Iterative-Constructive Standard Cell Placer for High Speed and Low Power

Sungjae Kim and Eugene Shragowitz
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455
Email: {sukim, shragowi}@cs.umn.edu

***Abstract*— Timing and low power emerge as the most important goals in contemporary design. Meanwhile, the majority of placement algorithms developed by industry and academia still compete on the bases of the shorter combined interconnection length. In this paper, we present a standard cell placer that has timing and power minimization as main goals and is substantially superior in this respect to the popular commercial timing-driven placer. Simultaneously, the new placer is at least as good as the commercial placer w/r to the combined interconnection length. The improvement in timing and power is achieved by careful balancing between iterative and constructive parts of the placement procedure implemented by dynamic reevaluation of constraints for the constructive part of the algorithm. The new placer improves timing by 23% on the average, or for the same timing, reduces power by 14% on the average. It runs more than 2 times faster than the commercial tool.**

I. Introduction

Timing, power, and their trade-offs are central to contemporary design. These problems emerge in block design as well as in SoC design. Placement tools play a central role in achieving these goals due to rising importance of interconnects for timing and power consumption. While, in many synthesis systems, driver sizing is decided on synthesis step, it is very beneficial to allow a placer to include driver resizing in the set of parameters determined by placement.

The placer described in this paper denoted in the text as the ICP (**I**terative-**C**onstructive **P**lacer) combines coarse-grain timing-driven partitioning in the Initial Step with the adaptive timing-driven construction of rows and selection of driver sizes.

A bibliography on placement algorithms is very large. The recent comprehensive review of placement algorithms is given in [1]. In this review, two papers [2], [3] are mentioned in a category of algorithms that use coarse placement solutions to guide delay budgeting step. This category is, probably, the closest in goals to our work. In [2], LP model is used to influence timing on each level of the partitioning process. Timing-driven Dragon [3] combines timing budgeting based on modification of Zero-slack algorithm [4] with the quadrasection approach. According to the tables given in the paper, timing-driven Dragon was marginally better than the Cadence *QPlace* in timing, while 10-15 times slower.

There are fundamental differences between the works cited above and our work.

1. In works that belong to this group, bounds on net delay are derived from the coarse placement, while in our work, coarse placement is derived from the delay bounds.

2. Delay bounds in [3] are computed by the modified Zero-slack algorithm, while in our work, the IMP algorithm (Iterative Minmax Pert) [5] that provides the optimal values of bound is applied.

3. Actual placement in aforementioned works is iterative. For example, in [3] clusters of cells are moved by the SA (Simulated Annealing) algorithm. In our work, actual placement of cells is constructive, i.e. new cells are added to the partial solution. The placement process is controlled by dynamic evaluation of criticality for non-placed nets. Criticality, in its own term, is defined by dynamic recomputation of net delay bounds. The method of delay bounds conversion to the criticality metrics is given in the paper. The placement problem even for one row is NP-hard. We derived conditions for the optimal solution using a notion of the ”feasible region” and developed a constructive procedure that in many occasions delivers the low bound solutions. This constructive procedure is very fast and produces placement that is correct-by-construction in terms of constraints.

4. Gate resizing is available in the ICP as a part of a net placement. This feature allows to improve results substantially. No such capabilities are observed in the aforementioned group of algorithms.

The rest of the paper is organized as follows: In Section 2, a brief overview of the algorithm is provided. Section 3 presents the initial step of the ICP. Section 4 describes general adaptive step of the placer. Section 5 presents final optimization step. Experimental results are given in Section 6, followed by Conclusion in Section 7.

