Friedrich-Alexander-Universität
Technische Fakultät

FAU

Masterarbeit - Abschlussvortrag

WATWA-OS: A Minimalistic, Statically Analyzable Operating System for Resource-Constrained Real-Time Systems

Tobias Häberlein
12.11.2024

Motivation

- Energiebeschränkte Mikrocontroller (→ Batterien, Solarenergie)
- Anwendungsfall: **Lese Daten von Sensor** → **Verarbeite die Daten**

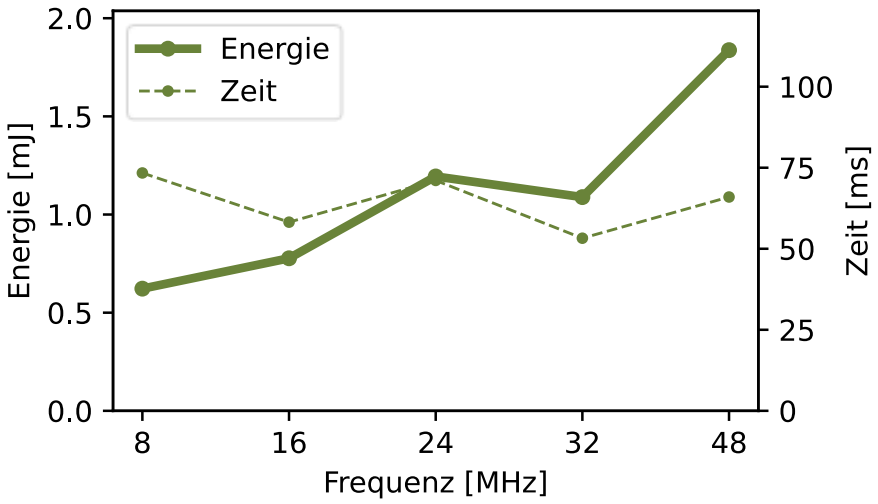
Ziel: Energieeffizienz

Motivation

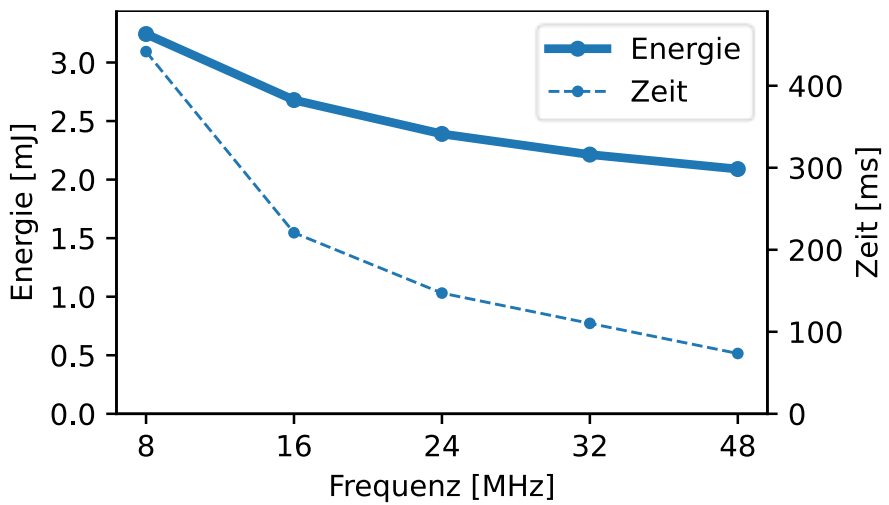
- Energiebeschränkte Mikrocontroller (→ Batterien, Solarenergie)
- Anwendungsfall: **Lese Daten von Sensor** → **Verarbeite die Daten**

Ziel: Energieeffizienz

Daten von Sensor lesen



Daten verarbeiten



Motivation

- **Dynamische Ansätze (zur Laufzeit)**
  - Verringere Taktfrequenz während I/O-Operationen

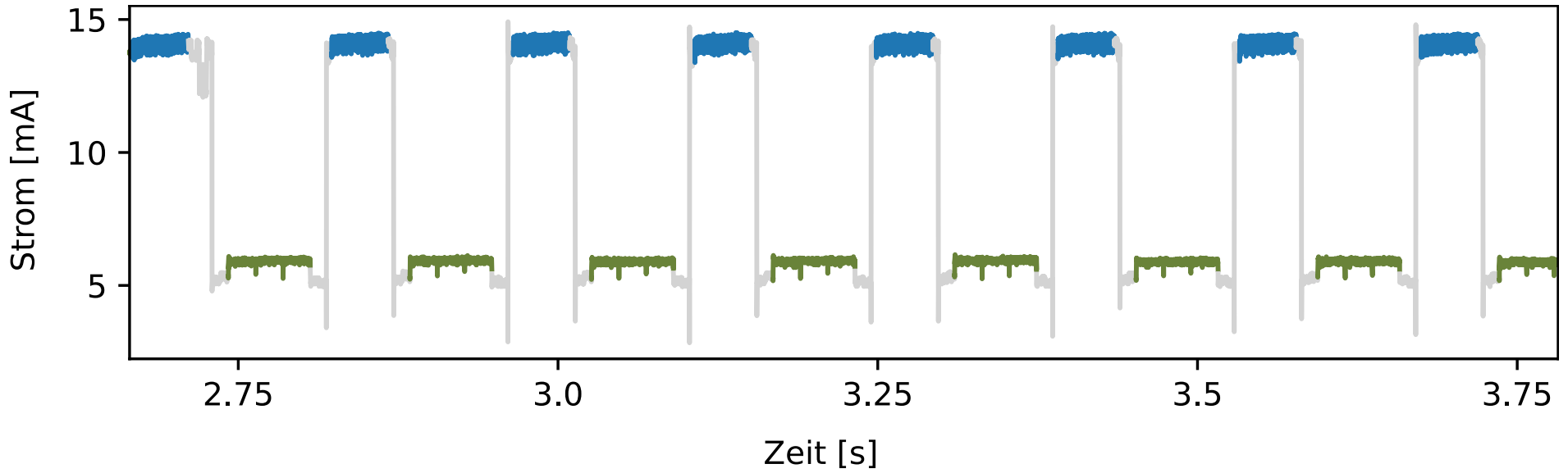
- **Vorteile**
  + Dynamischer Frequenzwechsel zur Laufzeit spart Energie
  + Ansätze lassen sich im Betriebssystem einbauen
    → kein zusätzlicher Code durch den Entwickler

Michel Rottleuthner et al. 2023.
Dynamic Clock Reconfiguration for the Constrained IoT and Its Application to Energy-Efficient Networking.
In Proceedings of the 2022 International Conference on Embedded Wireless Systems and Networks. 168–179.

Holly Chiang et al. 2021.
Power Clocks: Dynamic Multi-Clock Management for Embedded Systems.
In Proceedings of the 2021 International Conference on Embedded Wireless Systems and Networks. 139–150.

