Post-silicon validation and challenge

- Post-silicon validation is crucial to stimulate corner cases that are unverified pre-silicon validation.
- Random pseudo instruction tests.
- Inefficient instruction selection.
- Assumption: only architectural state is observable at the end of test.
- Bug-masking can happen due to limited observability.

Baseline: information-flow tracking

- Adopting information-flow tracking used in software security analysis.
- Treat each instruction’s target (e.g., register, memory) as target source.
- Checking for information reachability.
- Analyzing use-def chains along instruction sequence.

Limitation of information-flow tracking

- Buggy information can be masked by subsequent instructions depending on how instructions propagate the information.

Bug model

- Example test: $d = 0x00000000$
- Actual results depend on register $r2$ value.
  - $d = 0x00000000$ (true)
  - $d = 0x0000FFFF$ (false)

Experimental evaluation

- Two ISAs evaluated: IBM Power and DEC Alpha
  - IBM Power: approximately 4,000 source-target pairs
  - Power generator: Threadmill (DAC’11).
  - DEC Alpha: 218 source-target pairs (subsets of instruction).
  - Test generator: in-house fully-random test generator.

Bug model

- Minimizing micro-architectural bugs by slightly altering architectural state.
- Random erroneous value update for architectural state (register / memory location).

BugMAPI execution time

- BugMAPI (11.3 seconds) vs. dynamic analysis (~25,000 seconds): 3-orders of-magnitude speedup.

Application: bug-masking reduction

- Reducing bug-masking incidence in random instruction tests.
- Two steps: Identification and patching.
  - (1) Identifying mask-causing instructions by using BugMAPI.
  - (2) Patching mask-reducing instructions.

Discussion:

- Sources of inaccuracies:
  - Coarse granularity of information-flow tracking.
  - Approximated probability computation.
  - Lack of consideration of instruction blocks (e.g., address calculation).
- Bug model dependency:
  - 5 models considered: random value overwrite, single bit flip, binary complementation, missing update, spurious update.
- Multithread programs:
  - Non-deterministic execution, inter-thread data dependency.
- Profiling frequency of inter-thread data dependency.