Heurística para o Particionamento de Células e Pinos de I/O Dirigidas para Circuitos VLSI 3D

Sandro Sawicki[1,2], Gustavo Wilke[1], Marcelo Johann[1], Ricardo Reis[1],

[1] UFRGS – Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
{sawicki, wilke, johann, reis}@inf.ufrgs.br

[2] UNIJUI – Universidade Regional do Noroeste do Estado do Rio Grande do Sul
Departamento de Tecnologia, Ijui, Brazil
sawicki@unijui.edu.br

Abstract. Partitioning algorithms are responsible for dividing random logic cells and ip blocks into the different tiers of a 3D design. Cells partitioning also helps to reduce the complexity of the next steps of the physical synthesis (placement and routing). In spite of the importance of the cell partitioning for the automatic synthesis of 3D designs it been performed in the same way as in 2D designs. Graph partitioning algorithms are used to divide the cells into the different tiers without accounting for any tier location information. Due to the single dimensional alignment of the tiers connections between the bottom and top tiers have to go through all the tiers in between, e. g., in a design with 5 tiers a connection between the top and the bottom tiers would require 4 3D vias. 3D vias are costly in terms of routing resources and delay and therefore must be minimized. This paper presents a methodology for reducing the number of 3D vias during the circuit partitioning step by avoiding connections between non-adjacent tiers. The proposed algorithm minimizes the total number of 3D vias and long 3D vias while respecting area balance, number of tiers and I/O pins balance. Experimental results show that the number of 3D-Vias was reduced by 19%, 17%, 12% and 16% when benchmark circuits were designed using two, three, four and five tires.

Keywords: 3D VLSI Circuits, partitioning, CAD

1. Introdução

O projeto de circuitos 3D já é uma realidade tanto na indústria VLSI quanto na academia. Grandes empresas como IBM, Intel, AMD, Samsung, Micron, Cadence, Infineon, e *startups* como Ziptronix, Xanoptix, ZyCube e Tezzaron estão investindo

em soluções relacionadas a essa área. Na academia, são destacadas iniciativas do MIT, Stanford University, University of Cornell, University of Hannover Leibniz, University of Minessotta, IMEC, Purdue University, Tohuku University entre outras.

Embora as mais recentes tecnologias de fabricação discutam questões relacionadas às conexões (tais como integridade do sinal, potência e atraso), o projeto 3D surge como uma nova maneira de tratá-las [1-3]. Entretanto, a tecnologia 3D também introduz suas próprias questões. Uma delas é a dissipação térmica, a qual é bastante estudada no nível de *floorplanning* [4] e também no nível de posicionamento [3]. Outra importante questão diz respeito ao mecanismo de comunicação "intra-tier", conhecido como via-3D, pois impõe significantes limitações para o projeto em três dimensões. Percebe-se, que a literatura que aborda os circuitos 3D não trata essa questão com devida propriedade, contudo, é fácil enumerar diversas questões sobre esse assunto que demandam cuidados, como por exemplo: (1) a superpopulação de vias-3D gera o aumento de área ativa, com isso, ocupa o lugar dos transistores e, consequentemente, aumenta a área do circuito; (2) no momento 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 de sua estrutura, as vias-3D tem características elétricas diferentes das conexões normais (o problema ocorre quando a via-3D faz parte do caminho crítico); (4) as vias-3D são obstáculos, como isso utilizam recursos de roteamento; (5) o pitch de uma via-3D é muito grande se comparado com o de uma conexão normal, entre outros.

A integração 3D pode atuar em três níveis de projeto, todos baseados em sua granularidade. O primeiro, chamado de "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 grandes blocos de circuitos (IP *cores*) entre diferentes tiers, promovendo, dentro destas, uma integração acoplada. Por isso, os blocos podem ser distribuídos de forma organizada, em cada *tier* ou entre elas, praticamente 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 em granularidade menor (células). Essa metodologia trabalha com um nível de integração elevado e impõe ainda, por parte de alguns trabalhos, certa resistência à idéia de posicionar células em 3 dimensões [6]. Nesse caso, pode-se entender que quanto maior a granularidade maior a demanda por vias-3D o que pode gerar mais limitações físicas. Por outro lado, a evolução do tamanho das vias-3D está aumentando rapidamente e já é viável (para alguns modelos) realizar esse tipo de integração [2, 5, 11, 13], pois já se podem construir vias-3D com *pitch face-to-face* 0,5 μm [6] e *face-to-back* 2,4 μm [5] .

