A Cells and I/O Pins Partitioning Refinement Algorithm for 3D VLSI Circuits

Sandro Sawicki[1, 2] sawicki@inf.ufrgs.br
Gustavo Wilke[1] wilke@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 — **Partitioning algorithms are responsible for the assignment of the random logic blocks 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 cells partitioning for the automatic synthesis of 3D designs it has 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 five tiers a connection between the top and the bottom tiers would require four 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. Our algorithm minimizes the total number of 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.**

I. INTRODUCTION

The design of 3D circuits is becoming a reality in the VLSI industry and academia. While the most recent manufacturing technologies introduces many wire related issues due to process shrinking (such as signal integrity, power, delay and manufacturability), the 3D technology seems to significantly aid the reduction of wire lengths [1-3] consequently reducing these problems. However, the 3D technology also introduces its own issues. One of them is the thermal dissipation problem, which is well studied at the floorplanning level [4] as well as in placement level [3]. Another important issue introduced by 3D circuits is how to address the insertion of the inter-tier communication mechanism, called 3D-Via, since it introduces significant limitations to 3D VLSI design. This problem has not been properly addressed so far since there are some aspects of the 3D via insertion problem that seem to be ignored by the literature: 1) all face-to-back integration of tiers imply that the communication elements occupy active area, limiting the placement of active cells/blocks; 2) the 3D-Via density is considerably small compared to regular vias, and it will not allow to place as many vertical connections EDA tools usually demand; 3) timing of those elements can be bad specially considering that a vertical connection needs to cross all metal layers in order to get to the other tier ; 4) 3D-Vias impose yield and electrical problems not only because of their recent and complex manufacture process but also because they consume extra routing resources.

The 3D integration can happen in many granularity levels, ranging from transistor level to core level. While core level and block level integration are well accepted in the literature, there seem to exist some resistance to the idea of placing cells in 3D [6]. One of the reasons is that finer granularity demands higher 3D-Via demand, which might fail to meet the physical constraints imposed by them. On the other hand, the evolution of the via size is going fast and is already viable (for most designs) to perform such integration [2, 5, 11, 13] since we already can build 0.5 μm pitch face-to-face vias [6] and 2.4 μm pitch on face-to-back [5]; we believe that this limitation is more in the design tools side, since those are still not ready to cope with the many issues of 3D-Vias [7, 13].

While it is known that the addition of more 3D-Vias improves wire length [5], this design methodology might fail to solve the issues highlighted above. We believe that EDA tools can play a major role to enable random logic in 3D by minimizing 3D-Vias to acceptable levels. The number of 3D-Vias required in a design is determined by the tier assignment of each cell, which is performed during the cell partitioning. The cell partitioning is usually performed by hypergraph partitioning tools (since it is straightforward to map a netlist into a hypergraph) such as hMetis [8] as done in [2] for the same purpose that is addressed here. On the other hand, hypergraph tools do not understand the distribution of partitions in the space (in 3D circuits they are distributed along in a single dimension) and fail to provide optimal results. It is important to understand that the amount of resources used is proportional to the vertical distance

[2] On Leave from UNIJUI – Northwest University of Rio Grande do Sul, DETEC – Department of Technology, Ijui, Santa Rosa, RS, Brazil.

between tiers; in fact, considering that the path from a tier to an adjacent involves regular vias going through all metal layers plus one 3D-Via, it is clear that any vertical connection larger than adjacency might be too expensive in terms of routing resources and delay.

In this paper we present an iterative cells and I/O pins heuristic that refines the partitioning produced by traditional hypergraph algorithms that further minimizes the 3D-Via count while maintaining vertically shorter nets and keeping good I/O and cells area balance. The remainder of the paper is organized as follow. Section 2 presents the problem being addressed and section 3 presents how we draft a solution to this problem while comparing it to similar approaches. Section 4 presents the details of our refinement heuristic and finally section 5 presents conclusions and directions for future work.

II. PROBLEM FORMULATION

Let us consider a random logic circuit netlist and a target 3D circuit floorplan (including area and number of tiers). We seek to compute the partitioning of the I/O pins as well as the partitioning of cells into tiers such that the 3D-vias count is minimized, constrained to keeping a reasonable balance of both, I/Os pins and cells, along the tiers.

