Modellbasierte Generierung von statischen Schedules für sicherheitskritische, eingebettete Systeme mit Multicore Prozessoren und harten Echtzeitanforderungen

J. Reinier van Kampenhout
Robert Hilbrich
Hans-Joachim Goltz

Workshop Echtzeit 2011

Fraunhofer FIRST

Inhalt

I. Einführung

– Anzahl von software-intensiven eingebetteten Systeme steigt weiter

– Beispiel Automotive:
  · Prädiktive Kollisionsvermeidung
  · Adaptive Geschwindigkeitsregelung
  · Intelligente Navigationsführung

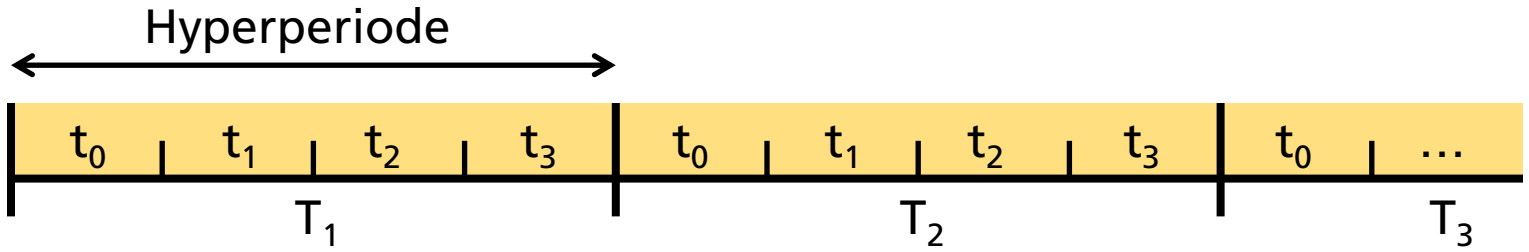
Copyright OPENSYNERGY

– Zertifizierung:
  · ISO 61508
  · ISO 26262
  · DO-178B (Luftfahrt)

– Das Verhalten von hochkomplexen Systeme muss zu jedem Zeitpunkt genau bekannt sein ➔ Scheduling vom Echtzeit-Betriebssystem

II. Erstellung von statischen Schedules

- Repetitive Ausführung von Schedule (Hyperperiode)
- Beachten von Variationsmöglichkeiten und Randbedingungen:
  - Unterschiedliche Periodenlängen
  - Abhängigkeiten
  - Zugriffe auf externe Ressourcen
- Ziele:
  - Optimale Ausnutzung der vorhanden Ressourcen
  - Korrektes Echtzeitverhalten
- Für Single-Core Prozessoren (weitgehend) manuell, aber für Multicore Prozessoren wird die Komplexität zu groß



Fraunhofer FIRST

III. Modellbasierte Generierung von statischen Schedules

PRECISION PRO:

- Generator für statische Schedules von periodischen Anwendungen auf sicherheitskritischen, eingebetteten Systemen
- Ergebnis ist *Correct by Construction*