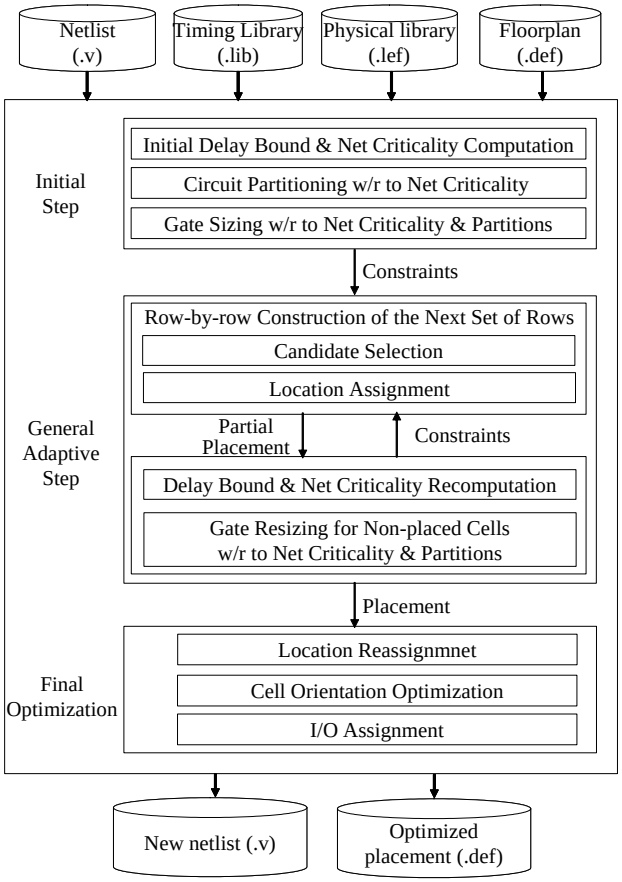


Fig. 1. Overall structure.

II. Overview of the Algorithm

The algorithm consists of 3 main parts: *Initial Step*, *General Adaptive Step*, and *Final Optimization*. Figure 1 shows the structure of the proposed algorithm.

The Initial Step starts with computing delay bounds and criticalities of all nets in design. Computed net criticalities are used as weights to guide recursive circuit partitioning. Gates in the original net list are resized based on net criticalities and delay bounds.

The General Adaptive Step performs constructive placement row-by-row. Construction of each row includes candidate selection based on timing criticality and accumulated length of interconnection to already placed cells for each candidate cell. Cells are placed in positions that minimize timing and interconnection length. After several rows are placed, the delay bounds for non-placed nets are recomputed to reflect results of partial placement, and drivers of these nets are resized to reflect changes in delay bounds. New delay bounds are used for the next iteration of the General Adaptive Step.

The Final Optimization Step minimizes interconnection length in the already completed placement by making cells in one row movable and all others fixed. The same location assignment algorithm, that originally was used in the General Adaptive Step, is applied for this purpose.

III. Initial Step

In this section, algorithms used in the initial step are described. The same delay bound, net criticality computation, and gate sizing are used in the general adaptive step.

A. Net Delay Bounds

Importance of timing budgeting on different stages of VLSI design has been recognized for some time. Computation of bounds on net delays plays an important role in timing budgeting for floorplanning and placement. Two groups of algorithms were introduced for computing bounds: The Zero-slack algorithm [4] and the IMP algorithm [5]. The Zero-slack algorithm finds a node with the minimal positive slack and distributes it along the single path. This algorithm was used in [3]. The IMP algorithm finds slacks at all circuit outputs and distributes them globally based on circuit topology and physical characteristics of nets, i.e. nets with similar physical characteristics may have different bounds depending on their position in the circuit. The IMP algorithm was used in such system-design tools as IBM's HDP [6], Chip Architect from Synopsis and some others.

In the ICP, delay bounds are computed repeatedly by the IMP algorithm, first in the Initial step (Figure 1) to be used in circuit partitioning and gate sizing, and then recomputed after placement of each next set of rows for the purpose of ordering cells for placement and gate resizing. As a result, delay bounds play a much stronger role in this tool than in other previously introduced algorithms.

B. Net Criticality Metric

Prior to the net placement, net criticality could be defined as:

$$Net\ Criticality\ Metric = \frac{Projected\ Net\ Delay}{Net\ Delay\ Bound} \quad (1)$$

Projected Net Delay is difficult to compute prior to layout. But, for the purpose of ranking, it is sufficient to find such approximation of *Projected Net Delay* that is easily computable and to retain ordering of nets in ranking. Under the assumption about normal distribution of the probability density function $f(x)$, the *Net Criticality Metrics* can be rewritten as

$$Net\ Criticality\ Metric = \frac{2m_x}{b_x} \quad (2)$$

where m_x is mathematical expectation of $f(x)$, i.e. the center of the interval for possible values of random variable x (projected value of net delay) and b_x is a net delay bound. The net delay characteristics of the circuit, i.e. mathematical expectation of net delay $m(x)$ could be derived from the statistical data. However, such statistical data are also unknown.

