DESIGN, AUTOMATION & TEST IN EUROPE
14 – 15 March 2022 · on-site event
16 – 23 March 2022 · online event
The European Event for Electronic System Design & Test

DREAMPlace 4.0: Timing-driven Global Placement with Momentum-based Net Weighting

Peiyu Liao[1,2], Siting Liu[1,2], Zhitang Chen[3], Wenlong Lv[3], Yibo Lin[1], Bei Yu[2]

[1]Peking University [2]The Chinese University of Hong Kong
[3]Huawei Noah's Ark Lab

Outline

Outline

VLSI Placement and Challenges

Modern VLSI size and design complexity are growing rapidly.

- Billions of cells in a single design.
- Performance requirements.
- Different design rules and constraints.

VLSI Placement and Challenges

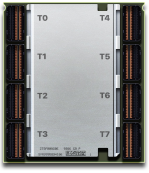
Modern VLSI size and design complexity are growing rapidly.

- Billions of cells in a single design.
- Performance requirements.
- Different design rules and constraints.

Apple A15

15 B

IBM Power10



18 B

Apple M1 Max

57 B

VLSI Placement and Challenges

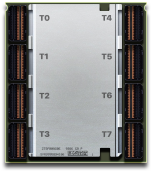
Modern VLSI size and design complexity are growing rapidly.

- Billions of cells in a single design.
- Performance requirements.
- Different design rules and constraints.

Apple A15

15 B

IBM Power10

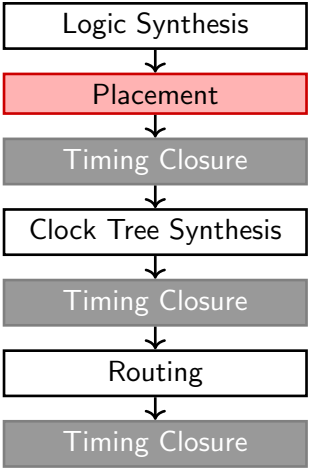


18 B

Apple M1 Max

57 B

Placement and timing closure are challenging tasks in modern VLSI design.



VLSI Placement and Challenges

Good wirelength does not mean good timing.

- Making nets on critical paths shorter is beneficial to timing.

VLSI Placement and Challenges

Good wirelength does not mean good timing.

- Making nets on critical paths shorter is beneficial to timing.
- We have to sacrifice wirelength of those nets sharing common cells.

VLSI Placement and Challenges

Good wirelength does not mean good timing.

- Making nets on critical paths shorter is beneficial to timing.
- We have to sacrifice wirelength of those nets sharing common cells.
- Other paths may become critical after some cell movements during global placement.

VLSI Placement and Challenges

- It is hard to directly optimize timing in normal wirelength-driven placement.

VLSI Placement and Challenges

- It is hard to directly optimize timing in normal wirelength-driven placement.
- Timing-driven placement seeks cell locations to optimize timing violations.

$$s_{\text{TNS}} = \sum_{j \in \mathcal{PO}} s_j, \quad s_{\text{WNS}} = \min_{j \in \mathcal{PO}} s_j. \tag{1}$$

- How can we optimize timing violations in global placement?

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{e \in E} W(e; \boldsymbol{x}, \boldsymbol{y}) + \lambda D(\boldsymbol{x}, \boldsymbol{y}).$$

VLSI Placement and Challenges

- It is hard to directly optimize timing in normal wirelength-driven placement.
- Timing-driven placement seeks cell locations to optimize timing violations.

$$s_{\text{TNS}} = \sum_{j \in \mathcal{PO}} s_j, \quad s_{\text{WNS}} = \min_{j \in \mathcal{PO}} s_j. \tag{1}$$

- How can we optimize timing violations in global placement?

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{e \in E} W(e; \boldsymbol{x}, \boldsymbol{y}) + \lambda D(\boldsymbol{x}, \boldsymbol{y}).$$

  - Provide hints to placer by **net weighting**.

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{e \in E} w_e W(e; \boldsymbol{x}, \boldsymbol{y}) + \lambda D(\boldsymbol{x}, \boldsymbol{y})$$

    so that the placer will try to make nets with higher weights shorter.

Prior Works

- Most timing-driven optimization techniques are performed incrementally.
- Path-based approaches may not be suitable for global placement as the runtime overhead is large.
- Prior works on timing-driven global placement using net weighting are compared in the following table.

