ISE SRA

Leibniz Universität Hannover

Interaction-Aware Analysis and Optimization of Real-Time Application and Operating System

Christian Dietrich

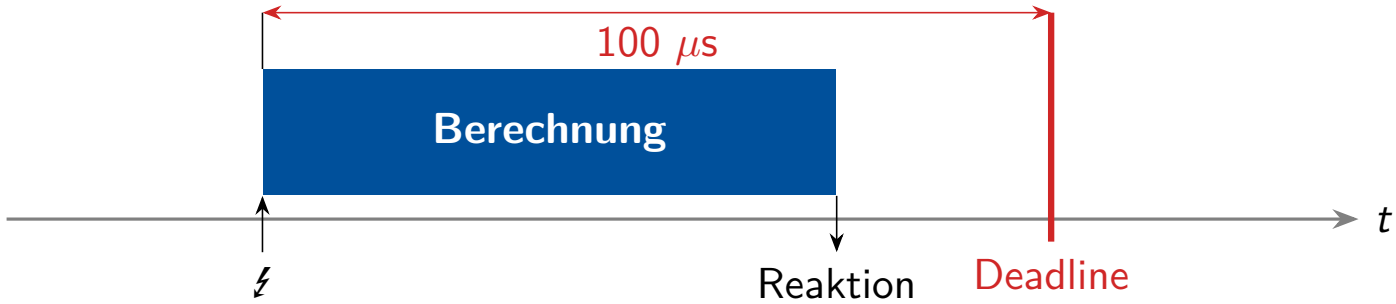
12. März 2021

Wir sind umgeben von eingebetteten Systemen

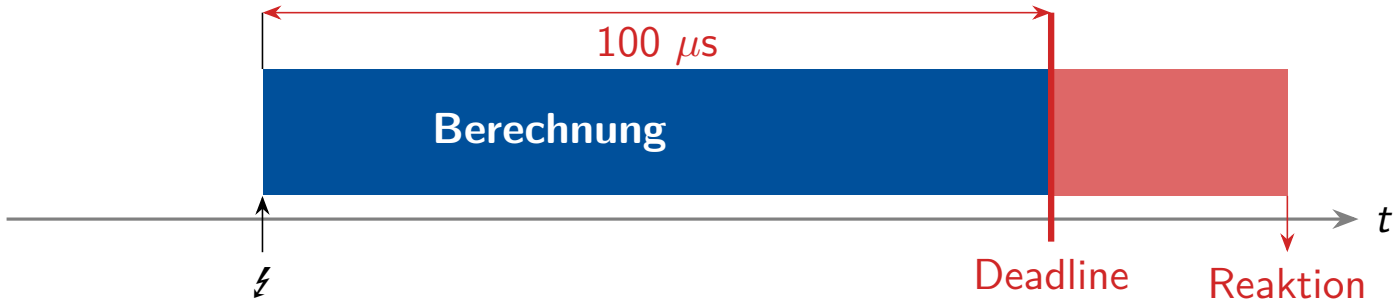
Wir sind umgeben von eingebetteten Systemen

98% aller Prozessoren

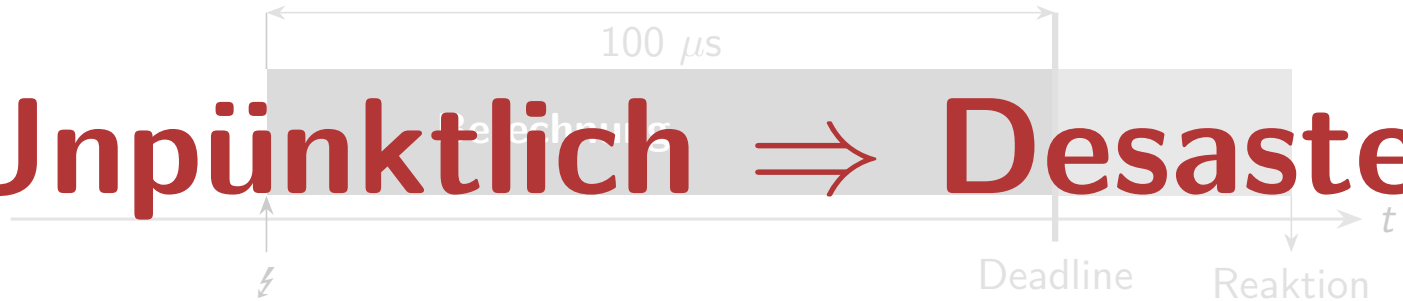
Wir sind umgeben von eingebetteten Systemen



Wir sind umgeben von eingebetteten Systemen



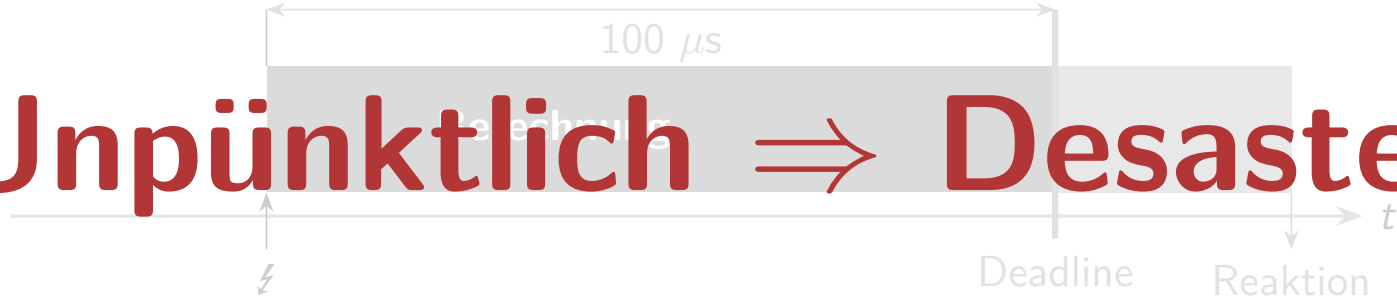
Wir sind umgeben von eingebetteten Systemen



Unpünktlich ⇒ Desaster

Wir sind umgeben von eingebetteten Systemen



Unpünktlich $\Rightarrow$ Desaster

Wir sind umgeben von eingebetteten Systemen





100 μs

Unpünktlich $\Rightarrow$ Desaster

t

Deadline Reaktion

$-$0,01 € $\Rightarrow$ $+$110 000 €

...

Wir sind umgeben von eingebetteten Systemen



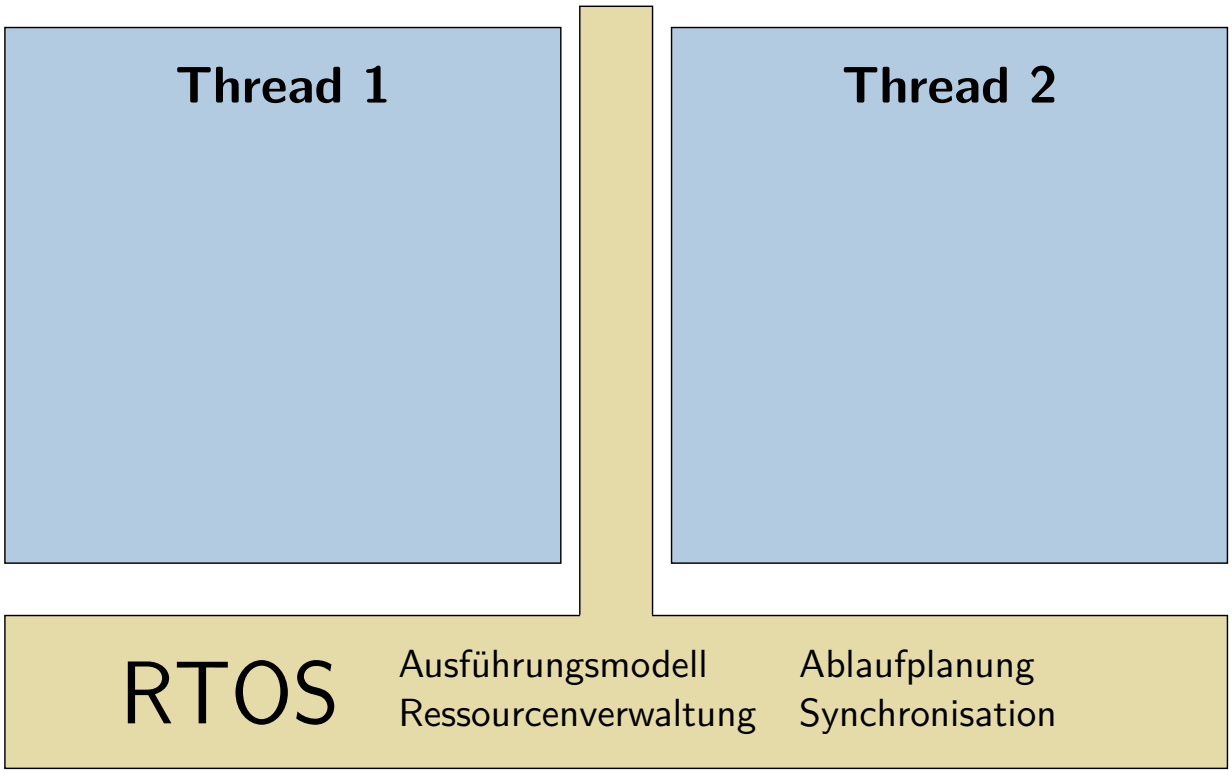
100 μs

Zwei zentrale Aufgaben für den Ingenieur:

- Nachweis der geforderten Funktionalität
- Minimierung des Ressourcenverbrauchs

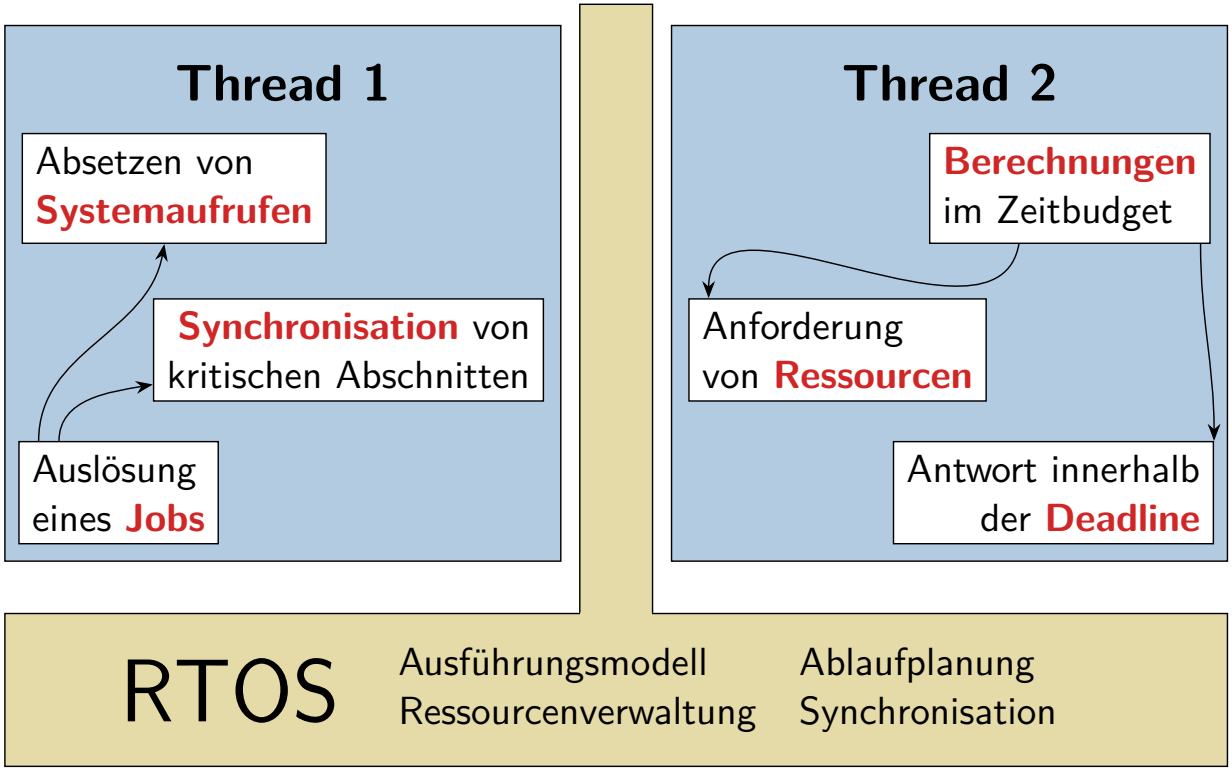
Betriebssystem: Vermittler und Organisator

- Ein Betriebssystem (RTOS) erleichtert die Anwendungsentwicklung.



Betriebssystem: Vermittler und Organisator

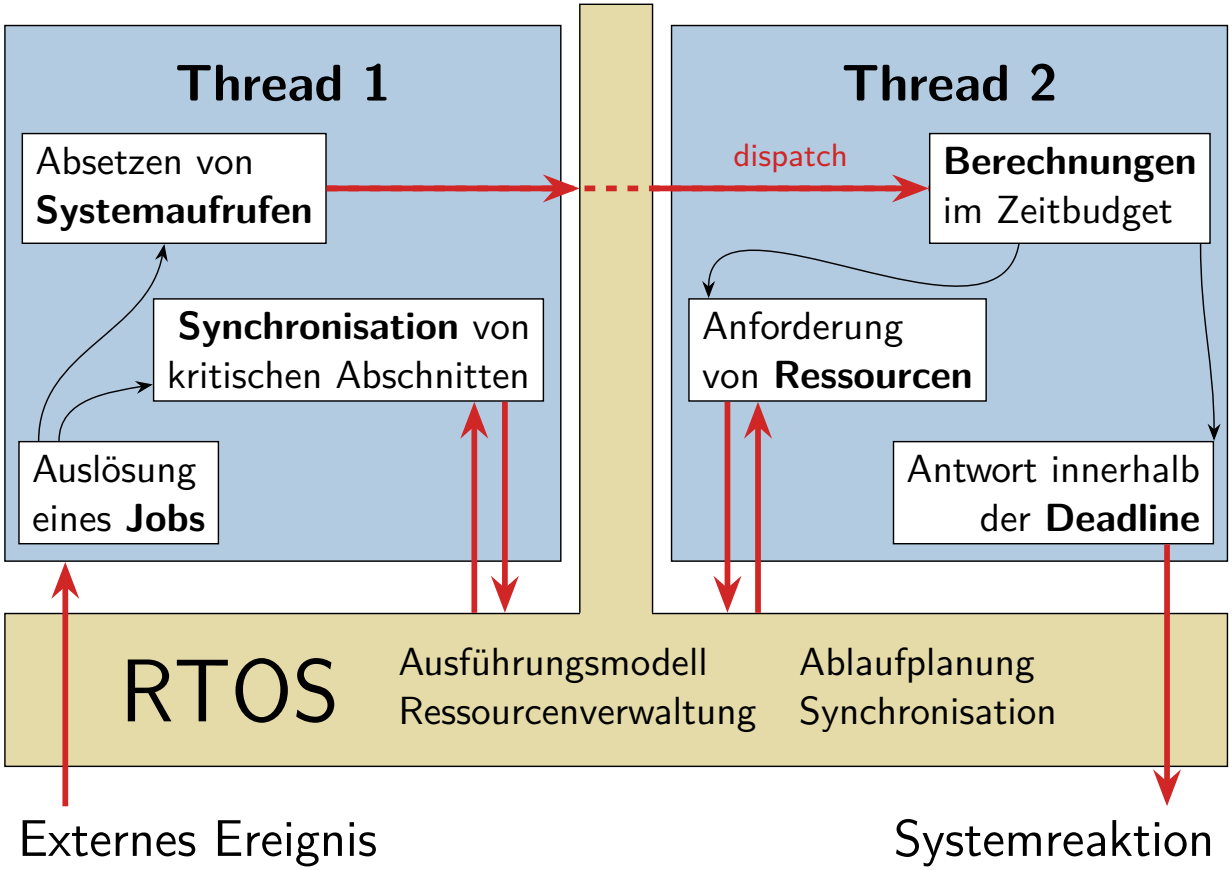
- Ein Betriebssystem (RTOS) erleichtert die Anwendungsentwicklung.
- Die Anwendung kommt mit Forderungen, Abläufen und Aktionen.



