Tile-Graph-Based Power Planning

Jyh Perng Fang and *Sao Jie Chen*
Graduate Institute of Electronics Engineering and
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: csj@cc.ee.ntu.edu.tw

Abstract

In this paper, we introduce a tile-graph-based approach to power planning. For a given flooplan solution, the power inputs are modeled into a tile graph, the minimum capacity of each power input and the maximum power need of each module in a floorplan are accumulated in an associated tile. An efficient cost evaluation algorithm is adopted to calculate the cost of power planning. As its computation time is quite short, it is reasonable to integrate such an algorithm into an iterative floorplanning environment.

1. Introduction

With the progress of modern VLSI technology, the chip density has beeen highly raised and the voltage of power supply is getting lower. As the power lines are resistive, a longer power line will proportionally cause larger voltage drop, which will decrease the noise margin. Besides, longer power line tends to cause electromagnetic interference, especially for a high frequency circuit. Therefore, sustaining the voltage of power supply from source to sink becomes an important issue for designers.

As an early physical design stage, floorplanning is used to plan the location and aspect ratio of modules in a chip. Traditionally, area is the most concerned issue, there have been different structures [4-6] presented for area minimization. These structures can be classified as slicing structures and non-slicing structures. To decrease design iteration, the wire planning and the buffering which were conventionally processed during routing stage are now considered at floorplanning [1, 2]. However, most of the previous work did not consider the power planning during the floorplanning stage, which implicates that a post-process for power planning is necessary, and we may have no solution because of the constraints caused by area packing.

Instead of post-floorplanning power planning, Liu *at al.* [7] integrates power planning into floorplanning. They assigned local power lines to power bumps on an array-like power bump architecture to shorten the current path from power bumps to circuit blocks. The assignment problem was transformed into a network-flow problem, and then a max-flow algorithm was used to optimally solve the problem. However, the power supply in [7] is confined as an array like power-bump architecture, and the additional computation time is not negligible when integrating the power planning into a floorplanning algorithm.

In addition to relax the constraint on the power supply architecture as in [7], we introduce a more efficient power planning model, which facilitates the integration of power planning and floorplanning. We transform the power inputs into a tile graph, the power supply of power inputs and the power demand of circuit blocks are accumulated in each tile. Based on the tile graph, an efficient algorithm is proposed to promptly obtain the cost of local power lines for each floorplan solution.

To show the effctiveness of our power-planning algorithm, we respectively integrate this algorithm into a non-slicing and a slicing floorplanners, and the experimental results showed are very competent.

This paper is organized as follows. Section 2 introduces the problem formulation. Section 3 describes the algorithm of cost evaluation of local power lines. The experimental results on the floorplanning platform are presented in Section 4. The last section draws a conclusion.

2. Problem Formulation

The architecture of power supply considered in our work is either flip-chip type or traditional wire-bond type. The former is a flip chip packaged on the top of a circuit die, bumps on the flip chip are arranged like an array; the latter uses bonding wire for connecting pads on the die to chip I/O pins. For convenience, the power bumps of flip-chip type and the power pads of wire-bond type are termed as power inputs hereinafter. The power supplied by each power input is assumed to be different as a multi-voltage power supply may exist. Besides, the maximum power demand of a circuit block is assumed to be proportional to the area it occupied. The power planning problem is thus formulated as follows:

Given: A floorplan of circuit blocks $B = \{b_1, b_2, \ldots, b_m\}$, and a set of power inputs $P = \{p_1, p_2, \ldots, p_n\}$; the maximum power demand of each circuit block is d_i, and the minimum power that each power input can afford is s_i.

Goal: Find an assignment of power inputs to circuit blocks such that the power demand is satisfied and the total length of local power lines is minimized.

Some terminologies are defined in the following.

Tile Graph: A tile graph $G(V,E)$ contains a set of tiles V and a set of edges E. For any two neighboring tiles u and v, there exists an edge $e_{u,v}$ connecting these two tiles. Each tile v has a weighting $w(v)$, and each edge has a weighting $w(e_{u,v})=1$.

Tile Distance (*TD*): For any two tiles on a tile graph, the tile distance is the number of edges between them.

