Preserving HDL Synthesis Hierarchy for Cell Placement

Yu-Wen Tsay, Wen-Jong Fang, Allen C.-H. Wu, and Youn-Long Lin
Department of Computer Science
Tsing Hua University
Hsinchu, Taiwan, 300, Republic of China

Abstract

We propose an integrated HDL-synthesis and placement method for row-based layouts. Our approach bridges the gap between HDL synthesis and placement by fully utilizing design hierarchy. It first synthesizes an HDL design specification into a hierarchy of subcircuits. It then groups subcircuits to form strongly connected macro cells, followed by performing a macro-cell placement to determine the location of the macro cells on the layout plane. Finally, it maps the resulting macro-cell placement into a row-based placement and applies a simulated-annealing procedure to refine the row-based placement. Experiments on a number of large industry designs demonstrate that the proposed method achieves, on the average, 22% area reduction, 18% wire length reduction, and several times of speed up compared to that without the hierarchy information.

1 Introduction

Due to the pressure of designing more complex chips quickly and the maturity of the tools, more and more VLSI designers are using the HDL-based synthesis approach. The synthesized results, in the form of gate-level netlist, have to be fed into the physical design back-end. It would be helpful if, in addition to the connectivity netlist, the synthesizers can provide and the back-end can utilize some useful information generated during the synthesis process.

Structural hierarchy is one such kind of information. Since most HDL-based design specifications are hierarchical, their synthesized results are naturally hierarchical. Even before the era of synthesis-based design, many placement approaches have tried to extract hierarchy out of a flatten netlist because it has been shown that structural hierarchy helps in not only the computation efficiency but also the layout quality. In the deep submicron era, hierarchy-preserving layout is even more important because otherwise the unpredictable wiring delay will void any synthesis decisions at the high level.

Recently, several studies observed the phenomena of natural structures. In [1], the ratio-cut cost-function was proposed to help find the so-called "natural" clusters of a circuit. Dey and Brglez [2] define *corolla* as a set of cells with strong connections. Their method partitions a circuit into a set of disjoint *corollas* to assist logic synthesis and resynthesis. Odawara et al [3] exploit the circuit structure and incorporate it into a gate-array placement method based on the force-directed heuristic. In [4], a placement method was proposed to utilize circuit structural properties for row-based placement. It first extracts strongly connected subcircuits, called *cones*, and then performs a macro-cell placement treating each *cone* as a soft macro. Finally, it maps the resulting macro-cell placement into a row-based placement. These methods demonstrate that the layout quality can be significantly improved when the circuit structural information is considered during the placement process. However, all of these methods try to extract circuit structural properties from a flattened netlist. It is a non-trivial problem. Consequently, the quality of extraction will directly affect the quality of the final layout.

In this paper, we propose an integrated HDL-synthesis and placement approach for row-based layouts. Our approach bridges the gap between HDL synthesis and placement by making HDL hierarchy available to the placement tool. It first synthesizes an HDL design specification into a hierarchy of subcircuits. It then groups subcircuits to form strongly connected macro cells, and performs a macro-cell placement to determine the location of the macro cells on the layout plane. Finally, it maps the resulting macro-cell placement into a row-based one and applies a simulated-annealing procedure to refine the row-based placement. Experiments on a set of industry designs are reported. The results demonstrate that the proposed approach can produce high-quality row-based layouts very quickly.

The rest of paper is organized as follows. Section 2 describes the problem and the motivation behind this work. Section 3 presents the proposed approach. Section 4 presents some experimental results. Finally, section 5 gives our concluding remarks.

This work was supported in part by the National Science Council of R.O.C. under contracts NSC 86-2221-E-007-019 and NSC 86-2221-E-007-047.

