A Timing-Driven Soft-Macro Placement and Resynthesis Method in Interaction with Chip Floorplanning

Hsiao-Pin Su, Allen C.-H. Wu, *Member, IEEE*, and Youn-Long Lin, *Member, IEEE*

***Abstract*— In this paper, we present a complete chip design method which incorporates a soft-macro placement and resynthesis method in interaction with chip floorplanning for area and timing improvements. We present a performance-driven soft-macro clustering and placement method which preserves hardware descriptive language (HDL) design hierarchy to guide the soft-macro placement process. We develop a timing-driven design flow to exploit the interaction between HDL synthesis and physical design tasks. During each design iteration, we resynthesize soft macros with either a relaxed or a tightened timing constraint which is guided by the post-layout timing information. The goal is to produce area-efficient designs while satisfying the timing constraints. Experiments on a number of industrial designs ranging from 75-K to 230-K gates demonstrate that the proposed soft-macro clustering and placement method improves critical-path delays on an average of 22%. Furthermore, the results show that by effectively relaxing the timing constraint of noncritical modules and tightening the timing constraint of critical modules, a design can achieve 11% to 30% timing improvements with little to no increase in chip area.**

***Index Terms*— Floorplanning, placement, resynthesis, soft-macro, timing-driven.**

I. Introduction

OVER past decades, academia and industry have invested much effort in physical design related research, including floorplanning, partitioning, placement, and routing. Several excellent reviews of physical design techniques are given by [1]–[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 (IC) designers use an hadrware descriptive language (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 IC designers. This has led to a new research direction in design automation at synthesis and physical levels. Recently, several papers [5]–[8] have addressed the challenges and considerations in physical designs targeted to deep-submicron processes.

A typical HDL-based design flow involves multilevel design tasks. Over the years, much effort has been invested to improve the quality of design tasks at each design level. Few studies have been conducted to investigate the interaction between different design tasks. Pedram and Bhat [9], [10] presented several technology mapping techniques by considering net lengths for area and delay optimization. Liu *et al*. [11] presented a resynthesis technique that resynthesizes the most congested region of the chip to reduce routing area. Stenz *et al.* [12] proposed a timing-driven placement method in interaction with netlist transformations. The netlist transformation procedure is integrated into the placement process so that accurate delay models are available to guide the transformation process. Their results showed that delay reduction is achieved with almost no increase in chip area. Holt and Tyagi [13] proposed an integrated approach that incrementally develops a placement during the logic synthesis process for power minimization.

In this paper, we present a complete chip design method which incorporates a soft-macro placement and resynthesis method in interaction with chip floorplanning for area and timing improvement. The main objective is to develop a timing-driven design flow by exploiting the interaction between HDL synthesis and physical design tasks. 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 II describes the problem. Section III presents the proposed design flow. Section IV gives experimental results. Finally, Section V draws concluding remarks.

II. Problem Description

Fig. 1(a) shows a typical HDL-based chip design flow. It consists of five steps: 1) HDL synthesis, 2) floorplanning, 3) placement-and-routing, 4) back annotation, and 5) post-layout timing analysis. The inputs to the design flow include a mixed register transfer level (RTL) and gate-level HDL description in Verilog or VHDL, and a set of timing constraints. The HDL-based design specification is usually described as a set of

Manuscript received June 9, 1998; revised September 25, 1998. This work was supported by the National Science Council of R.O.C. under Contracts NSC-88-2215-E-007-024 and NSC-88-2215-E-007-030. This paper was recommended by Associate Editor M. Wong.

The authors are with the Department of Computer Science, Tsing Hua University, Hsinchu, Taiwan 300, R.O.C. (e-mail: chunghaw@cs.nthu.edu.tw).

Publisher Item Identifier S 0278-0070(99)02319-2.

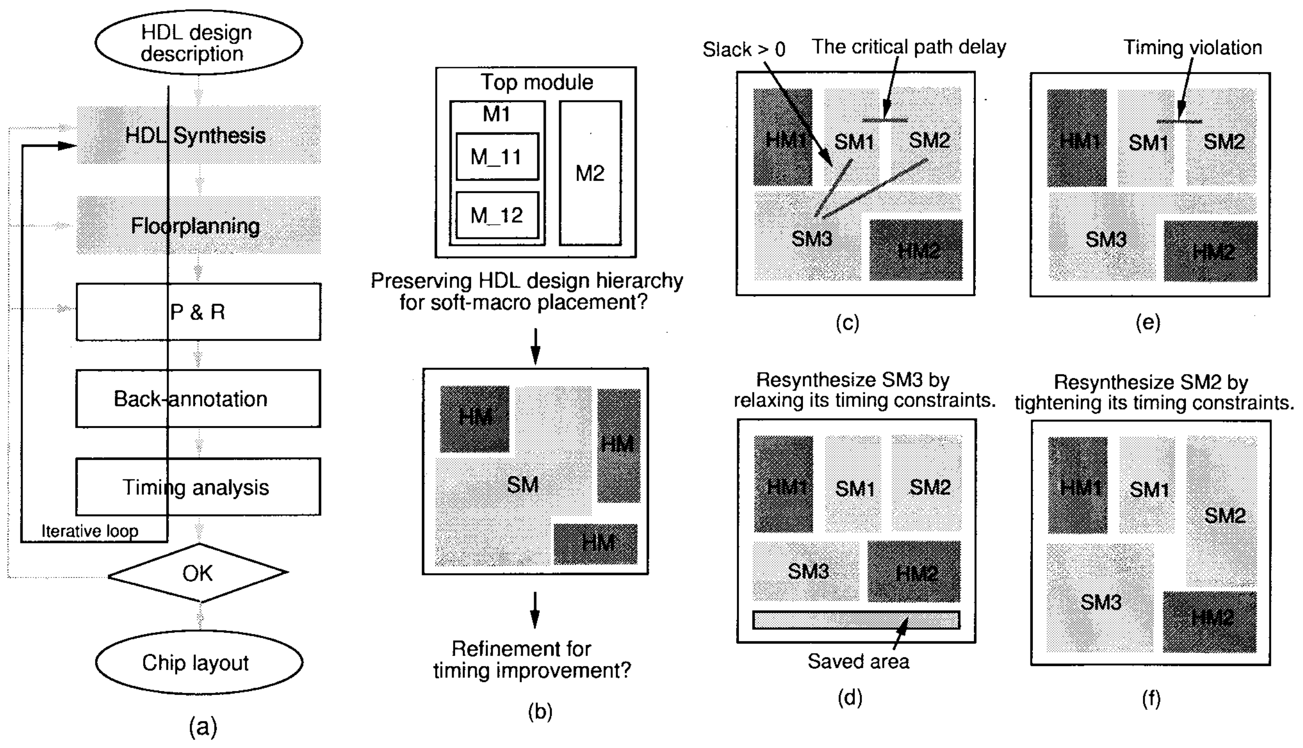


Fig. 1. A typical HDL-based method for macro-based designs: (a) the design flow, (b) soft-macro placement by preserving HDL design hierarchy, (c) design with positive slack values, (d) design with timing-relaxed resynthesis, (e) design with timing violation, and (f) design with timing-tightened resynthesis.

