An Algorithm for I/O Pins Partitioning Targeting 3D VLSI Integrated Circuits

Sandro Sawicki[1, 2] sawicki@inf.ufrgs.br
Renato Hentschke[1] renato@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*—This paper shows the impact of I/O pins partitioning on 3D circuits. Previous works on 3D placement did not focused on the I/Os partitioning and placement. This work presents an algorithm based on the logic proximity of the pins, which is used as weights to a min-cut partitioning. Our method calculates the area of[1] the tiers while placing the I/Os on the boundaries. Initial whitespaces and aspect ratio as well as the initial pins orientation and ordering are preserved. The method is compared to two other simplistic methods for pins partitioning. Our experimental results show that our method is efficient since it can balance the I/O pins distribution in the various tiers while leading to improvements in wire length and number of 3D vias.**

I. INTRODUCTION

It is common sense that interconnects optimization is a huge issue for physical design algorithms. High delay, hard manufacturability, power, noise and crosstalk are some of the issues related to wires. As the technology advances, there is an increasing need for research on interconnect optimization. The coming technologies and designs will demand new solutions and the use of 3D VLSI circuits is one possibility to improve interconnects as a new paradigm for high performance VLSI circuits. Major companies like IBM, Intel, Samsung, Micron, Cadence and Infineon are investing in solutions regarding 3D circuits [1].

The idea of partitioning a single random logic block into two or more 3D tiers is starting to be explored [2-6]. In this approach, the Placement stage has to partition the cells and place them in separate tiers. Theoretically [7] and empirically [3] it is shown that 3D circuits can potentially reduce wire lengths. Additionally, the amount of reduction in relation to 2D circuits grows as circuit sizes increases [5].

This paper assumes that the I/O pins delimit the boundary of a random logic block, in 2D circuits. We also assume that the I/Os can be moved to any tier. In order to place circuits in various tiers, some partitioning (and new placement) of the I/O pins must be performed. Many cell placement algorithms are guided by the fixed pins locations. The quadratic placement algorithm [8], for instance, that is largely used in the academia [9] and industry [10] requires a definition of the position of I/O pins positions in order to compute a solution.

This paper studies the I/O pins partitioning problem. It is well known that I/O pins placement is more effective if performed during floorplanning. On the other hand, almost all of nowadays designs and tools target 2D technologies; an automated technique to migrate from a 2D placement to 3D placement can significantly reduce design cycle time Although previous works [2-6] used some criteria to make the partitioning of the I/O boundary, the details of how they were actually performed were omitted in the papers. We assume that simplistic solutions are being adopted. To the authors' knowledge, this work is the first to study the impact of partitioning of I/O pins into two or more tiers in the final solution.

This paper is organized as follows: section II present some of the 3D ICs placement issues and published work on the field. We highlight some of their drawbacks that we are addressing in this paper. Section III formulates the I/O partitioning and placement problem for 3D circuits. Section IV presents an algorithm to solve the problem optimizing the logic proximity between pins in the same tier. The last sections present experimental results and the conclusions of this work.

II. 3D CIRCUITS PLACEMENT

A 3D circuit is constructed by stacking two or more separate 2D circuits [11]. The 3D placement problem, besides placing cells in the 2D space, relies on the partitioning of the cells into different layers as show in figure 1. Space for 3D vias should be properly allocated in order to

[2] On live from UNIJUI, Northwest University of Rio Grande do Sul, DETEC – Departamento de Tecnologia, Santa Rosa, Brazil.

* This research was partially supported by a grant from CNPq, Brazil.

guarantee that there will be enough room for vertical connections. The 3D placer should take advantage of the extended placement space to improve wire length, timing, area and power.

Goplen and Sapatnekar present in [4] a force directed cell placer for 3D ICs with thermal forces in order to improve chip temperature. The thermal issue is a major concern on 3D circuits, since density is higher while insulator layers harden the heat dissipation. The work uses the hMetis algorithm [12] to make the partitioning of the cells into several tiers while the number of 3D vias is minimized.

As the work of Sapatnekar [3][13], the approach in [5] is also based on the force directed placement method, starting from a random solution and improving it based on forces. Both works do not mention how they extended the I/Os from their original 2D arrangement to 3D. They used standard 2D benchmarks in their experiments. Possibly, the I/O pins were ignored since their force directed methods do not require them.

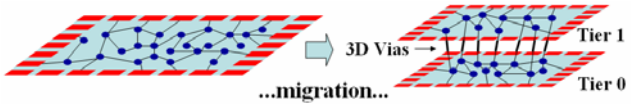

Figure 1. General idea of 3D ICs with 2 tiers

The work in [6] presents a 3D cell placer based on the Capo partitioning approach. The placement flow is very straightforward: first the netlist is partitioned into several tiers and then each tier is placed separately with information of the placed tiers in order to reduce the length of 3D connections.

