Two Approaches for Timing-Driven Placement by Lagrangian Relaxation

Gang Wu
Synopsys, Inc., Hillsboro, OR 97124
Email: gang.wu@synopsys.com

Chris Chu
Iowa State University, Ames, IA 50011
Email: cnchu@iastate.edu

***Abstract*—In this work, we propose Lagrangian relaxation based algorithms to optimize both circuit performance and total wirelength at the global placement stage. We introduce a general timing-driven global placement problem formulation that is applicable to three different circuit design styles: synchronous circuits, synchronous circuits with sequential optimization techniques and asynchronous circuits. Lagrangian relaxation is applied to handle the timing constraints of the formulated problem. Based on how the cell spreading constraints are handled, two different approaches are proposed: One approach handles the spreading constraints inside the Lagrangian relaxation framework and transforms the timing-driven placement problem into a series of weighted wirelength minimization problems, which can be solved by directly leveraging existing wirelength-driven placers. The other approach handles the spreading constraints outside the Lagrangian relaxation framework. Thus, only timing constraints need to be taken care of in the Lagrangian relaxation framework and better solutions can be expected. In both approaches, we simplified the Lagrangian relaxation subproblem using Karush-Kuhn-Tucker conditions. Our algorithms are implemented based on a state-of-the-art wirelength-driven quadratic placer. The experiments demonstrate that the proposed algorithms are able to achieve significant improvements on circuit performance compared with a commercial wirelength-driven placement flow and a commercial asynchronous timing-driven placement flow.**

I. Introduction

Placement is a critical step in VLSI design flow, as the placement quality and optimization metrics can greatly affect the performance, routability, heat distribution and power consumption of a design [1]. In advanced technology, the importance of placement continues to grow, since it plays an important role to determine the interconnect delay, which has been a dominating factor of the circuit delay.

Traditional wirelength-driven placement algorithms only consider minimizing the total chip wirelength and do not take into account circuit timing during the optimization process. As a result, the wirelength-driven placement algorithms are no longer sufficient to close timing at modern technology nodes, which often have stricter timing constraints. Therefore, the quests for timing-driven placement (TDP) algorithms start to receive closer attention.

Timing-driven placement problem has been extensively studied for decades. One category of TDP algorithms optimizes the circuit timing by capturing the timing criticality of each net through net weighting (e.g., [2] [3]) or net constraints (e.g.,[4]). These algorithms are often referred to as net based algorithms. However, net based approaches only estimate the circuit timing locally and do not have a global view on the entire timing paths. Another category of TDP algorithms directly work on a set of critical timing paths and ensure all considered timing paths meet the constraints. This category is often referred to as path based algorithms (e.g., [5]). To avoid explicitly considering all the timing paths, the number of which can be exponential to the circuit size, timing graph based approaches embed the timing graph into the formulation of timing-driven placement problem, and all topological timing paths can then be implicitly considered [6].

Depending on the specific circuit design style, the corresponding TDP problem can target at optimizing either the most critical path between registers (as in synchronous circuits) or the most critical cycle (as in asynchronous circuits [7][8][9] or synchronous circuits with sequential optimization techniques [10]). Normally, at the global placement stage, cells are placed to improve circuit performance based on a rough timing estimation, with a small amount of cell overlapping allowed. Next, at the legalization stage, cells are moved to legal locations with the disturbance of the timing at global placement stage minimized [10] or performing further timing improvement by applying certain techniques [11][12].

Lagrangian relaxation (LR) technique has been widely used for timing-driven placement due to several important reasons: First, it relaxes the complex timing constraints in the formulated problem and results in an LR subproblem which is much easier to solve [13]. Second, given the special circuit structure and the Karush-Kuhn-Tucker (KKT) conditions, we are able to further simplify the LR subproblem to get rid of its arrival time and cycle time variables [14]. Finally, Lagrangian relaxation has great flexibility and is capable of handling various objectives and complex design constraints. LR-based algorithms have shown to be successful in handing timing constraints in many previous works. In [15][16], Lagrangian relaxation is combined with the wirelength-driven placer GORDIAN [17] to solve the timing-driven placement problem for synchronous circuits. The relaxed LR-subproblem is either solved as a quadratic program [15] or through the resistance network approach [16]. In [10], Lagrangian relaxation is used as a refinement step during the global placement stage, in order to improve the circuit performance for synchronous circuits with sequential optimization techniques.

Apart from the complex timing constraints of the timing-driven placement problem, we also need to consider its cell

This work is supported in part by NSF award CCF-1219100.

spreading constraints, which can be even more difficult to handle. The reason is that these constraints are often non-convex, not differentiable, and may even do not have an exact formulation [18][19]. Thus, it is not straightforward to solve them mathematically.

In this paper, we apply Lagrangian relaxation to handle the timing constraints, while also explore different approaches to incorporate the cell spreading techniques with the LR framework. In particular, we formulate a general timing-driven global placement problem which is applicable to synchronous circuits, synchronous circuits with sequential optimization techniques, and asynchronous circuits. We propose two different approaches for handling the cell spreading constraints: One approach handles the spreading constraints inside the LR framework and the LR subproblem becomes a weighted wirelength minimization problem, which can be solved effectively using existing wirelength-driven placers with the ability to handle net weights. The other approach spreads the cells outside the LR framework and the resulting LR subproblem becomes an unconstrained optimization problem which can be solved optimally using standard mathematical techniques.

The proposed approaches are implemented based on the state-of-art quadratic placer POLAR [19]. We evaluated both of our approaches on quasi-delay-insensitive (QDI) Pre-Charged Half Buffer (PCHB) asynchronous designs synthesized using the Proteus asynchronous synthesis flow [20]. The experimental results of both approaches are compared with a commercial wirelength-driven placement flow and a commercial timing-driven placement flow.

The main contributions of this paper are as follows:

- A general problem formulation for the timing-driven placement problem is proposed which can be applied to a large variety of design styles.
- Two different approaches of applying Lagrangian relaxation to the formulated problem are presented.
- For both approaches, better computational efficiency is achieved by simplifying the LR subproblem using the KKT conditions.
- An effective approach to handle timing-driven placement at the detailed placement stage is proposed.
- Promising experimental results are presented.

The rest of this paper is organized as follows. Section II presents the problem formulations for various design styles. Section III elaborates two different approaches we used to apply Lagrangian relaxation framework to the formulated timing-driven placement problem. Section IV presents the detailed implementation of the proposed approaches. Section V shows our experimental results compared with other placement approaches. Finally, Section VI concludes the paper.

II. Problem Formulations

In Section II-A, we will first discuss the problem formulation for the pure wirelength-driven placement problem. Next, in Section II-B, we introduce three different design styles: synchronous circuits, synchronous circuits with sequential optimization techniques, and asynchronous circuits. Their corresponding formulations for the timing-driven placement problem will also be presented. Finally, in Section II-C, we summarize all the design styles and propose a general problem formulation covering all of them.

A circuit can be represented by a hypergraph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ corresponds to the set of cells, and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ corresponds to the set of nets. In addition, we use vector $\mathbf{x} = (x_1, x_2, \cdots, x_{|V|})$ to denote the x-coordinates of the cells and vector $\mathbf{y} = (y_1, y_2, \cdots, y_{|V|})$ to denote the y-coordinates of the cells.

The wirelength of a hyperedge e depends on the locations of the cells associated to it. Thus, we use $\mathrm{WL}_e(\mathbf{x}, \mathbf{y})$ to denote the wirelength of a hyperedge e. The definition of this wirelength function is ignored at this point to make our problem formulation general. In practice, the wirelength function can be modeled as HPWL, quadratic, Log-Sum-Exp function, etc. [1].

Some notations used in this paper are shown in Table I.

Table I. The key notations used in this paper.

$\mathcal{WDP}$	wirelength-driven placement
$\mathcal{TDP}$	timing-driven placement
$\mathcal{STDP}$	synchronous timing-driven placement
$\mathcal{STDPS}$	synchronous timing-driven placement with sequential optimization techniques
$\mathcal{ATDP}$	asynchronous timing-driven placement
$\mathcal{GTDP}$	general timing-driven placement
$\mathcal{LRS}$	Lagrangian relaxation subproblem
$\mathcal{LRS}$-$\mathcal{S}$	simplified Lagrangian relaxation subproblem
$\mathcal{LDP}$	Lagrangian dual problem
a_i	arrival time variable at node i
D_{ij}	delay value associated with edge (i, j)
λ_{ij}	Lagrange multiplier associated with edge (i, j)
τ	cycletime variable
$q(\boldsymbol{\lambda})$	optimal value of $\mathcal{LRS}$ for a given vector $\boldsymbol{\lambda}$
$\mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x}, \mathbf{y}, \mathbf{a}, \tau)$	objective function of $\mathcal{LRS}$
$\mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x}, \mathbf{y})$	objective function of $\mathcal{LRS}$-$\mathcal{S}$

A. *Wirelength-driven Placement Problem ($\mathcal{WDP}$)*

The wirelength-driven global placement tries to minimize the total chip wirelength by assigning cells to locations on the chip, while keeping the cells spread out. Therefore, the wirelength-driven placement problem can be formulated as:

$$\begin{aligned} \mathcal{WDP}: \quad & \text{Minimize} \quad \sum_{e \in E} \mathrm{WL}_e(\mathbf{x}, \mathbf{y}) \\ & \text{Subject to} \quad \text{cell spreading constraints} \end{aligned}$$

Here, we also skip the details about the cell spreading constraints to make our problem formulation general. Various techniques can be practically used to implement the cell spreading constraints, such as the center-of-gravity (COG) constraints [17], spreading forces [3], density penalty functions [21], etc.

