Performance-Driven Soft-Macro Clustering and Placement by Preserving HDL Design Hierarchy

Hsiao-Pin Su, Allen C.-H. Wu, and Youn-Long Lin
Department of Computer Science, Tsing Hua University
Hsinchu, Taiwan, 300, Republic of China

Abstract

In this paper, we present a performance-driven soft-macro clustering and placement method which preserves HDL design hierarchy to guide the soft-macro placement process. We also present a complete chip design methodology by integrating the proposed method and a set of commercial EDA tools. Experiments on three industrial designs ranging from 75K to 230K gates demonstrate that the proposed soft-macro clustering and placement method improves critical-path delay on an average of 24%.

1 Introduction

Over the past decades, academia and industry have invested vast effort in physical design related research, including floorplanning, partitioning, placement, and routing. Several excellent reviews of physical design techniques are given by [1, 2, 3, 4]. By integrating various techniques, many design methods and software systems have been developed for chip designs. One of the most popular design methods uses schematics as the design entry, followed by floorplanning, placement, and routing to produce final chip layouts. This design method is very effective and efficient on small to medium-scaled designs. However, with the advent of deep-submicron technology, more and more devices can be packed into a very complex single chip. Due to the time-to-market pressure of designing complex chips and the maturity of synthesis tools, more and more integrated-circuit designers use an HDL-based synthesis approach to develop and manage large designs. Furthermore, as devices geometries shrink, a new set of design challenges, especially in electrical characteristics of circuits, are faced by integrated-circuit designers. This has led to a new research direction in design automation at synthesis and physical levels. Recently, several papers [5, 6, 7, 8] have addressed the challenges and considerations in physical designs targeted to deep-submicron processes.

This work was supported by the National Science Council of R.O.C. under Grant NSC 87-2215-E-007-011 and NSC 87-2215-E-007-017.

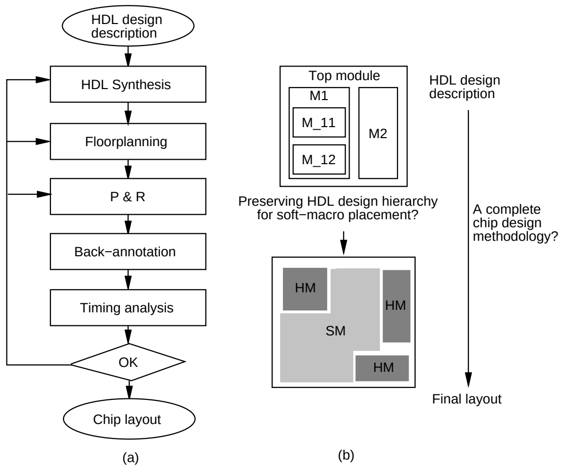

Figure 1: A typical HDL-based method for macro-based designs: (a) the design flow, (b) problem description.

In this paper, we propose a soft-macro clustering and placement technique which preserves the HDL design hierarchy to guide the soft-macro placement process. We present a complete design methodology by incorporating the proposed soft-macro clustering and placement technique for high-performance chip designs. Experiments on a number of industrial designs have been conducted to demonstrate the effectiveness of the proposed method.

The rest of the paper is organized as follows. Section 2 describes the problem. Section 3 presents the proposed design flow. Section 4 gives experimental results. Finally, Section 5 draws concluding remarks.

2 Problem description

Figure 1(a) shows a typical HDL-based chip design flow. It consists of five steps: (1) HDL synthesis, (2) floorplanning, (3) place and route, (4) back annotation, and (5) post-layout timing analysis. The input to the design flow is a mixed RTL and gate-level HDL description in Verilog or VHDL. The HDL-based design specification is usually described as a set of hierarchical modules containing hard macros (e.g., predefined blocks with fixed size and IO-pin location) and soft macros (e.g., can be implemented with a flexible layout style such as standard-cell style).

