Cypress: VLSI-Inspired PCB Placement with GPU Acceleration

Niansong Zhang*, Anthony Agnesina, Noor Shbat, Yuval Leader, Zhiru Zhang,
Haoxing (Mark) Ren | **ISPD 2025**

Cornell University, NVIDIA

Best Paper Nominee

*This work is conducted during an internship at NVIDIA

ACKNOWLEDGEMENT

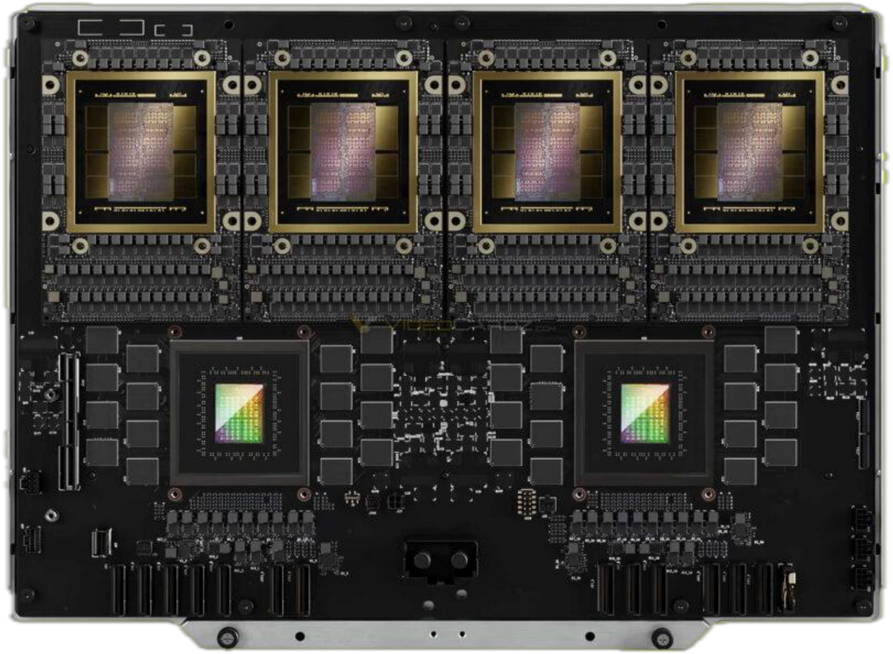
Many thanks to production teams and internal collaborators on this work

OUTLINE

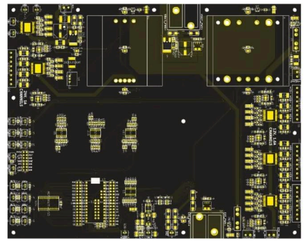
- **Introduction**
- VLSI-Inspired PCB Placement with GPU Acceleration
- Experimental Results
- Conclusion

PCB PLACEMENT AUTOMATION

- Printed Circuit Boards (PCB) have increased significantly in scale, modern HPC and AI PCBs have **>10,000** components.
- However, the PCB layout tasks is typically carried out manually.
- **~50%** of the total PCB design time is spent on layout[1].
- **~$46B** annual global spend on designing circuit boards[1].
- Despite decades of research, PCB placement automation remains slow and limited in quality and scalability.



The Scale of Current Commercial PCB: >10,000 components
NVIDIA GB200 NVL4 PCB



The Capability of SoTA Automatic Placer: <500 components
Example Design from JITX[2]

[1] QUILTER AUTOMATED CIRCUIT BOARD LAYOUT. Link: https://www.quilter.ai/technology. Accessed: 02/25/2025.
[2] AUTOMATED PCB DESIGN USING ARTIFICIAL INTELLIGENCE (AI). Link: https://pallavaggarwal.in/automated-pcb-design-using-ai/. Accessed: 02/25/2025.

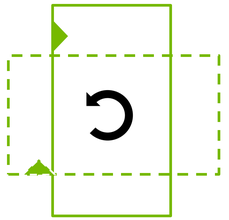
CHALLENGES: PCB PLACEMENT AUTOMATION

- Complex Design Space
  - Components have diverse **sizes** and **shapes**.
  - Components can be **rotated**.
  - Surface-mounted devices (SMDs) can be placed on **two sides**.

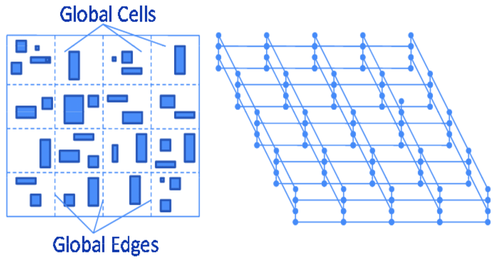
- Limited Routing Resources
  - PCB has **fewer** copper layers for routing, unlike VLSI which easily has 20 metal layers.
  - Most routing happens on the surface metal layers, due to via cost.
  - Net crossing is harder to deal with.

- Better and More Scalable Solutions
  - Existing solutions have long run time, limited **routability**, and limited **scalability**.
  - Quilter, a Reinforcement Learning (RL) –based automatic PCB layout tool, supports <500 components, takes a few hours to generate design candidates[1].

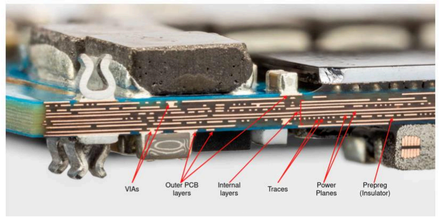
Diverse sizes and shapes



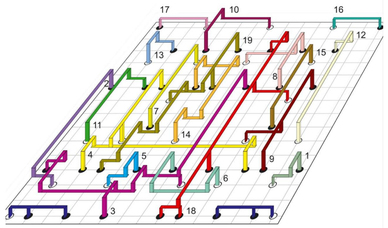
Component rotation



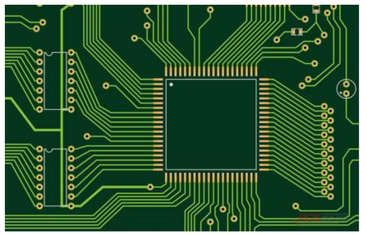
VLSI Metal Layers[2]



PCB Copper Layers[3]



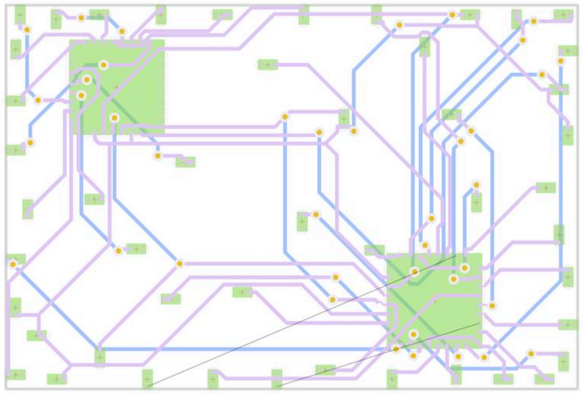
VLSI Routing Result[4]



PCB Routing Result[3]

[1] QUILTER AUTOMATED CIRCUIT BOARD LAYOUT. Link: https://www.quilter.ai/technology. Accessed: 02/25/2025.
[2] https://www.lukevassallo.com/wp-content/uploads/2023/09/automated_pcb_component_placement_using_rl_msc_thesis_v2_1_lv.pdf
[3] Yen-Chun Liu, Tung-Chieh Chen, Yao-Wen Chang, and Sy-Yen Kuo. 2019. MDP-trees: multi-domain macro placement for ultra large-scale mixed-size designs. In Proceedings of the 24th Asia and South Pacific Design Automation Conference (ASPDAC '19). Association for Computing Machinery, New York, NY, USA, 557–562. https://doi.org/10.1145/3287624.3287677
[4] Hao, R., Cai, Y., and Zhou, Q. Intelligent and kernelized placement: A survey. Integration, 86:44–50, 2022. ISSN 01679260. doi: 10.1016/j.vlsi.2022.05.002.

