SLICE

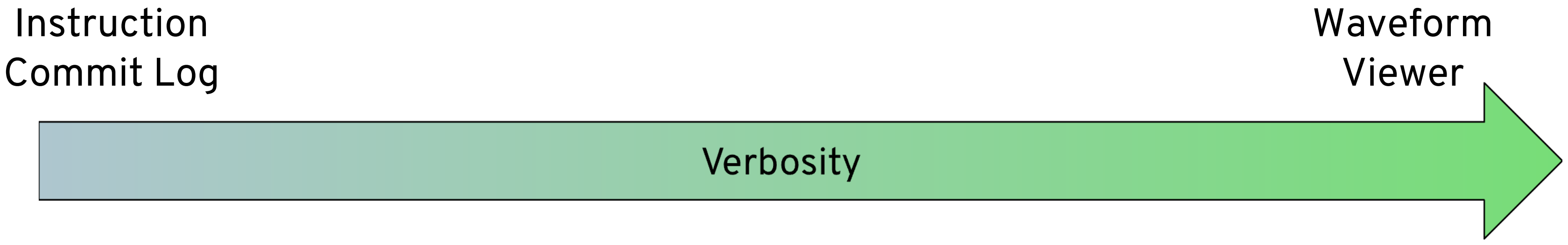
Iris: Microarchitectural Event Database

5/20/2024

Nicolas Castaneda, Kevin He, Jerry Zhao

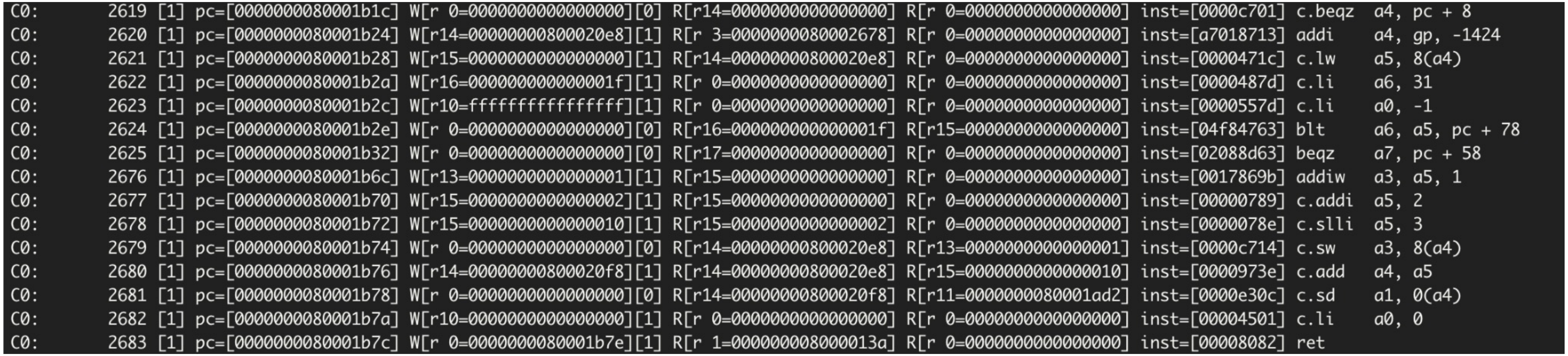
Motivation for Event Logs

- Goal: Understand RTL Behavior
  - Debug increasingly complex cores
  - Add features to complex systems and identify tradeoffs
  - Identify performance bottlenecks
  - Current solutions either fail to capture sufficient information or fail to deliver it in a comprehensible manner



Motivation for Event Logs

- Trace-like logs are too coarse-grained
  - No information regarding the relationship between multiple resident instructions and the microarchitecture
  - No way to reason about how instructions interact with the microarchitecture
  - Insufficient information regarding timing

