HAMMER

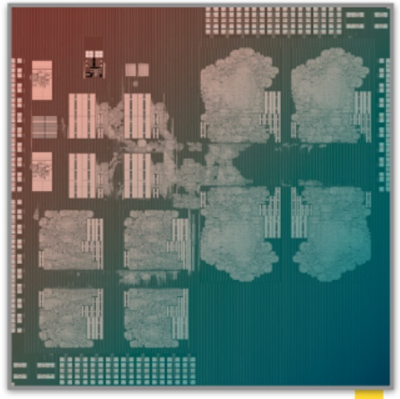
Beyond Make: DAG-Driven Physical Design and Intelligent Dependencies with SledgeHammer

Andre Green, Desvaun Drummond, Brandon Kim, Juhyun Do, Lawrence Rhee, Isabelle Hsu, Henry Wilson, Colin O'Brien, Sagar Karandikar, and Borivoje Nikoli´c

Open-Source Computer Architecture Research (OSCAR) - ISCA '26
June 28, 2026

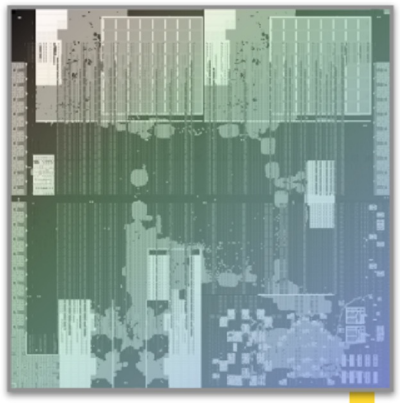
Tapeout: Recent History

Cygnus
(Q2 2024)



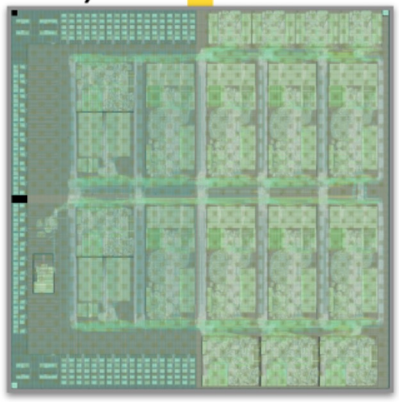
- DSP oriented
- 4x “Big” **Saturn** (VLEN=512)
- 4x “Little” **Saturn** (VLEN=128)
- **C2C TileLink**
- 512KB LLC
- 128KB SPAD

KODIAK
(Q3 2025)



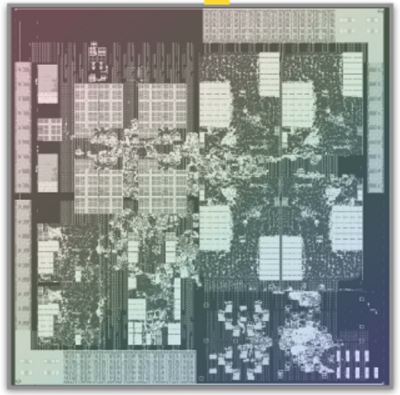
- AI (existing + emerging) oriented
- 1x Rocket
- 2x **Saturn** (VLEN=512) w OPU
- 1x Improved **UCIe-Lite Module**
- **C2C** (Tilelink-based protocol)
- **TACIT** Instr. Tracer
- 512KB LLC
- 128KB Global SPAD

MAVERIC
(Q4 2023)



- ML oriented
- 4x Rocket
- 5x **Gemini** (FP16)
- 8x **Gemini** (INT8)
- 512KB LLC
- 1MB Global SPAD

Sirius
(Q3 2024)

