos planares. Ou seja, o circuito é particionado como se fosse um hipergrafo tradicional, e as células são assinaladas diretamente para as partições, com o objetivo de reduzir somente as conexões que cruzam as partes. Contudo essa solução é simplista e faz com que o algoritmo não perceba a criação de conexões longas, o que aumenta o número de vias do circuito e, consequentemente, sua área. É importante compreender que o valor dos recursos usados é um múltiplo da distância vertical das camadas. Na verdade, considerando-se que o caminho de uma camada para outra adjacente atravessa todos os níveis de metal, é evidente que qualquer ligação vertical superior à adjacente pode ser proporcionalmente mais custosa para o roteamento, sem mencionar o atraso provocado e o quanto da área ativa é ocupada. Em vista disso, este trabalho apresenta um conjunto de algoritmos desenvolvidos para reduzir o número de vias em circuitos VLSI 3D. A otimização é obtida pelo uso de duas estratégias distintas: a análise prévia da estrutura interna do circuito e a redução do número de conexões verticais não-adjacentes. Os algoritmos propostos, além de reduzir o número de vias-3D, adaptam a lógica dos circuitos 2D para os 3D mantendo o balanceamento de área e dos pinos de I/O entre as diferentes camadas. Os resultados experimentais mostram que essas técnicas reduzem o número total de vias-3D em 19%, 18%, 12% e 16% em duas, três, quatro e cinco *tiers,* respectivamente, comparados com os resultados das abordagens atuais.

Palavras-chave: Circuitos VLSI 3D, particionamento, posicionamento, pinos de I/O, *pads* de I/O.

Cells and I/O Pads Partitioning Targeting 3D VLSI Integrated Circuits

ABSTRACT

A 3D circuit is the stacking of regular 2D circuits. The advances on the fabrication and packaging technologies allow interconnection of stacked 2D circuits. However, 3D-vias can impose significant obstacles and constraints to the 3D placement problem. Most of the existing placement and partitioning algorithms completely ignore this fact, but they do optimize the number of vias using a min-cut partitioning applied to a generic graph partitioning problem. This work proposes a new approach for I/O pads and cells partitioning addressing 3D-vias reduction and its impact on the 3D circuit design. The approach presents two distinct strategies: the first one is based on circuit structure analyses and the second one reducing the number of connections between non-adjacent tiers. The strategies outperformed a state-of-the-art hypergraph partitioner, hMetis and other approaches by providing a reduction of the number of 3D-vias 19%, 17%, 12% and 16% using two, three, four and five tiers.

Key-words: 3D VLSI Integrated Circuits, Partitioning, Placement, I/O pads, CAD

1 INTRODUÇÃO

A indústria de semicondutores demanda por ferramentas de CAD capazes de tratar sistemas cada vez mais complexos. O desafio é processar, em tempo aceitável, circuitos com centenas de milhões de transistores, com os problemas intrínsecos à grande integração e à tecnologia submicrônica. Infelizmente, tempo de execução e qualidade são, muitas vezes, incompatíveis, e o *time-to-market* cada vez menor exige que as ferramentas de auxílio de projeto sejam cada vez mais eficientes.

Projetar circuitos de alta performance que dissipem pouca potência é o grande objetivo da indústria. Este motivo move o desenvolvimento/aperfeiçoamento de novos algoritmos. Percebe-se, porém, que os atuais algoritmos para síntese física, em especial os que tratam da minimização de interconexões (afetando diretamente o congestionamento, o *timing* e a potência do circuito), ainda estão longe do resultado ótimo. Segundo o artigo publicado pelo grupo do professor Jason Cong, da UCLA (CHANG C.; CONG, J., 2003), ainda existe uma grande distância entre as soluções apresentadas pelas ferramentas de posicionamento e as soluções ótimas. De acordo com os experimentos efetuados pelo Prof. Cong, o comprimento de fios requerido pelas soluções das ferramentas de que se dispõe é, em média, de 1,46 a 2,38 vezes o das soluções ótimas. Os autores apresentam também o quanto essa qualidade de solução piora conforme aumenta o tamanho dos circuitos (de 4 a 25% para circuitos 10 vezes maiores). Ora, considerando-se que os elementos a serem tratados estão cada vez menores, que o número de transistores aumenta de forma impressionante – "Realismo" da Lei de Moore (MOORE, 2006) –, assim como os efeitos elétricos, dificilmente o tamanho das interconexões poderá, no paradigma atual, ser reduzido e tratado a fim de que se alcance o objetivo almejado pelas indústrias de semicondutores (mais performance, menos potência). Nesse sentido, são necessários não apenas o reprojeto de células e a melhoria constante dos algoritmos existentes, mas também uma mudança de paradigma no projeto de circuitos.

Os circuitos em três dimensões (3D) surgem como uma opção, uma nova forma de reduzir as interconexões. Sua estrutura consiste no empilhamento de vários *chips* pré-fabricados (cada *chip* é uma camada, também chamada de *tier*). A comunicação entre as diferentes *tiers* é realizada por meio de conexões verticais conhecidas como TSV (Through-Silicon Via) ou vias-3D. Teórica (BANERJEE; SOURI; KAPUR; SARASWAT, 2001) e empiricamente (DAVIS et al., 2005), (DAS; FAN; CHEN; TAN; CHECKA; REIF, 2004), (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005), (CAMPARDO, G.; RIPAMONTI, G.; MICHELONI, R , 2009) circuitos 3D podem reduzir o tamanho das interconexões. O trabalho de Obenaus e Szymanski (OBENAUS; SZYMANSKI, 2003) mostra que essa redução é proporcional ao tamanho do circuito. Além disso, é razoável considerar que a

implementação de um projeto 3D possa melhorar o *timing*, se comparada com uma implementação 2D. Para isso, novas teorias e algoritmos devem ser desenvolvidos.

Grandes empresas, como IBM, Intel, AMD, Samsung, Micron, ST, Cadence, Infineon, e *startups*, como Ziptronix, Xanoptix, ZyCube e Tezzaron, estão investindo em soluções relacionadas a essa área (TEZZARON HOMEPAGE, 2009), (ZYCUBE TECHNOLOGY: HOMEPAGE, 2009), (3D ICS INDUSTRY SUMMARY HOMEPAGE, 2009). Na academia, são destacadas iniciativas do MIT, Cornell University, University of Minessotta, Stanford University, University of Hannover Leibniz, IMEC, Purdue University, Tohuku University entre outras.

Além da diferença topológica entre as estruturas 2D e 3D, é necessária a atenção especial para dois tópicos: questões térmicas (DAS; CHANDRAKASAN; REIF, 2004), (RAHMAN; REIF, 2001), (GOPLEN; SAPATNEKAR, 2003) e vias-3D (DAVIS et al., 2005), (KOYANAGI, M.; FUKUSHIMA, T.; TANAKA, T., 2009). Problemas de aquecimento podem ser tratados com a adição de vias térmicas (GOPLEN; SAPATNEKAR, 2005) e com os posicionadores e roteadores desenvolvidos especificamente para esse propósito (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005). Já o mecanismo de comunicação intracamadas, chamado de via-3D, impõe inúmeras limitações para o projeto VLSI 3D por quatro razões principais (ainda não abordadas satisfatoriamente na literatura): (1) todos os elementos de comunicação que implicam os níveis de integração *face-to-back* e *back-to-back* ocupam espaço na camada ativa, limitando o posicionamento de células e blocos (DAVIS, et al., 2005); (2) a dimensão física da via-3D é consideravelmente grande se comparada com as vias normais, de modo que sua superpopulação pode inviabilizar um projeto 3D (PAVLIDIS; FRIEDMAN, 2009); (3) as vias-3D têm características elétricas diferentes das conexões normais (JANG, D. M., et al., 2007), (RYU, C. et al., 2005), especialmente se considerarmos que as conexões verticais precisam atravessar todas as camadas de metal; (4) vias-3D impõem problemas elétricos e de *yield* tanto por seus elementos serem difíceis de fabricar quanto por consumirem muitos recursos de roteamento (RYU, C. et al., 2005), (PAVLIDIS; FRIEDMAN, 2009).

Como mencionado, é importante entender que o valor dos recursos usados pelas vias-3D é muito grande. Considerando-se que o caminho de uma *tier* para outra adjacente atravessa todas as camadas de metal, é sensato dizer que qualquer ligação vertical superior à adjacente pode ser muito mais custosa para o roteamento, sem mencionar o atraso provocado e quanta área ativa é ocupada.

Em vista disso, esta tese tem por objetivo o desenvolvimento de algoritmos para a redução do número de vias-3D. Assim, é proposto um fluxo para o projeto de circuitos 3D que considera a área do circuito, conexões verticais longas, número de *tiers*, número máximo de vias-3D, balanceamento de áreas e pinos de I/O. Estuda-se o problema pela perspectiva do particionamento, explorando-se métodos que auxiliam na compreensão da estrutura interna do circuito. O conjunto de algoritmos (fluxo completo desenvolvido) está incorporado a uma ferramenta chamada de LH-3D, descrita com detalhes ao longo do trabalho. Métodos e contribuições são apresentados e validados por meio de experimentos práticos.

O capítulo 2 deste trabalho apresenta um estudo bibliográfico sobre os circuitos VLSI 3D, suas características, tipos, metodologias de projeto e classificações. Com base nessa pesquisa, o capítulo 3 discute algumas questões que ainda estão em aberto por terem sido pouco exploradas pela literatura. Dentre elas, destacam-se as vias-3D,

descritas em detalhes na Seção 3.2.3. Os objetivos deste trabalho também são apresentados na sequência desse capitulo. Já o capítulo 4 realiza um levantamento bibliográfico de algoritmos de particionamento e posicionamento de circuitos, suas evoluções, classificações, estratégias mais usadas, trabalhos e grupos de pesquisa. No decorrer desse capítulo, discute-se, por meio da análise de trabalhos relacionados, a utilização desses algoritmos sob a perspectiva de um projeto 3D. Identifica-se, pelos estudos, que aqueles usados para reduzir o número de vias-3D são adaptações dos circuitos planares. Assim, com base nos aspectos citados, no capítulo 5 é proposto um fluxo de particionamento chamado de LH-3D, que utiliza duas estratégias distintas: análise prévia da estrutura do circuito e redução das conexões não-adjacentes. Foram escolhidos para comparação os algoritmos Alternate Pins e ZPlace (HENTSCHKE, 2007), sendo este último uma derivação do algoritmo FastPlace (VISWANATHAN; PAN; CHU, 2005) aplicado para circuitos 3D, além do algoritmo de particionamento de hipergrafos (*state-of-the-art*) mais usado no domínio VLSI 3D: hMetis (KARYPIS, 1997), (SELVAKKMARAN; KARYPIS, 2006), (HMETIS HOMEPAGE, 2009). A análise dos resultados e demais discussões acerca dos algoritmos também são feitas nesse capítulo. Na conclusão, são debatidas as contribuições deste trabalho, a análise dos algoritmos, os resultados e as possibilidades de trabalhos futuros. Na Figura 1.1 é apresentado um fluxograma que ilustra a organização deste trabalho.

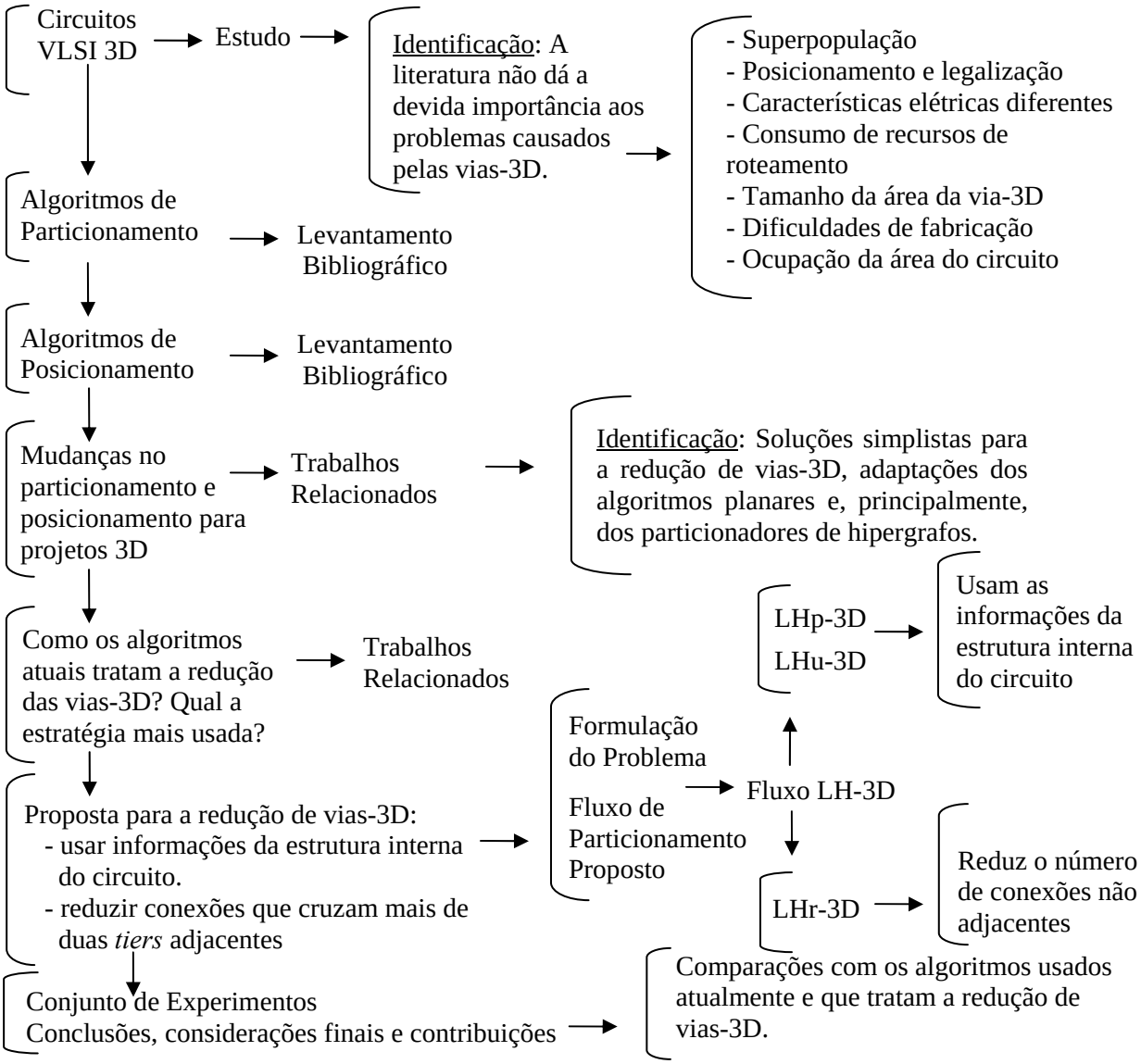


Figura 1.1: Organização do Trabalho

2 CIRCUITOS VLSI 3D

Circuitos 3D surgem como uma mudança no paradigma de projetos de circuitos integrados. São construídos através da integração de vários *chips* 2D fabricados separadamente. Cada circuito é chamado na literatura de *tier***,** e os fios que conectam *tiers* adjacentes são conhecidos como vias-3D (conexões verticais). A fabricação de um circuito 3D baseia-se na metodologia de montagem e empilhamento, na técnica de polimento para melhorar a integração, na tecnologia de alinhamento de *chips* e nas tecnologias de colagem de *chips* (colas orgânicas, metal-para-metal, etc). Este capítulo descreve alguns desses itens, assim como os desafios impostos por esse tipo de tecnologia ao desenvolvimento de metodologias de CAD.

2.1 Estratégias de Montagem de Circuitos 3D

2.1.1 Empilhamento de *Chips*

Consiste simplesmente na sobreposição vertical de *chips* pré-fabricados. A comunicação entre eles é realizada através de conexões de I/O conhecidas na literatura como "*wire bonding*" (DAVIS et. al., 2005). Essas conexões passam pelo lado de fora do circuito, fazendo a ligação entre os diferentes *chips* (Figura 2.1). Tal metodologia não provê nenhuma vantagem à performance e à potência do circuito em comparação com outras técnicas de montagem, pois sua integração é fracamente acoplada devido às conexões externas. Essa técnica reduz somente a área ocupada pelo *chip* na placa. Por esse motivo, é muito utilizada em dispositivos portáteis e telefones celulares.

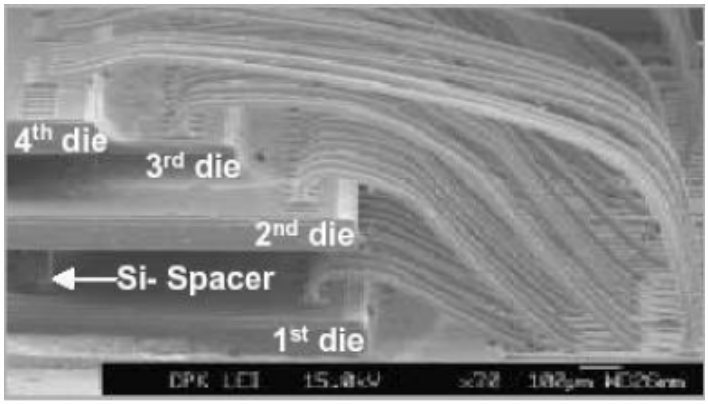


Figura 2.1: Empilhamento de *chips* de tamanhos diferentes por meio de uma estrutura *wire bonded* de comunicação (BEYNE, 2006).

2.1.2 Empilhamento de *Chip-Sobre-Wafer*

Circuitos pré-testados são empilhados sobre o *wafer* e posicionados individualmente com o uso de equipamentos *pick-and-place*. Patti (PATTI, 2006) descreve que a precisão no alinhamento depende do tipo de via utilizada para comunicar os *chips*. Atualmente, o erro no alinhamento varia em torno de 10µm. Essa técnica provê melhor integração em comparação ao empilhamento de *chips*, pois a comunicação é fortemente acoplada, ou seja, as conexões cruzam pela parte interna do *chip*. Essa tecnologia de montagem 3D é adotada por companhias como ZyCube (ZYCUBE TECHNOLOGY: HOMEPAGE), Ziptronix (ZIPTRONIX TECHNOLOGY: HOMEPAGE) e Xanoptix (CUBIC WAFER TECHNOLOGY: HOMEPAGE).

2.1.3 Empilhamento de *Wafers*

Outra técnica empilha *wafers* inteiros. Tezzaron é uma companhia que trabalha com esse tipo de montagem. Comparada com a técnica *chip-sobre-wafer*, a tecnologia utilizada pela Tezzaron (TEZZARON TECHNOLOGY: HOMEPAGE) causa erros de alinhamento inferiores a 1µm e resulta numa comunicação mais integrada e de superfície planar maior (GUPTA et. al., 2005). A principal diferença entre a técnica de empilhamento de *wafers* e a de *chip-sobre-wafer* é o número de *chips* tratados ao mesmo tempo e o baixo nível de erros de alinhamento da primeira.

2.1.4 Empilhamento de Transistores

Atualmente, ambas as técnicas de montagem (*chip-sobre-wafer* e empilhamento de *wafers*) são usadas comercialmente. A técnica de empilhamento de transistores integra camadas ativas fabricadas em um mesmo *chip* descartando qualquer tipo de alinhamento. Em outras palavras, a deposição e remoção de materiais formam os *chips*. Essa é a técnica ideal devido ao preciso alinhamento das vias e ao forte acoplamento que a caracterizam. Contudo, as altas temperaturas resultantes desse procedimento inviabilizam sua aplicação. A tecnologia de fabricação de transistores de alta performance demanda temperaturas que destroem o cobre e o alumínio utilizado para as camadas de metal. Porém, é uma técnica muito promissora e com um grande campo para pesquisa (GUPTA et al., 2005).

2.2 Estratégias de Integração 3D

Como vimos anteriormente, os circuitos 3D são construídos por meio da sobreposição de vários *chips.* Cada *chip* empilhado representa uma camada do circuito é conhecido na literatura como *tier.* Eles são cuidadosamente alinhados e colados. Existem três estratégias de integração: *face-to-back*, *face-to-face* e *back-to-back*, como mostra a Figura 2.2.

Na estratégia *face-to-back*, os *chips* são empilhados todos numa mesma orientação. No topo da última camada de metal do *chip* 1 – como no exemplo da Figura 2.2 (a) – existe uma camada de isolante para que depois seja posicionado o substrato do *chip* 2. Para a fabricação das vias-3D, deve-se abrir uma fenda no isolante e no substrato. Posteriormente, essas fendas são preenchidas com metal. A via deve conectar a última camada de metal do *chip* 1 e a primeira camada de metal do *chip* 2.

Na estratégia *face-to-face*, os chips são empilhados um de frente para o outro - como na Figura 2.2(b). A última camada de metal do chip 1 é colocada com a frente

voltada para a última camada de metal do chip 2 (separado somente por um isolante). A via que conecta o chip 1 com o chip 2 ocupa uma área menor do que a resultante da estratégia *face-to-back*. Por outro lado, na estratégia *face-to-face*, apenas duas camadas ativas podem ser empilhadas, pois as seguintes tendem a ser integradas conforme as estratégias *back-to-back* ou *back-to-face*. A fabricação que segue a estratégia face-to-back é mais fácil, pois são necessárias poucas mudanças nos processos tradicionais.

A Figura 2.2 (c) mostra duas camadas ativas conectadas, ilustrando a estratégia de integração *back-to-back*. Nesse caso, para que a comunicação entre os *chips* seja realizada, ambas as camadas ativas devem ser perfuradas, o que cria o espaço para a via-3D. Percebe-se, na Figura 2.2 (c), que essa via é bastante grande se comparada a um transistor, o que é ruim devido às características elétricas incorporadas pelos tipos de materiais. As próximas seções destacam, com maiores detalhes, os tipos de vias e sua influência no projeto de circuitos 3D. Todavia, a cada integração que utiliza a estratégia *back-to-back*, duas novas integrações *face-to-face* são possíveis.

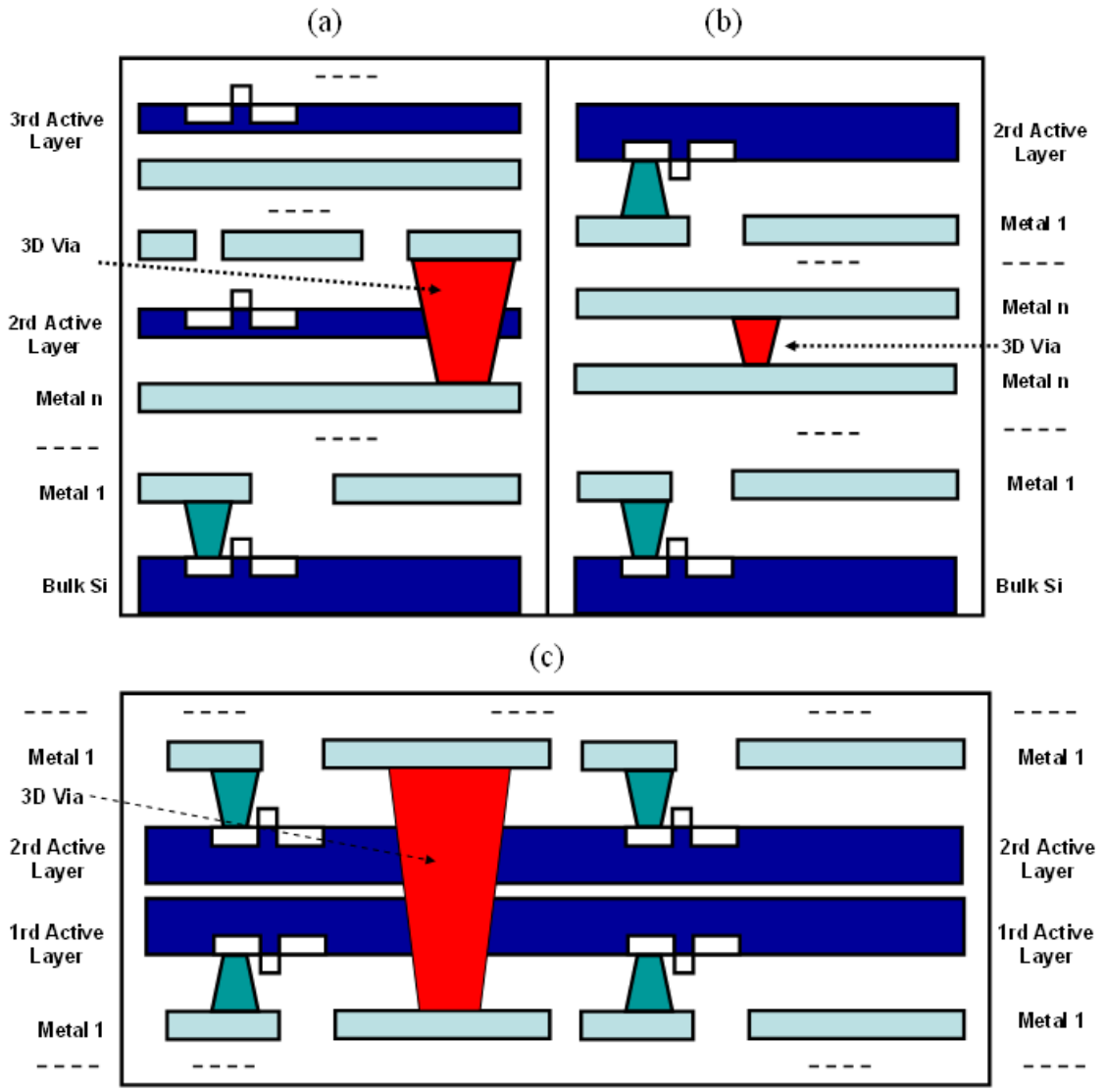


Figura 2.2: Estratégias de integração: (a) *face-to-back*; (b) *face-to-face*; (c) *back-to-back.*

As três estratégias citadas podem ser combinadas com tecnologias SOI (*Silicon on Insulator*) ou *bulk*. A Figura 2.3 mostra a combinação de tecnologias e estratégias de integração. No uso da estratégia *face-to-back* com tecnologia *Bulk*, percebe-se que o custo de perfuração é maior se comparado com a tecnologia SOI. Consequentemente, o tamanho da via-3D também aumenta. Atualmente, a tecnologia SOI é a mais utilizada, mas é difícil definir com precisão a proporção de sua utilização, em razão dos diferentes tipos de processos empregados e dos benefícios específicos de cada tecnologia.

São referidas como "*mixed integration*" as estratégias, comumente utilizadas, que combinam técnicas *back-to-back, face-to-face e face-to-back.* A Tezzaron, por exemplo, utiliza-a para alcançar o maior número de integrações *face-to-face*, nem que isso custe integrações *back-to-back.* A Figura 2.3 mostra as diferentes tecnologias e estratégicas empregadas nesse tipo de integração.

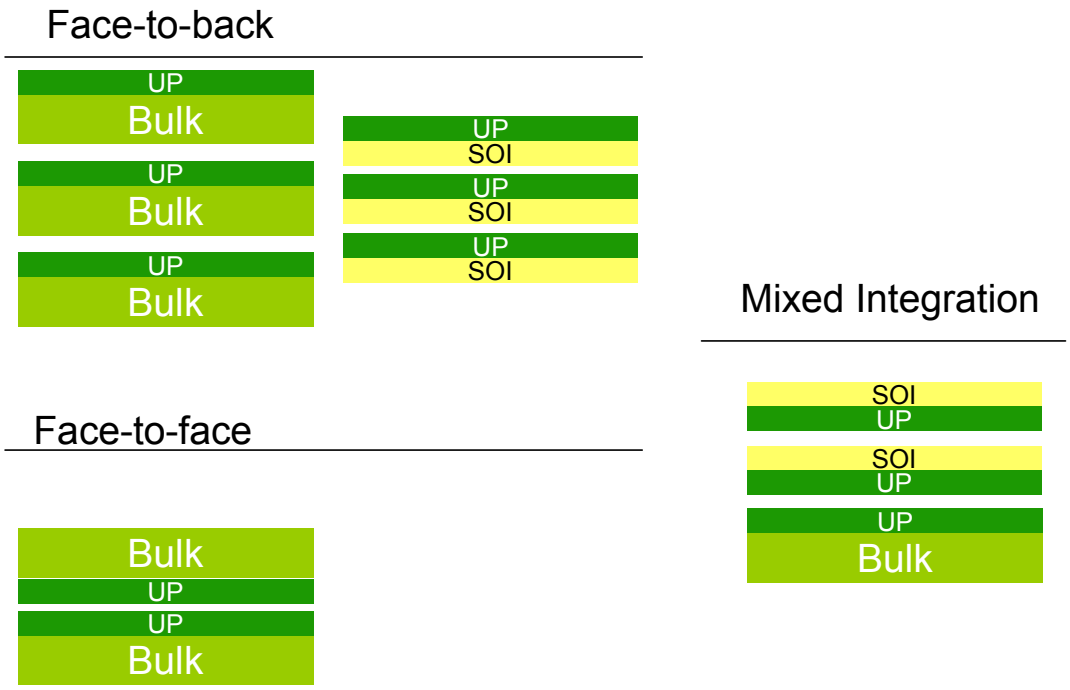


Figura 2.3: Estratégias de integração *face-to-back, face-to-face* e *mixed integration* com tecnologias *bulk* e SOI.

2.3 Estratégias de Comunicação 3D

Esta seção descreve algumas estratégias de comunicação que ligam as diferentes camadas do circuito 3D. De acordo com Davis (DAVIS et. al., 2005), a comunicação pode utilizar quatro tipos de estratégias: *wire bonded, microbumps, contactless* e vias-3D.

A tecnologia *wire bonded* utiliza *chips* empilhados de tamanhos diferentes e *pads* de I/O posicionados em suas bordas. Seguindo-se essa estratégia, não se utiliza conexões *interchips*, de modo a evitar o congestionamento interno que seria imposto por cada uma das camadas. A principal desvantagem dessa tecnologia é justamente as conexões serem traçadas pelo lado externo dos *chips*, o que torna a integração fraca e

aumenta o atraso da comunicação entre eles. Além disso, o número de repetidores e o tamanho dos *pads* aumentam.

A tecnologia *microbumps* utiliza microcontatos de ouro ou cobre (*bumps*) posicionados no topo da camada de metal. Algumas vezes, esses contatos podem ser bloqueados por outras conexões. Assim, pode-se precisar de mais espaço para roteamento. Contudo, para posicionar *chips face-to-face*, não são necessários canais de roteamento. Uma desvantagem dessa estratégia de integração é estar ela limitada a duas camadas, exceto quando for utilizada a integração *back-to-back*. Ambas são conectadas, obtendo-se o contato físico entre os *bumps* das duas.

A tecnologia *contactless* pode ser resumida como uma conexão indutiva e outra capacitiva (CULURCIELLO, E.; ANDREOU, 2006). Esta requer que os *chips* estejam posicionados na forma *face-to-face* devido à proximidade entre os contatos. Com isso, limita-se novamente o número de camadas a duas. Já a acoplagem indutiva é usualmente utilizada nas estratégias de integração *face-to-back*.

Por fim, o uso de vias-3D consiste em escavar um buraco de uma camada para outra, quando a estratégia de integração é *face-to-back* ou *back-to-back.* Em alguns casos, tal como na tecnologia MITLL 3D (DAVIS et. al., 2005), as primeiras duas camadas são integradas *face-to-face* e as restantes *face-to-back*. No primeiro caso, a comunicação é realizada por meio de contatos, enquanto, no segundo, são necessários *pads* de I/O. A tradicional tecnologia *bulk*, em comparação com o processo SOI, requer vias bastante grandes (MITLL-SOI *face-to-back* exige um *pitch* de 5 µm, enquanto a via-3D em *bulk face-to-back*, um *pitch* de 50 µm) e, em consequência, uma distância maior entre elas, acarretando, assim, o aumento de área.

Tabela 2.1: Resumo das tecnologias para integração e comunicação 3D (extraído de DAVIS et. al., 2005)

Tecnologia de comunicação	**Tecnologia de integração**
Wire bonded	Fios do lado de fora do *chip*
Microbump	Conexões *face-to-face*
Contactless	Conexões capacitivas Conexões indutivas
Through via	*Bulk* SOI

A fabricação de um circuito 3D depende dos seguintes quesitos: (1) metodologia de montagem e empilhamento, (2) técnica de polimento usada para melhorar a integração, (3) tecnologia de alinhamento de *chips* e (4) tecnologia de colagem de *chips* (colas orgânicas, metal-para-metal, etc.). Atualmente, uma grande variedade de tecnologias é encontrada em diversos projetos, e cada uma delas afeta de forma diferente a geometria, o roteamento e as características elétricas das vias-3D. Por exemplo, a distância entre estas varia significativamente conforme a qualidade do

alinhamento nas etapas de montagem. Todas essas tecnologias estão em constante aperfeiçoamento. Como resultado, espera-se, para o futuro, um maior avanço no projeto de vias-3D.

O comprimento das vias e a distância entre elas afetam bastante o projeto do circuito. A distância entre vias, numa mesma *tier*, determina quantas podem ser efetivamente utilizadas pelas ferramentas de CAD e pelos projetistas. Determina, também, como as conexões verticais podem ser posicionadas, sem mencionar a área ocupada na camada ativa. O tamanho das vias (que depende do tipo de integração, se *face-to-face, face-to-back ou back-to-back*) tem extrema importância e deve ser levado em conta para o cálculo do *wirelenght*.

Características do roteamento em vias também são importantes para o projeto de circuitos 3D. Algumas formas de construção de vias bloqueiam camadas de metal. Nesses casos, o roteamento passa a ser bastante custoso. Quando se utiliza estratégias de integração *face-to-back* e *back-to-back*, é necessário furar a camada ativa. Nessa hipótese, a localização da via-3D torna-se uma barreira e ocupa área de posicionamento.

As características elétricas das conexões são muito importantes para computar o atraso e o consumo de potência. Capacitância e resistência altas podem invalidar o uso de vias-3D para conexões críticas do circuito. Contudo, alguns estudos buscam posicionar as células que compõem os caminhos críticos em uma camada, procurando evitar que as suas conexões tornem-se vias-3D.

2.3.1 Tecnologia Tezzaron

A tecnologia Tezzaron baseia-se na estratégia de montagem denominada empilhamento de *wafers*, descrita na Seção 2.1.3. Os caminhos verticais empregados nessa estratégia são vias-3D (perfuração da camada ativa) e contatos *microbumps*. Algumas informações didáticas de seus processos estão disponíveis no *website* da empresa (TEZZARON HOMEPAGE, 2009), mas essas descrições não são completas.

A Tezzaron conecta as camadas utilizando estratégias de integração *face-to-face* e *face-to-back*. Cada uma das *tiers* contém *pads* de cobre que são acoplados no momento da junção. Na integração *face-to-face*, o contato físico dos cobres serve como vias-3D, similar à tecnologia *microbump,* descrita na seção anterior.

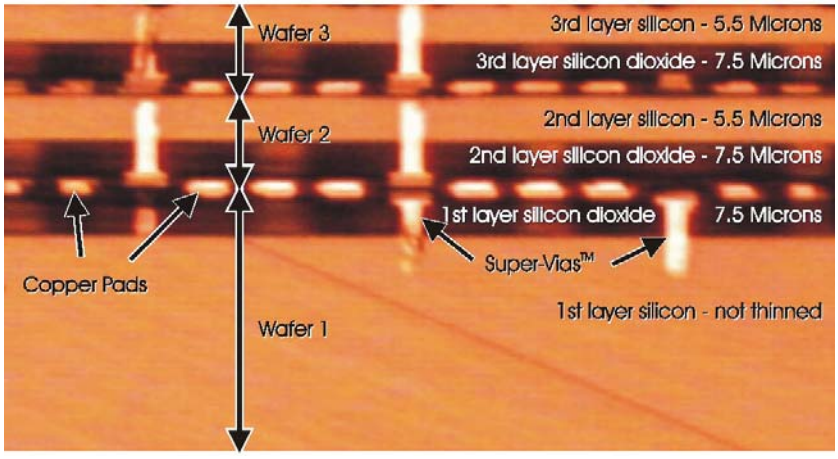


Figura 2.4: Diferentes estratégias de integração utilizando conexões supervias (TEZZARON HOMEPAGE, 2006).

Atualmente a Tezzaron trabalha com duas tecnologias de vias, supervias e supercontatos (ambas marcas registradas pela empresa). A supervia conecta os *pads* de cobre por uma perfuração que atravessa todas as camadas de metal. Por essa razão, ela bloqueia a passagem dos metais para roteamento. O supercontato conecta os *pads* de cobre com auxílio das camadas de metal. Assim, ao contrário da supervia não torna-se um obstáculo para o roteamento.

A Figura 2.4 ilustra algumas conexões supervia, duas ao topo do *wafer,* duas ao centro e duas mais à direita da última camada. Percebem-se, também, os *pads* de cobre entre o primeiro e o segundo *wafers* integrados conforme a estratégia *face-to-face.*

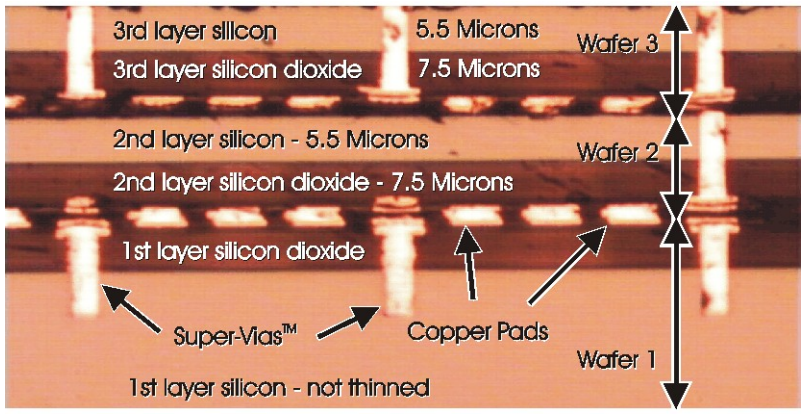


Figura 2.5: Conexões supervia. Três na parte superior, uma ao centro e três na camada de baixo. O *wafer* 1 está integrado ao *wafer* 2 conforme a estratégia *face-to-face* (TEZZARON HOMEPAGE, 2009).

A Figura 2.5 ilustra um processo da Tezzaron com integrações *face-to-face* e *face-to-back.* Verifica-se que a estratégia *face-to-face* foi utilizada para integrar a *tier* 1 à 2, e a *face-to-back* para a integração das tiers 2 e 3. Os *pads* de cobre conectam as *tiers* 1 e 2. Na integração *face-to-back* das *tiers* 2 e 3, a conexão entre as duas camadas é realizada por uma supervia que conecta os *pads* de cobre. A *tier* 1 conecta os *pads* de I/O do circuito que estão localizados abaixo do *bulk*.

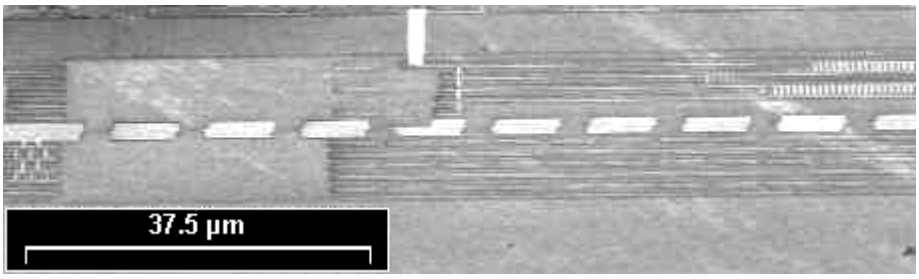


Figura 2.6: Processo de alinhamento de *pads* (TEZZARON HOMEPAGE, 2007).

A Figura 2.6 mostra dez pares de *pads* de cobre, nas bordas, que conectam dois *wafers*. Constata-se a dificuldade dos seus alinhamentos. Cinco camadas de alumínio podem ser visualizadas em cada *wafer*. Conexões verticais entre essas camadas estão localizadas nos lados inferior esquerdo e superior direito. Um supercontato aparece no topo central desta figura (TEZZARON HOMEPAGE, 2007).

A tecnologia de supercontatos foi introduzida na segunda geração de processos de fabricação da Tezzaron. Essa tecnologia requer o uso de todas as camadas de metal. A Figura 2.7 mostra um processo atual da Tezzaron utilizando essa estratégia em duas *tiers* conectadas *face-to-face*. No canto superior visualiza-se claramente que o supercontato não atravessa as camadas de metal. Essa conexão para na primeira camada de metal, facilitando o trabalho do roteamento.

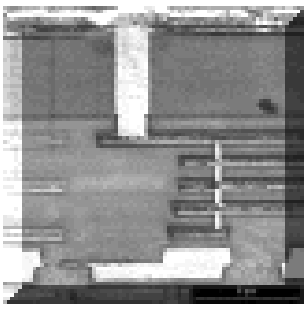


Figura 2.7: Corte de um *wafer*. Um *pad* de cobre aparece abaixo, e pequenas interconexões entre camadas de metal, no centro. Um supercontato conecta um *pad* à última camada de metal (TEZZARON HOMEPAGE, 2009).

2.3.2 Tecnologia 3D MITLL

A Tecnologia 3D MITLL (SUNTHARALINGAM et. al., 2005), (DAS, S.; CHANDRAKASAN, A.; REIF, R., 2004a) utiliza uma estratégia de integração similar à da Tezzaron. Constrói-se uma pilha de *tiers* com as duas primeiras camadas *face-to-face* e todas as outras *face-to-back.* Na Figura 2.8 mostra, de forma didática, a tecnologia utilizada pela MITLL.

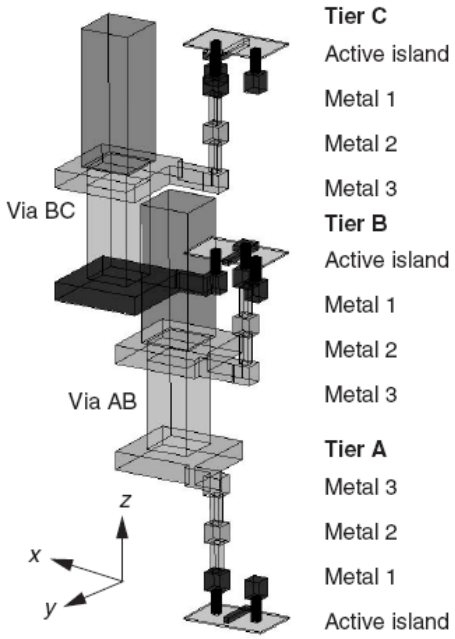


Figura 2.8: Modelo 3D do processo MITLL (DAVIS et. al., 2005).

Seguindo-se a estratégia de integração MITLL *face-to-face*, em oposição à tecnologia Tezzaron, porém, utiliza-se conexões verticais mais custosas. Isso acontece

porque é necessária uma conexão reta e, com isso, bloqueiam-se as demais camadas de metal, ao passo que a tecnologia Tezzaron exige uma conexão somente até a primeira camada.

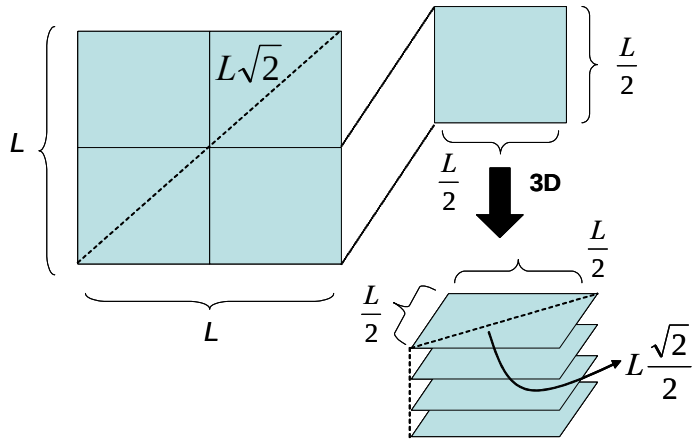


Figura 2.9: Redução do tamanho das conexões em circuitos 3D.

2.4 Projeto de Circuitos 3D

A tecnologia 3D provê grandes possibilidades para melhorias de desempenho, potência, *timing, wirelength*, entre outros. Porém, seu uso exige mudanças nos processos de fabricação. Com isso, vêm à tona questões delicadas, como novos paradigmas de projeto e, consequentemente, novas metodologias e ferramentas. Este tópico explora três metodologias de projeto: nível de camadas, nível de blocos e nível de células; destaca-se, aqui, seu grau de integração e granularidade. Discute-se ainda o impacto dessas estratégias nas etapas de particionamento e posicionamento VLSI. Problemas como *yield*, questões térmicas, vias-3D, entre outros, também são abordados nesta seção.

2.4.1 Metodologias de Projeto 3D

A primeira metodologia, conhecida como "integração em nível de *tier*", empilha *chips* de diferentes naturezas. Essa metodologia não afeta as de projetos já existentes, pois cada *chip* (*tier*) pode ser projetado separadamente e, em seguida, integrado com facilidade. Nesse nível, a granularidade é alta, pois o contexto interno das *tiers* não são discutidos e projetados em conjunto. A segunda metodologia, chamada de "integração em nível de *IP core*", particiona seus blocos (*IP cores*) entre diferentes *tiers*, promovendo, dentro destas, uma integração acoplada. Os blocos podem ser distribuídos de forma organizada, em cada *tier* ou entre elas, como um *floorplanning* 3D, o que resulta num nível de granularidade médio. Por fim, a "integração em nível de lógica aleatória" particiona um simples bloco de lógica entre diferentes *tiers*, gerando uma granularidade menor (células). Essa metodologia trabalha com um nível de integração elevado.

Nível de *tier*: Atualmente, na área de SoCs, a fabricação 3D cria novas possibilidades. A integração 3D permite acomodar elementos de um sistema em diferentes *chips*, evitando problemas, como ruídos em sistemas analógicos (JAIN, V. K.; BHANJA, S.; CHAPMAN, G. H.; DODDANNAGARI, L., 2005). Componentes de diferentes naturezas, fabricados por diferentes processos (lógica aleatória, RF, DRAM,

SRAM, lógica *low-power*, partes analógicas, plataformas programáveis, como *software*, FPGAs, *Flash*) podem ser posicionados em camadas distintas, conforme ilustrado na Figura 2.10.

De acordo com Patti (PATTI, et. al., 2006), a Tezzaron apresenta alguns projetos empilhando *tiers* de naturezas diferentes. Porém, o nível de integração é baixo, pois a granularidade é alta. Um dos benefícios dessa metodologia é que ela utiliza as ferramentas de CAD já existentes.

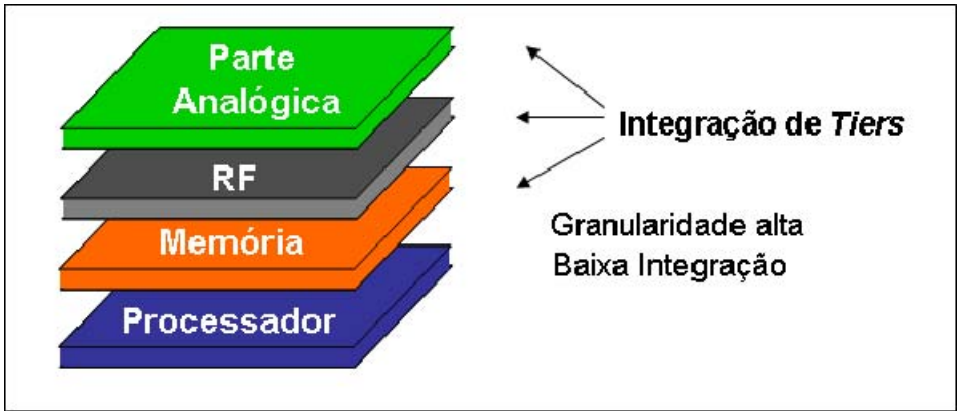


Figura 2.10: Integração em nível de *tier.*

Nível de *IP Core*: Esta seção discute a metodologia de particionamento de projeto em *IP cores*, como ilustrado na Figura 2.11. A idéia é aplicada no nível de posicionamento de macroblocos direcionado para melhorias de *wirelength* e *timing.* Comparado à metodologia em nível de *tier*, a integração *IP core* demanda mais esforço de projeto. Os elementos a serem tratados são menores, criando uma granularidade média de integração.

A etapa de *floorplaning* caracteriza-se pelo posicionamento de macroblocos em uma determinada área e pelo redimensionamento dos blocos, o que restringe a área e a relação de aspecto. Em outras palavras, cria-se uma planta baixa com o objetivo de encontrar o melhor arranjo entre os blocos. Tal arranjo visa reduzir o tamanho das conexões e a área do circuito. Algumas metas do *floorplaning* 3D são estas: (1) prover uma boa compactação dos blocos para que o melhor potencial dos circuitos 3D possa ser explorado; (2) prover um bom particionamento de blocos dentro da *tier* (esse particionamento auxilia o restante do processo para que se alcance boas soluções); (3) prover resultados melhores de redução do tamanho das conexões do que os das soluções 2D; (4) dar atenção especial a problemas de *timing* e potência, além de lidar com redes críticas; (5) prover soluções para problemas térmicos em circuitos 3D.

A diferença entre *floorplanners* (ferramentas de planta baixa automatizadas) está na estrutura de dados usada para armazenar o *layout* e nos métodos para otimizá-los. Cong e Zhang (CONG; WEI; ZHANG, 2004) apresentam um *floorplaning* 3D baseado em *simulated annealing* (KIRKPATRICK; GELATT; VECCHI, 1983). Nesse trabalho, foi introduzida uma estrutura de dados aplicada para otimização de conexões e temperatura. Baseia-se ela em três diferentes modelos termais que não consideram as vias-3D como possíveis dissipadores de calor. Esses autores mencionam que essa restrição deve ser incorporada como uma característica do *simulated annealing.* Essa metodologia particiona as células entre as *tiers* levando em consideração o *wirelength* e

os problemas térmicos. Cong conclui que seu metodo é capaz de reduzir o tamanho das conexões em 29% e a temperatura máxima do *chip* em 56%.

Uma metodologia para particionamento de blocos IP em três dimensões chamada de hierárquica foi estudada Zhuoyuan e Jinian (ZHUOYUAN; XIANLONG; QIANG; SHAN; JINIAN, 2006). Primeiro, os blocos são particionados e, depois, atribuídos às respectivas *tiers*. Destaca-se que esse particionamento reduz o tamanho do problema, provendo maior convergência. Contudo, limita as possibilidades do algoritmo encontrar bons resultados em *wirelength*.

Um método híbrido para a otimização do *floorplaning* 3D proposto por Healy e Loh (HEALY, M.; LOH, G. H., 2007) baseou-se em programação linear e *simulated annealing*. Sua discussão gira em torno da eficácia do uso de vias-3D. Os autores defendem que elas devem ser evitadas devido ao grande *pitch* e aos problemas provocados com *yield*, ao passo que vias *face-to-face* podem ser utilizadas largamente, desde que contribuam para a redução do *wirelenght*. Essa abordagem provê um modelo para medir a temperatura do *chip* e não insere vias térmicas[1].

A questão da inserção de vias térmicas em seus *floorplanners* 3D foi discutida por Wong e Lim (WONG; LIM, 2006). Concluíram que essas vias não devem ser separadas da otimização do *floorplaning*. Em resumo, os trabalhos estudados constataram que o tratamento térmico com *floorplaning* é capaz de prover resultados consideráveis de otimização.

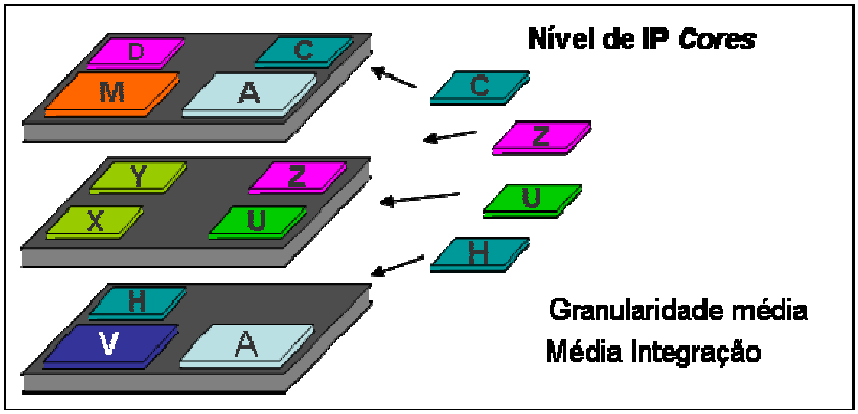


Figura 2.11: Integração em nível de *IP cores.*

Nível de integração em lógica aleatória: Este nível de integração provoca baixa granularidade, permitindo uma maior integração. Esse fato acontece porque os elementos a serem tratados são pequenos (células). Quanto menores os elementos, mais detalhada é a otimização, o que resulta em uma maior integração, como mostra a Figura 2.11.

Nesta seção, discute-se a idéia de que um circuito *standard cell* seja particionado e posicionado em mais de uma *tier*. Essa abordagem atinge diretamente sua estrutura

[1] São vias-3D isoladas eletricamente que cruzam todas as camadas e servem como um caminho para a dissipação do calor.

geométrica (área do circuito), a posição das células e as redes, pois a *tier* deve ser levada em consideração. Os pontos importantes a serem tratados são estes (1) considerar que algumas conexões irão para o topo da camada de metal e passarão através de uma conexão (via-3D) que possui características elétricas diferentes; (2) posicionamento e roteamento de células terão mais obstáculos (vias-3D); (3) os pinos de I/O deverão ser levados em consideração, pois muitos dos algoritmos analíticos de posicionamento necessitam dos *pads* fixos para posicionar as células. Com isso, os algoritmos aplicados em estruturas 2D devem ser revisados e adaptados para esse novo paradigma.

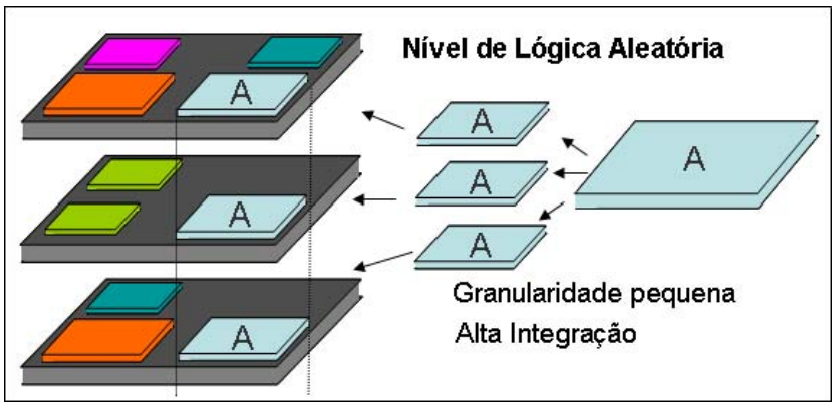


Figura 2.12: Integração em nível de lógica aleatória.

