NTUplace: A Ratio Partitioning Based Placement Algorithm for Large-Scale Mixed-Size Designs *

Tung-Chieh Chen†, Tien-Chang Hsu†, Zhe-Wei Jiang†, and Yao-Wen Chang†‡
Graduate Institute of Electronics Engineering†
Department of Electrical Engineering‡
National Taiwan University
Taipei 106, Taiwan
{donnie, tchsu, crazying}@eda.ee.ntu.edu.tw; ywchang@cc.ee.ntu.edu.tw

ABSTRACT

In this paper, we present a hierarchical ratio partitioning based placement algorithm for large-scale mixed-size designs. The placement algorithm consists of three steps: global placement, legalization, and detailed placement; it works in a hierarchical manner and integrates net-weighting partitioning, whitespace management, look-ahead bipartitioning, and fast legalization to handle the large-scale mixed-size placement problems. Unlike the traditional partitioning-based technique that is based on balanced partitioning, we apply ratio partitioning in each level. Further, applying the look-ahead bipartitioning technique in each level, we can evaluate the feasibility of the placement for sub-partitions more accurately. Therefore, we can find better ratios for the partitions, leading to easier legalization for the global placement result and finally a better detailed placement solution. Experimental results show the efficiency and effectiveness of our algorithm.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids [Placement and Routing]

General Terms: Algorithms, Performance

Keywords: Placement, Mincut, Ratio Cut

1. INTRODUCTION

As the process technology advances, the feature size is getting smaller and smaller, which makes it possible to integrate an entire system with one billion transistors on a single chip. At the same time, the Intellectual Property (IP) modules and pre-designed macro blocks (such as embedded memories, analog blocks, pre-designed datapaths, etc.) are often reused, and thus many IC designs contain hundreds of thousands of modules with different sizes. Hence, we need an efficient algorithm to solve such a placement problem with a large number and hybrid sizes of modules and cells.

Traditional standard-cell placement techniques assume that cells have the similar sizes. Thus, they cannot handle problems with different sizes well. Many approaches have been presented to handle such a mixed-size placement problem. Those approaches can be classified into three categories: (1) the analytical-based approach [8, 9, 10], (2) the simulated-annealing-based approach [5], and (3) the partitioning-based approach [3, 4, 11]. Among them, the analytical-based and simulated-annealing-based approaches are often too slow to handle very large problem sizes. Therefore, we shall develop our placement algorithm based on the partitioning-based approach.

In this paper, we present a hierarchical ratio partitioning based placement algorithm for large-scale mixed-size designs. The placement algorithm consists of three steps: global placement, legalization, and detailed placement; it works in a hierarchical manner and integrates net-weighting partitioning, whitespace management, look-ahead bipartitioning, and fast legalization to handle the large-scale mixed-size placement problems. Unlike the traditional partitioning-based technique that is based on balanced partitioning, we apply ratio partitioning in each level. Further, traditional partitioning-based methods often do not consider the feasibility of the placement for sub-partitions. Applying the look-ahead bipartitioning technique in every level, in contrast, we can evaluate the feasibility of the placement for sub-partitions more accurately. Therefore, we can find better ratios for the partitions, leading to easier legalization for the global placement result and finally a better detailed placement solution. Experimental results show the efficiency and effectiveness of our algorithm.

2. PROPOSED ALGORITHM

Our placement algorithm adopts the top-down recursive ratio bipartitioning technique. The algorithm consists of three steps: global placement (GP), legalization (LG), and detailed placement (DP) (see Figure 1). To improve the solution quality, we apply the following techniques to perform global placement and legalization: (1) net-weighting partitioning (GP), (2) whitespace distribution (GP), (3) look-ahead bipartitioning (GP), and (4) fast legalization (LG). The result is then fed into an existing detailed placer to further refine the placement solution. We shall explain these techniques in the following subsections.

2.1 Net-Weighting Partitioning

At the initial top-level, the locations of all blocks are set to the center of the chip region. To prevent from generating sub-regions of large aspect ratios, we choose the longer side

*This work was partially supported by MediaTek, Inc. and National Science Council of Taiwan under Grant No's. NSC 93-2215-E-002-009, NSC-93-2220-E-002-001, and NSC 93-2752-E-002-008-PAE.