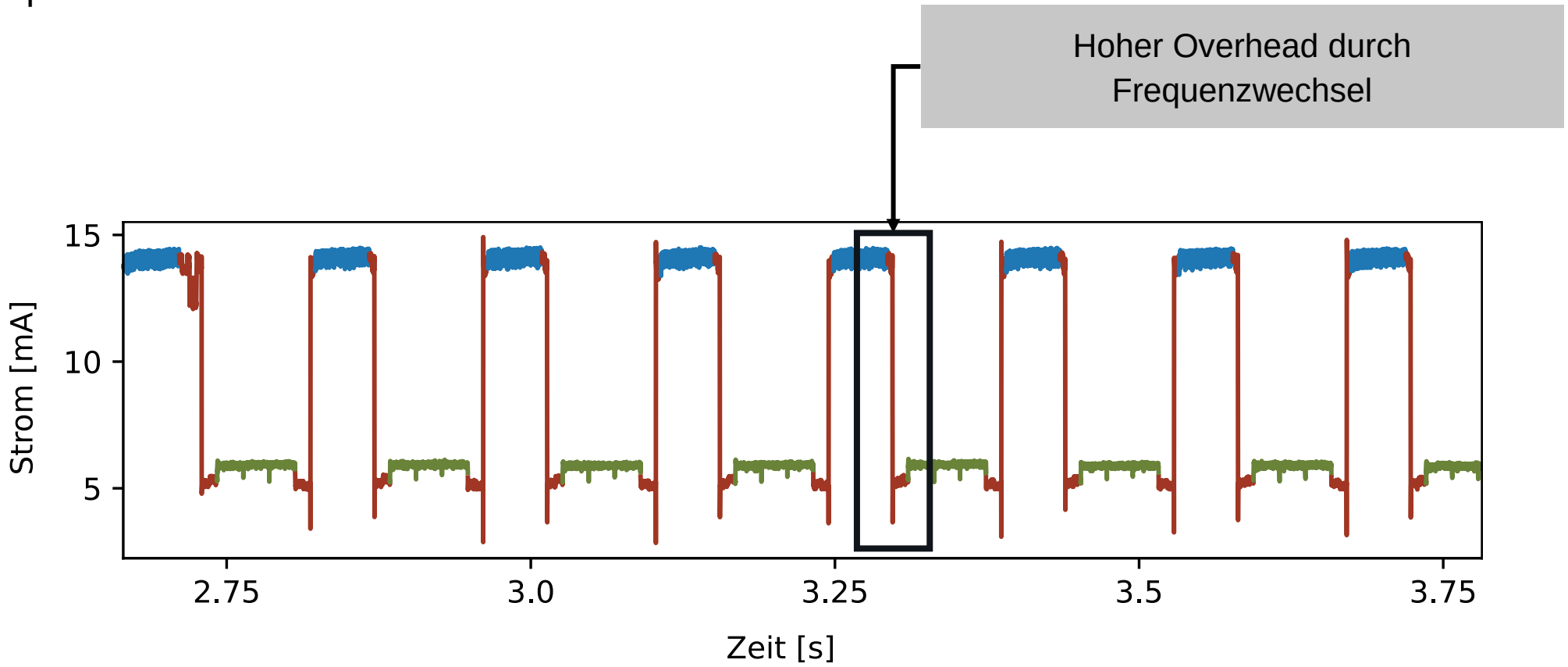
Motivation

- **Nachteil**
  - Frequenzwechsel nicht immer sinnvoll



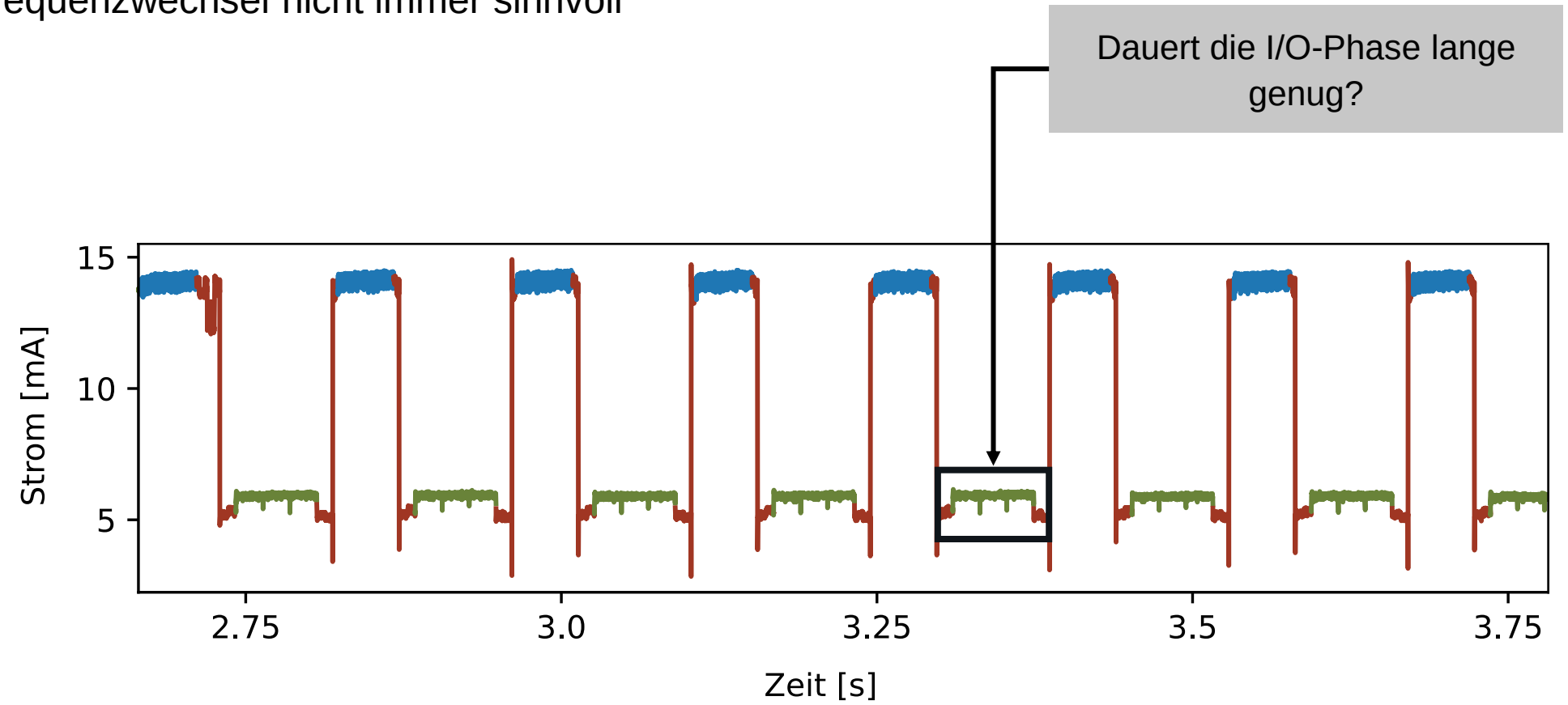
Motivation

- **Nachteil**
  - Frequenzwechsel nicht immer sinnvoll



Motivation

- **Nachteil**
  - Frequenzwechsel nicht immer sinnvoll



Problem dynamischer Ansätze

- Overhead zur Laufzeit des Systems
- Keine optimalen Taktkonfigurations-Wechsel

Problem dynamischer Ansätze

- Overhead zur Laufzeit des Systems
- Keine optimalen Taktkonfigurations-Wechsel

WATWA-OS: Statischer Ansatz

- Verwende existierende statische Analysemethoden[1] der *Worst-Case Response Energy Consumption*
- Erweiterung um Analyse und Optimierung verschiedener Taktkonfigurations-Wechsel
- Ziel: Optimierte Instanzen eines möglichst generischen Betriebssystems