B. *Timing-driven Placement Problems*

To further capture the timing information of the circuit, we introduce a timing graph $G' = (V, E')$, where E' denotes the set of timing edges. In particular, we use V_I to denote the set of vertices representing the starting points of timing paths, i.e., the output pins of registers or the primary inputs. Similarly, we use V_O to denote the set of vertices representing the timing path end points, which are the input pins of registers or the primary outputs. In addition, let $AT = \{a_1, a_2, \ldots, a_{|V|}\}$ be the set of arrival time variables associated with each node. Let τ denotes the minimum cycle time to ensure hazard-free operation of the circuit. Also, we denote the delay associated with the edge between node v_i and v_j in the timing graph as D_{ij}, whose value depends on the interconnection between v_i and v_j and hence depends on the placement.

1) Synchronous timing-driven placement ($\mathcal{STDP}$):

The cycle time for synchronous circuits is bounded by the delay of its longest timing path. However, the total number of timing paths is exponential to the circuit size. Therefore, instead of explicitly considering all timing paths, we capture the circuit timing using the set of worst case arrival times, which can be calculated by propagating the largest arrival time at each node:

$$a_i + D_{ij} \leq a_j \quad \forall (i,j) \in E'$$

Then, the synchronous timing-driven placement problem, which simultaneously minimizes the total wirelength and cycle time can be formulated as:

$$\begin{aligned}
\mathcal{STDP}: \quad & \text{Minimize} \quad \sum_{e \in E} \text{WL}_e(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
& \text{Subject to} \quad a_i + D_{ij} \leq a_j \quad \forall (i,j) \in E' \qquad (1) \\
& \qquad a_k \leq \tau \qquad \forall k \in V_O \qquad (2) \\
& \qquad a_k \geq W_k \quad \forall k \in V_I \qquad (3) \\
& \qquad \text{cell spreading constraints}
\end{aligned}$$

Here, α is a constant value which we can use to adjust the effort between optimizing wirelength and cycle time. W_k denotes the constant delay value that signal arrives at the primary inputs or the output of registers. Please note that we ignore the hold time violations in the formulation of $\mathcal{STDP}$, as designers normally only consider the longest timing paths at the global placement stage. The shortest paths causing hold time violations are fixed at a later stage, i.e., after detailed placement and routing.

2) Synchronous timing-driven placement with sequential optimization techniques ($\mathcal{STDPS}$):

Retiming [22] [23] and clock skew scheduling [24] are two commonly used sequential optimization methods. Retiming improves the circuit performance through changing the structural location of registers. Instead, clock skew scheduling preserves the circuit structure, while intentionally introduces skews to registers to improve the performance of a circuit.

Let c denote a timing loop composed by a set of timing path segments. The basic idea for both of the above two sequential optimization methods is to perform a coarse balancing on the timing budgets of the path segments along the timing loop. Therefore, the optimization potential of these methods is bounded by the maximum mean delay over all timing loops:

$$\tau \geq \max_{c \subset G'} \left\{ \frac{\sum_{(i,j) \in c} D_{ij}}{\text{\# of registers in } c} \right\}$$

Instead of enumerating all the timing loops whose number is exponential to circuit size, we can simply obtain the cycle time by solving the following linear program [25]:

$$\begin{aligned}
& \text{Minimize} \quad \tau \\
& \text{Subject to} \quad a_i + D_{ij} - m_{ij}\tau \leq a_j \quad \forall (i,j) \in E'
\end{aligned}$$

where $m_{ij} = 1$ if the corresponding edge is a fanout edge of node $v \in V_I$, and $m_{ij} = 0$ otherwise.

Accordingly, to increase the optimization potential of such sequential methods, we should target improving the maximum mean cycle delay during the placement stage. Thus, the synchronous timing-driven placement problem with sequential optimization techniques is formulated as:

$$\begin{aligned}
\mathcal{STDPS}: \text{Minimize} \quad & \sum_{e \in E} \text{WL}_e(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
\text{Subject to} \quad & a_i + D_{ij} - m_{ij}\tau \leq a_j \; \forall (i,j) \in E' \\
& \text{cell spreading constraints}
\end{aligned}$$

3) Asynchronous timing-driven placement ($\mathcal{ATDP}$):

Instead of governing the circuit using global clock signals, an asynchronous circuit only synchronizes neighboring stages through the handshaking signals [26]. Similar to synchronous circuits with sequential optimization techniques, the performance of asynchronous circuits is also bounded by the maximum mean cycle delay, while the difference is that the average-case performance for asynchronous circuits is achieved naturally without needing extra optimization techniques.

Depending on the timing assumptions made by the specific logic implementation style, different types of timing constraints need to be satisfied for asynchronous circuits. Except for delay-insensitive (DI) designs [27] which are premised on the fact that they will function correctly regardless of the delays of the gates and the wires, timing constraints for other asynchronous designs fall into two categories [26].

First are explicit timing constraints in the form of minimum and maximum bounded delay values for gates and wires in the circuit. An example is the bounded-delay asynchronous circuits in [28].

Let U_{ij} be the maximum bounded delay and L_{ij} be the minimum bounded delay between nodes v_i and v_j. Let E_e be the set of node pairs which we need to enforce explicit delay bounds. The explicit timing constraints can be written as:

$$L_{ij} \leq a_j - a_i \leq U_{ij} \quad \forall (i,j) \in E_e \qquad (4)$$

Second are relative timing constraints, referred to as *relative timing* [29], which dictate the relative delay of two paths that stem from a common point of divergence. Example design styles that have relative timing constraints include the quasi-delay-insensitive (QDI) design style, such as WCHB, PCHB and the Multi-Level Domino (MLD) template [26].

For a relative timing constraint from a node v_k and forking into two nodes v_i and v_j, constraints can be written as:

$$|(a_i - a_k) - (a_j - a_k)| \leq I_{ij} \quad \forall (i,j) \in E_r \tag{5}$$

This bounds the maximum difference in time that the signals arrive at the two end-points of the fork. This type of constraint captures the notion of an $isochronic\ fork$ [26], a common type of constraint in quasi-delay-insensitive designs. Here I_{ij} is the delay bound for $isochronic\ fork$. E_r is the set of node pairs which have relative timing constraints.

Combining everything together, the asynchronous timing-driven placement problem, which minimizes both total wire-length and cycle time subject to timing constraints (4), (5) can be formulated directly as:

$$\begin{aligned}
&\mathcal{ATDP}: \\
&\text{Minimize} \quad \sum_{e \in E} \text{WL}_e(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
&\text{Subject to} \quad a_i + D_{ij} - m_{ij}\tau \leq a_j \quad \forall (i,j) \in E' \qquad (6) \\
&\qquad L_{ij} \leq a_j - a_i \leq U_{ij} \quad \forall (i,j) \in E_e \qquad (7) \\
&\qquad |(a_i - a_k) - (a_j - a_k)| \leq I_{ij} \\
&\qquad\qquad \forall (i,j) \in E_r \qquad (8) \\
&\qquad \text{cell spreading constraints}
\end{aligned}$$

where $m_{ij} = 1$ if the corresponding edge is a fanout edge of a token buffer, and $m_{ij} = 0$ otherwise.

C. A General Timing-driven Placement Problem ($\mathcal{GTDP}$)

In this section, we show that all the three different types of problems we presented in Sec. II-B can be generalized to the timing-driven placement problem shown as follows:

$$\begin{aligned}
\mathcal{GTDP}: \text{Minimize} \quad & \sum_{e \in E} \text{WL}_e(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
\text{Subject to} \quad & a_i + \hat{D}_{ij} - \hat{m}_{ij}\tau \leq a_j \quad \forall (i,j) \qquad (9) \\
& \text{cell spreading constraints}
\end{aligned}$$

Here, $\hat{D}_{ij}$ captures the wire delay after we combine everything together. For $\mathcal{STDP}$, $\hat{m}_{ij}$ is always equal to 0. For $\mathcal{STDPS}$ or $\mathcal{ATDP}$, $\hat{m}_{ij} = 1$ when the corresponding edge is a fanout edge of a register or a token buffer, and $\hat{m}_{ij} = 0$ otherwise.

It is obvious that $\mathcal{STDPS}$ is directly equivalent to the formulation of $\mathcal{GTDP}$. Next, we show how $\mathcal{STDP}$ and $\mathcal{ATDP}$ can also be reduced to this form.

1) Transform $\mathcal{STDP}$ to $\mathcal{GTDP}$:

We add two new nodes into the timing graph: v_s and v_t and let their corresponding arrival times to be a_s and a_t. In addition, we add the set of edges (s, v_i) $\forall v_i \in V_I$ and (v_j, t) $\forall v_j \in V_O$ into the timing graph, with the edge delay $D_{sv_i} = W_i$ and $D_{v_jt} = 0$. Finally, we add edge (v_t, v_s) into the timing graph with $D_{v_tv_s} = -\tau$. The new timing graph after this modification is shown in Fig. 1.

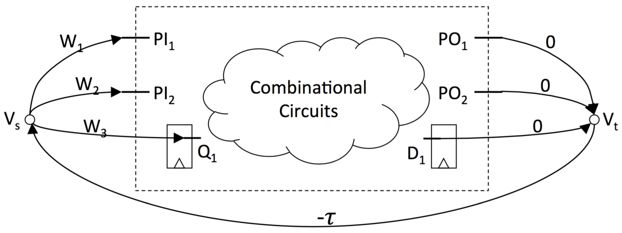


Fig. 1: Modified timing graph for $\mathcal{STDP}$.

After this graph transformation, timing constraints (2) and (3) can be rewritten as follows:

$$a_s + W_i \leq a_i \quad \forall i \in V_I \tag{10}$$

$$a_j + 0 \leq a_t \quad \forall j \in V_O \tag{11}$$

$$a_t - \tau \leq a_s \tag{12}$$

The new constraints (10) (11) (12) can easily fit into the constraints (9) of $\mathcal{GTDP}$. Then, $\mathcal{STDP}$ can be transformed to $\mathcal{GTDP}$ by combining constraints (1) (10) (11) (12) into constraints (9).