Most of the previous works agree that the number of connections between tiers must be minimized and most of them use min-cut partitioning, such as hMetis, to accomplish that. Since the 3D vias are very big, via congestion can be improved with a minimization of the number of vias. However, the I/O partitioning will indirectly influence the quality (cut size) of the gate partitioning and so the number of 3D Vias. For this reason, we expect improvements in the number of vias and wire lengths with a good I/O partitioning algorithm.

This paper proposes an algorithm that is based on the logic distance of the I/Os as a criterion for their partitioning. Summarizing our motivation, we want to find a good partitioning method for the I/Os that is able to maintain a good I/O pins balancing leading to area balance between the tiers. At the same time, we indirectly address the minimization of 3D Vias and a reduction of the wirelength.

III. Problem Definition

Before placement, a 2D circuit netlist *NL* composed by a set of gates $G = \{g_1, g_2, g_3, \dots, g_n\}$, a set of I/O pins $P = \{p_1, p_2, p_3, \dots, p_m\}$ and a set of nets connecting them $N = \{n_1, n_2, n_3, \dots, n_o\}$. 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 $\{1,2,\dots,z\}$. 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 $T = \{t_1, t_2, \dots, t_z\}$ (*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:

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

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

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

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

$$\forall (i \in Z) AR_i \approx AR \quad (5)$$

$$\forall (i \in Z)(\forall a \in P_i (OR_i(a) = OR(a))) \quad (6)$$

$$\forall (t \in Z)(\forall a,b \in P_t (OR(a) = OR(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b])) \quad (7)$$

$$\forall (t \in Z)(\forall a,b \in P_t (OR(a) = OR(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b])) \quad (8)$$

In other words, each tier will have its own set of I/O pins and no tier will share an I/O (equations 1 and 2); the whitespace and aspect ratio must be evenly allocated (equations 3,4 and 5); the orientation and ordering of the pins must be preserved (equations 6,7, and 8).

IV. Proposed Algorithm

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

1 Compute $LD(i,j)$ $\forall i,j \in P$

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

3 Perform the partitioning of *PG* into $P_1, P_2, \dots, P_z$ aiming at min-cut optimization and very good number of pins between partitions.

4 $$\forall (i \in Z) GA_i \approx \frac{GA}{z} \quad (9)$$

5 $$\forall (i \in Z) A_i = GA_i \times (1 + S_i) \quad (10)$$

6 $$\forall (i \in Z) W_i = \sqrt{A_i} \times AR_i; H_i = \frac{\sqrt{A_i}}{AR_i} \quad (11)$$

7 $$\forall (i \in Z)\forall (p \in P_i) X_i[p] = x_{ini} + \frac{(X[p] - x_{ini}) \times W_i}{W} \quad (12)$$

8 $\forall(i \in Z)\forall(p \in P_i)Y_i[p] = y_{ini} + \frac{(Y[p] - y_{ini}) \times H_i}{H}$ (13)

9 Legalize I/O positions. (14)

The first step of our algorithm is illustrated in figure 2. 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 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 *LD* are used to create a *PG* graph connecting all pairs of I/O pins, as shown in figure 2. For the third step, we used hMetis tool [14]. 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.

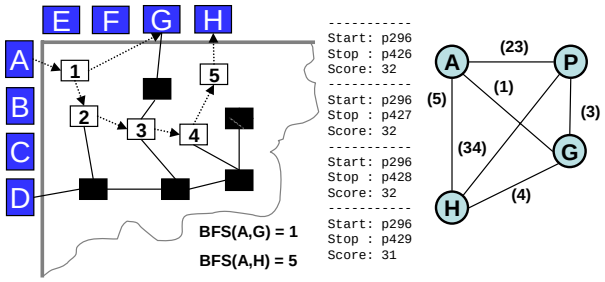

Figure 2. Ilustration of the shortest path between two I/O pins and a portion of the correspondent complete graph of all boundary pins.

The forth step can be accomplished by the division of the total gate area per 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 the aspect ratio and whitespace is preserved from the original 2D circuit. At this point, a 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/Os placement is a mapping from their original position to a smaller area. A legalization (step 9) is performed in the end to assure that the I/Os do not overlap.

V. Experimental Results

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

1. A min-cut partitioning of *HG* is performed. The I/O pins, that have already an assigned partition, are used as fixed nodes. We use hMetis for this step. The tool is configured to keep the area as balanced as possible.
2. A separate set of benchmarks is generated and placed independently. The number of 3D vias is accounted for, but vertical connections are ignored by the placer at this time. Our cell placer uses Quadratic Placement for global placement and Simulated Annealing for detailed placement.

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 1,2,3 and 4 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 1,2,3 and 4 of our flow, but steps 5,6,7,8 and 9 are done.

