

# Effizienter Design-Flow für Achszähler

Sebastian Kinder, Rolf Drechsler  
University of Bremen  
{kinder,drechsle}@informatik.uni-bremen.de



# Überblick

- Motivation
- Grundlagen
  - SystemC
  - Property Specification Language
  - Bounded Model Checking
- Mehrachszählverfahren
  - Modellierung
  - Validierung
  - Verifikation
- Zusammenfassung & Ausblick

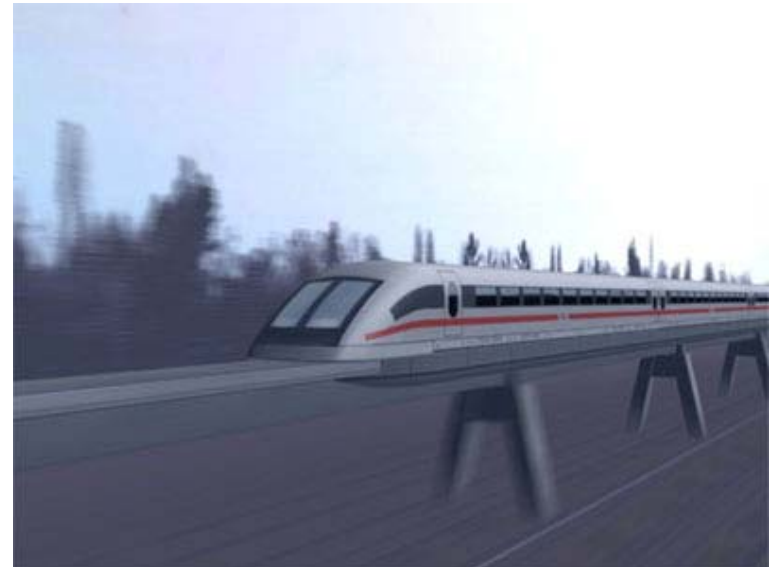
# Motivation

Häufige Verwendung von Schaltungen und Steuerungskomponenten



# Motivation

- Einsatz in Sicherheitskritischen Bereichen
  - Medizinische Apparaturen
  - Stellwerke
  - Zugsteuerungs-computer
- Steigende Komplexität
  - ➔ Simulation nicht ausreichend
  - ➔ Formale Techniken notwendig



# SystemC

- Open SystemC Initiative (OSCI)
- C++ Klassenbibliothek
  - Zyklengenaue Modelle
    - Software Algorithmen
    - Hardware Architekturen
  - System-Level Entwürfe
- Benötigt C++ Compiler (ANSI)
- IEEE Standard

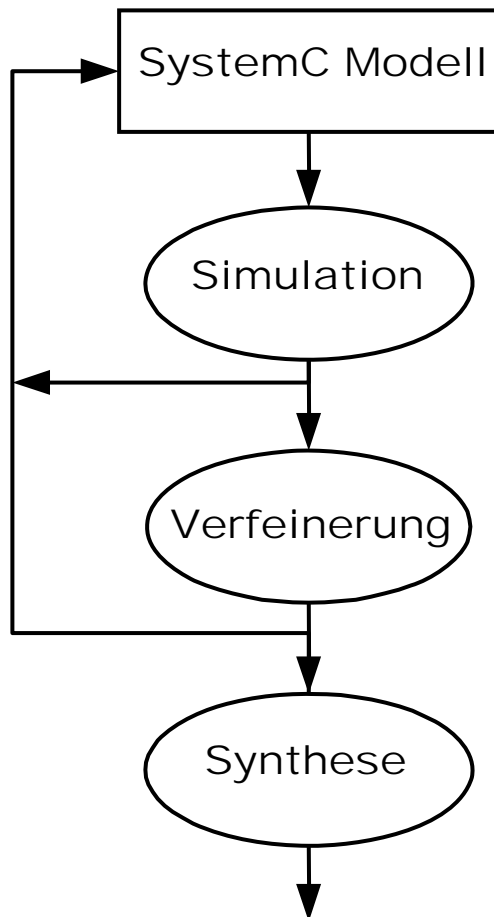
# SystemC – Modellierung

- Konstrukte zur Modellierung von Systemarchitekturen
  - Hardware timing
  - Nebenläufigkeit
  - ...
- Ausführbare Spezifikation
  - Prüfung der Vollständigkeit
  - Eindeutige Interpretation
  - Validierung von Systemfunktionalität

