An Exact Solution to Simultaneous Technology Mapping and Linear Placement Problem*

Jinan Lou, Amir H. Salek, Massoud Pedram
Department of Electrical Engineering – Systems
University of Southern California, Los Angeles, CA 90089

Abstract

In this paper, we present an optimal algorithm for solving the simultaneous technology mapping and linear placement problem for tree-structured circuits with the objective of minimizing the post-layout area. The proposed algorithm relies on generation of gate-area versus cut-width curves using a dynamic programming approach. A novel design flow, which extends this algorithm to minimize the circuit delay and handle general DAG structures, is also presented. Experimental results on MCNC benchmarks are reported.

I. Introduction

In a top-down design flow, optimizations at each step are made without considering their impacts on the subsequent design steps. For example, technology mapping algorithms minimize gate area or delay without considering the interconnect structure and routing overhead of the resulting circuit. This lack of information reduces the effectiveness of the optimization techniques. Existing enhancements to synthesis and physical design tools, such as better delay and wire load models, post-layout re-mapping, and gate resizing have not been able to solve this problem. This means that designers have to resort to costly design iterations. This problem is compounded as feature sizes shrink to quarter micron and below because interconnect delay and area become even more dominant, the number of gates in the circuit increases, and the number of nets that are at a performance risk rises rapidly. It is therefore necessary to develop new design methodologies and tools to cope with the problem.

This paper proposes a solution by introducing a new algorithm for simultaneous technology mapping and linear placement of tree-structured circuits. This algorithm combined with a novel floorplan-driven design flow leads to a highly effective approach for performing technology mapping and cell placement for general circuits, hence combing the two design steps.

This paper is organized as follows: In Section II, we give the background knowledge for technology mapping and linear placement algorithms. Section III presents details of our algorithm for combining technology mapping and linear placement for trees. We introduce a design flow which uses this algorithm to simultaneously map and place general circuits and minimize circuit delay in section IV. Experimental results and concluding remarks are given in Sections V and VI, respectively.

II. Background

The problem of technology mapping for general circuit structures is NP-hard [5]. In 1987 Keutzer [7] pointed out the similarity between the library binding problem and the optimal code generation in a compiler. In his algorithm the circuit is partitioned into tree sub-graphs and each tree sub-graph is mapped using a dynamic programming algorithm which finds the minimum gate area mapping of the tree in polynomial time. This work was later extended by Rudell [14] to minimum delay technology mapping and by Touati et al. [19] to minimize area mapping under delay constraints. In [1], Chaudhary and Pedram presented a dynamic programming algorithm to construct the set of all possible mappings of a tree with different area-delay trade-off.

Neither of the above-mentioned works considers wiring area or delay during technology mapping. This was the motivation for Pedram and Bhat's work in [11] to couple technology mapping and placement in order to consider the effect of wires during mapping. Their proposed algorithm assumes that the dynamic programming principle holds during the bottom-up process of concurrent mapping and placement. This is however only an approximation.

For the sake of completeness, we give the following (well-known) result:

Lemma 1: Keutzer's algorithm produces the best gate area mapping of a tree [7].

The linear placement problem of a graph has been extensively studied. The MINSUM problem is to find a linear placement that minimizes the total length of connecting wire segments. On the other hand, the MINCUT problem is to find a linear placement that minimizes the maximum cut-width. Both problems are NP-hard for the general graphs [21]. In 1979, Shiloach [18] gave an $O(n^{2.2})$ algorithm to solve the MINSUM problem for trees, and in 1984 Chung [2] improved it to $O(n^{1.58})$. For MINCUT problems, Lengauer [9] introduced a polynomial time algorithm for trees whose cost is within a factor of two of the optimal in 1982, and

* This work was supported in part by NFS under contract No. MIP-94/57392, and by grants from IBM Corp. and ViewLogic Inc.

Yannakakis [23] gave an O(n*log*n) dynamic programming algorithm to find the optimal solution for trees in 1985. In this paper, we exploit Yannakakis' algorithm for the MINCUT problem.

Lemma 2: Yannakakis' algorithm produces the minimum cut width linear placement of a tree [21].

III. SiMPA

In this section, we introduce our Simultaneous Technology Mapping and Linear Placement Algorithm (SiMPA) which finds the minimum total area (gate plus routing) implementation of a tree.

