

# Zeit ist Geld – Echtzeit ist viel Geld

## Systemoptimierungen am Beispiel der BMW-Aktivlenkung



**Hans Sarnowski**  
EF-611

**SymTA/S NewsConference “ Braunschweig**  
18.09.2007

**BMW Group**



# Freude am Lenken

## Agenda

- 1. Einleitung**
- 2. Systemübersicht Aktivlenkung**
- 3. Einsatz von Timingtools bei der Aktivlenkung**
- 4. Bewertung der Ergebnisse**

# Einleitung.

## Entwicklungsziele.

- Konventionelle Lenksysteme sind ausgereizt; das Potential durch Variation der Handmomente (Servotronic) ist weitgehend erschöpft.
- Fahrleistungen steigen kontinuierlich; der Zielkonflikt zwischen Fahr- und Geradeauslaufstabilität auf der einen sowie Handlichkeit auf der anderen Seite lässt sich mit einer festen Lenkungsauslegung nur schwer lösen.
- Die Fahrstabilisierung über radindividuelle Bremseingriffe wird zunehmend als unkomfortabel empfunden.



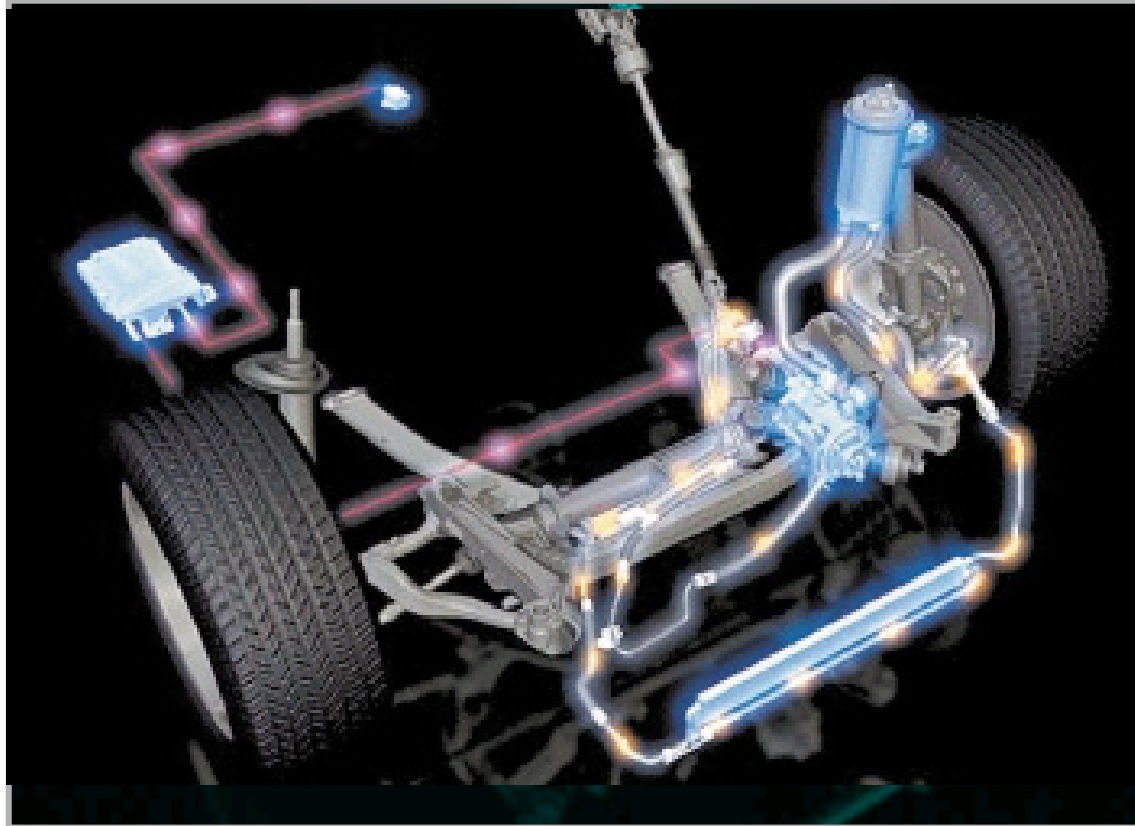
Die Überlagerungslenkung verbindet die Vorteile konventioneller Lenkungen mit den Funktionen der Steer-by-wire-Systeme.



- Reine by-wire-Systeme sind (noch?) zu komplex, zu teuer und weisen ein synthetisches Lenkgefühl auf. Ihre Akzeptanz am Markt ist fraglich.

