Journal of Southeast University (English Edition) Vol. 24, No. 4, pp. 462 – 467 Dec. 2008 ISSN 1003—7985

Recursive bisection placement algorithm with the predicted wirelength

Hao Jie Ma Hong Peng Silong

(National ASIC Design Engineering Center, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: To obtain a better placement result, a partitioning-based placement algorithm with wirelength prediction called HJ-Pl is presented. A new method is proposed to estimate proximity of interconnects in a netlist, which is capable of predicting not only short interconnects but long interconnects accurately. The predicted wirelength is embedded into the partitioning tool of bisection-based global placement, which can guide our placement towards a solution with shorter interconnects. In addition, the timing objective can be handled within the algorithm by minimizing the critical path delay. Experimental results show that, compared to Capo10.5, mPL6, and NTUplace, HJ-Pl outperforms these placers in terms of wirelength and run time. The improvements in terms of average wirelength over Capo10.5, mPL6 and NPUplace are 13%, 3%, and 9% with only 19%, 91%, and 99% of their runtime, respectively. By integrating the predicted wirelength-driven clustering into Capo10.5, the placer is able to reduce average wirelength by 3%. The timing-driven HJ-Pl can reduce the critical path delay by 23%.

Key words: hierarchy; interconnect; placement; VLSI circuit; wirelength prediction

Placement is an important step in the overall IC design process in deep submicron technologies. A lot of algorithms[1–2] have been proposed in the past 30 years, including partitioning-based methods[3–5], analytical methods[6], iterative methods, etc. Partitioning-based placement algorithms determine cell locations by recursively dividing an initial region with successive bisections[3–4] or quadrisections[5]. Advances in partitioning research have provided a number of fast algorithms which produce extremely good results.

The placement problem can be driven by different objectives, such as timing, routability, thermal distribution, or a combination of them. The classical placement objective function is the total wirelength, which correlates well with global routing resource demand. Interconnection prediction is very important for early feasibility studies in design flow. During the past two decades, different wirelength prediction approaches have been proposed because of the need for early interconnect optimization in the design flow to achieve timing closure. Rent's rule[7] has been a basic tool to estimate the wirelength[7–8]. In Refs. [9 – 11], wirelengths are estimated based on circuit characteristics. Most of the estimation techniques provide estimates of just the average wirelength. We develop an individual wirelength prediction based on circuit characteristics to estimate the wirelength of a layout design in advance before placement. In fact, the final wirelengths depend on placement algorithms. Wirelength predictions that are accurate for one placement algorithm may be inaccurate for another. In our placement tool, the predicted wirelength as a main optimal objective is embedded into placement flow. Experimental results show that the wirelengths of final placement can trend to predicted wirlengths.

In this paper, we propose a novel partitioning based placement algorithm for standard cells. The major contributions of this paper can be summarized as follows:

1) We find that the basic circuit characteristics (e.g. net degree and net area) and node level can be used to predict wirelengths in the final layout. These prelayout measures demonstrate good correlations with postlayout interconnect lengths. We propose to couple the wirelength predictions with our placement flows.

2) In conventional partitioning-based placements, partitioning tools are typically done with the min-cut objective. In order to reduce the lengths of intra-cluster interconnects, we introduce a new objective function that incorporates a predicted wirelength component for partitioning tools. Experiments show that we can obtain a better wirelength result with little loss in time.

1 Overall of Placement Tool

Fig. 1 shows that our placement framework consists of four stages. We predict individual wirelengths for all nets based on circuit characteristics in stage 1. In stage 2, we partition both the chip region and the circuit netlists recursively by horizontal and vertical cut lines. In the partitioning phase, wirelength-driven clustering is performed to reduce the wirelengths of intra-cluster interconnects. After clustering, for better min-cut, wirelength-driven refinement is executed without loss of wirelength quality. The subcircuits af-

