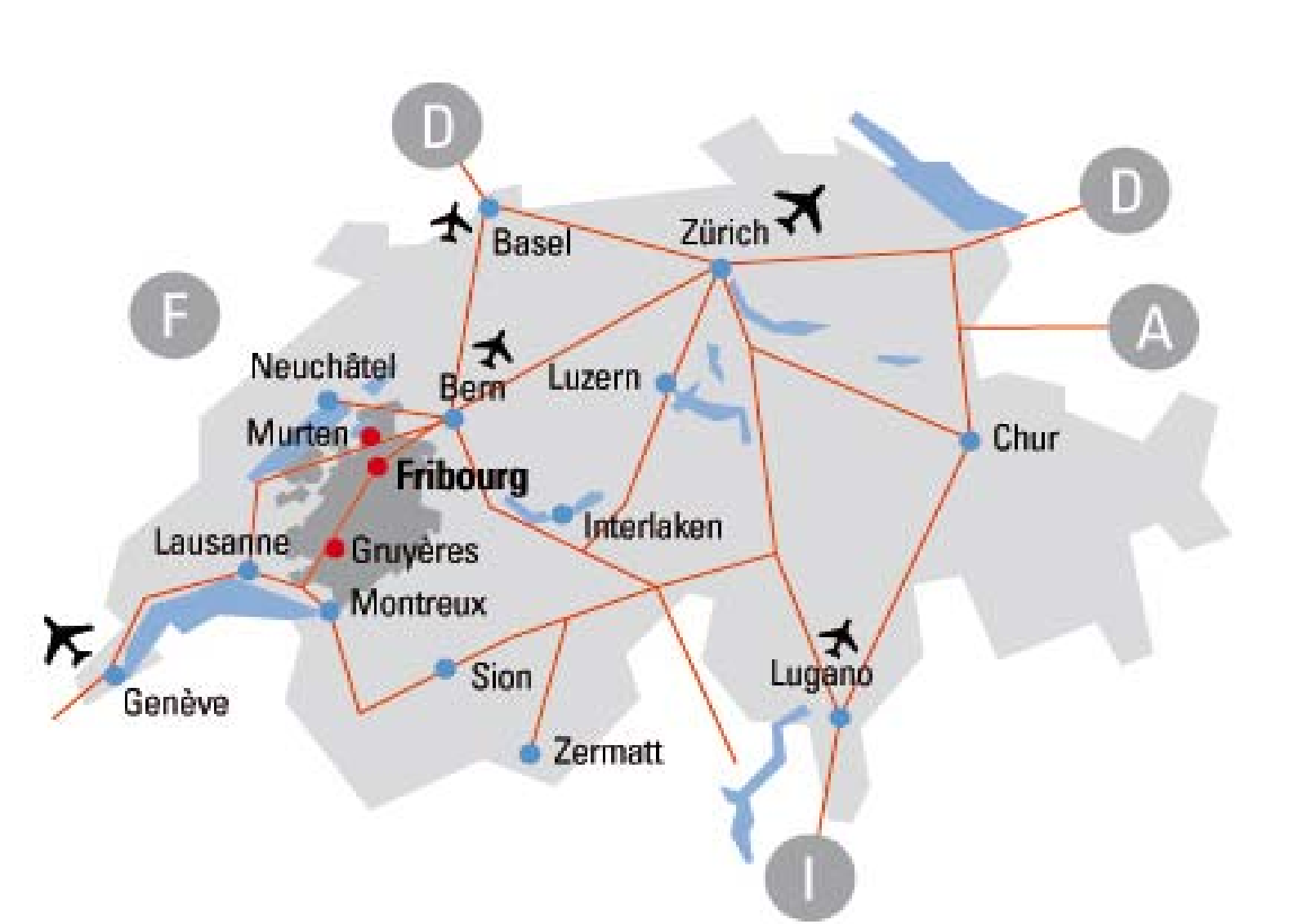
Einführung in Low-Power Linux

Wolfram Luithardt und Sylvain Décastel

**Hochschule für Technik und Architektur,
Fribourg, Schweiz**

Wo kommen wir her ?



Inhaltsverzeichnis

Low Power Linux

Server: Performance ist wichtiger als Energie-verbrauch. Low Power primär aus Kostengründen

Desktop: Performance ist oft wichtiger als Low Power

Laptops: Low Power primär zur Verlängerung der Batterielaufzeit

Mobile: Batterielaufzeit

Embedded: Oft nur wenig Energie vorhanden.

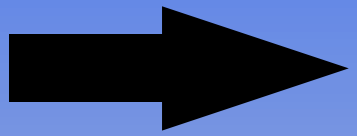
Motivation

Embedded System Roadmap 2002:

"Power is considered as the most important constraint in embedded systems"

Energy-Harvesting:

- Photovoltaik
- Thermoelektrik
- Vibrationsgeneratoren
- Mikro-Luftgeneratoren



Leistungsabgaben an ein System ca: 100 uW – 10mW

Langfristiges Ziel:
Ein Linuxrechner, der mit einer Energyharvesting-Versorgung gespeist wird.

