Robustness of formal verification of x86 microprocessors

Anna Slobodova
anna@centtech.com

Sol Swords, Rob Sumners and Shilpi Goel

High Confidence Software and Systems Conference May 2021

Outline

- Why do we have a need for robustness of proofs at Centaur Technology?
- Challenges to robustness
- Centaur’s response
- Conclusion

Why do we have need for robustness of proofs at Centaur?

- Advancement of the application of formal methods from point proofs to becoming a part of the design process
- Involvement of FV engineers in early stages of the project
- Life cycle of proofs is much longer — months and even years
- FV is part of continuous integration
- Design process relies on FV —> need for robustness

Challenges to robustness

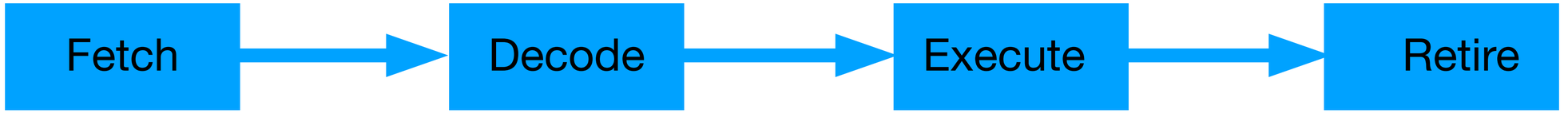
- stability of the tools and libraries
- stability of the specification
- (in)stability of the design
- stability of the proofs

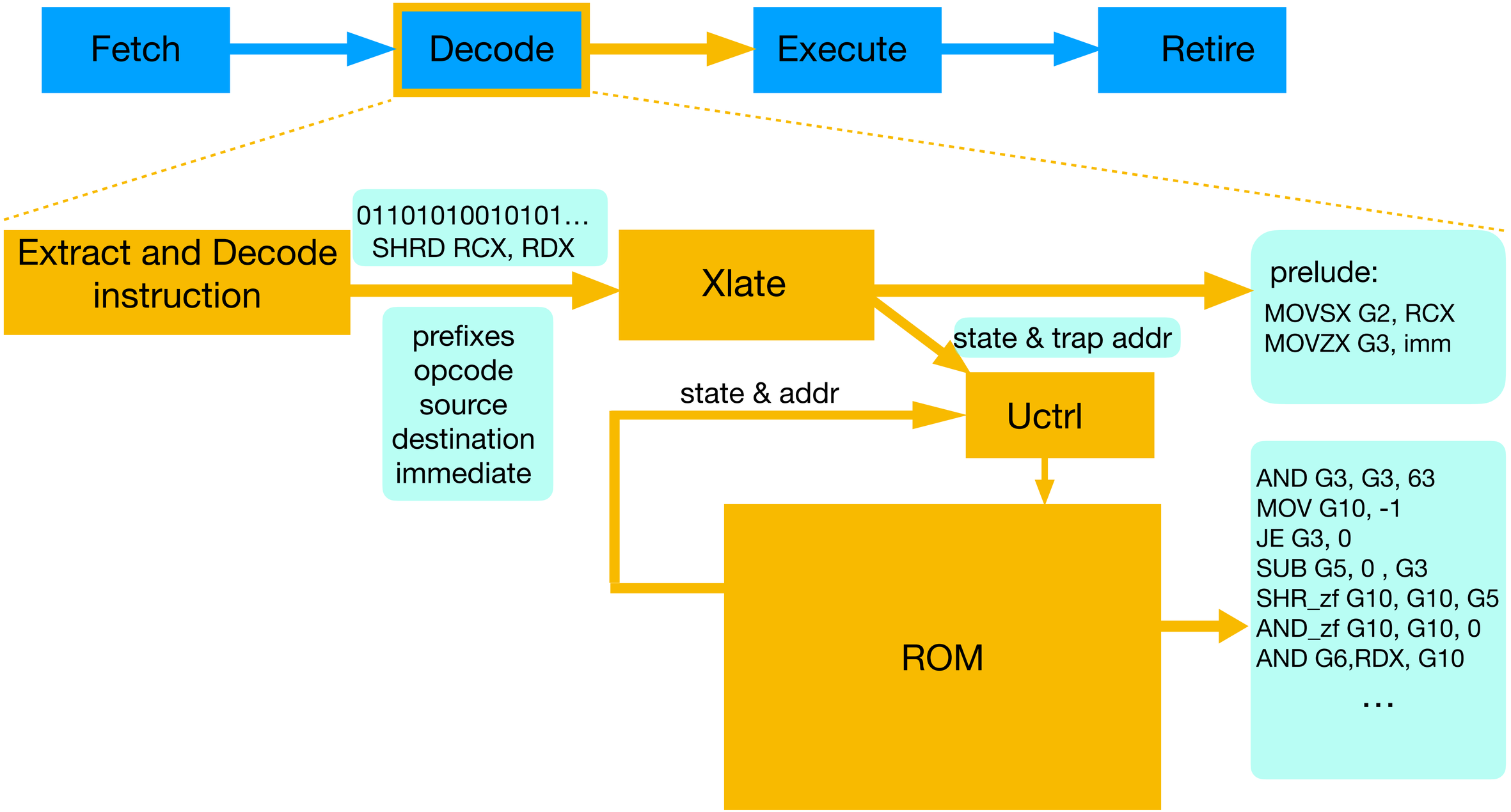
Centaurs's response
stability of the tools and libraries