III. PARTITIONING ALGORITHM AND RELATED WORK

Our previous work [9] presents a solution for the presented problem. In this approach, we concentrated in improving the I/O pins partitioning while cells partitioning was naturally improved by 5.33% (2 tiers), 8.29 (3 tiers), 9.59% (4 tiers) and 16.53% (5 tiers). We have used hMetis for cell partitioning while the I/Os were fixed by our method. Experimental results have shown that the initial I/O placement improves the ability of hMetis to partition cells. In this paper, we will tackle the cells partitioning method as well. An iterative improvement heuristic to handle the presented problem (section 2) is proposed. The algorithm is inspired on simulated annealing [14], but instead of accepting uphill solutions to avoid local minima (the heuristic disregards any possible local minima because it obtains a good initial solution using the approach presented in [9]). Besides, the improvements are achieved without any performance penalty.

The main difference between our new approach and a hypergraph partitioner such as hMetis is that our approach accounts for the partitions location. In fact, in a 3D circuit, the partitions can be understood as a vertial line, which implies in the notion of adjacent partitions (that are cheap in terms of cut) and distant partitions (that are expensive since they demand extra 3D-vias). Having the goal of minimizing the 3D-vias as a whole, we naturally want to overpenalize the cut of distant partitions, which is not done in multi-way hypergraph partitions. For example, the algorithm in [2, 9] employs hMetis to partition the cells into groups, and in a second stage employs a post-processing method to distribute the partitions in the 1D space (line) such that the number of 3D-vias is minimized (as illustrated in figure 1.a). While this approach targets the same goals, it clearly has limited freedom since partitions cannot be changed after defined. The algorithm proposed in this paper allows cells to move from one partition to the other as long as the final cost is reduced, as illustrated in figure 1.b. The definition of the cost function is responsible for keeping a good I/O pins and cells balance among the different tiers.

IV. IMPROVEMENT METHOD FOR HEURISTIC PARTITIONING

The proposed algorithm picks an initial solution and improves it iteratively using random perturbations of the existing solution. The perturbations might be accepted or rejected depending on the cost variation. Any perturbation that improves the current state is accepted and all perturbations that increase the cost is rejected (greedy behavior).

A. Pertubation Procedure

The perturbation function designed for our application attempts to move cells across partitions. Although they are random in nature, we perform two different kinds of perturbations for better diversity: single movement or double exchange (swap). The double and single perturbations are alternated with 50% probability. They work as follows:

- The single perturbation can randomly pick a cell or an I/O pin (with 50% probability each) and move it to a different tier (also chosen randomly).
- The double perturbation randomly selects a pair of elements to switch partitions. Each element can be either a cell or I/O pin with 50% of selecting each, totaling 4 different double perturbation combinations, each having 25% probability of happening.

B. Cost Function

Any intermediate state of the partitioning process can have its quality measured by a cost function. In the cost function, we model all metrics of interest in a single value that represents the cost.

Our cost function is divided in three distinct parts: a cost ***v*** associated to the usage of 3D-Via resources, a value ***a*** for the area balance and finally a cost ***p*** for the I/O pins balance. The cost reported is a combination of the three parcels; in order to be able to add them together, we normalize each parcel by dividing them from their initial values v_i, a_i and p_i (computed before the first perturbation). In addition, we also impose weights (w_v, w_a and w_p) in order to be able to fine tune the cost function to vias in the optimization process, as shown in equation 1.

$$c = \frac{(w_v \times v)}{v_1} + \frac{(w_a \times a)}{a_1} + \frac{(w_p \times p)}{p_1} \qquad (1)$$

The values v, a and p are computed as follows.

- For each net, we compute the square of the via count; add the computed number of each net to obtain v. The square is applied to highly punish nets having high 3D-via counts and to encourage short vertical connections.

- To compute *a*, we first calculate the cells area of all tiers; the unbalance cost is a subtraction of the largest by the smallest area.
- The value *p* is computed similarly to *a*.

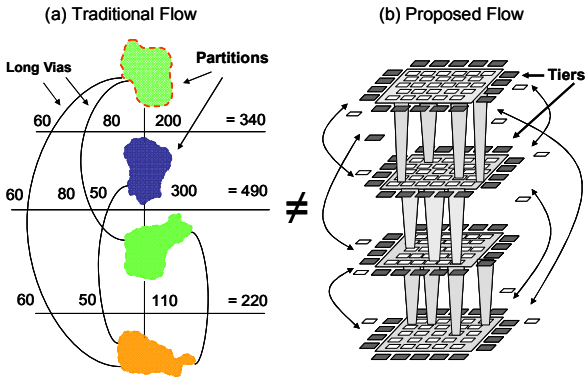

Figure 1: Fixed tiers method

V. EXPERIMENTAL RESULTS

A. *Experimental Setup*

To evaluate the quality of the partitions generated by the algorithm proposed in section IV, we have compared it to two others approaches. The first approach relies on partitioning the cells using a state-of-the-art hypergraph partitioner called hMetis. The number of cells in each partition is constrained to be approximately the same. Cells are is partitioned into *n* partitions, where *n* is also the number of tiers. In a subsequent stage, each partition is assigned to a tier, as illustrated in figure 1a. Partitions are assigned to tiers in such a way that the total number of vias is minimized. This step is done by an algorithm that tests all possible tier assignments, e.g., considering 5 tiers, only 120 combinations have to be tested. This solution is known as hMetis.