Embora seja visível que a inserção de mais vias-3D melhora o tamanho das conexões [5], essa metodologia de projeto ignora as questões levantadas acima. Acredita-se que as ferramentas de EDA podem desempenhar um importante papel na redução de vias-3D em circuitos com lógica aleatória. O número de vias-3D exigido em um projeto é determinado pelo nível de atribuição das células, que é realizada durante a etapa de particionamento. Atualmente, essa etapa é realizada por

particionadores de hipergrafos, tais como hMetis [8] como no trabalho de Ababei [2]. Contudo, algoritmos de particionamento de hipergrafos não foram projetados para atuar em problemas aplicados aos circuitos 3D, pois a eles são atribuídas células ao longo de uma única dimensão. 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 *tier* para outra adjacente atravessa todas as camadas 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.

Este artigo pretende demonstrar que utilizando uma nova heurística baseada em *Simulated Annealing* para refinar o particionamento utilizado pelos algoritmos tradicionais, pode-se minimizar o número total de vias-3D enquanto mantém o equilíbrio da área de células e número de pinos de I/O, reduzindo o número de vias-3D longas (aumento do número de conexões curtas). O restante desse artigo está organizado da seguinte forma. A Seção 2 apresenta a formulação do problema, a Seção 3 descreve como é tratado o particionamento de células em circuitos 3D usando exemplos em trabalhos relacionados. A Seção 4 apresenta a heurística desenvolvida para lidar com as vias-3D e seu mecanismo de otimização. A Seção 5 discute os resultados experimentais e logo em seguida apresenta as conclusões do trabalho.

2. 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 de pinos de I/O e também o particionamento de células nas *tiers*, tal que, a quantidade de vias-3D seja minimizada enquanto restringe o número de redes verticais longas, reduz a área e equilibra o número de pinos de I/O entre as tiers.

3. Trabalhos Relacionados e Algoritmo Proposto

Apesar da qualidade dos resultados oferecidos pelos algoritmos de particionamento [8, 15, 16] sua aplicação em circuitos VLSI 3D não atua na identificação das conexões que cruzam por mais de duas tiers adjacentes [1, 2, 3]. Isso ocorre pois sua estrutura não foi desenvolvida para atuar em problemas que visam reduzir conexões longas. Em geral, a atuação desses algoritmos resulta em um grafo completo, onde o peso nas arestas indica o número de redes que conectam as diferentes partições. Assim, no momento de fixar as partições em linha, surgem as conexões longas.

Trabalhos em [1, 2, 3, 17, 18], particionam o circuitos através do particionador de hipergrafos hMetis [8]. Lee e Lim [19] utilizam *min-cut* por meio do algoritmo de Fiduccia-Matheyses [15]. Entretanto, como mencionado anteriormente, para problemas em três dimensões, tanto hMetis quanto FM não foram desenvolvidos para tal propósito.

Este tópico propõe uma heurística iterativa para manipular o problema do surgimento das conexões longas. O algoritmo desenvolvido é inspirado em *Simulated Annealing* [14], contudo, ao invés de aceitar as soluções piores para evitar mínimos locais, essa heurística utiliza-se de uma boa solução inicial (suficientemente próxima da ótima) utilizando um trabalho anterior apresentado em [9].

A principal diferença entre a nova abordagem e o particionador de hipergrafos hMetis é que ela conhece a localização das partições. Na verdade, em um circuito 3D, as partições são organizadas em linha, isso implica no conhecimento das partições adjacentes (que são baratas em termos de corte) e partições distantes (que são caras, pois demandam Vias-3D extras).