Maior integração pode acarretar a utilização de mais vias-3D. Patti (PATTI, et. al., 2005) discutem a relação entre o aumento do número de vias e a redução do tamanho das conexões. Contudo, esse é um assunto que deve ser analisado com maior profundidade, pois a perfuração ocasionada pelas vias-3D pode comprometer o *yield* do circuito. Pode-se utilizar estratégias *face-to-face* (existe limitação, como visto nos tópicos anteriores) quando necessária, evitando-se a perfuração do substrato.

Atualmente, tópicos como “Integração 3D” e “Conexões 3D” estão ganhando mais espaço em conferências técnicas especializadas. Grandes organizações investem em pesquisa e desenvolvimento de circuitos integrados 3D. A empresa Tezzaron Semiconductor (TEZZARON TECHNOLOGY: HOMEPAGE) comenta que esse interesse reflete o reconhecimento de que, no futuro, os *chips* serão construídos em três dimensões. Alguns centros de pesquisa, produtos e serviços estão descritos no anexo 1 desta tese.

2.5 Resumo do Capítulo

Este capítulo apresentou um levantamento bibliográfico sobre os circuitos VLSI 3D, suas metodologias de projeto, classificações, características e tipos.

Conforme foi explicitado, as metodologias de projeto podem ser representadas em três níveis: o das *tiers*, o de *ips* e o da lógica aleatória. Além disso, é importante que a granularidade seja considerada em um projeto 3D. Por exemplo, em nível de *tiers*, podem-se acomodar elementos heterogêneos em cada uma das camadas, evitando-se problemas de ruídos, além de ser possível adaptar sistemas de diferentes naturezas. Nesse caso, a granularidade é dita alta, contudo com integração baixa. No nível de *ips*, os blocos que compõem o circuito são divididos entre as *tiers*. Assim, eles podem migrar para qualquer uma das camadas do circuito. Caracteriza-se essa metodologia

como de granularidade e integração médias. Já o nível de lógica aleatória, por sua vez, trabalha com elementos ainda menores, as células. Percebe-se na literatura que a ideia de dividir blocos de lógica aleatória é pouco explorada, mas, em projetos em três dimensões, estudos mostram que ela reduz o *wirelenght* e, consequentemente, o atraso e a área em comparação com projetos 2D tradicionais (DAVIS et al., 2005), (BANERJEE; SOURI; KAPUR; SARASWAT, 2001), (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005), (OBENAUS; SZYMANSKI, 2003). Além disso, sua granularidade é pequena, o que possibilita uma maior integração.

A integração dos elementos em um circuito é um fator determinante para o aumento de sua performance. Quanto menores os elementos envolvidos, maior será sua aproximação física e comunicação. Com base nessa constatação, o foco deste trabalho direcionou-se para projetos com granularidade pequena. Além disso, tratando-se de elementos com esse nível de granularidade, pode-se facilmente utilizar as soluções encontradas para outros níveis de projeto.

Verificou-se também que a estratégia conhecida como empilhamento de *wafers* é a mais usada atualmente. Isso se deve à sua produtividade que permite empilhar vários *chips* de uma só vez. Mas sua maior vantagem está na precisão do alinhamento. Seu uso resulta numa comunicação mais integrada e seus erros são inferiores a 1µm. Contudo, a estratégia conhecida como empilhamento de transistores é a mais adequada, pois trabalha com deposição e remoção de materiais. Com isso, ela evita o problema do alinhamento devido a empilhamento dos elementos. O objetivo do empilhamento de transistores é criar um circuito 3D, do início ao fim, somente por meio de deposição e remoção de materiais (é importante ressaltar que essa estratégia ainda está em estudos, pois esse processo ocasiona o aquecimento dos materiais, o que inviabiliza o projeto).

No próximo capítulo, são destacadas as vantagens dos circuitos 3D, juntamente com uma discussão envolvendo as questões que ainda estão em aberto no projeto 3D. Ao final, são apresentados os objetivos deste trabalho e quais os caminhos a serem seguidos para alcançá-los.

3 MOTIVAÇÃO

Este capítulo discute as motivações relacionadas ao estudo de algoritmos para a redução de vias 3D. Inicialmente na seção 3.1 são apresentadas as vantagens e as motivações para o uso de circuitos 3D. A seguir, na seção 3.2, são abordados os tópicos em aberto associados à síntese de circuitos 3D. Por fim, na seção 3.3 são descritos os objetivos deste trabalho.

3.1 Vantagens dos Circuitos 3D

As análises de Banerjee (BANERJEE et. al., 2001) e resultados práticos publicados por Davis (DAVIS et. al., 2005) e Pavlidis (PAVLIDIS, V., 2009) descrevem melhorias obtidas pelos circuitos VLSI através da metodologia 3D. Além disso, esse potencial pode ser explorado com propriedade pelas ferramentas de CAD.

Algumas vantagens são listadas abaixo e discutidas nas próximas seções.

- Redução do tamanho dos fios: É muito claro o potencial de redução do tamanho das conexões em circuitos 3D. O tamanho máximo possível em um *chip* é igual ao seu semiperímetro total, ou seja, *largura+altura* (PAVLIDIS, V. F.; FRIEDMAN, E.G, 2009). Em 3D, ambas, altura e largura, diminuem, pois a área total 2D deve ser particionada em duas ou mais *tiers*, como mostra a Figura 2.9.
- Redução da média do tamanho das conexões: Esta é uma conclusão que aparece em praticamente todos os artigos que tratam de circuitos 3D (DAVIS et. al., 2005). Reduzindo-se o tamanho médio das conexões, diminui-se o tamanho total dos fios.
- Redução da potência: Com o *wirelenght* reduzido, diminuem-se os focos de congestionamento, pois aumenta o número de conexões curtas (JOYNER; MEINDL, 2002). Consequentemente, com os caminhos mais curtos, a média da capacitância e da resistência são reduzidas, assim como o número de repetidores para os caminhos mais longos. Anteriormente, esse quadro acarretava um crescimento significativo de consumo. Davis (DAVIS et. al., 2005) também discute a questão de potência, e, em seus experimentos, a potência total melhorou em até 23%.
- Melhorias de *timing*: A redução das conexões auxilia a melhoria do *timing.* Espera-se que as metodologias dirigidas a *timing* consigam alcançar bons resultados na otimização de alguns caminhos (JOYNER, J. W. et al., 2001),

(STROOBANDT, D.; CAMPENHOUT, J. V., 2000), (PAVLIDIS, V. F.; FRIEDMAN, E.G., 2009).

- Melhorias de ruído: A redução total do *wirelenght* e do número de repetidores causam menor ruído, promovendo uma melhor integridade do sinal (CHERKAUER, B. S.; FRIEDMAN, E. G., 1995), (PAVLIDIS; FRIEDMAN, 2008).
- Redução de área: A possibilidade de empilhar camadas ativas permite a redução da área de um *chip*. Pesquisas (IEONG, 2003) mostram que a área total do *layout* (a soma da área de componentes e metais de roteamento) de um circuito 3D, comparada com a de um circuito 2D, pode ser reduzida em aproximadamente 30%, utilizando-se circuitos "standard-cells" (PAVLIDIS; FRIEDMAN, 2008).
- Melhorias na densidade: A melhoria de performance pode ser alcançada com a redução do tamanho das conexões. Por exemplo, a Intel (LOH, G.; XIE, Y.; BLACK, B, 2007) demonstrou que a modificação no *pipeline* para duas camadas resulta na melhora da performance em aproximadamente 15%. Vaidyanathan (VAIDYANATHAN, et. al., 2007) também mostraram que o projeto de uma unidade lógica aritmética em duas camadas pode diminuir de 10 a 30% do atraso devido à redução do tamanho das conexões. Outro fator importante é a flexibilidade do projeto. A tecnologia 3D permite que as memórias sejam posicionadas acima ou abaixo de componentes relacionados, resultando num ganho significativo de performance na comunicação entre memória e microprocessador. Atualmente, como a quantidade de memória em *chips* está aumentado (por exemplo, a maior parte dos *chips* é ocupada por memórias), o caminho da lógica dos componentes para a memória torna-se um fator limitante em alguns sistemas (DENG; MALY, 2005). O empilhamento de lógica e memória já foi explorado por Kunio (KUNIO, 1989) e o particionamento de *cache* por Tsai, Wang e Xie (TSAI, et. al., 2008).
- Melhorias de funcionalidade: A integração 3D permite a incorporação de novos elementos, impedidos pela atual convenção planar. Em circuitos 3D, o projeto se torna mais flexível, o que permite a inserção de diferentes combinações de tecnologias para criar um circuito híbrido (JAIN, V. K.; BHANJA, S.; CHAPMAN, G. H.; DODDANNAGARI, L., 2005). A fabricação 3D abre novas possibilidades, porque, além de encurtar conexões, essa integração viabiliza o arranjo inteligente de diferentes elementos de um sistema em um único *chip*. Componentes de naturezas distintas, tais como lógica aleatória, DRAM, SRAM, lógica *low-power* e alta performance, parte analógica, RF, plataformas programáveis (*software*, FPGAs, *Flash*) podem ser posicionados em diferentes camadas ativas. Além de uma melhor organização, os efeitos causados pelos ruídos são diminuídos (MATSUO, et. al., 1997). Cada parte tem seu próprio silício separado por um isolante.

3.2 Questões em Aberto no Projeto 3D

3.2.1 Questões Térmicas

As questões térmicas podem ser tratadas de diversas formas e níveis: algoritmos de *floorplannig* voltados aos problemas térmicos (CONG; WEI; ZHANG, 2004),

algoritmos de posicionamento (OBERMEIER; JOHANNES, 2004), algoritmos de roteamento (ABABEI et al., 2005), (ZHANG; ZHANG; SAPATNEKAR, 2006) e inserção de vias-3D térmicas (GOPLEN; SAPATNEKAR, 2005). Obermeier (OBERMEIER; JOHANNES, 2005) implementou um posicionador baseado na idéia de posicionamento quadrático e usou duas estratégias para a redução da temperatura: espalhar as células e encurtar as redes com alto chaveamento. Vias térmicas também são uma interessante solução devido à sua eficiência na dissipação de calor. Trata-se de uma idéia simples: uma via eletricamente isolada é inserida em um *chip* somente para esse fim.

A integração 3D aumenta a densidade dos transistores e, consequentemente, a temperatura interna do *chip.* Múltiplas camadas ativas empilhadas também contribuem para as altas temperaturas. Além disso, os circuitos 3D são compostos por material isolante entre as camadas ativas, o que dificulta a dissipação do calor, como mostra a Figura 3.1.

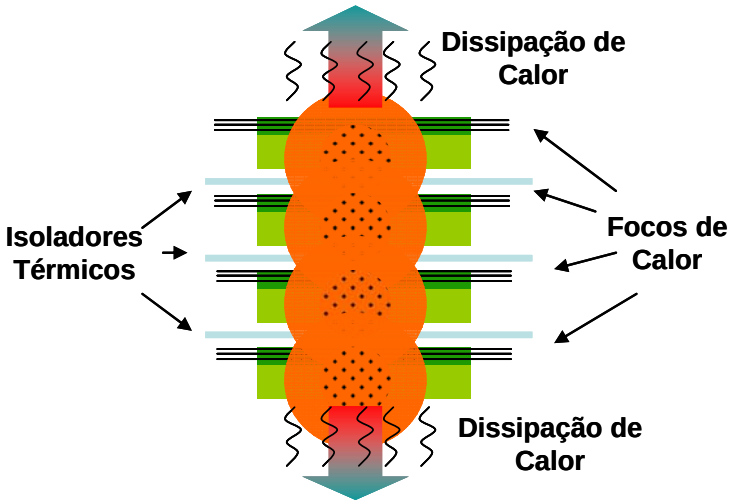


Figura 3.1: Problemas térmicos.

Tezzaron e MITLL em suas pesquisas mencionam que o problema térmico é um fator que limita o número de camadas empilhadas a 10. É fácil compreender que *tiers* intermediárias formem ilhas de calor. Uma boa técnica para dissipar o calor é a inserção de vias térmicas (GOPLEN; SAPATNEKAR, 2005), o que consiste em adicionar vias *dummy* (não conectadas ao *netlist*) com o propósito único de resolver esse problema.

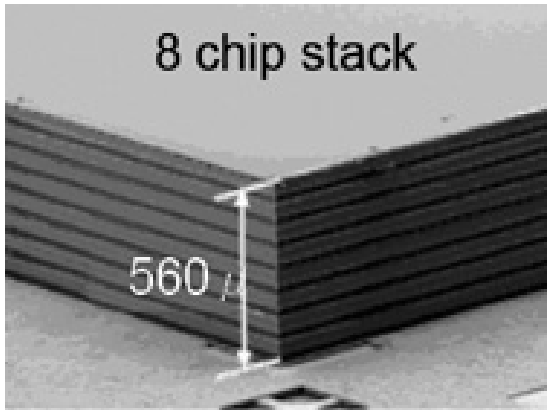


Figura 3.2: Empilhamento de *chip* com oito camadas (ZIPTRONIX HOMEPAGE, 2009).

Como visto anteriormente, existem boas soluções para problemas térmicos, porém verifica-se que esse é um problema crítico que necessita de mais pesquisa e discussão. Técnicas de empilhamento com um número reduzido de *tiers* obtêm resultados térmicos aceitáveis (duas ou três *tiers*; os maiores problemas ocorrem com mais de 10 *tiers*). A Figura 3.2 ilustra um processo da Ziptronix, que empilha oito camadas.

3.2.2 Yield

As vias-3D são difíceis de fabricar, além disso, podem ocorrer erros de alinhamento durante o processo de montagem. Como revisado na Seção 2.1, o empilhamento de *chips* insere em sua estrutura *tiers.* Apesar dos defeitos no processo de montagem, o empilhamento de *wafers* pode ser considerado a melhor opção (PATTI, 2006). Ainda assim, são necessários estudos para melhorar as técnicas de fabricação de vias e empilhamento.

A Figura 3.3 mostra um circuito 3D construído em quatro camadas por meio do empilhamento de quatro *wafers* (estratégia de montagem).

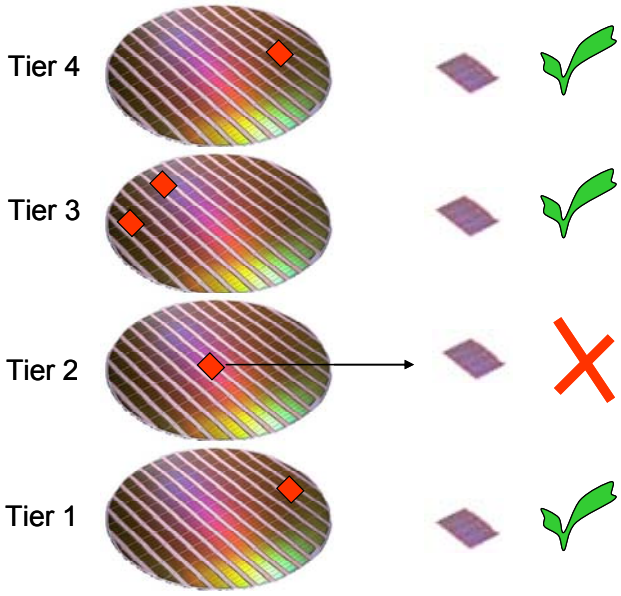


Figura 3.3: Empilhamento de *Wafers*.

Percebe-se pela Figura 3.3 que foram identificadas em todas as camadas falhas em *chips.* Caso esses *chips* não tenham interseções com as demais camadas, o *yield* será bem ruim. Por exemplo, se pegarmos a *tier* 2, a qual contém falha em um único *chip,* no momento da sobreposição das camadas, prejudicará três *chips* com funcionamento normal.

3.2.3 Vias-3D ou TSV (Through –Silicon Via)

Artigos sobre a tecnologia 3D não enfatizam os problemas relacionados ao número, à área e à posição das vias. Contudo, elas têm um papel importante na redução do tamanho das conexões e, consequentemente, no atraso e na área do circuito. Além da sua concepção física, o número de conexões verticais pode aumentar essa área, causar efeitos elétricos indesejáveis e bloquear a área de roteamento, que pode até mesmo ser inviabilizada. Essas questões certamente potencializam novos problemas e, em conseqüência, novos projetos, algoritmos e ferramentas CAD.

Superpopulação de TSVs. Dada uma área dividida em camadas (*tiers*), a comunicação entre elas necessita de conexões verticais. A inserção indiscriminada de TSVs pode afetar diretamente o aumento dessa área (MOTOYOSHI, M, 2009), (KOYANAGI, M.; FUKUSHIMA, T.; TANAKA, 2009) – Figura 3.5 (h).

Posicionamento e legalização de TSVs. Vias que perfuram o substrato ocupam o lugar de outros elementos. Essa afirmação se destaca nas integrações *face-to-back* e *back-to*-back (as vias não podem ser sobrepostas). Na integração *face-to-face*, o substrato não é perfurado, mas, mesmo assim, a distância entre a posição das vias deve ser respeitada (KAYA; OLBRICH; BARKE, 2004). Seu posicionamento leva em consideração a redução do tamanho das conexões – Figura 3.5 (a).

Obstáculos móveis. Na integração *face-to-back,* as TSVs ocupam lugar na área ativa e não podem sobrepor às células. Como são objetos móveis, esse problema é definido como posicionamento de células com obstáculos móveis (KAYA; OLBRICH; BARKE, 2004) – Figura 3.5 (c).

Características elétricas. Nos processos 2D, os detalhes de resistência, capacitância e do tamanho dos fios são otimizados, existindo uma estrutura pesquisada há anos, com modelos bem definidos. No paradigma 3D, essas características são diferenciadas, pois inúmeras conexões podem atravessar diversas camadas com diferentes tecnologias de construção de TSVs (JANG, D. M., et al., 2007), (RYU, C. et al., 2005) – Figura 3.5 (b).

Roteamento. TSVs consomem inúmeros recursos de roteamento. Posicionamento, obstáculos e número de vias são alguns fatores que interferem diretamente nessa etapa – Figura 3.5 (f).

Número de vias e tamanho das conexões. Das (DAS et. al., 2005) desenvolveu uma heurística capaz de melhorar o *wirelength* com a adição de vias-3D. Mas isso não elimina o *tradeoff* existente entre essas duas soluções. Esse problema precisa ser investigado de forma mais aprofundada, pois os resultados podem ser muito úteis para o desenvolvimento de novos algoritmos e ferramentas CAD – Figura 3.5 (g).

Particionamento 3D para o empilhamento de *tiers*. Esse problema já foi explorado em pesquisas (SAWICKI, et. al., 2006) e foca a minimização de vias-3D com o uso de abordagens de particionamento. Os particionadores de hipergrafos dividem um *netlist* em diferentes grupos, com o objetivo de minimizar o corte entre eles.

Direcionando essa abordagem para o escopo do problema em circuitos 3D, cada partição é interligada por arestas e seu empilhamento segue um arranjo que diminui o número total de conexões entre as camadas – Figura 3.5 (e).

Minimizar o número de TSVs. Em conexões críticas do *netlist*, a TSV (via-3D) pode ser nociva para a performance do circuito (PAVLIDIS, V. F.; FRIEDMAN, E.G., 2009). Essa questão precisa ser estudada para que novos algoritmos sejam capazes de minimizar ou eliminar as vias-3D nesses caminhos.

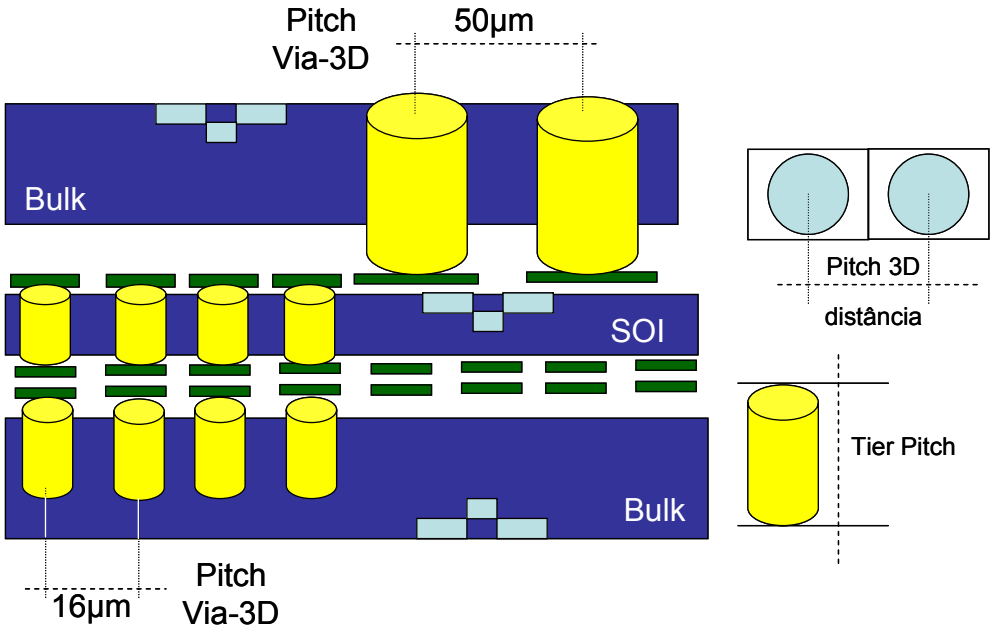


Figura 3.4: Vias *pitch* e *tiers pitch* em tecnologias SOI e *bulk*.

Na Figura 3.4, pode-se observar dois tipos de vias-3D que perfuram a camada ativa, *bulk* e SOI. Percebe-se que seus tamanhos também são diferentes (uma análise mais detalha dos tamanhos das vias-3D é apresentada na Tabela 3.1). Essa ilustração procura mostrar didaticamente a diferença entre um *pitch* via-3D e um *pitch tier*. No lado superior direito, aparecem duas vias grandes (*bulk*) posicionadas uma ao lado da outra. A distância lateral entre elas é calculada através da restrição *pitch* 3D (distância calculada através da metade da via-3D de origem até a metade da via ao lado). A restrição do pitch 3D também é ilustrada pela comparação com a tecnologia SOI, como pode-se ver no lado inferior esquerdo.

A *tier pitch* mede verticalmente o mesmo que a distância entre uma *tier*. Para cálculos de *wirelenght*, a distância vertical entre as vias tem de ser considerada.

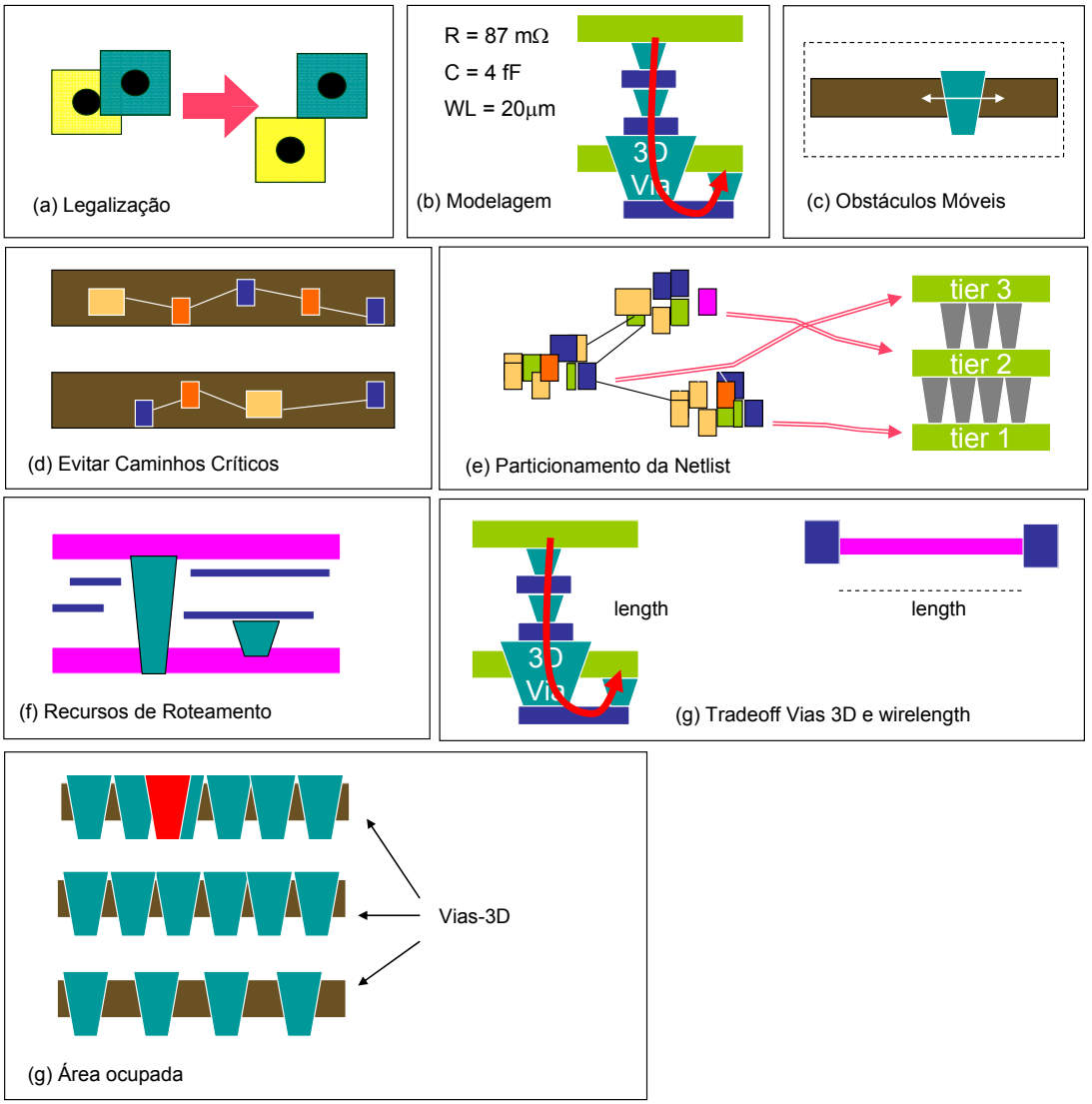


Figura 3.5: Questões relacionadas às vias-3D: (a) Legalização; (b) modelagem; (c) obstáculos móveis; (d) caminhos críticos; (e) particionamento; (f) recursos de roteamento; (h) área ocupada; (g) *tradeoff* (HENTSCHKE, 2007).

Na Tabela 3.1, pode-se observar que existe uma enorme variedade de *pitches*, alguns deles ocupando a área ativa. A melhor combinação para vias-3D seria o menor *pitch*, com o menor *tier pitch* e sem que a área ativa fosse ocupada. Contudo, essa combinação não é fácil de ser obtida. Abaixo segue uma pequena análise.

O aspecto mais importante a ser considerado é a distância entre as vias-3D, calculada através da variável *pitch*. Com um *pitch* 3D pequeno, é possível alojar mais vias de comunicação entre *tiers*, mas mais interessante ainda seria minimizar o número de vias entre as camadas. Outro objetivo importante é a diminuição das restrições para o posicionamento das vias e a possibilidade de melhorar o *wirelength* e a performance do projeto. Essas razões fazem com que a estratégia de integração *face-to-face* seja mais atrativa.

A distância vertical é computada pela restrição da *tier pitch*. Nesse caso, calcula-se a distância entre duas *tiers*.

Pode-se perceber na Tabela 3.1 que estratégias *face-to-face* para *vias-3D*, *contactless* e *microbump* não consomem área ativa. Essas estratégias são as melhores escolhas em relação às restrições de *pitches.* Contudo, limitadas a duas *tiers* (para novas conexões *face-to-face*, a estratégia deve ser combinada com a *back-to-back*, como visto na seção anterior). Por essa razão, MITLL e Tezzaron utilizam a integração *face-to-face* nas duas primeiras camadas e a *face-to-back* nas demais. Integrar várias *tiers* com a estratégia *face-to-face* e empilhá-las com integrações *back-to-back* poderia ser vantajoso, mas estas exigiriam muitas perfurações em áreas ativas, o que é um custo elevado (as duas *tiers* integradas sofreriam perfurações).

Com a integração *face-to-face*, as vias-3D não consomem área ativa e possuem o melhor *pitch*. Nesse sentido, é razoável esperar melhoras no *wirelength* e, consequentemente, menor atraso e potência, já que as conexões seriam mais curtas.

Tabela 3.1: Tecnologias para vias-3D

VIA-3D	**ESTRATÉGIA DE INTEGRAÇÃO**	***TIER PITCH***	***VIA-3D PITCH***
Tezzaron (*pads* de cobre)	*face-to-face*	16-20 µm	2.4 µm
Tezzaron	*face-to-face*	16-20 µm	1.46 µm
Microbump	*face-to-face*	16-20 µm	10-100 µm
Contactless (capacitivo)	*face-to-face*	16-20 µm	50-200 µm
MIT (Cobre)	*face-to-face*	16-20 µm	5 µm
TSV *face-to-face*	*face-to-face*	16-20 µm	0.5 µm
Tezzaron supervia	*face-to-back*	11-15 µm	60.8 µm
Tezzaron supercontato	*face-to-back*	11-15 µm	< 4 µm
Microbump 3D *Package*	*face-to-back*	11-15 µm	25-50 µm
Contactless Indutivo	*face-to-back*	11-15 µm	50-150 µm
MITLL via-3D (SOI)	*face-to-back*	9-12 µm	5 µm
Via-3D (*bulk*)	*face-to-back*	11-15 µm	50 µm
Back-to-back via-3D (SOI)	*back-to-back*	6-8 µm	15 µm

Analisando a Tabela 3.1 percebe-se que os níveis de integração *face-to-back* e *back-to-back* necessitam que a camada ativa seja perfurada para que a via-3D possa conectar as *tiers* adjacentes e não adjacentes. São elas: Tezzaron supervia, Tezzaron supercontato, *Contactless* Indutivo, MITLL via-3D (SOI), Via-3D (*Bulk*), *Back-to-back* via-3D.

Em alguns projetos, a IBM reduziu o *pitch face-to-face* para 0,2 x 0,2 μm^2 usando SOI *(Silicon on Insulator)* (YOUNG, 2005). Entretanto, devido à pequena dimensão de sua área, aconteceram problemas no alinhamento das camadas, prejudicando a confiabilidade do circuito (BERNSTEIN, 2006), (PAVLIDIS, V. F.; FRIEDMAN, E.G., 2009). Além disso, como dito anteriormente, as estratégias de integração *face-to-face* limitam-se a somente duas camadas.

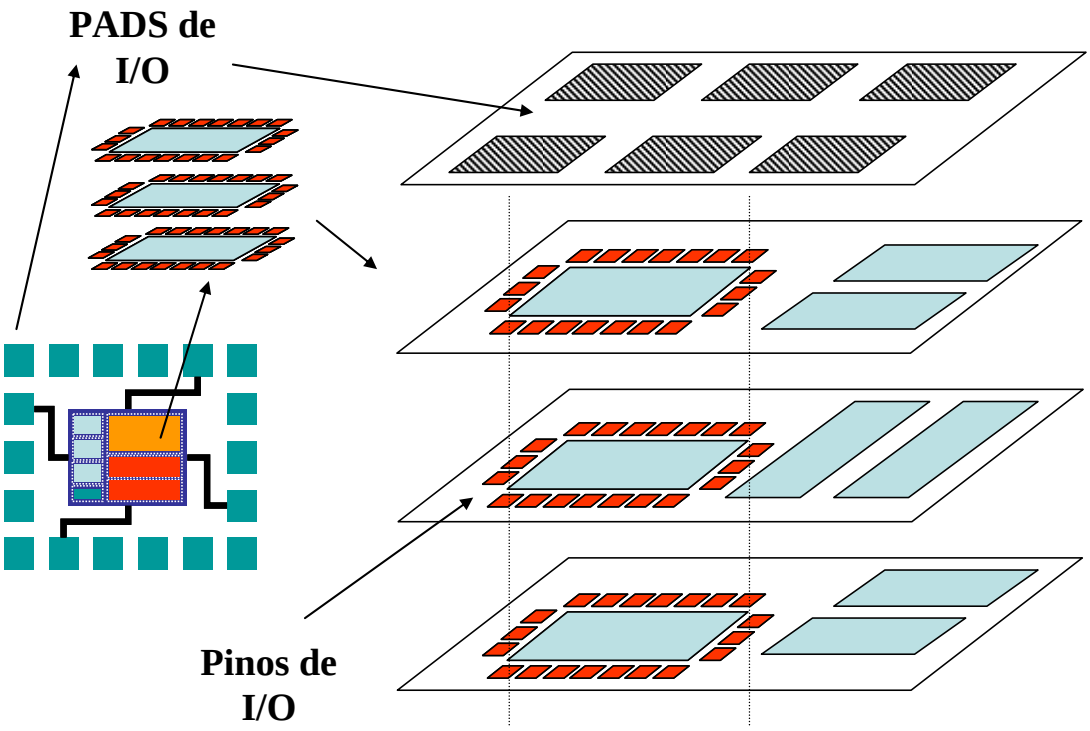


Figura 3.6: Migração de circuitos 2D para circuitos 3D com *pads* e pinos de I/O.

3.2.4 Projeto de pads e Pinos de I/O

Soluções de particionamento, posicionamento e *floorplaning* encontradas na literatura sobre circuitos 3D não descrevem o projeto de pinos e *pads* de I/O. Contudo, os resultados desta tese, apresentados no capítulo 5, mostram que a elaboração de tal projeto tem consequências diretas no particionamento e no posicionamento de células.

Esta seção descreve o projeto de *pads* de I/O em circuitos 3D e a sua relação com pinos de I/O. Perspectivas em *floorplaning* 2D e 3D e metodologias de projeto 3D são abordadas, juntamente com novas soluções relacionados ao particionamento e posicionamento de células e I/Os.

3.2.4.1 Projeto Topológico (Floorplaning)

A qualidade da solução do planejamento topológico de um circuito (planta baixa) influencia diretamente a dificuldade das etapas seguintes, podendo até inviabilizá-las. Quanto melhor for o *floorplanning*, menor será a área do leiaute final. Além disso, o desempenho e o consumo de potência do circuito são diretamente afetados pelas decisões tomadas na etapa de definição da planta baixa.

No planejamento topológico, são considerados **blocos duros** (REIS, 2003) aqueles desenvolvidos por um grupo de projetistas e que não podem ter sua estrutura interna, seu leiaute e sua orientação e posicionamento de pinos de I/O alterados. Esses blocos podem ser posicionados em qualquer lugar da área especificada. Outros tipos de blocos a serem considerados durante o *floorplaning* são os **blocos flexíveis**. Nesse caso, o planejamento topológico tem a liberdade de alterar a sua forma geométrica, a orientação e a posição dos pinos de I/O, como mostra a Figura 3.7. Um circuito integrado pode conter ambos os tipos de blocos, ficando a cargo do planejamento topológico o trabalho de encontrar o melhor arranjo.

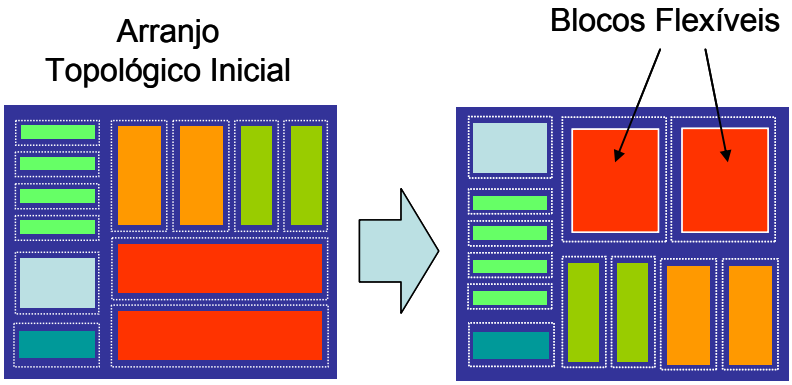
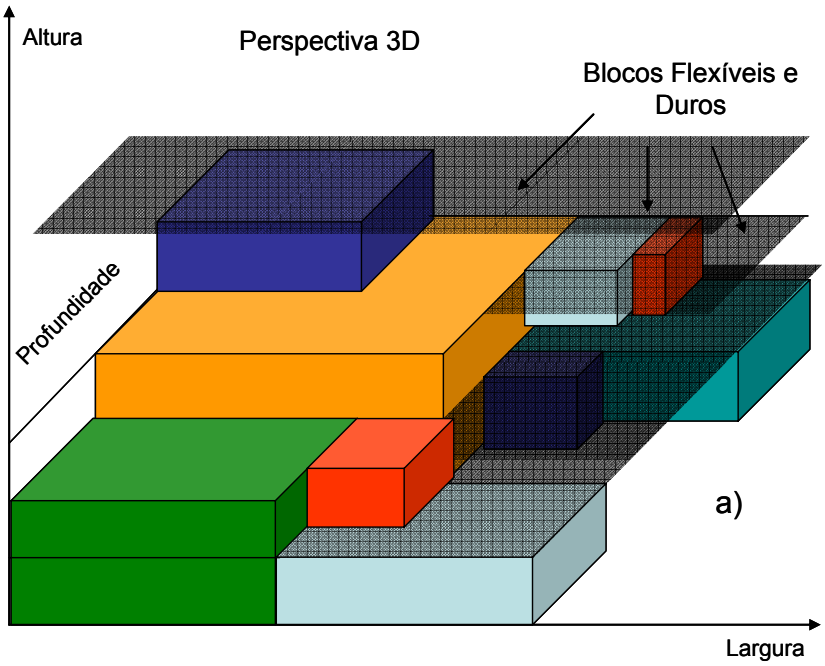


Figura 3.7: Planejamento topológico 2D.

No planejamento topológico 3D, em nível de *tiers*, os blocos são divididos conforme sua estrutura e funcionalidade (tipo de material, RF, memória, etc.). Com isso, seu nível de integração é baixo e a comunicação é realizada através de pinos de I/O entre blocos e *tiers*. Em nível de IP, os blocos rígidos são divididos entre as *tiers* de acordo com fatores compostos, como *wirelenght* e problemas térmicos. Já no nível de lógica aleatória, o planejamento topológico lida com blocos flexíveis, podendo, até mesmo, executar o remanejamento em três dimensões. Já que a comunicação entre *tiers* é realizada através de interconexões internas ao bloco (vias-3D), esse nível tem alta integração.

A Figura 3.8 (a) mostra o planejamento topológico 3D com blocos flexíveis e duros. Percebe-se que o *floorplaning* com blocos flexíveis permite o dimensionamento em *n* camadas (formando um cubo). A Figura 3.8 (b) ilustra uma visão lateral do *floorplaning*. Percebe-se que o projeto é realizado em cinco *tiers* e contém blocos flexíveis projetados em mais de uma camada. A Figura 3.8 (c) mostra a visão aérea do mesmo projeto topológico. A Figura 3.8 não ilustra a posição dos *pads* de I/O; esse tópico é discutido nas próximas seções deste trabalho.



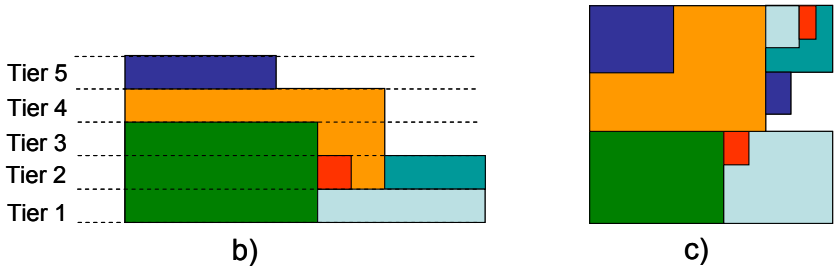


Figura 3.8: Planejamento topológico 3D. (a) *Floorplaning* 3D; (b) visão lateral do *floorplaning* 3D*;* (c) vista aérea do *floorplaning* 3D.

3.2.4.2 Pads de I/O

Numa perspectiva 2D, como ilustra a Figura 3.9, verifica-se que os *pads* de I/O podem ser blocos grandes (as dimensões são discutidas a seguir) com a função de realizar a comunicação do interior do *chip* com os elementos externos. Observa-se nessa figura que os *pads* estão localizados nas bordas do circuito. A informação de sua posição e orientação pode ser fornecida pelos projetistas ou organizada de acordo com o resultado do *floorplaning*.

No projeto de circuitos 3D, por sua vez, os *pads* de I/O podem ser posicionados de 4 maneiras: (1) distribuídos entre as bordas das *tiers* (BEYNE, 2006)*;* (2) posicionados na primeira e na última camada (PATTI, et. al., 2005); (3) posicionados somente na primeira camada (KAYA; BARKE, 2004) (DAVIS et.al., 2005); (4) posicionados somente na última camada (KAYA; BARKE, 2003) (DAVIS et. al., 2005). A Figura 3.10 ilustra a distribuição de *pads* de I/O entre as diferentes camadas do circuito. Percebe-se que os *chips* que formam a pirâmide têm tamanhos diferentes, o que facilita a posição dos *pads*.

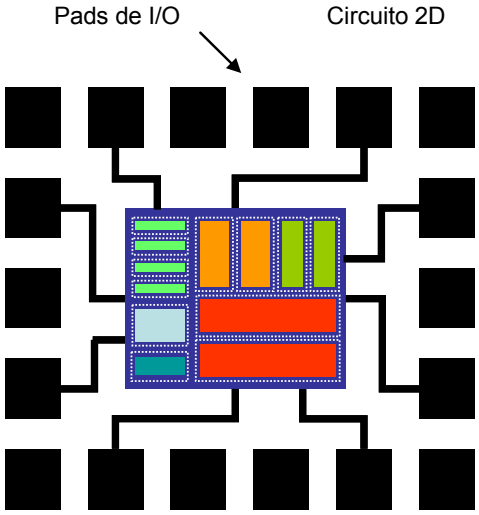


Figura 3.9: Circuito 2D com *pads* de I/O.

Os *pads* de I/O fazem a interface com o plano exterior. Por isso, suas posições devem ser de fácil acesso. Na Figura 3.11, pode-se observar variações dessas posições. Atualmente, a indústria (PATTI, et. al., 2006) posiciona os *pads* de I/O na primeira e na última camada do circuito. Essa metodologia facilita o roteamento dos *pads* até aos blocos e evita uma seqüência grande de perfurações. Outro motivo da divisão é a área

dos *pads*. Por causa desta e da distância separadora (*pitches*), muitas vezes eles precisam de duas *tiers*.

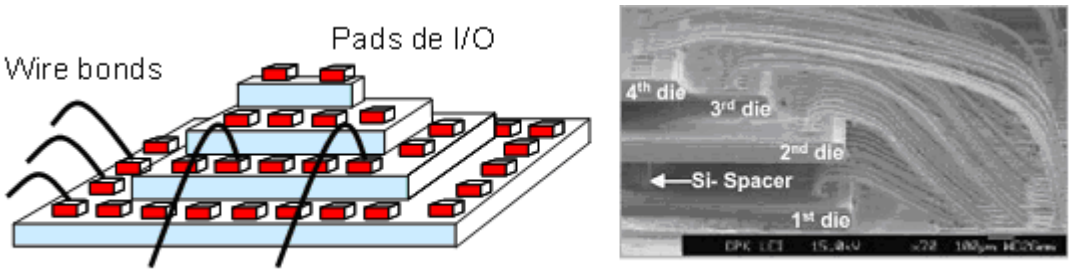


Figura 3.10: Empilhamento de *chips* e *wire bonds.*

As dimensões dos *pads* de I/O variam pouco de acordo com a tecnologia adotada. A Figura 3.12 mostra o *layout* de um *pad* de I/O MOSIS, do início da década de 1990, com tecnologia de 1,6µm e com dimensões de 200µm x 200µm. A distância mínima entre os *pads*, nesse caso, é de 200µm. Já em tecnologias mais recentes, como AMS de 0,35µm (Figura 3.13), a dimensão dos *pads* de I/O é de 102µm x 348µm, com distância entre elas de 150µm. Percebe-se que a influência da tecnologia nas dimensões dos *pads* não é tão significativa.

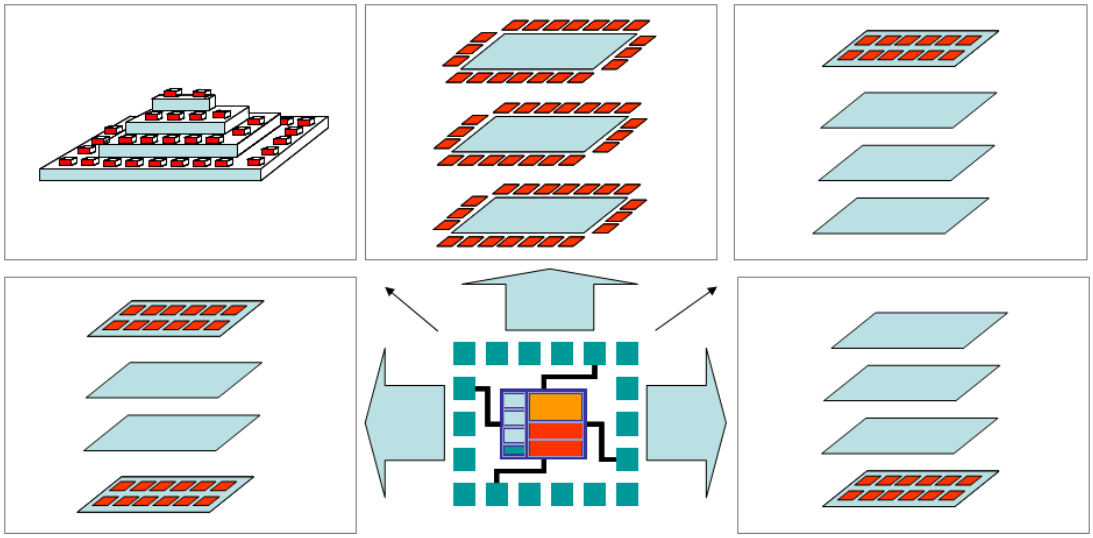


Figura 3.11: Variações de *pads* de I/O em circuitos 3D.

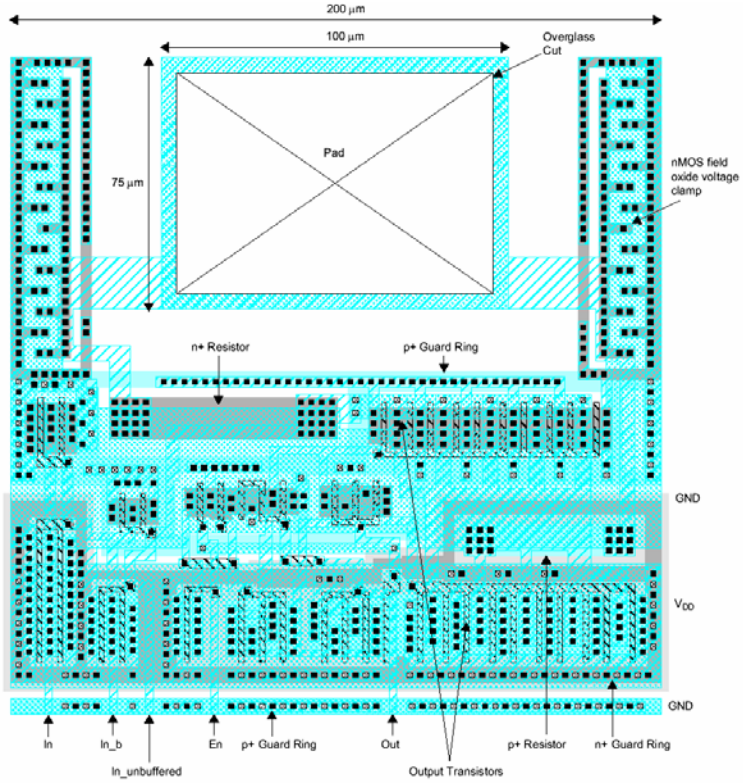


Figura 3.12: Leiaute MOSIS I/O, *pads* 1,6 µm.

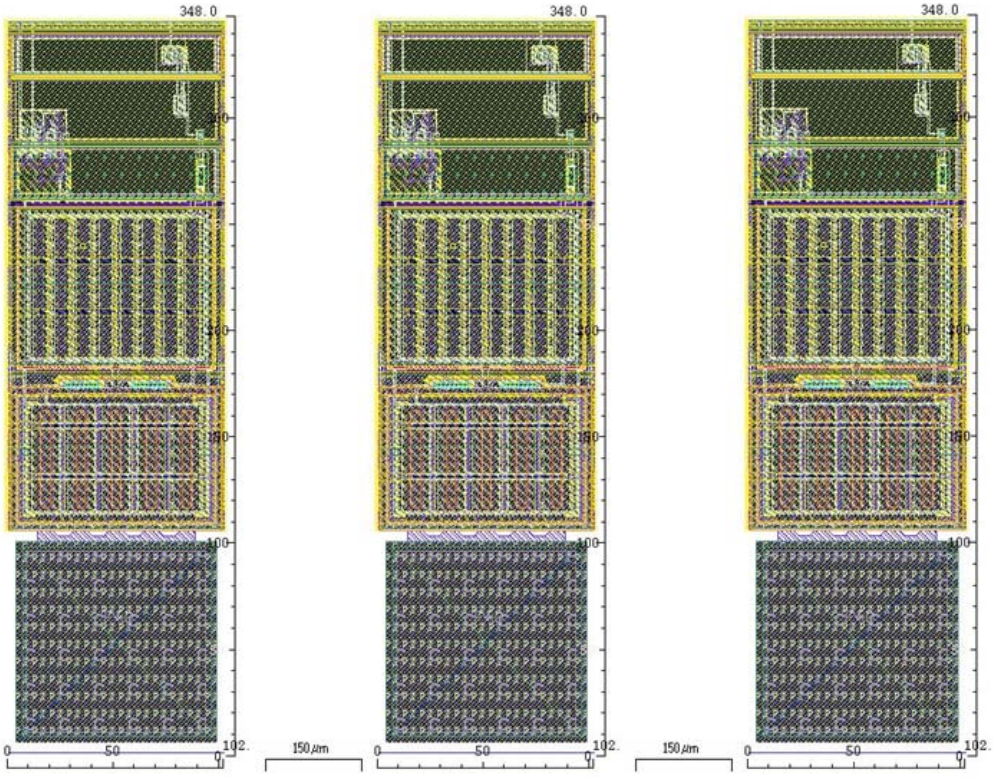


Figura 3.13: *Layout* AMS I/O, *pads* 0,35 µm.

3.2.4.3 Pinos de I/O

Sabe-se que existe uma hierarquia de pinos de I/O (por exemplo, pinos que representam entradas e saídas de uma célula e pinos que representam a entrada e saída de um bloco). Porém, nesta tese, tal hierarquia se resume a *pads* de I/O do circuito e pinos de I/O de blocos. Esta seção discute os pinos de I/O que representam as entradas e saídas de um bloco. Sua ligação pode conectar-se tanto a um pino de outro bloco quanto aos *pads* de I/O que são a saída do circuito.

O posicionamento de pinos de I/O em blocos é mais eficiente se executado durante o *floorplanning*. Porém, atualmente, em circuitos 3D, quase todos os projetos e ferramentas são direcionados às tecnologias 2D. É claro que uma técnica automática para migrar de uma tecnologia 2D para uma 3D pode reduzir significativamente o tempo de projeto. Embora alguns trabalhos (GOPLEN; SAPATNEKAR, 2005), (DENG; MALY, 2005) possam ter usado alguns critérios para fazer o particionamento de pinos de I/O, os detalhes de como esse processo foi realizado não estão referidos pelos autores. Assume-se, então, que eles adotam soluções simplistas. Conforme a pesquisa realizada para esta tese, este é o primeiro trabalho que estuda as consequencias, na solução final do projeto, do particionamento de pinos de I/O em duas ou mais *tiers*.

Os pinos de I/O, em blocos duros, já têm seu posicionamento e orientação definidos pelo grupo de projetistas. Além disso, esses blocos não podem ser redimensionados na etapa de planejamento topológico. Já no caso de blocos flexíveis, o posicionamento dos pinos pode, muitas vezes, ser alterado. Nesse ponto, o algoritmo proposto nesta tese possibilita a redução de vias-3D e da área do circuito com base em informações prévias a respeito da posição dos pinos (discutido no capitulo 4).

As dimensões dos pinos de I/O são muito menores se comparadas com as dos *pads* de I/O, como mostra a Figura 3.14. A dimensão do pino também varia de acordo com a tecnologia. No caso da AMS 0,35µm, ela pode chegar a 3µm x 3µm. Ao contrário dos *pads* de I/O, os pinos não ocupam muita área.

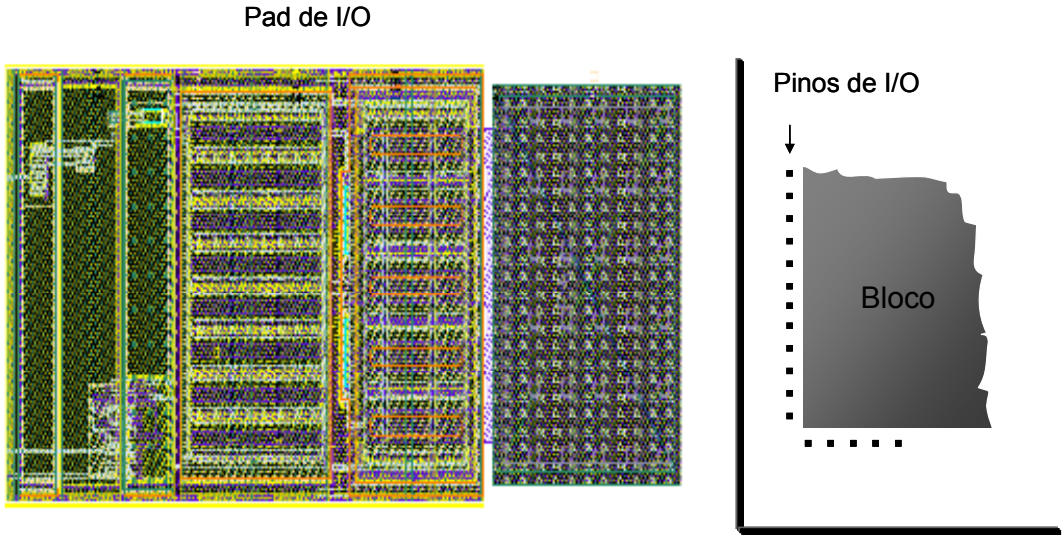


Figura 3.14: Relação entre um *pad* e um pino de I/O.

