An Improved Clock-Aware Global Placement Algorithm

Ziang Ge and Pingqiang Zhou
School of Information Science and Technology
Shanghaitech University, Shanghai, P. R. China
{geza2022, zhoupq}@shanghaitech.edu.cn

2024 IEEE 17th International Conference on Solid-State & Integrated Circuit Technology (ICSICT) | 979-8-3503-6183-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSICT62049.2024.10831277

***Abstract*—Traditional very-large-scale integrated (VLSI) circuits physical design flows that optimize clock networks after placement are limited by the quality of register placement. Prior research on clock-aware placement shows effectiveness in minimizing clock-net wirelength during placement but suffers from large runtime overhead due to iterative clock-tree construction process. In this paper, we propose to accelerate clock-aware placement with a fast clock-tree synthesis method and a preconditioner for clock wirelength gradient. Experimental results show that the proposed method can save 30% runtime over prior work with only 1% placement quality degradation.**

***Index Terms*—Placement, clock-network synthesis, nonlinear optimization, low power.**

I. Introduction

Due to their frequent switching and large capacitance, clock networks often account for over 30% of total power consumption in synchronous sequential designs [1]. Total wirelength is a critical optimization objective in clock network synthesis, since larger clock wirelength results in greater clock capacitance and thus require more power for distribution of clock signals [2].

Fig. 1 illustrates the steps of VLSI physical design. Clock network synthesis is usually performed after placement, which determines the locations of the sequential logics such as registers. Considering that the quality of synthesized clock network is greatly affected by register locations, such design flow limits the design space for clock network synthesis. To overcome such limitation, previous works propose to optimize the total switching power of clock networks by minimizing clock wirelength during placement stage [1], [4], [5]. This is achieved by adding additional force on registers during global placement. The direction of the additional force is estimated by constructing a virtual clock-tree at each global placement iteration. However, the runtime overhead for iterative virtual clock-tree construction become non-negligible over the many iterations of global placement. [5] proposes a grid-based algorithm to accelerate DME-based virtual clock-tree construction, at the cost of a 20% runtime overhead.

To further accelerate clock-aware global placement, we apply a much faster clock-tree synthesis method for virtual clock-tree construction. The key insight is that the purpose of virtual clock-tree construction is to estimate the direction of force to pull registers together, rather than obtaining a high-quality clock-tree. Thus, we can sacrifice virtual clock-tree quality for lower runtime. Furthermore, for the first time, we propose a preconditioner for clock wirelength gradient to balance the optimization of clock-net wirelength and signal wirelength. The proposed technique is integrated into ePlace [6]. Experimental results on eight CLKISPD'05 benchmarks [1] show that, compared to the state-of-the-art clock-aware placement algorithm [5], our methods can reduce the runtime of clock-aware global placement by 30% on average.

This work was supported in part by the National Natural Science Foundation of China under the Grant No. 62074100.

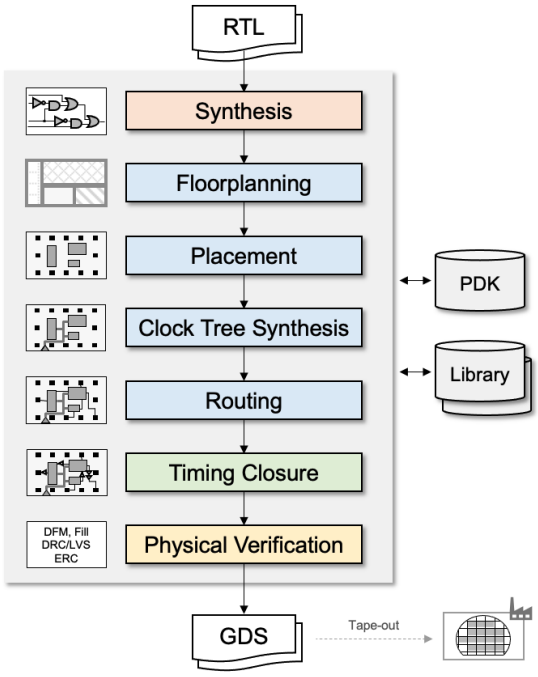

Fig. 1. VLSI physical design flow [3].

