Breaking Down Physical Design Barriers with Open and Agile Flow Tools (A Perspective from DAC 2022)

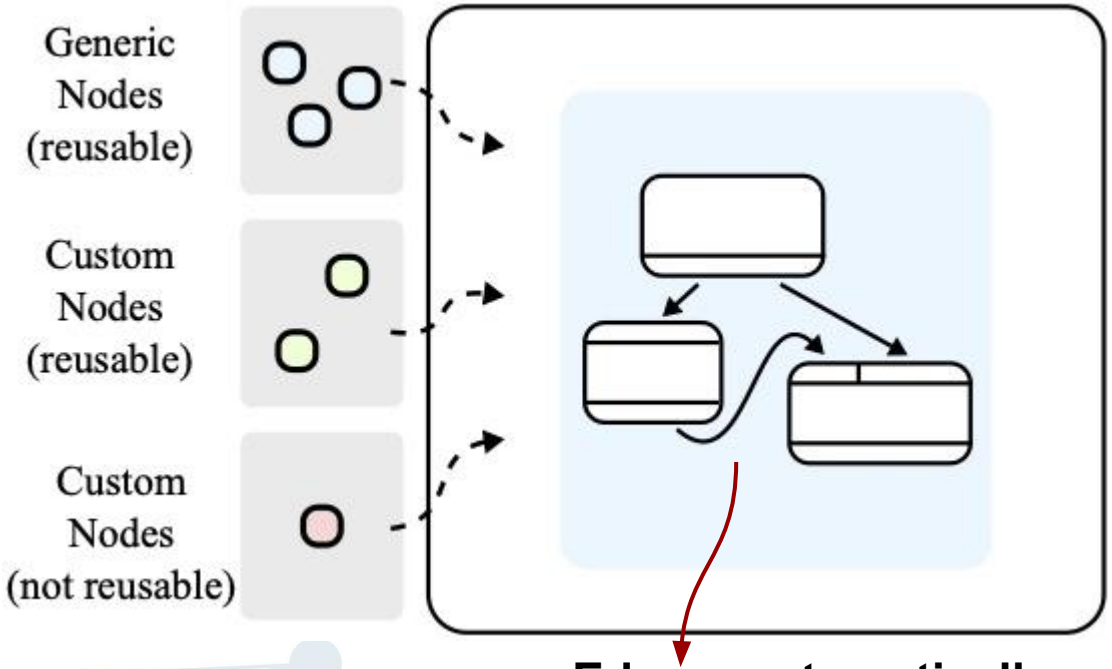
Christopher Torng

Stanford AHA! Agile Hardware Center

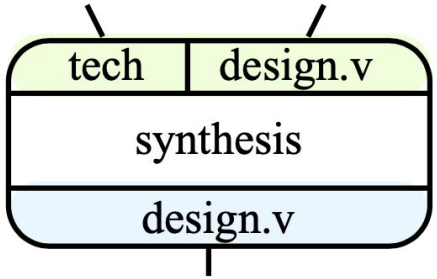
Stanford University

August 25, 2022

The High-Level Bit of Modular Flow Generators



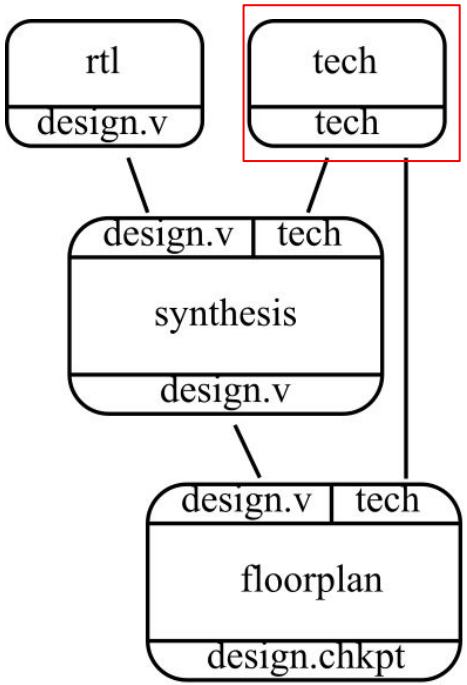
Graph View



Function signature with file-based inputs and outputs

Edges: automatically generates code that moves files to next island

Simple Teaching Graph



Flexibility to assemble

- Basic teaching graphs
- Complete tapeout-ready graphs

Technical challenge

- Achieve extreme reuse (90%+)

