Technology Mapping Using Logical Effort for Solving the Load-Distribution Problem

Shrirang K. Karandikar and Sachin S. Sapatnekar, *Fellow, IEEE*

***Abstract*—Technology mapping is a crucial step in the synthesis of digital designs and can be used to obtain mapped circuits that are optimized for delay or area. Current tree-based mapping algorithms break the circuit into individual trees and map these optimally. However, these solutions are not globally optimal. This paper presents a new approach to delay-optimal mapping based on the principle of logical effort. This algorithm maps individual trees such that the solution of the entire circuit is optimal. In traditional technology mapping, the best match for a gate depends on the load being driven, which is not known at the matching stage. Current algorithms handle this situation by generating matches for all loads and selecting the best match at a later stage. This strategy works for fan-out-free circuits but breaks down at multiple fan-out points where each fan-out has to be sized correctly, depending on its criticality. This can have a significant impact on the selection of matches as well but has not been adequately addressed in the published literature. We refer to the correct sizing of branches of multiple fan-out points as the load-distribution problem, which is formally defined and solved in the context of technology mapping in this paper. The effect of the new logical effort-based mapping algorithm, combined with correct sizing of individual branches of a multiple fan-out point, leads to implementations that are closer to the global optimum. On the average, benchmark circuits mapped using our approach are 39.45% faster and 32.77% smaller than those obtained using SIS.**

***Index Terms*—Algorithms, circuit synthesis, CMOS digital integrated circuits, combinational logic circuits, design automation, high-level synthesis, very-large-scale integration.**

I. Introduction

THE CONVERSION of a register-transfer level description of a design into an implementation in silicon starts with logic synthesis, which consists of technology independent optimization, followed by technology mapping. In the latter step, the design is mapped to cells belonging to the target library while optimizing one or more performance metrics, such as delay, area, or power. High-performance designs use rich libraries, with multiple instances of each cell, which have various delay, area, and drive capabilities. Technology mapping has to identify not only the best logic functionalities of cells to be used to implement some logic but also the best instance of each selected cell.

Manuscript received June 26, 2006; revised February 28, 2007. This work was supported in part by the Semiconductor Research Corporation under Contract 2001-TJ-884 and in part by the National Science Foundation under Award CCR-0205227. This paper was recommended by Associate Editor I. Bahar.

S. K. Karandikar is with the Computational Research Laboratories, Pune 411 016, India (e-mail: shrirang@crlindia.com).

S. S. Sapatnekar is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455-0213 USA.

Digital Object Identifier 10.1109/TCAD.2007.907067

This paper addresses the problem of delay-optimal technology mapping, for which a number of algorithms have been proposed in the past, such as tree-mapping [1] and DAG-mapping [2], using load-dependent delay models [3], constant delay models [4], [5] as well as using logical effort [6]. Recent approaches also integrate technology mapping with physical design [7] and logic optimization [8]–[10]. In this paper, the current state of delay-optimal technology mapping is extended in two directions.

The first contribution of this paper is a new logical-effort-based technology mapping algorithm. Logical effort [11], [12] has been widely used in a variety of application domains [5], [13]–[15] as well as in industry standard EDA synthesis tools [16], [17]. The method of logical effort primarily provides a quick means of estimating the delay of a path of logic but has significant drawbacks when analyzing entire circuits having gates with multiple fan-outs. This technique also provides a means of calculating the gate sizes that lead to the estimated minimum delay. Consequently, given multiple implementations of the same path of logic and input and output capacitances for the path, logical effort can be used to easily determine the minimum delay implementation. This notion can be used to constructively map a path of logic to the minimum delay implementation, in a manner similar to traditional mapping algorithms, by enumerating choices and eliminating suboptimal ones. This is the idea behind the new technology mapping algorithms presented in this paper. Our approach, described in Section IV, has a few advantages over previous methods. First, unlike conventional methods, the selection of gate sizes in the solution is implicit and does not have to be determined during matching. Second, the delay model is inherently load dependent, and there is no need to enumerate solutions for all possible load values as is done in the traditional mapping approach [3]. Finally, the size of the library used is much smaller since each gate need not be instantiated for each available size, leading to faster matches. Combined, these features make our algorithm faster than current algorithms for fan-out-free circuits.

The second contribution of this paper is the formulation and solution of the "load-distribution problem," which, to the best of our knowledge, has not been formally addressed in the literature previously. Traditional technology mapping approaches partition a circuit into fan-out-free trees and map each tree separately. Every gate within such a tree drives one other gate that is also contained in the tree. The output gate of the tree drives either a primary output or gates that are input gates of other trees. Within a tree, traditional technology mapping approaches recognize the fact that the delay-optimal match of a gate, and its corresponding size, depend on the load being

driven. This is true for the output gate of the tree as well, which drives multiple fan-outs. Selecting the correct solution for this output gate is crucial since this solution, in turn, determines the load for other gates in the tree. However, the load being driven by the output gate of a tree is not known in advance. Additionally, indiscriminately selecting the best solution for each path in a tree can lead to the input gates of the tree having large sizes. While this may lead to each tree in the circuit having the best input-to-output delay, such a solution for the complete circuit is severely suboptimal. What is required is an approach that takes into account the criticality of each component of multiple fan-out points and selects solutions that optimize the delay of the entire circuit. The problem of assigning correct sizes (or, equivalently, capacitances) to gates at multiple fan-out points is referred to as the load-distribution problem and is described in greater detail in Section III-B. A solution to this issue in the context of gate sizing has been presented in [18]–[20] and is extended to the technology mapping arena in this paper.

II. Background

We first present a brief overview of the method of logical effort and how it applies to sizing a path for minimum delay. This is followed by a discussion of algorithms that are currently used for technology mapping and the delay models used in these algorithms.

A. Logical Effort and Gate Sizing

In the method of logical effort, the delay of a gate is estimated by modeling it as a linear function of the load being driven as

$$D = g \times \frac{c_l}{c_i} + p = g \times h + p = f + p \tag{1}$$

where g is the logical effort, $h = C_L/c_i$ is the electrical effort, $f = gh$ is the effort delay and p is the parasitic delay of the gate.

This formulation separates the different components that contribute to the delay of a gate. More importantly, it leads to a natural extension for estimating the minimum delay, $\hat{D}$, of a path of logic as

$$\hat{D} = NF^{1/N} + P \tag{2}$$

where $F = GH$ is referred to as the path effort, P as the path parasitic delay, and N as the number of gates on the path under consideration [12]. The path logical effort, G, is the product of the logical efforts of the gates on the path, and the path electrical effort, H, is the product of the gate electrical efforts. The minimum delay of (2) is obtained by distributing the path effort F equally to each gate on the path.

The path electrical effort can also be calculated as the ratio of output and input capacitances of the path. Consider Fig. 1, which shows a simple path of four gates—A, B, C, and D. Each of these gates have input capacitances $c_{\mathrm{in_A}}$, $c_{\mathrm{in_B}}$, $c_{\mathrm{in_C}}$, and $c_{\mathrm{in_D}}$ and drive output capacitances $c_{\mathrm{out_A}}$, $c_{\mathrm{out_B}}$, $c_{\mathrm{out_C}}$, and $c_{\mathrm{out_D}}$, respectively. The input capacitance of the path c_{in} is the input capacitance of gate A, and the output capacitance of the path C_L

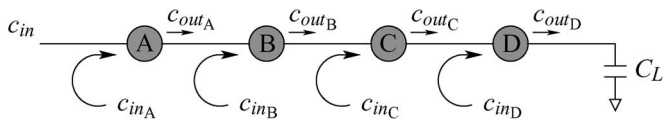

Fig. 1. Calculating the electrical effort of a path.

is the output capacitance of gate D. The path electrical effort, H, the product of the gate electrical efforts, telescopes, since the input capacitance of each gate is the load capacitance of its input (e.g., $c_{\mathrm{in_C}} = c_{\mathrm{out_B}}$). Thus

$$\begin{aligned} H &= \frac{c_{\mathrm{out_A}}}{c_{\mathrm{in_A}}} \times \frac{c_{\mathrm{out_B}}}{c_{\mathrm{in_B}}} \times \frac{c_{\mathrm{out_C}}}{c_{\mathrm{in_C}}} \times \frac{c_{\mathrm{out_D}}}{c_{\mathrm{in_D}}} \\ &= \frac{c_{\mathrm{out_A}}}{c_{\mathrm{in}}} \times \frac{c_{\mathrm{out_B}}}{c_{\mathrm{in_B}}} \times \frac{c_{\mathrm{out_C}}}{c_{\mathrm{in_C}}} \times \frac{C_L}{c_{\mathrm{in_D}}} \\ &= \frac{c_{\mathrm{out_A}}}{c_{\mathrm{in}}} \times \frac{c_{\mathrm{out_B}}}{c_{\mathrm{in_B}}} \times \frac{c_{\mathrm{out_C}}}{c_{\mathrm{in_C}}} \times \frac{C_L}{c_{\mathrm{in_D}}} = \frac{C_L}{c_{\mathrm{in}}}. \end{aligned} \tag{3}$$

The traditional logical effort approach is well suited for estimating the minimum delay that can be achieved by sizing a path of logic [using (2)] if the electrical effort, H, of the path is known. The individual gate sizes that are required to achieve this minimum delay can be calculated as follows: Each gate is assigned a gate effort of $f = F^{1/N}$. Starting with the gate at the output that drives a known load of C_L, the size of each gate is successively determined. Since the logical effort g of a gate is fixed, if an effort delay f is assigned to a gate, the input capacitance c_{in} that meets this effort delay can be calculated as

$$c_{\mathrm{in}} = \frac{g \times c_l}{f} \tag{4}$$

where c_l is the load begin driven by the gate under consideration.

