



Agentic AI for EDA: Observations and Outlook

Andrew B. Kahng

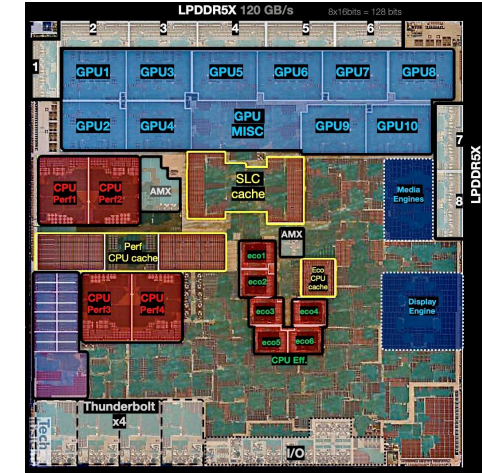
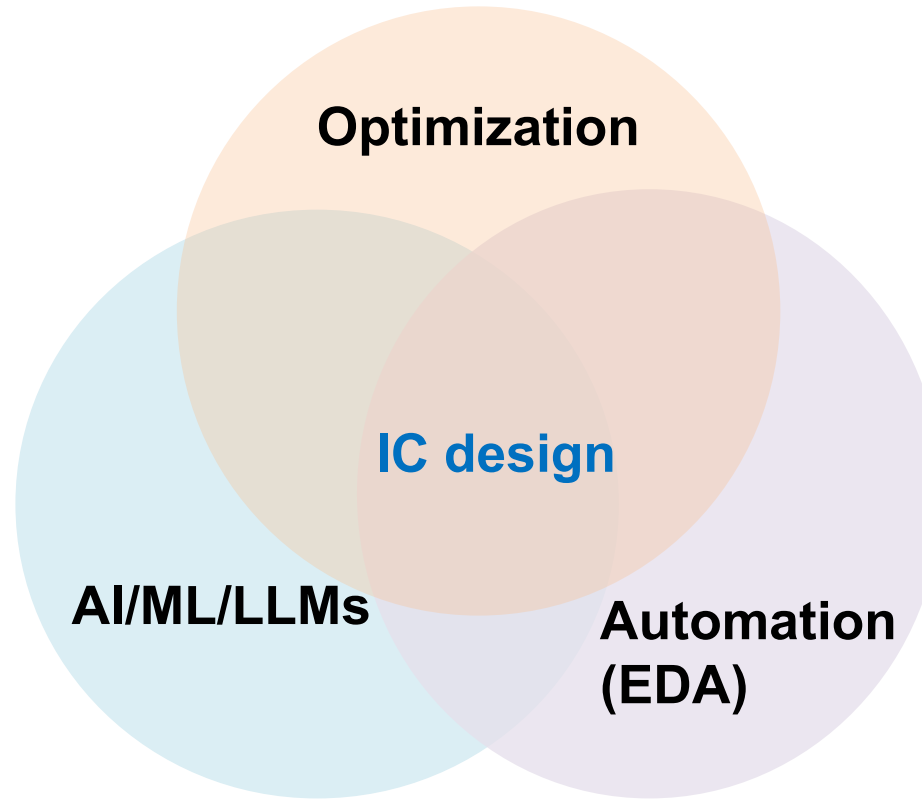
University of California, San Diego

abk@ucsd.edu



The need

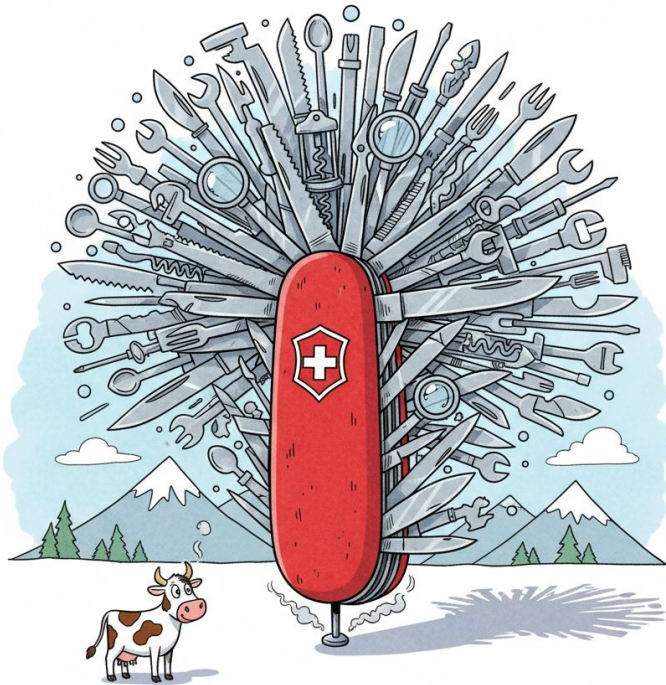
- **IC designers need better design automation tools**
 - Core challenge of IC design: **OPTIMIZATION**





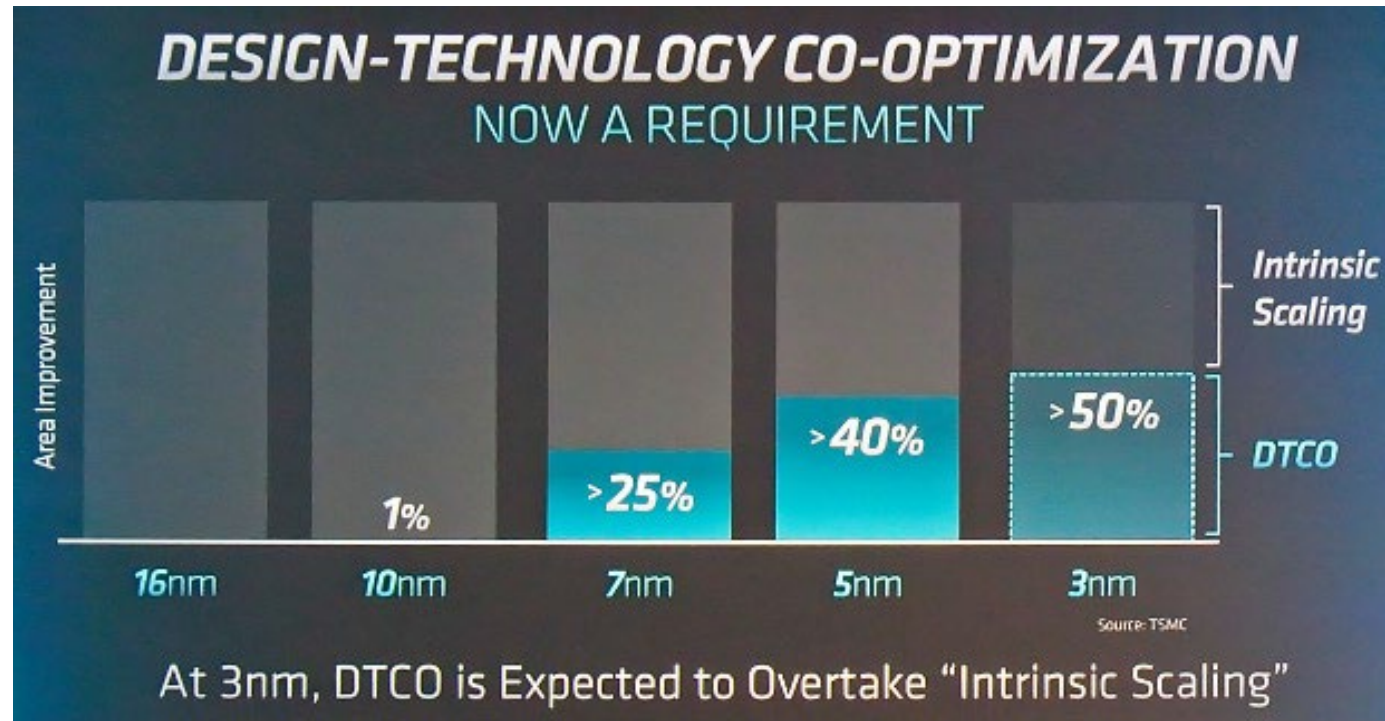
The need

- IC designers need better design automation tools
 - Core challenge of IC design: **OPTIMIZATION**
- **Differentiated design capability means bespoke tools**
 - “10,000 tapeout rules”: But how many of those tapeouts were on YOUR design ?



The need

- IC designers need better design automation tools
 - Core challenge of IC design: **OPTIMIZATION**
- Differentiated design capability means bespoke tools
 - “10,000 tapeout rules”: But how many of those tapeouts were on YOUR design ?
- **Scaling is design-based**



Mark Papermaster, keynote, Design Automation Conference, July 2022

Implications for optimization → implications for AI/ML

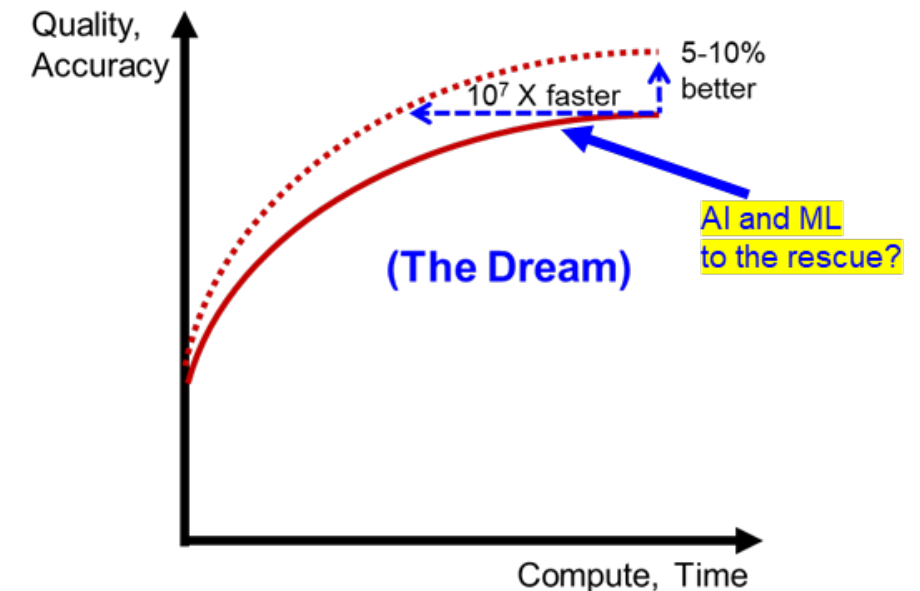
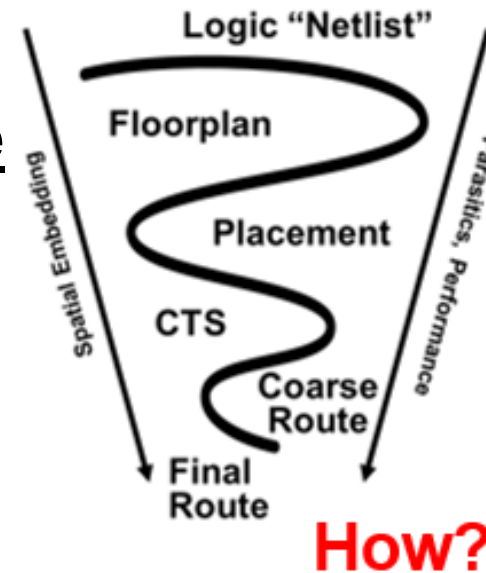
- Predictions today are **Constructive**
 - Quick-and-dirty, under the hood

- Optimizations today are **Iterative**
 - “Construct by Correction”

- **Need Fast Simulations** to guide Optimization
 - “Construct by Corrections ...
... that are **Correct by Construction**”