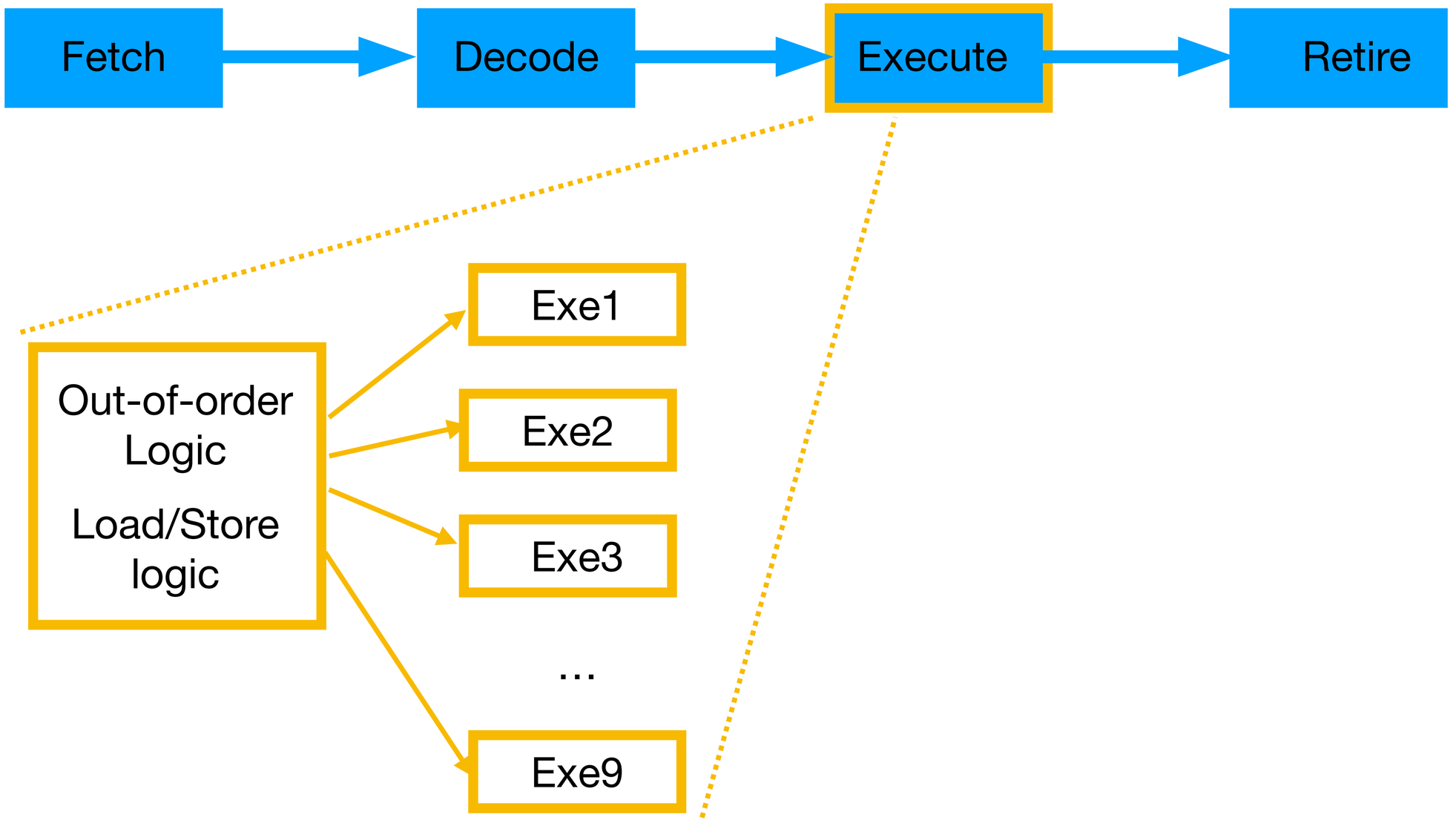
- Centaur FV team uses the ACL2 system for all its work
  - open source, the core is very stable, developers in town
  - numerous libraries that are under development (contributors from Kestrel Inst., Oracle, Centaur, ARM, individuals) — coordinated via Github
  - external tools - SAT solvers, ABC, Z3
  - internal tools

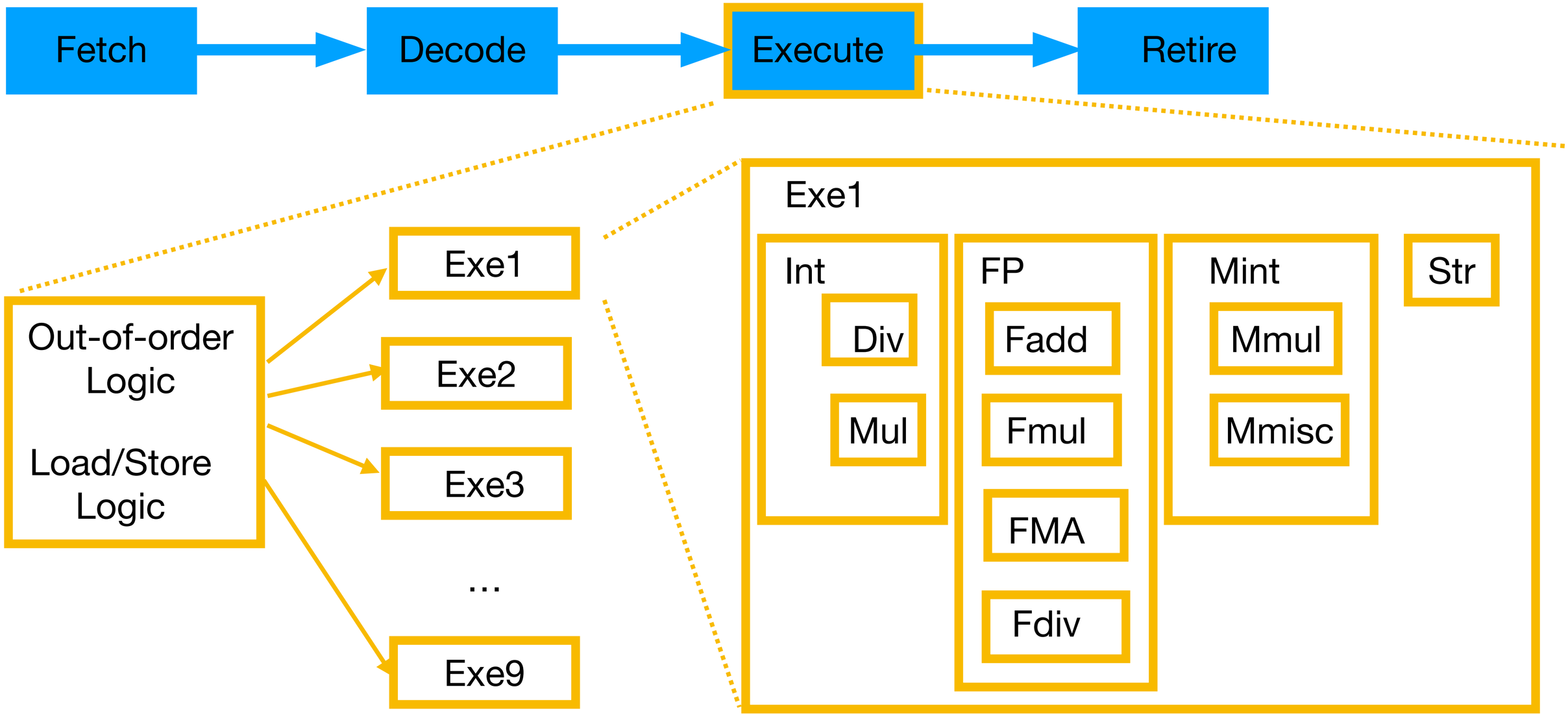
Centaur's response
stability of specification

- x86 ISA specification - *architectural model*
  - stable but growing
- *micro-architectural model*
  - project specific and changing
  - memory hierarchy, set of micro-operations, timing, algorithm implementation
- explained on an example of processing an x86 instruction


Fetch
Decode
Execute
Retire


Fetch
Decode
Execute
Retire
Extract and Decode instruction
01101010010101…
SHRD RCX, RDX
prefixes
opcode
source
destination
immediate
Xlate
prelude:
MOVSX G2, RCX
MOVZX G3, imm
state & trap addr
state & addr
Uctrl
ROM
AND G3, G3, 63
MOV G10, -1
JE G3, 0
SUB G5, 0 , G3
SHR_zf G10, G10, G5
AND_zf G10, G10, 0
AND G6,RDX, G10
…

