SensTDDP: A Timing Sensitivity Analysis Framework with Application to Timing-Driven Detailed Placement

QIANG YANG, Fuzhou University, China
JIAHUI LI, Fuzhou University, China
HONGXI WU, Fuzhou University, China
XINGQUAN LI, Peng Cheng Laboratory, China
BEI YU, The Chinese University of Hong Kong, Hong Kong
WENXING ZHU, Fuzhou University, China

Timing convergence is paramount for the feasibility of VLSI circuit design, which is highly dependent on timing optimization during VLSI placement. Timing-driven placement usually achieves timing optimization by optimizing the locations of timing-violating cells. These cells are often characterized by timing-criticality in global and detailed placement. However, we find that this metric cannot accurately capture the cells whose movement will affect the overall timing results. To bridge this gap, this paper proposes a timing sensitivity analysis framework to precisely quantify the impact of physical objects (pins, combinational cells, FFs, nets) on overall timing. Within this framework, we derive the TNS and WNS sensitivities of pins, combinational cells, FFs and nets. Moreover, we introduce a metric of total timing sensitivity to estimate how much physical objects affect the total timing. To validate its effectiveness, the timing sensitivity analysis framework is utilized to refine the combinational cell movement techniques in Rsyn [6, 7]. Moreover, we develop an FF classification and moving scheme based on the timing sensitivity analysis framework, to further enhance timing optimization. Experimental results show that our approach achieves remarkable average improvements on TNS and WNS without compromising total wirelength and routability, compared to the state-of-the-art timing-driven detailed placer.

CCS Concepts: • **Hardware → Placement**.

Additional Key Words and Phrases: timing optimization, timing sensitivity analysis, timing-driven detailed placement, cell movement.

ACM Reference Format:
Qiang Yang, Jiahui Li, Hongxi Wu, Xingquan Li, Bei Yu, and Wenxing Zhu. 2026. SensTDDP: A Timing Sensitivity Analysis Framework with Application to Timing-Driven Detailed Placement. *ACM Trans. Des. Autom. Electron. Syst.* 1, 1 (February 2026), 32 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In VLSI design, the performance of a placement engine is crucial to the overall physical design flow [20]. The main task of placement is to position circuit modules without overlaps within a die, and optimize some specific metrics. Traditionally, the main focus of placement has been on

Authors' addresses: Qiang Yang, 230320047@fzu.edu.cn, Fuzhou University, Fuzhou, China; Jiahui Li, 225420004@fzu.edu.cn, Fuzhou University, Fuzhou, China; Hongxi Wu, 225410009@fzu.edu.cn, Fuzhou University, Fuzhou, China; Xingquan Li, lixq01@pcl.ac.cn, Peng Cheng Laboratory, Shenzhen, China; Bei Yu, byu@cse.cuhk.edu.hk, The Chinese University of Hong Kong, Hong Kong, Hong Kong; Wenxing Zhu, wxzhu@fzu.edu.cn, Fuzhou University, Fuzhou, China.

1084-4309/2026/2-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

wirelength, often neglecting timing. While optimizing wirelength can improve timing to some extent, it tends to ignore the significant difference between critical and non-critical nets, potentially yielding longer critical paths and resulting in poor timing performance. However, the convergence of VLSI design is largely dependent on achieving the final timing goal. In practice, many designs require several rounds of placement and routing to meet the goal, prolonging the design process. Therefore, timing-driven placement is indispensable to speed up the design cycle.

Placement in VLSI physical design is usually divided into three stages: global placement, legalization, and detailed placement. At each stage, the timing target can be considered and optimized. Modern timing-driven placement approaches can typically be grouped into two main stages: timing-driven global placement [8, 9, 15–18, 24, 26, 27] and timing-driven detailed placement [3, 4, 6, 7, 10–12, 14, 19, 22]. Timing-driven placement approaches all aim to reduce timing violations in a circuit by rationally planning the positions of cells. A good timing evaluation metric will greatly help timing-driven placement. The timing-criticality of a cell is often defined as the ratio of the negative slack value of the cell to the WNS, which reflects whether the cell has a timing violation and the severity of the timing violation. Therefore, the timing-criticality metric is widely used in timing-driven global placement and timing-driven detailed placement.

Popular timing-driven global placement can be categorized into weight-based [8, 15, 17, 18, 27] and gradient-based methods [9, 16]. Weight-based methods typically optimize the timing by assigning higher weights to critical nets or paths, and the setting of the weights is often directly related to the criticality of the cell. Gradient-based methods compute the gradient of the timing objective with respect to a cell's coordinates, guiding the movement of the cell.

The timing-criticality metric is not only widely used for setting weights in timing-driven global placement, but also for the selection and ordering of cells in timing-driven detailed placement. Timing-driven detailed placement, moving cells locally, can further enhance the timing result of global placement. Typically, timing-driven detailed placement optimizes cells' positions along critical paths using path-based methods. Recently, Lagrangian relaxation (LR)-based approaches have been widely adopted [10, 17, 19, 22]. They use TNS as the objective function for LR, guiding discrete local search algorithm to relocate cells. However, LR-based approaches are time-consuming, since they require local search to find cells' optimal positions from multiple candidates.

Another effective approach to timing-driven detailed placement involves straightening critical paths [3, 12]. This approach often reduces the delay in the critical path by adjusting the positions of cells on the critical path. Rsyn [6, 7] combines quadratic placement with various cell movement methods, such as clustered movement, buffer balancing, cell balancing, and load optimization, to achieve state-of-the-art timing result in timing-driven detailed placement.

Detailed placement does not expect to move all cells. All existing timing-driven detailed placement approaches select cells for moving solely based on criticality, i.e., the presence of timing violation. They overlook the fact that not all cells with timing violation will significantly impact timing, and some cells without timing violation can still influence overall timing performance. This fact will be demonstrated in Subsection 3.1. Moreover, cells' moving order also significantly affects the final timing results. Conventional approaches typically sort cells based on timing criticality [6, 7], and specifically the cells on the same timing path are assigned identical criticality and sorted in reverse or forward topological order [17, 22]. These approaches have not considered the timing impact of cells, often leading to suboptimal solutions.

Therefore, it is important to find a metric that can assess the extent of the cell's impact on timing. The concept of timing gradient, first proposed by [9], quantifies the pin-to-timing contribution at the global placement stage. However, this approach requires smoothing a large number of functions (including the maximum function and the distance function). This coarse-grained evaluation loses accuracy [27].

Furthermore, as the start and end points of timing paths, FFs play a critical role in timing optimization. Most strategies for FFs focus on addressing early timing violations by optimizing clock paths [11, 14]. However, improving FFs' placement to mitigate late timing violations is equally important yet often overlooked, including Rsyn [6, 7].

Considering the pros and cons of previous work, this paper introduces a timing sensitivity analysis framework and applies it to the combinational cell movement techniques of Rsyn [6, 7]. Additionally, we propose an FF movement scheme based on the timing sensitivity analysis framework to further mitigate late timing violations. The main contributions of this work are summarized as follows:

- We give the definition of timing sensitivity that directly estimates the impacts of position changes of pins, combinational cells, and FFs, as well as wirelength changes of nets, on TNS and WNS. The definition is calculated directly as the variation of timing metric with respect to wirelength, instead of cell's coordinate.
- We propose a timing sensitivity analysis framework and derive in detail the timing sensitivities of individual physical objects (pins, combinational cells, FFs, and nets). Specifically, the timing sensitivity computation is performed by analyzing finite slew propagation and cell delay variation. Moreover, we introduce the total timing sensitivity metric to assess the impact of physical objects on overall timing.
- We apply the timing sensitivity analysis framework to the combinational cell movement schemes in Rsyn [6, 7]. Our approach focuses on selecting combinational cells that have a direct impact on TNS and WNS, prioritizing them based on their total timing sensitivity values.
- We classify FFs by their pins' total timing sensitivities and propose an FF movement strategy. This strategy minimizes the local delay influenced by FFs, and complements the enhanced combinational cell movements to enable further timing optimization.
- Experiments on placements generated by DREAMPlace 4.0_GP [17] and ICCAD 2015 contest benchmarks show that our approach achieves significant improvements in WNS and TNS, compared to state-of-the-art timing-driven detailed placer. Meanwhile, we demonstrate through ablation experiments that both our timing sensitivity framework and FF movement strategy are effective.

This paper is organized as follows. Section 2 presents the preliminaries, introducing the concepts of timing and the techniques already available in Rsyn. Section 3 gives our definition of timing sensitivity, and provides the framework of timing sensitivity analysis. Moreover, this section also derives in detail the timing sensitivities of physical objects (pins, combinational cells, FFs, and nets). Section 4 applies our timing sensitivity analysis framework in Rsyn for timing-driven detailed placement. Section 5 details our FF movement scheme. Experimental results are presented in Section 6. Section 7 concludes this work.

2 PRELIMINARIES

This section first introduces the basic concepts of Static Timing Analysis (STA), and then outlines the late timing violation optimization techniques of Rsyn [6, 7].

2.1 Static Timing Analysis

Static Timing Analysis [2] starts by representing a circuit as a directed acyclic graph (DAG), in which each pin is treated as a node and connections among pins as edges. Each edge carries an associated delay, which can be a cell delay or a wire delay. After determining the delay for each edge, the delay is propagated, which involves both forward propagation and backward propagation. The processes are used to calculate the actual arrival time (AAT) and required arrival time (RAT) for each node, respectively. The slack at any node u is the difference between its RAT and AAT:

$$Slack(u) = RAT(u) - AAT(u).$$

A negative slack indicates that a timing violation occurs at this node. WNS is the worst slack among all timing endpoints, which are the primary outputs or data inputs of latches or flip-flops, defined by:

$$WNS = \min_{u \in P_o} Slack(u),$$

where P_o is the set of timing endpoints. TNS is the total sum of negative slacks at the timing endpoints, defined by

$$TNS = \sum_{u \in P_o} \min(0, Slack(u)).$$

Slack, WNS, and TNS are crucial timing metrics for characterizing circuit timing convergence. These metrics can be obtained by running an STA tool during the optimization phase.

In this work, we adopt the Non-Linear Delay Model (NLDM) for cell delay calculation and the Elmore delay model for net delay estimation [5]. An NLDM cell delay is calculated from look-up tables (LUTs), and the prevailing approach involves processing lookup tables through bilinear interpolation [2, 8, 9], so we also express it in the form of bilinear interpolation. Specifically, for a cell c_k, the cell delay D_{c_k} and the output slew S_{c_k} are given by

$$D_{c_k} = LUT_{D_{c_k}}(\text{Slew}_{c_k}, \text{Cap}_{c_k}) = d_I + a_k \cdot \text{Slew}_{c_k} + b_k \cdot \text{Cap}_{c_k} + z_k \cdot \text{Slew}_{c_k} \cdot \text{Cap}_{c_k}, \tag{1}$$

$$S_{c_k} = LUT_{S_{c_k}}(\text{Slew}_{c_k}, \text{Cap}_{c_k}) = s_I + w_k \cdot \text{Slew}_{c_k} + v_k \cdot \text{Cap}_{c_k} + u_k \cdot \text{Slew}_{c_k} \cdot \text{Cap}_{c_k}, \tag{2}$$

where Slew_{c_k} and Cap_{c_k} are the input slew and the total load capacitance of the cell, respectively. d_I, a_k, b_k, z_k, s_I, w_k, v_k, u_k are the constant terms when using the interpolation method. It is worth noting that the cell delay is essentially defined from pin to pin of the cell, hence the parameters Slew_{c_k} and Cap_{c_k} are only related to the cell's pins.

For a net e, the wire delay D_{e_u} and slew S_{e_u} from its source i to a sink u are calculated by the Elmore delay model:

$$D_{e_u} = R_d C_{\text{total}} + rC_l L_u + \frac{rcL_u^2}{2}, \tag{3}$$

$$S_{e_u} = K_s \left(R_d C_{\text{total}} + rC_l L_u + \frac{rcL_u^2}{2} \right), \tag{4}$$

where R_d is the effective output resistance of the driver to which source i is attached, K_s is a constant in the timing library and is set to 2.2. C_{total} is the total load capacitance from all sinks to the source i, and C_l is the load capacitance of the receiver which is attached to sink u. r and c are the wire resistance and capacitance per unit wirelength, respectively. L_u is the wirelength from the source i to sink u.

2.2 Late Timing Violation Optimization Techniques of Rsyn

This subsection introduces the main optimization techniques in Rsyn [6, 7] for resolving late timing violations. Rsyn integrates several combinational cell movement techniques to optimize late timing violations, including quadratic placement and clustered movement for synchronized cells' movement, as well as buffer balancing, cell balancing, and load optimization for single-cell movement.

The quadratic placement technique in Rsyn optimizes late timing violations by assigning higher weights to the wires of the top-three timing critical paths, and minimizes the total weighted quadratic wirelength:

$$W(x,y) = \sum_{i}^{N} \sum_{j}^{N} g_{ij} (L_{ij})^2, \tag{5}$$

where L_{ij} is the length of the interconnection from cell i to cell j, and g_{ij} is the weight of the interconnection [7]. The quadratic placement technique is commonly used in detailed placement to quickly place cells.

The clustered movement technique in Rsyn clusters combinational cells with timing violations. For a cluster, the target position of clustered cells is computed as [6]:

$$targetPos(cluster) = \frac{\sum_{i=1}^{n} pos(N_i) \cdot slack(N_i)}{\sum_{i=1}^{n} slack(N_i)}, \tag{6}$$

where n is the number of the cluster's neighboring critical pins, $pos(N_i)$ and $slack(N_i)$ are the position and the slack of the cell N_i to which the cluster's neighboring critical pin i is attached. By optimizing the positions of these clusters, the clustered movement technique helps avoid suboptimal placement solutions to some extent.

The buffer balancing and cell balancing techniques in Rsyn relocate buffers and combinational cells so that they lie along the interconnect path between their driver cell and load cell respectively, thereby minimizing local delay. Rsyn applies this technique to all buffers and combinational cells with timing violations. Specifically, for a buffer with timing violation, the local delay value D from its driver cell to its load cell can be expressed as [6]:

$$D = R_0(C_1 + d_0 C_w) + d_0 R_w \left(C_1 + \frac{d_0 C_w}{2}\right) + p_0 + R_1(C_2 + d_1 C_w) + d_1 R_w \left(C_2 + \frac{d_1 C_w}{2}\right) + p_1, \tag{7}$$

where R_0 is the resistance of the buffer's driver, C_1 is the input pin load capacitance of the buffer, d_0 is the wirelength from the buffer to its driver, R_w and C_w are the wire resistance and wire capacitance per unit wirelength respectively, R_1 is the buffer's resistance, d_1 is the wirelength from the buffer to its load cell, C_2 is the load capacitance at the input sink pin of the buffer's load cell. p_0 and p_1 are parasitic delays of the driver and buffer, respectively. By fixing the positions of the driver cell and load cell, buffer balancing determines the optimal position for each buffer by minimizing the D value.

