TILA: Timing-Driven Incremental Layer Assignment

Bei Yu[1,2], Derong Liu[1], Salim Chowdhury[3], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]CSE Department, Chinese University of Hong Kong, NT, Hong Kong
[3]Oracle Corp., Austin, TX, USA
{bei, deronliu, dpan}@cerc.utexas.edu, salim.chowdhury@oracle.com

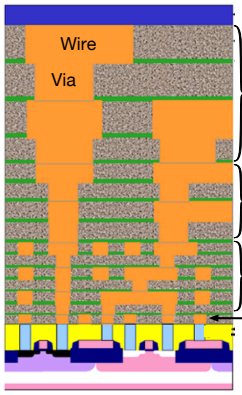
***Abstract*—As VLSI technology scales to deep submicron and beyond, interconnect delay greatly limits the circuit performance. The traditional 2D global routing and subsequent net by net assignment of available empty tracks on various layers lacks a global view for timing optimization. To overcome the limitation, this paper presents a timing driven incremental layer assignment tool, TILA, to reassign layers among routing segments of critical nets and non-critical nets. Lagrangian relaxation techniques are proposed to iteratively provide consistent layer/via assignments. Modeling via min-cost flow for layer shuffling avoids using integer programming and yet guarantees integer solutions via uni-modular property of the inherent model. In addition, multiprocessing of $K \times K$ partitions of the whole chip provides run time speed up. Certain parameters introduced in the models provide trade-off between timing optimization and via count. Experimental results in both ISPD'08 and industry benchmark suites demonstrate the effectiveness of the proposed incremental algorithms.**

I. Introduction

As VLSI technology scales to deep submicron and beyond, interconnect delay plays a determining role in timing [1]. Therefore, interconnect synthesis, including buffer insertion / sizing and timing-driven routing, becomes a critical problem for achieving timing closure [2]. Global routing is an integral part of a timing convergence flow to determine the topologies and layers of nets, which greatly affect the circuit performance [3]–[9]. In emerging technology nodes, back-end-of-line (BEOL) metal stack is with heterogeneous routing resources, i.e., dense metal at the lower layers and wider pitches at the upper layers. Fig. 1 gives one example of cross section of IC interconnection stack in advanced technology nodes [10], where wires and vias on top metal layers are much wider and much less resistive than those on lower metals. Besides, the normalized pitch lengths of different metal layers from [11] are also listed. Therefore, advanced routing algorithms should not only be able to achieve routability, but also intelligently assign layers to overcome interconnect timing issues.

Layer assignment is an important step in global routing to assign each net segment to the corresponding metal layer. It is commonly generated during or after the wire synthesis to meet tight frequency targets, and to reduce interconnect delay on timing critical paths [12]. In layer assignment, wires on thick metals are much wider and thus, less resistive than those on thin metals. If timing critical nets are assigned to lower layers, it will make timing worse due to narrower wire width/spacing. Although top metal layers are less resistive than those in lower (thin) metals, it is impossible to assign all wires to top layers. That is, layer assignment should satisfy the capacity constraints on thick metals. If excessive number of wires are assigned to a particular layer, it will aggravate congestion and crosstalk. In addition, the delay due to vias cannot be ignored in emerging technology nodes [1].

Recently, layer assignment has been considered in two design stages, i.e., buffered tree planning, and 3D global routing. Some studies consider layer assignment during buffer routing trees design



Layer	Capacitance
M10	1.00
M9	1.00
M8	0.63
M7	0.63
M6	0.63
M5	0.63
M4	0.63
M3	0.33
M2	0.35
M1	0.33

Fig. 1. The cross section of IC interconnection stack in advanced technology nodes [10], where wires and vias on top metal layers are much wider and much less resistive than those on lower metals. The normalized pitch lengths of different metal layers are listed in the table (source: [11]).

[12]–[14]. Li et al. [12] proposed a set of heuristics for simultaneous buffer insertion and layer assignment. Hu et al. [13], [14] proved that, even if buffer positions are determined, the layer assignment with timing constraints is NP-complete. During 3D global routing, layer assignment is a popular technique for via minimization. [3] proposed an integer linear programming (ILP) based method to solve the layer assignment problem. Since via minimization is the major objective of this work, all wires tend to be assigned onto the lower layers. [15], [16] applied dynamic programming to solve optimal layer assignment for single net. To overcome the impact of net order, different heuristics or negotiation techniques are proposed in [17], [18]. Ao et al. [18] considered the delay in layer assignment, but since via capacity was not considered, more segments can be illegally pushed onto higher routing layers. A min-cost flow based refinement was developed in [19] to further reduce the via number.

Existing layer assignment studies suffer from one or more of the following limitations: (1) Most works only target at via number minimization, but no timing issues are considered. Since timing requirements within a single net are usually different for different sinks, assigning all segments of a set nets on higher metal layers is not the best use of critical metal layer resources. That is, intelligent layer assignment should not blindly assign all segments of a net to a set (a pair, for example) of higher metal layers. It should be aware of capacitive loading of individual segments within a net to achieve better timing with the limited available higher metal layer resources. (2) In emerging technology nodes, the via delays contribute a significant part of total interconnect delay. But the delay impact derived from vias is usually ignored in previous layer assignment works. (3) The net-by-net strategy may lead to local optimality, i.e., for some nets the timings are over-optimized, while some other nets

978-1-4673-8388-2/15/$31.00 ©2015 IEEE

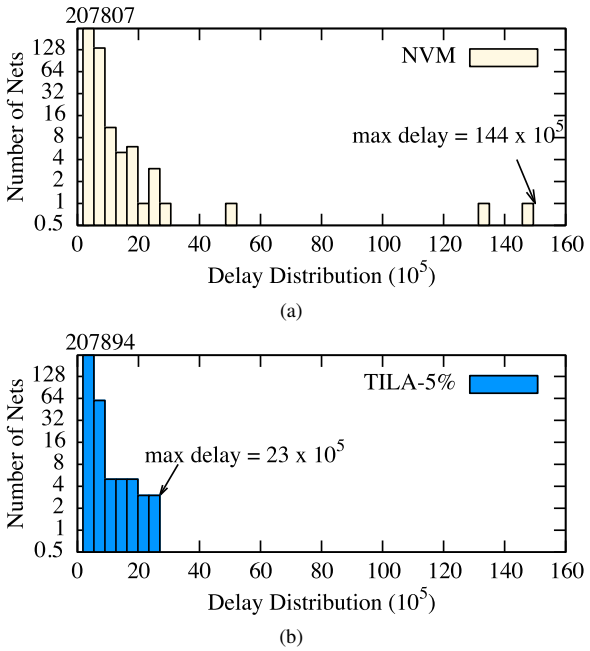


Fig. 2. Net delay distribution for benchmark adeptect2. (a) Result by layer assignment solver NVM [17]; (b) Result by our timing-driven incremental layer assignment solver TILA, where 5% most critical nets are reassigned layers.

