Clock-Wirelength-Driven Detailed Placement

Ziang Ge
ShanghaiTech University
School of Information Science and Technology
Shanghai, China
geza2022@shanghaitech.edu.cn

Yikai Liu
ShanghaiTech University
School of Information Science and Technology
Shanghai, China
liuyk2023@shanghaitech.edu.cn

Jindong Zhou
ShanghaiTech University
School of Information Science and Technology
Shanghai, China
Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences
Shanghai, China
University of Chinese Academy of Sciences
Beijing, China
zhoujd@shanghaitech.edu.cn

Pingqiang Zhou
ShanghaiTech University
School of Information Science and Technology
Shanghai, China
zhoupq@shanghaitech.edu.cn

Abstract

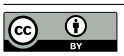
Minimizing clock wirelength via clock sink location optimization during placement has been proven effective in reducing the power consumption of clock trees. Previous works on clock wirelength minimization only focus on global placement stage and ignore significant potential improvement in detailed placement stage. In this paper, we develop a clock-wirelength-driven detailed placement flow which achieves co-optimization of signal-net and clock-net wirelengths. We propose a clock wirelength estimation technique based on K-means clustering, and adapt cell matching to achieve effective and explicit clock wirelength reduction. To reduce signal-net wirelength while preserving sink locations, we further develop customized local reordering and global swap techniques. Experimental results on CLKISPD'05 benchmarks validate the effectiveness of our approach. Compared with NTUplace3, our detailed placer reduces clock wirelength by 19% and 12% on legalized placement solutions that are produced by clock-wirelength-driven global placer and signal-wirelength-driven global placer, respectively.

Keywords

Low power, Clock network synthesis, Placement

ACM Reference Format:
Ziang Ge, Yikai Liu, Jindong Zhou, and Pingqiang Zhou. 2025. Clock-Wirelength-Driven Detailed Placement. In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30–July 02, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3716368.3735165

This work was supported by the Science and Technology Commission of Shanghai Municipality (STCSM) under Grant 24JD1402500.



GLSVLSI '25, New Orleans, LA, USA

ACM ISBN 979-8-4007-1496-2/25/06
https://doi.org/10.1145/3716368.3735165

1 Introduction

Power is a major concern in modern integrated circuit (IC) design. Due to their frequent switching and large load capacitance, clock networks often account for over 30% of total power consumption in a processor chip [12]. Therefore, reducing the power consumption of the clock network is crucial in IC designs. Since longer clock wirelength results in larger clock capacitance and thus implies more power consumption for distribution of clock signals [2], total wirelength is a critical optimization objective in clock tree synthesis (CTS) [2, 13].

In traditional physical design flow, CTS is performed after the placement stage when the locations of the clock sinks (latches and registers) are determined. Considering that the quality of the clock tree generated by CTS is greatly affected by the input sink locations, various clock-wirelength-driven placement algorithms have been proposed to optimize the total wirelength of the clock tree during global placement. Cheon *et al.* [6] proposed to cluster registers based on Manhattan distance and represent these clusters with bounding boxes to model clock-net wirelength with half-perimeter wirelength (HPWL). Net weighting is then applied for clock-net reduction. Wang *et al.* [19] integrated dynamic virtual clock tree construction into quadratic placement to obtain attractive force between registers and sibling internal nodes, which pulls registers together to reduce clock tree wirelength. Lee *et al.* [12] introduced obstacle-aware arboreal attractive force between parent and child nodes in the virtual clock tree and combined it with SimPL [11] to minimize the size of the clock tree. A recent work [9] achieved shorter clock tree wirelength by incorporating clock tree contraction force into the advanced electrostatics-based nonlinear global placement algorithm [5].

Our work is motivated by the observation that *although it has been reported that significant total clock wirelength reduction (45.1% in[9]) can be achieved by clock-wirelength-driven global placement, the optimized sink locations may be seriously destroyed by a subsequent detailed placement stage.* We evaluate the impact of detailed

placement on clock wirelength by running [9] as clock-wirelength-driven global placer on CLKISPD'05 benchmarks [12], and then NTUplace3 [4] as detailed placer. Results show that NTUplace3 brings 13% increase in clock tree wirelength. Therefore, it is important to consider clock wirelength optimization in detailed placement. However, existing works on detailed placement often focus on objectives or constraints such as signal-net wirelength [4, 18], routability [8], GPU acceleration [15] and mixed-cell-height designs [3]. *To the best of our knowledge, there is no clock-wirelength-driven detailed placement work in the literature.*

