Routing Topology and Time-Division Multiplexing Co-Optimization for Multi-FPGA Systems

Tung-Wei Lin[1], Wei-Chen Tai[1], Yu-Cheng Lin[2], and Iris Hui-Ru Jiang[12]
[1]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
Email: {waynelin567, willytai43, yclin7777777, huiru.jiang}@gmail.com

***Abstract*—Time-division multiplexing (TDM) is widely used to overcome bandwidth limitations and thus enhances routability in multi-FPGA systems due to the shortage of I/O pins in an FPGA. However, multiplexed signals induce significant delays. To evaluate timing degradation, nets with similar criticalities are often grouped to form NetGroups. In this paper, we propose a framework concerning routing topology and time-division multiplexing co-optimization for multi-FPGA systems. The proposed framework first generates high-quality topologies considering NetGroup criticalities. Then, inspired by column generation, TDM ratio assignment is solved optimally by Lagrangian relaxation. Experimental results show that our approach outperforms the top three entries of ICCAD 2019 CAD Contest. Moreover, our TDM ratio assignment algorithm can further improve the results of the top three winners to almost as good as ours.**

***Index Terms*—Multi-FPGA Systems, Time-Division Multiplexing, Lagrangian Relaxation, Column Generation, Simple Moving Average**

I. Introduction

Field-programmable gate arrays (FPGAs) have gained in popularity nowadays because of numerous advantages such as reprogrammability, short turn-around time, and non-recurring costs [1]. The use of FPGAs can be found in deep-learning implementation and acceleration [2], [3], ASIC prototyping [4], and cloud computing [5]. Compared with the traditional single-die FPGAs, multi-FPGA systems accommodate larger logic complexity and design capacity. However, existing multi-FPGA systems suffer from limited inter-chip communication bandwidth because the number of available I/O pins in an FPGA is disproportionately smaller than the number of inter-FPGA signals. Therefore, time-division multiplexing (TDM) [6] technique is proposed to remedy this issue. TDM overcomes pin limitations by multiplexing each physical I/O pin among multiple inter-FPGA signals. This technique allows an FPGA to transmit multiple signals in a system clock cycle and increases routing capability in multi-FPGA systems.

As shown in Fig. 1 (b)(c), consider signals 1, 2, 3 are scheduled to transmit through the edge (physical connection) between FPGA F_2 and FPGA F_3 with the TDM I/O structure, which is driven by a much faster TDM clock. Notice that since signal 1 has a higher priority, it accounts for 4 time slots, as opposed to signals 2 and 3, both of which only account for 2 time slots in every clock cycle. There are in total 8 time slots in one cycle in Fig. 1 (c). Therefore, the TDM ratio of signal 1 can be computed as $\frac{8}{4} = 2$, while that of signals 2 and 3 are both $\frac{8}{2} = 4$. Due to hardware limitations, the transmission of time-multiplexed signals are completed within the half cycle of the system clock. The TDM ratio of a signal thus must be a positive integer multiple of 2. Also, the reciprocals of TDM ratios of all the multiplexed signals on an edge must have a sum no greater than 1. Though TDM alleviates the problem caused by I/O pin shortage, the delay of transmitting signals with TDM is much larger than that without and thus deteriorates the timing of certain nets. A method to analyze the impact on timing is to form NetGroups where nets with similar timing criticalities are grouped together.

Multiple works have been proposed to approach the performance degradation throughout the compilation flow of a multi-FPGA system, involving netlist partitioning, inter-FPGA routing, TDM ratio

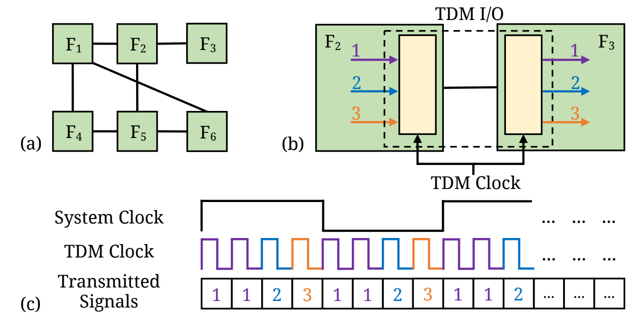

Fig. 1: (a) An example FPGA graph with 6 FPGAs and 7 edges. (b) The implementation of TDM I/O. (c) The waveforms of the system clock and the TDM clock along with the schedule to transmit signals.

assignment, pin assignment, etc. [4]. A typical compilation flow of multi-FPGA systems is shown in Fig. 2 (a). To divide netlists into multiple dies, Chen *et al.* [1] propose simultaneous signal partitioning and grouping. As for inter-FPGA routing, Turki *et al.* [7] and Farooq *et al.* [8] adapt Pathfinder [9] to iteratively route inter-FPGA signals. However, neither of the works takes the concept of NetGroup into consideration. To optimize TDM ratios, Inagi *et al.* [10] propose an ILP-based formulation to assign timing-critical nets to normal wires and non-critical nets to TDM wires but the natural limitations of ILP confine it to problems of smaller sizes. While Pui *et al.* propose to use conjugate gradient [2] and Lagrangian relaxation [3] to optimize system clock period, these works impose more restrictions on TDM ratios. Specifically, they require TDM ratios to be 1 or integer multiples of 8. Also, only signals with the same TDM ratio can be assigned to the same edge. Finally, Kuo *et al.* [11] address the pin assignment problem.

TDM ratio assignment along with inter-FPGA routing is especially important because TDM ratios directly affect the system clock period [2], [3] and are bounded by the inter-FPGA routing results. Therefore, we propose an inter-FPGA routing and TDM ratio assignment co-optimization framework, as shown in Fig. 2 (b), to generate a high-quality solution which will benefit subsequent compilation stages.