ISPD'05, April 3–6, 2005, San Francisco, California, USA.
ACM 1-59593-021-3/05/0004.

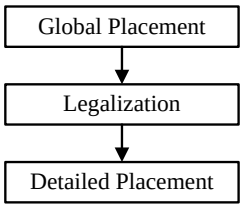

Figure 1: The flow of our algorithm.

to divide the region into two sub-regions. After the shapes of two sub-regions are determined, we move the blocks to the two centers of the two sub-regions to minimize the half-perimeter wirelength (HPWL).

The block-location determination problem can be formulated as a hypergraph partitioning problem. Then, the hypergraph is partitioned using a *weighted-net*, min-cut bipartitioner to obtain the minimum HPWL. The new locations of the blocks are determined by the partitioning. Each partition corresponds to a sub-region.

The circuit is modeled as a hypergraph. Each node in the hypergraph corresponds to a block inside the region, with the node weight being set to the area of the corresponding block. Each hyperedge corresponds to a multi-terminal net in the circuit, with the hyperedge weight being set to the value of the HPWL contribution if the hyperedge is cut. The hyperedge weight is determined as follows. Take Figure 2 as an example for easier explanation. Without loss of generality, we assume that the region R is divided vertically into two sub-regions, R_1 and R_2, with p_1 and p_2 being the respective centers of the two sub-regions. t_1 and t_2 are two external terminals outside the region R. In Figure 2(a), if a net connects with only blocks inside the region R, the weight of the corresponding hyperedge is set to the distance between p_1 and p_2, denoted by $dist(p_1, p_2)$, since the HPWL value equals $dist(p_1, p_2)$ if the net is cut. In Figure 2(b), if a net connects with external terminals t_1 and t_2, and the terminals are outside the horizontal range of p_1 and p_2 and closer to p_1, the hyperedge weight is set to $dist(p_1, p_2)$, and we add a fixed dummy node with weight equal to 0 at p_1 and connect the net to the dummy node. The rationale is that the HPWL is smaller if all blocks connecting the net are located at p_1, and we will favor the blocks connecting with the net to be moved to the left partition during the partitioning because of the fixed dummy node. In Figure 2(c), a net connects with external terminals t_1 and t_2, and the terminals are inside the horizontal range of p_1 and p_2. If all blocks connecting the net are located at p_1, the HPWL equals $dist(p_1, t_2)$. If all blocks connecting the net are located at p_2, the HPWL is $dist(p_2, t_1)$. Since $dist(p_2, t_1) > dist(p_1, t_2)$, we add a fixed dummy node with weight equal to 0 at p_1 and connect the net to the dummy node. The hyperedge weight is set to the value of $dist(p_2, t_1) - dist(p_1, t_2)$, which is the HPWL increase if the net is cut during the partitioning. The rationale is that the HPWL is smaller if all blocks connecting with the net are located at p_1, and we will favor the blocks connecting with the net to be moved to the left partition during the partitioning because of the fixed dummy node. In Figure 2(d), a net connects with the external terminals t_1 and t_2, and the terminals are on different sides of the range of p_1 and p_2. The hyperedge weight is set to 0 because the block locations will not affect the HPWL. By using the above net-weighting scheme, the cut size is proportional to the HPWL, and finding the min-cut bi-partitioning leads to the assignments of block locations with the minimal HPWL.

2.2 Whitespace Distribution

If a large block exists in the region, we cannot always use a balanced region bipartition, or the block may not fit into any sub-region. In this case, we have to perform *ratio partitioning*. We add a dummy node connecting to the node corresponding to the large block, and set the net-weight to a large negative value. As the example shown in Figure 3, if we want to obtain a ratio partition of w_1/w_2 (or w_2/w_1), where w_1 and w_2 are the expected widths for the two sub-regions, the weight of the dummy node shall be set to $A|w_1 - w_2|/(w_1 + w_2)$, where A is the total area of the blocks inside the region. Since the net-weight of the edge connecting the dummy node is a large negative value, the min-cut bi-partition will cut the edge. Since the dummy node does not occupy any area in the region, it results in a ratio partition, and the cut-line moves from c_1 to c_2.

