Multilevel Interconnect-Driven Floorplanner

Evangeline F.Y. Young and Joseph C.S. Lau
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

***Abstract*—As technology continues to scale down, the number of transistors on a chip has increased rapidly and interconnect delay has become a dominant factor of system performance. Scalability and routability are two major concerns in floorplanning. In this paper, we will present a multilevel floorplanner that addresses these important issues: congestion estimation, buffer planning and scalability. Experimental results show that this integrated multilevel approach, not only can handle large size problems, can also improve the routability of the solution significantly by considering the interconnect issues.**

I. Introduction

With the scaling down of the technology in IC development, the number of transistors that can be built into a standard size chip has increased rapidly. We need to handle problems of very large size in floorplanning. Scalability has become an important issue [7]. Most traditional floorplanners are unscalable since the process to obtain a good packing when everything is still flexible is non-trivial. This is even more problematic when interconnect issues are considered because interconnect optimization is a time consuming process. However interconnect optimization is a major concern and cannot be ignored.

Multilevel approach is a good solution to address the scalability problem. Multilevel approach has been used in circuit partitioning [8], [1], [12] to handle large size circuits. In multilevel floorplanning, the process is divided into two phases, clustering and packing. Modules heavily connected with one another will be placed close to each other in the clustering phase, and this can shorten the overall runtime of the floorplanner. This technique can solve the scalability problem and help to reduce the wiring congestion.

Several previous works have addressed the interconnect-driven floorplanning problem. Most of them perform grid based probabilistic analysis, and some will consider buffer insertion simultaneously [2], [6], [4], [10], [5]. Buffer insertion is one of the most popular and effective techniques [3] to achieve timing closure. A good planning of the module positions during the floorplanning stage so that buffers can be inserted wherever needed in the later routing stages will be useful. In our floorplanner, we will consider congestion with buffer insertion during the packing phase. We adopted the *variable interval buffer insertion constraint* introduced in the paper [11], i.e., buffers are constrained to be inserted for long enough wires such that the distances between adjacent buffers are lying within a range $[L, U]$ given by the users. This constraint in buffer locations provides flexibility for the later routing stage and allows users to specify their requirements accordingly. To further improve the efficiency of our floorplanner and to consider bus-based routing, we employ a net grouping technique. Grouping of nets with related topology and circuit properties into buses can improve the efficiency of the design process and allow faster convergence of the solution. At the end of the packing phase, we will apply the Lagrangian relaxation technique [14] to optimize the total area of the packing while keeping the interconnect cost unchanged. Experimental results show that our multilevel floorplanner is promising in solving the scalability and routability problem. We will define the problem in the next section. Our proposed integrated floorplanner will be introduced and described in details in section 3 to 5. Section 6 will show the experimental results.

The work described in this paper was fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region. (Project no. CUHK4231/01E)

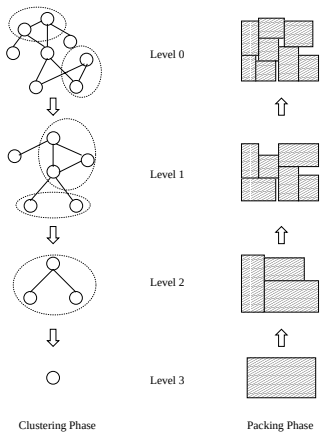

Fig. 1. A simple example to illustrate the multilevel approach in floorplanning

II. Problem Formulation

We assume that wires are routed over-the-cell and buffers can only be inserted in unoccupied spaces between the logic modules. Given a lower and upper bound $[L, U]$ for the variable interval buffer insertion constraint, a set of m nets and a set of n modules where each module M_i has an area A_i and an aspect ratio bound $[r_i, s_i]$, we want to obtain a non-overlap packing of these modules such that the area of the packing, the total wirelength and the congestion cost are small, every net satisfies its buffer insertion constraint and every module satisfies its area and aspect ratio constraint.