In the first step, a synthesizer converts an HDL design description into a hierarchical gate-level netlist by performing HDL compilation and a series of RTL and logic synthesis tasks. In the second step, a floorplanning procedure is invoked to determine the location of each macro on the layout plane. In the third step, a placement-and-routing procedure is used to perform detailed gate-level placement and routing. In the fourth step, the circuit parasitic information is extracted. Finally, a post-layout timing analysis procedure is performed to determine the most critical paths and their delays. If the timing does not satisfy the design requirement, a refinement iteration will proceed until the timing requirement is satisfied. The refinement procedure can be applied at different design levels. For instance, we can re-synthesize certain modules or insert drivers along the critical paths to speed up the circuit timing. We can also adjust the floorplan or re-run a performance-driven floorplanning procedure guiding by the post-layout timing information. Furthermore, we can adjust the soft-macro placement or re-run the detailed placement and routing procedure.

In general, an HDL-based design flow involves multi-level design tasks. Over the years, vast effort has been invested to improve the quality of design task at each design level. Very few studies have been conducted to investigate the interaction between different design tasks. This motivates us to investigate how to develop a complete chip design methodology by integrating multi-level design tasks and exploiting the interaction between them.

In this study, we focus on developing a complete design methodology which incorporates a soft-macro clustering and placement technique by preserving HDL design hierarchy. The main objectives of this research are twofold, as depicted in Figure 1(b). The first one is how to utilize design structural hierarchy to guide soft-macro placement. Several recent studies [9, 10, 11, 12, 13, 14, 15] have demonstrated that considering the circuit structural properties during the placement process can improve the placement result. In this study, we investigate how to preserve HDL design hierarchy to improve the quality of soft-macro placement. The second objective is to develop a complete design methodology incorporating the proposed soft-macro clustering and placement technique for high-performance chip designs.

3 The proposed design flow

3.1 Overview

Figure 2(a) depicts the proposed design flow which consists of nine steps: (1) HDL synthesis, (2) pre-layout timing analysis, (3) structural-tree construction, (4) soft-macro formation, (5) floorplanning and soft area extraction, (6) soft-macro placement, (7) place and route, (8) back annotation, and (9) post-layout timing analysis. The input to the design flow is a mixed RTL and gate-level HDL description in Verilog. In the first step, an HDL-based synthesizer converts the Verilog design description into a hierarchical gate-level netlist by performing HDL compilation and a series of RTL and logic synthesis tasks. In the second step, a timing analysis procedure is applied to perform unit-delay-based timing analysis of the design. A set of critical paths will be identified and used to guide the following macro-clustering, floorplanning, and placement-and-routing procedures. In the third step, an HDL structural tree of the design is constructed to preserve the design hierarchy. In the fourth step, the system groups small subcircuits to form large macros and decomposes extremely large macros into smaller ones. In the fifth step, we use a commercial floorplanner to perform macro floorplanning to determine the locations of hard macros and then extracts the available area for soft macros. In the sixth step, a soft-macro placement procedure is applied to determine the relative location of each soft macro on the layout plane. In the seventh to ninth steps, we use commercial tools to perform placement, routing, back annotation, and post-layout timing analysis.

Figure 2(b) illustrates an example. In this example, the HDL design description is first converted into a structural tree which contains two hard macros $HM1$ and $HM2$, and five soft leaf-macros $SM1$-$SM5$. During soft-macro formation, four soft macros $SM1$, $SM2$, $SM3$, and $SM4,5$ are formed. After the floorplanning process, a rectilinear area is extracted (dotted-line re-

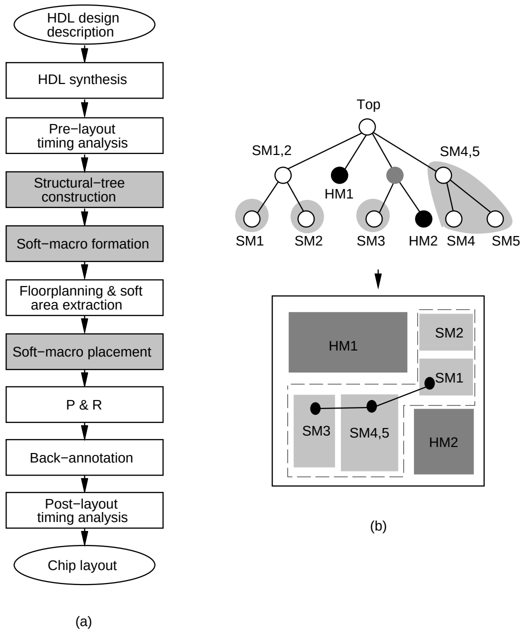

Figure 2: The proposed method: (a) the design flow, (b) an example.