Local Power Lines: power lines that deliver current from power inputs to circuit blocks.

Positive (Negative) Tile: a tile has positive (negative) weighting.

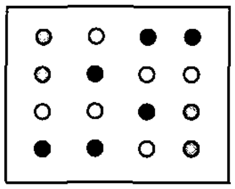
Balanced Tile: a tile with zero weighting.

3. Successive Elimination Method

In this section, the procedure of building the tile graph and a tile-graph-based Successive Elimination algorithm are introduced.

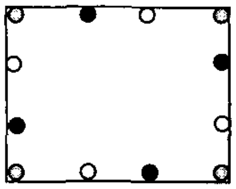
3.1. Building a Supply-Demand Tile-Graph

For a given set of power inputs $P = \{p_1, p_2, \ldots, p_n\}$, we can build a tile graph $G(V,E)$ according to the spatial relationships of power inputs, in which each tile represents a power input and is weighted by the minimum power s_i that the corresponding power input can afford. Fig. 3.1(a) shows the power inputs on a flip-chip architecture, in which circles in dark gray and light gray represent power inputs of different voltages, and circles in white represent bumps for conveying signals rather than power. The tile graph corresponding to Fig. 3.1(a) is represented as a 2-D array, as shown in Fig. 3.1(b), in which weighting on each tile represents the power capacity of its corresponding power input. Similarly, power inputs on a wire-bond type package is illustrated in Fig. 3.1(c), the power of each corresponding power input is represented as a tile weighting in Fig. 3.1(d).


(a)

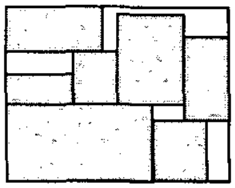
6	0	8	8
6	8	0	0
0	0	8	6
8	8	0	6

(b)


(c)

6	8	0	6
0	0	0	8
8	0	0	0
6	0	8	6

(d)


(e)

-4	-2	-3	-1
-2	-4	-4	-4
-4	-4	-4	-4
-4	-4	-4	-1

(f)

Figure 3.1 (a) Bumps of a flip-chip type package, and (b) their power inputs in a tile graph; (c) Pads of a wire-bond type package, and (d) their power inputs; (e) A floorplan, and (f) its power demands.

Analogously, given a floorplan of circuit blocks, we can build a tile graph according to the area occupation of circuit blocks, in which the size of tile graph is the same as that built for power inputs. Fig. 3.1(e) shows a given floorplan, and Fig. 3.1(f) its corresponding tile graph, in which negative weightings highlight the power demands of circuit blocks.

Respectively combining the tile graphs of power inputs in Fig. 3.1(b) and Fig. 3.1(d) to that of the floorplan in Fig. 3.1(f), we obtain two supply-demand tile graphs, as shown in Fig. 3.2(a) and Fig. 3.2(b). In the above tile graphs, a positive weighting indicates that the corresponding power input has more power capacity than needed, and a negative weighting suggests that the circuit blocks corresponding to the tiles need power supplied by other power inputs located at other tiles. And whenever the supply and demand relation of a tile is balanced, the corresponding tile will have a zero weighting.

2	-2	5	7
4	4	-4	-4
-4	-4	4	2
4	4	-4	5

(a)

2	6	-3	5
-2	-4	-4	4
4	-4	-4	-4
2	-4	4	5

(b)

Figure 3.2 Two supply-demand tile graphs.

3.2. Successive Elimination

On a tile graph, the length of a local power line is represented as a Tile Distance (*TD*). For a power line delivering from a power input to the circuit blocks located at the same tile, its *TD* is 0; while the *TD* of a power line delivering from a power input located at one tile to circuit blocks located at its neighboring tile is 1. To summarize, the length of a power line delivering from a power input to a circuit block located at different tile is represented as the *TD* between these two tiles.

As the supply-demand relationship of a power input and circuit blocks at the same tile has been extracted and merged as a tile weighting of the tile graph, the problem of supply-demand between power inputs and circuit blocks is simplified as balancing the supply-demand between different tiles.

