A Method for I/O Pins Partitioning Targeting 3D VLSI Circuits

Renato Hentschke, Sandro Sawicki [*], Marcelo Johann, and Ricardo Reis

UFRGS - Universidade Federal do Rio Grande do Sul - Instituto de Informatica
Av. Bento Goncalves, 9500. CEP 91501-970. Porto Alegre, Brazil
{renato,sawicki,johann,reis}@inf.ufrgs.br

Abstract. This paper presents an algorithm for I/O pins partitioning and placement targeting 3D circuits. The method starts from a standard 2D placement of the pins around a flat rectangle and outputs a 3D representation of the circuit composed of a set of tiers and pins placed at the four sides of the resulting cube. The proposed algorithm targets a balanced distribution of the I/Os that is required both for accommodating the pins evenly as well as to serve as an starting point for cell placement algorithms that are initially guided by I/O's locations, such as analytical placers. Moreover, the I/O partitioning tries to set pins in such a way the it allows the cell placer to reach a reduced number of 3D-Vias. The method works in two phases: first the I/O partitioning considering the logic distances as weights; second, fix the I/Os and perform partitioning of the cells. The experimental results show the effectiveness of the approach on balance and number of 3D-Vias compared to simplistic methods for I/O partitioning, including traditional min-cut algorithms. Since our method contains the information of the whole circuit compressed in a small graph, it could actually improve the partitioning algorithm at the expense of more CPU time. Additional experiments demonstrated that the method could be adapted to further reduce the number of 3D-Vias if the I/O pin balance constraint can be relaxed.

1 Introduction

Many of existing design issues rely on wiring problems. Issues like delay, variability and manufacturability are highly valuable research subjects in the present days. Timing is importantly affected by wires that contribute to more than 50% of the critical path delay. The power consumption produced by the switching activity of wires, specially clock signals, also contribute to a very large chunk of total power dissipated. Reliability and manufacturability are also related to chip wires.

3D circuits appear as a change of design paradigm, providing higher integration and reducing wire lengths [10]. By either analytical methods [5] [8] [18] [19] and practical experimentation [10] [9] [15] [2], it is well known that 3D circuit technology has the potential of providing many improvements to VLSI circuits,

[*]On leave from UNIJUI - Universidade Reg. do Noroeste do Estado do RS - Brazil

including: reduction of the size of the longest wires [9]; average wire length reduction (from 15% to 50%) [9]; dynamic Power reduction of up to 22% [19] [10]; chip area reduction [18].

Among the new issues introduced by 3D circuits, the communication elements (known as 3D-Vias) between adjacent tiers impose several constraints to the physical design of those circuits. Firstly, their electrical characteristics are differentiated from regular wires. From the routing perspective, in order to connect to a 3D-Via, a wire is required to cross all metal layers. More importantly, 3D-Vias require significant sizes for design rules such as minimum pitch. As more detailed in section 2.2, face-to-back communication imposes more restrictions, since it digs a hole through the Bulk of a tier occupying active area and compromising reliability. All those factors make 3D-Via planning a complex issue that must be addressed by CAD tools.

The new 3D issues must be addressed with proper CAD tools able to synthesize in a new design paradigm to take full advantage of 3D integration. Among possible design methodologies, the integration granularity will impact possible benefits and the type of problems to be solved. Initially consider a *tier level integration*, which stacks separated tiers of different nature. It is the most coarse level granularity and do not severely affect existing design methodologies, since each tier can be designed separately with a simple glue logic to integrate them. Secondly, consider an *ip core level integration* that partitions big circuit blocks (ip cores) into different tiers, providing a tighter integration (more communication between tiers). Finally, *random logic level* partitioning breaks random logic into 3D. Figure 1 illustrates a random logic block broken into 2 tiers.

Basically, the finer the integration grain is, the bigger the potential vertical communication requirement is, causing two effects: 1) more potential benefits (as listed above); 2) more complex 3D-Via related problems to solve. The higher complexity of 3D-Via planning must be addressed by physical design algorithms, encouraging research on this field. The random logic integration granularity with the usage of more 3D-Vias while optimizing wire length of a block on 3D leads to a better usage of the 3D resources and helps reducing wire length, as demonstrated by [9].

The problem of partitioning a block into 2 or more tiers starts with the definition of an I/O interface. Although all the existing 3D placement literature ignores this problem, possibly using some simplistic solution, an appropriate placement of the I/Os in the boundary of the block has a very important impact on the cells placement. I/O pins play two important roles in the placement of a block: first, I/Os limit the area boundary of the block; second, the pins are used as tips for many placement algorithms to reduce wire lengths. Consider the Quadratic Placement algorithm [4], that is used by the leading industry and most of the existing academic cell placers. It requires I/Os at the boundary in order to compute a solution. If all I/Os are assigned to stay in a unique tier, the quadratic placement method will not be able to move the cells in 3D.

This paper proposes a method for the I/O partitioning of a random logic block based on the logic distance of the I/Os as partitioning criterion. Summarizing the motivation, the goal is to find a good partitioning method for the I/Os that is able to maintain a good I/O pins balance leading to area balance between the tiers. At the same time, we indirectly address the reduction of 3D-Vias. Our insight is that a low 3D-Via starting point leaves more room for a 3D placer to insert 3D-Vias while improving wire length if there is available space. The rest of the paper is organized as follows. Section 2 presents a few details on 3D VLSI circuits that will be helpful to understand the experimental results and motivation. Section 3 defines the problem we are addressing. Section 4 presents the I/O partitioning algorithm. Sections 5 and 6 present experimental results while conclusions are discussed in section 7.

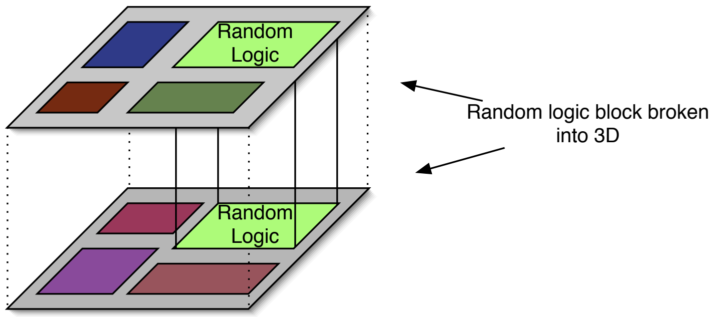

Fig. 1. Random logic blocks could be broken into 3D.