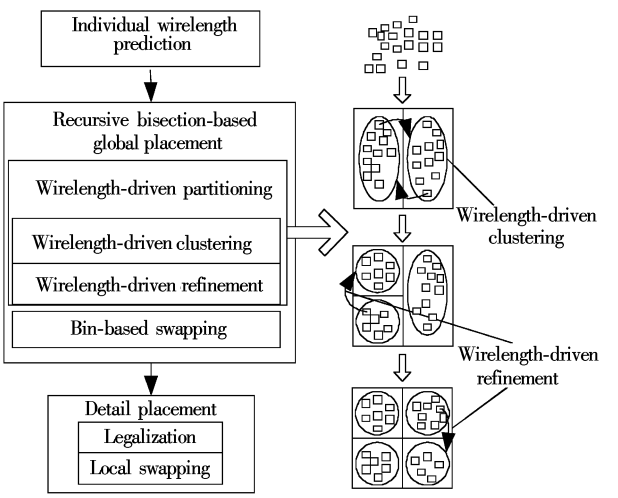

Fig. 1 Framework of our placement tool

Received 2008-05-12.

Biographies: Hao Jie (1981—), male, graduate; Peng Silong (corresponding author), male, doctor, professor, silong.peng@ia.ac.cn.

Foundation item: The National Key Project of Scientific and Technical Supporting Programs (No. 2006BAK07B04).

Citation: Hao Jie, Ma Hong, Peng Silong. Recursive bisection placement algorithm with the predicted wirelength[J]. Journal of Southeast University (English Edition), 2008, 24(4): 462 – 467.

ter partitioning are assigned to rectangular bins. A bin-based simulated annealing is performed to improve the current placement. The final step simply spreads overlapped cells and makes local improvements to obtain the detailed placement.

2 Wirelength Prediction

In this section, we explain our wirelength prediction technique in detail. A circuit netlist can be modeled by a hypergraph $H(V, E)$ with a node(cell) set $V = \{v_i \mid i = 1, 2, \ldots, n\}$ and a hyperedge (net) set $E = \{e_j \mid j = 1, 2, \ldots, m\}$. Each net e_j is a subset of V with cardinality $|e_j| \geqslant 2$. An edge $(s, t) \in e_j$ is an output of source node s and an input of sink node t, which is a subset of the hyperedge. The degree of the node v_i, denoted by $d(v_j)$, is the number of nets incident to it. The degree of net e_j, denoted by $d(e_j)$, is the number of nodes incident to it. Each node is associated with an area cost, area (v_i). Net area is the total area of nodes belonging to that net, namely, area $(e_j) = \sum_{v_i \in e_j} \text{area}(v_i)$.

Individual wirelength is dependent on three factors in this paper: 1) The net degree and the net area, which are the bases of the wirelength; 2) The shape distribution of a node level; 3) The range of a node level. Wirelength prediction is performed in three phases in which we calculate three different weights for all the nets according to the above three factors. The final predicted wirelength is obtained by combining these weights.

2.1 Basic wirelength

Obviously, the net with a larger net degree and net area tends to have a larger fan-out range. It is intuited that larger fan-out nets usually correspond to longer net lengths. We use net degree and net area of net as its basic wirelength. The area factor of net e is computed by

$$A(e) = \sum_{v_i \in e} \frac{\text{area}(v_i)}{d(e)} \tag{1}$$

The basic wirelength of net e is the combination of the area factor and the number of nodes belonging to the net and is given as

$$L_{\text{basic}}(e) = 1 + \frac{A(e)}{A_{\max}} + \frac{d(e)}{d_{\max}} \tag{2}$$

where $A_{\max}$ is the largest node area amongst all nodes; $d_{\max}$ is the largest net degree over all nets.

