PDL: A New Physical Synthesis Methodology

Toshiyuki Shibuya
Fujitsu Laboratories LTD.
Kawasaki, Japan
shibu@jp.fujitsu.com

Rajeev Murgai
Fujitsu Laboratories of America, Inc.
Sunnyvale, USA
murgai@fla.fujitsu.com

Tadashi Konno
Fujitsu LTD.
Kawasaki, Japan
konno.tadashi@jp.fujitsu.com

Kazuhiro Emi
Fujitsu LSI Technology LTD.
Kawasaki, Japan
emi@flt.fujitsu.co.jp

Kaoru Kawamura
Fujitsu Laboratories LTD.
Kawasaki, Japan
kawamura@jp.fujitsu.com

Abstract

In this paper, we propose a new physical synthesis methodology, PDL, which relaxes the timing constraints to obtain best optimality in terms of layout quality and timing quality. It provides a common database for delay calculation, logic optimization, placement, and routing tools so that they can work and interact closely. We present results on industrial circuits showing the efficacy of this methodology.

1. Introduction

With the advent of deep sub-micron technologies, interconnect delay is becoming dominant. Paths traveling across the chip or going back and forth, called global paths, limit the circuit performance.

Many timing-driven techniques have been proposed for logic synthesis, placement, and routing. However, the traditional design flow of iteratively performing logic synthesis, placement, and routing runs into serious timing convergence problems. Although each tool has its own role in timing optimization, neither do they share a common strategy nor have any interaction during optimization. For example, logic synthesis tools can compute gate delay accurately, but not interconnect delay because it is estimated using statistical wire load models. Placement tools can place cells lying on a critical path close to each other in order to reduce the lengths and delays of wires connecting them after routing. But the interconnect delay estimated by the placement tool may differ significantly from that of the final layout. If buffer insertion capability is available during placement, placement tools do not have to put such cells adjacent to each other, since inserting a single buffer could improve the timing. Finally, routing tools can minimize interconnect delay either by using minimum-length wires from the source to critical sinks or by constructing a tree topology that isolates non-critical sinks. However, the minimum distance between the source and a critical sink can be improved if cells are allowed to move during the routing phase. ECO tools are expected to fix timing errors by inserting buffers and re-routing nets, but they have to optimize timing without significantly modifying the given layout.

To overcome the aforementioned problems inherent in the traditional flow, several approaches have been proposed. These can be divided into three broad categories: pre-layout estimation, unification-based, and pre-routing optimization.

The pre-layout estimation approach attempts to estimate the impact of layout without actually performing it and then uses these estimates to guide the optimization at pre-layout stage [11][15]. One problem with this class of methods is that they do not actually work on a real layout. They predict or derive an intermediate layout and generate a netlist accordingly. The actual layout generated by the layout tool may be entirely different.

The unification-based approach tries to perform a group of design and optimization steps simultaneously rather than sequentially [6][10]. Although these algorithms are promising in concept, they guarantee optimality on very restricted netlist topologies. Also, their applicability to industrial-strength designs remains to be seen.

Finally, the pre-routing optimization approach tries to push certain optimization steps (such as gate resizing, buffering, resynthesis and remapping) to later stages of the design flow where more authentic physical information is available [3][12]. The main drawback of this approach is that the final routing information is still not available during optimization.

In this paper, we present a methodology, PDL, to handle the timing problems on practical ASIC designs. Of the three categories of previous work, PDL methodology is closest to the pre-routing approach, although with significant differences, as described below. The PDL methodology addresses new physical synthesis optimization problems with timing constraints.

The rest of the paper is organized as follows. In Section 2 we address the problems of current design flows. Section 3 discusses the PDL methodology. System overview and implementation issues are described in Section 4. Section 5 presents experimental results of the proposed methodology. We conclude with some directions for future work in Section 6.