The second cells partition approach is presented in [9]. In this approach, the I/O pins are partitioned considering the logic distance between pair of I/O pins. Following the I/O partitioning, we fix the pins and perform hMetis for the cells partitioning. This method was named I/O pins.

The method proposed for this work is analyzed in the figures below. Please, note that this method starts from the I/O pins solution and runs our improvement heuristic to refine both the cells and the pins partitioning. Therefore it should be noticed that the proposed algorithm assures a good balanced solution with respect to both, cells area and number of pins. The balance is achieved by penalizing unbalanced solutions, as shown in equation 1.

Each algorithm was evaluated by measuring the different aspects of the partition for the ISPD2004 benchmark suite [12]. All methods were constrained to distribute area evenly, which resulted in a worst case of 0.1% unbalance. The I/O balancing is not imposed in the hMetis since it would overconstrain the method. For this reason, hMetis has the worst I/O balancing while the I/O Pins has the best.

B. *Results*

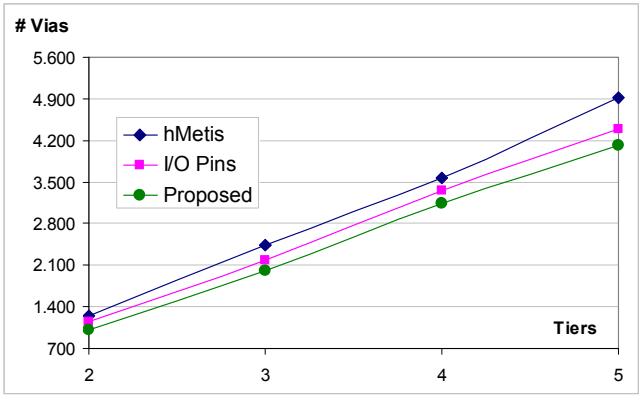

Figure 2: Number of 3D-vias

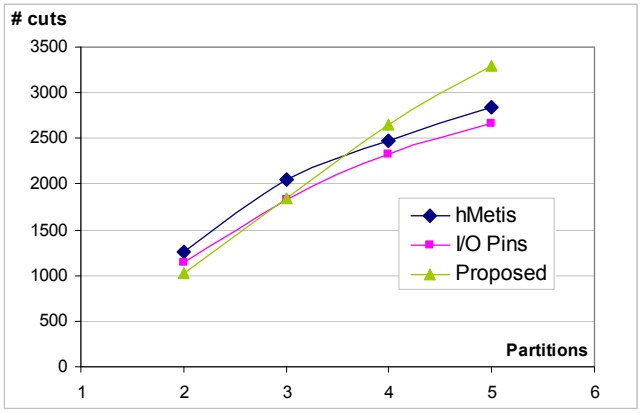

Figure 3: Min-cut quality

Figure 2 shows the average 3D-via count comparison among the methods. The benchmark circuits were partitioned into 2, 3, 4 and 5 tiers using the evaluated algorithms. The proposed method obtains the average least amount of 3D-vias. More specifically, the proposed method produced a 3D-via count average improvement of 19% and 11% compared to hMetis and I/O pins respectively for 2 tiers, 17% and 8% for 3 tiers, 12% and 6% for four and finally 16% and 7% for 5 tiers. For a larger number of tiers the proposed algorithm presents a larger improvement when compared to hMetis. This is a direct consequence of the partitioning refinement step that targets at reducing the number of vias in long connections (i. e., connection between non-adjacent tiers), therefore, the larger is the number of tiers the larger is the number of large connections and the improvement obtained by the proposed algorithm. Since the partitioning refinement step is done after the partitions have been assigned to the tiers, it takes advantages from the knowledge of the actual partition locations, reducing the number of connections between non-adjacent tiers and increasing the number of connections between adjacent ones. This strategy yields a smaller number of 3D-Vias.