3.3 Objetivos e Metas do Trabalho

Esse trabalho visa estudar técnicas para a redução do número de vias-3D em circuitos 3D. Como mencionado na seção 3.2.3 é clara a influência das vias na área e no projeto do circuito. Os tópicos abaixo apresentam os objetivos deste trabalho:

- realizar um levantamento bibliográfico dos algoritmos de particionamento e posicionamento VLSI. Entender seu funcionamento e aplicação a fim de encontrar a melhor solução para a otimização de conexões verticais em circuitos 3D;
- efetuar análise experimental entre diferentes algoritmos que seguem o propósito de otimização de corte e vias-3D;
- melhorar o corte gerado pelo algoritmo hMetis quando aplicado para circuitos 3D. Para isso é proposta a utilização de informações da estrutura interna do circuito;
- estudar qual o melhor arranjo das partições em um grafo em uma dimensão;
- reduzir o número total de vias-3D através da redução do número de conexões que geram vias que atravessam mais de duas *tiers* adjacentes. Esse procedimento tem que distribuir as conexões de forma equilibrada para que não aumente o número total e máximo de vias. As vias longas devem migrar para camadas adjacentes, contudo, de forma balanceada para que não haja um número excessivo de migrações para um único par de *tiers*. Algoritmos com função de custo flexível, e que atuam de forma iterativa vão ser estudados;
- o balanceamento da área de células entre as *tiers* deve se manter equilibrado a fim de não prejudicar a redução da área total do circuito. Para isso, as funções de balanceamentos serão bastante rígidas;
- características dos circuitos 2D como, percentual de espaços em branco, relação de aspecto e orientação dos pinos devem ser preservados;

Os tópicos descritos representam os objetivos gerais do trabalho. No próximo capítulo, faz-se um levantamento bibliográfico dos algoritmos de particionamento e posicionamento, sob os focos 2D e 3D, para compreender suas atuações no tratamento das vias-3D.

4 ALGORITMOS DE PARTICIONAMENTO E POSICIONAMENTO E SUAS MUDANÇAS PARA O PROJETO DE CIRCUITOS VLSI 3D

Há décadas estudam-se algoritmos de particionamento e posicionamento de células para circuitos planares (2D). Com o surgimento de circuitos em três dimensões, muitos desses algoritmos foram adaptados para o novo paradigma de projeto (3D). Este capítulo discute o impacto causado por esta mudança nas etapas de particionameto e posicionamento. Além disso, apresenta uma revisão bibliográfica dos algoritmos clássicos, suas classificações e evoluções.

4.1 Particionamento

O particionamento de um sistema faz-se necessário quando o tamanho do sistema é demasiadamente grande para ser projetado de forma global, podendo ser usado de maneira hierárquica até que cada subsistema criado tenha um tamanho aceitável (gerenciável). O particionamento é uma técnica encontrada em várias áreas. Por exemplo, no projeto de algoritmos, a abordagem de "dividir para conquistar" é rotineiramente usada para particionar problemas complexos em subsistemas de menor complexidade. Essa técnica consiste em dividir sucessivamente o problema em problemas menores até que se encontre uma solução simples, de forma que a combinação dessas soluções parciais forme a solução completa do problema.

No domínio VLSI, o particionamento tem um papel fundamental no projeto de computadores, em geral, e de *chips*, em particular. Por exemplo, o processo de desenvolvimento de um sistema computacional composto por milhões de transistores pode ser particionado em vários módulos ou blocos menores. Após, cada bloco utiliza seus terminais de entrada e saída, localizados na periferia, e a especificação das conexões para realizar a comunicação entre os diferentes módulos.

Em circuitos planares, o particionamento muitas vezes auxilia o trabalho dos posicionadores, como indicam as pesquisas de Agnihoti, Ono e Madden (AGNIHOTRI; ONO; MADDEN, 2005) e as de Roy e colaboradores (ROY et. al., 2005).

Em circuitos 3D, o particionamento serve como uma técnica de auxílio à divisão de células e blocos entre as diferentes camadas. Diversos trabalhos (ABABEI et. al., 2005), (DAVIS et. al., 2005), (DENG; MALY, 2001) executam *min-cut* utilizando a ferramenta hMetis (KARYPIS et. al., 1999) para atribuir células para *tiers*, minimizando o número de vias-3D entre as camadas. Pode-se ainda usar uma sequência de novos particionamentos em cada uma das *tiers*, o que auxilia o posicionamento e reduz o tamanho das conexões. Contudo, vários autores (KAYA et. al., 2004), (LIU et.

al., 2005), (DAS; CHANDRAKASAN; REIF, 2003) observaram que essa abordagem pode causar um *wirelength* maior.

4.2 Formulação do Problema de Particionamento

O problema de particionamento pode ser expresso com clareza na teoria dos grafos. Seja *G* um grafo *G=(V,E)*, onde *V* é um conjunto de vértices, e *E*, um conjunto de arcos, sendo que cada elemento *V* é uma tupla (*a,b*) onde *a* e *b* são vértices, ou seja, cada arco liga dois e somente dois vértices. Agora, seja *H* um hipergrafo *H=(A,V)* onde *A* é um conjunto de hiperarcos e *V* um conjunto de vértices. Cada hiperarco consiste em um subconjunto *N* de vértices distintos com $|N_i| \geq 2$.

Um particionamento de um hipergrafo *H* é um conjunto de subconjuntos disjuntos e não-vazios de $N = \{Ni,...,Nr\}$, tal que $U_{i=1}^{r} N_i = N$ e $N_i \cap N_j = 0$ para $i \neq j$.

O particionamento em qualquer nível ou estilo de projeto trabalha com alguns parâmetros:

Interconexão entre partições: O número de interconexões, em qualquer nível, tem de ser minimizado. Reduzir as conexões não diminui somente o atraso, mas também a interface entre as partições, permitindo a independência de projeto e fabricação. Grandes números de interconexões aumentam a área total do projeto e dificultam a tarefa dos algoritmos de posicionamento e roteamento. A minimização desse número é chamada de *min-cut* e é um objetivo funcional muito importante dos algoritmos de particionamento.

Atraso devido ao particionamento: O particionamento de um circuito pode fazer com que um caminho crítico cruze as partições várias vezes. Como o atraso entre as partições é relativamente grande, em comparação ao provocado por um caminho interno a uma partição, ele é um fator importante e deve ser considerado sempre que circuitos que exigem alta performance são particionados.

Número de terminais: Os algoritmos de particionamento podem ser utilizados quando o número de redes necessárias para conectar um subcircuito a outro não exceder o número de terminais. No caso do particionamento de blocos, esse número é determinado pelo perímetro da área usada por ele.

Área de cada partição: Quanto menor a granularidade, maiores as preocupações com o tamanho da área, o que exige cuidados com o balanceamento.

Número de partições: O número de partições é determinado, em parte, pela qualidade do algoritmo de particionamento.

4.3 Classificação dos Algoritmos de Particionamento

Os algoritmos de particionamento podem ser classificados em relação *à existência ou não de uma partição inicial*. Caso esta não exista, os algoritmos são ditos *construtivos*, pois abastecem uma partição a partir da estrutura de interconexões do circuito. Os algoritmos *iterativos* recebem um conjunto de partições já realizadas e alteram as partições do mesmo *netlist* para melhorar algum critério imposto pelo particionamento.

Outra classificação enquadra os algoritmos de particionamento como *determinísticos* ou *probabilísticos*. Nesse caso, os primeiros são os que apresentam sempre a mesma solução para um mesmo conjunto de entradas, e os segundos (também chamados de combinatórios), os que utilizam opções aleatórias como técnica de variação para explorar as opções de otimização.

Segundo a técnica, os algoritmos de particionamento podem ser descritos como de *migração de grupos*, utilizando duas células, nos quais, iterativamente, uma célula de cada partição é escolhida para trocar de grupo (KERNIGHAN; LIN, 1970), ou utilizando uma célula; nesse caso, apenas uma é escolhida (FIDUCCIA; MATTHEYSES, 1982). Há ainda a técnica de *replicação de componentes*, em que não há movimentações de células, mas uma replicação das que estejam prejudicando o corte (MAK, 2002), e a ténica de *crescimento de aglomerados*, um método construtivo para criar dois ou mais grupos de células (KARYPIS, et. al, 1997).

O critério usado para avaliar a qualidade do particionamento pode ser uma combinação dos seguintes itens: corte mínimo; mínimo número de cortes entre as partições; atraso em redes críticas, que pode ser muito maior para as conexões entre as partições, ou por um caminho crítico cruzar várias vezes a mesma partição; área de cada partição, também restrita por uma unidade ou custo e pelo número de partições.

4.4 Revisão dos Algoritmos de Particionamento

Em geral, os algoritmos de particionamento têm como objetivo a distribuição das células por duas ou mais partições, de forma que exista equilíbrio das áreas e que sejam minimizadas as conexões que cruzam as partições. Abaixo segue a descrição de alguns algoritmos de particionamento, em especial, Kerninghan-Lin, um dos pioneiros em ferramentas de CAD, e Fidducia-Matheyses, utilizado até os dias de hoje por ser uma heurística rápida e de bons resultados.

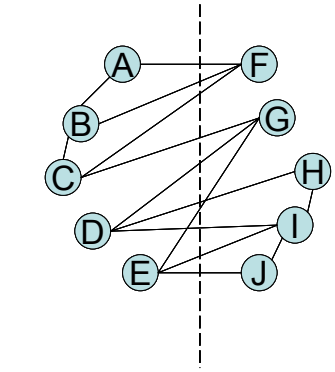
4.4.1 Algoritmo de Kernighan-Lin (KL)

Baseado em migração de grupos, o algoritmo de Kernighan-Lin utiliza como entrada de um grafo *G=(V,E)*, em que cada vértice *v* está localizado em uma de duas partições. A cada iteração do algoritmo, duas células são selecionadas para que o corte seja diminuído. Calcula-se duas funções para cada célula: *inedge(cell) e outedge(cell)*. A primeira retorna o número de redes da célula *(cell*) que não cruzam as partições. Já a segunda retorna o número de redes que as cruzam (KERNIGHAN; LIN, 1970). Com o resultado de ambas as funções, *inedge(cell) e outedge(cell)*, é calculado o ganho *G (cell)*, conforme mostra a equação abaixo:

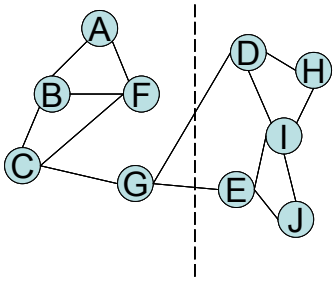
$$G(cell) = inedge(cell) - outedge(cell)$$

Logo após o cálculo do ganho *G* de cada célula, é realizada a troca das duas células de maior ganho de cada partição. O término do processo é efetuado quando não há mais trocas que possam diminuir o corte.

O algoritmo de Kernighan-Lin apresenta uma série de desvantagens. Sua ordem de complexidade é cúbica $O(n^3)$, pois não possui um mecanismo eficaz de atualização dos ganhos. Assim, a cada iteração, o ganho de todas as células é recalculado. Outra desvantagem se deve ao fato de não conseguir balancear as partições por área, o que não ocorreria se as células tivessem tamanhos iguais. Com o uso desse algoritmo, o balanceamento ficaria restrito ao número de células.



nodos	inedge	outedge	G
A	1	1	0
B	2	1	1
C	1	2	-1
D	0	3	3
E	0	3	-3
F	0	3	-3
G	0	3	-3
H	1	1	0
I	2	2	0
J	1	1	0



nodos	inedge	outedge	G
A	2	2	0
B	3	0	3
C	3	0	3
D	2	1	1
E	2	1	1
F	3	0	3
G	1	2	-1
H	2	0	2
I	4	0	4
J	2	0	2

Figura 4.1: Kernighan-Lin após troca de dois pares de vértices.

A Figura 4.1 apresenta o resultado de duas trocas. A primeira, entre os vértices (F,D), e a segunda, entre os vértices (G,E). Cada par é fixado para que não seja mais selecionado até que não existam mais vértices livres. O algoritmo se repete até que, ao final de uma iteração, não haja mais redução do tamanho do corte entre as partições. Percebe-se que o ganho alcançado no número de redes que cruzam a partição após as trocas não foi igual a $G(a) + G(b)$.

A Algoritmo 4.1: Pseudocódigo do algoritmo de Kernighan-Lin fornece o pseudocódigo do algoritmo de Kernighan-Lin.

Algoritmo de Kernighan-Lin

Criar um particionamento inicial.
  Enquanto a solução estiver melhorando.
    Enquanto existir vértices trancando.
      Escolher a ε A e b ε B, sendo que a e b não estão bloqueados e $G(a) + G(b)$ é máximo.
      Trocar a por b, onde (a ε B e b ε A).
      Bloquear a e b.
  Avaliar resultado atingido na iteração anterior.
Liberar todos os vértices.

$O(n^3)$

Algoritmo 4.1: Pseudocódigo do algoritmo de Kernighan-Lin

4.4.2 Algoritmo de Fiduccia-Matteyses (FM)

Uma modificação do algoritmo de Kernighan-Lin foi proposta por Fiduccia e Mattheyses (FIDUCCIA;MATHEYSES, 1982). O novo algoritmo faz migração simples de grupos (uma célula) e apresenta uma complexidade de tempo inferior a (On^2). Talvez a modificação mais importante em relação a KL esteja no corte das redes. Para FM, não interessa quantas células de uma rede existem em cada partição, mas, sim, se a rede atravessa a partição.

A Figura 4.2 ilustra a comparação entre o corte de redes e o tradicional. Observa-se que há uma rede formada por quatro pinos. Três deles encontram-se em um determinado conjunto (X), e o quarto pino localiza-se no conjunto (Y). No corte tradicional, três redes cruzam as partições, resultando em um corte = 3. No corte de redes, somente uma delas cruza as partições, com isso, tem-se corte = 1, o que resulta numa maior facilidade de cálculo, permitindo assim o aumento da performance. Outro ponto importante é a semelhança dessa estratégia com os circuitos reais.

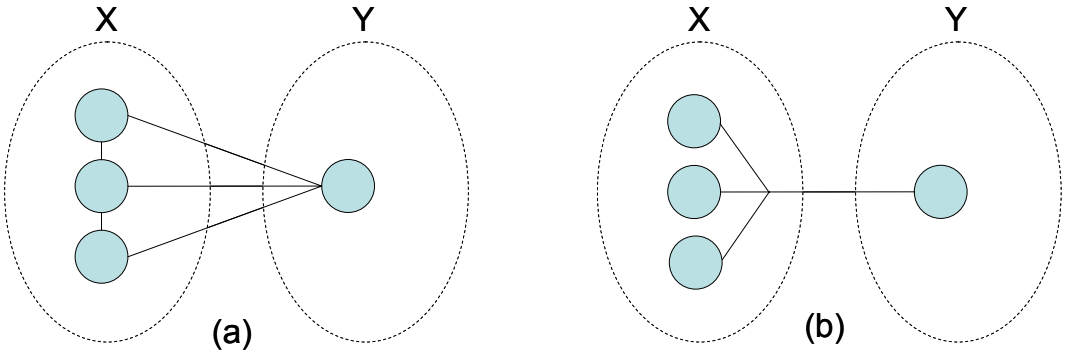


Figura 4.2: (a) Corte tradicional; (b) corte de redes.

Para entender de forma mais clara o cálculo do ganho, deve-se introduzir alguns conceitos. Por exemplo, o *estado de corte* de uma rede refere-se a seu possível cruzamento pelas partições. Uma *rede crítica* é aquela que pode ter seu estado de corte alterado pelo movimento de uma célula que lhe pertença (HENTSCHKE; REIS, 2002). Já uma *célula crítica* é aquela responsável pela criação de uma nova rede crítica.

A vantagem mais significativa do cálculo de ganho Fidducia-Matheyses é o não atrelamento com as células, mas com as redes. Calcula-se, então, o ganho com base em dois contadores (*cont1 e cont2*), localizados em cada uma das redes, que se referem à quantidade de células existentes em cada partição. Por exemplo, na figura Figura 4.2 (a), tais contadores valem 3 e 1.

Outra modificação importante é a possibilidade do balanceamento das partições por área. Isso acontece porque FM possibilita a troca individual de células e, também, exige que seja desenvolvida uma função de avaliação que controle a migração das mesmas com base em um critério de balanceamento. Caso a movimentação não cumpra tal critério, não será considerada. A função faz com que o algoritmo não junte todas as células em uma mesma partição.

O algoritmo de Fidducia-Matheyses é bantante complexo e exige uma análise mais detalhada. Na Algoritmo 4.2: Pseudocódigo do algoritmo de Fidducia-Matheyses

, ele é escrito em pseudoliguagem e analisado passo a passo.

Algoritmo de Fidducia-Matheyses	
Passo 1	Computar o ganho de todas as células na *netlist*.
Passo 2	i = 1; selecionar uma célula base. Caso não haja, encerrar o algoritmo.
Passo 3	Bloquear a célula base. Atualizar os ganhos.
Passo 4	Caso não haja células bloqueadas, pegar nova célula base. Caso haja, i++ e ir para passo 3.
Passo 5	Selecionar a melhor sequência de movimentos m_1, m_2, m3,..., m_r ($1 \leq r \leq i$), no qual o somatório de $m_1+m_2+m_3$,.. + m_r seja máximo.
Passo 6	Tornar os movimentos m_1, m_2, m3,..., m_r permanentes.
	Desbloquear todas as células.
	Retornar para o passo 1.
$O(n^2)$	

Algoritmo 4.2: Pseudocódigo do algoritmo de Fidducia-Matheyses

O algoritmo executa dois laços de repetição. No laço externo, é computado o valor do ganho de todas as células. No interno, o algoritmo executa uma série de movimentos sem repetir as células e sem recalcular o custo. O laço interno é executado até que sejam trocadas todas as células livres que não prejudiquem o balanceamento. As trocas são efetivadas nesse passo. No laço externo, o algoritmo escolhe um número *k* de movimentos. Assim, o laço externo é repetido enquanto $k \geq 1$, ou seja, enquanto o laço interno conseguir encontrar células para movimentação.

Passo 1: Antes de computar o ganho, os contadores *cont_A* e *cont_B* de cada uma das redes devem ser atualizados. A partição *A* refere-se ao contador *cont_A*. Inicialmente cada rede executa uma busca a fim de contar o número de células que estão conectas a ela, em cada uma das partições.

Seguindo o algoritmo, o ganho é calculado conforme as relações seguintes:

- *S* refere-se ao conjunto de origem da célula;
- B refere-se ao conjunto destino da célula;
- *S(n)* obtém o número de células da rede *n* que estão no conjunto origem;
- *B(n)* obtém o número de células da rede *n* que estão no conjunto destino.

```
G = 0; //ganho
Para cada rede n da célula
          Se S(n) == 1 então G = G + 1;
          Se B(n) == 0 então G = G - 1;
```

O pseudoalgoritmo descrito acima realiza dois testes para descobrir se a célula em questão é crítica. Somente as críticas têm valores de ganho diferentes. Caso o algoritmo considere a existência de *pads*, a célula pode deixar de ser crítica.

Passo 2: No laço interno do algoritmo, a variável *i* controla o número de movimentos realizados. Após, o algoritmo escolhe a célula que tenha o maior ganho sem criar desbalanceamento. O valor desse ganho pode ser negativo, possibilitando ao

algoritmo evitar os mínimos locais. Já o critério de balanceamento não permite que o algoritmo mova todas as células para uma partição. Isso pode ser expresso pela seguinte equação:

$$U \times |C| - mc \leq |N| \leq r \times |C| + mc$$

Nela,

- *C* é o conjunto de todas as células;
- |C| é a área do conjunto C;
- N é o conjunto ao qual a célula pertence;
- U é o coeficiente de balanceamento calculado por |N| / |C|;
- *mc* é o tamanho da maior célula da *netlist*.

Passo 3: Inicialmente, fixa-se a célula-base escolhida para mudar de partição com o intuito de que ela não mova novamente.

Passo 4: Procura-se por células livres. Caso exista alguma, procura-se uma delas que satisfaça ao critério de balanceamento. Se esta for encontrada, a variável *i* é incrementada e o algoritmo retorna para o passo 3. Se não, ele segue para o passo 5.

Passo 5: São escolhidas as *k* iterações, em que $k < i$ do laço interno que serão realizadas. Com isso, o algoritmo precisa somente armazenar o ganho de cada iteração.

Passo 6: Nessa etapa, todos os movimentos menores ou iguais a *k* devem ser efetivados. Após, todas as células são liberadas e o algoritmo retorna ao passo 1.

Atualmente, o algoritmo de Fiduccia-Matteyses continua sendo o mais utilizado. Contudo, muitos outros trabalhos aprimoraram seu processo (WANG; LIM; CONG; SARRAFZADEH, 2000), (KARYPIS; AGGARWAL; KUMAR; SHEKHAR., 1999), (CALDWELL; KAHNG; KENNINGS; MARKOV, 1999), (LEE, Y.; KIM; HUANG, G.; BAKIR, M.; JOSHI, T.; LIM, S., 2009).

4.4.3 Algoritmo de Goldberg

Estudos realizados por Goldberg (GOLDBERG, et. al., 1983) constataram que a qualidade dos algoritmos de particionamento por migração de grupos depende do número médio de arestas por nodo (*fan-out* médio). Kernighan-Lin, por exemplo, tem um bom aproveitamento se a média for superior a 5. Por outro lado, Goldberg observou que, em circuitos, a média gira em torno de 1,8 e 2,5. Com base nessa observação, desenvolveu uma técnica para unir vértices, aumentando sua conectividade (JOHANN; REIS, 2000). O algoritmo cria grupos de células contidas em um único elemento. Com essa técnica, Fiduccia-Mattheyses Kernighan-Lin são melhor aproveitados, particularmente graças à redução do número de elementos a serem tratados pelo algoritmo de particionamento.

4.4.4 Replicação de Componentes

Uma técnica bastante conhecida é a de replicação de componentes (SHERWANI, 1998). Caso uma determinada célula seja requisitada em ambas as partições, ela pode ser replicada, com o objetivo de diminuir o corte. Contudo, tal técnica deve ser utilizada com cautela, pois a replicação de muitos vértices pode ocasionar aumento de consumo e área do circuito. Já o trabalho de Wai-Kei Mak (MAK, 2002) utiliza uma variação,

chamada de *replicação funcional*. Segundo esse pesquisador, tal técnica pode levar a resultados de corte melhores que a replicação tradicional, por considerar a dependência lógica de diferentes sinais de saída de uma porta em sua entrada.

4.4.5 Particionamento Multi-Partições

A maioria dos algoritmos de particionamento trabalha somente com duas partições. Contudo, quando o número de elementos aumenta, apenas biparticioná-los não é suficiente para reduzir a complexidade do problema.

Existem muitas adaptações dos algoritmos de biparticionamento que trabalham com *n* partições. Um exemplo é a extensão do algoritmo de Fiduccia-Matheyses realizada por Karypis e Kumar (KARYPIS; KUMAR, 1998a). Abaixo são apresentadas três dessas novas abordagens.

A Figura 4.3 ilustra o primeiro método, uma extensão direta de Fiduccia-Matheyses. Cada um dos nodos selecionados pode se mover para qualquer uma das partições. Com isso, o valor do ganho é computado a cada movimentação, sendo escolhida a possibilidade que garanta o melhor resultado A desvantagem desse método é que ele necessita de bastante memória, a fim de armazenar todos os ganhos.

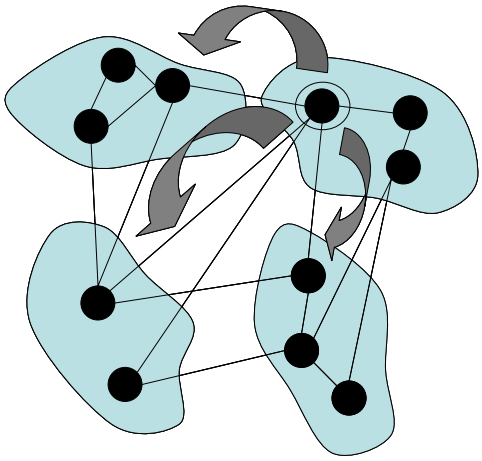


Figura 4.3: Extensão direta de Fidducia-Matheyses.

A Figura 4.4 mostra a segunda abordagem, conhecida como iterativa. O algoritmo inicia com uma solução com *n* partições. Após, duas destas são selecionadas, um biparticionamento é realizado, e as redes que cortam as partições são minimizadas.

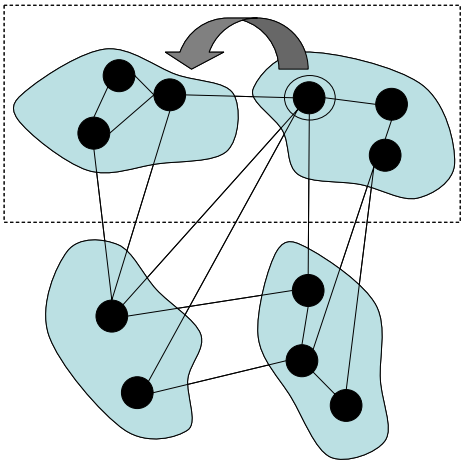


Figura 4.4: Abordagem com biparticionamentos.

O terceiro método é conhecido como hierárquico e construtivo. Inicia com uma partição e a biparticiona. Após, executa novos biparticionamentos com as duas partições criadas. Segue os mesmos passos até criar um grafo com *n* partições hierarquicamente. Essa técnica está ilustrada na Figura 4.5.

Analisando-se os três métodos, verifica-se que o primeiro, por necessitar de bastante memória para armazenar todos os ganhos, precisa de mais tempo de busca para calcular e percorrer todos os valores. De fato, não se trata de uma boa estratégia. Segundo Wang e Sarrafzadeh (WANG; LIM; CONG; SARRAFZADEH, 2000), tal procedimento pode ser 50% mais lento em comparação com o método hierárquico, que, conforme o autor, aceita somente resultados melhores que os anteriores. Além disso, experimentos mostram que a abordagem hierárquica produz resultados 7% melhores que o método iterativo.

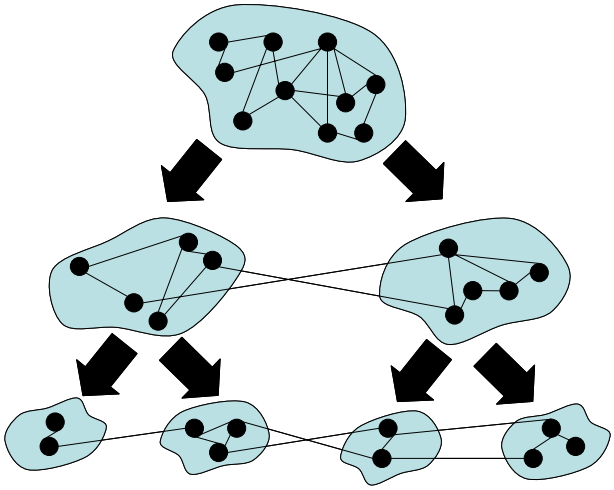


Figura 4.5: Abordagem hierárquica.

4.4.5.1 hMetis (Particionamento de Hipergrafos Multinível)

O particionamento de hipergrafos é um problema conhecido da teoria dos grafos e tem inúmeras aplicações em muitas áreas, incluindo o projeto VLSI (ALPERT, et. al., 1995), as distribuições de conteúdos em banco de dados (LIU, D.; SHEKHAR, S., 1996), o gerenciamento de transporte e o *datamining* (HAN, et. al., 1997). Ele consiste

em particionar vértices de um hipergrafo em *k* partes aproximadamente iguais, tal que o número de hiper-redes que conectam os vértices em diferentes partições seja minimizado. Um hipergrafo é uma generalização de um grafo em que o conjunto de redes é conhecido como hiper-redes, o que estende a noção de rede por permitir que mais de dois vértices sejam conectados.

Na área de projeto VLSI, hMetis é um *software* amplamente utilizado para particionamento de circuitos. Seus algoritmos são baseado em particionamento de hipergrafos com multinível (KARYPIS, et. al., 1998), (KARYPIS, et. al., 1997). São uma extensão dos grafos descritos nos estudos de Karypis e colaboradores (KARYPIS, et. al, 1998a), (KARYPIS, et.al., 1998b). Para esses autores, os algoritmos de particionamento de grafos tradicionais computam a partição diretamente no grafo original, como ilustra a Figura 4.6 (a). Conforme os estudos citados, tais algoritmos são, em sua grande maioria, lentos ou produzem resultados de pouca qualidade. Por outro lado, algoritmos de particionamento em multinível trabalham com uma abordagem diferente. Como ilustrado na Figura 4.6 (b), reduzem o tamanho do grafo (ou hipergrafo) aglutinando seus vértices e arestas (etapa de compactação) em partições de grafos menores. Após, é executada a etapa de descompactação, reconstituindo a partição para o grafo original. O *software* hMetis reduz o corte entre as partições durante a etapa de descompactação do hipergrafo através de um processo de refinamento.

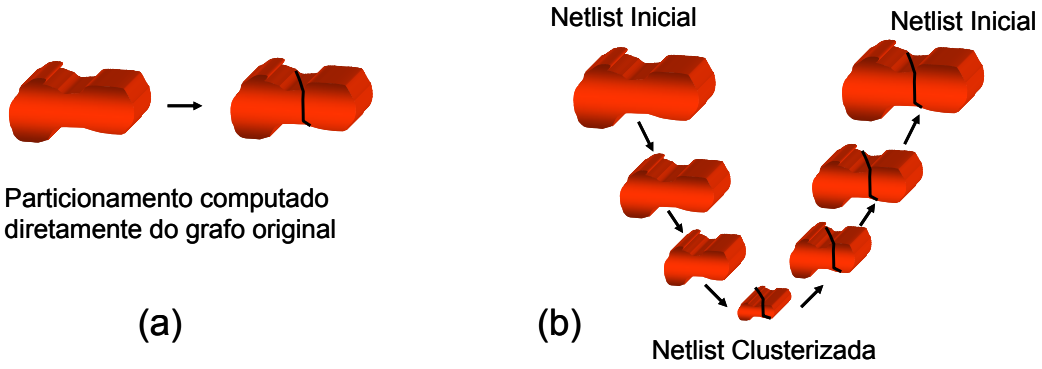


Figura 4.6: Estratégia multinível (HMETIS HOMEPAGE, 2009).

Abaixo, apresenta-se um resumo das etapas realizadas pelo algoritmo multinível. Mais detalhes podem ser encontrados nos estudos de Karypis (KARYPIS, 1997).

Fase de compactação: Durante a fase de compactação do hipergrafo, uma sequência de hipergrafos menores é criada. O objetivo dessa fase é criá-los tal que uma boa bisseção do hipergrafo pequeno não seja significativamente pior do que a obtida diretamente do hipergrafo original. Além disso, a fase de compactação também ajuda na redução das hiper-redes. Após vários níveis de compactação, grandes hiper-redes são contraídas para que se obtenha novas com apenas poucos vértices. Essa é uma particularidade útil, uma vez que heurísticas baseadas em Kernighan-Lin e Fiduccia-Matheyses (KERNIGHAN-LIN, 1970), (FIDDUCIA-MATHEYSES, 1982), ainda que muito eficientes no refinamento de pequenos grafos, são ineficazes para o refinamento de hiper-redes com grande número de vértices que migrem para direfentes partições. O grupo de vértices contraído para formar um vértice simples pode passar por diferentes caminhos.

Fase do particionamento inicial: Durante essa fase, é computada uma bisseção do hipergrafo compactado. Este tem poucos vértices (usualmente, menos que 100). Diferentes algoritmos podem ser usados sem grandes mudanças no tempo de execução médio. Essa fase utiliza o algoritmo Fidducia-Matheyses para o refinamento.

Fase de descompactação e refinamento: Durante a fase de descompactação, o particionamento do hipergrafo compactado é utilizado para obter o resultado do hipergrafo final. Isso é realizado através de sucessivos refinamentos com algoritmos para a redução de corte. À medida que a redução do biparticionamento termina, os vértices começam a ser descompactados, aumentando o hipergrafo até que chegue ao tamanho original já particionado.

Refinamento em *v*-ciclos: O objetivo dessa fase é o uso do paradigma de multinível para melhorar a qualidade da bisseção. O algoritmo de refinamento em *v*-ciclos consiste em duas fases: compactação e descompactação. A primeira preserva o particionamento inicial como entrada do algoritmo. Nesse esquema, os grupos de vértices são combinados para formar os grupos de um grafo compactado, correspondente aos vértices que pertencem somente a uma de duas partições. Em consequência, a bisseção original é preservada na saída do processo de compactação e inicia-se o primeiro particionamento, executando-se o refinamento durante a fase de descompactação. A fase de descompactação do refinamento em *v*-ciclos é idêntica ao particionamento de hipergrafos multinível, descrito anteriormente.

4.4.5.2 Posicionamento baseado em Particionamento

Essa técnica utiliza o particionamento como um auxíliar durante o posicionamento. Contudo, para essa estratégia, o conceito de particionamento torna-se mais impreciso, pois, ao contrário do que se estuda na teoria dos grafos, esse tipo de algoritmo conhece a localização das partições. O particionamento é aplicado na resolução de problemas de posicionamento. No uso dessa técnica, a estratégia de corte é um problema muito interessante que pode modificar significativamente o resultado do posicionamento, como observa Yildiz (YILDIZ, 2001). Abaixo são listadas algumas estratégias que podem ser aplicadas a circuitos:

São conhecidas na literatura (SHERWANI, 1998) as seguintes estratégias de corte: (a) orientada a corte; (b) quadratura; (c) bisseção hierárquica em bandas; (d) bisseção em fatias.

Atualmente, as estratégias mais utilizadas pelos posicionadores baseados em particionamento são a quadratura e a bisseção hierárquica em bandas. Tais estratégias realizam bipartições de tamanhos iguais, o que facilita o particionamento.

Ilustrada na Figura 4.7 (a), a *estratégia orientada a corte* realiza bipartições sucessivas sequencialmente, sem hierarquia. Com isso, o algoritmo tem de lidar com partições de tamanhos diferentes. Tal estratégia é pouco utilizada, pois a minimização do corte entre as primeiras partições será muito bom, mas as demais terão grande prejuízo.

Sherwani (SHERWANI, 1998) afirma que a estratégia de *bisseção em fatias* diminui o congestionamento na periferia do circuito, caso existam muitas conexões de I/O, mas apresenta os mesmos problemas da estratégia orientada a corte, em função das partições de tamanhos diferentes.

O objetivo da *bisseção hierárquica em bandas* é dividir o circuito e organizá-lo com particionamentos verticais. Cria-se uma partição do circuito em duas fatias horizontais. Para cada uma delas, a função é realizada recursivamente, até que o circuito esteja dividido em fatias horizontais da altura de uma banda de células. Após, cada banda é particionada no sentido vertical. O processo termina conforme um dos critérios de parada: (1) número fixo de níveis de hierarquia, (2) partições com menos de *x* elemento(s), (3) espaço menor que *y* micrometros.

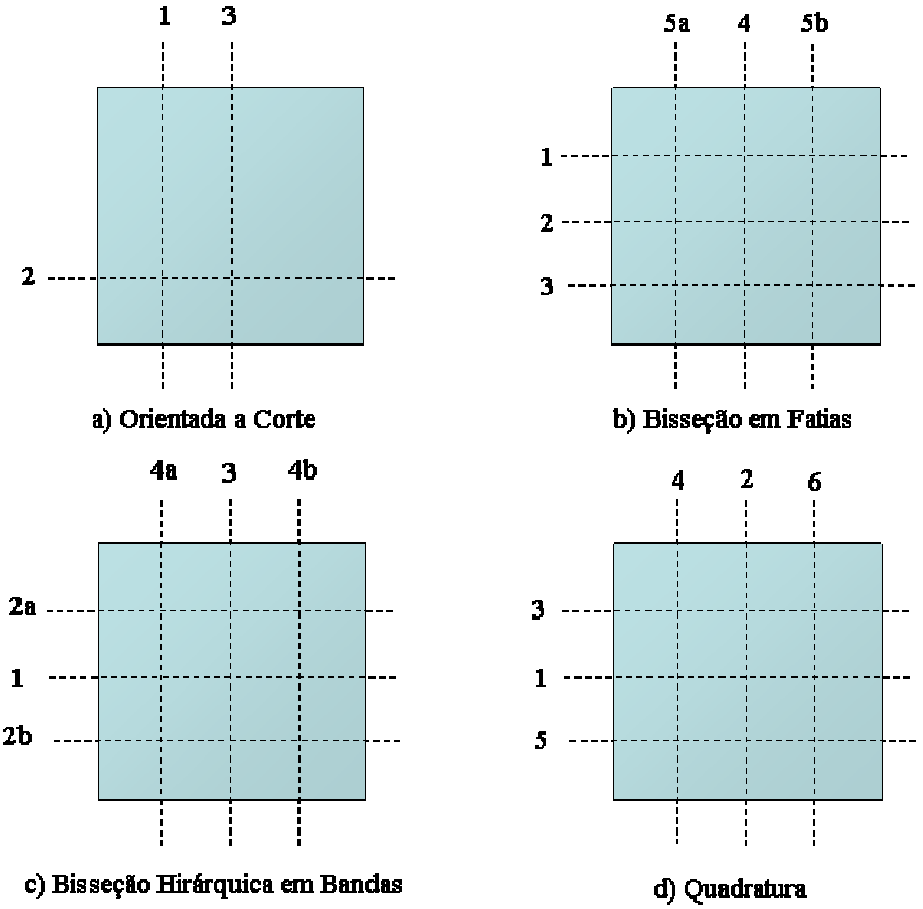


Figura 4.7: Estratégias de corte (SHERWANI, 1998).

Analisando-se os critérios de parada, verifica-se que o primeiro não é eficiente. Os circuitos grandes exigem mais níveis hierárquicos do que os pequenos. Porém, é importante que haja esse tipo de limitação, juntamente com algum outro critério, para que não aconteça uso excessivo de memória devido a recursividade. A terceira alternativa, limitar o espaço físico de cada partição, resolve o problema da diferença de níveis, o que é necessário em circuitos grandes, mas pode fazer com que algoritmos percam tempo com partições sem elementos.

O critério que considera um número *x* de elementos é o mais utilizado. Quanto maior o valor de *x*, maior será a liberdade do posicionador. Caldwell (CALDWELL et al., 2000) usou *x* próximo de 15, juntamente com um posicionador baseado em busca exaustiva. Isso garante que o posicionamento em cada partição seja ótimo. Seus experimentos constataram que 15 células foram adequadas para atingir a eficiência desejada.

A *quadratura* é baseada em particionamentos quádruplos. Usualmente, utiliza-se o método hierárquico. Nesse caso, ela se assemelha a uma bisseção em direções alternadas. É importante destacar que Yildiz (YILDIZ, 2001) mostrou que essa estratégia de corte é a melhor (conforme relatado abaixo). O particionamento seguido na estratégia de quadratura é repetido até que a partição tenha a altura das bandas. A partir disso, pode-se levar em consideração outros critérios, tais como parar o particionamento ou seguir com cortes verticais. Pode ser interessante utilizar uma busca exaustiva, como relatado por Caldwell (CADWELL et al., 2000), ou seguir com cortes verticais, o que leva ao uso dos critérios citados. Contudo, é importante definir qual o ponto de parada ideal conforme cada problema específico.

Há anos especulava-se que essa sequência fosse de fato a melhor, porém Yildiz demonstra-o por meio de um modelo formal. O autor definiu um método para encontrar a sequência ótima de corte. Usando a *regra de Rent*[2]*,* ele define a estimativa do tamanho dos fios de um dado corte antes de efetivamente realizá-lo. A partir disso, ele modela, utilizando programação linear, como encontrar a melhor posição para o corte e qual a porcentagem de área de cada partição.

Com esse modelo, foram realizados alguns experimentos que mostraram duas propriedades muito interessantes: (1) todos os cortes foram bisseções de área totalmente balanceadas, ou seja, 50% para cada partição; (2) se a relação de aspecto de bandas por colunas exceder um certo valor, o particionamento é feito horizontalmente, caso contrário, verticalmente. Isso leva à conclusão de que a definição da direção do corte pode ser feita apenas com uma análise da relação de aspecto da região e, ainda, de que o corte sempre deve ser dado na metade da área.

4.4.5.3 Propagação de Terminal

Este tópico descreve uma abordagem relacionada ao posicionamento baseado em particionamento, a técnica de propagação de terminal (*terminal propagation).* Quando se utiliza o particionamento para auxiliar o posicionamento, é necessário considerar-se a localização das células. Por exemplo, se duas células conectadas, *x* e *y*, estão localizadas em partições diferentes, e o algoritmo inicia uma nova sequência de particionamentos independente da ligação entre elas, ele está ignorando tal conexão. A Figura 4.8 ilustra essa situação, juntamente com a solução por meio da técnica de *terminal propagation*, que quebra as conexões e insere entre elas um novo terminal, semelhante a um *pad* de entrada e saída. Com isso, o algoritmo mantém a proximidade das células.

[2] Introduzida pela primeira vez por Landman, em 1971, é utilizada em estimativas de tamanho dos fios e congestionamentos.

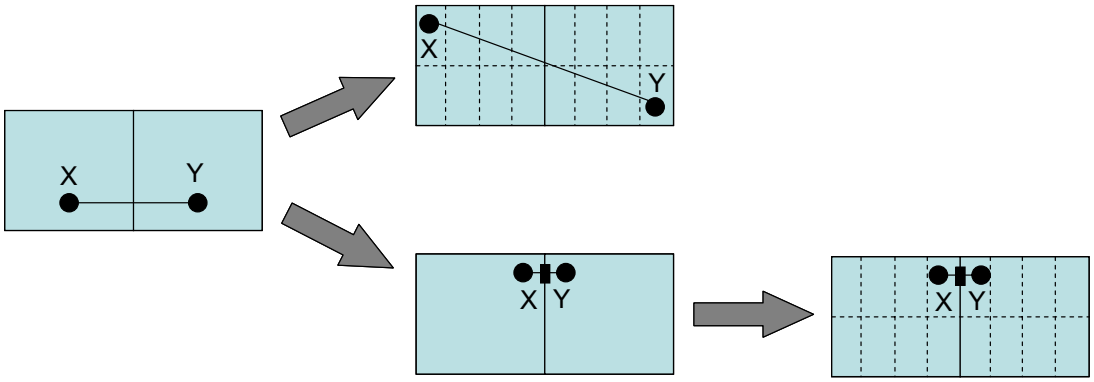


Figura 4.8: *Terminal propagation*.

4.4.6 Outras Técnicas de Particionamento

Outra classe de algoritmos busca aumentar o número de conexões internas. Seu objetivo é aglutinar um grande número de elementos a fim de reduzir quantos serão tratados pelo algoritmo de particionamento. Por exemplo, Fidducia-Matheyses possui complexidade de tempo quadrático, Kernighan-Lin, cúbico; isso indica a importância de reduzir o número entradas. Essa classe de algoritmos é conhecida como *formação de aglomerados.*

Técnicas baseadas em meta-heurísticas, como *simulated annealing* e algoritmos genéticos (GOLDBERG, D., 1989), também são muito utilizadas nas etapas de particionamento. São conhecidas como *simulação de fenômenos.* Outras meta-heurísticas usadas para esse tipo de problema são GRASP (*Greedy Randomized Adaptive Search Procedure*) (FEO, T. A.; RESENDE, M. G. C., 1995) e Busca Tabu (GLOVER, F.; LAGUNA M., 1997)

4.4.7 Grupos de Pesquisa em Particionamento de Grafos

Muitos grupos de pesquisa trabalham com combinações de estratégias de particionamento tradicionais, tais como refinamento multinível, Kernighan-Lin, Fiduccia-Matheyses e *n*-partições. Contudo, a grande maioria dessas estratégias não trata do particionamento de hipergrafos (o qual se assemelha a uma *netlist*). Além disso, no caso do particionamento de grafos, tais ferramentas obtêm resultados de corte (*min-cut*) inferiores aos de hMetis (KARIPYS, 1997). Abaixo é efetuada uma breve descrição de algumas ferramentas e grupos de pesquisa.

4.4.7.1 CHACO – Particionamento de Grafos

O particionador de grafos Chaco foi desenvolvido por Hendrickson e Leland (HENDRICKSON; LELAND, 1995), pesquisadores do Sandia National Laboratories. Seu objetivo é reduzir de forma recursiva o número de cortes entre as partições. Sua estrutura inclui o método Kernighan-Lin e técnicas de particionamento multinível. Contudo, Chaco não atua em hipergrafos.

4.4.7.2 Party – Particionamento de Grafos

Desenvolvida por Preis e Diekmann (PREIS; DIEKMANN, 1997) da Universidade de Paderborn. Party consiste em uma biblioteca que contém um conjunto de algoritmos

de particionamento de grafos. Guarda a implementação de inúmeras heurísticas, tais como Kernighan-Lin, Fiduccia-Matheyses, biparticionamentos e *n*-particionamentos.

4.4.7.3 JOSTLE - Particionamento de Grafos

JOSTLE é um particionador de grafos paralelo desenvolvido pelo Grupo de Pesquisa em Processamento Paralelo da Universidade de Greenwich (WALHAW; CROSS, 2007). Ele utiliza estratégias de particionamento multinível com redução do corte entre as partições por meio do algoritmo Kernighan-Lin.

4.4.7.4 Scotch – Particionamento de Grafos

Scotch é um projeto do Laborabório de Pesquisa em Informática (LaBRI) da Universidade de Bordeaux I (PELLEGRINI, 2007). Seu foco é estudar a teoria dos grafos para computação científica usando a abordagem "dividir para conquistar". Utiliza a combinação de algoritmos de particionamento clássicos com diferentes estruturas de dados.

4.5 Particionamento em Circuitos 3D

Em circuitos planares, o particionamento trabalha, muitas vezes, auxiliando os posicionadores (ROY et. al., 2005), (AGNIHOTRI; ONO; MADDEN, 2005). Alguns processos 2D são modificados e adaptados para atuarem em circuitos 3D, tais como: redução do tamanho do problema; particionamento de blocos com tamanhos fixos; particionamento de blocos com tamanhos variados; redução da complexidade na etapa de posicionamento.

Em tecnologias 3D, o particionamento tem papel fundamental no assinalamento de células ou blocos para as camadas do circuito, na redução de vias-3D, no consequente tamanho de conexões, entre outros (ABABEI et. al., 2005), (DAVIS et. al., 2005), (DENG; MALY, 2001), (KARYPIS et. al., 1999). Surgem assim novas questões que demandam a atenção no particionamento em 3 dimensões, como: criação de *tiers*; particionamento de blocos entre *tiers*; particionamento dirigido a caminhos críticos; particionamento dirigido à redução de vias-3D; particionamento de *pads* de I/O; particionamento de pinos de I/O. É importante ressaltar que o particionador mais utilizado em projetos 3D é o hMetis (KARYPIS et. al., 1997).

Abaixo são discutidos esses tópicos. Grande parte dessas questões são ignoradas pela maioria da literatura a respeito de circuitos 3D.

4.5.1 Criação de *Tiers*

Os algoritmos de particionamento atuam na criação de partições e podem agir em todos os níveis de projeto 3D. Por exemplo, no nível de *tier,* agem por meio da divisão dos blocos heterogêneos do circuito. No nível de blocos *ips* e também lógica aleatória atuam considerando o número de conexões que cruzam as partições e também o balanceamento do circuito e dos blocos. Contudo, a maioria das abordagens existentes (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005), (DENG, MALY, 2001) e (GOPLEN; SAPATNEKAR, 2005) ignoram completamente as restrições impostas pelas vias-3D trabalhando assim, sem a informação real da localização da *tiers*.

A questão está relacionada ao momento em que as partições tornam-se *tiers*. Essa diferença conceitual afeta diretamente a qualidade do particionamento, pois o surgimento de conexões verticais longas acontece no momento em que as partições se posicionam em linha. Assim, com o arranjo fixo, cada antiga partição se torna uma *tier* e seu conjunto forma um circuito 3D.

4.5.2 Particionamento de Blocos entre *Tiers*

O particionamento de blocos trabalha com a idéia de reduzir as conexões que cruzam as partições, mantendo o balanceamento de área entre elas. Nesse nível o tamanho dos blocos é variado e pode-se migrar para qualquer partição. Nas metodologias de projeto 3D já mencionadas, é classificado como sendo nível de *ips*.

4.5.3 Particionamento Dirigido a Caminhos Críticos

Como mencionado, as vias-3D tem características elétricas e dimensões diferentes das conexões normais. Com isso, é interessante que os caminhos críticos permaneçam em uma mesma camada. Isso pode ser realizada por meio do conhecimento prévio dos elementos que compõem o caminho crítico.

4.5.4 Particionamento dirigido à redução de Vias-3D

A maioria dos trabalhos existentes desconsidera os problemas causados pelas vias-3D e particiona o circuito por meio de particionadores de hipergrafos (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005), (DENG, et al., 2001) e (GOPLEN; SAPATNEKAR, 2005). Contudo, no momento do assinalamento das partições para as *tiers*, surgem as conexões longas. Outros trabalhos como os de (HENTSCHKE, et al., 2007), (KAYA; OLBRICH; BARKE, 2004), (OBERMEIER; RANKE; JOHANNES, 2005) utilizam a posição das *tiers* fixas e buscam a otimização das vias-3D por meio de algoritmos direcionados a força. Essa abordagem também não se mostra eficiente, mas é explorada em detalhes no capítulo 4.

A abordagem deste trabalho segue dois caminhos. O primeiro é o de melhorar os resultados de corte entre as partições dos algoritmos atuais. O segundo é o de reduzir o número de conexões não-adjacentes de forma que haja convergência para a redução do número de vias total.

4.5.5 Particionamento de *Pads* de I/O

Os *pads* de I/O possibilitam a comunicação interna do circuito com os dispositivos externos. Em circuitos 3D, podem ficar posicionados acima ou abaixo das *tiers*, entre as *tiers* ou até mesmo em cima e em baixo, como detalhado na seção 3.2.4.2. A definição de como serão particionados pode ser realizada antes ou depois da realização do particionamento e posicionamento final do circuito. No caso de se definir antes, as células e blocos com conectividade são direcionados para *tiers* localizadas próximas aos *pads*. Se a decisão for depois do particionamento, os *pads* são posicionados para ficar o mais próximo possível das saídas internas do circuito. Contudo, pode-se correr o risco de superpopulação de *pads* em uma única *tier*. Uma abordagem diferente e ainda não explorada na literatura seria a de particionar o circuito em função de seus *pads* de I/O.

4.5.6 Particionamento de Pinos de I/O

Os pinos de I/O referidos nesse ponto do trabalho dizem respeito as entradas e saídas dos blocos internos que compõem o circuito. Os pinos de I/O fazem a comunicação da parte interna do circuito e também com os *pads* de I/O. Uma forma de se melhorar a qualidade dos resultados do particionadores de hipergrafos atuais foi explorada neste trabalho e relaciona diretamente os pinos de I/O. Trabalhou-se em um nível de granularidade baixo (nível de células) e verificou-se que os pinos se comportaram como referências para o deslocamento das células.

4.6 Posicionamento

Pode-se definir posicionamento como a etapa de síntese física em que se determina a posição de cada célula no circuito. O posicionamento é um problema NP-completo (GAREY; JOHNSON, 1979), podendo ser classificado pela complexidade do melhor algoritmo que é capaz de resolvê-lo (considera-se o melhor algoritmo aquele que tem a menor complexidade).

4.6.1 Objetivos do Posicionamento

4.6.1.1 Dissipação de Potência

O posicionamento leva em conta a potência total dissipada e a distribuição da dissipação de potência

A dissipação de potência do circuito está estritamente relacionada ao tamanho dos fios e, consequentemente, à capacitância dos mesmos. Minimizar a potência total dissipada pode gerar áreas com maior dissipação que outras. Nesse caso, a distribuição da dissipação deve ser modelada separadamente. Tsai (TSAI, 2000) apresentou um algoritmo de posicionamento que visa a equilibrar a dissipação de potência em todo o circuito e a realizar a minimização total dos fios.

4.6.1.2 Roteabilidade

Circuitos mal posicionados dificilmente podem ser roteados completamente. Nesse sentido, pode-se afirmar que o posicionamento é o principal responsável pela roteabilidade.

Considera-se que, para tornar um circuito roteável, seja necessário diminuir o tamanho médio das conexões. Seguindo essa lógica, é imperativo estimar o tamanho total das conexões do circuito para, a partir do processo de posicionamento, diminuir-se tal média. O somatório total das conexões é conhecido como *wirelength*. Minimizar esse somatório equivale a diminuir a conexão média. No entanto, isso não garante a distribuição equivalente dos fios no circuito. Deve-se evitar o congestionamento de determinadas regiões. Para estimar o *wirelength*, é necessário que se utilize algoritmos capazes de se adaptar a esse propósito. Esses algoritmos são descritos no item abaixo.

4.6.1.3 Tamanho dos fios

O posicionamento não conhece o tamanho das conexões, apenas a posição x e y de cada célula. As células são chamadas de “caixas pretas”, pois não se sabe a localização das posições dos contatos de entrada e saída. Trabalha-se, então, com *estimativas de*

tamanho das conexões. Em 1995, Sait e Youssef (SAIT; YOUSSEF, 1995) apresentaram algumas estratégias para estimar o tamanho dos fios para hiper-redes.

a) Semiperímetro
b) Grafo completo
c) Origem para destino
d) Menor cadeia
e) Menor *spanning tree*
f) Árvore de *Steiner*.

4.6.1.4 Congestionamento

Não basta minimizar o tamanho médio das conexões para termos um circuito roteável. É necessário que haja uma distribuição equilibrada das conexões ao longo do circuito. Se houver regiões com grande congestionamento, o algoritmo de roteamento pouco poderá fazer.

É importante destacar que minimizar o tamanho médio das conexões diminui o congestionamento, mas não garante que as conexões estejam distribuídas de forma equilibrada dentro do circuito.

O congestionamento pode ser definido como a relação entre a demanda por conexões e a oferta de espaço para elas. Agora, como estimar essa demanda? A pergunta deve ser respondida através de modelos de congestionamento, mas qualquer estimativa que desconheça o algoritmo de roteamento pode ser imprecisa. Alguns modelos são apresentados abaixo:

a) **Densidade máxima por *bounding-boxes***: Com esse método, cria-se regiões retangulares idênticas em todo o circuito. Para cada região é calculado quantas conexões a cruzam. A região a qual cruzam mais redes termina a área com maior congestionamento.

b) **Densidade máxima usando roteamento global**: O desconhecimento do algoritmo de roteamento leva ao desenvolvimento de modelos de congestionamento imprecisos, como a intercessão por *bounding-boxes*. Wang (WANG, et. al., 2000) relata que modelos de congestionamento usando roteamento global são mais precisos em comparação aos baseados em *bounding-boxes*.

c) **Corte máximo:** Com esse método, procura-se o ponto do circuito onde passa o maior número de conexões. São realizados cortes horizontais e verticais em toda a área do circuito. Seleciona-se o local onde cruza o maior número de conexões horizontal e vertical, escolhendo-se aquele que contém o maior corte global.

d) ***Overflows***: Trata-se da quantidade de conexões excedentes em uma determinada região, calculada com o uso de duas variáveis: demanda e oferta. Demanda é a quantidade de conexões que precisam passar pela região, e oferta é o número de trilhas oferecidas para que essas conexões passem.