But, normal distribution is characterized by a linear regression function $m(x) = \alpha x^*$, where x^* is a net parameter.

$$\frac{m(x_2)}{b_2} > \frac{m(x_1)}{b_1} \Leftrightarrow \frac{x_2^*}{b_2} > \frac{x_1^*}{b_1} \quad (3)$$

i.e. ranking of nets is preserved when mathematical expectation of a net delay is replaced by a net parameter, i.e. relation x_i^*/b_i can be used for net ranking [7]. Therefore, *Net Criticality* for the purpose of ranking nets could be computed as:

$$Net\ Criticality\ Metric^* = \frac{Net\ Parameter}{Net\ Delay\ Bound} \quad (4)$$

In our current implementation, we used the number of pins as the *Net Parameter*.

C. Circuit Partitioning

The ICP uses a state-of-the-art min-cut hypergraph partitioner [8]. Unlike other partitioning-based systems, which produce fine-grain partitions [3], [9], the ICP produces the coarse partitions (200-300 gates on the average for each partition). Hypergraph partitioners allow to assign different weights to edges and to target different functions in the partitioning process. In our case, weights are used to influence timing. They are derived from the net criticality metrics. Partitioning proceeds recursively until partitions achieved the size limit.

D. Gate Sizing

Gate resizing is a traditional technique to improve timing. Typically this technique is used either before or after physical design and leads to increase in number of iterations and design time. The ICP allows to merge this step with the placement. Gate resizing for the standard cell placer is implemented as selection of the new gate sizes from the standard cell library. The drivers of nets with the high values of *Net Criticality Metric* are selected for upsizing. Reduction of delays on such nets may reduce delays of the critical paths. Constraints in upsizing algorithm come from the potential capacitance limit violation for drivers of upsized cells, as well as from the ability of partitions to accommodate larger cells. After the new sizes are selected, the new utilization factors are generated based on the area increments and remaining capacities of partitions.

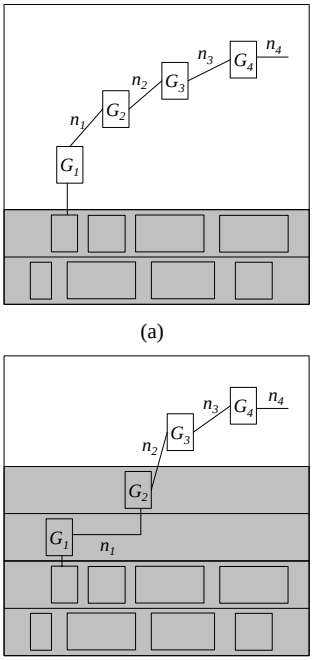
If the *Net Criticality Metric* is low, the driver of such net could be downsized. The downsizing is constrained by the net delay bounds. The new gate size is selected in such way that the increased delay $\leq$ the net delay bound. The smallest gate size that satisfies this constraint and does not violate the capacitance limit is selected as the new size of the gate.

IV. General Adaptive Step

The general adaptive step consists of several major tasks:
- Recomputation of net delay bounds.
- Candidate selection for placement inside the slice with the substeps:
  - -Feasible region identification
  - -Feasible region verification
- Cell location assignment

A. Adaptive Timing Budgeting

Because the partial placement provides more realistic data on actual delays of placed nets, this information should be taken into account in recomputing delay bounds of non-placed nets. The adapted bounds will guide the placement of non-placed cells. Figure 2 shows an example.



(a)

(b)

Fig. 2. Timing rebudgeting.

In Figure 2(a), the cells G_1, G_2, G_3, and G_4 have not been placed yet. After iterations of construction process, G_1 and G_2 are placed as shown in Figure 2(b). After the placement of G_1 and G_2, delay of the net n_1 is reevaluated and the net delay bounds for n_2, n_3, and n_4 are recomputed.

The exact routed wire length of net n_1 cannot be computed in the placement stage because it is determined by a router after placement. The ICP, however, can compute the exact HPWL from the partial placement. Using the HPWL, resistance and capacitance of the net are estimated from the wire load table of the standard cell library. The estimated resistance and capacitance are used to compute the delay of

net n_1 using the RC delay model.

Without this adaptive timing rebudgeting in response to the dynamic delay change, the paths which were not critical may emerge as the critical paths.

B. Candidate Selection

Figure 3 illustrates the candidate selection process. Dark region is a partial placement.

