Empowering Software Debugging Through Architectural Support for Program Rollback

Radu Teodorescu and Josep Torrellas
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

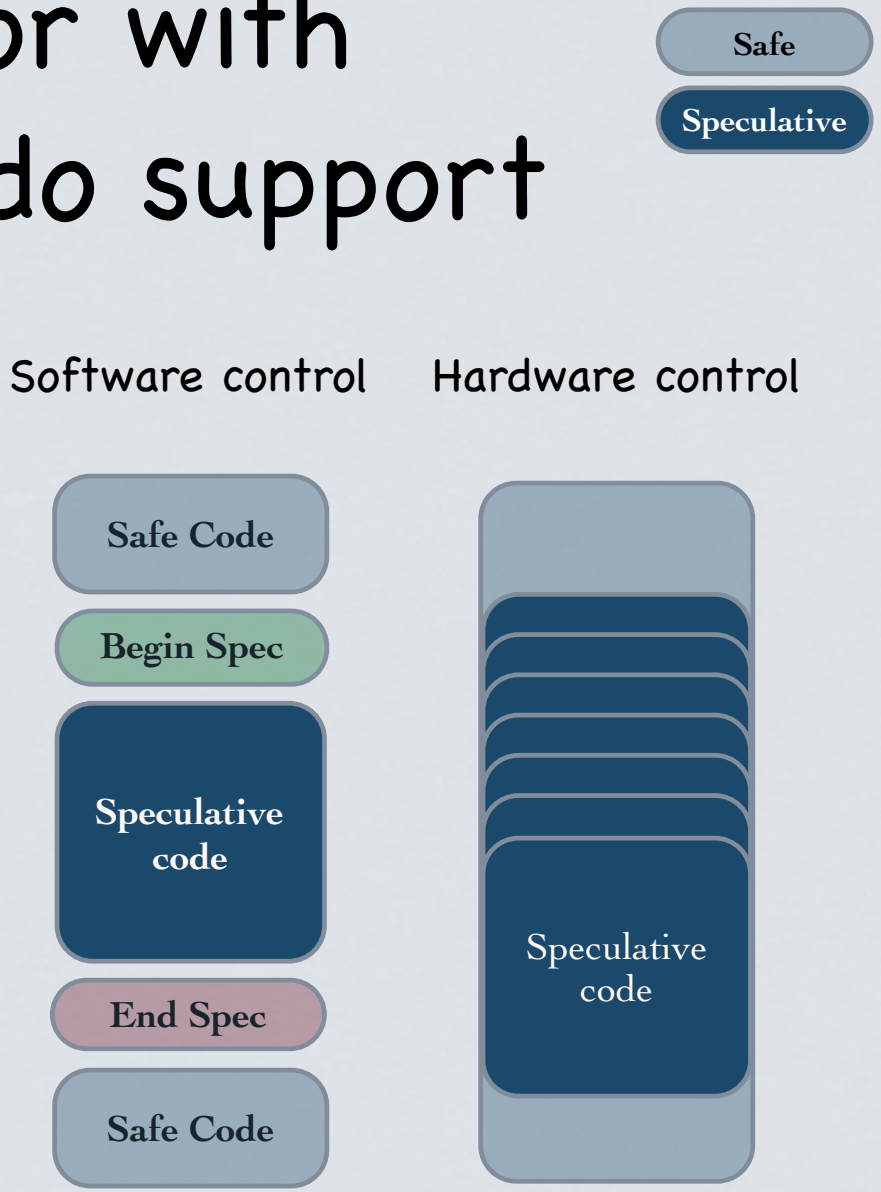
i-acoma group

Motivation

- Production software is hard to debug
- Need lightweight, continuous monitoring system
- We propose: hardware/software approach:
  - Architectural support for program undo
  - Monitoring and recovery from bugs in production systems

Processor with program undo support

- Rollback/re-play of large code sections
- Very low overhead
- Speculation control:
  - In software: spec control instructions
  - In hardware: dynamic sliding window



Contributions

- We implemented an FPGA-based prototype of a processor with undo support
- We show that simple hardware can provide powerful debugging tools
- We discuss possible applications to software debugging
- Initial assessment using buggy programs