4.6.1.5 Área

Dado um conjunto de células, o posicionador deve encontrar a área mínima para alocar cada uma delas adequadamente. Porém, dependendo da complexidade do circuito, o posicionador pode não conseguir torná-lo roteável, o que exige o aumento da área.

4.6.1.6 Timing

Como mencionado anteriormente, as tecnologias atuais agregaram características novas, as quais devem ser consideradas para diminuir o atraso do circuito. A resistência das conexões aumenta em função da diminuição destas. Os *chips* são cada vez maiores e, com isso, a resistência das conexões que o cruzam tornam-se muito grandes, de modo que precise inserir repetidores. O canal dos transistores diminuiu nos últimos anos, consequentemente, as células lógicas operam mais rapidamente. Atualmente, o atraso médio das conexões é da mesma ordem de grandeza do atraso dos transistores. *Se a aproximação feita pelo posicionamento não for eficiente, o roteamento pouco poderá contribuir para diminuir o atraso de uma conexão crítica que cruze o circuito*.

Além do tamanho das conexões, o *crosstalk* pode influenciar o atraso do circuito. Uma linha de roteamento pode influenciar a outra, deixando o sinal lento, ou até mesmo mudando seu estado lógico.

4.7 Classificação e Revisão dos Algoritmos de Posicionamento

Os algoritmos de posicionamento podem ser classificados em três grandes grupos: baseados em particionamento (ROY; PAPA; ADYA; CHAN; NG; LU; MARKOV, et. al., 2005), (AGNIHOTRI; ONO; MADDEN, 2005); *simulated annealing* (SECHEN; VICENTELLI, 1985), (TAGHAVI; YANG; CHOI, 2005) e direcionados a força (EISENMANN; JOHANNES, 1998). Um importante subconjunto dos métodos direcionados à força são os algoritmos quadráticos (OBERMEIER; RANKE; JOHANNES, 2005), (HU; ZENG; MAREK-SADOWSKA, 2005), (KAHNG; REDA; WANG, 2005), (VISWANATHAN; PAN; CHU, 2005).

Os algoritmos baseados em *simulated annealing* (simulação de têmpera) têm tempo de CPU bastante elevado, pois executam largas buscas para evitar mínimos locais. No entanto, posicionadores como o Dragon, o qual usa s*imulated annealing* (TAGHAVI; YANG; CHOI, 2005), são conhecidos por alcançar resultados de alta qualidade. Na estrutura interna dessa ferramenta, um processo de particionamento é utilizado para diminuir o tempo de CPU causado pelo *simulated annealing*.

Os algoritmos de posicionamento baseados em particionamento utilizam-se de técnicas para dividir o problema em partes menores (AGNIHOTRI; ONO; MADDEN, 2005). Essa heurística tem menor complexidade em comparação com métodos baseados em *simulated annealing*.

Métodos direcionados à força aplicam estratégias de atração e repulsão entre células. Faz-se uma analogia a um sistema físico de forças de atração e repulsão. No método de posicionamento quadrático (proposto inicialmente por Hall (HALL, 1970), a função-objetivo é igual ao quadrado do tamanho dos fios. Encontrando-se a derivada da função-objetivo, minimiza-se o quadrado da distância euclidiana entre células conectadas. A solução ótima para o posicionamento é, então, encontrada obtendo-se a

derivada da função-objetivo igual a zero. Isso é alcançado pela resolução de um sistema linear que pode ser eficientemente resolvido por métodos numéricos, o que equivale a achar o ponto de equilíbrio de um sistema de molas, porém aplicado a solução a células e *pads*. Os métodos de posicionamento quadrático buscam o resultado ótimo no estado de equilíbrio do sistema de molas, desconsiderando a sobreposição entre células (*overlaps*). Existem muitas técnicas para remover *overlaps* enquanto as células são distribuídas na área. Os métodos quadráticos são conhecidos pela rapidez e bons resultados.

As três técnicas citadas anteriormente (baseada em particionamento, *simulated annealing* e direcionada à força) são utilizadas tanto na indústria quanto na academia, e é muito difícil determinar qual é a mais eficiente.

Outro tema de destaque relaciona os algoritmos rápidos e a baixa complexidade. Para o posicionamento de células existem dois importantes objetivos: estimar o tamanho das conexões e processar o resultado final do posicionamento. Um posicionador de células é capaz de estimar com precisão o tamanho dos fios, mas CPU e qualidade são, muitas vezes, incompatíveis. Quando estimamos o tamanho das conexões, o tempo de CPU é muito importante, pois a execução do posicionador é realizada inúmeras vezes. Temos, então, com rapidez, uma estimativa aproximada do tamanho das conexões naquele momento. Já para o posicionamento final, a qualidade deve ser considerada; em contrapartida a CPU não é tão importante quanto o fornecimento rápido de um resultado preciso do tamanho das conexões.

Com o passar dos anos, o tamanho dos problemas continuam aumentando. Mais e menores elementos a serem tratados acarretam novas questões elétricas que, em tecnologias anteriores, eram irrelevantes. Com isso, no que diz respeito a circuitos 3D, é razoável imaginar que essas questões continuarão a aumentar. Por exemplo, a otimização através da técnica de *simulated annealing*, apesar das questões descritas acima (tempo de CPU e tamanho do circuito), ainda é bastante empregada em combinação com outras técnicas. Atualmente, por exemplo, ela é utilizada para resolver problemas de *floorplaning* (número pequeno de entradas) e posicionamento detalhado, desde que o tamanho do problema seja reduzido (por exemplo, com técnicas de “clusterização”).

Existem muitos posicionadores com tempos de CPU aceitáveis. A ferramenta *FastPlace* é a mais rápida entre os existentes. Ela foi criada com base no método de posicionamento quadrático (VISWANATHAN; PAN; CHU, 2005) e utiliza três passos distintos: posicionamento global, tamanho das conexões e posicionamento detalhado.

Como mencionado, a resolução do sistema linear gera muitos *overlaps* (sobreposições) e deve ser intercalado com técnicas de espalhamento. O algoritmo de deslocamento de células (*cell shifting*) espalha-as adicionando molas conectadas aos pinos na borda do circuito. O objetivo dessa técnica é dividir o circuito em regiões regulares e de mesmo tamanho, chamadas de *bins,* e calcular a sua utilização. Após, as regiões mais utilizadas são aumentadas, e as menos, diminuídas, de forma a criar uma melhor acomodação dentro das regiões. Então, as células são mapeadas partindo-se das regiões regulares para as aumentadas e diminuídas. Em seguida, molas são adicionadas para puxarem cada célula para a posição adequada. Finalmente, o sistema linear é resolvido novamente. Esse processo se repete até que a utilização das regiões seja pequena.

O refinamento local iterativo divide o circuito assim como o deslocamento de células. A partir daí, estas são posicionadas nas quatro regiões ortogonalmente adjacentes, e um escore para cada movimento é atribuído com base na melhoria do tamanho dos fios e na utilização. Ao final, a célula é movida para a região com maior escore não-negativo. O posicionamento detalhado usa uma técnica similar à do refinamento local iterativo, entretanto, atribui um peso maior para a utilização.

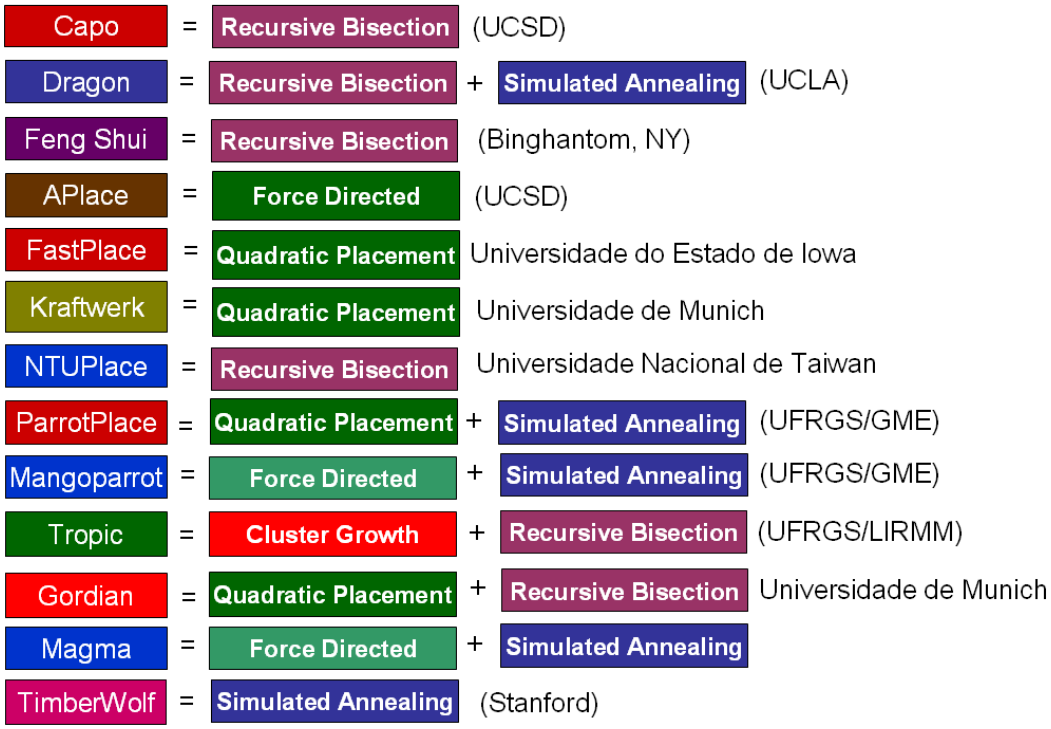


Figura 4.9: Posicionadores e suas combinações.

4.8 Posicionamento de Circuitos 3D

O posicionamento é a etapa de síntese na qual é definido uma posição física para cada célula, no circuito, sem que haja intercessão da área de células. Trata-se de um ponto-chave para projetos VLSI. Há décadas se estudam algoritmos de posicionamento, mas, conforme avança a tecnologia, o problema torna-se mais complexo. Com a mudança de paradigma de projeto, alguns algoritmos 2D são estendidos para posicionadores 3D, tais como: otimização do tamanho das conexões (2D); posicionamento com blocos fixos; posicionamento de blocos com tamanhos variados; posicionamento dirigido a *timing*; posicionamento dirigido a *power*; posicionamento dirigido a questões térmicas.

Devido ao surgimento de novas restrições, impostas pelas tecnologias atuais, o posicionamento em circuitos de 3 dimensões demanda atenção a novos problemas, como: posicionamento e legalização de vias-3D; otimização do tamanho das conexões em 3 dimensões; posicionamento 3D voltado a questões térmicas; posicionamento com obstáculos móveis; posicionamento 3D dirigido a *timing*.

Nos próximos parágrafos, os tópicos acima são tratados de acordo com trabalhos publicados. Do mesmo modo, apresenta uma revisão dos algoritmos de posicionamento e discute as novas questões relativas a circuitos 3D.

O posicionamento com obstáculos é um problema importante com que lida a maioria dos posicionadores. Isso acontece quando, por exemplo, um posicionamento *standard cell* tenha de interagir com uma área compartilhada com macroblocos fixos. Nesse caso, existem áreas posicionáveis e não-posicionáveis. Estas são as conhecidas na literatura como "obstáculos". O posicionamento com obstáculos foi um assunto avaliado no ISPD de 2005, por meio de uma competição (NAM et. al., 2005).

Muitos autores ignoram inicialmente os obstáculos, resolvendo-os somente no estágio de legalização (BRENNER; VYGEN, 2004). Outra técnica possível é a *landscape smoothing,* usada pelo APlace – vencedor da competição de posicionamento do ISPD 2005 (KAHNG; REDA; WANG, 2005). Pode-se ainda executar um algoritmo de *floorplaning* considerando-se as células como blocos livres e mantendo-se todos os blocos duros fixos (VISWANATHAN; PAN; CHU, 2005). Todos os posicionadores utilizados nas competições do ISPD, em 2005 e 2006, fornecem opções de manipulação de obstáculos. Informações mais detalhadas podem ser encontradas nos artigos dos participantes, que estão disponíveis na *homepage* dessas conferências.

Além disso, as vias-3D podem ser consideradas como obstáculos, pois ocupam a área ativa e não podem ser utilizadas pelas células.

Já no posicionamento com blocos de tamanhos variados, muitos projetos contêm a lógica aleatória misturada a macroblocos móveis. Com isso, o algoritmo que os posiciona deve considerar não apenas os blocos de mesma altura, mas também os de alturas diferentes. Essa estratégia é chamada na literatura de posicionamento com tamanhos de células e blocos variados, ou apenas *mixed-size* (VISWANATHAN; PAN; CHU, 2006).

Capo (ROY et. al., 2005) foi inicialmente proposto por Caldwell (CALDWELL; KAHNG; MARKOV, 2000). Basicamente, ele executa um *min-cut* conforme técnicas de biparticionamento estendido para manipular blocos de tamanhos variados com o auxílio de um *floorplaner.*

Viswanathan, Pan e Chu apresentaram uma nova versão do algoritmo FastPlace para manipular blocos com tamanhos variados (VISWANATHAN; PAN; CHU, 2006). Eles argumentam que métodos direcionados à força, tal como posicionamentos quadráticos, podem manipular facilmente objetos de tamanhos variados sem a necessidade de que se empregue técnicas adicionais, tais como particionamento e "clusterização". Nesse artigo, uma extensão do método de *cell shifting* é utilizado para manipular blocos de tamanhos variados, e não são realizadas mudanças no método de otimização quadrático, mas esse método ainda precisa considerar as dimensões dos blocos a fim de computar a utilização dos *bins* e adicionar novas forças para espalhar as células.

O posicionamento dirigido a *timing* pode ser classificado em dois grupos: os métodos com base no caminho (*path-based*) e os com base na rede (*net-based).* As abordagens mais comuns consideram pesos nas redes (KAHNG; WANG, 2004), (XIU; RUTENBAR, 2005), (RIESS; ETTELT, 1995), (KONG, 2002), (YANG; CHOI;

SARRAFZADEH, 2002). Outros métodos são baseados em *slacks* (KAHNG; WANG, 2004), (LUO; NEWMARK; PAN, 2006), (XIU; RUTENBAR, 2005).

Nas tecnologias modernas de fabricação de circuitos integrados, observa-se uma série de diferenças que tornam o atraso das conexões um dado muito relevante para o atraso total do circuito. A resistência das conexões que cruzam internamente o *chip* aumenta ainda mais esse atraso, pois os circuitos estão cada vez maiores (apesar da redução dos elementos, mais componentes são tratados). O mesmo vale para a capacitância das conexões que cruzam o *chip*. Ainda, o canal do transistor é produzido em tamanho progressivamente menor, fazendo com que as células lógicas operem de forma mais rápida. Como a etapa de posicionamento é responsável por definir a posição física das células no caminho crítico, o tamanho das conexões é significativamente influenciado por essa decisão.

Em recentes trabalhos desta tese, procurou-se posicionar todas as células que compõem o caminho crítico em uma *tier*. O objetivo era evitar que as redes que fazem parte do caminho crítico cruzassem *tiers*.

O posicionamento dirigido a potência pode otimizar duas variáveis relacionadas à dissipação de potência: (1) potência total dissipada e (2) distribuição da dissipação de potência. O tamanho dos fios de roteamento está diretamente relacionado à capacitância dos mesmos e, portanto, à dissipação de potência do circuito. Assim, a minimização do tamanho total dos fios também auxilia a dissipação total de potência (ZHANG, T.; ZHANG Y.; SAPATNEKAR, S., 2006).

Porém, minimizar a potência total dissipada pode criar áreas com maior dissipação que outras. Por essa razão, a distribuição da dissipação é uma variável que deve ser modelada em separado. Tsai e Kang (TSAI; KANG, 2000), apresentam um algoritmo de posicionamento que visa equilibrar a dissipação ao longo do circuito e a minimização do tamanho total dos fios. O autor desenvolve um modelo de dissipação de temperatura com base apenas na posição das células (não considera a dissipação das conexões).

Em circuitos 3D, a potência é uma melhoria que atua diretamente com o tamanho das conexões e identificação das conexões críticas.

O posicionamento dirigido a questões térmicas é um assunto bastante estudado em relação a circuitos 2D. Em circuitos 3D, o posicionamento dirigido a questões térmicas ganhou grande importância devido ao acúmulo de calor nas camadas intermediárias do *chip*. Por isso, os algoritmos de posicionamento e *floorplanning* necessitam, cada vez mais, de rapidez e eficiência para identificar os focos desse calor.

Goplen e Sapatnekar apresentaram um posicionador dirigido a forças térmicas que melhoram a temperatura do *chip* (GOPLEN; SAPATNEKAR, 2005). Elas movem as células dos focos de calor para outros lugares. Esses autores relatam que a média de diminuição da temperatura do *chip* é de 17%. Seu trabalho não considera o custo das vias-3D; Balakrishnan (BALAKRISHNAN et al., 2005) propôs um posicionador que utiliza dois passos. Inicialmente um algoritmo com base em particionamento usa uma solução focada no congestionamento dos fios e na redução do consumo dinâmico de potência. Depois, um refinamento com base em *simulated annealing* melhora a temperatura do *chip*, o tamanho das conexões, o congestionamento e o número de vias-

3D com múltiplas funções-objetivo. Os autores observaram que o congestionamento dos fios está relacionado com os focos de calor. Esse método utiliza interessantes *tradeoffs* entre as variáveis das funções-objetivo.

O posicionamento com obstáculos móveis trata outros elementos como se fossem células ou blocos. Como mencionado anteriormente, as vias-3D que furam a camada ativa ocupam espaço nas diferentes camadas de um *chip* 3D. Kaya (KAYA et. al., 2004) apresentou um posicionador dirigido a forças que tem boas propriedades de manipulação e posicionamento de vias. Basicamente, qualquer via-3D que fure a camada ativa não deve ser sobreposta por outra célula em uma mesma camada. Nesse caso, as vias podem ser modeladas como obstáculos móveis. Caso a altura da via seja maior do que a de uma célula (*standard cells*), podem ser utilizadas estratégias de posicionamento com blocos de tamanhos variados.

O posicionamento de vias-3D pode ser tratado como um elemento adicional na etapa de posicionamento. Conforme discutido anteriormente, o posicionamento e a legalização podem ser definidos e resolvidos em cada uma das camadas.

Apresentou-se uma formulação simples para esse problema em (YAN; LI; ZHOU; HONG, 2005). Primeiramente, uma grade de posicionamentos válidos é definida conforme as restrições de distância entre as vias-3D. Em seguida, computa-se uma matriz de pesos que determina uma função-custo para cada via, em todos os possíveis lugares do leiaute. Esse custo poderá ser o tamanho das conexões, a potência, os efeitos térmicos, entre outros. Ao final, as vias são posicionadas no lugar de menor custo conforme o critério escolhido.

4.9 Resumo do Capítulo

Este capítulo apresentou um levantamento bibliográfico dos algoritmos de particionamento e posicionamento que atuam no projeto de circuitos. A primeira parte descreveu a etapa de particionamento, sua definição do problema, características, classificações, algoritmos tradicionais utilizados e suas evoluções. Discutiu-se também a aplicação desses algoritmos em projetos 3D e investigou-se sua atuação na redução de vias-3D. A segunda parte abordou a etapa de posicionamento, suas características, objetivos, classificações e discussões em torno de sua forma de atuação em projetos 3D, e quais as estratégias adotadas para a redução do número de vias-3D.

Percebeu-se pela literatura que há anos estudam-se algoritmos de particionamento. O reflexo são muitos algoritmos com resultados de *min-cut* de boa qualidade. Entre os algoritmos tradicionais destacam-se os de migração de grupos, Kernighan-Lin e Fiduccia-Matheyses. Ambos baseiam-se na redução do número de cortes entre as partições. O primeiro realiza troca de pares de células entre as partições, mas apresenta uma série de desvantagens. Sua ordem de complexidade é cúbica $O(n^3)$, pois não possui um mecanismo eficaz de atualização dos ganhos. Além disso, não consegue balancear as partições por área, o que seria possível se as células tivessem tamanhos iguais. Com o uso desse algoritmo, o balanceamento fica restrito ao número de células, e as que estão localizadas em uma determinada partição, não na rede propriamente dita, servem de base para efetuar o corte.

Já Fiduccia-Mattheyses apresenta evoluções em relação ao algoritmo de Kernighan-Lin. Ele faz migração simples de uma célula e apresenta uma complexidade de tempo

inferior a $O(n^2)$. A modificação mais importante está no corte das redes. Para Fiduccia-Mattheyses, não interessa o número de células localizadas em cada partição, mas se esta é atravessada pela rede.

Em seguida, observou-se que, à medida que o número de elementos aumenta, mais complexa é a resolução do problema. Assim, em 1983, Goldenberg (GOLDENBERG, et al., 1983) desenvolveu uma técnica para unir vértices, o que aumenta sua conectividade. Com isso, o número de elementos a serem tratados diminui significativamente. Associados a essa técnica, Fiduccia-Mattheyses e Kernighan-Lin são melhor aproveitados.

Em 1997, Karypis (KARYPIS, 1997) desenvolveu uma técnica de particionamento multinível combinada com algoritmos de migração de grupos. Essa técnica consiste em reduzir o tamanho do hipergrafo por meio da união de vértices (seguindo a ideia de Golbenberg). Entretanto, assim que o hipergrafo é reduzido, iniciam-se os biparticionamentos, e os vértices começam a ser descompactados, aumentando-se o hipergrafo até que ele chegue ao tamanho original, porém particionado. Para isso, executam-se quatro etapas: (1) compactação, (2) particionamento, (3) descompactação e (4) refinamento. Todas essas fases foram apresentadas em detalhes na Seção 4.4.5.1. Atualmente, hMetis é o algoritmo de particionamento mais utilizado no domínio VLSI (*state-of-the-art*) por promover resultados de *min-cut* de qualidade e balanceamento entre as partições com boa performance.

Foram analisadas também técnicas que utilizam meta-heurísticas, como *simulated annealing* e algoritmos genéticos, ambos baseados em simulação de fenômenos. Percebe-se que essas técnicas obtêm bons resultados de corte e balanceamento, contudo, quanto maior o número de entradas, mais complexa fica a resolução do problema. Basicamente, elas são combinadas com técnicas de clusterização ou utilizadas a partir de uma boa solução inicial.

Outra questão relacionada ao particionamento diz respeito aos algoritmos de posicionamento baseados em particionamento. Nesse caso, o posicionamento utiliza o particionamento como um auxíliar. Entretanto, o conceito de particionamento torna-se mais confuso, pois, ao contrário do que se estuda na teoria dos grafos, esse tipo de algoritmo já tem a informação a respeito da localização das partições.

Relacionado a esse assunto, em 2001, Yildiz mostrou, por meio de um modelo formal, a sequência ótima de corte. O modelo proposto encontra a melhor posição para o corte e a porcentagem de área de cada partição. Assim, pode-se concluir que a definição da direção do corte pode ser feita apenas com uma análise da relação de aspecto da região e, ainda, que o corte sempre deve ser feito exatamente no meio da área. Seus experimentos mostraram que a bisseção em direções alternadas é a melhor estratégia de corte.

Contudo, verificou-se que, além de utilizar a melhor estratégia de corte, é necessário considerar a localização das células. Pois, se duas delas, x e y, estiverem conectadas e localizadas em partições diferentes, quando o algoritmo inicia uma nova sequência de particionamentos independente da ligação entre elas, existe uma grande possibilidade de as células se distanciarem. Nesse caso, a técnica conhecida como propagação de terminais se faz necessária, já que ela promove a aproximação das células localizadas em partições adjacentes.

A Seção 4.5 discutiu a importância do particionamento no projeto 3D. Verificou-se que os algoritmos que atuam nesse tipo de projeto são adaptações dos tradicionais, sendo sua proposta focada especificamente para a redução do número de cortes entre as partições e para o balanceamento da área e do número de elementos. Percebeu-se que as soluções voltadas para a redução de vias-3D estão intimamente relacionadas à qualidade do resultado dos particionadores. Então, quanto menor o corte entre as partições, menor será o número de vias-3D do circuito. Em vista disso, um problema abordado neste trabalho se resume a uma pergunta: **Como melhorar a qualidade da solução obtida pelos particionadores atuais?** Seguindo-se a lógica mencionada, se o número de cortes entre as partições diminuir, o número total de vias-3D também será menor. Tendo em vista essa relação, o capítulo 5 descreve em detalhes uma proposta para melhorar a qualidade das soluções obtidas pelos algoritmos de particionamento atuais, em especial, hMetis, dentro do domínio VLSI.

Analisamos também o problema de os particionadores atuais executarem suas estratégias sem a informação da posição das partições. Isso tem consequências negativas em um projeto 3D, visto que as *tiers* estão posicionadas em linha e, portanto, sabe-se a posição de cada uma delas. Assim, obter uma boa redução do número de conexões que cruzam as partições não garante bons resultados, já que, quando estas forem posicionadas em linhas, o número de conexões não-adjacentes pode ocasionar um aumento significativo do número de vias-3D e, até mesmo, do de conexões longas. Por isso, investigamos **como reduzir o número de vias verticais que através mais de uma camada, de modo que se contribua para a redução total do seu número total.**

Foram discutidos também os algoritmos de posicionamento tradicionais, assim como suas aplicações em projetos 3D. Percebeu-se que os objetivos do posicionamento giram em torno da redução do tamanho das conexões, da fuga por áreas congestionadas, da dissipação de potência, do *timing* e da redução de área. Contudo, todos esses objetivos são secundários a que se encontre uma solução que seja roteável (a roteabilidade é fundamental).

A classificação dos algoritmos de posicionamento e o levantamento bibliográfico foram discutidos em detalhes na Seção 4.7. Desses algoritmos, os voltados para os projetos 3D foram abordados na Seção 4.8.

Percebeu-se que, devido ao surgimento de novas restrições, impostas pelas tecnologias atuais, o posicionamento em circuitos 3D demanda atenção a novos problemas, como posicionamento e legalização de vias-3D, otimização do tamanho das conexões, solução de questões térmicas, obstáculos móveis e posicionamento dirigido a *timing*. Tais tópicos foram desenvolvidos a partir da leitura de trabalhos sobre projetos 3D. Pôde-se verificar que os estudos publicados relacionados a posicionamento não abordam especificamente a redução de vias-3D.

Contudo, analisando-se as estratégias de posicionamento, nota-se que a distribuição de células e blocos entre as *tiers* de um projeto 3D é realizada, muitas vezes, por meio de algoritmos direcionados a forças (como os baseados em *quadratic placement*). Um exemplo é a estratégia apresentada por Hentschke (HENTSCHKE, et al., 2007). Nela, os pinos de I/O são distribuídos e posicionados na borda das *tiers* para que as células possam ser guiadas pelas posições dos pinos. Essa abordagem é realizada com as *tiers* empilhadas, como em um projeto 3D real. Ela pode direcionar seu foco para a redução do número de vias-3D devido à flexibilidade oferecida pela função de custo.

Experimentos sobre o potencial dessa estratégia foram realizados, apresentados e discutidos na Seção 5.6.4.

O próximo capítulo apresenta um fluxo de projeto 3D voltado para a redução do número de vias. Tal fluxo mantém a área das *tiers* e o número de pinos de I/O balanceados enquanto prioriza em sua solução a redução das vias-3D longas e da área do circuito. Essa proposta foi criada com base no levantamento bibliográfico realizado até o presente momento e pela identificação da carência de uma metodologia para tratar dos problemas causados pelas vias-3D.

5 LH-3D: ALGORITMOS DE PARTICIONAMENTO VLSI 3D

5.1 Introdução

Na literatura que aborda circuitos 3D não se tem encontrado trabalhos dedicados aos problemas causados pelas vias-3D (descrito em detalhes na Seção 2.6.4). Contudo, é fácil enumerar diversas questões sobre este assunto que demandam cuidados, por exemplo: (1) a superpopulação de vias-3D gera o aumento de área ativa, ocupando o lugar dos transistores e, consequentemente, aumentando a área do circuito; (2) no momento em que uma via-3D cruza duas camadas adjacentes, surge a necessidade de posicionar e legalizar a sua estrutura (podem ser tratados como obstáculos móveis); (3) pela diferença física estrutural, as vias-3D têm características elétricas diferentes das conexões normais (o problema ocorre quando a via-3D faz parte do caminho crítico); (4) as vias-3D são obstáculos e, como tal, utilizam recursos de roteamento; (5) o *pitch* de uma via-3D é muito grande se comparado com o de uma conexão normal, isso consome bastante área do circuito. Com base nessas informações e em pesquisas específicas sobre o assunto descritas ao longo de todo este trabalho, nota-se que grande parte dos estudos explora a divisão de células e blocos entre camadas sob a perspectiva da redução do número de conexões entre partições, ou seja, uma abordagem que adapta de forma simplista as estratégias dos circuitos 2D para resolver problemas em três dimensões. Nessas soluções, o problema ocorre no momento em que a célula ou partição é atribuída à camada, pois os algoritmos tradicionais que atuam no particionamento de hipergrafos não percebem o arranjo proposto pela tecnologia 3D (arranjo em linha).

Assim, diversos trabalhos (TSAI; WANG; NARAYANAN; IRWIN, 2008), (ABABEI; FENG; SAPATNEKAR, 2005), (GOPLEN, SAPATNEKAR, 2005), (DAVIS et al., 2005), (DENG et al., 2005), (MAIDEE; BAZARGAN, 2005), utilizam o particionador de hipergrafos hMetis (KARYPIS, 1997). Outros autores, como Lee e Lim (LEE, Y.; KIM; HUANG, G.; BAKIR, M.; JOSHI, T.; LIM, S., 2009), utilizam *min-cut* por meio do algoritmo de Fiduccia-Matheyses (FIDDUCIA; MATHEYSES, 1982). Entretanto, como mencionado anteriormente, para problemas específicos de circuitos 3D (como é o caso da redução de vias-3D), tanto hMetis quanto FM não conseguem obter resultados satisfatórios.

Outros estudos (HENTSCHKE; REIS, 2007), (KAYA; OLBRICH; BARKE, 2004), (OBERMEIER; RANKE; JOHANNES, 2005), (HU; ZENG; MAREK-

SADOWSKA, 2005), (KAHNG; REDA; WANG, 2005), (VISWANATHAN; PAN; CHU, 2005) utilizaram algoritmos direcionados a forças para realizar o particionamento do circuito. Nesse contexto, as células e blocos sofrem a ação de forças que os deslocam para outras camadas. Também esses trabalhos não abordam pontualmente formas de redução total de vias-3D.

Apesar de os estudos citados acima relatarem os problemas causados pelo excesso de vias-3D no projeto dos circuitos, nenhum deles apresenta especificamente uma solução para tal.

Este trabalho apresenta duas estratégias eficientes para reduzir o número total de vias em circuitos VLSI 3D. A primeira trabalha com a análise prévia da estrutura do circuito e a segunda reduz o número de conexões verticais não adjacentes. Essa pesquisa resultou no desenvolvimento da ferramenta LH-3D, um fluxo de projeto que auxilia o particionamento de células e pinos em circuitos VLSI 3D (*standard cells*). Sua estrutura é composta por um conjunto de algoritmos de otimização que adaptam a lógica dos circuitos 2D para os 3D. Os algoritmos do fluxo LH-3D estão divididos em três grupos: LHp-3D, LHu-3D e LHr-3D. Todos descritos em detalhes nas próximas seções.

Os algoritmos pertencentes ao grupo LHp-3D atuam de forma diferente das ferramentas de particionamento atuais, pois apresentam uma metodologia de análise prévia da estrutura interna do circuito que é executada antes do particionamento das células. Essa análise consiste em encontrar o menor caminho lógico entre pares de pinos de I/O a fim de mapear as informações de conectividade das células (detalhes no item 5.4). Com base nos experimentos realizados pelo LHp-3D, criou-se o grupo LHu-3D. Esse grupo visa reduzir o número de vias-3D por meio do aumento do desvio padrão do número de pinos de I/O que estão distribuídos pelas *tiers* (detalhes no item 5.5). Por último, o grupo LHr-3D que tem por objetivo convergir para a redução do número total de vias-3D por meio da minimização das vias-3D que cruzam mais de duas *tiers* adjacentes (detalhes no item 5.6). Nas seções seguintes, o fluxo de projeto e os demais algoritmos utilizados pelo LH-3D são descritos e discutidos com maiores detalhes.

5.2 *Benchmarks*

Os resultados deste trabalho foram gerados a partir do conjunto de *benchmarks* ISPD 2004 (ISPD 2004 - IBM STANDARD CELL BENCHMARKS WITH PADS, 2004) resumidos na Tabela 5.1, abaixo. Esse conjunto de circuitos *benchmarks* 2D foram otimizados e convertidos em circuitos 3D e vêm a complementar o conjunto de *benchmarks* disponíveis no meio acadêmico. Foram criados circuitos com duas, três, quatro e cinco *tiers* de todo o conjunto de *benchmarks.*

Os algoritmos desenvolvidos neste trabalho foram comparados com o particionador de hipergrafos mais utilizado no domínio VLSI 3D e que mostra os melhores resultados na redução de conexões que cruzam partições (*state-of-the-art*), hMetis (KARIPYS, 1997); outras comparações foram realizadas com o algoritmo Alternate Pins, uma estratégia que fixa os pinos de I/O alternadamente entre as partições; e também com o algoritmo ZPlace (HENTSCHKE, et al., 2007) uma estratégia direcionada a forças que usa o método *quadratic placement* para distribuir as células entre as camadas.

Tabela 5.1: *Benchmarks* IBM ISPD 2004 com *pads* (ISPD IBM BENCHMARK WITH PADS, 2004)

Bench	***#Cells***	***#Pads***	***#Nets***
ibm01	12.506	246	14.111
ibm02	19.342	259	19.584
ibm03	22.853	283	27.401
ibm04	27.220	287	31.970
ibm05	28.146	1201	28.446
ibm06	32.332	166	34.826
ibm07	45.639	287	48.117
ibm08	51.023	286	50.513
ibm09	53.110	285	60.902
ibm10	68.685	744	75.196
ibm11	70.152	406	81.454
ibm12	70.439	637	77.240
ibm13	83.709	490	99.666
ibm14	147.088	517	152.772
ibm15	161.187	383	186.608
ibm16	182.980	504	190.048

5.3 Fluxo de Particionamento 3D Proposto

O LH-3D inicia com a leitura de uma *netlist* 2D com pinos de I/O e, após a execução dos algoritmos de otimização, a transforma em uma *netlist* 3D. As informações analisadas pela ferramenta são as seguintes:

a) lista de células com informação individual de largura e altura;
b) lista de conexões de I/O;
c) lista de redes (conexões entre células e/ou pinos de I/O);
d) descrição física da área do circuito 2D (altura e largura do bloco com informações da posição dos pinos de I/O nas bordas e percentual de utilização dos espaços em branco);
e) informações do circuito: número de tiers, nível de balanceamento de pinos e sua orientação, opções de otimização.

A saída do LH-3D produz o seguinte:

a) circuitos 3D (circuitos empilhados – número de *tiers* escolhido conforme parâmetro inicial);
b) informações do posicionamento dos pinos de I/O (quando distribuídos entre as *tiers*, ficam localizados nas bordas);
c) *netlist* otimizada (número de vias-3D, vias longas e área);
d) informações do circuito: altura e largura de cada *tier*, posição de cada célula na *tier*, tamanho da célula (altura e largura), número total e máximo de vias-3D, área de células de cada *tier*, área de cada *tier* (área de células + espaços em branco) e área total do circuito (área de células + espaços em branco).

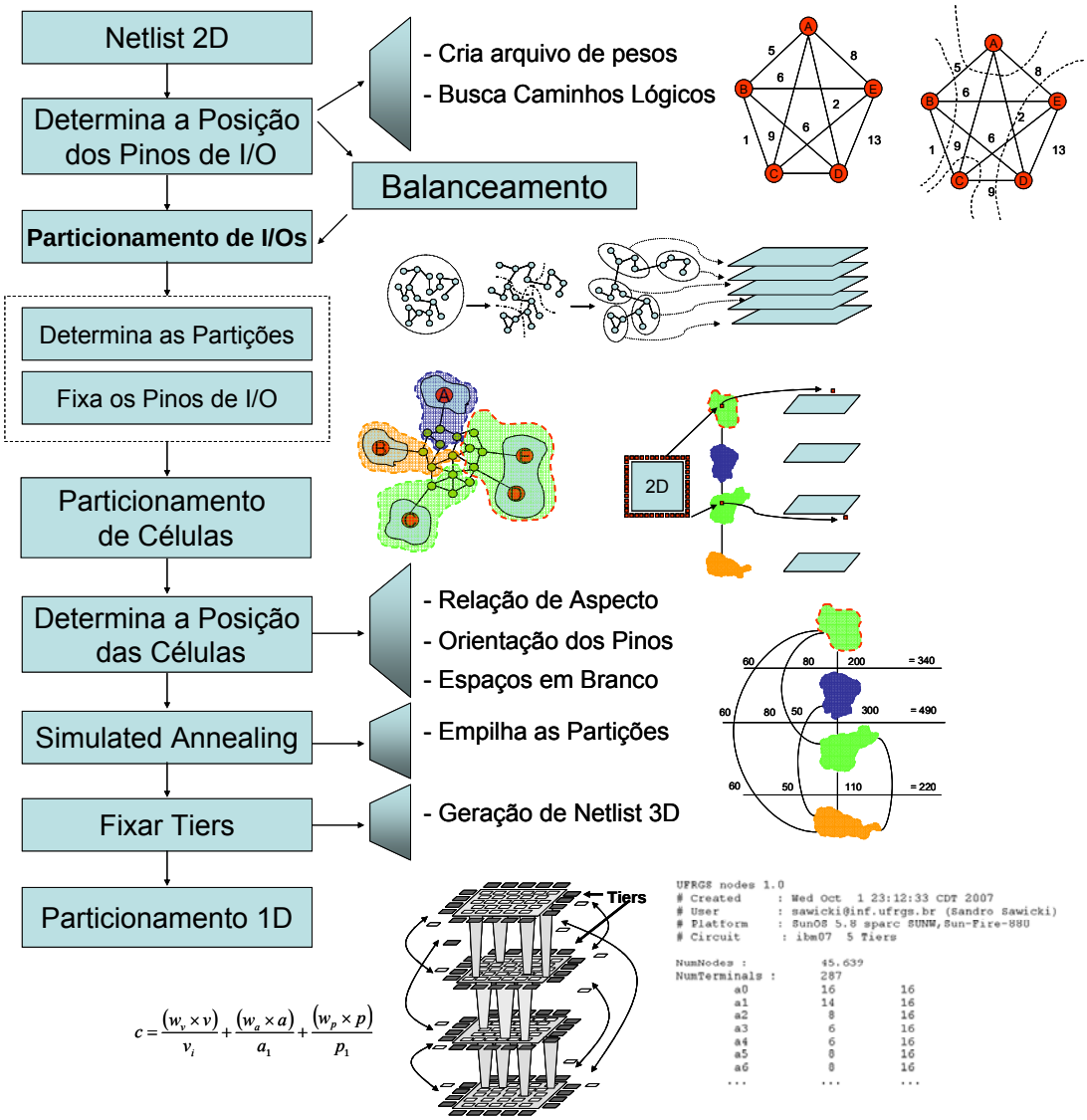


Figura 5.1: Fluxo de migração e otimização de circuitos 2D para circuitos 3D.

O LH-3D envolve diversos algoritmos que estão explicados em detalhes nas próximas seções. Os algoritmos que fazem parte do grupo LHp-3D, executam o particionamento em dois passos. O primeiro realiza a busca de informações sobre o circuito (*netlist*) que possam contribuir para o processo de migração de 2D para 3D. A informação retirada do circuito é a distância lógica entre pares de pinos de I/O. De posse dessa informação, inicia-se o *pré-particionamento.* Nesse passo, somente os pinos de I/O são particionados (conforme a distância lógica) e posicionados nas partições. O segundo passo do algoritmo é o particionamento das células. De acordo com a informação da posição dos pinos de I/O, as células próximas a eles são guiadas para a mesma partição. Detalhes de todos esses passos até a geração das *netlists* 3D estão descritos na Seção 5.4. Os algoritmos que compõem o grupo LHu-3D aumentam o desvio padrão do número de pinos de I/O entre as partições com o objetivo de reduzir ainda mais o número de vias-3D. Um estudo sobre o impacto dessa mudança está detalhado na Seção 5.5, juntamente com seus novos resultados.

O grupo de algoritmos LHr-3D, realiza um particionamento iterativo. Essa etapa reduz o número de vias que cruzam mais de duas *tiers* adjacentes convergindo para a

redução total do número de vias-3D. O desenvolvimento do processo de refinamento está descrito na Seção 4.6. O fluxo LH-3D, em todas as etapas, mantém balanceadas as áreas das *tiers* e os pinos de I/O.

5.4 Algoritmo para o Particionamento de Pinos de I/O (LHp-3D)

A ideia de particionar um bloco de lógica aleatória em duas ou mais *tiers* de um circuito 3D já foi explorada em outros estudos (DAVIS et al., 2005), (GOPLEN et al., 2005), (DENG et al., 2005). Nessa abordagem, o estágio de posicionamento particiona e distribui as células em *tiers* separadas. Teórica (BANERJEE et al., 2001) e empiricamente empiricamente (DAVIS et al., 2005), (DAS; FAN; CHEN; TAN; CHECKA; REIF, 2004), (ABABEI et al., 2005), mostra-se que circuitos 3D podem reduzir o tamanho dos fios.

Esse método assume que a borda de um bloco aleatório, em circuitos 2D, é delimitada por pinos de I/O. Também é assumido que esses pinos possam ser movidos para qualquer *tier.* Outros algoritmos de posicionamento de células são dirigidos pela localização fixa dos pinos. Os de posicionamento quadráticos (ALPERT; CHAN; HUANG; MARKOV; YAN, 1997), por exemplo, amplamente usados em pesquisas acadêmicas (VISWANATHAN et al., 2005) e indústria (VILLARRUBIA et al., 1990), requerem que a posição dos pinos já esteja determinada para que seja computada a solução.

Este tópico estuda o problema de particionamento de pinos de I/O e suas consequências no número de vias-3D e no tamanho dos fios (*wirelenght*). No que tange o conhecimento do autor, este é o primeiro trabalho que estuda o impacto, na solução final, do particionamento de pinos de I/O em duas ou mais *tiers*.

5.4.1 Definição do Problema

Antes do posicionamento, uma *netlist* 2D N_l é composta por um conjunto de células $G = \{g_1, g_2, g_3, \ldots, g_n\}$, um conjunto de pinos de I/O $P = \{p_1, p_2, p_3, \ldots, p_m\}$ e um conjunto de redes $N = \{n_1, n_2, n_3, \ldots, n_o\}$. Um hipergrafo H_g representa a *netlist*, onde $G \cup P$ é o conjunto de nodos, e N é o conjunto de hiperarestas. A posição fixa de cada pino de I/O p_i é representada por $X_{[i]}$ e $Y_{[i]}$ ($i \leq m$), e sua orientação, por $O_r(p_i) \in \{north, south, east, west\}$. A área A (altura H, largura W) a informação de seu canto inferior esquerdo pela coordenada (x_{ini}, y_{ini}). Usualmente, os pinos de I/O cobrem toda a borda do circuito. A relação de espaços em branco S na área do posicionamento é alcançada pela subtração da área total de células (G_a) pela área disponível dentro dos pinos de I/O. A relação de aspecto A_r é computada pela divisão de W por H.

Se Z é o conjunto dos números que representam as *tiers* $\{1, 2, \ldots, z\}$, então, o problema pode ser definido por essa diretriz: dado uma *netlist* 2D N_l com pinos de I/O fixos, encontre o conjunto de *tiers* $T = \{t_1, t_2, \ldots, t_z\}$ e seus correspondentes A_i, A_{ri}, G_{ai}, W_i, H_i, P_i, S_i, O_{ri}, X_i e Y_i ($i \leq z$) tal que

$$P_1 \cup P_2 \cup ... \cup P_t = P \quad (1)$$

$$\forall (a,b \in Z)(a \neq b \rightarrow P_a \cap P_b = \varnothing) \quad (2)$$

$$\forall (i \in Z) S_i \approx S \quad (3)$$

$$\forall (i \in Z) \forall (j \in Z) W_i = W_j \wedge H_i = H_j \quad (4)$$

$$\forall (i \in Z) A_{ri} \approx A_r \quad (5)$$

$$\forall (i \in Z)(\forall a \in P_i (O_{ri}(a) = O_r(a))) \quad (6)$$

$$\forall (t \in Z)(\forall a,b \in P_t (O_r(a) = O_r(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b])) \quad (7)$$

$$\forall (t \in Z)(\forall a,b \in P_t (O_r(a) = O_r(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b])) \quad (8)$$

Em outras palavras, cada *tier* terá seu próprio conjunto de pinos de I/O (equações 1 e 2), os espaços em branco (*whitespaces*) e a relação de aspecto (*aspect ratio*) preservados na mesma proporção do circuito 2D original (equações 3, 4 e 5), assim como a orientação e a ordem dos pinos (equações 6, 7, e 8).

5.4.2 Algoritmo Proposto Baseado no Menor Caminho Lógico

Seja $L_d(p_i, p_j)$ o tamanho do menor caminho em H_g de p_i para p_j (por exemplo, a distância lógica entre p_i e p_j), pode-se descrever o algoritmo de particionamento de pinos de I/O da forma detalhada abaixo:

Algoritmo Proposto (LHp-3D)		
1	Computar $L_d(i, j)$ $\forall i,j \in P$	
2	Criar um grafo completo P_g tal que P seja o conjunto de nodos e $L_d(i, j)$ $(i, j \in P)$ seja o custo das aresta conectando os nodos i e j.	
3	Executar o particionamento de P_g em $P_1, P_2, ... P_z$ buscando a minimização do corte (*min-cut*) e o balanceamento de pinos entre as partições.	
4	$\forall (i \in Z) G_{a_i} \approx \frac{G_a}{z}$	(9)
5	$\forall (i \in Z) A_i = G_{a_i} \times (1 + S_i)$	(10)
6	$\forall (i \in Z) W_i = \sqrt{A_i} \times A_{ri}; H_i = \frac{\sqrt{A_i}}{A_r}$	(11)
7	$\forall (i \in Z) \forall (p \in P_i) X_i[p] = x_{ini} + \frac{(X[p] - x_{ini}) \times W_i}{W}$	(12)
8	$\forall (i \in Z) \forall (p \in P_i) Y_i[p] = y_{ini} + \frac{(Y[p] - y_{ini}) \times H_i}{H}$	(13)
9	Legalizar os pinos de I/O	(14)

Algoritmo 5.1: Algoritmo LHp-3D

Considerando-se que, em um circuito real, os *fanouts* são limitados, uma simples busca BFS terá uma complexidade *O(n)*. Portanto, usando-se um algoritmo de busca simples para computar o custo de um pino p_i para todos $p \in P$, a complexidade será $O_{(mn)}$.

A busca pelo menor caminho em um grafo (*shortest-path*) é um problema bem conhecido na ciência da computação. É utilizada em aplicações reais, tais como mapas rodoviários, jogos e roteamento de linhas telefônicas, de circuitos integrados e de redes, etc. (DIJKSTRA, 1959), (HART; NILSSON; RAPHAEL, 1968), (SHERWANI, 1998), (JOHANN; CALDWELL; KAHNG; REIS, 2001).

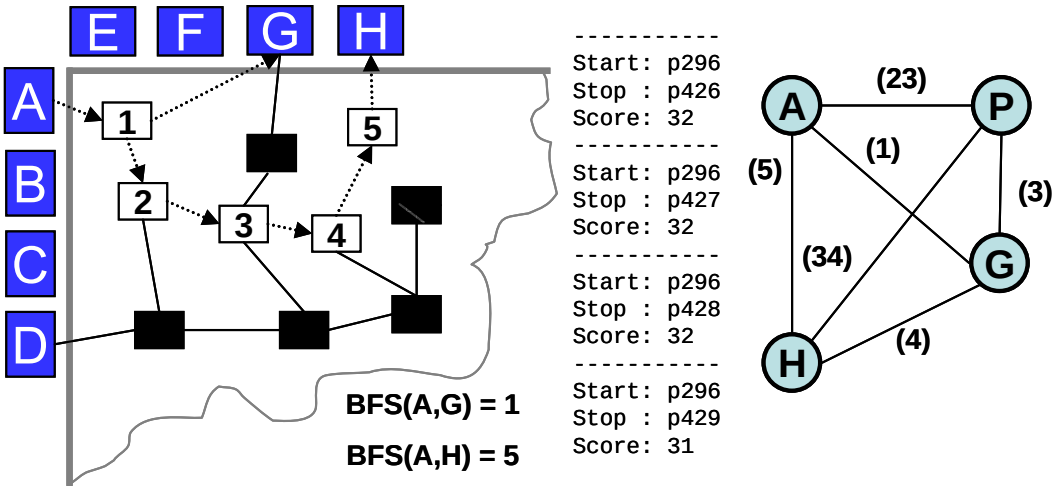


Figura 5.2: Ilustração do menor caminho entre dois pinos de I/O e seus pesos correspondentes no grafo completo.

Os valores de L_d são usados para criar um grafo completo P_g conectando todos os pares de pinos de I/O, como mostra a Figura 5.2. No terceiro passo, usa-se a ferramenta hMetis (KARYPIS et al., 1997) (HMETIS HOMEPAGE, 2007) para particionar os pinos de I/O (Figura 5.3). Essa ferramenta aceita a inserção de valores (pesos) para as células e arestas. Atribui-se o inverso do custo das arestas como peso. Além disso, é imposto um balanceamento rígido a fim de manter uma quantidade similar de pinos de I/O entre as *tier*.

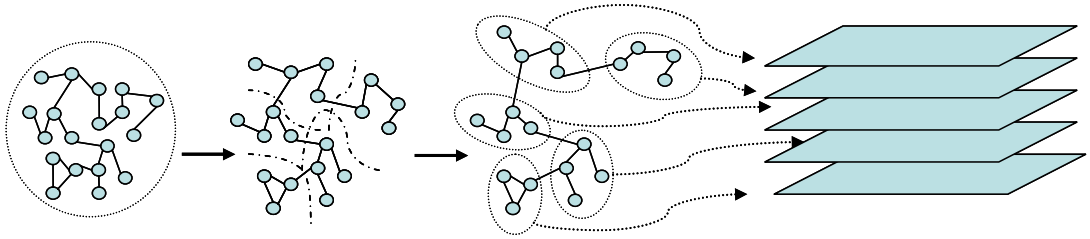


Figura 5.3: *Netlist* inicial seguindo o fluxo de particionamento.

O quarto passo é realizado por meio da divisão do número total da área de células pelo número de *tiers*. Os passos 5 e 6 computam a área destas, tal que a relação de aspecto (*aspect ratio*) e os espaços em branco (*whitespaces*) do circuito 2D original sejam preservados. Nesse ponto, a nova relação de aspecto ou espaços em branco pode ser usada.

Finalmente, os passos 7 e 8 computam as coordenadas *x* e *y* dos pinos de I/O para as *tiers* a que são destinados. A orientação original é preservada, de modo que os pinos originais sejam mapeados para as *tiers* de tamanho menor. Por fim, a legalização (passo 9) é executada para garantir que não existam *overlaps* entre os pinos de I/O.

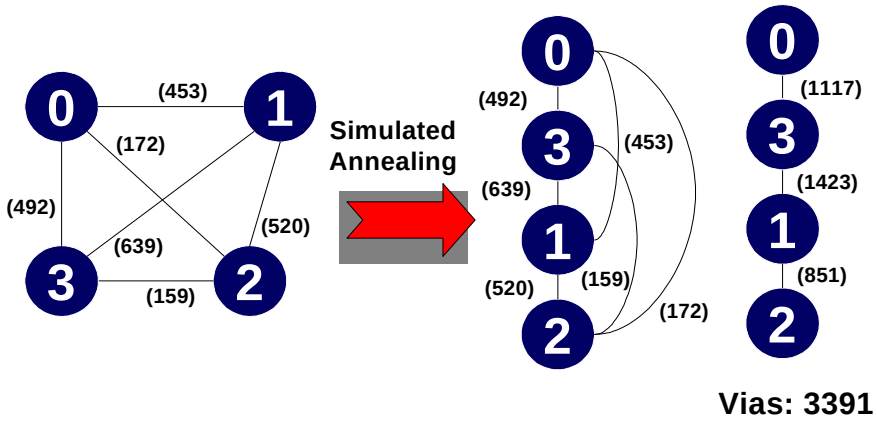


Figura 5.4: Fluxo ilustrando os custos das arestas e a execução do *simulated annealing* para minimizar o número de vias-3D.

Após o término do passo 9, é necessário descobrir qual a melhor sequência de *tiers* que auxilia a minimização de vias. Utilizou-se o algoritmo de *simulated annealing* para otimizar o número total de vias-3D, como mostra a Figura 5.4. A Figura 5.5 mostra duas ilustrações. A primeira, localizada à esquerda, representa a migração de um circuito 2D para um 3D. A segunda, à direita, apresenta uma *netlist*, sem definições, transformada em um circuito 3D.

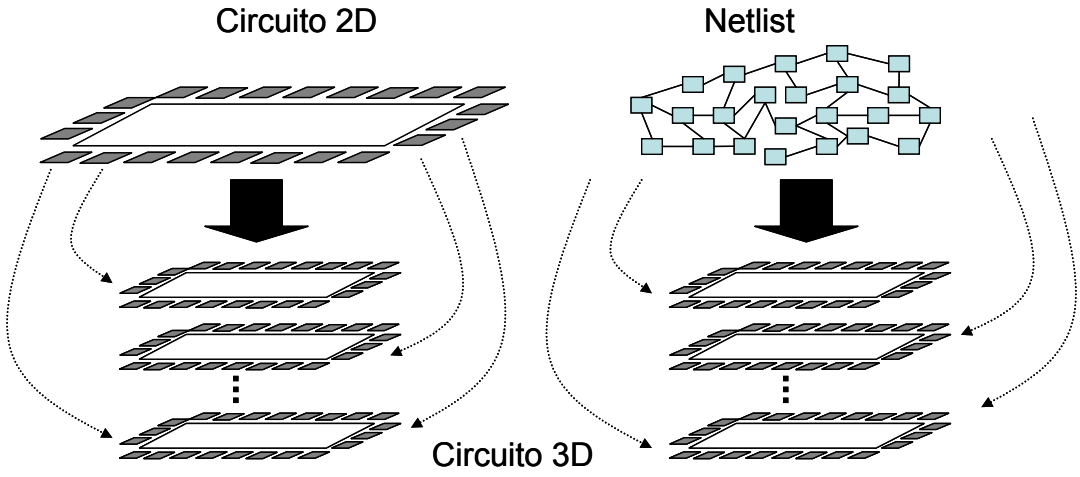


Figura 5.5: Migração de circuitos 2D para circuitos 3D.

