International Symposium on Physical Design

IncreMacro: Incremental Macro Placement Refinement

Yuan Pu[1,2], Tinghuan Chen[2], Zhuolun He[1,2], Chen Bai[2], Haisheng Zheng[1], Yibo Lin[3], Bei Yu[2]

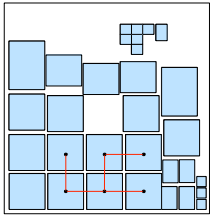
[1]Shanghai AI Laboratory, Shanghai, China
[2]The Chinese University of Hong Kong, Hong Kong SAR
[3]Peking University, Beijing, China

Outline

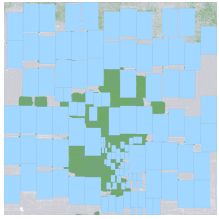
Introduction

Background

- Macro placement plays an important role for the QoR of the following physical design flows (eg. std placement, routing).
- Analytical placers (e.g. DREAMPlace[1]) and AutoDMP[2] may cause macro blockage, leading to discontinuous space for standard cell placement, thus bad wirelength, routability and timing.



(a) Macro placement by analytical placers.



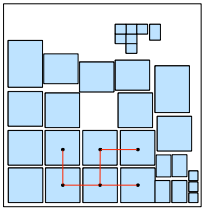
(b) Macro placement by DREAMPlace

[1] Yibo Lin et al. (2019). "Dreamplace: Deep learning toolkit-enabled GPU acceleration for modern vlsi placement". In: *Proc. DAC*, pp. 1–6.

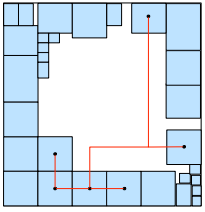
[2] Anthony Agnesina et al. (2023). "AutoDMP: Automated DREAMPlace-based Macro Placement". In: *Proc. ISPD*, pp. 149–157.

Previous Work[3]

- Data Structure-based Metaheuristics
- Methodology:
  - Three-stage macro placement. (**placement prototype→macro placement→standard cell placement**).
  - Macro positions → specific data structure (e.g., MP tree[3]).
  - Simulated Annealing: iteratively perturbation.
- Objective: push macros to chip boundary, optimize macro displacement, wirelength, routability, etc.
- Disadvantage:
  - Disturb relative macro positions by placement prototype → disturb the timing optimization by analytical placers.



(a) Macro placement by analytical placers



(b) Macro placement by academic methods

[3]Tung-Chieh Chen et al. (2007). "MP-trees: A packing-based macro placement algorithm for mixed-size designs". In: *Proc. DAC*, pp. 447–452.

Previous Work

- RL-based Approach[4][5].
- Macro placement → sequential Markov Decision Process (MDP).
  - State: intermediate macro placement solution.
  - Action: assign one macro to a legalized position on the chip.
- Generate macro placement solution **from scratch**.
- Disadvantages:
  - From scratch→ computationally prohibitive for parameters exploration.
  - No utilization of timing-optimization by analytical placement prototype.

[4] Ruoyu Cheng and Junchi Yan (2021). "On Joint Learning for Solving Placement and Routing in Chip Design". In: *Proc. NeurIPS* 34, pp. 16508–16519.

[5] Azalia Mirhoseini et al. (2020). "Chip placement with deep reinforcement learning". In: *arXiv preprint arXiv:2004.10746*.

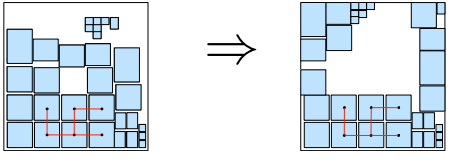
Problem Formulation

Incremental Macro Placement Refinement

Given the placement prototype by an analytical placer, find the legalized coordinates of macros to optimize the objective of wirelength.