The cell balancing technique is similar to that of buffer balancing. However, a combinational cell usually has multiple input pins and multiple output pins, and in this case Rsyn will compute a weighted position based on multiple target positions. The load optimization technique in Rsyn reduces the load on a critical net by placing non-critical load cells on the net as close as possible to their driver cell.

3 OUR TIMING SENSITIVITY ANALYSIS

In this section, we first demonstrate the insufficiency of moving only timing-violating combinational cells and FFs. Next, we give our definition of timing sensitivity and propose our timing sensitivity analysis framework. Finally, we derive in detail the timing sensitivities for pins, combinational cells, FFs, and nets.

3.1 Insufficiency of Moving Only Timing-Violating Combinational Cells and FFs

Popular timing-driven detailed placement approaches [6, 22] primarily utilize single-cell movement techniques to move all combinational cells and FFs with timing violations to improve TNS and WNS. However, these techniques overlook the fact that not all combinational cells with timing-violation will affect TNS and WNS, as the slacks of certain combinational cells may not propagate backward to contribute to TNS and WNS [24]. Furthermore, they also disregard the fact that moving combinational cells and FFs without timing violations could potentially improve TNS and WNS. Fig. 1 provides a simple example to illustrate this fact, where the slack value of an object is denoted by S.

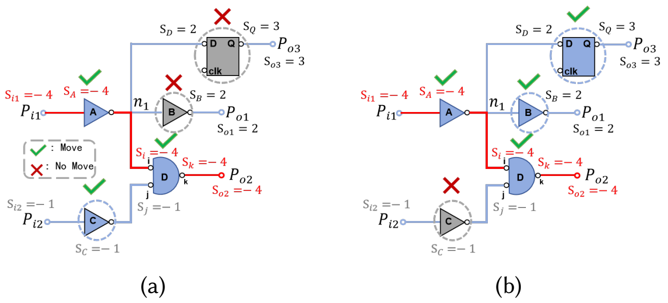

Fig. 1. (a) Combinational cells and FFs identified for moving by the single-cell movement techniques in [6, 22]. (b) Combinational cells and FFs identified for moving by our sensitivity analysis.

In the circuit of Fig. 1(a), both the TNS and WNS in the circuit are −4. The single-cell movement techniques in [6, 22] would choose to move combinational cells with negative slacks (indicating timing violation) for timing optimization, i.e., combinational cells A, C, and D, prioritized by their timing criticality. The combinational cell B and the FF would not be selected for moving due to positive slack. Since combinational cells A and D are on the critical path, colored in red, moving them will result in significant improvements in TNS and WNS. However, relocating combinational cell C is not essential, because the improvement in slack from moving combinational cell C does not propagate backward to the slack −4 at pin k of combinational cell D. Consequently, there is no change in slack −4 at the output port P_{o2} after relocating cell C, and therefore there is no change in TNS and WNS.

In Fig. 1(b), combinational cells A, B, D, and the FF should be moved to optimize TNS and WNS, since A and D are on the critical path, colored in red, and moving them can directly improve TNS and WNS. Moreover, moving combinational cell B alters the length of net n_1, and the length adjustment affects the actual arrival time (AAT) at combinational cell D's most critical input pin i, which then propagates backward to the slack −4 of the output port P_{o2}, and ultimately affecting the TNS and WNS. However, moving the combinational cell B is often overlooked by the single-cell movement techniques in [6, 22], due to no timing violations. The same situation applies to the FF in the figure. Additionally, the techniques may inappropriately move combinational cell C, which does not affect TNS and WNS but could adversely affect other paths associated with combinational cell C.

From the above analysis, it is evident that traditional critical path-based methods cannot accurately identify the cells that need to be moved to optimize WNS and TNS, as they rely solely on slack as the selection metric. To more accurately identify cells for timing optimization, a new metric should be developed for effectively quantifying a cell's impact on overall timing, i.e., timing sensitivity.

Traditionally, timing sensitivity analysis focuses on nets directly [21, 24], indicating the extent to which a change in the wirelength of a critical net could potentially affect timing. However, the definition of net timing sensitivity in [24] has not considered the effect of wirelength change on cell delay and slew propagation, and the net timing sensitivity in [21] is not directly correlated with the final timing metrics, TNS and WNS. Recently, the timing gradient with respect to cell coordinates,

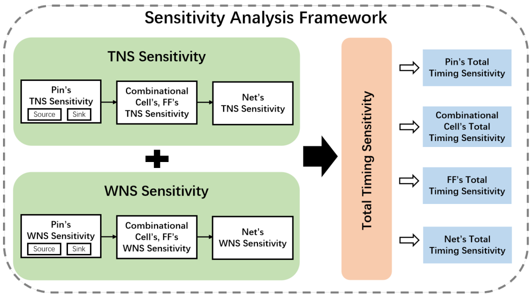
Sensitivity Analysis Framework
TNS Sensitivity
Pin's TNS Sensitivity
Source
Sink
Combinational Cell's, FF's TNS Sensitivity
Net's TNS Sensitivity
WNS Sensitivity
Pin's WNS Sensitivity
Source
Sink
Combinational Cell's, FF's WNS Sensitivity
Net's WNS Sensitivity
Total Timing Sensitivity
Pin's Total Timing Sensitivity
Combinational Cell's Total Timing Sensitivity
FF's Total Timing Sensitivity
Net's Total Timing Sensitivity

Fig. 2. Our timing sensitivity analysis framework.

proposed in [9], quantifies the contribution of each pin's arrival time to the overall timing metrics. However, calculating the gradient requires smoothing a large number of functions so that they are differentiable, which may potentially compromise accuracy [27]. Hence, developing a more comprehensive and accurate timing sensitivity analysis framework is necessary.

3.2 Our Timing Sensitivity Definition and Timing Sensitivity Analysis Framework

This subsection gives our definition of timing sensitivity. More specifically, we define TNS and WNS sensitivities for pin, cell (including combinational cell and flip-flop), and net.

Definition 3.1. **TNS and WNS Sensitivities of a pin or cell:** The degrees to which TNS and WNS are affected by wirelength variations caused by the position change of a pin or cell. They are denoted respectively by

$$S_{\text{TNS}}(X) = \frac{\Delta\text{TNS}}{\Delta L_X}, \quad S_{\text{WNS}}(X) = \frac{\Delta\text{WNS}}{\Delta L_X},$$

where ΔTNS and ΔWNS are the variations of TNS and WNS, X is the physical object (pin or cell), ΔL_X is the wirelength variation caused by the position change of X.

Definition 3.2. **TNS and WNS Sensitivities of a net:** The degrees to which TNS and WNS are affected by the wirelength change of a net. They are denoted respectively by

$$S_{\text{TNS}}(net) = \frac{\Delta\text{TNS}}{\Delta L_{net}}, \quad S_{\text{WNS}}(net) = \frac{\Delta\text{WNS}}{\Delta L_{net}},$$

where ΔTNS and ΔWNS are the variations of TNS and WNS, ΔL_{net} is the wirelength change of the net.

Unlike the timing gradient proposed in [9], our timing sensitivity is not defined with respect to the spatial coordinates of pins or cells, but rather in terms of the wirelength changes induced by their movement. This is due to the fact that delays (Eqs. (1) and (3)) are a function of wirelength rather than the absolute positions of pins or cells.

Since calculating TNS and WNS is essentially performed at pins, we first derive the timing sensitivities of pins, then we calculate the timing sensitivities of cells and subsequently the nets. Fig. 2 presents our overall timing sensitivity analysis framework. It begins with the timing sensitivity of the pin, followed by that of the combinational cell and flip-flop, and finally the net. To effectively evaluate the TNS and WNS sensitivities of a net, it is critical to calculate the TNS and WNS sensitivities of the associated cells first, since the net length change primarily results from the cells' movements. Furthermore, calculating a cell's TNS and WNS sensitivities must assess the

TNS and WNS sensitivities of its pins first. This step is crucial because the timing information is defined directly at the pin level, and relocating a cell affects its pins' positions. In addition, to comprehensively estimate the impact of physical objects (pins, combinational cells, FFs and nets) on overall timing, we introduce a metric of total timing sensitivity. This metric combines the WNS and TNS sensitivities of these physical objects linearly. The next subsection outlines how to compute the timing sensitivities of these objects effectively.

3.3 Computation of Timing Sensitivity

In this subsection, we describe the computation of TNS and WNS sensitivities for pins, combinational cells, FFs, and nets, and then introduce the concept of total timing sensitivity.

STA represents a circuit as a directed acyclic graph DAG, where each pin corresponds to a vertex. According to Fig. 2, we first compute the TNS and WNS sensitivities for the pins, and then propagate them to obtain the timing sensitivities of cells and nets. It is worth noting that when computing the timing sensitivity of a pin, we assume that all other pins in the DAG are fixed, including the other pins attached to the same cell.

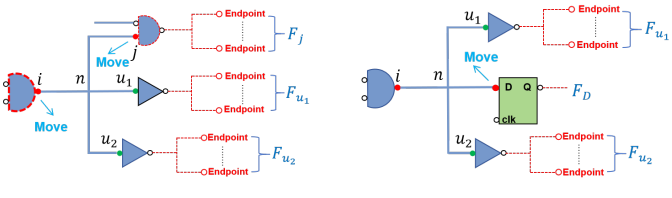

Fig. 3. (a) An example for illustrating the effect of position changes of sink j and source i on TNS and WNS. AATs of green pins are affected by the position change of sink j. F_j is the number of timing critical endpoints influenced by sink j. (b) An example illustrating the effect of the position change of sink D pin of the FF on TNS and WNS. AATs of green pins are affected by the position change of the D pin.

*3.3.1 **Computation of Pin's Timing Sensitivity.***

There exist two types of pins: source and sink pins, which serve as input and output signal points, respectively. To facilitate comprehension, we use Fig. 3 to illustrate the timing sensitivities of sinks and sources, which are the input and output pins of some combinational cells or FFs, respectively. As shown in Fig. 3(a), changing the position of sink j alters the wirelength from it to source i, which in turn affects the delays of all timing paths passing through the two pins. Yet, changing the position of source i affects the wirelengths from source i to all sinks j, u_1, and u_2, indicating that the wirelength of the entire net n changes, which in turn affects all timing paths through the net n. So the effect of the source's position change on the timing is equivalent to the effect of the position changes of all the sinks on the timing. Therefore, the position change of these two kinds of pins yields disparate impacts on timing, and we need to compute the timing sensitivities of the sink and source separately.

First, we calculate the TNS and WNS sensitivities of a sink. We use L_j to denote the wirelength from sink j to its source, i.e., the wirelength from sink j to source i of net n in Fig. 3(a). Let AAT_u be the actual arrival time at sink u, and let ΔAAT_u and ΔL_j be the variations of AAT_u and L_j, respectively. Moreover, let a_u, z_u, b_i, z_i, u_i and v_i be the fitting coefficients defined in Eqs. (1) and (2), with each cell having its own distinct set of fitting parameters. Cap_{c_u} and $Slew_{c_i}$ are the current load capacitance and output slew of the cell corresponding to pins u and i, respectively. Furthermore, let r and c be the resistance of the wire and the capacitance per unit wirelength, respectively. R_d denotes the effective output resistance of the driver to which the source i is attached, and let C_l be the load capacitance of the receiver to which the sink u is attached. K_s is the constant 2.2 in the timing library.

We first compute $S_{L_j}^{AAT_u}$, which indicates the actual arrival time (AAT) sensitivity to wirelength L_j at any sink u of net n. Note that, for any load cell of net n, the delay change of the load cell actually affects the AAT of its output pin; however, to simplify the presentation we assume that the delay change of the load cell affects the AAT of its input pin of the net n instead of the output pin. This assumption does not affect the result. We have the following two lemmas:

Lemma 3.3. *Consider a net n with source i and multiple sinks. If $j \in Sink(n)$ is a combinational cell's pin, the AAT sensitivity of each sink $u \in Sink(n)$ to wirelength L_j is:*

$$S_{L_j}^{AAT_u} = \frac{\Delta AAT_u}{\Delta L_j}$$
$$= \begin{cases} (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j) & \\ +(a_u + z_u \cdot Cap_{c_u})(v_i c + u_i \cdot Slew_{c_i} \cdot c + rK_sC_l + cK_sR_d + K_s rcL_j), & if\, u = j; \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d) + (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot (v_i + u_i \cdot Slew_{c_i} + K_sR_d), & otherwise. \end{cases} \tag{8}$$

Proof. See Appendix A. □

Next, it is necessary to calculate the sensitivity of the D pin of an FF separately, since it is the endpoint of a timing path, as shown in Fig. 3(b).

Lemma 3.4. *Consider a net n with source i and multiple sinks. If $j \in Sink(n)$ is the D pin of an FF, the AAT sensitivity of each sink $u \in Sink(n)$ to wirelength L_j is:*

$$S_{L_j}^{AAT_u} = \frac{\Delta AAT_u}{\Delta L_j}$$
$$= \begin{cases} b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j, & if\, u = D\, pin; \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d) + (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot (v_i + u_i \cdot Slew_{c_i} + K_sR_d), & otherwise. \end{cases} \tag{9}$$

Proof. See Appendix B. □

Unlike Lemma 3.3, in Lemma 3.4 when j is the D pin of the FF, the AAT_j sensitivity to wirelength L_j is not correlated with the parameters a_u, z_u and Cap_{c_u}, which also reflects the fact that the variation of L_j does not affect the delay of the FF.

Lemmas 3.3 and 3.4 show that, a change in L_j affects the AATs at all sinks in a net. But these AATs do not necessarily contribute to the calculation of TNS and WNS. Therefore, we further calculate a sink's TNS and WNS sensitivities based on Lemmas 3.3 and 3.4.