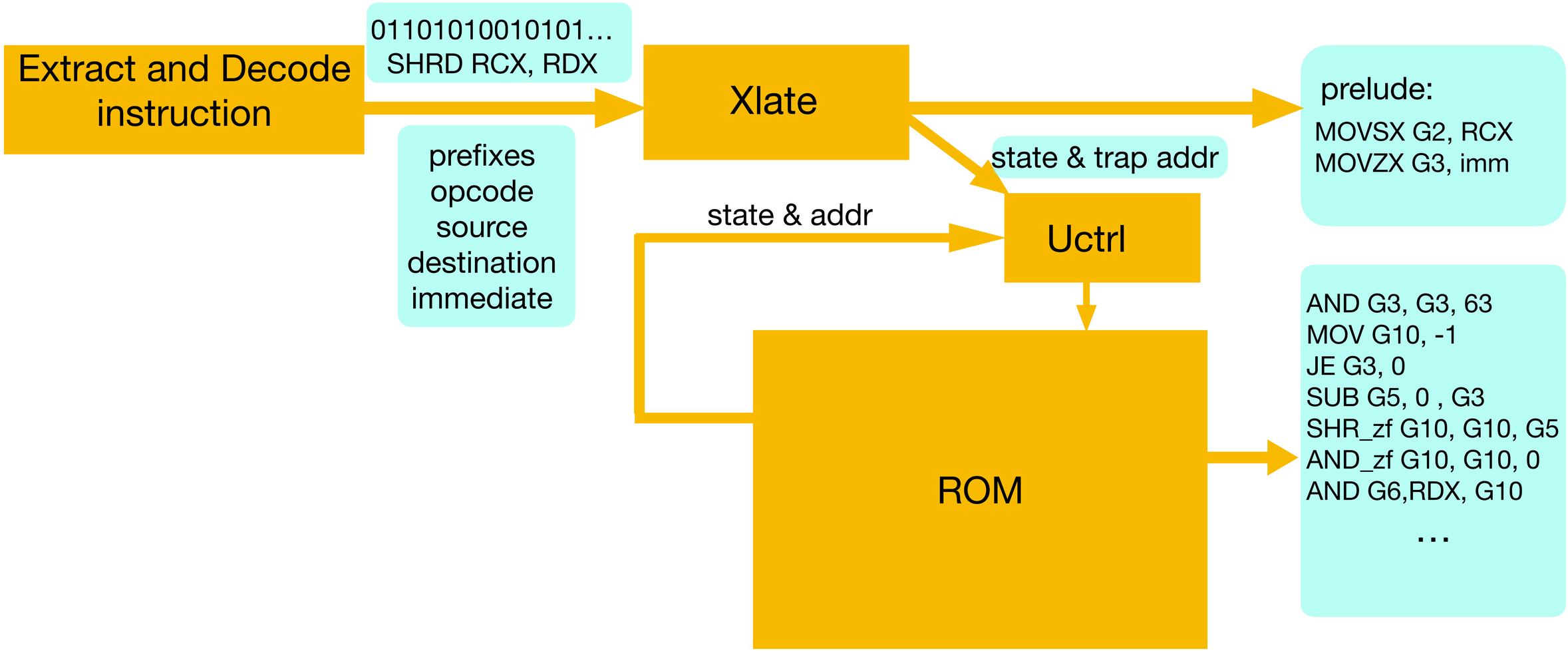

Fetch
Decode
Execute
Retire
Out-of-order Logic
Load/Store logic
Exe1
Exe2
Exe3
...
Exe9


Fetch
Decode
Execute
Retire
Out-of-order Logic
Load/Store Logic
Exe1
Exe2
Exe3
...
Exe9
Exe1
Int
Div
Mul
FP
Fadd
Fmul
FMA
Fdiv
Mint
Mmul
Mmisc
Str

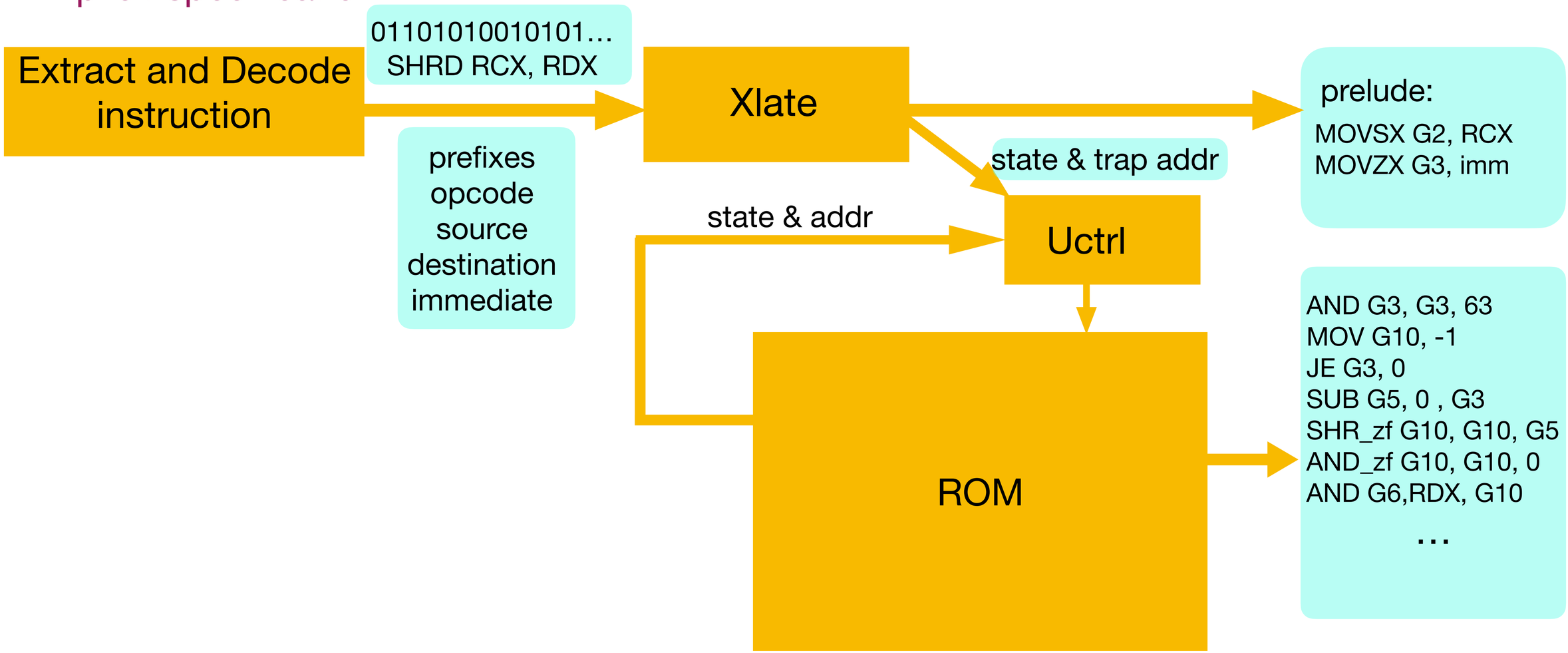
Centaur's response
stability of specification

- Theory: Commutative diagram: *architectural model* ==> *micro-architectural model*
- both models complex, micro-architectural much more so, and is changing rapidly
- example: front-end decode and translate, microcode controller

- How to write specs for Extract/Decode/Xlate/Uctrl?
  - as complex as the design
  - changing
  - unmaintainable



- How to write specs for Extract/Decode/Xlate/Uctrl?
  - as complex as the design
  - changing
  - unmaintainable
- Implicit specification



