SledgeHammer: Scalable, Collaborative, Physical Design Flow Orchestration for Agile Hardware Teams

Andre Green
University of California, Berkeley
Berkeley, CA, USA
andre_green@berkeley.edu

Isabelle Hsu
University of California, Berkeley
Berkeley, CA, USA
isabellehsu@berkeley.edu

Harrison Liew*
Altera
Portland, OR, USA
harrison.liew@altera.com

Sunjin Choi
University of California, Berkeley
Berkeley, CA, USA
sunjin_choi@berkeley.edu

Ken Ho
University of California, Berkeley
Berkeley, CA, USA
ken_ho@berkeley.edu

Daniel Lovell
University of California, Berkeley
Berkeley, CA, USA
dlovell98@berkeley.edu

Sagar Karandikar
University of California, Berkeley
Berkeley, CA, USA
skarandikar@berkeley.edu

Borivoje Nikolić
University of California, Berkeley
Berkeley, CA, USA
bora@berkeley.edu

***Abstract*—The increasing complexity of system-on-chip (SoC) and system-in-package (SiP) designs, together with the impact of shrinking technology nodes, has amplified the need for more flexible, scalable, and efficient physical design flows. To address the limitations of existing physical design (PD) flow tools, we present SledgeHammer, an extension of the Hammer PD framework that incorporates principled workflow orchestration, improves modularity, and enables collaborative physical design. SledgeHammer overhauls Hammer's underlying architecture by replacing its rigid, timestamp-based build system with a directed acyclic graph (DAG) driven model for fine-grained dependency management and customizable flow execution. SledgeHammer supports a shared VLSI workspace with a web-based graphical user interface (GUI), enabling real-time flow visualization, parseable logging and relevant metric tracking, along with providing a multi-user collaborative workspace. Through integration with the Chipyard SoC design environment, SledgeHammer has been validated on designs across various nodes, from 130nm to 16nm, including the Rocket 5-stage RISC-V in-order scalar core and a chiplet with UCIe interfaces. We demonstrate how SledgeHammer reduces redundant computation, streamlines debugging, and enables future enhancements such as cloud deployment, advanced checkpointing, and further support for large-scale SoCs and SiPs. By bridging modern workflow orchestration with physical design, SledgeHammer redefines the possibilities of VLSI development, empowering teams to collaborate more effectively and build the next generation of SoCs and SiPs.**

***Index Terms*—Very large scale integration design, Electronic design automation, Physical design (EDA), Methodologies for EDA, Software tools for EDA, System on a chip.**

I. Introduction

As SoC complexity has grown and technology nodes have continued to scale rapidly, the role of physical design frameworks has evolved from enabling basic modular and automated chip implementations to supporting more sophisticated orchestration of heterogeneous tools and processes [1]. In response to these increasing demands, it has become essential for modern EDA tools to integrate advanced features that maintain design efficiency and streamline the overall chip development process.

One tool that effectively addresses modern physical design challenges is Hammer [2], an open-source physical design flow generator recognized for enabling code reuse and improved design efficiency in VLSI development [3]. Hammer supports synthesis, place and route, DRC & LVS checks, formal verification, static timing analysis, and power analysis. Engineering change order patches are not yet supported. Alternative open-source tools, such as mflowgen [4] and SiliconCompiler [5], have addressed similar challenges by exploring modular flow generation and task management. Features such as nodal flow encapsulation and task-specific targeting across different technologies and tools in mflowgen, along with metric tracking, flow automation, and a Python-based API for simplified flow parameterization in SiliconCompiler highlight key strengths of these tools along with areas that would be highly beneficial to adopt within Hammer's architecture. Despite their benefits, these tools do not provide the same level of flow reusability supported by Hammer's structural approach towards decoupling project, environment, tool, and technology configurations through its intermediate representation. While Hammer has been instrumental in the design of numerous SoCs, we identified several areas for improvements, such as its timestamp-based build system, which can occasionally trigger unnecessary reruns of earlier stages, and a rigid dependency structure that restricts flow flexibility.