Technical questions

- Reuse code that is technology or design specific?
- Composability

Community-driven physical design

DAC 2022 Special Session

Three perspectives: Challenges and opportunities in open and agile physical design

mflowgen: A Modular Flow Generator and Ecosystem for Community-Driven Physical Design

Christopher Torng (Stanford)

Focus:

Composability and Community-driven PD

Hammer: A Modular and Reusable Physical Design Flow Tool

Borivoje Nikolic (Berkeley)

Focus:

Architect's interface to PD

SiliconCompiler: A Distributed Approach to Silicon Compilation

Andreas Olofsson (Zero ASIC)

Focus:

Cloud

What are we agreeing on?

Physical design flows are too difficult to navigate, and it has real impacts on designer enablement

- **Students** -- workforce enablement
- **Research groups and small startups** -- innovation
- **Larger industry** -- cost and time

Shared: Making a difference in this space will mean something

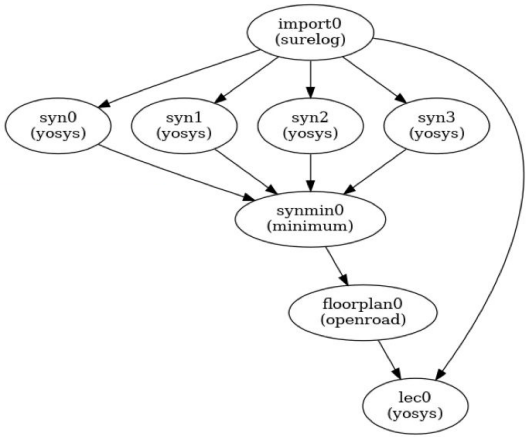
What are we agreeing on?

Making use of higher level languages

```python
import siliconcompiler
syn_np = 4
chip = siliconcompiler.Chip()

# nodes
chip.node('import', 'surelog')
for i in range(syn_np):
    chip.node('syn', 'yosys', index=i)
chip.node('synmin', 'minimum')
chip.node('floorplan', 'openroad')
chip.node('lec', 'yosys')

# edges
for i in range(syn_np):
    chip.edge('import', 'syn', head_index=i)
    chip.edge('syn', 'synmin', tail_index=i)
chip.edge('synmin', 'floorplan')
chip.edge('floorplan', 'lec')
chip.edge('import', 'lec')
chip.write_flowgraph("pattern_general.png")
```



1. Encapsulate functionality of underlying PD tools
2. Add a layer *above* the captured functionality
3. New features operate at this higher level

SiliconCompiler flow graph assembly

What are we finding differently interesting?

- We have identified **different challenges with uniquely new value**
- Technical challenges in each one, it is not just culture changes
  - mflowgen: These have to do with composability and extreme reuse
  - SIliconCompiler: These are about representations and infrastructure necessary to enable local vs cloud
  - Hammer: It is the integration into Chisel / FIRRTL and potential of generators

What are we finding differently interesting?

How is each party looking at scaling challenges?

- mflowgen and Hammer have both established user bases
- The **benefits can be seen** in practice -- do these benefits scale?
  - mflowgen
    - EE272 taught by Priyanka, taping out in SKY130 (open PDK)
    - Research groups across Stanford, Cornell, Georgia Tech, USC, Germany
  - Hammer
    - 30 undergrads?
    - Many more examples
- SiliconCompiler is directly addressing scale to some degree

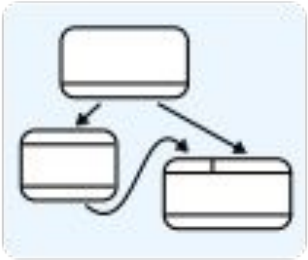
Some more observations and shared challenges

- **Progression** of this small field happened
  - Zero ASIC learned from mflowgen and hammer
- YAML/JSON representations not enough
  - Move to a DSL
- Interface to the technology too ad hoc
  - **mflowgen**: "ADK" (asic design kit)
  - **Hammer**: "technology plugin"
  - **SiliconCompiler**: same idea

Breaking Down Physical Design Barriers with Open and Agile Flow Tools (A Perspective from DAC 2022)

