Delay-Optimal Simultaneous Technology Mapping and Placement with Applications to Timing Optimization

Yifang Liu
Department of ECE
Texas A&M University
College Station, TX
Email: yliu@ece.tamu.edu

Rupesh S. Shelar
Technology & Manufacturing Group
Intel Corporation
Hillsboro, OR
Email: rupesh.s.shelar@intel.com

Jiang Hu
Department of ECE
Texas A&M University
College Station, TX
Email: jianghu@ece.tamu.edu

Abstract— **Technology mapping and placement have significant impact on the delays in standard cell based very large scale integrated (VLSI) circuits. Traditionally, these steps are applied separately to optimize 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 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 (DAGs). Experimental results on benchmark circuits in a 70 nm technology show that our algorithms improve timing significantly with remarkably less run-times compared to a competitive approach of iterative conventional timing driven mapping and multi-level placement.**

I. Introduction

A. Motivation

In today's technologies, interconnects contribute to significant portion of the overall delay in VLSI circuits. The trend is likely to continue, or worsen, as the technology scaling continues, since the wire delays do not scale as well as cell delays. 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 a 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 CAD researchers over the last several years.

B. Previous Work

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 dynamic programming technique and runs in polynomial time in the size of the tree, ensuring optimality for the 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.*, where an initial placement of a subject graph and the assumption about the placement of a match was employed to evaluate wire- and cell-delays to derive a delay-optimized mapped netlist [3]. 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 and that 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 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-dimensional (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. Recently, Wang *et al.* proposed an iterative mapping scheme [8] employing multipliers, similar to those in Lagrangian relaxation technique, to optimize the area/power under fixed cell-delay model; the wire-delays based on the placement, however, are not considered.

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 [9] 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.* presented $O(n \log n)$ algorithm for the optimal placement minimizing the sum of weighted edge-lengths for a tree with n leaves [10]; recent work includes a linear time algorithm to minimize the sum of half-perimeter wirelengths for all nets in a tree [11]. 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 [12] employed in [4] to perform simultaneous mapping and linear placement. However, the problem of delay-optimal placement for trees seem to have received relatively less attention in the published literature, despite the potential usefulness of the solution.

C. Our Contributions

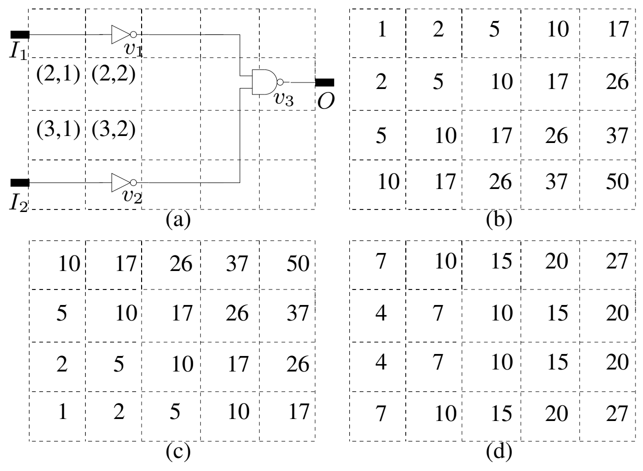
Since the technology mapping and placement have great impact on the overall delays in the circuit, exploring these two spaces simultaneously can result in circuits with better delays than the conventional approach of searching those sequentially, which results in the search in a relatively small solution space. A fundamental contribution of this work 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 directed acyclic graphs (DAGs), we propose an iterative algorithm, based on Lagrangian relaxation (LR) technique, which employs the simultaneous technology mapping and placement in the inner loop. The comparison of results on 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 [13] and timing driven placement mPL [14] shows more than 60% slack improvement with 7 times speed-up in runtime, on an average, implying that the proposed algorithms are practical and can be employed to optimize timing during physical synthesis.

