Energy-Aware Computing Systems

Energiebewusste Rechensysteme

IV. Energy Management

Timo Hönig

May 28, 2020

Agenda

Preface

Terminology

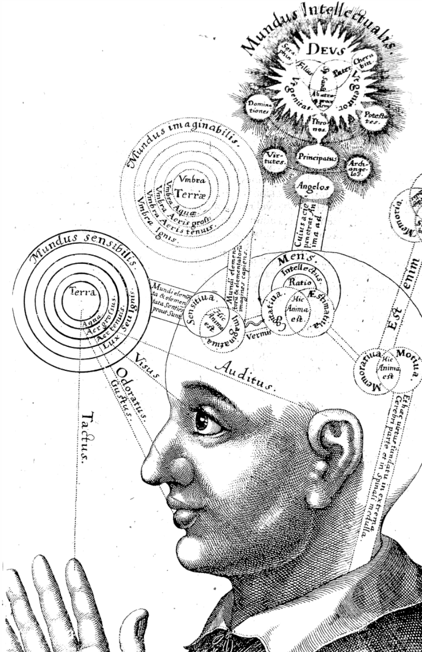
Resource Management and Control
- Resource Management
- Control Theory and Practice

Energy Management
- Control Methods and Characteristics
- Non-Blocking Methods
- Blocking Methods

Summary

Preface: Awareness

- awareness is the first step towards exercising control
  - sensing (passive perception) and managing (active control)
  - threeness $\rightarrow$ *towards*, *inside* and *away from* the system
- micro- vs. macrocosm
  - measure to analyze the whole (i.e., determine actual state)
  - reflect and control (i.e., enforce necessary system properties)



Robert Fludd (Astrologer, Philosopher, 1574 – 1637)

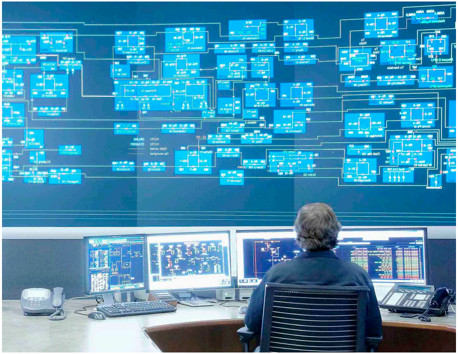
Abstract Concept: Energy Management

- energy **management**
  - *manage* originates from (it.) maneggiare: to handle, especially tools
  - derives from the two Latin words:
    - *manus* (hand)
    - *agere* (to act)

freeimages.co.uk

Abstract Concept: Energy Management

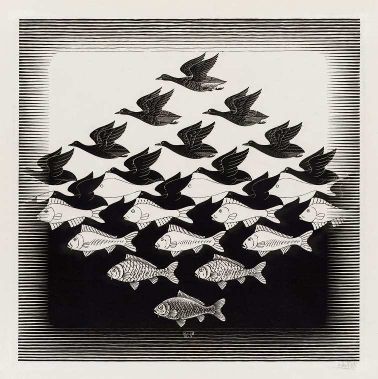
- **energy management**
  - limited operating resource
  - maximum rate to be spent
  - motivation
    - technical (i.e., quality of service, battery life)
    - economic (i.e., reduction of cooling costs)



news.cision.com/abb/i/abb-control-room,c2097912

Separation of Concerns and Powers, Duality

- managing energy as an operating (system) resource
- software
  - ...controls itself and the hardware
  - ...tracks state, influences control mechanisms (i.e., energy management)
- cooperation of soft- and hardware
  - software enforces control decisions that are executed by the hardware
  - hardware is responsible for state reporting (i.e., thermal conditions); reacts self-initiated in critical situations



- blurred lines
  - duality of responsibilities
  - temporal overrule situations

Resource Management

- managing energy as an operating (system) resource

Finite

- systems with **finite** energy resources
- global **operating time** depends on amount of available resources
- actively manage energy demand to increase power-on time

Revolving

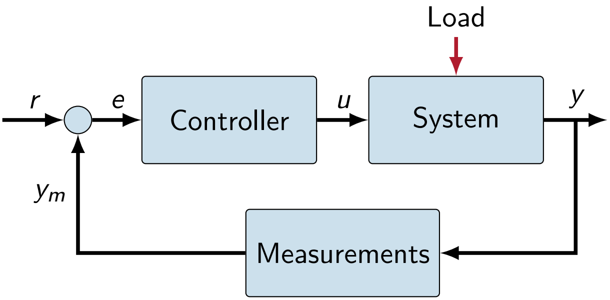
- systems with **revolving** energy resources
- **adverse effects** of unmanaged energy demand
- actively manage energy demand to adhere operating constraints