ISPD '97 Napa Valley, California USA

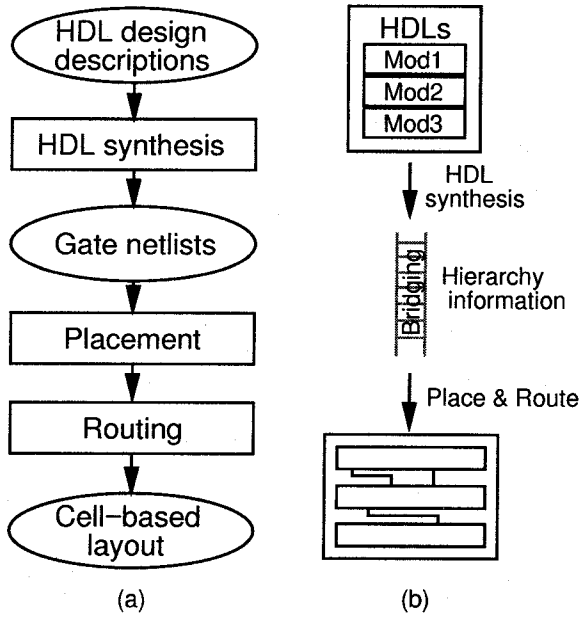

Figure 1: An HDL-based design flow for cell-based layouts: (a) a traditional design flow, (b) bridging HDL synthesis and place-and-route.

2 Problem description and motivation

Figure 1(a) depicts a typical HDL-based design flow for cell-based layouts which consists of two independent design steps: (1) HDL synthesis and (2) placement and routing. In the first step, a synthesizer converts an HDL design description into a gate-level netlist by performing a series of tasks, including HDL compilation, RTL synthesis, logic synthesis, and technology mapping. In the second step, a placer-and-router first maps the gate-level design onto the layout plane and then realizes the interconnections between cells.

Because these design tasks are executed independently, this approach does not take into account the interaction between HDL synthesis, placement, and routing. Furthermore, most existing approaches based on this design flow apply the place-and-route procedure directly on a flattened gate-level netlist without exploiting the structural information. Recently, several approaches [1, 2, 3, 4] try to extract circuit structural properties from flattened netlists so that this information can be used to improve the quality of the placement result. Using this approach, the quality of the extracted strongly-connected subcircuits will directly affect the quality of the final placement result. As a result, how to properly extract strongly-connected subcircuits has become a key to achieve high quality placement. However, circuit extraction is a non-trivial problem. This motivates us to investigate how to bridge synthesis and placement design tasks into a single design flow so that the HDL structural properties can be preserved and passed down to the placement process, as depicted in Figure 1(b).

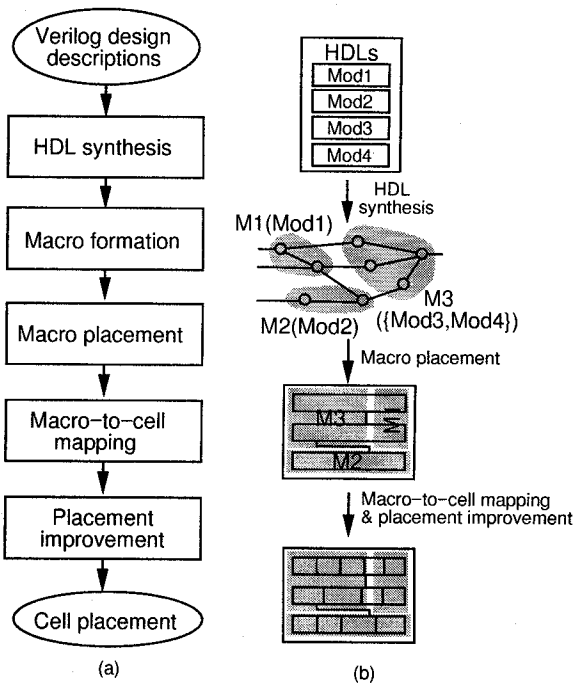

Figure 2: The proposed method: (a) the design flow, (b) an example.

