# Assessment of Reinforcement Learning for Macro Placement

**Chung-Kuan Cheng[1], Andrew B. Kahng[1][2], Sayak Kundu[2], Yucheng Wang[1] and Zhiang Wang[2]**

[1]CSE and [2]ECE Departments, UC San Diego

GitHub: https://github.com/TILOS-AI-Institute/MacroPlacement

# June 2021: Google's Nature Paper

- Google Brain's highly acclaimed *Nature* paper proposed a reinforcement learning (RL) based macro placer
  - **Did not release** code or dataset

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature › articles › article

Article | Published: 09 June 2021

## A graph placement methodology for fast chip design

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter & Jeff Dean

*Nature* **594**, 207–212 (2021) | Cite this article

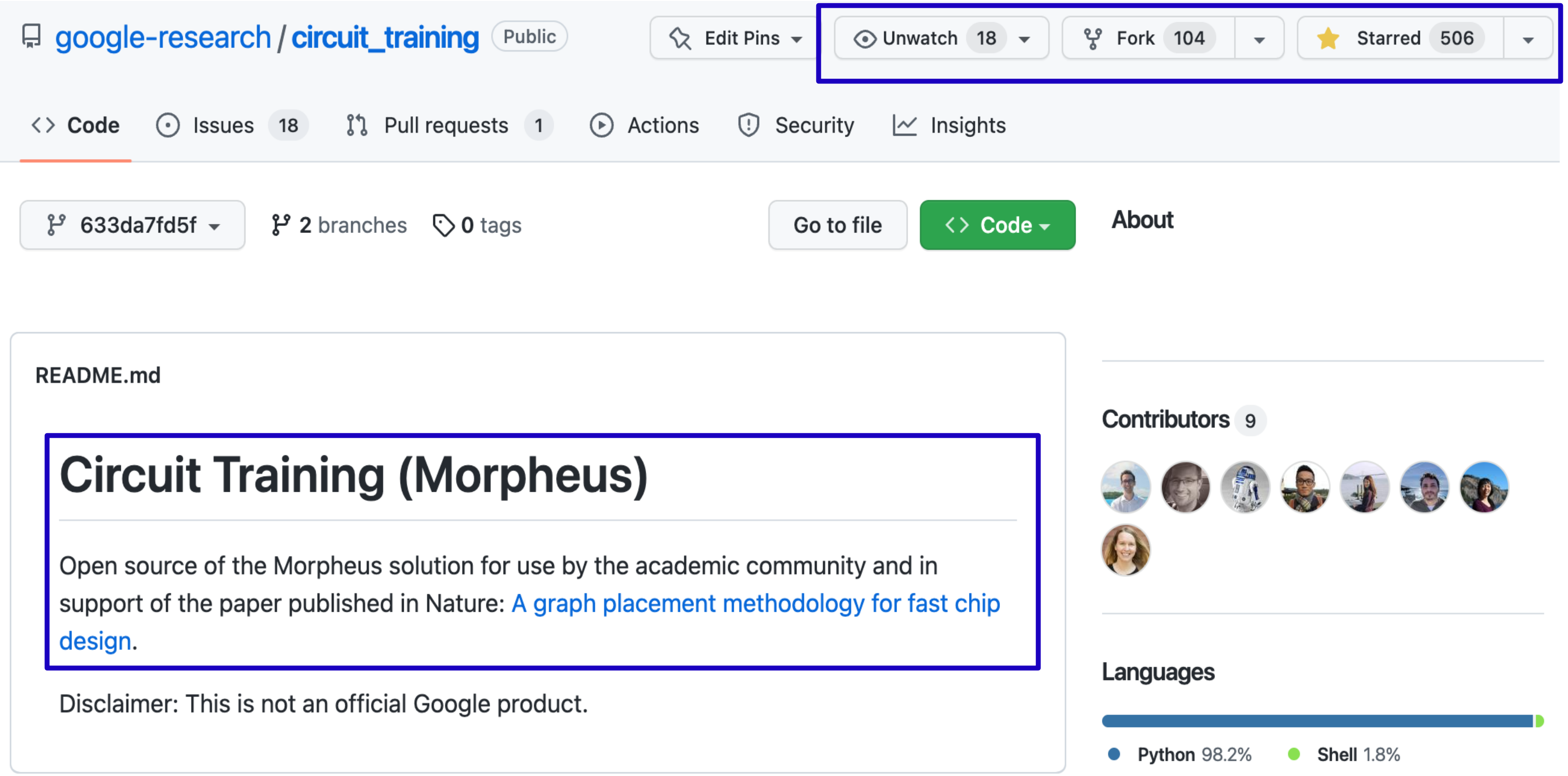
**43k** Accesses | **98** Citations | **2077** Altmetric | Metrics

Claimed **superior** or **comparable** macro placement solutions compared to human experts, **in under six hours!**

UCSD

# January 2022: Google's Circuit Training

- Circuit Training open-sourced *"reproduces the methodology published in the Nature 2021 paper"*
  - **Missing** dataset and code elements: format translator, simulated annealing




**No dataset, and insufficient code to reproduce the *Nature* results**

UCSD

# March 28, 2022: Stronger Baselines

- “Stronger Baselines” manuscript was made available
  - Evaluated on Google’s **internal** benchmarks and **old** academic benchmarks
  - Used **weak** evaluation metrics
  - **No open-source** code

**Stronger Baselines for Evaluating Deep Reinforcement Learning in Chip Placement**

**States shortcomings of the *Nature* paper, but also lacks code and dataset to reproduce results**

UCSD

# Why *MacroPlacement*?

**March 28, 2022**: Stronger Baselines manuscript

**Stronger Baselines for Evaluating Deep Reinforcement Learning in Chip Placement**

**States Shortcomings**

**June 2021**: ***Nature* Paper**

nature

Explore content ⌄ About the journal ⌄ Publish with us ⌄

nature > articles > article

Article | Published: 09 June 2021

**A graph placement methodology for fast chip design**

**January 2022**: **Circuit Training**

## Circuit Training (Morpheus)

Open source of the Morpheus solution for use by the academic community and in support of the paper published in Nature: A graph placement methodology for fast chip design.

Disclaimer: This is not an official Google product.