- systems switch between the two categories

↪ dynamic control of energy demand

Control Theory and Practice

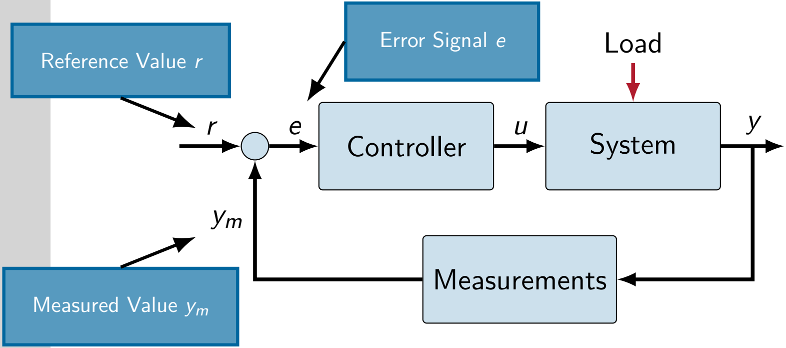
- measurement-based analysis with a feedback control system



- controller operates system: **closed control loop** ⇒ feedback control
- **control:** control variable u
  measure: process variable y

Control Theory and Practice

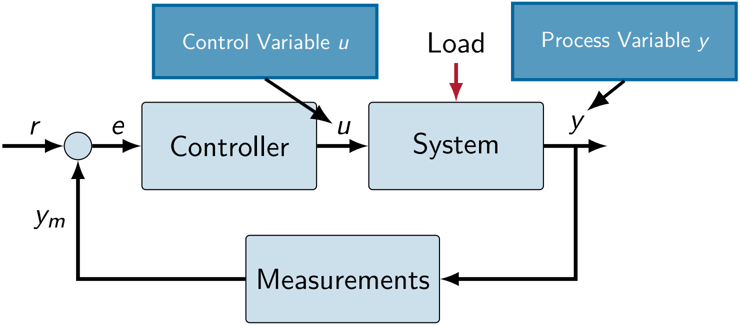
- measurement-based analysis with a feedback control system



- **controller input:** error signal $e = \Delta(r, y_m)$
- determine and enforce control variable $u \rightarrow$ purposed system behavior and corresponding response

Control Theory and Practice

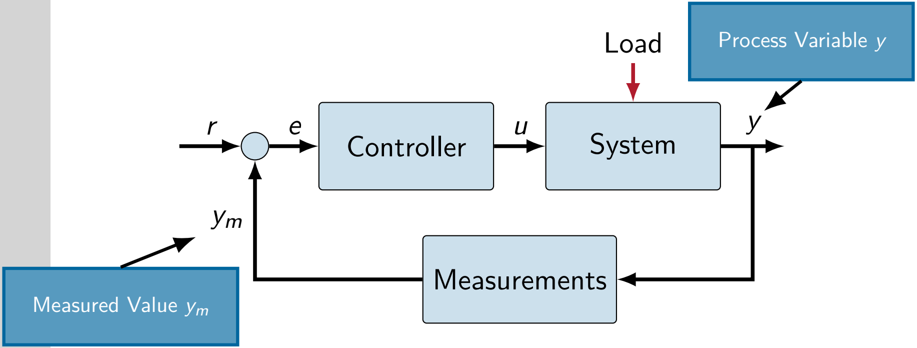
- measurement-based analysis with a feedback control system



- **controller output:** control variable $u \Rightarrow$ process variable y
- **process variable** y depends on
  system configuration and dynamic **system state** ($\rightarrow$ load)

Control Theory and Practice

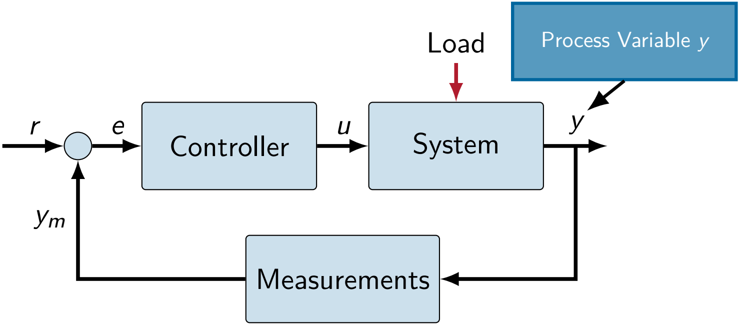
- measurement-based analysis with a feedback control system