3 An integrated HDL-synthesis and placement approach

3.1 Overview

Figure 2(a) depicts the proposed design flow consisting of five steps: (1) HDL synthesis, (2) macro formation, (3) macro placement, (4) macro-to-cell mapping, and (5) placement improvement. Input to the system is a Verilog design description. Its output is the location on the layout plane for each cell.

In the first step, an HDL-based synthesizer converts the Verilog design description into a hierarchical structural tree by performing HDL compilation and a series of RTL and logic synthesis tasks. In the second step, small subcircuits are grouped to form large macros. In the third step, a macro-cell placement procedure is applied to determine the relative location of each macro on the layout plane. In the fourth step, a macro-to-cell mapper maps a layout in the macro-cell layout architecture into the one in the row-based layout architecture which will be treated as an initial placement. Finally, a placement refinement procedure takes the mapped placement as its initial placement and performs placement improvements to obtain the

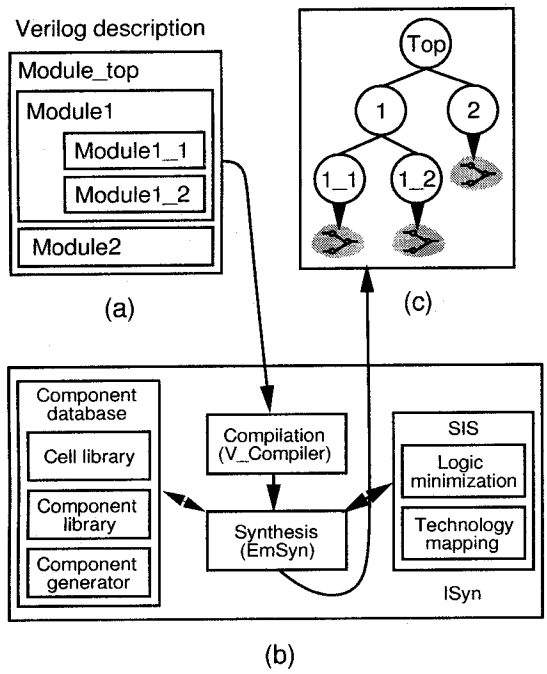

Figure 3: A hierarchical HDL synthesis method: (a) a Verilog description example, (b) the *ISyn* system, (c) the structural tree.

final placement.

Figure 2(b) illustrates the proposed design flow. In this example, the Verilog design description consists of four modules $Mod1$, $Mod2$, $Mod3$, and $Mod4$ forming three macros $M1$, $M2$ and $M3$. After performing macro placement and macro-to-cell mapping, the cells in the same macro are placed closely on the row-based layout plane.

3.2 A hierarchical HDL synthesis method

The main objective of the HDL synthesizer is to generate a design while preserving the design structural information for macro formation. Using an HDL-based design flow, the design is specified as a mixed RTL/logic/gate-level description in some Hardware Description Languages (HDLs) such as VHDL and Verilog. An HDL description usually consists of a hierarchy of interconnected modules, as shown in Figure 3(a). 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 synthesis system *ISyn* [5] to convert a Verilog design description into a structural tree and its corresponding gate-level netlists. Figure 3(b) shows the block diagram of *ISyn* consisting of four subsystems: a Verilog compiler (*V_Compiler*), a synthesizer (*EmSyn*), a logic synthesis system (*SIS*), and a component database. *V_Compiler* transforms the input Verilog description into an intermediate format. *EmSyn* performs RTL synthesis including component and interconnect binding. It also interfaces to the SIS system [6] which generates the netlist after performing logic minimization and technology mapping. The component database provides *EmSyn* with both RTL components and library cells. Finally, *ISyn* generates an HDL structural tree with the gate-level netlists.