It is well known that Lagrangian relaxation (LR) has a strong relation with column generation (CG). Nonetheless, CG has several disadvantages. For instance, the simplex algorithm to derive optimal dual variables is computationally expensive and the tailing effect causes slow convergence rate [12]. On the other hand, in LR, the subgradient method to update Lagrangian multipliers (LMs) is inexpensive but convergence is not guaranteed when the parameters are not well-adjusted. To tackle this issue of LR, we propose a novel LM update strategy using the Sigmoid function and simple moving average (SMA) [13] inspired by timing optimization [14]–[16]. Our LR formulation emulates the CG formulation used in GRIP [17] to represent routing topologies. Also, we integrate the concept of pattern generation in cutting stock [18] into TDM ratio assignment. They are similar in the sense that cutting stock generates patterns for each fixed-width raw stock to minimize total waste while TDM ratio assignment generates patterns for each edge to minimize the total TDM ratio of the most critical NetGroup. To be more specific, cutting stock problem solves Knapsack problems [19] for its pricing problems while TDM

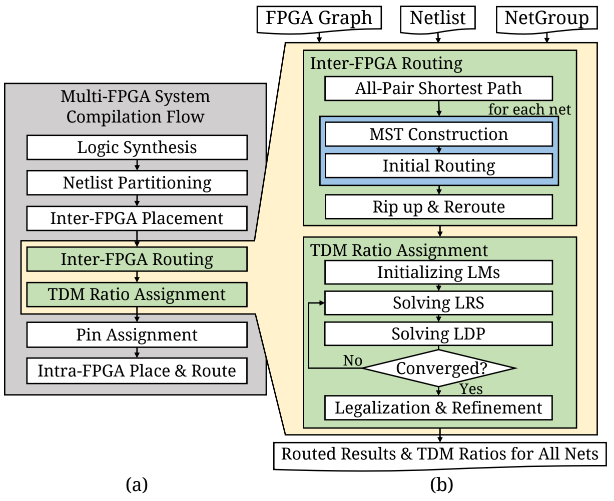

Fig. 2: (a) A typical compilation flow of multi-FPGA systems. (b) Our framework for inter-FPGA routing and TDM ratio assignment.

assignment solves Cauchy-Schwarz Inequality [20] for the Lagrangian relaxation subproblem. The main contributions of this paper are as follows:

- Inspired by CG, we formulate and solve the TDM ratio assignment problem with LR optimally, the solution of which is then legalized and refined by a fast and effective algorithm.
- We propose a novel update strategy for LMs utilizing the Sigmoid function and SMA.
- We propose a NetGroup-aware inter-FPGA routing scheme taking into account of the criticality of each NetGroup.

Our experiments are conducted on the benchmark suite released by ICCAD 2019 CAD Contest [21]. Experimental results show that our framework outperforms the top three winners. Moreover, our TDM ratio assignment algorithm can further improve the results of the top three winners to almost as good as ours.

The remainder of this paper is organized as follows: Sec. II introduces the problem of inter-FPGA routing and TDM ratio assignment along with the overview of our proposed method. Sec. III presents the proposed inter-FPGA routing algorithm. Sec. IV details the proposed TDM ratio assignment algorithm. Sec. V shows experimental results. Finally, Sec. VI concludes this work.

II. Routing and TDM Ratio Assignment

A. Problem Formulation

A multi-FPGA system can be represented as an undirected FPGA graph $\mathbf{G}(\mathbb{V},\mathbb{E})$, as shown in Fig. 1 (a), where $\mathbb{V}$ represents the set of FPGAs, and $\mathbb{E}$ represents the set of edges (physical connections) with TDM I/O structures. Given a netlist $\mathbb{N}$ of two-pin or multi-pin nets to be routed, each NetGroup g is formed by a subset of $\mathbb{N}$. A TDM ratio is a positive integer multiple of two, and the reciprocals of the TDM ratios on an edge must have a sum no greater than 1. The TDM ratio of a net n is the sum of the TDM ratios assigned to n on all its routed edges (i.e. $e \in n$). The TDM ratio of a NetGroup g is the sum of all TDM ratios of its nets (i.e. $n \in g$).

We formally define the problem of **inter-FPGA routing and TDM ratio assignment**:

Given an FPGA graph $\mathbf{G}(\mathbb{V},\mathbb{E})$, a set $\mathbb{N}$ of nets, a set $\mathbb{G}$ of NetGroups, the goal is to route all the nets in $\mathbb{N}$ via edges in $\mathbf{G}$ and assign a TDM ratio to each routed edge of each net such that the maximum TDM ratio over all NetGroups is minimized.

B. Overview

Fig. 2 (b) illustrates the proposed optimization framework, which consists of two major stages: inter-FPGA routing and TDM ratio assignment. The first stage generates routing topologies for each net considering the NetGroup information. The second stage assigns TDM ratios so that the maximum group TDM ratio is minimized.

III. Inter-FPGA Routing

With an FPGA graph, a circuit netlist, and NetGroup information, we perform NetGroup-aware inter-FPGA routing. The consideration of NetGroup criticalities in this stage is of great importance because optimizing TDM ratios is bounded by the routing topologies. All nets are first routed based on KMB algorithm [22], which is a Steiner tree approximation algorithm. Then, rip up and reroute takes place to further improve the routing result.

A. Initial Routing

According to KMB algorithm, for each net n, a complete graph $\mathbf{G}_c(\mathbb{V}_n)$ is first constructed, where $\mathbb{V}_n$ denotes the set of terminals (FPGAs) in n. The cost of edge (u, v) that connects vertices u and v in $\mathbf{G}_c(\mathbb{V}_n)$ is set as the shortest path distance from u to v in the FPGA graph $\mathbf{G}$. Second, we apply Kruskal's algorithm [23] to find the minimum spanning tree $\mathbf{G}_m(\mathbb{V}_n, \mathbb{E}_n)$ of $\mathbf{G}_c(\mathbb{V}_n)$, where $\mathbb{E}_n$ is the set of tree edges. To speed up this process, all-pair shortest path distances are calculated and stored in a look-up table beforehand.

It is known that net ordering impacts the quality of the routing result [24]. Therefore, we devise a scoring function $\theta(n)$ for each net to evaluate its criticality and route those with smaller $\theta(n)$ first, where

$$\theta(n) \;=\; \max_{g\in\mathbb{G}_n}\left\{\sum_{\hat{n}\in g} cost(\mathbf{G}_m(\mathbb{V}_{\hat{n}},\mathbb{E}_{\hat{n}}))\right\}. \tag{1}$$

$\mathbb{G}_n$ denotes the set of NetGroups that include net n, and $cost(\mathbf{G}_m(\mathbb{V}_n,\mathbb{E}_n))$ is the sum of edge costs in $\mathbf{G}_m(\mathbb{V}_n,\mathbb{E}_n)$. $cost(\mathbf{G}_m(\mathbb{V}_n,\mathbb{E}_n))$ estimates the number of edges net n requires to complete routing. It is obvious that the larger $\theta(n)$ is, the more likely net n is in the NetGroup that will eventually have the maximum group TDM ratio. Since the objective is to minimize the maximum group TDM ratio, nets with larger $\theta(n)$ are more important and should be routed with caution. Following the order of increasing $\theta(n)$, for each net n, we replace each edge (u, v) in $\mathbf{G}_m(\mathbb{V}_n,\mathbb{E}_n)$ by the shortest path found by Dijkstra's algorithm [25] on $\mathbf{G}$. When running Dijkstra's algorithm, the edge cost of $e \in \mathbb{E}$ is set as the number of nets already routed on e. In this way, nets routed later can avoid edges that are already crowded. Since nets routed earlier lack this information, nets that are less important (i.e. those with smaller $\theta(.)$) are routed first.