# SystemC – Features

- Unterstützung von Hardware-Software Co-Design
- Zyklensbasierter Simulator
- Modellierung auf verschiedenen Abstraktionsebenen
- Kommunikationsprotokolle
- Debugging Unterstützung
- Waveform-Tracing

# SystemC – Methodik



- **Verfeinerungskonzept:**

**Keine Übersetzung von C  
nach HDL (timing-Konstrukte)**

- **System- bis RTL-Modell eine Sprache**

# Property Specification Language

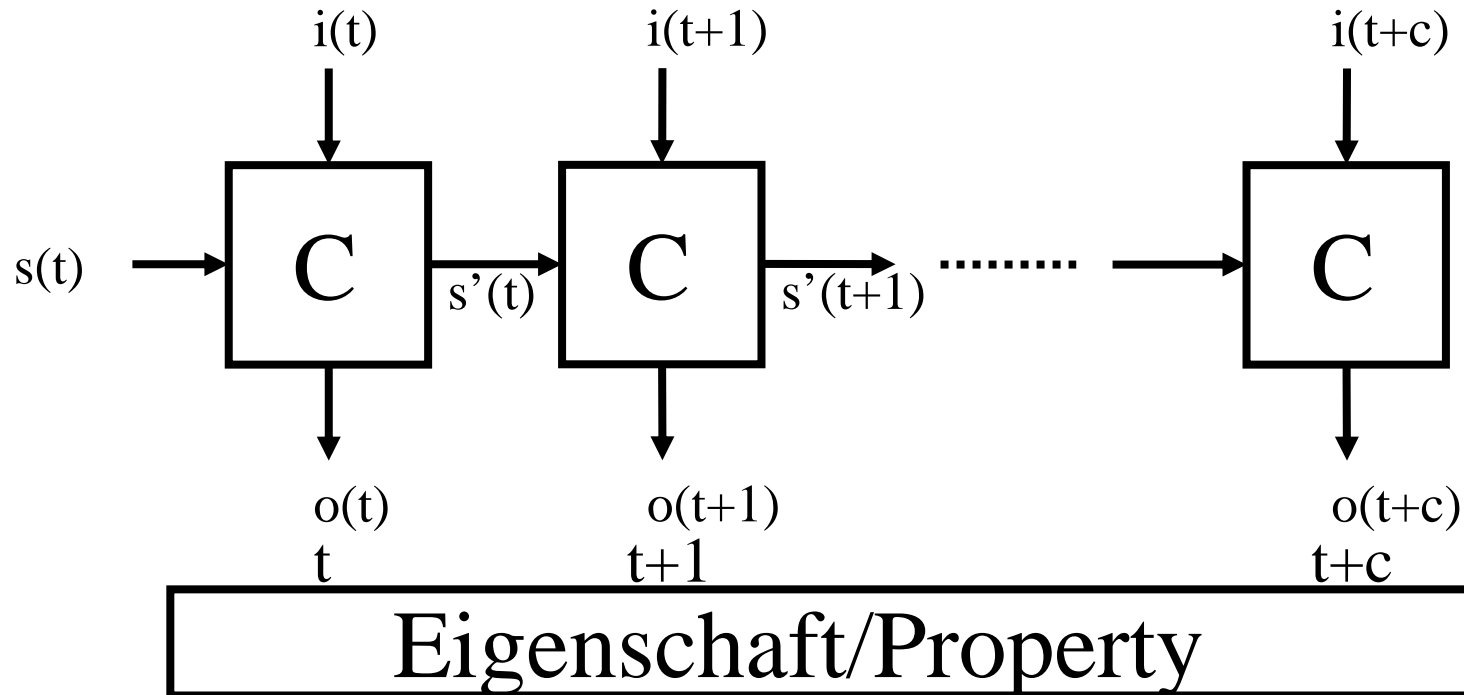
- PSL von Accellera
- Eigenschaften bestehen aus 2 Teilen:
  - Liste von Annahmen (Assumptions)
  - Liste von Zusicherungen (Commitments)
- Im Allgemeinen gilt:  
Wenn die Annahmen gelten, müssen auch die Zusicherungen gelten.

