An Algorithm for I/O Partitioning Targeting 3D Circuits and Its Impact on 3D-Vias

Renato Hentschke renato@inf.ufrgs.br
Sandro Sawicki sawicki@inf.ufrgs.br
Marcelo Johann johann@inf.ufrgs.br
Ricardo Reis reis@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul – Instituto de Informática
Av. Bento Gonçalves, 9500
CEP 91501-970 - Porto Alegre - RS – Brazil

***Abstract*— In this paper we discuss the migration of a 2D netlist with pre-placed I/Os to 3D circuits. For that, we present an algorithm to perform the partitioning of the I/O pins into various tiers targeting at I/O balancing and 3D-Vias minimization. We formulate the netlist migration constrained with respect to the preservation of the original netlist properties. The I/O partitioning algorithm is based on the logic distance between I/Os. Since there is no literature on I/O partitioning for 3D circuits we compared our algorithm with two simplistic approaches that targeted balance and min-cut respectively. Experimental results show that our algorithm can reduce the number of 3D-Vias compared to both algorithms, while balance is kept close to optimal. Additionally, we studied the area impact of the 3D-Vias resulted by the three algorithms targeting two different technologies for 3D circuits. We observed that in Bulk based technologies the 3D-Via penalty is huge, favoring our algorithm that minimizes the number of 3D-Vias. Targeting SOI technologies, the area impact is very small, leading to the conclusion that there is no need for deep 3D-Via minimization. Our I/O partitioning is still recommended in such case while the cell partitioning should consider wire length instead of 3D-Via minimization.**

I. INTRODUCTION

In the nanometer VLSI, the technology shrinking imposes many challenges to the design of circuit interconnect instead of providing shorter wire lengths. Issues like delay, variability and manufacturability are highly valuable research subjects in the present days.

3D circuits appear as a change of design paradigm, providing higher integration and reducing wire lengths [6]. From the CAD perspective, there is a huge research space for the development of new algorithms and new methodologies to take full advantage of the 3D integration.

Among the new problems and algorithms to be migrated from 2D to 3D, cell placement is a key problem for interconnect optimization [14]. A block of standard cells can be partitioned into several tiers to achieve a better wire length [1, 4, 6, and 12] and delay [2]. Compared to 2D placement, the problem has other aspects to be considered, such as:

- The third dimension (*Z*) disposes the cells into different tiers. A Standard-Cell block needs to be partitioned in a set of different sub-blocks. It is reasonable, though, to constrain the width and height of the whole block by the largest sub-block. For this reason, the Z coordinate spreading of the cells should be constrained by a tight equilibrium requirement in order to minimize the area.
- Thermal issues become critical due to higher integration and hard heat dissipation [6, 14]. Thermal vias [9] and thermal driven floorplanning [5] and placement [1, 8] are possible solutions being used.
- Communication between different tiers is accomplished by 3D-Vias. There are various possible technologies for 3D-Via fabrication, depending on the disposal of the various tiers as face-to-face or face-to-back and on the bulk technology (in contrast with SOI) [6]. The 3D-Vias imposes challenges such as high consumption of routing resources, obstacles to placement and large pitch.

Because of the high penalties imposed by 3D-Vias, a common approach in the placement phase is to minimize them by using min-cut partitioning. In [1] the min-cut partitioning algorithm hMetis [13] is used to partition the cells into different tiers. Force Directed Placement and Simulated Annealing algorithms follow to improve wire lengths. In [2], a partitioning based placement algorithm using hMetis also minimizes the number of 3D Vias. The work in [7] presents a similar approach using the Capo placer to place each tier separately (with terminal propagation from other tiers) proceeded by min-cut partitioning for tier assignment.

The existing algorithms for 3D placement found in the literature are natural extensions of their 2D version. Most of the reviewed works use iterative Force Directed approach

and partitioning based methods. The quadratic placement algorithm [3] is widely used by the leading industry [15] and academia [16] because it is very fast and scalable. However, there seems to be no work with the quadratic placement algorithm for 3D circuits. Different from other approaches, this algorithm requires I/O pins in order to compute a solution. A 3D extension of this algorithm would require I/O pins to be previously placed in 3D as well.