Objetivando minimizar as Vias-3D como um todo, pretende-se penalizar o corte das partições distantes que não são tratadas por particionadores de hipergrafos. Por exemplo, o algoritmo apresentado em [9] 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 1-a). Embora essa abordagem tenha o mesmo objetivo, é clara a sua limitação, pois os grupos não podem ser quebrados. O algoritmo proposto nesse item é a fusão dos dois passos referidos, como ilustrado na Figura 1-b). Essa heurística resolverá principalmente essa limitação.

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

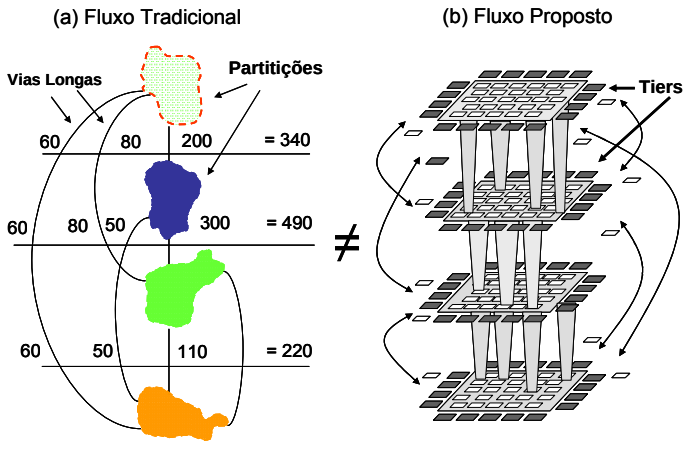

Figura 1: Posição das tiers e partições.

3. Heurística de Particionamento Proposta

O algoritmo proposto escolhe a solução inicial apresentada em trabalhos anteriores [9] e 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.

3.1 Procedimento de Perturbação

A função de perturbação projetada move as células através das partições. Embora sejam de natureza aleatória, é executada de duas diferentes formas: movimentação *simples* ou *dupla troca*. A perturbação simples e dupla são alternadas com 50% de chances de serem executadas 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 (cada elemento localizado em uma partição diferente). Cada elemento pode ser tanto uma célula, quanto um pino (com 50% chances de cada elemento ser selecionado), totalizando 4 diferentes perturbações duplas, cada uma tendo 25% de probabilidade de ocorrer.

3.2 Função de Custo

Qualquer estado intermediário no processo de particionamento pode ter sua qualidade medida por uma função de custo. Na função de custo, foram modeladas todas as métricas de interesse em um único número que representa o custo.

A função de custo é dividida em três partes distintas: um custo ***v*** associado aos recursos utilizados pelas Vias-3D, um valor ***a*** para o balanceamento da área e um custo ***p*** para o balanceamento de pinos de I/O. O custo relatado é uma combinação de três partes. Com o intuito de utilizá-las em conjunto, a equação foi normalizada dividindo 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 a função de custo para 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; adicionar o número computado para cada rede para que seja obtido em ***v***. O quadrado é aplicado para punir 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 na mesma maneira que o valor de *a*.

$$: \frac{(w_v \times v)}{v_i} + \frac{(w_a \times a)}{a_i} + \frac{(w_p \times p)}{p_i} \qquad (1)$$

3.3. Função de *Schedule*

Essa função computa o valor da temperatura inicial e sua variação ao longo do algoritmo. Determina também o comportamento da aceitação da simulação de têmpera. Por exemplo, considere a abordagem clássica de *Simulated Annealing*, começando com temperaturas altas. Neste caso, a variação da temperatura é de fundamental importância, pois ela determinará quanto tempo o algoritmo será aleatório e quanto tempo será guloso e decidirá, também, o impacto dessa mudança.

Nessa etapa do trabalho, a *netlist* inicial já foi convertida para netlist 3D e otimizada pelo algoritmo descrito em [9]. Sendo assim, o refinamento parte de uma solução inicial muito boa. Por esse motivo, o algoritmo de *Simulated Annealing* é definido com temperatura constante em zero. Em outras palavras, o algoritmo não aceita movimentos que piorem o estado atual do algoritmo segundo a definição da função de custo.

