DATE 26
DESIGN, AUTOMATION AND TEST IN EUROPE
THE EUROPEAN EVENT FOR ELECTRONIC SYSTEM DESIGN & TEST
20 – 22 APRIL 2026
VERONA, ITALY
PALAZZO DELLA GRAN GUARDIA

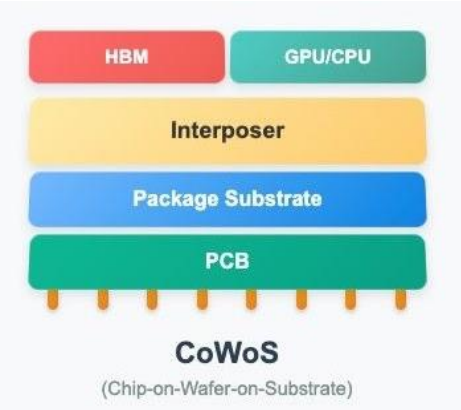
PCB-Migrator: Automated PCB PnR Migration

Yaohui Han
Department of Computer Science and Engineering
The Chinese University of Hong Kong

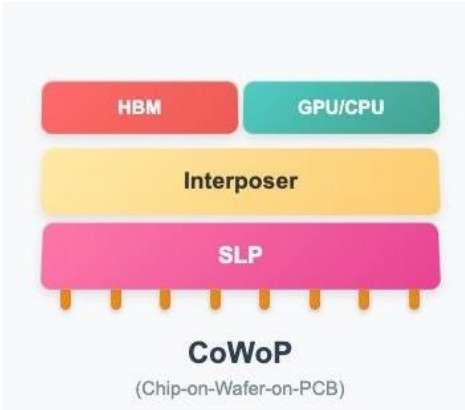
香港中文大學
The Chinese University of Hong Kong

Background: Growing Demands for PCB

- Chip-on-Wafer-on-Platform PCB (CoWoP) is a novel technology to improve the performance of electronic systems, proposed by Nvidia.



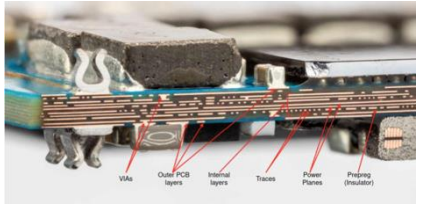
New integration method



CoWoP directly connect the interposer and PCB, omitting the substrate. This will significantly improve the overall performance, but it also places higher demands on PCB designs.

Background: Introduction of PCB Design

- PCB design can be divided into Placement and Routing.
  - Placement is to physically position components on the board.
  - Routing is the process of creating traces to connect all nets.

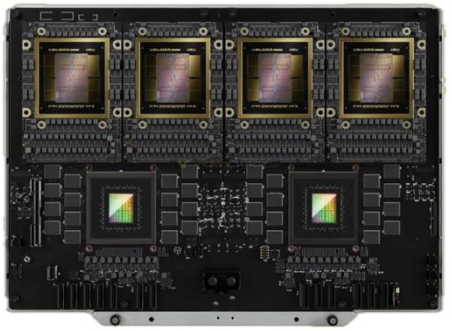


Multiple layers of PCB

Meanwhile, modern PCB designs are complex:

1. There may be **multiple layers** in modern PCB designs.
2. The **large solution space** of PCB PnR, both the numbers of components and nets are growing.
3. Several **performance** consideration.

The performance of existing automated PCB design frameworks is still unsatisfactory for actual use.



The scale of commercial PCB: >10,000 components

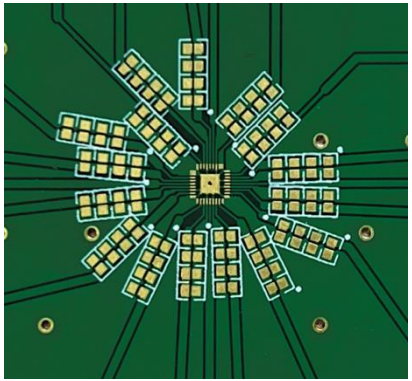
Background: Difference with VLSI Design

VLSI Physical Design:

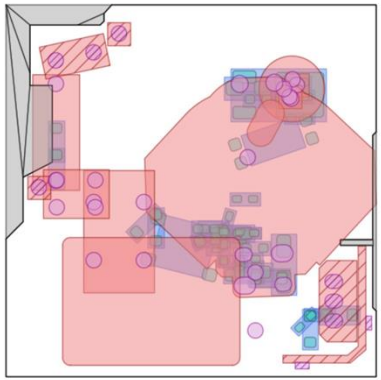
1. Much more cells and nets to do PnR.
2. The cells and traces are placed based on tracks.
3. Floorplan is necessary for the number of cells.

