Eliminating Single Points of Failure in Software-Based Redundancy

Peter Ulbrich, Martin Hoffmann, Rüdiger Kapitza, Daniel Lohmann, Reiner Schmid and Wolfgang Schröder-Preikschat

Diskussionskreis Fehlertoleranz
November 22, 2012

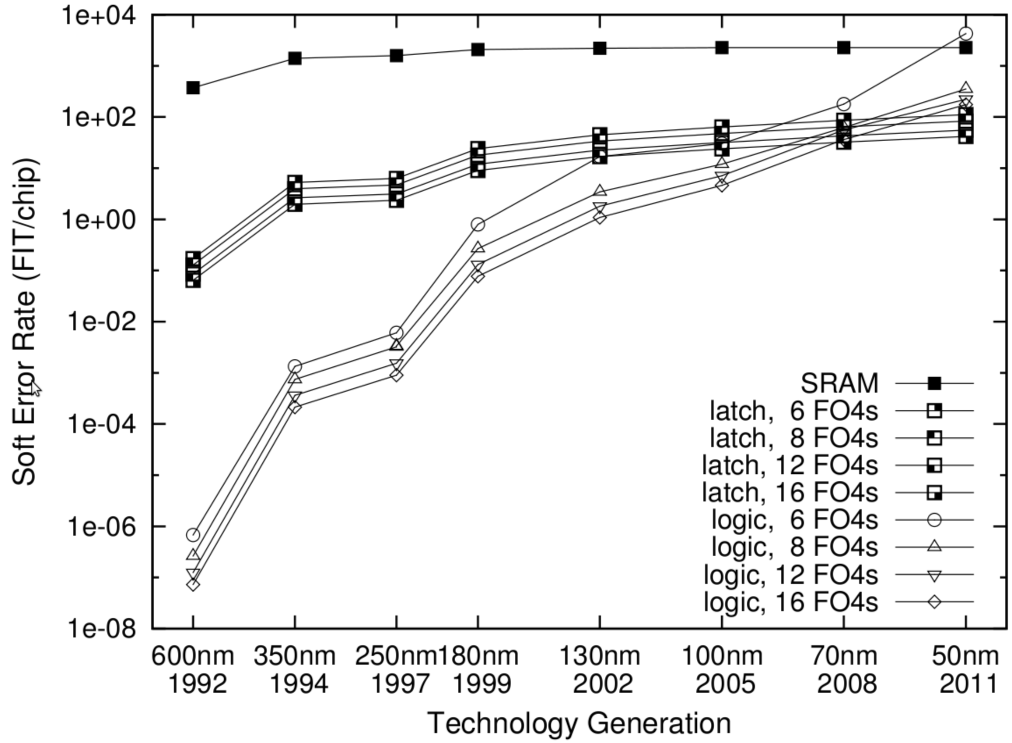
ANWENDUNGSZENTRUM esi
Embedded Systems Initiative

CHAIR IN DISTRIBUTED SYSTEMS
AND OPERATING SYSTEMS

SIEMENS

FAU
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Transient Hardware Faults – A Growing Problem

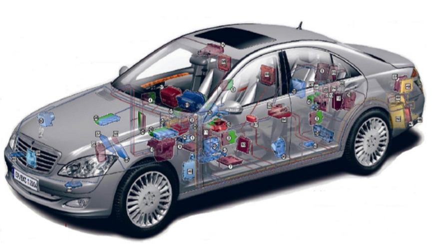


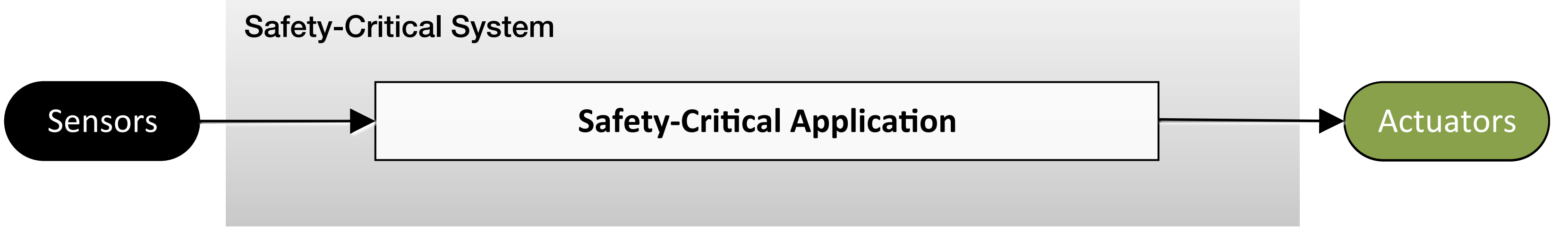
[3] (Shivakumar, 2002)

- **Transient hardware faults (Soft-Errors)**
  - Induced by e.g., radiation, glitches, insufficient signal integrity
  - Increasingly affecting microcontroller logic

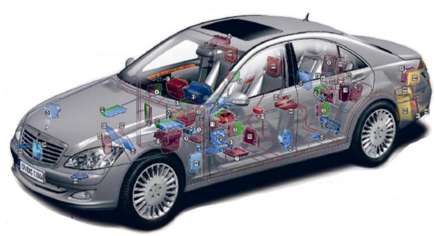
- Future hardware designs:
  Even more performance and parallelism
  → **On the price of being less and less reliable**