Betriebssystem: Vermittler und Organisator

- Ein Betriebssystem (RTOS) erleichtert die Anwendungsentwicklung.
- Die Anwendung kommt mit Forderungen, Abläufen und Aktionen, ...

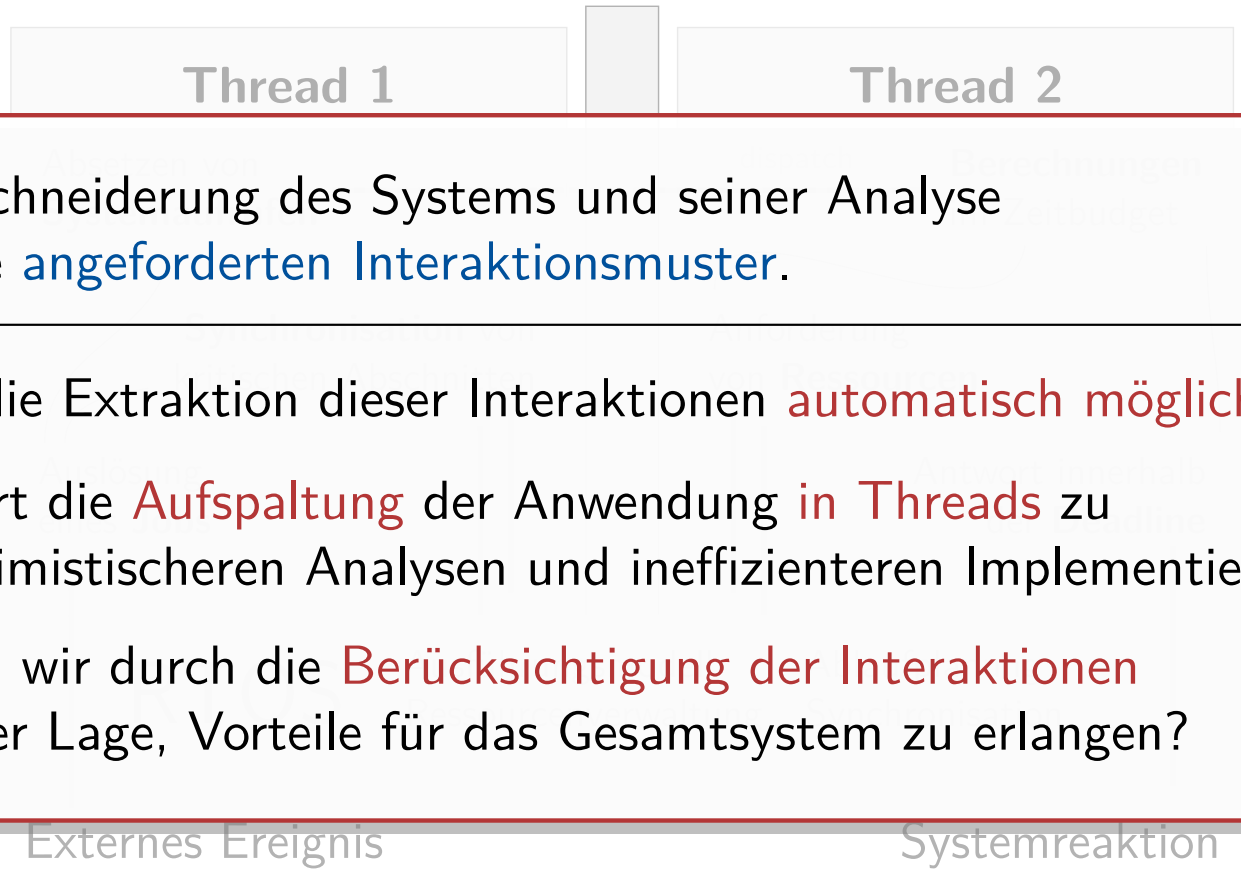
...die sich in **Interaktionen** mit dem RTOS manifestieren.



Betriebssystem: Vermittler und Organisator

- Ein Betriebssystem (RTOS) erleichtert die Anwendungsentwicklung.
- Die Anwendung kommt mit Forderungen, Abläufen und Aktionen, ...

...die sich in **Interaktionen** mit dem RTOS manifestieren.



Vision: Maßschneiderung des Systems und seiner Analyse an die angeforderten Interaktionsmuster.

Frage 1: Ist die Extraktion dieser Interaktionen automatisch möglich?

Frage 2: Führt die Aufspaltung der Anwendung in Threads zu pessimistischeren Analysen und ineffizienteren Implementierungen?

Frage 3: Sind wir durch die Berücksichtigung der Interaktionen in der Lage, Vorteile für das Gesamtsystem zu erlangen?

Übersicht

Grundlagen und Vorarbeiten *(LCTES'15, TECS'17)*

- Feingranulare Interaktionsanalyse
- Systemzustände und Globaler Kontrollflussgraph

Interaktionsgewahre Systemanalyse

- SysWCET: Flusssensitive Antwortzeitanalyse *(RTAS'17, ECRTS'18)*
- Automatische Verifikation des Kernverhaltens *(FMCAD'17)*

Interaktionsgewahre Systemoptimierung

- Semi-Extended Tasks: Stack als geteilte Ressource *(RTSS'18)*
- OSEK-V: RTOS als Prozessorerweiterung *(OSPERT'15, LCTES'17)*

Übersicht

Grundlagen und Vorarbeiten *(LCTES'15, TECS'17)*

- Feingranulare Interaktionsanalyse
- Systemzustände und Globaler Kontrollflussgraph

Interaktionsgewahre Systemanalyse

- SysWCET: Flusssensitive Antwortzeitanalyse *(RTAS'17, ECRTS'18)*
- Automatische Verifikation des Kernverhaltens *(FMCAD'17)*

Interaktionsgewahre Systemoptimierung

- Semi-Extended Tasks: Stack als geteilte Ressource *(RTSS'18)*
- OSEK-V: RTOS als Prozessorerweiterung *(OSPERT'15, LCTES'17)*

Systemmodell für meine Betrachtungen

- Ereignisgesteuerte Echtzeitbetriebssysteme als Ausführungsmodell
  - Aktivitäten: Threads und benutzerdefinierte Interrupt-Service-Routinen
  - Fixed-Priority-Scheduling: höchstpriorer Thread läuft bis zur Vollendung
  - Synchronisation und Benachrichtigungen zwischen Aktivitäten

  $\Rightarrow$ Relevant für die Industrie: OSEK/AUTOSAR, FreeRTOS, eCos, ...

Systemmodell für meine Betrachtungen

- Ereignisgesteuerte Echtzeitbetriebssysteme als Ausführungsmodell
  - Aktivitäten: Threads und benutzerdefinierte Interrupt-Service-Routinen
  - Fixed-Priority-Scheduling: höchstpriorer Thread läuft bis zur Vollendung
  - Synchronisation und Benachrichtigungen zwischen Aktivitäten

  $\Rightarrow$ Relevant für die Industrie: OSEK/AUTOSAR, FreeRTOS, eCos, ...

- Statisches System als Anwendungsmodell
  - Definition aller Aktivitäten und anderer Systemobjekte vor der Laufzeit
  - Statische Anwendungsstruktur und Interaktionspunkte

  $\Rightarrow$ Bereits notwendig für strikte Echtzeitanalyse

Systemmodell für meine Betrachtungen

- Ereignisgesteuerte Echtzeitbetriebssysteme als Ausführungsmodell
  - Aktivitäten: Threads und benutzerdefinierte Interrupt-Service-Routinen
  - Fixed-Priority-Scheduling: höchstpriorer Thread läuft bis zur Vollendung
  - Synchronisation und Benachrichtigungen zwischen Aktivitäten

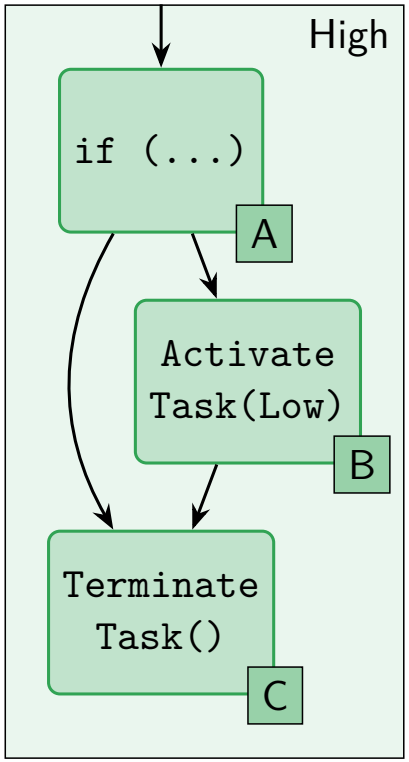
  $\Rightarrow$ Relevant für die Industrie: OSEK/AUTOSAR, FreeRTOS, eCos, ...

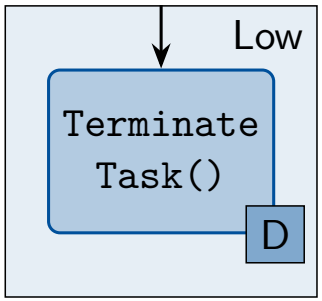
- Statisches System als Anwendungsmodell
  - Definition aller Aktivitäten und anderer Systemobjekte vor der Laufzeit
  - Statische Anwendungsstruktur und Interaktionspunkte

  $\Rightarrow$ Bereits notwendig für strikte Echtzeitanalyse

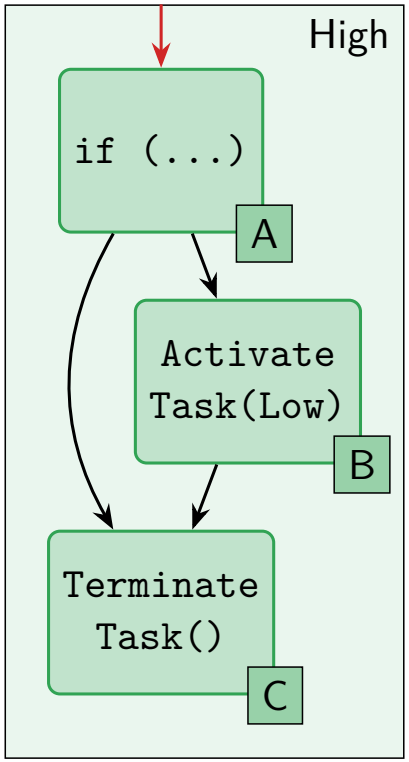
- OSEK/AUTOSAR als representativer und konkreter RTOS-Standard
  - Standardisierung durch die Automobilindustrie (1994–heute)
  - Statische Konfiguration des RTOS zur Übersetzungszeit
  - Einkern- und partitionierte Mehrkernsysteme

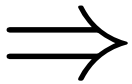
Feingranulare Interaktionsanalyse



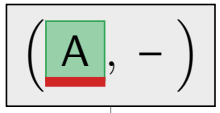


Feingranulare Interaktionsanalyse

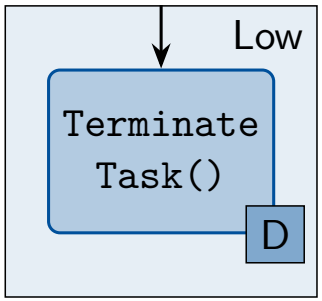




Systemzustandsgraph (SSTG)



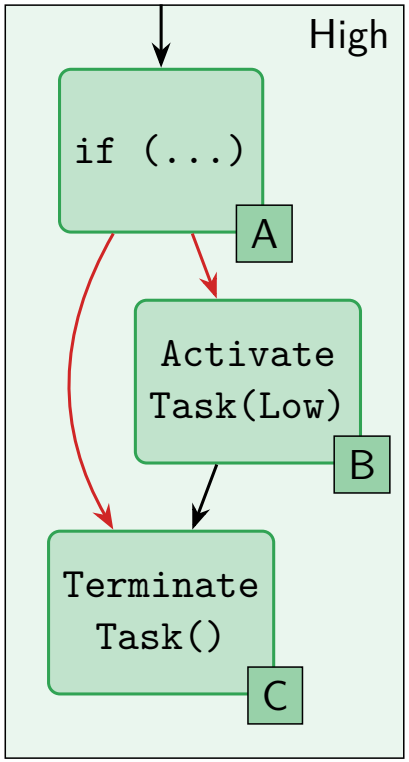
Liste laufähiger Threads

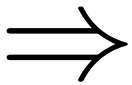


1. Statische Aufzählung der Systemzustände

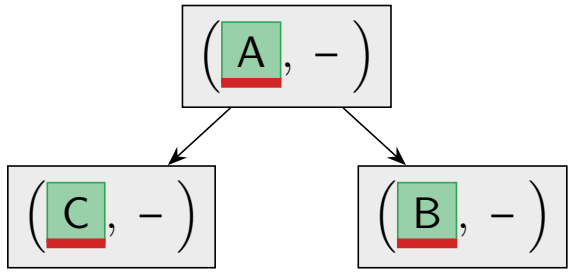
- Systemkonfiguration: Anwendung und RTOS

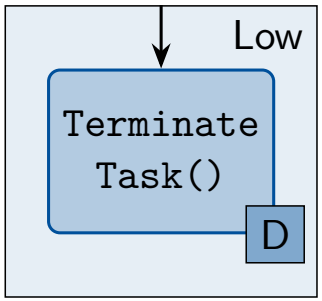
Feingranulare Interaktionsanalyse





Systemzustandsgraph (SSTG)

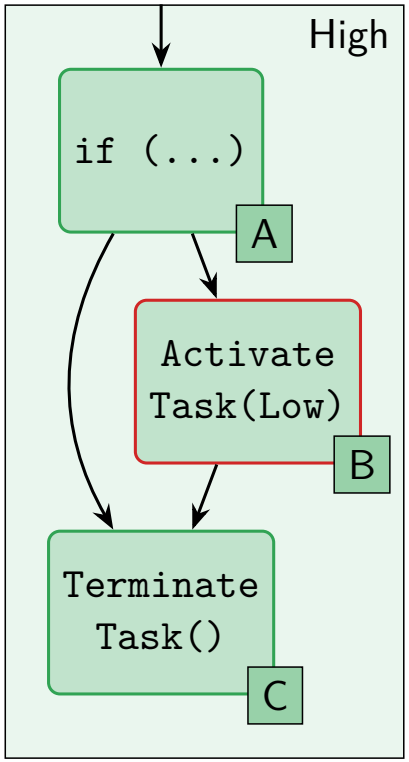


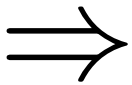


1. Statische Aufzählung der Systemzustände

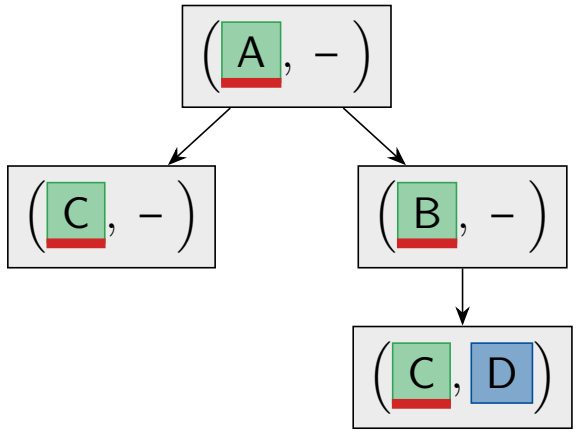
- Systemkonfiguration: Anwendung und RTOS
- Spaltung des Zustandsraums bei Bedingungen

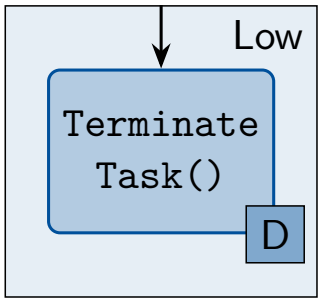
Feingranulare Interaktionsanalyse





Systemzustandsgraph (SSTG)