# PSL – Beispiel

Immer wenn Signal  $x = 1$  wird, muss Signal  $y = 2$  sein (2 Taktzyklen später)

```
always(  
    //assumption  
    (x = 1)  
    //proof  
    ->  
    next[2] (y = 2)  
)
```

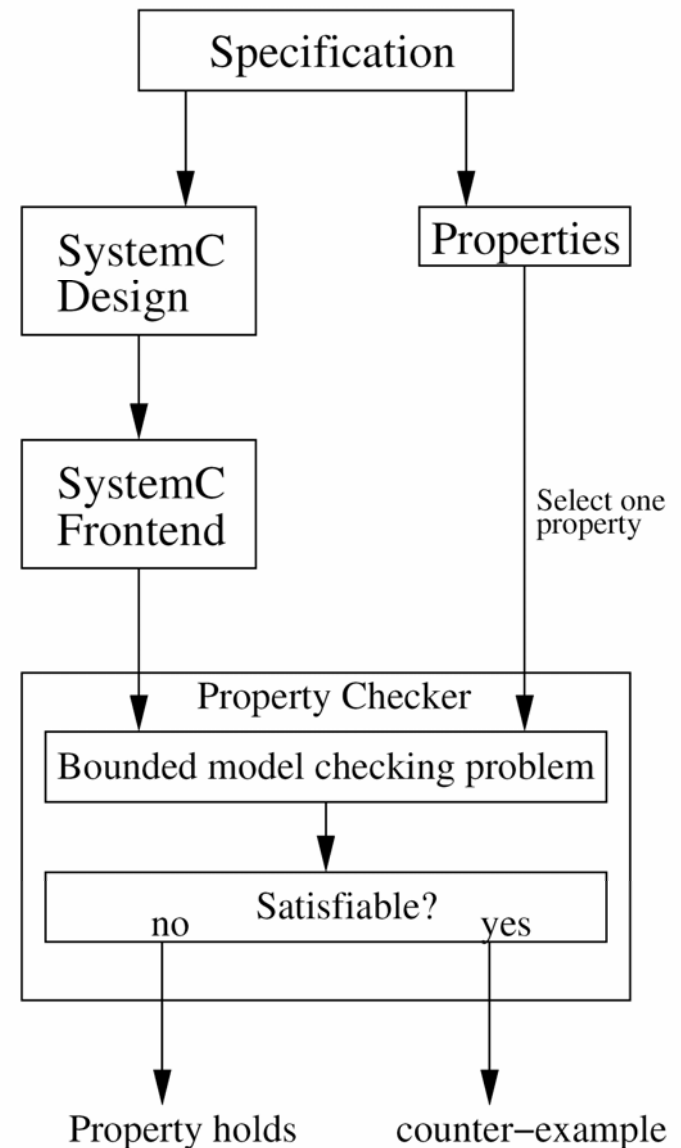
# Bounded Model Checking (BMC)



- Umwandeln eines sequentiellen Problems in ein kombinatorisches Problem durch Abrollen des Designs
- Beobachtungszeitraum: von  $t$  bis  $t + c$

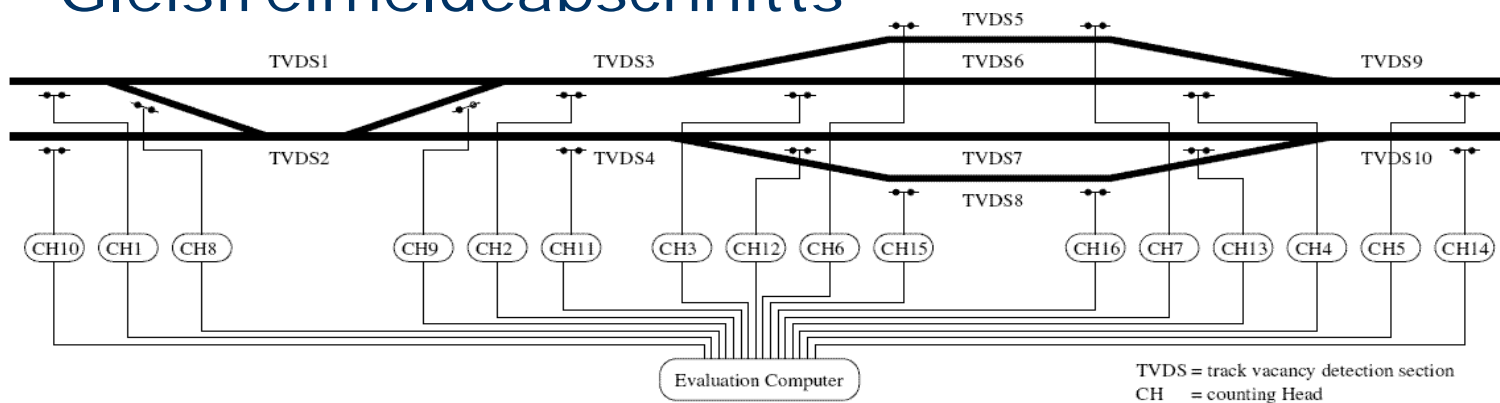
# BMC flow

- SystemC Beschreibung in einen endlichen Zustandsautomaten überführen (FSM)
- FSM und Eigenschaft in ein SAT Problem transformieren
- SAT Instanz auf Erfüllbarkeit prüfen