4. Resultados Experimentais

O algoritmo de particionamento proposto foi comparado com o particionador de hipergrafos estado-da-arte, *hMetis* e também com o algoritmo I/O Pins. Tais algoritmos 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 subseqüente, as partições são indicadas para as *tiers*, como ilustrada na Figura 1-a, inspirada no método apresentado por [2]. O algoritmo *hMetis* tem a liberdade particionar livremente, pinos de I/O e células. No trabalho anterior [9] executou-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 um bom balanceamento, e uma heurística foi executada a fim de auxiliar na redução do corte. Nesse trabalho foi demonstrado que o método é capaz de reduzir o corte em 5,28% (2 tiers), 33,28% (3 tiers), 9,53% (4 tiers) e 16,50% (5 tiers) em comparação com *hMetis*. Como esse método foi concentrado nos pinos de I/O, chamamos de *I/O Pins.* Note que o método proposto para esse trabalho inicia com a solução obtida pelo particionamento de pinos de I/O e executa a nova heurística para refinar células e pinos de I/O.

A configuração dos experimentos são as seguintes. Foram utilizados circuitos *benchmarks* ISPD 2004 [12] e o projeto foi desenvolvido em 2, 3, 4 e 5 tiers. Os três métodos referidos (hMetis, I/O Pins e Proposto) foram comparados entre si. Em todos os casos a distribuição da área foi rígida, resultando como pior caso de desbalanceamento de área em 0,1%. O balanceamento de pinos de I/O não foi imposto com hMetis, pois não ele não contém essa restrição. Por essa razão, o hMetis teve o pior caso de desbalanceamento de pinos de I/O. Já o método proposto dá pouca liberdade para os pinos de I/O para melhorar a quantidade de vias-3D.

A Figura 2 mostra o número total de Vias-3D comparado com os demais métodos. O método proposto obtém os melhores resultados (a média mais baixa de Vias-3D). As melhorias estão na ordem de 18% e 11% comparado com hMetis e I/O Pins respectivamente para 2 tiers, 17% e 9% para 3 tiers, 11% e 6% para 4 tiers e, finalmente, 16% e 7% para 5 tiers.

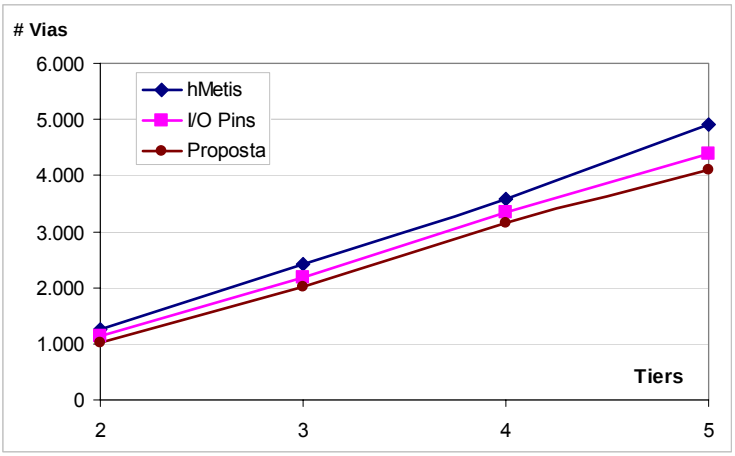

Figura 2: Número Total de Vias-3D

A Figura 3 apresenta o particionamento final sob a perspectiva de um hipergrafo. A comparação é realizada com outros dois particionadores, hMetis e I/O pins. O eixo-y mostra a média do corte entre as diferentes partições. Pode-se observar que o algoritmo proposto aumenta o valor do corte no momento que o número de partições aumenta. Entretanto, quando somente duas partições são criadas o algoritmo obtém o melhor corte se comparado com as outras estratégias de particionamento. Esse comportamento é explicado pela a função de custo usada para otimizar, pois, ao contrário do hMetis e I/O pins, o algoritmo proposto 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 proposto 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 conduz para um aumento do corte entre diferentes partições enquanto reduz o número total de vias-3D.

