SysWCET: Ende-zu-Ende-Antwortzeiten für OSEK-Systeme

Christian Dietrich, Peter Wägemann

dietrich@sra.uni-hannover.de
waegemann@cs.fau.de

ESE Kongress

6. Dezember 2017

Computer Science 4
Operating Systems and Distributed Systems
FAU

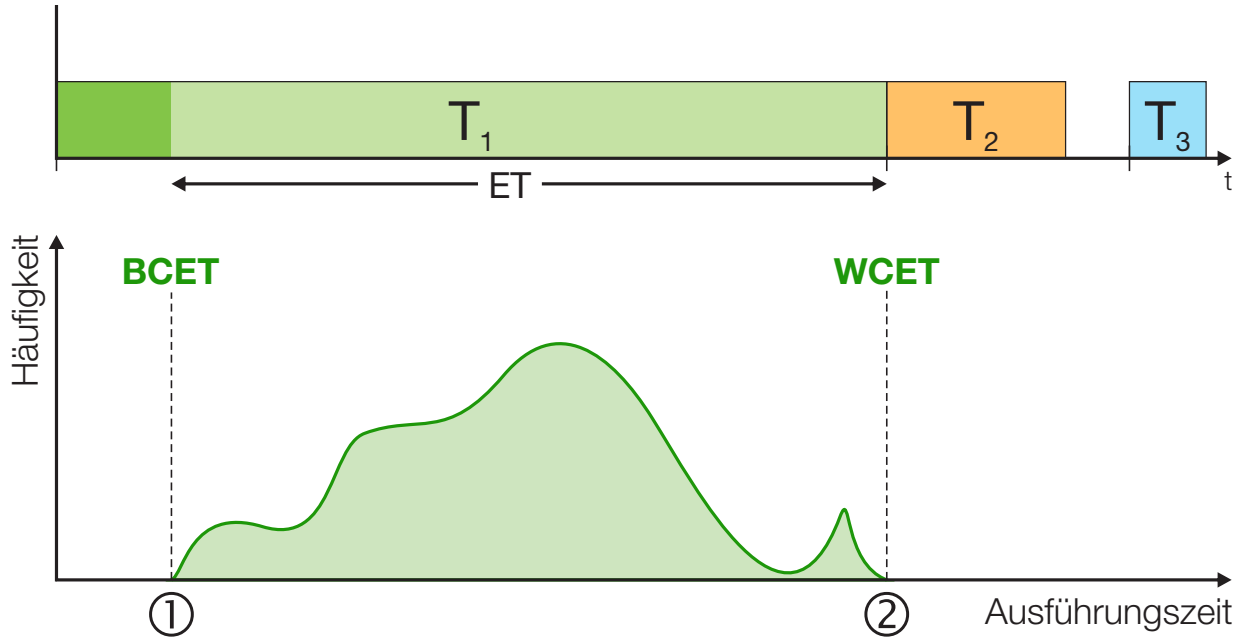
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

ISE
SRA

Institute of Systems Engineering
System and Computer Architecture
Leibniz Universität Hannover

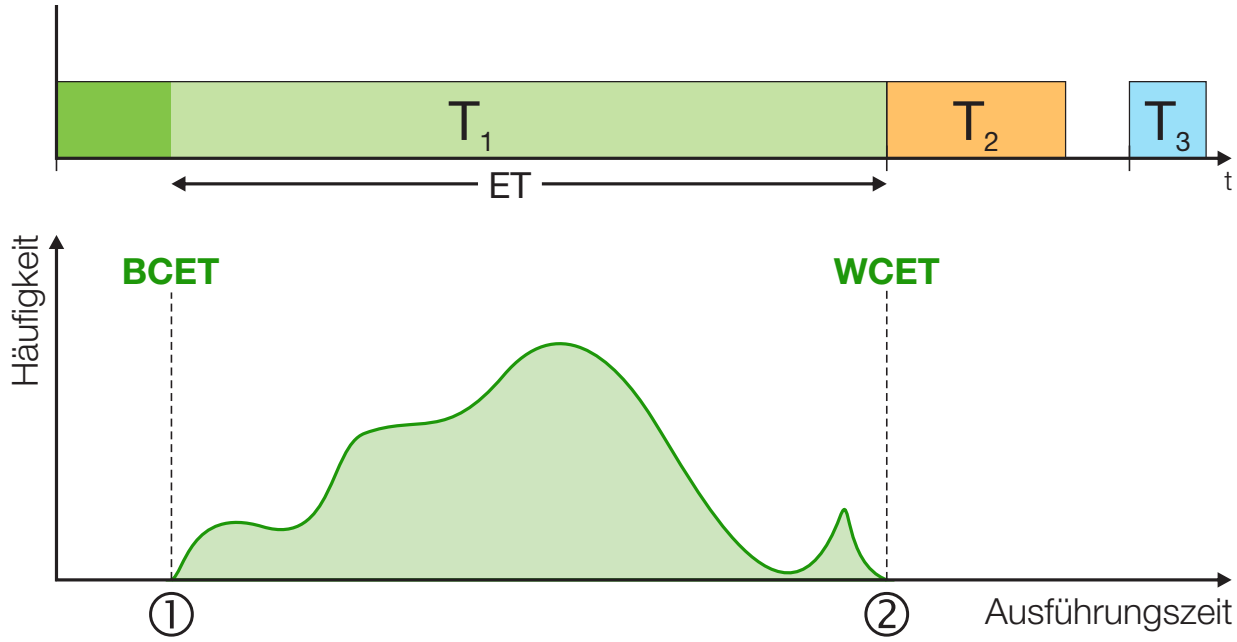
Leibniz
Universität
Hannover

Die **maximale** Ausführungszeit



- *Maximale Ausführungszeit*
  - Worst-Case Execution Time (*WCET*)
  - Unabdingbares Maß für *zulässigen Ablaufplan*
- Tatsächliche Ausführungszeit zwischen *BCET* und *WCET*

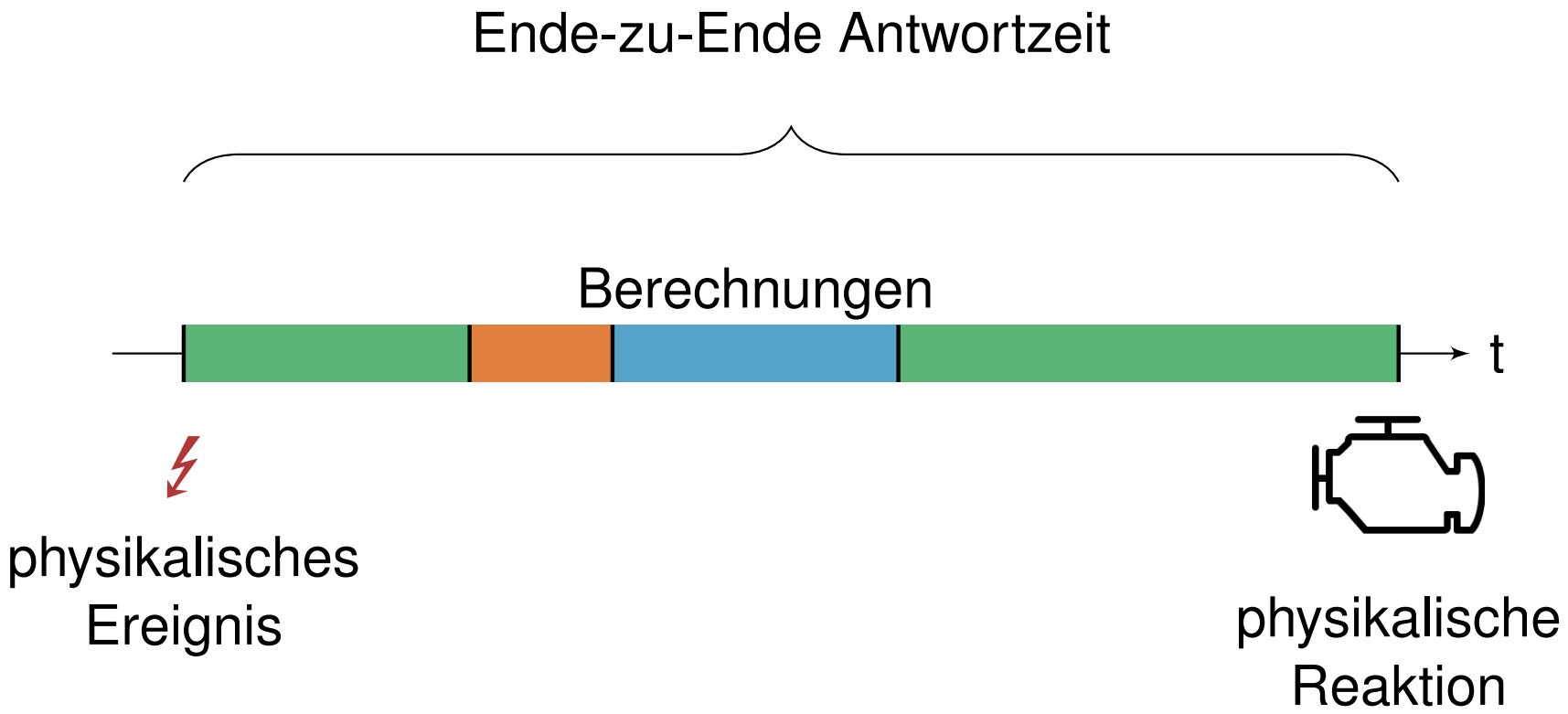
Die **maximale** Ausführungszeit



- *Maximale Ausführungszeit*
  - Worst-Case Execution Time (*WCET*)
  - Unabdingbares Maß für *zulässigen Ablaufplan*
- Tatsächliche Ausführungszeit zwischen *BCET* und *WCET*
- Viele Systeme (z.B. OSEK) erlauben **Ereignisse & Verdrängungen**

