



Freie Universität



Berlin

Dynamic Clock Reconfiguration for the Constrained IoT and its Application to Energy-efficient Networking

Michel Rottleuthner - HAW Hamburg

Thomas C. Schmidt - HAW Hamburg

Matthias Wählisch - Freie Universität Berlin

NOx & VOC values high

Structural instability

PHILIP Turn on Ventilation

Motion detected

Escalator must be greased

Train operator asleep

Low glucose Level

Trash Bin Full

Crossover is stuck

Door doesn't open

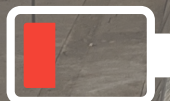
Heartrate high

Sync Time

Light bulb broken

Needs maintenance

Motivation



Replace Filter!

You made 10.000 steps!

Stop Escalator

IoT Firmware Development

ARMmbed

R IOT



Unified tooling



- Agile development
- Faster time to market
- Better interoperability
- Improved software support and updates (better security ?)

Zephyr™

CONTIKI
NEXT GENERATION

IoT Firmware Development



- Agile development
- Faster time to market
- Better interoperability
- Improved software support and updates (better security ?)



- Limited access to very hardware specific features
 - Needed for low-power optimizations
 - Must be added to the HAL to employ it cross-platform

Outline

- Motivation
 - A Catch with Modern IoT Firmware Development
- Energy in the Constrained IoT
- Dynamic Clock Configuration
- Our Approach: ScaleClock
 - Evaluation
 - Application: Energy-efficient Networking
- Conclusion & Future Work
- Q&A



Freie Universität

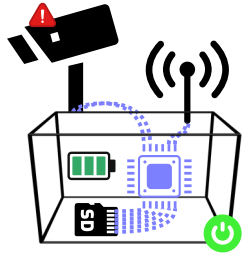


Berlin

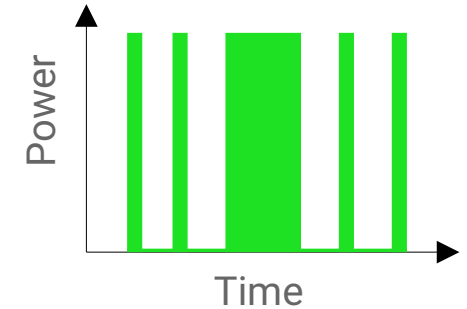
Dynamic Clock Reconfiguration for the Constrained IoT and its Application to **Energy**-efficient Networking

In the Constrained IoT

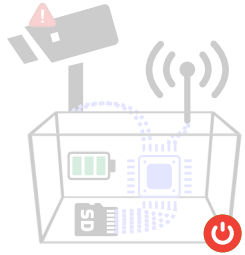
IoT Power Management



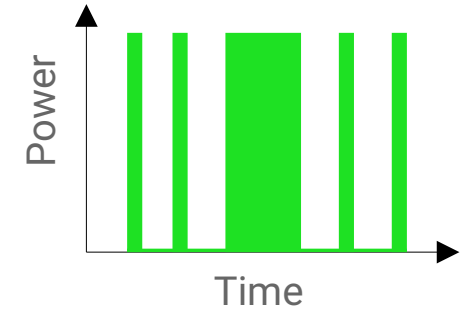
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep



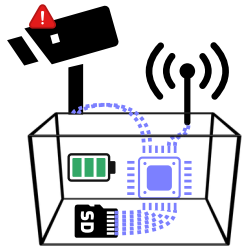
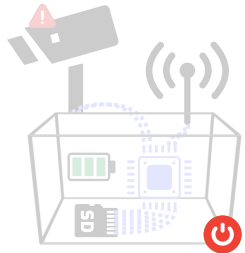
IoT Power Management



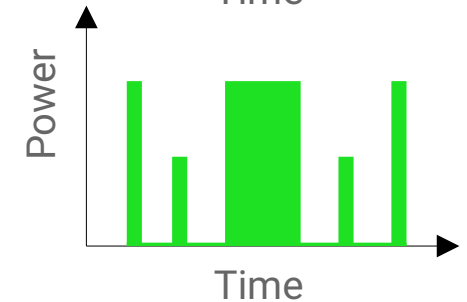
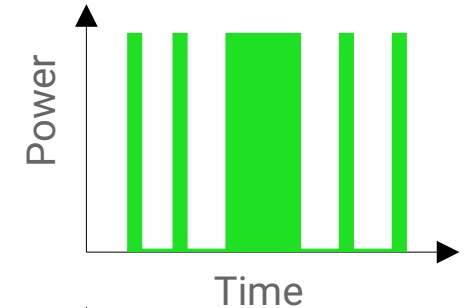
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep



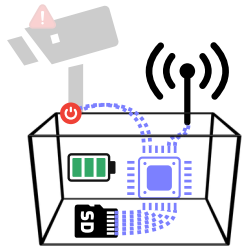
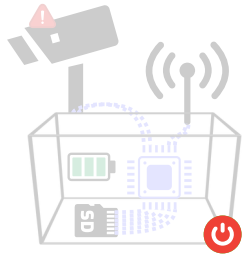
IoT Power Management



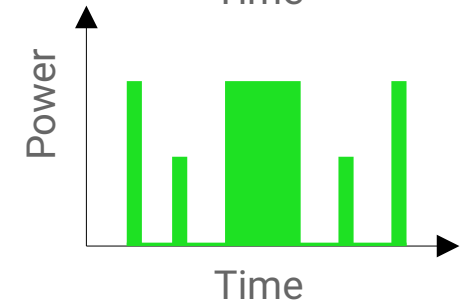
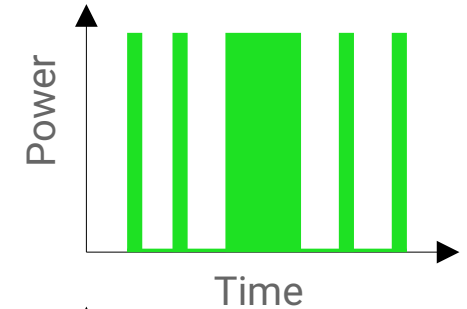
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



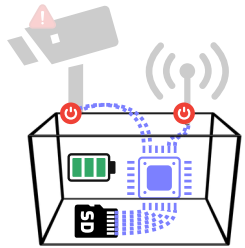
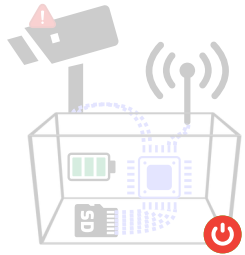
IoT Power Management



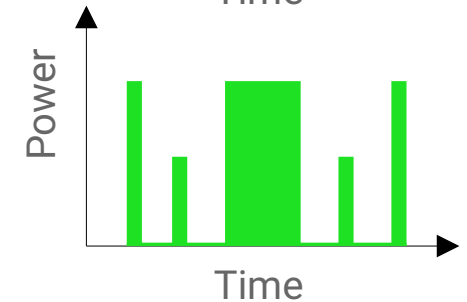
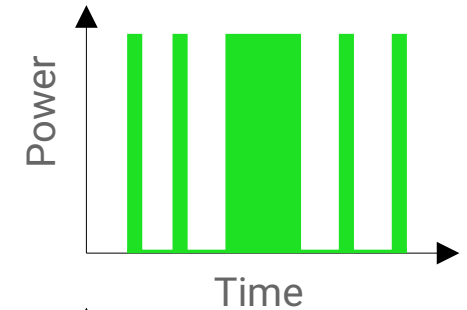
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



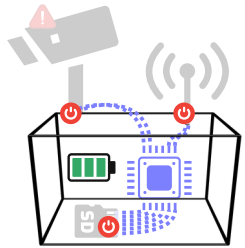
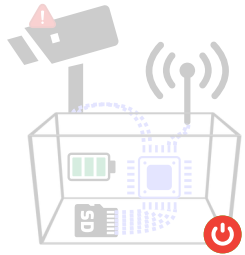
IoT Power Management



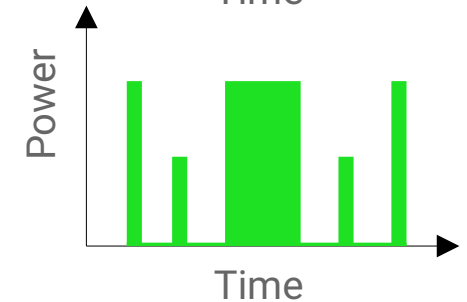
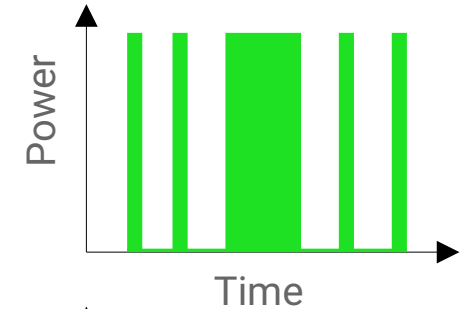
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



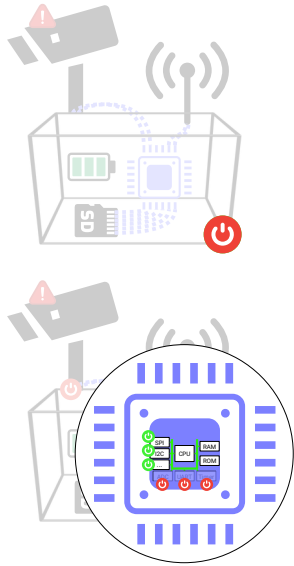
IoT Power Management



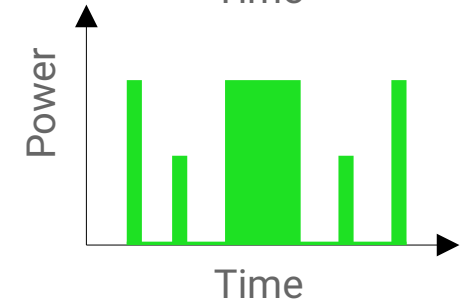
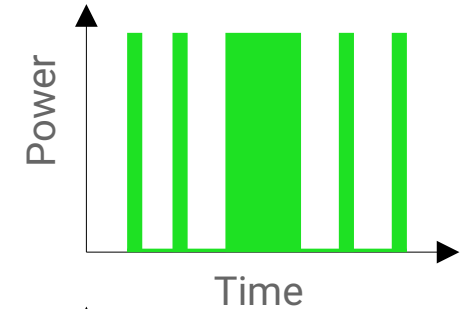
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



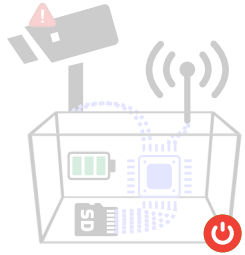
IoT Power Management



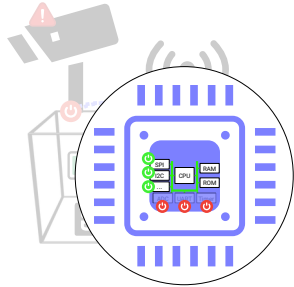
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



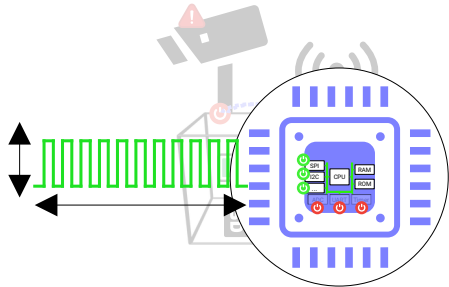
IoT Power Management



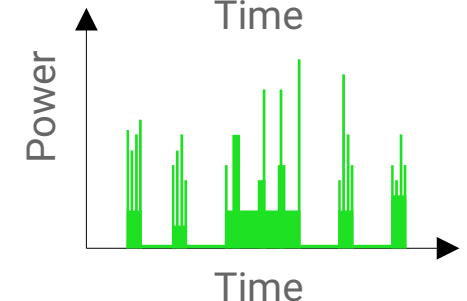
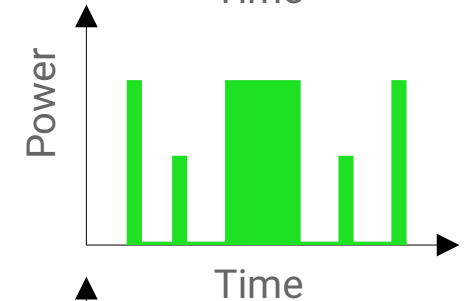
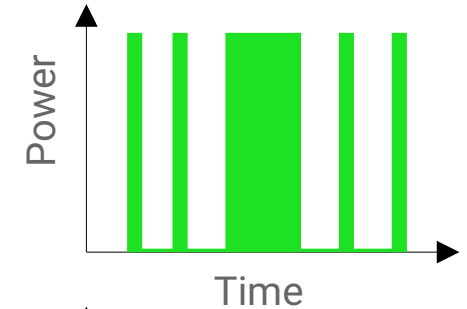
- Duty Cycling
 - Put system to sleep on idle
 - Wakeup via timers or interrupts
 - No processing during sleep



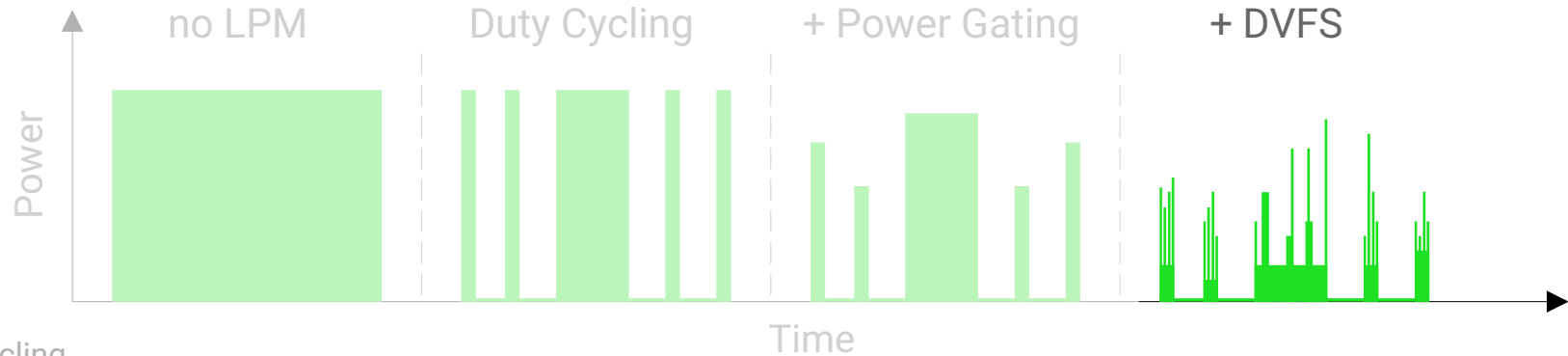
- Power Gating
 - Disable unneeded peripherals
 - External and MCU-internal peripherals



- Dynamic Voltage and Frequency Scaling
 - Fine grained performance control
 - Low-power processing



IoT Power Management



- Duty Cycling

- Set CPU to sleep on Idle
- Wakeup on event
- No processing during sleep

- Power Gating

- Turn off unused subsystems

- Dynamic Voltage and Frequency Scaling (DVFS)

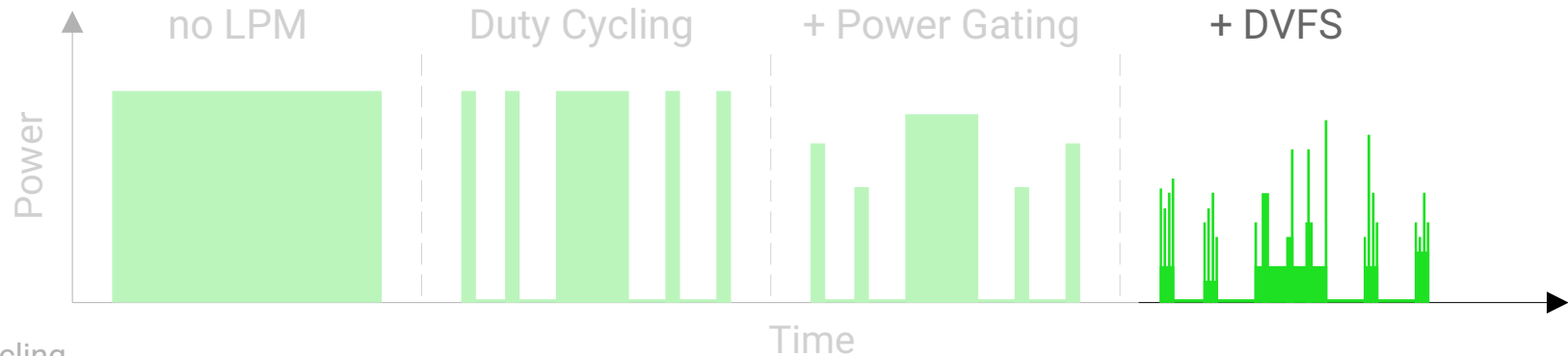
- Precise performance control
- Requires access to the clock configuration subsystem

Can be used complementary!