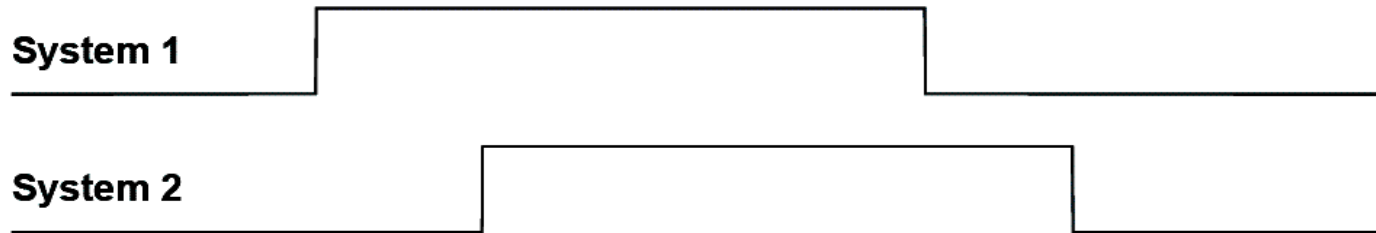
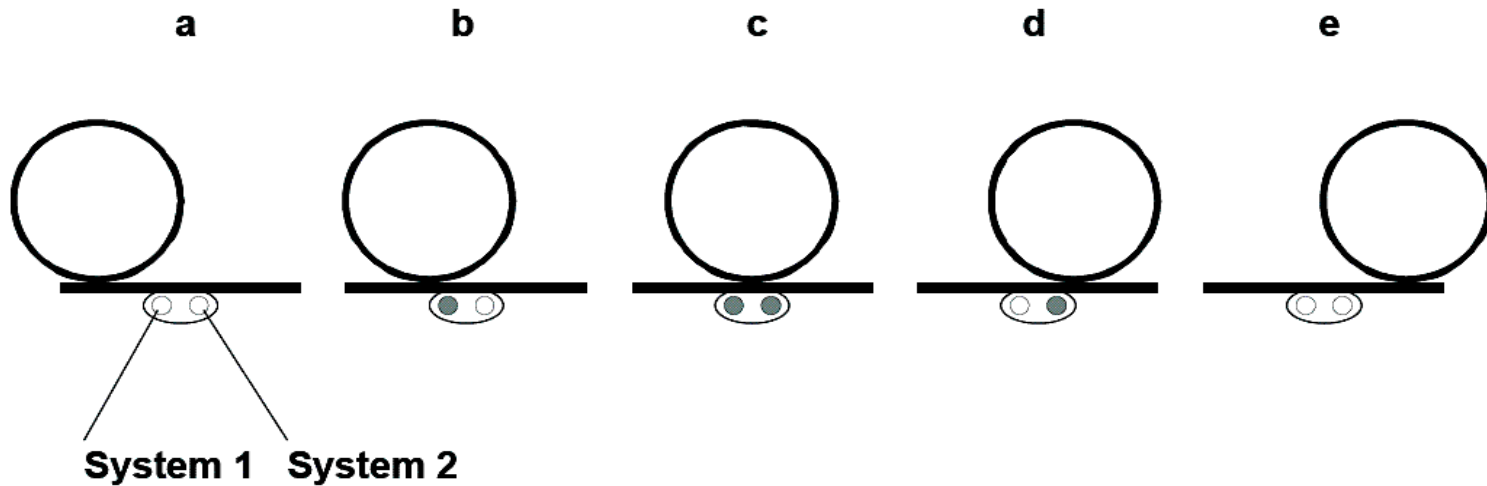
# Mehrachszählverfahren (MAZV)

- Verfahren zur Zählung von Achsen
- Zählpunkt am Anfang und Ende eines Gleisfreimeldeabschnitts

