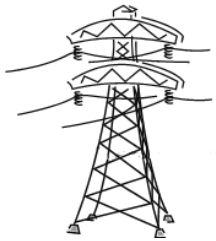
Cyber-physical systems: opportunities, problems and (some) solutions

Peter Marwedel*

TU Dortmund (Germany), Informatik 12

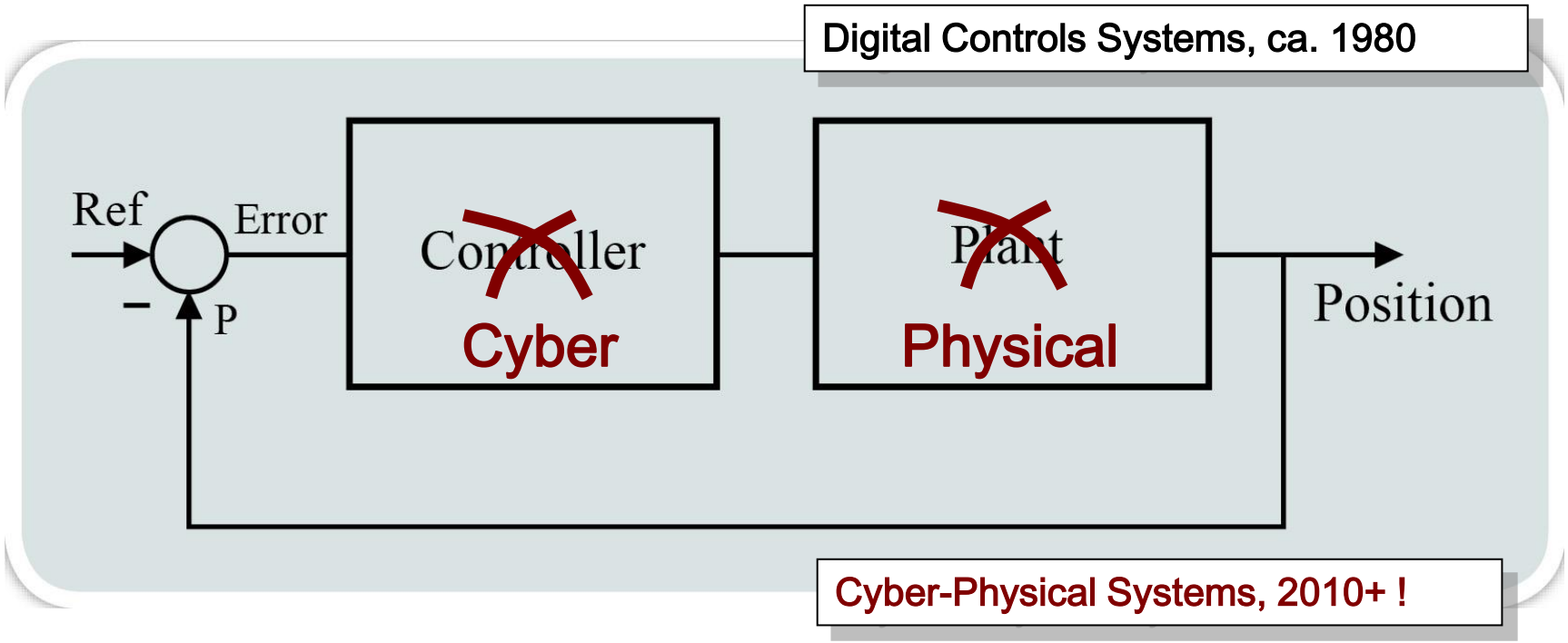




2015年11月09日

* + contributions by PhD students

Photos/Graphics: P. Marwedel + Microsoft

What is a Cyber-Physical System?

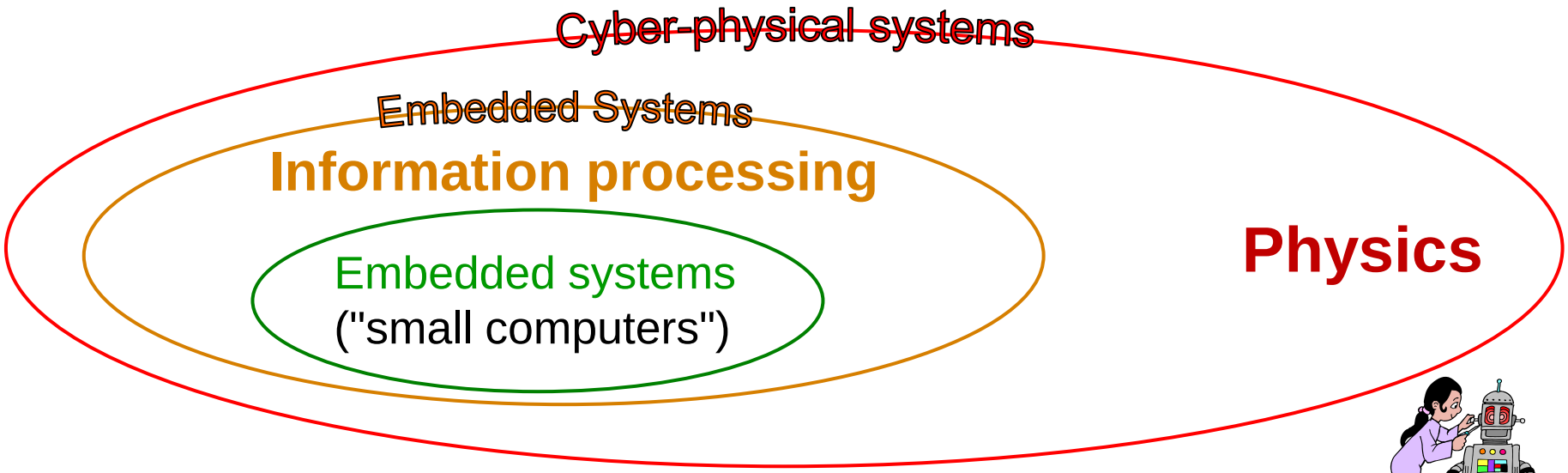


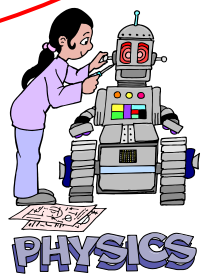
Emperor's New Cloths?

Just wearing the inside out?

Cyber-physical systems and embedded systems

- Embedded systems (ES): information processing embedded into a larger product [Peter Marwedel, 2003]
- Cyber-physical systems (CPS): integrations of computation with physical processes [Edward Lee, 2006].





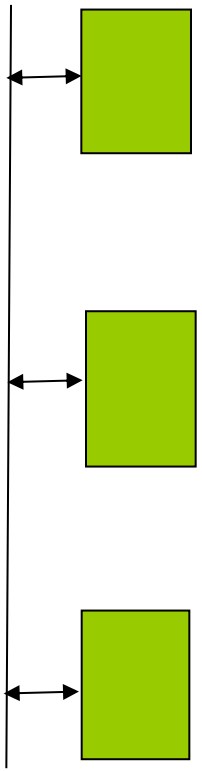
CPS = ES + physical environment

Definitions

Cyber-physical systems (CPS) are engineered systems that are built from and depend upon the ***synergy*** *of* ***computational*** *and* ***physical components.***

Emerging CPS will be ***coordinated, distributed****, and* ***connected****, and must be* ***robust*** *and* ***responsive****.* [National Science Foundation (US)]

cps-vo.org



… represent ***networked****,* ***software-intensive embedded systems in a control loop,*** *provide* ***networked*** *and* ***distributed*** *services* [akatech]

Akatech: Cyber-Physical Systems. Driving force for innovation in mobility, health, energy and production, http://www.acatech.de/de/publikationen/stellungnahmen/kooperationen/detail/artikel/cyber-physical-systems-innovationsmotor-fuer-mobilitaet-gesundheit-energie-und-produktion.html

Description according to H-2020-ICT-2014-1

Cyber-Physical Systems (CPS) refer to ***next generation embedded ICT systems*** *that are* ***interconnected*** *and* ***collaborating*** *including through the Internet of things, and providing citizens and businesses with a wide range of innovative applications and services.*

These are the ICT systems increasingly embedded in all types of artefacts making ***"smarter", more intelligent, more energy-efficient*** *and* ***more comfortable*** *our transport systems, cars, factories, hospitals, offices, homes, cities and personal devices.*

Often endowed with ***control, monitoring and data gathering functions****, CPS need to comply with essential requirements like* ***safety, privacy, security*** *and* ***near-zero power consumption*** *as well as* ***size, usability*** *and* ***adaptability constraints****.*

http://bit.ly/13I7985

http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/78-ict-01-2014.html

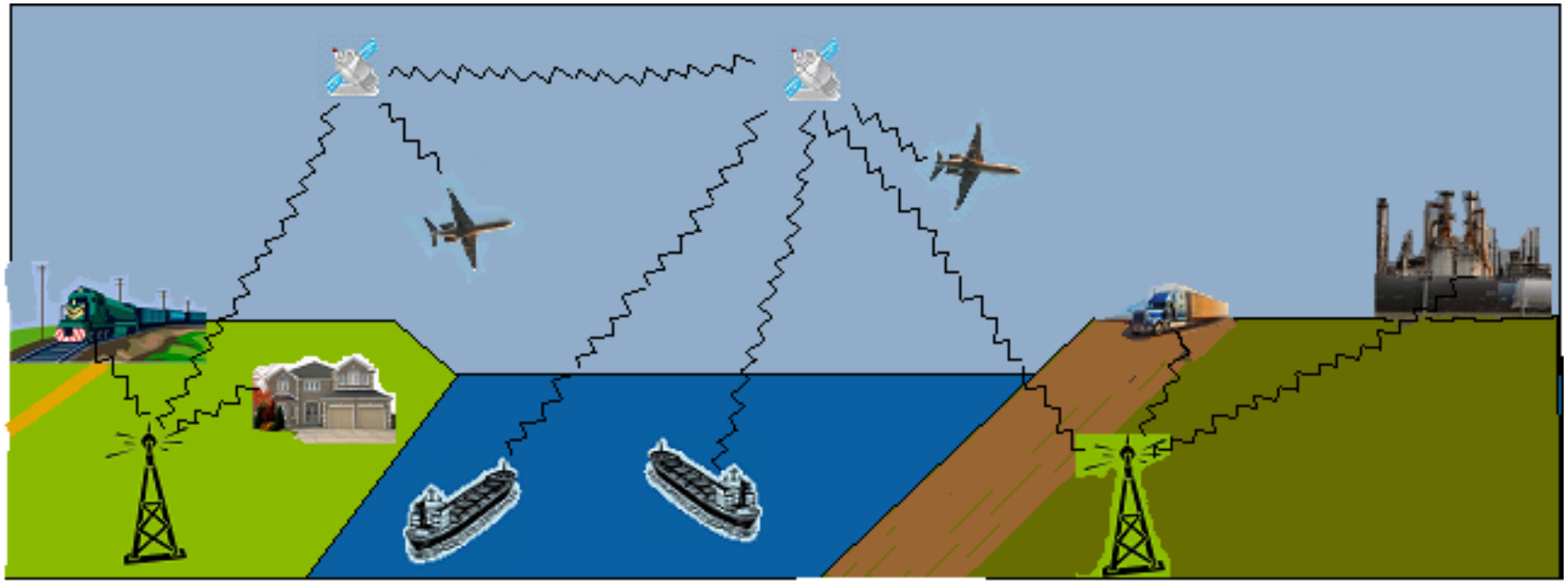
Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Increased connectivity of systems



Transportation

- Automotive

- Avionics

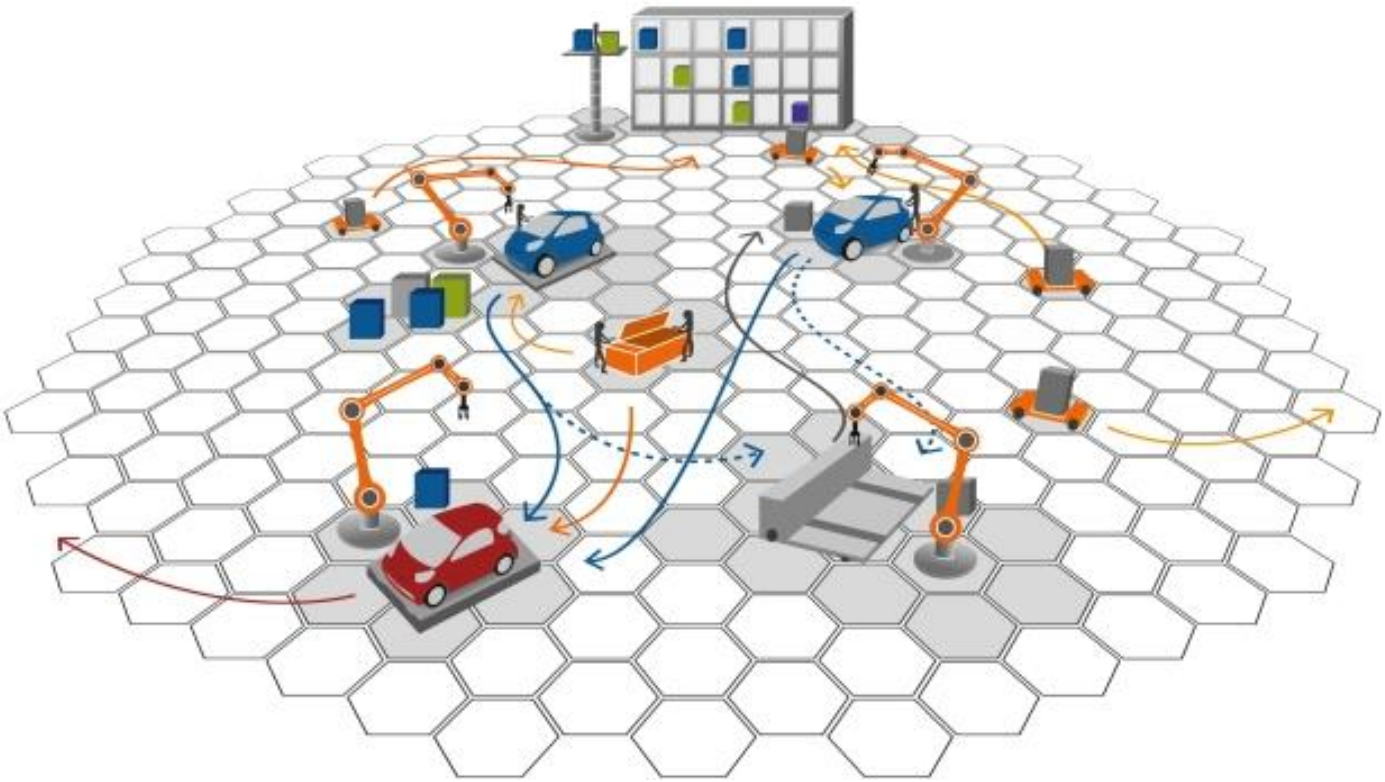
- Rail

- Ships

Comprise many embedded systems, increasingly connected

More flexible production planning (“Industrie 4.0”)

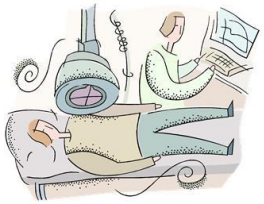
E.g. flexible islands in automotive production



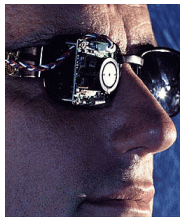
http://www.autonomik40.de/en/SMART%20FACE.php

Health

- Diagnosis
  - Sensors
  - Patient information
  - Data analysis
  - Risk analysis
- Support of therapies
  - Personalized medication
  - Medical devices
  - Support for handicapped patients
- Result monitoring



☞ *Insup Lee*



© www.dobelle.com, ~2000

One of the areas covered in the acatech report

Smart Home

- Generates as much energy as it is consuming
- Provides maximum comfort
- Offers safety without constraining users
- Supports users, ambient assisted living

More Opportunities

- Smart Grid
- Disaster recovery
- Public safety

© umweltbundesamt.de

© www.spiegel.de

Social spaces ☞ Tarek F. Abdelzaher

- Structural health monitoring
- Scientific experiments

© CERN

© P. Marwedel

Microsystems ☞ Jan Madsen

More Opportunities (2)

- Robotics

- Military systems

© www.spiegel.de

- Telecommunication

- Consumer electronics

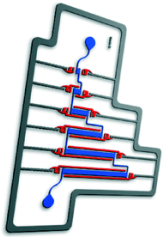
☞Wide scope!

NSF-CPS Sample topics

Listed at cps-vo.org:

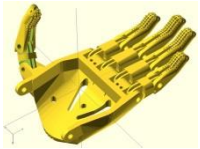
- Networked Sensor Swarm of Underwater Drifters

- Towards a Programmable and Reconfigurable Lab-on-Chip



Micro-systems ☞ Jan Madsen

- Context-dependent control of smart artificial hands

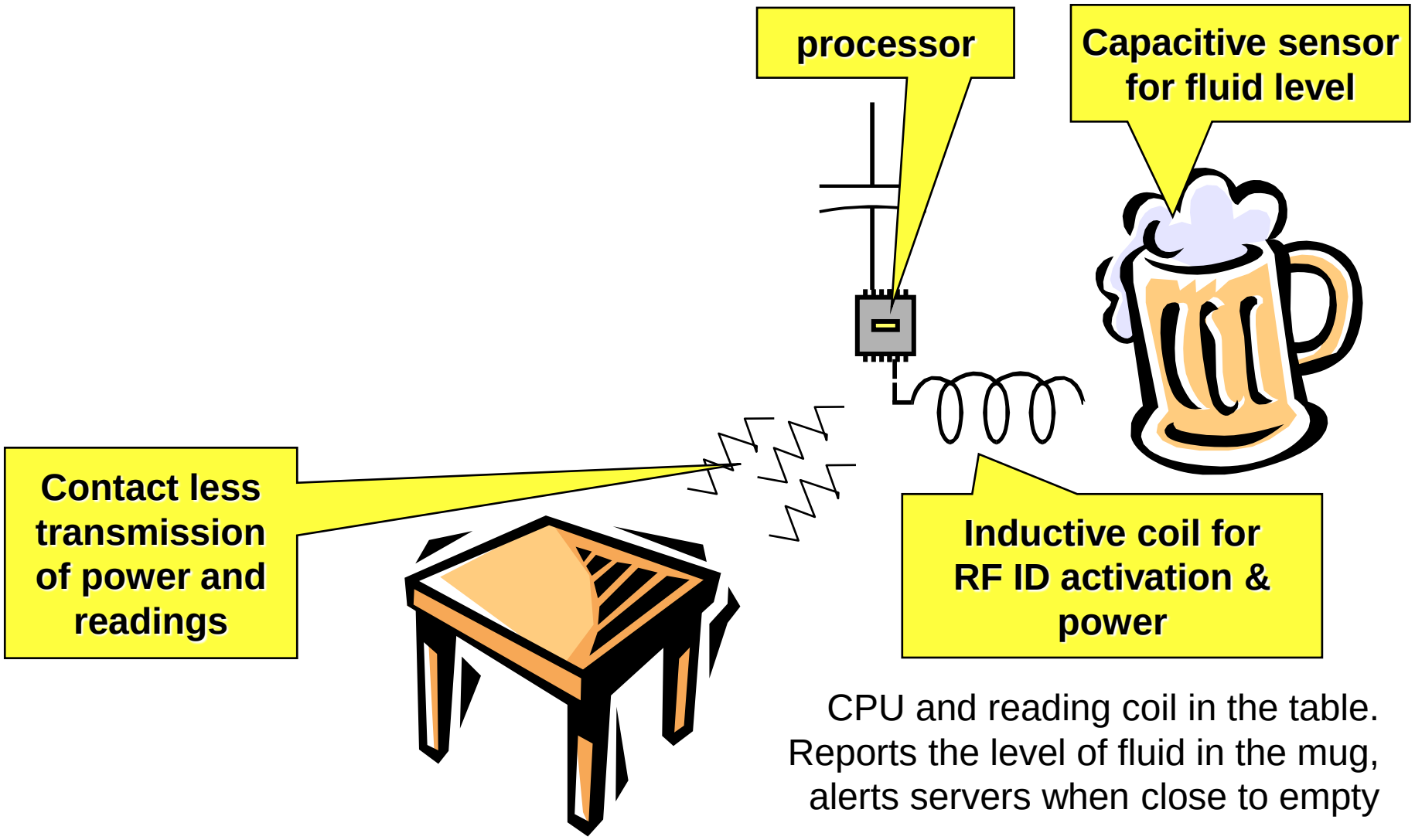


- Cloud Environmental Analysis and Relief

- Timing-Centric Software

☞ Wide range of opportunities**!**

Smart Beer Mug



CPU and reading coil in the table.
Reports the level of fluid in the mug,
alerts servers when close to empty

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Security & Safety

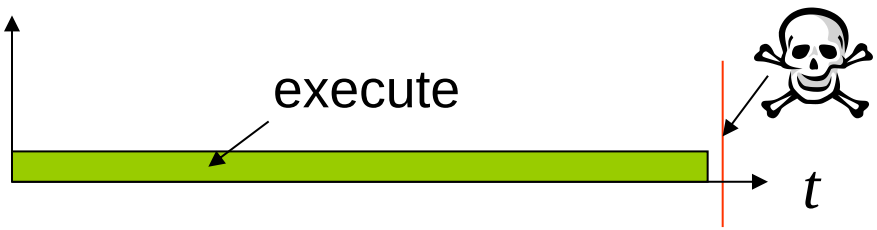
- Increased connectedness increases risk of attacks
  - complete systems can be affected
- Defending against
  - Cyber crime
  - Cyber attacks (☞ Stuxnet)
  - Cyber terrorism
  - Cyber war

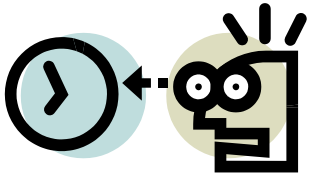
Social spaces ☞ Tarek F. Abdelzaher

- Local islands provide some protection,
  but are in contradiction to the idea of global connectedness

Timing predictability

The system must react within a time interval dictated by the environment [Adopted from Bergé]





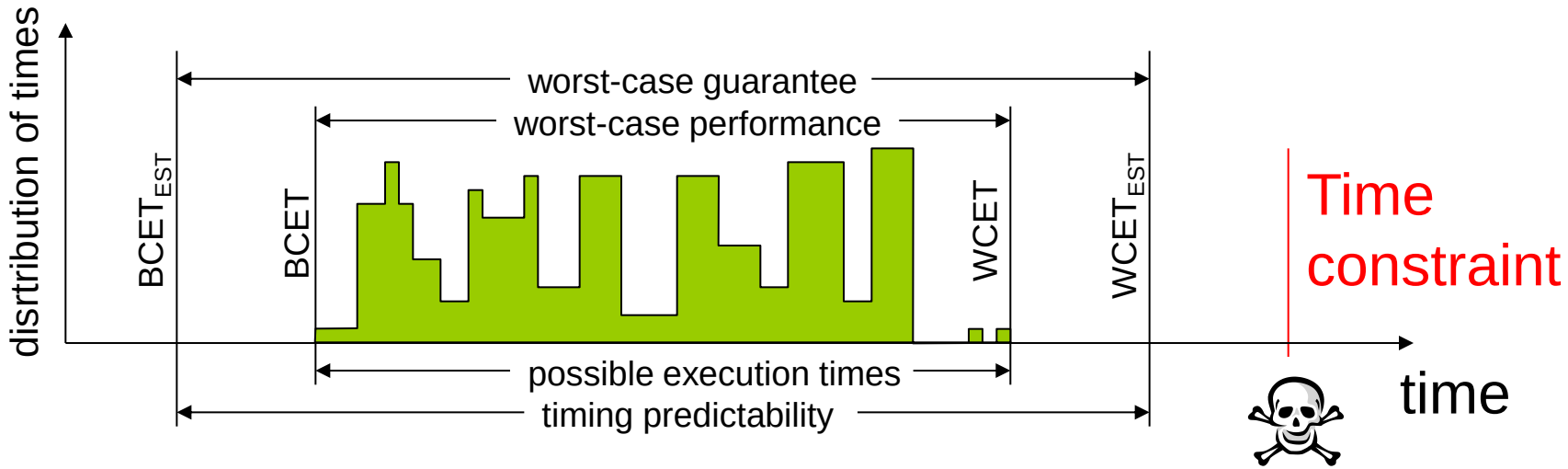
The lack of timing in the core abstraction (of computer science) *is a flaw* [Edward Lee]

Consequences underestimated

Example: Problems with new ICE trains: slow propagation of signals from one end to the other.