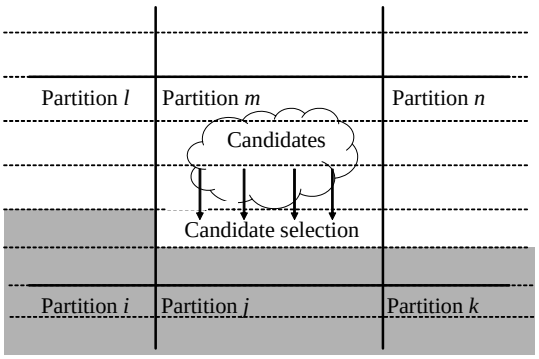


Fig. 3. Partial placement and candidate selection.

Candidate cells are selected for each slice defined as a part of the row inside of one partition. Before the selection of candidates, the total available routing tracks and the slice capacity are computed. The *Cell Criticality* is calculated for all non-placed cells in the partition. The cells are sorted by the values of *Cell Criticality Metrics* and placement starts from the cell with the highest value of *Cell Criticality*. The feasible region for a candidate is identified and verified. The candidate cells which have the verified feasible region are selected for placement in the slice. After the selection of each candidate, the total available routing tracks and capacity of the slice are updated.

1) Cell Criticality Metric: Criticality metric for cells is defined based on criticality of the nets incident to the cell weighted by coefficients measuring connectivity to already placed and not yet placed cells. The general form of the criticality metric for a cell C is defined as:

$$Cell\ Criticality\ Metric(C) = \sum_{n \in C} NCM_n \times CP_n, \quad (5)$$

where NCM_n = *Net Criticality Metric* of net n and CP_n = connectivity to the partial placement of the net n. CP_n is given by (# of placed cell in net n / # of pin of net n). In fact, based on the previous consideration, for the purpose of cell ordering, this measure could be replaced by

$$Cell\ Criticality\ Metric(C) = \sum_{n \in C} NCM_n^* \times CP_n, \quad (6)$$

where NCM_n^* is presented in the form x_i^*/b_i.

2) Feasible Region Identification: The feasible region for the candidate cell is a region in a row where the sum of the horizontal components of connections to the already placed incident cells is a minimal constant.

The placement of the newly selected candidate increases the wire length of the partial placement. Increment should be minimized by placing the candidates in their feasible regions. Figure IV-B.2 shows examples of feasible regions.

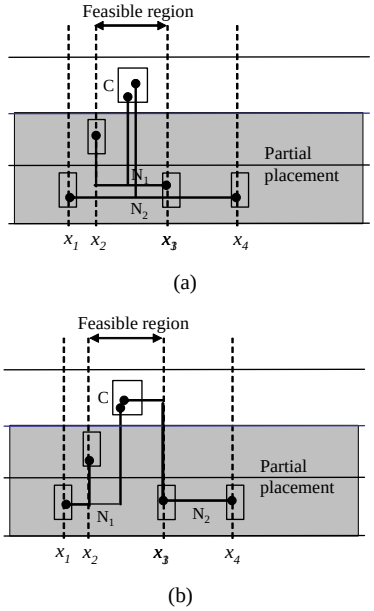


Fig. 4. Examples of feasible regions.

In Figure IV-B.2, the candidate cell C is considered. C is connected to the nets N_1 and N_2. N_1 and N_2 have other already placed cells in the partial placement. In (a), the horizontal coordinates of N_1 and N_2 are [x_2 x_3] and [x_1 x_4]. For any position of the cell C in the region [x_2 x_3], the sum of horizontal components of new connections to the nets N_1 and N_2 is the minimal constant. In case (b), the feasible region is defined as [x_2 x_3] and the minimal size of the horizontal components is ($x_3 - x_2$).

3) Feasible Region Verification: The feasible region for the candidate could be already occupied by the candidates with the higher criticality values. So, the availability of the space in the feasible region for placement of the candidate should be verified before placing it. For the purpose of verification, all candidate cells are assumed to be placed in the centers of their feasible regions. The *capacity* of the feasible region is $FR_{max} - FR_{min}$, where FR_{max} and FR_{min} are maximum and minimum X-coordinates of the feasible region respectively. The *sum* of widths of already selected candidates, which are assumed to be placed in [FR_{min} FR_{max}], is computed. If *capacity* is greater than the (sum + the width of the candidate), the candidate is finally selected for placement and guaranteed to be placed in the

feasible region. Otherwise, the candidate is deferred to the next row. However, there are cases where deferring increases vertical connection to such extent that the total wire length increment is larger than the horizontal connection increment for placing the candidate out of the feasible region. In such cases, the candidate is still selected for placement in the row and its feasible region is extended.