III. An Overview of the Floorplanner

Our multilevel floorplanner has two phases, the clustering phase and the packing phase. At each level of the clustering phase, modules that are heavily connected with each other will be clustered together to form a new module. These new modules, each actually is a collection of modules, will go through the same clustering process in the next level and this clustering process will be repeated recursively until the number of modules remain is small

0-7803-8346-X/04/$20.00

enough to be handled efficiently. Unclustering and packing will be performed in the packing phase. At each level of the packing phase, the modules in a cluster will be unclustered and packed with a basic floorplanning algorithm. These unclustering and packing steps will be repeated in the next level using the result obtained from the current level as the initial solution. These steps will be repeated recursively in each successive level until all the basic modules are obtained. An example is shown in Figure 1.

IV. Clustering Phase

In the clustering phase, modules that are heavily connected with each other will be grouped together to form new modules for the next level. The area of the new module will be the sum of the module areas inside the cluster. The netlist information will be reconstructed. The nets connecting modules in the same cluster will be removed, while those connecting modules in different clusters or connecting with an I/O pin will remain. This step will be repeated at each level until only one cluster remains. We use a mixed policy in this phase to group the modules. At each level, the hyperedge clustering method [9] will be applied first. If the percentage of modules grouped is less than a threshold r, the heavy edge clustering method [9] will be applied to increase the number of grouped modules to the required threshold. In this way, the speed at which the graph is reducing can be controlled.

A. Hyperedge Clustering

In hyperedge clustering, all the hyperedges (nets in the circuits) are first sorted in a non-increasing order of their *weights* (the number of hyperedges connecting the same set of modules). This sorted list will be scanned and the modules in a hyperedge will be grouped together if none of them has been clustered yet (an *independent hyperedge*).

B. Heavy Edge Clustering

After performing hyperedge clustering, heavy edge clustering will be applied if the percentage of modules grouped is less than a given threshold r. In heavy edge clustering, the modules will be clustered in pairs. The module pairs will be considered in a non-increasing order of their weights, i.e., the number of hyperedges connecting them. The two modules in a pair will be grouped together if both of them are not clustered yet. This grouping step will be repeated until the total number of clustered modules is increased to the required threshold.

C. Area Constraints in Clustering

It is well understood that a tighter packing can be obtained if the areas of the modules in a data set are similar. Therefore, we have imposed some constraints on the areas of the modules being clustered and on the areas of the clusters at the same level during the clustering phase in order to prepare good sets of data for the packing phase. First of all, to ensure that the areas of the clusters at each level will not vary too much, we impose the following constraint on the area of a cluster at level l:

$$\sum_{M_i \in C} A_i < \alpha l$$

where C is a cluster at level l and α is a constant. For the same reason, we impose the following constraint to discourage large modules from being clustered with small ones as packing modules of very different sizes is difficult:

$$\beta_{min} \leq \frac{A_i}{A_j} \leq \beta_{max}$$

where M_i and M_j are modules being clustered together, and β_{min} and β_{max} are given by the users.

V. Packing Phase

In the packing phase, ungrouping and packing will be performed at each level. Simulated annealing is used in our floorplanner to pack the modules. The solution obtained at one level will be used as the initial solution of the annealing process in the next level after ungrouping the modules. Experimental results show that the multilevel approach is advantageous to interconnect optimization since the multilevel hierarchy is built according to the interconnect structure. In order to improve interconnect further, our packing algorithm is interconnect-driven [13]. Buffer locations, wiring congestion, critical path length and total wirelength are considered. Net grouping is also performed to reduce the runtime and to take bus-based routing into account. More details will be given in the following sections.

