Timing-Driven Placement using Design Hierarchy Guided Constraint Generation *

Xiaojian Yang Bo-Kyung Choi Majid Sarrafzadeh

Computer Science Department, University of California, Los Angeles, CA 90095
xjyang,bkchoi,majid@cs.ucla.edu

Abstract

Design hierarchy plays an important role in timing-driven placement for large circuits. In this paper, we present a new methodology for delay budgeting based timing-driven placement. A novel slack assignment approach is described as well as its application on delay budgeting with design hierarchy information. The proposed timing-driven placement flow is evaluated using an industrial place and route flow. All results are reported after detailed routing and timing analysis. Compared to Cadence QPlace, the proposed placement flow generates placements with shorter clock cycle and better routability. Our preliminary experimental results show that considering design hierarchy is a promising way to handle timing optimization problem.

1. Introduction

Timing-driven placement is one of the most important steps to meet circuit performance in VLSI design. The problem has been studied for two decades, yet it remains challenging because of the dramatically increasing circuit size and the dominance of the interconnect delay in the deep sub-micron design. Traditionally, successful timing-driven placement approaches fall into three categories: path-based algorithms [15, 14, 22], net weighting algorithms [9, 1] and delay budgeting algorithms [19, 12, 21]. Recent works on timing-driven placement present novel approaches. A generic placement framework based on quadratic and force-directed method was proposed in [11]. In [6], the authors presented a modified force-directed method using additional links between source and sink of a path. The approach in [4] suggests a new model of net criticality that can be used to guide commercial placement tools.

Delay optimization is indeed path-based problem. However, path-based algorithms are usually computationally expensive because of the exponential number of near-longest paths. Net-based algorithms, including net weighting and delay budgeting, are generally faster but lack accuracy. Net weights cannot directly affect placement results since it is hard to control the wirelength by assigning a weight to each net. Delay budgeting usually over-constraints the placement problem, and the efficient way to achieve good delay bounds remains a hard problem. In practice, the good approach of the timing-driven placement often relies on the complexity and features of the designs, and the tightness of the design constraints.

Recent work on wirelength optimization suggests that hierarchical [2] or multi-level [3, 23] approach is indispensable to efficiently solve large-scale placement problem. Correspondingly, timing-driven placement problem using hierarchical or multi-level framework is worth research attention. Several recent work address this problem. Ou et al. [20] adopted a net-cut control method in min-cut placement. Halpin et al. [13] presented a linear programming based net length control for recursive bisection placement. Kahng et al. [16] extended the placement flow in [2] to incorporate direct minimization of the critical path. The above approaches shows the trend of the combination between traditional timing-driven approach and top-down placement flow. However, delay budgeting, as a previously effective approach, has not been applied into the top-down placement framework.

In this paper, we study the delay budgeting problem in hierarchical or multi-level placement flow. Our contribution in this paper can be summarized as follows:

- We have designed a new model to describe the criticality of the interconnect in hierarchical multi-level placement;
- We have extended the previously known zero-slack assignment algorithm to achieve flexibility based on the proposed criticality model; and
- We have implemented a timing-driven placement tool which aims at solving large-scale placement problems. It combines the delay budgeting approach and a new multi-level placement flow.

The rest of this paper is organized as follows. Section 2 gives preliminaries. Section 3 describes our novel model of net criticality for interconnection, In Section 4, we introduce a multi-level placement flow which considers the global interconnects of different design hierarchies and use this information to guide the slack assignment of the placement. We show our experimental results in Section 5, and conclude in Section 6.

2. Preliminaries

2.1 Timing constraint graph

We use the similar model as in [19, 21]. A *circuit* corresponds to a hypergraph $G(V, E)$, which consists of a set of cells $V = \{v_1, v_2, \ldots v_n\}$ and a set of nets $E = \{e_1, e_2, \ldots, e_m\}$. We form a *directed acyclic graph (DAG)* based on the hypergraph. Each node in the DAG corresponds to a non-sequential cell, a I/O pad, or an input/output pin of a sequenstial cell. We define the *source* nodes as the input pads or the output pins of the sequential cells, and the *sink* nodes as the output pads or the input pins of the sequential cells. There is a direct edge from node v_i to v_j if the output of cell v_i connects to the input of cell v_j. We assume that each cell has only one output, otherwise the node is duplicated to ensure the single output.