2.2 Shape distribution of node level

Define level (v), the level of node v, as the maximum topological depth over all directed paths beginning at a PI (primary input) and terminating at node v. num_level(v) is the number of nodes at level(v). Fig. 2 shows the shape of the rd73 circuit in the MCNC (Microelectronics Center of North Carolina) benchmark. In Fig. 2, the number of nodes at level 1 is maximal and the placement tool needs more sites for the nodes at level 1. So the wirelength of nets with nodes at level 1 may dilate during the placement phase. The nets connecting the nodes with larger num_levels will have larger fan-out ranges in the final layout. The factor DNL(e) used to measure the dilatability due to the shape distribution of node levels for net e can be calculated as

$$\text{DNL}(e) = \sum_{v_i \in e} \frac{\text{num\_level}(v_i)}{d(e)} \tag{3}$$

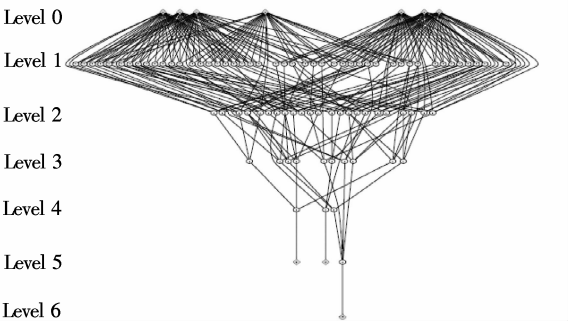

Fig. 2 Shape distribution of rd73

For example, referring to Fig. 3, num_level(x_0) = num_level(u) = 1, num_level(x_1) = num_level(x_2) = num_level(x_3) = num_level(x_4) = 5, num_level(x_6) = num_level(x_7) = 2.

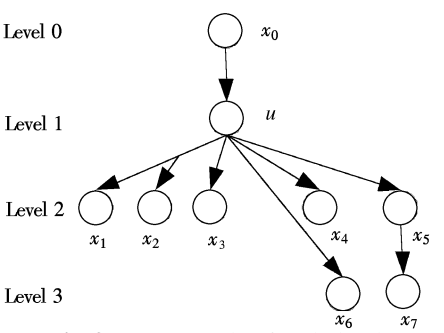

Fig. 3 An example of node level

Then, we can obtain DNL$(u, x_0) = 1$, DNL$(u, x_1, x_2) = 3.67$, DNL$(u, x_3) = \ldots = \text{DNL}(u, x_5) = 3$, DNL$(u, x_6) = 1.5$, DNL$(u, x_7) = 3.5$.

Since DNL(u, x_1, x_2) is larger than others, it indicates that net (u, x_1, x_2) has a stronger dilatability than others.

2.3 Range of node level

If node levels are within a limited range, the wirelength of the output net will likely to be less than that of a net with widely distributed node levels. To capture the ranges of the node levels of net e, we define the factor RNL(e) as

$$\text{RNL}(e) = \sum_{\substack{v_i, v_j \in e \\ i \neq j}} \frac{\text{level}(v_i) - \text{level}(v_j)}{d(e)} \tag{4}$$

If the nodes connecting the net have the same node level, the RNL factor of the net is 0. In Fig. 3, from our definitions, RNL$(u, x_3) = 1$, RNL$(u, x_6) = 2$. That is, by the metric, net (u, x_6) has a longer wirelength than net (u, x_3). Note that sometimes the RNL and DNL are approximately "orthogonal". When the RNL factor of a net is 0, the DNL factor of the net may be very large.

The predicted result of the apex2 circuit is shown in Fig. 4. In these figures, on the x-axis are net id and on the y-axis are net lengths. The solid line represents predicted wire-

lengths, and the points represent the actual wirelengths (in percentages) in the final placement using our HJ-Pl. The shape distribution of node level, DNL, and the range of node level, RNL, in estimation of circuits are presented in Figs. 4 (b) and (c). As shown in Fig. 4, wirelengths of long interconnects increase rapidly with an increase in RNL. DNL is more efficient for short interconnects.