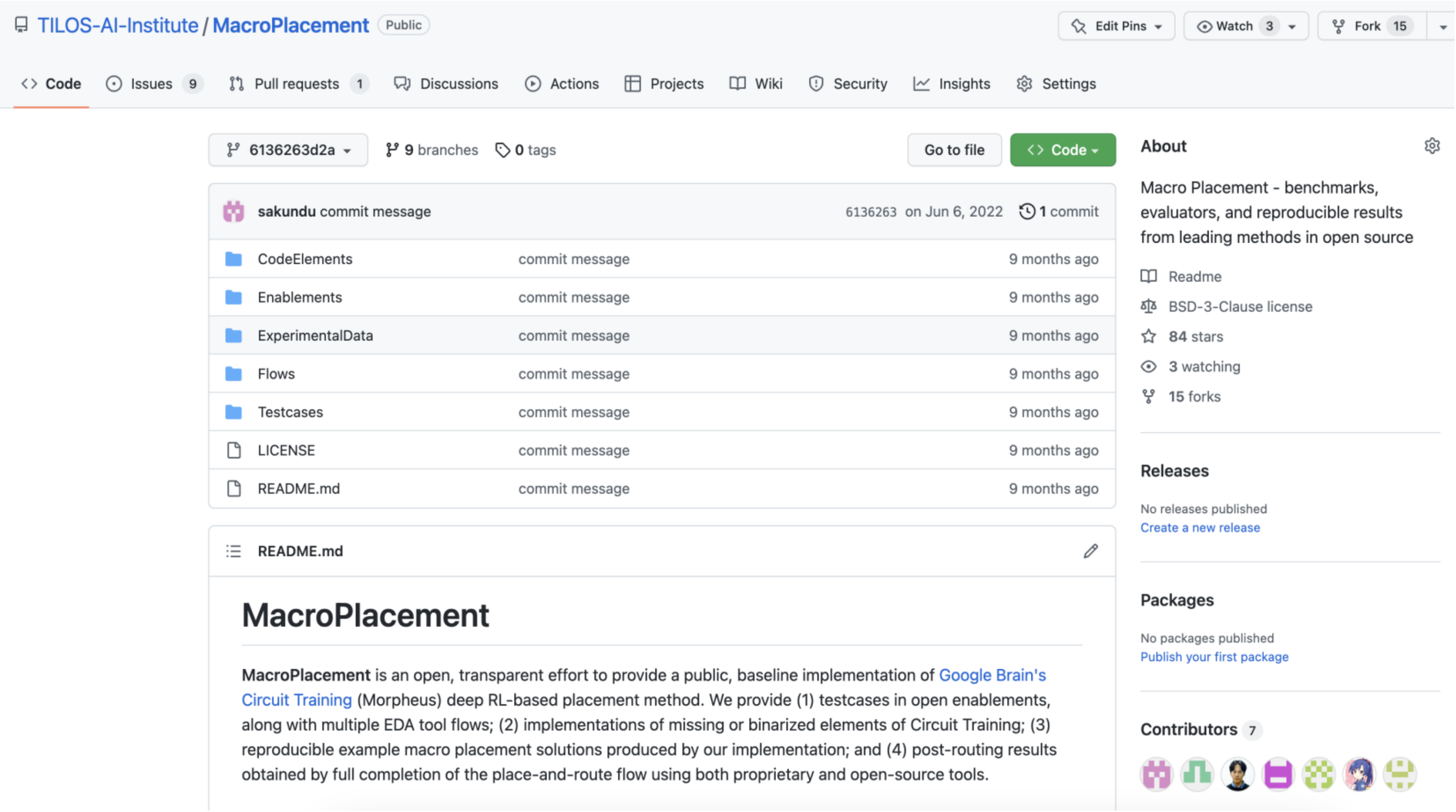
**Breakthrough Claims**

**Partial Release of Supporting Code**

# Our *MacroPlacement* Effort

- **Key elements:**
  - New testcases
  - Open enablements
  - Public evaluation flows
  - Reverse engineering
  - Reimplementation in open source
  - Stronger baselines

**June 2022**: ***MacroPlacement*** repository open-sourced

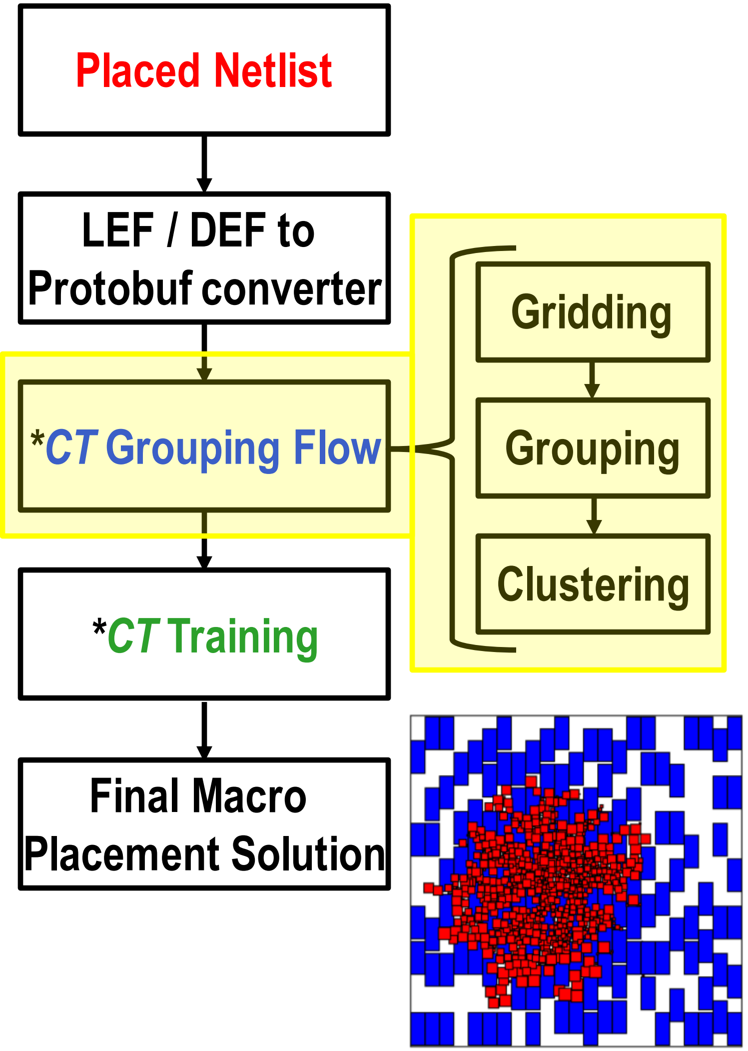



- **This talk and paper: The Story of *MacroPlacement***

# Outline

- Background and Motivation
- Replication of Circuit Training (*CT*)
  - Mismatch of *CT* and Google *Nature* paper
  - Blackbox and missing elements of *CT*
- *MacroPlacement* Repository
- Experimental Results
- Academic Benchmarks
- Conclusion

# Circuit Training (*CT*) Flow

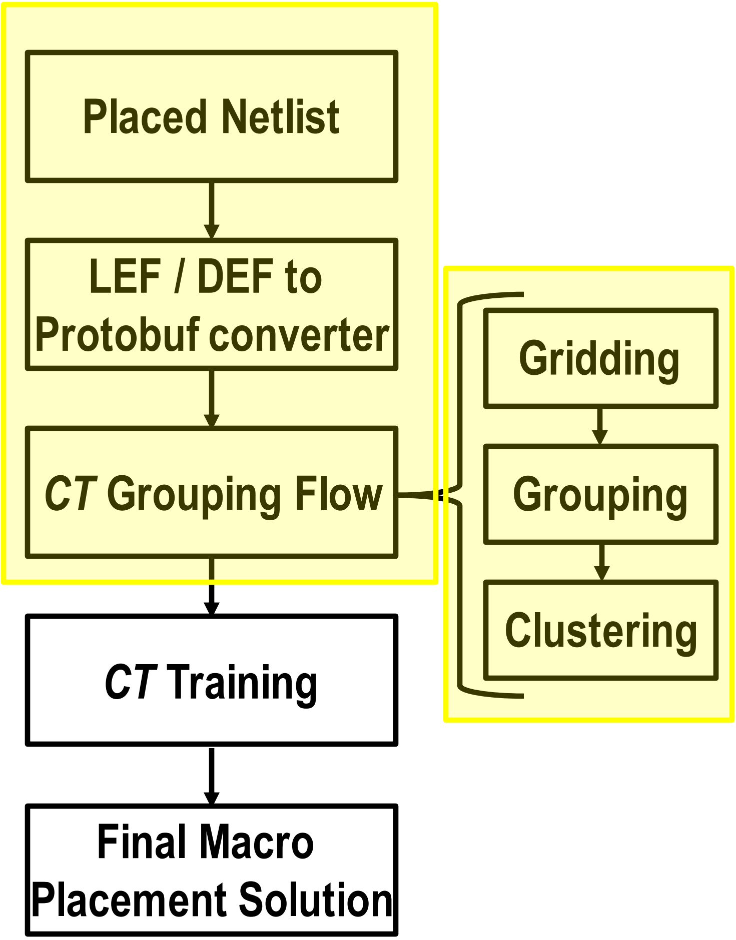



- Input to *CT*: Placed Netlist
- *CT’s* “grouping” flow consists of three steps
  - **Gridding:** divides the chip canvas into *grid cells* whose size promotes close packing of macros
  - **Grouping:** creates groups to ensure closely connected logic elements stay together
  - **Clustering:** creates standard-cell clusters
- *CT* training
  - Input: clustered netlist
  - Optimizes the proxy cost ($R$):

$$R = Wirelength + \gamma \times Density + \lambda \times Congestion \quad (1)$$

*Grouping and training flows are open-sourced in CT repository by the **TensorFlow Agents** team