- system response is measured and used as **feedback**
- **next control** action (u) depends on **currently measured system property** (y_m) $\Rightarrow$ time dependence

Control Theory and Practice

- measurement-based analysis with a feedback control system



- Example: controlling voltage and/or frequency
  - *u*: supply voltage, frequency
  - *y*: power demand, heat dissipation

Control Methods and Characteristics

- energy management at system level
- *what* system properties to control?
  - analyze cause and effect (cf. Lecture 3)
  - identify relevant system loads (software level) and levers (hardware level)
  - processes to supervise → energy saving features to control
- *when* to enforce the control?
  - proactive or reactive approach
  - explicit or implicit influence
  - temporal aspects ⇔ localization aspects
- interdependencies and side effects
  - recognize and quantify penalties (e.g., throughput, latency, performance)
  - counter measures to mitigate side effects (i.e., prepone operations ahead of sleep → latency hiding)
  - consider **restructuring** instead of enforcing management techniques

Control Methods and Characteristics

Non-Blocking

- progress guarantee
- low latency in order to be effective
- explicit vs. implicit

Blocking

- prone to starvation
- high latency in order to be reversed
- local vs. global

- positioning within system $\Leftarrow$ availability of necessary input
  - requires specifications to control separation of concerns and powers
  - software/hardware-only, **interlocked** software/hardware approaches
- energy management features with varying characteristics
  - effective on their individual purpose
  - **but**: combination of heterogeneous measures (i.e., non-blocking and blocking methods) to improve impact

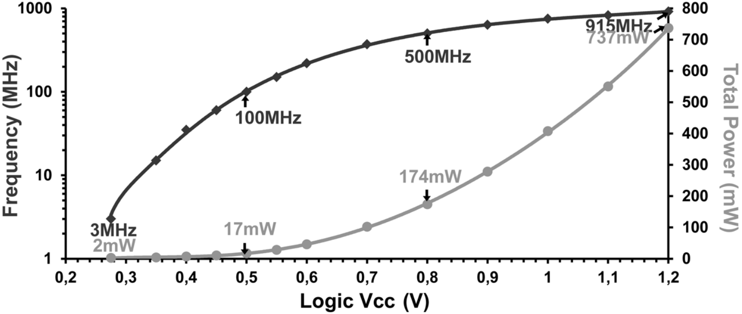
Non-Blocking Methods

Non-Blocking

- progress guarantee
- low latency in order to be effective
- explicit vs. implicit

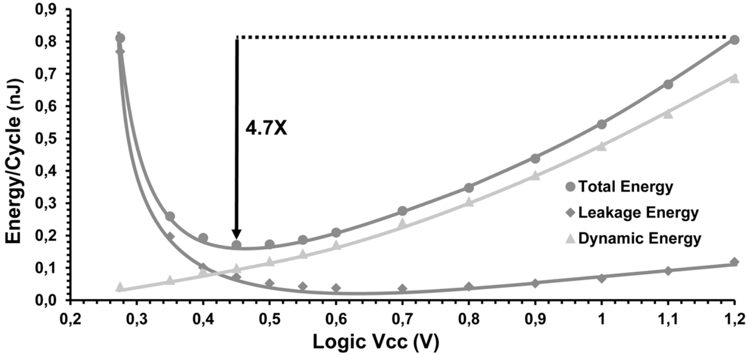
- non-blocking methods do not stall system progress
  but: (may) influence the quality of the progress
- non-blocking methods can be **explicit** or **implicit**
  - **explicit**: reduce energy demand with direct changes of electric circuitry (with likeliness to impact other system properties as backlash)
  - **implicit**: impact on energy demand by changing the demand of another resource (i.e., memory) or changing other system properties

Dynamic Voltage and Frequency Scaling



- Shailendra Jain, Surhud Khare, Satish Yada et al.
  A 280mV-to-1.2V Wide-Operating-Range IA-32 Processor in 32 nm CMOS
  IEEE International Solid-State Circuits Conference (ISSCC), 2012.

Dynamic Voltage and Frequency Scaling



- Shailendra Jain, Surhud Khare, Satish Yada et al.
  A 280mV-to-1.2V Wide-Operating-Range IA-32 Processor in 32 nm CMOS
  IEEE International Solid-State Circuits Conference (ISSCC), 2012.