Agile flow tools are bringing something different to how we build complex systems

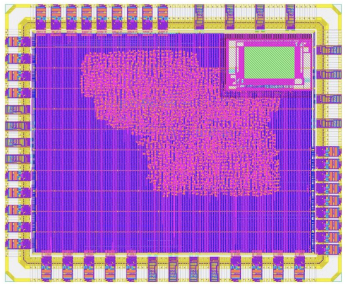
- These tools are still **early maturity**
- There is **evidence for benefits** (see chips and metrics)
- Already significant impact on **training of future workforce**



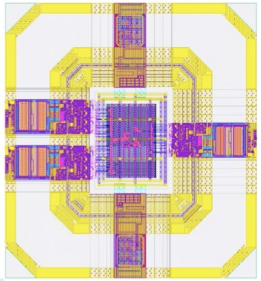
Feel free to ping me about this or to find the slides online for mflowgen, Hammer, and SiliconCompiler!

Takeaway: Accessibility of PD is expanding through agile flow tools

SiliconCompiler

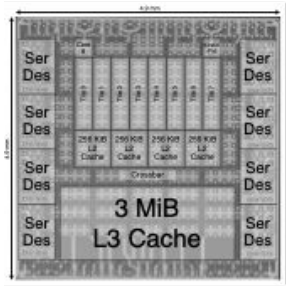


SKY130

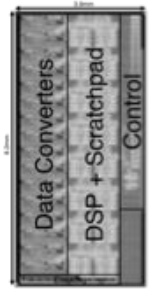


SKY130

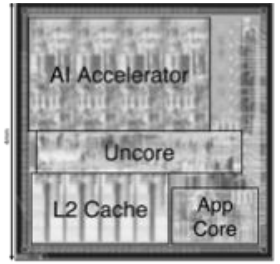
Hammer (11+ fabricated chips, 4 classes)



16nm

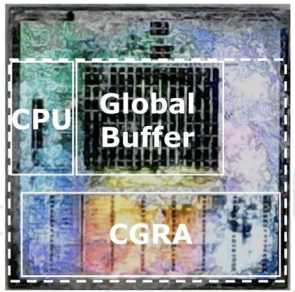


22nm

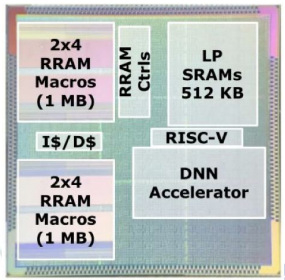


12nm

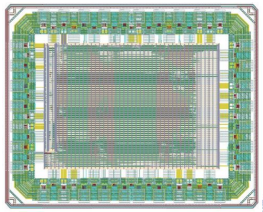
mflowgen (12+ fabricated chips, 3 classes)