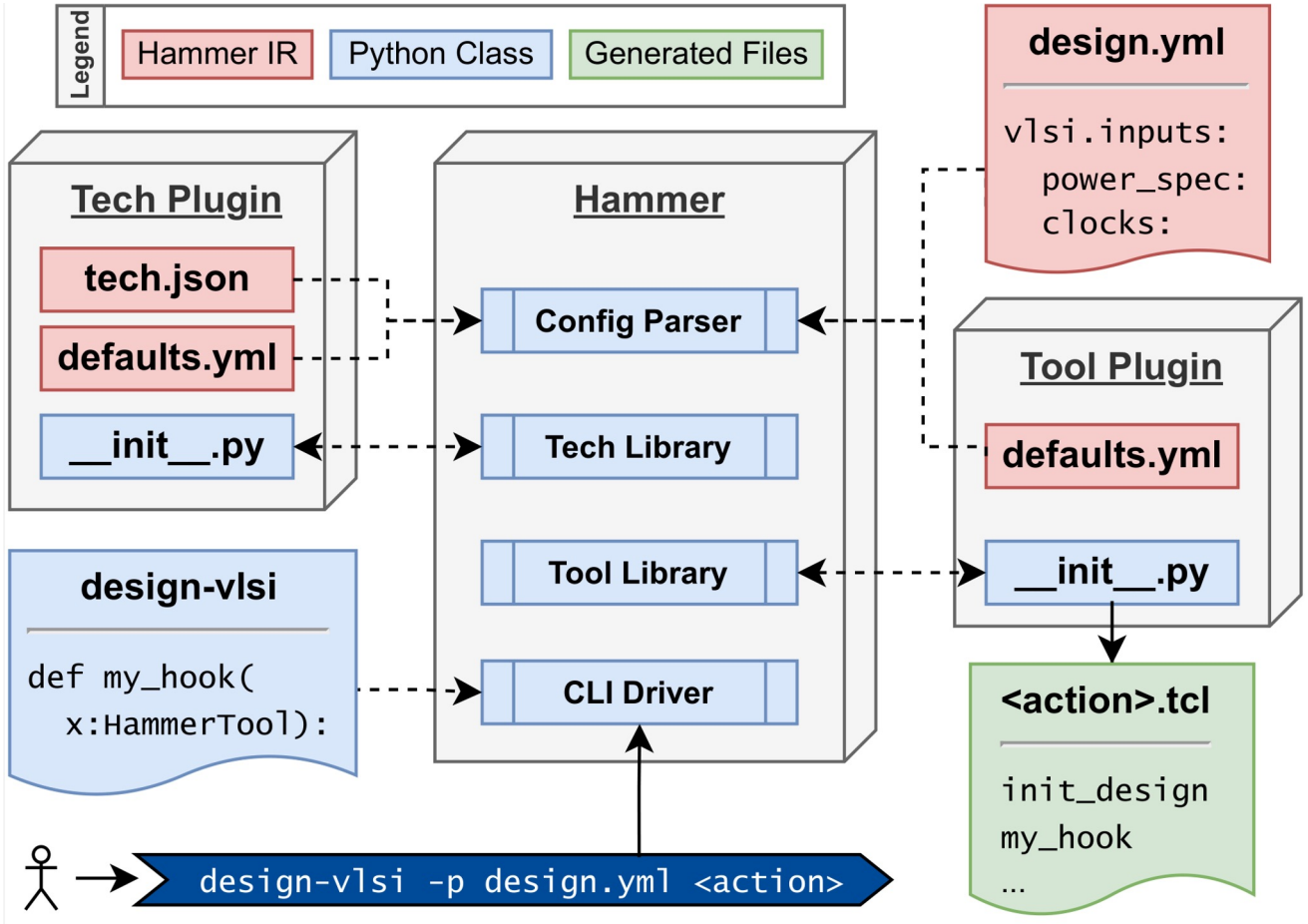


- AI/LLM oriented
- 1x Rocket
- 4x Dense **Gemmini** (FP16)
- 4x **Saturn** (VLEN=512)
- 2x **Stellar** (FP16)
- 1x **UCIe-Lite Module**
- **C2C TileLink**
- 512KB LLC
- 256KB Global SPAD

Hammer Physical Design Framework

- Python physical design framework (RTL -> GDS).
- Parameterizable flows via interchangeable technology (PDK) and EDA tool plugins.
- Supports custom TCL hooks injected at any flow step.
- Rapid prototyping through aggressive code reuse.

HAMMER



Limitations

HAMMER

- **Pessimistic Make Dependencies:** Tracks file timestamps instead of content, so minor edits may trigger redundant reruns of earlier stages.
- **Rigid Flow Structure:** Difficult to integrate custom physical design stages once the flow is initialized.
- **Strictly Command-Line Interface:** No visual flow mapping or real-time progress tracking, which can create a learning curve for new users.
- **Extensive Logging:** Critical debugging data and tool errors can end up buried deep inside massive, unorganized text logs.
- **Isolated Workspaces:** Lacks a shared backend or multi-user infrastructure for real-time collaboration and artifact reuse.

Limitations

HAMMER

- **Pessimistic Make Dependencies:** Tracks file timestamps instead of content, so minor edits may trigger redundant reruns of earlier stages.

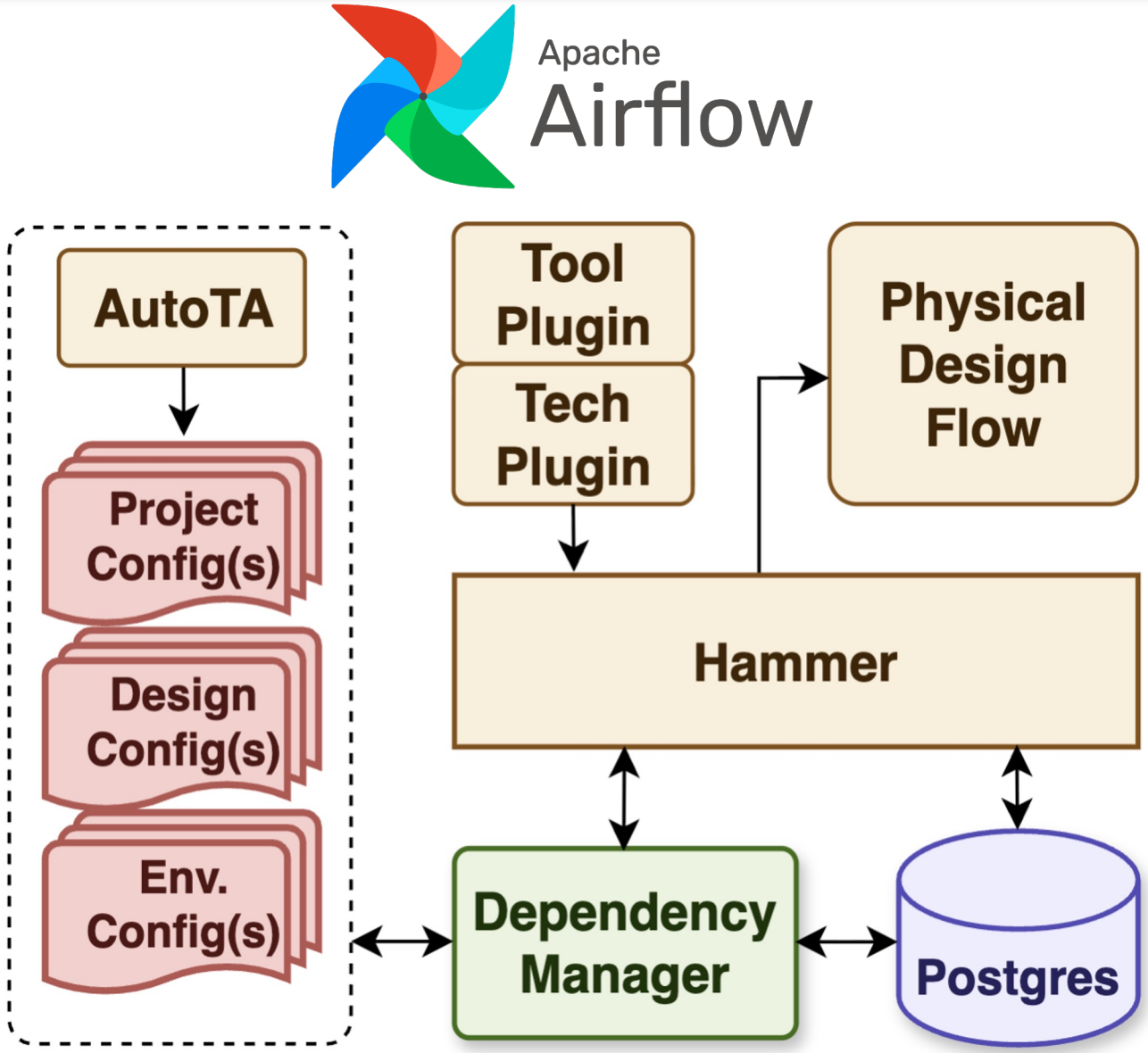
SledgeHammer aims to tackle many of these limitations by integrating workflow orchestration for efficient physical design

- **Extensive Logging:** Critical debugging data and tool errors can end up buried deep inside massive, unorganized text logs.

- **Isolated Workspaces:** Lacks a shared backend or multi-user infrastructure for real-time collaboration and artifact reuse.