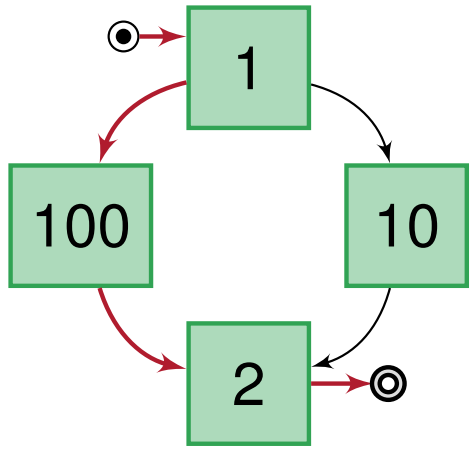
☞ Analyse der **End-zu-Ende Antwortzeiten** notwendig

Antwortzeit eines Kontrollsystems



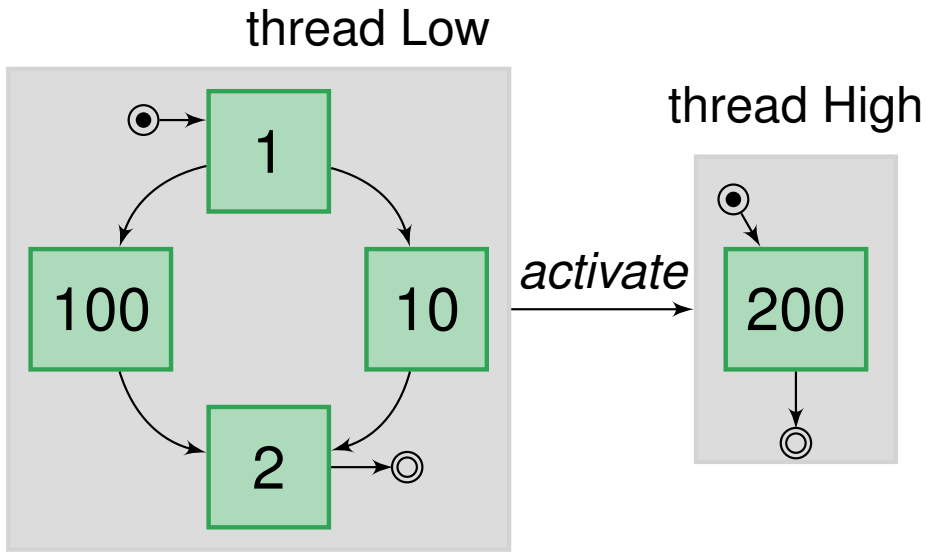
- Beispiel
  - Ereignis: Drücken eines Tasters
  - Ergebnis: Setzen von Stellwert eines Ventils
- Verdrängungen von Aufgaben mit höherer Priorität

Schlimmstmöglichte Antwortzeit eines Systems (WCRT)



Worst-Case Response Time: 103 Zyklen

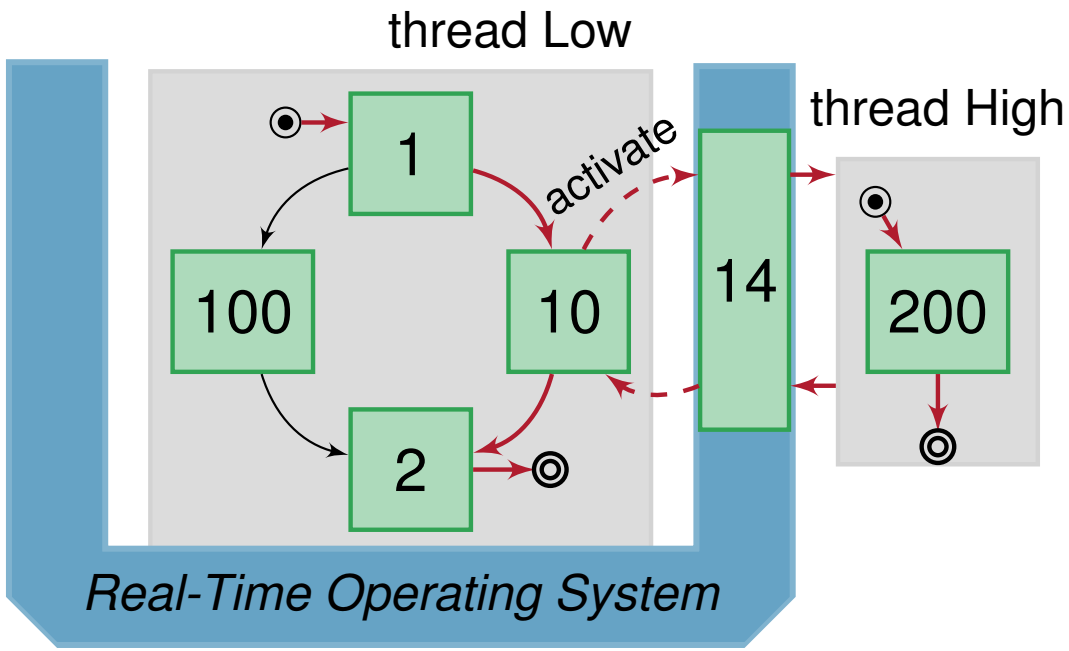
Schlimmstmöglichte Antwortzeit eines Systems (WCRT)



Worst-Case Response Time: 103 + 200 + t(RTOS) Zyklen?

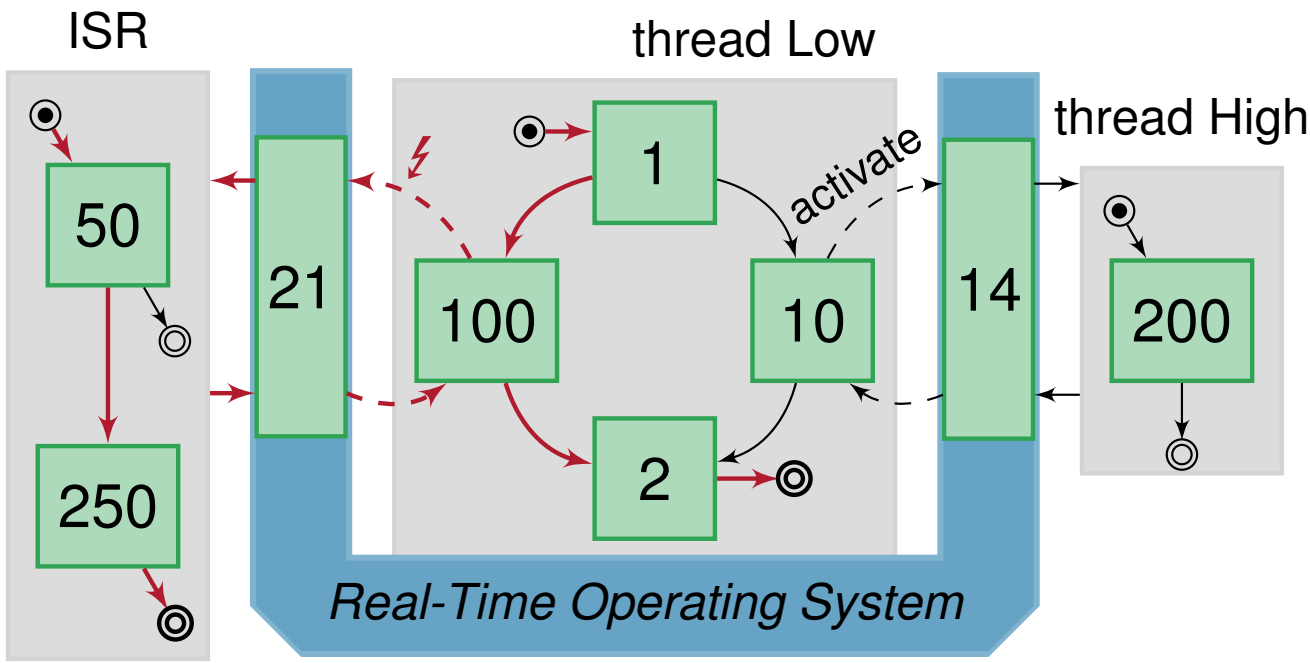
- Real-Time Operating System (RTOS)

Schlimmstmöglichte Antwortzeit eines Systems (WCRT)



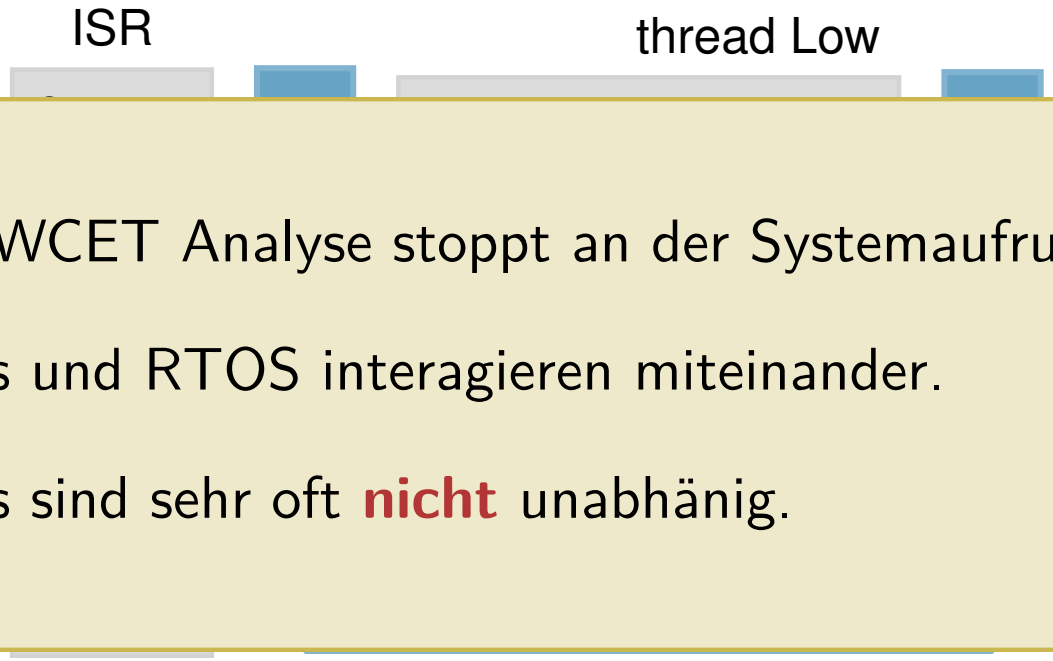
Worst-Case Response Time: 241 Zyklen

Schlimmstmöglichte Antwortzeit eines Systems (WCRT)



Worst-Case Response Time: 445 Zyklen

Schlimmstmöglichte Antwortzeit eines Systems (WCRT)



Lokale WCET Analyse stoppt an der Systemaufrufschnittstelle.

