2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

Formal Verification of Macro-op Cache for Arm Cortex-A77, and its Successor CPU

Vaibhav Agrawal

Principal Engineer, Arm

March 4, 2020

arm

accellera SYSTEMS INITIATIVE

Cortex A-77

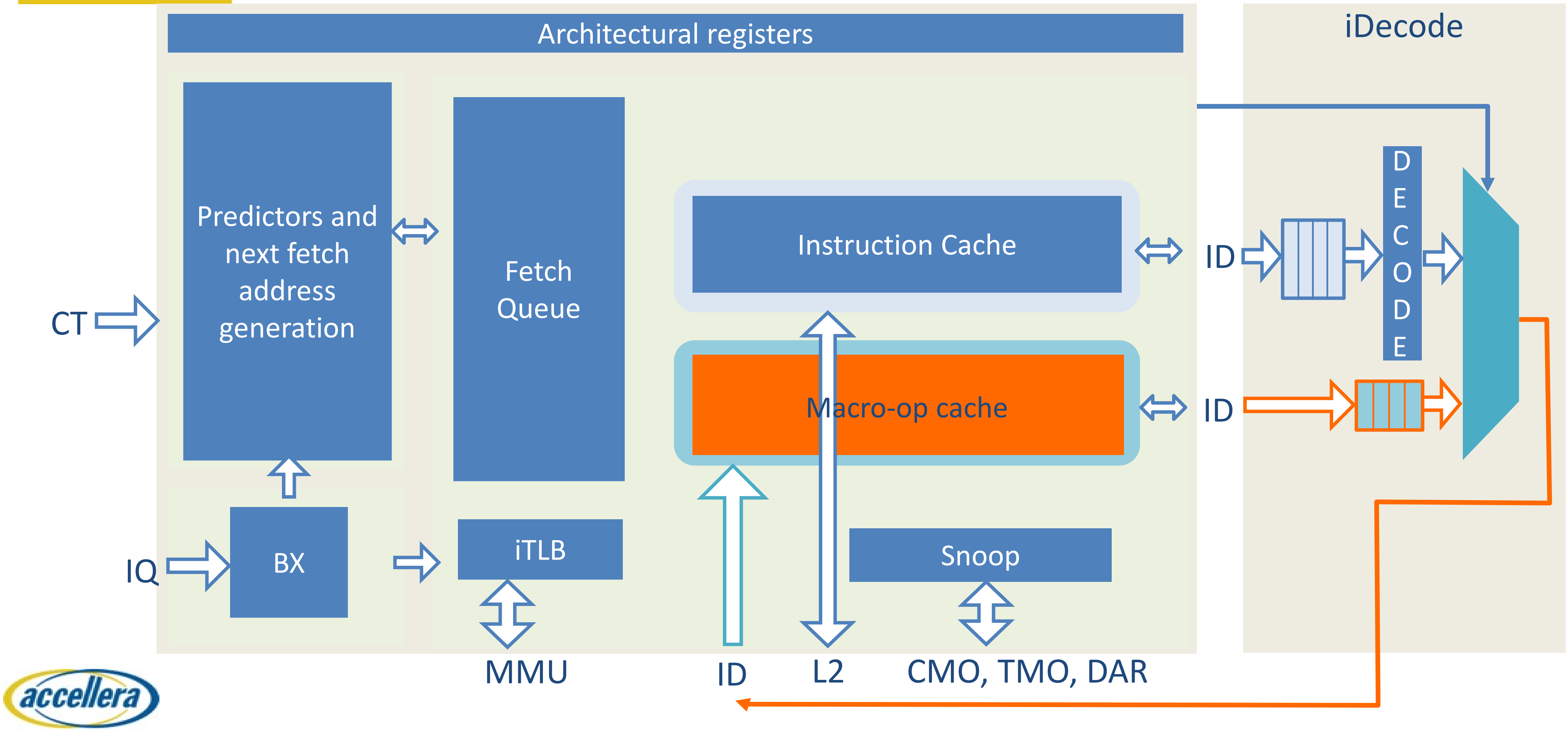
- Announced in Q2, 2019

- From Wikipedia:
  - The **Arm Cortex-A77** is a microarchitecture implementing the ARMv8.2-A 64-bit instruction set designed by ARM Holdings' Austin design centre. The Cortex-A77 is a 4-wide decode out-of-order superscalar design ***with a new 1.5K macro-OP (MOPs) cache***. The instruction fetch is 6-wide (up from 4-wide). The backend is 12 execution ports with a pipeline depth of 13 stages and the execution latencies of 10 stages.

Agenda

- Instruction fetch unit overview
- Reducing design complexity
- End to end checker design, including macro-op cache and instr cache coherence
- Results and summary

Instruction Fetch Unit Overview



Verification Goal for Formal

- Very high flop and gate count => high control complexity
- Therefore, goal was to find bugs (full proofs a bonus)
- Complement simulation for functional verification
- Primary vehicle for ensuring no hangs (forward progress)