Beyond Hammer's limitations in its build system and dependency structure, there is a recognized need for features that reduce debugging complexity, accelerate multi-user design tasks, and provide a more consistent and integrated physical design flow experience. The ability to reorder PD stages and efficiently modify dependencies is essential for achieving greater flexibility and customization across a wide range of design projects.

*Affiliated with University of California, Berkeley at the time of this work

To overcome these challenges, we introduce **SledgeHammer**, a scalable and collaborative physical design flow orchestrator that builds upon Hammer's foundation, extending its architecture with new capabilities to improve efficiency and address prior design constraints. Preliminary results demonstrate that SledgeHammer not only replicates Hammer-generated flows, but does so more efficiently by leveraging intelligent dependency tracking and providing users with finer-grained control over the PD process. The subsequent sections explore the architecture of SledgeHammer, II, followed by an evaluation of its effectiveness, III.

II. SledgeHammer Architecture

To accommodate the increasing complexity of chip architectures and PD flows, SledgeHammer adopts a different design philosophy. It removes the strict build system responsible for dependency management, as its inefficiencies often lead to substantial overhead that adds hours, if not days, to the PD of complex SoCs. We moved toward the integration of workflow orchestration to coordinate physical design stages using a directed acyclic graph (DAG)-based model, improving dependency tracking and providing user's with greater control over their flows.

A. Apache Airflow

Apache Airflow is a workflow orchestration framework that utilizes DAGs to define and control the execution order of user-specified tasks within complex processes [6]. When paired with physical design, workflow orchestration tools, such as Airflow, provide a flexible interface to control the order and behavior of flow execution during the VLSI process. This is accomplished by defining stages of the PD flow as tasks (nodes) within the DAG structure and specifying their upstream and downstream relationships (edges) based on the desired execution order (Fig. 1). These stages, such as synthesis or place and route, are represented as a node to encapsulate the existing Hammer execution, while capturing valuable metrics throughout the flow.

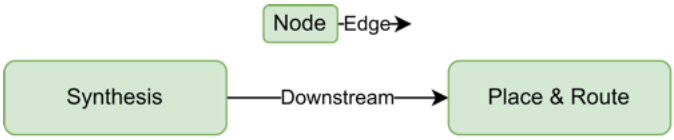

Fig. 1. Apache Airflow's nodal behavior for setting upstream/downstream tasks.

B. SledgeHammer Integration

Airflow's Python-native configuration aligned well with Hammer's architecture, making it a strong candidate for developing our next-generation framework. In Hammer's base architecture, configuration settings are passed into a design through a combination of project-specific and environment-level parameters, along with plugins for the target technology and EDA tools. A new flow can be created by adjusting these inputs to the Hammer driver, which then generates the appropriate sequence of steps to be taken during PD.

This is where Hammer excels in promoting code reusability, by simplifying the generation of new flows and abstracting away tool or process-specific details from the core design process. However, Hammer relies on timestamp-based checks of input configurations to decide if a stage should be reran, meaning inconsequential changes to a file could unnecessarily trigger re-execution. With the removal of this timestamp-based build system, SledgeHammer's shift to workflow-orchestrated dependency management provides a more flexible and extensible framework for users to define, manage, and execute their flows (Fig. 2).

