GrandPlan: Differentiable, Simultaneous Top-Level Floorplanning and Partition-Level Cell Placement for Large-Scale IP-Cores

Zhili Xiong[1], Yi-Chen Lu[2], David Z. Pan[1], Haoxing Ren[2]

The University of Texas at Austin[1], Nvidia[2]

Agenda

Motivation

- The problem statement & related work

Methodology

- Flat IP-cores level placement
- Boundary extraction and refinement
- Fence-region placement

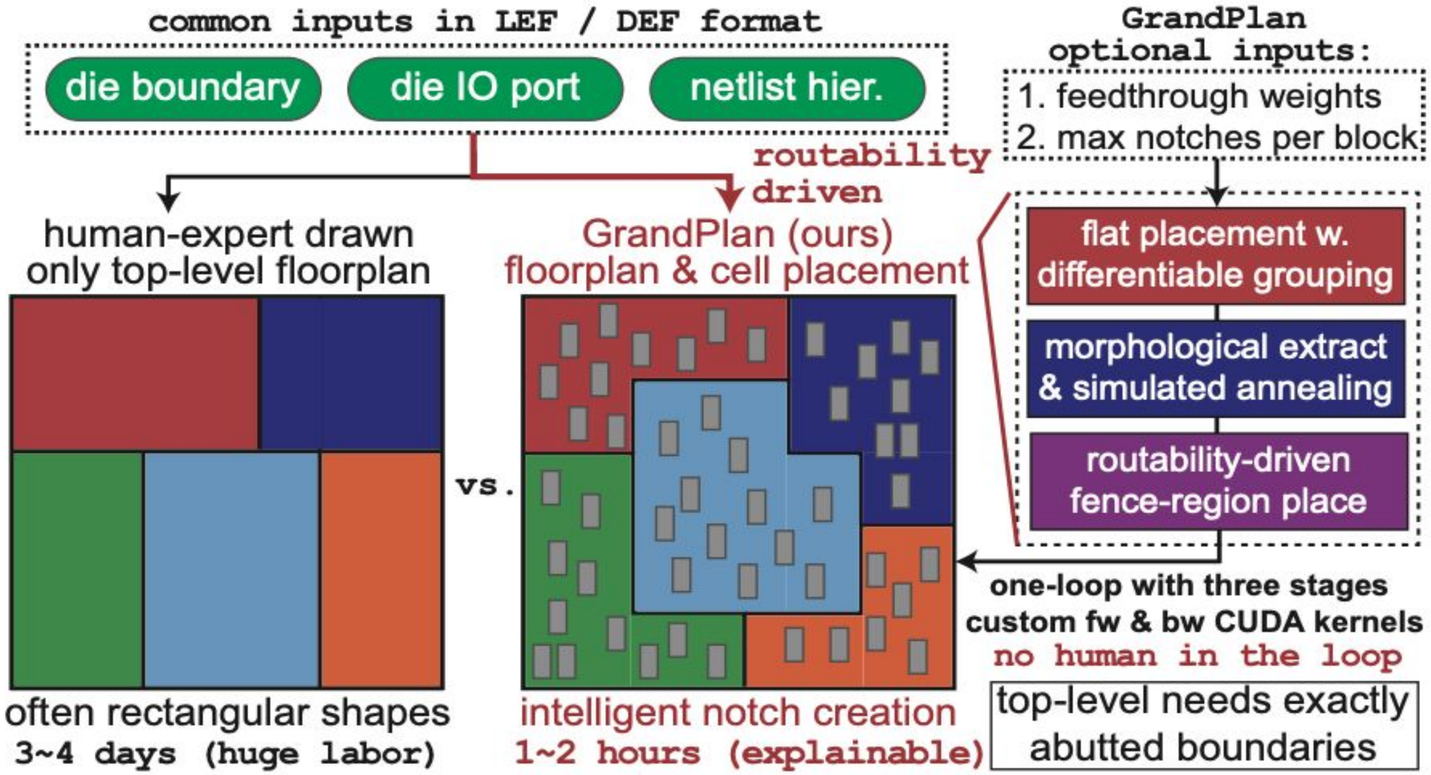
Experiments

- Main results from placement evaluation
- Ablation studies

Motivation

Problem statement

- Existing top-level chiplet floorplanning is labor-intensive and sub-optimal, where engineers are hand-drawing **rectangular** partition shape based on prior knowledge.
- No existing commercial tool can handle >20 million cell placement, thus it's limited to the **two stage: floorplan partition-level placement** scheme
- Globally optimal partition-shape would not be rectangular **intelligent notch creation** will improve connectivity



