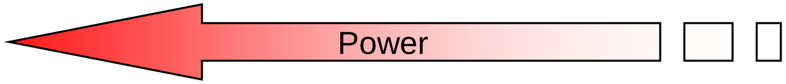
Energy-efficient embedded computing

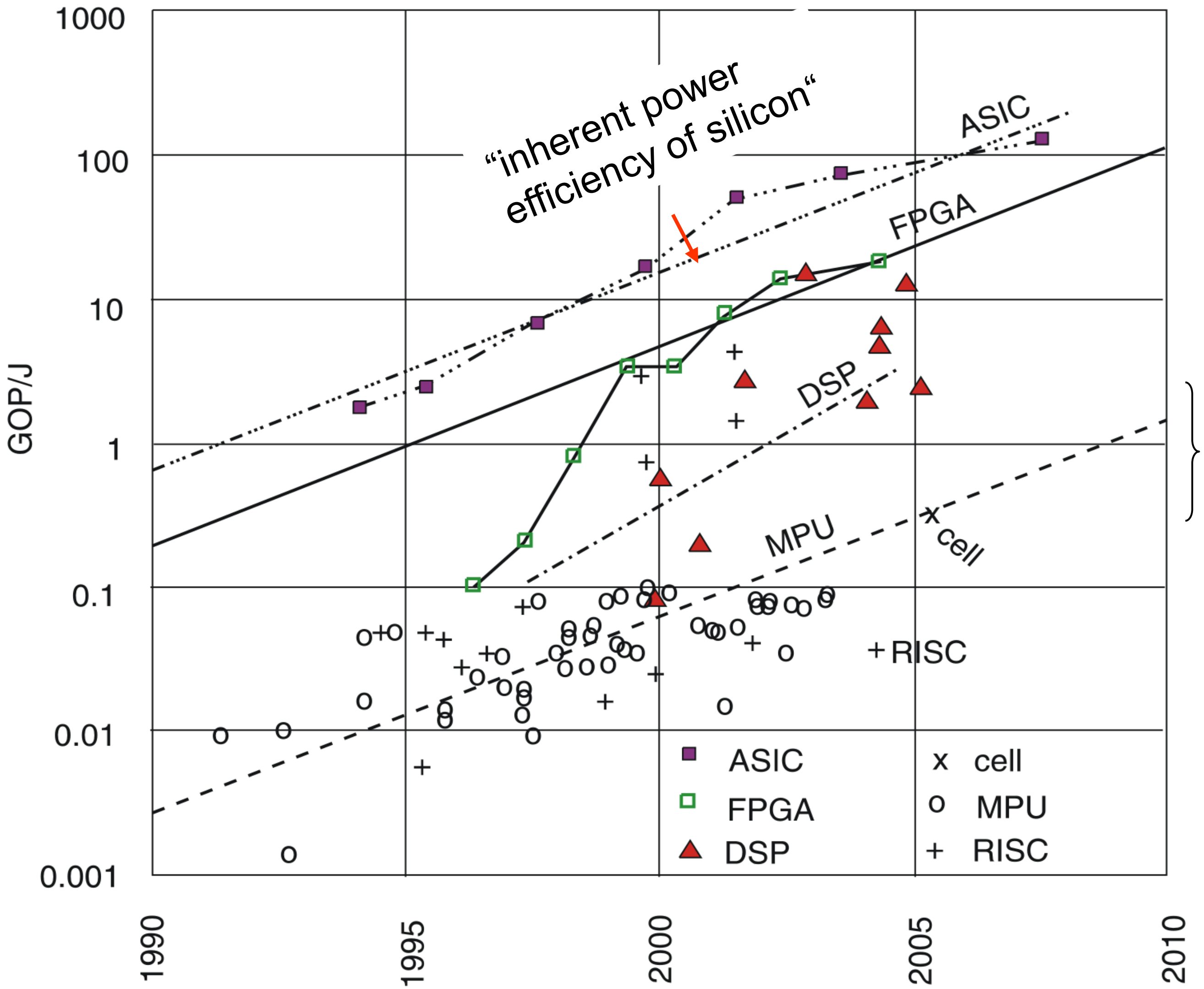
Peter Marwedel
TU Dortmund, Informatik 12
Germany

Why energy-efficient computing?

	Relevant during use?		
Execution platform	Plugged	Potentially unplugged	Unplug-ged/em-bedded
E.g.	Server/ desktop	laptop	Phone
Global warming	☑	☐	☐
Cost of energy	☑	☐	☐
Increasing performance	☑	☑	☑
Avoiding high currents & metal migration	☑	☑	☑
Problems with cooling, avoiding heat	☑	☑	☑
Reliability	☑	☑	☑
Energy a very scarce resource	☐	☑	☑



Energy consumption trends



Unplugged embedded computing (or high-performance plugged)

Computing, plugged

© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, *Keynote Slides*, ACACES, 2007

Distribution of energy consumption: Computing, plugged

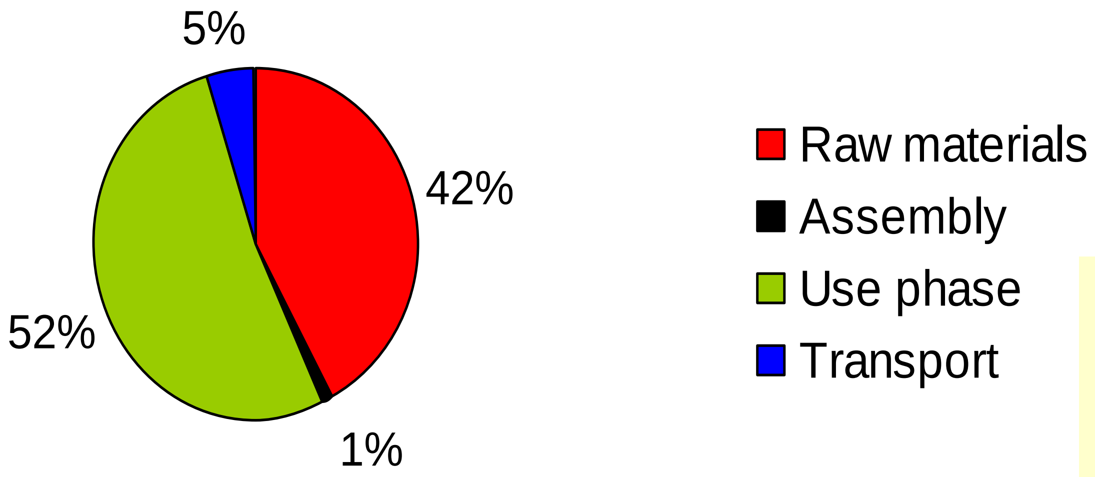
- Gartner: 2% of CO_2-Emissions are due to IT operation,
  - with 23% caused by computing centers and
  - 40% due to PC's and displays
  - Energy is also **saved** due to IT operation!

reduction by "green computing"

- However, **production** is responsible for many emissions

  E.g.: Carbon Footprint ESPRIMO E9900 desktop

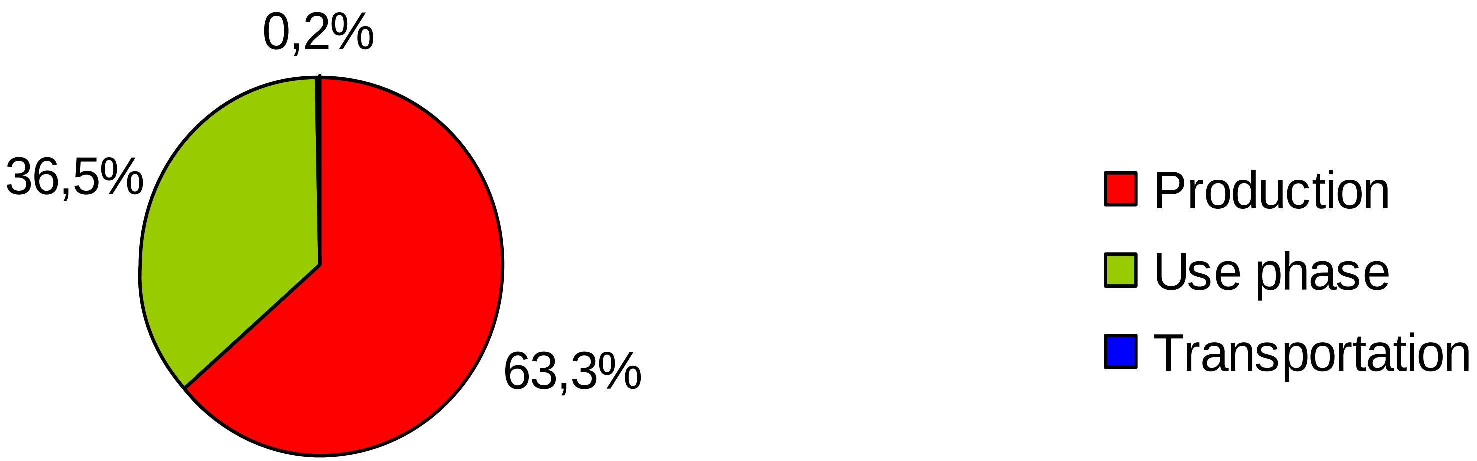
reduction by "green production"



Fujitsu: Life Cycle Assessment and Product Carbon Footprint – Fujitsu ESPRIMO E9900 Desktop PC, 2010, http://fujitsu.fleishmaneurope.de/?attachment_id=2148

Distribution of energy consumption: Computing, plugged (2)

Desktop PC + CRT-Monitor

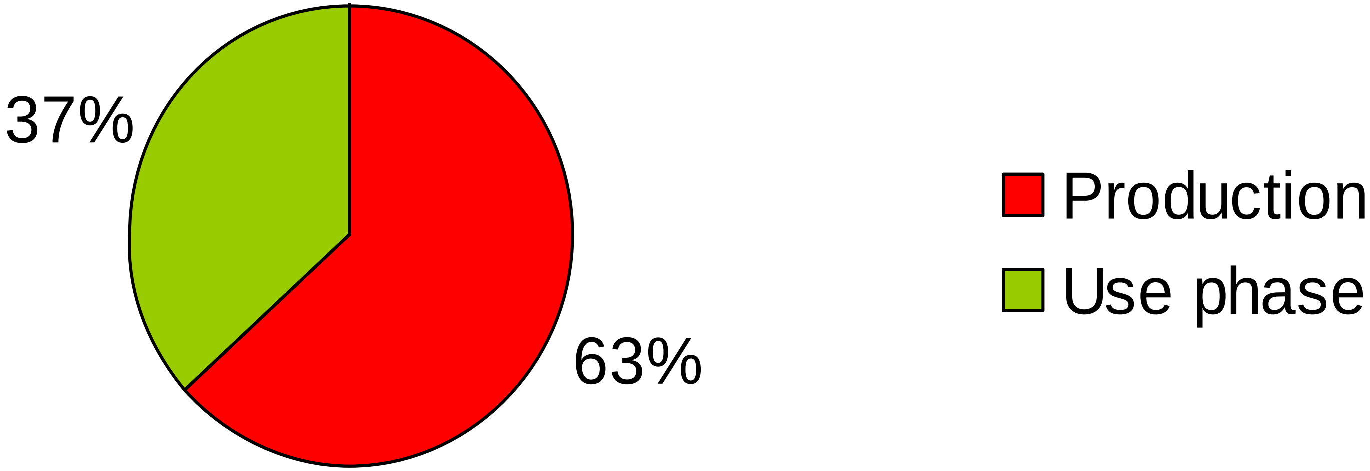


- No networking considered
- No disposal considered
- Larger share of use phase with a/c
- Smaller share of use phase in cold countries

P. Marwedel, M. Engel, Plea for a Holistic Analysis of the Relationship between Information Technology and Carbon-Dioxide Emissions, ARCS Workshop on Energy-aware Systems and Methods, Hannover, 2010; using data from R. Kühr und E. Williams: Computer and the Environment – Understanding and Managing their Impacts, Kluwer, 2003