2. Problems of Current Timing-driven Physical Synthesis Flows

A typical physical synthesis methodology consists of tool elements related with layout, such as placer, router, and logic optimizer. The goal of timing-driven physical synthesis is to complete the circuit layout without violating the timing constraints. One important issue left unaddressed previously is whether the timing constraints should be deemed as absolute (i.e., must be satisfied at all costs) or not.

To complete a design considering timing constraints as absolute, the shape of the die should be flexible. This is because there is no guarantee that the design can be completed under the absolute timing constraints with the given die size. This means the die size may shrink or enlarge depending on the given timing constraints.

Generally the die size of ASIC designs is determined before the layout phase. The sizes of RAMs and I/O macros dominate the chip size. Once they are placed on the chip, designers try to complete the layout without changing the die size. The difficulty in this situation is that the layout may result in too many routing errors and timing errors.

The PDL methodology we are proposing in this paper gives a clear direction as to what to do for ASIC designs with the given die size.

3. PDL Methodology

The layout phase in PDL methodology consists of global layout and detail layout. Global layout optimizes cell positions using *bi-partitioning-based placer*. Bi-partitioning-based placer determines cell positions on a given layout area by hierarchically dividing the circuit into two sub-circuits and assigning them to the divided layout areas called *blocks*. At each *level of partitioning*, the global router generates a loose route assigning blocks to nets. The logic optimizer performs gate sizing, buffer optimization, and re-synthesis, interacting with the placer and router. Detail layout determines the exact cell positions and completes the detail routing.

3.1 Policies

The PDL methodology is based on the following policies:

1) *Routability and wire length should be the top priorities*. Route completion is very important for designs in deep sub-micron (DSM) technologies, because the main impact of such technologies, such as interconnect delay and cross-talk noise, can be measured only after the final routing stage. The delay and noise measurements are meaningless if there are routing errors.

The completed layout even with timing errors can provide useful information to designers, such as critical paths, noise-sensitive nets, accuracy of timing constraints, etc. When the layout results in routing errors, with routability and wire length being the top priorities, the designers should check if the floorplan (including macro placement), cell usage ratio, and routing resources used for power routing are appropriate for the technology and design.

2) *Only the global critical paths should be optimized at each level*. The placement and routing tools optimize the true critical paths in the design, called the *critical limit paths*. The tools decide that these paths cannot be fixed by other tools either at that level or at subsequent levels. However, if fixing a critical limit path degrades routability or wire length, it is discarded.

Since the PDL methodology fixes global critical paths hierarchically, local critical paths may be left unfixed in final layout. However, they can be easily fixed during post-layout optimization or clock skew adjustment.

If global critical paths are left in the final layout, they should be carefully analyzed. In our methodology, this can happen under three scenarios. First, making the path shorter might cause routability problems. The timing constraints should be checked, e.g., whether a timing over-constraint or dropping false path specification exists at the path. Second, the path could not be optimized due to layout constraints. For instance, the path had to go around several floorplan blocks, or a wire was routed through a no-buffering-resource region. Third, design constraints, such as an option specifying the logical hierarchy to be unchanged, prevent it from being touched during optimization.

3) *Timing constraints for non-critical limit paths should be relaxed for better routability and wire length*. The most critical problem physical synthesis optimization should avoid is the negative feedback optimization. Sub-optimality in routability and wire length induces detours and increased wire length, resulting in new critical paths. Such critical paths may be improved through buffer insertion, for instance. Buffer insertion without any regards to routability may cause other nets to detour around the buffers. This, in turn, may generate new critical paths, yielding a negative feedback in routability and timing.

Net-based timing-driven placement algorithms [4][14] control the delay of a path either by imposing maximum separation between the cells on the path or by adding weights to the nets on the path. However, the layout constraints on cells and the weights on nets deteriorate the quality of placement in terms of wire length and routability. This is because these constraints, by restricting the solution space, can prevent the tool from reaching the optimum solution.

Another example is the case of too many critical paths limiting the possibility of tree topology optimization in routing. This over-constraint restricts the solution space for finding the optimal path in terms of routability. Optimality in wire length is to be sought even in the case of timing-driven layout [2].