Motivation

Related work

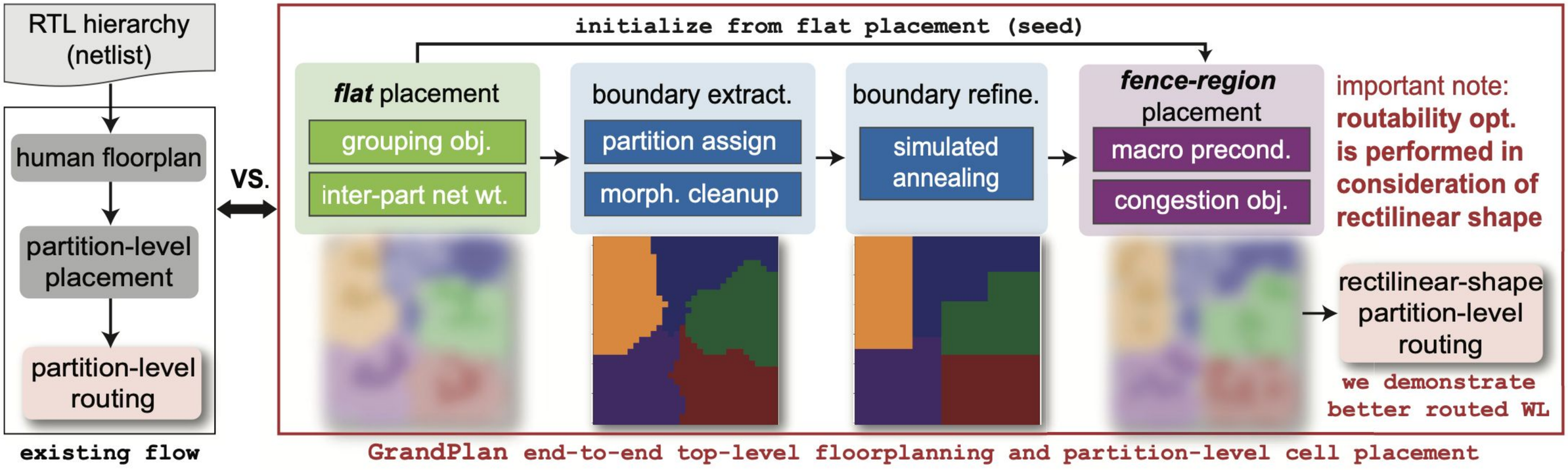
Existing automation frameworks considers **only inter-partition connectivity**, which prevents a full-design view and can lead to suboptimal PPA.

Category	Paper	Partition Shape	Methodology	Consider intra-partition connectivity?
Packing-based	Sequence-pair [ICCAD'95], BSG [ASPDAC'96]	Rectangular	Constraint graph, longest path algorithm	N
	O Tree [DAC'99], B* Trees [DAC'00], MDP Tree [ASPDAC'19]	Rectangular/Rectilinear	Tree construction, simulated annealing	N
Connectivity-driven	UFO [ASPDAC'10] [TCAD'11]	Rectilinear	AR/PP model in global floorplan, legalization	N
	SDP [DAC'23]	Rectilinear	SDP in global floorplan, legalization	N
	Soft Modules [ICCAD'24]	Rectilinear	Differentiable area in global floorplan, legalization	N
Learning-based	GoodFloorplan [TCAD'21] etc	Rectilinear	RL for local heuristic search	N

Methodology

Overview

Grandplan - Simultaneous placement and floorplanning overview

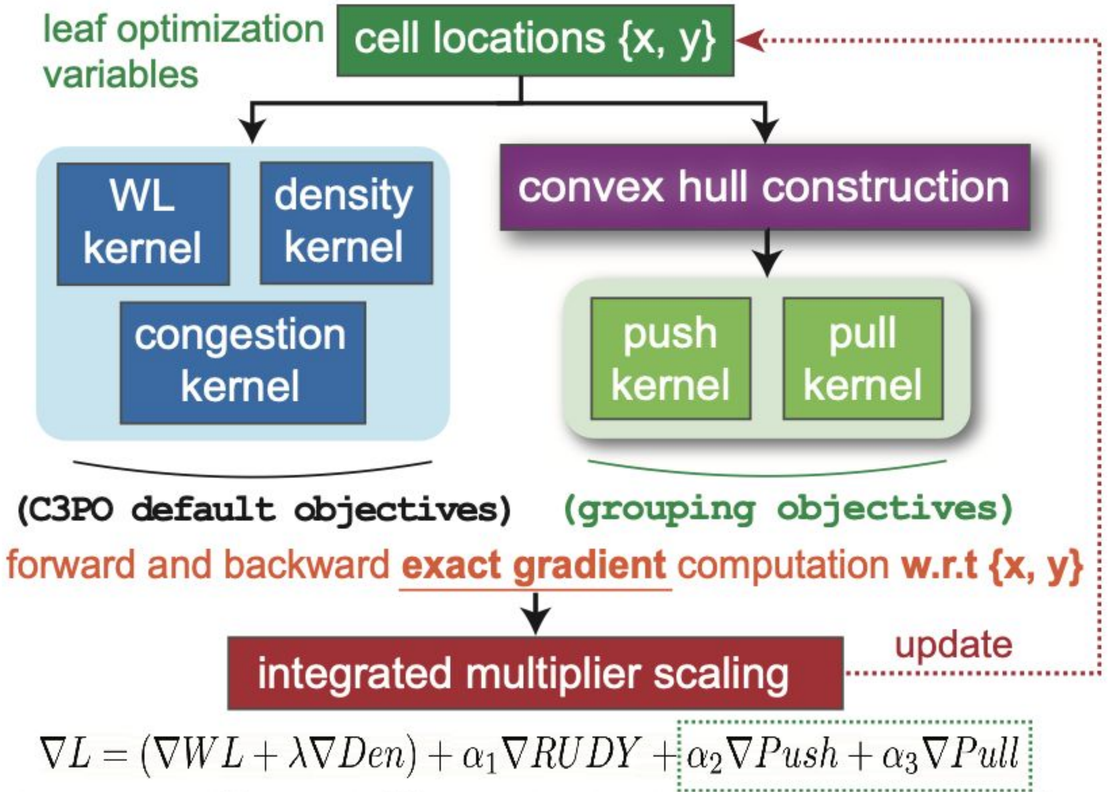


Flat Placement

Overview

Flat placement formulation

- Leverage WL, density, and congestion kernels from C3PO [ASPDAC'26]
- Introduce **grouping objectives** to derive meaningful floorplan boundaries directly from the flat placement solution.
- Macro Handling (Two-Stage Flow)
  - Perform mixed-size global placement followed by macro legalization.
  - Fix macro positions and re-run global placement for std. cells to finalize placement quality.