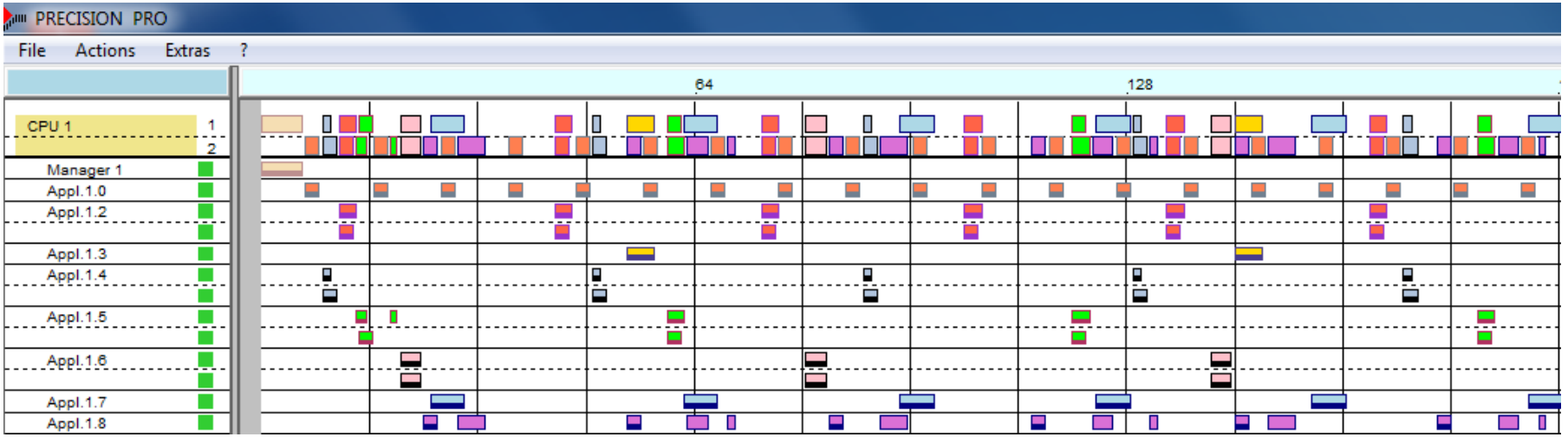
Eingaben:

- Hardware-Angebot: vereinfachtes Modell
- Software-Anforderungen: WCET, Periode, Abhängigkeiten, …
- Zuordnung von Applikationen zu Prozessoren

III. Modellbasierte Generierung von statischen Schedules

Vergleich mit andere Ansätze:

- Das Konstruieren eines korrekten Schedules, auch für sehr komplexe Problemen
- Modellierung von speziellen Ressourcen
- A priori Verhinderung von nicht-deterministische Auflösung von Ressourcenkonflikten (interferences) im Entwicklungsprozess



Fraunhofer FIRST

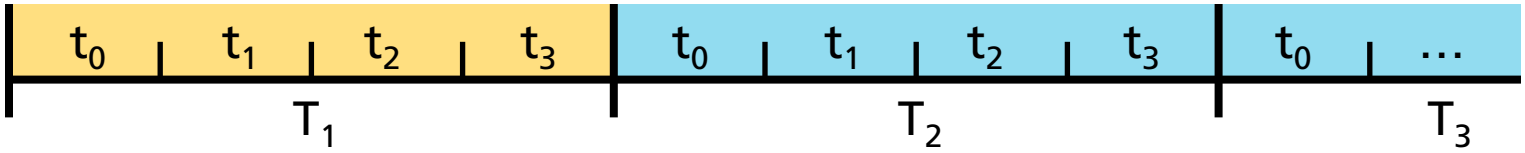
III. Modellbasierte Generierung von statischen Schedules

Vergleich mit Dynamisches Scheduling:

- Höhe Maße an Vorhersagbarkeit und Determinismus
- Nachweis zeitlicher Eigenschaften mit Scheduling-Analysewerkzeug nicht nötig
- Flexibilität im Entwicklungsprozess ist eingeschränkt
  - Kleine Änderungen im Software ändern das Schedule (radikal)
- Umgang mit Event-getriebenen Funktionen schwieriger

***Mode-basierte* statische Schedules:**

- Mehrere unterschiedliche statische Schedules
- Dynamische wechsel für Anpassung an Umgebungsbedingungen zur Laufzeit