Worst case execution time

Definition of worst case execution time:



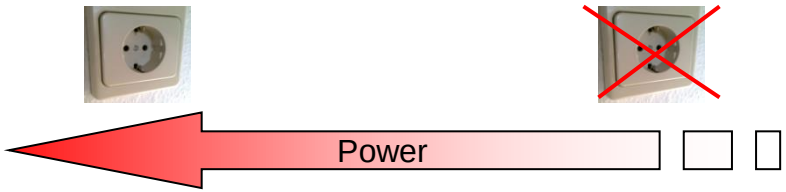
$WCET_{EST}$ must be

1. safe (i.e. ≥ WCET) and
2. tight ($WCET_{EST}$-WCET≪$WCET_{EST}$)

Compute **average** and **worst case** execution times!

Why care about energy efficiency ?

$$E = \int_0^{t_{max}} P_{el}(t)\, dt, \;\; t_{max} \leq \text{WCET}$$



	Relevant during use?		
Execution platform	Plugged	Uncharged periods	Unplug-ged
E.g.	Factory	Car	Sensor
Global warming	☑	☐	☐
Cost of energy	☑	☐	☐
Increasing performance	☑	☑	☑
Problems with cooling, avoiding hot spots	☑	☑	☑
Avoiding high currents & metal migration	☑	☑	☑
Reliability	☑	☑	☑
Energy a very scarce resource	☐	☑	☑

Portable systems must be energy efficient

Carrying energy or energy harvesting

Energy Harvesting with 400lx solar cell

Energy storage
Discharge < 2% p.a.
7.000 telegrams
without re-charging

- Many opportunities!
- Electrical energy converted into thermal energy

Fraunhofer IML

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Thermal requirements

$P_{el}(t) \rightarrow T$; Increasingly important

- since temperatures become more relevant due to increased performance,
- and since temperatures affect usability
- and dependability:

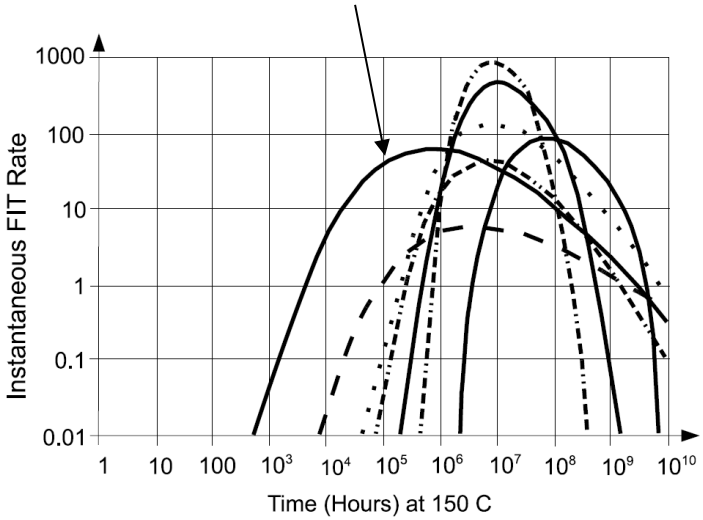
$$\mathrm{MTTF} = \frac{A}{j_e^{\,n}}\, e^{\frac{E_a}{kT}}$$

(see slide 68 for details)

Reliability/dependability/resilience

The system must be reliable

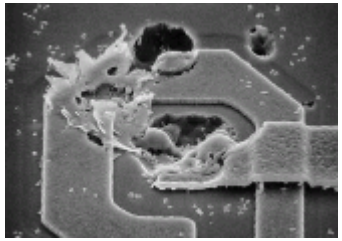
Different devices



Failure rates for GaAs at 150°C
[www.triquint.com/company/quality/faqs/faq_11.cfm]

- Consequences frequently underestimated
- Order of magnitude of probabilities: 10^{-9}/hr = 1 FIT

- Reduced reliability due to smaller patterns within semiconductor chips [ITRS, 2009]
- Transient & permanent faults



www.jrwhipple.com/computer_hangs.html

Terms

- "*A **service failure**, often abbreviated here to **failure**, is an event that occurs when the delivered service of a system deviates from the correct service.*"
- *"The definition of an **error** is the part of the total state of the system that may lead to its subsequent service failure"*
- "*The adjudged or hypothesized cause of an error is called a **fault**. Faults can be internal or external of a system.*"

Example:

- Transient **fault** flipping a bit in memory.
- After this bit flip, the memory cell will be in **error**.
- **Failure***:* if the system service is affected by this error.

[Laprie et al., 1992, 2004]

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

A new look at control loops

- Many CPS comprise control loops
  - Voltage control in dynamic voltage scaling
  - Overheating control
  - Bandwidth control
  - CPU resource management

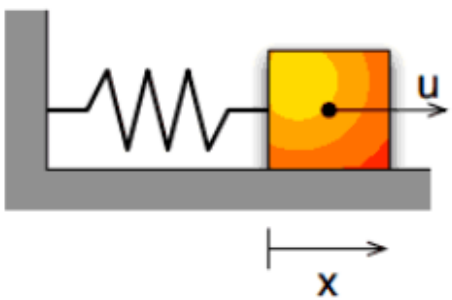
☞ *Karl-Erik Årzén*

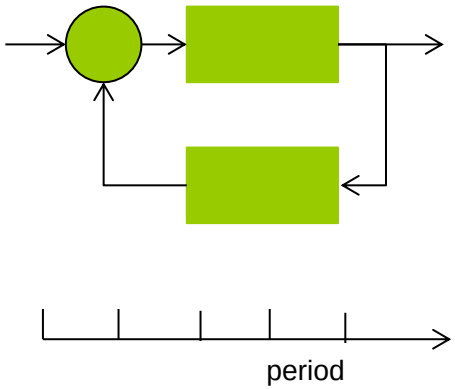
- Feedback ≠ from classical analog feedback:
  - Discrete instead of continuous feedback
  - Quantization effects
  - Larger delays (e.g. with complex algorithms)
  - Feedback broken at times (e.g. wireless)
- Don‘t forget stability issues!

Cocktails of time and control

The context

We start with a textbook example to provide a concrete context for our cocktails.





$$\begin{aligned} \dot{x} &= v \\ \dot{v} &= -\frac{k}{m}x + \frac{1}{m}u \end{aligned}$$

The objective is to design a feedback control law $u = Kx$ to stabilize the mass at $x = 0$.

For simplicity we take $m = 1$ and $k = 10$.

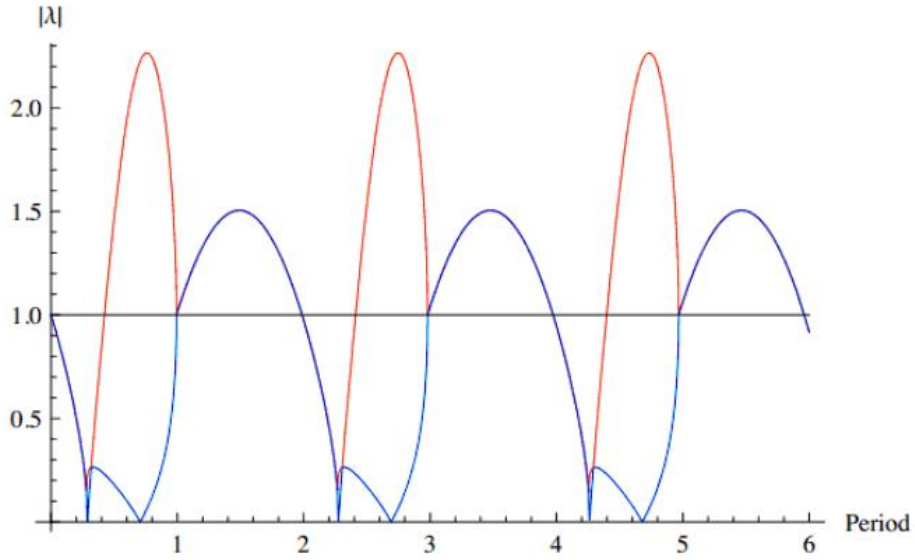
Paulo Tabuada: Is it about time for control?, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5997

Cocktails of time and control

The first cocktail

For a periodic implementation asymptotic stability is equivalent to the closed-loop eigenvalues being strictly inside the unit disk.

Let us plot the location of the eigenvalues as a function of the chosen period.



Stability is achieved with small periods as well as with (arbitrarily) large periods!

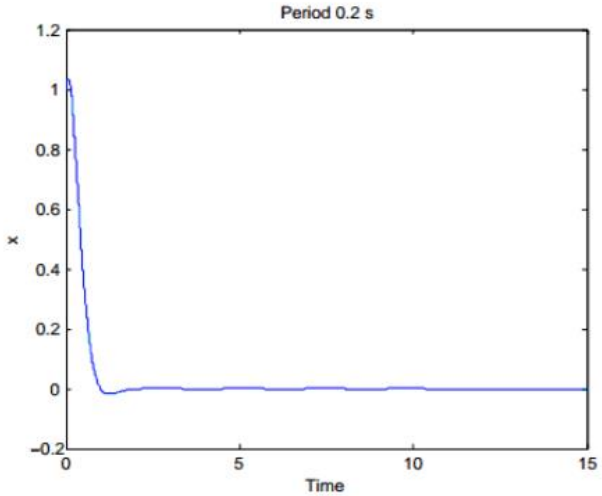
Paulo Tabuada: Is it about time for control?, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5997

Cocktails of time and control

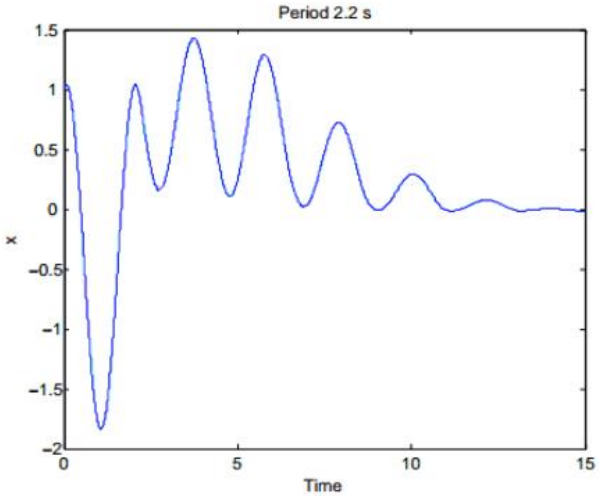
The second cocktail: adding the performance ingredient

Stability, however, is not the full story.

Period 0.2s



Period 2.2s



Is performance monotonically increasing with decreasing periods?

No! Shorter periods magnify the undesired effects of sensor noise and quantization.[2]

[2] B. Bamieh. *Intersample and finite wordlength effects in sampled-data problems.* IEEE Transactions on Automatic Control, 48(4), 639–643, 2003.

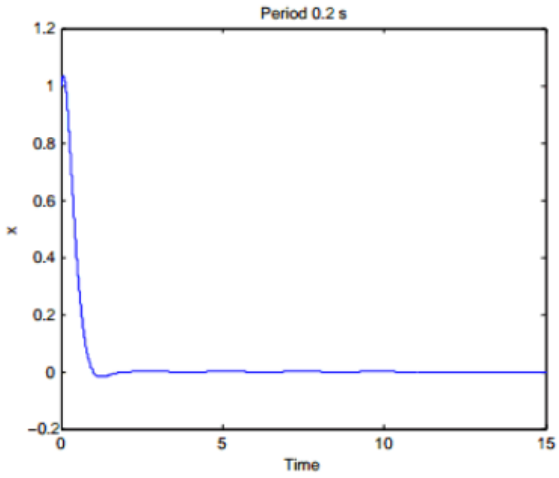
Paulo Tabuada: Is it about time for control?, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5997

Cocktails of time and control

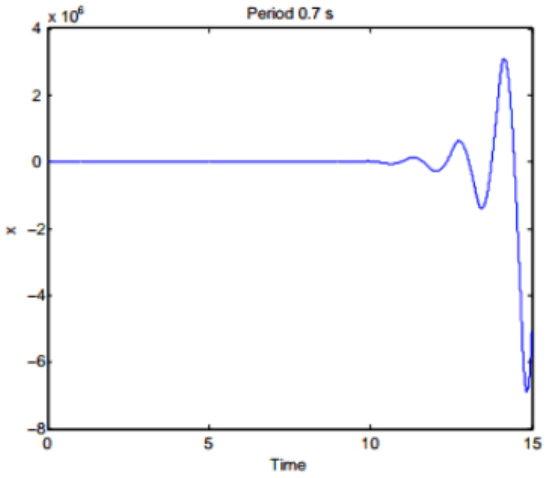
The third cocktail: mixing periods

Periods and performance, different cocktails for different tastes.

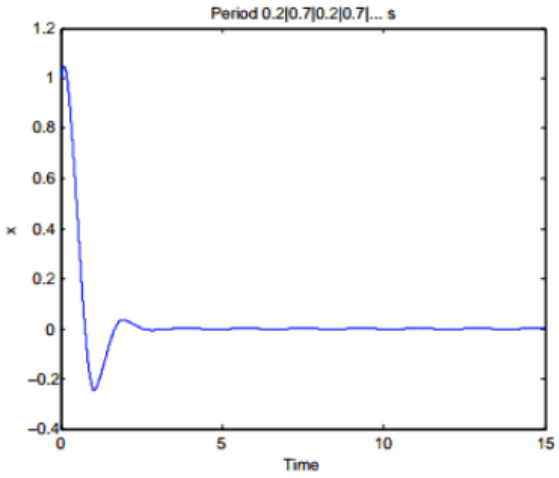
Period 0.2s



Period 0.7s



Period 0.2|0.7|0.2|0.7s

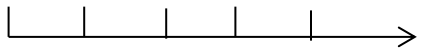


Good performance

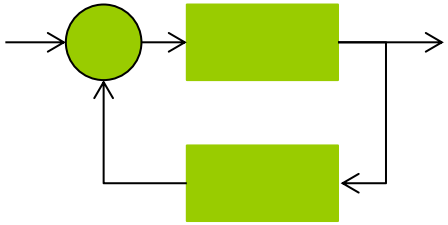
Paulo Tabuada: Is it about time for control?, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5997

A new look at control loops

- Periodic sampling worked well



- CPS: more flexible control: when to sample?
- A case for machine learning:
  - Longer sampling periods required by machine learning become acceptable
  - Time for the next sampling may itself be predicted with machine learning

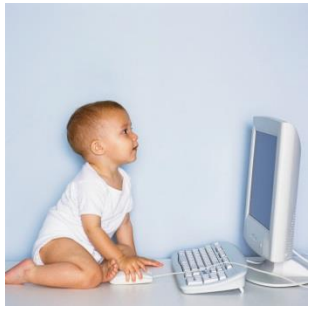


Paulo Tabuada: Is it about time for control?, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5997

Human/social & legal issues

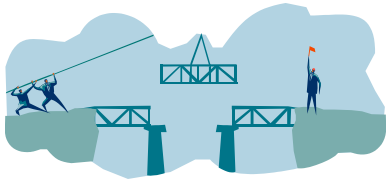
Human/social issues

- What is the impact of CPS on human society?
- How to ensure benefit of humans?
- How to ensure acceptance of technology?



Legal issues

- How to ensure necessary cooperation (e.g. in the smart grid)?
- How to solve intellectual property issues?
- Who owns the data?
- Who is liable?



Dynamism & heterogeneity

Environment is frequently changing

- Various network standards
- Networks are less reliable and not permanently available
- Data rates and costs vary
- Dynamic handover

Components are very heterogeneous

- Analog and digital
- Different standards
- …

Verification problem

- Real time model checking
- Statistical model checking
- ..
- "Fuzzy" Verification
  - Implementation will not correspond to specification ***exactly***

  ☞ Boolean decision "specification is met/not met" has to be replaced by a level of confidence l, $0 \leq l \leq 1$

  ☞ New verification techniques required

☞ *Kim G. Larsen*

☞ *Axel Legay*

George Pappas: Science of Cyber-Physical Systems Bridging CS and Control, 2012 NSF CPS PI meeting, http://cps-vo.org/node/5876

Real vs. floating point numbers

Physics:

- Many physical quantities described as (real number, unit)
- Infinitely many real numbers

Cyber:

- Floating point numbers as an approximation
- Finite number of floating point values
- Absolute resolution depends on size of value

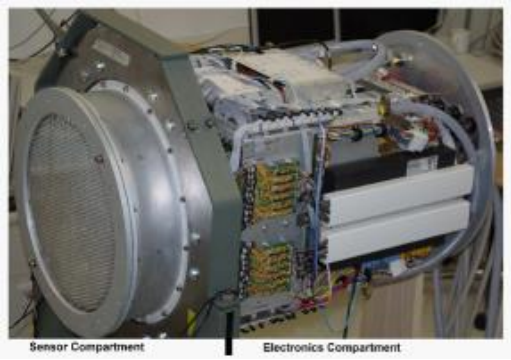
Mismatch!

W. Taha, R. Cartwright: Some Challenges for Model-Based Simulation, The 4th Analytic Virtual Integration of Cyber-Physical Systems Workshop, Dec. 2013, Vancouver

Big Data

Examples:

- LHC/Geneva: 25 Petabytes/year
- Sloan Digital Sky Survey: 200 GB/night
- Data from navigation systems
- Data from social networks



Getting to the limits of classical data bases

Zeno behavior

Example:

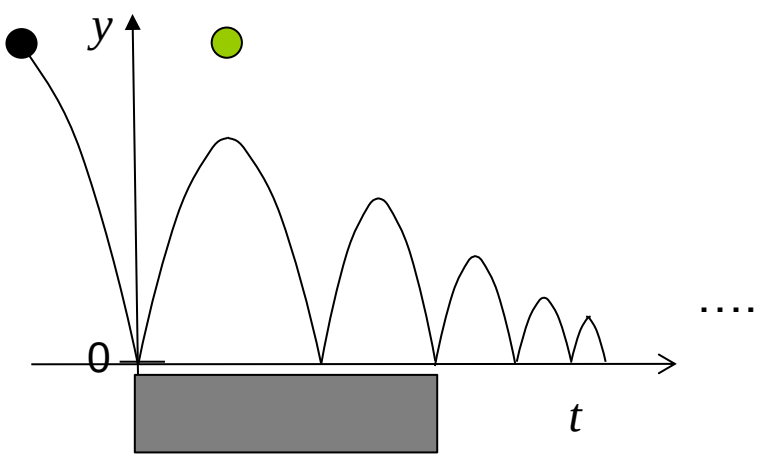
- Ideal collision & damping
- assuming point mass

Ideal model for one collision at t=0:

y>0, t<0: $\ddot{y}(t)=-g$

y=0: $\dot{y}(0)=-\gamma \max_{t<0}(\dot{y}(t))$

Assume repeated collisions



- Due to damping, each cycle is shorter than the previous one.
- Let Δ be a time interval
- Is there an upper bound on the number of bounces in Δ ?

  NO

- A system shows Zeno behavior if there can be an unlimited number of jumps in a finite time.
- Requires hybrid simulation

More objectives/constraints

In addition to challenges already mentioned:

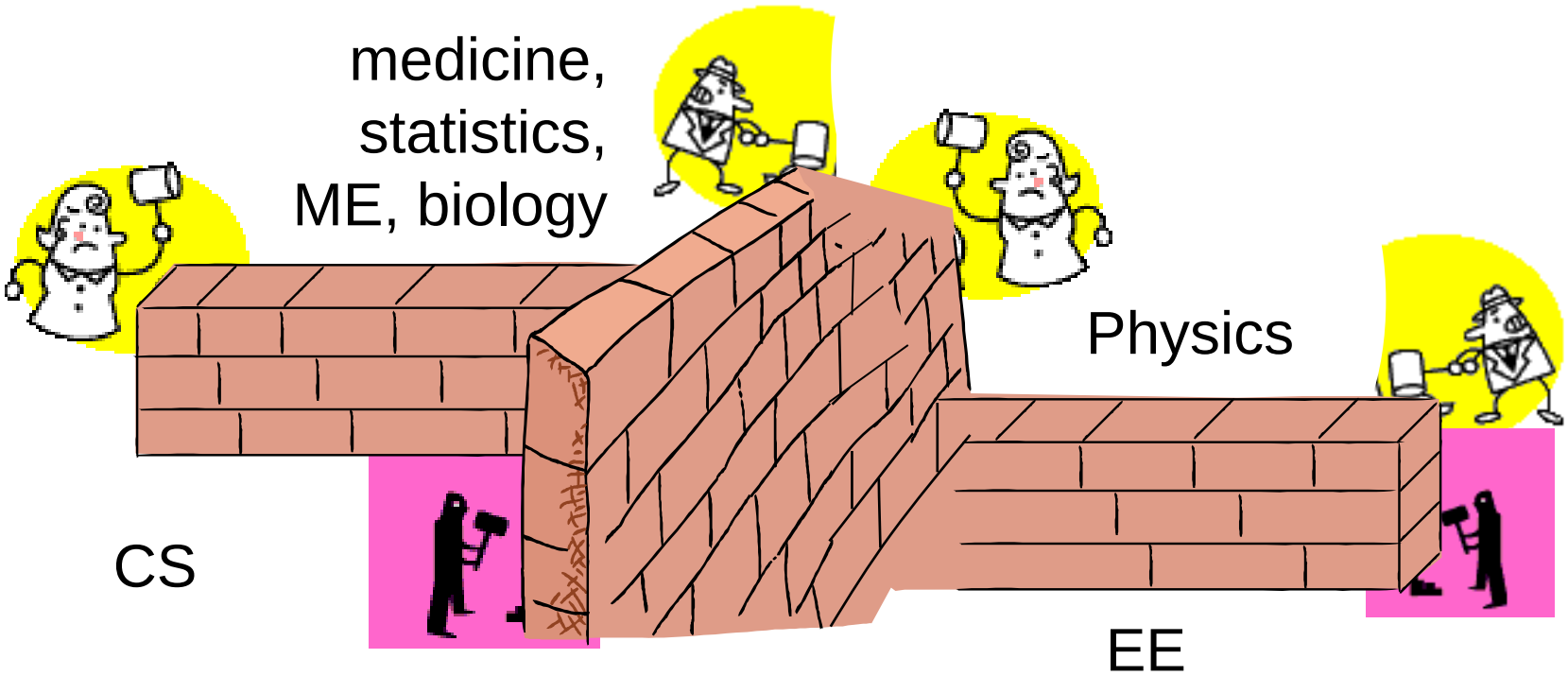
- quality of the results (QoS, ..)
- cost, size
- weight
- EMC characteristics
- radiation hardness
- environmental friendliness, ..





CPS design is multidisciplinary

Knowledge from many areas must be available,
Walls between disciplines must be torn down



Problems with classical CS theory and von Neumann (thread) computing

Even the core … notion of "computable" is at odds with the requirements of embedded software.

In this notion, useful computation terminates, but termination is undecidable.

In embedded software, termination is failure, and yet to get predictable timing, subcomputations must decidably terminate.

What is needed is nearly a reinvention of computer science.

Edward A. Lee: Absolutely Positively on Time, *IEEE Computer*, July, 2005

☞ Search for non-thread-based, non-von-Neumann MoCs.

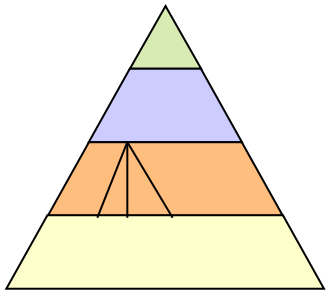
Models

Definition: *A model is a simplification of another entity, which can be a physical thing or another model. The model contains exactly those characteristics and properties of the modeled entity that are relevant for a given task. A model is minimal with respect to a task if it does not contain any other characteristics than those relevant for the task*.

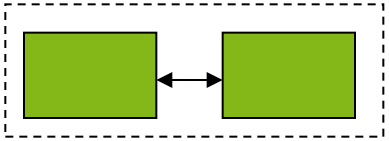
[Jantsch, 2004]:

Wishlist for specification & modeling techniques

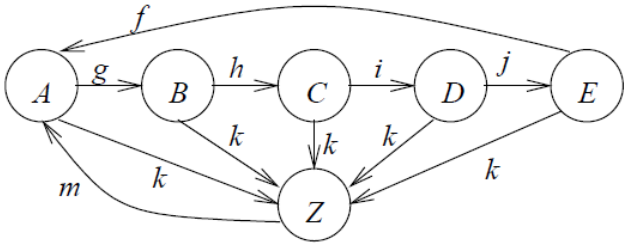
- **Hierarchy**
  - Humans cannot understand systems containing > ~5 objects.
    Most actual systems require more objects
    ☞ Hierarchy (+ abstraction)



- **Component-based design**
  - Systems must be designed from components

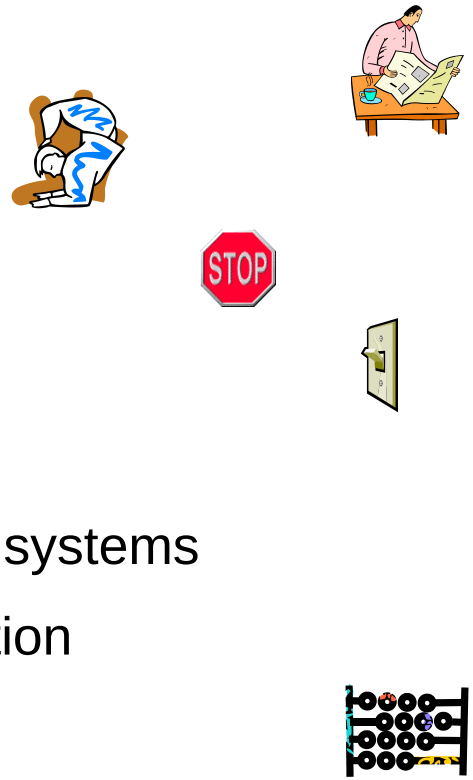


- **Support for designing reactive systems**
  - State-oriented behavior
  - Event-handling
  - Exception-oriented behavior



Wishlist for specification & modeling techniques (2)

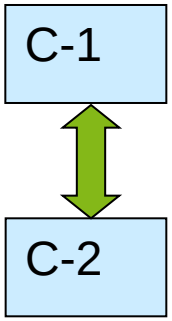
- Presence of programming elements
- Executability (no algebraic specification)
- Support for the design of large systems (☞ Object Orientation)
- Domain-specific support
- Readability
- Portability and flexibility
- Termination
- Support for non-standard I/O devices
- Non-functional properties
- Support for the design of dependable systems
- No obstacles for efficient implementation
- Adequate model of computation



Models

Models of computation should define:

- Components and an execution model for computations for each component
- Communication model for exchange of information between components.



Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Some models of computation

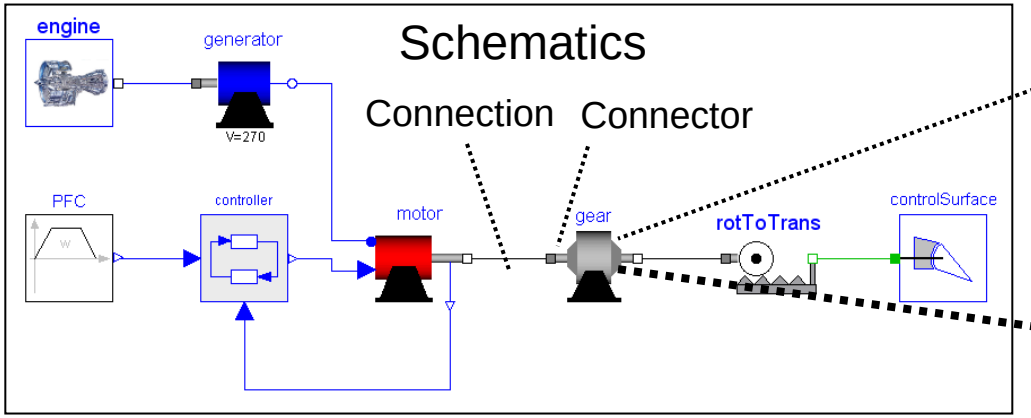
More details in my textbook & on youtube (cyphysystems)

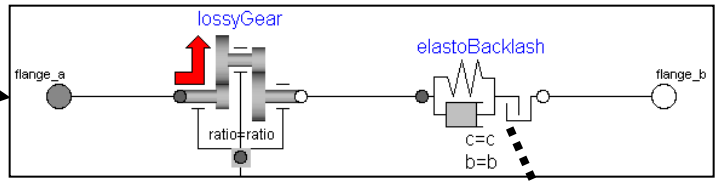
Communication/ local computations	Shared memory	Message passing Synchronous | Asynchronous	
Undefined components	Plain text, use cases | (Message) sequence charts		
Different. equations	Modelica, Matlab, ….		
CFSM ☞ *Kim G. Larsen*	StateCharts		SDL
Data flow	Scoreboarding + Tomasulo Algor. -> Comp.Architecture		Kahn networks, SDF
Petri nets		C/E nets, P/T nets, …	
Discrete event (DE) model	VHDL,Verilog, SystemC*, …	Only experimental systems, e.g. distributed DE in Ptolemy	
V.Neumann threads	C, C++,Java	C, C++, Java with libraries CSP, ADA |	

Radu Grosu

* Classification based on implementation with centralized data structures

Integrated simulation: Modelica






```
equation
        0 = flange_a.tau + flange_b.tau;
  phi_rel = flange_b.phi - flange_a.phi;
    w_rel = der(phi_rel);
        ...
```

- Each **Icon** represents a **physical component**. (electrical resistance, mechanical device, pump, ...)
- A **connection line** represents the actual physical **coupling** (wire, fluid flow, heat flow, ...)
- A component consists of **connected** sub-components (= hierarchical structure) and/or is described by **equations**.
- By **symbolic** algorithms, the high level Modelica description is transformed into a set of explicit differential equations:

$$0 = \mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{y}(t), t)$$

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t), t)$$

https://www.modelica.org/education/educational-material/ lecture-material/english/ ModelicaOverview.ppt

Break

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Worst case execution times (WCETs)

Complexity:

- in the general case: undecidable if a bound exists.
- for restricted programs: simple for “old“ architectures, complex for new architectures with pipelines, caches, interrupts, virtual memory, etc.

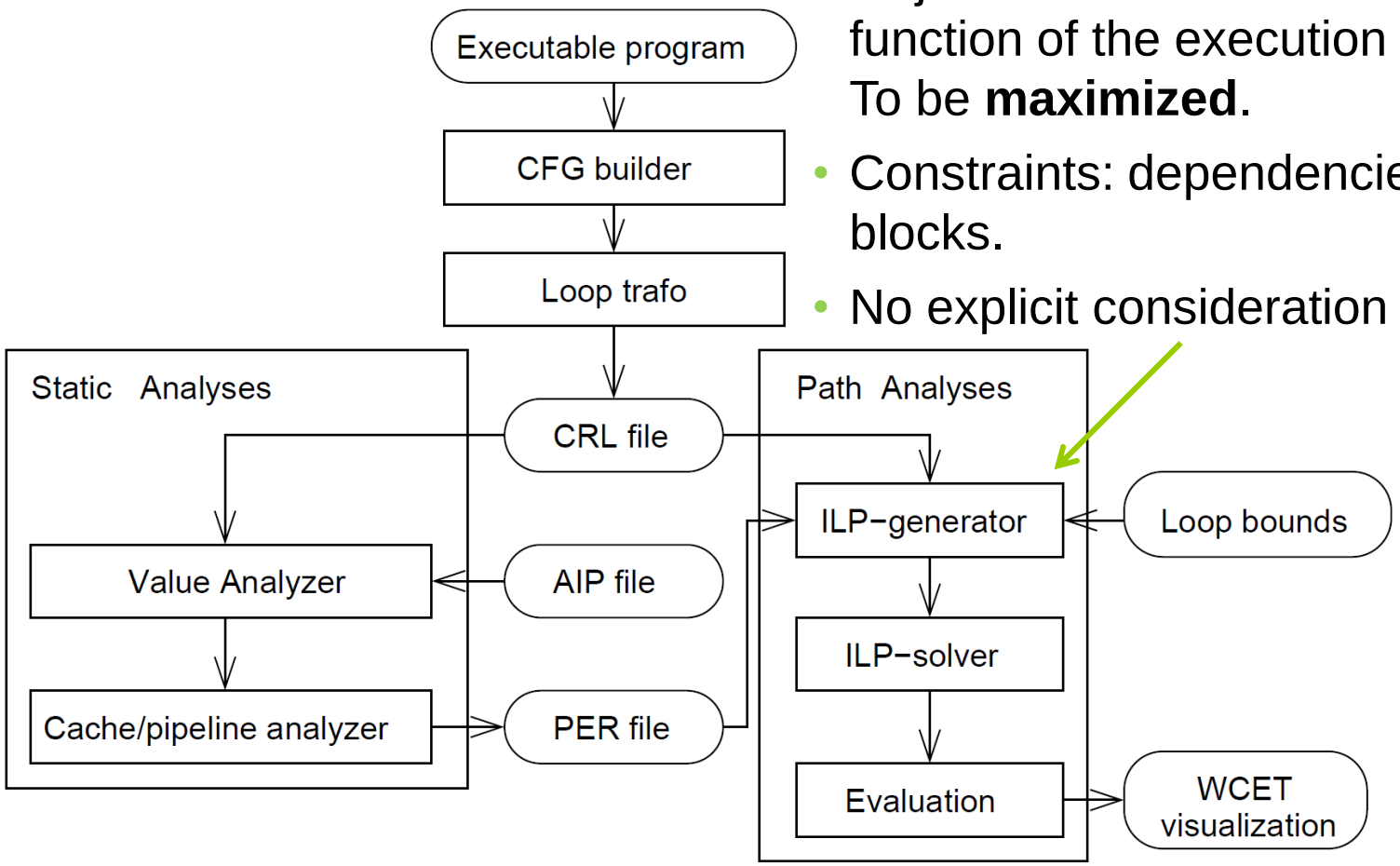
Approaches:

- for hardware: requires detailed timing behavior
- for software: requires machine programs; complex analysis

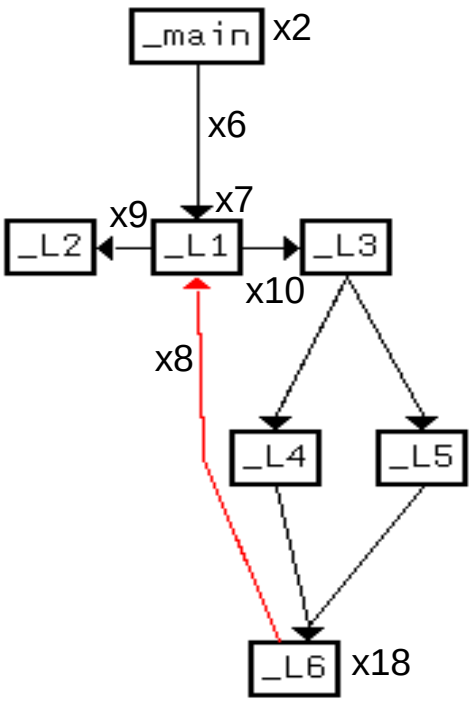
WCET estimation: AiT (AbsInt)

Implicit path enumeration technique

- Objective function: execution time as a function of the execution time of blocks. To be **maximized**.
- Constraints: dependencies between blocks.
- No explicit consideration of all paths



Implicit path enumeration technique (IPET): Example



_main: 21 cycles
_L1: 27
_L3: 2
_L4: 2
_L5: 20
_L6: 13
_L2: 20

Equations per node

- Sum of incoming edge variables = #exec. of node
- Sum of outgoing edge variables = #exec. of node

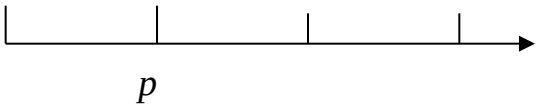
ILP

```
/* Objective function = WCET to be maximized*/
21 x2 + 27 x7 + 2 x11 + 2 x14 + 20 x16 + 13 x18 + 20 x19;
/* CFG Start Constraint */ x0 - x4 = 0;
/* CFG Exit Constraint */  x1 - x5 = 0;
/* Constraint for flow entering function main */
x2 - x4 = 0;
/* Constraint for flow leaving exit node of main */
x3 - x5 = 0;
/* Constraint for flow entering exit node of main */
x3 - x20 = 0;
/* Constraint for flow entering main = flow leaving main */
x2 - x3 = 0;
/* Constraint for flow leaving CFG node _main */
x2 - x6 = 0;
/* Constraint for flow entering CFG node _L1 */
x7 - x8 - x6 = 0;
/* Constraint for flow leaving CFG node _L1 */
x7 - x9 - x10 = 0;
/* Constraint for lower loop bound of _L1 */
x7 - 101 x9 >= 0;
/* Constraint for upper loop bound of _L1 */
x7 - 101 x9 <= 0; ….
```

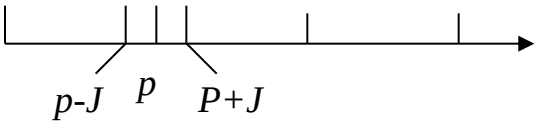
Real-time calculus (RTC)/ Modular performance analysis (MPA)

Streams of events important: Examples

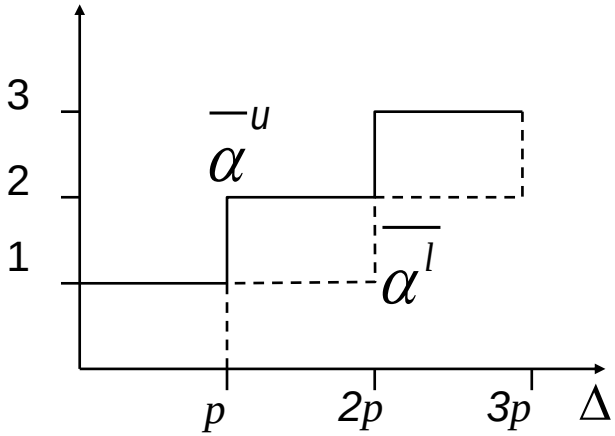
periodic event stream

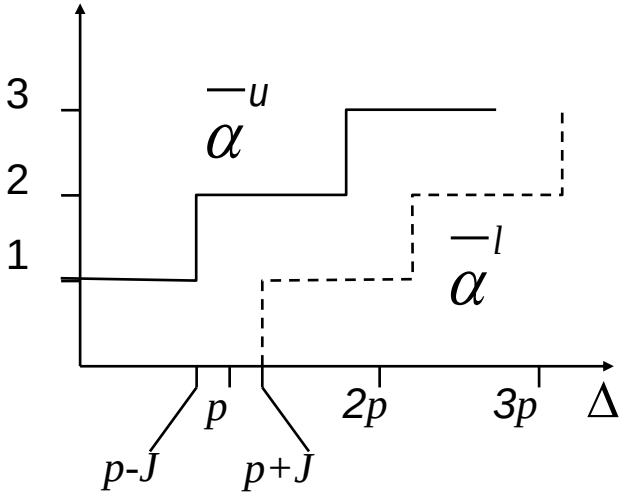


periodic event stream with jitter



Thiele et al. (ETHZ): Extended **network calculus:**
Arrival curves describe the maximum and minimum number of events arriving in some time interval Δ.

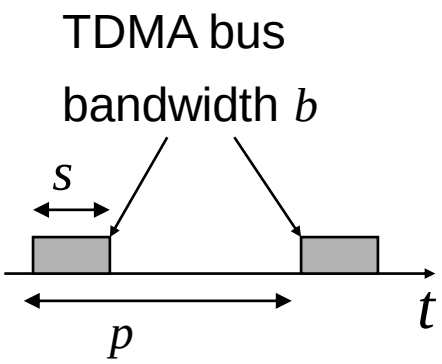


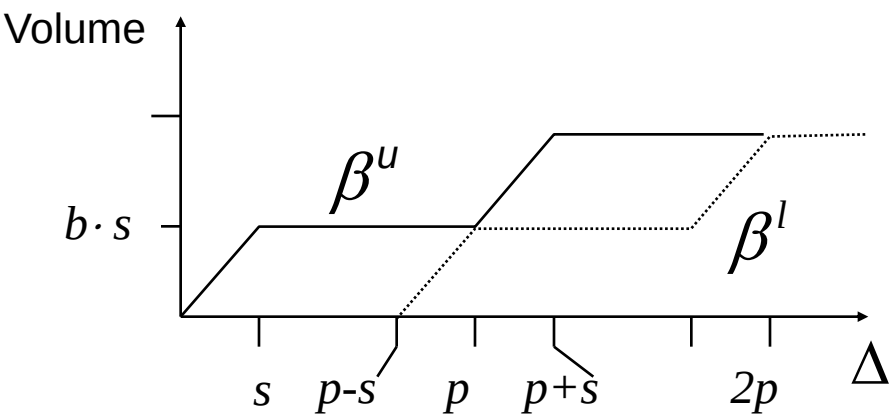


RTC/MPA: Service curves

Service curves β^u resp. β^ℓ describe the maximum and minimum service capacity available in some time interval Δ

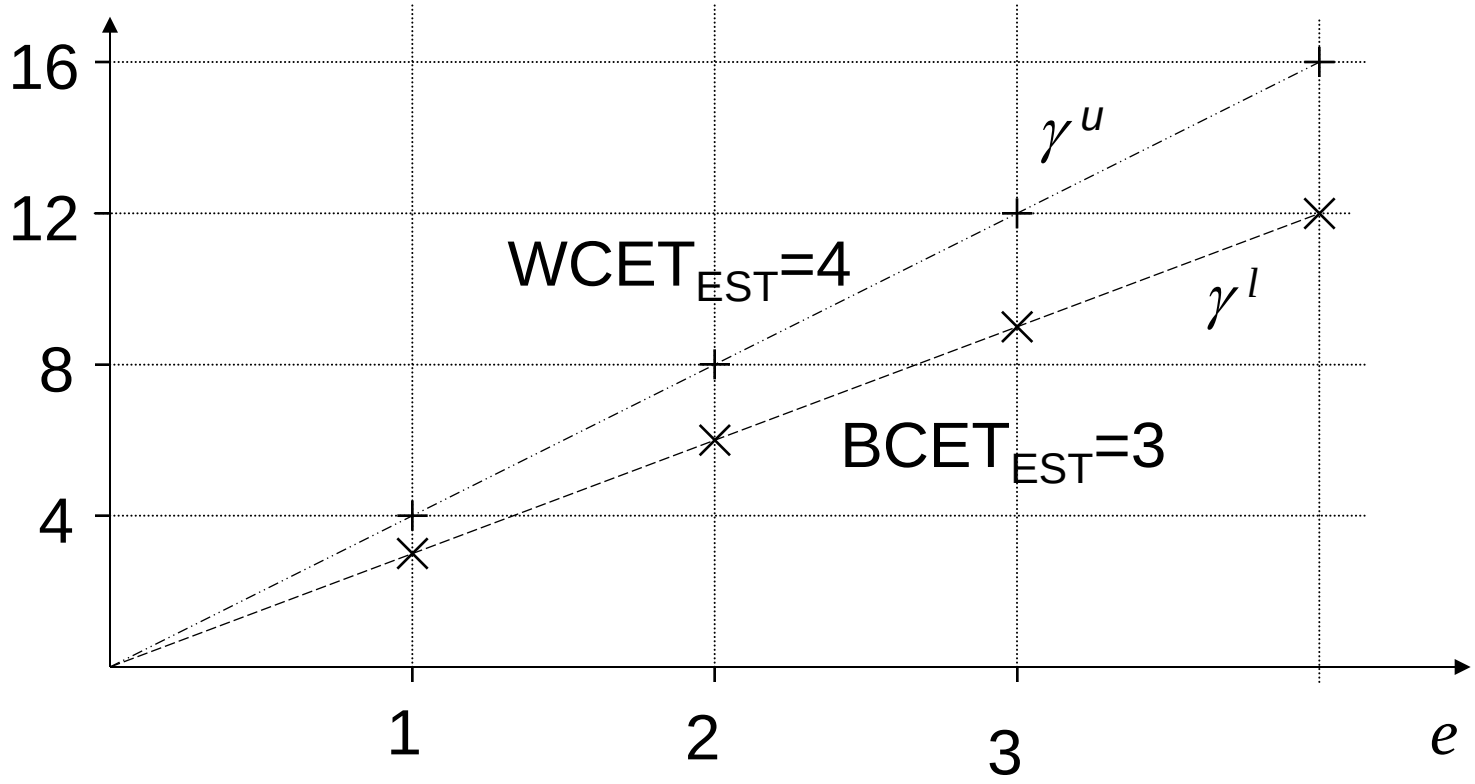
Example:





RTC/MPA: Workload characterization

γ^u resp. γ^ℓ describe the maximum and minimum service capacity required as a function of the number e of events. Example:



(Defined only for an integer number of events)

RTC/MPA: Workload required for incoming stream

Incoming workload

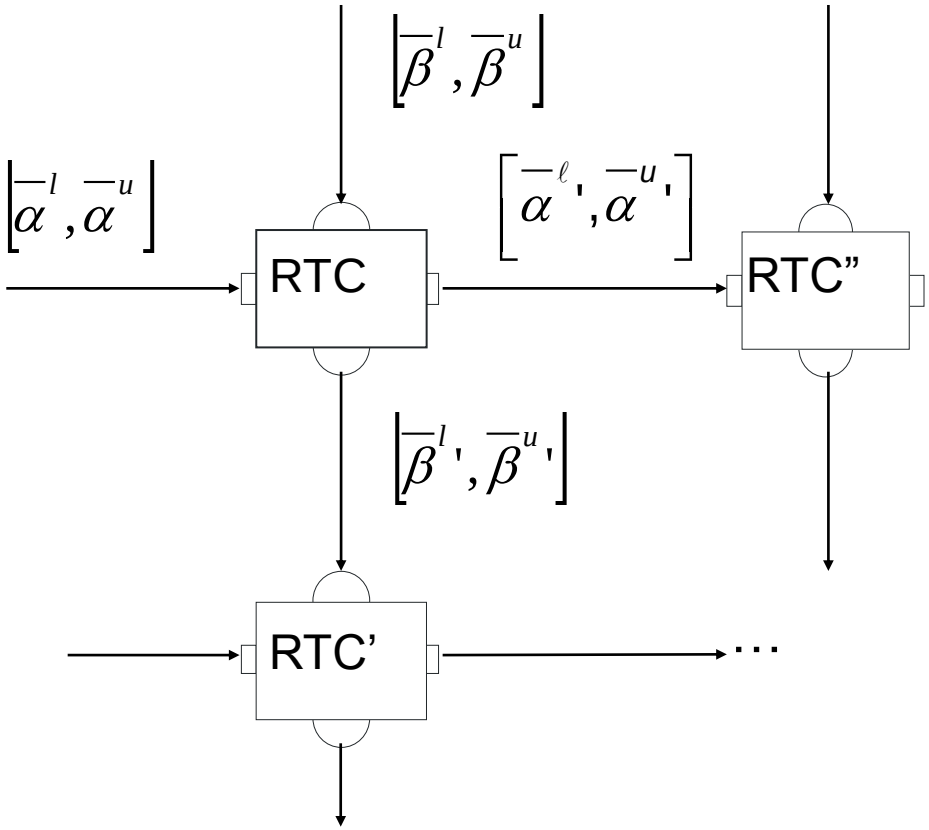
$$\alpha^u(\Delta) = \gamma^u\left(\overline{\alpha^u}(\Delta)\right) \qquad \alpha^l(\Delta) = \gamma^l\left(\overline{\alpha^l}(\Delta)\right)$$

Upper and lower bounds on the number of events

$$\overline{\beta}^u(\Delta) = \left(\gamma^l\right)^{-1}\left(\beta^u(\Delta)\right) \qquad \overline{\beta}^l(\Delta) = \left(\gamma^u\right)^{-1}\left(\beta^l(\Delta)\right)$$

RTC/MPA: System of real time components

Incoming event streams and available capacity are transformed by real-time components:



Outline

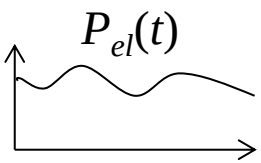
- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Evaluation of energy consumption E: Challenges

$$E = \int_0^{t_{\max}} P_{el}(t)\,dt, \quad t_{\max} \leq \text{WCET}_{\text{EST}}$$



- E hardly predictable from source code
  Small code variations $\rightarrow$ possibly large variations in E

☞ E must be predicted from executable code
(like the $WCET$)

- E might depend on instance of HW

How to obtain $P_{el}(t)$?