If the verification of the feasible region for a candidate fails, the candidate is not considered again for the selection in the row and delayed for the next row.

C. Cell Location Assignment

After feasible region identification and verification, the selected candidates are placed in their feasible regions. The location assignment algorithm produces overlap-free partial placement. The search for the optimal location with the minimal length of interconnection for each candidate inside the feasible region is performed by the linear complexity search of all possible locations.

V. Final Optimization

In the final optimization step, the locations of all cells are reassigned to their best position in a row by a backward propagation path from the last row assigned to the first row using the same location assignment algorithm as in the general adaptive step.

Proper cell orientations decrease the wire length and increase routability. For each cell, the HPWLs for the current and flipped orientation are computed. The orientation with the smaller HPWL is assigned as the final orientation of the cell.

The ICP can handle both the fixed and non-fixed I/O pins. The fixed I/O pins affect the feasible region identification for the cells connected to I/Os during the construction process. The non-fixed I/O pins are placed as regular cells after placement of all cells is completed. The only difference is that their potential locations are on the periphery of the chip.

VI. Experimental Results

The ICP was implemented in the C language. 18 large test cases were selected from ITC'99 [10] benchmark suites. The test cases were mapped to the 0.18μm, 5-metal layer standard cell library provided by Virtual-Silicon Technology Inc. [11]. Table 1 summarizes the statistics of the benchmark suite. All the experiments were performed on Sun UltraSPARC machines with the 1.5GHz CPU and 1GB memory.

The results were compared with the leading-edge industry placer, the Cadence AMOEBA placer (SOC Encounter v4.1, May 2005) in the timing-driven placement mode. The initial floorplans were produced by the SOC Encounter with utilization factor 0.85 and the same floorplans were applied to the AMOEBA and to the ICP for each test case. All placement results were sent to the Cadence *WRoute* for routing. Every wire length reported in the following experiments is the routed wire length produced by the WRoute. Every placement produced by the AMOEBA and the ICP was successfully routed by the WRoute without any violations. Timing analysis and power estimation were performed by the Cadence SOC Encounter.

TABLE I
Benchmark Statistics.

Ckt	#Gates	#Nodes	#Rows
b05	961	974	25
b12	1065	1072	30
b14_1	6814	6848	67
b15	8816	8854	80
b14	10012	10070	81
b15_1	12992	13032	93
b20_1	14389	14425	98
b21_1	14388	14470	98
b20	20172	20208	115
b21	20517	20609	116
b22_1	21718	21816	121
b22	29897	30005	140
b17	32192	32511	150
b17_1	39531	40128	163
b18_1	108423	109629	268
b18	114562	115780	275
b19_1	219373	221769	381
b19	231269	233685	391

Table 2 summarizes the experimental results. Columns 2-4 show the critical path delays produced by each placer and their comparison. As data show, the ICP produces solutions with 23% better timing than the timing-driven mode of the AMOEBA on the average.

The experimental results on power consumption are presented in columns 5-7. Because the power consumption of a circuit is proportional to the clock speed, circuits with the shorter delay consume more power. For the fair comparisons, the same clock speeds, which were defined by the AMOEBA solutions, are used in comparison of power estimations for both solutions in the same test case. The power consumption of solutions produced by the ICP is 14% smaller than solutions from the AMOEBA on the average.

In Table 2, columns 8-10 show data on wire length. The ICP produces as good wire length as the AMOEBA does, i.e. improvement in timing by the ICP is achieved without increase in wire length.

Run times are summarized in columns 11-13 of Table 2. As can be seen, the ICP is 2.19 times faster than the AMOEBA on the average.

TABLE II

EXPERIMENTAL RESULTS.