5.4.2.1 Interpretação e Formato de Arquivos do Fluxo Proposto

Como mencionado na Seção 5.4.2, o processo inicia-se com a leitura do circuito 2D no formato *bookshelf* (*nodes*, *net*, *pl*, *scl, aux*), e as informações de número e tamanho

de células, quantidade de redes e pinos de I/O são carregadas e organizadas na estrutura de dados da ferramenta.

O formato do arquivo segue a seguinte definição: um hipergrafo $H = (V, E)$, com V vértices e E hiper-redes, é armazenado em um arquivo texto contendo $|E| + 1$ linhas, caso não existam pesos nos vértices, e $|E| + |V| + 1$ linhas caso existam. Qualquer linha que iniciar com "%" é considerada comentário e descartada.

A primeira linha pode conter dois ou três números inteiros. O primeiro é o número de hiper-redes $(|E|)$, o segundo é o número de vértices $(|V|)$, e o terceiro (f) informa o tipo do hipergrafo. Dependendo do valor de f, o hipergrafo H pode ter peso tanto nas hiper-redes quanto nos vértices, ou em ambos. Caso H não tenha nenhum peso atribuído, todas as hiper-redes e vértices têm o mesmo peso e, com isso, f é omitido.

Depois da primeira linha, as restantes $|E|$ linhas armazenam os vértices contidos em uma hiper-rede (uma linha por hiper-rede). Segundo o fluxo proposto, o primeiro particionamento necessita da informação do peso atribuído a cada uma das arestas. Nesse caso, o arquivo de entrada gerado segue o formato ilustrado na Figura 5.6 (a), sendo todo o peso atribuído à aresta armazenada sempre na primeira posição da linha.

Após a execução da busca pelo menor caminho lógico entre pares de pinos de I/O, um grafo completo é criado e os pesos das distâncias são adicionados a cada uma das arestas. Na primeira linha, o primeiro número inteiro corresponde à quantidade de hiper-redes existentes, o segundo, ao número de vértices, e o terceiro indica que somente pesos nas arestas serão considerados. As próximas linhas representam as hiper-redes, sendo os números inteiros correspondentes aos vértices do hipergrafo. Terminado o particionamento dos pinos de I/O, um novo arquivo é gerado contendo duas informações em cada uma das n linhas, sendo $n = |V|$ (nesse caso, somente pinos). A primeira informação é do pino de I/O, e a segunda, a do número da partição. Esse arquivo é convertido como uma nova entrada (com vértices fixos) para que os pinos sejam fixados antes do particionamento das células.

0	pin_1
0	pin_2
2	pin_3
4	pin_4
2	pin_5
1	pin_6
3	pin_7
- 1	**cell_1**
- 1	**cell_2**
- 1	**cell_3**
0	pin_7
3	pin_8

(a) Pinos Fixos e Células Livres

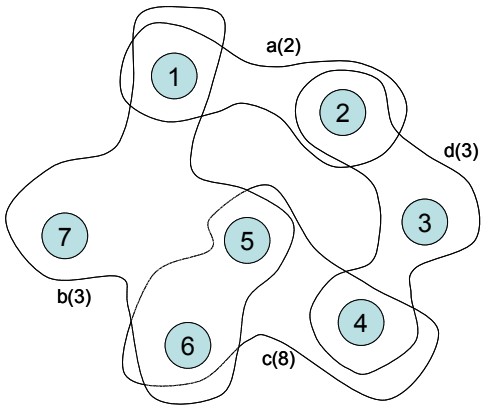
0	pin_1
0	pin_2
2	pin_3
4	pin_4
2	pin_5
1	pin_6
3	pin_7
4	**cell_1**
2	**cell_2**
3	**cell_3**
0	pin_7
3	pin_8

(b) Pinos e células particionadas

Figura 5.6: (a) Arquivo gerado após o primeiro particionamento, fixando os pinos de I/O; (b) com base na posição dos pinos, as células são particionadas e fixadas nas partições.

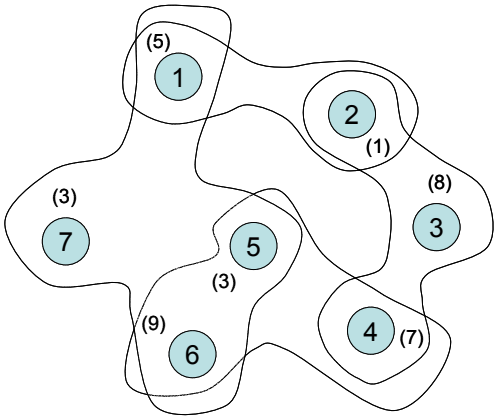
No arquivo de vértices fixos, t é o valor inteiro correspondente ao número da partição. Assim, quando $t \geq 0$, fixa-se o vértice na partição corrente, e quando $t < 0$, os vértices ficam livres para migrar para qualquer partição (esse caso acontece na segunda

fase, em que os pinos estão fixos, e as células, livres para o próximo particionamento), como ilustra a Figura 5.6 (b). Cada linha do arquivo corresponde a um vértice particionado.




```
Grafo
4  7  1
2  1  2
3  1  7  5  6
8  5  6  4
7  2  3  4
```

Figura 5.7: Hipergrafo contendo pesos nas hiper-redes e seu formato de arquivo correspondente.




```
Grafo
4  7  10
1  2
1  7  5  6
5  6  4
2  3  4
5
1
8
7
3
9
3
```

Figura 5.8: Hipergrafo contendo pesos nos vértices e seu formato de arquivo correspondente.

Na etapa de particionamento de células (segundo passo), o critério a ser considerado é o balanceamento entre as partições, o que pode envolver o número de células em cada partição ou considerar a área de cada célula. Como, em circuitos *standard cells*, as células têm tamanhos variados (mesma altura e largura diferentes), considerou-se a área destas em cada partição, como ilustra a Figura 5.9. No momento da movimentação de uma célula para outra partição, a área da célula é subtraída da área da partição atual e incrementada na área da partição-destino. Utilizou-se um balanceamento rígido de células para manter as áreas das partições equilibradas. Detalhes do algoritmo de balanceamento executado nessa etapa estão descritos no item 5.5.

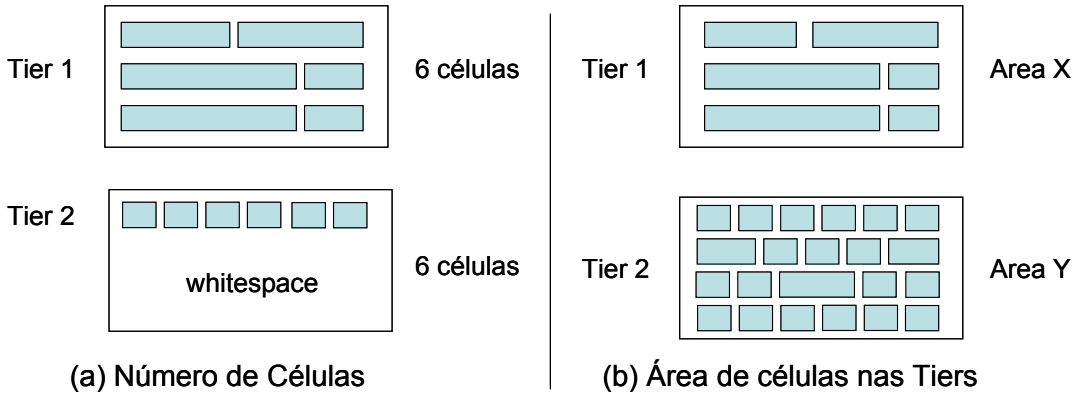


Figura 5.9: (a) Balanceamento de *tiers* baseado no número de células; (b) balanceamento de *tiers* baseado na área de células contidas nas *tiers*.

5.4.3 Resultados Experimentais

O objetivo deste trabalho é estudar o impacto causado pelo particionamento de pinos de I/O na área e número de vias do circuito. Os resultados obtidos são descritos abaixo:

1. O particionamento de *Hg* é executado visando-se a minimizar os cortes entre as partições (*min-cut*). Os pinos de I/O particionados são usados como nodos fixos em cada uma das *tiers*. O particionamento é realizado com o uso da ferramenta hMetis, configurada para manter o balanceamento das áreas do *tiers* o mais rígido possível.
2. Um conjunto de *benchmarks* é gerado e posicionado independentemente. O número de vias-3D é computado, mas o custo das conexões verticais são ignorados pelo posicionador.

Utilizou-se *quadratic placement* (VISWANATHAN et al., 2005), para o posicionamento global, e *simulated annealing* para o posicionamento detalhado.

O algoritmo de particionamento de pinos de I/O foi comparado com outros dois algoritmos que seguem a mesma formulação do problema descrita na Seção 5.4.1. O primeiro método é executado por meio do algoritmo hMetis (*state-of-the-art*) e chamado *unlocked pins*. Neste, permite-se que a partição dos pinos seja realizada livremente junto com as células do circuito. O segundo algoritmo é chamado de *alternate pins*. Trata-se de um particionamento pseudoaleatório que divide os pinos alternadamente em cada uma das *tiers* com o objetivo de preservar o balanceamento inicial dos pinos de I/O. Ambos os algoritmos são ilustrados na figura 4.5 e substituem os passos 1, 2, 3 e 4 do fluxo, fazendo-se necessários apenas os passos 5, 6, 7, 8 e 9.

As tabelas 4.2, 4.3 e 4.4 mostram os resultados de experimentos realizados com duas *tiers* e com o uso dos *benchmarks* ISPD 2004 (ISPD IBM BENCHMARK, 2004). A coluna que apresenta a área da *tier* é calculada antes da partição atual de células. A *tier* com maior área é usada como padrão para as demais. A área total é simplesmente ***n* vezes a área da maior *tier***. O *wirelenght* total é a soma do *wirelenght* encontrado pelo posicionador em cada *tier* separadamente. O número de vias e pinos de I/O também são relatados nas tabelas. Estas mostram o desvio padrão do número de pinos a fim de avaliar os balanceamentos. Analisando-se as tabelas 4.2, 4.3 e 4.4, pode-se fazer as seguintes observações:

- O número de pinos de I/O é muito bem balanceado com o método *alternate pins*, tendo uma média de desvio padrão menor do que 1. A média atingida com o algoritmo proposto por este trabalho é de 5 pinos. Contudo, o método ***unlocked pins (hMetis)*** **apresenta um desbalanceamento no número de pinos que o invalida completamente** (média do desvio padrão de 150 pinos).
- O número de vias encontradas pelo método *unlocked pins* e *alternate pins* é sempre pior do que o método LHp-3D, mostrando que **um particionamento de pinos de I/O simplista pode piorar o algoritmo de minimização de cortes (*min-cut*).**
- O número de vias do algoritmo LHp-3D é sempre menor do que o dos outros métodos. Este é um resultado importante, pois mostra que este **método auxilia o particionamento das células, de modo que se encontre o melhor corte e um balanceamento eficiente entre as *tiers*.** O resultado do tamanho dos fios obtido pelo algoritmo proposto é menor, em média, do que o encontrado com o método *alternate pins.* O método *unlocked pins* leva ao melhor *wirelength*. Contudo, não foram computadas as conexões o tamanho das conexões verticais.

Tabela 5.2: Resultados experimentais encontrados com o uso do algoritmo LHp-3D, com duas *tiers*

Dados dos Circuitos				**Dados do Particionamento (LHp-3D)**				
Benchmarks	**#I/Os**	**Área 2D**	**Área Tier**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/Os**	**#Vias**
ibm01	246	2.380,800	1.209,856	2.11E+06	120	126	4	374
ibm02	259	3.064,208	1.547,568	4.50E+06	126	133	5	396
ibm03	283	3.751,968	1.896,128	6.48E+06	138	145	5	1.064
ibm04	287	4.782,848	2.417,664	7.02E+06	147	140	5	735
ibm06	166	4.106,592	2.078,784	7.51E+06	81	85	3	1.059
ibm07	287	7.136,672	3.612,960	1.23E+07	140	147	5	992
ibm08	286	7.403,840	3.799,840	1.07E+07	140	146	4	1.298
ibm09	285	8.617,104	4.328,208	1.41E+07	139	146	5	699
Média	264	5.155,504	2.590,126	8.08E+06	129	134	5	826

Tabela 5.3: Resultados experimentais encontrados com o uso do algoritmo hMetis, com duas *tiers*

Dados dos Circuitos				**Dados do Particionamento (hMetis)**				
Benchmarks	**#I/Os**	**Área 2D**	**Área Tier**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/Os**	**#Vias**
ibm01	246	2.380,800	1.209,856	2.14E+06	0	246	174	441
ibm02	259	3.064,208	1.578,784	4.39E+06	259	0	183	547
ibm03	283	3.751,968	1.897,280	6.22E+06	283	0	200	1.146
ibm04	287	4.782,848	2.414,592	7.30E+06	287	0	203	738
ibm06	166	4.106,592	2.078,784	7.73E+06	75	91	11	1061
ibm07	287	7.136,672	3.596,112	1.13E+07	0	287	203	994
ibm08	286	7.403,840	3.737,920	1.03E+07	127	159	23	1.324

ibm09	285	8.617,104	4.326,144	1.40E+07	0	285	202	806
Média	264	5.155,504	2.583,684	7.91E+06	129	134	150	882

Tabela 5.4: Resultados experimentais encontrados com o uso do algoritmo *Alternate Pins*, com duas *tiers*

Dados dos Circuitos				**Dados do Particionamento (Alternate Pins)**				
Benchmarks	**#I/Os**	**Área 2D**	**Área Tier**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/Os**	**#Vias**
ibm01	246	2.380,800	1.209,856	2.35E+06	123	123	0	428
ibm02	259	3.064,208	1.548,884	4.30E+06	130	129	1	503
ibm03	283	3.751,968	1.877,280	6.13E+06	142	141	1	1.099
ibm04	287	4.782,848	2.414,592	7.54E+06	144	143	1	750
ibm06	166	4.106,592	2.098,784	7.37E+06	83	83	0	1.075
ibm07	287	7.136,672	3.596,112	1.18E+07	144	143	1	1.049
ibm08	286	7.403,840	3.797,920	1.11E+07	143	143	0	1.307
ibm09	285	8.617,104	4.326,144	1.43E+07	143	142	1	780
Média	264	5.155,504	2.583,684	8.10E+06	132	131	0,63	874

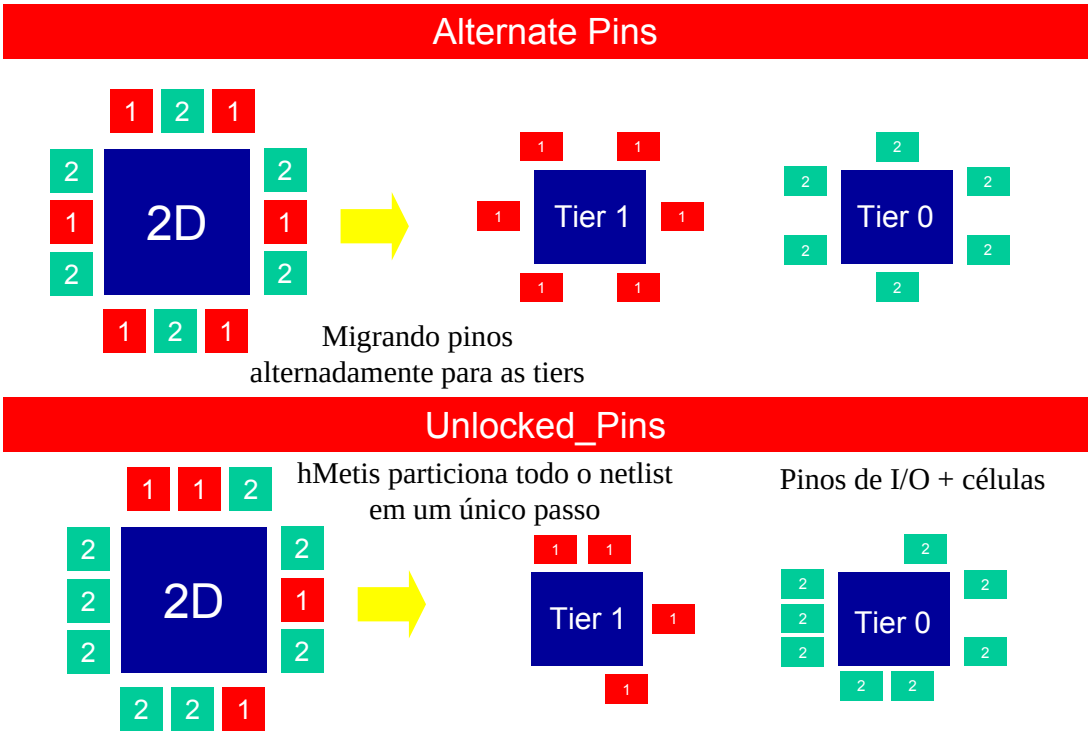


Figura 5.10: Algoritmo *Alternate Pins* e algoritmo *Unlocked Pins (hMetis).*

É importante ressaltar que o *wirelenght* medido é impreciso, pois as conexões entre as *tiers* foram desconsideradas. Uma das vantagens obtidas pelo método proposto é a de melhorar o resultado das conexões verticais (vias-3D).

As tabelas 4.5 e 4.6 consideram experimentos com número de *tiers* de 2 a 5.

A Tabela 5.7 apresenta a média dos resultados do desvio padrão do número de pinos de I/O em cada uma das cinco partições. O algoritmo *alternate pins* apresenta um resultado excelente, obviamente porque a métrica utilizada teve essa finalidade. O método *unlocked pins* teve um desvio padrão enorme; em muitos casos, várias *tiers* ficaram sem nenhum pino de I/O. **Esse desbalanceamento invalida o método *unlocked pins***. O algoritmo proposto por este trabalho se aproxima do balanceamento ótimo.

A Tabela 5.5 permite a comparação dos resultados experimentais quanto ao número total de vias-3D, quando se utilizam os três algoritmos de particionamento com duas, três, quatro e cinco *tiers*. As médias demonstram que **os métodos simplistas de particionamento de pinos de I/O não são eficientes na minimização do corte entre as *tiers.*** Contudo, percebe-se que as informações do menor caminho lógico entre os pinos melhoram a qualidade da heurística de minimização de vias-3D, mantendo o equilíbrio entre as *tiers.*

Tabela 5.5: Número total de vias-3D

	hMetis				***Alternate pins***				**LHp-3D (Proposto)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	441	735	838	1.439	428	881	977	1.372	374	525	837	1.162
ibm02	547	882	1.214	1.600	503	829	1.340	1.691	396	747	1.156	1.533
ibm03	1.146	2.257	2.693	4.020	1.099	2.530	3.602	4.266	1.064	2.174	2.610	3.974
ibm04	738	1.583	2.516	3.202	750	1.629	2.461	4.275	735	1.511	2.371	2.852
ibm05	2.417	4.651	6.653	9.651	2.576	5.428	7.037	12.400	2.258	4.311	6.489	9.193
ibm06	1.061	1.827	3.128	3.566	1.075	1.729	3.429	3.507	1.059	1.642	2.934	3.477
ibm07	994	2.038	3.302	4.605	1.049	3.423	3.482	6.523	992	2.050	3.219	4.400
ibm08	1.324	2.814	4.184	5.698	1.307	3.431	4.183	6.327	1.298	2.697	4.018	5.346
ibm09	806	1.904	2.763	3.518	780	2.186	3.757	3.556	699	1.872	2.495	3.343
ibm10	1.771	3.555	4.675	7.116	1.821	4.062	4.358	8.492	1.490	2.661	4.004	5.216
ibm11	1.490	2.581	3.958	5.697	1.494	3.629	4.923	7.437	1.190	2.240	3.685	4.620
ibm12	2.594	4.470	7.259	9.187	2.556	5.569	8.996	12.515	2.293	4.094	6.581	8.191
ibm13	1.193	2.298	3.264	4.557	1.170	2.912	4.618	4.874	1.042	1.893	3.099	3.742
ibm14	2.171	4.561	6.584	8.085	2.310	5.090	7.564	10.113	2.121	3.886	5.342	6.667
ibm15	3.002	7.863	9.082	11.707	3.126	7.970	11.144	13.857	2.890	4.827	7.022	9.283
ibm16	2.237	5.816	6.235	9.300	2.280	6.216	9.525	10.903	2.102	5.476	5.918	8.920
Média	1.496	3.111	4.272	5.809	1.520	3.595	5.087	7.007	1.375	2.663	3.861	5.120

Tabela 5.6: Número máximo de vias-3D entre duas *tiers* adjacentes

	hMetis				***Alternate pins***				**LHp-3D (Proposto)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	441	467	377	573	428	483	406	480	369	305	322	406
ibm02	547	496	485	552	503	469	498	553	396	413	403	594

ibm03	1.146	1.143	1.021	1.334	1.099	1.320	1.485	1.210	1.059	1.187	1.086	1.225
ibm04	738	862	1.067	1.039	750	913	1.033	1.454	735	887	992	887
ibm05	2.417	2.765	2.478	2.712	2.576	2.814	2.526	3.974	2.258	2.203	2.469	2.729
ibm06	1.061	924	1.134	935	1.075	915	1.193	937	1.059	849	1.135	948
ibm07	994	1.980	1.510	1.525	1.049	2.050	1.590	2.402	992	1.332	1.433	1.524
ibm08	1.324	1.436	1.445	1.788	1.307	1.919	1.448	1.833	1.298	1.448	1.397	1.610
ibm09	806	1.598	1.092	1.249	780	1.356	1.684	1.137	699	1.057	1.008	1.075
ibm10	1.771	1.883	1.741	1.986	1.821	2.247	1.724	2.898	1.490	1.450	1.590	1.750
ibm11	1.490	1.909	1.810	2.230	1.494	1.856	1.802	2.610	1.190	1.485	1.605	1.719
ibm12	2.594	2.820	2.747	2.962	2.556	3.160	3.113	4.205	2.293	2.278	2.422	3.173
ibm13	1.193	1.606	1.500	1.755	1.170	1.611	1.905	1.954	1.042	1.269	1.548	1.781
ibm14	2.171	2.375	2.307	2.881	2.310	2.619	3.274	3.283	2.121	2.272	2.248	2.459
ibm15	3.002	4.188	3.377	4.099	3.126	4.207	4.385	4.163	2.890	2.857	3.199	3.395
ibm16	2.237	3.185	2.266	3.355	2.280	3.704	3.794	3.443	2.102	2.164	2.212	2.625
Média	1.496	1.852	1.647	1.936	1.520	1.978	1.991	2.284	1.375	1.466	1.567	1.744

Tabela 5.7: Desvio padrão do número de pinos de I/O

Algoritmo	***hMetis***	***Alternate pins***	***LHp-3D***
Média σ I/O 2 *tiers*	233	0,4	7
Média σ I/O 3 *tiers*	252	0,4	6
Média σ I/O 4 *tiers*	177	0,4	5
Média σ I/O 5 *tiers*	189	0,4	6
Média total σ I/O	**213**	**0,4**	**6**

Os resultados expressos na Tabela 5.5 podem ser visualizados por meio de um gráfico na Figura 5.11. O gráfico traça a relação entre o número total de vias-3D e o de *tiers*. Percebe-se que a linha que representa o algoritmo proposto segue sempre com o número de vias-3D menor que o dos demais algoritmos de particionamento.

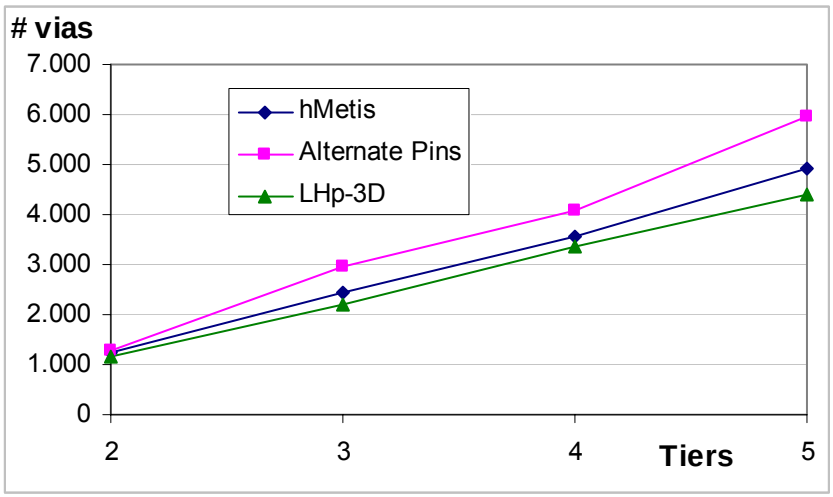


Figura 5.11: Número de vias-3D

O representado na

Tabela 5.6 também pode ser expresso como no gráfico da Figura 5.12, o qual relaciona o número máximo de vias-3D com o número de *tiers.* O resultado da redução do número máximo de vias é uma consequência natural da redução do número total de vias. Logo, os valores de redução observados na Tabela 5.5 e

Tabela 5.6 tem um comportamento similar. A linha que representa o número máximo de vias do algoritmo proposto segue, em todas as *tiers,* sempre abaixo das que representam os dos demais algoritmos. Percebe-se que seu crescimento é diferente dos demais, pois o número de vias-3D aumenta de forma quase linear.

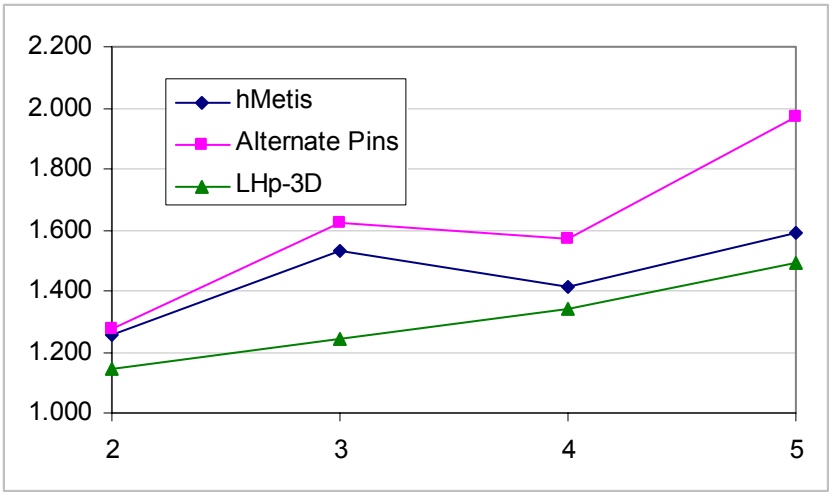


Figura 5.12: Número máximo de vias-3D entre duas *tiers* adjacentes.

A

Tabela 5.6 mostra o número máximo de vias-3D do algoritmo proposto comparado com os algoritmos *alternate pins* e *unlocked pins*. A tabela considera a relação do número máximo e da área ocupada com o tamanho das vias em tecnologia SOI e *Bulk.* Analisando-se a

Tabela 5.6, pode-se fazer as seguintes considerações:

- As tecnologias baseadas em *Bulk* provocam desperdício de área ativa, a cada inserção de vias-3D. Isso é devido ao tamanho da área das vias (cerca de 50µm). Existem casos em que a área ocupada pelas vias é maior do que a área da *tier.* Assim, uma importante conclusão é a necessidade de reduzir o número de vias-3D em tecnologia CMOS *Bulk.*
- Na tecnologia baseada em SOI, o número de vias-3D alcança cerca de 3% da área do circuito.

Tabela 5.8: Comparação dos três algoritmos considerando-se a área das *tiers* em relação ao número máximo de vias-3D, em tecnologias SOI e *Bulk* baseada nos dados da

Tabela 5.6.

#Tiers	Algoritmo	Area Tier	Max #Vias	Vias-3D (Bulk) 50µm		Vias-3D (SOI) 5µm	
2	*LHp-3D*	3.833.189	1.375	3.437.500	90%	34.375	1%
3		2.595.794	1.466	3.665.000	141%	36.650	1%

4		1.934.213	1.567	3.917.500	203%	39.175	2%
5		1.559.209	1.744	4.360.000	280%	43.600	3%
2	*hMetis*	3.844.129	1.496	3.740.000	97%	37.400	1%
3		2.587.478	1.852	4.630.000	179%	46.300	1%
4		1.917.220	1.667	4.167.500	217%	41.675	2%
5		1.553.191	1.936	4.840.000	312%	48.400	3%
2	*Alternate pins*	3.835.153	1.520	3.800.000	99%	38.000	1%
3		2.583.077	1.978	4.945.000	191%	49.450	2%
4		1.930.053	1.991	4.977.500	258%	49.775	3%
5		1.555.691	2.284	5.710.000	367%	57.100	3%

A Tabela 5.9 apresenta o somatório de todas as arestas que cruzam duas, três, quatro e cinco partições. Por exemplo, supondo que exista três partições, P_1, P_2 e P_3, a soma se dá pelas arestas que cruzam por: P_1 e P_2; P_1 e P_3; P_2 e P_3. Analisando os dados percebe-se que a ideia de usar os pinos de I/O como “pistas” para o particionador melhora a qualidade dos resultados de *min-cut.* Se comparada com os resultados de hMetis, essa redução, em média, foi de 10%, 11%, 6% e 6% para duas, três, quatro e cinco partições, respectivamente, o que comprova a eficácia de utilizar a informação dos pinos, pois as células próximas logicamente dos pinos migram para a mesma partição.

A redução do número de vias é consequência da melhoria do corte. No entanto, não há uma relação direta porque as arestas do hipergrafo têm pesos diferentes associados ao número de vias que elas representam.

Tabela 5.9: Somatório das arestas que cruzam duas, três, quatro e cinco partições usando os algoritmos LHp-D e hMetis

Benchs	**hMetis** 2	**LHp-3D** 2	**HMetis** 3	**LHp-3D** 3	**hMetis** 4	**LHp-3D** 4	**hMetis** 5	**LHp-3D** 5
ibm01	441	374	601	464	588	578	824	815
ibm02	547	396	714	614	873	841	1.131	1.042
ibm03	1.146	1.064	1.700	1.626	1.879	1.814	2.200	2.181
ibm04	738	735	1.474	1.386	2.009	1.918	2.229	2.170
ibm05	2.417	2.258	3.708	3.292	4.068	4.012	4.577	4.362
ibm06	1.061	1.056	1.522	1.376	1.871	1.646	1.897	1.871
ibm07	994	992	2.009	2.007	2.550	2.380	2.915	2.951
ibm08	1.324	1.298	2.410	2.280	2.738	2.654	3.207	3.154
ibm09	806	699	1.791	1.544	2.016	1.966	2.375	2.251
ibm10	1.771	1.490	2.719	2.114	3.080	2.874	3.929	3.277
ibm11	1.490	1190	2245	2026	2923	2708	3382	3151
ibm12	2.594	2293	3795	3314	5106	4570	5588	4930
ibm13	1.193	1042	1952	1687	2391	2272	2701	2534
Média	1.271	1.145	2.049	1.825	2.469	2.326	2.843	2.668

5.4.4 Resumo do Fluxo Proposto (LHp-3D)

O fluxo do método de particionamento de pinos de I/O proposto está apresentado na Figura 5.13. Basicamente, ele inicia com a leitura dos *benchmarks* IBM ISPD, no formato *bookshelf*, com *pads* de I/O. A distância lógica entre os pares de pinos

é calculada por meio da varredura realizada pelo algoritmo BFS. Com o resultado obtido, executa-se o primeiro particionamento. O particionador considera os valores encontrados como pesos nas arestas que ligam os pares. Depois do particionamento, cada pino de I/O é fixado nas partições determinadas pelo hMetis.

Estando os pinos fixados, as células são particionadas e tendem a ficar perto deles, o que diminui o corte entre as partições. A cada célula e pino é atribuída uma partição. O algoritmo recupera as informações da *netlist* inicial relacionando a posição inicial dos pinos, a orientação destes, o espaço em branco do circuito e a relação de aspecto. As proporções são mantidas e a área de cada *tier* é calculada. Opta-se pela *tier* com maior área, e essa área é aplicada demais. A posição das *tiers* é determinada através da execução do algoritmo de *simulated annealing*, o qual procura o arranjo que mais reduz o número de vias-3D.

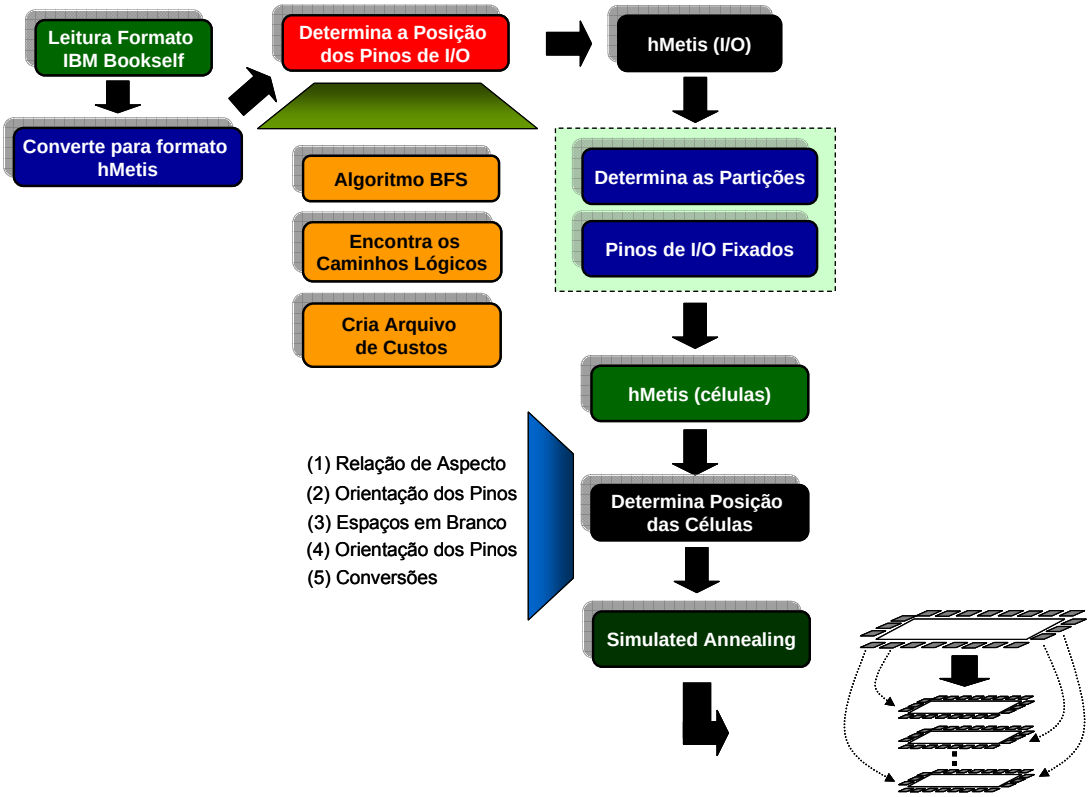


Figura 5.13: Fluxo do particionador de pinos de I/O (LHp-3D).

5.4.5 Considerações sobre o Algoritmo Proposto

Esta seção apresentou um método para adaptar o particionamento e posicionamento de pinos de I/O de circuitos 2D para circuitos 3D. Conforme a pesquisa realizada, este é o primeiro trabalho que foca esse problema e estuda o seu impacto na área do circuito, no balanceamento de pinos e no tamanho das conexões. O trabalho propõe que o particionamento de pinos de I/O seja realizado antes do das células. Neste trabalho, foi demonstrado empiricamente que fazer o particionamento de pinos junto com o das células cria uma irregularidade no número de pinos entre as *tiers*, o que invalida o método.

O algoritmo desenvolvido busca manter os pinos logicamente próximos, em uma mesma *tier*. Os resultados experimentais mostram que o método é eficiente e permite a

distribuição balanceada de pinos, diminuindo o tamanho dos fios e o número de vias-3D em comparação com os resultados obtidos por meio das outras abordagens.

De acordo com os resultados experimentais, um particionamento simplista de pinos de I/O pode aumentar o número de vias-3D. Além disso, percebe-se que, usando-se um particionamento regular (células e pinos de I/O juntos), obtém-se um desbalanceamento enorme de pinos. Conclui-se, então, que as informações do menor caminho lógico entre esses pinos pode ajudar a heurística de particionamento de células a minimizar o número de vias-3D.

5.5 Desequilibrando o Particionamento de Pinos de I/O (LHu-3D)

Esta seção mostra a modificação realizada no algoritmo de particionamento de pinos de I/O (LHp-3D) e os novos resultados alcançados. O objetivo é observar o comportamento das vias-3D através do desequilíbrio dos pinos de I/O entre as *tiers.* Essa etapa é chamada de LHu-3D e seu fluxo está ilustrado na Figura 5.14.

Na Seção 5.4, o algoritmo proposto obteve um bom equilíbrio do número de pinos de I/O entre as partições e uma redução significativa do número de vias-3D comparado com as outras abordagens estudadas. A modificação do algoritmo por meio do aumento do desequilíbrio dos pinos de I/O entre as *tiers* melhorou em cerca de 2% os resultados alcançados anteriormente. Detalhes dos experimentos estão descritos na Seção 5.5.1.

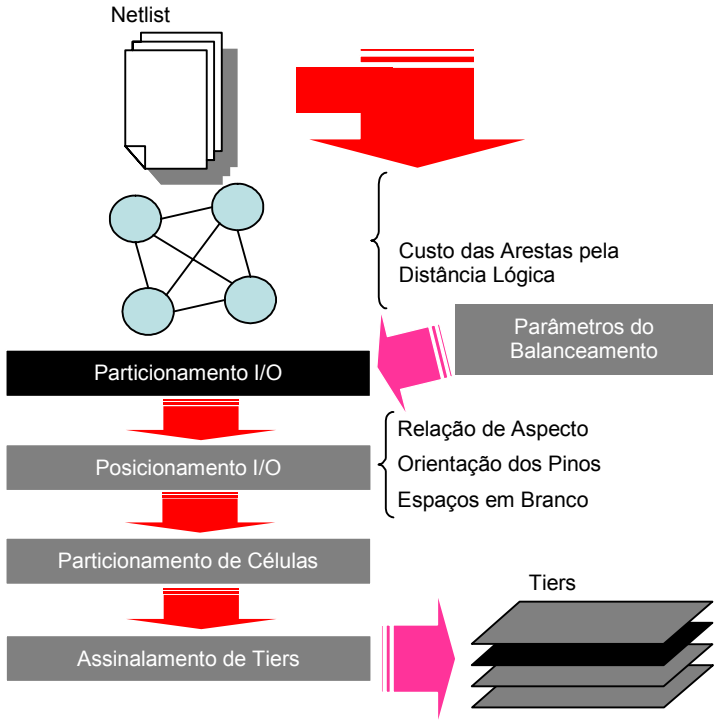


Figura 5.14: Alteração do fluxo para a inserção de parâmetros para balanceamento.

A Figura 5.14 mostra uma nova abordagem do algoritmo proposto na Seção 5.4. Porém, existem variações no balanceamento dos pinos de I/O entre as *tiers* (SAWICKI, et al., 2006a). Em trabalhos anteriores (SAWICKI, et al., 2006), foi imposto um controle de balanceamento rígido com o objetivo de alcançar-se um equilíbrio

comparável à abordagem do método *alternate pins*. Contudo, nesta seção, o foco principal é reduzir o número de vias-3D com um maior desequilíbrio do número de pinos de I/O. Nesse trabalho, investiga-se o impacto do balanceamento dos pinos de I/O no restante do fluxo.

Quando a etapa de particionamento começa, os pinos já estão fixados. Por isso, a área de cada partição é restrita pela área das células e pelos espaços em branco. Nesta nova abordagem, o parâmetro usado para especificar o desbalanceamento das partições durante as bipartições é um número inteiro entre 1 e 49. Por exemplo, considerando-se um hipergrafo com n vértices, cada um contendo um peso, dado u como o parâmetro do desbalanço, se o número de partições consideradas fosse dois, o número de vértices assinalados para cada uma deveria estar compreendido pelo intervalo determinado pela equação 15:

$$\left[\frac{(50-u)\times n}{100};\frac{(50+u)\times n}{100}\right] \qquad (15)$$

Dado u = 10, o balanço da bisseção será de 40 a 60%. Porém, supondo-se que o número de partições seja quatro e o fator de desbalanceamento seja 10, as partições poderão conter entre $0.40^2n = 0.15n$ e $0.60^2n = 0.35n$ vértices.

5.5.1 Resultados Experimentais

O conjunto de *benchmarks* utilizado para os experimentos foi o ISPD IBM 2004 (ISPD04, 2008). Os resultados dessa abordagem foram comparados com dados obtidos em trabalhos anteriores (SAWICKI et al., 2006), expostos na Tabela 5.5 e

Tabela 5.6, descritas na Seção 5.4.3. Nesse trabalho, o critério de balanceamento foi rígido, com u = 1. Contudo, para os experimentos atuais de desbalanceamento utilizou-se u = 10 e u = 25.

Os resultados comparativos são apresentados na Tabela 5.10, Tabela 5.11 e Tabela 5.12. Nelas são descritos os efeitos do balanceamento do número de pinos de I/O.

Tabela 5.10: Tabela do número de vias com balanceamento rígido

	Algoritmo LHu-3D u=1			
Tiers	**2**	**3**	**4**	**5**
Ibm01	374	525	837	1.162
ibm02	396	747	1.156	1.533
ibm03	1.064	2.174	2.610	3.974
ibm04	735	1.511	2.371	2.852
ibm05	2.258	4.311	6.489	9.193
ibm06	1.059	1.642	2.934	3.477
ibm07	992	2.050	3.219	4.400
ibm08	1.298	2.697	4.018	5.346
ibm09	699	1.872	2.495	3.343
ibm10	1.490	2.661	4.004	5.216
ibm11	1.190	2.240	3.685	4.620
ibm12	2.293	4.094	6.581	8.191

ibm13	1.042	1.893	3.099	3.742
ibm14	2.121	3.886	5.342	6.667
ibm15	2.890	4.827	7.022	9.283
ibm16	2.102	5.476	5.918	8.920
Média	1.375	2.663	3.861	5.120

Tabela 5.11: Número de vias com balanceamento *u=10*

	Algoritmo LHu-3D u=10			
Tiers	**2**	**3**	**4**	**5**
ibm01	364	504	882	1.110
ibm02	389	730	1.128	1.485
ibm03	1.036	2.133	2.514	3.895
ibm04	711	1.499	2.333	2.790
ibm05	2.260	4.273	6.536	8.890
ibm06	1.048	1.628	2.951	3.450
ibm07	952	2.091	3.257	4.356
ibm08	1.272	2.686	3.993	5.372
ibm09	724	1.773	2.625	3.333
ibm10	1.527	2.799	3.937	5.215
ibm11	1.232	2.280	3.632	4.765
ibm12	2.248	4.067	6.446	8.111
ibm13	1.086	1.822	3.170	3.699
ibm14	2.078	3.828	5.201	6.661
ibm15	2.854	4.819	7.015	9.158
ibm16	2.095	4.376	5.928	7.219
Média	1.367	2.582	3.847	4.969

Tabela 5.12: Número de vias com balanceamento *u = 25*

	Algoritmo LHu-3D u=25			
Tiers	**2**	**3**	**4**	**5**
Ibm01	325	471	773	1.098
ibm02	407	427	1.099	1.407
ibm03	994	2.078	2.538	3.854
ibm04	664	1.465	2.252	2.617
ibm05	2.258	4.327	6.530	8.878
ibm06	1.047	1.595	2.926	3.435
ibm07	942	2.007	3.122	4.546
ibm08	1.231	2.679	3.956	5.282
ibm09	684	1.763	2.509	3.256
ibm10	1.522	2.875	3.843	5.211
ibm11	1.194	2.249	3.688	4.760
ibm12	2.182	4.053	6.257	8.110
ibm13	1.035	1.773	3.126	3.694
ibm14	2.000	3.658	5.200	6.456
ibm15	2.856	4.859	7.026	9.101

ibm16	1.996	4.296	5.659	7.126
Média	1.334	2.536	3.782	4.927

A Figura 5.15 ilustra, por meio de um gráfico, os resultados descritos nas tabelas 5.10, 5.11 e 5.12 (aumento do desvio padrão entre as *tiers*). O gráfico relaciona a variação do desbalanceamento (*u=1, u=10 e u=25)* e a melhoria na redução do número de vias-3D. Parte-se do balanceamento rígido e conforme aumenta o desbalanceamento o número de vias também é reduzido. Contudo, percebe-se que a redução é pequena. Para *u=10* a maior redução ocorre em um projeto com cinco tiers (1%), já com *u=25*, o melhor resultado é de 2,70%, com duas *tiers*.

A redução do número de vias em relação com o número de *tiers* não segue um padrão se relacionado ao valor do desbalanceamento dos pinos. O algoritmo promove melhora no corte, contudo não reflete da mesma forma a melhora no número de vias. Pode-se perceber no gráfico da Figura 5.15 que a curva que representa o ganho no número de vias no caso de duas *tiers* é idêntica a melhoria do corte. No entanto, para os outros casos a relação varia para mais ou para menos. Quanto maior o número de *tiers* maior é a variação. Logo, percebe-se nessa figura que as curvas que descrevem mais *tiers* apresentam uma maior variação.

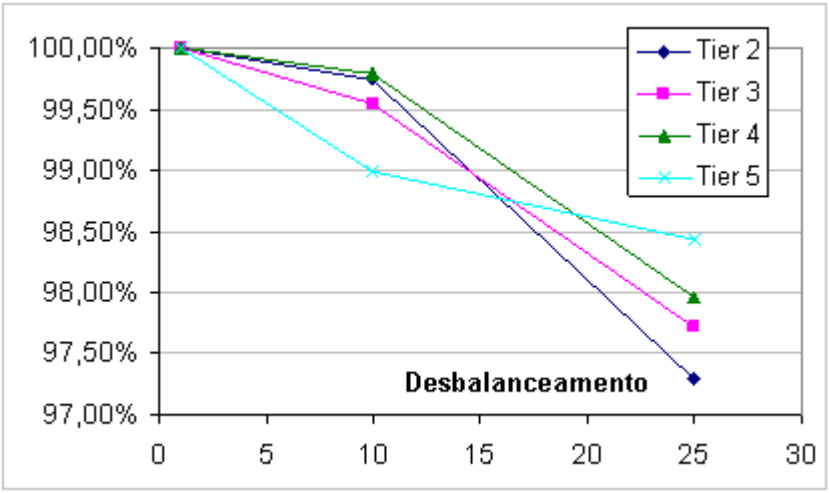


Figura 5.15: Desbalanceamento de pinos de I/O.

A Figura 5.16 apresenta a média do desvio padrão em relação à variação do desbalanceamento e ao número de *tiers.* A Tabela 5.13 apresenta o crescimento do desvio padrão para cada variação do parâmetro *u*. O desvio padrão é somente uma métrica para representar a variação do número de pinos entre as diferentes *tiers*. No entanto, o valor calculado não possui significância estatística devido ao reduzido tamanho da amostra. Isso significa que nada pode ser inferido a respeito do número de pinos em um dado tier com base nesse valor.

Tabela 5.13: Aumento do desvio padrão dos pinos de I/O entre *tiers*

	tier 2	***tier 3***	***tier 4***	***tier 5***
U=1	7	6	5	6
U=10	64	54	48	41

U=25	158	141	103	100

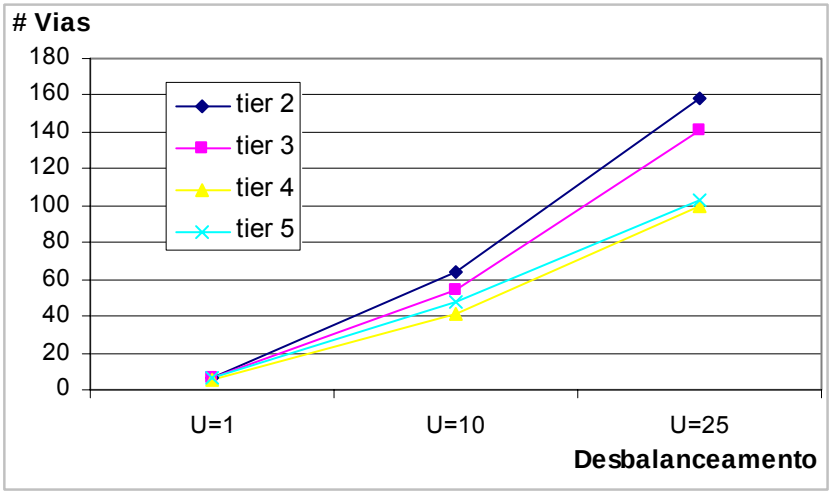


Figura 5.16: Desvio padrão dos pinos de I/O entre partições.

5.5.2 Considerações Finais sobre a Variação do Algoritmo

A Seção 5.4 mostrou a evolução do algoritmo de particionamento de pinos de I/O em circuitos 3D por meio do algoritmo LHu-3D. Foi apresentado um método automático para a migração de uma *netlist* 2D para uma 3D. Nesta, os pinos são particionados e posicionados na borda do circuito. Em trabalhos anteriores, estudou-se o impacto de um método de particionamento eficaz de pinos que consiga reduzir o número de vias-3D em comparação com outras abordagens.

O foco deste trabalho foi investigar alternativas para aumentar a qualidade do corte, em outras palavras, diminuir ainda mais o número de vias-3D. Para tal experimento, o impacto do desbalanceamento de pinos foi estudado na tentativa de descobrir como essa característica poderia ajudar no processo de particionamento subsequente de células.

5.6 Algoritmo para Redução de Vias-3D Longas (LHr-3D)

Este tópico propõe uma heurística iterativa para tratar o problema do surgimento das conexões que cruzam mais de duas *tiers* adjacentes (vias-3D longas). O algoritmo LHr-3D tem como objetivo contribuir para a redução do número total de vias-3D mediante o aumento das conexões curtas. O algoritmo desenvolvido foi chamado de LHr-3D (SAWICKI, et. al., 2009).

Apesar da qualidade dos resultados oferecidos pelos algoritmos de particionamento (KARIPYS, 1997), (FIDUCCIA-MATHEYSES, 1982), (KERNIGHAN-LIN, 1970), sua aplicação em circuitos VLSI 3D (ABABEI; FENG; GOPLEN; MOGAL; ZHANG; BAZARGAN; SAPATNEKAR, 2005) não atua na identificação das conexões que cruzam mais de duas *tiers* adjacentes. Isso ocorre, pois sua estrutura não foi desenvolvida para resolver problemas em que as partições estejam organizadas em linha. Em geral, a atuação desses algoritmos resulta em um grafo completo, no qual o peso das arestas indica o número de redes que conectam as diferentes partições. Assim, no momento de alinhar as partições (arranjo 1D), surgem as conexões longas.

O algoritmo desenvolvido é inspirado em *simulated annealing* (KIRKPATRICK, 1983). Contudo, em vez de aceitar as soluções piores para evitar mínimos locais, essa heurística utiliza-se de uma boa solução inicial (suficientemente próxima da ótima) utilizando nosso trabalho anterior, apresentado na Seção 5.4.

A principal diferença entre a nova abordagem e os particionadores de hipergrafos é que ela considera a localização das partições. Na verdade, em um circuito 3D, as partições são organizadas em linha, o que implica o conhecimento das partições adjacentes (baixo custo em termos de corte) e das distantes (custo elevado, pois demandam vias-3D extras).

Objetivando minimizar as vias-3D como um todo, pretende-se reduzir as conexões que cruzam mais de duas *tiers* adjacentes, que não são consideradas por particionadores de hipergrafos. Por exemplo, o algoritmo apresentado na Seção 5.4 utiliza particionamento de hipergrafos para dividir as células em grupos e, em um segundo estágio, executa um pós-particionamento para distribuir as partições em um espaço 1D (linha), de modo que o número total de vias-3D seja minimizado – como ilustrado na Figura 5.17 (a). Embora essa abordagem tenha o mesmo objetivo, é clara a sua limitação, pois os grupos não podem ser quebrados. O algoritmo proposto neste item é a fusão dos dois passos referidos, como ilustrado na Figura 5.17 (b). Essa heurística resolverá principalmente tal limitação.

A abordagem iterativa proposta permite aos projetistas escolher exatamente a função de custo que se adapta ao seu projeto, além da distribuição de pinos e células e da otimização da área.

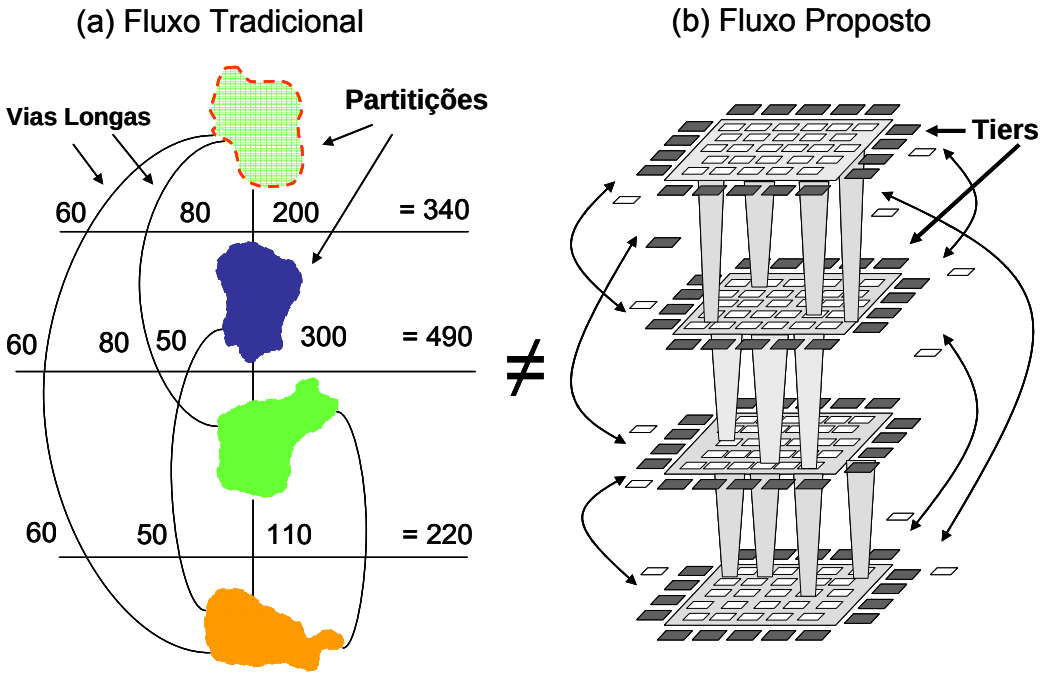


Figura 5.17: Posição das *tiers* e partições.

5.6.1 Formulação do Problema

Considere um circuito de lógica aleatória e o *floorplanning* de um circuito 3D (incluindo área e número de *tiers*). Execute o particionamento das células e dos pinos de I/O nas *tiers*, tal que a quantidade de vias-3D seja minimizada enquanto reduz o número de redes verticais longas e a área do circuito, equilibrando o número de pinos de I/O distribuídos entre as *tiers*.

