Particionamento de Pinos de I/O e seu Impacto no Tamanho das Interconexões e Número de Vias em Circuitos VLSI 3D

Sandro Sawicki[1,2], Renato Hentschke[1], Marcelo Johann[1], Ricardo Reis[1]

[1]UFRGS – Universidade Federal do Rio Grande do Sul
PPGC - Instituto de Informática
Porto Alegre, Brasil, 91501-970
{sawicki, renato, johann, reis}@inf.ufrgs.br

[2]UNIJUI – Universidade Regional do Estado do Rio Grande do Sul
DETEC – Departamento de Tecnologia
Ijui, Brasil, 98700-000
sawicki@unijui.tche.br

Abstract

The 3D Circuit technologies appear as a possible solution for interconnect optimization. For most of the 3D technologies, the 3D-Vias represent a very complex issue because of large pitch requirements and heavy usage of routing constraints. This paper studies the impact of I/O pins partitioning in 3D circuits. Previous works on 3D placement did not focus on the I/Os partitioning and placement. This work presents an algorithm based on the logic proximity of the pins, which is used as weights to a min-cut partitioning. Our method calculates the area of the tiers while placing the I/Os on the boundaries. Initial whitespace and aspect ratio as well as the initial pins orientation and ordering are preserved. We compared to two other methods for pins partitioning. Our experimental results show that our method is efficient since it can balance the I/O pins distribution in the various tiers while leading to improvements in wire length and number of 3D vias.

Keywords: 3D VLSI Circuits, Placement, Partitioning, CAD.

Resumo

Circuitos 3D surgem como uma nova solução para a otimização das interconexões. Este artigo estuda o impacto do particionamento de pinos de I/O na minimização de vias-3D e tamanho das interconexões. Trabalhos anteriores envolvendo circuitos 3D não mostram como os pinos de I/O são migrados de circuitos 2D para circuitos 3D. Este trabalho apresenta um algoritmo baseado na proximidade lógica dos pinos utilizando pesos entre pares de pinos de I/O para obter o mínimo corte entre as camadas *(tiers)*. O método proposto calcula a área das *tiers* enquanto posiciona os pinos de I/O. O espaço em branco, a relação de aspecto, assim como a orientação dos pinos são preservados do *netlist* inicial. Este algoritmo foi comparado com dois outros métodos para particionamento de pinos. Os resultados obtidos mostram que o nosso método é eficiente e mantém a distribuição balanceada dos pinos entre as partições, minimizando o número de vias.

Palavras chave: Circuitos VLSI 3D, Posicionamento, Particionamento, CAD.