Theorem 3.5. *Consider a net n with source i, the TNS sensitivity of sink $j \in Sink(n)$ is:*

$$S_{TNS}(j) = \frac{\Delta TNS}{\Delta L_j} = -\sum_{u \in Sink(n)} F_u \cdot S_{L_j}^{AAT_u}, \tag{10}$$

where ΔTNS and ΔL_j are the variations of TNS and L_j, F_u is the number of timing critical endpoints influenced by the sink u.

Proof. See Appendix C. □

Theorem 3.5 suggests that TNS is more sensitive to the wirelength L_j, if the net n has a larger number of sinks, or these sinks include a greater number of affected timing-critical endpoints F_u, or these sinks have a larger value of $S_{L_j}^{AAT_u}$. It is worth noting that, F_u is not calculated simply as the sum of all critical timing endpoints fanned out by sink u, since if u is on a non-critical or sub-critical path of a particular endpoint, the slack value of that endpoint is not affected by u. Therefore, we need to check whether u is on the same most critical path as this endpoint to determine whether this timing endpoint will be affected by u. The computation of F_u is based on the algorithm proposed in [24].

F_u allows us to determine whether the arrival time change of pin u will affect the final timing result. With F_u, Theorem 3.5 calculates the TNS sensitivity of sink j by accurately assessing the contribution of AAT to TNS for all sinks. Although [9] has proposed the idea of quantifying the contribution of the arrival times of pins to the overall timing, it requires smoothing a large number of maximum functions such that they are differentiable, which may potentially compromise accuracy [27].

Theorem 3.6. *Consider a net n with source i, the WNS sensitivity of sink $j \in Sink(n)$ is:*

$$S_{WNS}(j) = \frac{\Delta WNS}{\Delta L_j} = \begin{cases} -S_{L_j}^{AAT_u}, & \text{if there exists a sink } u \in Sink(n) \text{ on the worst timing path;} \\ 0, & \text{otherwise.} \end{cases}$$

where ΔWNS and ΔL_j are the variations of WNS and L_j.

Proof. See Appendix D. □

Theorem 3.6 implies that WNS is sensitive to the change of the wirelength L_j, only if there exists a sink u in the net n and u lies on the worst timing path. Note that the WNS sensitivity of sink j of net n depends on whether any sink on net n is in the worst timing path, not only on j itself.

In the above derivation for sink pins, the position change of a sink impacts only a single two-pin wire. In contrast, the timing sensitivities of source pins, cells and nets affect multiple two-pin wires, requiring an analysis of their combined timing impact, which is more complex and involves multiple scenarios. To account for the worst-case timing scenario, we focus on the most sensitive case to derive their timing sensitivities. In the subsequent derivations, we will provide a clear explanation of these most sensitive cases.

Next, we calculate the TNS and WNS sensitivities of source pin in the most sensitive case. The WNS and TNS sensitivities of the Q pin of an FF can be derived similarly.

Theorem 3.7. *Consider a net n with source i, the TNS sensitivity of source i under its maximum impact on TNS is:*

$$S_{TNS}(i) = \frac{\Delta TNS}{\Delta L_n} = \sum_{u \in Sink(n)} S_{TNS}(u), \tag{11}$$

where L_n is the wirelength of net n, ΔTNS and ΔL_n are the variations of TNS and L_n.

Proof. (See Fig. 3(a)) Assume that all other pins' positions are fixed. Moving source i only affects the wirelength of net n. Also, changing the position of source i can be equivalently viewed as the distance change from the source i to every sink $u \in Sink(n)$, which affects the AATs of these sinks, and each sink influences multiple timing critical endpoints, which subsequently affects TNS. Therefore, in the most sensitive case, the effect on TNS by moving source i of net n is equivalent to summing the effects on TNS by moving all sinks $u \in Sink(n)$ of the net n. The detailed proof is provided in Appendix E. □

Theorem 3.8. *Consider a net n with source i, the WNS sensitivity of source i under its maximum impact on WNS is:*

$$S_{WNS}(i) = \sum_{u \in Sink(n)} S_{WNS}(u).$$

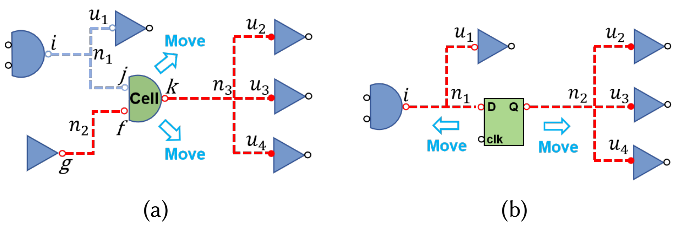

Fig. 4. (a) A simple example to show the impact of moving combinational cell on TNS and WNS. Blue dotted line indicates non-critical net, while red dotted line indicates critical net. (b) A simple example to show the impact of FF movement on TNS and WNS.

Proof. (See Fig. 3(a)) By the proof of Theorem 3.7, relocating source i of net n is equivalent to moving all sinks $u \in Sink(n)$ of the net n. Based on Theorem 3.6, each sink might affect the WNS. Thus, in the most sensitive case, the effect of moving source i of net n on WNS is equivalent to summing the effects on WNS by moving all sinks $u \in Sink(n)$ of the net n. The detailed proof is provided in Appendix F. □

3.3.2 *Computation of Combinational Cell's and FF's Timing Sensitivity.*

Subsection 3.3.1 has obtained the TNS and WNS sensitivities for input and output pins of a combinational cell. Subsequently, based on the timing sensitivities of these pins, we can derive the combinational cell's TNS and WNS sensitivities in the following theorems.

Theorem 3.9. *The TNS sensitivity of a combinational cell (c-cell) under its maximum impact on TNS is:*

$$S_{TNS}(c\text{-}cell) = \sum_{u \in \ input\ pins} S_{TNS}(u) + S_{TNS}(outpin).$$

Proof. See Appendix G. □

Theorem 3.9 implies that, the TNS sensitivity of a combinational cell is determined by the TNS sensitivity of all its pins. Similarly, it is straightforward to get the WNS sensitivity of a combinational cell in the following Theorem 3.10.

Theorem 3.10. *The WNS sensitivity of a combinational cell (c-cell) under its maximum impact on WNS is:*

$$S_{WNS}(c\text{-}cell) = \sum_{u \in \ input\ pins} S_{WNS}(u) + S_{WNS}(outpin).$$

Proof. See Appendix H. □

Theorem 3.9 and Theorem 3.10 indicate that, a combinational cell with a larger number of timing-sensitive pins and higher timing sensitivity values of these pins will lead to a higher timing sensitivity value of the combinational cell. Moreover, Theorem 3.9 and Theorem 3.10 show that, the timing sensitivity of a combinational cell can be directly computed from the timing sensitivities of its pins, since a position change of the cell is effectively equivalent to simultaneous position changes of all its pins.

Since we have derived the timing sensitivities of the D pin and Q pin of an FF in Subsection 3.3.1, it is evident that we can directly determine the TNS and WNS sensitivities of the FF as well, similar to the process for combinational cell. From Fig. 4(b), similar to the proofs of Theorems 3.9 and 3.10, we can derive the WNS and TNS sensitivities of FF as follows:

THEOREM 3.11. *The TNS sensitivity of a flip-flop under its maximum impact on TNS is:*

$$S_{TNS}(FF) = S_{TNS}(D\ pin) + S_{TNS}(Q\ pin).$$

THEOREM 3.12. *The WNS sensitivity of a flip-flop under its maximum impact on WNS is:*

$$S_{WNS}(FF) = S_{WNS}(D\ pin) + S_{WNS}(Q\ pin).$$

Theorems 3.11 and 3.12 show that, the TNS (WNS) sensitivity of an FF is the summation of the TNS (WNS) sensitivities of its D pin and Q pin, reflecting the extent to which the position change of the FF influences TNS (WNS). Here, we do not consider the clock (clk) pin sensitivity for two main reasons. First, including the clk pin sensitivity would classify too many FFs as sensitive, as all FFs driven by the same Local Clock Buffer (LCB) would be marked as sensitive if one FF is critical. Second, our focus is on optimizing the positions of FFs within the logic path, and the clk pin's sensitivity is irrelevant to this aim.

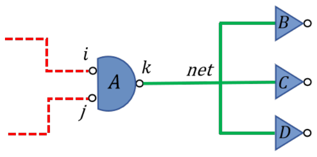

Fig. 5. A simple example for illustrating the impact of net length change on TNS and WNS.

3.3.3 *Computation of Net's Timing Sensitivity.*

Subsection 3.3.2 has derived the TNS and WNS sensitivities of cells (including combinational cells and FFs). Next, we derive the net's TNS and WNS sensitivities based on cell's timing sensitivity.

THEOREM 3.13. *The TNS sensitivity of a net under its maximum impact on TNS is:*

$$S_{TNS}(net) = \sum_{receiver\ u} S_{TNS}(u) + S_{TNS}(driver).$$

PROOF. See Appendix I. □

THEOREM 3.14. *The WNS sensitivity of a net under its maximum impact on WNS is:*

$$S_{WNS}(net) = \sum_{receiver\ u} S_{WNS}(u) + S_{WNS}(driver).$$

PROOF. See Appendix J. □

Theorems 3.13 and 3.14 show that we can directly compute the timing sensitivity of a net from the timing sensitivity of cells. Moreover, Theorems 3.13 and 3.14 indicate that, a net with a larger number of timing-sensitive cells and higher timing sensitivity values of these cells will lead to a higher timing sensitivity value of the net.

Wirelength change of a net is essentially caused by the movement of cells on the net, hence the wirelength change of one net must cause the wirelength of its neighboring net to change as well. However, the net timing sensitivity proposed in [21] and [24] is derived under the assumption that the wirelengths of other nets remain unchanged, which is inaccurate. While, Theorems 3.13 and 3.14 show that we can directly compute the timing sensitivity of a net from the timing sensitivity of cells, which allows us to take into account the wirelength changes of other nets as well when deriving a net's timing sensitivity.

3.3.4 ***Total Timing Sensitivity Metric.*** We have derived the TNS and WNS sensitivities for pins, combinational cells, FFs, and nets. Next, we introduce a total timing sensitivity metric to estimate the overall impact of physical objects (pins, combinational cells, FFs, and nets) on timing, defined as follows:

$$S_{\text{total}}(X) = \gamma \, |S_{\text{TNS}}(X)| + (1-\gamma) \, |S_{\text{WNS}}(X)| \, , \tag{12}$$

where $S_{\text{TNS}}(X)$ is the TNS sensitivity of a physical object X, which can be a pin, combinational cell, FF, or net. Similarly, $S_{\text{WNS}}(X)$ is the WNS sensitivity of a physical object X. The coefficient $\gamma \in [0, 1]$ is a hyperparameter that we set to 0.5.

According to Theorems 3.5 and 3.6, it can be seen that the TNS and WNS sensitivity values of an object are negative when timing is sensitive to the wirelength change induced by the object movement, and zero when it is not. Therefore, to make this more intuitive, we use their absolute values to calculate the total timing sensitivity.

4 APPLICATION TO TIMING-DRIVEN DETAILED PLACEMENT

In this section, we apply the timing sensitivity metric derived in Section 3.3 to timing-driven detailed placement. We first present the timing-driven detailed placement flow enhanced with our proposed timing sensitivity framework, and then present specific algorithms for each technique in the flow.

4.1 Flow of Timing-Sensitivity Enhanced Timing-Driven Detailed Placement

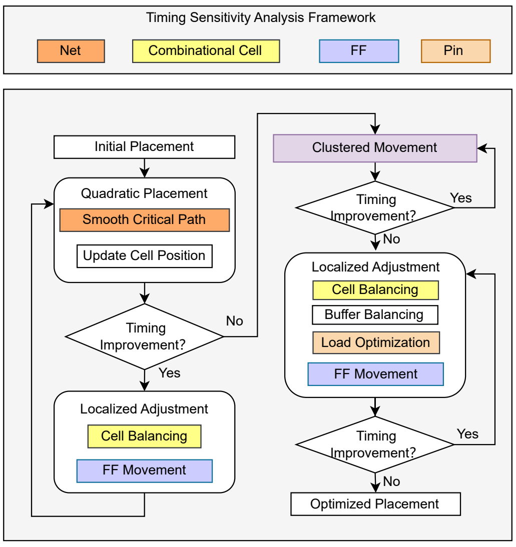

Fig. 6. Flow of timing-sensitivity enhanced timing-driven detailed placement.

Fig. 6 provides an overview of the timing-sensitivity enhanced timing-driven detailed placement flow. The flow is based on and aligns with Rsyn [6, 7], while incorporating significant enhancements. Specifically, we refine the techniques of smoothing critical paths, cell balancing, and load optimization in Rsyn by leveraging our proposed timing sensitivity analysis. In addition, we incorporate our developed FF movement strategy into the flow based on the timing sensitivity. Each technique with applied timing sensitivity analysis is distinctly highlighted in color for clarity in the figure.

In Fig. 6, initially we take the current placement as input and apply the quadratic technique improved from [7] to smooth timing-critical path, until no further timing improvement is observed. Meanwhile, cell balancing and FF movement are performed once when timing is improved via the quadratic placement technique. Then, we execute clustered movement until no additional timing improvements are observed. Finally, we iteratively and alternately apply the movement techniques for combinational cells (cell balancing, buffer balancing, and load optimization) and FFs until no further timing enhancements are observed, culminating in an optimized placement. These techniques are detailed in the following subsections.

4.2 Timing-Sensitivity Enhanced Combinational Cell and FF Movements

The combinational cell movement techniques proposed in Rsyn [6, 7] have yielded excellent results for timing-driven detailed placement. However, as demonstrated in Subsection 3.1, relying solely on criticality cannot accurately identify the cells that truly affect the timing result. Moreover, Rsyn ignores the movement of FFs when optimizing late timing violations. In this subsection, we explain how we specifically apply the timing sensitivity analysis framework in Section 3 to enhance combinational cell movement and move FFs for timing optimization. Details of the techniques are explained below.

Quadratic Placement: Rsyn [7] used the driver's criticality as a metric for net weight and applied quadratic placement to reduce the critical path's net length. However, this weighting may yield suboptimal result as it ignores the effect of net length on timing. To improve this, we calculate the total timing sensitivity $S_{\text{total}}(\text{net})$ for each net and normalize it to replace the criticality parameter in the original weighting formula. The new weight for the net is:

$$w(net) = \frac{\alpha R_{\text{driver}}(1 + S_{\text{total}}(\text{net})/S_{\max})}{k-1}, \tag{13}$$

where α is a constant set to 5, R_{driver} is the resistance of the driver in the net, $S_{\max}$ is the maximum total timing sensitivity value among all nets, k is the number of pins in the net. Assigning higher weights to more sensitive nets would enhance more the timing result due to the wirelength change. The process of the improved quadratic placement is given in Algorithm 1.

