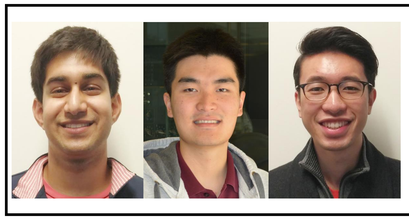


saveCHIMP: Application-aware Testbench for Chip Multi-Processors

Arjun Khurana, Dong-hyeon Park, Timothy Wong

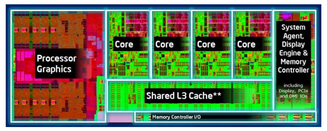
UNIVERSITY OF MICHIGAN

EECS 578 Correct Operation for Processors and Embedded Systems

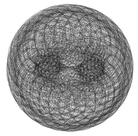
Motivation

Existing test generation solutions rely on random or directed tests that blindly explore the design space, which are inefficient.

If testing is not limited to a relevant pool of scenarios, resources spent on verifying complex systems in the future will grow without bound.



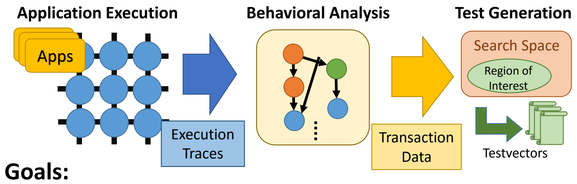
Intel Haswell i7-4700K



"Gemini" 3D Torus

saveCHIMP (save **CHI**p **M**ulti-**P**rocessor)

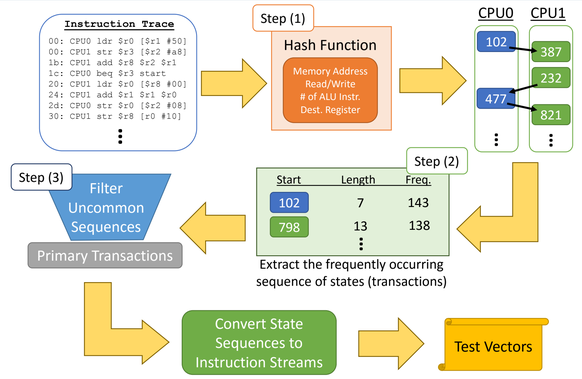
Test generation platform that generates tests based on the behavior of applications that are to run on the multiprocessor system



Goals:

- Characterize the execution space of the target application.
- Generate tests that exercise the same execution space as the target application.
- Speedup the validation process by generating tests with significantly less overhead than original program.

saveCHIMP Overview



Characterization and Filtering

A sample trace:
270 -> 370 -> 470 -> 578
270 -> 370 -> 478 -> 578
... (the rest of Window0)
312 -> 320 -> 370 -> 373 -> 427 -> 527 ->627
312 -> 101 -> 270 -> 215 -> 311 -> 427 ->627
... (the rest of Window1)

Source	Length	Freq.
270	4	852
312	7	734
...	...	...

Window	Most Common State
0	578
1	627
...	...

Instruction Trace

```
00: CPU0 ldr $r0 [$r1 #50]
00: CPU1 str $r3 [$r2 #a8]
1b: CPU1 add $r8 $r2 $r1
1c: CPU0 beq $r3 start
20: CPU1 ldr $r0 [$r8 #00]
24: CPU1 add $r1 $r1 $r0
2d: CPU0 str $r0 [$r2 #08]
30: CPU1 str $r8 [r0 #10]
```

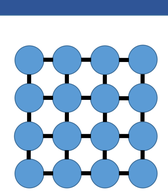
Filter identical sequences

A set of reduced patterns

Experimental Setup

Gem 5 Simulation Configuration

CPU	ARM, TimingSimple
Topology	4x4 Mesh
Coherency Protocol	MOESI CMP Directory
L1 Instruction Cache	4-way 32 KB
L1 Data Cache	4-way 32 KB
L2 Cache	8-way 1024 KB
DRAM	DDR3 1600MHz 4GB



4x4 mesh network

Target Application: Parsec's Blackscholes benchmark.

Random: Randomly generated stream of memory and ALU instructions.

Step (1): Tests based only on hashing data.

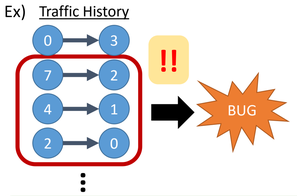
Step (1) + (2): Tests from hashing and characterization, but without filtering.

Step (1) + (2) + (3): Tests from full saveCHIMP process.

Bug Injection Model

Our Bug Model:
A specific sequence of source-destination network traffic.

Ex) Traffic History



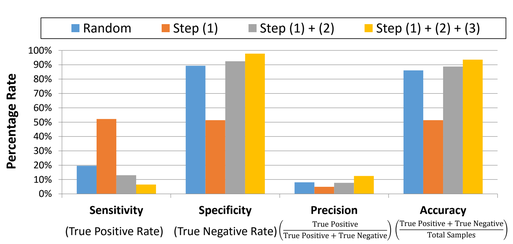
Random Bug Injection

- At each iteration, we inject 10 bugs of length 3 into the gem5 simulation platform.
- Whenever gem5 detects a sequence of traffic that matches one of the injected bugs, it flags that a bug has been triggered.

Goal of Our Bug Model:
To detect that two programs exercised similar sequence of network behaviors.

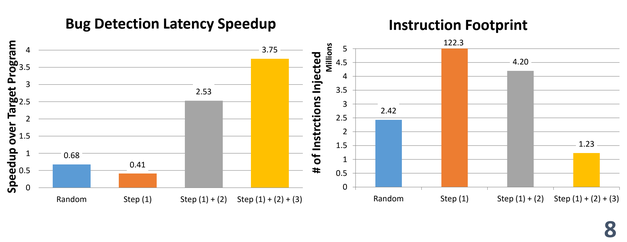
Experimental Results – Coverage

- Randomly generated testvectors act as the baseline to assess how well tests generated from saveCHIMP reflects the behavior of target program.
- Impact of applying each step of the saveCHIMP solution is shown.
- Final solution achieves 93% accuracy, due to its high true negative rate.
- Sensitivity and Precision are two areas that need to be improved upon.



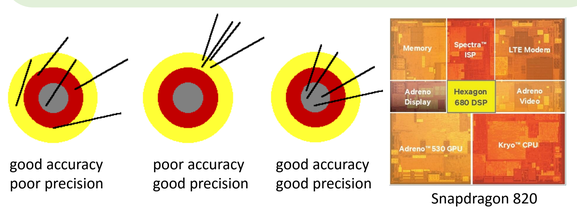
Experimental Results – Overhead

- Tests generated from saveCHIMP is able to detect bugs more quickly with a smaller instruction footprint.
- Transaction characterization and filtering steps of saveCHIMP significantly reduce the bug detection latency.
- Transaction filtering help reduce the size of the testvectors by more than 70%.



Conclusion/Future Work

- Improve on sensitivity and precision of test generation process, while maintaining accuracy.
- Incorporate network traffic behavior into state characterization
- Expand saveCHIMP to analyze and compare behaviors of multiple applications.
- Apply saveCHIMP to heterogeneous systems.



good accuracy
poor precision

poor accuracy
good precision

good accuracy
good precision

Snapdragon 820