# Systemübersicht

## Komponenten der Aktivlenkung



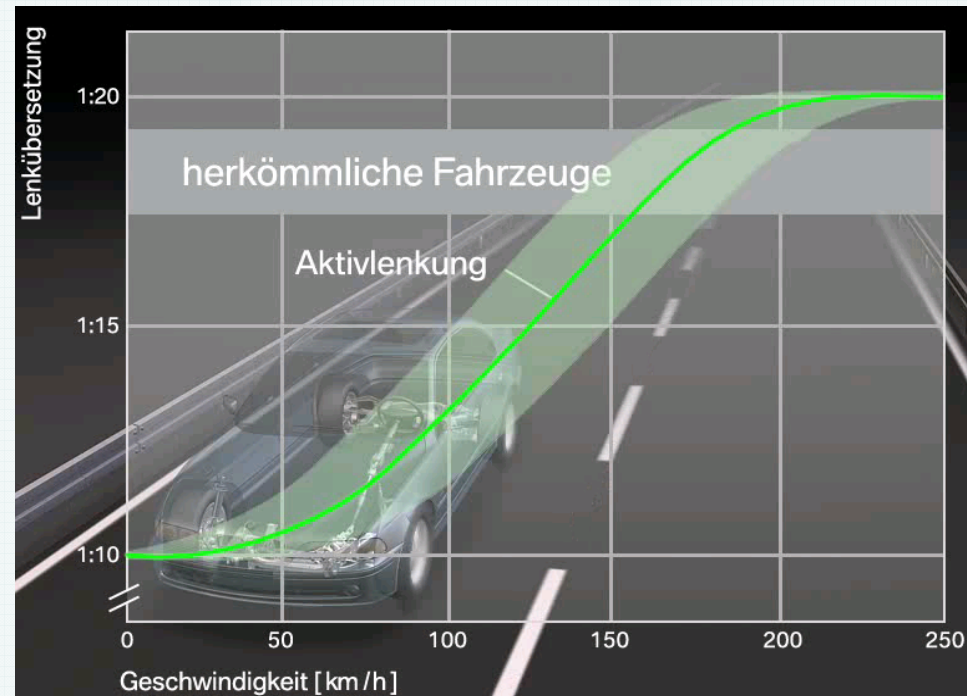
### Komponenten der Aktivlenkung:

- Zahnstangenlenkung
- E-Motor
- Überlagerungsgetriebe
- Lenksäule
- Steuergerät
- Sensorcluster
- Lenkhilfepumpe, Ölbehälter
- Schläuche, LH-Kühler

# Funktionsübersicht

## Lenkfunktionen

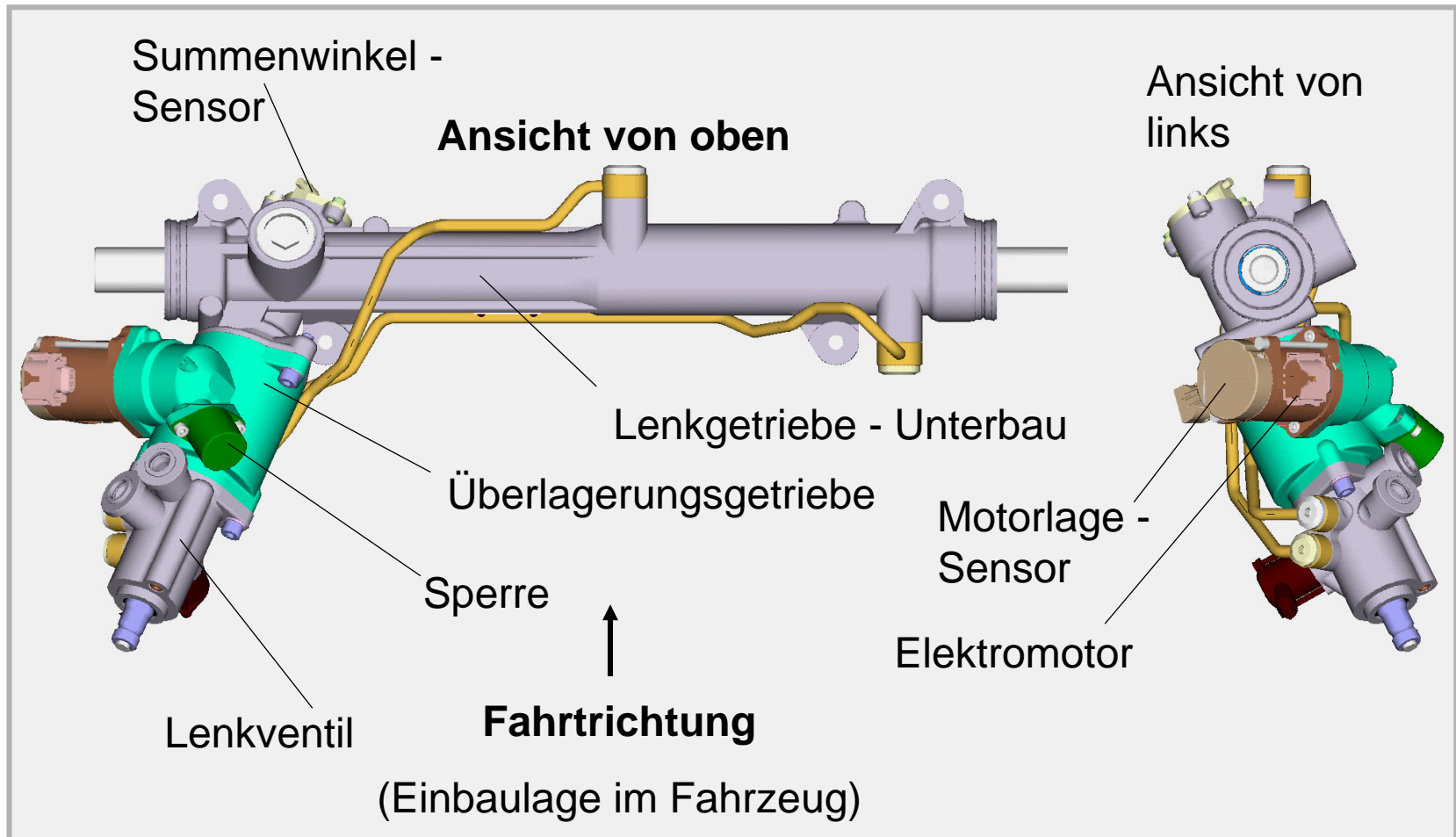
- Lenkübersetzung abhängig von Lenkradwinkel und Fahrzeuggeschwindigkeit
- Abstimmung über objektive Zielvorgaben hinsichtlich der Fahrdynamik des Gesamtfahrzeugs
- Kopplung mit Servotronik (gemeinsame Abstimmung von Lenkübersetzung und Lenkkraftniveau)



