Another approach to runtime checking

- Typical runtime checking is by duplicating entire CPU
  - Expensive in power, area
  - No protection from design errors
  - Does not survive permanent faults
- Last time we saw an approach that leveraged SMT
  - Somewhat better in power and area, but more complex
  - Still no protection from design errors
  - Still doesn’t survive permanent faults
- DIVA: custom-design a checking module
  - Simple, small addition to commit stage
  - May be able to formally verify
  - Fabricate for extra robustness
  - Can take over execution on permanent fault
  - Authors claim negligible performance hit
Another approach to runtime checking

- Typical runtime checking is by duplicating entire CPU
  - Expensive in power, area
  - No protection from design errors
  - Does not survive permanent faults

- Last time we saw an approach that leveraged SMT
  - Somewhat better in power and area, but more complex
  - Still no protection from design errors
  - Still doesn’t survive permanent faults

- DIVA: custom-design a checking module
  - Simple, small addition to commit stage
  - May be able to formally verify
  - Fabricate for extra robustness
  - Can take over execution on permanent fault
  - Authors claim negligible performance hit
Details

Traditional Out-of-Order Core

- IF ID REN
- ROB
- CT
- EX

out-of-order execute
in-order issue
nonspec results
in-order retirement

DIVA Core

- IF ID REN
- ROB
- EX

out-of-order execute
in-order issue
instructions with inputs and outputs

DIVA Checker

- WT
- CHK CT

Shaded components must be verified for correct operation

- Replaces commit stage of traditional OOO pipeline
- Fed all instructions with inputs and outputs
- CHK stage repeats all calculations before allowing commit
- On error, replaces erroneous result with its own calculation and restarts main processor
- WT (watchdog timer) ensures forward progress
What is validated

DIVA Core

DIVA Checker

▶ DIVA assumes accurate:
  ▶ Decoded instructions arriving at CHK stage
  ▶ Values fetched from memory
  ▶ Values fetched from architectural registers

▶ Issues its own reads to memory and register file
▶ Validates address calculation
▶ Validates all arithmetic
▶ Validates order of operations
More efficient than paired CPUs in lockstep...

- Checker pipeline does much less work than a second CPU
  - No inter-instruction dependencies
  - No second register file
  - No cache

- Main CPU can rely on DIVA to catch errors—can be simplified

- Only data fetches from memory are duplicated

- 0.3% slower than unchecked CPU (with extra data-cache memory port)

Shaded components must be verified for correct operation
Other advantages

- Traditional Out-of-Order Core
- DIVA Core
- DIVA Checker

Shaded components must be verified for correct operation

- **For sure:**
  - Can recover from permanent faults in core
  - Can even recover from completely dead core
  - Core only needs design validation for performance
  - Scales better to multicore designs

- **More speculative:**
  - Practical to build checker with bigger transistors, higher voltages, more robust circuitry
  - Practical to formally verify checker (?)
  - Could use fault rate to tune clock speed, temperature
Disadvantages and handwaves

- Correct behavior totally dependent on checker
- Pipeline lengthened
- Needs ECC register file and caches, with lots of ports
- Performance sims assume checker ALU is as fast as core ALU...
- correctness benefits assume checker ALU is simpler than core ALU