III.1. Problem Formulation

PROBLEM: *Given a library L and a tree T, find a simultaneous technology mapping and linear placement solution for T which has the smallest total area (gate plus routing area).*

Keutzer's algorithm (KA) finds the minimum gate area technology mapping for a tree. KA cannot account for the wiring area since it has no knowledge about the gate positions. Yannakakis' algorithm (YA) finds the minimum cut-width linear placement of a tree. What remains for us to do is to combine the two algorithms.

Lemma 3: The following equation gives the exact total area for a one-dimensional standard-cell layout (cf. Figure 1):

$$A = a + \frac{\beta}{h} \cdot (a \cdot c)$$

where $a = W \cdot h$ is the total gate area, W is sum of the cell widths, h is the cell height (assumed to be a constant), β is the minimum distance between the centers of two adjacent wires, and c is the maximum cut width (also referred to as the cut-density) [†].

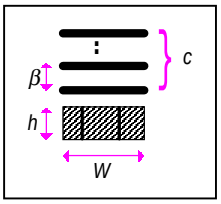

Figure 1. Total area calculation

The above equation is exact since in one-dimensional routing, there are no vertical constraints and the channel height is set by the maximum clique size in the horizontal constraint graph [17]. Clearly, a is determined from the technology mapping whereas c is determined from the placement. To find the minimum total area A, technology mapping and placement must be done simultaneously.

III.2. Gate-area versus cut-width curve

During KA, mapping solutions which have larger area than the best solution (found up to that point) are dropped. However, this notation of inferiority does not hold for simultaneous technology mapping and linear placement. The following counter example shows how this can happen.

Example: Assume at one step of dynamic programming, when solving the problem of minimizing the total area of a tree, there are two sub-problems, T_1 and T_2. For T_1 one design S_1, and for T_2 two designs $S_{2,1}$ and $S_{2,2}$, have been found (Figure 2), and $totalArea(S_{2,1}) < totalArea(S_{2,2})$, that is, $S_{2,2}$ appears to be inferior to $S_{2,1}$. However, figure 3 shows that the combination of S_1 and $S_{2,1}$ is inferior to the combination of S_1 and $S_{2,2}$. Therefore, we cannot simply throw away $S_{2,2}$. That is, we cannot drop a solution simply because it has a larger total area than that of the best solution found so far.

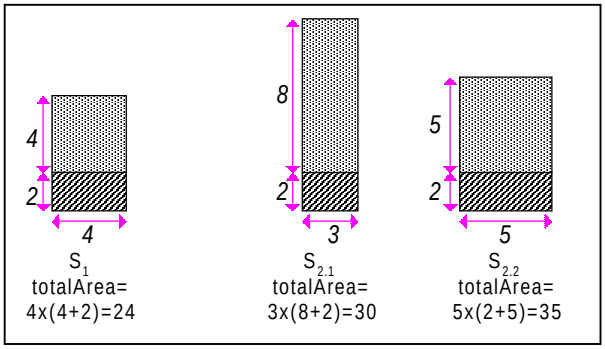

Figure 2. Example before combining sub-solutions

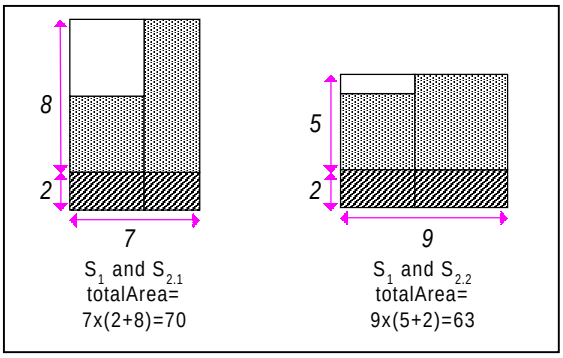

Figure 3. Example after combining sub-solutions

Definition: S_1 is inferior to S_2 if and only if the following conditions hold:
$gateArea(S_1) >= gateArea(S_2)$ **and**
$cutWidth(S_1) >= cutWidth(S_2)$

Our algorithm is to maintain all non-inferior <gateArea, cutWidth> points for each node in the tree. SiMPA thus combines KA and YA; its pseudo code is given next:

1. Technology decompose circuit N
2. Perform a Depth-First-Search (DFS) on N
3. For each node n in the reverse-DFS order do
4. For every match m of n do
5. gateArea = sum of gateAreas of all inputs of this match + gate area of this match
6. cutWidth = the cut width from linear placement of the mapped tree rooted at m
7. Store the solution <gateArea, cutWidth> in n
8. Prune inferior solutions from the curve of n
9. Select the best solution by starting from the primary output, and recursively finding the solutions for all inputs of this best solution

[†] Proofs can be found in [10].