B. Rip up and Reroute

Rip up and reroute is widely used in routing [9]. To decide which nets to be ripped up, we devise a scoring function $\phi(g)$ to estimate the group TDM ratio of a NetGroup g without actually performing TDM ratio assignment:

$$\phi(g) = \sum_{n\in g}\psi(n),\;\text{ where }\psi(n) = \sum_{e\in n}|\mathbb{N}_e|. \tag{2}$$

$\mathbb{N}_e$ denotes the set of nets that route through edge e, while $|\mathbb{N}_e|$ denotes the cardinality of $\mathbb{N}_e$. $\psi(n)$ estimates the TDM ratio of a net n by summing the number of routed nets on all edges routed by n. The reasoning is that we temporarily assume the TDM ratios of all nets on edge e have the same value, i.e. $|\mathbb{N}_e|$. Since this assumption honors the constraint that the reciprocals of TDM ratios on an edge must have a sum no greater than 1, $\phi(g)$ is a reasonable upper bound on the group TDM ratio of g.

Then, we find the NetGroup $g_{\max}$ with the largest value of $\phi(.)$ and rip up all the nets that belong to $g_{\max}$. Then, we perform KMB algorithm [26] to reroute the nets sequentially. The edge cost of $e \in \mathbb{E}$ is set as the number of nets in $g_{\max}$ that are already routed on e. This cost function encourages the nets in $g_{\max}$ to route through edges not utilized by other nets in $g_{\max}$, which prevents nets within $g_{\max}$ from contending for the resource of TDM ratios on the same edge.

IV. TDM Ratio Assignment

Given the routing topology generated by the previous stage, we perform TDM ratio assignment with an objective to minimize the maximum group TDM ratio. In this stage, we first relax the TDM ratios to be non-negative real numbers and solve optimally by Lagrangian relaxation (LR). The proposed LR formulation emulates the Column Generation (CG) formulation of GRIP [17] and integrates the concept of pattern generation in cutting stock[1] [18]. Subsequently, the solution is legalized and refined using a fast and effective refinement algorithm.

LR is a general framework to solve constrained optimization problems. The Lagrangian relaxation subproblem ($\mathcal{LRS}$) can be obtained through dualizing the difficult constraints in the original primal problem ($\mathcal{PP}$). Therefore, $\mathcal{LRS}$ is an easier problem to solve and whose optimal solution is a lower bound (for minimization problems) on $\mathcal{PP}$ [27]. The violated constraints in $\mathcal{PP}$ are penalized by the Lagrangian multipliers (LMs) in $\mathcal{LRS}$. Additionally, the problem of maximizing the lower bound with respect to LMs is known as the Lagrangian dual problem ($\mathcal{LDP}$). Since it is often difficult to find the optimal LMs, which optimizes $\mathcal{PP}$, the iterative method is adopted to update LMs. We propose a novel update strategy to reduce the number of iterations until convergence using the Sigmoid function and simple moving average (SMA).

A. Lagrangian Relaxation

In the LR formulation to minimize the maximum group TDM ratio on continuous domain, we introduce an auxiliary variable $z > 0$. Define the parameter $a_{ne} = 1$ if net n is routed through edge e, $a_{ne} = 0$ otherwise. Let t_{en} be the TDM ratio assigned to net n on edge e. Therefore, the minimization problem under the constraint that the sum of the reciprocals of TDM ratios on an edge is less than or equal to one can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{t}} \quad & z && (\mathcal{PP}) \\ s.t. \quad & \sum_{n\in g}\sum_{e\in\mathbb{E}} a_{ne}t_{en} \le z, \quad \forall g\in\mathbb{G} \\ & \sum_{n\in\mathbb{N}_e}\frac{1}{t_{en}} \le 1, \quad \forall e\in\mathbb{E}. \end{aligned} \tag{3}$$

It is clear that the coefficients of the objective function and constraints in (3) are all positive and can be rewritten in posynomial form [27]. We dualize the first set of constraints in (3) by introducing a vector of non-negative LMs $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_g, \cdots, \lambda_{|\mathbb{G}|})$, where λ_g denotes the corresponding LM of NetGroup g. Let

$$L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z) = z + \sum_{g\in\mathbb{G}}\lambda_g\left(\sum_{n\in g}\sum_{e\in\mathbb{E}}a_{ne}t_{en} - z\right). \tag{4}$$

The Lagrangian relaxation subproblem ($\mathcal{LRS}$) becomes:

$$\begin{aligned} \min_{\boldsymbol{t}} \quad & L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z) && (\mathcal{LRS}) \\ s.t. \quad & \sum_{n\in\mathbb{N}_e}\frac{1}{t_{en}} \le 1, \forall e\in\mathbb{E}. \end{aligned} \tag{5}$$

The optimal solution of $\mathcal{LRS}$ given a set of $\boldsymbol{\lambda}$, $\mathcal{LRS}^*(\boldsymbol{\lambda})$, provides a lower bound on $\mathcal{PP}$. The maximization of $\mathcal{LRS}^*(\boldsymbol{\lambda})$ maximizes the lower bound. Therefore, the Lagrangian dual problem ($\mathcal{LDP}$) is defined as follows:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \mathcal{LRS}^*(\boldsymbol{\lambda}) && (\mathcal{LDP}) \\ s.t. \quad & \lambda_g \ge 0, \forall g\in\mathbb{G}. \end{aligned} \tag{6}$$

B. Solving $\mathcal{LRS}$

By the Karush-Kuhn-Tucker (KKT) condition $\partial L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)/\partial z = 0$, we have $1 - \sum_{g\in\mathbb{G}}\lambda_g = 0$, which the LMs must satisfy at the optimal solution of $\mathcal{PP}$. $L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)$ can thus be further simplified:

$$L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z) = \sum_{g\in\mathbb{G}}\lambda_g\sum_{n\in g}\sum_{e\in\mathbb{E}}a_{ne}t_{en}. \tag{7}$$

[1]The cutting-stock problem is the problem of cutting fixed-width pieces of stock material into pieces of specified widths while minimizing material wasted.