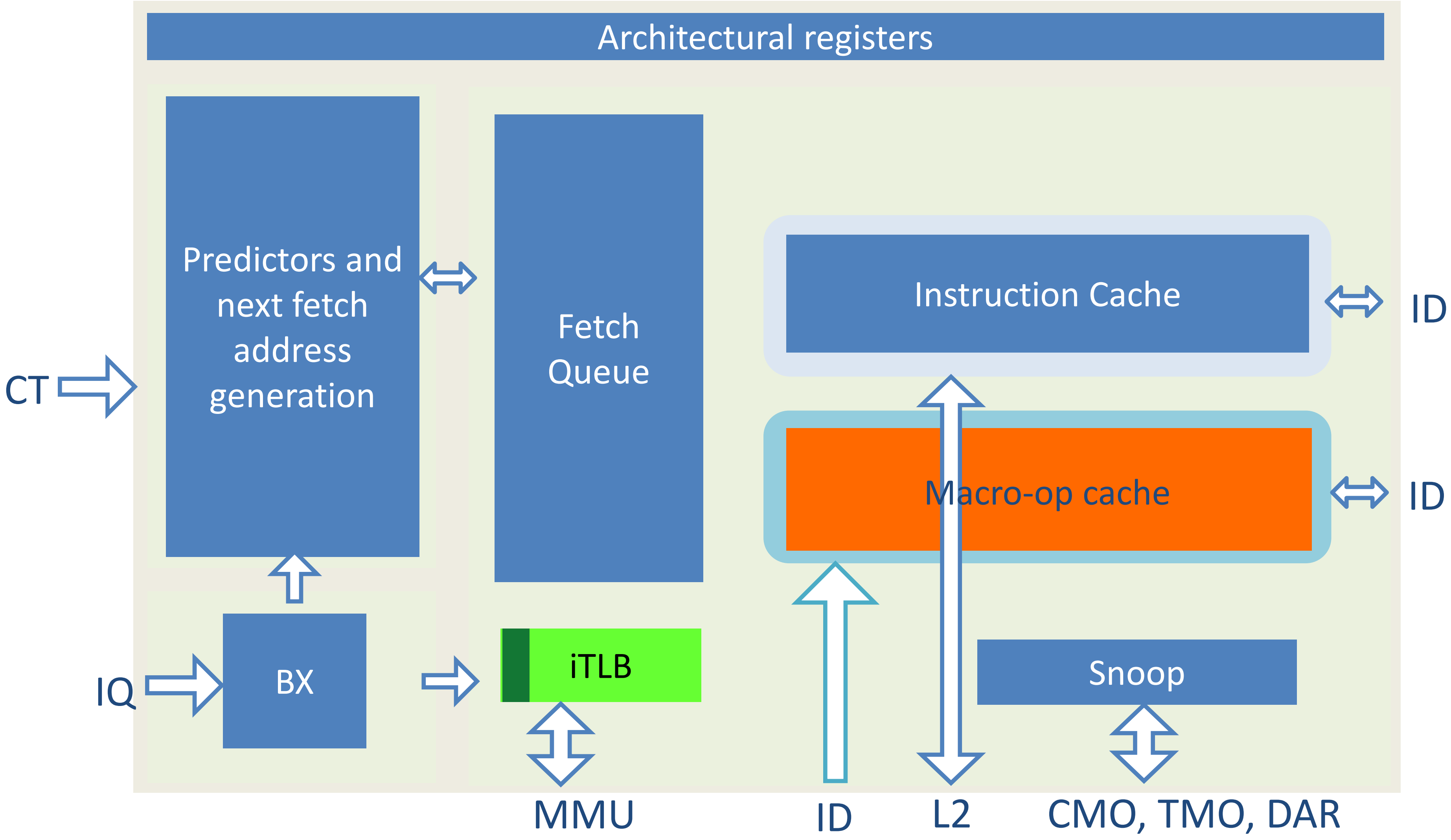
MANAGING COMPLEXITY

Managing Complexity

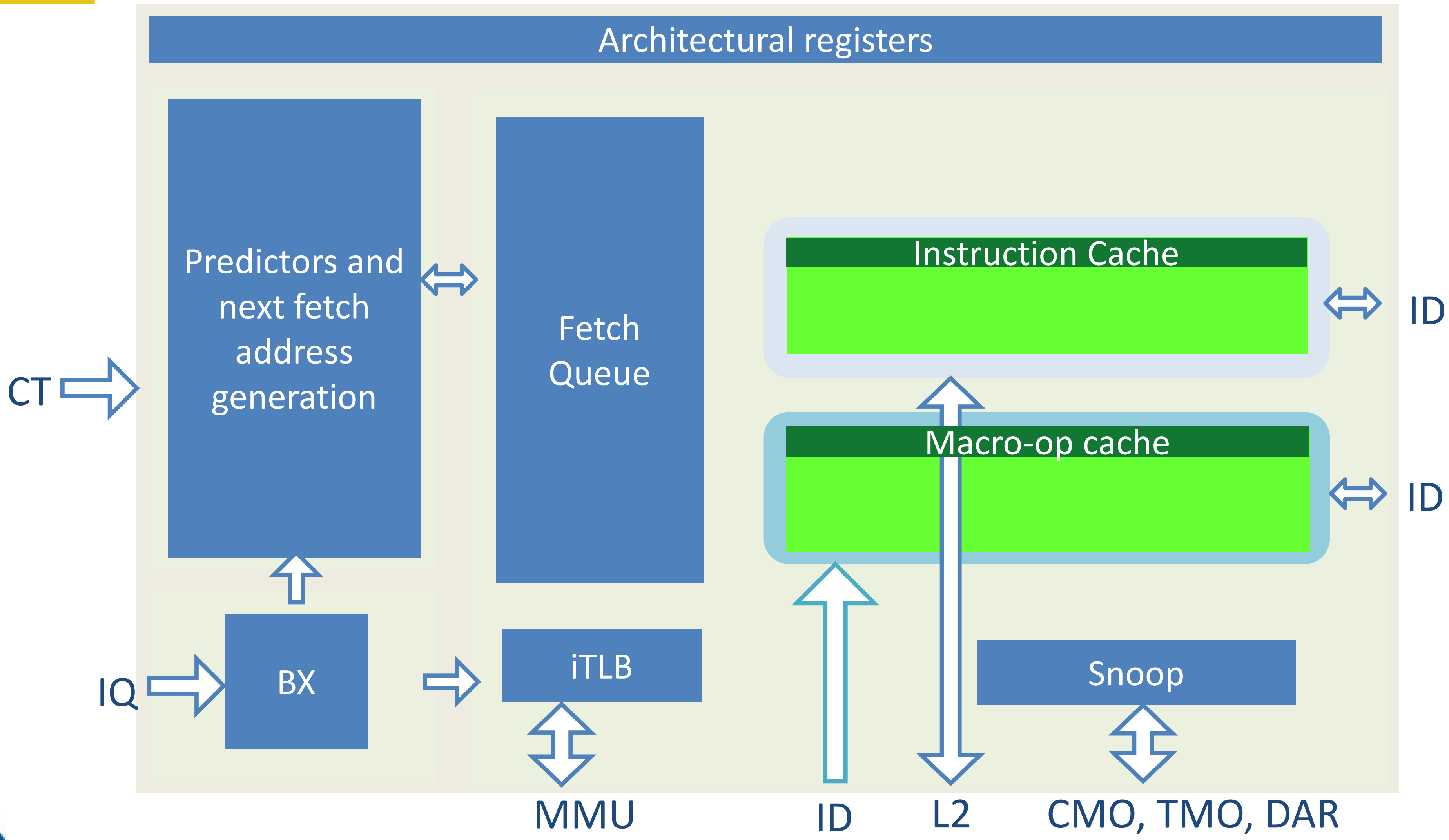
- Complexity reduction
  - Reducing the sequential depth and/or time required to reach interesting design states