Work	Dynamic	Slack-based	History Information
Riess'95	✓		✓
Eisenmann'98	✓		✓
Kong'02		✓	
Kahng'04	✓	✓	
Ren'05		✓	
Ours	✓	✓	✓

Overall Contribution

We integrate timing optimization into **electrostatic-based** global placement.

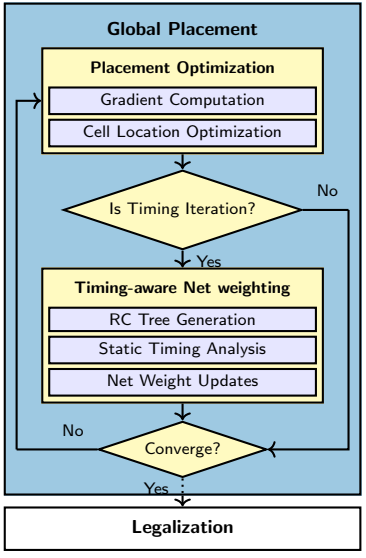
- **Momentum-based net weighting**. At each timing iteration, we expect to assign weights to different nets by incorporating the current slacks within the existing criticality information.
- **Preconditioning technique for net weighting**. The preconditioner proposed by the original ePlace[1] algorithm does not consider different net weights. We enhance the preconditioner to adapt different net weights when optimizing cell locations.

Experimental results using the ICCAD 2015 incremental TDP contest benchmarks show that our net weighting scheme is effective.

[1] Jingwei Lu et al. (2015). "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method". In: 20.2, pp. 1–34.

Outline

Overall Flow



We focus on the following questions.

- How to determine whether to perform timing analysis at each gradient-based iteration.
- How to construct RC trees and feed them into the timer for static timing analysis.
- How to update net weights according to the results of static timing analysis.

Overall Flow

How to determine whether to perform timing analysis at each gradient-based iteration?

- Cell locations at earlier stages of global placement are unreliable.
- We do not have enough budget to optimize cell locations at very late stages of global placement, as they almost converge.
- Timing analysis is time-consuming, so it should not be called too frequently.

Overall Flow

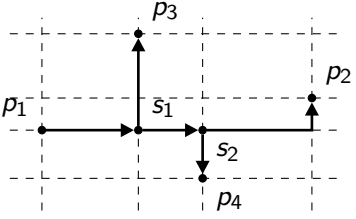
How to determine whether to perform timing analysis at each gradient-based iteration?

- Cell locations at earlier stages of global placement are unreliable.
- We do not have enough budget to optimize cell locations at very late stages of global placement, as they almost converge.
- Timing analysis is time-consuming, so it should not be called too frequently.
- Empirically, we perform timing analysis every 15 iterations after the 500th iteration of global placement, when the overflow is less than 0.5.

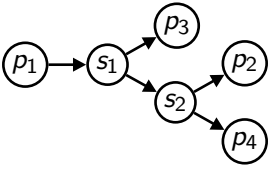
RC Tree construction

How to construct RC trees and feed them into the timer for static timing analysis?

- We need to translate cell locations into certain RC information.
- The Elmore delay model is used. In global placement stage, routing is not done yet, so it is not meaningful to consider very accurate timing model.
- We use FLUTE[2] to generate Steiner trees.



(a) A Steiner tree example



(b) The corresponding RCT

[2] Chris Chu and Yiu-Chung Wong (2007). "FLUTE: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design". In: *IEEE TCAD* 27.1, pp. 70–83.

Net Criticality

Our placement database considers criticality value as a guide to update net weights. Let $c_e^{(m)}$ denote the criticality of net e at the m-th **timing** iteration.

- For arbitrary net e, the initial criticality value $c_e^{(0)}$ defaults to be 0.

Net Criticality

Our placement database considers criticality value as a guide to update net weights. Let $c_e^{(m)}$ denote the criticality of net e at the m-th **timing** iteration.

- For arbitrary net e, the initial criticality value $c_e^{(0)}$ defaults to be 0.
- The momentom of criticality value of net e is defined as

$$c_{\mathrm{mom},e}^{(m)} = \begin{cases} 0, & \text{if } s_{\mathrm{wns}}^{(m)} \geq 0; \\ \max\left\{0, \dfrac{s_e^{(m)}}{s_{\mathrm{wns}}^{(m)}}\right\}, & \text{otherwise,} \end{cases} \tag{2}$$

where $s_e^{(m)}, s_{\mathrm{wns}}^{(m)}$ are the net slack of e and the worst negative slack at the m-th timing iteration, respectively.