Die Überlagerungslenkung verbindet die Vorteile konventioneller Lenkungen mit den Funktionen der Steer-by-wire-Systeme.

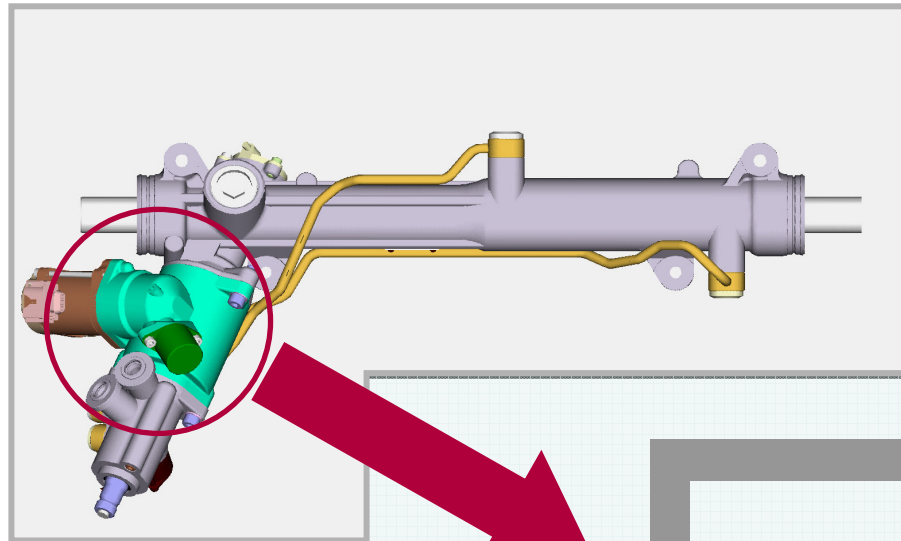
# Mechanik.