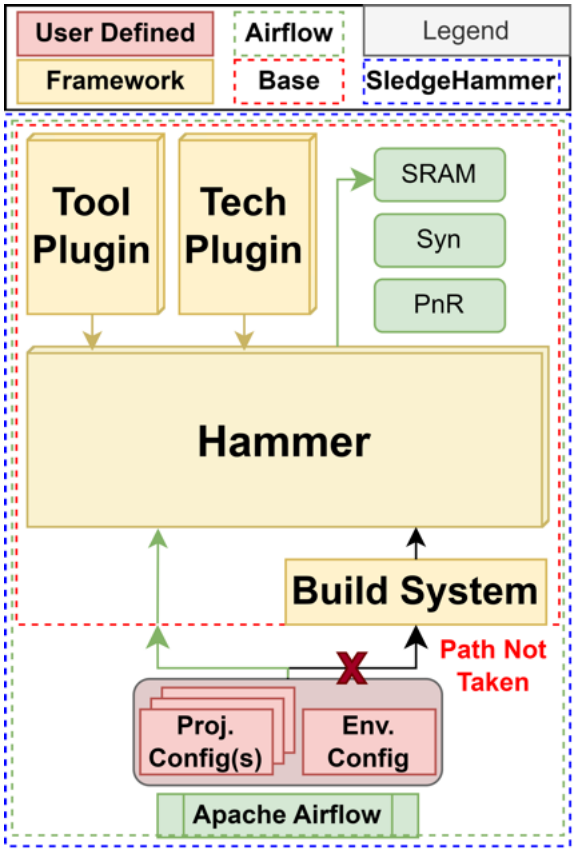

Fig. 2. SledgeHammer architecture.

Beyond improved dependency handling, SledgeHammer introduces a wide range of features that directly benefit the physical design process, including:

1) Modular dependency management
2) Flow visualization
3) Collaborative VLSI workspace
4) Graphical user interface
5) Task scheduling & automation
6) Web server hosting and cloud deployment
7) Improved debugging via parseable logs
8) Metric tracking and flow monitoring
9) Improved checkpointing capabilities

C. Running Flows

SledgeHammer supports flow execution through both a command-line interface (CLI) and a GUI provided by the integrated Airflow web server. Upon hosting a web server, users gain access to a centralized dashboard that aggregates all shared flows available within their workspace. This facilitates collaborative flow management, allowing users to run their own workflows and interact with those created by others.

For each flow instance, SledgeHammer constructs a visual representation of the DAG-defined dependencies, with individual nodes corresponding to specific stages of the physical design process (Appendix A). This gives users the flexibility to adjust the order of steps, provided that there are no conflicting dependencies, to better align the flow with their design intent. Customizable decision logic can also be implemented to provide even greater control over the behavior of individual physical design steps.

SledgeHammer pairs visualization with detailed, parseable logs for each flow step, offering improved visibility and ease of debugging. In addition to logs, SledgeHammer tracks metrics on task and flow performance, including execution time, success rates, failure trends, and other indicators that can help identify bottlenecks in the design.

As multiple users can be added to a given workspace, productivity improves when working on joint designs (Fig. 3). A potential use case involves the following:

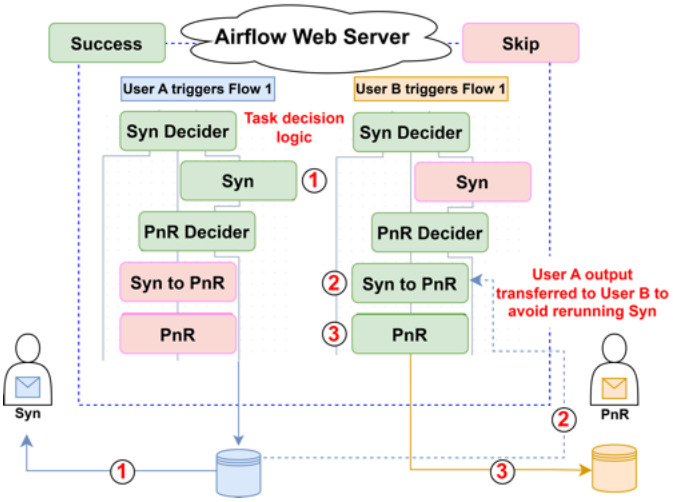

Fig. 3. Collaborative VLSI workspace.

1) User A completes synthesis and receives the output in their build directory.
2) User B, rather than waiting to rerun synthesis, uses User A's results as input for place and route.
3) User B completes place and route and now receives the output in their build directory.

SledgeHammer's approach towards encoding essential PD specifications offers significant efficiency gains, while simultaneously reducing redundancy and promoting collaboration across teams.