2) Transform $\mathcal{ATDP}$ to $\mathcal{GTDP}$:

We can rewrite the timing constraints in Equations (7) and (8) into the same form as Equation (9) as follows:

$$(a_i + L_{ij} \leq a_j) \land (a_j - U_{ij} \leq a_i) \tag{13}$$

$$(a_j - I_{ij} \leq a_i) \land (a_i - I_{ij} \leq a_j) \tag{14}$$

Then, combining Equation (6) with the reformulated Equations (13) and (14), we can easily transform $\mathcal{ATDP}$ to $\mathcal{GTDP}$.

III. Our Proposed Approaches on Solving $\mathcal{GTDP}$

It is difficult to directly solve $\mathcal{GTDP}$, due to its complex timing constraints (9) and cell spreading constraints. In this section, we discuss how we handle the $\mathcal{GTDP}$ problem and propose two different approaches which can solve $\mathcal{GTDP}$ effectively. For both approaches, we use Lagrangian relaxation to relax the timing constraints of $\mathcal{GTDP}$, while the difference is how we handle the spreading constraints. In particular, one approach, as we will present in Section III-A, handles the spreading constraints inside the LR framework during the LR subproblem. We refer this approach as the spreading-inside approach. The other approach, which we will present in Section III-B, handles the cell spreading outside the Lagrangian relaxation framework, while only takes care of the timing constraints within the LR framework. We refer this approach as the spreading-outside approach.

A. Spreading-inside Approach

The spreading-inside approach extends one of our previous works in [7][30]. An overview of the spreading-inside approach is shown in Fig. 2(a). To make things clear, we highlighted the cell spreading step, which is inside the LR framework denoted as the red dotted box. In the beginning, we relax all the timing constraints of $\mathcal{GTDP}$ and initialize a vector of $\boldsymbol{\lambda}$ satisfying the KKT conditions. The relaxed

LR subproblem is denoted as $\mathcal{LRS}$, which still contains the spreading constraints. Next, at each iteration, instead of directly solving $\mathcal{LRS}$, we explore the special structure of $\mathcal{GTDP}$ and solve an equivalent yet simpler version $\mathcal{LRS}$-$\mathcal{S}$ of the subproblem. In particular, $\mathcal{LRS}$-$\mathcal{S}$ is a weighted wirelength minimization problem that can be solved by a standard wirelength-driven placer. Typically, the wirelength-driven placer incorporates the spreading constraints into the objective function of $\mathcal{LRS}$-$\mathcal{S}$ and solve it as an unconstrained optimization problem. After the $\mathcal{LRS}$-$\mathcal{S}$ is solved, we update the vector of $\boldsymbol{\lambda}$ using any standard method[1]. The Lagrangian relaxation loop terminates when there is no improvement in the objective function or the runtime limit is exceeded.

1) Lagrangian Relaxation Subproblem ($\mathcal{LRS}$):

We relax the timing constraints of $\mathcal{GTDP}$ following the Lagrangian relaxation procedure and introduce a nonnegative Lagrange multiplier λ_{ij} for each timing constraint. Let $\boldsymbol{\lambda}$ be a vector of all the Lagrange multipliers.

$$\begin{aligned}\text{Let}\quad \mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{y},\mathbf{a},\tau) &= \sum_{e\in E}\mathrm{WL}_e(\mathbf{x},\mathbf{y}) + \alpha\tau \\ &+ \sum_{\forall(i,j)}\lambda_{ij}(a_i + \hat{D}_{ij} - \hat{m}_{ij}\tau - a_j)\end{aligned}$$

Then the LR subproblem, which gives a lower bound for $\mathcal{GTDP}$ for any $\boldsymbol{\lambda} \geq \mathbf{0}$ [13], can be formulated as:

$$\begin{aligned}\mathcal{LRS}: \quad &\text{Mimimize} \quad \mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{y},\mathbf{a},\tau) \\ &\text{Subject to} \quad \text{cell spreading constraints}\end{aligned}$$

Please note that the cell spreading constraints are not relaxed in our spreading-inside approach.

2) Simplified LR Subproblem ($\mathcal{LRS}$-$\mathcal{S}$):

Inspired by [14], we rearrange the terms here and the Lagrangian function $\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{a},\tau)$ can be rewritten as:

$$\begin{aligned}\mathcal{L} = \sum_{e\in E}\mathrm{WL}_e(\mathbf{x},\mathbf{y}) &+ (\alpha - \sum_{\forall(i,j)}\lambda_{ij}\hat{m}_{ij})\tau \\ &+ \sum_{k\in V}(\sum_{\forall(k,j)}\lambda_{kj} - \sum_{\forall(i,k)}\lambda_{ik})a_k \\ &+ \sum_{\forall(i,j)}\lambda_{ij}\hat{D}_{ij}\end{aligned}$$

Here, we heuristically limit our search of $\boldsymbol{\lambda}$ by the KKT conditions, which imply $\partial\mathcal{L}/\partial a_i = 0$ for $1 \leq i \leq |V|$ and $\partial\mathcal{L}/\partial\tau = 0$ at the optimal solution of the primal problem. Then the optimality conditions $\mathcal{K}$ on $\boldsymbol{\lambda}$ can be obtained as:

$$\begin{aligned}\mathcal{K}: \quad &\alpha = \sum_{\forall(i,j)}\lambda_{ij}\hat{m}_{ij} \\ &\sum_{\forall(k,j)}\lambda_{kj} = \sum_{\forall(i,k)}\lambda_{ik} \quad \forall k \in V\end{aligned}$$

[1]Detailed methods we used to update $\boldsymbol{\lambda}$ is presented in Section IV-C.

Apply the optimality conditions into $\mathcal{LRS}$, we can obtain a simplified Lagrangian relaxation subproblem $\mathcal{LRS}$-$\mathcal{S}$:

$$\begin{aligned}&\mathcal{LRS}\text{-}\mathcal{S}: \\ &\quad\text{Minimize} \quad \mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{y}) = \sum_{e\in E}\mathrm{WL}_e(\mathbf{x},\mathbf{y}) + \sum_{\forall(i,j)}\lambda_{ij}\hat{D}_{ij} \\ &\quad\text{Subject to} \quad \text{cell spreading constraints}\end{aligned}$$

It can be seen that when the given $\boldsymbol{\lambda}$ satisfies the KKT conditions, solving $\mathcal{LRS}$ is equivalent to solving $\mathcal{LRS}$-$\mathcal{S}$.

3) Lagrangian Dual Problem ($\mathcal{LDP}$):

Let the function $q(\boldsymbol{\lambda})$ be the optimal value of the problem $\mathcal{LRS}$. We are interested in finding the values for the Lagrange multipliers $\boldsymbol{\lambda}$ to give the maximum lower bound of $\mathcal{GTDP}$. This problem is called the Lagrangian dual problem and is defined as follows:

$$\begin{aligned}\mathcal{LDP}: \quad &\text{Maximize} \quad q(\boldsymbol{\lambda}) \\ &\text{Subject to} \quad \text{the optimality conditions } \mathcal{K} \text{ on } \boldsymbol{\lambda}\end{aligned}$$

Solving $\mathcal{LDP}$ will provide a solution to the primal problem.

4) Solving $\mathcal{LRS}$-$\mathcal{S}$:

The detailed timing model is irrelevant to our problem formulation and transformation presented in the previous sections, but it is required when we start to discuss how to solve these problems. Therefore, in this subsection, we first present the timing model used in this paper.

Since detailed placement and routing are not performed yet, it will be wasteful and time consuming to use an accurate delay model during the global placement stage. Thus, as an approximation, we use a linear delay model which sets the wire delay $\hat{D}_{ij}$ to be proportional to the wirelength of the hyperedge e associated with nodes i and j:

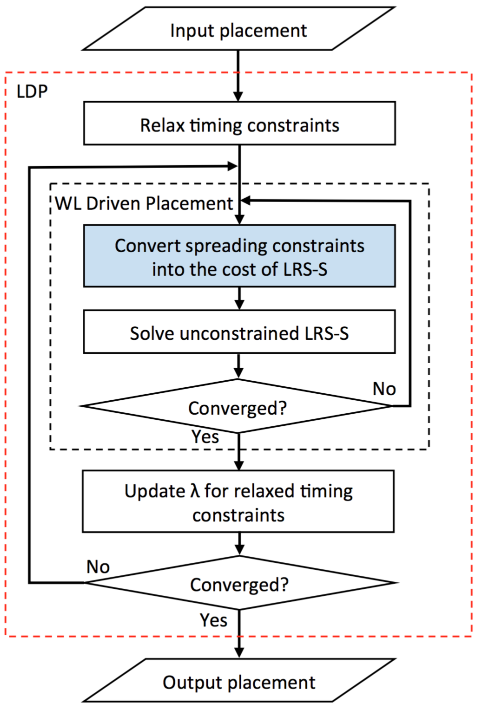
$$\hat{D}_{ij} = d_i + WL_e(\mathbf{x},\mathbf{y}) \cdot \gamma_e \tag{15}$$

where d_i is the intrinsic gate delay and $WL_e(\mathbf{x},\mathbf{y}) \cdot \gamma_e$ is the total wire load delay. γ_e is a constant value associated with each edge and depends on the driver cell, load cells and electrical characterization for the wires. More details about the derivation of d_i and γ_e is presented in Section IV-A.