II. Related Works

In this section, we present an overview of the state-of-the-art clock-aware global placement algorithm [5]. RePlAce [7] is chosen as the placement backbone, in which the global placement problem is formulated as an unconstrained optimization problem

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda N(\mathbf{v}) \tag{1}$$

where $\mathbf{v} = (\mathbf{x}, \mathbf{y})^T = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$ denotes a solution to accommodate all n objects and $W(\mathbf{v})$ is the wirelength function. $N(\mathbf{v})$ is the density function, which is modeled by eDensity [6], and λ is the penalty factor for adjusting the ratio between wirelength and density.

[5] extends the original optimization problem to consider clock-tree wirelength and the new objective function is defined as

$$f(\mathbf{v}) = W_u(\mathbf{v}) + \eta W_c(\mathbf{v}) + \lambda N(\mathbf{v}) \tag{2}$$

where $W_u(\mathbf{v})$ denotes signal-net wirelength and $W_c(\mathbf{v})$ is the clock-net wirelength. η is the weight used to scale the clock-net wirelength. The nonlinear optimization problem in Eq. (2) is then solved using Nesterov's method [6]. Solving such problem using Nesterov's method involves efficiently computing the gradient function. The gradient function $f(v)$ over one variable x_i can be derived as

$$\frac{\partial f(\mathbf{v})}{\partial x_i} = \frac{\partial W_u(\mathbf{v})}{\partial x_i} + \eta \frac{\partial W_c(\mathbf{v})}{\partial x_i} + \lambda \frac{\partial N(\mathbf{v})}{\partial x_i} \tag{3}$$

where the density gradient is computed using fast Fourier transform (FFT) [6]. The gradient of signal-net wirelength is obtained by differentiating the WA wirelength model [8]. The weight η is used to explore the optimization tradeoff between signal-net wirelength and clock-net wirelength. It should be noted that in the practice of nonlinear optimization, the gradient vector ∇f is often multiplied by a preconditioner [6] to accelerate its convergence process.

Since the existing wirelength models for signal-nets do not offer accurate estimation of clock-net wirelength [1], to obtain the gradient of clock-net wirelength, a virtual clock-tree is constructed at each global placement iteration. Then the gradient of clock-net wirelength for each register is estimated by accumulating gradients along the clock-tree. Specifically, the constructed virtual clock-tree is decomposed into a set of two-pin pseudo-nets and the clock-net gradient on leaf nodes (registers) is estimated by accumulating the wirelength gradients of the two-pin nets along each root-to-leaf path. Thus, the clock-net gradient function is defined as

$$\frac{\partial W_c(\mathbf{v})}{\partial x_i} = \sum_{t_j \in \mathcal{T}} w_{l_j} \frac{\partial W_{t_j}(\mathbf{v})}{\partial x_i} \tag{4}$$

where $\mathcal{T}$ is the set of all two-pin pseudo-nets from a register r_i to the root, and w_{l_j} denotes the weight of gradient of level l_j at which the two-pin pseudo-net t_j is located in the clock-tree. Fig. 2 gives an example of arboreal clock-net gradient.

The runtime overhead of constructing the virtual clock-tree using nearest-neighbor graph (NNG) and deferred-merge embedding (DME) at every global placement iteration can be significant. For example, the complexity of naive NNG construction is $O\left(n^3\right)$ for n registers. Therefore, a grid-based algorithm for NNG construction is proposed in [5] to accelerate virtual clock-tree construction. The NNG has a time complexity of $O\left(n^2 \log n + k^2 n\right)$, where k is the average number of sinks in a grid.

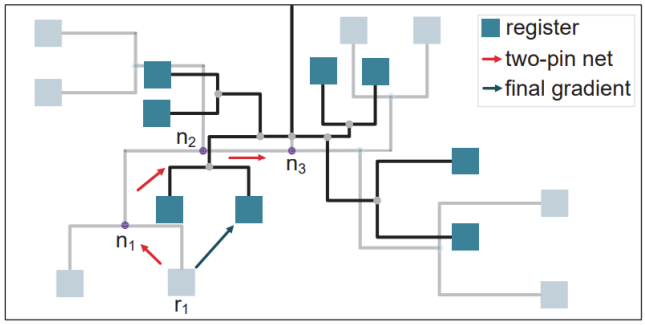

Fig. 2. Arboreal gradient accumulation in [5].

