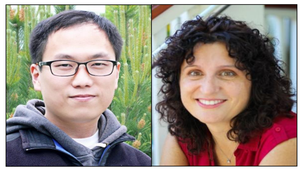


MTraceCheck: Validating Non-Deterministic Behavior of Memory Consistency Models in Post-Silicon Validation

Doowon Lee and Valeria Bertacco, University of Michigan

Center for Future Architectures Research

CFAR

STARnet

Theme 2384.008 – Threats

Task 8.1 – Extreme scaling for design and verification

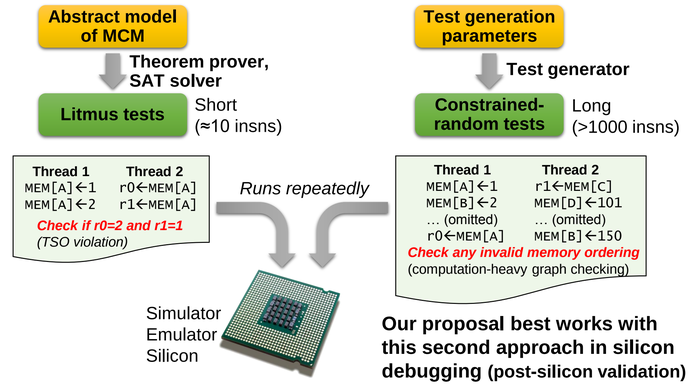
1. Memory Consistency Models

- Memory consistency models specify observable orderings of memory accesses
  - Sequential consistency (MIPS R10K), total store order (x86, SPARC), weakly-ordered models (ARM, IBM POWER), etc.
- Multi-threaded programs experience various memory ordering patterns during their program executions
  - Due to μ-arhictectural optimizations related to out-of-order execution, coherent caches, multiple memory channels, on-chip interconnect etc.
  - Memory ordering violations may cause incorrect execution results in multi-threaded programs
  - Many hard-to-find bugs (errata bugs) are related to memory behaviors in multi-threaded programs

Research goal:
Improving the efficiency of memory consistency validation

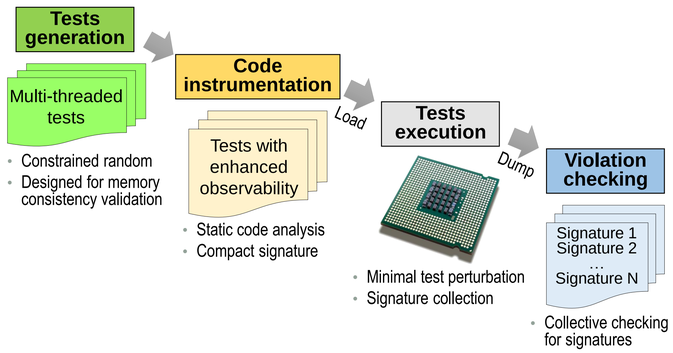
2. Prior Memory Consistency Validation

- Formal verification approach [Alglave'14, Lustig'17, etc.]
- Constrained-random testing approach [Hangal'04, Elver'16, etc.]



Our proposal best works with this second approach in silicon debugging (post-silicon validation)

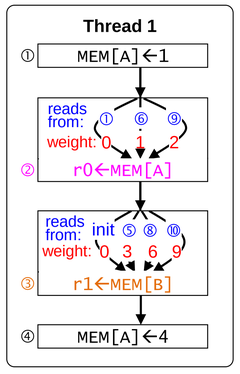
3. MTraceCheck Workflow



4. Memory-Access Interleaving Signature

- Memory-access interleaving signature encapsulates loaded values over the execution of test program

Thread 1	Thread 2	Thread 3
① MEM[A]←1	⑤ MEM[B]←5	⑧ MEM[B]←8
② r0←MEM[A]	⑥ MEM[A]←6	⑨ MEM[A]←9
③ r1←MEM[B]	⑦ r2←MEM[A]	⑩ MEM[B]←10
④ MEM[A]←4		



Step 1. Identify all **reads-from** relationships for each load operation

Step 2. Assign a **unique weight** for each reads-from relationship*

* Inspired by control flow path profiling (Ball and Larus, "Efficient path profiling" [MICRO'96])

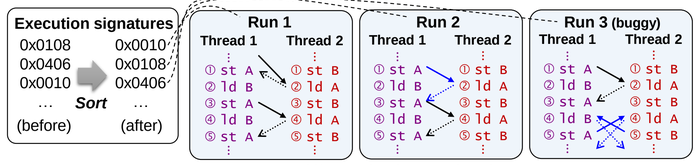
Step 3. **Accumulate weights** observed at runtime

e.g. ② from ⑨ and ③ from ⑧ ➔ Signature = **8**
(weight: 2) (weight: 6)

