CLEI ELECTRONIC JOURNAL, VOLUME 13, NUMBER 3, PAPER 1, DECEMBER 2010

3D-Via Driven Partitioning for 3D VLSI Integrated Circuits

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

[1], UFRGS – Universidade Federal do Rio Grande do Sul
PPGC/PGMicro – Instituto de Informática
Porto Alegre, RS, Brasil

[2], UNIJUI – Universidade Regional do Noroeste do Estado do Rio Grande do Sul
DETEC – Departamento de Tecnologia
Ijui, Santa Rosa, RS, Brasil
{sawicki, wilke, johann, reis}@inf.ufrgs.br

Abstract. A 3D circuit is the stacking of regular 2D circuits. The advances on the fabrication and packaging technologies allowed interconnecting stacked 2D circuits by using 3D vias. However, 3D-vias can impose significant obstacles and constraints to the 3D placement problem. Most of the existing placement algorithms completely ignore this fact, but they do optimize the number of vias using a min-cut partitioning applied to a generic graph partitioning problem. This work proposes a new approach for I/O pads and cells partitioning addressing 3D-vias reduction and its impact on the 3D circuit design. The approach presents two distinct strategies: the first one is based on circuit structure analyses and the second one reducing the number of connections between non-adjacent tiers. The strategies outperformed a state-of-the-art hypergraph partitioner, hMetis [8] in the number of 3D-vias 19%, 17%, 12% and 16% using two, three, four and five tiers.

Keywords: Partitioning, 3D VLSI Circuits, CAD, I/O Pins

1 Introduction

While the most recent manufacturing technologies introduce 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, 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, i.e. a 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 maximum density is considerably small compared to regular vias, which won't allow as many vertical connections as could be desired by EDA tools; 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 manufacturing 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-vias, 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] 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, 14, 15].

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]. 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 of the 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.

This paper presents a new approach for I/O pads and cells partitioning targeting 3D-vias reduction. In section 2, we present the problem formulation. Section 3 describes an algorithm for I/O pins partitioning based on the circuit structure analyses. We them propose an algorithm on section 4 for non-adjacent 3D-vias reduction. Finally, the experimental results and conclusions are presented in Section 5 and 6 respectively.

2 Problem Formulation

Consider a random logic circuit netlist and a target 3D circuit floorplan (including area and number of tiers), 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; be constrained by keeping a reasonable balance of both, I/Os and cells, along the tiers, as shown in Figure 1.

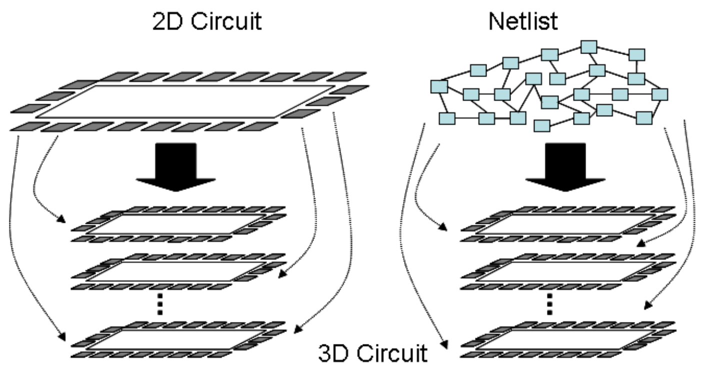

Fig. 1. Migration from 2D to 3D VLSI integrated circuit.

3 Proposed Algorithm Based on Circuit Structure

For this approach, we analyze the random logic block structure and create an I/O partitioning flow. The algorithm firstly calculates the logical distance between pair of I/O pins. Next, it creates a complete graph of I/O pins considering the logical distance as a weight. Finally, it partitions the graph using hMetis and considering the logical distance between I/O pins. The I/O pins are locked to its partitions. Based on the I/O pins location, the cells are partitioned. In the end, the simulated annealing [12] is applied to find the best stacking arrangement. The I/O placement preserves the same I/O pins orientation, whitespaces and aspect ratio of the original netlist. This method was named I/O pins. More details can be found in [9] and [11].