Neue Low Power Komponenten

- Display: **bistabiles LCD**: hält sein Bild auch im stromlosen Zustand
- Ram: **PASR (Partitial Array Self Refresh)**: dynamisches Ram, das nur teilweise refreshed wird.
- Ram: **NVRam** (Nichtflüchtiges Ram): $>10^{15}$ Schreib-Löschzyklen
- Energiespeicher: **Supercaps**

Leistung einzelner PC-Komponenten

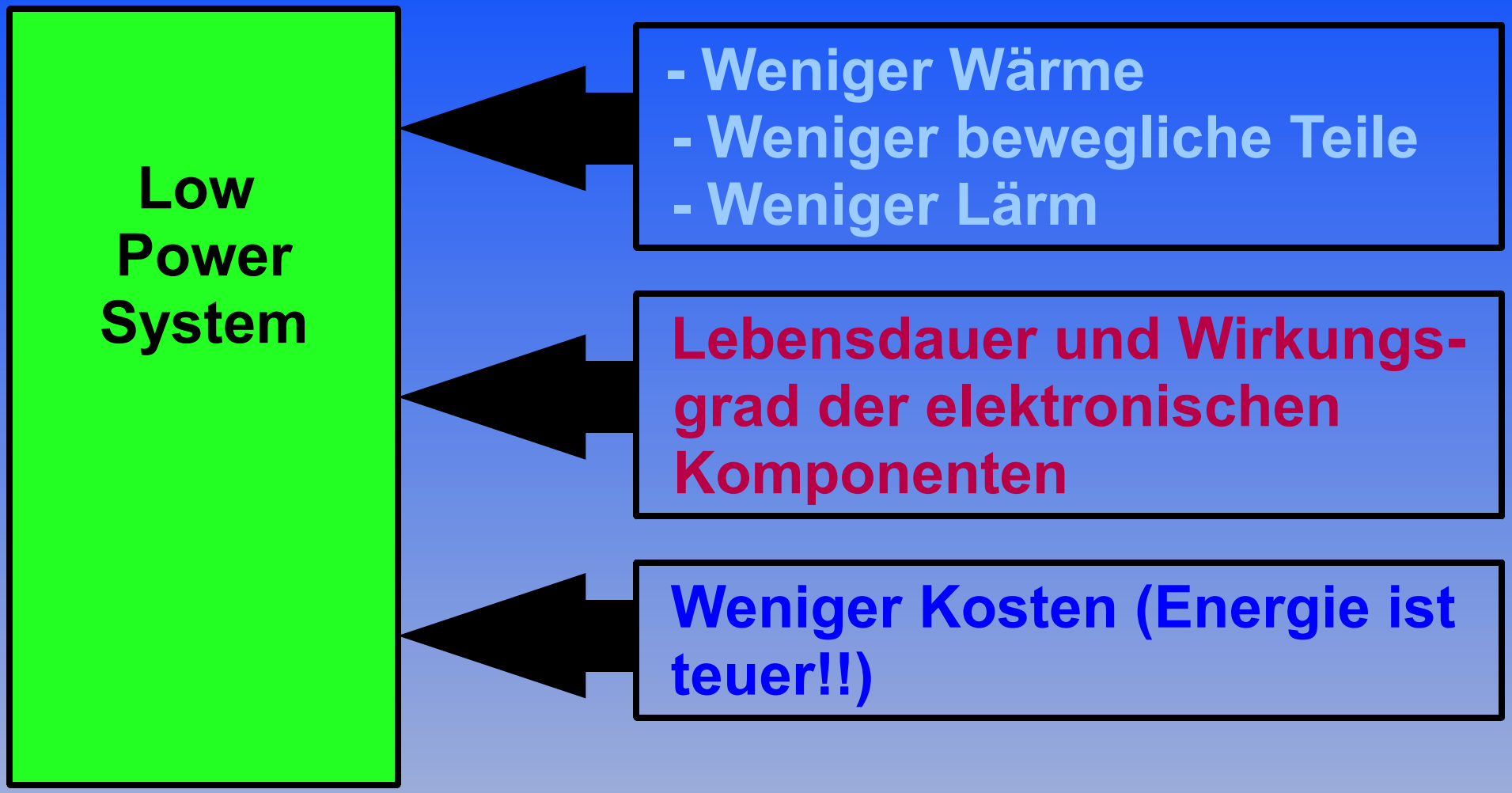
* Mainboard: 25 Watt
* Gehäuselüfter: 5 Watt
* Prozessorlüfter: 3 Watt
* Speichermodul: 10 Watt
* PCI Soundkarte/Netzwerkkarte/Modem: 5 Watt
* Festplatte: 30 Watt
* CD/DVD Laufwerk/Brenner: 30 Watt
* Diskettenlaufwerk: 4 Watt
* Tastatur/Maus/andere USB-Geräte: 3 Watt/Stück
* Grafikkarte: ca. 50-100 Watt je nach Hersteller
* Prozessor:

    Duron: 27Watt /600MHz - 80Watt /1,6 GHz

    Athlon XP: 60Watt /1500+ - 110Watt / 3200+

    Intel P4: 50Watt / 1,6GHz - 100 Watt / 3,2 Ghz

Warum Low-Power Systeme ?



Low Power Systeme sind im wahrsten Sinne des Wortes:

cool !

Bekannte Energiespar-Methoden

- Screen off-mode

- Suspend to Ram

- Suspend to disk (Hibernation mode)

Vergleich verschiedener Linux-fähiger Prozessor-familien

Intel Pentium: 120 Watt

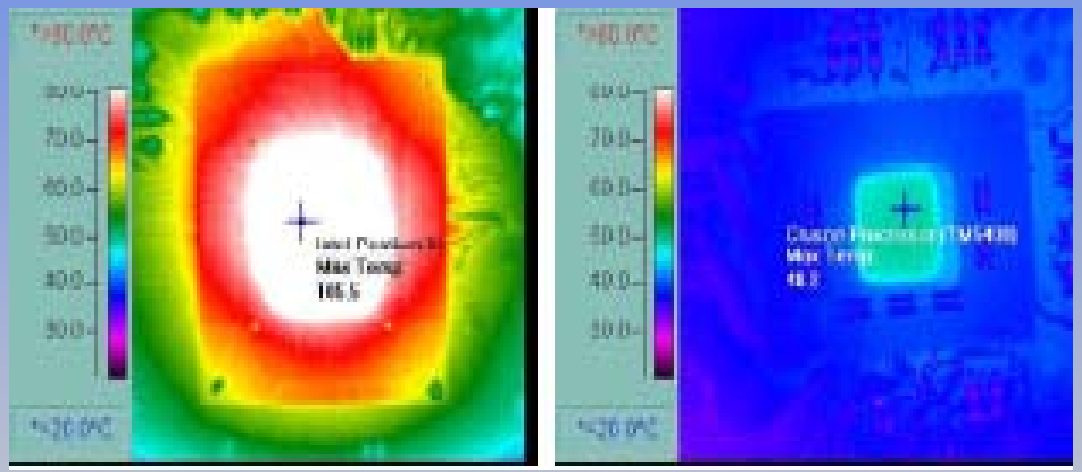
Intel Duo 2 Core (z.B. E8600): 65 Watt

Intel Atom: 2-6 Watt

Transmeta Crusoe: 1 Watt

ARM: z.B. AT91SAM9260: 200 mW

Atmel AVR32: 80mW.