Distribution of energy consumption: Computing, potentially unplugged

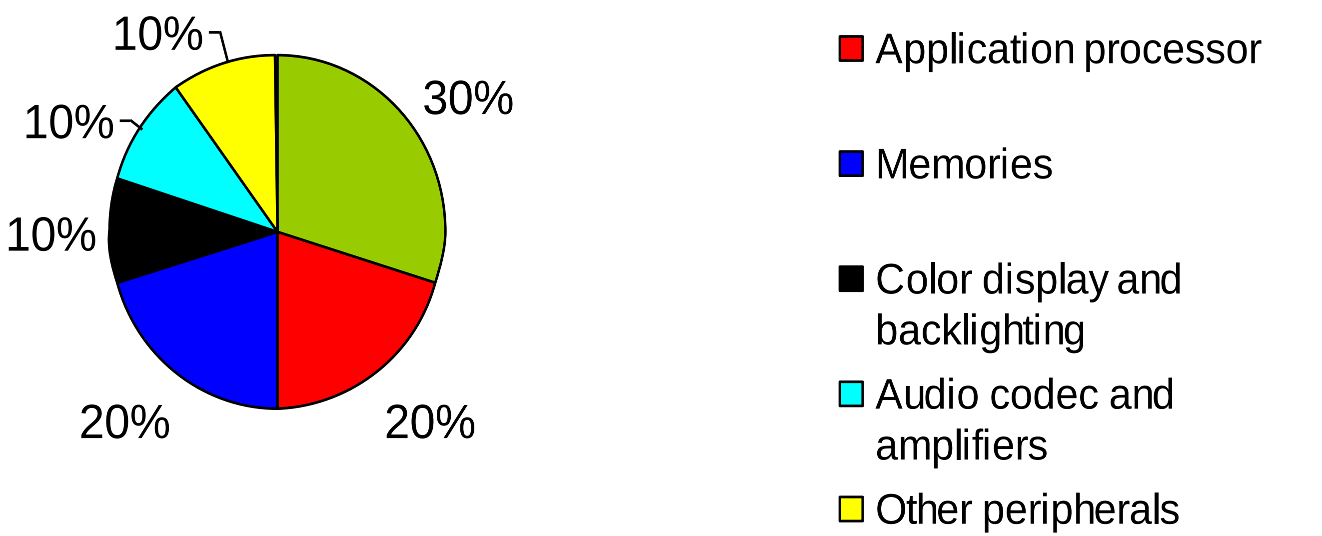
Dell Inspiron 2500 laptop (2001)



Liqiu Deng, Callie W. Babbitt and Eric D. Williams: Economic-balance hybrid LCA extended with uncertainty analysis: case study of a laptop computer, *Journal of Cleaner Production*, Vol. 19, Issue 11, Juli 2011, S. 1198-1206

Distribution of energy consumption: Computing, unplugged

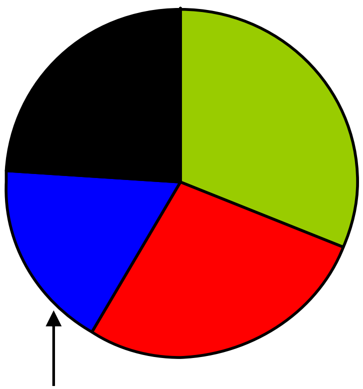
Mobile phone use



O. Vargas (Infineon Technologies): Minimum power consumption in mobile-phone memory subsystems; Pennwell Portable Design - September 2005

Distribution of energy consumption: Computing, unplugged (2)

Mobile phone use, breakdown by type of computation



C.H. van Berkel: Multi-Core for Mobile Phones, DATE, 2009; (no explicit percentages in original paper)

- ☞ Consider **production** and **use** of computing devices!
- ☞ During use, all components & computations relevant
- ☞ Optimizations need energy models

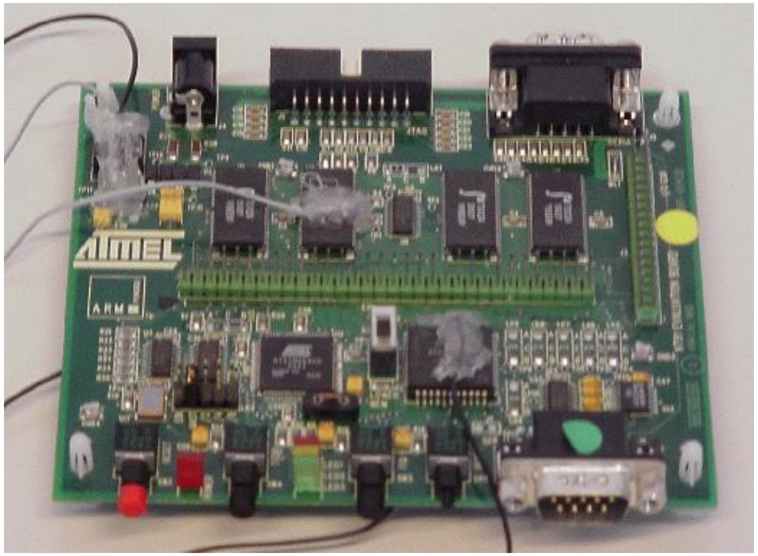
Outline

- Motivation
  - Where is energy converted into heat?
- Energy models (during use phase)
- Energy optimization (during use phase)
  - Standard optimizations
  - Exploitation of scratch pad memories in embedded computing
  - Memory-architecture aware design framework
  - Energy-efficiency of graphics processors
- Future work
- Summary

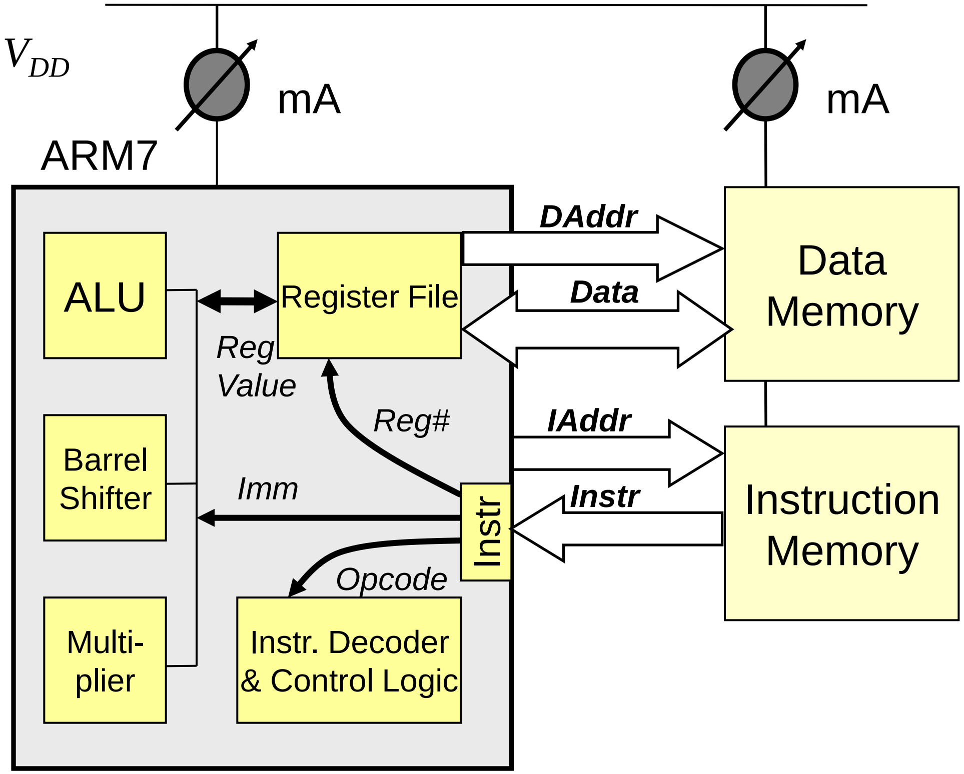
Energy models

- Measurements: (potentially) precise, fixed architecture
  - Tiwari (1994): Energy consumption within processors
  - Russell, Jacome (1998): Measurements for 2 fixed configurations
  - Simunic (1999): Using values from data sheets. Allows modeling of all components, but not very precise.
- Models: flexible architecture, less precise
  - CACTI [Jouppi, 1996]: Predicted energy consumption of caches
  - Wattch [Brooks, 2000]: Power estimation at the architectural level, without circuit or layout
- Combined models
  - Steinke et al., TU Dortmund (2001): mixed model using measurements and prediction

Steinke's model



E.g.: ATMEL board with ARM7TDMI and ext. SRAM



$$E_{total} = E_{cpu\_instr} + E_{cpu\_data} + E_{mem\_instr} + E_{mem\_data}$$

S. Steinke, M. Knauer, L. Wehmeyer, P. Marwedel: An Accurate and Fine Grain Instruction-Level Energy Model Supporting Software Optimizations, Int. Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2001

Example:
Instruction dependent costs in the CPU

Cost for a sequence of m instructions

$$
\begin{aligned}
E_{cpu\_instr} = \sum MinCostCPU(\mathbf{Opcode}_i) + FUCost(\mathbf{Instr}_{i-1},\mathbf{Instr}_i) + \\
\alpha_1 * \sum w(\mathbf{Imm}_{i,j}) + ß_1 * \sum h(\mathbf{Imm}_{i-1,j}, \mathbf{Imm}_{i,j}) + \\
\alpha_2 * \sum w(\mathbf{Reg}_{i,k}) + ß_2 * \sum h(\mathbf{Reg}_{i-1,k}, \mathbf{Reg}_{i,k}) + \\
\alpha_3 * \sum w(\mathbf{RegVal}_{i,k}) + ß_3 * \sum h(\mathbf{RegVal}_{i-1,k}, \mathbf{RegVal}_{i,k}) + \\
\alpha_4 * \sum w(\mathbf{IAddr}_i) + ß_4 * \sum h(\mathbf{IAddr}_{i-1}, \mathbf{IAddr}_i)
\end{aligned}
$$

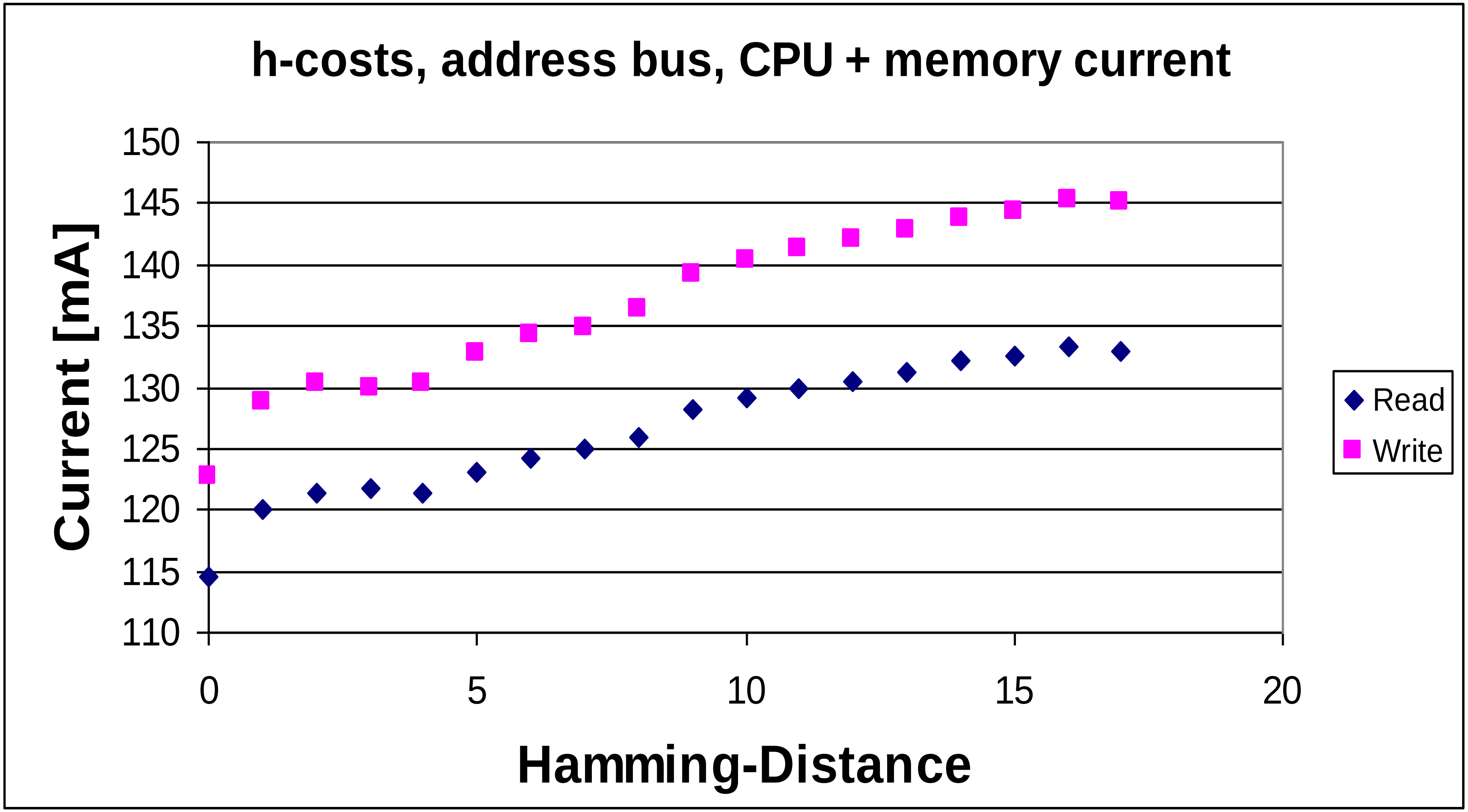
w: number of ones;

h: Hamming distance;

$FUCost$: cost of switching functional units;

α, ß: determined through experiments.