may have no enough resources in high layers. To close on timing for critical nets that need to go long distances, layer assignment needs to be controlled by multi-net global optimization. For example, Fig. 2 compares the delay distributions of benchmark 'adeptect2' by conventional layer assignment solver [17] and our incremental timing-driven solution. We can see that, since conventional layer assignment only targets at via minimization, the maximum delay can be very huge. Since our timing-driven planner is with global view, the maximum delay can be much better, i.e., the normalized maximum delay can be reduced from 144×10^5 to 23×10^5.

For very large high-performance circuits, either long computation times have to be accepted or routing quality must be compromised. Therefore, an incremental layer assignment to iteratively improve routing quality is a must. In this paper, we propose an incremental layer assignment framework targeting at timing optimization. Incremental optimizations or designs are very important in physical design and CAD field to achieve good timing closure [20]. Fast incremental improvements are developed in different timing optimization stages, such as incremental clock scheduling [21], [22], incremental buffer insertion [23], and incremental clock tree synthesis [24]. To further improve timing, incremental placement is also a very typical solution [25], [26]. Besides, there are several incremental routing studies (e.g. [27]) to introduce cheap and incremental topological reconstruction.

In this paper, we propose a comprehensive study to the timing-driven incremental layer assignment problem. To the best of our knowledge, this work is the first incremental layer assignment work integrating via delay and solving all the nets simultaneously. A multilayer global router can either route all nets directly on multilayer solution space [4], [5] or one layer routing followed by post-stage layer assignment [6]–[9]. Note that as an incremental layer assignment solution, our tool can smoothly work with either type of global router. Our contributions are highlighted as follows.

- A mathematical formulation to give the layer assignment solutions with optimal total wire delays and via delays.

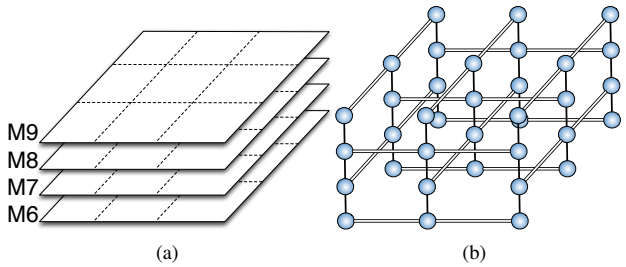


Fig. 3. Layer design and grid models. (a) A design with four routing layers {M6, M7, M8, M9}; (b) Grid model with preferred routing directions.

- A Lagrangian relaxation based optimization to iteratively improve the layer assignment solution.
- Lagrangian relaxation subproblem (LRS) is solved via min-cost flow model.
- Multiprocessing of $K \times K$ partitions of the whole chip provides run time speed up.
- Both ISPD'08 and industry benchmarks demonstrate the effectivenesses of our framework.

The remainder of this paper is organized as follows. Section II provides some preliminaries and the problem formulation. Section III gives mathematical formulation, and also proposes sequence of multi-threaded min-cost flow algorithm to achieve further speed-up. Section IV reports experimental results, followed by conclusion in Section V.

II. Preliminaries and Problem Formulation

In this section we introduce the graph model and the timing model applied in this paper. Then the problem formulation of timing-driven incremental layer assignment is provided.

A. Graph Model

Similar to the 3D global routing problem, layer assignment problem can be modeled on a 3D grid graph, where each vertex represents a rectangular region of the chip, so called a global routing cell (**G-Cell**), while each edge represents the boundary between two vertices. In the presence of multiple layers, the edges in the z-direction represent vias connecting different layers. Fig. 3(a) shows a grid graph for routing a circuit in multi-metal layer manufacturing process. Each metal layer is dedicated to either horizontal or vertical wires. The corresponding 3D grid graph is shown in Fig. 3(b).

To model the capacity, for each edge, we denote its maximum routing capacity as c_e. Besides, the via capacity of each vertex, denoted by c_v, is computed as in [28]. In brief, the via capacity refers to the available space for vias passing through the cell, and is determined by the available routing capacity of those two edges connected with the vertex.

B. Timing Model

We are given a global routing of nets, where each net is a tree topology with one source and multiple sinks. Based on the topology, for each net we have a set of segments S. Here we give an example of net model in Fig. 4, where each net contains two segments. To evaluate the timing of each net, we adopt *Elmore* delay model, which is widely used during interconnect synthesis in physical design, as the delay model. The delay of a segment s_i on a layer l, denoted by $d_e(i,l)$, is computed as follows:

$$d_e(i,l) = R_e(l) \cdot (C(l)/2 + C_{down}(s_i)) \quad (1)$$

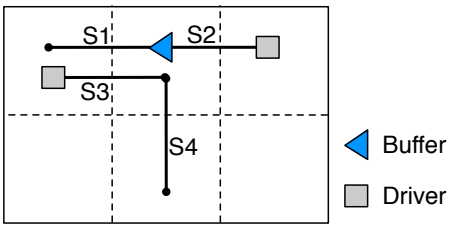


Fig. 4. Example of net model.

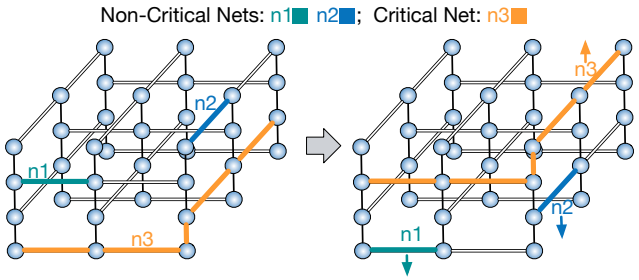


Fig. 5. An example of timing driven layer assignment. In initial layer assignment net n_3 is timing critical net. Through resource releasing from nets n_1 and n_2, the total timing gets improvement.

where $R_e(l), C(l)$ refer to the edge resistance on layer l, and edge capacitance on layer l, respectively. $C_{down}(s_i)$ refers to the down stream capacitance of s_i. Note that the down stream capacitance is determined by the assigned layers of down stream segments. To calculate the down stream capacitance for each s_i, we traverse the net tree from sinks to source in a bottom-up manner. Therefore, the down stream capacitance of the source segment, i.e. the segment connected with the driver pin, should be calculated after all the other segments have obtained their down stream capacitances.

