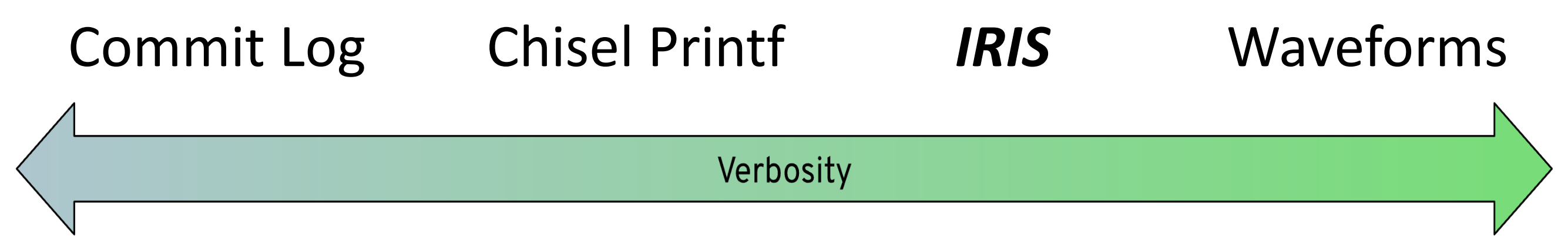
Iris: Microarchitectural Event Database

Nicolas Castaneda, Kevin He, Jerry Zhao

Motivation

Provide a general API for extracting micro-architectural events and data in Chipyard

- Instruction commit logs are too coarse-grained
- Waveforms are too fine-grained, slows debugging
- Event Logs provide a balance of visibility and useability
- Identify tradeoffs and performance bottlenecks
- Output logs can be easily post-processed for further analysis and debugging
- Quickly view dependencies without detailed knowledge of RTL design signals
- Educational tool for understanding Chipyard microarchitectures on real workloads



Tool Flow

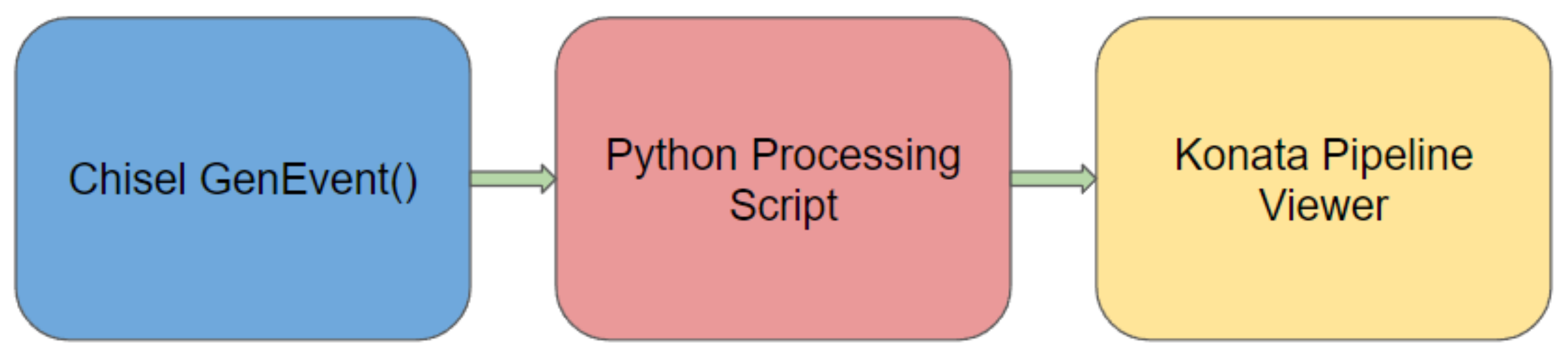
Annotate RTL with GenEvent tags

- Tag events of interest: reqs, resps, valids, hits, misses, pipeline stages, etc. with GenEvent module
- Pass GenEvent output parent/child tags through logic to connect GenEvent instances OR specify common instruction ID such as PC
- Extract logic signals: addresses, instructions, register values, IDs
- Output logs with DPI-C or stdout in RTL-sim