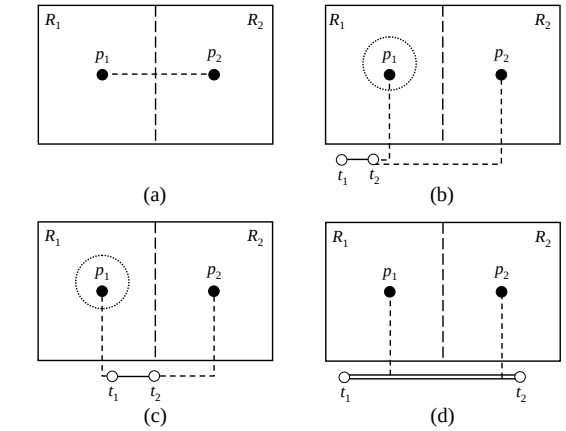

Figure 2: An example of determining a net weight. Here, p_1 and p_2 are the respective centers of the two sub-regions R_1 and R_2, and t_1 and t_2 are external terminals outside the region.

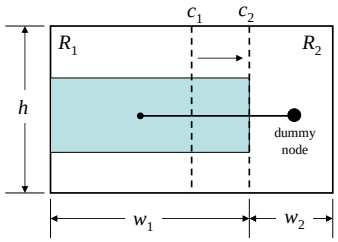

Figure 3: An example of the ratio partitioning.

Traditional min-cut based placers uniformly distribute whitespace in the chip, producing excessive wirelength when large amounts of whitespace are present. Adya et al. [2] use filler cells to control the whitespace allocation. However, this method significantly increases the number of cells and thus leads to longer running time. In contrast, we directly control the *imbalance factor* for a partitioner to allocate whitespace. Let the total areas occupied by the blocks in the two sub-regions be A_1 and A_2. The imbalance factor is defined as $\max\{A_1/A_2, A_2/A_1\} - 1$, measuring the imbalance between two partitions during recursive bisection. To identify a feasible imbalance factor for partitioning, we first set one sub-region's utilization ratio (block area / sub-region area) to 1.0 to compute the maximum imbalance ratio of the cell area for the partitioning. We then compute the maximum imbalance ratio of the other sub-region similarly. Then, the imbalance factor for the partitioning is set to the smaller imbalance ratio for the two sub-regions to avoid overfilling

the sub-region. Thus, the partitioner can find the minimum cut under the balance constraint.

2.3 Look-Ahead Bipartitioning

The look-ahead bipartitioning concept is proposed in [7] to solve floorplanning problems. It integrates the cutsize-driven bipartitioning and the area-driven floorplanning. Before each application of cutsize-driven bipartitioning, area-driven floorplanner is used to check whether the given subproblem can be legalized. If it can be legalized, then recursive cutsize-driven area bipartitioning continues in both subregions at the current level. If not, the cutsize-driven solution at that level is discarded, and the previously computed legal, "look-ahead" solution is used instead.

For our look-ahead bipartitioning, we resort to a first-fit bin-packing heuristic to check if the subproblem can be legalized. If such a legalization solution exists, the cutsize-driven bipartitioning continues like [7]. If not, we move the cut-line toward the partition with a smaller utilization ratio. Then, we redo the ratio partitioning by modifying the imbalance factor. Again, we check for the existence of a legalization solution, and the ratio partitioning is repeated until we find a legalization solution.

The partitioning stage continues until the numbers of blocks in all partitions are fewer than a threshold. Then, the global placement is obtained.

2.4 Fast Legalization

In this step, all cells are moved into rows and all overlaps are removed. To legalize a large-scale global placement solution effectively and efficiently, we propose a fast three-stage legalization algorithm: (1) place cells into their nearest rows (legalize the y-coordinates of all cells), (2) sort all standard cells according to their sizes, from the largest to the smallest, and (3) assign the x-coordinates for all cells according to the sorted order. If no overlap occurs, the resulting x-coordinate of the cell is kept; otherwise, we will find a nearest empty slot to place the cell.

After all cells are legalized, any existing detailed placement algorithm can be applied to further refine the placement solution.