For a via v_m connecting segments between layers l and $l+1$, its delay can be calculated as follows.

$$d_v(v_m, l) = R_v(l) \cdot \min\{C_{down}(s_i), s_i \in N(v_m)\} \tag{2}$$

Here $R_v(l)$ is the resistance of via between layers l and $l+1$, and $C_{down}(s_i)$ is the down stream capacitance of segment s_i. $N(v_m)$ is the set of neighboring segments of via v_m. For a via connected to source or sink, its delay is set to zero due to the zero value of minimum down stream capacitance.

In addition, buffer positions can be considered in our delay model. That is, for one segment s_i, if there is one buffer at its end point, its down stream capacitance $C_{down}(s_i) = 0$. As shown in Fig. 4, $C_{down}(s_2)$ is set to zero due to the buffer. Therefore, our framework can handle timing optimization for both pre-buffered and post-buffered designs.

C. Problem Formulation

Based on the grid model and timing model discussed in the preceding section, we define the timing-driven incremental layer assignment (TILA) problem as follows:

Problem 1 (TILA)**.** *Given a global routing grid, set of net segments and layer capacity information, timing-driven incremental layer assignment assigns each segment passing through an edge to a layer, so that layer assignment costs (weighted sum of segment delays) can be minimized, while the capacity constraints of each edge on each layer are satisfied.*

One instance of TILA problem with three nets is demonstrated in Fig. 5, where nets n_1 and n_2 are non-critical nets, while net n_3 is timing critical net. In the initial layer assignment as shown in Fig. 5(a), net n_3 is assigned lower layers. Since the routing resources are utilized by nets n_1 and n_2, n_3 cannot be shuffled into higher layers to improve timing. Through a global layer reassignment, a better timing assignment solution is shown in Fig. 5(b), where both n_1 and n_2 release high layer resources to n_3.

Naclerio et al. proved that even if no timing is considered, the decision version of layer assignment for via minimization is NP-complete [29]. Thus the decision version of TILA problem is NP-complete as well.

III. TILA Algorithms

In this section, we introduce our framework to solve the TILA problem. First a mathematical formulation to the TILA problem will be given. Then a Lagrangian relaxation based optimization methodology is proposed to solve this problem. For convenience, some notations used in this section are listed in Table I.

TABLE I
NOTATIONS USED IN THIS PAPER.

L	number of layers
S	set of all segments
E	set of all edges
E_x	set of all pairs of crossing segments
V	set of all vias
$P(s_i)$	set of pins in segment s_i
$N(v_m)$	set of neighboring segments of via v_m
$S_e(i)$	set of segments assigned to the same edge as s_i
a_{ij}	binary variable; if i-th segment is assigned to layer j then $a_{ij} = 1$, otherwise $a_{ij} = 0$
$d_e(i,j)$	timing cost if s_i is assigned to layer j
$d_v(i,p,k)$	timing cost of via v from layer k to $k+1$, where $v \in P(s_i) \cap P(s_p)$
$c_e(i)$	routing capacity of edge e where s_i is assigned
$c_v(k)$	via capacity of via v on layer k

A. Mathematical Formulation

The mathematical formulation of TILA problem is shown in formula (3). In the objective function, the first term is to calculate the cost from segments, while the 2nd term is to calculate the cost from vias. Here $d_e(i,j)$ is calculated through Eqn. (1), and $d_v(i,p,k)$ is derived from Eqn. (2).

$$\min \sum_{i=1}^{S}\sum_{j=1}^{L} d_e(i,j)\cdot a_{ij} + \sum_{(i,p)\in E_x}\sum_{j=1}^{L}\sum_{q=1}^{L}\sum_{k=j}^{q-1} d_v(i,p,k)\cdot a_{ij}\cdot a_{pq} \tag{3}$$

$$\text{s.t.} \quad \sum_{j} a_{ij} = 1, \qquad \forall i \in [1,S] \tag{3a}$$

$$\sum_{s_i\in S_e(i)}\sum_{j} a_{ij} \le c_e(i), \qquad \forall e \in E \tag{3b}$$

$$\sum_{(i,p)\in E_x} a_{ij}\cdot a_{pq} \le c_v(k), \quad \forall k, j<k<p \tag{3c}$$

$$a_{ij} \text{ is binary} \tag{3d}$$

Constraint $(3a)$ is to ensure that each segment of nets would be assigned to one and only one layer. Each edge $e \in E$ is associated with one capacity $c_e(i)$, and constraint $(3b)$ is for the edge capacity of each layer. Constraint $(3c)$ is for the via number capacity in each layer.

First, we show that if each $C_{down}(s_i)$ is constant, the TILA can be formulated as an integer linear programming (ILP), then a mature ILP solver is possible to be applied. Here $C_{down}(s_i)$ is down stream capacitance of segment s_i. We can use a boolean variables $\gamma_{ij,pq}$ to replace each non-linear term $a_{ij} \cdot a_{pq}$. Then formula (3) can be transferred into ILP through introducing the following artificial constraints:

$$\begin{cases} a_{ii} + a_{pq} \leq \gamma_{ij,pq} + 1 \\ a_{ik} \geq \gamma_{ij,pq}, \quad a_{jk} \geq \gamma_{ij,pq} \end{cases} \tag{4}$$

Due to the computational complexity, ILP formulation suffers from serious runtime overhead, especially for those practical routing test cases. A popular speedup technique is to relax the ILP into linear programming (LP) by removing the constraint $(3d)$. It is obvious that the LP solution provides a lower bound to the original ILP formulation. We observe that the LP solution would be like this: each a_{ij} is assigned to 0.5 and each $\gamma_{ij,pq}$ is 0. By this way, all the constraints are satisfied, and the objective function is minimized. However, all these 0.5 values to a_{ij} provide no useful information in guiding the layer assignment, as we prefer each a_{ij} closes to either 0 or 1. In other words, the LP relaxation is hard to provide reasonable good solution. Instead of expensive ILP formulation or its LP relaxation, our framework proposes a Lagrangian relaxation based algorithm to solve the original formula (3).