III. Physical Design Evaluations

A. Chipyard Integration

SledgeHammer aims to retain the main strengths of Hammer, including its support for large-scale SoC design, while resolving prior limitations and incorporating new capabilities to address the growing complexity of the chip design process. An integral tool that has played a key role in Hammer's development is Chipyard, an open-source SoC framework that enables end-to-end design and integration into the already existing Hammer system [7]. Through integration with Chipyard, SledgeHammer's effectiveness across various designs and process technologies has been evaluated.

B. Physical Design Effectiveness

Chipyard has enabled SledgeHammer to validate its viability across a diverse set of designs and technology nodes, ranging from 130nm to 16nm. A fully-featured design evaluated was **Rocket** (Fig. 4), a compact 350 um x 200 um five-stage RISC-V scalar core supporting RV32G and RV64G [8]. Throughout the design effort, we took advantage of SledgeHammer's refined flow capabilities, including its collaborative web server and flexible dependency management, which contributed to a reduction in overall PD time. As we continue to refine the interface between Chipyard and SledgeHammer, larger SoCs will become easier to integrate into the tool. Looking ahead, we aim to continue advancing SledgeHammer's capabilities to support the demands of scaling to smaller technology nodes.

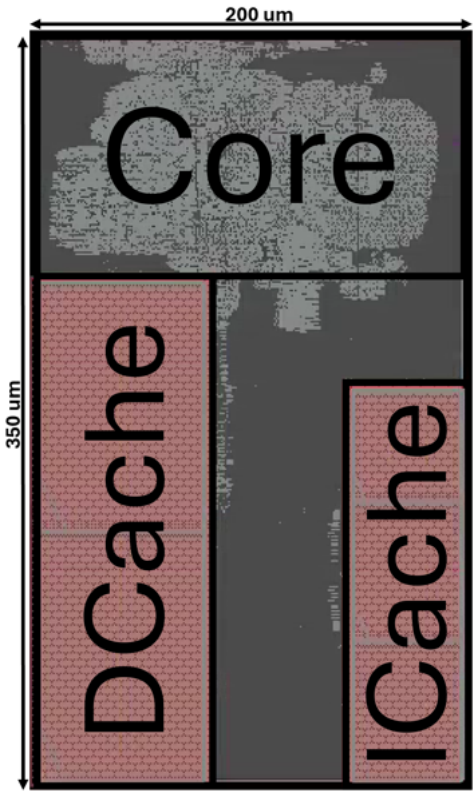

Fig. 4. Rocket: 350 um x 200 um, 16nm technology node.

By integrating a more flexible hierarchical mode that extends one of Hammer's key functionalities, SledgeHammer enables the efficient management of complex SoCs through iterative PD of individual blocks followed by top-level integration (Appendix A). SledgeHammer's advanced orchestration capabilities provide users with the flexibility to run full end-to-end flows across all blocks, target individual blocks for standalone execution, or modify the execution order of PD stages through its dependency management system.

To assess its effectiveness, we evaluated the tool on a 4 mm x 4 mm chiplet with UCIe interfaces, incurring negligible overhead from the additional orchestration extensions (Fig. 5). When taking the Rocket block from RTL to GDS within the SoC, we found an average queue time of the PD stages to be 6 seconds, which accounted for 0.27% of the entire execution time across synthesis & place and route. SledgeHammer's flexible dependency management was also observed to significantly reduce redundant execution, saving up to 22% in runtime during Rocket parameter sweeps, compared to Hammer, with even greater savings realized as the number of reruns increased.