Algorithm 1 Quadratic Placement

```
1: Perform static timing analysis;
2: Identify critical paths;
3: Calculate normalized total timing sensitivity for all nets;
4: for each net in netlist do
5:     Net weighting by Eq. (13);
6: Perform critical path smoothing with quadratic placement;
```

Clustered Movement: In Rsyn [6], clustered movement is used for moving combinational cells locally to optimize timing, and avoid suboptimal solution to some extent. The specific cluster formation and clustered movement direction and distance are defined in detail in Rsyn [6]. We directly adopt this technique with additionally including FFs in the generation and movement of clusters. Since the cell movement in the clustered movement technique is usually relatively small, this technique is also effective in handling combinational cells and FFs with only tiny slack values. The process of the improved clustered movement is given in Algorithm 2.

Cell Balancing: In Rsyn [6], the cell balancing technique effectively improves timing by moving critical combinational cells locally. Specifically, each combinational cell in a timing path has a driver cell and a load cell. The cell balancing technique finds a target location that minimizes the local

Algorithm 2 Clustered Movement

```
1: Forming clusters of multiple cells;
2: while timing improved do
3:     Determine moving direction and distance for each cluster;
4:     Move combinational cells and FFs in a cluster to target locations;
```

delay, and then moves the combinational cell to that location. Additionally, Rsyn considers the case that a combinational cell has multiple driver and load cells. This technique is applied to all combinational cells with criticality greater than 0.

However, our sensitivity analysis shows that this technique does not accurately identify the cells that need to be moved. To address this issue, we calculate the total timing sensitivities of all combinational cells and apply the cell balancing technique to those with total timing sensitivities greater than 0, and in the order of descending total timing sensitivity. The process of the improved cell balancing is shown in Algorithm 3.

Algorithm 3 Cell Balancing

```
1: Compute total timing sensitivity for all combinational cells by Eq. (12);
2: Select combinational cells with sensitivity greater than 0;
3: Sort the selected combinational cells in descending order based on their total timing sensitivities;
4: for each cell in the sorted list do
5:     Perform cell balancing;
```

Load Optimization: In Rsyn [6], the load optimization technique improves timing by reducing the load capacitance of critical nets. Specifically, it relocates combinational cells with non-critical pins (i.e., pins with slack > 0) within the critical net to positions near the driver cell of the net. We have already stated the necessity of moving cells and pins determined by their sensitivities. Considering that the timing information of the pins is used several times in this technique, we apply the total timing sensitivity of the pins in this technique as shown in Algorithm 4.

Algorithm 4 Load Optimization

```
1: for each net in netlist do
2:     Compute total timing sensitivity for the source of net by Eq. (12);
3:     if the source's total timing sensitivity $> 0$ then
4:         for each sink in the net do
5:             Compute total timing sensitivity for the sink;
6:             if total timing sensitivity of the sink $> 0$ and slack of the sink $> 0$ then
7:                 Move non-critical receivers associated with the sink close to the driver's position;
```

FF Movement: Rsyn [6] ignores the movement of FFs when optimizing late timing violations, we propose an FF movement technique and incorporate it into Rsyn's flow. Similar to the cell balancing technique, our FF movement technique identifies FFs with the total timing sensitivity greater than 0 and prioritizes them in descending order. This method effectively identifies the FFs that truly impact timing result and addresses the limitation of traditional method [22], which only target timing-critical FFs. We will give the details of this specific method for moving FFs in the next section. The process of FF movement is given in Algorithm 5.

Algorithm 5 FF Movement

```
1: Compute total timing sensitivity for all FFs by Eq. (12);
2: Select FFs with total timing sensitivity greater than 0;
3: Sort the selected FFs in descending order based on their total timing sensitivity;
4: for each FF in the sorted list do
5:     Perform FF Movement;
```

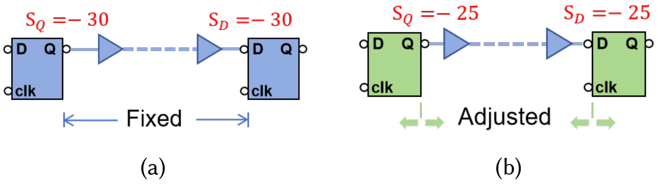

Fig. 7. (a) Combinational cell movement with fixed FFs' locations. (b) Combinational cell movement when FFs' positions are changed.

It is worth noting that, the detailed placement results must satisfy the condition that there is no overlap between cells. Therefore, Rsyn integrates the Jezz [23] legalizer. Jezz employs a nearest-available-space legalization technique, which ensures placement legality by searching for the closest vacant position to the target location of a cell movement each time a cell is relocated. In our detailed placement algorithm, we also use the exact same legalization as Rysn.

5 FF MOVEMENT BASED ON TIMING SENSITIVITY

In this section, we outline the benefit of moving FFs for timing optimization, followed by presenting our FF-moving scheme based on the timing sensitivity analysis framework.

5.1 Benefit of Moving FFs

In timing-driven detailed placement, traditional FF movement strategies [11, 14] are primarily employed to address early timing violations. For late timing violations, many approaches [6, 12] heuristically move combinational cells while keeping FFs' positions unchanged. However, an effective FF movement strategy can directly enhance timing performance, since FFs are starting points and endpoints of timing paths. Moreover, adjusting the position of an FF can optimize the moving range of the combinational cells along the timing path, thereby increasing the possibility of further timing optimization. This case is demonstrated as in Fig. 7.

As shown in Fig. 7, we denote the slack values at the pins as S. Fig. 7(a) indicates that, when the FFs' positions are fixed, the combinational cells on the timing path can only be moved within a fixed range. In Fig. 7(b), when the FFs' positions are adjusted, the slack values on the FFs' pins change, directly improving the timing result. At the same time, the moving range of the combinational cells is also improved, allowing for shorter wirelength between the cells on the critical path, thereby optimizing the timing.

5.2 FF Movement

As demonstrated in Section 3, moving FFs that are not timing-critical may still improve the timing result. Moreover, we have demonstrated in Subsection 3.3.4 that, if the total timing sensitivity of an FF is greater than 0, a position change will inevitably affect the timing result. Hence by leveraging

FF's timing sensitivity, we can effectively identify the FFs that are guaranteed to have an impact on overall timing.

We focus on finding the optimal positions for the FFs along the logic path, as discussed after Theorem 3.12. Firstly, depending on whether the total timing sensitivity values of the D and Q pins are positive, we classify all FFs into four cases:

Case 1). $S_{\text{total}}(D) > 0$, $S_{\text{total}}(Q) > 0$;
Case 2). $S_{\text{total}}(D) > 0$, $S_{\text{total}}(Q) = 0$;
Case 3). $S_{\text{total}}(D) = 0$, $S_{\text{total}}(Q) > 0$;
Case 4). $S_{\text{total}}(D) = 0$, $S_{\text{total}}(Q) = 0$.

Next, depending on the cases of FFs, we develop different FF movement strategies. For Case 1), as shown in Fig. 8, $S_{\text{total}}(D) > 0$ indicates that the position change of the D pin will impact the timing result, meaning there exists a timing violation in the input net n_1 of the FF (i.e., net n_1 is a critical net). Similarly, since $S_{\text{total}}(Q) > 0$, there exists a timing violation in the output net n_2 of the FF (i.e., net n_2 is also a critical net). Then, we search for the driver cell in the input net n_1 and the most timing-critical load cell (colored in red) in the output net n_2.

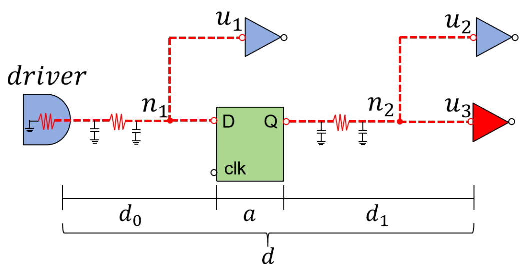

Fig. 8. An example of FF with total timing sensitivity values of both the D pin and the Q pin greater than zero. The red dotted line indicates the critical net. The red cell indicates the most timing-critical load cell of net n_2.

Since this FF has timing violations at both its input and output sides, we use the same idea as the buffer balancing technique in Rsyn to compute the local delay it affects, and determine the optimal position for such FF by minimizing this local delay. Similar to Eq. (7), we obtain the local delay affected by FF as:

$$D = R_0(C_D + d_0c) + d_0r(C_D + \frac{d_0c}{2}) + p_0 + R_1(C_3 + d_1c) + d_1r(C_3 + \frac{d_1c}{2}) + p_f, \tag{14}$$

where R_0 is the resistance of the FF's driver, C_D is the D pin's load capacitance, d_0 is the wirelength from the driver to the FF, r is the wire resistance per unit wirelength, c is the wire capacitance per unit wirelength, R_1 is the FF's resistance, d_1 is the wirelength from the FF to its most timing-critical load cell, C_3 is the load capacitance at the input sink pin u_3. p_0 and p_f are parasitic delays of the driver and FF, respectively.

Let $d = d_0 + a + d_1$, where a is the distance between FF's D pin and Q pin. The minimum D can be determined by setting $\frac{\partial D}{\partial d_0} = 0$, which leads to

$$d_0 = \frac{c(R_1 - R_0) + r[C_3 - C_D + c(d - a)]}{2rc}. \tag{15}$$

Eq. (15) gives the optimal distance between the FF and its driver cell. Then, similar to the buffer balancing in Rsyn [6], we place the FF on the interconnect path between its driver cell and the

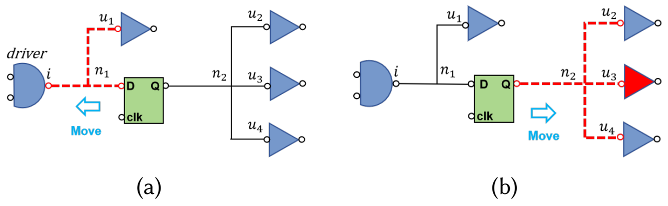

Fig. 9. (a) A simple example on how to move the FF with only the D pin's total timing sensitivity value greater than 0. The red dotted line indicates critical net. (b) A simple example on how to move the FF with only the Q pin's total timing sensitivity value greater than 0. The red cell indicates the most timing-critical load cell in the net n_2.

most timing-critical load cell. Specifically, the position of the FF can be set as follows:

$$P_{FF} = P_d + \frac{d_0}{d} \cdot (P_s - P_d), \tag{16}$$

where P_{FF} is the new FF position, P_d is the driver cell's position and P_s is the most timing-critical load cell's position.

Note that FFs may not always be placed at their calculated optimal locations due to possible overlaps. Our FF movement technique integrates with the legalization process from Rsyn [6]. Specifically, if an overlap occurs after moving an FF to its optimal location, the FF will be placed at the available empty position closest to the optimal location. By combining these two approaches, we can effectively identify the optimal locations for such FFs while ensuring placement legality.

For an FF in Case 2) or Case 3), the FF only has a positive total timing sensitivity value at either the D pin or the Q pin, which means that only the FF's input net or output net is critical. We adopt a more intuitive movement approach. For Case 2), timing violation exists only in the input net of the FF, we move the FF to the available empty position closest to the driver cell in the net n_1 associated with the D pin, as shown in Fig. 9(a). This adjustment reduces the wirelength of the net n_1, thereby optimizing the timing violation in the input net. Similarly, for Case 3), we optimize timing by moving the FF to the available empty position closest to the most timing-critical load cell of the net associated with the Q pin, as in Fig. 9(b).

For Case 4), Theorems 3.11 and 3.12 confirm that these FFs have a total timing sensitivity value of 0, which does not impact timing optimization. Therefore, the locations of such FFs will not be updated.

Note that, the above moving methods may cause over-movement of an FF, if the absolute slacks of the D pin or Q pin in the FF are too small. For example, when the slack of an FF's D pin is -1 and the slack of the Q pin is +1, after moving, the slack of the D pin becomes +2, but the slack of the Q pin becomes -2. However, this case is rare in our approach, since such FFs have been optimized by our clustered movement technique.

6 EXPERIMENTAL RESULTS

In this section, we conduct experiments to validate the effectiveness of the timing-driven detailed placement flow (Fig. 6) enhanced with our proposed timing sensitivity framework as well as the novel FF movement scheme.

6.1 Experimental Setup

We implement our timing-driven detailed placement flow based on the open-source framework Rsyn [6, 7] in single-thread mode. Rsyn has state-of-the-art timing-driven detailed placement techniques, and provides all the necessary functionality, as well as the required incremental timing analysis and legalization tool. The ICCAD 2015 Contest benchmarks [13] are used for a comprehensive evaluation and comparison. The statistics of these benchmarks are presented in Table 1.

Table 1. Statistics of ICCAD 2015 contest benchmarks [13].

Benchmark	#Cells	#Nets	#Pins	#Rows
superblue1	1209716	1215710	3767494	1829
superblue3	1213253	1224979	3905321	1840
superblue4	795645	802513	2497940	1840
superblue5	1086888	1100825	3246878	2528
superblue7	1931639	1933945	6372094	3163
superblue10	1876103	1898119	5560506	3437
superblue16	981559	999902	3013268	1788
superblue18	768068	771542	2559143	1788

The ICCAD 2015 contest provides maximum cell displacement constraints for its initial solution. However, in this work, we also need to conduct experiments without displacement constraints. Therefore, in all of the following experiments, we refer to Rsyn with displacement constraints as Rsyn [6, 7], Rsyn without displacement constraints as Rsyn†, our method with displacement constraints as Ours, and our method without displacement constraints as Ours†. [1]

All experiments are performed on a 64-bit Linux machine with an Intel Core CPU at 2.4 GHz and 64 GB RAM. We report the experimental results for TNS, WNS, Steiner tree wirelength (StWL), and runtime (RT). All results reported are tested using the evaluation script provided by the competition organizers, which uses UI-Timer for the STA.

