Simultaneous Technology Mapping and Placement for Delay Minimization

Yifang Liu, *Student Member, IEEE,* Rupesh S. Shelar, *Member, IEEE,* and Jiang Hu, *Senior Member, IEEE*

***Abstract*—Technology mapping and placement have a significant impact on delays in standard cell-based very large scale integrated circuits. Traditionally, these steps are applied separately to optimize the delays, possibly since efficient algorithms that allow the simultaneous exploration of the mapping and placement solution spaces are unknown. In this paper, we present an exact polynomial time algorithm for delay-optimal placement of a tree and extend the same to simultaneous technology mapping and placement for the optimal delay in the tree. We extend the algorithm by employing Lagrangian relaxation technique, which assesses the timing criticality of paths beyond a tree, to optimize the delays in directed acyclic graphs. Experimental results on benchmark circuits in a 70 nm technology show that our algorithms improve timing significantly with remarkably less runtimes compared to a competitive approach of iterative conventional timing-driven mapping and multilevel placement.**

***Index Terms*—Algorithms, directed acyclic graph, physical synthesis, placement, technology mapping, tree.**

I. Introduction

A. Motivation

IN TODAY'S technologies, interconnects contribute to significant portion of the overall delay in very large scale integrated circuits. The trend is likely to continue, or worsen, as the technology scaling continues, since the wire-delays as well as cell-delays do not scale. The interconnect delay depends on the topology and layer assignment, which is determined by the routing step. This freedom available in the routing phase is often insufficient to optimize the circuit for the required performance. The placement and technology mapping steps also have great impact on the interconnect delay, since the former decides where the locations of the driver and receivers of a net are and the latter decides which nets exist in the design. Consequently, the algorithms for layout-driven technology mapping, timing-driven placement, and physical synthesis have received attention from computer-aided design researchers over the last several years.

Manuscript received December 4, 2009; revised April 26, 2010 and August 26, 2010; accepted September 27, 2010. Date of current version February 11, 2011. This paper was recommended by Associate Editor I. Markov.

Y. Liu is with Google Inc., Mountain View, CA 94043 USA. This work was done when he was a graduate student at Texas A&M University, College Station, TX 77843 USA (e-mail: ifnliu@gmail.com).

R. S. Shelar is with Intel Corporation, Hillsboro, OR 97124 USA (e-mail: rupesh.s.shelar@intel.com).

J. Hu is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: jianghu@ece.tamu.edu).

Digital Object Identifier 10.1109/TCAD.2010.2089569

B. Previous Work

The technology mapping problem minimizing metrics such as total cell area for a directed acyclic graph (DAGs) is known to be NP-hard. For relatively simple structures such as trees, however, the problem can be solved optimally in a polynomial time. The technology mapping algorithm to map individual trees rooted at multi-fanout points or primary outputs in a DAG on to a set of cells in a library was first proposed by Keutzer [1]. The algorithm employs a dynamic programming (DP) technique and runs in polynomial time in the size of a tree, ensuring optimality for metrics such as total cell-area. Most of the subsequent work employs the same technique to optimize various cost functions involving area, delay, power, possibly subject to constraints, as in [2]. The layout-driven technology mapping was proposed by Pedram *et al.* [3], where an initial placement of a subject graph and the assumption about the placement of a match was employed to evaluate wire-delays and cell-delays to derive a delay-optimized mapped netlist. An obvious limitation of the work is that even for a tree, the placement of the subject graph and that of the mapped netlist can be quite different. Moreover, there are multiple placement possibilities for a choice at each node in the tree, whereas only one placement, that of the center of gravity based on the locations of choices at fanins and (unmapped) fanouts, is considered. The second limitation was partially eliminated in the subsequent work [4], which solved the problem of simultaneous technology mapping and linear placement of trees in polynomial time. However, the assumption about the placement of the cells in a tree in a single row is not practical, since the cells are allowed to be placed in different rows in 2-D area. To overcome this limitation, the subsequent work employed iterative technology decomposition, mapping, and placement [5]–[7] to place the primitive gates in a given area, perform mapping with assumptions about the placement of a mapped cell, and then place the mapped netlist or derive the placement of the subject graph from the same for the next iteration.

Many industrial tools, which perform physical synthesis, are believed to employ similar iterative mapping and placement schemes to improve the delays locally in parts of the circuit. The limitation of such an approach is that it neither ensures optimality nor guarantees convergence, as a different mapping solution leads to a new placement. Thus, the problem of simultaneous technology mapping and 2-D placement even for trees remains unsolved even today. Hrkić *et al.* [8] have proposed a DP-based approach for timing-driven logic replication. It is unclear whether the algorithm in their work can result in, or be applied for, delay-optimal placement of a tree, since pruning out placement choices during the bottom-up solution generation may eliminate optimal placement ones. The optimal placement choices are dependent on the location(s) of the root(s) and do not become evident till the reverse topological traversal to select among the generated solutions. Therefore, any approach that prunes out placement choices during the bottom-up solution generation may result in suboptimal delay placement. Moreover, they do not apply the algorithm to simultaneous technology mapping and placement. Recently, Wang *et al.* [9] proposed an iterative mapping scheme employing multipliers, similar to those in a Lagrangian relaxation (LR) technique, to optimize the area/power under fixed cell-delay model; the wire-delays based on the placement, however, are not considered. Another body of work on technology mapping includes so-called DAG-mapping work [10], which allows the mapping across the multi-fanout points but results usually in large area penalty because of the uncontrolled logic replication. The DAG-mapping ensures the delay-optimality under the constant delay assumption for cells and does not consider wire-delays based on placement. It is possible to extend the same to use in iterative mapping/placement schemes, but with the same limitations as tree-based mapping about no guarantee on either delay optimality or the convergence. Moreover, uncontrolled logic duplication may also lead to possibly increased routing congestion, affecting the convergence.

Similar to technology mapping, placement for general graphs to optimize useful objectives is a difficult problem and has been well researched over the last few decades; see [11] for the recent literature survey. The placement of special structures such as trees, however, can be performed in a polynomial time optimizing certain metrics. For example, Fischer *et al.* [12] presented the $O(n \log n)$ algorithm for the optimal placement minimizing the sum of weighted edge-lengths for a tree with n leaves; recent work includes a linear time algorithm to minimize the sum of half-perimeter wirelengths for all nets in a tree [13]. The special case of linear placement for trees is also studied well and several exact polynomial time algorithms exist to minimize total wirelength or the cutwidth, for instance, Yannakakis's algorithm [14] employed in [4] to perform simultaneous mapping and linear placement. However, the problem of delay-optimal placement for trees seems to have received relatively little attention in the published literature, despite the potential usefulness of the solution.

C. Our Contributions

Since the technology mapping and placement have great impact on the overall delays in the circuit, exploring these two spaces simultaneously can result in circuits with better delays than the conventional approach of searching those sequentially, which results in the search in a relatively small solution space. A fundamental contribution of this paper is an exact polynomial time, $O(nm^2 f_{max} P_{max}^2)$, algorithm for delay-optimal simultaneous technology mapping and 2-D placement of trees, where n, m, f_{max}, and P_{max} are the number of nodes in the tree, the number of candidate locations in 2-D area, maximum fanin over all the matches at any node, and the maximum number of matches at any node in the tree, respectively. The algorithm is based on the extension of an exact polynomial time, $O(nm^2 f_{max})$, delay-optimal placement algorithm for trees, which is another important contribution. To optimize timing in DAGs, we propose an iterative algorithm, based on LR technique, which employs the simultaneous technology mapping and placement in the inner loop. The comparison of results on IWLS'05, ITC'99, and ISCAS'85 benchmarks, with a cell library characterized for a 70 nm technology, due to the algorithm with those due to the conventional iterative delay-oriented mapping in SIS [15] and timing-driven placement mPL [16] shows more than 12% (60%) delay (slack) improvement with seven times speed-up in runtime, on average, implying that the proposed algorithms are practical and can be employed to optimize timing during physical synthesis. The placement of flip-flops can be handled by placing them in the first round of placement and subsequently performing our methods on the rest of combinational circuit. The technology mapping part typically does not apply to sequentials, since the mapping is trivial, i.e., flip-flops/latches with/without set/reset signals are mapped on to corresponding ones in the library directly. The placement algorithms presented in this paper can still place flip-flops or latches as long as those are part of a tree.