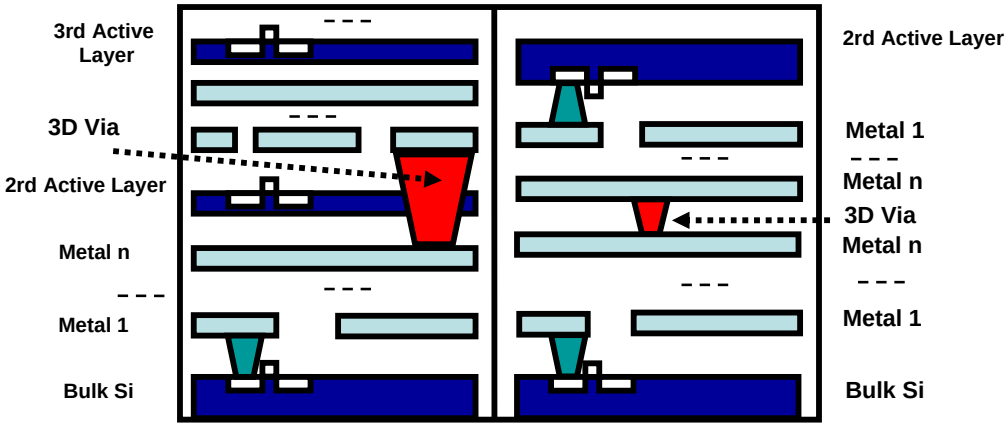
1. INTRODUCTION

É de senso comum que a otimização das interconexões é uma questão de extrema importância para desempenho dos circuitos integrados. Atraso, potência, ruído e crosstalk são algumas das questões relacionadas ao tamanho dos fios. Da mesma forma, a tecnologia envolvida no projeto de semicondutores avança rapidamente. Um reflexo dessa evolução pode ser percebido claramente através do crescimento do numero de elementos dentro de um chip. Assim, diversas questões elétricas que eram desprezadas em tecnologias anteriores começam a ser consideradas. Por exemplo, efeitos de uma conexão sobre outra, integridade do sinal, distribuição do consumo de potência, freqüência de relógio como forte limitador, indutância, etc. Essas questões reforçam ainda mais a necessidade da criação de novos algoritmos e metodologias. Nesse sentido, o uso de circuitos VLSI 3D é uma possibilidade real para melhorar a qualidade das

interconexões. Grandes empresas como IBM, Intel, Samsung, Micron, Cadence e Infineon estão investindo em soluções relacionadas a essa área [1].
A idéia de particionar um bloco de lógica aleatória em duas ou mais tiers (camadas) de um circuito 3D já foi explorado em [2-6]. Nessa abordagem, o estágio de posicionamento tem que particionar e posicionar células em tiers separadas. Teoricamente [7] e empiricamente [3] mostra-se que, potencialmente, circuitos 3D podem reduzir o tamanho dos fios.
Este trabalho assume que a fronteira de um bloco aleatório, em circuitos 2D, é delimitada por pinos de I/O. Também assume-se que os pinos de I/O podem ser movidos para qualquer tier. Muitos algoritmos de posicionamento de células são dirigidos pela localização fixa dos pinos. Algoritmos de posicionamento quadráticos [8], por exemplo, que são largamente usados na academia [9] e indústria [10] requerem que a posição dos pinos de I/O já esteja determinada para que seja computada a solução.
Este artigo estuda o problema de particionamento de pinos de I/O e seu impacto no número de 3D vias e tamanho dos fios (wirelenght). Sabe-se que o posicionamento de pinos de I/O é mais eficiente se executado durante a etapa de floorplanning. Por outro lado, atualmente, quase todos os projetos e ferramentas são direcionadas às tecnologias 2D. Percebe-se, contudo, que uma técnica automática para migrar de uma tecnologia 2D para 3D pode reduzir significativamente o tempo de projeto. Embora alguns trabalhos [2-6] possam ter usado alguns critérios para fazer o particionamento de pinos de I/O, os detalhes de como foi implementado foram omitidos. Assume-se, então, que eles adotam soluções simplistas para o tratamento dos pinos de I/O. No que tange o conhecimento dos autores, este é o primeiro trabalho que estuda o impacto na solução final do particionamento de pinos de I/O em duas ou mais tiers.
Este artigo é organizado da seguinte forma: A seção 1.1 relata questões relacionadas aos circuitos 3D. A seção 1.2 apresenta trabalhos que envolvem o posicionamento VLSI 3D. Na seção 2.1 define-se o problema relacionado com o particionamento e o posicionamento de pinos de I/O em circuitos 3D. A seção 2.2 apresenta um algoritmo para resolver o problema otimizando a lógica aproximada entre pinos no mesmo tier A última seção mostra os resultados experimentais e as conclusões deste trabalho.

1.1. Circuitos VLSI 3D

Circuitos 3D surgem como uma mudança no paradigma de projeto de circuitos integrados. Entre as vantagens do projeto 3D, pode-se destacar a facilidade de integração e a redução do tamanho das conexões [2]. A possibilidade de criação de projetos 3D só é possível devido ao avanço das tecnologias de fabricação e encapulamento, pois permite a comunicação entre diferentes circuitos. Cada circuito é chamado na literatura como tier. Referimos-nos às 3D-Vias como sendo os fios que conectam duas tiers diferentes.
Nesse sentido, existem duas estratégias de integração: face-to-back e face-to-face, como mostra a figura 1. Na estratégia face-to-back, os chips são empilhados um em cima do outro. No topo da última camada de metal do chip 1 (como no exemplo da figura 1 (a)) existe uma camada de isolante para que depois seja posicionado o bulk (substrato) do chip 2. Para a fabricação das vias 3D, deve-se abrir uma fenda no isolante e no bulk. 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 face2face os chips são empilhados de frente para o outro (como na figura 1 (b)).