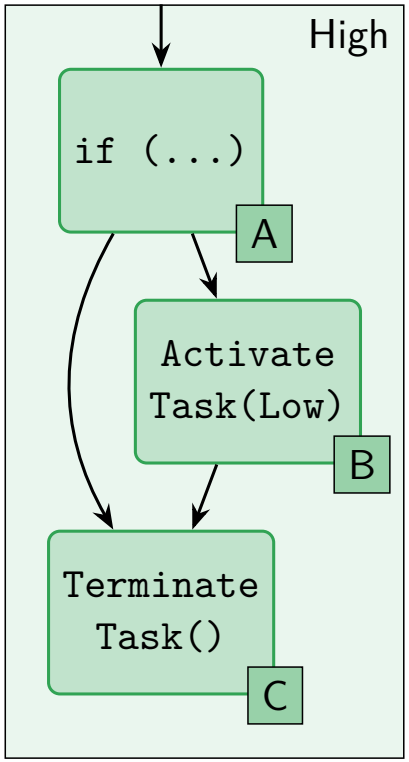


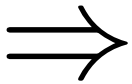


1. Statische Aufzählung der Systemzustände

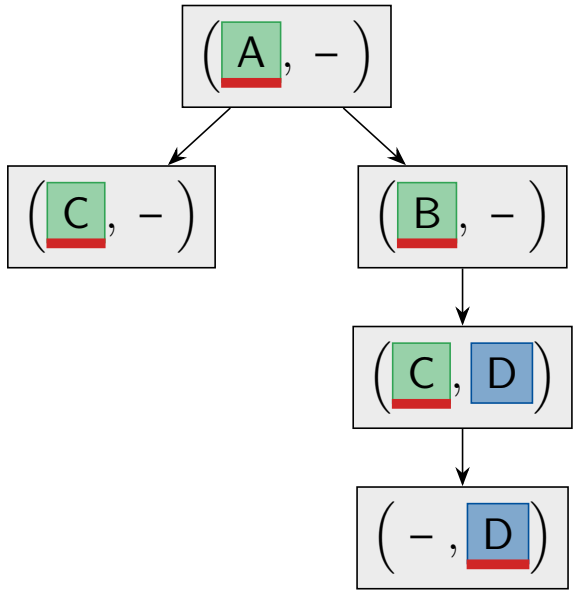
- Systemkonfiguration: Anwendung und RTOS
- Spaltung des Zustandsraums bei Bedingungen
- Systemaufruf modifizieren RTOS-Zustand

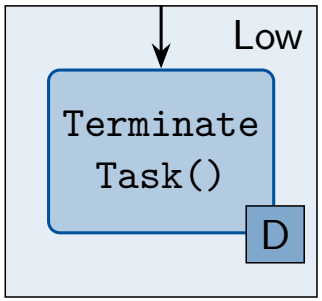
Feingranulare Interaktionsanalyse





Systemzustandsgraph (SSTG)

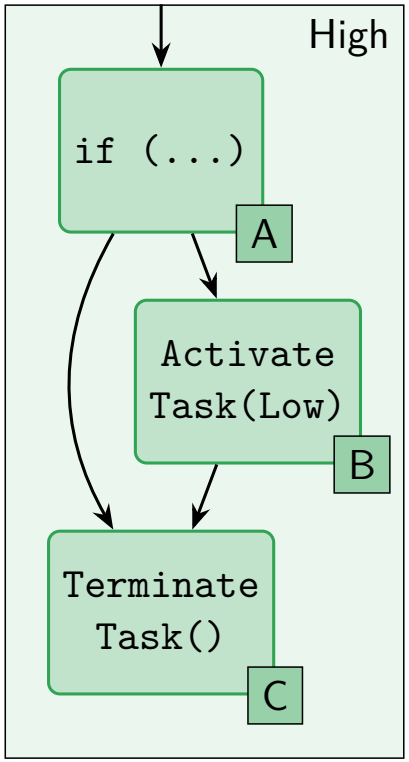


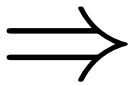


1. Statische Aufzählung der Systemzustände

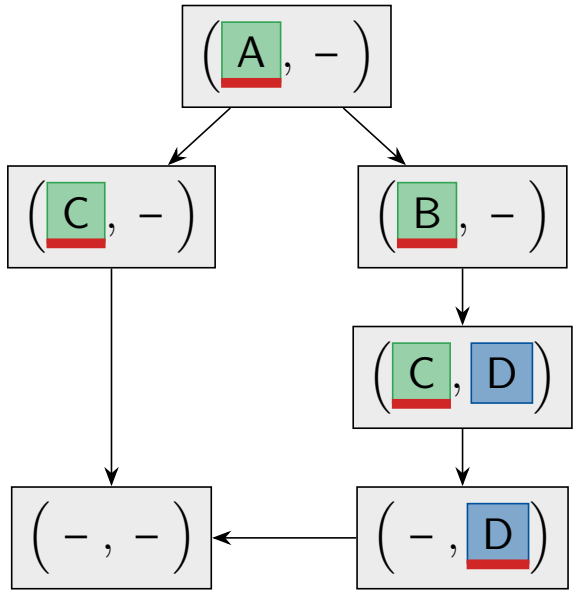
- Systemkonfiguration: Anwendung und RTOS
- Spaltung des Zustandsraums bei Bedingungen
- Systemaufruf modifizieren RTOS-Zustand
- RTOS-Semantik bestimmt Threadwechsel

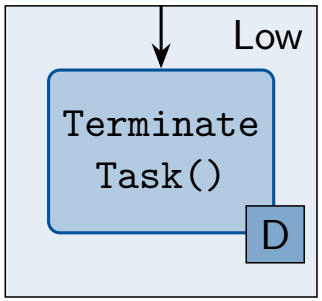
Feingranulare Interaktionsanalyse





Systemzustandsgraph (SSTG)

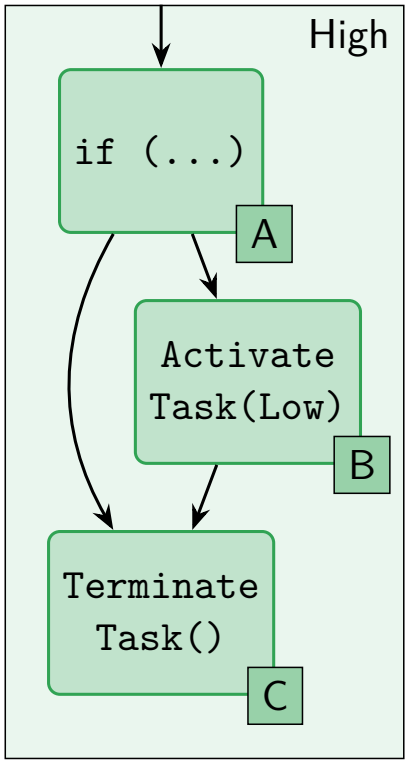


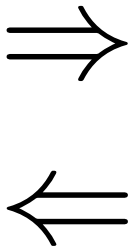


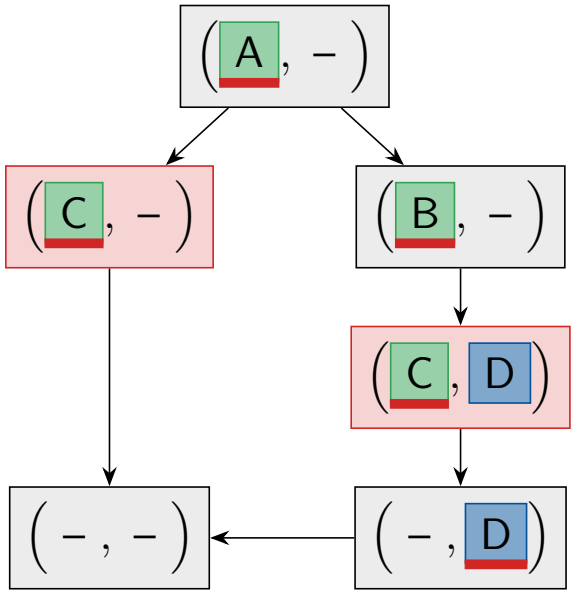
1. Statische Aufzählung der Systemzustände

- Systemkonfiguration: Anwendung und RTOS
- Spaltung des Zustandsraums bei Bedingungen
- Systemaufruf modifizieren RTOS-Zustand
- RTOS-Semantik bestimmt Threadwechsel

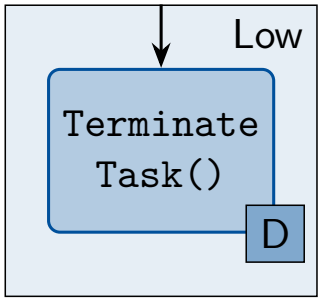
Feingranulare Interaktionsanalyse







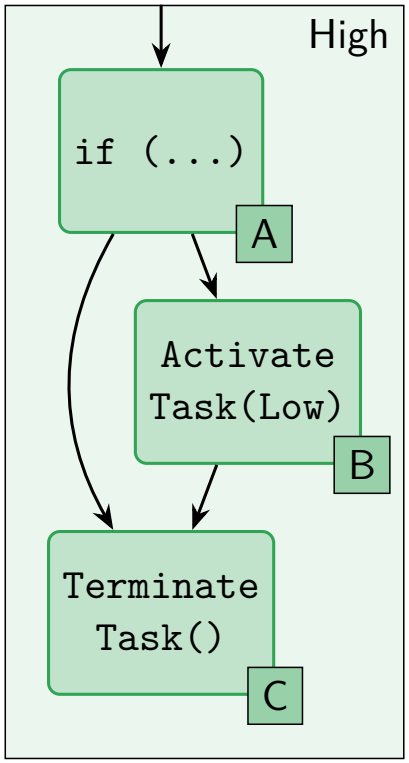
Systemzustandsgraph (SSTG)

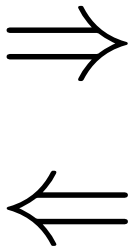


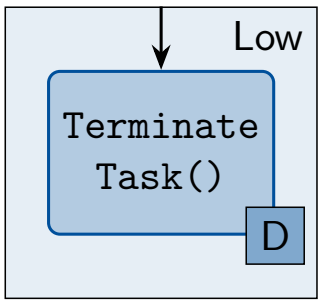
2. Globaler Kontrollflussgraph (GCFG)

- Zusammenfassung von Zuständen

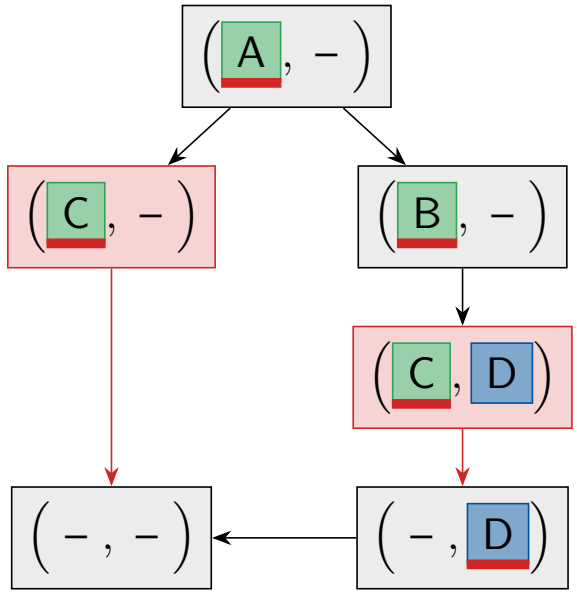
Feingranulare Interaktionsanalyse







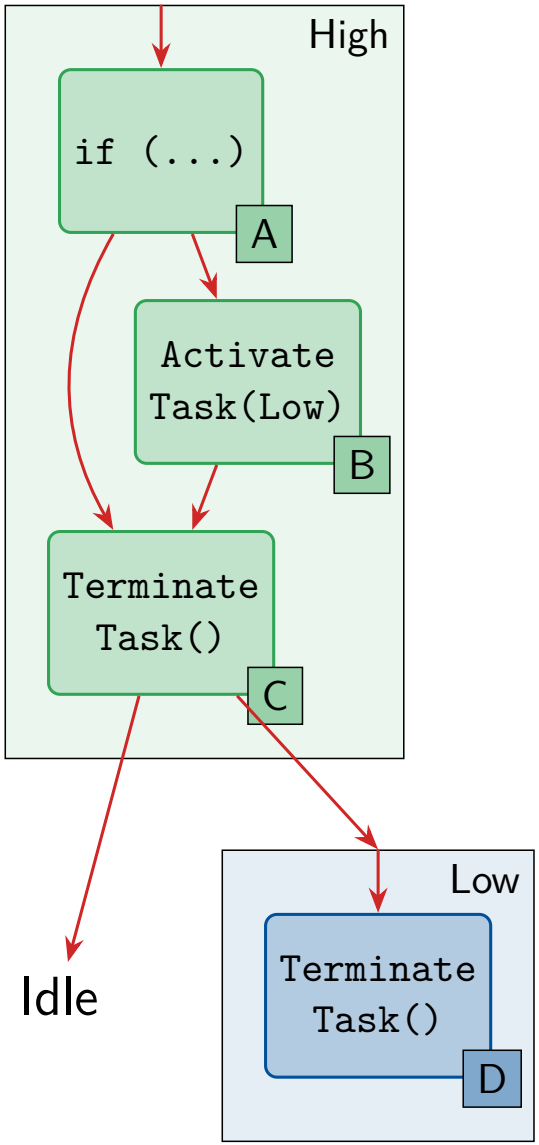
Systemzustandsgraph (SSTG)

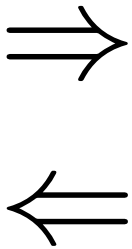


2. Globaler Kontrollflussgraph (GCFG)

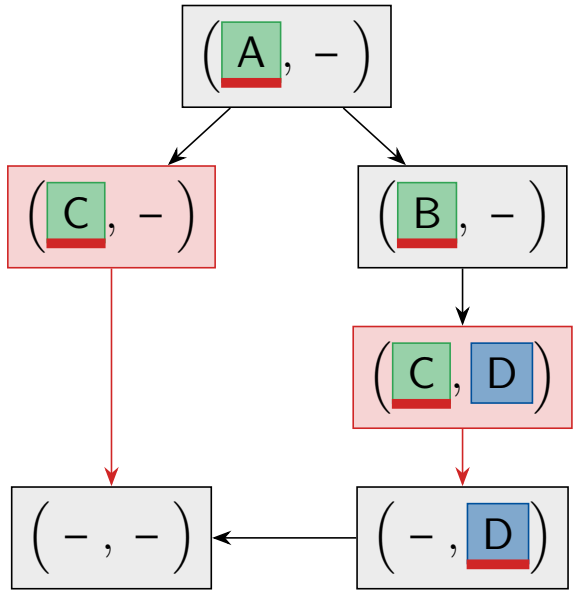
- Zusammenfassung von Zuständen
- Kanten enthalten die RTOS-Entscheidungen

Feingranulare Interaktionsanalyse





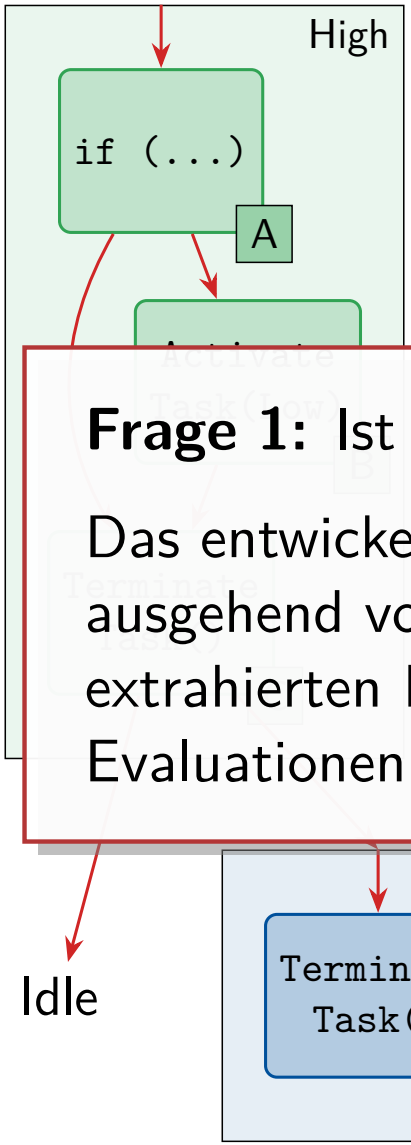
Systemzustandsgraph (SSTG)



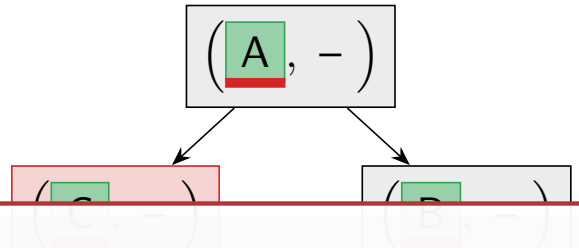
2. Globaler Kontrollflussgraph (GCFG)

- Zusammenfassung von Zuständen
- Kanten enthalten die RTOS-Entscheidungen
- “Gröberes” Interaktionsmodell als SSTG: überdeckt alle potentiellen Pfade

Feingranulare Interaktionsanalyse



Systemzustandsgraph (SSTG)



Frage 1: Ist die Extraktion dieser Interaktionen automatisch möglich?