Requirements satisfied by IncreMacro:

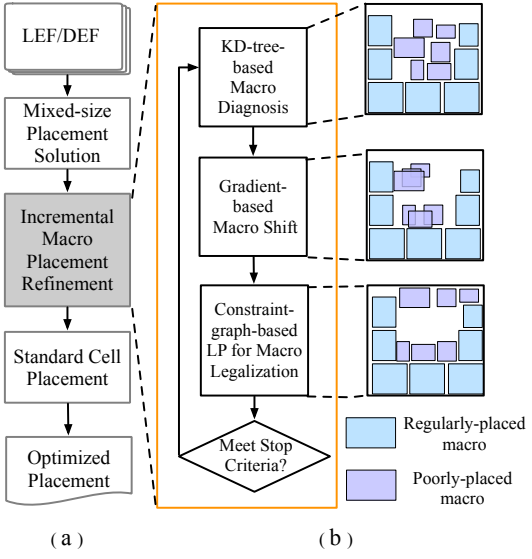
- Pushing macros to chip periphery.
- Macro relative position (timing-opt by analytical placers) preserved.



Algorithm

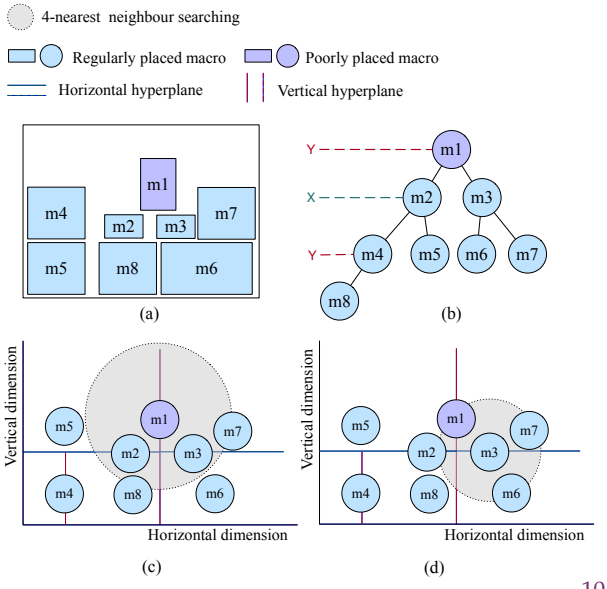
Framework

- Overall flow of IncreMacro:

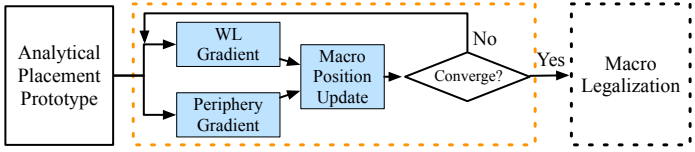


KD-tree-based Macro Diagnosis

- Regularly-placed macro satisfies one of the following conditions:
  - clings to chip boundary
  - nearest four 4 macros surround it
- Motivation: only poorly-placed macros need to be adjusted.
- Methodology for macro diagnosis: KD-tree based nearest neighbor search
- Example:
  - m_1: **poorly-placed**
  - m_3: **regularly-placed**



Gradient-based Macro Shift



The Macro shifting gradient descent process.

- Make analogy between macro placement and deep learning:
  - macro coordinate offset $\rightarrow$ model weight
  - wirelength and periphery cost $\rightarrow$ cost function
  - netlist $\rightarrow$ data instance
- Implemented by pytorch
- Only shift poorly-placed macros

$$\min \quad \mathcal{L} = \sum_{e \in Net_{macro}} WL_e + \alpha \sum_{m_i \in \mathcal{M}_{poor}} P_i, \tag{1}$$

where WL_e is wirelength cost for net e, P_i is periphery cost for macro m_i.