hierarchical modules containing hard macros [e.g., predefined blocks with fixed size and input–output (I/O)-pin location] and soft macros (e.g., blocks that can be implemented with a flexible layout style such as standard-cells).

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 be executed until the timing requirement is satisfied. The refinement procedure can be applied at different design levels. For instance, we can resynthesize certain modules or insert drivers along the critical paths to speed up the circuit timing. We can also adjust the floorplan or rerun a performance-driven floorplanning procedure guiding by the post-layout timing information. Furthermore, we can adjust the soft-macro placement or rerun the detailed placement and routing procedure.

Typically, an HDL-based design flow involves multilevel design tasks. Over the years, much effort has been invested to improve the quality of design tasks 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 multilevel design tasks and exploiting the interaction between them.

In this study, we focus on developing a complete chip design methodology which incorporates a soft-macro placement and resynthesis method in interaction with chip floorplanning for area and timing improvement. There are two main objectives to this research. The first one is to develop a method which can utilize design structural hierarchy to guide soft-macro placement. Several recent studies [14]–[20] 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, as shown in Fig. 1(b).

The second objective is to develop a timing-driven soft-macro resynthesis method by exploiting the interaction between HDL synthesis and physical design tasks, as depicted in Fig. 1(a). Consider a design which consists of five macros, two hard macros and three soft macros. Initially, each soft macro is synthesized into a gate-level netlist. After the floorplanning, placement-and-routing, and post-layout timing analysis, there are two possible cases. Fig. 1(c) shows the first case in which the design satisfies the timing constraint and the critical path occurs between soft macros $SM1$ and $SM2$. Consider that the slack between $SM3$ and $\{SM1, SM2\}$ is larger than zero. This indicates that we may have provided an excessive timing constraint to $SM3$ during the synthesis process. In this case, we can resynthesize $SM3$ with a relaxed timing constraint which usually produces a more area-efficient design, as depicted in Fig. 1(d). Fig. 1(e) shows the second case in which a timing violation occurs between $SM1$ and $SM2$. This indicates that we may have provided an under-estimated timing constraint to either $SM1$ or $SM2$ during the synthesis process. In this

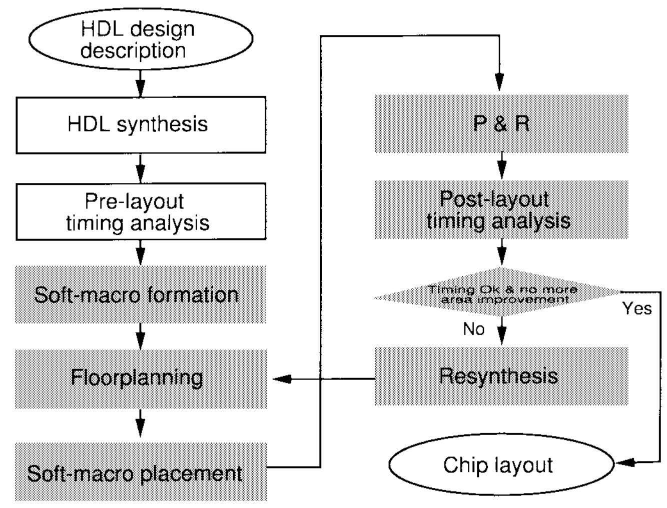


Fig. 2. The proposed design flow.

case, we may have to resynthesize *SM*2 with a tightened timing constraint which can produce a timing-violation free design but costs some area overhead, as depicted in Fig. 1(f). The goal is to produce the most area-efficient design while satisfying the timing constraints.

III. The Proposed Method

A. Overview

Fig. 2 depicts the proposed design flow which consists of eight steps: 1) HDL synthesis, 2) prelayout timing analysis, 3) soft-macro formation, 4) floorplanning, 5) soft-macro placement, 6) placement-and-routing, 7) post-layout timing analysis, and 8) resynthesis. The input to the design flow is an RTL design 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 prelayout timing analysis of the design. A set of critical paths will be identified and used to guide the following macro-clustering, floorplanning, soft-macro placement, and placement-and-routing procedures. In the third step, the system groups soft macros connected to the same clock sources into the same cluster. It also groups small subcircuits to form large macros and decomposes extremely large macros into smaller ones. In the fourth step, we use a commercial floorplanner to perform macro floorplanning to determine the locations of hard macros and then extract the available area for soft macros. In the fifth step, a soft-macro placement procedure is applied to determine the relative location of each soft macro on the layout plane. In the sixth step, we use a commercial tool to perform placement and routing tasks. In the seventh step, a post-layout timing analysis procedure is invoked to compute the final timing of the design. Finally, if there exits a timing violation or there is a chance for area reduction, a soft-macro resynthesis procedure is invoked. The system iterates steps four through the final step until all the timing constraints are satisfied and no more area improvement can be achieved.

In the followings sections, we will discuss the soft-macro formation, soft-macro placement, and soft-macro resynthesis in details.

B. Soft-Macro Formation

There are two main considerations in soft-macro formation. First, in many of today's applications, such as multimedia chips, designs usually have multiple clock sources with different rates. It is beneficial to group soft macros associated with the same clock source into the same cluster. Second, using an HDL-based synthesis method, 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 because a large cluster is too rigid for 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 a large number of macros will increase the computational complexity of the macro-cell placement process.