Centaur's response
stability of specification

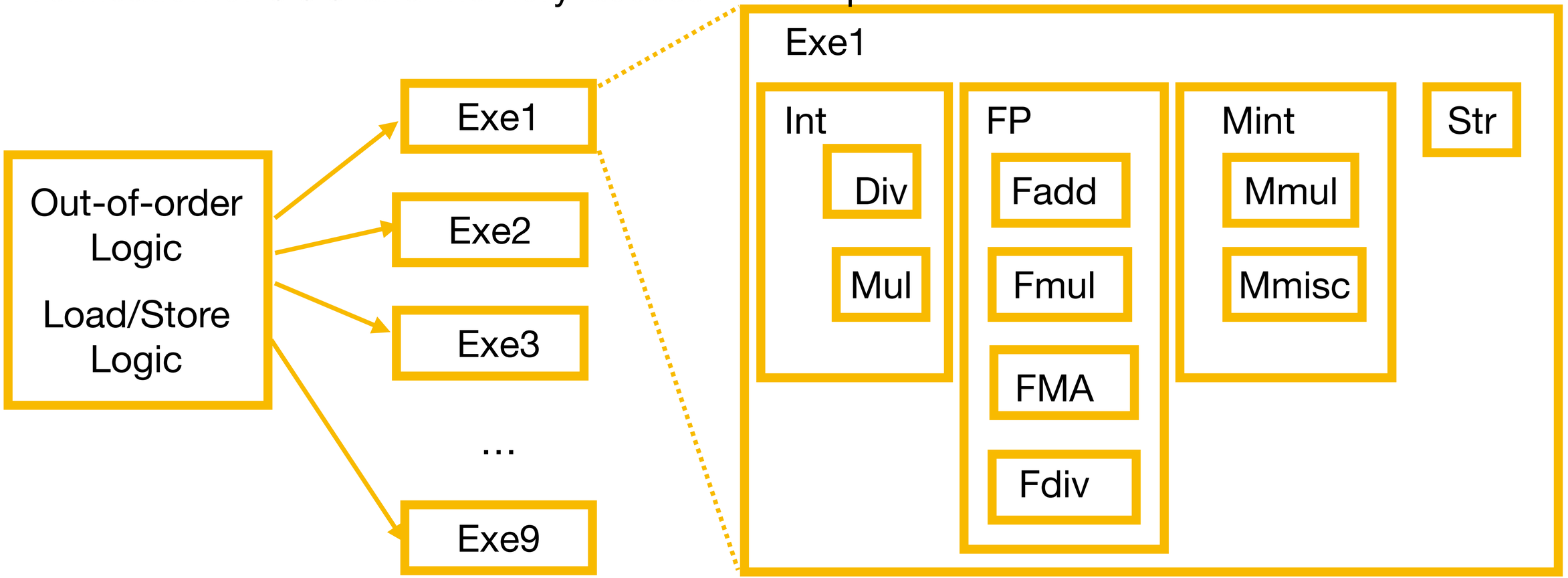
- Theory: Commutative diagram: *architectural model* ==> *micro-architectural model*
- both models complex, micro-architectural much more so, and changing rapidly
- example: front-end decode and translate
- solution: micro-architectural model is a combination of parts defined implicitly by symbolic execution of parts of the design, and partly explicitly defined by describing operational semantics of individual micro-operations

Centaur's response
stability of specification

- Theory: Commutative diagram: *architectural model* ==> *micro-architectural model*
- both models complex, micro-architectural much more so, and changing rapidly
- example: front-end decode and translate
- solution: micro-architectural model is a combination of parts defined implicitly by symbolic execution of parts of the design, and partly explicitly defined by describing operational semantics of individual micro-operations
- explicitly defined parts of the micro-architectural model require validation - verifying that each micro-operation is consistent with our specification

Micro-operations (excluding Ld/St) are executed in respective Exe modules

- their specification is proprietary, changing with projects
- most operations have fixed latency, known FV methods
- verification of Exe important part of validation of micro-architectural model
- proof regressions catch any changes in the specification, or bugs in design
- verification of OOO and memory-access micro-operations - future work



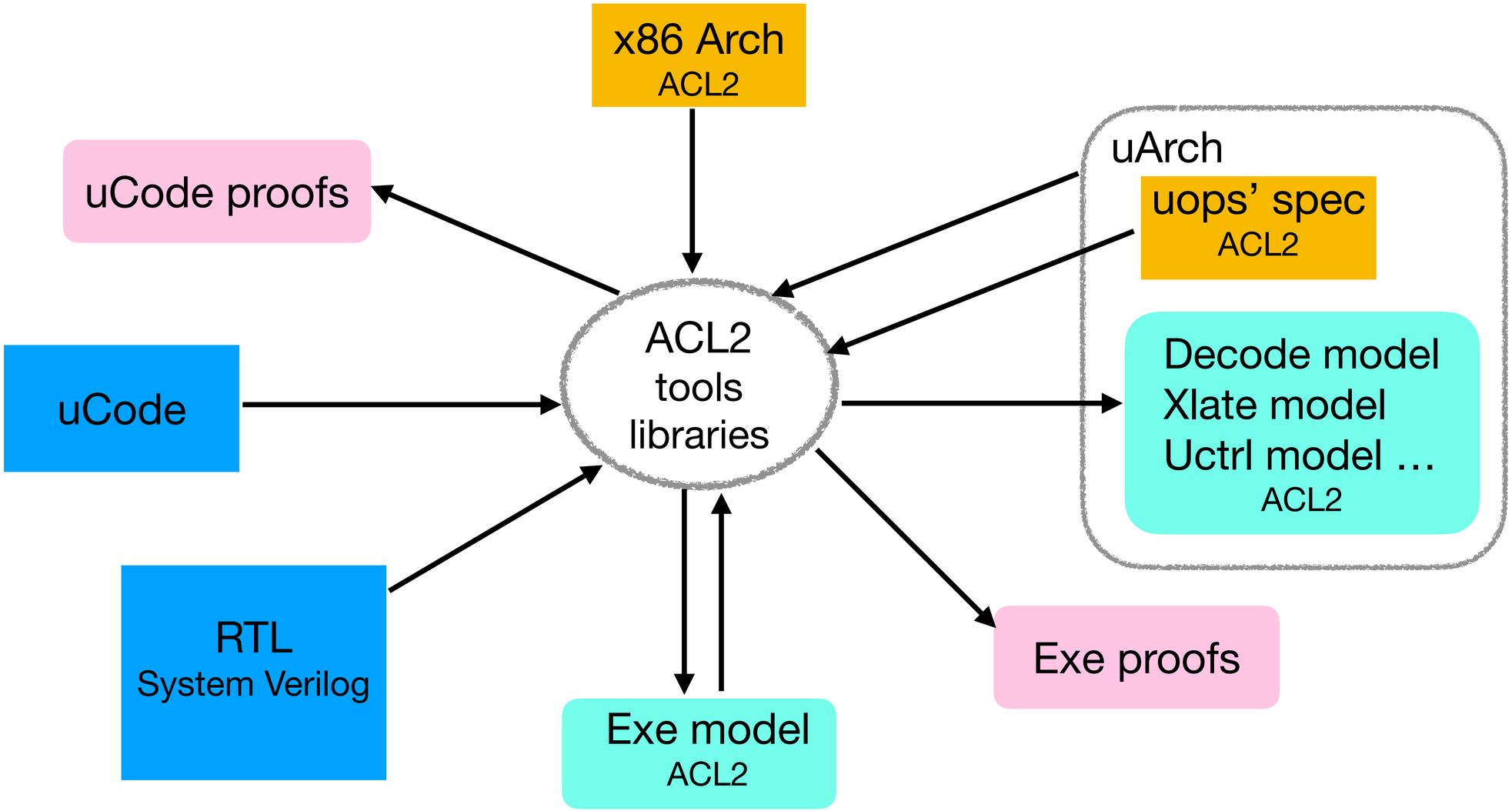
Centaur's response
(in)stability of design