5.6.2 Heurística de Particionamento

O algoritmo proposto escolhe a solução inicial apresentada na Seção 5.4 e a melhora iterativamente utilizando perturbações aleatórias existentes na solução. As perturbações podem ser aceitas ou rejeitadas dependendo da variação do custo. Qualquer perturbação que melhore o estado corrente é aceita e todas as perturbações que pioram o custo são rejeitadas.

5.6.2.1 Procedimento de Perturbação

A função de perturbação projetada move as células e pinos entre as partições. Embora seja de natureza aleatória, é executada de duas diferentes formas: movimentação *simples* ou *dupla troca*. A perturbação simples e dupla são alternadas (tendo 50% de chances cada de ser executada) e trabalham da seguinte forma:

- A perturbação **simples** pode mover uma célula ou um pino de I/O (com 50% de chances probabilísticas cada) para uma *tier* diferente (escolhida também aleatoriamente).
- A perturbação **dupla** seleciona aleatoriamente um par de elementos (localizados em partições diferentes). Cada elemento pode ser tanto uma célula quanto um pino (tendo cada elemento 50% de chances de ser selecionado), totalizando quatro diferentes perturbações duplas, cada uma tendo 25% de probabilidade de ocorrer.

Função de Perturbação

Escolha o tipo de perturbação (troca dupla, troca simples)
Se (**troca simples**)
Escolha uma **célula** ou um **pino**
Se a escolha for um **pino**
Escolher pino e tier origem
Escolher *tier* destino, tal que, *tier destino* seja diferente de *tier origem*
Efetuar troca do pino
Se a escolha for uma **célula**
Escolher célula e tier origem
Escolher *tier* destino, tal que, *tier destino* seja diferente de *tier origem*
Efetuar troca da célula
Se (**troca dupla**)
Escolher (**célula** ou **pino**) da tier origem
Escolher (**célula** ou **pino**) da tier destino, tal que, *tier destino* seja diferente de *Tier origem*
Se a escolha for (pino, pino) efetuar troca
Se a escolha for (pino, célula) efetuar troca

```
        Se a escolha for (célula, célula) efetuar troca
        Se a escolha for (célula, pino) efetuar troca
Fim
```

Algoritmo 5.2: Pseudocódigo da função de Perturbação

5.6.2.2 Função de Custo

Qualquer estado intermediário do processo de particionamento pode ter sua qualidade medida por uma função de custo. Nesta são modeladas todas as métricas de interesse em um único número, que representa esse custo.

Essa função é dividida em três partes distintas: um custo ***v***, associado aos recursos utilizados pelas vias-3D; um valor ***a*** para o balanceamento da área; um custo ***p*** para o balanceamento de pinos de I/O. O custo relatado é uma combinação das três partes. Com o intuito de utilizá-las em conjunto, a equação foi normalizada por meio da divisão de cada número pelo seu valor inicial v_i, a_i e p_i (computada antes da primeira perturbação). Além disso, foram impostos pesos (w_v, w_a e w_p) a fim de ajustar-se a função de custo para a otimização de vias, como mostra a equação 1.

Os valores de *v*, *a* e *p* são computados da seguinte forma:

- Para cada rede, calcula-se o quadrado do número de vias; adiciona-se o número computado para cada rede para que seja obtido em ***v***. Aplica-se o quadrado do valor para punir as redes que tenham conexões longas e incentivar o aumento de redes curtas.
- Para computar *a*, calcula-se primeiro a área de células de todas as *tiers*; o custo do desbalanceamento é a subtração da maior área pela menor área.
- O valor de *p* é calculado da mesma maneira que o valor de *a*.

$$c = \frac{(w_v \times v)}{v_1} + \frac{(w_a \times a)}{a_1} + \frac{(w_p \times p)}{p_1} \qquad (16)$$

5.6.2.3 Função de Schedule

A função de s*chedule* computa o valor da temperatura inicial e sua variação ao longo do algoritmo. Determina também o comportamento da aceitação de *simulated annealing*. Por exemplo, na abordagem clássica desse algoritmo, começando-se com temperaturas altas, a variação da temperatura é de fundamental importância, pois ela determinará por quanto tempo ele será aleatório e por quanto será guloso. Decidirá, também, o impacto dessa mudança.

Nessa etapa do trabalho, a *netlist* inicial já foi convertida para *netlist* 3D e otimizada pelo algoritmo descrito na Seção 5.4. Sendo assim, o refinamento parte de uma solução inicial muito boa. Por esse motivo, o algoritmo *simulated annealing* é definido com temperatura constante em zero. Em outras palavras, ele não aceita movimentos que piorem seu estado atual segundo a definição da função de custo.

Função de *Schedule*

Para cada iteração de *n* repetições
Ao final de cada iteração
Calcular a diferença de custo com a iteração anterior
Δ Custo = **Custo** (Nova Solução) – **Custo** (Solução);
Se abs(Δ Custo) < 0,7%, considera-se que não houve alteração
Se houver *x* iterações sem alteração, fim.

Algoritmo 5.3: Pseudocódigo da função de *Schedule*

Para executar o algoritmo de refinamento, são necessários cinco requisitos:

(a) *Solução inicial*: pode ser aleatória ou uma boa solução inicial. Nessa etapa, trabalha-se com o refinamento da *netlist.* Assim, a solução inicial é a encontrada pelo algoritmo descrito na Seção 5.4, já que o *simulated annealing* é usado iterativamente e, portanto, sempre melhora a solução inicial. Experimentos adicionais dessa tese mostraram que soluções iniciais aleatórias nesta etapa do algoritmo são muito inferiores a solução adotada, sendo completamente descartadas.

(b) *Tamanho do problema*: o tamanho do problema é calculado pela soma do número total de células da *netlist* com o número total de pinos de I/O. Está relacionado com o número de iterações do algoritmo.

(c) *Função de custo*: citada anteriormente, a função de custo considera três pontos: (1) penalização das conexões longas; (2) área das *tiers*, representada pela área das células contidas em cada *tier*; (3) número de pinos de I/O entre as *tiers*.

(d) *Perturbação*: a função de perturbação deve ser (1) simples, para fazer uma perturbação de granularidade pequena, a fim de que, por ela, atinja-se qualquer estado durante o particionamento; (2) aleatória, para que todas as células e pinos atinjam o espaço de soluções pesquisado.

(e) *Função desfazer perturbação (função* undo*)*: Toda a função de perturbação deve ser revertida caso não seja aceita pela função de avaliação. Para que essa reversão seja realizada de forma eficiente, deve ser implementada uma função de *undo* para cada tipo de perturbação desenvolvida. A etapa de refinamento descrita nesta etapa trabalha com três funções de perturbação: combinações de trocas entre pinos e células, células e células e pinos e pinos, além de trocas individuais de pinos e células.

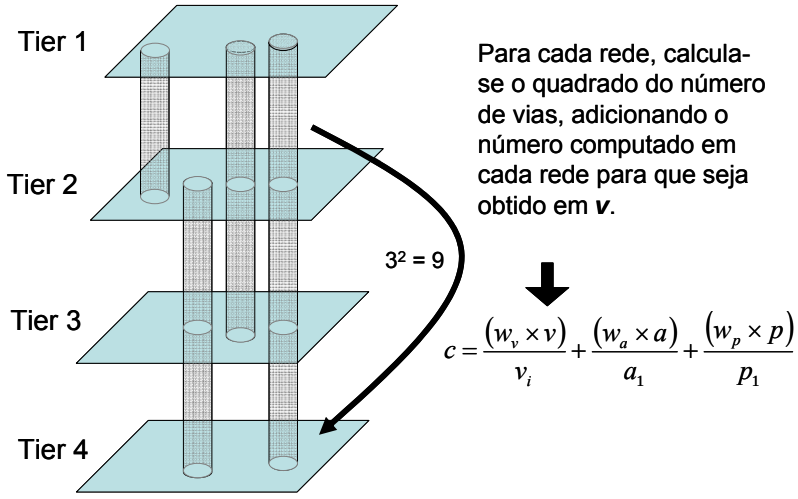


Figura 5.18: Função que penaliza conexões verticais longas.

Algoritmo proposto para redução de conexões longas (LHr-3D)

Passo 1	Computar *netlist* mantendo orientação das *tiers*
Passo 2	Calcular número inicial de: • vias-3D • desbalanceamento da área de células; • pinos de I/O
Passo 3	Executar **procedimento de perturbação**
Passo 4	Efetuar a troca: *{(pino, pino) | (pino, célula) | (célula, pino) | (célula, célula)}*
Passo 5	Calcular o custo da troca: Δ Custo = **Custo** (Nova Solução) – **Custo** (Solução); (16)
Passo 6	Se (Δ Custo < 0) Aceita Troca (); Passo 3 Senão Rejeita Troca (); Desfaz Troca (); Passo 3
Passo 7	Se o custo não mudou (0,7%) em *n* iterações seguidas então fim. Senão Passo 3

Algoritmo 5.4: Pseudocódigo do algoritmo LHr-3D

5.6.3 Resultados Experimentais

Nesta seção, o algoritmo de particionamento proposto, LHr-3D, foi comparado com o particionador de hipergrafos hMetis e com o algoritmo LHp-3D. Estes algoritmos, por sua vez, têm a liberdade de particionar a *netlist* (incluindo células e pinos de I/O) para *n* partições, onde *n* é o número de *tiers*. No estágio subsequente, as partições são assinaladas para as *tiers*, como ilustrado na Figura 5.17a, conforme o método

apresentado por Ababei (ABABEI, 2005). O algoritmo hMetis particiona livremente pinos de I/O e células. No trabalho que desenvolvemos anteriormente, apresentado na Seção 5.4, utilizou-se uma abordagem diferente. Primeiro, os pinos de I/O de um bloco foram divididos e fixados nas diferentes partições. Um sistema de controle foi projetado para garantir um bom balanceamento, e uma heurística foi executada a fim de auxiliar na redução do corte. No subitem 5.4.3, foi demonstrado que esse método é capaz de reduzir o número de vias em 10% (duas *tiers*), 10% (três *tiers*), 6% (quatro *tiers*) e 11% (cinco *tiers*), em comparação com hMetis (*unlocked pins*). Esse método foi chamado de LHp-3D. Note-se que o método LHr-3D, proposto neste subcapítulo, inicia com a solução obtida pelo LHp-3D e executa a nova heurística para melhorar a qualidade de seus resultados.

Para os experimentos, foram utilizados circuitos *benchmarks* ISPD 2004, e o projeto foi desenvolvido em duas, três, quatro e cinco *tiers*. Os três métodos referidos (hMetis, LHp-3D e LHr-3D) foram comparados entre si. Em todos os casos, a distribuição da área foi rígida, sendo 0,1% o pior resultado de desbalanceamento de área encontrado. O balanceamento de pinos de I/O não foi imposto para o algoritmo hMetis, pois ele não contém essa restrição. Por essa razão, esse algoritmo teve o pior resultado de desbalanceamento de pinos. Já o método LHr-3D dá pouca liberdade para os pinos de I/O, a fim de diminuir o número de vias-3D.

A Figura 5.19 mostra o número total de vias-3D de todos os três métodos. O LHr-3D obteve os melhores resultados. As melhorias estão, em comparação com hMetis e LHp-3D, respectivamente, na ordem de 19 e 11% para duas *tiers*, 18 e 8% para três *tiers*, 12 e 6% para quatro *tiers* e, finalmente, 16 e 6% para cinco *tiers*.

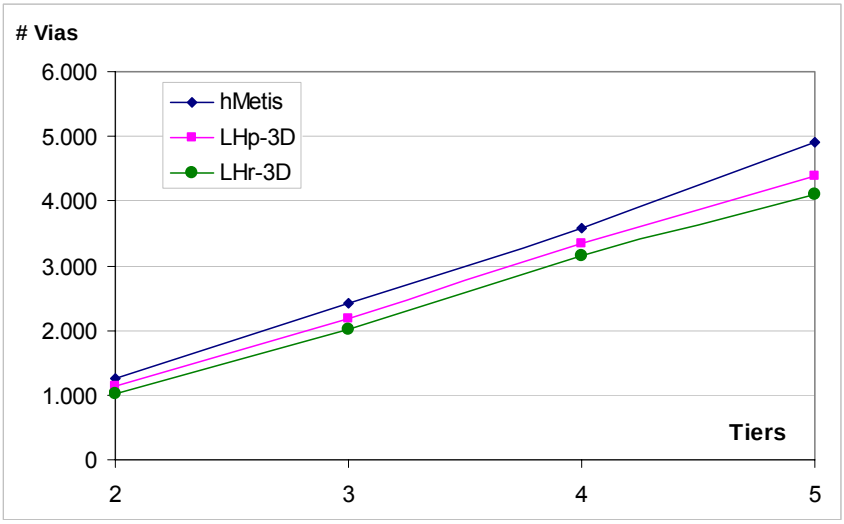


Figura 5.19: Número total de vias-3D após refinamento.

Os valores absolutos da Figura 5.19 estão descritos na

Tabela 5.14. Nesta percebe-se que todos os *benchmarks* usados na comparação convergiram para a redução total do número de vias-3D quando foi utilizado o LHr-3D, em oposição ao que ocorreu com os algoritmos hMetis e LHp-3D, tendo este obtido resultados melhores em comparação com o método hMetis.

Tabela 5.14: Número total de vias-3D

	hMetis				LHp-3D				LHr-3D			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	441	735	838	1.439	374	525	837	1.162	274	430	737	952
ibm02	547	882	1.214	1.600	396	747	1.156	1.533	376	583	1.055	1.229
ibm03	1.146	2.257	2.693	4.020	1.064	2.174	2.610	3.974	963	2.040	2.530	3.804
ibm04	738	1.583	2.516	3.202	735	1.511	2.371	2.852	614	1.348	2.140	2.514
ibm05	2.417	4.651	6.653	9.651	2.258	4.311	6.489	9.193	1.931	3.905	6.015	7.936
ibm06	1.061	1.827	3.128	3.566	1.059	1.642	2.934	3.477	986	1.539	2.876	3.443
ibm07	994	2.038	3.302	4.605	992	2.050	3.219	4.400	875	1.937	3.216	4.310
ibm08	1.324	2.814	4.184	5.698	1.298	2.697	4.018	5.346	1.161	2.655	4.088	5.276
ibm09	806	1.904	2.763	3.518	699	1.872	2.495	3.343	652	1.726	2.385	3.167
ibm10	1.771	3.555	4.675	7.116	1.490	2.661	4.004	5.216	1.340	2.577	3.467	5.263
ibm11	1.490	2.581	3.958	5.697	1.190	2.240	3.685	4.620	1.085	2.131	3.476	4.349
ibm12	2.594	4.470	7.259	9.187	2.293	4.094	6.581	8.191	2.114	3.708	6.129	7.911
ibm13	1.193	2.298	3.264	4.557	1.042	1.893	3.099	3.742	866	1.482	2.799	3.256
ibm14	2.171	4.561	6.584	8.085	2.121	3.886	5.342	6.667	2.017	3.529	5.140	6.429
ibm15	3.002	7.863	9.082	11.707	2.890	4.827	7.022	9.283	2.877	4.670	6.916	9.062
ibm16	2.237	5.816	6.235	9.300	2.102	5.476	5.918	8.920	2.061	5.386	5.838	8.811
Média	1.496	3.111	4.272	5.809	1.375	2.663	3.861	5.120	1.262	2.478	3.675	4.857

A Figura 5.20 mostra o número máximo de vias-3D entre um par de *tiers* adjacentes. Para se compreender melhor essa figura, imagine que, entre cada um desses pares, existe uma via. Quanto menor for a área dessa via, menor será o espaço ocupado. O mesmo vale para o número máximo de vias que cruzam um par de *tiers*: quanto maior o número de conexões, maior será a área das camadas. Os resultados obtidos pelo método proposto neste subcapítulo reduzem, em todos os casos, esse número de forma similar à utilizada para a redução do número total de vias.

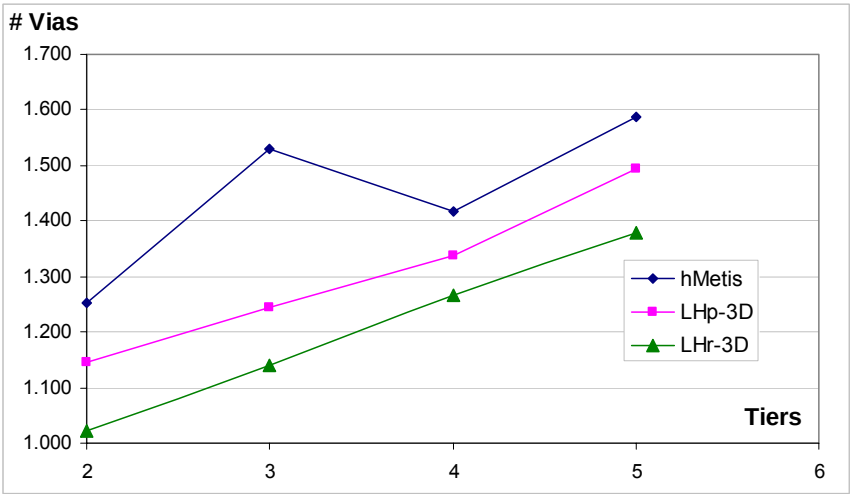


Figura 5.20: Número máximo de vias-3D após refinamento.

Os números absolutos da redução da quantidade máxima de vias-3D acompanham a diminuição do número total dessas vias, como mostra a Tabela 5.15. Todos os *benchmarks* contribuem para essa redução. Nesse caso, a redução obtida pelo LHr-3D foi de 19%, 26%, 11% e 13% para duas, três, quatro e cinco *tiers*, respectivamente, em comparação com o algoritmo hMetis, e 11%, 8%, 5% e 7% em comparação com o algoritmo LHp-3D.

Tabela 5.15: Número máximo de vias-3D entre duas *tiers* adjacentes

	hMetis				**LHp-3D**				**LHr-3D**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	441	467	377	573	374	305	322	406	274	293	284	341
ibm02	547	496	485	552	396	413	403	594	376	318	384	492
ibm03	1.146	1.143	1.021	1.334	1.059	1.187	1.086	1.225	963	1.103	1.018	1.175
ibm04	738	862	1.067	1.039	735	887	992	887	614	815	942	844
ibm05	2.417	2.765	2.478	2.712	2.258	2.203	2.469	2.729	1.931	1.995	2.245	2.225
ibm06	1.061	924	1.134	935	1.059	849	1.135	948	986	779	1.094	922
ibm07	994	1.980	1.510	1.525	992	1.332	1.433	1.524	907	1.239	1.462	1.473
ibm08	1.324	1.436	1.445	1.788	1.298	1.448	1.397	1.610	1.161	1.364	1.502	1.528
ibm09	806	1.598	1.092	1.249	699	1.057	1.008	1.075	652	975	954	957
ibm10	1.771	1.883	1.741	1.986	1.490	1.450	1.590	1.750	1.340	1.377	1.374	1.634
ibm11	1.490	1.909	1.810	2.230	1.190	1.485	1.605	1.719	1.085	1.460	1.467	1.615
ibm12	2.594	2.820	2.747	2.962	2.293	2.278	2.422	3.173	2.114	2.062	2.285	3.096
ibm13	1.193	1.606	1.500	1.755	1.042	1.269	1.548	1.781	866	1.033	1.461	1.640
ibm14	2.171	2.375	2.307	2.881	2.121	2.272	2.248	2.459	2.017	2.034	2.103	2.245
ibm15	3.002	4.188	3.377	4.099	2.890	2.857	3.199	3.395	2.877	2.715	2.939	3.125
ibm16	2.237	3.185	2.266	3.355	2.102	2.164	2.212	2.625	2.061	1.987	2.033	2.412
Média	1.496	1.852	1.647	1.936	1.375	1.466	1.567	1.744	1.264	1.347	1.472	1.608

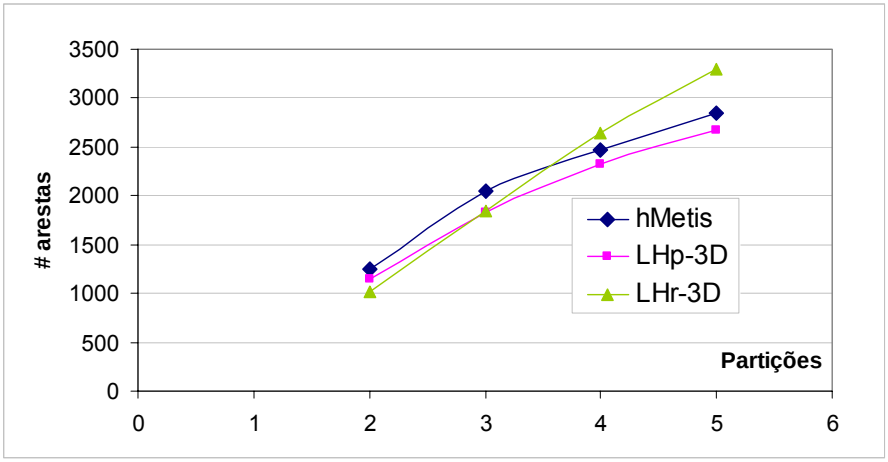


Figura 5.21: Qualidade do corte.

A Figura 5.21 apresenta o particionamento final por meio de um hipergrafo. Compara-se o LHr-3D com outros dois particionadores, hMetis e LHp-3D. O eixo y mostra a média do número de arestas entre as diferentes partições. Pode-se observar que o algoritmo proposto aumenta o número de arestas no momento em que o número de

partições cresce. Entretanto, quando somente duas partições são criadas, o mesmo algoritmo obtém o melhor corte em comparação com as outras estratégias. Esse comportamento é explicado pela função de custo que foi usada para a otimização, pois, ao contrário de hMetis e LHp-3D, o LHr-3D não reduz o corte entre as partições, mas sim o número total de vias. Quando somente duas partições são consideradas, o número de vias é obtido pelo valor do corte do algoritmo de particionamento. Por outro lado, quando mais partições são criadas, o algoritmo aumenta o número de conexões entre partições adjacentes a fim de reduzir o número de conexões entre *tiers* não-adjacentes (como mencionado anteriormente). Tal comportamento leva a um aumento do corte entre diferentes partições ao mesmo tempo em que reduz o número total de vias-3D.

A Figura 5.22 apresenta o mesmo particionamento que a Figura 5.21, mas sob a perspectiva do número de vias-3D. As faixas do gráfico representam o número total de vias obtidas por cada algoritmo, ao longo de cinco *tiers*. Cada faixa é dividida em quatro partes. A identificada pelo número 1 representa o número de vias-3D que conecta *tiers* adjacentes, enquanto as outras três, identificadas pelos números 2, 3 e 4, representam o número de vias que conecta as não-adjacentes. Estes três últimos blocos informam a quantidade de vias-3D necessárias para conectar diretamente duas *tiers*. Nesse caso, o uso de duas vias indica que a conexão precisa atravessar uma camada. O bloco identificado com o número 3 indica que é necessário atravessar duas delas e o identificado pelo número 4 sinaliza a necessidade de atravessar três *tiers*.

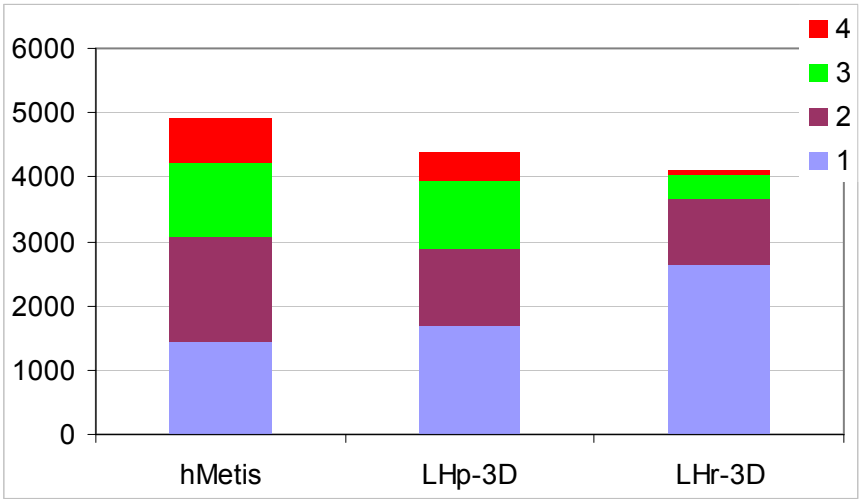


Figura 5.22: Distribuição das vias-3D em um projeto de cinco *tiers*.

Tabela 5.16: Distribuição das vias-3D em um projeto de 5 *tiers*.

Benchs	**hMetis**				**LHp-3D**				**LHr-3D**			
	1	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
ibm01	416	482	441	100	532	454	156	20	640	246	66	0
ibm02	796	423	273	108	776	346	334	77	851	162	216	0
ibm03	1.008	1.527	954	531	1.062	1.446	1.063	403	1.922	1.124	714	44
ibm04	1.419	1.330	381	72	1.598	944	270	40	2.052	406	48	8
ibm05	1.465	2.426	2.928	2.832	1.792	2.798	2.523	2.080	4.537	2.764	519	116
ibm06	925	814	1.299	528	927	782	1.332	436	1.634	918	831	60

ibm07	1.907	1.443	921	334	2.585	923	599	293	3.307	958	45	0
ibm08	1.393	2.556	549	1.200	2.456	898	1.320	672	3.063	1.316	669	228
ibm09	1.535	1.110	561	312	1.470	1.066	555	252	2.425	414	312	16
ibm10	1.858	2.382	1.932	944	1.951	1.550	1.467	248	4.170	938	87	68
ibm11	1.706	2.398	945	648	2.094	1.500	606	420	2.953	1.060	252	84
ibm12	3.270	2.416	2.817	684	2.874	2.050	2.571	696	4.503	2.230	1.014	164
ibm13	1.034	1.859	1.200	464	1.622	1.284	732	104	2.370	688	114	84
Média	1.441	1.628	1.169	674	1.672	1.234	1.041	442	2.648	1.017	376	67

A Tabela 5.16 descreve claramente a redução das vias-3D não-adjacentes obtida pelo algoritmo LHr-3D. Cada algoritmo está dividido em quatro colunas, indicadas pelos números 1, 2, 3 e 4, que apresentam os resultados de um projeto com cinco *tiers.* Cada um desses números representa a quantidade de vias necessárias para conectar duas camadas. Por exemplo, analisando os dados dos algoritmos hMetis e LHr-3D do *benchmark* ibm01, verifica-se que o primeiro precisa de 416 vias-3D para conectar duas *tiers* adjacentes, portanto não há necessidade de perfuração das camadas. Já para o algoritmo LHr-3D, são necessárias 640 vias-3D. Tal comportamento é esperado, pois sua função de custo faz com que as vias-3D longas migrem para *tiers* adjacentes. Na coluna 3 do algoritmo hMetis, percebe-se que, para conectar duas *tiers*, são necessárias 441 vias-3D. Isso significa que 147 destas cruzam duas *tiers* para conectar as camadas origem e destino. Cria-se assim, para cada uma dessas conexões, três novas vias (147 x 3 = 441). Já o algoritmo LHr-3D necessita de 66 vias-3D para conectar as mesmas *tiers*. Assim, tem-se 22 vias cruzando duas *tiers*, o que cria um total de 66 vias-3D (22 x 3 = 66).

A coluna 4 do algoritmo hMetis indica que são necessárias 100 vias-3D para conectar as *tiers* origem e destino. Assim, para cada conexão entre estas, criam-se quatro novas vias (25 x 4 = 100). Por outro lado, o algoritmo LHr-3D elimina todas as vias entre as *tiers* origem e destino (0 x 4 = 0). Esse comportamento se repete entre os demais *benchmarks* como pode ser verificado nos gráficos abaixo.

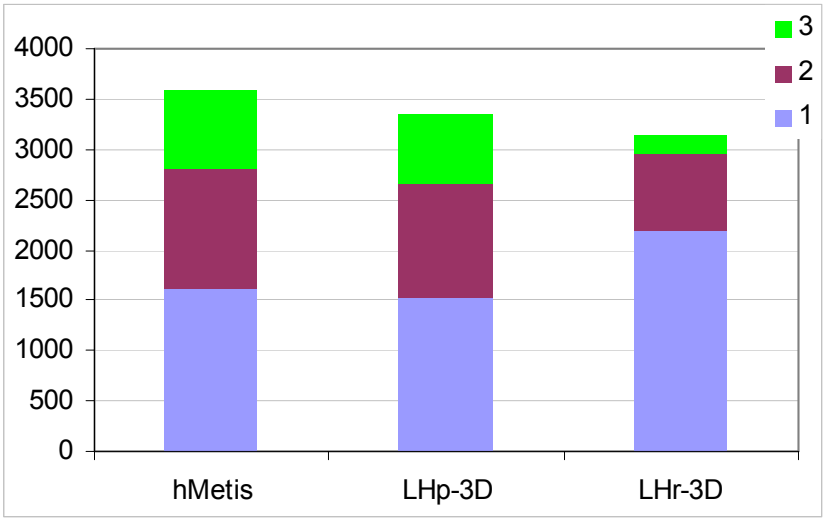


Figura 5.23: Distribuição das vias-3D em um projeto de 4 *tiers*.

A Figura 5.23 apresenta o gráfico de um projeto com quatro *tiers*. Percebe-se a redução do número total de vias-3D do algoritmo LHr-3D em comparação com os algoritmos hMetis (12%) e LHp-3D (5%). O comportamento do algoritmo proposto neste trabalho continua penalizando as conexões longas, como mostram as faixas 1, 2 e 3. O maior número de vias-3D se concentra nas *tiers* adjacentes.

A Tabela 5.17 mostra o número de vias-3D necessárias para um projeto com quatro *tiers*, usando-se os algoritmos hMetis, LHp-3D e LHr-3D. O comportamento dos dados dessa tabela pode ser visto na Figura 5.23, acima. O algoritmo LHr-3D, em comparação com hMetis, obteve redução de 75% das vias que cruzam duas *tiers* e redução de 72% em relação ao LHp-3D. Para vias que cruzam uma *tier*, a redução foi de 37%, em comparação com o algoritmo hMetis, e 34% em oposição ao LHp-3D. Contudo, para vias que não necessitam cruzar camadas, o aumento foi de 37% em comparação com o algoritmo hMetis e 44% em comparação com o LHp-3D. Esses resultados demonstram a maior concentração de vias-3D curtas nos resultados do algoritmo LHr-3D e a convergência para a redução do número total dessas vias.

Tabela 5.17: Distribuição das vias-3D em um projeto de 4 *tiers*.

Benchs	**hMetis**			**LHp-3D**			**LHr-3D**		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
ibm01	404	236	198	402	246	189	521	192	24
ibm02	336	410	468	575	434	147	716	258	81
ibm03	1.218	1.016	459	1.114	1.208	288	1.610	728	192
ibm04	1.589	666	261	1.524	670	177	1.799	296	45
ibm05	2.072	2.814	1.767	2.087	2.746	1.656	4.365	1.536	114
ibm06	882	1.442	804	702	1.200	1.032	1.214	1.152	510
ibm07	1.974	800	528	1.748	850	621	2.483	658	75
ibm08	1.733	1.128	1.323	1.682	1.160	1.176	2.373	1.226	489
ibm09	1.452	762	549	1.550	606	339	1.741	500	144
ibm10	2.135	1.190	1.350	1.995	1.256	753	2.683	628	156
ibm11	2.117	1.154	687	1.845	1.506	334	2.642	732	102
ibm12	3.296	2.934	1.029	3.065	1.998	1.518	4.482	1.200	447
ibm13	1.730	898	636	1.638	882	579	2.072	634	93
Média	1.611	1.188	774	1.533	1.136	678	2.208	749	190

A Figura 5.24 apresenta a distribuição das vias-3D em um projeto com três *tiers*. As reduções do número total dessas vias pelo algoritmo LHr-3D, em comparação a hMetis e LHp-3D, foram de 17 e 8%, respectivamente. O gráfico também mostra o aumento das vias-3D adjacentes em detrimento das longas, ocasionado pela função de custo do algoritmo. Os valores que geraram esse gráfico estão descritos na Tabela 5.18 e são resultados do seguinte processo: dado um projeto 3D com três camadas denominadas *tier 1, tier 2 e tier 3,* reduza o número de conexões entre a primeira e a terceira de modo que haja convergência para a diminuição do número total de vias-3D. Assim, os resultados mostram que, para o algoritmo hMetis, a média das vias que cruzam as *tiers 1* e *3* é de 821. Já para o algoritmo LHr-3D, essa média cai para 330, uma redução de 60%. Ao mesmo tempo, ocorre um aumento de apenas 4% no número de vias-3D entre as *tiers 1 e 2*, para esse algoritmo (65 vias).

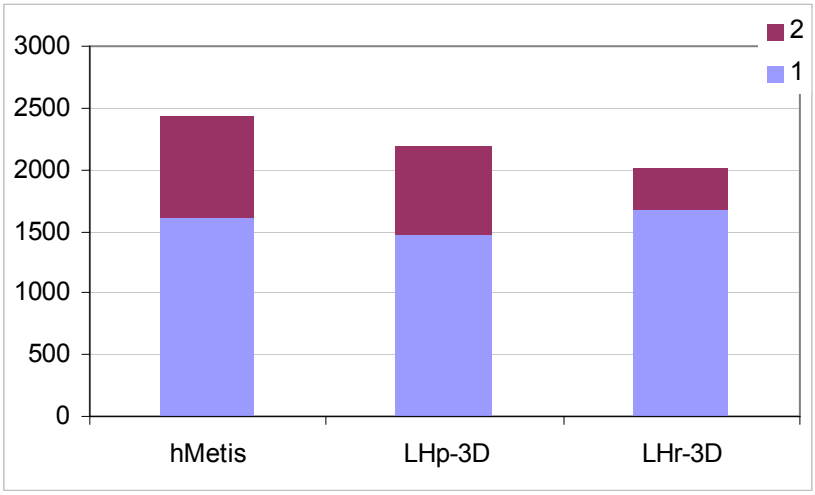


Figura 5.24: Distribuição das vias-3D em um projeto de 3 *tiers*.

Tabela 5.18: Distribuição das vias-3D em um projeto de 3 *tiers.*

Benchs	**hMetis**		**LHp-3D**		**LHr-3D (Proposta)**	
	1	**2**	**1**	**2**	**1**	**2**
Ibm01	467	268	403	122	386	44
Ibm02	546	336	481	266	503	80
Ibm03	1.143	1.114	1.078	1.096	1.444	596
Ibm04	1.365	218	1.261	250	1.276	72
Ibm05	2.765	1.886	2.273	2.038	2.947	958
Ibm06	1.217	610	1.154	488	1.213	326
Ibm07	1.980	58	1.972	78	1.899	38
Ibm08	2.006	808	1.863	834	2.197	458
Ibm09	1.518	386	1.216	656	1.436	290
Ibm10	1.883	1.672	1.567	1.094	2.219	358
Ibm11	1.909	672	1.812	428	1.927	204
Ibm12	2.520	1.950	2.534	1.560	2.996	712
Ibm13	1.606	692	1.481	412	1.334	148
Média	1.610	821	1.469	717	1.675	330

5.6.3.1 Área de Células x Área de vias-3D

Como mencionado anteriormente, as vias-3D ocupam maior área do que as conexões normais. Desse modo, para estratégias *face-to-back e back-to-back*, a área dessas vias pode ser maior do que a das células. Este tópico busca representar e discutir essa proporção conforme experimentos realizados no conjunto de circuitos *benchmark* ISPD 2004.

As figuras Figura 5.25, Figura 5.26, Figura 5.27 e Figura 5.28 comparam o algoritmo mais usado para o particionamento de células e pinos de I/O em circuitos 3D, hMetis, e o algoritmo proposto neste subcapítulo, LHr-3D. Esse experimento utiliza tecnologia de vias-3D *bulk pitch* 50 μm com duas, três, quatro e cinco *tiers.* O experimento explora a média do maior número de vias-3D que cruzam duas camadas adjacentes, juntamente com a média da *tier* que tem a maior área de células do conjunto de *benchmarks.* A Tabela 5.19 apresenta os dados absolutos do experimento. Este foi organizado com uma coluna "área de células", que mostra os dados da *tier* com a maior área de células (não necessariamente o maior número destas, visto que o balanceamento das camadas é realizado de acordo com a área e não com o número de células, como mencionado anteriormente). A coluna "max # de vias-3D" representa o maior número de vias que cruzam duas *tiers* adjacentes. Já a coluna "área de vias-3D *pitch* 50 μm" indica a maior área de vias ocupada no circuito, sendo ela a multiplicação da coluna "max # vias-3D" pela área da via. Nesse caso, a área de cada via corresponde a 2.500μm^2. A coluna "área total", por sua vez, representa a soma da maior área de células com a maior área ocupada pelas vias-3D.

O experimento representado na Figura 5.25 mostra a proporção ocupada pela área de células e pela área de vias-3D. Foram utilizados os algoritmos hMetis e LHr-3D com duas *tiers*. Verifica-se pelo gráfico que, com hMetis, a área ocupada pelas vias-3D é de 47%, e a área de células chega a 53%. Nesse caso, a área ocupada pelas vias é 6% menor do que a ocupada pelas células. Já o algoritmo LHr-3D reduz a área dessas vias de 47 para 40%, aumentando, proporcionalmente, a área ocupada pelas células de 53 para 60%.

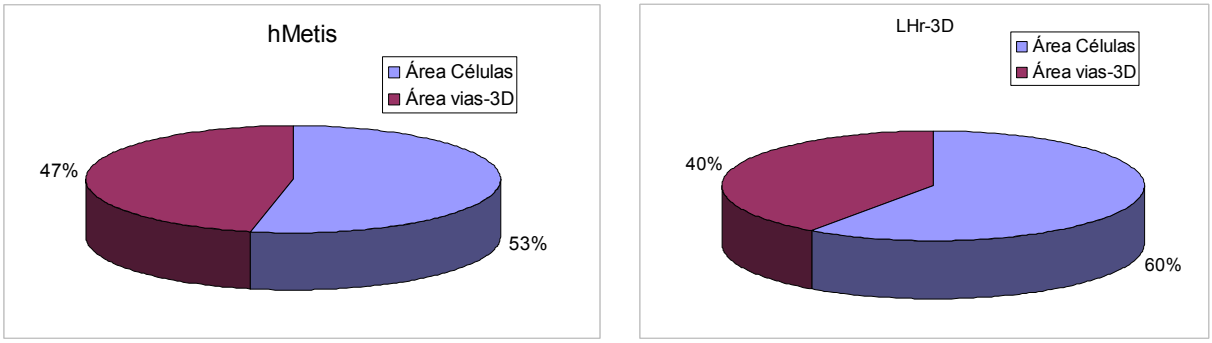


Figura 5.25: Espaço ocupado pelas vias-3D e pela área de células em um projeto de 2 *tiers* utilizando tecnologia *bulk* 50 μm

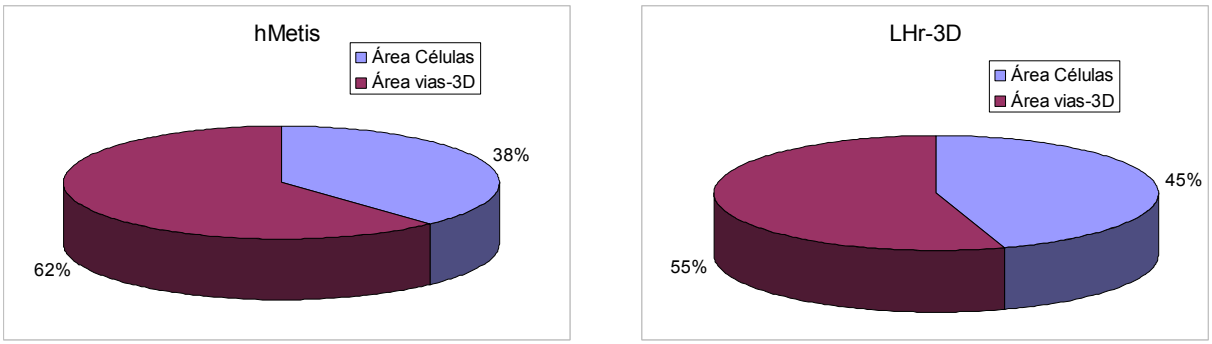


Figura 5.26: Espaço ocupado pelas vias-3D e pela área de células em um projeto de 3 *tiers* utilizando tecnologia *bulk* 50 μm.

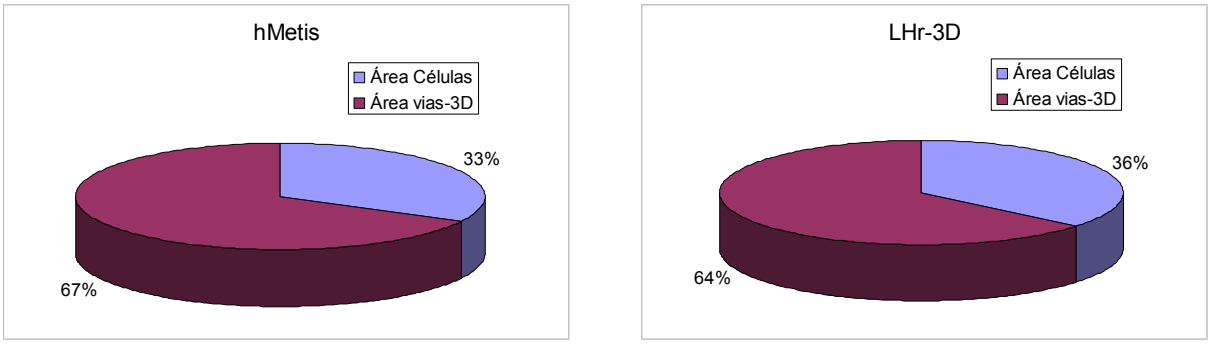


Figura 5.27: Espaço ocupado pelas vias-3D e pela área de células em um projeto de 4 *tiers* utilizando tecnologia *bulk* 50 μm.

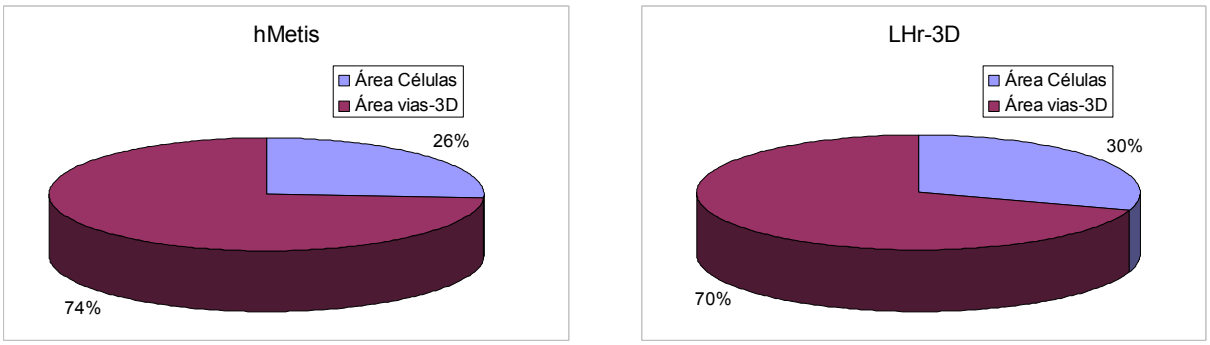


Figura 5.28: Espaço ocupado pelas vias-3D e pela área de células em um projeto de 5 *tiers* utilizando tecnologia *bulk* 50 μm.

A Figura 5.26 mostra que a proporção entre a área de células e a de vias-3D começa a mudar. Percebe-se que, quanto maior o número de *tiers*, maior o aumento do número de vias e, consequentemente, maior a diferença entre elas. Nota-se que a área ocupada pelas células, usando-se o algoritmo hMetis com três *tiers*, é de 38%, e a área de vias-3D cresce para 62%. Nesse caso, a área de vias é 24% maior do que a de células. Em comparação a esses resultados, obtém-se, com o algoritmo LHr-3D, 55% de área de vias-3D e 45% de área de células, diminuindo-se a diferença entre as ocupações entre elas para 10%. Na relação entre os dois algoritmos, nota-se que a área de vias-3D de hMetis é 7% maior do que a área obtida usando-se o LHr-3D.

A Figura 5.27 ilustra a continuação do aumento do número de vias-3D em relação à área das células. São ocupados 67% da área do circuito com as vias e apenas 33% com

células, segundo os resultados obtidos pelo algoritmo hMetis. Por outro lado, com LHr-3D, a área de células torna-se de 36% e a de vias, 64%.

A Figura 5.28 confirma o aumento do número de vias-3D em relação à área ocupada pelas células. Usando-se o algoritmo hMetis, 74% da área do circuito fica ocupada pelas vias e apenas 26% pelas células. Já os resultados do LHr-3D são de 70% de área de vias-3D e 30% da de células.

Percebe-se que, em circuitos com mais de duas *tiers,* a área ocupada pelas vias-3D é maior do que a que as células ocupam. Isso se deve, primeiramente, ao tamanho das vias proposto para os experimentos. Por outro lado, a percentagem de redução é bastante interessante, visto que o número absoluto das vias cresce juntamente com o número de *tiers* (40% em média para cada nova *tier*). Assim, como mostram os resultados, o número de vias-3D reais que são eliminadas do projeto é bastante significativo (em média, 200 a 400 vias, dependendo do número de camadas). Com essa proporção, a margem de erros em fabricação, alinhamento e ocupação de recursos de roteamento também são menores.

Tabela 5.19: hMetis e LHr-3D área da via-3D *bulk* 50μm

Algoritmos	Tiers	Area Células Maior Tier	Max # vias-3D	Area vias-3D pitch 50 μm	Area Total da Tier
hMetis	2	3.844.129	1.496	3.740.000	7.584.129
	3	2.587.478	1.852	4.630.000	7.217.478
	4	1.917.220	1.667	4.167.500	6.757.220
	5	1.553.191	1.936	4.840.000	6.393.191
LHr-3D	2	3.769.135	1.264	3.160.000	6.929.135
	3	2.329.226	1.347	3.367.500	5.696.726
	4	1.739.185	1.472	3.680.000	5.419.185
	5	1.401.041	1.608	4.020.000	5.421.041

A

Tabela 5.20 mostra dados das áreas ocupadas pelas células e pelas vias-3D num projeto com a tecnologia *bulk pitch* 25µm. A redução da área das vias pelo LHr-3D, em comparação com o algoritmo hMetis, foi de 15%, 25%, 10% e 13% para duas, três, quatro e cinco *tiers*, respectivamente. Como destacado anteriormente, a proporção entre a área ocupada pelas vias-3D e a ocupada pelas células está diretamente relacionada à área da via. Quanto maior esta é, maior será a área que seu conjunto ocupa. No experimento anterior, utilizou-se a tecnologia *bulk pitch* 50µm. Percebeu-se que a área de vias-3D ocupada em um circuito é bastante grande em comparação com a área de células, quando se utiliza mais do que duas *tiers*. Por outro lado, existem tecnologias que utilizam *pitches* menores, principalmente quando a metodologia de integração é *face-to-face* (o que limita o projeto a duas camadas). Contudo, quanto menor a área da via-3D, mais difícil é o seu alinhamento, como é perceptível no momento da sobreposição das *tiers.* Pesquisas em (BERNSTEIN, 2006) demonstram que a redução do *pitch* das vias-3D aumenta o número de erros de fabricação e, principalmente, dos erros de alinhamento das vias, o que diminui sua confiabilidade.

Tabela 5.20: hMetis e LHr-3D área da via-3D *bulk pitch* 25μm

Algoritmos	Tiers	Area Células Maior Tier	Max # vias-3D	Area vias-3D pitch 25 μm	Area Total da Tier
hMetis	2	3.844.129	1.496	935.000	4.779.129
	3	2.587.478	1.852	1.157.500	3.744.978
	4	1.917.220	1.667	1.041.875	2.959.095
	5	1.553.191	1.936	1.210.000	2.763.191
LHr-3D	2	3.769.135	1.264	790.000	4.559.135
	3	2.329.226	1.347	841.875	3.171.101
	4	1.739.185	1.472	920.000	2.659.185
	5	1.401.041	1.608	1.005.000	2.406.041

Analisando os aspectos descritos, conclui-se que a redução do número de vias é importante para a redução da área das *tiers*, o que reflete, por sua vez, na redução das conexões, dos atrasos e do consumo.

5.6.4 Comparação com ZPlace

Este tópico compara os algoritmos hMetis, LHp-3D e LHr-3D com a ferramenta ZPlace (HENTSCHKE et al., 2007). ZPlace é um algoritmo direcionado a forças que usa o modelo de *quadratic placement* para distribuir as células entre as *tiers*, como mencionado na Seção 4.7. Assim, o particionamento é formulado como um sistema que usa molas para modelar as conexões entre os nodos (células e pinos). As molas atraem os nodos e os conecta um perto do outro. Na ausência de nodos fixos (pinos de I/O), todos eles tendem a se aglomerar num mesmo ponto. Portanto, em um algoritmo direcionado a forças, os pinos têm um papel importante na distribuição dos nodos móveis (células), pois serve de referência para a posição das células.

Com o uso da ferramenta ZPlace, o espaço de posicionamento formado pelos pinos de I/O resulta em um cubo. O sistema de molas posiciona as células em qualquer lugar dentro desse cubo. Não há uma divisão clara que indique a qual *tier* elas pertençam. O ZPlace utiliza, então, a posição *z* das células para escolher a camada em que as células serão temporariamente colocadas. Essa *tier* é definida varrendo-se as células do menor para o maior valor de *z* e colocando-se tantas células quanto couberem na primeira *tier*, passando-se então para a segunda, e assim por diante.

Após definida a camada a que cada célula pertence naquela iteração, forças (molas) são adicionadas para atrair todas as células para planos verticais que representam as *tiers* dentro do cubo. À medida que o processo avança, as forças adicionadas tentam aproximar as células do plano que representa a tier em que elas serão posicionadas.

Como mencionado, a ferramenta ZPlace necessita da informação da posição dos pinos para funcionar. Para os primeiros experimentos, essa posição é obtida por meio do particionador de hipergrafos hMetis. Nesse caso, hMetis particiona primeiro os pinos de I/O e retorna a informação para a ferramenta. A ferramenta ZPlace foi configurada para obter a máxima redução do número total de vias-3D. Os resultados obtidos pelos algoritmos LHr-3D e ZPlace estão ilustrados na Tabela 5.21. Percebe-se pelos dados que, mesmo com as *tiers* em linha (como é a proposta do LHr-3D), o algoritmo direcionado a forças proposto na ferramenta ZPlace não se mostra muito eficiente para a redução do número de vias-3D. Nota-se que LHr-3D teve melhor desempenho em todos os circuitos *benchmarks*. **Isso é devido ao fato da ferramenta ZPlace considerar**

somente as conexões que correm no eixo *z* como critério de redução de vias, sendo em sua maioria posicionados em apenas duas *tiers* adjacentes. Assim, visivelmente os resultados dessa ferramenta foram piores do que LHr-3D.

Tabela 5.21: Número de vias-3D entre LHr-3D e ZPlace (hMetis).

	LHr-3D				**ZPlace (hMetis)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	274	430	737	952	547	733	1.161	1.479
ibm02	376	583	1.055	1.229	581	885	1.519	2.723
ibm03	963	2.040	2.530	3.804	1.595	2.450	3.333	4.684
ibm04	614	1.348	2.140	2.514	691	1.847	3.123	3.250
ibm05	1.931	3.905	6.015	7.936	3.315	5.790	6.677	9.796
ibm06	986	1.539	2.876	3.443	1.367	2.764	3.194	4.268
ibm07	875	1.937	3.216	4.310	1.035	2.447	3.450	4.626
ibm08	1.161	2.655	4.088	5.276	1.748	4.103	5.342	7.252
ibm09	652	1.726	2.385	3.167	1.134	2.042	3.655	3.571
ibm10	1.340	2.577	3.467	5.263	1.819	4.719	6.057	7.538
ibm11	1.085	2.131	3.476	4.349	1.821	3.091	5.340	5.698
ibm12	2.114	3.708	6.129	7.911	3.437	4.629	9.132	10.229
ibm13	866	1.482	2.799	3.256	1.388	2.351	3.513	5.273
ibm14	2.017	3.529	5.140	6.429	2.836	4.251	6.074	8.351
ibm15	2.877	4.670	6.916	9.062	3.462	6.485	8.319	11.060
ibm16	2.061	5.386	5.838	8.811	2518	5.781	8.197	9.575
Média	1.262	2.478	3.675	4.857	1.831	3.398	4.880	6.211

O gráfico da Figura 5.29 mostra a quantidade de vias-3D distribuídas em projetos com duas, três, quatro e cinco *tiers*. O algoritmo LHr-3D obteve redução do número de vias-3D com percentuais de 35%, 31%, 26% e 24% em comparação com a ferramenta ZPlace para duas, três, quatro e cinco *tiers* respectivamente.

Ambas as estratégias posicionam suas *tiers* em linha, simulando um circuito 3D real. Contudo, fica claro pelos experimentos, que a estratégia direcionada a forças, não mostra soluções eficientes, pois as abordagens atuais não são direcionadas especificamente para a redução de vias-3D. Nesse sentido, para uma comparação justa, o experimento modificou a função de custo a fim de priorizar a redução das conexões verticais, constatou-se pelos resultados obtidos por essa estratégia não convergiu para a redução total do número de vias-3D.

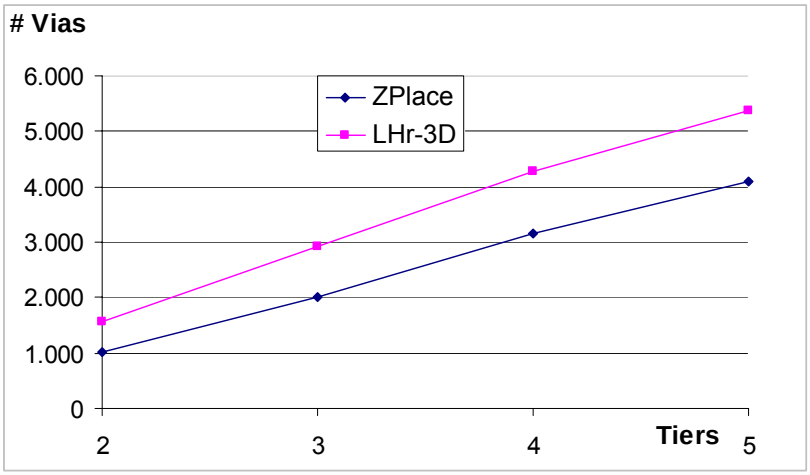


Figura 5.29: Número de vias entre ZPlace (hMetis) e LHr-3D.

A Tabela 5.22 mostra resultados obtidos pelos algoritmos ZPlace e LHr-3D relativos a redução do número máximo de vias-3D. Percebe-se, pela tabela, que todos os circuitos *benchmarks* do algoritmo LHr-3D alcançaram resultados melhores do que o hMetis. Em média, a redução foi de 35%, 33%, 32% e 26% para duas, três, quatro e cinco *tiers* respectivamente. Sabe-se que quanto maior o número de *tiers*, maior será o número de vias-3D.