## Mechanischer Aufbau.

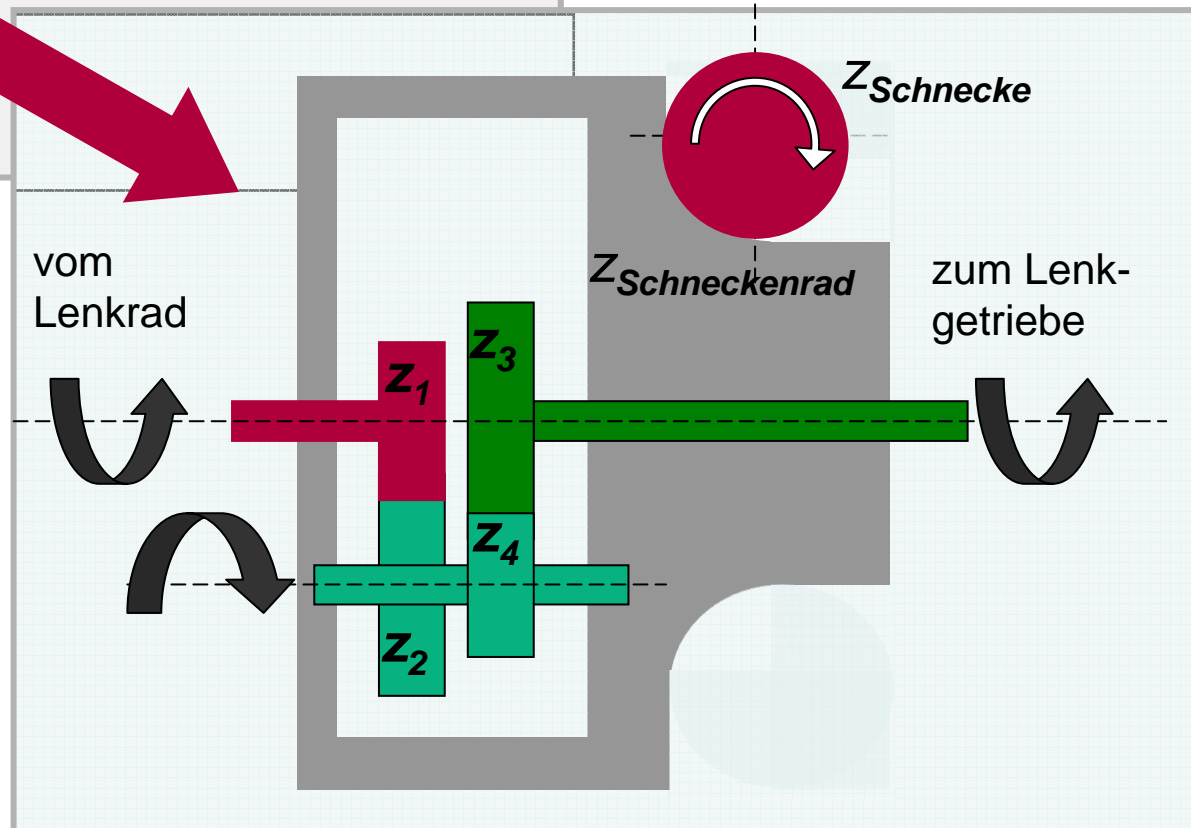


# Mechanik.

## Funktion Überlagerungsgetriebe.

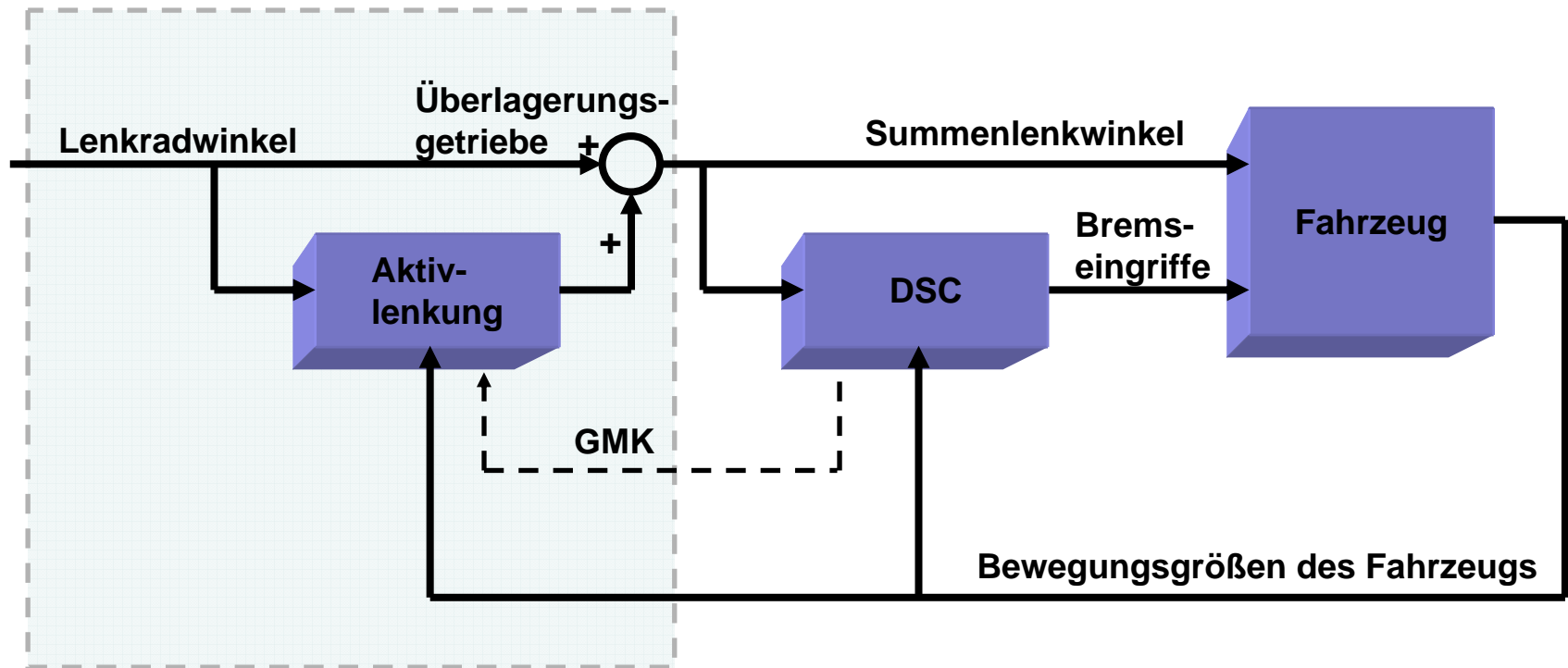


- Lenkstrang formschlüssig
- Winkeladdition
- Reales Lenkmoment
- Inhärente Rückfallebene