- Instability of the design is inherent to our job
  - FV starts in early stages of the design
    - not just proofs at the end but includes bug finding throughout the design process
  - specification has to accommodate incomplete design
- Instability of the design can be mitigated by increasing the scope of the proofs - we migrated from smaller units (Fadd, Fdiv, Mul) to large modules (Exe)
  - less frequent changes of interface
  - less frequent changes of timing
  - less assumptions about interface
  - the goal: top theorem expresses correctness with respect to top-level module

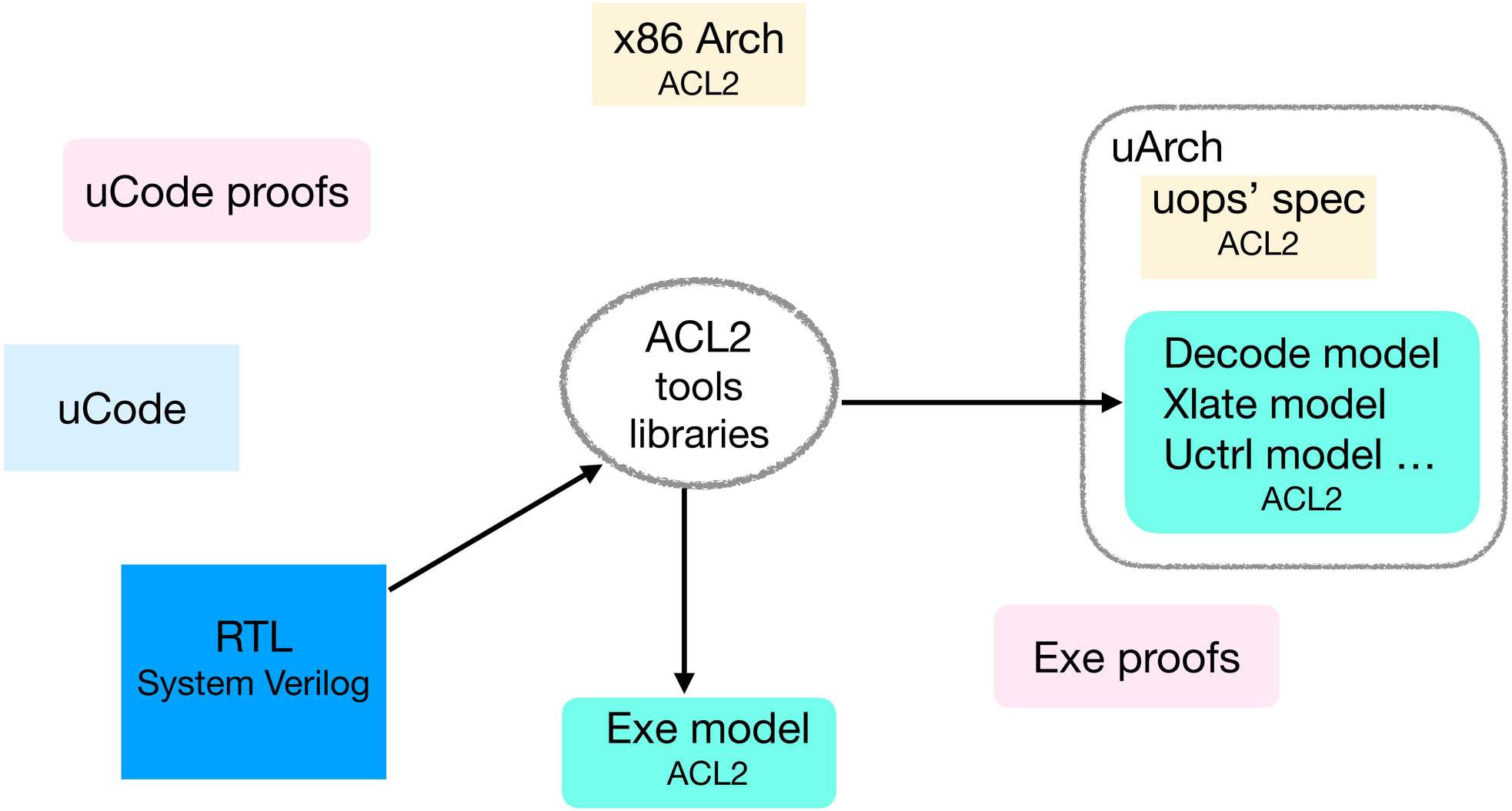
Centaur's response
stability of proofs

- What helped us to increase the scope of our proofs?
  - Improvement in our model build
    - it takes just minutes to build our model of top-level execution unit with all sub-units executing arithmetic, boolean, and string, scalar and SIMD operations from System Verilog design
  - FGL - symbolic simulator with rewriting capabilities
    - See our upcoming paper at CAV 2021: *Balancing automation and control for formal verification of microprocessors*.
    - FGL is formally verified and integrated into ACL2
    - publicly available
  - Improvements in AIG manipulation algorithms that reduce their size
  - Improvements in SAT solvers increase capacity of our tools
    - can be added to ACL2 as trusted tools, but their results can be verified