# Mismatch Between *CT* and The *Nature* Paper

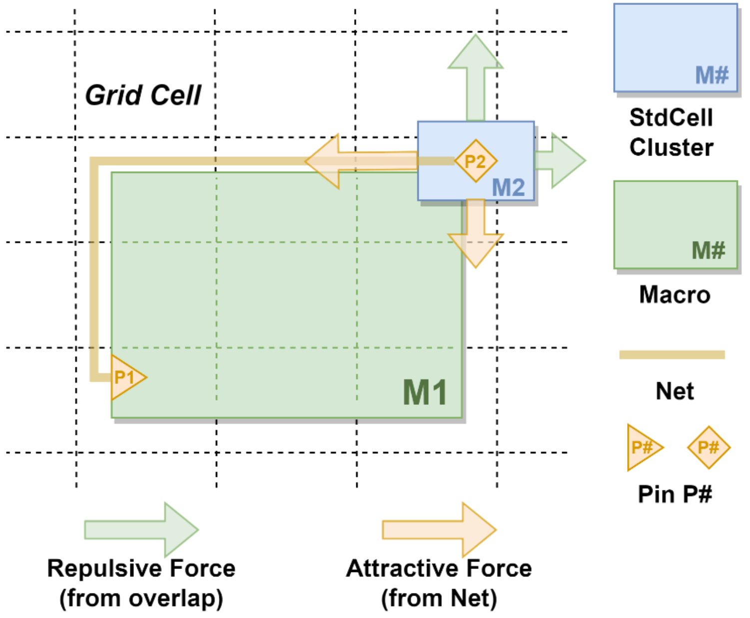



- ***CT* assumes that the input is a placed netlist**
  - *Nature* paper: does not mention this
  - **Critical:** poor initial placement increases routed wirelength by 7%
- Different weight of proxy cost components
  - *Nature*: $\gamma = 0.01$ and $\lambda = 0.01$
  - *CT*: $\gamma = 1.0$ and $\lambda = 0.5$
  - Suggested by Google: $\gamma = 0.5$ and $\lambda = 0.5$*
  - Result for different weight combinations in Slide 25
- Adjacency matrix generation
  - Nature considers register distance for a pair of nodes
  - *CT* does not consider register distance

*G. Wu, Google Brain, personal communication, August 2022

# Blackbox Element: Force-Directed Placement

- Main **blackbox** elements of *CT*: hidden behind *plc_client*
  - Force-directed (FD) placement
  - Proxy cost components: Wirelength, Density and Congestion cost




- FD consists of two force components
  - Attractive force: $F_{a_x}$, attractive factor: $k_a$

$$F_{a_x} = k_a \times abs(P1.x - P2.x) \quad (2)$$

  - Repulsive force: $F_{r_x}$, repulsive factor: $k_r$ and max repulsive force: $F_{r_{max}}$

$$F_{r_x} = k_r \times F_{r_{max}} \times \frac{abs(M1.x - M2.x)}{dist(M1, M2)} \quad (3)$$

→ Horizontal force $F_x = F_{a_x} + F_{r_x} \quad (4)$

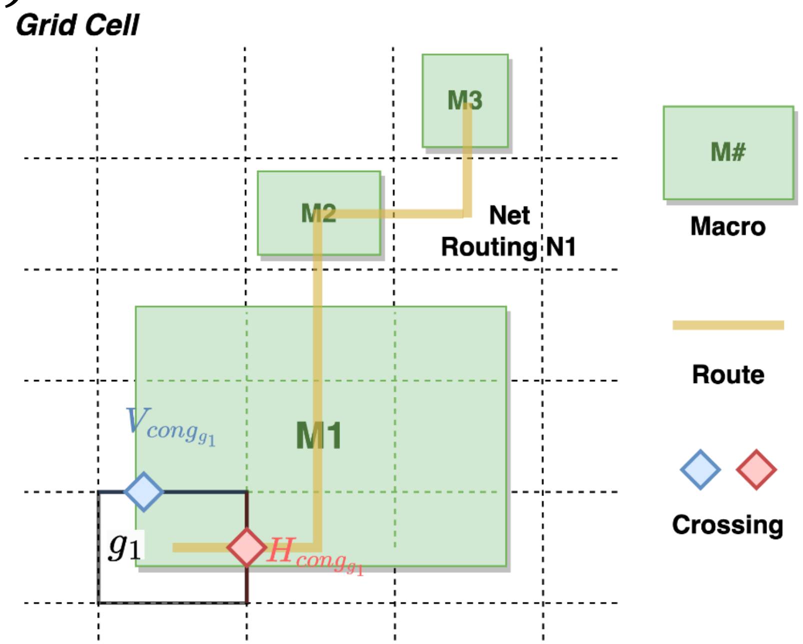
- More details: **Section 3.2.1** of our paper

UCSD

# Blackbox Element: Proxy Cost

- **Wirelength cost:**

$$\frac{1}{|nets|}\sum_{net}\frac{net.weight \times HPWL(net)}{canvas.width + canvas.height} \quad (5)$$




- **Density cost:** Average density of the top 10% densest grid cells
- **Congestion cost:** Average of top 5% grid cell $H_{cong}$ and $V_{cong}$ values
- Two components of congestion cost:
  - **Macro congestion:** Routing layers blocked by macros
  - **Routing congestion:** Routing resources occupied by routed nets

  → **Grid cell congestion:** Sum of macro and routing congestion

**RL Agent optimizes these proxy cost components**

# Missing Elements: Format Translator and SA

- Format translators to generate Protobuf netlist
  - *MacroPlacement* repo includes two format translators
    - LEF/ DEF → Protobuf
    - Bookshelf → Protobuf
- Simulated Annealing (SA) implementation
  - Our SA implementation follows the Stronger Baselines (SB) description
  - **Five actions** of our SA
    - *Nature* includes **swap**, **shift**, and **mirror** actions
    - *Nature* does not include **move** and **shuffle**
  - **Two initializations**
    - *Nature* includes **greedy packing**
    - *Nature* does not include **spiral** initialization

**Our implementation of missing elements enables anyone to run CT and SA on their own designs !!**

UCSD

# Outline

- Background and Motivation
- Replication of Circuit Training (*CT*)
- *MacroPlacement* Repository
  - New benchmarks
  - Commercial evaluation flow
- Experimental Results
- Academic Benchmarks
- Conclusion

# *MacroPlacement:* Modern Benchmarks

- **Modern benchmarks**: Open testcases on open enablements
  - **Testcases:** Ariane, BlackParrot, MemPool Group and NVDLA
  - **Enablements:** SKY130HD FakeStack, NanGate45 and ASAP7 (includes FakeRAM generator)

| Testcase | #FFs | #Macros | #Macro Types | #Insts on NG45 |
|---|---|---|---|---|
| **Ariane** | 19,807 | 133 | 1 | 117,433 |
| **NVDLA** | 45,295 | 128 | 1 | 155,711 |
| **BlackParrot** | 214,441 | 220 | 6 | 768,631 |
| **MemPool Group** | 360,724 | 324 | 4 | 2,729,405 |