(a) Face-to-Back (b) Face-to-Face

Figura 1. Métodos de Integração 3D Face-to-Back e Face-to-Face

A última camada de metal do chip 1 é colocada de frente com a última camada de metal do chip 2. (separado somente por um isolante). A via que conecta o chip 1 com o chip 2 é composta por uma área menor do que na estratégia face-to-

2back. Além disso, face-to-face necessita do dobro de camadas de metal para roteamento, agravando o problema de congestionamento. Por outro lado, a estratégia face-to-face claramente limita o número de camadas ativa empilhadas para 2. Face2back é mais fácil para fabricar, pois poucas mudanças nos processos tradicionais são realizadas.

1.2. Posicionamento de Circuitos 3D

Como dito anteriormente, um circuito 3D é construído através da união (empilhamento) de dois ou mais circuitos 2D separados por fios conhecidos como vias-3D [11]. O posicionamento 3D, além de posicionar células em espaços 2D, realiza o particionamento de células em diferentes camadas, como mostra a figura 2. Os espaços para vias 3D devem ser alocados previamente a fim de garantir que existam espaços suficientes para as conexões verticais. O posicionador 3D deve tirar vantagem desse espaço de posicionamento estendido para melhorar o tamanho das conexões, atraso, área e potência.

Goplen e Sapatnekar apresentam em [4] um posicionador de células dirigido a forças voltado aos problemas térmicos dos circuitos 3D. Questões térmicas são uma das grandes preocupações dessa área. Nesse trabalho utiliza-se o algoritmo baseado em bipartições recursivas encontrado na ferramenta hMetis [12]. Sua função é dividir as células entre os vários tiers enquanto as vias 3D são minimizadas.

Como nos trabalhos de Sapatnekar [3, 13], a abordagem de Obenaus [5] também apresenta um método de posicionamento direcionado a forças. Ele inicia com uma solução aleatória e melhora os resultados com base nas forças de atração. Ambos os trabalhos não mencionam como foi realizada a migração dos pinos de I/O de um arranjo original 2D para um arranjo 3D. Para seus experimentos foram utilizados Bechmarks padrão 2D. Assim, possivelmente, os pinos de I/O foram ignorados.

O trabalho de Deng [6] apresenta um posicionador 3D baseado na abordagem da ferramenta Capo. Seu fluxo de posicionamento é muito simples: a netlist é particionada em várias tiers. Após, cada tier é posicionada separadamente com informações das tiers já posicionadas. Essa metodologia tem como objetivo reduzir o tamanho das conexões 3D através das informações de tiers previamente posicionadas.

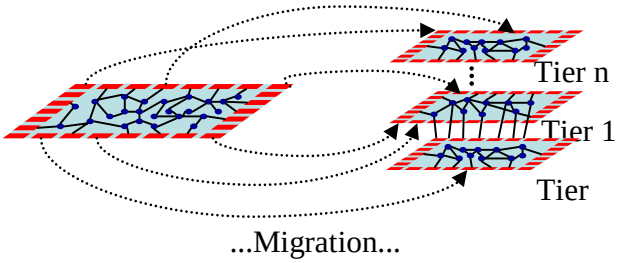


Figura 2. Idéia Geral dos Circuitos 3D com *n* Tiers

Inúmeros trabalhos relacionados a esse assunto concordam que número de conexões entre as tiers deve ser minimizado. Muitos deles utilizam o particionador hMetis [12, 14] para realizar essa tarefa. Além disso, é de conhecimento que as vias 3D são muito grandes, e o seu congestionamento pode ser melhorado com a sua minimização. Contudo, o particionamento de pinos de I/O pode influenciar indiretamente na qualidade do corte. Por essa razão, espera-se melhorar o número de vias e tamanho dos fios através de um algoritmo que considera o impacto do particionamento de pinos de I/O.