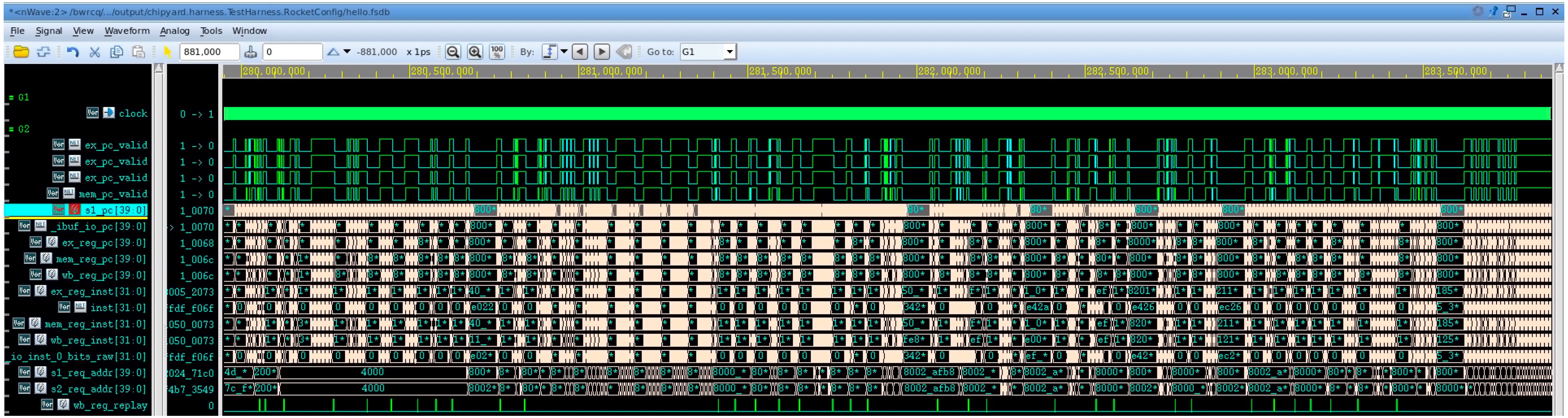


```
C0:        2619 [1] pc=[0000000080001b1c] W[r 0=0000000000000000][0] R[r14=0000000000000000] R[r 0=0000000000000000] inst=[0000c701] c.beqz  a4, pc + 8
C0:        2620 [1] pc=[0000000080001b24] W[r14=00000000800020e8][1] R[r 3=0000000080002678] R[r 0=0000000000000000] inst=[a7018713] addi    a4, gp, -1424
C0:        2621 [1] pc=[0000000080001b28] W[r15=0000000000000000][1] R[r14=00000000800020e8] R[r 0=0000000000000000] inst=[0000471c] c.lw    a5, 8(a4)
C0:        2622 [1] pc=[0000000080001b2a] W[r16=000000000000001f][1] R[r 0=0000000000000000] R[r 0=0000000000000000] inst=[0000487d] c.li    a6, 31
C0:        2623 [1] pc=[0000000080001b2c] W[r10=ffffffffffffffff][1] R[r 0=0000000000000000] R[r 0=0000000000000000] inst=[0000557d] c.li    a0, -1
C0:        2624 [1] pc=[0000000080001b2e] W[r 0=0000000000000000][0] R[r16=000000000000001f] R[r15=0000000000000000] inst=[04f84763] blt     a6, a5, pc + 78
C0:        2625 [1] pc=[0000000080001b32] W[r 0=0000000000000000][0] R[r17=0000000000000000] R[r 0=0000000000000000] inst=[02088d63] beqz    a7, pc + 58
C0:        2676 [1] pc=[0000000080001b6c] W[r13=0000000000000001][1] R[r15=0000000000000000] R[r 0=0000000000000000] inst=[0017869b] addiw   a3, a5, 1
C0:        2677 [1] pc=[0000000080001b70] W[r15=0000000000000002][1] R[r15=0000000000000000] R[r 0=0000000000000000] inst=[00000789] c.addi  a5, 2
C0:        2678 [1] pc=[0000000080001b72] W[r15=0000000000000010][1] R[r15=0000000000000002] R[r 0=0000000000000000] inst=[0000078e] c.slli  a5, 3
C0:        2679 [1] pc=[0000000080001b74] W[r 0=0000000000000000][0] R[r14=00000000800020e8] R[r13=0000000000000001] inst=[0000c714] c.sw    a3, 8(a4)
C0:        2680 [1] pc=[0000000080001b76] W[r14=00000000800020f8][1] R[r14=00000000800020e8] R[r15=0000000000000010] inst=[0000973e] c.add   a4, a5
C0:        2681 [1] pc=[0000000080001b78] W[r 0=0000000000000000][0] R[r14=00000000800020f8] R[r11=0000000080001ad2] inst=[0000e30c] c.sd    a1, 0(a4)
C0:        2682 [1] pc=[0000000080001b7a] W[r10=0000000000000000][1] R[r 0=0000000000000000] R[r 0=0000000000000000] inst=[00004501] c.li    a0, 0
C0:        2683 [1] pc=[0000000080001b7c] W[r 0=0000000080001b7e][1] R[r 1=000000008000013a] R[r 0=0000000000000000] inst=[00008082] ret
```