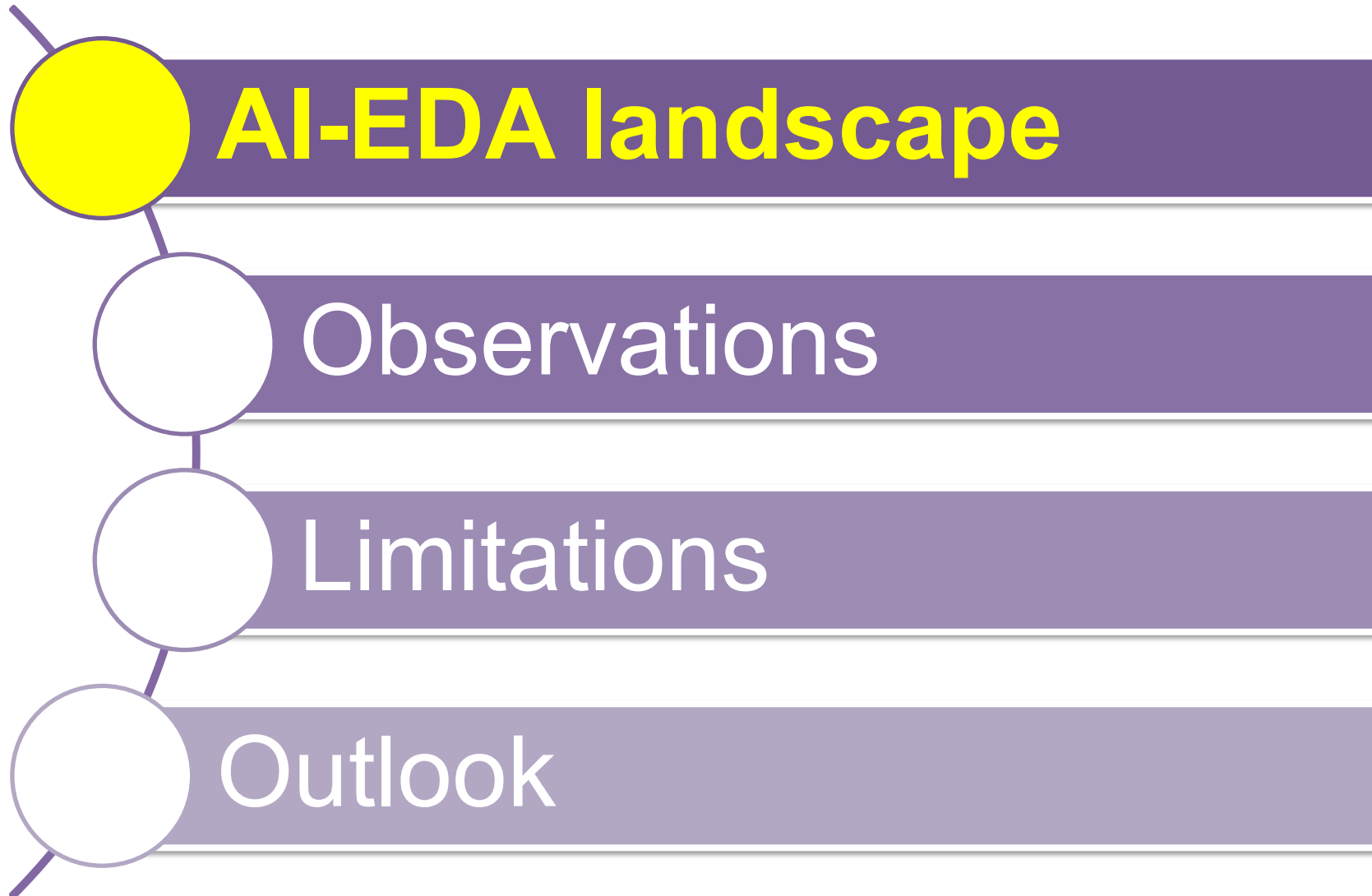
Catechism:

- A. You need **models** to have predictions
- B. You need predictions to **leverage** in exploration
- C. What you can't predict, you guardband
- D. What you don't explore, you leave on the table



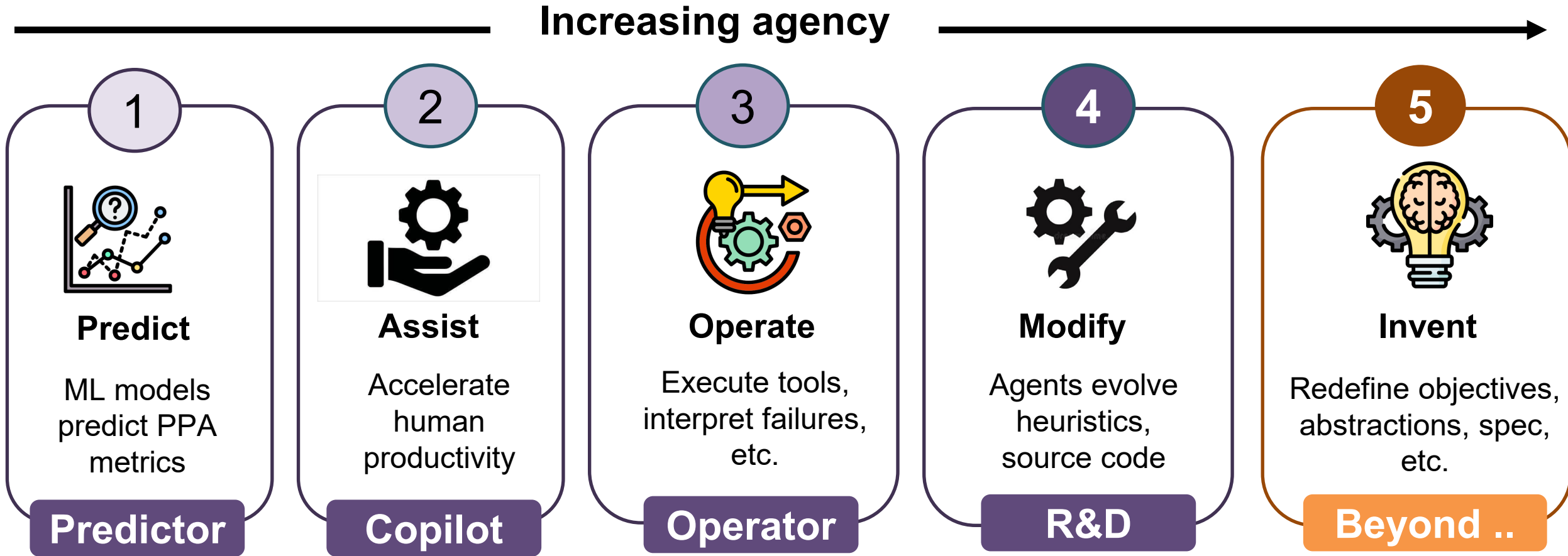


Agenda



Strata in the AI-EDA landscape

- From prediction models → copilots → closed-loop agents → autonomous R&D systems



The agentic lever (*LLMs → Agents → R&D Acceleration*)

- Scaling is **design-based**
- Tools are **bespoke**
- Ecosystem is **consolidated**
- **Optimization imperative**
= do better and faster, with less

LLMs in EDA (2026 ++)

Multi-hours/multi-days reasoning

Comprehend large codebases

Invoke tool commands

Self iteration with feedback

IC design optimization
(PPA, schedule, cost)

LLMs + tool use



EDA = tools
(synthesis, P&R, STA, etc.)

(Human +) Agentic R&D loop
(read logs, tweak params, edit code, check/eval + iterate)

From “EDA **tools**” ➡ to “EDA R&D **collaborators**”



From assistance to autonomy: **two levels**

- **Fact:** Agents (in EDA) will operate across differing levels of **abstraction**
 - **Abstraction** determines **search space**, **feedback signal**, and **risk**
 - *Open-source EDA is the perfect (and our only) testbed !!!*
- **Two levels (~patterns)**
 - **Code-level agents** modify algorithmic structure (source code)
 - ← "Improve the global swap operator in the dpl module of OpenROAD"
 - **Flow-level agents** modify control variables (tool flow, parameters, objectives)
 - ← "Minimize ECP with < 2% area/power/PDP degradation by tuning ORFS configs"
- *Is the agent reasoning about **heuristics** (code and operators) or about **policies** (parameters and sequencing)? (Inevitably, agents will reason about both!)*

OpenROAD as a foundation for agentic EDA research

94% pass-rate vs. 0% for foundational models”

Assistants

OR-Assistant,
OpenROAD-Agent
→ scripts + user
queries



OpenROAD

Open code
Open APIs
Open benchmarks

Code-level

GR-evolve → grt

HELIX → dpl, gpl

AuDoPEDA →
dpl, gpl, rsz

Up to 9% HPWL, 23% RWL reduction
across 2 PDKs

Flow-level

ORFS-Agent
→ flow knobs

“~13% WL/ECP gains, 40%
fewer iterations”

Three frontiers of recent progress in OpenROAD

Flow tuners

- Treat EDA tools as **black-boxes**; **tune** parameters and orchestrate flows
- **ORFS-Agent** (UCSD)
OpenROAD-Agent (ASU)
ORAssistant (PII)
- **Algorithms** **unchanged**; agents learn the policy

“~40% fewer iterations than autotuners”



Repo-scale coding

- Agents can **read**, **edit**, **compile** and **benchmark** across **multi-language** **codebases**
- **AuDoPEDA** (UCSD)
- **Algorithms** **change**; formal verification; closed-loop guardrails

“~19% power gains”



Bespoke, design-adaptive tools

- **Per-design** code evolution; tool **specializes** to the workload
- **HELIX** (UCSD)
GR-Evolve (ASU)
- **Algorithms** **change**; **one tool per design** vs. one tool for all designs

“~9% HPWL gains”



Repository-grounded coding agents for OpenROAD

Repo graph + **documentation**

Literature-grounded planning

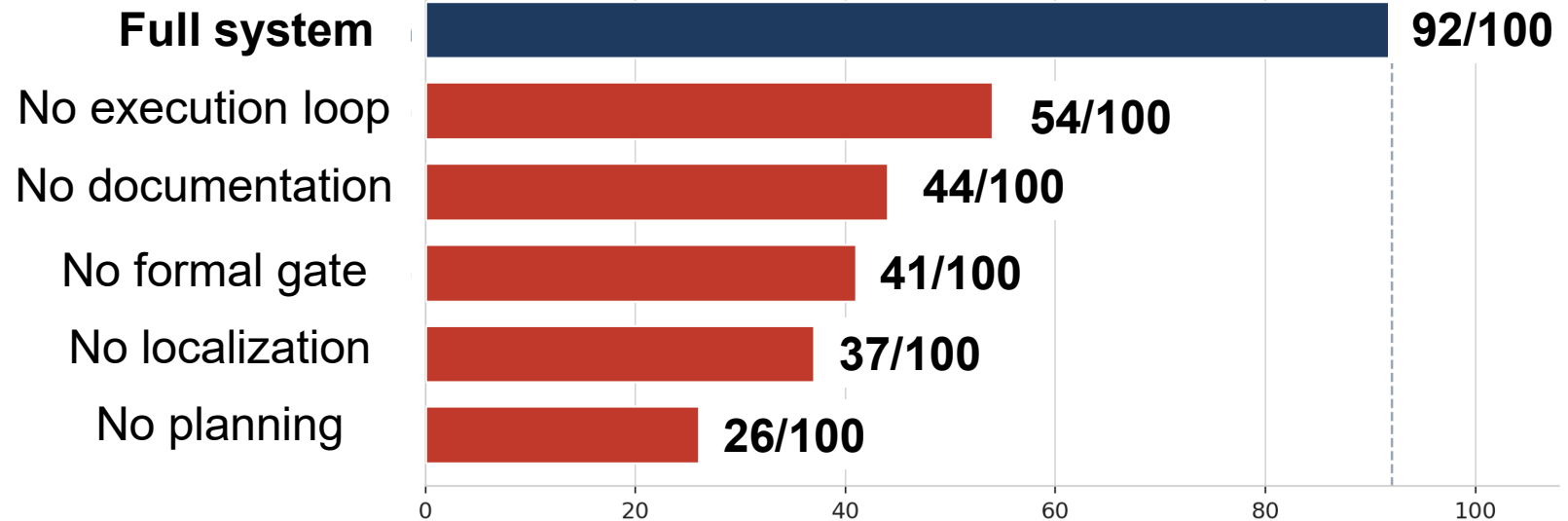
Plan **localization**
(map plan to edit surfaces)

Autonomous execution loop
(apply diffs, build, run flow,
measure QoR)

Formal diff checking
(Lean; Aristotle; AXLE)

Target	Module(s)	Best Δ	Diff scope
Routed wirelength	dpl	-5.4%	11 files, ~1k LoC
Effective clock period	gpl, rsz	-10.0%	11 files, ~330 LoC
Total power	rsz	-19.4%	9 files, ~1k Loc

Removing any single stage dramatically reduces valid diff-rate!



<https://arxiv.org/abs/2601.06268>

Valid diffs out of 100 trials

A merge-ready OpenROAD PR (*agent-driven*)

What “merge-ready” requires

- Passed regression/PPA guardrails
- Iterated on review feedback + merge conflicts
- Auto-generated docs + configs + PR descriptions
- Only manual step: human-expert sign-off
- End-to-end LLM cost: ~\$30
- Human-edits: 0 (except sign-off)

DPL Two-Pass with toggle #9038 **PR number**

Status of the PR: merged (!)

Agent-generated

Human-expert sign-off

The following content was AI generated.

Two-Pass Congestion-Aware Global Swap (DPL) — Optional “Extra DPL” Path

Summary

This PR adds an optional alternative implementation of the `gs` (global swap) phase used by `improve_placement` (DPO / detailed placement optimization). The new path is designed to help high core-utilization designs where routability-driven metrics (e.g., DRWL) become limiting.

Expected outcome (high-util targets): ~2-3% DRWL reduction near the highest utilization where the design still completes the full flow; outliers up to ~8% were observed.