Net Criticality

Our placement database considers criticality value as a guide to update net weights. Let $c_e^{(m)}$ denote the criticality of net e at the m-th **timing** iteration.

- For arbitrary net e, the initial criticality value $c_e^{(0)}$ defaults to be 0.
- The momentom of criticality value of net e is defined as

$$c_{\mathrm{mom},e}^{(m)} = \begin{cases} 0, & \text{if } s_{\mathrm{wns}}^{(m)} \geq 0; \\ \max\left\{0, \dfrac{s_e^{(m)}}{s_{\mathrm{wns}}^{(m)}}\right\}, & \text{otherwise,} \end{cases} \tag{2}$$

  where $s_e^{(m)}, s_{\mathrm{wns}}^{(m)}$ are the net slack of e and the worst negative slack at the m-th timing iteration, respectively.
- We care negative slack only.

Net Criticality

How to iteratively update criticality value?

- We consider **logarithm** criticality value instead.

$$\tilde{c}_e^{(m)} = \ln(1 + c_e^{(m)}), \quad \tilde{c}_{\mathrm{mom},e}^{(m)} = \ln(1 + c_{\mathrm{mom},e}^{(m)}). \tag{3}$$

Net Criticality

How to iteratively update criticality value?

- We consider **logarithm** criticality value instead.

$$\tilde{c}_e^{(m)} = \ln(1 + c_e^{(m)}), \quad \tilde{c}_{\mathrm{mom},e}^{(m)} = \ln(1 + c_{\mathrm{mom},e}^{(m)}). \tag{3}$$

- Criticality update rule.

$$\tilde{c}_e^{(m+1)} = \alpha \tilde{c}_e^{(m)} + (1 - \alpha)\tilde{c}_{\mathrm{mom},e}^{(m)}, \tag{4}$$

where $\alpha \in [0, 1]$ is a hyperparameter.

Net Criticality

How to iteratively update criticality value?

- We consider **logarithm** criticality value instead.

$$\tilde{c}_e^{(m)} = \ln(1 + c_e^{(m)}), \quad \tilde{c}_{\mathrm{mom},e}^{(m)} = \ln(1 + c_{\mathrm{mom},e}^{(m)}). \tag{3}$$

- Criticality update rule.

$$\tilde{c}_e^{(m+1)} = \alpha \tilde{c}_e^{(m)} + (1 - \alpha)\tilde{c}_{\mathrm{mom},e}^{(m)}, \tag{4}$$

  where $\alpha \in [0, 1]$ is a hyperparameter.

- Equation (4) considers both history logarithm criticality $\tilde{c}_e^{(m)}$ and the momentum term computed at the current timing iteration, to avoid possible oscillation.

Net Weighting

The core part is to update net weights according to the critivality values.

- Considering that net weights are always positive, we use logarithm net weight, $\tilde{w}_e^{(m)} = \ln w_e^{(m)}$.

[3] Hans Eisenmann and Frank M Johannes (1998). "Generic global placement and floorplanning". In: *Proc. DAC*, pp. 269–274.

Net Weighting

The core part is to update net weights according to the critivality values.

- Considering that net weights are always positive, we use logarithm net weight, $\tilde{w}_e^{(m)} = \ln w_e^{(m)}$.
- The net weighting scheme is determined by $\tilde{w}_e^{(m+1)} = \tilde{w}_e^{(m)} + \tilde{c}_e^{(m)}$.

[3] Hans Eisenmann and Frank M Johannes (1998). "Generic global placement and floorplanning". In: *Proc. DAC*, pp. 269–274.

Net Weighting

The core part is to update net weights according to the critivality values.

- Considering that net weights are always positive, we use logarithm net weight, $\tilde{w}_e^{(m)} = \ln w_e^{(m)}$.
- The net weighting scheme is determined by $\tilde{w}_e^{(m+1)} = \tilde{w}_e^{(m)} + \tilde{c}_e^{(m)}$.
- The scheme also follows $w_e^{(m+1)} = (1 + c_e^{(m)}) w_e^{(m)}$[3], however, the criticality value is calculated according to specific slack values given by STA, and updated differently considering history information.