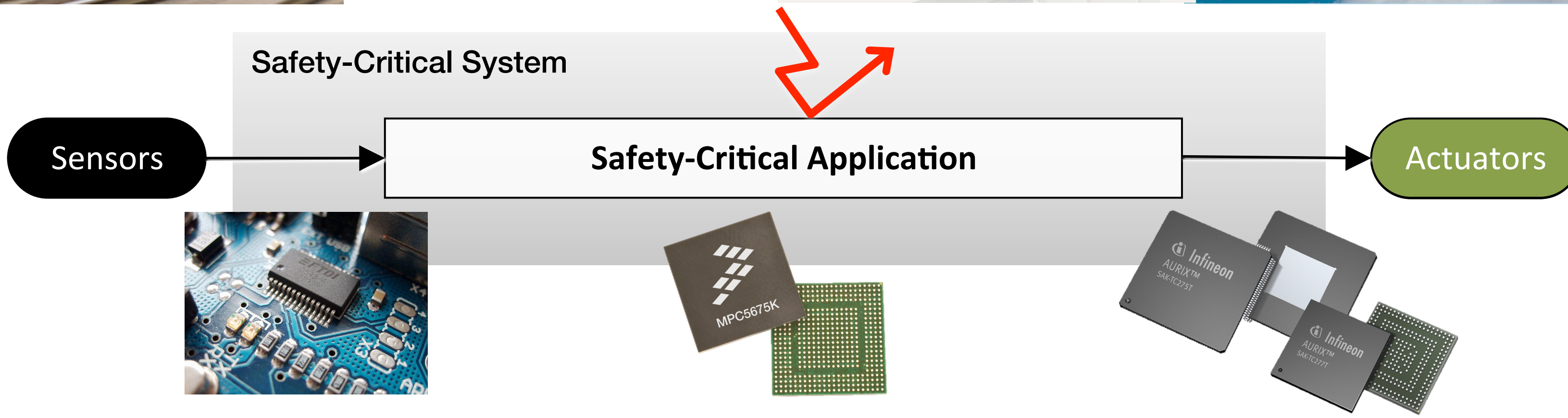
Countermeasures – Hardware Redundancy



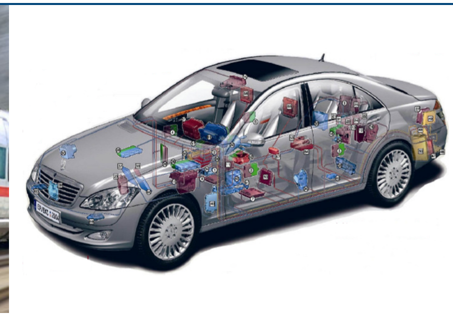
Countermeasures – Hardware Redundancy

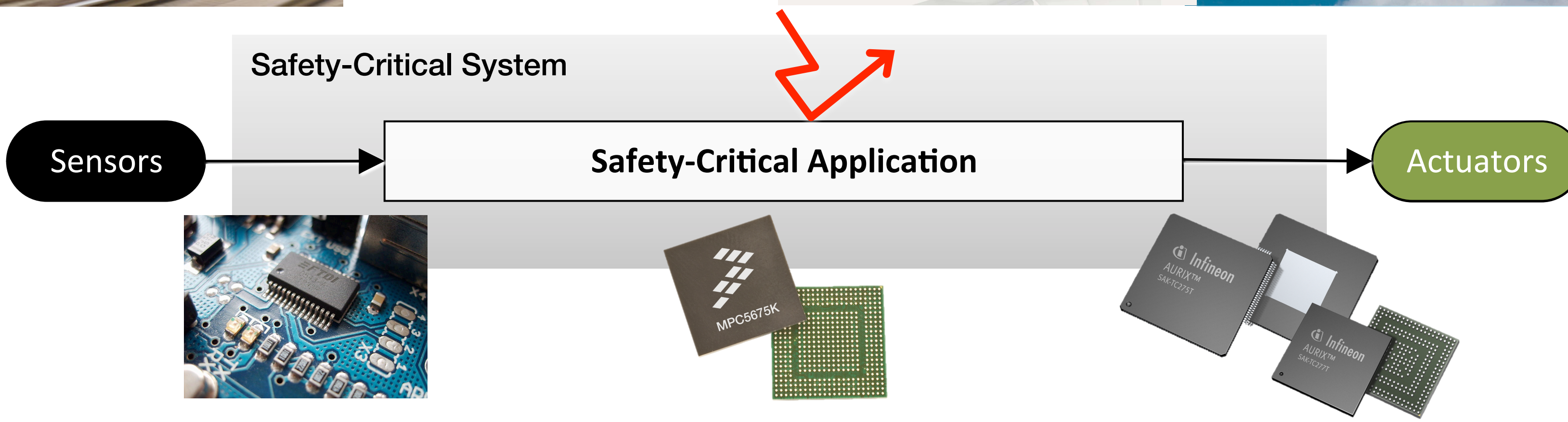




- Hardware-based redundancy
  - **Application-specific design** or **specialised hardware**
  - For example ECC, lock-step

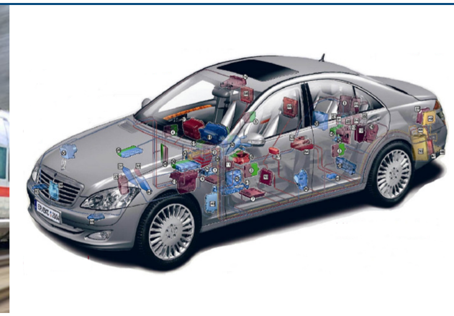
Countermeasures – Hardware Redundancy

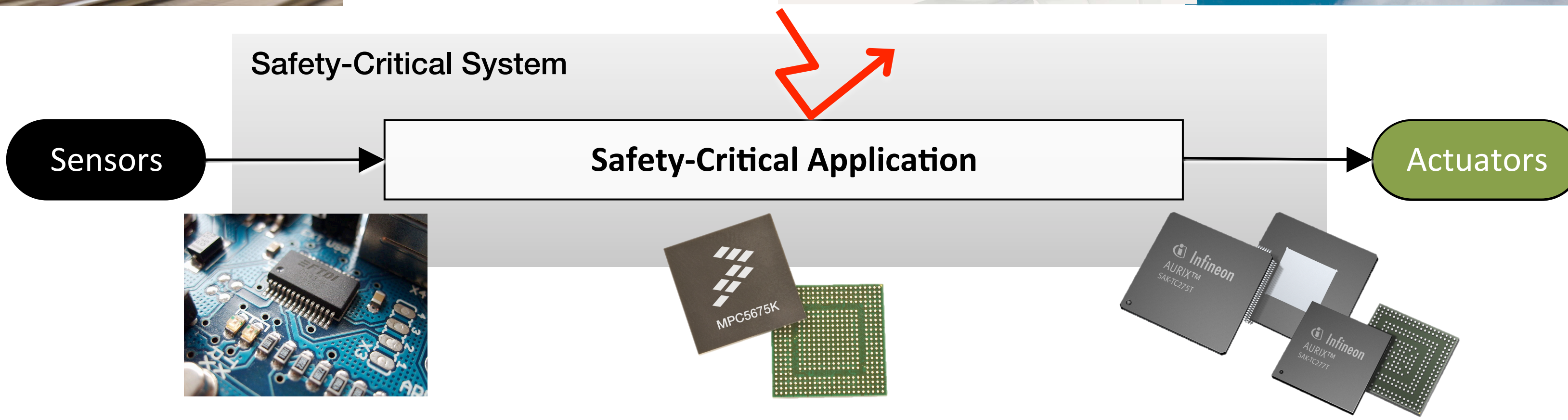





- Hardware-based redundancy
  - **Application-specific design** or **specialised hardware**
  - For example ECC, lock-step
  - ✓ **Pragmatic** and **safe** (tackles problem right at source)

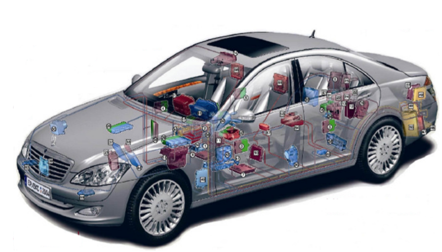
Countermeasures – Hardware Redundancy

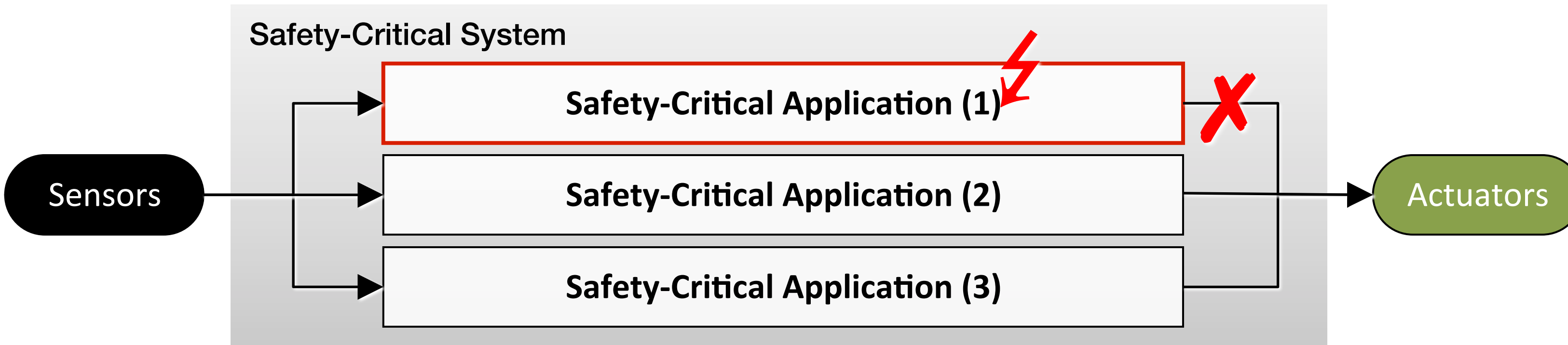




- Hardware-based redundancy
  - **Application-specific design** or **specialised hardware**
  - For example ECC, lock-step
  - ✓ **Pragmatic** and **safe** (tackles problem right at source)
  - ✗ **Hardware costs** (e.g., core, checker, …)
  - ✗ **Selectivity** and **adaptivity** (e.g., multi-application systems)
  - ✗ **Development costs** (diverse safety concepts and HW, (re-)certification)