Die grundlegende Gleichungen

$$P_{gesamt} = P_{statisch} + P_{dynamisch}$$

$$P_{statisch} = \Sigma\ U * I_{Leak}$$

U = Corespannung
I_{Leak} = Leckstrom

$$P_{dynamisch} = \alpha C U^2 f$$

α: Aktivitätsfaktor
C: Parasitäre Kapazität (Transistoren und Leitungen)
U: Corespannung
f: Frequenz

Technologische Entwicklungen

Von Generation zu Generation:

- **Parasitäre Kapazität/Transistor: ↓**
- **Corespannungen: ↓**
- **Taktfrequenz: ↑**
- **Anzahl-Transistoren pro Chip: ↑**

Dynamische Leistung: =/↑

Statische Leistung: ↑

Technologische Entwicklungen

zur Reduktion statischer Verluste:

$$P_{statisch} = \Sigma\ U * I_{Leak}$$

- **Neue Dielektrika**
- **DVTCMOS (Dual Voltage Threshold-CMOS):**

  Man benützt je nach Aufgabe 2 verschiedene Arten von Transistoren (schnell mit hohem I_{Leak} oder langsam mit niedrigem I_{Leak}

- **STI (Sleep Transistor Insertion): Transistoren, die für eine gewisse Funktionalität nicht benötigt werden, werden stromlos geschaltet.**

Technologische Entwicklungen

zur Reduktion dynamischer Verluste:

$$P_{dynamisch} = \alpha C U^2 f$$

- **ULV-Prozessoren**

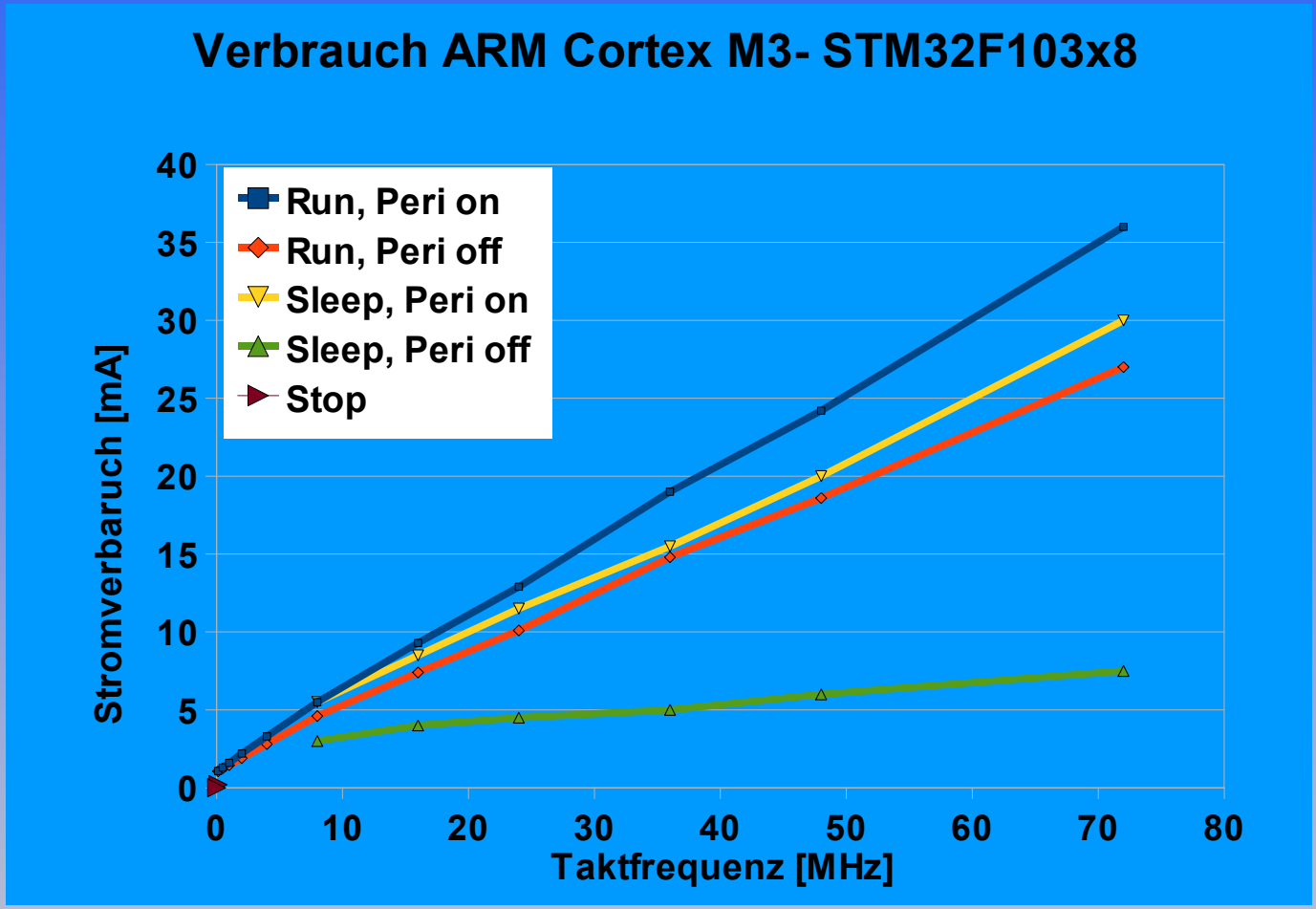
- **Clock-Gating: Optimierung der Anzahl der Transistoren, die bei einer gewissen Operation getaktet werden müssen.**

