The Future of Custom Cell Generation in Physical Synthesis

Martin Lefebvre
Cadabra Design Libraries Inc.
One Antares Drive, Suite 300
Nepean, Ont.
K2E 8C4
martinl@cadabratech.com

David Marple
Synopsys Inc.
700 East Middlefield Road
Mountain View, CA
94043-4033
marple@siarc.com

Carl Sechen
University of Washington
Electrical Engineering Department
Box 352500
Seattle, WA 98195-2500
sechen@ee.washington.edu

***Abstract** - We present a subjective review of custom cell generation methods in the context of future advances in state-of-the-art digital circuit synthesis. In particular, we describe three opportunities for coupling circuit optimization operations with the library development process. These operations include electrical optimization, technology mapping, and cell level place and route.*

1. Introduction

Several methods have been developed in recent years for the automatic generation of cell libraries. This effort has been largely motivated by a need for alternatives to manual layout thereby reducing library development costs and time to market. The most common approaches are layout generators, re-compaction of existing libraries, and automatic cell synthesis.

Procedural layout generators, which are either language-based or interfaced with a symbolic layout system, are a means of capturing the layout design in a somewhat design-rule independent fashion. This is useful in reducing the layout creation effort as well as in retargeting existing libraries to new foundry rules. This productivity benefit is mitigated by the fact that each cell must be supported by its own specific generator which must be created by an experienced cell designer. Generators also tend to be unfriendly to drastic changes in cell architecture and interconnect technology.

Compaction of existing layout data provides a somewhat more elegant solution to the cell generation problem since a single generic program can process a wide variety of cells. A significant drawback, however, is that compaction requires the availability of existing layouts substantially the same as the intended cell library as a starting point. Like procedural generators, compactors do not lend themselves well to architectural changes.

Cell synthesis, in contrast to the above methods, is concerned with creating cell layouts starting with only a transistor level netlist for each cell. It is completely flexible in terms of a target library architecture and does not require any pre-existing cell-specific layout information. It is, however, a non-trivial problem to automate the creation of cell layouts competitive with handcrafted quality layouts. To date, the sheer complexity of the cell synthesis process has been the main impediment to its commercial success.

Whereas procedural layout generators and compactors can be used to speed up the library generation process, this paper advocates the use of cell layout synthesis as the dominant method. This is based on the overall flexibility that layout synthesis provides particularly in the context of future advances in the state of the art in digital circuit synthesis. Indeed the goal of any synthesis technique, whether it is behavioral, logic, or layout, is to ultimately produce optimum circuits that meet the designer's specifications. By optimum circuit, we mean an implementation where the designer and/or synthesis tool achieves the best tradeoff of design metrics for the entire circuit. The most common metrics for digital CMOS circuits are delay (e.g. input to output propagation, clock period), silicon area, and dynamic power. Static power may also be important for circuits powered by batteries. Metrics specific to layout synthesis include layout porosity, manufacturing yield, and circuit reliability (e.g. noise margins, latch up immunity).

The realization of any custom circuit, whether it be a standard cell or a complete block, requires three primary implementation phases:

1. Creation of a transistor circuit topology that provides a specific digital function. By topology we mean the allocation of N and P type transistors (or other devices) and their connectivity.

2. Sizing and ordering the transistors in the circuit topology. Sizing means altering the MOS transistor's width and ordering means swapping positions of MOS transistors connecting in series chains or swapping logical pins.

3. Placing, routing and compacting the above transistors into a layout.

All of the above phases involve tradeoffs of design metrics which must be optimized not only within each phase but also across all phases. To date only phase 3 is automated by most layout synthesis tools.

The coupling between circuit optimization, such as behavioral, logic or layout, is the key to better utilization of integrated circuit technology resulting in better products.

This paper describes opportunities for circuit optimization across design phases. Following a review of state-of-the-art cell synthesis techniques in Section 2, we identify three specific opportunities for expanding cell layout synthesis technology. The first is the integration of cell generation and transistor sizing and re-ordering for the purpose of electrical optimization (Section 3). The second is the coupling between cell generation and the technology mapping phase of logic synthesis, effectively replacing the concept of a cell library by a mechanism for generating cells on-demand (Section 4). The third is a reduction of silicon area made possible by a tighter coupling between cell generation and the automatic place and route system (Section 5).