Fraunhofer FIRST

III. Modellbasierte Generierung von statischen Schedules

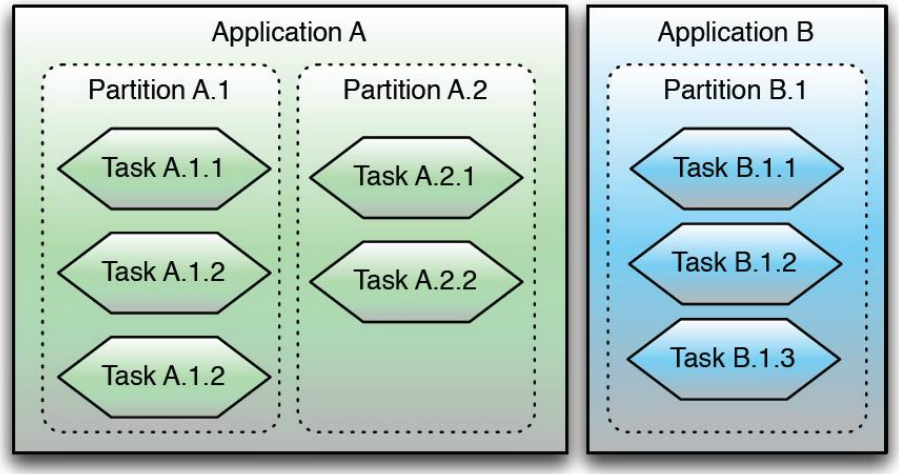
- Die strukturellen Nachteile eines statischen Schedules werden mit unserem Tool nicht beseitigt

- Die *Aufwand* für umfangreiche Schedules kann aber signifikant reduziert werden, was zu schnelleren Iterationen leitet
  - Beispiel: Konstruktion eines Schedules von 2000 ms für 30 Anwendungen kostet nur wenige Minuten auf einem aktuelle PC

- Korrektes Timing-Design und optimierte Ressourcenauslegung sind das Resultat einer korrekten *Konstruktion* unter Einbeziehung zeitlicher Anforderungen

IV. Modellierung eines Scheduling Problems

Terminologie von Precision Pro:



Software Architektur (logische Sicht)