gion in Figure 2(b)) for soft macros. A soft-macro placement procedure is then applied to determine the relative location of each soft macro based on the criticality and the connectivity between macros. After performing the placement and routing procedures, layout parasitic information is back-annotated and post-layout timing analysis is conducted to identify the most critical signal path, as depicted in Figure 2(b).

3.2 Structural-tree construction

The main objective of the structural-tree construction is to preserve the design structural information from an HDL design description for macro formation. Using an HDL-based design flow, the specification of a design is described as a mixed RTL/logic/gate-level description in HDLs such as VHDL and Verilog. An HDL description usually consists of a hierarchy of interconnected modules. During synthesis, each leaf module is synthesized into a gate-level circuit. The final circuit of the design is a composition of all modules.

In our approach, we use a commercial synthesis system to convert a Verilog design description into a hierarchical gate-level netlists. We use an HDL structural tree to represent the structural hierarchy of the Verilog design description. In an HDL structural tree, the root node represents the top design, and each intermediate node represents a module construct. Each leaf node represents a circuit block generated from a leaf module.

3.3 Soft-macro formation

The synthesized subcircuit of each leaf module is naturally a closely-connected cluster. However, a design may also contain extremely large modules containing tens of thousands of gates. This is undesirable in our approach because a large cluster is too rigid for the macro placement and may often result in poor placement results. Furthermore, a design may also contain a large number of small subcircuits. This is also undesirable because when each subcircuit is treated as a soft macro, the computational complexity of the macro-cell placement process will increase and may often result in longer inter-macro wiring.

First, we determine the large-macro candidates which need to be decomposed into smaller ones. The selection of large-macro candidates is based on the size of the macros. We define the threshold value M_{th} of a large-macro candidate as:

$$M_{th} \quad = \quad k * S_{avg},$$

where S_{avg} is the average macro size $\frac{\#Total\ cells}{\#Macros}$, $\#Total\ cells$ and $\#Macros$ are the total number of cells and the number of soft macros in the design, respectively. k is a user-defined threshold parameter for controlling the size of the large-macro. If a macro is larger than M_{th}, then it is selected as a large-macro candidate. For each large macro, we use the partitioning method [14] to decompose large macros into smaller clusters.

Then we use a clustering algorithm [16] to group small macros into large ones based on the size constraint, and the criticality and connectivity between macros. Let $G = \{V, E\}$ be the connected graph where V is the set of macro nodes and E the set of edges. An edge e_{ij} exists if there exists at least a signal flow between macros v_i and v_j. A weight is associated with each edge indicating the number of connections between two corresponding macros. We define the connectivity $Conn_{ij}$, the criticality $Crit_{ij}$, and the closeness C_{ij} of two macros, v_i and v_j, as below.

$$Conn_{ij} = \begin{cases} (\frac{w_i+w_j}{w_i+w_j-2w_{ij}}) \times (\frac{s_i+s_j}{s_{th}}), & \frac{s_i+s_j}{s_{th}} \leq 1; \\ 0, & \frac{s_i+s_j}{s_{th}} > 1, \end{cases} \tag{1}$$

$$Crit_{ij} = \begin{cases} 1, & \textbf{if } Crit\_Path(v_i \Leftrightarrow v_j) \textbf{ and } \frac{s_i+s_j}{s_{th}} \leq 1; \\ 0, & \textbf{else}, \end{cases} \tag{2}$$

$$C_{ij} = \alpha\ Conn_{ij} + \beta\ Crit_{ij}, \tag{3}$$

where
w_i denotes the total connection weight of v_i,
w_{ij} denotes the total connection weight between v_i and v_j,
s_i denotes the size of v_i,
s_{th} is the upper bound on the size of a cluster set by the user,
$Crit\_Path(v_i \Leftrightarrow v_j)$ denotes that there is a critical path traveling across v_i and v_j, and
α and β are two coefficients set by the user.