[3] Hans Eisenmann and Frank M Johannes (1998). "Generic global placement and floorplanning". In: *Proc. DAC*, pp. 269–274.

Net Weighting

The core part is to update net weights according to the critivality values.

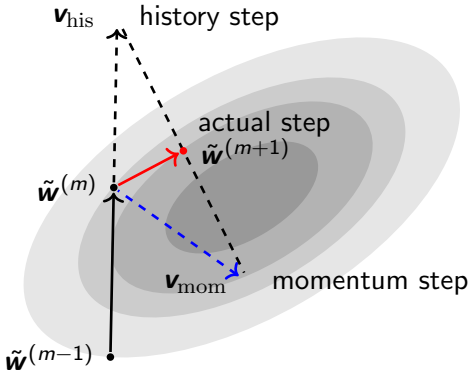
- Considering that net weights are always positive, we use logarithm net weight, $\tilde{w}_e^{(m)} = \ln w_e^{(m)}$.
- The net weighting scheme is determined by $\tilde{w}_e^{(m+1)} = \tilde{w}_e^{(m)} + \tilde{c}_e^{(m)}$.
- The scheme also follows $w_e^{(m+1)} = (1 + c_e^{(m)}) w_e^{(m)}$[3], however, the criticality value is calculated according to specific slack values given by STA, and updated differently considering history information.
- If we write $\tilde{c}_e^{(m)}$ as $\Delta\tilde{w}_e^{(m)}$, we have

$$\Delta\tilde{w}_e^{(m+1)} = \alpha\Delta\tilde{w}_e^{(m)} + (1-\alpha)\tilde{c}_{\mathrm{mom},e}^{(m)}, \tag{5}$$

  which is in fact a momentum-based update.
- we work on all nets instead of those only on critical paths.

[3] Hans Eisenmann and Frank M Johannes (1998). "Generic global placement and floorplanning". In: *Proc. DAC*, pp. 269–274.

Net Weighting



- Prevent drastic update.
- Update net weights based on history steps.
- Adjust weighting update direction according to the momentum step determined by STA results.

Preconditioning

In numerical optimization, preconditioning is a very important step to reduce the condition number of an optimization problem.

- The ePlace[4] preconditioner only considers diagonal entries of the Hessian matrix.

[4] Jingwei Lu et al. (2015). "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method". In: 20.2, pp. 1–34.

Preconditioning

In numerical optimization, preconditioning is a very important step to reduce the condition number of an optimization problem.

- The ePlace[4] preconditioner only considers diagonal entries of the Hessian matrix.
- Let f be the overall objective function of global placement. The i-th entry of the Hessian matrix H_f with respect to horizontal location $\boldsymbol{x}$ is

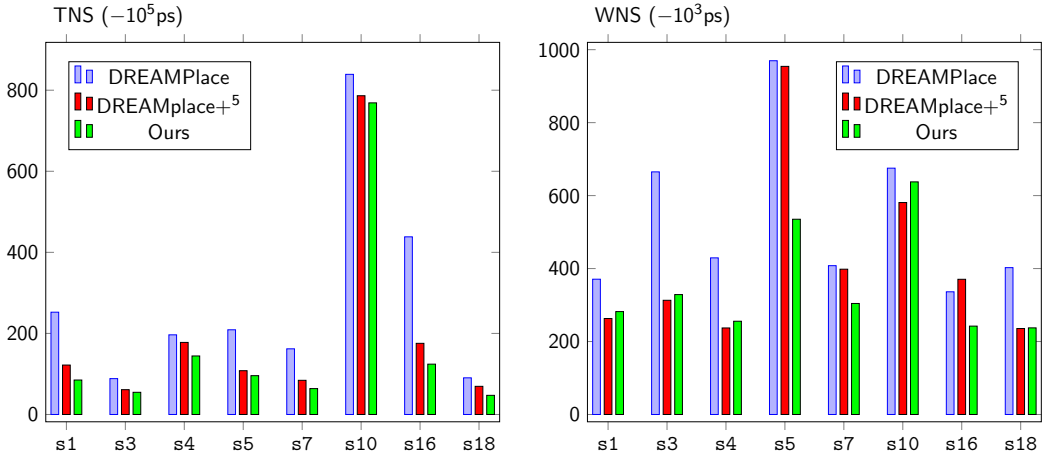
$$\frac{\partial^2 f}{\partial x_i^2} = \sum_{e \in E} w_e \frac{\partial^2 W(e; \boldsymbol{x}, \boldsymbol{y})}{\partial x_i^2} + \lambda \frac{\partial^2 D(\boldsymbol{x}, \boldsymbol{y})}{\partial x_i^2} \approx \sum_{e \in E} w_e + \lambda q_i, \tag{6}$$

  where q_i is the quantity of electrical charge of the i-th node.