The I/O pins plays two important roles in the placement of a block: first, the area boundary is limited by the I/Os; second, the pins are used as tips for the placement algorithm to reduce wire lengths. The same facts are needed for 3D placement. It is known that I/O pins tier-assignment and placement can be more effective if performed during floorplanning. On the other hand, an automatic migration algorithm at the placement level could facilitate the decision or can be plugged in an automated flow of a front-end synthesis that targets 2D.

This paper studies the partitioning of I/O pins into various tiers and the impact of different algorithms in the number of 3D-Vias and their area requirements. To our knowledge, **we are the first to study this problem and to propose a reasonable solution**. In section II, we present a brief introduction to the 3D Circuits Placement focusing on 3D-Via constraints, reinforcing the importance to minimize them. Section III presents our definition of the 3D I/O pin partitioning. We assume that the I/Os are placed on the boundary and that can be moved to any tier. We formulate the problem to **evenly distributed whitespace and the gates area while aspect ratio is preserved**. We them propose an algorithm on section IV with two objectives: first**, to balance the number of pins** in each tier in order to shrink down the area of blocks bounded by the I/Os; second, to provide a smart starting point partitioning solution of the netlist that will effectively **reduce the number of 3D-Vias**. Section V presents our experimental results with circuits placed in various tiers while also studying the impact on the requirements of 3D-Vias. Section VI discusses our conclusions with respect to the number of 3D-Vias. It is shown how they are affected by the I/O partitioning algorithm and how they impact the area and obstacles constraints on the circuit, leaving room for future work on 3D-Via planning.

II. 3D Circuits Placement

A. *Introduction to 3D Circuits*

A 3D circuit is actually the stacking of regular 2D circuits. The advances on the fabrication and packaging technologies allows to interconnect different 2D stacked circuits. Each circuit is named in the literature as a **tier**. We refer to **3D-Vias** as the piece of wire that connects two different tiers. According to [6] there are several types of 3D-Vias. The so called **thought vias** organizes the circuit in face-to-back tiers while the 3D-Vias dig a hole in the bulk. The highest possible via density is allowed by SOI technologies (5 μm pitch) while Bulk-based technologies present a smaller density compared to SOI (50 μm pitch). This fact and the circuit organization are illustrated in figure 1.

It is quite clear in figure 1 that 3D Vias can impose significant obstacles and constraints to the 3D placement problem. Most of the existing approaches, such as [1, 7, and 8] completely ignore this fact on placement, but they do optimize the number of vias with min-cut partitioning. The via minimization and the via impact on the final area seeing from I/O pins perspective are studied in this paper.

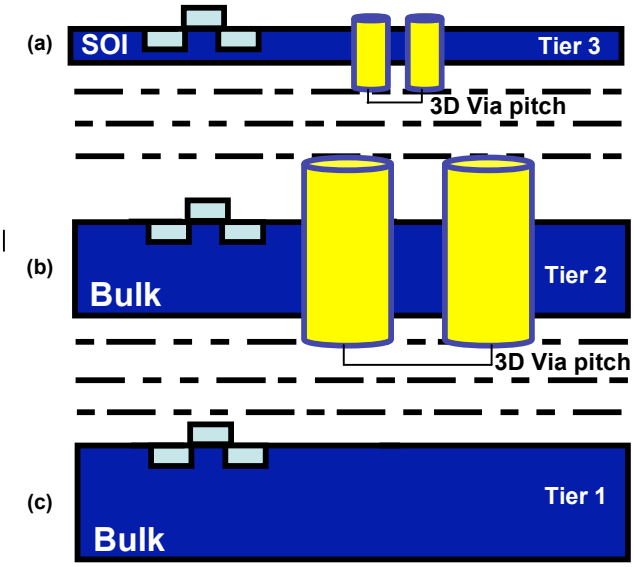

Figure 1. 3D Circuit organization and different types of 3D –Vias

B. *Migrating from 2D to 3D Placement*

Given a 2D placement netlist with pre-placed I/O pins in the boundary of the region available for standard cell placement, the migration to a 3D netlist (ready for 3D placement) has the following tasks:

- Area allocation: the width and height of the tiers will be calculated according to the number of tiers.
- I/O partitioning: the I/Os must be partitioned in different tiers.
- I/O placement: the I/Os must be placed in the boundary of the block, delimiting the area for standard cell placement.

We classify the cell partitioning as a placement task. Figure 2 illustrates the I/O pins migration as well as a possible cell partitioning.