1.1 Related Work

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. The works from [2], [10], [11] for instance, apply min-cut partitioning (usually with hMetis tool [14]) to assign cells into tiers, minimizing the number of 3D-Vias. A subsequent step performs 2D placement on each tier separately; the already placed tiers can serve as a guide to subsequent tiers in order to minimize wire length. However, [15] [16] [7] already identified that this approach leads to worse results in terms of wire length. We call True 3D placer a method that is able to both measure and optimize wire length in all the axis at the same time.

Liu et. al [16] builded a two step 3D placement flow similar to the one mentioned above using hMetis for partitioning the cells into tiers. They argue that building a True 3D flow is very hard and for this reason they concentrate on improving the partitioning step. They observed that the insertion of 3D-Vias could potentially improve wire length. For this reason, their cell partitioner does not perform min-cut partitioning, but tries to maximize the 3D-Vias under an upper bound constraint. In fact, since face-to-face integration allows 3D-Vias with no cost to yield or area, they could be inserted freely in order to improve wire length. Some preliminary evaluation could be performed to analise a reasonable upper bound for those 3D-Vias. Liu's algorithm cannot achieve the exact via count provided, but tries to get a close approximation by an iterative algorithm. After the tier assignment, the algorithm uses the Capo tool [6] to place the cells in each tier.

Das et. al [7] [9] builded a true 3D partitioning based placement engine. It recursively cuts the placement cube performing min-cut partitioning. A wire length and 3D-Via trade-off can be obtained by controlling the point at which the cut is performed into the Z axis (i.e. the point at which the design is partitioned into tiers). The optimal solution for wire length is obtained when the aspect ratio drives the cut direction. The solution with fewer 3D-Vias can be obtained in the case where the first cut is made on the Z axis (method that would be equivalent to the ones based on hMetis assignment mentioned above).

Goplen and Sapatnekar [12] formulate the 3D placement problem as a True 3D placement. They provide an analytical force directed algorithm that minimizes the squared 3D wire length. Their method is iterative; at each iteration repulsive forces related to thermal issues or cell overlaps are inserted in the system. This process makes cells spread into the placeable volume. The authors do not detail how they handle I/Os into the tiers; however, on quadratic placement methods the cells will not move in the Z axis unless the I/Os are placed in different tiers. In this case, it can be understood that the repulsive forces are responsible for moving cells into other tiers. After placement is completed, the cells are sorted in the Z axis and finally assigned to a circuit tier. This method may fall into a false wire length optimization since actually cells cannot be placed into continuous coordinate; the rounding of their coordinated could potentially decrease circuit wire length.

Obenaus et. al, in [20], present an iterative force directed method for 3D placement. Different from Goplen's placer, it is not an analytical method but moves all cells (cell-by-cell) to an optimal position according to its connections. They define the 3D placement problem to minimize wire length only, which handles the problem as true 3D method. 3D-Via costs and constrains are not considered. No repulsive forces are added to the system, but a bucket re-scaling method similar to cell shifting from [21] spreads out the cells.

2 3D VLSI Integrated Circuits

A 3D circuit can be defined as a VLSI chip with stacked active layers called **tiers**. In the following sections, more details of the 3D fabrication and impacts on design methodologies will be presented. Figure 2 provides a didactic view of a 3D Chip with active layers and metal layers. Depending on the integration strategy used there may be or not metal layers above the last tier of active area. Also, depending on the integration strategy, more metal layers can be contained between a pair of adjacent tiers. More details on the technologies and how they are manufactured are provided in the following sections.

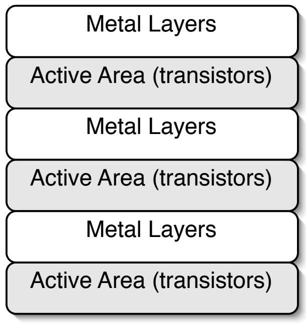
Metal Layers
Active Area (transistors)
Metal Layers
Active Area (transistors)
Metal Layers
Active Area (transistors)

Fig. 2. A didactic picture of 3D circuit, tiers of active area and metal layers

2.1 Manufacturing technologies

According to [17], the assembly of 3D Chips is performed in different integration granularities.

Chip stacking is simply the vertical stacking of fully pre-manufactured chips. The chips have regular buffered I/O connections integrated usually by wire bonding [10]. Since all inter-chip communication must pass through the I/O buffers going outside of the chip, this methodology does not provide any advantage to circuit performance and power, reducing only the area occupied by the chip on the board. This technique is applied for cell-phones and other portable devices.

Die-on-wafer stacking is performed by stacking individual tested dies into a host wafer. Positions of the host wafer can also be pre-tested. The individual dies are placed using a pick-and-place equipment, that is a bottleneck for the cost, quality and size of inter-chip communication. Patti [17] reports that the placement misalignment today is about 10 μm.

Wafer-level stacking bonds entire wafers into a stack. Tezzaron is one company working with this kind of integration. Compared to die-on-wafer stacking, Tezzaron's technology [13] achieves better alignment (1 μm) and a more planar surface, leading to more integrated communication.

Finally, the **transistor stacking** methodology is an ideal integration of active layers fabricated in the same die, dismissing all equipment for wafer alignment. Today, those devices cannot be fabricated mainly due to high temperature process during the wafer manufacturing. Basically, the technology for fabricating high-performance transistors demands temperatures that would destroy any copper or aluminum used to manufacture metal layers bellow it. There is ongoing research in order to solve this issue and in the future this technology is very promising.

According to [10] the types of 3D-Vias can be classified into wire bounded, microbump, contactless and through vias.

In **wire bonded** technology, tiers of different sizes are stacked and I/O Pads are placed in the boundary of the tiers in such a way that they are not blocked by the upper tier. The main disadvantage of this technology is that wires are out of the chip scope, so they must be buffered and the pads consume very large areas.