S. No.	Complexity management technique
1	Table mutations (designer assisted, or tool assisted)
2	RAMs: Replace number of tracked entries (sets for caches)
3	Initial Value abstractions
4	Reduce data width if implementation doesn’t care
5	Black boxing

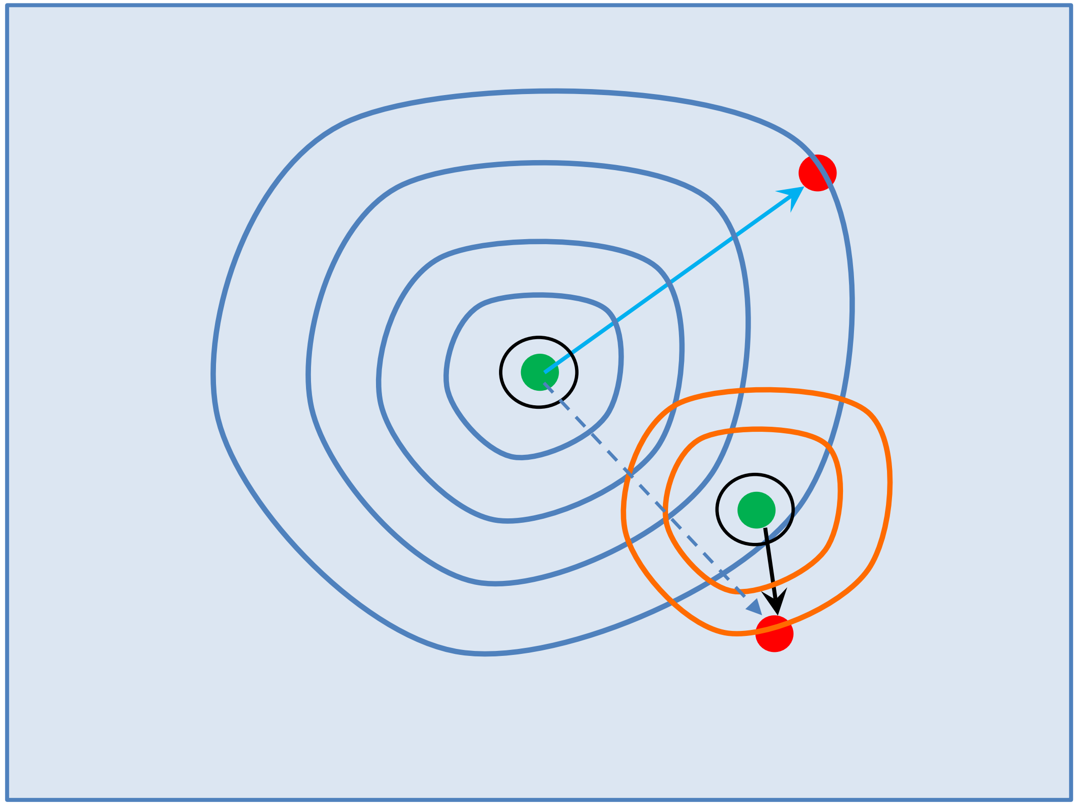
Table Mutations



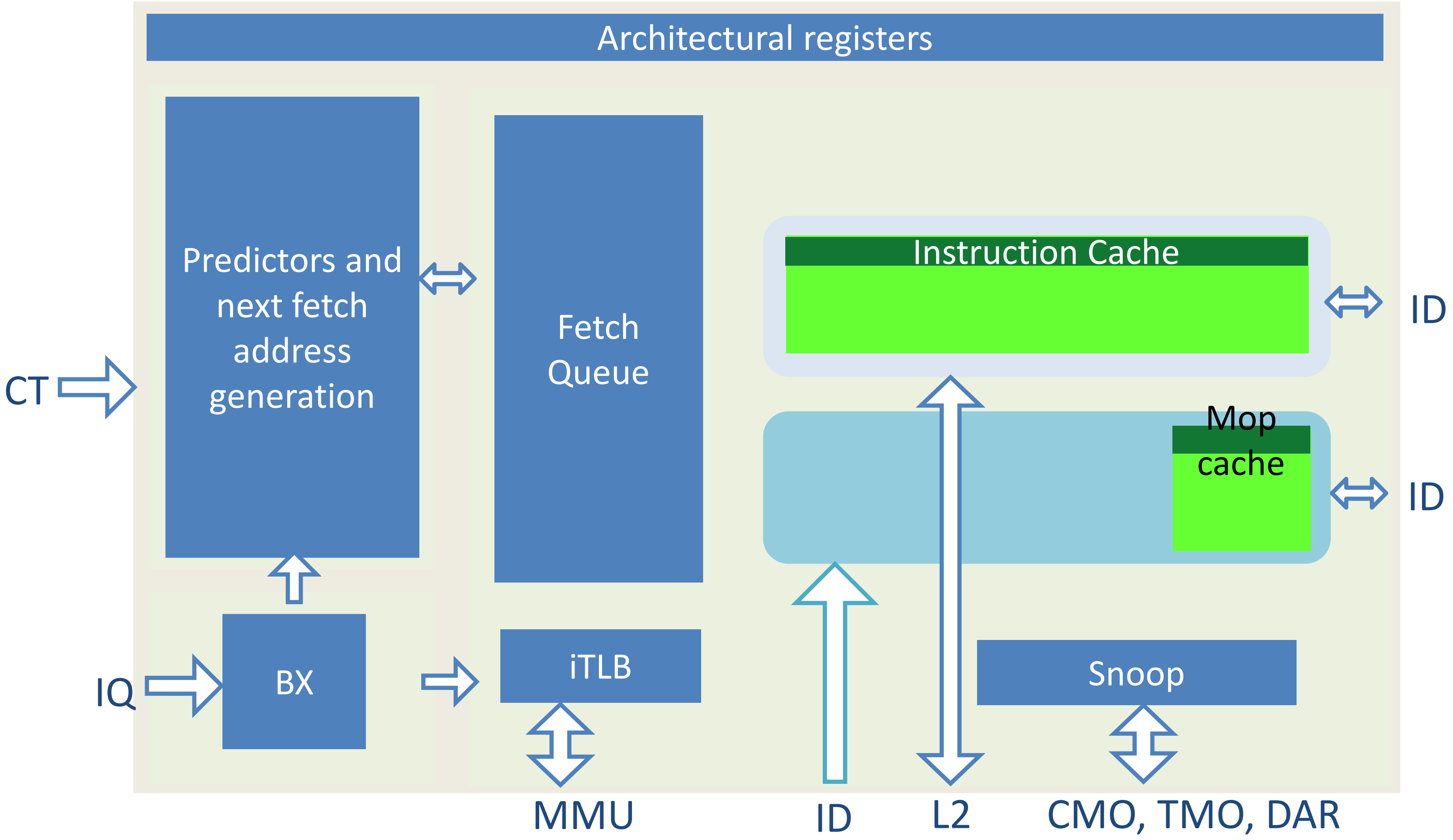
Reduce Number of Tracked Sets in Caches



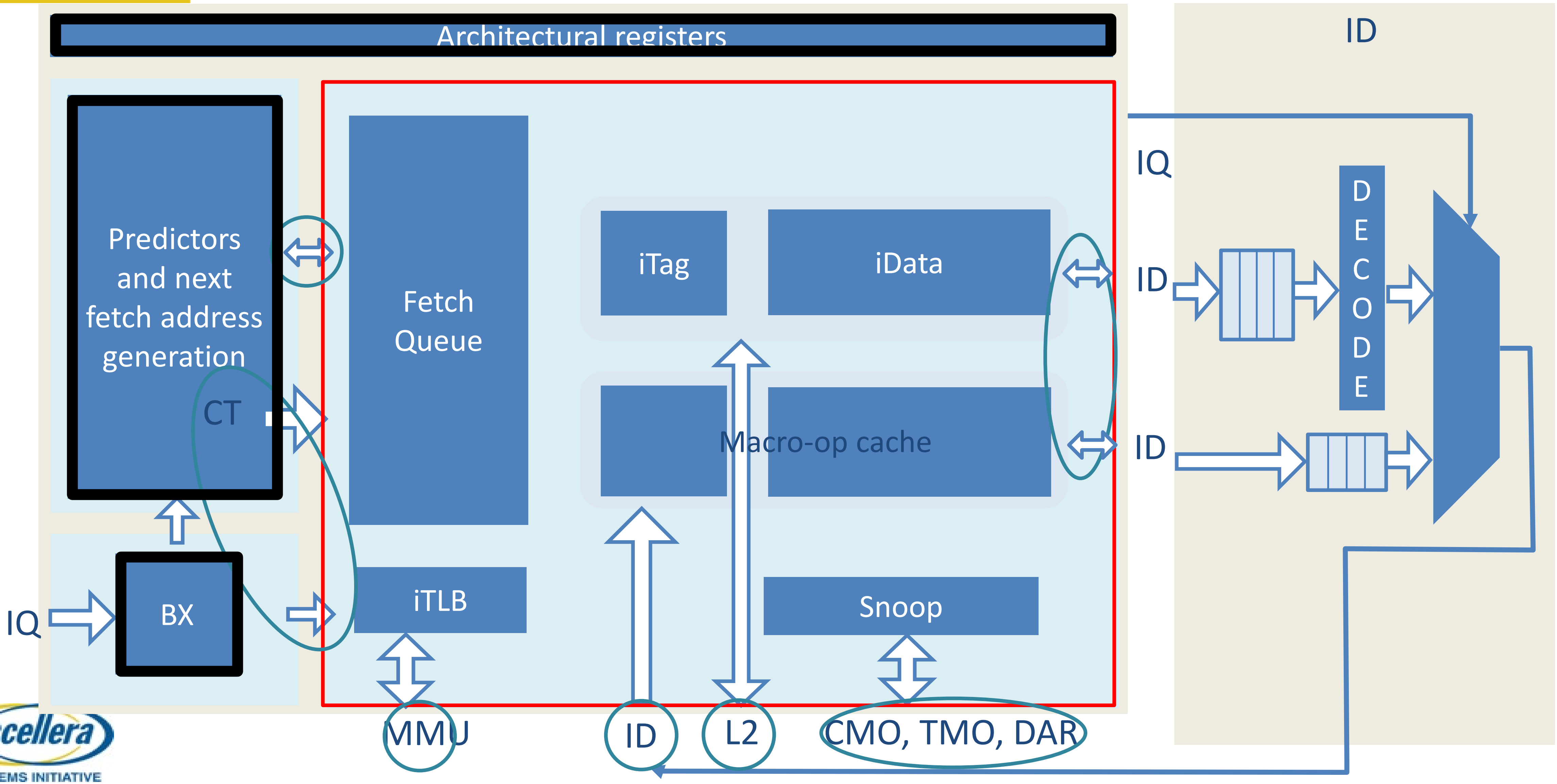