- Measurements (potentially precise, fixed architecture):
  - Currents difficult to measure precisely
  - Tiwari (1994), Russell, Jacome (1998), ..
  - Odroid (see later slide)

- Models (flexible architecture, less precise):
  - (H)SPICE, CACTI [Jouppi, 1996], Wattch [Brooks, 2000]:
  - Commercial offerings
- Combined models
  - Steinke et al., TU Dortmund (2001), …

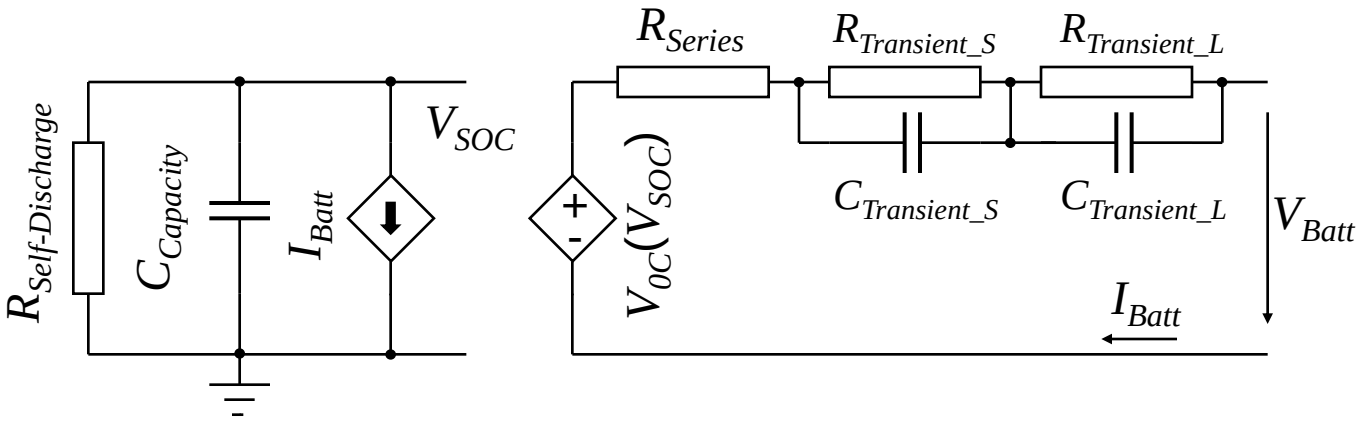
Battery models

- (Chemical) & physical models
  e.g. concentrated solution theory, partial differential eq.s
- Empirical models
  Simple equations, inaccurate
  - Peukert's law: lifetime= C/I^{α}, with empirical α
  - Weibull fit
- Abstract models
  - Electrical circuit models
  - Discrete time model (e.g. in VHDL)
  - Stochastic models (e.g. Markov processes)
- Mixed models
  e.g. electrical models with physical explanation

frequently with fitting

Sources include: R. Rao, S. Vrudhula, D.N. Rakhmatov: Battery Modeling for Energy-Aware System Design, IEEE Computer, 2003, pp. 77

Model proposed by Chen and Rincón-Mora



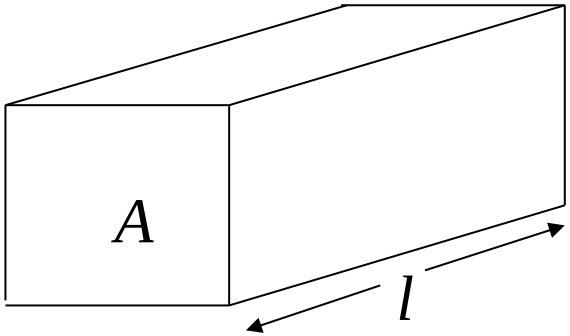
Source: M. Chen, G. A. Rincón-Mora: Accurate Electrical Battery Model Capable of Predicting Runtime and *I-V* Performance, *IEEE Trans. on Energy Conversion*, 2006, pp. 504

- Full charge capacitor: $C_{Capacity} = 3600 \cdot Capacity \cdot f_1(\text{cycle}) \cdot f_2(\text{Temp})$
- Self-discharge resistor: $R_{Self\text{-}Discharge}$ (might depend on parameters)
- Current dependency of V_{Batt}: modeled by $R_{series}+R_{Transient\_S}+R_{Transient\_L}$
- I_{Batt} charges and discharges $C_{Capacity}$
- Voltage controlled voltage source V_{0C} captures nonlinear dependency between the state of charge and V_{0C} (measurement can take days)
- R_{Series}: models immediate voltage drop at load change

Evaluation of thermal behavior

Thermal conductivity

$$P_{th} = \kappa \frac{\Delta T \cdot A}{l} \quad (1)$$



Where

P_{th}	: thermal power transferred
κ	**: thermal conductivity**
ΔT	: temperature difference
A	: area
l	: length

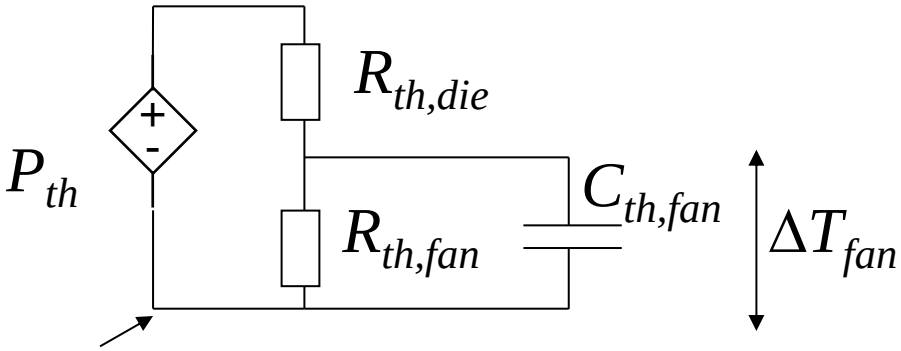
Thermal resistance

$$R_{th} = \frac{\Delta T}{P_{th}} = \frac{l}{\kappa \cdot A} \quad (2)$$

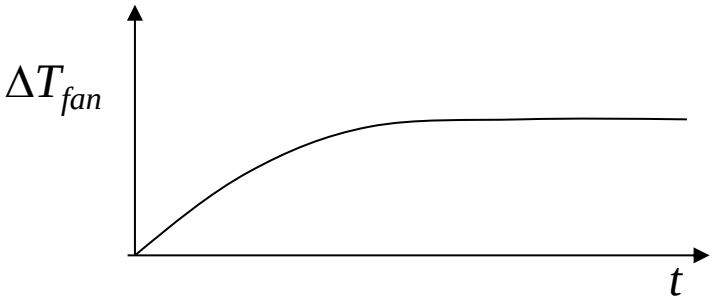
Dynamic thermal behavior

Thermal capacity:

$C_{th} = m \cdot c$, C_{th}: thermal capacity, m: mass, c: specific heat



Ground ≈ Reference temperature



ΔT_{fan}: transient with time constant

Equivalences

Electrical model		Thermal model	
Current	I	Thermal flow, flow of “power”	$P_{th}=\dot{Q}$
Total charge	$Q = \int I\, dt$	Thermal energy	$E_{th} = \int P_{th}\, dt$
Resistance	R	Thermal resistance	R_{th}
Potential	φ	Temperature	T
Voltage = potential difference	$U=\Delta\varphi$	Temperature difference	ΔT
Capacitance	C	Thermal capacitance	C_{th}
Ohm’s law	$U = R\, I$	ΔTemperature at R_{th}	$\Delta T = R_{th} \cdot P_{th}$

Impact of temperature on reliability

Approximation: Black's Equation:

$$\mathrm{MTTF} = \frac{A}{{j_e}^n}\, e^{\frac{E_a}{kT}}$$

where

A : constant

j_e : current density

n : constant (1..7, controversial, 2 according to Black)

E_a : activation energy (e.g. ~ 0.6 eV)

k : Boltzmann constant (~ 8.617 10^{-5} eV/K)

T : Temperature

- J. Bernstein et al.: Electronic circuit reliability modeling, Microelectronic Reliability 46 (2006) 1957-1979
- J.R. Black.: Electromigration Failure Modes in Aluminium Metallization for Semiconductor Devices. Proceedings of the IEEE, Vol. 57, p. 1587-1594, 1969

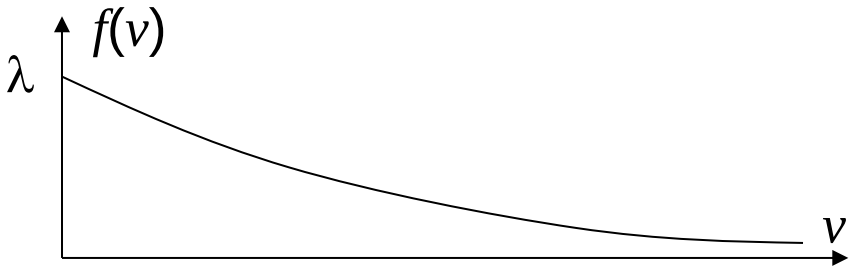
Reliability: $f(v)$, $F(v)$

- Let V: time until first failure (random variable) *
- Let $f(v)$ be the density function of V *

Example: Exponential distribution

$f(v)=\lambda e^{-\lambda v}$

Appropriate for constant failure rate λ



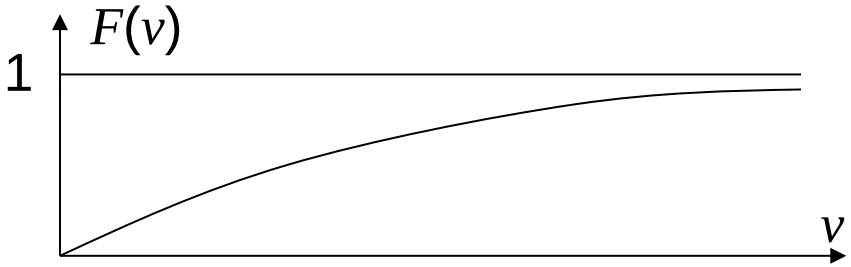
- $F(v)$ = probability of the system being faulty at time v:

$F(v) = \Pr(V \leq v)$

$$F(v) = \int_0^v f(x)dx$$

Example: Exponential distribution

$$F(v) = \int_0^v \lambda e^{-\lambda x} dx = -[e^{-\lambda x}]_0^v = 1 - e^{-\lambda v}$$



* Replacing commonly used T by V and t by v to avoid confusion with temperatures.

MTTF = $E\{V\}$, the *statistical mean* value of V

$$\text{MTTF} = E\{V\} = \int_0^\infty v \cdot f(v)\, dv$$

According to the definition of the statistical mean value

Example: Exponential distribution

$$\text{MTTF}_{\exp} = \int_0^\infty v \cdot \lambda e^{-\lambda v} dv = -\cancel{\left[v \cdot e^{-\lambda v}\right]_0^\infty} + \int_0^\infty e^{-\lambda v} dv$$

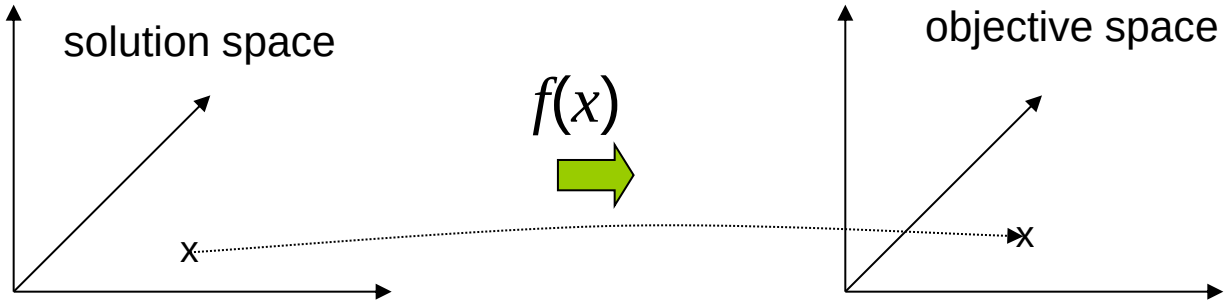
$$\int u \cdot w' = u \cdot w - \int u' \cdot w$$

$$\text{MTTF}_{\exp} = -\frac{1}{\lambda}\left[e^{-\lambda t}\right]_0^\infty = -\frac{1}{\lambda}\left[0 - 1\right] = \frac{1}{\lambda}$$

MTTF is the reciprocal value of failure rate.

Which design should we select?

- Let X: m-dimensional **solution space** for the design problem. Example: dimensions correspond to # of processors, size of memories, type and width of busses etc.
- Let F: n-dimensional **objective space** for the design problem. Example: dimensions correspond to average and worst case delay, power/energy consumption, size, weight, reliability, …
- Let $f(x)=(f_1(x),\ldots,f_n(x))$ where $x \in X$ be an **objective function.** We assume that we are using $f(x)$ for evaluating designs.



Pareto points

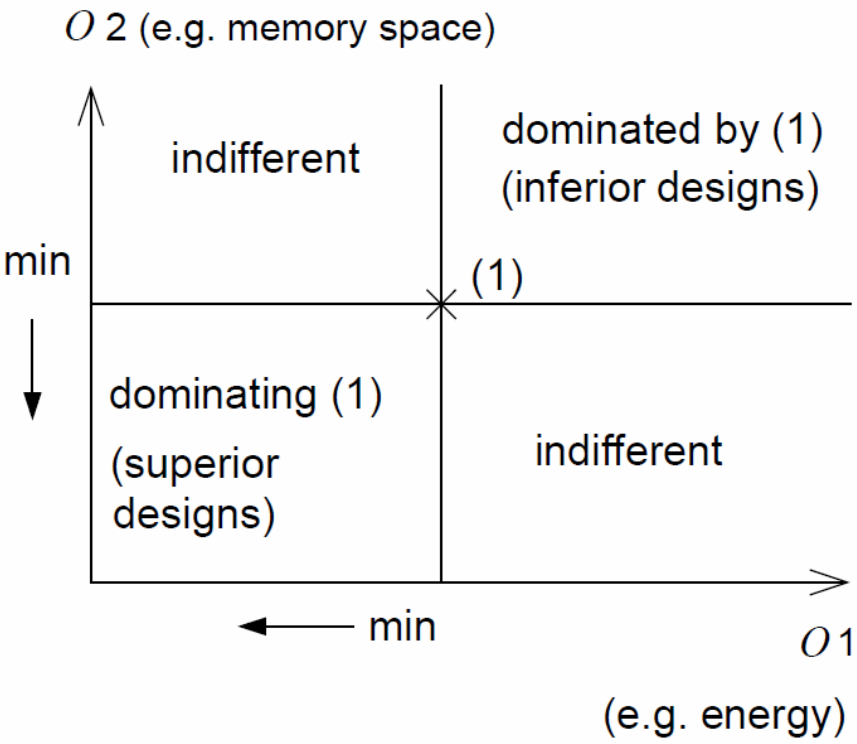
- We assume that, for each objective, an order < and the corresponding order $\leq$ are defined.
- **Definition**:
  Vector $u=(u_1,\ldots,u_n)\in F$ **dominates** vector $w=(w_1,\ldots,w_n)\in F$
  $\Leftrightarrow$
  u is "better" than w with respect to one objective and not worse than w with respect to all other objectives:

$$\forall i \in \{1,\ldots,n\} : u_i \leq w_i \wedge$$
$$\exists i \in \{1,\ldots,n\} : u_i < w_i$$

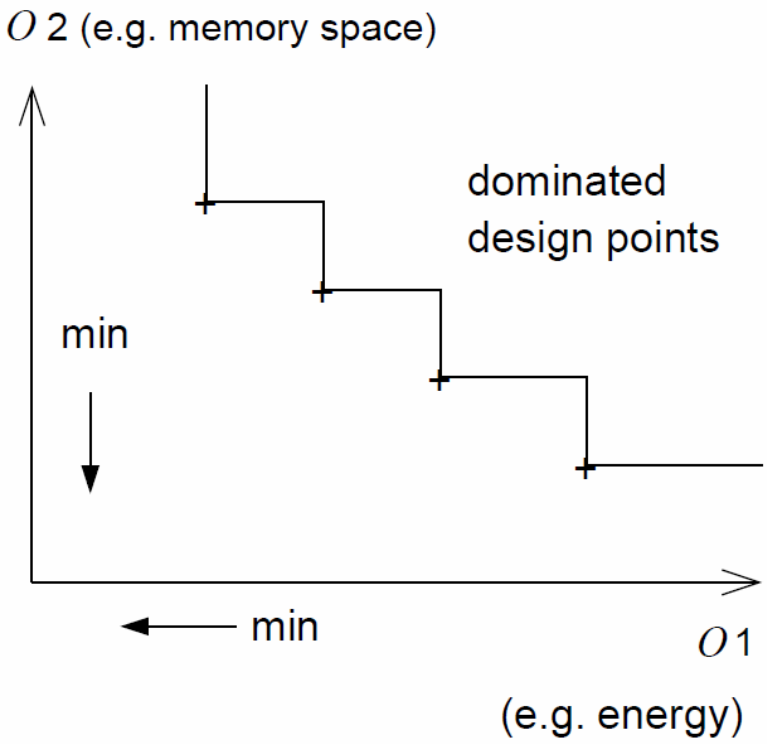
- A solution $x\in X$ is called **Pareto-optimal** with respect to X $\Leftrightarrow$ there is no solution $y\in X$ such that $u=f(x)$ is dominated by $w=f(y)$. x is a **Pareto point**.

Graphically

Pareto point



Pareto front



Outline

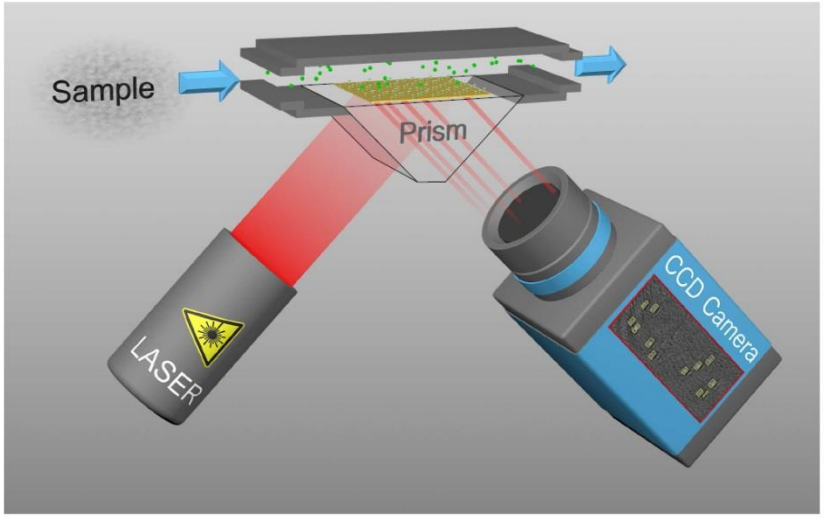
- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

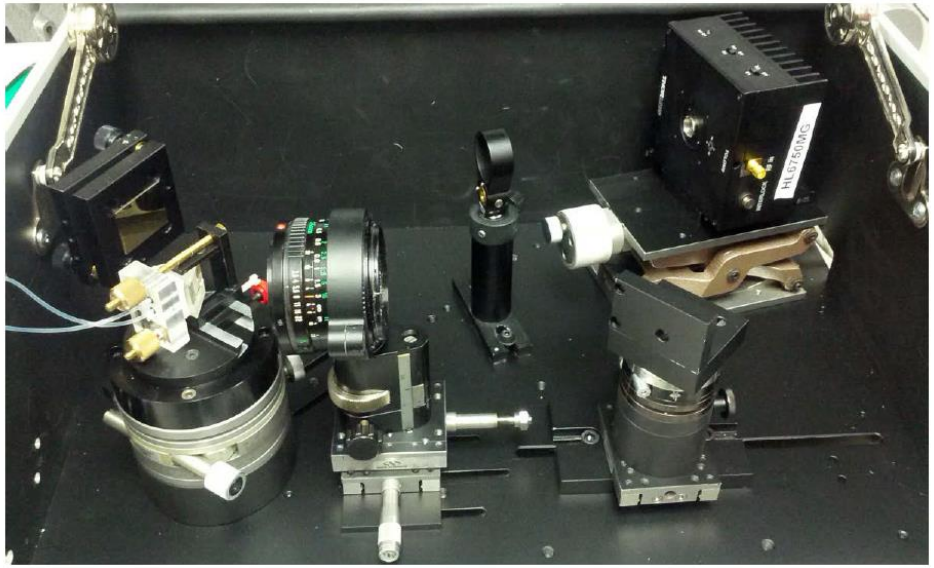
Standard techniques

Own results

Energy efficient (bio-) virus detection

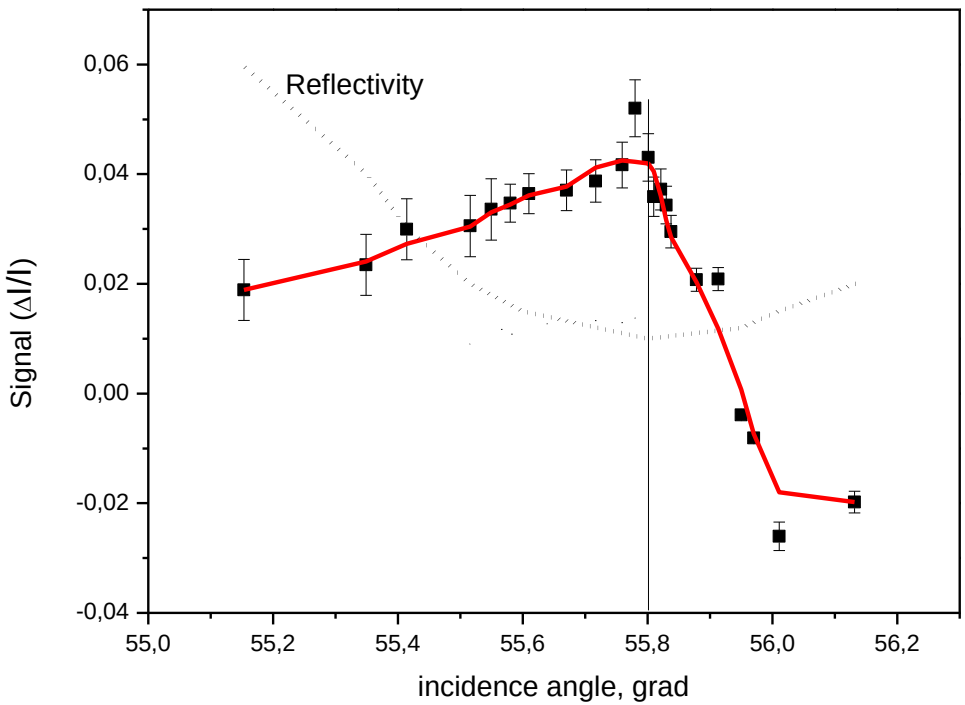
Detection of nano objects using **plasmon effect**
(Optical effect of objects ≪ wavelength of light)

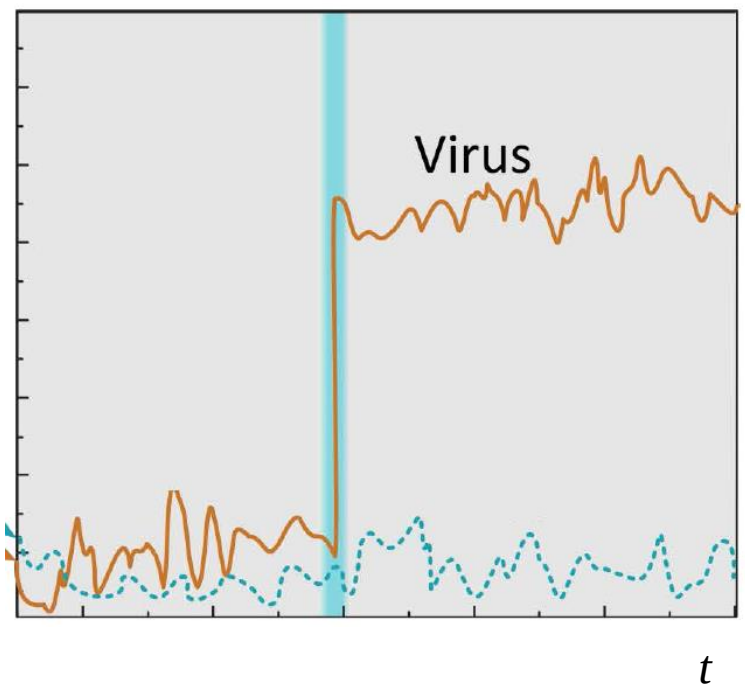




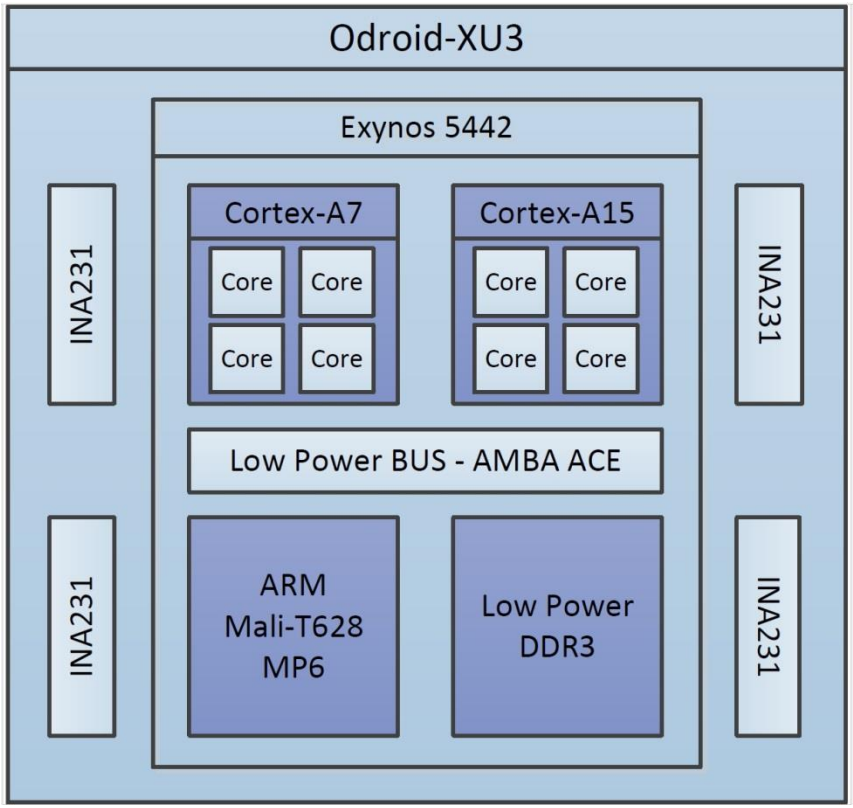
Truly a cyber-physical system

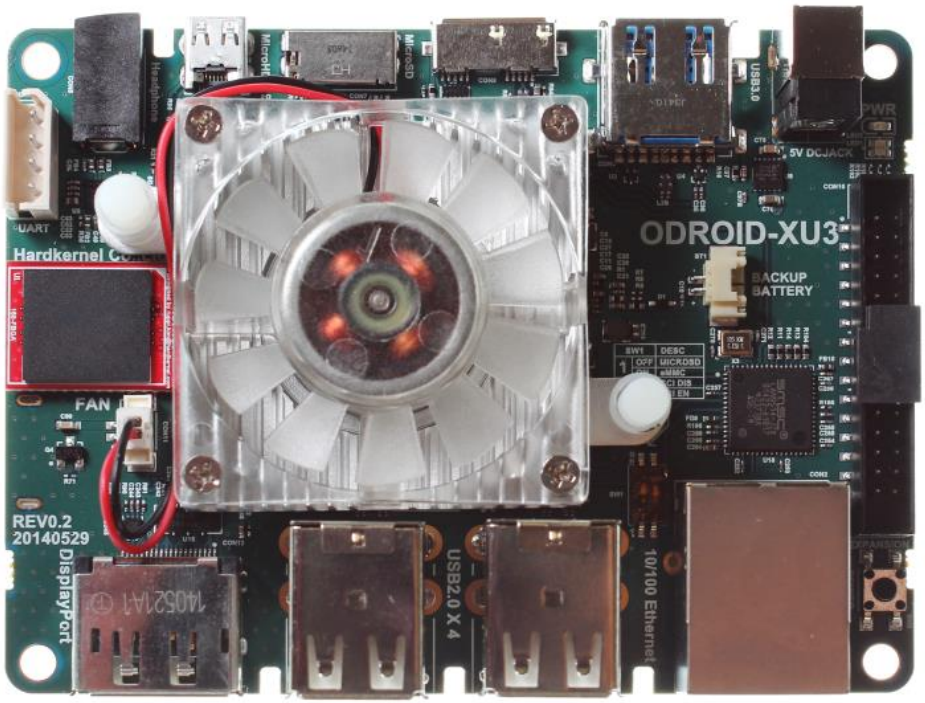
Reflectivity changed if virus attaches to surface