In general, the input capacitance of the kth gate from the output, c_{in_k}, can be calculated as

$$c_{\mathrm{in}_k} = \frac{\prod_{i=1}^{k} g_i}{f^k} \cdot C_L. \tag{5}$$

Further additions to deal with multiple fan-outs in circuits have been presented in [12]. However, as discussed in [20], the "branching effort" has significant drawbacks due to the lack of *a priori* knowledge of the input and load capacitances of individual branches in a circuit. These drawbacks can create inaccuracies when calculating the minimum achievable delay of a circuit and the corresponding gate sizes, as discussed in greater detail in Section III-B.

B. Traditional Technology Mapping

We now briefly summarize the state of technology mapping and the delay models used in these algorithms. Cell- or library-based technology mapping is the process of binding a technology-independent logic level description of a circuit to a library of gates in the target technology. A dynamic-programming algorithm based on tree covering was proposed

in [1] and has served as the basis of later technology mapping algorithms. This is a two-step algorithm.

1) In the matching step, matches for all gates are generated in an input-to-output traversal of the circuit, and the optimum match (based on its cost and the cost at its inputs) and the corresponding matches at the inputs are stored as the solution for that gate.
2) In the covering step, the solution for the entire circuit is generated by an output-to-input traversal of the circuit. At the primary outputs, the best match is selected, and the covering recurses on the inputs of this match.

The delay of a match is a function of the load it is driving—a quantity that is not known during the matching step. In order to account for this in delay-optimal technology mapping, sets of solutions are stored at each gate, each solution being the optimal one for a specific load value. In the covering step, the load is known, and the corresponding optimal match can be selected.

One of the drawbacks of this approach is that the circuit to be mapped (represented by a "subject graph") is partitioned into disjoint fan-out-free trees, which are then optimally mapped. However, this leads to restrictions on the solutions, since matches cannot cross tree boundaries. In [2], it was pointed out that, if duplication at tree boundaries were to be allowed, DAG mapping, as opposed to tree mapping, would provide optimal results. However, this paper allocates a fixed load-independent delay to each gate and assumes that later gate sizing and buffer tree insertion can achieve the delay assigned. Thus, it does not address the load-distribution problem described shortly.

A number of different delay models are used in technology mapping, such as load-independent, load-dependent, constant delay models [4], [5] and gain-based delay models [6]. These approaches try to account for the fact that gate delays depend on the load being driven. While technology mapping makes use of these approaches to generate optimal solutions locally, the global picture is left unfinished, i.e., while the selection of the gate types and gate sizes may be optimal within fan-out-free trees, the traditional algorithms degenerate into heuristic or greedy approaches at multi-fan-out points. This is illustrated in Section III.

III. Drawbacks of Traditional Methods

In traditional tree-mapping methods, the input circuit, represented as a DAG, is partitioned at multi-fan-out points into trees, which are then mapped optimally. These algorithms address the fact that size and functionality selection of matches have a significant impact on the quality of the final mapped solution. However, there are a few issues that are not taken into account. First, the selection of the optimal match and the corresponding size is based on the load being driven and ignores any constraints on the input capacitance of the tree. If this input capacitance is bounded, the best solution may be different from the one selected, as we show in Section III-A. Typically, bounds on the input capacitance are needed so that the driving gates do not see an unnecessarily large load. The second issue is related–rather than an arbitrarily bound on the input capacitances of a tree (which are multifan-out points in the original circuit), an optimal assignment of capacitances to all fan-outs and the driving gate, based on their respective criticalities, can lead to superior solutions, as discussed in Section III-B. Finally, optimally mapping trees need not lead to optimal solutions for the entire circuit, as shown in Section III-C.

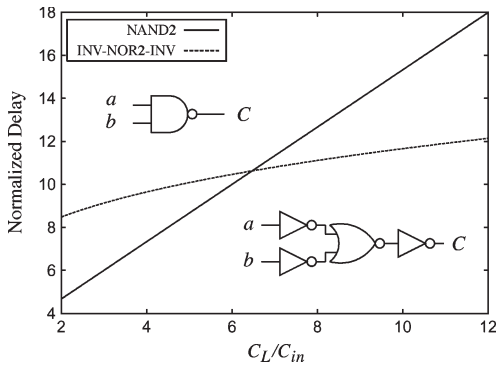

Fig. 2. Influence of load on solutions.

A. Load Dependence of Optimal Matches

Consider Fig. 2, where output C is the NAND of two inputs, a and b. The load at the output of C is C_L, and the input capacitance is a fixed $C_{\rm in}$. This functionality can be obtained by either selecting a NAND2 gate directly, as shown on the top, or by selecting an INV–NOR2–INV chain as shown at the bottom. It may seem that the smaller solution will outperform the larger one. However, consider the delay equations for each option, assuming the following values: $g_{\rm INV} = 1$, $g_{\rm NAND2} = 4/3$, $g_{\rm NOR2} = 5/3$, $p_{\rm INV} = 1$, and $p_{\rm NAND2} = p_{\rm NOR2} = 2$. The minimum delay that can be achieved by each option can be calculated using (2), where the number of stages, N, is 1 for the NAND2 solution and 3 for the INV–NOR2–INV solution

$$\hat{D}_{\rm NAND2} = \frac{4}{3} \times \frac{C_L}{C_{\rm in}} + 2$$

$$\hat{D}_{\rm INV-NOR2-INV} = 3 \cdot \left[\frac{5}{3} \times \frac{C_L}{C_{\rm in}} \right]^{\frac{1}{3}} + 4. \qquad (6)$$

Fig. 2 also plots the minimum delay of (6) as a function of the electrical effort $C_L/C_{\rm in}$. It is obvious that there is no universally better choice—for small values of electrical effort, the NAND2 has lower delay, while the INV–NOR2–INV is better for larger values of electrical effort. Thus, input and output capacitance, rather than the output capacitance only, determine the optimal match, a point that is largely ignored by the traditional technology mapping algorithms. Typically, bigger gates are faster than smaller ones but have a correspondingly higher input capacitance. In order to ensure that the driving gates at the tree inputs do not see an excessive load, limits on the maximum input capacitance may be enforced. However, these limits are not taken into account when selecting the optimal match.

This example serves to highlight a crucial point—in order to determine optimal matches, we need to know the output as well as input capacitances of logic under consideration. To the best of our knowledge, this aspect of technology mapping has not been taken into account previously. However, it is implicit

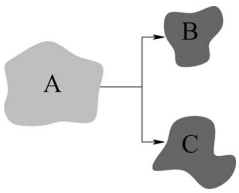

Fig. 3. Assigning capacitance at multiple fan-outs for optimizing circuit delay.

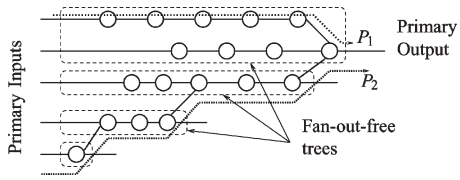

Fig. 4. Critical path in a tree and in the circuit.

in our formulation of logical-effort-based technology mapping and in the load-distribution problem and its solution.

B. Load-Distribution Problem

As seen in the previous section, the optimal solution for a fan-out-free tree of logic depends on input as well as output capacitance. The natural question that arises is as follows: in the context of the entire circuit, what are the values of input and output capacitances of the component trees that minimize the delay of the entire circuit? This issue is analyzed in this section.

Consider the situation shown in Fig. 3, with a block of logic A fanning out to two other blocks, B and C, which eventually drive primary outputs. Each of A, B, and C is a fan-out-free region of the circuit. The optimal solution selected for A depends directly on the load being driven at its output, which, in this case, is the input capacitance of B and C. There are two situations that have to be considered.

1) *The interaction between A and its outputs.* Assigning a larger input capacitance to B and C makes them faster, at the cost of increasing the load on A and slowing it down, and vice versa. What is the optimum value of capacitance that should be assigned to the output of A so that the delay of the entire circuit is minimized?
2) *The interaction between B and C.* The delays of these two fan-out-free regions to the primary outputs of the circuit are influenced by their constituent logic and respective input and output capacitances. If the two blocks of logic have very different delays, we would like the critical branch to have a larger input capacitance. On the other hand, if B and C have the same delay, they should have the same input capacitance. Thus, even if we could determine the optimal load that A should be driving, what is the best distribution of this capacitance to each fan-out?

We refer to these two problems together as the load-distribution problem. Given a load at a multiple fan-out point in the circuit, current algorithms can determine the best mapping for the logic up to that point. However, this load is typically estimated using heuristics, and since the mapped solution depends directly on the load being driven, wrong estimates can lead to suboptimal solutions.

We solve the load-distribution problem by integrating the approach suggested in [20] with technology mapping. This enables us to accurately determine the optimal load that should be driven at a multiple fan-out point and how this load should be distributed, in the form of input capacitance, to each fan-out. Once this load has been calculated (as against being estimated), we can use our technology mapping approach to map the circuit.

C. Critical Paths of Trees and of the Entire Circuit

When dealing with fan-out-free regions, or trees, in isolation, it is easy to determine the critical input of the tree—this is the tree input that has the maximum delay to the tree output. The path from this critical input to the tree output is the critical path of the tree. Note that the critical path of the mapped circuit may be significantly different from the critical path of the unmapped circuit, depending on the target library. The critical path of the mapped tree is not known during the matching phase and may change depending on the choice of the matches made and the corresponding sizes selected. However, traditional tree mapping algorithms can map trees so that the delay on the critical path of the mapped tree is minimized.

Now, consider the situation when the tree is part of a bigger circuit. In this case, the desired mapped solution is the one that minimizes the delay of the critical path of the circuit, which may not correspond to the critical path of the constituent trees. This is shown in Fig. 4, which shows a part of a circuit that has been divided into fan-out-free trees. Each of these trees fans out to multiple outputs or to primary outputs. In the topmost tree, path P_1 is the path with longest delay. However, the critical path of the circuit is path P_2, which traverses multiple trees. As in the case of trees, the critical path of the unmapped circuit can be very different from the critical path of the mapped circuit and therefore cannot be used to determine the critical inputs of individual trees.