SOLUTIONS: PCB PLACEMENT AUTOMATION

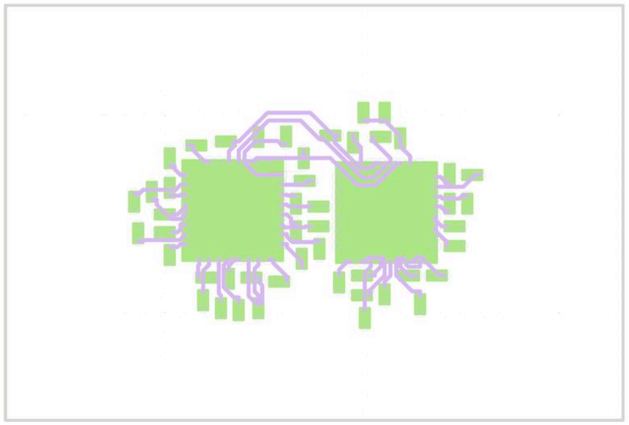


- Does not model **routing conflicts**
- Lacks support for **orientation/rotation**

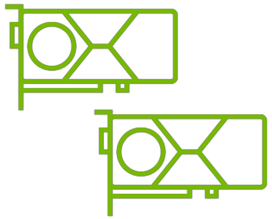
- No acceleration
- **Slow:** 1-2 hours run time
- **Small:** <500 components

Lacks realistic benchmarks

State of the Art



- **PCB-specific** cost functions
- **Orientation-aware**



- **GPU-accelerated**
- **Fast:** placement in minutes
- **Scalable:** 10,000 components

An **open-source** benchmark suite derived from **real designs**

Our Solution

OUTLINE

- Introduction
- **VLSI-Inspired PCB Placement with GPU Acceleration**
- Experimental Results
- Conclusion

VLSI-INSPIRED PCB PLACEMENT WITH GPU ACCELERATION

Framework Overview

- Problem Formulation
  - Surfaced-mounted, **noncritical** components on rectilinear PCB top/bottom.
  - Takes a PCB netlist with pre-assigned side as input.
  - Constraints: spacing, non-overlap, fixed critical, mechanical components.

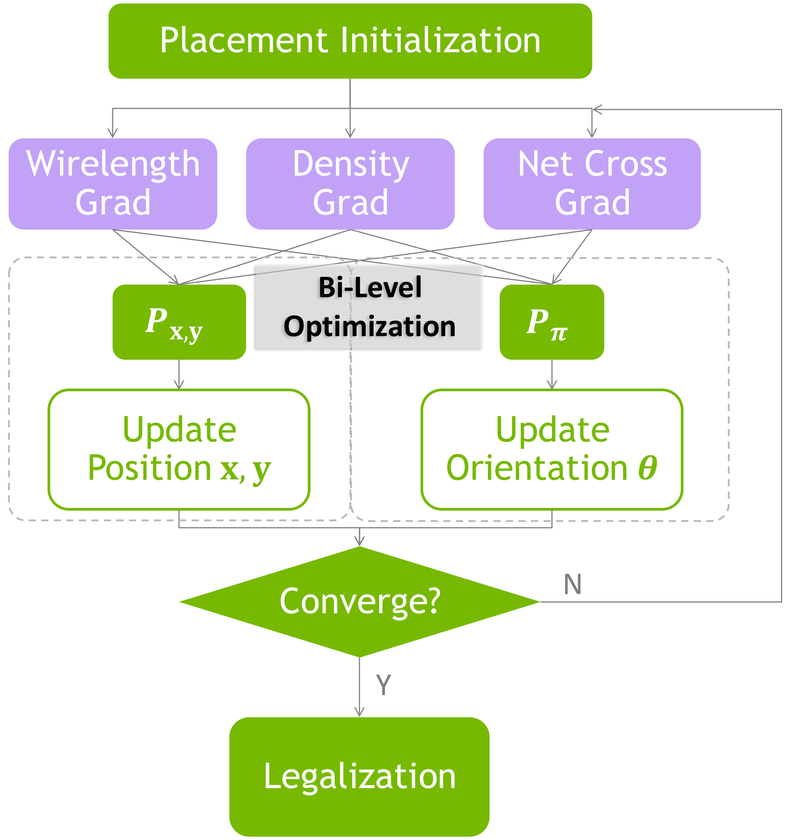
- Optimization for Routability
  - **Orientation-aware wirelength** model: extends LogSumExp.
  - **Density**: two-sided electro-static density map.
  - **Net crossing**: PCB routing conflict model.

Component center positions

Component orientations

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \mathrm{WL}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) + \lambda_D \, \mathrm{D}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) + \lambda_{NC} \, \mathrm{NC}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta})$$

Wirelength | Density | Net Crossing



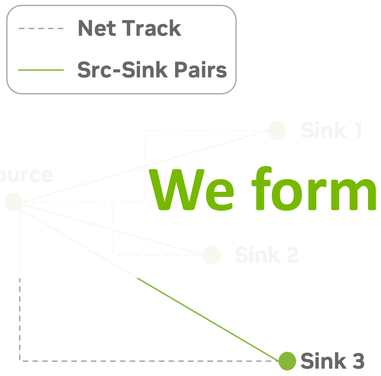
Cypress Workflow

TAILORED COST FUNCTIONS

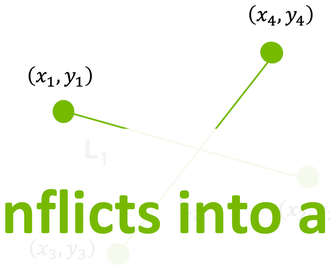
Net Crossing

- Due to limited metal layers, routing **crossing nets** is much more difficult.
- We introduce a metric, net crossing, to capture **routing conflicts** on the same copper layer.
  - Net crossing is defined as the sum of smoothed first-degree Bézier parameters of all source-sink pin pairs.

① Represent a net as source-to-sink pin pairs.



② Obtain the crossing of pin pairs.



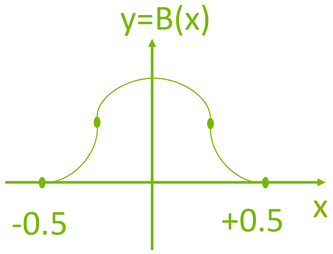
For example:

$$L_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + t \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + u \begin{bmatrix} x_4 - x_3 \\ y_4 - y_3 \end{bmatrix}$$

If $0 \le t \le 1$ and $0 \le u \le 1$ simultaneously, there is an intersection.

We formulate PCB routing conflicts into a differentiable cost function

③ Pass t, u through a bell function, then multiply.