Das entwickelte *dOSEK*-Framework führt die Interaktionsanalyse ausgehend von Quellcode und Konfiguration **automatisch** durch. Die extrahierten Modelle (SSTG, GCFG) waren die Grundlage für die Evaluationen in **10 Publikationen**.

✓

2. Globaler Kontrollflussgraph (GCFG)

- Zusammenfassung von Zuständen
- Kanten enthalten die RTOS-Entscheidungen
- "Gröberes" Interaktionsmodell als SSTG: überdeckt alle potentiellen Pfade

Übersicht

Grundlagen und Vorarbeiten *(LCTES'15, TECS'17)*

- Feingranulare Interaktionsanalyse
- Systemzustände und Globaler Kontrollflussgraph

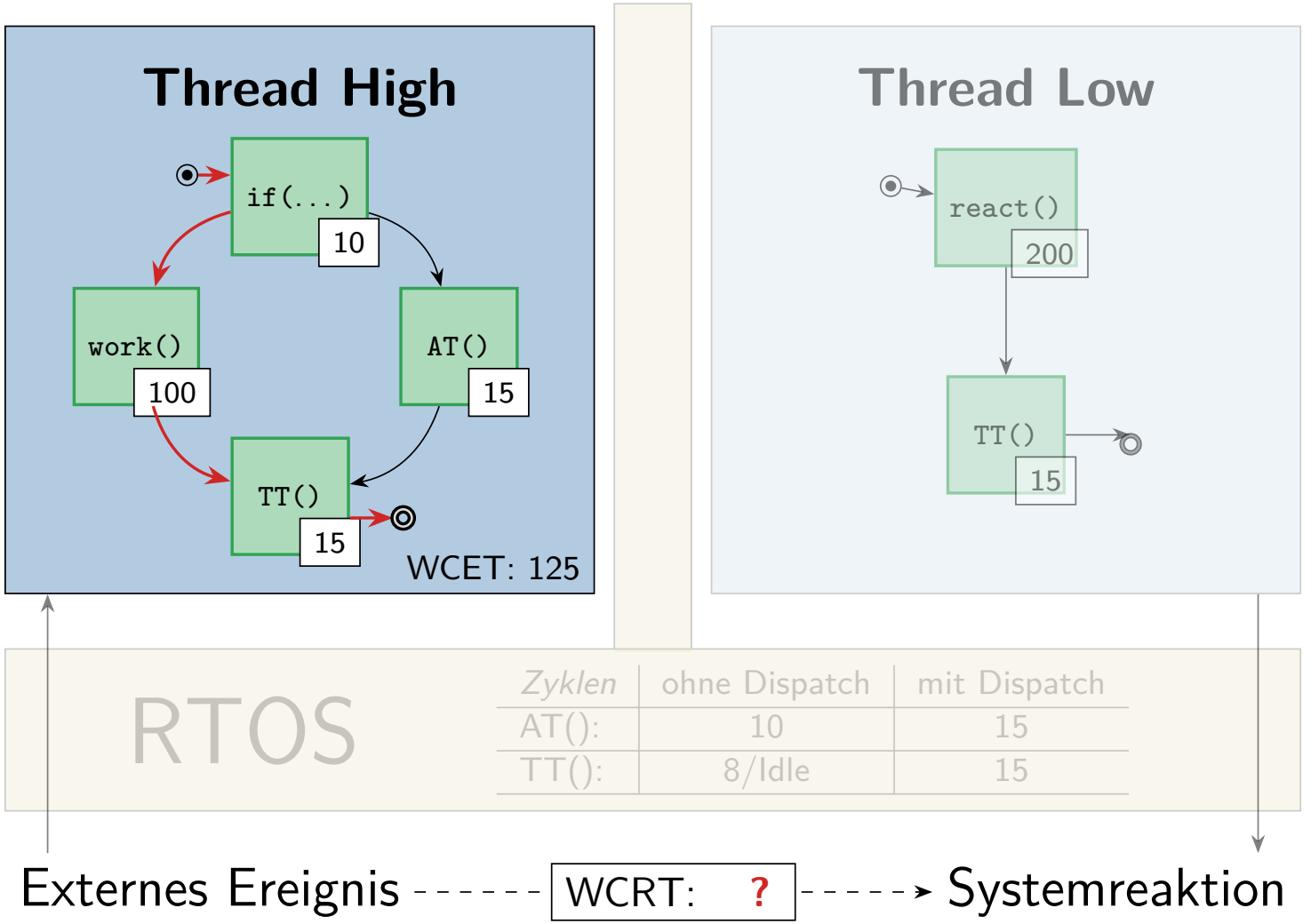
Interaktionsgewahre Systemanalyse

- SysWCET: Flusssensitive Antwortzeitanalyse *(RTAS'17, ECRTS'18)*
- Automatische Verifikation des Kernverhaltens *(FMCAD'17)*

Interaktionsgewahre Systemoptimierung

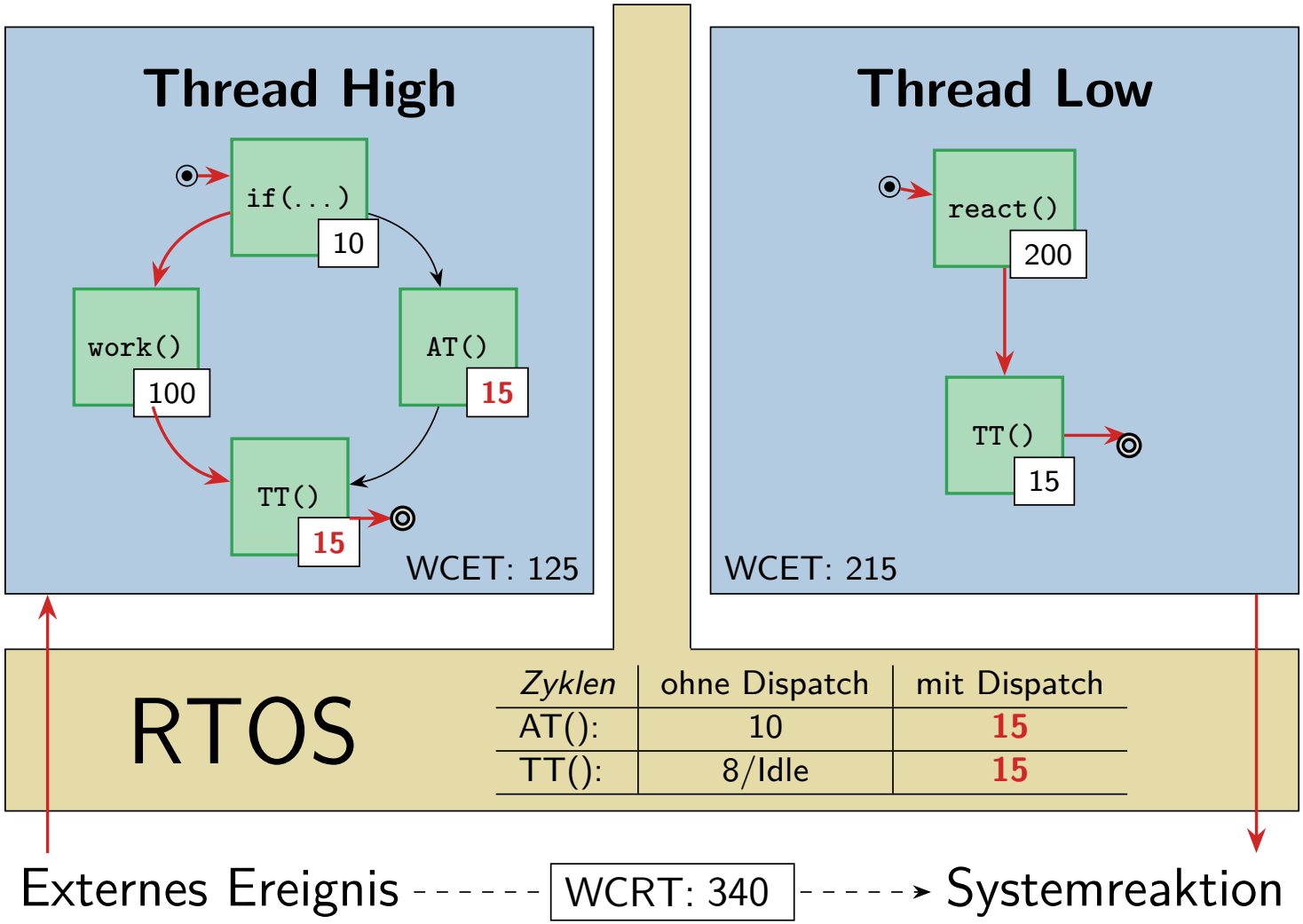
- Semi-Extended Tasks: Stack als geteilte Ressource *(RTSS'18)*
- OSEK-V: RTOS als Prozessorerweiterung *(OSPERT'15, LCTES'17)*

Probleme der kompositionellen Antwortzeitanalyse



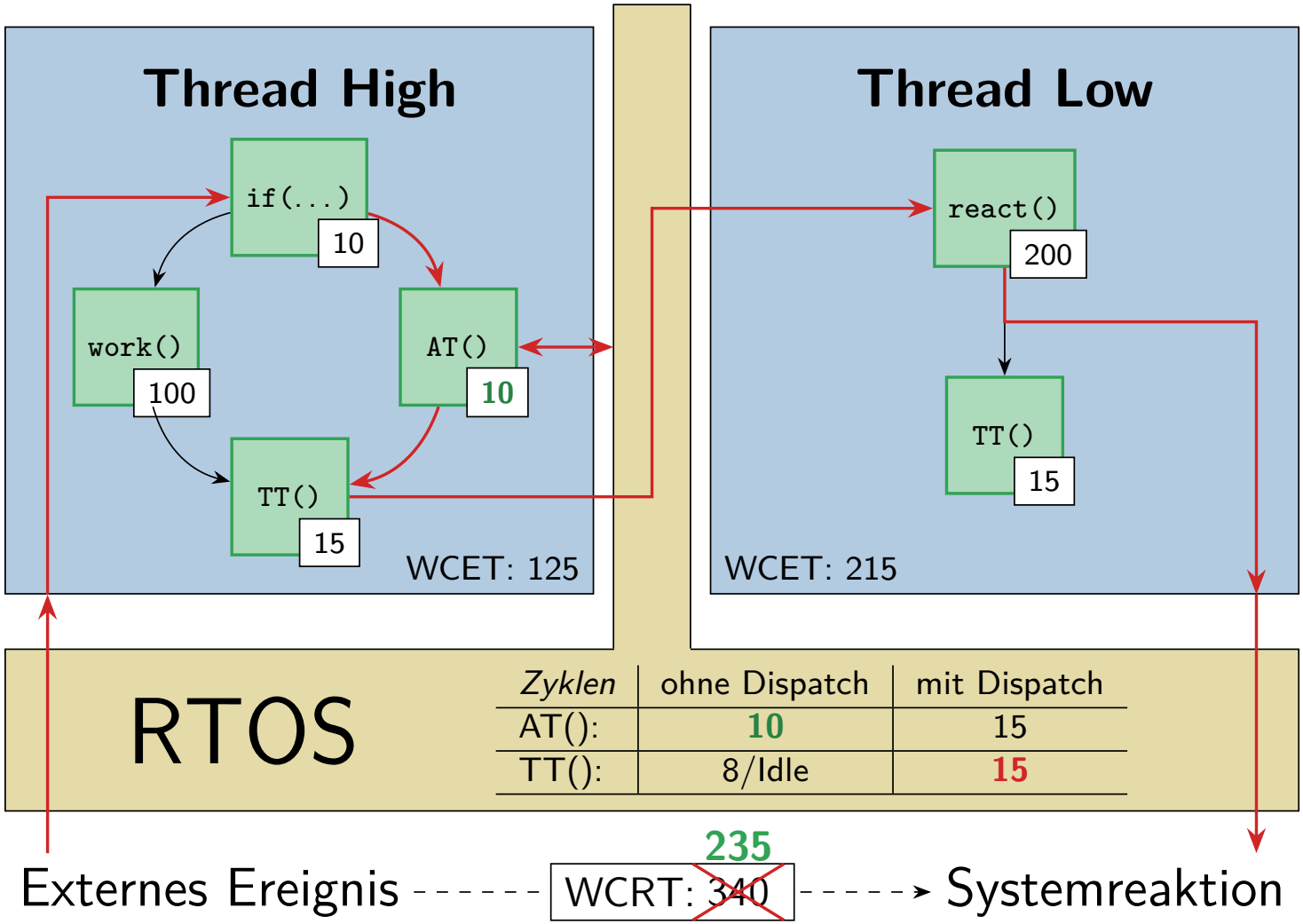
1. Berechnung der Worst-Case Execution Time pro Thread (pessimistisch)
2. Verrechnung der WCETs anhand des Tasks und der Systemparameter

Probleme der kompositionellen Antwortzeitanalyse



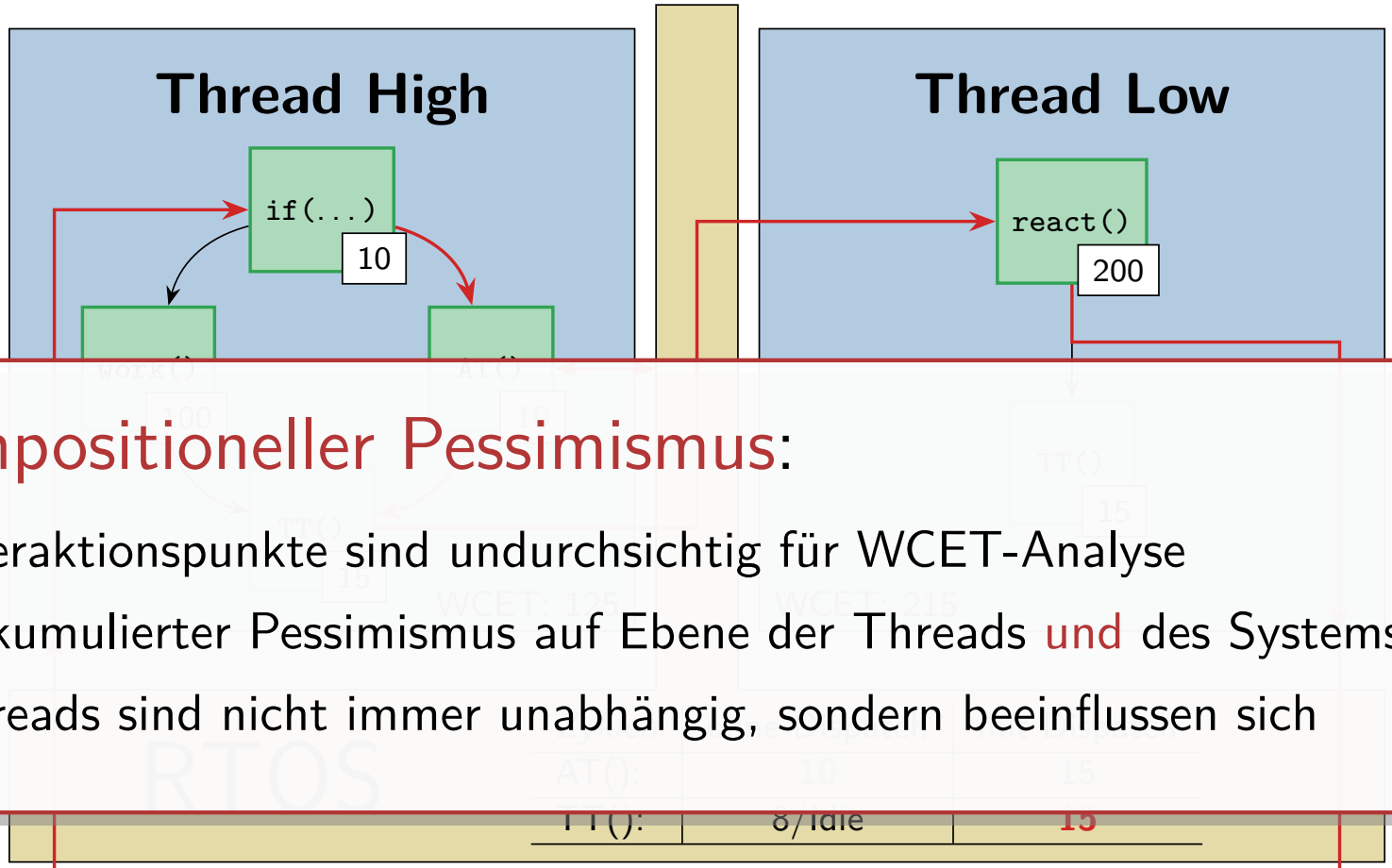
1. Berechnung der Worst-Case Execution Time pro Thread (pessimistisch)
2. Verrechnung der WCETs anhand des Tasks und der Systemparameter