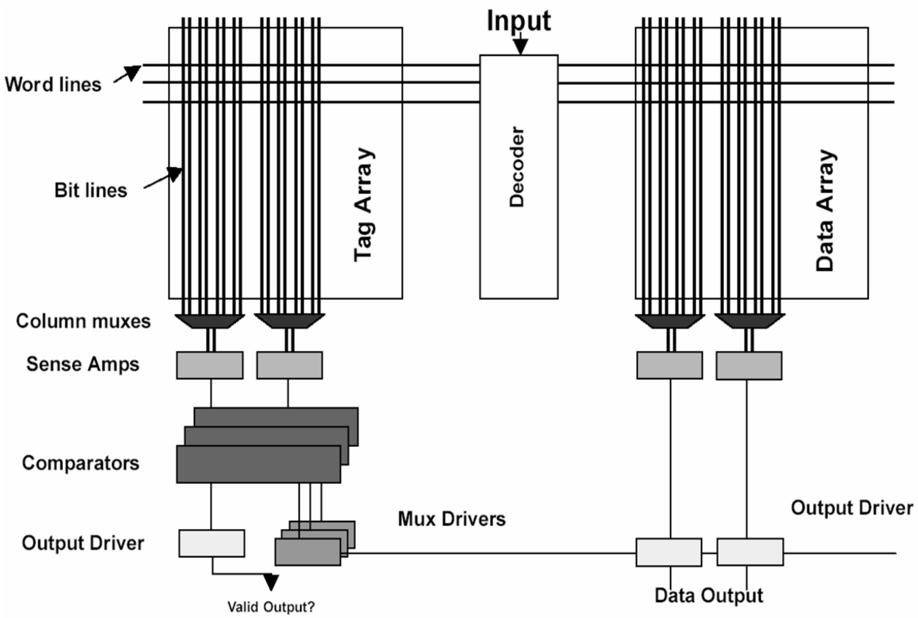
Hamming Distance between adjacent addresses is playing major role



Results

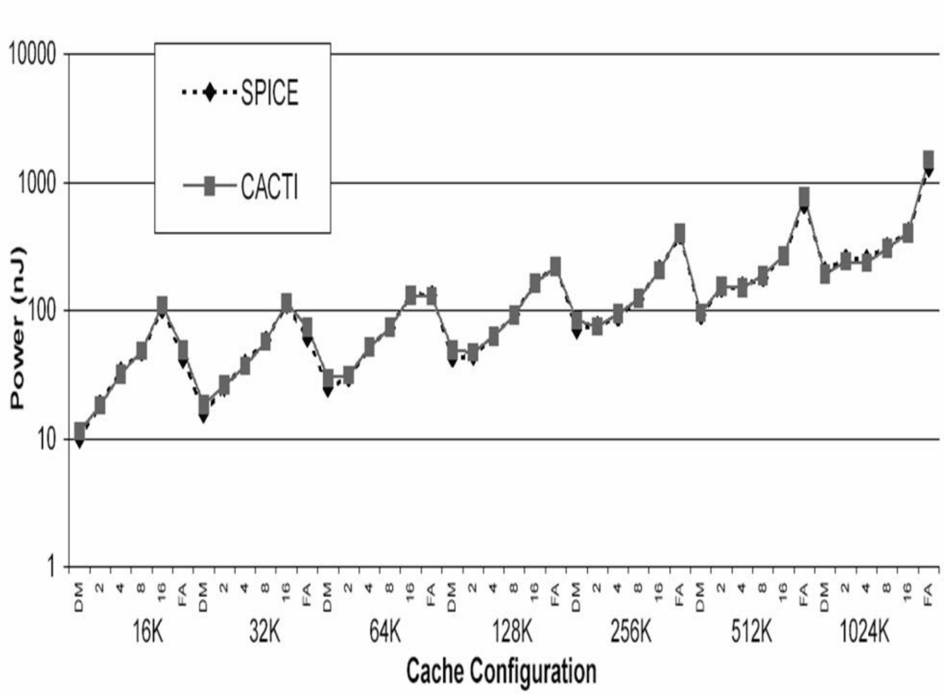
- It is not important, which address bit is set to ‘1‘
- The number of ‘1‘s in the address bus is irrelevant
- The cost of flipping a bit on the address bus is independent of the bit position.
- It is not important, which data bit is set to ‘1‘
- The number of ‘1‘s on the data bus has a minor effect (3%)
- The cost of flipping a bit on the data bus is independent of the bit position.

CACTI model



Cache model used

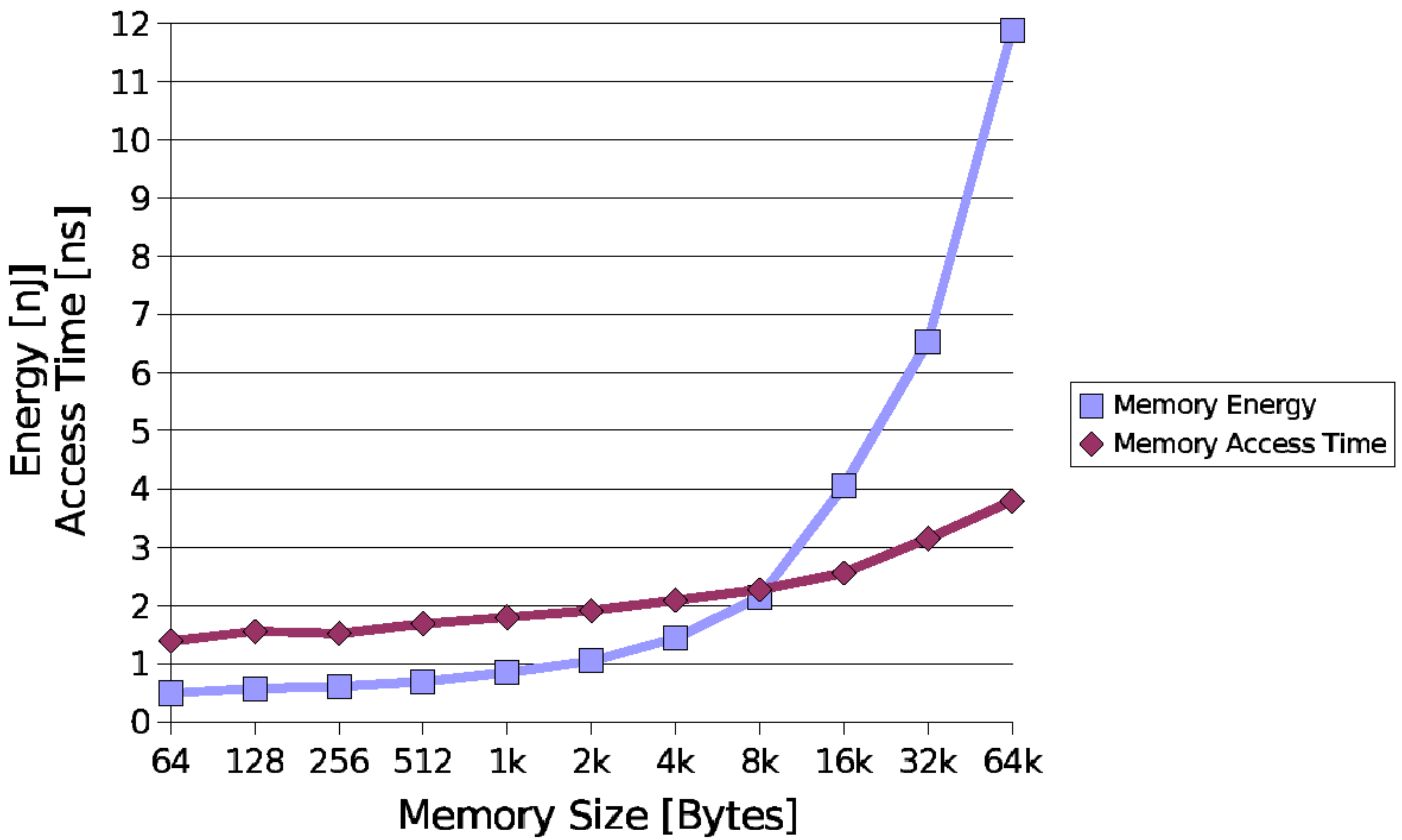
Comparison with SPICE



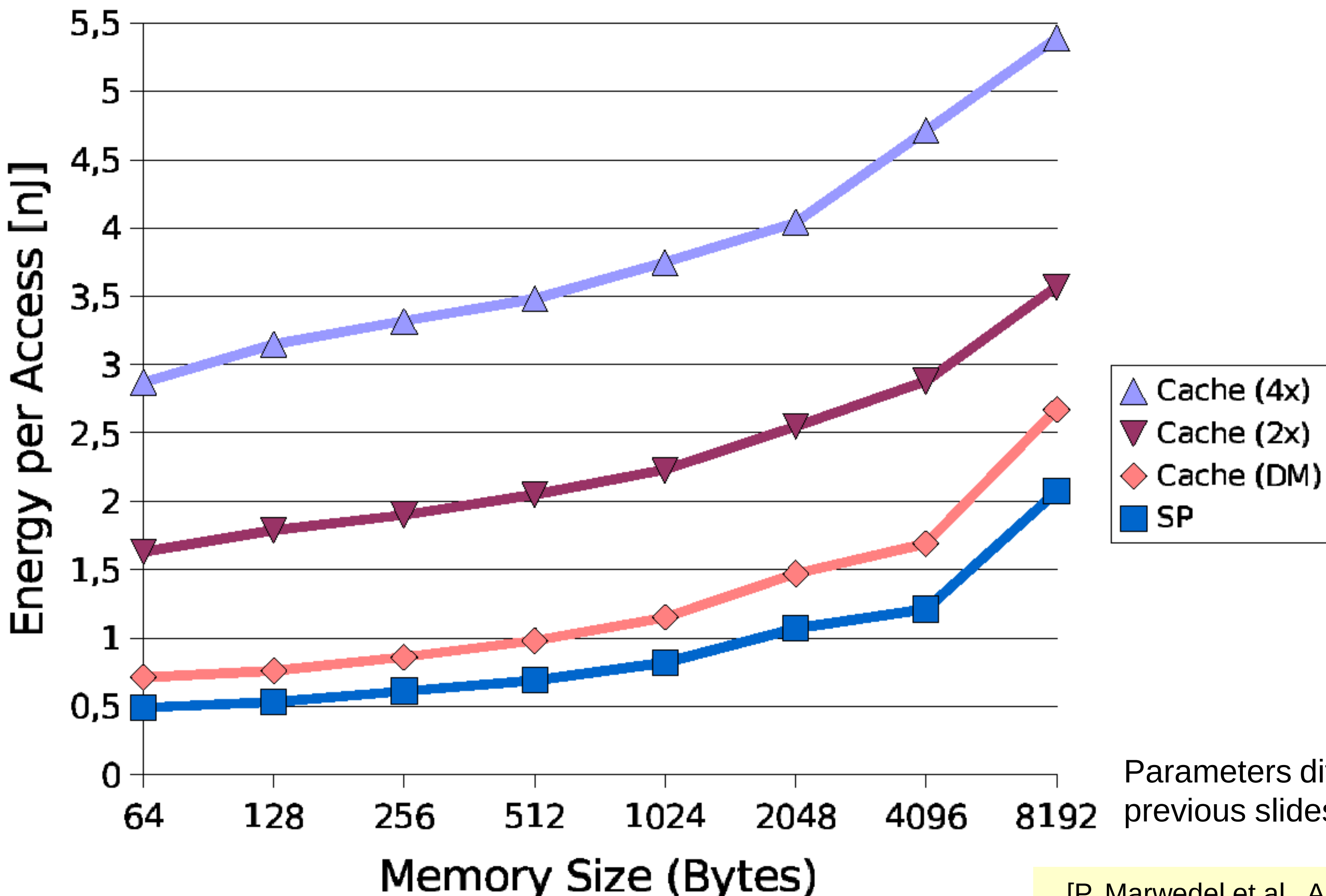
http://research.compaq.com/wrl/people/jouppi/CACTI.html

