Um Algoritmo de Particionamento Iterativo de Células e Pinos de I/O para Circuito VLSI 3D

Sandro Sawicki[1, 2]
sawicki@inf.ufrgs.br

Marcelo Johann[1]
johann@inf.ufrgs.br

Ricardo Reis[1]
reis@inf.ufrgs.br

[1]UFRGS – Universidade Federal do Rio Grande do Sul
PPGC - Instituto de Informática
Porto Alegre, Brazil.

Abstract— **Os algoritmos de particionamento de células em circuitos 3D são fundamentais para o assinalamento de células e blocos, criação de camadas (tiers), além de reduzir a complexidade dos posicionadores. Contudo, abordam o problema de particionamento da mesma forma que em circuitos planares, adaptando os mesmos algoritmos para atuarem em circuitos 3D . Essa solução é simplista e faz com que o algoritmo não perceba a criação de conexões longas, aumentando assim, o número de vias do circuito e, consequentemente, sua área. Na verdade, considerando que o caminho de uma tier para outra tier adjacente envolve atravessar todas as camadas de metal, é evidente que qualquer ligação vertical superior a adjacente pode ser muito mais custosa para o roteamento, sem mencionar o tamanho da área ativa ocupada e atraso. Este artigo apresenta uma metodologia, baseada em Simulated Annealing, que reduz o número total de conexões verticais considerando o tamanho das vias longas, área do circuito, balanceamento da área das células, número de tiers e pinos de I/O. Resultados experimentais, mostram que essa técnica reduz o número total de vias-3D em 22,80%, 17%, 13,60 e 19,50% e o número máximo de vias-3D entre um par adjacente de tiers em 22,80%, 34,27%, 11,75% e 15,04% para duas, três, quatro e cinco tiers respectivamente, em comparação com a abordagem planar que utiliza o algoritmo hMetis.**

I. INTRODUÇÃO

O projeto de circuitos VLSI 3D está se tornando realidade na indústria de semicondutores e academia. Recentes tecnologias de fabricação introduzem inúmeros problemas relacionados às conexões, tais como, integridade de sinal, potência, *yield* e atraso. A tecnologia 3D surge como uma ajuda significante para a redução do tamanho das conexões e, conseqüentemente, atenuar essas questões [1-3].

Entretanto, a tecnologia 3D também insere suas próprias questões. Um exemplo bastante estudado trata das questões térmicas, com trabalhos em nível de *floorplaning* [4] e posicionamento [3]. O mecanismo de comunicação intra-camadas (*inter-tiers*), chamado Via-3D, dispõe de inúmeras limitações para o projeto VLSI 3D por quatro razões principais: (que ainda não foram bem abordadas na literatura) 1) todos os elementos de comunicação que implicam no nível de integração *face-to-back* ocupam espaço na camada ativa, limitando o posicionamento de células e blocos; 2) a densidade da Via-3D é consideravelmente pequena comparado as vias normais, assim, não permite grande número conexões verticais; atualmente, são características desejadas pelas ferramentas de CAD; 3) as Vias-3D tem características elétricas diferentes das conexões normais, especialmente se considerarmos que as conexões verticais necessitam atravessar todas as camadas de metal; 4) Vias-3D impõem problemas de *yield* e elétricos não somente porque seus elementos são difíceis de fabricar e também porque consomem muitos recursos de roteamento.

A integração 3D pode ter muitos níveis de granularidade, variando do nível de transistores para o nível de *tiers*. Enquanto o nível de blocos e o nível de *tiers* são bem aceitos na literatura, parece que existe alguma resistência à idéia de posicionar células em 3D [6]. Uma das razões é que a baixa granularidade demanda maior número de vias-3D para integração, impondo restrições físicas. Por outro lado, o tamanho da via-3D está em constante evolução e já é viável (para a maioria dos projetos) realizar essa integração [5, 10, 13]. Uma via-3D já pode ser construída em 0.5 μm para integração *face-to-face* [6] e 2.4 μm em *face-to-back* [5]; acredita-se que essa limitação também está inserida nas ferramentas, pois muitos algoritmos de otimização para o tratamentos de circuitos 3D são adaptações de ferramentas 2D, não acoplando algumas questões, como por exemplo, as Vias-3D [7].