Traditional technology mapping algorithms can estimate solutions (optimal matches and sizes) at tree boundaries [21], but these solutions are based on the delays estimated at each fan-in of the current tree under consideration, and they do not take into account the effect of the rest of the circuit. Thus, these estimates can turn out to be in error, leading to globally suboptimal solutions. In such a scenario, optimally mapping individual trees and connecting these mapped trees together can lead to solutions for circuits that are suboptimal. We address this issue in Section IV-C3 by generating solutions for each input of a fan-out-free segment and selecting the solution corresponding to the critical input, once it is known, so that the delay through the entire circuit is minimized.

IV. Logical Effort-Based Technology Mapping

In this section, we present our approach to mapping a circuit to a target library, which addresses the drawbacks of traditional approaches presented in the previous section. We first present a logical-effort-based technology mapping algorithm for fan-out-free circuits. This algorithm is modified in Section IV-B to handle the fact that the critical path in the circuit is not known during the matching step. A key feature of this algorithm is

that the solutions generated are functions of input and output capacitances. This allows us to address the load-distribution problem, which uses the concept of delay–$C_{\rm in}$ curves and their application to dealing with multifan-out points, presented in Section IV-C. The combination of our logical effort-based technology mapping algorithm and delay–$C_{\rm in}$ curves leads to our approach for mapping entire circuits optimally, as presented in Section IV-D.

A. Optimal Technology Mapping for Fan-Out-Free Circuits

We first present our algorithm for a simple path of logic, with each gate having a single input and single output. We then show how this algorithm can be extended to fan-out-free circuits, with each gate having multiple inputs but a single output. This algorithm is optimal for trees, but as described in Section III-C, this optimality may not extend to entire circuits. We present a modified version of the matching step, which, when used in conjunction with delay–$C_{\rm in}$ curves, leads to solutions closer to the global optimum.

In traditional technology mapping, in an input-to-output traversal, all possible matches are generated at each gate. However, only one match (the optimal match, as determined by the cost function), has to be stored for every gate. For minimum-delay technology mapping, the cost function at a gate is the delay of any path from a primary input to the output of the gate and can be calculated as the sum of the delay of the match itself and the maximum delay at its inputs. This approach fits the classical dynamic programming paradigm—the delay of the path is optimal only if the delay of a subpath is optimal; hence, all nonoptimal matches can be discarded. When gate sizing is taken into consideration, a set of solutions, each corresponding to different possible load values, have to be evaluated and stored, rather than a single solution.

Our approach uses the cumulative path logical effort, G, as the cost function. Recall that G is the product of all individual gate logical efforts[1] g on any path from a primary input to the gate under consideration. We will show shortly that, if the number of gates on a path is taken into account, minimizing G is equivalent to minimizing the delay of that path. This formulation also fits into the optimal substructure property of dynamic programming, since by definition, the minimum value of cumulative logical effort of a path is obtained when the cumulative logical effort of any subpath is minimum. This formulation has a couple of advantages over previous methods. First, different load values are automatically accounted for, and therefore, only one solution has to be stored for each gate. Second, the solution for a path is a function of the input and output capacitances of that path, as shown later in this section. This leads to globally optimal solutions when extended for complete circuits.

[1]Here, "gate logical effort" refers to the logical effort of individual matches.

We now show how minimizing path logical effort G leads to minimum-delay solutions. Recall (2), which can be expanded as

$$\hat{D} = N(GH)^{1/N} + P \tag{7}$$

where $H = C_L/c_{\rm in}$.

Given a path having N stages of logic, path logical effort G and path electrical effort H, the minimum delay in (7) can be obtained by sizing gates on the path appropriately. If we were to have the freedom of replacing gates on the path with functionally equivalent choices, while maintaining the path length and electrical effort, it is obvious that the optimal solution after gate sizing is obtained when the path logical effort G is minimized. Thus, in an input-to-output traversal of a path, selecting the match that minimizes the cumulative logical effort for a certain path length will lead to a solution that minimizes the delay of the mapped path after sizing. The parasitic delay P can be used as a secondary criterion to break ties when we have solutions with equal path logical effort.

The length of the path can vary depending on the matches selected. At each gate, we store a set of optimal matches for different path lengths. Correspondingly, at the primary output, we obtain a set of solutions of different path lengths and different values of cumulative logical effort. We can then use (7) to determine the combination of path length N, cumulative logical effort G, and parasitic delay P that will minimize the delay of the mapped circuit after sizing.

Our logical effort-based approach to technology mapping, for a simple path, with a known input and output capacitance, can be summarized as follows.

1) In the matching step, traverse the path from the primary input to the primary output. For each match at a gate, the cost function is computed as the product of the logical effort of the match and the cumulative logical effort at the input of the match. The length of the path is the length of the input of the match plus 1. For all path lengths, store the best match.
2) At the primary output, determine the combination of G, P, and N that will minimize the delay after sizing, as calculated by (7).
3) In the covering step, traverse the path from the primary output to the primary input, generating the solution as in regular technology mapping. In addition, calculate the correct sizes of each gate using (5).

The above description was restricted to simple paths of logic where each gate has a single fan-in and a single fan-out. We can now generalize this approach to circuits with gates having multiple fan-ins (the case with multiple fan-outs is handled in the following sections). Since each gate has a single fan-out, there is a single path from a primary input to any gate in this circuit. In traditional technology mapping, for some gate t, the input to a match at t with the maximum delay from a primary input is defined as the critical input, and this delay is used in combination with the delay of the match to determine the delay (and hence the cost) at the output of the match. In our approach, the minimum delay of a path is achieved by minimizing the

cumulative logical effort for a selected path length. In this case, we determine the critical input and calculate the cost function, as follows.

Let the match at gate t have r inputs $I_1, I_2, \ldots, I_r$. Consider the situation where this match has some input capacitance c_{in_t}, and the path length from a primary input to any input of the match at t is the same. The input capacitance of the match is the load capacitance that each of $I_1, I_2, \ldots, I_r$ has to drive. If the input capacitance at each primary input were also fixed (e.g., equal to $c_{\mathrm{in}_{PI}}$), the electrical effort of all the paths from primary inputs to the input of gate t would be $H = c_{\mathrm{in}_t}/c_{\mathrm{in}_{PI}}$ and would be equal. Thus, on the basis of (7), the critical input I_c, which has maximum delay from its primary input, is the one that has the maximum cumulative logical effort for that path length. When calculating the cost of a match at gate t for some value of path length N, we need to determine the cumulative logical efforts at each input of this match for path lengths $N-1$, and the maximum of these is used to calculate the cumulative logical effort at the output of t.

The above argument makes a couple of assumptions that may seem restrictive. However, we will show that these do not affect the definition of the critical input. The first assumption is that a match will present the same input capacitance on every input pin. This is true for symmetric gates (such as NANDs or NORs), but not for asymmetric gates such as AOIs or OAIs. However, we show in Lemma 1 that even with each input to a match having different electrical efforts, the critical input is still the one with maximum cumulative logical effort, as defined above. The other assumption is that every input has solutions corresponding to a given path length. That is, when determining the optimal solution for path length n, our approach assumes that solutions of path length $n-1$ are available at each input of the match. This may not be the case in general, but if the library includes appropriately sized buffers (inverter pairs), solutions of all path lengths are now available.

A final consideration that has to be accounted for is the load seen by noncritical inputs of a match after sizing the final mapped circuit. The cumulative logical effort up to a gate being mapped is calculated by taking the product of the logical effort of the match itself and the cumulative logical effort of the critical input of the match. Tracing the path from the primary output to a primary input, following the critical input at each gate defines the critical path in the circuit under consideration. At the primary output, the cumulative logical effort is used to size this critical path, as shown in Section II-A, so that the delay on this path is minimized. The noncritical inputs of gates on this path, however, have no choice in this sizing. Is it possible that the load seen by noncritical inputs becomes large enough, so as to make them critical? As the following Lemma shows, this is not the case.

Lemma 1: Let I_c be the critical input of a gate t. After sizing t and its outputs, I_c is still the critical input of t.

Proof: See Appendix. ■

Our logical effort-based technology mapping procedure for fan-out-free circuits can be carried out in a manner similar to the approach for simple paths. The optimum solution at each gate is determined for all values of path lengths. For some path length N under consideration, the cost of each match is the product of the logical effort of the match, and the maximum of the costs at its inputs, corresponding to lengths $N-1$. At the primary output, for some electrical effort, the combination of path length, cumulative logical effort and parasitic delay that minimizes the delay of the circuit as determined by (7) is selected (this assumes that gate sizing will be applied to the selected solution). If only such a fan-out-free circuit is to be mapped, the electrical effort is known. If this fan-out-free circuit is part of a bigger circuit, the electrical effort is determined using delay–C_{in} curves, as discussed later.

Algorithm 1 Optimal LE-Based Technology Mapping for Fan-out-Free Regions

```
// G_t is the cumulative logical effort at the output of gate t, indexed by path lengths
// M_t is the set of selected matches at t, indexed by path length
// M is the set of all possible matches at gate t
// I is the set of inputs to match m

// initialize
for each primary input p do
   G_p[0] = 1
end for

// Phase I: Matching
for each gate t in topological order do
   set M_t[n] = 0 for all n
   for each m ∈ M, with logical effort g_m do
      for each available path length n do
         // calculate cumulative effort G_t[n] from the inputs,
         // using solutions corresponding to path length n − 1
         temp = g_m × max_{i∈I} G_i[n − 1]
         if M_t[n] = 0 OR temp < G_t[n] then
            G_t[n] = temp
            P_t[n] = p_m + P_i[n]
            M_t[n] = m
         end if
      end for
   end for
end for

// Phase II: Selecting Solution
at the primary output, select the combination of G, H and N that minimizes delay

// Phase III: Covering
select matches in a traversal from the primary output to primary inputs, sizing the
matches appropriately
```

The pseudocode of our dynamic-programming-based algorithm for technology mapping fan-out-free circuits is presented in Algorithm 1.

An Illustrative Example: We use Fig. 5 to illustrate Algorithm 1. Here, a simple chain of three gates, A, B, and C, is to be mapped to a library of three cells, X, Y, and Z, with logical efforts g_X, g_Y, and g_Z and parasitic delays p_X, p_Y, and p_Z.