Mapping to Embedded Odroid (1)

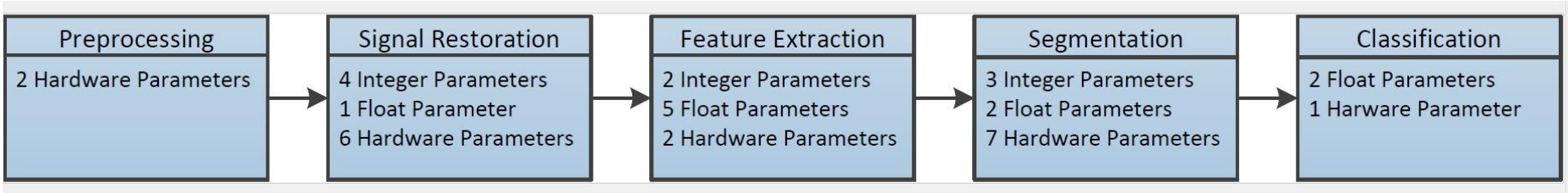


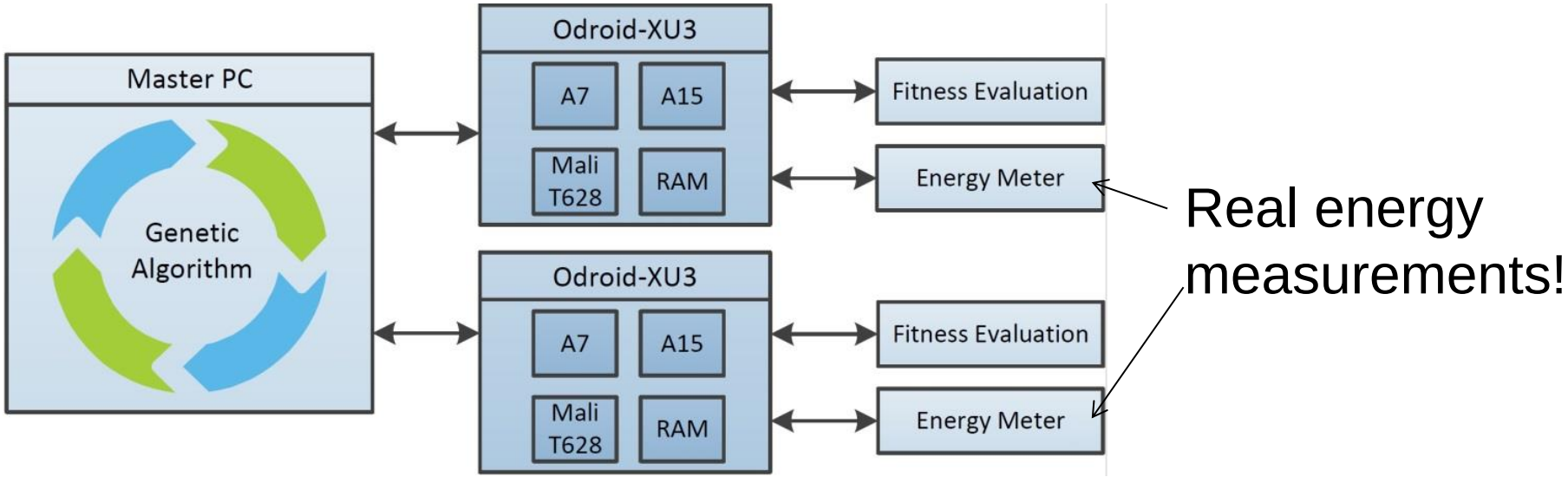


Allows power measurements!

O. Neugebauer, P. Libuschewski, M. Engel, H. Müller, P. Marwedel: Plasmon-based Virus Detection on Heterogeneous Embedded Systems, SCOPES, 2015

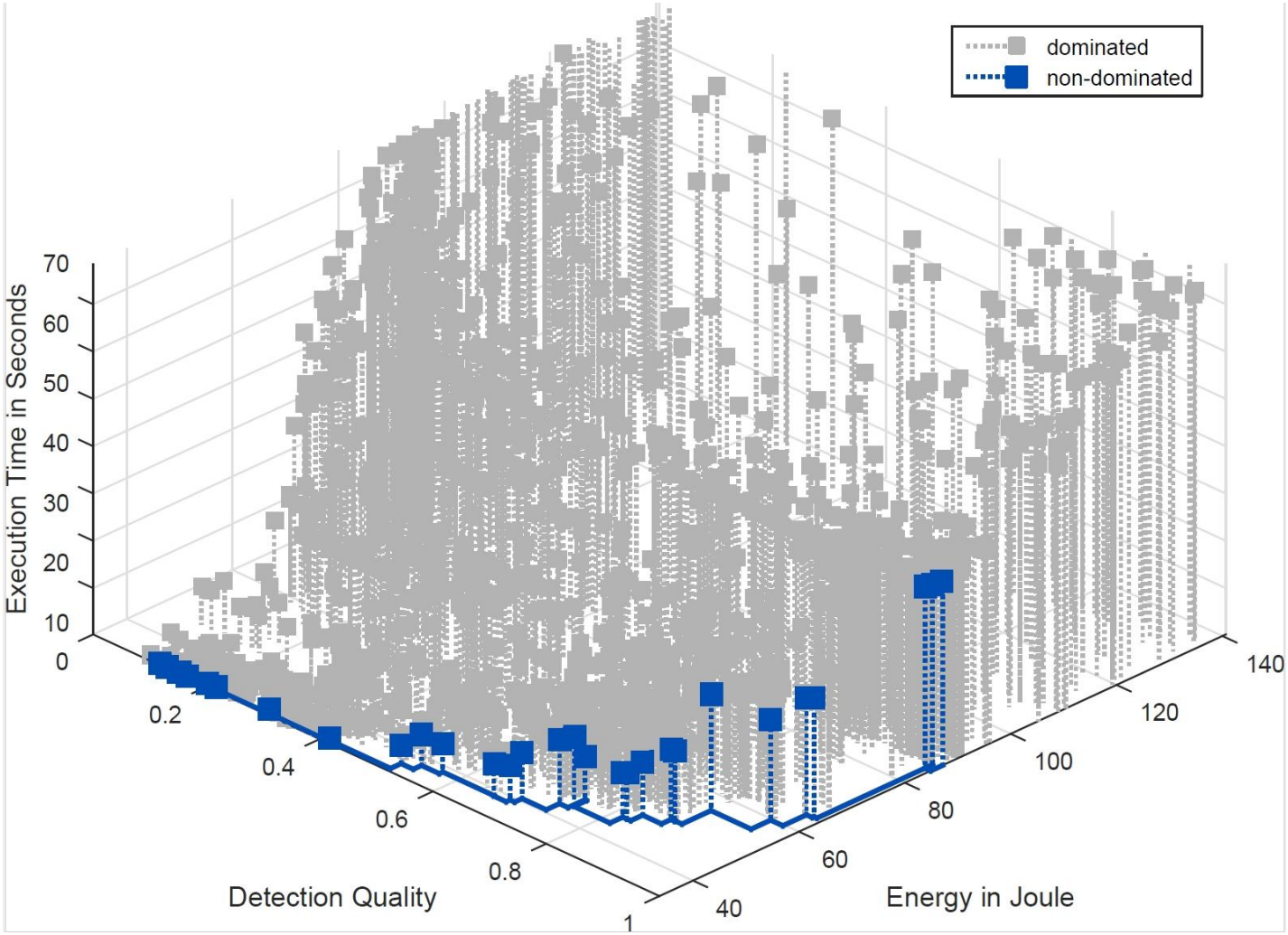
Mapping to Embedded Odroid (2)





O. Neugebauer, P. Libuschewski, M. Engel, H. Müller, P. Marwedel: Plasmon-based Virus Detection on Heterogeneous Embedded Systems, SCOPES, 2015

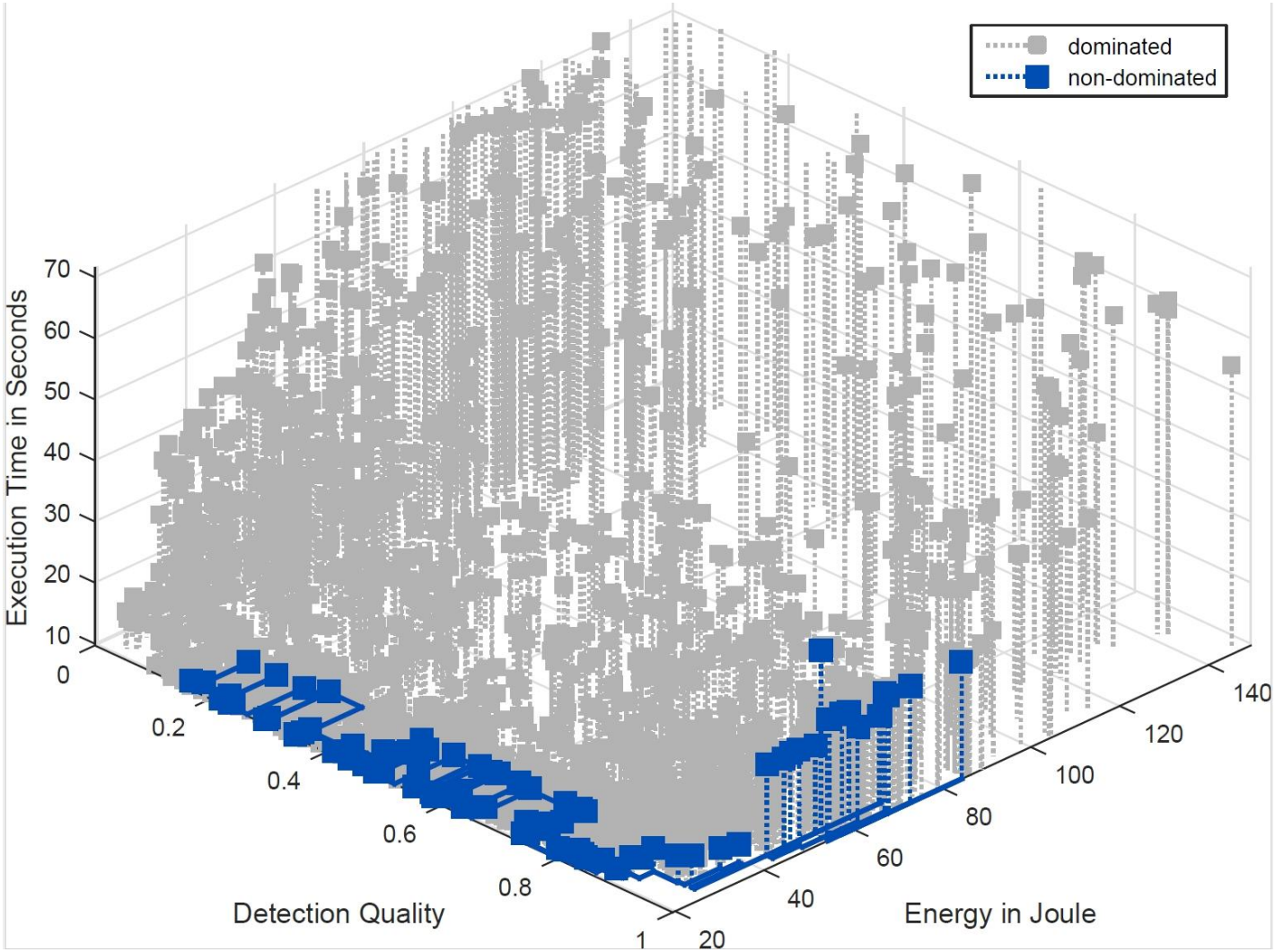
Results (1)



Tradeoff between detection quality and energy consumption!

O. Neugebauer, P. Libuschwewski, M. Engel, H. Müller, P. Marwedel: Plasmon-based Virus Detection on Heterogeneous Embedded Systems, SCOPES, 2015

Results (2)



Optimization of both hardware and software

O. Neugebauer, P. Libuschwewski, M. Engel, H. Müller, P. Marwedel: Plasmon-based Virus Detection on Heterogeneous Embedded Systems, SCOPES, 2015

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

How to take timing constraints into account?

1. Specification of CPS/ES system
2. Generation of Code (ANSI-C or similar)
3. Compilation of Code
4. Execution and/or simulation of code,
   using a (e.g. random) set of input data
5. Measurement-based computation of "estimated worst case execution time" ($WCET_{meas}$)
6. Adding safety margin m on top of $WCET_{meas}$:
   $WCET_{hypo} := (1 + m) * WCET_{meas}$
7. If "$WCET_{hypo}$" > deadline: change some detail, go back to 1 or 2.

Trial-and-error approach!

Problems with this Approach

Dependability

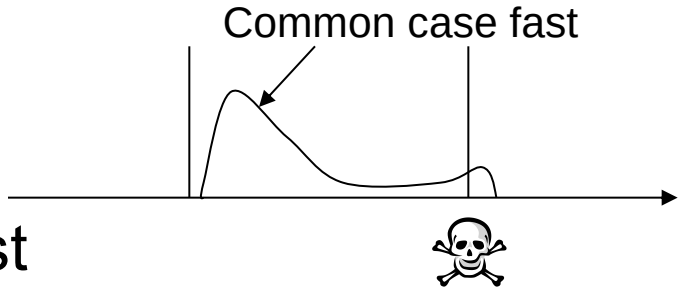
- Computed "$WCET_{hypo}$" not a safe approximation
- ☞ Time constraint may be violated

Design time

- How to find necessary changes?
- How many iterations until successful?

"Make the common case fast" a wrong approach for RT-systems

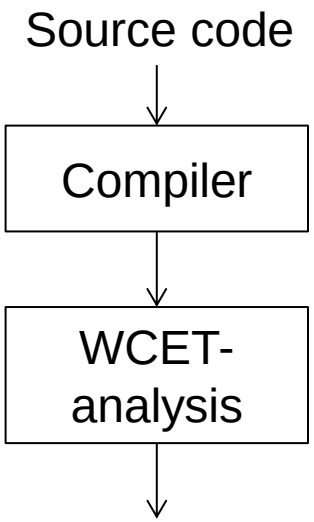
- Computer architecture and compiler techniques focus on average speed
- Circuit designers know it's wrong
- Compiler designers (typically) don't



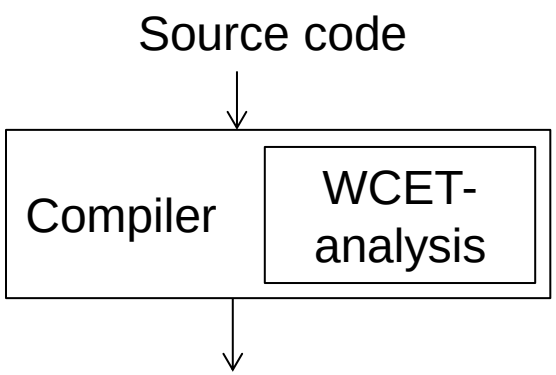
"Optimizing" compilers unaware of cost functions other than code size

Integration of formal $WCET_{EST}$ estimation

- **First step**: formal analysis of $WCET_{EST}$, using, for example, aiT of AbsInt
  - Remaining problem:
    Lack of cost function in compiler, code not optimized for $WCET_{EST}$



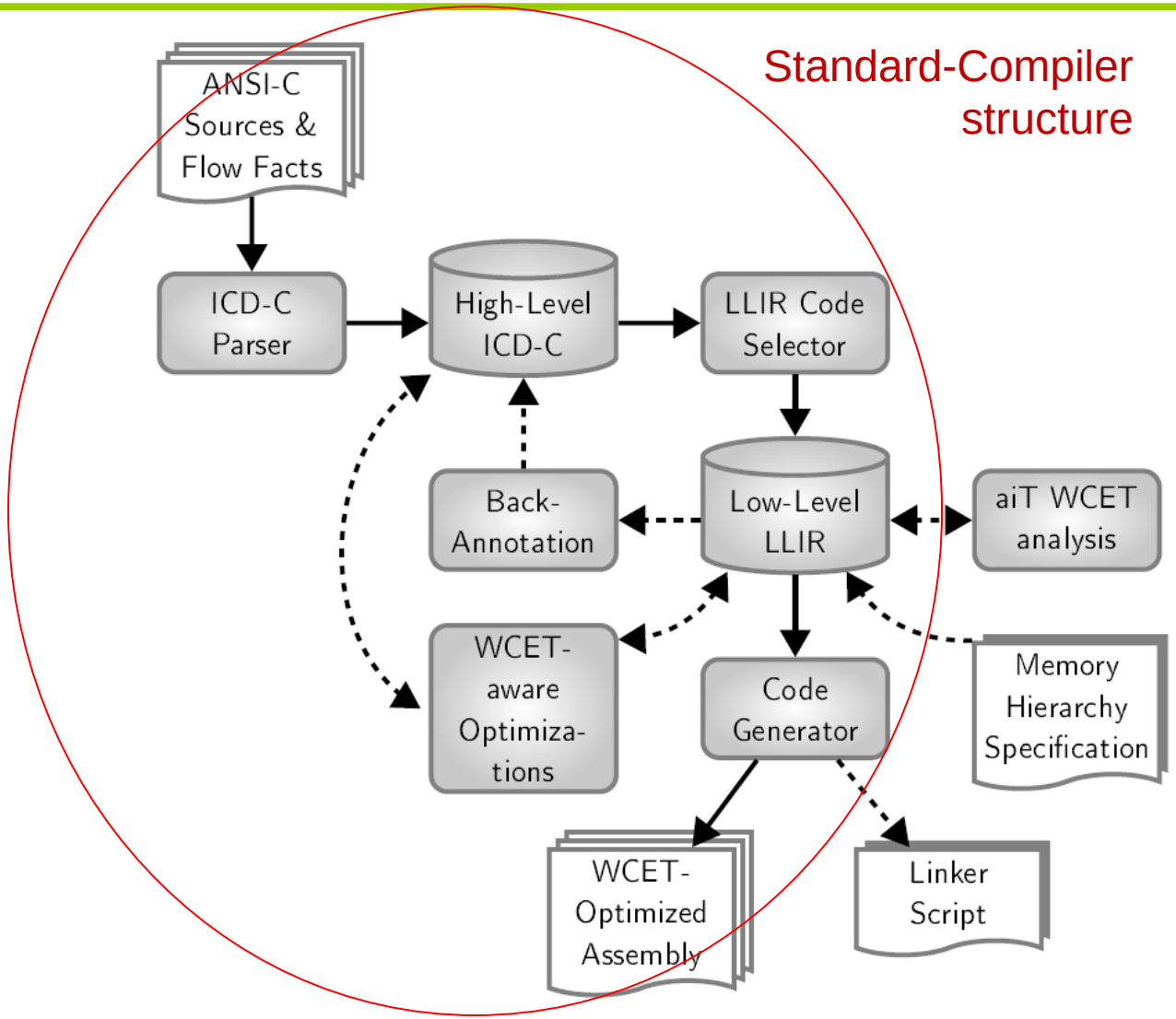
- **Second step**: integration of WCET bound computation (e.g. aiT) into compiler, generation of $WCET_{EST}$-optimized code



Integration of $WCET_{EST}$ estimation and compilation

Why not consider $WCET_{EST}$ as an objective function already in the compiler?

