Softwarebasierte Redundanz für zuverlässige eingebettete Echtzeitsysteme

CoRed - Combined Software Redundancy

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

21. Oktober 2010

ESI Embedded Systems Institute

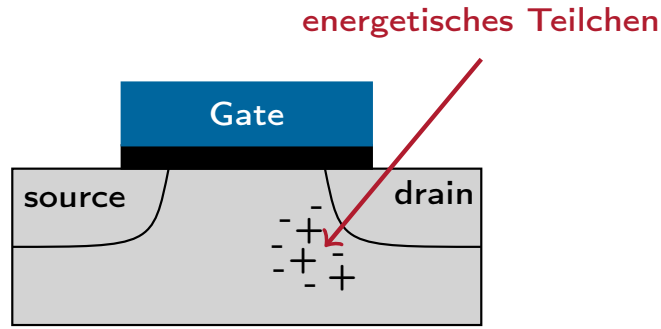
Friedrich-Alexander-Universität Erlangen-Nürnberg

Sicherheitskritische eingebettete Systeme

- Hohe Zuverlässigkeits- und Sicherheitsanforderungen
  - Safety Integrity Level (SIL)
  - Fehlertoleranz
- Hohe Kostensensitivität
  - Trend zu Multiapplikationssystemen
  - Einsatz von Mehrkernarchitekturen

Fehlermodell - Transienter Hardwarefehler

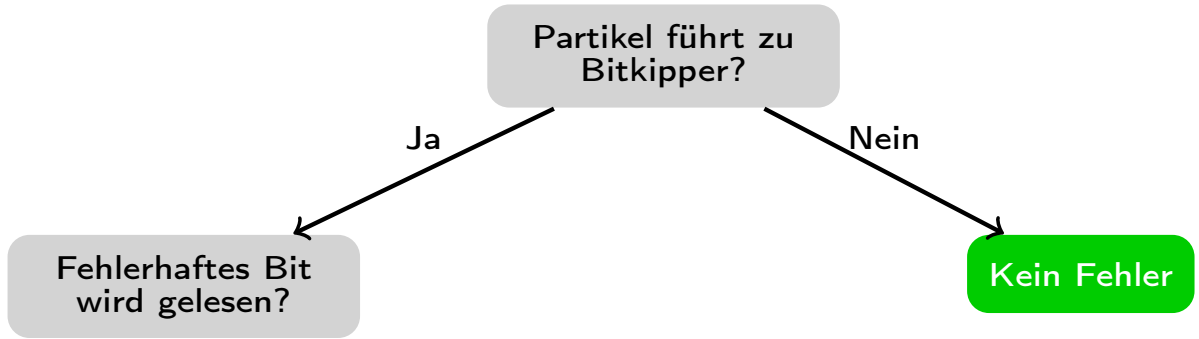
- Steigende Fehleranfälligkeit durch sinkende Strukturgrößen
- Transienter Hardwarefehler (Single Event Upset, Soft-Error)
- Verursacht durch:
  - Ionisierende, elektromagnetische Strahlung
  - Spannungsschwankungen
  - Rauschen, Übersprechen auf Leitungen



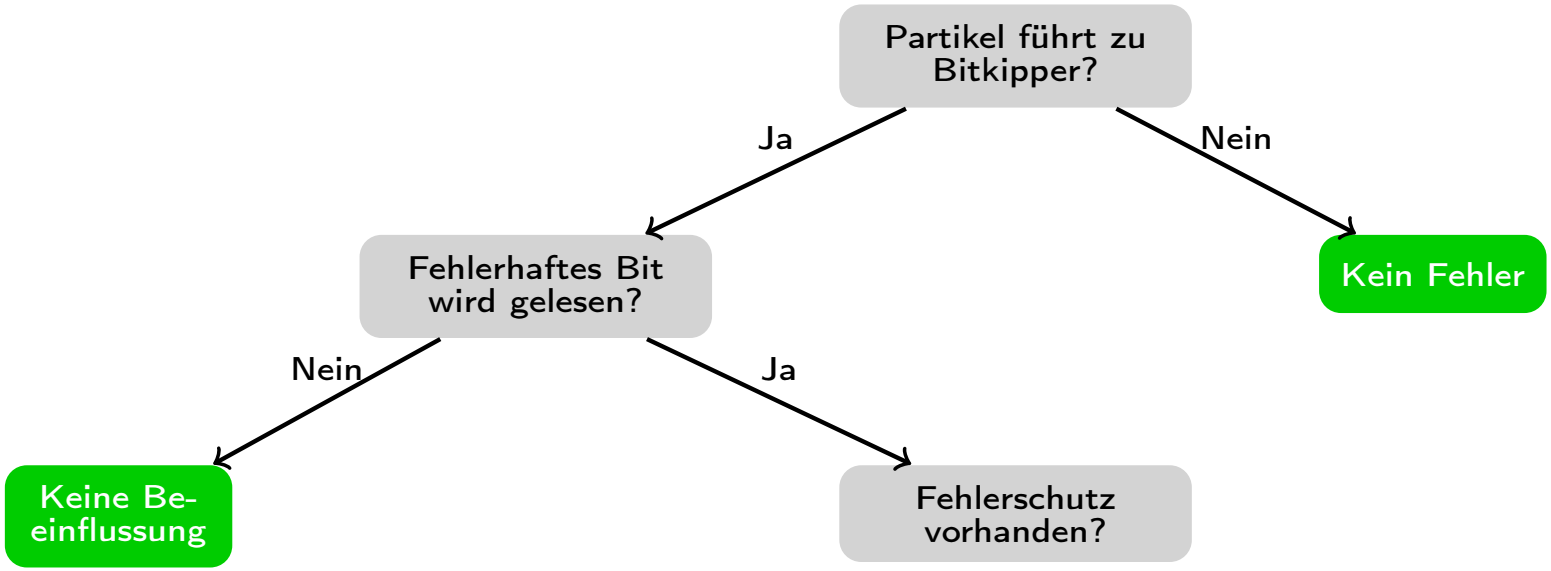
Auswirkung transienter Fehler:

"99 % of the SEUs cause single-bit flips."