To achieve clock wirelength minimization in detailed placement stage, there are two major issues that need to be addressed:

- *Evaluation of clock wirelength changes after perturbations*: Detailed placement algorithms typically take in a legalized placement as input and apply perturbations on a set of cells in every iteration, the gain on preset objectives of each perturbation is then evaluated, and the best perturbation which achieves largest gain is applied. Therefore, in order to reduce clock tree wirelength during detailed placement, an efficient method to evaluate the change in clock tree wirelength after perturbations of cells is necessary. Unfortunately, the strategies used in previous works for estimating partial clock tree wirelength by 1) HPWL [6], or 2) iterative virtual clock tree construction [9, 12, 19], in clock-wirelength-driven global placement are not feasible in detailed placement. First, although HPWL estimates signal-net wirelength reasonably well, it does not offer accurate estimates of clock-net wirelength because clock-net routing is very different from signal-net routing [12]. Furthermore, it is unrealistic to perform virtual clock tree construction to evaluate wirelength change after each perturbation of cell locations because there are usually hundreds of thousands of perturbations to be evaluated in one detailed placement iteration [4], and the runtime cost for virtual clock tree construction would be unacceptable.
- *Co-optimization of clock and signal-net wirelengths*: Since clock sinks are connected to both clock-nets and signal-nets, attempts to minimize clock-net wirelength by altering sink locations could bring increase in signal-net wirelength, which is not desired. Previous works on clock-wirelength-driven global placement achieves co-optimization of clock-net and signal-net wirelengths by applying additional clock-force [12] or adding clock tree wirelength into the original nonlinear optimization objective function [9]. These strategies are highly coupled with the characteristics of their respective global placer, and can not be easily transferred to detailed placement stage.

In this paper, we propose ideal approaches to address these two issues and present a clock-wirelength-driven detailed placement flow. We propose a novel method to partition the clock tree into subtrees (partial clock trees) and estimate the change in subtree wirelength after perturbing cell locations. Based on this method, we propose several effective detailed placement techniques for simultaneous optimization of clock tree wirelength as well as signal-net wirelength. Our algorithm can be applied either independently on legalized results by signal-wirelength-driven global placement algorithms, or as a part of a complete clock-wirelength-driven placement flow.

The main contributions of our work are summarized as follows:

- We propose an efficient and accurate method to estimate wirelength of partial clock tree in a legalized placement solution based on K-means clustering.
- We propose a cell matching technique which is capable of simultaneous optimization of both clock-net wirelength and signal-net wirelength based on partial clock tree wirelength estimation.
- We propose a clock-aware local reordering technique and a clock-aware global swap technique to reduce signal-net wirelength without increasing the total wirelength of the final clock tree.
- Experimental results validate the effectiveness of our proposed algorithm with 19% less clock tree wirelength compared with the widely used detailed placer NTUplace3[4] on placement solution produced by a clock-wirelength-driven global placer. Furthermore, our algorithm decreases clock tree wirelength by 12% when applied on placement solution produced by a signal-wirelength-driven global placer.

The rest of this paper is organized as follows. Section II provides the problem formulation. Section III describes the overview of our clock-wirelength-driven detailed placement algorithm and the techniques used in our detailed placement algorithm in details. Experimental results and discussions are presented in Section IV, and Section V concludes this paper.

2 Problem Statement

The input of clock-wirelength-driven detailed placement problem is a legalized row-based standard cell placement of circuit $G(V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of all cells and $E = \{e_1, e_2, ..., e_m\}$ is the set of m singal-nets. Let V_r denote the set of all clock sinks in the circuit, and we have $V_r \in V$. We denote the final clock tree by T, which is a special net consists of all sinks. In this work, we assume there is only one clock source in the circuit. The placeable segments are the rows specified in the placement region. When there are fixed macros, the rows blocked by macros are sliced into new rows to generate placeable segments.

Our primary objective in clock-wirelength-driven detailed placement is to reduce the total wirelength of the final clock tree T by adjusting sink locations while satisfying the legality constraint. Altering sink locations could cause change in signal-net wirelength because sinks are also connected to signal-nets. And our secondary objective is to reduce or preserve the optimized signal-net wirelength in the input placement, which is evaluated with HPWL

$$\text{HPWL} = \sum_{e \in E} \left(\max_{v_i, v_j \in e} \left| x_i - x_j \right| + \max_{v_i, v_j \in e} \left| y_i - y_j \right| \right) \tag{1}$$

where x_i, y_i denote the x and y coordinates of v_i.