Bell Function

$$\text{Crossing}(L_1, L_2) = B(t - 0.5)B(u - 0.5)$$

④ Define net crossing for all nets on the same layer.

$$\text{NC} = \sum_{(i,j)} \text{Crossing}(L_i, L_j)$$

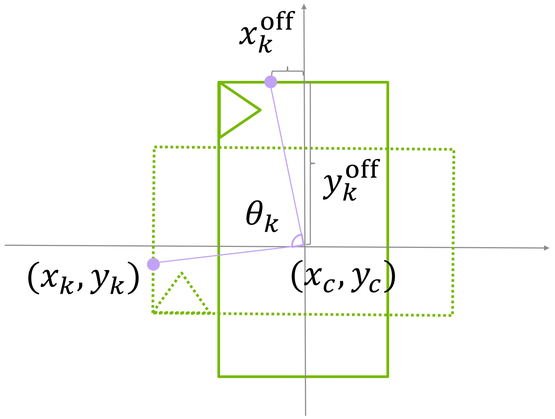
where Li, Lj are pin pairs of nets on the same metal layer.

TAILORED COST FUNCTIONS

Orientation

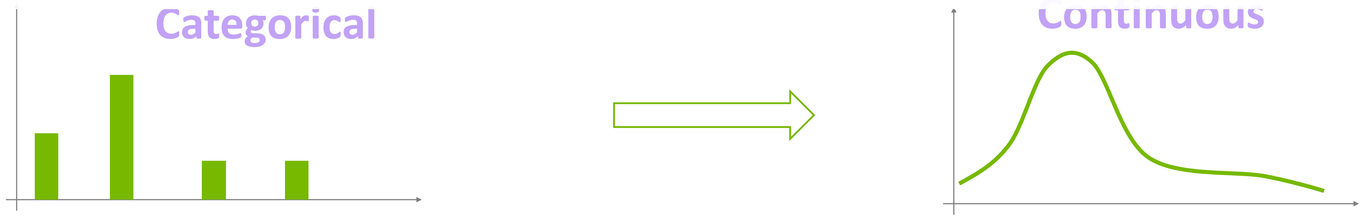
- Components have four **legal** orientations: **N, E, S, W**. We map them to 0°, 90°, 180°, 270°.
- For density map (electrostatic), we calculate new height, width, after rotation for **two sides**.
- For net crossing, and wirelength (LogSumExp), we calculate the **rotated pin locations**.
  - For pin k, the coordinate after rotation is:

$$x_k = x_c + x_k^{\text{off}} \cos\theta_c - y_k^{\text{off}} \sin\theta_c$$
$$y_k = y_c + x_k^{\text{off}} \sin\theta_c - y_k^{\text{off}} \cos\theta_c$$



- Note that orientation is a discrete, categorical variable.
- We cannot directly optimize orientation through differentiable learning.

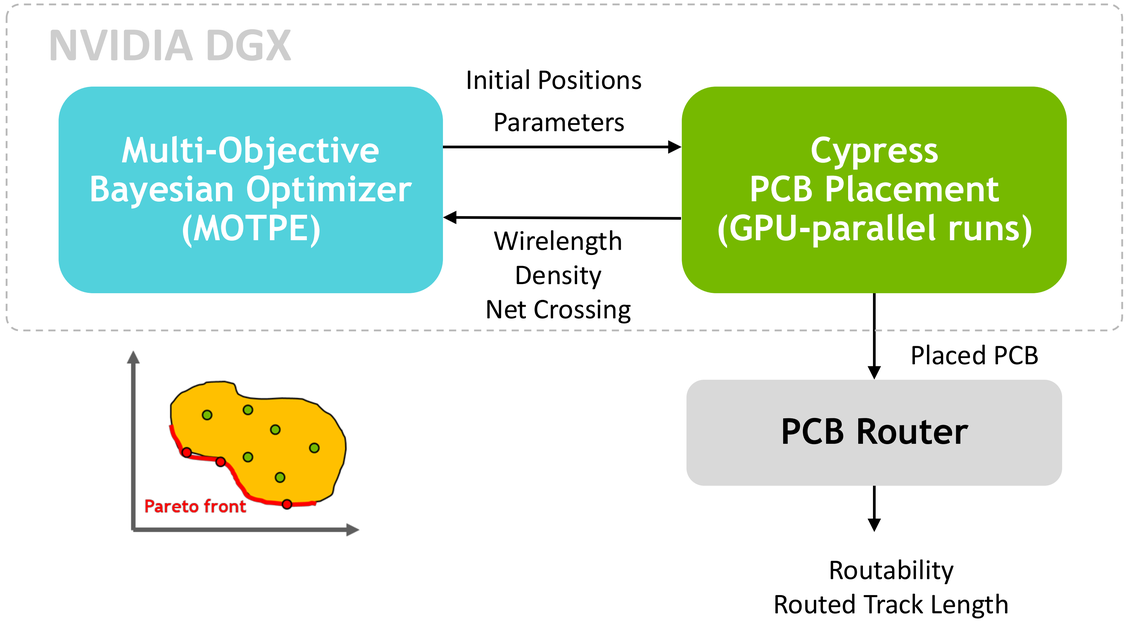
Cypress jointly optimizes position and orientation through differentiable learning



- This implies a **bi-level placement optimization** process for position and orientation.
  - Use different learning schedules, fix one while optimizing the other.

PARAMETER SPACE EXPLORATION

- Optimizer parameters affect the final solution quality.
- We use **multi-objective Bayesian optimization (MOTPE)** for parameter space exploration.
- The exploration process **runs in parallel across multiple GPUs**.



MOTPE-Based Optimization Parameter Space Exploration

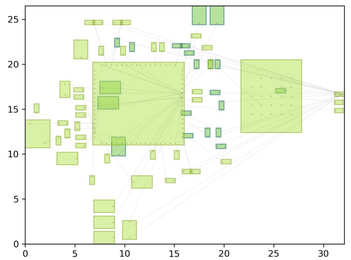
Table. Cypress Parameter Space

Parameter	Search Range
horiz. initial position	[0.2, 0.8](%)
vertical initial position	[0.2, 0.8](%)
init. density weight	[1e-6, 1.0]
target density	[0.1, 1.0]
init. net crossing weight	[1e-6, 1.0]
LogSumExp init. γ	[0.1, 0.5]
init. learning rate	[1e-4, 1e-2]
optimizer	{Adam, Nesterov}

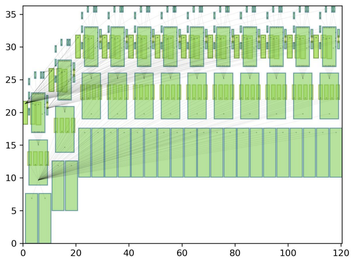
PCB PLACEMENT BENCHMARK

Open-Source Suite

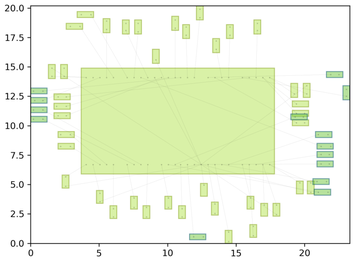
- We consider how PCB engineers approach placement: they start with **local placement** on a subset of components, then position the group on the main board. These groups are chosen based on functionality.
- Based these insights, we synthesize 10 small benchmarks based on **functionality**.
  - Include component shapes, side assignment, pin positions, net connections.
  - Based on Ethernet IC controller, power controller, GPIO expander, CPLDs, etc.