The rest of this paper is organized as follows. Section II describes the formal notation employed in this paper. Section III presents an algorithm for delay-optimal placement of trees. Section IV extends the algorithm to perform delay-optimal simultaneous technology mapping and placement. Section V describes the algorithm based on LR for simultaneous mapping and placement for DAGs. Section VI introduces a placement density constraint into the simultaneous mapping and placement problem formulation and uses LR to solve the updated problem. Section VII discusses the results due to the algorithms and compares them with those due to the competitive approach, and Section VIII concludes the paper.

II. Preliminaries

Traditionally, a technology-independent Boolean network is first decomposed into a circuit containing only primitives such as two-input NANDs and inverters, which are then mapped on to standard cells in a library during technology mapping to create a mapped netlist. Subsequently, placement is carried out on the mapped netlist to assign each cell a location in a given area. The graph structure underlying either the Boolean network or the technology decomposed circuit or the mapped netlist is a DAG $G(V, E)$, where a node $v \in V$ represents a standard cell in case of a mapped netlist or a primitive in case of the technology decomposed circuit. The primary inputs and outputs of the DAG are denoted by $input(G)$ and $output(G)$, respectively. Each directed edge $e(v_i, v_j) \in E$ represents a net whose driver (receiver) is the standard cell represented by v_i (v_j). Each node $v_i \in V$ is associated with the actual (required) arrival time a_i (q_i); the slack for the node is computed as $q_i - a_i$. The delay between nodes v_i and v_j is denoted by $d(v_i, v_j)$, which comprises the cell-delay, $d^{cell}(v_i)$, and the wire-delay, $d^{wire}(e(v_i, v_j))$. The timing constraint on a timing arc (v_i, v_j) can then be expressed in nodal form as $a_j \geq a_i + d^{cell}(v_i) + d^{wire}(e(v_i, v_j))$, where the cell-delay is not counted in the arrival time at the cell itself, but counted into the downstream delay. For a primary input i to the circuit, $d^{cell}(i)$ is simply the actual arrival time of that input. The delay of an input-output path π is denoted by $d(\pi) = \sum_{(v_i, v_j)\in\pi} d(v_i, v_j)$. The slack of the path is computed as $s(\pi) = q - d(\pi)$, where q is the required arrival time at the output of the path. Paths with the minimum slack are critical paths in the circuit. A rooted tree is a tree $T(V_T, E_T)$, with one of its nodes designated as a root. The tree may be a part of a DAG $G(V, E)$, i.e., $V_T \subseteq V$, $E_T \subseteq E$. The inputs to the tree are referred to as leaves.

III. Delay-Optimal Tree Placement

In this section, we introduce a polynomial time algorithm for the delay-optimal placement of a tree and describe its extension to simultaneous mapping and placement in the next.

We want to place a tree $T(V_T, E_T)$ in a layout area, which is divided into bins or tiles, similar to those in conventional global placement [16]. Specifically, we want to assign each node $v \in V_T$ a bin (x, y). The leaves of the tree and the root are assumed to have fixed locations. There are several possible placements leading to different delays, since the wire-delays and cell-delays are functions of the locations of the driver and the receiver. Among these placements, we want to find one with the minimum delay. Formally, the problem of delay minimization during tree placement can be stated as follows.

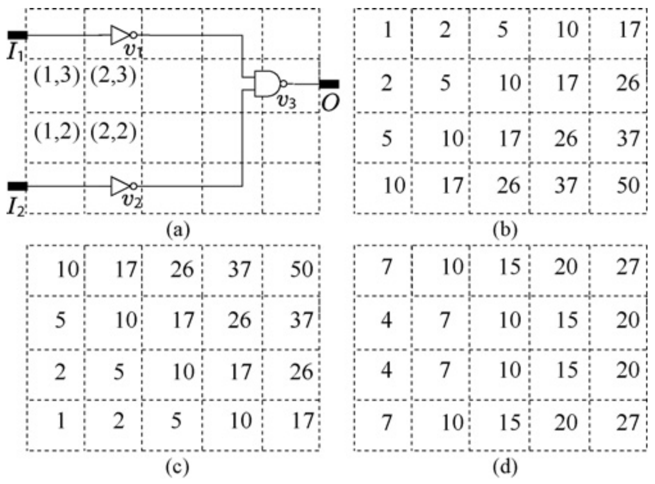
Problem Definition 3.1: Given a tree $T(V_T, E_T)$, and a set of candidate locations, Z_i, for each node v_i

$$\text{Min:} \max_{\pi\in input-root\ \ paths} d(\pi)$$

$$\text{s.t. } (x_i, y_i) \in Z_i, \quad \forall v_i \in V_T.$$

In a legal solution to the problem above, there should not be any pair of cells overlapping with each other geographically. This legalization requirement is taken care of in our method by dealing with placement density constraint and the final legalization step, which arranges the cells in each individual bin.

The delay-optimal tree placement problem has optimal substructure, i.e., the delay-optimal placement for a tree rooted at a node v contains the delay-optimal placements for subtrees rooted at its fanins, since, otherwise we can change the placement for the subtrees to yield delays smaller than that due to the delay-optimal placement for the tree, leading to a contradiction. We exploit this optimal substructure property to come up with a tree placement algorithm based on the DP. The tree placement algorithm has two phases: first phase of bottom-up solution generation that includes the construction of placement-delay tables and the second phase of actually choosing a placement from those solutions, given the fixed location of the root.



(b)

1	2	5	10	17
2	5	10	17	26
5	10	17	26	37
10	17	26	37	50

(c)

10	17	26	37	50
5	10	17	26	37
2	5	10	17	26
1	2	5	10	17

(d)

7	10	15	20	27
4	7	10	15	20
4	7	10	15	20
7	10	15	20	27

Fig. 1. (a) Tree with fixed I/Os I_1, I_2, O, and cells v_1, v_2, and v_3, placeable in 4×5 grid. (b) Placement-delay table for v_1, where the entry in bin (i, j) indicates the delay of the subtree rooted at v_1, when v_1 is placed in (i, j). (c) Placement-delay table for v_2. (d) Placement-delay table for v_3, obtained by using the optimal locations for fanins v_1 and v_2.