IVAs – Initial Value Abstractions



Reduce Instructions & Mops Size & Number



Black-boxing Unneeded Design Components



END TO END CHECKER DESIGN

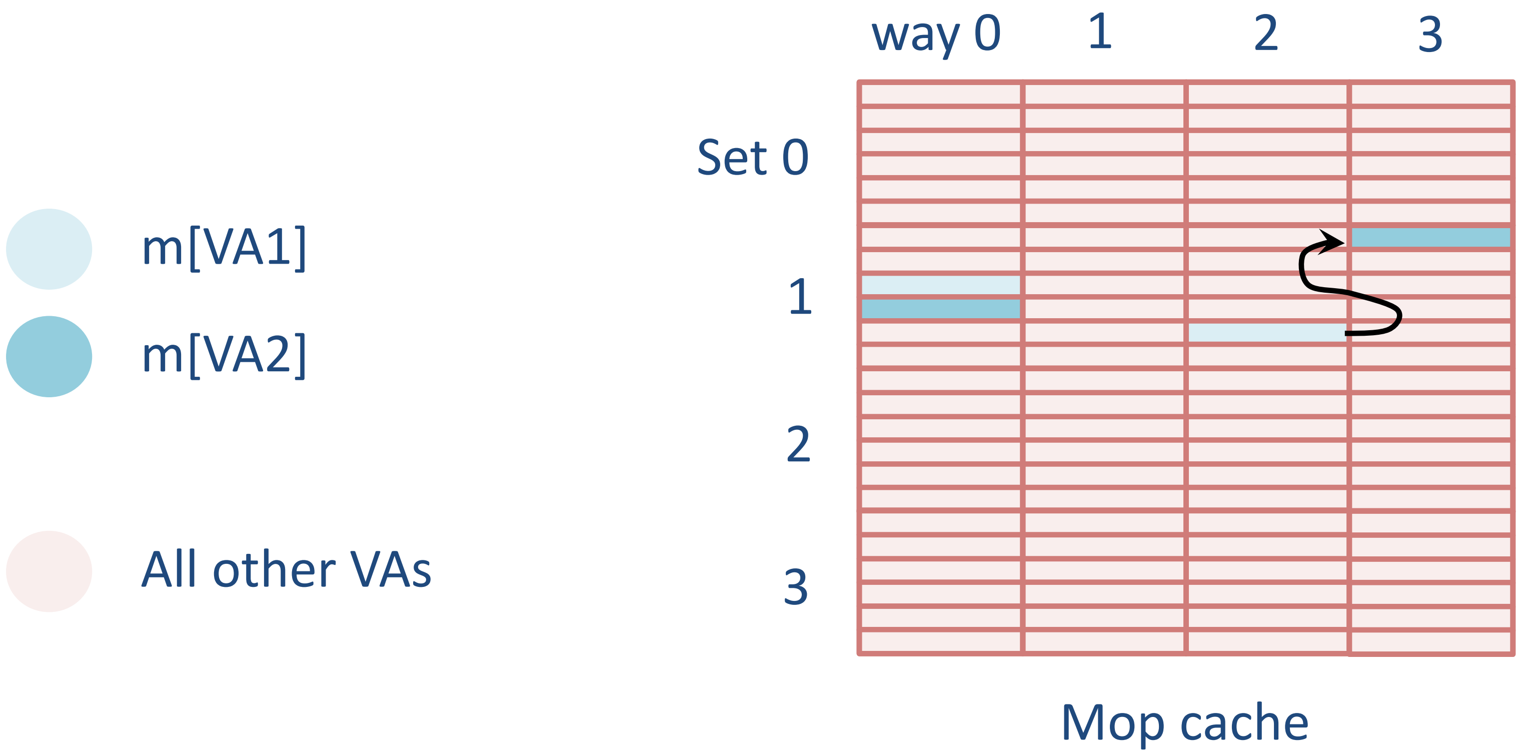
The Basics of Formal Checker Design

- Minimize the number of Flops (both TB and RTL)
- Avoid complex tracking structures
- Think pseudo-constants
  - assume: ##1 $stable (signal)
- Think coloring (tagging) techniques
- Oracles based management of conflicting constraints for assertions

Macro-op Cache Ordering Checker

- 2 Tracked VAs
- Constraint: VA1→VA2
- Constraint: color mops at tr_VA{1,2}
- Check on outputs: m[VA1]→m[VA2]

- Constrain both preloading and fills



Macro-op Fusion with Nop Elimination: Ordering Check

Certain instructions are fused for performance

- 3 Tracked VAs
- Constraint:
  - VA1→VA2→VA3
  - Coloring
  - Fuse m[VA2]↔m[VA3]
  - m[VA2] is eliminated-nop
- Check on outputs: m[VA1]→m[VA3]