Motivation / When It Helps

At high utilization (typically ~85%+ and/or “near max util that still converges”), a pure HPWL-driven global swap can “clump” cells into locally congested regions. This often harms routability and increases DRWL. The new algorithm explicitly trades bounded HPWL for congestion relief.

Reviews: `github-actions[bot]`, `maliberty`

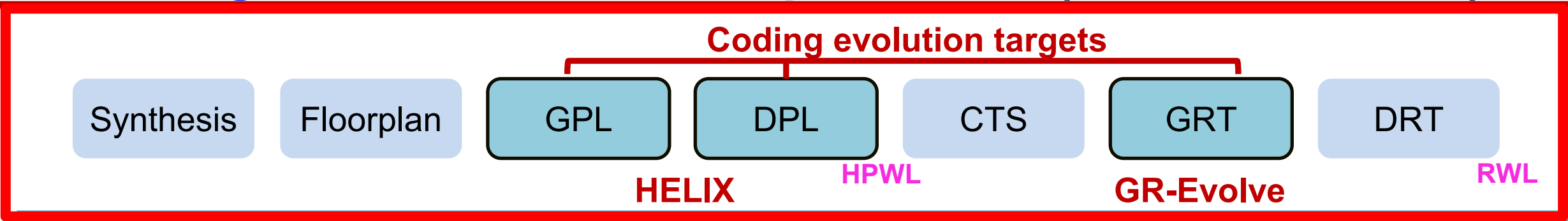
Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

<https://github.com/The-OpenROAD-Project/OpenROAD/pull/9038>

Per-design tool evolution in OpenROAD (UCSD and ASU)

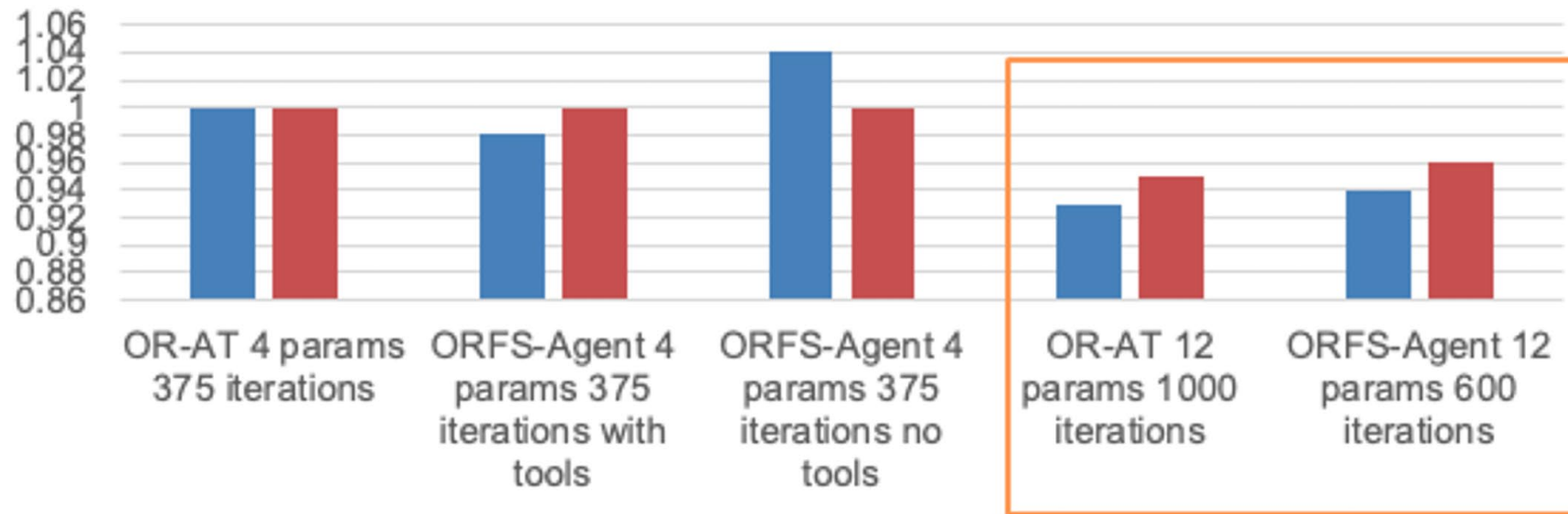


Attributes	HELIX (placement) – UCSD	GR-Evolve (routing) – ASU
Target	Global + detailed placement (gpl , dpl)	Global routing (grt)
Method	(μ + λ) evolutionary search with literature-guided operators	Stateless agentic loop with persistent QoR history
Knowledge base	97 papers + documentation + pseudocode + parameter list	Distilled router summaries + source
Key results	9% HPWL and 23% RWL (max) 4.5% / 5.6% geomean	8.7% RWL (max)

ORFS-agent – LLM-driven flow autotuning

<https://arxiv.org/abs/2506.08332>

- **Design flow tuning is high-dimensional**
 - 1000s (actually 10000s) of knobs in RTL-to-GDS
- **Plain Bayesian Optimization struggles** – weak domain priors; poor high-D scaling
- **Key idea: LLM-driven flow autotuner → ORFS-agent**
 - Read logs; reason in context; interpret bottlenecks; propose structured parameter updates; iterate with tool feedback



Baseline: OR-AT
(4 vars, 375 iters) $\equiv$ 1.0

ORFS-agent uses ~40% fewer iterations, iso-QOR

Also: $\approx$ 13% gains in WL or ECP (single-objective)
(details: [MLCAD25 paper](#))

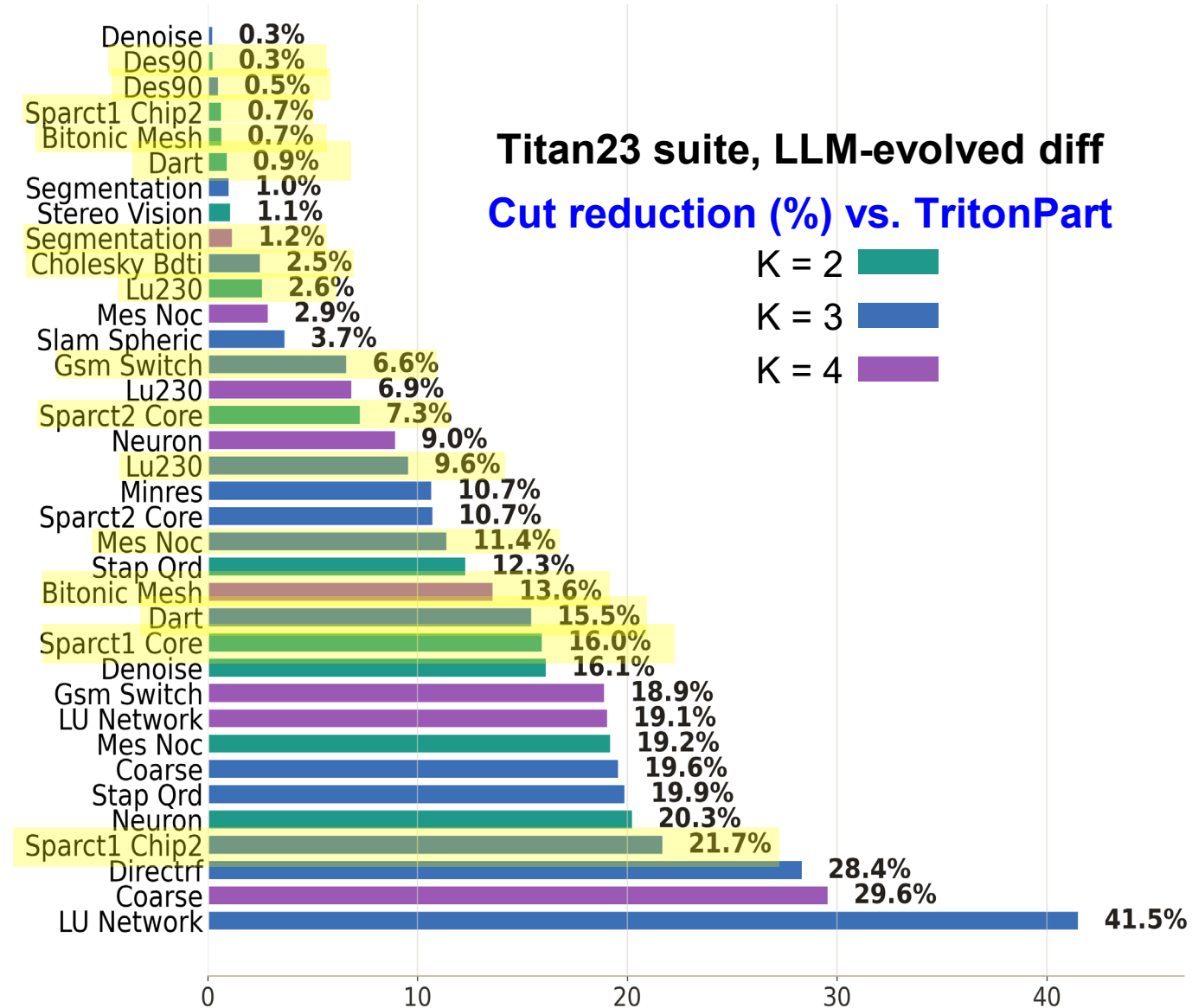
Comparison of ORFS-agent and OR-AutoTuner w.r.t. **wirelength** and **ECP**