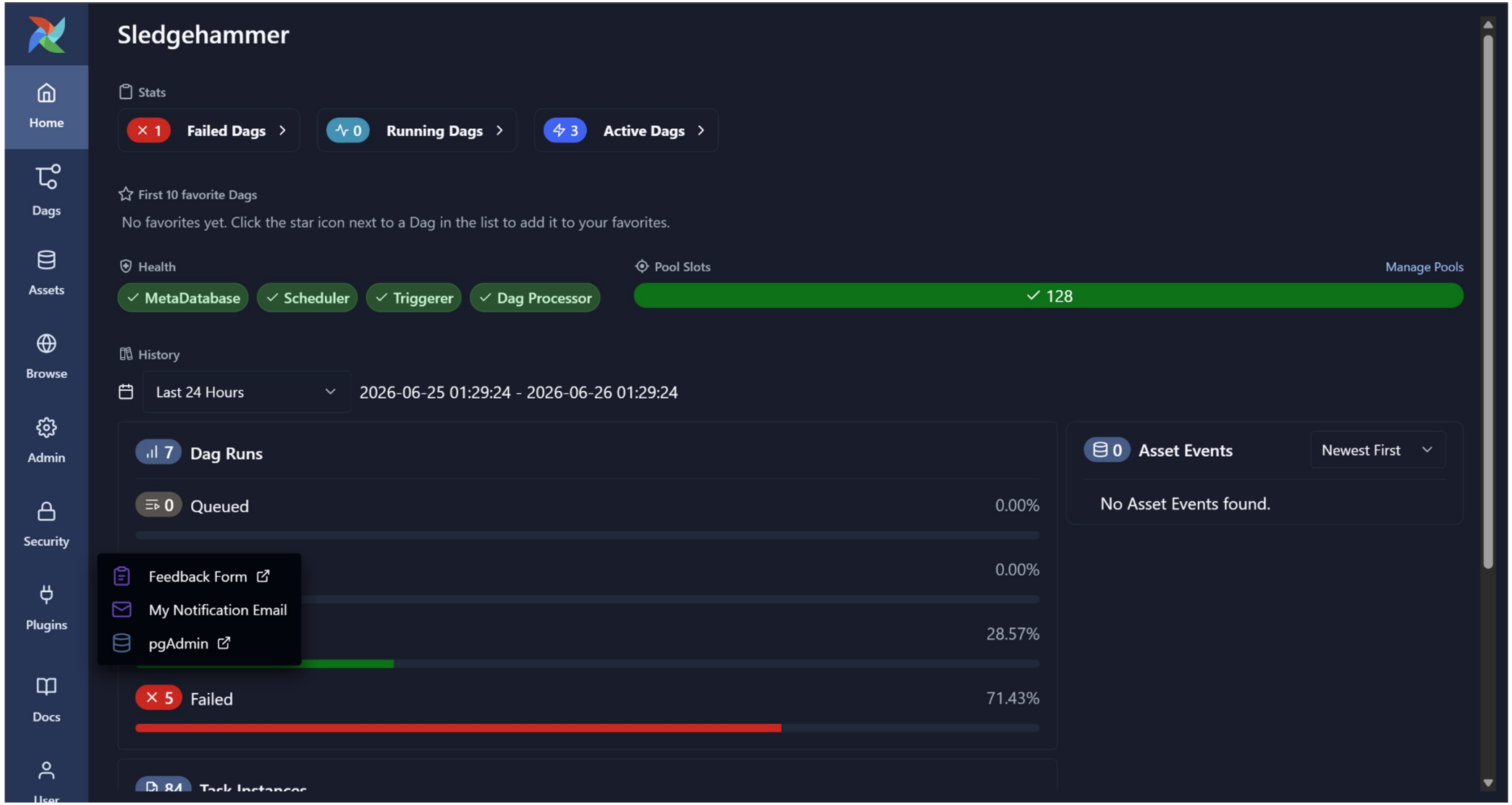
SledgeHammer Architecture

- Backed by Apache Airflow, SledgeHammer replaces the Make build system with directed acyclic graphs (DAGs) to orchestrate the physical design flow.
- PD stages are encapsulated as nodes, while their dependent stages are specified by their edges.
- Adds a **GUI** to visualize the workflow, improves **dependency management**, integrates a **database** backend for Airflow & SledgeHammer metadata, and many more features that will be discussed throughout the presentation.