III. Methodology

Although the grid-based algorithm can accelerate virtual clock-tree construction, its runtime overhead is still non-negligible (20% as reported in [5]). More importantly, the effort to build a well-optimized virtual clock-tree with NNG and DME algorithm can be wasteful, because

- DME-based algorithm tries to minimize clock skew while our work targets the minimization of clock-tree wirelength.
- In each iteration of global placement, previous virtual clock-tree is discarded and a new one is built from scratch with updated register locations. Therefore, over-optimizing a virtual clock-tree with current register locations can be meaningless especially during the early stage of global placement, when the registers are far from their final positions.

For efficient and effective virtual clock-tree construction, we propose to replace NNG and DME with the method of means and medians (MMM) [9], which is much faster and adequately effective to guide the movement of registers.

A. Virtual Clock-Tree Construction with MMM

We adopt ePlace [6] as our placement backbone and use the objective function defined in Eq. (2). We apply the MMM algorithm to construct a virtual clock-tree with current register locations in one global placement iteration and follow the arboreal clock-net gradient accumulation to obtain clock-net wirelength gradient. The overall computation flow is shown in Fig. 3. MMM is a classic algorithm for clock-tree synthesis. It recursively partitions the set of sinks (registers) into two subsets of equal cardinality (median). Then, the center of mass of the set is connected to the centers of mass of the two subsets (mean). The MMM algorithm has a time complexity of only $O\left(n \log n\right)$. Notice that the effectiveness of MMM depends heavily on the choice of partition directions for median computation, in this work, we alternate the cut direction during a single MMM run because all benchmark circuits we use in this work have an approximate square shape. The main drawback of MMM is that it only minimizes clock skew heuristically. But this is acceptable because we only need to optimize clock wirelength during global placement.

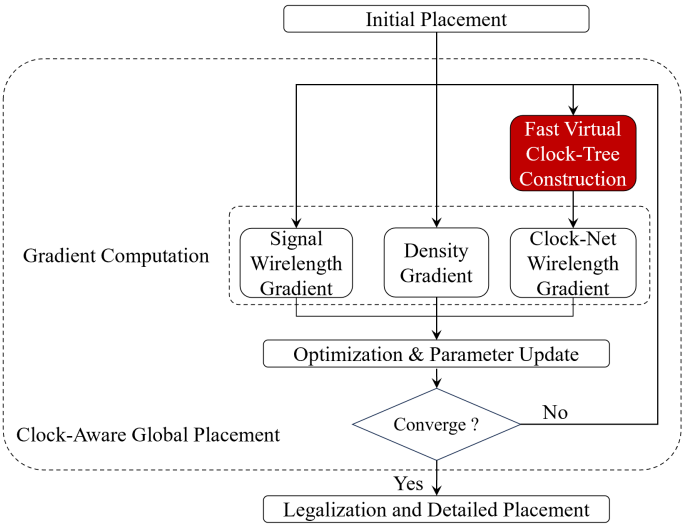

Fig. 3. Overall flow of our algorithm.

The strategy for adjusting η in [5] is adopted and η is defined as

$$\eta = k_1 e^{k_2(k_3-\tau)} \tag{5}$$

where k_1, k_2, k_3 are chosen to be $6, 10, 0.1$ respectively in our experiments, and τ is the global density overflow, which is defined as the total overflow area in all placement bins over the area of total movable blocks [10] and it usually starts from around 1.0 and ends with 0.1. We observe that η is very small during early stage of global placement (when the global density overflow is fairly high) and $\eta\frac{\partial W_c(\mathbf{v})}{\partial x_i}$ is thus close to 0. Therefore, we only conduct virtual clock-tree construction when $\tau < 0.6$ and clock-net gradient is set to 0 when $\tau \geq 0.6$.

The weight of gradient at level i is calculated as

$$w_{l_i} = \frac{1}{\sqrt{2\pi}\sigma} e^{\left(-\frac{i_i^2}{2\sigma^2}\right)} \tag{6}$$

where l_i is defined as

$$l_i = 3\left(1 - \frac{i}{L_t}\right) \tag{7}$$

Here L_t is the total level count of the clock-tree. We follow [5] for the definition of σ in Eq. (6)

$$\sigma = p - \frac{1}{e^{\tau+q}} \tag{8}$$

where p and q are set to 10.35 and -1.68 respectively.