Energy consumption of memories

Example (CACTI Model):



Influence of cache associativity



Parameters different from previous slides

[P. Marwedel et al., ASPDAC, 2004]

Steinke's "combined" model

- Measured values for the processor
- Model-based values for memories
(validated against existing measurements)

Outline

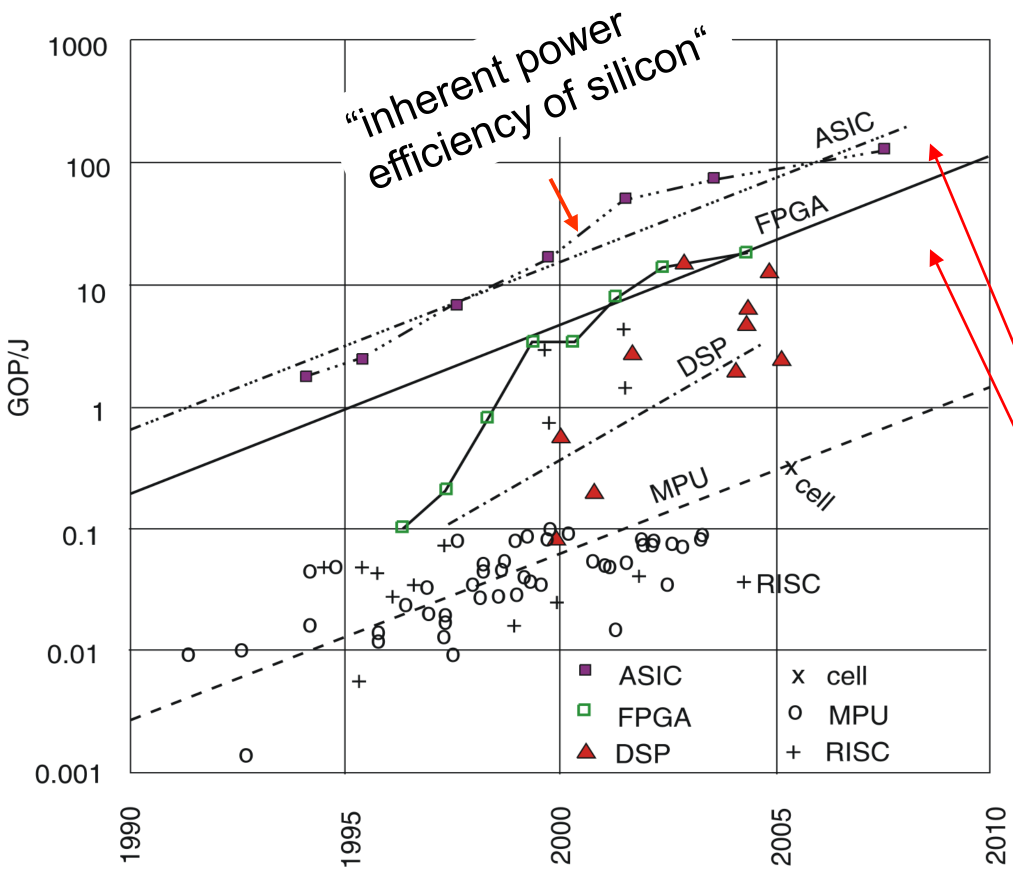
- Motivation
  - Where is energy converted into heat?
- Energy models (during use phase)
- Energy optimization (during use phase)
  - Standard optimizations
  - Exploitation of scratch pad memories in embedded computing
  - Memory-architecture aware design framework
  - Energy-efficiency of graphics processors
- Future work
- Summary

Techniques for energy reduction

$E = \int P dt$ ☞ Low-power design ≅ energy-efficient design

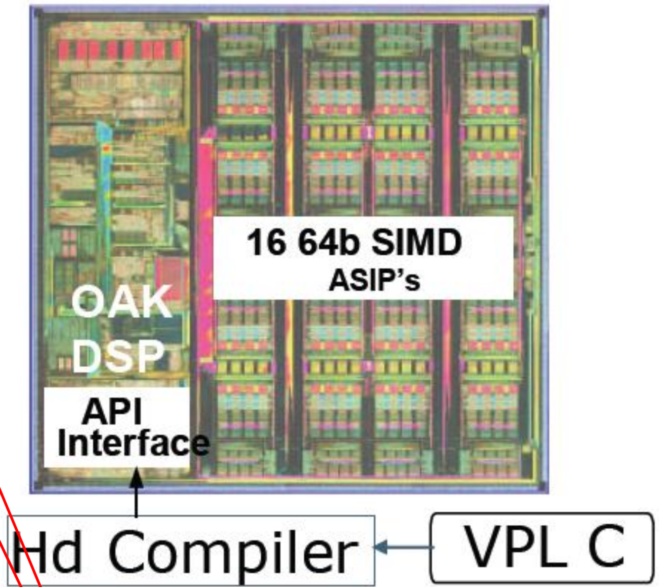
- Process technology improvements
- Circuit design
- Instruction scheduling
- Strength reduction e.g. replace * by + and <<
- Small bit width of loads and stores
- Access of energy efficient memory
- Voltage scaling
- Frequency scaling
- Low power modes (DPM)
- Energy-efficient architectures

Energy-efficient architectures: Domain- and application specific



© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, *Keynote Slides*, ACACES, 2007

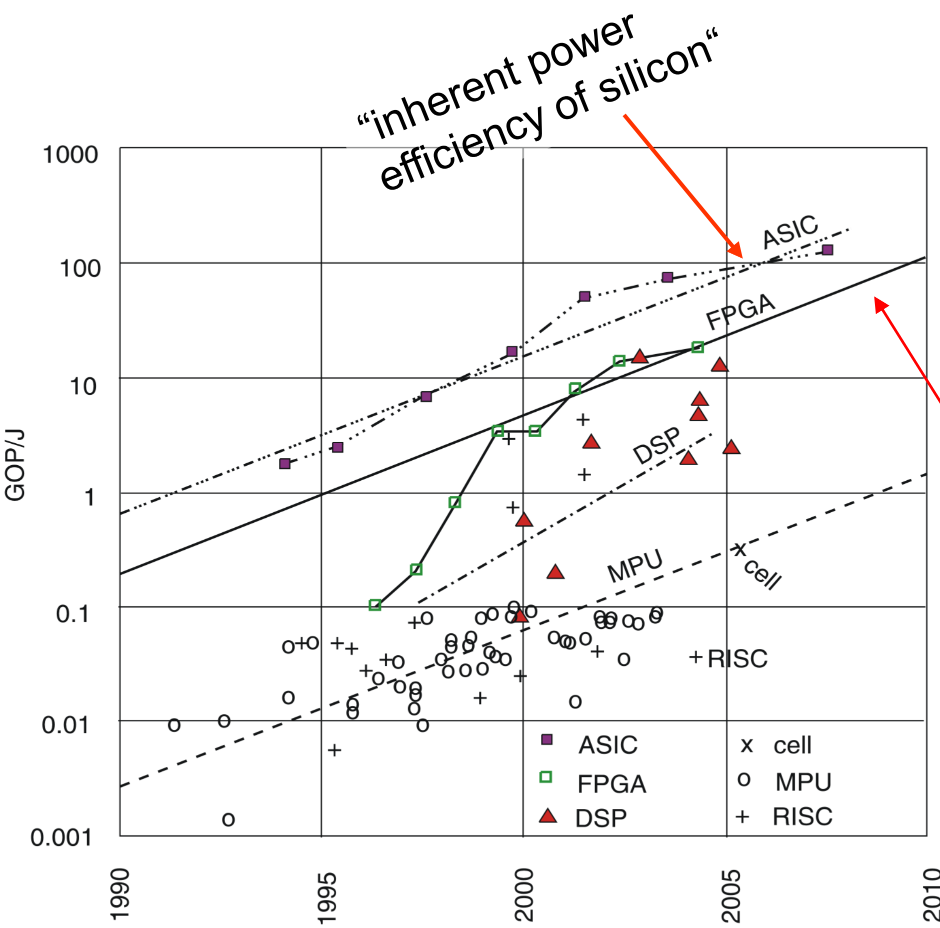
VIP for car mirrors Infineon



200MHz , 0.76 Watt
100Gops @ 8b
25Gops @ 32b

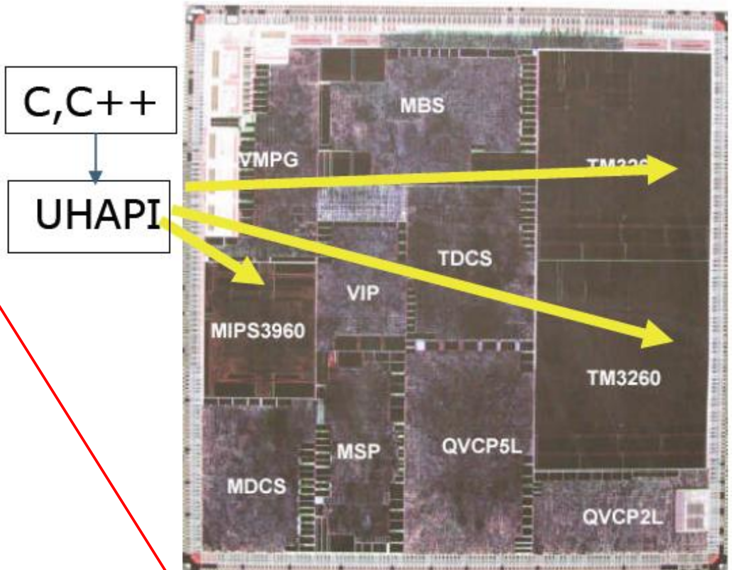
Close to power efficiency of silicon

Energy-efficient architectures: Domain- and application specific



© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, *Keynote Slides*, ACACES, 2007

Nexperia Digital Video Platform NXP



1 MIPS, 2 Trimedia
60 coproc, 250 RAM's
266MHz, 1.5 watt 100 Gops

Close to power efficiency of silicon

Energy-aware compilation

- Standard compiler optimizations with energy as a cost function, e.g.
  - instruction selection
  - register pipelining

```
for i := 1 to 10 do
    c[i] := 2 * a[i] + a[i-1];
```

⇨

```
R2 := a[0];
for i := 1 to 10 do
 begin
   R1 := a[i];
   c[i] := 2 * R1 + R2;
   R2 := R1;
  end;
```

Exploitation of the memory hierarchy

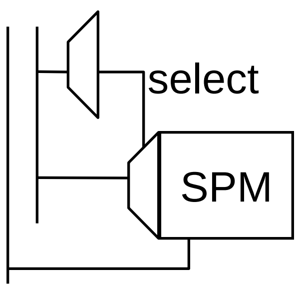
Outline

- Motivation
  - Where is energy converted into heat?
- Energy models (during use phase)
- Energy optimization (during use phase)
  - Standard optimizations
  - Exploitation of scratch pad memories in embedded computing
  - Memory-architecture aware design framework
  - Energy-efficiency of graphics processors
- Future work
- Summary

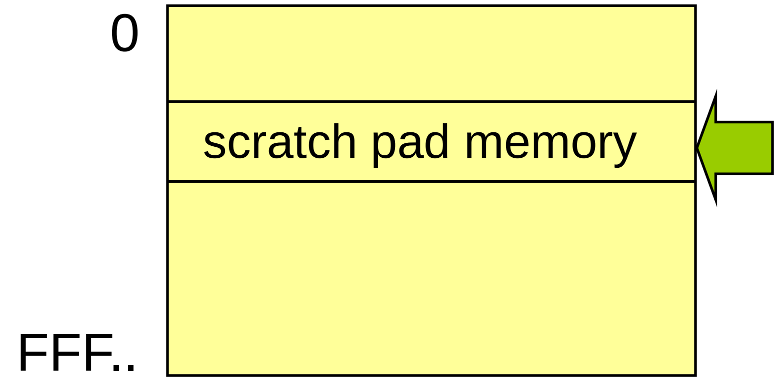
Scratch pad memories (SPM): popular in embedded (unplugged) computing

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

Selection is by an appropriate address decoder (simple!)