The soft-macro formation procedure consists of three steps: 1) clock-based clustering, 2) large-macro decomposition, and 3) small-macro clustering. In our approach, we first use a commercial synthesis system to convert a Verilog design description into a hierarchical gate-level netlist. We then construct an HDL-based 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.

After constructing the HDL structural tree of a design, we first group the macros connected to the same clock source into the same cluster. Then 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

$$M_{th} = k * S_{\text{avg}} \tag{1}$$

where S_{avg} is the average macro size *(#Total cells)/(#Macros)*, *#Total cells* and *#Macros* are the total number of cells and the number of soft macros in the design, respectively, and 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 FM partitioning method [21] to recursively decompose large macros into smaller clusters.

Finally, we use a clustering algorithm [22] 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

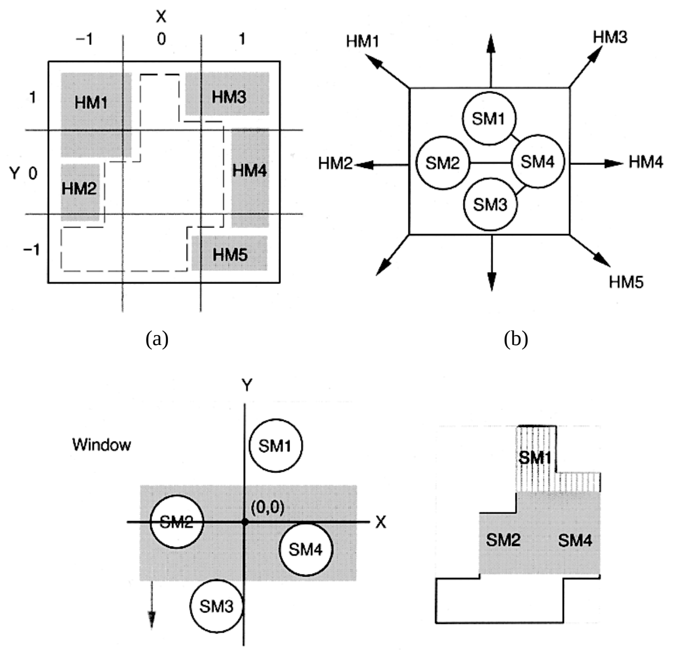


Fig. 3. Soft-macro placement: (a) floorplanning and soft-macro area extraction, (b) force-directed-based placement, and (c) sweeping-based soft-macro assignment.

and v_j, as

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

$$Crit_{ij} = \begin{cases} 1, & \textbf{if } Crit\_Path(v_i \Leftrightarrow v_j) \textbf{ and } \frac{s_i+s_j}{s_{th}} \le 1 \\ 0, & \textbf{else} \end{cases} \tag{3}$$

$$C_{ij} = \alpha\ Conn_{ij} + \beta\ Crit_{ij} \tag{4}$$

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;
α and β	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 zero.

C. 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 Fig. 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. 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 Fig. 3(a). Initially, each hard-macro is assigned into its corresponding region according to the floorplanning result. Then, we apply the force-directed algorithm [23] to determine the relative location of each soft macro, as shown in Fig. 3(b). Let $M = \{m_1, \cdots, m_n\}$ be a set of hard and soft macros. Let $\Delta x_{ij} = |x_i - x_j|$, $\Delta y_{ij} = |y_i - y_j|$, 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

$$F_x^i = \sum_{j=1}^{j=n} (-w_{ij} \times \Delta x_{ij} + r \times \Delta x_{ij}/\Delta d_{ij}) - F_{hm}^x \tag{5}$$

$$F_y^i = \sum_{j=1}^{j=n} (-w_{ij} \times \Delta y_{ij} + r \times \Delta y_{ij}/\Delta d_{ij}) - F_{hm}^y \tag{6}$$

$$F_{hm}^x = \left[\sum_{j=1}^{j=m} \sum_{j=m+1}^{j=n} (-w_{ij} \times \Delta x_{ij} + r \times \Delta x_{ij}/\Delta d_{ij})\right] \Big/ m \tag{7}$$

$$F_{hm}^y = \left[\sum_{j=1}^{j=m} \sum_{j=m+1}^{j=n} (-w_{ij} \times \Delta y_{ij} + r \times \Delta y_{ij}/\Delta d_{ij})\right] \Big/ m \tag{8}$$

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 directions, respectively.

After computing the forces in both x and y directions for each soft macro, we can calculate the relative x and y coordinates of each soft macro on the layout plane, as shown in Fig. 3(c). We then apply a sweeping-based method to assign soft macros into the available layout area, which is described as follows. First, we estimate the width and height of each macro. Since the layout area of each cell can be found in a cell library 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. Second, we compute the width and height of the macros based on 1:1 aspect ratio. Third, we use a window which sweeps from top to bottom in the y direction. The size of the window can be set by the user. In our implementation, we set the window size as the average height of all macros. We then compute the total required area for the soft macros covered by the window, and follow that by allocating a region on the available layout area which is large enough to accommodate the soft macros. 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 to each 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

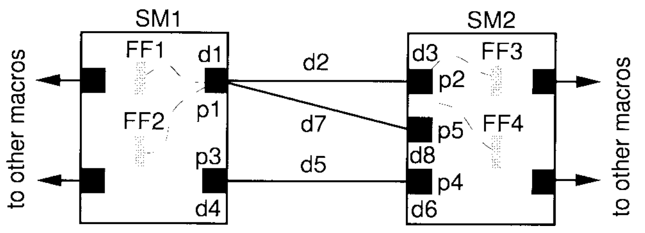


Fig. 4. The slack computation.

assignment procedure continues until all the soft macros are assigned to the layout plane. For example, in Fig. 3(c), the sweeping window first covers one soft macro *SM*1 which is assigned to the top region of the layout plane. In the second sweeping, the window covers two soft macros *SM*2 and *SM*4 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.

D. Soft-Macro Resynthesis in Interaction with Floorplanning

The key issues for the resynthesis process are twofold. First, how to determine which soft macro should be resynthesized. Second, if a soft macro needs to be resynthesized, to what extent can its timing constraint be relaxed or tightened. Our resynthesis procedure consists of two steps: 1) slack computation and 2) soft-macro resynthesis candidate selection.