Normalization: Results with OR-AT4 params and 375 iterations set as 1.0

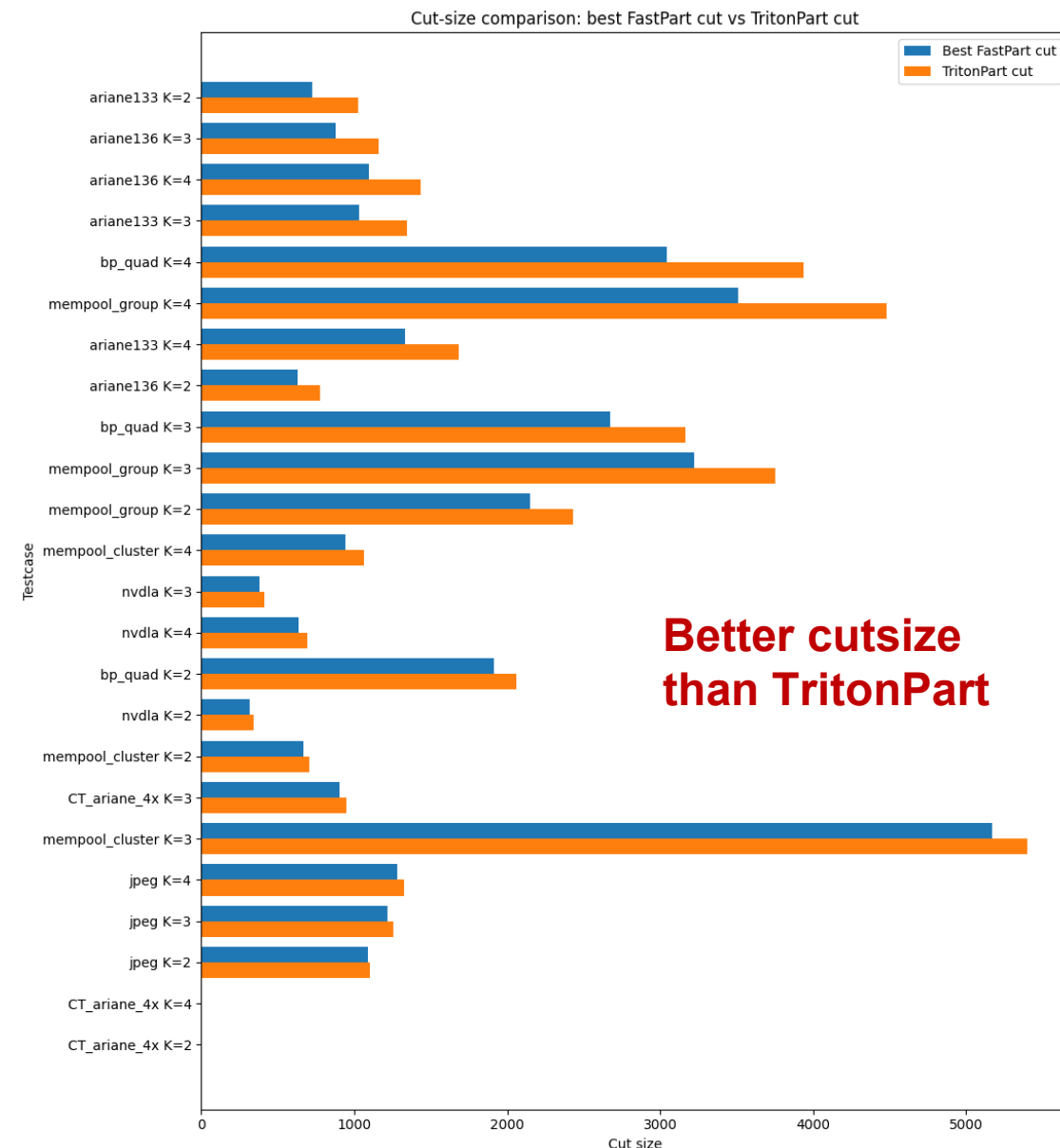
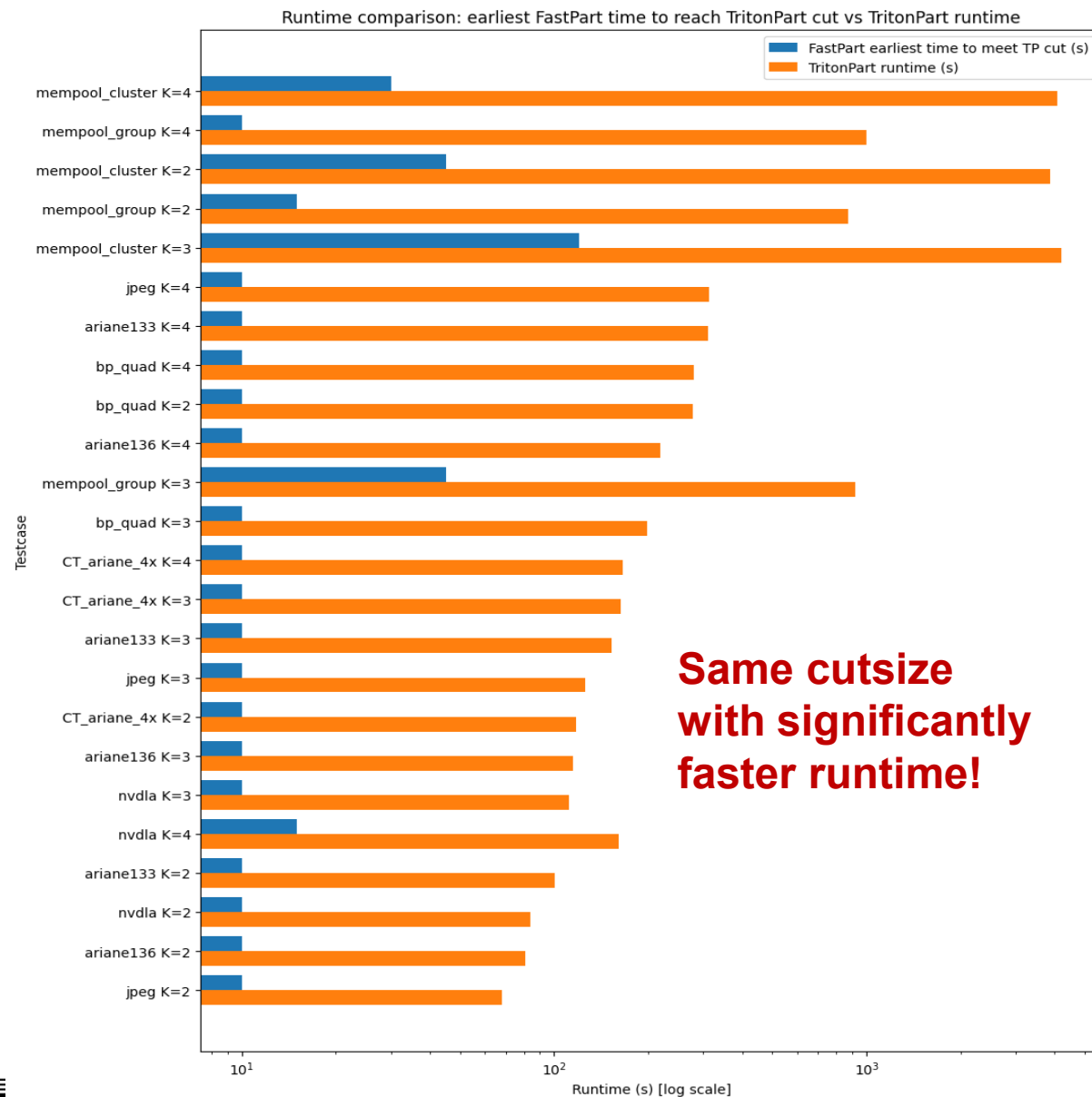


Heuristic-space exploration – min-cut hypergraph partitioning

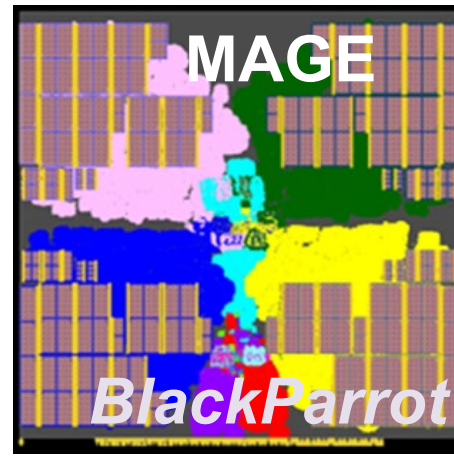
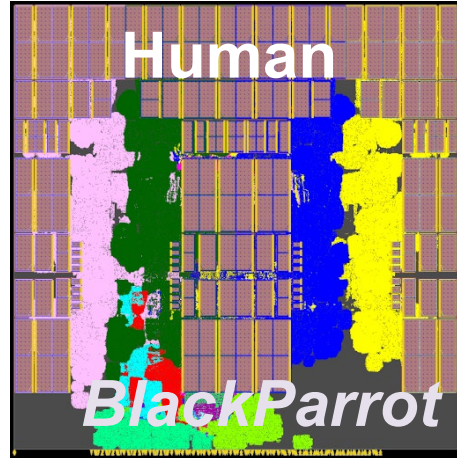
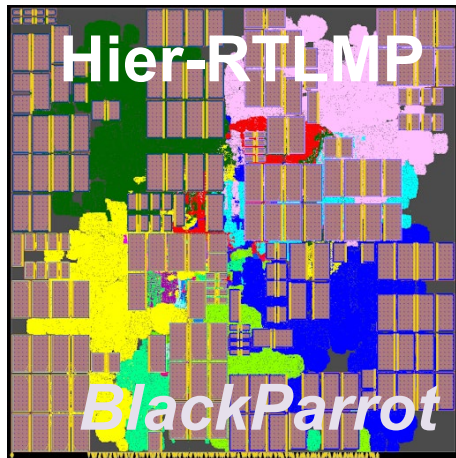
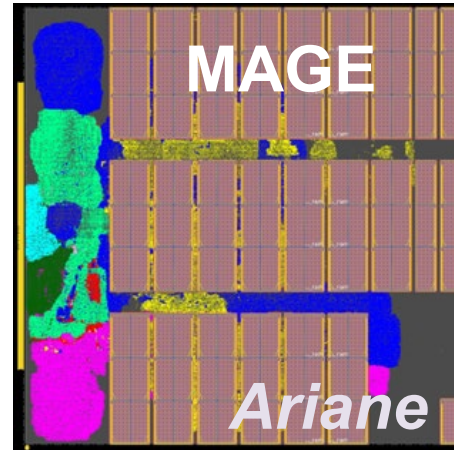
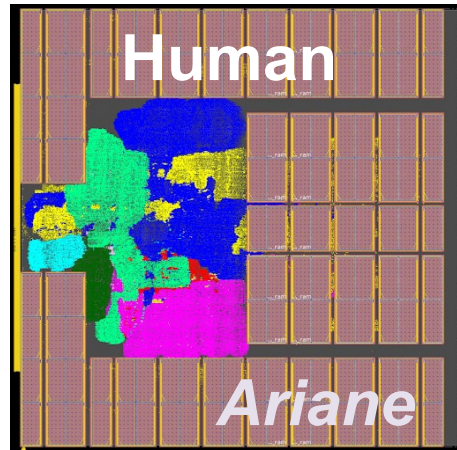
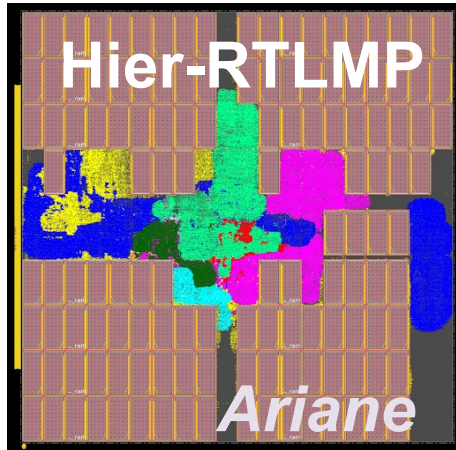
- TritonPart = SOTA hypergraph partitioner (> hMETIS, KaHyPar)
- Has ILP “boundary optimization” step (with CPLEX or OR-Tools)
- **Autonomous AI diff “lowers the variance” of solver step**
- Substantial wins over TritonPart
- Even with the weaker OR-Tools solver, finds **world’s best-ever** solutions!



Hypergraph partitioning – beyond quality: speed



Agentic (**human-like**) macro placement leveraging vision



MAGE-generated macro placements show structure similar to Human

MAGE-generated macro placements

- **MAGE** = agentic human-like macro placer
- **Automates** the human floorplanner's role
- **Vision-in-the-loop**
- Follows a **large set** of human macro placement principles: boundary placement, macro stacking, etc.
- **Better PPA** than commercial and open baselines

(Don't forget the optimization lever !)

- **Partcl/HRT Macro Placement Challenge**

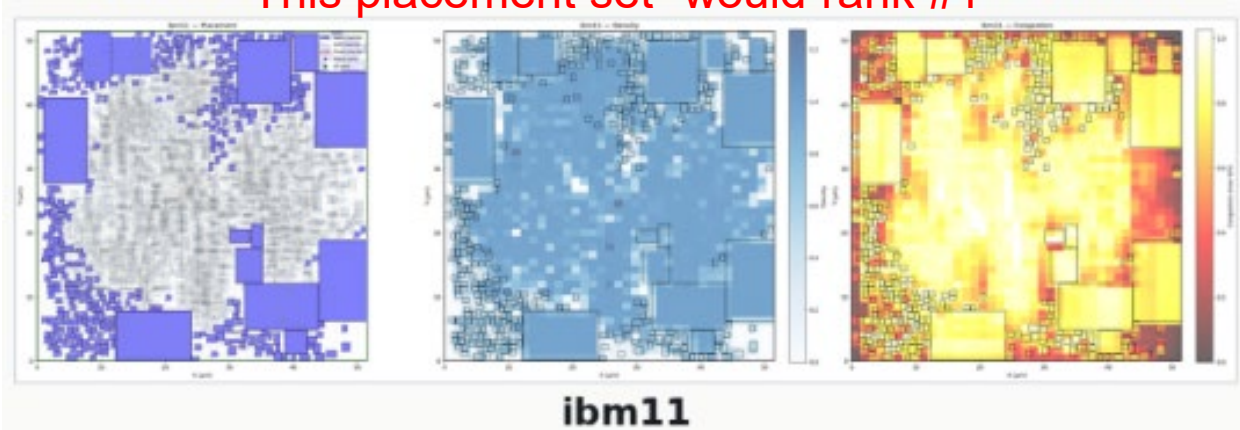
- <https://github.com/partcleda/macro-place-challenge-2026>
- Ended May 21 with 120 teams
- \$20K prize money
- Results changing online ...
but “This placement set” is still #1

As of
June 22:

Rank	Team	Avg Proxy Cost
1	"Archgen"	0.9507
2	"Carrotato"	0.9522
3	"JaneRT"	0.9694
4	"Shoom"	0.9842
5	"Klein-4"	0.9846
6	"tobias-x"	0.9884

Rank	Entry	Avg. proxy
1	This placement set	0.9473
2	Klein-4	0.9846
3	tobias-x	0.9884
4	Vibe	0.9939
5	Macro Polo	0.9965
6	thinkorplace	0.9974
7	JaneRT	0.9978
8	Carrotato	1.0103