[4] Jingwei Lu et al. (2015). "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method". In: 20.2, pp. 1–34.

Outline

Results

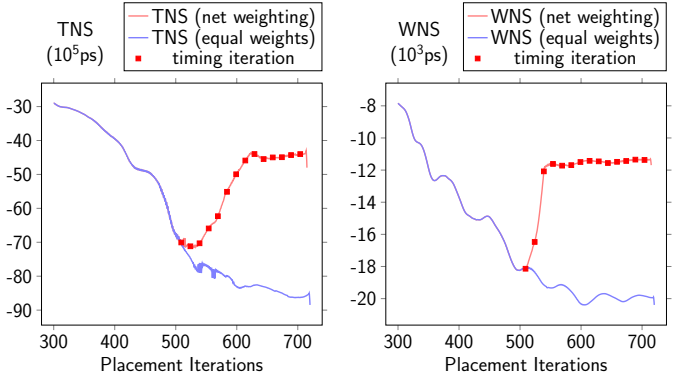


- Compared to DREAMPlace: 47% TNS and 30% WNS improvement.
- Compared to DREAMPlace+5: 20% TNS and 9% WNS improvement.

[5] Hans Eisenmann and Frank M Johannes (1998). "Generic global placement and floorplanning". In: *Proc. DAC*, pp. 269–274.

Results

- We conduct the experiments on the ICCAD2015 contest benchmark suites. The TNS and WNS values at each placement iteration after the 300th iteration for `superblue18` are shown in the following figure.



(c) The TNS curve in placement (d) The WNS curve in placement

Results

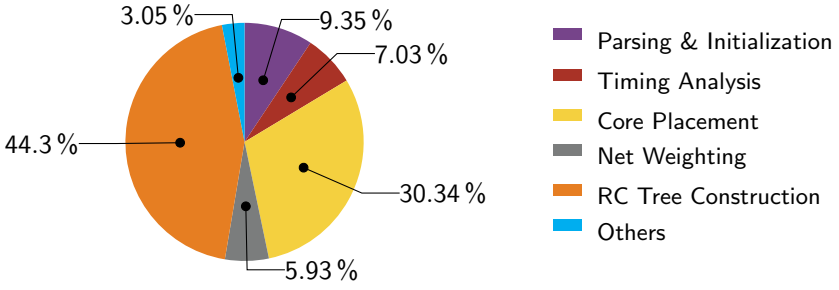
- The wirelength-driven placer is DREAMPlace[6]. The timer we use is OpenTimer[7].
- At nearly every timing iteration, marked with red color, TNS can get improved at once, especially when starting to break the balance of net weights.
- WNS will quickly and significantly be optimized after one or two net weighting steps. After that, it almost remains stable during the later stages of global placement.
- At later stages, other critical or nearly critical paths will be taken more into consideration, and that is an important reason why it is hard to further optimize WNS during global placement.

[6] Yibo Lin et al. (2020). "Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement". In: *IEEE TCAD*.

[7] Tsung-Wei Huang and Martin DF Wong (2015). "OpenTimer: A high-performance timing analysis tool". In: *Proc. ICCAD*, pp. 895–902.

Results

The runtime breakdown on ICCAD2015 contest benchmark `superblue18`.



- We are still facing the runtime bottleneck dominated by the RC tree construction.
- Modifying net weights will also affect the total number of iterations to converge.
- Considering that STA must be called multiple times to incorporate changes of cell locations, the overhead of RC tree construction and STA should be the main focus for acceleration.

Outline

Summary

- We propose a momentum-based net weighting scheme for timing-driven global placement and improve the preconditioner accordingly.
- The evaluation results on ICCAD2015 contest benchmarks show that we can achieve a significant improvement on both TNS and WNS.
- The results of this paper enlighten us that, although most timing-aware optimization methods are performed at incremental stages, it is still very effective to consider timing at the earlier stages of physical design, especially global placement.

Thank You!