Dynamic Voltage and Frequency Scaling

- dynamic voltage and frequency scaling (DVFS)

$$P_{dynamic} \propto C_{load} \cdot f_p \cdot A \cdot {V_{dd}}^2$$

$$\qquad\qquad\qquad\qquad \uparrow \qquad \uparrow$$

- power-performance trade-off: control f_p **and** supply voltage V_{dd}
- dynamic power depends on **frequency**, **supply voltage**

  ...and leakage depends on V_{dd}, too
- performance: **linear impact** $\Rightarrow$ advocate use of multiple cores

- **Interlude:** Scheduling for Reduced Energy
  - Mark Weiser et al.
    Scheduling for Reduced CPU Energy
    Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation (OSDI'94), 1994.

Dynamic Voltage and Frequency Scaling

- **system model**: dynamic voltage and frequency scaling
  ↪ but: idle CPU does not clock-gate or enter sleep state
  ↪ idle time represents wasting of energy (→ static energy demand)
- goal: lengthen execution time to **minimize idle time**
- proposed scheduling algorithms:

`OPT`	unbounded-delay perfect-future
`FUTURE`	bounded-delay limited-future
`PAST`	bounded-delay limited-past

- **Interlude:** Scheduling for Reduced Energy
  - Mark Weiser et al.
    Scheduling for Reduced CPU Energy
    Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation (OSDI'94), 1994.

Interlude: Scheduling for Reduced Energy

- common assumptions
  - scheduling with fixed-length intervals, **theoretical** approaches
  - adjust CPU clock for next interval at time of scheduling decisions
- `OPT` algorithm (unbounded-delay perfect-future)
  - simplified Oracle algorithm which entirely eliminates idle time
  - undesirable characteristics, e.g., stretching of I/O wait times
  - impractical: needs perfect knowledge on future
- `FUTURE` algorithm (bounded-delay limited-future)
  - like `FUTURE` but has only perfect knowledge for next time interval
  - impractical: (still) needs knowledge on future
- `PAST` algorithm (bounded-delay limited-past)
  - analyze past interval ⇒ predict future intervals
  - determine carryover of cycles from last interval ⇒ adapt CPU clock
  - practical

Interlude: Scheduling for Reduced Energy

```
idle_cycles = hard_idle + soft_idle;
run_cycles += excess_cycles;
run_percent = run_cycles / (idle_cycles + run_cycles);
IF excess_cycles > idle_cycles
THEN newspeed = 1.0;
ELSEIF run_percent > 0.7 THEN
   newspeed = speed + 0.2;
ELSEIF run_percent < 0.5 THEN
   newspeed = speed - (0.6 - run_percent);
IF newspeed > 1.0 THEN
   newspeed = 1.0;
IF newspeed < min_speed THEN
   newspeed = min_speed;
speed = newspeed;
```

- PAST algorithm (bounded-delay limited-past)
  - analyze past interval ⇒ predict future intervals
  - determine carryover of cycles from last interval ⇒ adapt CPU clock
  - practical

…at least back in the days

Dynamic Voltage and Frequency Scaling

- dynamic voltage and frequency scaling (DVFS)

$$P_{dynamic} \propto C_{load} \cdot f_p \cdot A \cdot {V_{dd}}^2$$

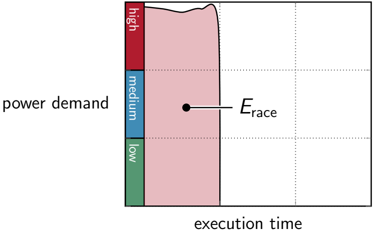
$$\qquad\qquad\qquad\qquad\qquad\qquad\uparrow \qquad \uparrow$$

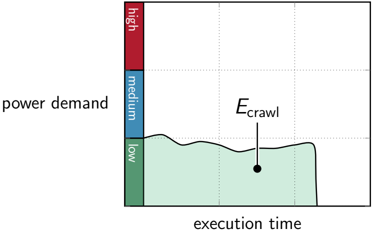
- power-performance trade-off: control f_p **and** supply voltage V_{dd}
- dynamic power depends on **frequency**, **supply voltage**

  ...and leakage depends on V_{dd}, too
- performance: **linear impact** $\Rightarrow$ advocate use of multiple cores

- **strategies**
  - multi-core CPUs: reduce clock frequency and execute in parallel
  - explore and exploit reduction of energy demand $\rightarrow$ execution modes

