

Impact of Soft Errors in a Brake-by-Wire System

Daniel Skarin and Johan Karlsson

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

Martin Sanfridson

*Volvo Technology
Göteborg, Sweden*

Outline

- Automotive control systems
- Evaluation of a prototype brake-by-wire system
 - ▶ Objectives
 - ▶ Experimental setup
 - ▶ Results
- Observations and future work



Automotive control systems

Electronic control systems today provide functions that assist driving

- Anti-lock braking system (ABS)
- Electronic stability program (ESP)
- Adaptive cruise controller (ACC)

Safe shutdown is a viable approach to handling failures in these systems.

Future systems will provide mandatory functionality

- Brake-by-wire
- Steer-by-wire

These systems can not be shut down!



Automotive control systems

Advanced active safety systems can take control of the vehicle when a collision is imminent

- Collision avoidance
- Collision mitigation

Important to consider the impact of soft errors in active safety systems

- Large number of vehicles
- Probability for erroneous activations must be kept very low
 - ▶ False activations can have severe consequences



Brake-by-wire evaluation

Objectives

Experimentally evaluate the impact of soft errors in a brake-by-wire system

- Assess the risk that soft errors cause the system to produce dangerous outputs
- Study the impact of program-level error masking
- Develop techniques for dealing with soft errors in active safety systems

So far, we have conducted experiments with a system without software-implemented error detection.

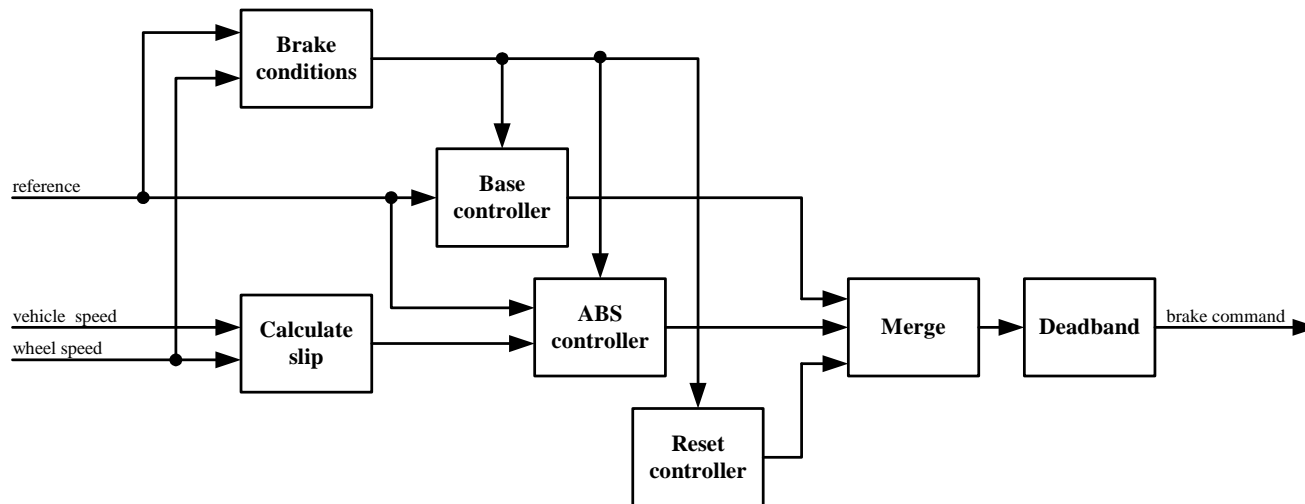


Brake-by-wire evaluation

Prototype brake controller

Prototype brake-by-wire controller developed by Volvo Technology

- Developed using Matlab and Simulink from MathWorks, Inc
- C code generated using TargetLink from dSPACE GmbH



Brake-by-wire evaluation

Fault injection

The brake-by-wire controller was evaluated using the GOOFI tool

- Fault/error injection tool developed at Chalmers
- Inserts soft errors (single and multiple bit-flips) in CPU registers and memory
- Pre-injection analysis to avoid injections into dead registers
 - ▶ A register is considered to be dead at a time, t , if it at that time holds data which will never be read again
 - ▶ Soft errors affecting dead registers can not propagate