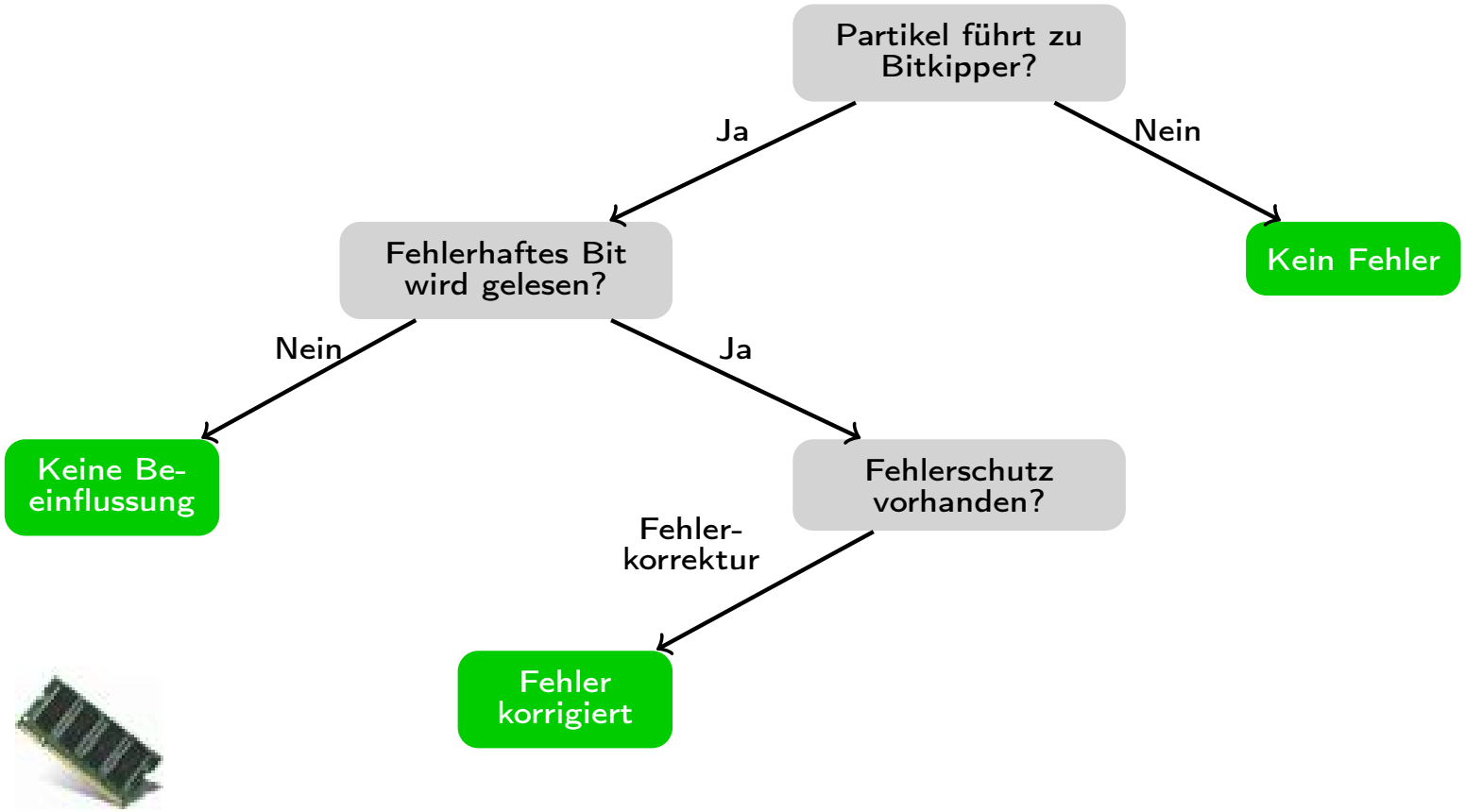
Auswirkungen transienter Fehler



Auswirkungen transienter Fehler

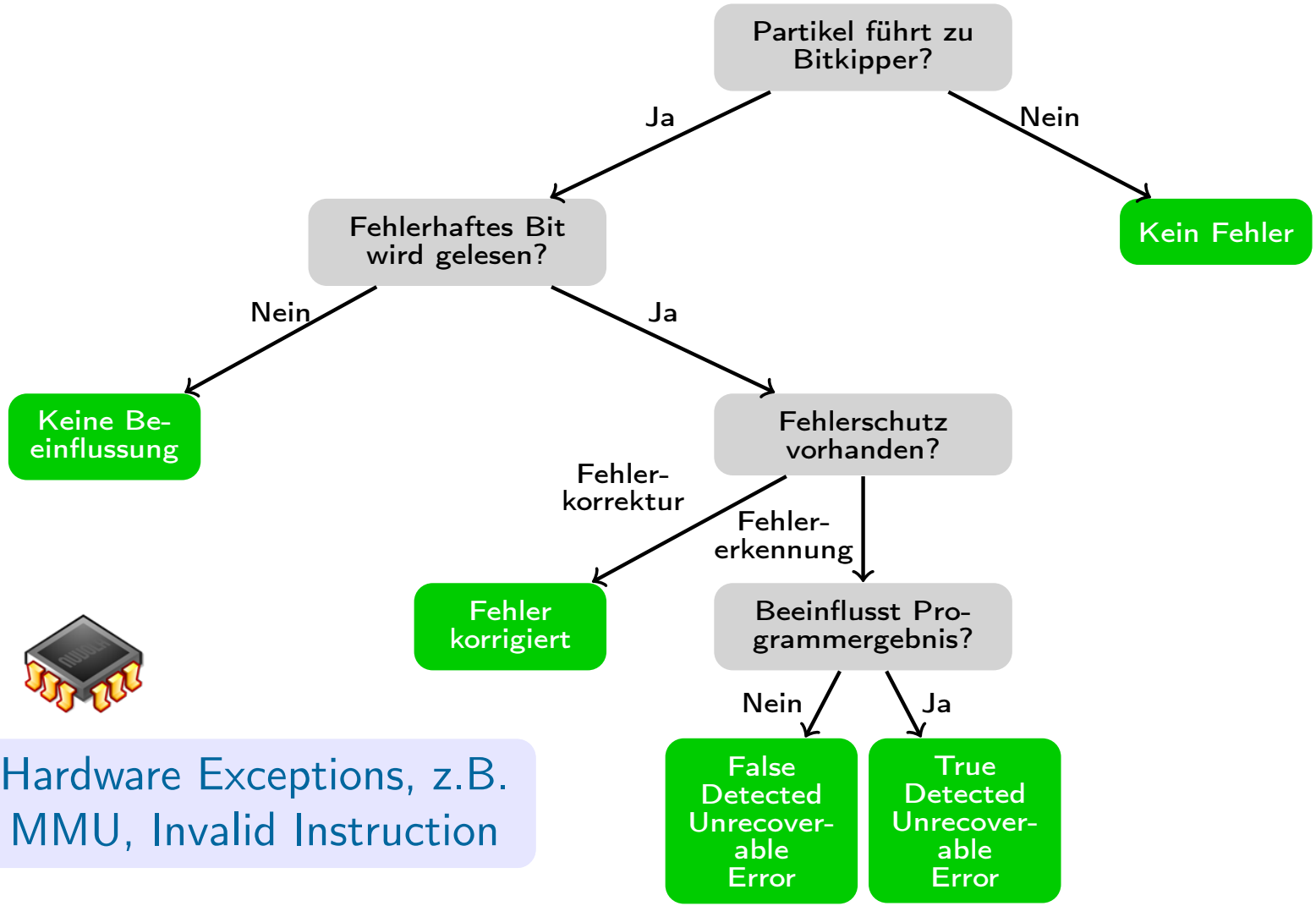


Auswirkungen transienter Fehler



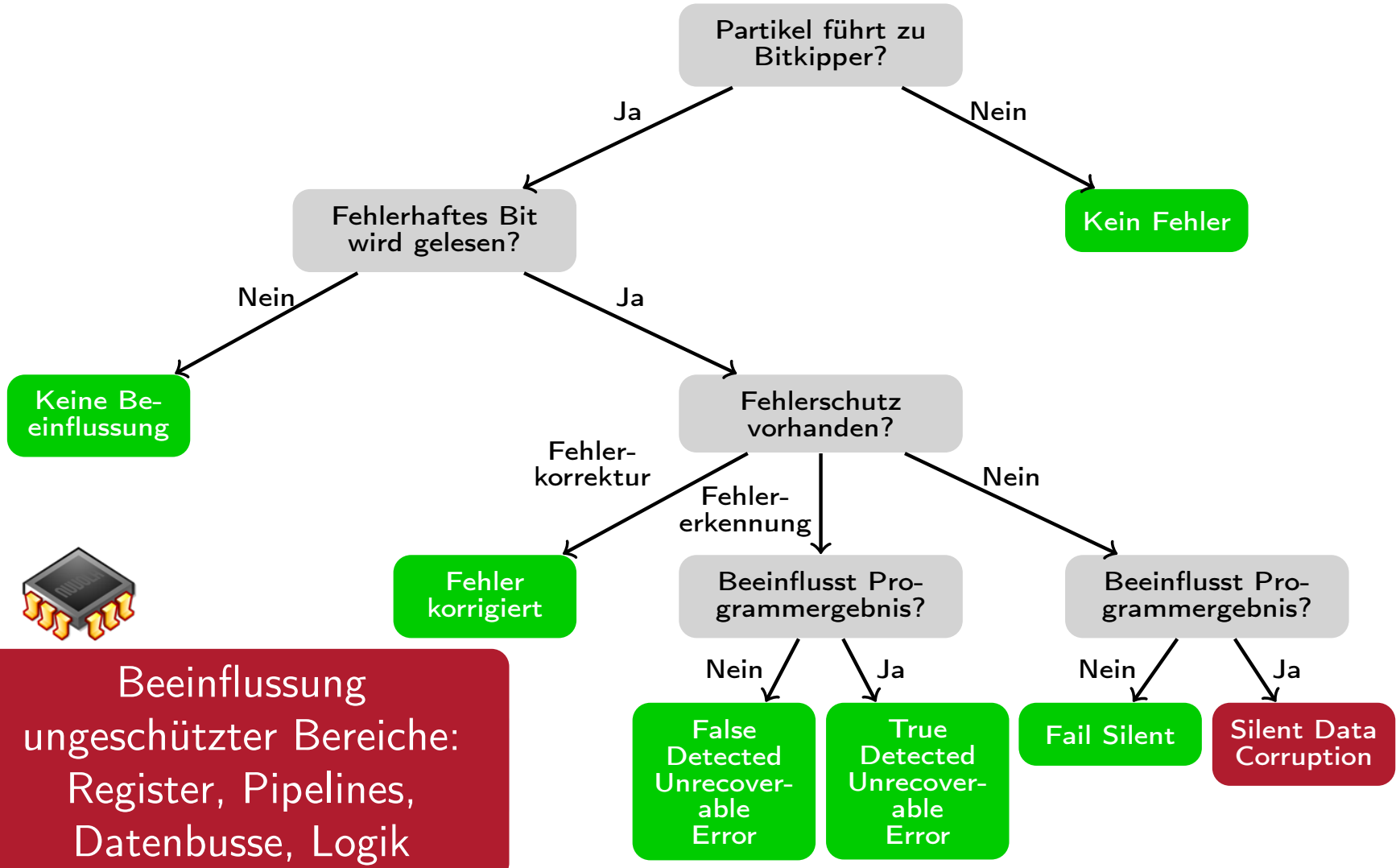
Fehlerkorrektur, z.B. ECC
RAM, Caches

Auswirkungen transienter Fehler



Hardware Exceptions, z.B. MMU, Invalid Instruction

Auswirkungen transienter Fehler



Transiente Fehler

- Aktuelle Hardware kann bestimmte Zuverlässigkeitsanforderungen nicht mehr erfüllen!

International Roadmap for Semiconductors 2002:

"Below 100 *nm*, single event upsets severely impact field-level product reliability, not only for memory, but for logic as well."

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

"Probability of failures per hour for usual microcontroller core system is not reaching SIL3 requirements."

- Chiphersteller empfehlen Einsatz von geeigneten Gegenmaßnahmen

Redundanz als Gegenmaßnahme (I)

- Hardware Redundanz: Triple Modular Redundancy (TMR)