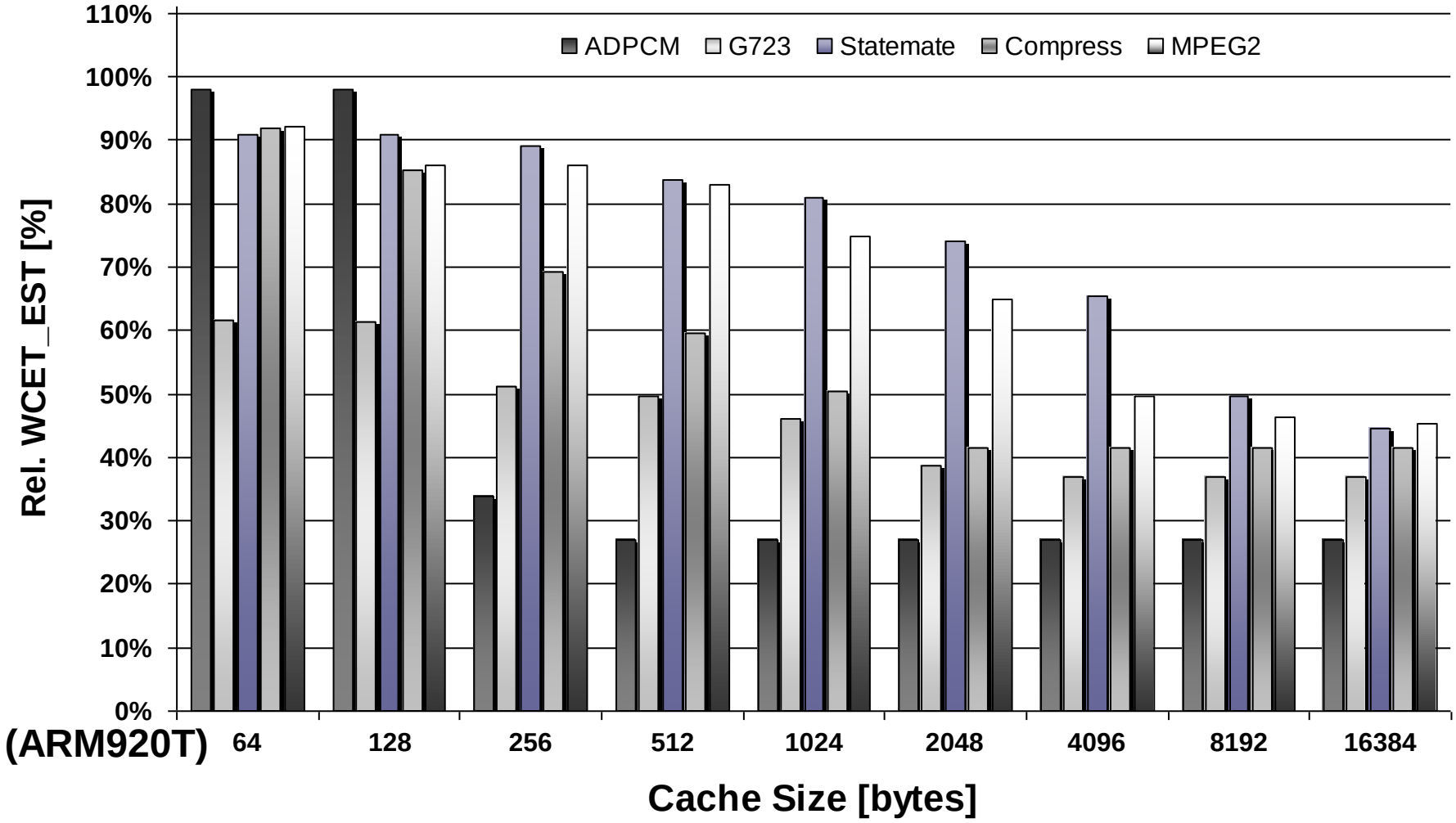
☞ Integration of $WCET_{EST}$ analysis tool aiT into compiler



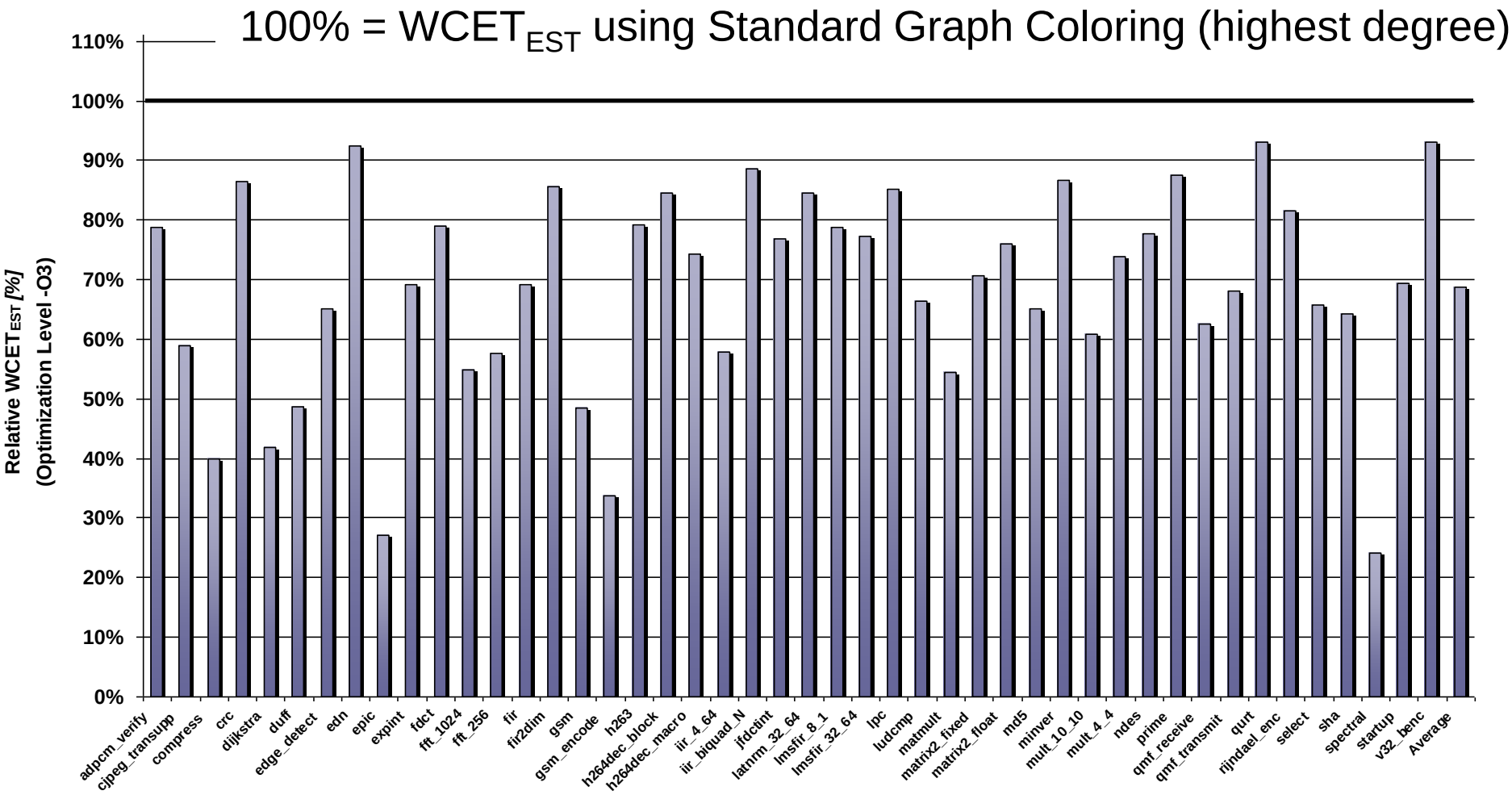
WCET-oriented optimizations

- Extended loop analysis (CGO 09)
- Instruction cache locking (CODES/ISSS 07, CGO 12)
- Cache partitioning (WCET-Workshop 09)
- Procedure cloning (WCET-WS 07, CODES 07, SCOPES 08)
- Procedure/code positioning (ECRTS 08, CASES 11 (2x))
- Function inlining (SMART 09, SMART 10)
- Loop unswitching/invariant paths (SCOPES 09)
- Loop unrolling (ECRTS 09)
- Register allocation (DAC 09, ECRTS 11))
- Scratchpad optimization (DAC 09)
- Extension towards multi-objective optimization (RTSS 08)
- Superblock-based optimizations (ICESS 10)
- Surveys (Springer 10, Springer 12)

Relative $WCET_{EST}$ with I-*Cache Locking 5 Benchmarks/ARM920T/Postpass-Opt*



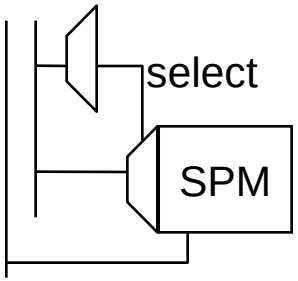
Register Allocation

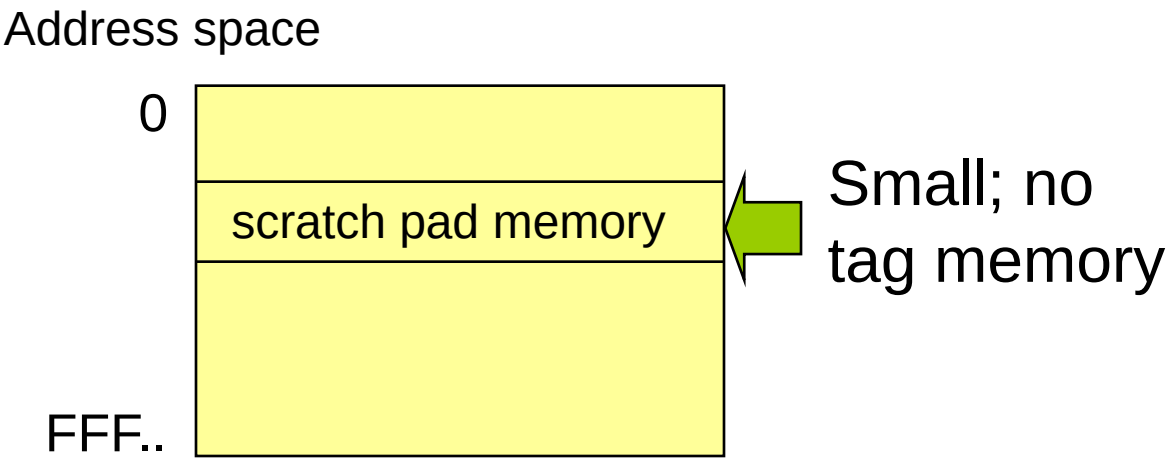


Scratch pad memories (SPM): Fast, energy-efficient, timing-predictable

SPMs are small, physically separate memories mapped into the address space;

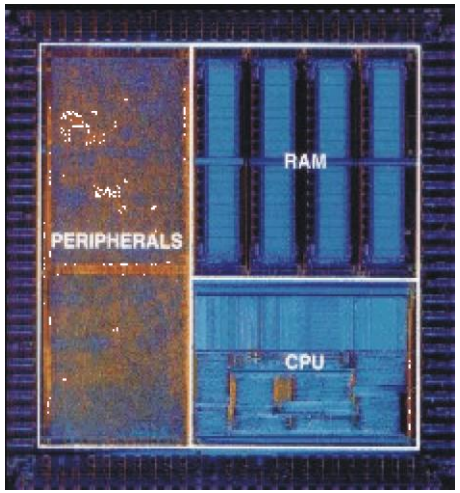
Selection is by an appropriate address decoder (simple!)





Example

ARM7TDMI cores, well-known for low power consumption



$WCET_{EST}$-aware SPM allocation

Setup

- Bosch Democar: Runnable: IgnitionSWCSync
  Part of task actuator, activated every 90° of crankshaft angle ☞ time-critical
- $WCET_{EST}$-aware scratch pad memory (SPM) allocation of program code by WCC, including fully automated $WCET_{EST}$ analyses using aiT and solution of the ILP for SPM allocation
- $WCET_{EST}$ reduced to about 50%, compared to gcc.

☞ PREDATOR project partners Bosch & Airbus interested in WCC

Outline

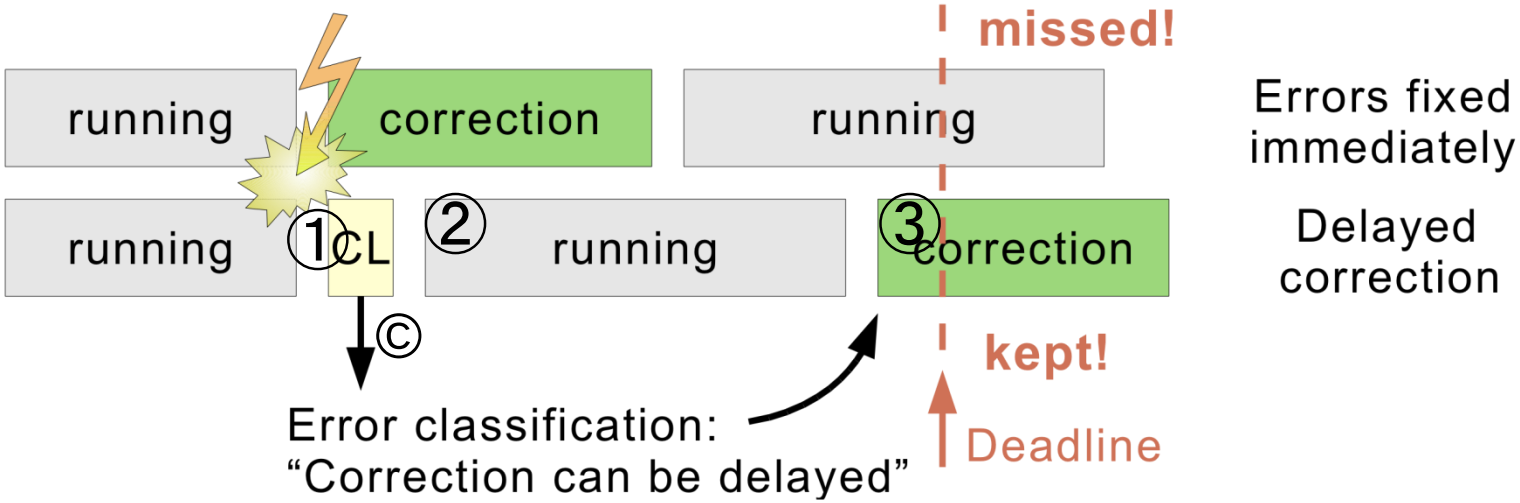
- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

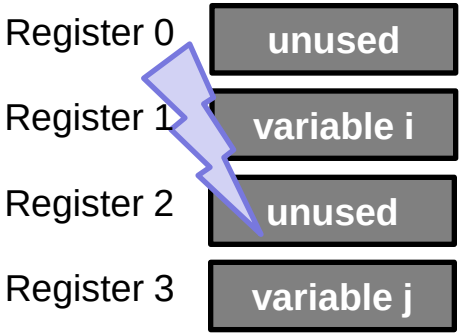
Tradeoff between reliability and timeliness

- Traditional error correction requires resources, including time, may violate time constraint of CPS
- Conflicting with timing requirements of real-time systems

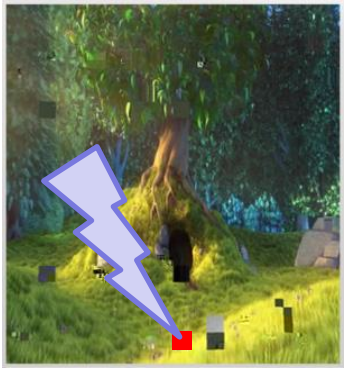


Flexible Error Correction

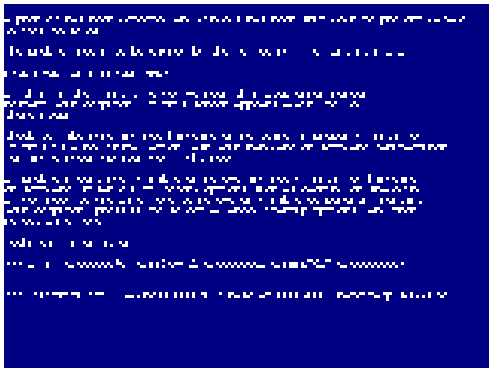
- In some cases, errors have no effect
- In some cases, timing requirements are dominating & error correction may be compromised
- In other cases, error correction requirements are dominating & timing requirements may be compromised



No effect



Imprecise output



System crash

- Classification requires application knowledge

Reliability Annotations

- Annotate impact of faults in the source code using type qualifiers
  - Errors cannot be tolerated

```
reliable   int r;
```

  - Errors will have non-fatal consequences

```
unreliable int u;
```

- Inhibit propagation of faults to reliable data
  - Unreliable data must not have any effect on reliable data

```
r = u; // invalid
u = r; // okay
```

- Use of type qualifiers can be checked by static analysis

Compile-time actions

Source code with annotations providing application knowledge

Application Knowledge:

- Impact of errors
- Feasible correction methods

Compiler

- Source code analysis

```
unreliable int j;
reliable   int control;
```

Propagation of annotations and inference of error classes

```
reliable int y = control;
```

- Encoding of classification

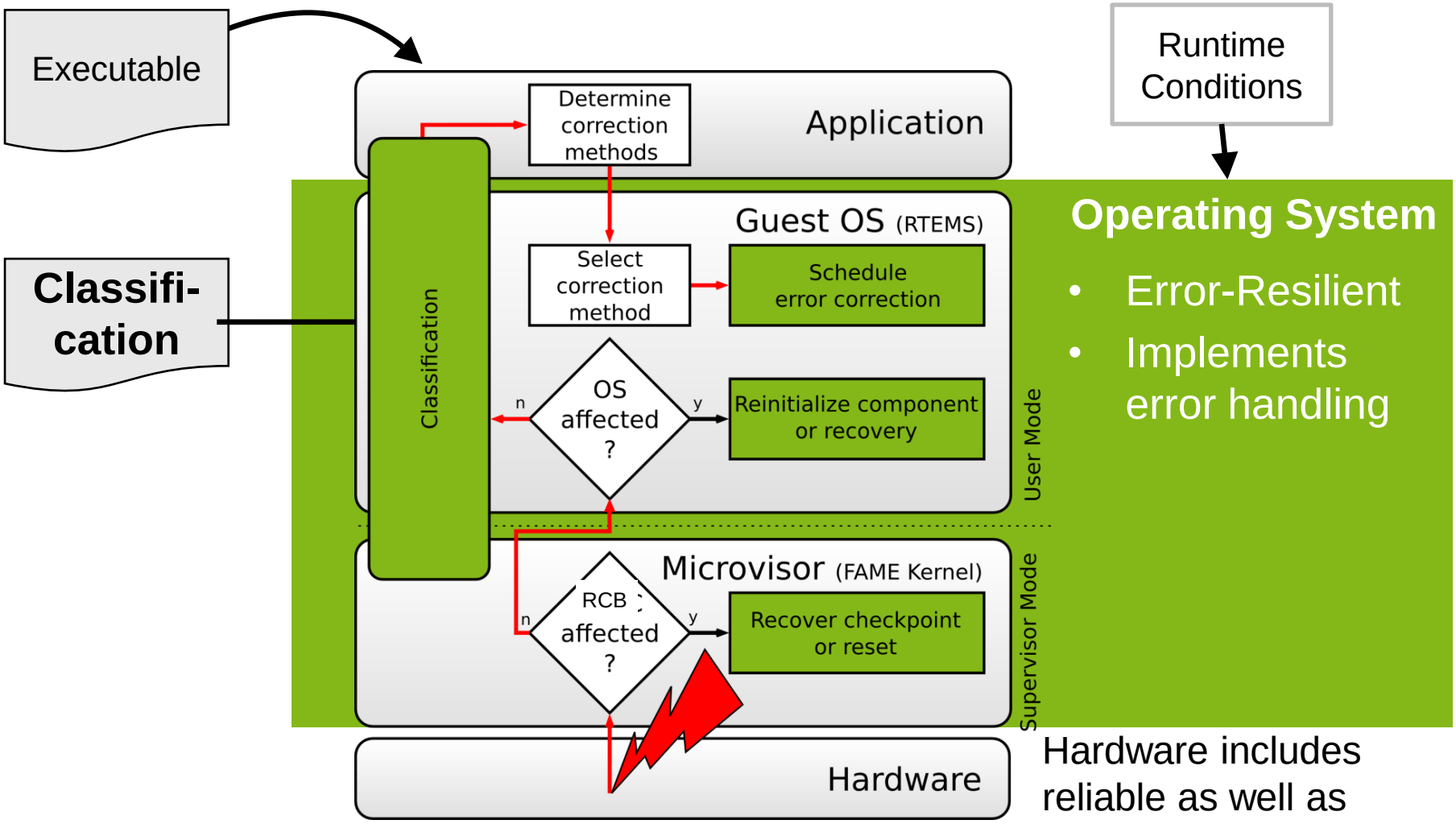
variables: x, y, control
address: 0xe01fd08c
size: 12 bytes
class: reliable
correction: recovery_2

0110110101110111
0001010111011001
1101110101101100
1101010100100101

Executable

Classifi-cation

Run-time actions



Hardware includes reliable as well as unreliable components

Results

- Added type qualifiers to C source code of JPEG decoder and constrained baseline H.264 decoder (> 3000 LOC)
- Fault injection experiment
  - No faults are injected into reliable data
  - 240 faults per second are injected into the memory ranges storing unreliable data

  ☞ No unintended terminations (crashes) if faults are injected only into the unreliable data objects

  ☞ Average PSNR of frames with errors: 40.89dB (still recognizable)

Results (cont.)

- Results for H.264 decoder

Video resolution	Memory size of reliable data	Memory size of unreliable data
176 x 144	91 kB (55%)	74 kB (45%)
352 x 288	223 kB (43%)	297 kB (57%)
1280 x 720	1,585 kB (37%)	2,700 kB (63%)

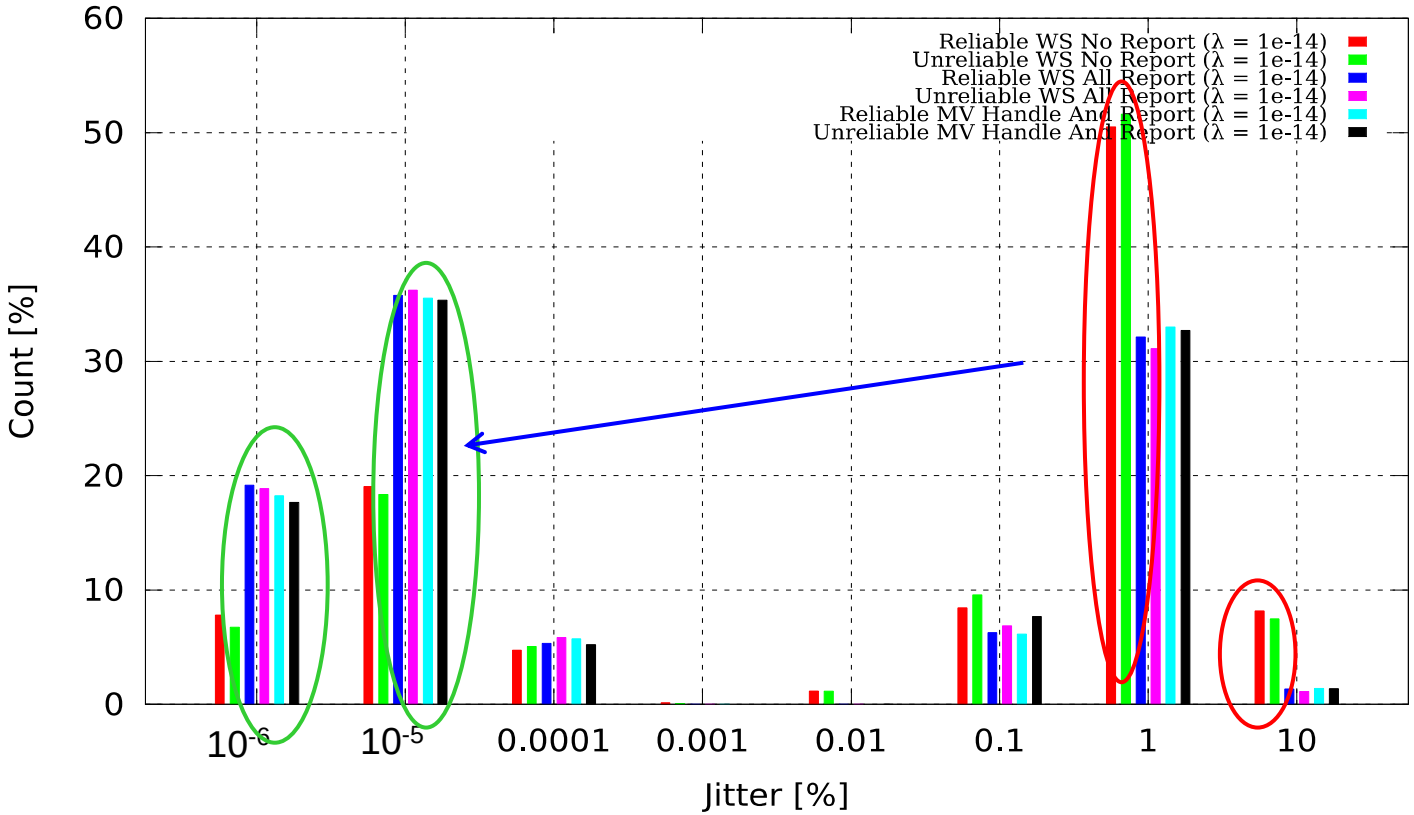
Impact of uncorrected errors on the quality of service:

Injection into unreliable memory	240 faults / sec	80 faults / sec
Average PSNR of frames with errors	40.89 dB	50.57 dB

Deteriorated reliability/QoS, timeliness maintained!

Reduction of jitter

Real-Time Impact $(11 fps)$



Flexible error handling allows significant jitter reduction even at high fault injection rates

Schmoll et al.: Improving the Fault Resilience of an H.264 Decoder using Static Analysis Methods. ACM TECS, 2013
Heinig et al.: Classification-based Improvement of App. Robustness and QoS in Probabilistic Computer Systems, ARCS'12
Heinig et al.: Using Application Knowledge to Improve Embedded Systems Dependability, HotDep 2011

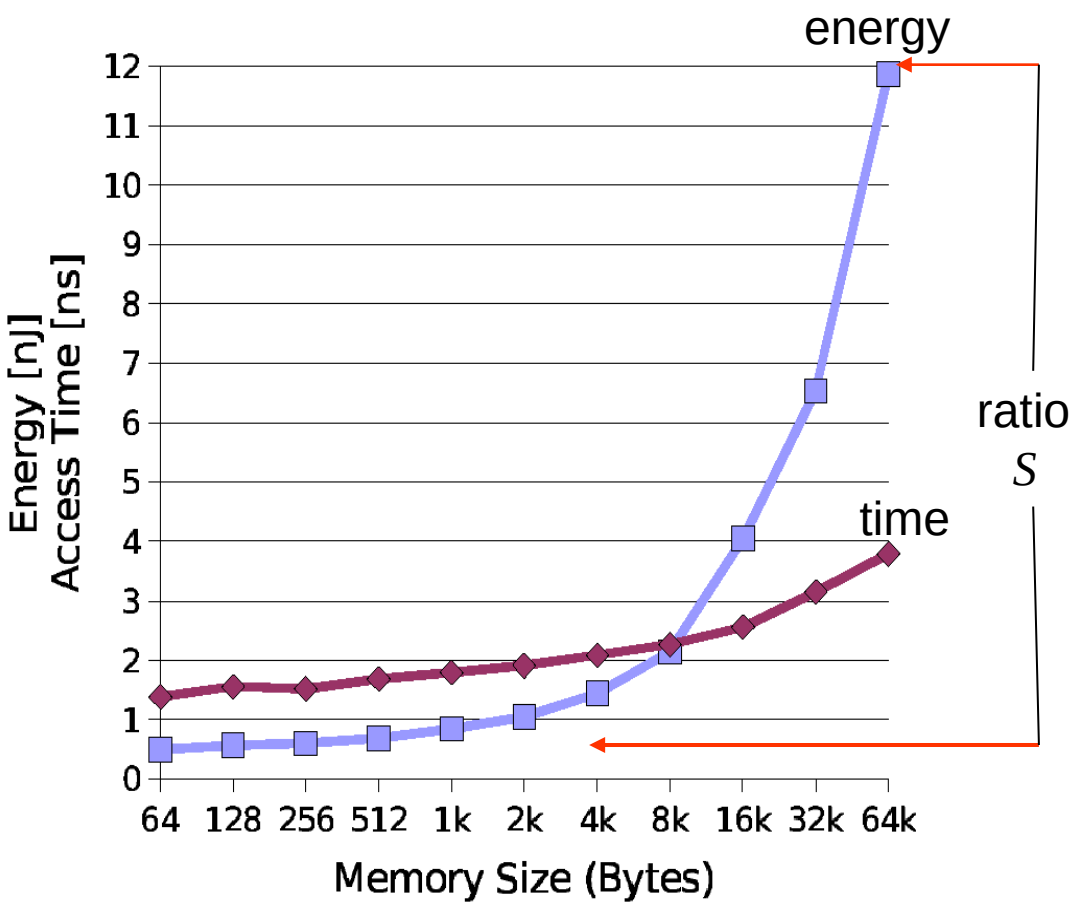
Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

Two metrics for the behavior of memories



Potential improvement is determined by the ratio S between objectives for large and small memories.