As discussed before, we store optimal solutions for each legal value of path length. For each gate t, we keep track of the accumulated product of logical efforts G_t, the sum of the parasitic delays P_t, and the corresponding matches $\mathcal{M}_t$, indexed according to the length of the path. The path length is obtained by the length at the inputs to the match at t plus one for the match itself.[2]

In the example, the only match of a library pattern at gate A is that of pattern X, and the corresponding solution for A is $G_{\mathrm{A}}[1] = g_x$, $P_{\mathrm{A}}[1] = p_x$, $\mathcal{M}_{\mathrm{A}}[1] = X$. At gate B, however, we have two possible matches, the match of X, with solution $G_{\mathrm{B}}[2] = g_x^2$, $P_{\mathrm{B}}[2] = p_x + p_x$, $\mathcal{M}_{\mathrm{B}}[2] = X$, and the match of

[2]For example, $G_t[3]$ is the cumulative logical effort of a path having length 3, and the corresponding match is stored in $\mathcal{M}_t[3]$.

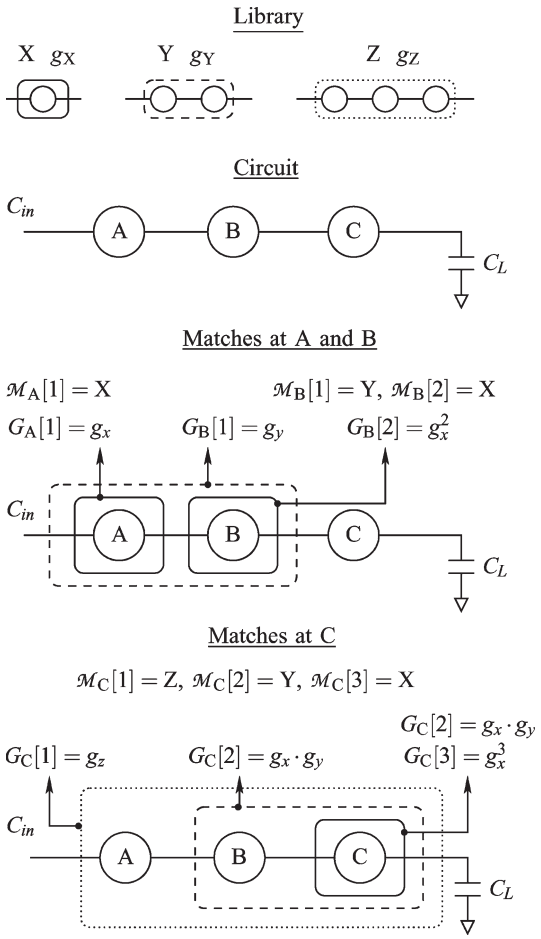

Fig. 5. Example of LE-based technology mapping.

Y, with solution $G_{\rm B}[1] = g_y$, $P_{\rm B}[1] = p_y$, $\mathcal{M}_{\rm B}[1] = Y$. Thus, B has two solutions of length 1 and 2. At gate C, all three library patterns match, generating the following solutions.

1) Match of Z: This is straightforward, with the solution being $G_{\rm C}[1] = g_z$, $P_{\rm C}[1] = p_z$, $\mathcal{M}_{\rm C}[1] = Z$.
2) Match of Y: In this case, the input to the match is A, and the only input solution available is of length 1. Hence, the corresponding solution for C, of length 2, is $G_{\rm C}[2] = g_x \cdot g_y$, $P_{\rm C}[2] = p_x + p_y$, $\mathcal{M}_{\rm C}[2] = Y$.
3) Match of X: The input to this match is B, which has two solutions. Each of these leads to two solutions for C, of length 2: $G_{\rm C}[2] = g_y \cdot g_x$, $P_{\rm C}[2] = p_y + p_x$, $\mathcal{M}_{\rm C}[2] = X$ and of length 3: $G_{\rm C}[3] = g_x^3$, $P_{\rm C}[3] = 3 \cdot p_x$, $\mathcal{M}_{\rm C}[3] = X$.

Note that we now have two solutions at circuit node C of length 2, due to the matches of Y and X. We store the solution with the minimum value of cumulative logical effort.

As we have reached the primary output, the matching step is complete. We have three possible solutions at the primary output, of lengths 1, 2, and 3. The load C_L is known for each solution and assumes that the primary input has a fixed drive capability of $C_{\rm in}$. This determines the electrical effort $H = (C_L/C_{\rm in})$, and we can calculate the minimum delay corresponding to each available solution using (7), and select the minimum. In this case, we have the minimum delay of the mapped circuit with path length 1 to be

$$\hat{D} = 1 \cdot (G_{\rm C}[1] \cdot H)^{1/1} + P_{\rm C}[1] = g_z \cdot H + p_z.$$

For path length 2

$$\begin{aligned}\hat{D} &= 2 \cdot (G_{\rm C}[2] \cdot H)^{1/2} + P_{\rm C}[2] \\ &= 2 \cdot (g_y \cdot g_x \cdot H)^{1/2} + p_y + p_x.\end{aligned}$$

For path length 3

$$\begin{aligned}\hat{D} &= 3 \cdot (G_{\rm C}[3] \cdot H)^{1/3} + P_{\rm C}[3] \\ &= 3 \cdot \left(g_x^3 \cdot H\right)^{1/3} + 3 \cdot p_x.\end{aligned}$$

As discussed before, the individual gate sizes can then be determined using (5).

As described in Section II-B, traditional approaches calculate and store solutions for all possible load values. We trade this off with generating solutions for different values of path length N. The number of legal values of N depends on the circuit being mapped but also depends on the library. During the mapping stage of Algorithm 1, complex gates, if available in the library, will cover a greater part of the circuit while increasing path length by 1. In comparison, simple gates will increase the path length by a larger amount. We also note that the number of path lengths is not the same for all gates in a path but increases as we traverse the path from the input to the output. In practice, we find that keeping track of N solutions at each gate is faster than keeping track of solutions for each load value, as done in the traditional approach.

If we map fan-out-free regions only, Algorithm 1 provides optimal solutions. The path effort F can be used to calculate the sizes of each match selected on the critical path. Matches that are not on the critical path can be sized once the critical path has been fixed. The noncritical paths have a certain amount of slack in the delay that they have to meet. This slack, and the fact that the critical path is now presenting a load that is smaller than previously anticipated, can be used in a possible optimization to control the area of the implementation.

B. Generalized Matching for Fan-Out-Free Regions

As presented in Section III-C, the critical path of a circuit may not correspond to the critical path of a fan-out-free region or tree. Therefore, while Algorithm 1 can optimally map fan-out-free regions, the optimal solution for a tree may not provide the minimum delay for the entire circuit. This drawback also exists in current tree-mapping algorithms. The main problem with correctly addressing this issue is that the critical input (for minimizing circuit delay) of a fan-out-free region is not known during the matching phase and can change depending on the selections made (and their corresponding size assignment) during mapping. In this section, we modify the matching step of Algorithm 1 so as to obtain generalized solutions for fan-out-free trees. In the covering step, these solutions can be used to select the true critical input and the corresponding optimal matches.

Algorithm 2 Optimal LE-Based Matching for Fan-out-Free Regions

```
// The fanout-free region has k inputs, s1, s2,...,sk
// G_{sj→t} is the cumulative logical effort from input sj to the output of gate t, indexed by path lengths
// M_{sj→t} is the set of selected matches at t for each input sj, indexed by path length
// M is the set of all possible matches at gate t
// initialize
for each input sj do
   G_{sj→sj}[0] = 1
end for
for each gate t in topological order do
   set M_{sj→t}[n] = 0 for all inputs sj and path lengths n
   for each m ∈ M do
      for each available path length n do
         // calculate cumulative effort G_{sj→t}[n] from the inputs of the match,
         // using solutions corresponding to path length n − 1
         for each input i of match m, having logical effort g_{mi} do
            for each input sj of the fan-out-free region having a path to i do
               temp = g_{mi} × G_{sj→i}[n − 1]
               if M_{sj→t}[n] = 0 OR temp < G_{sj→t}[n] then
                  G_{sj→t}[n] = temp
                  P_{sj→t}[n] = p_m + P_{sj→i}[n]
                  M_{sj→t}[n] = m
               end if
            end for
         end for
      end for
   end for
end for
```

Consider a fan-out-free region having inputs $s_1, s_2, \ldots, s_k$. In a tree, each gate has only one output, and therefore, if there is a path from an input s_j to a gate t in the fan-out-free region, this path is unique. In Algorithm 1, we stored one solution for each path length for gate t, with cost $G_t[n]$, and the optimality of this solution was based on determining the critical input to the match. In the context of the entire circuit, we can no longer use the critical input within a fan-out-free region to determine global optimality. Instead, we store k solutions for each path length, corresponding to each input s_j of the fan-out-free region, denoted by $G_{s_j \to t}[n]$.

The modified version of the matching step is shown in Algorithm 2. Consider the situation when matching at some gate t. Each input i of the match has a set of solutions of the form $G_{s_j \to i}[n]$, where there exists a path from the jth input of the tree to i. This can be combined with the current match to obtain the solution for gate t, $G_{s_j \to t}[n+1]$. Other matches for gate t of path length $n+1$ from tree input s_j can produce different costs, and as before, we store the minimum cost solution. In this manner, we track solutions of different path lengths from each tree input and defer the determination of criticality to a later stage. The complexity of doing so increases by $O|\mathrm{FI}|$, where $|\mathrm{FI}|$ is the number of tree inputs, which can be potentially large. However, we show in Section V that this number is less than three on the average. In addition, note that the number of tree inputs that fan-in to a gate increases with the depth of the tree, and it is only the output of the tree that has to keep track of solutions from each tree input.