2. Review of Cell Synthesis

Currently cell synthesis consists of mapping a set of sized transistor netlists into cell layouts based on a target cell architecture. In addition to a netlist for each cell, the inputs to a cell synthesis system include the design rules for the target process and a description of the architecture and layout guidelines. In general, a single architecture description can be used for a complete cell library. The output of a cell synthesis system is a collection of cell layouts which result from the successive solution of the three sub-problems of placement, routing and compaction.

2.1 Transistor Placement

Placement is the problem of determining the relative position of each transistor within the target cell architecture. A common strategy is to maximize the number of connections that may be made between the devices through common diffusions and vertically aligned gate wires. Transistor placement may also be subject to physical constraints such as keeping the order of IO ports, minimizing the span of critical nets and accommodating a variety of transistor sizes. Furthermore, provisions should be made to place substrate contacts and allow space for feed-through nets.

A reasonably large number of notable contributions have been made over the last fifteen years to the field of transistor placement [1,2,3,4,5,6,7,8]. Generally, transistor placement methods combine the following operations:

1. Transistor pairing: p-type and n-type transistors are paired if they have common gate and/or source/drain signal names or if they are associated with the same gate level structure;

2. Group/chain formation: more compact layouts result when forming groups or chains of series connected transistors which lend themselves to a locally optimal layout. Such groups can result from heuristic algorithms, chaining algorithms, partitioning, or even branch-and-bound optimization.

3. Transistor folding: large transistors may need to be decomposed into a set of smaller ones connected in parallel. This is the case when transistor sizes exceed the height (width) provided by the target architecture or when large imbalances between large and small transistors would result in a sub-optimal solution.

4. Group/chain placement: final placement may be accomplished by a variety of classical placement methods either constructive or iterative. Classical constructive placement methods include branch-and-bound and min-cut partitioning. A classical iterative method is simulated annealing. The objective of either placement method is to maximize the value of some merit function which reflects the “goodness” of the cell in terms of measurable attributes such as anticipated layout area and wire length metrics.

2.2 Routing

Leaf cell routing has been addressed by a number of different methods. Most reported systems use classical routing algorithms such as maze, line-probe, river routing and the left edge algorithm. Several researchers reported using a two phase approach as follows:

1. Routing over the transistors using special purpose algorithms to implement horizontal connections between source/drain diffusions within the same transistor row.

2. River/channel routing methods to complete all the remaining connections in the center region of the cell between the p- and n-style rows of transistors.

Routers based on the above combination [2] generally rely on some net classification scheme based on the location of terminals associated with each net.

General-purpose area or maze routers with obstacles have also been reported. Maze routers tend to be more flexible in terms of layout style and circuit structure but are more difficult to fine tune to obtain good results in a cell synthesis system. In [1] Poirier describes the use of the A-star procedure which is an improvement over Lee maze routing based on directing the search preferentially in the direction of the target subnet.

In addition, some routers complete routing by automatically inserting either vertical and/or horizontal routing tracks on the fly [1]. This track insertion can be triggered by detection of an unfeasible solution or a time out.

2.3 Compaction

Compaction is the process of generating compact and error free layout from an initial symbolic or generic layout. Compaction is also used to transform a layout from another, usually older, technology to a new target technology with different design rules. Automatic compactors for cells have been around for a long time, most of which compact in only one direction at a time [10].

A somewhat dated review of compactors can be found in [9]. Some researchers have reported on two dimensional compactors [11], where the width and height of a cell are compacted at the same time. Compactors that usually produce the best quality results are based on a constraint graph model, where nodes in the graph represent layout elements or polygon edges and arcs in the graph represent layout design rule and/or connectivity constraints amongst the elements/edges.