B. Lagrangian Relaxation based Optimization

Lagrangian relaxation [30] is a solution technique for solving optimization problems with difficult constraints, where some or all hard constraints are moved into objective function. In the updated objective function, each new term is multiplied with a constant known as Lagrange multipliers (**LM**). Our idea is to relax the via capacity constraint $(3c)$ and incorporate it into the objective function. We specify each $a_{ij,pq}$ a non-negative LM $\lambda_{ij,pq}$, and move the constraint into objective function. The modified formula is called Lagrangian relaxation subproblem (LRS), as shown in formula (5).

$$\begin{aligned} \min \ & \sum_{i=1}^{S}\sum_{j=1}^{L} d_e(i,j) \cdot a_{ij} + \sum_{(i,p)\in E_x}\sum_{j=1}^{L}\sum_{q=1}^{L}\sum_{k=j}^{q-1} d_v(i,p,k) \cdot a_{ij} \cdot a_{pq} \\ & + \sum_{(i,p)\in E_x} \lambda_{ij,pq}(a_{ij} \cdot a_{pq} - c_v(k)) \end{aligned} \tag{5}$$

$$\text{s.t.} \quad (3a) - (3b), (3d)$$

It is known that for any fixed set of LM $\lambda_{ij,pq}$, the optimal result to the LRS problem is smaller or equal to the optimal solution of the original formula (3) [30]. That is, the original formulation is the primal problem and the Lagrange multiplier optimization is the dual problem. Therefore, the Lagrangian dual problem (LDP) is to maximize the minimum value obtained for the LRS problem by updating LMs accordingly.

Algorithm 1 gives a high level description of our Lagrangian relaxation based framework to the TILA problem. The inputs are an initial layer assignment solution and a critical net ratio value α. Based on the α value we select some critical nets and non-critical nets (line 1). All the segments belonging to these nets are reassigned layers by our incremental framework. Please refer to Section III-D for more details of our critical and non-critical net selection. Based on the initial layer assignment solution, we initialize all the $C_{down}(s_i)$ for each selected segment s_i (line 2). We initialize the LMs in line 3. In our implementation, the initial values of all LMs are set to 2000. Our framework iteratively solves a set of Lagrangian relaxation subproblem (LRS), with fixed LM values (lines 4–8). In solving LRS, we minimize the objective function in Eqn. (5) based on the current set of LMs. The details of solving LRS are discussed in Section III-C. After solving each LRS, we re-calculate the down stream capacitances of all the segments $C_{down}(s_i)$ based on Eqn. (1) (line 6). We use a subgradient-based algorithm [31] to update the LMs to maximize LDP (line 7). In our implementation, the iteration in line 4 will end if one of the following two conditions is satisfied: either the iteration number is larger than 20; or both the wire delay improvement and the via delay improvement are less than a pre-specified fraction.

Algorithm 1 TILA

Require: Initial layer assignment solution;
Require: Critical net ratio α;
1: Select all segments based on α [Sec. III-D];
2: Initialize $C_{down}(s_i)$ for each segment s_i;
3: Initialize LMs;
4: **while** no converge **do**
5: Solve LRS [Sec. III-C];
6: Update $C_{down}(s_i)$ for all s_i;
7: Update LMs;
8: **end while**

C. Solving Lagrangian Subproblem (LRS)

Through removing the constant items in objective function of (5), we re-write LRS into formula (6).

$$\min \sum_{i=1}^{S}\sum_{j=1}^{L} c(i,j) \cdot a_{i,j} + \sum_{(i,p)\in E_x}\sum_{j=1}^{L}\sum_{q=1}^{L} c(i,j,p,q) \cdot a_{ij} \cdot a_{pq} \tag{6}$$

$$\text{s.t.} \quad (3a) - (3b), (3d)$$

where

$$\begin{cases} c(i,j) = d_e(i,j) \\ c(i,j,p,q) = \sum_{k=j}^{q-1} d_v(i,p,k) + \lambda_{ij,pq} \end{cases}$$

Theorem 1. *For a set of fixed $\lambda_{ij,pq}$,* LRS *is NP-hard.*

Due to space limit, the detailed proof is omitted. Because of nonlinear term $a_{ij} \cdot a_{pq}$, the proof can be through a reduction from quadratic assignment problem [32]. In addition, unless $\mathsf{P} = \mathsf{NP}$, the quadratic assignment problem is not approximable in polynomial time within some finite approximation ratio [33].

We propose the following heuristic to linearize term $a_{ij} \cdot a_{pq}$:

$$c(i,j,p,q) \cdot a_{ij} \cdot a_{pq} \approx c(i,j,p,q) \cdot (a'_{pq} \cdot a_{ij} + a'_{ij} \cdot a_{pq}) \tag{7}$$

where a'_{pq} is the value of a_{pq} in previous iteration, and a'_{ij} is the value of a_{ij} in previous iteration.

Through the linearization technique in Eqn. (7), the objective function in formula (6) is a weighted sum of all the a_{ij}. We will show that the linearized LRS can be solved through a min-cost network flow model. The basic idea is that the weighted sum of all the a_{ij} can be viewed as several assignments from segments to layers, while the weight of each a_{ij} is the cost to assign segment i to layer j. Constraints $(3a)$ and $(3b)$ can be integrated into the flow model through specified edge capacity. Constraint $(3d)$ is satisfied due to the inherent uni-modular property [31].

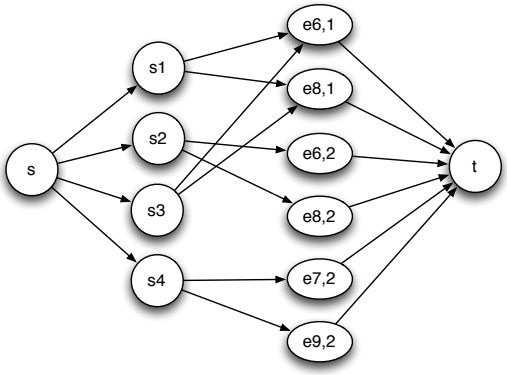


Fig. 6. An example of min-cost flow model.