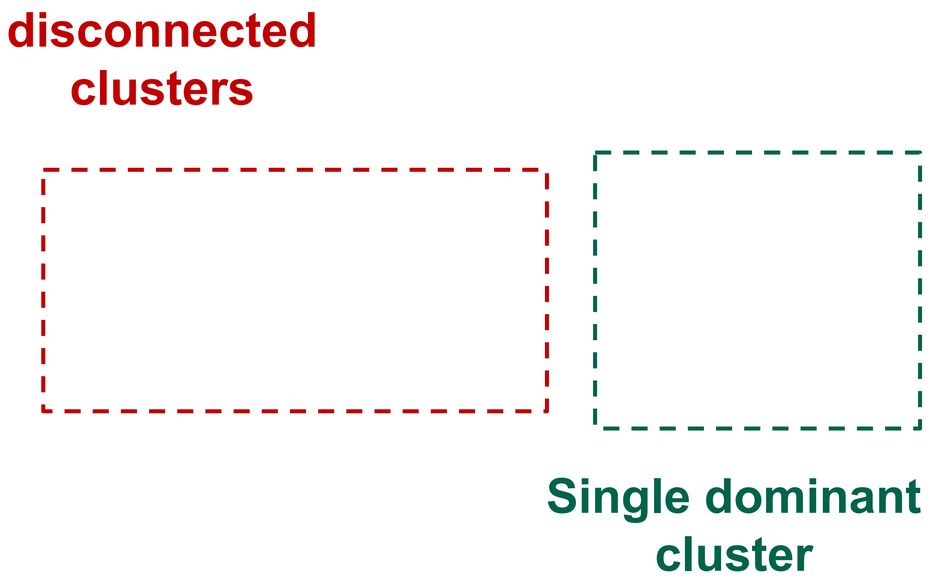
Yi-Chen Lu, Hao-Hsiang Hsiao, Rongjian Liang, Wen-Hao Liu, and Haoxing Ren.2026. "C3PO: Commercial-Quality Global Placement via Coherent, Concurrent Timing, Routability, and Wirelength Optimization"

Naïve DREAMPlace is not Enough

Grouping objectives

Grouping Objectives in flat placement

- **Soft** grouping constraints encourage instances from the same hierarchy to be placed within the same cluster
- Construct **convex hulls** during flat placement iterations to capture evolving partition shapes
- Grouping objectives
  - **Push-pull**: quadratic pushing and pulling force



W/o grouping

With grouping

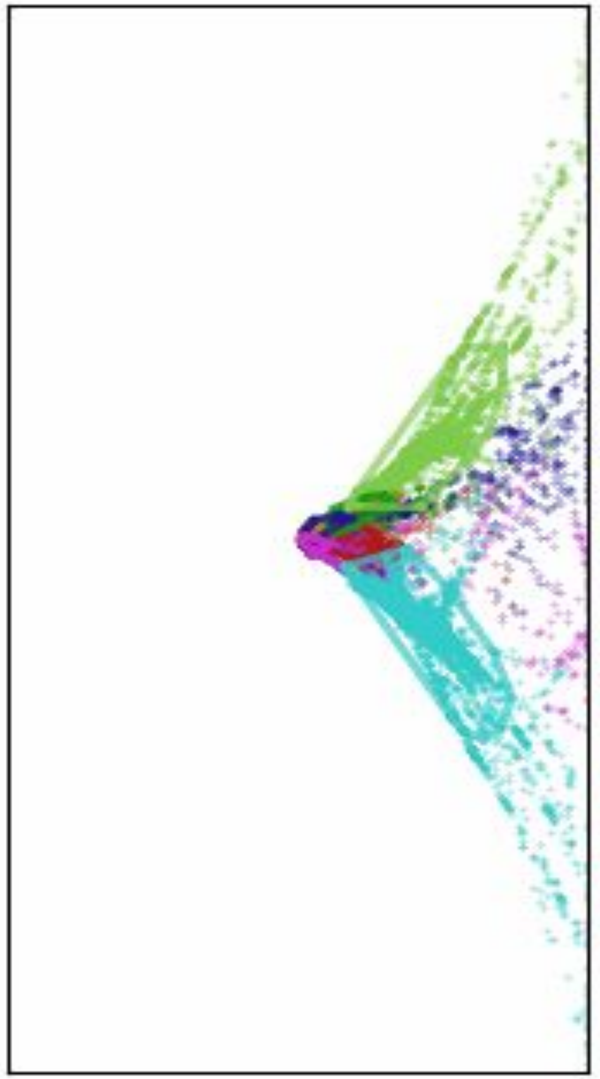
Convex-hull Construction

Convex-hull construction

Convex-hull construction

- Directional-extrema based points extraction
  - Filtering out outlier cells/noise during global placement, and reduce the problem size
  - Algorithm:
    1. For each partition, iterating m unit directions $u_j = (cos\theta_j, sin\theta_j), \theta_j = 2\pi j/m$
    2. Project all points x_j onto u_j: $s_{ij} = u_j.x_i$
    3. Trim extrema: take q-quantile of s_{ij} and collect the points near that quantile, which kills outliers
    4. Union all the candidates across all the directions and de-duplicate
- Accelerated on GPU