# Funktionsübersicht.

## Integration Aktivlenkung - DSC.



## Zielkonflikt

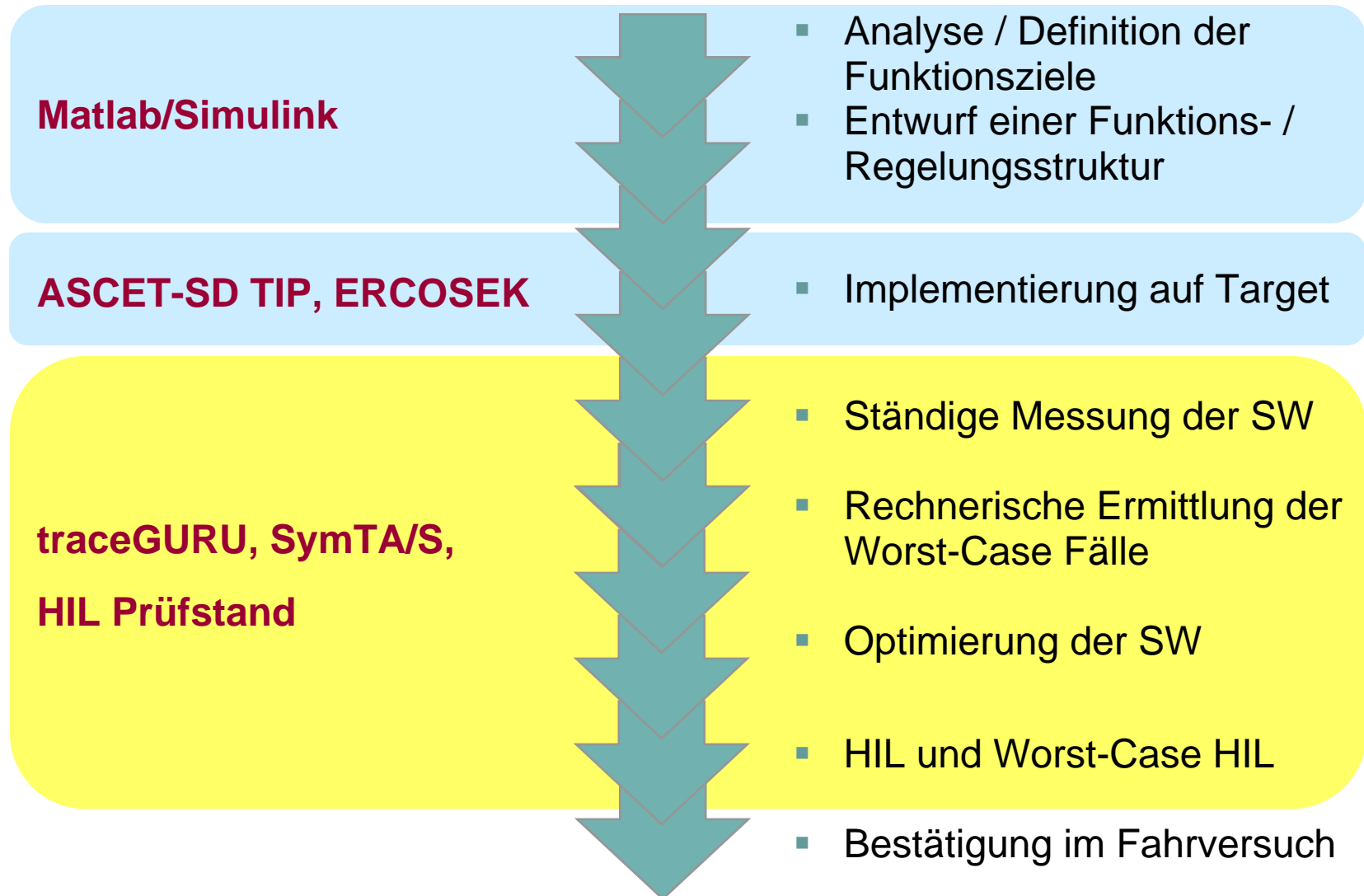
Einsatz einer deutlich günstigeren einfacheren und langsameren Prozessorhardware bei vermehrter Funktionalität