We introduce vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_n, \cdots, \pi_{|\mathbb{N}|})$, where π_n represents the sum of LMs of all NetGroups that include n, i.e. $\pi_n = \sum_{g\in\mathbb{G}_n}\lambda_g$. Rewrite (7) with the notations:

$$L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z) = \sum_{e\in\mathbb{E}}\sum_{n\in\mathbb{N}_e}\sum_{g\in\mathbb{G}_n}\lambda_g t_{en} = \sum_{e\in\mathbb{E}}\sum_{n\in\mathbb{N}_e}\pi_n t_{en}. \tag{8}$$

Since each summation term in (8) is independent of one another, $\mathcal{LRS}$ possesses optimal substructure property and can be modified as follows.

$$\begin{aligned} \min_{\boldsymbol{t}} L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z) &= \min_{\boldsymbol{t}}\sum_{e\in\mathbb{E}}\sum_{n\in\mathbb{N}_e}\pi_n t_{en} \\ &= \sum_{e\in\mathbb{E}}\left(\min_{\boldsymbol{t}_e}\sum_{n\in\mathbb{N}_e}\pi_n t_{en}\right), \end{aligned} \tag{9}$$

where $\boldsymbol{t}_e$ represents the TDM ratios of the nets on edge e, which can be viewed as a TDM ratio pattern of edge e. That is to say, we can sum the minimum of each substructure to obtain the minimum of $L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)$, where the substructure of edge e takes the form:

$$\begin{aligned} \min_{\boldsymbol{t}_e} \quad & \sum_{n\in\mathbb{N}_e}\pi_n t_{en} \\ s.t. \quad & \sum_{n\in\mathbb{N}_e}\frac{1}{t_{en}} = 1. \end{aligned} \tag{10}$$

Because the optimal solution to $\mathcal{LRS}$ must occur when $\sum_{n\in\mathbb{N}_e}\frac{1}{t_{en}} = 1$ in (5), we adopt a more stringent constraint in each substructure as stated in (10). This is analogous to the pattern generation in cutting stock, where a suitable pattern for each fixed-width raw stock is calculated by solving a Knapsack problem. Here, by applying Cauchy-Schwarz Inequality [20], we can generate the optimal pattern for each edge e:

$$\begin{aligned} \left(\sum_{n\in\mathbb{N}_e}\sqrt{\pi_n t_{en}}^2\right)\left(\sum_{n\in\mathbb{N}_e}\frac{1}{\sqrt{t_{en}}^2}\right) &\ge \left(\sum_{n\in\mathbb{N}_e}\sqrt{\pi_n}\right)^2 \\ \rightarrow \quad \sum_{n\in\mathbb{N}_e}\pi_n t_{en} \quad &\ge \left(\sum_{n\in\mathbb{N}_e}\sqrt{\pi_n}\right)^2. \end{aligned} \tag{11}$$

According to Cauchy-Schwarz Inequality, equality holds when

$$\frac{\frac{1}{t_{e1}}}{\sqrt{\pi_1}} = \frac{\frac{1}{t_{e2}}}{\sqrt{\pi_2}} = \cdots = \frac{\frac{1}{t_{en}}}{\sqrt{\pi_n}} = \cdots = \frac{\frac{1}{t_{e|\mathbb{N}_e|}}}{\sqrt{\pi_{|\mathbb{N}_e|}}}. \tag{12}$$

Combining (12) with the constraint in (10) gives

$$t_{en} = \frac{\sum_{\hat{n}\in\mathbb{N}_e}\sqrt{\pi_{\hat{n}}}}{\sqrt{\pi_n}}, \forall n\in\mathbb{N}_e. \tag{13}$$

After solving all substructures, the minimum value for $L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)$ and the corresponding pattern for each edge e can be obtained.

C. Solving $\mathcal{LDP}$

$\mathcal{LDP}$ aims at finding the suitable set of $\boldsymbol{\lambda}$ that penalizes the violated constraints in $\mathcal{PP}$ just the right amount. The proposed multiplier update strategy is inspired by the common technique used in timing optimization [14]–[16], where LMs are updated in proportion to the timing criticality of the timing arc then projected to meet the KKT condition. Similarly, we define:

$$\lambda_g^{i+1} = \lambda_g^i\left(1 + \frac{\sum_{n\in g}\sum_{e\in\mathbb{E}}a_{ne}t_{en}^i - z^i}{z^i}\right)^{K_g^i}, \tag{14}$$

where the superscripts represent the iteration and K the acceleration factor. The numerator in the parentheses is the gradient of $L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)$ with respect to λ_g, which can be derived by $\partial L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)/\partial\lambda_g$ in (4). (14) can be further simplified:

$$\lambda_g^{i+1} = \lambda_g^i\left(\frac{\sum_{n\in g}\sum_{e\in\mathbb{E}}a_{ne}t_{en}^i}{z^i}\right)^{K_g^i} = \lambda_g^i(\overline{TDM_g}^i)^{K_g^i}, \tag{15}$$

Algorithm 1 LR Based TDM Assignment

```
Inputs: $\epsilon$, $lim$
Output: optimized relaxed TDM ratio assignment, $LB$
 1: $i, z, LB \leftarrow 0$ // Initialize iteration, $z$, and lower bound
 2: $\lambda_g^i \leftarrow \frac{1}{|\boldsymbol{\lambda}^i|}, 1 < g \leq |\mathbb{G}|$
 3: repeat
 4:     Calculate $\boldsymbol{\pi}^i$
 5:     Solve $\mathcal{LRS}$
 6:     $z \leftarrow \max(\sum\sum a_{ne} t_{en})$
 7:     $LB \leftarrow L_{\boldsymbol{\lambda}}(\boldsymbol{t}, z)$
 8:     Solve $\mathcal{LDP}$
 9:     $i \leftarrow i + 1$
10: until $i \geq lim$ or $\epsilon \geq \frac{z-LB}{LB}$
```

where $\overline{TDM_g}^i$ represents the normalized group TDM ratio of NetGroup g with respect to the maximum group TDM ratio in the i-th iteration. As stated in (13), the optimal solution of t_{en} is in the form of a fraction, whose value is undefined when the denominator is zero. With the proposed update strategy, we are able to avoid the ill-conditioned situation since $\overline{TDM_g}^i \in (0, 1]$. Afterwards, we project $\boldsymbol{\lambda}^{i+1}$ to satisfy the KKT condition by dividing all λ_g^{i+1} by $\sum_{g\in\mathbb{G}} \lambda_g^{i+1}$.