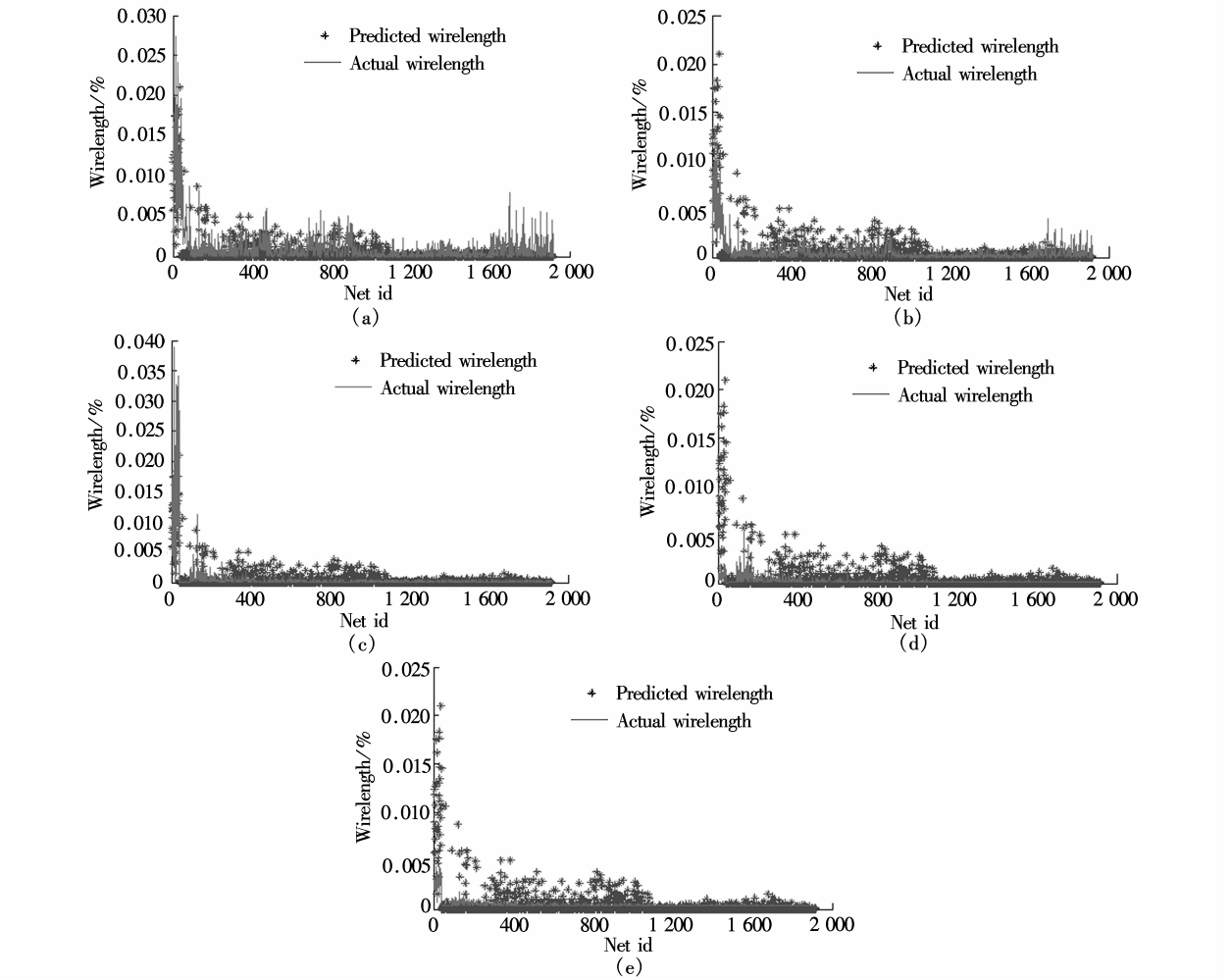

Fig. 4 Prediction results of apex2. (a) L_{pre} vs. actual wirelength; (b) DNL vs. actual wirelength; (c) RNL vs. actual wirelength; (d) Connectivity vs. actual wirelength; (e) Edge separability vs. actual wirelength

2.4 Individual wirelength

The wirelength of net e, L_{pre}, is its basic wirelength $L_{basic}(e)$ adjusted by DNL(e) and RNL(e):

$$l_{pre}(e) = a \times L_{basic}(e) \times \mathrm{DNL}(e) + b \times L_{basic}(e) \times \mathrm{RNL}(e) \tag{5}$$

$$L_{pre}(e) = \frac{l_{pre}(e)}{l_{max}} \tag{6}$$

where a, b are parameters that control the trade-off between DNL and RNL; l_{max} is the largest $l_{pre}(e)$ amongst all nets.