3.2 Approach

Our approach to make the logic optimizer, placer, and router work together has the following key cornerstones.

1) *Critical path identification*: One of the most important issues is estimating the interconnect delay. However, it is difficult to estimate this delay accurately in hierarchical global routing. Because locations of cells are not determined yet, the wire length for each wire segment may change depending on the actual cell position within a block.

We think accuracy in the interconnect delay is not essential during initial levels of partitioning. A consistent delay model for optimization is more critical. Our approach considers that a cell, if not pre-placed, is placed at the center of the block. This means the inter-block wire length is estimated from the center of a block to that of the other. The intra-block wire length, corresponding to the wire connecting cells within a block, is estimated as the minimum distance between two cells, e.g., the width of cells or zero.

The main advantage of this model is that when a net that connects cells placed in the same block is partitioned, the wire length and delay will apparently increase. This gives the placer a consistent direction for delay optimization. Another way to estimate wire length is to predict the cell position within the block area. Although this seems accurate at the level of hierarchy, it lacks a consistent direction for optimization. This is because an intra-block wire after partitioning may become longer than an inter-block wire.

The disadvantage is that the delay value is not accurate at higher levels of hierarchy. However, as the partitioning process proceeds, the margin of error becomes smaller and smaller. We expect the margin at the final levels is much smaller than the uncertainty of detail layout.

The second important issue is that the correctness of timing constraint specification should be carefully discussed to determine the *critical limit paths*. Designs may have multiple timing modes, such as normal mode and test modes. Especially for signal processing circuits handling several standards, such as NTSC and PAL for video chips, several clock modes are specified for the same circuit. This is modeled by multiple timing constraint sets, usually one in each file. This is not a problem for static timing analysis (STA), which can analyze each file independently and eventually determine criticality over all files. However, multiple, separated timing constraint files may cause problems in timing-driven layout, since the layout process usually considers only one constraint set. The layout thus obtained may violate the constraints in other sets. Extra iterations are required to fix the timing. One way to avoid such iterations is to manually merge all the constraint files into one, without causing any timing constraint conflicts. This, however, is prone to human errors. In any case, multiple timing constraint sets (files) should be handled efficiently by the PDL methodology.

The third important issue is that the delay models should properly handle operating conditions, such as process, temperature, and voltage, both for cell delay and interconnect delay. Otherwise, extra timing margin may be needed on the critical paths, which makes achieving the timing closure more difficult.

2) *Risk estimation of critical paths*: As per the consistent interconnect delay model discussed above, when an intra-block wire is partitioned at a level and becomes inter-block at the next level, its wire length or delay increases by the width or height of the block.

The placer checks whether a net will induce a *critical limit path* when the net is partitioned based on the consistent delay model. If the net does induce a *critical limit path*, which means increased interconnect delay does not improve by buffer insertion, the placer tries to place the cells on the net in the same block and renders them fixed.

3) *Timing relaxation*: As mentioned in the PDL methodology, existence of too many critical paths imposes cell partitioning and routing space restrictions, degrading optimality in terms of routability and wire length. Therefore, at each level of partitioning, logic optimizer not only tries to maximize the worst negative slack and the sum of negative slacks, but also attempts to reduce the total cell size to improve routability and increase the possibility for buffer insertion. The placer calculates an upper bound of cell usage ratio that guarantees routability. The logic optimizer improves timing under this upper bound. Once the cell usage ratio of a region reaches the upper bound, timing improvement by logic optimizer may not occur unless cell usage improvement takes place. In our methodology, the placer adjusts the local cell usage ratio by moving the cells on the non-critical paths from an

over-congested block to other blocks, thus increasing routability and possibility of logic re-synthesis.