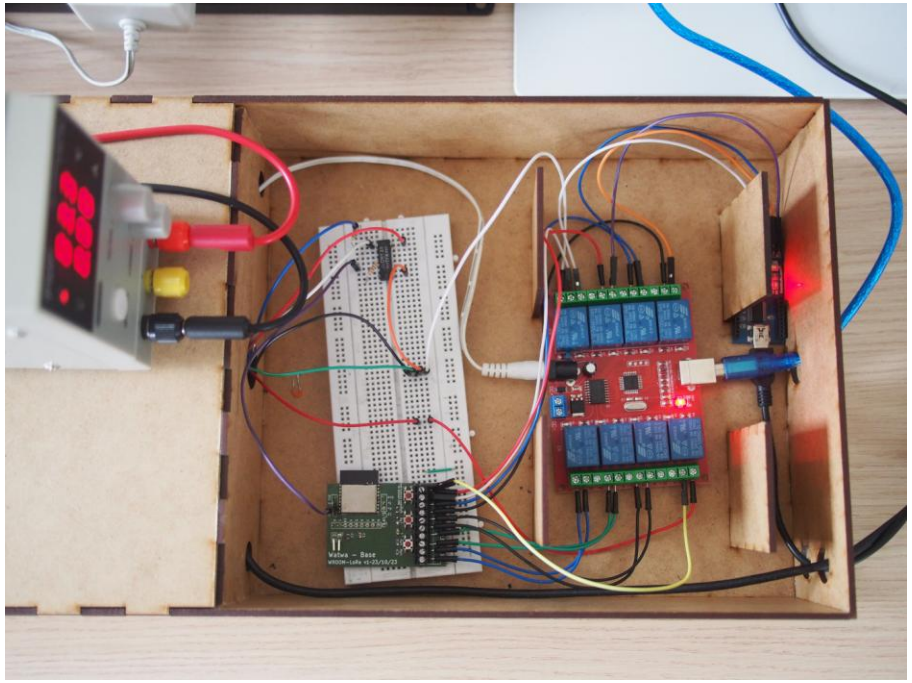
[1]Peter Wägemann et al. 2018.
Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems.
In 30th Euromicro Conference on Real-Time Systems (ECRTS 2018).

WATWA-OS

Systemmodell

Betriebssystem:

- Prioritätsbasiertes, präemptives Scheduling
- Vordefinierte Tasksets
- Interrupts erlaubt



Testplattform: ESP32-C3

Hardware-Plattform:

- Unterstützung kleiner Microcontroller
- Single-Core, keine Caches, deterministisches Verhalten
- Verschiedene Peripheriegeräte und Taktkonfigurations-Netzwerk

Statische Analyse

Beispiel

```
void task1() {
  compute_1();
  compute_2();
  if (cond) {
    uart_on();
    uart_write();
  }
  compute_3();
}
```

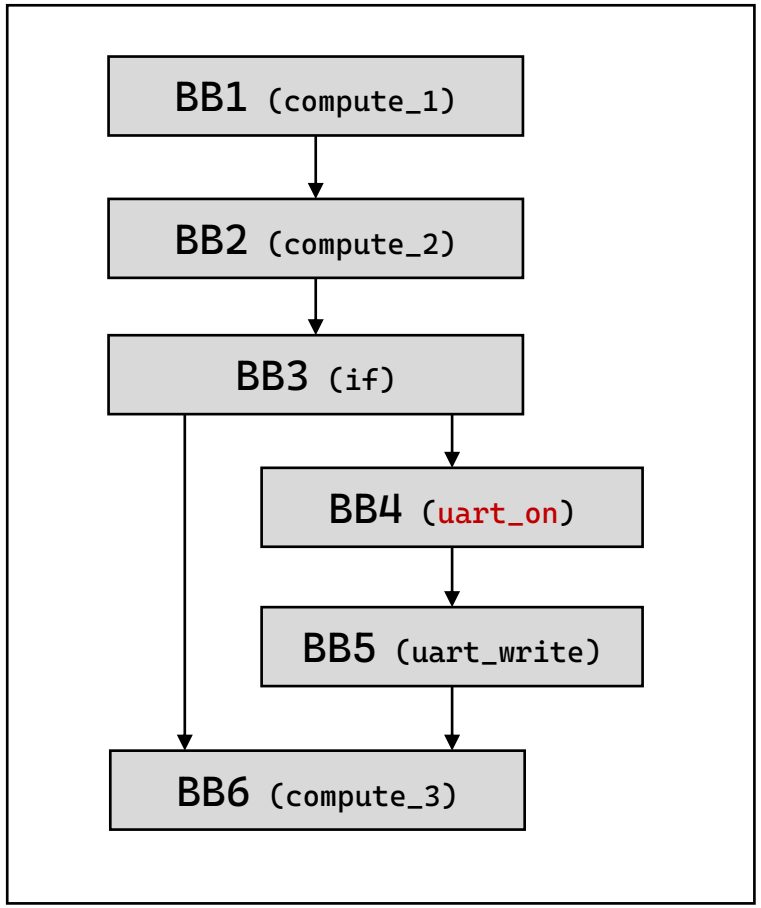
Task – C Programm

Statische Analyse

Beispiel

```
void task1() {
  compute_1();
  compute_2();
  if (cond) {
    uart_on();
    uart_write();
  }
  compute_3();
}
```

Task – C Programm



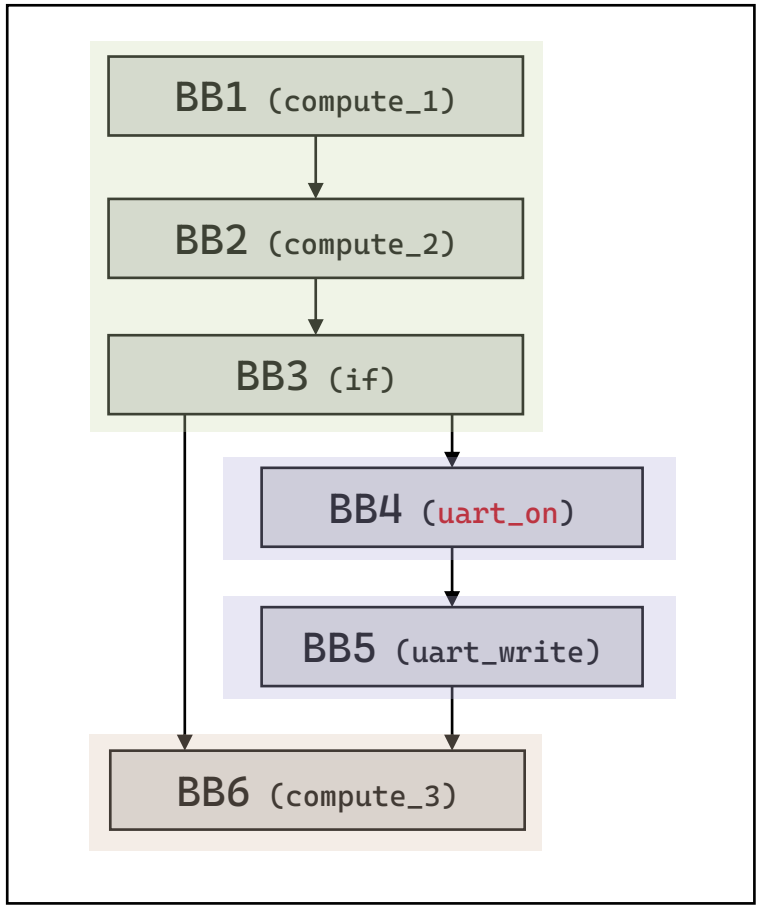
Kontrollflussgraph mit Basisblöcken (BBs)

