FAULT INJECTION-BASED BENCHMARKING OF SOFTWARE COMPONENTS

Johan Karlsson, Fatemeh Ayat, Behrooz Sangchoolie

Department of Computer Science and Engineering
Chalmers University of Technology
MOTIVATION

Trends

• Autonomous driving and advance active systems
  ➢ Move from *fail safe* systems to *fail operational* systems poses major challenge for the automotive industry
  ➢ Need for *low-cost* solutions to fault tolerance
  ➢ Increasing focus on functional safety (ISO 26262)

• Component-based and agile software development
  ➢ Repositories of software components for safety-critical systems

• Technology scaling is making transistors less and less reliable
  ➢ Increasing interest in Software Implemented Hardware Fault Tolerance (SIHFT) techniques

⇒ Need for techniques for assessing/measuring the sensitivity of software components to hardware faults
TECHNOLOGY SCALING AND RELIABILITY

Kelin J. Kuhn, Intel Fellow, Director of Advanced Device Technology Intel Corporation, 2009

“As near as I can tell: THE key challenge is that the transistors get smaller”
TRENDS IN THE BATHTUB CURVE

- **Infant mortality**: Increasing manufacturing defects
- **Constant failure rate**: Increasing rate of transient, intermittent and permanent faults
- **Wearout**: Acceleration of aging phenomena

Source: Vikas Chandra, ARM R&D, Dependable Design in Nanoscale CMOS Technologies: Challenges and Solutions Keynote address, WDSN, Estoril, Portugal, June 29, 2009
LAYERED FAULT TOLERANCE

System failure modes

Node failure modes

Errors escaping detection by HW

Hardware-layer mechanisms

- HW Design Fault
- SW Design Fault
- Physical Fault

Software-layer mechanisms

- Detected Error
- Undetected Error

System-layer mechanisms

- Interference failure
- Value failure
- Timing failure
- Fail signal
- Fail silent

Error removed

3rd line of defence

2nd line of defence

1st line of defence

Cost balancing

- Catastrophic failure
- Benign failure
- Safe Shutdown

BeSafe Benchmarking of Functional Safety

ICES Workshop (November 14, 2012)
EVALUATION DRIVEN DESIGN OF SIHFT MECHANISMS (FOR SW COMPONENTS)

- Requirements
- Known principles
- Team experience

Iterative design process

Design of fault tolerance mechanisms
Evaluation (e.g. fault injection)

SW-components + Evaluation and Test Results

SW-component repository
CHALLENGES IN FAULT INJECTION-BASED ASSESSMENT OF SW COMPONENTS

• The aim of fault injection is to determine the failure mode distribution of a software component.

• The failure mode distribution depends on several parameters:
  • Fault model
  • Target processor
  • Compiler
  • Inputs processed by the program
  • Implementation
The experiments are conducted automatically with the Generic Object-Oriented Fault Injection tool (Goofi-2).

Key features:
- Nexus-based fault injection
- Golden run
- Pre-injection analysis
- SQL Database
**FAULT MODEL**

Single / Multiple bit flips in ISA (Instruction Set Architecture) registers and main memory.

- Bit flipping is used to emulate the effect of single event upset (SEU), e.g. errors caused by ionizing particles.

A fault injection *experiment* consists of injecting one fault and observing its impact on a workload.
# Target Fault Injection Locations

## Target Memory Sections
- Stack
- Data
- Sdata
- Bss
- Sbss

## Target Registers
- General Purpose Registers (GPR)
- Floating Point Registers (FPR)
- Program Counter Register (PCR)
- Link Register (LR)
- Condition Register (CR)
- Integer Exception Register (XER)
HARDWARE FAULT SENSITIVITY MEASURES FOR SOFTWARE COMPONENTS

- **No Effect**
  - No Impact
  - Corrected by Software

- **Detected**
  - Detected by Hardware
  - Detected by Software
  - Timeout

- **Failure**
  - Value Failure
  - Timing Failure

Error coverage

Failure Mode Distribution

BeSafe
Benchmarking of Functional Safety
TARGET APPLICATIONS (From MiBench Suite)