Ckt	Delay (ns)			Power (mw)			Wire Length (μm)			Run Time (sec)		
	AM	ICP	$\frac{ICP}{AM}$	AM	ICP	$\frac{ICP}{AM}$	AM	ICP	$\frac{ICP}{AM}$	AM	ICP	SpeedUp
b05	1.375	1.265	0.92	14.03	12.87	0.92	23188	23090	1.00	2	2	1.00
b12	1.167	0.998	0.86	17.70	16.45	0.93	30869	30343	0.98	2	2	1.00
b14_1	3.472	3.033	0.87	47.34	38.63	0.82	218774	229242	1.05	24	12	2.00
b15	5.093	4.407	0.87	39.84	36.34	0.91	368319	380230	1.03	43	19	2.26
b14	4.489	3.373	0.75	52.96	41.27	0.78	325605	308941	0.95	38	23	1.65
b15_1	3.514	3.260	0.93	78.98	70.05	0.89	454143	451851	0.99	45	24	1.88
b20_1	6.917	3.374	0.49	48.15	40.78	0.85	480349	454253	0.95	59	32	1.84
b21_1	5.452	3.525	0.65	61.49	50.69	0.82	479643	466506	0.97	60	31	1.94
b20	7.324	4.561	0.62	62.08	53.36	0.86	649549	630226	0.97	103	43	2.40
b21	6.313	4.794	0.76	69.78	60.89	0.87	662127	637840	0.96	97	45	2.16
b22_1	7.891	6.329	0.80	65.01	54.27	0.83	752695	731909	0.97	102	48	2.13
b22	7.379	5.357	0.73	104.38	85.03	0.81	993037	1000694	1.01	181	68	2.66
b17	7.972	5.650	0.71	93.75	83.54	0.89	1365154	1345432	0.99	235	82	2.87
b17_1	5.500	3.752	0.68	158.84	134.38	0.85	1469435	1469378	1.00	379	194	1.95
b18_1	5.523	4.681	0.85	437.14	379.01	0.87	3746058	3686867	0.98	1773	471	3.76
b18	6.125	4.777	0.78	409.10	357.91	0.87	4104256	3784747	0.92	1822	492	3.70
b19_1	6.977	5.484	0.79	666.67	584.28	0.88	7538895	7612921	1.01	2812	1328	2.12
b19	6.319	5.613	0.89	739.48	661.28	0.89	7888017	8108235	1.03	2848	1402	2.03
Avg.			0.77			0.86			0.99			2.19

TABLE III

EXPERIMENTAL RESULTS WITHOUT GATE SIZING.

	Delay	Power	Wire Length	SpeedUp
Avg.	0.91	0.95	0.98	2.48

Table 3 shows the experimental results with the gate-sizing option of the ICP disabled. Compared with the timing-driven mode of the AMOEBA, the ICP without gate sizing produces solutions with 9% better timing, 5% smaller power consumption, and 2% shorter wire length on the average. The ICP without gate sizing is 2.48 times faster than the AMOEBA. These results show that the gain in performance of the ICP over the AMOEBA depends only partially on the gate sizing.

VII. CONCLUSION

We have presented a fast iterative-constructive standard cell placer, the ICP, which can integrate gate sizing in the placement step. The ICP is implemented as a dynamically adaptable structure capable of adjusting parameters of the constructive placement procedure. This type of structure allows easy modifications and has flexibility in emphasizing selected goals. When compared with the timing-driven AMOEBA, the ICP produces solutions with the average improvement by 23% in timing, 14% in power dissipation and 2.19 times faster run time.

REFERENCES

[1] J. Cong, T. Kong, J. Shinnerl, M. Xie, and X. Yuan, "Large Scale Circuit Placement," In *ACM Transaction on Design Automation of Electronic Systems*, Vol. 10, No. 2, pp. 389-430, 2005.

[2] B. Halpin and C. Chen and N. Sehgal, "Timing Driven Placement using Physical Net Constraints," In Proc. of *Design Automation Conference*, pp. 780-783, 2001.

[3] X. Yang, B. Choi, and M. Sarrafzadeh, "Timing-Driven Placement using Design Hierarchy Guided Constraint Generation," In Proc. of *International Conference on Computer-Aided Design*, pp. 177-180, 2002.

[4] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of Performance Constraints for Layout," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 8, No. 8, pp. 860-874, 1989.

[5] H. Youssef, R. Lin, and E. Shragowitz, "Bounds on Net Delays for VLSI Circuits," In *IEEE Transactions on Circuits and Systems*, Vol. 39, No. 11, pp. 815-824, 1992.

[6] J. Y. Sayah et. al., "Design planning for high-performance ASICs", In *IBM Journal of Research and Development*, Vol. 40, No. 4, pp. 431-452, 1996.

[7] 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 Proc. of *International Symposium on Physical Design*, pp. 155-160, 2002.

[8] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Design," In Proc. of *Design Automation Conference*, pp. 526-529, 1997.

[9] A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements?," In Proc. of *Design Automation Conference*, pp. 477-482, 2000.

[10] ITC99 Benchmarks, http://www.cerc.utexas.edu/itc99-benchmarks/bench.html.

[11] Virtual-Silicon Technology Inc., http://www.virtual-silicon.com.