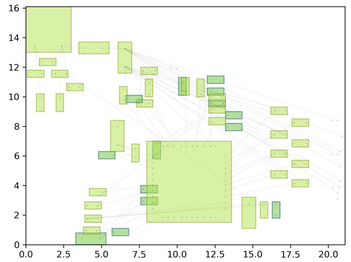
Small-1



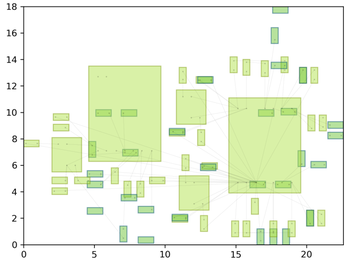
Small-2



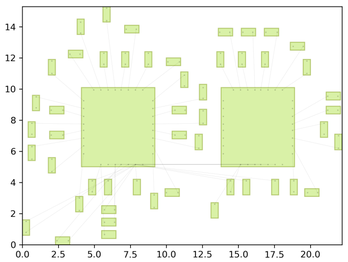
Small-4



Small-5



Small-6



Small-7

Selected Small Benchmarks, color indicates top/btm sides

Table. PCB Benchmark Suite Statistics

	#components	#movable	#pins	#nets
small-1	57	57	167	85
small-2	240	240	612	84
small-3	63	63	156	51
small-4	43	43	129	82
small-5	41	41	105	49
small-6	46	46	85	26
small-7	50	50	153	95
small-8	115	115	5334	199
small-9	476	476	2212	1513
small-10	136	136	582	403
big-1	6589	1714	18140	5620
big-2	6537	1918	21824	7190
big-3	5118	1293	11618	2914
big-4	6542	1922	21893	7166
big-5	6628	2152	20838	6937

NVIDIA.

OUTLINE

- Introduction
- VLSI-Inspired PCB Placement with GPU Acceleration
- **Experimental Results**
- Conclusion

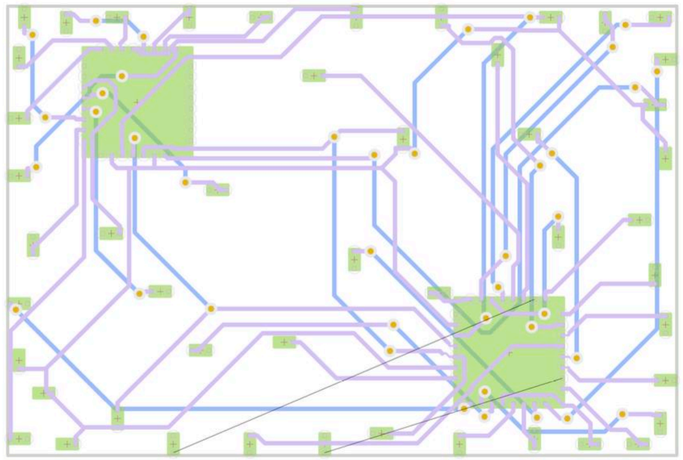
EXPERIMENTAL SETUP

- Hardware
  - NVIDIA V100 GPUs + Intel Xeon 2.2GHz, 80GB memory.
- Tools
  - Freerouting as an evaluator for routability
    - V1.9.0, two free copper layers, default settings, routability defined as the percentage of routed pin pairs.
- Benchmarks
  - 10 small benchmarks + 5 commercial big benchmarks.
- Baselines
  - **SA-PCB**
    - Community open-source solution from OpenROAD.
    - Annealing-based method optimizing for low density.
  - **NS-Place**
    - Academic open-source (ASP-DAC’22)
    - Gradient-based optimization for net convex hull separation + MILP legalization.
  - **NS-Place+**
    - Replace MILP with our legalizer: Tetris legalization, followed by Abacus row-based legalization.
  - **Quilter**
    - Commercial closed-sourced solution
    - Uses reinforcement learning (RL) -based method.

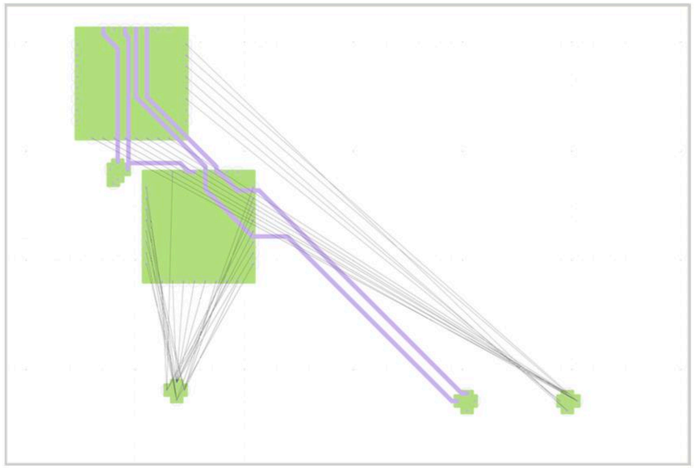
NVIDIA.

SMALL BENCHMARK RESULTS

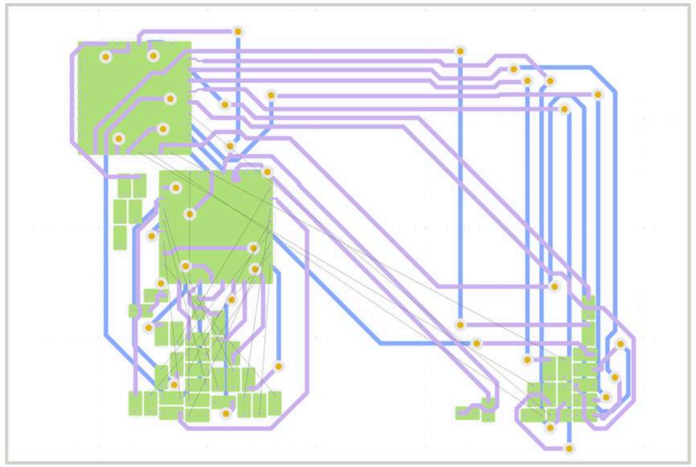
Visualization



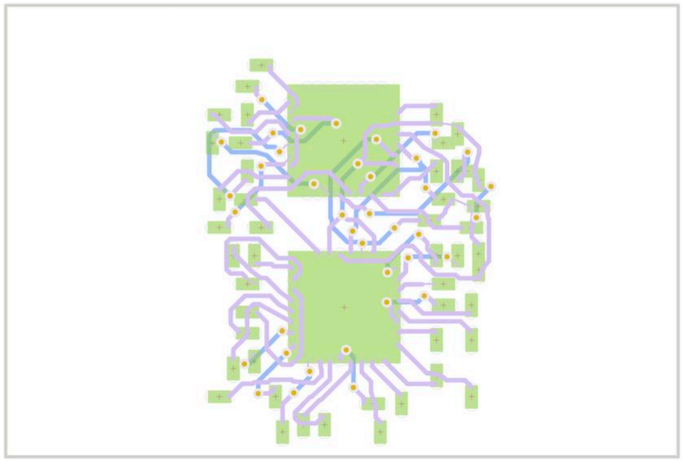
(a) SA-PCB (97% routable)