Algorithm 2 is an algorithm for the matching step for fan-out-free circuits, which generates sets of solutions for the fan-out-free region corresponding to different inputs of the region and different path lengths. Given an electrical effort (output load capacitance and input capacitance), the optimal matching solution is readily determined. The covering step is based on delay–C_{in} curves and is described in the following sections. Determining which input is critical is handled after covering.

C. Delay–C_{in} Curves and Their Efficient Calculation

We now turn to the case of gates having multiple fan-outs. Here, the correct choice when mapping and sizing each branch depends on which one is critical, as formalized by the load-distribution problem. Traditional approaches handle each branch separately, but it is clear that this can lead to suboptimal solutions. Our approach to a globally optimal solution uses the notion of delay–C_{in} curves, previously developed for gate sizing [20]. We show how delay–C_{in} curves can be calculated easily, when integrated with Algorithm 2.

The delay–C_{in} curve is characterized at the input of each fan-out-free segment of the circuit. Each point on the curve corresponds to the minimum delay of the critical path from that input to some primary output, for different values of input capacitance. Which primary output terminates the critical path is immaterial; in fact, it is possible that different paths are critical for different values of input capacitance. For some input s, the minimum delay on its delay–C_{in} curve is represented by $D_{s \to \mathrm{PO}}[c_{\mathrm{in}}]$. The critical path may traverse multiple trees, each of which has multiple fan-outs. As shown in Section III-B, correctly assigning capacitance to each fan-out can have a large effect on circuit performance. Delay–C_{in} curves keep track of this information in addition to the minimum delay values for different input capacitances.

We calculate delay–C_{in} curves for each input of every tree in the circuit in a recursive manner, starting from the primary outputs and traversing trees in reverse topological order to the primary inputs. There are three main situations that have to be handled: the base case of the primary outputs; trees that only drive primary outputs; and finally, trees that have multiple fan-outs, as follows.

1) Primary Outputs: The delay–C_{in} curve at a primary output has only one point—a delay of zero for the fixed load being driven. If a required arrival time is specified for each primary output, a proportional delay value can be used in the delay–C_{in} curve. In addition, if the circuit being mapped is part of a bigger design, the effects of different load capacitances at the primary outputs can be captured by adding these to the delay–C_{in} curve of the primary output, thus allowing for an exploration of a much larger space of solutions. This does not add to the complexity of our algorithm.

2) Trees Driving Primary Outputs: When the fan-out-free region drives a primary output, the load being driven is known and is fixed, and the delay–C_{in} curve is straightforward to calculate.

Consider Fig. 6(a), where the circuit drives a fixed load of C_L at a primary output. Our matching algorithm generates four possible solutions, of lengths one to four. For each solution, we know the minimum delay can be obtained by

$$\hat{D} = N(GH)^{1/N} + P = N\left(G \times \frac{C_L}{c_{\mathrm{in}}}\right)^{1/N} + P$$

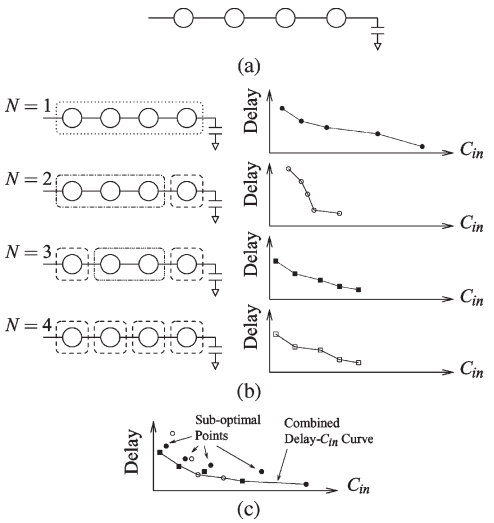

Fig. 6. Delay–$C_{\rm in}$ curves calculation for a tree driving a primary output. (a) Original circuit to be mapped. (b) Solutions from the matching step and corresponding delay–$C_{\rm in}$ curves. (c) Combined delay–$C_{\rm in}$ curve, with suboptimal points removed.

where the matching step gives different values of cumulative logical effort, G, for each value of path length, N. The minimum delays for each solution are therefore functions of the input capacitance $c_{\rm in}$, i.e.,

$$\hat{D} = 1 \cdot \left(G[1] \cdot \frac{C_L}{c_{\rm in}} \right)^{1/1} + P[1], \qquad \text{for } N = 1$$

$$\hat{D} = 1 \cdot \left(G[2] \cdot \frac{C_L}{c_{\rm in}} \right)^{1/2} + P[2], \qquad \text{for } N = 2$$

$$\hat{D} = 1 \cdot \left(G[3] \cdot \frac{C_L}{c_{\rm in}} \right)^{1/3} + P[3], \qquad \text{for } N = 3$$

$$\hat{D} = 1 \cdot \left(G[4] \cdot \frac{C_L}{c_{\rm in}} \right)^{1/4} + P[4], \qquad \text{for } N = 4.$$

Different values of $c_{\rm in}$ give us different delays in the primary output for each of the above possibilities. These constitute the delay–$C_{\rm in}$ curves for each of four mapped solutions, as shown in Fig. 6(b) (these curves are representative curves that are used for illustrative purposes). It is not necessary to keep track of each of these curves, instead, they can be combined to obtain the curve shown in Fig. 6(c) by selecting the minimum delay value for each possible $c_{\rm in}$. Points on this plot that do not lie on the delay–$C_{\rm in}$ curve are suboptimal and can be disregarded, since they represent solutions that have higher input capacitance and greater delay than the points on the curve. Thus, in the context of technology mapping, the delay–$C_{\rm in}$ curve also keeps track of which path length each point on the curve corresponds to.

Note that by calculating the delay–$C_{\rm in}$ curve, we have still not selected any particular match as optimal at this stage. After the matching step, solutions were generated for different path lengths, and the electrical effort was not known. After calculating delay–$C_{\rm in}$ curves, the dependence on path lengths is removed, since each point on the curve explicitly corresponds to the best value of path length for that input capacitance. In addition, the set of solutions is now a function of input capacitance—once this is known, the optimal match is also known. In the current case of the tree driving a primary output, the load capacitance is fixed; however, in the general case presented next, the optimal load capacitance is also determined when calculating the delay–$C_{\rm in}$ curves.

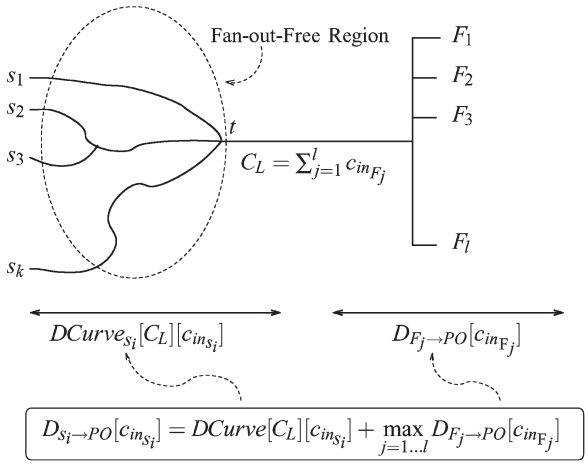

Fig. 7. Delay–$C_{\rm in}$ curves and multiple outputs.

The example in Fig. 6 shows a single input fan-out-free region. In general, delay–$C_{\rm in}$ curves can be calculated for every input of a tree with multiple inputs. In this calculation, the implicit assumption is that the input being considered is the critical input. Whether this is really the case is not known until the covering step, when the appropriate solution is selected.

3) Trees With Multiple Fan-Outs: The most general case when calculating delay–$C_{\rm in}$ curves is that of an intermediate tree, such as the one shown in Fig. 7.

Since this is a fan-out-free region, there is exactly one path from each of the s_i to output t, and after the matching step, we have solutions for each input of the fan-out-free region, for different path lengths. For some load C_L at the output of t, we can calculate the minimum delay from s_i to t for different values of $c_{{\rm in}_{s_i}}$, as done in the previous section. This is denoted by $\text{DCurve}_{s_i}[C_L][c_{{\rm in}_{s_i}}]$ and is one component of the critical path delay from s_i to a primary output. The second component is the delay from the output of t (or equivalently, the input of the critical fan-out, F_j) to a primary output. In order to determine which fan-out is critical, we need to know the minimum delays from each fan-out to any primary output. However, this information is readily available in the delay–$C_{\rm in}$ curves of the fan-outs—recall that this is denoted by $D_{F_j \to {\rm PO}}[c_{{\rm in}_{F_j}}]$. In order to calculate the delay–$C_{\rm in}$ curve at s_i, the above sum of $\text{DCurve}[C_L][c_{{\rm in}_{s_i}}]$ and $\max_{j=1,\ldots,l} D_{F_j \to {\rm PO}}[c_{{\rm in}_{F_j}}]$ has to be repeated for all values of C_L, and the minimum is taken. Thus

$$D_{s_i \to {\rm PO}}[c_{{\rm in}_{s_i}}] = \min_{C_L} \Big\{ \text{DCurve}[C_L][c_{{\rm in}_{s_i}}] + \max_{j=1,\ldots,l} D_{F_j \to {\rm PO}}[c_{{\rm in}_{F_j}}] \Big\}. \quad (8)$$

Algorithm 3 Calculating the Delay-C_{in} Curve for Input s_i of a Multiple-Fan-out Tree

Calculate $DCurve_{s_i}[C_L][c_{in_{s_i}}]$
// s_i is the input, t is the output of the path
for all values of path length n **do**
  $temp = n\left[G_{s_i \to t}[n] \times \frac{C_L}{c_{in_{s_i}}}\right]^{1/n} + P_{s_i \to t}[n]$
  if $temp < DCurve_{s_i}[C_L][c_{in_{s_i}}]$ **then**
    $DCurve_{s_i}[C_L][c_{in_{s_i}}] = temp$
    Store n
  end if
end for