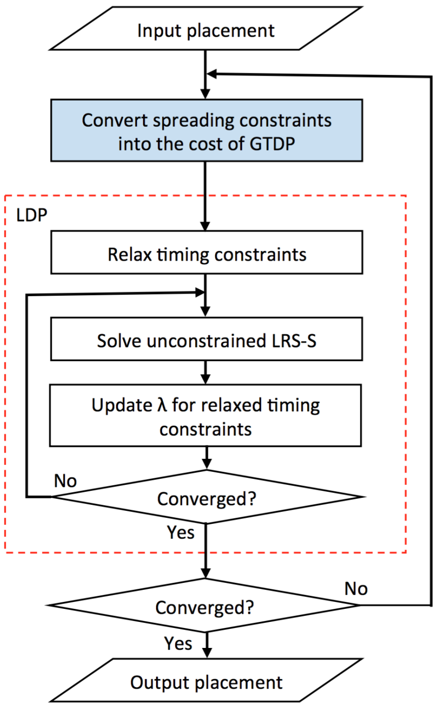
Based on the linear delay model proposed above, $\mathcal{LRS}$-$\mathcal{S}$ can be written as:

$$\begin{aligned}&\text{Minimize} \quad \mathcal{L}_{\boldsymbol{\lambda}}(\mathbf{x},\mathbf{y}) \\ &\quad = \sum_{e\in E}\mathrm{WL}_e(\mathbf{x},\mathbf{y}) + \sum_{\forall(i,j)}\lambda_{ij}(d_i + WL_e(\mathbf{x},\mathbf{y}) \cdot \gamma_e) \\ &\qquad + \text{terms independent of } \mathbf{x}, \mathbf{y} \\ &\quad = \sum_{e\in E}\mathrm{WL}_e(\mathbf{x},\mathbf{y}) + \sum_{\forall(i,j)}WL_e(\mathbf{x},\mathbf{y}) \cdot \lambda_{ij}\gamma_e \\ &\qquad + \text{terms independent of } \mathbf{x}, \mathbf{y} \\ &\text{Subject to} \quad \text{cell spreading constraints}\end{aligned}$$

The new objective function only contains **x**, **y** as variables. Therefore, $\mathcal{LRS}$-$\mathcal{S}$ becomes a weighted wirelength minimization problem for a set of hyperedges, which can be solved well by existing wirelength-driven placement engine with the ability to handle net weights.


Input placement
LDP
Relax timing constraints
WL Driven Placement
Convert spreading constraints into the cost of LRS-S
Solve unconstrained LRS-S
Converged?
No
Yes
Update λ for relaxed timing constraints
Converged?
No
Yes
Output placement

(a) The spreading-inside (SI) approach.


Input placement
Convert spreading constraints into the cost of GTDP
LDP
Relax timing constraints
Solve unconstrained LRS-S
Update λ for relaxed timing constraints
No
Converged?
Yes
Converged?
No
Yes
Output placement

(b) The spreading-outside (SO) approach.

Fig. 2: The flow chart of two proposed approaches.

5) Solving $\mathcal{LDP}$:

In general, $\mathcal{LDP}$ can be solved by solving a sequence of $\mathcal{LRS}$-$\mathcal{S}$. Many previous works use or modify the subgradient optimization method to solve $\mathcal{LDP}$ (e.g., [14][15]). The basic idea of the subgradient optimization method is straightforward. At each iteration, we first update $\boldsymbol{\lambda}$ based on the criticality of all timing edges. Then, based on the updated $\boldsymbol{\lambda}$, we solve $\mathcal{LRS}$-$\mathcal{S}$ again to generate a new placement. Besides the subgradient optimization method, we can also use the direction finding approach [31], which has been shown to have better convergence in practice.

B. Spreading-outside Approach

The benefit of the spreading-inside approach is that one can leverage an existing wirelength-driven placer as a black box to solve $\mathcal{GTDP}$ without any modification. However, cell spreading constraints are non-convex by nature. Besides, they are usually modeled by non-continuous and non-differentiable functions in modern placers [18][19]. Thus, this approach cannot guarantee that an optimal solution of the $\mathcal{LDP}$ is also optimal for the primal problem. To avoid this issue, we propose another approach to solve $\mathcal{GTDP}$, which we referred to as the spreading-outside approach.

An overview of the spreading-outside approach is shown in Fig. 2(b). As implied by the name, we handle the cell spreading constraints outside the LR framework. In the beginning, similar to typical wirelength-driven placement algorithms, we convert the cell spreading constraints into a cost which is incorporated into the objective function of the placement problem. Different from wirelength-driven placement, the resulting problem still has the timing constraints instead of being unconstrained. Here, we leverage the LR framework to solve this problem, since LR has shown to be very effective in handling the timing constraints. After the LR loop converges, we will update and convert the cell spreading constraints again if needed.

We neglect the derivation of the LR subproblem and Lagrangian dual problem for the spreading-outside approach, as it is similar to what we have presented in Section III-A, except we do not have the cell spreading constraints this time. In particular, for this approach, the $\mathcal{LRS}$-$\mathcal{S}$ will just be an unconstrained optimization problem and can be easily solved using any standard methods.

By tackling the complex and non-differentiable cell spreading constraints outside the LR framework, LR only needs to handle a problem with timing constraints. The timing constraints can be relaxed and converted into terms linear to wirelength in the objective function of $\mathcal{LRS}$-$\mathcal{S}$. Thus, better solutions can be expected at the LR step. In particular, if wirelength is modeled as convex function and the spreading constraints are converted into convex functions, the problem to be solved by LR is a convex optimization problem. Then, the strong duality will hold and this convex problem can be solved optimally using the LR framework. The disadvantage of the spreading-outside approach is that existing weighted wirelength minimization placers will no longer be directly applicable. Instead, we need to either implement the cell spreading step by ourselves or detach the cell spreading part from the wirelength-driven placer in order to use it.

C. Comparing our approaches with previous Lagrangian relaxation based $\mathcal{TDP}$ algorithms

In this subsection, we discuss in details about the differences between our approaches and several previous works.

In [15], the proposed LR framework relaxes the cell spreading constraints together with the timing constraints. However, this framework only works for placers with explicit modeling of spreading constraints, i.e., GORDIAN with center-of-gravity constraints [17]. For some state-of-the-art placers, this framework might not work, since the spreading constraints are often handled implicitly using heuristic algorithms [18][19]. In [16], the spreading constraints are not relaxed, while the path based approach makes the proposed framework not applicable for large-scale circuits. In [10], Lagrangian relaxation is only used as a refinement step after global placement, and the COG based cell spreading constraints are not updated. Therefore, the effectiveness of the proposed approach is greatly limited. In addition, some of the previous works did not simplify the LR subproblems through exploring the special structure of the circuit graph. Thus, extra effort is required to calculate the cycle time and arrival time variables. In addition, more iterations are required to search for the optimal $\boldsymbol{\lambda}$ in $\mathcal{LDP}$ as $\boldsymbol{\lambda}$ is not limited by the optimality conditions $\mathcal{K}$.

Different from previous works, both the spreading-inside approach and spreading-outside approach proposed by us simplify the $\mathcal{LRS}$ using KKT conditions based on the special structure of the circuit. In addition, our approaches do not relax the cell spreading constraints by LR. Thus they are more compatible with various type of placement techniques, especially for modern placers with implicit modeling of cell spreading constraints. Finally, our approaches can be used to optimize either the critical paths or the critical cycles of the circuit, and hence are suitable for different circuit design styles.

IV. Detailed Implementation

In this section, we talk about the detailed implementation of the timing-driven placement approaches proposed in Section III. In particular, our TDP tool, which is referred to as TD-POLAR here, incorporates the proposed timing-driven placement approaches with the state-of-the-art quadratic placer POLAR [19]. In Section IV-A, we first discuss more details about our linear delay model. Next, in Section IV-B, we discuss the quadratic placement and rough legalization techniques which are the core techniques used in the POLAR algorithm. Finally, in Section IV-C, we will discuss how we leverage POLAR to solve the $\mathcal{GTDP}$ problem using the proposed approaches.

A. Linear Delay Model

The delay of a net is considered as two parts: cell delay and interconnect delay. For the cell delay, it is typically defined using two dimensional lookup tables, as a function of input slew and output capacitance with a linear interpolation between the data points. This function is non-differentiable and therefore not satisfying the requirement of our Lagrangian relaxation framework. Fig. 3 shows an example about how we perform a linear approximation for the delay function. The x-axis denotes the output capacitance (C_l) and the y-axis denotes the cell delay. In addition, each curve corresponds to a delay function with different input slew value. First, we choose a typical slew value (i.e. the blue curve), while ignoring the other possible slew values (i.e. the gray curves). Next, we use a linear function $\mu \cdot C_l + \phi$ to better approximate the middle portion of the delay function, where delay value normally falls into. Here, μ and ϕ denotes the slope and y-intercept of the linear function separately.

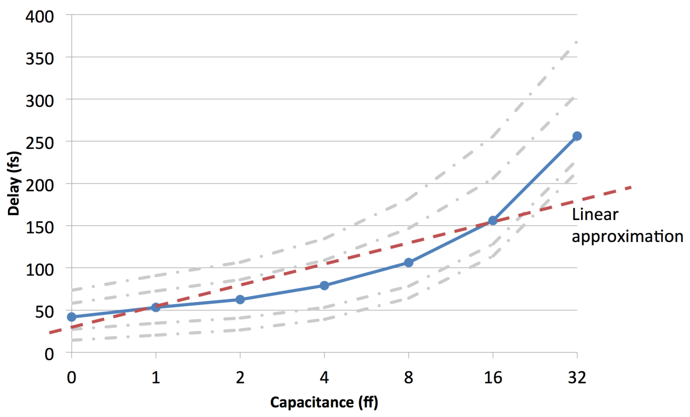


Fig. 3: Linear approximation of the delay function.

The interconnect delay depends on the wire resistance and wire capacitance. We ignore the wire resistance and calculate the wire capacitance using: $C_{wire} = WL_e(\mathbf{x}, \mathbf{y}) \cdot C_u$. Here, $WL_e(\mathbf{x}, \mathbf{y})$ is the HPWL of the net and C_u is the capacitance per unit wirelength.