“This placement set” would rank #1

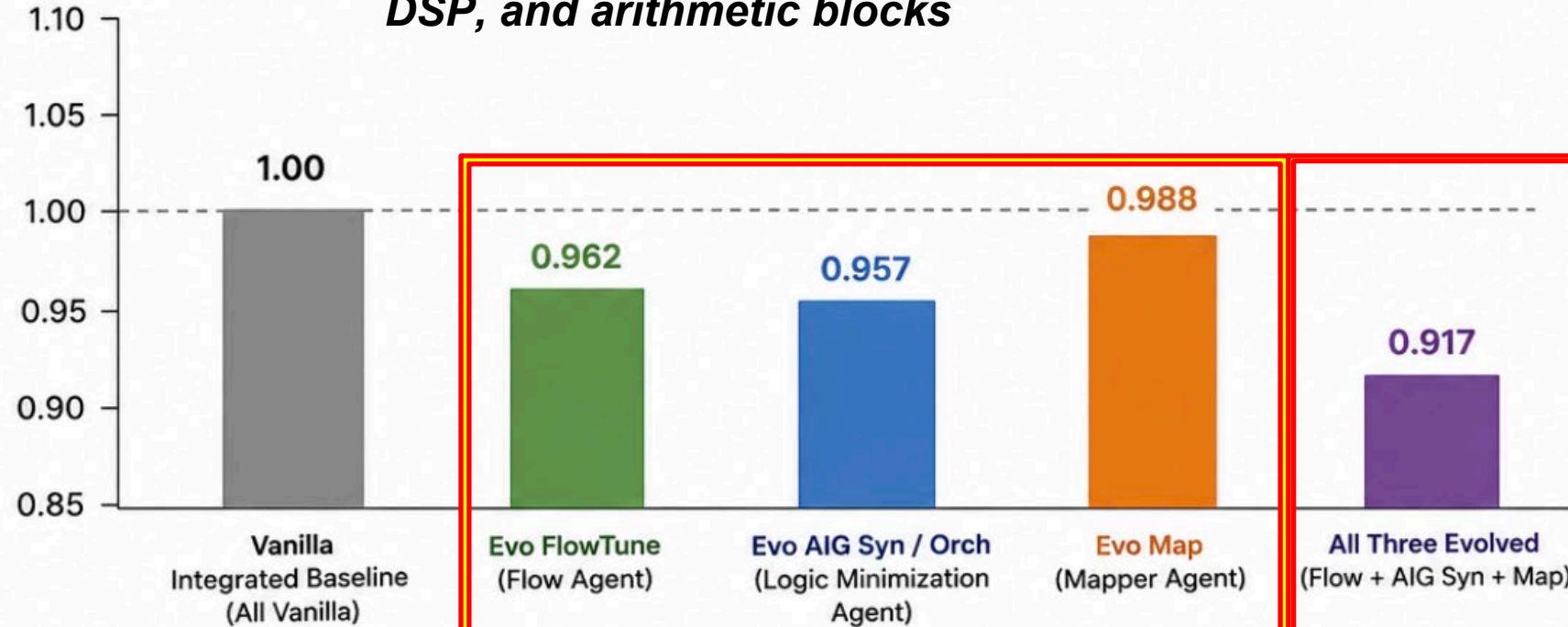


Co-evolution of ABC by NVIDIA

C. Yu and H. Ren, <https://arxiv.org/pdf/2604.15082>

Normalized QoR
(Area-Delay Product)
Lower is Better

*Evaluated across ISCAS'85/'89/'99, EPFL, VTR,
DSP, and arithmetic blocks*



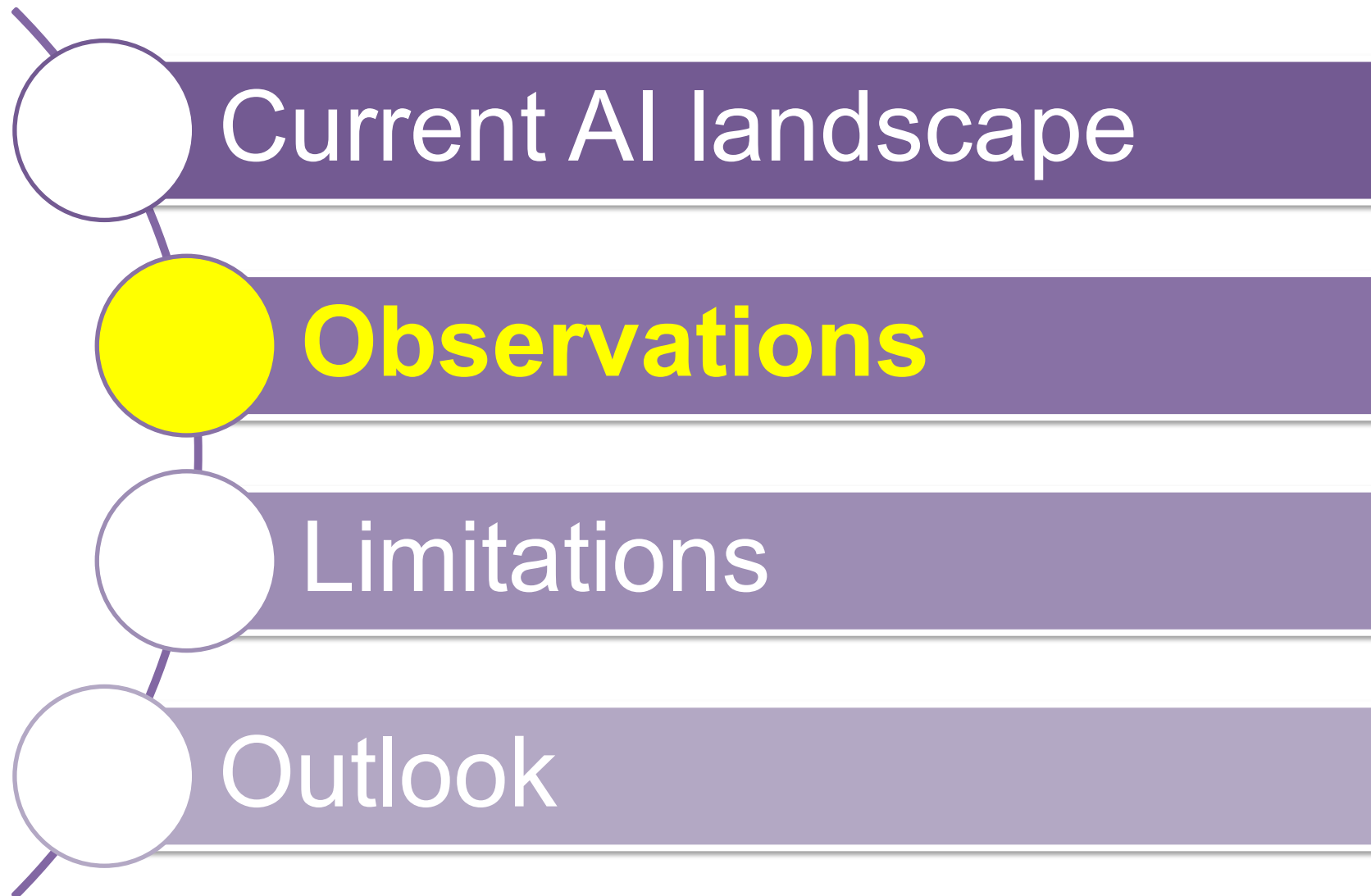
Normalized QoR
(area-delay product)

Vanilla integrated baseline	1.00
Evo FlowTune	0.962
Evo AIG Syn	0.957
Evo Map	0.988
All three evolved	0.917

- Single agent evolution delivers modest gains (~1-4%)
- Pairwise co-evolution generates stronger improvements (~6-8%)
- Joint evolution of all subsystems yields the most improvement (~8.3%)



Agenda



1. Value and ROI – the missing denominator



OpenROAD
PR #9038 (agentic)

- ~\$30 end-to-end
- 0 human edits

\$30

??? (engineer hours; license hours)

- **We show the cost, but NOT what the agent replaced / saved**
- ROI demands a benefit metric **that can be measured**
- Denominator monetized today by EDA incumbents: (license cost + compute saved)
- **Cadence shifting to pricing based on productivity gain**

Cadence Unveils Industry's First Fully Autonomous Virtual Engineer for Chip Design, powered by NVIDIA

When agents do the heavy lifting, what exactly will be sold -- and what will be bought?

2. Human vs. agents – generation vs. accountability

Agent

explores + generates

- *Proposes diffs*
- *Searches heuristics and operators*
- *Tunes knobs, runs trials*
- ***Performs exploration***

Human

judges + is accountable

- *Defines objectives and trade-offs*
- *Sets guardrails and gates*
- *Reviews, signs off*
- ***Owns the risk and the result***

AI ENGINEERING / AI STRATEGY / OPEN SOURCE

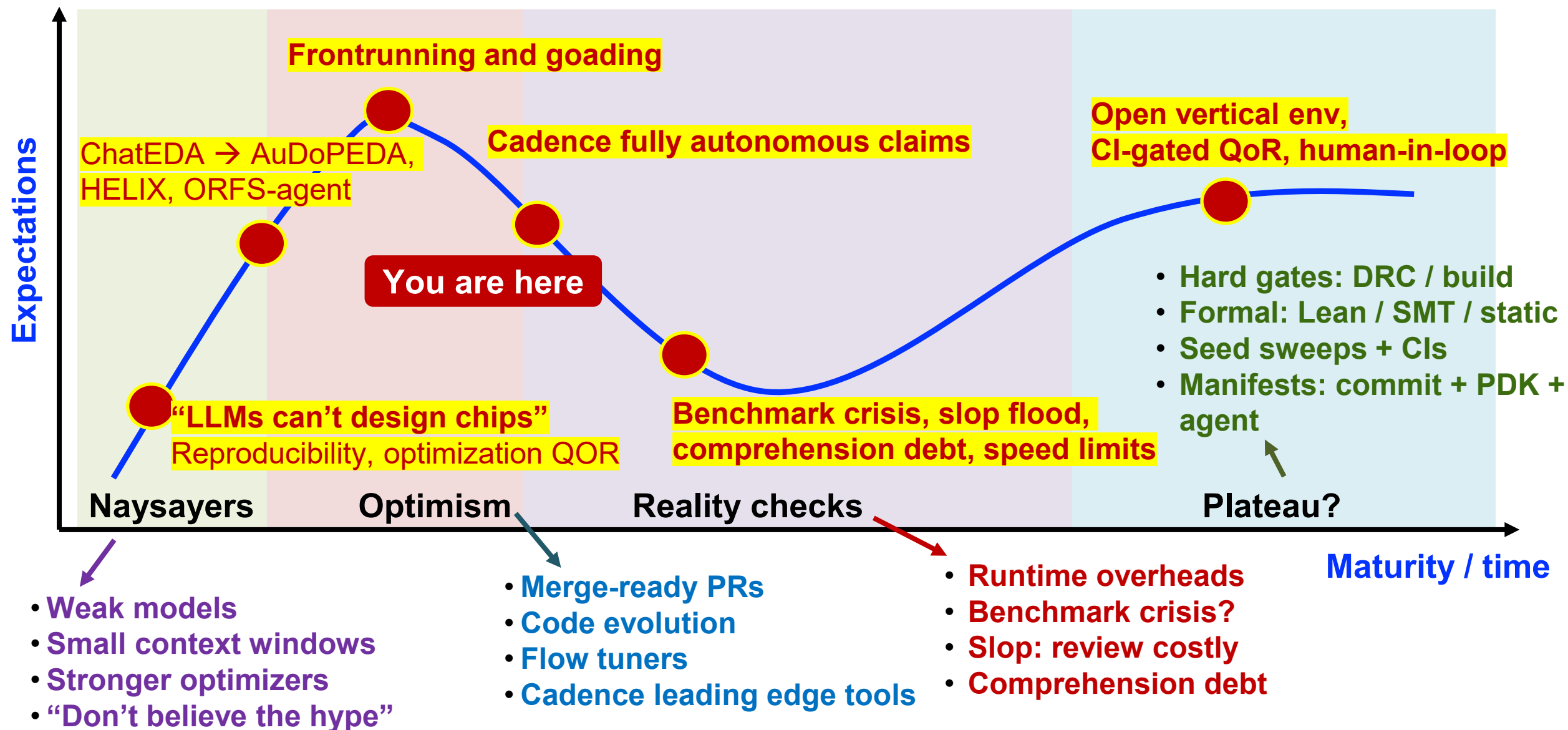
Open source maintainers are drowning in AI-generated pull requests. Enterprise teams are next.

AI is flooding open source with low-quality PRs. Learn how enterprise teams can avoid burnout by fixing the code validation bottleneck.