The output of clock-wirelength-driven detailed placement is another legalized placement of circuit $G(V, E)$ with optimized sink locations.

3 Clock-Wirelength-Driven Detailed Placment Algorithm

The overall flow of our clock-wirelength-driven detailed placement algorithm is presented in Fig. 1. Given a legalized placement result, we first apply K-means clustering to obtain sink clusters and windows (bounding boxes) of the clusters. We then minimize clock wirelength with clock-wirelength-driven cell matching in each window of the sink clusters in the first loop. In the second loop, clock-aware local reordering and global swap are called iteratively until there is no significant improvement in signal-net wirelength. The proposed

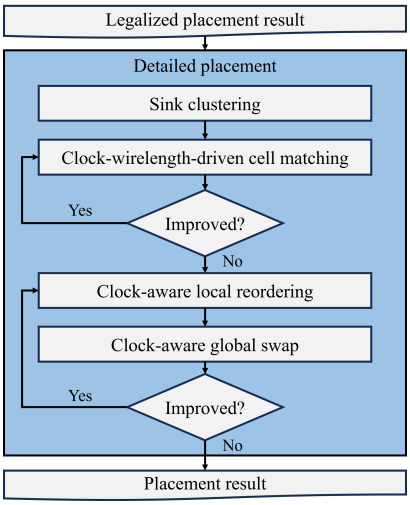


Figure 1: Our clock-wirelength-driven detailed placement flow.

detailed placement algorithm consists of four key techniques: wirelength estimation for partial clock tree, clock-wirelength-driven cell matching, clock-aware local reordering and clock-aware global swap. In wirelength estimation for partial clock tree, we partition all clock sinks into many small subsets, and estimate the wirelength of sub-clock-trees in each subset. Then a clock-wirelength-driven window-based cell matching technique is applied on each subset to optimize the wirelength of the sub-clock-trees in the subset. To ensure that the clock wirelength does not deteriorate when minimizing signal-net wirelength, we propose clock-aware local reordering and clock-aware global swap. Clock-aware local reordering solves for the best order in terms of signal-net wirelength for a group of consecutive standard cells within placeable segments. And to further explore the global solution space, clock-aware global swap technique tries to find a position for sinks and normal standard cells inside their respective optimal regions. In the following subsections, we will present the details of the aforementioned four techniques.

3.1 Wirelength Estimation for Partial Clock Tree

In this section, we present our method of wirelength estimation for partial clock tree that can evaluate the clock wirelength change in an accurate way, without time-consuming CTS after perturbation of sink locations during detailed placement. Thus, it enables the detailed placer to decide whether to accept a perturbation according to its impact on clock wirelength.

There are two unique characteristics of clock-net that make it hard for us to use simple net models (such as the HPWL) for signal-net wirelength estimation to estimate the total wirelength of a complete clock tree. First, the size of the clock-net is significantly larger than most signal-nets. As reported in [6], on average 14.65% of cells in industrial designs are sinks that are connected to the clock-net, while the majority of signal-nets are two-pin nets. Second, clock-net routing considers clock skew as a crucial objective, thus requires longer routes to ensure low skew, which is very different from signal-net routing [12].

Therefore, our idea is to first partition the large clock-net into smaller sub-nets. Next, we approximate the wirelength of the sub-clock-trees with more accurate metrics than HPWL. Fig. 2 provides

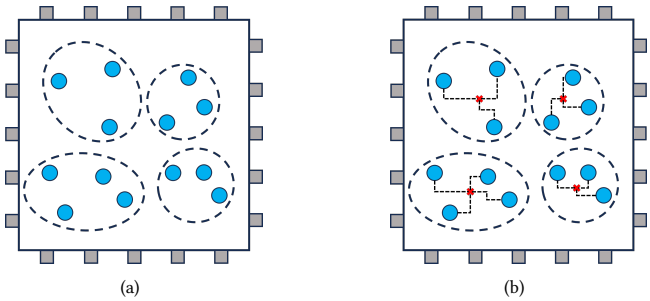


Figure 2: Steps for wirelength estimation of partial clock tree. (a) K-means clustering of sinks. (b) Subtree wirelength estimation with total Manhattan distance to centroid in each subset.