Since the selection of the acceleration factor K affects the convergence rate and end result [15], [16], we detail on the proposed tuning method of K. Due to the fact that $\mathcal{PP}$ is convex, there is only one global optimum and no other local minimum. Namely, through iterations, the difference in $\boldsymbol{\lambda}^i$ and $\boldsymbol{\lambda}^{i+1}$ should diminish and $\boldsymbol{\lambda}$ should gradually converge to the optimal $\boldsymbol{\lambda}^*$. If the multiplier of a group λ_g^i has drastic change in value between iterations, this implies that it is still far from optimal at the i-th iteration. Therefore, the acceleration factor K_g^i should be assigned a larger value because $\overline{TDM_g}^i \leq 1$. On the other hand, those multipliers with stable values across iterations should have smaller acceleration factors. Thus, we replace the step function that is often used in previous works [15], [16] to adjust K with the Sigmoid function, which has a smooth transition of values. Moreover, in statistics, simple moving average (SMA) along with windowing is often used to smoothen a time series [13]. Hence, we are able to tell if λ_g is converging by calculating the SMA of $\overline{TDM_g}$. Based on the described observations, we define the acceleration factor K_g^i for group g in the i-th iteration:

$$\begin{aligned} K_g^i &= (\alpha - 1)\left(\frac{1}{1+e^{-\beta x_g^i}}\right) + 1 \\ x_g^i &= \frac{\overline{TDM_g}^i - avg_{w,g}^i}{std_{w,g}^i}, \end{aligned} \tag{16}$$

where α and β are preset constants that represent the magnitude and the steepness of the Sigmoid function, respectively. w is a preset constant that represents the window width. $avg_{w,g}^i$ and $std_{w,g}^i$ are the average and standard deviation of w sampled $\overline{TDM_g}$'s in the i-th iteration (i.e. $\overline{TDM_g}^{i-1}, \overline{TDM_g}^{i-2}, ..., \overline{TDM_g}^{i-w}$), respectively. Windowing provides a more robust indicator of convergence because it eliminates outdated data that are out of the window width w, which compensates for the naive initialization of $\boldsymbol{\lambda}^0$ as stated in line 2 in Algorithm 1. The overall flow of the LR based TDM ratio assignment can be found in Algorithm 1. With an input convergence criterion, ϵ, and a limit on the number of iterations, lim, we are able to produce an optimized continuous TDM ratio assignment along with a lower bound LB.

D. Connection with Column Generation

As stated in Sec. I, LR has a strong relation with CG. In fact, the LR formulation of TDM ratio assignment is inspired by its CG counterpart. Let $\mathbb{T}_e = (\boldsymbol{t}_{e1}, \boldsymbol{t}_{e2}, \cdots, \boldsymbol{t}_{ej})$ be the collection of *all* legal patterns for edge e, where $\boldsymbol{t}_{ej}$ represents the j-th pattern and t_{enj} represents the TDM ratio of net n of the j-th pattern. Let the binary decision variable $x_{ej} = 1$ if the j-th pattern in $\mathbb{T}_e$ is chosen, $x_{ej} = 0$ otherwise. Therefore, the integer linear program master problem ($\mathcal{ILPM}$) can be formulated:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & z & (\mathcal{ILPM}) \\ s.t. \quad & \sum_{j=1}^{|\mathbb{T}_e|} x_{ej} = 1, \quad \forall e \in \mathbb{E} \\ & \sum_{n\in g}\sum_{e\in\mathbb{E}}\sum_{j=1}^{|\mathbb{T}_e|} a_{ne} t_{enj} x_{ej} \leq z, \quad \forall g \in \mathbb{G}. \end{aligned}$$

The dual problem ($\mathcal{DP}$) of the linear relaxation of $\mathcal{ILPM}$ can be derived by introducing dual variables $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_e, \cdots, \mu_{|\mathbb{E}|})$ and $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_g, \cdots, \sigma_{|\mathbb{G}|})$, where $\mu_e \in \mathbb{R}, \forall e \in \mathbb{E}$ and $\sigma_g \leq 0, \forall g \in \mathbb{G}$. Thus, we get:

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\sigma}} \quad & \sum_{e\in\mathbb{E}} \mu_e & (\mathcal{DP}) \\ s.t. \quad & \mu_e + \sum_{g\in\mathbb{G}} \sigma_g \sum_{n\in g} a_{ne} t_{enj} \leq 0, \ \forall e \in \mathbb{E}, j = 1, \cdots, |\mathbb{T}_e| \\ & \sum_{g\in\mathbb{G}} \sigma_g \geq -1. \end{aligned}$$

However, it is impractical to enumerate all legal patterns on every edge e. Therefore, the restricted linear master problem ($\mathcal{RLMP}$) is derived, where $\mathbb{T}_e$ in $\mathcal{ILPM}$ is replaced by a candidate pattern set T_e and the binary requirement on x_{ej} is relaxed so that $0 \leq x_{ej} \leq 1$. T_e is a subset of all legal patterns (i.e. $T_e \subset \mathbb{T}_e$) to which patterns will be added after iterations.

Solving $\mathcal{RLMP}$ yields the optimal dual variables, $\boldsymbol{\mu}^*$ and $\boldsymbol{\sigma}^*$. If for an edge e, a new pattern $\boldsymbol{t}_{e\ell}$ can be found so that the constraint in $\mathcal{DP}$ is violated, i.e. $\sum_{g\in\mathbb{G}} \sigma_g \sum_{n\in g} a_{ne} t_{en\ell} > -\mu_e$, it suggests that $\boldsymbol{t}_{e\ell}$ should be added to T_e to reduce z. This itself is an optimization problem called the pricing problem ($\mathcal{PP}(T_e)$):

$$\begin{aligned} \min_{\boldsymbol{t}_{e\ell}} \quad & \sum_{g\in\mathbb{G}} |\sigma_g| \sum_{n\in g} a_{ne} t_{en\ell} \quad (\mathcal{PP}(T_e)) \\ s.t. \quad & \sum_{n\in\mathbb{N}_e} \frac{1}{t_{en\ell}} \leq 1, \end{aligned} \tag{17}$$

where the constraint that the TDM ratio must be an integer multiple of two is relaxed. Because the optimal solution must occur when the equality holds in the constraint in (17), $\mathcal{PP}(T_e)$ can be rewritten in the form identical to the substructure of e in $\mathcal{LRS}$ (i.e. (10)) and be solved with Cauchy-Schwarz Inequality. However, since LR possesses several advantages over CG, the problem of TDM ratio assignment is approached with LR instead.