Not part of unified tooling

IoT Power Management



- Duty Cycling
 - Set CPU to sleep on Idle

How to provide dynamic clock configuration with unified tooling?

- Power Gating
 - Turn off unused subsystems
- Dynamic Voltage and Frequency Scaling (DVFS)
 - Precise performance control
 - **Requires access to the clock configuration subsystem**



Not part of unified tooling



Freie Universität

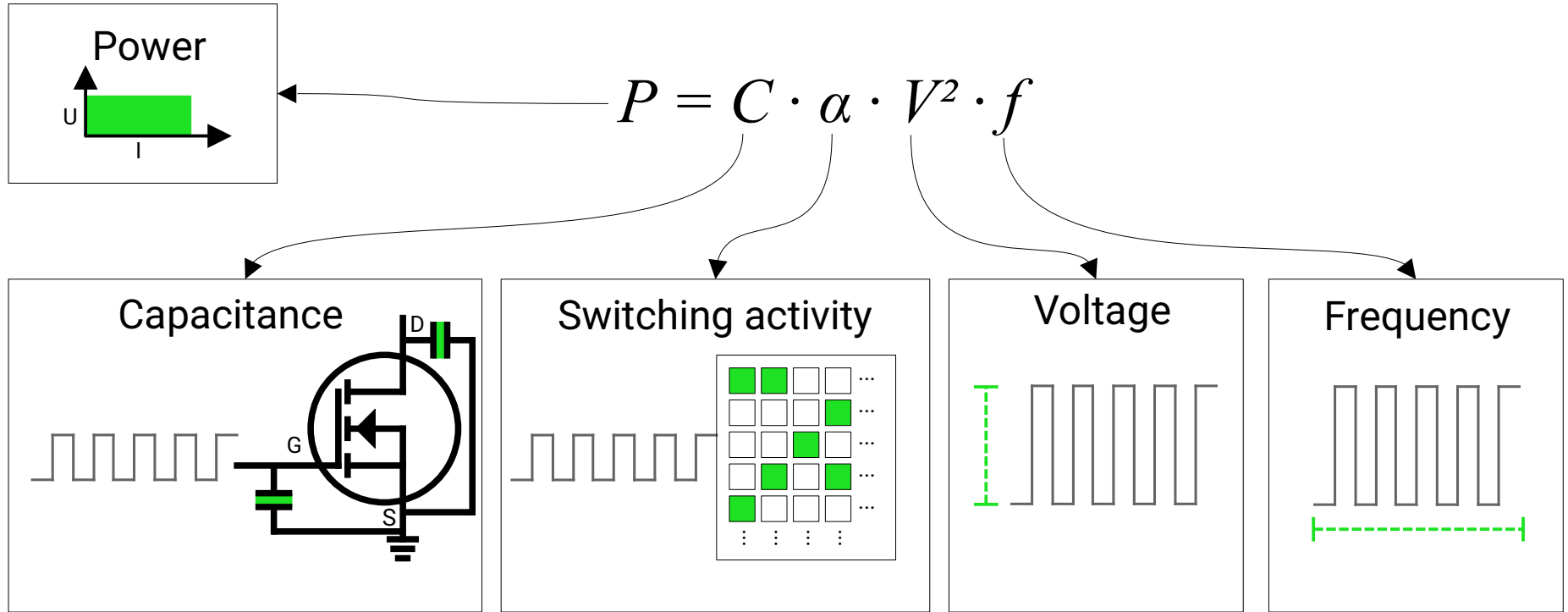


Berlin

Dynamic Clock Reconfiguration for the Constrained IoT and its Application to Energy-efficient Networking

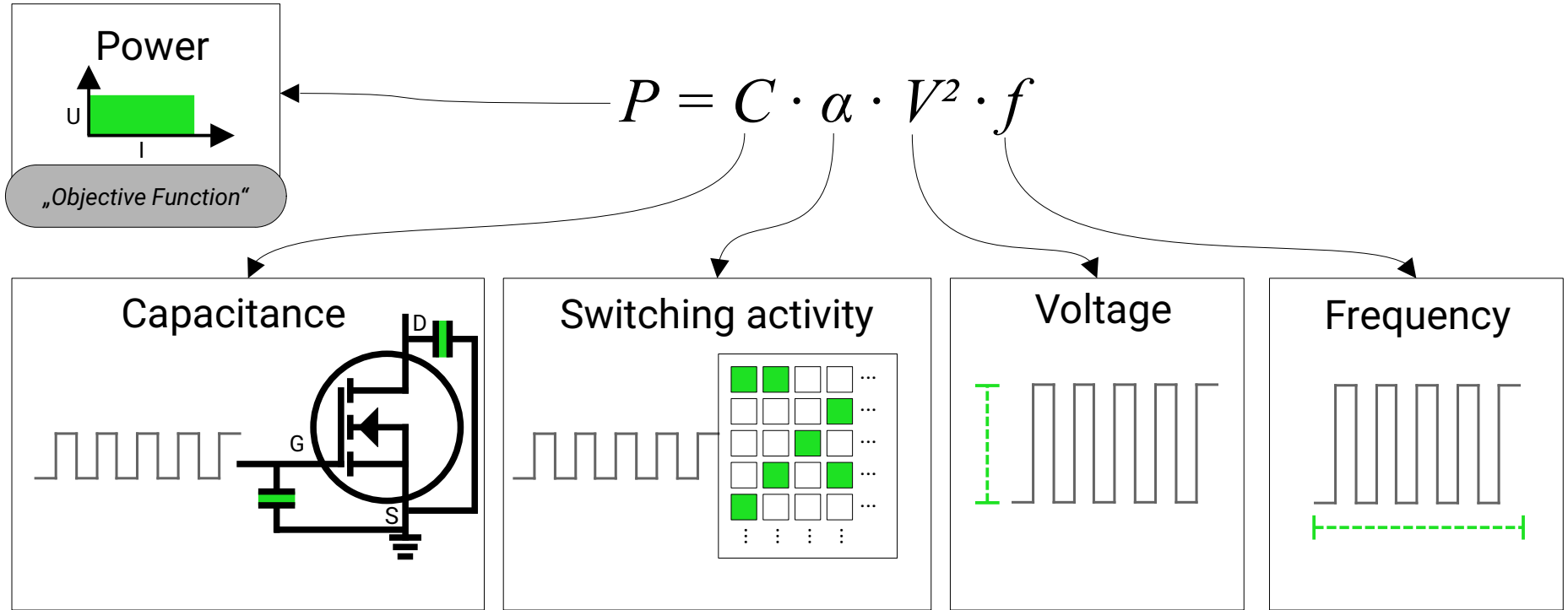
Problem Overview

Dynamic Voltage and Frequency Scaling



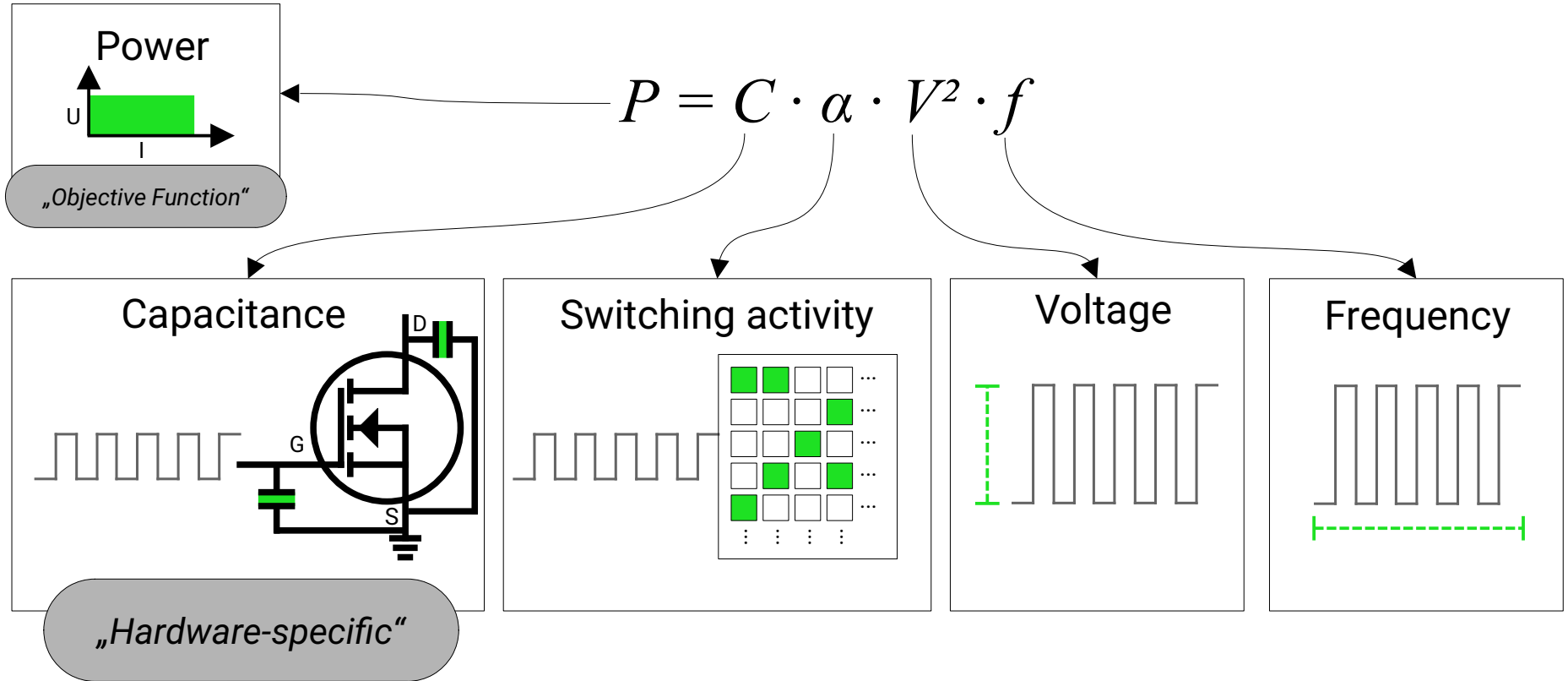
[6], [7]

Dynamic Voltage and Frequency Scaling



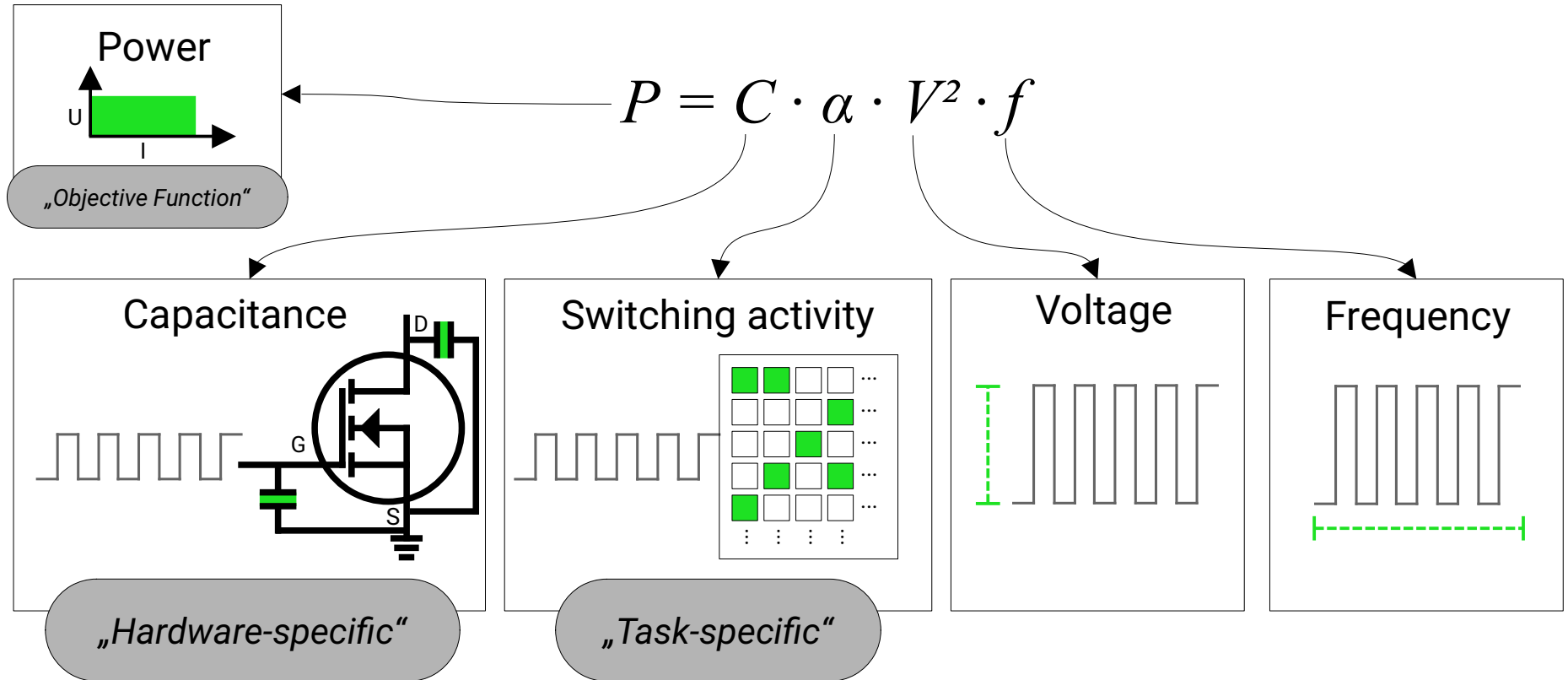
[6], [7]