Compaction methods usually require the following operations:

1. Constructing a constraint graph from an initial layout: A compactor performs this step by first constructing the graph nodes after scanning all layout geometry, then constructing the graph arcs by again scanning all layout geometry and registering possible design rule or electrical interactions of neighboring geometry.

2. Solve for the critical (longest) path(s) in the graph: Algorithms applied are topological sort for simple constraint graphs [10], or linear programming methods [11] for two dimensional constraint graphs.

3. Insert jogs on wires linked to critical nodes (elements): This process helps reduce the cell's width sometimes at the expense of the cell's height, or vice-versa. The algorithms used to insert jogs are usually heuristic.

4. Minimize wire lengths for wires not associated with critical nodes: This step prevents wires from being stretched by the compaction process and ensures the resistance of each wire is minimized. Graph theoretic [10] or linear programming methods [11] can be applied for this problem.

Once compaction is complete, the layout is usually design rule correct by construction. In some rare cases, the final layout may be overconstrained and some violation may remain, which the compactor will report to the user.

2.4 Other Issues

Several key issues must be considered in order to synthesize leaf cells of hand-crafted character. In particular, the architecture specification must be extremely flexible in order to accommodate the wide variety of design styles mandated by different technologies and circuit applications. A partial list of architectural features include:

- position and size of power supply rails
- nature and spacing of substrate and well ties
- boundary constraints on each design layer
- size constraints and alignment rules for wells
- size and offset of I/O routing grid in each dimension
- constraints on the overall cell size (both height and width)
- policy governing the insertion of I/O pins
- policy governing the use of multiple diffusion contacts for source/drain connections
- policy governing the use of non-Manhattan geometries
- policy governing the insertion on jogs and bends in wires
- individual routing policies associated with each routing layer

In addition to a flexible architecture specification, the system must also provide user control over selected layout features which have a direct impact on circuit performance. These include minimization of diffusion regions, frequency of diffusion strapping, choice of routing layers for critical nets, etc., and ability to produce results which are friendly to the leading place-and-route systems. The latter requires full control over pin location, shape and layer assignment, as well as various cell porosity requirements.

3. Coupling with Electrical Optimization

Choosing the best widths of MOS transistors, called transistor sizing, can improve circuit performance. Given a transistor circuit, several researchers have automated transistor sizing and today some commercial tools can perform transistor sizing automatically. The approaches used in [13,14,15,16] as well as several others, rely on modelling the transistor-sizing problem as a nonlinear program and solving the program with well-known nonlinear programming algorithms. Other approaches to transistor sizing rely on using heuristics [12], but do not achieve the quality of results as a nonlinear programming approach [16].

The nonlinear program formulations for optimizing transistor sizes usually take one of two forms:

1. minimize: D_{max}
   subject to:

$$\sum_j d_j(\overline{w}) \le D_{max}$$

$$\sum_i w_i = W_{max}$$

2. minimize: W_{max}
   subject to:

$$\sum_j d_j(\overline{w}) \le D_{max}$$

$$\sum_i w_i = W_{max}$$

In the above formulations, D_{max} represents the worst case path delay, W_{max} the circuit active area or dynamic power, w the size of a transistor, and d(w) the delay of a logic gate as a continuous nonlinear function of w. Some researchers consider only a single worst case path in the formulation while others consider multiple worst-case paths [13]. Also, some researchers [13,16] have suggested better area models than the sum of transistor sizes (W_{max}) to reflect the structure of the target layout realization. The algorithms used to solve either formulation 1 or 2 include Lagrangian methods, such as Augmented Lagrangian [13,14,16], or curve fitting methods, such as Levenberg-Marquardt [15].

The researchers in [17] automated transistor reordering and demonstrated the benefits of ordering on speed critical paths. They showed that the optimum order requires the consideration of the complete electrical environment of the circuit, specifically the input rise/fall transition rates. These researchers used a heuristic approach to optimize transistor order and mention a method for including transistor sizing into their heuristic. To date, no one has integrated transistor ordering with a nonlinear programming approach to transistor sizing. It may be possible to model both transistor ordering and sizing with a single nonlinear programming model, but this has yet to be publicly demonstrated.