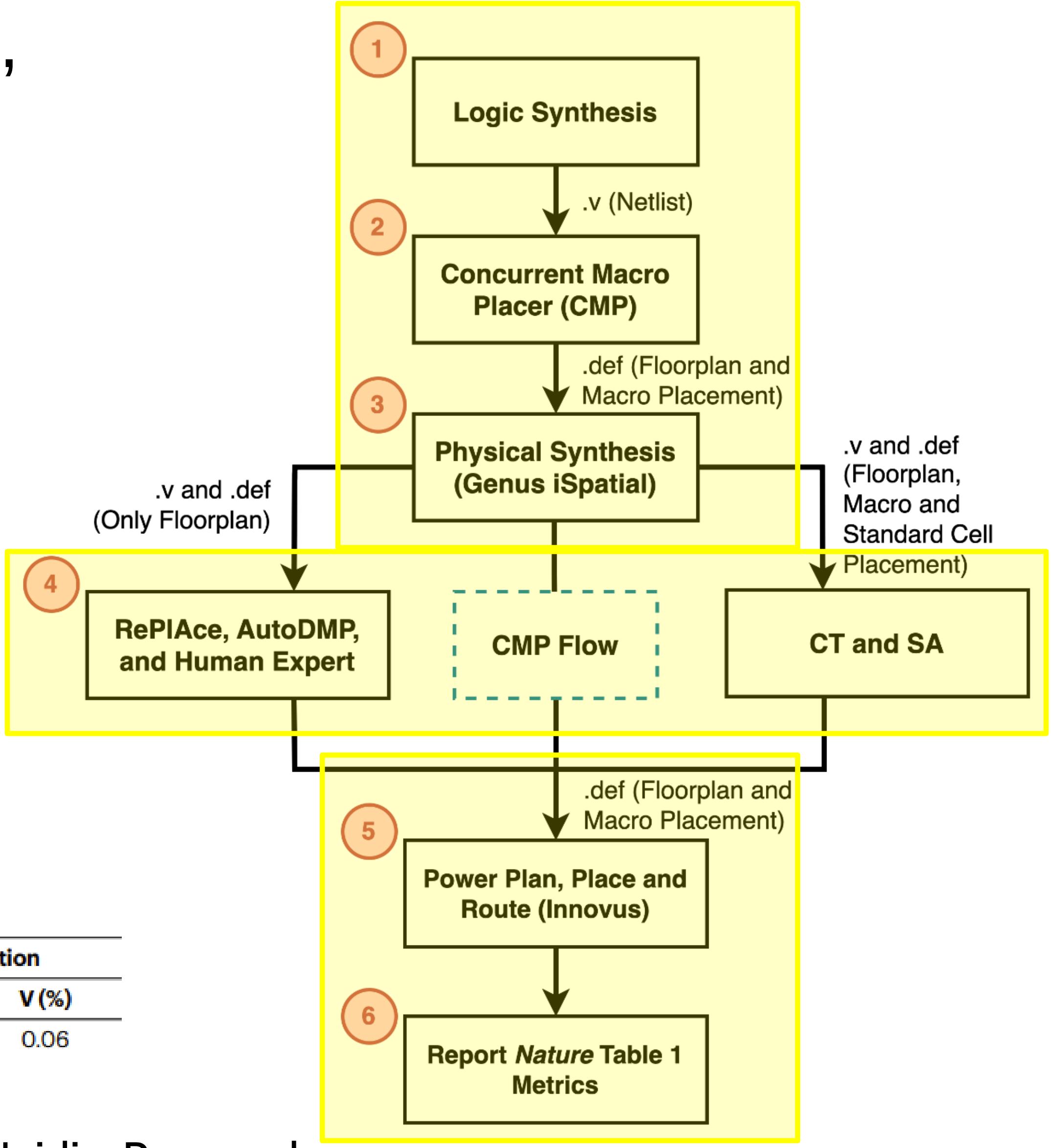
- **Major changes in EDA vendor policies allow us to share our Tcl scripts in GitHub for research purposes!** ← ***Kudos and thanks to Cadence and Synopsys !!!***
- Our repo includes commercial synthesis, place-and-route (SP&R) tool flow scripts
  - Cadence Genus iSpatial physical synthesis flow and Cadence Innovus P&R flow
  - Synopsys Design Compiler Topographical physical synthesis flow

# Commercial Evaluation Flow

- **Macro placers:** *CT*, *CMP, SA, RePlAce, **AutoDMP and Human expert
- Logic synthesis: Genus 21.1
- Physical synthesis:
  - Genus iSpatial flow
  - Design Compiler Topographical R-2020.09
- Place and route: Cadence Innovus 21.1
- **Ground truth / *Nature* Table 1 metrics**:
  - ***postRouteOpt wirelength, WNS, TNS, power, standard cell area and DRC count

**Table 1 | Comparisons against baselines**

| Name | Method | Timing | | Total area (μm²) | Total power (W) | Wirelength (m) | Congestion | |
|---|---|---|---|---|---|---|---|---|
| | | WNS (ps) | TNS (ns) | | | | H (%) | V (%) |
| Block 1 | RePlAce | 374 | 233.7 | 1,693,139 | 3.70 | 52.14 | 1.82 | 0.06 |




*CMP: Cadence Innovus Concurrent Macro Placer
**AutoDMP: DREAMPlace-based macro placer from Nvidia Research
****Nature* paper reports postPlaceOpt metrics

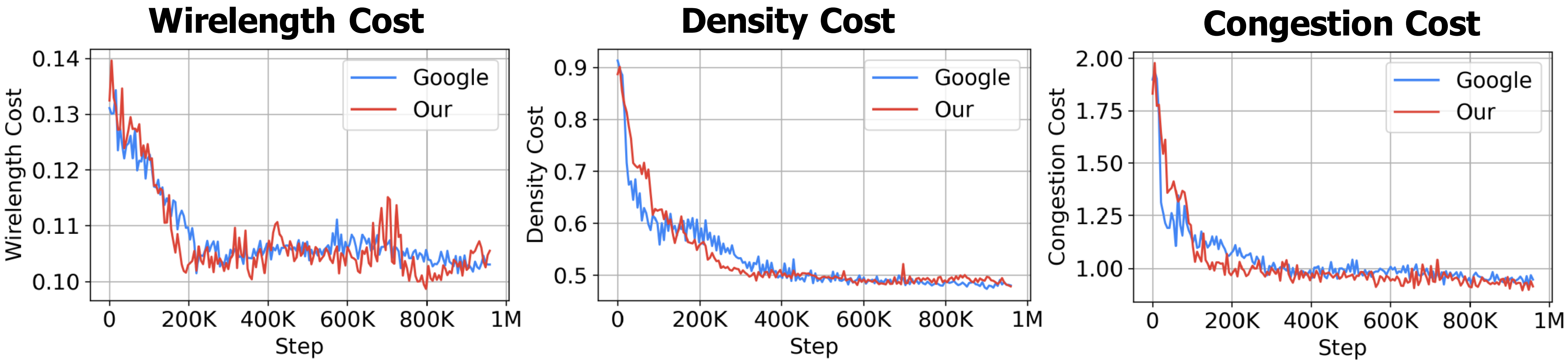
UCSD

# Outline

- Background and Motivation
- Replication of Circuit Training (*CT*)
- *MacroPlacement* Repository
- Experimental Results
  - Ablation studies
  - Different macro placement solutions
- Academic Benchmarks
- Conclusion