Dynamic Voltage and Frequency Scaling



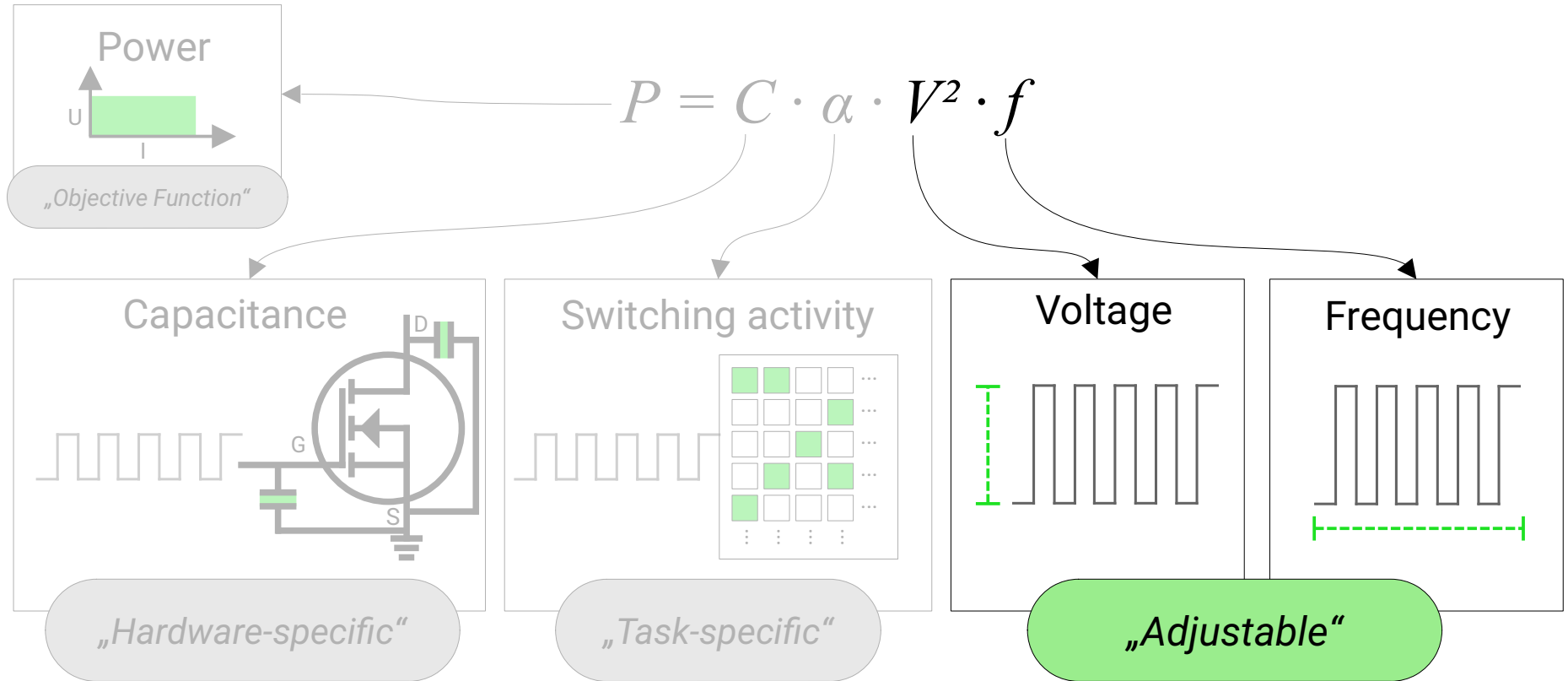
[6], [7]

Dynamic Voltage and Frequency Scaling



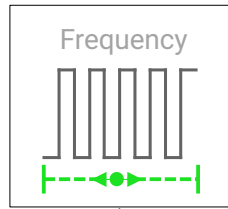
[6], [7]

Dynamic Voltage and Frequency Scaling

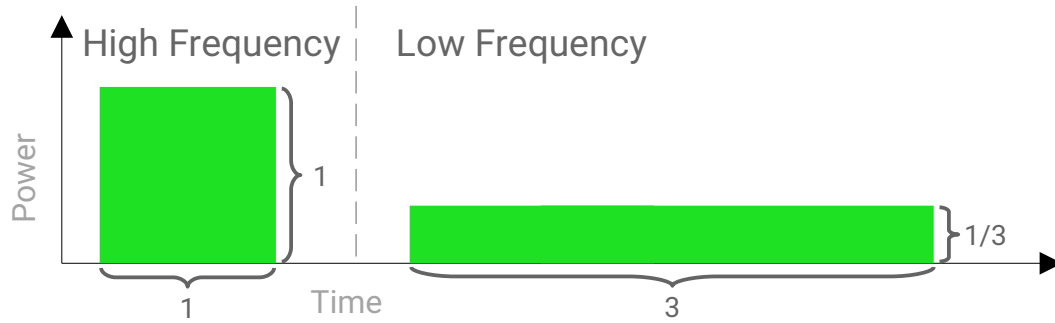


[6], [7]

Dynamic Voltage and Frequency Scaling




$$P = C \cdot \alpha \cdot V^2 \cdot f$$



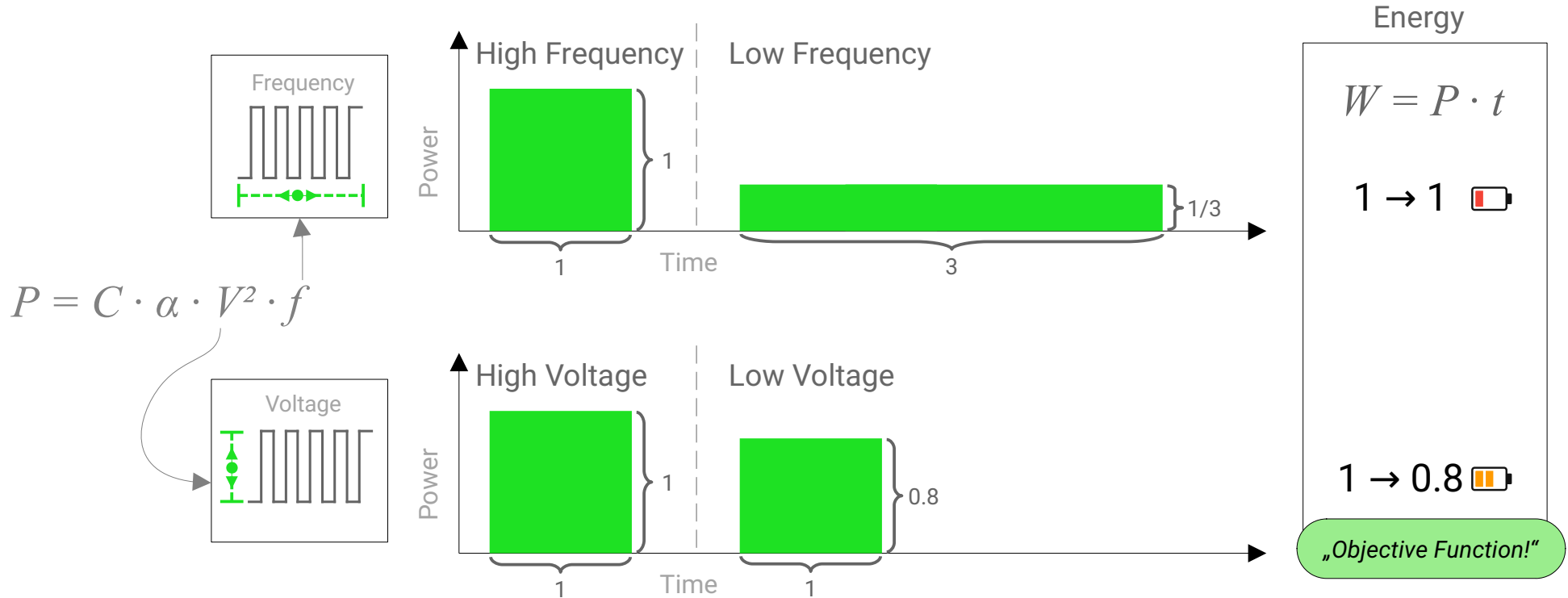
Energy

$$W = P \cdot t$$

1 → 1 

„Objective Function!“

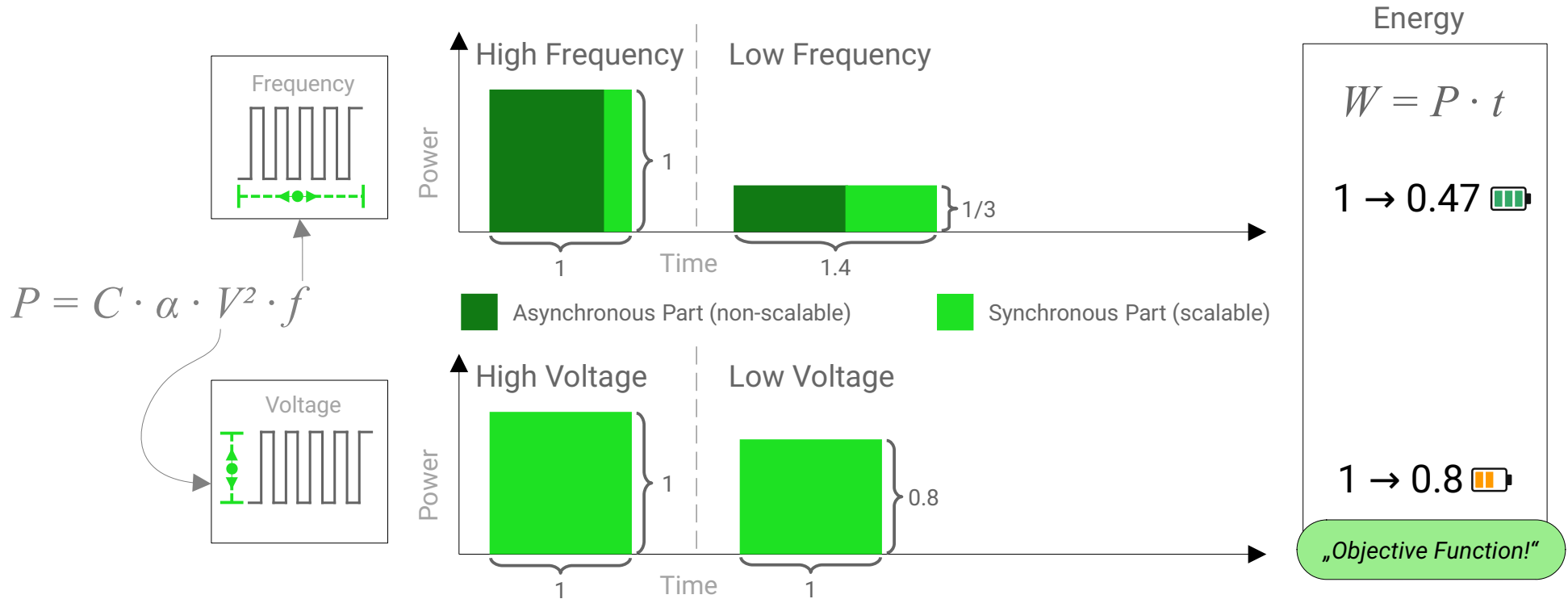
Dynamic Voltage and Frequency Scaling



$$P = C \cdot \alpha \cdot V^2 \cdot f$$

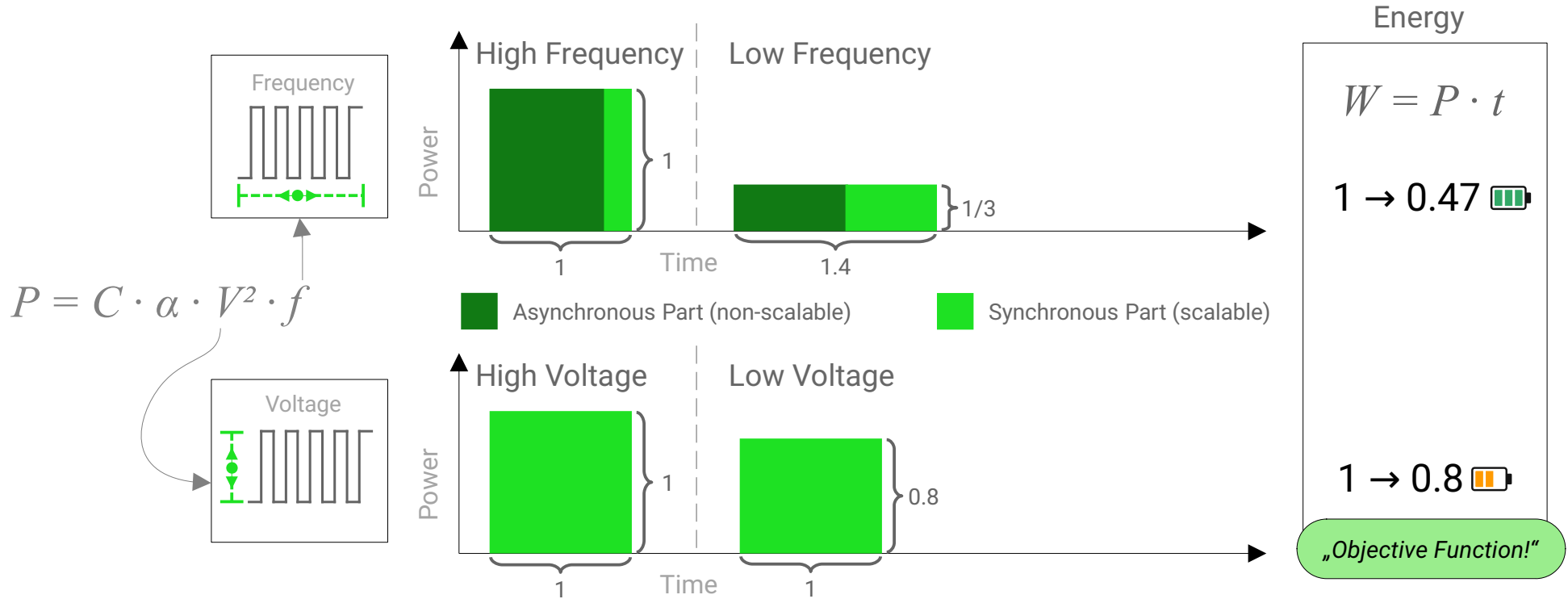
- range(frequency) >> range(voltage) and voltage \propto frequency

Dynamic Voltage and Frequency Scaling



- range(frequency) >> range(voltage) and voltage \npropto frequency
- Operations not always scale well with frequency (memory and peripheral access, radio communication, ...)
- Scalability bottlenecks waste energy

Dynamic Voltage and Frequency Scaling

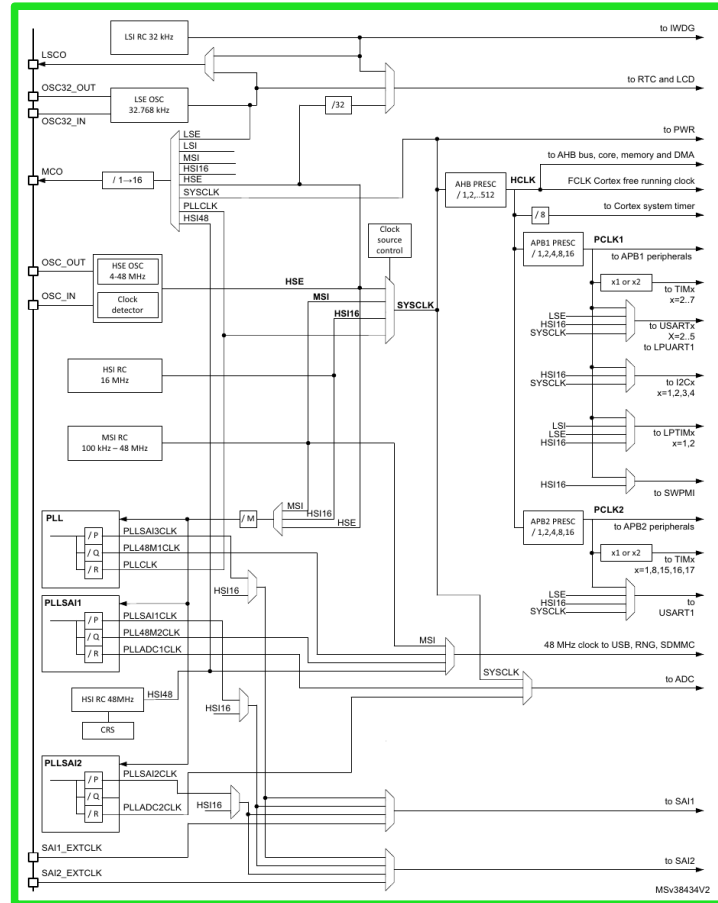


$$P = C \cdot \alpha \cdot V^2 \cdot f$$

- range(frequency) >> range(voltage) and voltage \propto frequency

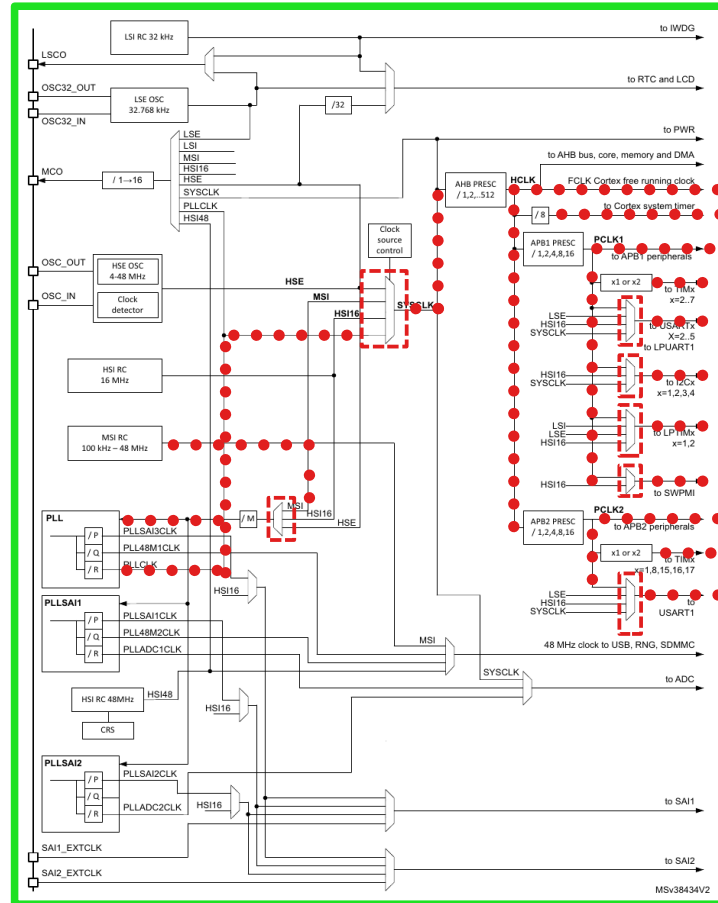
How to detect those bottlenecks to save energy?