16nm



40nm



28nm

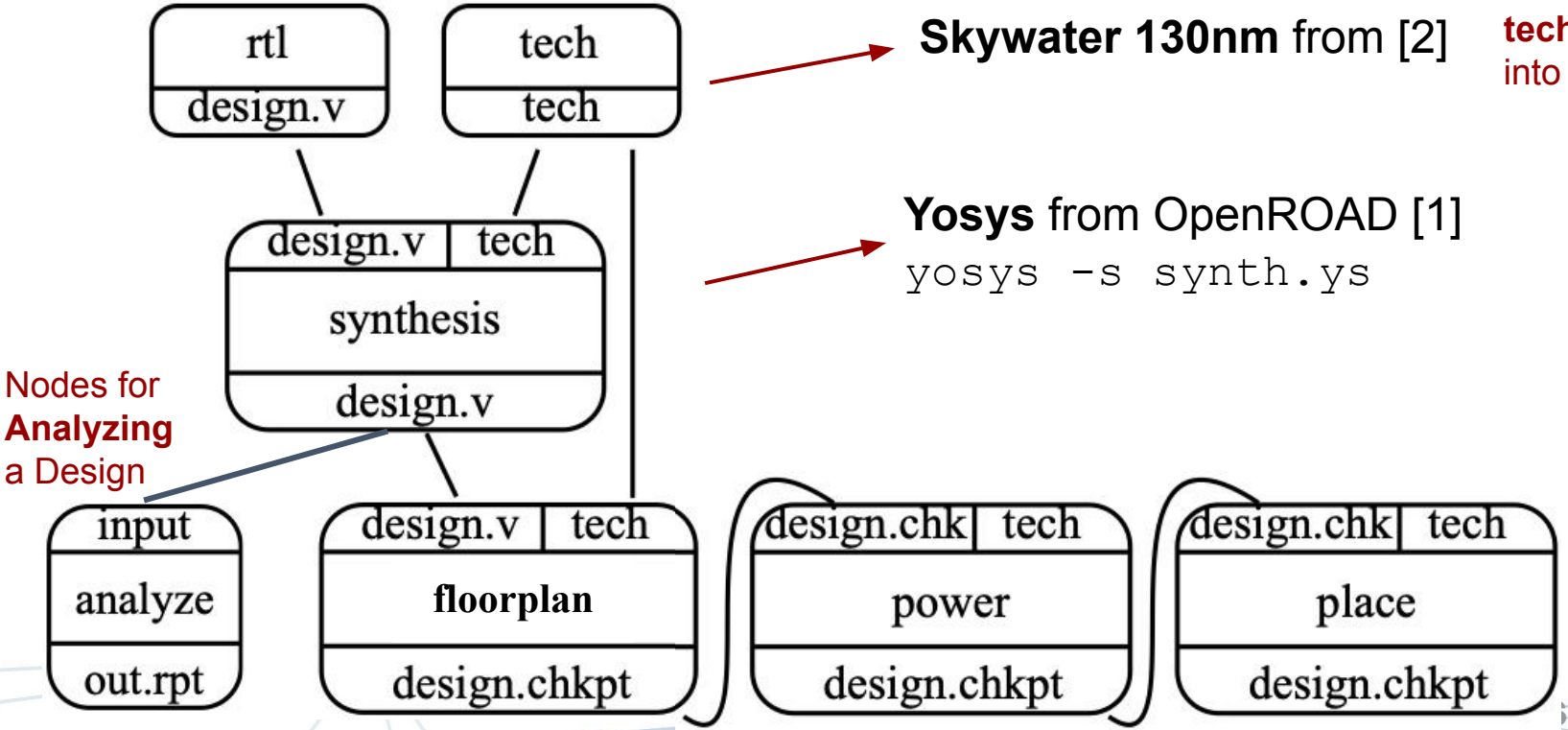
- **Students**: Stanford, Berkeley, Cornell, Georgia Tech, USC
- **Industry**: SiliconCompiler investment "build to scale" and "plan for 10+ years"

Backup Slides

Example teaching graph

[1] https://theopenroadproject.org
[2] https://github.com/google/skywater-pdk

Other nodes are
Static Vendor Packages



Skywater 130nm from [2]

Abstract the **technology files** into one node

Yosys from OpenROAD [1]