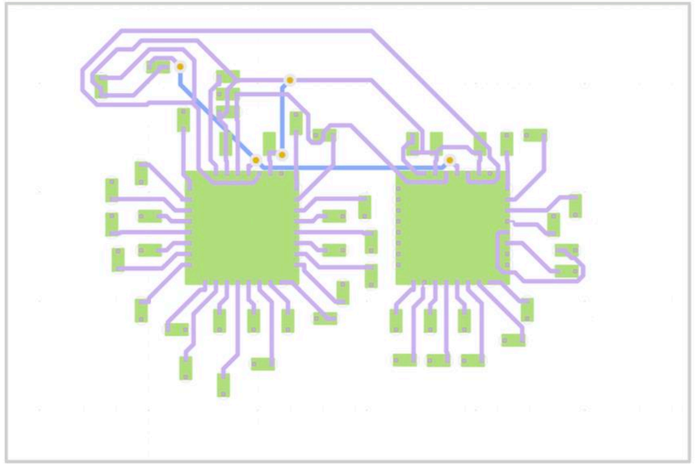
(b) NS-Place (17% routable)



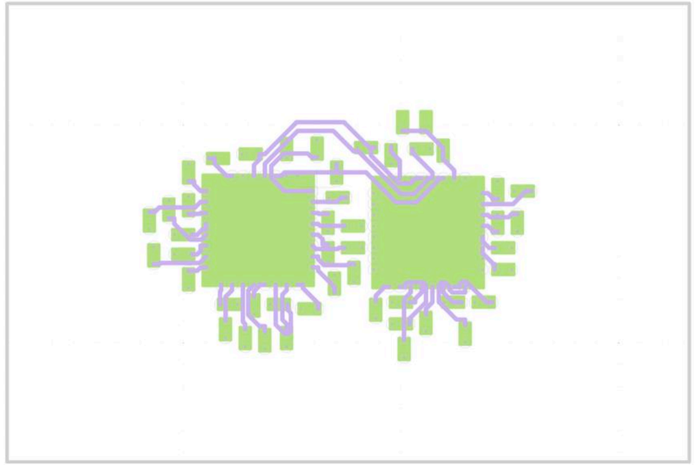
(c) NS-Place+ (62% routable)



(d) Quilter (100% routable)



(e) Human (100% routable)



(f) Cypress (100% routable)

SA-PCB: Annealing **NS-Place:** Net separation + MILP **NS-Place+:** Enhanced Legalization **Quilter:** Reinforcement Learning

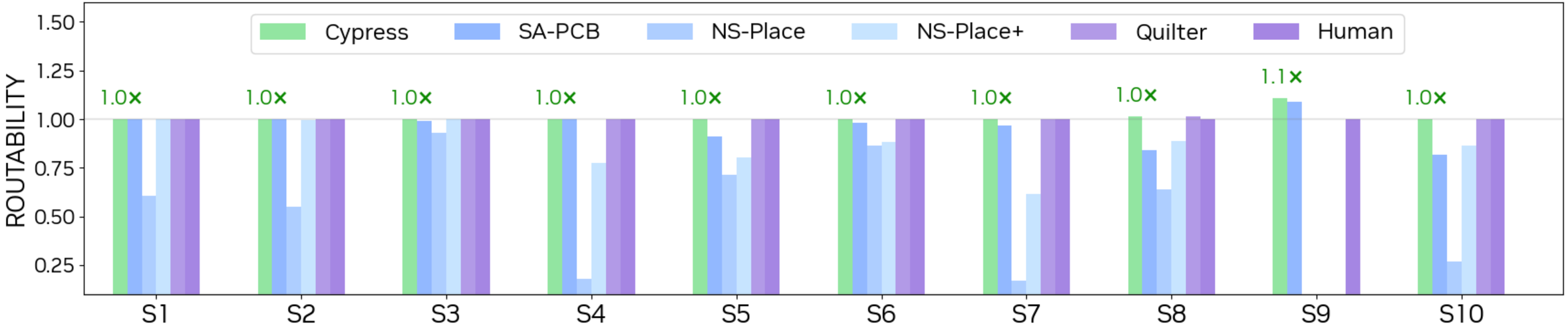
NVIDIA.

SMALL BENCHMARK RESULTS

Routability

Higher is better

Normalized against human baseline



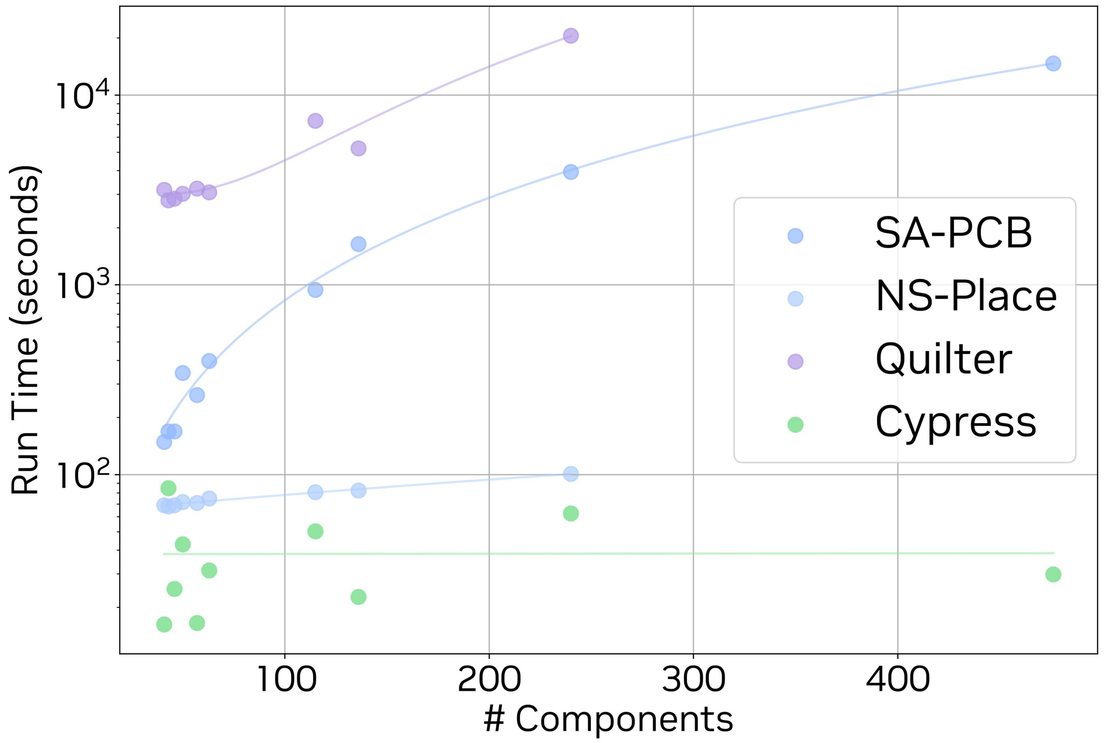
- Cypress improves routability by up to **1.2X** over SA-PCB, **5.9X** and **1.6X** over NS-Place, NS-Place+.
- SA-PCB's strategy works for small designs but hurts larger ones.
- Even with stronger legalization, NS-Place+ still fails in most cases.