Clock Trees



(from STM32L476RG reference manual RM0351)

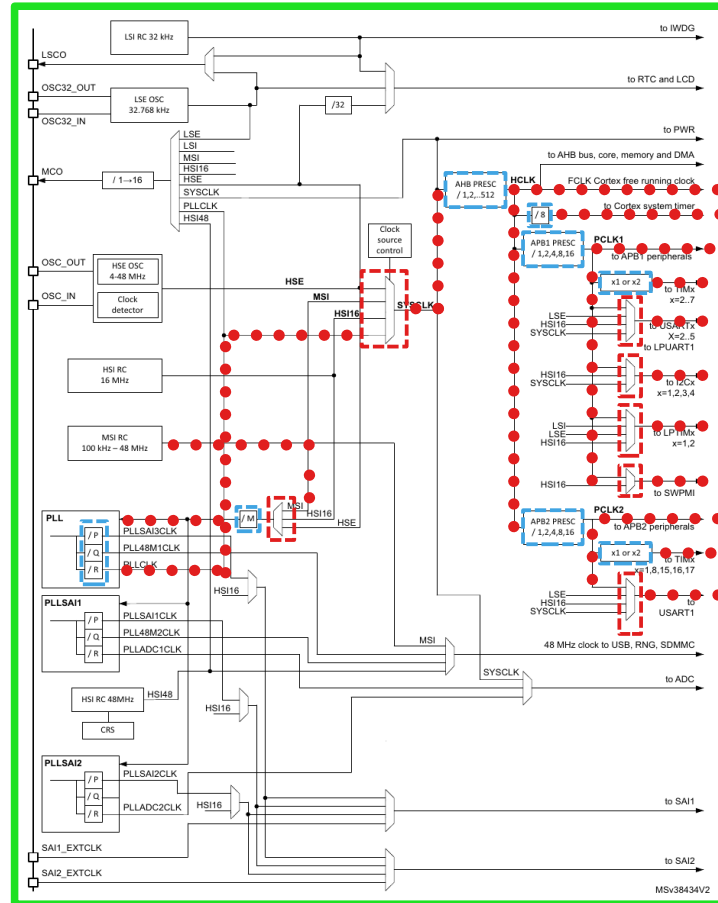
Clock Trees



Control
Topology

(from STM32L476RG reference manual RM0351)

Clock Trees



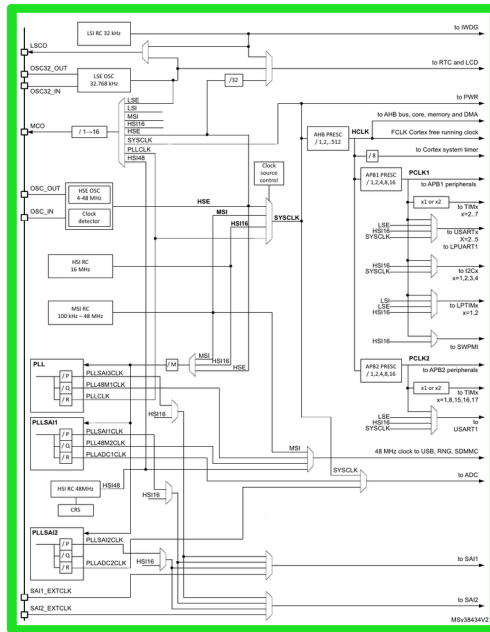
Control

Topology

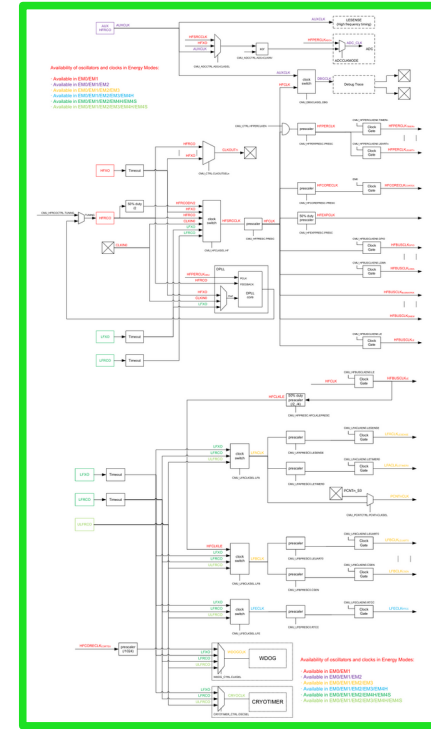
Frequency

(from STM32L476RG reference manual RM0351)

Clock Trees



STM32L476RG



EFM32PG12B



Freie Universität

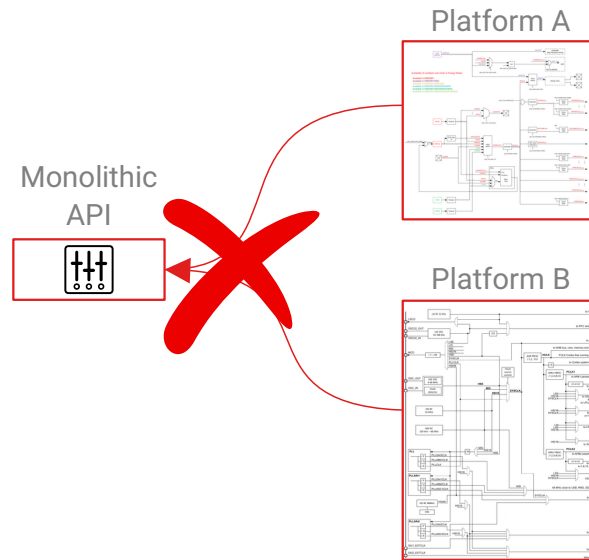


Berlin

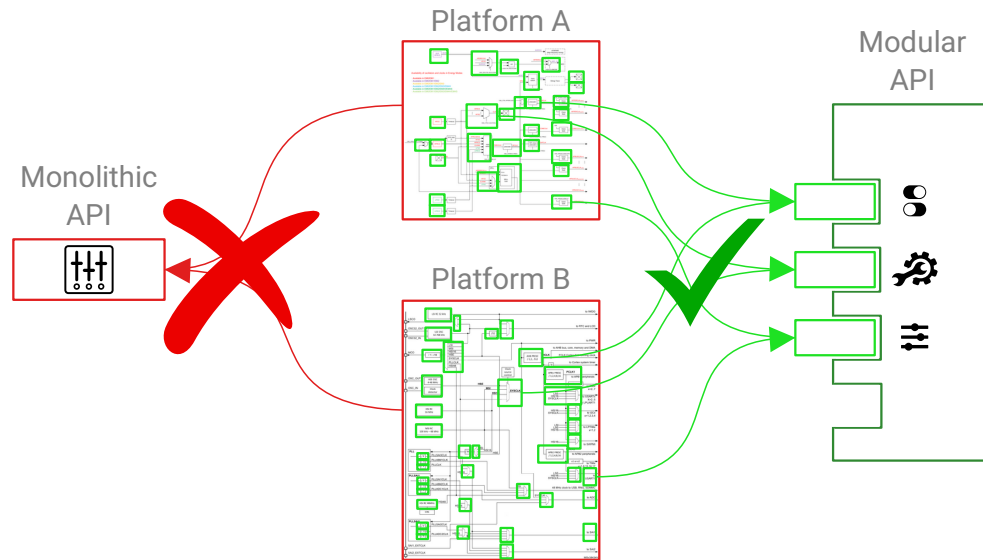
Dynamic Clock Reconfiguration for the Constrained IoT and its Application to Energy-efficient Networking