A Figura 4 analisa o comportamento da Figura 3 sob a ótica do número de vias-3D. Nesse experimento, cada faixa (*bar*) do gráfico representa do número total de vias-3D obtidos por cada algoritmo ao longo de 5 tiers. Cada faixa é dividida em quatro partes, uma parte representa o número de vias-3D que conecta *tiers* adjacentes, enquanto as outras três partes representam o número de vias-3D que conecta *tiers* não-adjacentes. O bloco identificado pelo número 2 descreve a quantidade de vias-3D necessárias para conectar diretamente duas *tiers*. Nesse caso, o uso de duas vias-3D indica que a conexão necessita cruzar através de uma *tier*. Por sua vez, os blocos identificados pelos números 3 e 4, representam também quantidade de vias-3D necessárias para se conectar diretamente duas tiers. Da mesma forma, o bloco identificado com o número 3 indica que é necessário atravessar duas tiers, já para o bloco identificado com o número 4 sinaliza a necessidade de se atravessar três tiers.

A Figura 3 mostra claramente que as conexões que cruzam mais de duas tiers adjacentes migram para tiers adjacentes, reduzindo o número total de vias-3D. Tal

comportamento se deve pela função de custo do algoritmo que penaliza conexões longas usando o quadrado do número de conexões a cada nova iteração (descrito na Seção 3.2).

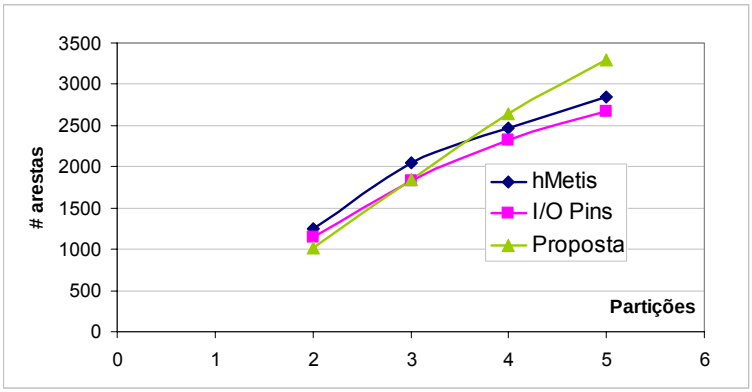

Figura 3: Qualidade do corte *(min-cut)*

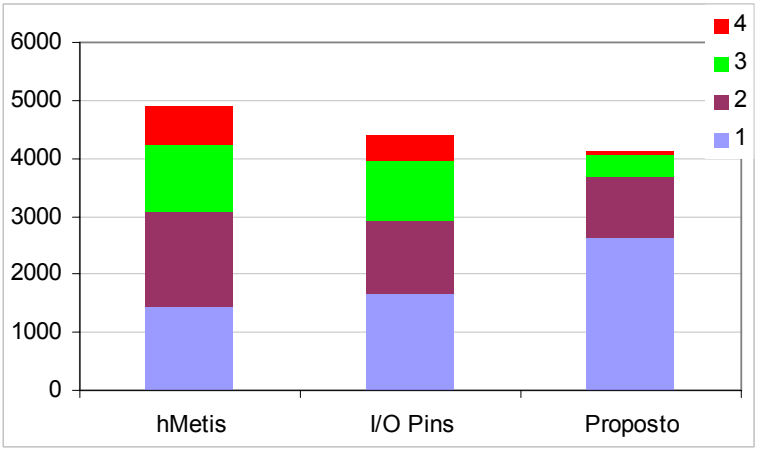

Figura 4: Distribuição das vias-3D em um projeto de 5 *tiers*.

A Tabela 1 abaixo mostra a comparação entre o algoritmo proposto e os algoritmos hMetis e I/O Pins. Analisando a tabela, percebe-se que ela descreve o número de vias-3D que cruzam 2, 3 e 4 tiers e também as conexões que não necessitam cruzar nenhuma *tier*. Constata-se pela Tabela 1 o aumento das conexões curtas em detrimento das conexões longas, mesmo comportamento da Figura 3.