an illustration of our method. Inspired by recent works on CTS [7, 13], we apply K-means clustering to partition V_r into K subsets, with the assumption that sinks in the same subset are in the same subtree (partial clock tree) of the final clock tree. We denote the set of all subsets by $U = \{u_1, u_2, ..., u_K\}$, and the subset that includes sink v_i by $CL(v_i)$. Let $DTC(v_i)$ denote the Manhattan distance from sink v_i to the centroid of $CL(v_i)$, and the x (y) coordinate of the centroid of a subset is the mean of the x (y) coordinates of all the sinks in that subset. The wirelength of the subtree $WL(u_i)$ in each subset u_i is then approximated as follows:

$$WL(u_i) = \sum_{v_j \in u_i} DTC(v_j) \tag{2}$$

Compared with HPWL, our method compensates for the extra wirelength for skew control in clock-net routing and provides more accurate approximation. The accuracy of our method is closely related to the choice of K. Intuitively, increasing K would decrease the average size of the subtrees in each cluster and improve the accuracy of wirelength estimation. However, larger K also leads to significantly longer runtime for running K-means clustering. In this work, we adopt the implementation of K-means clustering in ALGLIB[1], and we set $K = |V_r|/20$ for best balance between estimation accuracy and runtime. Furthermore, it should be pointed out that although we ignore load capacitance of sinks and only

consider Manhattan distance during K-means clustering, it is easy to apply weight-balanced K-means clustering [7] instead of the vanilla K-means clustering, and take load capacitance as weight.

3.2 Clock-Wirelength-Driven Cell Matching

With the subsets obtained through K-means clustering in Section 3.1, we then reduce total wirelength of the final clock tree by minimizing the individual total wirelength of the subtree in each subset. We adapt the window-based cell matching algorithm [4] for this task, which is a widely used and effective detailed placement technique[15]. During window-based cell matching, a set of cells with same width in a given window are extracted in each iteration and removed from the positions they were in to form a corresponding set of empty slots. We then compute the cost of assigning cells in the cell set to each slot in the empty slot set and find the optimal assignment that minimizes the total cost by solving a bipartite matching problem. In the end, the cells are re-inserted into the empty slots following the optimal assignment.

We iteratively apply window-based cell matching on each of the subsets obtained in Section 3.1, and the window of a subset is set to be the bounding box of all sinks in that subset. Fig. 3 gives an

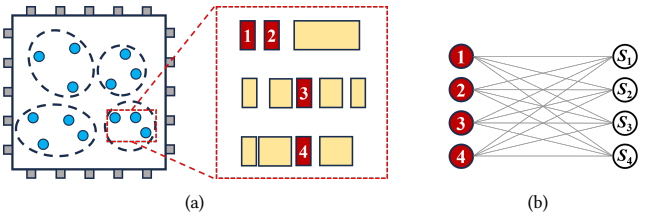


Figure 3: Apply cell matching on bounding boxes of sink clusters. (a) Cells extracted for matching (colored in red) in the window. (b) The bipartite graph constructed to find the best assignment. S_i denotes the empty slot which is originally occupied by cell v_i.

example of the procedure. We treat clock sink cells and normal cells identically when extracting cells from the window, and we switch to the next subset of sinks until all movable cells in one window are extracted. In our experiment, the maximum number of cells extracted each time is set to 90 for best performance. During the construction of the bipartite matching problem, unlike the previous works [4] that take HPWL as cost, we compute the cost c_{ij} of assigning cell v_i to slot s_j as

$$c_{ij} = \begin{cases} \alpha \cdot |E_{v_i}| \cdot DTC(v_i) + (1-\alpha) \cdot HPWL(v_i), v_i \in V_r \\ HPWL(v_i), v_i \notin V_r \end{cases} \quad (3)$$

where E_{v_i} denotes the set of all signel-nets incident to v_i and $HPWL(v_i)$ is the sum of HPWL of all nets in E_{v_i}. We apply $|E_{v_i}|$ as a scaling factor of $DTC(v_i)$ because $DTC(v_i)$ is usually much smaller than $HPWL(v_i)$. α is a tunable factor for adjusting the weight of $DTC(v_i)$ in the cost function. We then solve the bipartite matching problem with the shortest augmenting path algorithm [10] and assign cells to their best slots respectively. By solving for the best cell matching result with the cost function defined in Eq. (3), the detailed placer can reduce HPWL as well as $WL(u_i)$ for each cluster u_i. Furthermore, we can adjust the placer's emphasis on clock tree wirelength reduction by tuning the factor α. When the improvement of one cell matching run in one window becomes marginal in later iterations, we switch to only extracting independent cells [15] in one window for further optimization.