Our Approach: ScaleClock

Clock Tree Abstraction in ScaleClock

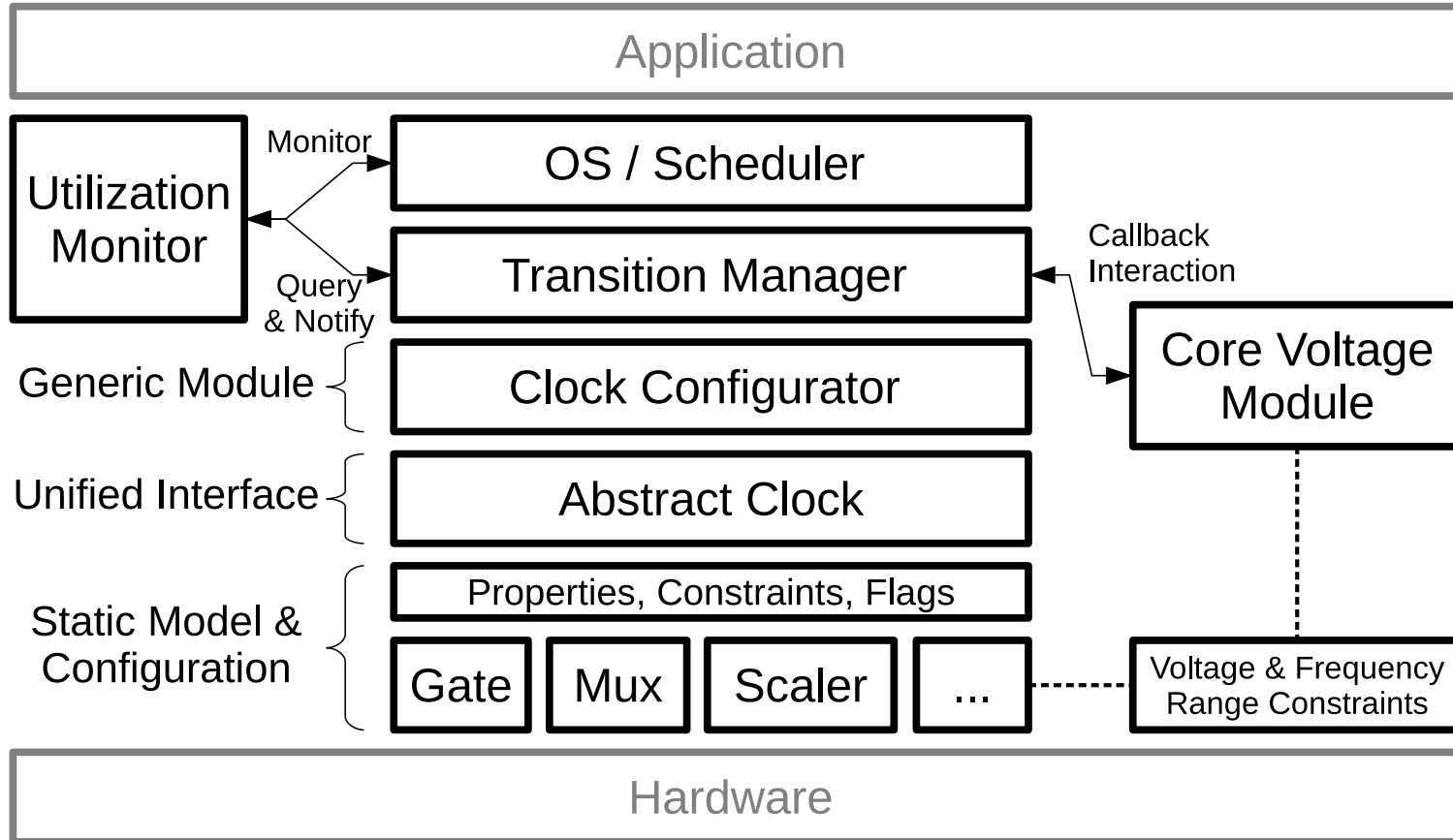


Clock Tree Abstraction in ScaleClock



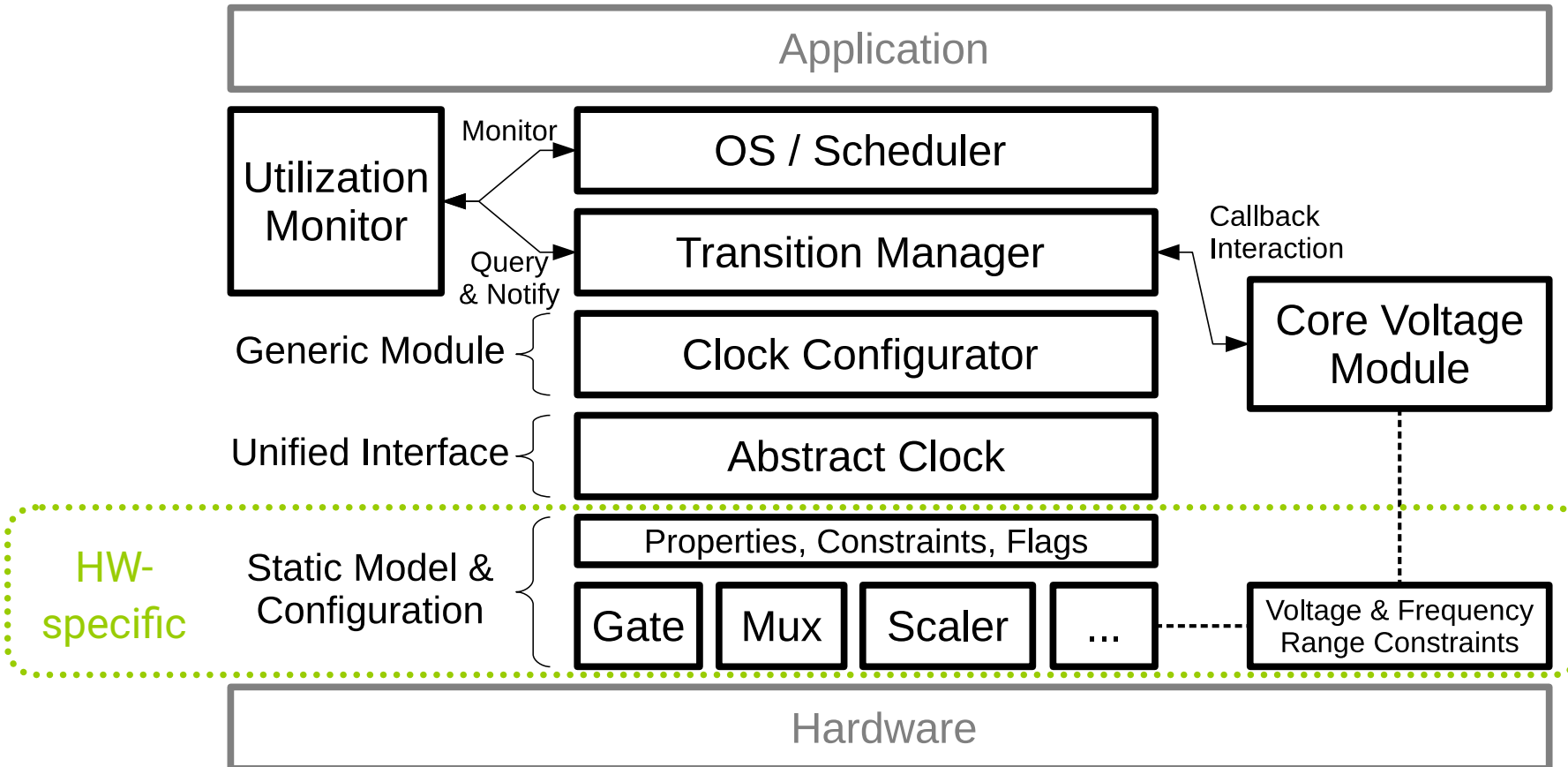


ScaleClock Architecture



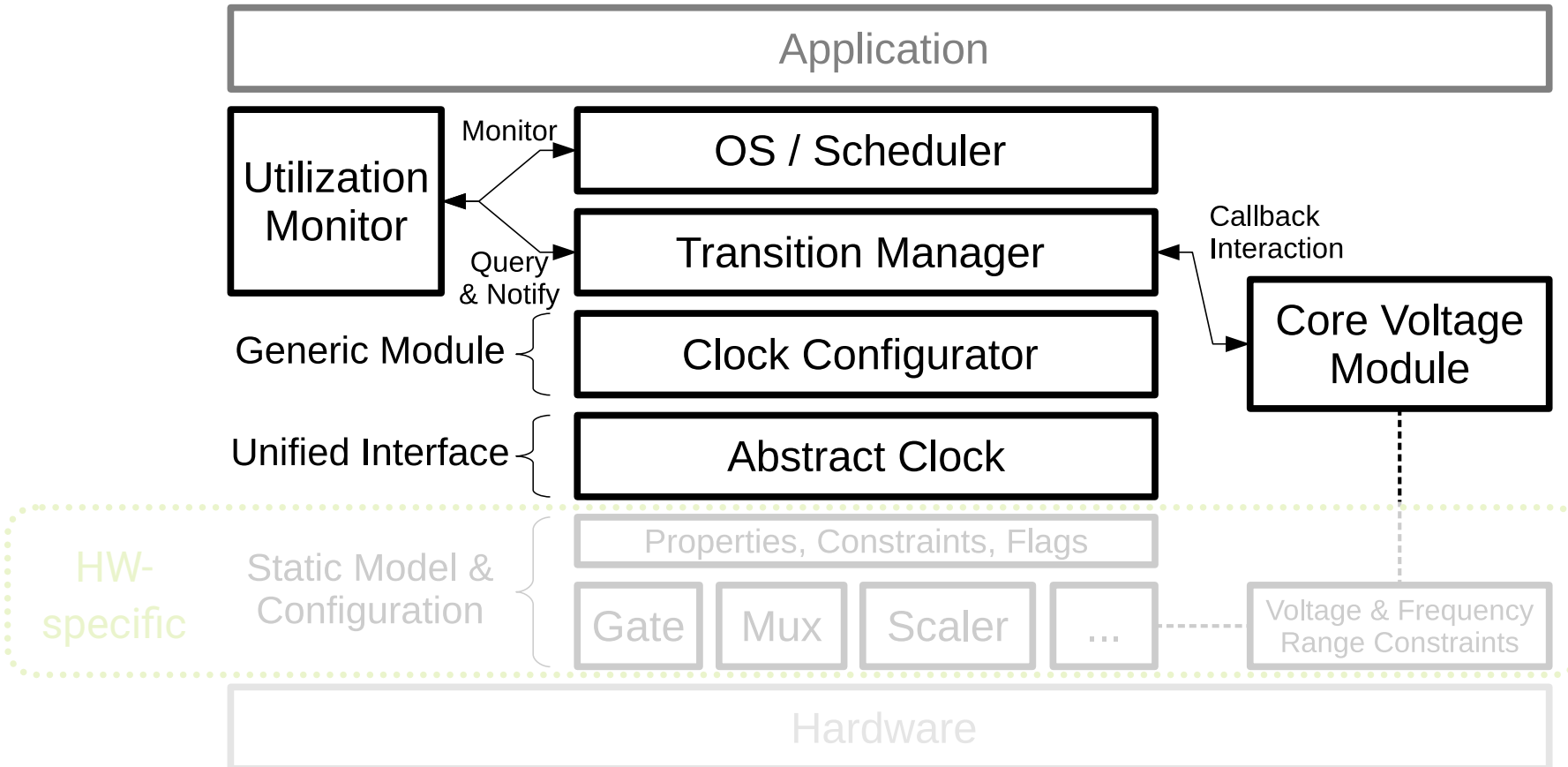


ScaleClock Architecture



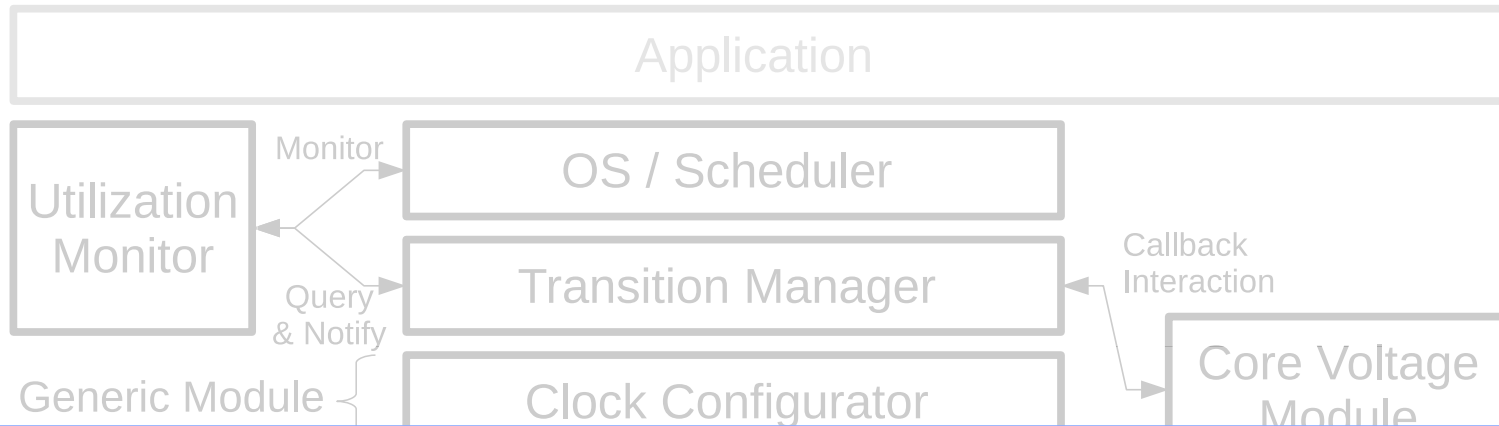


ScaleClock Architecture





ScaleClock Architecture



How to provide dynamic clock configuration with unified tooling?





ScaleClock Architecture



How to detect bottlenecks to save energy?

Performance Utilization Metric

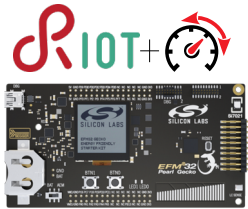
$$Load = \frac{t_{busy}}{t_{busy} + t_{idle}} \quad \times$$

$$PU = \frac{t_{busy}(F_1)}{t_{busy}(F_2)} \cdot \frac{F_1}{F_2} \quad \checkmark$$

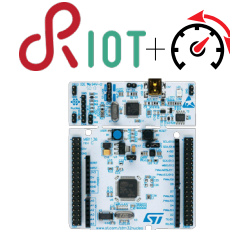
How to detect bottlenecks to save energy?



Keithley DMM7510



Silicon Labs
EFM32 slstk3402a

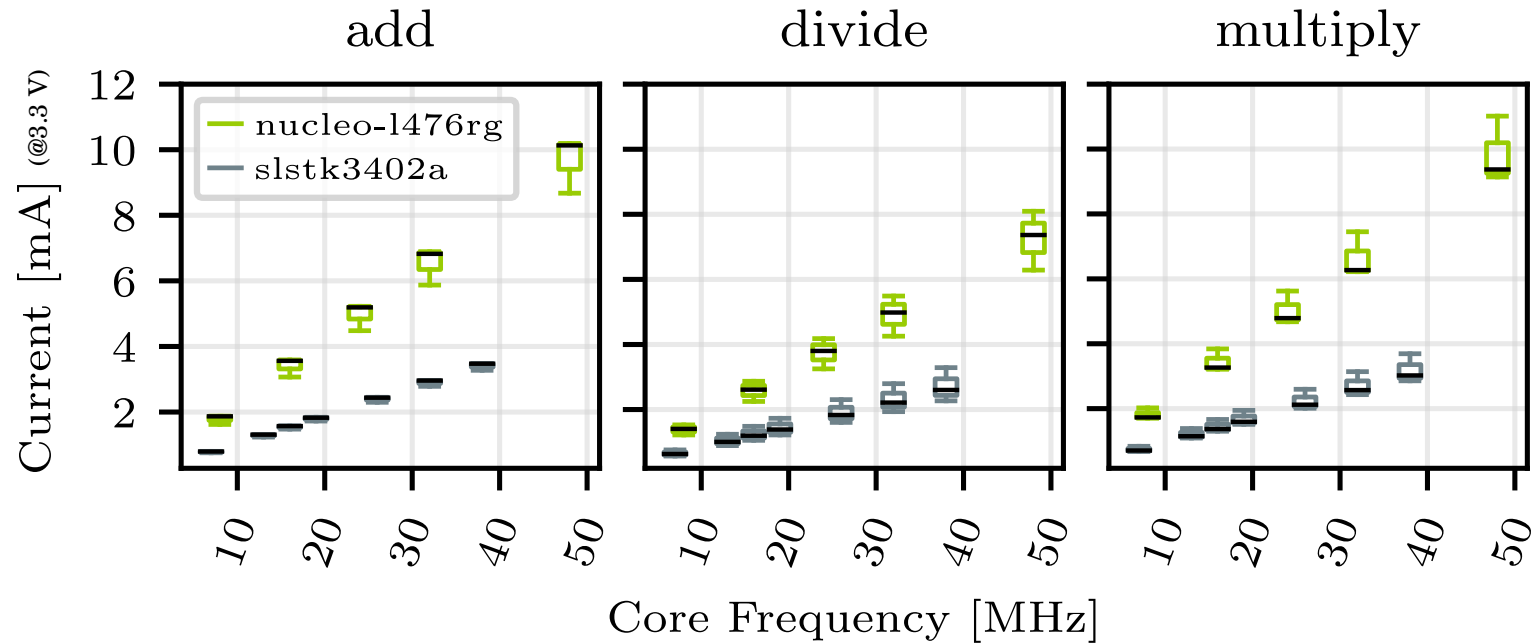


STMicroelectronics
nucleo-l476rg

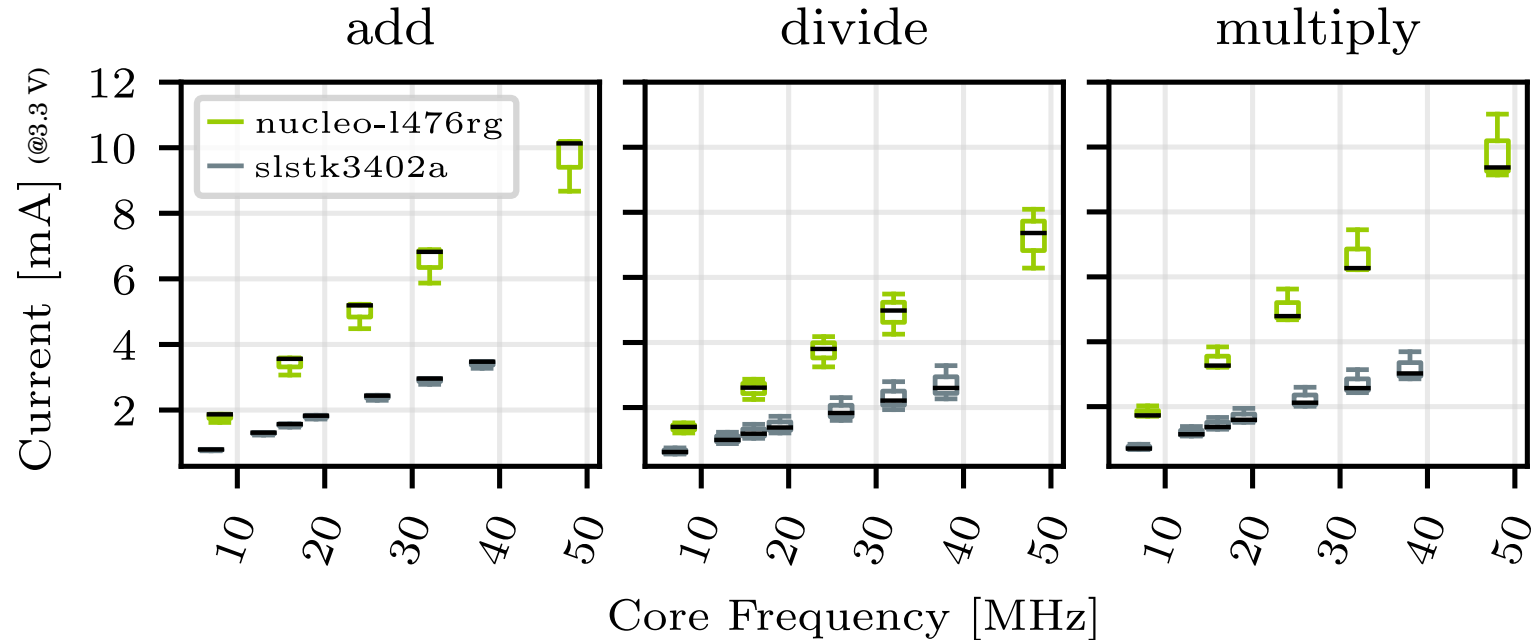
Dynamic Clock Reconfiguration for the Constrained IoT and its Application to Energy-efficient Networking

Experiments & Evaluation Results

Power Reduction



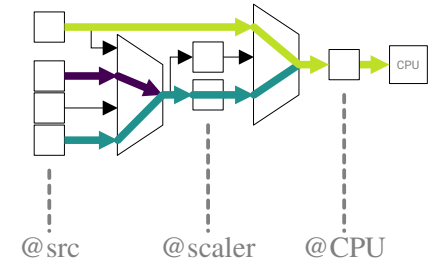
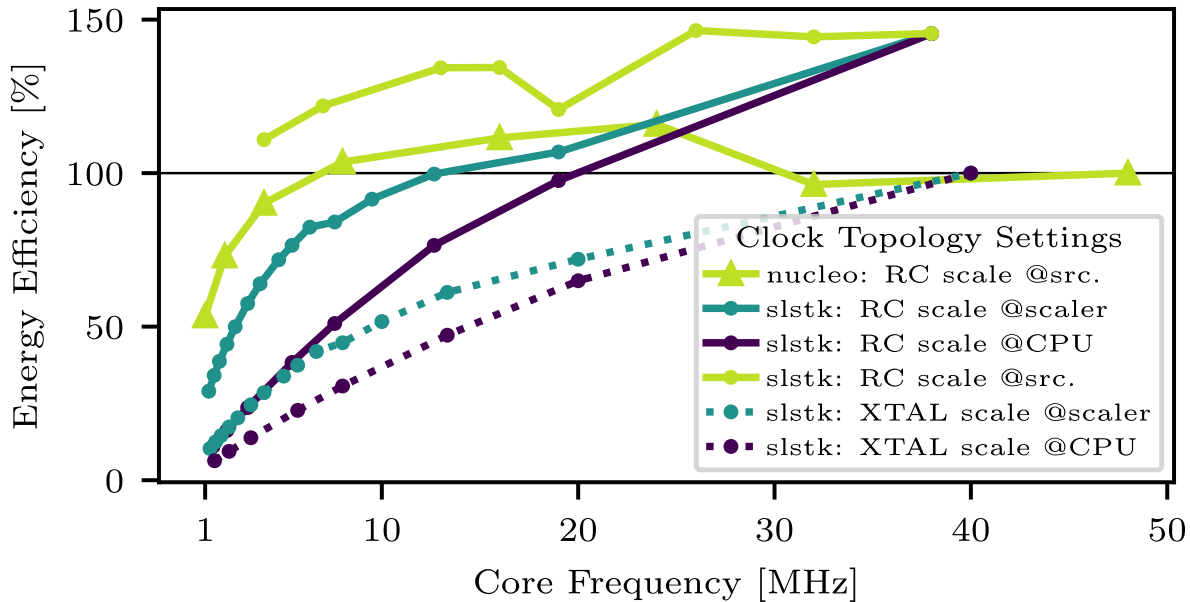
Power Reduction