Calculate $D_{s_i \to PO}[c_{in_{s_i}}]$
// t has l outputs, $\mathrm{F}_0, \mathrm{F}_1 \ldots \mathrm{F}_l$
for every combination of c_{in_j} of all fanouts F_j **do**
  if the selected combination is not redundant **then**
    $C_L = \sum_{j=1}^{l} c_{in_j}$
    Calculate $DCurve_{s_i}[C_L][c_{in_{s_i}}]$
    $temp = DCurve[C_L][c_{in_{s_i}}] + \max_{j=1..l} D_{F_j \to PO}[c_{in_{\mathrm{F}_j}}]$
    if $temp < D_{s_i \to PO}[c_{in_{s_i}}]$ **then**
      $D_{s_i \to PO}[c_{in_{s_i}}] = temp$
      Store the input capacitances $c_{in_{\mathrm{F}_j}}$ of each fanouts
    end if
  end if
end for

Algorithm 4 MELT: Technology Mapping using Logical Effort

Divide the circuit into fan-out-free regions

PI→ PO Traversal: generate matches for each fan-out-free region using Algorithm 2, storing optimal matches for each input of the fan-out-free region

PO→ PI Traversal: calculate Delay-C_{in} curves for each input to the fan-out-free region using Algorithm 3

PI→ PO Traversal: select the optimal electrical effort for each fan-out-free region, and the corresponding lengths

Covering: use the assigned output and input capacitances to generate the corresponding optimal covers for each fan-out-free region

Algorithm 3 shows how the delay–C_{in} curve of input s_i of a fan-out-free region terminating in t can be calculated. Given an electrical effort, $H = C_L/c_{\mathrm{in}_{s_i}}$, the first procedure, calculate DCurve_{s_i}, is used to calculate the best delay of the fan-out-free region from all the solutions of different lengths that have been generated by Algorithm 2. Given the electrical effort $H = C_L/c_{\mathrm{in}}$, this procedure determines the length of the path and the cumulative logical effort that supply the best delay. The procedure, calculate $D_{s_i \to \mathrm{PO}}$, of Algorithm 3 determines the best load and the best distribution of this load to all fan-outs for the given input capacitance, as described above.

While it may seem that the total number of combinations of $c_{\mathrm{in}_{F_j}}$ is large (it is in fact $O(|c_{\mathrm{in}_{\mathrm{F}1}}| \times |c_{\mathrm{in}_{F2}}| \times \cdots \times |c_{\mathrm{in}_{Fl}}|)$, where $|c_{\mathrm{in}_{F_j}}|$ is the number of possible values of input capacitance of F_j), it is shown in [20] that the number of combinations that actually have to be considered is much smaller $(O(|c_{\mathrm{in}_{F1}}| + |c_{\mathrm{in}_{F2}}| + \cdots + |c_{\mathrm{in}_{Fl}}|))$. This is accomplished as follows: We take the minimum values of $c_{\mathrm{in}_{F_j}}$ for all j as the first combination and sort the fan-outs by delay. For example, suppose F_1 is the critical fan-out. Any other combination of $C_{\mathrm{in}_{F_1}}$ with other values of $C_{\mathrm{in}_{F_j}}$, $j \neq 1$ can be ignored, since they will lead to a higher value of C_L and have the same maximum delay to a primary output. The next combination can be obtained by selecting the next value of $c_{\mathrm{in}_{F_1}}$ and again determining the most critical fan-out.

As discussed in the previous section, in the matching step (Algorithm 2), for this fan-out-free region, we had obtained a set of solutions for different values of path lengths N for input s_i, and the electrical effort (the ratio of the output and input capacitances) was an unknown. After the delay–C_{in} curve has been calculated, we now have a solution for each value of input capacitance, and the dependence on path length has been removed. The unknown value of output capacitance, C_L, that gives us the best delay is now a known quantity and is embedded in the delay–C_{in} curve.

Recall the load-distribution problem at a gate with multiple fan-outs, in which the correct assignment of capacitances between a gate and its fan-outs and the correct assignment of capacitances between the fan-outs could have a large impact on the overall delay of the circuit. In Algorithm 3, we consider all values of load capacitance C_L when selecting the optimal solution for input s_i of the fan-out-free region. This handles the first part of the load-distribution problem, that of the correct distribution of capacitance between a gate and its multiple fan-outs. The different values of load capacitance C_L are obtained by considering all combinations of input capacitances of the fan-outs, $c_{\mathrm{in}_{F_j}}$. This implicitly handles the second aspect of the load-distribution problem, that of distributing a capacitance between multiple fan-outs. For some values of C_L, a particular fan-out F_x may be critical; for some others, another fan-out, F_y, may be critical. This is taken care of by the formulation of (8) and in Algorithm 3.

Algorithm 3 is a dynamic programming algorithm. The delay–C_{in} curves of one fan-out-free region are calculated based on the curves at its outputs, and a particular critical path delay is obtained by simply taking the combination of the delay of the fan-out-free region with the maximum critical path delay of the outputs. This also exhibits optimal substructure, since the delay of the critical path is minimized only when the delay of each component of the path is minimized. Hence, the delay curves obtained at primary input encode globally optimal solutions to the load-distribution problem.

D. Comprehensive Technology Mapping Approach

The complete approach for logical effort-based technology mapping addressing the load-distribution problem, called Technology Mapping Using Logical Effort (MELT) (the order of letters are suggestive of the multiple input-output-input traversals of the circuit required by our approach), is presented in Algorithm 4. After the first three steps, which have been described previously, we have delay–C_{in} curves at the primary inputs of the circuit. At each primary input, the load that minimizes the maximum delay to any output is selected.[3] The primary inputs are processed in decreasing order of this delay. A forward traversal from the primary inputs using the selected loads fixes the input and output capacitances and the lengths of each fan-out-free region. This information, in turn, can be used to select the matches of the optimal solution.

In Algorithm 4, there are two issues that restrict the optimality of the final solution. First, the processing of each

[3]We assume that each primary input is represented by a mid-sized inverter, the delay of which is taken into account so as to avoid overloading primary inputs with overlarge capacitances.

input of a fan-out-free region is carried out independent of other inputs of this region. The solutions generated by different inputs may contradict each other. Second, in general, circuits have reconvergent fan-outs. The interaction between multiple, overlapping reconvergent paths is difficult to analyze efficiently. For these cases, we use the heuristic of assuming that all paths are independent and make the best choice available. If interconnect capacitances can be estimated at the technology mapping stage, they can be incorporated into the delay–$C_{\rm in}$ curve calculation. However, within a fan-out-free region, logical effort cannot account for interconnect capacitance, and the traditional mapping will be more accurate than MELT.

The first step in Algorithm 4, that of generating matches, takes time $O(|V| + |E|) \cdot |L| \cdot |N| \cdot |\mathrm{FI}|$ for each tree, where $|V|$ is the number of nodes and $|E|$ is the number of edges in the tree, $|L|$ is the size of the library, $|N|$ is the maximum path length, and $|\mathrm{FI}|$ is the number of inputs to the tree. In traditional approaches, matches for every load value have to be determined and stored, and the library used for matching includes multiple sizes of each gate. In contrast, our approach stores solutions for all values of $|N|$ (which is small, on the average), and the library has only one instance of each gate type. Since we store solutions for each path length and each input to a fan-out-free region, the storage requirement is $O(|V| \cdot |N| \cdot |\mathrm{FI}|)$. Note that this is a very loose upper bound. We show in the results section that $|N|$ is relatively small, and while $|\mathrm{FI}|$ can be exponentially large in theory ($O(2^P)$, if the entire fan-out-free region is a tree of two-input gates for a path length of P from input to output), in practice, it is much smaller.

Calculating the delay–$C_{\rm in}$ curves dominates the running time of our algorithm. The time complexity of this step is $O(|\mathrm{FI}| \cdot |c_{\rm in}|^2 \cdot |\mathrm{FO}|^2)$, where $|c_{\rm in}|$ is the number of possible values for input capacitances, and $|\mathrm{FO}|$ is the number of fan-outs at a multiple fan-out point. This bound too is very loose, and for benchmark circuits, the running time is of the same order as that of SIS.

V. Results

In order to validate our approach, we have implemented Algorithm 4 and used it to map ISCAS and MCNC combinational benchmark circuits. These results were compared with SIS [22]. The library used for SIS was generated by calibrating INV; two-, three-, and four-input NAND and NOR gates; and a variety of AOI and OAI gates on a 0.1μ technology using the Berkeley Predictive Technology Model [23]. Twenty sizes of each gate were generated, for a total library size of approximately 400 elements. These gates were also calibrated in order to obtain the logical effort and parasitic delays, which constitute the library used by our algorithm, with 23 elements, one for each gate type. In our approach, calculating gate sizes as described in Section IV can lead to arbitrary values (less than the largest gate size). In order to make a fair comparison with SIS, gate sizes are normalized to the 20 sizes of each gate that are used by SIS.

Table I presents structural statistics of the benchmark circuits used in our experiments. For each circuit, we list the size, as determined by the number of gates, and the number of trees, after the circuit has been broken into fan-out-free regions. Next, we present the minimum, maximum, and average values of the sizes of the trees; the number of fan-ins and fan-outs of each tree; and the path lengths within each tree. The traditional technology mapping approach maps each tree separately, whereas our approach deals with the entire circuit as a whole. Consequently, the running time of our algorithm depends not only on the sizes of each tree but also on the number of fan-ins, fan-outs, and the path lengths within each tree. As can be seen from Table I, in the worst case, each of these can be large; however, the average case listed in the last row is much more tractable. For example, the circuit pair has a tree with 41 outputs. Combining the delay–$C_{\rm in}$ curves of these outputs is expensive if these curves have approximately similar delay values. If this were the case for all trees, the runtime would be prohibitive. However, other trees in this circuit have much smaller fan-outs, and an average fan-out of 3.79 is tractable. Similarly, the values of the other structural parameters presented in Table I affect the runtime of our algorithm. MELT determines and stores matches for all values of path lengths for each input of fan-out-free regions of the circuit. These are then examined to obtain the minimum achievable delays for the fan-out-free regions. Therefore, having long path lengths and a large number of inputs can lead to large run times. As before, although some paths can be large and a few trees have a large number of fan-ins, the averages are skewed toward small values.