Statische Analyse

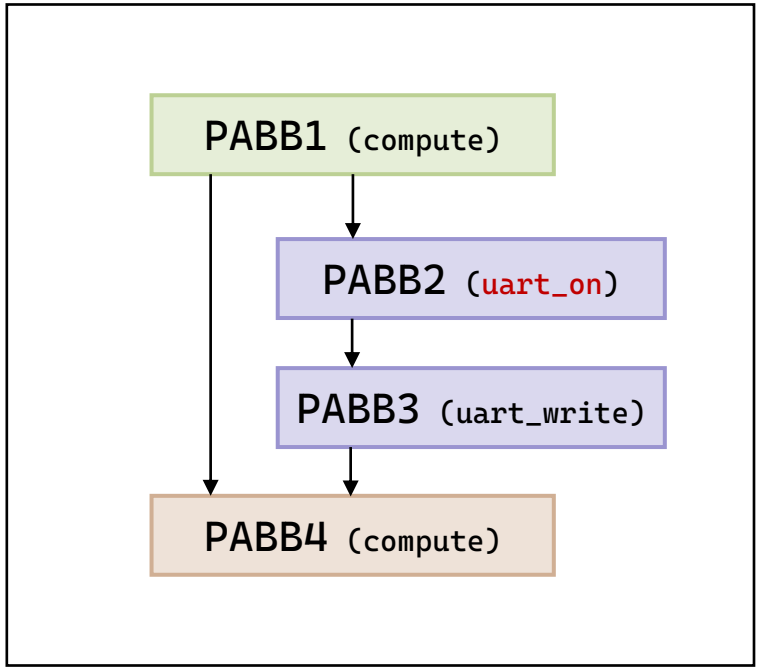
Beispiel

```
void task1() {
  compute_1();
  compute_2();
  if (cond) {
    uart_on();
    uart_write();
  }
  compute_3();
}
```

Task – C Programm



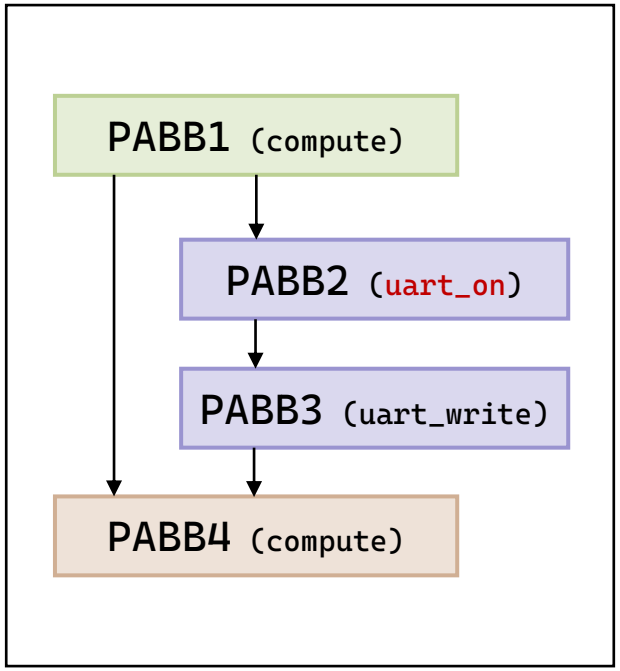
Kontrollflussgraph mit
Basisblöcken (BBs)



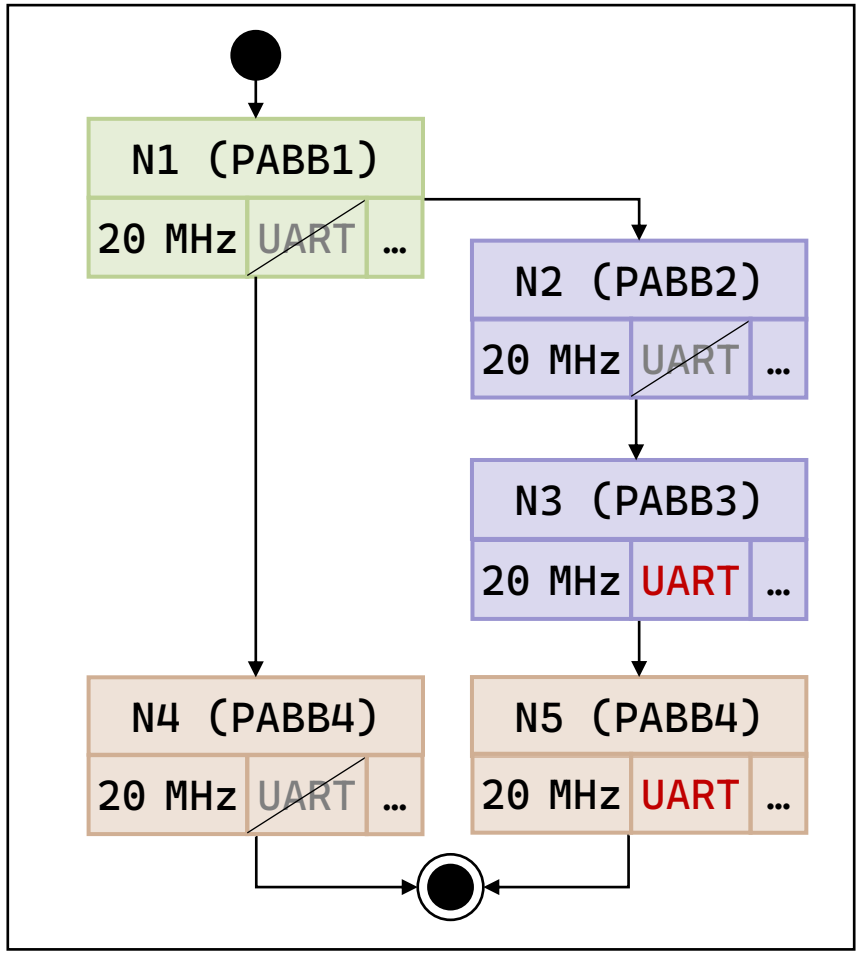
Kontrollflussgraph mit
Power-Atomic Basic Blocks (PABBs)

Statische Analyse

Beispiel



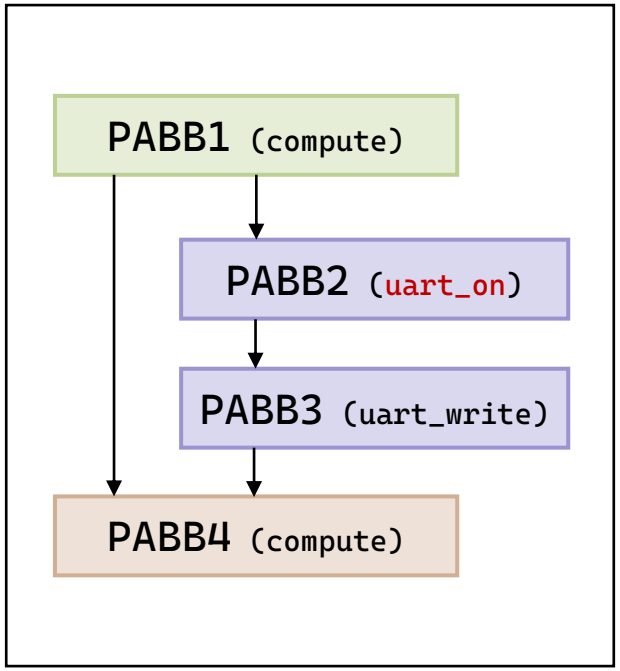
Kontrollflussgraph mit
Power-Atomic Basic Blocks (PABBs)