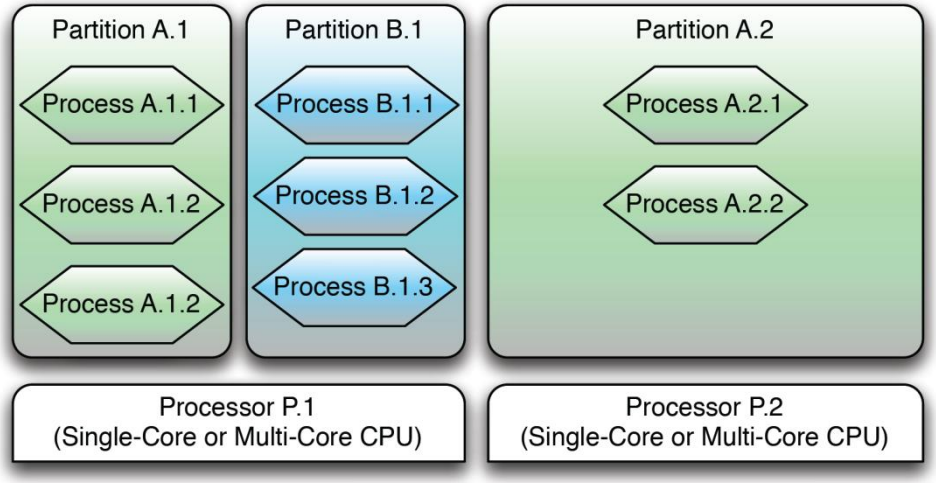


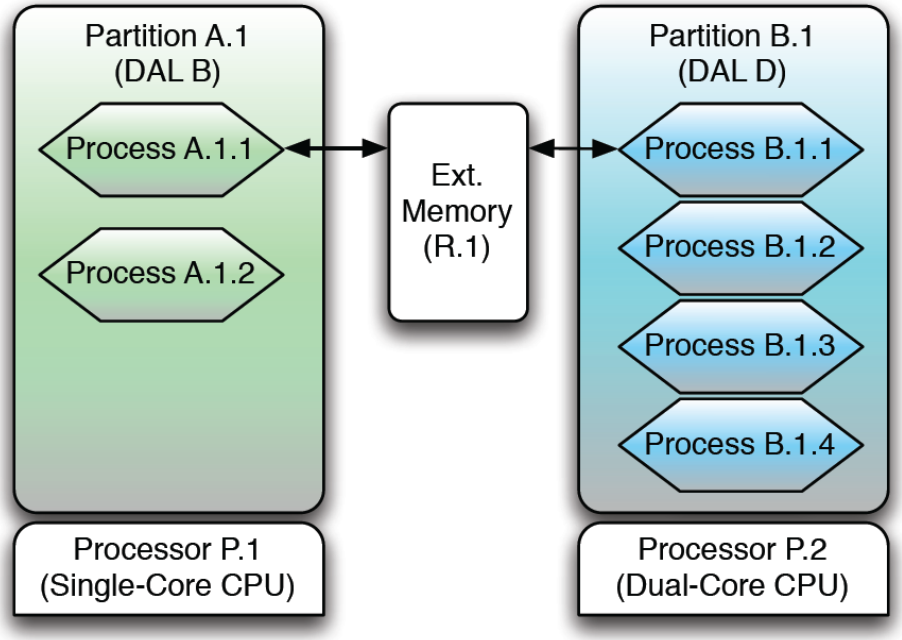
Abbildung der Software auf die Hardware

– Prozess ist grundlegende Ausführungseinheit, welche unterbrochen und in *Slices* aufgeteilt werden kann

Fraunhofer FIRST

IV. Modellierung eines Scheduling Problems

Beispiel:



- Exklusiver Zugriff auf gemeinsam genutztes Speicher erforderlich
- Anwendung A liest Sensordaten (A.1.1) und berechnet ob Aktion notwendig ist (A.1.2)
- Anwendung B holt die Daten (B.1.1) und stellt sie die Prozessen B.1.2 … B.1.4 intern zu Verfügung

IV. Modellierung eines Scheduling Problems

Weitere Echtzeitanforderungen:

- *Slicing* kostet 1 ms Verzögerung für einen Kontextwechsel
- Die Granularität (*base scheduling unit*) ist 1 ms

Prozess	WCET $[ms]$	Periode $[ms]$
A.1.1	5	20
A.1.2	10	40
B.1.1	5	20
B.1.2	11	20
B.1.3	7	40
B.1.4	3	20

- Definitionen:

```
def_global
    scheduling_period(200),
    unit_based(1:ms),
    change_delay(1).
```

IV. Modellierung eines Scheduling Problems

Hardware Architektur:

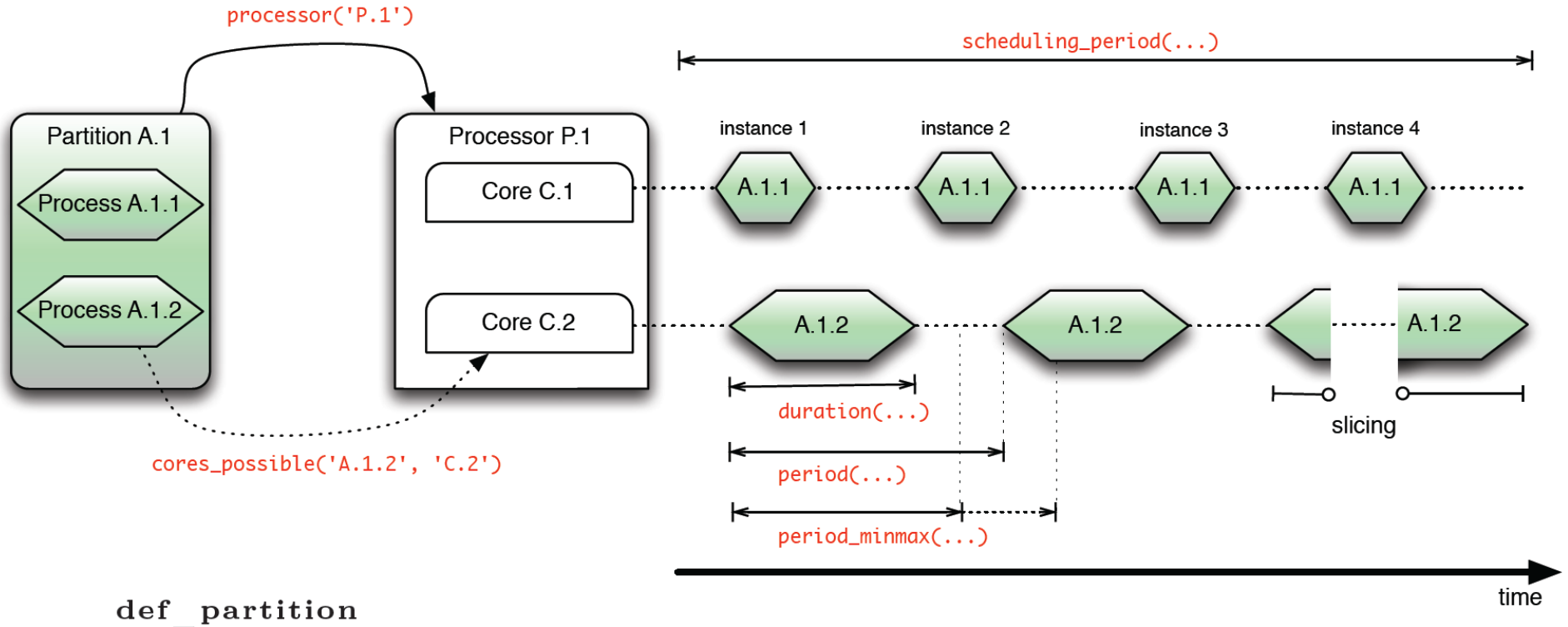
- *Cores* sind homogene, generische Ausführungseinheiten
- Aspekte wie Pipelining, Speculative Execution und der Einfluss der Speicherhierarchie sind im WCET enthalten
- Angenommen ist ein „Uniform Memory Architecture“
- Modellierung der Abhängigkeiten zwischen mehrere (Multicore) Prozessoren

```
def_processor
    id('P.1'),
    cores(1).

def_processor
    id('P.2'),
    cores(2).
```

IV. Modellierung eines Scheduling Problems

Anwendungen und ihre zeitlichen Eigenschaften:




```
def_partition
    id('B.1'),
    parallel(4),
    duration(1, 5), duration(2, 11), duration(3, 7), duration(4, 3),
    period(20), period(3, 40).
```


Fraunhofer FIRST

IV. Modellierung eines Scheduling Problems

Externe Ressourcen:

- Exklusiver zugriff (Kapazität 1)
- Kumulativer zugriff (Kapazität n)
  - Z.b. Netzwerkbandbreiten

```
def_resource
    id('R.1'),
    max(1).

def_partition
   id('A.1'),
   resource(1, 'R.1', 1),
   ...

def_partition
    id('B.1'),
    resource(1, 'R.1', 1),
    ...
```