Probleme der kompositionellen Antwortzeitanalyse



1. Berechnung der Worst-Case Execution Time pro Thread (pessimistisch)
2. Verrechnung der WCETs anhand des Tasks und der Systemparameter

Probleme der kompositionellen Antwortzeitanalyse



Kompositioneller Pessimismus:

- Interaktionspunkte sind undurchsichtig für WCET-Analyse
- Akkumulierter Pessimismus auf Ebene der Threads und des Systems
- Threads sind nicht immer unabhängig, sondern beeinflussen sich

Externes Ereignis ------ WCRT: ~~340~~ 235 ------➤ Systemreaktion

1. Berechnung der Worst-Case Execution Time pro Thread (pessimistisch)
2. Verrechnung der WCETs anhand des Tasks und der Systemparameter

SysWCET: Interaktionsgewahre Antwortzeitanalyse

- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

SysWCET: Interaktionsgewahre Antwortzeitanalyse

- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

- SysWCET hebt WCET-Analyse auf die Systemebene

> *Die Antwortzeit eines Jobs*
> *im Kontext eines Gesamtsystems...*
>
> *...ist die Ausführungszeit des Systems*
> *beim Verarbeiten des Jobs.*

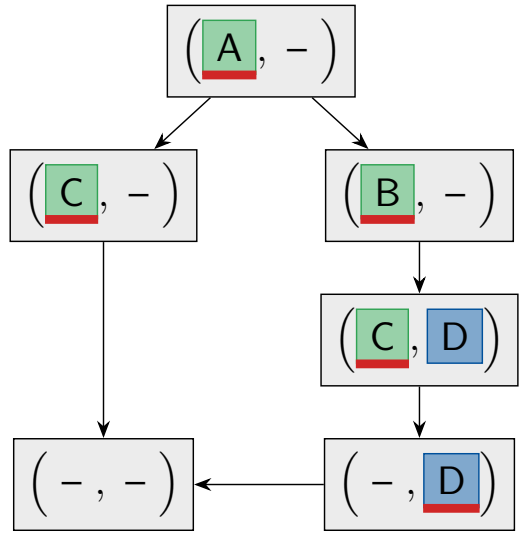
SysWCET: Interaktionsgewahre Antwortzeitanalyse

- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

- SysWCET hebt WCET-Analyse auf die Systemebene

> *Die Antwortzeit eines Jobs*
> *im Kontext eines Gesamtsystems. . .*
>
> *. . . ist die Ausführungszeit des Systems*
> *beim Verarbeiten des Jobs.*

  - SSTG enthält alle Pfade durchs System

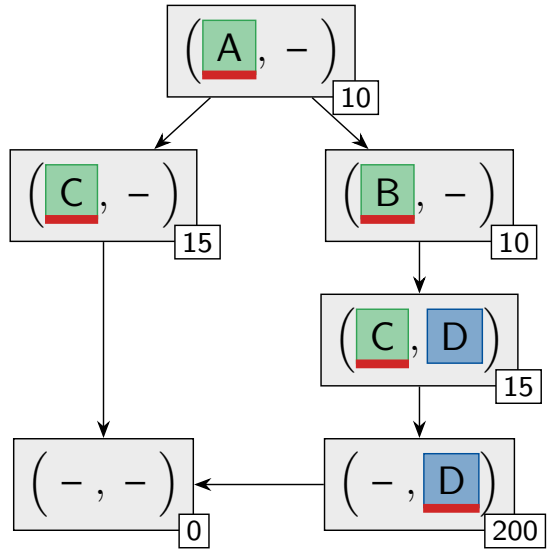


SysWCET: Interaktionsgewahre Antwortzeitanalyse

- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

- SysWCET hebt WCET-Analyse auf die Systemebene

> *Die Antwortzeit eines Jobs*
> *im Kontext eines Gesamtsystems. . .*
>
> *. . . ist die Ausführungszeit des Systems*
> *beim Verarbeiten des Jobs.*



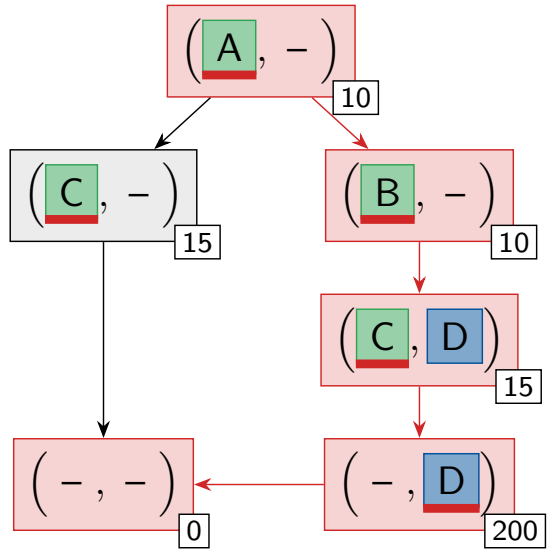
  - SSTG enthält alle Pfade durchs System
  - Jeder SSTG-Zustand führt einen Block aus.

SysWCET: Interaktionsgewahre Antwortzeitanalyse

- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

- SysWCET hebt WCET-Analyse auf die Systemebene

> *Die Antwortzeit eines Jobs*
> *im Kontext eines Gesamtsystems. . .*
>
> *. . . ist die Ausführungszeit des Systems*
> *beim Verarbeiten des Jobs.*



  - SSTG enthält alle Pfade durchs System
  - Jeder SSTG-Zustand führt einen Block aus.
  - Finde längsten Pfad im Zustandsgraphen

SysWCET: Interaktionsgewahre Antwortzeitanalyse

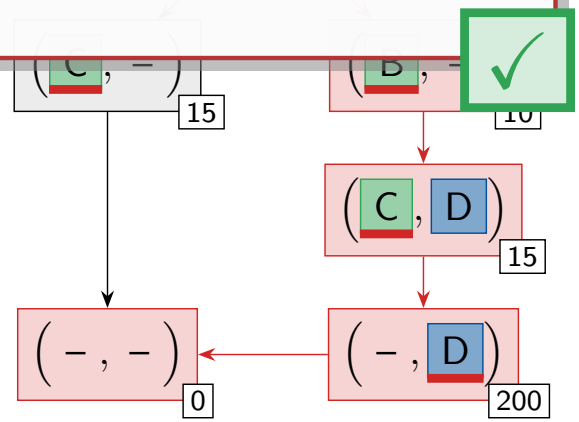
- Klassische WCET-Analyse für einzelne Funktionen/Threads
  - Finde längsten/zeitlich teuersten Ausführungspfad im Programm
  - Modellierung des längsten Pfades als Flussproblem (inkl. Schleifen)
  - Implizite Pfadaufzählung (IPET), Ganzahlig lineares Programm (ILP)

Frage 2: Führt die Aufspaltung der Anwendung in Threads zu pessimistischeren Analysen und ineffizienteren Implementierungen?

Die kompositionelle WCRT-Analyse behandelt Threads getrennt und ist daher pessimistischer als eigentlich nötig. SysWCET überwindet diese **Trennung** und kann engere Schranken (bis zu 10.5% enger) angeben.

beim Verarbeiten des Jobs.

- SSTG enthält alle Pfade durchs System
- Jeder SSTG-Zustand führt einen Block aus.
- Finde längsten Pfad im Zustandsgraphen



Übersicht

Grundlagen und Vorarbeiten *(LCTES'15, TECS'17)*

- Feingranulare Interaktionsanalyse
- Systemzustände und Globaler Kontrollflussgraph

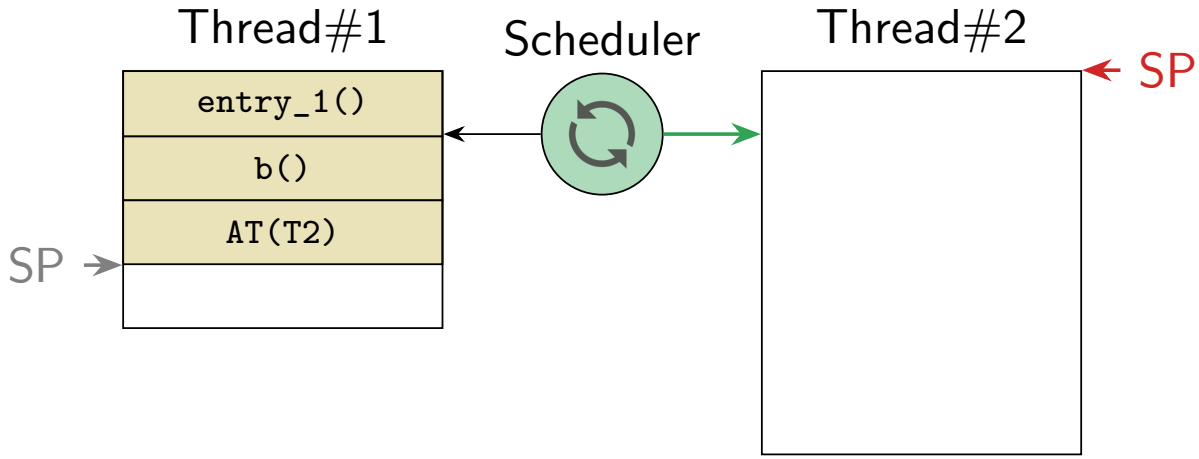
Interaktionsgewahre Systemanalyse

- SysWCET: Flusssensitive Antwortzeitanalyse *(RTAS'17, ECRTS'18)*
- Automatische Verifikation des Kernverhaltens *(FMCAD'17)*

Interaktionsgewahre Systemoptimierung

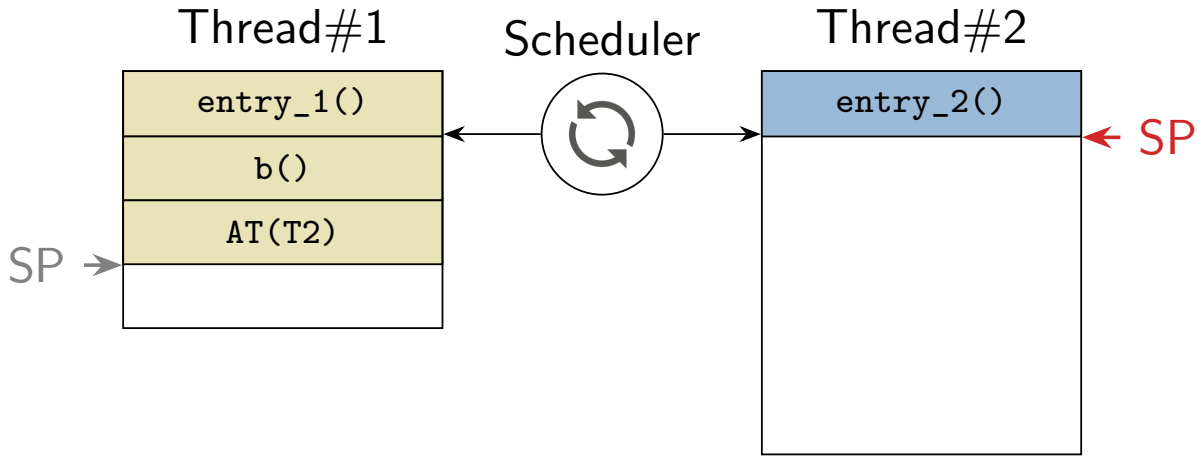
- Semi-Extended Tasks: Stack als geteilte Ressource *(RTSS'18)*
- OSEK-V: RTOS als Prozessorerweiterung *(OSPERT'15, LCTES'17)*

Statische Allokation von Stackreservierungen



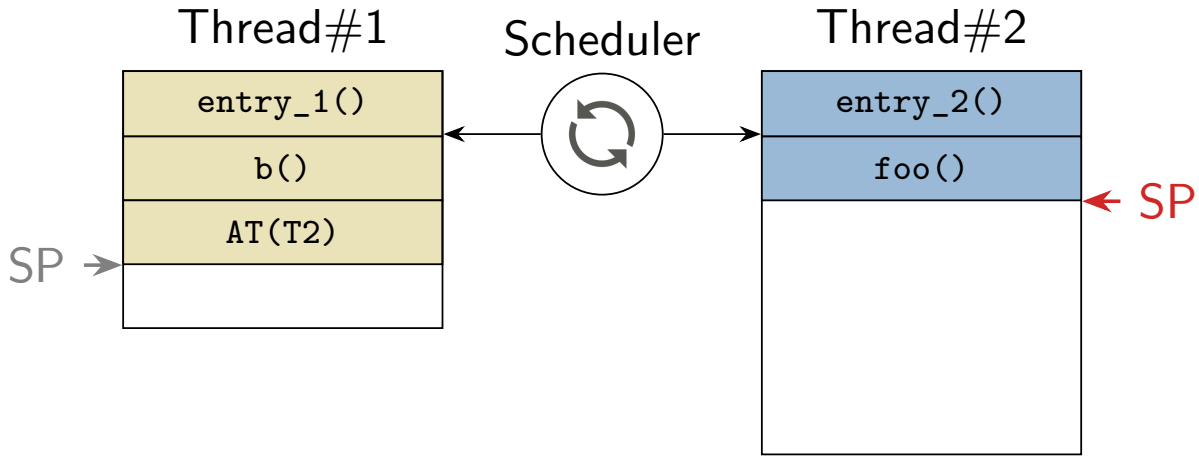
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion

Statische Allokation von Stackreservierungen



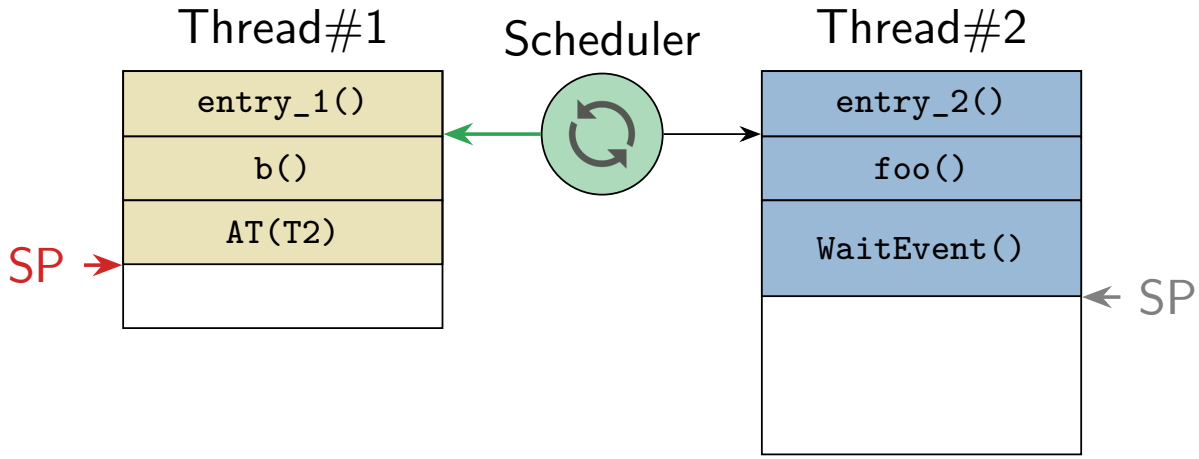
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion

Statische Allokation von Stackreservierungen



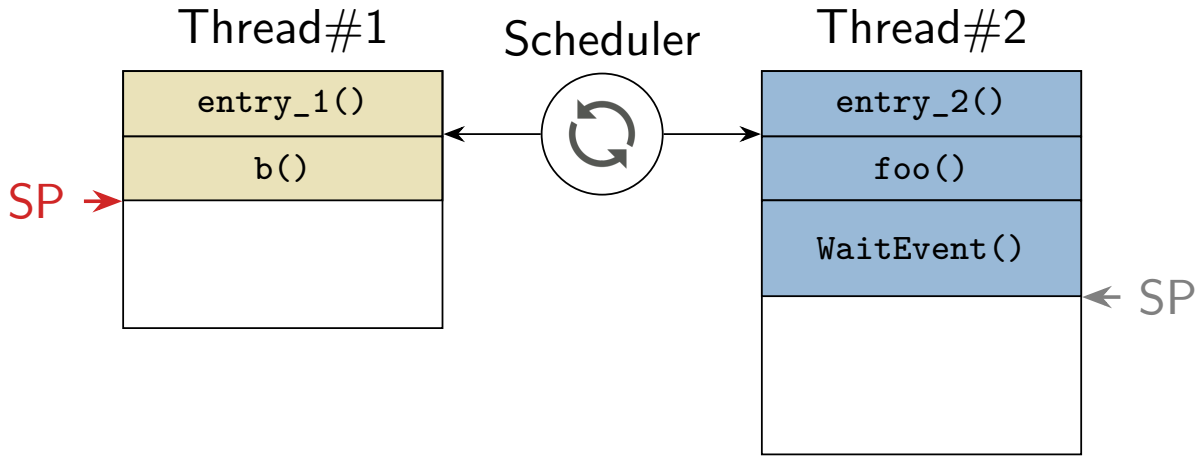
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion

Statische Allokation von Stackreservierungen



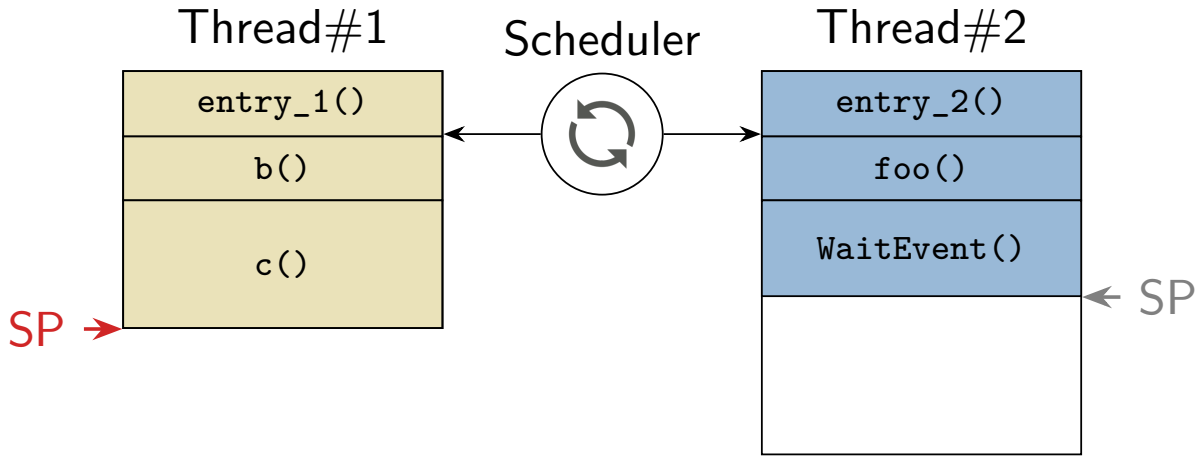
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion
  - RTOS wechselt mit dem Thread auch den Stack

Statische Allokation von Stackreservierungen



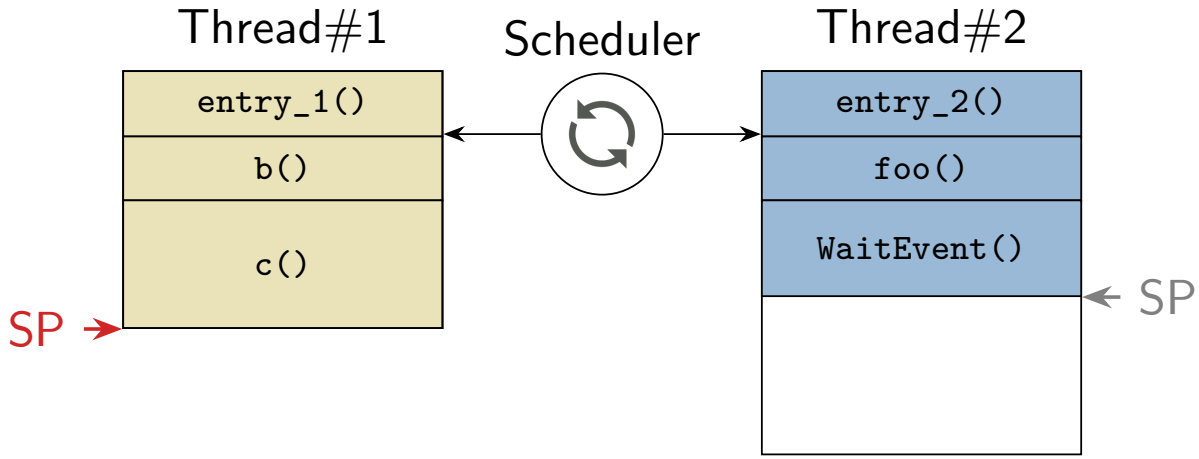
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion
  - RTOS wechselt mit dem Thread auch den Stack

Statische Allokation von Stackreservierungen



- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion
  - RTOS wechselt mit dem Thread auch den Stack

Statische Allokation von Stackreservierungen



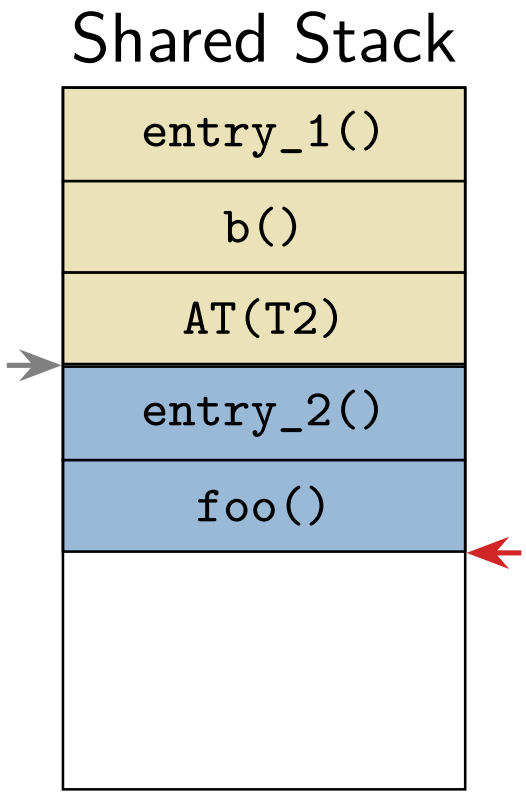
- Thread-private Stackreservierungen bieten Platz für Call-Frames
  - Allokation von Call-Frames ohne RTOS Interaktion
  - RTOS wechselt mit dem Thread auch den Stack

- **Private** Reservierungen bieten eine hohe Flexibilität
  - \+ (Re-)Scheduling und Stackwechsel zu jedem Zeitpunkt möglich
  - − Größe der Reservierung: oberen Schranke pro Thread

Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

⇒ Industriestandard: OSEK Basic Tasks

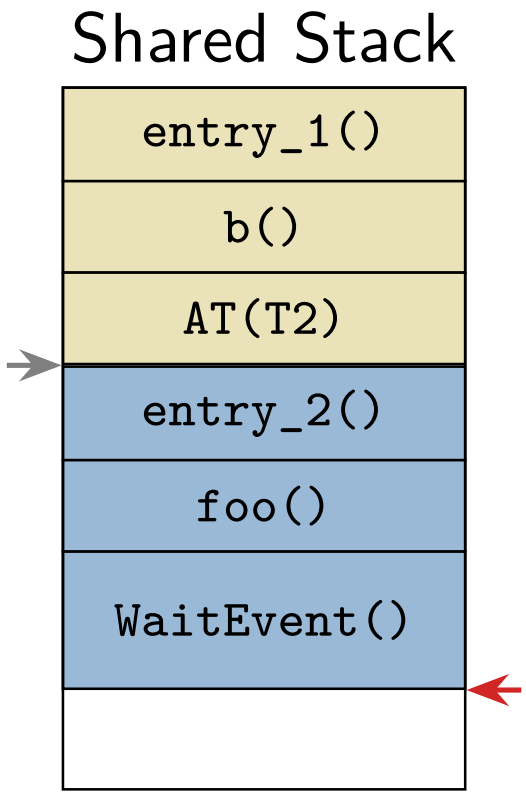


Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

  ⇒ Industriestandard: OSEK Basic Tasks

− Basic Tasks dürfen nicht passiv warten.
  - Verdeckte Threads können Lücken erzeugen.
  - Aktive Call-Frames können zerstört werden.

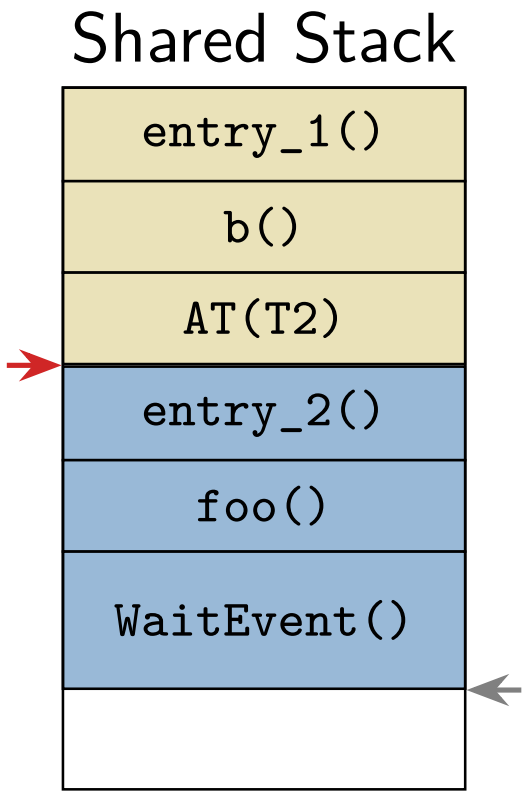


Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

  ⇒ Industriestandard: OSEK Basic Tasks

− Basic Tasks dürfen nicht passiv warten.
  - Verdeckte Threads können Lücken erzeugen.
  - Aktive Call-Frames können zerstört werden.

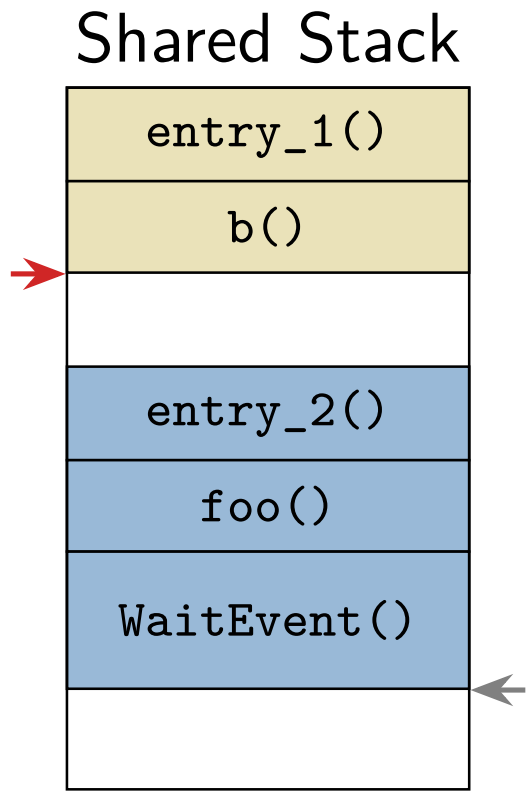


Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

  $\Rightarrow$ Industriestandard: OSEK Basic Tasks

− Basic Tasks dürfen nicht passiv warten.
  - Verdeckte Threads können Lücken erzeugen.
  - Aktive Call-Frames können zerstört werden.

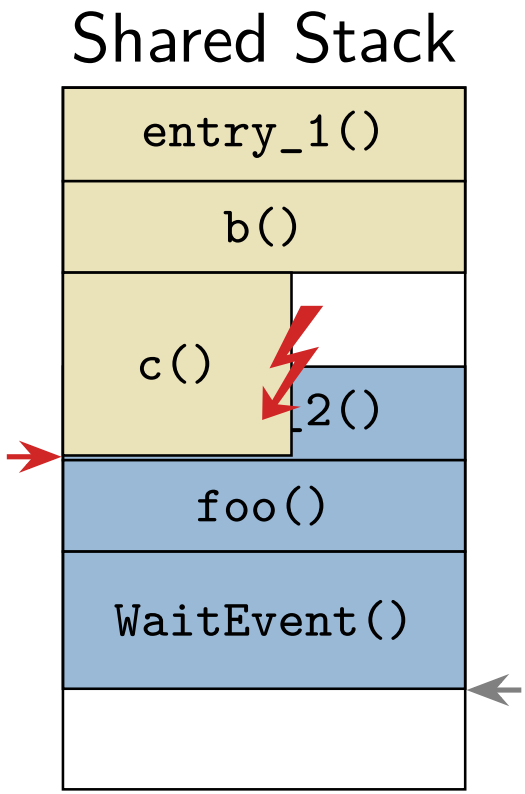


Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

  ⇒ Industriestandard: OSEK Basic Tasks

− Basic Tasks dürfen nicht passiv warten.
  - Verdeckte Threads können Lücken erzeugen.
  - Aktive Call-Frames können zerstört werden.