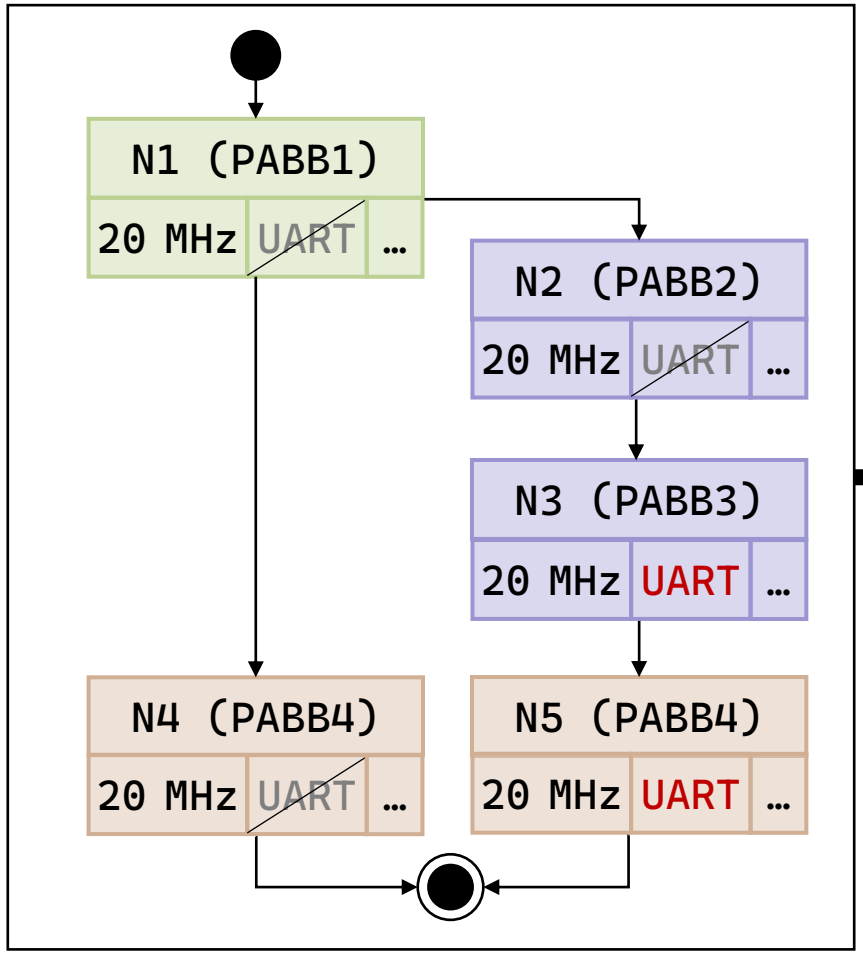
Zustandsübergangsgraph

Statische Analyse

Beispiel



Kontrollflussgraph mit
Power-Atomic Basic Blocks (PABBs)