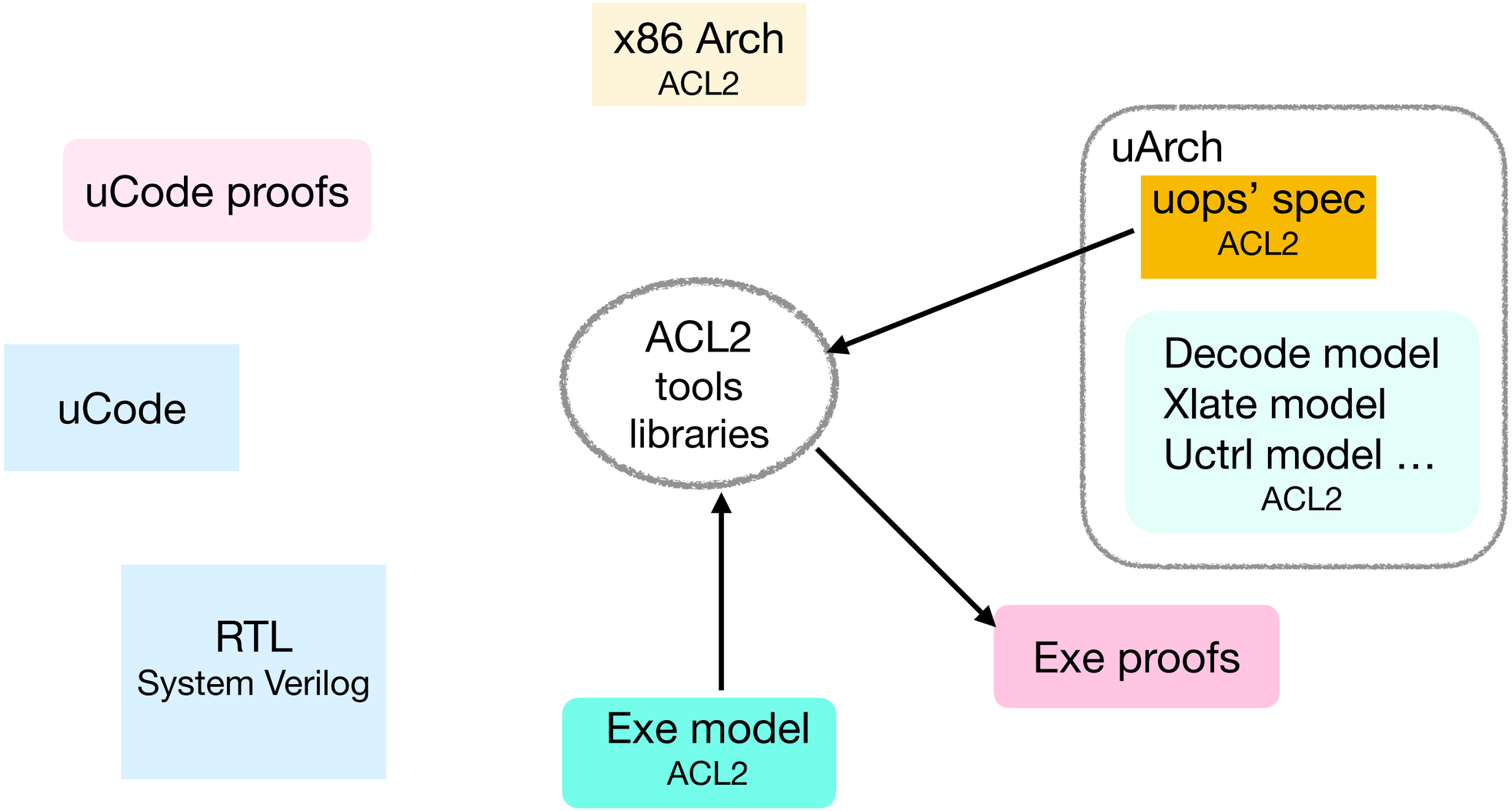
Interdependence of all proof artifacts



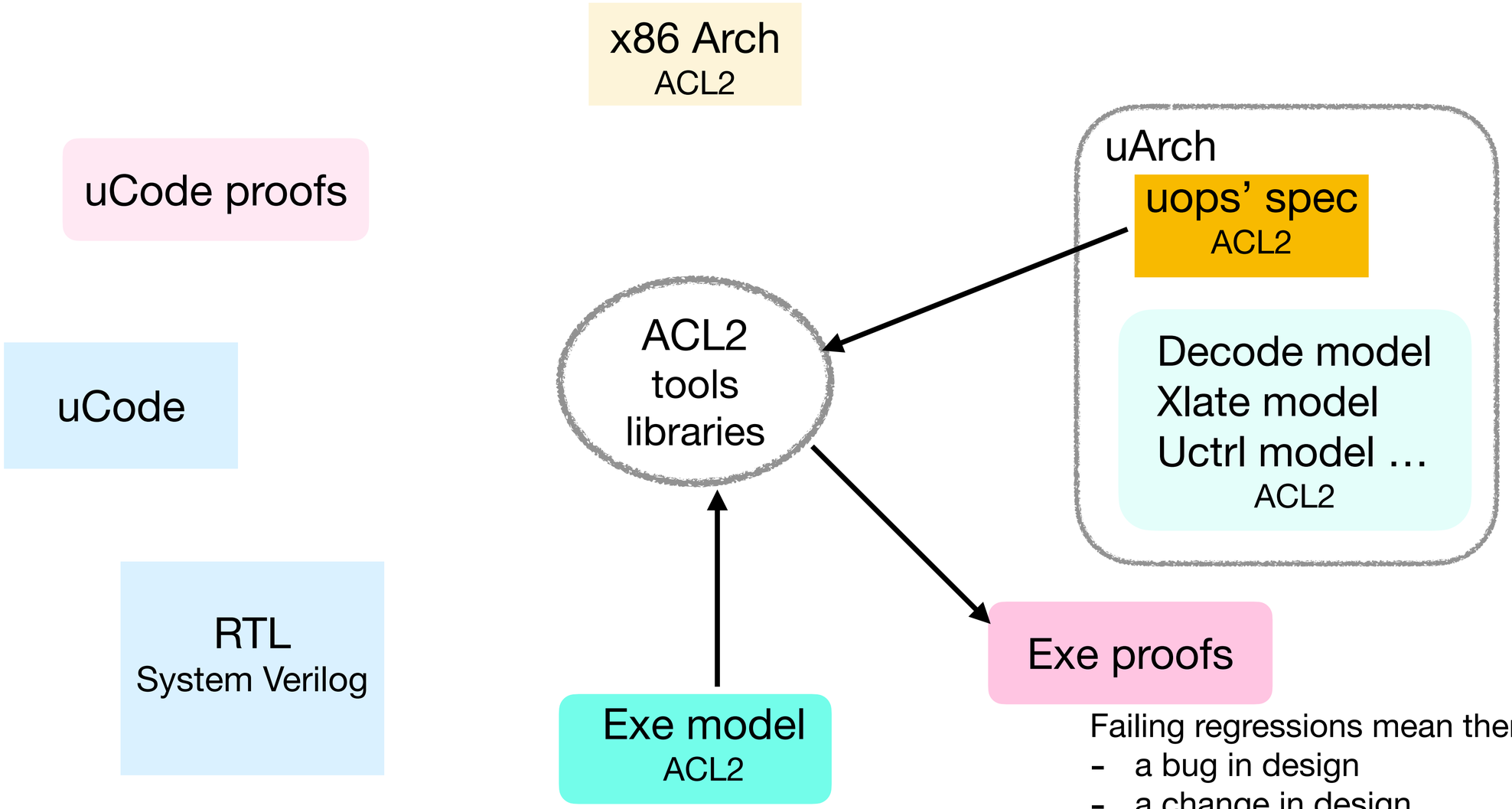
Interdependence of all proof artifacts



Interdependence of all proof artifacts