4. PDL System

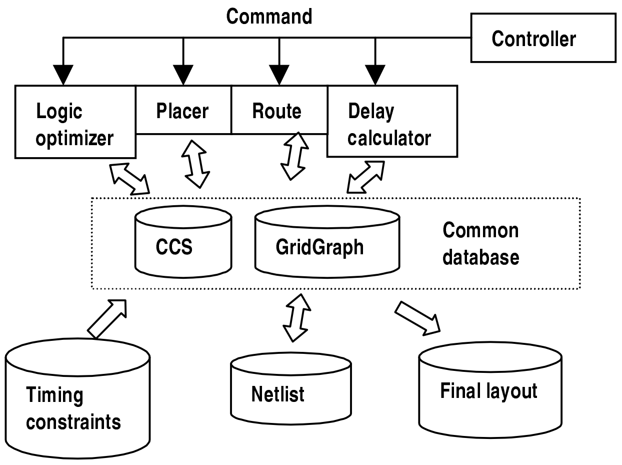

Figure 1: PDL System

4.1 Overview

The PDL system consists of a controller, delay calculator, logic optimizer [7] [8] [9] [13], placer [1], and router [5] (Fig. 1). Each tool is executed as a single process so that large circuits can be handled using multiple CPUs.

The layout information is shared among the tools by exchanging an abstract layout model, called *GridGraph data*. The GridGraph data expresses a partitioned block as a node and a routing resource between adjacent blocks as an edge (Fig. 2). The number of nodes doubles at each level of partitioning. The locations of cells are specified as the corresponding nodes. Any tool can plug-in to the system and modify layout or netlist as long as it obeys the GridGraph updating rules. We now describe each component of the PDL system in detail.

4.2 Controller

The controller reads the netlist and designer's options. Then it translates them into data that each tool can use as a common database (Fig. 1). At each level of partitioning, the controller may issue commands for any tool as per the flow specified by the designers. Finally, it outputs the updated netlist and layout results.

The controller can store the entire GridGraph data at each level if so desired by the designers. This enables designers to resume the layout process from any partitioning level. This function is useful when designers want to modify (manually or automatically) the layout result at an intermediate level and use it as the starting point for the next level to obtain higher chip performance.

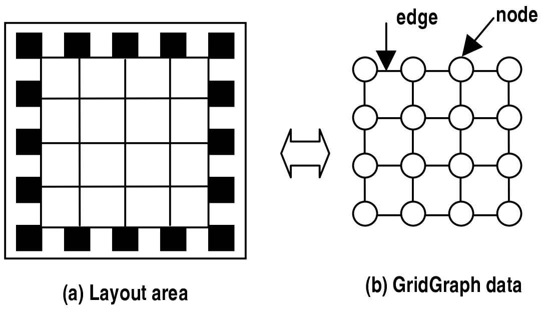

Figure 2: GridGraph representation of layout area

4.3 Delay Calculator

Timing-driven layout tools invoke delay calculation several times. Therefore, a fast delay calculator is required. Most timing sign-off tools adopt a path-based static timing analysis method. This is time-consuming, because delay of every path may need to be reported to designers. However, information about the most critical paths is sufficient for the delay calculators in timing-driven layout. The PDL system uses topology-based STA to calculate delay quickly.

To solve the multiple timing constraint file problem, we propose a new timing constraint model, *Compatible Constraint Set (CCS)*. This enables the PDL system to handle all timing constraints at the same time. CCS is a set of flip-flops (FFs) and paths on which actual arrival time and required arrival time can be specified without any timing constraint conflicts. The delay of FF pairs in a CCS can be calculated independently of the other CCSs. The delay of a set of CCSs can be calculated at the same time if each set of FF pairs in a CCS has no intersection. In the PDL system, all the timing constraint files are transferred into a set of CCSs, which is then used for delay calculation.

The increase in timing exceptions, especially multi-cycle, and multiple clocks make topological-based STA complicated, because a FF now has more than one set of actual arrival time and required arrival time. The analysis of CCSs enables designers to determine which constraints will critically impact the timing quality.

Gate and interconnect delays are computed by using the same function libraries that are used for the in-house timing sign-off tool. PDL system accepts the same timing constraint files that are used by the timing sign-off tool.