For each node v_i in the DAG, we define arrival time a_i and require time r_i as the latest input arrival time and earliest input require time of the cell which corresponds to node v_i.

The arrival time and require time for all nodes can be computed recursively by:

$$a_i = \max_{j \in FI_i} a_j + d_j \quad (1)$$

*This work was supported by NSF under Grant #CCR-0090203

$$r_i = \max_{j \in FO_i} r_j - d_i \quad (2)$$

where FI_i and FO_i are all fanins and fanouts of node v_i, respectively, and d_i is the internal delay[1] of the cell corresponding node v_i. Initially, we assume that all delays caused by interconnects are zero.

We associate a slack s_i to node [2] v_i as the additional delay of v_i. Thus,

$$a_i + d_i + s_i \leq a_j \quad \forall \quad v_i \in V, \quad v_j \in FO_i \quad (3)$$

Assuming that the arrival time of source nodes are zero, and that the timing constraint of the circuit is T, we have

$$a_i = 0 \quad \forall \quad v_i \in V_{source} \quad (4)$$

$$a_i \leq T \quad \forall \quad v_i \in V_{sink} \quad (5)$$

where V_{source} and V_{sink} are the set of source nodes and sink nodes, respectively.

The delay budgeting problem seeks to assign non-negative values to slacks. The assignment is *feasible* if the constraints of (3), (4) and (5) are satisfied.

The solution of delay budgeting problem can be converted into delay bounds for interconnects. Specifically, if we assume that a linear delay model[3] is adopted, the slack s_i of node v_i indicates that the maximum delay of the net driven by v_i is no larger than s_i. A placement is *timing feasible* if all nets satisfy the delay bounds converted from a feasible slack assignment (and thus the entire circuit satisfies the timing constraint). The conversion of delay bounds from slacks depends on the delay model, driving strength of the cell, and load characteristics.

2.2 Delay model

In this work, we use the delay model shown in Figure 1.

$$d = I_i + r(c_e + c_j) + r_e c_j$$

where I_i is the intrinsic delay of gate i, c_e and r_e are the capacitance and resistance of net e, respectively. c_j is the input pin capacitance of gate j. If gate i drives multiple gates, the summation of c_j for all fanout gates replaces c_j.

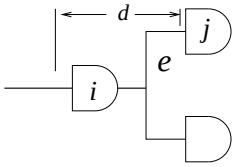

Figure 1: Delay model used in this paper

We use wireload model to estimate the resistance of a net. Half perimeter of the bounding box is the estimated wire length for a net. The intrinsic delays of the gates are not provided in our library file. Instead, we use the result from linear regression on the data in library look-up table.

[1]Here we assume that the internal delay from different input pins to the output pin are the same. We will interchangeably use *slack of node* and *slack of net* in this paper.

[2]It is also associated to the net which node v_i drives, under the assumption that each node has one output and the net delay is calculated using the bounding box of the net.

[3]The delay of a net is proportional to the half-perimeter of the bounding box of the net.

3. Slack Assignment

3.1 Previous work

Previous approaches on slack assignment include zero-slack assignment (ZSA) [19], weighted slack assignment [24], fast slack allocation [18], assignment with maximum flexibility [21], and maximum independent set algorithm (MISA) [5]. Efficient slack allocation for large circuits is still a hard problem, and it is the crucial step in delay budget based timing-driven placement.

In delay budgeting problem, the goal of slack assignment is to achieve a set of delay bounds for all nets. It is therefore desirable to assign non-zero delay to all nets. On the other side, evenly distributing slacks (ZSA) ignores net criticalities, thus over-constraints longer wires while giving freedom to shorter wires.

A number of approaches have been proposed to obtain uneven slack distribution. In [24] the slacks are assigned proportionally to the weights of the nets, which are derived from the circuit characteristics. The same method can be used in [19]. In [12, 21], $log(x)$ as an objective function was adopted to measure the quality of the slack assignment. In this work, we propose a new objective function which tries to better describes the slack requirement in placement problem.