  - Umfassender Schutz
  - Hohe Kosten, Gewicht, Platzverbrauch
  - Keine Selektivität (Multiapplikation)

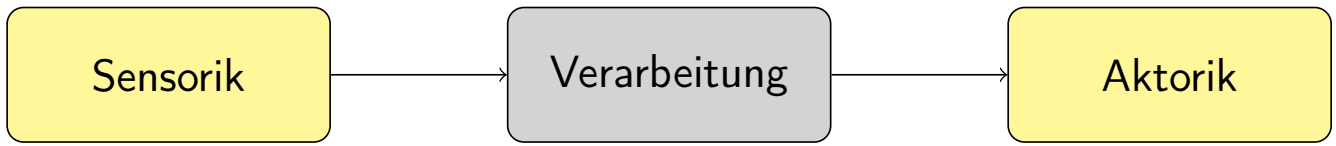
Redundanz als Gegenmaßnahme (II)

- Softwarebasierte Redundanz
  - Watchdog Konzepte (anwendungsspezifisch, komplexe Systemzustände)
  - Informationsredundanz (speicher- und rechenintensiv)
  - Zeitliche Redundanz
    - Replikation auf Instruktionsebene (benötigt spezielle Toolchains)
    - Replikation auf Softwaremodulebene (flexibel)

Redundanz als Gegenmaßnahme (II)

- Softwarebasierte Redundanz
  - ~~Watchdog Konzepte (anwendungsspezifisch, komplexe Systemzustände)~~
  - Informationsredundanz (speicher- und rechenintensiv)
  - Zeitliche Redundanz
    - ~~Replikation auf Instruktionsebene (benötigt spezielle Toolchains)~~
    - Replikation auf Softwaremodulebene (flexibel)

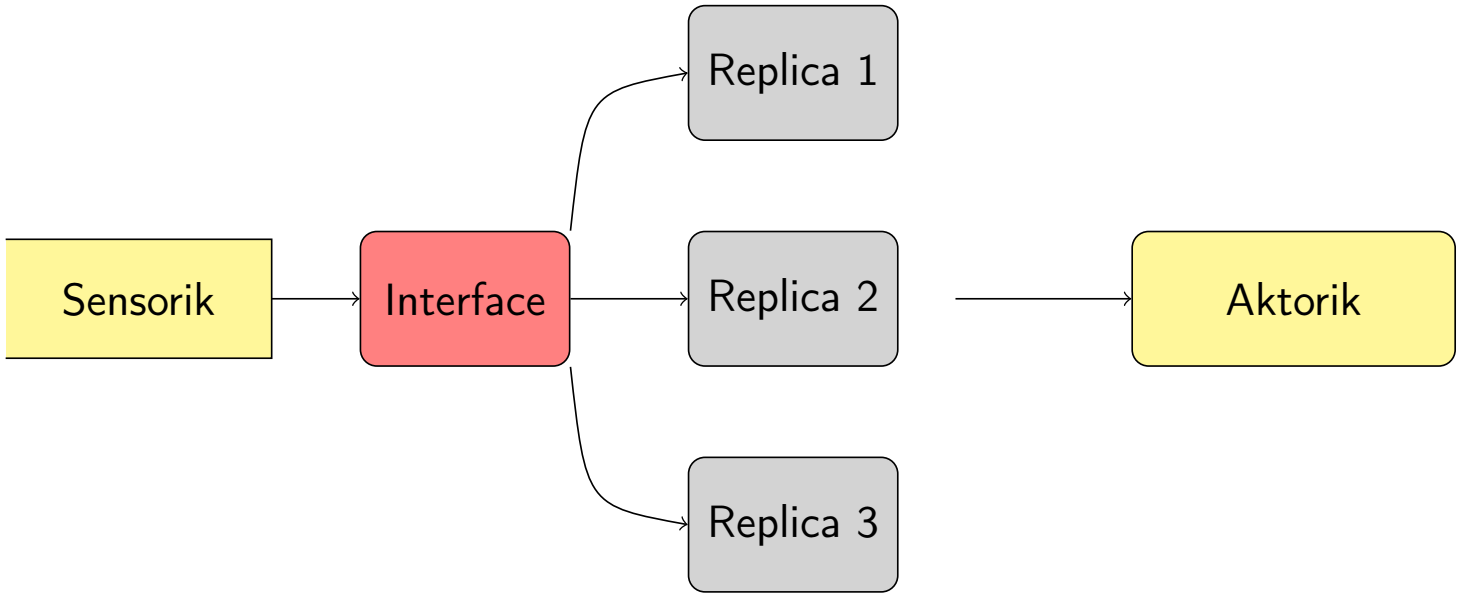
Klassische “Triple Modular Redundancy” (TMR)



Klassische “Triple Modular Redundancy” (TMR)