### Maßnahmen:

- Ständige Kontrolle mit Mess-und Visualisierungstools **traceGURU** (Gliwa GmbH)
- Einsatz von Scheduling Analysetool **SymTA/S** (Symtavision)
- Aufbau eines bordnetznahen Hardware in the Loop-Prüfstands mit „Worst-Case“ Konfiguration

# Entwicklungsprozess

## Toolkette



# Entwicklung

## Einsatz von Timingtools

Für die folgenden Tätigkeiten kamen Timingtools zum Einsatz

- Prozessorauswahl
- Laufzeitmessungen zu Integrationsstufen (Abgabeständen)
- Entwicklung des Timinglayout
- Entspannung der Laufzeitsituation
- Ausmessen einzelner Funktionen
- Laufzeitoptimierung
- Aufspüren von Timingproblemen
- Absicherung von Corner-Cases

# Einsatz von Timingtools

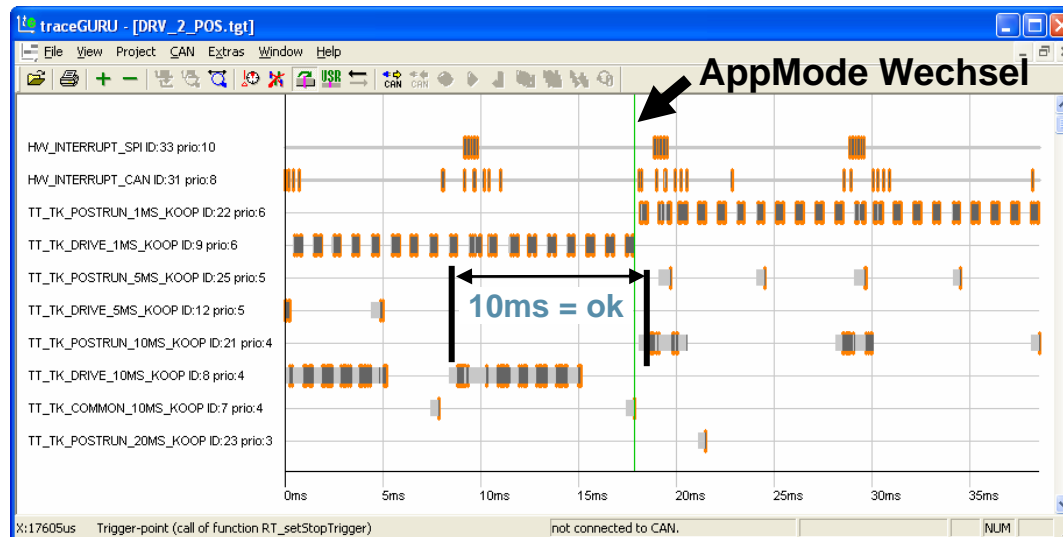
## Debugging von Timingproblemen I

Einsatz von **traceGURU** als Debugwerkzeug bei konkreten Timingproblemen

**Beispiel:** gelegentliche große Abweichung von der vorgegebenen 10 ms Periode bei CAN Botschaften

*Ursache:* Bei Applicationmode Wechseln wurden die Timer neu aufgesetzt, sodass das geforderte 10ms Raster verletzt wurde.

*Lösung:* Anpassen von Delays der periodischen Tasks, die den CAN Treiber bedienen