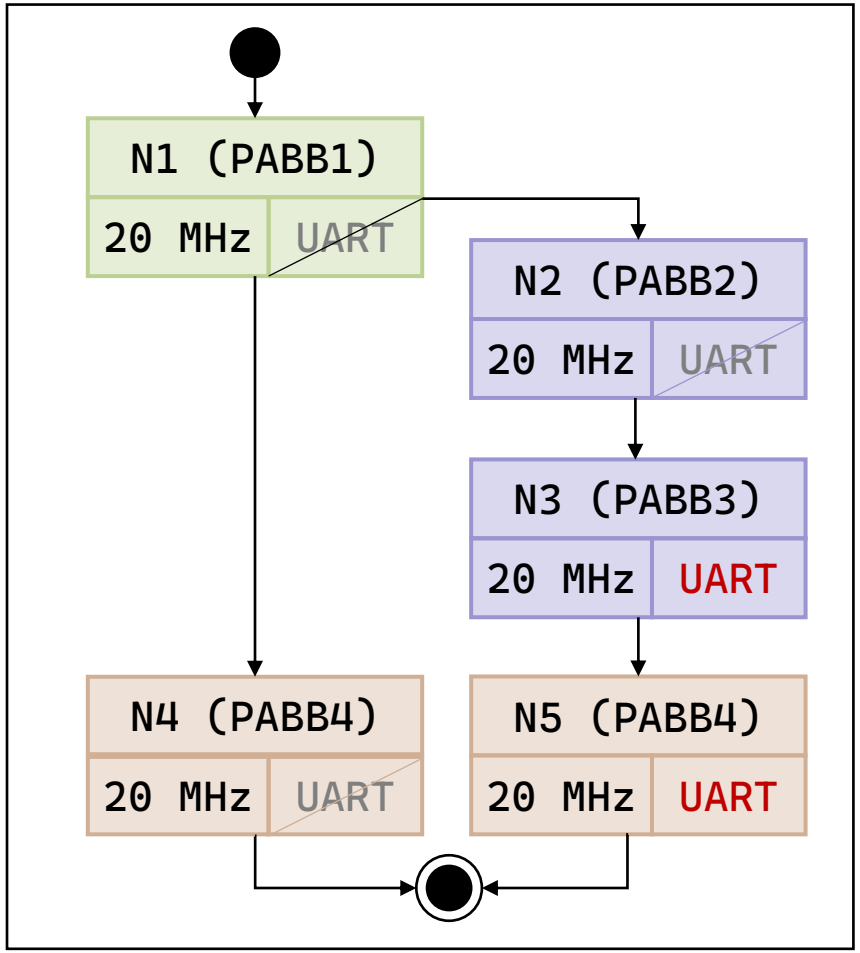


Zustandsübergangsgraph

Finde Pfad mit höchstem Energieverbrauch

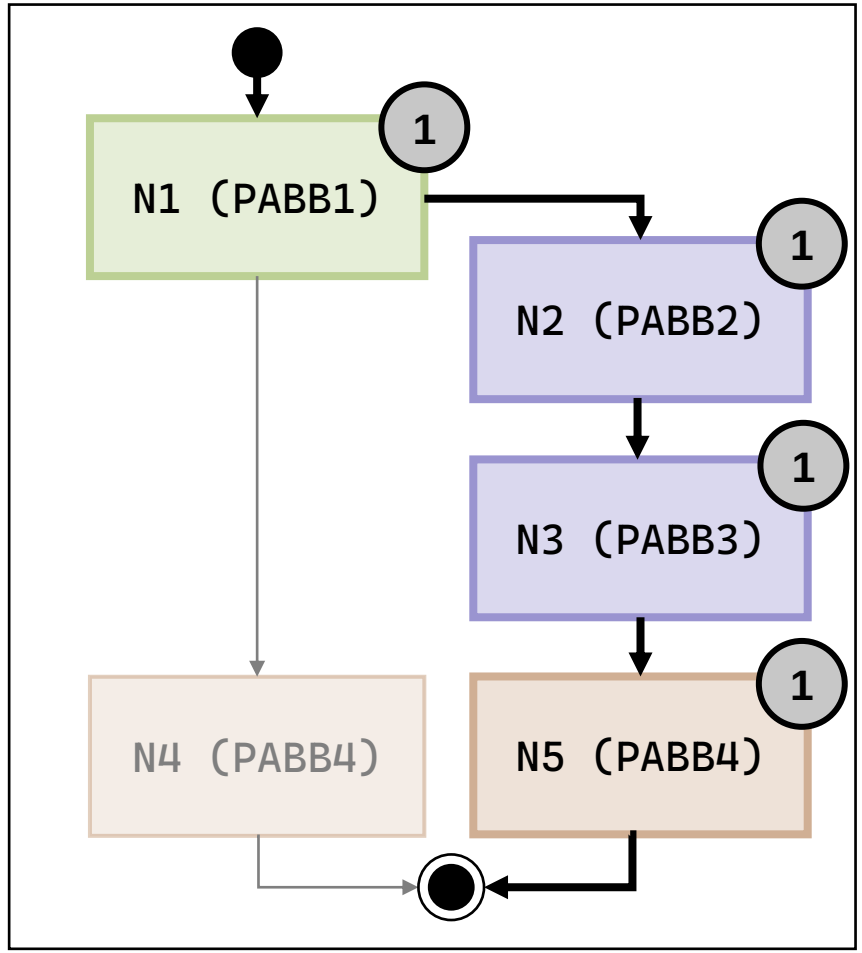
Statische Analyse

Beispiel

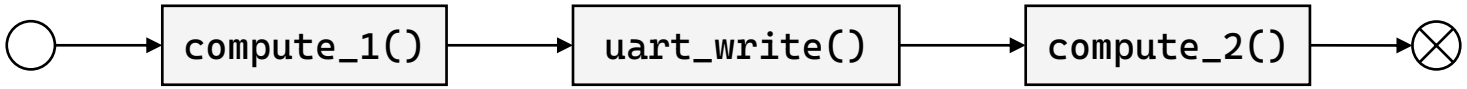


Zustandsübergangsgraph

Mathematisches Optimierungsproblem (**I**nteger **L**inear **P**rogram – **ILP)**

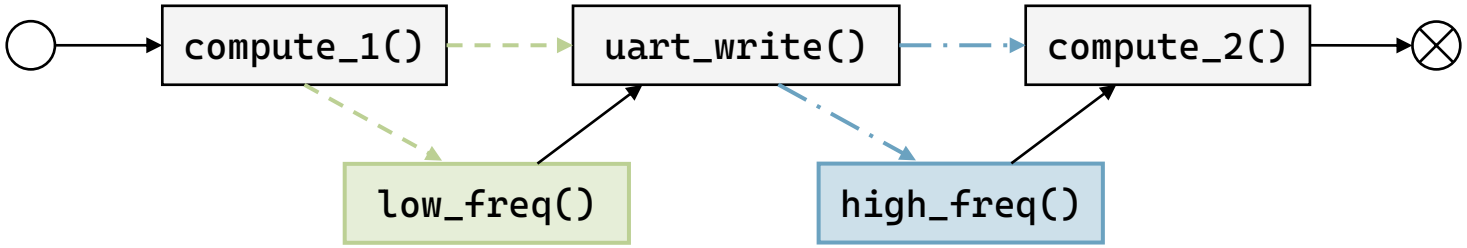


ILP-Lösung



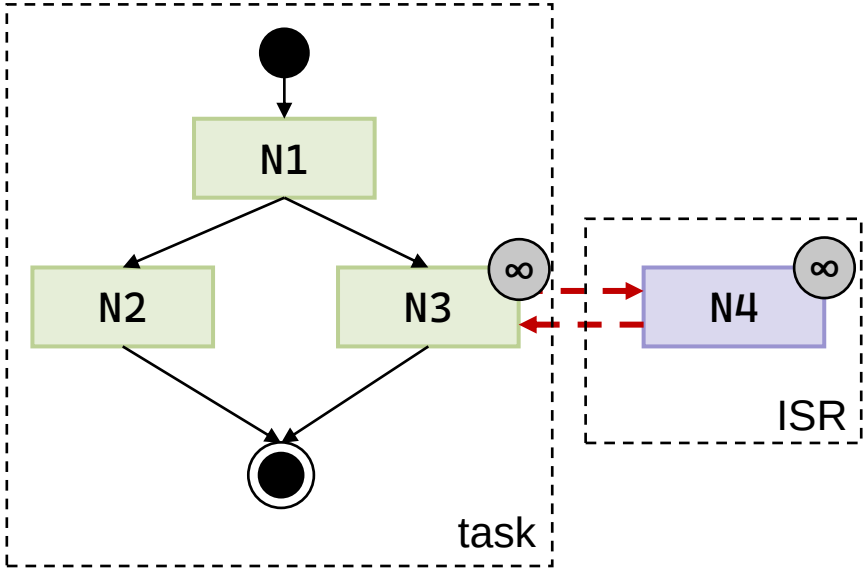
Optimierung der Taktfrequenz:

- Erweitere Zustandsübergangsgraph um künstliche Taktkonfigurationswechsel → „Entscheidungsknoten“
  - Vor und nach I/O-Operationen
  - Vor und nach Schleifen mit I/O-Operationen
  - Am Anfang und Ende eines Tasks



Optimierung der Taktfrequenz:

- Erweitere Zustandsübergangsgraph um künstliche Taktkonfigurationswechsel → „Entscheidungsknoten“
  - Vor und nach I/O-Operationen
  - Vor und nach Schleifen mit I/O-Operationen
  - Am Anfang und Ende eines Tasks
- Analysiere Energieverbrauch **aller** Taktkonfigurationen (*hier: 4 Pfade = 4 ILPs*)
- Lösung mit niedrigstem Energieverbrauch → Erzeuge entsprechende Betriebssystem-Instanz

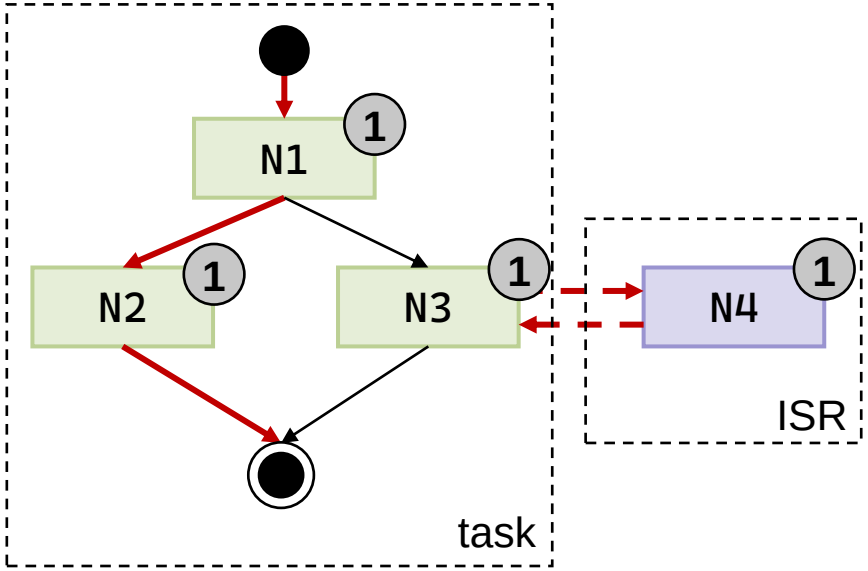


Interrupts

- `N3` und `N4` können unendlich oft aktiviert werden

  → Ankunftshäufigkeit angeben (z.B. alle 500ms)