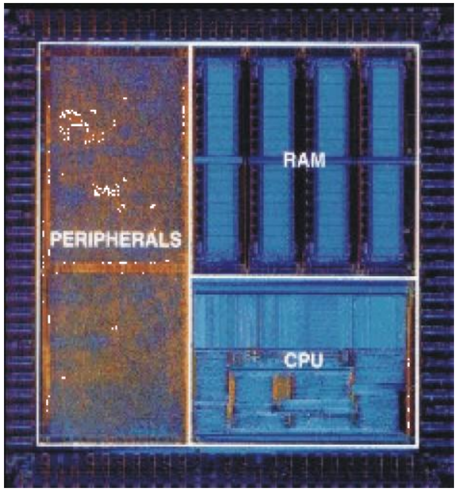
Address space



Small; no tag memory

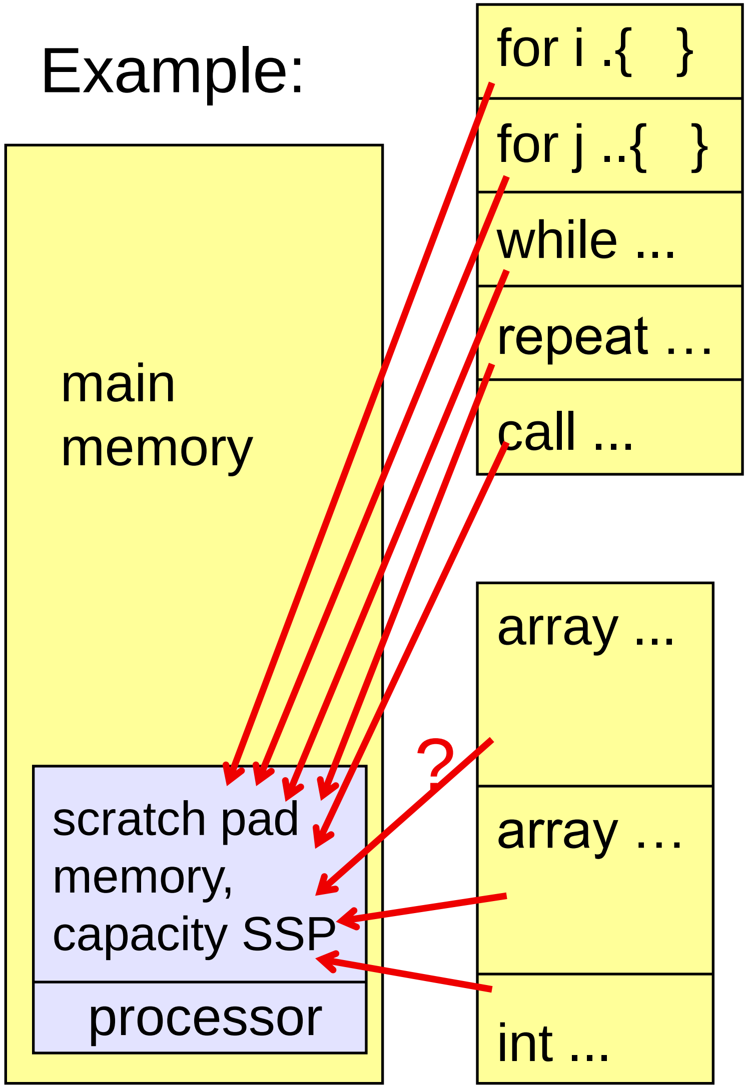
Example

ARM cores, well-known for low power consumption



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. Maximise gain $G = \Sigma g_k$, respecting <u>s</u>ize of <u>SP</u>M $SSP \geq \Sigma\, s_k$.

Solution: knapsack algorithm.

Overlaying ("dynamic") allocation:

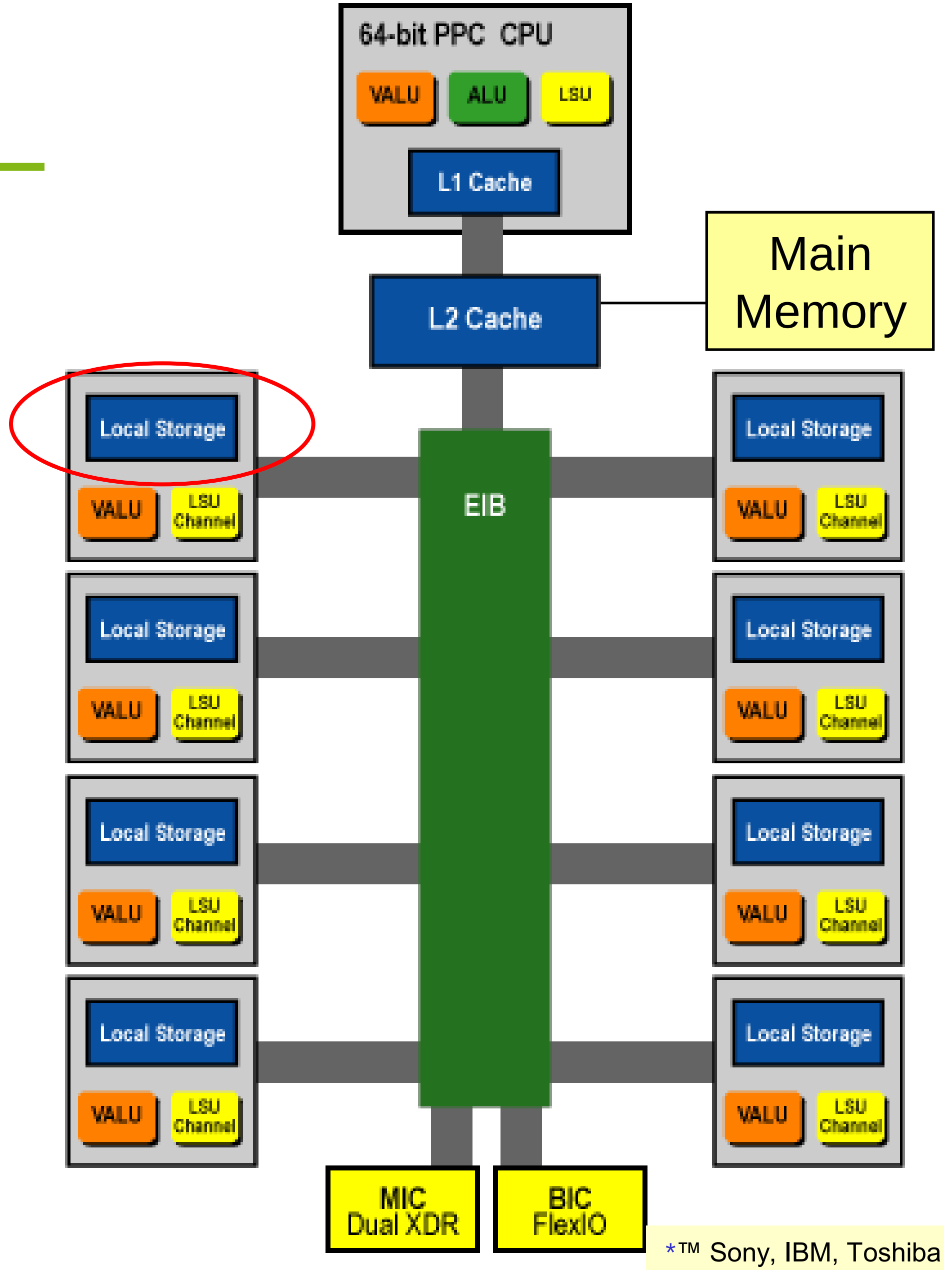
Moving objects back and forth

Similar Concept for the Cell processor *

Local SPE processors fetch instructions and data from local storage LS (256 kB).
LS **not** designed as a cache.
Separate DMA transfers required to fill and spill.

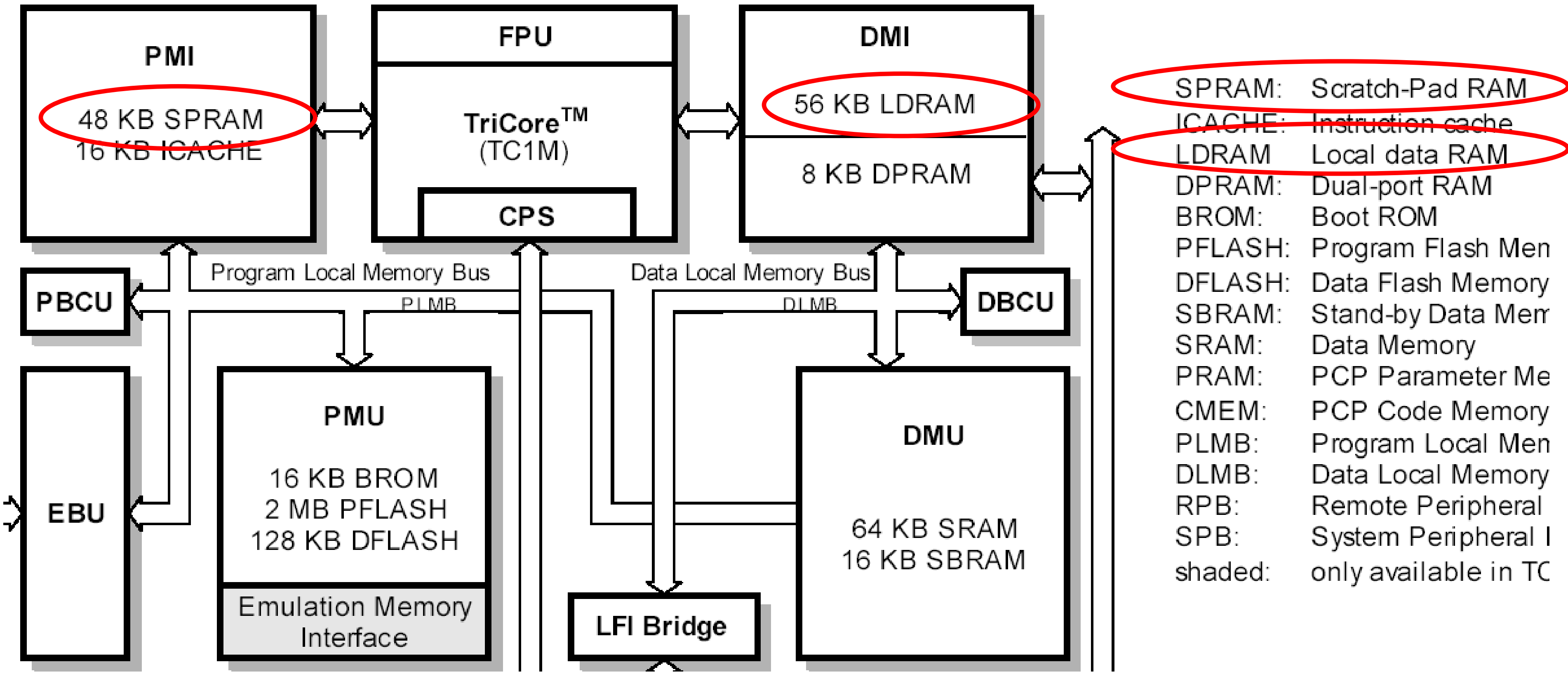
Motivation same as for this tutorial:

- Large memory latency
- Huge overhead for automatically managed caches



*™ Sony, IBM, Toshiba

Infineon TriCore
(Popular for automotive applications)



© Infineon, 2005

ILP representation
- migrating functions and variables-

Symbols:

$s(var_k)$ = size of variable k

$n(var_k)$ = number of accesses to variable k

$e(var_k)$ = energy **saved** per variable access, if var_k is migrated

$E(var_k)$ = energy **saved** if variable var_k is migrated (= $e(var_k)\ n(var_k)$)

$x(var_k)$ = decision variable, =1 if variable k is migrated to SPM,
=0 otherwise

K = set of variables; similar for functions I

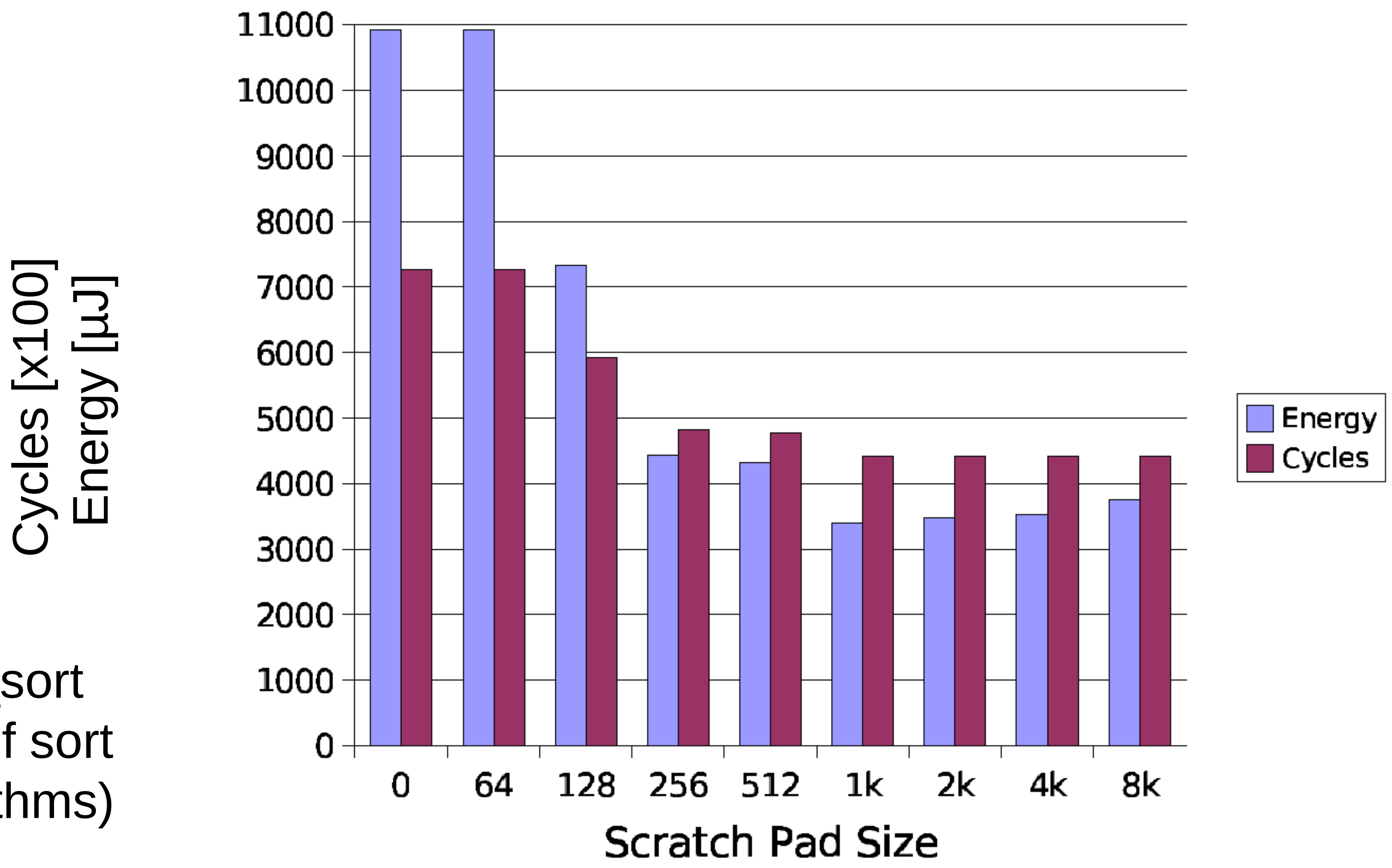
Integer programming formulation:

Maximize $\sum_{k \in K} x(var_k)\ E(var_k) + \sum_{i \in I} x(F_i)\ E(F_i)$

Subject to the constraint

$\sum_{k \in K} s(var_k)\ x(var_k) + \sum_{i \in I} s(F_i)\ x(F_i) \leq SSP$

Reduction in energy and average run-time

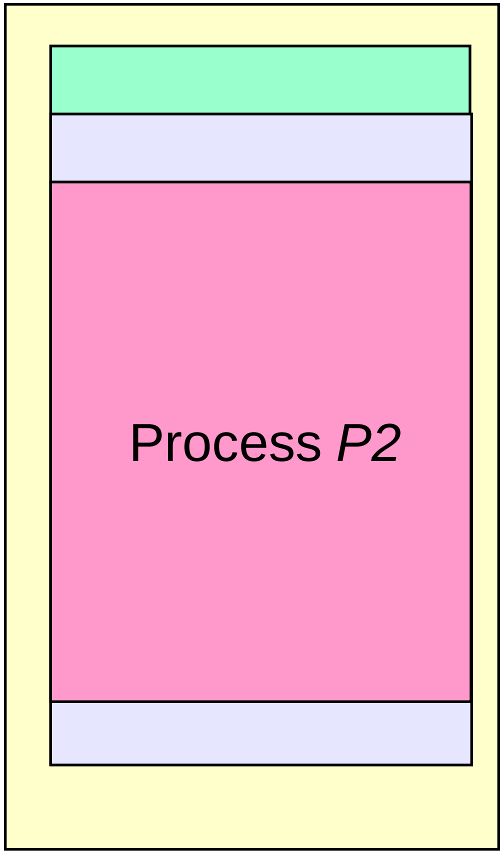


Multi_sort (mix of sort algorithms)

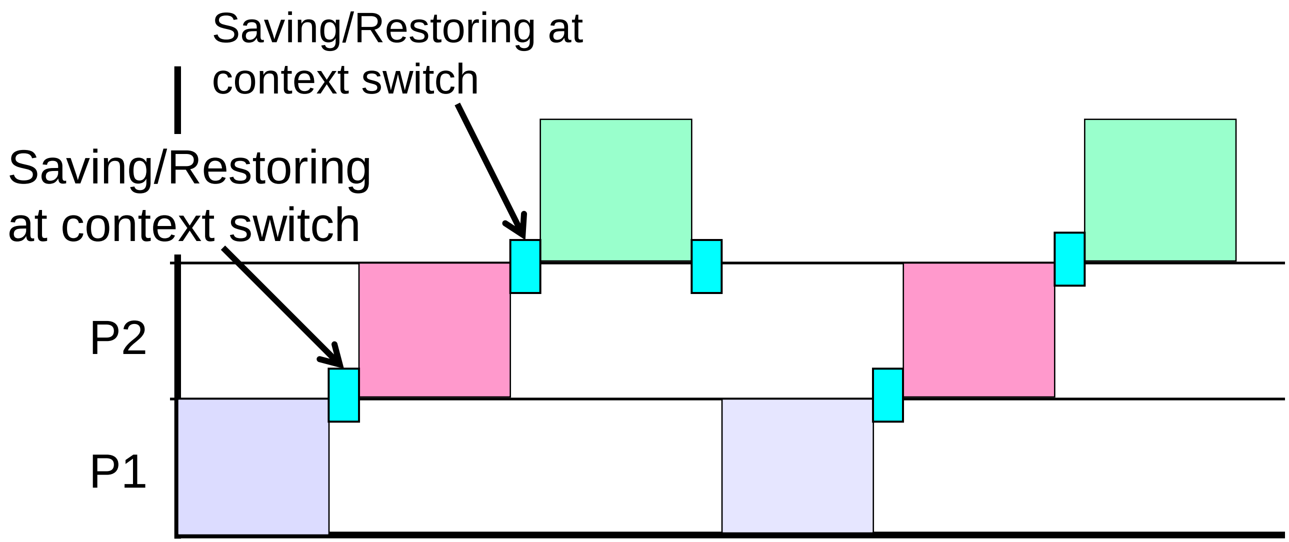
Measured processor / external memory energy + CACTI values for SPM (combined model)

Numbers will change with technology, algorithms remain unchanged.

Saving/Restoring Context Switch



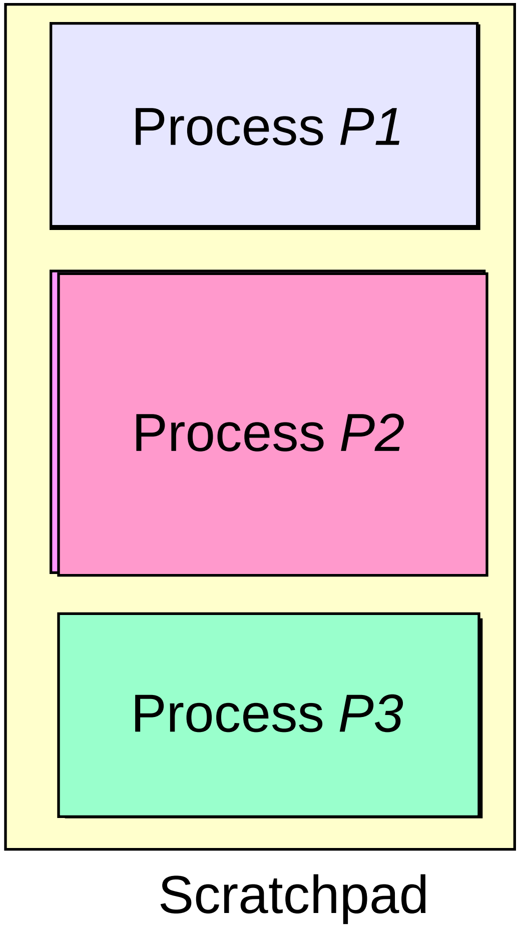
Scratchpad

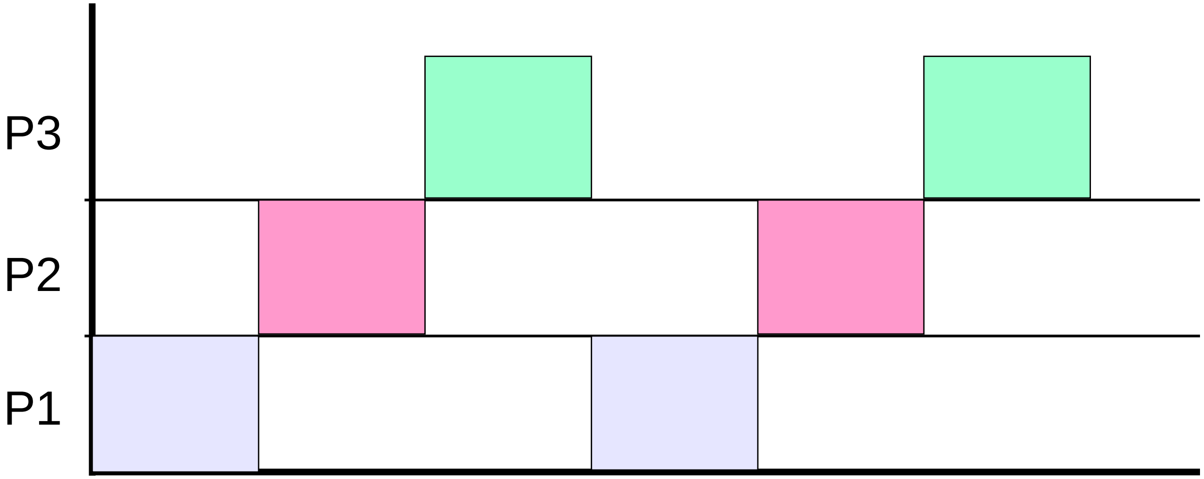


Saving Context Switch (Saving)