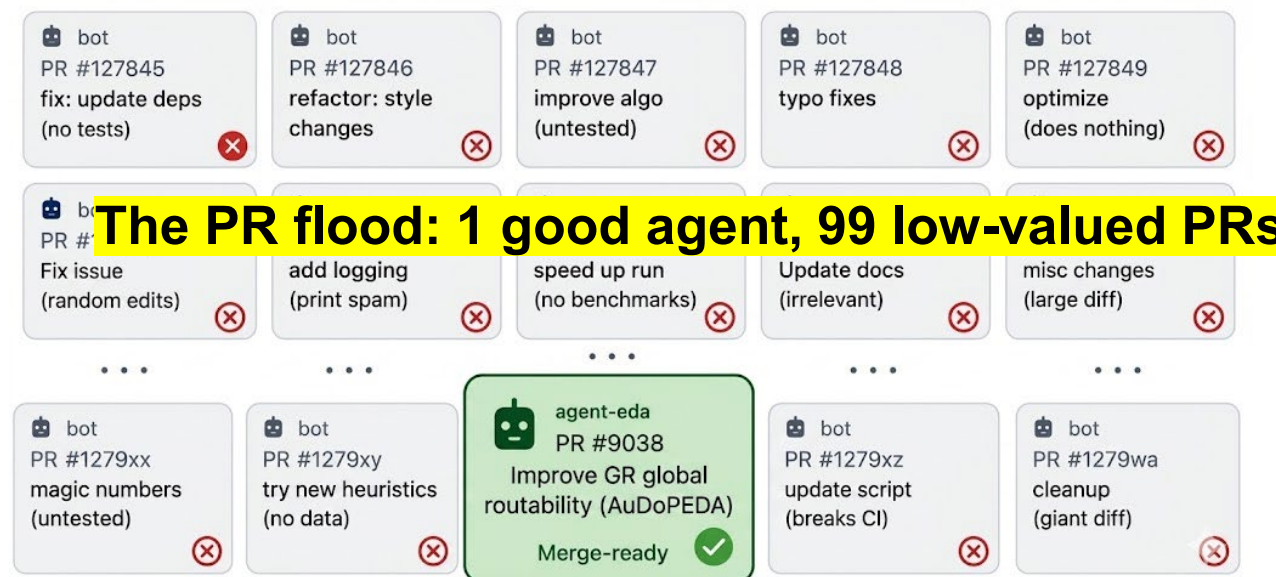
Apr 9th, 2026 11:00am by [Arjun Iyer](#)

- Accountability lands on the human, for whatever the agent explores
- **Risk:** deskilling → **dependence** on an opaque oracle
- **Comprehension debt:** developers were **19% slower** when using AI tools ← developers understand less of AI-accumulated code over time

3. Agentic EDA mapped to the hype cycle



4. Governance and the “slop” flood



EDA governance: mechanical guardrails that scale

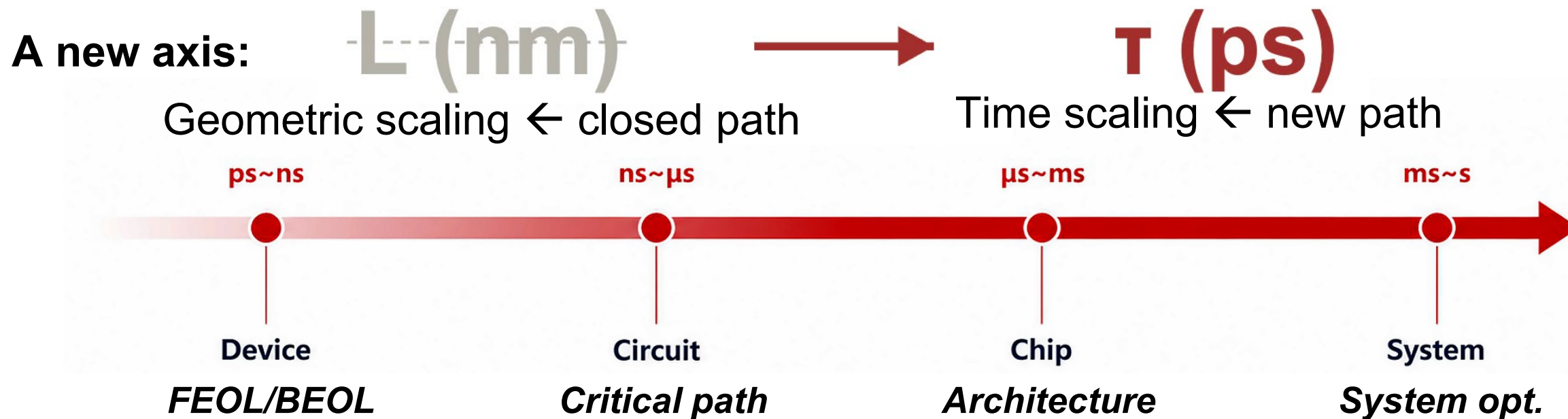
- CI enforced QoR gates
- Provenance for every diff
- Versioned manifests
- Human-in-the-loop control

Build the quality gate first – and THEN open the floodgates

- More PRs/day from bots vs. humans → low-value PRs (“slop”) → longer review latency → good PRs miss maintainer attention
- Why this hurts EDA more?
 - QoR claims without proof (“how do we reproduce that timing number on Ariane”)
 - Expensive validation (full RTL-to-GDS flow) ← bad PRs burn compute and human time
 - One bad merge can break flows, invalidate legacy data, etc. ← regressions cost a LOT

Also: Agency over the problem ...

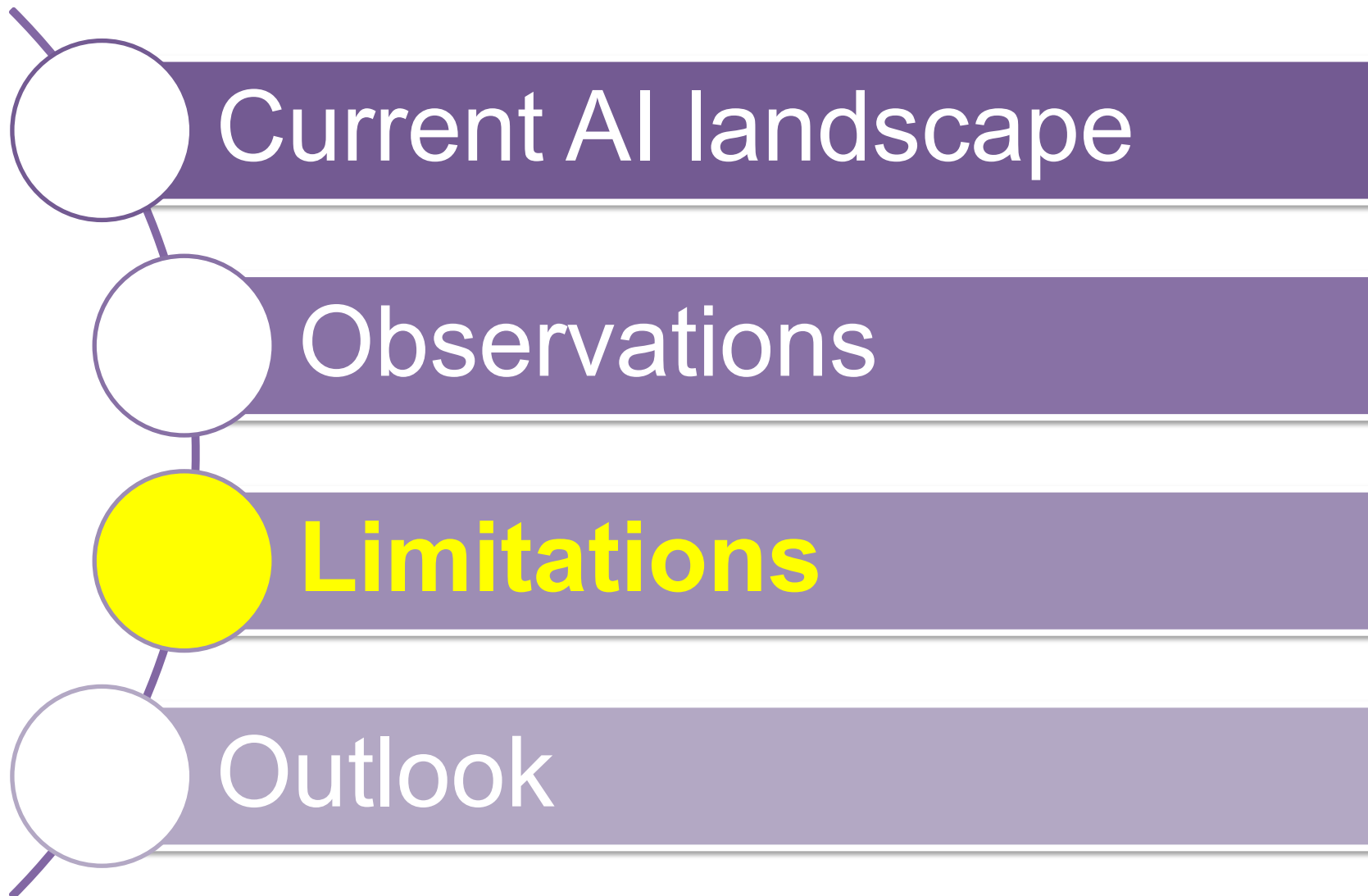
- “Tau Scaling” can be seen as an example of “problem-level agency” (!!!)
- ISCAS26 keynote: [[YouTube](#)] [[paper](#)]



- EDA problem statements and methodology have been at best “inertial”
- What new problems will agentic AI pose for EDA (and agentic R&D engineers)?



Agenda



Gaps in current agentic EDA

Brittle

Reliability / correctness

- Non-determinism
- What is “correct”?
- Verifiable guarantees

Gamed

Meaningful benchmarks

- Better proxies are needed
- Static leaderboards
- Retractions + harness swings

Ad-hoc

Human-in-the-loop

- Review can not scale to machine-speed PRs
- Sign-off and accountability undefined
- Comprehension debt

Afterthought

Security / privacy / IP

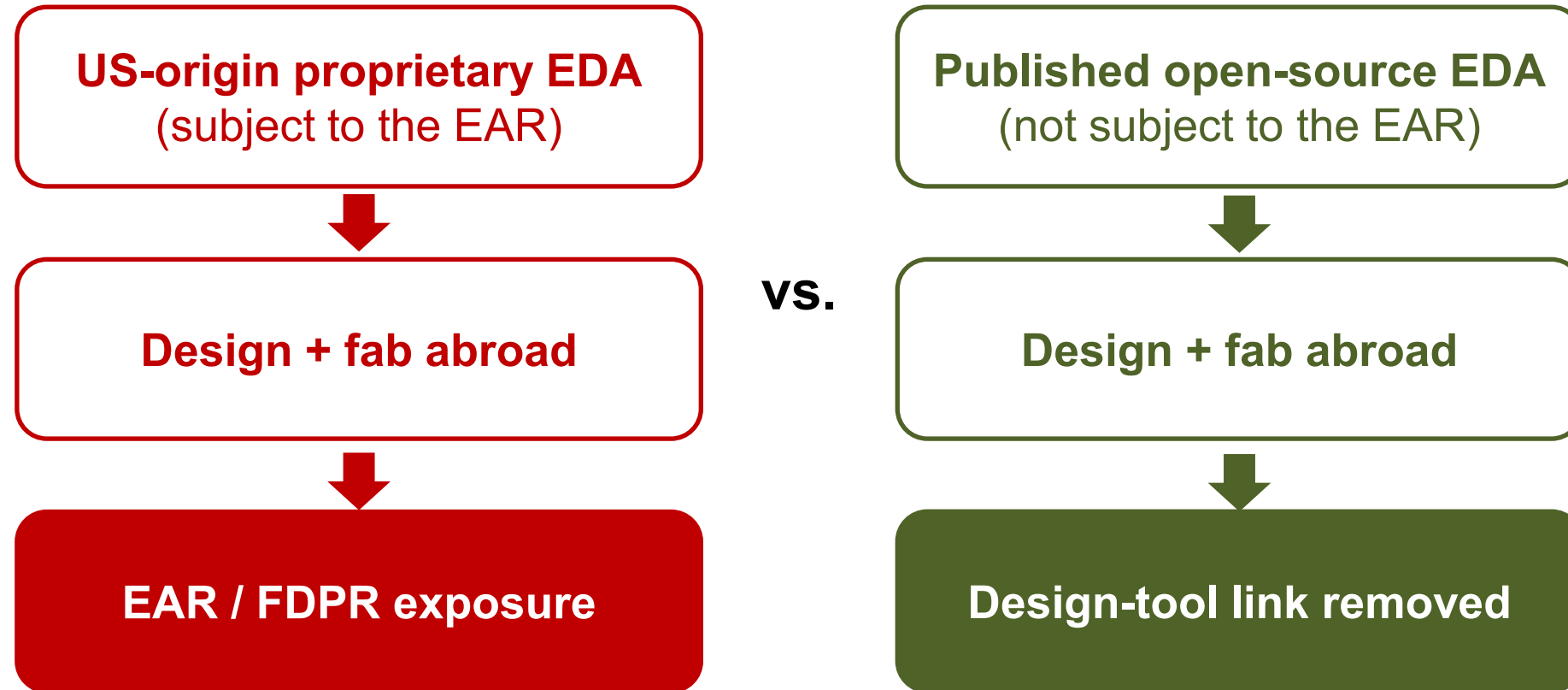
- Agent conceived → no inventor
- Agents and tools are export-gateable
- Slop flood + data-leakage / supply-chain risk

A. Ghose, A. B. Kahng and B. Pramanik, “Special Research Session: Progress and Roadmap Toward an Automated EDA R&D Engineer”, Proc. DAC’26 (upcoming)



Sovereignty – the EDA toolchain is a jurisdiction

- **Export-control jurisdiction follows the tools your workflow used**
← published / open-source avoids this – 2 layers (agentic + EDA tool)