Behandlung von Interrupts und Schleifen



Interrupts

- `N3` und `N4` können unendlich oft aktiviert werden

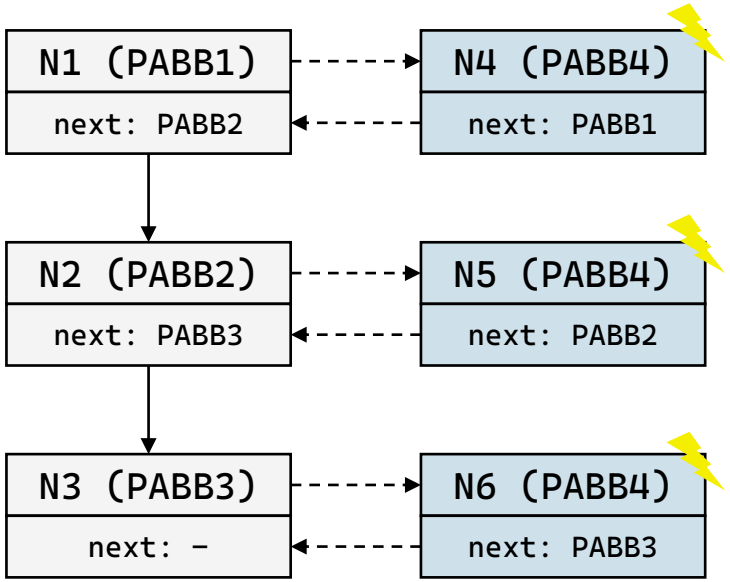
  → Ankunftshäufigkeit angeben (z.B. alle 500ms)

Schleifen

- `N3` und `N4` können sich gegenseitig aktivieren
- Auch dann, wenn sie nicht auf dem Worst-Case-Pfad liegen

  → Schleife darf nur dann ausgeführt werden,
  wenn mindestens eine Eingangskante ausgeführt wird

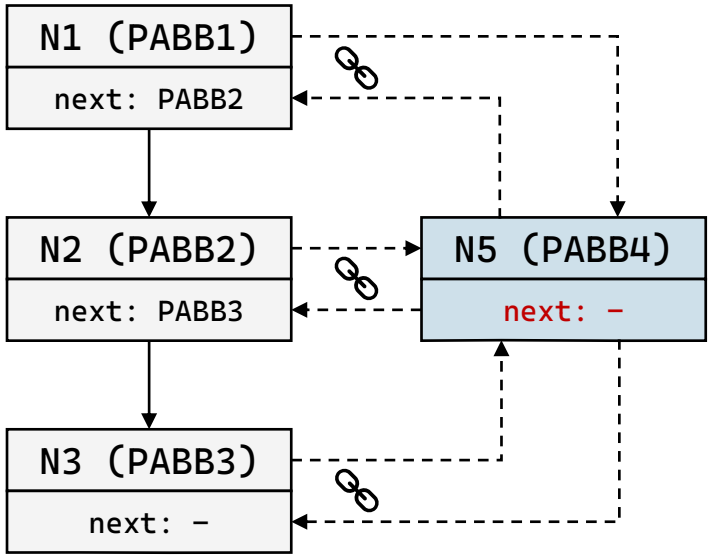
Anzahl der Zustände



Herausforderung:

- Viele Knoten mit sehr ähnlichen Zuständen (→ ISRs)

Anzahl der Zustände



Herausforderung:

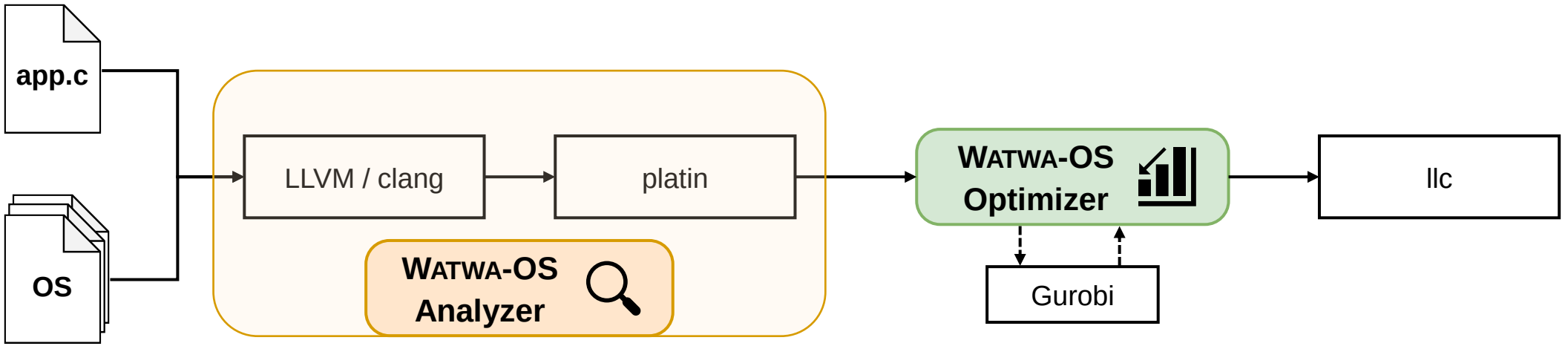
- Viele Knoten mit sehr ähnlichen Zuständen (→ ISRs)

Lösung:

- Verschmelze ähnliche ISR-Knoten
  - → Scheduler-Zustand geht verloren
  - → Zusätzliche Bedingungen im ILP nötig

WATWA-OS

Implementierung - Überblick



WATWA-OS Analyzer:

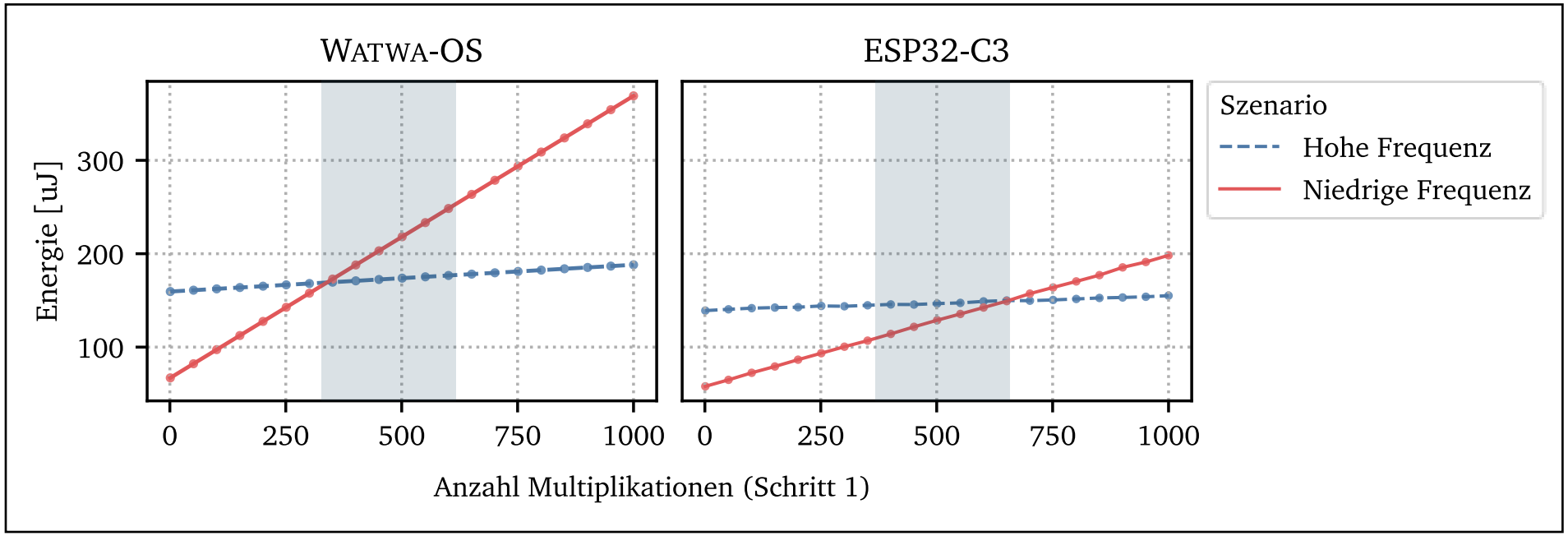
- Konstruktion des Zustandsübergangsgraphs (**LLVM**)
- Transformation des Graphs in ein ILP (**platin**)

