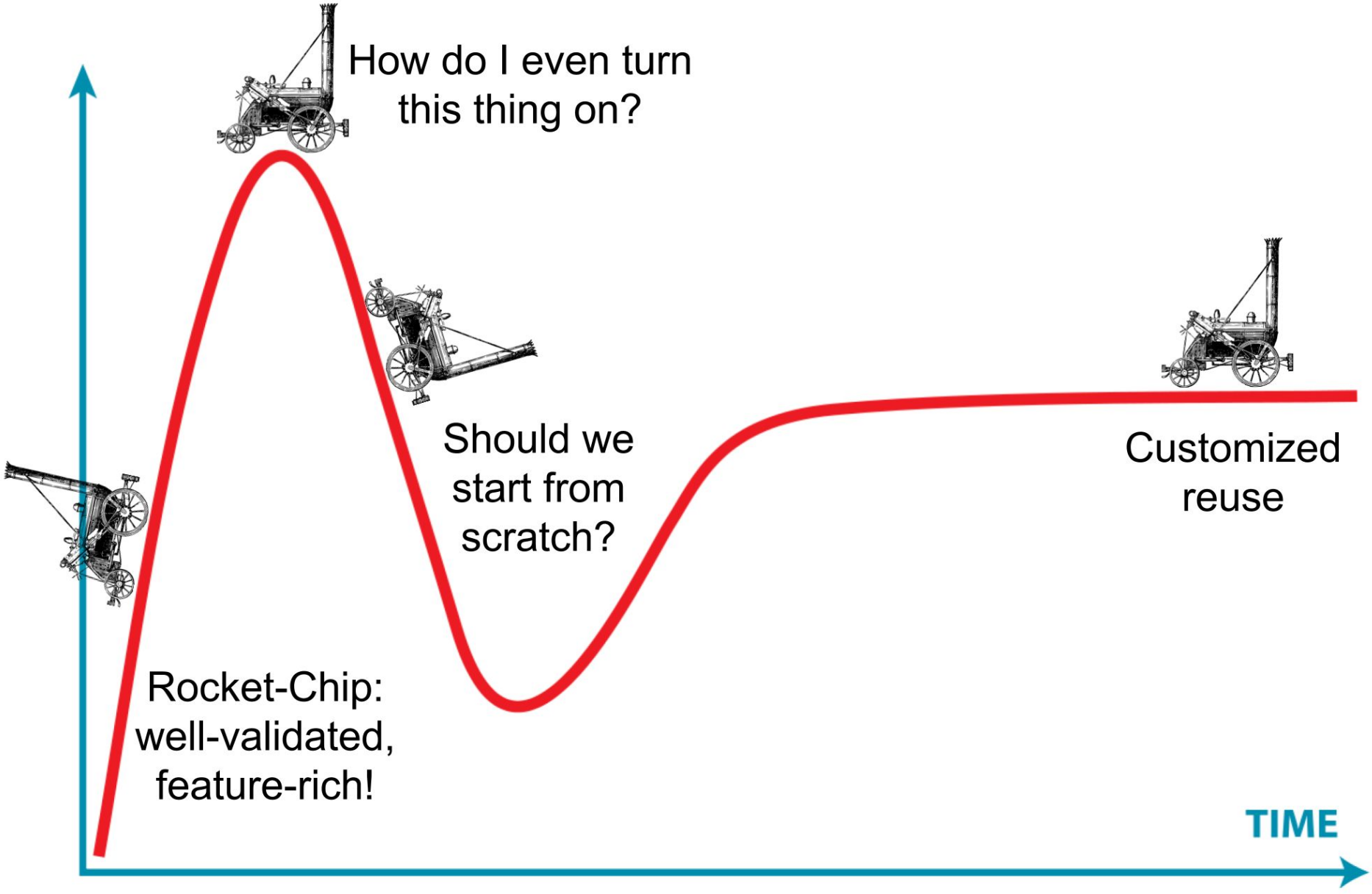
# Rocket Engines

## Easy, Custom RISC-V Cores Through Reuse

**Albert Magyar**
**UC Berkeley / Google**

Don Stark
Google

# The cycle of Rocket-chip reuse



# Why so many RISC-V cores?
# (Or: why not reuse Rocket?)

Good reasons:

- Desire to match interfaces to “drop in” for existing core
- Ability to tailor microarchitecture to custom extensions
- The perennial demand for “tinier cores”
- Avoiding complexity of SMP chip framework

(Less) good reasons:

- Why not / Not invented here
- Fear of Chisel

# Pitfalls for customized cores

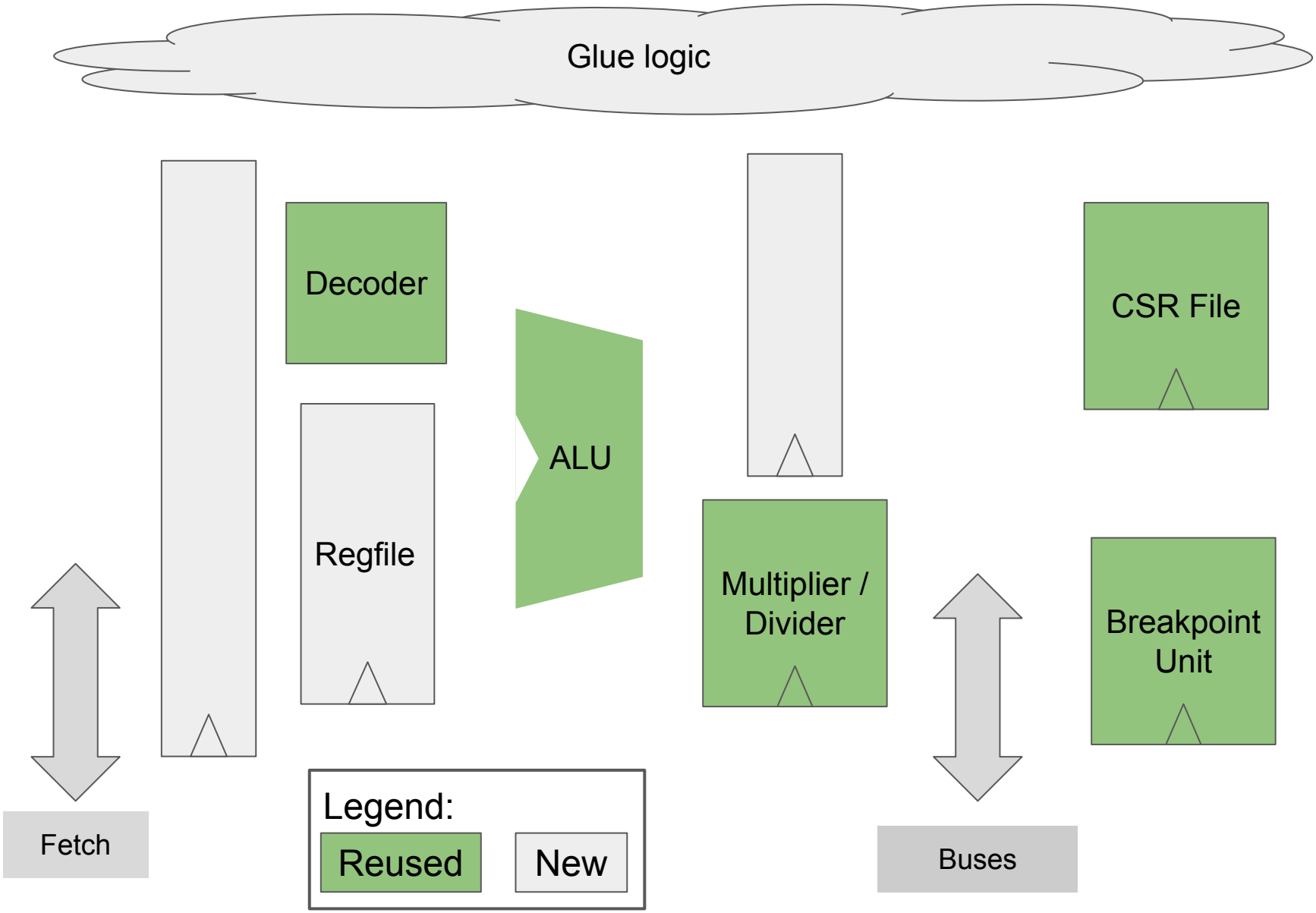
- Many bugs arise from ISA spec misinterpretation
  - Often more than in the “special sauce” of a particular implementation’s microarchitecture
- Rocket has a well validated privileged implementation
  - Especially relevant for “far corners” of the spec
  - Why throw this away?
- Maintenance under a (currently) unfrozen spec
  - Has effectively killed multiple open-source projects

Can design reuse be pervasive in standalone cores?