A. Construction of Placement-Delay Tables

The first phase of construction of placement-delay table traverses the tree in a topological order and stores the delays due to optimal placements for subtrees rooted at all nodes, assuming that the roots are fixed in all possible candidate locations. It can be explained employing the example in Fig. 1(a), where a tree with fixed locations for inputs I_1, I_2, and an output O is shown. The cells v_1, v_2, and v_3 are to be placed in a 4×5 grid so that delay on any path from I_1 or I_2 to O is minimum. For the sake of illustration, the following assumptions are made: inputs arrive at 0, the cell-delay for v_1, v_2, and v_3 is 1, and the wire-delay equals the square of Manhattan distance between nodes, which is same as the Elmore delay model with unit resistance and capacitance per unit wirelength. Consider a location (1, 2) for the cell v_1; the delay for the subtree rooted at v_1 is sum of the arrival time at I_1, $d^{cell}(I_1) = 0$, the wire-delay from I_1 to v_1, $d^{wire}(e(I_1, v_1)) = (|1 - 1| + |4 - 2|)^2 = 4$, and the cell-delay for v_1, $d^{cell}(v_1) = 1$. Therefore, the optimal delay of the subtree rooted at v_1, when the location of v_1 is fixed at (1, 2), is 5. Similarly, when v_1 is fixed at (1, 3), the optimal delay for the subtree rooted at v_1 is 2, since the wire-delay $d^{wire}(e(I_1, v_1)) = (|1 - 1| + |4 - 3|)^2 = 1$ and the cell-delay is also 1. There are 20 possible locations for v_1 and for each of those locations, the optimal delays for the subtree rooted at v_1 are shown in Fig. 1(b) depicting a table, referred to as a placement-delay table. Notice that the delay values in bins (1, 2) and (1, 3) are 5 and 2, respectively, as explained before; the delay values in other bins are derived similarly. The placement-delay table for v_2 can be constructed in a similar fashion and is shown in Fig. 1(c). The tables are constructed for nodes v_1 and v_2 before generating that for v_3, since these nodes occur before v_3 in the topological order. Now, consider the construction of the placement-delay table for v_3. For each position (x, y) for v_3, we consider the optimum location of v_1 and v_2 to compute the delay. Therefore, when v_3 is placed in (1, 1), the location chosen for v_2 is also (1, 1), since that yields the minimum delay of the path from I_2 to v_3, which is 2 (1, optimal delay for the subtree at v_2, when v_2 is fixed at (1, 1), $+0^2$, wire-delay, $+1$, cell-delay for v_3). Similarly, two locations (1, 2) and (1, 3) for v_1 result in the least path delay of 7. Choosing either of those leads to the same delay, which is minimum for the path from I_1 to v_3, when v_3 itself is placed at (1, 1). The overall delay for the subtree rooted at v_3, when it is placed in (1, 1) is $max(2, 7) = 7$; this is reflected in the bin (1, 1) in placement-delay table for v_3, shown in Fig. 1(d). Other entries in the table are derived similarly. Thus, each entry at (x, y) location in placement-delay table for a node v corresponds to the optimal delay of the subtree rooted at v, when v itself is fixed at (x, y), and is computed as follows:

$$a_v(x, y) = max_{i \in fanin(v)}\{min_{\forall (x_i, y_i) \text{locations of } i}\{a_i(x_i, y_i) + d^{wire}(e(i, v)) + d^{cell}(v)\}\}. \quad (1)$$

The following proposition states the optimality of the delay values stored in placement-delay table for all nodes.

Proposition 1: The delay $a_v(x, y)$ is the optimal delay for the placement of the subtree rooted at v, when v is fixed at (x, y).

Proof: We use induction on the depth of the node. Basis step: depth = 1. In this case, all fanins to the node v are from fixed leaf nodes. If v is also fixed at (x, y), then there is only one possible delay for the subtree rooted at v and, therefore, $a_v(x, y)$ is trivially optimal. Induction step: depth > 1. Assume that the proposition is true for all the nodes with depth $< k$. We will prove that it is true for a node with depth k. Consider such a node v, for which $a_v(x, y)$ is given by (1). Suppose $a_v(x, y)$ is not optimal. This implies that there exist some fanin node i, for which $a_i(x_i, y_i)$ is not optimal—a contradiction, since the depth of i is $< k$, because of which $a_i(x_i, y_i)$ is optimal. Therefore, $a_v(x, y)$ must also be optimal. ∎

Note that 1 is derived assuming constant cell-delay model, for the sake of simplicity. However, it can be extended to the load-dependent cell-delay model. Assume that the cell-delay is given by $\alpha C^{load} + \delta$, where α and δ are constants for a given cell, and C^{load} is the load seen by the cell. Then, the above proof also applies for the equation as follows:

$$a_v(x, y) = max_{i \in fanin(v)}\{min_{\forall (x_i, y_i) \text{locations of } i}\{a_i(x_i, y_i) + \text{cell-delay}(i) + d^{wire}(e(i, v))\}\}$$
$$a_v(x, y) = max_{i \in fanin(v)}\{min_{\forall (x_i, y_i) \text{locations of } i}\{a_i(x_i, y_i) + \alpha_i C^{load}_{wire(i,v)+input(v)} + \delta_i + d^{wire}(e(i, v))\}\}. \quad (2)$$

The difference in 1 and the above one is that $a_v(x, y)$ refers to the maximum arrival time at the output of the node v in 1, whereas in 2 it means the maximum arrival time at any of the inputs to the node v. Apart from those differences in delay/arrival time calculations, to accommodate the load-dependent delay, rest of the algorithms presented in this paper stay the same; the optimalities of the algorithms for the tree placement and simultaneous tree mapping and placement also hold regardless of cell-delay or wire-delay model. It is worth noting that using constant cell-delay model,

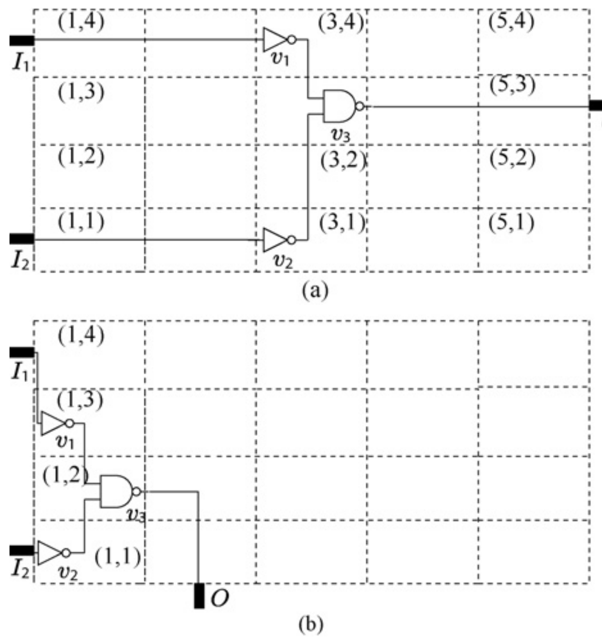


Fig. 2. Delay-optimal placements vary depending on where the output is. (a) Delay-optimal placement corresponding to the location of output O fixed at (5, 3); the corresponding delay at O is 14. (b) Delay-optimal placement corresponding to the location of output O fixed at (2, 1); the corresponding delay at O is 8. In either case, the placements for v_3, v_2, and v_1 are chosen from the placement-delay tables in Fig. 1, in reverse topological order.

the placement-delay tables can also be constructed for DAGs, but the delay-optimal DAG placement still faces a difficulty in the second step of actually choosing locations from placement-delay tables. The reason is that in case of a DAG multiple paths from root(s) to a node may imply different placements for that node. This situation does not arise in case of trees, since there is a unique path from the root to any node. It is also worth noting that if load-dependent cell-delay model is employed for DAG placement, it is difficult even to construct placement-delay tables in polynomial time, since the delays at the outputs of the multi-fanout points depend on the placement of multiple receivers. These observations are in line with the difficulties, due to computation of load at multi-fanout points, faced in case of technology mapping or sizing problems optimizing delays in DAGs.

B. Choosing Locations from Placement-Delay Tables

After the construction of placement-delay tables, the second phase of the algorithm proceeds, traversing the tree in a reverse topological order to choose the locations for v_3, v_2, and v_1. Since the root node O is fixed in the location (5, 3), as shown in Fig. 1(a), the optimal location of v_3, which results in the minimum delay, is (3, 3). It yields the delay of 14 (10, $a_{v_3}(3, 3)$, i.e., delay of the subtree rooted at v_3, $+2^2$, wire-delay from (3, 3) to (5, 3)). Note that, $a_{v_3}(3, 3) = 10$ is from the placement-delay table for v_3, shown in Fig. 1(d), created during the first step. The optimal locations of v_1 and v_2, which resulted in the delay of 10 for the subtree rooted at v_3 are (3, 4) and (3, 1), respectively; these are found out in a constant time by storing additional information (the locations of fanins leading to that delay—l_{opt} on line 7 in Algorithm