- **Cross Talk Avoidance: z. B. durch Vertauschen von Leitungen etc.**

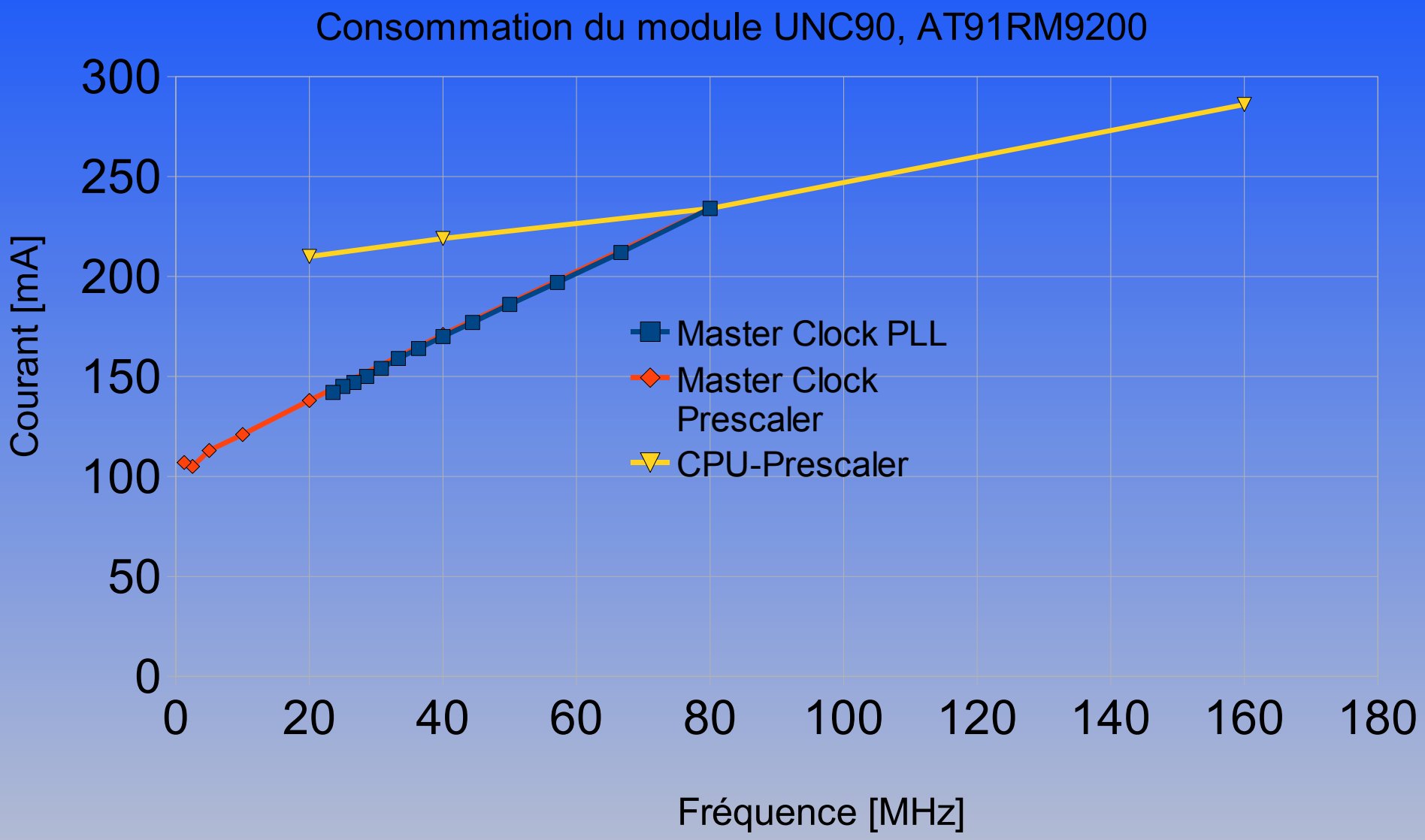
Sparmöglichkeiten

$$P_{dynamisch} = \alpha C U^2 f$$

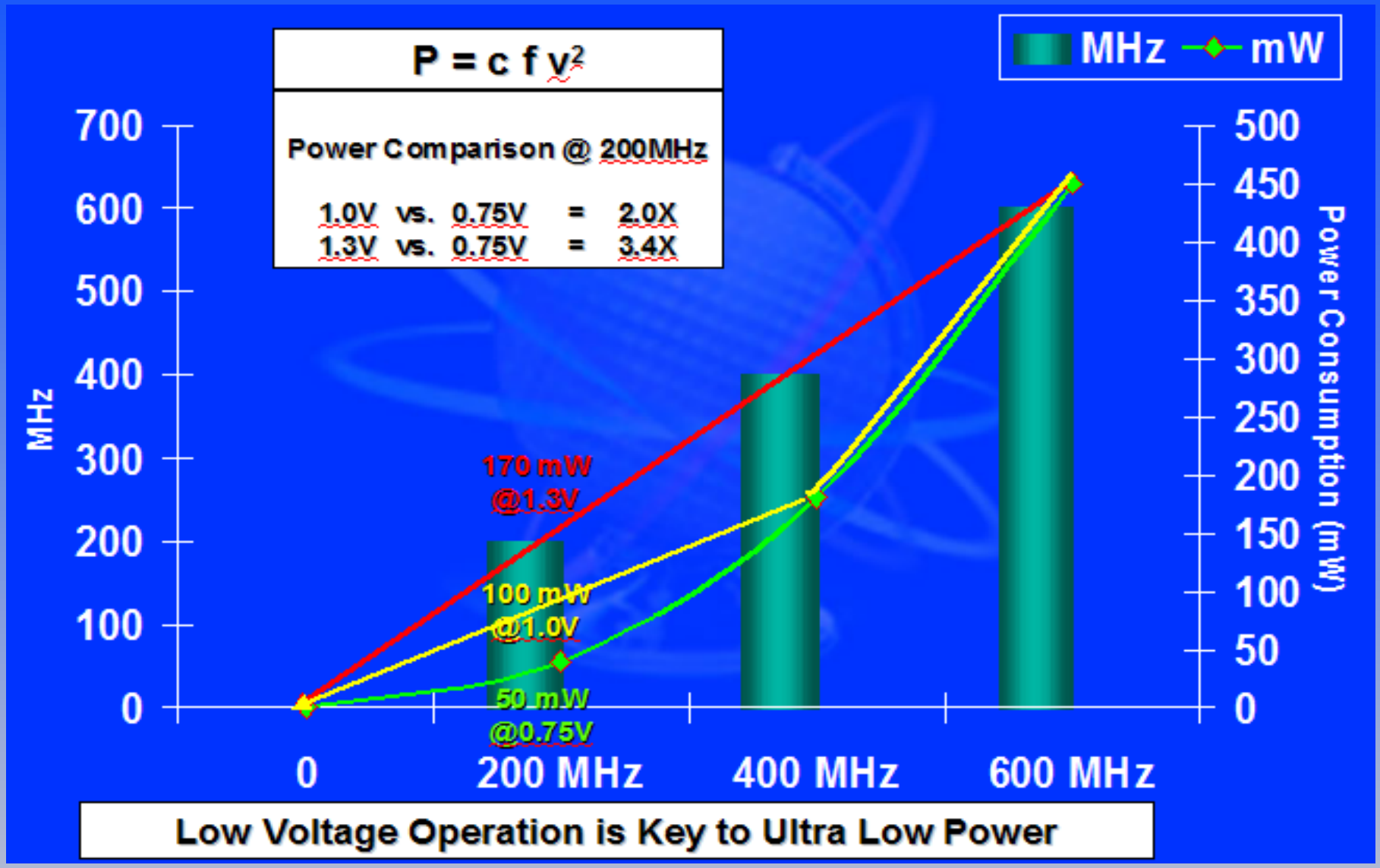
- Abschaltung von nicht benötigter Peripherie
- Reduktion der Taktfrequenz
- Reduktion der Versorgungs-spannung
- Kombination von Allem !!!!



The real world



Abhängigkeit von Spannung und Frequenz



XScale, Quelle Intel

X86 Sleep-modes

P-States:

Verringerung der Frequenz und der Corespannung

C-States:

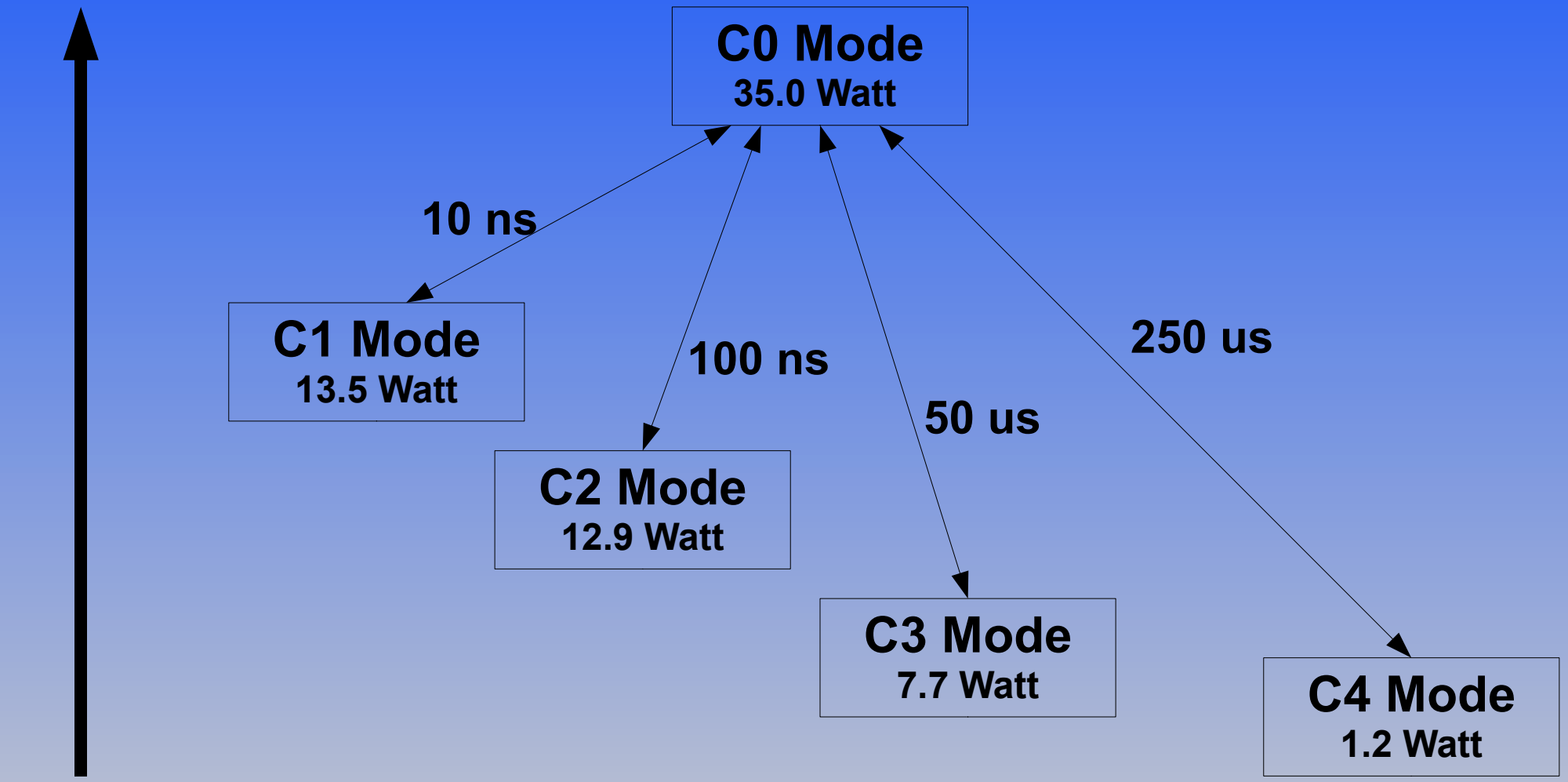
Beispiel: Intel® Core™2 Duo

C-State		max Leistung
C0	Running Mode	35.0 Watt
C1	Idle Mode	13.5 Watt
C2	Ohne Datenbus	12.9 Watt
C3	Deaktivierung PLL	7.7 Watt
C4	Enhanced Deeper Sleep	1.2 Watt

http://www.lesswatts.org/documentation/silicon-power-mgmnt/

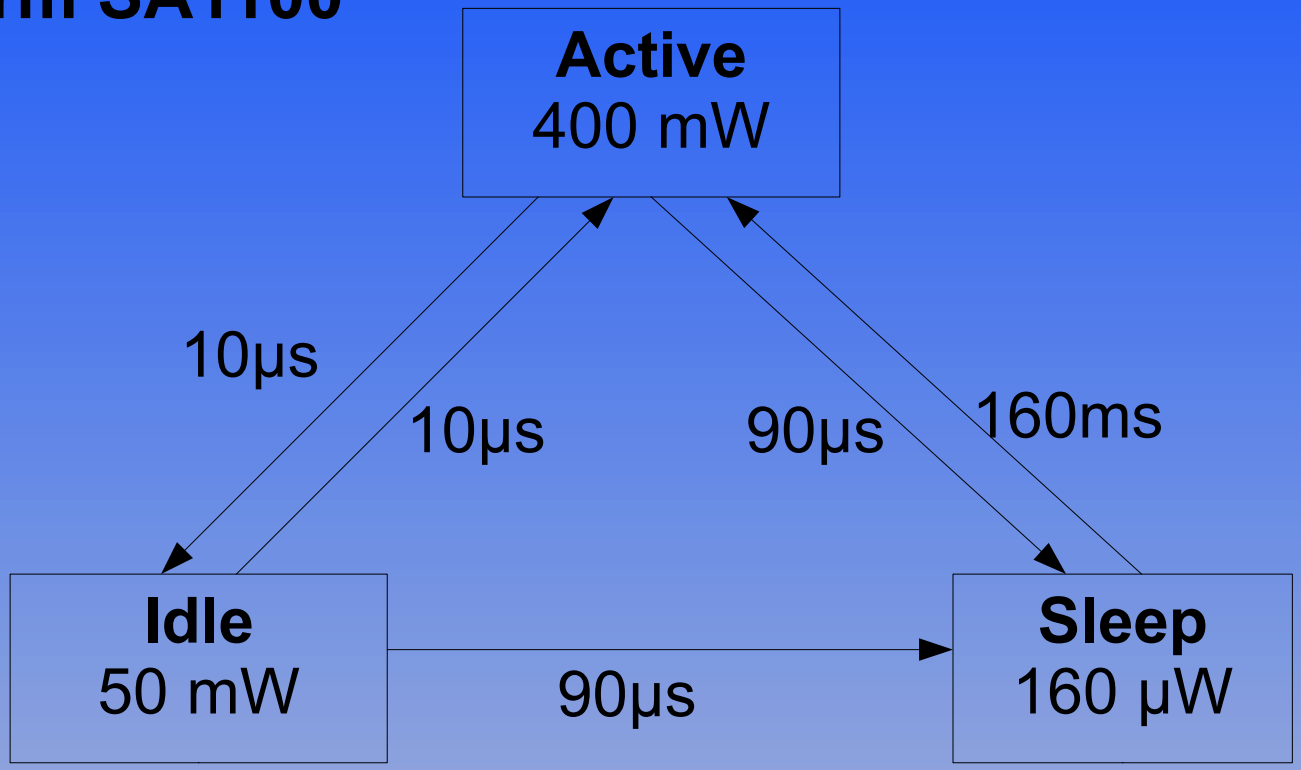
Warum verschiedene Sleep-Modes?

Beispiel: Intel® Core™2 Duo



Beispiel Sleep-Modes: ARM

Strongarm SA1100



Woher weiss der Prozessor, in welchen Zustand er gehen kann ?