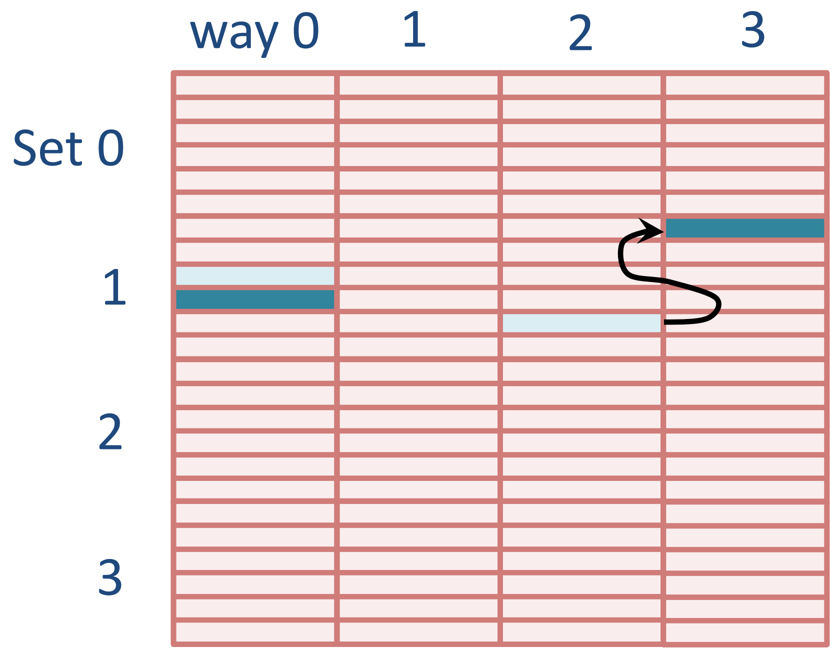
- Constrain both preloading and fills

○ m[VA1] = 2’b01

● m[VA2] = 2’b10

● m[VA3] = 2’b11

○ All other VAs = 2’b00

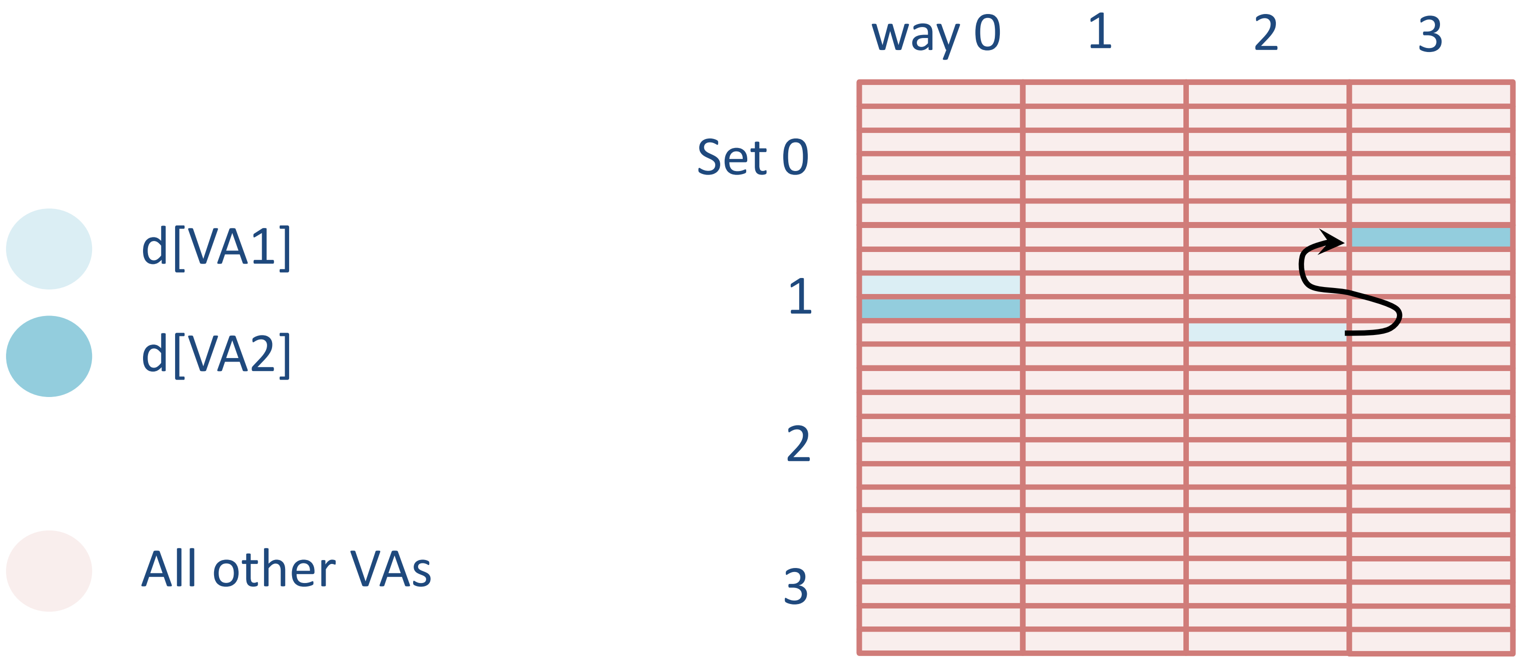


Mop cache

Breakpoint Check

- 2 Tracked VAs
- Constraint: VA1→VA2
- Constraint: color mops at VA{1,2}
- Constraint: bkpt always on VA2; never on VA1
- Liveness Chk: d[VA1] must eventually be seen
- Safety Chk: d[VA2] not seen