Embora se saiba que a inserção de Vias-3D melhora o comprimento dos fios [5], essa metodologia de projeto pode resolver outras questões, como assinalado no primeiro parágrafo. Acredita-se que ferramentas para EDA possam desempenhar um papel importante, permitindo otimizar lógica aleatória em circuitos 3D. Uma das possíveis abordagens é minimizar Vias-3D em níveis aceitáveis.

Dentre os novos problemas da EDA relacionados à tecnologia 3D, o particionamento de componentes, mais especificamente de células *(standard cells)* ainda necessitam de mais pesquisa. O particionamento de células é usualmente executada através do particionamento de hipergrafos, porque

UNIJUI – Universidade Regional do Noroeste do Estado do Rio Grande do Sul
DeTec – Departamento de Tecnologia, Santa Rosa, RS, Brazil

o procedimento de mapear a *netlist* em um hipergrafo se torna mais simples pela semelhança estrutural [8]. Por outro lado, ferramentas de particionamento de hipergrafos não compreendem a distribuição nas células no espaço (circuitos 3D são distribuídos em um plano 1D). Assim, com esse procedimento, o particionamento deixa de obter melhores resultados, pois as conexões longas são inevitáveis.

É importante compreender que o valor dos recursos usados é um múltiplo da distância vertical das *tiers*; na verdade, considerando que o caminho de uma *tier* para outra *tier* adjacente envolve atravessar todas as camadas de mental, é evidente que qualquer ligação vertical superior a adjacente pode ser muito mais custoso para o roteamento, sem mencionar o tamanho da área ativa ocupada e atraso.

Neste artigo, pretende-se demonstrar que utilizando uma heurística gulosa, baseada em *Simulated Annealing* é possível encontrar melhores resultados na redução de Vias-3D mantendo equilíbrio no número de pinos de I/O entre as partições, minimizando o número de conexões verticais longas e equilibrando a área. O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta o problema a ser abordado, a seção 3 descreve a solução e abordagens relacionadas a esse trabalho. A Seção 4 apresenta detalhes do algoritmo proposto. A Seção 5 apresenta as conclusões e direções para trabalhos futuros.

II. Formulação do Problema

Considere um circuito de lógica aleatória e o *floorplaning* 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, reduzindo a área e mantendo equilibrado o número de pinos de I/O distribuindo-os entre as tiers.

III. Algoritmo de Particionamento e Trabalhos Relacionados

Em trabalhos anteriores [9] apresentamos uma solução para o problema proposto. Aquele trabalho concentrou-se no particionamento de pinos de I/O antes do particionamento de células, produzindo melhorias na ordem de 5,33% (2 tiers), 33,86% (3 tiers), 9,59% (4 tiers) e 16,53% (5 tiers). Foi utilizado a ferramenta hMetis [8] para particionar as células enquanto os pinos de I/O foram fixados pelo método. Isso melhorou a capacidade do hMetis de obter melhor corte entre as partições devido à informação da localização dos pinos.

Este artigo propõe uma heurística de melhoramento iterativo para manipular o problema proposto na seção II. O algoritmo é inspirado em *Simulated Annealing* [13], porém, ao invés de aceitar soluções piores para evitar mínimos locais, essa heurística entende que pode obter uma boa solução inicial (suficientemente próxima da ótima) utilizando o trabalho anterior apresentado em [9].

A principal diferença entre da 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] emprega o hMetis para particionar 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 artigo 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.

IV. Heurística de Particionamento

O algoritmo proposto escolhe a solução inicial objetiva do trabalho anterior apresentado em [9] 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.