Quick-hull algorithm on extracted points



all IO ports are fixed on the right

Grouping objectives

Quadratic push-pull forces

Push-pull

- Quadratic distance from hull anchor points to partition instances

Forward:

$$\mathrm{Pull}(c_k) = \frac{\alpha_{\mathrm{pull}}}{2} \|\mathbf{x}_c - \Pi_{H_k}(\mathbf{x}_c)\|_2^2 \, \mathbf{1}_{\{\mathbf{x}_c \notin \Omega_k\}}$$

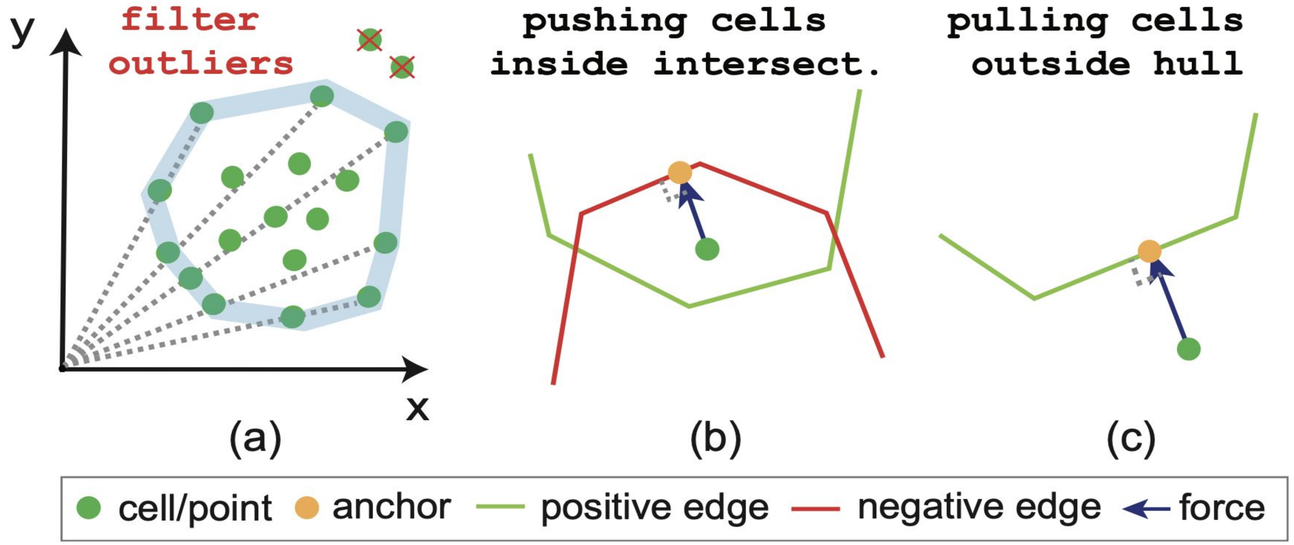
$$\mathrm{Push}(c_k) = \frac{\alpha_{\mathrm{push}}}{2} \sum_{s \neq k} \|\mathbf{x}_c - \Pi_{\partial H_s}(\mathbf{x}_c)\|_2^2 \, \mathbf{1}_{\{\mathbf{x}_c \in \Omega_s\}}.$$

Backward:

$$\nabla_{\mathbf{x}_c} \mathrm{Pull} = \alpha_{\mathrm{pull}} \left(\mathbf{x}_c - \Pi_{H_k}(\mathbf{x}_c)\right) \mathbf{1}_{\{\mathbf{x}_c \notin \Omega_k\}}$$

$$\nabla_{\mathbf{x}_c} \mathrm{Push} = \alpha_{\mathrm{push}} \sum_{s \neq k} \left(\mathbf{x}_c - \Pi_{\partial H_s}(\mathbf{x}_c)\right) \mathbf{1}_{\{\mathbf{x}_c \in \Omega_s\}}.$$

- Anchor points
  - Push anchors - pushing cells in the intersection area to closest negative edge
  - Pull anchors - pulling cells in the outside area to closest positive edge