PCB Physical Design:

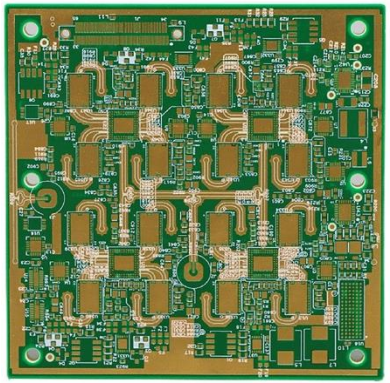
1. Less components, but they can be placed more freely.
2. Less nets, but the traces can be routed from more angles.
3. Copper pouring can be also used to connect nets.



Any-oriented components



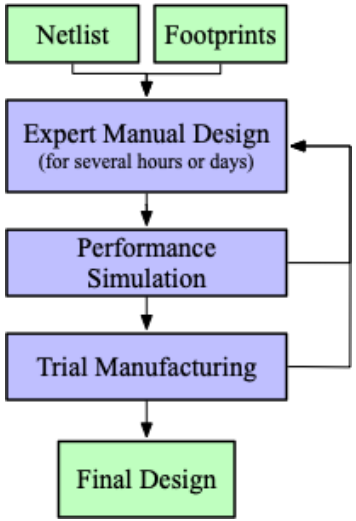
Irregular-shaped components



High density, with copper pouring

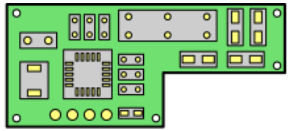
Background: The Gap between Engineers and Algorithms

- All existing PCB layout design frameworks cannot reach the level of expert manual design.

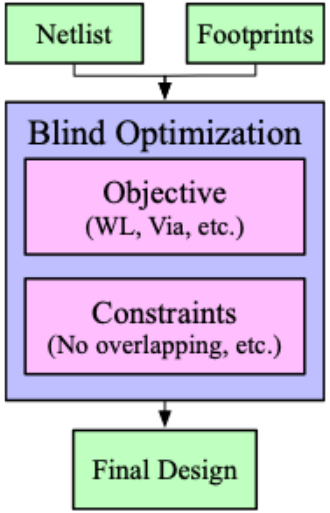


Expert Manual Design:

1. Great performance.
2. Foresight for routing at the placement stage.
3. **Time-consuming.**



Expert-level, great performance



Automated Frameworks:

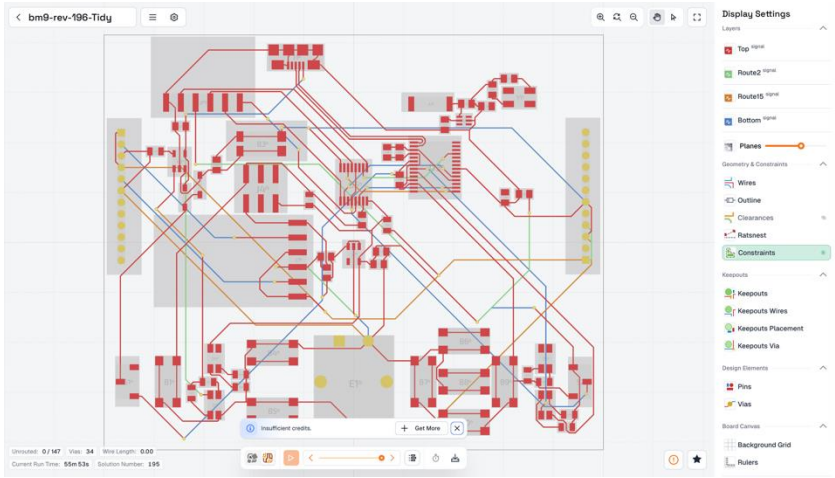
1. Usually short runtime.
2. Pretty good WL/Via, compared to the expert manual designs.
3. **Unable to consider actual needs.**
4. **Poor performance and usability.**

Impractical, or even unroutable

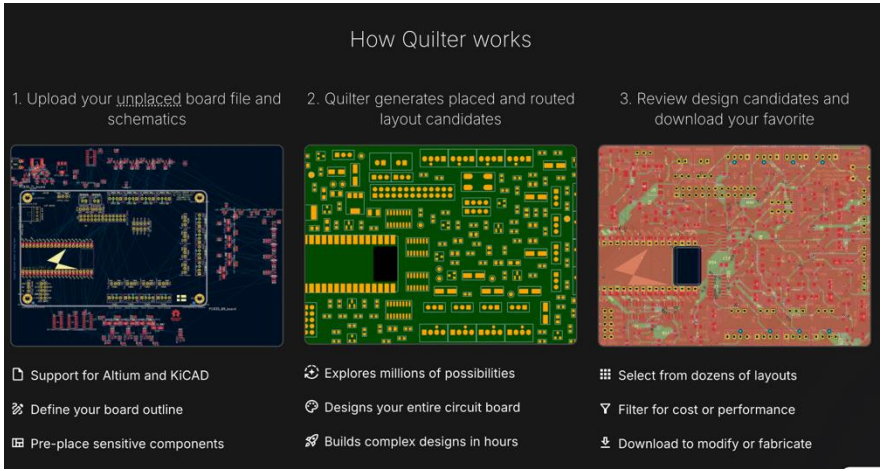
Thus, there is a big **gap** between human engineers and automated algorithms!