Gradient-based Macro Shift: Wirelength Cost

- Half-perimeter wirelength (HPWL) for net e:

$$HPWL_e = \max_{v_i \in e}\{x_i\} - \min_{v_i \in e}\{x_i\} + \max_{v_i \in e}\{y_i\} - \min_{v_i \in e}\{y_i\}. \tag{2}$$

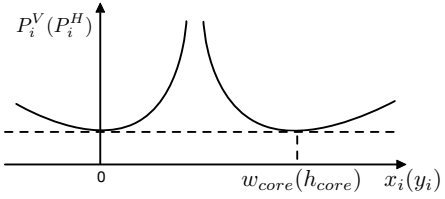
- HPWL is not differentiable.
- Weighted-avergae wirelength (WA)[6] for a net e:

$$WA_e = \frac{\sum_{v_i \in e} x_i e^{\frac{x_i}{\gamma}}}{\sum_{v_i \in e} e^{\frac{x_i}{\gamma}}} - \frac{\sum_{v_i \in e} x_i e^{-\frac{x_i}{\gamma}}}{\sum_{v_i \in e} e^{-\frac{x_i}{\gamma}}} + \frac{\sum_{v_i \in e} y_i e^{\frac{y_i}{\gamma}}}{\sum_{v_i \in e} e^{\frac{y_i}{\gamma}}} - \frac{\sum_{v_i \in e} y_i e^{-\frac{y_i}{\gamma}}}{\sum_{v_i \in e} e^{-\frac{y_i}{\gamma}}}. \tag{3}$$

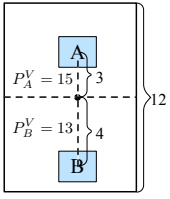
- Use WA as wirelength cost.

[6] Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang (2013). "TSV-aware analytical placement for 3-D IC designs based on a novel weighted-average wirelength model". In: *IEEE TCAD* 32.4, pp. 497–509.

Gradient-based Macro Shift: Periphery cost



(a) Visualization of the peripheral cost

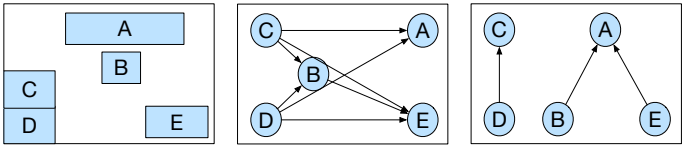


(b) Vertical peripheral cost

- Motivation: push macros to chip boundary.

$$
\begin{aligned}
P_i^H &= \left|\frac{w_{core}}{2} - x_i\right| + \frac{(\frac{w_{core}}{2})^2}{\left|\frac{w_{core}}{2} - x_i\right|}, \\
P_i^V &= \left|\frac{h_{core}}{2} - y_i\right| + \frac{(\frac{h_{core}}{2})^2}{\left|\frac{h_{core}}{2} - y_i\right|}, \\
P_i &= P_i^H + P_i^V .
\end{aligned} \tag{4}
$$

Constraint-graph based LP for Macro Legalization[7]



(c) Design Sample (d) Horizontal constraint graph (e) Vertical constraint graph

- Purpose:
  - Eliminate overlaps generated by last step (Macro shift)
  - Further push macros to nearest chip periphery
- Constraint graphs:
  - Relative positional relationship among macros, horizontally and vertically.
  - Constructed on the macro placement prototype by analytical placers.
- Objective for LP (minimization):
  - Macro displacement
  - Distance to the nearest periphery

[7] Jason Cong and Min Xie (2006). "A robust detailed placement for mixed-size IC designs". In: *Proc. ASPDAC*, 7–pp.