Brake-by-wire evaluation

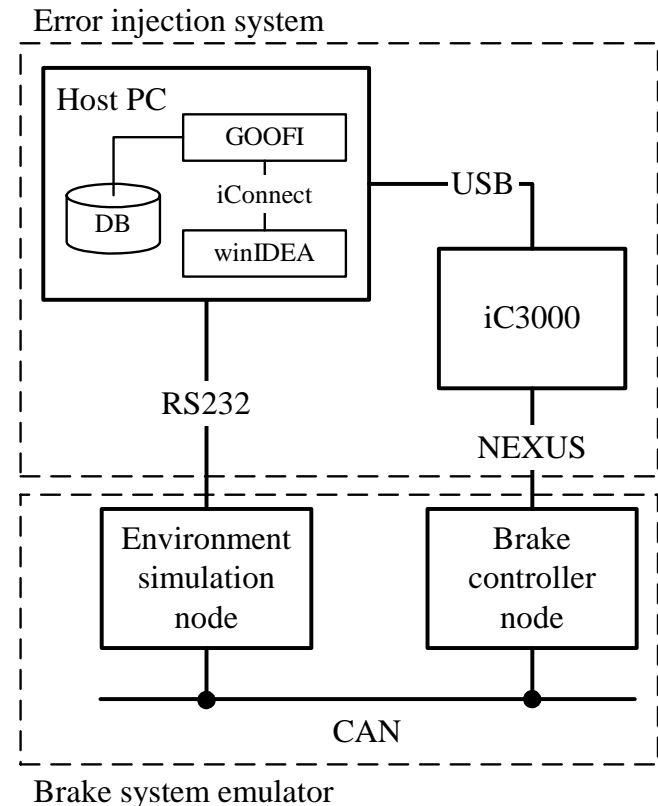
Experimental setup

Brake system emulator

- Two single board computers based on the MPC565 from Freescale
 - ▶ Brake controller
 - ▶ Environment simulation model

Error injection system

- GOOFI
 - ▶ OFFSET - Pre-injection analysis tool
- Debug environment from iSYSTEM
- Single bit-flips injected into the brake controller node
 - ▶ CPU registers
 - ▶ Main memory



Brake-by-wire evaluation

Results

The presentation of the results will focus on

- Classification of error impact
- Critical failures
- Program-level error masking



Brake-by-wire evaluation

Classification of error impact

Outcomes of the experiments were classified in three groups

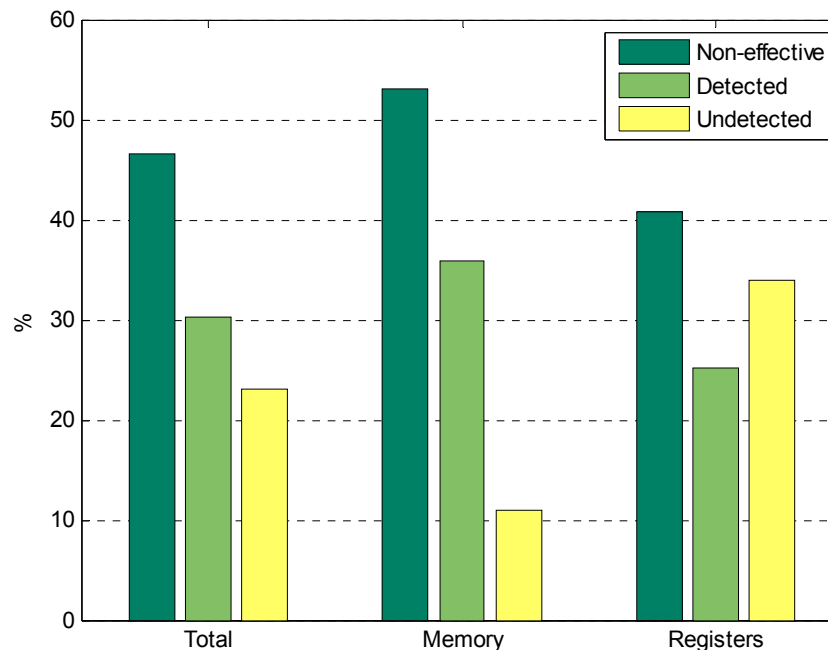
- Non-effective – Only correct outputs were observed
- Detected – The error was detected by a hardware exception
- Undetected – An erroneous brake command was produced



Brake-by-wire evaluation

Classification of error impact

- About 30% (1754 of 5802) of the errors caused erroneous outputs
 - ▶ Memory errors were more likely than register errors
- A majority of the errors were non-effective, 47%, or detected by hardware exceptions, 23%.