- Constrain both preloading and fills

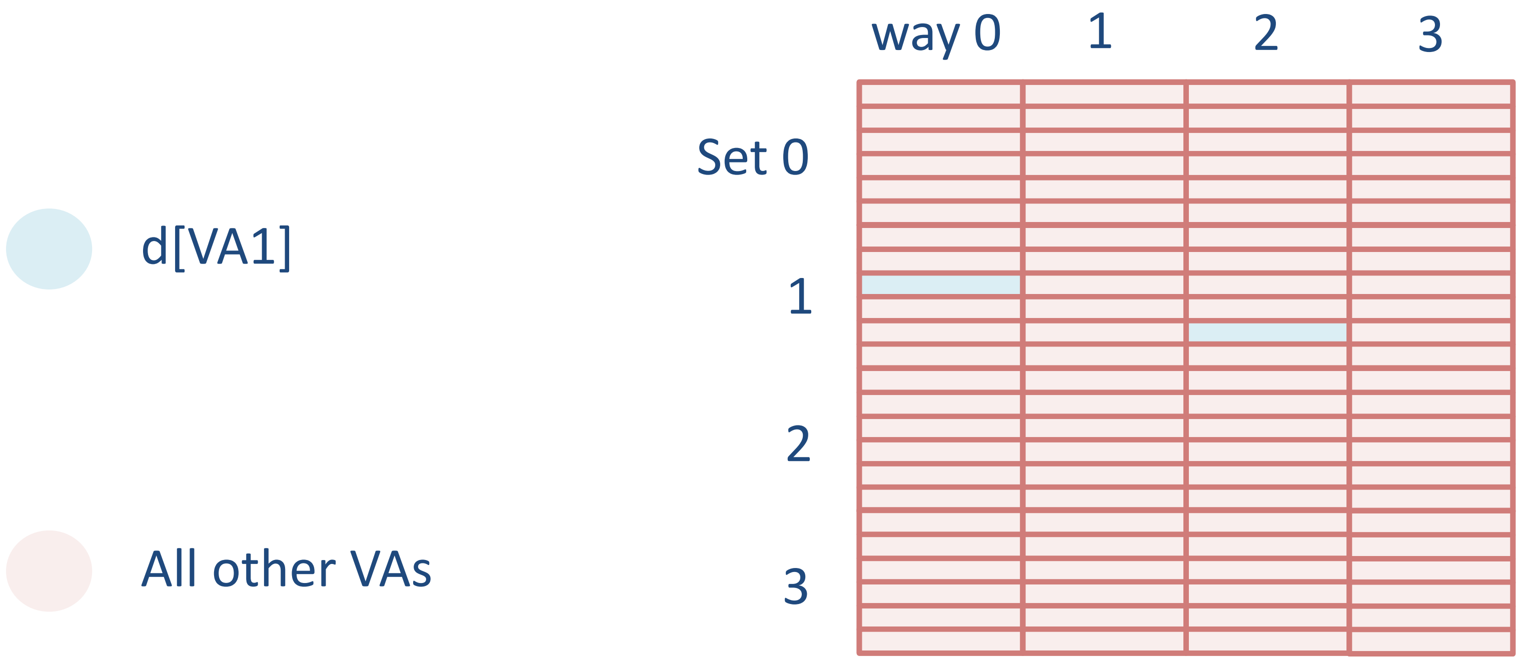


Example cache

Coherence Checking Between Instr Cache and Macro-op cache

- 1 Tracked VA: VA1
- Constraint: color mops & instrs at VA1
- Constraint: d[VA1] in macro-op cache only if also in instr cache
- Check: If d[VA1] absent from instr cache, then it must also be absent from macro-op cache

- Constrain both preloading and fills



Example cache

RESULTS AND SUMMARY

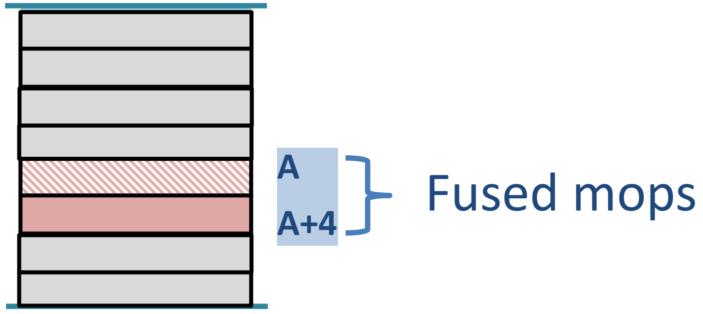
Macro-op / Instr Cache Switching Bug

- For Power: macro-op cache hit => instr cache lookup suppressed
- Instr cache awakens if an address misses in macro-op cache
- Bug in icac wakeup functionality after <N> consecutive back to back macro-op cache hits

MTFF < 10 mins

Mopc Fusion with Nop Elimination

Fused instructions: (A, A+4)



Predicted taken or breakpoint

- If
  - Instruction{A} predicted as taken branch, OR
  - Instruction{A+4} has a breakpoint
- Then
  - Instr{A+4} cannot be delivered from macro-op cache