Algorithm 1 *PlaceTree*(T)

```
1: for all $v_j$ in $V_T$ in topological order do
2:   for all tiles $(x_j, y_j)$ in candidate locations set of $v_j$ do
3:     for all fanins $v_i$ of node $v_j$ do
4:       Choose $(x_i, y_i)$, the location for $v_i$, which yields the minimum value for delay $d(v_i, v_j) + a(v_i)$.
5:     end for
6:     Update arrival time:
         $a_{v_j}(x_j, y_j) = \max_{v_i \in fanin(v_j)}(d(v_i, v_j) + a(v_i))$
7:     Record corresponding optimal fanin locations:
         $\forall v_i \in fanin(v_j), l_{opt}(v_i, v_j, x_j, y_j) = (x_i, y_i)$
8:   end for
9: end for
10: for all $v_j$ in $V_T$ in reverse topological order do
11:   if $v_j$ != root($T$) then
12:     f = fanout($v_j$)
13:     placement($v_j$) = $l_{opt}(v_j, f, x_f, y_f)$
14:   end if
15: end for
```

1) along with the placement-delay table. Thus, the optimal placement for the tree is as follows: $v_1(x_{opt}, y_{opt}) = (3, 4)$, $v_2(x_{opt}, y_{opt}) = (3, 1)$, and $v_3(x_{opt}, y_{opt}) = (3, 3)$. This placement is shown in Fig. 2(a). However, if the root node O is placed at some other location, say, (2, 1), as in Fig. 2(b), the placements of cells leading to minimum delay will be different. The corresponding placement can be found out similarly by first choosing the location of v_3, which leads to minimum delay. Such a location is (1, 2), leading to minimum delay of 8 (4, $a_{v_3}(1, 2)$, the delay due to optimal placement of subtree rooted v_3, when v_3 is placed at (1, 2), $+2^2$, wire-delay from (1, 2) to (2, 1)). The corresponding locations of v_1 and v_2 that lead to the least delay from primary input to v_3 are (1, 3) and (1, 1), respectively. This placement is shown in Fig. 2(b). Observe that the minimum delays due to the optimal placements for the cases with two different locations of root node are different and both, along with the corresponding placements, are found by the algorithm.

The delay-optimal placements vary depending on the locations of inputs and outputs. Since the algorithm stores delays due to all possible optimal placements in the first phase, it is able to select the optimum one, depending on the location of the root, in the second stage. This is possible in case of rooted tree structure, since there is a unique path from a node to the root. Therefore, the placement of a node corresponding to the minimum delays for the root can always be chosen. In case of DAGs, however, different paths from roots to a node may imply different placements for multi-fanout nodes (and for subtrees rooted at those) and choosing any of those may lead to suboptimal delays.

C. Pseudo-Code and Complexity of Delay-Optimal Tree Placement Algorithm

The pseudo-code for the tree placement is shown in Algorithm 1. It processes nodes in the tree in a topological order and for each node v_j, it considers all the possible locations (x_j, y_j). For each of those placements, it finds out the placement for each fanin resulting in the minimum delay. This operation requires $O(m \times |fanin(v_j)|)$ time, since for each node, we store the arrival times, $a_v(x, y)$, indexed by location (x, y) and these represent the optimal delays for the placement of the subtree rooted at v, when v itself is placed at (x, y). Considering the minimum arrival times from the fanins, the arrival times for the delay-optimal placements of the subtree rooted at v_j are computed and stored by indexing on the locations (x_j, y_j). Other auxiliary information such as the optimal locations of fanins for each placement of v_j is also stored so that the delay-optimal placement can be created, employing reverse topological traversal, after all the nodes are processed. The amount of memory required to store the optimal delay values and other auxiliary information for an entire tree is $O(nmf_{max})$, for the tree containing n nodes, each with m placement possibilities, and the maximum fanin of f_{max}. The time complexity of the algorithm is $O(nm^2 f_{max})$, since it is dominated by the search for the optimal-delay placement for each fanin of a given node.

Proposition 2: The tree placement procedure shown in Algorithm 1 returns optimal-delay placement.

Proof: During the topological traversal, $l_{opt}(i, v, x, y)$ is populated and it stores the delay-optimal locations for fanins i for all possible locations (x, y) of all nodes $v \in V_T$. Considering the location of the root, which is fixed, the reverse topological traversal, assigns the optimal locations to all nodes from those stored in $l_{opt}(i, v, x, y)$ based on the location of their fanouts. ■

Even though we explained the tree placement algorithm employing constant and Elmore delay models for cell-delays and wire-delays, respectively, the algorithm ensures delay-optimality with other delay models as well. For instance, asymptotic waveform evaluation can be employed to compute wire-delays and without any changes, the algorithm still ensures the optimality. Similarly, the load-dependent cell-delay models can be used, with slight changes in the computation of delays, without affecting the optimality.

IV. Delay-Optimal Simultaneous Technology Mapping and Placement for Trees

Delay-optimal tree placement algorithm presented in the previous section can be extended to perform simultaneous technology mapping and placement. Traditionally, technology mapping transforms a Boolean network containing primitive gates such as two-input NANDs and inverters into an implementation based on the set of cells in a library. It is carried out in two steps: matching and covering. For conventional delay-oriented technology mapping employing load-dependent delay model [15], the matching phase processes each node in a topological order and stores a piece-wise linear load-delay curve corresponding to mapping solutions due to non-inferior matches, found either by structural or Boolean techniques, at that node. In the covering phase, the mapping solution is generated by a reverse topological traversal, by selecting the minimum delay matches for given loads. For trees, this algorithm results in delay-optimal solution, ignoring the wire-delays based on placement. To account for placement-based

Algorithm 2 *MatchPlaceTree*(T)

```
1: for all nodes $v_j$ in topological order do
2:   for all matches $g_j$ corresponding to cells in the library do
3:     for all bins $(x_j, y_j) \in \mathcal{Z}_j$, set of candidate locations, do
4:       for all fanins $i$ of pattern $g_j$ matched at node $v_j$ do
5:         Choose $(g_i, x_i, y_i)$ that gives the minimum value of delay $d(v_i, v_j) + a(v_i)$.
6:       end for
7:       Update arrival time:
           $a_{v_j}(g_j, x_j, y_j) = \max_{i \in fanin(g_j)} (d(v_i, v_j) + a(v_i))$
         and record corresponding solutions of all its fanins: $\{(g_i, x_i, y_i) | i \in fanin(g_j)\}$
8:     end for
9:   end for
10: end for
```

wire-delays, the approaches in the paper such as [3], [5], and [6] either assume that the match is placed at some location or iterate between the mapping, placement, and technology decomposition steps. Obviously, these approaches do not claim delay-optimality considering the wire-delays based on the actual placement, even for trees.

To overcome the limitations of the previous approaches, we propose a simultaneous mapping and placement algorithm, which returns the delay-optimal mapped netlist and its placement in a polynomial time for a tree. The algorithm relies on the matching step to store both the mapping choices and their delay-optimal placements, whereas the covering phase, which is same as that in the traditional algorithm, generates a mapping solution with a reverse topological traversal by selecting the delay-optimal choices. Since all the mapping choices and their delay-optimal placements are considered, the final mapping and placement solution is optimal. The algorithm makes the same assumption, as in previous section, that the locations of the inputs and output of a tree are fixed beforehand. The inputs to the tree are either the primary inputs or outputs from the multi-fanout roots of other trees in the DAG; the output is either a primary output or serves as an input to other trees.

The pseudo-code for the matching step is shown in Algorithm 2. Similar to that in conventional approaches, it processes nodes in the tree in a topological order. For each node v_j, it considers all possible matches corresponding to the cells in the library. For each match g_j, it considers all possible placements (x_j, y_j) in $\mathcal{Z}_j$ and for each of those, it finds out the optimal-delay due to the mapping solution and the placement for each fanin (line 5 in the pseudo-code). This search for optimal delay value at each node requires $O(mP_{max})$ time, since for each node, v_j, we store optimal delay values $a_{v_j}(g_j, x_j, y_j)$ indexed by a match g_j and its placement (x_j, y_j) (line 7). The auxiliary information about the matches at the fanins and their locations is also indexed similarly and is employed during the covering phase to actually build the mapped netlist and its placement. The amount of memory required to store the optimal delay values and other auxiliary information for entire tree is $O(nmf_{max}P_{max})$, since there are n nodes with P_{max} possible matches and m placement possibilities for those matches. The time-complexity of the matching is dominated by the search for the optimal delay value choice and its location at the fanin of a match, placed at all possible locations, for a node. Since there are n nodes with P_{max} matches at most, each of which has m placement possibilities and have f_{max} fanins at most, the time complexity is $O(nm^2 f_{max} P_{max}^2)$.