6.2 Effectiveness of Our Timing-Driven Detailed Placer

To evaluate the effectiveness of our timing-driven detailed placement, we conduct experiments on two sets of benchmarks: the ICCAD 2015 contest benchmarks, and the global placement results after performing timing-driven global placement and legalization with DREAMPlace 4.0 [17] (denoted by "DREAMPlace 4.0_GP"). DREAMPlace 4.0_GP [17] is an open-source tool of timing-driven global placement that can be applied on the ICCAD 2015 Contest benchmarks [13].

6.2.1 Evaluation on ICCAD 2015 Benchmarks. On the ICCAD 2015 contest benchmarks, we compare our method with several state-of-the-art timing-driven detailed placement approaches. Specifically, we evaluate the performance of our algorithm against the initial placement of the ICCAD 2015 Contest benchmarks, the 1st place solution from the ICCAD 2015 Contest [13], TCFF [22], Rsyn [6, 7]. Note that we present the results of 1st place, TCFF [22] and Rsyn [6, 7] under the long-displacement constraints of the ICCAD 2015 Contest [13], as they achieved the best performance in this way.

Table 2 gives the results of TNS, WNS, StWL, and RT. The results of TCFF are directly cited from [22] in which StWL is not reported. Compared to initial placement, 1st place, TCFF, and Rsyn, our proposed algorithm achieves significant improvements in both the TNS and WNS. Specifically, the TNS improvements are 109.5%, 34.8%, 21.0% and 8.8%, respectively, while the WNS improvements are 27.2%, 15.3%, 11.8% and 1.8%, respectively. At the same time, the StWL results remain comparable, and the runtime performance is also satisfactory.

[1]The symbol † indicates that the method does not apply cell displacement constraints, while the absence of the symbol indicates that long displacement constraints in the ICCAD 2015 Contest are applied.

Table 2. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among initial placement provided by ICCAD 2015 contest, 1st Place, TCFF [22], Rsyn [6, 7] and Ours. The best WNS and TNS results are bold-faced.

Benchmark	**Initial Placement**			**1st Place**				**TCFF [22]**			**Rsyn [6, 7]**				**Ours**			
	WNS	TNS	StWL	WNS	TNS	StWL	RT	WNS	TNS	RT	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-4.98	-4.6	9.59	-4.57	-3.51	9.61	1917.6	-4.42	-3.24	2580	-4.246	-3.307	9.92	111.31	**-4.157**	**-3.115**	9.858	184.63
superblue3	-10.15	-15.03	11.43	-8.71	-11.6	11.46	1600.8	-8.27	-8.82	2940	-6.782	-7.902	11.6	111.27	**-6.579**	**-7.431**	11.58	120.45
superblue4	-6.22	-34.77	7.15	-5.76	-24.65	7.16	1117.2	-5.6	-23.1	3900	-4.602	-22.93	7.55	81.84	**-4.445**	**-19.84**	7.621	230.57
superblue5	-25.7	-69.65	10.75	-24.29	-58.42	10.78	1519.8	-24.7	-63.28	1200	-21.06	-54.38	11.02	57.90	**-20.6**	**-52.23**	11.01	158.54
superblue7	**-15.22**	-18.57	14.01	**-15.22**	-15.11	14.03	3186.6	**-15.22**	-14.54	2460	**-15.22**	-11.65	14.24	179.56	**-15.22**	**-11.18**	14.19	182.79
superblue10	-16.49	-331.5	20.53	-16.08	-315.2	20.55	2245.8	-16.13	-294.5	5280	-15.55	-278.6	21.07	303.13	**-15.41**	**-273.5**	21.22	577.29
superblue16	-4.58	-7.76	9.33	-3.85	-2.66	9.37	1345.2	-3.35	-2.1	2460	-3.161	-1.735	9.55	102.02	**-3.118**	**-1.408**	9.538	108.15
superblue18	-4.55	-10.35	5.77	-3.82	-7.76	5.78	951.6	-3.8	-7.01	1920	-3.684	-6.305	5.9	77.94	**-3.645**	**-5.783**	5.879	85.14
Avg. Ratio	1.272	2.095	0.974	1.153	1.348	0.976	10.38	1.118	1.210	16.35	1.018	1.088	1.000	0.702	**1.000**	**1.000**	1.000	1.000

Note: Rsyn [6, 7] indicates that these two works together constitute the Rsyn.

Table 3. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among DREAMPlace 4.0_DP† [17], Rsyn [6, 7], Rsyn† and our timing-driven detailed placer on the global placement results of DREAMPlace 4.0_GP [17]. The best WNS and TNS results are bold-faced.

Benchmark	**DREAMPlace 4.0_GP [17] + DREAMPlace 4.0_DP†**				**DREAMPlace 4.0_GP [17] + Rsyn [6, 7]**				**DREAMPlace 4.0_GP [17] + Rsyn†**				**DREAMPlace 4.0_GP [17] + Ours†**			
	WNS	TNS	StWL	RT*	WNS	TNS	StWL	RT*	WNS	TNS	StWL	RT*	WNS	TNS	StWL	RT*
superblue1	-13.06	-67.04	9.12	1245.4	-10.88	-19.70	9.931	109.3	-5.628	-6.206	10.08	212.3	**-5.507**	**-5.375**	8.989	198.17
superblue3	-16.26	-52.28	10.00	914.0	-12.79	-22.5	10.89	280.6	-6.148	-8.086	10.57	155.26	**-5.872**	**-6.093**	9.838	113.49
superblue4	-11.06	-136.17	6.89	1429.8	-8.404	-63.78	7.701	108.9	**-4.31**	-32.20	7.683	129.01	-4.50	**-29.75**	7.015	157.17
superblue5	-23.70	-94.00	11.11	1147.5	-22.08	-99.62	10.79	131.3	-15.14	-34.98	11.86	191.2	**-14.77**	**-28.24**	11.16	157.07
superblue7	**-15.22**	-58.37	12.64	1576.2	**-15.22**	-22.28	12.82	223.7	**-15.22**	-15.9	13.1	248.8	**-15.22**	**-11.7**	12.36	196.24
superblue10	-25.38	-705.80	26.15	3196.6	-21.46	-481.1	26.65	290.7	-16.58	-332.9	26.59	402.5	**-16.28**	**-325.3**	25.74	562.9
superblue16	-11.28	-118.70	9.49	1419.5	-6.882	-13.42	10.32	159.6	-5.676	-5.261	9.861	80.1	**-2.824**	**-2.537**	9.681	148.25
superblue18	-11.69	-43.49	5.21	800.2	-7.324	-20.24	5.551	83.6	-3.839	-6.927	5.281	62.87	**-3.627**	**-6.541**	5.138	78.83
Avg. Ratio	2.372	11.20	1.005	8.022	1.786	3.101	1.061	1.430	1.142	1.290	1.061	0.975	**1.000**	**1.000**	1.000	1.000

* Runtime of DREAMPlace 4.0_GP [17] is not included in the metric RT.

6.2.2 Evaluation on the Global Placement Results of DREAMPlace 4.0. The timing-driven global placement solutions provided by DREAMPlace 4.0 [17] do not require displacement constraints. We therefore verify the effectiveness of our detailed placement method without displacement constraints on these solutions. Specifically, on the global placement results of DREAMPlace 4.0_GP [17], we separately perform: 1) our timing-driven detailed placement which is without cell displacement constraints (denoted as "DREAMPlace 4.0_GP + Ours†"), 2) Rsyn [6, 7] without cell displacement constraints (denoted as "DREAMPlace 4.0_GP + Rsyn†"), and 3) Rsyn with cell displacement constraints (denoted as "DREAMPlace 4.0_GP + Rsyn [6, 7]"). Timing-driven detailed placement is also available in DREAMPlace 4.0 (denoted by "DREAMPlace 4.0_DP†"). Table 3 lists the results of DREAMPlace 4.0_GP + DREAMPlace 4.0_DP†, DREAMPlace 4.0_GP + Rsyn, DREAMPlace 4.0_GP + Rsyn† and DREAMPlace 4.0_GP + Ours†.

The results in Table 3 show that, compared to DREAMPlace 4.0_GP + DREAMPlace 4.0_DP† [17] and DREAMPlace 4.0_GP [17] + Rsyn [6, 7], on average our timing results are much better meanwhile the StWL results are comparable or shorter. Moreover, compared to DREAMPlace 4.0_GP + Rsyn†, on average our approach achieves improvements of TNS by 29.3%, WNS by 14.3%, and StWL by 6.1%.

6.2.3 Runtime Comparison and Analysis. As shown in Tables 2 and 3, our runtime is comparable to that of Rsyn on the benchmarks. This can be attributed to two key reasons. On the one hand, the sensitivity update for combinational cells incurs a runtime cost due to their large quantity. However, since the sensitivity values of these cells are slightly affected by local movements, we update their sensitivities only during the first iteration, where combinational cell movement alternates with FF movement. This approach significantly reduces runtime while preserving excellent timing performance.

On the other hand, since the number of FFs is relatively small, we update FFs' sensitivities at each iteration. It is worth noting that this approach does not increase the runtime. In fact, the alternating movement of FFs and combinational cells enables the cells to reach their optimal positions much faster, significantly reducing the total number of iterations and thus decreasing the overall runtime. This result can be further demonstrated in the ablation experiments of Subsection 6.4.

Meanwhile, we also analyze the proportion of runtime contributed by each stage. Using Benchmark 1 from Table 2, we measure the runtime fraction for each stage and present the corresponding pie chart. Fig. 10 shows the breakdown of total runtime for the main stages of our method, where "input" denotes the stage of inputting benchmark information. It can be seen that quadratic placement, cell balancing, and load optimization dominate the runtime, contributing 30%, 26%, and 19%, respectively. In contrast, flip-flop movement, clustered movement, and buffer balancing consume relatively little time, accounting for only 4%, 2%, and 1%, respectively.

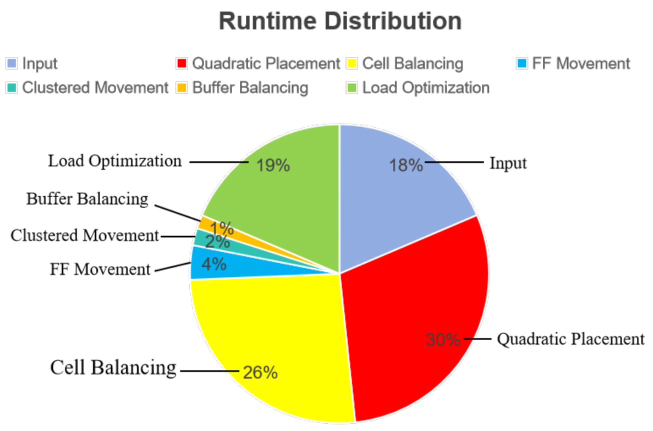

Fig. 10. Breakdown of runtime for each stage of our method on Benchmark 1.

6.2.4 Cells That Are Timing-Sensitive But Not Timing-Critical. It is claimed in Subsection 3.1 that, the metric of timing sensitivity can be used to identify cells that are timing-sensitive in addition to being timing-critical, thereby overcoming the limitation of traditional methods (e.g., Rsyn [6, 7]) that focus only on timing-critical cells. To investigate whether timing-sensitive yet non-critical cells account for a significant proportion among all timing-sensitive cells, we evaluate the metric on the initial placements of the ICCAD 2015 contest benchmarks. Specifically, for each placement we calculate the number of cells with total timing sensitivity greater than 0 (denoted by "Sens."), the number of cells with timing-criticality greater than 0 (denoted by "Crit."), and the number of cells with timing-criticality equal to 0 but total timing sensitivity greater than 0 (denoted by "SensOnly"). Additionally, we calculate the ratio of the number of timing-sensitive but not timing-critical cells to all timing-sensitive cells (denoted by "SensOnly/Sens."). The results are given in Table 4. From the table, it can be seen that among all the timing-sensitive cells, the timing non-critical cells account for a significant portion.

Table 4. The Number of timing-sensitive and the number of timing-critical cells in ICCAD 2015 contest benchmarks [13].

Benchmark	Sens.	Crit.	SensOnly	SensOnly/Sens.
superblue1	66468	37393	29115	43.8%
superblue3	25437	16075	9940	39.1%
superblue4	129372	86686	42754	33.0%
superblue5	58975	39873	19171	32.5%
superblue7	31821	20921	10974	34.5%
superblue10	258752	200329	58467	22.6%
superblue16	37370	25170	12233	32.7%
superblue18	21081	12607	8510	40.4%
Avg.	78659	54882	23896	34.8%

6.3 Discussion of cell displacement constraints

It is worth noting that the purpose of setting maximum displacement constraints for cells in the ICCAD 2015 Contest is to prevent detailed placement from causing significant disruption to the structure of the global placement solution, thereby preserving the optimization metrics of the global placement. However, while displacement constraints are generally enforced, there are specific scenarios where tools or designers may selectively relax this constraint [1, 25]. For example, during critical path optimization, if a critical path exhibits severe timing violations that cannot be resolved through minor local adjustments, allowing certain key cells along the path to move more freely may be the preferred strategy. Consequently, we believe that appropriately relaxing displacement constraints while minimizing disruption to the placement solution represents a more effective approach.

Accordingly, using the initial solution provided by the ICCAD 2015 Contest, we conduct two comparative experiments to evaluate our method. The first applies our detailed placement algorithm with at most 0.1% of cells exceeding displacement constraints, in order to validate its optimization efficacy. The second applies the algorithm without any displacement constraints, to assess both the improvements achieved and the extent of disruption introduced to the placement solution.

6.3.1 Evaluation on the ICCAD 2015 benchmarks with ≤0.1% of cells exceeding displacement constraints. To more effectively manage displacement constraints, we conduct the following experiment: we evaluate the effectiveness of our method while ensuring that the number of cells exceeding displacement constraints remains below 0.1% of the total cell count. This approach ensures that cell movement does not significantly disrupt the structural integrity of the global placement solution, while also appropriately mitigating overly restrictive aspects of the displacement constraints. Note that the setting of this ratio is fully controllable.

In practical implementation, we adopt the following rules to control the ratio: 1. Reposition cells with minor displacement constraint violation ($\leq 1.1 \cdot D_{max}$, where D_{max} represents maximum displacement) back within the permissible boundary; 2. Allow movement of cells with significant displacement ($> 1.1 \cdot D_{max}$). Once the total proportion of violating cells reaches 0.1%, all subsequent movements must strictly adhere to the displacement constraints.