In order to eliminate small macros and prevent the formation of large clusters, the user can set the upper bound on the size of a cluster. When the size of a new macro formed by merging two macros is larger than the upper bound, the closeness value between these two macros is 0.

3.4 Soft-macro placement

Prior to the soft-macro placement, a floorplanning procedure is invoked to determine the locations of hard macros. Then, the available area for soft macros is extracted, as shown in Figure 3(a) (the dotted area). The main objective of the soft-macro placement is to determine the relative location of each macro on the layout plane. The soft-macro placement consists of two steps: (1) force-directed-based placement and (2) sweeping-based soft-macro assignment. In the first step, we determine the relative location of each soft macro. In the second step, we assign each soft macro into the layout plane.

We divide the layout plane into nine regions, as depicted in Figure 3(a). Initially, each hard-macro is assigned into its corresponding region according to the floorplanning result. Then, we apply the force-directed algorithm [17] to determine the relative location of each soft macro, as shown in Figure 3(b). Let $M = \{m_1, ..., m_n\}$ be a set of hard and soft macros. Let $\Delta x_{ij} = |x_i \Leftrightarrow x_j|$, $\Delta y_{ij} = |y_i \Leftrightarrow y_j|$, and $\Delta d_{ij} = ((\Delta x_{ij})^2 + (\Delta x_{yj})^2))^{1/2}$. Let F_x^i (F_y^i) be the total force enacted upon macro i by all the other macros in the x-direction (y-direction). The force equations can be expressed as:

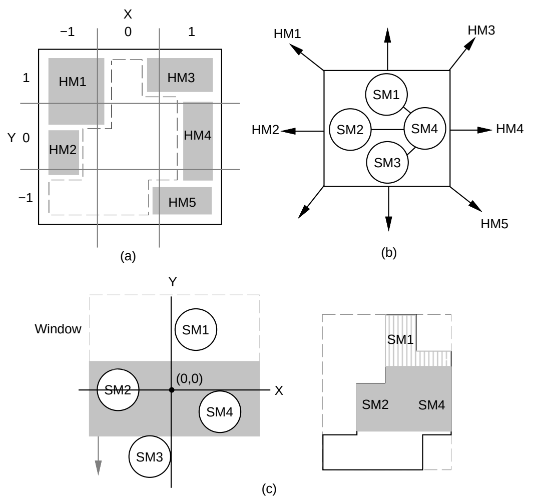

Figure 3: Soft-macro placement: (a) floorplanning and soft-macro area extraction, (b) force-directed-based placement, (c) sweeping-based soft-macro assignment.

$$F_x^i = \sum_{j=1}^{j=n} (\Leftrightarrow w_{ij} \times \Delta x_{ij} + r \times \Delta x_{ij}/\Delta d_{ij}) \Leftrightarrow F_{hm}^x, \quad (4)$$

$$F_y^i = \sum_{j=1}^{j=n} (\Leftrightarrow w_{ij} \times \Delta y_{ij} + r \times \Delta y_{ij}/\Delta d_{ij}) \Leftrightarrow F_{hm}^y, \quad (5)$$

$$F_{hm}^x = [\sum_{j=1}^{j=m} \sum_{j=m+1}^{j=n} (\Leftrightarrow w_{ij} \times \Delta x_{ij} + r \times \Delta x_{ij}/\Delta d_{ij})]/m, \quad (6)$$

$$F_{hm}^y = [\sum_{j=1}^{j=m} \sum_{j=m+1}^{j=n} (\Leftrightarrow w_{ij} \times \Delta y_{ij} + r \times \Delta y_{ij}/\Delta d_{ij})]/m, \quad (7)$$

where m is the number of soft macros. r is the repulsion constant. r is 1 when there is no connection between macros i and j. F_{hm}^x and F_{hm}^y are the force acted upon the set of all soft macros by the hard macros in the x- and y-direction, respectively.