3.2 A new objective function for slack assignment

In slack assignment, it is natural to expect that all nets are assigned to non-zero slacks, otherwise the delay budgets corresponding to this slack assignment will never be met during the placement. Therefore, for a given net in the slack assignment problem, the maximum gain is obtained when we increase the slack from zero. However, if we keep increasing the slack for this net, the gain from the newly assigned slacks decreases. If the slack of a net is beyond a certain threshold, there is no gain from allocating more slack on the net, i.e., the gain is zero. We seek for objective function to describe this feature. Previously used $log(x)$ is close but not an ideal candidate. We propose a new objective function:

$$f(x) = 1 - e^{-cx} \quad (6)$$

where c is a constant determined by the criticality of the net.

Comparing this objective function (6) with previously used $log(x)$, one can see that there is an upper bound for the former but not the latter. This upper bound refrains excessive slack allocation to a single net. Introducing the objective function helps modeling and solving slack assignment problem.

3.3 Slack assignment for paths

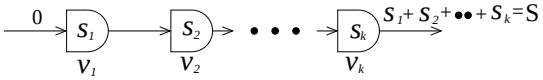

Figure 2: Slack assignment problem in the single path

Consider a slack assignment problem for a single path (Figure 2). There are k nodes on the path. Each node is originally assigned a slack s_i. The total (incremental) slack to be assigned along this path is S. For node v_i assigned with slack x_i, the gain of the slack is $1 - e^{-c_i x_i}$ where c_i is the coefficient of the objective function of node v_i. We want to solve the following problem:

$$\text{maximize} \quad f(\mathbf{x}) = \sum_{i=1}^{k} (1 - e^{-c_i(s_i + x_i)})$$

$$\text{s.t.} \quad \sum_{i=1}^{k} x_i = S$$

Let

$$g(x) = f(\mathbf{x})|_{x_k = S - x_1 - x_2 - \ldots - x_{k-1}}$$

Then solving the problem is equal to solving the following system:

$$\begin{cases} \partial g(\mathbf{x})/\partial x_1 & = & 0 \\ \partial g(\mathbf{x})/\partial x_2 & = & 0 \\ \cdots & & \\ \partial g(\mathbf{x})/\partial x_{k-1} & = & 0 \\ x_1 + x_2 + \ldots + x_k & = & S \end{cases}$$

We can rewrite the system as:

$$\begin{pmatrix} \mathbf{A} & \mathbf{a} \\ \mathbf{1}' & 1 \end{pmatrix} \mathbf{x} = \mathbf{b} \tag{7}$$

where

$$\mathbf{A} = \begin{pmatrix} c_1 & & & \\ & c_2 & & \\ & & \ddots & \\ & & & c_{k-1} \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} -c_k \\ -c_k \\ \vdots \\ -c_k \end{pmatrix}$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} \ln(c_1/c_k) - c_1 s_1 + c_k s_k \\ \ln(c_2/c_k) - c_2 s_2 + c_k s_k \\ \vdots \\ \ln(c_{k-1}/a_k) - c_{k-1} s_{k-1} + c_k s_k \end{pmatrix}$$

This sparse linear system can be directly solved as:

$$x_k = \frac{S - \sum_{i=1}^{k-1} ((\ln(c_i/c_k) - c_i s_i + c_k s_k)/c_i)}{1 + \sum_{i=1}^{k-1} (c_k/c_i)}$$

$$x_i = \frac{\ln(c_i/c_k) - c_i s_i + c_k s_k}{c_i} + \frac{c_k}{c_i} x_k$$

The proposed slack assignment approach has the following features:

- It assigns slacks to the nodes based on their weights (which are reflected by c_i). Nodes with higher weights (thus smaller c_i in the objective function) will be assigned larger slacks.
- It can be extended to trees instead of paths. The similar linear system can be formed and solved. While it is not clear how to handle trees using proportional assignment method [24].
- It is fast and can be used in incremental assignment. Previously assigned slacks can be increased or decreased, constructing more reasonable slack distribution. Additional constraints can ensure positive incremental slacks. In our implementation we set s_i to zero for $i = 1, \ldots, k$, ensuring the slack assignment approach converges fast.

3.4 Modified ZSA slack assignment

For the entire circuit, we use modified ZSA to assign slacks. The algorithm starts from a timing analysis and computation of slacks for each node. Then a path with minimum positive slack on each node is identified. The proposed approach is then applied to optimally assign slacks on this path. The slack assigned to the node is added to its delay. An incremental timing analysis is followed and the new slacks for each node are updated. This completes one iteration and the next minimum slack path is identified.