Grundlegende Prinzipien

Grundzustand aktiv, schlafen nur, wenn nichts zu tun ist

Wartendes Betriebssystem

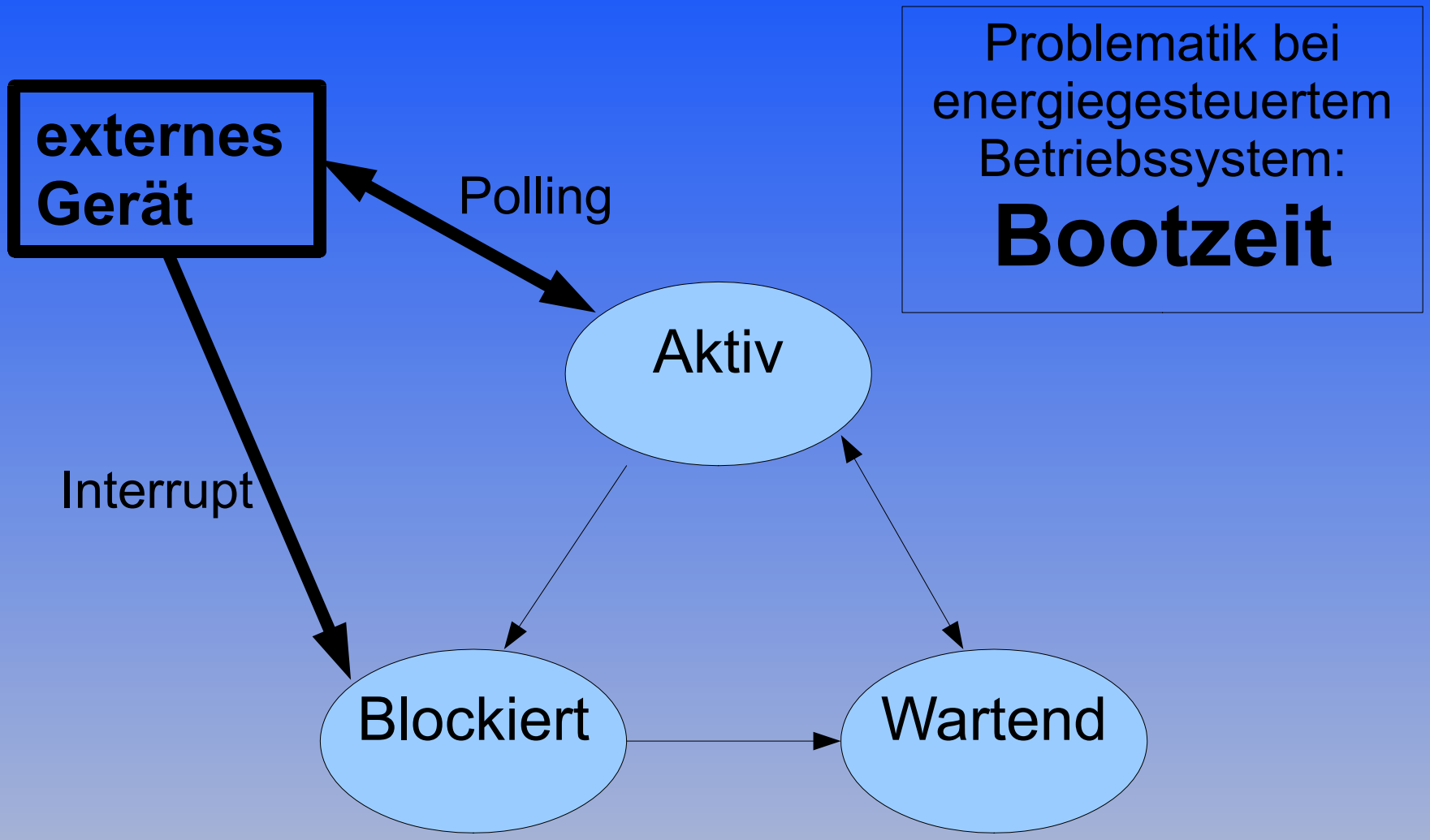
Grundzustand schlafend, aktiv nur, wenn etwas zu tun ist

Ereignisgesteuertes Betriebssystem

Grundzustand aus, aktiv nur, wenn genügend Energie vorhanden ist

Energiegesteuertes Betriebssystem

Unix- Prozessmodel



Problematik bei energiegesteuertem Betriebssystem:

Bootzeit

PowerTop

PowerTOP version 1.9 (C) 2007 Intel Corporation

Cn	Verweildauer	P-States (Frequenzen)	
C0	0,3%	2,00 GHz	0%
C1	0,0%	1,60 GHz	0%
C2	99,7%	1,20 GHz	100 %

Aufwachen pro Sekunde : 52,1 Intervall: 15,0s

Häufigste Ursachen für das Aufwachen:

```
 32,5% (20,0)      beagled : futex_wait (hrtimer_wakeup)
 16,3% (10,0)    <kernel core> : ehci_work (ehci_watchdog)
 11,1% (  6,8)    <interrupt> : ahci
  6,9% (  4,3)    <interrupt> : acpi
  6,5% (  4,0)    <kernel module> : usb_hcd_poll_rh_status (rh_timer_func)
  5,2% (  3,2)    <interrupt> : extra timer interrupt
  4,9% (  3,0)      kicker : schedule_timeout (process_timeout)
  2,2% (  1,3)      klipper : schedule_timeout (process_timeout)
```

Nach allen Tipps

Cn	Verweildauer	P-States (Frequenzen)	
C0	0,1%	2,00 GHz	0,0%
C1	0,0%	1,60 GHz	0,0%
C2	99,9%	1200 MHz	100,0%

Aufwachen pro Sekunde : 21,8 Intervall: 20,0s

Häufigste Ursachen für das Aufwachen:

```
 42,6% ( 10,0)  <kernel core> : ehci_work (ehci_watchdog)
 17,3% (  4,0)  <kernel module> : usb_hcd_poll_rh_status (rh_timer_func)
  8,7% (  2,0)    kicker : schedule_timeout (process_timeout)
  7,9% (  1,9)    kwin : schedule_timeout (process_timeout)
  5,3% (  1,2)    klipper : schedule_timeout (process_timeout)
  4,3% (  1,0)    kwrapper : do_nanosleep (hrtimer_wakeup)
  4,3% (  1,0)   <kernel core> : queue_delayed_work_on
  1,7% (  0,4)    kdesktop : schedule_timeout (process_timeout)
  1,3% (  0,3)    kded : schedule_timeout (process_timeout)
  1,1% (  0,2)    Xorg : do_setitimer (it_real_fn)
```

Software

Genaue Analyse der Peripherie, die ich wirklich benötige:
- Alles andere ausschalten!!!!