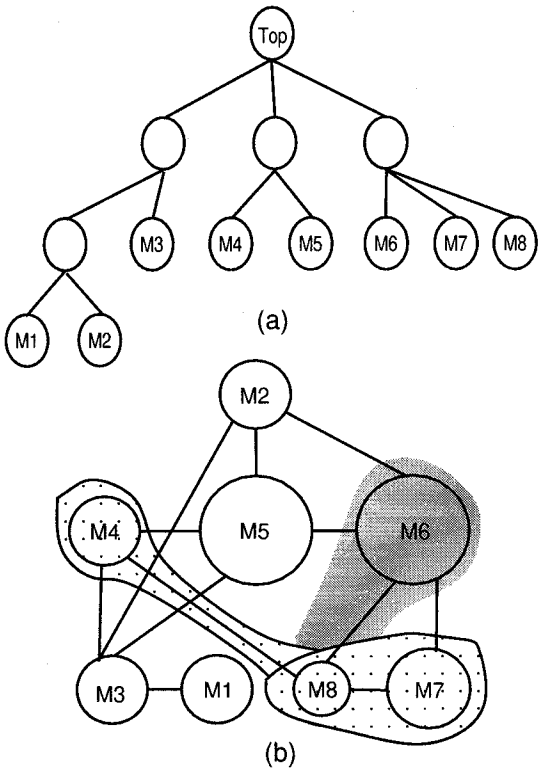

Figure 4: The macro formation: (a) an HDL structural tree, (b) it corresponding connected graph.

We use an HDL structural tree to represent the structural hierarchy of a Verilog design description. In an HDL structural tree, the root node represents the design, and each intermediate node represents a module construct. Each leaf node represents a circuit block generated from a leaf module. For example, Figure 3(c) depicts the HDL structural tree corresponding to the Verilog description shown in Figure 3(a).

3.3 Macro formation

The synthesized subcircuit of each leaf module is naturally a closely-connected cluster. However, for designs containing a large number of module descriptions we will get many small subcircuits. This is undesirable for two reasons. First, we treat each subcircuit as a soft macro. Hence, a large number of macros

will increase the computational effort of the macro-cell placement process. Second, during the development of the proposed method we have observed that a design with small macros often results in longer inter-macro wiring.

We use a clustering algorithm [7] to group small macros into large ones. We first construct a connected graph directly from the HDL structural tree. For example, Figure 4(b) illustrates a connected graph derived from the structural tree shown in Figure 4(a).

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 is 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 closeness C_{ij} of two macros, v_i and v_j, as below.

$$C_{ij} = \begin{cases} \left(\frac{w(v_i)+w(v_j)}{w(v_i)+w(v_j)-2w_{ij}}\right) \times \left(\frac{s(v_i)+s(v_j)}{s_{th}}\right), & \frac{s(v_i)+s(v_j)}{s_{th}} \leq 1 \\ 0, & \frac{s(v_i)+s(v_j)}{s_{th}} > 1 \end{cases}$$

where

$w(v_i)$ denotes the total connection weight of v_i,
$s(v_i)$ denotes the size of v_i, and
s_{th} is the upper bound on the size of a cluster given by the user.

In order to eliminate the small macros and prevent the formation of large clusters, the user can setup an upper bound on the size of a cluster. When the size of a new macro by merging two macros is larger than the upper bound, the closeness value between these two macros is 0. For example, in Figure 4(b), macro $M8$ can be merged with either $M4$, $M6$, or $M7$. However, if the merging of $M8$ and $M6$ violates the size constraint, $M8$ should be merged with either $M4$ or $M7$ according to their closenesses. The clustering procedure continues until no more macro can be merged.

3.4 Macro placement

After forming macros, we apply a macro-cell placement procedure to reduce the inter-macro wire length. Before performing the macro-cell placement, we estimate the size of each macro cell to be the total size of its constituent primitive cells. We also assume that every macro has an aspect ratio between 0.5 and 2.0.