The experimental results are shown in Table 5, where we compare them to the best results achievable using Rsyn. On average, our approach achieves improvements of 2.4% in WNS, 9.6% in TNS, and 0.8% in StWL compared to Rsyn.

6.3.2 Evaluation on the ICCAD 2015 benchmarks without displacement constraints. Under the initial solution provided by ICCAD 2015, we compare the results of our method without displacement

Table 5. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among initial placement provided by ICCAD 2015 contest, Rsyn and Ours′. The best WNS and TNS results are bold-faced.

Benchmark	**Initial Placement**			**Rsyn**				**Ours′**			
	WNS	TNS	StWL	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-4.98	-4.6	9.59	-4.246	-3.307	9.92	111.31	**-3.126**	**-4.172**	9.914	159.73
superblue3	-10.15	-15.03	11.43	-6.782	-7.902	11.6	111.27	**-6.543**	**-7.295**	11.61	123.03
superblue4	-6.22	-34.77	7.15	-4.602	-22.93	7.55	81.84	**-4.394**	**-20.66**	7.571	232.29
superblue5	-25.7	-69.65	10.75	-21.06	-54.38	11.02	57.90	**-20.05**	**-50.33**	10.31	178.8
superblue7	**-15.22**	-18.57	14.01	**-15.22**	-11.65	14.24	179.56	**-15.22**	**-12.25**	14.20	175.09
superblue10	-16.49	-331.5	20.53	**-15.55**	-278.6	21.07	303.13	-15.82	**-274.5**	21.22	555.72
superblue16	-4.58	-7.76	9.33	-3.161	-1.735	9.55	102.02	**-3.038**	**-1.256**	9.565	106.48
superblue18	-4.55	-10.35	5.77	-3.684	-6.305	5.9	77.94	**-3.623**	**-5.804**	5.885	88.95
Avg. Ratio	1.281	2.161	0.981	1.024	1.096	1.008	0.710	**1.000**	**1.000**	1.000	1.000

Note: Ours′ denotes our method that controls the number of cells exceeding the displacement constraint within 0.1%.

Table 6. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among initial placement provided by ICCAD 2015 contest, Rsyn, Ours, Rsyn† and Ours†. The best WNS and TNS results are bold-faced.

Benchmark	**Initial Placement**			**Rsyn**				**Ours**				**Rsyn†**				**Ours†**			
	WNS	TNS	StWL	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-4.98	-4.6	9.59	-4.246	-3.307	9.92	111.31	**-4.157**	-3.115	9.858	184.63	-4.279	-3.319	9.932	142.51	-4.167	**-3.100**	9.887	161.8
superblue3	-10.15	-15.03	11.43	-6.782	-7.902	11.6	111.27	-6.579	-7.431	11.58	120.45	-8.075	-8.859	11.61	135.43	**-6.543**	**-7.295**	11.61	117.38
superblue4	-6.22	-34.77	7.15	-4.602	-22.93	7.55	81.84	-4.445	-19.84	7.621	230.57	-4.955	-21.27	7.582	172.71	**-4.303**	**-19.51**	7.628	206.09
superblue5	-25.7	-69.65	10.75	-21.06	-54.38	11.02	57.90	-20.6	-52.23	11.01	158.54	-24.24	-64.40	10.94	162.12	**-19.99**	**-49.44**	10.54	137.97
superblue7	**-15.22**	-18.57	14.01	**-15.22**	-11.65	14.24	179.56	**-15.22**	-11.18	14.19	182.79	**-15.22**	-13.09	14.22	227.94	**-15.22**	**-10.62**	14.20	185.88
superblue10	-16.49	-331.5	20.53	-15.55	-278.6	21.07	303.13	-15.41	-273.5	21.22	577.29	**-14.94**	-280.8	21.09	333.36	-15.43	**-269.5**	21.31	508.53
superblue16	-4.58	-7.76	9.33	-3.161	-1.735	9.55	102.02	-3.118	-1.408	9.538	108.15	-2.983	-1.474	9.577	136.26	**-2.981**	**-1.266**	9.557	110.27
superblue18	-4.55	-10.35	5.77	-3.684	-6.305	5.9	77.94	-3.645	-5.783	5.879	85.14	-3.656	-6.442	5.908	101.76	**-3.617**	**-5.703**	5.884	77.89
Avg. Ratio	1.292	2.207	0.977	1.033	1.129	1.004	0.743	1.015	1.036	1.004	1.078	1.075	1.156	1.004	1.059	**1.000**	**1.000**	1.000	1.000

constraints against those of Rsyn. The results in Table 6 show that, compared to Rsyn†, on average our approach achieves improvements of WNS by 7.5%, TNS by 15.6%, and StWL by 6.1%. Additionally, Table 6 also presents results for Rsyn [6, 7] and our method with/without displacement constraints for comparison. It can be observed that our method achieves better results than Rsyn regardless of displacement constraints. Although our method without displacement constraints yields slightly inferior WNS on two benchmarks compared to the constrained version, it consistently achieves the best composite timing performance. This is fundamentally because our definition of total timing sensitivity as a weighted sum (Equation (12)) implies that the algorithm optimizes the composite objective $\gamma \cdot \text{TNS} + (1 - \gamma) \cdot \text{WNS}$, rather than WNS exclusively.

To demonstrate that our timing-driven detailed placement without displacement constraints does not compromise routability, we leverage Rudy [28], a widely adopted routing congestion evaluation tool, to evaluate the routability of the initial placement, the detailed placement results after applying Rsyn, and those by our method without displacement constraints. Rudy [28] efficiently evaluates routing congestion by analyzing each net in the spatial dimension. The evaluation results of metrics routing_overflow and pin_overflow are reported in Table 7. The results show that even without displacement constraints, our method maintains a congestion level very close to that of Rsyn with displacement constraints.

The reason is that, in our approach, although some cells exceed the maximum displacement, the number of such cells is small. Many of the techniques in our approach, such as cell balancing, buffer balancing, load optimization, and FF movement, implicitly limit a cell to move to the position

Table 7. Comparison of routing overflow and pin overflow among initial placement provided by ICCAD 2015 contest, and the results by Rsyn [6, 7], Rsyn†, Ours and Ours†.

Benchmark	Initial Placement		Rsyn [6, 7]		Rsyn†		Ours		Ours†	
	routing _overflow	pin _overflow	routing _overflow	pin _overflow	routing _overflow	pin _overflow	routing _overflow	pin _overflow	routing _overflow	pin _overflow
superblue1	4.03E-02	3.44E-07	4.69E-02	3.44E-07	4.69E-02	3.44E-07	4.80E-02	3.44E-07	4.60E-02	3.44E-07
superblue3	1.38E-02	2.29E-06	1.49E-02	1.12E-05	1.50E-02	1.20E-05	1.47E-02	9.28E-06	1.47E-02	7.22E-06
superblue4	2.84E-02	6.42E-03	4.27E-02	6.08E-03	4.19E-02	6.05E-03	4.39E-02	6.18E-03	4.46E-02	6.36E-03
superblue5	1.56E-02	1.22E-06	2.09E-02	1.17E-06	1.08E-02	1.17E-06	2.10E-02	1.24E-06	2.16E-02	1.24E-06
superblue7	1.62E-02	5.88E-04	1.62E-02	5.73E-04	1.61E-02	5.66E-04	1.62E-02	5.99E-04	1.62E-02	6.01E-04
superblue10	1.19E-01	4.05E-06	1.21E-01	3.87E-06	1.20E-01	3.78E-06	1.22E-01	3.81E-06	1.23E-01	4.29E-06
superblue16	2.15E-02	7.83E-05	2.23E-02	4.53E-05	2.39E-02	4.38E-05	2.21E-02	4.45E-05	2.19E-02	4.34E-05
superblue18	1.28E-02	1.88E-05	1.35E-02	1.56E-05	1.37E-02	1.56E-05	1.37E-02	1.88E-05	1.41E-02	1.93E-05
Avg. Ratio	0.881	1.002	0.990	1.020	0.982	1.024	0.998	1.017	**1.000**	**1.000**

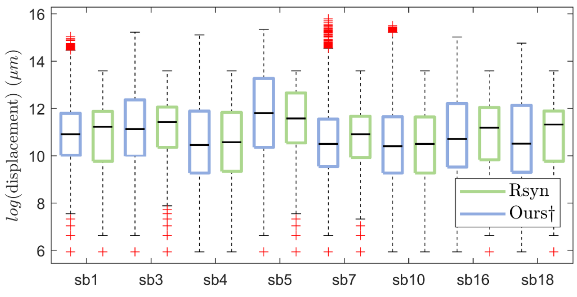

Fig. 11. Displacement distribution comparison of Rsyn [6, 7] and Ours†.

between their driver and load cells, effectively preventing excessive displacement. Therefore, most cells end up with a small total displacement, ensuring the overall quality of detailed placement.

To further observe the displacement phenomenon of our detailed placement method, we generate in Fig. 11 a box plot to visualize the distribution of cell displacement distance of Ours†, and that of Rsyn with displacement constraints. The box plot presents the maximum, minimum, median, and quartiles of the data. Specifically, the top and bottom edges of a box represent the third quartile (Q3) and first quartile (Q1), respectively; while the horizontal line within the box denotes the median (Q2), and the topmost and bottommost horizontal lines represent the maximum and minimum displacement values. The results show that, although some individual cells in our method exhibit large displacements, the overall displacement distribution closely resembles that of Rsyn with displacement constraints, showing no significant anomalies.

6.4 Ablation Experiment

Our timing-driven detailed placement approach contains two parts: the combinational cell movement technique in Rsyn [6, 7] enhanced with the proposed timing sensitivity analysis framework, and our newly introduced FF movement technique. In this subsection, we conduct ablation experiments to demonstrate the effectiveness of each. Since Rsyn† gives more stable experimental results than Rsyn [6, 7] in Tables 3 and 2, it is adopted for comparisons.

The ablation experiments are performed on the ICCAD 2015 contest benchmarks. Our timing-driven detailed placer excluding the FF movement technique is denoted as “Ours†_c-cell_move",

Table 8. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among Rsyn[†] and Ours[†]_c-cell_move on the initial placement provided by ICCAD 2015 contest [13].

Benchmark	**Rsyn[†]**				**Ours[†]_c-cell_move**			
	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-4.279	-3.319	9.932	142.51	**-4.249**	**-3.192**	9.907	167.8
superblue3	-8.075	-8.859	11.61	135.43	**-6.587**	**-7.490**	11.59	100.65
superblue4	-4.955	-21.27	7.582	172.71	**-4.197**	**-20.41**	7.585	205.28
superblue5	-24.24	-64.40	10.94	162.12	**-19.87**	**-51.03**	11.01	127.29
superblue7	**-15.22**	-13.09	14.22	227.94	**-15.22**	**-10.78**	14.18	153.66
superblue10	**-14.94**	-280.8	21.09	333.36	-15.48	**-271.7**	21.2	470.56
superblue16	**-2.983**	-1.474	9.577	136.26	-3.059	**-1.343**	9.541	88.93
superblue18	-3.656	-6.442	5.908	101.76	**-3.629**	**-6.065**	5.874	79.18
Avg. Ratio	1.073	1.117	1.001	1.165	**1.000**	**1.000**	1.000	1.000

Table 9. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7\mu$m) and RT (seconds) among Rsyn[†] and Ours[†]_FF_move on the initial placement provided by ICCAD 2015 contest [13].

Benchmark	**Rsyn[†]**				**Ours[†]_FF_move**			
	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-4.279	-3.319	9.932	142.51	**-4.250**	**-3.296**	9.930	84.42
superblue3	-8.075	-8.859	11.61	135.43	**-6.757**	**-7.754**	11.64	96.85
superblue4	-4.955	-21.27	7.582	172.71	**-4.200**	**-20.31**	7.635	113.78
superblue5	-24.24	-64.40	10.94	162.12	**-20.15**	**-50.78**	11.06	91.28
superblue7	**-15.22**	-13.09	14.22	227.94	**-15.22**	**-10.89**	14.25	142.61
superblue10	-14.94	-280.8	21.09	333.36	**-14.41**	**-272**	21.19	253.65
superblue16	**-2.983**	-1.474	9.577	136.26	-3.127	**-1.458**	9.582	68.45
superblue18	-3.656	-6.442	5.908	101.76	**-3.620**	**-6.107**	5.924	70.69
Avg. Ratio	1.073	1.096	0.996	1.590	**1.000**	**1.000**	1.000	1.000

and Rsyn[†] incorporated with our proposed FF movement technique is denoted as “Ours[†]_FF_move". The experimental results are presented in Tables 8 and 9. From the average results in the tables, we can observe that both of our techniques effectively improve timing result.

6.5 Additional Experiment

Recently, the work in [27] outperforms DREAMPlace 4.0 and achieves better timing-driven global placement results, and has been open-sourced. We also evaluate our detailed placement method on [27]’s global placement results.

Based on the global placement results of Shi [27], we perform timing-driven detailed placement using the following three approaches: our method without cell displacement constraints (denoted as “Shi_GP + Ours[†]”), Rsyn [6, 7] without cell displacement constraints (denoted as “Shi_GP + Rsyn[†]”), and Rsyn with cell displacement constraints (denoted as “Shi_GP + Rsyn [6, 7]”). The global placement results of [27] are also given (denoted by “Shi_GP”). Table 10 lists the results of Shi_GP, Shi_GP + Rsyn, Shi_GP + Rsyn[†] and Shi_GP + Ours[†].

The results in Table 10 show that, our approach significantly improves the timing results compared to Shi_GP [27] and Shi_GP [27] + Rsyn [6, 7], improving TNS by an average of 355.9% and 162.1%, and WNS by 80.2% and 42.8%, respectively. Moreover, compared to Shi_GP + Rsyn[†], on average our approach achieves improvements of TNS by 34.0%, WNS by 6.4%, and StWL by 6.7% respectively, without compromising the runtime.

Table 10. Comparison of WNS (10^3ps), TNS (10^5ps), StWL ($10^7 \mu$m) and RT (seconds) among Shi_GP [27], Rsyn [6, 7], Rsyn† and our timing-driven detailed placer on the global placement results of Shi [27]. The best WNS and TNS results are bold-faced.