GUI

Allows Users to Run Physical Design Flows Directly From GUI

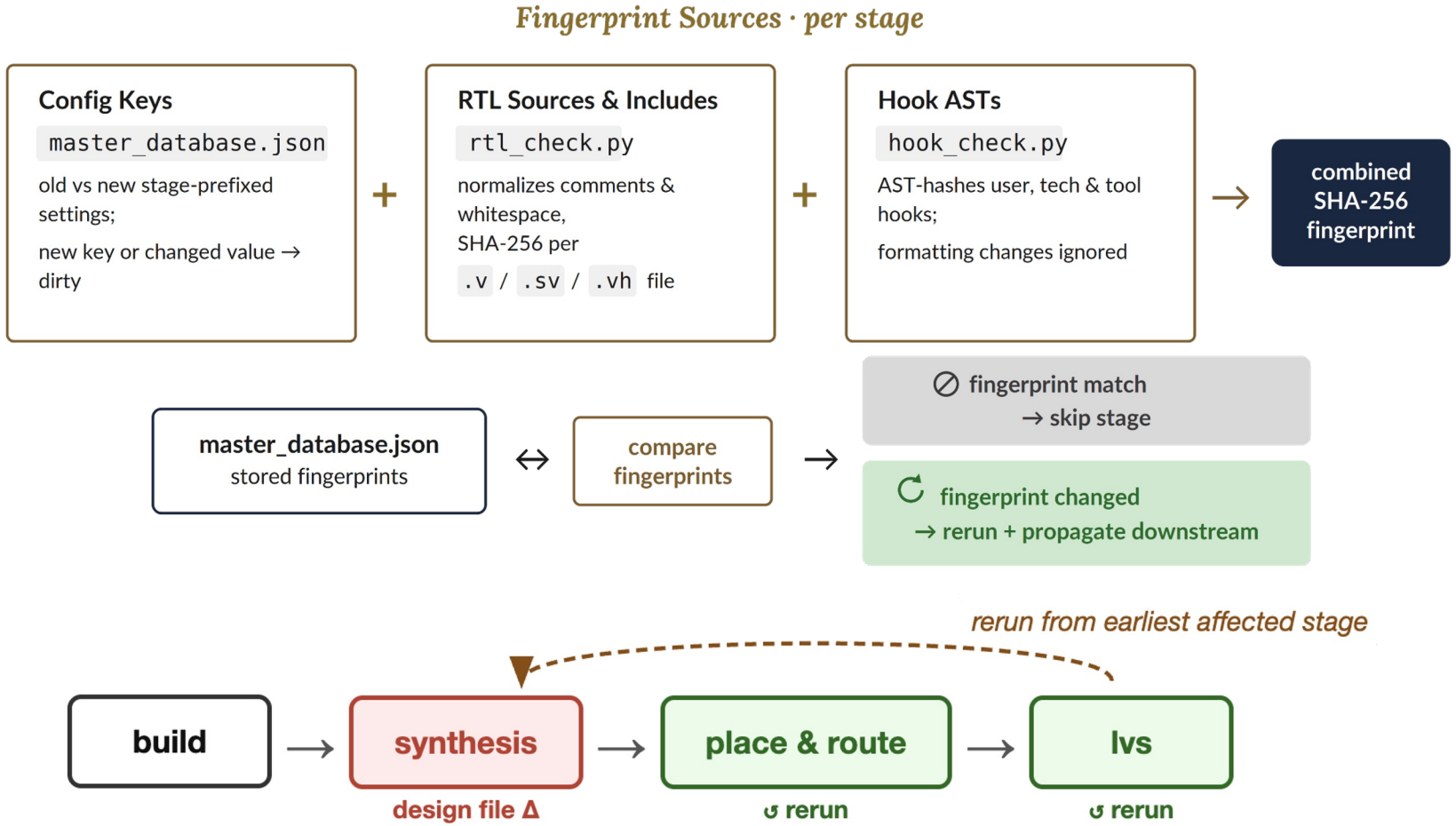


Dependency Management

Fine-Grained Dependency Management

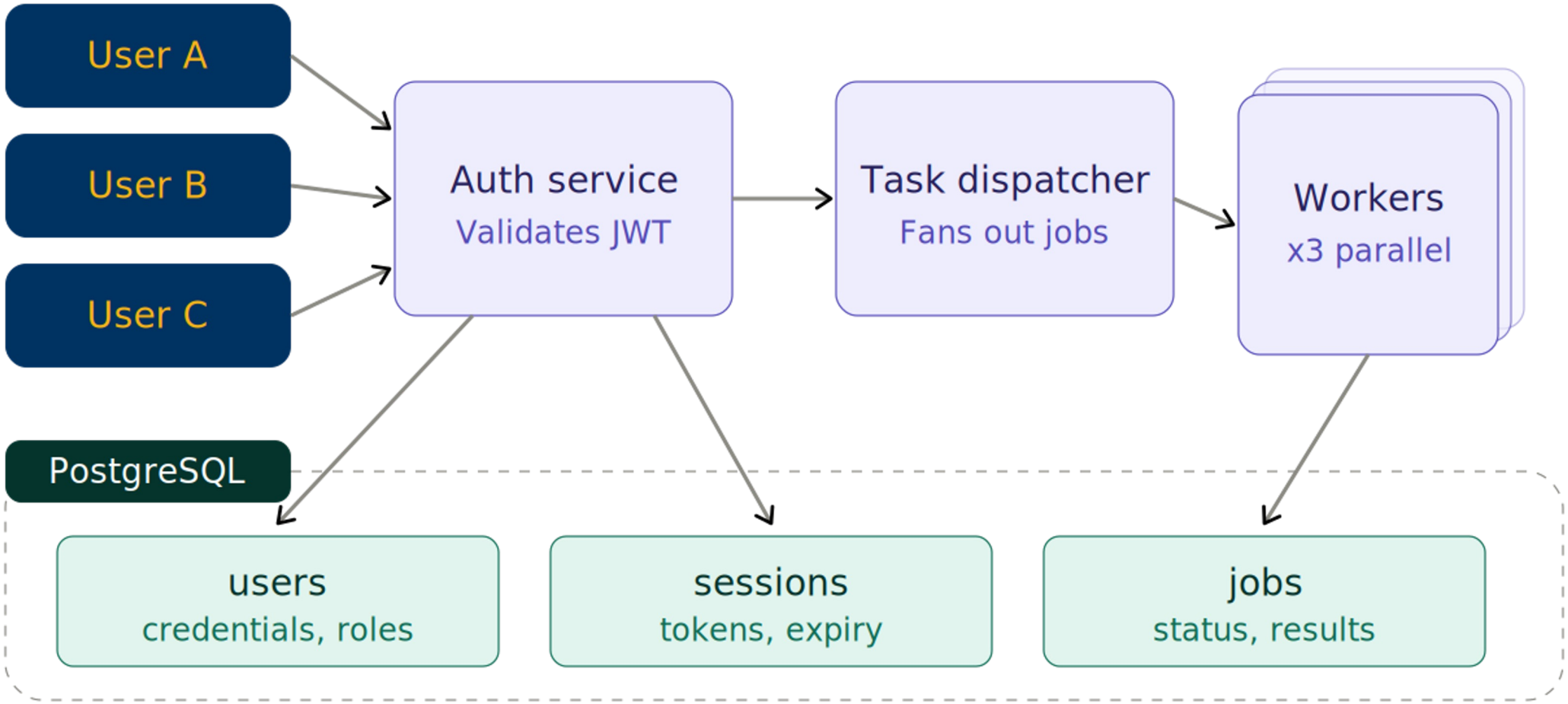
Make Build System: Timestamp-based dependency checks.

SledgeHammer: Monitors independent configuration settings, design files, and hook changes.



