BugMD: Automatic Mismatch Diagnosis for Bug Triaging

Biruk Mammo, Milind Furia, Valeria Bertacco, Scott Mahlke, and Daya S. Khudia

Center for Future Architectures Research (CFAR)

STARnet

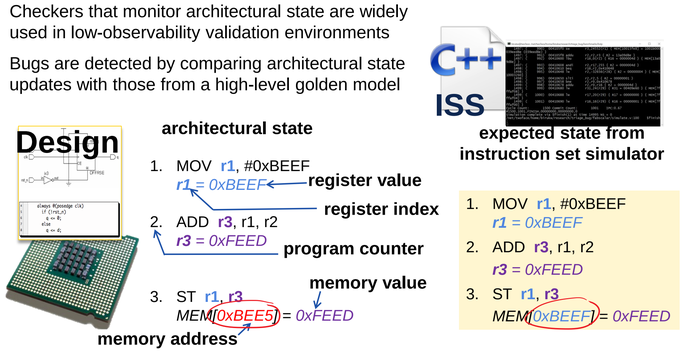
EECS ECE CSE

MICHIGAN

1. Motivation

Checkers that monitor architectural state are widely used in low-observability validation environments

Bugs are detected by comparing architectural state updates with those from a high-level golden model



***0xBEE5 ≠ 0xBEEF* : Mismatch!**

2. Limitations

- Diagnosis is difficult
  - Several thousands of cycles between bug occurrence and manifestation
  - Low-observability, limited information
- Simulation stops typically after first mismatch
  - Patterns may emerge if simulation continues to collect more mismatches

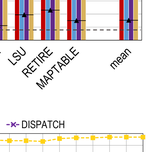
3. (0x0C) ST r1, r3
   MEM[0xBEE5] = 0xFEED
4. (0x10) MOV r2, #4
   r2 = 0x0004
5. (0x14) ADD r4, r3, #1
   r4 = 0xFEE4
6. (0x18) BR +0x10
   PC = 0x0020

Instr. count	mismatch type	instr. type	design value	ISS value
3	MEM ADDR	MEM	0xBEE5	0xBEEF
5	REG VALUE	ALU	0xFEE4	0xFEEE
6	PC VALUE	CONTROL	0x0020	0x0028

What happens when design execution diverges?

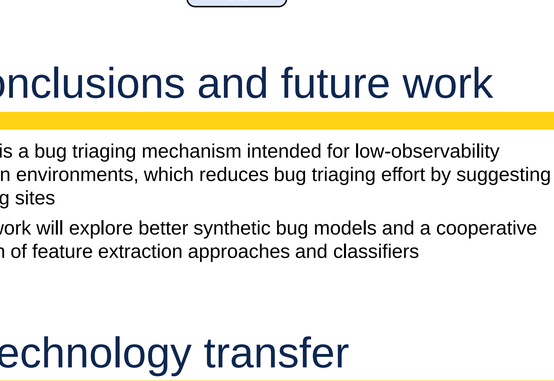
How do we automatically identify patterns?

How do we get enough data to learn multiple patterns?

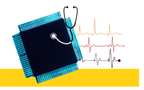


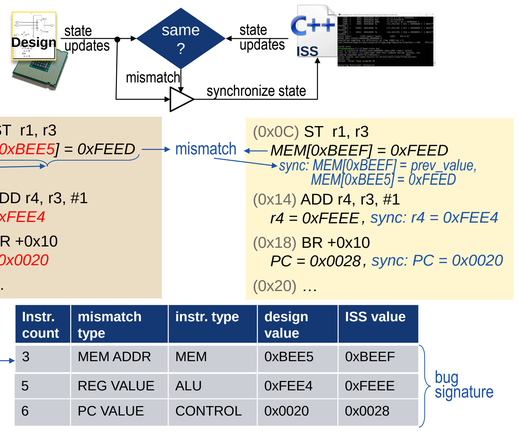
3. BugMD: key contributions

- Collects multiple, independent symptoms
  - Runs simulation past first mismatch
  - Avoids divergence by synchronizing ISS with design
- Learns patterns, without detailed µArch. knowledge
  - Generates feature vectors from multiple mismatches
  - Trains a machine-learning classifier
- Generates synthetic training data
  - Automatically injects thousands of bugs into the design



