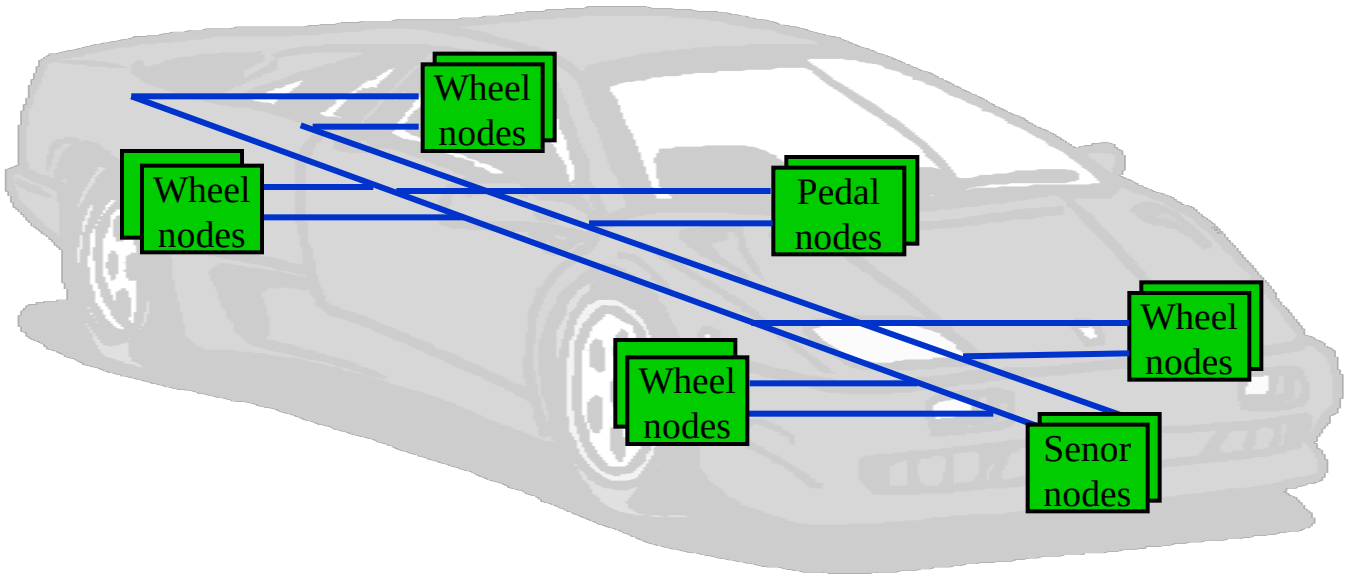
Impact of soft errors on a brake-by-wire controller