PostgreSQL Database

Run Parallel Flows & Store Data Across Users



Chipyard Integration

CHIPYARD

Chipyard VLSI Makefile

1 Parses configs

`CONFIG` `VLSI_TOP` `tech_name` `INPUT_CONFS`

SoC config, target module, tech, YAML files

2 Generates collateral

`make verilog`

Elaborates Chisel→Verilog + tech mapping

3 Generates the DAG

`make build`

Builds the PD flow graph from the stages

HAMMER

DAG Generation

1 Takes Hammer configs

`project` `environment` `technology` `tool` `plugin`

Project, environment, tool & technology settings

2 Builds the flow graph

Each stage becomes a task with the commands its tool needs

Wired by upstream / downstream dependencies

3 Emits the Airflow DAG

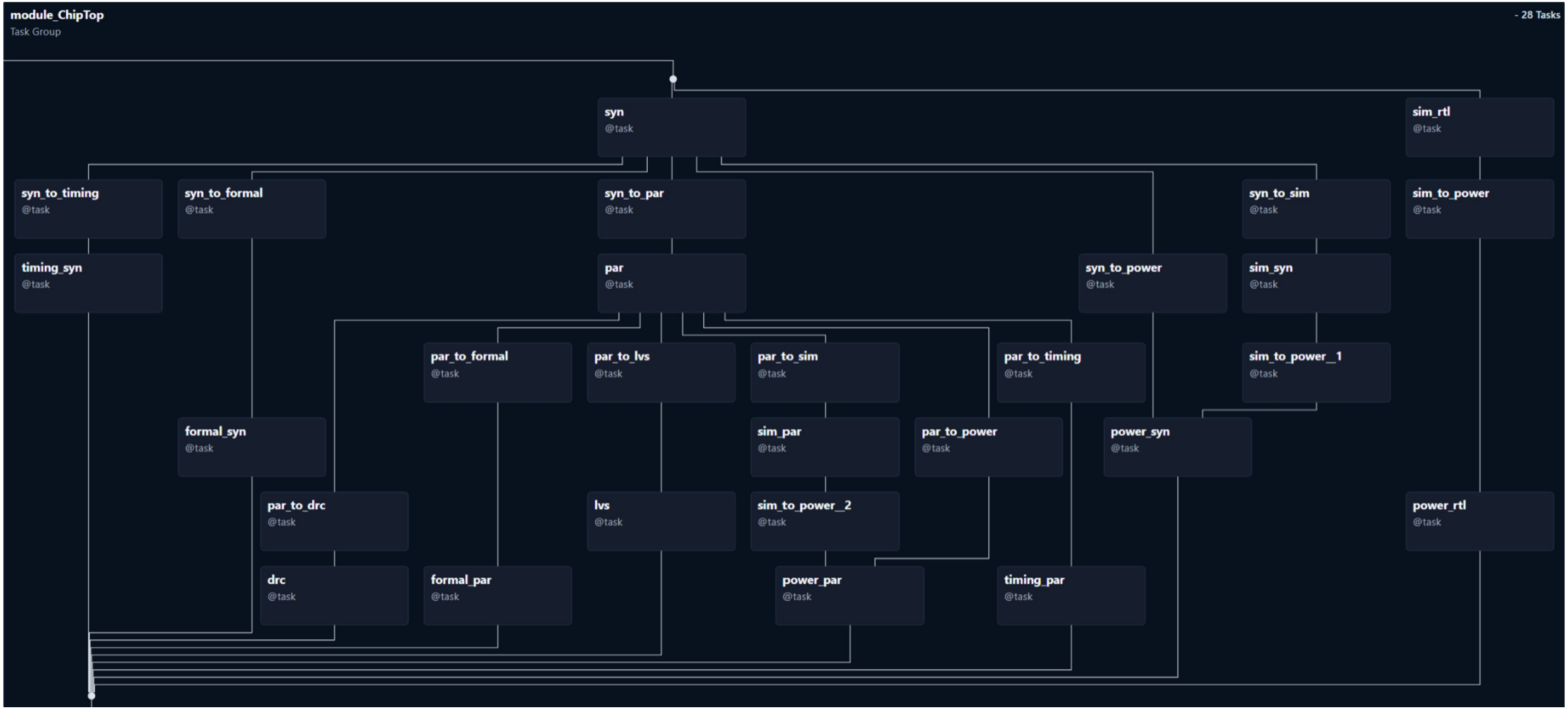
`replaces hammer.d`

Airflow runs the stages as a DAG

DAG Generation

Default PD flow after initialization

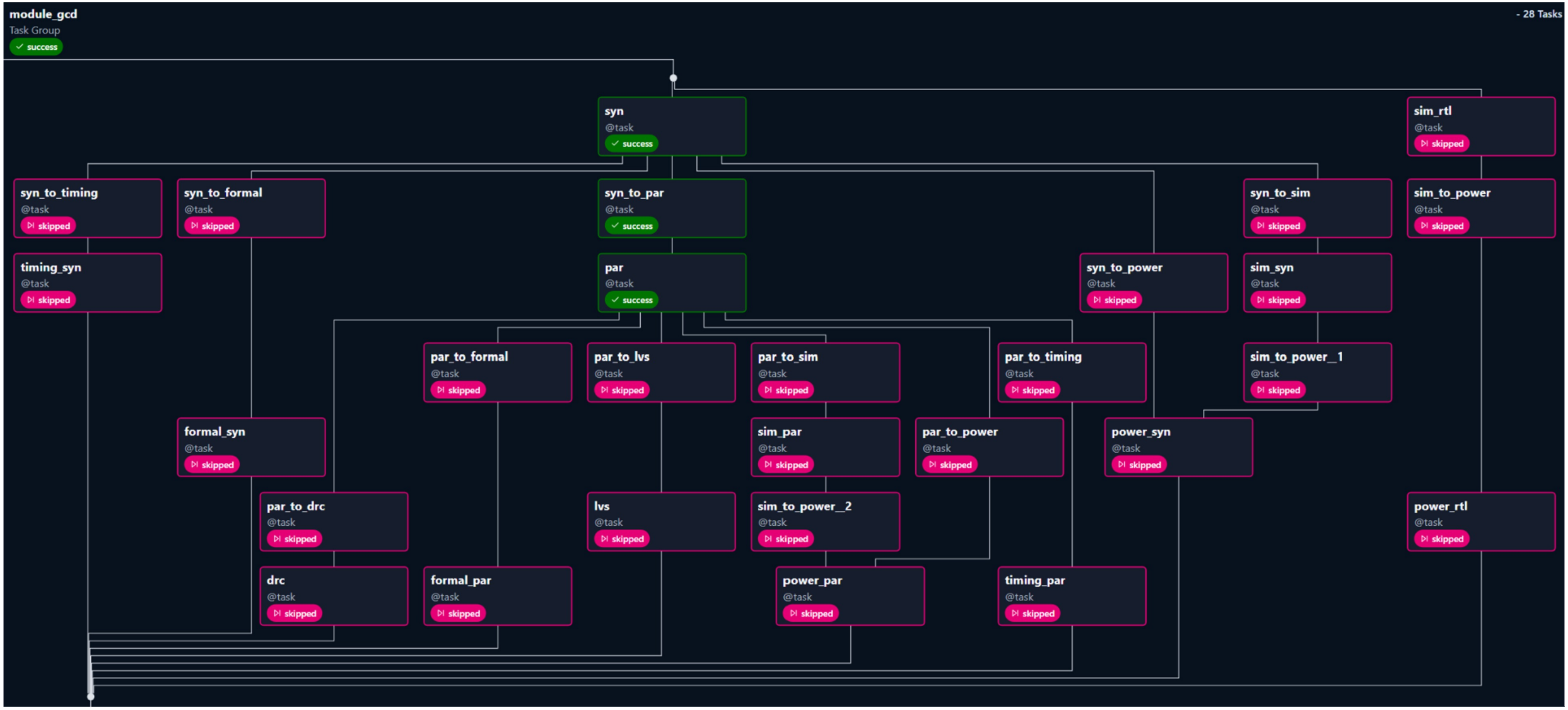
Pair With Chipyard Environment to Automatically Generate PD Flows