In the first step, we start by back-annotating the delay information for each I/O port of soft macros. The delay information is extracted from a post-layout timing report. We then compute the slack value for each inter-macro signal path. Finally, we assign a slack value for each I/O port of soft macros. The value is computed using the following formula: $Slack(p_i) = MIN\{Slack(p_i, p_j),\ p_i \in SM_k \text{ and } p_j \in SM_l\}$, where (p_i, p_j) denotes the interconnection between ports p_i and p_j, and SM_k and SM_l denote two soft macros.

The slack of an I/O port is defined as the minimum slack value of all the signal paths associated with this I/O port. For example, Fig. 4 depicts a slack computation example between two soft macros *SM*1 and *SM*2, where $d1$, $d3$, $d4$, and $d6$ denote delays at ports $p1$, $p2$, $p3$, and $p4$, and $d2$, $d5$, and $d7$ denote the wiring delays. The delay at a port p_i is defined as the longest path delay reaching this port. For example, assuming that there are two signal paths in *SM*1 that reach port $p1$, $FF1 \rightarrow p1$ and $FF2 \rightarrow p1$, the delay at port $p1$ is defined as the longest path delay of these two paths. Given a timing constraint T_{const}, the slack between $p1$ and $p2$ is $T_{\text{const}} - (d1 + d2 + d3)$ ($Slack(p1, p2)$), between $p1$ and $p5$ is $T_{\text{const}} - (d1 + d7 + d8)$ ($Slack(p1, p5)$), and between $p3$ and $p4$ is $T_{\text{const}} - (d4 + d5 + d6)$ ($Slack(p3, p4)$). Hence, $Slack(p3)$ and $Slack(p4)$ are equal to $Slack(p3, p4)$, $Slack(p1)$ equals $MIN\ Slack(p1, p2)$, $Slack(p1, p5)$, and $Slack(p2)$ and $Slack(p5)$ is equal to $Slack(p1, p2)$ and $Slack(p1, p5)$, respectively.

In the second step, we use two cost functions to determine which soft macro should be resynthesized next so that maximal area and/or timing improvement can be achieved. The cost functions $POS(SM_i)$ and $NEG(SM_i)$ are defined as the sum of the positive and negative slack values of all the I/O ports in a soft macro

$$POS(SM_i) = \sum Slack(p_j), \quad \text{for all } Slack(p_j) > 0 \quad (9)$$

$$NEG(SM_i) = \sum Slack(p_j), \quad \text{for all } Slack(p_j) < 0. \quad (10)$$

If there exists a negative slack value associated with any soft macro, a timing violation occurs. In this case, we select the soft macro with the highest $NEG(SM_i)$ as the candidate for resynthesis because it should be the most critical one. If all timing satisfies the timing constraint, we select the soft macro with the highest $POS(SM_i)$ as the resynthesis candidate because resynthesizing it with relaxed timing constraints should result in a maximal area reduction. After selecting a candidate, we use a commercial synthesis tool to resynthesize the soft macro by specifying the I/O-ports' timing constraints according to their slack values. Subsequently, we invoke a floorplanning procedure to adjust the chip floorplan by preserving the original relative locations of all soft and hard macros.

The proposed timing-driven soft-macro placement and resynthesis (TSPR) method is described as

```
Procedure_TSPR($D_{\text{hdl}}, T_{\text{const}}$)
begin
  $D_{\text{gate}}$ ← HDL_Synthesis($D_{\text{hdl}}$);
  Pre_layout_timing_analysis($D_{\text{gate}}$);
  $S_{\text{tree}}$ ← Structural_tree_construction ($D_{\text{gate}}$);
  Soft_macro_formation($S_{\text{tree}}$);
  Floorplanning($S_{\text{tree}}$, $T_{\text{const}}$);
  Soft_macro_placement($S_{\text{tree}}$, $T_{\text{const}}$);
  Place_route($D_{\text{gate}}$, $T_{\text{const}}$);
  RC_extraction($D_{\text{gate}}$);
  Post_layout_timing_analysis($D_{\text{gate}}$);
  Slack_computation($S_{\text{tree}}$, $D_{\text{gate}}$);
  while (timing constraint is violated or more area
    can be reduced)
  begin
    $POS(SM_i)$_and_$NEG(SM_i)$_computation($S_{\text{tree}}$);
    $SM_i$ ← Soft_macro_candidate_selection($S_{\text{tree}}$);
    $S_{\text{tree}}$ ← HDL_Synthesis($SM_i$);
    Floorplan_place_route_ECO($S_{\text{tree}}$, $T_{\text{const}}$);
    RC_extraction($D_{\text{gate}}$);
    Post_layout_timing_analysis($D_{\text{gate}}$);
    Slack_computation($S_{\text{tree}}$, $D_{\text{gate}}$);
  end_of_while
end_of_procedure
```

The inputs to the system include an HDL design description (D_{hdl}) and timing constraints (T_{const}). Let D_{gate} and S_{tree} denote the gate-level design and structural tree. Initially, the system performs HDL synthesis, prelayout timing synthesis, structural-tree construction, soft-macro formation, floorplanning, soft-macro placement, placement-and-routing, and post-layout timing analysis to produce an initial chip layout. During the resynthesis iteration, the system first computes the slack value for each soft macro I/O port, and then computes the slack value for each soft macro. Following, the system selects one soft-macro candidate which contributes the most in timing or area improvement. After resynthesizing the soft macro, the system performs floorplanning and placement-and-routing