Daniel Skarin
Johan Karlsson

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

Brake-by-wire and Collision Mitigation System



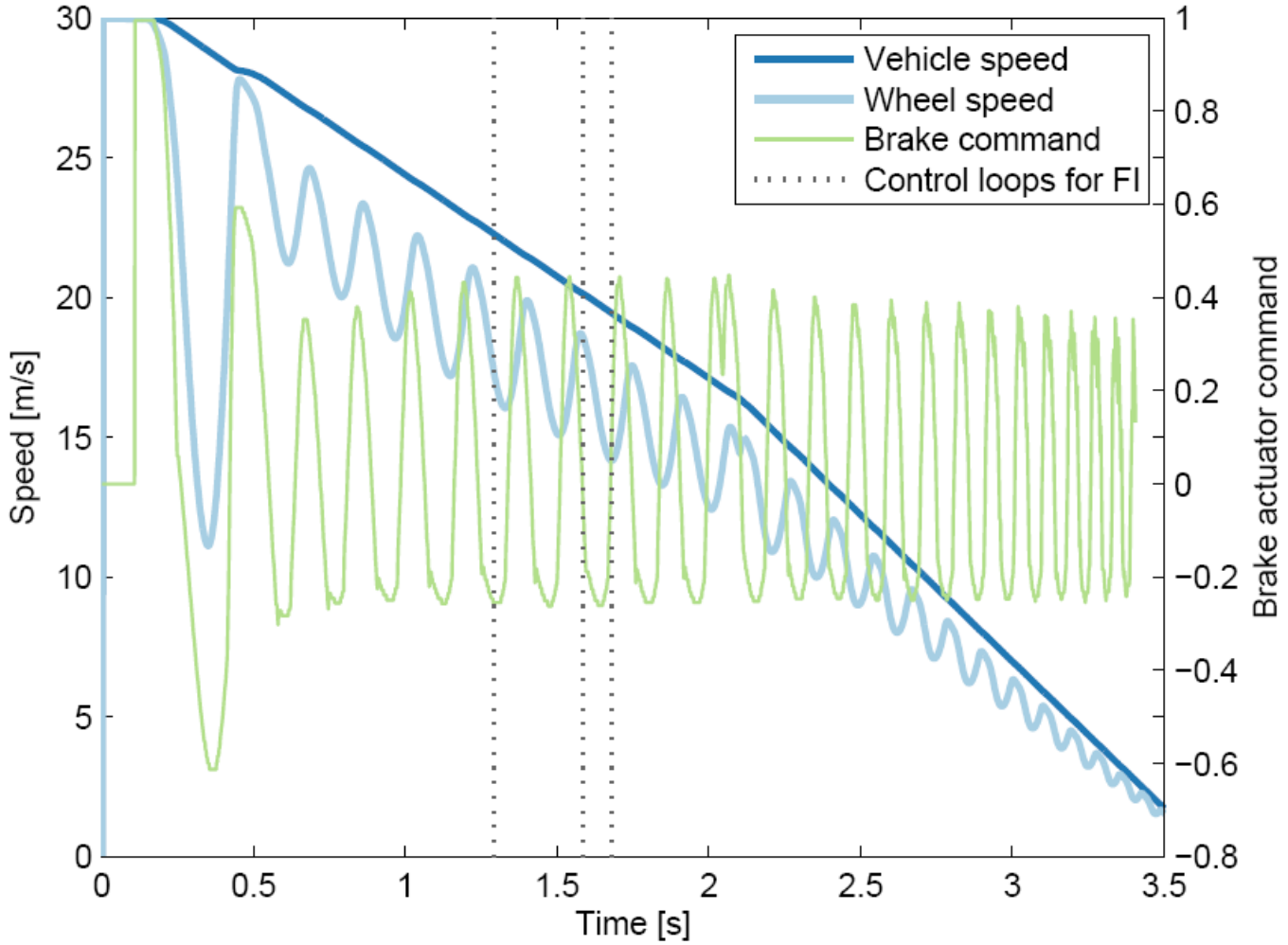
Assumptions

- Future microcontrollers will be manufactured in circuit technologies (e.g. 90 nm or 65 nm CMOS) that are sensitive to cosmic ray induced soft errors
- These microcontrollers will be equipped with error detection and error correction mechanisms that can detect, mask and recover from a majority of the soft errors
- However, these mechanisms are not likely to provide perfect error coverage
- Hence, some soft errors will propagate to the architected state (CPU registers and main memory)