Assuming that the weight of a positive tile *PTi* is *w(PTi)* and the weight of a negative tile *NTj* is *w(NTj)*, the supply-demand relationship of the tile graph with *n* tiles possesses two properties:

Capacity constraint: The weighting sum of all negative tiles should be smaller than that of all positive tiles,

$$\sum_{j\le n} w(NTj) \le \sum_{i\le n} w(PTi).$$

Flow conservation: The total current flow out from the positive tiles should be equal to that flow into the negative tiles. Representing the current flow out as $\Delta w(PTi)$ and the current flow in as $\Delta w(NTj)$, this property can be formalized as:

$$\sum_{j\le n} \Delta w(NTj) = \sum_{i\le n} \Delta w(PTi).$$

A Successive Elimination (SE) method is thus proposed to solve this problem. It successively balances the positive tiles and the negative tiles by invoking an Elimination (EL) algorithm, which is used to eliminate the weighting *w(PTi)* of a given positive and the weighting *w(NTj)* of a given negative tile, and then accumulate the cost according to the amount of elimination. As shown in Fig. 3.3, Step 1, Step5, and Step 8 handle the conditions of $|w(PTi)| > |w(NTj)|$, $|w(PTi)| = |w(NTj)|$, and $|w(PTi)| < |w(NTj)|$, respectively.

Adding a negative weighting *w(NTj)* to a positive weighting *w(PTi)* implies routing power lines from the power input located at a positive tile *PTi* to circuit blocks located at a negative tile *NTj*. Besides, the length of needed local power lines is proportional to the amount of elimination. Therefore, for a given power input and the circuit blocks at a given tile, the length of local power lines is estimable according to the amount of elimination and the *TD* between the positive tile and the negative tile, *i.e.*, *TD(i,j)*.

```
Algorithm EL
Assume $w(PTi)$ = weight of a positive tile $PTi$, $w(NTj)$ =
weight of a negative tile $NTj$, and $AD$ = allowed distance
from $PTi$ to $NTj$ for elimination.
1. if   ($w(PTi)+ w(NTj) > 0$)    // Case 1
2.        $w(PTi) = w(PTi) + w(NTj)$;
3.        $w(NTj) = 0$;
4.        $Cost = Cost - AD \times w(NTj)$;
5. else if   ($w(PTi)+ w(NTj) == 0$)    // Case 2
6.        $w(PTi) = w(NTj) = 0$;
7.        $Cost = Cost + AD \times w(PTi)$;
8. else    // Case 3
9.        $w(PTi) = 0$;
10.       $w(NTj) = w(PTi) + w(NTj)$;
11.       $Cost = Cost + AD \times w(PTi)$;
```

Figure 3.3 Algorithm Elimination.

```
Algorithm SE
In a tile graph with $n$ tiles, $TD(i,j)$ = distance between
tiles $PTi$ and $NTj$, $1 \le i, j \le n$.
1. $Cost = 0$; $AD=1$;
2. do
3.    for each positive tile $PTi$
4.       while(($w(PTi) > 0$) && (there exists a
              negative tile $NTj$ s.t. $TD(i,j)$ equals $AD$ ))
5.          call algorithm EL;
6.    $AD = AD + 1$;
7. Until no negative tile exists;
```

Figure 3.4 Algorithm Succesive Elimination.

The SE method runs iteratively. At each iteration, for each positive tile, only negative tiles at a distance of AD are permitted for elimination. The algorithm starts from the shortest AD (*i.e.* $AD = 1$), and then AD is increased by 1 for each next iteration. The algorithm in Fig. 3.4 describes the method, where Step 3 successively finds a positive tile; Step 4 and Step 5 take the neighboring negative tiles at a distance of AD for elimination.