DVFS: Race-to-Sleep vs. Crawl-to-Sleep

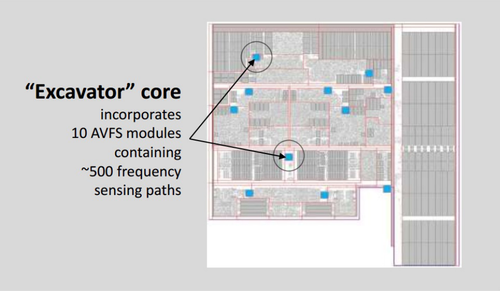




- race-to-sleep
  - motivation: maximize sleep time using a *blocking* management method after finishing pending work
  - rampant processes (i.e., compute-intensive operations)
- crawl-to-sleep
  - motivation: configure system at minimum voltage and clock rate, low average/peak power
  - thwarted processes (i.e., memory bus, I/O, network operations)

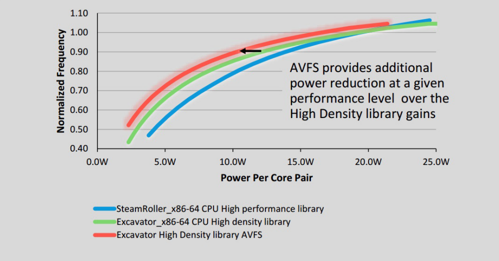
Adaptive Voltage and Frequency Scaling

- Adaptive Voltage and Frequency Scaling (AVFS)
  - motivation: consider device-specific variability in fabrication
  - exploit headroom of current DVFS designs **at hardware-level**
- AMD Excavator Family 15h [3], x86-64, fabrication: 28 nm
  - data of various frequency sensing paths determine strength of chip
  - transparent adjustment of V_{dd} and f_p at hardware-level
  - low-latency path to adapt to internal properties (i.e., thermal conditions)



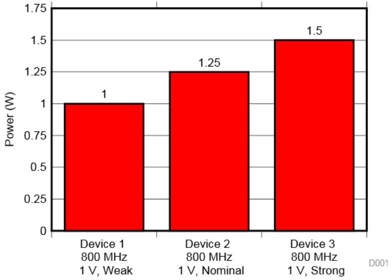
Adaptive Voltage and Frequency Scaling

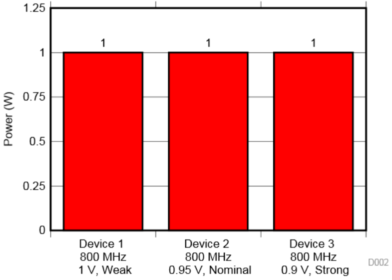
- Adaptive Voltage and Frequency Scaling (AVFS)
  - motivation: consider device-specific variability in fabrication
  - exploit headroom of current DVFS designs **at hardware-level**
- AMD Excavator Family 15h [3], x86-64, fabrication: 28 nm
  - data of various frequency sensing paths determine strength of chip
  - transparent adjustment of V_{dd} and f_p at hardware-level
  - low-latency path to adapt to internal properties (i.e., thermal conditions)



Adaptive Voltage and Frequency Scaling

- Adaptive Voltage and Frequency Scaling (AVFS)
  - motivation: consider device-specific variability in fabrication
  - exploit headroom of current DVFS designs **at hardware-level**
- AMD Excavator Family 15h [3], x86-64, fabrication: 28 nm
  - data of various frequency sensing paths determine strength of chip
  - transparent adjustment of V_{dd} and f_p at hardware-level
  - low-latency path to adapt to internal properties (i.e., thermal conditions)



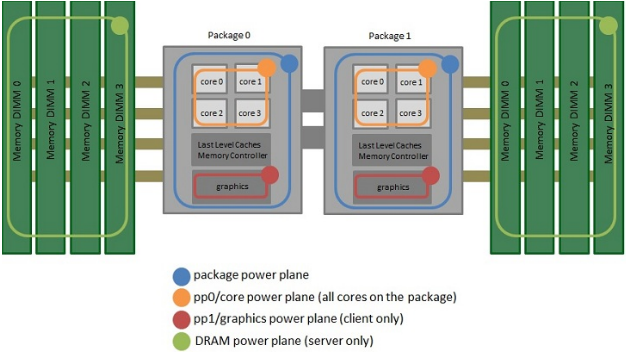


Running Average Power Limit

- Running Average Power Limit (RAPL)
- between the worlds: logical and physical measurements
  - originally, RAPL was using a software power model → **logical** measurements with hardware performance counters and I/O models
  - recent Intel CPUs (i.e., Haswell and onwards) → **physical** measurements
- hybrid approach towards energy-aware systems
  - adjusting performance levels (i.e., dynamic voltage and frequency scaling) ⇒ impacting power demand
  - **adjusting power levels (i.e., power capping)** ⇒ impacting performance