B. Clock-Net Wirelength Preconditioner

Preconditioning helps reduce the condition number of a problem and has been verified to be effective for accelerating the nonlinear global placement [6]. Signal wirelength preconditioner and density preconditioner have been used in [6] for signal wirelength optimization.

In this work, we propose a clock-net wirelength preconditioner to 1) speed up the convergence of clock-net wirelength optimization, and 2) relieve the imbalance between object gradients in Eq. (3). We approximate $\mathbf{H_f}$ of Eq. (2) to be a positive definite diagonal matrix $\widetilde{\mathbf{H}}_\mathbf{f}$

$$\mathbf{H_f} \approx \widetilde{\mathbf{H}}_\mathbf{f} = \begin{pmatrix} \widetilde{\mathbf{H}}_{\mathbf{f_{x,x}}} & 0 \\ 0 & \widetilde{\mathbf{H}}_{\mathbf{f_{y,y}}} \end{pmatrix} \tag{9}$$

where

$$\widetilde{\mathbf{H}}_{\mathbf{f_{x,x}}} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{10}$$

is the x part of the precondioner, and the y part $\widetilde{\mathbf{H}}_{\mathbf{f_{y,y}}}$ can be expressed in a similar way. Then we use $\widetilde{\mathbf{H}}_\mathbf{f}$ as the preconditioner and calculate $\nabla f_{\text{pre}} = \widetilde{\mathbf{H}}_\mathbf{f}^{-1}\nabla f$ in nonlinear optimization.

To obtain $\widetilde{\mathbf{H}}_{\mathbf{f_{x,x}}}$, according to Eq. (2), we have

$$\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} = \frac{\partial^2 W_u(\mathbf{v})}{\partial x_i^2} + \eta\frac{\partial^2 W_c(\mathbf{v})}{\partial x_i^2} + \lambda\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} \tag{11}$$

We then need to compute or estimate $\partial^2 W_u(\mathbf{v})/\partial x_i^2$, $\partial^2 W_c(\mathbf{v})/\partial x_i^2$ and $\partial^2 N(\mathbf{v})/\partial x_i^2$ separately. The readers are referred to [6] for the estimations of the terms $\partial^2 W_u(\mathbf{v})/\partial x_i^2$ and $\partial^2 N(\mathbf{v})/\partial x_i^2$. For the second term $\partial^2 W_c(\mathbf{v})/\partial x_i^2$, after differentiating Eq. (4) with respect to x_i, we can get the second-order gradient of the clock-net wirelength function as

$$\frac{\partial^2 W_c(\mathbf{v})}{\partial x_i^2} = \sum_{t_j \in \mathcal{T}} w_{l_j} \frac{\partial^2 W_{t_j}(\mathbf{v})}{\partial x_i^2} \tag{12}$$

Here we use the degree of two-pin pseudo-nets, 2, as the estimation of $\partial^2 W_{t_j}(\mathbf{v})/\partial x_i^2$, so we get

$$\frac{\partial^2 W_c(\mathbf{v})}{\partial x_i^2} = 2\sum_{t_j \in \mathcal{T}} w_{l_j} \tag{13}$$

IV. Experimental Results

Our algorithm is implemented in C++ and ran on a Linux machine with Intel i7 12700 2.1GHz CPU and 16GB memory. We conduct experiments on the CLKISPD'05 benchmarks [1], which include register lists generated based on the ISPD2005 benchmarks [11]. 15% of standard cells are selected to be registers in each benchmark. Table II shows the details of the benchmarks.

For fair comparison, we implement the algorithm [5] in C++ and run on the same machine. Notice that although RePlAce is chosen as the placement backbone in [5], it shares the core algorithm with ePlace. Therefore we choose ePlace as the placement backbone of our algorithm. We propose two techniques, MMM and preconditioner for clock wirelength, to speed up the clock-aware global placement algorithm. To show their effectiveness, we design two algorithms for comparison. One is *Our-MMM*, which only includes MMM for virtual clock-tree construction, and the other is *Ours-MMM-Pre* which includes both techniques.