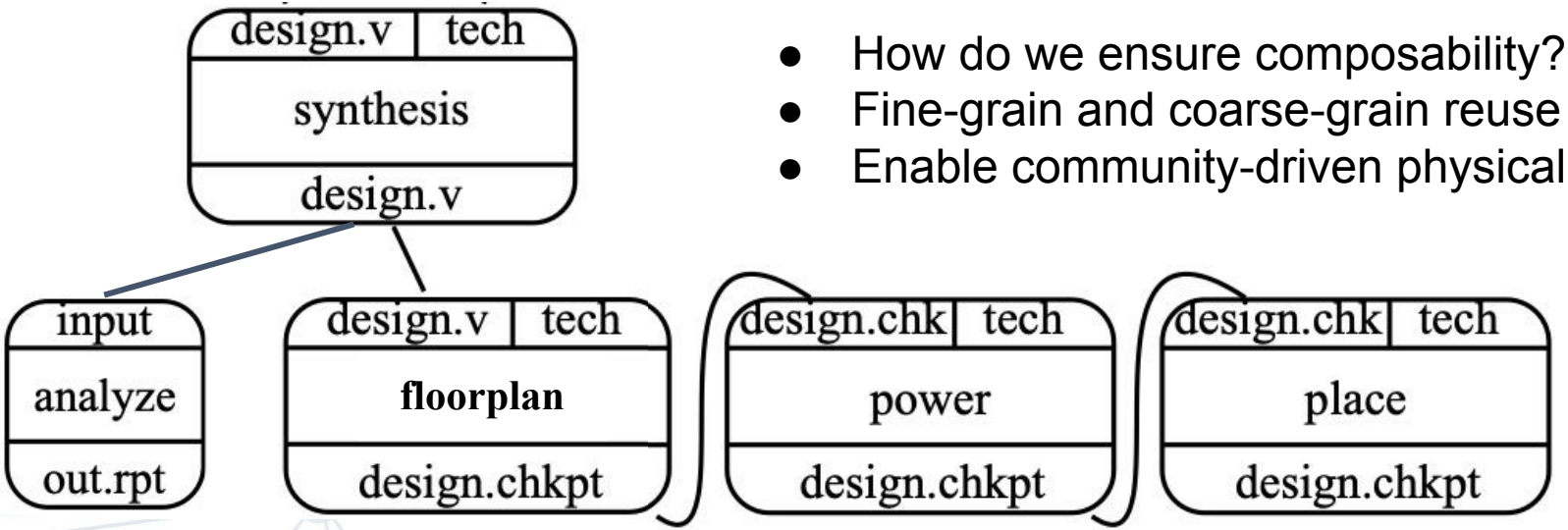
```
yosys -s synth.ys
```

Nodes for **Analyzing** a Design

Nodes for **Transforming** a Design

Takeaway: Open and agile flow tools are broadening to address different problems

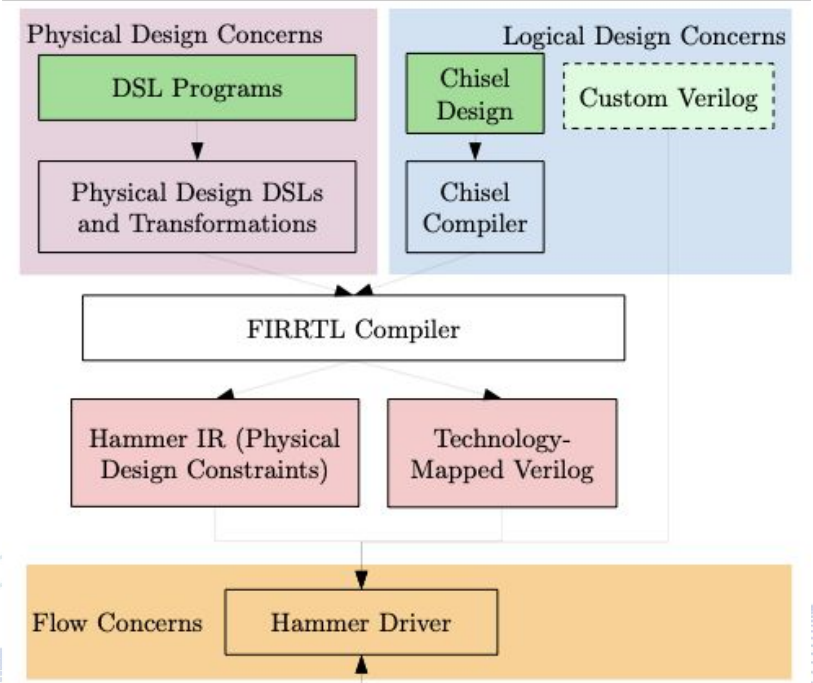
mflowgen | **Hammer** | **SiliconCompiler**



- How do we ensure composability?
- Fine-grain and coarse-grain reuse
- Enable community-driven physical design