Rocket Instruction Commit Log

Motivation for Event Logs

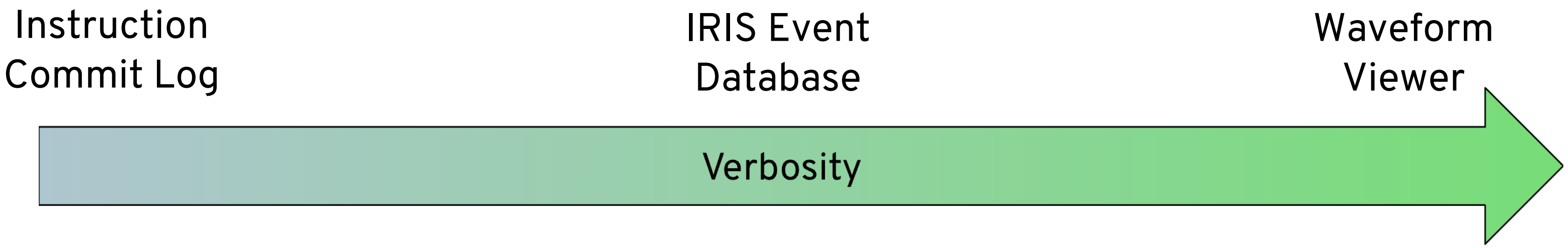
- Waveforms contain too much information
  - Value for every bit in the RTL design over millions of cycles
  - Requires extensive knowledge of microarchitecture signals
  - Time consuming for initial debugging
  - Dependency chains between signals aren't captured



Rocket Waveforms

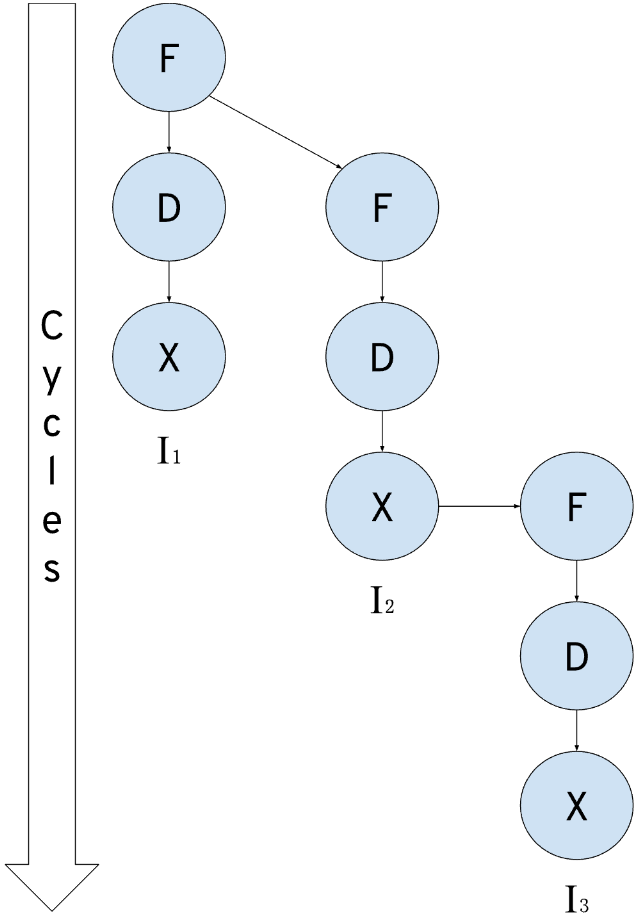
Microarchitectural Event Annotation API

- Our solution: IrisDB
  - Flexible API for extracting microarchitectural events and data in RTL
  - Middle ground between waveforms and instruction commit logs
    - Provides a good starting point for debugging
  - Outputs event log for post-processing or analysis



