DREAMPlace 3.0: Multi-Electrostatics Based Robust VLSI Placement with Region Constraints

Jiaqi Gu\textsuperscript{1}, Zixuan Jiang\textsuperscript{1}, Yibo Lin\textsuperscript{2} and David Z. Pan\textsuperscript{1}

\textsuperscript{1}ECE Department, The University of Texas at Austin
\textsuperscript{2}Peking University

jqgu@utexas.edu
VLSI Placement and Challenges

- Modern VLSI scale and design complexity grow rapidly
  - Billion-cell design
  - More design rules and constraints
  - Higher performance requirements

- Placement plays a critical role in design closures
  - Wirelength
  - Congestion / Routability
  - Timing
  - …

[Courtesy RePIAce]
Recent Development of VLSI Placement

*Data collected from RePIAce [Cheng+, TCAD’18] and [http://vlsi-cuda.ucsd.edu/~ljw/ePlace/](http://vlsi-cuda.ucsd.edu/~ljw/ePlace/) on ISPD 2005 benchmarks*
DREAMPlace Evolution

0.0
• DAC 2019
• DREAMPlace: VLSI placement using DL framework

1.0
• TCAD 2020
• Improved kernels; Routability-driven placement

2.0
• TCAD 2020
• ABCDPlace: accelerated detailed placement

3.0
• ICCAD 2020
• Multi-electrostatics-based placement
Placement with Region Constraints

- Place cells with the same function in a confined subregion
  - Support voltage islands
  - Improve manufacturability
  - Reduce datapath delay
  - Decrease clock power

- Fence region
  - Member-hard and non-member-hard
  - Cell assignment is exclusive
  - Hard constraints

- Severe quality loss if not considered
Placement Formulation with Fence Region

\[
\min_{x,y} \sum_{e \in E} \text{WL}(e; x, y) \quad \rightarrow \quad \min_v \sum_{e \in E} \text{WL}(e; v) + \langle \lambda, D(v, r) \rangle
\]
\[
\text{s.t.} \quad D(x, y) \leq \hat{D},
\]
\[
v_k = (x_k, y_k) \in r_k, \quad k = 0, \ldots, K
\]
\[
\lambda = (\lambda_0, \cdots, \lambda_K)
\]
\[
D(v, r) = (D(v_0, r_0), \cdots, D(v_K, r_K))
\]

Previous solutions

- NTUplace4dr: region-aware clustering + new wirelength model [Huang+, TCAD’18]
- RippleDR: upper-bound-lower-bound + look-ahead legalization [Chow+, SLIP’17]
- ePlace-family: not supported

Challenge: Efficient and robust region-aware placement with a global view
Intuition Behind Cell Assignment

- Clustering & Partitioning [NTUplace4dr]
  - Local view ×
  - Region capacity aware ✓
  - Suboptimal solution ×
Cell Assignment via Multi-Electrostatics

- Multi-electrostatic system
  - Global view for cell assignment ✓
  - Low computation complexity ✓
  - Region capacity aware ✓
Proposed Method

- Multi-Electrostatics based placement

Subregion Partitioning

Insert Virtual Blockage

Isolated Density Optimization

Individual Density Weight Scheduling

Robust Optimizer with Entropy Injection

\[ O \left( \sum_{r_k} |V_k| \right) = O(|V|) \]
Virtual Blockage Insertion

- Virtual blockage insertion
  - Rectangle slicing

Region mask $\rightarrow$ Slicing $\rightarrow$ Merging

ISPD2015 superblue_16a

$\hat{D}_k = \max \left( \text{LocalAreaUtil} + \epsilon, \hat{D} \right) = \max \left( \frac{\text{Area}(v_k)}{\text{Area}(r_k \setminus m)} + \epsilon, \hat{D} \right)$
**Quadratic Density Penalty**

- Modified augmented Lagrangian formulation [Zhu+, DAC 2018]
  \[
  f = \sum_{e \in E} W(e; v) + \left< \lambda, D(v, r) + \frac{1}{2} \mu \mathcal{P}_\lambda \otimes D^2(v, r) \right>
  \]

- Wirelength [Hsu+, TCAD 2013]
  - Weighted-average WL model with smoothness control

- Quadratic term
  - Accelerate initial spreading

- Density weight \(\lambda = (\lambda_0, \cdots, \lambda_K)\)
  - Independent for each region
  - Also controls quadratic term
Density Weight Scheduling

- Update Lagrangian multiplier $\lambda$
  - Normalized preconditioned sub-gradient descent
    \[ \hat{\nabla}_\lambda f = \nabla_\lambda f \odot \mathcal{P}_\lambda \]
    \[ \lambda \leftarrow \min \left( \lambda_{\text{max}}, \lambda + \alpha \frac{\hat{\nabla}_\lambda f}{\|\hat{\nabla}_\lambda f\|_2} \right) \]

- Adaptive step size $\alpha$
  - Exponentially increased step size based on density
    \[ \alpha \leftarrow \gamma(D, \mathcal{P}_\lambda) \alpha \]
Preconditioned Nesterov’s Optimizer