Threads und RTOS interagieren miteinander.

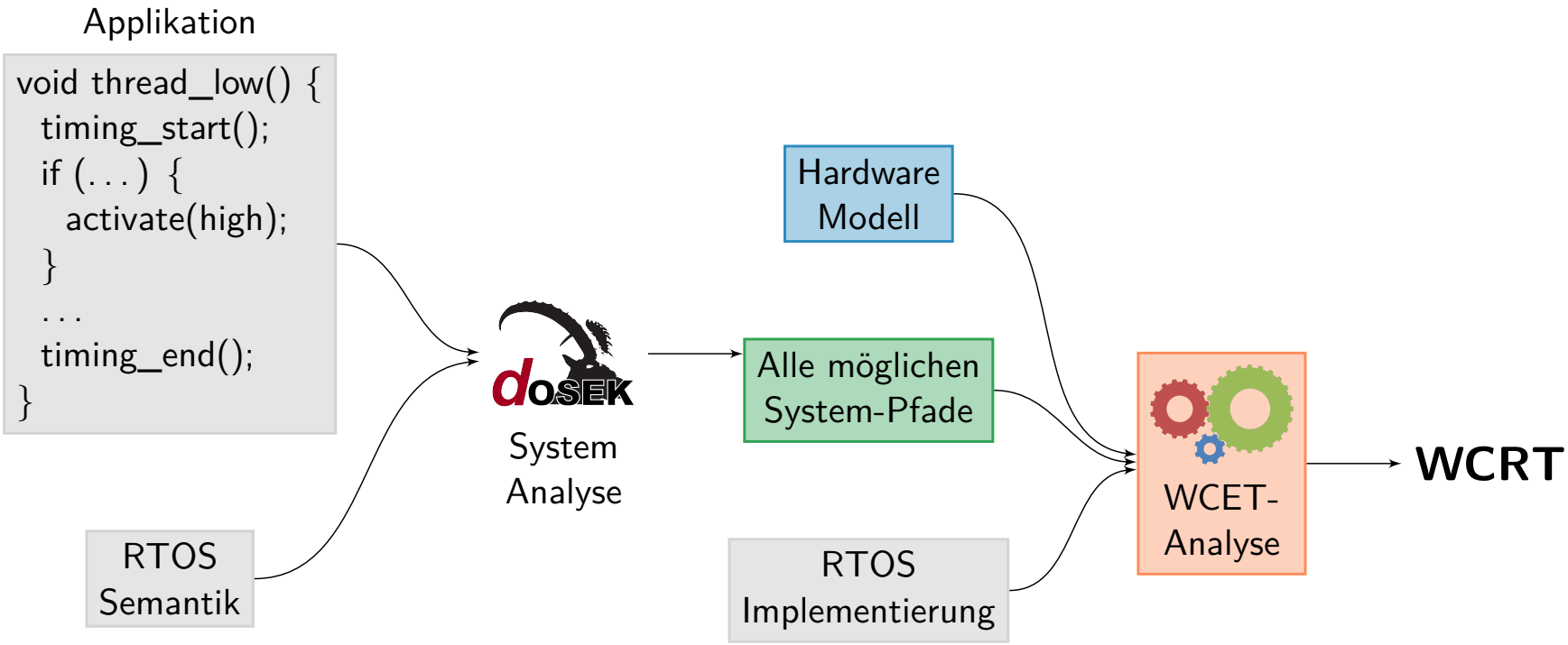
Threads sind sehr oft **nicht** unabhänig.

Worst-Case Response Time: 445 Zyklen

Probleme mit zusammengesetzer Antwortzeitanalyse

- Die verbreiteten Ansätze arbeiten kompositionell
  - Eine WCET wird für jede Komponente pessimistisch in Isolation bestimmt
  - Aggregation von WCETs anhand ihrer möglichen Interaktionen
- Jede WCET für sich muss pessimistisch sein um Safety zu bieten
  - Immer den längsten Pfad in jedem Thread
  - Immer die längste Dauer durch den Kern beim Systemaufruf
- $\Rightarrow$ Integrierte Formulierung einer systemweiten WCRT Analyse
  - Erfassen der Maschienenebene und der Scheduling Analyse
  - RTOS Verhalten und Umweltmodell müssen integriert werden
  - Formulierung thread-übergreifender Flow-Facts (z.B. gegenseitiger Ausschluss)

Die Werkzeugkette im Überblick



C. Dietrich, P. Wägemann, P. Ulbrich, D. Lohmann

SysWCET: Whole-System Response-Time Analysis for Fixed-Priority Real-Time Systems

23rd Real-Time and Embedded Technology and Applications Symposium **(IEEE RTAS '17)**

Gliederung

Gliederung

Ereignisgesteuerte Echtzeitsysteme

- Systemmodell
  - Ereignisgesteuertes Ausführungsmodell: Threads, ISRs, etc.
  - Fixed-priority Scheduling Semantik
  - Kenntnis über das statisches Systemwissen
    - Kernobjekte (Thread, Locks, Periodische Signale) und ihre Konfiguration
    - Anwendungsstruktur (CFG) und Systemaufrufe (Ort, Argumente)

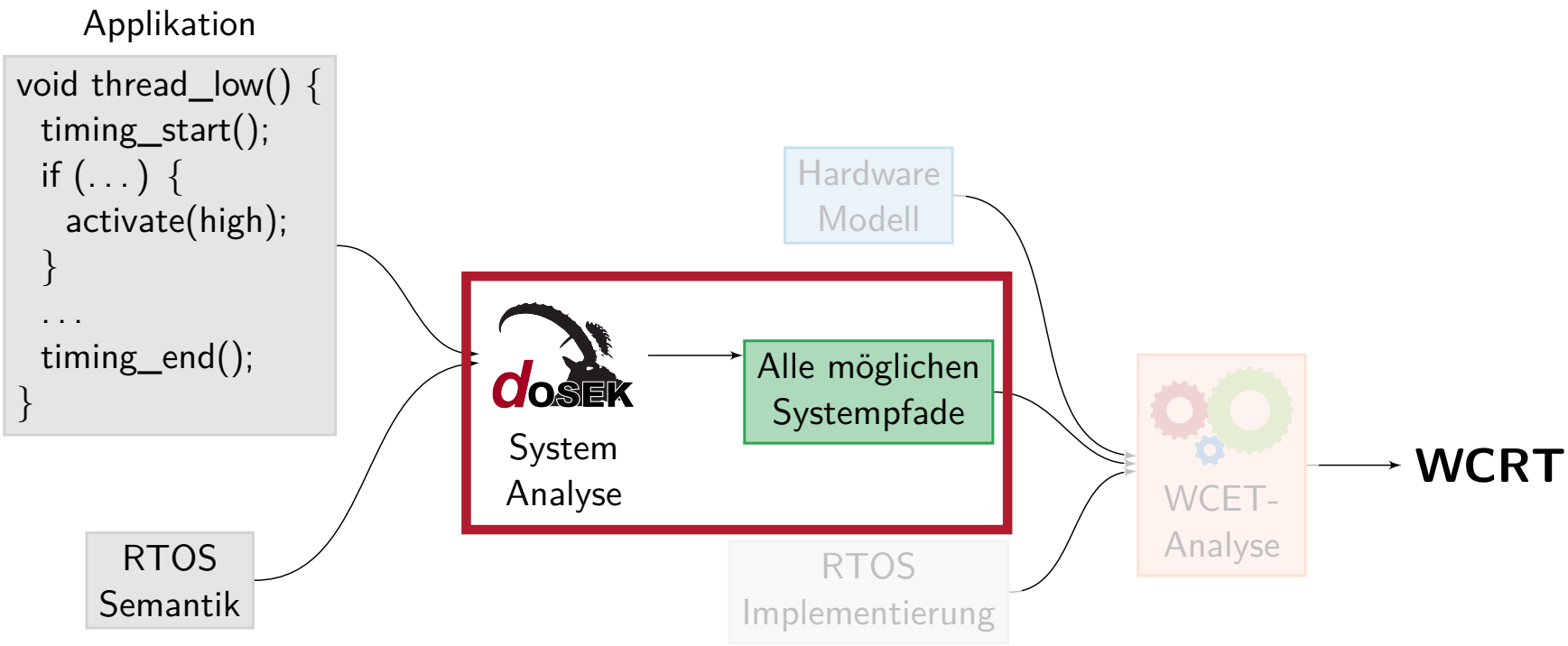
Ereignisgesteuerte Echtzeitsysteme

- Systemmodell
  - Ereignisgesteuertes Ausführungsmodell: Threads, ISRs, etc.
  - Fixed-priority Scheduling Semantik
  - Kenntnis über das statisches Systemwissen
    - Kernobjekte (Thread, Locks, Periodische Signale) und ihre Konfiguration
    - Anwendungsstruktur (CFG) und Systemaufrufe (Ort, Argumente)