4. Multi-Level Placement with Predefined Hierarchy

In this section, we propose a new timing-driven placement flow based on delay budgeting method. Consider a top-down placement using recursive quadrasection approach (Figure 3). We name the wires connecting different partitions by the first quadrasection *level-1* nets. Similarly, the wires connecting different partitions by the second quadrasection but not *level-1* nets are called *level-2* nets, and so on. It is obvious that, after non-timing driven placement, higher level nets (lower level number) have longer average length than lower level nets. It is therefore necessary to take the net length into consideration in delay budgeting process. Previous delay bound approaches have not considered the effect of global nets in the slack assignment.

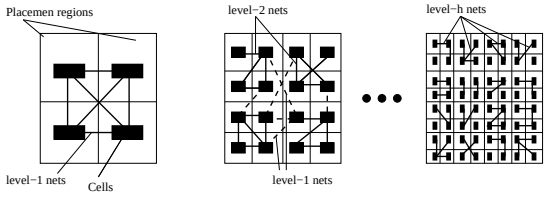

Figure 3: Global nets of different level in top-down quadrasection placement

In delay budgeting for net bound placement, it is desirable to assign larger slacks to potentially longer nets, as long as the slack assignment is feasible. If slacks are evenly distributed to nets with different lengths, the delay bounds will over-constraint the placement process and in some cases even cause the placement failure on finding a feasible solution.

In timing-driven placement, we aim to shorten those *critical* global nets. The optimization step, e.g. partitioning or annealing, requires the delay budget information to guide the detailed operations. In the traditional hierarchical ([10, 2]) or multi-level ([23, 3]) placement flow, the hierarchy information are not known until the finer placement level. It is not effective to perform timing optimization at this point, because the global placement has been determined. Delay bound information should be obtained at earlier placement level.

4.1 Achieving Hierarchies

The authors in [25] proposed a new methodology of multi-level placement flow. The main method is to partition the circuit into small size clusters, and then execute multi-level placement based on the known design hierarchy. We use the similar flow in this work. At the beginning of the placement, a recursive bisection step is applied on the circuit. After every two bisections (i.e. one quadrasection), the global nets at this level are identified and labeled. The bisection step stops when the average number of cells for each cluster is less than a threshold. The remaining unlabeled nets are also tagged as the highest level number. The net level information will guide the delay budgeting process.

4.2 Estimating Net Lengths

Estimation on length of external net has been studied previously. We adopt the same method used in [8, 7]. In a hierarchical placement flow, the average length of the global nets of level k can be estimated as the following:

$$\overline{l_k} = \frac{4(\frac{4\lambda}{3} - \frac{1}{3\lambda}) + 4\lambda}{6}$$

where $\lambda = 2^{H-k}$ and H is the total number of hierarchical levels.

The estimated wirelength of the nets are converted into net criticality, and then the weight in delay budgeting algorithm.

4.3 Delay Budgeting considering Design Hierarchy

We use the following function to convert the estimated net length into the node weight[4] for slack assignment:

$$c_i = \frac{\max_{v_j \in V}(\sqrt{l_j \cdot f_j})}{\sqrt{l_i \cdot f_i}} \quad (8)$$

where l_i is the estimated wirelength and f_i is the fanout of the net which node v_i drives. In general, a lower hierarchical level number or a larger fanout indicates a longer net. Therefore a lower coefficient c_i is given for allocating a larger slack in the later delay budgeting algorithm.

We integrate the slack assignment approach described in Section 3 into modified ZSA algorithm to assign slacks for each node. After the slack assignment, the bounds of length for all nets are determined. A multi-level placement flow is then applied to optimize the placement using the net bound information.

4.4 Multi-level Placement using net bounds

In this modified multi-level placement flow, we use simulated annealing approach to improve the placement. At each level, the cells are clustered based on their hierarchical labels and these clusters are placed into rectangular placement regions. The clusters can be entirely exchanged with other clusters during the annealing. The cost function is:

$$WL + \lambda_1 \sum_{i=1}^{|E|} max(l_i - b_i, 0)$$

where l_i is the current length and b_i is the bound of net i. WL is the total bounding box wirelength. λ_1 is a parameter for adjustment between wirelength optimization and delay optimization. Both items in the cost function can be incrementally updated when two clusters are switched position.