4.4 Logic Optimization

The logic optimizer in the PDL system fixes the timing gap caused by the delay model differences between the wire load and global routing patterns. It also reduces layout constraints for the placer and router to obtain best performance in routability and wire length. The requirement for logic optimizer is to cope with physical information, layout constraints, and increasing circuit size.

The logic optimizer calculates interconnect delay using the common delay calculator and routing patterns, which are shared through GridGraph data. It maximizes the worst negative slack and the total negative slack under the cell usage ratio constraints, as mentioned in "*Timing relaxation,*'' Section 3. It is also important to handle slew constraints and maximum signal transition times in order to prevent nets from cross-talk noise.

Circuit designs today may contain tens of millions of gates. The logic optimizer has to handle them in reasonable CPU time and memory. We adopted an iterative optimization method that applies local transformations, such as buffer insertion [9], gate resizing, and re-synthesis [13] [14], repeatedly.

4.5 Placer

The objective of the placer in the PDL system is to minimize the number of cut nets among partitioned sub-circuits. This is achieved using SNR [1] approach. There are two drawbacks of the partitioning-based approach. First, top-down optimization such as partitioning solves global problems quite efficiently. However, local problems may exist within the partitioned blocks, because the top-down decision may not always be optimum for the local problem. For example, the partitioning-based placement yields a smaller wire length, but the existence of locally congested area deteriorates routability. Second, min-cut is not suitable for timing optimization. As mentioned above, imposing maximum distance or adding a weight on the critical net may limit the solution space considerably.

The problem of the existence of locally congested area is solved with refinement process in the PDL system. At latter levels of partitioning, the router provides accurate routing information including routing congested block information. The placer moves non-critical cells from the highly congested blocks to the less congested blocks during the refinement process. The critical cells are locked so as not to degrade the timing.

As discussed in "*Risk estimation of critical paths,*'' Section 3, the cells on a critical net will be placed in the same block if the placer predicts that buffer insertion cannot eliminate the critical net. The difficulty in this approach is that since nets belonging to a critical path share cells, too many cells may need to be placed in the same block. This may be beyond the area resources available in that block. We are then forced to partition the adjacent critical nets. Therefore, we have to carefully analyze which cells on critical nets should be placed in the same block. We use critical path mapping technique to identify them. This technique assigns weights to the critical nets according to their criticality and then determines the optimal positions of cells connected to these nets in a block using quadratic placement. If all the cells of a critical net are assigned to one block by the quadratic placement algorithm, they are fixed to that block; all other cells can be freely optimized.

4.6 Router

The main objectives of the router are to estimate routability and routing patterns hierarchically during partitioning. The remaining tasks are handled by other tools. For instance, the placer determines net lengths and logic optimizer handles fanouts. However, three issues should be considered in the router of the PDL system.

First, at any level the router should generate routing patterns that are consistent with those at the previous level. The router explores the solution space using the patterns at the previous level. This improves the run time of the router. Figure 3 shows an example of routing pattern transformation of a net. The correlation between routing patterns at the current level and the previous one gives the placer a consistent optimization direction for terminal propagation and the logic optimizer, a consistent

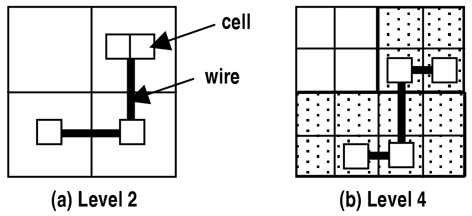

Figure 3: Routing pattern transformation

buffer insertion strategy. To achieve the correlation between global and detail routing patterns, detail router also uses the global routing patterns.

Second, even though the solution space is limited by the routing patterns at the previous level, detours may happen at a congested block. We prioritize the nets on critical paths to have them routed in shortest distance.

Finally, the previous two constraints limit the solution space for the router, which influences the routability and wire length. The router in the PDL system ignores the constraints and expands the solution space to find better routing patterns.