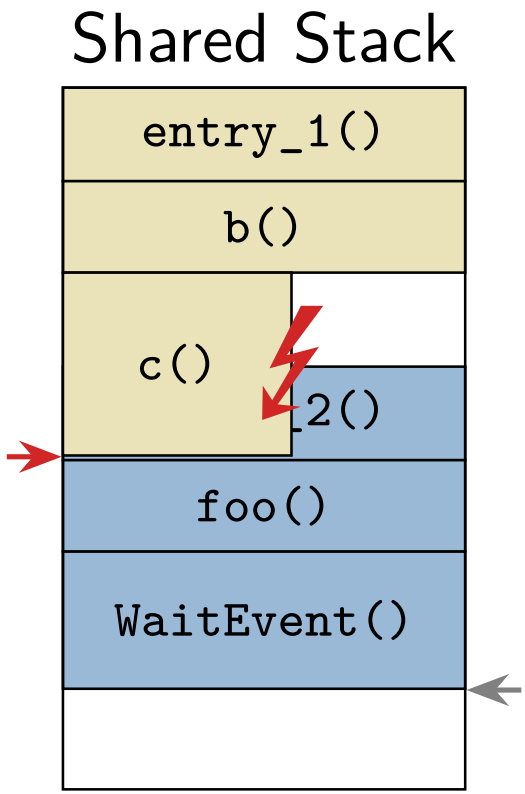


Stackreservierung als geteilte Ressource

+ Geteilte Reservierungen effizienter nutzbar
  - Doppelnutzung bei gegenseitigem Ausschluss
  - Nur oberster Thread darf lauffähig sein

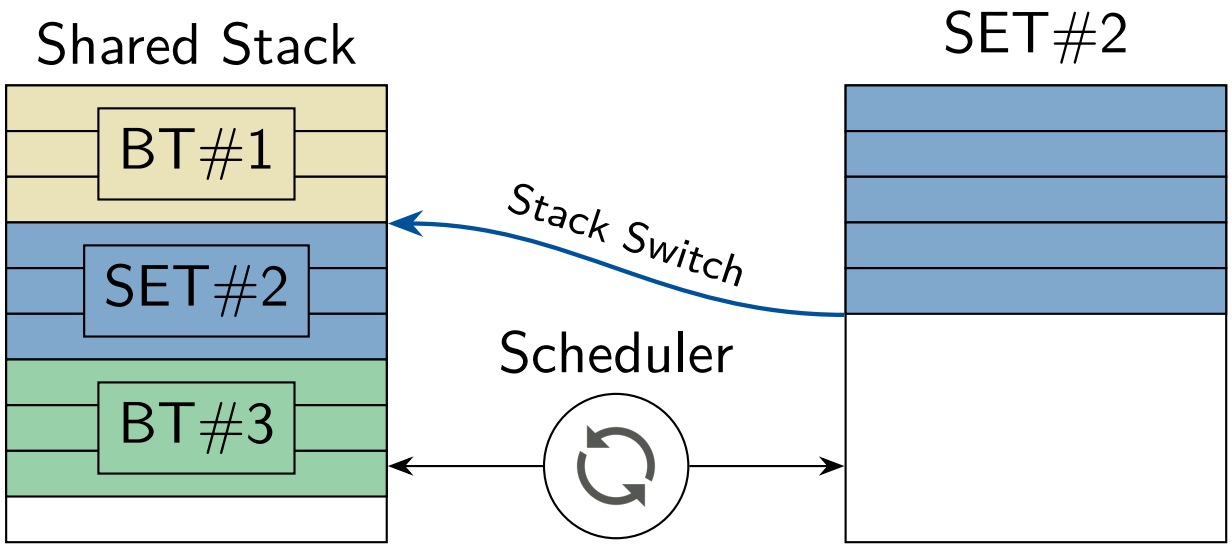
  ⇒ Industriestandard: OSEK Basic Tasks

– Basic Tasks dürfen nicht passiv warten.
  - Verdeckte Threads können Lücken erzeugen.
  - Aktive Call-Frames können zerstört werden.



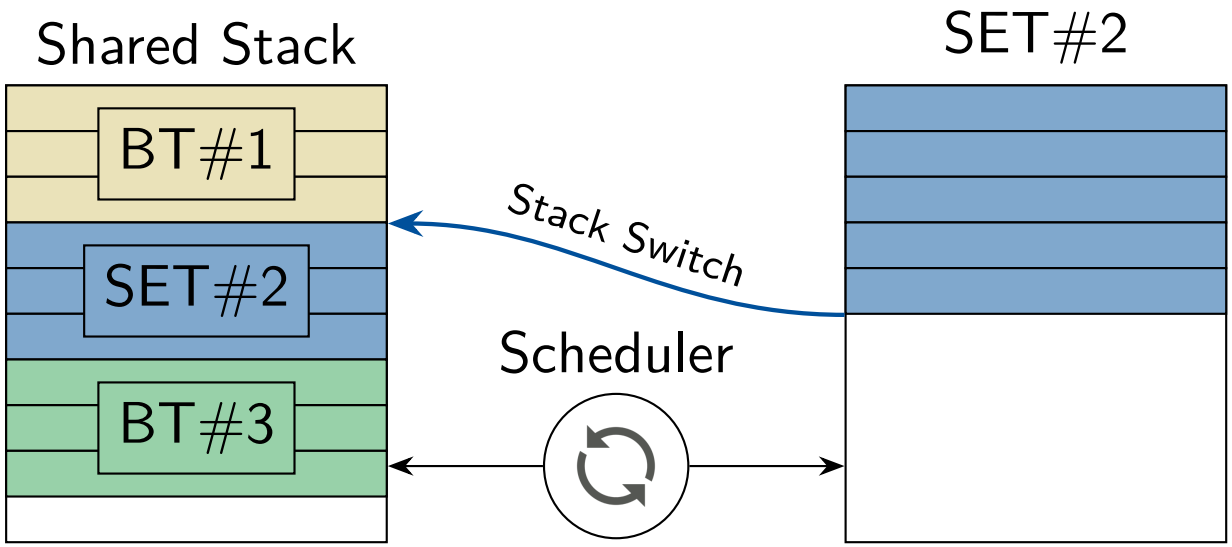
Problem 1: Wie können wir Reservierungen teilen, obwohl gewartet wird?
Problem 2: Wie groß muss die geteilte Reservierung sein?

Semi-Extended Tasks: Teilen trotz passiven Wartens



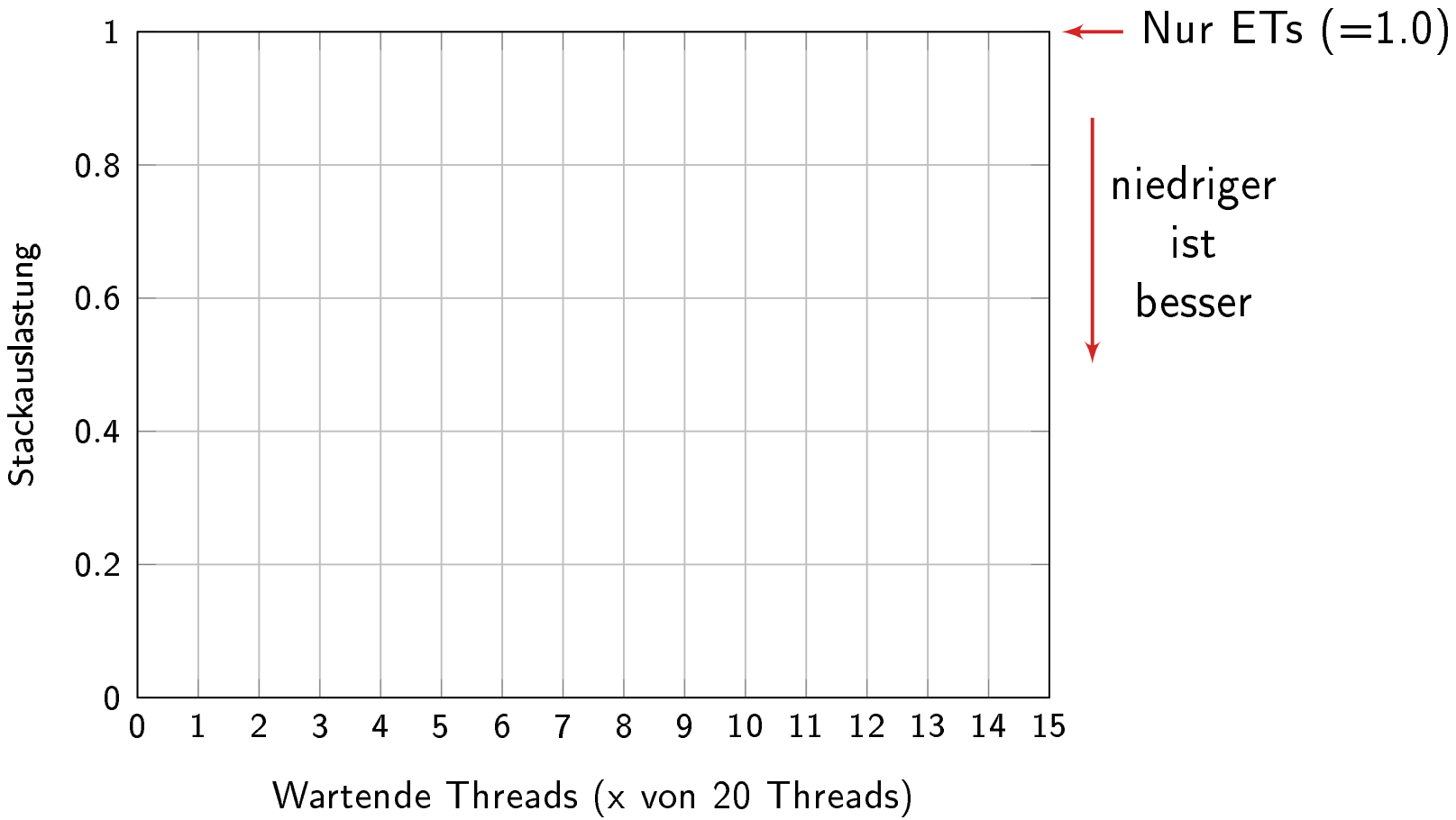
- SET platziert Frames in privaten und geteilten Reservierungen
  - Thread startet im Privaten und wechselt für wartefreie Phasen
  - Wechsel ohne RTOS-Interaktion durch modifizierte Funktionsprologe
  - Mehr Threads in der gleichen Reservierung: höheres Einsparpotential

Semi-Extended Tasks: Teilen trotz passiven Wartens

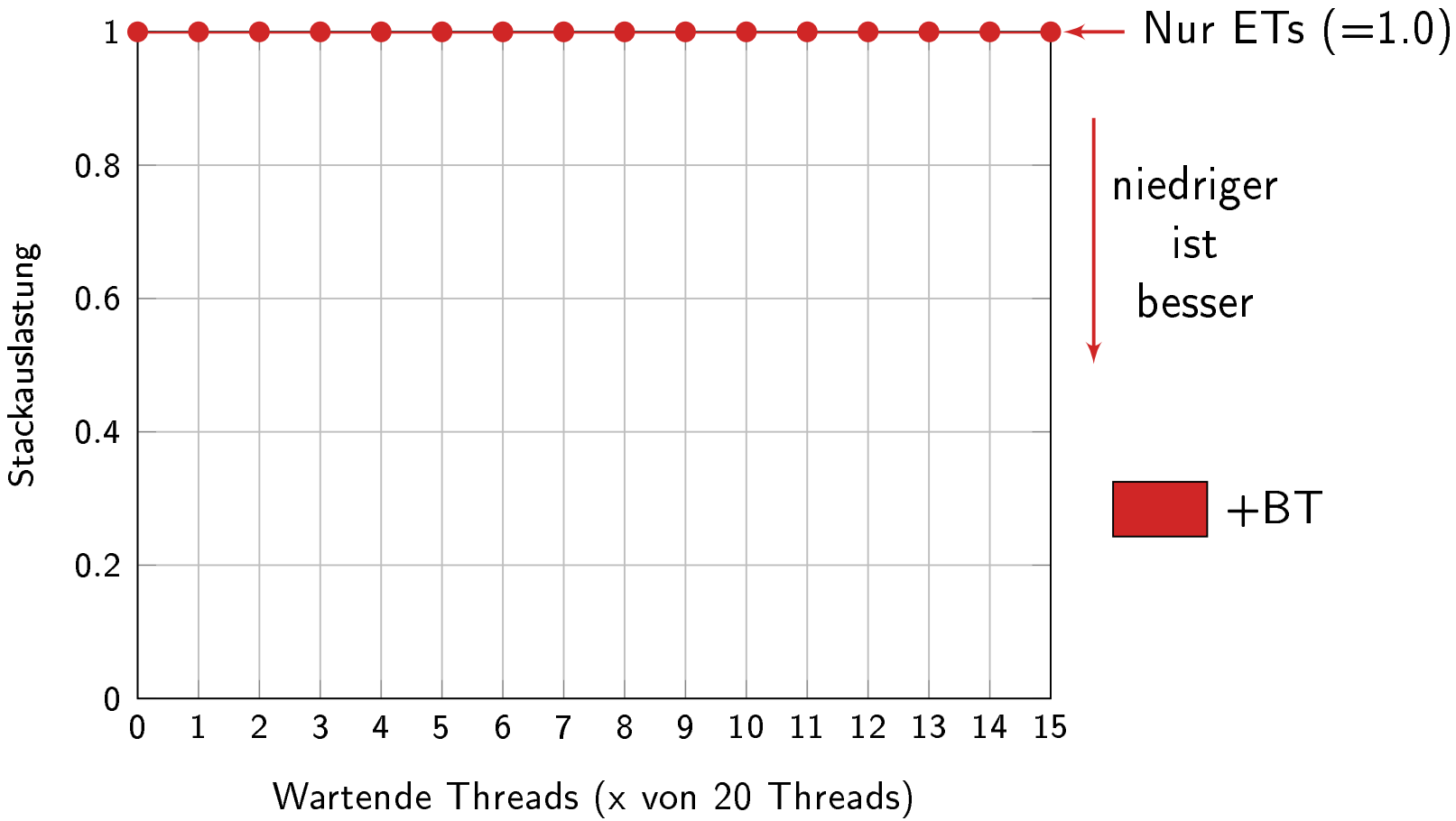


- SET platziert Frames in privaten und geteilten Reservierungen
  - Thread startet im Privaten und wechselt für wartefreie Phasen
  - Wechsel ohne RTOS-Interaktion durch modifizierte Funktionsprologe
  - Mehr Threads in der gleichen Reservierung: höheres Einsparpotential
- Optimierung des Speicherbedarfs durch **Interaktionswissen**
  1. Erkennung wartefreier Phasen: statische Analyse des Aufrufgraphen
  2. Genetischer Algorithmus findet gute Konfiguration der Umstiegspunkte
  3. Worst-Case Stack Consumption Analyse für verschiedene Konfigurationen