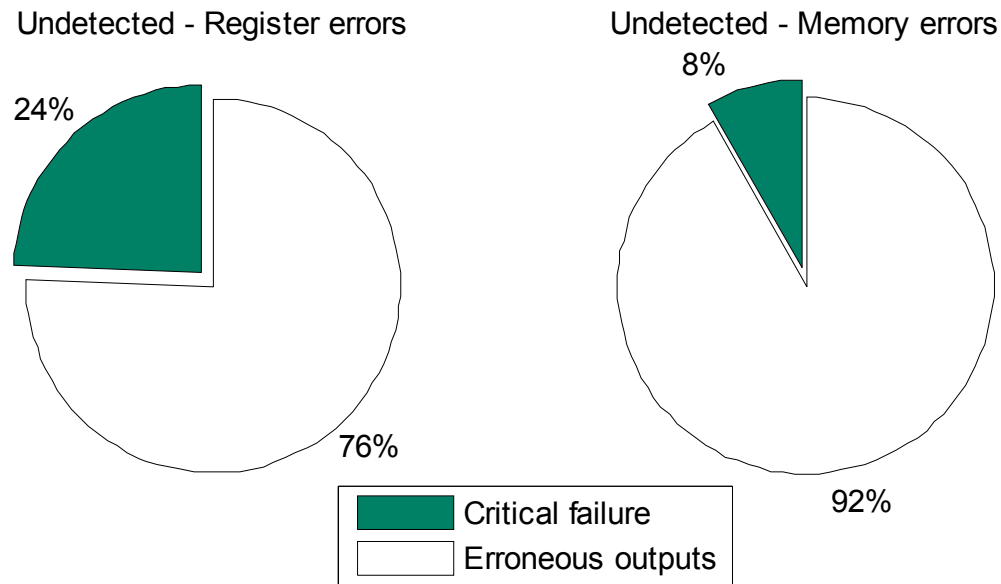


Brake-by-wire evaluation

Critical failures

About 15% (268 of 1754) of the errors that propagated to the output resulted in a critical failure

- Wheel being locked (41% of the critical failures)
- Loss of braking (59% of the critical failures)

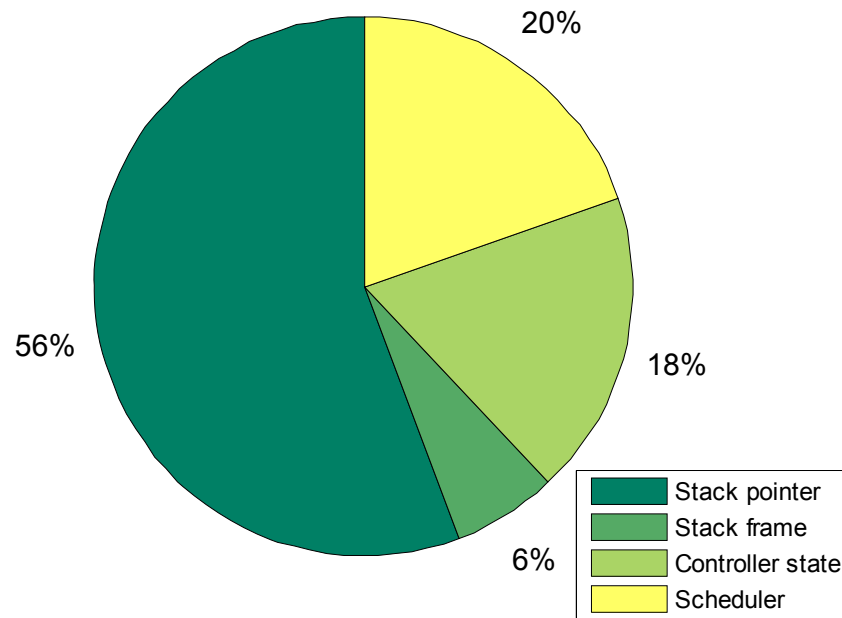


Brake-by-wire evaluation

Critical failures

A majority of the critical failures were caused by

- Errors injected into the stack pointer
- Errors affecting the scheduler
- Errors affecting the controller state



Brake-by-wire evaluation

Program-level error masking

About 47% of the injected errors were non-effective even though errors were injected into live data

- Measure of program-level error masking
- Memory errors were masked more often than register errors
 - ▶ 51% compared to 40%
- Analysis of non-effective error propagation
 - ▶ Many errors affected branches



Brake-by-wire evaluation

Observations and future work

- High degree of program-level error masking
 - ▶ 47% of the injected errors did not have an effect on the produced brake command
- 4.6% of the injected errors resulted in outputs causing a critical failure. A majority of the failures of the brake controller were caused by
 - ▶ Errors injected into the stack pointer
 - ▶ Errors affecting the controller state
- Future work
 - ▶ Evaluate application-level techniques for dealing with soft errors
 - ▶ Investigate advanced active safety systems



Thank you for your attention!

Comments / Questions?