- Static power offset and slope differ between platforms
- Instruction types impact consumption considerably

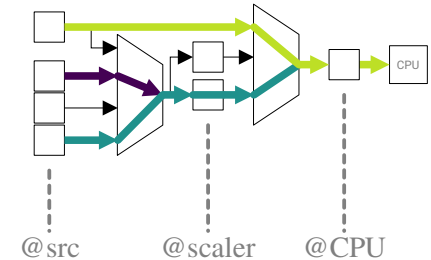
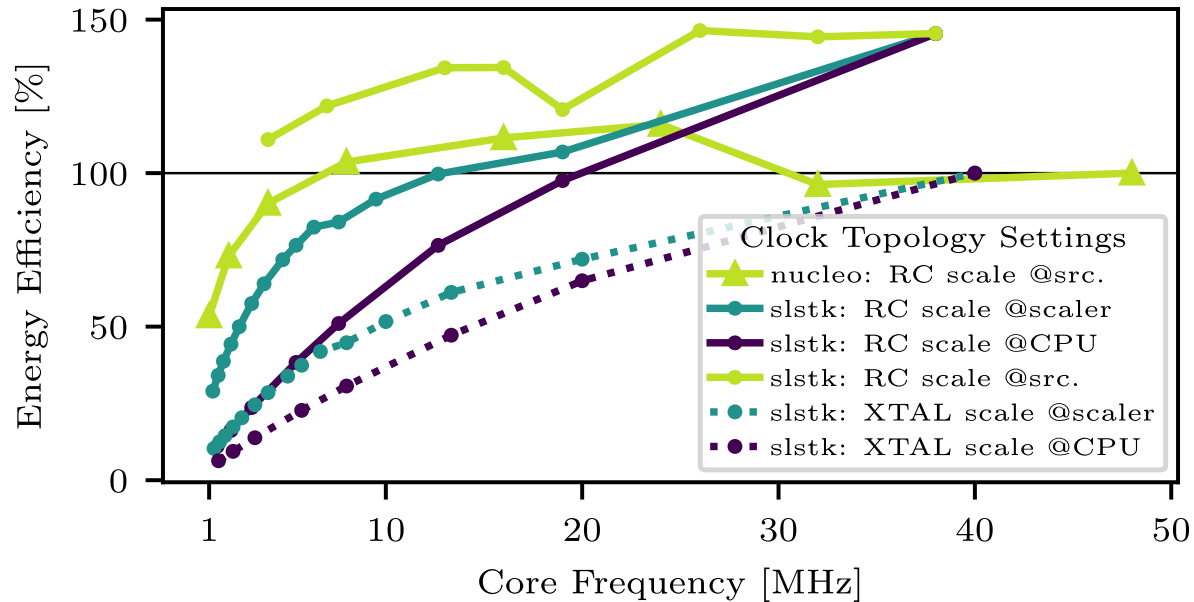
Topology Impact

(same task at different clock topology and frequency; compared to default)



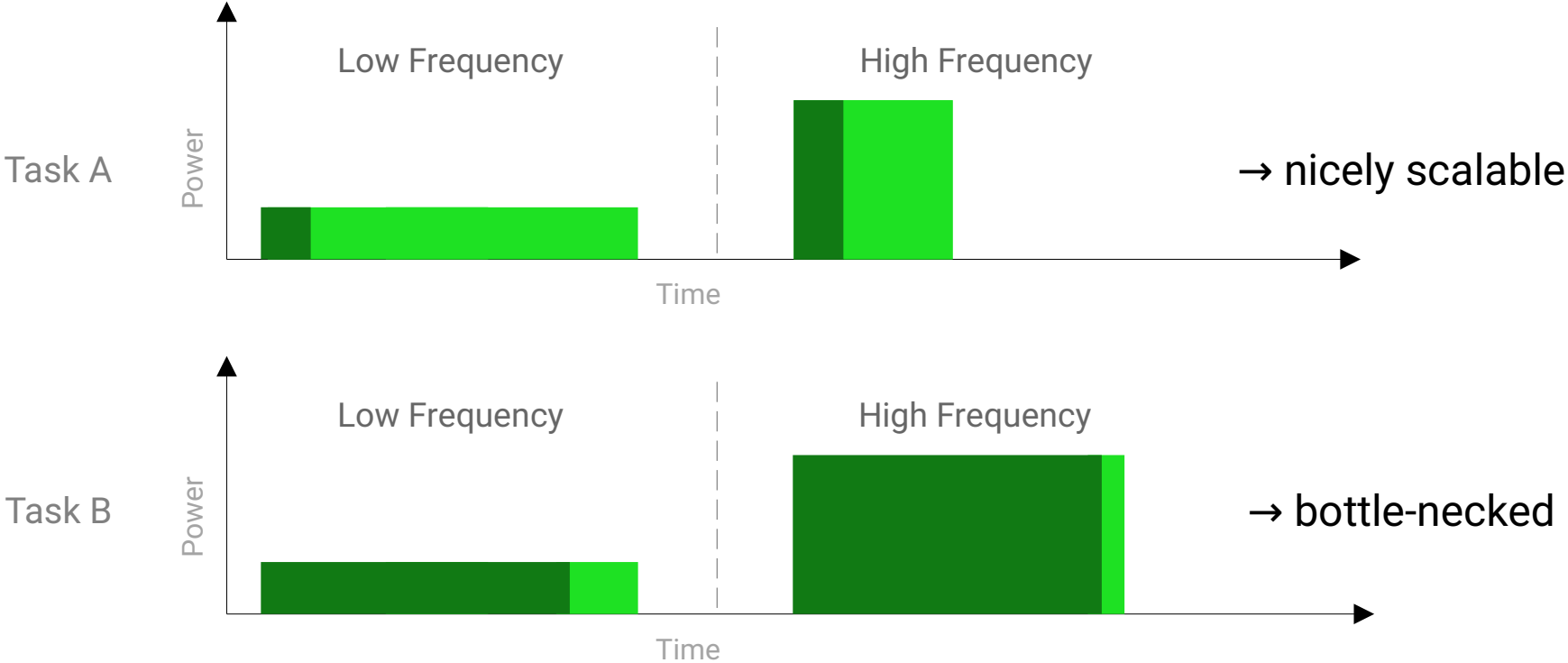
Topology Impact

(same task at different clock topology and frequency; compared to default)

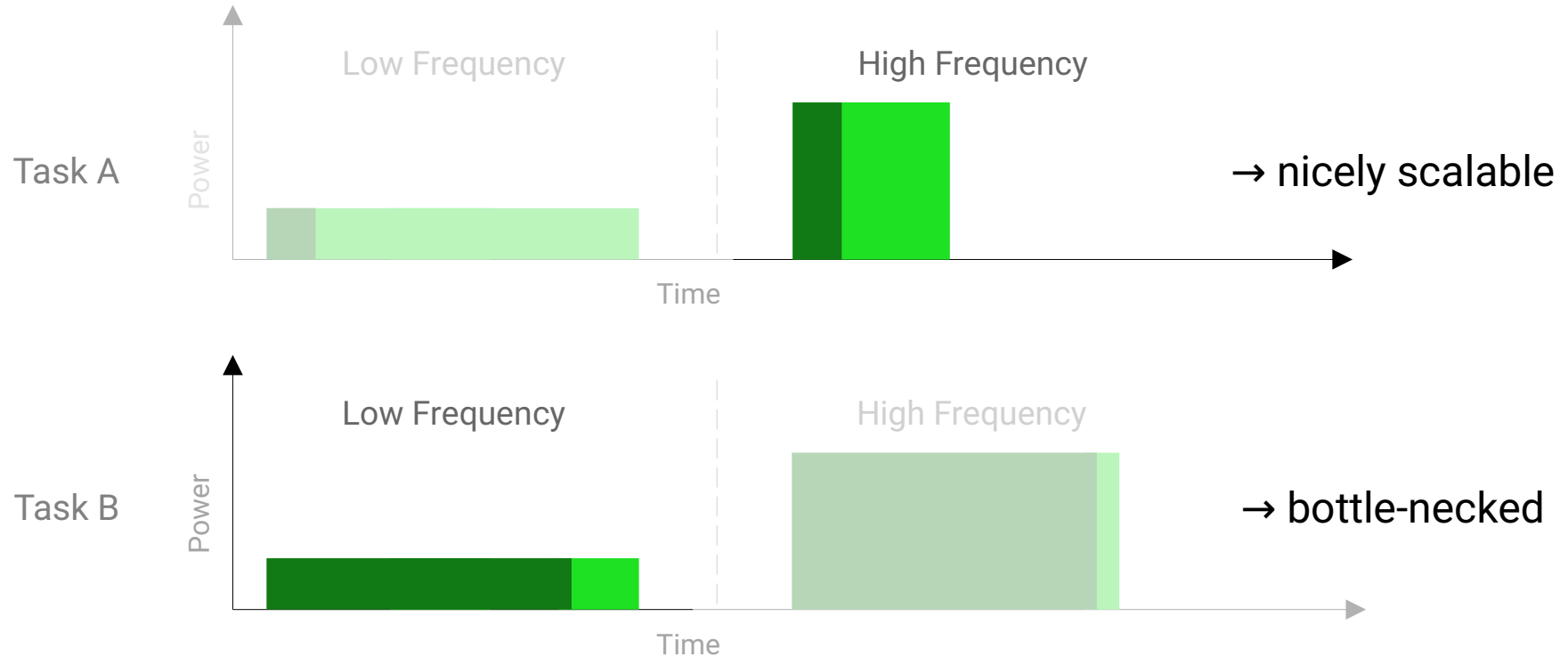


- Explicit topology control is needed due to its significant impact
- Scaling closer to the source is preferable