Tables 1, 2 and 3 report our experimental results. We used ISPD 2004 benchmarks [15]. In this paper we focus on experiments with two tiers, but our algorithm could be applied for any number of tiers. The column “tier area” is calculated after the actual partitioning of the gates. We got numbers very close to the ones suggested by the step 4 of the algorithm. The worst tier area is used as the area of all tiers. The total area is simply twice the tier area. The total WL column is simply the sum of the wirelength found by our placer in each of the separate tiers (not considering vertical connections). The number of I/Os and vias are also reported in the table. The table also shows the standard deviation of the number of I/Os in order to evaluate how good their balance is.

Analyzing the tables 1, 2 and 3 the following topics can be observed:

- The number of I/Os is very well balanced on the *alternate_pins* method (average standard deviation is less than 1 pin). The balancing of our algorithm with an average standard deviation of 5 pins. However, the ***unlocked_pins* algorithm led to very unbalanced number of pins that completely invalidates the method** (average standard deviation of 150 pins).
- The *alternate_pins* method derive a slightly better area (1% improvement) compared to both of the other algorithms. Possibly, the good I/O helps hMetis to achieve a better balancing of the gate area.
- The number of vias found by the *alternate_pins* method is always worse than the others (by an average of 30 vias), showing that **a simplistic I/O pin partitioning will deteriorate the min-cut algorithm**.

- The number of vias obtained with our algorithm is always better than in the other methods. This is a very important result showing that **our I/O partitioning method aids the gate partitioning to achieve a better cut while effectively balancing the I/O pins**.
- The wirelength resulted from our partitioning followed by 2D placement is smaller in average than the *alternate_pins* method while the *unlocked_pins* method led to the best wirelength.

It is important to point-out that the wirelength is not an accurate metric since vertical connections are not accounted for. For this reason, the advantage got by our method will increase since the amount of vertical connections resulted by our partitioning is smaller.

TABLE I. EXPERIMENTAL RESULTS USING OUR I/O PARTITIONING ALGORITHM TARGETING 2 TIERS.

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

TABLE II. EXPERIMENTAL RESULTS USING UNLOCKED_PINS PARTITIONING ALGORITHM TARGETING 2 TIERS.

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

TABLE III. EXPERIMENTAL RESULTS USING ALTERNATE_PINS PARTITIONING ALGORITHM TARGETING 2 TIERS.

Circuit Data						Partitioning Data				
Name	**#cells**	**#I/Os**	**#nets**	**2D area**	**Tier Area**	**Total WL**	**#I/Os tier 0**	**#I/Os tier 1**	**σ #I/Os**	**#Vias**
ibm01	12,506	246	14,111	2,380,800	1,182,416	2.35E+06	123	123	0	429
ibm02	19,342	259	19,584	3,064,208	1,517,568	4.30E+06	130	129	1	477
ibm03	22,853	283	27,401	3,751,968	1,865,920	6.13E+06	142	141	1	1,117
ibm04	27,220	287	31,970	4,782,848	2,375,760	7.54E+06	144	143	1	751
ibm06	32,332	166	34,826	4,106,592	2,080,464	7.37E+06	83	83	0	1,132
ibm07	45,639	287	48,117	7,136,672	3,588,624	1.18E+07	144	143	1	1,065
ibm08	51,023	286	50,513	7,403,840	3,697,920	1.11E+07	143	143	0	1,301
ibm09	53,110	285	60,902	8,617,104	4,307,568	1.43E+07	143	142	1	787
Avg.	33,002	264	35,928	5,155,504	2,577,030	8.10E+06	132	131	0,63	882

VI. CONCLUSIONS

This paper presented a method for the partitioning and placement of the I/O pins of a 2D block to a 3D circuit. To the authors' knowledge, this is the first paper to address this problem and to study its impact on the circuit area, pin balancing and wirelength. We propose 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. The experimental results show that the method is efficient since it can balance the I/O pins distribution in the various tiers while leading to improvements in wirelength and number of 3D vias compared to a simplistic approach.

VII. FUTURE WORK

In future work we first will address balancing of I/O pins and cells. In this paper, the balancing was considered as a constraint to the partitioning algorithm, while a small relaxation could lead to better results.

Also, in future work we will report experiments with an improved 3D placer that will consider vertical connections and their feasibility in terms of 3D via congestion.

VIII. REFERENCES

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

[2] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer and P. D. Franzon; Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design and Test of Computers – special issue on 3D integration*; pp 498-510, Nov.-Dec. 2005.

[3] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan and S. Sapatnekar. Placement and Routing in 3D Integrated Circuits. *IEEE Design and Test of Computers – special issue on 3D integration*; pp 520-531, Nov.-Dec. 2005.

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

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

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

[7] K. Banerjee and S. Souri and P. Kapur and K. Saraswat. 3D-ICs: A Novel Chip Design for Improving Deep Submicrometer Interconnect Performance and Systems on-Chip Integration. *Proceedings of IEEE*, vol 89, issue 5, 2001.

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

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

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

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

[12] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. *In Proceedings of 34th Annual Conference on. Design Automation*, DAC 1997, pages 526–529, 1997.

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

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

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