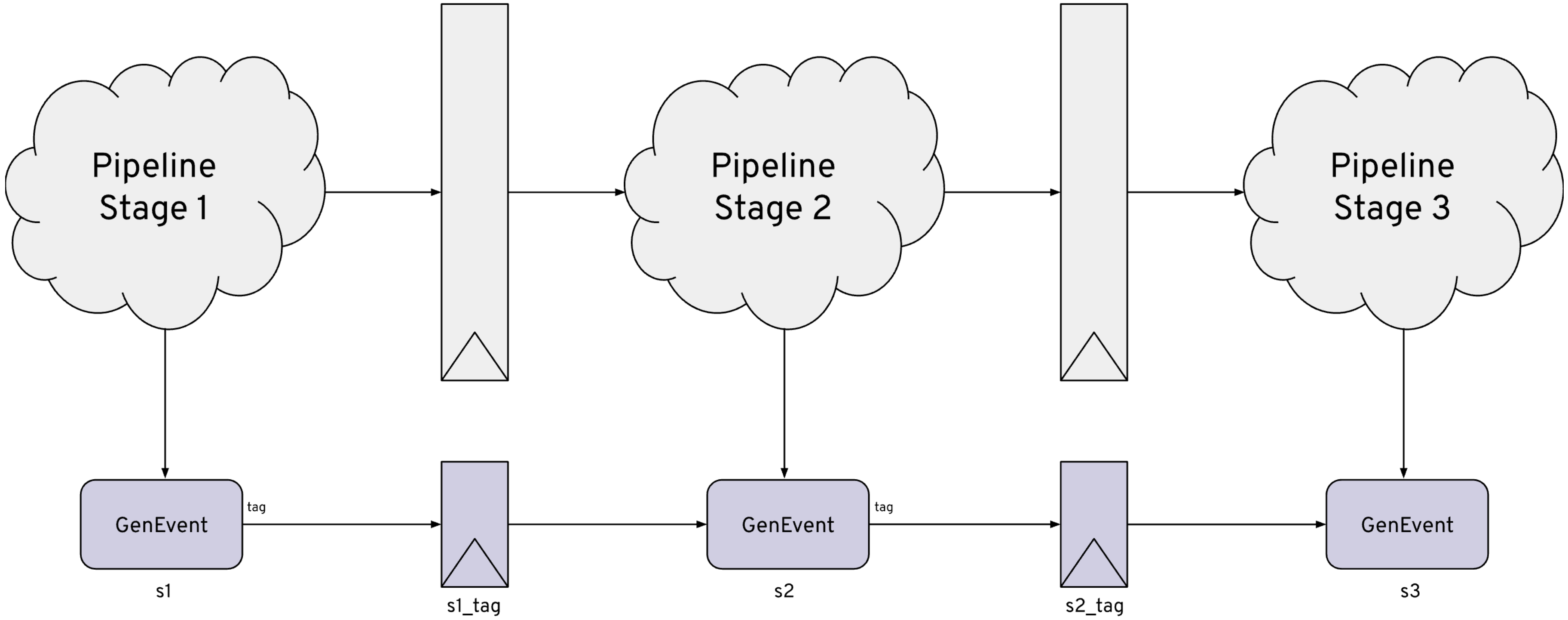
Event Graph Model

- Represents the microarchitectural state as a sequence of dependent events
  - Microarchitecture agnostic representation
  - Nodes represent single cycle events
  - Edges represent the resolution of a hazard, allowing the subsequent event to occur
- Easily configured and analyzed
  - Exposes a standard DB schema
  - Can easily change resolution of events
  - Can easily query graph for specific information (i.e. an instruction trace)