Conclusões

Este artigo apresentou um método para refinar o particionamento de células e pinos de I/O em circuitos VLSI 3D. Essa heurística iterativa é capaz de melhor o resultado obtido pelos particionadores de hipergrafos. O novo método demonstra que os

particionadores de hipergrafos não são eficientes quando aplicados em problemas que exigem o conhecimento prévio da localização das partições. No caso dos circuitos 3D, as partições estão em linha, fazendo com que os particionadores de hipergrafos não impeçam o surgimento de conexões verticais longas (vias-3D que cruzam tiers não adjacentes).

Nesse contexto, foi demonstrado que a heurística proposta nesse trabalho foi capaz de melhorar o número total de vias-3D e vias-3D longas considerando a posição de cada *tier* dentro de um circuito 3D. Além disso, essa heurística executa o particionamento enquanto mantém o número de pinos de I/O e células balanceados entre todas as tiers.

Tabela 1: Número vias-3D longas distribuídas entre 5 *tiers*

Bench	hMetis				I/O Pins				Proposta			
	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

Agradecimentos

Os autores agradecem a contribuição do Dr. Renato Hentschke. Dr. Hentschke participou do início desse trabalho enquanto desenvolvia sua tese de doutorado pela Universidade Federal do Rio Grande do Sul.

Referências

1. W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer and P. D. Franzon; Demystifying 3D ICs: The Pros and Cons of Going Vertical. IEEE Design and Test of Computers – special issue on 3D integration; pp 498-510, Nov.-Dec. 2005.
2. C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan and S. Sapatnekar. Placement and Routing in 3D Integrated Circuits. IEEE Design and Test of Computers – special issue on 3D integration; pp 520-531, Nov.-Dec. 2005.

3. B. Goplen; S. Sapatnekar; Efficient Thermal Placement of Standard Cells in 3D ICs using Forced Directed Approach. In: Internation Conference on Computer Aided Design, ICCAD'03, November, San Jose, California, USA, 2003.
4. E. Wong; S. Lim. 3D Floorplanning with Thermal Vias. In: DATE '06: Proceedings of the Conference on Design, Automation and Test in Europe, 2006. p.878–883.
5. 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 59 Physical Design, 2004, New York, NY, USA. Anais. . . ACM Press, 2004. p.108–115.
6. 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.
7. 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.
8. G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel Hypergraph Partitioning: Application in VLSI domain. In Proceedings of 34th Annual Conference on. Design Automation, DAC 1997, pages 526–529, 1997.
9. Sawicki, S.; 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.
10. K. Bernstein; P. Andry; J. Cann; P. Emma; D. Greenberg; W. Haensch; M. Ignatowski; S. Koester; J. Magerlein; R. Puri; A. Young. 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
11. K. Banerjee and S. Souri and P. Kapur and K. Saraswat. 3D-ICs: A Novel Chip Design for Improving Deep Submicrometer Interconnect Performance and Systems on-Chip Integration. Proceedings of IEEE, vol 89, issue 5, 2001.
12. ISPD 2004 - IBM Standard Cell Benchmarks with Pads. http:// www. public. Iastate .edu/~nataraj/ISPD04_ Bench.html#Benchmark_Description. Access on Mar 2009.
13. R. Hentschke, G. Flach, F. Pinto, and R. Reis, "Quadratic Placement for 3D Circuits Using Z-Cell Shifting, 3D Iterative Refinement and Simulated Annealing," Proc. Symp. on Integrated Circuits and Syst. Des. '06, 220-225.
14. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, Science, 1983, 220, pages 671-680.
15. C. Fiduccia, M. Mattheyses. A Linear Time Heuristic for improving network partitions. In. Proceedings 19th IEEE Desing Automation Conference. Pages 175-181, 1982.
16. B. Kernighan, S. Lin. An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal, no. 49, February 1970, pp. 291-308.
17. Tsai, Yuh-Fang; 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.
18. 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.
19. 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.