Macro Legalization: Linear Programming Formulation

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{|M_{poor}|} |x_i' - x_i| + |y_i' - y_i| + \min(x_i, W - x_i) + \min(y_i, H - y_i) \\
\text{s.t.} \quad & x_j' - x_i' \geq \frac{w_i + w_j}{2}, && \forall e_{ij} \in G_h \\
& y_j' - y_i' \geq \frac{h_i + h_j}{2}, && \forall e_{ij} \in G_v \\
& \frac{w_i}{2} \leq x_i' \leq W - \frac{w_i}{2}, \frac{h_i}{2} \leq y_i' \leq H - \frac{h_i}{2}. && \forall m_i \in \mathcal{M}_{poor}
\end{aligned}
\tag{5}
$$

- x_i/x_i': x-coordinate before/after macro legalization.
- Original objective for macro displacement: $|x_i' - x_i| + |y_i' - y_i|$.
- $\min(x_i, W - x_i) + \min(y_i, H - y_i)$: distance to the nearest boundary.
- First and second inequality: non-overlap constraint.
- Last inequality: out of boundary constraint.

Macro Legalization: Linear Programming Formulation (Con't)

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{|M_{poor}|} x_i^p + x_i^q + y_i^p + y_i^q + \min(x_i, W - x_i) + \min(y_i, H - y_i) \\
\text{s.t.} \quad & x_i^p - x_i^q = x_i' - x_i,\ y_i^p - y_i^q = y_i' - y_i, && \forall m_i \in \mathcal{M}_{poor} \\
& x_i^p \geq 0,\ x_i^q \geq 0,\ y_i^p \geq 0,\ y_i^q \geq 0, && \forall m_i \in \mathcal{M}_{poor} \\
& x_j' - x_i' \geq \frac{w_i + w_j}{2}, && \forall e_{ij} \in G_h \\
& y_j' - y_i' \geq \frac{h_i + h_j}{2}, && \forall e_{ij} \in G_v \\
& \frac{w_i}{2} \leq x_i' \leq W - \frac{w_i}{2}, \frac{h_i}{2} \leq y_i' \leq H - \frac{h_i}{2}. && \forall m_i \in \mathcal{M}_{poor}
\end{aligned}
\tag{6}
$$

- First and second constraints: transformation on displacement objective (remove absolute value).

Experimental Results

Experiment Setting

- Benchmark: OpenSouce RISC-V SOCs from chipyard[8].
- PDK: ASAP7[9].
- VLSI flow: standard cell placement $\rightarrow$ CTS $\rightarrow$ routing (commercial tool).
- Baseline: DREAMPlace, AutoDMP.

Table: Design information

Benchmark	# Macros	# Std cells	# Nets	freq. (MHz)
Rocket	121	203633	208595	500.0
GemminiRocket	737	1176141	1189717	333.3
Sha3Rocket	121	235944	240907	666.7
LargeBoomAndRocket	636	856576	874017	333.3
FFTRocket	121	211543	216507	1000.0
SmallBoom	314	362479	372623	500.0
HwachaRocket	292	679667	687204	250.0

[8] Alon Amid et al. (2020). "Chipyard: Integrated design, simulation, and implementation framework for custom SoCs". In: *IEEE Micro* 40.4, pp. 10–21.

[9] Lawrence T Clark et al. (2016). "ASAP7: A 7-nm FinFET predictive process design kit". In: *Microelectronics Journal* 53, pp. 105–115.

Experiment: Post-Route (DREAMPlace)

Table: Post-Route PPA results for the incorporation of IncreMacro to DREAMPlace

Benchmark	DREAMPlace (two stage placement)					DREAMPlace + IncreMacro				
	WL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)	WL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)
Rocket	14838552	-1.40	-67.16	72.54	64.55	**11794141**	**-0.05**	**-0.88**	**68.90**	101.77
GemminiRocket	**62420588**	-3.06	-15146.20	520.73	207.83	69708870	**-2.45**	**-4927.20**	**497.50**	418.82
Sha3Rocket	11584499	-0.22	**-40.84**	95.49	77.15	**11570282**	**-0.21**	-48.60	**95.08**	99.61
LargeBoomAndRocket	36201457	0	0	132.57	298.71	**33286377**	0.09	0	**131.46**	533.97
FFTRocket	10740986	-0.01	-0.02	70.80	63.19	**9716426**	**0.02**	**0**	**67.06**	72.89
SmallBoom	13967377	-0.13	-22.01	98.10	112.40	**13547457**	**0.01**	**0**	**96.55**	153.29
HwachaRocket	48096700	-0.10	-2.82	145.55	146.64	**40552488**	**0.03**	**0**	**137.48**	253.25
Normalize	1.065	1.599	1.639	1.033	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.600