Event Graph

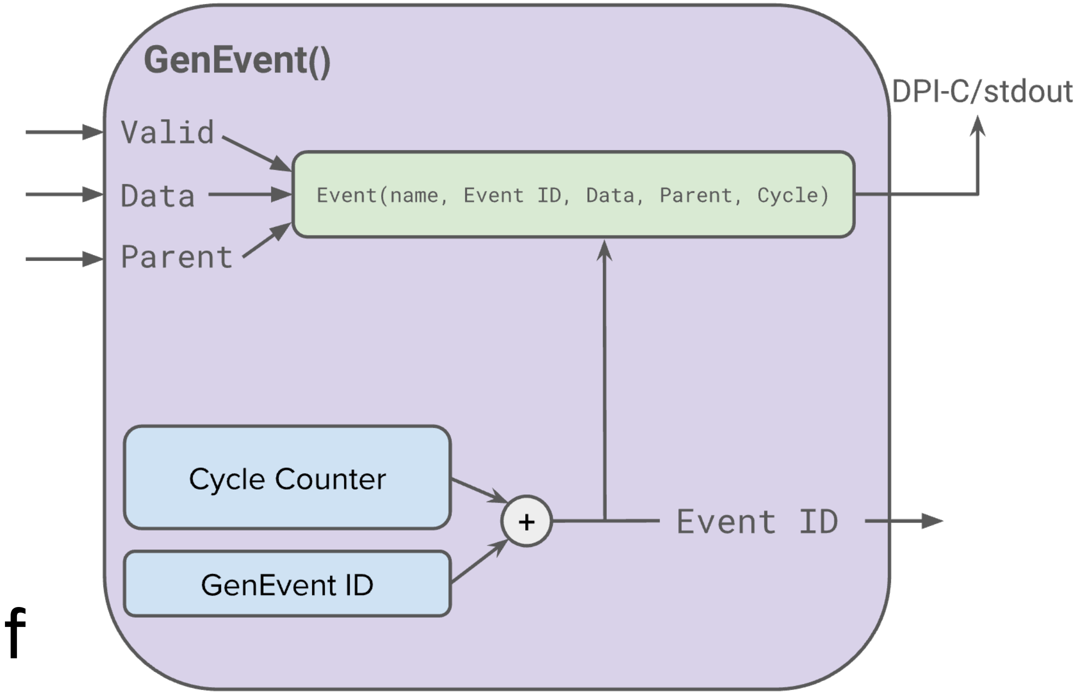
GenEvent Chisel Tag Passing



```
s1_tag := GenEvent(“s1”, s1_valid, s1_data, None)
s2_tag := GenEvent(“s2”, s2_valid, s2_data, s1_tag)
GenEvent(“s3”, s3_valid, s3_data, s2_tag)
```

