IEICE Electronics Express, Vol.9, No.16, 1310–1315

TimFastPlace: Critical-path based timing driven FastPlace

Jiliang Zhang[1,2a)], **Yongqiang Lu**[1b)], **Qiang Zhou**[1], **Qiang Wu**[2], **Yaping Lin**[2], **and Kang Zhao**[1]

[1] *Department of Computer Science and Technology, Tsinghua University*
Beijing, China

[2] *College of Information Science and Engineering, Hunan University*
Changsha, China

a) *hnu.zjl@gmail.com*
b) *luyq@tsinghua.edu.cn*

Abstract: In this paper, we propose a critical-path based timing driven FastPlace, named TimFastPlace, which uses an iterative critical path-based weighting model to optimize the critical path delay at the equation solving stage. Experimental results on several industry cases and ISCAS89 cases show that we are able to obtain up to 30.83% Worst Negative Slack (WNS), an average of 23.42% WNS and 18.87% Total Negative Slack (TNS) improvement in circuit delays at an average of 2.54% wire length increase. Besides, runtime is kept at the same level as FastPlace.

Keywords: physical design, critical path, static timing analysis
Classification: Integrated circuits

References

[1] T. Kong, "A Novel Net Weighting Algorithm for Timing-Driven Placement," *Proc. Int. Conf. Computer Aided Design*, San Jose, USA, pp. 172–176, Nov. 2002.
[2] W. Hou, X. Hong, W. Wu, and Y. Cai, "A Path-based Timing-driven Quadratic Placement Algorithm," *Proc. Conf. Asia and South Pacific Design Automation*, Kitakyushu, Japan, pp. 745–748, Jan. 2003.
[3] N. Viswanathan and C. C.-N. Chu, "FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 722–733, 2005.
[4] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," *Proc. Int. Symp. Field Programmable Gate Arrays*, Monterey, USA, pp. 203–213, Feb. 2000.
[5] A. B. Kahng and Q. Wang, "Implementation and extensibility of an analytic placer," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734–747, 2005.
[6] W. Swartz and C. Sechen, "Timing driven placement for large standard cell circuits," *Proc. Conf. Design Automatic*, San Francisco, USA, pp. 211–215, June 1995.

© IEICE 2012
DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012

[7] M. Wang, X. Yang, and M. Sarrafzadeh, "Dragon2000: Standard-cell Placement Tool for Large Industry Circuits," *Proc. Int. Conf. Computer-Aided Design*, San Jose, USA, pp. 260–264, Nov. 2000.

1 Introduction

Modern physical design methodology often needs design iterations for performance/power issues. Especially at top-level design in planning stage, fast evaluation and initial optimization iterations are very important for designers to make decisions about the entire product performance and design cost. Hence, crucial requirements on critical path timing convergence in iterative performance driven VLSI physical designs have emerged many great works in both academia and industry. Since the dominant interconnect delay can be handled in placement by the interconnect wire length optimization, much work can be done at placement stage. So far, timing-driven placement algorithms have been studied extensively, and they can be roughly categorized into two classes, i.e. the net-based [1] and the path-based [2]. The net-based usually transforms timing information into the net weights or net constraints, while the path-based algorithm direct works on the specific paths by handling the nets/cells on the paths. However, it is hard for the net-based algorithms to determine the exact net constraints and there may exist over-constraints since the delay budgeting is usually done in the structure domains of circuit. In order to quickly determine the convergence of the design in the design planning stage, developing a fast placement algorithm, especially the timing driven fast placement algorithm is extremely urgent. Fastplace [3] is a well-known fast placement algorithm. However, the timing driven placement based on Fastplace has not yet been developed. Our work focuses on solving the problems faced in the fast timing-driven global placement stage. In the subsequent design stage, the detailed placement algorithms with heavy timing strategy still need to be used to further optimize timing behavior, such as Dragon2000 [7], Aplace [5] etc. In this paper, specially designed for fast performance evaluation and convergence in higher planning levels of physical design, a critical-path based timing driven placement algorithm, TimFastPlace, is proposed.

2 FastPlace

FastPlace uses a classic quadratic programming based placement model which is formulated as Eq. (1) shows.

$$\min \phi(x,y) = 1/2\boldsymbol{X}^T\boldsymbol{C}\boldsymbol{X} + \boldsymbol{d}_x\boldsymbol{X} + 1/2\boldsymbol{Y}^T\boldsymbol{C}\boldsymbol{Y} + \boldsymbol{d}_y\boldsymbol{Y} \tag{1}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the coordinates of cells, $\boldsymbol{d_x}$ and $\boldsymbol{d_y}$ are generated by the connection between the cells and pads, the connection matrix $\boldsymbol{C}$ represents the relationship between cells. Since the programming in (1) is convex, analytic placement such as FastPlace just needs to solve a linear system in each

© IEICE 2012
DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012

iteration as Eq. (2) shows.

$$\boldsymbol{CX} + \boldsymbol{d}_x = 0 \tag{2}$$

FastPlace also employs a cell shifting method to spread the cells after each time of equation solution. For further wire length improvement it also uses a heuristic local refinement procedure.