Background: Existing Attempts

- There are some commercial tools, such as DeepPCB and Quilter, which try to improve PCB design quality with Reinforcement Learning (RL).



The GUI of DeepPCB platform.



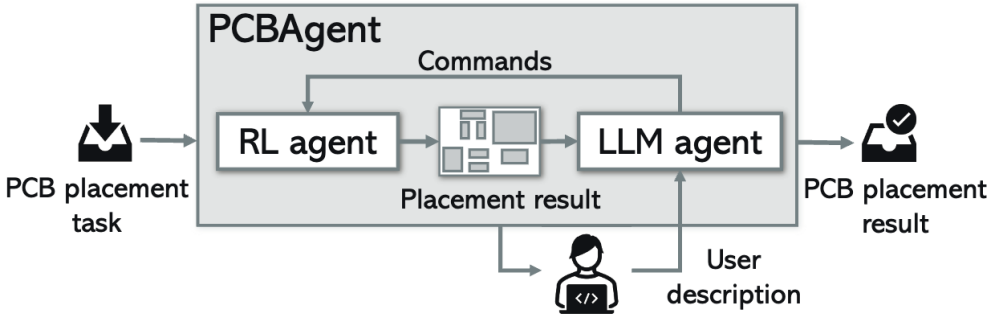
The GUI of Quilter platform.

Judging from results, existing RL-based commercial tools still cannot do high quality PCB PnR.

Challenges: Design Expert-level PCB

- There are mainly two paths to design expert-level PCB to meet the actual requirements from industrial:
  - **LLM4PCB:** leveraging the generative ability of LLM.
  - **Migration:** referencing existing expert designs.

Difficult to convergence, effective only on small designs:



Example flow of LLM for PCB placement.

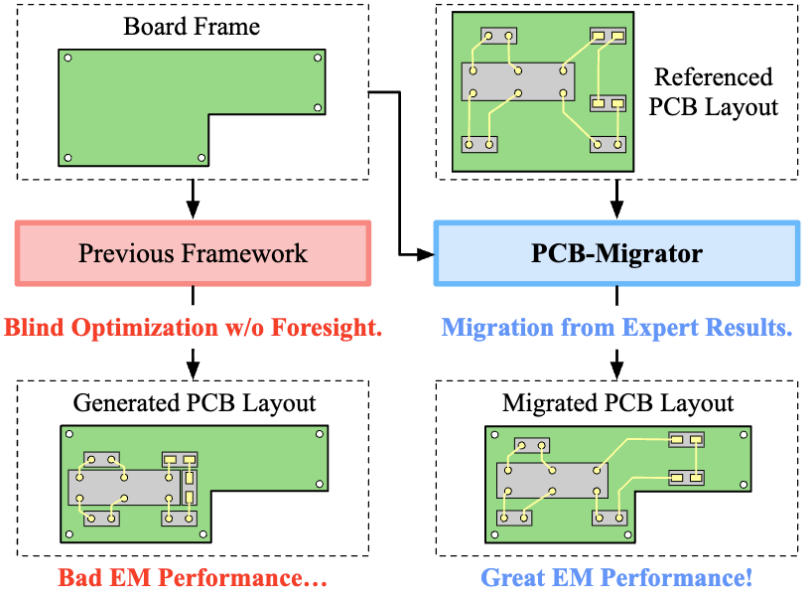
[1] Chen et al. "PCBAgent: An Agent-based Framework for High-Density Printed Circuit Board Placement", ASP-DAC, 2025

Solution: PCB PnR Migration

- **Motivations:**

1. Existing automated "design-from-scratch" PCB design frameworks:
   - Lack the foresight for routing in the placement stage.
   - Difficult to reach the industrial-grade performance, especially EM.