The rest of the paper is organized as follows. Section II describes the formal notation employed in this article. Section III presents an algorithm for delay-optimal placement of trees, whereas Section IV extends the algorithm to perform delay-optimal simultaneous technology mapping and placement. Section V briefly describes the algorithm based on LR for simultaneous mapping and placement for DAGs. Section VI discusses the results due to the algorithms and compares them with those due to the competitive approach, and Section VII 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 the technology mapping to create a mapped netlist. Subsequently, the placement is carried out on the mapped netlist to assign each cell a location in a given area. The graph theoretic 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 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, $d^{wire}(e(v_i, v_j))$. 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.

III. Delay-optimal Placement for Trees



(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) 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) The 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) The placement-delay table for v_2. (d) The placement-delay table for v_3, obtained by using the optimal locations for fanins v_1 and v_2.

We introduce a polynomial time algorithm for the delay-optimal placement of a tree in this section and describe its extension to simultaneous mapping and placement in the next. 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, also referred to as the leaves, have fixed locations and so does the root of the tree. We want to place the tree in a layout area, which is divided into bins or tiles, similar to those in conventional global placement [14]. Specifically, we want to assign each node $v \in V_T$ a bin (x, y). There are several possible placements leading to different delays, since the wire- and cell-delays are functions of the locations of the driver and the receiver. Among these placements, we want to find the 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, minimize

$$\max_{\pi \in input-root\ paths} d(\pi),$$

subject to

$$(x_i, y_i) \in Z_i, \quad \forall v_i \in V_T.$$

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 dynamic programming.

A. Tree Placement Algorithm

The tree placement algorithm has two phases: first phase of bottom-up solution generation and the second phase of actually choosing a placement from those solutions, given the fixed location of the root. The first phase 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, 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 wire-length. Consider a location $(3, 1)$ 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, $d^{wire}(e(I_1, v_1)) = (|1-3|+|1-1|)^2 = 4$, 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 $(3, 1)$, is 5. Similarly, when v_1 is fixed at $(2, 1)$, the optimal delay for the subtree rooted at v_1 is 2, since the wire delay $d^{wire}(e(I_1, v_1)) = (|1 - 2| + |1 - 1|)^2 = 1$ 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 $(3, 1)$ and $(2, 1)$ 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 depicted 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 $(4, 1)$, the location chosen for v_2 is also $(4, 1)$, 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 $(4, 1)$, + 0^2, wire-delay, + 1, cell-delay for v_3). Similarly, two locations $(3, 1)$ and $(2, 1)$ 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 $(4, 1)$. The overall delay for the subtree rooted at v_3, when it is placed in $(4, 1)$ is $max(2, 7) = 7$; this is reflected in the bin $(4, 1)$ in placement-delay table for v_3, shown in Figure 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 Eq. (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. ■

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 $(2, 5)$, the optimal location of v_3, which results in the minimum delay is $(2, 3)$, yielding the delay of 14 (10, $a_{v_3}(2, 3)$, *i.e.,* delay of the subtree rooted at v_3, + 2^2, wire-delay from $(2, 3)$ to $(2, 5)$). The optimal locations of v_1 and v_2, which resulted in the delay of 10 for the subtree rooted at v_3 are $(1, 3)$ and

$(4,3)$, respectively; these can be found out in a constant time by storing additional information along with the placement-delay table. Thus, the optimal placement for the tree is as follows: $v_1(x_{opt}, y_{opt}) = (1,3)$; $v_2(x_{opt}, y_{opt}) = (4,3)$; and $v_3(x_{opt}, y_{opt}) = (2,3)$.

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
```

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 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- and wire-delays, respectively, the algorithm ensures delay-optimality with other delay models also. For instance, asymptotic waveform evaluation (AWE) 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

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
```

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 2-input NANDs and inverters into an implementation based on the set of cells in a library and is carried out in two steps: matching and covering. For conventional delay oriented technology mapping employing load-dependent delay model [13], 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 wire-delays, the approaches in the literature such as [3], [5], [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 novelty of the algorithm lies in its polynomial time and space complexities, despite storing the delay-optimal placements for all the mapping solutions. 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 $O(nm^2f_{max}P_{max}^2)$.