3.3 Clock-Aware Local Reordering

Our clock-wirelength-driven cell matching technique presented in Section 3.2 can explicitly reduce the estimated clock wirelength at the cost of a slight performance loss in signal-net wirelength optimization. To compensate for that, we employ a clock-aware local reordering technique, which reduces signal-net wirelength and does not change sink locations. Therefore, the optimized result obtained in Section 3.2 is preserved.

Local reordering [4] is based on a sliding window, which slides through the cells in each placable segment and finds the best permutation of the k cells included in the window at each step. However, in our implementation, to preserve sink locations during local reordering, we exclude sink cells when the sliding window moves along each placable segment. Fig. 4 shows the idea of our strategy.

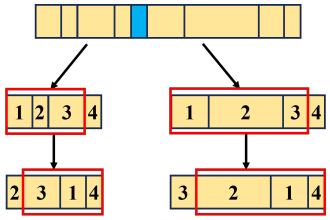


Figure 4: Clock-aware local reordering with sliding window (red rectangle). Sink cell is colored with blue.

A sink cell divides the consecutive cells in the placable segment into two groups, and we apply local reordering sequentially on each group of cells. In our work, the value of k is set to 3 for best efficiency.

3.4 Clock-Aware Global Swap

Although our clock-wirelength-driven cell matching (see Section 3.2) achieves co-optimization of clock-net wirelength and signal-net wirelength, it only explores the solution space in the given window, which has a limited size in trade for the accuracy of wirelength estimation for a partial clock tree. And the clock-aware local reordering presented in Section 3.3 has an even more local view with only exchanges between several consecutive cells. On top of these two techniques, we further propose a clock-aware global swap technique that aims at globally reducing signal-net wirelength by directly moving cells to their optimal regions while keeping the optimized clock tree wirelength intact.

The total HPWL of all the nets incident to a cell v_i is minimum when v_i is placed in its optimal region while all other cells in the circuit are fixed. In our work, the optimal region for a cell $v_i \in V$ is calculated as that in [18]. Then, we search in the optimal region to find a position to insert v_i, which is achieved by swapping v_i with an empty slot or another cell with same width in the optimal region.

However, when v_i is a sink cell, moving v_i into its optimal region may cause v_i to leave the bounding box of $CL(v_i)$ and lead to an potential increase in clock tree wirelength. Let B denote the HPWL change after the swap ($B < 0$ if HPWL decreases after the swap), and v_j denote the cell or slot chosen for swap. Since global swap involves sinks may cause increase in clock tree wirelength, we need to carefully decide whether to accept each swap. Our strategies for choosing cells (slots) for swap are described as follows:

(1) If v_i is a normal cell, and v_j is also a normal cell or an empty slot: the swap is applied for v_i when $B < 0$.
(2) If v_i is a normal cell and v_j is a sink cell: swap between v_i and v_j leads to increase in $DTC(v_j)$ if v_j is moved out of the bounding box of $CL(v_j)$. Therefore, we first try to skip v_j and try the next cell or slot for swap. If there is no other choices, we first check if v_i is inside the bounding box of one of the subsets in U. Because if it is, and we apply the swap, it can be considered that v_j leaves $CL(v_j)$ and joins another cluster (denoted as $CL'(v_j)$). Therefore, if v_i is within the bounding box of a certain cluster in U, we calculate $DTC'(v_j)$ with $CL'(v_j)$ after the swap, and we only apply the swap for v_i when $B < 0$ and $DTC'(v_j) \leq DTC(v_j)$.
(3) If v_i is a sink cell and v_j is a normal cell or empty slot: similarly, if v_j is within the bounding box of a certain cluster in U, we calculate $DTC'(v_i)$ after the swap, and we only apply the swap for v_i when $B < 0$ and $DTC'(v_i) \leq DTC(v_i)$.
(4) If v_i and v_j are both sink cells: it can be considered that v_i and v_j also swap their belonging clusters after exchanging positions, and we have $DTC(v_i) + DTC(v_j) = DTC'(v_i) + DTC'(v_j)$. Thus, this swap would not increase the approximated subtree wirelength of $CL(v_i)$ and $CL(v_j)$, and we can just apply the swap if $B < 0$. Fig. 5 gives an intuitive example to illustrate this situation.