A. Computations of Congestion

In the packing phase, routability and buffer locations will be considered. The estimation is done iteratively by considering each net one after another. We divide a floorplan into a 2-dimensional array of fixed-size grids. All multi-pin nets are first decomposed into a set of two-pin nets using the MST method. For each two-pin net i, we will select the best possible buffer locations that satisfy the buffer insertion constraint. This buffer insertion process can be done efficiently by dynamic programming and will be explained in more details in the next section. After computing the buffer locations of a net, we will estimate the congestion due to this net by considering all the source-buffer pair, buffer-buffer pairs and buffer-sink pair along the route, assuming that every multi-bend route of the shortest Manhattan distance is feasible. The congestion information at each grid will then be updated. This process will be repeated until all the nets are routed and analyzed.

B. Computations of Buffer Locations

To consider buffer insertions, we assume the variable interval buffer insertion constraint. To estimate the buffer locations that satisfy these constraints, dynamic programming can be used to scan the grids lying within the rectangle bounded by the source s and the sink t one by one from s to t. At each grid (x, y), we will check whether (x, y) is a feasible buffer location according to the variable interval buffer insertion constraint. If (x, y) is a feasible buffer location, we will compute the best previous buffer location if a buffer is inserted at (x, y). This process is repeated until reaching t. When t is reached, we will be able to backtrack the sequence of the best possible buffer locations from t to s.

To find the best previous buffer location, we need to define the availability of a grid for buffer insertion. This is computed based on the wiring congestion of that grid, the amount of empty space in it and the number of buffers already inserted there. For a grid at (x, y), its availability is computed as:

$$\begin{aligned} Res(x,y) \;=\; & p_1 \times \text{congestion at } (x,y) + \\ & p_2 \times \frac{\text{no. of buffers inserted at } (x,y)}{\text{max. no. of buffers allowed at } (x,y)} \end{aligned}$$

where p_1 and p_2 are parameters for adjusting the importance of the wiring congestion term and the buffer resources term.

C. Ungrouping of Modules

At each level, unclustering will be performed before applying the simulated annealing process to pack the modules. One reason that sequence pair is used as the representation in our floorplanner because unclustering can be done directly on the sequence pair efficiently.

D. Annealing Schedule

At each level of the packing phase, simulated annealing is used to pack the ungrouped modules. In the annealing process, the temperature is initialized to 10^6, the cooling rate is set to 0.9 and the following set of moves is used:

1. Rotate a module.
2. Interchange two modules in the first sequence.
3. Interchange two modules in both sequences.

In order to shorten the runtime, we can vary the number of iterations in the annealing process according to the level number. At the beginning levels, there are only a few modules, so the number of iterations needed is small. In the last few levels, the initial packing is already close to the final solution, and we can again use a fewer number of iterations. In our implementation, the number of iterations at each level is adjuated as:

$$N(l) = -\frac{l^2}{L} + l + \frac{L}{4}$$

where $N(l)$ is the number of iterations in the annealing process at level l and L is the maximum level number.

E. Cost Function

Four criteria are considered in the cost function of the annealing process. They are the area (A), total wirelength (W), critical path length (L) and congestion (C):

$$cost = \alpha A + \beta W + \gamma L + \delta C$$

where α, β, γ and δ are the weights. From level L to level one, the weights α, β, γ and δ are adjusted in such a way that the ratio of importance of A, W, L and C is 1:1:1:0. At level zero, this ratio becomes 1:1:1:1. Congestion is considered only at the last level because it is a very time consuming process.

F. Net Grouping

Grouping of nets with related topology and circuit properties into buses can improve the efficiency of the design process and allow faster convergence of the solution. In order to consider bus-based routing, we used a net grouping approach in our floorplanner. A bus is formed by bundling together the nets that start and end at the same grid. This approach will also lead to a significant reduction in the complexity of the floorplanning process since a much smaller number of net groups will be resulted. Our grouping method is divided into two levels. In the first level, we will group the nets that connect the same set of modules together. After the first stage grouping, multi-pin nets are decomposed into sets of two-pin nets by the MST approach. The second level grouping will group together all those two-pin nets with the same source and sink.

In the computation of the congestion information, we will route those grouped nets in sub-groups of size K, where K is a certain percentage of the total number of nets in that group. K is called the *net grouping factor* and is input by the users. In our implementation, K is set to $\frac{m}{100}$ where m is the total number of nets.