Benchmark	**Shi_GP [27]**			**Shi_GP [27] + Rsyn [6, 7]**				**Shi_GP [27] + Rsyn†**				**Shi_GP [27] + Ours†**			
	WNS	TNS	StWL	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT	WNS	TNS	StWL	RT
superblue1	-7.702	-15.92	8.442	-5.748	-5.603	9.327	113.89	-5.823	-6.086	9.726	325.68	**-5.227**	**-4.443**	8.624	180.12
superblue3	-12.04	-21.18	9.327	-6.716	-7.675	10.7	190.74	-6.636	-7.331	10.43	156.47	**-6.38**	**-6.567**	9.457	132.88
superblue4	-10.47	-102.7	6.393	-9.268	-90.7	6.467	166.52	-4.73	**-31.92**	7.496	110.4	**-4.379**	-33.24	6.73	261.13
superblue5	-24.71	-64.19	9.574	-24.71	-64.61	9.664	241.07	-17.01	-27.48	10.89	188.54	**-16.7**	**-24.41**	9.916	232.57
superblue7	**-15.22**	-38.84	12.23	**-15.22**	-13.66	13.1	270.81	**-15.22**	-12.41	12.55	314.92	**-15.22**	**-10.79**	12.45	292.27
superblue10	-23.32	-551.4	19.01	-21.46	-359.2	20.07	411.18	-13.95	-296.7	19.48	512.8	**-13.62**	**-277**	18.79	557.71
superblue16	-8.249	-22.55	9.434	-6.882	-13.42	10.32	157.95	-3.802	-4.004	9.829	86.94	**-3.032**	**-1.44**	9.69	148.4
superblue18	-6.705	-16.6	4.805	-4.063	-7.869	5.161	97.2	-3.731	-7.055	5.33	98.78	**-3.707**	**-6.148**	5.15	87.51
Avg. Ratio	1.802	4.559	0.973	1.428	2.621	1.042	0.944	1.067	1.340	1.067	0.987	**1.000**	**1.000**	1.000	1.000

Finally, it should be remarked that, although the experiments in Tables 2, 3 and 10 begin from quite different global placement results, the timing results after our timing-driven detailed placement do not differ much.

7 CONCLUSION

This paper has presented a comprehensive timing sensitivity analysis framework that can precisely estimate the impact of pins, combinational cells, FFs and nets on overall timing. By calculating the cell delay with the NLDM and the wire delay with the Elmore delay model, we derive in detail how much the overall timing changes when the target object causes a change in the wirelength. Moreover, we demonstrate in experiments the superiority of our timing sensitivity over timing-criticality, a traditional and commonly used timing evaluation metric. The timing sensitivity analysis framework is applied to the state-of-the-art timing-driven detailed placer Rsyn [6, 7]. Experimental results show that our timing sensitivity analysis framework brings significant improvements to the timing performance, without compromising the wirelength and routability.

The objective of this work is to establish a comprehensive timing sensitivity analysis framework and to demonstrate its effectiveness through application in timing-driven detailed placement. Note that our current implementation focuses on sensitivity-guided timing optimization, rather than serving as a fully-featured detailed placement engine. From an engineering perspective, simultaneously co-optimizing timing, routability, and power remains an open challenge. Consequently, developing more sophisticated multi-objective optimization strategies to balance these trade-offs in detailed placement, as well as extending the framework to timing-driven global placement, is our direction for future research.

REFERENCES

[1] C. J. Alpert, G.-J. Nam, P. Villarribua, and M. C. Yildiz. Placement stability metrics. In *Proc. ASP-DAC*, pages 1144–1147, New York, NY, USA, 2005. ACM.

[2] J. Bhasker and R. Chadha. *Static Timing Analysis for Nanometer Designs: A Practical Approach*. Springer Science & Business Media, 2009.

[3] A. Bock, S. Held, N. Kámmerling, and U. Schorr. Local search algorithms for timing-driven placement under arbitrary delay models. In *Proc. DAC*, pages 1–6, San Francisco, CA, USA, 2015. IEEE.

[4] A. Chowdhary, K. Rajagopal, S. Venkatesan, T. Cao, V. Tiourin, Y. Parasuram, and B. Halpin. How accurately can we model timing in a placement engine? In *Proc. DAC*, pages 801–806, Anaheim, CA, USA, 2005. IEEE.

[5] W. C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *J. Appl. Phys.*, 19(1):55–63, 1948.

[6] G. Flach, M. Fogaça, J. Monteiro, M. Johann, and R. Reis. Drive strength aware cell movement techniques for timing driven placement. In *Proc. ISPD*, pages 73–80, New York, NY, USA, 2016. ACM.

[7] M. Fogaça, G. Flach, J. Monteiro, M. Johann, and R. Reis. Quadratic timing objectives for incremental timing-driven placement optimization. In *Proc. ICECS*, pages 620–623, Monte Carlo, Monaco, 2016. IEEE.

[8] B. Fu, L. Liu, M. D. Wong, and E. F. Young. Hybrid modeling and weighting for timing-driven placement with efficient calibration. In *Proc. ICCAD*, pages 1–9, New York, NY, USA, 2024. ACM.

[9] Z. Guo and Y. Lin. Differentiable-timing-driven global placement. In *Proc. DAC*, pages 1315–1320, New York, NY, USA, 2022. ACM.

[10] C. Guth, V. Livramento, R. Netto, R. Fonseca, J. Güntzel, and L. Santos. Timing-driven placement based on dynamic net-weighting for efficient slack histogram compression. In *Proc. ISPD*, pages 141–148, New York, NY, USA, 2015. ACM.

[11] C.-C. Huang, Y.-C. Liu, Y.-S. Lu, Y.-C. Kuo, Y.-W. Chang, and S.-Y. Kuo. Timing-driven cell placement optimization for early slack histogram compression. In *Proc. DAC*, pages 1–6, Austin, TX, USA, 2016. IEEE.

[12] J. Jung, G.-J. Nam, L. N. Reddy, I. H.-R. Jiang, and Y. Shin. OWARU: Free space-aware timing-driven incremental placement with critical path smoothing. *IEEE TCAD*, 37(9):1825–1838, 2017.

[13] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In *Proc. ICCAD*, pages 921–926, Austin, TX, USA, 2015. IEEE.

[14] S. Kim, S. Do, and S. Kang. Fast predictive useful skew methodology for timing-driven placement optimization. In *Proc. DAC*, pages 1–6, Austin, TX, USA, 2017. IEEE.

[15] T. Kong. A novel net weighting algorithm for timing-driven placement. In *Proc. ICCAD*, pages 172–176, San Jose, CA, USA, 2002. IEEE.

[16] W. Li, Y. Kukimoto, G. Servel, I. Bustany, and M. E. Dehkordi. Calibration-based differentiable timing optimization in non-linear global placement. In *Proc. ISPD*, page 31–39, New York, NY, USA, 2024. ACM.

[17] P. Liao, D. Guo, Z. Guo, S. Liu, Y. Lin, and B. Yu. DREAMPlace 4.0: Timing-driven placement with momentum-based net weighting and Lagrangian-based refinement. *IEEE TCAD*, 42(10):3374–3387, 2023.

[18] J.-M. Lin, Y.-Y. Chang, and W.-L. Huang. Timing-driven analytical placement according to expected cell distribution range. In *Proc. ISPD*, pages 177–184, New York, NY, USA, 2024. ACM.

[19] V. Livramento, R. Netto, C. Guth, J. L. Güntzel, and L. C. V. D. Santos. Clock-tree-aware incremental timing-driven placement. *ACM TODAES*, 21(3):1–27, 2016.

[20] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. ePlace: Electrostatics-based placement using Fast Fourier Transform and Nesterov's method. *ACM TODAES*, 20(2):1–34, 2015.

[21] T. Luo, D. Newmark, and D. Z. Pan. A new LP based incremental timing driven placement for high performance designs. In *Proc. DAC*, pages 1115–1120, San Francisco, CA, USA, 2006. IEEE.

[22] D. Mangiras, A. Stefanidis, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos. Timing-driven placement optimization facilitated by timing-compatibility flip-flop clustering. *IEEE TCAD*, 39(10):2835–2848, 2020.

[23] J. C. Puget, G. Flach, M. Johann, and R. Reis. Jezz: An effective legalization algorithm for minimum displacement. In *SBCCI*, pages 1–5, New York, NY, USA, 2015. ACM.

[24] H. Ren, D. Pan, and D. Kung. Sensitivity guided net weighting for placement-driven synthesis. *IEEE TCAD*, 24(5):711–721, 2005.

[25] J. A. Roy and I. L. Markov. Eco-system: Embracing the change in placement. *IEEE TCAD*, 26(12):2173–2185, 2007.

[26] S. N. Shahsavani and M. Pedram. TDP-ADMM: a timing driven placement approach for superconductive electronic circuits using alternating direction method of multipliers. In *Proc. DAC*, pages 1–6, San Francisco, CA, USA, 2020. IEEE.

[27] Y. Shi, S. Xu, S. Kai, X. Lin, K. Xue, M. Yuan, and C. Qian. Timing-driven global placement by efficient critical path extraction. In *Proc. DATE*, pages 1–7, Lyon, France, 2025. IEEE.

[28] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Proc. DATE*, pages 1–6, Nice, France, 2007. IEEE.

APPENDIX A: PROOF OF LEMMA 3.3

Proof. (See Fig. 3(a)) The actual arrival time change ΔAAT_u of any sink u caused by the position change of sink j consists of three components [21]. First, due to the wirelength change between source i and sink j, the delay of wire $e_u = (i, u)$ is altered, denote the amount by ΔD_{e_u}. Second, this wirelength change subsequently affects the load capacitance that combinational cell c_i (we use c_i to denote the cell to which pin i belongs) drives, causing a delay change of combinational cell c_i, represented by ΔD_{c_i}. Third, this wirelength change also impacts the input slew of combinational cell c_j, leading to a delay change of combinational cell c_j, denoted as ΔD_{c_j}. Although for any sink $u \in Sink(n)$, ΔD_{c_u} essentially affects the AAT at the output pin of cell c_u, we incorporate ΔD_{c_u} into the calculation of $\Delta \mathrm{AAT}_u$. This approach allows us to account for the influence of ΔD_{c_u} on both the TNS and WNS during the subsequent derivation of sink pin's TNS and WNS sensitivities, and consequently, the derivation is more convenient and intuitive. Therefore, we define the AAT change at any sink u as:

$$\Delta AAT_u = \Delta D_{c_i} + \Delta D_{e_u} + \Delta D_{c_u}. \tag{17}$$

Next, we derive the ΔD_{c_i}, ΔD_{e_u} and ΔD_{c_u} in Eq. (17). We calculate cell delay and slew using NLDM in Eqs. (1) and (2), and calculate wire delay and slew using the Elmore delay model in Eqs. (3) and (4).

From Eqs. (3) and (4), we have

$$\Delta D_{e_u} = \begin{cases} (cR_d + rC_l + rcL_j)\,\Delta L_j + \frac{rc(\Delta L_j)^2}{2}, & \text{if } u = j; \\ c \cdot R_d \cdot \Delta L_j, & \text{otherwise,} \end{cases} \tag{18}$$

$$\Delta S_{e_u} = \begin{cases} K_s\left[(cR_d + rC_l + rcL_j)\Delta L_j + \frac{rc(\Delta L_j)^2}{2}\right], & \text{if } u = j; \\ c \cdot K_s \cdot R_d \cdot \Delta L_j, & \text{otherwise.} \end{cases} \tag{19}$$

In Eqs. (18) and (19), ΔD_{e_u} and ΔS_{e_u} can be obtained directly for $u = j$. If $u \neq j$, the changes of delay and slew are only contributed through the driver and are caused by the term $R_d C_{\text{total}}$, since the wirelength L_u from pin u to source i remains a constant.

Next, in Eq. (17), the delay change ΔD_{c_i} of combinational cell c_i mainly originates from the load capacitance Cap_i it drives, whereas the delay change ΔD_{c_u} of combinational cell c_u is contributed by its input slew Slew_i in Eqs. (1) and (2) of the main text. Thus from Eqs. (1) and (2) of the main text, we get:

$$\Delta D_{c_i} = (b_i + z_i \cdot Slew_{c_i})\Delta \text{Cap}_{c_i} = (b_i + z_i \cdot Slew_{c_i}) \cdot c \cdot \Delta L_j, \tag{20}$$

$$\Delta D_{c_u} = (a_u + z_u \cdot Cap_{c_u}) \cdot \Delta \text{Slew}_{c_u} = (a_u + z_u \cdot Cap_{c_u}) \cdot (\Delta S_{c_i} + \Delta S_{e_u}), \tag{21}$$

$$\Delta S_{c_i} = (v_i + u_i \cdot Slew_{c_i}) \cdot \Delta \text{Cap}_{c_i} = (v_i + u_i \cdot Slew_{c_i}) \cdot c \cdot \Delta L_j. \tag{22}$$

Substituting Eq. (22) and Eq. (19) into Eq. (21), we obtain

$$\Delta D_{c_u} = \begin{cases} (a_u + z_u \cdot Cap_{c_u})(v_i c + u_i \cdot Slew_{c_i} \cdot c + rK_sC_l + cK_sR_d + K_s rcL_j)\Delta L_j + \\ \qquad (a_u + z_u \cdot Cap_{c_u}) \cdot \frac{rcK_s(\Delta L_j)^2}{2}, & \text{if } u = j; \\ (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot \Delta L_j (v_i + u_i \cdot Slew_{c_i} + K_sR_d), & \text{otherwise.} \end{cases} \tag{23}$$

Then, substituting Eqs. (18), (20) and (23) into Eq. (17), we get:

$$\Delta AAT_u =$$

$$\begin{cases} (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j)\Delta L_j + \frac{rc(\Delta L_j)^2}{2} + (a_u + z_u \cdot Cap_{c_u})(v_i c + \\ u_i \cdot Slew_{c_i} \cdot c + rK_sC_l + cK_sR_d + K_s rcL_j)\Delta L_j + (a_u + z_u \cdot Cap_{c_u}) \cdot \frac{rcK_s(\Delta L_j)^2}{2}, & \text{if } u = j; \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d)\Delta L_j + (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot \Delta L_j (v_i + u_i \cdot Slew_{c_i} + K_sR_d), & \text{otherwise.} \end{cases}$$

Hence, we can obtain $S_{L_j}^{AAT_u}$ based on the ΔAAT_u as follows:

$$S_{L_j}^{AAT_u} = \frac{\Delta AAT_u}{\Delta L_j}$$

$$= \begin{cases} (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j) \\ +(a_u + z_u \cdot Cap_{c_u})(v_i c + u_i \cdot Slew_{c_i} \cdot c + rK_sC_l + cK_sR_d + K_s rcL_j), & \text{if } u = j; \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d) + (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot (v_i + u_i \cdot Slew_{c_i} + K_sR_d), & \text{otherwise.} \end{cases}$$

$\square$

APPENDIX B: PROOF OF LEMMA 3.4

Proof. (See Fig. 3(b)) The proof is similar to that of Lemma 3.3. Lemma 3.4 differs from Lemma 3.3 in that, when u serves as the D pin of an FF, the value of ΔD_{c_u} in Eq. (21) will be 0. This is due to the fact that the FF is the starting and ending point of a timing path. Because we use Eq. (1) for cell delay, as mentioned in [4], when the cell is the starting point of the timing path (i.e., FF), the cell delay needs to be set with a boundary condition, and the value of $Slew$ corresponding to the FF will be fixed to zero. In this case, even if the D pin position is changed, it will not affect the delay calculation of FF. When u is a pin of other combinational cell, then its AAT sensitivity can be calculated as Lemma 3.3.