Most proposed wirelength predictions have poor results for long interconnects. Figs. 4 (d) and (e) are the predicted results using connectivity and edge separability[10-11], which are not sensitive for long connections. Fig. 4(a) shows that our approach has good predictions for long interconnects.

3 Recursive Bisection-Based Global Placement

Recursive bisection based placement algorithms seek to decompose a given placement instance into smaller instances by subdividing the placement region, assigning cells to subregions, reformulating constraints and cutting the netlist (see Fig. 1). The top-down placement process can be viewed as a sequence of passes where each pass examines all blocks and, if required, divides them into two smaller blocks using min-cut partitioning. Such netlist decomposition is typically done with the min-cut objective. A novel clustering-refinement multilevel partitioning algorithm by incorporating a min-wirelength objective is proposed in this paper. The global placement algorithm is as follows:

GLOBAL_PLACEMENT ($H(V, E)$, Layout)//Q: the queue of placement bins

While bin size is big enough

do $Q \rightarrow a$ bin;

Choose a (horizontal or vertical) cut-line for the bin;

Partitioning attempts to split each bin roughly in half;

Build partitioning hypergraph from netlist and cells contained in the bin;

Partition the bin into smaller bins using WL_PAR-

TITIONING($H_i(V_i, E_i)$);
//$H_i(V_i, E_i)$ is a sub-hypergraph of $H(V, E)$
Q←each child bin;
for all finest bins//bin-basedo swapping
do swap two bins;
if HPWL increases
then undo swapping;

WL_PARTITIONING($H(V, E)$)
for $e \in E$//clustering phase
do if the area of partition ⩽ area_th && the number of all cells ⩽ num_th
//area_th is the area threshold; num_th is the minimal specified number of cells after clustering
then Cluster the cells of net e in a nonincreasing net weight order;
Initial random bisection;
for $v \in$ partition bounder//refinement phase
do Compute $g_w = \lambda L_{pre} + (1 - \lambda) g_o$;
Choose cell move according to FM-scheme;

3.1 Wirelength-driven clustering

The nets are initially sorted in a nonincreasing net weight order which is computed as the predicted wirelength, L_{pre}. The nets of the same weight are sorted in a nondecreasing net area order. Then, the nets are visited in that order and for each net that connects cells that have not yet been grouped, the cells are grouped together. Thus, this scheme gives preference to the nets that have large weight. After all of the nets have been visited, the groups of cells that have been matched are contracted together to form the next level coarser netlist. The cells that are not parts of any contracted nets are simply copied to the next level coarser netlist.

3.2 Timing-driven clustering

In addition, timing objectives can be handled within our algorithm by minimizing the critical path delay. The critical path delay can be determined by computing the arrival time, iteratively, making use of the following equation:

$$\mathrm{ARR}(t) = \begin{cases} 0 & t \in \mathrm{PI} \\ \max\limits_{(s,t)\in E} \{\mathrm{ARR}(s) + d(s,t)\} & \text{otherwise} \end{cases} \quad (7)$$

where $d(s, t)$ is the delay of edge (s, t). The critical path delay is

$$T = \max_{t \in \mathrm{PO}} \mathrm{ARR}(t) \quad (8)$$

Similarly we can also compute the required arrival time for each node using the following equation:

$$\mathrm{REQ}(s) = \begin{cases} T & s \in \mathrm{PO} \\ \min\limits_{(s,t)\in E} \{\mathrm{ARR}(t) - d(s,t)\} & \text{otherwise} \end{cases} \quad (9)$$