An example of such min-cost flow model is illustrated in Fig. 6. Given four different segments s_1, s_2, s_3, s_4 and several edges, we build up a directed graph $G = (V, E)$ to represent the layer assignment relationships. The vertex set V includes four parts: start vertex s, net vertices V_N, layer vertices V_L, and end vertex t. The edge set E is composed of three sets of edges: $\{s \rightarrow V_N\}$, $\{V_N \rightarrow V_L\}$, and $\{V_L \rightarrow t\}$. We define all the edge **costs** as follows: the cost of one edge from V_N to V_L is the penalty assigning the segment to corresponding layer; the costs of all other edges are set to 0. We define all the edge **capacities** as follows: the capacity of one edge from V_L to node t is the capacity of the corresponding edge; while the capacities of all other edges are set to 1.

D. Critical & Non-Critical Net Selection

Given an input ratio value α, our framework would automatically identify $\alpha\%$ of the total nets as critical nets, while other $\alpha\%$ of the total nets as non-critical nets. Both the selected critical nets and the selected non-critical nets would be reassigned layers. The motivation of critical net selection is to reassign their layers to improve timing, while the motivation of non-critical net selection is to release some high layer resources to the critical nets. By this way, our incremental layer assignment flow is able to overcome the limitation of any net order in original layer assignment. In our implementation, the default value of α is set to 1, which means 1% of nets would be identified as critical nets, while the other 1% of nets are selected as non-critical nets.

To identify all the critical nets can be trivial: first all the net delays in original layer assignment are calculated based on our delay model as in Section II, then the $\alpha\%$ of worst delays are selected. Yet, non-critical net selection is not so straightforward, as randomly selecting $\alpha\%$ of best timing nets may not be beneficial to improve critical net timing. Therefore, we prefer to select those nets sharing more routing resources with the critical nets. In our implementation, we check the $2 \cdot \alpha$ best timing nets to associate each net with a score representing its ability in sharing resources to critical nets. Then we select half of them with best scores as non-critical nets.

E. Parallel Scheme

Our framework supports parallel scheme that the global routing graph is divided into $K \times K$ parts. An example of such division is illustrated in Fig. 7, where $K = 4$. The timing-driven incremental layer assignment is solved in each partition separately. The reason of such division is twofold. Firstly, our Lagrangian relaxation based optimization is to solve a set of min-cost flow models, as discussed in Section III-B and Section III-C. The runtime complexity to solve a single flow model is $O(|V| \cdot |E|)$, where $|V|$ and $|E|$ are the vertex number and the edge number of the graph. Dividing the whole problem into a set of sub-problems can achieve some speed-up. In addition, multithreading is applied to provide further speed-up. For instance, in Fig. 7 four threads are used to solve different regions simultaneously. Secondly, inspired by the Gauss-Seidel method [34], when one thread is solving flow model in one partition, the most recently updated results by peer threads are taken into account, even if the updating occurs in the current iteration.

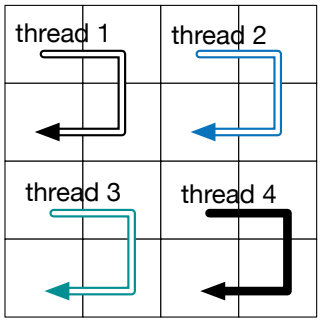


Fig. 7. Our parallel scheme to support multithreading computing on $K \times K$ partitions. (Here $K = 4$).

IV. Experimental Results

We implemented the proposed timing-driven incremental layer assignment framework in C++, and tested it on a Linux machine with eight 3.3GHz CPUs. We selected open source graph library LEMON [35] as our min-cost network flow solver, and utilized OpenMP [36] to provide parallel computing. In our implementation, the default K value is set to 6, and the default thread number is set to 6.

A. Evaluation on ISPD'08 Benchmarks

TABLE II
Normalized Capacitance and Resistance.

Wire [11]			Via	
Layer	C	R	Layer	R
M1	1.14	23.26	$v_{1,2}$	25.9
M2	1.05	19.30	$v_{2,3}$	16.7
M3	1.05	23.26	$v_{3,4}$	16.7
M4	0.95	5.58	$v_{4,5}$	16.7
M5	1.05	3.26	$v_{5,6}$	5.9
M6	1.05	3.26	$v_{6,7}$	5.9
M7	1.05	3.26	$v_{7,8}$	5.9
M8	1.00	3.26	$v_{8,9}$	1.0
M9	1.05	1.00	$v_{9,10}$	1.0
M10	1.00	1.00	-	-

In the first experiment, we evaluate our timing-driven layer assignment framework on ISPD'08 benchmarks [37]. The NCTU-GR 2.0 [9] is utilized to generate the initial global routing solutions. The initial layer assignment results are from NVM [17], which is targeting at via number and overflow minimization. Our framework is tested the effectiveness to incrementally optimize the timing. To calculate the wire delay in Eqn. (1) and via delay in Eqn. (2), all the metal wire resistances, metal wire capacitances, and via resistances are listed in Table II. Column **C** lists the capacitance. Columns **R**