Microbump technology provides micro contacts (bumps) placed in the top metal layer (sometimes the top two metal layers may be blocked for other routing). For this technology, chips can be stacked face-to-back and the package itself can provide routing space (3D package). On the other-hand, stacking the chips in a face-to-face fashion provides simpler (and consequently better) routing requiring no wiring channels in the package. The tiers are placed in such a way that their respective bumps are physically connected. Face-to-face integration is limited to two tiers.

The **contactless** technologies can be summarized as capacitive and inductive coupling. The capacity coupling technologies require the chips to be placed face-to-face because the contacts have a very tight proximity constraint. Inductive coupling is usually integrated face-to-back.

Finally, **Through Vias** consists of digging a hole though the tier for face-to-back comunication. Sometimes, such as in MITLL 3D technology [10], the first two tiers are integrated face-to-face while the rest of the tiers are stacked face-to-back. Even two chips connected face-to-face will need face-to-back communication with the I/O pads. Due to silicon polishing issues, the traditional Bulk technologies requires a much larger pitch compared to SOI processes for 3D, such as in the MITLL. But still in the face-to-face integration, the technology for digging the hole in the oxide and depositing metal is similar. So far, this kind of technology is the one that provides the tighter integration between tiers because they are assembled in the wafer level.

Figure 3 illustrates a 3D circuit layout with Though Vias and Microbump technology for face-to-face connection. Note that there are microbumps in the top of the last metal layers that serves the purpose of connecting the tier to its neighbors.

2.2 Summary of 3D-Vias related information

In this section, some important data for the development of the paper is summarized. 3D-Vias are classified according to the following characteristics:

- The strategy used to integrate the tiers connected by the 3D-Via, that can be either face-to-face, face-to-back or back-to-back;
- The pitch of the 3D-Via;
- Whether the 3D-Via occupies active area or not;

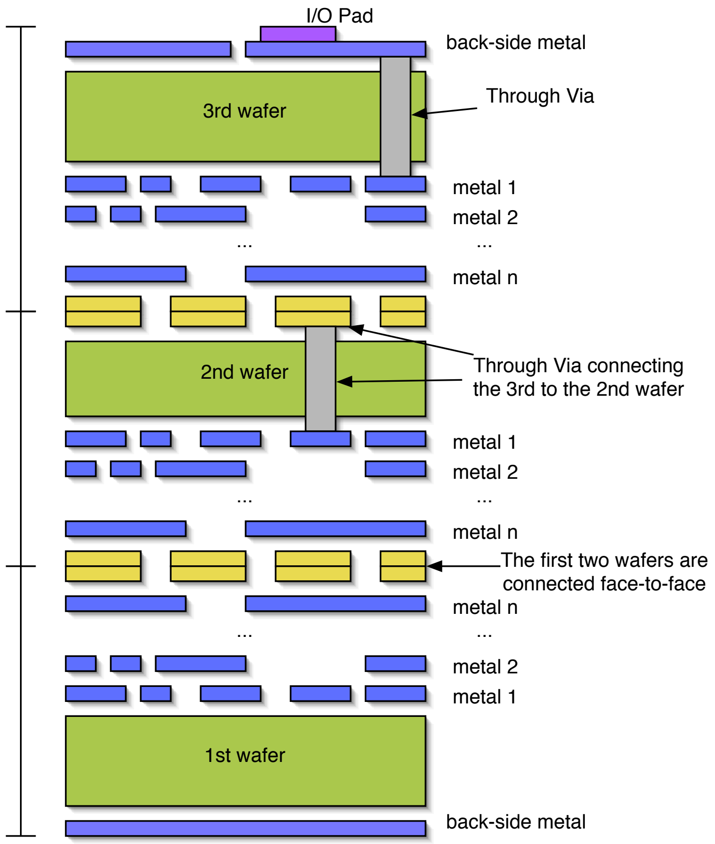

Fig. 3. The layout of a 3D circuit, containing three tiers integrated face-to-face and face-to-back respectively.

A list of some 3D-Vias and its characteristics is presented in table 1.

We can observe that there is a variety of pitches while some 3D-Vias occupy active areas. The methodology for introducing 3D-Vias during 3D placement must be subject to the 3D-Via characteristics. Consider, for instance, the face-to-face 3D-Vias, that can reach pitches in the order of 1 μm. For such technology, 3D-Vias could be used plenty. On the other hand, a face-to-back 3D-Via of, for instance, 50μm would require a huge amount of active area; for this example it would be reasonable to strongly reduce their count.

Table 1. Summary of collected data for 3D-Vias.

3D-Via	Integration Strategy	3D-Via Pitch	Occupy Active Area
Tezzaron (Copper Pads)	face-to-face	2.4 μm	no
Tezzaron (Projected)	face-to-face	1.46 μm	no
Microbump	face-to-face	10-100 μm	no
Contactless (Capacitive)	face-to-face	50-200 μm	no
MIT (Copper/Tantalum Pads)	face-to-face	5 μm	no
TSV face-to-face	face-to-face	0.5 μm	no
Tezzaron Super-Via^{TM}	face-to-back	6.08 μm	yes
Tezzaron Super-Contact^{TM}	face-to-back	¡ 4 μm	yes
Microbump 3D Package	face-to-back	25-50 μm	no
Contactless Inductive	face-to-back	50-150 μm	yes
MITLL Through Via (SOI)	face-to-back	5 μm	yes
Through Via (regular Bulk)	face-to-back	50 μm	yes
Back-to-back 3D-Via	back-to-back	15 μm	yes

3 I/O Partitioning and Placement Problem

Given a 2D placement netlist with pre-placed I/O pins at the boundary of the region available for Standard Cell placement, the migration to a 3D netlist (ready for 3D placement) has the following goals:

– Area allocation: the width and height of the tiers must be calculated according to the number of tiers.
– I/O partitioning: the I/Os must be partitioned into different tiers.
– I/O placement: the I/Os must be placed at the boundary of the block, delimiting the area for Standard Cell placement.