E. Legalization and Refinement

Before the output of our final solution, the result obtained from LR is legalized and refined.

We iterate through every net on each edge and legalize the assigned TDM ratios by taking the ceiling. If this produces odd numbers, we increase them by 1 to make them even. Since the increase of a number decreases its reciprocal, the sum of the reciprocals of TDMs on each edge is guaranteed to be less than 1. Nevertheless, the summations on some edges may be far from 1 after the legalization. If this margin can be utilized efficiently, we can further decrease the maximum group TDM ratio.

We propose a refinement method that iteratively decreases the TDM ratios of some nets on an edge until no margin is left. To show how our refinement is done, we shall first introduce the following two observations:

- Not every net needs to be refined since refining nets that are not in the maximum NetGroup only wastes the margin that could be more efficiently utilized by those that are.
- The greater the TDM ratio, the higher the priority for a net to be refined. It is obvious that decreasing the same amount on a greater TDM ratio consumes less margin.

Based on these two observations, for each edge e, a set of candidates $\widetilde{\mathbb{N}}_e$ ($\widetilde{\mathbb{N}}_e \subseteq \mathbb{N}_e$) will be selected first, then the TDM ratios of the candidates will be decreased. $\widetilde{\mathbb{N}}_e$ is defined as:

$$\begin{aligned}\widetilde{\mathbb{N}}_e &= \arg\max_{n\in\mathbb{N}_e} \Gamma(n) \\ \Gamma(n) &= \max_{g\in\mathbb{G}_n} \Big\{ \sum_{\hat{n}\in g}\sum_{e\in\mathbb{E}} a_{\hat{n}e} t_{e\hat{n}} \Big\}.\end{aligned} \tag{18}$$

Function $\Gamma(n)$ returns the maximum group TDM ratio among $\mathbb{G}_n$ of net n. $\widetilde{\mathbb{N}}_e$ is composed of nets in $\mathbb{N}_e$ that have the maximum $\Gamma(.)$ value.

A naive implementation to decrease TDM ratios of the candidates is to heapify the TDM ratios and decrease the maximum value by 2 every iteration until no margin is left. However, it may take many iterations to complete the refinement process. Also, it is required to update the heap after each iteration, which causes great computational overhead. Therefore, this problem is addressed in another way.

We sort the candidates according to their TDM ratios in non-increasing order once in the beginning. To maintain this order without a heap, a reasonable decrement d is calculated every iteration and only TDM ratios with the maximum value are decreased by d. In sum, our refinement procedure is described below:

1) For each edge e, calculate its candidates $\widetilde{\mathbb{N}}_e$.
2) For each edge e, initialize margin ξ_e as 1-*tol*-$\sum_{n\in\mathbb{N}_e}\frac{1}{t_{en}}$, where *tol* is a preset tolerance due to floating point imprecision.
3) For each edge e, decrease the TDM ratios of its candidates by running Algorithm 2.

In Algorithm 2, starting from line 3 to 10, the TDM ratios of the candidates are refined and the margin is updated. m and d in line 5 represent the number of TDM ratios with the maximum value and the decrement, respectively. Note that there may be more than one TDM ratio with the same maximum value on edge e. The mechanism of updating ξ_e is

$$\xi_e^{i+1} = \xi_e^i - m\Big(\frac{1}{t_{\max}-d} - \frac{1}{t_{\max}}\Big), \tag{19}$$

because there are m TDM ratios with the value of $t_{\max}$ being decreased by d. The function CALCMD calculates the decrement, d, and updates m. Ideally, the entire margin is consumed after subtracting d from each $t_{\max}$. Simply put, we expect

$$\xi_e = m\Big(\frac{1}{t_{\max}-d} - \frac{1}{t_{\max}}\Big). \tag{20}$$

Thus, we get

$$d = \frac{\xi_e t_{\max}^2}{\xi_e t_{\max} + m}. \tag{21}$$

However, to maintain the order as described previously, d must not exceed b, which is the difference between the largest and the second largest value. Otherwise, d will be pruned in line 17. Then, d is legalized in line 18.

V. Experimental Results

The proposed optimization framework is implemented in the C++ programming language and evaluated on a workstation with 197GB memory and 2 Intel Xeon E5-2650 v2 @ 2.6GHz CPUs. The experiments are conducted on the benchmark suite released by ICCAD 2019 CAD Contest [21]. Table I lists the benchmark statistics.

Aside from comparing our results with the '1st', '2nd', and '3rd' place winners, based on their routing topologies, we apply our TDM ratio assignment algorithm to show that their solutions can be further improved. The binaries provided by the top three winners are evaluated on our platform for fair comparison. The LR convergence criterion ϵ is set as 0.27% from synopsys01 to synopsys05 and as 0.05% from synopsys06 to hidden03 because their lower bounds are much larger. Note that when $\epsilon = 0.27\%$, the difference between the maximum group TDM ratio and the lower bound is about 100 for synopsys01. The window size w, the magnitude α, and the steepness β of the Sigmoid function are set as 10, 3, and 10 for all benchmarks, respectively.