5. Collective Graph Checking

- Collective graph checking reduces result-checking computation by exploiting similarity among graphs from repeated runs

Step 1: Find similar graph We use memory-access interleaving signature to estimate graph similarity



Diff graphs in a pair-wise manner

Step 2: Re-sort topological order We re-sort only the region enclosed by backward edge(s)

6. Non-Determinism in Memory Interleaving

Systems under validation

	x86-64 (x86-TSO)	ARMv7 big.LITTLE (weakly-ordered)
Processor	Intel Core 2 Quad Q6600	Samsung Exynos 5422
Operating frequency	2.4 GHz	800 MHz
Number of cores	4	4 (Cortex-A7) + 4 (Cortex-A15)
Cache architecture	32 + 32 kB (L1), 8 MB (L2)	32 + 32 kB (L1), 512 kB + 2 MB (L2)
Cache configuration	Write back (both L1 and L2)	Write back (L1), Write through (L2)

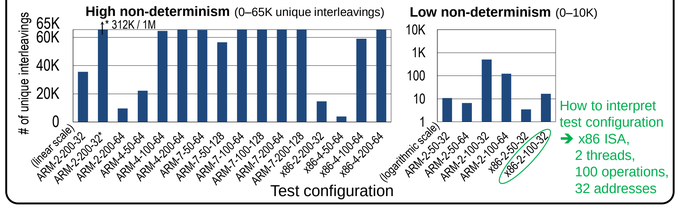
Test program configurations

Number of test threads	2, 4, 7
Number of memory operations per thread	50, 100, 200
Number of distinct shared memory addresses	32, 64, 128

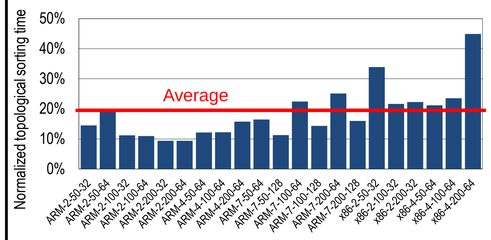
21 test configurations (each with 10 randomly generated tests)

Measured non-determinism in memory-access interleaving

We ran each test 65,536 times and measured the number of unique memory-access interleavings



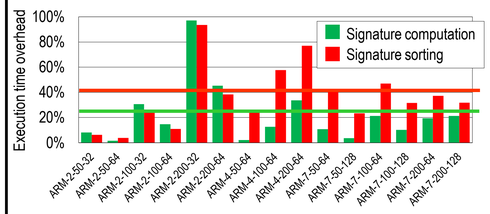
7. MTraceCheck Validation Performance



Violation checking time

– 81% reduction in graph-checking time, compared to individual graph checking

– High benefit in relaxed memory model (ARM)

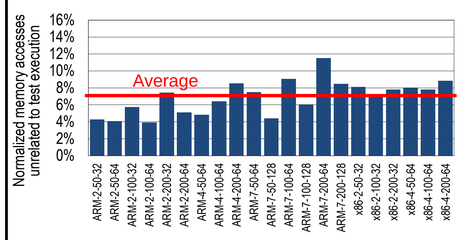


Runtime overheads

– 22% / 38% runtime increase due to signature computation / sorting, respectively

– This increase is marginal (approximately 5 times) compared to time saving in violation checking

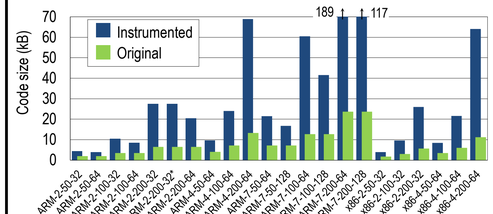
8. MTraceCheck Instrusiveness



Memory-access perturbation

– 93% reduction in memory accesses unrelated to test execution

– Higher benefit in less contentious tests



Code size

– 3.7× larger than original test

– Small enough to fit in L1 caches

9. Conclusions

- MTraceCheck improves the efficiency of memory consistency validation by using (1) **memory-access interleaving signature** (93% reduction in logging memory operations) and (2) **collective graph checking** (81% reduction in graph-checking time)

10. Technology Transfer

- Industry interaction and internship
  - IBM Research in Israel (Summer 2015)
  - Post-silicon validation research is motivated from this internship
- An extended version has been presented at the 44th International Symposium on Computer Architecture (ISCA 2017)
- Source code available at: https://github.com/leedoowon/MTraceCheck