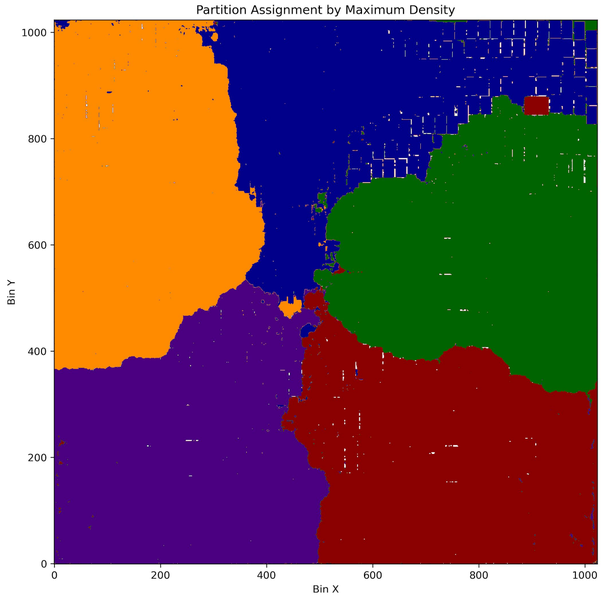


Boundary Extraction

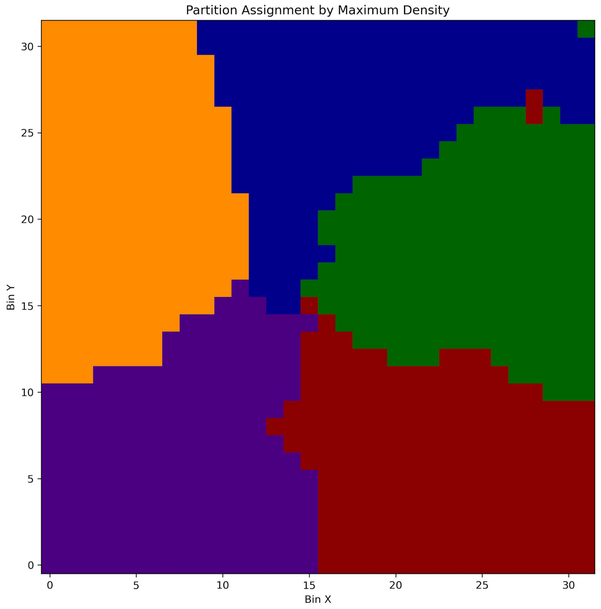
Boundary extraction

Morphological Boundary Extraction

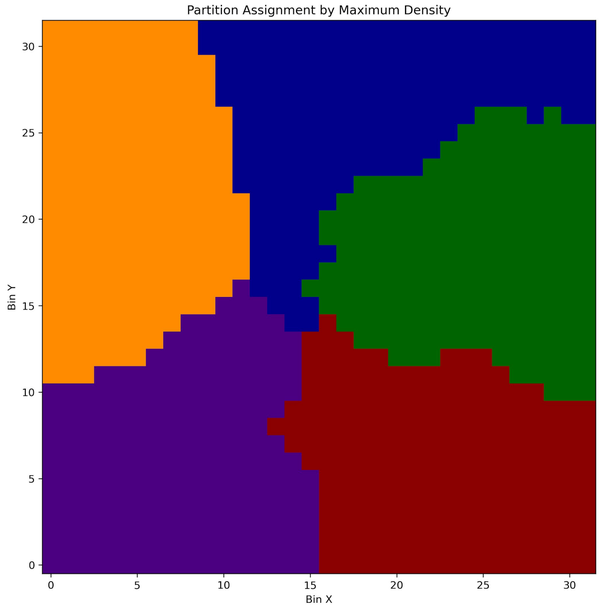
- **Step1** : Partition assignment from density map by majority vote
- **Step2** : Down sample the partition assignment to coarser grids by majority vote
- **Step3** : Morphological cleanup for each partition, and keep the largest connected bin cluster