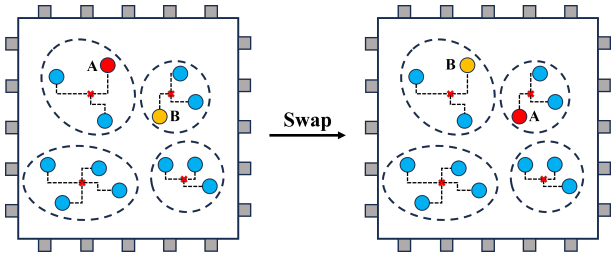


Figure 5: Swap between two clock sinks (highlighted in red and yellow respectively).

Notice that we do not seek for a best cell that gives us maximum improvement during cell swap. Instead, to save runtime, we apply the swap for v_i as soon as we find a cell v_j that is suitable according to our strategies.

4 Results and Discussions

We implement our detailed placement algorithm in C++, and the experiments are conducted on a Linux machine with Intel i7 12700 2.1GHz CPU and 16GB memory. We test the performance of our algorithm on the CLKISPD'05 benchmarks [12], which is used in recent work on clock-wirelength-driven global placement[9] as well.

Table 1: Statistics of CLKISPD'05 benchmarks.

Name	#Cells	#Registers	#Nets	#Macros
clkad1	210K	32K	221K	56
clkad2	255K	38K	266K	177
clkad3	451K	68K	466K	721
clkad4	494K	74K	516K	1329
clkbb1	278K	42K	284K	30
clkbb2	535K	84K	577K	923
clkbb3	1095K	165K	1123K	666
clkbb4	2169K	327K	2230K	639

Table 1 shows the details of the CLKISPD'05 benchmarks, in which 15% of standard cells are selected to be registers. We implement the ZST-DME algorithm[2] to generate the final zero-skew clock tree in all our experiments. The wirelength of signal-nets is estimated with HPWL, and wirelength of the final clock tree is calculated by summing the Manhattan length of all clock edges. Furthermore, we set $\alpha = 0.6$ in the cost function defined in Eq. (3) throughout our experiments.

As stated in Section 1, our clock-wirelength-driven detailed placement algorithm can be applied either as a part of a complete clock-wirelength-driven placement flow, or independently on legalized results by signal-wirelength-driven global placement algorithms. To evaluate the performance of our detailed placement algorithm in both scenarios, we re-implement the state-of-the-art clock-wirelength-driven global placement algorihtm [9] and apply NTUplace3 [4] as legalizer to produce two sets of legalized placement solutions of CLKISPD'05 benchmarks as input of our algorithm:

- *CWL+WL-OPT*: Placement solutions produced by 1) the original global placer [9], and 2) NTUplace3 leaglizer;
- *WL-OPT*: Placement solutions produced by 1) modified global placer [9] that turns off clock wirelength optimization, and 2) NTUplace3 leaglizer.

We first evaluate the performance of our detailed placement algorithm when it takes CWL+WL-OPT as input. Since there is no existing work on clock-wirelength-driven detailed placement, we compare our algorithm with the detailed placement module of NTUplace3, which is widely used in placement research [5, 14, 16], and is still competitive in terms of signal-net wirelength optimization on the ISPD'05 benchmark[17] compared with a recent work on detailed placement [15]. The final clock tree wirelength (ClkWL), HPWL and runtime results are presented in Table 2, and the normalized ratios based on NTUplace3 is shown in the last row. Although NTUplace3 is able to reduce HPWL by 4%, it damages the optimized register locations obtained by clock-wirelength driven global placement and causes 13% increase in clock wirelength. In contrast, our algorithm is able to further improve clock wirelength by 6% over clock-wirelength-driven global placement. Additionally, our placer reduces signal-net wirelength by 2% when compared to CWL+WL-OPT. Compared with NTUplace3, our placer obtains 19% shorter final clock tree wirelength.

Table 2: Comparison with NTUplace3 on clock-wirelength-driven global placement on CLKISPD'05 benchmarks.