As formulated in the next section, the netlist migration preserves some properties of the 2D solution, such as whitespace, aspect ratio, I/O pins orientation and ordering. Our objective is **to provide a migration algorithm that facilitates the 3D-Via minimization.** From the perspective of the I/O pin partitioning it is as good starting point for the cell partitioning. The algorithm should provide good I/O pins balance and respecting the mentioned properties.

Once the netlist is migrated (the I/O pins are placed) a partitioning process can distribute the cells on the tiers. This methodology aims at 3D-Via minimization since the layer assignment is done before placement by min-cut partitioning. In this paper, we propose to study the impact of the 3D-Vias in the tier area and we will investigate further whether 3D-Via minimization is in fact a good methodology for 3D Placement.

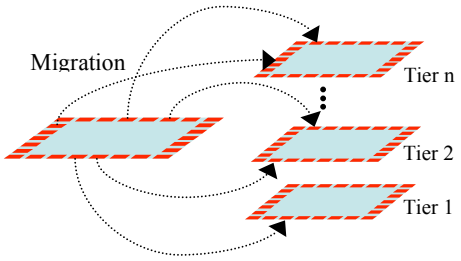

Figure 2. Migration (from 2D to 3D) of a netlist with pre-placed I/O Pins

III. 3D I/O PARTITIONING PROBLEM DEFINITION

Before placement, a 2D circuit netlist *NL* is composed by a set of gates $G = \{g_1, g_2, g_3, \ldots, g_n\}$, a set of I/O pins $P = \{p_1, p_2, p_3, \ldots, p_m\}$ and a set of nets connecting them $N = \{n_1, n_2, n_3, \ldots, n_o\}$. A hypergraph *HG* represents the netlist, where $G \cup P$ is the set of nodes and N is the set of hyperedges. The fixed position of each I/O pin p_i is given by $X[i]$ and $Y[i]$ ($i \leq m$) and its orientation by $OR(p_i) \in \{north, south, east, west\}$. The area A (height H and width W having its bottom left corner at coordinate (x_{ini} ,y_{ini}) position) inside the I/O pins is assigned for cell placement. Usually, I/O pin positions covers the entire boundary, leaving no room for additional connections or area reduction. The whitespace ratio S on the placement area is achieved by subtracting the total gate area (GA) from the area available inside the I/Os normalized by GA. The aspect ratio AR is computed by W divided by H.

Let Z be the set of tier numbers $\{1,2,\ldots,z\}$. netlist migration is defined as follows: given a 2D placement netlist *NL* with fixed I/O pins, find a set of tiers $T = \{t_1, t_2, \ldots, t_z\}$ (z is the number of tiers) and their correspondent A_i, AR_i, GA_i, W_i, H_i, P_i, S_i, OR_i, X_i and Y_i ($i \leq z$The following constraints should be met:

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

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

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

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

$$\forall (i \in Z) \forall (j \in Z) GA_i \approx GA_j \quad (5)$$

$$\forall (i \in Z) AR_i \approx AR \quad (6)$$

$$\forall (i \in Z)(\forall a \in P_i(OR_i(a) = OR(a))) \quad (7)$$

$$\forall (t \in Z)(\forall a,b \in P_i(OR(a) = OR(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b])) \quad (8)$$

$$\forall (t \in Z)(\forall a,b \in P_i(OR(a) = OR(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b])) \quad (9)$$

IV. THE ALGORITHM FOR I/O PARTITIONING

Our partitioning algorithm is a heuristic to be combined with existing min-cut partitioning approaches. We perform the I/O partition in two steps: first, a complete graph of the I/O pins is created with costs associated to each edge; second, a min-cut partitioning is performed considering the calculated costs. The following steps of the algorithm will calculate the area of each tier and the consequent I/O placement.

Let $LD(p_i,p_j)$ be the length of the shortest path in *HG* from p_i to p_j (i.e. the logic distance between p_i and p_j). Our idea is to keep the closer I/Os in the same tier. Accomplishing that, we intuitively expect that the cell partitioning will be able to minimize further the number of vias, while we can control the number of I/Os in each tier.

The algorithm for I/O partitioning is described as follows:

1) **Compute $LD(p_i,p_j)$ for every pair of pins p_i and $p_j \in P$.** This step is illustrated in figure 3. It can be accomplished with m BFS searches from every pin p_i to the whole graph, resulting in a $O(mn)$ complexity.