V. Handling DAGs by Lagrangian Relaxation

In reality, the topologies of most circuits are DAGs. DAG topology poses significant more difficulty than tree topology to circuit design, specifically for our technology mapping and placement problem in this paper. Two well-known difficulties caused by DAGs are the reconvergent paths and the multi-fanout load estimation. Our solutions for these two difficulties are presented as follows.

The difficulty with reconvergent paths in DAGs can be illustrated by the following example. Suppose there are multiple paths from cell s to cell t, two of which are $s \rightsquigarrow u \rightarrow t$ and $s \rightsquigarrow v \rightarrow t$. The DP solution for t is composed of the DP solutions for u and for v, each of which depends on solutions for s, which may very possibly be inconsistent for the solution at u and v; the solution at s (including the mapping and placement of s that is best for u is not in general the same as the one that is best for v. This inconsistency in solutions due to reconvergent paths in DAGs cannot be solved straightforwardly, since either historical solution recording or judicious tradeoff between fanouts is very complex.

Our solution for the reconvergent path issue is an innovative method, Joint relaxation and restriction, which we proposed in [17] for gate sizing and V_t assignment. Essentially, this method relaxes the solution consistency constraint in the first stage of a topological order solution propagation, which comes up with a potentially inconsistent solutions best for different fanout paths, respectively. Then, the solution consistency constraint is enforced in the second stage by choosing the best solution at each node in a reverse topological order. The coupled relaxation and restriction strategy provides a good global view of the circuit with efficient central processing unit (CPU) runtimes. By doing this coupled relaxation and restriction iteratively, a high-quality solution for the whole circuit can be obtained when the iteration converges. The details of this method can be found in [17].

The difficulty of multi-fanout load estimation is due to the fact that different fanouts of a gate affect each other on timing, since the load capacitance to the multi-fanout node include the capacitance of all fanout cells. As a result, DP, which deals with single fanout without properly incorporating the interactive effect between different fanouts, can hardly find the overall best solution on the fanout cone. This limits the application of DP to delay-optimal mapping and placement on DAGs. This issue is illustrated by a simple example in Fig. 3.

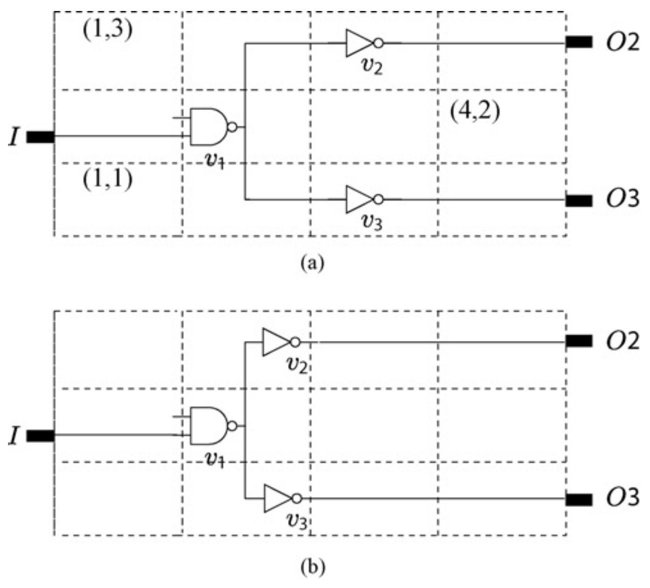


Fig. 3. (a) Two cells (v_2 and v_3) driven by a multi-fanout cell (v_1) placed on a 3×4 grid without consideration of interactive effect between multiple fanouts. I/Os and the multi-fanout cell v_1 are fixed. (b) Optimal placement of v_2 and v_3, considering the load affected by both fanouts of v_1.

Consider the placement of a NAND gate and two INV gates drives it in a 3×4 grid as in Fig. 3(a). The primary input I, two outputs O_2, O_3, and cell v_1 are fixed at the locations shown in the figure. Here, the cell-delay is load-dependent, i.e., the delay of a cell is linear to the load capacitance it drives and its resistance. In this example, we use the Elmore model for the wire-delay. For the sake of clarity, we assume unit wirelength to cross each bin and that the wire, every cell, and every I/O pin has unit resistance and unit capacitance. In this case, if we still use the tree placement algorithm in Section III to come up with the delay table for all placements of each node in a topological order traversal, the delays at v_2 and v_3 are considered independently from each other, which does not completely reflect the load-dependent cell-delay. The independent delay calculation leads to an independent placement of v_2 and v_3 as in Fig. 3(a). Cells v_1 and v_2 are uniformly spaced over the upper path, so are v_1 and v_3 over the lower path. This placement is optimal for either the upper path or the lower path, individually. However, this placement is not optimal for all the three cells, because the load v_1 drives is doubled due to multi-fanout. The best placement of v_2 and v_3 are shown in Fig. 3(b), in which v_2 and v_3 are closer to v_1 to compensate its larger load. One may argue that this issue can be resolved by estimating the overall load when considering the solution at one fanout. Unfortunately, this is not true. No matter how much the estimated load on the upper path is, v_3 still needs to be placed on the middle point between v_1 and O_3, because of the quadratic relation between wire-delay and wirelength on the lower linear path. The same happens to the placement of v_2.

To overcome the difficulty, we propose a method based on LR; it applies the simultaneous tree mapping and placement to minimize delays weighted by Lagrangian multipliers iteratively. The algorithm stops, if there is no significant improvement in the slack. The whole circuit delay is broken into timing constraints on every timing arc in nodal form. Then, the weighted delay is expressed in the form of timing arc delay summation. Timing points are at the inputs of the gates. Each timing arc, connecting two timing points, spans from the input of a cell to an input of its fanout. For example, there are two timing arcs covering gate v_1 in Fig. 3(a): one is from the input of v_1 to the input of v_2 and another is from the input of v_1 to the input of v_3. The basic idea behind the LR approach is to use weights (Lagrangian multipliers) to put different focus on different parts of timing. We will explain how the weights (Lagrangian multipliers) encode the mutual effect between multiple fanouts of a cell into our problem with more details on the LR method next.

Let $PO(G)$ be the set of primary outputs in G, and $PI(G)$ be the primary inputs in G. The mapping and placement problem in a general circuit is then formulated as follows:

Problem Definition 5.1: DAG Mapping and Placement: Given the netlist of a decomposed circuit as a DAG $G(V, E)$, a set of candidate locations $\mathcal{Z}_i$ for each gate in the circuit, and a given cell library $\mathcal{B}$, perform technology mapping and cell placement of the circuit to maximize the circuit slack as follows:

$$\begin{aligned}
&\text{Min:}\\
&\qquad -s\\
&\text{s.t.}\\
&\qquad q_i - a_i \geq s, \ \ \forall v_i \in PO(G),\\
&\qquad a_j \geq a_i + D_{ij}, \forall v_j \in V \cup PO(G), \forall v_i \in input(v_j)\\
&\qquad (x_i, y_i) \in \mathcal{Z}_i, \ \ \forall v_i \in V\\
&\qquad v_i \in g, \qquad \forall v_i \in V, \exists g \in \mathcal{B}.
\end{aligned}$$

Notice that the arrival time a_i at $v_i \in PI(G)$ and the required arrival time q_j at $v_j \in PO(G)$ are constants given by the problem.