Combining everything together, we have the delay written as Equation 15, with $\gamma_e = \mu \cdot C_u$ and $d_i = \mu \cdot C_{pin} + \phi$. Here, C_{pin} denotes the capacitance of the output pin of the cell.

B. POLAR: a wirelength-driven placer based on quadratic and rough legalization techniques

1) Quadratic Placement:

Assuming all the nets $e \in E$ are two pin nets. The wirelength for a particular net e can be modeled using the HPWL, which is given by the Manhattan distance between the two cells connected by e. Let $\mathbf{x}, \mathbf{y}$ denote the cell coordinates similar to what we have defined in Section II. Then, the total wirelength can be calculated by the total sum of HPWL for all the nets:

$$\text{HPWL}(\mathbf{x}, \mathbf{y}) = \sum_{e \in E} [\max_{i \in e} x_i - \min_{i \in e} x_i + \max_{i \in e} y_i - \min_{i \in e} y_i]$$

The function $\text{HPWL}(\mathbf{x}, \mathbf{y})$ is convex, but it is not differentiable. To make the optimization easier, the quadratic technique approximates the Manhattan distance of the two pin net by the squared Euclidean distance, also known as quadratic wirelength. Let Q_x and Q_y be the connection matrices. The objective of the wirelength-driven placement can be defined as:

$$\text{Minimize} \quad \phi = \frac{1}{2}\mathbf{x}^T Q_x \mathbf{x} + \mathbf{c}_x^T \mathbf{x} + \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} + \mathbf{c}_y^T \mathbf{y} + const$$

It can be proved that both Q_x and Q_y are symmetric positive definite matrices. Thus, ϕ is convex and differentiable, and the minimum solution of ϕ can be found by setting its derivatives to 0 and solving the resulting system of linear equations:

$$Q_x \mathbf{x} + \mathbf{c}_x + Q_y \mathbf{y} + \mathbf{c}_y = 0 \tag{16}$$

For more details about the definition of $\mathbf{c}_x, \mathbf{c}_y$ and the derivation of Equation 16, we refer the reader to Chapter 11 of [1].

2) Rough Legalization:

If we consider minimizing ϕ alone, the cells will not be spread out and the placement solution will not be legalizable. Therefore, extra techniques are required to avoid excessive cell overlapping.

POLAR adopts the rough legalization (RL) [18] approach to reduce the cell overlapping. At each placement iteration, RL quickly spreads out the cells and generates an almost legal placement, as shown in Fig. 4(b). The roughly legalized placement is used to generate the spreading forces, which are incorporated into the objective function of the wirelength-driven placement problem and guide the quadratic placement on the next iteration, as shown in Fig. 4(c).

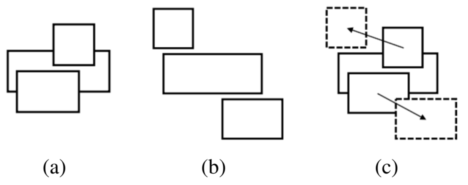


Fig. 4: (a) Cell overlaps after quadratic placement. (b) An almost legal placement obtained by rough legalization. (c) Use of the roughly legal placement to guide the spreading force generation.

C. TD-POLAR: a general timing-driven placement tool

For both the spreading-inside and the spreading-outside approaches, we use the direction finding approach inspired by [31] to solve $\mathcal{LDP}$. We are not using the subgradient optimization method as it requires a projection of $\boldsymbol{\lambda}$ unto the optimality conditions $\mathcal{K}$ after each iteration to maintain $\boldsymbol{\lambda}$ within the feasible region of $\mathcal{LDP}$. For $\mathcal{STDP}$ problem, projection can be done by simply traversing the circuit in topological order since the corresponding graph of the circuit is a directed acyclic graph. For $\mathcal{STDPS}$ or $\mathcal{ATDP}$, it will not be easy to redistribute $\boldsymbol{\lambda}$ since the corresponding circuit structure contains loops.

In particular, the direction finding approach wants to find an improving feasible direction $\Delta\boldsymbol{\lambda}$ and a step size β such that at each step we have:

$$q(\boldsymbol{\lambda} + \beta\Delta\boldsymbol{\lambda}) > q(\boldsymbol{\lambda})$$

The improving feasible direction $\Delta\boldsymbol{\lambda}$ can be found by solving the following linear program:

$$\begin{aligned} \mathcal{DF}: \quad & \text{Maximize} \quad \sum_{\forall(i,j)} \Delta\lambda_{ij}\hat{D}_{ij} \\ & \text{Subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0},\ \boldsymbol{\lambda} \in \mathcal{K} \\ & \qquad\qquad\quad \max(-u, -\lambda_{ij}) \leq \Delta\lambda_{ij} \leq u \end{aligned}$$

where u is used to bound the objective function from going to infinity, similar to [31].

1) Implementation of the spreading-inside approach:

It is straightforward to incorporate POLAR with our timing-driven placement flow using the spreading-inside approach. Since the $\mathcal{LRS}$-$\mathcal{S}$ is a weighted wirelength optimization problem with spreading constraints, we can directly call POLAR to solve it. To capture the timing of the circuit, we add an extra pseudo two-pin net for each timing edge to the circuit. After the $\boldsymbol{\lambda}$ update step, the weights of the added two-pin nets should be updated accordingly, while the weights of original nets are kept the same.

2) Implementation of the spreading-outside approach:

The implementation of the spreading-outside approach requires a tighter integration with the placement engine. We split POLAR into the quadratic placement step, which we have presented in Section IV-B 1), and the rough legalization step, which is done by a heuristic algorithm. Then, for the spreading-outside approach as shown in Fig. 2(b), the step of solving the unconstrained $\mathcal{LRS}$-$\mathcal{S}$ will be similar to the quadratic placement step of POLAR, except now there are weights associated with nets given by the Lagrange multipliers. In addition, the step of converting spreading constraints into the cost function will be replaced by the rough legalization step of POLAR. Therefore, for the spreading-outside approach, we first perform the rough legalization to generate the spreading forces for the current placement. The spreading forces are then incorporated into the objective function of $\mathcal{GTDP}$ to guide the placement process in the quadratic placement step. Next, we apply Lagrangian relaxation framework on the quadratic placement step to handle the timing constraints of $\mathcal{GTDP}$. The iteration continues until there is no improvement in the objective function or we exceed the runtime limit.

D. Timing-driven Detailed Placement

Since traditional detailed placement algorithms only target reducing the total chip wirelength, timing degradation might happen if we directly use them to optimize the global placement results generated by TD-POLAR. In order to minimize the disturbance on circuit timing, we developed a timing-driven detailed placement step to further optimize the global placement results and also help generating a legalized placement. In particular, we leveraged the existing wirelength-driven detailed placement engine FastDP [32] and applied net weights into its cost function. The pseudo nets to capture timing at global placement are kept. The net weights we used for the pseudo nets are the same as those at the final round of the global placement stage. Thus, it reflects the timing criticality for each net. By doing this, FastDP is able to respect the timing criticality at the global placement stage during its optimization process and the disturbance on timing is greatly reduced.

V. Experiments

The proposed approach is implemented using C++. All experiments were run on a Liunx PC with 47GB of memory and Intel Core-i3 3.3GHz CPU.

We first demonstrate our approaches using asynchronous circuits since it is the most general circuit design style among the three design styles which we have introduced in Section II. In particular, the asynchronous circuits we used are based on the PCHB template [33], which is a QDI template designed with dual-rail asynchronous channels and 1-of-N handshaking protocol [26]. Fig. 5 shows a three-stage PCHB pipeline structure with control circuit (CTRL), C-element (C) and domino logic (LOGIC) for computation.

Marked lines in Fig. 5 show an example of timing assumptions made by PCHB. It requires the input to the domino block go low before a rising transition on the control signal 'en' occurs. This timing assumption is a relaxed interpretation of the $isochronic\ fork$ assumption [34] and can easily be met without special care. We ignore this timing constraint at global placement stage and leave it to be checked after detailed placement and routing, similar to [20] and [35].

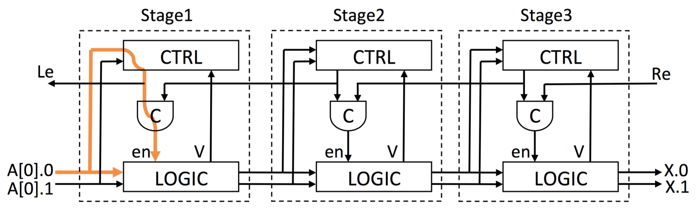


Fig. 5: PCHB pipeline template.

We are using the Proteus standard cell library [20] which is based on an implementation of the PCHB template. We run TD-POLAR on two sets of benchmarks. First is a set of ISCAS89 benchmark circuits which are converted to unconditional asynchronous circuits using the front-end synthesis flow of Proteus. In particular, flip-flops from ISCAS89 are mapped to token buffers and combinational gates are mapped to logic cells in PCHB cell library. The second set consists of several benchmarks synthesized from RTL to netlist using Proteus. For ALU and Accumulator (ACC) design, we choose different bit width for the datapath to create a set of benchmarks with different number of cells.

The statistics of our benchmark circuits are shown in Table II. The column "# of vertices" shows the total number of cells in each design. The column "# of edges" shows the total number of edges, which includes the original nets of the circuit and the added two-pin timing edges. An estimation of the total number of variables for the corresponding $\mathcal{GTDP}$ problem is reported in column "# of vars".