Step 1: dim1024x1024



Step 2: dim 32x32



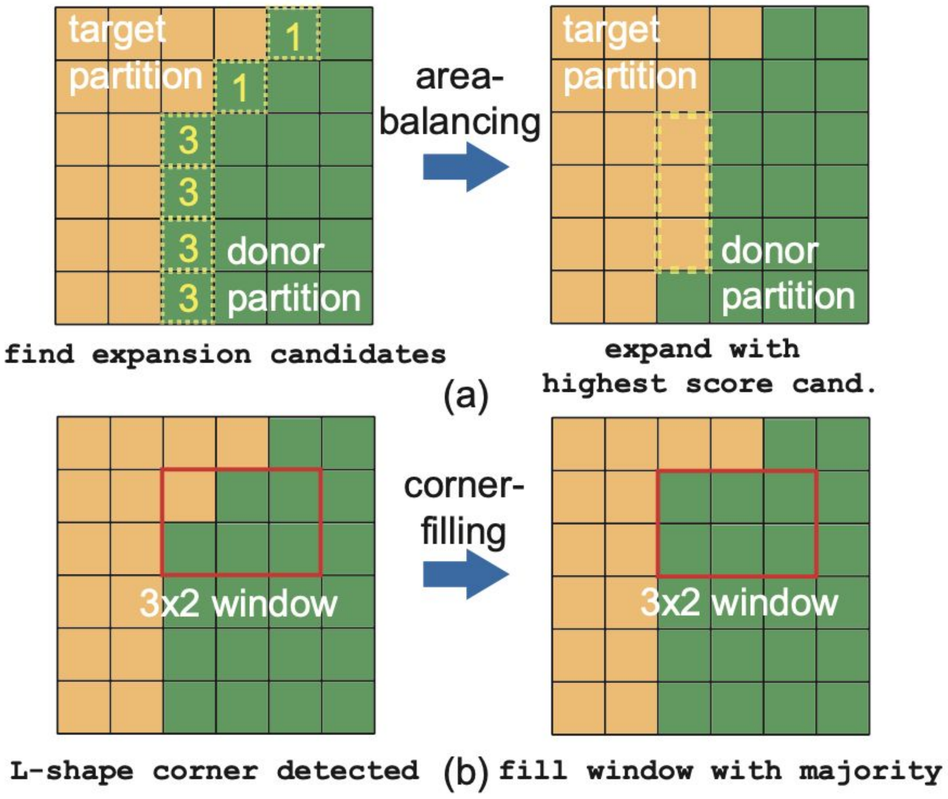
Step 3: dim32x32

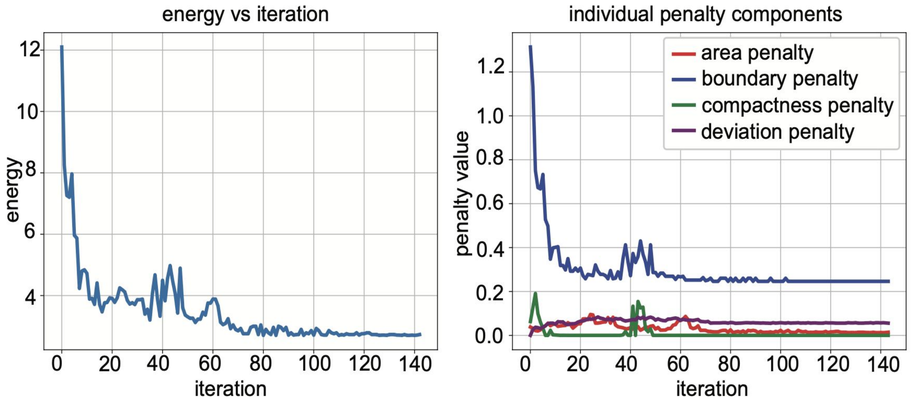
Boundary Refine

Simulated Annealing Based Boundary Refinement

- Energy definition
  - **Area penalty –** penalize the partition area falls below the minimum threshold
  - **Boundary penalty –** discourage excessive notch shapes along the shared boundary
  - **Compactness penalty –** encourage rectangular shaped partitions
  - **Deviation penalty** – the refined bin map does not deviate excessively from the initial starting point

- Move types
  - **Area balancing** – addresses violations of the minimum-area constraint.
  - **Corner filling** – aims to smooth irregular partition boundaries

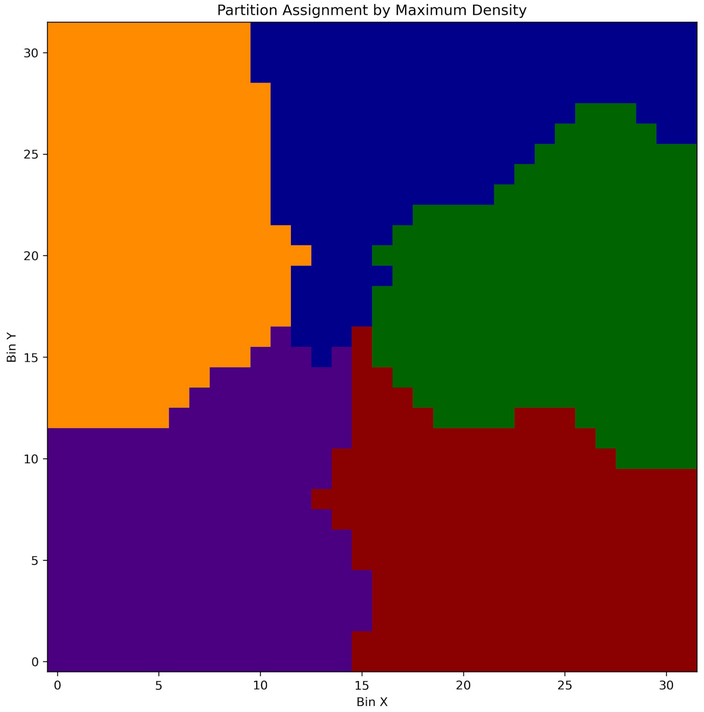




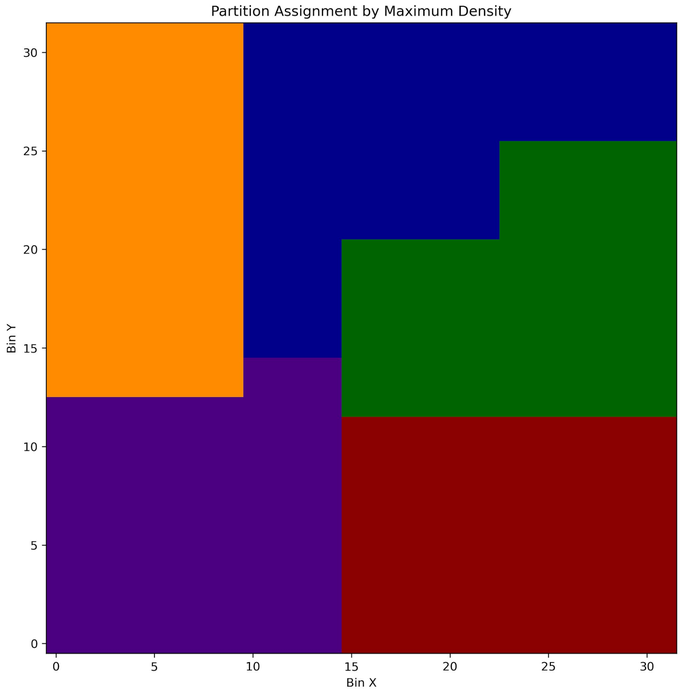
Boundary Extraction

Boundary Refine

Refined boundary with less irregular shapes



Simulated Annealing



Initial boundary

Best boundary

Fence-region Placement

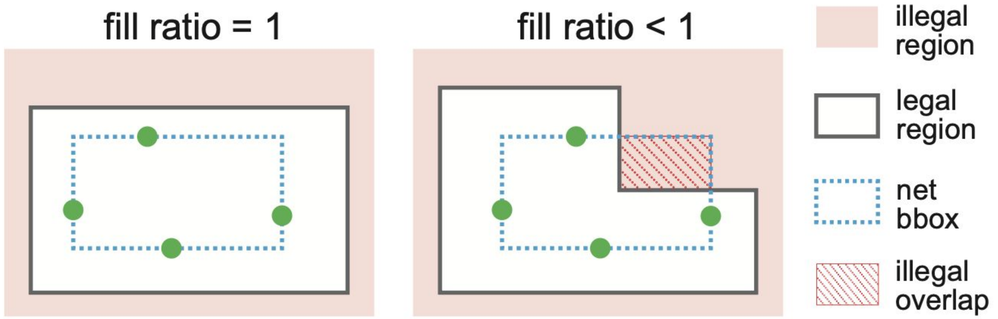
Routability-aware fence-region placement