Este trabalho propõe um algoritmo baseado na distância lógica entre pinos de I/O como critério de particionamento. Resumindo a motivação deste artigo, deseja-se encontrar um bom método para o particionamento de pinos de I/O que seja capaz de manter um balanceamento equilibrado entre as tiers, minimizar o número de vias 3D e tamanho das conexões.

2. ALGORITMO PARA POSICIONAMENTO DE PINOS DE I/O

2.1. Definição do Problema

Antes do posicionamento, uma netlist 2D NL é composta por um conjunto de células G = {g1, g2, g3, … , gn}, um conjunto de pinos de I/O P = {p1, p2, p3, … , pm} e um conjunto de redes N = {n1, n2, n3, … , no}. Um hipergrafo HG representa a netlist, onde G∪P é o conjunto de nodos e N é o conjunto de hipergrafos. A posição fixa de cada pino de I/O pi é representada por X[i] e Y[i] (i≤m) e sua orientação por OR(pi) ∈ {north, south, east, west}. A área A (altura H largura W) tem seu canto inferior esquerdo na coordenada (xini ,yini). 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 (GA) pela área disponível dentro dos pinos de I/O. A relação de aspecto AR é computada pela divisão de W por H.
Se Z é o conjunto dos números que representam as tiers {1,2,…,z}. O problema pode ser definido como: dado uma netlist 2D NL com pinos de I/O fixos, encontre o conjunto de tiers T = {t1, t2, … , tz} e seus correspondentes Ai, ARi, GAi, Wi, Hi, Pi, Si, ORi, Xi and Yi (i≤z) tal que:

$$P_1 \cup P_2 \cup ... \cup P_t = P \quad (1)$$
$$\forall (a,b \in Z)(a \neq b \rightarrow P_a \cap P_b = \varnothing) \quad (2)$$
$$\forall (i \in Z) S_i \approx S \quad (3)$$
$$\forall (i \in Z)\forall (j \in Z) W_i = W_j \wedge H_i = H_j \quad (4)$$
$$\forall (i \in Z) AR_i \approx AR \quad (5)$$
$$\forall (i \in Z)(\forall a \in P_i (OR_i(a) = OR(a))) \quad (6)$$
$$\forall (t \in Z)(\forall a,b \in P_t (OR(a) = OR(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b])) \quad (7)$$
$$\forall (t \in Z)(\forall a,b \in P_t (OR(a) = OR(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b])) \quad (8)$$

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

2.21 Algoritmo Proposto Baseado no Menor Caminho Lógico
Seja LD(pi,pj) o tamanho do menor caminho em HG de pi para pj (por exemplo, a distância lógica entre pi e pj). Pode-se descrever o algoritmo de particionamento de pinos de I/O de forma detalhada abaixo:

1 Computar LD(i,j) $\forall i, j \in P$
2 Criar um grafo completo PG tal que P seja o conjunto de nodos e LD(i,j) (i,j ∈ P) seja o custo das aresta conectando os nodos i e j.
3 Executar o particionamento de PG em P1, P2, …, Pz buscando a minimização do corte (min-cut) e o balanceamento de pinos entre as partições.
4 $\forall (i \in Z) GA_i \approx \frac{GA}{z}$ (9)
5 $\forall (i \in Z) A_i = GA_i \times (1 + S_i)$ (10)
6 $\forall (i \in Z) W_i = \sqrt{A_i} \times AR_i; H_i = \frac{\sqrt{A_i}}{AR_i}$ (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)

Considerando que em um circuito real os fanouts são limitados. Então, uma simples busca BFS terá uma complexidade O(n). Portanto, usando uma busca simples para computar o custo de um pino pi para todos p ∈ P, a complexidade será O(mn).
Os valores de LD são usados para criar um grafo completo PG conectando todos os pares de pinos de I/O, como mostra a figura 3. No terceiro passo, usa-se a ferramenta hMetis [14] para particionar os pinos de I/O. Essa ferramenta aceita a inserção de 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 a quantidade de pinos de I/O similar entre as tier.
O quarto passo é realizado através da divisão do número total da área de células (gates) pelo número tiers. Os passos 5 e 6 computam a área das tiers, tal que a relação de aspecto (aspect ratio) e espaços em branco (whitespaces) sejam preservados do circuito 2D original. Neste ponto, a nova relação de aspecto ou espaços em branco pode ser usada.