- Multi-field divergence-aware preconditioning
  - Stabilize optimization for the exterior region
    \[
    \hat{\nabla} f = \nabla f \odot \mathcal{P}
    \]
    \[
    \mathcal{P}_K = \min \left( 1, \left( \nabla^2_{\varphi_K} \sum WL(e, v) + \beta \lambda_K \nabla^2_{\varphi_K} \mathcal{D}(v_K, r_K) \right)^{-1} \right)
    \]
- Wirelength Hessian [Courtesy ePlace]
  - Estimate the diagonal by pin count of an instance
- Density Hessian [Courtesy ePlace]
  - Estimate the diagonal by instance area
- Exponentially increased $\beta$ factor to slow down large-cell movement
**Intuition Behind Optimizer Robustness**

- **Slow convergence**
  - Slow spreading
  - 30%-50% runtime for spreading

- **Optimizer divergence**
  - Stagnant density overflow
  - Increasing wirelength

- **Stuck in saddle-point**
  - Saddle-point circle that harms the HPWL

---

- 200 iter
- 400 iter

---

![Diagram showing 200 and 400 iter comparisons](ISPD'19 test1)

---

![Diagram showing wirelength and overflow comparison](ISPD'2015 mgc_fft_1)

---

![Diagram showing density and cell location](ISPD'19 test1)
Robust Placement

- Adaptive quadratic penalty and entropy injection
  - Window-based plateau detector
    \[
    \text{PLT} = \begin{cases} 
    \frac{\max_L (OVFL) - \min_L (OVFL)}{\text{avg}_L (OVFL)} & < \delta_{PLT}, \quad OVFL > 0.9 \\
    \text{False,} & OVFL \leq 0.9, 
    \end{cases}
    \]
  - Quadratic penalty with doubled density weight if triggered
  - Entropy injection as location perturbation and shrinking
    - Escape saddle-point
    - Faster convergence
  - Divergence-aware rollback

- Divergence-aware rollback
Post-GP Placement

- Fence region aware legalization
  - Per region greedy legalization ($g1$) with virtual blockage

\[ v^g_k \leftarrow g1(v^m_k, m, b_k) \]

- Abacus ($al$) [Spindler+, ISPD’08] algorithm to minimize displacement with virtual blockage

\[ \tilde{v}_k \leftarrow al(v^m_k, v^g_k, m, b_k) \]

- Finish the flow with detailed placement using ABCDPlace [Lin+, TCAD 2019]
  - Support fence region constraints
Experimental Setup

♦ Machine
  › Intel Core i9-7900X CPUs (3.3 GHz and 10 cores)
  › 128 GB RAM
  › NVIDIA TitanXp GPU

♦ Benchmark suits
  › ISPD 2015
  › ISPD 2019 (used as placement benchmarks)
  › ICCAD 2014

♦ Baseline
  › DREAMPlace [Lin+, DAC 2019] and ABCDPlace [Lin+, TCAD 2020]

♦ Placers for comparison
  › NTUplace4dr [Huang+, TCAD 2018]
  › Eh?Placer [Darav+, TODAES 2016]
  › DREAMPlace [Lin+, DAC 2019]
DREAMPlace3.0 significantly outperforms other region-aware placers on ISPD15

- 20.6% better than Eh?Placer
- 13.3% better than NTUplace4dr
DREAMPlace3.0 outperforms other placers on ISPD15

- 17.0% better than Eh?Placer
- 7.4% better than NTUplace4dr
- 1.2% better than DREAMPlace
DREAMPlace3.0 outperforms other region-aware placers on ISPD15

- 12.4% better than Eh?Placer *reported by NCTU-GR [Dai+, TVLSI 2012]
- 11.2% better than NTUplace4dr

Top 5 OVFL Comparison (w/ Region)
DREAMPlace3.0 outperforms other region-aware placers on ISPD15:

- 3.8% better than Eh?Placer
- 2.9% worse than NTUplace4dr
- 3.3% better than DREAMPlace
On ISPD 2015 (w/ region), GPU-based DREAMPlace 3.0 is

- 3.7× faster than 8-threaded Eh?Placer
- 34.8× faster than 8-threaded NTUplace4dr

On ISPD 2015 (w/o region), GPU-based DREAMPlace 3.0 is

- 13.9× faster than 8-threaded Eh?Placer
- 37.8× faster than 8-threaded NTUplace4dr
- 1.9% faster than DREAMPlace

On ISPD 2019 and ICCAD 2014, GPU-based DREAMPlace 3.0 is

- 10.8% faster than DREAMPlace
- More stable in convergence with similar solution quality
Conclusion and Future Direction

♦ Conclusion
  › **Multi-electrostatics system**: handle fence region constraints with a *global view*
  › **Virtual blockage and field isolation**: parallel multi-region placement
  › **Adaptive quadratic penalty and entropy injection**: more stable convergence
  › >13% better HPWL and 11% better *overflow* than region-aware placers
  › 10% faster and more stable than DREAMPlace

♦ Future direction
  › Honor more placement constraints
  › Other optimization algorithms
  › New acceleration methods in multi-field placement
Thank you!