3 TimFastPlace: Critical-path based timing driven FastPlace

3.1 The slack of a path

Given a circuit, the register/latch output ports and the circuit Primary Input (PI) ports would be the combinational logic path starting points, and the register/latch input ports and the Primary Output (PO) ports would be the path ending points. Generally the criticality of a path is justified according to the slack which is defined to be the maximum amount of delay that can be added to the path before it becomes critical. Thus a path's slack can be computed as:

$$slack(\pi) = T - delay(\pi) \tag{3}$$

where T is the longest path delay, $delay\ (\pi)$ is the delay of path π. According to the slacks, the setup/hold violations due to late or early signal skews can be found, on which the optimization methods can work.

3.2 The critical path based timing model

In this section, a Critical Path Based Timing Model (CTM) is proposed. Path-based algorithms handle critical nets by assigning them different weights/constraints. We consider two aspects for assigning net weights. One is the criticality counting, which means that an edge having more critical paths passing should get a higher weight. CTM is based on the net weighting scheme as Eq. (4) shows.

$$w(e) = \sum_{e \in \pi} D(slack(\pi), T) \tag{4}$$

where, D is a given weighting function; $slack(\pi)$ is the slack of the path π, and T is the longest path delay. The weighting function D is normally given as a monotonically decreasing function which means to put bigger weights on smaller-slack paths (minus slacks). The other aspect we considered is to determine the exact weight on a given net as to one path, i.e. to determine the function D in Eq. (4). We use the Eq. (5) to determine the function D.

$$D(slack(\pi), T) = \begin{cases} (1 - slack(\pi)/T)^{\delta} & , s < 0 \\ 1 & , s \geq 0 \end{cases} \tag{5}$$

Furthermore, the wire model for FastPlace is the hybrid model, which uses the clique model for 2/3-pin nets, and the star model for nets with four or more pins [3]. During timing-driven placement, we need to assign different net weight according to different net type. Therefore, we introduce a weight

© IEICE 2012
DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012

control parameter γ for the CTM, as shown in Eq. (6).

$$w(e) = \begin{cases} f(D(slack(\pi),T)) \times k \times \gamma & k > 3, s < 0 \\ f(D(slack(\pi),T)) \times \gamma & 2 \leq k \leq 3, s < 0 \\ 1 & s \geq 0 \end{cases} \tag{6}$$

where

$$f(D(slack(\pi),T)) = 1 + \left(\sum_{e \in \pi} (D(slack(\pi),T) - 1) \right)$$

k is the number of pins of the net, δ is the criticality exponent, $T = (1 - \mu) max_{\pi}\{delay(\pi)\}$, μ is the percentage of expected improvement of the longest path delay after several timing-driven iterations [5].

In general, the critical path delay on path π can be formulated as the sum of all the critical net delays on the path. The method for computing the delay on a critical path is given in [6] as Eq. (7) shows.

$$delay(\pi) = \sum_{\forall e \in \pi} R_n \left[C_{L_x} S_x(e) + C_{L_x} S_y(e) \right] + \sum_{\forall e \in \pi} \left[T_n + R_n C_{G_n} \right] \tag{7}$$

where T_n is the intrinsic gate delay, C_{Gn} is the gate input capacitance, R_n is the equivalent driver resistance, C_{Lx} and C_{Ly} are the capacitance per length in the x and y directions respectively, and $S_x(e)$ and $S_y(e)$ are the horizontal and vertical spans, respectively, of the bounding box of net e.

3.3 TimFastPlace: Design and implementation

The proposed timing driven model is applied in the design planning step in physical design, in which the critical paths are normally extracted by the standalone Static Timing Analysis (STA) tools for fast design convergence in both industry and academia. In this paper, we use Synopsys Astro to extract the critical paths; our algorithm handles those critical paths in a fast manner which can greatly help the design planning convergent. We compute the weight of each critical net according to the Eq. (6) for each critical path. Hence, the time complexity of the proposed method is O(n), where n is the number of critical nets. Analytical timing-driven process often requires a few iterations, and the weight assigned to the net on the critical path should be accumulated during iterations. We dynamically update the critical path set according to the results of the STA and adjust net weights based on timing criticality of each net during iterations. TimFastPlace uses the proposed CTM to optimize the critical path delay at the equation solving stage as following steps.

Step 1: Analyze a critical path subset to find out cells which the critical path passed, and then compute delay (π) according to Eq. (7).

Step 2: Establish matrix to be solved by inputting circuit information, divide placement region into equal-sized bins, and solve equation to obtain initial coordinates of each cell.

Step 3: Update matrix: compute for each net, compute $w(e)$ for each net e based Eq. (6), then use the weight $w(e)$ to modify matrix $\boldsymbol{C}$ and vector $\boldsymbol{d}$ in Eq. (2) to optimize the timing behavior.