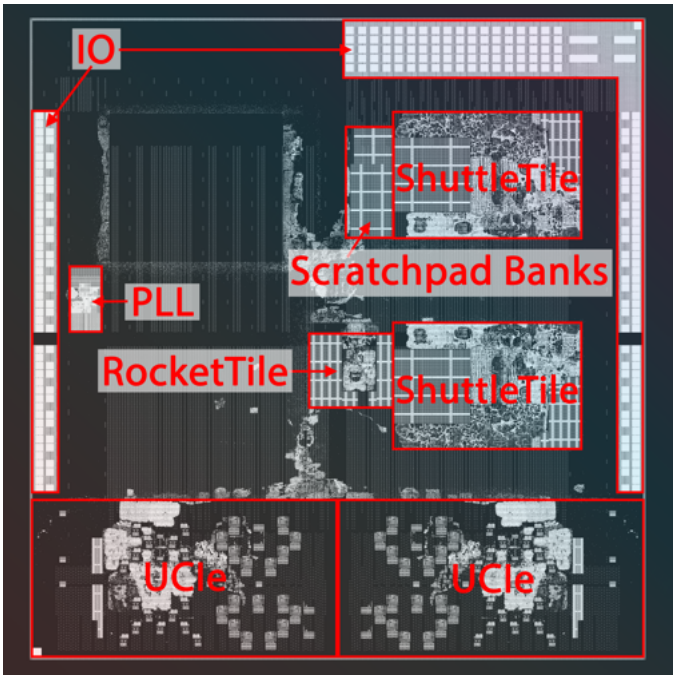

Fig. 5. Chiplet with UCIe interfaces: 4 mm x 4 mm, 16nm technology node.

IV. Future Work

The continued development of SledgeHammer presents many opportunities to extend its capabilities as a robust tool for modern physical design. Current versions of SledgeHammer require added support for DAG regeneration and conditional flow logic. Additionally, as SoCs grow in scale and technology nodes continue to shrink, the need for finer granularity in flow modularity will become more critical. By targeting the generated TCL scripts instead of solely higher abstraction levels, SledgeHammer enables finer decomposition of the PD process into smaller, more modular stages. With finer stage granularity and increasing design complexity, effective checkpointing will be critical to preserving efficiency and accelerating iteration during development.

While SledgeHammer's web server is currently hosted locally, Airflow's native support for cloud deployment paves the way for future integration with cloud services, opening the door to greater scalability and remote accessibility, while also addressing potential security risks associated with distributed workflows. In addition, it provides greater flexibility for scheduling design runs in the future, which is especially valuable for batch runs across a wide range of design parameters, reducing the need for manual user intervention.

Enhancing collaboration remains our focus as we further develop the SledgeHammer build system. These improvements will simplify collaboration by allowing sharing of flows and files to reduce overall design time. We are also investigating mechanisms for storing prior flow runs to allow users to revisit and refine previous iterations. Shared flows and configurations can be observed by collaborators in the workspace, with metadata tracked across inputs to ensure consistency in design files. Additionally, incorporating a more advanced executor into the web server could enable greater forms of parallelism and allow simultaneous execution on larger volumes of tasks and flows.

V. Conclusion

This work highlights opportunities to increase design efficiency through the integration of workflow orchestration into the physical design process. By integrating Apache Airflow into Hammer's architecture, SledgeHammer arose as a collaborative VLSI workspace, enhancing dependency management, enabling flow visualization, and reducing end-to-end design time. Upon the many changes integrated into SledgeHammer, new avenues can be pursued to improve scalability in VLSI as the tool continues to mature. These avenues include advancing cloud integration, enabling the design of more intricate SoCs, and ensuring the reproducibility of PD flows. With SledgeHammer, we lay the groundwork for a new generation of PD tools, ones that promote collaboration, scalability, and readiness to meet the challenges of tomorrow's most demanding chip designs.

Acknowledgment

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 2146752. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This prototype technology was partially supported by the Microelectronics Commons Program, a DoD initiative, under award number N00164-23-9-G057.

REFERENCES

[1] A. Carsello, J. Thomas, A. Nayak, P.-H. Chen, M. Horowitz, P. Raina, and C. Torng, "Enabling reusable physical design flows with modular flow generators," 2021. [Online]. Available: https://arxiv.org/abs/2111.14535

[2] E. Wang, C. Schmidt, A. Izraelevitz, J. Wright, B. Nikolić, E. Alon, and J. Bachrach, "A methodology for reusable physical design," in *Proceedings of the 21st International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2020, pp. 243–249.