2) **Create a complete graph *PG* such that *P* is the set of nodes and $LD(p_i,p_j)$ (p_i and $p_j \in P$) is the cost of the edge connecting nodes *i* and *j*.** This step is illustrated in figure 3.

3) **Perform the partitioning of *PG* into z partitions $PART_i$ ($i \leq z$) aiming at weighted min-cut optimization**. We used hMetis tool [10, 13] as it accepts weighted graph imposing a tight balance.

4) **Perform the min-cut cell partitioning of the graph *HG* into the sets $PART_i$ ($i \leq z$)** in order to estimate the minimum number of 3D-Vias between the partitions. Note that area balance should be respected according to equation (5).

5) **Perform the tier-assignment problem from each $PART_i$ to P_1, P_2, …, P_z** . This step aims at via minimization and can be formulated as a single dimension placement problem, as illustrated in figure 4. We used a Simulated Annealing optimization to minimize the total number of 3D-Vias.

6) For each pair of adjacent tiers i and j ($i < j$), there is a 3D-Via layer V_i containing NV_i 3D-Vias and a total area of VA_i. **Compute VA_i for all $i<z$** by multiplying the NV_i with the 3D-Via minimum pitch according to the target technology.

7) **Compute GA_i for all tiers** using the cell partitioning performed at step 4.

6) **Compute the area A_i of each tier *i (i>0)*** by adding GA_i, VA_{i-1} and the whitespace S_i. For the first tier ($i=0$) we do not consider the 3D-Via area. The largest tier area is taken for all tiers, according to the restriction of equation (4). At this point, we observe that VA_i should be much smaller than GA_i to not unbalance the area of the tiers. Whitespace could be used to compensate the unbalance.

7) The width W_i and height H_i of each tier i is calculated according to equation (10).

8) The x and y coordinates of every pin p are calculated according to equations (11) and (12).

9) Legalize I/O pin positions.

$$W_i = \sqrt{A_i} \times AR_i \qquad H_i = \frac{\sqrt{A_i}}{AR_i} \tag{10}$$

$$X_i[p] = x_{ini} + \frac{(X[p] - x_{ini}) \times W_i}{W} \tag{11}$$

$$Y_i[p] = y_{ini} + \frac{(Y[p] - y_{ini}) \times H_i}{H} \tag{12}$$

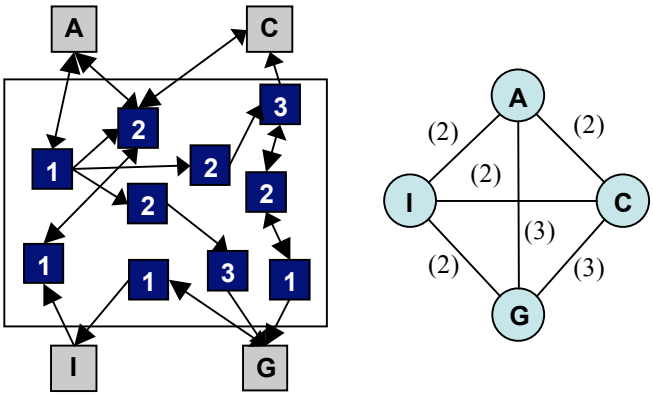

Figure 3. An ilustration of the steps 1 and 2 of our algorithm

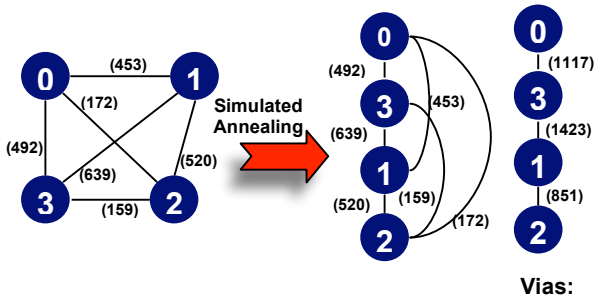

Figure 4. An ilustration of the tier-assignment problem and the consequent number of 3D-Vias

V. EXPERIMENTAL RESULTS

Our goal is to study the impact of the I/O partitioning in the number of vias and consequently the area requirements with respect to the 3D-Vias. Two types of 3D CMOS technology are considered: regular bulk and SOI, according to the data presented in [6]. Our experiments are summarized in three tables: table 1 summarizes our data on I/O pins balancing; table 2 presents the resultant number of 3D-Vias comparing with other I/O partitioning algorithms; table 3 presents the area requirements analysis that we made with respect to the 3D-Vias constraints. All tables are based on the ISPD 2004 benchmark set [11].