# Ablation Study of *CT* (Section 5.2)

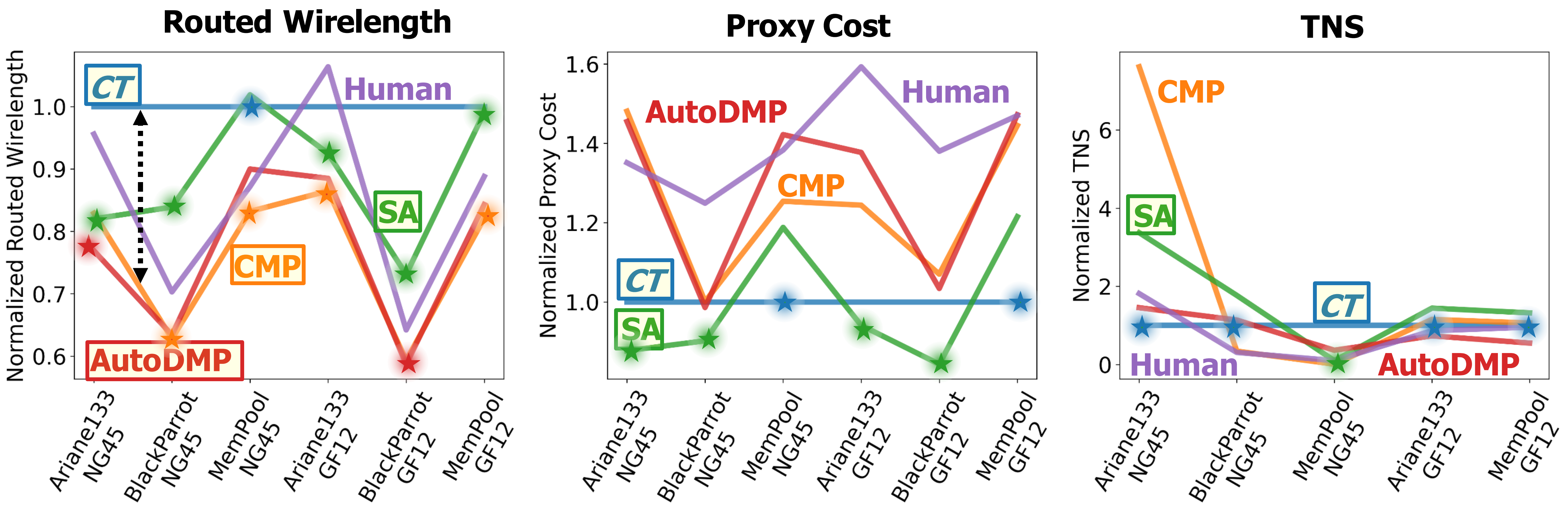
- Similar training curve and *Nature* Table 1 metrics of Ariane-NG45 for our and Google's *CT* runs → **Correct *CT* Setup** (Ref. [50] of our paper)




- **Effect of initial placement**
  - Ran *CT* for three vacuous placements: all standard cells and macros are placed at lower left corner, at upper right corner and at (600, 600)
  - The routed wirelength of our baseline *CT* solution is 7.24%, 8.17% and 10.32% less than the above three cases respectively

**Initial placement is important for CT !!**

# *Nature* Table 1 Metrics: Modern Benchmarks




CMP outperforms *CT*

- Lower values in normalized routed wirelength, proxy cost, and TNS → Better performance
- Data is normalized based on *CT*
- More details: **Table 1** of our paper

# Human Outperforms *CT* for Macro-Heavy Designs

| Design Enablement | Macro Placers | Area (um²) | rWL (mm) | Power (mW) | WNS (ps) | TNS (ns) |
|---|---|---|---|---|---|---|
| BlackParrot NG45 | *CT* | 1,956,712 | 36,845 | 4627.4 | -185 | -1040.8 |
| | Human | 1,919,928 | 25,916 | 4469.6 | -97 | -321.9 |
| MemPool NG45 | *CT* | 4,890,644 | 123,330 | 2760.5 | -69 | -119.3 |
| | Human | 4,873,872 | 107,598 | 2640.0 | -49 | -11.9 |
| Note: Metric values for GF12 are normalized | | | | | | |
| BlackParrot GF12 | *CT* | 0.179 | 1.000 | 1.000 | 0.000 | 0.0 |
| | Human | 0.178 | 0.642 | 0.928 | 0.000 | 0.0 |
| MemPool GF12 | *CT* | 0.410 | 1.000 | 1.000 | -0.195 | -1849.4 |
| | Human | 0.406 | 0.888 | 0.920 | -0.149 | -1766.5 |

**For GF12 metric values**

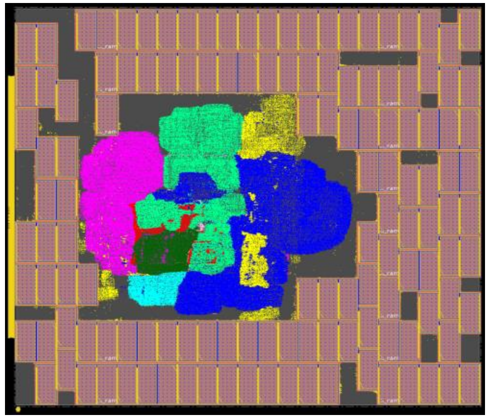
- Routed wirelength and total power: normalized based on *CT* result
- TNS and WNS: normalized based on target clock frequency
- Standard cell area: normalized based on canvas area

**In terms of all the *Nature* Table 1 metrics, Human outperforms *CT* for macro-heavy testcases**

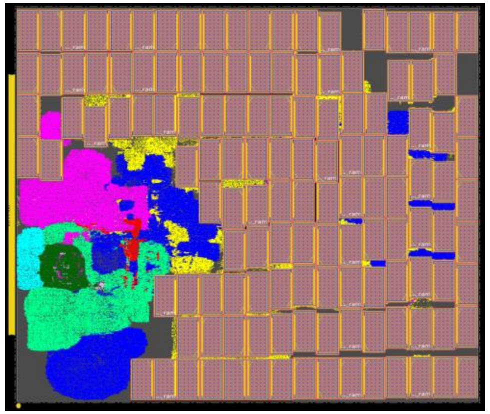
# Macro Placement Solutions: Ariane133

- *__Ariane133__ with **68%** utilization, **1.3ns** target clock period on **NG45**