G. Module Sizing

After the packing phase, we will apply the Lagrangian relaxation technique to the final solution to change the dimensions of the soft modules [14] to further improve the packing quality. Experimental results show that deadspace can be reduced by 3% to 10% by this post-processing step without affecting the interconnect cost much.

Circuit	# of Modules	# of Nets	# of Two-pin Nets
ami33	33	123	304
ami49	49	408	535
playout	62	1611	2122
n2000	60	2000	2843
n2500	75	2500	3609
n3000	90	3000	4310
data100	100	1000	2108
data200	200	2000	3310
data300	300	3000	4528
data400	400	4000	5780
data500	500	5000	7225
data600	600	6000	8523
data800	800	8000	11096

TABLE I
Testing Data Sets

Circuit	Time	Deadspace	Wire-length	Blocked Wire	Critical Path
ami33	32.8%	9.6%	0.3%	-	-2.9%
ami49	-43.6%	9.1%	-1.3%	-	-0.6%
playout	-19.0%	3.1%	-2.0%	-57.1%	-7.0%
n2000	-16.1%	-1.9%	-20.3%	-1.8%	-6.2%
n2500	-9.9%	-3.9%	-2.6%	3.0%	-4.6%
n3000	-15.2%	21.9%	-1.0%	-0.8%	-15.4%
data100	-24.8%	11.8%	0.7%	-2.8%	2.6%
data200	-30.7%	16.2%	1.7%	-3.0%	1.6%
data300	-31.8%	5.6%	3.6%	-1.4%	-1.6%
data400	-38.2%	8.8%	0.8%	0.5%	-1.2%
data500	-22.6%	17.9%	3.3%	-7.5%	-1.1%
data600	-28.6%	7.9%	0.9%	-4.3%	-6.8%
data800	-38.4%	4.1%	0.4%	3.5%	1.5%
average	-22.0%	8.5%	-1.2%	-6.5%	-3.2%

TABLE III
Improvement of the Simple Multilevel Floorplanner

Circuit	Time	Deadspace	Wire-length	Blocked Wire	Critical Path
ami33	122.4%	18.8%	2.5%	-	-2.5%
ami49	72.7%	24.1%	1.9%	-100%	-1.9%
playout	52.9%	12.2%	3.3%	-83.3%	-8.2%
n2000	173.2%	12.3%	55.5%	-28.7%	-35.2%
n2500	153.9%	8.3%	8.1%	-32.2%	-27.0%
n3000	161.5%	22.6%	9.1%	-19.9%	-27.5%
data100	202.6%	-13.4%	21.0%	-29.0%	-9.4%
data200	221.1%	0.9%	8.7%	-27.0%	-13.8%
data300	301.4%	12.0%	6.1%	-51.2%	-7.1%
data400	314.2%	64.5%	6.2%	-64.2%	-17.9%
data500	293.4%	11.8%	2.3%	-60.3%	-9.4%
data600	196.7%	-5.3%	0.2%	-12.6%	-15.5%
data800	205.9%	2.7%	2.8%	-19.6%	-13.6%
average	190.2%	13.2%	9.8%	-44.0%	-14.5%

TABLE V
Improvement of the Multilevel Floorplanner with Interconnect Optimization

VI. Experimental Results

We implemented our multilevel floorplanner on a Pentium III 1GHz machine with 1GB memory. We tested our floorplanner with three MCNC building block benchmarks, ami33, ami49 and playout, and some densely connected randomly generated data sets. Table I shows the information of all the data sets. A simple global router is used to evaluate the performance of the floorplan solution. In the global router, multi-pin nets are first decomposed into two-pin nets based on the MST method and the two-pin nets are routed one after another. If a net can be routed from its source to its sink in the shortest Manhattan distance with all the required buffers inserted successfully and without exceeding the wiring capacity, i.e., the