TABLE III
PERFORMANCE COMPARISONS ON ISPD'08 BENCHMARKS

bench	NVM [17]				TILA-1%					TILA-5%				
	OV#	D_{avg} (10^3)	D_{max} (10^3)	via# (10^5)	OV#	D_{avg} (10^3)	D_{max} (10^3)	via# (10^5)	CPU (s)	OV#	D_{avg} (10^3)	D_{max} (10^3)	via# (10^5)	CPU (s)
adaptec1	48521	7.26	8776.6	19.03	50727	6.90	7184.7	19.49	72.1	57452	6.48	7141.7	20.75	94.5
adaptec2	45667	4.35	14424.9	19.01	41039	3.62	2369.3	19.54	63.9	36113	3.22	2375.8	20.97	91.8
adaptec3	90574	9.70	24998.9	36.29	88626	8.68	7839.1	37.13	288.4	91938	7.97	7841.2	39.87	562.1
adaptec4	76183	6.96	38646.7	31.56	66815	5.90	9703.7	32.80	235.1	56751	5.29	9701.5	35.55	459.8
adaptec5	95971	10.95	9958.0	54.30	97288	10.05	8709.7	56.01	287.2	100175	9.23	8673.3	59.73	357.0
bigblue1	44595	13.50	3675.4	21.25	48377	13.05	3474.5	21.98	87.4	57204	12.27	3411.0	23.41	113.1
bigblue2	105420	3.02	58259.1	42.70	99023	2.64	18279.4	43.82	108.6	97619	2.47	18285.7	45.94	160.6
bigblue3	84329	4.98	3122.2	51.29	76623	4.17	2712.7	53.23	274.0	69064	3.55	2726.3	59.68	541.4
bigblue4	115155	8.22	53401.4	107.65	109037	7.10	35424.8	111.18	401.4	115102	6.11	35414.7	121.32	578.7
newblue1	58578	1.21	670.7	22.03	56607	1.01	565.8	22.52	65.6	52179	0.94	565.6	23.97	82.2
newblue2	72403	4.31	12265.2	28.36	51714	3.97	10569.5	29.34	108.5	24985	3.58	10584.0	31.85	144.7
newblue4	108898	4.17	15478.3	46.85	102261	3.90	8978.7	48.19	204.0	89931	3.59	8969.9	51.42	271.8
newblue5	205297	6.19	11910.3	84.61	182923	5.69	4574.1	88.05	431.8	164212	5.17	4551.6	96.11	650.9
newblue6	115663	7.28	18987.0	77.43	111559	6.63	3969.1	79.36	349.4	117278	6.05	3964.4	83.68	432.0
newblue7	189457	7.01	13416.0	160.57	166207	5.92	12080.8	166.79	630.9	164586	5.08	12052.7	183.35	846.0
average	97114.1	6.61	19199.4	53.5	89921.7	5.95	9095.7	55.30	240.6	86305.9	5.40	9084.0	59.84	359.1
ratio	**1.00**	**1.00**	**1.00**	1.00	**0.93**	**0.90**	**0.47**	1.03	–	**0.89**	**0.82**	**0.47**	1.12	–

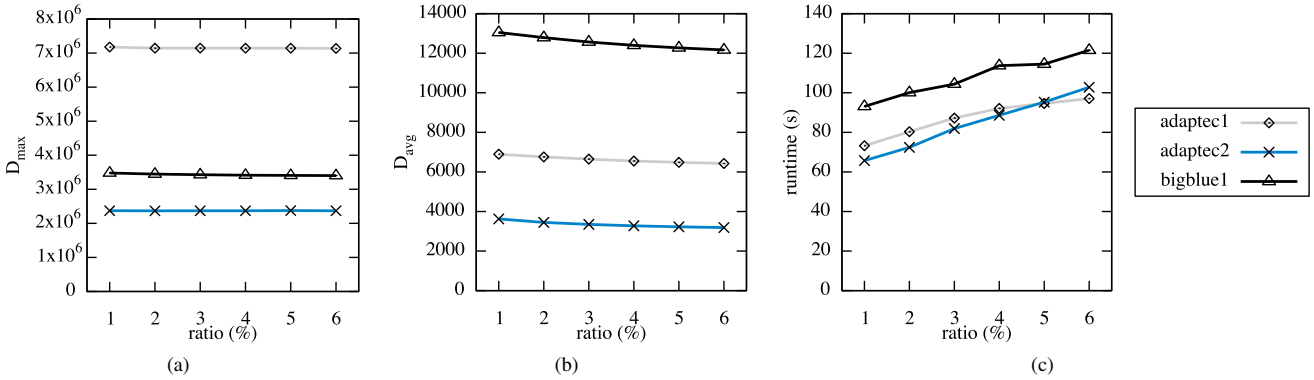


Fig. 8. Performance impact on different ratio values. (c) The impact of ratio on maximum delay; (b) The impact of ratio on average delay; (c) The impact of ratio on runtime.

list the resistances for wire layers and via layers, respectively. The resistances and capacitances of wires are directly from [11], while the via resistance values are normalized from industry settings in advanced technology nodes.

Table III compares NVM [17] with our incremental layer assignment tools TILA-1% and TILA-5%. In "**TILA-1%**" and "**TILA-5%**" the ratio value α are set to 1% and 5%, respectively. For each methodology, columns "OV#", "D_{avg}", "D_{max}", and "via#" list the overflow number, average delay, maximum delay, and total via number, separately. Besides, "CPU(s)" reports the runtime in seconds. We do not test our tools on test case "newblue3" as NCTU-GR [9] cannot generate global routing solution with zero edge overflow. We also can not report the results from another recent work [18], as for this benchmark suite their binary gets assertion fault before dumping out results.

From Table III we can see that in TILA-1%, when 1% of the most critical nets are shuffled layers, maximum delay can be reduced by 53% on the ISPD'08 benchmarks. Meanwhile, the overflow number and the average delay are reduced by 7% and 10%, respectively. The penalty of such timing improvement is that the via number is increased by only 3%. On the average, TILA-1% requires around 240 seconds for each test case. Compared with extreme fast net-by-net solver NVM, although our planner solves a global optimization problem, its runtimes are reasonable. For instance, based on [17], for test cases "adaptec1" and "adaptec5", NVM needs around 80 and 230 seconds, respectively. Our planner needs around 72 and 287 seconds, respectively. In TILA-5%, when 5% of the most critical nets are reassigned layers, the maximum delay is reduced by 53%. Meanwhile, the overflow number and the average delay are reduced by 11% and 18%, respectively. The penalty of TILA-5% is that the via number is increased by 12%. From Table III we can see that even small amount of critical nets (e.g. 1%) are considered, the maximum delay can be effectively optimized. When more nets are inputted in our planner, better average delay and less overflow number are expected. In addition, our framework is with good scalability, i.e., with problem size increases fivefold, the runtime of TILA-5% is just around one and half times of TILA-1%.