AUTOSAR

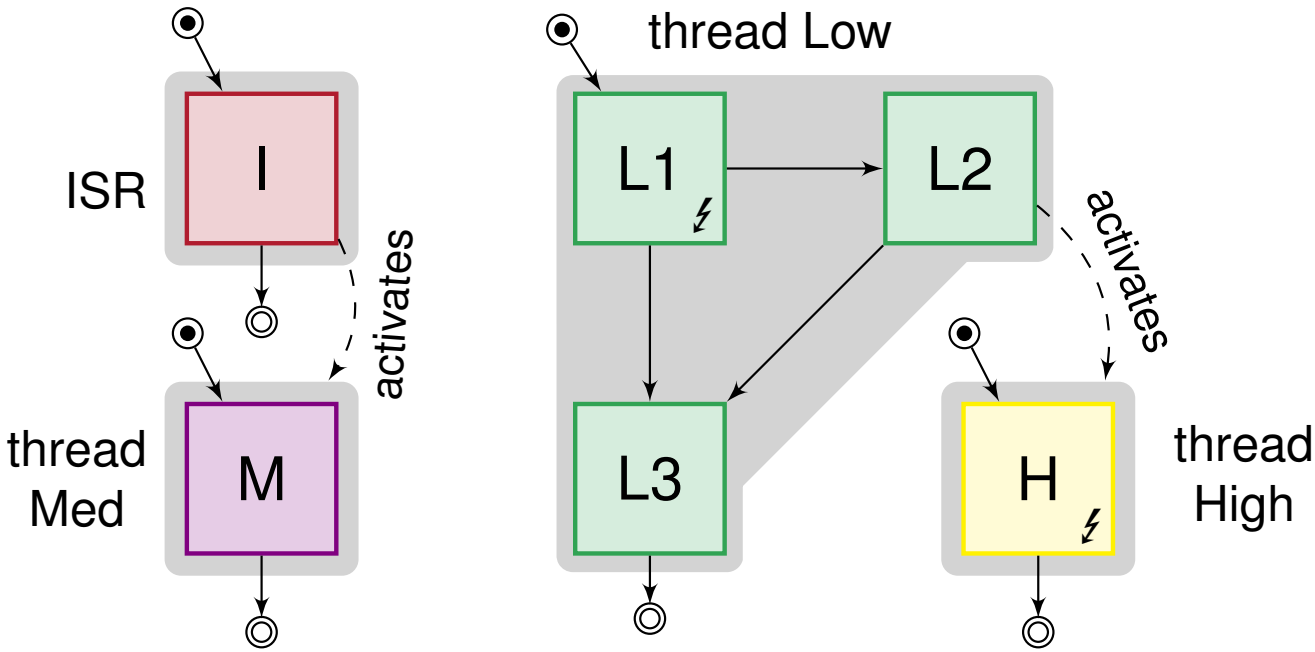
- Systeme die unserem Systemmodell entsprechen: OSEK, AUTOSAR
  - Industriestandard aus der Automobilbranche
  - Statische Konfiguration zur Übersetzungszeit
  - Fixed-priority Scheduling mit Threads und ISRs
  - Stapelbasiertes Priority-Ceiling-Protokoll (PCP) für Locks

Bestimmung systemweiter Programmpfade

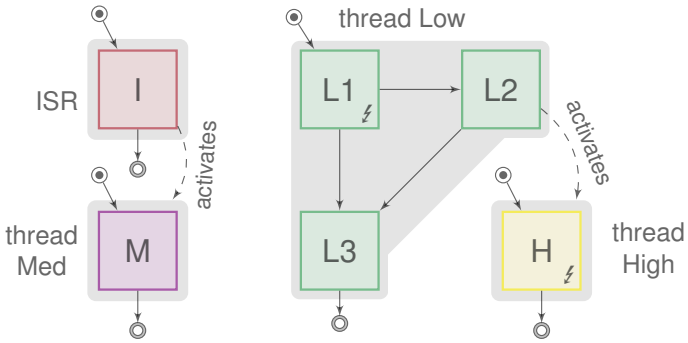


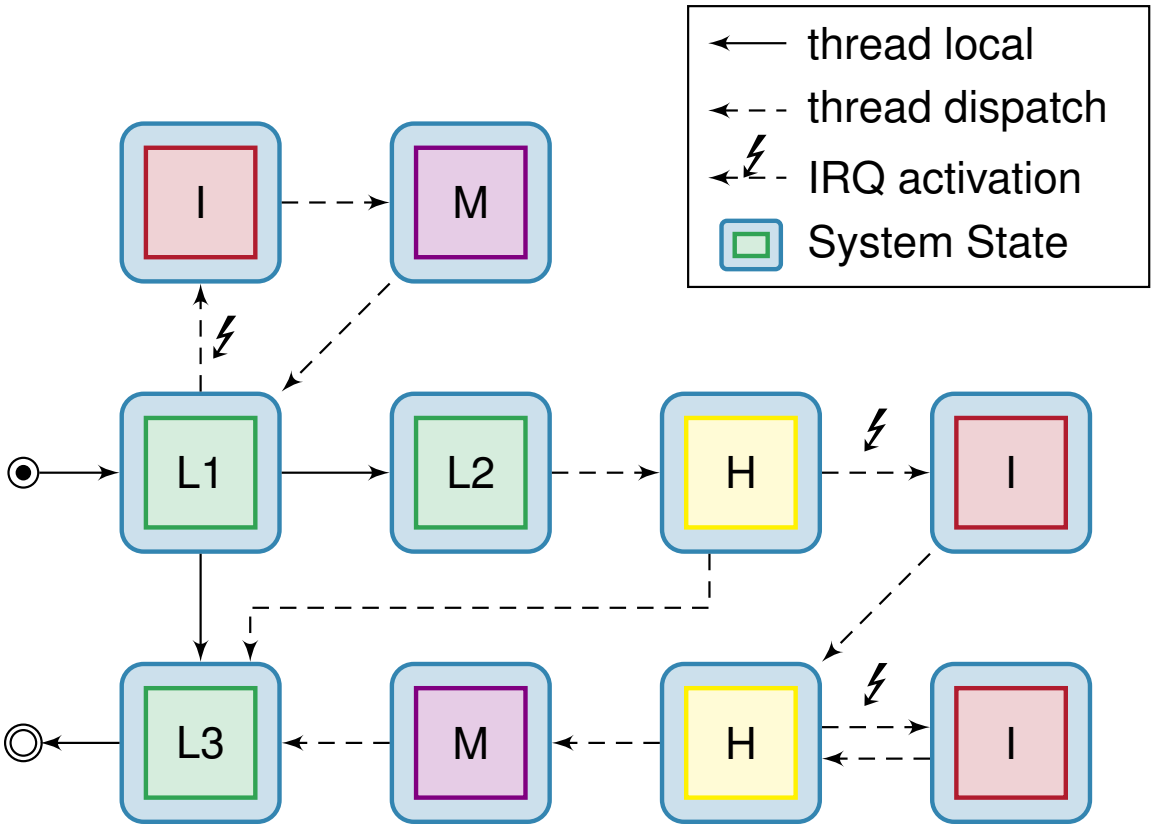
- Bestimmung aller möglichen Programmpfade des Gesamtsystems
- Berücksichtigung von
  1. Applikationscode
  2. RTOS Semantik