At the transition from one placement level to the next, since the design hierarchy has been obtained in the previous recursive bisection step, no more partitioning need to be performed. We exploit the existing partition information (which should be saved) to split the clusters into smaller clusters and the placement flow enters the next level. Therefore, the total time of this mutlileve placement flow does not increase, while the design hierarchy information is obtained earlier.

The entire placement flow is shown in Figure 4. We use hMetis [17] as the partitioning tool. The original circuit is recursively bipartitioned till the pre-calculated level. Global nets of each level are identified and are assigned to an estimated wirelength. The local nets within partitioned clusters are regarded as the *level-H* nets. The coefficient c_i in (6) is determined by the estimated wirelength and the net fanout using (8). Next the modified ZSA algorithm is called to assign slacks according to the new objective function. Such slacks are then converted into delay bounds in terms of net length. During the conversion we assume a RC delay model and the total fanout capacity of the net is considered.

Next, delay bounds are used to evaluate the placement quality in the multi-level placement flow. At each level, the optimization step tried to reduce a combination between total bounding box wirelength and total delay violation. The placement meets the timing constraints if the total delay violation is zero. In general a lower delay violation value corresponds to a better placement in terms of delay.

[4]The node which drives that net will be assigned the weight

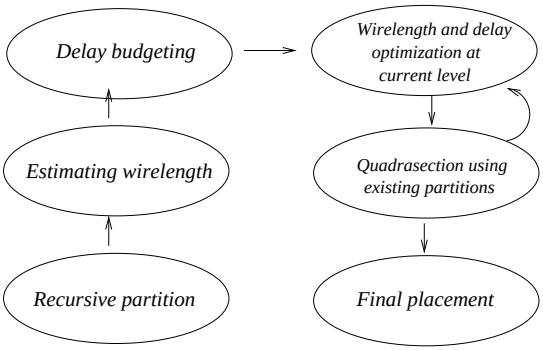

Figure 4: Multi-level placement flow with a prior hierarchy information

5. Experimental Results

We have implemented (a) the proposed new slack assignment approach, (b) a simulated annealing engine with cost function including delay bound violation, and (c) the multi-level placement flow with pre-defined hierarchy information. To evaluate our approach, we test our placement tool using an industrial place and route flow. The results after global and final routing are reported, and they are compared to the industrial place and route outputs.

Specifically, we compare our placement tool with Cadence QPlace (Silicon Ensemble 5.3). Both our placer and QPlace read the same LEF/DEF files and the outputs are fed into Cadence WRoute. After global and final routing, we use Pearl timing analysis tool to show the slack distribution. The final routed wirelength and the minimum slack are reported.

The statistics of the benchmarks are summarized in Table 1[5]. The circuits are acquired from a website provided by [6][6]. We use a 0.18μm standard-cell library as the LEF file. We determined the clock cycles for the circuits by the following way. For each circuit we first use QPlace (non timing-driven mode) to place it. After routing and timing analysis, the minimum cycle period is recorded as the cycle of the design. Therefore for all the circuits, the minimum slacks by QPlace non timing-driven mode are zero (see Table 2). This clock cycle is then used as the constraint for QPlace timing-driven mode.

circuits	*cells*	*nets*	*rows*	*routing layers*	*clock cycle*
matrix	3,083	3,200	56	4	3.89
VP2	8,714	8,789	100	4	4.57
32-MAC	8,902	9,115	101	5	3.85
64-MAC	25,616	26,017	134	5	7.67

Table 1: Tested circuit statistics, including number of cells, number of nets, number of rows, number of routing layers and clock cycle (ns).

Table 2 shows the results by five different placement approaches: QPlace non timing-driven, QPlace timing-driven, our placer without timing optimization, our placer with ZSA delay budgeting, and our placer with new slack assignment described in Section 3. Our placer with hierarchy based slack assignment consistantly achieves the best among five approaches, including QPlace in timing-driven

[5]These are all benchmarks (in LEF/DEF format) available for us at this moment. We are looking for more benchmarks for this work.

[6]Note that not all the benchmarks in [6] are available.