Benchmarks	CWL+WL-OPT		CWL+WL-OPT + detailed placement by NTUplace3[4]			CWL+WL-OPT + our detailed placement flow		
	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	Runtime (s)	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	Runtime (s)
clkad1	1.37	77.47	1.54	75.61	12	1.29	77.09	42.74
clkad2	1.53	85.79	1.67	84.46	14	1.43	85.44	56.95
clkad3	3.08	209.29	3.46	202.83	30	2.87	206.32	156.66
clkad4	3.17	193.66	3.69	187.63	31	3.00	191.78	168.47
clkbb1	1.65	94.05	1.85	92.19	17	1.54	93.93	70.33
clkbb2	3.40	158.02	4.10	148.76	54	3.21	155.62	281.51
clkbb3	5.47	393.53	6.57	367.48	89	5.20	385.86	878.23
clkbb4	12.04	879.42	13.75	846.30	176	11.15	865.43	3134.47
Avg	0.87×	1.04×	1.00×	1.00×	1.00×	0.81×	1.02×	6.91×

Table 3: Comparison with NTUplace3 on signal-wirelength-driven global placement on CLKISPD'05 benchmarks.

Benchmarks	WL-OPT		WL-OPT + detailed placement by NTUplace3[4]			WL-OPT + our detailed placement flow		
	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	Runtime (s)	ClkWL ($\times 10^6$)	HPWL ($\times 10^6$)	Runtime (s)
clkad1	1.66	74.68	1.67	73.89	11	1.47	74.97	43.05
clkad2	1.80	84.39	1.82	83.55	14	1.62	84.58	60.46
clkad3	3.72	198.47	3.74	196.02	25	3.31	197.89	143.44
clkad4	4.06	182.69	4.05	180.34	28	3.62	182.54	163.64
clkbb1	1.98	90.33	1.98	89.73	14	1.75	91.04	67.46
clkbb2	4.44	141.82	4.45	139.32	41	3.89	142.07	283.35
clkbb3	7.30	318.05	7.35	310.92	64	6.40	314.56	696.21
clkbb4	15.25	767.87	15.29	755.29	183	13.11	764.34	3361.9
Avg	0.99×	1.01×	1.00×	1.00×	1.00×	0.88×	1.01×	7.60×

To validate the effectiveness of our algorithm without clock-wirelength-driven global placement, we further test the performance of our detailed placer as an independent tool for post-legalization clock tree wirelength optimization by taking WL-OPT as input. NTUplace3 is again chosen for comparison. Table 3 shows the corresponding ClkWL, HPWL and runtime results. As an independent tool, our algorithm can reduce clock wirelength by 12% with no increase in HPWL compared with the original placement solution. Fig. 6 shows the comparison of register placement of *clkad1* in WL-OPT before and after running our detailed placement flow.

Runtime overhead is a common problem in previous works on clock-wirelength-driven global placement[9, 12]. And it is noticeable that the runtime of our algorithm reported in Table 2 and Table 3 is 6.91× and 7.6× longer than NTUplace3 respectively.

Fig. 7 shows the runtime breakdown of our algorithm on CLKISPD'05 benchmarks. It can be seen that K-means clustering (see Section 3.1) consumes over half of the runtime. In our present implementation, we directly adopt the K-means clustering in ALGLIB, and we believe that a more efficient implementation of K-means clustering is available, but this would be our future work. In addition, our algorithm can achieve further speedup by GPU acceleration since the recent work [15] has shown how to use GPU to accelerate key steps used in our work such as cell matching, local reordering and global swap.

5 Conclusion

In this paper, we present a clock-wirelength-driven detailed placement algorithm. It is able to effectively reduce total wirelength of the final clock tree by shrinking the subtrees with a cell matching technique supported by subtree wirelength estimation. Furthermore, our placer includes clock-aware global swap and clock-aware local reordering technique to reduce signal-net wirelength while preserving the optimized sink locations. We propose a flow to combine these techniques to realize co-opimization of signal-net and clock-net wirelength. Experimental results on CLKISPD'05 benchmarks show that our algorithm outperforms NTUplace3 in terms of clock-net wirelength minimization when adopted both independently and jointly with a clock-wirelength-driven global placer. Our future work targets acceleration of our algorithm with GPUs and extension towards other design objectives in CTS like buffering, skew, and etc..