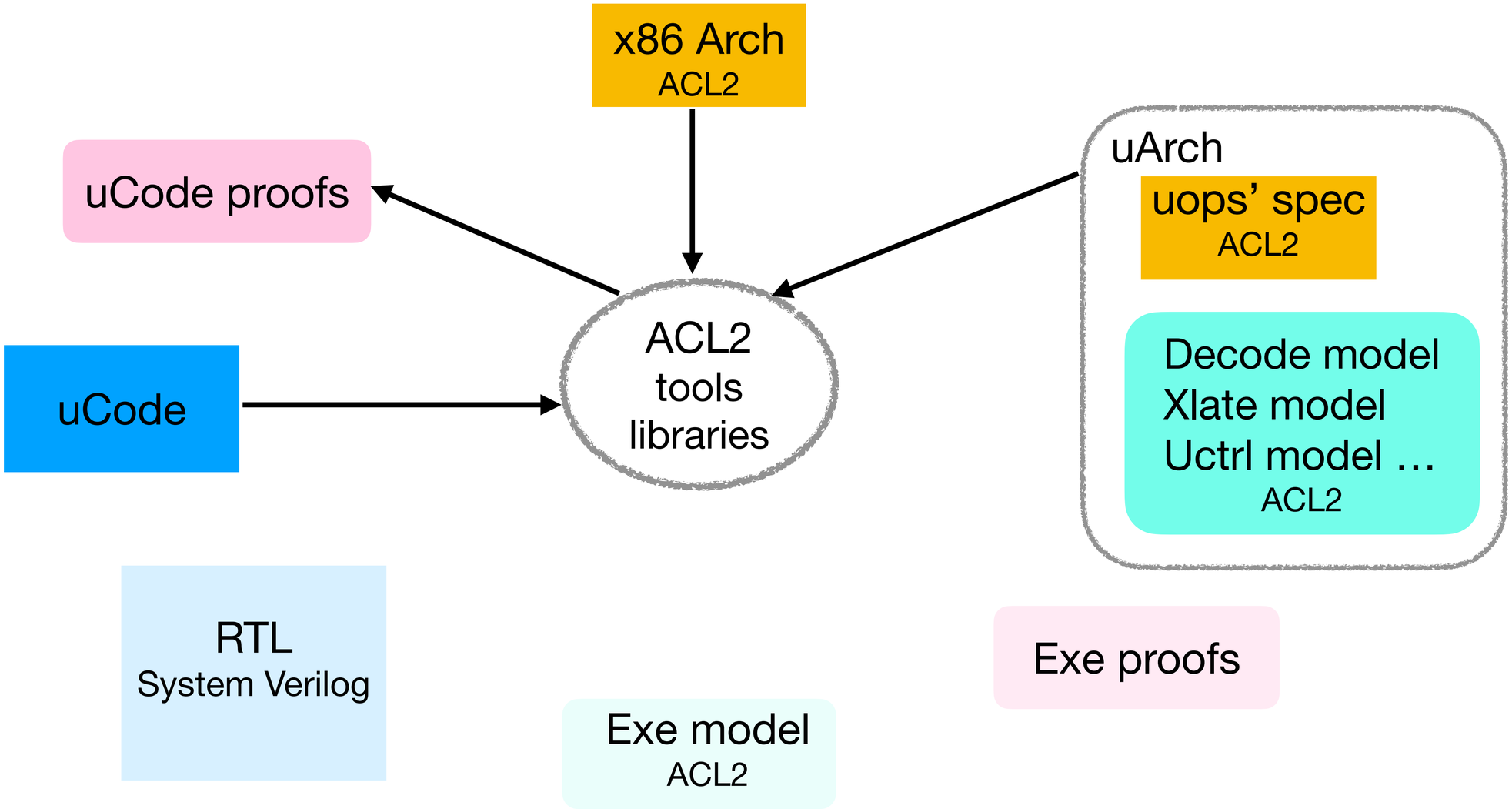
Interdependence of all proof artifacts



Failing regressions mean there is:

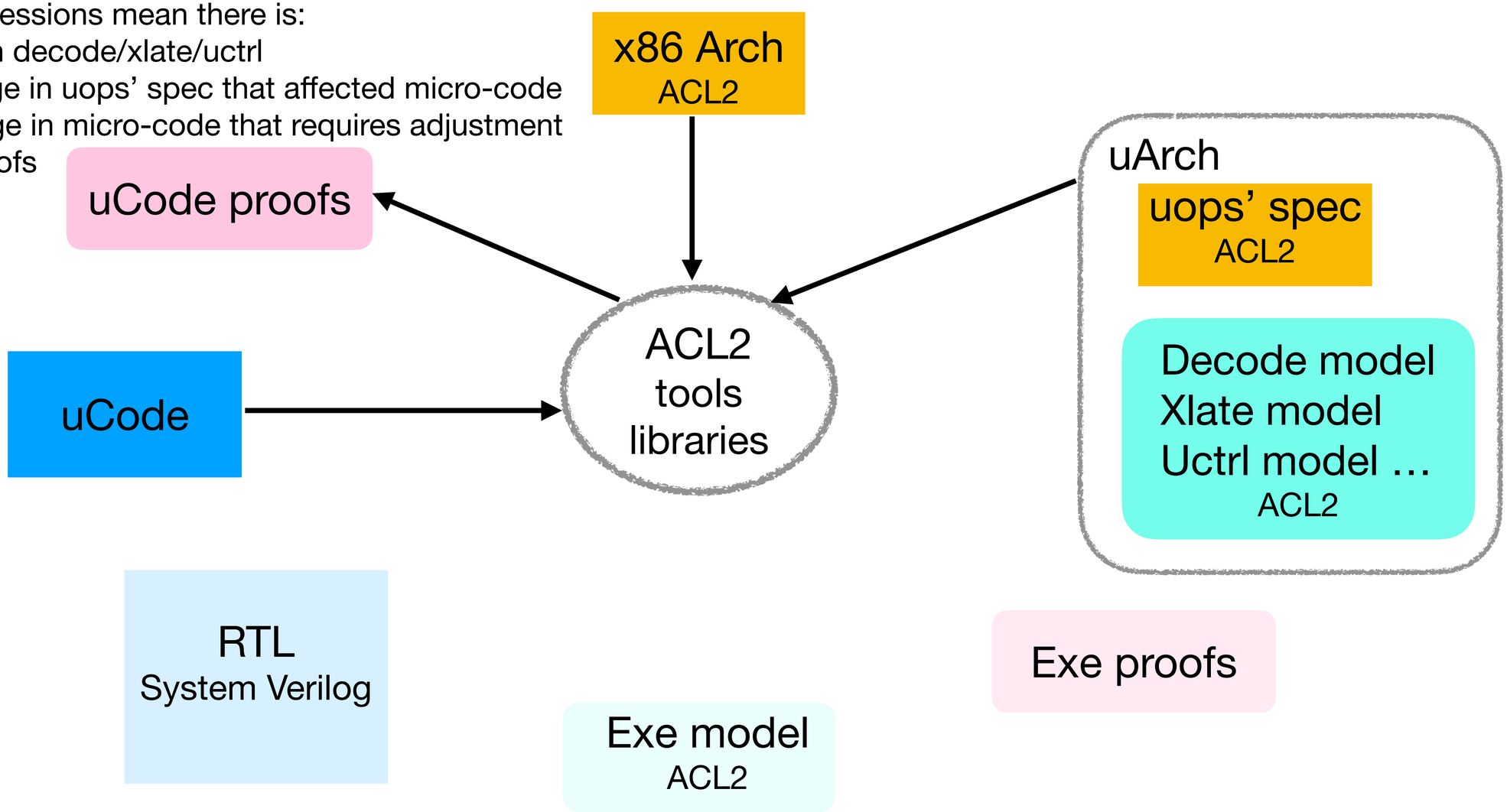
- a bug in design
- a change in design
- a change in the uop spec