TABLE I
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

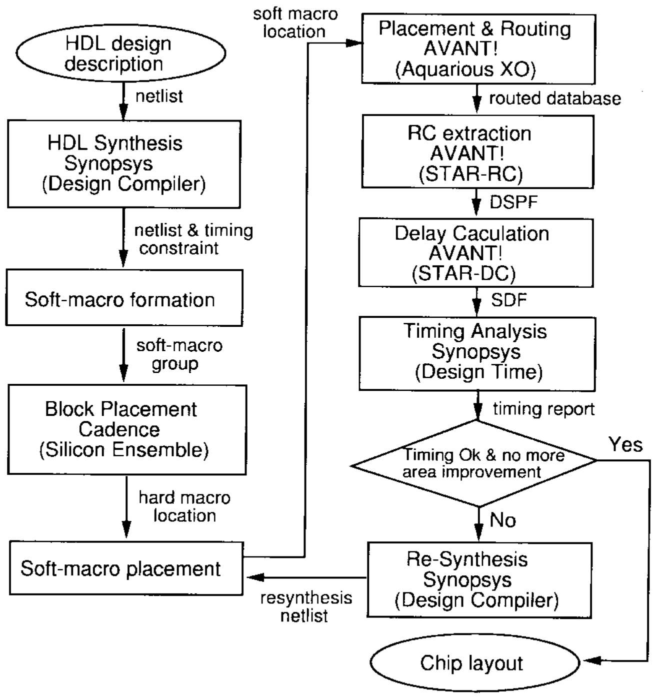

HDL design description
netlist
HDL Synthesis Synopsys (Design Compiler)
netlist & timing constraint
Soft-macro formation
soft-macro group
Block Placement Cadence (Silicon Ensemble)
hard macro location
Soft-macro placement
soft macro location
Placement & Routing AVANT! (Aquarious XO)
routed database
RC extraction AVANT! (STAR-RC)
DSPF
Delay Caculation AVANT! (STAR-DC)
SDF
Timing Analysis Synopsys (Design Time)
timing report
Timing Ok & no more area improvement
Yes
No
Re-Synthesis Synopsys (Design Compiler)
resynthesis netlist
Chip layout

Fig. 5. The experimental flow.

ECO, followed by RC parasitic extraction and post-layout timing analysis. Finally, if there is improvement, then the resynthesis iteration continues. Otherwise, the system stops and reports the final chip layout.

IV. EXPERIMENTS

We have tested the proposed method on three industrial designs. All three designs are described as hierarchical, mixed RTL and gate-level netlists in Verilog. Table I shows the characteristics of the designs in which *Nets*, *I/Os*, *HMs*, *SMs(Before/After)*, *SMs(Cells/Gates)*, and *Total Gates* denote the number of nets, I/O pins, hard macros, soft macros before and after performing clustering, cells/gates of soft macros, and total gate-count of the design. In the three designs, *ind*1 contains three clock sources, and *ind*2 and *ind*3 contain two clock sources. In all experiments, we set the threshold values $k=2$ for large-macro decomposition and $S_{th} = 0.1S_{\text{avg}}$ for small-macro clustering.

Fig. 5 shows the experimental flow. In the first step, we used Synopsys' *Design Compiler* [24] to convert the input Verilog design description into a hierarchical, gate-level netlist and then performed timing analysis to report the 200 most critical paths. In the second step, we used our proposed soft-macro formation method as a preprocessing step to generate soft-macro clusters. In the third step, we used Cadence's *Silicon Ensemble* (*Block Placement*) [25] to perform chip floorplanning and determine the location of hard macros.

TABLE II
COMPARISON OF THE CRITICAL DELAY GENERATED BY THE DESIGN METHODOLOGY WITHOUT/WITH (METHOD1/METHOD 2) OUR PROPOSED METHOD

Designs	Area(μm)	Delay(ns) Method 1	Delay(ns) Method 2	%
Ind1	5,025×5,025	22.5	18.35	-19
Ind2	5,300×5,275	47.2	38.25	-19
Ind3	7,300×7,200	27.6	19.80	-29

It first imported the entire design, then flattened the soft macros into a random-logic circuit, and finally performed hard-macro placement. Because the three designs we used in the experiments contain analog and memory modules, an inferior floorplan can be expected to greatly reduce chip quality. Hence, after generating the initial floorplan, we consulted the designers and fine-tuned the floorplan manually. In the fourth step, we used the proposed performance-driven soft-macro placement algorithm to perform soft-macro placement. In the fifth step, we used AVANT!'s *Aquarious XO* [26] to perform detailed placement and routing. In the sixth step, AVANT!'s *STAR-RC* [27] was used to extract parasitic layout information. In the seventh step, we used AVANT!'s *STAR-DC* tool [28] to perform delay calculations and generate an standard delay format (SDF) file. In the eighth step, we used Synopsys' *Design Time* [24] to perform post-layout timing analysis. Finally, we applied the proposed soft-macro resynthesis iteration to incrementally improve the area and timing of the layout. During each resynthesis iteration, we first used Synopsys' *Design Compiler* to perform logic resynthesis by supplying a relaxed or a tightened timing constraint to the soft macros. We then applied an ECO function supported by AVANT!'s *Aquarious XO* to perform placement and routing. For all experiments, we provided the floorplanner (the third step) and the placer-and-router (the fifth step) with the most critical 200 paths (generated in the first step) as the timing constraints.

We have conducted two experiments to further examine the effectiveness of our proposed methods. In the first experiment, we compared the layouts generated by applying our proposed soft-macro clustering and placement method (Fig. 5) and those without (in this case we used Cadence's *Silicon Ensemble* [25] to perform chip floorplanning and Cadence's *HLDS* tool [25] to perform soft-macro placement). In this experiment, we used the Taiwan Semiconductor Manufacturing Company, Ltd. (TSMC) 0.5-μm cell library [29].

Table II compares the design quality without/with our proposed soft-macro clustering and placement method. The delay values are the worst path delays obtained from the post-layout timing analysis. The results show that the design methodology that uses our proposed method outperforms the one that does not by demonstrating an average 22% reduction in the most critical-path delay. Figs. 6 and 7 show the most critical path of *ind*1 with and without our proposed method.