As we cannot find any previous work for I/O partitioning, we assume that simplistic approaches are being adopted in the existing 3D placement literature. We compared our I/O partitioning algorithm with two other simplistic algorithms that follow the same formulation described in section III. The first method is called *unlocked_pins*. In this method, we allow hMetis to partition the I/Os as free nodes, replacing the steps 1, 2 and 3 of our algorithm. The following steps of our algorithm are done for the *unlocked_pins* as well. The second algorithm is called *alternate_pins*. This method is a pseudo-random partitioning that goes thought the boundary line of the chip picking nodes for each partition alternatively. The idea is to preserve the initial I/O balanced distribution. Just as for *unlocked_pins*, the *alternate_pins* replaces steps 1,2 and 3 our flow, but steps 5-9 are done. The *unlocked_pins* cannot control the I/O balancing, but the heuristic for min-cut partitioning should reach a good solution in terms of number of 3D-Vias. On the other hand, the *alternate_pins* has complete control over-constrain the balancing, but may constraint the cell partitioning.

TABLE I. COMPARISON OF THE I/O PINS DISTRIBUTION IN THE TIERS CONSIDERING THE THREE ALGORITHMS

#Tiers	Algorithm	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

Table 1 presents the average results on the standard deviation of the number of I/O pins. The *alternate_pins* algorithm is optimal in this metric and obviously has the best I/O balance. The method *unlocked_pins* has a very large standard deviation; in many cases, several tiers had no pins. **The strong unbalance of the I/Os, especially in the case of**

few tiers, invalidates the *unlocked_pins* method. Our algorithm has a close to optimal pin balancing.

Table 2 presents our experimental results looking for a comparison of the total number of 3D-Vias using the three partitioning algorithms. The average numbers show that the *alternate_pins* algorithm has the worst results leading to the conclusion that **a simplistic I/O pin partitioning method over-constraints the cell partitioning that follows the I/O partitioning, resulting in increased cut size.** Additionally table 1 shows that, in average, our algorithm outperforms the *unlocked_pins* as well, leading to the conclusion that **the two phase partitioning process with minimum path information of the I/Os actually improves the min-cut heuristic quality,** while keeping the I/O balance.

Table 3 presents the area impact study of the 3D-Vias considering the three algorithms (the numbers are averaged for all benchmarks). The column "Max #3D-Vias" reports the maximum NV_i ($i < z$) that represents the maximum number of 3D-Vias connecting two tiers. This number will impact the area requirements for the 3D-Vias. The table shows that **our algorithm outperforms the others in 3D vias minimization, except for the four tiers case**. The following columns report the area impact targeting a Bulk-based 3D technology and finally an SOI technology, based on the information presented in [6]. However, there is no information of the actual width and height of the 3D-Vias. For this reason, we are taking the pitch as the 3D-Vias dimensions. Note that the 3D-Via area requirements reflects the circuit area in two manners: in the active layer, the places for an arising 3D connection are going to be constrained by the actual width and height of the 3D-Via; on the other hand, the 3D-Via layer (between two tiers) is going to be constrained by the pitch of the 3D-Via, as we measured. Additionally, we observe that the pitch used is a measure for a 0.35μm technology, according to [6], while we don't know the source technology of our benchmarks. Our goal is not to provide accurate comparison numbers, but to have an idea of the requirements for the 3D-Vias.

Analyzing the data from table 3 we make the following considerations:

- The Bulk-based 3D technologies suffer from a very high penalty for the 3D-Vias. With 2 tiers, there is a penalty around 85% of the tier area (note that our algorithm results in less vias and also less tier area than the others). For the cases with 3 to 5 tiers, the 3D-Via area is bigger than the tier area! Our algorithm could save up to 10% of 3D-Via area compared to the tier area. The important conclusion here is that when **targeting a CMOS Bulk based technology it is mandatory to minimize the number of 3D-Vias to obtain a feasible solution.**
- The SOI based technology suffers around 2% area penalty related to 3D-Vias, which is actually small**, leaving room for more 3D-Vias if they are helpful**.

VI. CONCLUSIONS

In this paper we presented an algorithm for the partitioning of I/Os targeting 3D circuits. This is a necessary step for the physical synthesis of 3D circuits for two reasons: to alleviate the boundary of the block, leaving room for area reduction; to perform 3D placement with the quadratic placement algorithm [3].