Tabela 5.22: Número máximo de vias-3D entre LHr-3D e ZPlace (hMetis).

	LHr-3D				**ZPlace (hMetis)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	274	293	284	341	547	443	586	595
ibm02	376	318	384	492	581	515	621	755
ibm03	963	1.103	1.018	1.175	1.595	1.287	1.210	1.357
ibm04	614	815	942	844	691	1.507	1.427	1.331
ibm05	1.931	1.995	2.245	2.225	3.315	2.995	2.546	3.212
ibm06	986	779	1.094	922	1.367	1.528	1.136	1.227
ibm07	907	1.239	1.462	1.473	1.035	1.533	1.523	1.695
ibm08	1.161	1.364	1.502	1.528	1.748	2.242	2.239	2.491
ibm09	652	975	954	957	1.134	1.478	1.531	1.533
ibm10	1.340	1.377	1.374	1.634	1.819	2.432	2.602	2.758
ibm11	1.085	1.460	1.467	1.615	1.821	2.349	2.033	1.763
ibm12	2.114	2.062	2.285	3.096	3.437	2.511	4.632	3.443
ibm13	866	1.033	1.461	1.640	1.388	1.268	1.960	1.987
ibm14	2.017	2.034	2.103	2.245	2.836	2.134	2.368	3.210
ibm15	2.877	2.715	2.939	3.125	3.462	3.511	3.351	4.162
ibm16	2.061	1.987	2.033	2.412	2.518	3.170	2.948	3.365
Média	1.264	1.347	1.472	1.608	1.831	1.931	2.045	2.180

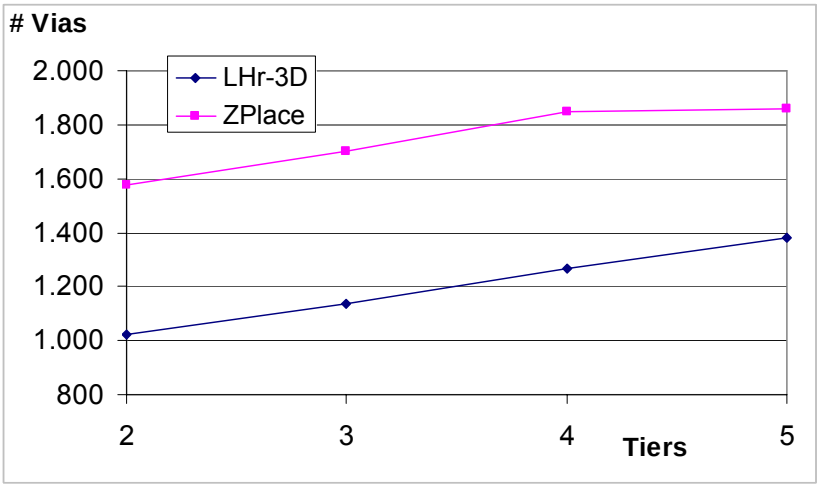


Figura 5.30: Número máximo de vias entre ZPlace (hMetis) e LHr-3D.

A Tabela 5.23 compara os algoritmos hMetis e ZPlace em número de vias-3D. Percebe-se que ZPlace obteve, em média, resultados piores do que o algoritmo hMetis, mesmo com sua configuração dirigida para a redução de vias-3D. O algoritmo hMetis obteve, em média, reduções de 21%, 16%, 16% e 8% para circuitos com duas, três, quatro e cinco *tiers*, o que comprova a qualidade de sua estrutura, mesmo adaptada para resolver questões diferentes.

Tabela 5.23: Número de vias-3D entre hMetis e ZPlace.

	hMetis				**ZPlace (hMetis)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
Ibm01	441	735	838	1.439	547	733	1.161	1.479
Ibm02	547	882	1.214	1.600	581	885	1.519	2.723
Ibm03	1.146	2.257	2.693	4.020	1.595	2.450	3.333	4.684
Ibm04	738	1.583	2.516	3.202	691	1.847	3.123	3.250
Ibm05	2.417	4.651	6.653	9.651	3.315	5.790	6.677	9.796
Ibm06	1.061	1.827	3.128	3.566	1.367	2.764	3.194	4.268
Ibm07	994	2.038	3.302	4.605	1.035	2.447	3.450	4.626
Ibm08	1.324	2.814	4.184	5.698	1.748	4.103	5.342	7.252
Ibm09	806	1.904	2.763	3.518	1.134	2.042	3.655	3.571
Ibm10	1.771	3.555	4.675	7.116	1.819	4.719	6.057	7.538
Ibm11	1.490	2.581	3.958	5.697	1.821	3.091	5.340	5.698
Ibm12	2.594	4.470	7.259	9.187	3.437	4.629	9.132	10.229
Ibm13	1.193	2.298	3.264	4.557	1.388	2.351	3.513	5.273
Ibm14	2.171	4.561	6.584	8.085	2.836	4.251	6.074	8.351
Ibm15	3.002	7.863	9.082	11.707	3.462	6.485	8.319	11.060
Ibm16	2.237	5.816	6.235	9.300	2518	5.781	8.197	9.575
Média	1.496	3.111	4.272	5.809	1.831	3.398	4.880	6.211

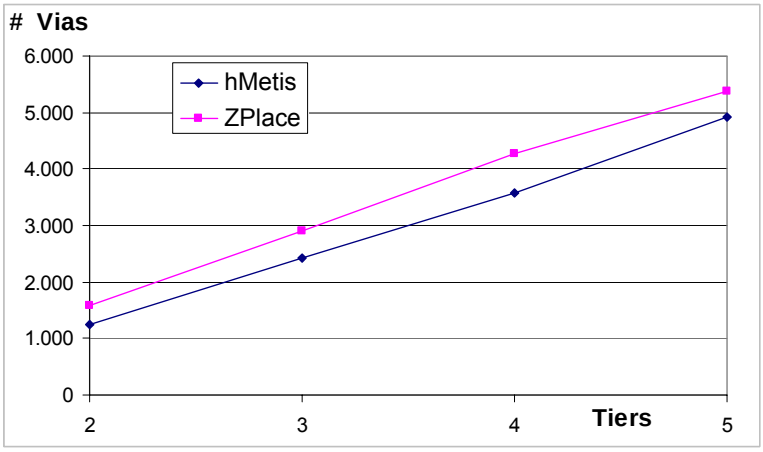


Figura 5.31: Número de vias entre ZPlace e hMetis.

O experimento ilustrado pela Figura 5.32 mostra a ferramenta ZPlace usando diferentes estratégias de particionamento de pinos de I/O. Como se sabe, a redução do número de vias-3D se dará por meio de forças de atração impostas pelos pinos de I/O. Nesse caso, o deslocamento das células ocorre entre as *tiers*, através do eixo *z*.

O experimento chamado de ZPlace (hMetis) usa a informação dos pinos de I/O obtida pela ferramenta hMetis. Esse é o procedimento utilizado pela maioria dos posicionadores. É importante ressaltar que o item 5.4 apresentou uma forma de melhorar a qualidade dos resultados de corte dos particionadores de hipergrafos atuais. O estudo de caso focou na ferramenta hMetis, atualmente, a mais utilizada no domínio VLSI.

O intuito do novo experimento é verificar se o critério do menor caminho lógico desenvolvido, possibilita também a redução do número de vias-3D em algoritmos direcionados a força com *tiers* fixas, como é o caso do ZPlace.

Já o experimento nominado ZPlace (I/O Pins), assim como, o algoritmo LHp-3D, usam a informação do pinos de I/O para realizar o particionamento das células. A principal diferença entre essas duas estratégias é que ZPlace (I/O Pins) trabalha com as *tiers* em linha, enquanto LHp-3D melhora o corte entre as partições e com isso contribui para a redução do número total de vias. Para tanto, na comparação entre as duas estratégias de particionamento de pinos de I/O, verificou-se que o ZPlace, usando o menor caminho lógico, obteve resultados melhores do que o simples particionamento realizado pelo hMetis. A redução do ZPlace (I/O Pins) foi de 5%, 6%, 6% e 4% para duas, três, quatro e cinco *tiers* respectivamente em comparação com o ZPlace (hMetis). Esse resultado mostra que o particionamento dos pinos de I/O usado de forma simplista piora o resultado do número de vias dos particionadores, mesmo atuando em camadas fixas.

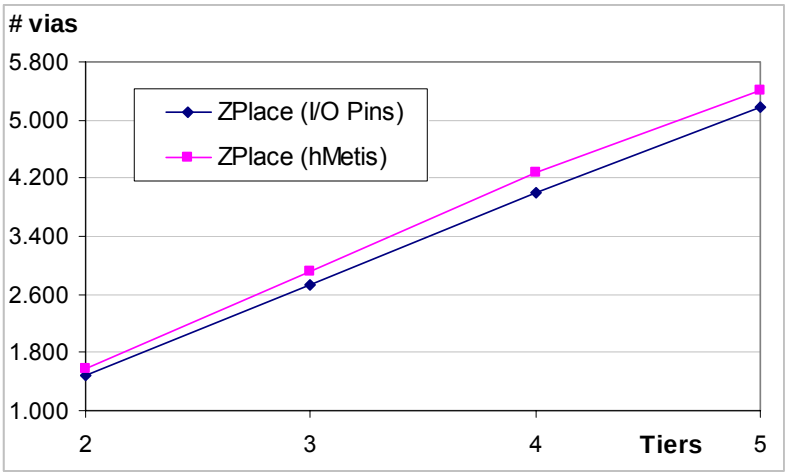


Figura 5.32: Número de vias-3D obtidas pela ferramenta ZPlace usando diferentes estratégias de pinos de I/O.

Tabela 5.24: Número total de vias com ZPlace usando diferentes posições de pinos de I/O

	ZPlace (I/O Pins)				ZPlace (hMetis)			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	509	694	865	1.464	547	733	1.161	1.479
ibm02	417	824	1.590	2.699	581	885	1.519	2.723
ibm03	1.421	2.288	3.404	4.278	1.595	2.450	3.333	4.684
ibm04	1.524	2.286	2.668	3.605	691	1.847	3.123	3.250
ibm05	3.223	4.272	6.308	9.687	3.315	5.790	6.677	9.796
ibm06	1.227	2.431	3.429	4.269	1.367	2.764	3.194	4.268
ibm07	1.525	2.671	3.688	4.965	1.035	2.447	3.450	4.626
ibm08	1.482	3.393	5.447	6.920	1.748	4.103	5.342	7.252
ibm09	728	2.463	3.351	4.216	1.134	2.042	3.655	3.571
ibm10	1.696	3.325	5.449	6.451	1.819	4.719	6.057	7.538
ibm11	1.468	2.682	4.486	5.148	1.821	3.091	5.340	5.698
ibm12	2.806	5.241	7.078	8.851	3.437	4.629	9.132	10.229
ibm13	1.338	2.903	4.104	4.679	1.388	2.351	3.513	5.273
ibm14	2.841	4.246	6.204	8.296	2.836	4.251	6.074	8.351
ibm15	3.342	6.823	8.825	10.915	3.462	6.485	8.319	11.060
ibm16	2.409	4.963	7.634	8.762	2518	5.781	8.197	9.575
Média	1.747	3.219	4.658	5.950	1.831	3.398	4.880	6.211

A Tabela 5.24 mostra os resultados do experimento entre as duas estratégias de particionamento de pinos. Verifica-se que a informação do critério do menor caminho lógico contida no algoritmo ZPlace melhorou o resultados em todos os circuitos *bechmarks* executados. Isso reforça a idéia deste trabalho de que a estrutura interna do

circuito contém dados que podem ajudar a melhorar ainda mais a qualidade dos resultados atuais.

5.7 Resumo das Reduções do Número Total e Máximo de vias-3D

Essa seção resume os resultados gerados pelos algoritmos LHp-3D, LHr-3D, hMetis, Alternate Pins, ZPlace (I/O Pins) e ZPlace (hMetis) nas reduções total e máxima do número de vias em circuitos 3D. Os valores encontrados são apresentados na Tabela 5.25, Tabela 5.26 e Tabela 5.27 e comparados na Figura 5.33 e Figura 5.34.

Tabela 5.25: Número máximo de vias-3D entre hMetis e ZPlace (I/O Pins)

	hMetis				**ZPlace (I/O Pins)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	441	467	377	573	509	453	364	452
ibm02	547	496	485	552	417	420	606	802
ibm03	1.146	1.143	1.021	1.334	1.421	1.197	1.502	1.363
ibm04	738	862	1.067	1.039	1.524	1.608	1.190	1.361
ibm05	2.417	2.765	2.478	2.712	3.223	2.367	2.211	3.090
ibm06	1.061	924	1.134	935	1.227	1.322	1.262	1.305
ibm07	994	1.980	1.510	1.525	1.525	1.922	1.534	1.713
ibm08	1.324	1.436	1.445	1.788	1.482	1.926	2.327	2.286
ibm09	806	1.598	1.092	1.249	728	1.260	1.474	1.221
ibm10	1.771	1.883	1.741	1.986	1.696	1.995	2.288	2.350
ibm11	1.490	1.909	1.810	2.230	1.468	1.852	1.842	1.688
ibm12	2.594	2.820	2.747	2.962	2.806	3.457	2.766	3.228
ibm13	1.193	1.606	1.500	1.755	1.338	2.131	2.129	2.129
ibm14	2.171	2.375	2.307	2.881	2.841	2.846	3.248	2.709
ibm15	3.002	4.188	3.377	4.099	3.242	4.491	4.043	3.725
ibm16	2.237	3.185	2.266	3.355	2.841	2.647	2.877	3.145
Média	1.496	1.852	1.647	1.936	1.768	1.993	1.979	2.035

Tabela 5.26: Número máximo de vias-3D LHp-3D e ZPlace (hMetis)

	LHp-3D				**ZPlace (hMetis)**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	369	305	322	406	547	443	586	595
ibm02	396	413	403	594	581	515	621	755
ibm03	1.059	1.187	1.086	1.225	1.595	1.287	1.210	1.357
ibm04	735	887	992	887	691	1.507	1.427	1.331
ibm05	2.258	2.203	2.469	2.729	3.315	2.995	2.546	3.212
ibm06	1.059	849	1.135	948	1.367	1.528	1.136	1.227
ibm07	992	1.332	1.433	1.524	1.035	1.533	1.523	1.695
ibm08	1.298	1.448	1.397	1.610	1.748	2.242	2.239	2.491
ibm09	699	1.057	1.008	1.075	1.134	1.478	1.531	1.533
ibm10	1.490	1.450	1.590	1.750	1.819	2.432	2.602	2.758
ibm11	1.190	1.485	1.605	1.719	1.821	2.349	2.033	1.763
ibm12	2.293	2.278	2.422	3.173	3.437	2.511	4.632	3.443

ibm13	1.042	1.269	1.548	1.781	1.388	1.268	1.960	1.987
ibm14	2.121	2.272	2.248	2.459	2.836	2.134	2.368	3.210
ibm15	2.890	2.857	3.199	3.395	3.462	3.511	3.351	4.162
ibm16	2.102	2.164	2.212	2.625	2.518	3.170	2.948	3.365
Média	1.375	1.466	1.567	1.744	1.831	1.931	2.045	2.180

Tabela 5.27: Número máximo de vias-3D entre LHr-3D e ZPlace (I/O Pins)

	LHr-3D				**Alternate Pins**			
Tiers	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**5**
ibm01	274	293	284	341	428	483	406	480
ibm02	376	318	384	492	503	469	498	553
ibm03	963	1.103	1.018	1.175	1.099	1.320	1.485	1.210
ibm04	614	815	942	844	750	913	1.033	1.454
ibm05	1.931	1.995	2.245	2.225	2.576	2.814	2.526	3.974
ibm06	986	779	1.094	922	1.075	915	1.193	937
ibm07	907	1.239	1.462	1.473	1.049	2.050	1.590	2.402
ibm08	1.161	1.364	1.502	1.528	1.307	1.919	1.448	1.833
ibm09	652	975	954	957	780	1.356	1.684	1.137
ibm10	1.340	1.377	1.374	1.634	1.821	2.247	1.724	2.898
ibm11	1.085	1.460	1.467	1.615	1.494	1.856	1.802	2.610
ibm12	2.114	2.062	2.285	3.096	2.556	3.160	3.113	4.205
ibm13	866	1.033	1.461	1.640	1.170	1.611	1.905	1.954
ibm14	2.017	2.034	2.103	2.245	2.310	2.619	3.274	3.283
ibm15	2.877	2.715	2.939	3.125	3.126	4.207	4.385	4.163
ibm16	2.061	1.987	2.033	2.412	2.280	3.704	3.794	3.443
Média	1.264	1.347	1.472	1.608	1.520	1.978	1.991	2.284

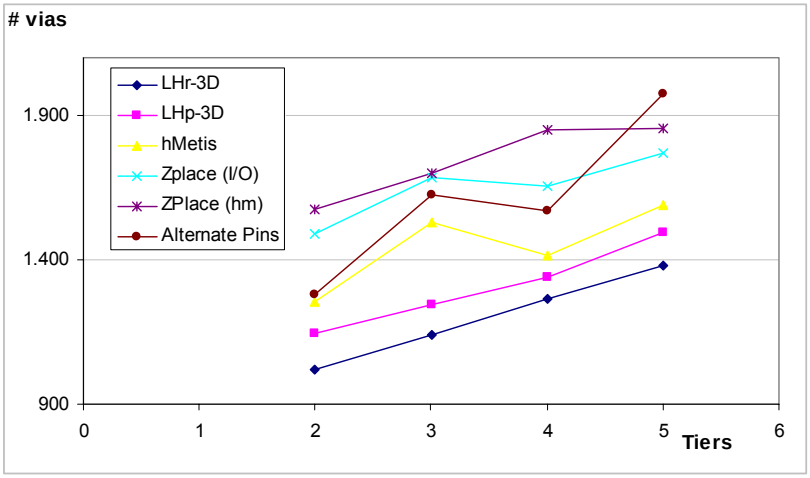


Figura 5.33: Máximo número de vias-3D entre LHp-3D, LHr-3D, hMetis, Alternate Pins, ZPlace (I/O Pins) e ZPlace (hMetis).

Ao observar a Figura 5.33 percebe-se que o comportamento dos algoritmos LHr-3D e LHp-3D são bastante parecidos. Ambos têm o crescimento praticamente linear (verifica-se organização no crescimento). Por outro lado, nos demais algoritmos

avaliados, o crescimento do número máximo de vias não segue o mesmo padrão. Os algoritmos hMetis, Alternate Pins e ZPlace (I/O) tem um grande pico quando projetados em três *tiers* e uma queda logo em seguida na quarta *tier*. Constou-se, contudo, que esses algoritmos não obtiveram bons resultados na redução do número de vias em três camadas e ainda mantiveram muitas conexões cruzando sua *tier* não-adjacente. A soma das conexões da sua *tier* adjacente com a não-adjacente ocasionou o aumento do número de conexões entre duas específicas. Já no projeto de quatro *tiers*, ocorreu uma distribuição das vias equilibrada entre as camadas, entretanto para o algoritmo Alternate Pins o comportamento anterior ocorreu novamente. Isso mostra que atualmente a execução dos algoritmos de particionamento que atuam em projetos 3D é simplista, pois desconhecem completamente algumas restrições de projeto, como é o caso, por exemplo, do número máximo de conexões.

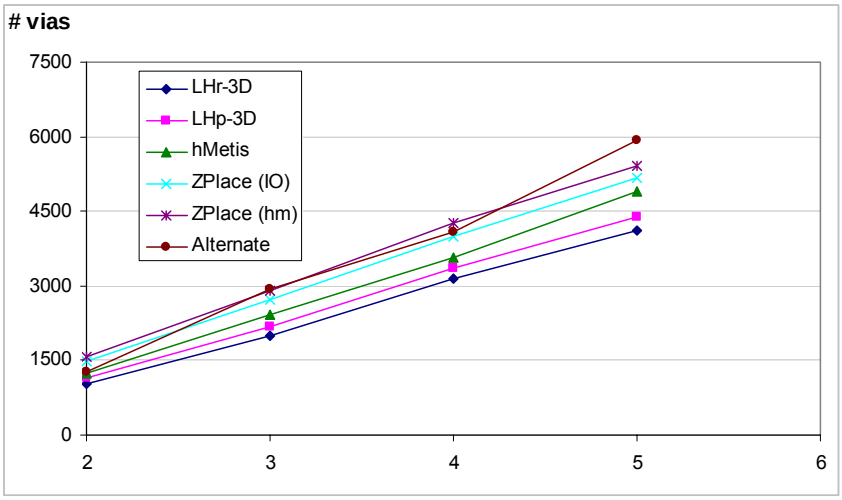


Figura 5.34: Número total de vias-3D entre LHp-3D, LHr-3D, hMetis, Alternate Pins, ZPlace (I/O Pins) e ZPlace (hMetis).

A Figura 5.34 mostra o número total de vias gerados pelos algoritmos LHp-3D, LHr-3D, hMetis, Alternate Pins, ZPlace (I/O Pins) e ZPlace (hMetis) em projetos com duas, três, quatro e cinco *tiers*. Percebe-se que os melhores resultados foram obtidos pelos algoritmos LHr-3D e LHp-3D.

5.8 Tempos de Execução

Esta seção apresenta o tempo de execução dos algoritmos que compõem o LH-3D. O cenário de experimentações foi um processador PowerPC G5-Dual 1.8GHZ, 4GB SDRAM DDR. Os algoritmos desenvolvidos foram implementados em linguagem C++ (compilador g++ 4.0.1 (GCC) 4.0.1 Apple Computer, Inc. build 5370.

A Tabela 5.28 mostra o tempo gasto para encontrar o menor caminho lógico entre pares de pinos de I/O. Analisando os dados da tabela, verifica-se que o maior tempo corresponde ao circuito ibm16, o qual o contém maior número de células e redes. Constata-se, contudo, que o número de redes e células é um dos fatores que influenciam no tempo de execução do algoritmo, mas não é o determinante. Em alguns casos, o tempo de execução foi menor, mesmo com o número de células e redes maior. Nesse caso, o que influenciou foi o número de células percorridas pelo algoritmo para encontrar os caminhos (distâncias lógicas maiores).

O caminho lógico é calculado somente uma vez para cada circuito. Como sua estrutura não é modificada, essa informação é utilizada para qualquer número de partições, com isso, o custo computacional fica bastante reduzido, pois não há necessidade de calculá-lo a cada execução do algoritmo.

A Tabela 5.29 apresenta o tempo de execução somente da etapa de particionamento utilizando as informações do menor caminho lógico. Essa é a etapa mais custosa do algoritmo LHp-3D. Percebe-se que o tempo aumenta de acordo com o número de partições. Os circuitos ibm05 e ibm10 obtiveram tempos de execução diferente dos demais. Entretanto, analisando a estrutura desses circuitos verifica-se que eles contêm o maior número de pinos de I/O, 1.201 e 744. Isso explica o tempo gasto pelo particionador, pois tais elementos devem ser fixados antes do particionamento das células. Como é um número grande de pinos comparado com os demais, a avaliação do particionador se torna mais complexa, mesmo assim, o resultado final é obtido em um tempo computacional aceitável.

Tabela 5.28: Caminhos Lógicos

Benchs	# Células	# Redes	#Pinos	Tempo
ibm01	12.506	14.111	246	6s
ibm02	19.342	19.584	259	11s
ibm03	22.853	27.401	283	15s
ibm04	27.220	31.970	287	19s
ibm05	28.146	28.446	1201	16s
ibm06	32.332	34.826	166	40s
ibm07	45.639	48.117	287	35s
ibm08	51.023	50.513	286	2m38s
ibm09	53.110	60.902	285	1m24s
ibm10	68.685	75.196	744	1m44s
ibm11	70.152	81.454	406	2m25s
ibm12	70.439	77.240	637	2m05s
ibm13	83.709	99.666	490	4m12s
ibm14	147.088	152.772	517	3m36s
ibm15	161.187	186.608	383	5m26s
ibm16	182.980	190.048	504	6m50s

O número de *tiers* e o número de células têm influência direta no tempo de execução. Quanto maior o número de células, maior o tempo execução e, quanto maior o número de *tiers*, também é maior o tempo de execução. Sabe-se que o crescimento do número *tiers* aumenta proporcionalmente o número total e máximo de vias, com isso, é natural que o algoritmo necessite de mais perturbações. Da mesma forma acontece com o número de células, visto que o tamanho do problema também aumenta. Isso acarreta no aumento do número de comparações e, consequentemente, do tempo de execução. O tempo obtido nas execuções não é linear, contudo, comporta-se melhor do que um crescimento quadrático. Isso acontece, pois o algoritmo LHr-3D trabalha com uma

solução inicial muito boa. Seu objetivo é realizar perturbações de células a fim de reduzir o número de conexões não-adjacentes. Nesse caso, o algoritmo age como um refinador que melhora o resultado após um conjunto de iterações. Sua condição de parada ocorre quando não se consegue encontrar resultados melhores que os anteriores após um conjunto de tentativas.

Tabela 5.29: Tempo de execução do particionador usando o caminho lógico.

Benchs	**Tempo de Execução/Partições**			
	2	**3**	**4**	**5**
Ibm01	1s	1,5s	1,9s	2,3s
Ibm02	1,1s	1,6s	2s	2,4s
Ibm03	2,3s	1,7s	2,1s	2,6s
ibm04	1s	1,6s	2,1s	2,3s
ibm05	57s	76s	79s	84s
ibm06	0,8s	1,1s	1,6	1,9s
ibm07	1s	1,6s	2,1s	2,3s
ibm08	1s	1,6s	2s	2,1s
ibm09	1s	1,5s	2,1s	2,2s
ibm10	14s	18s	19s	20s
ibm11	3s	4s	5s	6s
ibm12	8s	13s	14s	15s
ibm13	4s	6s	6,5s	7s
ibm14	5s	6,7s	7,4s	7,7s
ibm15	2,3s	3,7s	4s	4,4s
ibm16	4,4s	6,3s	7,1s	7,4s

Os tempos de execução obtidos neste trabalho poderiam ser reduzidos com otimizações no código. No entanto, o objetivo deste trabalho não é o desenvolvimento de uma ferramenta de CAD para distribuição. Portanto, não foi despendido grande esforço na tarefa de sintetizar o código fonte.

5.9 Resumo do Capítulo

Levando-se em consideração os aspectos da otimização de circuitos 3D abordados, julgou-se necessário o desenvolvimento de uma estratégia direcionada especificamente à redução de vias. Em vista disso, este capítulo apresentou uma nova metodologia com base em duas estratégias distintas. A primeira, descrita nas seções 4.4 e 4.5, usa as informações do menor caminho lógico entre pares de pinos para auxiliar a redução do corte entre as partições. A segunda, apresentada na Seção 5.6, reduz vias não-adjacentes a partir da modificação de uma meta-heurística baseada em *simulated annealing*. Todos os experimentos foram gerados a partir do conjunto de circuitos *benchmark* IBM ISPD 2004.

Como mencionado, a primeira estratégia considera algumas informações da estrutura interna do circuito. Essa ideia partiu da observação de que os atuais algoritmos

de particionamento de hipergrafos não efetuam nenhuma análise prévia do circuito antes da sua execução. A partir dessa constatação, iniciou-se uma pesquisa sobre a atuação dos particionadores a fim de que se encontrasse alguma característica interna à estrutura do circuito que auxiliasse a obtenção de melhores resultados. No decorrer da análise do comportamento dos algoritmos posicionadores direcionados a forças, verificou-se que os pinos de I/O serviam de referência para o deslocamento das células. Durante a análise da estrutura das *netlists*, buscou-se por meio da distância lógica entre pares de pinos uma informação para fazer com que as células ligadas aos pinos com maior conectividade fossem direcionadas para as mesmas partições Para representar essa estrutura, criou-se um grafo completo que continha em suas arestas o peso da menor distância lógica entre os pinos. Baseado nessa informação, o grafo foi particionado e os pinos fixados nas diferentes partições. Após, um segundo particionamento foi executado somente com as células. Esse algoritmo foi chamado de LHp-3D e descrito na Seção 5.4.

O método foi avaliado por meio de experimentos que o compararam aos algoritmos hMetis e ZPlace. Como citado, hMetis é, atualmente, referência para todos os projetos e o mais utilizado em aplicações que necessitam desse tipo de solução. Já ZPlace é um algoritmo recente e direcionado a forças, com estratégias eficientes para a redução de *wirelenght*. A Figura 5.35 mostra o número total de arestas em duas, três, quatro e cinco partições. Verificou-se que, de posse das informações do circuito, o algoritmo LHp-3D obteve, para todas as partições, redução do número de arestas. Já a análise da Tabela 5.30 indica que o algoritmo ZPlace obteve o pior resultado. Isso se deve ao fato de as partições estarem alinhadas, o que provoca que muitas arestas sejam direcionadas para partições adjacentes de forma desequilibrada.

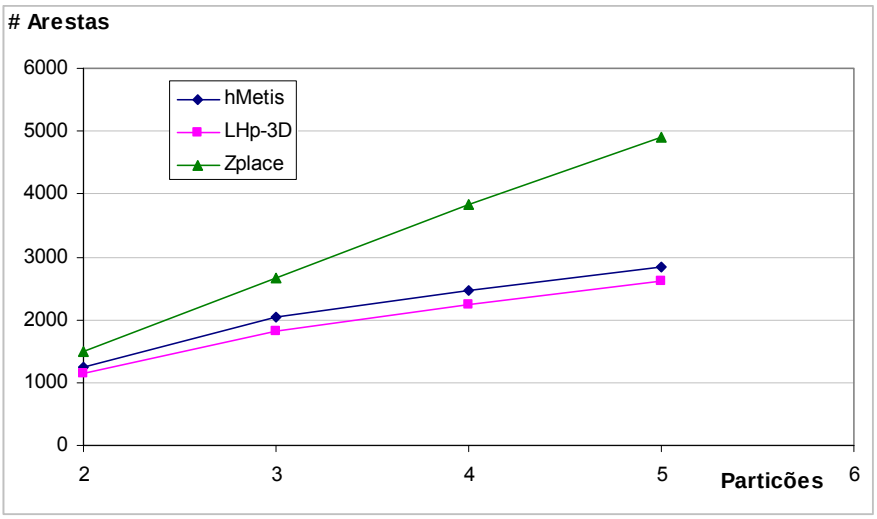


Figura 5.35: Qualidade do corte entre hMetis, LHp-3D e ZPlace.

A Tabela 5.30 apresenta os números absolutos ilustrados pela Figura 5.35. Tais números correspondem à soma de todas as arestas que cruzam as partições. Por exemplo, o circuito ibm01, com três partições, obteve o resultado de 464 arestas quando executado pelo LHp-3D. Isso significa que esse número corresponde à soma das arestas que cruzam as partições 1 e 2, 2 e 3, e 1 e 3. Percebeu-se que, para todos os circuitos do experimento, LHp-3D obteve redução do número de arestas. Se comparada com os resultados de hMetis, essa redução, em média, foi de 10%, 11%, 6% e 6% para duas,

três, quatro e cinco partições, respectivamente, o que comprova a eficácia de utilizar a informação dos pinos. Em comparação ao algoritmo ZPlace, a redução foi de 23%, 31%, 39% e 46% para as mesmas duas, três, quatro e cinco partições. A redução no corte também provocou a diminuição do número total de vias-3D, a qual foi, em média, de 10%, 10%, 6% e 11%, em comparação com o algoritmo hMetis.

Tabela 5.30: Número total de arestas obtidas pelos algoritmos hMetis, LHp-3D e ZPlace (hMetis)

Benchs	**hMetis** 2	**LHp-3D** 2	**ZPlace** 2	**hMetis** 3	**LHp-3D** 3	**ZPlace** 3	**hMetis** 4	**LHp-3D** 4	**ZPlace** 4	**hMetis** 5	**LHp-3D** 5	**ZPlace** 5
ibm01	441	374	509	601	464	690	588	578	810	824	815	1.407
ibm02	547	396	417	714	614	765	873	841	1.562	1.131	1.042	2.540
ibm03	1.146	1.064	1.421	1.700	1.626	2.217	1.879	1.814	3.287	2.200	2.181	4.058
ibm04	738	735	1.524	1.474	1.386	2.275	2.009	1.918	2.594	2.229	2.170	3.475
ibm05	2.417	2.258	3.223	3.708	3.292	3.920	4.068	4.012	6.008	4.577	4.362	8.672
ibm06	1.061	1.056	1.227	1.522	1.376	2.422	1.871	1.646	3.393	1.897	1.871	4.139
ibm07	994	992	1.525	2.009	2.007	2.656	2.550	2.380	3.562	2.915	2.951	4.838
ibm08	1.324	1.298	1.482	2.410	2.280	3.299	2.738	2.654	5.204	3.207	3.154	6.616
ibm09	806	699	728	1.791	1.544	2.461	2.016	1.966	3.292	2.375	2.251	4.087
ibm10	1.771	1.490	1.696	2.719	2.114	3.246	3.080	2.874	5.230	3.929	3.277	6.233
ibm11	1.490	1190	1.468	2245	2026	2.667	2923	2708	4.403	3382	3151	5.045
ibm12	2.594	2293	2.806	3795	3314	5.099	5106	4570	6.625	5588	4930	8.074
ibm13	1.193	1042	1.338	1952	1687	2.870	2391	2272	4.002	2701	2534	4.497
Média	1.271	1.145	1.490	2.049	1.825	2.661	2.469	2.326	3.844	2.843	2.668	4.899

A Seção 5.5 discutiu o desbalanceamento dos pinos de I/O. Analisou-se qual a influência do aumento do desvio padrão do número de pinos entre as partições em função dos resultados das vias-3D. Percebeu-se que, à medida que o desvio padrão aumentava, o número de vias era reduzido (considerando-se o menor caminho lógico entre pares de I/O). Para os experimentos, abordaram-se dois níveis de desbalanceamento. O primeiro considerou, em média, o desvio padrão de 50 pinos (u=10), e o segundo, de 100 pinos (u=25). Pelo crescimento do desvio padrão, a tendência foi o deslocamento dos pinos para uma só partição, o que se assemelhou bastante ao que aconteceu com o algoritmo hMetis. Já a redução do número de vias ocorreu em todos os níveis de desbalanceamento e número de *tiers*, sendo o melhor resultado o alcançado com duas camadas: 3%, em média, para o desbalanceamento usando u=25 (100 pinos).

Constatou-se, contudo, que, apesar da redução do número de vias-3D, o circuito gerado tinha inúmeras conexões que cruzavam mais de duas *tiers* adjacentes. Como mencionado, esse tipo de conexão impõe inúmeras restrições ao projeto 3D. Percebeu-se, durante a pesquisa bibliográfica, que a meta-heurística *simulated annealing* era bastante utilizada em problemas de posicionamento local, gerando bons resultados. Com base nisso, a Seção 5.6 apresentou uma metodologia para a redução de conexões não-adjacentes por meio da modificação dessa meta-heurística. As funções de perturbação, o custo e a avaliação foram modificadas, e sua execução considerou a posição das *tiers*. Esse algoritmo foi chamado de LHr-3D. Este permite que as conexões localizadas entre camadas não-adjacentes migrem para as adjacentes sem aumentar o número total de vias-3D. Isso mostra que a função de custo criada equilibrou a

distribuição das vias, fazendo com que seus números máximo e total não aumentassem. A Figura 5.36 ilustra a comparação entre os algoritmos hMetis, LHp-3D, LHr-3D e ZPlace (hMetis), todas configuradas para a máxima redução de vias. Percebe-se que, em todos os casos, o algoritmo LHr-3D obteve um número de vias-3D menor do que os demais.

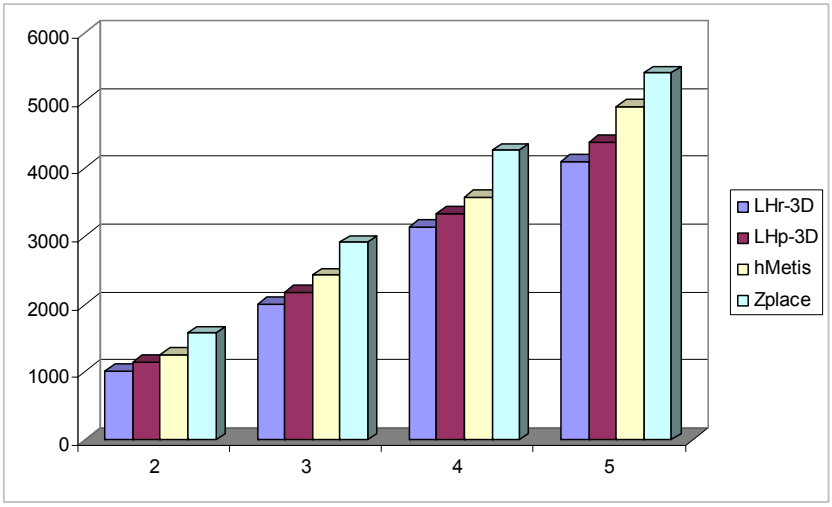


Figura 5.36: Comparação dos números de vias obtidos por ZPlace, hMetis, LHp-3D e LHr-3D.

A Tabela 5.31 apresenta um resumo das reduções dos números de vias-3D obtidas pelo LHr-3D em comparação com as dos demais algoritmos. Percebeu-se que a menor redução foi obtida pelo LHp-3D, o que mostra a qualidade da solução inicial apresentada na Seção 5.4. A diferença entre os resultados de LHr-3D e ZPlace (hMetis) foi a maior tanto na redução do número de vias-3D quanto no número máximo de vias entre duas *tiers* adjacentes. Pode-se verificar a representação em gráfico, na Figura 5.37, das máximas reduções obtidas desses números. A função de custo de ZPlace não proporcionou a distribuição equilibrada de vias entre as *tiers*, o que ocasionou a migração delas, na maioria das vezes, somente para um único par de camadas adjacentes.

Tabela 5.31: Percentual de redução do número total e máximo de vias-3D comparado com o algoritmo LHr-3D.

	Número de vias-3D				**Número Máximo de vias-3D**			
	2	3	4	5	2	3	4	5
LHp-3D	11%	23%	34%	11%	11%	8%	5%	7%
hMetis	19%	18%	12%	16%	19%	26%	11%	13%
ZPlace	35%	31%	26%	24%	35%	33%	32%	26%

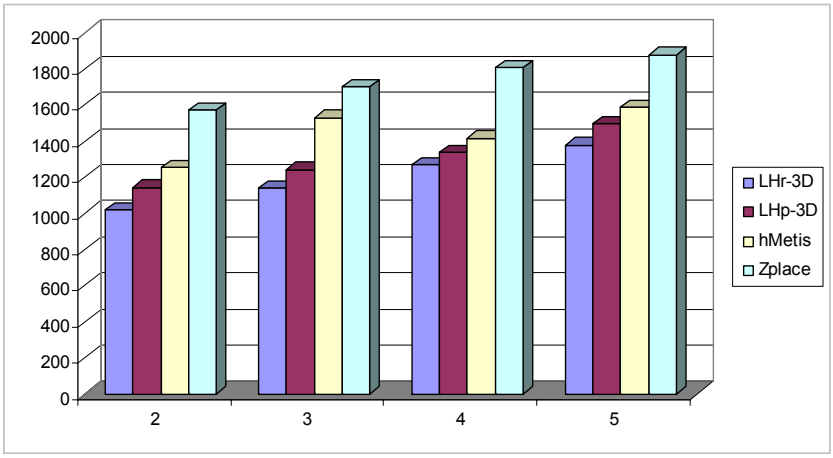


Figura 5.37: Máximo número de vias-3D cruzando *tiers* adjacentes.

Analisando-se o fluxo proposto neste trabalho, constatou-se que as etapas de otimização aqui realizadas contribuíram diretamente para a redução do número de vias-3D. Os resultados experimentais mostraram o ganho obtido pelas duas estratégias propostas. Em todas as etapas do fluxo a orientação dos pinos, espaços em branco e relação de aspecto foram mantidos. Para finalizar, a proposta desenvolvida gerou uma ferramenta que converte circuitos 2D em seus equivalentes 3D sem que as características originais sejam prejudicadas, assim como obtém a máxima redução de vias, consequentemente, da área do circuito.

6 CONCLUSÕES

Sabe-se que os circuitos em três dimensões (3D) surgem como um novo paradigma de projeto e uma nova forma para reduzir a potência, a área e as interconexões. Contudo, como indicado neste trabalho, seu mecanismo de comunicação intracamadas, chamado via-3D, impõe inúmeras limitações para o projeto. Levando-se em consideração esses aspectos, foi apresentada uma metodologia de particionamento direcionada para a redução do número de vias em circuitos VLSI 3D. O fluxo desenvolvido baseou-se em duas estratégias distintas: análise da estrutura interna do circuito e redução das conexões verticais não-adjacentes.

O princípio desta tese apresentou um levantamento bibliográfico envolvendo circuitos VLSI 3D. Buscou-se entender o funcionamento das estratégias de montagem, integração e comunicação utilizadas com essa tecnologia. Quanto à estratégia de montagem, percebeu-se que a mais usada atualmente trabalha com empilhamento de *wafers*, por permitir que se empilhe grande número de *chips* de uma só vez e por obter maior precisão de alinhamento. Seu uso resulta em erros de alinhamento inferiores a 1µm, uma vantagem em comparação com o da estratégia que empilha *chips* individualmente. Verificou-se também o surgimento da estratégia conhecida como empilhamento de transistores, que, diferente das demais, não trabalha com o empilhamento de *chips* pré-fabricados, mas sim com a deposição e remoção de materiais. Essa estratégia não causa problemas de alinhamento; contudo, seu procedimento ocasiona aquecimento dos materiais no momento do processo de fabricação, o que inviabiliza sua utilização atualmente.

Pelo estudo realizado, tornou-se possível perceber que são três as estratégias de integração: *face-to-face*, *face-to-back* e *back-to-back*. Verificou-se que as duas últimas necessitam que a área ativa seja perfurada para a passagem da via-3D. Levando-se em consideração as suas dimensões físicas, a grande quantidade de vias-3D pode aumentar bastante a área do circuito. A análise dos experimentos realizados neste trabalho demonstrou que o número total de vias aumenta proporcionalmente ao aumento do número das *tiers.* Em média, esse aumento é de 40% para cada nova camada. Já as vias que cruzam duas *tiers* adjacentes aumentam, em média, 10%. Além disso, nesse caso, as *tiers* com menor número de células e vias precisam ter a mesma área que a maior camada, a fim de que todas fiquem alinhadas. Nos experimentos com tecnologia de vias *bulk pitch* 50µm, constatou-se que somente em duas *tiers* a área de células era menor que a área de vias. Isso se devia ao pequeno número de vias que separavam os dois *chips*. Já no momento em que o número de *tiers* aumentou, como mencionado, o mesmo ocorreu com o número de vias, de modo que sua dimensão superou a área das células.

Percebeu-se, por parte dos fabricantes de semicondutores, que existe um fator que limita fisicamente a redução da área das vias, relacionado diretamente com as tecnologias atuais de empilhamento. Tal limitação segue a seguinte premissa: quanto menor for a área da via, maior a possibilidade de o encaixe desta entre as *tiers* dar errado. Essa é uma premissa válida e que reforça novamente a abordagem deste trabalho.

Pode-se combinar ainda diferentes estratégias de integração, contudo a Tezzaron e MITLL mencionam, em suas pesquisas, que o problema térmico limita o número de camadas empilhadas a 10 (já que *tiers* intermediárias formam focos de calor). Ainda assim, constatou-se que as questões térmicas são bastante abordadas pela literatura, sendo desenvolvidos diversos trabalhos voltados a *floorplaning*, posicionamento e inserção de vias térmicas.

Analisou-se também as metodologias de projeto. Concluiu-se que um *chip* 3D pode ser projetado de três maneiras: posicionando-se elementos heterogêneos em diferentes camadas, particionando-se os blocos entre estas, ou dividindo-se um bloco de lógica aleatória em várias camadas. Percebeu-se que muitos estudos dedicam-se à primeira estratégia, pois entendem que a idéia de acomodar elementos heterogêneos em cada uma das camadas, evita problemas de ruídos e pode ser adaptada a sistemas de diferentes naturezas. Contudo, ela foi analisada, neste trabalho, de acordo com outros critérios. Considerou-se o nível de granularidade do projeto, por entender que, quanto menor for o elemento a ser tratado, maior será seu nível de integração (acoplamento). Essa análise partiu do fato de um bloco de lógica aleatória projetado, por exemplo, em quatro *tiers* ser muito similar a um *chip* 3D, exceto por seus respectivos tamanhos. Portanto, tratando-se de elementos menores, pode-se utilizar as mesmas metodologias para ambas as estruturas. Nesse caso, a lógica aleatória citada como exemplo pode conter um conjunto de *cores*, em vez de um conjunto de células. A questão a ser observada é que, independente da abordagem e da granularidade usadas, a solução proposta para se resolver os problemas de elementos menores, como as células, pode atuar perfeitamente em situações em que os elementos tiverem dimensões maiores.

Ainda nesse contexto, buscou-se compreender como são tratados os pinos e *pads* de I/O. Antes dessa análise, essas estratégias foram diferenciadas. Considerou-se *pads* como sendo os elementos que realizam a comunicação entre a parte interna do *chip* com os seus componentes externos. Já os pinos de I/O, como os elementos de comunicação internos ao *chip*, como os pinos de um bloco *ip* ou de lógica aleatória. Percebeu-se que os *pads*, em projetos 3D, são bem maiores que os pinos. Portanto, em circuitos com grande integração e performance, os *pads* de I/O têm de ser projetados e acomodados em *tiers* exclusivas. Verificou-se também que a definição de sua posição depende exclusivamente da decisão do projeto, o qual pode fixá-los na parte superior, na inferior ou até mesmo em ambas as extremidades do *chip* (não está se levando em consideração aqui o empilhamento de *chips* e as *wire bounds*, que permitem a localização dos *pads* entre as camadas, por entender que essa tecnologia de montagem é de baixa integração e performance). A Tezzaron, por exemplo, projeta os *pads* nas partes superior e inferior, pois entende que esse procedimento é capaz de evitar que vias cruzem toda a extensão do *chip*, caso os *pads* estejam localizados em extremidades opostas. Esse argumento é muito interessante e torna-se mais forte à medida que o número de *tiers* aumenta. Já os pinos de I/O podem ser projetados da mesma forma que os *pads*, e sua posição pode ser também distribuída entre as *tiers*. Muitos trabalhos exploraram a divisão de blocos de lógica aleatória utilizando a localização dos pinos somente em uma das extremidades. Já neste trabalho, entende-se que eles serem distribuídos entre as *tiers* auxilia a etapa de

particionamento. De fato, os experimentos realizados distribuíram criteriosamente os pinos de I/O pelas *tiers* e obtiveram, com isso, redução do corte entre as partições, diminuição do número de vias-3D e, consequentemente, da área do circuito.

Pelas análises feitas durante a preparação deste trabalho, percebeu-se que a literatura pouco menciona como os algoritmos tratam a redução do número de vias em projetos 3D. Constatou-se que a maioria dos trabalhos aborda a redução nas etapas de particionamento e posicionamento, contudo não de forma detalhada. Em vista disso, realizou-se um levantamento bibliográfico dos algoritmos de particionamento e posicionamento com o objetivo de se entender sua aplicação em projetos 3D.

Percebeu-se que há anos estudam-se algoritmos de particionamento, de modo que existem muitos deles com resultados de *min-cut* de boa qualidade. Atualmente, algoritmos tradicionais de migração de grupos, como Kernighan-Lin e Fiduccia-Matheyses, ainda são bastante utilizados. Ambos baseiam-se na redução do número de cortes entre as partições, mas Fiduccia-Mattheyses apresenta evoluções em relação a Kernighan-Lin, por realizar a migração simples de uma célula e apresentar uma complexidade de tempo inferior a $O(n^2)$, ao passo que a de Kernighan-Lin é de $O(n^3)$. Já a modificação mais importante, porém, é a do corte das redes. Para Fiduccia-Mattheyses, não interessa o número de células localizadas em cada partição, mas se esta é atravessada pela rede.

Em seguida, observou-se que, à medida que o número de elementos aumenta, mais complexa é a resolução do problema. Por isso, técnicas de união de vértices foram desenvolvidas a fim de conter, em um só elemento, inúmeros outros de menor tamanho (formando um *cluster*). Assim, Fiduccia-Mattheyses e Kernighan-Lin são atualmente melhor aproveitados.

Já na Seção 4.5 discutiu-se a importância do particionamento no projeto 3D. Verificou-se que os algoritmos comumente utilizados são adaptações dos tradicionais, sendo sua proposta focada especificamente em redução do número de cortes entre as partições, balanceamento da área e número de elementos. Percebeu-se, entretanto, que as soluções voltadas para a redução de vias-3D estão intimamente relacionadas à qualidade do resultado dos particionadores. Dessa constatação, surgiu a ideia inicial desta tese: melhorar a qualidade do corte entre as partições sem prejudicar a performance e balanceamentos. Porém, como tornar isso possível, tendo-se em vista que há anos estuda-se esse mesmo tipo de problema e que, desde o surgimento do algoritmo hMetis em 1997, nenhuma outra ferramenta aplicada a circuitos obteve melhores resultados? Este argumento foi levado em consideração e analisado. Contudo, pelo estudo da estrutura dos algoritmos de particionamento, em especial do hMetis, percebeu-se que suas execuções não consideram nenhum elemento interno do circuito. A partir disso, procurou-se entender quais são as informações internas que poderiam ser pertinentes para auxiliar as ferramentas atuais a obterem maior qualidade nos resultados de corte. Com base em observações do comportamento de alguns posicionadores direcionados a forças, percebeu-se que, nesses modelos, as células que compõem uma *netlist* são atraídas pelos pinos de I/O que estão posicionados na borda do circuito. Sob a ótica 3D, percebeu-se que os pinos em blocos de lógica aleatória têm de ficar posicionados na borda de cada camada, a fim de formarem um cubo. Contudo, a questão principal é que este só é formado após o particionamento da *netlist* (como indicado neste trabalho, partições e *tiers*, em certo momento, são coisas distintas).

Foi observado então que inúmeros vértices ficavam posicionados nos arredores dos pinos de I/O, em geral compartilhando a mesma rede. Surgiu então a ideia de encontrar o menor caminho lógico entre os pares de pinos para que aqueles que tivessem maior conectividade com suas células vizinhas fossem agrupados em *tiers* iguais. Os pinos foram particionados e distribuídos de forma balanceada entre as partições e serviram de guias no momento do particionamento das células. Utilizou-se o algoritmo hMetis e o Alternate Pins para validar o experimento. Os resultados foram apresentados as Seções 5.4 e 5.5 e comprovaram que, aplicada a circuitos, a ideia de fixar os pinos de I/O por meio da informação do menor caminho lógico melhora a qualidade do corte.

Como a redução do número de vias está diretamente relacionada à qualidade do corte obtida no particionamento, utilizou-se, primeiramente, o algoritmo de *simulated annealing* para encontrar o arranjo de partições que gerasse o menor número total de vias-3D. Após observações, optou-se pelo uso de um algoritmo de busca exaustiva, já que o número de *tiers* em um projeto 3D é relativamente pequeno (atualmente, menor que 10). Os experimentos da Seção 5.4.3 mostraram que, em média, a qualidade do corte reduziu em 9%, 10%, 6% e 11% o número de vias-3D para duas, três, quatro e cinco *tiers* respectivamente comparado ao hMetis.

Nesse experimento, o balanceamento do número de pinos de I/O entre as partições foi bastante rígido e despertou o interesse em observar o comportamento do corte à medida que seu balanceamento fosse alterado. Percebeu-se então que, conforme o desvio padrão dos pinos entre as partições aumentava, o número de vias também era reduzido, em média, 2%. Contudo, a tendência verificada foi a migração por parte dos pinos de I/O para uma única partição ao passo que seu balanceamento começava a relaxar. Essa característica se assemelhou bastante aos resultados do algoritmo hMetis.

Considerando-se os aspectos relatados, percebeu-se que o estudo da estrutura interna do circuito contribuiu diretamente para melhorar a qualidade do corte do hMetis entre partições. É importante ressaltar que os experimentos somente foram testados em circuitos (*netlists*). Contudo, pretende-se estender a estratégia apresentada neste trabalho para que seja utilizada em qualquer tipo de hipergrafo. Sabe-se que, em hipergrafos normais, não existe distinção entre pinos de I/O e células: todos os elementos são conhecidos como nodos. Nesse caso, faz-se necessário encontrar, a partir de sua estrutura, quais destes têm as mesmas características dos pinos de I/O em circuitos.

O fluxo de particionamento proposto por meio da análise da estrutura interna do circuito foi dividido em dois algoritmos: LHp-3D e LHu-3D, como apresentados nas Seções 5.4 e 5.5. O primeiro trata os pinos de I/O com um balanceamento rígido entre as partições, e o LHu-3D possibilita a modificação do balanceamento do número de pinos.

Enquanto era analisada a redução do corte entre as partições e, consequentemente, do número de vias-3D e da área, verificou-se outro problema, o das vias-3D longas. Identificou-se que tal problema ocorria no momento em que o empilhamento das partições era realizado. Estava claro que não bastava somente melhorar a qualidade do corte entre as partições para se reduzir o número das vias. Era necessário diminuir também o número daquelas que cruzassem as *tiers* não-adjacentes. Contudo, esse novo procedimento não poderia aumentar o número total de vias nem o número máximo destas entre duas camadas adjacentes (pois isso aumentaria a área do circuito). Teriam de ser encontrados um equilíbrio e uma convergência tanto da redução das vias não-adjacentes quanto das que cruzassem duas *tiers* adjacentes.

Concluiu-se então que a maneira mais eficiente de obter uma melhor otimização do circuito era manter as *tiers* empilhadas. Seria necessário criar um algoritmo que permitisse várias formas de perturbação e que oferecesse função de custo e avaliação flexíveis. Durante o levantamento bibliográfico, percebeu-se que a meta-heurística conhecida como *simulated annealing* era frequentemente utilizada em problemas de refinamento e legalização de células em posicionamentos locais, gerando resultados muito bons. Por esse motivo, estudou-se esse algoritmo, e, de acordo com essa análise, algumas modificações foram propostas para que ele atuasse no fluxo de redução de conexões não-adjacentes.

A primeira modificação foi inserir, no procedimento de perturbação, vários tipos de combinações de trocas entre pinos e células. A segunda foi fazer a função de custo considerar o número de vias e o desbalanceamento da área de células e do número de pinos de I/O. Para manter todos esses elementos heterogêneos na mesma unidade, eles foram normalizados, sendo todos sempre divididos pelo seu valor inicial. A função de custo penalizava o surgimento de novas conexões não-adjacentes, com a elevação do número total de vias ao quadrado. Além disso, foi possível atribuir pesos a fim de priorizar as reduções de vias, área ou número de pinos. Optou-se por manter o balanceamento de área de células e o número de pinos rígidos. Para finalizar, em sua função de avaliação, o algoritmo proposto aceitava somente soluções melhores do que as anteriores, o que não comprometia seu tempo de execução, pois se trata de um elemento que penaliza apenas as conexões longas e as distribui de forma equilibrada entre as *tiers* adjacentes.