Considering that in a real circuit net, fanouts are limited, node degrees can be considered bounded or constant for the sake of complexity analysis. For that, a single BFS search will have an *O(n)* complexity. Our algorithm be performed by m^2 BFS searches in *HG* resulting in a $O(m^2n)$ complexity. Since the number of I/O pins do not exceed a few thousand, it is feasible to use BFS. By using a single search to compute the distance from a pin p_i to every $p \in P$, the complexity can go down to *O(mn)*.

The values of shortest path are used to create a complete graph connecting all pairs of I/O pins, as shown in Figure 2. For the cells partitioning, we used hMetis tool [8]. The tool accepts weights for the cells. We assigned the inverse of the edge costs as their weights. We imposed a very tight balance in order to keep a similar amount of I/Os in each tier.

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

```
1   Compute the logic distance
2   Create a complete graph of pairs of I/O pinos
    considering the logical distance as a weight.
```

3	Perform the partitioning of complete graph aiming at min-cut optimization and very good number of pins between partitions.
3	Lock the I/O pins into partitions
4	Perform the cells partitioning considering I/O positions
4	Aspect ratio (from original netlist)
5	Pins orientation (from original netlist)
6	Whitespaces (from original netlist)
9	Legalize I/O positions.

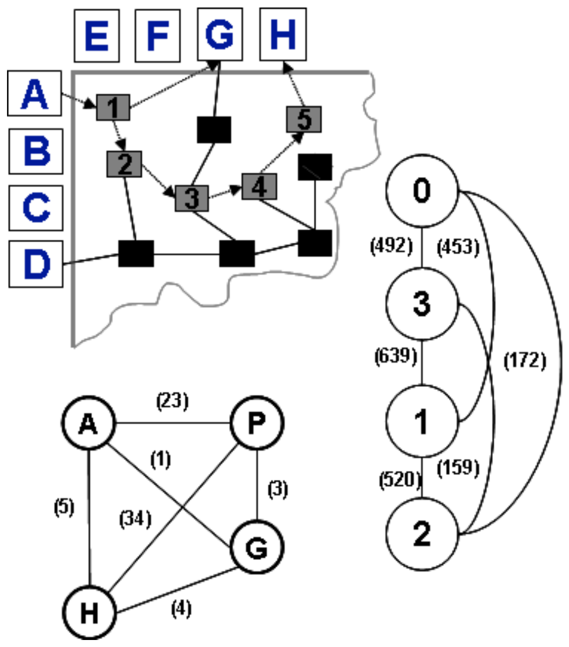

Fig. 2. Illustration of the shortest path between two I/O pins and a portion of the correspondent complete graph of all boundary pins.

4 Reducing Non-Adjacent 3D-Vias

The algorithm presented here is called Refinement and it picks an initial solution and improves it iteratively using random perturbations of the existing solution without any penalty performance. 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 are rejected. The cost function is divided into three distinct parts: a cost ***v*** associated to the usage of 3D-vias 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; 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 fine tune the cost, as shown in equation 1. Any intermediate state of the partitioning process can have its quality measured by this cost function. In the cost function, we model all metrics of interest in a single value that represents the cost.

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

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

- For each net, 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.

4.1 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. More details can be found in [16]

4.2 Proposed Flow

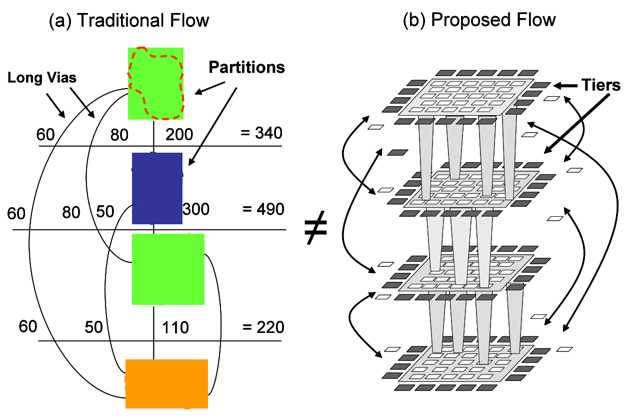

Fig. 3. Fixed tiers method

Figure 3 illustrates the differences between the proposed cell partitioning flow and the standard state-of-the-art flow. The algorithm proposed here allows cells to move from one partition to the other as long as the final cost is reduced, as illustrated in figure 3.b and it is described as follows:

```
Step 1   Compute the netlist keeping tiers orientation
         Calculate the initial number of:
Step 2       • 3D-vias;
             • Cells area unbalance;
             • I/O  pins;
Step 3   Perform the perturbation procedure
Step 4   Execute the change procedure:
         {(pin, pin) | (pin, cell) | (cell, pin) | (cell, cell)}
Step 5   Calculate the cost:
         Δ Cost = Cost (new Solution) – Cost (Solution);
            If (Δ Cost < 0,7)
               Accept ();
               Step 3
Step 6      Else
               Reject ();
               Undo ();
               Step 3
          If do not have any chance procedures
             Exit();
Step 7   Else
             Step 3
```