Then, similar to the proof of Lemma 3.3, for the net to which D pin belongs, the AAT change at any sink u is:

$$\Delta AAT_u =$$
$$\begin{cases} (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j)\Delta L_j + \frac{rc(\Delta L_j)^2}{2}, & \text{if } u = \text{D pin;} \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d)\Delta L_j + (a_u + z_u \cdot Cap_{c_u})c(v_i + u_i \cdot Slew_{c_i} + K_s R_d)\Delta L_j, & \text{otherwise.} \end{cases}$$

Hence, we can obtain $S_{L_j}^{AAT_u}$ based on the ΔAAT_u as follows:

$$S_{L_j}^{AAT_u} = \frac{\Delta AAT_u}{\Delta L_j}$$
$$= \begin{cases} b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d + rC_l + rcL_j, & \text{if } u = \text{D pin;} \\ (b_i c + z_i \cdot Slew_{c_i} \cdot c + cR_d) + (a_u + z_u \cdot Cap_{c_u}) \cdot c \cdot (v_i + u_i \cdot Slew_{c_i} + K_s R_d), & \text{otherwise.} \end{cases}$$

□

APPENDIX C: PROOF OF THEOREM 3.5

Proof. (See Fig. 3(a)) When the position of sink j is adjusted, it directly influences the wirelength between source i and sink j, thereby altering the wire delay from i to j. Moreover, this modification in wirelength impacts the load capacitance driven by the driver, leading to a change in the combinational cell delay of the driver. Consequently, this change in the driver's cell delay affects the actual arrival time at all sinks within the net n. Additionally, a single sink influences multiple timing critical endpoints, and its AAT change alters the slacks of these endpoints, and finally leads to a change in TNS. The precise amount of TNS variation ΔTNS induced by the position change of the sink j is:

$$\Delta \text{TNS} = -\sum_{u \in \text{Sink}(n)} F_u \cdot \Delta AAT_u = -\sum_{u \in \text{Sink}(n)} F_u \frac{\Delta AAT_u}{\Delta L_j} \Delta L_j = -\sum_{u \in \text{Sink}(n)} F_u \cdot S_{L_j}^{AAT_u} \Delta L_j,$$

where ΔAAT_u is the variation of AAT at sink u, F_u is the number of timing critical endpoints influenced by the sink u, which is calculated by the algorithm proposed in [24]. Furthermore, L_j is the wirelength between sink j and the source i, ΔL_j is the wirelength variation amount. Thus, we have:

$$S_{\text{TNS}}(j) = \frac{\Delta \text{TNS}}{\Delta L_j} = -\sum_{u \in \text{Sink(n)}} F_u \cdot S_{L_j}^{AAT_u}.$$

□

APPENDIX D: PROOF OF THEOREM 3.6

Proof. According to the proof of Theorem 3.5, a position change of sink j affects the AATs of all sinks of the net n, and each sink affects multiple timing critical endpoints. However, since WNS characterizes the delay of the worst path, we need to determine which sink $u \in Sink(n)$ is on the worst path, due to that the sink u's AAT contributes to the WNS. Hence, the precise amount of WNS variation induced by the position change of the sink j is:

$$\Delta\text{WNS} = \begin{cases} -\Delta AAT_u = -\frac{\Delta AAT_u}{\Delta L_j}\Delta L_j = -S_{L_j}^{AAT_u} \cdot \Delta L_j, \\ \qquad \text{if there exists a sink } u \in Sink(n) \text{ on the worst timing path;} \\ 0, \ \text{otherwise.} \end{cases}$$

Thus, we have

$$S_{\text{WNS}}(j) = \frac{\Delta\text{WNS}}{\Delta L_j} = \begin{cases} -S_{L_j}^{AAT_u}, & \text{if there exists a sink } u \in Sink(n) \text{ on the worst timing path;} \\ 0, & \text{otherwise.} \end{cases}$$

□

APPENDIX E: PROOF OF THEOREM 3.7

Proof. (See Fig. 3(a)) Since we assume that the positions of all other pins are fixed when deriving the timing sensitivity of a pin, moving source i only affects the change in the wirelength of net n, which in turn affects the actual arrival times at all sinks of net n. This behavior is analogous to the case when the sinks' positions change. Therefore, similar to Theorem 3.5, we can easily conclude that the TNS sensitivity of the source i is:

$$S_{\text{TNS}}(i) = \frac{\Delta\text{TNS}}{\Delta L_n} = -\sum_{u' \in \text{Sink(n)}} F_{u'} \cdot S_{L_n}^{AAT_{u'}}, \tag{24}$$

where L_n is the wirelength of net n, ΔTNS and ΔL_n are the variations of TNS and L_n, $F_{u'}$ is the number of timing critical endpoints influenced by the sink u', $S_{L_n}^{AAT_{u'}}$ is the actual arrival time (AAT) sensitivity to wirelength L_n of any sink u' in net n.

Next, we derive the $S_{L_n}^{AAT_{u'}}$ in Eq. (24). As shown in Fig. 3(a), the distances from the source pin to each sink pin are denoted as L_j, L_{u_1} and L_{u_2}. In practice, a change in L_n may cause L_j, L_{u_1} and L_{u_2} to vary in different directions (i.e., they may not increase or decrease simultaneously), which in turn can impact differently the AAT of the sink pins. From Lemma 3.3, we know that for any sink u', the values of $S_{L_j}^{AAT_{u'}}$, $S_{L_{u_1}}^{AAT_{u'}}$ and $S_{L_{u_2}}^{AAT_{u'}}$ must be non-positive. Thus, we can use their sum to express the case that the change in L_n has the maximum effect on AAT at sink u' (i.e., L_j, L_{u_1} and L_{u_2} increase or decrease at the same time). Therefore, we can conclude:

$$S_{L_n}^{AAT_{u'}} = \frac{\Delta AAT_{u'}}{\Delta L_n} = \sum_{u \in \text{Sink(n)}} \frac{\Delta AAT_{u'}}{\Delta L_u} = \sum_{u \in \text{Sink(n)}} S_{L_u}^{AAT_{u'}}. \tag{25}$$

We substitute Eq. (25) into Eq. (24) to obtain:

$$\begin{aligned} S_{\text{TNS}}(i) &= -\sum_{u' \in \text{Sink(n)}} F_{u'} \cdot S_{L_n}^{AAT_{u'}} \\ &= -\sum_{u' \in \text{Sink(n)}} F_{u'} \cdot \sum_{u \in \text{Sink(n)}} S_{L_u}^{AAT_{u'}} \\ &= \sum_{u' \in \text{Sink(n)}} \sum_{u \in \text{Sink(n)}} -F_{u'} \cdot S_{L_u}^{AAT_{u'}} \\ &= \sum_{u \in \text{Sink(n)}} \sum_{u' \in \text{Sink(n)}} -F_{u'} \cdot S_{L_u}^{AAT_{u'}} \end{aligned}$$

$$= \sum_{u \in \text{Sink(n)}} S_{\text{TNS}}(u).$$

□

APPENDIX F: PROOF OF THEOREM 3.8

Proof. (See Fig. 3(a)) Based on the proof of Theorem 3.6, we can analyze by considering whether or not there is a sink $u \in Sink(n)$ on the worst timing path in two cases.

If all sinks of net n are not on the worst timing path, a position change of source i does not affect WNS, and it naturally follows that $S_{\text{WNS}}(i) = 0$. Furthermore, as clearly stated in Theorem 3.6, for any sink u of net n, we have $S_{\text{WNS}}(u) = 0$. At this point, we can conclude:

$$S_{\text{WNS}}(i) = \sum_{u \in \text{Sink(n)}} S_{\text{WNS}}(u) = 0.$$

If there exists a sink $u' \in Sink(n)$ on the worst timing path, then, similar to the analysis in the proof of Theorem 3.6, only the change in the actual arrival time at sink u' affects the WNS. So we have

$$S_{\text{WNS}}(i) = \frac{\Delta \text{WNS}}{\Delta L_n} = -S_{L_n}^{AAT_{u'}},$$

where L_n is the length of the net n, ΔWNS and ΔL_n are the variations of WNS and L_n, $S_{L_n}^{AAT_u}$ is the actual arrival time (AAT) sensitivity to wirelength L_n of sink u in net n.

Based on the same analysis for Eq. (25), we can get

$$S_{\text{WNS}}(i) = -S_{L_n}^{AAT_{u'}} = -\sum_{u \in \text{Sink(n)}} S_{L_u}^{AAT_{u'}} = \sum_{u \in \text{Sink(n)}} -S_{L_u}^{AAT_{u'}} = \sum_{u \in \text{Sink(n)}} S_{\text{WNS}}(u).$$

Hence, Theorem 3.8 holds. □

APPENDIX G: PROOF OF THEOREM 3.9

Proof. (See Fig. 4(a)) A change in the location of a combinational cell (c-cell) is essentially equivalent to the position changes of all the pins located on the c-cell, which includes both critical pins such as pin f and non-critical pins such as pin j. Therefore, a wirelength change caused by moving a c-cell is essentially equivalent to the wirelength changes caused by moving all these pins.

As shown in Fig. 4(a), we use L_j to denote the wirelength from sink j to its source, L_f to denote the wirelength from sink f to its source, L_k to denote the wirelength of the net n_3 in which source k is located. However, a change in the c-cell's position may cause the three wirelengths to change differently (i.e., L_j, L_f and L_k may not increase or decrease simultaneously), which in turn affects the TNS differently. According to Eq. (10) and Eq. (11), the effects of all the three wirelength variations on TNS can be expressed as the TNS sensitivity of the pin. Also, from Theorem 3.5 and Theorem 3.7 we know that the value of the TNS sensitivity of the pin must be non-positive. Thus, we can use their sum to express the case that the change in the c-cell's position has the maximum effect on TNS (i.e., L_j, L_f and L_k increase or decrease simultaneously). So, we can get:

$$S_{\text{TNS}}(\text{c-cell}) = \frac{\Delta \text{TNS}}{\Delta L_j} + \frac{\Delta \text{TNS}}{\Delta L_f} + \frac{\Delta \text{TNS}}{\Delta L_k} = S_{\text{TNS}}(j) + S_{\text{TNS}}(f) + S_{\text{TNS}}(k), \tag{26}$$

where ΔTNS, ΔL_j, ΔL_f and ΔL_k are the variations of TNS, L_j, L_f and L_k.

Thus, it can be seen that in the most sensitive case, the TNS sensitivity value of a combinational cell is equal to the sum of the TNS sensitivity values of all its pins. Hence, Theorem 3.9 holds. □

APPENDIX H: PROOF OF THEOREM 3.10

Proof. (See Fig. 4(a)) Similar to the analysis in Theorem 3.9, we can get:

$$S_{\text{WNS}}(\text{c-cell}) = \frac{\Delta\text{WNS}}{\Delta L_j} + \frac{\Delta\text{WNS}}{\Delta L_f} + \frac{\Delta\text{WNS}}{\Delta L_k} = S_{\text{WNS}}(j) + S_{\text{WNS}}(f) + S_{\text{WNS}}(k),$$

where ΔWNS is the variation of WNS, and the other parameters are the same as those in Theorem 3.9.

Thus, it can be seen that in the most sensitive case, the WNS sensitivity value of a combinational cell is equal to the sum of the WNS sensitivity values of all its pins. Hence, Theorem 3.10 holds. □

APPENDIX I: PROOF OF THEOREM 3.13

Proof. (See Fig. 5) The wirelength change of the net colored in green is caused by moving its associated cells A, B, C, or D. Therefore, the wirelength change of a net is equivalent to the position changes of some cells on the net, which in turn corresponds to the locations changes of all the pins of these cells. All these pins' locations, when changed, lead to wirelength changes of nets in different regions, which in turn may have different effects on TNS. We use the same analysis as in Theorem 3.9 to consider the case that the wirelength change of the net has the maximum impact on TNS, and calculate the TNS sensitivity of the net. Similar to Eq. (26), we can get:

$$\begin{aligned} S_{\text{TNS}}(\text{net}) &= \sum_{u_1 \in \text{Cell A}} S_{\text{TNS}}(u_1) + \sum_{u_2 \in \text{Cell B}} S_{\text{TNS}}(u_2) + \sum_{u_3 \in \text{Cell C}} S_{\text{TNS}}(u_3) + \sum_{u_4 \in \text{Cell D}} S_{\text{TNS}}(u_4) \\ &= S_{\text{TNS}}(A) + S_{\text{TNS}}(B) + S_{\text{TNS}}(C) + S_{\text{TNS}}(D), \end{aligned}$$

where u_1, u_2, u_3 and u_4 denote the pins belonging to cells A, B, C and D, respectively.

Thus, in the most sensitive case, the impact of a net length change on the TNS is equivalent to the sum of the effects on the TNS by moving the driver and all receivers associated to that net. So Theorem 3.13 holds. □

APPENDIX J: PROOF OF THEOREM 3.14

Proof. (See Fig. 5) Similar to the analysis in Theorem 3.13, we can easily have:

$$S_{\text{WNS}}(\text{net}) = S_{\text{WNS}}(A) + S_{\text{WNS}}(B) + S_{\text{WNS}}(C) + S_{\text{WNS}}(D).$$

Thus, it can be seen that in the most sensitive case, the WNS sensitivity value of a net is equal to the sum of the WNS sensitivity values of all its cells. Hence, Theorem 3.14 holds. □