4. Symptom collection

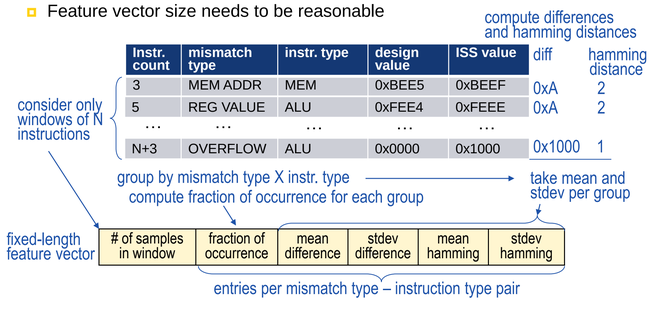




Instr. count	mismatch type	instr. type	design value	ISS value
3	MEM ADDR	MEM	0xBEE5	0xBEEF
5	REG VALUE	ALU	0xFEE4	0xFEEE
6	PC VALUE	CONTROL	0x0020	0x0028

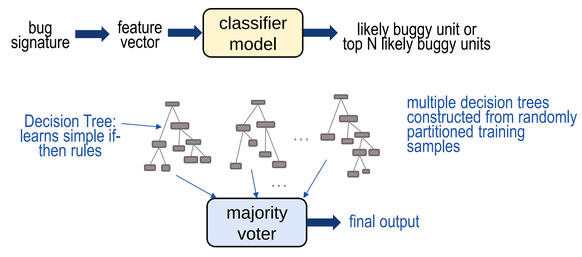
5. Learning patterns: features

- Unprocessed bug signatures are not amenable to machine learning
  - ML algorithms take fixed-size, real valued feature vectors
  - Bug signatures have arbitrary sizes, from 0 to millions of entries per simulation
  - Symptoms need to be converted to meaningful, real-valued features
  - Feature vector size needs to be reasonable



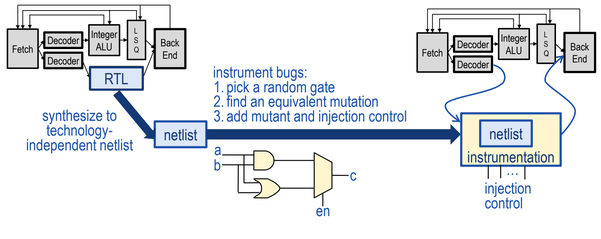
6. Learning patterns: classifier

- Investigated several machine-learning algorithms
  - Classic machine-learning classifiers (SVM, naïve bayes, etc.)
  - Multi-layer perceptron neural network
  - Convolutional neural network
- ***Random decision forest*** algorithm performed the best



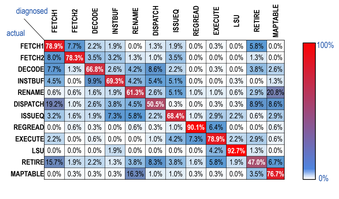
7. Injecting synthetic bugs

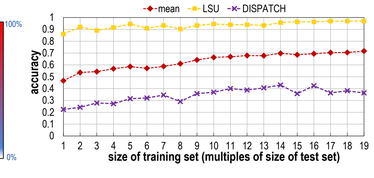
- Classifier model has to be trained on known bugs
  - Known buggy unit along with associated bug signature(s)
  - Large amount of training data required for learning
- We developed a synthetic bug model and injection framework
  - Random gate mutations to model bugs
  - Automatic bug instrumentation and flexible injection control during simulation



8. Results

- 4-wide, out-of-order FabScalar core, 12 design units
- 6 instruction types
- 34 mismatch types
- 10,000 instruction window, feature vectors of size 470
- 7080 synthetic bugs, over 40,000 bug signatures





9. Conclusions and future work

- BugMD is a bug triaging mechanism intended for low-observability validation environments, which reduces bug triaging effort by suggesting likely bug sites
- Future work will explore better synthetic bug models and a cooperative selection of feature extraction approaches and classifiers

10. Technology transfer

- Industry interactions
  - Dr. Daya S. Khudia was hired by Intel Corporation
- Publications/presentations
  - To appear in proceedings of ICCAD'16
  - Manuscript uploaded to SRC