Running Average Power Limit

- power limiting (power capping)
  - setting power limits on individual domains
    → fine-grained control of the overall power demand
  - domains are, for example, package, memory (DRAM), CPU core, graphics



[1]

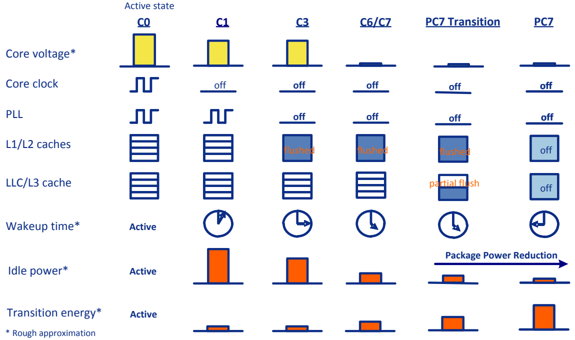
Blocking Methods

Non-Blocking	Blocking
■ progress guarantee ■ low latency in order to be effective ■ explicit vs. implicit	■ prone to starvation ■ high latency in order to be reversed ■ local vs. global

- blocking methods stall system progress due to inactivity (i.e., sleep)
- reduced energy demand for idle periods $\rightarrow$ demand for wakeup signal
- blocking methods are either **local** or **global**
  - **local**: components are dynamically put into low power states (i.e., device-specific sleep state)
  - **global**: system is put into a global low power state (i.e., system-wide sleep state), may need external interrupt to wake

CPU and Package Sleep States

- C-States (Cn) reduce power consumption of CPU cores when idle → local impact
  - State C0: core is active, code execution
  - State Cn with $n >= 1$: idle core is in sleep mode, no code execution
  - orthogonal to DVFS (→ P-States) and AVFS

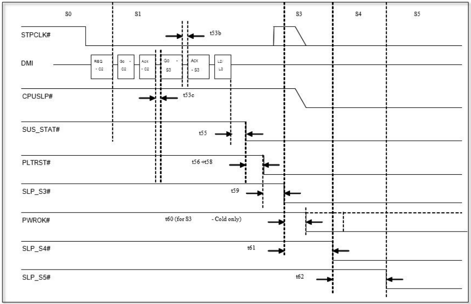


[2]

System Sleep States

- S-States (Sn) reduce power consumption of the overall system $\rightarrow$ global impact
  - State S0: system is awake and operates
  - State Sn with $n >= 1$: system is in global sleep state

Sleep State Entry Sequence



[2]

Considerations and Caveats

- energy management at (operating) system level
  - manage energy as an operating resource
  - what system properties to control?
  - control proactively or reactively?
- non-blocking method
  - explicit or implicit control energy demand dynamically at runtime
  - orthogonal to non-blocking methods ↓
- blocking methods
  - local or global suspension of operation (i.e., enter sleep mode)
  - orthogonal to blocking methods ↑

Subject Matter

- consider **energy** as an **operating resource** that must be managed, **enforcement** of system policies (i.e., power demand vs. performance)
- requires smooth interaction between hardware and software (i.e., sleep state transitions)
- orthogonal non-blocking and blocking methods

- reading list for Lecture 5:
  - Vishal Gupta et al.
    The Forgotten "Uncore":
    On the Energy-Efficiency of Heterogeneous Cores
    Proceedings of the USENIX Annual Technical Conference (ATC), 2012.

Reference List I

[1] Dimitrov, M. :
Intel Power Governor.
https://software.intel.com/en-us/articles/intel-power-governor,

[2] Intel:
Energy-Efficient Platforms – Considerations for Application Software and Services.
https://www.intel.com/content/dam/doc/white-paper/energy-efficient-platforms-2011-white-paper.pdf,

[3] Munger, B. ; Akeson, D. ; Arekapudi, S. ; Burd, T. ; Fair, H. R. ; Farrell, J. ; Johnson, D. ; Krishnan, G. ; McIntyre, H. ; McLellan, E. ; Naffziger, S. ; Schreiber, R. ; Sundaram, S. ; White, J. ; Wilcox, K. :
Carrizo: A High Performance, Energy Efficient 28 nm APU.
In: *IEEE Journal of Solid-State Circuits* 51 (2016), Jan, Nr. 1, S. 105–116