Figure 3 shows the final partitioning from a hypergraph perspective. The y-axys shows the average cut among the different partitions when the circuits are divided into 2, 3, 4

and 5. Reported numbers represent an average over all benchmarks. It can be observed that the proposed algorithm yields a higher cut when the number of partitions is larger, however when only two partitions are created it achieves a smaller cut value than hMetis and I/O pins. This behavior is explained by the cost function used to optimize the final partition set. The proposed algorithm, unlike hMetis and I/O pins, does not tackle at cut reduction but in reduction of total number of vias. When only two partitions are considered the number of vias is equal to the partitioning cut. When more partitions are created the proposed algorithm increases the number of connections between adjacent partitions in order to reduce the number of connections between non-adjacent ones, as previously mentioned. This behavior leads to an increase in the partitioning cut number while the total number of 3D-vias can be further reduced.

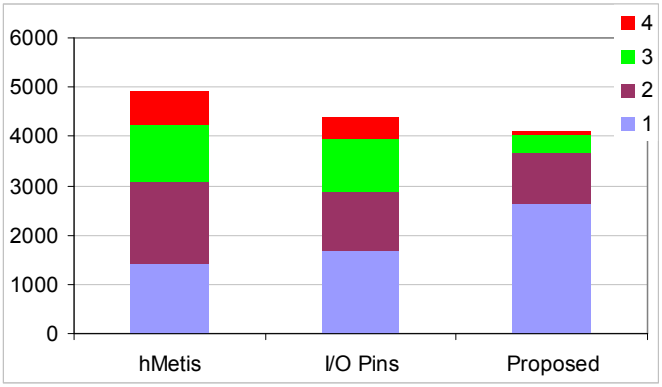

Figure 4: Vias distribution for a 5-tier design

Figure 4 presents a more detailed look into the via distribution among the different tiers for the 5-tier configuration. Each bar represents the total number of 3D-Vias obtained by each algorithm. The bars are divided into four parts. The lower part represents the number of vias that belong to adjacent connections between tiers, while the others represent 3D-vias in non-adjacent connections. The block identified by the number 2 represent amount of 3D-vias in connections that are one tier away, i. e., for each connection two 3D-vias are needed. Blocks identified by 3 and 4 represent 3D-vias introduced by connections that are 2 and 3 tiers away respectively. It should be noticed that Figure 4 shows the number of vias that belong to the different types of connection, i. e., if a design presents three connections between tiers that are 3 tiers away the number reported in figure 4 is 12, since each connection requires 4 vias. The 3D-Via reduction using the proposed algorithm was of 804 vias (16%) when compared to the hMetis approach and 280 (6%) when compared to I/O Pins. Experiments using, 2, 3 and 4 tiers were performed and present the same behavior.

VI. CONCLUSIONS

This paper presented a method to refine the I/O pins and cells partitioning for placement in a 3D VLSI circuit. We were able to develop a heuristic iterative improvement algorithm that achieves better partitioning than what is possible by just applying hypergraph partitioners. The method demonstrates that hypergraph partitioners are not well suited for cell and I/O partitioning into 3D circuit because they do not handle long connections properly, affecting the total 3D-via count. We demonstrated that our heuristic was able to improve the 3D-via count considering the positions of each tier within the chip while partitioning the cells among the tiers. Finally, we highlight that our heuristic was able to perform partitioning while keeping cells area and I/O pins count balanced for all tiers without.

ACKNOLEDGMENTS

The authors thank the extremely valuable contribution of Dr. Renato Hentschke.

REFERENCES

[1] W. R. Davis 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, 2005.

[2] C. Ababei, et al. 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: ICCAD'03*, November, San Jose, California, USA, 2003.

[4] E. Wong; S. Lim. 3D Floorplanning with Thermal Vias. *In: DATE '06*, 2006. p.878–883.

[5] Das, S et al. Technology, performance, and computer-aided design of three-dimensional integrated circuits. *In: ISPD'04, 2004*, New York, NY, USA. 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: ISVLSI, 2007*, p.67–72.

[8] G. Karypis et al. Hypergraph Partitioning: Application in VLSI domain. *In Proceedings of 34th Annual Conference on. Design Automation, DAC 1997*, pages 526–529, 1997.

[9] Sawicki, S., et al. 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 et al. Interconnects in the Third Dimension: Design Challenges for 3D ICs. *In: DAC'07. 44th ACM/IEEE. 2007* Page(s):562 - 567

[11] K. Banerjee et al. 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 et al. Quadratic Placement for 3D Circuits Using Z-Cell Shifting, 3D Iterative Refinement and Simulated Annealing, *In: SBCCI 2006*, 220-225.

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