SA-PCB: Annealing **NS-Place:** Net separation + MILP **NS-Place+:** Enhanced Legalization **Quilter:** Reinforcement Learning

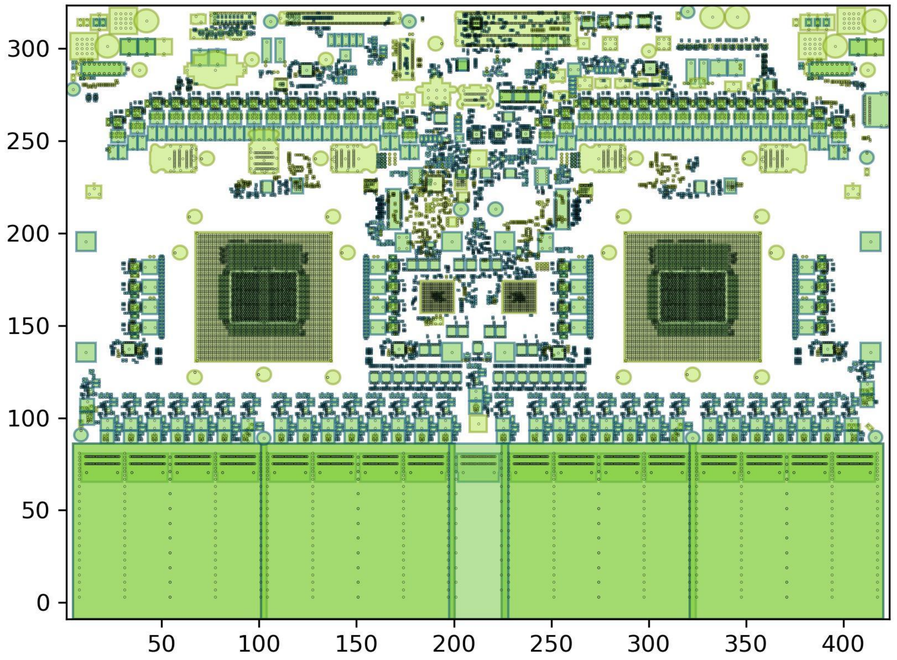
SMALL BENCHMARK RESULTS

Scalability

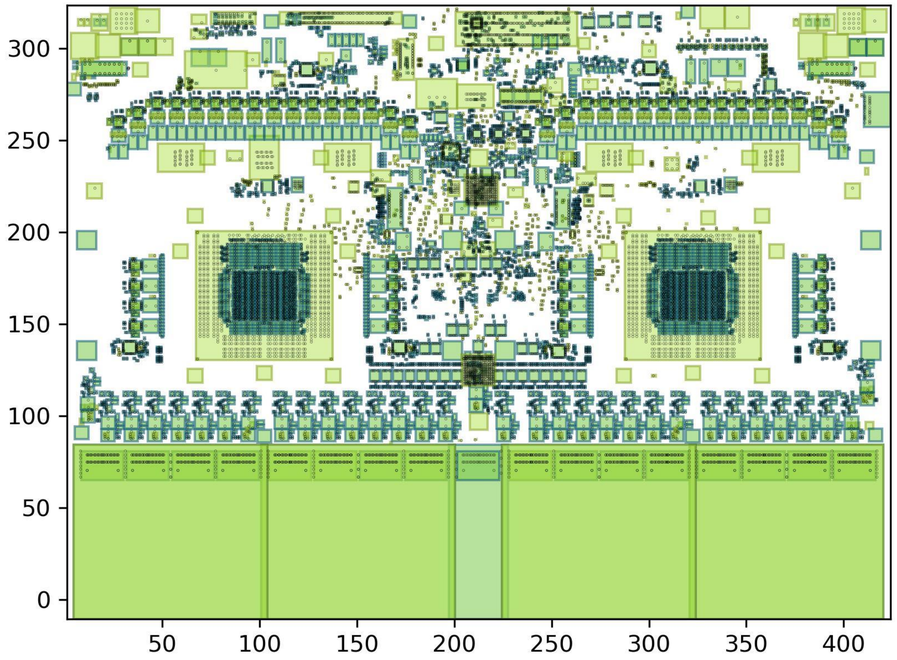
- Cypress is much more scalable compared to other methods, benefiting from its GPU acceleration.
  - Up to **492.3X** faster than SA-PCB.
  - Up to **4.3X** faster than NS-Place and NS-Place+.
  - Up to **329.7X** faster than Quilter.
- Cypress parameter space exploration runs across multiple GPUs, enabling faster exploration by scaling the number of GPUs.
- Using four V100 GPUs:
  - S1, S3, S4, S5, S6, S7, S10 completes in **200.5** to **350.7** seconds.
  - S9 requires **584** seconds, and S2 takes **1764.7** seconds.



BIG BENCHMARK RESULTS



B1, Human Placement

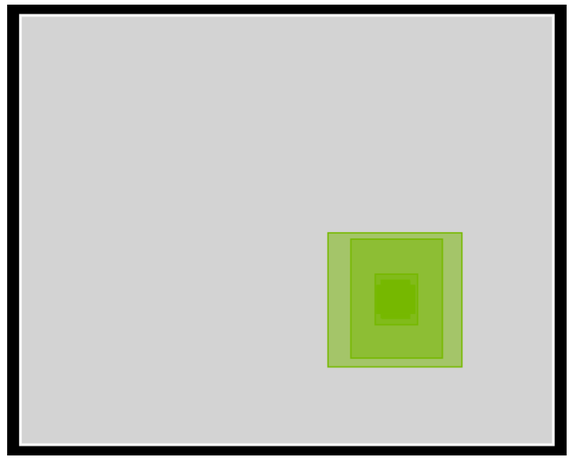


B1, Cypress Placement

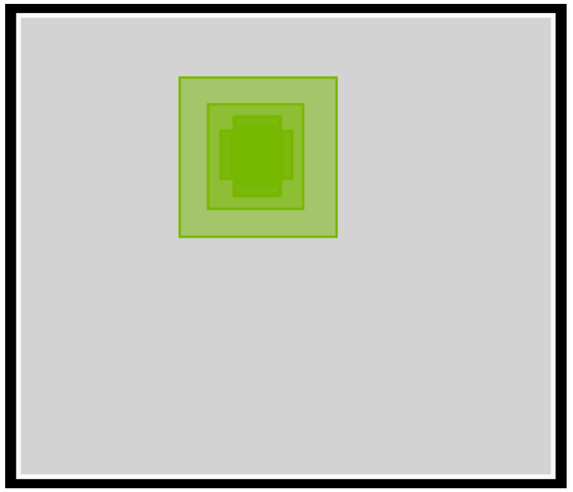
	WL + Density		+ Net Crossing		+ Orientation	
	HPWL	NC	HPWL	NC	HPWL	NC
B1	2.87M	100.7K	3.04M	31.4K	2.87M	15.3K
B2	6.53M	84.1K	6.53M	15.6K	6.53M	15.3K
B3	1.33M	91.5K	1.36M	33.5K	1.33M	37.4K
B4	6.58M	78.9K	7.20M	9.7K	6.57M	16.8K
B5	5.24M	89.9K	7.25M	46.4K	6.28M	34.0K
Avg	1.0	1.0	1.13	0.31	1.05	0.27

- Ablation study & demonstrate scalability
- Add net crossing alone worsens wirelength
- Further enabling orientation achieves better overall layouts.