2. There are already many expert PCB designs that can be used to guide new designs.

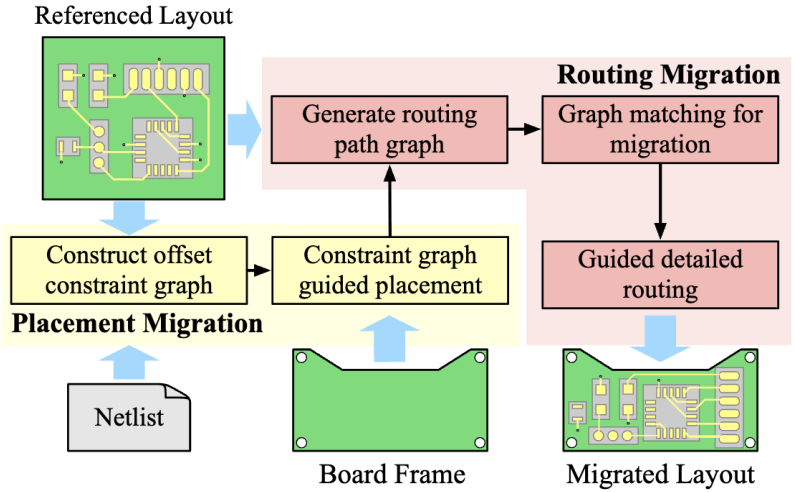


Comparison between previous framework and PCB-Migrator.

[1] PCB-PR-App, https://github.com/The-OpenROAD-Project/PCB-PR-App.
[2] DeepPCB, https://app.deeppcb.ai/.
[3] Freerouting, https://www.freerouting.app/.
[4] Cheng et al. "Net separation-oriented printed circuit board placement via margin maximization", ASP-DAC, 2022

Overall Flow of PCB-Migrator

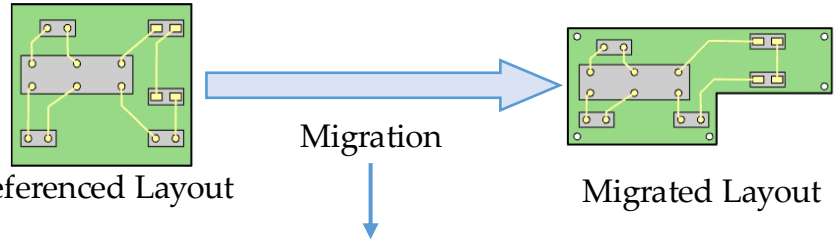
- There are two main stages in PCB-Migrator:
  - Placement Migration
  - Routing Migration
- **Problem Formulation**:
  - **Input**: referenced layout, netlist, new board frame.
  - **Output**: the migrated layout.
  - **Goal**:
    1. Maintain the key characteristics of referenced PCB.
    2. Keep the expert-level performance (WL, via, electromagnetic, etc.)



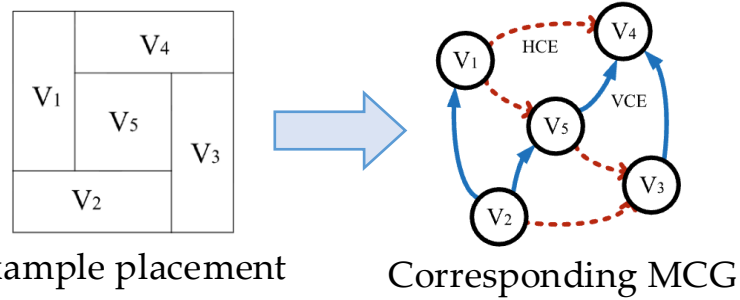
The overall flow of PCB-Migrator.

Mixed Constraint Graph

- In previous EDA research, the relative positions in placement are represented by Mixed Constraint Graph (MCG).



It is necessary to extract the relative positions between components



Example placement

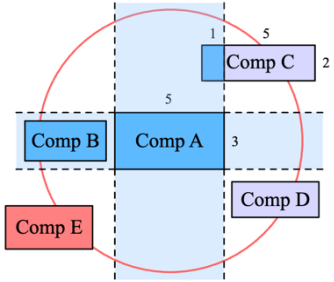
Corresponding MCG

- In MCG, constraints are limited to between two adjacent components. And there are only two types (vertical and horizontal) of constraints.
- Thus, the conventional MCG cannot precisely represent the relative positions between components…

[1] Young et al, "Placement Constraints in Floorplan Design", TVLSI, 2004
[2] Gao et al, "Interactive analog layout editing with instant placement and routing legalization", TCAD, 2022

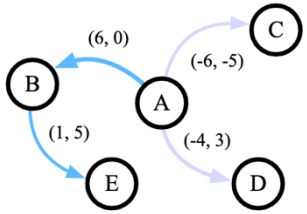
Offset Constraint Graph

- To represent the relative position and rotation between components, we propose the Offset Constraint Graph (OCG):



- If the component is within the distance threshold of another component, then add an edge between them.
- The edge weight between component i and j is calculated by the following formula:

$$\mathcal{W}_{ij} = e^{-d_{ij}} + \mu \cdot \frac{S_{ij}}{\min(S_i, S_j)}$$



S_{ij}: the common area between the overlapping area of component i's crossing region and component j.

Corresponding OCG ⇨ The relative position is stored in the edge to support the placement.