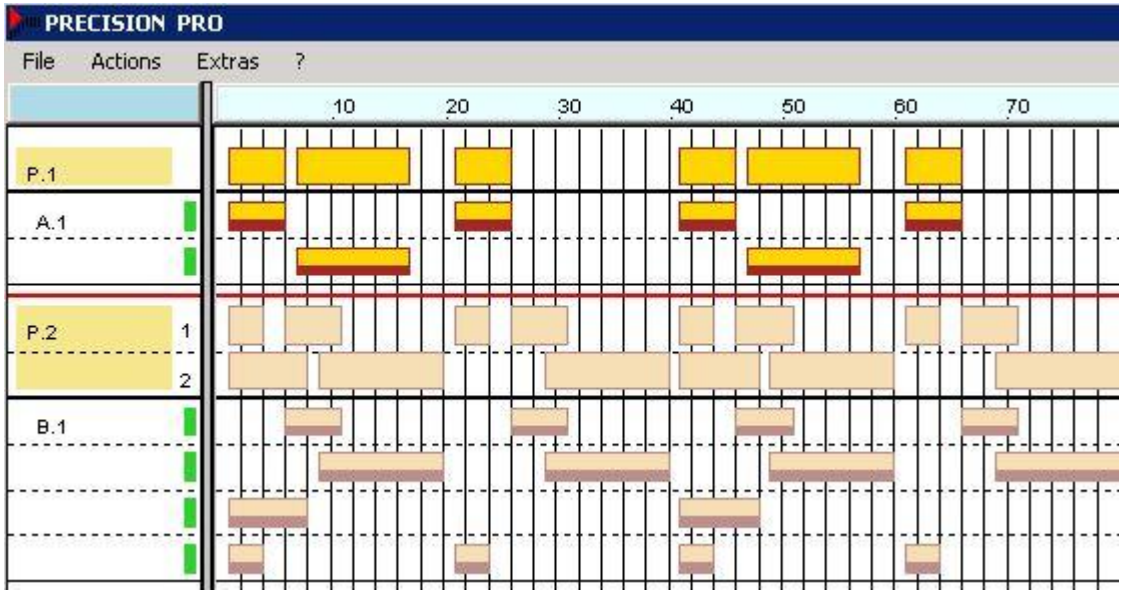
Zuordnung der Software auf der Hardware:

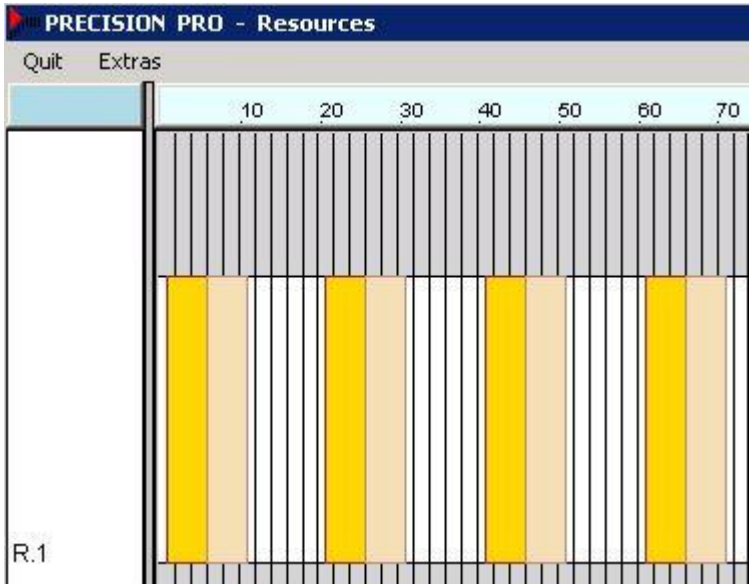
- Manuell, wegen nicht-funktionale Aspekte
- Aktuelle Arbeit: automatische Verteilung

IV. Modellierung eines Scheduling Problems

Ergebnis:

– Wenn ein Schedule existiert für die spezifizierten Anforderungen, findet PRECISION PRO dieses in kurzer zeit ➔ ~1 Sekunde für das Beispiel