5 Experimental Results

5.1 Experimental Setup

We used benchmarks IBM ISPD 2004 [10] for our experiments. The proposed algorithms, I/O Pins and Refinement were compared with the state-of-the-art partitioning algorithm, called hMetis (the same approach of [2]). The Table I shows the number of cells, pads and nets for each one of the benchmark circuits.

Table 1. Benchmarks ISPD 2004 with PADS

Benchs	*#Cells*	*#Pads*	*#Nets*
ibm01	12.506	246	14.111
ibm02	19.342	259	19.584
ibm03	22.853	283	27.401
ibm04	27.220	287	31.970
ibm05	28.146	1201	28.446
ibm06	32.332	166	34.826
ibm07	45.639	287	48.117
ibm08	51.023	286	50.513
ibm09	53.110	285	60.902
ibm10	68.685	744	75.196
ibm11	70.152	406	81.454
ibm12	70.439	637	77.240
ibm13	83.709	490	99.666

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 is the best since the proposed method uses a little freedom on the I/O balancing to improve the 3D-Via count.

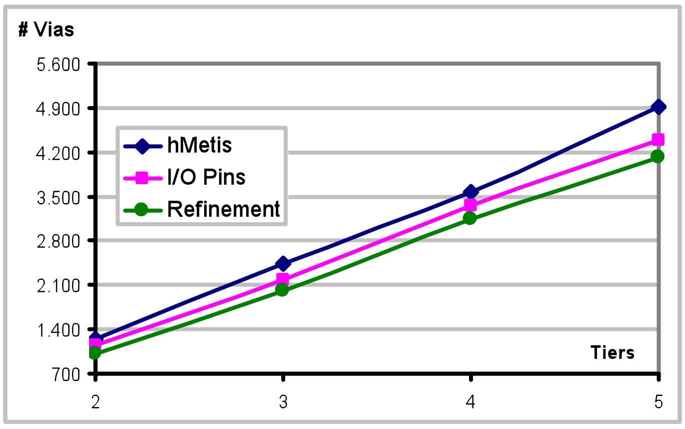

Fig. 4. Number of 3D-Vias

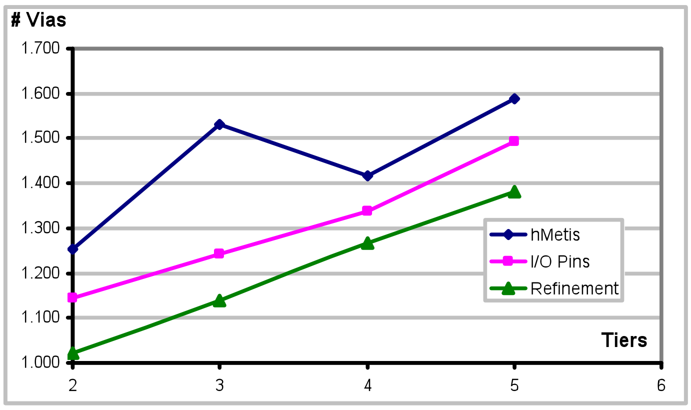

Fig. 5. Max number of 3D-vias

5.2 Results

The Figure 4 shows the average 3D-vias count comparison between the methods. The benchmark circuits were partitioned into two, three, four and five tiers using the evaluated algorithms. The Refinement algorithm obtains the average least amount of 3D-vias. More specifically, Refinement lead to an average 3D-vias count 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 Refinement 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 long connections and the improvement obtained by this algorithm. Since 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.

The Figure 5 shows the max number of 3D-vias between a pair of adjacent *tiers*. The Refinement algorithm obtains the average least amount of 3D-vias. The improvements are 19%, 26%, 11% and 13% using two, three, four and five tiers respectively compared to hMetis and 11%, 8%, 5% and 7% compare to I/O pins.