We understand the the I/O partitioning problem should not determine the cells partitioning as well; it is a task of the cell placement. Figure 4 illustrates the I/O pins migration. As formulated formally 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 our idea is to provide a good starting point for the cell partitioning. The algorithm should provide good I/O pins balance and respect the mentioned properties.

Once the netlist is migrated (the I/O pins are placed) we follow the methodology found in [3] that performs min-cut partitioning for the cells and tier assignment with Simulated Annealing as illustrated by figure 6. In our case, though, the min-cut have initially pre-placed fixed pins (I/Os). In this paper, we propose to study the impact of the 3D-Vias in the tier area.

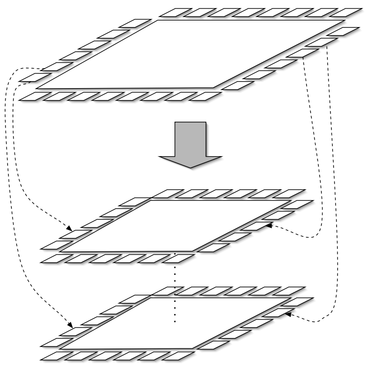

Fig. 4. Migration (from 2D to 3D) of a netlist with pre-placed I/O Pins

3.1 Formal definition

Before placement, a 2D circuit netlist Nl composed by a set of gates $G = \{g_1, g_2, g_3, , g_n\}$, a set of I/O pins $P = \{p_1, p_2, p_3, , p_m\}$ and a set of nets connecting them $N = \{n_1, n_2, n_3, , n_o\}$. A hypergraph Hg represents the netlist, where $G \bigcup P$ 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 $Or(p_i)\epsilon\{north, south, east, west\}$. 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. 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 and dividing the result by Ga. The aspect ratio Ar is computed by W divided by H.

Let Z be the set of tier numbers $\{1, 2, ..., z\}$. The problem to be solved is defined as follows: given a 2D placement netlist Nl with fixed I/O pins, find a set of tiers $T = \{t_1, t_2, , t_z\}$ (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 $(i \leq z)$ such that equations 1-8 hold.

$$P_1 \cup P_2 \cup ... \cup P_z = P \tag{1}$$

$$(\forall a, b\epsilon Z)(a \neq b \rightarrow P_a \bigcap P_b = \emptyset) \tag{2}$$

$$(\forall i\epsilon Z)(Wh_i \approx Wh) \tag{3}$$

$$(\forall i\epsilon Z)(Ar_i \approx Ar) \tag{4}$$

$$(\forall i\epsilon Z)(\forall j\epsilon Z)(W_i = W_j \wedge H_i = H_j) \tag{5}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(Or_i(a) = Or(a)) \tag{6}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(\forall b\epsilon P_i)(Or(a) = Or(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b]) \tag{7}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(\forall b\epsilon P_i)(Or(a) = Or(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b]) \tag{8}$$

In other words, each tier will have its own set of I/O pins and no tier will share an I/O; the whitespace and aspect ratio must be evenly allocated; the orientation and ordering of the pins must be preserved.

4 Proposed algorithm

Let $Ld(p_i, p_j)$ be the length of the shortest path in Hg from p_i to p_j (e.g. the logic distance between p_i and p_j). The algorithm for I/O partitioning is described as follows.

Algorithm 1 I/O Pins Partitioning and Placement algorithm

```
1: Compute $Ld(i, j) \forall i, j \epsilon P$
2: Create a complete graph $Pg$ such that $P$ is the set of nodes and $Ld(i, j)(i, j \epsilon P)$ is
   the cost of the edge connecting nodes $i$ and $j$.
3: Perform the partitioning of $Pg$ into $P_1, P_2, , P_z$ configured to perform min-cut op-
   timization at a 1% maximum unbalance ratio.
4: Compute $Ga_i$ $(i \epsilon Z)$ by $Ga$ divided by $z$
5: Compute $A_i$ by adding $Wh_i$ to $Ga_i$
6: Compute the dimensions of the tiers based on equation 9.
7: Place the I/O pins around the boundary of the block by simple stretching according
   to equation 10.
8: Legalize I/O Positions
```

$$\begin{gathered} W_i = \sqrt{A_i} \times Ar_i \\ H_i = \frac{\sqrt{A_i}}{Ar_i} \end{gathered} \tag{9}$$

$$\begin{aligned} (\forall i \epsilon z)(\forall p \epsilon P_i) X_i[p] &= \frac{(X[p] - x_{ini}) \times W_i}{W} \\ (\forall i \epsilon z)(\forall p \epsilon P_i) Y_i[p] &= \frac{(Y[p] - y_{ini}) \times H_i}{H} \end{aligned} \tag{10}$$

The first step of the algorithm is illustrated in 5.(a). Considering that in a real circuit net fanouts are limited, node degrees can be considered bounded or constant for the sake of complexity analysis. Thus, a single BFS search has an $O(n)$ complexity. The algorithm can be performed by m^2 BFS searches in Hg resulting in a $O(m^2 n)$ time 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 $p \epsilon P$, the complexity can go down to $O(mn)$.

On step2, the values of Ld are used to create a Pg graph connecting all pairs of I/O pins, as shown in figure 5.(b).

For the third step, we used the hMetis tool [14]. The tool accepts cell weights. We assigned the inverse of the edge costs as their weights and imposed a very tight balance in order to keep a similar amount of I/Os in each tier. In section 6.3 the effects of unbalancing the I/O pins are discussed.

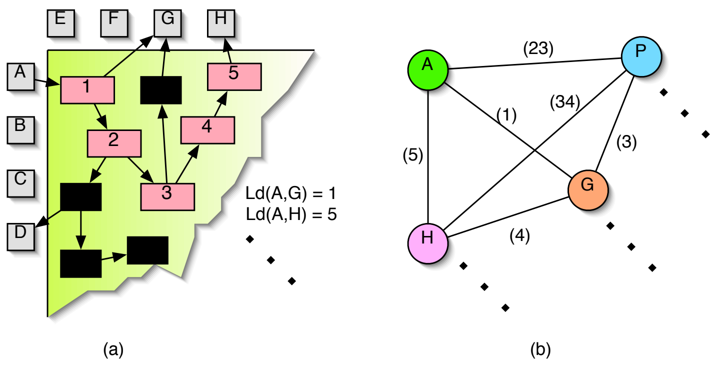

Fig. 5. An ilustration of the logic distance between I/O pins (a) and a part of the correspondent complete graph (b)

The forth step can be accomplished by a simple division of the total gate area by the number of tiers. So far, it is not possible to know whether such perfect cells partitioning will be achievable, but it is a reasonable assumption. Nevertheless, S_i could be changed to compensate the Ga_i inaccuracy.