Takeaway: Open and agile flow tools are broadening to address different problems

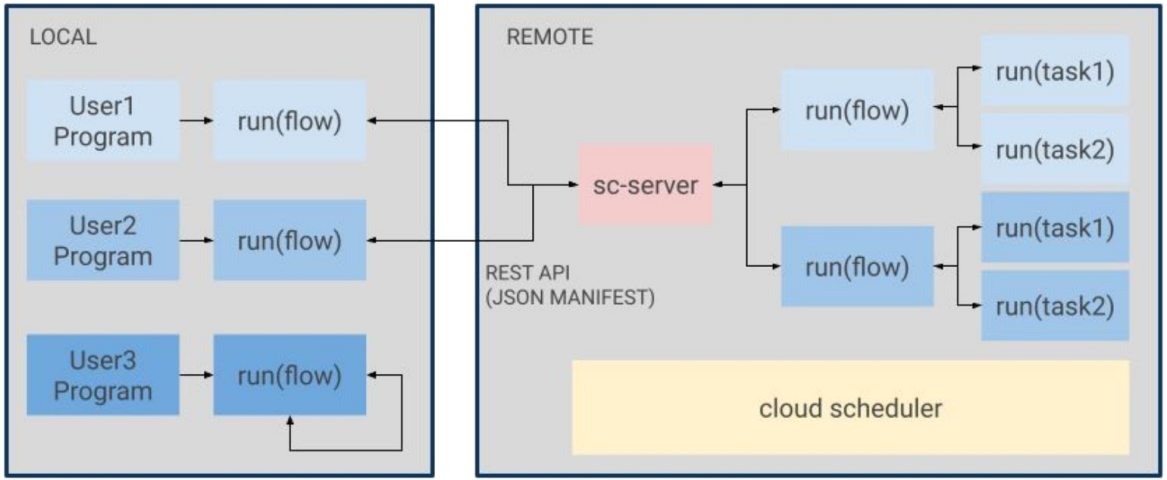
mflowgen | **Hammer** | **SiliconCompiler**



- How can we accelerate architecture design-space exploration?
- Integrate PD decisions into the RTL (Chisel) and IR (FIRRTL) levels
- and generating the PD flow

Takeaway: Open and agile flow tools are broadening to address different problems

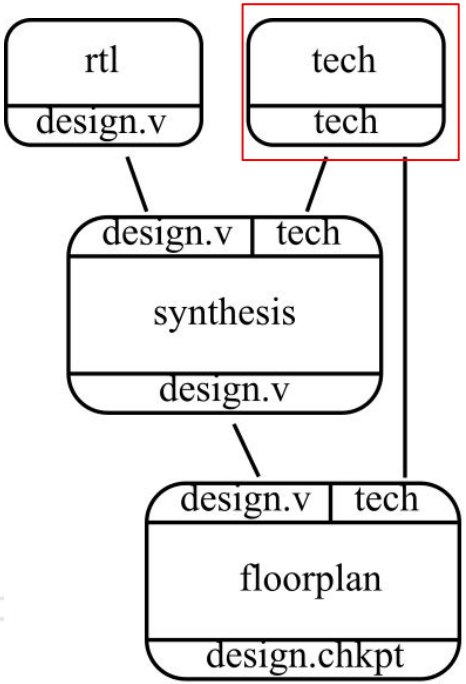
mflowgen | **Hammer** | **SiliconCompiler**



- How to design an agile flow tool for the cloud?
- Platform-agnostic **scalable execution model**
- Zero-install client/server remote REST API

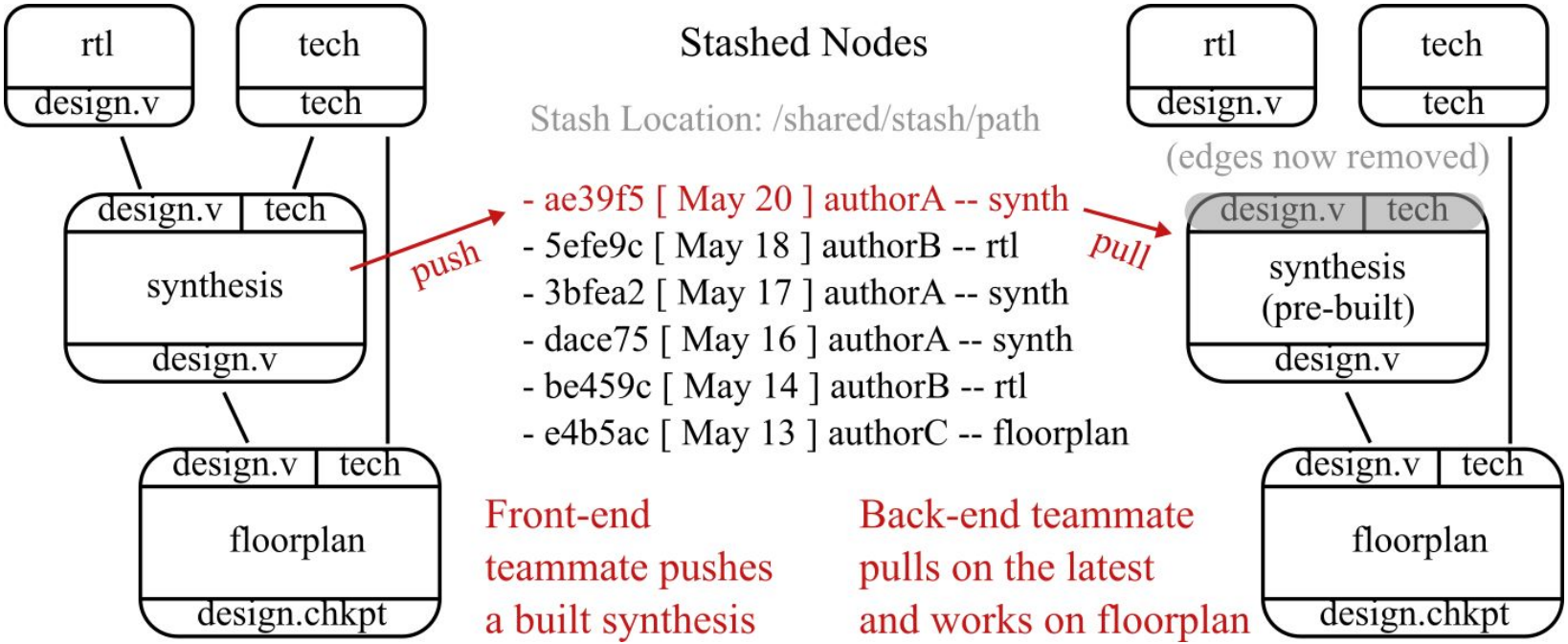
What's next for agile flow tools?