- Utilizes SPM as a common region shared by all processes
- Contents of processes are copied on/off the SPM at context switch
- Works for small scratchpads

Non-Saving Context Switch

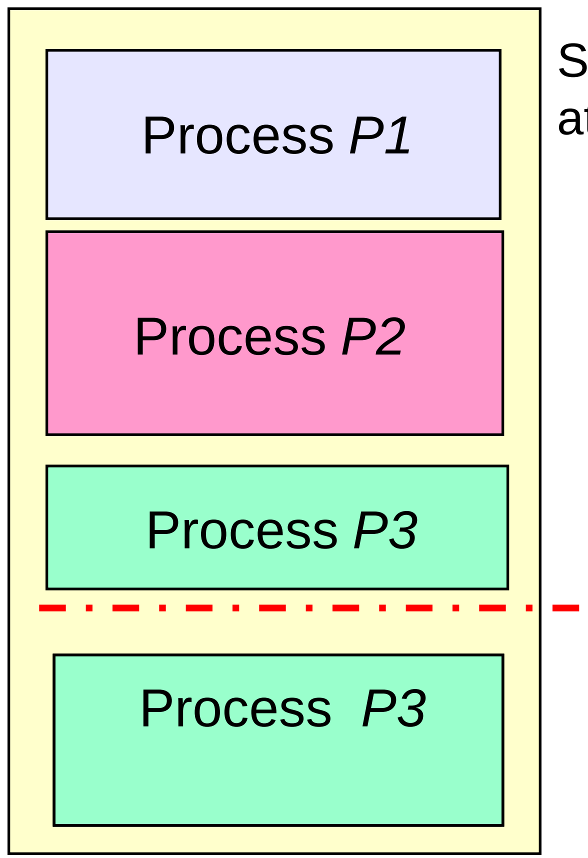




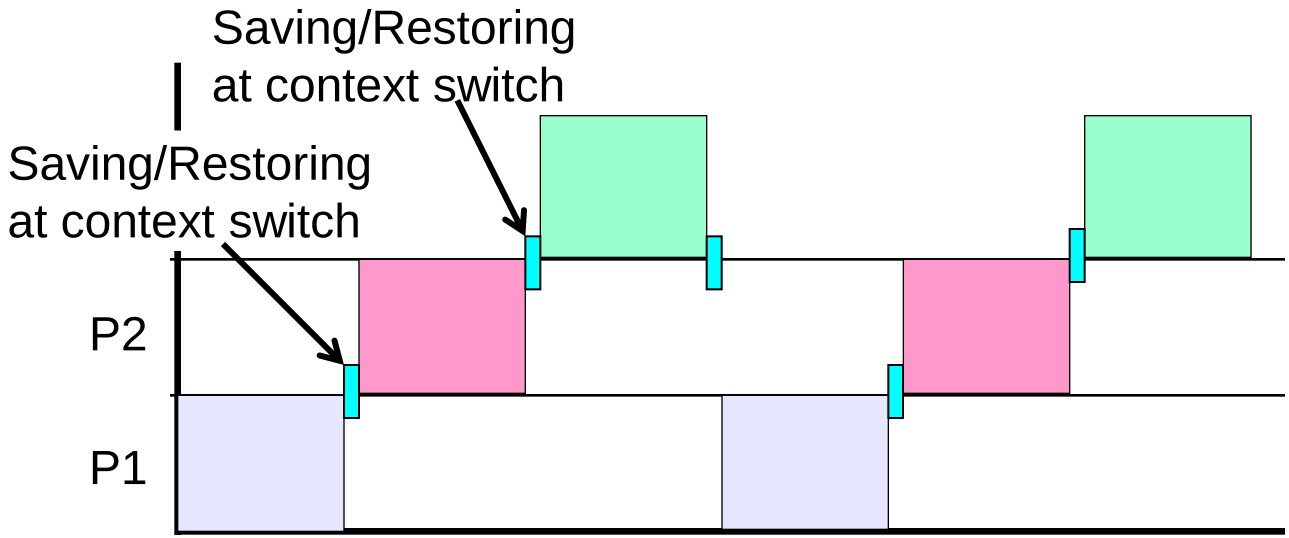
Non-Saving Context Switch

- Partitions SPM into disjoint regions
- Each process is assigned a SPM region
- Copies contents during initialization
- Good for large scratchpads

Hybrid Context Switch



Scratchpad



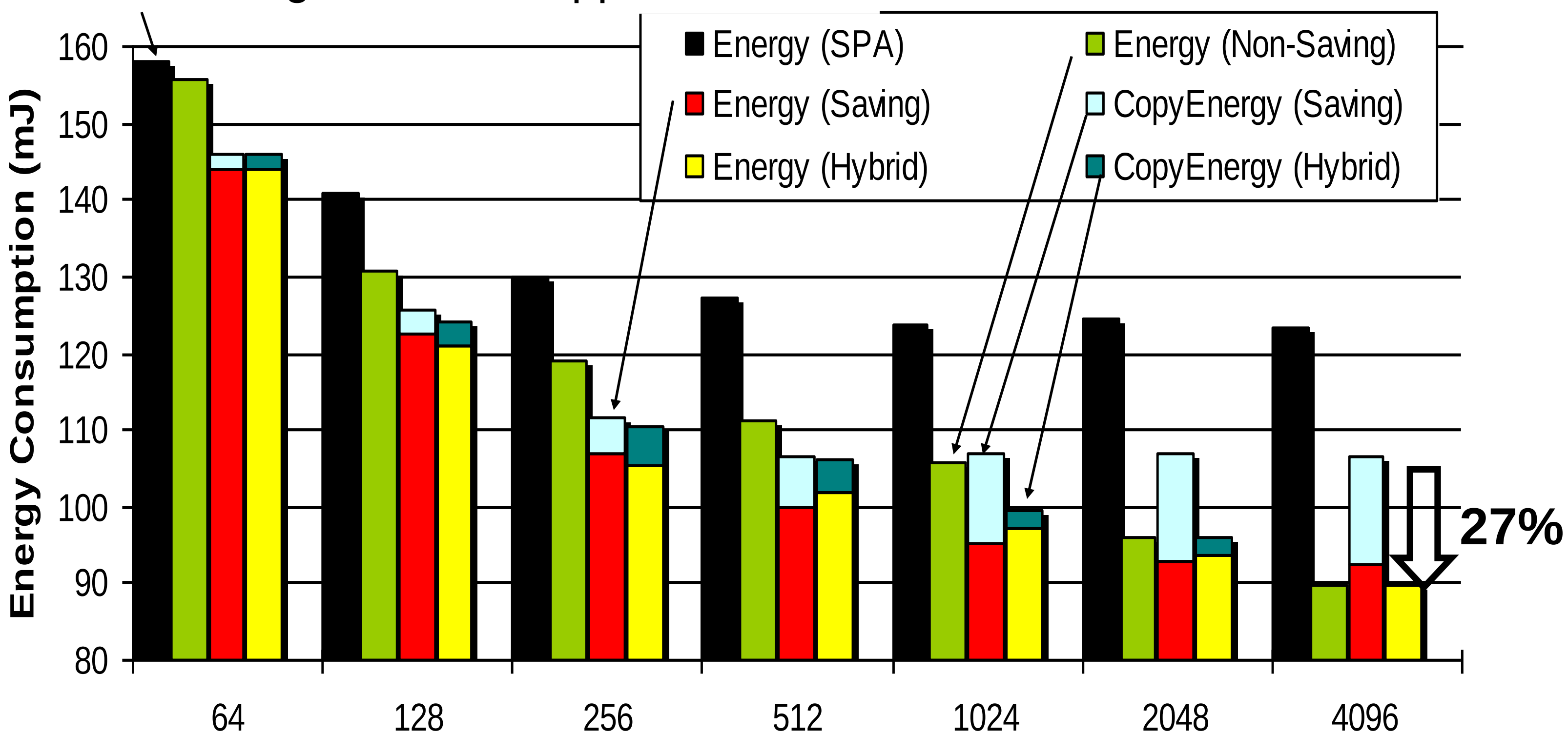
Hybrid Context Switch (Hybrid)

- Disjoint + Shared SPM regions
- Good for all scratchpads

M. Verma, K. Petzold, L. Wehmeyer, H. Falk, P. Marwedel: Scratchpad Sharing Strategies for Multiprocess Embedded Systems: A First Approach, IEEE 3rd Workshop on Embedded System for Real-Time Multimedia (ESTIMedia), 2005

Multi-process Scratchpad Allocation: Results

SPA: Single Process Approach



- For small SPMs (64B-512B) Saving is better
- For large SPMs (1kB- 4kB) Non-Saving is better
- Hybrid is the best for all SPM sizes.
- Energy reduction @ 4kB SPM is 27% for Hybrid approach

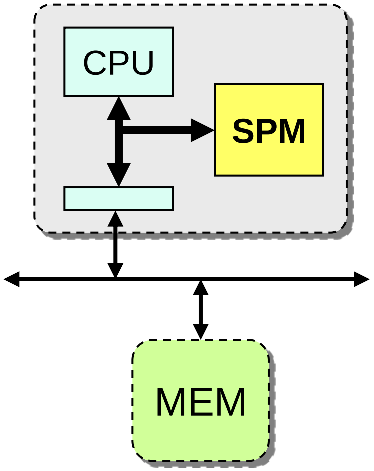
edge detection, adpcm, g721, mpeg

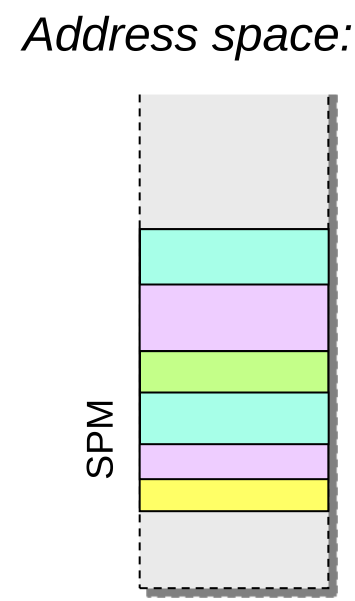
Dynamic set of multiple applications

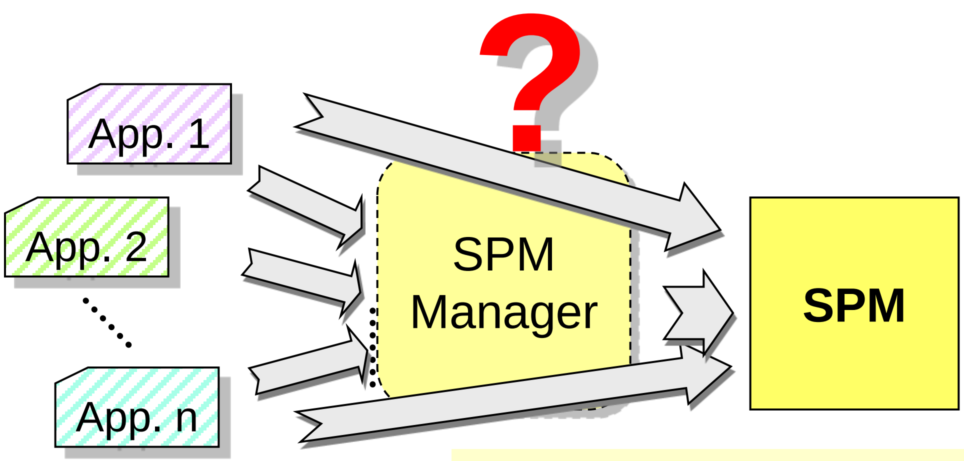
Compile-time partitioning of SPM no longer feasible