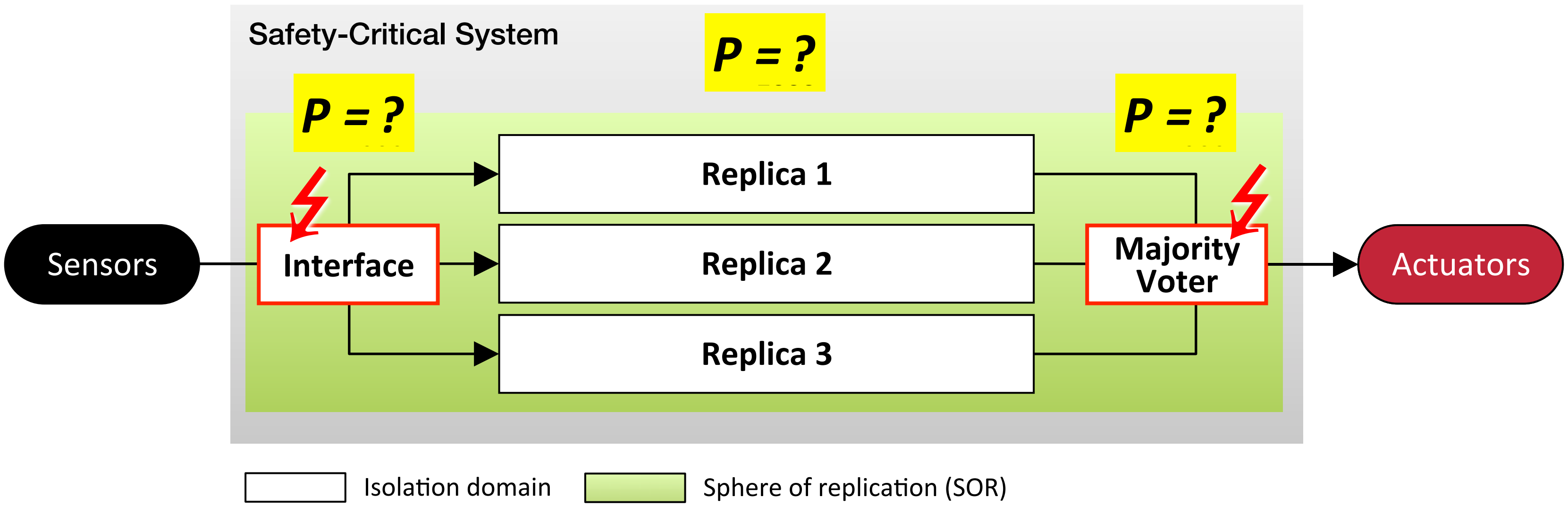
Countermeasures - Software



- **Software-based redundancy**
  - For example **Triple Modular Redundancy** (TMR) (e.g., recommended for ASIL D error handling)
  - ✓ **Selective** and **adaptive** (e.g., application or module level)
  - ✓ **Resource efficient** (protects only what is really necessary)
  - ? **Pragmatic** and **safe** (RTOS support, input-output safety)
  - ? **Development costs** ((re-)certification)

Software-Based Redundancy in Detail



- Softw
  - T
  - I
- Single
  - N
  - V
- Risk a
  - I
  - Random error distribution? (Nightingale, 2011)

Aims

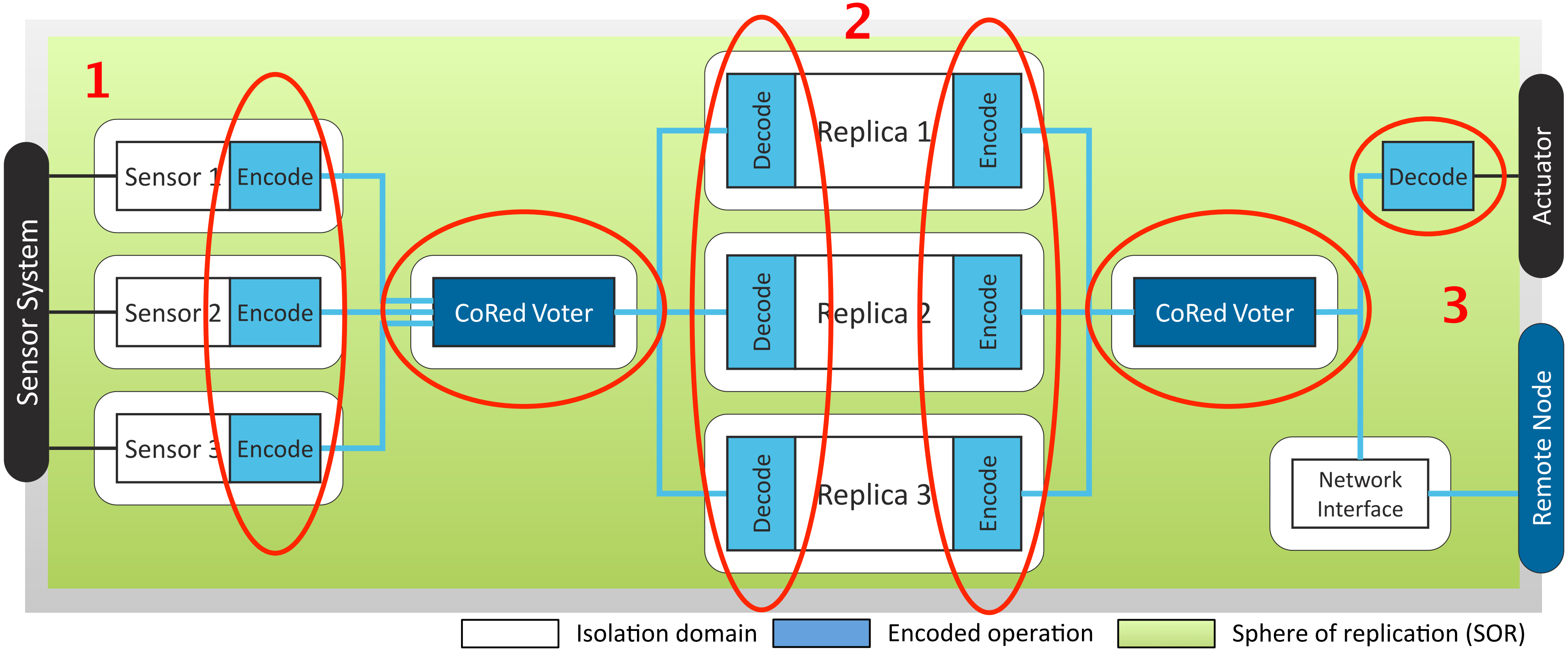
- **Eliminating single points of failure**
- **Input-to-output protection**
- **Safety* as a system software service**

* w.r.t. soft-errors

Agenda

- Introduction
- **The <u>Co</u>mbined <u>Red</u>undancy Approach**
  - Eliminating Vulnerabilities
  - High-Reliability Voters
- **Example: UAV Flight Control**
  - CoRed Implementation
  - Target System: I4*Copter*
- **Evaluation**
  - Experimental Setup
  - Results
- **Conclusion**