OCG-Guided Placement

$$\min \sum_{i=1}^{|C|}\sum_{j=i}^{|C|} \mathcal{W}_{ij} \cdot (|\Delta x_{ij}^{o} - \Delta x_{ij}^{n}| + |\Delta y_{ij}^{o} - \Delta y_{ij}^{n}|) + \gamma_1 \cdot \sum_{i=1}^{|C|} \Delta\theta_i + \gamma_2 \cdot \sum_{m=1}^{|net|} \mathrm{HPWL}(x, y, \theta),$$

This is an MILP formulation, which aims to minimize the positional variance between components and rotations in new placement, and the component pairs with higher edge weight in OCG will be more influential.

No overlap allowed

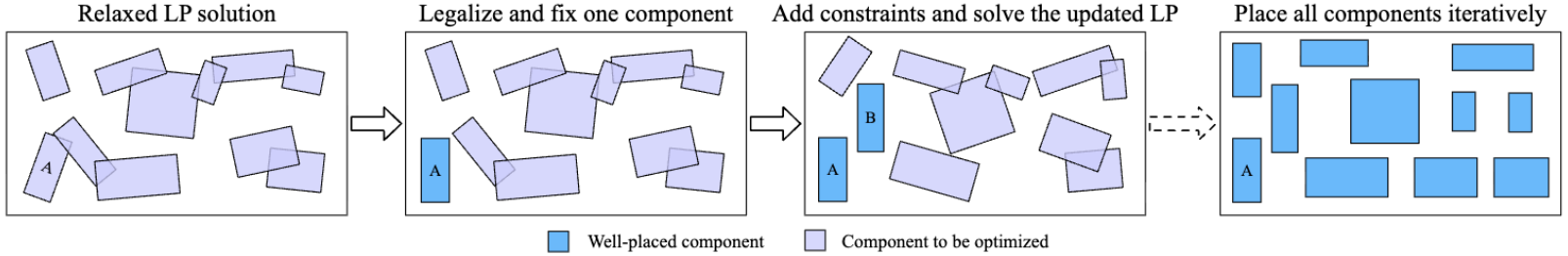
$$x_i^{proj} = |l_i \cdot cos(\theta_i)| + |w_i \cdot sin(\theta_i)|,$$
$$y_i^{proj} = |l_i \cdot sin(\theta_i)| + |w_i \cdot cos(\theta_i)|,$$
$$|\Delta x_{ij}| \geqslant \frac{x_i^{proj} + x_j^{proj}}{2} + s,$$
$$|\Delta y_{ij}| \geqslant \frac{y_i^{proj} + y_j^{proj}}{2} + s,$$

Spacing required

$$d + \frac{x_i^{proj}}{2} \leqslant x_i \leqslant X - d - \frac{x_i^{proj}}{2},$$
$$d + \frac{y_i^{proj}}{2} \leqslant y_i \leqslant Y - d - \frac{y_i^{proj}}{2},$$

Placement Speed-up

- Placement migration may be extremely inefficient in many situations if directly using the MILP solver.
- We proposed a speed-up method for placement migration:
  - Relax the rotation constraint and solve the LP problem.
  - Adjust the left-bottom component for legalization and fix it.
  - Perform another LP and repeat the above processes.
  - Until all components are well-placed.

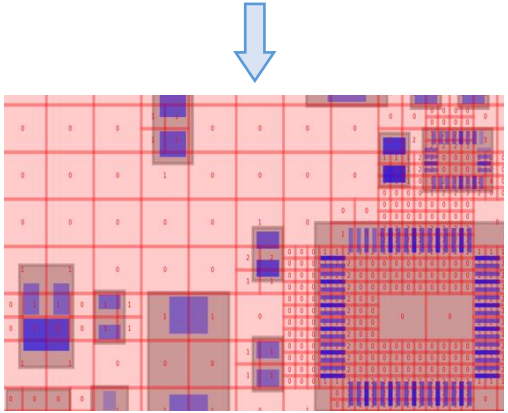


Convert Layout to Graph

- With routing path graph, routes can be represented in a rotational and symmetry invariance.

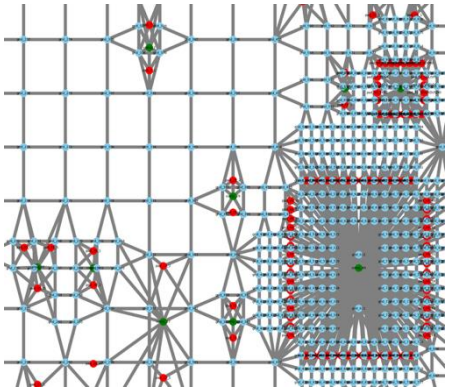
To more precisely preserve the characteristics of routing, the grid density will be higher in areas where pads are present.

In the routing path graph, component and pad nodes act as references for grid nodes to guide the subsequent graph matching.