Algorithm 2 Decreasing TDM Ratios

```
Inputs: edge $e$, legalized TDM ratios on $e$, $\widetilde{\mathbb{N}}_e$, $\xi_e$
Output: the refined result
1: $\widetilde{\mathbb{T}}_e \leftarrow$ TDM ratios of the candidates sorted in non-increasing order
2: $m \leftarrow 0$
3: while $\xi_e$ is large enough do
4:     $t_{\max} \leftarrow \widetilde{\mathbb{T}}_e[0]$
5:     $m, d \leftarrow$ CALCMD($m, \xi_e, t_{\max}, \widetilde{\mathbb{T}}_e$)
6:     for $j \leftarrow 0$ to $m-1$ do
7:         $\widetilde{\mathbb{T}}_e[j] \leftarrow \widetilde{\mathbb{T}}_e[j] - d$
8:     end for
9:     Update margin $\xi_e$ with Eq. 19
10: end while

11: function CALCMD($m, \xi_e, t_{\max}, \widetilde{\mathbb{T}}_e$)
12:     repeat
13:         $m \leftarrow m + 1$
14:     until $t_{\max} \neq \widetilde{\mathbb{T}}_e[m]$
15:     $b \leftarrow t_{\max} - \widetilde{\mathbb{T}}_e[m]$ // the budget for the decrement
16:     $d \leftarrow$ Calculate decrement with Eq. 21
17:     $d \leftarrow \min(d, b)$
18:     $d \leftarrow$ greatest even integer less than or equal to $d$
19:     return $m, d$
20: end function
```

Table II shows our results and comparisons with the top three winners. It can be seen that the proposed framework achieves much better quality. Specifically, an average improvement by 4.53%, 1.89%, and 1.13% over '1st', '2nd', and '3rd' is observed, respectively.

As stated in Sec. IV-E, legalization and refinement are performed after LR to satisfy all constraints on TDM ratios. In Table II, it can be seen that the maximum group TDM ratios ('GTR$_{\text{max}}$') of all benchmarks after refinement are consistently better than those without ('GTR$_{\text{noref}}$'). Furthermore, legalization and refinement are efficient and only account for 0.44% of total runtime, as shown in Fig. 3 (a).

With LR being the runtime bottleneck, we present the LR convergence graph of synopsys01 in Fig. 3 (b). It is obvious that the overall convergence curve is smooth except for the little bump at about the sixth iteration. Our conjecture is that it is caused by the LM projection to satisfy the KKT condition.

Finally, we read in the routing topologies of the top three winners and apply our TDM ratio assignment algorithm, which includes LR, legalization, and refinement. The runtimes of TDM ratio assignment only ('Time$_{\text{TA}}$') are listed in Table II, where the runtime of file I/O is subtracted for fair comparison. We can greatly improve the solutions of the top three winners and achieve results almost as good as our proposed framework. For pursuing a better solution quality, our runtimes are acceptable for practical use of large-scale multi-FPGA systems.

The lower bounds ('LBs') on the maximum group TDM ratios at convergence are listed in Table II as well. As stated in Sec. III, TDM ratios are bounded by the routing topologies. It is theoretically impossible to achieve maximum group TDM ratio lower than LB given that the integrality constraint of TDM ratios is relaxed in LR. Take synopsys04 and synopsys05 for example, the LB of the top three winners are all larger than our legalized and refined solution. Therefore, with their routing topologies, there exists no TDM ratio assignment solution that is able to yield a better result than ours.

VI. Conclusion

This paper presents an inter-FPGA routing and TDM ratio co-optimization framework. First, routing topologies of high quality are generated taking into account of the NetGroup criticalities. Second, an LR formulation inspired by CG is proposed to solve TDM ratio assignment optimally, followed by a fast and effective legalization and refinement algorithm. Furthermore, a novel LM update strategy is proposed to achieve faster LR convergence. Experimental results based on ICCAD 2019 CAD Contest show that the proposed framework is effective and outperforms the top three winners. Moreover, the

TABLE I: Statistics of the ICCAD 2019 CAD Contest benchmark suite.

Benchmark	synopsys01	synopsys02	synopsys03	synopsys4	synopsys05	synopsys06	hidden01	hidden02	hidden03
#FPGAs	43	56	114	229	301	410	73	157	487
#Edges	214	157	350	1087	2153	1852	289	803	2720
#Nets	6.85×10^4	3.50×10^4	3.03×10^5	5.52×10^5	8.81×10^5	7.86×10^5	5.43×10^4	6.11×10^5	7.21×10^5
#NetGroups	4.06×10^4	5.60×10^4	3.35×10^5	4.65×10^5	8.79×10^5	9.11×10^5	5.04×10^4	5.02×10^5	8.87×10^5

TABLE II: Comparison with top three winners of ICCAD 2019 CAD Contest. 'TA' stands for our TDM Ratio Assignment algorithm, 'GTR_{max}' stands for the maximum group TDM ratio, 'LB' is the lower bound when convergence occurs, 'Iter' is the number of iterations until convergence, 'Time_{all}' stands for the runtime of the program while 'Time_{TA}' stands for the runtime of TDM ratio assignment only, '$\text{GTR}_{\text{noref}}$' represents the maximum group TDM ratio after legalization without refinement. All runtimes are reported in seconds.

Benchmark		synopsys01	synopsys02	synopsys03	synopsys04	synopsys05	synopsys06	hidden01	hidden02	hidden03	Ratio
1st	GTR_{max}	40,498	32,008,188	128,205,938	7,049,316	5,190,132	15,754,099,480	409,306,936	45,942,235,550	4,867,325,852	1.045
	Time_{all}	0.387	0.575	7.100	20.692	47.863	82.228	1.205	27.562	84.181	0.259
1st +TA	GTR_{max}	37,090	31,622,072	127,061,384	6,206,132	4,540,842	15,743,818,488	408,516,862	45,934,879,556	4,859,550,180	1.001
	LB	36,955	31,547,226	126,722,203	6,199,868	4,535,910	15,735,999,959	408,313,892	45,930,800,654	4,857,160,610	1.001
	Iter	60	17	45	241	205	16	9	470	34	1.337
	Time_{TA}	0.604	0.950	21.861	233.350	252.292	200.467	1.384	1,503.890	369.918	1.812
2nd	GTR_{max}	40,190	32,093,856	129,290,206	6,334,724	4,613,110	15,759,302,934	409,891,196	45,952,594,974	4,872,933,150	1.019
	Time_{all}	0.866	0.791	11.074	66.365	144.283	109.127	1.610	38.900	125.514	0.459
2nd +TA	GTR_{max}	37,184	31,588,462	127,040,124	6,203,572	4,538,206	15,743,640,666	408,501,736	45,934,800,476	4,859,377,142	1.001
	LB	37,086	31,545,724	126,700,671	6,196,627	4,534,101	15,735,809,893	408,298,954	45,930,656,201	4,857,004,279	1.001
	Iter	56	15	47	231	214	16	13	306	34	1.175
	Time_{TA}	0.508	0.964	26.277	177.140	283.146	160.464	1.527	681.736	292.768	1.323
3rd	GTR_{max}	37,970	31,739,100	127,594,618	6,397,142	4,662,236	15,749,500,940	408,864,514	45,941,043,974	4,865,375,688	1.011
	Time_{all}	1.126	1.158	10.910	30.803	69.500	62.716	2.031	35.307	62.349	0.452
3rd +TA	GTR_{max}	37,098	31,642,464	127,270,870	6,220,422	4,539,304	15,744,786,882	408,523,146	45,934,499,142	4,860,416,914	1.002
	LB	36,938	31,573,875	126,934,795	6,204,359	4,535,243	15,737,428,024	408,326,740	45,931,024,081	4,857,979,920	1.001
	Iter	62	17	49	172	227	18	9	78	34	0.920
	Time_{TA}	0.444	1.123	28.051	136.239	331.423	175.820	1.350	242.230	354.342	1.127
Ours	$\text{GTR}_{\text{noref}}$	37,084	31,646,556	127,207,730	6,191,608	4,540,368	15,749,712,634	408,785,238	45,936,434,558	4,862,187,438	1.001
	GTR_{max}	**37,030**	**31,572,618**	**127,010,440**	**6,183,236**	**4,530,238**	**15,743,597,786**	**408,485,298**	**45,930,934,516**	**4,859,356,834**	**1**
	Time_{all}	0.756	1.468	27.459	143.650	448.940	267.530	3.930	285.370	410.100	1
	LB	36,937	31,534,791	126,674,838	6,174,412	4,527,638	15,735,824,363	408,311,023	45,930,673,641	4,856,987,237	1
	Iter	62	15	47	185	320	16	15	106	34	1
	Time_{TA}	0.480	0.817	20.179	120.438	385.511	149.124	1.514	229.310	265.431	1