CoRed Overview – Holistic Protection Approach



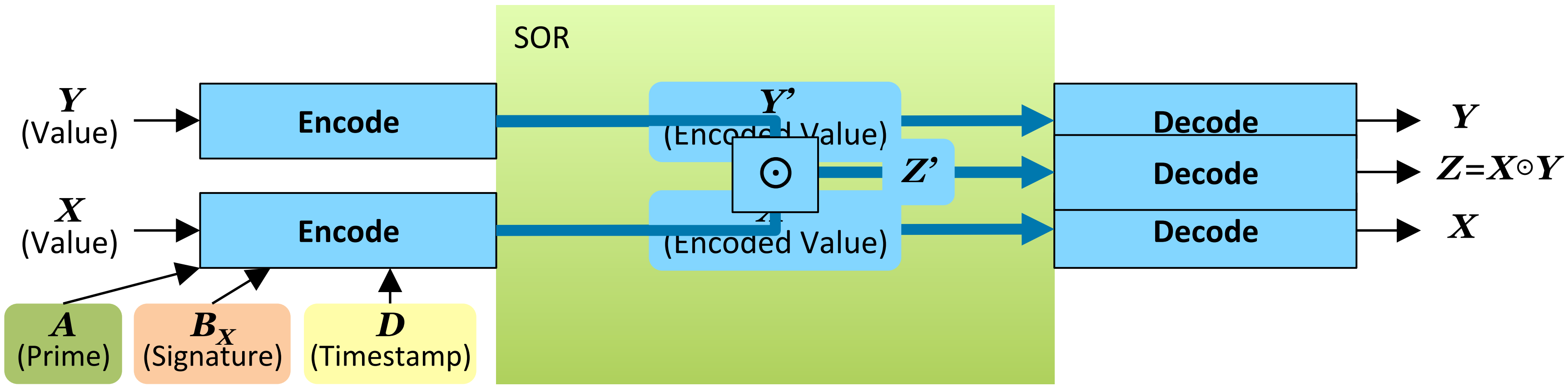
- **The Combined Redundancy Approach (CoRed)**

  TMR + { **Data-flow encoding** / **High-reliability voters** }

- **Holistic Protection Approach**
  - **Input to output protection**

    1 Reading inputs → **2** Processing → **3** Distributing outputs
  - **Composability** → On application and system level

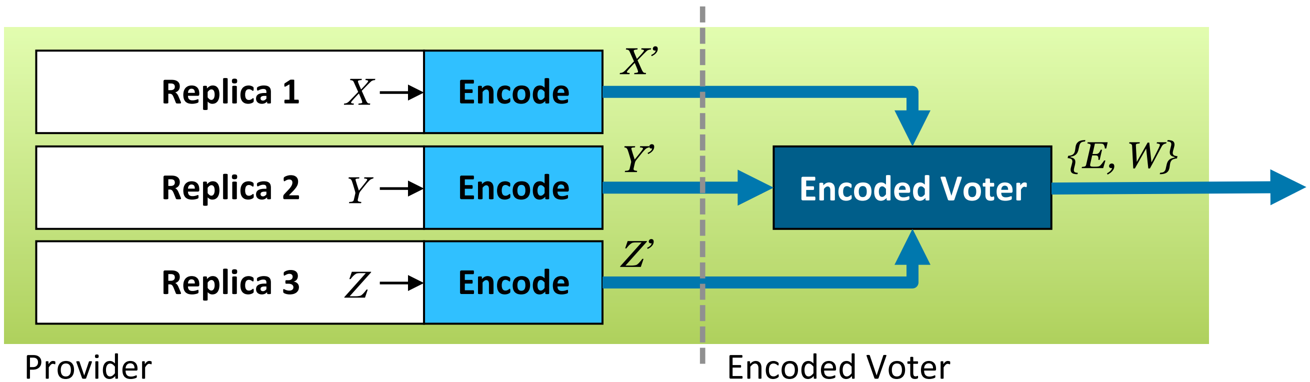
Eliminating Input and Output Vulnerabilities



- Inter-domain data-flow protection
  - **Checksum** vs. **Arithmetic code** (AN code)
  - AN Code → **Encoded data operations**
  - **Enabler for high-reliability voter**
- CoRed: **Extended AN code** (EAN code)
  - Based on VCP (Forin, 1989)
  - Data integrity: Prime
  - Address integrity: Per variable signature
  - Outdated data: Timestamp
  - Set of **arithmetic operands** (+, -, *, =, …)
  - Tailored for **efficient encoded data voting**

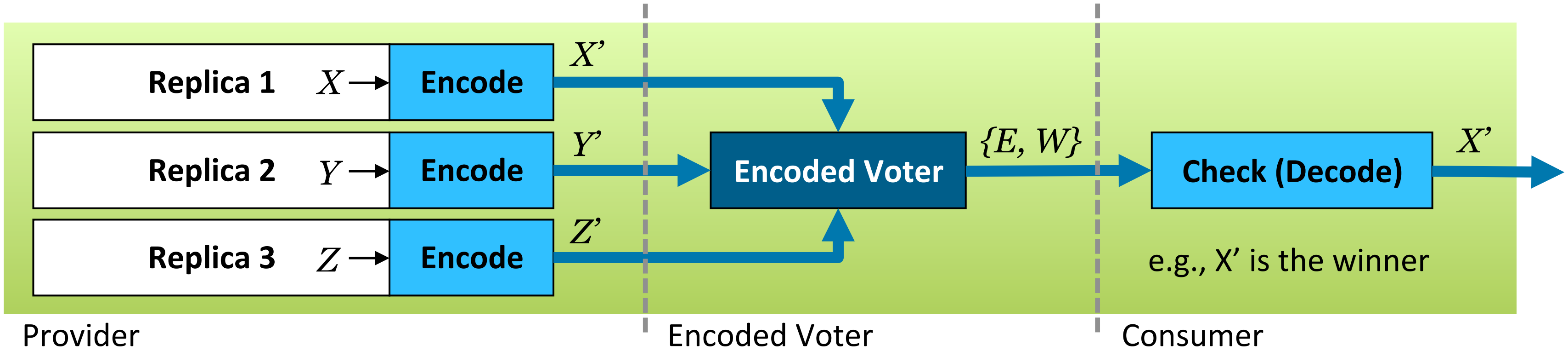
$$X' = X \times A + B_X + D$$

High-Reliability Voter – Basics (1)



- CoRed **Encoded Voter**
  - **Input**: variants (X', Y', Z')
  - **Output**: Equality set (E) and winner (W)
  - Based on EAN operations → **No decoding necessary**
- **Branch decisions (equality) on encoded data**
  - IFF difference of encoded values equals difference of static signatures
    $X = Y \Leftrightarrow X' - Y' = B_X - B_Y$
  - Each branch decision → **Unique signature**