© IEICE 2012

DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012

Step 4: Perform cell shifting to evens out the placement by distributing the cells over the placement region.
Step 5: Solve equation to obtain the coordinates after cells moving.
Step 6: Repeat 3-5, until Bin Utilization is less than α.
The value of δ in Eq. (5) is specified every time between 1 and 20 in each iteration, the weight of each critical path will be successively calculated, by which the connection matrix $\boldsymbol{C}$ and vector $\boldsymbol{d}$ in Eq. (2) will be modified and the linear system is then solved. The placement will exit until Bin Utilization [2] is less than α, which is set to 1.0 in our experiment. Finally, LEF/DEF is input to STA to verify timing behavior of cases.

4 Experimental results

FastPlace and TimFastPlace are implemented in C++ and tested on the Intel Xeon E5620 server with 8 GB memory. The test circuits used include four industry cases and three ISCAS89 cases which are remapped to the SMIC 180 nm technology for real industrial STA evaluation. Worst Negative Slack (WNS) and Total Negative Slack (TNS) are two timing closure metric to measure the effectiveness of timing driven placement algorithms.

For a direct review of the effectiveness of the proposed path based timing-driven FastPlace, Fig. 1 is the trend graph after the global placement for s5378 of ISCAS89 benchmark. The abscissa in Fig. 1 represents the number of iterations and the vertical axis is the approximate delay of the longest critical path (denoted by approx_delay). We can see from this graph that delay of the critical path increases with the number of iterations first increased and then decreased. At first, the iteration of all the cells from the middle of placement region begin to spread out, and this moment the length of the longest critical path increases, so the delay also increases; After the cells even basically, our algorithm makes the length of the longest critical path decrease, and thus delay of the longest critical path begins to decline. As seen from Fig. 1, TimFastPlace can make delay of the longest critical path greatly reduced. It shows the timing driven strategy works well on this case

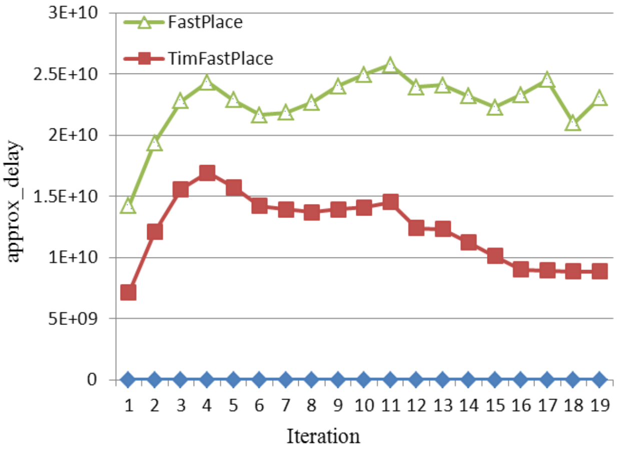

Fig. 1. Optimization trend in comparison with FastPlace

© IEICE 2012
DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012

in comparison with FastPlace.

Table I gives the detailed results in comparison with the original FastPlace. The critical path driven FastPlace of this paper can achieve up to 30.83% (23.42% on average) WNS improvement and up to 25.07 (18.87% on average) TNS by STA. The Total Wire Length (TWL) is increased by 2.54% on average and the run time (in seconds) is kept in the same level as FastPlace.

Table I. Comparison of experimental results with FastPlace and TimFastPlace

Circuits	#cells	#nets	FastPlace				TimFastPlace			
			CPU	TWL	WNS	TNS	CPU	TWL Inc.	WNS Imp.	TNS Imp.
s5378	1.5k	1.5k	2	8.96e7	-4.22	-125.83	2	0.56%	25.17%	20.99%
s38584	12.2k	12.2k	32	1.39e9	-14.81	-434.70	36	2.16%	28.63%	25.07%
s38417	14.8k	14.8k	46	1.54e9	-16.45	-489.31	53	4.55%	30.83%	19.13%
F1	6.1k	7.5k	13	2.34e6	-6.71	-181.40	13	1.19%	15.77%	11.26%
Te2	9.2k	7.2k	24	1.12e6	-7.28	-213.17	25	0.53%	18.91%	16.24%
Te4	18.4k	14.3k	51	4.12e6	-19.14	-570.92	58	2.87%	23.04%	20.59%
F3	62.1k	62.9k	217	1.02e7	-27.09	-792.68	241	5.93%	21.60%	18.80%
Average								2.54%	23.42%	18.87%

5 Conclusion

A critical-path based timing driven FastPlace, named TimFastPlace, has been described. It uses an iterative critical path based net weighting model to optimize the critical path delay at the equation solving stage. Experimental results show that TimFastPlace can effectively optimize the timing behavior.

Acknowledgments

This work was supported by the State Key Development Program for Basic Research of China under Grant No 2011CB302902.

© IEICE 2012

DOI: 10.1587/elex.9.1310
Received July 22, 2012
Accepted July 30, 2012
Published August 17, 2012