1. Common theme of a reusable interface to the technology



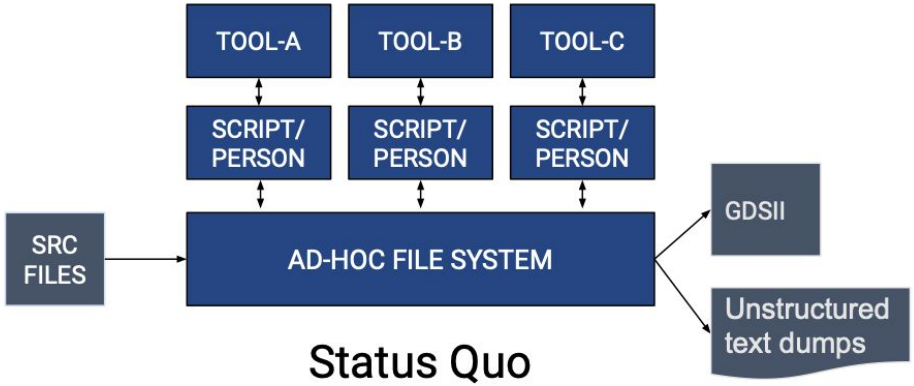
What's next for agile flow tools?

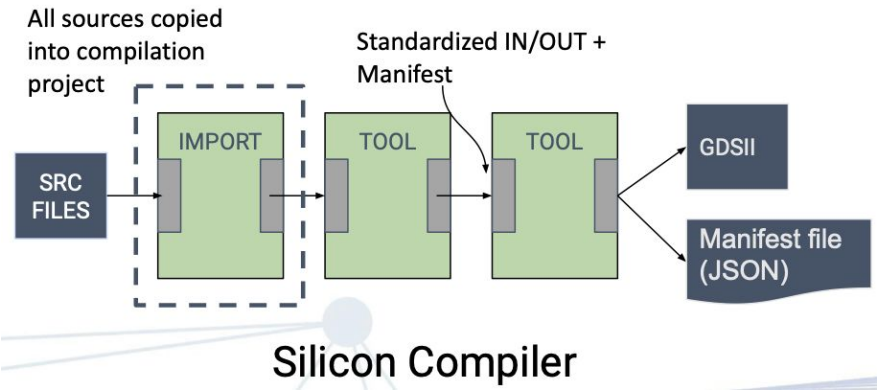
2. Converge on what is useful -- mflowgen and its stash feature



What's next for agile flow tools?

2. Converging on what is useful -- SiliconCompiler and its manifest





What's next for agile flow tools?

3. Learning to scale with complexity

Can these agile flow tools support scaling to 100+ corners?	How to manage reuse across hundreds of chips in different technologies?	How to approach designer enablement for both experts (scale) and novices (basic)?