We can now compute the slack value for each edge, which measures how much additional delay can be added to an edge without increasing the critical path delay of the whole circuit. The slack of a given edge (s, t) can be computed as

$$\mathrm{slack}(s,t) = \mathrm{REQ}(t) - \mathrm{ARR}(s) - d(s,t) \quad (10)$$

where $d(s, t) = r(s, t)(c(s, t)/2 + c(T_v))$, $c(s, t) = (c_a w(s, t) + c_f) l(s, t)$, $r(s, t) = r_o l(s, t)/w(s, t)$, $l(s, t) = L_{pre}(e)/d(e)$. $w(s, t)$, c_a, c_f and r_o are wire width, area capacitance, fringing capacitance and resistance for unit-width wire, respectively; T_v is the subtree rooted at v; $c(T_v)$ is the capacitance of a dc-connected subtree in T_v rooted at T_v's root. Finally, according to Eq. (10), we can obtain the net weight with timing objective:

$$\mathrm{weight}(e) = 1 - \sum_{(s,t)\in e} \frac{\mathrm{slack}(s,t)}{s_{\max}} \quad (11)$$

where s_{max} is the largest slack sum over all nets. For optimizing timing objectives, as the net weight, weight(e) instead of L_{pre} is embedded into the clustering phase. The net with small slack will be protected by the timing-driven clustering, which can effectively minimize critical path delay during placement.

3.3 Wirelength-driven refinement

In the refinement phase, the FM algorithm[12] is used to reduce the cutsize. We change the original gain of FM by introducing a predicted wirelength objective to reduce the degeneration of the final wirelength. The wirelength-aware gain of FM swapping is given as

$$g_w = \lambda L_{pre} + (1 - \lambda) g_o \quad (12)$$

where λ is a parameter that controls the trade-off between wirelength and the original gain of FM, which is less than 1 and larger than 0.

3.4 Bin-based swapping

After clustering for minimizing the wirelengths of intra-partition interconnects, bin-based simulated annealing is conducted to find a good location for each partition to be placed in, thus, minimizing the total wirelength between bins. There are three types of moves in bin-based simulated annealing: horizontal switch, vertical switch, and diagonal switch. These moves switch two adjacent bins.

4 Detail Placement

In the detailed placement step, overlaps between cells have to be resolved to obtain a legal placement. When placing cells to remove overlaps, we have to consider two conflicting factors: the total wirelength and the legality of result. We use a greed heuristic to obtain a legal placement, while the local swapping tries to reduce the wirelength of the placement.

After global placement, we divide each row in the placement region into placeable segments based on the overlap of the blockages with the row. We now use a greedy heuristic to bring every row in the placement region to within its capacity by moving the standard cells. Once the rows have been brought under capacity, we move the cells among the placeable segments to satisfy their respective capacities. The cells are then assigned to legal positions within each segment.

After all overlaps are removed, greedy local improvement

is performed. We try to switch adjacent standard cells to see if total wirelength can be improved. This step will cure the wirelength loss caused by the blind cell spreading step of legalization. For each row, all the bins from left to right are traversed and an attempt is made to place in the bin onto the site array. If there are blocked sites, we ignore it and go to the right side of the chunk of blocked sites.

5 Experiment

Our placement algorithm is implemented in C programming language and compiled with gcc 4.0 on a Linux PC with an Intel Pentium Ⅳ1.5 GHz CPU and 768 MB memory. These algorithms are tested using a set of benchmarks published in ISPD2002, ISPD2005, ISCAS89 and PEKO. In the first experiment, we compare HJ-Pl with three well-known academic placement tools, namely Capo10.5[3], mPL6[4] and NTUplace[13]. Moreover, the wirelength-driven clustering algorithm is an individual soft package which can embed other open source tools. For the second experiment, we integrate the wirelength clustering with Capo10.5 due to its code availability. Our wirelength prediction tool estimates only HPWL in all the experiments. For the last experiment, we compare the timing-driven HJ-Pl with the original HJ-Pl.

5.1 Comparison with other placers

For all experiments, we give an average of 10 runs and use the ISPD2002 benchmarks with 20% total white-space and random pad locations. The statistical information of benchmarks is listed in Tab. 1. Tab. 1 shows that HJ-Pl obtains a HPWL improvement of 13% versus Capo10.5 and is 5.24 times faster than Capo10.5. The runtime of Capo10.5 for the ibm16 circuit is very long. HJ-Pl (WL) reduces runtime by 10% and 9% over mPL6 and NTUplace with 3% and 1% shorter wirelengths, respectively.