In the second experiment, we tested the proposed timing-driven resynthesis method. We have conducted two sets of experiments. In the first experiment, we used the TSMC 0.5-μm cell library [29]. In the second experiment, we used the TSMC 0.25-μm cell library [29]. Note that *ind*1 and *ind*3 contain a phase-locked loops (PLL) module. Unfortunately, the

TABLE III
THE AREA-DELAY COMPARISONS OF *ind*1 USING THE 0.5-μm LIBRARY

Iter	IO	#HM	#SM(B/A)	$Gate_{SM}$	$Gate_{Tot}$	Area(μm^2)	Delay(ns)	T_{resyn}(hr)	T_{eco}(hr)
1	83	13	157/22	38,240	75,000	25,250,625	18.35	6	4
2	83	13	157/22	38,279	75,039	25,251,118	15.87	5	4
3	83	13	157/22	38,260	75,020	25,252,218	13.91	4	3

TABLE IV
THE AREA-DELAY COMPARISONS OF *ind*2 USING THE 0.5-μm LIBRARY

Iter	IO	#HM	#SM(B/A)	$Gate_{SM}$	$Gate_{Tot}$	Area(μm^2)	Delay(ns)	T_{resyn}(hr)	T_{eco}(hr)
1	155	8	150/28	75,361	95,000	27,957,500	38.25	9	7
2	155	8	150/28	75,501	95,140	28,037,599	33.71	8	5
3	155	8	150/28	75,533	95,172	28,039,765	33.30	6	4

TABLE V
THE AREA-DELAY COMPARISONS OF *ind*3 USING THE 0.5-μm LIBRARY

Iter	IO	#HM	#SM(B/A)	$Gate_{SM}$	$Gate_{Tot}$	Area(μm^2)	Delay(ns)	T_{resyn}(hr)	T_{eco}(hr)
1	73	31	292/50	124,180	230,000	52,560,000	19.83	13	9
2	73	31	292/50	124,479	230,299	52,563,260	18.95	8	7
3	73	31	292/50	124,611	230,431	52,565,788	17.64	7	6

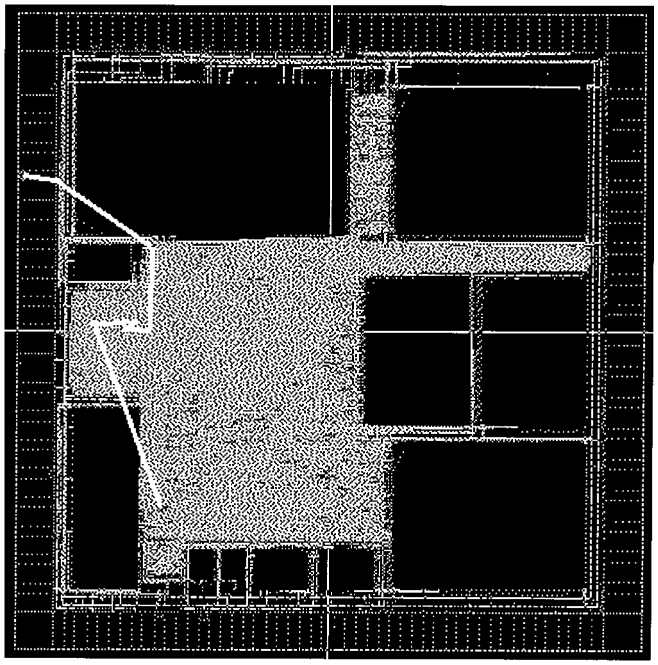


Fig. 6. The most critical path generated without our proposed method.

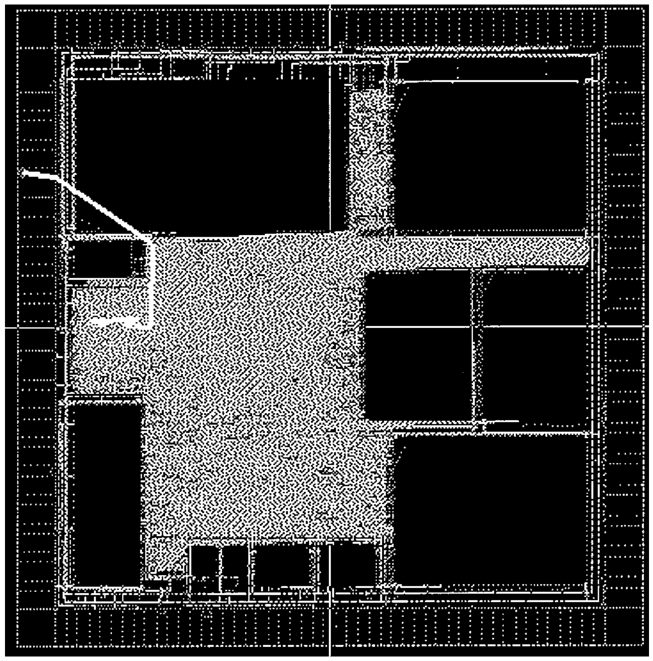


Fig. 7. The most critical path generated with our proposed method.

0.25-μm-based PLL module is not available and we could not perform the experiment on the both designs using the TSMC 0.25-μm cell library. Hence, in this paper, we only present the result of *ind*2 using the TSMC 0.25-μm cell library.

Table III shows the area-delay comparisons of *ind*1 using the 0.5-μm library, in which IO denotes the number of I/O pins, $\#HM$ the number of hard macros, *#SM*(B/A) the number of soft macros before and after applying the soft macro clustering, *Gate*$_{SM}$ the total gate count of soft macros, *Gate*$_{Tot}$ the total gate count, *Area* the chip area, *Delay* the worst path delay, T_{resyn} the resynthesis run times in hours, and T_{eco} the ECO run times. The results show that by resynthesizing some soft macros, the timing was improved up to 20% with almost no area penalty. Table IV shows the area-delay comparisons of *ind*2 using the 0.5-μm library. We obtained the same result as that of *ind*1, in which the timing was improved up to 13% with almost no area penalty. Table V shows the area-delay comparisons of *ind*3 using the 0.5-μm library. The results show that the timing was improved up to 11% with almost no area penalty. Fig. 8 shows the critical path before resynthesis (*Iteration* 1 in Table IV). After two resynthesis iterations, the new critical path is shown in Fig. 9 (*Iteration* 3 in Table IV). Table VI shows the area-delay comparisons of *ind*2 using the 0.25-μm library. The results show that the timing was improved up to 30% with 11% area penalty.