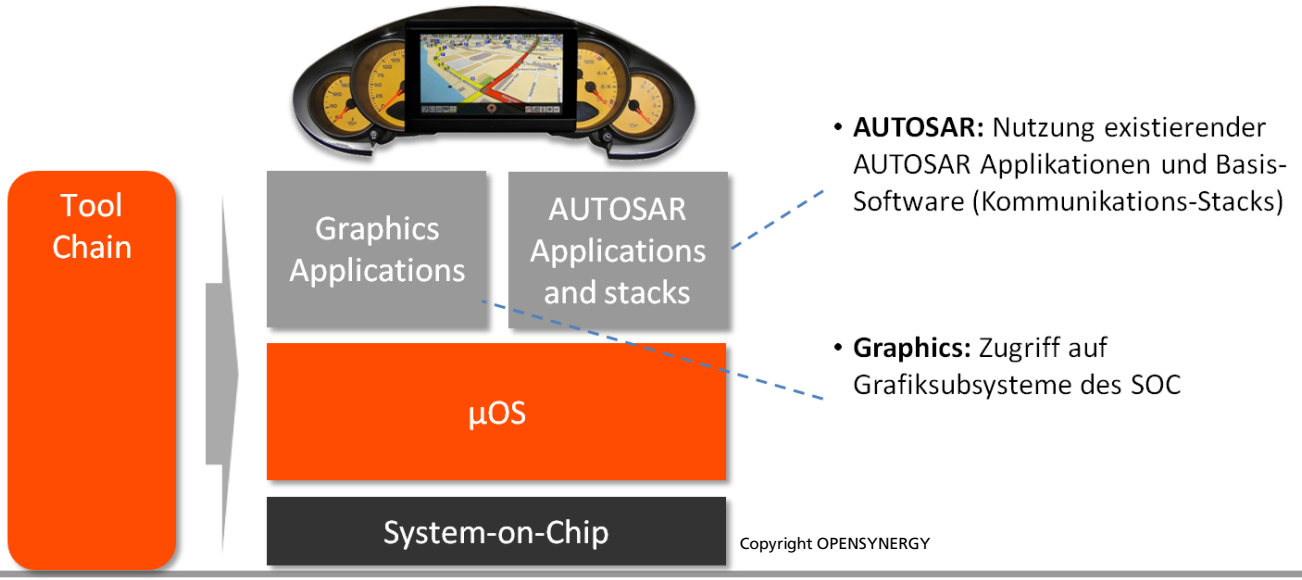


Fraunhofer FIRST

V. Aktuelle Arbeiten: Hierarchischen Schedules

VirtuOS: Virtualisierung im Bereich Automotive

- Fallbeispiel: Instrument-Cluster zur Integration von Infotainment und sicherheitskritischen Automotive Funktionen
- Mikrokernel isoliert die Partitionen



Fraunhofer FIRST

V. Aktuelle Arbeiten: Hierarchischen Schedules

Zweistufige Scheduling-Hierarchie:

- Nicht-sicherheitskritische Infotainment Partition mit dynamischem Scheduling

- Sicherheitskritische AUTOSAR Partition mit Prozesse, die harte Echtzeitbedingungen unterliegen

- Aktuelle Arbeit: Erstellung von statischen Schedules, die:
  - Eine feste Aufteilung der Systemressourcen sicherstellen
  - Deadlines von Prozessen innerhalb einer Partition berücksichtigen
  - Die Umschaltzeiten minimieren

Fazit

- Automatisierten, modellbasierte Generierung von statischen Schedules
  - *Correct by construction*
  - Determinismus und vorhersagbarkeit
  - Wenig Flexibilität zur Laufzeit ➔ Mode-basierten Schedules
- Frühe Bewertung von Hardware- und Softwarearchitekturen
- Optimierung der Systemauslastung
- Modellieren von Speziellen Ressourcen
- Ansatz zur Benützung von Multicore Prozessoren in sicherheitskritischen Echtzeitsysteme

Ende

Danke für Ihre Aufmerksamkeit!

Unsere nächste Veranstaltung:
Multicore-Workshop 2011: „Multicore-Prozessoren in Echtzeitsystemen“
24. November 2011, Fraunhofer FIRST, Berlin
www.first.fraunhofer.de

Fraunhofer FIRST