- Sleep-modes (Power mode), auch hier genau schauen, was ich benötige

- ACPI (Advanced Configuration and Power Interface): Version 3.0: Spezifikation 600 Seiten

Linux Kernel vor 2.6.21

Timer-Ticks (alle 1, 4 oder 10 ms)

Erhöhung der jiffies-Variable

Process Accounting: Wieviel Zeit hat jeder Prozess verbraucht ?

Scheduler: Muss ein Kontext-wechsel durchgeführt werden ?

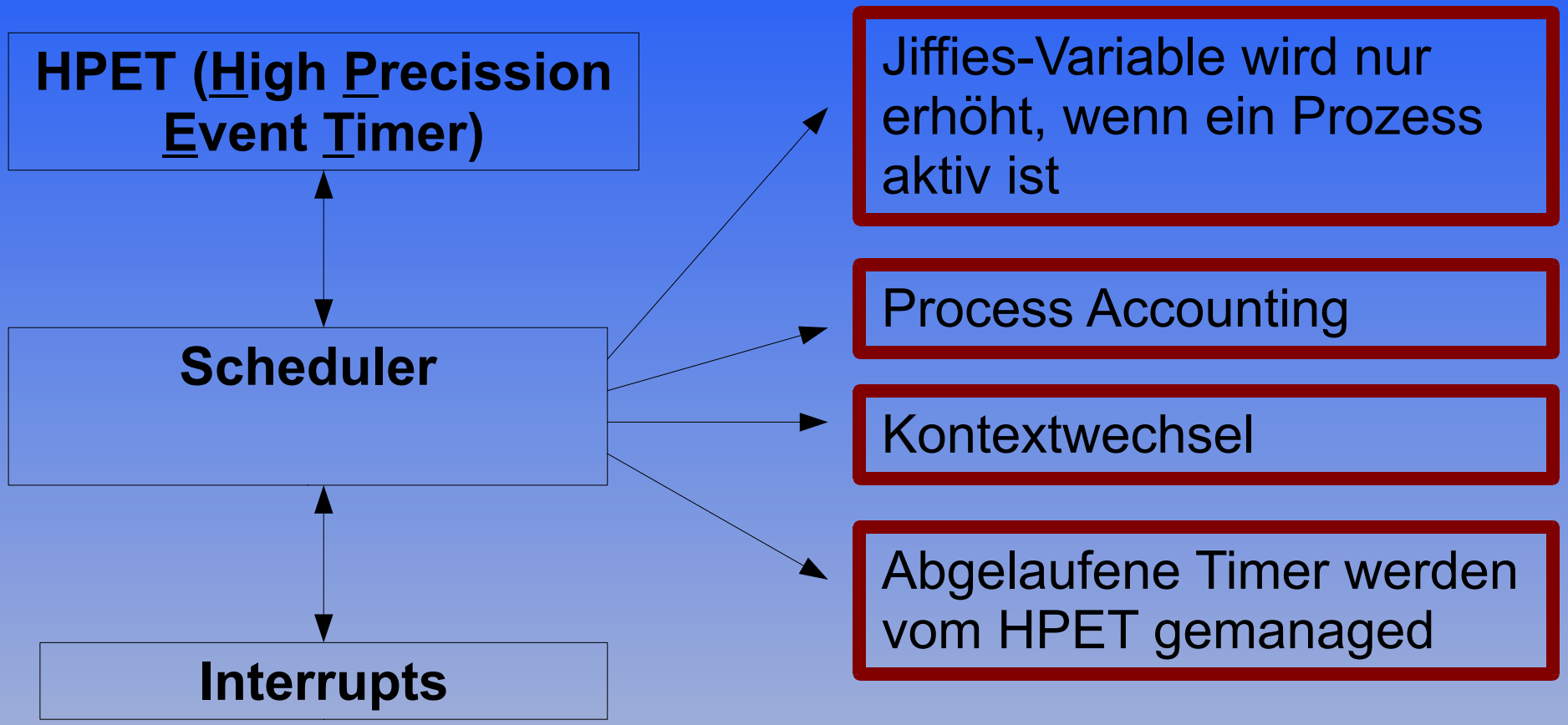
Sind SW-Timer abgelaufen ?

Nur benötigt, wenn ein Programm aktiv ist

Wird auch im Idle-Modus benötigt

Linux Kernel nach 2.6.21

Thomas Gleixner, Ingo Molnár et al.: Tickless Kernel



Wenn kein Prozess aktiv ist, kann der Prozessor im Idle-Mode bleiben.

Linux Kernel nach 2.6.21

Grouping:

Systemaufrufe:

unsigned long round_jiffies(unsigned long j);
unsigned long __round_jiffies_relative(unsigned long j, int cpu);

Neue **Applikationen** immer mit Power-Top oder ähnlichem testen !

Neue Plattform FRIARM

- Ereignisgesteuertes System (Interrupt-Handling)

- Dynamische Frequenz- und Corespannungs-Anpassung

- Ram/Flash wird durch NV-Ram ersetzt: Ausschalten des Stromes in Ruhephasen, Live-Daten im Ram, um schnelles Booten zu ermöglichen

- Display mit bistabilem LCD

Zusammenfassung

Energieeinsparungen können erreicht werden durch:

- Eine genaue Analyse der benötigten Funktionalitäten
- neue Technologien
- Entwicklungen an der Software hin zu ereignis-gesteuerten Systemen
- Verwendung von Sleep-modes
- schnelles Booten
- Linux bietet sehr gute Möglichkeiten, interessante Beiträge dahingehend zu leisten

Vielen Dank für die Aufmerksamkeit !

Weitere Infos:

wolfram.luithardt@hefr.ch
sylvain.decastel@hefr.ch