Tab. 1 The resulting HPWL and runtime for different placers

Circuit	#cell	HJ-Pl		Capo10.5		mPL6		NTUplace	
		HPWL/(10^6 μm)	Time/s	HPWL/(10^6 μm)	Time/s	HPWL/(10^6 μm)	Time/s	HPWL/(10^6 μm)	Time/s
ibm01	12 752	2.02	257.17	2.54	1 035.84	2.12	615.38	2.23	97.4
ibm02	19 601	4.24	315.32	5.63	1 585.57	4.63	765.09	4.74	165.1
ibm07	45 926	9.35	1131.41	11.13	1 656.26	9.68	1 707.49	10.24	993.8
ibm08	51 309	11.19	2 483.09	13.55	5 006.82	11.22	2 550.87	11.99	798.6
ibm09	53 395	12.19	2 153.11	14.33	1 983.73	12.08	2 777.82	12.30	959.8
ibm10	69 429	28.85	3 031.21			29.28	3 701.04	28.85	1 350.3
ibm11	70 558	17.80	3 069.43	20.02	6 835.09	17.43	3 569.79	17.79	1 644.9
ibm12	71 076	31.97	2 324.90	38.91	4 347.09	32.77	2 119.54	32.86	1 231.2
ibm16	183 484	51.49	5 485.23	61.70	98 889.93	53.45	5 646.05	55.49	6 286.4
ibm18	210 613	40.17	10 174.31	44.78	22 158.01	41.98	10 152.01	51.17	17 264.3
Ratio		1	1	1.13	5.24	1.03	1.10	1.09	1.01

5.2 Integration with Capo

We integrate our wirelength-driven clustering with Capo10.5, which is based on min-cut placement techniques. In addition, the wirelength-driven clustering using Connectivity[10-11] is embedded into Capo10.5 as a comparison with our method. Tab. 2 shows the results without and with wirelength-driven clustering based on the PEKO benchmark with 10% total white-space, uniform cell sizes, and random pad locations, and the ISPD2005 benchmarks. From Tab. 2, the original Capo10.5 averages 3% more wirelength than Capo10.5 with wirelength-driven clustering, and runs faster. The experimental results also reveal the fact that our wirelength prediction is superior to Connectivity for placement quality.

Tab. 2 HPWL and runtime comparison of original Capo and Capo with wirelength-driven clustering

Circuit	Capo10.5(WL)		Capo10.5(connectivity)		Capo10.5	
	HPWL/(10^6 μm)	Time/s	HPWL/(10^6 μm)	Time/s	HPWL/(10^6 μm)	Time/s
peko01	1.43	671.42	1.56	674.78	1.57	514.38
peko05	4.11	1 423.34	4.97	1 400.17	4.09	1 138.97
peko10	14.11	3 978.12	15.31	4 309.12	14.52	3 823.33
peko15	45.92	39 871.44	44.88	40 671.02	46.81	38 625.60
peko18	38.73	12 150.71	40.01	11 098.15	39.64	11 427.60
adaptec1	87.43	26 587.89	88.01	27 564.77	88.72	25 065.40
adaptec2	96.89	33 158.54	98.90	32 453.98	98.25	32 530.90
bigblue1	101.14	40 138.23	104.65	41 453.01	106.85	39 549.60
Ratio	1	1	1.02	1.01	1.03	0.96

5.3 Timing-driven HJ-Pl

The results without and with timing-driven clustering are shown in Tab. 3. After placement (using the ISCAS89 benchmark), we perform global and detailed routing, RC extraction and timing analysis using commercial tools. The electrical parameters have been chosen to resemble a typical 90 nm process. Tab. 3 shows that HJ-Pl(T) with timing-driven clustering reduces the critical path delay by about 23%.