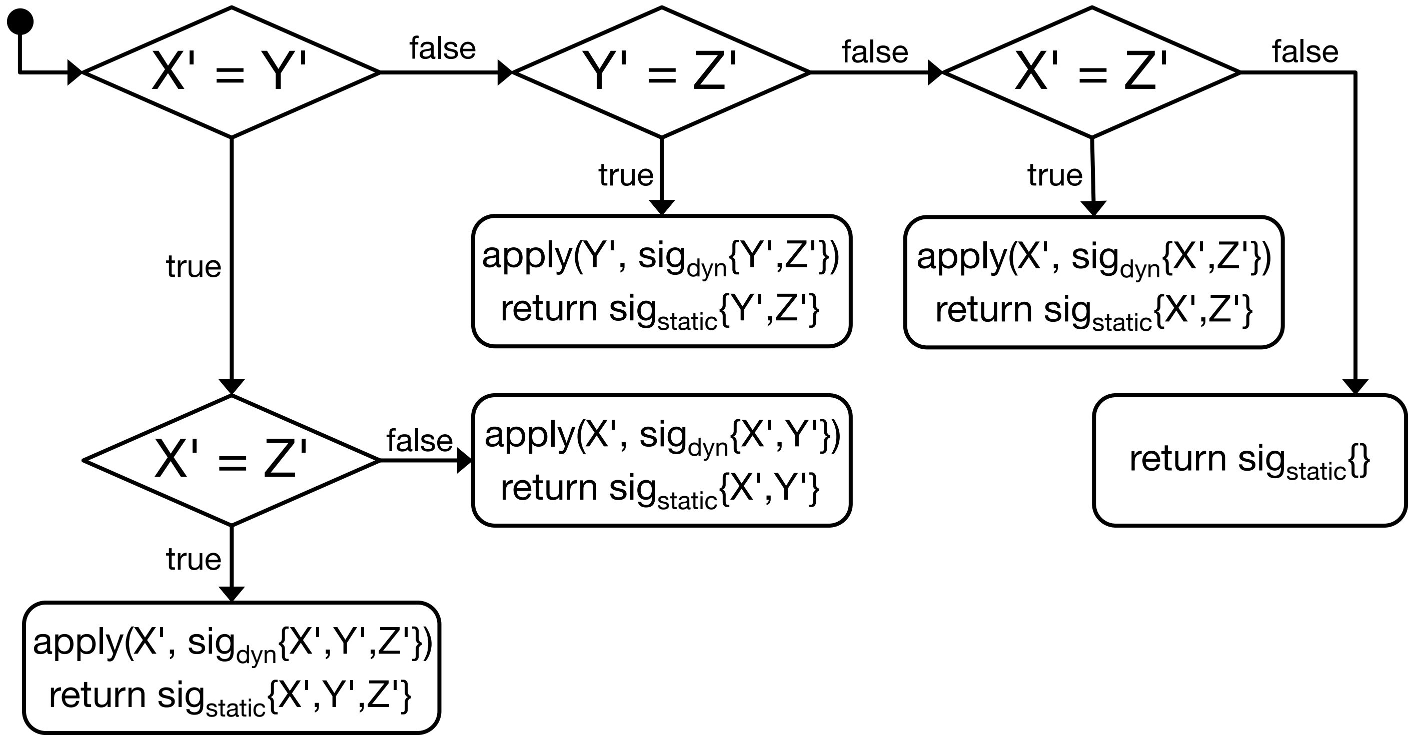
High-Reliability Voter – Basics (2)



- **Correct control-flow**
  - Valid decision → Unique control-flow path
  - Each path → **Unique signature**

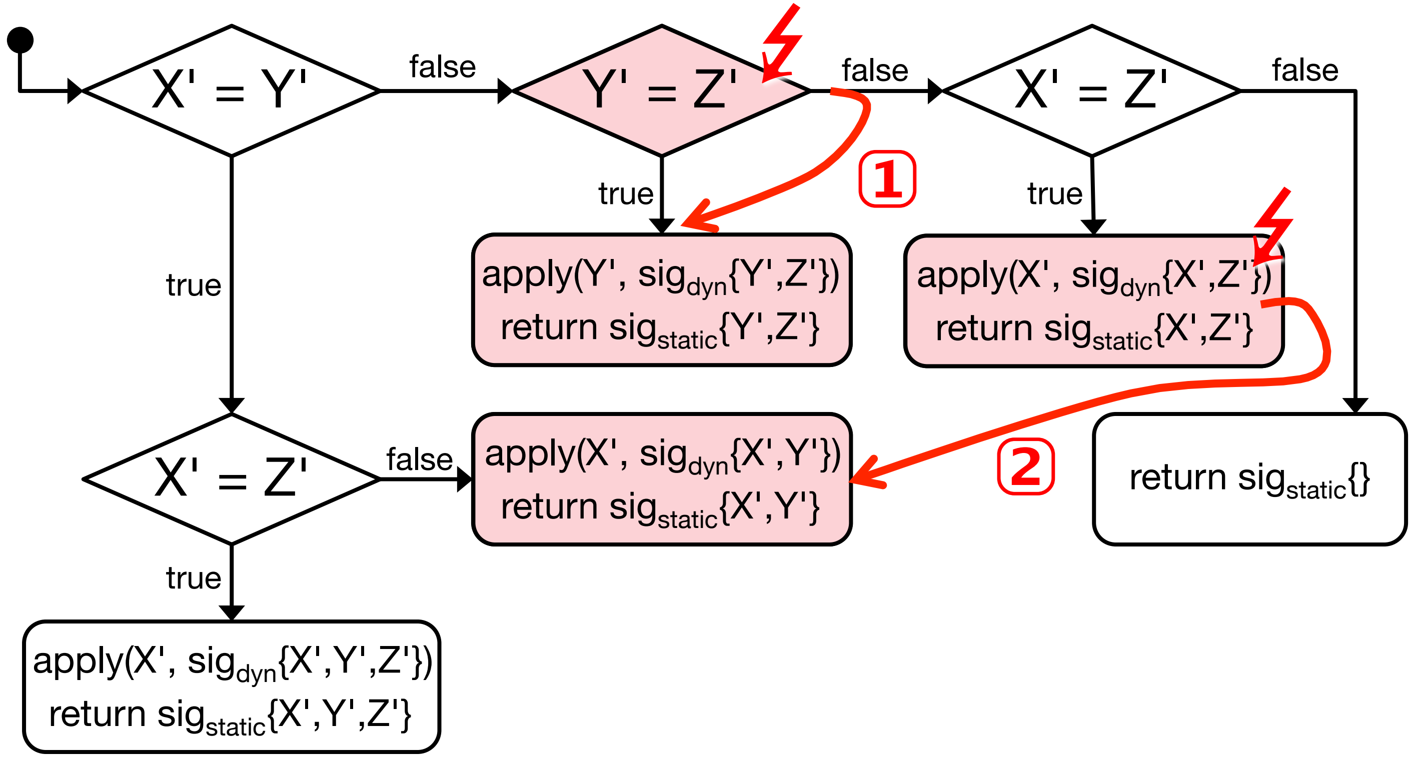
- **Control-flow signatures**
  - **Static signature** (expected value): Compile-time
    → Used as return value E
  - **Dynamic signature** (actual value): Runtime, computed from variants
    → Applied to winner W
  - **Validation**: Subsequent check (decode)