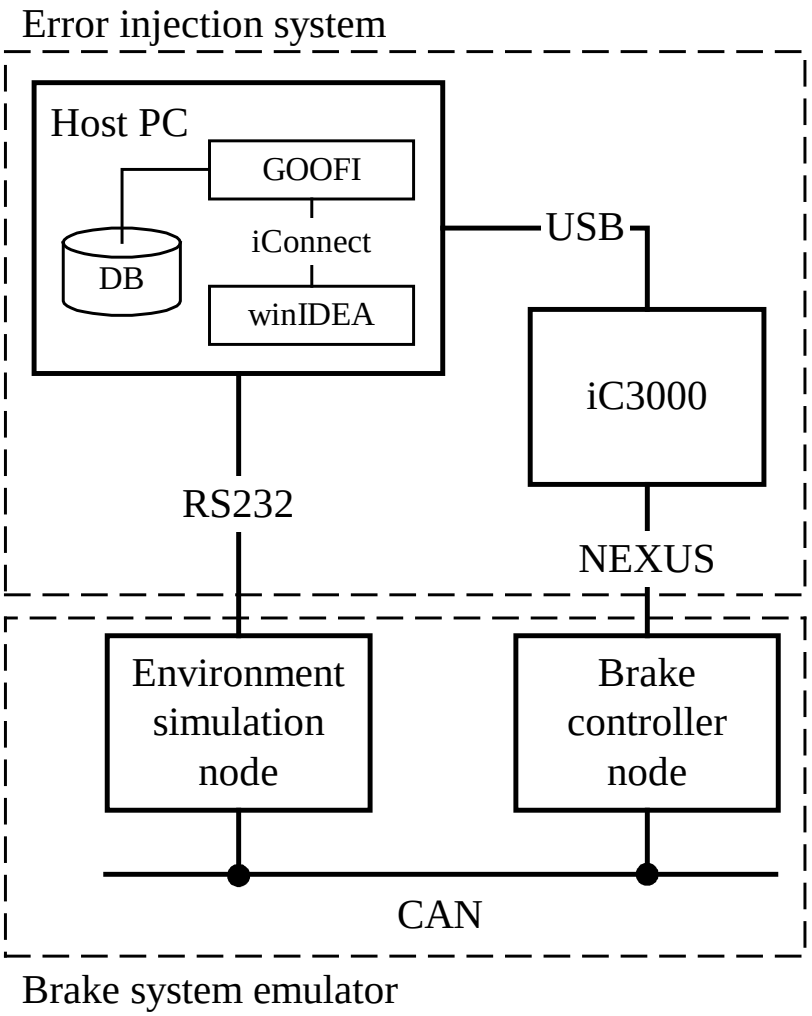
Research Questions

- What impact do soft errors that reach the architected state have on the brake-by-wire controller?
- How likely is it that these errors cause catastrophic failures?
- Can simple software implemented checks prevent catastrophic failures?
- If so, how effective are they?

Workload


Vehicle speed
Wheel speed
Brake command
Control loops for FI
Speed [m/s]
Brake actuator command
Time [s]
0
5
10
15
20
25
30
0.5
1
1.5
2
2.5
3
3.5
0.8
0.6
0.4
0.2
−0.2
−0.4
−0.6
−0.8