A. 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.

B. Função de Custo

Qualquer estado intermediário no processo de particionamento pode ter sua qualidade medida por uma

função de custo. Na função de custo, foi modelado 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 junto, a equação foi normalizada dividindo cada número pelo seu valor inicial v_i, a_i e pi (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.

$$c = \frac{(w_v \times v)}{v_i} + \frac{(w_a \times a)}{a_1} + \frac{(w_p \times p)}{p_1} \qquad (1)$$

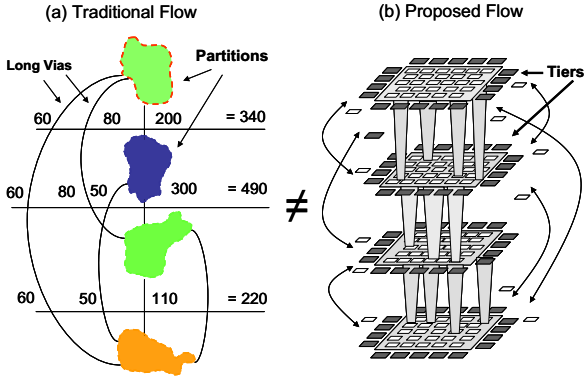

Figure 1. Posição das Tiers

V. RESULTADOS EXPERIMENTAIS

O algoritmo de particionamento proposto foi comparado com o particionador de hipergrados estado-da-arte, conhecido como hMetis. O algoritmo teve a liberdade de particionar a *netlist* (incluindo células e pinos de I/O) para n partições, onde n também era o número de *tiers*. No estágio subseqüente as partições são indicadas para as *tiers*, como ilustrada na figura 1, inspirada no método apresentado por [2]. O hMetis tem a liberdade particionar livremente, pinos de I/O e células. Chamamos esse método de hMetis para fins experimentais. No trabalho anterior executou-se uma abordagem ligeiramente 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. Em [9] foi demonstrado que esse método foi capaz de reduzir o corte em 5,33% (2 tiers), 33,86% (3 tiers), 9,59% (4 tiers) e 16,53% (5 tiers). 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 [11] e o projeto foi desenvolvido em 2, 3, 4 e 5 tiers. Os três métodos referidos (hMetis, I/O Pins e proposto) foram comparados. Em todos os casos a distribuição da área foi rígida, resultando como pior caso de desbalanceamento 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, enquanto o método I/O pins tem o melhor caso. 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 três métodos. O método proposto obtém os melhores resultados (a média mais baixa de Vias-3D). As melhorias estão na ordem de 23% e 12% comparado com hMetis e I/O Pins respectivamente para 2 tiers, 42% e 9% para 3 tiers, 14% e 6% para 4 tiers e finalmente 20% e 7% para 5 tiers.

A figura 3 mostra o número máximo de vias-3D entre um par de tiers adjacentes. Para compreender melhor essa figura, imagine que entre cada par de tiers adjacentes existe uma Via-3D. 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 adjacentes, quanto maior o número de conexões, maior será o aumento da área da tiers. Os resultados obtidos pelo método proposto reduz em todos os casos esse número, executando o método de forma similar à redução do número total de vias.

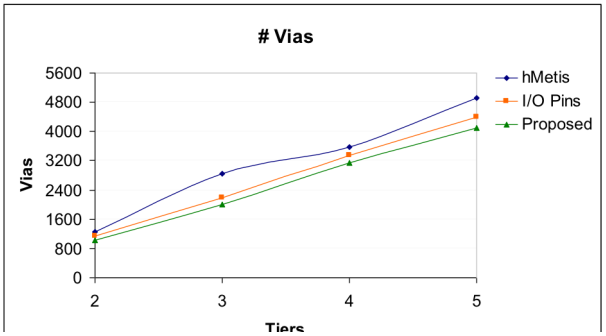

Figure 2. Número total de Vias-3D (o método proposto obtém menor número de vias-3D em todos os níveis)