References

[1] ALGLIB. [n. d.]. https://www.alglib.net/
[2] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and A.B. Kahng. 1992. Zero skew clock routing with minimum wirelength. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 39, 11 (1992), 799–814.
[3] Guohao Chen, Zheng Zeng, Benchao Zhu, Jiawei Li, Kun Wang, Jun Yu, and Jianli Chen. 2023. Mixed-cell-height Placement with Minimum-Implant-Area and Drain-to-Drain Abutment Constraints. In *ACM/IEEE Design Automation Conference*. 1–6.
[4] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. 2008. NTUplace3: An Analytical Placer for Large-Scale Mixed-Size

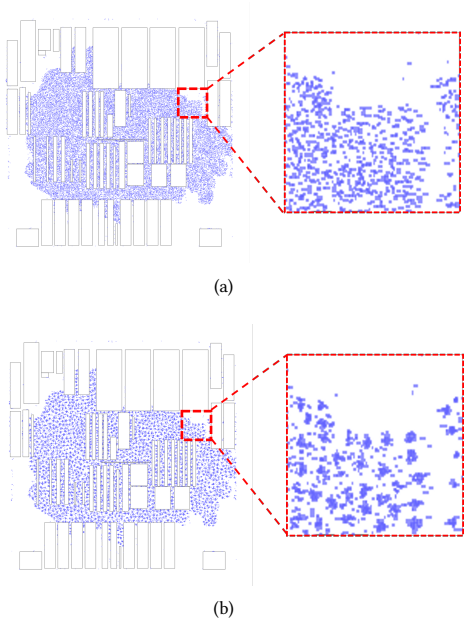


Figure 6: Register placement of *clkad1* before and after running our clock-wirelength-driven placement flow. (a) Register placement in WL-OPT. (b) Register placement after clock-wirelength-driven detailed placement.

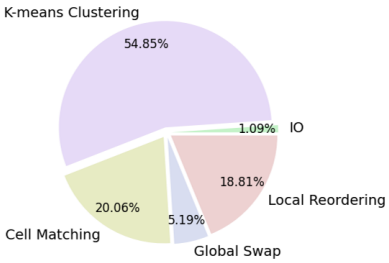


Figure 7: Runtime breakdown of our algorithm on CLK-ISPD'05 benchmarks.

Designs With Preplaced Blocks and Density Constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 7 (2008), 1228–1240.

[5] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. 2019. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2019), 1717–1730.

[6] Yongseok Cheon, Pei-Hsin Ho, Andrew B. Kahng, Sherief Reda, and Qinke Wang. 2005. Power-aware placement. In *ACM/IEEE Design Automation Conference*. 795–800.

[7] Byungho Choi, Yonghwi Kwon, Umar Afzaal, and Youngsoo Shin. 2023. Multi-source Clock Tree Synthesis Through Sink Clustering and Fast Clock Latency Prediction. In *IEEE International Symposium on Circuits and Systems*. 1–4.

[8] Wing-Kai Chow, Jian Kuang, Xu He, Wenzan Cai, and Evangeline F.Y. Young. 2014. Cell density-driven detailed placement with displacement constraint. In *International Symposium on Physical Design*. 3–10.

[9] Jinghao Ding, Linhao Lu, Zhaoqi Fu, Jie Ma, Mengshi Gong, Yuanrui Qi, and Wenxin Yu. 2023. Clock Aware Low Power Placement. In *IEEE/ACM International Conference on Computer-Aided Design*. 01–08.

[10] Roy Jonker and Ton Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 4 (1987), 325–340.

[11] Myung-Chul Kim, Dong-Jin Lee, and Igor L. Markov. 2012. SimPL: An Effective Placement Algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 1 (2012), 50–60.

[12] Dong-Jin Lee and Igor L. Markov. 2012. Obstacle-Aware Clock-Tree Shaping During Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 2 (2012), 205–216.

[13] Weiguo Li, Zhipeng Huang, Bei Yu, Wenxing Zhu, and Xingquan Li. 2024. Toward Controllable Hierarchical Clock Tree Synthesis with Skew-Latency-Load Tree. In *ACM/IEEE Design Automation Conference*.

[14] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2021. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 4 (2021), 748–761.

[15] Yibo Lin, Wuxi Li, Jiaqi Gu, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2020. ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 5083–5096.

[16] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method. *ACM Transactions on Design Automation of Electronic Systems* 20, 2 (2015), 1–34.

[17] Gi-Joon Nam, Charles J. Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. 2005. The ISPD2005 placement contest and benchmark suite. In *International Symposium on Physical Design*. 216–220.

[18] Min Pan, N. Viswanathan, and C. Chu. 2005. An efficient and effective detailed placement algorithm. In *IEEE/ACM International Conference on Computer-Aided Design*. 48–55.

[19] Yanfeng Wang, Qiang Zhou, Xianlong Hong, and Yici Cai. 2007. Clock-Tree Aware Placement Based on Dynamic Clock-Tree Building. In *IEEE International Symposium on Circuits and Systems*. 2040–2043.