We compare the two approaches implemented in TD-POLAR with a non-timing-driven placement flow and the

Table II. Statistics of asynchronous circuits

Design	# of vertices	# of edges	# of vars
s444	256	2719	8730
s510	519	5535	17676
s526	307	3366	10792
s526a	297	3284	10520
s641	636	5346	17328
s713	584	4904	15866
s820	681	6952	22268
s832	706	7327	23448
s838	707	7300	23466
s953	931	9805	31422
s1488	1314	15000	47950
s1423	1119	13010	41592
s9234	2108	22118	71058
s13207	5658	56164	181288
s38417	15447	182865	584402
ALU4	413	4239	13666
ALU8	916	10140	32550
ACC32	1187	11605	37252
ACC64	3355	32741	105706
GCD	1505	4901	15664
FU	5304	52023	167212

timing-driven commercial asynchronous optimization flow Proteus. For the non-timing-driven placement flow, we use the industrial placer Encounter in the default mode to place the design without setting any input timing constraint.

The Proteus flow performs both the global and detailed placement on the input circuits through leveraging synchronous placement tools. In particular, the Proteus flow breaks the timing loops according to the PCHB template and add explicit timing constraints on each path segments to improve the timing. To avoid the changing of input netlist, we also disable the gate resizing step during the placement stage of Proteus flow.

For our approaches, we first perform pure wirelength-driven placement on the asynchronous circuit by running POLAR in pure wirelength-driven mode, as a good starting placement with minimized wirelength is necessary in order to achieve better cycletime in later stage. Next, we perform the timing-driven placement using TD-POLAR at the global placement stage. At the detailed placement stage, we use the modified FastDP [32] presented in Section IV-D as our detailed placement engine. Finally, the placement results are exported to Encounter to perform routing.

The comparison results for asynchronous circuits are shown in Table III. The target density is set to be 0.5 for all the flows. Also, we normalized the total wirelength and cycle time in the objective function using the initial wirelength and cycle time. Thus, we set the parameter $\alpha = 1$ in the objective function to obtain a balanced effort on optimizing wirelength and cycle time. The "Routed wirelength" column shows the final detailed routed wirelength reported by Encounter for all flows. The "Cycletime" column shows the cycletime calculated based on our delay model. The "Encounter" column denotes the non-timing-driven placement performed by Encounter. The "SI" column denotes the spreading-inside approach. The "SO" column denotes the spreading-outside approach. Regarding the total routed wirelength, as expected, all the flows which

Table III. Comparison on non-timing-driven placement flow and commercial timing-driven placement flow

Design	Routed Wirelength x 10^6 (nm)				Cycletime (ns)				Runtime (s)			
	Encounter	Proteus	SI	SO	Encounter	Proteus	SI	SO	Encounter	Proteus	SI	SO
s444	10.46	11.32	10.35	10.60	6.06	5.72	4.22	4.22	8	245	8	16
s510	32.84	35.38	34.06	32.21	7.25	7.01	5.98	6.31	12	330	15	29
s526	14.24	14.85	13.14	12.92	4.20	4.83	3.37	3.60	8	257	8	18
s526a	12.22	13.79	12.87	13.04	4.89	3.75	3.69	3.30	8	249	8	18
s641	22.02	26.96	23.70	23.86	5.00	4.88	4.39	4.33	10	313	12	25
s713	21.06	24.38	21.78	20.87	6.15	5.71	5.87	5.18	9	228	11	23
s820	43.39	48.53	45.83	44.91	10.40	8.34	6.33	6.22	13	431	17	34
s832	45.84	53.04	45.89	46.48	6.82	7.32	5.79	5.81	15	465	18	37
s838	33.32	37.49	34.73	33.84	5.91	5.58	5.46	4.82	13	337	17	36
s953	61.84	71.85	64.11	63.77	6.12	7.10	7.51	5.63	18	576	24	48
s1488	130.13	137.06	129.38	127.53	12.79	11.35	10.58	10.64	27	771	42	78
s1423	64.48	71.25	61.68	63.59	14.85	8.37	6.91	7.13	20	692	34	64
s9234	119.40	134.48	117.46	120.20	9.83	8.19	7.22	6.58	31	517	56	106
s13207	338.18	386.13	341.98	328.83	13.72	11.86	12.01	10.66	67	1202	156	301
s38417	1253.42	1208.16	1267.24	1245.69	80.43	68.30	45.24	42.66	283	1050	615	997
ALU4	18.01	18.78	23.07	20.99	5.68	5.99	4.38	4.07	10	261	6	22
ALU8	55.03	53.89	76.26	71.44	8.43	5.11	4.65	4.13	16	470	14	52
acc32	49.09	59.87	59.46	58.13	6.45	5.57	4.39	5.10	17	528	15	54
acc64	138.06	145.88	144.53	146.51	10.01	7.37	5.55	5.88	39	757	48	154
GCD	23.32	24.17	27.37	26.50	20.05	20.68	16.35	17.32	21	604	7	16
FU	396.16	453.41	447.56	445.41	16.06	7.81	9.94	8.60	62	958	98	315
Average	1.000	1.051	1.042	1.026	1.000	0.846	0.689	0.660	1.000	15.900	1.739	3.456

Table IV. Comparison on ICCAD 2015 incremental timing-driven placement benchmarks

Design	# of vertices	# of edges	Wirelength x 10^6 (um)			Cycletime (ns)			Runtime (m)	
			INIT	SI	SO	INIT	SI	SO	SI	SO
superblue1	1216K	2544K	111.10	106.20	105.40	50.45	50.69	41.67	82.42	250.66
superblue3	1219K	2671K	119.50	115.90	115.30	56.27	55.03	54.41	169.73	146.87
superblue4	802K	1686K	70.94	68.97	69.25	30.06	27.88	24.44	50.83	78.95
superblue5	1091K	2140K	122.80	117.60	118.40	52.63	44.68	46.07	70.17	131.98
superblue7	1938K	4424K	151.90	143.70	144.90	27.21	27.27	21.20	150.57	305.93
superblue10	1888K	3650K	219.90	211.10	210.10	48.85	47.42	46.26	142.31	254.80
superblue16	986K	2006K	93.87	91.14	91.60	23.63	19.05	21.31	70.34	330.69
superblue18	772K	1782K	62.31	60.21	60.19	23.53	21.65	23.29	53.18	86.32
		Average	1.000	0.961	0.961	1.000	0.939	0.891	2.801	5.627

Table V. Runtime breakdown of the SO approach on ICCAD 2015 incremental timing-driven placement benchmarks (min)

Design	WL driven	Timing-driven Placement			TDP total	Total
		PCG	RL	Others		
superblue1	48.47	150.88	50.72	0.57	202.18	250.66
superblue3	26.14	84.15	36.12	0.46	120.72	146.87
superblue4	14.87	49.57	14.20	0.30	64.07	78.95
superblue5	23.52	63.72	44.39	0.35	108.45	131.98
superblue7	57.34	180.42	66.90	1.25	248.57	305.93
superblue10	41.22	155.82	56.97	0.74	213.53	254.80
superblue16	68.39	193.97	67.55	0.77	262.30	330.69
superblue18	14.47	54.82	16.71	0.30	71.83	86.32
Average	0.186	0.588	0.223	0.003	0.814	1.000

perform timing optimization of the circuit have a higher total wirelength than the non-timing-driven flow Encounter. Among the timing-driven placement flows, both the spreading-inside and spreading-outside approach can achieve a shorter total wirelength compared with the Proteus flow, while the spreading-outside approach achieves the smallest wirelength. Regarding the cycletime, the timing-driven placement flows can achieve much better cycletime than the non-timing-driven Encounter flow. In particular, the Proteus flow is 15.4% better than the Encounter flow, while the spreading-inside approach and spreading-outside approach is 31.1% better and 34% better than the Encounter flow respectively. This shows the importance of timing-driven placement on optimizing the timing of the circuits. It also shows our proposed approaches are more effective in improving the timing of asynchronous circuits than the Proteus flow. In particular, this is because our approaches consider all timing loops globally, while Proteus flow can only focus on some cycles by breaking the loops and leveraging synchronous EDA tools. Also, since timing loops are broken in the Proteus flow, time borrowing for neighboring timing paths along a cycle is not possible, but it is allowed in our approaches. In addition, on average, the spreading-outside approach achieves a shorter wirelength and a smaller cycletime compared with the spreading-inside approach. We