- **Both layers gate you:** the agent (frontier model suspended) and the tools (proprietary EDA = FDPR link) ← **open EDA + open weights = stack you can own**



Challenge 1: validation & evaluation

- **Agents are non-deterministic; flows are slow and expensive**
 - Reproducibility & traceability – same prompt, different code
 - Cost: minutes to hours per RTL-to-GDS candidate
 - Improvements at GR / placement may only show after detailed routing
- **What “correct” means is no longer obvious**
 - DRC-clean is necessary; not sufficient
 - Multi-objective: WL, timing, power, area, runtime
 - Generalization across designs and PDKs
- **What we need**
 - **Standardized testbeds**: ORFS, MacroPlacement, etc.
 - **Hard gates**: DRC, build, regression tests – **reject-before-scoring mechanisms**
 - **Formal contracts where possible**: Lean / SMT / static analysis
 - **Statistical rigor**: seed sweeps, confidence intervals, ablations



Challenge 2: humans + agents; IP & creators' rights

- **Where does the human belong in the loop?**

- Objective definition, trade-offs, safety -- not API debugging
- Agents explore; humans judge and are accountable
- **Risk:** **deskilling** if agents become opaque oracle “experts”

- **IP and creators' rights are murky** *(in the ‘agentic future’ ...)*

- **Who owns an LLM-evolved diff?** Model? Prompter? Repo? *(If it reads on a patent claim?)*
- Permissively-licensed code bases might accept agent diffs *(is usability clouded?)*
- **Training** with public “scraped” corpora *(creator rights, opt-ins, ... ?)*
- **Attribution and credit** when an agent rediscovers known techniques

- ***What does “authorship” mean for agent-generated EDA code?***



Challenge 3: governance – from project to ecosystem

- **Agents change the contribution model**

- PRs at machine speed: **how does review scale?**
- Bot accounts, auto-generated commits → **what counts as a maintainer signal?**
- **Risks:** hallucinated, or subtly-wrong code overwhelming reviewers

- **Ecosystem-level questions**

- **Reproducibility:** pin commits, PDKs, agents, prompts
- **Saturated benchmarks:** need open, evolving evaluation suites
- **Compute access:** who can run repo-scale evolution? ← **equity in research**
- **Co-evolution:** agent capabilities and tool APIs both **moving targets**

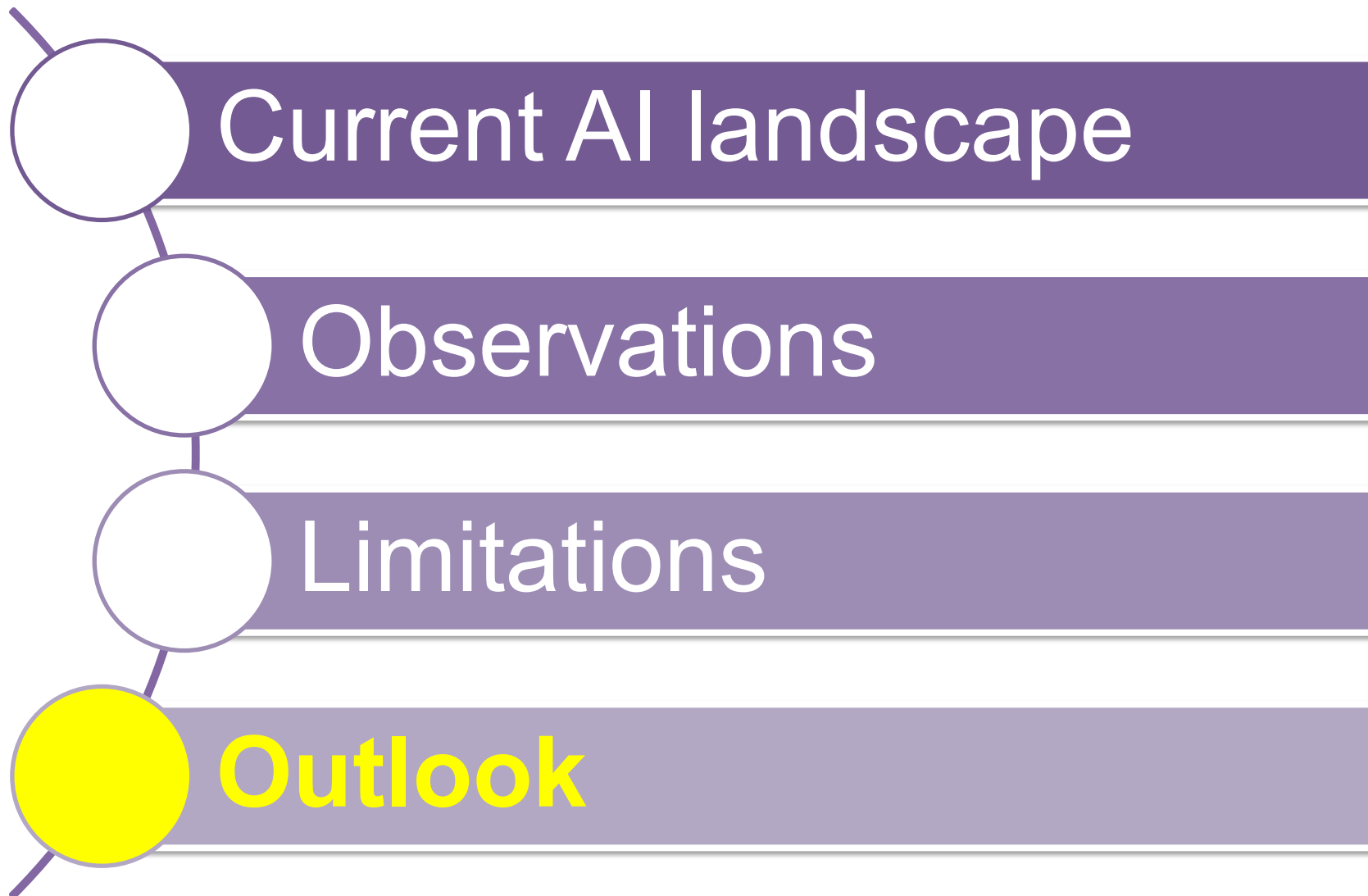
- **In an ideal (OpenROAD) future ...**

Note: <https://github.com/ieee-ceda-datc/OpenROAD-Research>

- **CI-enforced QoR gates:** non-regressing diffs, machine-checkable invariants
- **Provenance metadata:** model, prompt, evaluation context per diff
- **Versioned manifests:** OpenROAD commit + ORFS + PDK + agent → **replay any result**



Agenda



(1) Autonomous $\neq$ *Uncontrolled*

- **Safety, guardrails and governance** – autonomous R&D is not a free-for-all – it requires safe, auditable infrastructure



Human-in-the-loop

- PR reviews
- Code reviews
- Diff explanations
- “Blast radius” analysis



Golden evaluators

- Frozen regression suites
- DRC hard gates
- Timing regression thresholds
- Build + unit test enforcement



Auditability

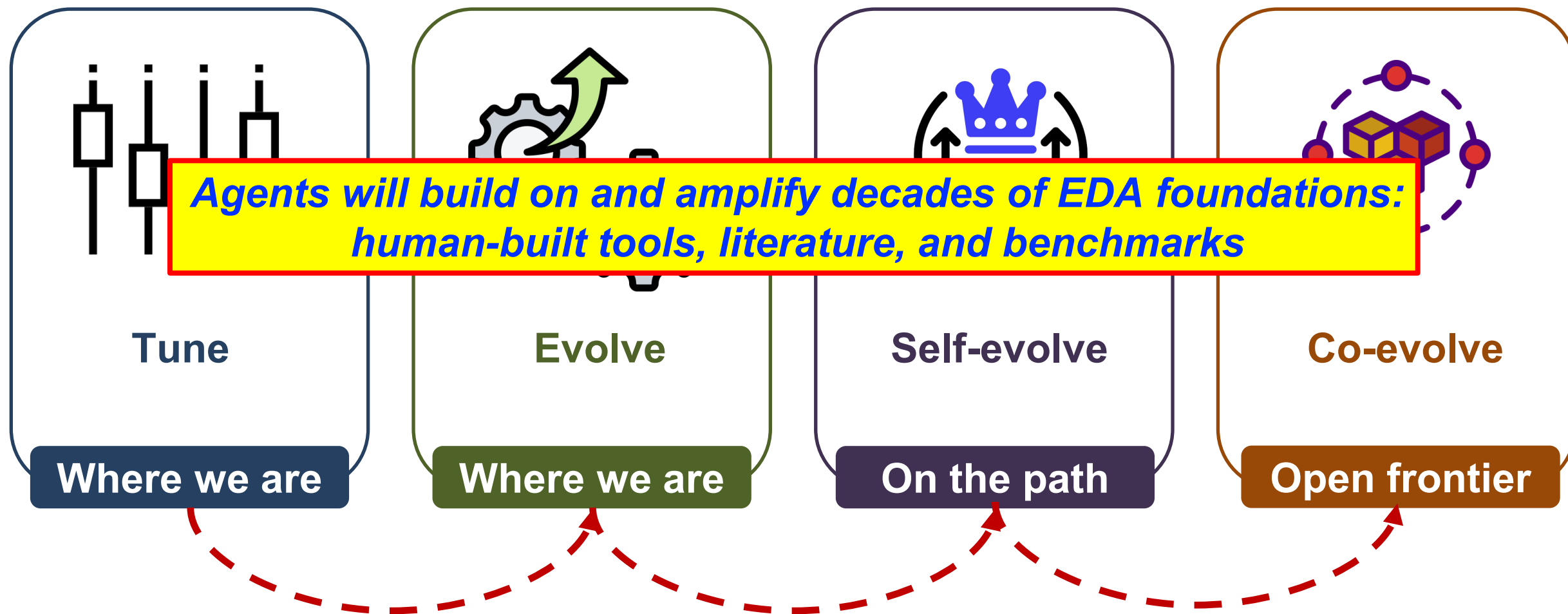
- Every step must be logged, versioned, reproducible
- Who approved what?
- Which QoR?
- Under which commit?

- **Agentic R&D requires MUCH stronger guardrails than human R&D !!!**

(2) Toward *an open vertical environment* for EDA agents

Environment scaffolding (The substrate agents act inside)	Agent capabilities (What we want agents to do)	Ecosystem and governance (What OpenROAD can achieve)
Episode schema for EDA (workspace, actions, constraints, QoR)	Lexicographic, constraint-aware rewards (hard gates → QoR frontier → cost)	Open evolving agent benchmark suite
Cross-stage state & observability (RTL ↔ verif ↔ synth ↔ PD ↔ signoff causality)	Cross-tier orchestration (flow-level + code-level in one loop)	Reproducibility manifests (pinned commits, PDKs, agent versions, prompts)
Multi-fidelity ladder (lint → sim → synth → PD → signoff audit)	Trajectory-grounded training + research-grade memory	CI-enforced QoR gates + machine-checkable invariants
Fast learned surrogates as first-class env citizens	Code-level R&D	Industrial calibration loop + compute equity

(3) The big shift – *scope is expanding*



The unit of optimization in EDA moved up a level – from **parameters** to the **algorithm's source code** itself (then, to **whole flows**)

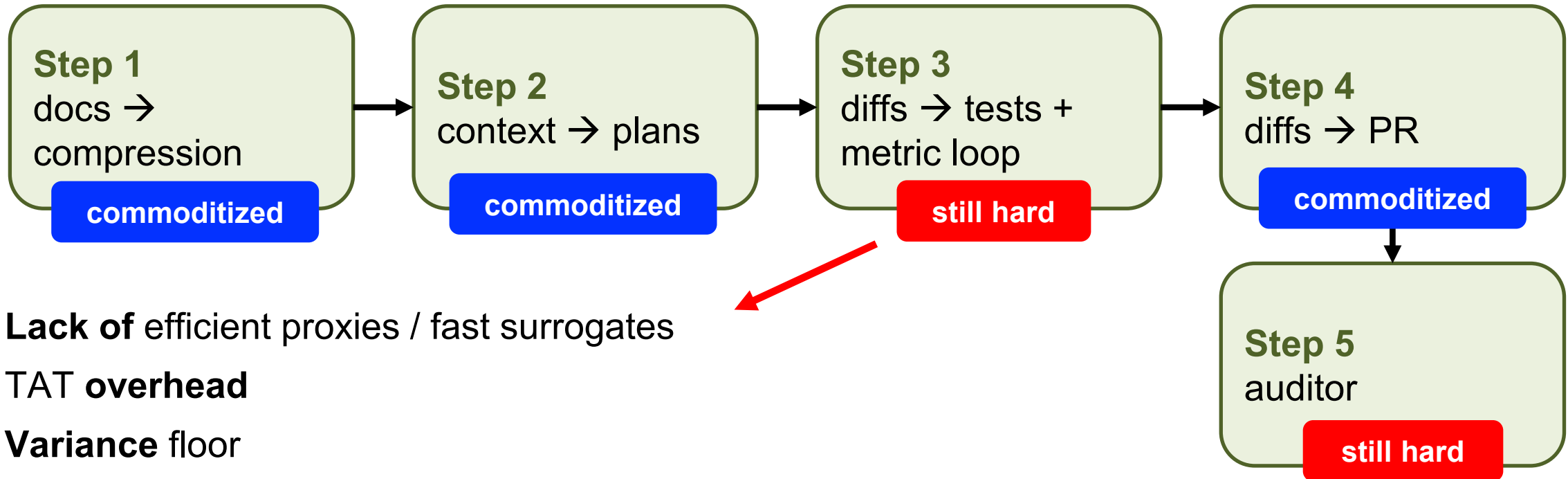


(4) Bottlenecks are moving / falling fast 2025 autonomous OpenROAD-agent

- **Step 1:** Documentation as **compression**
- **Step 2:** Fuse documentation with **non-code context** → implementation plans
- **Step 3:** Implementation plans → **diff** → tests + compile checks + in-loop optimization for metrics
- **Step 4:** Diff → **PR**, with unit tests, comments, mergeability
- **Step 5:** In-house **PR auditor** reviews outputs of Step 4, optionally return to **Step 3**
- **Building the autonomous loop:**
 - Feasible atop **OSS agents** (e.g., Cline, Codex)
 - Harder to integrate with **closed-source agents** (e.g., Cursor)