Run Python Script on Generated Log

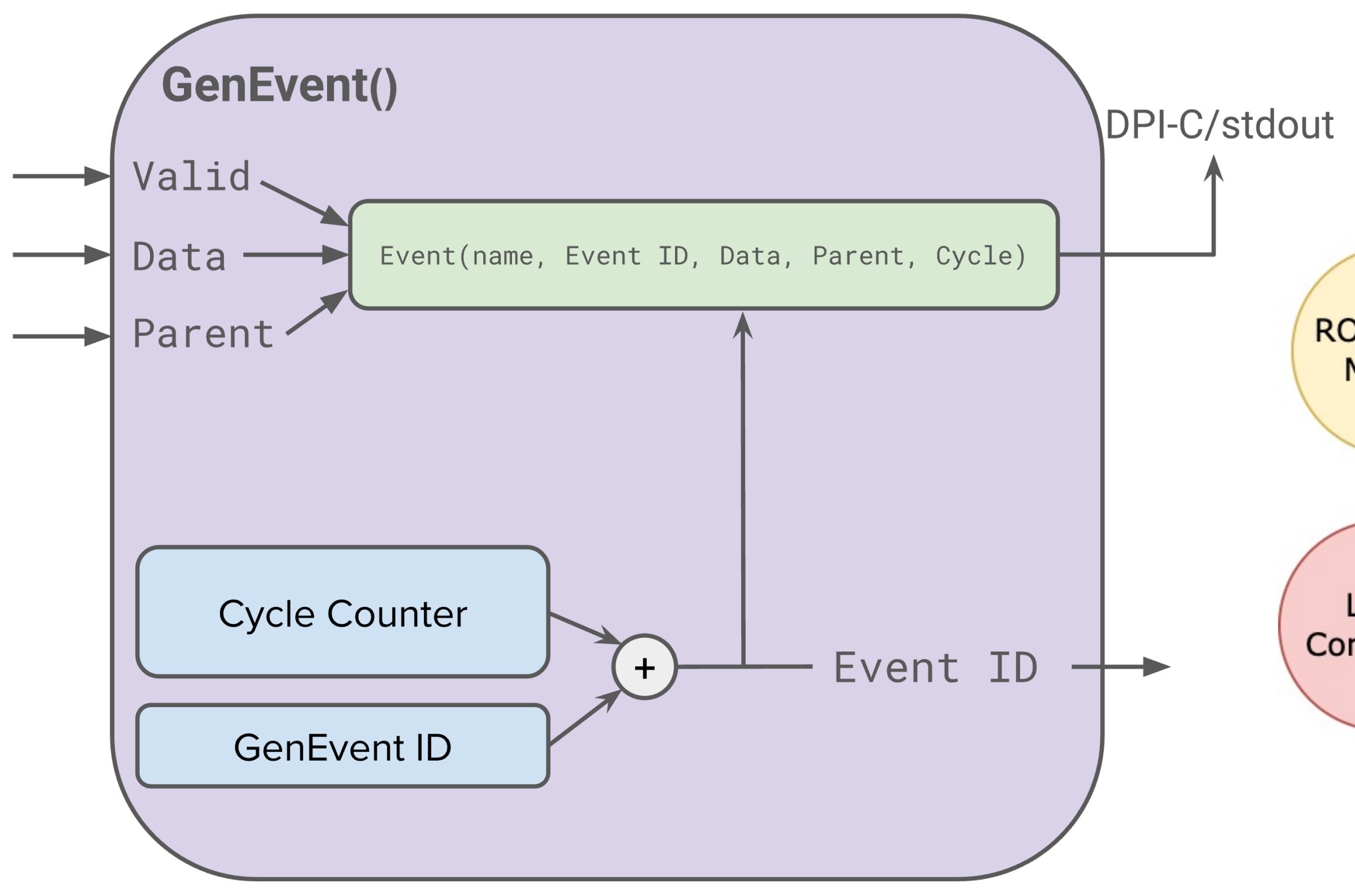
- Provide schema for particular core
- Disassembles instructions
- Generates graph of instruction events and converts instruction traces to Konata log