For each gate from the library that matches a set of nodes rooted at *n*, we calculate the gate area and the cut width of this match. The current match and the inputs to this match are stored with the gate area and cut width, so that later (i.e. during step 9) we can construct the best solutions for the inputs. Note that step 6 is performed using YA which itself uses dynamic programming. That is, we build the optimal linear placement of the problem at hand from optimal placement of its constituent sub-problems. After all matches are enumerated and all solutions are generated, we prune out the inferior solutions and only keep the non-inferior solutions in node *n*. After reaching the tree root, the best solution is selected from the curve of the root. Since we stored the match and inputs of *n*, we can then construct the best solutions for all of its inputs recursively.

In the example discussed in section III.1, since $gateArea(S_{2,1}) < gateArea(S_{2,2})$ but $cutWidth(S_{2,1}) > cutWidth(S_{2,2})$, we must keep both solutions as non-inferior solutions. In effect, we are generating a gate-area versus cut-width curve (Figure 4) at every node. The lower left corner points on the curve are non-inferior points with respect to each other. We will then generate the curve on the next higher level of the tree by combining the curves from the lower level curves and by pruning the inferior points.

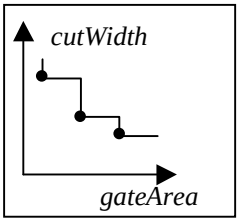

Figure 4. A gateArea versus cutWidth curve

Consider tree *T* with root *n*. Let *S* represent the mapping solution for *T* with match *m* at the root. Let a_i denote the gate area of the mapping solution S_i of child *i* of *m*. Let *c* denote the cut-width of the *S* obtained by YA.

Lemma 4: $$totalArea(S) = \sum_i a_i + \frac{\beta}{h}\left(c \cdot \sum_i a_i\right)$$

Consider two solutions S_1 and S_2. For tree *T*, these solutions are the same except that S_1 contains sub-solution $S_{i,1}$ and S_2 contains sub-solution $S_{i,2}$.

Lemma 5: If $S_{i,1}$ is inferior to $S_{i,2}$, then S_1 is inferior to S_2.

Theorem 1: The final curve generated by SiMPA at the root of tree *T* includes all non-inferior gate area versus cut width solutions for the simultaneous technology mapping and linear placement problem of *T*.

Corollary 1: Given an upper bound on the number of tracks, *maxCut*, the points in the gate-area versus cut-width curve with the cut width less than *maxCut* constitute all the non-inferior implementations of the circuit satisfying that constraint.

Corollary 2: Given an upper bound on the area, *maxArea*, the points in the gate-area versus cut-width curve with the gate area less than *maxArea* constitute all the non-inferior implementations of the circuit satisfying that constraint.

Corollaries 1 and 2 show how SiMPA can be used in applications such as layout-driven logic synthesis and post-layout re-synthesis. Given a cut width constraint, we can find the best gate area implementation which satisfies this constraint. Similarly, given a gate area constraint, we can find the best cut width implementation which meets this constraint.

Theorem 2: The solution SiMPA chooses in the last step (step 9), is the solution with the minimum total area.

IV. FPD-SiMPA

In this section, we present our floorplan driven design flow (FPD-SiMPA) which extends SiMPA to minimize the circuit delay and to handle general DAG structures. This new flow partitions the DAG into a set of trees, floorplans them, uses the floorplanning and global routing results to generate delay budgets for trees, and finally uses a timing-driven version of SiMPA to pick a timing correct solution for each tree.

IV.1 Extension to Delay Minimization

There are many different models for the gate delay calculation. In SIS [16], the gate delay is calculated using this equation for both rise and fall delays:

$$GateDelay = K_1 + K_2 \cdot load$$

where K_1 and K_2 are the intrinsic gate delay and the fanout-dependent load delay, respectively. This delay equation is pin-dependent, uses a linear model, and assumes step input signals. It is simple but rather inaccurate (especially in 0.5µ design and below).