Circuit	Traditional Floorplanner						Simple Multilevel Floorplanner						Wire Capacity
	Time (s)	Deadspace (%)	Wirelength ($10^6 \mu m$)	Congestion	Blocked Wire	Critical Path (nm)	Time (s)	Deadspace (%)	Wirelength ($10^6 \mu m$)	Congestion	Blocked Wire	Critical Path (nm)	
ami33	22.34	7.6	0.0608	4.92	0	1703	29.67	8.33	0.061	5.47	0	1654	22
ami49	62.6	6.57	1.0154	5.98	0	7632	35.33	7.17	1.002	5.43	1	7590	24
playout	319	8.275	0.8697	70.2	14	1526	258.3	8.53	0.852	68.7	6	1419	74
n2000	89	8.9	0.0958	88	110	193	74.67	8.73	0.0764	88	108	181	88
n2500	142	9.5	0.1598	88	268	191	128	9.13	0.1557	88	276	182.3	88
n3000	239	8.1	0.2005	103	374	251	202.7	9.87	0.1985	103	371	212.3	103
data100	471	8.02	1.0584	30	142	2095	354.3	8.97	1.066	30	138	2149	30
data200	1120	6.8	4.0984	35	168	3085	775.7	7.9	4.1679	35	163	3135	35
data300	1637	7.1	8.2549	42	291	7519	1117	7.5	8.5487	42	287	7402	42
data400	2513	5.7	10.5913	47	409	9826	1553	6.2	10.6724	47	411	9711	47
data500	6217	4.18	16.2832	53	597	11268	4809	4.93	16.8135	53	552	11145	53
data600	12213	6.3	20.0385	62	141	14295	8720	6.8	20.2098	62	135	13319	62
data800	20168	7.4	25.1132	70	227	16187	12431	7.7	25.2083	70	235	16432	70

TABLE II Comparison between the Simple Multilevel Floorplanner and the Traditional Floorplanner

Circuit	Simple Multilevel Floorplanner						Multilevel Floorplan with Interconnect Optimization						Wire Capacity
	Time (s)	Deadspace (%)	Wirelength ($10^6 \mu m$)	Congestion	Blocked Wire	Critical Path (nm)	Time (s)	Deadspace (%)	Wirelength ($10^6 \mu m$)	Congestion	Blocked Wire	Critical Path (nm)	
ami33	29.67	8.33	0.061	5.47	0	1654	66	9.9	0.0625	4.76	0	1613	22
ami49	35.33	7.17	1.002	5.43	1	7590	61	8.9	1.0215	6.07	0	7445	24
playout	258.3	8.53	0.852	68.7	6	1419	395	9.57	0.8802	67.3	1	1303	74
n2000	74.67	8.73	0.0764	88	108	181	204	9.8	0.1188	88	77	117.3	88
n2500	128	9.13	0.1557	88	276	182.3	325	9.89	0.1683	88	187	133	88
n3000	202.7	9.87	0.1985	103	371	212.3	530	12.1	0.2165	103	297	154	103
data100	354.3	8.97	1.066	30	138	2149	1072	7.77	1.29	30	98	1946	30
data200	775.7	7.9	4.1679	35	163	3135	2491	7.97	4.5308	35	119	2701	35
data300	1117	7.5	8.5487	42	287	7402	4484	8.4	9.0659	42	140	6875	42
data400	1553	6.2	10.6724	47	411	9711	6433	10.2	11.3296	47	147	7976	47
data500	4809	4.93	16.8135	53	552	11145	18920	5.51	17.2015	53	219	10096	53
data600	8720	6.8	20.2098	62	135	13319	25874	6.44	20.2512	62	118	11248	62
data800	12431	7.7	25.2083	70	235	16432	38023	7.91	25.9084	70	189	14194	70

TABLE IV Comparison between the Multilevel Floorplanners with and without Interconnect Optimization

maximum number of wires allowed in each grid, the net is said to be routable; otherwise, it is called a *blocked net*. We will compare the performance based on the deadspace percentage, total wirelength, congestion (average number of wires passing through the top 10% most congested grids), number of blocked two-pin wires and critical path length.