Operating-System State Transition Graph



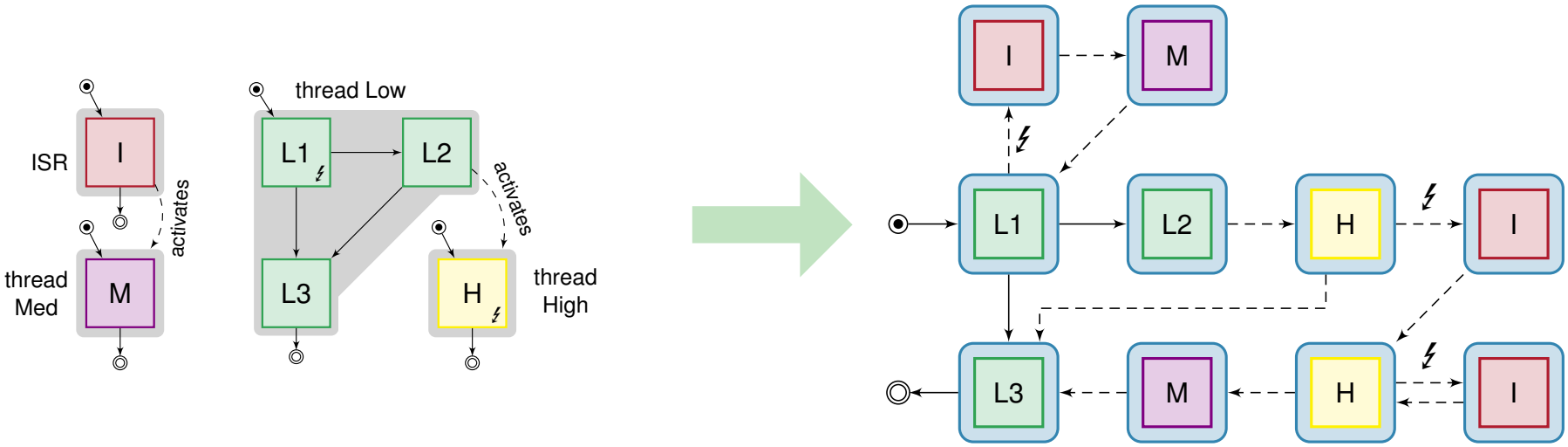
Operating-System State Transition Graph





Operating-System State Transition Graph

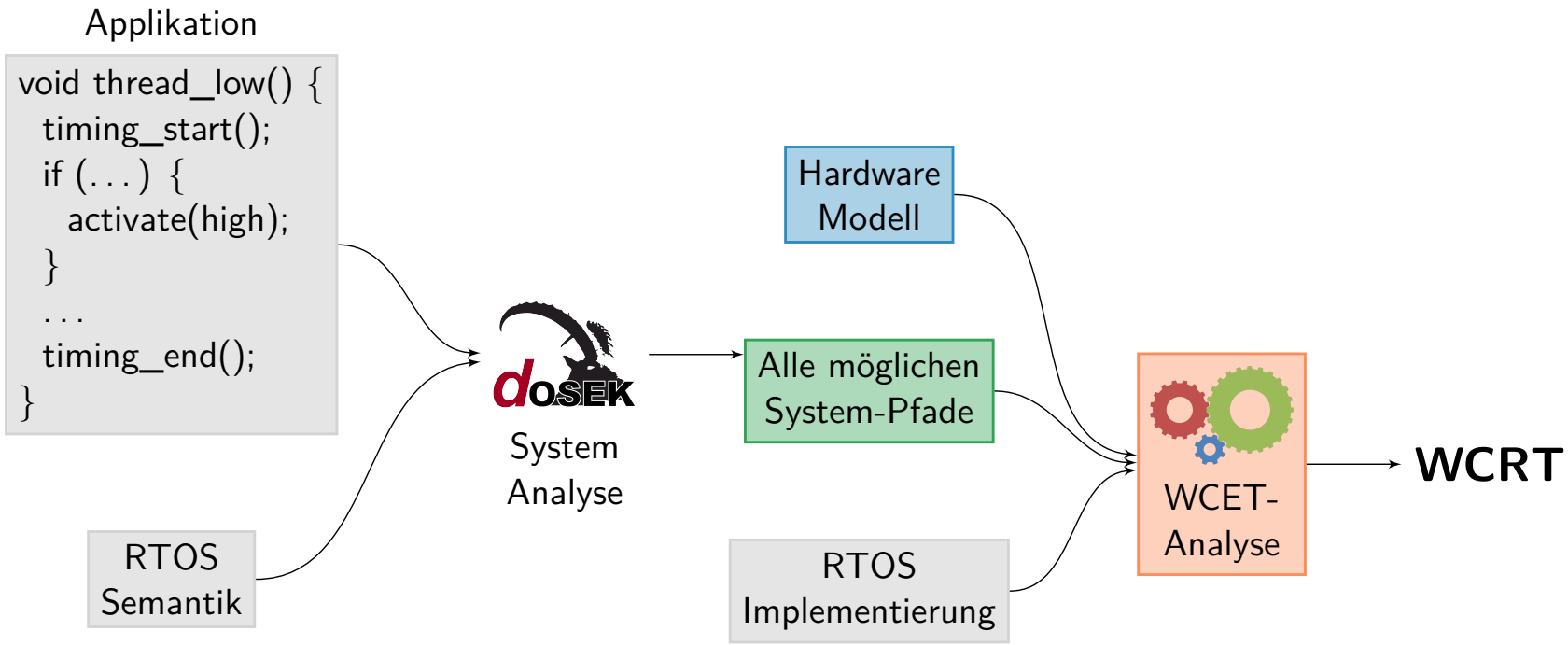
LCTES'15
TECS'17



- Explizite Aufzählung aller Systemzustände
- Betriebssystemkern $\leftrightarrow$ Anwendung $\leftrightarrow$ Umgebungsmodell
- Beinhaltet: Interruptaktivierungen, Lockprotokolle, Verdrängung

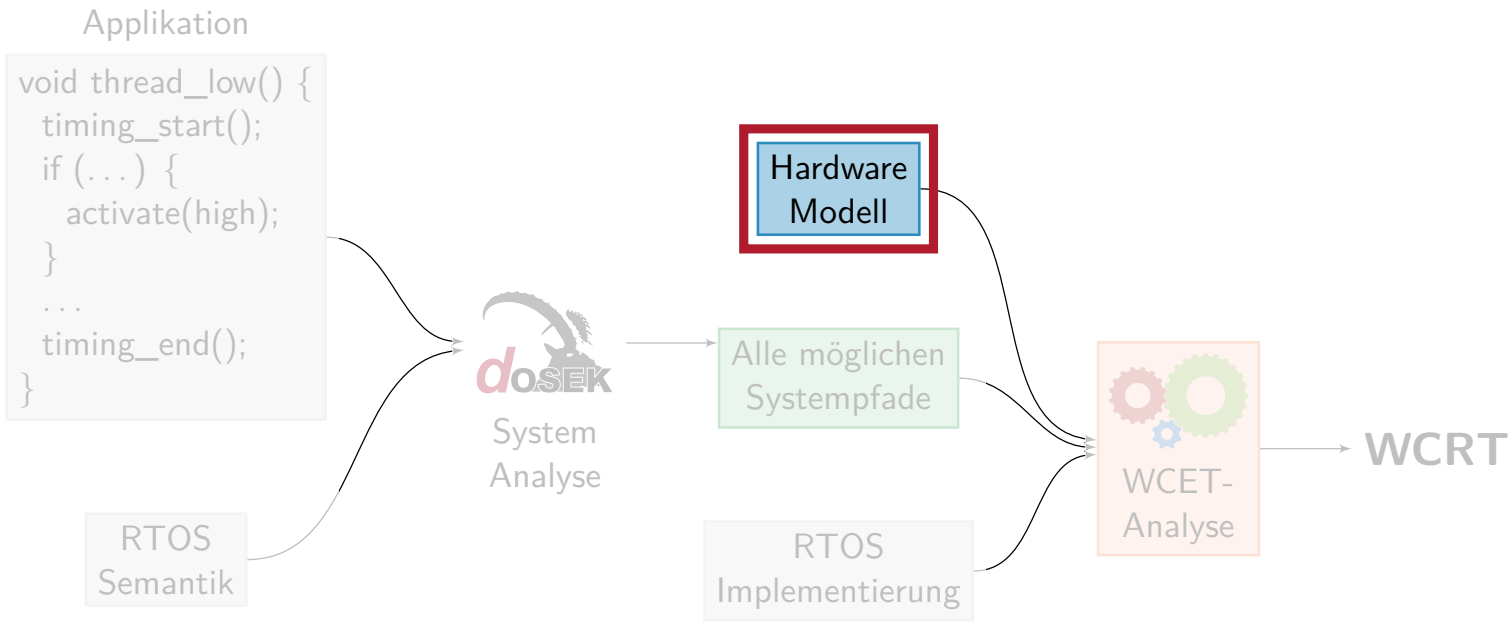
Gliederung

Schritte der WCET-Analyse



- Bestandteile der WCET-Analyse (IPET)
  1. Pfad-Analyse: Hardware-unabhängige Analyse
  2. Hardware-Analyse: abhängig von Zielplattform

Schritte der WCET-Analyse



- Bestandteile der WCET-Analyse (IPET)
  1. Pfad-Analyse: Hardware-unabhängige Analyse
  2. Hardware-Analyse: abhängig von Zielplattform

Ausführungszeit von Instruktionen

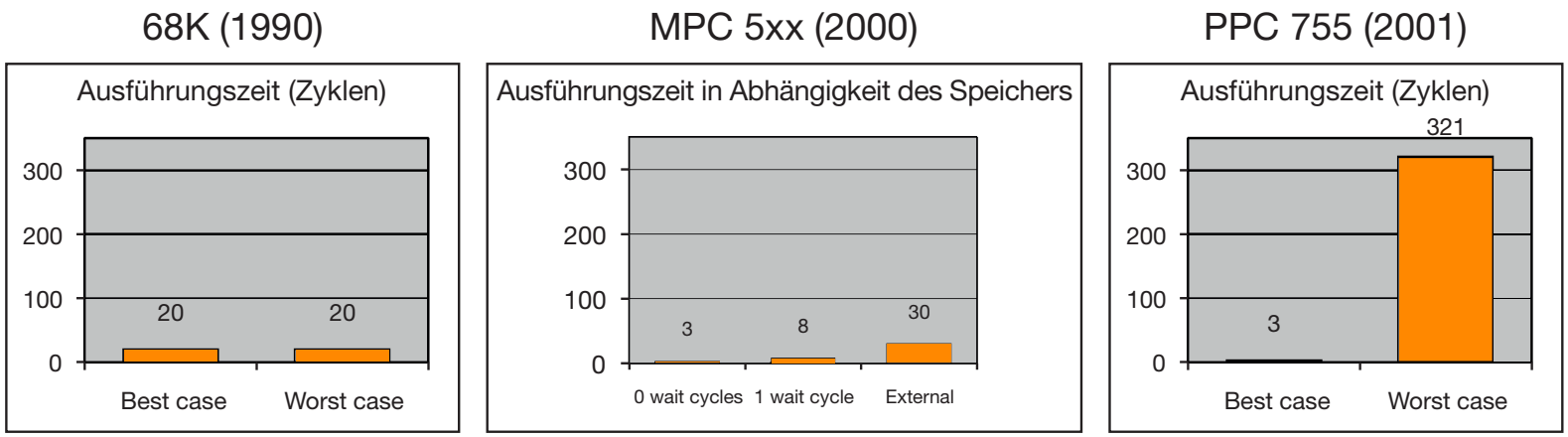
- ☞ Ausführungszeit von Maschineninstruktionen ist essentiell
- Beispiel

```
/* x = a + b */
LOAD r2, _a
LOAD r1, _b
ADD r3,r2,r1
```