- **CRC-32** (32-bit cyclic redundancy check)
- **SHA** (Secure hash algorithm)
- **BinInt** (Binary to integer convertor)
- **Quicksort** (Quick sort algorithm)
- **BitCnt** (Bit count program)
- **Isqrt** (Square root algorithm)
- **Cubic** (Solve a cubic polynomial)
AVERAGE FAILURE MODE DISTRIBUTION OVER ALL SWC

**Single Bit Flip**
- No Impact: 39%
- Value Failure: 30%
- Detected by Hardware: 29%
- Time out: 2%

**Multiple Bit Flips**
- No Impact: 24%
- Value Failure: 47%
- Detected by Hardware: 27%
- Time out: 2%
FAILURE MODE DISTRIBUTIONS
FOR SELECTED SWC

Single Bit Flip

- CRC: 44%
- SHA: 40%
- SHA: 16%

Multiple Bit Flips

- CRC: 53%
- SHA: 50%
- SHA: 10%

- No Impact
- Value Failure
- Detected by Hardware
- Time out

ICES Workshop (November 14th 2012)
AVERAGE FAILURE MODE DISTRIBUTIONS OF DIFFERENT IMPLEMENTATIONS (Single Bit Flip)

- **BitCnt1**:
  - No Impact: 6%
  - Value Failure: 32%
  - Detected by Hardware: 39%
  - Time out: 23%

- **BitCnt2**:
  - No Impact: 35%
  - Value Failure: 33%
  - Detected by Hardware: 32%
  - Time out: 0%

- **BitCnt3**:
  - No Impact: 0%
  - Value Failure: 38%
  - Detected by Hardware: 35%
  - Time out: 27%

- **BitCnt4**:
  - No Impact: 46%
  - Value Failure: 31%
  - Detected by Hardware: 22%
  - Time out: 1%

ICES Workshop (November 14th 2012)
CONCLUSIONS

• The multiple bit flip model in average results in less value failures in comparison to the single bit flip model.

• The failure mode distribution varies significantly for different software components.

• The implementation had a significant impact on the failure mode distributions.
FUTURE WORK

• Measure the failure mode distribution for software components equipped with software based fault tolerance techniques

• Extend the experiments with other software components

• Develop a repository of software components for safety critical applications

• Investigate the impact of compiler optimizations

• Develop a method to predict the fault sensitivity of complex software functions based on failure mode distribution of software components
REFERENCES


Thank you all for your attention!

Acknowledgements:

This work has partly been supported by VINNOVA-FFI BeSafe project
AVERAGE FAILURE MODE DISTRIBUTION OF EACH SWC

- Time out
- Detected by Hardware
- Value Failure
- No Impact

ICES Workshop (November 14th 2012)
HARDENING SOFTWARE COMPONENTS AGAINST HARDWARE ERRORS

Goal

• To minimize the risk that hardware errors generate undetected value failures, a.k.a. silent data corruptions.

How?

• A possible technique is the Triple Time-redundant Execution with Forward Recovery.
FAILURE MODE DISTRIBUTIONS
(WITH TRIPLE-TIME REDUNDANT EXECUTION)

**CRC-32**
- Value Failure: 0.70%
- No Impact: 1.60%
- Detected by Hardware: 20.80%
- Time out: 33.40%
- Corrected by Software: 43.20%

**SHA**
- Value Failure: 1.80%
- No Impact: 0.80%
- Detected by Hardware: 14.90%
- Time out: 0.10%
- Corrected by Software: 43.40%

**BinInt**
- Value Failure: 2.90%
- No Impact: 1.40%
- Detected by Hardware: 34.70%
- Time out: 0.90%
- Corrected by Software: 40.60%

**Quicksort**
- Value Failure: 2.80%
- No Impact: 5.40%
- Detected by Hardware: 28.70%
- Time out: 0.80%
- Corrected by Software: 20.40%

ICES Workshop (November 14th 2012)
ERROR COVERAGE VARIATION WITH RESPECT TO DIFFERENT INPUTS
Value Failure Distribution for Quick Sort (Single)

Value Failure Distribution for Quick Sort (Multiple)
Value Failure Distribution Bit Count (Single)

Value Failure Distribution Bit Count (Multiple)