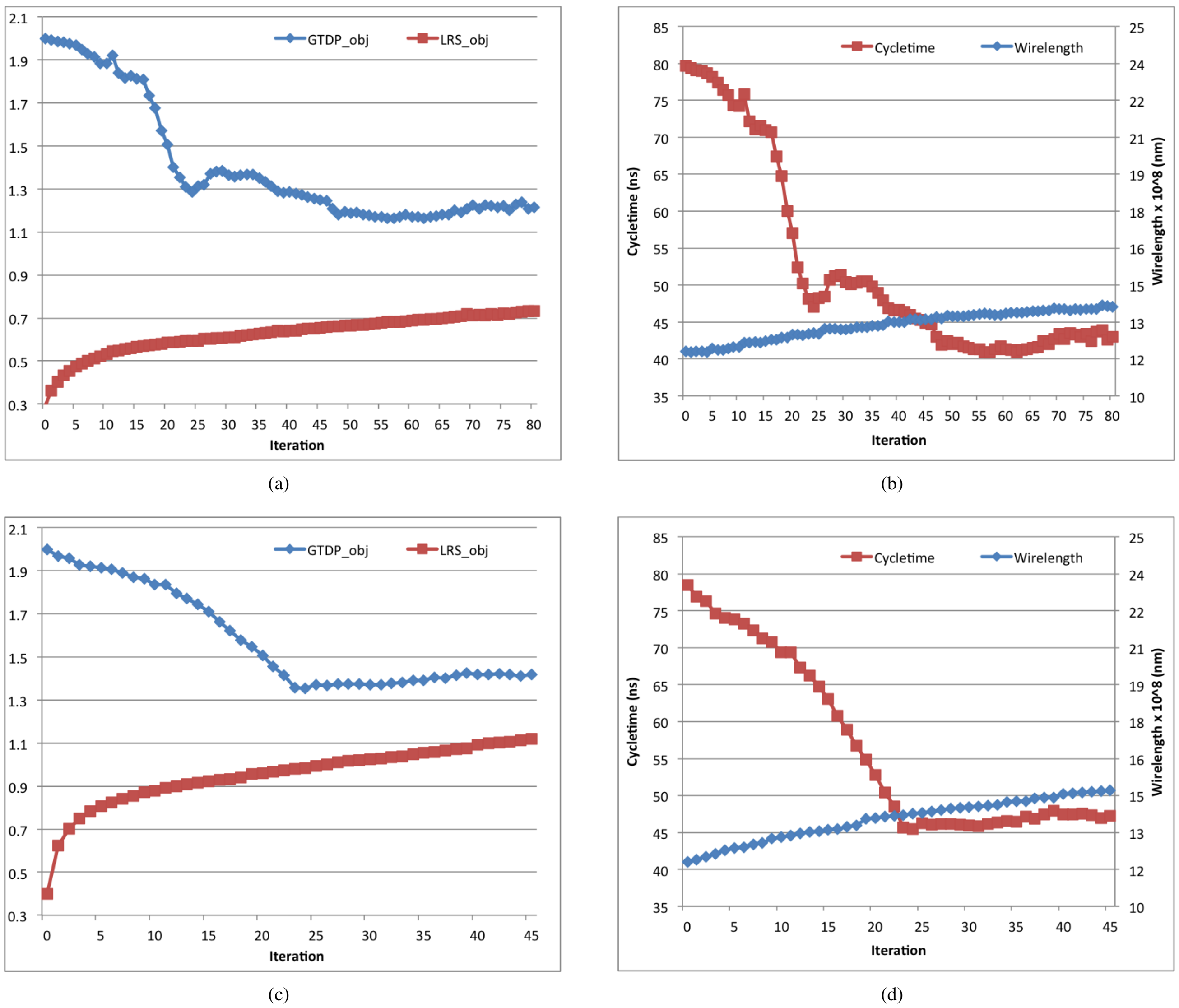


Fig. 6: (a) The convergence of s38417 by SI. (b) The wirelength and cycltime trend of s38417 by SI. (c) The convergence of s38417 by SO. (d) The wirelength and cycltime trend of s38417 by SO.

believe this is because the spreading-outside approach uses LR to handle a problem with timing constraints only, while the cell spreading constraints are handled separately outside the LR loop, and hence better solutions can be expected. Regarding the runtime, the Encounter flow has the shortest runtime, since it does not perform any optimization on circuit timing. The Proteus flow has the longest runtime, due to its added explicit timing constraints which can be exponential to the circuit size. Both our approaches are shown to be much faster and scalable in comparison with the Proteus flow. In particular, the spreading-inside approach is about 2X faster than the spreading-outside approach. This is because the spreading-inside approach converges faster and can be stopped earlier.

The convergence sequences of our largest circuit s38417 using the spreading-inside approach and the spreading-outside approach are shown in Fig. 6(a) and (c) respectively, where the blue line denotes the objective value of the $\mathcal{GTDP}$ and the red line denotes the objective value of the $\mathcal{LRS}$-$\mathcal{S}$. The corresponding changes of cycle time and total chip wirelength at each iteration is shown in Fig. 6(b) and (d) respectively, where the red line denotes the cycletime and the blue line denotes the wirelength. It can be seen that both approaches are very effective in reducing the cycletime of the circuit. The first iteration in Fig. 6 starts at a pure wirelength-driven solution. In addition, each iteration of the spreading-inside approach performs a full run of one iteration of POLAR, which only includes one step of LR and one step of rough legalization. Thus, even though the total number of iterations for the spreading-inside approach is larger than that of the spreading-outside approach in Fig. 6, it is actually stopped earlier. However, as shown in the figure, the spreading-outside approach converges smoother than the spreading-inside

Table VI. Comparison on DP algorithms using spreading-inside approach

Design	Routed Wirelength x 10^6 (nm)			Cycletime (ns)			Runtime (s)		
	FastDP	TD-FastDP	Legalize	FastDP	TD-FastDP	Legalize	FastDP	TD-FastDP	Legalize
s444	10.27	10.35	10.58	4.46	4.22	4.67	0.07	0.12	2.86
s510	31.99	34.06	33.10	6.67	5.98	6.87	0.13	0.17	3.13
s526	13.00	13.14	13.30	5.96	3.37	3.96	0.11	0.14	2.94
s526a	12.83	12.87	13.34	4.19	3.69	4.14	0.10	0.12	3.00
s641	22.79	23.70	22.83	4.33	4.39	4.41	0.15	0.20	3.39
s713	20.65	21.78	21.42	7.25	5.87	5.72	0.11	0.19	3.20
s820	43.60	45.83	44.30	6.73	6.33	7.10	0.14	0.23	3.11
s832	45.25	45.89	46.15	5.77	5.79	6.19	0.14	0.23	3.36
s838	34.24	34.73	38.65	5.25	5.46	4.88	0.13	0.23	3.26
s953	61.88	64.11	64.01	5.94	7.51	6.90	0.17	0.34	3.43
s1488	126.06	129.38	124.88	11.18	10.58	11.80	0.26	0.41	3.52
s1423	60.02	61.68	63.21	7.84	6.91	8.47	0.43	0.36	3.29
s9234	113.93	117.46	120.19	6.65	7.22	6.76	0.35	0.67	3.65
s13207	332.25	341.98	381.02	12.39	12.01	11.17	0.92	1.68	7.03
s38417	1228.40	1267.24	1310.20	59.35	45.24	42.44	3.59	7.90	7.37
ALU4	22.04	23.07	16.92	8.15	4.38	5.22	0.12	0.14	2.94
ALU8	74.17	76.26	50.31	3.64	4.65	4.44	0.19	0.27	3.17
ACC32	58.53	59.46	65.50	6.28	4.39	5.46	0.23	0.37	3.22
ACC64	141.41	144.53	161.67	6.04	5.55	5.15	0.55	0.83	4.52
GCD	26.31	27.37	23.55	17.06	16.35	18.63	0.11	0.15	3.14
FU	438.13	447.56	435.78	10.49	9.94	9.31	0.85	1.63	4.82
Average	1.000	1.029	1.049	1.000	0.875	0.893	1.000	1.850	8.850

Table VII. Comparison on DP algorithms using spreading-outside approach

Design	Routed Wirelength x 10^6 (nm)			Cycletime (ns)			Runtime (s)		
	FastDP	TD-FastDP	Legalize	FastDP	TD-FastDP	Legalize	FastDP	TD-FastDP	Legalize
s444	10.04	10.60	12.54	4.17	4.22	4.45	0.09	0.14	2.80
s510	30.96	32.21	35.78	5.92	6.31	7.07	0.12	0.19	3.05
s526	12.99	12.92	15.24	3.65	3.60	3.86	0.09	0.12	3.33
s526a	12.61	13.04	14.86	3.38	3.30	3.17	0.09	0.11	3.27
s641	22.45	23.86	25.55	4.60	4.33	4.60	0.13	0.16	3.19
s713	20.53	20.87	23.01	7.35	5.18	5.47	0.12	0.18	3.20
s820	43.70	44.91	51.25	6.52	6.22	7.15	0.14	0.22	3.12
s832	44.52	46.48	53.97	8.03	5.81	6.19	0.18	0.50	2.96
s838	33.81	33.84	40.62	5.31	4.82	4.99	0.17	0.74	3.01
s953	61.21	63.77	74.16	5.52	5.63	6.41	0.23	0.29	3.31
s1488	125.67	127.53	147.69	11.71	10.64	12.08	0.24	0.43	3.26
s1423	60.68	63.59	77.57	7.05	7.13	8.88	0.22	0.40	3.24
s9234	116.52	120.20	135.68	6.83	6.58	6.99	0.37	0.54	3.53
s13207	320.50	328.83	387.80	11.72	10.66	12.85	1.03	1.64	5.55
s38417	1201.03	1245.69	1509.41	63.77	42.66	45.38	3.70	9.25	7.14
ALU4	20.37	20.99	22.70	6.02	4.07	3.48	0.10	0.14	3.17
ALU8	70.43	71.44	76.23	3.46	4.13	6.63	0.39	0.31	3.44
ACC32	56.41	58.13	65.11	5.67	5.10	5.50	0.21	0.37	3.38
ACC64	144.50	146.51	162.78	6.61	5.88	5.90	0.53	1.04	4.11
GCD	24.87	26.50	29.88	10.36	17.32	22.95	0.38	0.19	3.04
FU	435.34	445.41	492.34	18.12	8.60	11.19	0.95	1.62	4.96
Average	1.000	1.023	1.131	1.000	0.475	0.618	1.000	1.957	8.013

approach, due to its more fine-grained optimization at each step. Therefore, after the detailed placement is performed, the spreading-outside approach is able to achieve better results.

Next, we compare different detailed placement techniques in Table VI and VII. For Table VI, the detailed placement are performed on the global placement results generated by the spreading-inside approach. For Table VII, the detailed placement are performed on the global placement results generated by the spreading-outside approach. For both tables, the column "FastDP" denotes the flow where we directly use FastDP to perform wirelength-driven detailed placement without adding weights. The column "TD-FastDP" denotes the timing-driven detailed placement, where we add the weight from the global placement stage into FastDP. The column "Legalize" denotes the detailed placement flow which only performs the legalization of the global placement results using Encounter.