The results of mapping these circuits are as shown in Table II. The first column lists the benchmark circuit. The next two, under the title SIS, show the best delay obtained for each circuit using the command `map -n 1` in SIS (which tries to minimize the delay of the mapped circuit), the area of the final solution, and the corresponding running time, T, in seconds. The performance of MELT for the same circuits is as shown. On the average, our algorithm generates circuits that are 39.45% faster and 32.77% smaller than those obtained from SIS. During the covering step, the load at multiple fan-out points is accurately known, as is the optimal electrical effort for individual segments, which is not taken into account by SIS. This leads to a higher incidence of complex gates in the MELT solution. The area tradeoff between using complex and simple gates depends on the sizes of each gate being selected. Smaller sizes of complex gates such as OAIs are more area-efficient than the equivalent circuit using simple gates such as NANDs, NORs, and INVs. However, the larger sizes of these complex gates occupy more area than the equivalent circuit using simple gates. Thus, while we usually have an area improvement for most circuits, for some circuits such as count and $i5$, we obtain more expensive (albeit faster) solutions from MELT. In the case of C6288, the circuits selected are very similar and consist largely of NAND2 gates. In this case, the gate sizes selected in MELT are larger than those selected by SIS. Once again, it is the electrical effort that guides this size selection. While we obtain mapped circuits that are faster, the area overhead in these cases is significant.

It is important to point out that it is quite understandable that the area numbers for our algorithm are higher, simply because MELT only optimizes the delay, and does not explicitly

TABLE I
CIRCUIT STATISTICS

Circuit	Number of		Tree Size Statistics			|FI| Statistics			|FO| Statistics			Path Length Statistics		
	Gates	Trees	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
C432	654	79	1	182	8.28	1	45	3.23	1	19	2.86	1	14	2.92
C499	789	91	1	42	8.67	1	16	3.26	1	12	3.16	1	9	4.23
C880	1272	143	1	58	8.90	1	17	3.42	1	8	3.18	1	16	3.64
C1355	1973	291	1	42	6.78	1	16	2.78	1	12	2.75	1	7	3.76
C1908	2769	375	1	100	7.38	1	50	2.65	1	16	2.62	1	18	4.06
C2670	3875	427	1	303	9.07	1	72	3.07	1	20	2.85	1	22	4.09
C3540	5375	566	1	227	9.50	1	62	3.24	1	30	3.19	1	29	4.04
C5315	8016	832	1	57	9.63	1	19	3.50	1	23	3.43	1	17	4.16
C6288	9568	1472	1	12	6.50	1	4	2.62	1	16	2.62	1	7	3.91
C7552	11491	1291	1	94	8.90	1	35	3.04	1	15	2.96	1	16	4.41
9symml	314	18	1	268	17.44	1	163	10.72	1	22	10.28	1	16	2.39
alu2	441	67	1	43	6.58	1	35	4.70	1	21	4.64	1	7	2.72
apex6	1536	303	1	46	5.07	1	26	3.05	1	34	2.93	1	14	2.66
b9	383	45	1	50	8.51	1	18	3.64	1	18	3.20	1	16	2.99
cc	157	31	1	17	5.06	1	9	3.13	1	14	3.10	1	7	2.73
count	237	48	1	13	4.94	1	6	2.65	1	16	2.25	1	10	2.94
cmb	85	17	1	27	5.00	1	16	3.29	1	4	2.59	1	12	2.09
decod	93	23	1	5	4.04	1	4	3.17	1	16	3.65	1	3	1.87
example2	510	127	1	14	4.02	1	14	2.88	1	20	2.73	1	6	2.42
i5	1009	193	1	28	5.23	1	15	2.85	1	10	2.50	1	6	2.23
pair	3646	414	1	66	8.81	1	22	3.87	1	41	3.79	1	20	3.79
pcler8	137	42	1	8	3.26	1	6	2.36	1	9	2.12	1	4	2.19
ttt2	444	43	1	43	10.33	1	22	6.12	1	19	6.05	1	13	3.44
vda	1665	146	1	53	11.40	1	51	9.90	1	92	10.05	1	4	2.50
x1	470	77	1	27	6.10	1	25	5.16	1	18	4.95	1	4	2.01
Average			Tree Size 7.58			|FI| 3.93			|FO| 3.78			Path Length 3.13		

TABLE II
TECHNOLOGY MAPPING: SIS VERSUS MELT

Circuit	SIS			MELT			% Change in	
	Delay(*ps*)	Area	*T*(s)	Delay(*ps*)	Area	*T*(s)	Delay	Area
C432	1629.21	1609.07	6.84	795.93	1407.17	4.03	51.15	12.55
C499	822.77	2143.44	11.53	658.15	2813.28	3.20	20.01	-31.25
C880	700.98	1634.78	7.58	643.43	1399.71	2.20	8.21	14.38
C1355	854.79	2527.04	11.44	678.19	2581.55	4.55	20.66	-2.16
C1908	1289.69	3431.64	21.99	868.42	2402.54	4.23	32.66	29.99
C2670	2161.43	5664.06	41.95	868.82	3515.99	8.19	59.80	37.92
C3540	2624.79	9720.64	45.40	1218.21	4650.69	12.90	53.59	52.16
C5315	1709.62	13790.11	103.58	971.36	7016.89	13.49	43.18	49.12
C6288	2931.69	11358.46	50.51	2893.31	18008.40	17.94	1.31	-58.55
C7552	1825.71	18687.65	183.32	1097.69	6903.13	19.64	39.88	63.06
9symml	1229.74	2301.95	9.76	346.78	610.05	0.55	71.80	73.50
alu2	2357.10	4123.53	21.19	1137.77	1449.93	2.76	51.73	64.84
apex6	769.49	3691.38	16.93	388.89	3687.13	3.86	49.46	0.12
b9	440.00	604.68	3.70	227.89	684.65	0.56	48.21	-13.23
cc	307.25	272.07	2.23	157.22	232.89	0.25	48.83	14.40
cmb	228.95	241.70	1.03	216.70	258.88	0.29	5.35	-7.11
count	907.22	553.14	2.84	592.17	854.11	0.38	34.73	-54.41
decod	326.95	232.98	2.50	98.50	291.79	0.26	69.87	-25.24
example2	711.10	1404.83	8.51	331.66	1179.15	1.46	53.36	16.06
i5	392.54	1419.17	6.81	222.76	2330.55	1.93	43.25	-64.22
pair	1224.90	8675.52	53.86	814.31	7053.61	11.78	33.52	18.70
pcler8	615.45	405.61	1.51	331.04	551.60	0.33	46.21	-35.99
ttt2	824.10	1724.02	15.50	279.97	854.25	0.68	66.03	50.45
vda	2193.22	11328.42	70.82	443.30	1888.76	20.75	79.79	83.33
x1	1055.87	2632.88	16.95	343.81	1444.36	1.21	67.44	45.14
Average							39.45	32.77

optimize the area. Such a delay minimizer is very useful in practice, since it allows a designer to determine the best possible performance that can be achieved by a circuit. It is easily seen that, in terms of delay, MELT always outperforms SIS, and the improvement over SIS ranges from just over 1% to nearly 80%.

This wide range in the achievable improvement can be explained by the circuit characteristics described in Table I. For example, C6288 has a large number of fan-out-free regions with small average path lengths, as compared to other circuits. For small path lengths, the effect of varying the electrical effort is limited, which, in turn, restricts the freedom that our algorithm has and results in solutions that are very similar to those that would be obtained by traditional methods.

VI. Conclusion and Future Directions

This paper presents a new approach to technology mapping, based on the theory of logical effort. Most of the improvement obtained by our algorithm is due to the solution of the load-distribution problem, which allows for accurate assignment of capacitances at multiple fan-out points. This leads to better selection of matches, since the exact load to be driven is known. We observe an average improvement of 39.45% in terms of delay, and 32.77% in terms of area, as compared to SIS.

In [24] and [25], all possible decompositions of circuits are considered during the matching step. The algorithm divides the circuit into disjoint ugates and applies technology mapping to each such ugate. Our algorithm can be extended to generate matches in each ugate and calculate delay–C_{in} curves by traversing the ugates. This approach can also be applied to DAG mapping [2], which allows matches across tree boundaries and therefore can generate better solutions. Here, multiple fan-out points are not well defined initially. However, once the matching has been done, the fan-out points are specified, and the delay–C_{in} curves can be calculated as before.

Appendix
Proof of Lemma 1

We first prove the case of symmetric gates, in which the delay characteristics of each input pin to the output of the gate are the same. The proof for the case of asymmetric gates is similar and follows from the proof for symmetric gates.

Consider the situation where we have a match at some gate t, with r inputs $I_1, I_2, \ldots, I_r$, each having cumulative logical effort for path of length n from the primary inputs $G_{I_1}[n], G_{I_2}[n], \ldots, G_{I_r}[n]$. Since gate t is symmetric, the load being driven by each of $I_1, I_2, \ldots, I_r$ is equal and is c_{in_t}, a value that is yet to be determined. Let I_c be the critical input, and let I_j denote the other noncritical inputs. As mentioned previously, I_c being the critical input implies that $G_{I_c}[n] \geq G_{I_j}[n] \forall j$. In this case, we select $G_{I_c}[n]$ to be multiplied with the gate effort of the match at t, g_{m_t} in order to obtain $G_t[n+1]$. This means that when the segment is sized, the size of the match at gate t (which determines the load c_{in_t} at the output of any I_j) will be determined by the value of $G_{I_c}[n]$, and this size will be different from the size determined if we had selected $G_{I_j}[n]$. We show that this is in fact the correct choice to make.

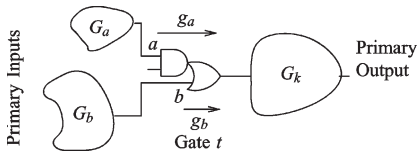

Fig. 8. Proof of Lemma 1 for asymmetrical gates.