The work in [13] shows a combination transistor sizing and layout compaction as part of a custom layout entry system called Tailor. Marple modelled the combination sizing and compaction problem as a general nonlinear programming problem, with a linear objective (cell width or height), linear layout constraints (design rules between layout edges), and nonlinear delay constraints (worst case input to output delays). The circuit delays were continuous functions of the transistor sizes (difference in layout edges), the layout parasitics, and external electrical environment (input rise/fall transitions and output loading). Rather than solving the complete nonlinear program with a general nonlinear programming algorithm, Marple solved the problem by employing a longest path compaction algorithm to reduce the number of linear constraints and then employing an Augmented Lagrangian algorithm to solve the reduced nonlinear program. This two step process would then repeat until the tool satisfied all layout constraints and delay constraints and fulfilled the conditions for an optimum circuit.

Marple's research on combination transistor sizing and compaction could be extended to address the broader requirements of a general layout synthesis tool. Firstly, a more accurate and flexible delay model for the circuit is needed to avoid verification with Spice (or like simulator) and to handle sequential circuits. Secondly, a single area objective (width times height) should be used which would involve the integration of a two dimensional compactor with transistor sizing. Thirdly, transistor placement, folding (creating parallel transistors), and routing could be integrated with transistor sizing/compaction to provide an automated iterative procedure to converge on a overall optimum layout realization. Finally, transistor ordering could be simultaneously handled with sizing and compaction.

4. Coupling with Logic Synthesis

Logic synthesis has become the dominant method for designing logic circuits. Technology mapping is the final phase of logic synthesis which consists of mapping an optimized technology independent logic network into a circuit realization in a given technology. Most commercial technology mapping systems require that all available physical components be represented explicitly in a pattern library. Sections of the logic network are then matched against this pattern library to identify suitable cells to implement the circuit. The size of the cell library may range from a few dozen cells up to several thousand and is generally governed by manpower restrictions, limitations/capabilities of the logic synthesis program and/or limitations/capabilities of the remaining design flow to silicon. Cell synthesis is an enabling technology which paves the way for significantly larger cell libraries than are feasible today. Cell synthesis may also obsolete pre-defined libraries with cell generation on-demand as part of the technology mapping process.

Some work on cell synthesis-based technology mapping has revealed some promise of exploiting large cell sets. Experiments with constructive matching [23], for example, have demonstrated some benefits, albeit modest, in both area and speed when targeting very large libraries. A more recent study [19] has shown that the area benefits of large cell sets increase as the number of interconnect layers increases.

This is in addition to the obvious power savings which would result if each non-critical path on a chip could be down-graded by reducing transistor sizes or selecting alternative cells so as to meet all timing requirements exactly. This power reduction could even take place post-layout once very accurate loading information is available for each cell.

5. Coupling with Placement and Routing

A cell library generation tool must be keenly aware of the nuances of placement and routing. One key issue is that the fundamental algorithms behind all of the leading commercial routers are grid based and a cell's pins must align on a routing grid in both the x and y direction. While it is true that some of these commercial routers can handle off-grid pins, their algorithms are inherently gridded and perform considerably better if the pins are all on-grid.

In a typical cell layout style, the first metal layer ("metal 1") is used to interconnect the various transistors comprising the cells. Use of the second metal layer ("metal 2") is generally avoided except for possibly the most complex cells, such as flip-flops. The lack of blockages on metal 2 ensures metal 2 can be well utilized for vertical routing. Depending on the cell layout architecture, it may be a good idea to bring the input and output pins out to metal 2. Allowing some other signal on metal 2 to pass directly over a pin left on metal 1 (or poly) may severely impact routability, in that metal 1 (or poly) is largely exhausted in interconnecting the cells and thus the router may find that the pin is inaccessible. If all input and output pins are brought out to metal 2 in the cell library, pin accessibility is almost guaranteed during routing. As long as blockages (*i.e.* prior uses of metal 2) are minimized, it will always be possible to route vertically straight up or down from a pin.