After computing the forces in both x- and y-direction for each soft macro, we can calculate the relative x- and y-coordinate of each soft macro on the layout plane, as shown in Figure 3(c). We then apply a sweeping-based method to assign soft macros into the available layout area, which is described as below. We first use a window which sweeps from top to bottom in the y-direction. We then compute the total required area for the soft macros covered by the window, followed by allocating a region on the available layout area which is large enough to accommodate the soft macros. Since the layout area of each cell can be found in the data book, the total required area for a soft macro is computed as the sum of layout areas of all cells in the soft macro. In our implementation, we use a commercial area router for the detailed routing. According to the vendor's recommendation, we allocate an area of 1.15 times total cell area of a macro to the macro in order to successfully complete the routing. Finally, we assign the soft macros into the allocated region from left to right in the x-direction. The soft-macro assignment procedure continues until all the soft macros are assigned to the layout plane. For example, in Figure 3(c), the sweeping window first covers one soft macro $SM1$ which is assigned into the top region of the layout plane. In the second sweeping, the window covers two soft macros $SM2$ and $SM4$ which will be assigned to the allocated area from left to right. After determining the location of each soft macro, we invoke a number of commercial tools to perform placement, routing, back-annotation, and post-layout timing analysis.

4 Experiments

We have tested the proposed method on three industrial designs. All three designs are described as hierarchical gate-level netlists in Verilog. Table 1 shows the characteristics of the designs in which *Nets*, *IOs*, *HMs*, *SMs*(*Before*/*After*), *SMs*(*Cells*/*Gates*), and *Total Gates* denote the number of nets, IO pins, hard macros, soft macros before and after performing clustering, cells/gates of soft macros, and total gate-count of the design. In all experiments, we used the TSMC $0.5\mu m$ cell library [22]. In addition, we set the threshold values $k = 2$ for large-macro decomposition and $S_{th} = 0.1\ S_{avg}$ for small-macro clustering.

In order to examine the effectiveness of our proposed method, we have conducted two sets of experiments, as depicted in Figure 4. In the first set of experiment, we adapt a commonly used design flow which is described as below. In the first step, we used Synopsys's *Design Compiler* [18] to convert the input Verilog design description into a hierarchical gate-level netlist and then performs timing analysis to report the 10 most critical paths. In the second step, we used Cadence's *Silicon Ensemble* [19] to perform chip floorplanning and determine hard-macros' locations. In the third step, we used Cadence's *HLDS* tool [19] to perform soft-macro placement. In the fourth step, we used Cadence's *Silicon Ensemble* [19] to perform detailed placement and routing. In the fifth step, the Cadence's *HyperExtract* [20] was used to extract parasitic information of the layout. In the sixth step, we used AVANT's *STAR*⇔*DC* tool [21] to perform delay calculations and generate an SDF file. Finally, we used Synopsys's *Design Time* [18] to perform post-layout timing analysis. In the second set of experiments, we used our proposed soft-macro formation method as a pre-processing step to generate soft-macro clusters for the floorplanner. In addition, we used our proposed soft-macro placement method to determine soft-macro

Table 1: Characteristics of the benchmarking designs.

Designs	Nets	IOs	HMs	SMs(Before/After)	SMs(Cells/Gates)	Total Gates
Ind1	15,373	83	13	157/22	15,086/38,240	75,000
Ind2	27,404	155	8	150/28	42,030/75,361	95,000
Ind3	53,344	73	31	292/50	45,378/124,180	230,000

Table 2: Comparison of the critical delay generated by the design methodology without/with (Method 1/Method 2) our proposed method.