Can even be larger for larger memories

☞ **Memory hierarchies**

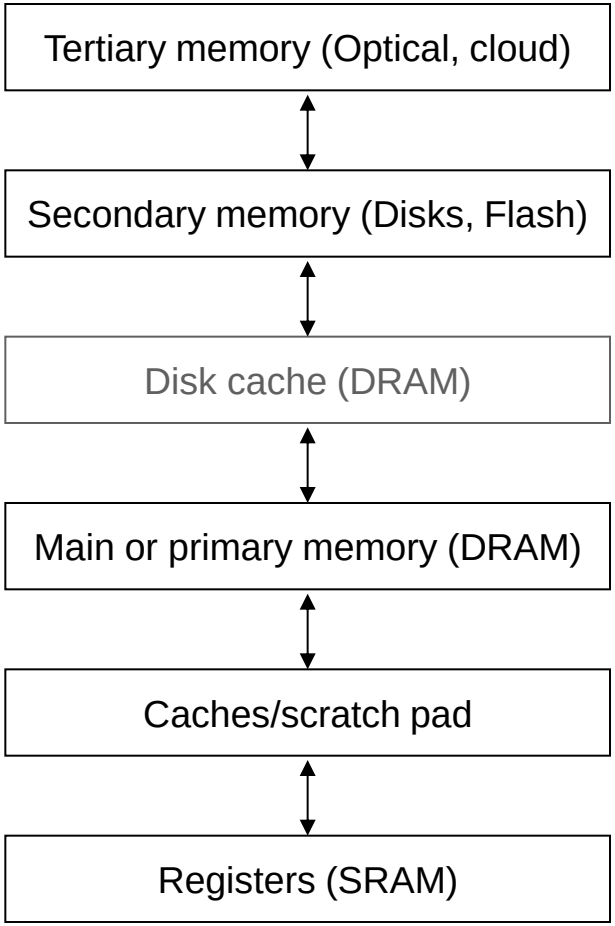
Introduction of memory hierarchies

"*Ideally one would desire an indefinitely large memory capacity such that any particular ... word ... would be immediately available - i.e. in a time which is ... shorter than the operation time of a fast electronic multiplier. ...*

It does not seem possible physically to achieve such a capacity.

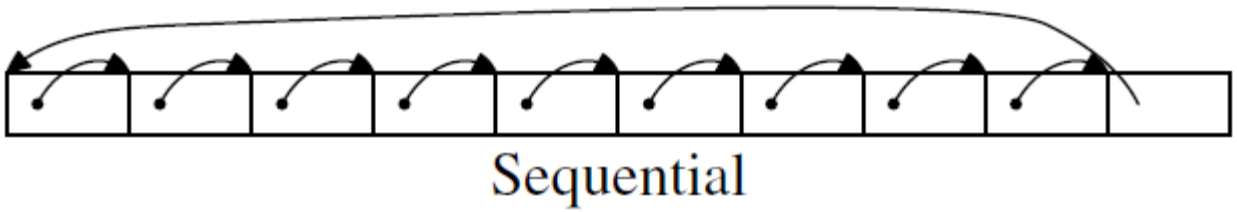
We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

A.W. Burks, H.H. Goldstine, and J. von Neumann: Preliminary Discussion of the Logical Design of an Electronic Computing Element, 1946



Performance impact of caches

- Access time for elements of size NPAD*8 Byte in a linked list, stored in an array

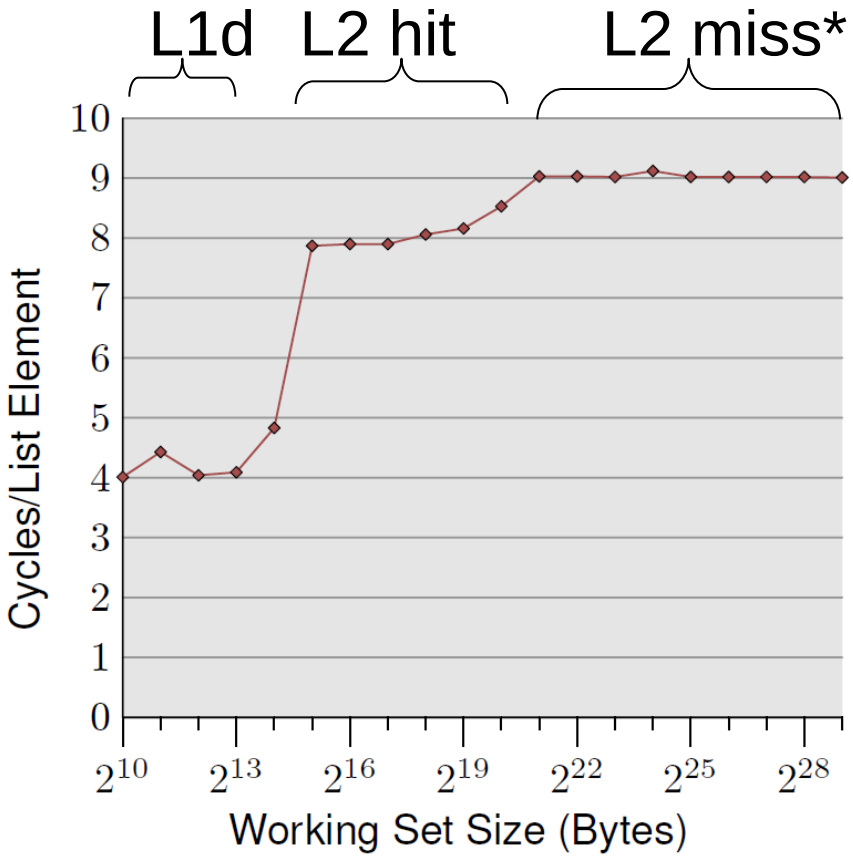


Configuration

- Pentium P4
- 16 kB L1 data cache, 4 cycles/access
- 1 MB L2 cache, 14 cycles/access
- main memory, 200 cycles/access

U. Drepper: *What every programmer should know about memory**, 2007, http://www.akkadia.org/drepper/cpumemory.pdf; thanks to J. Teubner for pointing to this source. * stimulated by David Goldberg, *What every programmer should know about floating point arithmetic, ACM Computing Surveys*, 1991

Cycles/access as a function of the size of the list



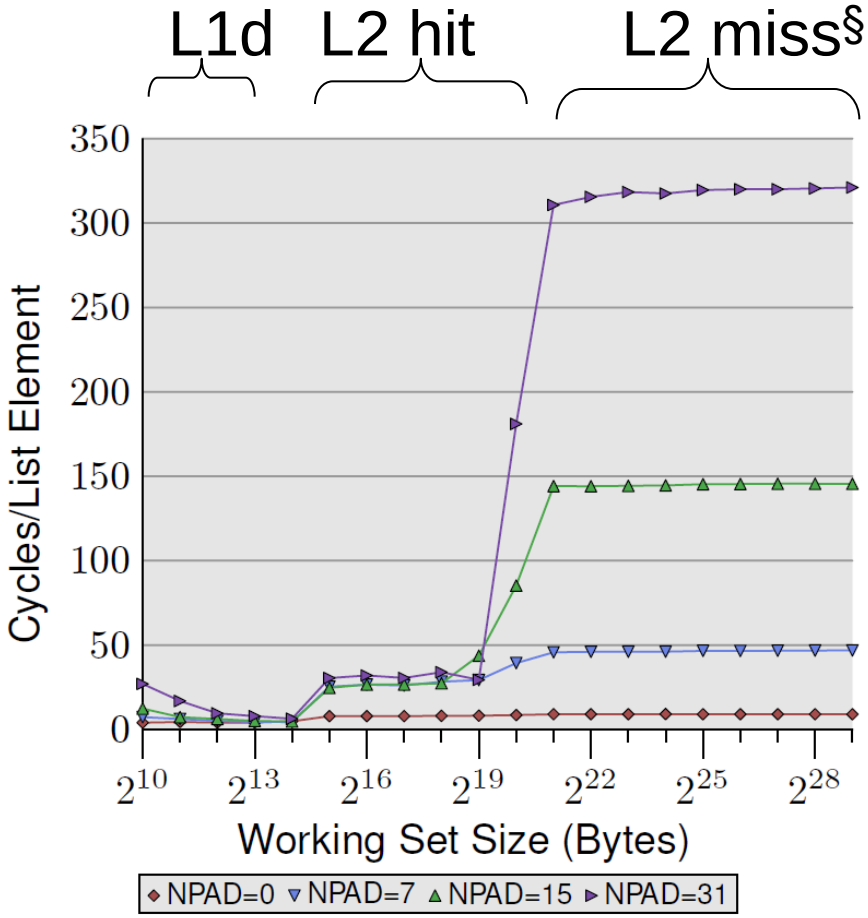


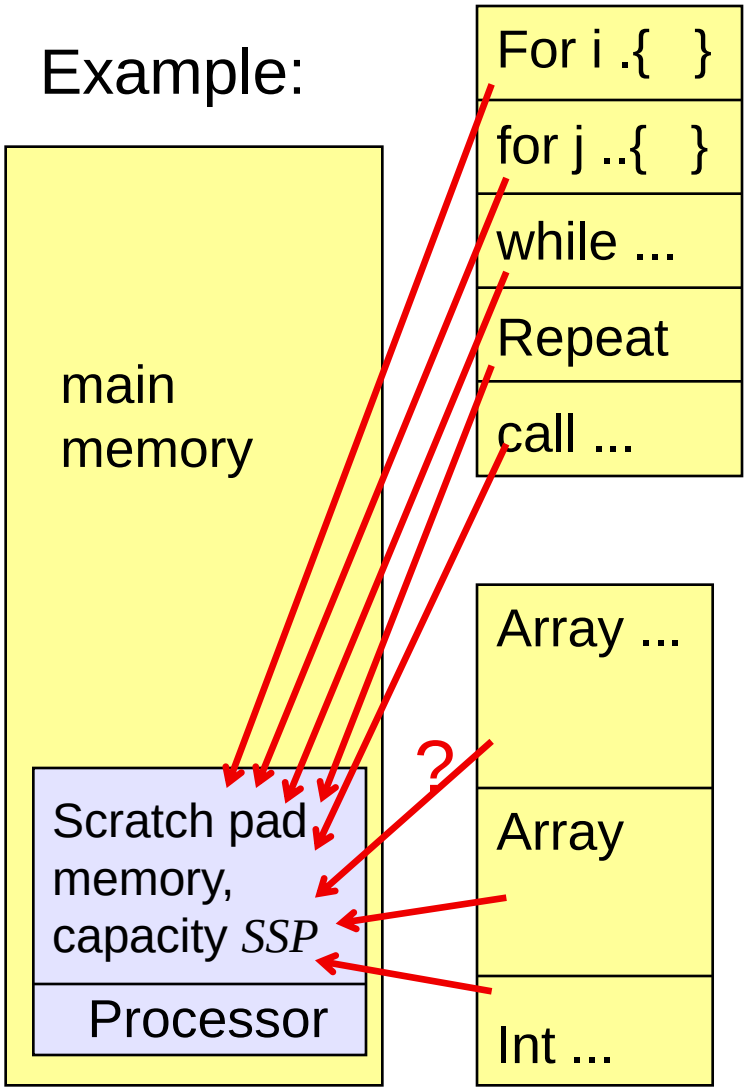
Figure 3.10: Sequential Read Access, NPAD=0

* prefetching succeeds

§ prefetching fails

Migration of data & instructions, global optimization model (TU Dortmund)

Example:



Which memory object (array, loop, etc.) to be stored in SPM?

Non-overlaying ("Static") allocation:

Gain g_k and size s_k for each object k. Maximize gain $G = \Sigma g_k$, respecting size of SPM $SSP \geq \Sigma\, s_k$.

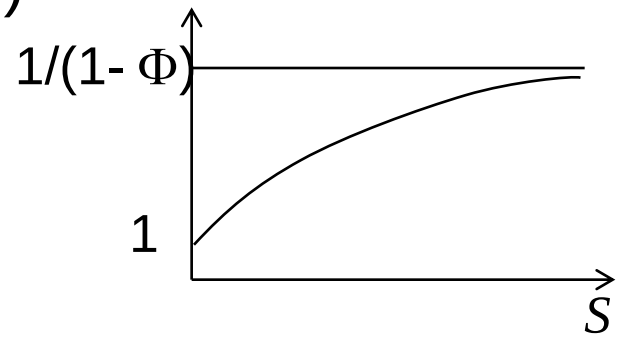
Solution: Knapsack algorithm.

Overlaying ("dynamic") allocation:

Moving objects back and forth

How much better can we get?

$$improvement = \frac{1}{(1-\Phi)+\frac{\Phi}{S}}$$ (Amdahl's law) *



where

- Φ: fraction of memory references replaced by faster/more energy efficient memory

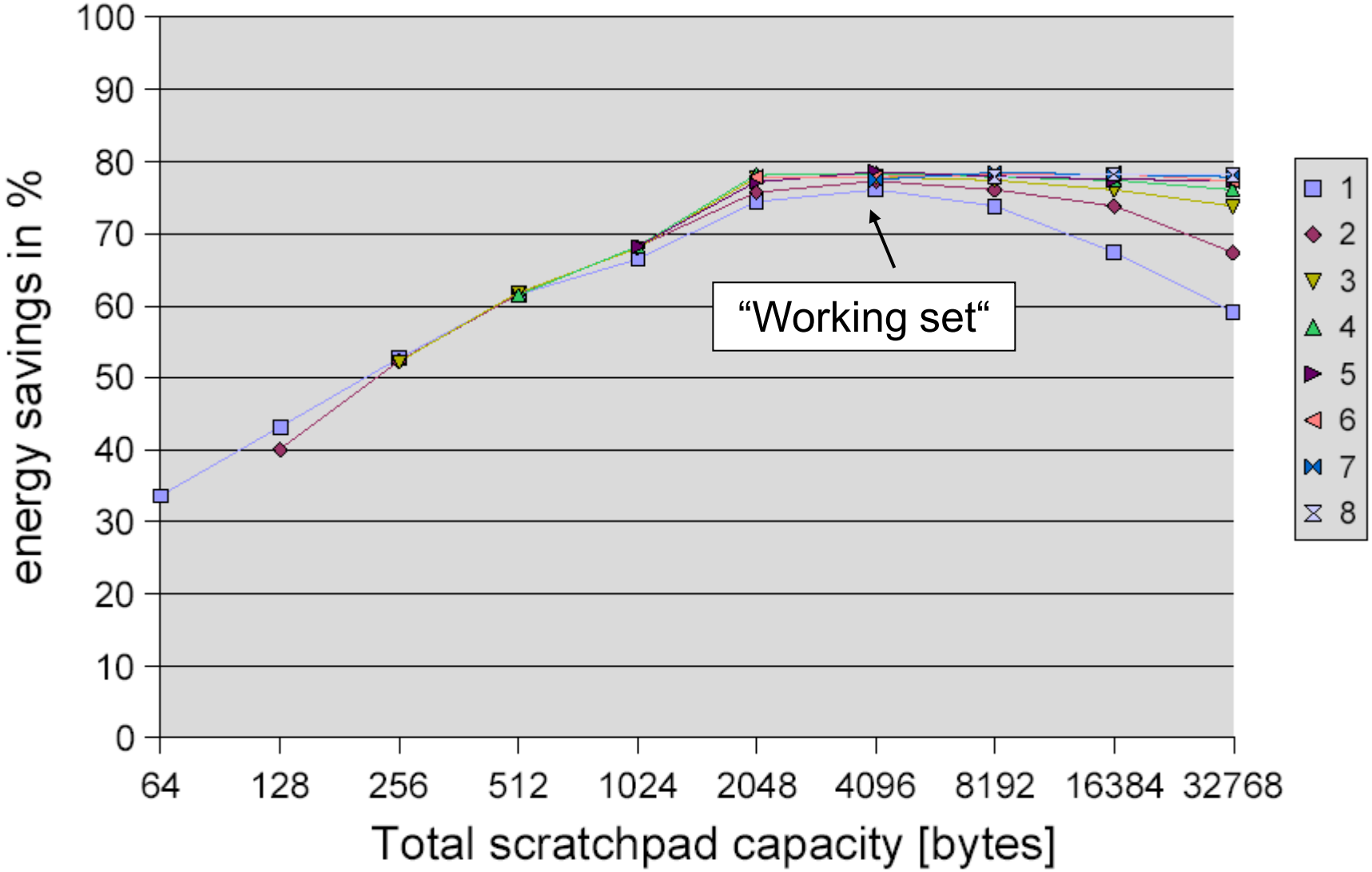
and

- S: speed/energy improvement

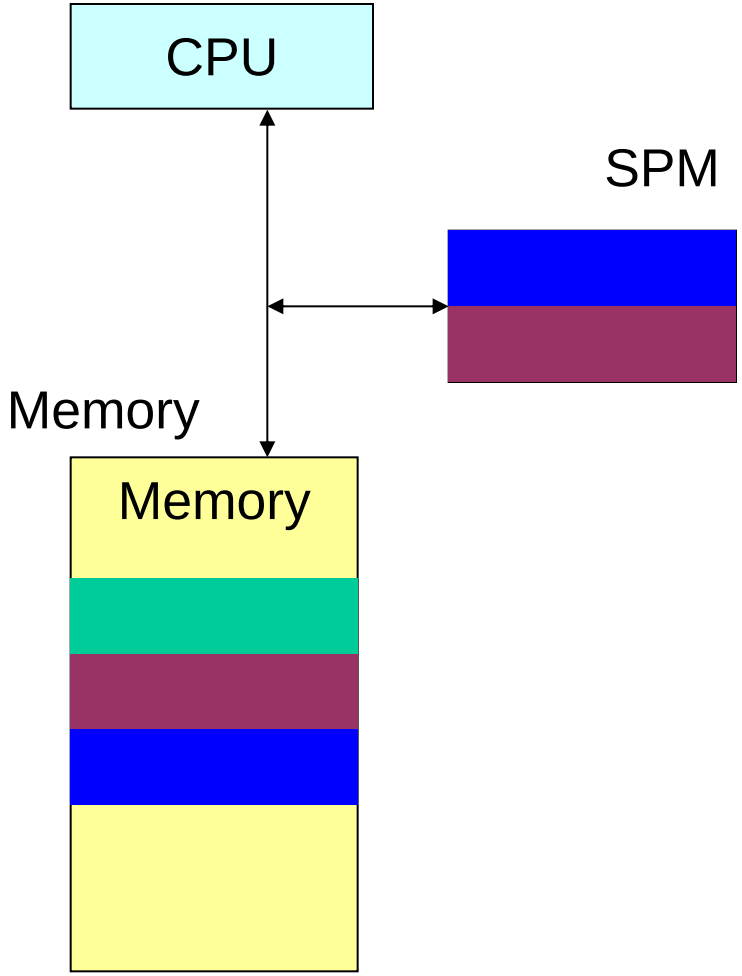
Important not to have many "untouchable" references (1- Φ), otherwise even $S \rightarrow \infty$ does not help

* Replacing commonly used P by Φ to avoid confusion with power

Results for parts of GSM coder/decoder

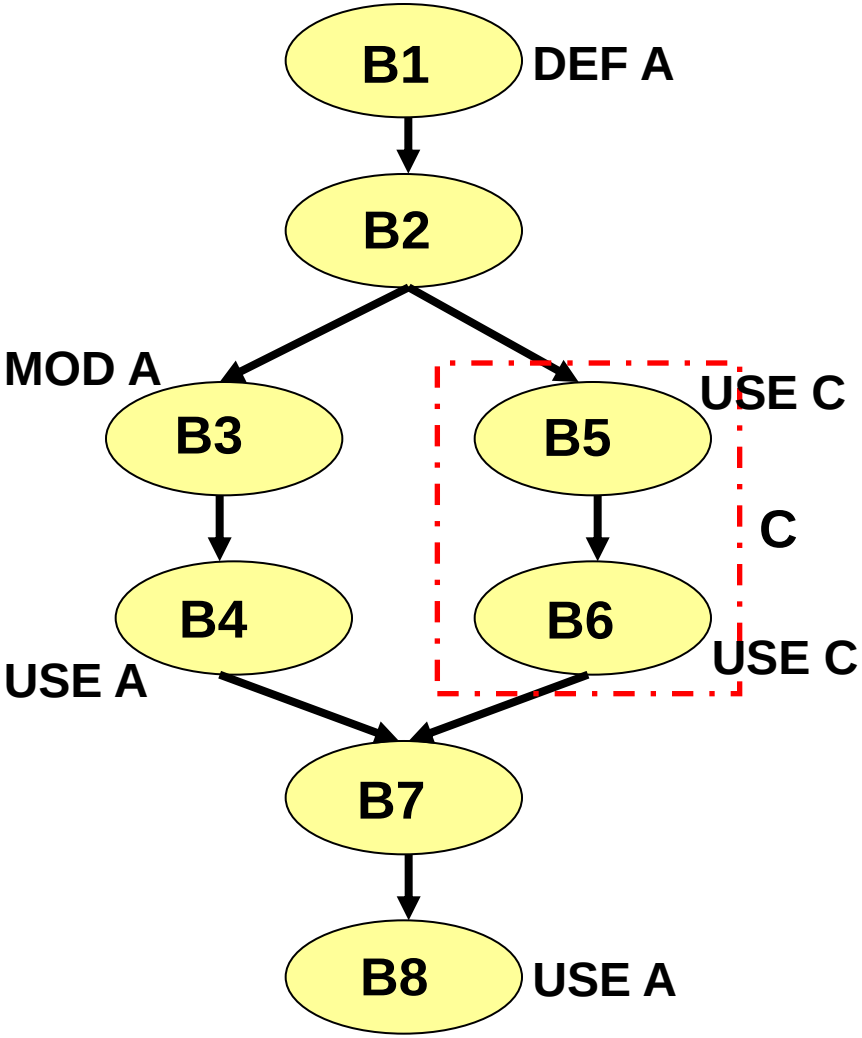


Non-overlaying allocation problematic for multiple hot spots ☞Overlaying allocation



- Effectively results in a kind of **compiler-controlled overlays** for SPM
- Address assignment within SPM required

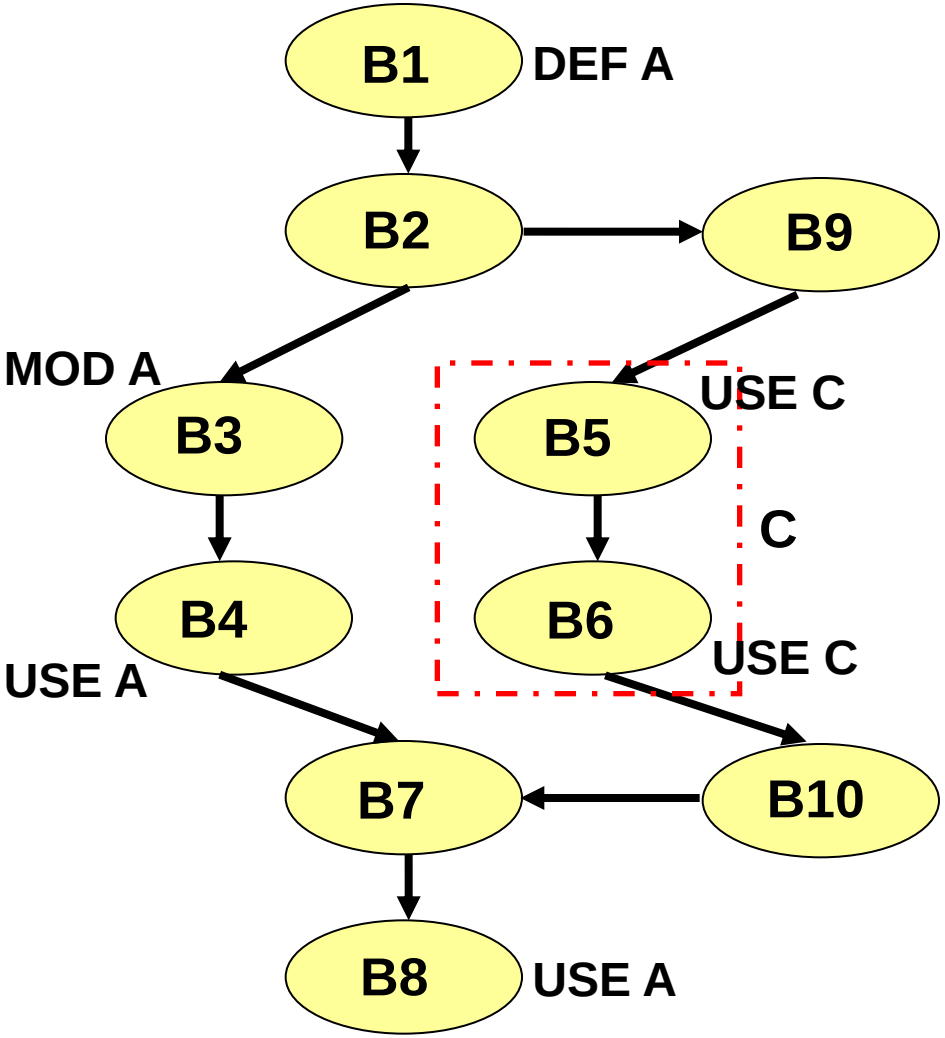
Overlaying allocation by Verma et al. (1)



Based on control flow graph.