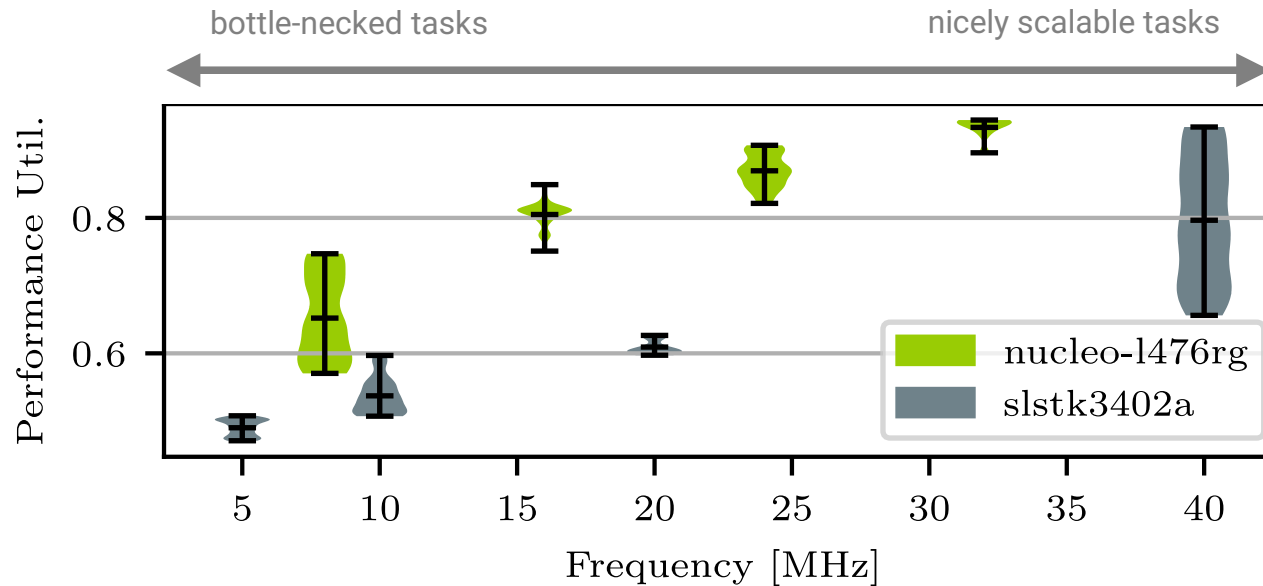
Task Scalability



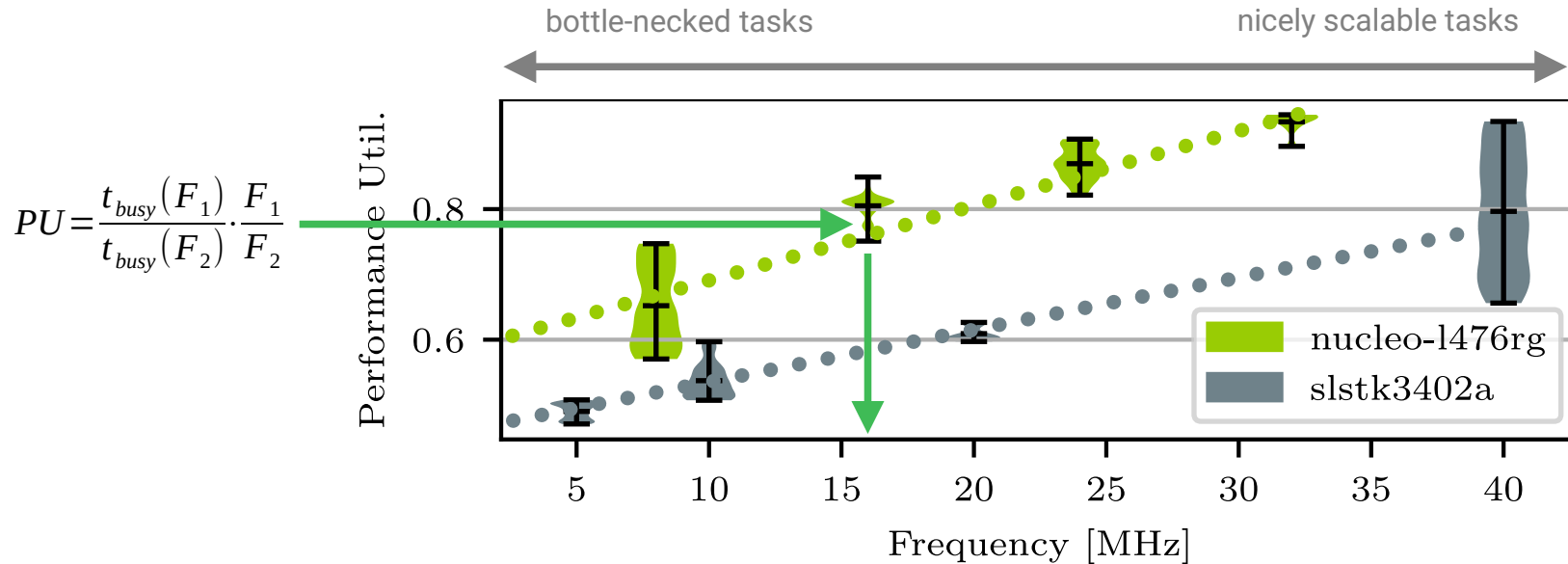
Task Scalability



Task-specific Performance Utilization



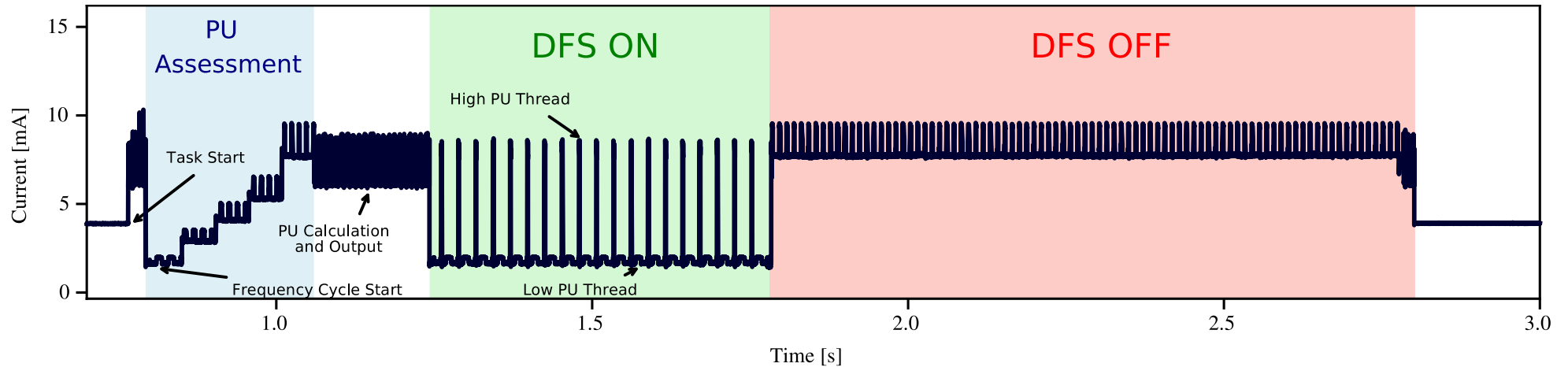
Task-specific Performance Utilization



→ Online assessed PU metric tracks energy-optimal frequency

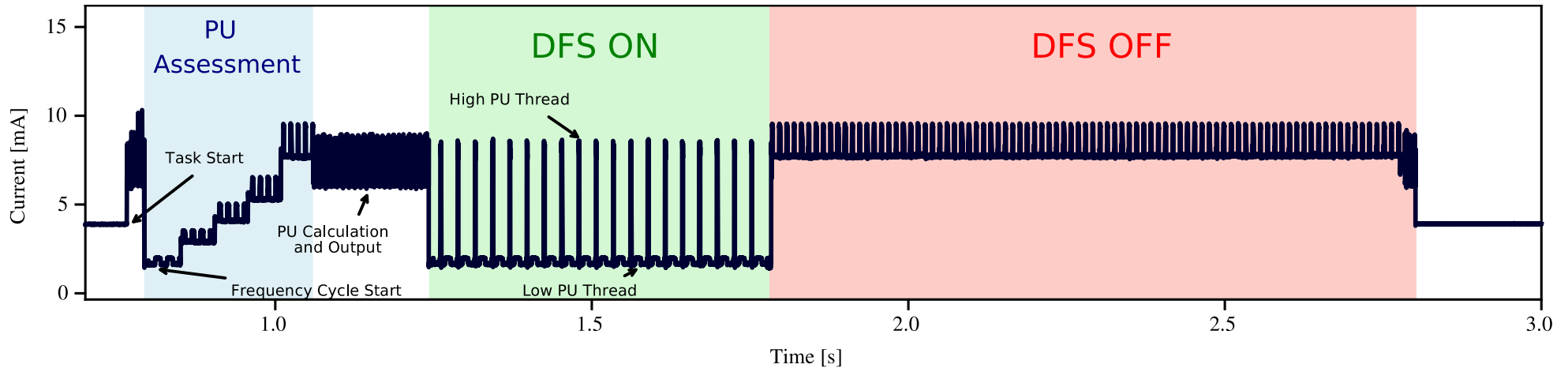
PU-based DFS Control of Multithreaded Applications

Two tasks, one with low PU value (acquisition) and one with high PU value (processing)

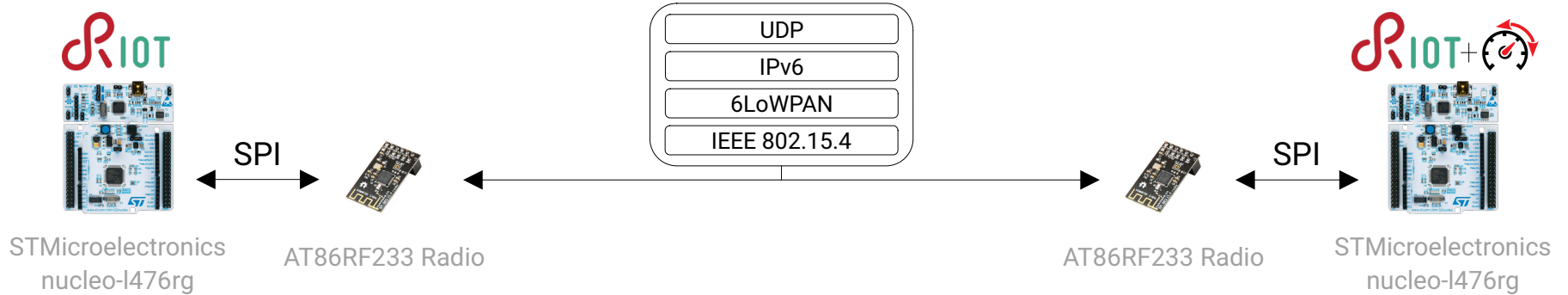


PU-based DFS Control of Multithreaded Applications

Two tasks, one with low PU value (acquisition) and one with high PU value (processing)



→ Performance Utilization metric serves as viable frequency control input



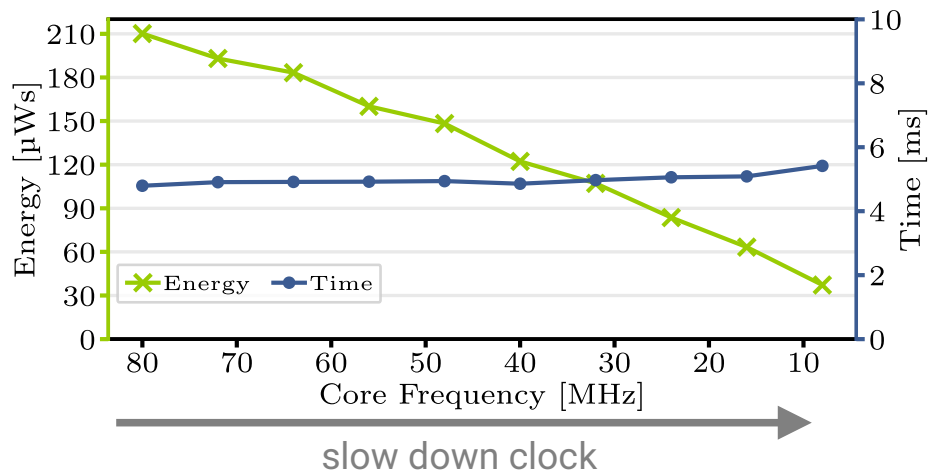
Dynamic Clock Reconfiguration for the Constrained IoT and its **Application to Energy-efficient Networking**

Case Study Results

Energy-efficient Networking

(Frequency Impact)

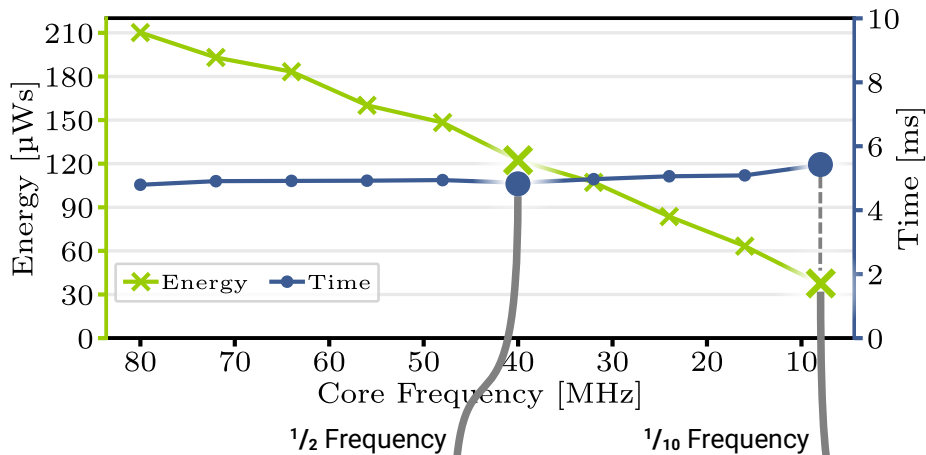
TX, 64 Bytes UDP



Energy-efficient Networking

(Frequency Impact)

TX, 64 Bytes UDP



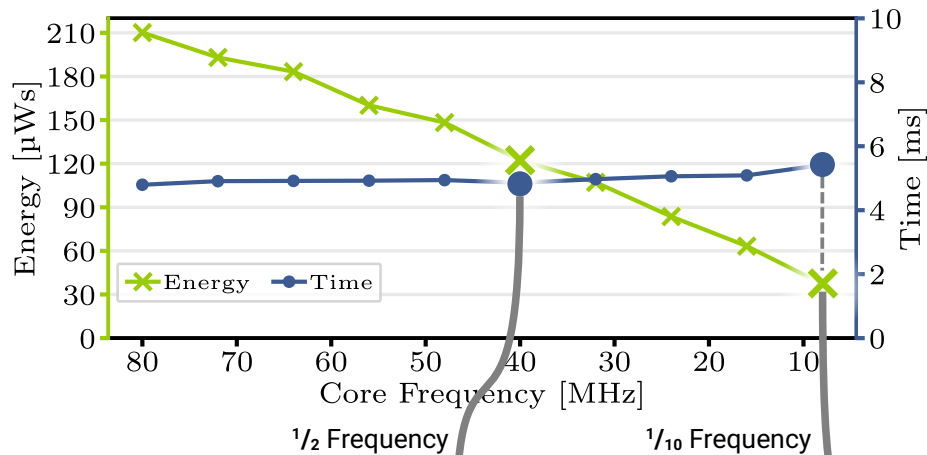
Energy: -42% Time: +1%

Energy: -82% Time: +14%

Energy-efficient Networking

(Frequency Impact)

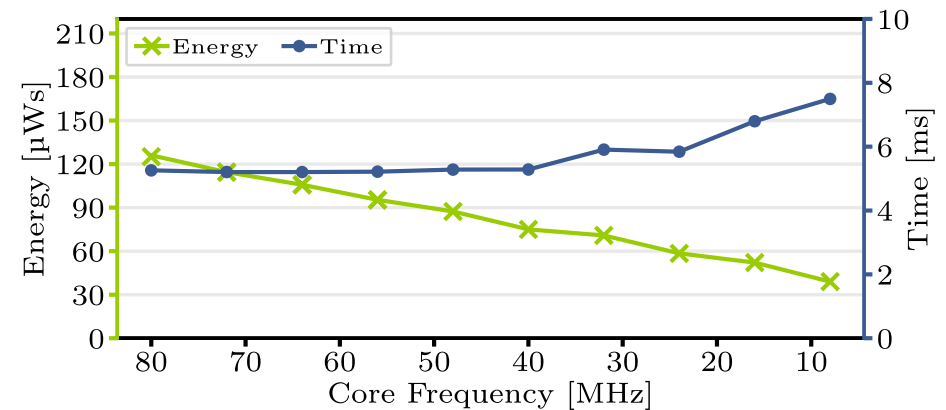
TX, 64 Bytes UDP



Energy: -42% Time: +1%

Energy: -82% Time: +14%

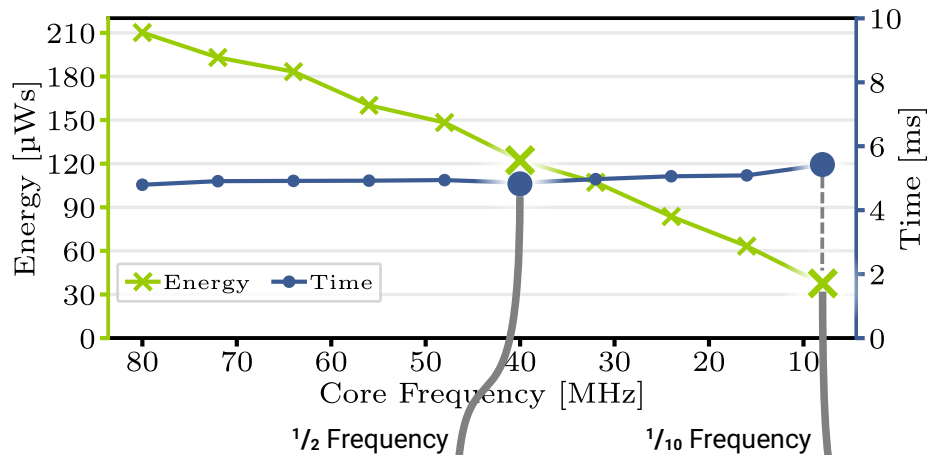
RX, 64 Bytes UDP



Energy-efficient Networking

(Frequency Impact)

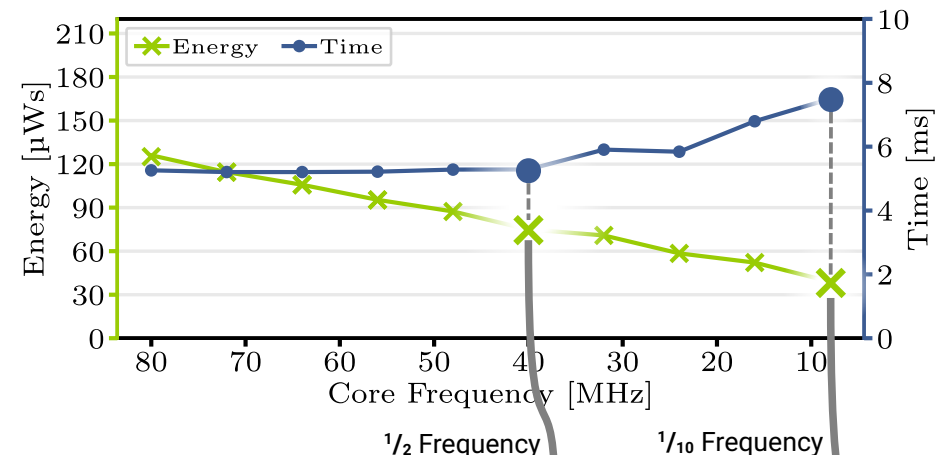
TX, 64 Bytes UDP



Energy: -42% Time: +1%

Energy: -82% Time: +14%

RX, 64 Bytes UDP



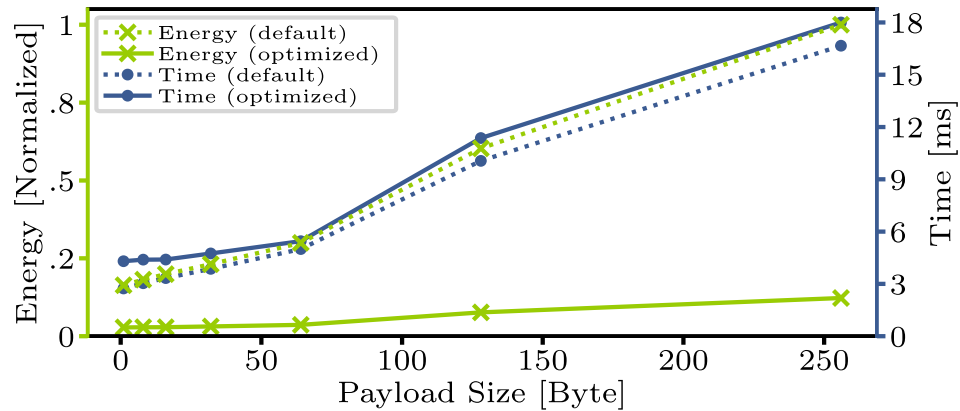
Energy: -40% Time: +2%

Energy: -69% Time: +44%

Energy-efficient Networking

(Payload Size Impact; Energy-optimized vs. default; Normalized to max payload)