PCB divided by grids

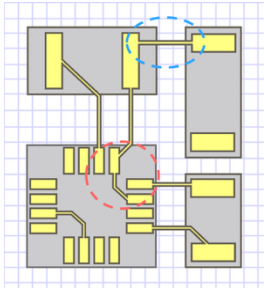
Extraction



Corresponding routing path graph

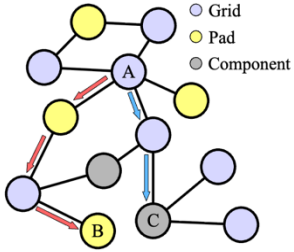
Routing Path Graph

- After placement, we perform routing migration to preserve the characteristics of reference in the new layout.



Pad-dense area with fine-grained routing characteristics (shown in red).

Regular routing area between components, with coarse-grained routing characteristics (shown in blue).



Routing Path Graph

- We calculated the neighborhood connection between grid node and pad or component nodes $f_{\{c,p\}}$ as follows:

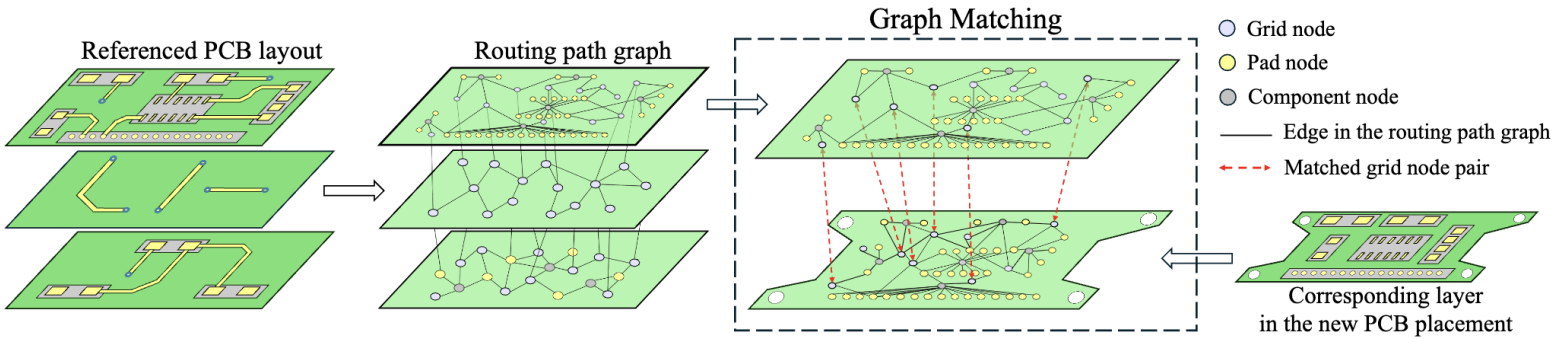
$$f_{\{c,p\}} = (t_{\{c,p\}})^{x_{\{c,p\}}},\ t_{\{c,p\}} < 1,\ x_{\{c,p\}} < x^{max}_{\{c,p\}},$$

- $t_{\{c,p\}}$ is the decay coefficient and $x_{\{c,p\}}$ is the max BFS steps for reaching the components or pads.

Routing Migration

- Since each grid has been represented by a vector, we can perform matching for grid nodes in routing path graph.
- **Problem Formulation**: The matching between placements can be formulated as a bipartite graph matching:

$$\max \sum_{u=1}^{|N_o|} \sum_{v=1}^{|N_n|} \cos(u,v) \cdot x_{uv}, \quad \text{s.t.} \sum_{v=1}^{|N_n|} x_{uv} = 1, \ \forall u \in N_o, \ x_{uv} \in \{0,1\}, \ \forall u, v,$$



Guided Detailed Routing

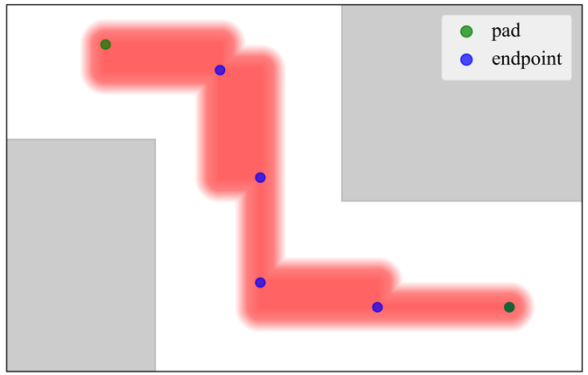
- After matching, the matched nodes are mapped on the new placement then guide the detailed routing through modified cost:

$$\Upsilon^l_{\min} = (e^{-\frac{|C_l|}{\psi}} + \eta) \cdot \Upsilon_b,$$

$$\Upsilon^l(d) = \frac{\Upsilon^l_{\min} + \Upsilon_b}{2} - \frac{\Upsilon^l_{\min} - \Upsilon_b}{2} \cos(\frac{\pi d}{d_{max}}),$$

The definition of the minimum cost value and the cost curve, guaranteeing:

1. The smooth transition of cost curve.
2. Layers with more components tend to have higher confidence levels and allow for greater cost adjustments.