Minimizing blockages on metal 2 and bringing the pins out to metal 2 or allowing multiple metal 2 access to the pins are excellent steps to take to enhance routability. However, another crucial issue is the vertical porosity of the cells. If a metal 2 pin appears on most metal 2 routing grids, a common situation for nonfolded combinational cells, the vertical porosity will be severely compromised if the pins appear on a horizontal line.

If a cell library is designed without any implicit feedthroughs in the cells, the width of a standard cell block placed and routed using these cells (with no vertical porosity) will be typically 25 to 40 percent wider than a comparable library with sufficient implicit feedthroughs.

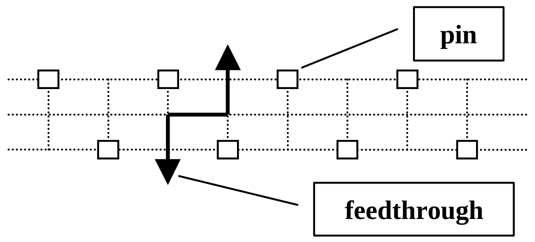

Figure 1. Staggered pin locations can be used to create implicit feedthroughs.

One way to increase vertical porosity and effectively the number of implicit feedthroughs, is to stagger the pin locations in the *y* direction as shown in Fig. 1. The pins are typically centered between the p and n diffusions. By using three vertical tracks (or pitches) for the pin locations and placing the pins alternately in the top or bottom track, a router can sneak a "dogleg" feedthrough over the cell. If there are *N* pins on a cell, an upper bound of *N*/2 implicit feedthroughs may be available. However, as illustrated in Fig. 2, it can happen, and often does, that the connection to a pin placed *a priori* in the bottom (top) row must be routed to a row above (below). In the worst case, no implicit feedthroughs are available. Even a moderate number of unfortunately directed connections can result in very few implicit feedthroughs as shown in Fig. 3. For *a priori* pin placement, since the directions of the connections are random, experiments have shown that the typical number of implicit feeds available is usually well short of the theoretical upper bound. Typically this causes the insertion of quite a number of actual feedthrough cells, which adds to the width of a placed and routed standard cell block.

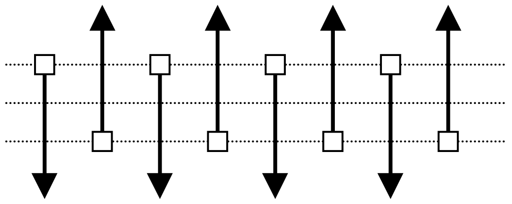
Figure 2. No implicit feeds are available even though pins are staggered.

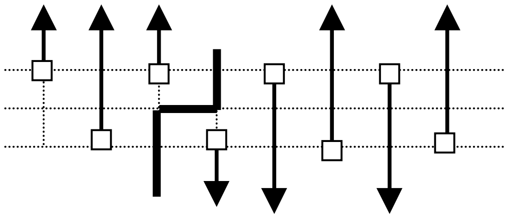
Figure 3. Illustrated here is a typical case in which one implicit feedthrough is available.

A very effective way to increase implicit feedthroughs is to provide each pin with two possible locations on the same metal 2(x) grid, one in the top of the three tracks and one in the bottom track, as illustrated in Fig. 4. When routing the top (bottom) pin location toward the row above (below), additional implicit feedthroughs are generated. For example in Fig. 3 only one implicit feed is available, but in Fig. 4, two additional implicit feeds are available due to the flexibility of pin location.

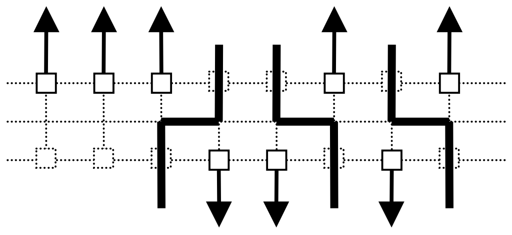
Figure 4. If the global router can "move" the pins, three implicit feeds result. The dashed squares indicate alternative pin locations.