Table 1: Circuit data characteristics

	clock rate	#cells	#BC	technology
CD1	75MHz	31,963	313,702	0.35um
CD2	266MHz	32,908	265,144	0.25um
CD3	100MHz	156,151	787,624	0.25um
CD4	258MHz	135,704	1,006,507	0.25um
CD5	125MHz	367,388	2,394,037	0.18um

5. Experimental Results

In this section, we present experimental results to show the effectiveness of the PDL methodology. Five practical circuit designs, CD1 through CD5, are used. Their characteristics are given in Table 1. The circuit sizes shown in the table are extracted before layout. One BC is the size of the smallest inverter in the technology.

In our experiments, first we show the timing and layout qualities. Then, the timing convergence behavior of the PDL system is discussed.

5.1 Timing and Layout Qualities

Table 2 compares the results of the PDL system with those of a conventional system, which is the PDL system without its timing-driven features. To evaluate routability and wire length, the detail router is executed at the end. The number of routing violations is the number of nets that could not be routed with a standard detail routing process. The number of setup errors, the minimum slack value, and total sum of negative slack values are analyzed before detail routing using an in-house timing sign-off tool. This evaluates the quality of the PDL system without any consideration for other factors that influence the delay, such as clock skew and cross-talk.

The results show that the sum of negative slacks improved in all the circuits without any deterioration in the layout quality. Critical paths still remain in four circuits. After analyzing the timing reports, we discovered that buffers inserted manually to fix the hold errors cause timing errors in the paths between different clock domains. In these experiments, clock skew is set to zero to remove the influence of the quality of clock tree optimization. We confirmed that most of the timing errors could be fixed by adding the actual clock phase delay to each clock domain source. Table 3 shows the change in the number of cells and area penalty during the process. Both values are small, sometimes even negative.

Table 2: Comparison of conventional system and PDL system

	convetional system					PDL system				
	wire length [m]	#routing violations	#setup errors	minimum slack [ps]	total negative slack [ns]	wire length [m]	#routing violations	#setup errors	minimum slack [ps]	total negative slack [ns]
CD1	13.6	282	194	-3,117	-157.8	13.3	497	0	0	0.0
CD2	11.5	92	2	-1,982	-2.4	11.6	101	2	-1,652	-2.1
CD3	76.7	7	17	-550	-3.4	86.0	0	0	0	0.0
CD4	61.8	63	4,274	-2,150	-2,615.5	62.4	184	1,670	-3,170	-335.3
CD5	129.5	36,411	21,334	-15,921	-27,501.9	128.1	546	330	-1,278	-93.5

5.2 Timing Convergence in PDL System

To show how timing converges in the PDL system and logic optimizer improves the criticality of global paths, the minimum slack and the total sum of negative slacks of CD5 are plotted at each partitioning level in Figure 4. Logic optimization is performed at every even level.

Although the placer and router try to improve timing, the minimum slack value becomes worse when partitioning moves to the next even levels. Newly generated inter-block wires with partitioning cause this degradation. However, logic optimizer improves the slack value efficiently, especially at levels four and eight.

The logic optimizer also improves the total sum of the negative slacks. Interestingly, it does not become worse with partitioning. This means it is less sensitive to the newly generated inter-block wires than the minimum slack. This is because dividing an intra-block wire into half segments and partitioning cells optimally might offset the impact of increased inter-block wires.

6. Conclusions and Future Plan

In this paper, we have proposed PDL, a new physical synthesis methodology. The approach is to identify critical limit paths, estimate their risk, and relax timing constraints of non-critical paths in order to obtain the best routability