Taking Fig. 3.5 as an example, the tile graph in Fig. 3.5(a) is eliminated to the tile graph of Fig. 3.5(b) at the first iteration and then eliminated to the tile graph of Fig. 3.5(c) at the second iteration. In Fig. 3.5(a), considering the shaded tiles, the weighting of tile located at upper left corner is 6, while the tile weightings at the right side and the lower side are −4 and −3, respectively. Adding −4 to 6, corresponding to Case 1 of Fig. 3.3, the tile weighting 6 and tile weighting −4 are eliminated to 2 and 0, respectively. Then adding −3 to 2, corresponding to Case 3 of Fig. 3.3, the tile weighting 2 is eliminated to 0 and the tile weighting −3 is eliminated to −1. For each positive tile, we take the negative tiles at a distance of AD for elimination. In Fig. 3.5(a), the eliminations occur between tiles at the first iteration is grouped in bold frame. After one iteration, as shown in Fig. 3.5(b), SE is repeated with AD increased, and the tiles qualified for elimination are shaded. The final tile graph after SE is depicted in Fig. 3.5(c).

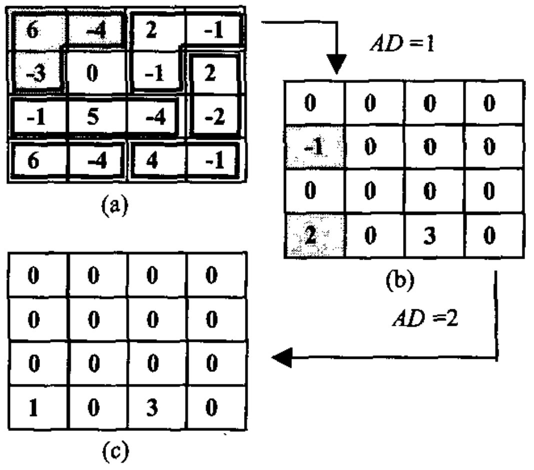


Figure 3.5 Example of SE method: (a) a supply-demand tile graph, (b) resultant tile graph at $AD = 1$, and (c) final tile graph at $AD = 2$.

3.3. Variation of SE Algorithm

At each iteration, algorithm SE successively searches for a positive tile and eliminates its nearest negative tiles. Alternatively, it is feasible to successively search for a negative tile, and then eliminate its nearest positive tiles. For these two options, the former is preferred as generally there are more negative tiles than positive tiles on a supply-demand tile graph.

For a large floorplan, it is possible to further enhance the computation speed of power planning. The SE algorithm is optionally revised with a bounded Maximumly Allowed Distance for elimination (*MAD*), i.e., the SE iterates until $TD(i,j)$ reaches *MAD* rather than until all negative tiles are eliminated. With the constraint of *MAD*, a floorplan violating the length constraint of local power lines can be filtered out more efficiently. Besides, there are two options to integrate the SE algorithm into a floorplanning algorithm based on simulating annealing. One option is to add the length cost of local power lines into the cost function, $cost(f)=A + \beta \times cost(p) + \lambda \times cost(s)$, in which β and λ are adjustable parameters; A is the area cost; cost(p) is the cost of local power line, cost(s) is other evaluation factors such as total wire length, and/or buffer cost. Another option is to evaluate whether the local power line of a floorplan with better cost has any length violation.

3.4. Analysis of SE Algorithm

For a network with V vertices and E edges, Ford-Fulkerson method runs in time $O(E|f^*|)$, where $|f^*|$ is the execution times of iteration to find the max flow, while Edmonds-Karp algorithm runs in $O(VE^2)$ time [3]. Moreover, Liu *at al.* [7] transform the problem of power planning into a network-flow problem, and then solve the problem in at least $O(m \log m)$ time, where m is the number of power inputs.

Comparatively, given a tile graph, each positive tile needs at most $4 \times AD$ times to run the EL algorithm, where the run time of EL is a constant. For a floorplan with m power inputs, as each iteration of the SE algorithm needs at most $O(m)$ time, and the upper bound of AD is $m^{1/2}$, SE needs at most $O(m^{3/2})$ time. And as the the length of a local power line can be constrained at certain *MAD*, the iteration times of SE can be further improved as a constant, and the run time becomes $O(m)$.

4. Experimental Results

The SE algorithm has been integrated into a non-slicing floorplanner based on B*-tree [6]. The code is implemented in C++ on a Sun Sparc machine with 240MB memory. We tested the code on the MCNC benchmark circuits. The power and ground of each circuit block are separately processed before combined into a supply-demand tile-graph. The test results are summarized in Table 4.1 and Table 4.2, in which the listed time and area are taken from the best of 20 runs.