Verificou-se, nos experimentos realizados, que as vias não-adjacentes migraram de forma equilibrada para as *tiers* adjacentes, o que não ocasionou o aumento do número máximo de vias; pelo contrário, ele diminuiu. Com essa migração, o resultado contribuiu também para a redução do número total de vias em comparação com o hMetis, em média, 19%, 18%, 12% e 16% para duas, três, quatro e cinco *tiers*, respectivamente. Quando comparada com o algoritmo direcionado a forças, ZPlace, essa redução foi, em média, de 35%, 31%, 28 e 24%. Essa etapa do fluxo foi chamada de LHr-3D. O comportamento interessante observado pelos experimentos foi de que, quando o algoritmo proposto forçou a migração das vias de *tiers* não-adjacentes para as adjacentes, a qualidade do corte entre as partições piorou, como observado na Seção 0. Isso aconteceu tanto porque as partições foram organizadas em linha quanto por ser o objetivo fazer com que o número de vias entre as *tiers* adjacentes aumentasse. Por isso, para melhorar esse resultado, a qualidade do corte foi comprometida. Isso comprova que, durante a fase de particionamento, quando não se considera a localização das partições, o empilhamento das partições torna as arestas em vias longas.

Abaixo, são ressaltadas as contribuições desta pesquisa, e são indicadas análises e trabalhos futuros.

- **Levantamento bibliográfico**: Buscou-se, por pesquisas na literatura, a compreensão do quê são os circuitos VLSI 3D. Nesse sentido, um vasto número de assuntos foram abordados em relação as estratégias de montagem, integração, comunicação, metodologias de projeto, grupos de pesquisa, projeto de pinos e *pads* de I/O, bem como as vantagens e desvantagens desse novo tipo de circuito. Foram abordados também os algoritmos que compõem as etapas de particionamento e posicionamento de circuitos. Discutiram-se estruturas de algoritmos tradicionais, suas classificações e evoluções, assim como seus usos em projetos 3D.

- **Desenvolvimento de um conjunto de circuitos *bechmarks* 3D:** Todo o conjunto de circuitos *benchmarks* ISPD 2004 foi convertido para circuitos 3D com duas, três, quatro e cinco *tiers*, vindo a complementar o conjunto de *benchmarks* disponíveis no meio acadêmico. A partir desses novos circuitos, outras estratégias e ferramentas podem utilizar as informações de área das *tiers*, número e localização das células, número de camadas, área de células e sua quantidade em cada *tier*, espaços em branco, números efetivo e máximo de vias, localização e orientação dos pinos.
- **Fluxo completo de migração de circuitos 2D para 3D:** Dado um circuito qualquer com pinos de I/O e a informação do número requerido de *tiers*, o fluxo, com base nas características do circuito, pode transformá-lo em uma estrutura 3D otimizada. Criou-se, então, um fluxo automático que converte estruturas de circuitos 2D em seu corresponde em 3D, para que o número total de vias e a área desse novo circuito seja a menor possível.
- **Nova estratégia de particionamento:** A pesquisa realizada mostrou que, por meio de informações contidas na estrutura interna do circuito, é possível melhorar as soluções dos particionadores atuais, em especial as do hMetis. Demonstrou-se que a informação do menor caminho lógico entre pares de pinos de I/O permite que as células com maior conectividade entre si sejam guiadas em conjunto para as mesmas partições. Essa estratégia torna possível melhorar a qualidade de corte, sem que se prejudique o desempenho do algoritmo.
- **Redução de vias-3D longas:** Aplicando-se somente o particionamento de hipergrafos planares, não se tem a informação da localização das partições. Com isso, o resultado é, muitas vezes, um grafo completo, contendo inúmeras arestas entre todas as partições. Essa estratégia aplicada em circuitos 3D, devido à etapa de empilhamento, tem consequência direta no surgimento de vias-3D longas. Neste trabalho desenvolveu-se uma estratégia para a redução das vias não-adjacentes por meio da modificação de uma meta-heurística baseada em *simulated annealing*. Com o uso dessa estratégia, as vias migraram de forma equilibrada entre *tiers* adjacentes, contribuindo para a redução de seu número total.
- **Redução de área:** O fluxo desenvolvido, além de manter a área de células equilibrada entre as diferentes *tiers*, reduz o número máximo de vias que cruzam duas camadas adjacentes. Considerando-se a dimensão da área das vias e sabendo-se que, no momento em que ela cruza a camada ativa, está ocupando o lugar de inúmeros transistores, conclui-se que essa área aumenta proporcionalmente ao aumento de seu número.
- **Redução máxima de vias:** Essa contribuição tem efeito direto na redução da área do circuito. A estratégia de integração não foi considerada como referência. Buscou-se priorizar a redução total do número de vias e sua contribuição para a redução do número máximo destas entre *tiers* adjacentes.
- **Análise Experimental:** O trabalho permitiu o uso e análise de diferentes algoritmos que seguem a mesma formulação do problema.

Abaixo são destacadas algumas aplicações do fluxo projeto:

- **Utilização em outras metodologias de projeto 3D:** Um bloco de lógica aleatória dividido entre diferentes camadas se assemelha bastante a um circuito 3D em outros níveis de projeto. Por isso, a metodologia desenvolvida neste trabalho, aplicada em granularidade baixa pode ser estendida para atuar em granularidades maiores. Sua estrutura algorítmica e seu fluxo não se modificariam. Entretanto, este trabalho priorizou sua aplicação em blocos com granularidades menores, como o das lógicas aleatórias, permitindo sua inserção em trabalhos futuros.
- **Vias térmicas:** A estratégia de inserção de vias térmicas atravessando todo o circuito é, atualmente, uma das soluções mais adotadas para tratar da dissipação de calor em circuitos 3D. Contudo, a inserção de novas vias no circuito tem conseqüência direta em sua área. O algoritmo apresentado neste trabalho permite a máxima redução de vias-3D, abrindo espaço para que vias térmicas sejam inseridas sem um prejuízo maior para a área do circuito.
- **Posicionamento:** A geração de *benchmarks* 3D e as informações contidas em sua estrutura tornam possível o posicionamento individual das células em cada uma das *tiers*. Assim, abre-se novas possibilidades de estudos em torno da legalização e posicionamento considerando as células localizadas no mesmo plano e em camadas adjacentes.

Os itens abaixo explicitam possibilidades para futuros trabalhos que dêem seguimento a este estudo:

- **Aplicar a nova estratégia em hipergrafos genéricos:** A nova estratégia, que se baseou no menor caminho lógico, foi testada somente em circuitos. Pelas informações obtidas neste trabalho, acredita-se que uma nova abordagem aplicada a hipergrafos genéricos poderia também melhorar a qualidade dos resultados. Diferente dos circuitos, que são compostos por células, pinos de I/O e hiper-redes, os hipergrafos se constituem somente de nodos e hiper-arestas. Portanto, seguindo-se a estratégia aplicada aos circuitos, seria necessário encontrar nodos que tenham informações de conectividade similares às dos pinos de I/O.
- **Clusterização:** Um estudo mais aprofundado sobre a clusterização de células pode ser realizado e aplicado tanto nos algoritmos LHp-3D e LHu-3D quanto no LHr-3D. Com isso, poderia se avaliar a qualidade dos resultados unindo-se os vértices, antes da migração entre as partições ou *tiers*.
- **Considerar tipos diferentes de vias e integrações:** A informação do tipo de via utilizada pode ser muito útil para o particionamento. Pode-se, além disso, aplicar diferentes estratégias de integração, *face-to-face, face-to-back* e *back-to-back*. Essa nova abordagem pode ser ampliada a fim de encontrar o melhor arranjo de estratégias de integração em combinação com os diferentes tipos de vias.
- **Caminhos críticos:** Apesar deste trabalho ter iniciado experimentos envolvendo caminhos críticos, não foi gerado nenhum dado relevante para incorporar ao texto. Contudo, sabe-se que encontrar o caminho crítico sem ter estimativas do tamanho da conexão é, muitas vezes, impreciso. Entende-se que isso pode ser uma etapa que anteceda ao particionamento, podendo ser

modificada durante a etapa de posicionamento. Trata-se de uma abordagem que pode ser somada ao trabalho aqui apresentado.

REFERÊNCIAS

3D ICS INDUSTRY SUMMARY. Disponível em: <http://www.tezzaron.com>. Acesso em Janeiro de 2009.

ABABEI, C.; FENG, Y.; GOPLEN, B.; MOGAL, H.; ZHANG, T.; BAZARGAN, K.; SAPATNEKAR, S. Placement and Routing in 3D Integrated Circuits. **Design and Test of Computers**, [S.l.], p.520–531, Nov-Dec 2005.

AGNIHOTRI, A. R.; ONO, S.; MADDEN, P. H. Recursive bisection placement: feng shui 5.0 implementation details. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.230–232.

ALPERT, C. J., KHANG, A. Recent directions in netlist partitioning. Integration, the VLSI Journal, 19(1-2):1–81, 1995.

ALPERT, C. J.; CHAN, T.; HUANG, D. J.-H.; MARKOV, I.; YAN, K. Quadratic placement revisited. In: DAC '97: PROCEEDINGS OF THE 34TH ANNUAL CONFERENCE ON DESIGN AUTOMATION, 1997, New York, NY, USA. **Anais. . .** ACM Press, 1997. p.752–757.

ALPERT, C. J.; HU, T. C.; HUANG, J. H.; KAHNG, A. B.; KARGER, D. Prim-Dijkstra Tradeoffs for Improved Performance-Driven Routing Tree Design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.14, p.890–896, 1995.

AL-SARAWI, S. F., D. Abbott, P. D. Franzon. A Review of 3-D Packaging Tecnology. IEEE Transaction Components, Pkg. & Manuf. Technol. B21, 1 (1998).

BALAKRISHNAN, K.; NANDA, V.; EASWAR, S.; LIM, S. K. Wire congestion and thermal aware 3D global placement. In: ASP-DAC '05: PROCEEDINGS OF THE 2005 CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.1131–1134.

BANERJEE, K.; SOURI, S.; KAPUR, P.; SARASWAT, K. 3D-ICs: A novel chip design for improving deep submicrometer interconnect performance and systems on-chip integration. **Proceedings of IEEE**, New York, v.89, p.602–633, 2001.

BERNSTEIN, K; ANDRY, J.; CANN, J; EMMA, P; GREENBERG, D.; HAENSCH, W.; IGNATOWSKI, M.; KOESTER, S.; MARGELEIN, J.; PURI. R; YOUNG, A.

Interconnects in the Third Dimension: Design Challenges for 3D ICs. In: Design Automation Conference, 2007. DAC'07. 44th ACM/IEEE. 2007 Page(s):562 – 567

BERSTEIN, K. Introduction to 3D Integration; tutorial no International Solid-State Circuits Conference (ISSCC), San Francisco, CA, 2006.

BEYNE, Eric; 3D Interconnection and Packaging; 2nd Electronics System-Integration Technology Conference, Londres, 2006.

BRENNER, U.; VYGEN, J. Legalizing a placement with minimum total movement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, New York, v.23, 2004.

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

CALDWELL, A.; KAHNG, A. B.; KENNINGS, A.; MARKOV, I., Hypergraph Partitioning for VLSI CAD: Methodology for Heuristic Development Experimentation and Reporting. In: DESIGN AUTOMATION CONFERENCE , DAC, 36., 1999, New Orleans, Proceedings… New York: ACM, 1999.
CAMPARDO, G.; RIPAMONTI, G.; MICHELONI, R; Scanning the Issue: #-D Integration Technologies, In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 5-8, 2009.

CHANG, C.; CONG, J. Multilevel Global Placement With Congestion Control. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.22, April 2003.

CHERKAUER, B. S.; FRIEDMAN, E. G.; A Unified Design Methodology for CMOS tapered Buffers, IEEE Transaction Very Large Scale (VLSI) System, vol. 3, pp. 00-111, 1995.

CONG, J.; KAHNG, A. B.; ROBINS, G.; SARRAFZADEH, M.;WONG, C. K. Provably Good Performance Driven Routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.11, p.739–752, 1992.

CONG, J.; KOH, C.-K.; MADDEN, P. Interconnect layout optimization under higher order RLC model for MCM designs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.20, p.1455–1463, 2001.

CONG, J.; LEUNG, K.-S.; ZHOU, D. Performance-driven interconnect design based on distributed RC delay model. In: DAC '93: PROCEEDINGS OF THE 30TH INTERNATIONAL CONFERENCE ON DESIGN AUTOMATION, 1993, New York, NY, USA. **Anais. . .** ACM Press, 1993. p.606–611.

CONG, J.; WEI, J.; ZHANG, Y. A Thermal-Driven Floorplanning Algorithm for 3D ICs. In: ICCAD 2004: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2004. **Anais. . .** [S.l.: s.n.], 2004.

CONG, J.; ZHANG, Y. Thermal via planning for 3-D ICs. In: ICCAD '05: PROCEEDINGS OF THE 2005 IEEE/ACM INTERNATIONAL CONFERENCE ON

COMPUTERAIDED DESIGN, 2005,Washington, DC, USA. **Anais. . .** IEEE Computer Society, 2005. p.745–752.

CUBIC WAFER TECHNOLOGY: HOMEPAGE. Disponível via WWW em < http://www. xanoptix.com/ >. Acesso em Fevereiro de 2009.

CULURCIELLO, E.; ANDREOU, A. G.; Capacitive inter-chip data and power transfer for 3-D VLSI, In: IEEE Transactions Circuits and Systems, II, Exp. Brief, vol. 53, pp. 1348-1352, 2006.

DAS, S.; CHANDRAKASAN, A.; REIF, R. Calibration of Rent's rule models for threedimensional integrated circuits. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, New York, v.12, p.359–366, 2004 (a).

DAS, S.; CHANDRAKASAN, A.; REIF, R. Design tools for 3-D integrated circuits. In: ASPDAC: PROCEEDINGS OF THE 2003 CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, 2003, New York, NY, USA. **Anais. . .** ACM Press, p.53– 56, 2003.

DAS, S.; CHANDRAKASAN, A.; REIF, R. Timing, energy, and thermal performance of three-dimensional integrated circuits. In: GLSVLSI '04: PROCEEDINGS OF THE 14TH ACM GREAT LAKES SYMPOSIUM ON VLSI, 2004, New York, NY, USA. **Anais. . .**ACM Press, p.338–343, 2004 (b).

DAS, S.; FAN, A.; CHEN, K.-N.; TAN, C. S.; CHECKA, N.; REIF, R. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In: ISPD '04: PROCEEDINGS OF THE 2004 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2004, New York, NY, USA. **Anais. . .** ACM Press, 2004. p.108–115.

DAVIS, J.A., VENKATESAN, R. KALOYEROS, A. M. Beylansky, S. J. SOURI, K. BANERJEE, K. C. SARASWAT, A. RAHAMN, R. REIF, J. D. MEINDL. Interconnect Limits on Gigascale Integration (GSI) in the 21st Century. Proceedings IEEE 89, 305 (2001).

DAVIS, W.; WILSON, J.; MICK, S.; XU, J.; HUA, H.; MINEO, C.; SULE, A.; STEER, M.; FRANZON, P. Demystifying 3D ICs: the pros and cons of going vertical. **Design and Test of Computers**, [S.l.], p.498–510, Nov-Dec 2005.

DENG, Y.; MALY, W. 2.5-Dimensional VLSI System Integration. **IEEE Transactions on Very Large Integration (VLSI) Systems**, New York, v.13, p.668, June 2005.

DENG, Y.; MALY, W. P. Interconnect characteristics of 2.5-D system integration scheme. In: ISPD '01: PROCEEDINGS OF THE 2001 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2001, New York, NY, USA. **Anais. . .** ACM Press, 2001. p.171–175.

DIJKSTRA, E. W.: A note on two problems in connexion with graphs. In: Numerische Mathematik. 1 (1959), S. 269–271

EISENMANN, H.; JOHANNES, F. M. Generic global placement and floorplanning. In: DAC ’98: PROCEEDINGS OF THE 35TH ANNUAL CONFERENCE ON DESIGN AUTOMATION, 1998, New York, NY, USA. **Anais. . .** ACM Press, 1998. p.269–274.

ELECTRONIC DESIGN AUTOMATION CONSORTIUM. 2006, EDA Consortium History & Milestones 1996. “http://www.edac.org/about_background.jsp”.

ELMORE, W. C. The Transient Response of Damped Linear Network with Particular Regard toWideband Amplifiers. **Journal of Applied Physics**, [S.l.], v.19, p.55–63, 1948.

FEO, T.A.; RESENDE, M. G. C., Greedy randomized adaptive search procedures. **Jounal of Global Optimization**, 6:109–133, 1995.

FIDUCCIA, C. M., MATTHEYSES. A Linear Time Heuristic for improving network partitions. In. Proceedings 19th IEEE Desing Automation Conference. Pages 175-181, 1982.

FLYNN, P. C.; KIM, Y. C.; YOON, I. S.; **3-D Stacked Package Technology and Trends**; In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 31-41, 2009.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**. [S.l.]: W. H. Freeman, 1979.

GAREY, M. R.; JOHNSON, D. S. The Rectilinear Steiner Tree is NP-Complete. **SIAM Jour. on App. Math**, [S.l.], v.23, p.826–834, 1977.

GLOVER, F.; LAGUNA M., Tabu Search, Kluwer Academic Publishers, Boston, 1997.

GOLDBERG M. K., BURSTEIN, M. Heuristic improvement technique for bisection of VLSI networks. **In Proceedings of the IEEE International Conference on Computer Design**, pages 122--125, Port Chester, NY, 1983. 31

GOLDBERG, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning. EUA: Addison-Wesley, 1989.

GOPLEN, B.; SAPATNEKAR, S. Efficient Thermal Placement of Standard Cells in 3D ICs using a Force Directed Approach. In: ICCAD ’03: PROCEEDINGS OF THE 2003 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2003, Washington, DC, USA. **Anais. . .** IEEE Computer Society, 2003. p.86.

GOPLEN, B.; SAPATNEKAR, S. Thermal via placement in 3D ICs. In: ISPD ’05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.167–174.

GUPTA, S.; HILBERT, M.; HONG, S.; PATTI, R. **Techniques for Producing 3D ICs with High-Density Interconnect**. Available at: <http://www.tezzaron.com/>. Access on: Aug. 2005.

HALL, K. M. An r-dimensional quadratic placement algorithm. **Management Science**, [S.l.], v.17, p.219–229, 1970.

HAN, E., KARYPIS, G., KUMAR, V., MOBASTER, B. Clustering based on association rule hypergraphs. **In Proc. of Workshop on Research Issues on Data Mining and Knowledge Discovery**, 1997.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A Formal Basis to the Heuristic Determination of Minimum Cost Path. **IEEE Transaction on Systems, Science and Cybernetics**, New York, p. 100-1007, 1968.

HEALY, M.; VITTES, M.; EKPANYAPONG, M.; BALLAPURAM, C. S.; LIM, S. K.; LEE, H.-H.; LOH, G. H. Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.26, p.38–52, 2007.

HENDRICKSON, B., LELAND, R. An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations. SIAM J. Sci. Stat. Comput., 16(2):452-469, 1995.

HENTSCHKE, R. et al. 3D-Vias Aware Quadratic Placement for 3D VLSI Circuits. In: IEEE Computer Society Anual Symposium on VLSI, ISVLSI, 2007, Porto Alegre, RS, Brazil. Proceedings. . . Los Alamitos: IEEE Computer Society, 2007. p.67–72.

HENTSCHKE, R. F.; OLIVEIRA JOHANN, M. de; REIS, R. A Study on the Performance of Fast Initial Placement Algorithms. In: VLSI-SOC 2003: IFIP INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, 2003. **Anais. . .** [S.l.: s.n.], 2003.

HENTSCHKE, R.; FLACH, G.; PINTO, F.; REIS, R. Quadratic placement for 3d circuits using z-cell shifting, 3d iterative refinement and simulated annealing. In: SBCCI ’06: PROCEEDINGS OF THE 19TH ANNUAL SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.220–225.

HENTSCHKE, R.; Algorithms forWire Length Improvement of VLSI CircuitsWith Concern to Critical. 2007. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

HENTSCHKE, R; Algoritmos para o Posicionamento de Células em Circuitos VLSI. 2002. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

HMETIS HOMEPAGE, Disponível via WWW em < http://glaros.dtc.umn.edu/gkh ome /metis/hmetis/overview>. Acesso em Maio de 2009.

HU, B.; ZENG, Y.; MAREK-SADOWSKA, M. mFAR: fixed-points-addition-based vlsi placement algorithm. In: ISPD ’05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.239–241.

HUA, H.; MINEO, C.; SCHOENFLIESS, K.; SULE, A.; MELAMED, S.; JENKAL, R.; DAVIS, W. R. Exploring compromises among timing, power and temperature in threedimensional integrated circuits. In: DAC ’06: PROCEEDINGS OF THE 43RD ANNUAL CONFERENCE ON DESIGN AUTOMATION, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.997–1002.

HWANG, C.; PEDRAM, M. Timing-driven placement based on monotone cell ordering constraints. In: ASP-DAC ’06: PROCEEDINGS OF THE 2006 CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.201–206.

ISPD04 - IBM Standard Cell Benckmarks with Pads. Disponível via WWW em:<http://www.public.iastate.edu/_nataraj/ISPD04_Bench.html>. Acesso em: Março de 2008.

IEONG, M; K. W. Guarini, V.Chan, K. Bernstein, R. Joshi, J. Kendzierski, W. Haensch. Three Dimensional CMOS Devices and Integrated Circuits. Proceedings of the IEEE Custom Integrated Circuits Conference, 2003, pp. 207-213.

JAIN, V. K.; BHANJA, S.; CHAPMAN, G. H.; DODDANNAGARI, L.; A Highly Reconfigurable Computing Array: DSP Plane of a 3D Heterogeneous SoC. In: Proceedings IEEE International System on Chip Conference, pp. 243-246, 2005.

JANG, D. M., et al., Development and evaluation of 3-D SiP with vertically interconnected through silicon vias (TSV), In: Proceedings of the IEEE Topical Meeting Electr. Performance, pp. 847-850, 2007.

JOHANN, M. **Novos Algoritmos para Roteamento de Circuitos Integrados**. 2001. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande

JOHANN, M. O. . Algoritmos para Síntese Física. In: Ricardo Augusto da Luz Reis. (Org.). Concepção de Circuitos Integrados. 1 ed. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2000, p. 139-162.

JOHANN, M.; CALDWELL, A.; KAHNG, A.; REIS, R. A New Bidirectional Heuristic Shortest Path Search Algorithm. In: INTERNATIONAL ICSC CONGRESS ON ARTIFICIAL INTELLIGENCE AND APPLICATIONS, 2000. **Proceedings. . .** [S.l.: s.n.], 2000.

JOYNER, J. W. et al., Impacts of Three-Dimensional Architectures on interconnects in gigascale integration, IEEE Transaction Very Large Scale (VLSI) System, vol. 9, pp. 992-928, 2001.

JOYNER, J. W.; MEINDL, J. D., Opportunities for reduced power distribution using three-dimensional integration, In: Proceeding of the IEEE Int. Interconnect Technology Confererence, pp. 148-150, 2002.

KAHNG, A. B.; REDA, S.; WANG, Q. APlace: a general analytic placement framework. In: ISPD ’05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.233– 235.

KAHNG, A. B.; WANG, Q. An analytic placer for mixed-size placement and timingdriven placement. In: ICCAD ’04: PROCEEDINGS OF THE 2004 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, 2004. **Anais. . .**[S.l.: s.n.], 2004. p.565–572.

KARYPIS, G., AGGARWAL, R., KUMAR, V., SHEKHAR, S.. Multilevel hypergraph partitioning: Application in vlsi domain. In Proceedings of the Design and Automation Conference, 1997.

KARYPIS, G., KUMAR, V. Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, 48(1):96–129, 1998 (a).

KARYPIS, G., KUMAR, V. A fast and highly quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 1998 (b).

KARYPIS, G.; AGGARWAL, R.; KUMAR, V.; SHEKHAR, S. Multilevel Hypergraph Partitioning: applications in vlsi domain. **IEEE Transactions on Very Large Integration (VLSI) Systems**, New York, v.7, p.69–79.

KAYA, I.; OLBRICH, M.; BARKE, E.; 3-D Placement Considering Vertical Interconnects; SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip], 2003.

KAYA, I.; SALEWSKI, S.; OLBRICH, M.; BARKE, E. Wirelength Reduction Using 3- D Physical Design. In: INTEGRATED CIRCUIT AND SYSTEM DESIGN – POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION; PROCEEDINGS OF 14TH INTERNATIONAL WORKSHOP, PATMOS 2004, 2004. **Anais. . .** [S.l.: s.n.], 2004.

KERNIGHAN, B. W., LIN, S. An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal, no. 49, February 1970, pp. 291-308.

KHANG, A.; ROBINS, G. **On Optimal Interconnects for VLSI**. Boston, MA: Kluwer Academic, 1995.

KHATKHATE, A.; LI, C.; AGNIHOTRI, A. R.; YILDIZ, M. C.; ONO, S.; KOH, C.-K.; MADDEN, P. H. Recursive bisection based mixed block placement. In: ISPD ’04: PROCEEDINGS OF THE 2004 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2004, New York, NY, USA. **Anais. . .** ACM Press, 2004. p.84–89.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science, Number 4598, 13 May 1983**, [S.l.], v.220, 4598, p.671–680, 1983.

KONG, T. T. A novel net weighting algorithm for timing-driven placement. In: ICCAD ’02: PROCEEDINGS OF THE 2002 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2002, Los Alamitos, CA, USA. **Anais. . .** IEEE Computer Society, 2002. p.172–176.

KOYANAGI, M.; FUKUSHIMA, T.; TANAKA, T.; **High-Density Through Silicon Vias for 3-D LSIs**;. In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 49-59, 2009.

KUNIO, T; K. Oyama, Y., M. Morimoto, Three-Dimensional IC’s Having Four Stacked Active Device Layers. IEDM Tech. Digest, p. 837 (1989).

LAM, J.; DELOSME, J.-M. Performance of a new annealing schedule. In: DAC ’88: PROCEEDINGS OF THE 25TH ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, 1988, Los Alamitos, CA, USA. **Anais. . .** IEEE Computer Society Press, 1988.p.306–311.

LEE, C. An algorithm for path connections and its applications. **IRE Transactions on Electronics Computers**, [S.l.], p.346–365, September 1961.

LEE, Y.; KIM; HUANG, G. BAKIR, M.;JOSHI, T.; LIM, S.; Co-Design of Signal, Power, and Thermal Distribution Networks for 3D ICs. In: DATE 2009.

LIU, D.; SHEKHAR, S.; Similarity Graphs: A Framework for Declustering Problems, Information Systems Journal, 1996, Vol. 26, Pages, 475-496.

LIU, G.; LI, Z.; ZHOU, Q.; HONG, X.; YANG, H. H. 3D placement algorithm considering vertical channels and guided by 2D placement solution. In: ASICON 2005: 6TH INTERNATIONAL CONFERENCE ON ASIC, 2005. **Anais. . .** [S.l.: s.n.], 2005. p.24–27.

LOH, G.; XIE, Y.; BLACK, B; Processor design in 3D die-stacking technologies. IEEE Micro, vol. 27, n 3, pp 31-48, 2007.

LUO, T.; NEWMARK, D.; PAN, D. Z. A new LP based incremental timing driven placement for high performance designs. In: DAC ’06: PROCEEDINGS OF THE 43RD ANNUAL CONFERENCE ON DESIGN AUTOMATION, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.1115–1120.

MAK, W.K., "Min-Cut Partitioning with Functional Replication for Technology Mapped Circuits using Minimum Area Overhead", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.21(4), pp.491-497, April 2002.

MATSUO, S.; T. Nakahara, K. Tateno, H. Tsuda, T. Kurokawa. Hybrid Integration of Smart Pixel with Vertical-Cavity Surface-Emitting Laser Using Polyimide Bonding. Proceedings of the Topical Meeting on Spatial Light Modulators, 1997, p. 39.

MOORES Law: made real by intel innovation. Available at: <http://www.intel.com /technology/silicon/mooreslaw/>. Acesso em: Aug. 2006.

MOTOYOSHI, M, Through-Silicon Via (TSV), In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 43-48, 2009.

NAM, G.-J.; ALPERT, C. J.; VILLARRUBIA, P.; WINTER, B.; YILDIZ, M. The ISPD2005 placement contest and benchmark suite. In: ISPD ’05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.216–220.

OBENAUS, S.; SZYMANSKI, T. Gravity: fast placement for 3-d vlsi. **ACM Transacions on Design Automation of Electronic Systems (TODAES)**, Volume 8, Issue 3, Pages: 298 - 315, 2003.

OBERMEIER, B.; JOHANNES, F. M. Temperature-aware global placement. In: ASPDAC ’04: PROCEEDINGS OF THE 2004 CONFERENCE ON ASIA SOUTH

PACIFIC DESIGN AUTOMATION, 2004, Piscataway, NJ, USA. **Anais. . .** IEEE Press, 2004. p.143–148.

OBERMEIER, B.; RANKE, H.; JOHANNES, F. M. Kraftwerk: a versatile placement approach. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.242–244.

OGAWA, Y.; PEDRAM, M.; KUH, E. S. Timing-driven placement for general cell layout. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 1990., 1990. **Proceedings. . .** [S.l.: s.n.], 1990. p.872–876.

PATTI, R. Three-dimensional integrated circuits and the future of system-on-chip designs. **Proceedings of IEEE**, [S.l.], v.94, p.1214–1224, 2006.

PAVLIDIS F.; FRIEDMAN, E. G., **Three-dimensional Integrated Circuit Design**; Morgan Kaufmann,304 pgs, 2008.

PAVLIDIS, V. F.; FRIEDMAN, E.G.; **Interconnect-Based Design Methodologies for Three-Dimensional Integrated Circuits**, In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 123-140, 2009.

PELLEGRINI, F., A parallelisable multi-level banded diffusion scheme for computing balanced partitions with smooth boundaries, in: Proceedings of Euro-Par, Rennes, LNCS, vol. 4641, 2007, pp. 191-200. 2007.

PLIMPTON, S; HENDRICKSON, B.; BURNS, S.; Parallel Sn Sweeps on Unstructured Grids: Algorithms for Prioritization, Grid Partitioning and Cycle Detection Steve Plimpton. In Nuclear Science & Engineering 150:267-283, 2005.

PRASITJUTRAKUL, S.; KUBITZ, W. J. A Timing-Driven Global Router for Custom Chip Design. In: ICCAD 2004: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1990. **Anais. . .** [S.l.: s.n.], 1990. p.48– 51.

PREIS, R., DIEKMANN, R. PARTY – Partitioning Library. http://wwwcs.uni-paderborn.de/fachbereich/AG/monien/RESEARCH/PART/party.html. Acesso em junho de 2008.

RAHMAN, A.; REIF, R. System-level performance evaluation of three-dimensional integrated circuits. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.8, December 2000.

RAHMAN, A.; REIF, R. Thermal analysis of three-dimensional (3-D) integrated circuits (ICs). In: IEEE 2001 INTERNATIONAL INTERCONNECT TECHNOLOGY CONFERENCE, 2001. **Proceedings. . .** [S.l.: s.n.], 2001. p.157–159.

REIS, Ricardo; Concepção de Circuitos Integrados. Editora Sagra Luzzatto, 2003.

RIESS, B. M.; ETTELT, G. G. SPEED: fast and efficient timing driven placement. In: ISCAS '95: PROCEEDINGS OF THE IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 1995. **Anais. . .** [S.l.: s.n.], 1995. p.377–380.

ROY, J. A.; PAPA, D. A.; ADYA, S. N.; CHAN, H. H.; NG, A. N.; LU, J. F.; MARKOV, I. L. Capo: robust and scalable open-source min-cut floorplacer. In: ISPD '05: PROCEEDINGS OF THE 2005 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.224–226.

RYU, C. et al., High frequency electrical circuit model of chip-to-chip vertical via interconnection for 3-D chip stacking package, **In: Proceedings of the IEEE Topical Meeting Electr. Performance**, pp. 151-154, 2005.

SAIT, S.; YOUSSEF, H. VLSI Physical Design Automation – Theory and Practice. New York: IEEE, 1995.

SAWICKI, S.; WILKE, G.; JOHANN, Marcelo; REIS, Ricardo. A Cells and I/O Pins Partitioning Refinement Algorithm for 3D VLSI Circuits. In: ICECS 2009, Hammamet, Tunísia (no prelo). 16th IEEE International Conference on Electronics, Circuits and Systems, 2009.

SAWICKI, Sandro; HENTSCHKE, Renato; JOHANN, Marcelo; REIS, Ricardo. An Algorithm for I/O Pins Partitioning Targeting 3D VLSI Integrated Circuits. In: 49TH IEEE INTERNATIONAL MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, 2006, Porto Rico. MWSCAS 2006. IEEE Computer Society, 2006.

SAWICKI, Sandro; HENTSCHKE, Renato; JOHANN, Marcelo; REIS, Ricardo. Unbalancing the I/O Pins Partitioning for Minimizing Inter-Vias in 3D VLSI Circuits. In: 13TH IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, 2006, Nice, França. ICECS 2006. IEEE Computer Society, 2006.

SAWICKI, S.; JOHANN, Marcelo; REIS, Ricardo. Heuristics for Cells and I/O Pads Partitioning Targeting 3D VLSI Circuits, 17th IFIP/IEEE INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, PhD Forum, VLSI-SoC, 2009.

SAWICKI, S.; JOHANN, Marcelo; REIS, Ricardo. A 3D I/O Pads Assignment Based on the Circuit Structure. In: Design Automation and Test in Europe, 2008, Munich. Design Automation and Test in Europe, Phd Forum DATE, 2008.

SAWICKI, S.; HENTSCHKE, Renato; JOHANN, Marcelo; REIS, Ricardo. Particionamento de Pinos de I/O e seu Impacto no Tamanho das Interconexões e Número de Vias em Circuitos VLSI 3D. In: 32a CONFERENCIA LATINOAMERICANA DE INFORMATICA, 2006, Santigo, Chile. CLEI 2006.

SAWICKI, S.; HENTSCHKE, Renato; FLACH, Guilherme; JOHANN, Marcelo; REIS, Ricardo. Studying the Influence of I/O Pads Placement on Wirelength and 3D-Vias of VLSI 3D Integrated Circuits. In: 22th South Symposium on Microelectronics, 2007, Porto Alegre. SIM/SAM. Sociedade Brasileira de Computação. Sociedade Brasileira de Microeletrônica. Porto Alegre : Evangraf, 2007. v. 22. p. 89-92.

SAWICKI, S.; WILKE, G.; JOHANN, Marcelo; REIS, Ricardo. An Iterative Partitioning Refinement Algorithm Based on Simulated Annealing for 3D VLSI Circuits. In: 24th South Symposium on Microelectronics, 2009, Pelotas. 24th South Symposium on Microelectronics, 2009.

SAWICKI, S.; HENTSCHKE, Renato; JOHANN, Marcelo; REIS, Ricardo. Uma Proposta para o Assinalamento de Pads de I/O e Redução de Vias em Circuitos 3D Baseada em Informações da Estrutura do Circuito. In: IBERCHIP XIV Workshop, 2008, Puebla. IWS 2008.

SAWICKI, S.; JOHANN, Marcelo; REIS, Ricardo. Um Algoritmo de Particionamento Iterativo de Células e Pinos de I/O para Circuitos VLSI 3D. In: IBERCHIP, 2009, Buenos Aires. IBERCHIP XV, 2009.

SAWICKI, S.; HENTSCHKE, Renato; JOHANN, Marcelo; REIS, Ricardo. A Proposal for I/O Pins Partitioning and Placement in 3D ICs. In: 21th South Symposium on Microelectronics, 2006, Porto Alegre. SIM 2006. Sociedade Brasileira de Computação. Sociedade Brasileira de Microeletrônica. Porto Alegre: Gráfica Guarani, 2006. p. 241-244.

SECHEN, C.; VICENTELLI, A. S. The Timberwolf Placement and routing package. IEEE Journal of Solid State Circuits, [S.l.], v.SSC-20, april 1985.

SELVAKKMARAN, N.; KARYPIS, G.; Multi-Objective Hypergraph Partitioning Algorithms for Cut and Maximum Subdomain Degree Minimization ,IEEE Transactions on CAD, 25(3), pp. 504-517, 2006

SHEKHAN, S., LIU, D. R. Partitioning similarity graphs: A framework for declustering problmes. Technical Report TR 94–18, University of Minnesota, Department of Computer Science, Minneapolis, MN, 1994.

SHERWANI, N. A. **Algorithms for VLSI Physical Design Automation**. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

SPINDLER, P.; JOHANNES, F. Fast and Robust Quadratic Placement Combined with an Exact Linear Net Model. In: ICCAD ’06: PROCEEDINGS OF THE 2006 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2006. **Anais. . .** [S.l.: s.n.], 2006.

STROOBANDT, D.; CAMPENHOUT, J. V.; Accurate interconnection lengths in three-dimensional computer systems, IEICE Transaction Inf. System (Special Issue on Physical Design in Deep Submicron, vol. 10, n. 1, pp. 99-105, 2000.

SUNTHARALINGAM, V.; BERGER, R.; BURNS, J. A.; CHEN, C. K.; KEAST, C.; KNECHT, J. M.; LAMBERT, R. D.; NEWCOMB, K. L.; O’MARA, D. M.; SHAVER, D. R. D. C.; SOARES, A. M.; STEVENSON, C. N.; TYRRELL, B. M.; WARNER, K.; WHEELER, B. D.; YOST, D.; YOUNG, D. J. Megapixel CMOS image sensor fabricated in three-dimensional integrated circuit technology. In: SOLID-STATE CIRCUITS CONFERENCE, 2005. DIGEST OF TECHNICAL PAPERS. ISSCC, 2005. **Anais. . .** IEEE International, 2005. v.1, p.356–357.

SWARTZ, W.; SECHEN, C. Timing driven placement for large standard cell circuits. In: DAC ’95: PROCEEDINGS OF THE 32ND ACM/IEEE CONFERENCE ON DESIGN AUTOMATION, 1995, New York, NY, USA. **Anais. . .** ACM Press, 1995. p.211–215.

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, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.245–247.

TEZZARON Homepage. Available at: <http://www.tezzaron.com>. Access on: Aug. 2005.

TSAI, C.-H.; KANG, S.-M. Cell-level placement for improving substrate thermal distribution. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.19, February 2000.

TSAI, Y.; WANG, F.; XIE, Y.; VIJAYKRISHNAN, N.; IRWIN, M.; Design Space Exploration for 3-D Cache; IEEE Transaction on Very Scale Integration (VLSI) Systems, Vol. 16. no 4, April, 2008.

VAIDYANATHAN, B.; HUNG, W.; WANG, F. XIE, Y; NARAYANAN, V.; IRWIN, M. J.; Architection microprocessor components in 3D design space. In. Proc, Int. Conf. VLSI Design, pp. 103-108, 2007.

VILLARRUBIA, P., “CPLACE: A standard cell placement program” IBM Tech. Dis. Bull.,vol 32 no. 10A, pp. 341-342, Mar. 1990.

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

VISWANATHAN, N.; PAN, M.; CHU, C. FastPlace 2.0: an efficient analytical placer for mixed-mode designs. In: ASP-DAC ’06: PROCEEDINGS OF THE 2006 CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.195–200.

WALSHAW, C; CROSS, M. JOSTLE: Parallel Multilevel Graph-Partitioning Software - An Overview. In F. Magoules, editor, *Mesh Partitioning Techniques and Domain Decomposition Techniques*, pages 27-58. Civil-Comp Ltd., 2007.

WANG, M.; LIM, S. CONG, J.; SARRAFZADEH, M., Multi-way Partitioning using Bi-partitioning Heuristics. In. ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP DAC, 2000. Proceedings… New York: ACM SIGDA, 2000.

WONG, E.; LIM, S. K. 3D floorplanning with thermal vias. In: DATE ’06: PROCEEDINGS OF THE CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE, 2006, 3001 Leuven, Belgium, Belgium. **Anais. . .** European Design and Automation Association, 2006. p.878–883.

XIU, Z.; RUTENBAR, R. A. Timing-driven placement by grid-warping. In: DAC ’05: PROCEEDINGS OF THE 42ND ANNUAL CONFERENCE ON DESIGN AUTOMATION, 2005, New York, NY, USA. **Anais. . .** ACM Press, 2005. p.585–591.

XU, J.; HONG, X.; JING, T.; CAI, Y.; GU, J. An Efficient Hierarchical Timing-Driven Steiner Tree Algorithm for Global Routing. In: ASP-DAC ’02: PROCEEDINGS OF THE 2002 CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION/VLSI DESIGN, 2002, Washington, DC, USA. **Anais. . .** IEEE Computer Society, 2002. p.473.

YAN, H.; LI, Z.; ZHOU, Q.; HONG, X. Via assignment algorithm for hierarchical 3D placement. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS, CIRCUITS AND SYSTEMS, 2005., 2005. **Proceedings. . .** [S.l.: s.n.], 2005. p.27–30.

YAN, T.; DONG, Q.; TAKASHIMA, Y.; KAJITANI, Y. How does partitioning matter for 3D floorplanning? In: GLSVLSI ’06: PROCEEDINGS OF THE 16TH ACM GREAT LAKES SYMPOSIUM ON VLSI, 2006, New York, NY, USA. **Anais. . .** ACM Press, 2006. p.73–78.

YANG, X.; CHOI, B.-K.; SARRAFZADEH, M. Timing-driven placement using design hierarchy guided constraint generation. In: ICCAD ’02: PROCEEDINGS OF THE 2002 IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 2002, New York, NY, USA. **Anais. . .** ACM Press, 2002. p.177–180.

YAO, B.; CHEN, H.; CHENG, C.-K.; GRAHAM, R. Revisiting floorplan representations. In: ISPD ’01: PROCEEDINGS OF THE 2001 INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2001, New York, NY, USA. **Anais. . .** ACM Press, 2001. p.138– 143.

YILDIZ, M. C.; MADDEN, P. H. Improved cut sequences for partitioning based placement. In: DAC ’01: PROCEEDINGS OF THE 38TH CONFERENCE ON DESIGN AUTOMATION, 2001, New York, NY, USA. **Anais. . .** ACM Press, 2001. p.776–779.

YOUNG, A.; Perspectives on 3D-IC technology; apresentado na 2nd Annual Conference 3D Architectures Semiconductor Integration and. Packaging., San Francisco, CA, Jun. 2005.

ZHANG, T.; ZHANG Y.; SAPATNEKAR, S.; Temperature-aware routing in 3-D ICs; In: Proceeding of IEEE Asia South Pacific Design Automation Conference, pp. 309-314, 2006.

ZHUOYUAN, L.; XIANLONG, H.; QIANG, Z.; SHAN, Z.; JINIAN, B. Efficient Thermal Via Planning Approach and Its Application in 3D Floorplanning. **IEEE Transactions on Computer-Aided Design of Integrated Circuits, 2006,** [S.l.].

ZIPTRONIX TECHNOLOGY: HOMEPAGE. Disponível via WWW em < http://www.ziptronix.com/>. Acesso em Maio de 2009.

ZYCUBE TECHNOLOGY: HOMEPAGE. Disponível via WWW em < http://www.zy-cube.com/e/index.html>. Acesso em Maio de 2009.

ANEXO 1

Tecnologias 3D, Produtos e Serviços

- **Tezzaron Technology:** Empilhamento de *Wafers* e padrão *"bulk"*; vias 3D. Seis dispositivos diferentes de dois níveis foram produzidos com sucesso em 2004. Quatro *wafers* testados e empilhados; *wafers bulk* tem sido empilhados com *wafers* SOI. (Artigos em IAB, 2004; TechVenture 2004, 2005, & 2006; VMIC 2004, 2005, & 2006; Semicon West 2006; 3D-SIC 2007; IMAPS 2007) (http://www.tezzaron.com/)

- **Ziptronix Techmology:** Empilhamento chip-sobre-wafer. 3D SoC anunciado em 2005. (Artigos no TechVenture 2004, 2005, & 2006; MRS 2006; 3D-SIC 2007)(http://www.ziptronix.com/)

- **Cubic Wafer**: Empilhamento de *chips* e *chip*-sobre-*wafer* empilhados usando comunicação através de *micro-bumps* e vias-3D. Trabalhando em parceria com DARPA. (Artigos em TechVenture 2004 & 2006) (http://www. xanoptix.com/)

- **ScanDisk:** Desenvolvimento de memórias 3D. Memórias com 8 camadas anunciadas em 2001. (Paper no VMIC 2005) (http://www.sandisk.com/ Oem/Default. aspx? CatID=1360)

- **ZyCube:** Empilhamento de *Wafers*; vias de tungstênio ou poli-silício; comunicação através de *micro-bumps* (Artigos no TechVenture 2005 & 2006)(http://www.zy-cube.com/e/technology/ZyCube%203D/3dtechnology.html)

- **Vertical Circuits:** *Chip*-sobre-*wafer*; Memórias (Artigo no TechVenture 2004) (http://www.verticalcircuits.com/)

- **FlipChip International:** Serviços 3D relacionados: posicionamento de *wafers*. (Artigo no TechVenture 2006)(http://www.flipchip.com/)

- **Amkor:** Encapsulamento 3D (Artigos no TechVenture 2004 & 2006) (http://www. amkor.com/)

- **ALLVIA**: Conexões Verticais (Artigo no TechVenture 2006) (http://www. allvia .com/)

Pesquisas em Tecnologias 3D

- **Cadence Design Systems**: Pesquisa ferramentas para o projeto de circuitos 3D (Artigos no TechVenture 2004, 2005, 2006, 2007, 2008, 2009) (http://www.cadence.com/).

- **DARPA** *(Defense Advanced Research Projects Agency)*: Pesquisas em processo 3D; integração heterogênea; arquiteturas 3D; FPGAs empilhados;. (Artigos em TechVenture 2004, 2005, 2006, 2007, 2008, 2009) (http://www.darpa.mil/).

- **3D-ROM:** Pesquisas em memórias 3D *read-only*. (http://www.3d-rom.net/).

- **Albany NanoTech:** Pesquisas em empilhamento de *wafers*; interconexões; integração heterogênea; (Artigo no TechVenture 2005) (http://www.albanynanotech.org/).

- **Anadigics**: Pesquisas em módulos de transmissão 3D para celulares. (Artigo no TechVenture 2004)(http://www.anadigics.com/)

- **ASET** (*Association of Super-Advanced Electronic Technologies):* Consórcio Japonês. Pesquisa sensores 3D (com Sanyo Electric) (Artigo no TechVenture 2004; organizador do 3D-SIC 2007)).

- **BeSang:** Pesquisas em torno de memórias e sensores de imagem 3D usando empilhamento de wafers (artigo no TechVenture 2006) (http://www.besang.com/).

- **CEA-Leti:** Pesquisas em empilhamento de *wafers* e empilhamento de chips. (Artigos no TechVenture 2006, 2007, 2008, 2009, IMAPS)(http://www-leti.cea.fr/).

- **Cornell:** Pesquisas relacionadas a problemas térmicos usando *wafers* SOI; crosstalk. FPGA de quatro camadas anunciado em 2005. (Artigos no VMIC 2004, 2005, 2006, 2007, 2008, 2009; VMIC.)(http://www.cornell.edu/).

- **Elpida Memory**: Pesquisas envolvendo memórias (Artigos no TechVenture 2006, 2007, 2008, 2009, 3D-SIC)(http://www.elpida.com/en/index.html)

- **Fraunhofer IZM:** Empilhamento de *Chips* e componentes 3D (sensores, CPUs, memórias). (Artigos no TechVenture 2005, MRS 2006, 3D-SIC 2007, IMAPS 2007)(http://www.pb.izm.fhg.de/).

- **Freescale Semiconductor:** Pesquisas em técnicas de empilhamento de *wafers*. (Artigos no TechVenture 2005 & 2006)(http://www.freescale.com/).

- **GE Global Research:** Pesquisas técnicas de montagem e sensores 3D (Artigo no TechVenture 2004) (http://www.crd.ge.com/)

- **Geórgia Tech:** Pesquisas em síntese física, projeto de ferramentas, questões térmicas e interconexões em circuitos 3D (http://www.3d.gatech.edu/).

- **IBM:** Pesquisas em empilhamento de wafers SOI e ferramentas de projeto 3D; testes de estruturas; metodologias (Artigos no TechVenture 2004 & 2005; VMIC 2005 & 2006; MRS 2006; 3D-SIC 2007; IMAPS 2007) (http://www.watson.ibm.com/).

- **IMEC**: Pesquisa novas abordagens para materiais heterogêneos em circuitos 3D. (Artigos em ISSCC 2004, TechVenture 2005 & 2006, IITC 2006) (http://www.imec.be/)

- **Infineon:** Chip-sobre-wafer, técnica demonstrada em 2002. (Artigos em TechVenture 2004, MRS 2006, 3D-SIC 2007)(http://www.infineon.com/).

- **Intel:** Pesquisas em empilhamento; interconexões 3D; componentes 3D. (Artigos em VMIC 2004, MRS 2006) (http://www.intel.com/).

- **Irvine Sensors**: Pesquisas em processo 3D e questões térmicas; sistemas de integração 3D. (Artigos em TechVenture 2004 & 2005, 3D-SIC 2007) (http://www.irvine-sensors.com/).

- **KINIK Company:** Pesquisas materiais para conexões verticais. (Artigo no VMIC 2006) (http://www.kinik.com/index_en.asp)

- **Laboratory for Physical Sciences (LPS, U. of Maryland):** Interconexões 3D; integração heterogênea 3D. (Artigo no TechVenture 2005, VMIC 2006).

- **Lincoln Labs (MIT):** Questões térmicas; sensores de imagem e radar; primeiro teste do sensor de imagem em 2005. (Artigos no TechVenture 2005, VMIC 2006, MRS 2006)(http://www.ll.mit.edu/).

- **MARCO (Microelectronics Advanced Research Corporation):** Interconexões. (Artigos em TechVenture 2004)(http://marco.fcrp.org/).

- **MCNC (Research & Development Institute**): Pesquisas em Vias 3D; integração heterogênea. (Artigo em TechVenture 2005)(http://www.mcnc.org/).

- **Micron:** Memórias 3D; re-distribuição de camadas; encapsulamento. (Artigos em MRS 2006) (http://www.micron.com/).

- **MIT:** Pesquisa questões de layout; empilhamento de *wafers*; inteconexões (Artigos em IAB 2004; TechVenture 2004 & 2005; MRS 2006; 3D-SIC 2007) (http://www-mtl.mit.edu/ researchgroups /icsystems /3dcsg/ publications.html).

- **NASA-JPL:** Pesquisa estratégias de montagem; sensores, memórias e encapsulamento 3D. (Artigos em TechVenture 2004) (http://www.jpl. nasa.gov/index.html).

- **North Carolina State University:** Pesquisa interconexões 3D, Trabalha com DARPA. (Artigo no VMIC 2006, TechVenture 2006) (http://www.ncsu.edu/).

- **Penn State University**: Pesquisa ferramentas para síntese física. (Artigo no TechVenture 2006) (http://www.cse.psu.edu/)

- **PTC:** Pesquisa em fluxos de projeto 3D; questões térmicas; verificação. (http://www. ptc.com/)

- **Rensselaer Polytechnic Institute (RPI):** Empilhamento de *wafers*; pads de cobre; alinhamento; interconexões. (Artigos em TechVenture 2004, 2005, & 2006; VMIC 2004, 2005, & 2006; MRS 2006) (http://www.rpi.edu/).

- **Reveo**: Pesquisa multi camadas de memória. (Artigo no TechVenture 2004) (http: // www.reveo.com/)

- **RTI (Research Triangle Institute**): *Chip*-sobre-*wafer*; integração heterogênea. (Artigos no TechVenture 2005 & 2006; MRS 2006; 3D-SIC 2007) (http://www.rti.org/)

- **Samsung:** Empilhamento de *chips*; *chip*-sobre-*wafer*; vias 3D. (Artigos no TechVenture 2005 & 2006) (http://www.samsung.com/)

- **SEMATECH R&D Consortium: Pesquisa em interconexões 3D;** Organizou o Simpósio "3D Chips: Critical and Practical Aspects of Manufacturing" em Abril de 2004 e o "3D Infrastructure Roadmap Workshop" em Junho de 2005. (Artigos no TechVenture 2005, VMIC 2006, 3D-SIC 2007) (http://www.sematech.org/)

- **Semico:** Pesquisa *drivers* para componentes 3D (Artigos no TechVenture 2004) (http://www.semico.com/)

- **Sharp Corporation:** Pesquisa em encapsulamento 3D. (Artigo no TechVenture 2004) (http://www.sharp-world.com/).

- **SRC (Semiconductor Research Corporation):** Pesquisa estruturas e processos 3D (Patrocinador da TechVenture 2005) (http://src.org/)

- **Stanford University**: Pesquisa interconexões 3D; integridade; performance; FPGAs 3D. (Artigos em TechVenture, Great Lakes Symposium of VLSI, International Symposium on System Designs (ISPD), and IAB, all in 2004; VMIC 2004, 2005, & 2006; TechVenture 2005 & 2006) (http://www.stanford.edu/)

- **STATSChipPAC:** Empilhamento de *wafers*; alinhamento (Artigos em TechVenture 2004 & 2006, IMAPS 2007) (http://www.statschippac.com/)

- **STMicroelectronics**: Pesquisas em alinhamento de *chips* e *wafers*. (Artigo em TechVenture 2006) (http://www.st.com/)

- **Sun Microsystems:** Pesquisas em interconexões 3D; potência; largura de banda. (Artigo noTechVenture 2004) (http://www.sun.com/)

- **Technische Universität Berlin:** Pesquisas em materiais para chips 3D. (Artigos em MRS 2006, IMAPS 2007) (http://www.tu-berlin.de/eng/index.html)

- **Tessera:** Encapsulamento 3D (Artigos em TechVenture 2004) (http:// www. tessera. com/)

- **Tohoku University:** Pesquisas sobre tecnologias 3D incluindo o empilhamento de *wafers*. (Artigos em VMIC 2004; TechVenture 2005) (http:// web.bureau. tohoku.ac. jp/international/home-e.html)

- **Toshiba**: Pesquisas em empilhamento de chips; células de memória 3D. (Artigos em TechVenture 2004, 2005, & 2006) (http://www.toshiba.com/)

- **UCLA (VLSI CAD LAB Group):** Pesquisa em ferramentas de projeto envolvendo problemas térmicos (Artigo em VMIC 2006) (http://cadlab.cs.ucla.edu/)

- **Xilinx**: Pesquisas em encapsulamento 3D; trabalhando com DARPA. (Artigo no TechVenture 2006) (http://www.xilinx.com/).