Forward Progress Bugs by Formal

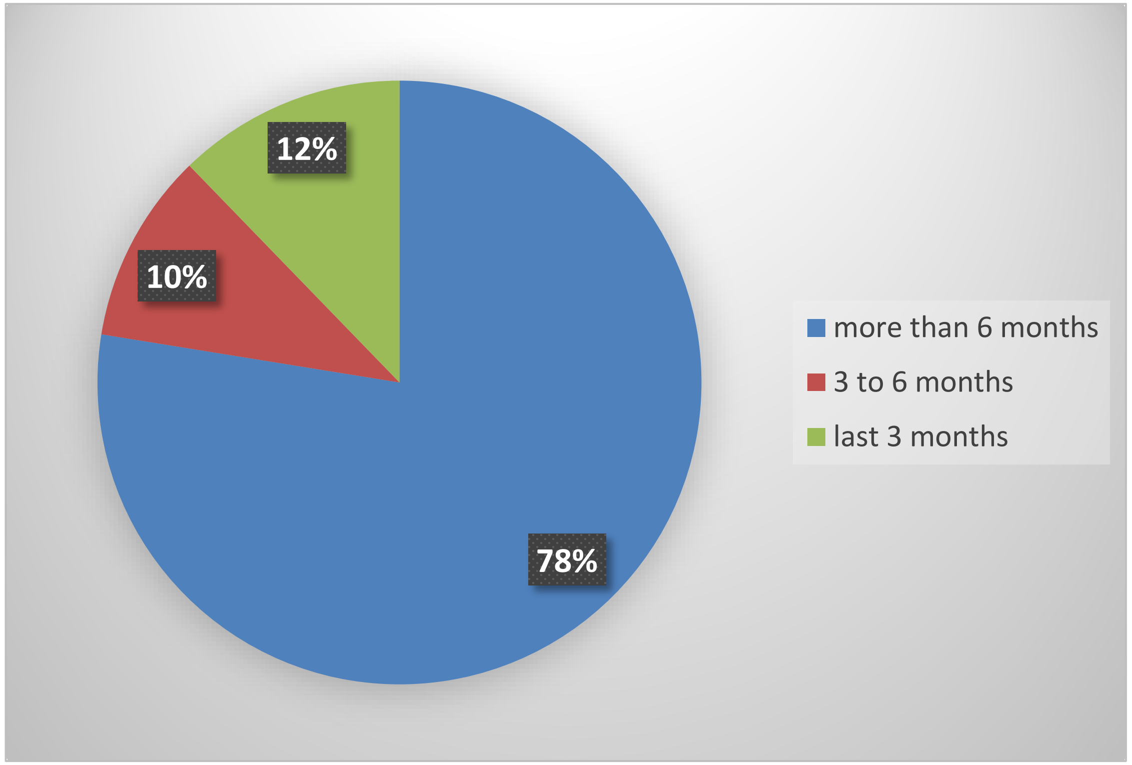
- 10+ bugs in database for forward progress; all were found by formal before other testbenches found them
- 2 were corner cases
- 2 bugs found by bug hunting

Bugs Found by Coherence Checker

- 7 bugs were found by formal checker
- Checker ready before sim environment could be brought up
- 2 of the bugs were “corner case” (included line victimizations, followed by fills, with parity errors, and snoops from L2)

Formal Bug dissection (all bugs)

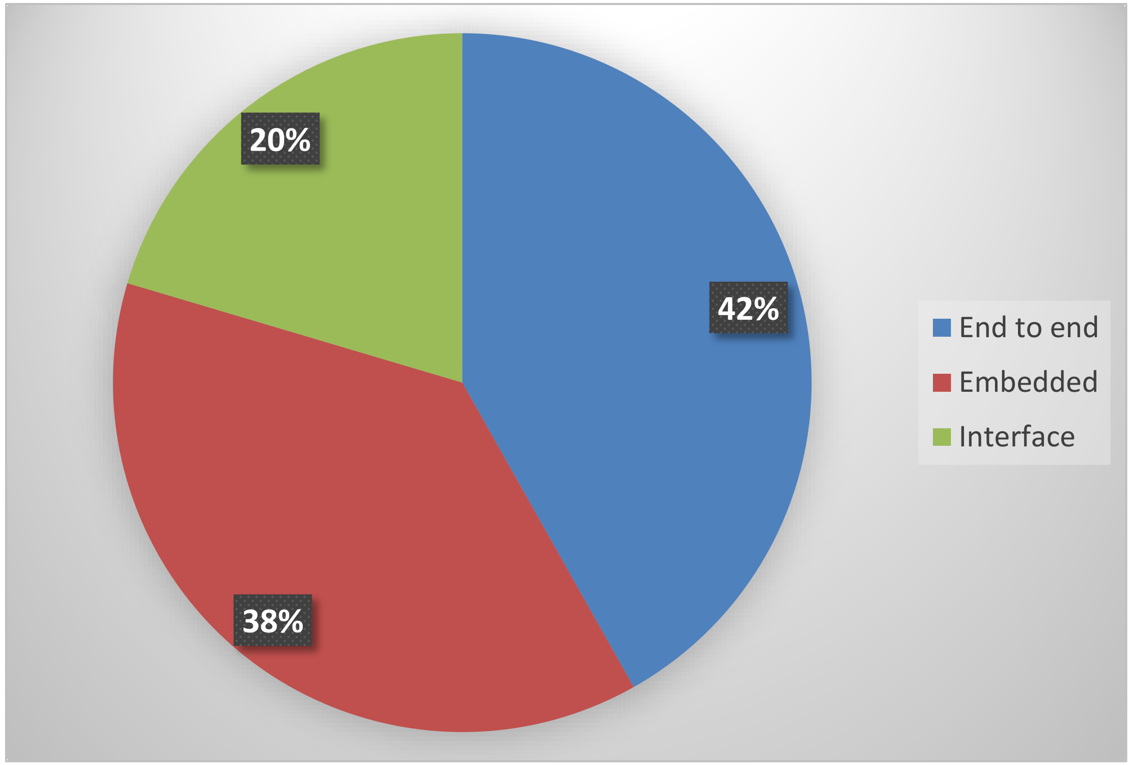
By RTL quality



Both RTL bring up and late in project

- 78% of bugs found during RTL bring up
- 12% found in the last 3 months

Formal Bugs by Property Type (all bugs)



Summary

- Definite impact on ensuring forward progress for high performance A class CPU core
- Exposed a high risk, leading to hardware change: deadlock avoidance mode was added
- Complexity management is crucial
- IVAs necessary, and highly effective
- Bugs found both for bring up, and later in the development cycle before LAC