The Figure 6 presents a more detailed look into the vias 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.

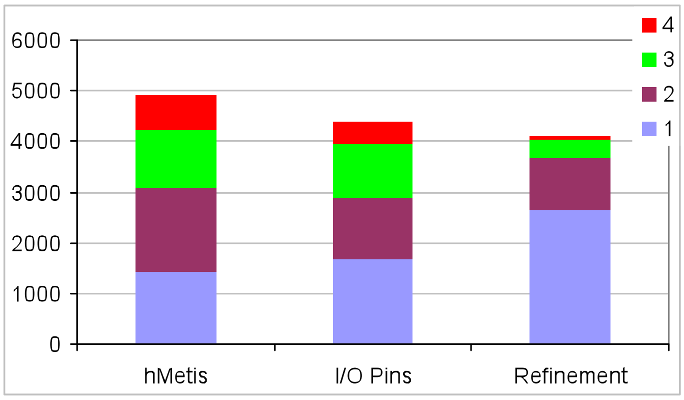

Fig. 6. 3D-vias distribution for a 5-tier design

The block identified by the number 2 represents the 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 6 shows the number of vias that belong to different types of connection, i. e., if a design presents three connections between tiers that are 3 tiers away the number reported in figure 6 is 12, since each connection requires 4 vias. The 3D-vias reduction using the Refinement algorithm was of 804 vias (16%) when compared to the hMetis approach and 280 (6%) when compared to I/O Pins. Experiments using, two, three, four and five tiers were performed and presented the same behavior.

6 Conclusions

This paper presented a new approach for I/O pads and cells partitioning targeting 3D-vias reduction. The methodology was based on two distinct strategies: circuit structure analyses and the number of connections between non-adjacent tiers. In the first strategy, we proposed that the I/O partitioning and placement is done upfront, while 3D placement will start from fixed I/O pins. In the paper, we showed empirically that doing the partitioning of I/O together with the cells leads to unbalanced number of pins, which invalidates the method. Our method is based on the idea of keeping the pins with logic proximity together in the same tier. In the second strategy 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 by considering the positions of each tier within the 3D chip while partitioning the cells among the tiers. Finally, we highlight that our heuristic was able to perform partitioning while keeping area and I/O pin count balanced for all tiers.

Acknowledgment

The authors thank the extremely valuable contribution of Dr. Renato Hentschke. Dr. Hentschke has actively participated on the development of this work while he was working towards competing his PhD at UFRGS.

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, Nov.-Dec. 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. S. Das, 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. R. Patti, Three-dimensional integrated circuits and the future of system-on-chip designs. Proceedings of IEEE, [S.l.], v.94, p.1214–1224, 2006.
7. R. Hentschke, et al. 3D-Vias Aware Quadratic Placement for 3D VLSI Circuits. In: IEEE Computer Society Anual Symposium on VLSI, ISVLSI, 2007, Porto Alegre, Brazil. Los Alamitos: IEEE Computer Society, 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. S. Sawicki, 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, 2006.
10. ISPD 2004 - IBM Standard Cell Benchmarks with Pads. http://www.public.Iastate .edu/ ~nataraj /ISPD04 _Bench.html #Benchmark _Description. Access on Mar 2009.
11. S. Sawicki, et al. "Unbalancing the I/O Pins Partitioning for Minimizing Inter-Vias in 3D VLSI Circuits". In: IEEE International Conference on Electronics Circuits and Systems, 2006.

12. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, Science, 1983, 220, pages 671-680.
13. P. Flynn, Y. Kim, I. Yoon. 3-D Stacked Package Technology and Trends; In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 31-41, 2009.
14. M. Motoyoshi. Through-Silicon Via (TSV), In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 43-48, 2009.
15. V. Pavlidis, E. G. Friedman. Interconnect-Based Design Methodologies for Three-Dimensional Integrated Circuits, In: Proceedings of the IEEE, Volume: 97, Issue: 1, On page(s): 123-140, 2009.
16. S. Sawicki, et al, "A Cells and I/O Pins Partitioning Refinement Algorithm for 3D VLSI Circuits". In: Proceeding of the IEEE International Conference on Electronics, Circuits and Systems, 2009.