We extend the SIS delay model to a four-parameter delay model where we add the effect of the input slew rate:

$$GateDelay = (K_1 + K_2 \cdot load) + slew \cdot (K_3 + K_4 \cdot load)$$

where K_1 to K_4 are constants. We use the same equation form but obviously with different coefficients to calculate the rise and fall delays, as well as the rise and fall slew rates for gate outputs. Our delay equation is also pin-dependent. We get these constants by doing curve fittings for every library cell based on the results from extensive circuit-level simulation using HSpice.

For wire delay calculation, the lumped RC model is widely used, but the Elmore delay model gives higher accuracy [4]. The latter model calculates the delay of each wire segment using the equation shown in Figure 5:

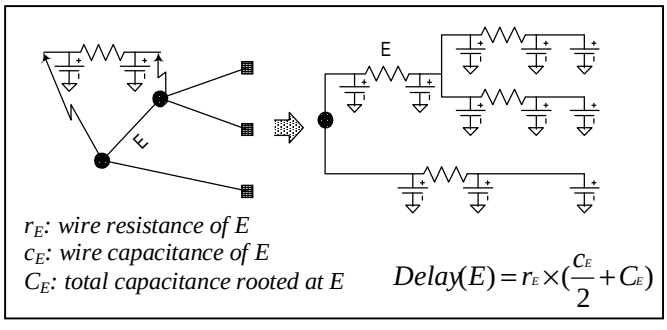

Figure 5. Elmore delay model

The delay minimization cannot be done exactly due to the following problems. During each step of dynamic programming, the mapping and placement information for the fanouts of the current node are not known. Therefore, the load of the current node cannot be accurately determined. Moreover, the delay of the wire cannot be calculated exactly since the sub-trees may be broken up and separated during placement of an ancestor node in the tree by YA. To solve the first problem, we use a heuristic similar to that in [14,19] to do delay calculation. During each step of the dynamic programming procedure, we assume a constant load ahead (that, is the input capacitance of a 2-input NAND gate and the capacitance of a wire segment whose length is statistically estimated based on the fanout count of the node [20]). After a match is found, we update the delay of all fanins of this match to use the correct load information since the match is henceforth known and the fanins are already placed and mapped. This is similar to the timing recalculation step of [19]. The second problem mentioned above introduces some errors in the delay calculation which cannot be helped.

As pointed out in section III.2, we maintain a gate area versus cut-width curve at each node. Adding delay to SiMPA means we must generate and store a three-dimensional curve (Figure 6) in each node during dynamic programming. The three dimensions are gate area, cut width and delay. Pruning is again performed to ensure the polynomial running time of the algorithm. However, note that the delay values in this curve are not exact as explained before.

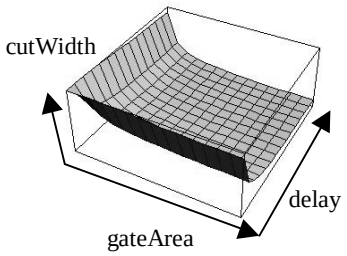

Figure 6 A three-dimensional curve

IV.2 Extension to DAGs

Similar to KA, SiMPA can be used as a basic routine for mapping and placement of general DAG-structured circuits as follows.

As a first step, we partition the original circuit into a set of trees. Restriction to tree partitioning may sacrifice the overall quality of the technology mapping, but we benefit from the exactness of SiMPA. Therefore, the overall chip area/delay can still be improved (see Section V). We treat each tree as a block (macro-cell). Because we target standard cell layout, the height of each block is the same as the height of the cells in the standard cell library. Next, a minimum gate-area mapping of each tree is performed to get an estimation of the width of the block. With the dimensions of the blocks known, we floorplan these blocks and route the global wires that are connecting them. The delay of each block is estimated from the number of logic levels in each block (which is obtained from the initial mapping solution), and the load of the block. We calculate the global wire delays using the Elmore delay model. Since the physical location of each block is known, the global wire delays are fairly accurate. Next, the delay budget for each block (tree) is obtained using a technique similar to that of [6]. We use the budget as the constraint to pick up a timing feasible solution from the set of non-inferior solutions generated by SiMPA for each tree. Finally, we refine the floorplan solution to eliminate overlaps, and generate the final chip layout. The pseudo code is given blow.