CoRed Encoded Voter – Example



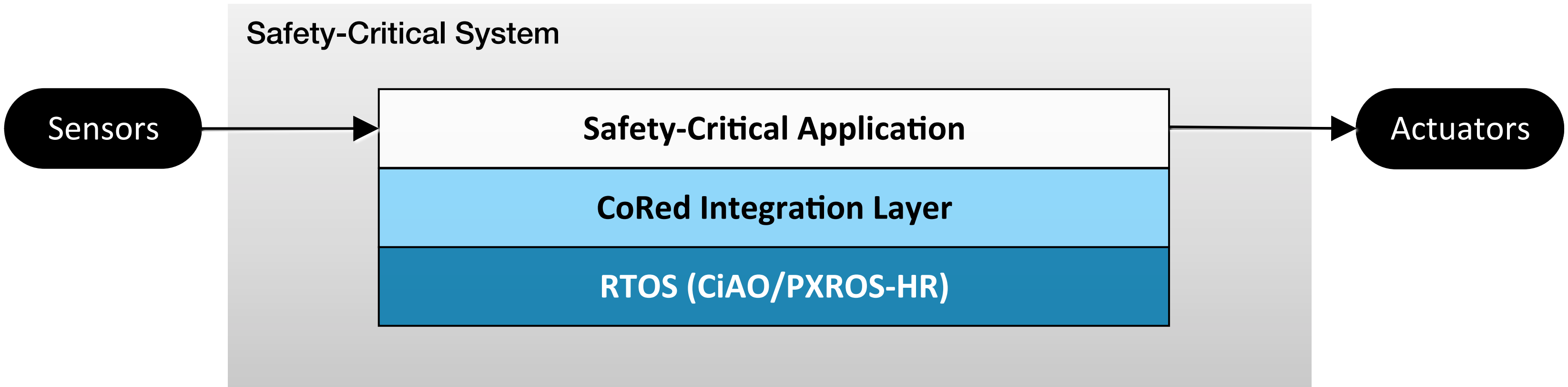
- Control-flow monitoring
  - **Finding quorum** → Static signature
  - **Reapply path specific EAN operations** → Sign winner with dynamic signature
  - **Check** → Subsequent decode

CoRed Encoded Voter – Example



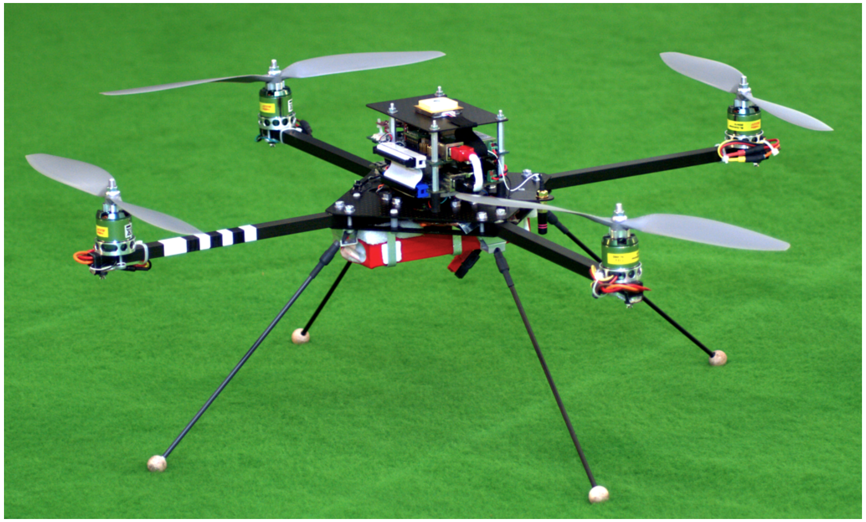
1. **Improper branch decision:** ***Y' ≠ Z'***
   - Voter elects *Y'* as winner (which is incorrect)
   - Returns *E* and *W* correctly
   - **Subsequent decode will fail!** → $sig_{static} \neq sig_{dyn}$
2. **Faulty jump**
   - Voter elects *X'* and computes *W* correctly
   - Returns incorrect *E* → Again **subsequent decode will fail!**

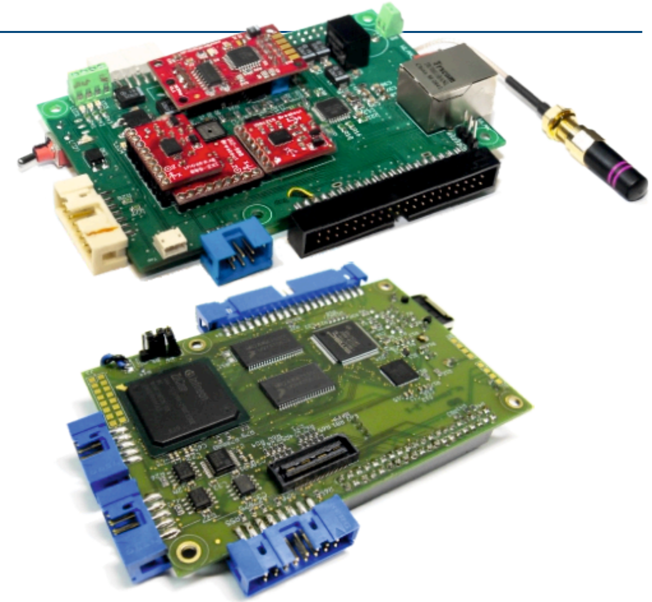
Implementation



- CoRed implementation
  - **Easy-to-use C++ templates and libraries**
  - **Hardware independent**: EAN Code and Encoded Voter
  - Thin **OS integration layer**
    - PXROS-HR (Industry-strength commercial RTOS)
    - CiAO (AUTOSAR-OS compatible)
  - CoRed artefacts → Real-time tasks and jobs
- Pragmatic
  - Allows for **implementing various redundancy patterns**
  - TMR, PaS, CP, …

CoRed Protected Flight Control

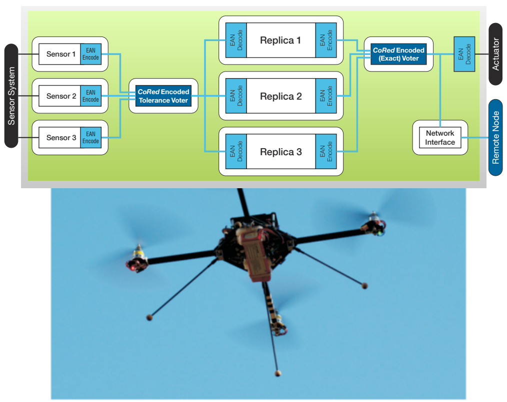




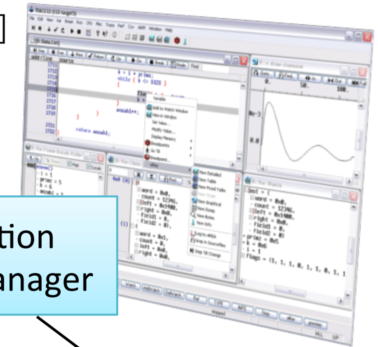
- **Target System:** I4*Copter* quadrotor platform
  - Industry-grade hardware and software
  - Triple **redundant sensor setting**
  - Multi-application system
- **Flight control application**
  - Safety-critical
  - Model-based: MATLAB Simulink
  - Embedded Coder → C++ code

Evaluation – Experimental Setup

Flight-Control Application



[Rebaudengo, 1999]



Fault-Injection Campaign Manager

Fault DB

Results DB

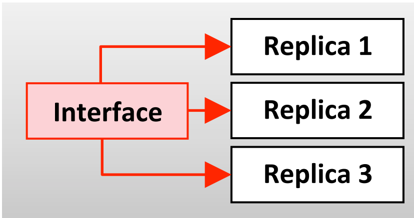
System Under Test ⟷ Hardware Debugger ⟷ Host Computer