The steps 5 and 6 compute the area of the tiers such that aspect ratio and whitespace are preserved from the original 2D circuit. At this point, new aspect ratio or whitespace could be used.

Finally, the steps 7 and 8 compute the x and y coordinates of the I/Os to their target tiers. The original orientation and ordering is preserved, since the I/O placement is a mapping from their original position into a smaller area. A legalization (step 9) is performed at the end to assure that the I/Os do not overlap.

5 Experimental Setup

The goal is to study the impact of the I/O pin partitioning in the area, number of vias and I/O pin balance. For that, we defined a simplistic 3D placement flow as follows:

1. Initially the I/O partitioning algorithm under study is performed.

2. A min-cut partitioning of Hg into z partitions is performed. The I/O pins, that have already an assigned partition, are used as fixed nodes. The hMetis tool is applied for this step. The tool is configured to keep the area as balanced as possible (maximum 1% unbalance).
3. A tier assignment (similar to the one from [3]) problem maps the sets $P_1, P_2, ..., P_z$ into tiers $t_1, t_2, ..., t_n$. A Simulated Annealing engine is used (see figure 6).
4. Cells could be placed separately in each tier. We skip this step since our goal at this point is to evaluate the number of 3D-Vias.

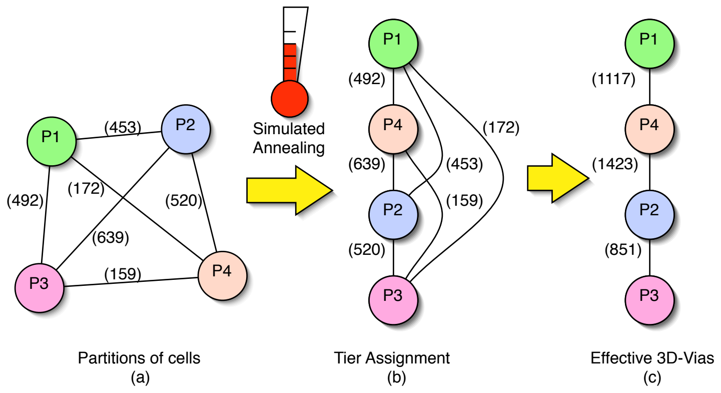

Fig. 6. A group of partitions (a) are assigned to tiers (b) using Simulated Annealing; the effective number of 3D-Vias is shown in (c)

As there is no published previous work on I/O pins handling, the proposed I/O partitioning algorithm is compared with two other simplistic algorithms that follow the same formulation described in section 3. The first algorithm is called *AlternatePins*, on figure 7.(a). This method is a pseudo-random partitioning that goes thought the boundary line of the chip picking nodes for each partition alternatively. The *AlternatePins* replaces steps 1,2 and 3 of the flow keeping steps 4,5,6,7 and 8 untouched in order to maintain the same I/O placement policy.

The idea behind the *AlternatePins* method is to provide an optimal solution in terms of balancing the I/Os. Balancing is important for the subsequent placement stage because the I/Os play a very important role in the quadratic placement engine [4]. This algorithm computes an optimal solution for the cell placement based on attraction forces between connected cells. I/O pins, placed at the boundary, are responsible for the spreading of the cells, since otherwise they would be placed at the center point.

The second method is called *UnlockedPins*, illustrated in figure 7.(c). 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 *UnlockedPins* as well.

The idea behind the *UnlockedPins* method is to provide a favorable solution in terms of 3D-Via minimization. Since hMetis is a leading edge hyper-graph partitioner, it will generate a netlist partitioning with close to optimal number of 3D-Vias. On the other hand, I/O pins will not be spread evenly.

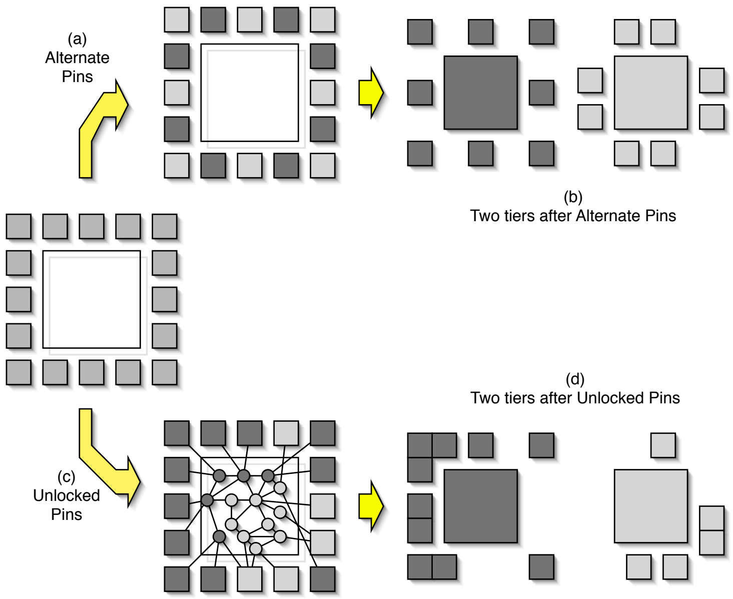

Fig. 7. An illustration of the Alternate Pins algorithm (a) resulting in a two tier circuit (b) with perfectly balance I/O pins; the Unlocked Pins algorithm (b) uses hMetis to partition the whole Netlist, which could result in unbalanced pins (d).

The method proposed here aims at a good solution both in terms of 3D-Vias and balancing. Section (6) presents experimental results comparing the algorithm under these metrics.