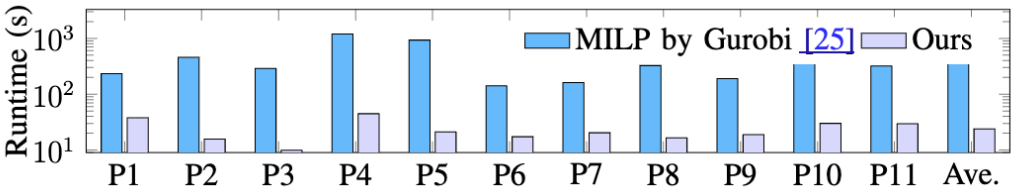
Example of modified cost map for detailed routing, the cost of red area is lower than regular area.

Experiment Results

TABLE I Benchmark statistics.

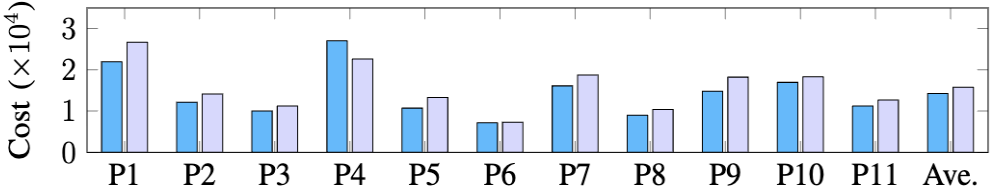
Case	Referenced Layout			Boundary		Obstacles	
	WL / Via	↓ IL	↓ XT	L (%)	W (%)	Num	Area (%)
P1	1300.7 / 44	3.63	-45.19	+36.42	-17.11	2	8.92
P2	1043.2 / 37	1.51	-42.73	+5.00	-12.20	1	5.62
P3	681.6 / 33	1.62	-35.44	-10.91	+21.43	1	15.97
P4	3721.3 / 166	3.26	-44.38	0.00	0.00	3	1.76
P5	1743.4 / 37	4.27	-43.32	+24.14	-26.89	2	4.47
P6	309.1 / 8	5.07	-38.17	-38.00	-18.18	4	14.91
P7	1125.6 / 72	5.43	-41.89	0.00	-33.21	2	5.71
P8	579.5 / 19	4.10	-44.81	-38.00	0.00	1	9.34
P9	257.3 / 16	1.51	-42.03	-16.84	0.00	2	3.36
P10	1523.7 / 99	1.67	-50.81	0.00	-12.73	4	2.17
P11	1245.8 / 34	2.95	-43.22	-7.17	-10.27	4	5.25

We collect benchmarks from OpenROAD and actual industrial applications.



Comparison between MILP by Gurobi and PCB-Migrator on Runtime.

PCB-Migrator can perform placement migration achieve highly optimal results with much shorter runtime.



Comparison between MILP by Gurobi and PCB-Migrator on Cost.

[1] OpenROAD, https://github.com/The-OpenROAD-Project
[2] Gurobi, https://www.gurobi.com/

Experiment Results

PCB-Migrator can migrate from expert-designed layout, saving WL and via, and showing great EM performance in a significantly short runtime.

TABLE II Performance comparison of PCB layout design.