Performance Improvement:

- wirelength: +6.5%
- power: +3.3%
- WNS: +59.9%
- TNS: +63.9%

Experiment: Post-Route (AutoDMP)

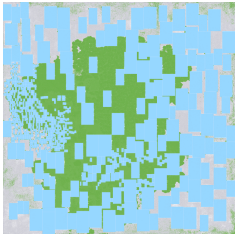
Table: Post-Route PPA results for the incorporation of IncreMacro to AutoDMP

Benchmark	AutoDMP (two-stage placement)					AutoDMP + IncreMacro				
	WL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)	WL (um)	WNS (ns)	TNS (ns)	Power (mW)	Runtime (s)
Rocket	14704880	-0.41	-354.87	74.33	69.24	**11186898**	**0.01**	**0**	**68.43**	82.99
GemminiRocket	41031737	-0.24	-17.76	301.39	205.95	**34580828**	**-0.01**	**-0.01**	**287.51**	381.58
FFTRocket	**8108137**	-0.001	-0.001	55.43	55.14	8176606	**0.05**	**0**	**55.31**	101.73
HwachaRocket	31675727	-0.05	-0.14	130.37	193.24	**22684613**	**0.35**	**0**	**121.66**	321.65
Normalize	1.168	1.996	1.999	1.049	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	1.640

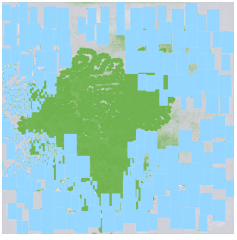
Performance Improvement:

- wirelength: +16.8%
- power: +4.9%
- WNS: +99.6%
- TNS: +99.9%

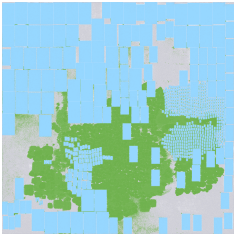
Result Visualization



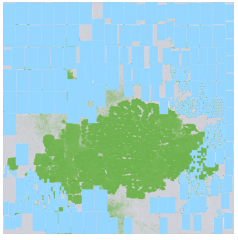
(f) GemminiRocket: DREAMPlace



(g) GemminiRocket: DREAMPlace + IncreMacro

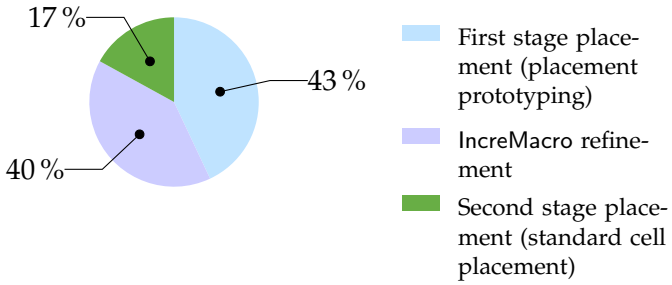


(h) GemminiRocket: AutoDMP



(i) GemminiRocket: AutoDMP + IncreMacro

Runtime Analysis



Average runtime breakdown of IncreMacro + DREAMPlace.

Conclusion

Conclusion

- IncreMacro: incrementally enhances macro placement by SOTA analytical placers.
- Three main techniques:
  - KD- tree-based macro diagnosis
  - gradient-based macro shift
  - constraint-graph-based linear programming for macro legalization
- Two requirements for macro placement:
  - push macros to the chip boundary
  - Macro relative position preserved

THANK YOU!