To demonstrate the impact of having two such locations for each pin, we augmented the TimberWolf router [18] to select the appropriate pin location during its optimization process. The results for five MCNC benchmark circuits are shown in Table 1. Notice that 2-4 times more implicit feedthroughs are available than the number required by the router. Past experimentation has revealed that any time the number of implicit feeds is at least double the number required, then very dense routing will be achieved without having to add any explicit feedthrough cells. In fact, we have yet to encounter a random logic netlist for which explicit feedthrough cells have to be added by TimberWolf when a pair of locations for each pin is provided as described above.

Circuit	No. of Cells	Feeds Required	Imp. Feeds Avail.
ex2	170	37	171
x1	177	38	116
apex6	406	21	317
C7552	1206	250	783
C6288	1731	500	999

Table 1. Feeds required is the number of row crossings desired by the router summed over all rows, and similarly implicit feeds available is summed over all rows. The number of implicit feeds exceeds the number required by a factor of 2-4x, permitting efficient routing.

An automatic cell generator could be used effectively with a place and route system by providing cell layouts with a wide variety of pin orderings. For example, all of the possible orderings, subject to a limit on the number of diffusion breaks, can be listed [19,21,22]. To minimize area, one might ask the cell generator to produce all possible pin orderings corresponding to cells of minimum width. Then, for a given instance of this cell in the layout, we can select the particular pin ordering which yields the least total wire length and/or wire area. Complex cells, such as flip-flops, or cells including folded transistors, tend to have very large numbers of possible pin orderings subject to a minimum number of diffusion breaks. Even nonfolded combinational cells tend to have at least a few possible orderings.

We examined the effectiveness of pin-ordering optimization on the same MCNC benchmark circuits shown in Table 1. We iteratively examined all of the cell instances in congested areas on the chip (i.e. cells bordering channels of peak or near peak density) and selected a legal pin ordering (i.e. which minimizes diffusion breaks) for each instance which minimized the total Steiner wire length during global routing.

Circuit	Orig. Pin Order #Tracks	Reordered Pins #Tracks	Track Reduction (%)
ex2	47	40	14.9
x1	63	55	12.7
apex6	75	68	9.3
C7552	177	160	9.6
C6288	191	177	7.3
Average			10.8

Table 2. The number of horizontal routing racks used by TimberWolf following pin-ordering optimization is reduced by 10.8% on average.

As the results in Table 2 show, about an 11% decrease in horizontal routing tracks was achieved, on average. A comparable decrease in total wire length was observed. While such area and wire length savings are smaller than those achieved by selecting pin locations during routing to maximize implicit feedthroughs, they are nonetheless significant.

A cell generator for combinational cells addressing the routability issues described in this section has been reported [19,20,21,22].

6. Summary

As covered in this paper, cell synthesis involves the deployment and integration of many design automation techniques to produce high quality layouts from an initial transistor level or low level boolean description. Key techniques for cell synthesis include transistor sizing, transistor/pin ordering, transistor/pin placement, transistor routing, and layout compaction. The commercial application of digital cell synthesis primarily is targeted to the creation of standard cell libraries used for traditional ASIC system design. This not only requires consideration of the target ASIC technology design rules, but also the desired cell architecture, I/O pin strategy, and tradeoffs of layout structures. Emerging applications of cell synthesis include automatic technology migration of cell libraries as well as on the fly cell creation as part of logic synthesis/technology mapping, cell-based place and route, and buffer optimization after place and route. These applications will become manifest with the future research and development of cell synthesis technology.

References

[1] Poirier C.J., "Excellerator: custom CMOS leaf cell layout generator", *IEEE Trans. On Computer-Aided Design*, July 1989, Vol. 8, No. 7, pp. 744-755.

[2] Ong C.-L., Li J.-T., and Lo C.-Y., "GENAC: An automatic cell synthesis tool", *IEEE 26th Design Automation Conference*, July 1989, pp. 239-244.

[3] Chen C.C. and Chow S.-L., "The layout synthesizer: an automatic netlist-to-layout system", *IEEE 26th Design Automation Conference*, June 1989, pp. 232-238.