6 Experimental Results

6.1 Effect on 3D-Vias

Experiments measuring the amount of 3D-Vias and the balancing of the algorithm are presented in this section. Tables 2, 3 and 4 report our experimental results. ISPD 2004 benchmarks [1] are used targeting circuits with two, three, four and five tiers.

First, table 2 reports the I/O balancing measured by the standard deviation of the number of I/O pins averaged from the whole IBM benchmark suite. The average number of I/O pins from the IBM benchmarks is 264. The method *AlternatePins* delivers the optimal solution while *UnlockedPins* is very unbalanced. In some situations, the strong unbalance practically invalidates the method. The proposed algorithm has close to optimal pin balancing.

Table 2. Comparison of the I/O pins distribution in the tiers considering the three studied algorithms averaged from ibm01 to 1bm18.

# tiers	Algorithm	σ # I/Os
2	Our Algorithm	7
	UnlockedPins	233
	AlternatePins	0.4
3	Our Algorithm	6
	UnlockedPins	252
	AlternatePins	0.4
4	Our Algorithm	5
	UnlockedPins	177
	AlternatePins	0.4
5	Our Algorithm	6
	UnlockedPins	189
	AlternatePins	0.4

Tables 3 and 4 presents our experimental results for the total number of 3D-Vias for the whole IBM benchmark suite. The *AlternatePins* method has the worst results under this metric, which is expected since it is a pseudo-random partitioning. This fact enforces the conclusion that a simplistic I/O partitioning leads to a worse cut size. On the other hand, the method *UnlockedPins*, which was expected to have the best cut among the three methods was outperformed by our algorithm. This fact can be explained by our pre-processing stage that computes the logic distance between I/Os. It seems that the logic distance is a way to summarize the information of the whole graph into a single edge that connects I/O pins (step 2 of the algorithm). Since the graph into this step is very small compared to the whole netlist hyper-graph, the partitioning algorithm (hMetis in this case) could achieve a good partitioning for the pins and for the netlist as well. This computation requires intensive CPU usage. To overcome this problem, the distances are pre-computed and stored in a file so that the I/O partitioning runtimes are not harmed.

Tables 5 and 6 present experimental results for the maximum number of 3D-Vias between pairs of tiers.

Table 3. Total number of 3D vias for the proposed algorithm.

	Our Algorithm # 3D-Vias			
# tiers	2	3	4	5
ibm01	374	525	837	1162
ibm02	396	747	1156	1533
ibm03	1064	2174	2610	3974
ibm04	735	1511	2371	2852
ibm05	2258	4311	6489	9193
ibm06	1059	1642	2934	3477
ibm07	992	2050	3219	4400
ibm08	1298	2697	4018	5346
ibm09	699	1872	2495	3343
ibm10	1490	2661	4004	5216
ibm11	1190	2240	3685	4620
ibm12	2293	4094	6581	8191
ibm13	1042	1893	3099	3742
ibm14	2121	3886	5342	6667
ibm15	3002	4827	7022	9283
ibm16	2102	4316	5774	7172
ibm17	2769	5611	8526	10114
ibm18	1676	3591	4985	6581
Average	1476	2814	4175	5381

6.2 Studding the area effect of 3D-Vias

Table 7 presents an 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 number of 3D-Vias connecting pairs of adjacent tiers; this data is extracted from tables 5 and 6. This number will impact the area requirements for 3D-Vias. The area study supposes 3D-Vias measuring $5\mu m$ and $50\mu m$, which represent a good 3D-Via pitch and a huge 3D-Via pitch respectively.

The following facts can be observed on table 7:

– The *big* 3D-Vias, that could be Bulk based face-to-back vias, suffer from a very high penalty for the 3D-Vias. With 2 tiers, there is a penalty of around 53% of the tier area (note that our algorithm results in less 3D-Vias and also less tier area than the others). For the cases with 4 and 5 tiers, the 3D-Via area is larger than the tier area. The important conclusion here is that when targeting a big via technology it is mandatory to minimize the number of 3D-Vias in order to obtain a feasible solution. As seen in previous tables (5 and 6) the proposed algorithm can save up to 34% which translates to area savings in the order of an entire tier.
– Technologies with small vias suffers from around 2% of the area penalty for the 3D-Vias, leaving room for more 3D-Vias if they are helpful.

Table 4. Comparison of the total number of 3D vias for the three studied algorithms for I/O pin partitioning over the others.

	UnlockedPins # 3D-Vias				AlternatePins # 3D-Vias			
# tiers	2	3	4	5	2	3	4	5
ibm01	441	857	838	1439	428	881	977	1372
ibm02	547	882	1214	1600	503	829	1340	1691
ibm03	1146	2282	2693	4020	1099	2530	3602	4366
ibm04	628	1583	2516	3202	750	1619	2461	4275
ibm05	2417	5372	6653	9651	2576	5428	7037	12400
ibm06	1057	1827	3128	3566	1075	1729	3429	3507
ibm07	880	3242	3302	4605	1049	3423	3482	6523
ibm08	1324	2814	4184	5698	1307	3431	4183	6327
ibm09	806	2828	2763	3518	780	2186	3757	3556
ibm10	1771	3565	4675	7116	1821	4062	4358	8492
ibm11	1490	3477	3958	5697	1494	3629	4923	7437
ibm12	2594	5350	7259	9158	2556	5569	8996	12515
ibm13	1193	3037	3264	4557	1170	2912	4618	4874
ibm14	2171	4561	6584	8085	2310	5090	7564	10113
ibm15	2890	7863	9082	11707	3126	7970	11144	13857
ibm16	2237	5816	6235	9300	2280	6216	9525	10903
ibm17	2539	7695	8733	10845	2847	8402	11420	14080
ibm18	1835	4686	5229	9072	1704	3899	5268	8193
Average	1554	3763	4573	6269	1604	3879	5449	7466
Our Improv.	5.29%	33.74%	9.53%	16.49%	8.72%	37.84%	30.52%	38.73%

6.3 Unbalancing the I/O pins

In the previous section we could observe that there is a trade-off between the I/O pins balance and the resulting number of 3D-Vias. The proposed algorithm for pin partitioning aims at good balance. However, it is well known that a tight balance requirement over-constraints the partitioning process [14]. In the proposed algorithm, the I/O balance can be controlled in step 3 that is performed by hMetis.