Ausführungszeit von Instruktionen

- ☞ Ausführungszeit von Maschineninstruktionen ist essentiell
- Beispiel

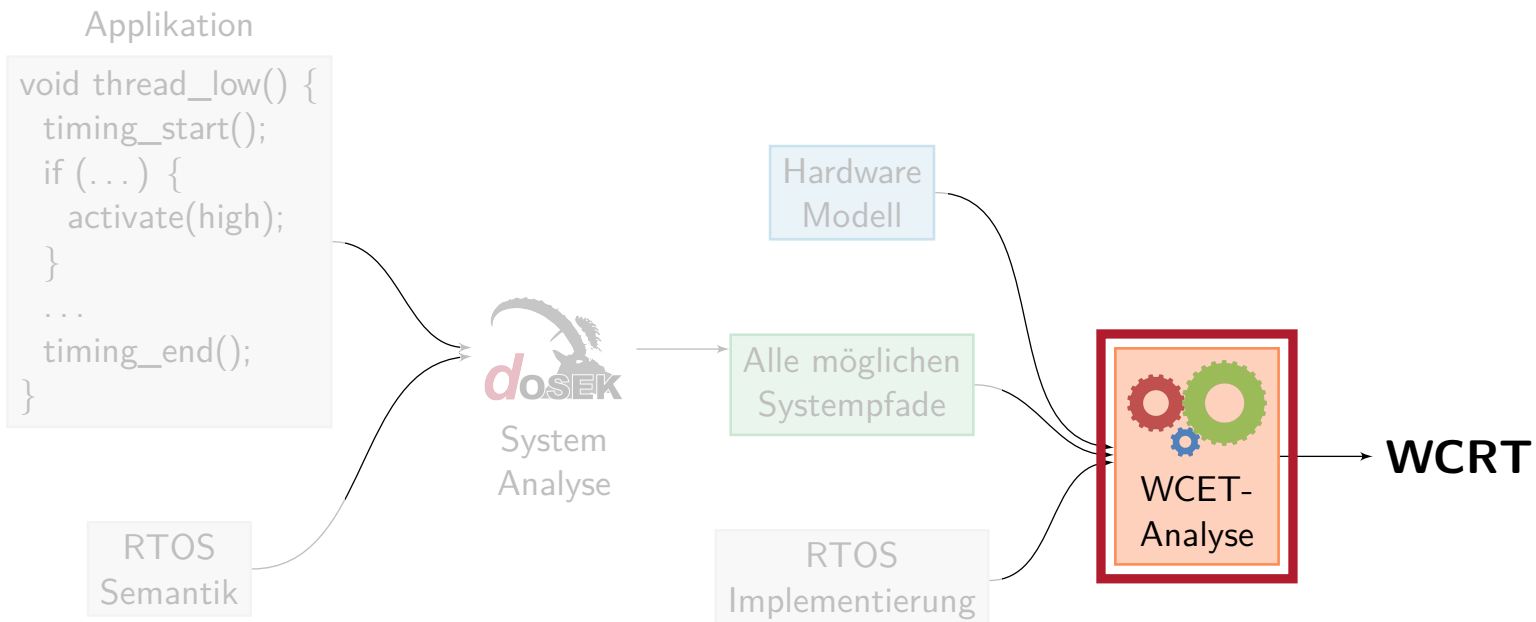
```
/* x = a + b */
LOAD r2, _a
LOAD r1, _b
ADD r3,r2,r1
```



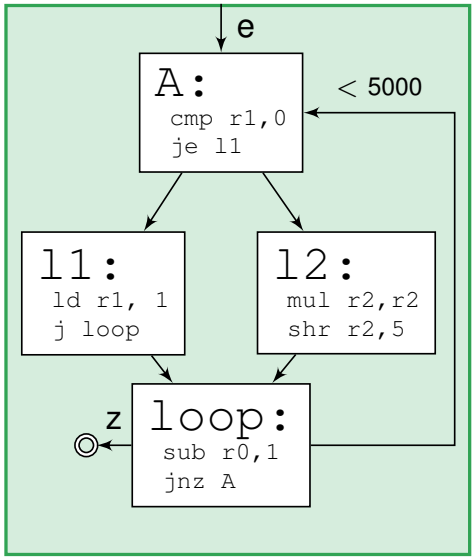
Quelle: C. Ferdinand

- Hardware-Analyse
  - Umfangreich untersucht: Standard-Techniken
  - Ergebnis: WCET für Sequenzen (Basisblöcke)
- ☞ **SysWCET**: *Pfad-Analyse* vollständiger Systeme

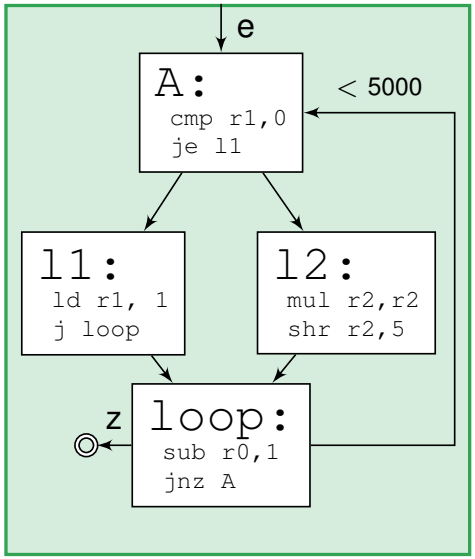
WCET-Analyse mittels Impliziter Pfadaufzählung



Darstellung von Pfaden



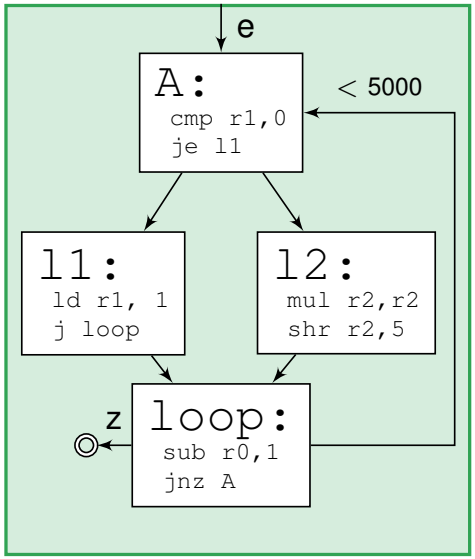
Darstellung von Pfaden



Problem: Erfordert **explizite Aufzählung** aller Pfade

- Auf Ebene der Systemzustände handhabbar ✓
- Auf Maschinencode-Ebene algorithmisch nicht möglich (Beispiel: 2^{5000})

Darstellung von Pfaden



Problem: Erfordert **explizite Aufzählung** aller Pfade

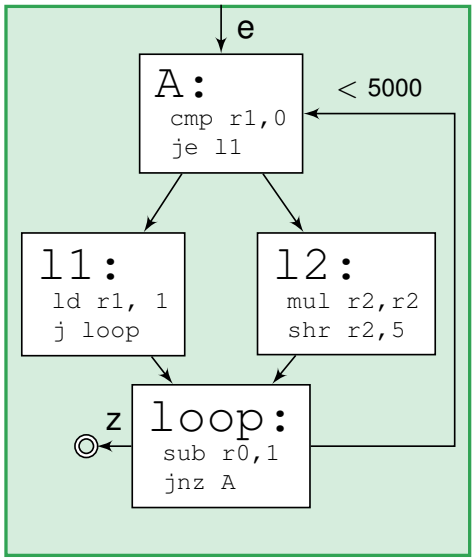
- Auf Ebene der Systemzustände handhabbar ✓
- Auf Maschinencode-Ebene algorithmisch nicht möglich (Beispiel: 2^{5000})

Lösung: Vereinfachung der konkreten Pfadsemantik

→ Abstraktion und **Abbildung auf ein Flussproblem**

- Flussprobleme sind mathematisch gut untersucht
- **Implicit Path Enumeration Technique (IPET)** [Puschner 1990]

Abbildung auf Flussproblem



- **Flusserhaltung** für Basisblöcke (eingehender = ausgehender Fluss)
- **Nebenbedingungen** für Variablen aus Kontrollfluss ableitbar

1. `A = e + loop`
2. `A = l1 + l2`
3. `...`

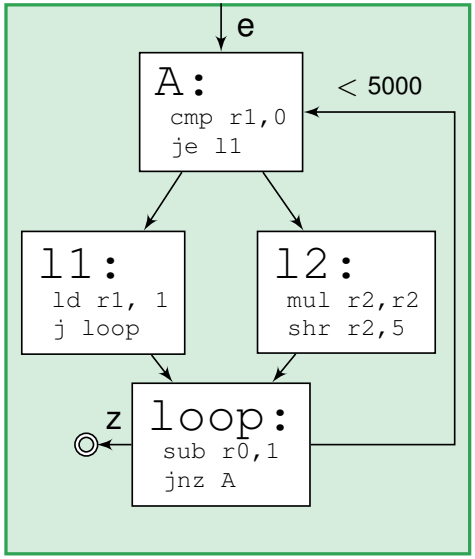
- **Schleifeniteration** benötigen obere Schranke

4. `A < 5000 + e`

- Einmalige Analyse des Fragments

5. `e = 1; z = 1`

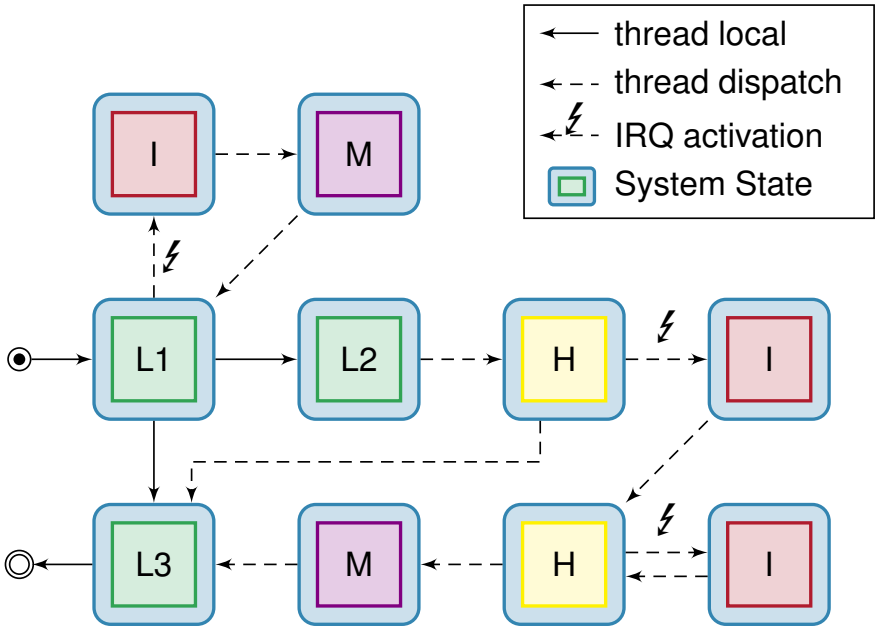
Abbildung auf Flussproblem



- **Kosten der Blöcke** aus Hardware-Analyse
  - `A`: 4 Zyklen
  - `l1`: 2 Zyklen
  - ...
- Zielfunktion (unter Einhaltung aller Nebenbedingungen)
  - Maximiere Summe von (Ausführungshäufigkeit · Kosten) aller Blöcke
  - `max: A·4 + l1·2 + ...`
  - Lösung mittels mathematischer Optimierungsprogramme (`Gurobi`)