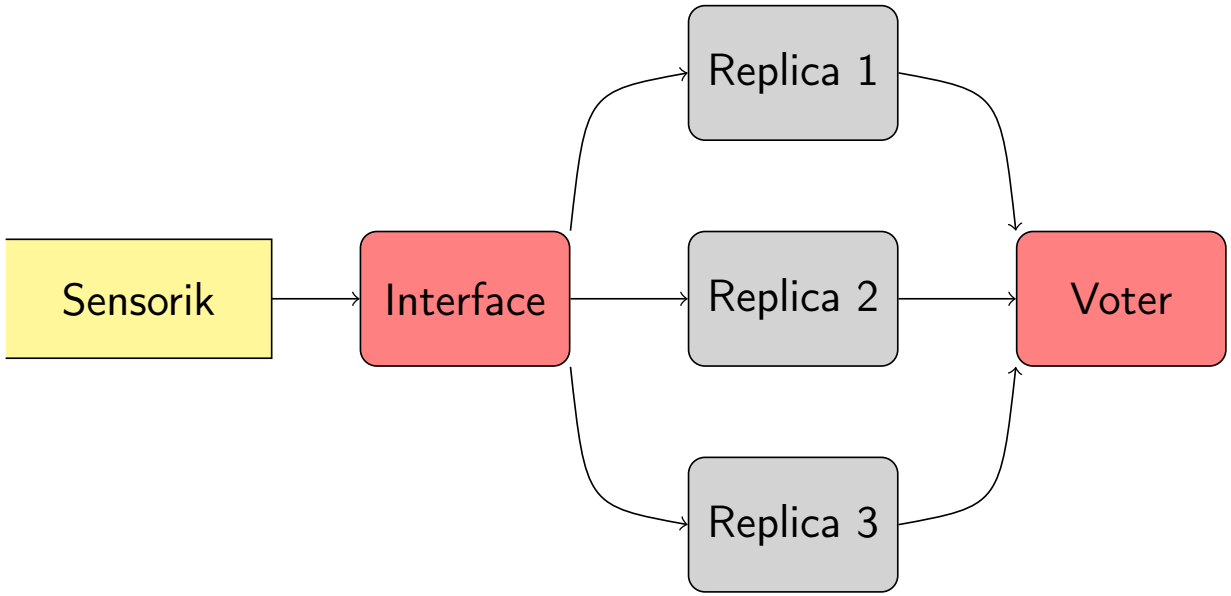
- Schnittstelle sammelt Eingangsdaten (Replikationsdeterminismus)

Klassische “Triple Modular Redundancy” (TMR)



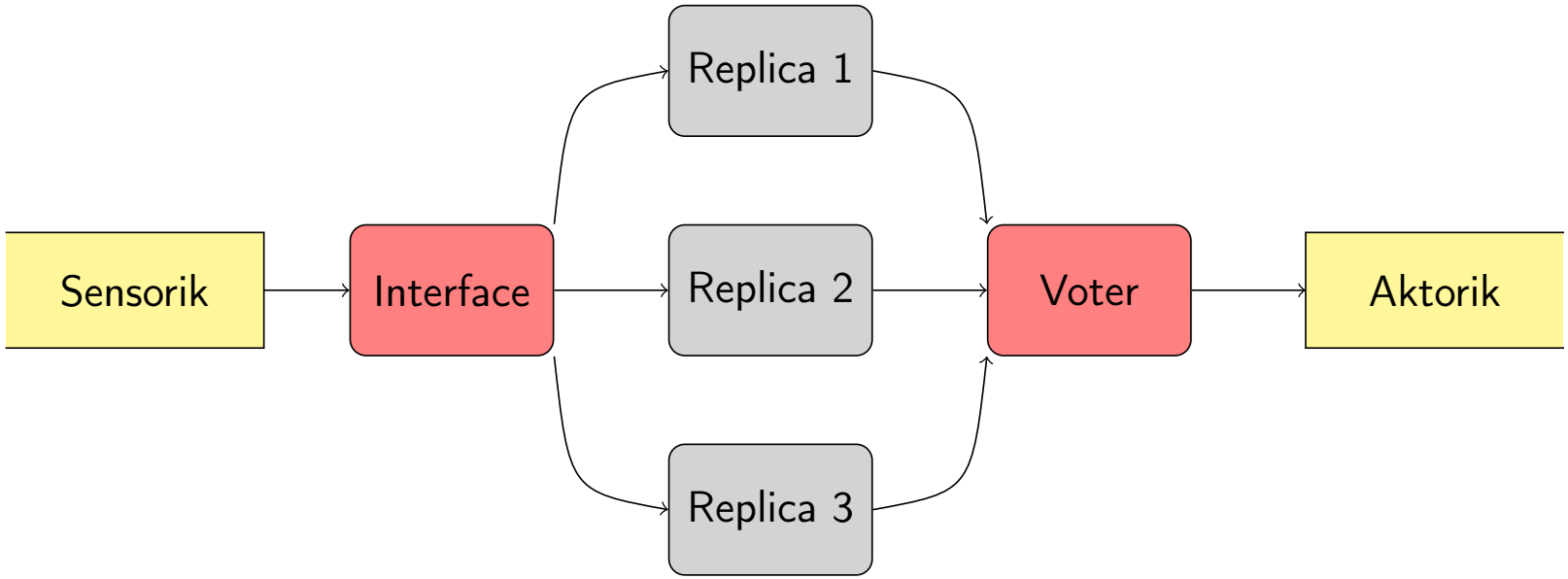
- Schnittstelle sammelt Eingangsdaten (Replikationsdeterminismus)
- Verteilt Daten und aktiviert Replikate

Klassische “Triple Modular Redundancy” (TMR)



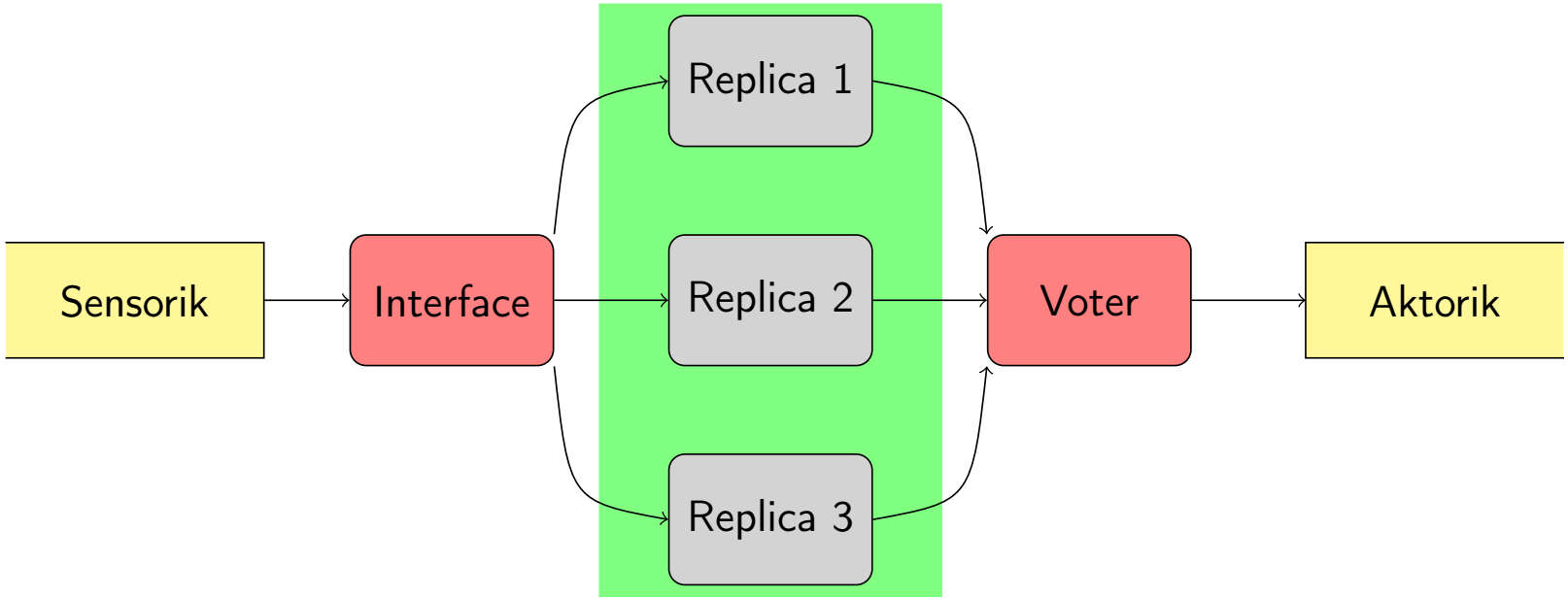
- Schnittstelle sammelt Eingangsdaten (Replikationsdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis

Klassische “Triple Modular Redundancy” (TMR)



- Schnittstelle sammelt Eingangsdaten (Replikationsdeterminismus)
- Verteilt Daten und aktiviert Replikate
- Mehrheitsentscheider (Voter) wählt Ergebnis
- Ergebnis wird an Aktuator versendet

Klassische “Triple Modular Redundancy” (TMR)

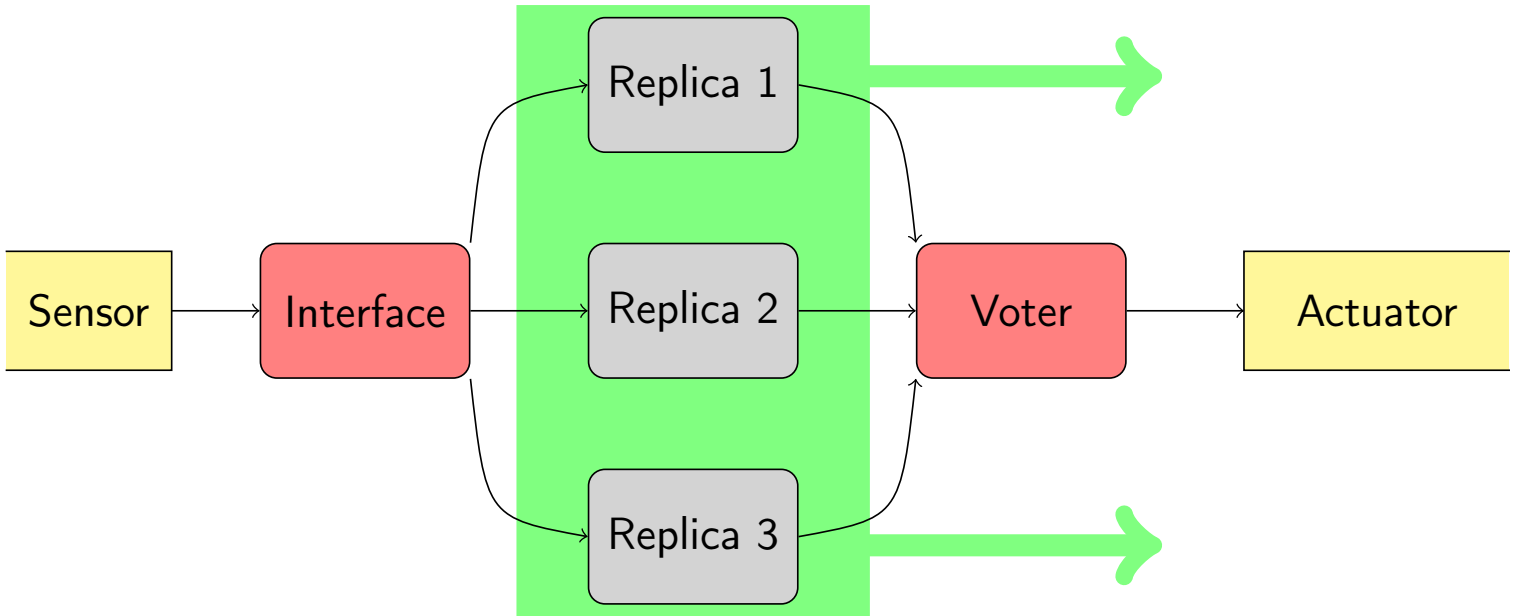


Redundanzbereich

Ausschließlich Replikatausführung.

Übersicht

Erweiterung I – kodierte Ausgangswerte

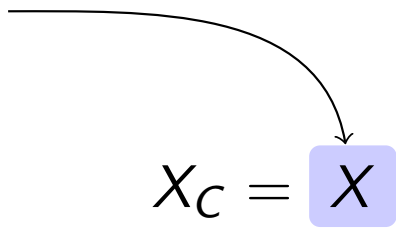


- Erweiterung der Ausgangsseite mit Informationsredundanz
- Mehrheitsentscheid über kodierte Prüfsumme

Erweiterte arithmetische Kodierung

nach Forin 1989: “Vital coded microprocessor principles and application for various transit systems”

- Arithmetisch kodierter Wert X_C
- Ausgangswert



Erweiterte arithmetische Kodierung

nach Forin 1989: “Vital coded microprocessor principles and application for various transit systems”

- Arithmetisch kodierter Wert X_C
- Ausgangswert
- Primzahl

$$X_C = X * A$$

Bitfehlererkennung
(Restfehlerwahrscheinlichkeit
$P = 1/A$)

Erweiterte arithmetische Kodierung

nach Forin 1989: “Vital coded microprocessor principles and application for various transit systems”

- Arithmetisch kodierter Wert X_C
- Ausgangswert
- Primzahl
- Variablenspezifische Signatur

$$X_C = X * A + B_X$$

Adressierungsfehlererkennung

Erweiterte arithmetische Kodierung

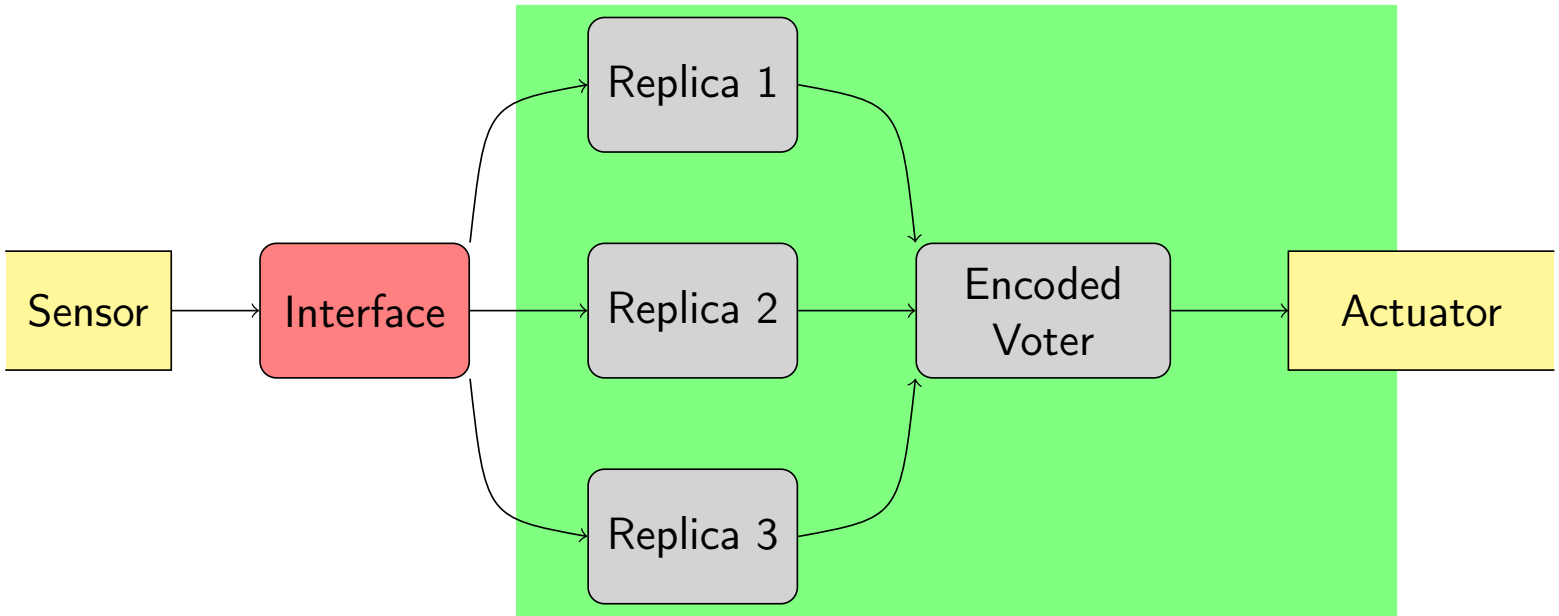
nach Forin 1989: “Vital coded microprocessor principles and application for various transit systems”