HMetis allows the user to configure the balance constraint for each bisection based on equation 11 where u is the unbalance parameter and n is the number of vertices on the hyper-graph.

$$\left[\frac{(50-u)\times n}{100}; \frac{(50+u)\times n}{100}\right] \tag{11}$$

For example, let u = 10, then the bisection balance will range from 40%-60% to 60%-40%. Now suppose that we have four partitions, then an unbalancing factor 10 will result in partitions that can contain between $0.40^2 \times n = 0.15 \times n$ and $0.60^2 \times n = 0.35 \times n$ vertices.

Table 5. Maximum number of 3D-Vias for proposed algorithm.

# tiers	2	3	4	5
ibm01	374	330	370	400
ibm02	396	413	403	594
ibm03	1064	1112	1088	1260
ibm04	735	887	992	887
ibm05	2258	2203	2469	2729
ibm06	1059	849	1135	948
ibm07	992	1332	1433	1524
ibm08	1298	1448	1397	1610
ibm09	699	1057	1008	1075
ibm10	1490	1450	1590	1750
ibm11	1190	1485	1605	1719
ibm12	2293	2278	2422	3173
ibm13	1042	1269	1548	1781
ibm14	2121	2272	2248	2459
ibm15	3002	2857	3199	3395
ibm16	2102	2164	2212	2625
ibm17	2769	3150	3601	3105
ibm18	1676	1871	1754	1782
Average	1476	1579	1693	1823

Our experimental results (averaged from all benchmark circuits) are reported on table 8 and figure 8. Table 8 presents the I/O pin unbalance measured by Standard Deviation. Figure 8 presents the benefits of unbalancing the I/Os to the 3D-Via count.

7 Conclusions

A method for the partitioning and placement of the I/O pins of a 2D block to a 3D circuit was proposed. An interesting analysis in our method lies in the fact that it actually improved the hypergraph partitioning algorithm cut by performing only shortest path analysis. Note that the method works in two phases: first the I/O partitioning considering the logic distances as weights; second, fix the I/Os and perform partitioning of the cells. In the first phase, the I/Os are arranged in a small graph (containing only the I/Os) weighted by the logic distance on the original graph. The edge weights actually contain information of the whole netlist, compressed in the small I/O graph. In the second phase, the whole netlist is partitioned, however some nodes (the I/Os) are fixed reducing the problem complexity and more importantly providing tips to the partitioning algorithm. We conclude that the reduced problem sizes with compressed information of the whole netlist actually improved the partitioning algorithm at the expense of more CPU time.

Empirically, we showed that doing the partitioning of I/O together with the cells (UnlockedPins method) leads to strongly unbalanced number of pins,

Table 6. Comparison of the maximum number of 3D vias for the three studied algorithms for I/O pin partitioning over the others.

	UnlockedPins Max # 3D-Vias				AlternatePins Max # 3D-Vias			
# tiers	2	3	4	5	2	3	4	5
ibm01	441	467	377	573	428	483	406	480
ibm02	547	496	485	552	503	469	498	553
ibm03	1146	1143	1021	1334	1099	1320	1485	1210
ibm04	628	862	1067	1039	750	913	1033	1454
ibm05	2417	2765	2478	2712	2576	2814	2526	3974
ibm06	1057	924	1134	935	1075	915	1193	937
ibm07	880	1980	1510	1525	1049	2050	1590	2402
ibm08	1324	1436	1445	1788	1307	1919	1448	1833
ibm09	806	1598	1092	1249	780	1356	1684	1137
ibm10	1771	1883	1741	1986	1821	2247	1724	2898
ibm11	1490	1909	1810	2230	1494	1856	1802	2610
ibm12	2594	2820	2747	2962	2556	3160	3113	4205
ibm13	1193	1606	1500	1755	1170	1611	1905	1954
ibm14	2171	2375	2307	2881	2310	2619	3274	3283
ibm15	2890	4188	3377	4099	3126	4207	4385	4163
ibm16	2237	3185	2266	3355	2280	3704	3794	3443
ibm17	2539	4165	3526	2990	2847	4539	5245	5053
ibm18	1835	2652	1852	2810	1704	2127	1856	2552
Average	1554	2025	1763	2043	1604	2128	2165	2452
Our Improv.	5.29%	28.24%	4.14%	12.06%	8.72%	34.76%	27.85%	34.51%

which invalidates the method. We also demonstrated the pseudo-random I/O partitioning approaches (such as AlternatePins) leads to a higher number of 3D-Vias. The proposed method demonstrated good effectiveness both in terms of I/O balance and resultant number of 3D-Vias (5% to 33% improvement on 3D-Via count compared to hMetis), outperforming both algorithms in both metrics.

After that, the area impact was studied under our simplified placement flow that minimizes the number of 3D-Vias. It was verified that the area overhead caused by 3D-Vias is prohibitively high for big (50μm pitch) 3D-Vias (in the order of 50% of the active area and up), requiring more research on via minimization methods. On the other hand, for small (5μm pitch) 3D-Vias, the impact was small (around 2% of the active area), leaving room for additional 3D-Vias if it can improve circuit performance. Any intermediary case would be able to trade 3D-Vias for performance limited by the area occupied by the 3D-Vias.

Finally, we investigated ways to further minimize the cut by working with the I/O pin balancing. We relaxed the I/O pin balance constraint keeping the area evenly distributed since the second partitioning process is still highly constrained. Adding up the advantage reported in previous works with the improvements achieved on this paper, we can outperform hMetis partitioning from 5.5% to 34% in average.

Table 7. Comparison of the 3D-Vias Area Impact Considering the Three Algorithms.