Critical net ratio α is a user-defined parameter to control how many nets are released to incremental layer assignment. In Table III, ratio α is set to 1% and 5%. Fig. 8 analyzes the impact of ratio value to the performance of incremental layer assignment framework. Fig. 8(a) shows the impact of ratio value on the maximum delay, where we can see that the maximum delays are kept the same. This means for these test cases, releasing 1% of critical nets is enough for maximum

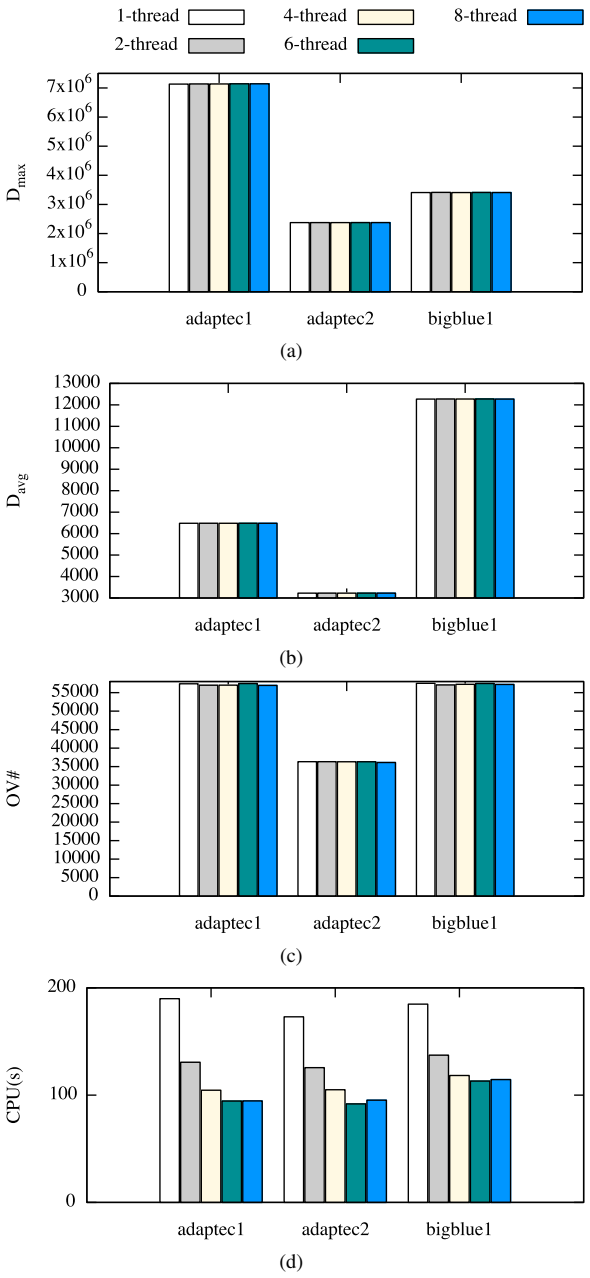


Fig. 9. Evaluation thread number impact on three test cases in ISPD'08 benchmark suite. (a) The impact on maximum delay; (b) The impact on average delay; (c) The impact on overflow; (d) The impact on runtime.

delay optimization. Fig. 8(b) shows the impact of ratio value on the average delay, where we can see increasing the ratio value can slightly improve the average delay. Fig. 8(c) is the impact on the runtime, where we can see that the runtime increases along with the increase of ratio value. From this figures we can see that the ratio value can provide a trade-off between average delay and the speed of our tool.

Our incremental layer assignment utilizes OpenMP [36] to implement multithreading. Fig. 9 analyzes the performance of our layer assignment framework under different thread numbers. From Fig. 9(a) and Fig. 9(b) we can see that the impact of thread number on both maximum delay and average delay is insignificant. Similarly, through Fig. 9(c) we can see the impact on overflow is also negligible. From Fig. 9(d) we can observe that more thread number can achieve more speed-up. However, when thread number is larger or equal to 8, the benefit to runtime is not clear. Therefore, in our implementation the thread number is set to 6.

B. Evaluation on 20nm Industry Benchmarks

In the second experiment, we test our incremental layer assignment framework on eight 20nm industry benchmarks ("Industry1" to "Industry8"). We called an industry tool to generate initial global routing and layer assignment solutions. Different from the preceding experiment, here we use industry resistance and capacitance values to calculate the wire delays and the via delays. Table IV lists the details of performance evaluation, where for each method columns "OV#", "D_{avg}", "D_{max}", and "via#" provide the overflow number, average delay, maximum delay, and total via number. Since all the critical nets are provided in the benchmarks, the critical and non-critical selection phases are skipped in this benchmark suite. We can see that compared with industry layer assignment solution, our framework can achieve 60% maximum delay improvement and 34% average delay improvement. The total via number after our iterative optimization is very similar to initial solution, i.e., 2% improvement. The initial layer assignment solution is with zero overflow, and our framework can also maintain such zero overflow performance. In summary, from Table IV we can see our incremental layer assignment framework can achieve significant timing improvement.

V. Conclusion

In this paper we have proposed a set of algorithms to the timing-driven incremental layer assignment problem. At first the mathematical formulation is given to search for optimal total wire delays and via delays. Then Lagrangian relaxation based method is proposed to iteratively improve the timing of all the nets. The Lagrangian relaxation subproblem (LRS) is modeled through min-cost flow model to provide effective integral solutions. In addition, multiprocessing of $K \times K$ partitions of the whole chip provides run time speed up. Our incremental layer assignment tool, TILA, is verified in both ISPD'08 and industry benchmark suites, and has demonstrated its effectiveness. As in emerging technology nodes, the routing algorithm should be able to adapt the heterogeneous layer structures, we believe this paper will stimulate more research for timing improvement in advanced routing.

Acknowledgment

This work is supported in part by NSF, Oracle, and Chinese Government Award for Outstanding Self-Financed Students Abroad. The authors would like to thank Shiyan Hu at Michigan Technological University and Wen-Hao Liu at Cadence Design Systems for helpful discussions and comments.

References

[1] J. H.-C. Chen, T. E. Standaert, E. Alptekin, T. A. Spooner, and V. Paruchuri, "Interconnect performance and scaling strategy at 7 nm node," in *IEEE International Interconnect Technology Conference (IITC)*, 2014, pp. 93–96.

[2] J. Cong, "An interconnect-centric design flow for nanometer technologies," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 505–528, 2001.

[3] M. Cho and D. Z. Pan, "BoxRouter: a new global router based on box expansion and progressive ILP," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 26, no. 12, pp. 2130–2143, 2007.

[4] J. Roy and I. Markov, "High-performance routing at the nanometer scale," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 6, pp. 1066–1077, 2008.