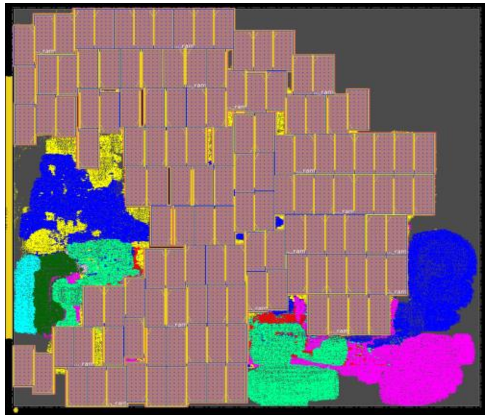
*CT*



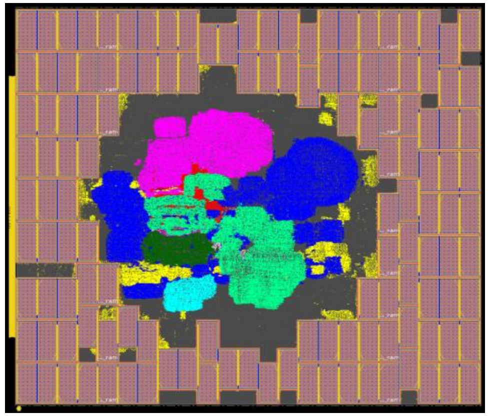
**CMP**



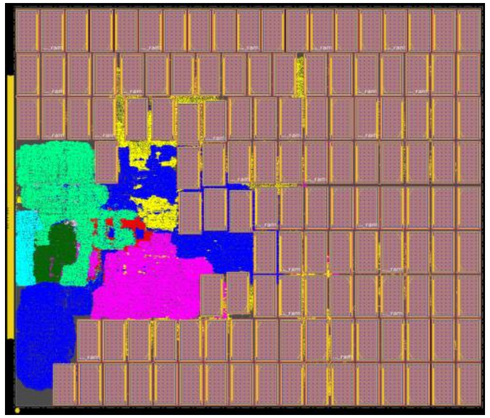
**RePlAce**



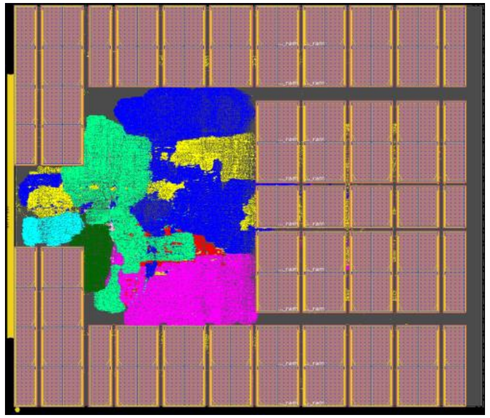
**SA**



**AutoDMP**



**Human**



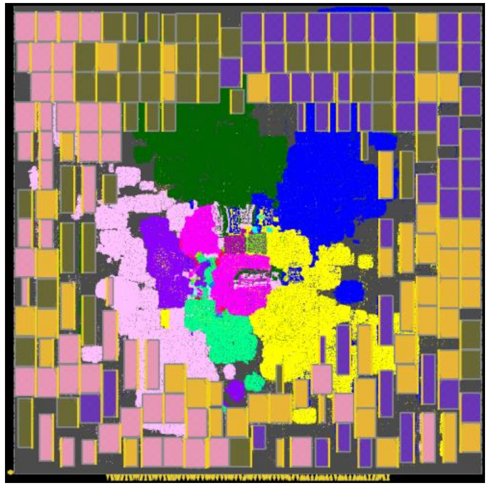
| Macro Placer | Area ($\mu m^2$) | rWL ($mm$) | Power ($mW$) | WNS ($ps$) | TNS ($ns$) | Proxy Cost |
|---|---|---|---|---|---|---|
| CT | 244,022 | 4,894 | 828.7 | **-79** | **-25.8** | 0.857 |
| CMP | 256,230 | 4,057 | 851.5 | -154 | -196.5 | 1.269 |
| RePlAce | 252,444 | 4,609 | 843.9 | -103 | -69.9 | 1.453 |
| SA | 248,344 | 4,014 | 831.9 | -111 | -87.0 | **0.752** |
| AutoDMP | **243,720** | **3,764** | **821.7** | -95 | -37.5 | 1.247 |
| Human | 249,034 | 4,681 | 832.4 | -88 | -46.8 | 1.158 |

*_Nature_ implements Ariane133 on a different enablement

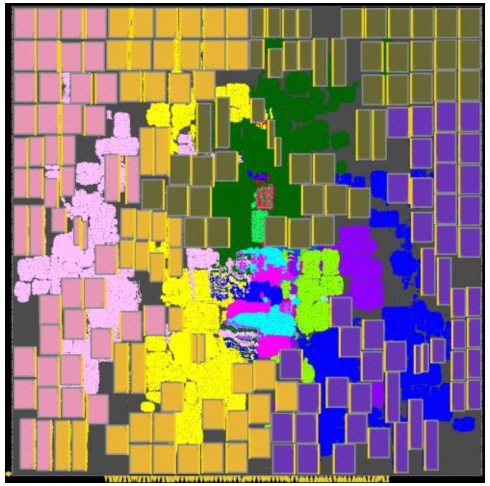
# Macro Placement Solutions: BlackParrot

- **BlackParrot (Quad-Core)** with **68%** utilization on **GF12**

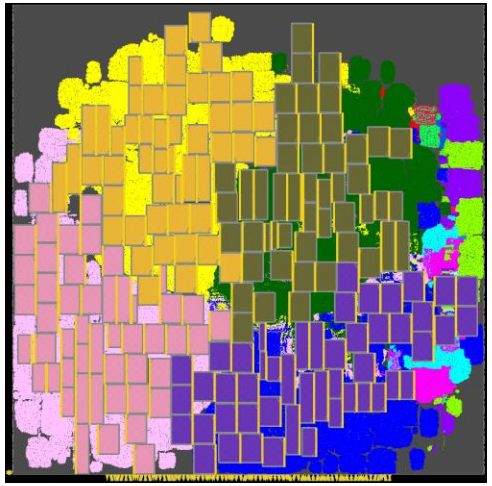
***CT***



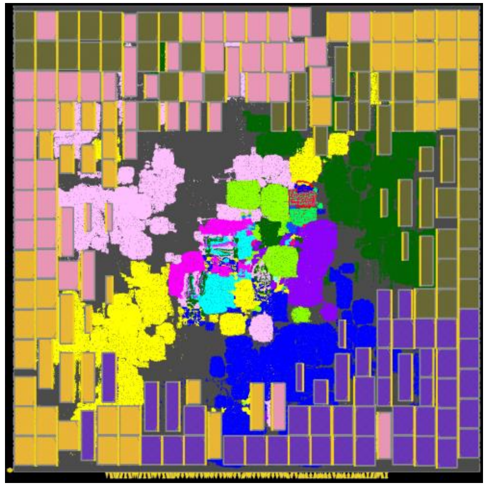
**CMP**



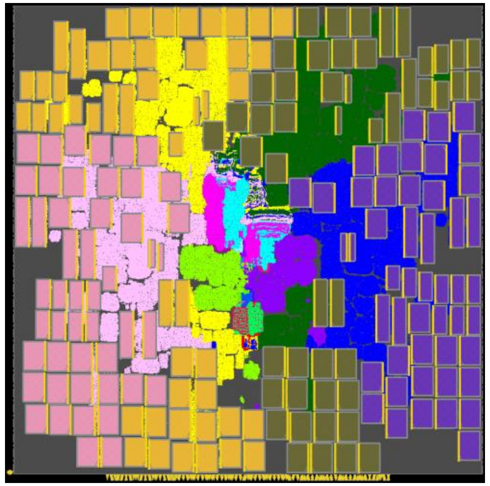
**RePlAce**



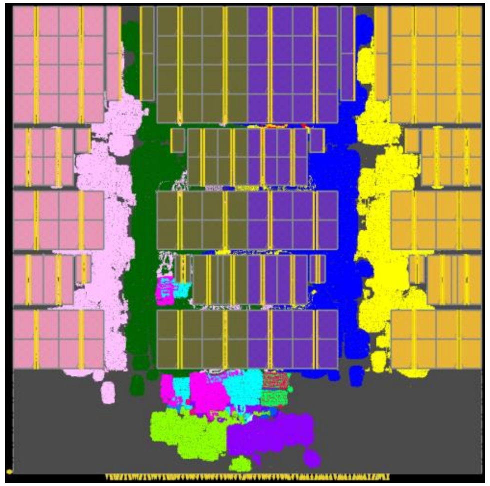
**SA**



**AutoDMP**



**Human**



| Macro Placer | Area ($\mu m^2$) | rWL ($mm$) | Power ($mW$) | WNS ($ps$) | TNS ($ns$) | Proxy Cost |
|---|---|---|---|---|---|---|
| CT | 0.179 | 1.000 | 1.000 | 0.001 | 0.000 | 0.789 |
| CMP | 0.178 | 0.593 | 0.918 | 0.001 | 0.000 | 0.844 |
| RePlAce | 0.178 | 0.798 | 0.959 | 0.000 | 0.000 | 1.121 |
| SA | 0.178 | 0.731 | 0.944 | 0.000 | 0.000 | **0.665** |
| AutoDMP | 0.178 | **0.587** | **0.917** | 0.000 | 0.000 | 0.816 |
| Human | 0.178 | 0.642 | 0.928 | 0.000 | 0.000 | 1.089 |