[3] H. Liew, D. Grubb, J. Wright, C. Schmidt, N. Krzysztofowicz, A. Izraelevitz, E. Wang, K. Asanović, J. Bachrach, and B. Nikolić, "Hammer: a modular and reusable physical design flow tool: Invited," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*. New York, NY, USA: ACM, 2022, pp. 1335–1338. [Online]. Available: https://doi.org/10.1145/3489517.3530672

[4] A. Carsello, J. Thomas, A. Nayak, P.-H. Chen, M. Horowitz, P. Raina, and C. Torng, "mflowgen: a modular flow generator and ecosystem for community-driven physical design: invited," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1339–1342. [Online]. Available: https://doi.org/10.1145/3489517.3530633

[5] A. Olofsson, W. Ransohoff, and N. Moroze, "A distributed approach to silicon compilation: invited," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1343–1346. [Online]. Available: https://doi.org/10.1145/3489517.3530673

[6] J. Yasmin, J. Wang, Y. Tian, and B. Adams, "An empirical study of developers' challenges in implementing workflows as code: A case study on apache airflow," 2024. [Online]. Available: https://arxiv.org/abs/2406.00180

[7] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanović, and B. Nikolić, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, 2020.

[8] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The rocket chip generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr. 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html

APPENDIX

Using the SledgeHammer GUI, we designed a chiplet with UCIe interfaces while taking advantage of the tool's new features. This included improved control over dependent hierarchical blocks, visualizing the PD flow through the generated nodal graph, and analyzing relevant metric data corresponding to each stage of the PD process. Figures 6-7 provide a view of the SledgeHammer interface.