i-acoma group

Debugging Production Code

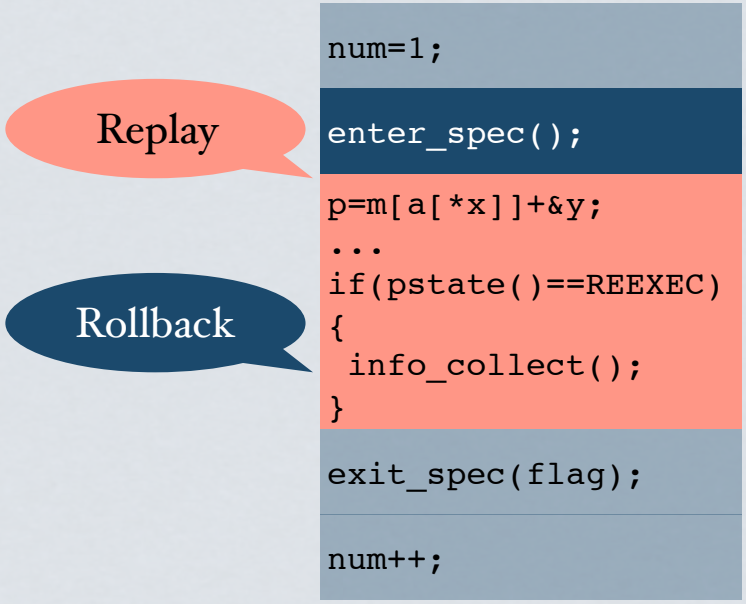
Original code

```
num=1;
...
p=m[a[*x]]+&y;
...
num++;
```

Instrumented code

```
num=1;
enter_spec();
p=m[a[*x]]+&y;
...
if(pstate()==REEXEC)
{
 info_collect();
}
exit_spec(flag);
num++;
```

Dynamic execution



Normal

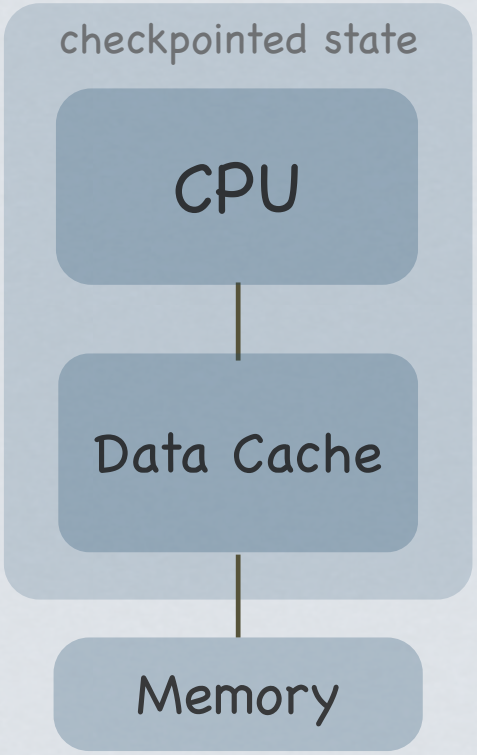
Speculative

Re-execute

i-acoma group

Implementation

- Save/restore processor state:
  - Register checkpointing and restoration
  - Data cache that buffers speculative data (commit or invalidate)
  - Instructions enable/disable speculation on-the-fly
- Limits: cache size, I/O



i-acoma group

Other uses of program rollback support

i-acoma group

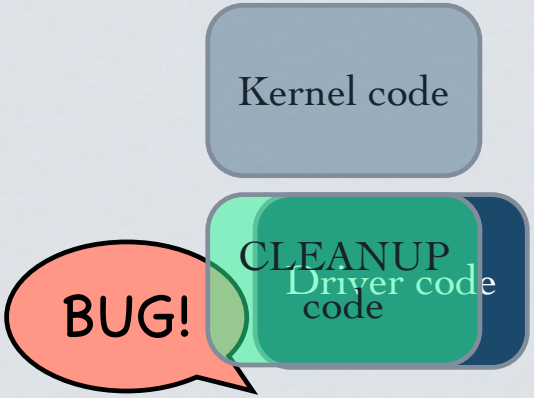
Code versioning

- Binary keeps two versions:
  - conservative - safer
  - aggressively optimized - potentially buggy
- Execute aggressive code speculatively
- If test fails, fall back on conservative version

i-acoma group

Sandboxing OS drivers

- Buggy drivers - main cause of OS crashes
- Kernel survival in the presence of faulty drivers
- Execute driver code speculatively
- If crash, re-initialize driver



Failure-oblivious computing

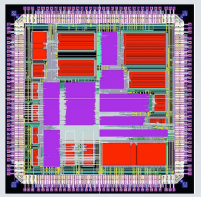
- Enables applications to execute beyond some errors [Rinard04]
- Invalid memory accesses are caught
  - write: ignore, continue execution
  - read: manufacture value, continue
- After invalid access - speculative execution for a certain duration