# Don’t reuse too little or too much

- Rocket is built from simple, well-defined subcomponents
  - Complexity at top level to hit high (scalar) IPC
- Steep learning curve comes from rocket-chip generator
  - Parametrization
  - Coherent caches & NoC
  - Maximally flexible generator with myriad features
- Reuse subcomponents to build microcontroller core
- Avoid using top-level rocket-chip framework at all
  - Major deviation from Z-Scale, BOOM

# Idea: Stitch together reusable Rocket components



# “Big 3” components

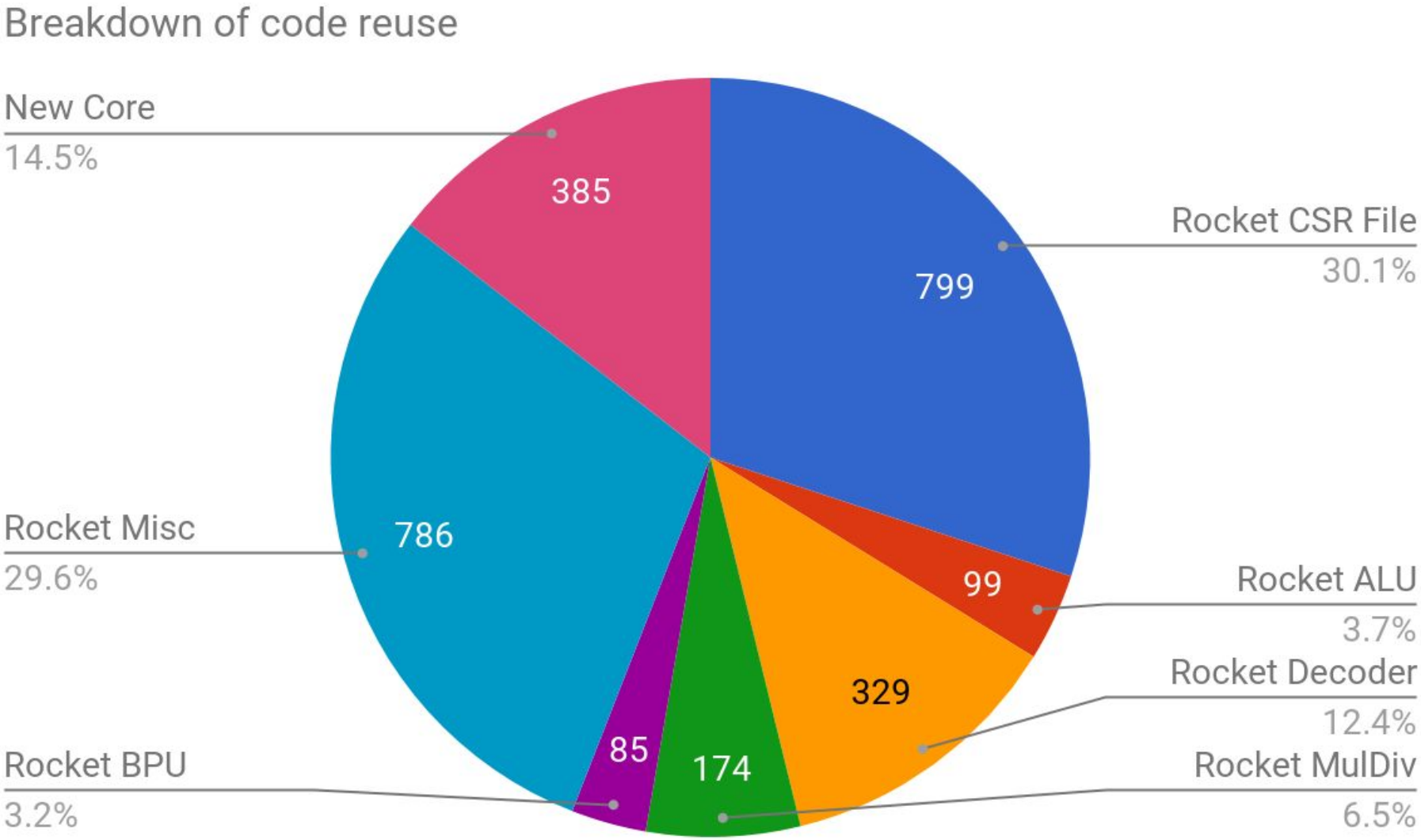
Well-defined, flexible blocks fit in an incredibly diverse array of microarchitectures with *no modifications*

- CSR file
  - More lines of code than simple in-order pipeline “top”
  - Provides a validated privileged implementation
- Decoder
  - Provides microarchitecturally-agnostic signals
- RISC-V Compressed (RVC) expander
  - Functional mapping rigidly defined by ISA