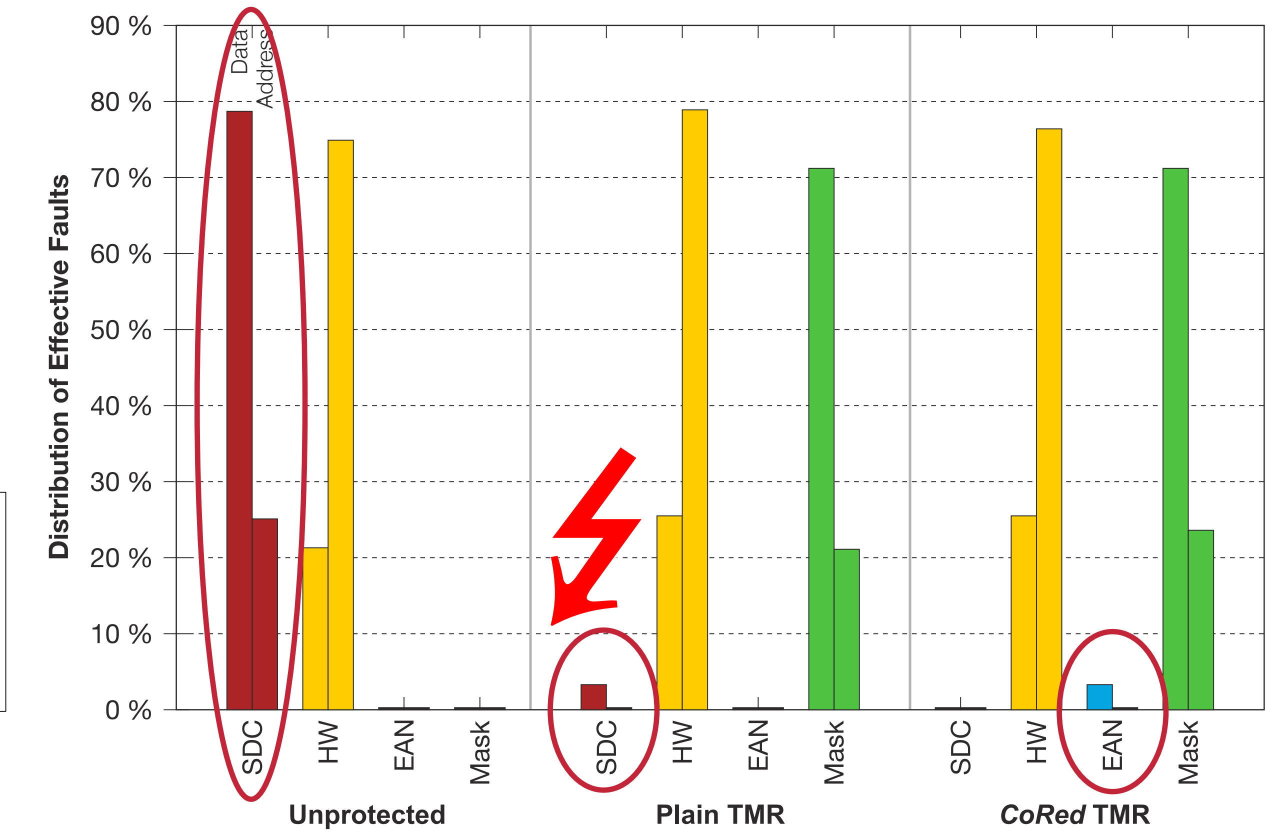
- **Fault injection → Using hardware debugger**
  - Injection of arbitrary fault patterns
  - **Minimal-intrusive** → Minimizing probe effects
- **Fault list generation** (Rebaudengo, 1999)
  - Bits × registers × instructions → Potentially **huge fault space**
  - Vast majority of faults are non-effective → Systematic elimination

Outcome: **401,592** experiments
Effective: **67,617** errors

Categories: Fail Silent, Masked, Hardware Detected, EAN-Code, Control-Flow, Silent Data Corruption

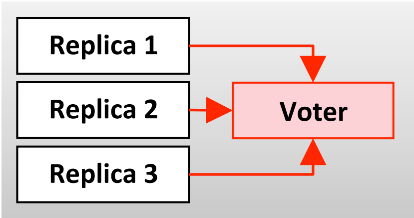
Evaluation – Experimental Results (1)

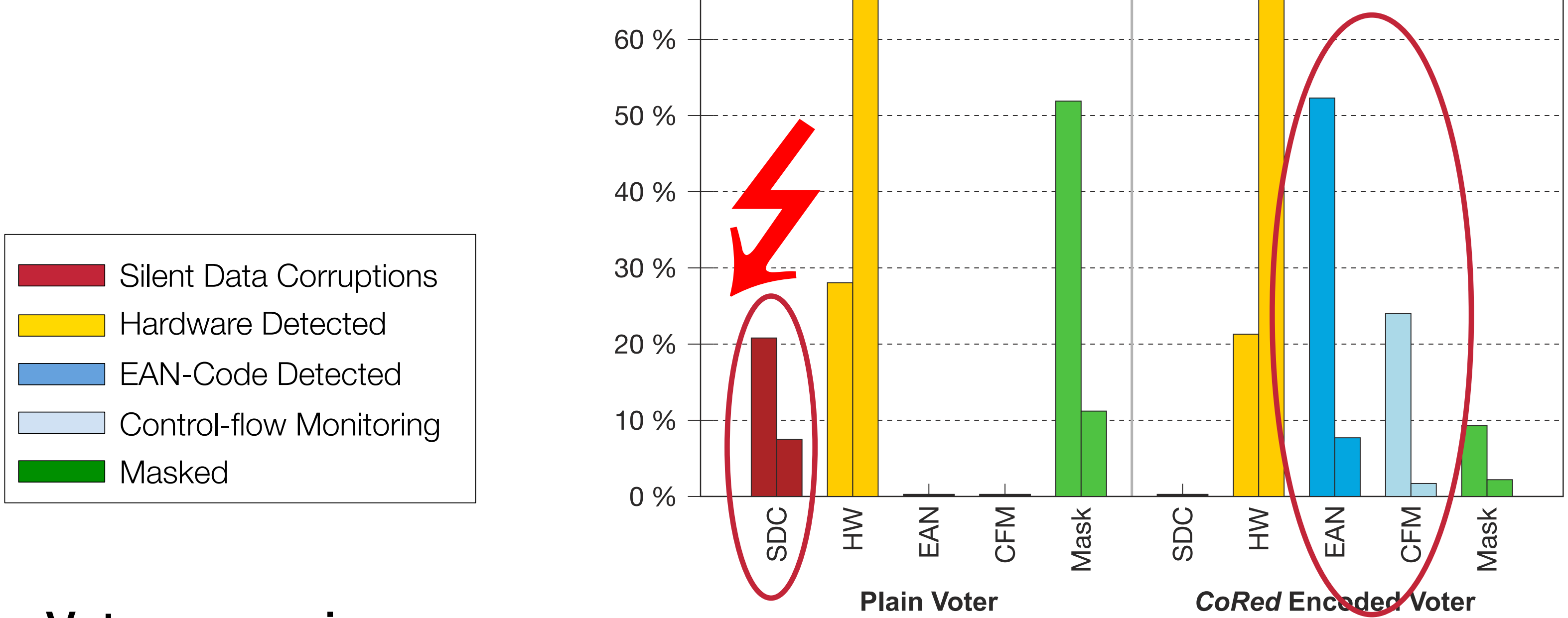




- **Redundant execution campaign (Interface)**
  - Total: ~45,000 Errors
  - **Unprotected**: Suffers from **3,622 corruptions**!
  - **TMR**: Suffers from **71 corruptions**!
  - **CoRed**: Remaining corruptions are covered → **0 corruptions**

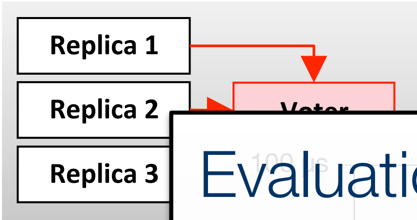
Evaluation – Experimental Results (2)