A non-negative Lagrangian multiplier is introduced for each constraint on arrival time, the second constraint above. The Lagrangian function is a summation of the objective and weighted timing constraints as follows:

$$\begin{aligned}
L_\lambda(s, \mathbf{a}, \mathbf{v}) = -s + &\sum_{v_i \in PO(G)} \lambda_{i0}(s + a_i - q_i)\\
&+ \sum_{v_j \in V - PI(G)} \sum_{v_i \in input(v_j)} \lambda_{ij}(a_i + D_{ij} - a_j). \qquad (3)
\end{aligned}$$

Then, the LR dual problem with given multiplier values is expressed as follows:

$$\begin{aligned}
\text{Min:}\quad & L_\lambda(s, \mathbf{a}, \mathbf{v})\\
\text{s.t.}\quad & (x_i, y_i) \in \mathcal{Z}_i, \forall v_i \in V\\
& v_i \in g, \quad \forall v_i \in V, \exists g \in \mathcal{B}.
\end{aligned}$$

As shown in [18], the problem can be simplified by eliminating the arrival times in the Lagrangian function according to the Kuhn-Tucker conditions [19] as follows:

$$L_\lambda(\mathbf{v}) = \sum_{v_i \in PO(G)} \lambda_{i0} q_i + \sum_{v_j \in V-PI(G)} \sum_{v_i \in input(v_j)} \lambda_{ij} D_{ij}. \quad (4)$$

In our LR framework, there are two problems to solve. The first one is the Lagrangian subproblem solved in each Lagrangian iteration, which is to minimize $L_\lambda(\mathbf{v})$ in (4) with specific multiplier values. The other problem is the Lagrangian dual problem, which updates the multipliers at the end of each Lagrangian iteration to maximize the minimum value of $L_\lambda(\mathbf{v})$ with optimal mapping and placement solutions.

The Lagrangian subproblem is solved using our combinatorial algorithm of simultaneous mapping and placement in Section IV. The same method is employed here, except the cost function used to evaluate each mapping and placement option is different; instead of minimizing the arrival time, we choose the options to reduce the summation of weighted delays. Specifically, line 5 in Algorithm 2 changes to use the formula as follows:

$$L_\lambda(v_i) + \lambda_{ij} D_{ij} \quad (5)$$

where v_is mapping and placement solutions are under consideration for the minimum cost function value at v_j.

The Lagrangian dual problem is solved by sub-gradient [19] method. The multipliers are updated employing sub-gradients [19], following the static timing analysis on the mapping and placement solution in the current iteration. Basically, timing arcs that are more critical are updated with larger multipliers. This way, more attention is focused on the critical parts in the circuit to reduce the overall delay.

The rational of Lagrangian multipliers explains why they help resolving the difficulty caused by multi-fanout in DAGs. Consider the same example in Fig. 3. As mentioned before, cell v_1 is covered by two Lagrangian multipliers—one for (v_1, v_2) and another one for (v_1, v_3). The weight on v_1s cell-delay is the summation of the two multipliers, thus it is higher than the weight on v_2 or v_3. As a result, in order to minimize the total weighted sum of delays, it is better to reduce the load of v_1 at the cost of increasing load of v_2 or v_3. Consequently, the DP applied on each of the two fanouts of v_1 would put v_2 and v_3 closer to v_1, specifically in bins (2, 3) and (2, 1). Therefore, using LR the best overall solutions can be found in this case.

The time complexity of our algorithm is dominated by the number of iterations in LR and the matching phase, whose complexity is same as that of *MatchPlaceTree*(T) in the previous section, since the simultaneous mapping/placement is carried out on individual trees in the DAG.

VI. Handling Placement Density Constraint

To this point, our algorithms ignore the possibility of overcrowded areas during cell placement. Cell overlapping may still happen, when re-placement is performed with carefully selected candidate locations for each cell in the whole under-utilized placable area. This overcrowding issue still needs to be taken care of, because a violation of non-overlapping constraint may result in unexpected timing penalty in following legalization stage, which resolves cell overlapping. Therefore, it is better to deal with the overlapping risk early during our cell placement by controlling the placement density in small tiles, each of which is composed of multiple bins, and all of which together form the whole placement area. We take this approach and enforce the density constraint on small tiles in our cell placement.

Suppose the whole placement area is divided into many small tiles, the kth of which is denoted by $\mathcal{Y}_k$. Let the upper bound of the tile density be γ, i.e., $\frac{\sum_{(x_i,y_i)\in\mathcal{Y}_k} |v_i|}{|\mathcal{Y}_k|} \le \gamma$ should hold, where $|v_i|$ and $|\mathcal{Y}_k|$ represent the area of the ith cell and the kth tile, respectively. Then, the formulation of our simultaneous mapping and placement problem can be updated as follows.

Problem Definition 6.1: Density-Constrained DAG Mapping and Placement: Given the netlist of a decomposed circuit as a DAG $G(V, E)$, a set of candidate locations $\mathcal{Z}_i$ for each gate in the circuit, a tile density constraint γ, and a given cell library $\mathcal{B}$, perform technology mapping and cell placement of the circuit to maximize the circuit slack as follows:

Min:

$$-s$$

s.t.

$$\begin{aligned} q_i - a_i &\ge s, && \forall v_i \in PO(G) \\ a_i &\ge a_j + D_{ji}, && \forall v_i \in V \cup PO(G), \forall v_j \in input(v_i) \\ (x_i, y_i) &\in \mathcal{Z}_i, && \forall v_i \in V \\ v_i &\in g, && \forall v_i \in V, \exists g \in \mathcal{B} \\ \frac{\sum_{(x_i,y_i)\in\mathcal{Y}_k} |v_i|}{|\mathcal{Y}_k|} &\le \gamma, \forall \mathcal{Y}_k. \end{aligned}$$

To solve this problem with extra density constraint on tiles, we employ LR again. Similar to how we deal with arrival time constraints, we turn the density constraint into a penalty term in the Lagrangian function (the cost function). Each density constraint on a specific tile $\mathcal{Y}_k$ is assigned with a Lagrangian multiplier μ_k. Thus, the Lagrangian function becomes as follows:

$$L_\lambda(\mathbf{v}) = \sum_{v_i \in PO(G)} \lambda_{i0} q_i + \sum_{v_j \in V-PI(G)} \sum_{v_i \in input(v_j)} \lambda_{ij} D_{ij} + \sum_{\mathcal{Y}_k} \mu_k \left(\frac{\sum_{(x_i,y_i)\in\mathcal{Y}_k} |v_i|}{|\mathcal{Y}_k|} - \gamma \right). \quad (6)$$

In each Lagrangian iteration, the subproblem of minimizing the Lagrangian function is solved using our combinatorial mapping and placement algorithm. The only difference induced by this subproblem is the cost function value in the characterization of each solution during the solution search. Specifically, to perform the task here Algorithm 2 is modified on line 5 using the formula as follows:

$$L_\lambda(v_i) + \lambda_{ij} D_{ij} + \mu_k \frac{|v_j|}{|\mathcal{Y}_k|} \quad (7)$$

where $\mathcal{Y}_k$ is the tile where the current candidate location of v_j resides, i.e., $(x_j, y_j) \in \mathcal{Y}_k$.

The Lagrangian dual problem is also solved by updating the multipliers using sub-gradient method. Besides the multipliers for timing constraints updated according to criticality on different timing arcs, the multipliers for tile placement density are updated to impose higher cost on tiles that are too crowded. Therefore, in the succeeding subproblem-solving iteration, the cells are pushed away from overcrowded tiles to tiles with lower density. This can be viewed as an analog to a flow driven by the difference of potential (multiplier) at different spots (tiles).

VII. Experimental Results

The algorithms described in this paper are implemented in a C++ program on the Windows platform with a 3.0 GHz Pentium IV processor and 4 GB memory. To evaluate the efficacy of the algorithms, the experiments are run on the set of ISCAS'85 benchmark circuits, and selected combinational circuits from ITC'99 and IWLS'05 benchmarks, with a standard cell library, including 64 gates, characterized employing 70 nm technology parameters [20]. Before the experiment, each benchmark circuit is synthesized using SIS and placed by mPL6. The initial technology mapping is performed for area reduction with SIS. In the initial mPL6 placement, the input pins are fixed along the boundary edges starting at (0, 0), while the output pins are fixed along the opposite boundary edges ending at (max_x, max_y). The primary inputs and outputs of the benchmark circuits are fixed at these locations throughout the experiment. The dimensions of the placement area used in the experiment is obtained from the initial placement. The clock period of a circuit is set to the largest path delay in the circuit after the initial synthesis and placement. Elmore delay model is used in the experiments. Typical cell utilization is around 50% for each of the benchmarks, which is normally the case for average synthesizable blocks in high performance microprocessor circuits. The results due to the four iterative approaches, whose goal is to maximize the worst case slack, are compared as follows.