TABLE I
RESULTS ON THE CLKISPD'05 BENCHMARK SUITE

Bench	[5]			Ours-MMM			Ours-MMM-Pre		
	ClkWL	HPWL	Time (s)	ClkWL	HPWL	Time (s)	ClkWL	HPWL	Time (s)
clkad1	1.37	77.47	97.37	1.34	77.92	78.1	1.37	77.89	82.64
clkad2	1.53	85.79	179.53	1.48	89.81	150.79	1.50	88.31	137.68
clkad3	3.08	209.29	349.62	3.13	207.93	253.22	3.19	206.90	245.31
clkad4	3.17	193.66	377.9	2.91	204.24	303.16	3.29	199.29	268.51
clkbb1	1.65	94.05	202.55	1.58	95.11	151.58	1.60	94.73	154.88
clkbb2	3.40	158.02	388.5	3.69	155.52	230.42	3.64	157.56	238.02
clkbb3	5.47	393.53	1314.02	5.12	470.51	1132.72	5.99	379.96	641.22
clkbb4	12.04	879.42	4113.9	9.93	951.90	3293.8	11.51	909.78	3011.31
Avg	1.00×	1.00×	1.00×	0.96×	1.05×	0.77×	1.01×	1.02×	0.70×

TABLE II
STATISTICS OF CLKISPD'05 BENCHMARKS.

Name	Cells	Regs	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

We use NTUplace3 [10] as our legalizer and skip detailed placement because it optimizes signal wirelength only and may harm the result of clock-net wirelength. After legalization, we build the final zero-skew clock-tree for evaluation using DME-based method [1], [5]. The wirelength of signal nets is estimated with HPWL, while the wirelength of the clock net is obtained by summing the Manhattan length of all edges in the final clock-tree. Since global placement might easily diverge, we record the result of best overflow for each run.

Experimental results are shown in Table I with wirelength in $\times 10^6$ and CPU time in seconds. Both Ours-MMM and Ours-MMM-Pre are faster than [5] with 23% and 30% lower runtime respectively. Compared to [5], Ours-MMM achieves 4% lower clock-net wirelength and 5% higher signal wirelength while Ours-MMM-Pre achieves a more balanced result with only 1% and 2% degradation on clock-net wirelength and signal wirelength respectively. Since we use the same placement backbone here, experimental results show that our method can effectively alleviate the overhead brought by iterative virtual clock-tree construction while maintaining high quality of placement solution. Furthermore, after applying the preconditioner for clock-net wirelength, our method can achieve faster convergence speed and better balance between the optimizations of signal wirelength and clock-net wirelength.

V. CONCLUSION

In this work, we propose an improved clock-aware placement methodology for electrostatics-based nonlinear placer to minimize runtime overhead. Experiments show that our method can achieve 30% lower runtime with negligible quality degradation compared with previous methods on the CLK-ISPD'05 benchmarks.

REFERENCES

[1] D.-J. Lee and I. L. Markov, "Obstacle-aware clock-tree shaping during placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 2, pp. 205–216, 2012.

[2] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, and A. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 11, pp. 799–814, 1992.

[3] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[4] Y. Wang, Q. Zhou, X. Hong, and Y. Cai, "Clock-tree aware placement based on dynamic clock-tree building," in *IEEE International Symposium on Circuits and Systems*, 2007, pp. 2040–2043.

[5] J. Ding, L. Lu, Z. Fu, J. Ma, M. Gong, Y. Qi, and W. Yu, "Clock aware low power placement," in *IEEE/ACM International Conference on Computer Aided Design*, 2023, pp. 01–08.

[6] 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 Transactions on Design Automation of Electronic Systems*, vol. 20, no. 2, pp. 1–34, 2015.

[7] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1717–1730, 2019.

[8] M.-K. Hsu, V. Balabanov, and Y.-W. Chang, "TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 4, pp. 497–509, 2013.

[9] M. Jackson, A. Srinivasan, and E. Kuh, "Clock routing for high-performance ICs," in *ACM/IEEE Design Automation Conference*, 1990, pp. 573–579.

[10] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, "NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1228–1240, 2008.

[11] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ISPD2005 placement contest and benchmark suite," in *International Symposium on Physical Design*, 2005, pp. 216–220.