1. *Cluster the original circuit into a set of trees*
2. *Estimate the area of each tree*
3. *Treat each tree as a macro-cell and do floorplanning*
4. *Perform global routing on wires that connect trees*
5. *Do delay calculation and budgeting for each tree*
6. *Call SiMPA on each tree*
7. *Use the delay budget generated in step 5 to pick the feasible solution from the set of all non-inferior solutions generated by SiMPA*
8. *Eliminate block overlaps and complete detailed routing*

V. Experimental Results

We present our experimental results in two tables. The area and delay in both tables refer to the total chip area and the total chip delay after detailed routing. In all cases, we use the four-parameter delay equation (see Section IV.1) to calculate the gate delays and the Elmore delay model to calculate the wire delays. We use a CASCADE generated standard cell library (0.5µ HP 14B CMOS process) to produce these results.

In Table I we present the results of SiMPA on trees and compare them with the results obtained by the conventional flow of performing technology mapping and linear placement separately. It is important to mention here that in order to keep the comparison fair, we also use Yannakakis' MINCUT placement algorithm to do the linear placement in the conventional flow. Otherwise, we would have obtained larger improvements because other placement tools, such as Gordian [8] or TimberWolf [15] do not generate the minimum cut width placement. The suffix of the tree name represents the number of inputs. Our results show an average improvement of 20% in the total post-layout area.

In Table II we present the results using a conventional flow and FPD-SiMPA. In the conventional flow, we use SIS to do a minimum delay technology mapping, followed by Gordian and Domino [3] for placement, TimberWolf for global routing and YACR [13] for detailed routing. In FPD-SiMPA, we use tree clustering and initial mapping of each tree for block width estimation, followed by Bear-FP [12] which does floorplanning and global routing. Then we call SiMPA for each tree subject to the delay budgets generated in the floorplanning stage. Domino is used to eliminate overlaps while TimberWolf is used to improve the global routing solution. Finally YACR is called to do the detailed routing. Our results show an average 21% improvement of the delay without any area penalty.

	Conventional			SiMPA			
	Gate Area	*	Total Area	Gate Area	*	Total Area	Ratio
tree8	64438	5	369054	62986	4	288590	0.78
tree10	94604	5	541823	94916	4	434888	0.80
tree20	30536	4	139910	30536	3	104933	0.75
tree24	97468	5	558226	100437	4	460184	0.82
tree64	74701	6	513400	77616	5	444528	0.87
							0.80

Table I Tree results (* column is the number of tracks)

	Conventional		FPD-SiMPA			
	Area	Delay	Area	Delay	Area Ratio	Delay Ratio
apex6	4212505	12.94	3753360	9.40	0.89	0.73
b12	422831	4.34	412191	3.40	0.97	0.78
b9	563030	4.00	667318	3.18	1.19	0.80
C17	40050	0.74	40762	0.38	1.02	0.51
C1908	4031118	16.99	3487815	12.86	0.87	0.76
C3540	9803363	28.86	7897600	22.49	0.81	0.78
C432	1747635	12.80	1362132	15.25	0.78	1.19
C7552	12631941	24.53	15747725	22.67	1.25	0.92
C880	2352900	10.00	2378453	9.46	1.01	0.95
cm150a	216952	4.10	207928	1.65	0.96	0.40
cm151a	117725	2.28	116477	1.44	0.99	0.63
cm162a	246800	2.13	201912	2.03	0.82	0.95
con1	74690	1.58	100441	0.88	1.34	0.56
cordic	398763	2.42	311535	2.04	0.78	0.84
dalu	9818750	25.90	8593920	25.30	0.88	0.98
frg1	689931	3.71	689142	3.18	1.00	0.86
lal	553268	4.79	613118	4.50	1.11	0.94
med	166453	3.48	142105	2.72	0.85	0.78
misex3c	2670030	20.67	2747928	19.07	1.03	0.92
pcle	365859	3.87	391327	3.92	1.07	1.01
pcler8	536095	5.43	518830	4.25	0.97	0.78
rd53	224296	3.90	222800	2.20	0.99	0.56
rd73	387907	5.59	348119	5.21	0.90	0.93
rd84	888122	8.75	771825	6.92	0.87	0.79
ritex1	91133	1.16	90025	0.84	0.99	0.72
squar5	327627	6.84	278800	5.40	0.85	0.79
vg2	546742	3.90	501599	3.35	0.92	0.86
xor5	121009	2.64	106945	1.37	0.88	0.52
z4ml	203600	3.97	202032	2.89	0.99	0.73
					0.96	0.79

Table II Benchmark results

VI. Conclusion

In this paper we proposed an algorithm for performing technology mapping and linear placement simultaneously in order to capture the information related to the wiring area during the mapping. This technique uses dynamic programming and has polynomial time complexity. We also presented a new design flow which uses SiMPA and a partitioning / floorplanning step to handle general circuit structures. Furthermore, we extended SiMPA to generate three-dimensional trade-off curves for minimum delay mapping, although delay calculation is not exact. Our future work will focus on developing an improved timing driven SiMPA.