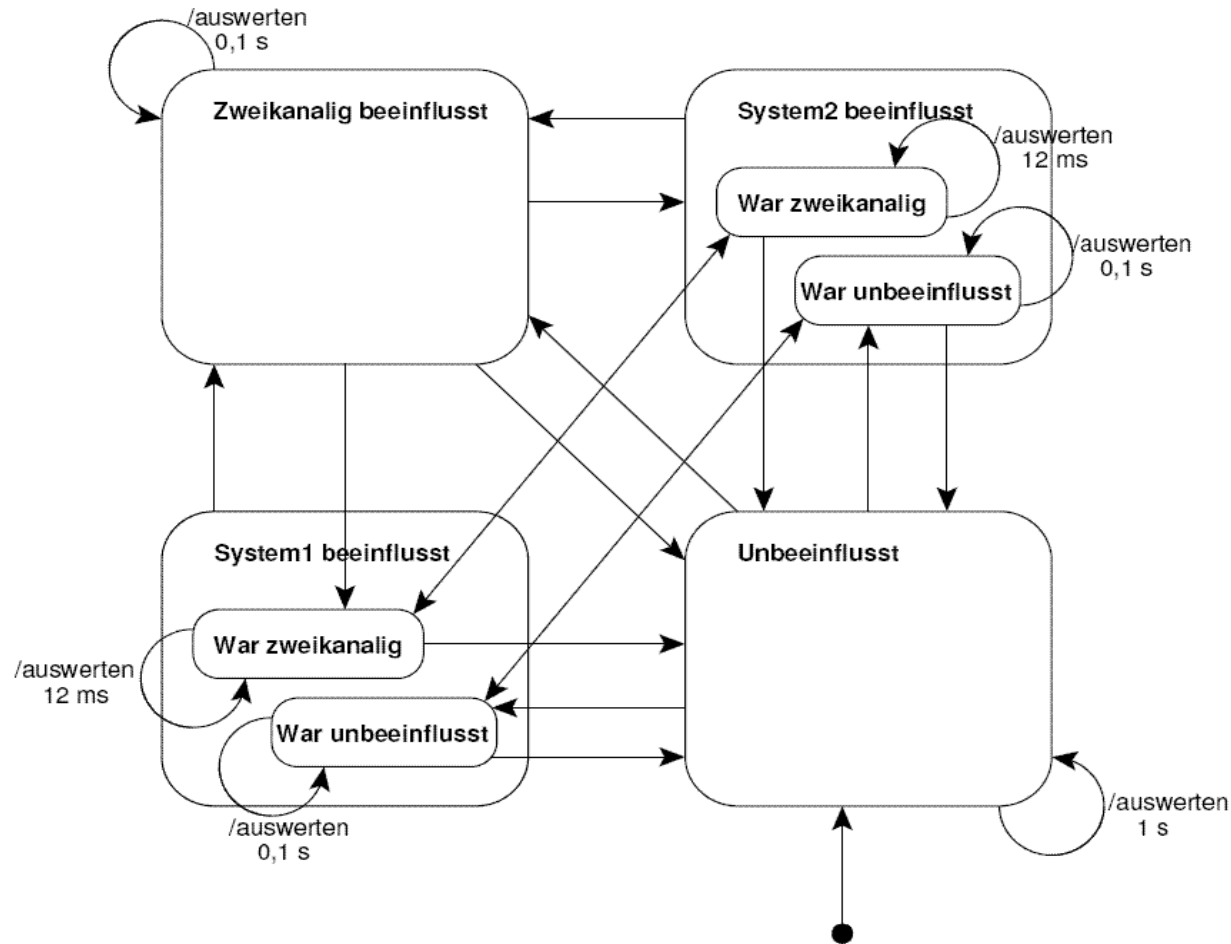


- ➔ Generierung von Achszähldaten
- ➔ Vergleich der Zählergebnisse
- ➔ Frei-/Besetztmeldung für den Abschnitt