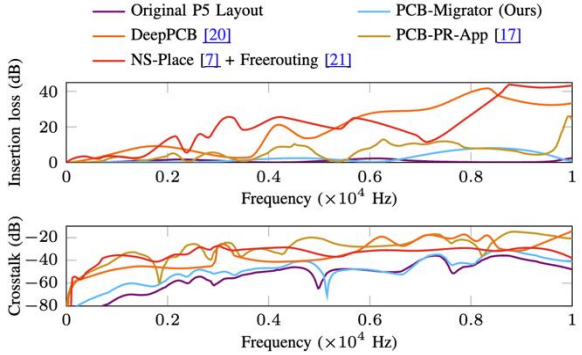
Case	PCB-PR-App				NS-Place + Freerouting				DeepPCB				**PCB-Migrator (Ours)**			
	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)	WL / Via	EM Perf.		RT (s)
		↓ IL	↓ XT			↓ IL	↓ XT			↓ IL	↓ XT			↓ IL	↓ XT	
P1	1765.6 / 153	7.38	-33.20	340	1510.9 / 63	7.29	-33.42	10992	1197.3 / 49	9.34	-37.16	12761	**1136.0 / 37**	**4.27**	**-37.35**	**180**
P2	1205.7 / 73	2.73	-38.51	182	1035.1 / 65	3.68	-37.11	6331	990.6 / 43	5.61	-40.63	5772	**895.7 / 28**	**1.80**	**-41.04**	**68**
P3	760.0 / 53	2.31	-34.81	164	**552.1** / 26	5.45	-34.48	5109	729.6 / 37	6.01	-34.83	4433	553.8 / **18**	**1.72**	**-35.23**	**55**
P4	4400.5 / 205	5.89	-39.29	657	3358.9 / 140	6.57	-29.49	11466	3705.2 / **101**	6.63	-36.65	7128	**3231.6** / 114	**3.69**	**-40.33**	**941**
P5	1701.7 / 49	8.31	-38.66	558	1881.2 / 74	8.71	-29.30	7825	1524.3 / 50	9.56	-38.71	6312	**1515.1 / 44**	**5.85**	**-40.56**	**280**
P6	650.7 / 38	11.71	-35.20	105	1018.8 / 42	7.18	-35.89	2869	688.0 / **29**	7.08	-36.41	4238	**541.0** / 37	**6.62**	**-37.22**	**74**
P7	1484.9 / 85	8.22	-39.91	500	1525.6 / 67	11.11	-40.24	10751	1389.7 / 37	8.82	-40.48	8985	**990.0 / 20**	**6.42**	**-41.16**	**90**
P8	694.3 / **11**	8.15	-36.26	140	1120.9 / 85	8.27	-36.18	11568	918.0 / 25	9.89	-37.18	9620	**584.6** / 18	**5.33**	**-38.40**	**57**
P9	737.3 / **19**	2.22	-37.63	147	914.4 / 37	2.32	-30.52	3248	988.8 / 26	2.10	-38.78	6957	**333.2** / 21	**1.71**	**-40.60**	**61**
P10	2171.6 / 137	2.81	-42.87	836	2096.9 / 109	3.41	-38.73	19856	1950.2 / **51**	7.08	-44.10	9282	**1880.1** / 87	**1.97**	**-45.60**	**152**
P11	**1244.3** / 28	8.63	-34.59	166	1329.6 / 65	8.20	-34.42	5495	1364.4 / 47	9.71	-34.09	5674	1382.4 / **21**	**3.77**	**-35.53**	**129**
Avg.	1528.78 / 77.36	6.21	-37.36	344.92	1485.85 / 70.27	6.56	-34.53	8682.73	1404.19 / 45.00	7.44	-38.09	7378.36	**1185.77 / 40.45**	**3.92**	**-39.37**	**189.69**

[1] PCB-PR-App, https://github.com/The-OpenROAD-Project/PCB-PR-App.
[2] DeepPCB, https://app.deeppcb.ai/.
[3] Freerouting, https://www.freerouting.app/.
[4] Cheng et al, "Net separation-oriented printed circuit board placement via margin maximization", ASP-DAC, 2022

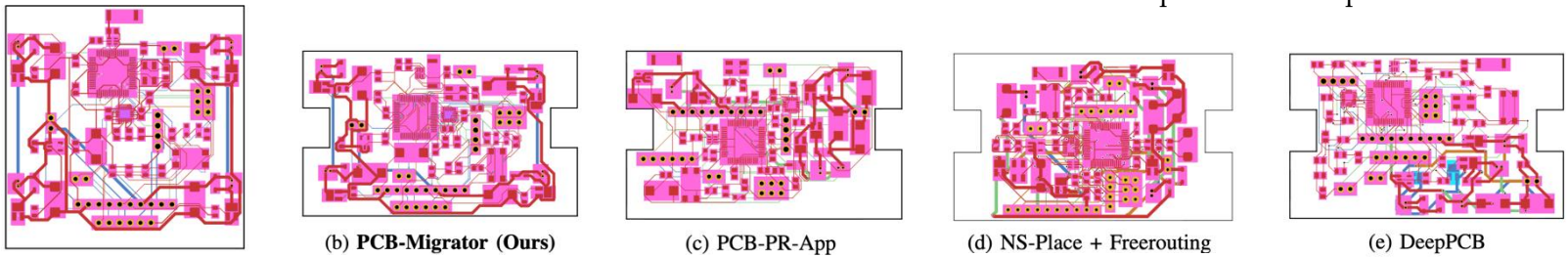
Experiment Results

PCB-Migrator can achieve excellent EM performance close to that of the expert results.

PCB-Migrator's layout is highly similar to the reference and more reasonable, resulting in improved EM performance.



Comparison of EM performance.



(a) Referenced PCB layout (b) **PCB-Migrator (Ours)** (c) PCB-PR-App (d) NS-Place + Freerouting (e) DeepPCB

Comparison of PCB layouts from expert design, baselines and our PCB-Migrator.

Thanks for Your Attention!