- Arithmetisch kodierter Wert X_C
- Ausgangswert
- Primzahl
- Variablenspezifische Signatur
- Zeitstempel

$$X_C = X * A + B_X + T$$

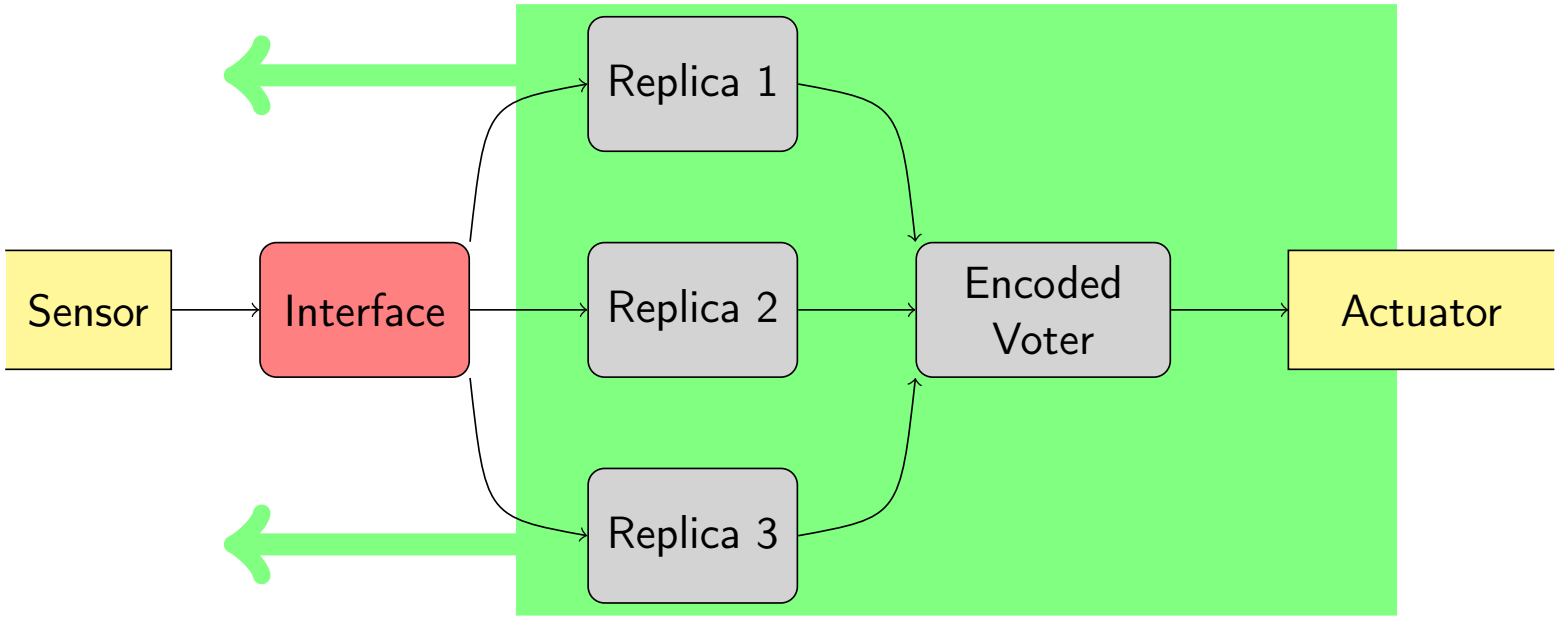
Erkennung veralteter Daten

Erweiterung I – kodierte Ausgangswerte

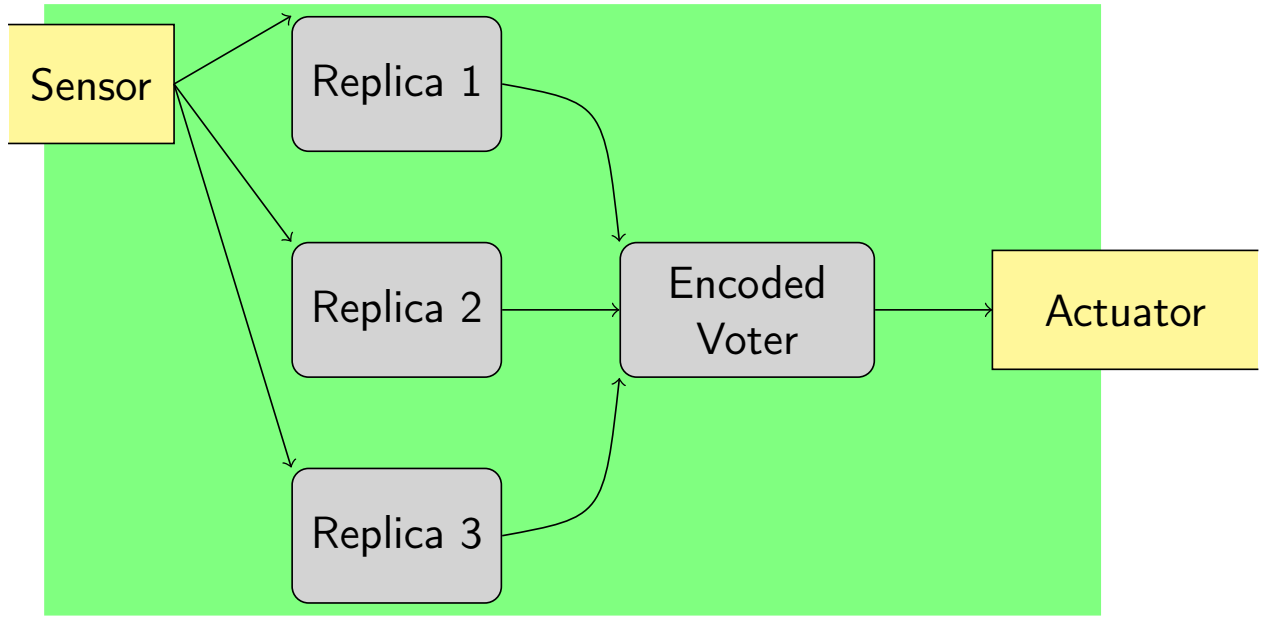


- Replikate liefern arithmetisch kodierte Ergebnisse
- Mehrheitsentscheid auf kodierten Prüfsummen
- Übertragung von kodierten Ergebnisse