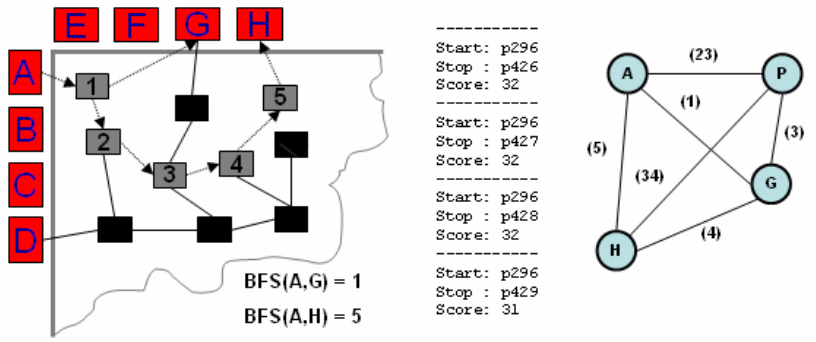


Figura 3. Ilustração do menor caminho entre dois pinos de I/O e seus pesos correspondentes no grafo completo.

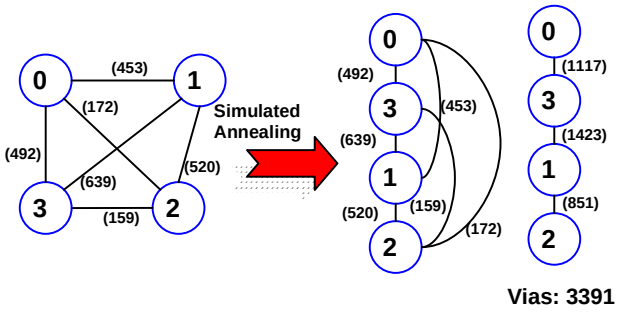


Figura 4. Ilustração do Problema da Seqüência de *Tiers* e Número de Vias

Finalmente, os passos 7 e 8, computam as coordenadas X e Y dos I/Os para suas tiers destino. A orientação original é preservada, assim os pinos de I/O originais 2D são 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.
Após o término do passo 9, é necessário encontrar qual a melhor seqüência de tiers que auxilie a minimização de vias. Utilizou-se Simulated Annealing para otimizar o numero total de vias 3D como mostra a figura 4 acima:

3. RESULTADOS EXPERIMENTAIS

O objetivo deste trabalho é estudar o impacto do particionamento de I/O na área, tamanho das conexões e número de vias. Os resultados realizados são descritos abaixo:

1. particionamento de HG é executado visando 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 foi realizado através da ferramenta hMetis. Essa ferramenta foi configurada para manter o balanceamento entre as áreas 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* [8] para 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 2.1. O primeiro método é chamado unlocked_pins. Neste método, permite-se que a partição dos pinos de I/O seja realizada livremente junto com as células do circuito. O segundo algoritmo é chamado de alternate_pins. Este método é um particionamento pseudo-aleatório que divide os pinos alternadamente em cada uma das tiers. A idéia é preservar o balanceamento inicial dos pinos de I/O. Ambos os algoritmos substituem os passos 1, 2,3 e 4 do fluxo, executando apenas os passos 5, 6, 7, 8 e 9.
Tabelas 1, 2 e 3 mostram os resultados experimentais com 2 tiers, usando os Benchmarks ISPD 2004 [15]. A coluna "área tier" é calculada antes da partição atual de células. A tier com maior área é usada como padrão para as demais tiers. A área total é simplesmente n vezes a área da maior tier. O WL total é a soma do wirelenght encontrado pelo posicionador em cada tier separadamente (não foram consideradas conexões entre tiers). O número de pinos de I/O e vias são relatados na tabela. A tabela também mostra o desvio padrão do número de pinos de I/O a fim de avaliar os balanceamentos. Analisando as tabelas 1,2 e 3 os seguintes tópicos podem ser observados:

- número de pinos de I/O é muito bem balanceado com o método alternate_pins (média do desvio padrão é menor do que 1). A média do desvio padrão do balanceamento do algoritmo proposto por esse trabalho é de 5 pinos. Contudo, o método **unlocked_pins apresenta um desbalanceamento no número de pinos que o invalida completamente** (média do desvio padrão de 150 pinos).
- O método alternate_pins é um pouco melhor em área (cerca de 1%) comparado com os outros métodos. Possivelmente, o bom balanceamento de pinos ajudou a ferramenta hMetis a encontrar um bom balanceamento de área.
- O número de vias encontradas pelo método alternate_pins é sempre pior do que os outros (média de 30 vias), 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 proposto é sempre melhor do que os outros métodos. Este é um resultado importante, pois mostra **que o método de particionamento de pinos de I/O auxilia o particionamento das células, encontrando o melhor corte e um balanceamento eficiente entre as tiers.** O resultado do tamanho dos fios obtido pelo algoritmo de particionamento de pinos é menor em média do que o método alternate_pins. O método unlocked_pins obteve o melhor wirelength.

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.
As tabelas 4 e 5 consideram experimentos com número de tiers de 2 a 5.
A tabela 4 apresenta a média dos resultados do desvio padrão do número de pinos de I/O em cada uma das 5 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, vários tiers ficaram sem nenhum pino de I/O. **Esse desbalanceamento invalida o método unlocked_pins**. O algoritmo proposto por esse trabalho se aproxima do balanceamento ótimo.
A tabela 5 apresenta os resultados experimentais comparando o número total de vias 3D usando os 3 algoritmos de particionamento com 5 tiers. As médias demonstram que **os métodos de particionamento de pinos de I/O simplistas não são eficientes na minimização do corte entre as tiers.** Contudo, percebe-se que as informações do menor caminho lógico entre os pinos de I/O melhoram a qualidade da heurística de minimização de vias 3D, mantendo o equilíbrio entre as tiers.

4. CONCLUSÕES

Este artigo apresentou um método para o particionamento e posicionamento de pinos de I/O de circuitos 2D para circuitos 3D. No que tange o conhecimento dos autores, este é o primeiro trabalho que foca este problema e estuda o seu impacto na área do circuito, balanceamento de pinos e tamanho das conexões. O trabalho propôs que o particionamento de pinos de I/O seja realizado antes do particionamento das células. Nesse trabalho, foi demonstrado empiricamente que fazendo o particionamento de pinos de I/O junto com as células cria um desbalanço no número de pinos entre as tiers, invalidando o método.
O método desenvolvido baseia-se na idéia de 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 melhorando o tamanho dos fios e o número de vias 3D comparado com as outras abordagens.
Esse algoritmo mantém um bom balanceamento de pinos de I/O entre as partições enquanto minimiza o número de vias através da heurística do menor caminho lógico. 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 um particionamento regular (células + pinos de I/O juntos) obteve-se um desbalanceamento enorme dos pinos de I/O. Conclui-se, então, que as informações do menor caminho lógico entre os pinos de I/O pode ajudar a heurística de particionamento de células a minimizar o número de vias 3D.

Referências

[1] 3D ICs Industry Sumary at Tezzaron homepage: http://www.tezzaron.com/ technology /3D%20IC%20Summary.htm . Access on Mar 2006.

[2] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer and P. D. Franzon; et. al. 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.

[3] 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.

[4] Brent Goplen; Sachin Sapatnekar; Efficient Thermal Placement of Standard Cells in 3D Ics usisng Forced Directed Approach. *In: Internation Conference on Computer Aided Design*, ICCAD'03, November, San Jose, California, USA, 2003.