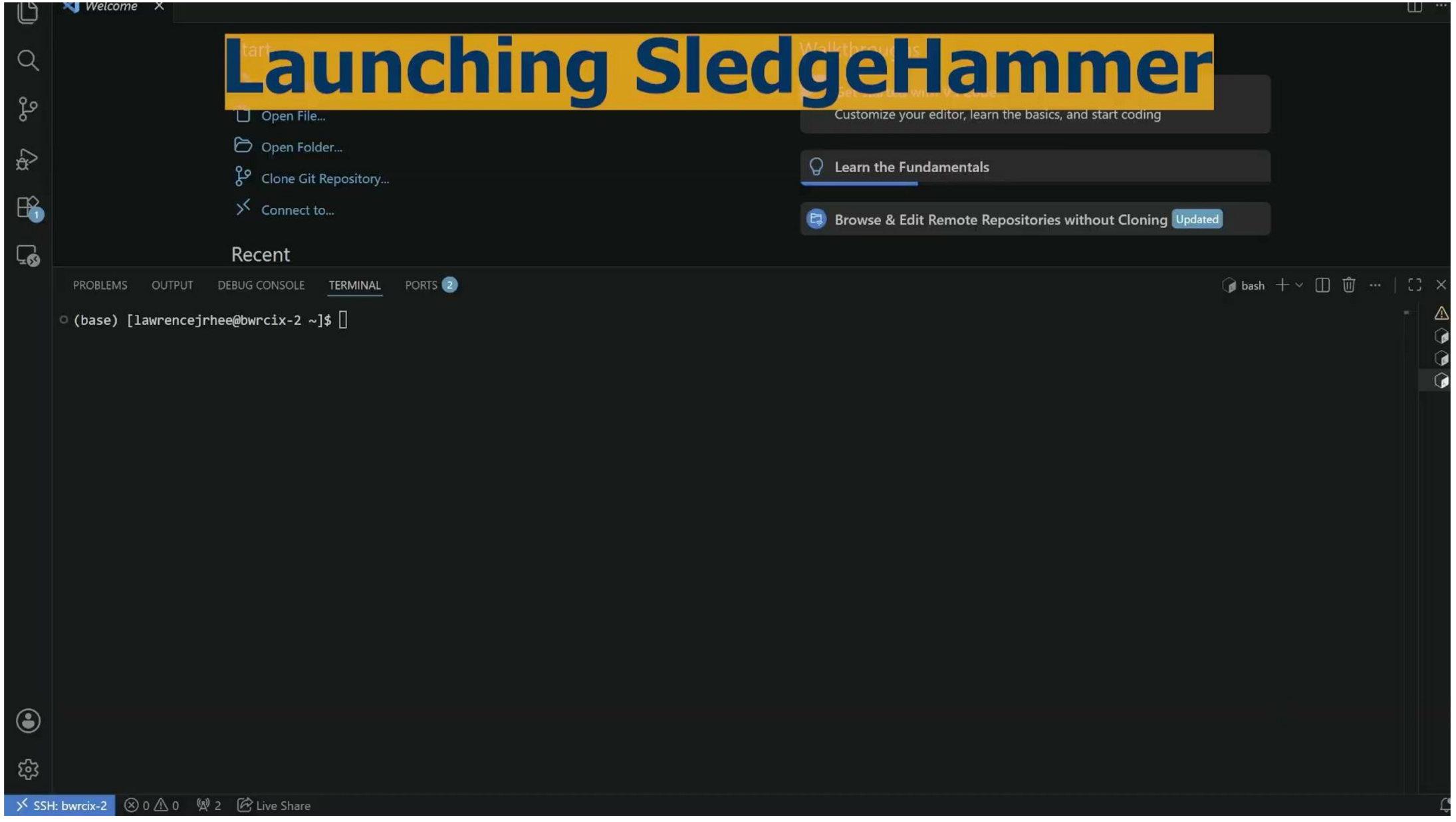
DAG Generation

Real-time visualization of flow progression

Pair With Chipyard Environment to Automatically Generate PD Flows



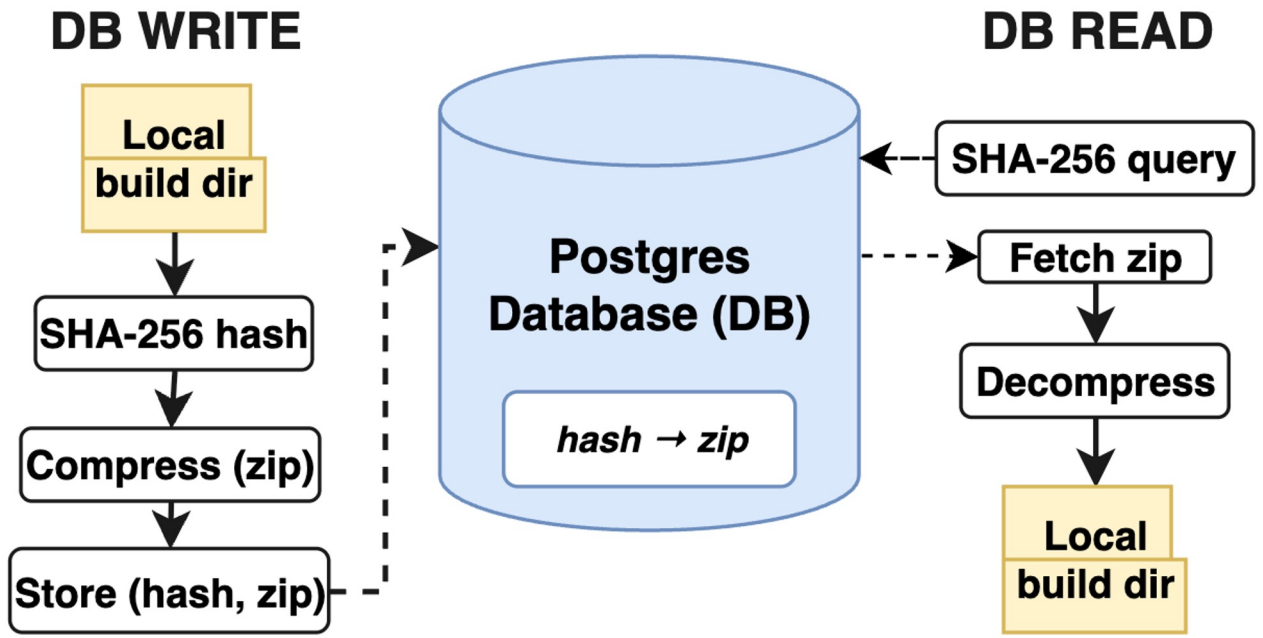
DEMO

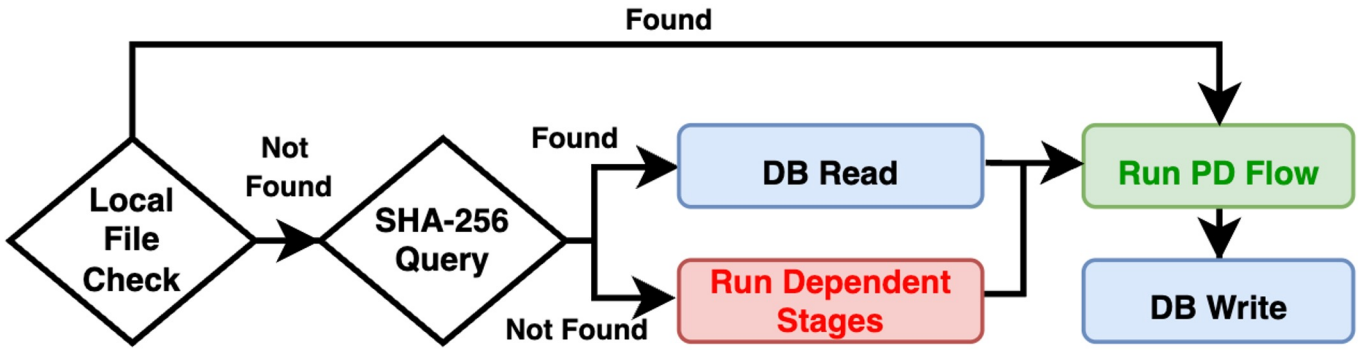

Launching SledgeHammer
Welcome
Open File...
Open Folder...
Clone Git Repository...
Connect to...
Recent
Customize your editor, learn the basics, and start coding
Learn the Fundamentals
Browse & Edit Remote Repositories without Cloning
Updated
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS 2
bash
(base) [lawrencejrhee@bwrcix-2 ~]$
SSH: bwrcix-2
Live Share

PD Caching