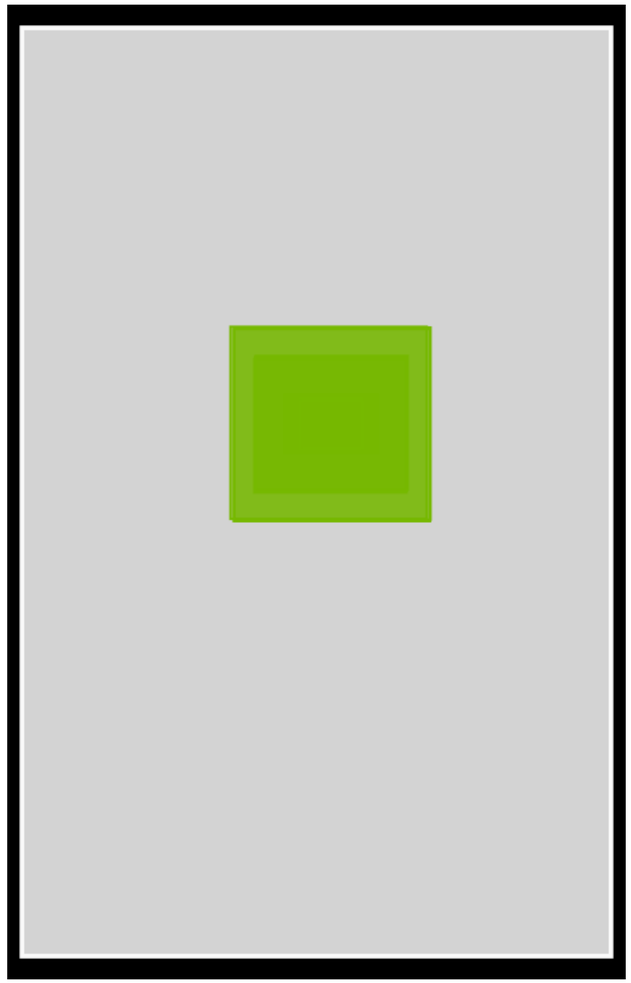
SMALL BENCHMARKS VISUALIZATION



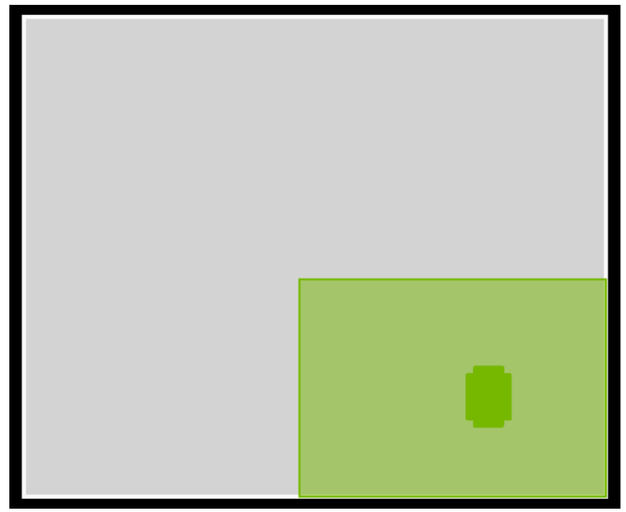
Small 1



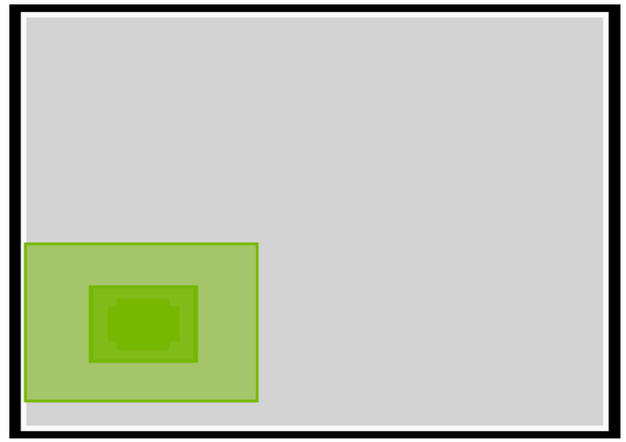
Small 5



Small 3



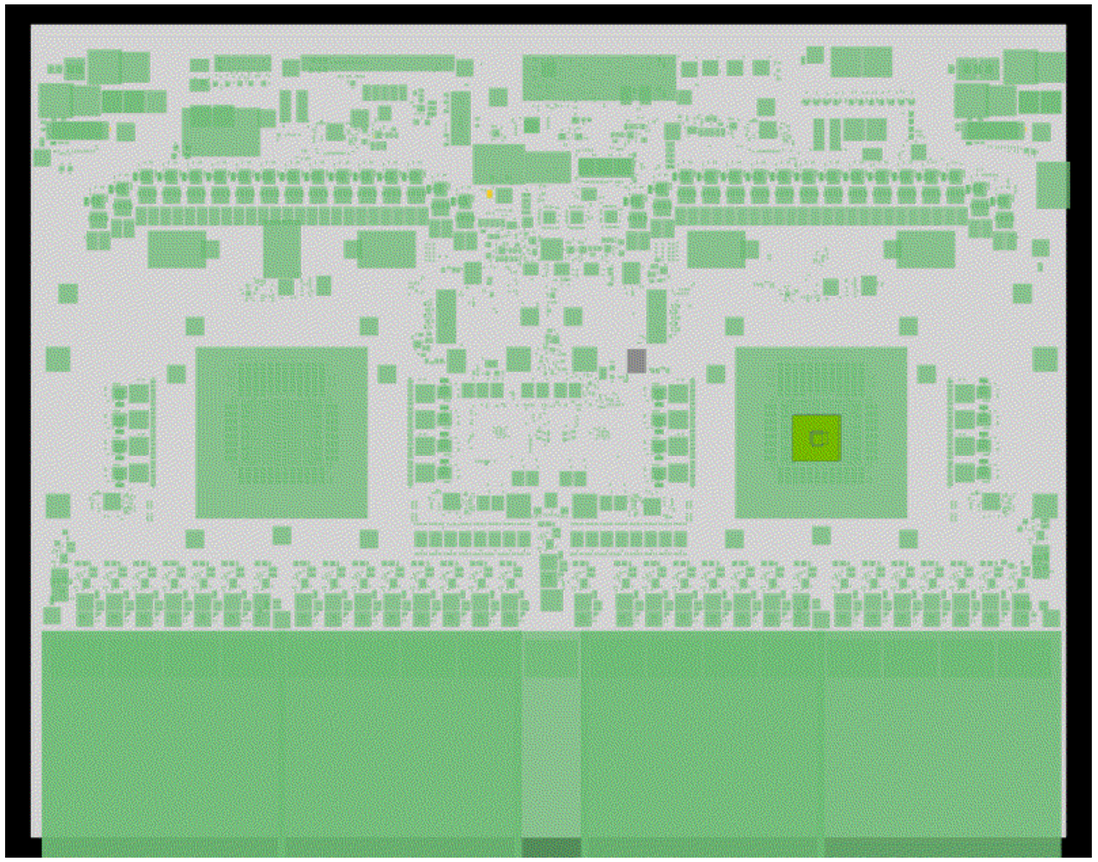
Small 4



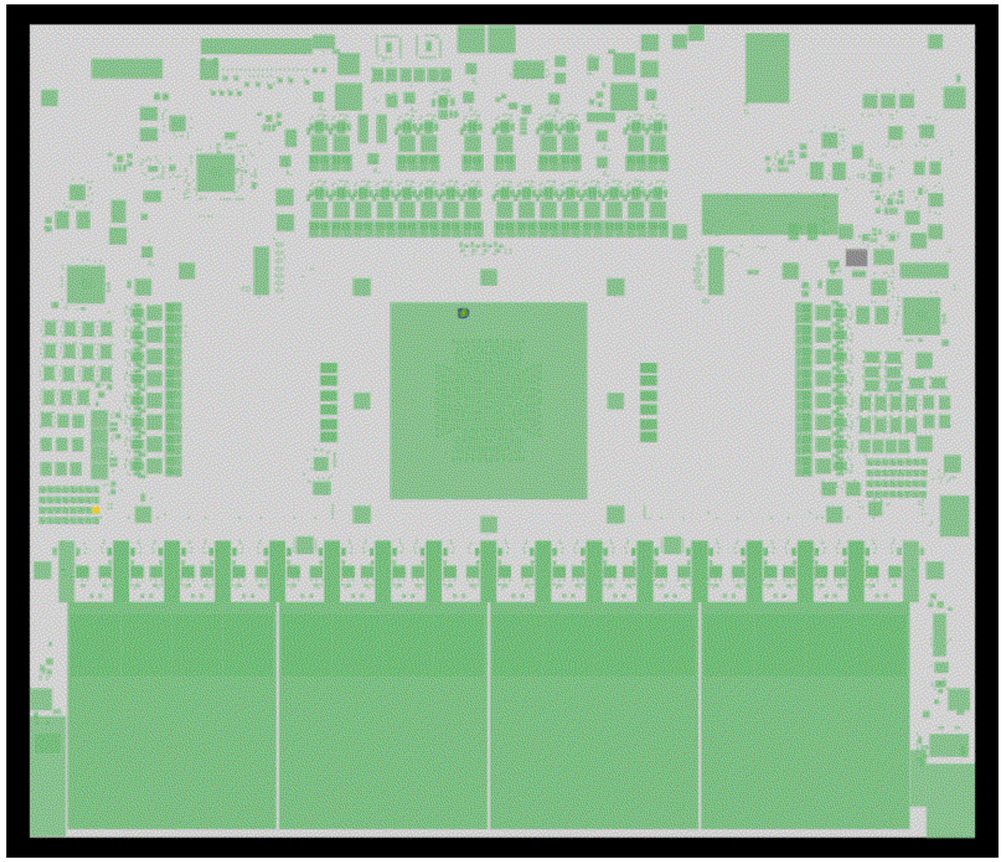
Small 6

NVIDIA.

BIG BENCHMARKS VISUALIZATION

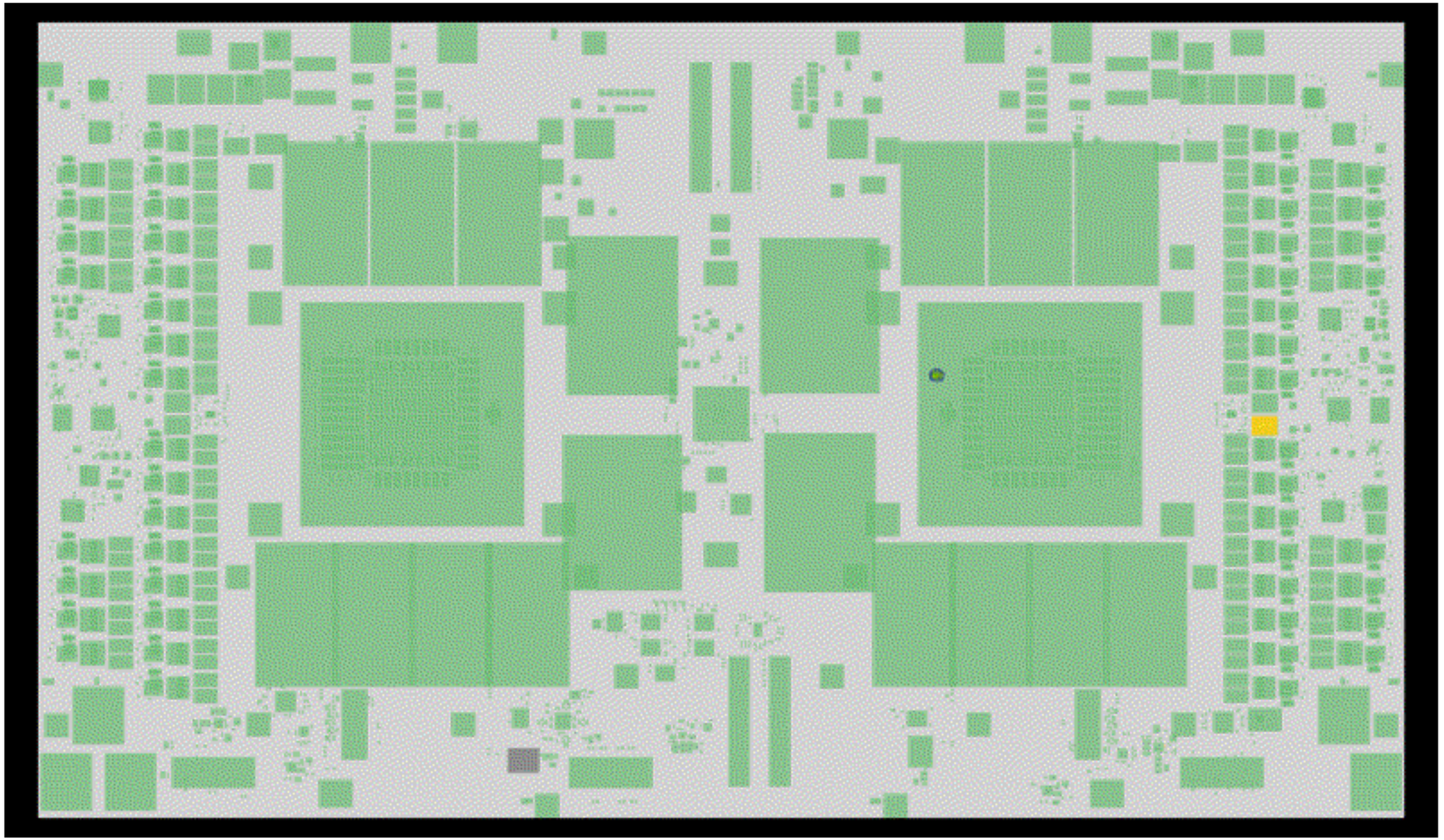


Big 1



Big 3

BIG BENCHMARKS VISUALIZATION



Big 4

NVIDIA.

OUTLINE

- Introduction
- VLSI-Inspired PCB Placement with GPU Acceleration
- Experimental Results
- **Conclusion**

CONCLUSIONS

- **Scalable GPU-accelerated PCB placement**
  - Inspired by VLSI techniques
  - Handles flexible design spaces & routing constraints
- **Outperforms state-of-the-art tools**
  - Improved routability, shorter track length, and faster run time
  - Scales well for large commercial designs
- **Synthesized benchmark suite**
  - Enables fair tool comparisons and tracks progress
- **Open-sourced code & benchmarks**
  - Supports further research in PCB placement

Cypress & Benchmarks are open-source on GitHub

https://github.com/NVlabs/Cypress

NVIDIA.