Tab. 3 Critical path delay comparison of original HJ-Pl and HJ-Pl with timing-driven clustering

Circuit	Critical path delay/ns	
	HJ-Pl(T)	HJ-Pl
C6288	14.21	18.34
C7552	18.77	23.90
S9234	27.96	32.89
S13207	26.33	37.41
S15850	35.47	39.19
Ratio	1	1.23

6 Conclusion

A new method for standard cell placement is presented in this paper. The proposed method is based on a new min-cut partitioning with wirelength-driven clustering and refinement. Experiments show that we can obtain shorter HPWL than Capo, mPL and NTUplace. Our tool is also capable of doing power-driven placement in the future.

References

[1] Adya S N, Markov I L. Combinatorial techniques for mixed-size placement[J]. *ACM Transactions on Design Automation of Electronic Systems*, 2005, **10**(1): 58 − 90.

[2] Doll K, Johannes F M, Antreich K J. Iterative placement improvement by network flow methods [J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1994, **13**(10): 1189 − 1200.

[3] Roy J A, Papa D A, Ng A N, et al. Satisfying whitespace requirements in top-down placement[C]//*Proc of International Symposium on Physical Design*. San Jose, California, USA: IEEE Press, 2006: 206 − 208.

[4] Wang M, Yang X, Sarrafzadeh M. Dragon 2000: fast standard-cell placement for large circuits[C]//*Proc of the International Conference on Computer-Aided Design*. San Jose, California, USA, 2000: 260 − 263.

[5] Vygen J. Algorithms for large-scale flat placement[C]//*Proc of Design Automation Conference*. Anaheim, California, USA, 1997: 746 − 751.

[6] Chan T, Cong J, Joseph R, et al. mPL6: enhanced multilevel mixed-size placement[C]//*Proc of International Symposium on Physical Design*. San Jose, California, USA, 2006: 212 − 214.

[7] Donath W E. Placement and average interconnection lengths of computer logic[J]. *IEEE Transactions on Circuits and Systems*, 1979, **26**(4): 272 − 277.

[8] Caldwell A E, Kahng A B, Mantik S, et al. On wire length estimations for row-based placement[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1999, **18**(9): 1265 − 1278.

[9] Balachandran S, Bhatia D. A priori wirelength interconnect estimation based on circuit characteristics[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005, **24**(7): 1054 − 1065.

[10] Hu B, Malgorzata Marck-Sadowaka. Wire length prediction based clustering and its application in placement[C]//*Proc of Design Automation Conference*. Anaheim, California, USA, 2003: 800 − 805.

[11] Liu Q, Malgorzata Marck-Sadowaka. Semi-individual wirelength prediction with application to logic synthesis[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2006, **25**(4): 611 − 624.

[12] Fiduccia C M, Mattheyses R M. A linear-time heuristic for improving network partitions[C]//*Proc of Design Automation Conference*. Las Vegas, Nevada, USA, 1982: 175 − 181.

[13] Chen Tung-Chieh, Hsu Tien-Chang, Jiang Zhe-Wei, et al. NTUplace: a ratio-partitioning-based placement algorithm for large-scale mixed-size designs[C]//*Proc of International Symposium on Physical Design*. San Francisco, California, USA, 2005: 236 − 238.

预测线长驱动的二分布局算法

嵩 杰 马 鸿 彭思龙

(中国科学院自动化研究所国家专用集成电路设计工程研究中心,北京 100190)

摘要: 为了有效提高布局质量,提出一种基于预测线长的二分布局算法 HJ-Pl. 该算法对长、短互连线都有很好的预测效果. 通过将预测线长嵌入到布局框架中,使算法可以预先控制布局后可能产生的长互连线,有效降低它们在布局过程中被分割的几率,从而达到减小总线长的目的. 另外,该算法通过最小化关键通路时延,得到了较好的时序优化效果. 实验表明,与现有的 Capo10.5, NTUplace 和 mPL6 算法相比,该布局算法可分别减小线长 13%,3%,9%. 将预测线长目标集成到 Capo10.5 中可减小线长 3%. 带有时序驱动功能的 HJ-Pl 可以减小关键通路时延 23% 左右.

关键词: 层次化;互连线;布局;超大规模集成电路;线长预测

中图分类号: TP391.72