# Achszählung mit Doppelsensor



# Zustandsautomat

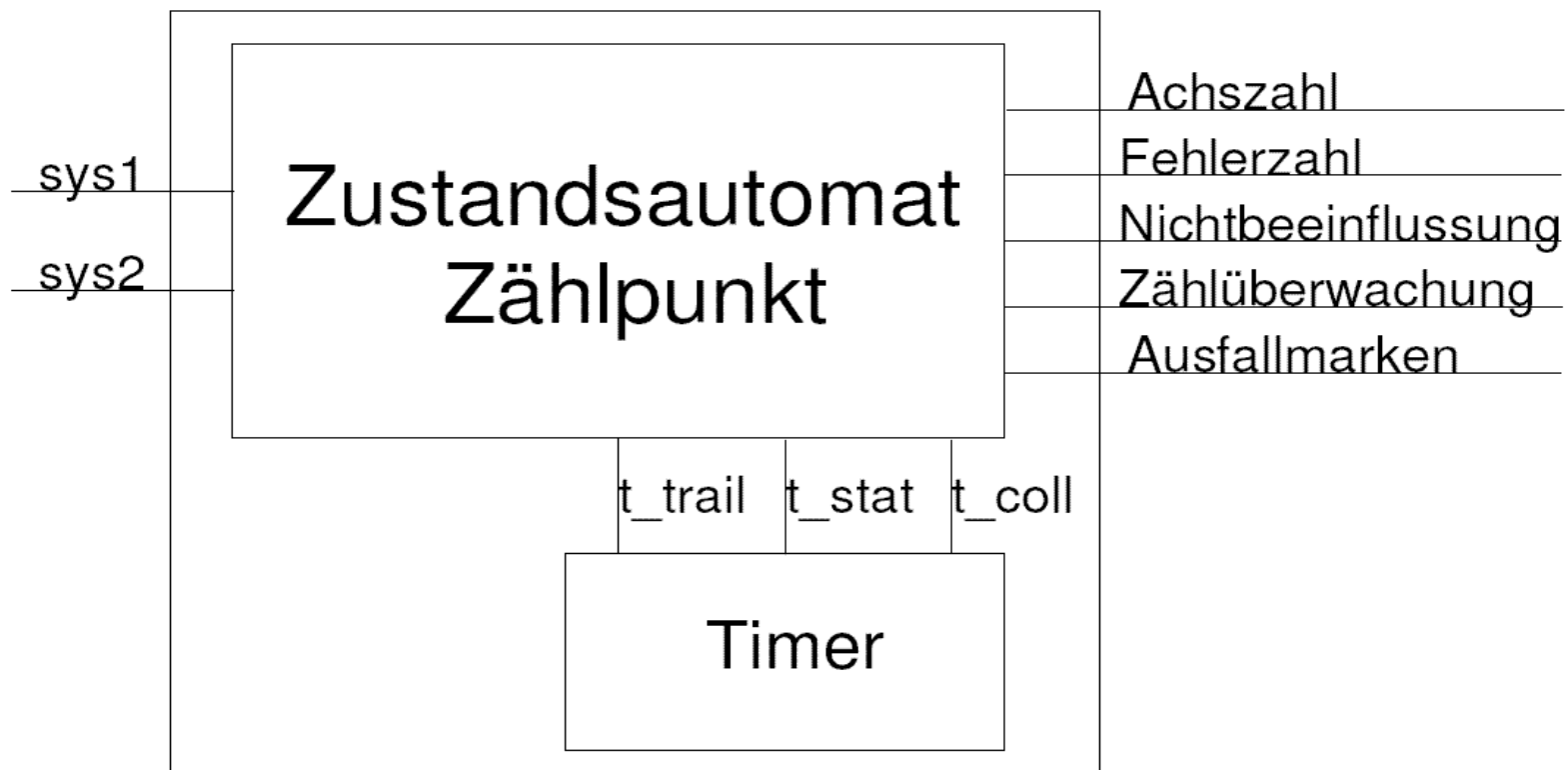


Quelle: Verfahrensbeschreibung MAZV

# MAZV – Modellierung

- Unterteilung der MAZV-FSM
  - Modul für die FSM selbst
  - Modul für das Timing
- Eingänge: Beeinflussung der Sensoren
- Ausgänge:
  - Achszahl
  - Fehlerzahl
  - ...
- Interne Variablen
- FSM