Table II compares a traditional floorplanner and a simple multilevel floorplanner *without* any routability control. The traditional floorplanner is a simulated annealing based floorplanner using the sequence pair representation and a cost function with total area, wirelength and critical path length weighted in the ratio of 1:1:1. Table III shows the percentage improvement of the simple multilevel floorplanning over the traditional one. We can see that with a small penalty in area (increase by 8.5% on average), the runtime and the number of blocked wires can be reduced significantly by 22% and 6.5% on average respectively. The total wirelength and critical path length are also slightly improved by 1.2% and 3.2% respectively. These results verify the idea that the multilevel approach is useful for large size problems and can benefit the routability of the solution.

Table IV and Table V compares the simple multilevel floorplanner with the multilevel floorplanner with interconnect optimization. We can see that interconnect optimization is an expensive process, but is still affordable to be applied in large size problems by using the multilevel approach. By performing interconnect optimization in the multilevel floorplanner, we can further reduce the number of blocked wires and the critical path length by 44.0% and 14.5% respectively. However the total wirelength is increased by 9.8% since some detours must occur to reduce congestion.

References

[1] C. J. Alpert, J.-H. Huang, and A. B. Kahng. Multilevel Circuit Partitioning. *Proceedings of the 34th ACM/IEEE Design Automation Conference*, 1997.

[2] H. M. Chen, H. Zhou, F. Y. Young, and D. F. Wong. Integrated Floorplanning and Interconnect Planning. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 354–357, 1999.

[3] J. Cong. Challenges and Opportunities for Design Innovation in Nanometer Technologies. *SRC Working Papers*, 1997.

[4] J. Cong, T. Kong, and D. Z. Pan. Buffer Block Planning for Interconnect-Driven Floorplanning. *Proceedings IEEE International Conference on Computer-Aided Design*, pages 358–363, 1999.

[5] F. F. Dragan, A. B. Kahng, S. Muddu, and A. Zelikovsky. Provably Good Global Buffering Using an Available Buffer Block Plan. *Proceedings of the International Conference on Computer-Aided Design*, pages 104–109, 2000.

[6] S. Krishnamoorthy J. Lou and H. S. Sheng. Estimating Routing Congestion using Probabilistic Analysis. *Proceedings of International Symposium on Physical Design*, pages 112–117, 2001.

[7] A. B. Kahng. Classical Floorplanning Harmful? *International Symposium on Physical Design*, pages 207–212, 2000.

[8] G. Karypis and V. Kumar. Multilevel Graph Partitioning Schemes. *Proceedings of the 1995 International Conference on Parallel Processing*, 3:113–122, 1995.

[9] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel Hypergraph Partitioning: Applications in VLSI Domain. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, 7(1), 1999. Also appeared in DAC 1997.

[10] P. Sarkar, V. Sundararaman, and C. K. Koh. Routability-Driven Repeater Block Planning for Interconnect-Centric Floorplanning. *International Symposium on Physical Design*, pages 186–191, 2000.

[11] C. W. Sham and Evangeline F.Y. Young. Routability Driven Floorplanner with Buffer Block Planning. *Proceedings of International Symposium on Physical Design*, pages 50–55, 2002.

[12] S. Wichlund and E. J. Aas. On Multilevel Circuit Partitioning. *Proceedings of the 1995 International Conference on Parallel Processing*, pages 505–511, 1998.

[13] Keith K.C. Wong and Evangeline F.Y. Young. Fast Buffer Planning and Congestion Optimization in Interconnect-driven Floorplanning . *Proceedings of 8th Asia and South Pacific Design Automation Conference*, pages 411–416, 2003.

[14] F. Y. Young, Chris C.N. Chu, W. S. Luk, and Y. C. Wong. Floorplan Area Minimization using Lagrangian Relaxation. *International Symposium on Physical Design*, pages 174–179, 2000.