Designs	Area(μm)	Delay(ns) Method 1	Delay(ns) Method 2	%
Ind1	5,025×5,025	22.5	18.3	-19
Ind2	5,300×5,275	47.2	35.9	-24
Ind3	7,300×7,200	27.6	19.8	-29

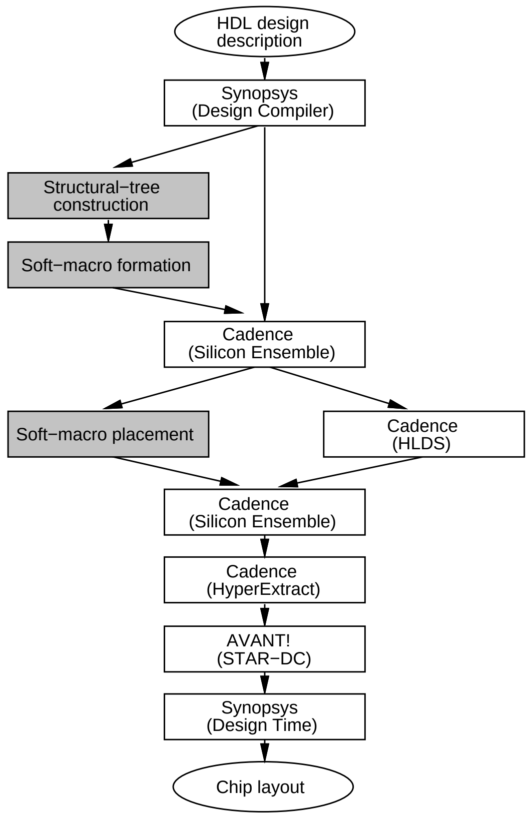

Figure 4: The experimental procedure

locations on the layout plane. For all experiments, we provided the floorplanner (the second step) and the placer-and-router (the fourth step) with the most critical 200, 200, and 400 paths (generated in the first step) as the timing constraints for $ind1$, $ind2$, and $ind3$, respectively.

Table 2 shows the comparison of the design methodology without/with our proposed soft-macro clustering and placement method, in which the delay values are the worst path delays obtained from the post-layout timing analysis. The results show that the design methodology with our proposed method outperforms the one without our proposed method, on an average 24% reduction in the most critical-path delay. Figures 5 and 6 show the most critical path of $ind1$ generated by without/with our proposed method.

5 Conclusions

We have presented a performance-driven soft-macro clustering and placement method by preserving HDL design hierarchy. We have also presented a complete chip design methodology by integrating the proposed method and a set of commercial EDA tools. Experiments on three industrial designs have demonstrated that preserving design hierarchy for soft-macro placement leads to significant improvements in circuit timing.

This is the first attempt to investigate the interaction between HDL synthesis, floorplanning, placement, and routing design tasks. There are many open problems need to be solved in order to successfully develop a deep-submicro-based design methodology. The problems include layout-driven HDL synthesis methods, design refinement methods at synthesis, floorplanning, and place-and-route levels, and accurate timing models development to support synthesis tasks.

References

[1] B. T. Preas and M. J. Lorenzetti, *Physical Design Automation of VLSI Systems*, Benjamin Cummings, Menlo Park, CA., 1988.

[2] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley, 1990.

[3] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, 2nd ed., Kluwer Academic Publishers, 1995.

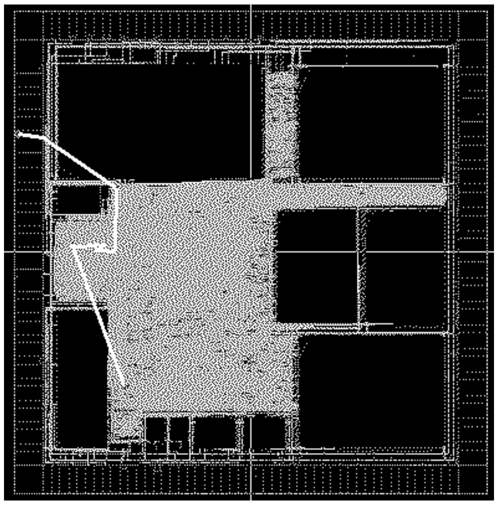

Figure 5: The most critical path generated by the design methodology without applying our proposed method.