[5] S. Obenaus, T. Szymanski. Gravity: Fast Placement for 3D VLSI. *ACM Transacions on Design Automation of Electronic Systems*, New York, v.8, p.69–79, March 1999.

[6] Y Deng; W. Maly. Interconnect Characteristics of 2.5-D System Integration Scheme. *In: Proc. of the International Symposium on Physical Design*, ISPD 2001, New York, NY, USA. Anais. . . ACM Press, 2001. p.171– 175.

[7] 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.

[8] C. J Alpert; T. Chan; D. J. Huang.; I. Markov; K. Yan. Quadratic placement revisited. *In: Proc. of the 34th Annual Conference on Design Automation*, DAC 1997, New York, NY, USA. Anais. . . ACM Press, 1997. p.752–757.

[9] N. Viswanathan; C.C.-N Chu. FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement,and a Hybrid Net Model. *IEEE Transactions on CAD*, Volume 24, Issue 5, pp 722-733, May 2005.

[10] P. Villarrubia, "CPLACE: A standard cell placement program" *IBM Tech. Dis. Bull.*,vol32 no. 10A, pp. 341-342, Mar. 1990.

[11] P. Benkart, A. Heittmann, H. Huebner, U. Ramacher, A. Kaiser, A. Munding, M. Bschorr, H-J Pfleiderer, E. Kohn. 3D Chip Stack Technology Using Through-Chip Interconnects. *IEEE Design and Test of Computers – special issue on 3D integration*; pp 512-517, Nov.-Dec. 2005.

[12] 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.

[13] S. Spatnekar and K. Nowka; New Dimensions in 3D Integration; *In: IEEE Design & Test of Computers*; *– special issue on 3D integration*; pp 496-497, Nov.-Dec. 2005.

[14] Hypergraph & Circuit Partitioning at hMetis Home Page, http://glaros.dtc.umn.edu/gkhome/views/metis/hmetis/. Access on Mar 2006.

[15] ISPD04 - IBM Standard Cell Benchmarks with Pads. http://www.public.iastate.edu/~nataraj/ISPD04_Bench.html#Benchmark_Description. Access on Mar 2006.

Tabela 1 – Resultados experimentais usando o algoritmo proposto em 2 tiers.

Dados do Circuito					Dados Particionados					
Bench	**#cells**	**#I/Os**	**#nets**	**2D Área**	**Tier Area**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/O**	**#Vias**
ibm01	12,506	246	14,111	2,380,800	1,209,856	2.11E+06	120	126	4	393
ibm02	19,342	259	19,584	3,064,208	1,517,568	4.50E+06	126	133	5	477
ibm03	22,853	283	27,401	3,751,968	1,896,128	6.48E+06	138	145	5	1,103
ibm04	27,220	287	31,970	4,782,848	2,417,664	7.02E+06	147	140	5	733
ibm06	32,332	166	34,826	4,106,592	2,038,784	7.51E+06	81	85	3	1,059
ibm07	45,639	287	48,117	7,136,672	3,612,960	1.23E+07	140	147	5	1,032
ibm08	51,023	286	50,513	7,403,840	3,699,840	1.07E+07	140	146	4	1,297
ibm09	53,110	285	60,902	8,617,104	4,328,208	1.41E+07	139	146	5	778
Avg.	33,002	264	35,928	5,155,504	2,590,126	8.08E+06	129	134	5	859

Tabela 2 – Resultados experimentais usando o algoritmo unlocked_pins em 2 tiers.