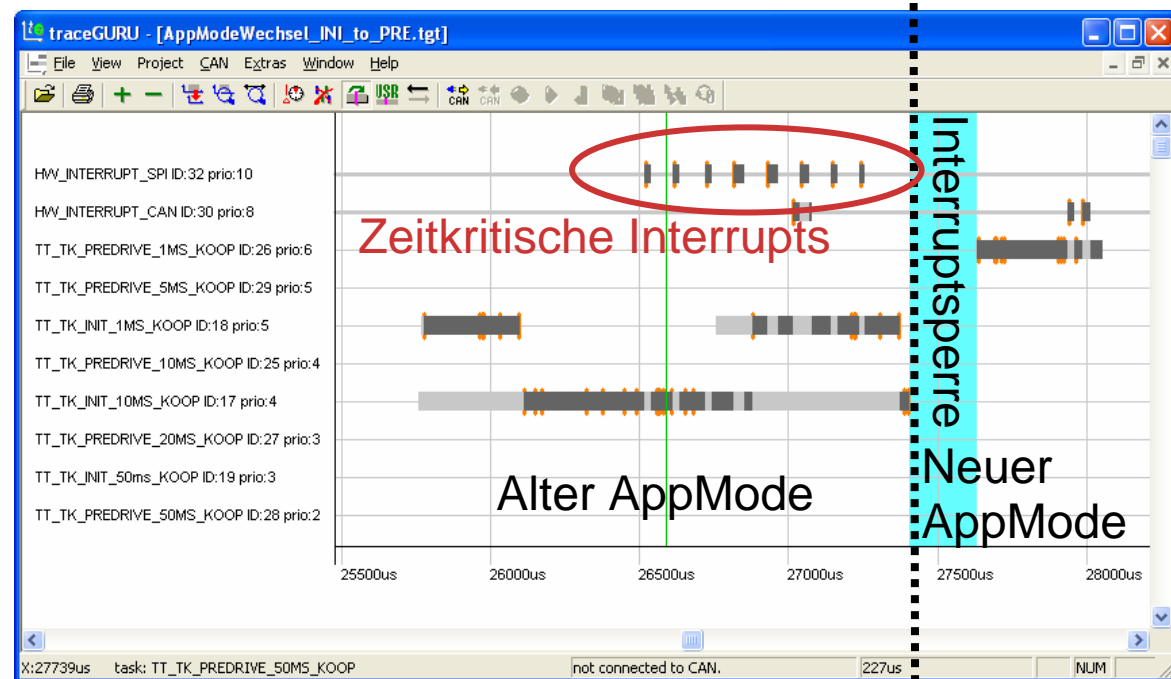
# Einsatz von Timingtools

## Debugging von Timingproblemen II

**Beispiel:** seltener Fehler bei der Berechnung des Stellwinkels

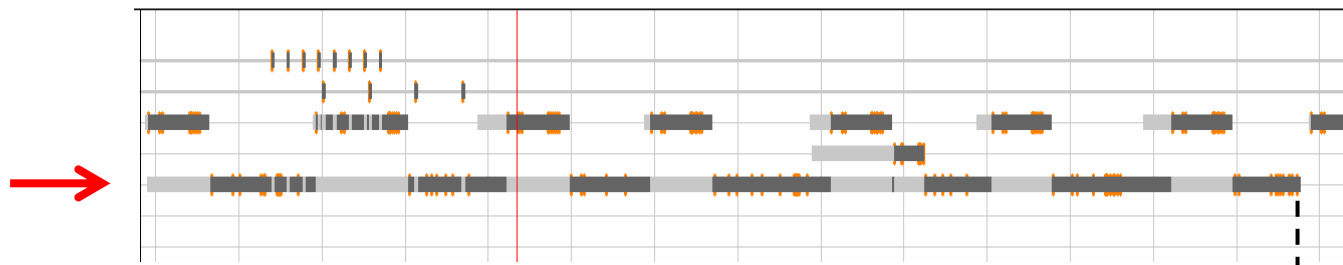
*Ursache:* Zusammenfallen einer Serie zeitkritischer Interrupts mit Applicationmode Wechsel und der damit verbundenen längeren Interruptsperre bei besonders **geringer(!)** Laufzeit der 10ms Task.

*Lösung :* Modifiziertes Timinglayout und deterministisches Umschalten der Applicationmodes, sodass Interrupts nicht mit Sperre zusammenfallen können

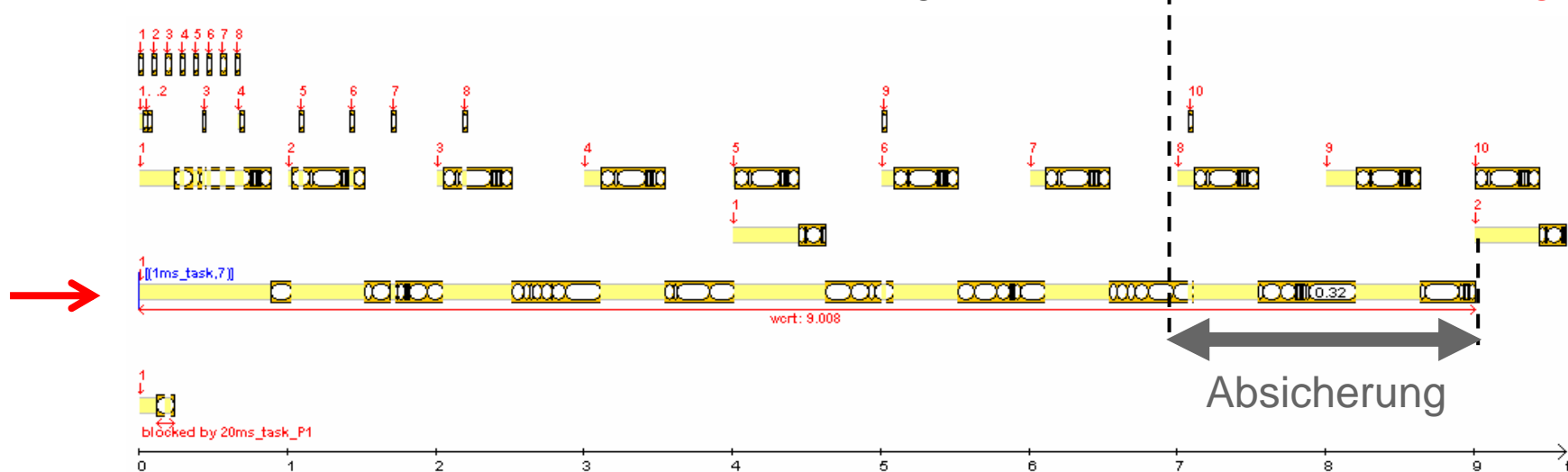


# Vergleich Tracing und Scheduling Analyse

- **Messung:** Antwortzeit **6,9ms**  
4 CAN, 8 SPI Interrupts, 7 Unterbrechungen durch 1ms Task