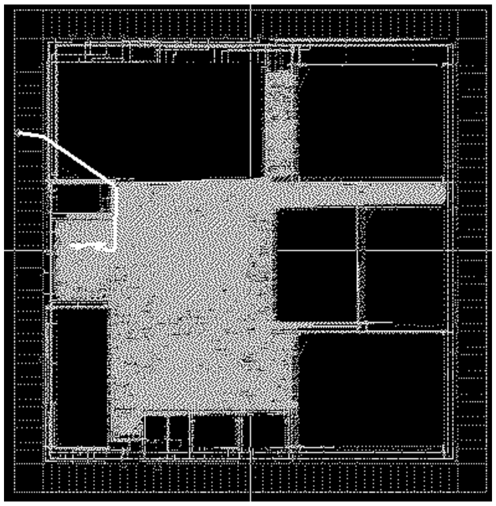

Figure 6: The most critical path generated by the design methodology with applying our proposed method.

[4] C.J. Alpert and A. B. Kahng, "Recent Direction in Netlist Partitioning: A Survey," *INTEGRATION: the VLSI Journal*, N19, pp. 1-81, 1995.

[5] E. S. Kuh, "Physical Design: Reminiscing and Looking Ahead," *Proc. of Int. Symp. on Physical Design*, pp. 206, 1997.

[6] R. Composano, "The Quarter Micro Challenge: Integrating Physical and Logic Design," *Proc. of Int. Symp. on Physical Design*, pp. 211, 1997.

[7] K. Keutzer, A. R. Newwton, and N. Shenoy, "The future of Logic Synthesis and Physical Design in Deep-Submicron Process Geometries," *Proc. of Int. Symp. on Physical Design*, pp. 218-223, 1997.

[8] R. G. Bushroe et al., "Chip Hierarchical Design System (CHDS): A Foundation for Timing-Driven Physical Design into the 21th Century," *Proc. of Int. Symp. on Physical Design*, pp. 212-217, 1997.

[9] G. Odawara, T. Hiraide and O. Nishina, "Partitioning and placement technique for CMOS gate arrays," *IEEE Trans. on Computer-Aided Design*, vol.CAD-6, pp.355-363, May 1987.

[10] Y. C. Wei and C.K. Cheng, "Ratio cut partitioning for hierarchical designs," *IEEE Trans. on Computer-Aided Design*, vol.CAD-10, pp.911-921, July 1991.

[11] S. Dey, F. Beglez and G. Kedem, "Circuit partitioning for logic synthesis," *IEEE Journal of Solid-Stage Circuits*, vol.26, pp.350-363, March 1991.

[12] G. Saucier, D. Brasen, J. P. Hiol, "Partitioning with cone structures," *Proc. of Int. Conf. Computer-Aided Design,* pp.236-239, 1993.

[13] J. Cong and D. Xu, "Exploiting signal flow and logic dependency in standard cell placement," *Proc. of Asia and South Pacific Design Automation Conf.*, pp.399-404, 1995.

[14] Y. W. Tsay and Y. L. Lin, "A Row-Based Cell Placement Method That Utilizes Circuit Structural Properties," *IEEE Trans. on Computer-Aided Design*, vol.14, No. 3, pp.393-397, March 1995.

[15] Y.-W. Tsay, W.-J. Fang, Allen C.-H. Wu, and Y.-L. Lin, "Preserving HDL Synthesis Hierarchy for Cell Placement," *Proc. of Int. Symp. on Physical Design*, pp. 169-174, 1997.

[16] D. M. Schuler and E. G. Ulrich, "Clustering and linear placement," *Proceedings of the 9th Design Automation Conference*, pp.412-419, 1972.

[17] N. R. Quinn, "The Placement Problem as Viewed from the Physics of Classical Mechanics," *Proc. of the 12th Design Automation Conference*, pp. 173-178, 1975.

[18] "HDL Compiler for Verilog Reference Manual Version 3.4b", Synopsys, 1996.

[19] "Silicon Ensemble Reference Manual Version 5.0", Cadence, 1996.

[20] "HyperExtract Reference Manual Version 4.0", Cadence, 1996.

[21] "STAR-DC Reference Manual Version 2.1.2", AVANT!, 1996.

[22] "TSMC ASIC Data Book TCB670", Taiwan Semiconductor Manufacturing Company, Ltd. 1997