# tiers	Algorithm	Area Tier	Max # 3D-Vias	Area 3D-Vias (big - 50μm)		Area 3D-Vias (small - 5μm)	
2	OurAlgorithm	6,934,347	1,476	3,690,000	53%	36,900	1%
3		4,660,116	1,579	3,947,500	85%	39,475	1%
4		3,490,471	1693	4,232,500	121%	42,325	1%
5		2,821,087	1823	4,557,500	162%	45,575	2%
2	UnlockedPins	6,936,553	1,554	3,885,000	56%	38,850	1%
3		4,658,909	2,025	5,062,500	109%	50,625	1%
4		3,481,276	1,763	4,407,500	127%	44,075	1%
5		2,817,413	2,043	5,107,500	181%	51,075	2%
2	AlternatePins	6,926,117	1,604	4,010,000	58%	40,100	1%
3		4,640,572	2,128	5,320,000	115%	53,200	1%
4		3,489,458	2,165	5,412,500	155%	54,125	2%
5		2,816,187	2,452	6,130,00	218%	61,300	2%

Table 8. The Unbalance of the I/O pins measured by the Standard Deviation

	2 tiers	3 tiers	4 tiers	5 tiers
u=1	7	6	5	6
u=10	64	54	41	48
u=25	158	141	100	103

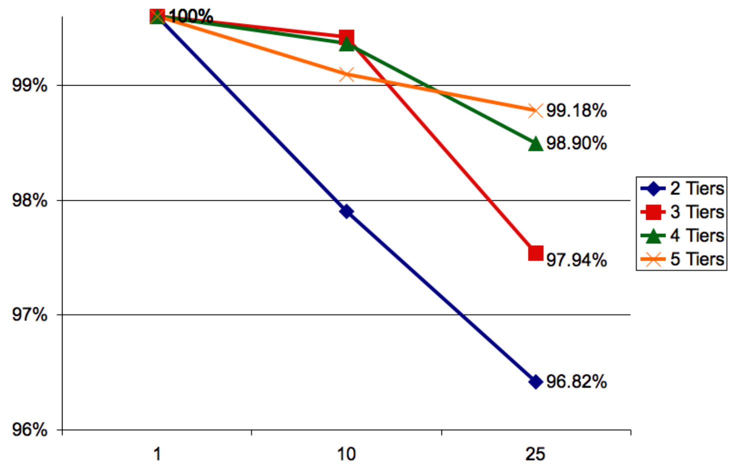

Fig. 8. The percentage improvement on 3D-Via count of unbalancing the I/O pins.

8 Acknowledgments

The authors would like to thank Robert Patti for providing valuable information about Tezzaron technology.

Bibliography

[1] Ispd04 - ibm standard cell benckmarks with pads., 2006.

[2] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. Sapatnekar. Placement and routing in 3d integrated circuits. *Design and Test of Computers*, pages 520–531, Nov-Dec 2005.

[3] C. Ababei, H. Mogal, and K. Bazargan. Three-dimensional place and route for fpgas. In *ASP-DAC '05: Proceedings of the 2003 conference on Asia South Pacific design automation*. IEEE Press, 2005.

[4] C. J. Alpert, T. Chan, D. J.-H. Huang, I. Markov, and K. Yan. Quadratic placement revisited. In *DAC '97: Proceedings of the 34th annual conference on Design automation*, pages 752–757, New York, NY, USA, 1997. ACM Press.

[5] K. Banerjee, S. Souri, 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*, 89:602–633, 2001.

[6] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection alone produce routable placements? In *DAC '00: Proceedings of the 37th conference on Design automation*, pages 477–482, New York, NY, USA, 2000. ACM Press.

[7] S. Das, A. Chandrakasan, and R. Reif. Design tools for 3-d integrated circuits. In *ASPDAC: Proceedings of the 2003 conference on Asia South Pacific design automation*, pages 53–56, New York, NY, USA, 2003. ACM Press.

[8] S. Das, A. Chandrakasan, and R. Reif. Calibration of rent's rule models for three-dimensional integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12:359–366, 2004.

[9] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In *ISPD '04: Proceedings of the 2004 international symposium on Physical design*, pages 108–115, New York, NY, USA, 2004. ACM Press.

[10] W. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. Sule, M. Steer, and P. Franzon. Demystifying 3d ics: The pros and cons of going vertical. *Design and Test of Computers*, pages 498–510, Nov-Dec 2005.

[11] Y. Deng and W. P. Maly. Interconnect characteristics of 2.5-d system integration scheme. In *ISPD '01: Proceedings of the 2001 international symposium on Physical design*, pages 171–175, New York, NY, USA, 2001. ACM Press.

[12] B. Goplen and S. Sapatnekar. Efficient thermal placement of standard cells in 3d ics using a force directed approach. In *ICCAD '03: Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, page 86, Washington, DC, USA, 2003. IEEE Computer Society.

[13] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for producing 3d ics with high-density interconnect, 2005. Available at: ¡http://www.tezzaron.com/¿. Access on: Aug. 2005.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. *IEEE Transactions on Very Large Integration (VLSI) Systems*, 7:69–79, March 1999.

[15] I. Kaya, S. Salewski, M. Olbrich, and E. Barke. Wirelength reduction using 3-d physical design. In *Integrated Circuit and System Design - Power and Timing Modeling, Optimization and Simulation; Proceedings of 14th International Workshop, PATMOS 2004*, 2004.

[16] G. Liu, Z. Li, Q. Zhou, X. Hong, and H. H. Yang. 3d placement algorithm considering vertical channels and guided by 2d placement solution. In *ASICON 2005: 6th International Conference On ASIC*, pages 24–27, 2005.

[17] R. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of IEEE*, 94:1214–1224, 2006.

[18] A. Rahman and R. Reif. System-level performance evaluation of three-dimensional integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8, December 2000.

[19] A. Rahman and R. Reif. Thermal analysis of three-dimensional (3-d) integrated circuits (ics). In *Proceedings of the IEEE 2001 International Interconnect Technology Conference*, pages 157–159, 2001.

[20] T. S. S. Obenaus. Gravity: Fast placement for 3-d vlsi. *ACM Transacions on Design Automation of Electronic Systems*, 8:69–79, March 1999.

[21] N. Viswanathan, M. Pan, and C. C.-N. Chu. Fastplace: an analytical placer for mixed-mode designs. In *ISPD '05: Proceedings of the 2005 international symposium on physical design*, pages 221–223, New York, NY, USA, 2005. ACM Press.