Starting from the **flat placement solution**

Treat the refined partition shapes as fixed (hard) boundaries during optimization

- **Multi-electrostatics** based formulation
  - Macro pre-conditioning for high density force at boundary
  - Differentiable region-wise illegal RUDY cost – Remove the overlap between **intra-partition nets** and **illegal** partitions

$$\min_{x,y}\left(\sum_{e\in E} W_e(x,y) + \left\langle \lambda \cdot D(x_k, y_k, k)\right\rangle + \alpha_1 \sum_{k\in\mathcal{P}} \mathrm{RUDY}_k^{\mathrm{illegal}}(x_k, y_k)\right).$$



Forward:

$$\mathrm{RUDY}_n^{\mathrm{illegal}}(i,j) = H_n(i,j) + V_n(i,j), \quad (i,j)\in B_k,\ n\in E_k.$$

Backward:

$$\nabla_{x_p}\,\mathrm{RUDY}^{\mathrm{illegal}} = \frac{w_n}{y_{\mathrm{span},n}}\frac{\partial \mathrm{ov}_n}{\partial x_p} + \frac{w_n}{x_{\mathrm{span},n}^2}\left(x_{\mathrm{span},n}\frac{\partial \mathrm{ov}_n}{\partial x_p} - \mathrm{ov}_n\frac{\partial x_{\mathrm{span},n}}{\partial x_p}\right)$$

Experimental setup

Machine

- GPU: NVIDIA A100 (PG506-230)100GB
- CPU: AMD EPYC 7742 64-Core Processor 3.4GHz

Metrics

- **IP-core–level cell HPWL**, **inter-partition HPWL**, a placer-estimated **RouteOVFL**, and total **runtime**.

Benchmarks

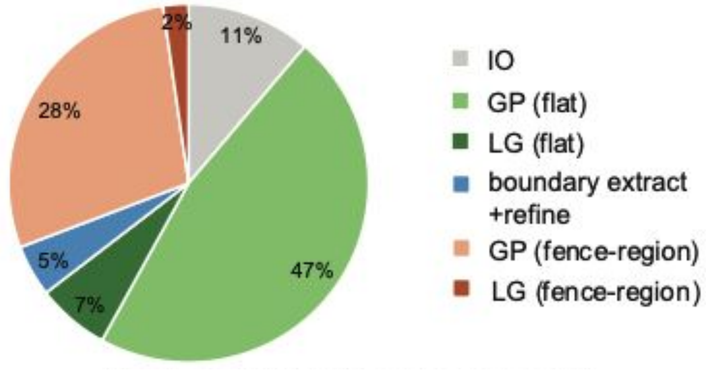
- **8 large-scale designs**, ranging from **2 to 20 million cells**, with **3 to 8 partitions** per design.

Comparison

- **Flat placement**: Flat placement without region constraint
- **Expert-crafted**: Center initialization, fence-region placement with human-drawn boundary
- **Refined**: Flat placement initialization, fence-region placement with refined rectilinear boundary

Experimental results

Placement HPWL



Main results from placement evaluation

- 4.2% (up to 13.5%) improvement on HPWL, and 27.2% improvement on inter-partition HPWL
- On average ~1.2 hours runtime

Table 1: Comparison of placement HPWL(scaled), inter-partition HPWL and RouteOVFL (placer evaluation).

design	flat placement*			expert-crafted region			ours			
name (#cells/#parts)	HPWL ($\times 10^6$)	$HPWL_{inter}$ ($\times 10^6$)	RouteOVFL	HPWL ($\times 10^6$)	$HPWL_{inter}$ ($\times 10^6$)	RouteOVFL	HPWL ($\times 10^6$)	$HPWL_{inter}$ ($\times 10^6$)	RouteOVFL	Runtime (seconds)
testcase1 (20.4M/6)	3536.0	143.0	0.37	3974.2	422.1	0.44	3439.1	204.1	0.36	13239
testcase2 (20.0M/5)	4304.5	154.7	0.49	4481.8	137.2	0.52	4084.1	177.9	0.47	6117
testcase3 (9.90M/8)	1968.6	36.5	0.25	2132.7	107.8	0.29	2049.0	116.5	0.38	3176
testcase4 (18.0M/6)	3728.3	34.8	0.41	4103.9	155.8	0.44	3849.9	170.9	0.42	5331
testcase5 (2.4M/3)	569.8	5.3	0.15	492.9	42.3	0.20	518.3	17.2	0.21	1537
testcase6 (7.9M/6)	2199.1	64.1	0.21	2627.2	567.4	0.23	2856.9	360.9	0.33	3360
testcase7 (11.0M/8)	3015.4	131.4	0.31	3733.1	513.5	0.31	3287.5	367.7	0.33	4227
testcase8 (9.8M/6)	2531.7	49.8	0.22	2569.3	571.1	0.27	2539.0	328.1	0.22	3410
Geo. Mean.	0.928	0.235	0.880	1.000	1.000	1.000	0.958(−4.2%)	0.728(−27.2%)	1.024	–

*flat placement**: For reference only. The flat placement does not honor region constraints and is not legal; it is shown solely to illustrate a near–global-optimal result.