i-acoma group

Evaluation

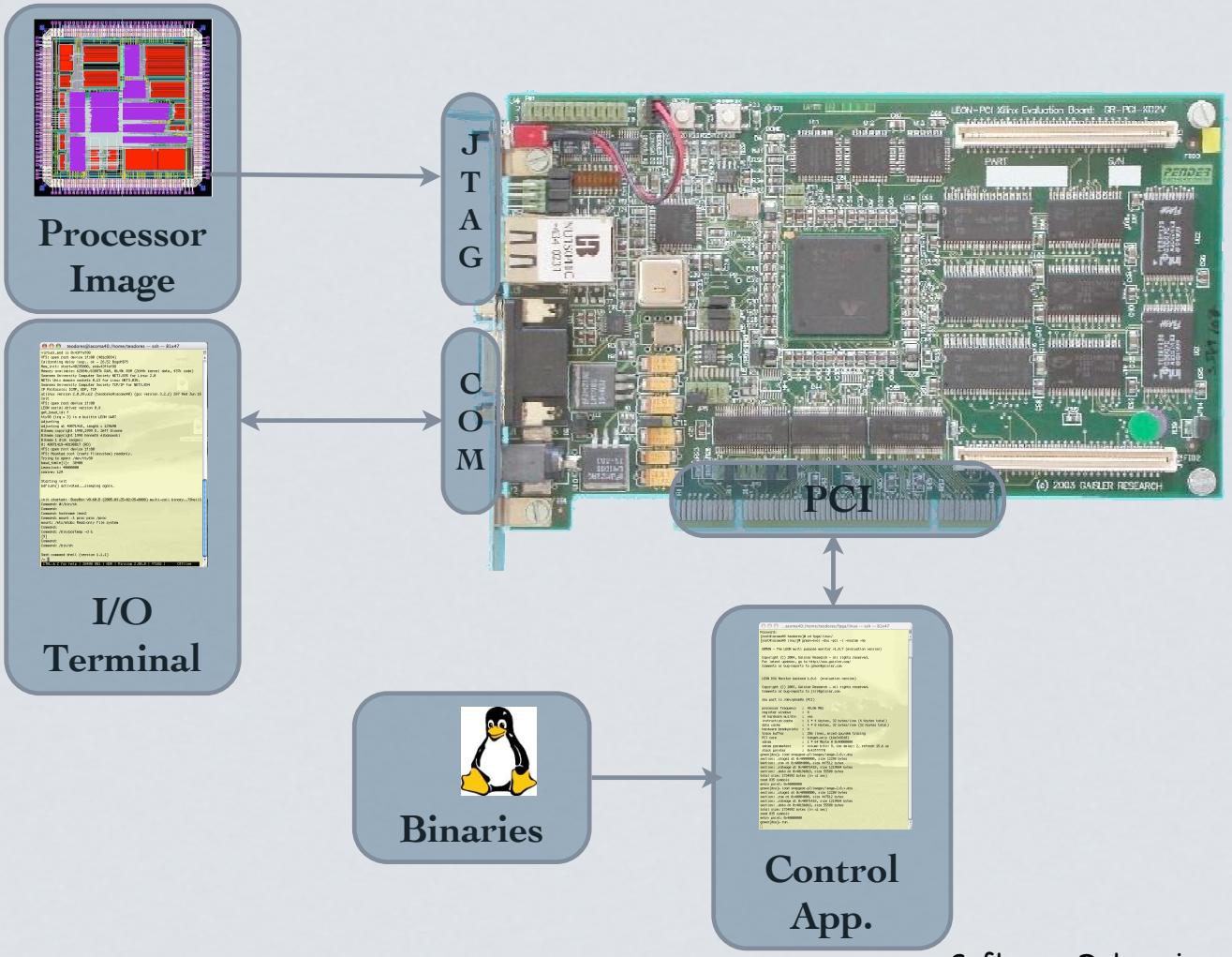
i-acoma group

Hardware prototype



- LEON2 - SPARC V8 compliant processor
- In-order, single issue, 5-stage pipeline
- Windowed register file
- L1 instruction and data caches
- Synthesizable, open source VHDL code
- Fully functional, runs Linux embedded

System Deployment



i-acoma group

Evaluation

- Applications with known bugs
- Manually instrument the code
- Detection window contains:
  - bug location
  - bug manifestation
- Determine if we can roll back the buggy code section

i-acoma group

Buggy applications

Application	Bug Description	Successful rollback	Dynamic Instructions
ncompress-4.2.4	Input file name longer than 1024 bytes corrupts stack	Yes	10653
polymorph-0.4.0	Input file name longer than 2048 bytes corrupts stack	No	103838
tar-1.13.25	Unexpected loop bounds causes heap object overflow	Yes	193
man-1.5h1	Wrong bounds checking causes static object corruption	Yes	54217
gzip-1.2.4	Input file name longer than 1024 bytes overflows a global variable	Yes	17535

i-acoma group

Conclusions

- Simple hardware can provide powerful debugging support
- We built an FPGA-based prototype of a processor with program undo support
- We describe a few possible applications to software debugging

i-acoma group

Thank you!

Discussions and demo