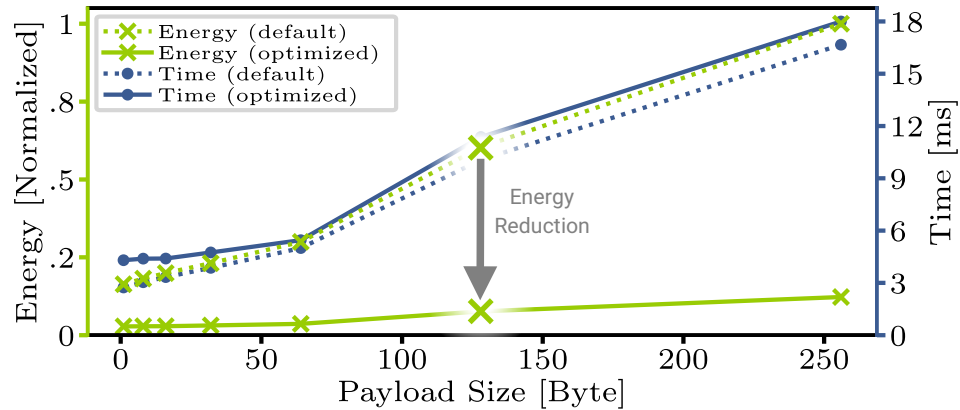
TX



Energy-efficient Networking

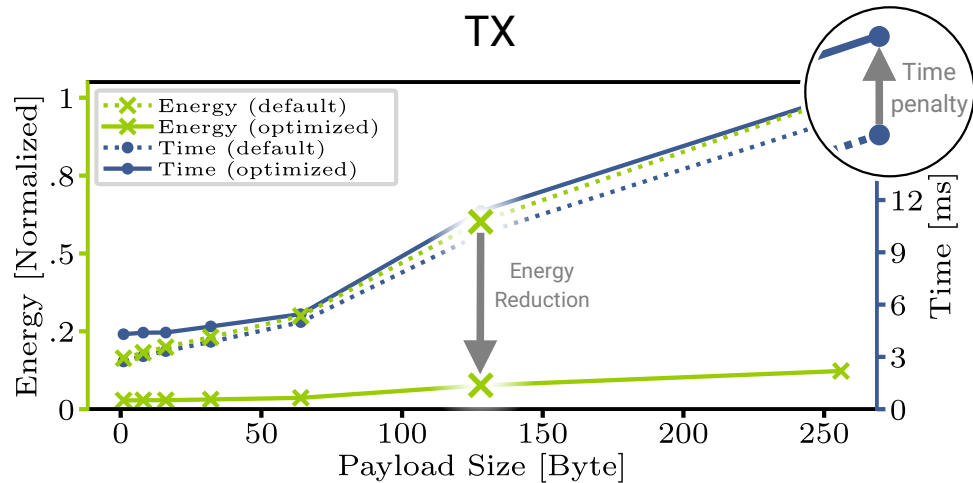
(Payload Size Impact; Energy-optimized vs. default; Normalized to max payload)

TX



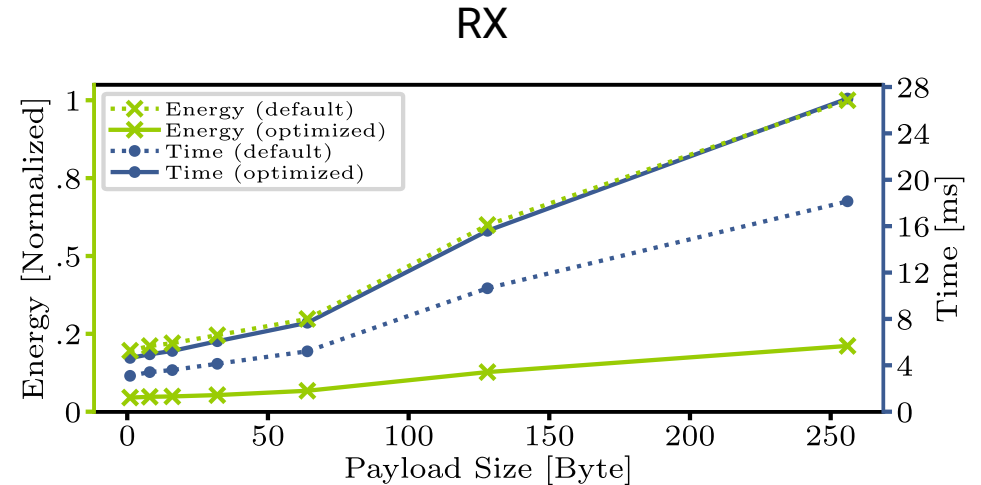
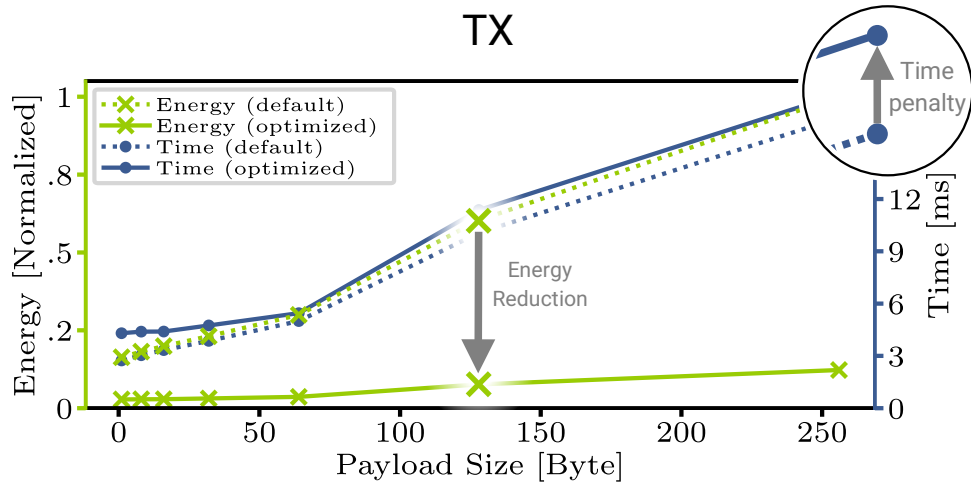
Energy-efficient Networking

(Payload Size Impact; Energy-optimized vs. default; Normalized to max payload)



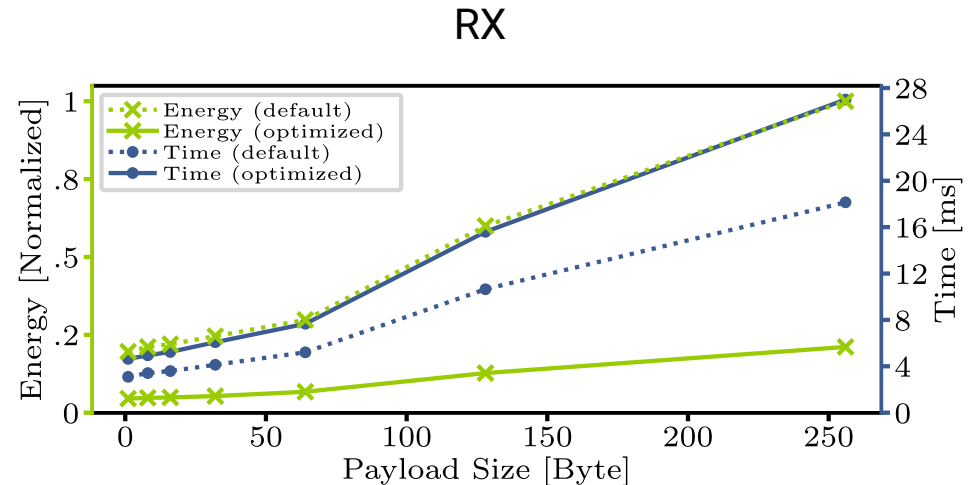
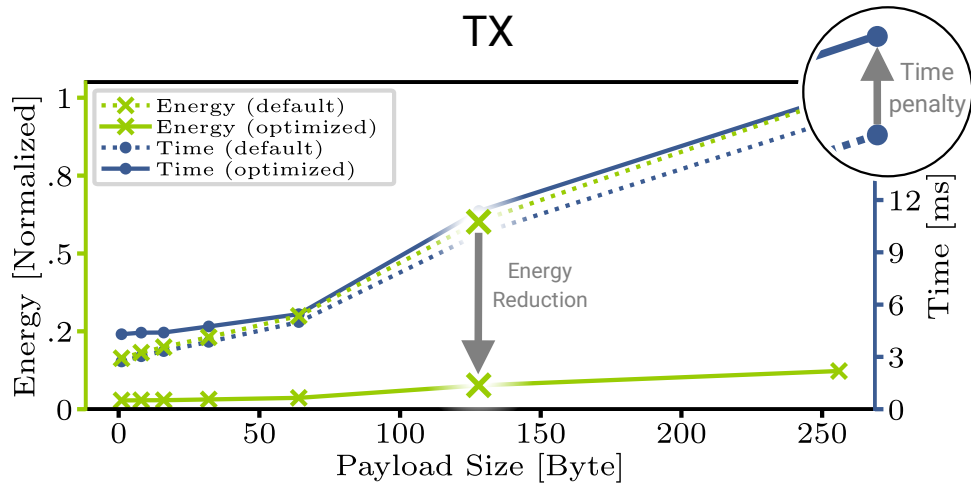
Energy-efficient Networking

(Payload Size Impact; Energy-optimized vs. default; Normalized to max payload)



Energy-efficient Networking

(Payload Size Impact; Energy-optimized vs. default; Normalized to max payload)



- Optimization beneficial for all UDP payload sizes
- Temporal performance impact higher for RX



Freie Universität



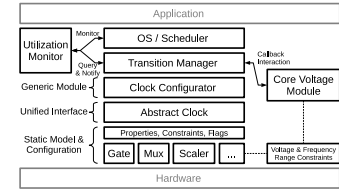
Berlin

Conclusion & Future Work

What did we learn and what's next?

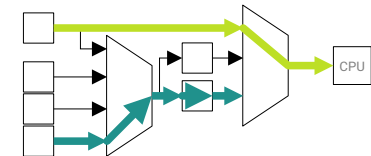
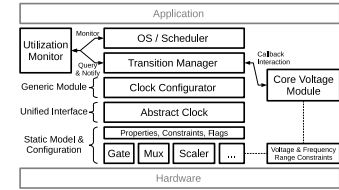
Conclusion

- Generic clock configuration feasible for common IoT platforms
 - Enables self-optimization for energy-aware systems



Conclusion

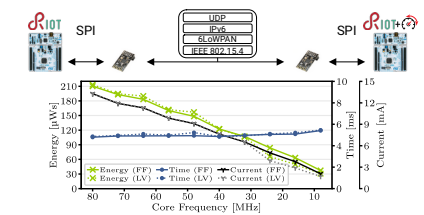
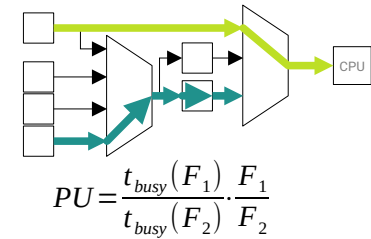
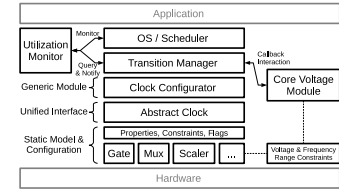
- Generic clock configuration feasible for common IoT platforms
 - Enables self-optimization for energy-aware systems
- Frequency scaling is not enough
 - Voltage and topology control offer significant benefits
 - Online PU-assessment for task specific performance



$$PU = \frac{t_{busy}(F_1)}{t_{busy}(F_2)} \cdot \frac{F_1}{F_2}$$

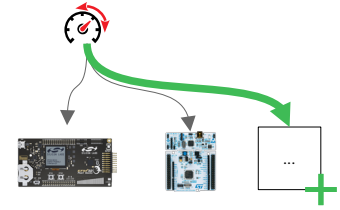
Conclusion

- Generic clock configuration feasible for common IoT platforms
 - Enables self-optimization for energy-aware systems
- Frequency scaling is not enough
 - Voltage and topology control offer significant benefits
 - Online PU-assessment for task specific performance
- For energy-efficient networking
 - ... 40% energy can be saved without noticeable performance impact
 - ... 80% energy can be saved in case of non-critical timing



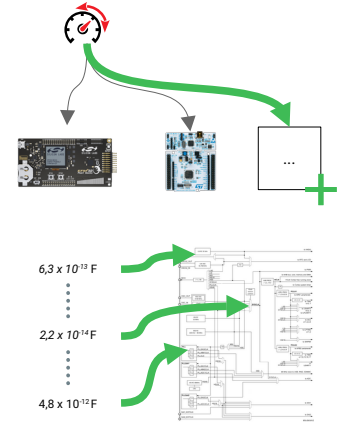
Future Work

- More platforms & applications



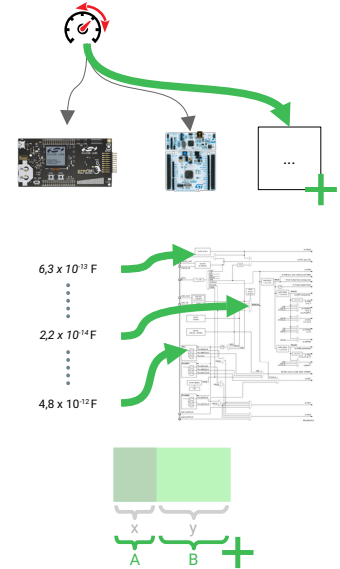
Future Work

- More platforms & applications
- Parametric power model for the clock subsystem
 - Determine parameters automatically



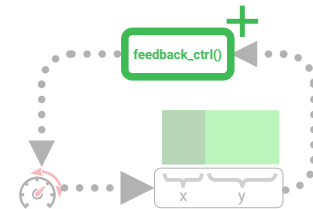
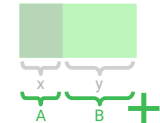
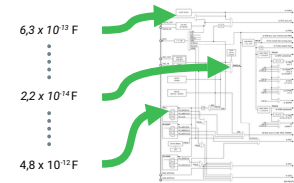
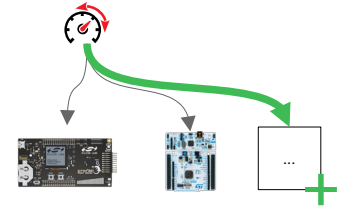
Future Work

- More platforms & applications
- Parametric power model for the clock subsystem
 - Determine parameters automatically
- Optimize task characterization



Future Work

- More platforms & applications
- Parametric power model for the clock subsystem
 - Determine parameters automatically
- Optimize task characterization
- Integration of different control mechanisms
 - Threshold selection, PID, AI, ...





Freie Universität



Berlin

Questions & Discussion



ScaleClock Sources

 <https://github.com/inetrg/RIOT/tree/ScaleClock>

Related Websites

Internet Technologies research group | <https://inet.haw-hamburg.de/>

RIOT OS | <https://www.riot-os.org/>

Contact

Michel Rottleuthner | michel.rottleuthner@haw-hamburg.de

Thomas C. Schmidt | t.schmidt@haw-hamburg.de

Matthias Wählisch | m.waehlich@fu-berlin.de