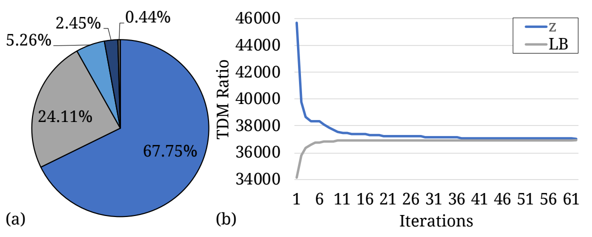

Fig. 3: (a) The pie chart to illustrate the average runtime of each stage in the proposed framework. (b) The LR convergence graph of synopsys01. The labels in (a) are as follows: 🟦Lagrangian Relaxation: 67.75%, ⬜Inter-FPGA Routing: 24.11%, 🟦Input File Parsing: 5.26%, ⬛Output File Writing: 2.45%, ⬛Legalization & Refinement: 0.44%.

proposed TDM ratio assignment algorithm can further improve their results to almost as good as ours.

References

[1] S.-C. Chen et al., "Simultaneous partitioning and signals grouping for time-division multiplexing in 2.5 D FPGA-based systems," in *Proc. ICCAD*, 2018, pp. 1–7.

[2] C.-W. Pui et al., "An analytical approach for time-division multiplexing optimization in multi-FPGA based systems," in *Proc. SLIP*, 2019, pp. 1–8.

[3] C.-W. Pui and E. F. Y. Young, "Lagrangian relaxation-based time-division multiplexing optimization for multi-FPGA systems," in *Proc. ICCAD*, 2019, pp. 1–8.

[4] W. N. Hung and R. Sun, "Challenges in large FPGA-based logic emulation systems," in *Proc. ISPD*, 2018, pp. 26–33.

[5] G. R. Chiu et al., "Flexibility: FPGAs and CAD in deep learning acceleration," in *Proc. ISPD*, 2018, pp. 34–41.

[6] J. Babb et al., "Logic emulation with virtual wires," *TCAD*, vol. 16, no. 6, pp. 609–626, 1997.

[7] M. Turki et al., "Iterative routing algorithm of inter-FPGA signals for multi-FPGA prototyping platform," in *Proc. ARC*, 2013, pp. 210–217.

[8] U. Farooq et al., "Inter-FPGA routing environment for performance exploration of multi-FPGA systems," in *Proc. RSP*, 2016, pp. 1–7.

[9] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," in *Proc. FPGA*, 1995, pp. 111–117.

[10] M. Inagi et al., "Globally optimal time-multiplexing in inter-FPGA connections for accelerating multi-FPGA systems," in *Proc. FPL*, 2009, pp. 212–217.

[11] W.-S. Kuo et al., "Pin assignment optimization for multi-2.5D FPGA-based systems," in *Proc. ISPD*, 2018, pp. 106–113.

[12] D. Huisman et al., *Combining column generation and Lagrangian relaxation*, 2005, pp. 247–270.

[13] A. Raudys et al., "Moving averages for financial data smoothing," in *Information and Software Technologies*, 2013, pp. 34–45.

[14] A. Sharma et al., "Fast Lagrangian relaxation based gate sizing using multi-threading," in *Proc. ICCAD*, 2015, pp. 426–433.

[15] A. Sharma et al., "Rapid gate sizing with fewer iterations of Lagrangian relaxation," in *Proc. ICCAD*, 2017, pp. 337–343.

[16] D. Mangiras et al., "Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering," *TCAD (Early Access)*, 2019.

[17] T. Wu et al., "GRIP: Global Routing via Integer Programming," *TCAD*, vol. 30, no. 1, pp. 72–84, 2011.

[18] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Oper. Res.*, vol. 9, no. 6, pp. 849–859, 1961.

[19] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, 1972, pp. 85–103.

[20] A. Z. Grinshpan, "General inequalities, consequences and applications," *Advances in Applied Mathematics*, vol. 34, no. 1, pp. 71–100, 2005.

[21] "2019 CAD Contest: System-level FPGA Routing with Time Division Multiplexing Technique," http://iccad-contest.org/2019/problems.html.

[22] L. Kou et al., "A fast algorithm for Steiner trees," *Acta informatica*, vol. 15, no. 2, pp. 141–145, 1981.

[23] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," in *Proc. American Mathematical Society*, vol. 7, no. 1, 1956, pp. 48–50.

[24] H.-Y. Chen and Y.-W. Chang, "Global and detailed routing," *Electronic Design Automation: Synthesis, Verification, and Test*, pp. 687–750, 2009.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[26] K. Mehlhorn, "A faster approximation algorithm for the Steiner problem in graphs," *Information Processing Letters*, vol. 27, no. 3, pp. 125–128, 1988.

[27] R. K. Ahuja et al., "Lagrangian relaxation and network optimization," *Network Flows: Theory, Algorithms, and Applications*, pp. 598–648, 1993.