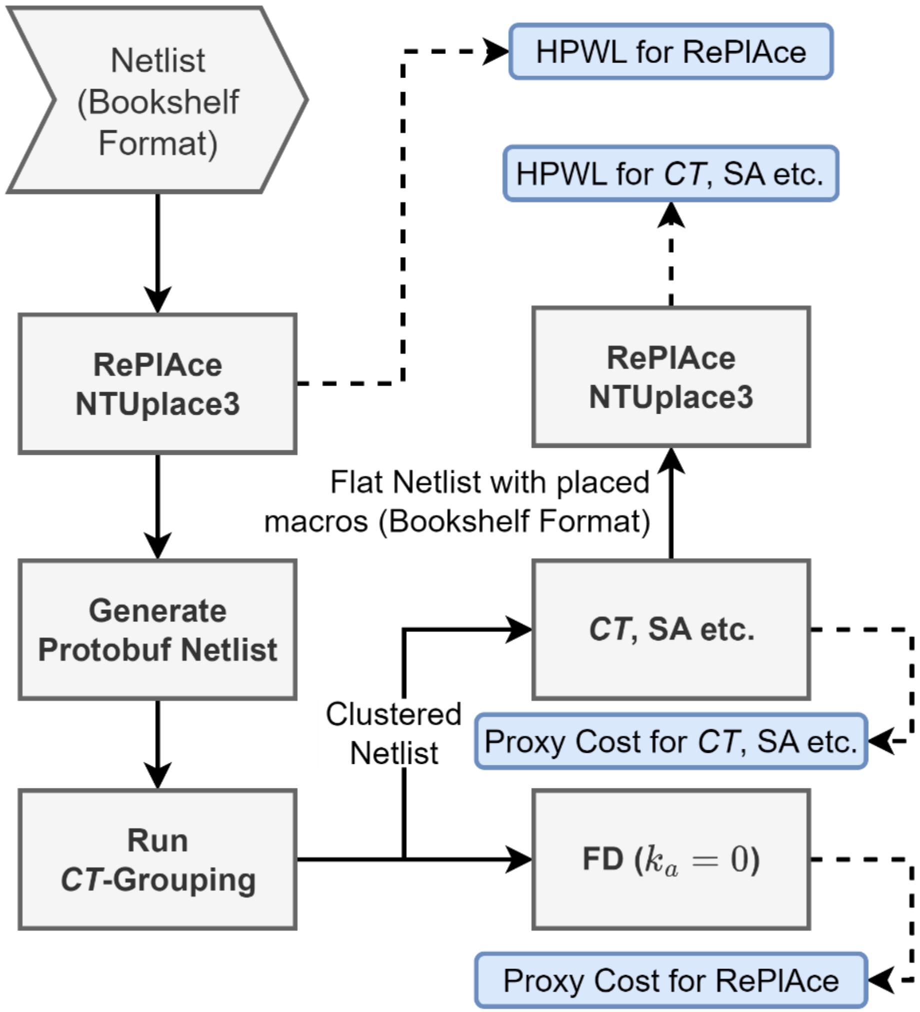
Note: Metric values (excluding proxy cost) are normalized

# Outline

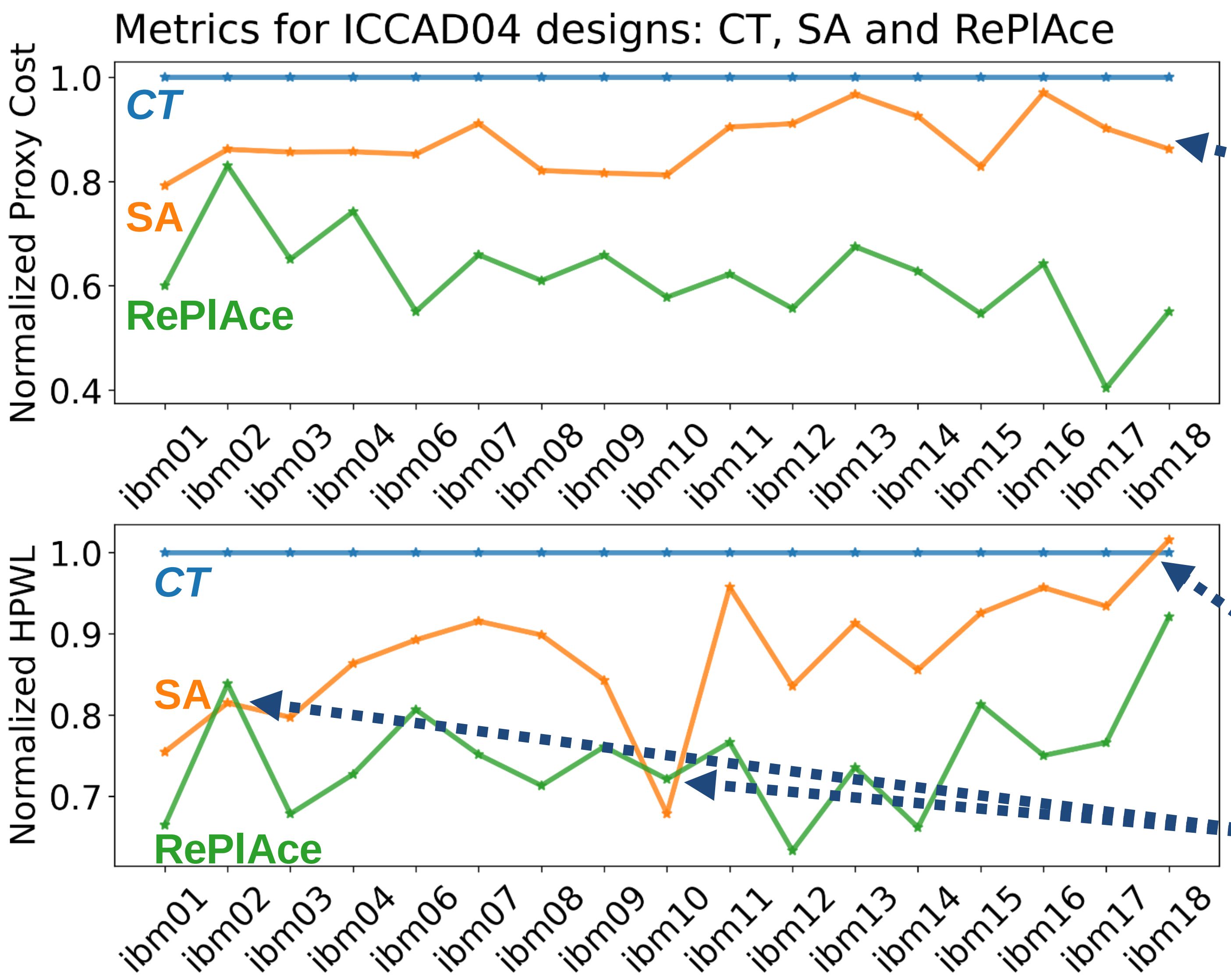
- Background and Motivation
- Replication of Circuit Training (*CT*)
- *MacroPlacement* Repository
- Experimental Results
- Academic Benchmarks
  - Evaluation flow and *CT*, SA and RePlAce results
  - *CT* vs. SA result for different weight combinations in proxy cost
- Conclusion

# Academic Benchmark: Evaluation Flow

- Study on **ICCAD04** mixed-size placement benchmarks includes *CT*, SA, and RePlAce
- **Initial placement**: RePlAce + NTUplace3
  - Used to generate clustered netlist for *CT* and SA
- Standard cell placement of *CT* and SA solutions → RePlace + NTUplace3
- **Evaluation metrics**:
  - Half-perimeter wirelength (HPWL) of placed design
  - Proxy cost reported using plc_client available in *CT*
- More details: **Section 6** of our paper