Placement		Results	Circuit			
			matrix	VP2	32-MAC	64-MAC
QPlace	non timing-driven	wirelength	126618	496905	511763	2163585
		slack	0.00	0.00	0.00	0.00
	timing-driven	wirelength	124980	517596	520468	unroutable
		slack	-0.25	-0.19	0.18	unroutable
Our approach	non timing-driven	wirelength	119577	430716	487035	2657055
		slack	-0.06	-0.06	0.09	-2.51
Our Approach timing-driven	ZSA slack assignment	wirelength	118823	430470	470711	2618864
		slack	0.00	0.12	0.11	-1.32
	New slack assignment	wirelength	118833	424379	482252	2655655
		slack	0.09	0.28	0.46	0.97

Table 2: Routed wirelength (micron) and minimum slack (ns) for five placement runs. "unroutable" shows the routing failure. New slack assignment is the method proposed in Section 3.

mode[7].

Figure 5 shows the comparison between QPlace's results and ours. QPlace's results are from timing-driven mode. Our placer uses design hierarchy guided new objective function in slack assignment. Our results are better than QPlace's except for the wirelength of 64-MAC.

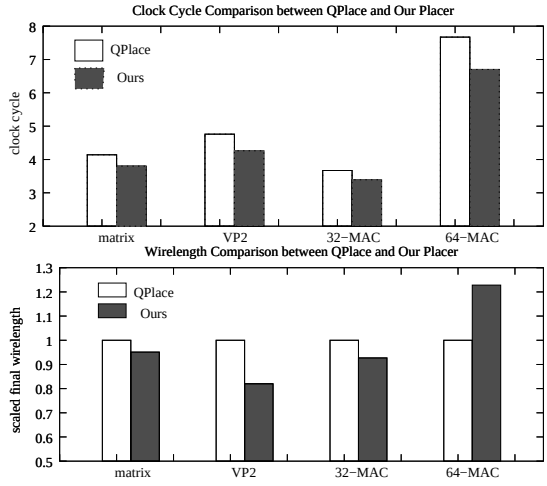

Figure 5: Clock cycle period and routed wirelength comparison between QPlace and our placer. Wirelengths are scaled with QPlace's result as unity. The QPlace's non timing-driven result are used to replace timing-driven result for 64-MAC.

Table 3 shows the total timing violation over all the nets using three different slack assignment. For each iteration in ZSA algorithm, we use even distribution (original ZSA), proportional distribution ([24]) and our exponential based method. The slack assignment created by above three methods are then converted into net length bounds. Such bounds are used for measuring the final placement created by QPlace. The total net length violation for three algorithms are reported. In most cases the new method corresponds to a lower net length violation.

6. Conclusion

[7]For two benchmarks QPlace timing-driven mode creates worse delays than non timing-driven mode. The reason is possibly that the tight constraint mis-guides the optimization during placement.

slack assignment	matrix	VP2	32-MAC	64-MAC
ZSA	1604	2634	2393	3117
proportional	1547	2461	2508	4206
Ours	1532	2070	2576	3502

Table 3: Total net length violation over net bounds created by three methods: ZSA, propotional ZSA and our method.

In this paper, we have presented a new methodology of timing-driven placement. A novel slack assignment approach is described as well as its application on delay budgeting with design hierarchy information. The proposed timing-driven placement flow is evaluated within an industrial place and route flow. The advantage of the new methodology has been confirmed by experimental results.

There are a number of unclear issues in delay budgeting based timing-driven placement approach. One open problem is the determation of the timing constraint during the placement. Neither very tight nor very loose constraint is appropriate for delay optimization. It is of value to find the proper value without accurate delay model and routing information. Another issue is the exact cost function for delay violation. Questions such as whether we should penalize larger violations remain unanswered.

The authors would like to thank the reviewers for their helpful comments. Special thanks to ChihChih Zhou and Stephanie Mantik for their great help in the project.

7. References

[1] M. Burstein and M. N. Youssef. "Timing Influenced Layout Design". In *Design Automation Conference*, pages 124–130, 1985.

[2] A. E. Caldwell, A. B. Kahng, and I. L. Markov. "Can Recursive Bisection Alone Produce Routable Placements?". In *Design Automation Conference*, pages 477–482. IEEE/ACM, June 2000.

[3] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl. "Multilevel Optimization for Large-Scale Circuit Placement". In *International Conference on Computer-Aided Design*, pages 171–176. IEEE, 2000.

[4] H. Chang, E. Shragowitz, J. Liu, H. Youssef, B. Lu, and S. Sutanthavibul. "Net Criticality Revisited: An Effective Method to Improve Timing in Physical Design". In *International Symposium on Physical Design*, pages 155–160. ACM, April 2002.