As mentioned previously, the sizes of gates are determined by applying (5) in a backward traversal. If the load at the primary output is C_L, and there are k gates from gate t to the primary output in the mapped solution, (5) can be applied to each gate successively to obtain

$$c_{\text{in}_t} = \frac{\prod_k g_k}{\hat{f}^k} \cdot C_L. \tag{9}$$

Note that the stage effort for optimal delay is in the denominator of (9), and $\hat{f} = F^{1/N} = (G \cdot H)^{1/N}$. Therefore, choosing $G_{I_c}[n]$ induces a size on gate t that is smaller than that which we would have obtained by using $G_{I_j}[n]$. This means that the delay $G_{I_j}[n]$ would have induced (e.g., D_{I_j}) would depend on a larger size of gate t. Since t is now smaller than anticipated by gate I_j, its load on I_j is smaller, and hence, the delay of the branch from an input to gate I_j does not increase (from the value it would have been, if the solution corresponding to I_j had been used to size t) by taking the choice of $G_{I_c}[n]$, i.e., I_c is still the critical input of t.

We now turn to the general case of asymmetric gates. In this case, the logical effort of each input of the gate to the output depends on the functionality. However, we show that the assertion of Lemma 1 still holds.

Consider asymmetric gate t with two of its inputs, a and b, having logical efforts g_a and g_b, respectively, as shown in Fig. 8. Let the cumulative logical effort up to each input be G_a and G_b, and the cumulative logical effort from the output of t to a primary output be G_k. Since gate t is asymmetric, the capacitance at each input is different, but this also implies that the logical efforts are in the same ratio, i.e., if $c_{\text{in}_b} = z \times c_{\text{in}_a}$, then $g_b = z \times g_a$ (this follows from the definition of logical effort). There are two cases to consider.

1) $G_a > G_b$: In this case, the solution at input a is selected as it is considered to be critical. Sizing gate t according to this solution will imply some capacitance $c^*_{\text{in}_b}$ at input b, and we need to show that $c^*_{\text{in}_b} \leq c_{\text{in}_b}$, which is what we would obtain if G_b were used to size gate t

$$\begin{aligned} c_{\text{in}_a} &= \frac{G_k \cdot g_a}{\hat{f}_a^k} \cdot C_L \\ c^*_{\text{in}_b} &= z \times c_{\text{in}_a} \\ &= z \times \frac{G_k \cdot g_a}{\hat{f}_a^k} f \cdot C_L = \frac{G_k \cdot g_b}{\hat{f}_a^k} \cdot C_L \\ c_{\text{in}_b} &= \frac{G_k \cdot g_b}{\hat{f}_b^k} \cdot C_L. \end{aligned}$$

Since $G_a > G_b$, $\hat{f}_a > \hat{f}_b$ and $c^*_{\text{in}_b} < c_{\text{in}_b}$.

2) $G_b > G_a$: As in the previous case, the solution at input b is selected, which implies some capacitance $c^*_{\text{in}_a}$ at input a. We need to show that $c^*_{\text{in}_a} \leq c_{\text{in}_a}$

$$c_{\text{in}_b} = \frac{G_k \cdot g_b}{\hat{f}_b^k} \cdot C_L$$
$$c^*_{\text{in}_a} = \frac{c_{\text{in}_b}}{z} = \frac{G_k \cdot g_b}{\hat{f}_b^k \times z} \cdot C_L = \frac{G_k \cdot g_a}{\hat{f}_b^k} \cdot C_L$$
$$c_{\text{in}_a} = \frac{G_k \cdot g_a}{\hat{f}_a^k} \cdot C_L.$$

Since $G_b > G_a$, $\hat{f}_b > \hat{f}_a$, and $c^*_{\text{in}_a} < c_{\text{in}_a}$.

Thus, in both cases, the noncritical input eventually drives a smaller load than anticipated, and therefore, the delay at the noncritical input does not increase to a value greater than that of the critical input. ■

REFERENCES

[1] K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 1987, pp. 341–347.

[2] Y. Kukimoto, R. K. Brayton, and P. Sawkar, "Delay-optimal technology mapping by DAG covering," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 1998, pp. 348–351.

[3] H. J. Touati, C. W. Moon, R. K. Brayton, and A. Wang, "Performance-oriented technology mapping," in *Proc. 6th MIT Conf. Adv. Res. VLSI*, 1990, pp. 79–97.

[4] J. Grodstein, E. Lehman, H. Harkness, B. Grundmann, and Y. Watanabe, "A delay model for logic synthesis of continuously-sized networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1995, pp. 458–462.

[5] L. Stok, M. A. Iyer, and A. J. Sullivan, "Wavefront technology mapping," in *Proc. Des. Autom. Test Eur. Conf.*, Mar. 1999, pp. 531–536.

[6] B. Hu, Y. Watanabe, A. Kondratyev, and M. Marek-Sadowska, "Gain-based technology mapping for discrete-size cell libraries," in *Proc. IEEE/ACM Des. Autom. Conf.*, Jun. 2003, pp. 574–579.

[7] W. Gosti, S. Khatri, and A. Sangiovanni-Vincentelli, "Addressing the timing closure problem by integrating logic optimization and placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 224–231.

[8] S. Chatterjee, A. Mishchenko, R. Brayton, X. Wang, and T. Kam, "Reducing structural bias in technology mapping," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2894–2903, Dec. 2006.

[9] A. Mishchenko, S. Chatterjee, R. Brayton, and M. Ciesielski, "An integrated technology mapping environment," in *Proc. Int. Workshop Logic Synth.*, 2005, pp. 383–390.

[10] A. Mishchenko, S. Chatterjee, J.-H. Jiang, and R. Brayton, "Integrating logic synthesis, technology mapping, and retiming," in *Proc. Int. Workshop Logic Synth.*, 2005, pp. 177–181.

[11] R. F. Sproull and I. E. Sutherland, "Theory of logical effort: Designing for speed on the back of an envelope," in *Proc. IEEE Adv. Res. VLSI*, 1991, pp. 1–16.

[12] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan Kaufmann, 1999.

[13] F. Beeftink, P. Kudva, D. Kung, and L. Stok, "Gate-size selection for standard cell libraries," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1998, pp. 545–550.

[14] W. Donath, P. Kudva, L. Stok, P. Villarrubia, L. Reddy, A. Sullivan, and K. Chakraborty, "Transformational placement and synthesis," in *Proc. Des. Autom. Test Eur. Conf.*, Mar. 2000, pp. 194–201.

[15] K. Sulimma, I. Neumann, L. van Ginneken, and W. Kunz, "Improving placement under the constant delay model," in *Proc. Des. Autom. Test Eur. Conf.*, Mar. 2002, pp. 677–682.

[16] L. Stok, D. S. Kung, D. Brand, A. D. Drumm, A. J. Sullivan, L. N. Reddy, N. Hieter, D. J. Geiger, H. H. Chao, and P. J. Osler, "BooleDozer: Logic synthesis for ASICs," *IBM J. Res. Develop.*, vol. 40, no. 4, pp. 407–430, 1996.

[17] *Gain Based Synthesis: Speeding RTL to Silicon*, 2002. Magma Design Automation White Paper. [Online]. Available: http://www.magma-da.com/c/@57hzNi1ExOwpA/Pages/Gainbasedoverview.html

[18] S. K. Karandikar and S. S. Sapatnekar, "Fast comparisons of circuit implementations," in *Proc. Des. Autom. Test Eur. Conf.*, Feb. 2004, pp. 910–915.

[19] S. K. Karandikar and S. S. Sapatnekar, "Fast estimation of area-delay tradeoffs in circuit sizing," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 3575–3578.

[20] S. K. Karandikar and S. S. Sapatnekar, "Fast comparisons of circuit implementations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 12, pp. 1329–1339, Dec. 2005.

[21] R. L. Rudell, "Logic synthesis for VLSI design," Ph.D. dissertation, Univ. California Berkeley, Berkeley, CA, 1989.

[22] 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," Electron. Res. Lab., Dept. Elect. Eng. and Comput. Sci., Univ. California, Berkeley, CA, Tech. Rep. UCB/ERL M92/41, May 1992.

[23] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2000, pp. 201–204.

[24] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic decomposition during technology mapping," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1995, pp. 264–271.

[25] E. Lehman, Y. Watanabe, J. Grodstein, and H. Harkness, "Logic decomposition during technology mapping," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 16, no. 8, pp. 813–834, Aug. 1997.

Shrirang K. Karandikar received the B.E. degree from the University of Pune, Pune, India, in 1994; the M.S. degree from Clarkson University, Potsdam, NY, in 1996; and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, in 2004.

He was with Intel's Logic and Validation Technology group from 1997 to 1999 and was a Research Staff Member with IBM's Austin Research Laboratory until 2007. He is currently with Computational Research Laboratories, Pune. His research interests cover logic synthesis, physical design, and their interaction.

Sachin S. Sapatnekar (S'86–M'88–SM'99–F'03) received the B.Tech. degree from the Indian Institute of Technology, Bombay, India, in 1987; the M.S. degree from Syracuse University, Syracuse, NY, in 1989; and the Ph.D. degree from the University of Illinois at Urbana–Champaign, Urbana, IL, in 1992.

From 1992 to 1997, he was an Assistant Professor with the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA. He is currently the Robert and Marjorie Henle Professor with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis. He has authored several books and papers in the areas of timing and layout.

Dr. Sapatnekar has held positions on the editorial board of the IEEE TRANSACTIONS ON VLSI SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, the IEEE DESIGN AND TEST, and the IEEE TRANSACTIONS ON CAD. He has served on the Technical Program Committee for various conferences, as Technical Program and General Chair for Tau and ISPD, and Technical Program cochair for DAC. He has been a Distinguished Visitor for the IEEE Computer Society and a Distinguished Lecturer for the IEEE Circuits and Systems Society. He is a recipient of the NSF Career Award, three best paper awards at DAC and one at ICCD, and the SRC Technical Excellence award.