From the experimental results in Table VI and VII, we can see that directly applying FastDP to perform detailed placement can achieve a smaller total wirelength, but the cycletime improvement we achieved at the global placement stage will be degraded a lot. In comparison, the weighted FastDP results in a small increase in the total wirelength, but the final cycletime of the circuit will be much better. Also, the TD-FastDP approach is better than the legalization approach, which does not perform any optimization on wirelength and timing. As extra computation is required to calculate the cost based on net weights, the TD-FastDP is slower than FastDP, but the runtime increase in absolute terms is not significant.

We also demonstrate our approaches using synchronous benchmarks of ICCAD 2015 incremental timing-driven placement contest [36]. For each benchmark, we first run POLAR in the pure wirelength-driven placement mode to obtain an initial global placement solution. Next, the spreading-inside / spreading-outside approach is applied on the initial global placement solution to optimize the worst arrival time (τ) at V_O of the circuit. Instead of solving $\mathcal{DF}$ using the linear program approach which can have a long runtime, we solve $\mathcal{DF}$ as a maximum cost flow problem using the network flow approach [31]. In particular, the KKT conditions can be formulated as flow constraints and the bounds on $\Delta\lambda_{ij}$ can be formulated as the flow capacities. The formulated maximum cost flow problem is solved using the LEMON graph library [37] based on the cost scaling approach. Finally, we use Encounter to legalize the design. The reported worst arrival time is obtained by running UI-Timer [38] provided by the contest organizer.

The comparison results for synchronous circuits are shown in Table IV. The target density is set based on the requirements of the contest [36]. In addition, we set the parameter $\alpha = 1$ in the normalized objective function. The "Wirelength" column shows the wirelength reported by Encounter. The "Cycletime" column shows the worst case arrival time obtained by UI-Timer. The "INIT" column denotes the initial wirelength-driven global placement results obtained by POLAR. The "SI" column denotes the spreading-inside approach. The "SO" column denotes the spreading-outside approach. For the wirelength, both the SI and SO approach is 3.9% better than the initial placement. The reason is that the wirelength of the initial placement is obtained by running POLAR in pure WL-driven mode for 75 iterations. Then, using the initial WL-driven placement result as an input, another 45 iterations of TD-POLAR is performed for the timing-driven placement. Since more global placement iterations are performed to generated the SI and SO wirelength, it is possible for the SI and SO approaches to have a better wirelength than the initial WL-driven placement. Compared with the cycletime, the spreading-inside approach and the spreading-outside approach can achieve respectively 6.1% and 10.9% improvement on the initial global placement results.

In the end, the runtime breakdown of the SO approach on ICCAD 2015 incremental timing-driven placement benchmarks is shown in Table V. The "WL-driven" column shows the runtime of running POLAR in pure wirelength driven mode to generate initial placement solution. The "PCG" column shows the runtime used for solving Equation 16 by the preconditioned conjugate gradient (PCG) method with incomplete Cholesky decomposition, as described in Section IV - B 1). The "RL" column shows the runtime used for performing rough legalization, as described in Section IV - B 2). The "TDP total" column shows the total runtime of the timing-driven placement stage.

VI. Conclusion

In this paper, we have formulated a general timing-driven placement problem which is applicable to various design styles. The proposed problem is solved through Lagrangian relaxation technique. We simplified the relaxed problem using KKT conditions and proposed two different approaches on incorporating the LR framework to solve the formulated general timing-driven placement problem. One approach provides a quick way to leverage existing wirelength-driven placers on solving the timing-driven placement problem. The other approach provides an option to tightly combine the LR framework with the existing wirelength-driven placer, and hence better results can be achieved. To demonstrate the proposed approaches, we implemented a placement tool based on a state-of-the-art wirelength driven quadratic placer. The experimental results shows our approaches can greatly improve the performance of the given circuits at the placement stage.

VII. Acknowledgments

We would like to give special thanks to Ismail Bustany of Mentor Graphics for providing invaluable insights and helping us to refine this paper.

References

[1] L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng in *Electronic Design Automation: Synthesis, Verification, and Test*, pp. 635–685, Morgan Kaufmann, 2009.

[2] M. Burstein and M. N. Youssef, "Timing Influenced Layout Design," in *DAC*, pp. 124–130, IEEE Press, 1985.

[3] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," in *DAC*, pp. 269–274, ACM, 1998.

[4] T. Gao, P. M. Vaidya, and C. Liu, "A Performance Driven Macro-cell Placement Algorithm," in *DAC*, pp. 147–152, IEEE, 1992.

[5] M. A. Jackson and E. S. Kuh, "Performance-driven Placement of Cell based IC's," in *DAC*, pp. 370–375, ACM, 1989.

[6] D. Z. Pan, B. Halpin, and H. Ren, "Timing-driven Placement," *Handbook of Algorithms for VLSI Physical Automation*, pp. 223–233, 2007.

[7] G. Wu, T. Lin, H.-H. Huang, C. Chu, and P. A. Beerel, "Asynchronous Circuit Placement by Lagrangian Relaxation," in *ICCAD*, pp. 641–646, IEEE Press, 2014.

[8] R. Karmazin, S. Longfield, C. T. O. Otero, and R. Manohar, "Timing Driven Placement for Quasi Delay-Insensitive Circuits," in *Asynchronous Circuits and Systems (ASYNC), 2015 21st IEEE International Symposium on*, pp. 45–52, IEEE, 2015.

[9] G. Wu and C. Chu, "Simultaneous slack matching, gate sizing and repeater insertion for asynchronous circuits," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2016*, pp. 1042–1047, EDA Consortium, 2016.

[10] A. P. Hurst, P. Chong, and A. Kuehlmann, "Physical Placement Driven by Sequential Timing Analysis," in *ICCAD 2004*, pp. 379–386, Nov 2004.

[11] G. Wu and C. Chu, "Detailed placement algorithm for VLSI design with double-row height standard cells," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1569–1573, 2016.

[12] G. Wu, Y. Xu, D. Wu, M. Ragupathy, Y.-y. Mo, and C. Chu, "Flip-flop Clustering by Weighted K-means Algorithm," in *DAC*, pp. 1–6, IEEE, 2016.

[13] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., 1993.

[14] C.-P. Chen, C. Chu, and D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," in *ICCAD 1998*, pp. 617–624, Nov 1998.

[15] A. Srinivasan, K. Chaudhary, and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 39, no. 11, pp. 825–840, 1992.

[16] T. Hamada, C.-K. Cheng, and P. M. Chau, "Prime: A Timing-driven Placement Tool using A Piecewise Linear Resistive Network Approach," in *DAC*, pp. 531–536, ACM, 1993.
[17] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *TCAD*, vol. 10, no. 3, pp. 356–365, 1991.
[18] M.-C. Kim, D.-J. Lee, and I. L. Markov, "SimPL: An Effective Placement Algorithm," *TCAD*, vol. 31, no. 1, pp. 50–60, 2012.
[19] T. Lin, C. Chu, J. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: Placement Based on Novel Rough Legalization and Refinement," in *ICCAD 2013*, pp. 357–362, Nov 2013.
[20] P. A. Beerel, G. Dimou, and A. Lines, "Proteus: An ASIC Flow for GHz Asynchronous Designs," *Design Test of Computers, IEEE*, vol. 28, pp. 36–51, Sept 2011.
[21] W. C. Naylor, R. Donelly, and L. Sha, "Non-linear Optimization System and Method for Wirelength and Delay Optimization for An Automatic Electric Circuit Placer," Oct. 9 2001. US Patent 6,301,693.
[22] C. Leiserson and J. Saxe, "Optimizing Synchronous Systems," *Journal of VLSI and computer systems*, vol. 1, no. 1, pp. 41–67, 1983.
[23] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, vol. 6, no. 1-6, pp. 5–35, 1991.
[24] J. P. Fishburn, "Clock Skew Optimization," *Computers, IEEE Transactions on*, vol. 39, no. 7, pp. 945–951, 1990.
[25] J. Magott, "Performance Evaluation of Concurrent Systems using Petri Nets," *Information Processing Letters*, vol. 18, no. 1, pp. 7–13, 1984.
[26] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.
[27] J. T. Udding, "A Formal Model for Defining and Classifying Delay-insensitive Circuits and Systems," *Distributed Computing*, vol. 1, no. 4, pp. 197–204, 1986.
[28] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.
[29] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 129–140, 2003.
[30] G. Wu, A. Sharma, and C. Chu, "Gate Sizing and Vth Assignment for Asynchronous Circuits Using Lagrangian Relaxation," in *Asynchronous Circuits and Systems (ASYNC)*, 2015.
[31] J. Wang, D. Das, and H. Zhou, "Gate Sizing by Lagrangian Relaxation Revisited," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 1071–1084, July 2009.
[32] M. Pan, N. Viswanathan, and C. Chu, "An Efficient and Effective Detailed Placement Algorithm," in *ICCAD 2005*, pp. 48–55, Nov 2005.
[33] A. M. Lines, *Pipelined Asynchronous Circuits*. Master's thesis, California Institute of Technology, 1998.
[34] M. Prakash, *Library Characterization and Static Timing Analysis of Asynchronous Circuits*. ProQuest, 2007.
[35] Y. Thonnart, E. Beigne, and P. Vivet, "A Pseudo-synchronous Implementation Flow for WCHB QDI Asynchronous Circuits," in *Asynchronous Circuits and Systems (ASYNC), 2012*, pp. 73–80, May 2012.
[36] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 CAD Contest in Incremental Timing-driven Placement and Benchmark Suite," in *ICCAD*, pp. 921–926, IEEE Press, 2015.
[37] LEMON Graph Library: http://lemon.cs.elte.hu/.
[38] T.-W. Huang, P.-C. Wu, and M. D. Wong, "UI-timer: An Ultra-fast Clock Network Pessimism Removal Algorithm," in *ICCAD*, pp. 758–765, IEEE Press, 2014.