Hash-Based Caching of PD Stage Outputs

- **Fingerprint every stage:** SHA-256 hash of the build dir as a unique key
- **Cache once, shared across teams:** stored in Postgres as hash → zip
- **Check local first** — reuse the local build dir if it's already there; the DB cache is the fallback
- **Cache hit →** skip the stage — fetch, decompress, continue the PD flow
- **Cache miss →** run, then cache — run stages, write result back for next time



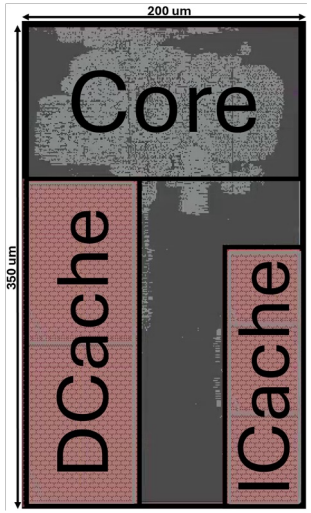


Evaluated Designs

Run Parallel Flows & Store Data Across Users

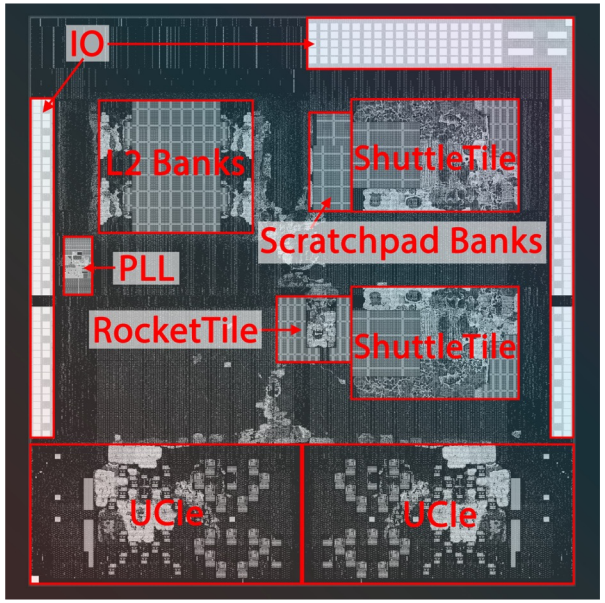
Scale up designs from individual modules -> SoCs