V. Handling DAGs by Lagrangian Relaxation

A circuit represented by a DAG may contain multi-fanout nodes. As a result, these nodes may have conflicting choices for delay-optimal mapping as well as placement from the perspective of different fanouts. This limits the application of dynamic programming to delay-optimal mapping and placement on DAGs. To overcome the difficulty, we propose a heuristic based on Lagrangian relaxation (LR): it applies the simultaneous tree mapping and placement to minimize delays weighted by Lagrangian multipliers iteratively; the multipliers are updated employing sub-gradients [15], following the static timing analysis on the mapping and placement solution in the current iteration; the algorithm stops, if there is no significant improvement in the slack. Our approach is similar to that proposed by Wang *et al.* in [8], but differs in the objective and in updating the multipliers. Their work applies multiplier-based heuristic to optimize area/power with fixed cell-delay model without considering the wire-delays based on placements, whereas the objective of our approach is the slack maximization accounting for the wire-delays based on placement. 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. Experimental Results

The algorithms described in this paper are implemented in a C++ program on Windows platform with 3.0 GHz Pentium IV processor. To evaluate the efficacy of the algorithms, the experiments are run on the set of ISCAS'85 combinational benchmark circuits with a standard cell library characterized employing 70 nm technology parameters [16]. 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 following four iterative approaches, whose goal is to maximize the worst case slack, are compared:

- **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 [13] and considers the wire-delays based on placements, whereas the timing driven placement is implemented by incorporating timing aware net weighting technique [17] with mPL6 [14].
- **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.
- **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.
- **LR with simultaneous tree 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 $10ps$ slack improvement in consecutive iterations. The results due to all the approaches are shown in Table I. As compared to the conventional approach, Lagrangian relaxation based algorithm improves the average slacks and maximum delays by 69% and 11%, respectively, with 7 times speed-up in the runtime. Similarly, tree based simultaneous mapping and placement leads to 62% and 7% improvements in the slacks and delays, respectively, with approximately 2 orders of magnitude

	Conventional			Conventional mapping with tree placement					Simultaneous tree mapping & placement					LR with simultaneous mapping & 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
Ave.	1748	77	700	1653	172	7.7			1610	215	9.5			1573	251	95		
Norm.	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

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 LAGRANGIAN RELAXATION (LR) WITH SIMULTANEOUS MAPPING AND PLACEMENT. THE MAXIMUM PATH DELAY AND THE MINIMUM SLACK ARE IN ps; CPU TIME IS IN SECONDS; TOTAL WIRE LENGTH, CELL AREA ARE NORMALIZED WITH RESPECT TO THE CORRESPONDING QUANTITIES DUE TO THE CONVENTIONAL APPROACH.

small run-times. 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 wirelength 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 wirelength 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.

VII. CONCLUSION

In this paper, we proposed 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 using Lagrangian relaxation technique. Compared to the conventional iterative mapping and timing driven placement approach, our methods improve the slacks by more than 60%, with 7 times or greater speed-up, and have negligible impact on total wirelength and cell area.

ACKNOWLEDGMENTS

The first author would like to thank his colleague, Charu Nagpal, for the help on experimental setup with SIS. Authors are grateful to anonymous reviewers for the valuable comments.

REFERENCES

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

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

[3] M. Pedram and N. Bhat, "Layout driven technology mapping," in *Proc. DAC*, June 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. ICCAD*, Nov. 1997, pp. 671–675.

[5] J. Y. Lin, A. Jagannathan, and J. Cong, "Placement-driven technology mapping for LUT-Based FPGA's," in *Proc. ISFPGA*, Feb. 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. ICCAD*, November 2001, pp. 224–231.

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

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

[9] J. Cong, J. Shinnerl, M. Xie, T. Kong, and X. Yuan, "Large-scale circuit placement," *ACM Trans. on Design Automation of Electronic Systems*, vol. 10, no. 2, pp. 1–42, Apr. 2005.

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

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

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

[13] E. M. Sentovich, "SIS: A system for sequential circuit synthesis," Memorandum No. UCB/ERL M92/41, May 1992.

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

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

[16] "Berkeley predictive technology model," http://www-device.eecs.berkeley.edu/~ptm/download.html.

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