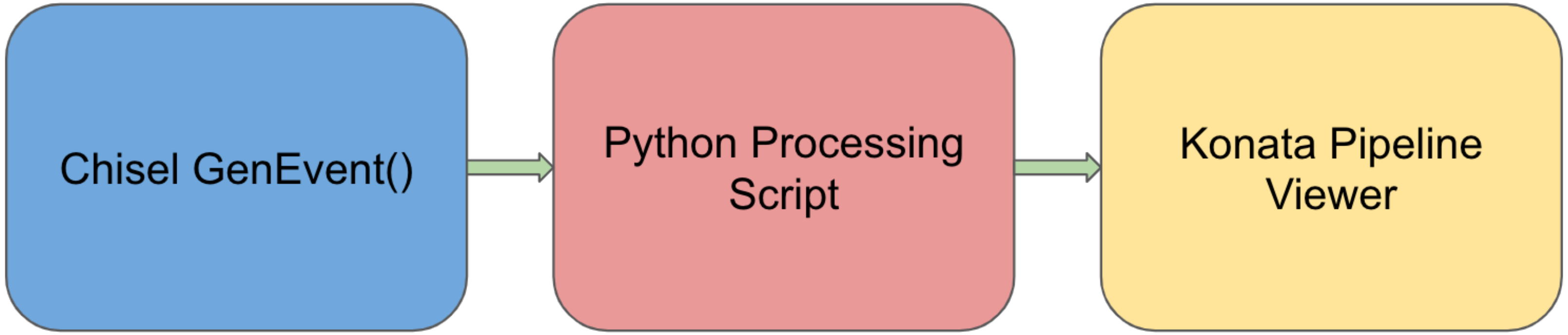
GenEvent Module

- Each GenEvent Module has a:
  - Event Name
  - Valid input
  - Data input
  - Optional Parent ID input
- When valid, GenEvent logs event name, cycle, inputs, and generates a unique Event ID Tag
- Event ID Tags are the primary keys of the event database
- Module outputs Event ID tag in RTL



Application: Konata Pipeline Viewer

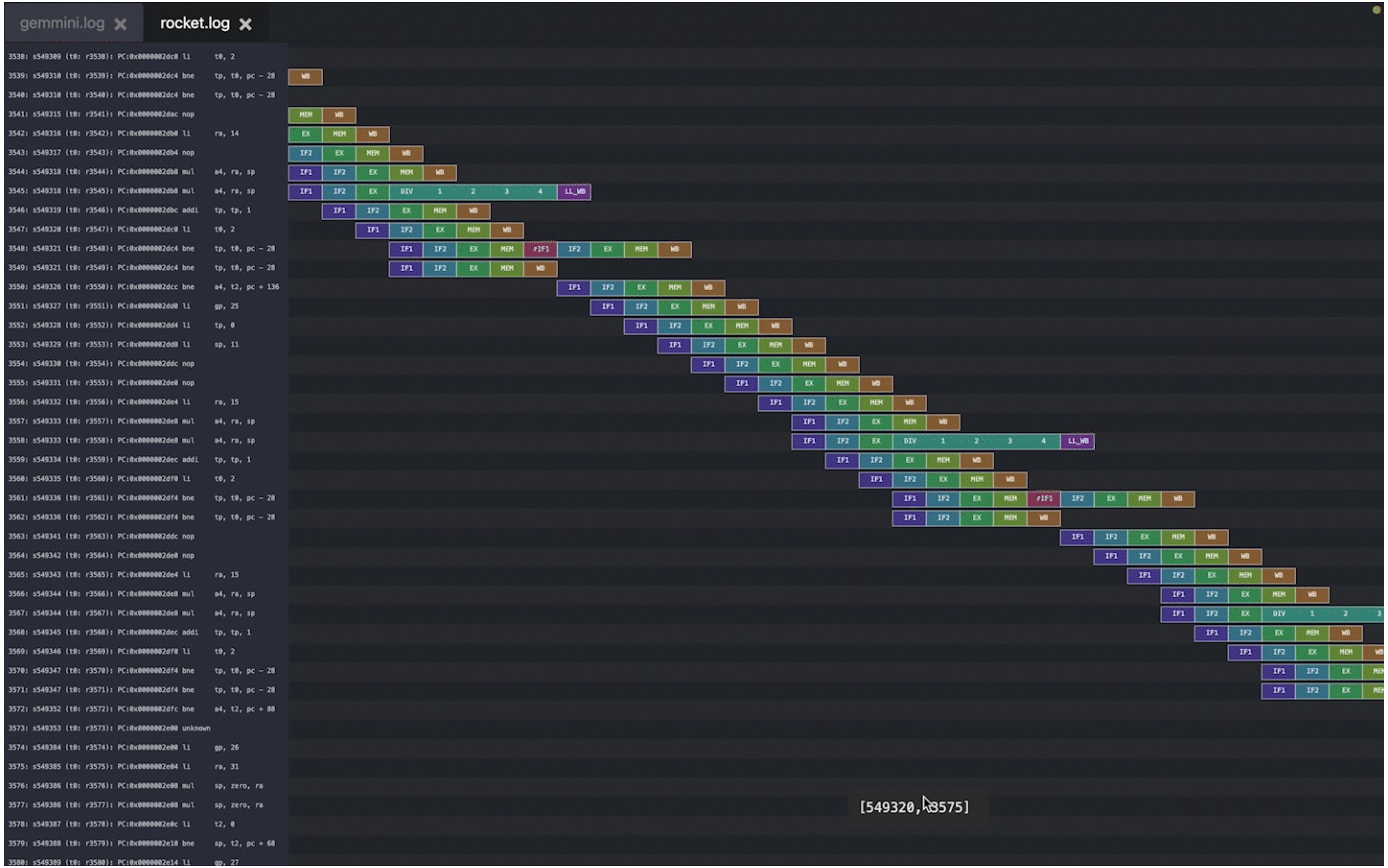
- Run RTL simulation with GenEvent annotated architecture
- Reconstruct the event graph using NetworkX
- Perform depth-first-search to construct the instruction traces
- Format sequences into a Konata log file
- Use Konata application for waterfall visualization of instruction execution