Gliederung

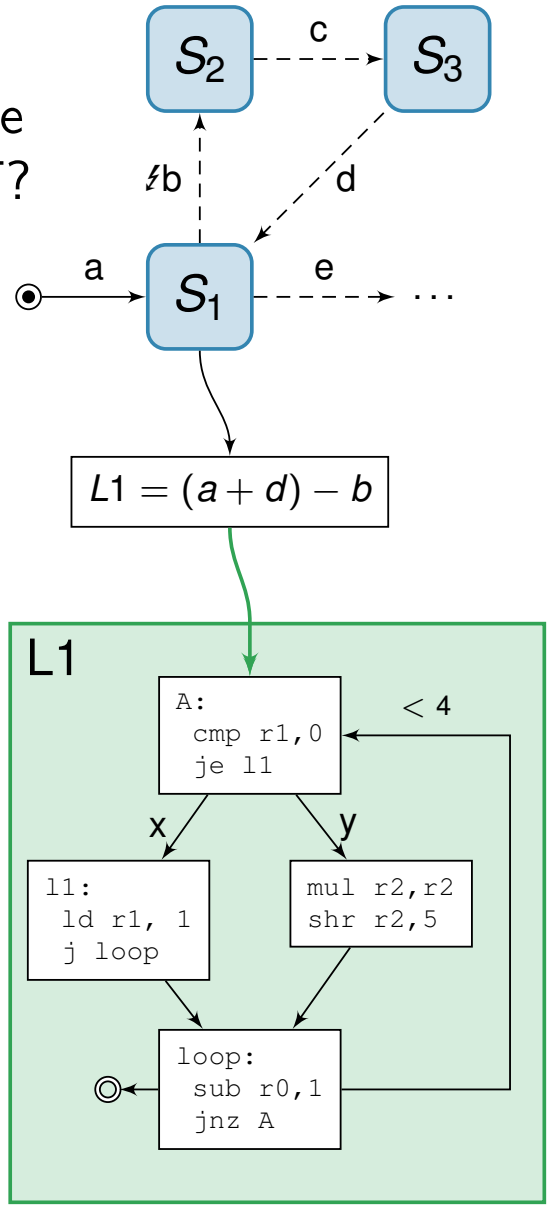
SysWCET IPET Formulierung



- SysWCET Idee kurz zusammengefasst
  1. IPET für den State Transition Graph: *Zustandshäufigkeit*
  2. Ausserdem: ein IPET Fragment für jeden Programmblock
  3. Ableitung einer Blockhäufigkeit aus der Zustandshäufigkeit

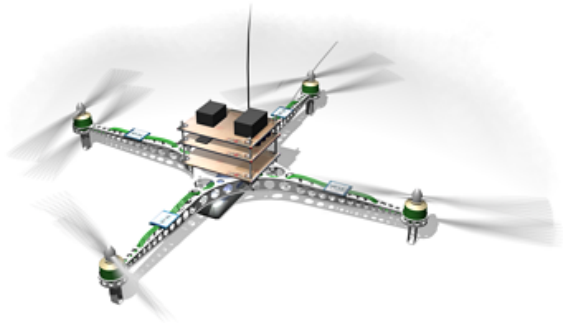
Geschichtete IPET Kontruktion

- **Systemschicht**
  - IPET-Variablen für Zustände & Zustandsübergänge
  - Wie häufig ist das System in S_1 im Fall der WCRT?
  - Umweltmodell beschränkt die Häufigkeit von IRQs $(b + \ldots) \cdot 1000 < T_{WCRT}$
- **Leimschicht**
  - Blockhäufigkeiten aus Zustandshäufigkeiten
  - Vollständige IRQ-iret Zyklen werden abgezogen
- **Maschinenschicht**
  - Ein IPET-Fragment aus jedem Codeblock
  - RTOS Code wird gleich behandelt
  - Blockhäufigkeiten aktivieren das Fragment
  - Eine IPET-Formulierung: ermöglicht Flow Facts innerhalb und zwischen Codeblöcken



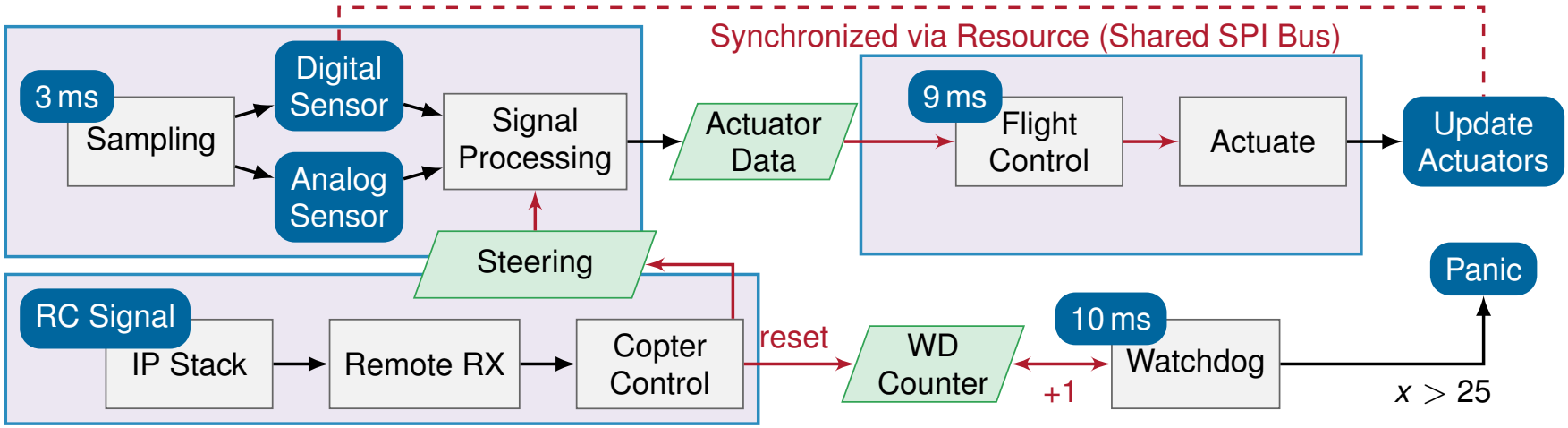
Gliederung

Hardware-Modell & Evaluationsszenario: i4Copter



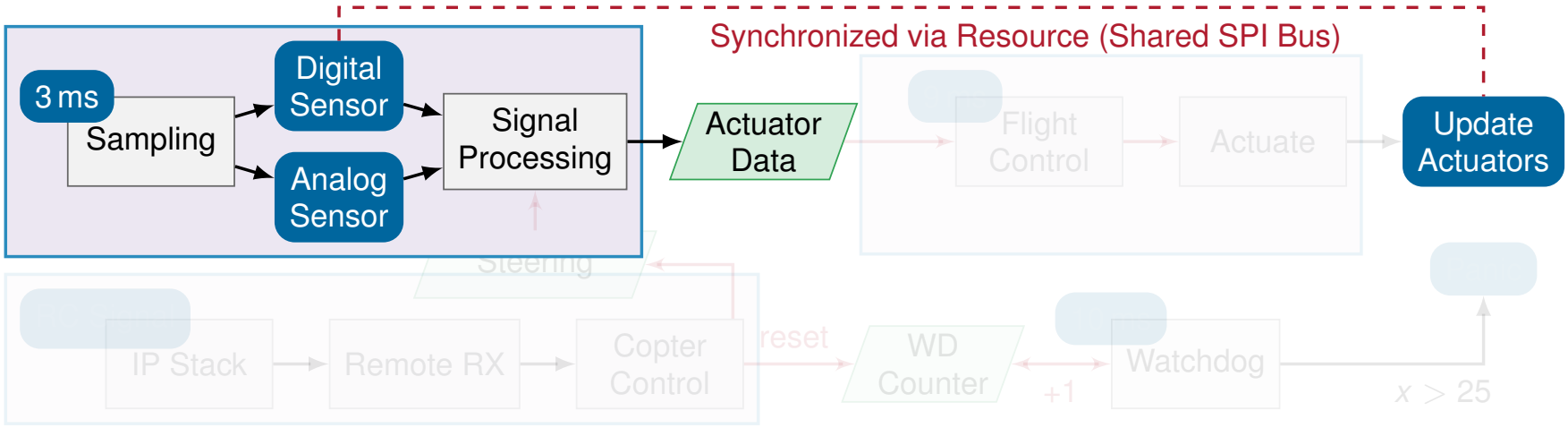
- Aktueller Stand: einfaches Prozessormodell
  - Jeder Befehl hat konstante Ausführungszeit
    keine Pipelines, keine Caches
  - SysWCET ist mit komplexeren Modellen kombinierbar
- `dOSEK` als Betriebssystemimplementierung
  - Generativer Ansatz mit detaillierten Anwendungsanalysen
  - Exportiert einen (partiellen) OS State Transition Graphen
- i4Copter: eingebettetes safety-critical Kontrollsystem
  - In Zusammenarbeit mit Siemens Corporate Technology entwickelt
  - 11 Threads, 3 periodische Signale, 1 Interrupt, Locks, Interruptsperren
  - Nur Threadinteraktionen, kein Signalverarbeitungscode

Evaluation



- Automatische SysWCET WCRT Analyse
  - Code Annotationen markieren Start- und Endpunkte
  - `dOSEK` berechnet OS State-Transition Graphen
  - `Platin` WCET Analysetool baut und löst IPET (mit `Gurobi`)
- Vergleich mit manuelle kompositioneller WCRT Analyse
  - Berechne Thread WCETs in Isolation mittels `Platin`
  - Manuelles Zusammenrechnen der Werte anhand der Systemspezifikation