We use an existing placement program, TW-MC [8], for macro-cell placement. We provide TW-MC with a control file to specify the aspect ratio of the overall placement plane (1.0) and each macro cell (0.5 to 2.0). We observed that a larger or smaller aspect ratio constraint often results in longer inter-macro wiring as well as longer computation time.

3.5 Macro to cell mapping

This task converts a layout in the macro-cell architecture into one in the row-based architecture. The main objective is to preserve as much as possible the topological relationship among the macro cells determined by the macro-cell placement.

We use a straightforward method for the mapping. First, the number of rows R of the row-based

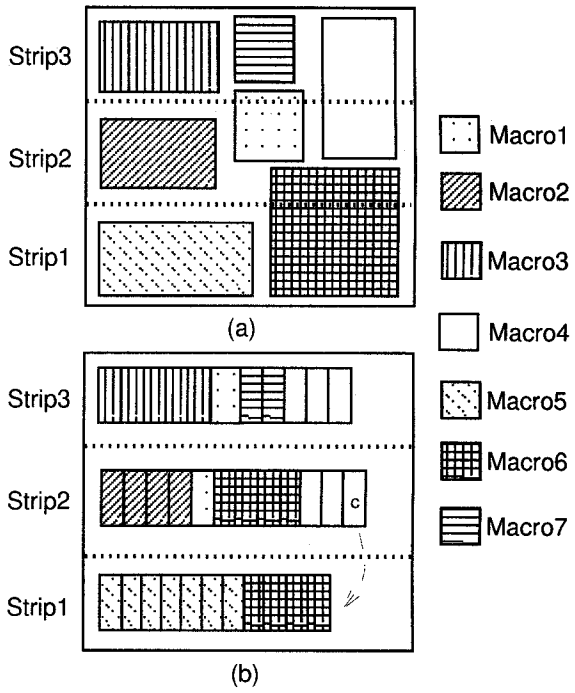

Figure 5: A macro-to-cell mapping example: (a) macro-cell placement, (b) cell placement.

layout is given by the user. Then, the macro-cell layout plane is divided horizontally into R strips. After that, we scan each strip from left to right and sort the encountered macros by the x-coordinate of their centers. Finally, for each row, we assign the cells in each encountered macro to the row from left to right. The number of cells taken from a macro to a strip is proportional to the size of its area within the strip, and the cells are chosen arbitrarily. The resultant rows may be of unequal lengths which will be fixed with a postprocessing procedure.

Figure 5 shows a macro-to-cell mapping example consisting of seven macro cells. Figures 5(a) and (b) depict the resultant macro-cell placement and the corresponding mapped row-based placement without the postprocessing of row-length equalization, respectively. In this example, both *Macros* 5 and 6 intersect *Strip* 1. Therefore, we calculate the number of cells from each intersected macro that are to be placed into the row. As shown in Figure 5(b) four cells from *Macro* 6 are placed into *Strip* 1. The mapping procedure continues until all of the cells are placed into the rows. However, the resultant rows may have unequal lengths as shown in Figure 5(b). The postprocessing procedure will be applied to equalize the row lengths by relocating cells from a long row to a shorter one. In this example, the rightmost cell c from *Macro* 4 in

