Testing

Tapeout Post-mortem and Planning

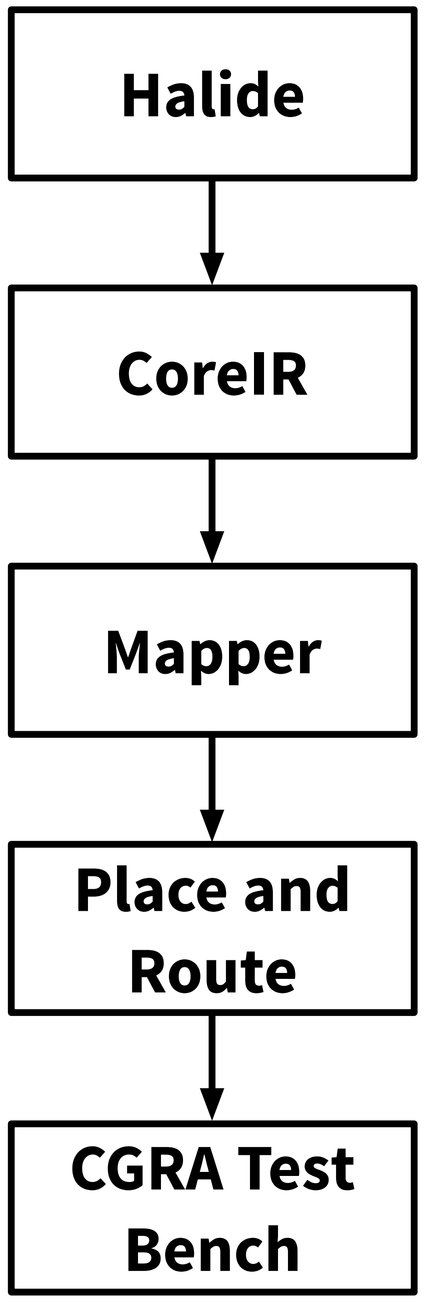
AHA Retreat Summer 2018

What we did: Unit Testing

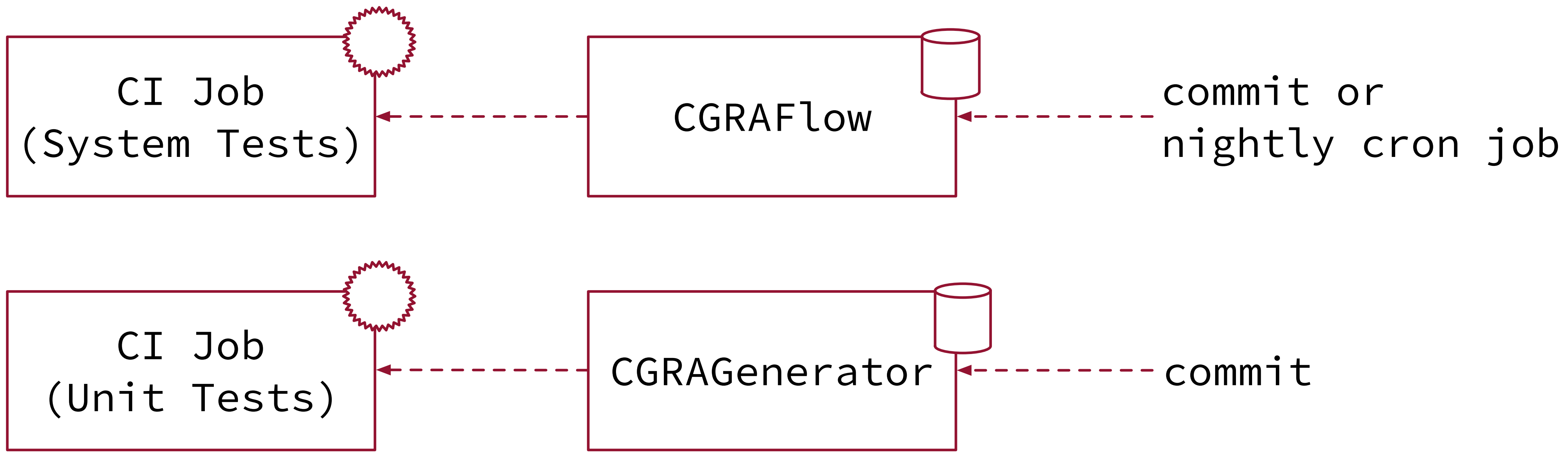
- **Unit Test** - a single RTL (verilog) module tested in simulation (verilator or ncsim)
- **PE** - simulation checked against functional model in Python
  - Exhaustive tests of operations up to 4-bits wide, random tests up to 16-bits wide
  - Exhaustive tests of condition flags, LUT, debug triggers, and input modes
  - Tests run any time a change is pushed to the CGRAGenerator repository
- **Memory** - Used Verilator's C++ direct programming interface for testing
  - SRAM Mode - Verified various combinations of writes and reads with random data
  - Linebuffer Mode - Verified writing random values and reading delayed values