Interdependence of all proof artifacts

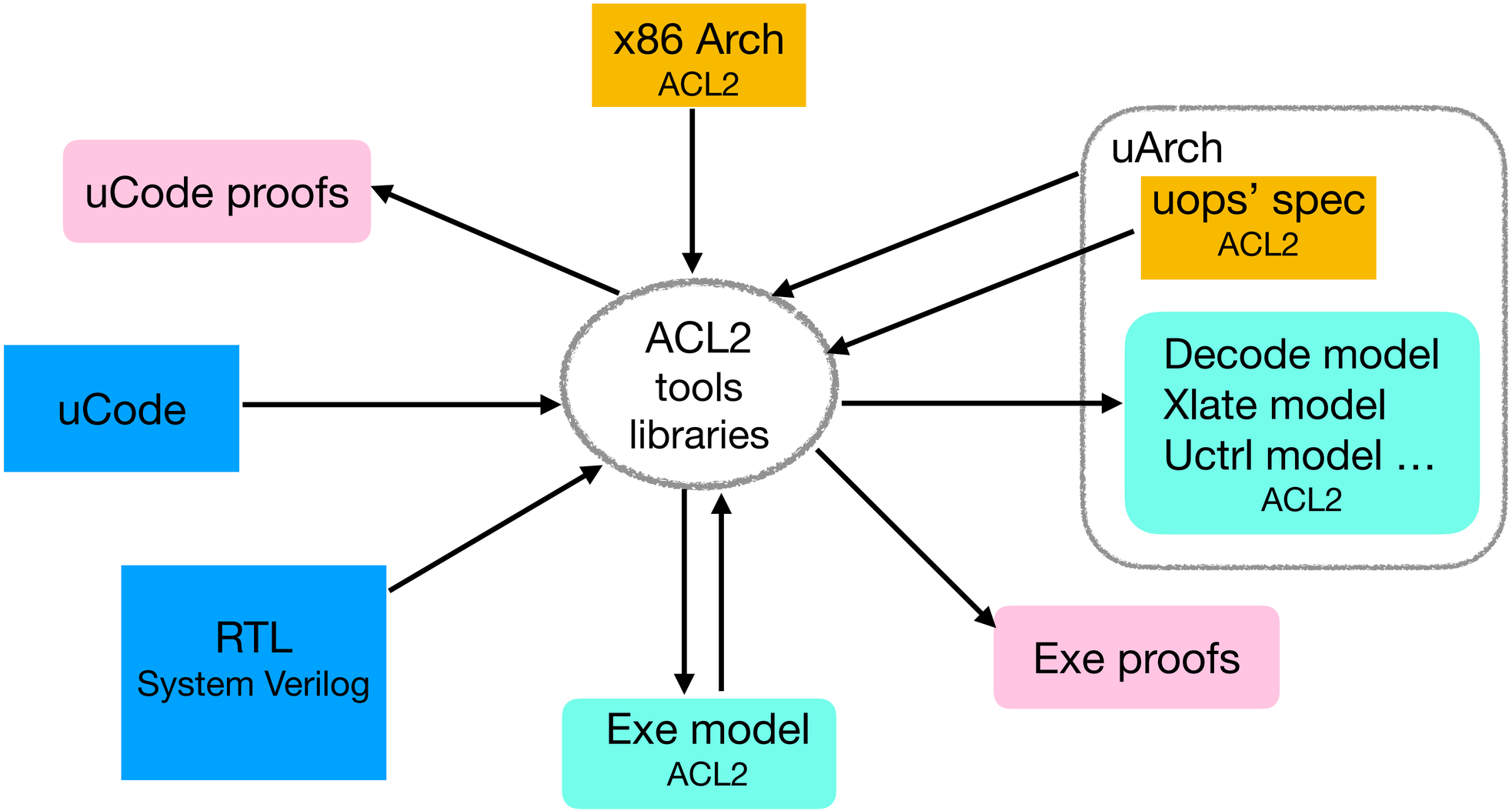


Interdependence of all proof artifacts

Failing regressions mean there is:
- a bug in decode/xlate/uctrl
- a change in uops' spec that affected micro-code
- a change in micro-code that requires adjustment of FGL proofs

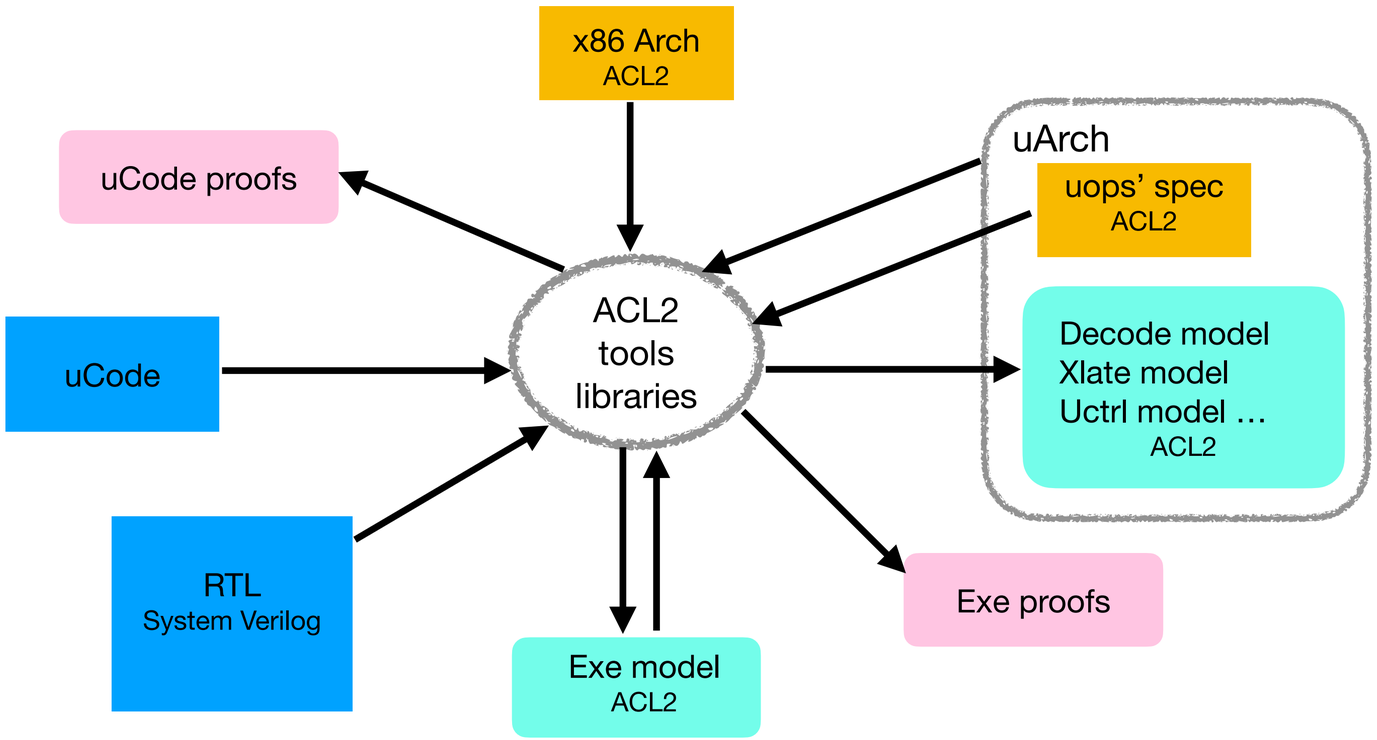


Interdependence of all proof artifacts



Regressions

- triggered by changes
  - changes in ACL2 or our tools
  - changes in micro-architecture
  - changes in design
  - changes in micro-code
- recurrent
- invoked manually



Conclusion

- industrial scale of FV requires robust tools and proofs
- build methodology that accounts for changes in the design, specification, and tools
  - many actors (logic team, ucode team, ACL2 team,…)
  - make specification reusable (generality, extensibility, implicit specification)
  - choose reliable tools
  - build and maintain extensive regressions suite
- interdependence of our proofs and tools enforces consistency
-  and  are our friends

Thank you!