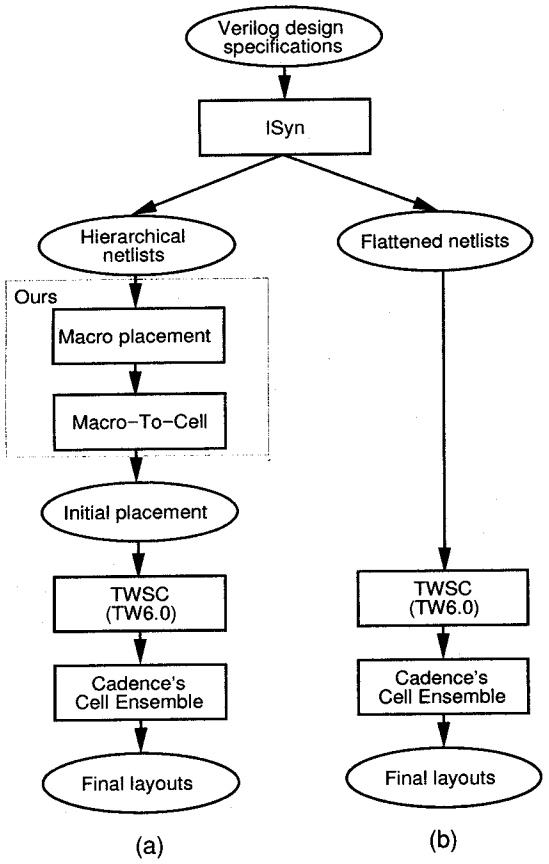

Figure 6: The experimental procedure: (a) our method, (b) the flatten method.

Strip 2 can be relocated to *Strip* 1 to balance the row length.

In the final step, we refine the row-based cell placement using the TW–SC to further reduce the wire length. Using the mapped cell placement as the initial placement, TW–SC refines the row-based cell placement starting with the low temperature annealing.

4 Experiments

We have implemented the proposed system in the C programming language running on a SUN Sparc20 workstation. We have tested the proposed method on five large industry designs. All five designs are described in Verilog. Table 1 shows the characteristics of the designs in which $\#HDL\ Lines$, $\#Mods$, $\#Cells$, $\#Gates$, $\#Nets$, and $\#IOs$ denote the the number of lines of Verilog code, modules, cells, equivalent gates, nets, and IO pins, respectively. In all experiments, we used the TSMC $0.8\mu m$ cell library [9].

In order to examine the effectiveness of our proposed method, we have conducted two sets of experiments. Figure 6 depicts the experimental procedure. In the first experiment, we first used our method to generate an initial placement. Then, we used the TW–SC (*TW*6.0 [8]) to refine it. Finally, we used a commercial tool, *Cadence's Cell Ensemble* [10], to perform routing, compaction and generate the final layout. In the second experiment, we first used the *ISyn* synthesis system to generate a flattened cell-level netlist. Then, we used the TW–SC to perform placement directly on the flattened circuits. Finally, we used the *Cadence's Cell Ensemble* to perform routing and compaction of the designs. Throughout the experiments, we used the same control parameters for the TW–SC and the same aspect ratio of final layouts for each benchmarking circuit. The machine used throughout the experiments is a SUN Sparc20 with 196M of RAM and 300M of swapping space.

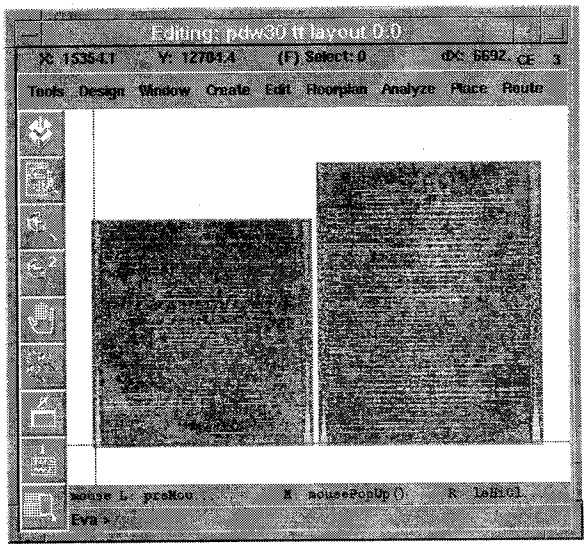

Figure 7: The final layouts of *Ind*3: by the proposed method (left) and by the traditional method (right).