Erweiterung II – Datendiversität

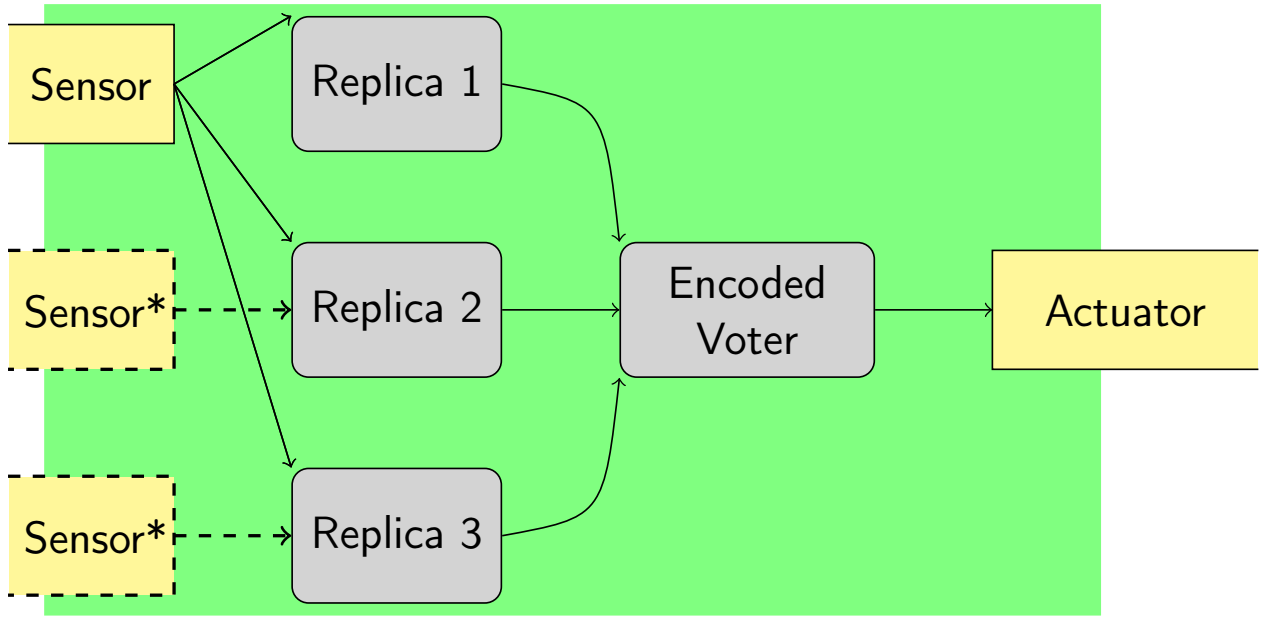


Erweiterung II – Datendiversität



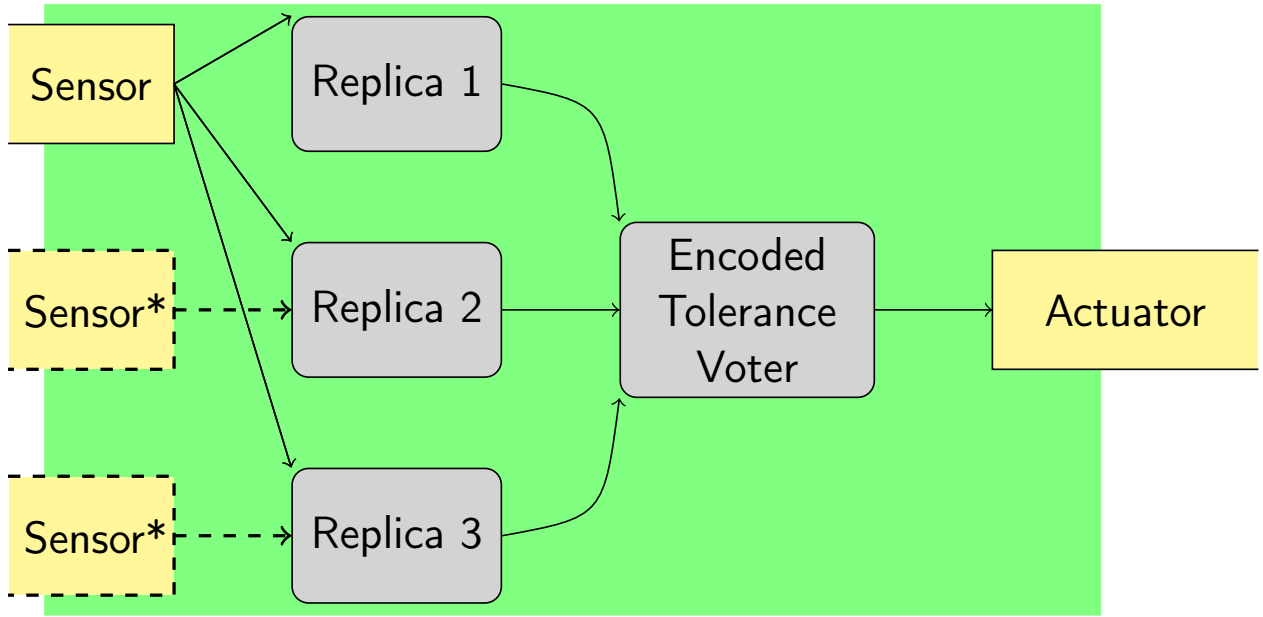
- Replikate ermitteln Eingangsdaten selbständig
- Diversitäre Eingangsdaten
  - Unterschiedliche Messzeitpunkte (zeitliche Redundanz)

Erweiterung II – Datendiversität



- Replikate ermitteln Eingangsdaten selbständig
- Diversitäre Eingangsdaten
  - Unterschiedliche Messzeitpunkte (zeitliche Redundanz)
  - Redundante Sensoren (physikalische Redundanz)

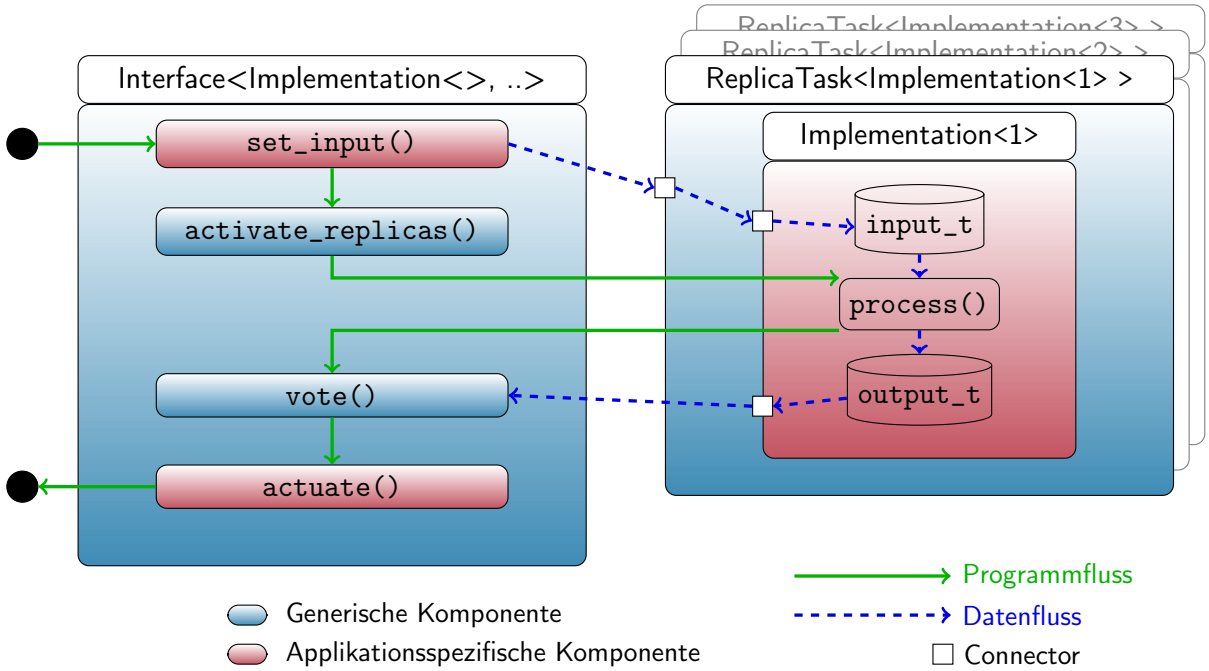
Erweiterung II – Datendiversität



- Replikate ermitteln Eingangsdaten selbständig
- Diversitäre Eingangsdaten
  - Unterschiedliche Messzeitpunkte (zeitliche Redundanz)
  - Redundante Sensoren (physikalische Redundanz)
- Mehrheitsentscheid mittels Toleranzbereich (Tolerance Voter)