Our algorithm has a good balancing on the number of I/Os per partition while it targets the minimization of the cut with a shortest-path heuristic. According to our experimental results, a simplistic I/O pin partitioning method will lead to larger amount of 3D-Vias. At the same time, by using a regular min-cut partitioning of the whole netlist (cells + I/O pins together) we got very unbalanced number of I/Os.

Our partitioning approach is done in two phases: first the I/O partitioning considering the whole netlist as weights; second, we fix the I/Os and perform partitioning of the cells. The experimental results also show that our two phase partitioning led to a better cut, in average, than the single phase partitioning. We conclude that the shortest-path information could actually improve the partitioning algorithm since it could work with smaller graphs containing information of the whole circuit.

Finally, we studied the area impact of the 3D-Vias targeting two types of 3D technologies. First, we considered the Bulk based technologies in which the 3D-Via area requirement is very big. We observed that there is a huge penalty associated with the 3D-Vias, making its minimization extremely necessary. In such cases, our algorithm gives acceptable results. For the SOI technologies, we observe that there is no need for a minimization, as the area impact is close to only 2%. The partitioning criteria should be wire length and/or timing, fact that is not being considered by most of the existing works on 3D placement available in the literature. Anyway, our algorithm is still recommended because the optimized I/O partitioning can facilitate the wire length minimization. This fact is going to be investigated in future work.

REFERENCES

[1] 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.

[2] C. Ababei; H. Mogal; K. Bazargan; Three-Dimensional Place and Route for FPGAs. *In:* Proceedings of the Design Automation Conference - Asia and South Pacific, ASP-DAC 2005. Volume: 2 18-21 Jan. 2005. Page(s): 773- 778 Vol. 2.

[3] C. 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.

[4] 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.

[5] J. Cong, J. Wei, Y Zhang. A Thermal-Driven Floorplanning Algorithm for 3D ICs. *Proceedings of the International Conference on Computer-Aided Design,* ICCAD 2004.

[6] 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.

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

[8] B. Goplen; S. Sapatnekar; Efficient Thermal Placement of Standard Cells in 3D ICs usisng Forced Directed Approach. *In:Proceedings of the Internation Conference on Computer Aided Design*, ICCAD 2003, November, San Jose, California, USA, 2003.

[9] B. Goplen, S. Sapatnekar. Thermal Via Placement in 3D ICs. *Proceedings of the Iinternational Symposium on Physical Design,* ISPD 2005.

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

[11] ISPD 2004 - IBM Standard Cell Benchmarks with Pads. http:// www.public.iastate.edu/~nataraj/ISPD04_Bench.html#Benchmark_Description. Access on Mar 2006.

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

[13] 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.

[14] 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.

[15] P. Villarrubia, "CPLACE: A Standard Cell Placement program" IBM *Tech. Dis. Bull*.,vol32 no. 10A, pp. 341-342, Mar. 1990.

[16] 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.

TABLE II. COMPARISON OF THE TOTAL NUMBER OF 3D VIAS IN UNLOCKED_PINS, ALTERNATE AND OUR ALGORITHM

	***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

TABLE III. COMPARISON OF THE 3D-VIAS AREA IMPACT CONSIDERING THE THREE ALGORITHMS

#tiers	**Algorithm**	**Area Tier** $max(GA_i + S_i)$	**Max #3D-Vias**	**Area 3D-Vias (Bulk)**		**Area 3D-Vias (SOI)**	
2	OUR ALGORITHM	2,341,829	859	2,147,500	92%	21,475	1%
3		1,757,626	1,045	2,612,500	149%	26,125	2%
4		1,299,808	1,335	3,337,500	257%	33,375	2%
5		1,046,700	1,422	3,555,000	340%	35,550	4%
2	UNLOCKED_PINS	2,583,684	882	2,205,000	85%	22,050	1%
3		1,728,396	1,054	2,635,000	152%	26,350	2%
4		1,295,760	1,281	3,202,500	247%	32,025	2%
5		1,047,158	1,449	3,622,500	346%	36,225	3%
2	ALTERNATE_PINS	2,577,030	882	2,205,000	86%	22,050	1%
3		1,741,360	1,106	2,765,000	159%	27,650	2%
4		1,294,344	1,292	3,230,000	250%	32,300	2%
5		1,049,190	1,457	3,642,500	347%	36,425	3%