Table 2 shows the area comparisons between the proposed method and the flatten method, in which $W \times H$ and $Area$ denote the width and height of the layouts and the final area. Table 3 shows the wire length and CPU time comparisons, in which WL and CPU denote the total actual wire length after detailed routing and the run time, respectively. Figure 7 shows the final layouts of *Ind*3. It is clear that the proposed approach is able to consistently produce high quality layout using less CPU time.

5 Conclusions

We have presented an integrated HDL-synthesis and placement method for row-based layouts. This study has demonstrated that by bridging the gap between HDL synthesis and placement design tasks, we

Table 1: Characteristics of the benchmarking circuits.

Designs	#HDL Lines	#Mods	#Cells	#Gates	#Nets	#IOs
Ind1	6,059	36	15,566	34,885	10,831	280
Ind2	23,579	28	40,623	64,788	40,623	399
Ind3	8,942	19	17,537	29,076	17,403	307
Ind4	7,456	35	20,987	40,449	20,673	566
Ind5	5,397	33	7,037	11,368	6,719	570

Table 2: The area comparisons between our approach and the flatten method.

Circuits	Flat		Ours		%
	W×H(μm)	Area($10^6 \mu m^2$)	W×H(μm)	Area($10^6 \mu m^2$)	
Ind1	7,940×9,965	79.12	7,690×8,234	63.32	-25.0
Ind2	18,173×23,034	418.60	18,315×20,344	372.60	-12.3
Ind3	8,537×10,926	93.27	8,427×8,681	73.15	-27.5
Ind4	7,921×15,625	123.76	7,432×12,906	95.92	-29.0
Ind5	4,701×8,012	37.66	4,621×6,883	31.80	-18.4

Table 3: The wire length and CPU-time omparisons between our approach and the flatten method.

Circuits	WL($10^6 \mu m$)			CPU(Sec.)	
	Flat	Ours	%	Flat	Ours
Ind1	18.3	16.3	-12.3	30,122	11,351
Ind2	91.1	82.1	-11.0	114,862	29,589
Ind3	20.9	17.5	-19.4	24,727	10,099
Ind4	33.0	25.4	-29.9	36,582	12,923
Ind5	9.1	7.6	-19.7	6,270	3,412

are able to preserve the natural design hierarchy and use it to guide the cell placement process. Experiments on a number of large industry designs have demonstrated that fully exploiting design hierarchy for cell placement lead to significant improvements in layout area, total wire length, and runtime.

This is our first attempt to investigate the interaction between HDL synthesis and placement. There are many open problems that need to be explored further including performance-driven placement methods, timing back-annotation methods, layout-driven HDL synthesis methods, and RTL floorplanning techniques.

References

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

[2] Sujit Dey, Franc Beglez and Gershon Kedem, "Circuit partitioning for logic synthesis," *IEEE Journal of Solid-Stage Circuits*, vol.26, pp.350-363, March 1991.

[3] Gotaro Odawara, Takahisa Hiraide and Osamu Nishina, "Partitioning and placement technique for CMOS gate arrays," *IEEE Trans. on Compter-Aided Design*, vol.CAD-6, pp.355-363, May 1987.

[4] Yu-Wen Tsay and Youn-Long Lin, "A Row-Based Cell Placement Method That Utilizes Circuit Structural Properties," *IEEE Trans. on Compter-Aided Design*, vol.14, no. 3, March 1995.

[5] W.-J. Fang, A C.-H. Wu, and D.-P. Chen, "Module Generation of Complex Macros for Logic Emulation Applications," *Proceedings of the FPGA'97*, 1997.

[6] E. K. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," *U.C. Berkeley Technical Report UCB/ERL M92/41*, May 1992.

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

[8] C. Sechen, "TimberWolf6.0: Mixed macro-standard cell floor planning, placement and routing package," user's manual, Yale University, Sep., 1991.

[9] TSMC450 Library, Taiwan Semeconductor Manufacturing Co. Ltd.

[10] "Cell Ensemble Reference Manual Version 4.3.4", Cadence, 1996.