M.Verma, P.Marwedel: Dynamic Overlay of Scratchpad Memory for Energy Minimization, *ISSS,* 2004

Overlaying allocation by Verma et al. (2)



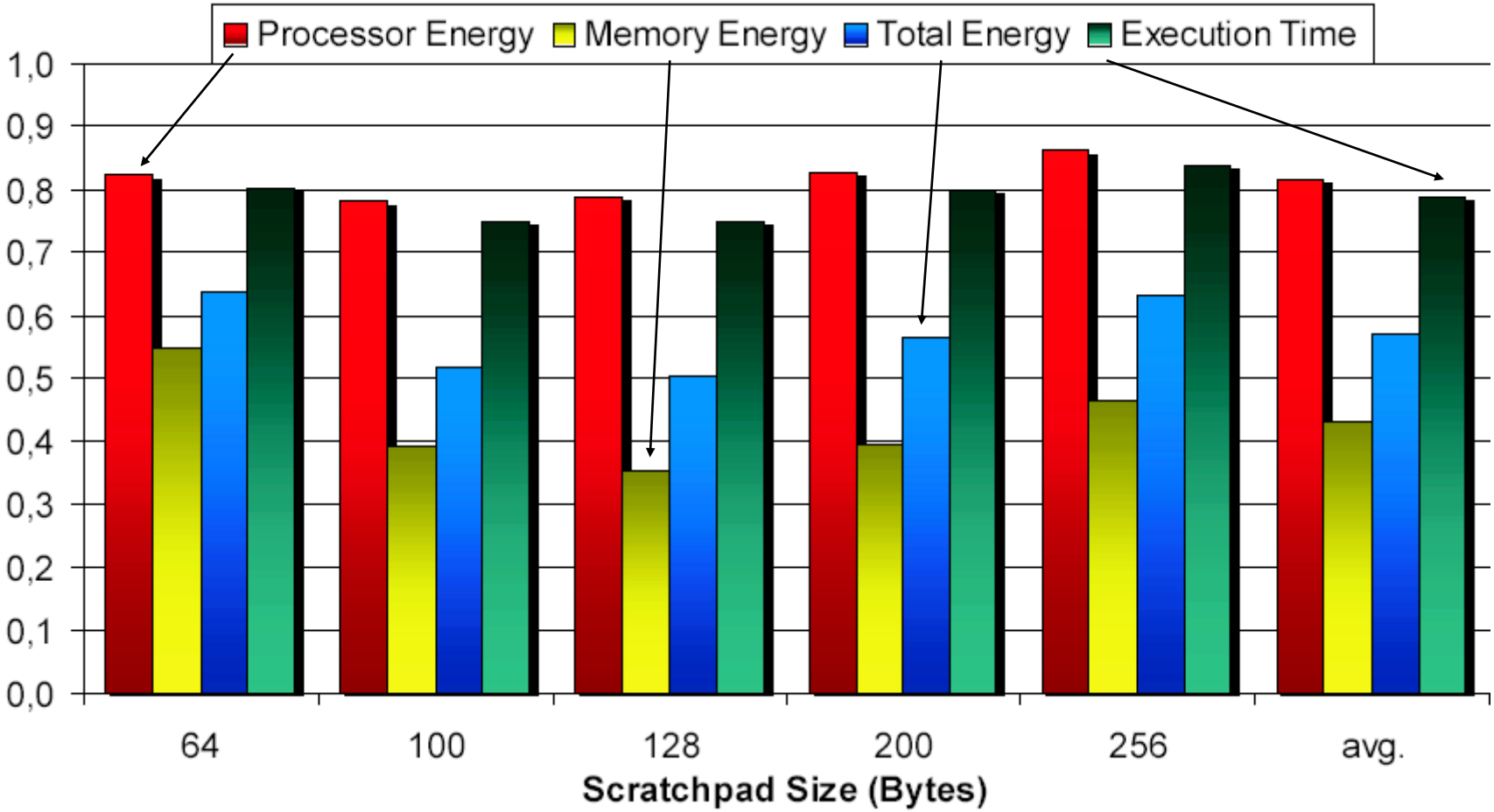

```
SPILL_STORE(A);
SPILL_LOAD(C);
```

Global set of ILP equations reflects cost/benefit relations of potential copy points

```
SPILL_LOAD(A);
```

Code handled like data

Runtime/energy reduction with respect to non-overlaying ("static") allocation



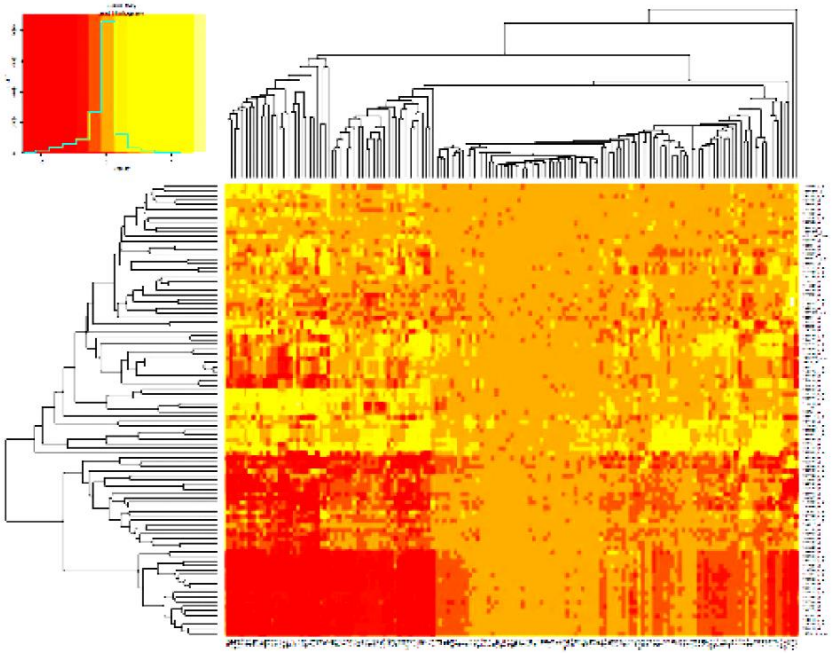
Less seriously …

Content not shown in public version of the slides

☞ Some people got already completely rid of cache

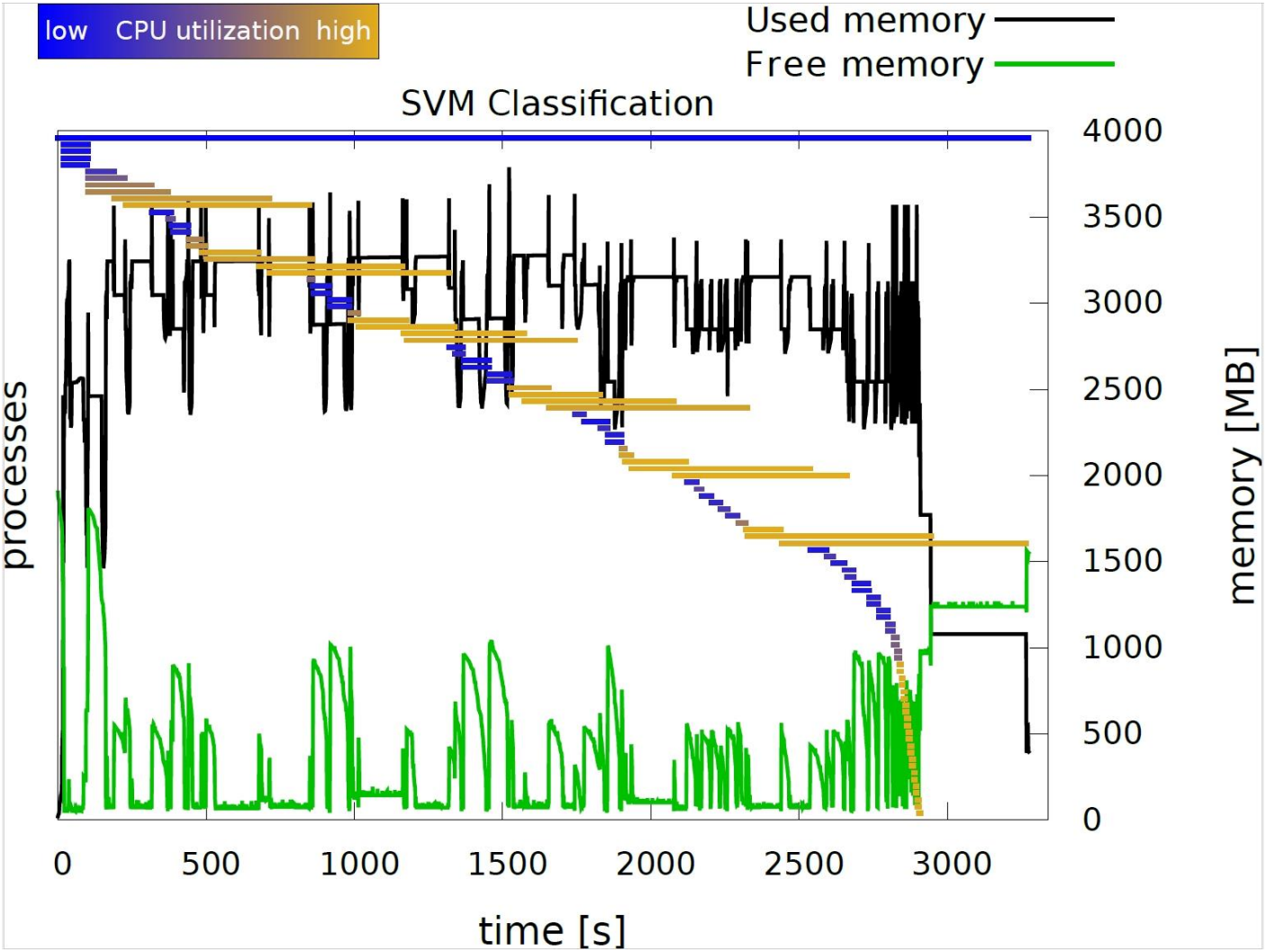
Resource aware data analysis

- Finding impact of treatment for class of patients
- Finding the best model to predict survival rates of cancer patients
- Requires a huge amount of model evaluations
- Can be mapped to multi-cores
- However: how many threads should be used?



Helena Kotthaus, Ingo Korb, Peter Marwedel: Performance Analysis for Parallel R Programs: Towards Efficient Resource Utilization, Technical Report 1/2015, TU Dortmund, CS Department

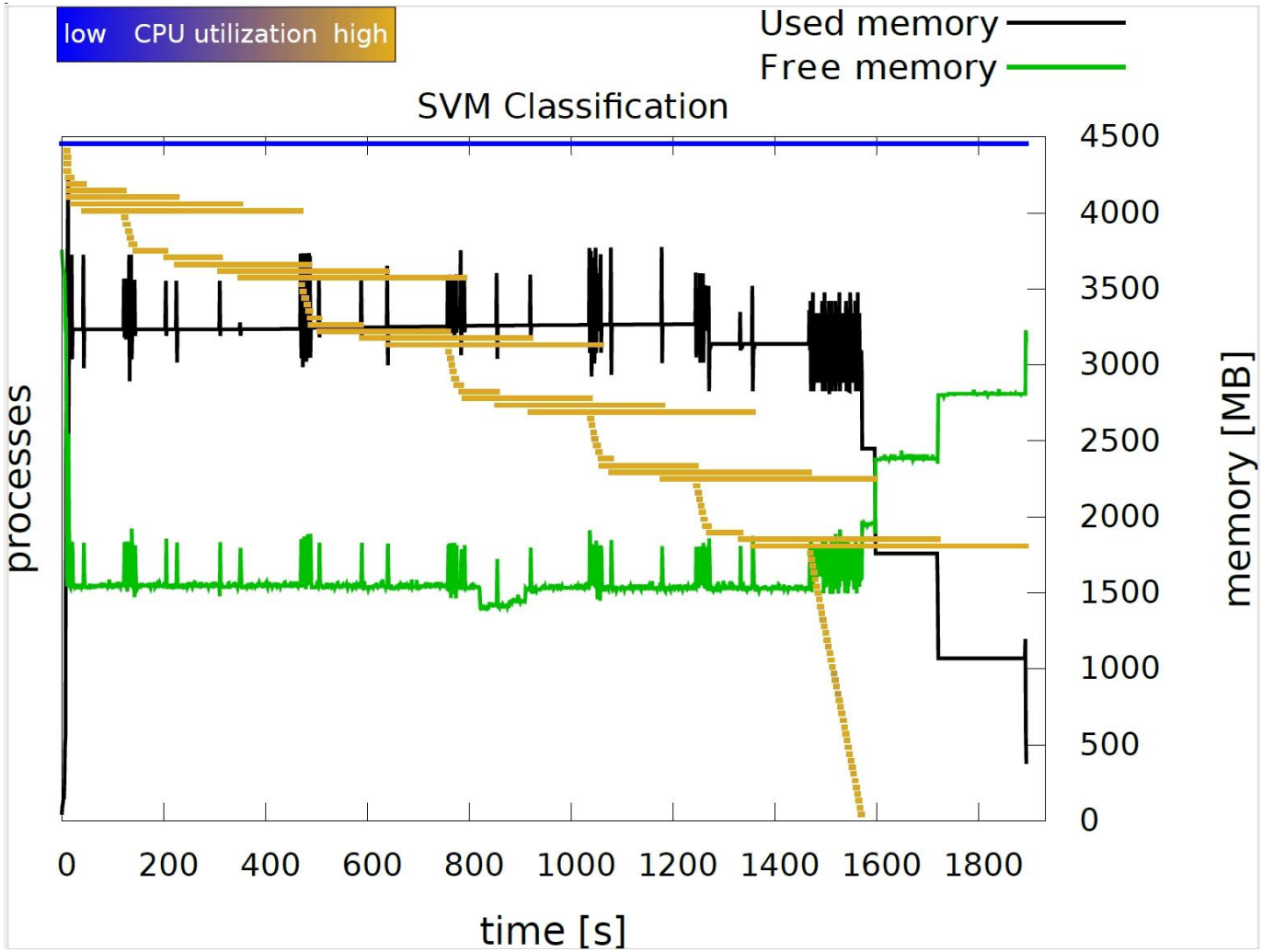
4 cores, no load balancing



Memory congestion ☞ low CPU utilizations

Helena Kotthaus, Ingo Korb, Peter Marwedel: Performance Analysis for Parallel R Programs: Towards Efficient Resource Utilization, Technical Report 1/2015, TU Dortmund, CS Department

4 cores, with load balancing



No memory congestion ☞ high CPU utilizations

Helena Kotthaus, Ingo Korb, Peter Marwedel: Performance Analysis for Parallel R Programs: Towards Efficient Resource Utilization, Technical Report 1/2015, TU Dortmund, CS Department

Outline

- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results

CS, EE or physics concentration vs. separate CPS/ES program

	Specialization	Separate program
Well-known degrees	+	-
Enough headroom for teaching integrated CPS/ES material	-	+
Headroom for more depth in physics, mechanical engineering, ..	-	+
Effort for introduction	small	large
No. of faculty members required	moderate	larger
Building community?	difficult	easier
Is it feasible?	For ES ok, for CPS questionable	Yes, but there are also constraints

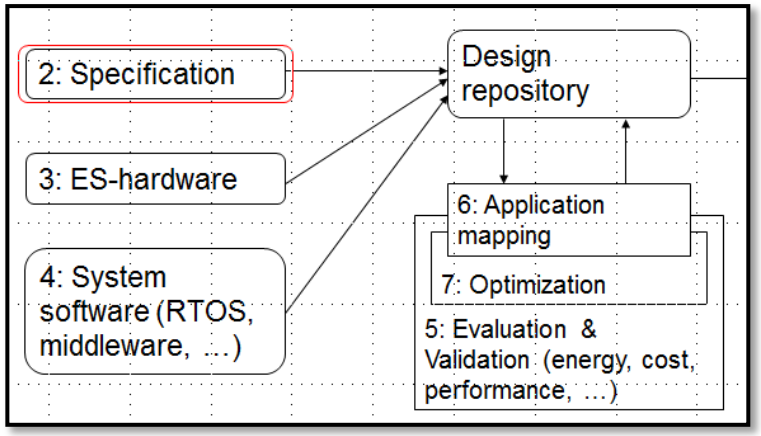
ES fundamentals of CPS selected

Books on fundamentals:

- 1st edition
  - English
    - Original hardcover version
    - Reprint, soft cover, 2006
  - German, 2007
  - Chinese, 2006
  - Macedonian, 2010
- 2nd edition, with CPS
  - English, Dec. 2010/Jan. 2011
  - Translated Chinese edition, 2013
  - Contract for German edition

Mature course
"Cyber-physical system fundamentals"

1. Introduction (applications, challenges, design flows)
2. Specification and modeling (models of computation, …)
3. Embedded system hardware (HW loop, converters, proc.s)
4. System software (real-time OS, priority inversion)
5. Evaluation & validation (>1 objective & Pareto optimality)
6. Application mapping (scheduling)
7. Optimizations
8. Test



Slides and videos available

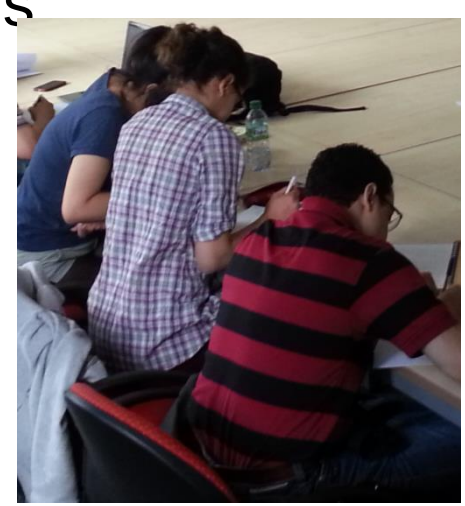
2nd generation videos are available at
http://www.youtube.com/user/cyphysystems

Value of lectures?

- Lectures: value compared to videos **if no interaction**?
- Lectures communicate facts, they are not training skills.
- *"Lectures aren't just boring, They're Ineffective, Too, Study Finds*"
  http://news.sciencemag.org/education/2014/05/lectures-arent-just-boring-theyre-ineffective-too-study-finds
- Use presence of students to their advantage! (≠ MOOCs)

Teaching style @ CPSF: flipped classroom

- Serious procrastination due to video availability
- ☞ Home work and work in the classroom are flipped:
  - Students watch videos at home
  - Physical presence is used to work on work sheets
- Interactive learning
- Training for teamwork & communication skills
- Taking advantage of student's presence
- Lab sessions focusing on practical training
- Re-discovery of the usefulness of books
- Early feedback for educator
- Higher quality of recorded videos

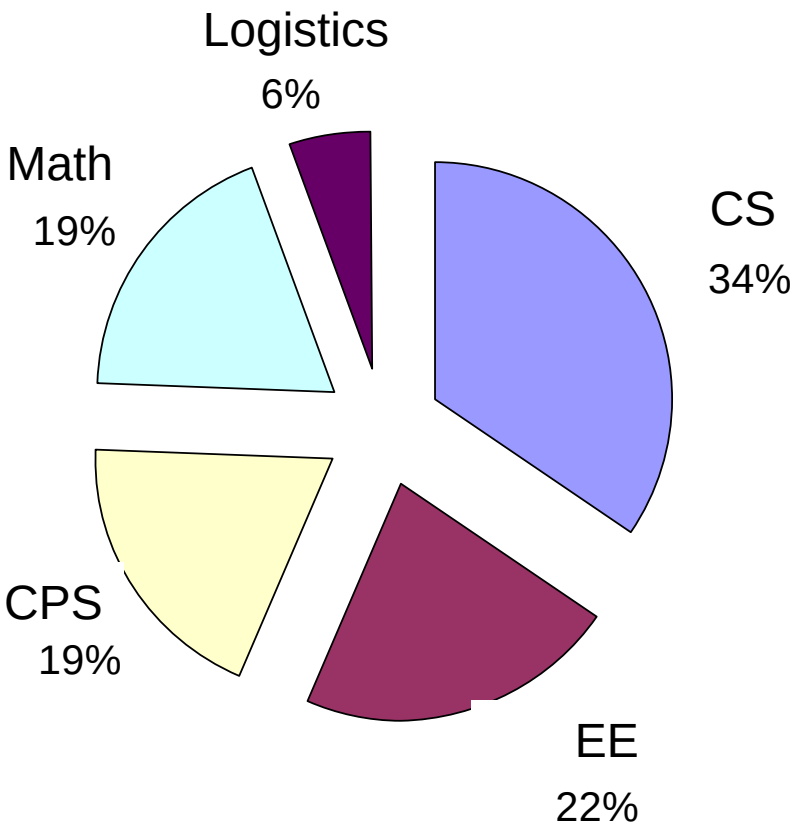


Related work (not so serious)

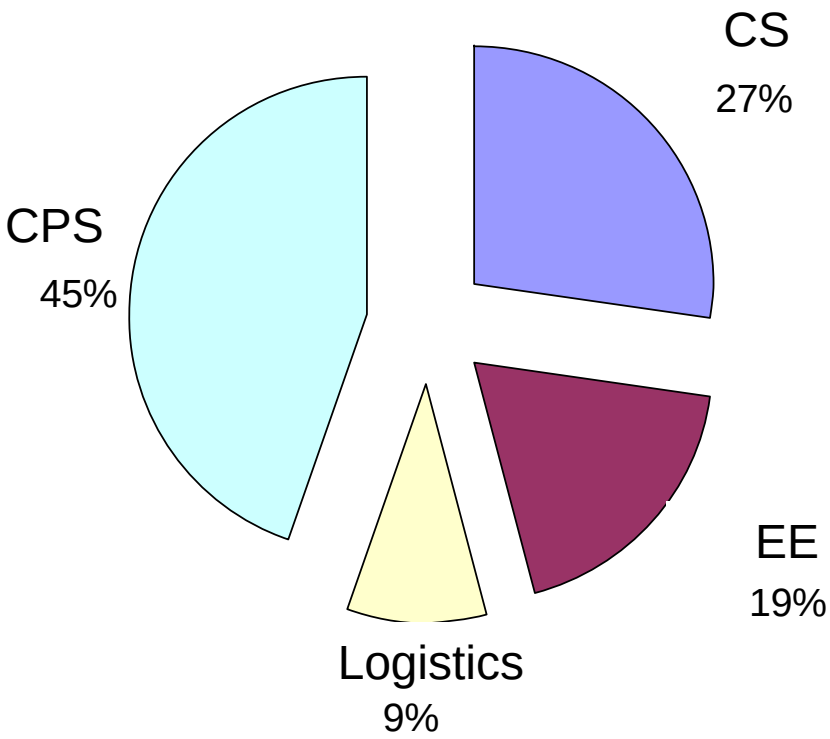
Content not shown in public version of the slides

Designed Integrated CPS program

Undergraduate level

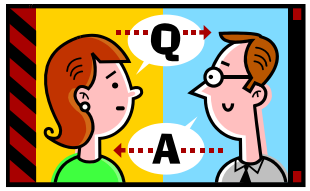


Graduate level



Summary

Q&A?



- Scope & definitions
- Opportunities & examples
- Challenges
  - Security, timing, energy, heat, reliability, control, .., specs
- (Some) solutions
  - Modeling techniques (Intro)
  - Evaluation techniques (WCET→RTC→E→T→MTTF)
  - Energy efficient virus detection with GPUs
  - Worst-case execution time aware compilation
  - Flexible error handling to maintain timeliness
  - Efficient memory accesses (SPM, thrashing)
  - Education in ES/CPS
  - Summary

Standard techniques

Own results