# *CT*, SA and RePlAce Result of ICCAD04 Benchmarks



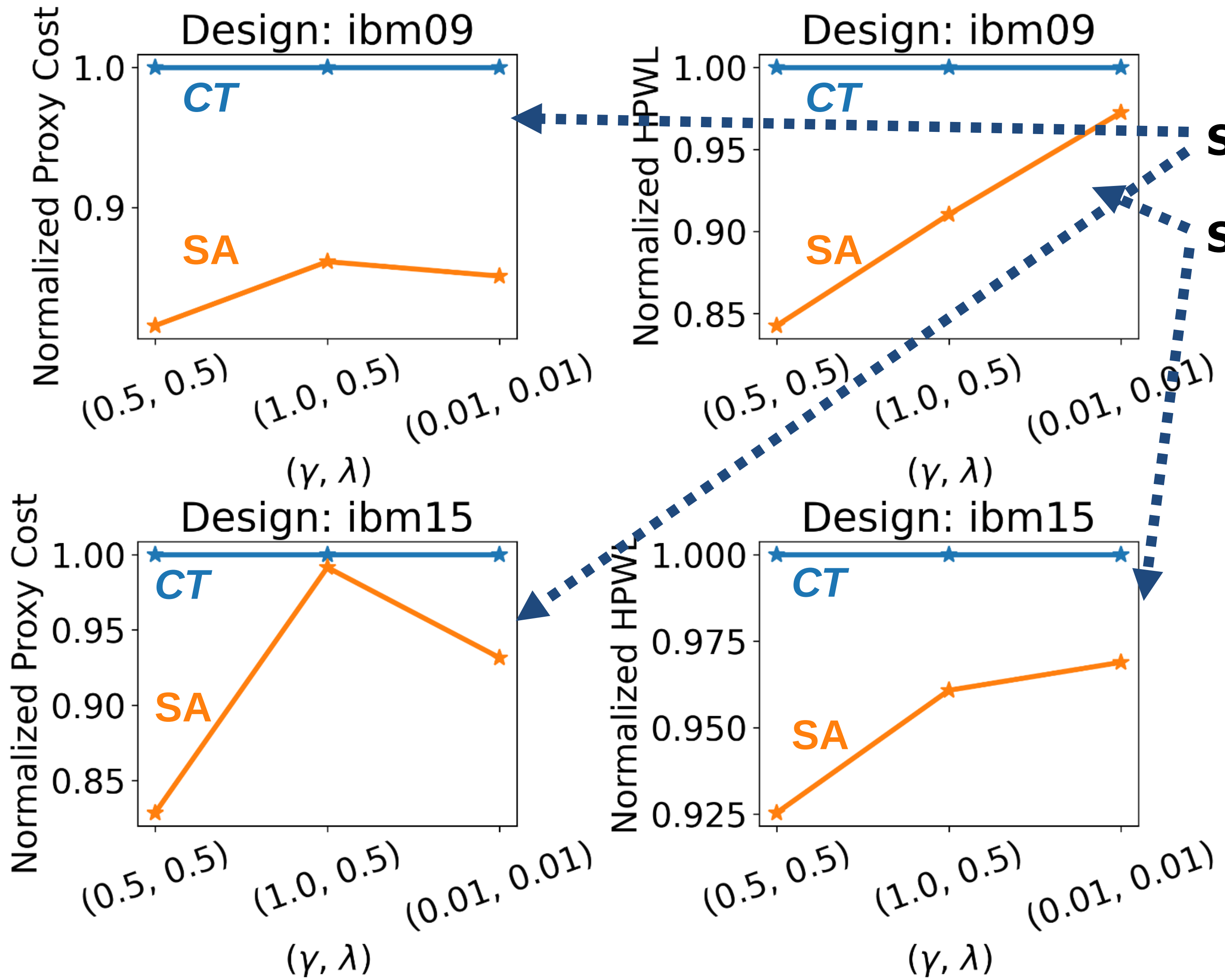

**Proxy Cost: RePlAce beats SA, and SA beats *CT***

- Data is **normalized** based on *CT*
- Table 6 of our paper gives raw data

**HPWL: RePlAce beats SA, and SA beats *CT*, except:**

- **HPWL: *CT* beats SA on 1 of 17 cases**
- **HPWL: SA beats RePlAce on 2 of 17 cases**

# *CT* vs. SA: Different Proxy Cost Weight Combinations




- Normalized data: based on *CT*
- Raw data: see Table 7 of the paper
- γ = Density weight
  λ = Congestion weight

SA produces better proxy cost and HPWL results than *CT*, across different weight combinations, for the above two testcases

# *MacroPlacement* Repo: More Ablation Studies

- How difficult is Ariane? → Shuffling test
- Are *CT* and SA results stable? → Variance test
- What is the correlation of proxy cost with *Nature* Table 1 metrics?
- What is the effect of physical synthesis tool choice on *CT* outcome?
- What is the effect of target clock period on *CT* outcome?
- What is the effect of utilization on *CT* outcome?
- What is the effect of the coordinate descent placer on *CT* outcome?

# Outline

- Background and Motivation
- Replication of Circuit Training (*CT*)
- *MacroPlacement* Repository
- Experimental Results
- Academic Benchmarks
- Conclusion

# Conclusion

- ***Nature* presents a novel orchestration of multiple elements**
  - Proxy cost function that combines wirelength, density and congestion
  - Sequential framework for deep RL-based macro placement
  - Grouping flow to manage instance complexity
  - Data and code (still) unavailable → our *MacroPlacement* assessment effort
- **Baselines (SA and Human experts) outperform *CT***
  - For 17 ICCAD04 designs and 4 out of 6 modern testcases, SA generates better proxy cost than *CT*
  - For large macro-heavy designs, human experts outperform *CT* in terms of *Nature* Table 1 metrics
  - *CT* benefits from placement information in the incoming physical synthesis netlist
- **"There is no substitute for source code (and data)"**

# Runtimes (Wall Times) of Different Macro Placers

| Design | *CT* (Hours) | CMP (Hours) | RePlAce (Hours) | SA (Hours) | AutoDMP (Hours) |
|---|---|---|---|---|---|
| Ariane-NG45 | 32.31 | 0.05 | 0.06 | 12.50 | 0.29 |
| BlackParrot-NG45 | 50.51 | 0.33 | 2.52 | 12.50 | 0.71 |
| MemPool-NG45 | 81.23 | 1.97 | *N.A. | 12.50 | 1.73 |

- *CT*: only includes *CT* training time
- SA: stopped after 12.5 hours automatically
- CMP: only the runtime of ***place_design -concurrent_macros*** command
- Resource required for different macro placers
  - *CT*: Training and evaluation jobs run on (8 NVIDIA-V100 GPU, 96 CPU thread, Memory: 354 GB) machine and 13 collector jobs on each of two (96 CPU thread, Memory: 354 GB) machines
  - SA: 320 parallel jobs where each job used 1 thread
  - RePlAce: used 1 thread
  - CMP: Innovus launched with 8 threads
  - AutoDMP: run on NVIDIA DGX-A100 machine with two GPU workers

*RePlAce run for MemPool Group did not complete

# FAQ

- What do your results tell us about the use of RL in macro placement?
  - The solutions typically produced by human experts and SA are superior to those generated by the RL framework in the majority of cases we tested.

- Did the work by Prof. David Pan show that Google open-source code was sufficient?
  - No. The arXiv paper “Delving into Macro Placement with Reinforcement Learning” was published in September 2021, before the open-sourcing of Circuit Training. To our understanding, the work focused on use of DREAMPlace instead of force-directed placement.

- Did you replicate results from Stronger Baselines?
  - We replicated RePlAce results and believe our SA obtains similar results. However, there is no code or data available to reproduce S.B.’s reported *CT* results, or proxy costs of SA results.

We thank David Junkin, Patrick Haspel, Angela Hwang and their colleagues at Cadence and Synopsys for policy changes that permit our methods and results to be reproducible and sharable in the open, toward advancement of research in the field. We thank many Google engineers (Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Eric Johnson, Roger Carpenter, Sergio Guadarrama, Guanhang Wu, Joe Jiang, Ebrahim Songhori, Young-Joon Lee and Ed Chi) and the TensorFlow Agents team for their time and discussions to clarify aspects of Circuit Training, and to run their internal flow with our data. We thank Ravi Varadarajan for early discussions and flow setup, and Mingyu Woo for guidance on RePlAce versions and setup. Support from NSF CCF-2112665 (TILOS) and DARPA HR0011-18-2-0032 (OpenROAD) is gratefully acknowledged.

**THANK YOU !**