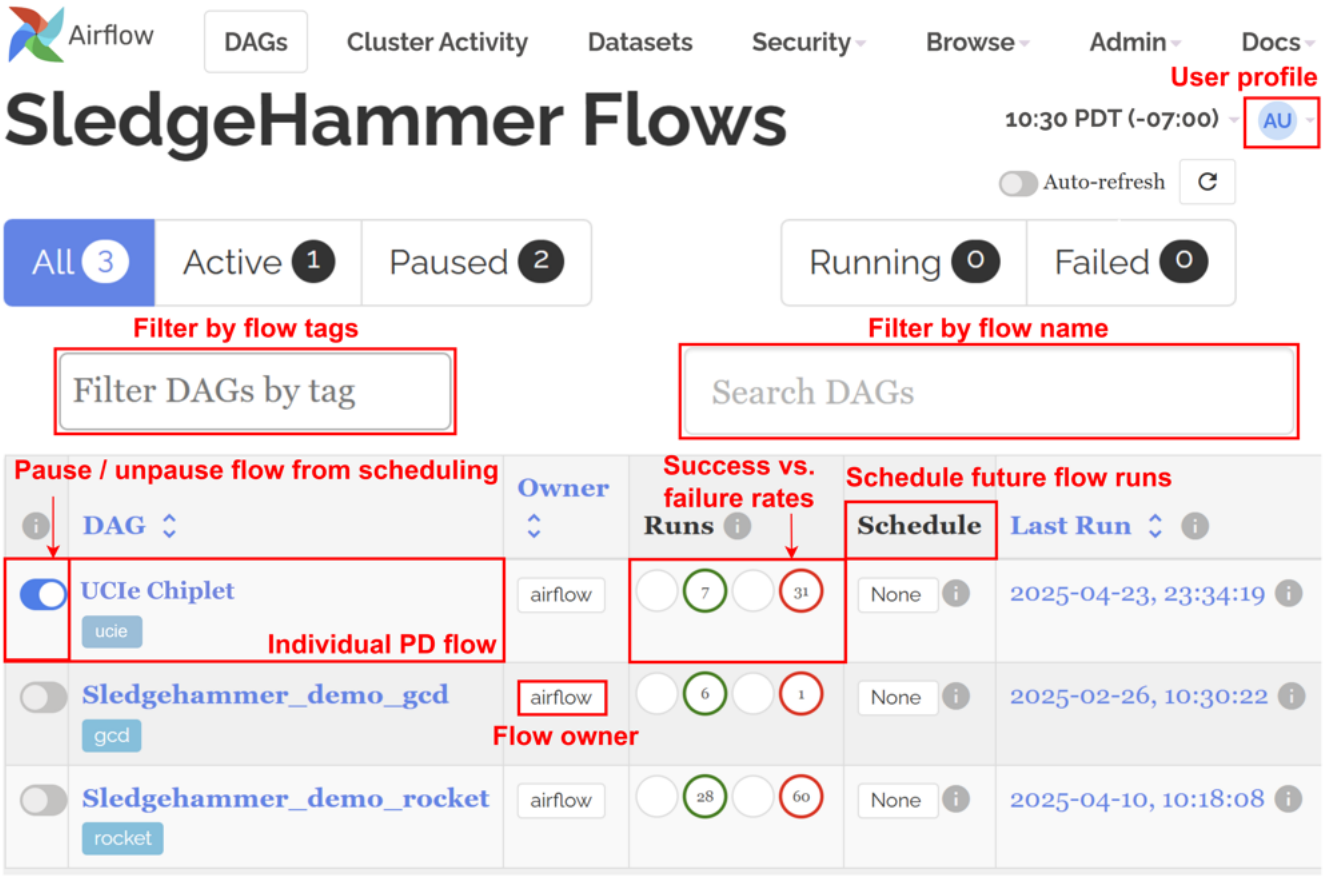

Fig. 6. SledgeHammer GUI (Appendix A).

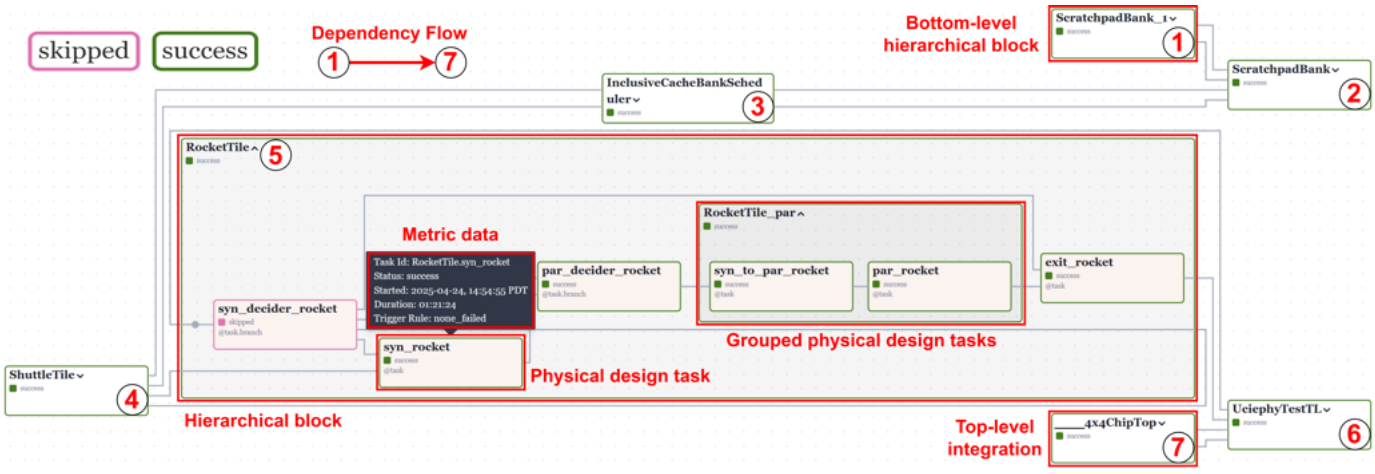

Fig. 7. Hierarchical physical design flow of Rocket block within chiplet featuring UCIe interfaces via SledgeHammer (Appendix A).