WATWA-OS Optimizer:

- Bestimmung optimaler Taktkonfigurations-Wechsel
  - → Lösen aller ILPs (**Gurobi**)
- Einbau der Taktfrequenzwechsel im LLVM-IR Code

Beispielanwendung, 1 Task:

1. Führe Berechnungen in Schleife durch (*s. Anzahl Multiplikationen*)
2. Übertrage Ergebnisse per UART (*20 Byte*)



Frequenzwechsel auf Task-Ebene

Evaluation – Analysedauer

Anwendung	Knoten	Gesamtzeit	Enumeration	Schleifenbehandlung
1 Task, 1 ISR (leer)	199	0.692 s	0.30 s	0.14 s
1 Task, 2 ISRs (leer)	299	1.949 s	0.67 s	1.05 s
1 Task, 1 ISR (mit Kommunikation)	664	30.810 s	3.36 s	26.82 s
1 Task, 2 ISRs (mit Kommunikation)	1104	1100 s	9.56 s	1054 s

Analysedauer (LLVM)

#ILPs	ILP Variablen	Optimierungszeit
4	178	0.02 s
64	915	2.50 s
256	933	6.87 s
1024	951	102.91 s

ILP-Lösungsdauer (Optimizer)

Evaluation – Knotenverschmelzung

Knotenverschmelzung:

- Erhebliche Auswirkungen auf die ILP-Lösungsdauer
- Weniger Variablen im ILP, mehr Bedingungen pro Variable

Anwendung	Typ	Variablen	Bedingungen	Lösungsdauer
1 Task, 1 ISR (mit Kommunikation)	Normal	612	608	374 ms
	Verschmolzen	512 (-16.3 %)	518 (-14.8 %)	123 ms **(-67.1 %)**
1 Task, 3 ISRs (mit Kommunikation)	Normal	1098	1280	860 ms
	Verschmolzen	798 (-27.3 %)	1010 (-21.1 %)	81 ms **(-90.6 %)**

ILP-Lösungsdauer (Optimizer)

Zusammenfassung

WATWA-OS

- Statische Analyse und Optimierung von Betriebssystem-Instanzen für energiebeschränkte Echtzeitsysteme
- Automatische Anpassung der Taktkonfiguration an bestimmten Punkten
- Vorteile gegenüber dynamischen Ansätzen
  - + Kein Overhead zur Laufzeit
  - + Worst-Case-Optimale Taktkonfigurations-Wechsel

Fragen?

WATWA-OS
Idee

Problem dynamischer Ansätze

- Overhead zur Laufzeit des Systems
- Keine optimalen Taktkonfigurations-Wechsel

WATWA-OS: Statischer Ansatz

- Verwende existierende statische Analysemethoden[1] der *Worst-Case Response Energy Consumption*
- Erweiterung um Analyse und Optimierung verschiedener Taktkonfigurations-Wechsel
- Ziel: Optimierte Instanzen eines möglichst generischen Betriebssystems

[1]Peter Wägemann et al. 2018.
Whole-System Worst-Case Energy-Consumption Analysis for Energy-Constrained Real-Time Systems.
In 30th Euromicro Conference on Real-Time Systems (ECRTS 2018).

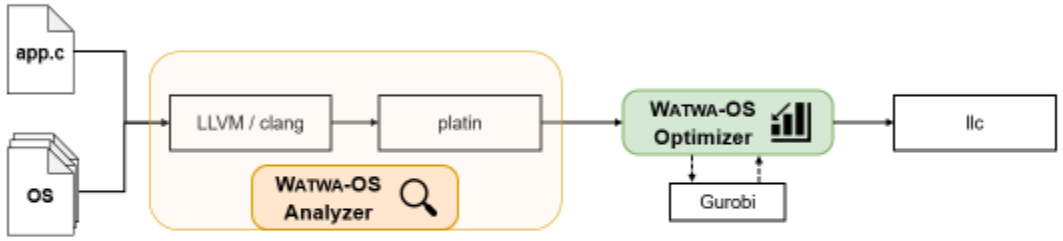
WATWA-OS
Optimierung

Optimierung der Taktfrequenz:

- Erweitere Zustandsübergangsgraph um künstliche Taktkonfigurationswechsel → „Entscheidungsknoten"
  - Vor und nach I/O-Operationen
  - Vor und nach Schleifen mit I/O-Operationen
  - Am Anfang und Ende eines Tasks
- Analysiere Energieverbrauch **aller** Taktkonfigurationen (*hier: 4 Pfade = 4 ILPs*)
- Lösung mit niedrigstem Energieverbrauch → Erzeuge entsprechende Betriebssystem-Instanz

Fragen?

WATWA-OS
Implementierung - Überblick



WATWA-OS Analyzer:

- Konstruktion des Zustandsübergangsgraphs (**LLVM**)
- Transformation des Graphs in ein ILP (**platin**)

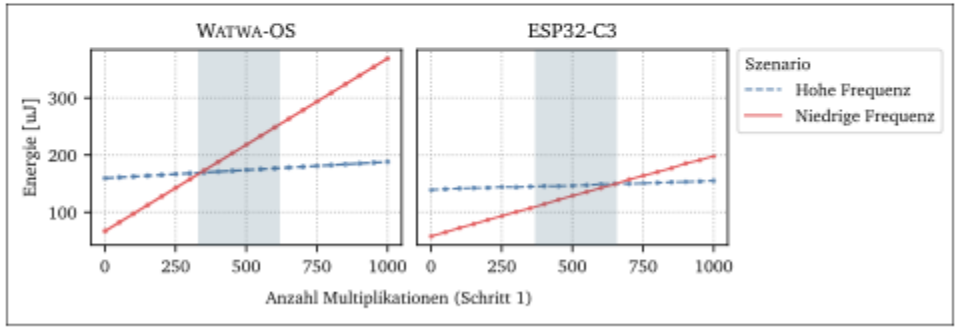
WATWA-OS Optimizer:

- Bestimmung optimaler Taktkonfigurations-Wechsel
  - → Lösen aller ILPs (**Gurobi**)
- Einbau der Taktfrequenzwechsel im LLVM-IR Code

WATWA-OS
Evaluation – Testanwendung

Beispielanwendung, 1 Task:

1. Führe Berechnungen in Schleife durch (*s. Anzahl Multiplikationen*)
2. Übertrage Ergebnisse per UART (*20 Byte*)



Frequenzwechsel auf Task-Ebene

Vielen Dank
für Eure Aufmerksamkeit!