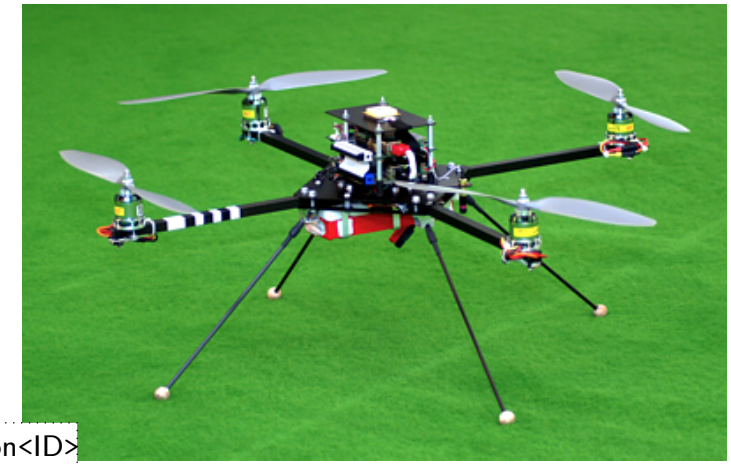
Umsetzung

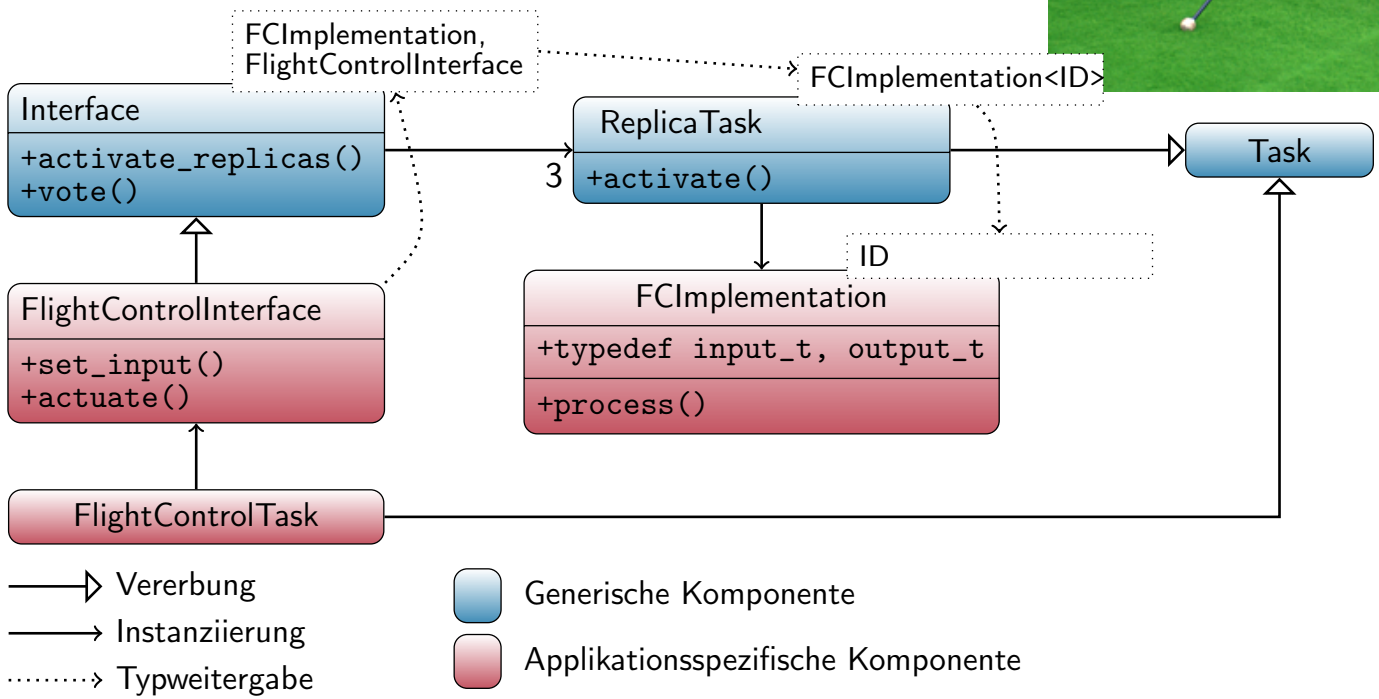
- Halb-automatische Schnittstellengenerierung
  - Nutzerdefiniert: Eingangsdaten sammeln, Agieren
  - Generiert: Ausführungskontrolle, Mehrheitsentscheid, Kodierung
- Automatisierte Komponentenreplizierung
  - Generische C++ Template Klassen (Code- und Datenreplikation)



Umsetzung

- Evaluationsplattform I4Copter
  - Integration in bestehendes System
  - Replikation der Fluglageregelung





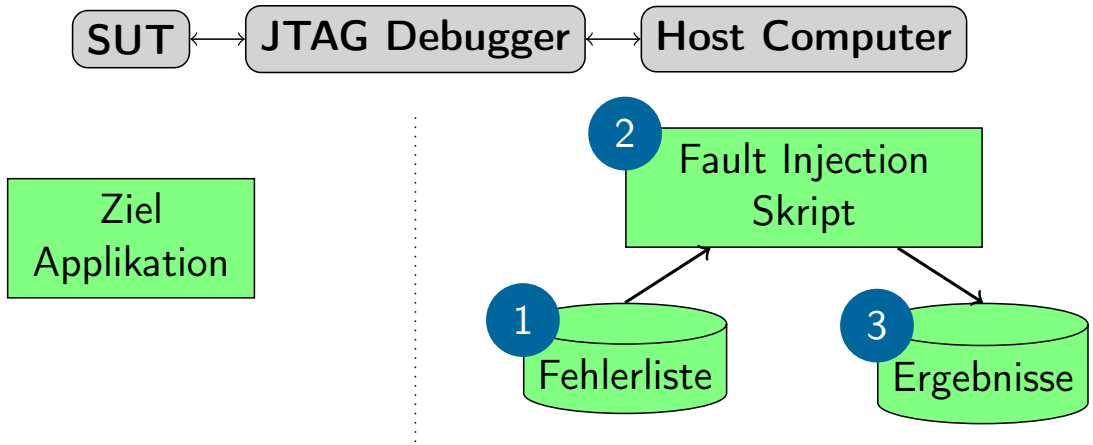
Resümee

CoRed - Combined Software Redundancy

- Selektiver Schutz vom Eingang bis zum Ausgang
  - Eliminierung der Single Points of Failure
  - Erweiterte arithmetische Kodierung am Ausgang
  - Datendiversität auf der Eingangsseite
- Einfache Handhabung
  - Flexible Replikation auf Anwendungsebene (Selektivität)
  - Zeitliche und räumliche Isolation (Echtzeitplanung, Speicherschutz)
  - Keine zusätzlichen Werkzeuge (Zertifizierung)

Übersicht

Funktionaler Test - Fehlerinjektion



- Minimal-invasiver Ansatz (probe effect)
  1. Offline generierte Fehlerliste
  2. Debugger-Skript arbeitet Fehlerliste ab
  3. Ergebnisauswertung
- Theoretischer Fehlerraum sehr groß
- Injektion an neuralgischen Punkten der redundanten Ausführung

Ergebnisse

exemplarische Fehlerinjektion I4Copter Fluglageregelung

	Nicht-Redundant	CoRed
Injizierte Fehler	1696	1696
Fail-Silent	1445	1445
Detektierte Fehler	87	53
Maskierte Fehler	0	198
Silent Data Corruption	164	0

- Injizierte Fehler werden erfolgreich maskiert/detektiert

Ergebnisse

exemplarische Fehlerinjektion I4Copter Fluglageregelung

	Nicht-Redundant	CoRed
Injizierte Fehler	1696	1696
Fail-Silent	1445	1445
Detektierte Fehler	87	53
Maskierte Fehler	0	198
Silent Data Corruption	164	0

- Injizierte Fehler werden erfolgreich maskiert/detektiert
- Ein Teil der Fehler wirkt sich auf das Betriebssystem aus

Ausblick

DanceOS – Projekt im Rahmen des SPP 1500:

Design and Architectures of Dependable Embedded Systems - A Grand Challenge in the Nano Age

- Dependability Aspects in Configurable Embedded Operating Systems
- Zuverlässigkeitsaspekte auf Betriebssystemebene
- Zusammenarbeit: FAU Erlangen und TU Dortmund
- http://www4.informatik.uni-erlangen.de/Research/DanceOS

Vielen Dank für Ihre Aufmerksamkeit!