- **Worst-Case Analyse mit SymTA/S:** Antwortzeit **9ms**  
10 CAN, 8 SPI Interrupts, 9 Unterbrechungen durch 1ms Task, **Blockierung**





# Einsatz von Timingtools

## Prozessorauswahl

Ziel:

Kostenreduktion bei höherer Funktionalität

*Situation:* MPC Prozessorfamilie gesetzt, aber welches Derivat?

–mit internem Flash (wie beim Vorgängerprojekt) oder

–ohne internen Flash (langsamer aber günstiger).

Nach genauen Laufzeituntersuchungen (***traceGURU/delayGURU\****, ***SymTA/S***) und einer Abschätzung zukünftiger Funktionalität wurde die günstigere Variante ohne internen Flash gewählt.

Im Nachhinein gesehen die richtige Entscheidung: die Prozessorauslastung der inzwischen fertigen Seriensoftware weicht von der Vorhersage nur marginal ab.

\****delayGURU***: gezielte und zur Laufzeit (bei Bedarf automatisierbar) skalierbare Verzögerung der Anwendung

# **Anforderung an die Sicherheit**

## **Verstehen des Systems mit der Zielsetzung**

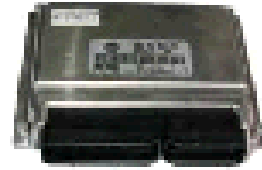
### **Vorhersagbares zeitlich stabiles Verhalten der Software**

Ähnlich dem Design der Funktionalität der Software („was passiert?“) soll das zeitliche Verhalten genau definiert sein („wann passiert es?“).

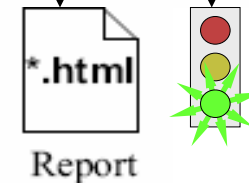
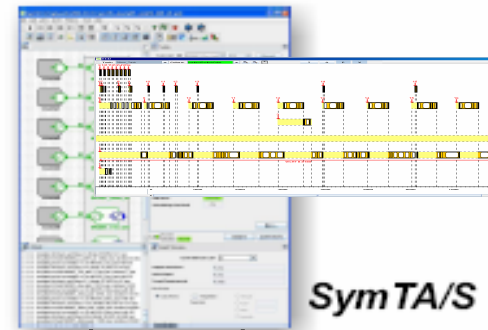
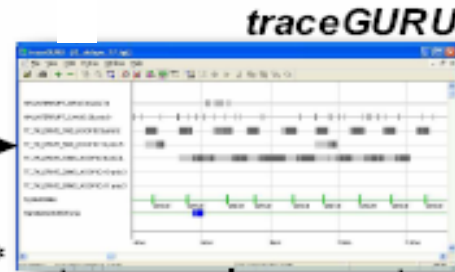
Fazit: Die grafische Darstellung der Abarbeitung von Tasks, Prozessen, Interrupts und beliebigen Codeabschnitten ermöglicht völlig neue Einblicke in die Software.

*„Man lernt sein System wirklich kennen und kann Laufzeitfehler in einem Bruchteil der Zeit aufdecken.“*

# Integration: traceGURU + SymTA/S



← CAN, Nexus,  
 Debugger,  
 oder FlexRay\* →



– Single function execution times

	A	B	C	D	E
1	Process	Task	BCET	WCET	Freq.
2	Proc_001	200ms_Task (3)	1,62	2,16	200
3	Proc_002	1000ms_Task (2)	3,08	3,16	1000
4	Proc_003	1000ms_Task (2)	2,36	2,36	1000
5	Proc_004	TSK_Dynamic (12)	2,76	2,76	500
6	Proc_005	TSK_Dynr	3,44	4,28	500
7	Proc_006	TSK_Dynr	4,13,7	623,46	500

– Interrupt Frequency

# Zusammenfassung

- **Stärken von Tracing**
  - Visualisierung führt zu schnellem Verständnis des Systems
  - Debugging spezieller, klar identifizierbarer Timingprobleme
  - Leichte Anbindung an reale ECU und reale Busse
  - Schnelles und effizientes Erfassen von Timing-Daten
  
- **Stärken von Scheduling Analyse**
  - Absicherung schwer zu konstruierender Worst-Cases
  - Analyse von End-to-End Timing
  - Schnelle Optimierung der Systemkonfiguration
  - Architekturexploration bereits in der Frühphase
  
- **Kombination ermöglicht schnelle und effektive Entwicklung.**