Dados do Circuito					Dados Particionados					
Bench	**#cells**	**#I/Os**	**#nets**	**2D Área**	**Tier Area**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/O**	**#Vias**
ibm01	12,506	246	14,111	2,380,800	1,209,856	2.14E+06	0	246	174	539
ibm02	19,342	259	19,584	3,064,208	1,518,784	4.39E+06	259	0	183	477
ibm03	22,853	283	27,401	3,751,968	1,867,280	6.22E+06	283	0	200	1,109
ibm04	27,220	287	31,970	4,782,848	2,414,592	7.30E+06	287	0	203	748
ibm06	32,332	166	34,826	4,106,592	2,038,784	7.73E+06	75	91	11	1,062
ibm07	45,639	287	48,117	7,136,672	3,596,112	1.13E+07	0	287	203	1,037
ibm08	51,023	286	50,513	7,403,840	3,697,920	1.03E+07	127	159	23	1,303
ibm09	53,110	285	60,902	8,617,104	4,326,144	1.40E+07	0	285	202	778
Avg.	33,002	264	35,928	5,155,504	2,583,684	7.91E+06	129	134	150	882

Tabela 3 – Resultados experimentais usando o algoritmo alternate_pins em 2 tiers.

Dados do Circuito					Dados Particionados					
Bench	**#cells**	**#I/Os**	**#nets**	**2D Área**	**Tier Area**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/O**	**#Vias**
ibm01	12,506	246	14,111	2,380,800	1,182,416	2.35E+06	123	123	0	429
ibm02	19,342	259	19,584	3,064,208	1,517,568	4.30E+06	130	129	1	477
ibm03	22,853	283	27,401	3,751,968	1,865,920	6.13E+06	142	141	1	1,117
ibm04	27,220	287	31,970	4,782,848	2,375,760	7.54E+06	144	143	1	751
ibm06	32,332	166	34,826	4,106,592	2,080,464	7.37E+06	83	83	0	1,132
ibm07	45,639	287	48,117	7,136,672	3,588,624	1.18E+07	144	143	1	1,065
ibm08	51,023	286	50,513	7,403,840	3,697,920	1.11E+07	143	143	0	1,301
ibm09	53,110	285	60,902	8,617,104	4,307,568	1.43E+07	143	142	1	787
Avg.	33,002	264	35,928	5,155,504	2,577,030	8.10E+06	132	131	0	882

Tabela 4 – Comparação da Distribuição dos pinos de I/O considerando os três Algoritmos.

.#Tiers	Algoritmos	σ #I/Os
2	our algorithm	5
	unlocked_pins	150
	alternate_pins	0,44
3	our algorithm	4
	unlocked_pins	141
	alternate_pins	0,43
4	our algorithm	3
	unlocked_pins	103
	alternate_pins	0,53
5	our_algorithm	4
	unlocked_pins	112
	alternate_pins	0,43

Tabela 5 – Comparação do número total de vias 3D com unlocked_pins, alternate_pins e o algoritmo proposto utilizando 5 *tiers*.

	***unlocked_pins* #vias**				***alternate_pins* #vias**				***Our Algorithm* #vias**			
	2 Tiers	**3 tiers**	**4 tiers**	**5 tiers**	**2 tiers**	**3 tiers**	**4 tiers**	**5 tiers**	**2 tiers**	**3 tiers**	**4 tiers**	**5 tiers**
ibm01	539	785	1,157	1,327	429	705	1,067	1,517	393	577	945	1,278
ibm02	477	1,009	1,567	*1,960*	477	858	1,682	2,085	477	851	1,365	***2,052***
ibm03	1,109	2,698	***3,307***	***5,241***	1,117	2,874	***3,347***	5,844	1,103	2,473	***3,391***	***5,257***
ibm04	748	1,598	3,057	3,651	751	***1,639***	3,067	3,729	733	***1,720***	2,955	3,046
ibm06	1,062	2,130	4,854	6,211	1,132	2,074	4,518	6,105	1,059	2,100	4,544	6,031
ibm07	1,037	2,093	3,875	5,655	1,065	2,229	4,353	5,771	1,032	2,286	3,960	***5,755***
ibm08	1,303	3,336	***5,394***	6,935	1,301	3,413	5,513	6,949	1,297	3,241	5,407	6,849
ibm09	778	1,960	3,228	***4,500***	787	2,068	3,128	***4,702***	778	1,853	3,103	***4,769***
Average	882	1,951	3,305	4,435	882	1,983	3,334	4,588	859	1,888	3,209	4,455