[5] C. Chen, X. Yang, and M. Sarrafzadeh. "Potential Slack: An Effective Metric of Combinational Circuit Performance". In *International Conference on Computer-Aided Design*, pages 198–201. IEEE, 2000.

[6] Y. C. Chou and Y. L. Lin. "A Performance-Driven Standard-Cell Placer Based on a Modified Force-Directed Algorithm". In *International Symposium on Physical Design*, pages 24–29. ACM, April 2001.

[7] P. Christie and D. Stroobandt. "The Interpretation and Application of Rent's Rule". *IEEE Transactions on VLSI Systems*, 8(6):639–648, 2000.

[8] W. E. Donath. "Placement and Average Interconnection Lengths of Computer Logic". *IEEE Transactions on Circuits and Systems*, 26(4):272–277, April 1979.

[9] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel. "Chip Layout Optimization Using Critical Path Weighting". In *Design Automation Conference*, pages 133–136. IEEE/ACM, 1984.

[10] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard Cell VLSI Circuits". *IEEE Transactions on Computer Aided Design*, 4(1):92–98, January 1985.

[11] H. Eisenmann and F. M. Johannes. "Generic Global Placement and Floorplanning". In *Design Automation Conference*, pages 269–274. IEEE/ACM, 1998.

[12] T. Gao, P. M. Vaidya, and C. L. Liu. "A New Performance Driven Placement Algorithm". In *International Conference on Computer-Aided Design*, pages 332–335. IEEE/ACM, 1991.

[13] B. Halpin, C. Y. Chen, and N. Sehgal. "Timing Driven Placement using Physical Net Constraints". In *Design Automation Conference*, pages 780–783. IEEE/ACM, 2001.

[14] T. Hamada, C. K. Cheng, and P. M. Chau. "Prime: A Timing-Driven Placement Tool Using a Piecewise Linear Resistive Network Approach". In *Design Automation Conference*, pages 531–536. ACM/IEEE, 1993.

[15] M. A. B. Jackson and E. S. Kuh. "Performance-Driven Placement of Cell Based IC's". In *Design Automation Conference*, pages 370–375. ACM/IEEE, 1989.

[16] A. B. Kahng, S. Mantik, and I. L. Markov. "Min-Max Placement For Large-Scale Timing Optimization". In *International Symposium on Physical Design*, pages 143–148. ACM, April 2002.

[17] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. "Multilevel Hypergraph Partitioning: Application in VLSI Domain". In *Design Automation Conference*, pages 526–529. IEEE/ACM, 1997.

[18] W. K. Luk. "A Fast Physical Constraint Generator for Timing Driven Layout". In *Design Automation Conference*, pages 626–631. IEEE/ACM, 1991.

[19] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa. "Generation of Performance Constraints for Layout". *IEEE Transactions on Computer Aided Design*, 8(no.8):860–874, Aug 1989.

[20] S. L. Ou and M. Pedram. "Timing-driven Placement Based on Partitioning with Dynamic Cut-net Control". In *Design Automation Conference*, pages 472–476. IEEE/ACM, June 2000.

[21] M. Sarrafzadeh, D. A. Knol, and G. E. Tellez. "A Delay Budgeting Algorithm Ensuring Maximum Flexibility in Placement". *IEEE Transactions on Computer Aided Design*, 16(11):1332–1341, 1997.

[22] W. Swartz and C. Sechen. "Timing Driven Placement for Large Standard Cell Circuits". In *Design Automation Conference*, pages 211–215. IEEE/ACM, 1995.

[23] M. Wang, X. Yang, and M. Sarrafzadeh. "Dragon2000: Fast Standard-cell Placement for Large Circuits". In *International Conference on Computer-Aided Design*, pages 260–263. IEEE, 2000.

[24] H. Youssef, R. Lin, and E. Shragowitz. "Bounds on Net Delays for VLSI Circuits". *IEEE Transactions on Circuits and Systems*, 39(11):815–824, November 1992.

[25] C-C. Zhang, J. Cong, Z. Pan, and X. Yuan. "Physical Hierarchy Generation with Routing Congestion Control". In *International Symposium on Physical Design*, pages 36–41. ACM, April 2002.