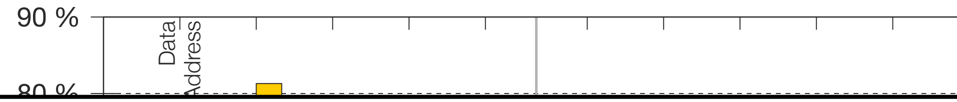


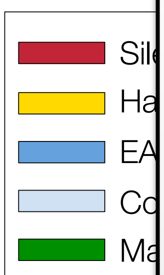


- Voter campaign
  - **Plain voter**:
    Total ~11,000 2,465 masked 7,245 retry **1,223 corruptions**
  - **CoRed Encoded Voter**:
    Total ~26,000 1,228 masked 24,682 retry **0 corruptions**

Evaluation – Experimental Results (2)





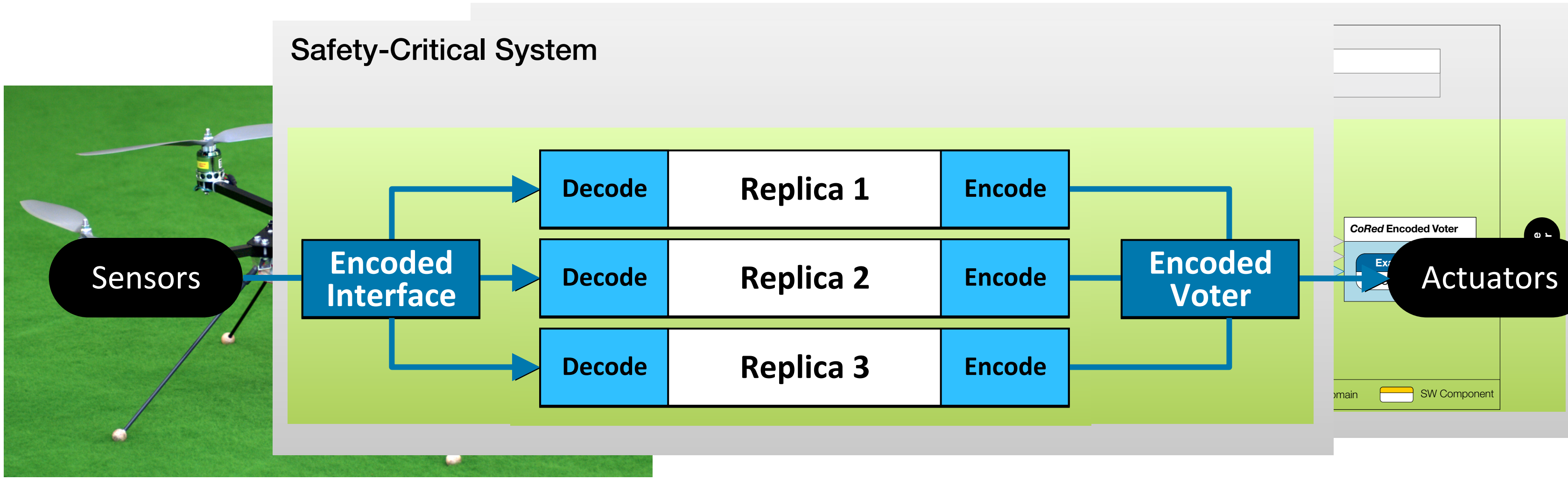


Evaluation – Overhead

- **Overhead Analysis**
  - I4*Copter* Flight-Control: **7.1% overhead** (compared to plain TMR)
  - Absolute numbers: 1,842µs (application) **10.2µs** (plain voter) vs. **77.6µs** (CoRed voter)
- **Selectivity**
  - I4*Copter* system CPU utilisation: 41% → **Full replication impossible**, CPU: 120%
  - Mission-critical replication of flight control → **possible with CoRed**, CPU: 60%

- **Voter**
  - **CoRed Voter**:
    Total ~26,000 | 1,228 masked | 24,682 retry | **0 corruptions**

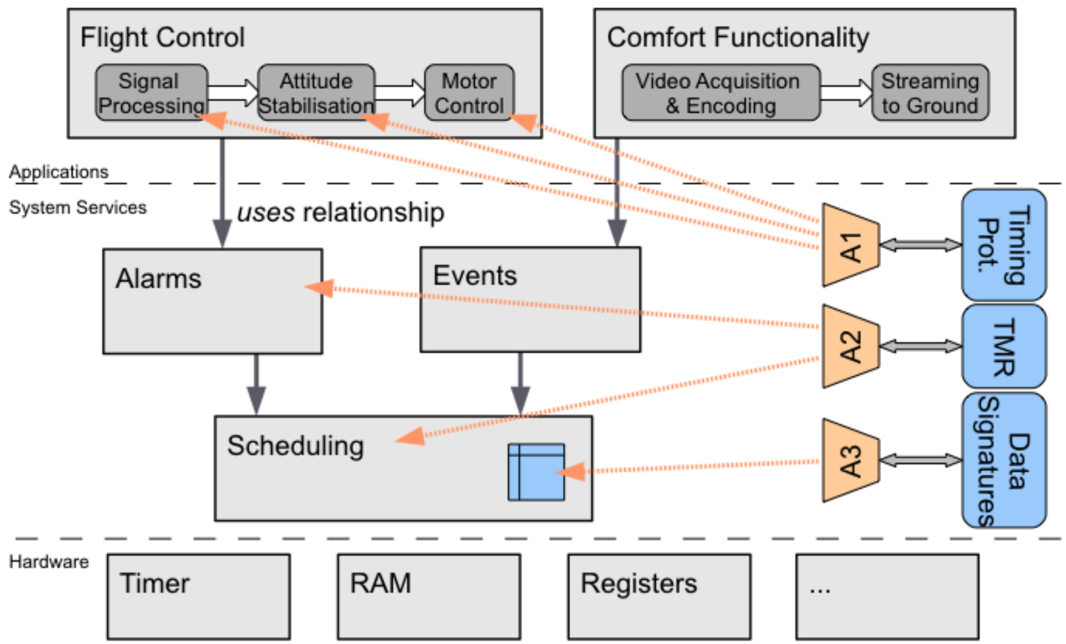
Conclusion



- The Combined Software Redundancy Approach (CoRed)
  - **Eliminate Single Points of Failure** in software-based TMR
  - No specific application knowledge necessary
  - Holistic approach: **input-to-output protection**
- Applicability: Flight control
  - I4*Copter* MAV
  - **Selective** and **composable**
- Experimental Results
  - **CoRed is effective** → Silent data corruptions can be eliminated
  - Only **7.1% overhead** (flight control example)

Outlook

- **danceOS**
  Dependability Aspects in Configurable Embedded Operating Systems
- **DFG SPP 1500, started Dec 2010**
  - Dependable Embedded Systems
  - Vision: Software-based fault tolerance for cheap but unreliable many-core hardware



Thank you!

?

Embedded Systems Initiative

CHAIR IN DISTRIBUTED SYSTEMS
AND OPERATING SYSTEMS

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SIEMENS

References

(1) International Roadmap for Semiconductors, 2001

(2) Implications of microcontroller software on safety-critical automotive systems (Infineon 2008)

(3) P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, “Modelling the effect of technology trends on the soft error rate of combinational logic,” in DSN ’02: Proceedings of the 2002 International Conference on Dependable Systems and Networks

(4) Edmund B. Nightingale, John R Douceur, and Vince Orgovan, Cycles, Cells and Platters: An Empirical Analysis of Hardware Failures on a Million Consumer PCs, in Proceedings of EuroSys 2011, Awarded "Best Paper", ACM, April 2011

(5) M. Rebaudengo and M. S. Reorda, “Evaluating the fault tolerance capabilities of embedded systems via bdm,” VTS 1999

(6) Forin, “Vital coded microprocessor principles and application for various transit systems”, 1989