We have also compared the average delays contributed by gates and interconnects using 0.5-μm and 0.25-μm technologies. Table VII shows the average gate and interconnect delay

TABLE VI
THE AREA-DELAY COMPARISONS OF *ind*2 USING THE 0.25-μm LIBRARY

Iter	IO	#HM	#SM(B/A)	$Gate_{SM}$	$Gate_{Tot}$	Area(μm^2)	Delay(ns)	T_{resyn}(hr)	T_{eco}(hr)
1	155	8	150/28	75,361	95,000	6,250,000	27.68	10	7
2	155	8	150/28	75,610	95,249	6,388,250	25.67	9	5
3	155	8	150/28	76,922	96,561	6,566,400	21.67	9	5
4	155	8	150/28	78,169	97,808	7,022,500	19.32	8	4

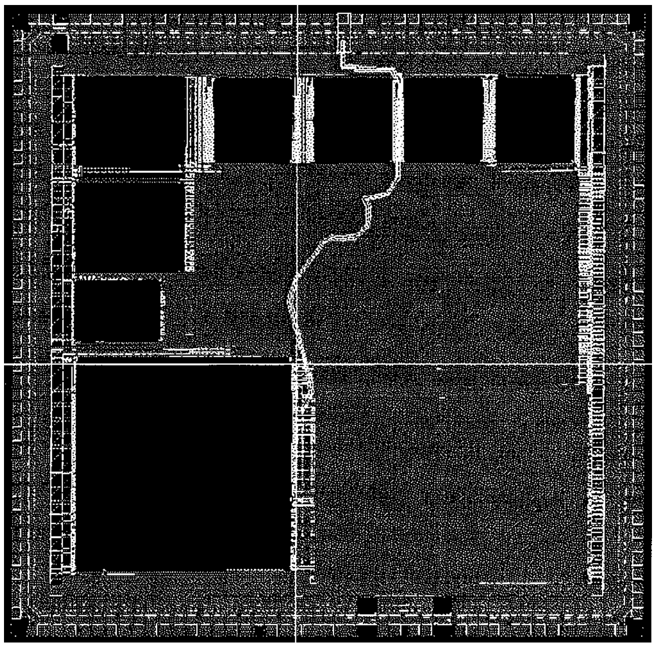


Fig. 8. The initial critical path of *ind*2 using the 0.5-μm library.

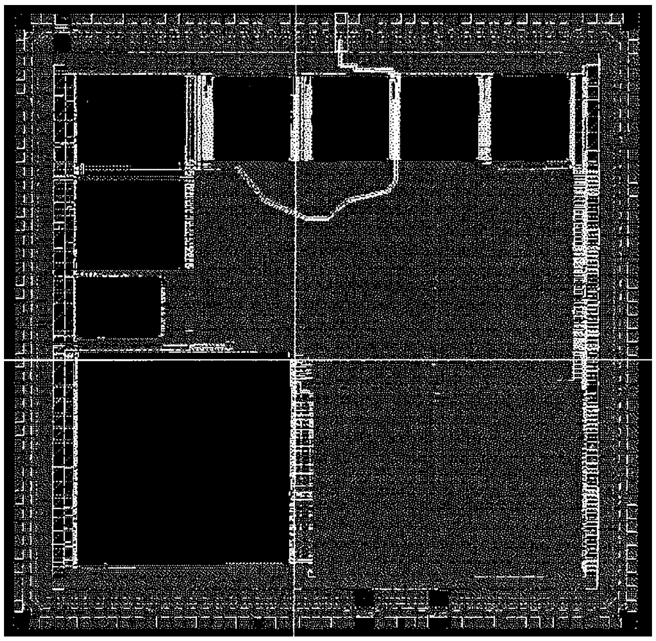


Fig. 9. The new critical path of *ind*2 using the 0.5-μm library after two resynthesis iterations.

comparisons of the most critical paths of *ind*2. The results show that using the 0.5-μm technology the average gate's intrinsic delay and interconnect delay are 0.171 ns and 0.277 ns, respectively. In addition, using the 0.25-μm technology, the average gate's intrinsic delay and interconnect delay are 0.107 ns and 0.325 ns, respectively. From the results, we observed that the average interconnect-delay versus gate-delay ratios of the 0.5-μm and 0.25-μm technologies are 1.62 and 3.04. This indicates that interconnect delays play an important role in deep-submicron technologies.

From the experiments, the following observations can be made. When using the 0.5-μm library for designs *ind*1, *ind*2, and *ind*3 our proposed method can improve timing from 11% to 20% with almost no area penalty. This demonstrates that by effectively relaxing the timing constraints of noncritical modules and tightening the timing constraints of the critical modules we can achieve significant timing improvements with little to no increases in chip area. When using the 0.25-μm library, our method can improve timing by 8%, 22%, and 30% with 2%, 5%, and 11% increase in chip area, respectively. We found that the 0.25-μm library supports a large set of components with a wide range driven capability. This feature provides more design alternatives during the synthesis process.

The experiments were conducted on an HP-C180 workstation with 750-Mb main memory. Tables III–VI show the run times for the resynthesis and placement and routing (P&R) engineering change order (ECO) iteration. For example, in the first iteration of the *ind*1 design (Table III), it took an average of 6 h and 4 h to run the synthesis and P&R and ECO tasks.

TABLE VII
THE COMPARISONS OF THE AVERAGE GATE AND INTERCONNECT DELAYS OF *ind*2 USING THE 0.5-μm AND 0.25-μm LIBRARIES

Lib.	Gate Delay(ns)[A]	Interconnect Delay(ns)[B]	[B]/[A]
0.5μm	0.171	0.277	1.62
0.25μm	0.107	0.325	3.04

V. CONCLUSION

In this paper, we have presented a complete chip design method which incorporates a soft-macro placement and resynthesis method in interaction with chip floorplanning for area and timing improvements. We have conducted a series of experiments on three industrial designs. The results have demonstrated that preserving design hierarchy for soft-macro placement leads to significant improvements in circuit timing. Furthermore, the results have also demonstrated that by effectively relaxing the timing constraints of noncritical modules and tightening the timing constraints of critical modules we can achieve significant timing improvements with very little area penalty.

In this study, we have shown that an integrated synthesis, floorplanning, placement, and routing design flow allows designers to perform design resynthesis and ECO-based placement-and-routing guided by accurate timing information. This method is very effective for timing improvement with very little increase in chip area. One drawback for such a design flow is that it is an extremely time-consuming task. It takes close to one full-day to run one resynthesis iteration. Shortening the iteration time will be a key factor in improving the design exploration process. One possible approach is to move the iteration loop to a higher-level, such as the floorplanning level. In order to make this happen, a more accurate delay and area estimation method is required. Another important issue is how to determine the initial timing budget for each module before synthesis. Good initial time-budgeting should shorten the number of resynthesis iterations and thus speed up the entire design process.