Experimental setup



Experimental Set-up

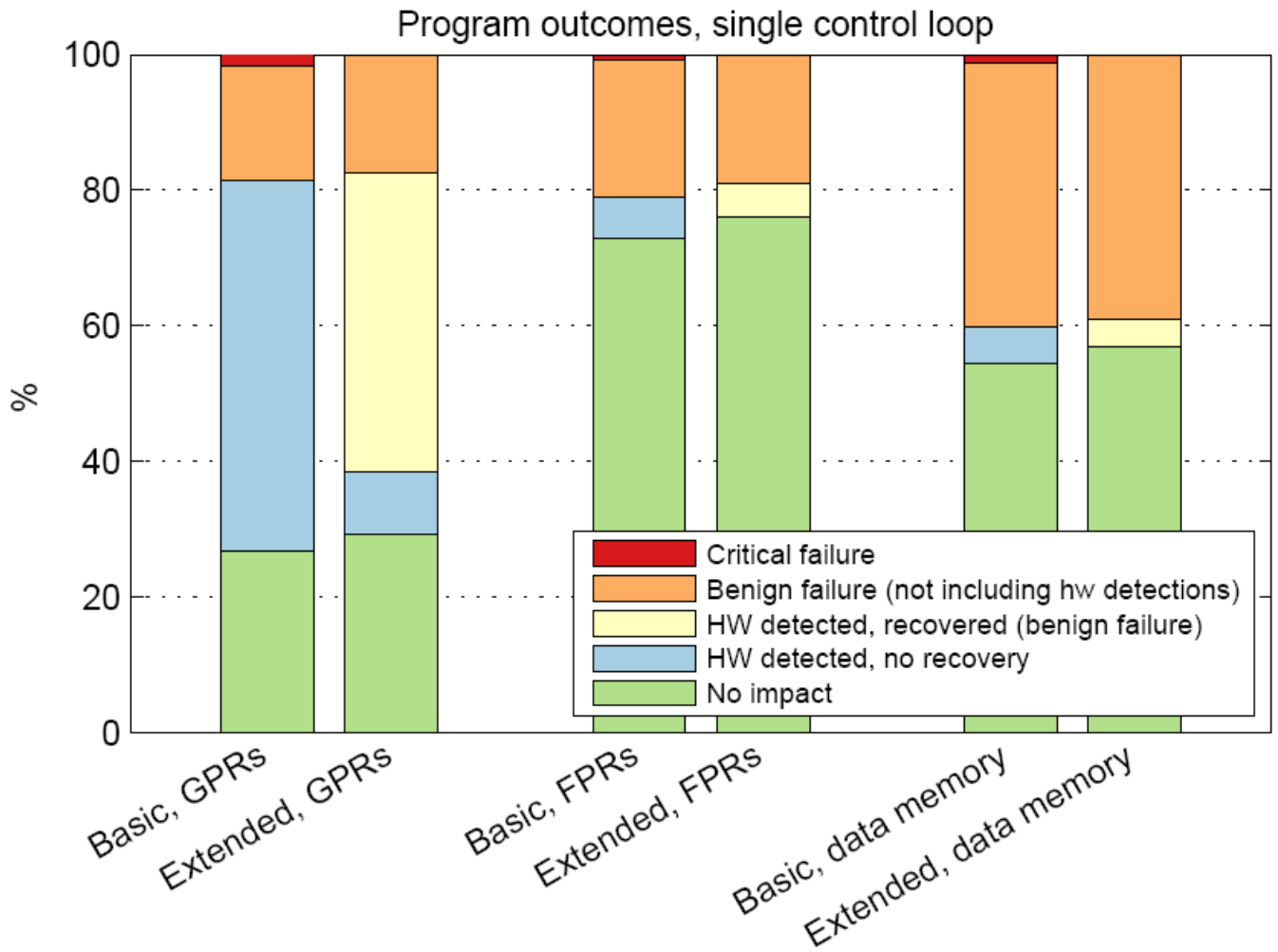
- Test-port based error injection on MPC565 (PowerPC architecture)
- Fault model: single bit-flips
- Full repeatability and controllability
- Pre-injection analysis ensures that bit-flips are only injected into *live* registers.
- Pre-injection analysis reduces the number of error injection targets one order of magnitude, from 200.000 to 20.000 for one control loop iteration.
- Exhaustive testing of several control loops

Brake controller versions

Brake controller versions:

- Basic version
  - Error detection: hardware exceptions in the microcontroller.
- Extended version
  - Error detection: hardware exceptions and software mechanisms (stack pointer check + integrator state check).
  - Error recovery: Reset or rollback to previous controller state.

Results



Control loop 665: Basic version executes 287 instructions, extended version 305 instructions.

Registers Tested / Not tested

Errors injected into

- General purpose registers (GPRs)
- Floating point registers (FPRs)
- Data memory

Errors not injected into

- Program counter
- Fixed-point exception register (XER) and floating-point status and control register (FPSCR) – These registers hold exception status
- Condition register (CR) – Holds ALU status information used by conditional branch instructions
- Link and count registers – Used by branch instructions
- Debug and control registers for the MPC565 microcontroller.

Conclusions

- Simple software mechanisms seems to be helpful in reducing risk of catastrophic failures in closed-loop control systems susceptible to soft error.
- Open Issues, limitations and uncertainties
  - Not all registers in the architected state tested
  - Need to exercise the system with other activation patterns
  - Validation of single-bit error model (requires simulation or radiation testing)
  - Abnormal behaviour of the microprocessor not tested (requires simulation or radiation testing)