What we did: System Testing

- Tests the entire CGRA design in RTL simulation (verilator)
- Halide CPU backend used to generate reference output images
- Automatically runs nightly and when changes are pushed to the CGRAFlow repository
- Franken-flow with manual steps used during late stage of tapeout
    - Bitstreams and reference output generated by one person using standard flow environment, sent to another person to be used with ncsim test bench



CI Architecture



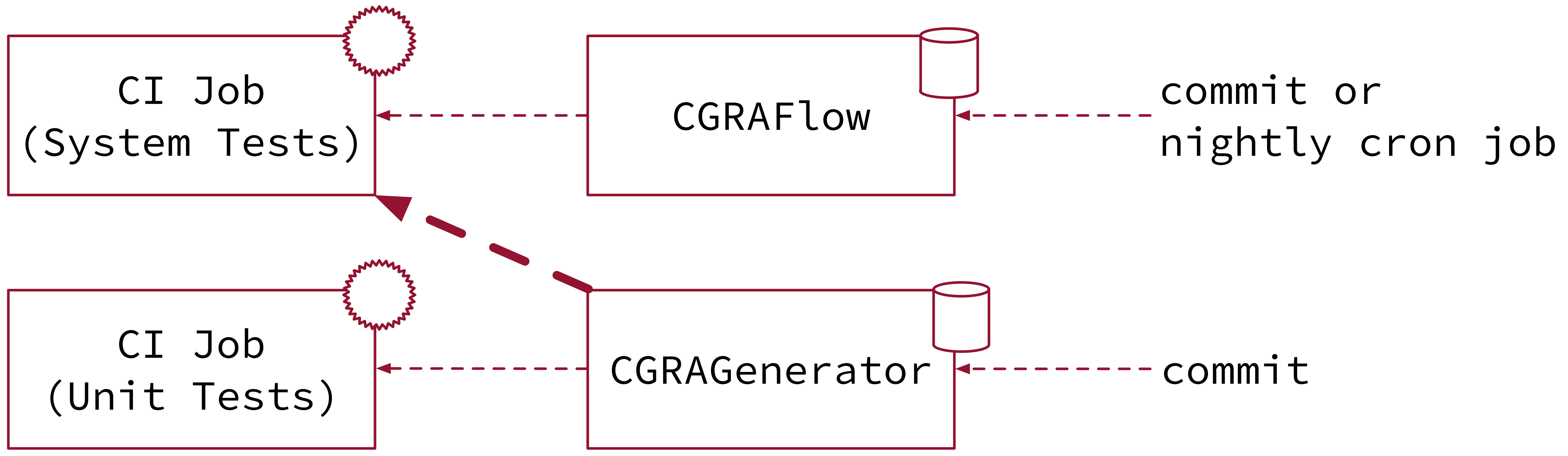
What went well

- Unit and System tests running on CI (automatically run when changes are pushed to repository)
- Tested PE and memory functionality in isolation
- Tested functionality of the entire design
- Tests can be reused for next generation of the chip
- Testing infrastructure was developed and can be reused (will be easier to write additional tests)

What could be improved

- Design not covered fully with unit tests (coverage wasn’t measured)
- Verification of synthesized design was manual (forgot to run unit tests)
- Didn’t perform integration tests with various sub-systems (single PE and Memory)
- Friction in integration
    - System tests (CGRAFlow) not run often enough (only nightly or upon commit)
    - Bugs appeared late in integration attempts (tests not running often enough)
    - Build times were slow (~30 min)
- Tests did not transition smoothly to commercial simulator

Improving CI



Next Steps

- Testing Infrastructure - make it easier to write more tests
  - Automate measuring of coverage, work towards 100%
    - Which coverage metric(s)? Toggle, source line, single/coupled state machine
  - Parameterized unit tests with automatic test case generation from functional model
  - Support integration tests for sub-system interfaces
  - Make tests portable across simulators
- CI Infrastructure - make it easier to integrate changes across project code bases
  - Run tests more often (ideally some continuously)
  - Improve build times by migrating to better infrastructure