REFERENCES

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

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

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

[4] C. J. Alpert and A. B. Kahng, "Recent direction in netlist partitioning: A survey," *INTEGRATION: The VLSI J.*, vol. N19, pp. 1–81, 1995.

[5] E. S. Kuh, "Physical design: Reminiscing and looking ahead," in *Proc. Int. Symp. Physical Design*, 1997, p. 206.

[6] R. Composano, "The quarter micro challenge: Integrating physical and logic design," in *Proc. Int. Symp. Physical Design*, 1997, p. 211.

[7] K. Keutzer, A. R. Newton, and N. Shenoy, "The future of logic synthesis and physical design in deep-submicron process geometries," in *Proc. Int. Symp. Physical Design*, 1997, pp. 218–223.

[8] R. G. Bushroe, S. DasGupta, A. Dengi, P. Fisher, S. Grout, G. Ledenbach, N. S. Nagaraj, and R. Steele, "Chip hierarchical design system (CHDS): A foundation for timing-driven physical design into the 21st century," in *Proc. Int. Symp. Physical Design*, 1997, pp. 212–217.

[9] M. Pedram and N. Bhat, "Layout driven logic restructuring/decomposition," in *Proc. Int. Conf. Computer-Aided Design*, 1991, pp. 134–137.

[10] ——, "Layout driven technology mapping," in *Proc. 28th Design Automation Conf.*, 1991, pp. 99–105.

[11] S. Liu, K. Pan, M. Pedram, and A. M. Despain, "Alleviating routing congestion by combing logic resynthesis and linear placement," in *Proc. Eur. Conf. Design Automation*, 1993, pp. 578–582.

[12] G. Stenz, B. M. Riess, B. Rohfleisch, and F. M. Johannes, "Timing driven placement in interaction with netlist transformations," in *Proc. Int. Symp. Physical Design*, 1997, pp. 36–41.

[13] G. Holt and A. Tyagi, "Minimizing interconnect energy through integrated low-power placement and combinational logic synthesis," in *Proc. Int. Symp. Physical Design*, 1997, pp. 48–53.

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

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

[16] S. Dey, F. Beglez, and G. Kedem, "Circuit partitioning for logic synthesis," *IEEE J. Solid-Stage Circuits*, vol. 26, pp. 350–363, Mar. 1991.

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

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

[19] Y. W. Tsay and Y. L. Lin, "A row-based cell placement method that utilizes circuit structural properties," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 393–397, Mar. 1995.

[20] Y.-W. Tsay, W.-J. Fang, Allen C.-H. Wu, and Y.-L. Lin, "Preserving HDL synthesis hierarchy for cell placement," in *Proc. Int. Symp. Physical Design*, 1997, pp. 169–174.

[21] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. 19th Design Automation Conf.*, 1982, pp. 175–181.

[22] D. M. Schuler and E. G. Ulrich, "Clustering and linear placement," in *Proc. 9th Design Automation Conf.*, 1972, pp. 412–419.

[23] N. R. Quinn, "The placement problem as viewed from the physics of classical mechanics," in *Proc. 12th Design Automation Conf.*, 1975, pp. 173–178.

[24] *HDL Compiler for Verilog Reference Manual Version 3.4b*, Synopsys, Mountain View, CA, 1996.

[25] Silicon Ensemble Reference Manual Version 5.0, Cadence, San Jose, CA, 1996.

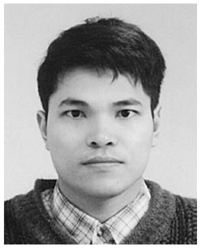
[26] *Aquarious XO Reference Manual Version 2.1.2*, AVANT!, Fremont, CA, 1998.

[27] *STAR-RC Reference Manual Version 2.2*, AVANT!, Fremont, CA, 1997.

[28] *STAR-DC Reference Manual Version 2.1.2*, AVANT!, Fremont, CA, 1996.

[29] *TSMC ASIC Data Book TCB670*, Taiwan Semiconductor Manufacturing Company, Ltd. Hsin-chu, Taiwan, R.O.C., 1997.

[30] TSMC DSD Data Book ACB872, Taiwan Semiconductor Manufacturing Company, Ltd., Hsin-chu, Taiwan, R.O.C., 1998.



Hsiao-Pin Su received the B.S. degree in computer science form Chung Yuan Christian University Chung-Li, Taiwan, R.O.C. He is currently a Ph.D degree candidate of the Department of Computer Science, Tsing Hua University, Hsinchu, Taiwan.

His primary research interest is in computer-aided design (CAD) of very large-scale integrated circuits (VLSI).

Allen C.-H. Wu (S'84–M'93) received the B.S. degree in electronic engineering from Taiwan Institute of Technology, Taipei, Taiwan, in 1983, the M.S. degree in electrical and computer engineering from University of Arizona, Tucson, in 1985, and the Ph.D. degree in computer science from University of California, Irvine, in 1992.

From 1985 to 1988, he was a Research Engineer in the Physiology Department at University of Arizona, Tucson. From 1995 to 1996, he was a Visiting Senior Staff Engineer at Quickturn Design Systems Inc., Mountain View, CA. He is currently a Professor of Computer Science at Tsing Hua University, Hsinchu, Taiwan. His research interests include high-level synthesis, FPGA synthesis, rapid prototyping, and system-level design methodology and environment. He is the Program Chair and General Chair of the ISSS'98 and ISSS'99, respectively.

Dr. Wu is a member of the IEEE Computer Society, the IEEE Circuits and Systems Society, and the Association for Computing Machinery (ACM).

Youn-Long Lin (S'86–M'87) received the B.S. degree in electronic engineering from Taiwan Institute of Technology, Taipei, Taiwan, in 1982, and the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, in 1987.

Upon his graduation, he joined the Department of Computer Science, Tsing Hua University, Hsinchu, Taiwan, where he is currently a Professor. His primary research interest is in CAD for VLSI with emphasis on physical design automation and high-level synthesis.

Dr. Lin was co-recipient the Outstanding Young Author Award from the IEEE Circuit and System Society in 1990 and received a Distinguished Researcher Award from the National Science Council, R.O.C., in 1992, 1994, and 1996. He has served on program committee, steering committee, and executive committee for several international conferences, and workshops on various aspects of CAD. He is a member of the IEEE Computer Society and of the IEEE Circuit and System Society.