# Software Mechanisms for Tolerating Soft Errors in an Automotive Brake-Controller

Daniel Skarin    Johan Karlsson

Department of Computer Science and Engineering
Chalmers University of Technology
Göteborg, Sweden

June 29, 2009

# Introduction

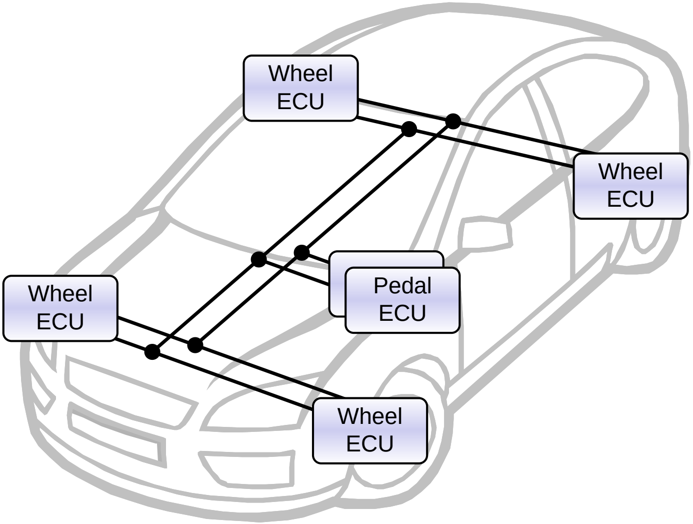
- Soft errors are becoming an increasingly important source of computer failures, also in embedded systems.
- The dominant cause of soft errors are terrestrial cosmic rays.
- Circuit- and architectural level mechanisms in microprocessors may not provide perfect error coverage.
  $\Rightarrow$ Soft errors can reach the architected state.

- Goal: Investigate the possibility of building a brake controller program, which is fail-bounded with respect to soft errors.

# Fail-bounded control systems

- Control systems can produce incorrect outputs and still provide acceptable performance.
- A fail-bounded system is allowed to produce incorrect outputs, which have a benign effect on the controlled object.
- Error detection mechanisms must enforce an upper bound on the difference between an incorrect output and the corresponding fault-free output.
- The concept of fail-bounded systems was introduced by Silva et al. in 1998.

# Example brake-by-wire system



# Research questions

General question

- Can we make a non-redundant control ECU fail-bounded with respect to soft errors?

Question addressed by this work

- Can we make a non-redundant control ECU fail-bounded with respect to single bit-flip errors in ISA registers and the data segment of the main memory?

# Contributions

Extensive evaluation of two simple software mechanisms aimed at achieving a fail-bounded brake controller.

- The error coverage of the mechanisms have been determined for single bit-flips in ISA registers and the data segment of the main memory.
- Exhaustive evaluation for three control loops: All possible single bit-flips injected.
- All ISA registers including the program counter tested.

# Limitations of the single bit-flip fault model

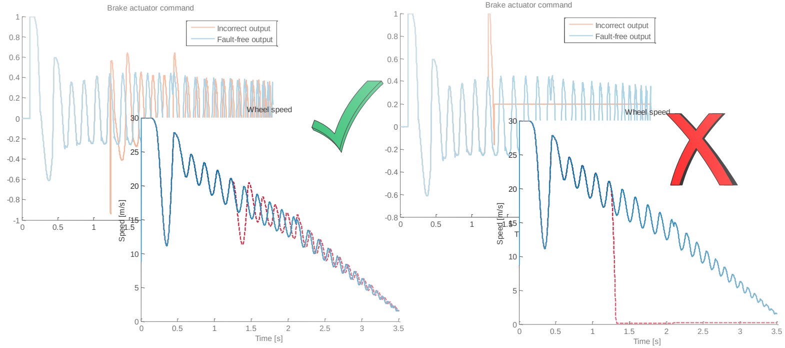
- We emulate soft errors in the architected state as single bit-flip errors in registers and memory.
- Single bit-flips are injected via the debug port of the target microcontroller.

## Uncertainties

- Soft errors may not manifest themselves as single bit-flips.
- Out-of-specification behaviors of the processor are not considered.

# Prototype brake controller

- Actuator commands are produced by a PI-controller
- We distinguish between *benign failures* and *critical failures*.



# Low-cost error detection and recovery

## Software mechanisms

- Error detection:
  - Run-time check for invalid transitions of the controller's integral state.
  - Stack pointer protected by duplication and comparison check.
- Error recovery:
  - Rollback to previous controller state
  - Soft reset

## Hardware exceptions for error detection

- Machine check exception, Alignment exception, Floating point assist exception, ...

# Experimental evaluation

We evaluated two versions of the brake controller:

- Basic version – Hardware exceptions for error detection.
- Robust version – Hardware exceptions and software implemented error detection and recovery.

Extensive fault injection experiments conducted for each version.

- For three control loops, we injected all possible single bit-flips in “live” ISA registers and the data segment of the memory.
- About 30 000 errors were injected for each program version and control loop iteration.

# Important observations

- Our software mechanisms combined with hardware exceptions reduced the proportion of critical failures significantly.
  - Only 0.04% of the injected errors resulted in critical failures, compared to 1.2% for the basic version.
- A dominant cause of critical failures was control-flow errors.
- In total, about 56% of the injected errors caused incorrect outputs in the robust version.
- These errors had no significant impact on the brake performance.

# Conclusions

- Our results show that simple mechanisms for error detection and recovery can effectively enforce fail-bounded semantics for the brake controller with respect to single bit errors.
- Open issues
  - How valid is the single bit-flip assumption?
  - How do we model multiple bit-flips?
  - What is the impact of out-of-specification behaviors of the microprocessor?

# Fault injection data available on-line

http://www.amber-project.eu