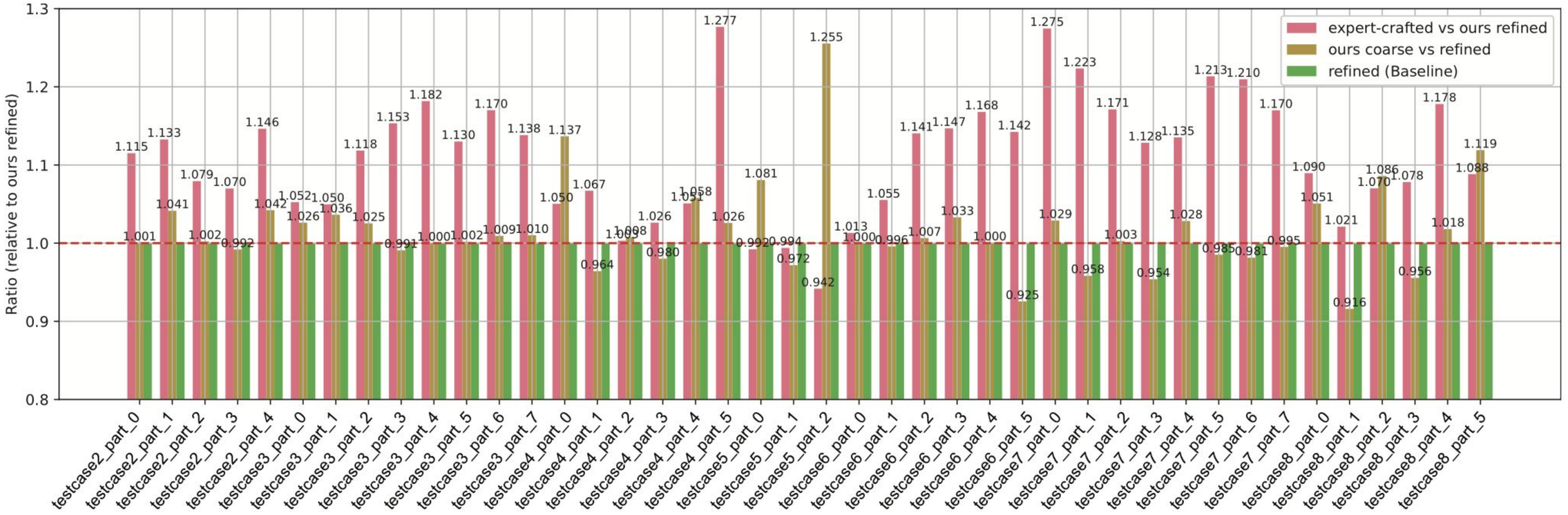
–: Expert-crafted regions are typically labor-intensive and require 1–2 weeks.

Experimental results

Ablation study: partition-level Routed WL

Use all the partition-level DEF files for routing (except testcase1 due to lack of macro pin definitions)

- All inter-partition nets are omitted
- Ours coarse is **1.5%** worse than refined
- Expert-crafted is **10.8%** worse than ours refined



Experimental results

Ablation study: net-weighting and initialization

Effect of extra weighting on inter-partition nets

- No effect on total HPWL
- Improved inter-partition HPWL

Effect of floorplan shapes and seed initialization

- Seed initialization improved the HPWL even with the expert-crafted region
- Only take our floorplan shapes didn't show improvement on the total HPWL

Table 2: Effect of net weighting.

design	w/o net weighting		with net weighting	
	HPWL ($\times 10^6$)	$HPWL_{inter}$ ($\times 10^6$)	HPWL ($\times 10^6$)	$HPWL_{inter}$ ($\times 10^6$)
testcase1	3463.0	367.3	3439.1	204.1
testcase2	4008.8	220.4	4084.1	177.9
testcase3	2092.9	217.7	2049.0	116.5
testcase4	3797.6	287.7	3849.9	170.9
testcase5	522.3	18.7	518.3	17.2
testcase6	2979.4	583.2	2856.9	360.9
testcase7	3207.7	533.9	3287.5	367.7
testcase8	2535.3	361.2	2539.0	328.1
Geo. Mean.	1.002	1.451	1.000	1.000

Table 3: Effect of floorplan shapes and seed initialization.

design	HPWL ratios			
	expert	ours (floorplan-only)	expert (seed flat-place)	ours
testcase2	1.097	1.098	1.063	1.000
testcase4	1.066	1.065	1.055	1.000
testcase7	1.136	1.133	1.027	1.000
Geo. Mean.	1.099	1.098	1.048	1.000

Conclusion

Conclusion

- We presented *GrandPlan*, a GPU-accelerated, differentiable, end-to-end framework that co-optimizes top-level floorplanning and partition-level placement within a single automated loop.

Future work

- **Timing/power-aware extensions** – Reducing total wirelength and feedthrough wirelength can improve the quality of critical inter-partition nets.
- **I/O and interface co-optimization** – A global view enables better boundary pin assignment and can be incorporated as feedback during boundary refinement.

Thank you