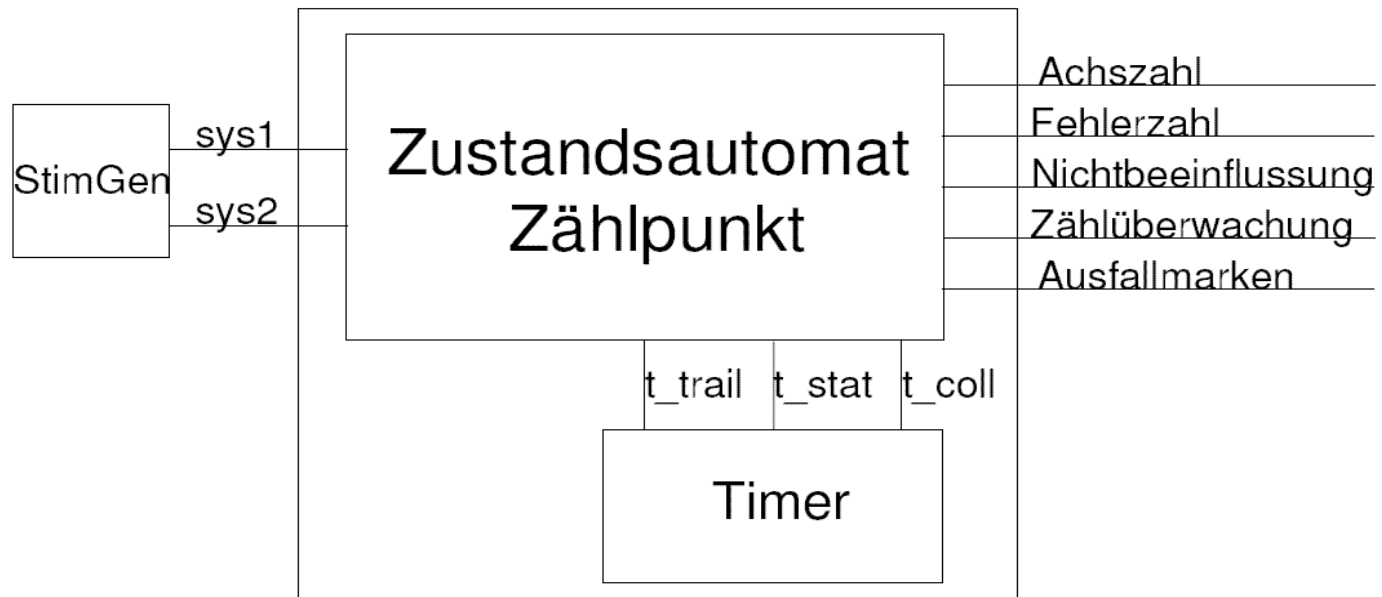
# MAZV – Modellierung



# MAZV – Validierung

Validierung:

- Manuell mit Stimuli Generator
- SystemC Verification Library (SCV)



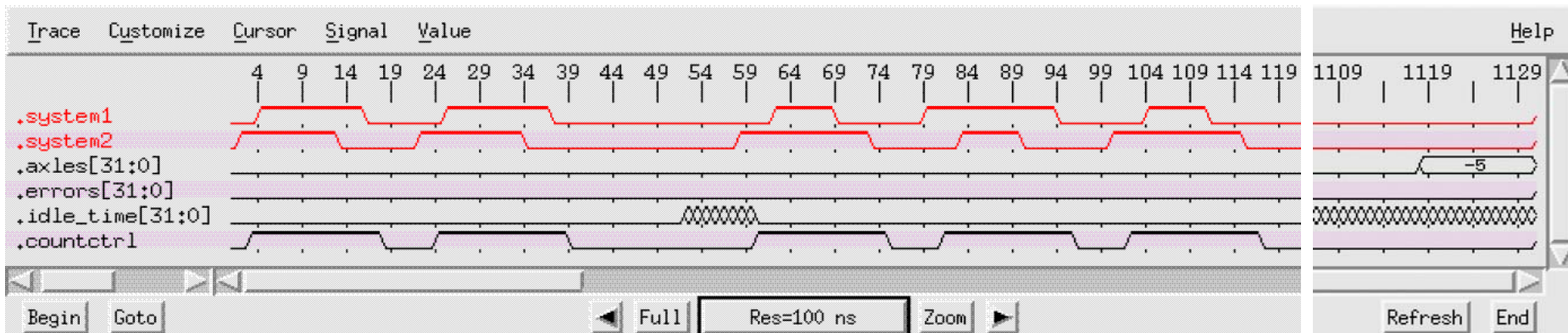
# MAZV – Validierung

Stimuli Generatoren für:

- Korrektes Befahren in/gegen Fahrtrichtung
- Fehlerhafte Befahrung
  - Im Toleranzbereich
  - Ausserhalb des Toleranzbereichs
- Andere „simulationswürdige“ Situation