1) *Conventional:* in this case, each iteration performs conventional delay-oriented technology mapping followed by timing-driven placement. The technology mapping algorithm is similar to that in [15], which is modified to consider the wire-delays based on placements, whereas the timing-driven placement is implemented by incorporating the timing aware net weighting technique [21] with mPL6 [16].
2) *Conventional with delay-optimal tree placement:* in each iteration, timing critical trees are optimized by conventional technology mapping followed by the delay-optimal tree placement algorithm described in Section III.
3) *Simultaneous delay-optimal tree mapping and placement:* in each iteration, timing critical trees are optimized by simultaneous mapping and placement algorithm discussed in Section IV.
4) *LR with simultaneous DAG mapping and placement:* in each iteration, timing critical cones are optimized by the LR-based extension of simultaneous tree mapping and placement to DAGs, presented in Section V.

The stopping criterion for all the approaches is less than 10 ps slack improvement in consecutive iterations. In our tree and DAG optimization methods, the incremental placement radius is set to 3 bins. That is, in each iteration a cell is not allowed to move in any direction beyond 3 bins away from its starting location at the beginning of the iteration. For all our tree and DAG approaches, the final placement is reached by legalizing the outcome of the tree placement or the DAG placement. The average cell utilization after legalization is 50–60% in our experiments. The results due to all the approaches are shown in Table I. Results for ISCAS'85, ITC'99, and IWLS'05 are listed in three boxes in the table, respectively. As compared to the conventional approach, LR-based algorithm improves the average slacks and maximum delays by 64–69% and 11–14%, respectively, with about seven times speed-up in the runtime. Similarly, tree based simultaneous mapping and placement leads to 59–62% and 7–13% improvements in the slacks and delays, respectively, with approximately two orders of magnitude smaller runtimes. The improvement in runtimes over the conventional approach comes from the absence of timing-driven net-weighting and the placement of whole circuit. Moreover, the conventional approach is likely to be more susceptible for divergence than tree placement or simultaneous tree mapping and placement. Even in case of LR approach, after the first iteration, we allow the placement of the cells within only certain radius, which, although, reduces the placement search space, still allows the complete exploration of the mapping space and ensures placement stability. The improvements highlight the fact that the simultaneous exploration of the mapping and placement spaces can lead to the timing convergence not only faster but also with better quality than exploring the mapping and the placement spaces separately, as in the conventional approach. One can observe that the proposed methods have limited impact on wire length and cell area, although these are not included in the problem formulation. The results due to employing only tree placement to improve timing show that it increases wire length and cell area marginally, but still improves the slacks considerably. This shows that employing simultaneous mapping and placement may be a better approach than applying delay-oriented mapping and placement separately, since the technology mapping which considers the wire-delays based on placement is sensitive to the placement of the subject graph and considering only center of gravity placements for the matches, as opposed to all possible placements in simultaneous mapping and placement approaches, limits the optimization scope.

In order to test the effectiveness of our method coping with the placement density constraint, we run our method on a series of problem settings with different cell utilizations. Three benchmark circuits from ISCAS'85, ITC'99, and IWLS'05, respectively, are chosen in this experiment. Each of the three circuits is tested with four cell utilization: 50%, 60%, 65%, and 70%. Higher cell utilization indicates smaller size of the placement area and, therefore, the cells are potentially placed

TABLE I
COMPARISON OF CONVENTIONAL DELAY-ORIENTED MAPPING FOLLOWED BY TIMING-DRIVEN PLACEMENT WITH PROPOSED APPROACHES EMPLOYING ONLY TREE PLACEMENT, SIMULTANEOUS TREE MAPPING AND PLACEMENT, AND LR WITH SIMULTANEOUS MAPPING AND PLACEMENT

	Conventional			Conventional Mapping with Tree Placement					Simultaneous Tree Mapping and Placement					LR with Simultaneous Mapping and Placement				
Circuit	Delay	Slack	CPU	Delay	Slack	CPU	Wire	Area	Delay	Slack	CPU	Wire	Area	Delay	Slack	CPU	Wire	Area
C432	1091	59	148	966	184	2	1.12	1.03	932	218	2	0.83	1.03	921	229	47	0.99	0.98
C499	1043	57	254	1003	97	2	1.57	1.00	933	167	2	1.01	1.13	925	175	31	1.12	1.09
C880	989	11	140	826	174	1	1.51	1.00	803	197	1	0.92	1.02	788	212	29	0.95	1.00
C1355	1240	60	193	1101	199	3	0.88	1.00	1099	201	1	0.94	1.01	1029	271	35	0.95	1.002
C1908	1465	85	290	1286	264	2	0.97	1.00	1221	329	2	0.92	0.96	1203	347	39	0.96	0.97
C2670	1229	71	564	1068	232	4	1.40	1.01	1039	261	6	1.03	1.07	1020	280	42	1.01	1.00
C3540	1760	90	637	1705	145	15	0.99	1.06	1672	178	43	1.00	1.08	1593	257	395	1.07	0.98
C5315	2011	89	1101	1894	206	12	1.01	1.00	1820	280	12	1.03	0.99	1799	301	102	1.02	1.00
C6288	5191	159	1118	5250	100	25	1.12	1.00	5169	181	14	1.00	0.81	5148	202	69	0.99	1.007
C7552	1465	85	2555	1431	119	11	1.10	1.00	1416	134	12	1.08	1.04	1307	243	165	1.06	1.008
Average	1748	77	700	1653	172	7.7			1610	215	9.5			1573	251	95		
Normalized	1	1	1	0.95	2.2	0.011	1.10	1.04	0.92	2.8	0.014	1.02	0.99	0.90	3.26	0.136	1.01	1.003
B14	3790	150	2025	3615	325	61	1.03	1.03	3574	366	51	0.9	1.03	3533	407	259	1.01	1.03
B15	4185	325	1302	4005	505	100	1.00	1.00	3792	718	268	1.01	1.02	3549	961	587	1.02	0.98
B20	4857	343	7154	4830	370	250	1.09	1.00	4296	904	232	1.11	1.00	4281	919	862	1.05	0.99
Average	4277	818	3493	4150	1200	137			3887	1988	183			3788	2287	569		
Normalized	1	1	1	0.97	1.46	0.04	1.05	1.007	0.884	2.43	0.05	1.001	1.012	0.881	2.80	0.163	1.02	0.998
USBfunct	8240	510	7501	8036	714	496	1.06	1.02	7392	1358	309	1.05	1.06	7103	1620	933	1.06	1.08
AES core	3512	282	10228	3493	301	536	1.02	1.01	3278	516	502	0.99	1.01	3193	601	1282	1.02	0.99
Average	5876	396	8865	5765	508	516			5335	937	406			5148	1110	1108		
Normalized	1	1	1	0.98	1.28	0.058	1.03	1.01	0.91	2.37	0.046	1.01	1.03	0.88	2.80	0.125	1.04	1.02

The maximum path delay and the minimum slack are in per second, CPU time is in seconds, and total wirelength, cell area are normalized with respect to the corresponding quantities resulting from the conventional approach.