Modular



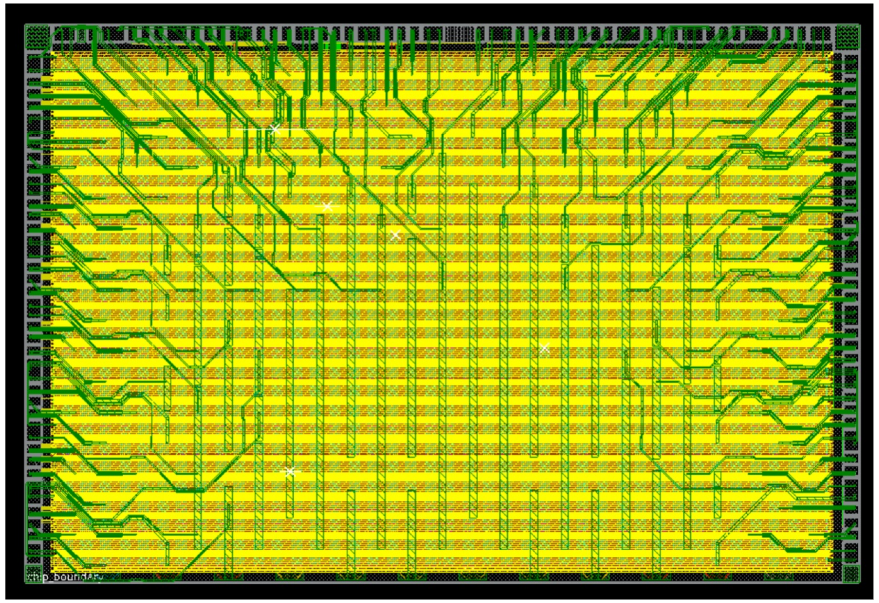
Rocket

Hierarchical



Kodiak

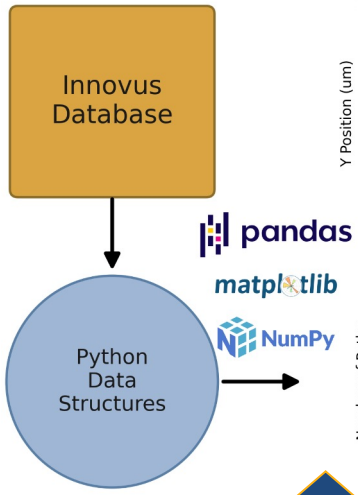
Large-Scale SoC Design

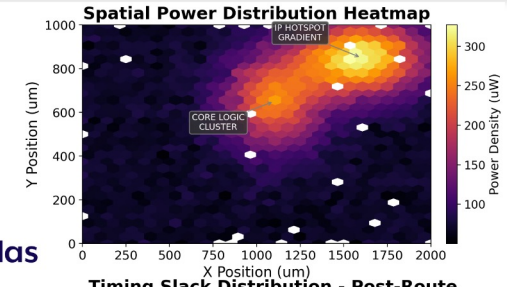


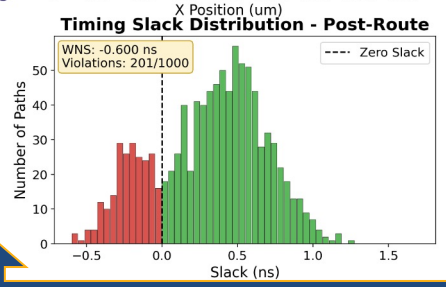
EE290 Tapeout Chip

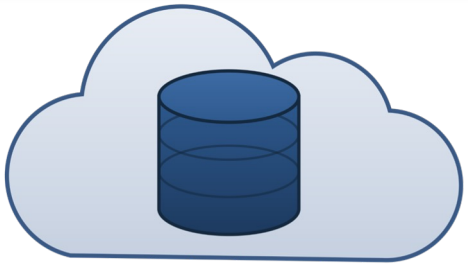
Future Work

Python Interface





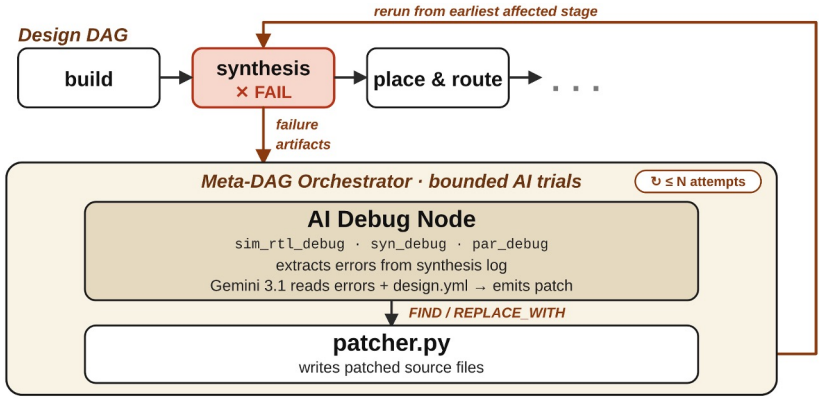




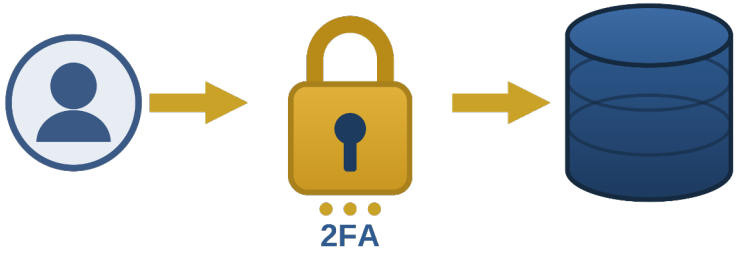
Cloud

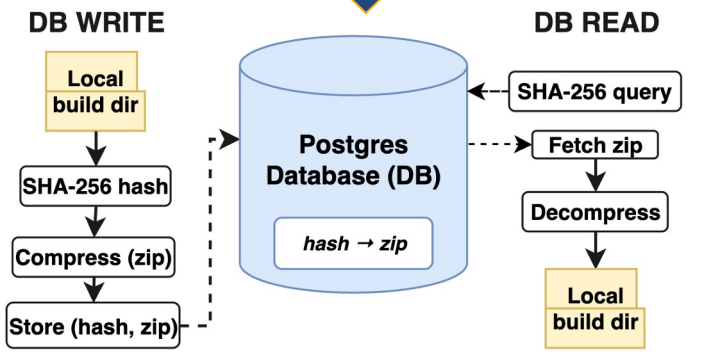
Collaborative VLSI Workspace

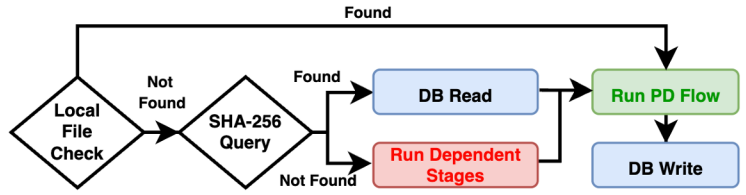
AutoTA



Security







Summary

- SledgeHammer targets the limitations of Hammer by:
  - **Reducing redundant PD stage execution** to improve overall TAT of RTL -> GDS
  - Incorporating a database backend to cache and **share physical design data across users**
  - **Visualize** the PD lifecycle in real-time, allowing for users to monitor per-stage parseable logs & view the flows of other users in the shared environment
  - **Automatically generate flows** & flexibly modify PD stages after initialization
- We intend to continue developing SledgeHammer to provide:
  - AutoTA integration into SledgeHammer to aid design space exploration/optimization by using LLMs to modify design and configuration files based on PD QoR
  - Improve security of the webserver and database
  - Launch SledgeHammer on cloud for remote flow execution
  - SledgeHammer Studio: Collaborative VLSI environment that will allow multiple users to design in a shared workspace while sharing collateral and artifacts amongst each other to reduce redundant flow runs and improve visibility across teams to aid the tapeout process

Questions?

Andre Green, 4th-year PhD Candidate
UC Berkeley
andre_green@berkeley.edu

HAMMER

https://github.com/ucb-bar/hammer