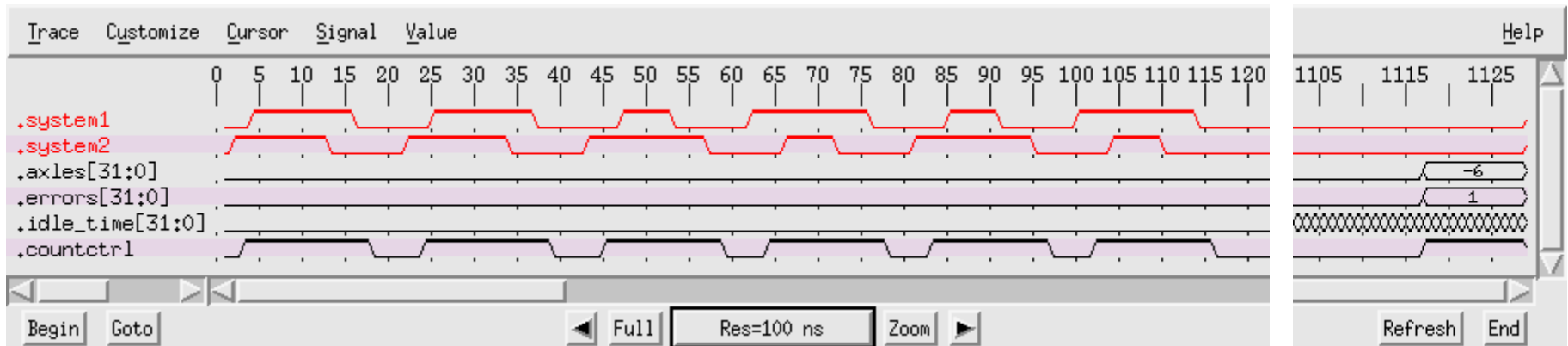
# Testbench (1)

- Viele Szenarien
- 2 Beispiele:
  - Tolerierte fehlerhafte Befahrung
  - Nicht tolerierte fehlerhafte Befahrung



# Testbench (2)

- Viele Szenarien
- 2 Beispiele:
  - Tolerierte fehlerhafte Befahrung
  - **Nicht tolerierte fehlerhafte Befahrung**



# MAZV – Validierung

Zusammenfassung:

- Simulierte Durchfahrt von 5/6 Achsen
- Simulierte Zeit jeweils ca. 1,5 s
- Simulationszeit jeweils  $< 0,03$  s
- ➔ Viele Simulationsdurchläufe in geringer Zeit → Ergebnisse schnell überprüfbar
- ➔ Zusätzliche Unterstützung durch SCV möglich

# MAZV – Verifikation

- Viele Eigenschaften notwendig für vollständige Verifikation
- ➔ Verifikation des korrekten Setzens der Zählüberwachungsmarke
  - Wird gesetzt, wenn „zu viele“ Unregelmäßigkeiten bei der Achszählung auftreten
  - Bewirkt Besetztmeldung des gesamten Abschnitts

# MAZV – Verifikation

2 unterschiedliche Ausnahmeereignisse:

- Einkanalige Beeinflussung
  - s: Zähler für einzelne Achsen
  - ss: Zähler für Achsverbund
- Zweikanalige Beeinflussung ohne Achszählung
  - c: Zähler für einzelne Achsen
  - cc: Zähler für Achsverbund

# MAZV – Verifikation

- Zählüberwachung wird ausgelöst, wenn
  - Zu viele einkanalige Beeinflussungen ( $ss > s_{max}$ ) ( $s_{max}$  ist vordefiniert)
  - oder
  - Zu viele zweikanalige Beeinflussungen ( $cc > c_{max}$ ) ( $c_{max}$  ist vordefiniert)
- ➔ Universell beweisbar
- ➔ **ABER:** Korrektheit der Zählerstände nachweisen

# MAZV – Verifikation

- Verifikation von  $ss$ 
  - $ss$  ist Funktion (Summe) von  $s$
  - Induktiver Beweis von  $s$ 
    - $s$  enthält korrekten Wert im Initialzustand
    - $s$  enthält korrekten Wert im Induktionsschritt
  - Beweis  $ss$  ist Summe aller  $s$  (gilt universell)  
Aber Einschränkung der Zählerstände zur Vermeidung von „false negatives“
- Verifikation von  $cc$  ist ähnlich

# MAZV – Verifikation

	Initial- zustand	Induktions- schritt	Universell
s	3.16 s	3.86 s	-
SS	-	-	3.28 s
c	3.22 s	3.39 s	-
CC	-	-	2.97 s
Zählüber- wachung	-	-	2.65 s

- Gesamtzeit: 22,53 s (AMD Athlon 64 3500+)
- Meiste Zeit für Abrollen und Erzeugen der SAT Instanz
- Nie mehr als 110 MB Speicher notwendig

# Zusammenfassung & Ausblick

- Zusammenfassung
  - SystemC-basierter Design-/Verifikations-flow
  - Modellierung, Validierung und Verifikation eine Achszählers
- Ausblick
  - Vollständiger Eigenschaftssatz
  - Nachweis der Vollständigkeit