Table 3: Increased area in PDL system

	increased #cells	increased #BC	BC increased ratio [%]
CD1	146	644	0.205
CD2	-1	-5	-0.002
CD3	9	-2,885	-0.366
CD4	-920	-3,478	-0.346
CD5	1,131	3,454	0.144

and wire-length layout result under the given timing

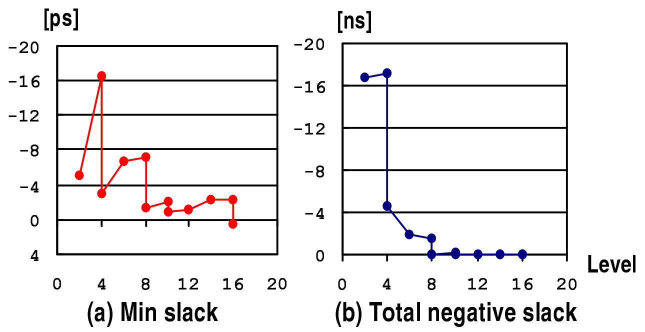

Figure 4: Slack value convergence (CD5)

constraints. The experiments show that the total negative slack value is significantly improved. The future work is in the following directions. We need to improve the number of routing violations. Also, clock tree synthesis is an important problem for higher performance. We plan to integrate it in our methodology.

7. References

[1] J. Cong, H. P. Li, S. K. Lim, T. Shibuya, and D. Xu, "Large Scale Circuit Partitioning with Loose/Stable Net Removal and Signal Flow Based Clustering", *ICCAD*, San Jose USA, November 1997, pp. 441-447.

[2] J. Cong and S. K. Lim, "Performance Driven Multiway Partitioning", *ASPDAC*, Yokohama Japan, January 2000, pp. 441-446.

[3] L. N. Kannan, P. R. Suaris, H. G. Fang, "A Methodology and Algorithms for Post-Placement Delay Optimization", *DAC*, June 1994, pp. 327-332.

[4] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel, "Chip Layout Optimization Using Critical Path Weighting", *DAC*, Los Angeles USA, June 1984, pp. 133-136.

[5] K. Kawamura, T. Shindo, T. Shibuya, H. Miwatari, and Y. Ohki, "Touch and Cross Router", *ICCAD*, Santa Clara USA, November 1990, pp. 56-61.

[6] J. Lou, A. H. Salek, and M. Pedram, "An Exact Solution to Simultaneous Technology Mapping and Linear Placement Problem", *ICCAD*, San Jose USA, November 1997, pp. 671-675.

[7] R. Murgai, "Performance Optimization Under Rise & Fall Parameters", *ICCAD*, San Jose, 1999, pp. 185-190.

[8] R. Murgai, "On the Global Fanout Optimization Problem", *ICCAD*, San Jose, Nov 1999, pp. 511-515.

[9] R. Murgai, "Layout-driven Area-constrained Timing Optimization by Net Buffering", *ICCAD*, San Jose USA, November 2000, pp. 379-386.

[10] T. Okamoto, and J. Cong, "Interconnect Layout Optimization by Simultaneous Steiner Tree Construction and Buffer Insertion", *ACM/SIGDA Physical Design Workshop*, Reston USA, April 1996, pp. 1-6.

[11] M. Pedram, and N. Bhat, "Layout Driven Logic Restructuring/Decomposition", *ICCAD*, Santa Clara USA, November 1991, pp. 134-137.

[12] G. Stenz, B. M. Reiss, B. Rohfleisch, and F. M. Johannes, "Timing Driven Placement in Interaction with Netlist Transformation", *ISPD*, April 1997, pp. 36-41.

[13] Y. Tamiya, "Robust Performance Optimization Using Padding Nodes and Separator Sets", *IEICE Transactions on Fundamentals*, vol. E84-A, No.11, November 2001, pp.2739-2745.

[14] R. S, Tsay and J. Koehl, "An Analytic Net Weighting Approach for Performance Optimization", *DAC*, San Francisco USA, Jun 1991, pp. 636-639.

[15] H. Vaishnav and M. Pedram, "Routability-Driven Fanout Optimization", *DAC*, Dallas, 1993, pp. 230-235.

7. Acknowledgements

The authors thank other members of this project who contributed to this paper: Yuzi Kanazawa, Tatsuya Nakae, Izumi Nitta, Hidetoshi Matsuoka, Yutaka Tamiya, Naomi Tashiro, and Yoshinori Tomita.