# New RISC-V core IP: “BottleRocket”

- Classic three-stage pipeline
  - Similar microarchitecture to Z-Scale, V-Scale
- Separate fetch and dmem/peripheral buses
- Implements RV32IMC
  - Constant-time, multi-cycle multiplier/divider
- Privileged architecture 1.10
  - Machine/user privileges, PMP
- Enhanced support for internal / external clock gating
- Generator produces a single, easy-to-connect tile

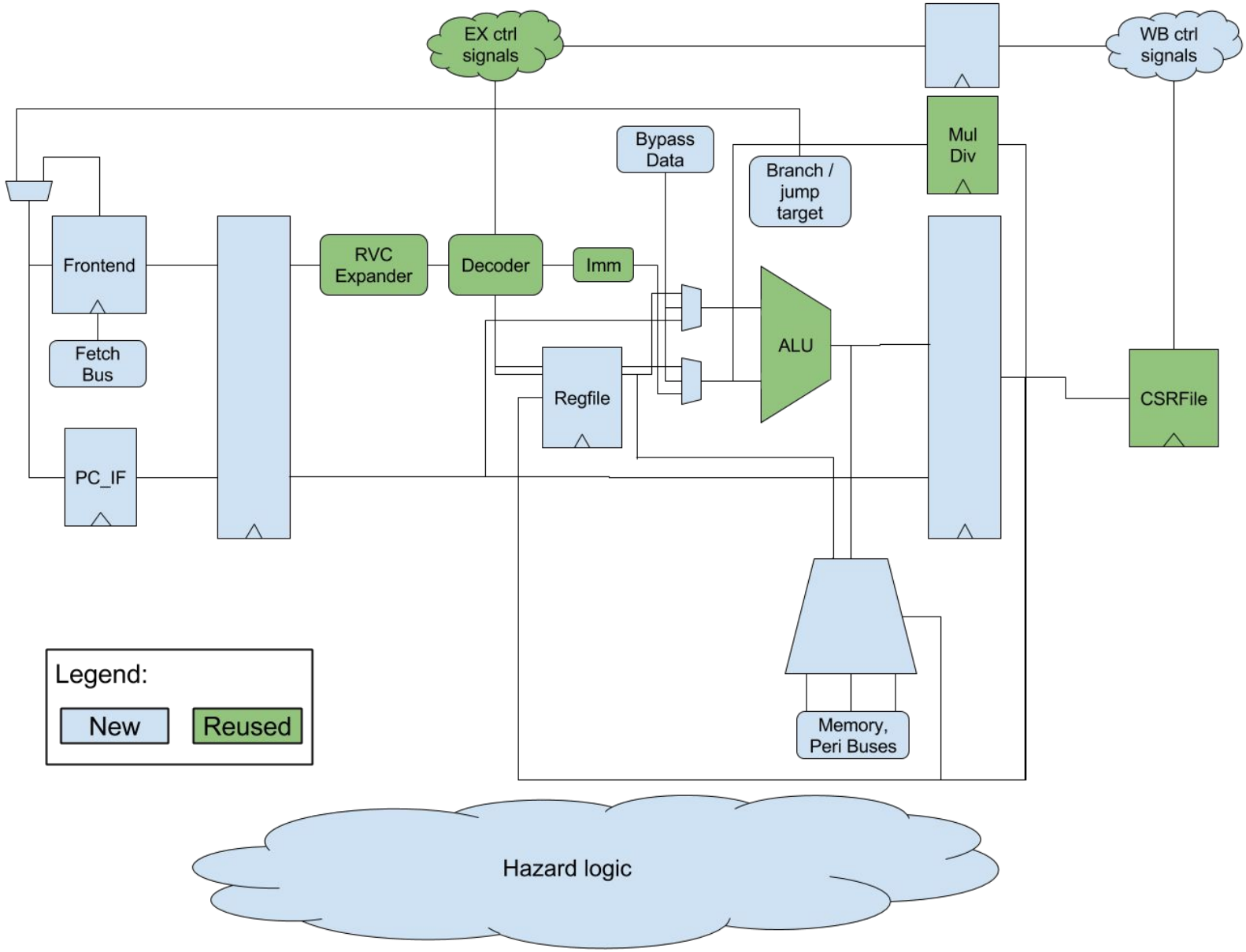
# Quantifying reuse



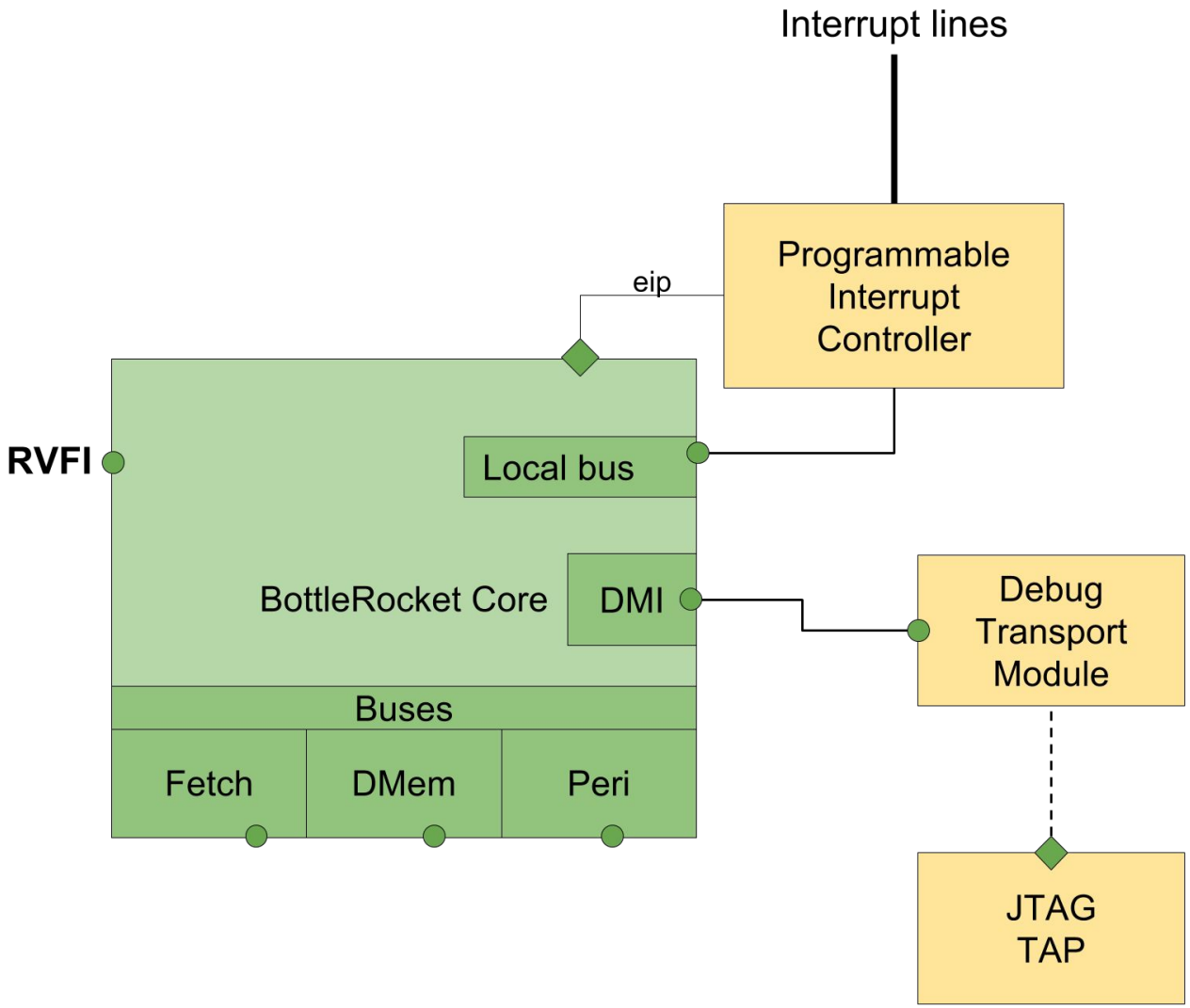
# Debug, Test, and Platform Features

- Supports SiFive debug spec v0.13
  - Access register, halt control, single step
  - System bus access
  - Custom debug control module, JTAG DTM
- Includes RISC-V Formal Interface (RVFI) trace port
  - Ongoing integration with riscv-formal tool suite
- Native & debug breakpoint support
- External interrupt controller for *n* interrupts
  - Single hart, small footprint

# Simplified BottleRocket pipeline



# BottleRocket Tile



**Reuse Delivers:**
Chip-level integration tests in weeks, not months

# Status & Roadmap

- Open-sourcing effort underway
  - Google now has >72,000 employees…
- Ongoing integration with riscv-formal
  - Starting with RVFI execution monitor

Questions?
Find me on the break!