Visualize Using Konata

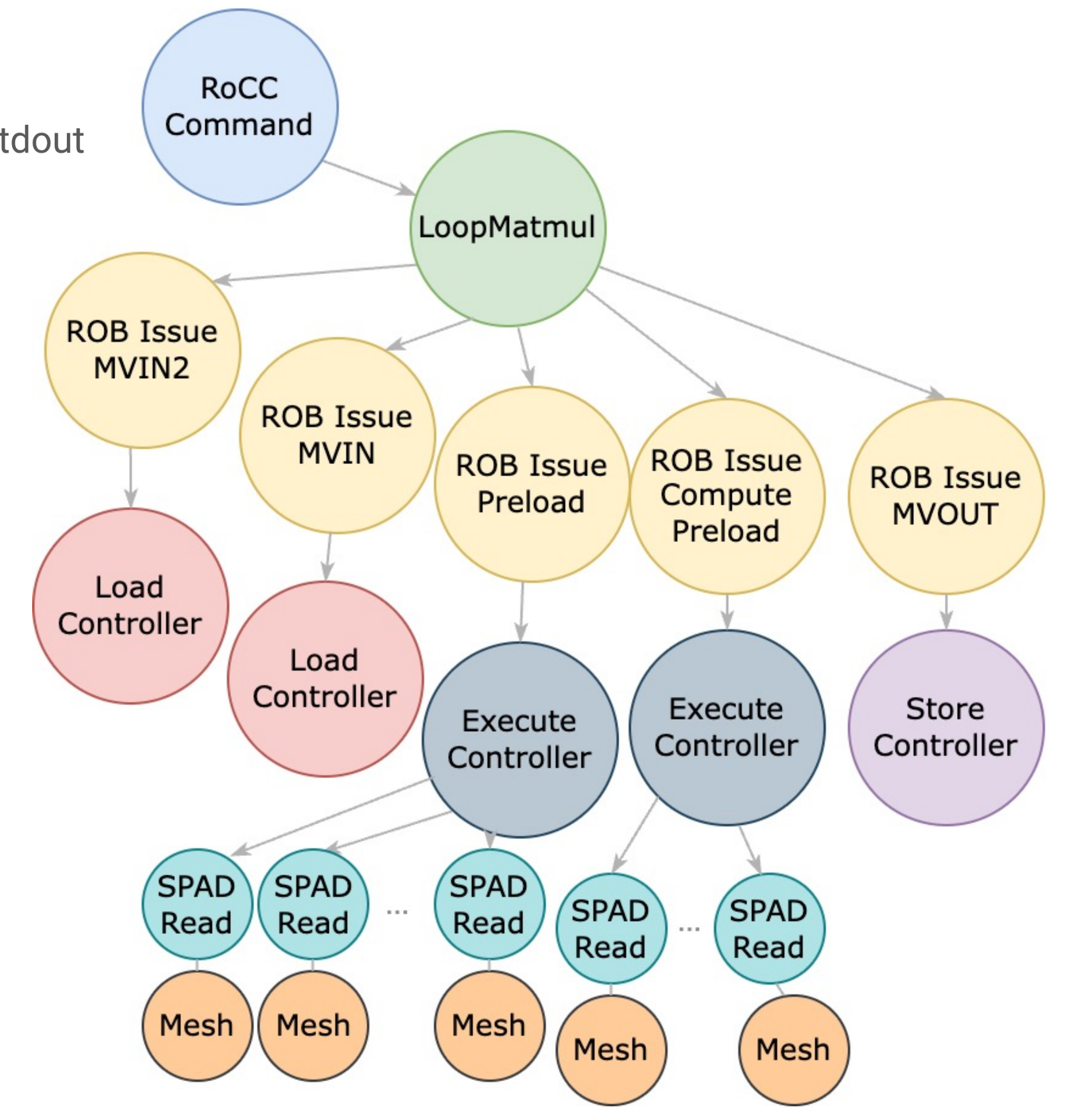


Event Graph Generation

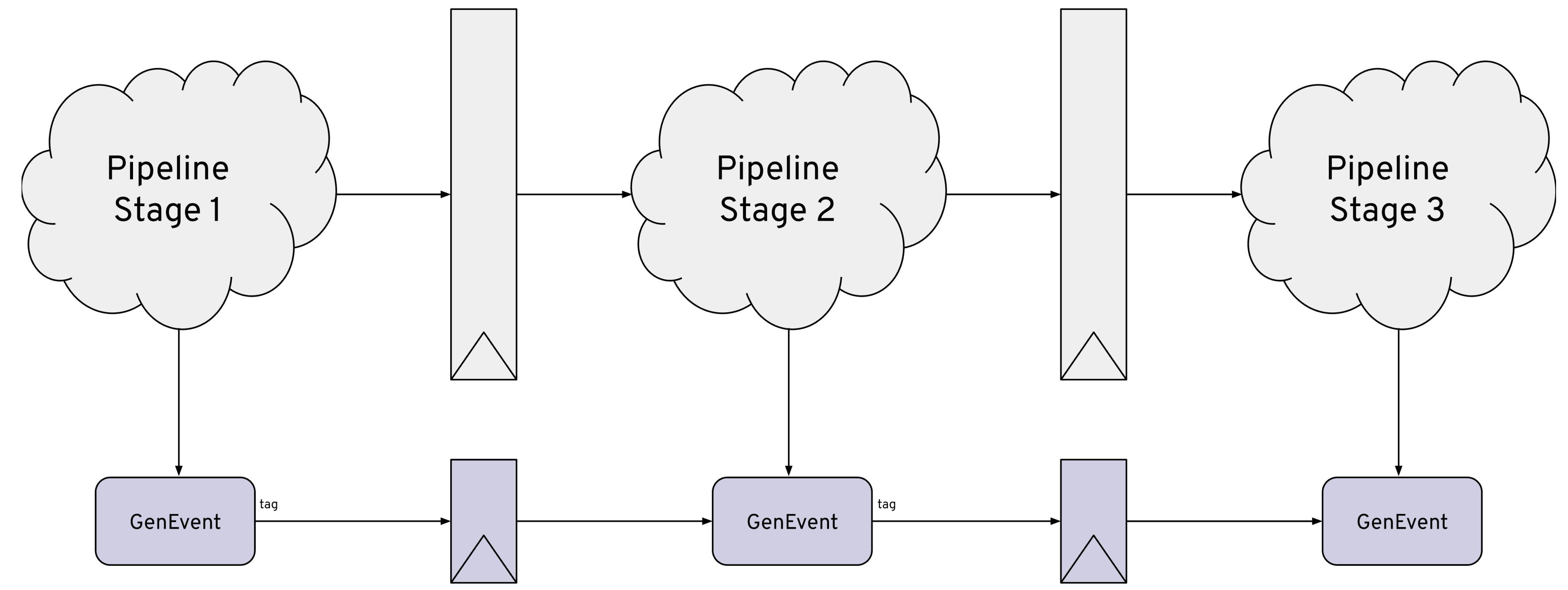
GenEvent Module



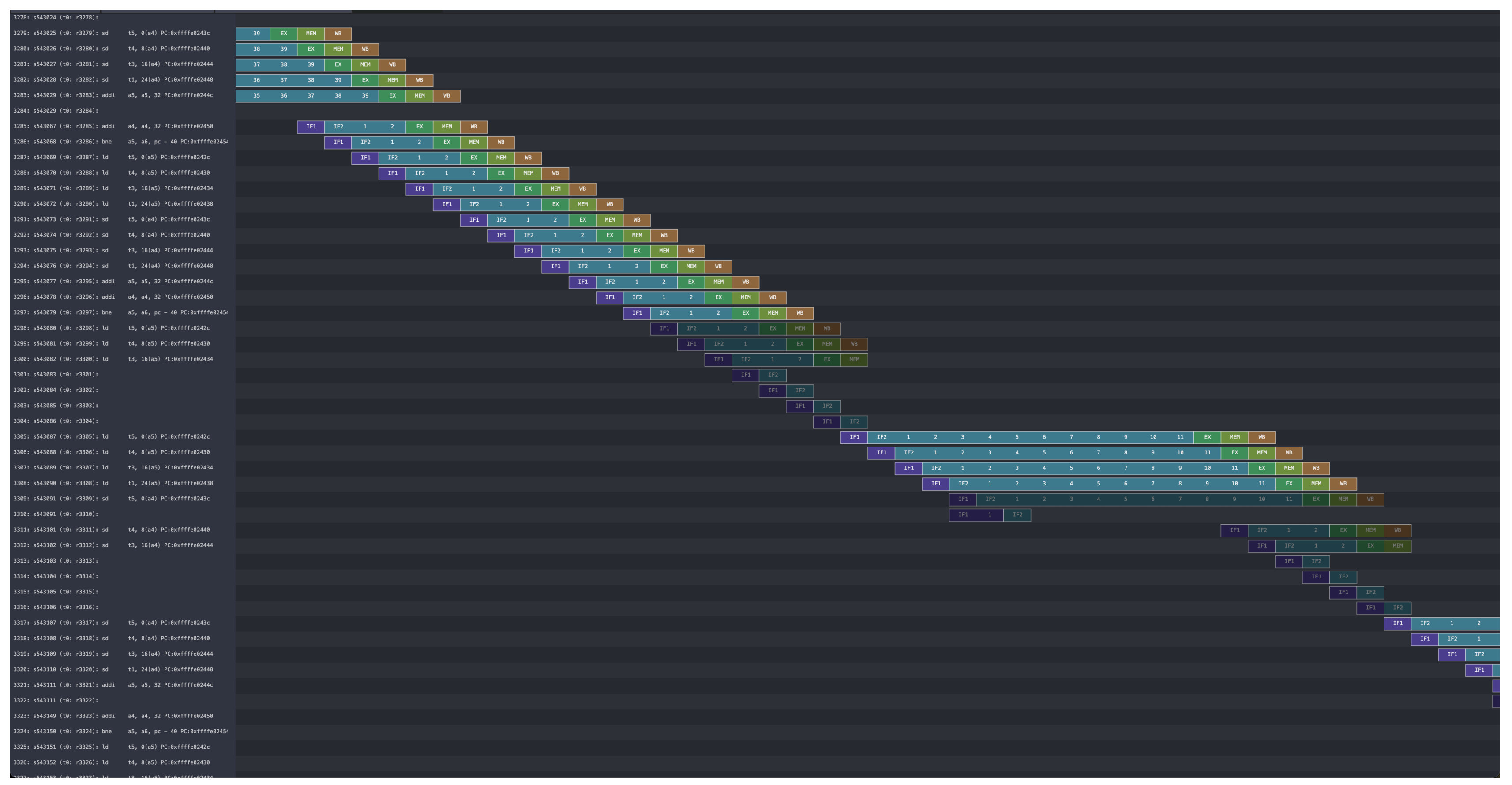
Gemmini Event Graph



3-stage CPU GenEvent Microarchitecture Annotation Example



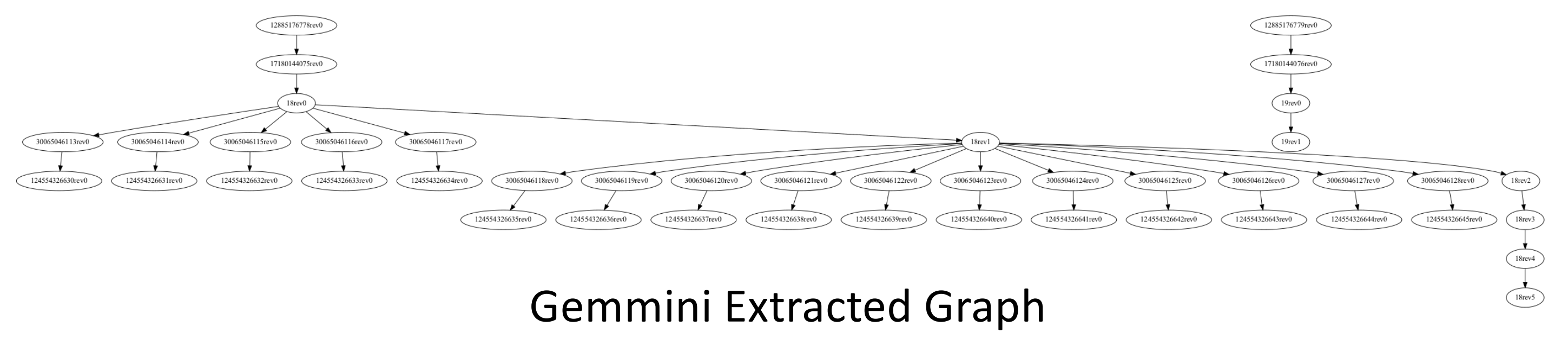
Pipeline Viewer



Konata Visualization of Rocket

Event Graph Model

- Represents the microarchitectural state as a sequence of dependent events
- Nodes represent single cycle events
- Edges represent the resolution of a hazard, allowing the subsequent event to occur
- Can easily query graph for specific information (i.e. an instruction trace)
- Each GenEvent Module has an event name, valid input, data input, and an optional Parent ID input
- When valid, GenEvent logs the data, cycle, parent ID, and event name, while outputting a unique Event ID.
- By passing Event ID tags, designers connect dependent microarchitectural events



Gemmini Extracted Graph

Implemented Cores

Rocket Core

- Integer, mul/div, and cache request/response pipelines annotated

Sodor Cores

- 1 stage, 2 stage, 5 stage, and microcoded cores annotated

Gemmini

- Load, Store, and Execution controllers, LoopMatmul and LoopConv FSMs, scratchpad reads/writes, mesh

Future Work

Annotating Additional Cores

- Saturn and BOOM

More Microarchitecture Detail

- Annotating additional latencies for debugging
- Event filtering tools, graph visualization, downstream analysis tools

https://github.com/ucb-bar/iris-event-utils
nicolas.a.castaneda@berkeley.edu, kevinjhe@berkeley.edu, jzh@berkeley.edu