(4) What shifted in 2026? (*Stronger evaluators / auditors*)

- The 2025 agentic pipeline is **mostly** commoditized now; the value has collapsed into **two steps**



- **Lack of** efficient proxies / fast surrogates
- TAT **overhead**
- **Variance** floor
- Multi-objective ← agents **struggle** with this
- Goodhart: target a metric → it gets **gamed**

- Review **can not scale**
- Confidence vs. correctness
- Provenance + determinism
- **“Spoofable”** gates



(5) Production AI-EDA needs a *stack*, *with human-in-the-loop*

Human control plane (define objectives, judge trade-offs, sign-off)

Governance and provenance (per-diff manifests, CI-gated QoR, license / export policy, inventorship log, slop admission gate)

Agent orchestration (planner + specialized sub-agents, evolution, memory, corpus)

Evaluation and verification (golden evaluators, hard gates, formal checks, Cis, seed runs)

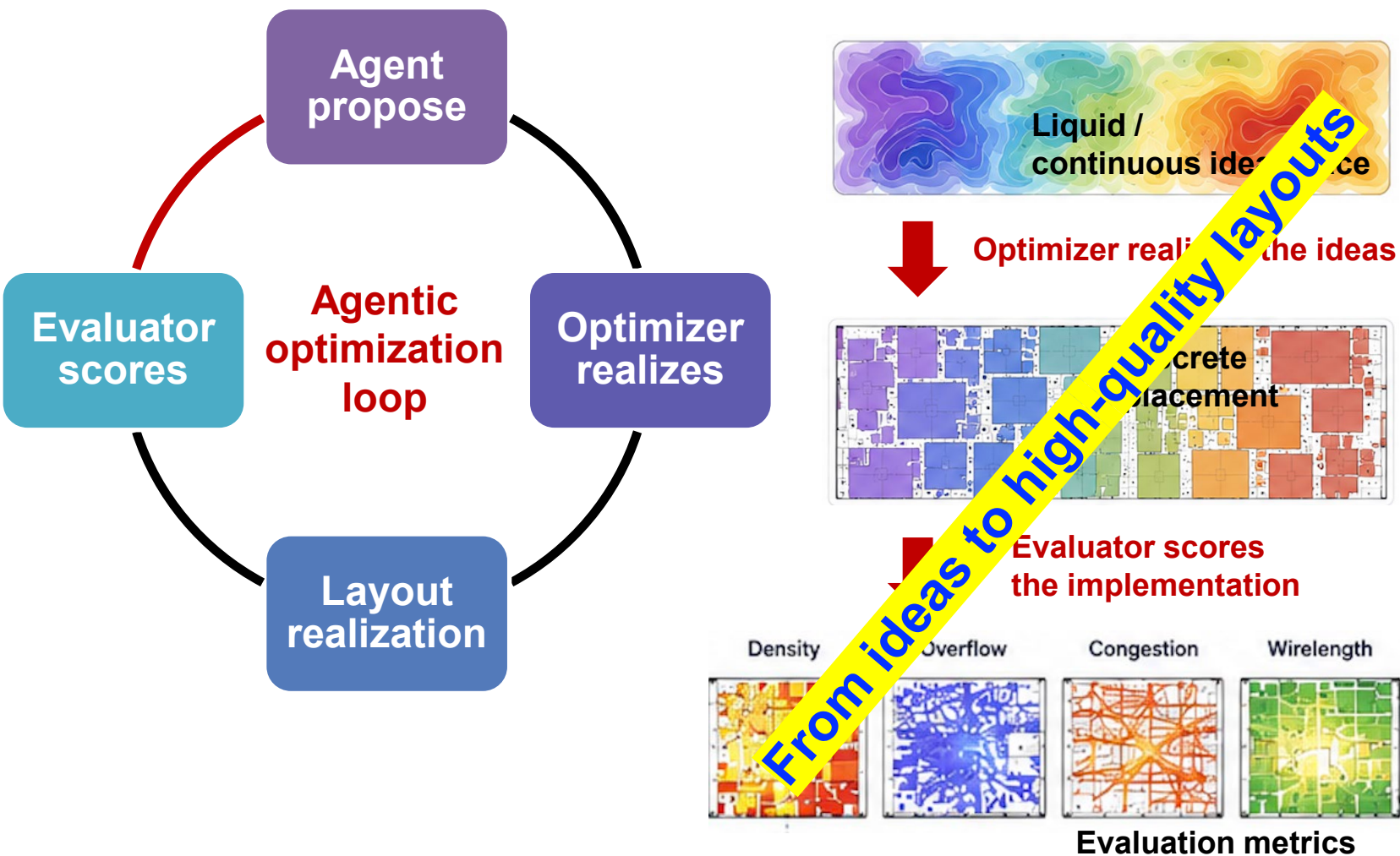
Open EDA foundation (OpenROAD / ORFS, open PDKs, MCP, benchmarks)

Compute + model substrate (open-weight, frontier models, sovereign / on-prem compute)

To reach production maturity

(6) Closing the loop: *optimization is still the lever*

Agents propose; evaluators score; but optimization moves the layout

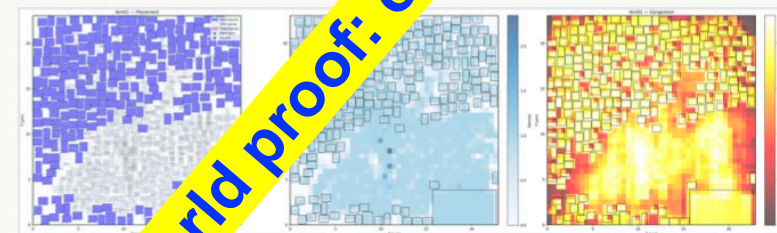


Partcl/HRT Macro Placement Challenge

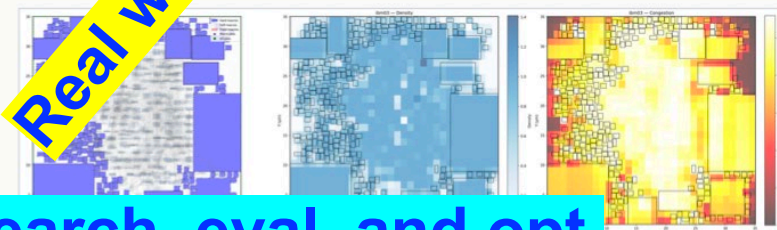


Leaderboard

Rank	Team	Design Proxy Cost
1	"Archgen"	0.9507
2	"Carrotato"	0.9522
3	"JaneRT"	0.9694



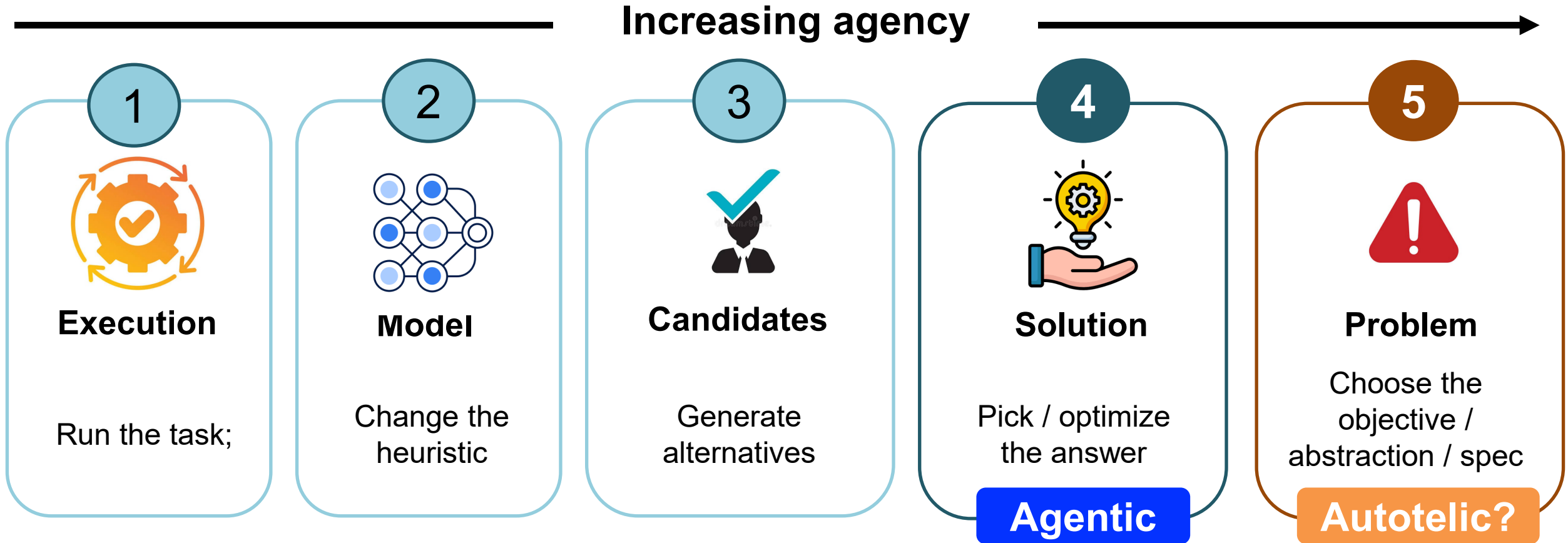
ibm01



Agentic EDA ROI is defined by how we couple search, eval. and opt.

Beyond agentic? → *agency over the problem ...*

- The next era will bring agency over the problem – not just the solution



- Agentic** = agency over the solution with **fixed** problem statement, objective functions, optimization surface

We are here

Sets goals / chooses the problem

- Beyond agentic** = **autotelic** = agency over the problem



Links

- GitHubs/Docker Hubs/etc.
- ORFS-Agent: <https://github.com/ABKGroup/ORFS-Agent>
- EDA corpus: <https://github.com/OpenROAD-Assistant/EDA-Corpus>
- OpenROAD-Agent: <https://github.com/OpenROAD-Assistant/OpenROAD-Agent>
- OpenROAD-Assistant: <https://github.com/OpenROAD-Assistant/OpenROAD-Assistant.git>
- EDA-AI: <https://github.com/Thinklab-SJTU/EDA-AI/tree/main>
- EvoPlace: <https://github.com/yaoxufeng/EvoPlace/tree/master>
- ChatEDA: <https://github.com/wuhy68/ChatEDA>
- AuDoPEDA: <https://hub.docker.com/r/abkgroupecsd/audopeda>
- HELIX: <https://github.com/ABKGroup/HELIX.git>
- MLCAD26 contest: <https://github.com/ASU-VDA-Lab/MLCAD26-Contest.git>
- GR-Evolve: <https://github.com/ASU-VDA-Lab/GR-Evolve>
- CHICO-Agent: <https://github.com/ASU-VDA-Lab/CHICO-Agent>
- OpenROAD: <https://github.com/The-OpenROAD-Project/OpenROAD.git>
- ORFS: <https://github.com/The-OpenROAD-Project/OpenROAD-flow-scripts.git>
- ORFS Docker image: <https://hub.docker.com/r/openroad/orfs>
- FastPart / hypergraph partitioning artifacts: <https://vlsicad.ucsd.edu/hypergraphs/>
- MacroPlacement benchmark/infrastructure: <https://github.com/TILOS-AI-Institute/MacroPlacement>
- CircuitNet: <https://github.com/circuitnet/CircuitNet>
- CircuitOps: <https://github.com/NVlabs/CircuitOps>



Thank You !