3. EXPERIMENTAL RESULTS

Our placement algorithm is implemented in the C++ programming language and was compiled with gcc 3.3.2 on a Linux PC with an Intel Pentium 4 3.2GHz CPU. We tested on the two sample benchmarks provided by the ISPD'05 placement contest committee, adaptec1 and adaptec3, which have respective 211,447 and 451,650 objects. Table 1 gives the statistics of the benchmark circuits. We adopted the public detailed placer [1] to refine our placement solutions.

Table 2 shows the resulting wirelengths for each stage. Our algorithm results in the wirelength of $9.305 \times 10^7 \mu m$ ($2.731 \times 10^8 \mu m$) for adaptec1 (adaptec3). As shown in the table, the legalizer increases the HPWL by 5% on average while the detailed placer decreases the HPWL by 3% on average. Table 3 gives the runtime required for each stage. Our algorithm requires 11 min (28 min) for adaptec1 (adaptec3). The global placement stage consumes about 50% of the total runtime on average. Figure 4 show the resulting placement of adaptec3.

Table 1: The statistics of the ISPD'05 placement contest sample benchmarks.

Name	# Total Objects	# Movable Objects	# Fixed Objects	# Nets	# Pins
adaptec1	211447	210904	543	221142	944053
adaptec3	451650	450927	723	466758	1875039

Table 2: Resulting wirelengths of the benchmarks.

Benchmark	Pin-to-Pin HPWL (μm)		
	Global Placement	Legalization	Detailed Placement
adaptec1	9.123 e7	9.594 e7	9.305 e7
adaptec3	2.669 e8	2.783 e8	2.731 e8

Table 3: Runtime requirements.

Benchmark	Runtime			
	Global Placement	Legalization	Detailed Placement	Total
adaptec1	4 min	$<$ 1 min	7 min	11 min
adaptec3	15 min	$<$ 1 min	13 min	28 min

4. REFERENCES

[1] Placement Utilities, http://vlsicad.eecs.umich.edu/BK/PlaceUtils/.

[2] S. N. Adya, I. L. Markov, and P. G. Villarrubia. On whitespace and stability in mixed-size placement and physical synthesis. *Proc. of ICCAD*, pages 311–318, 2003.

[3] A. Agnihotri, M. C. YILDIZ, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: Improved recursive bisection placement. *Proc. of ICCAD*, pages 307–310, 2003.

[4] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Can recursive bisection alone produce routable standard-cell layout. *Proc. of DAC*, pages 477–482, 2000.

[5] C.-C. Chang, J. Cong, and X. Yuan. Multi-level placement for large-scale mixed-size ic designs. *Proc. of ASPDAC*, 2003.

[6] J. Cong and J. R. Shinnerl. *Multielevel Optimization in VLSI/CAD*. Kluwer Academic Publisher, 2003.

[7] J. Cong and J. Xu. Fast floorplanning by look-ahead enable recursive bipartitioning. *Proc. of ASPDAC*, 2005.

[8] H. Eisenmann and F. M. Johannes. Multielevel optimization in vlsi/cad. *Proc. of DAC*, pages 269–274, 1998.

[9] A. B. Kahng and Q. Wang. Implementation and extensibility of an analytic placer. *Proc. of ISPD*, pages 18–25, 2003.

[10] A. B. Kahng and Q. Wang. An analytic placer for mixed-size placement and timing-driven placement. *Proc. of ICCAD*, pages 565–572, 2004.

[11] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh, and P. H. Madden. Recursive bisection based mixed block placement. *Proc. of ISPD*, pages 84–89, 2004.

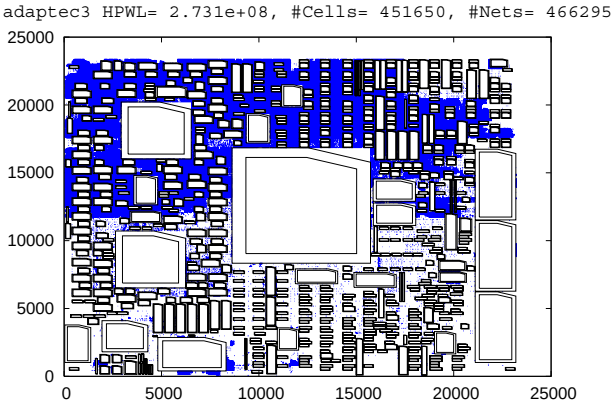

Figure 4: The resulting placement of adaptec3.