Table 4.1 Test results on a B*-tree floorplanner.

Test circuits	blocks	without power planning		with power planning	
		time(s.)	dead space(%)	time(s.)	dead space(%)
ami33	33	6.1	4.58	12.9	4.75
ami49	49	9.8	4.61	27.0	4.64
apte	9	0.6	2.03	2.2	6.37
hp	11	0.8	4.61	3.0	4.61
xerox	10	0.1	3.71	2.3	5.93

In Table 4.1, the supply-demand tile-graph is assumed to be 6×6 and *MAD* is assumed to be 3, *i.e.*, a given floorplan is determined as failed if the length of any local power line is longer than 3 tile units. The results reveal that the execution time is fast and the penalty of additional area is negligible.

Table 4.2 Test results of various tile sizes and *MADs*.

Tile size	*MAD*	ami33		ami49	
		time(s.)	dead space (%)	time (s.)	dead space (%)
5×5	2	12.9	5.98	35.5	4.38
	4	12.9	5.98	35.8	4.38
8×8	2	21.0	4.06	40	5.07
	7	21.8	4.06	40.2	5.04

In Table 4.2, after integrating SE algorithm into a B*-tree floorplanner, the resultant area and execution time of *ami*33 and *ami*49 are listed according to different tile sizes and *MADs*. Again, the results show the effectiveness and efficiency of our SE algorithm.

To further examine our algorithm, we integrate SE algorithm into a slicing floorplaner based on W-L algorithm [4], and run the floorplanner on a Pentium II-300 PC with 128M memory. As shown in Table 4.1, the supply-demand tile-graph is 6×6 and *MAD* is 3. The resultant area and execution time are listed in Table 4.3, which show that, even on a PC, a floorplanner integrating our algorithm can have an acceptable solution in a reasonable time.

Table 4.3 Test results on a W-L floorplanner.

Test circuits	without power planning		with power planning	
	time (.s)	dead space (%)	time (s.)	dead space (%)
ami33	8.52	7.00	26.59	7.00
ami49	10.50	7.06	33.35	7.60

5. Conclusion

The usefulness of a floorplanner depends not only on the ability of area packing, but also on the ability of preventing the design from failure during later design stage. Moreover, the operation speed of a floorplanner will be slowed down if an additional planning such as power planning has to be included in each evaluating iteration of the floorplanning algorithm.

We proposed an approach such that a floorplanner can simultaneously estimate both the area and the power for each floorplan solution. The experimental results show the reasonable efficiency of our approach in practice and thus the effectiveness of its integrating into a floorplanning algorithm. Besides, our approach is available both on slicing structure and non-slicing structure rather than being confined to any specified data structure.

Acknowledgments

This work was supported by the National Science Council, R.O.C., under Grant NSC91-2215-E002-042. The authors would like to thank Prof. Yao-Wen Chang not only for providing the code of B*-tree but also for his very helpful opinions.

References

[1] J. Cong, T. Kong, and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning," *Proc. ICCAD*, pp. 385-363, 1999.

[2] C. Alpert, J. Hu, S. Sapatnekar, and P. Villarrubia, "A practical methodology for early buffer and wire resource allocation," *Proc. DAC*, pp. 189-194, 2001.

[3] Cormen, Leiserson, and Rivest, *Introduction to Algorithms*, McGraw Hill/The MIT Press, 1990.

[4] D. F, Wong, and C. L. Liu, "A new algorithm for floorplan design," *Proc. DAC*, pp. 101-107, 1986.

[5] H. Murata, E. S. Kuh, "Sequence-pair based placement method for hard/soft/preplaced modules," *Proc. Intl. Symp. Physical Design*, pp. 167-172, 1998.

[6] Y.C.Chang, Y.-W. Chang, G.-M. Wu, and S.-W.Wu, "B*-tree: A new Representation for Non-Slicing Floorplans," *Proc. DAC*, pp. 458-463, 2000.

[7] I.-M. Liu, H.-M. Chen, T.-L. Chou, Adnan Aziz, D. F. Wong, "Integrated Power Supply Planning and Floorplanning," *Proc. ASP-DAC*, January 2001.