[4] Wimer S., Pinter R.Y., and Feldman J., "Optimal chaining of CMOS transistors in functional cells," *IEEE Trans. On Computer-Aided Design*, September 1987, Vol. 6, No. 5, pp. 795-801.

[5] Maziasz R.L. and Hayes J.P., "Layout minimization of CMOS cells", *Kluwer Acadamic Publishers*, Boston MA, 1992.

[6] Hwang C.Y., Hsieh Y.-C., Lin Y.-L. and Hsu Y.-C., "An efficient layout style for two-metal CMOS leaf cells and its automatic synthesis", *IEEE Transactions on Computer-Aided Design*, March 1993, Vol. 12, No. 3, pp. 410-424.

[7] Baltus D.G., and Allen J., "SOLO: A generator for efficient layouts from optimized MOS circuit schematics", *IEEE 25th Design Automation Conference*, June 1988, pp. 445-453.

[8] Domic A., and Shiple T., "CLEO: A CMOS Layout Generator", *IEEE International Conference on Computer-Aided Design*, 1989, pp. 340-343

[9] Boyer D., "Symbolic Layout Compaction Review", *Proceedings of the 25th Design Automation Conference*, June 1988, pp. 383-389.

[10] Marple D., Smulders M., and Hegen H., "An Efficient Compactor for 45 Degree Layout", *Proceedings of the 25th Design Automation Conference*, June 1988, pp. 396-402.

[11] Shigehiro Y., Nagata T., Shirakawa I., and Kambe T., "Optimal Layout Recycling Based on Graph Theoretic Linear Programming Approach", *Proceedings of the VLSI*, 1994, paper 1.2.1.

[12] Fishburn J. and Dunlop A., TILOS: "A Polynomial Programming Approach to Transistor Sizing", *Proc. 1985 International Conference on Computer-Aided Design*, November 1985, pp. 326-328.

[13] Marple D., "Transistor Size Optimization in the Tailor Layout System", *Proceedings of the 26th Design Automation Conference*, June 1989, pp. 43-48.

[14] Conn A., Coulman P., Haring R., Morrill G., and Visweswariah C., "Optimization of Custom MOS Circuits by Transistor Sizing", *1996 International Conference on Computer-Aided Design*, November 1996, pp. 174-180.

[15] Menezes N., Pullela S., and Pelegii L., "Simultaneous Gate and Interconnect Sizing for Circuit-Level Delay Optimization", *Proceedings of the 32nd Design Automation Conference*, June 1995, pp. 690-695.

[16] Obermeier F. and Katz R., "An Electrical Optimizer that Considers Physical Layout", *Proceedings of the 25th Design Automation Conference*, June 1988, pp. 453-459.

[17] Carlson B. and Chen C.Y., "Performance Enhancement of CMOS VLSI Circuits by Transistor Reordering", *Proceedings of the 30th Design Automation Conference*, June 1993, pp. 361-366.

[18] TimberWolf: Mixed Macro/Standard Cell Floorplanning, Placement and Routing Package (Version 1.3). TimberWolf Systems, Inc., Dallas, TX, 1997.

[19] Guan B. and Sechen C., "Large Standard Cell Libraries and Their Impact on Layout Area and Circuit Performance," *Procedures IEEE International Conference on Computer Design (ICCD)*, October 7-9, 1996, Austin, TX.

[20] Guan B. and Sechen C., "Efficient Standard Cell Generation When Diffusion Strapping Is Required," *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, May 1996, San Diego, CA.

[21] Guan B. and Sechen C., "An Area Minimizing Layout Generator for Random Logic Blocks," *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, May 1-4, 1995, Santa Clara, CA.

[22] Guan B., *Automatic Layout Generation of Static CMOS Combinational Cells and Blocks*, Ph.D. Dissertation, University of Washington, Seattle, August 1996.

[23] Lefebvre M., and Liem C., "Cell Generator-Based Technology Mapping by Constructive Tree-Matching and Dynamic Covering", *VLSI Design*, 1995, Vol. 3, No. 1, pp. 1-12.