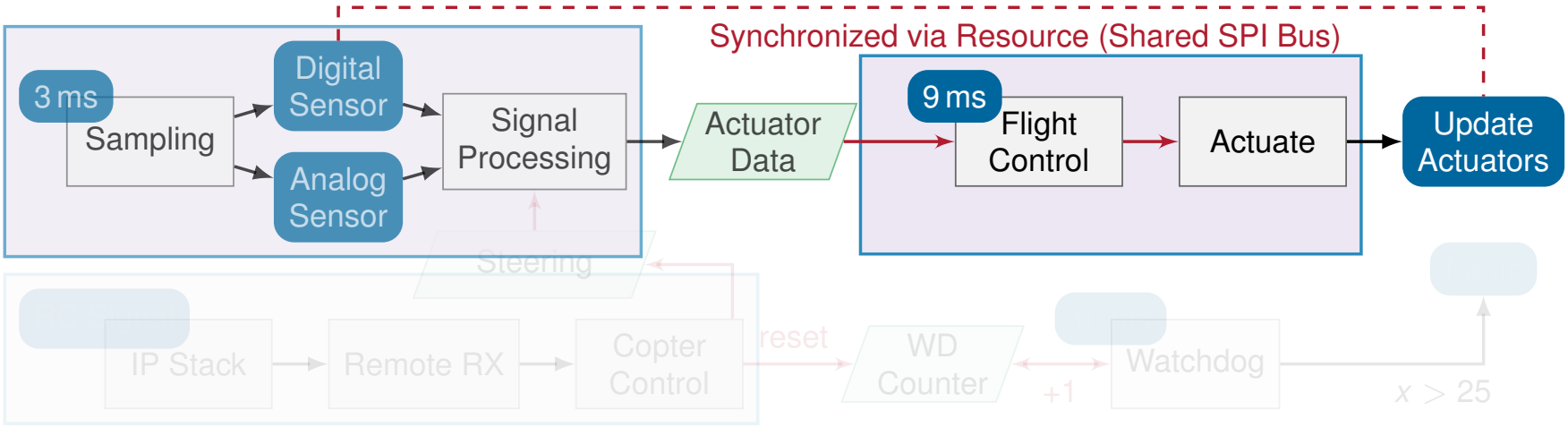
Evaluation



- Automatische SysWCET Analyse ⇔ Manuelle WCRT Analyse

			WCRT	
	States	Solve Time	SysWCET	Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.

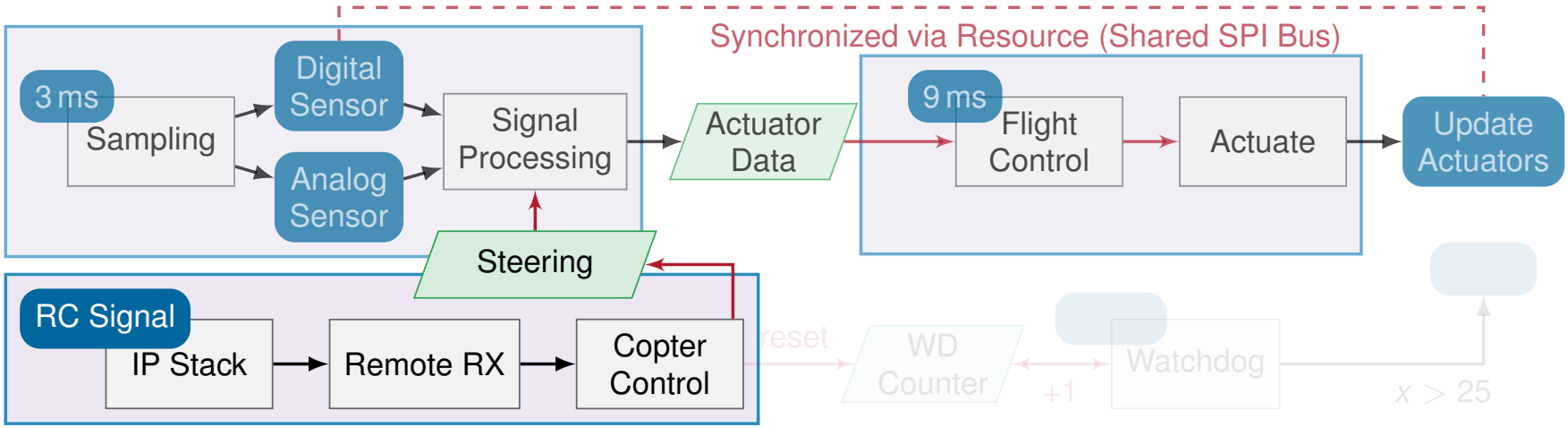
Evaluation



- Automatische SysWCET Analyse ⇔ Manuelle WCRT Analyse

	States	Solve Time	WCRT	
			SysWCET	Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.
Flight Control	7690	161.56 s	9279 cyc.	10057 cyc.

Evaluation



- Automatische SysWCET Analyse ⇔ Manuelle WCRT Analyse

	States	Solve Time	WCRT SysWCET	WCRT Manual
Signal Gathering	9506	14.72 s	5626 cyc.	6286 cyc.
Flight Control	7690	161.56 s	9279 cyc.	10057 cyc.
Remote Control	4608	92.57 s	9768 cyc.	10541 cyc.

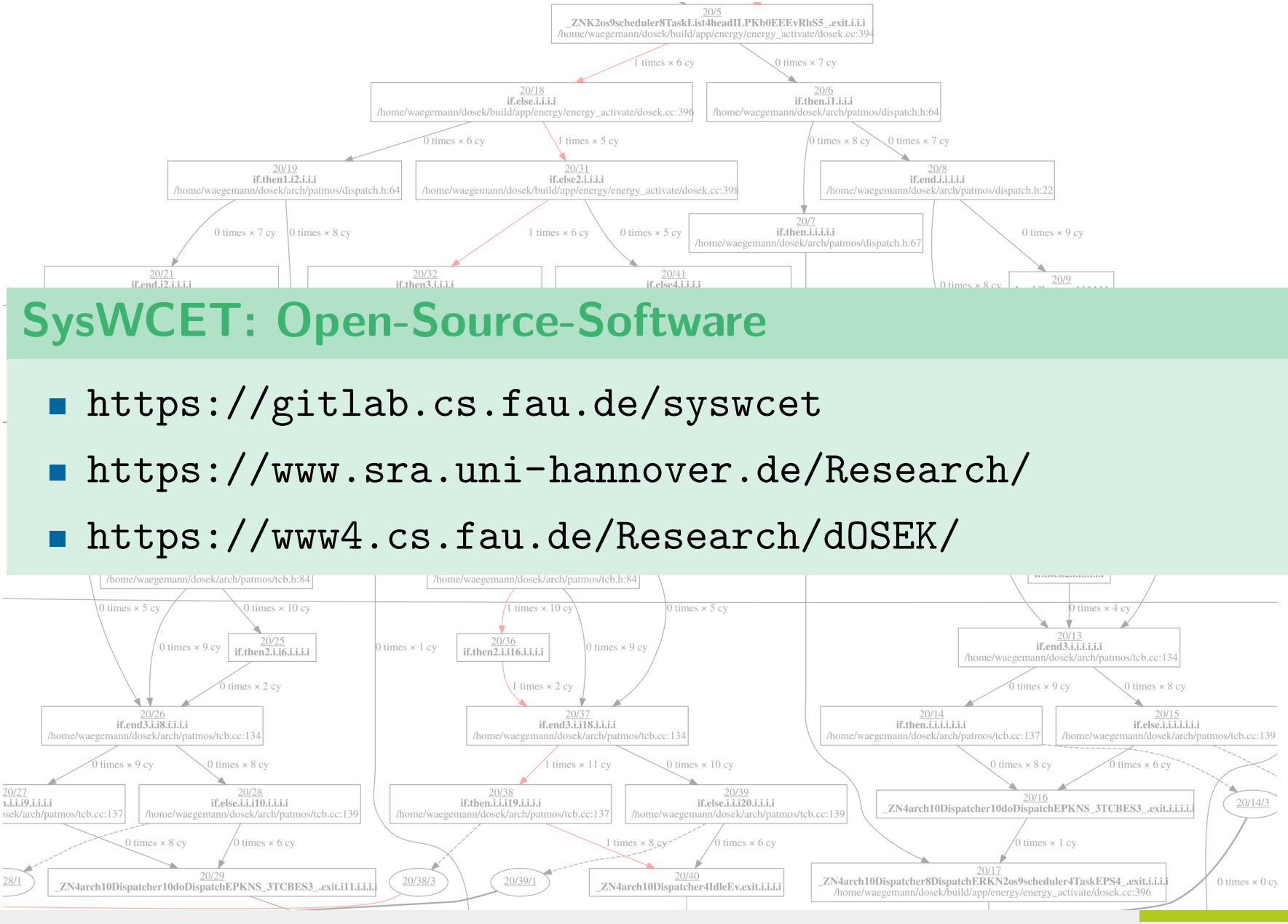
Gliederung

Zusammenfassung und Ausblick

- WCRT: WCET des Gesamtsystems, während ein Job ausführt wird
  - RTOS, Interrupts und andere Threads interferieren mit der Ausführung
  - Zusammengesetzte WCRT Analyse akkumuliert Pessimismus
- SysWCET bietet eine automatische systemweite WCRT Analyse
  - Ganzheitliche IPET Formulierung über Threadgrenzen hinweg
  - Unterstützung für ereignisgesteurte fixed-priority Systeme
  - Basierend auf RTOS-gewahrer Anwendungsanalyse
- Richtungen für zukünftige Forschung
  - Komplexere Hardwaremodelle (Pipelines, Caches,...)
  - Dichterer OS State Transition Graph für mehr Effizienz
  - Extraktion und Einbringen von thread-übergreifender Flow Facts

Schluss

Danke für die Aufmerksamkeit!



SysWCET: Open-Source-Software

- https://gitlab.cs.fau.de/syswcet
- https://www.sra.uni-hannover.de/Research/
- https://www4.cs.fau.de/Research/dOSEK/