GenEvent API to Konata visualizer flow

https://github.com/shioyadan/Konata

Application: Konata Pipeline Viewer



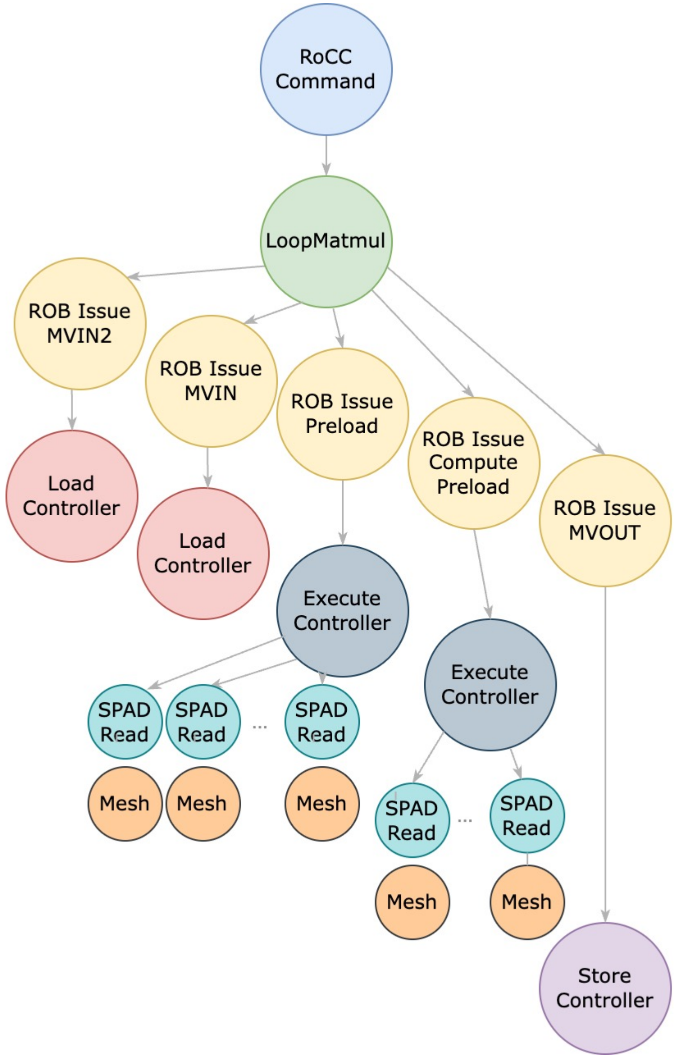
```
3538: s549309 (t0: r3538): PC:0x0000002dc0 li      t0, 2
3539: s549310 (t0: r3539): PC:0x0000002dc4 bne     tp, t0, pc - 28
3540: s549310 (t0: r3540): PC:0x0000002dc4 bne     tp, t0, pc - 28
3541: s549315 (t0: r3541): PC:0x0000002dac nop
3542: s549316 (t0: r3542): PC:0x0000002db0 li      ra, 14
3543: s549317 (t0: r3543): PC:0x0000002db4 nop
3544: s549318 (t0: r3544): PC:0x0000002db8 mul     a4, ra, sp
3545: s549318 (t0: r3545): PC:0x0000002db8 mul     a4, ra, sp
3546: s549319 (t0: r3546): PC:0x0000002dbc addi    tp, tp, 1
3547: s549320 (t0: r3547): PC:0x0000002dc0 li      t0, 2
3548: s549321 (t0: r3548): PC:0x0000002dc4 bne     tp, t0, pc - 28
3549: s549321 (t0: r3549): PC:0x0000002dc4 bne     tp, t0, pc - 28
3550: s549326 (t0: r3550): PC:0x0000002dcc bne     a4, t2, pc + 136
3551: s549327 (t0: r3551): PC:0x0000002dd0 li      gp, 25
3552: s549328 (t0: r3552): PC:0x0000002dd4 li      tp, 0
3553: s549329 (t0: r3553): PC:0x0000002dd8 li      sp, 11
3554: s549330 (t0: r3554): PC:0x0000002ddc nop
3555: s549331 (t0: r3555): PC:0x0000002de0 nop
3556: s549332 (t0: r3556): PC:0x0000002de4 li      ra, 15
3557: s549333 (t0: r3557): PC:0x0000002de8 mul     a4, ra, sp
3558: s549333 (t0: r3558): PC:0x0000002de8 mul     a4, ra, sp
3559: s549334 (t0: r3559): PC:0x0000002dec addi    tp, tp, 1
3560: s549335 (t0: r3560): PC:0x0000002df0 li      t0, 2
3561: s549336 (t0: r3561): PC:0x0000002df4 bne     tp, t0, pc - 28
3562: s549336 (t0: r3562): PC:0x0000002df4 bne     tp, t0, pc - 28
3563: s549341 (t0: r3563): PC:0x0000002ddc nop
3564: s549342 (t0: r3564): PC:0x0000002de0 nop