Stackverbrauch mit wartendenden Threads

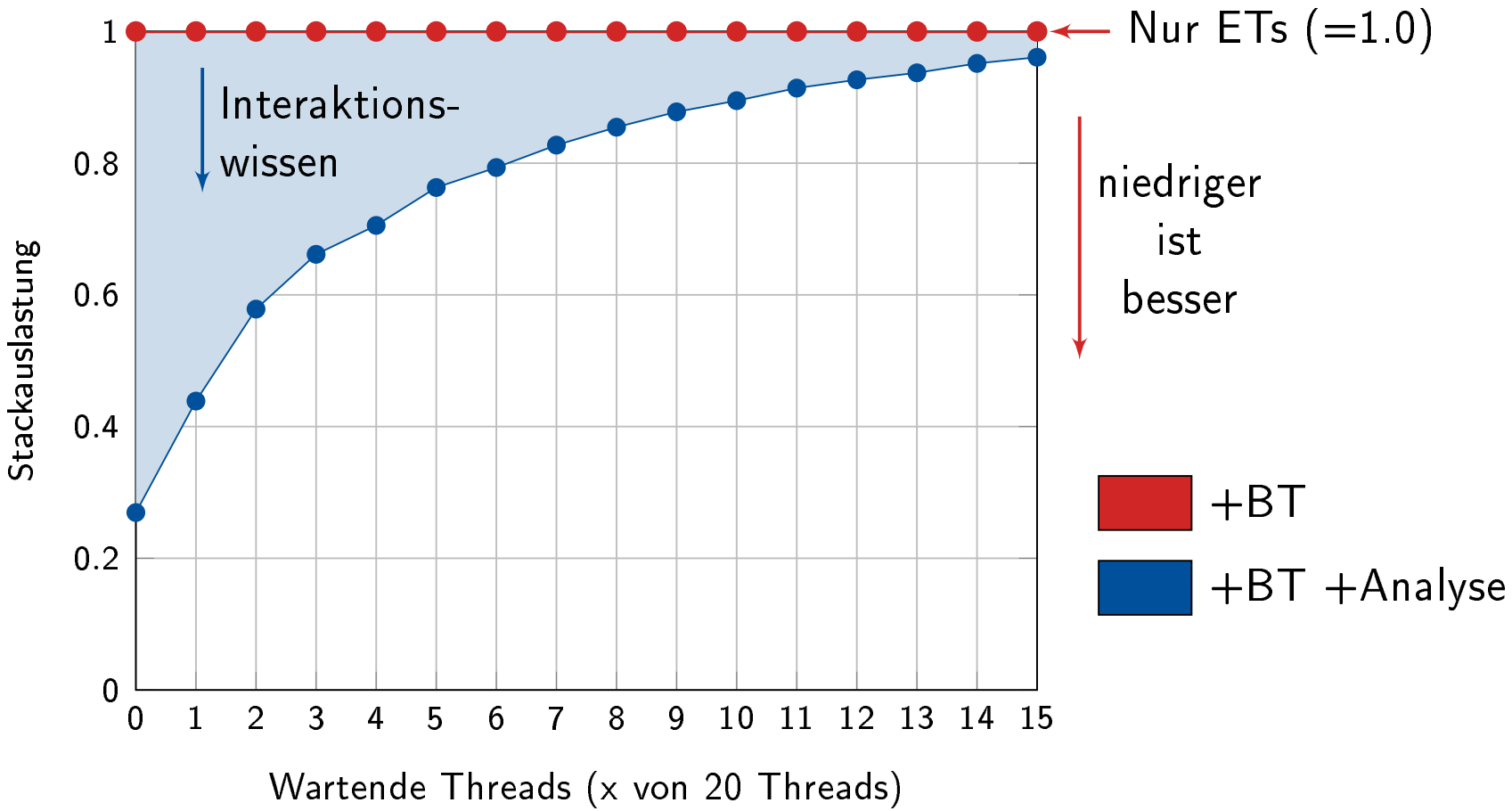


Stackverbrauch mit wartendenden Threads



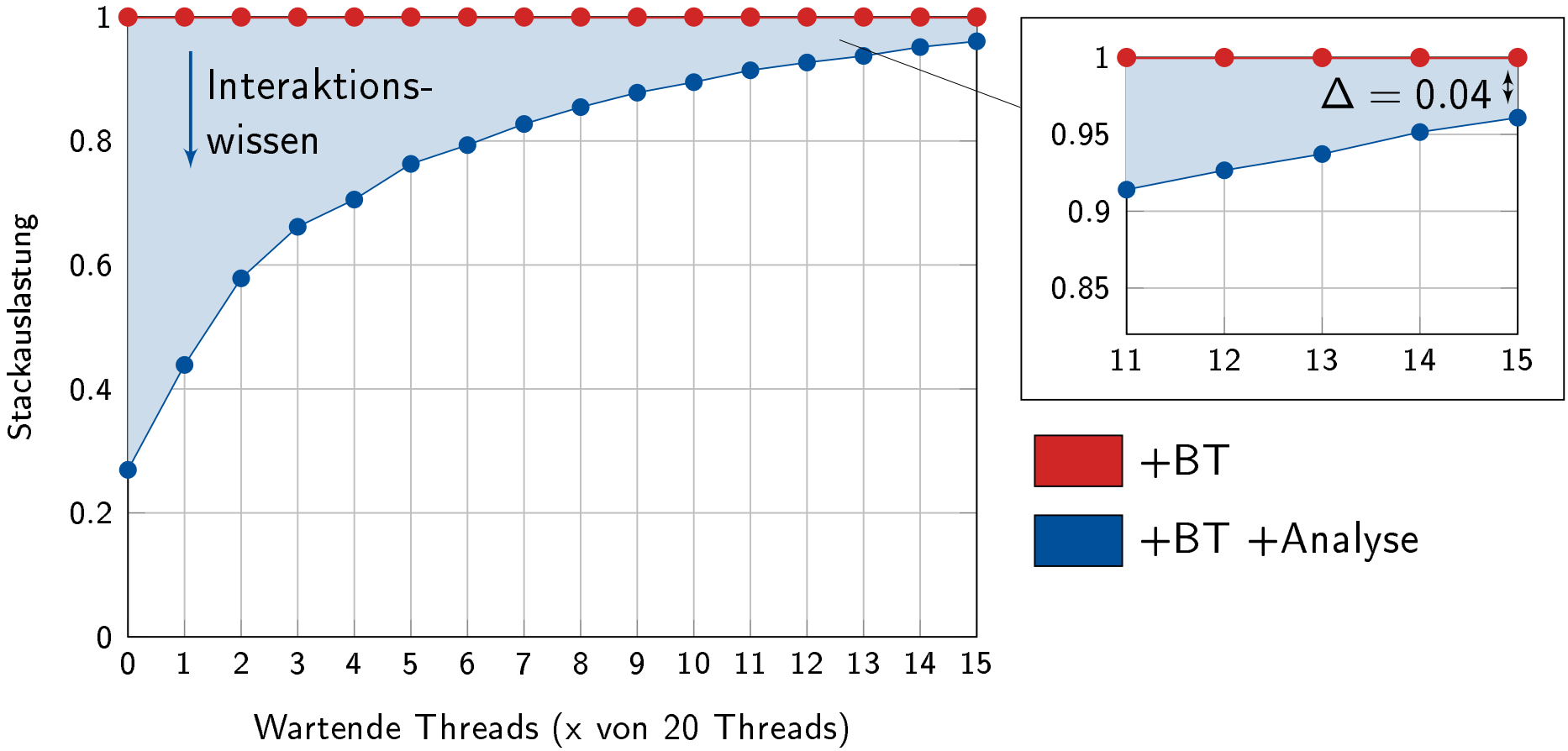
- Thread-granulare WCSC-Analyse zeigt trotz BTs keine Einsparungen.

Stackverbrauch mit wartendenden Threads



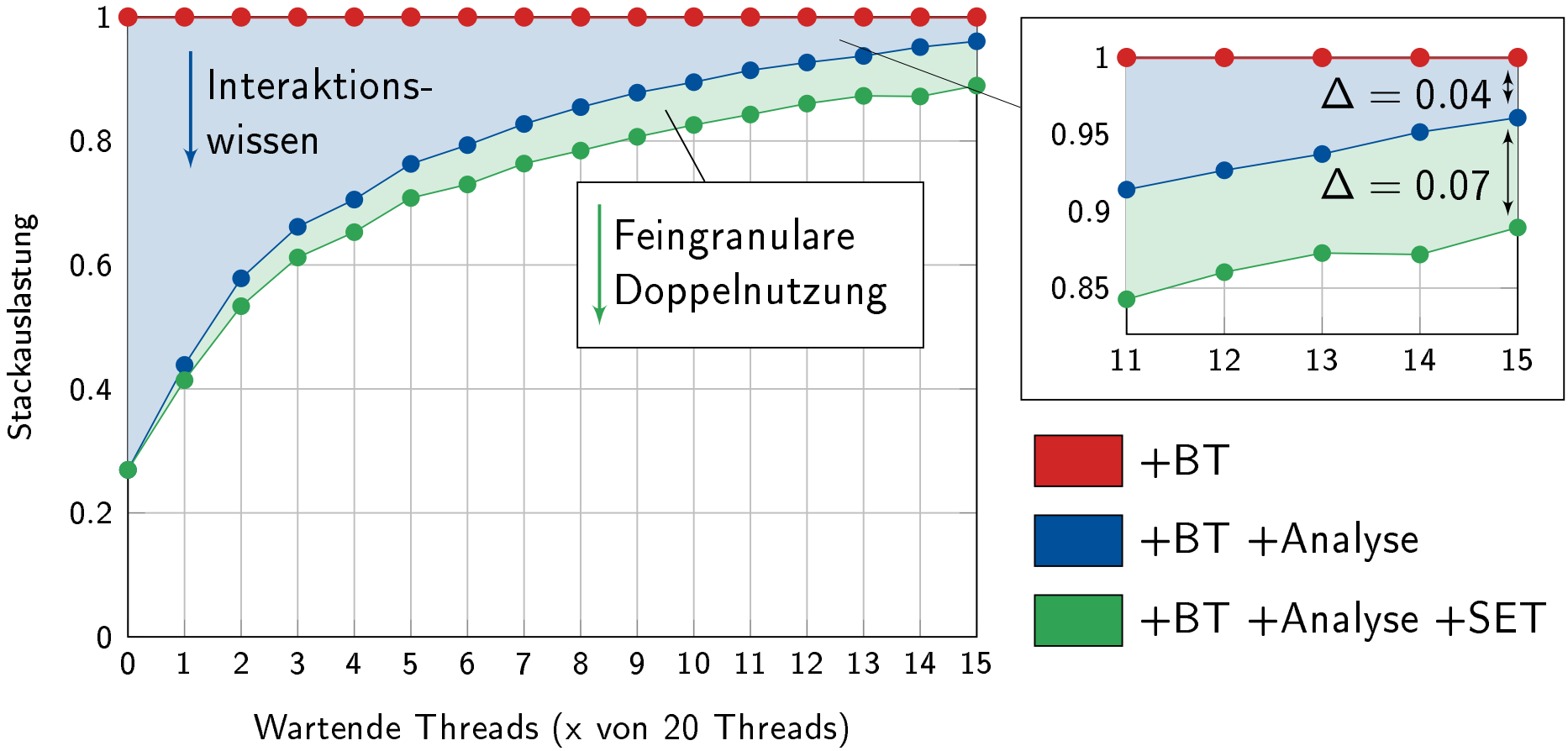
- Thread-granulare WCSC-Analyse zeigt trotz BTs keine Einsparungen.
- Interaktionswissen bringt deutlich engere obere Schranken.

Stackverbrauch mit wartendenden Threads



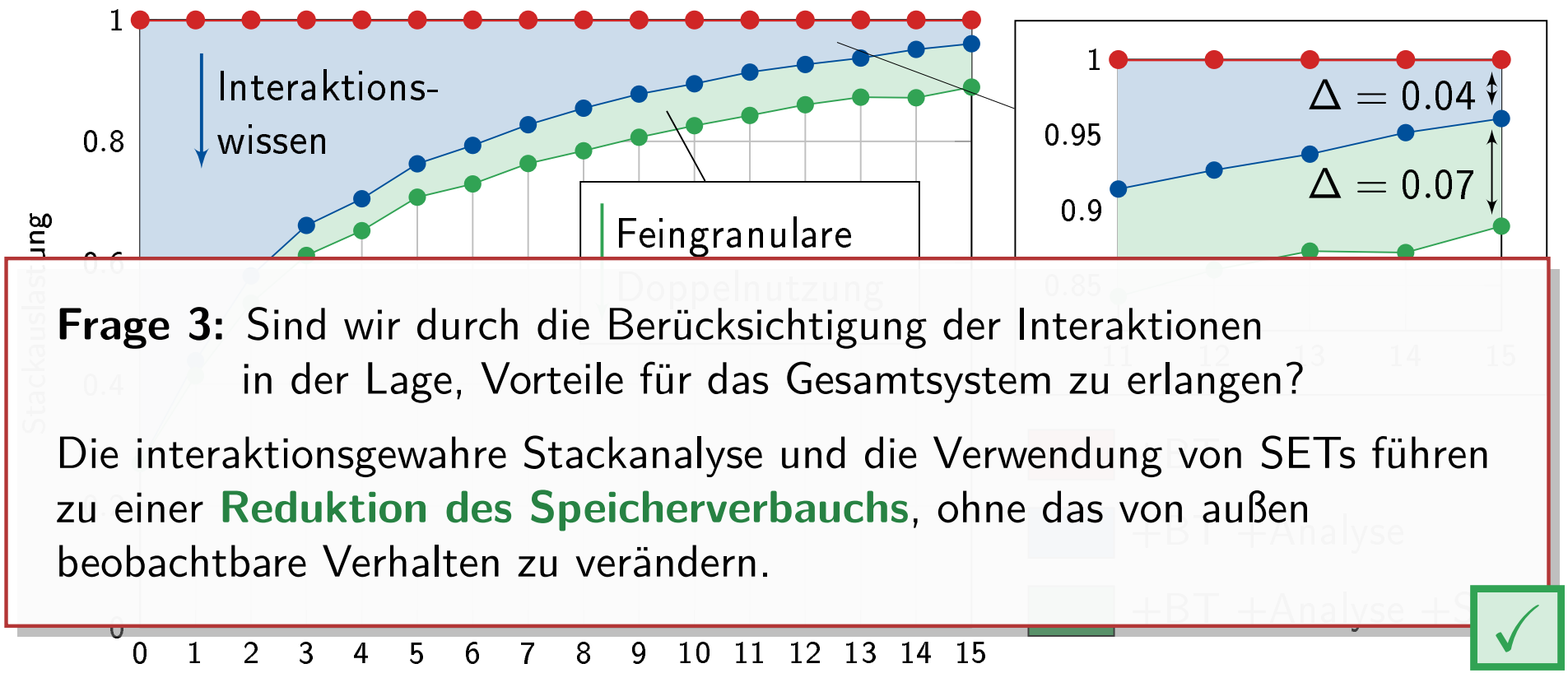
- Thread-granulare WCSC-Analyse zeigt trotz BTs keine Einsparungen.
- Interaktionswissen bringt deutlich engere obere Schranken.

Stackverbrauch mit wartendenden Threads



- Thread-granulare WCSC-Analyse zeigt trotz BTs keine Einsparungen.
- Interaktionswissen bringt deutlich engere obere Schranken.
- SETs mindern den Einfluss von passivem Warten nochmal deutlich.

Stackverbrauch mit wartendenden Threads



Frage 3: Sind wir durch die Berücksichtigung der Interaktionen in der Lage, Vorteile für das Gesamtsystem zu erlangen?

Die interaktionsgewahre Stackanalyse und die Verwendung von SETs führen zu einer **Reduktion des Speicherverbauchs**, ohne das von außen beobachtbare Verhalten zu verändern.

- Thread-granulare WCSC-Analyse zeigt trotz BTs keine Einsparungen.
- Interaktionswissen bringt deutlich engere obere Schranken.
- SETs mindern den Einfluss von passivem Warten nochmal deutlich.

Zusammenfassung

Extraktion, Analyse und Ausbeutung der Interaktionen

- Echtzeitanwendung und Betriebssystem interagieren wechselseitig
  - Syscalls beeinflussen Threadreihenfolge; Threads setzen Syscalls ab.
  - SSTG/GCFG bieten **integrierte Sicht auf die Interaktionen**.
  - **Automatisierte Werkzeuge** zur Extraktion und Ausbeutung stehen bereit.

Extraktion, Analyse und Ausbeutung der Interaktionen

- Echtzeitanwendung und Betriebssystem interagieren wechselseitig
  - Syscalls beeinflussen Threadreihenfolge; Threads setzen Syscalls ab.
  - SSTG/GCFG bieten integrierte Sicht auf die Interaktionen.
  - Automatisierte Werkzeuge zur Extraktion und Ausbeutung stehen bereit.
- Besserer/strikterer Nachweis der geforderten Systemfunktionalität
  - Interaktionsgewahre Antwortzeitanalyse **überwindet Threadgrenzen**.
  - Interaktionswissen führt zu **kleineren Schranken** beim Stackverbrauch.
  - **Erleichterte Verifikation** des Kernverhaltens anhand des SSTG

Extraktion, Analyse und Ausbeutung der Interaktionen

- Echtzeitanwendung und Betriebssystem interagieren wechselseitig
  - Syscalls beeinflussen Threadreihenfolge; Threads setzen Syscalls ab.
  - SSTG/GCFG bieten integrierte Sicht auf die Interaktionen.
  - Automatisierte Werkzeuge zur Extraktion und Ausbeutung stehen bereit.
- Besserer/strikterer Nachweis der geforderten Systemfunktionalität
  - Interaktionsgewahre Antwortzeitanalyse überwindet Threadgrenzen.
  - Interaktionswissen führt zu kleineren Schranken beim Stackverbrauch.
  - Erleichterte Verifikation des Kernverhaltens anhand des SSTG
- Optimierung des Ressourcenverbrauchs entlang der Interaktionen
  - Semi-Extended Tasks brechen mit der monolitische Sicht auf den Stack und erlauben die **feingranulare Mehrfachbelegung** der Reservierungen.
  - Minimierung von RTOS-Laufzeit und -Interferenz durch **interaktionsspezifische Spezialisierung** der Ausführungsplattform.

Extraktion, Analyse und Ausbeutung der Interaktionen

- Echtzeitanwendung und Betriebssystem interagieren wechselseitig
  - Syscalls beeinflussen Threadreihenfolge; Threads setzen Syscalls ab.
  - SSTG/GCFG bieten **integrierte Sicht auf die Interaktionen**.
  - Automatisierte Werkzeuge zur Extraktion und Ausbeutung stehen bereit.
- Besserer/strikterer Nachweis der geforderten Systemfunktionalität
  - Interaktionsgewahre Antwortzeitanalyse **überwindet Threadgrenzen**.
  - Interaktionswissen führt zu kleineren Schranken beim Stackverbrauch.
  - Erleichterte Verifikation des Kernverhaltens anhand des SSTG
- Optimierung des Ressourcenverbrauchs entlang der Interaktionen
  - Semi-Extended Tasks brechen mit der monolitische Sicht auf den Stack und erlauben die feingranulare Mehrfachbelegung der Reservierungen.
  - Minimierung von RTOS-Laufzeit und -Interferenz durch **interaktionsspezifische Spezialisierung** der Ausführungsplattform.