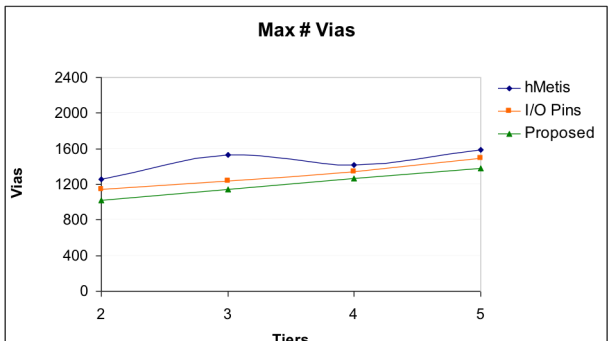

Figure 3. Máximo número de Vias-3D entre um par de tiers adjacentes

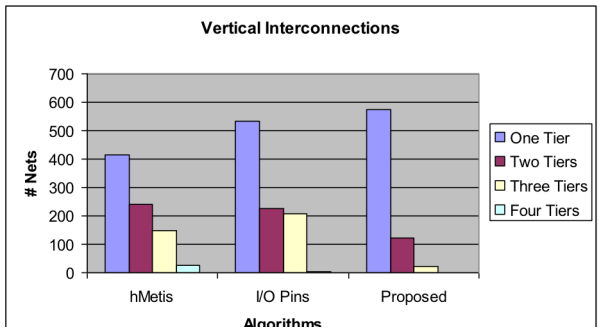

Figure 4. Distribuição do tamanho das redes verticais. O histograma mostra somente redes que cruzam as tires.

Finalmente, analisou-se o tamanho das conexões verticais longas. Na figura 4 foram agrupadas redes pelo comprimento das conexões verticais (omitiram-se redes que ficaram em uma única *tiers*). O gráfico demonstra claramente que o método proposto converge para a criação de conexões curtas.

VI. CONCLUSÕES

Este artigo apresentou um método de refinamento de pinos de I/O e particionamento de células para circuitos 3D. Desenvolveu-se uma heurística iterativa que obtém melhores resultados do que o partitionador de hipergrafos estado-da-arte chamado hMetis. O método demonstra que particionadores de hipergrafos não estão adaptados para trabalhar em aplicações com circuitos 3D, pois não entendem o arranjo estrutural dessa tecnologia, possibilitando o aumento de conexões verticais longas, aumento no número máximo de Vias-3D, área e número de vias-3D.

Demonstrou-se que o algoritmo proposto é capaz de reduzir significativamente o comprimento de vias-3D longas, através do quadrado do número de vias em cada rede adicionado à função de custo. Essas métricas desempenham um papel importante no projeto de blocos de lógica aleatória em circuitos 3D, afetando *timing, yield*, potência e área. A heurística proposta também é capaz de reduzir a área das *tiers* enquanto mantém os pinos de I/O equilibrados entre as camadas.

Resultados experimentais mostram que a medida que o algoritmo reduz as conexões verticais longas (aquelas vias 3D que cruzam mais de duas tiers adjacentes), ele aumenta o número de conexões curtas. Consequentemente, o número total de vias 3D também é reduzido. O algoritmo proposto nesse trabalho reduziu o número total de vias-3D em 22,80%, 17%, 13,60 e 19,50% e o número máximo de vias-3D entre um par adjacente de tiers em 22,80%, 34,27%, 11,75% e 15,04% para duas, três, quatro e cinco tiers respectivamente, em comparação com o algoritmo estado-da-arte hMetis.

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, Sandro ; Hentschke, Renato Fernandes ; Johann, M. O. ; Reis, Ricardo Augusto da Luz . An Algorithm for I/O Pins Partitioning Targeting 3D VLSI Integrated Circuits. In: IEEE International Midwest Symposium on Circuits and Systems, 2006, San Juan. Proceedings, 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] ISPD 2004 - IBM Standard Cell Benchmarks with Pads. http://www.public.iastate.edu/~nataraj/ISPD04_Bench.html#Benchmark_Description. Access on Mar 2008.

[12] 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.

[13] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, Science, 1983, 220, pages 671--680