3565: s549343 (t0: r3565): PC:0x0000002de4 li      ra, 15
3566: s549344 (t0: r3566): PC:0x0000002de8 mul     a4, ra, sp
3567: s549344 (t0: r3567): PC:0x0000002de8 mul     a4, ra, sp
3568: s549345 (t0: r3568): PC:0x0000002dec addi    tp, tp, 1
3569: s549346 (t0: r3569): PC:0x0000002df0 li      t0, 2
3570: s549347 (t0: r3570): PC:0x0000002df4 bne     tp, t0, pc - 28
3571: s549347 (t0: r3571): PC:0x0000002df4 bne     tp, t0, pc - 28
3572: s549352 (t0: r3572): PC:0x0000002dfc bne     a4, t2, pc + 88
3573: s549353 (t0: r3573): PC:0x0000002e00 unknown
3574: s549384 (t0: r3574): PC:0x0000002e00 li      gp, 26
3575: s549385 (t0: r3575): PC:0x0000002e04 li      ra, 31
3576: s549386 (t0: r3576): PC:0x0000002e08 mul     sp, zero, ra
3577: s549386 (t0: r3577): PC:0x0000002e08 mul     sp, zero, ra
3578: s549387 (t0: r3578): PC:0x0000002e0c li      t2, 0
3579: s549388 (t0: r3579): PC:0x0000002e10 bne     sp, t2, pc + 68
3580: s549389 (t0: r3580): PC:0x0000002e14 li      gp, 27
```

Rocket Konata Pipeline Visualization

https://github.com/shioyadan/Konata

Application: Annotated Microarchitectures

- Microarchitecture agnostic:
- Sodor Educational Cores
  - 1 stage, 2 stage, 5 stage, and microcoded cores annotated
- Rocket In-Order Core
  - Integer, mul/div, and cache request/response pipelines annotated
- Gemmini Accelerator
  - Load, Store, and Execution controllers, LoopMatmul and LoopConv FSMs, scratchpad reads/writes, mesh



Gemmini Event Graph
(Y-axis is time)

Conclusion

- Extensible RTL event logging API
- Flexible graph event representation
- Implemented in Chisel with GenEvent
- Graph to visualization flow with Konata

- Questions?

- Demo!

Acknowledgements

○ Thank you, Vighnesh Iyer, Joonho Whangbo, and Ethan Gao for your help!