VII. Reference

[1] K. Chaudhary, and M. Pedram, "A near-optimal algorithm for technology mapping minimizing area under delay constraints," in *Proceeding 29th Design Automation Conference*, June 1992.

[2] F. R. K. Chung, "On optimal linear arrangements of trees," in *Comput. Math. Applic*. 10, pages 43-60, 1984.

[3] K. Doll, F. M. Johannes, and G. Sigl, "DOMINO: Deterministic placement improvement with hill-climbing capabilities," in *IFIP International Conference VLSI*, 1991.

[4] R. Gupta, B. Krauter, B. Tutuianu, and T. Pileggi, "The Elmore Delay as a Bound for RC Trees with Generalized Input Signals," in *Proceedings of 32nd Design Automation Conference*, pages 364-369, 1995.

[5] G. D. Hachtel, and F. Somenzi, "Logic synthesis and verification algorithm," *Kluwer Academic Publishers*, Norwell, MA, 1996.

[6] P.S. Hauge, R. Nair, and E. J. Yoffa, "Circuit placement for predictable performance," in *Proceedings of IEEE Inter. Conference of Computer-Aided Design*, Nov. 1987, pp.88-91.

[7] K. Keutzer, "DAGON: Technology mapping and local optimization," *In Proceedings of Design Automation Conference*, pages 341-347, June 1987.

[8] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GRODIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," in *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 3, March, 1991.

[9] T. Lengauer, "Upper and lower bounds on the complexity of the min-cut linear arrangement problem on trees," in *SIAM J. Alg. Disc. Meth.*, vol. 3, no. 1, pages 99-113, March 1982.

[10] J. Lou and A. H. Salek, "Optimal Algorithm for Simultaneous Technology Mapping and Linear Placement," *CENG Technical Report*, Electrical Engineering-Systems, University of Southern California, July 1997.

[11] M. Pedram, and N. Bhat, "Layout driven technology mapping," *In Proceedings of the 28th Design Automation Conference*, pages 99-105, June 1991.

[12] M. Pedram, and E. Kuh, "BEAR-FP: A Robust Framework for Floorplanning," in *International Journal of High Speed Electronics*, Vol. 3, No.1, pages 137-170, 1992.

[13] J. Reed, A. Sangiovanni-Vincentelli, and M. Santamauro, "A new symbolic channel router: YACR2," in *IEEE Trans. on CAD*, Vol. CAD-4, pages 208-219, March 1985.

[14] R. Rudell, "Logic synthesis for VLSI design," *Memorandum UCB/ERL M89/49*, Ph.D. Dissertation, University of California at Berkeley, April 1989.

[15] C. Sechen, and A. Sangiovanni-Vincentelli, "TimberWolf3.2: A new standard cell placement and global routing package," in *Proc. of 23rd Design Automation Conf.*, pages 432-439, 1986.

[16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis", *Memorandum No. UCB/ERL M92/41*, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992.

[17] N. Sherwani, "Algorithms for VLSI Physical Design Automation," *Kluwer Academic Publishers*, 1993.

[18] Y. Shiloach, "A minimum linear arrangement algorithm for undirected trees," in *SIAM J. Comput*. 8, pages 15-32, 1979.

[19] H. J. Touati, C. W. Moon, R. K. Brayton, and A. Wang, "Performance oriented technology mapping," in *Proc. of Sixth MIT Conference Advanced Res. VLSI*, pages 79-97, April 1990.

[20] H. Vaishnav, and M. Pedram, "Logic extraction based on normalized netlengths," in *Proceedings of the International Conference on Computer Design*, October 1995.

[21] M. Yannakakis, "A polynomial algorithm for the min-cut linear arrangement of trees," *in Journal of the Association for Computing Machinery*, vol. 32, no. 4, pages 950-988, October 1985.