TABLE II
COMPARISON OF CONVENTIONAL DELAY-ORIENTED MAPPING FOLLOWED BY TIMING-DRIVEN PLACEMENT TO PROPOSED APPROACH OF LR WITH SIMULTANEOUS MAPPING AND PLACEMENT, ON DIFFERENT CELL UTILIZATION OF BENCHMARK CIRCUIT C7552, B20, AND AES CORE

				Conventional		LR with Simultaneous Mapping and Placement			
Circuit	No. of Gates	Memory (MB)	Cell Utilization (%)	Delay	CPU	Delay	Norm. Delay	CPU	Norm. CPU
C7552	3115	108.8	50	1465	2555	1307	0.89	165	0.06
			60	1326	2521	1160	0.88	171	0.07
			65	1298	2510	1101	0.85	189	0.08
			70	1252	2568	1076	0.86	220	0.09
B20	10 590	318.6	50	4857	7154	4281	0.88	862	0.12
			60	4206	7132	3659	0.87	891	0.12
			65	3989	7098	3450	0.86	920	0.13
			70	3796	7220	3188	0.84	1056	0.15
AES core	15 692	462.5	50	3512	10 228	3193	0.91	1282	0.13
			60	3298	10 189	2810	0.85	1297	0.13
			65	3201	10 172	2762	0.86	1418	0.14
			70	3121	10 312	2695	0.86	1506	0.15

The maximum path delay is in per second and CPU time is in seconds.

closer to each other. The primary inputs and outputs of the circuits are proportionally placed on the boundary according to the locations in the original placement area. The results with different cell utilizations are summarized in Table II. Essentially, our method maintains the advantage in circuit delay minimization over the conventional method. Due to the global view in our method, it may provide wider choices of placement options, so sometime its delay minimization advantage is enhanced in higher cell utilization problem setting. Because of the effort to meet the density constraint in our LR framework under higher cell utilization setting, our method takes more time to converge to a legal solution and, therefore, the runtime increases with the cell utilization. In addition, from columns 2 and 3 in Table II, one can see that the amount of memory used in running our methods scales well (linearly) to the size of the circuit.

VIII. CONCLUSION

In this paper, we proposed exact polynomial time algorithms for delay-optimal placement as well as simultaneous technology mapping and placement for trees. We extended the simultaneous mapping and placement algorithm to DAGs and placement density constraints using a LR technique. Compared to the conventional iterative mapping and timing-driven placement approach, our methods improve the slacks by more than 60%, with at least seven times speed-up, and have negligible impact on total wirelength and cell area. Based on the fully simultaneous cell placement and technology mapping approach in this paper, a potential future work is to explore different degree of synergy between placement and mapping, e.g., by performing incremental placement following incremental mapping.

References

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

[2] K. Chaudhary and M. Pedram, "A near optimal algorithm for technology mapping minimizing area under delay constraints," in *Proc. IEEE/ACM DAC*, Jun. 1992, pp. 492–498.

[3] M. Pedram and N. Bhat, "Layout driven technology mapping," in *Proc. IEEE/ACM DAC*, Mar. 1991, pp. 99–105.

[4] J. Lou, A. H. Salek, and M. Pedram, "An exact solution to simultaneous technology mapping and linear placement problem," in *Proc. ACM/IEEE ICCAD*, Nov. 1997, pp. 671–675.

[5] J. Y. Lin, A. Jagannathan, and J. Cong, "Placement-driven technology mapping for LUT-based FPGAs," in *Proc. ISFPGA*, 2003, pp. 121–126.

[6] W. Gosti, S. R. Khatri, and A. L. Sangiovanni-Vincentelli, "Addressing timing closure problem by integrating logic optimization and placement," in *Proc. ACM/IEEE ICCAD*, Nov. 2001, pp. 224–231.

[7] D. Pandini, L. T. Pileggi, and A. J. Strojwas, "Global and local congestion optimization in technology mapping," *IEEE Trans. Comput.-Aided Des.*, vol. 22, no. 4, pp. 498–505, Apr. 2003.

[8] M. Hrkić, J. Lillis, and G. Beraudo, "An approach to placement-coupled logic replication," in *Proc. IEEE/ACM DAC*, Jun. 2004, pp. 711–716.

[9] X. Wang and S. Burns, "Technology mapping using a fixed delay and variable area-power model," in *Proc. IWLS*, Jun. 2007.

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

[11] J. Cong, J. Shinnerl, M. Xie, T. Kong, and X. Yuan, "Large-scale circuit placement," *ACM Trans. Des. Automat. Electron. Syst.*, vol. 10, no. 2, pp. 1–42, 2005.

[12] M. Fischer and M. Paterson, "Optimal tree layout (preliminary version)," in *Proc. STOC*, 1980, pp. 177–189.

[13] S. Chatterjee, Z. Wei, A. Mischenko, and R. Brayton, "A linear time algorithm for optimum tree placement," in *Proc. IWLS*, Jun. 2007.

[14] M. Yannakakis, "A polynomial algorithm for the min-cut linear arrangement of trees," *J. ACM*, vol. 32, no. 4, pp. 950–988, 1985.

[15] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Dept. Electric. Eng. Comput. Sci., College Eng., Univ. California, Berkeley, Tech. Rep. UCB/ERL M92/41, 1992.

[16] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *Proc. ISPD*, 2005, pp. 185–192.

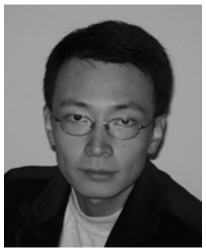
[17] Y. Liu and J. Hu, "A new algorithm for simultanueous gate sizing and threshold voltage assignment," *IEEE Trans. Comput.-Aided Des.*, vol. 29, no. 2, pp. 223–234, Feb. 2010.

[18] C. Chen, C. C. N. Chu, and D. F. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Trans. Comput.-Aided Des.*, vol. 18, no. 7, pp. 1014–1025, Jul. 1999.

[19] M. Bazaraa, H. Sherali, and C. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. New York: Wiley, 2003.

[20] *Berkeley Predictive Technology Model* [Online]. Available: http://www-device.eecs.berkeley.edu/ptm/download.html

[21] A. Marquardt, V. Betz, and J. Rose, "Timing-driven placement for FPGAs," in *Proc. ISFPGA*, 2000, pp. 203–213.



Yifang Liu (S'09) received the B.S. degree in computer engineering from the University of Electronic Science and Technology, Chengdu, China, the M.S. degree in computer science from the University of Maryland, College Park, and the Ph.D. degree in electrical and computer engineering from Texas A&M University, College Station.

He was with IBM T. J. Watson Research Center, Yorktown Heights, NY, in 2008. Currently, he is a Software Engineer with Google Inc., Mountain View, CA. He is a co-author of the book *GPU Computing Gems* (Elsevier, 2010). His current research interests include algorithm design and analysis, applied optimization, and statistical modeling and optimization in computer-aided design for very large scale integrated circuits.

Dr. Liu is an elected member of Phi Kappa Phi honor society.

Rupesh S. Shelar (S'00–M'05) received the B.E. degree in instrumentation engineering from the Marathwada University, Aurangabad, India, in 1997, the M.Tech. degree in electrical engineering with specialization in microelectronics from the Indian Institute of Technology, Mumbai, India, in 1999, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2004.

He is currently a Staff Design Engineer in the low power IA Group at Intel Corporation, Hillsboro, OR. He has contributed to Core i5/Core i7 microprocessor designs in the areas of local/global clocking and interconnect impact analysis on timing/power. He is a co-author of the book *Routing Congestion in VLSI Circuits: Estimation and Optimization* (Springer, 2007).

Jiang Hu (M'01–SM'07) received the B.S. degree in optical engineering from Zhejiang University, Zhejiang, China, in 1990, the M.S. degree in physics in 1997, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2001.

He was with IBM Microelectronics, Armonk, NY, from 2001 to 2002. Currently, he is an Associate Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. His current research interests include computer-aided design for very large scale integrated circuits, especially on interconnect optimization, clock network synthesis, variation tolerance technology, and design for manufacturability.

Dr. Hu was the recipient of a Best Paper Award at the ACM/IEEE Design Automation Conference in 2001 and an IBM Invention Achievement Award in 2003. He has served as a Technical Program Committee Member for DAC, ICCAD, ISPD, ISQED, ICCD, DATE, and ISCAS. He is the Technical Program Chair for ACM International Symposium on Physical Design 2011. Currently, he is an Associate Editor of IEEE Transactions on Computer Aided Design.