TABLE IV
PERFORMANCE COMPARISONS ON 20NM INDUSTRY BENCHMARKS

bench	Industry Layer Assignment				TILA				
	OV#	D_{avg}	D_{max}	via#	OV#	D_{avg}	D_{max}	via#	CPU(s)
Industry1	0	6204.0	68444.6	51805.0	0	3643.5	28565.7	49453.0	6.2
Industry2	0	6049.6	68713.0	52996.0	0	3750.0	27962.6	50676.0	6.7
Industry3	0	6025.4	81030.3	53905.0	0	3862.8	36434.6	51860.0	8.1
Industry4	0	5702.8	58478.5	56393.0	0	3621.9	24271.4	54145.0	9.4
Industry5	0	5531.4	78391.4	58944.0	0	3740.6	34766.2	56747.0	12.0
Industry6	0	5443.5	77803.0	60082.0	0	3633.6	34287.7	57879.0	14.2
Industry7	0	5066.0	114597.7	70658.0	0	3648.4	28844.7	71230.0	48.5
Industry8	0	4096.4	46893.7	75790.0	0	3034.9	20662.7	80500.0	120.3
average	0	5514.9	74294.0	60071.6	0	3616.9	29474.4	59061.3	28.2
ratio	**0**	**1.00**	**1.00**	**1.00**	**0**	**0.66**	**0.40**	**0.98**	-

[5] T.-H. Wu, A. Davoodi, and J. T. Linderoth, "GRIP: Global routing via integer programming," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 30, no. 1, pp. 72–84, 2011.

[6] M. D. Moffitt, "MaizeRouter: Engineering an effective global router," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 11, pp. 2017–2026, 2008.

[7] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang, "Multilayer global routing with via and wire capacity considerations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 5, pp. 685–696, 2010.

[8] Y.-J. Chang, Y.-T. Lee, J.-R. Gao, P.-C. Wu, and T.-C. Wang, "NTHU-Route 2.0: a robust global router for modern designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 12, pp. 1931–1944, 2010.

[9] W.-H. Liu, W.-C. Kao, Y.-L. Li, and K.-Y. Chao, "NCTU-GR 2.0: multithreaded collision-aware global routing with bounded-length maze routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 5, pp. 709–722, 2013.

[10] "ITRS," http://www.itrs.net.

[11] M.-K. Hsu, N. Katta, H. Y.-H. Lin, K. T.-H. Lin, K. H. Tam, and K. C.-H. Wang, "Design and manufacturing process co-optimization in nano-technology," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2014, pp. 574–581.

[12] Z. Li, C. J. Alpert, S. Hu, T. Muhmud, S. T. Quay, and P. G. Villarrubia, "Fast interconnect synthesis with layer assignment," in *ACM International Symposium on Physical Design (ISPD)*, 2008, pp. 71–77.

[13] S. Hu, Z. Li, and C. J. Alpert, "A polynomial time approximation scheme for timing constrained minimum cost layer assignment," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 112–115.

[14] ——, "A faster approximation scheme for timing driven minimum cost layer assignment," in *ACM International Symposium on Physical Design (ISPD)*, 2009, pp. 167–174.

[15] T.-H. Lee and T.-C. Wang, "Congestion-constrained layer assignment for via minimization in global routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 27, no. 9, pp. 1643–1656, 2008.

[16] K.-R. Dai, W.-H. Liu, and Y.-L. Li, "NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing," *IEEE Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 20, no. 3, pp. 459–472, 2012.

[17] W.-H. Liu and Y.-L. Li, "Negotiation-based layer assignment for via count and via overflow minimization," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2011, pp. 539–544.

[18] J. Ao, S. Dong, S. Chen, and S. Goto, "Delay-driven layer assignment in global routing under multi-tier interconnect structure," in *ACM International Symposium on Physical Design (ISPD)*, 2013, pp. 101–107.

[19] T.-H. Lee and T.-C. Wang, "Simultaneous antenna avoidance and via optimization in layer assignment of multi-layer global routing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 312–318.

[20] O. Coudert, J. Cong, S. Malik, and M. Sarrafzadeh, "Incremental CAD," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2000, pp. 236–244.

[21] C. Albrecht, "Efficient incremental clock latency scheduling for large circuits," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2006, pp. 1–6.

[22] H.-Y. Chang, I.-R. Jiang, and Y.-W. Chang, "Timing ECO optimization via Bézier curve smoothing and fixability identification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 31, no. 12, pp. 1857–1866, 2012.

[23] S. K. Karandikar, C. J. Alpert, M. C. Yildiz, P. Villarrubia, S. Quay, and T. Mahmud, "Fast electrical correction using resizing and buffering," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2007, pp. 553–558.

[24] S. Roy, P. M. Mattheakis, L. Masse-Navette, and D. Z. Pan, "Clock tree resynthesis for multi-corner multi-mode timing closure," in *ACM International Symposium on Physical Design (ISPD)*, 2014, pp. 69–76.

[25] J. A. Roy and I. L. Markov, "ECO-system: Embracing the change in placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 26, no. 12, pp. 2173–2185, 2007.

[26] T. Luo, D. A. Papa, Z. Li, C. N. Sze, C. J. Alpert, and D. Z. Pan, "Pyramids: an efficient computational geometry-based approach for timing-driven placement," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 204–211.

[27] Y. Zhang and C. Chu, "GDRouter: Interleaved global routing and detailed routing for ultimate routability," in *ACM/IEEE Design Automation Conference (DAC)*, 2012, pp. 597–602.

[28] C.-H. Hsu, H.-Y. Chen, and Y.-W. Chang, "Multi-layer global routing considering via and wire capacities," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 350–355.

[29] N. J. Naclerio, S. Masuda, and K. Nakajima, "The via minimization problem is NP-complete," *IEEE Transactions on Computers*, vol. 38, no. 11, pp. 1604–1608, 1989.

[30] A. P. Ruszczyński, *Nonlinear Optimization*. Princeton university press, 2006, vol. 13.

[31] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall/Pearson, 2005.

[32] R. G. Michael and S. J. David, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[33] M. Queyranne, "Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems," *Operations Research Letters*, vol. 4, no. 5, pp. 231–234, 1986.

[34] A. D. Gunawardena, S. Jain, and L. Snyder, "Modified iterative methods for consistent linear systems," *Linear Algebra and its Applications*, vol. 154, pp. 123–143, 1991.

[35] "LEMON," http://lemon.cs.elte.hu/trac/lemon.

[36] "OpenMP," http://www.openmp.org/.

[37] G.-J. Nam, C. Sze, and M. Yildiz, "The ISPD global routing benchmark suite," in *ACM International Symposium on Physical Design (ISPD)*, 2008, pp. 156–159.