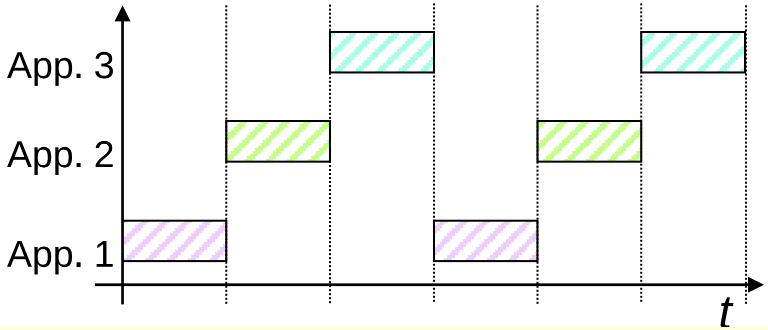
☞ Introduction of SPM-manager

- Runtime decisions, but compile-time supported









R. Pyka, Ch. Faßbach, M. Verma, H. Falk, P. Marwedel: Operating system integrated energy aware scratchpad allocation strategies for multi-process applications, *SCOPES*, 2007

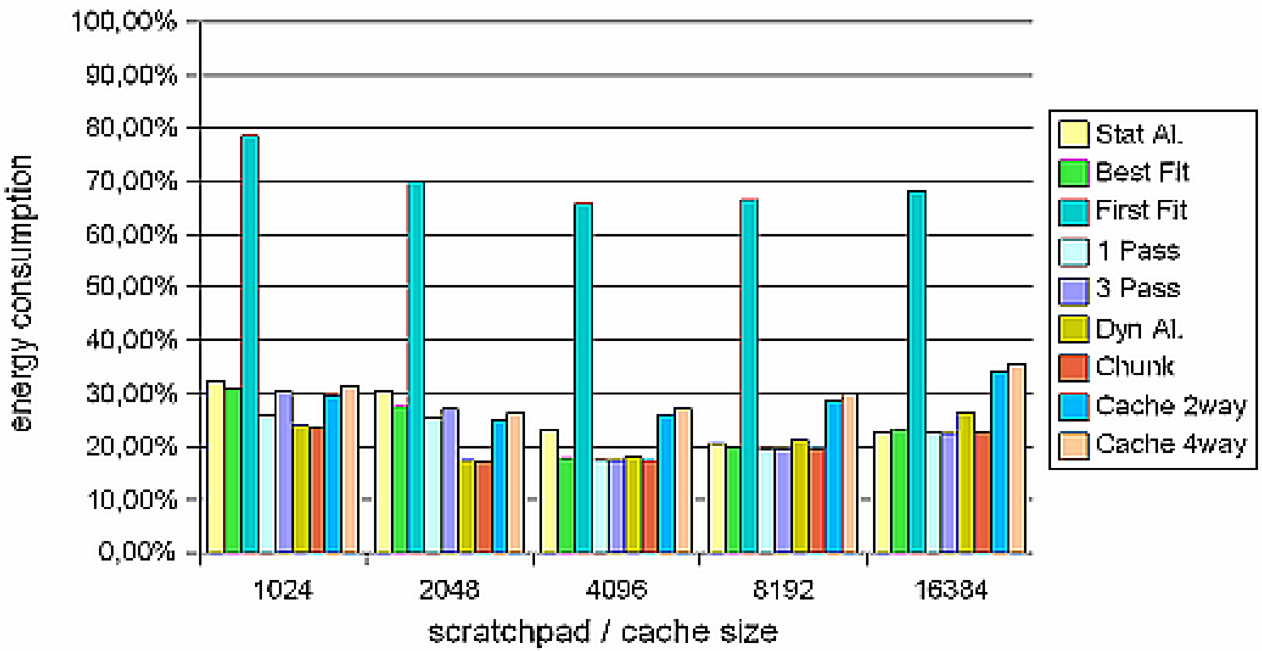
Comparison of SPMM to Caches for SORT

- Baseline: Main memory only
- SPMM peak energy reduction by 83% at 4k Bytes scratchpad
- Cache peak: 75% at 2k 2-way cache

- SPMM capable of outperforming caches
- OS and libraries are not considered yet

Chunk allocation results:

SPM Size	Δ 4-way
1024	74,81%
2048	65,35%
4096	64,39%
8192	65,64%
16384	63,73%

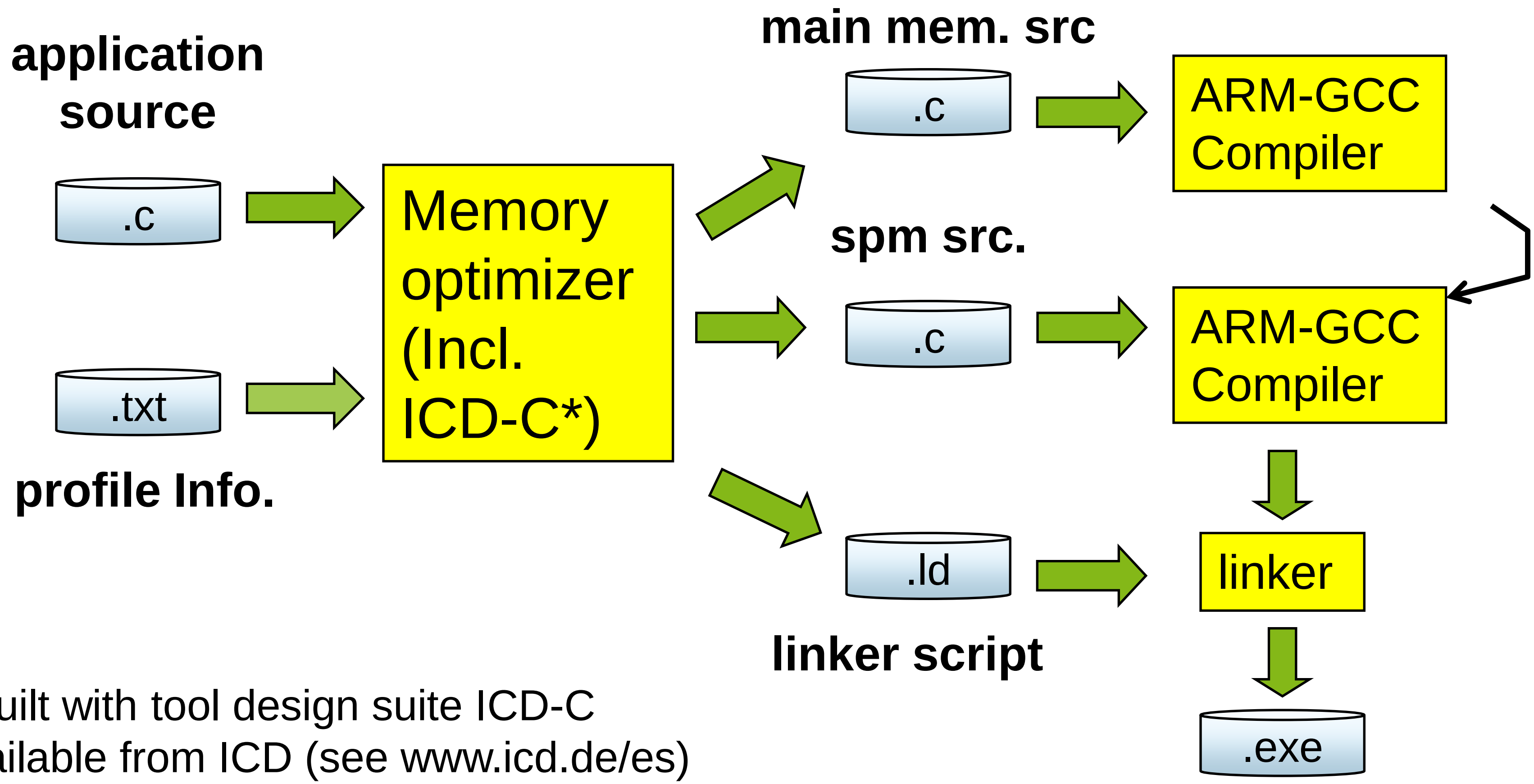


Outline

- Motivation
  - Where is energy converted into heat?
- Energy models (during use phase)
- Energy optimization (during use phase)
  - Standard optimizations
  - Exploitation of scratch pad memories in embedded computing
  - Memory-architecture aware design framework
  - Energy-efficiency of graphics processors
- Future work
- Summary

Using these ideas with an gcc-based tool flow

Source is split into 2 different files by specially developed memory optimizer tool *.

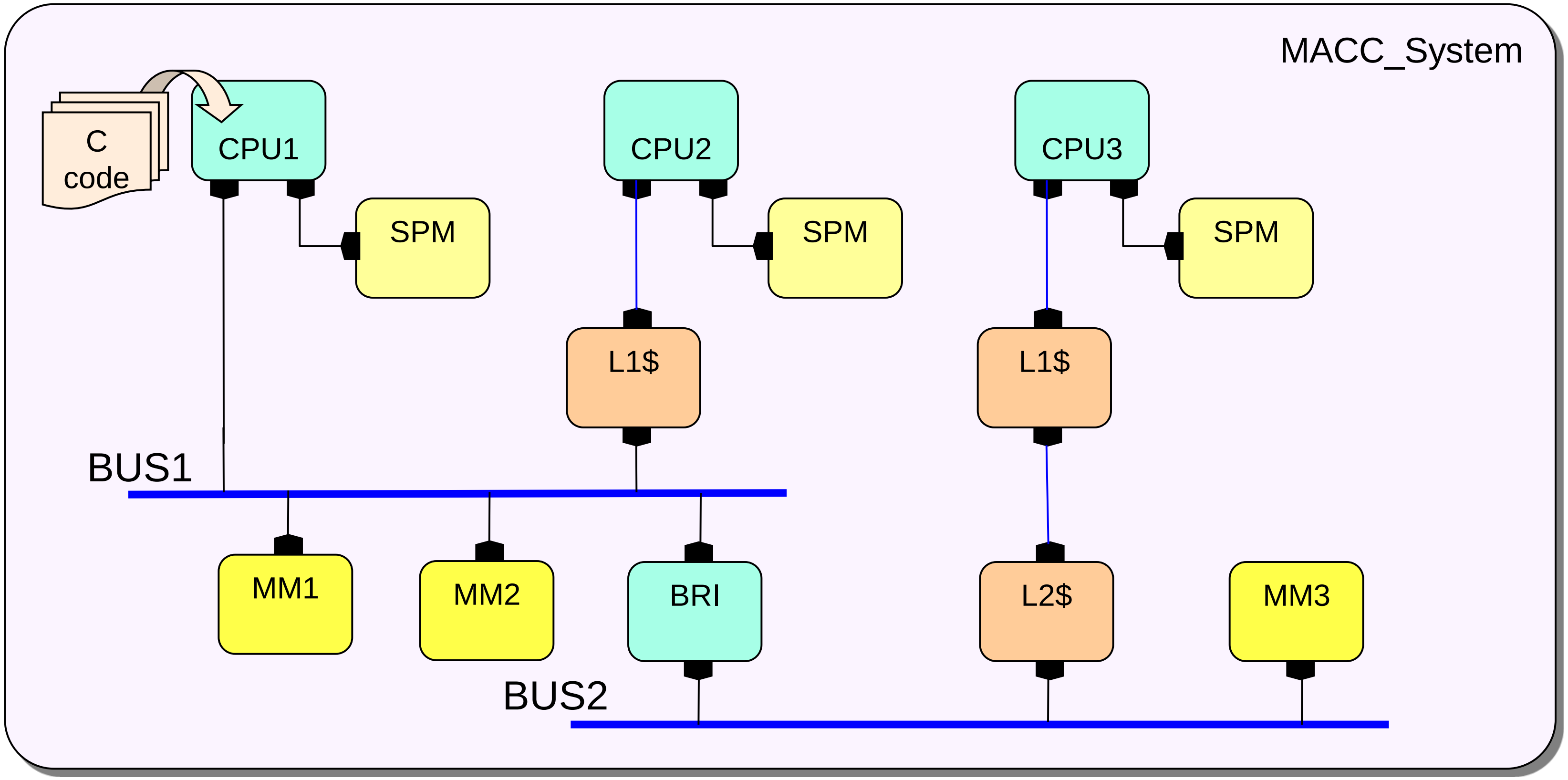


* Built with tool design suite ICD-C available from ICD (see www.icd.de/es)

Introduction of Memory Architecture-Aware Optimization The MACC PMS (Processor/Memory/Switch) Model

- Explicit memory architecture
- API provides access to memory information

www.mnemee.org

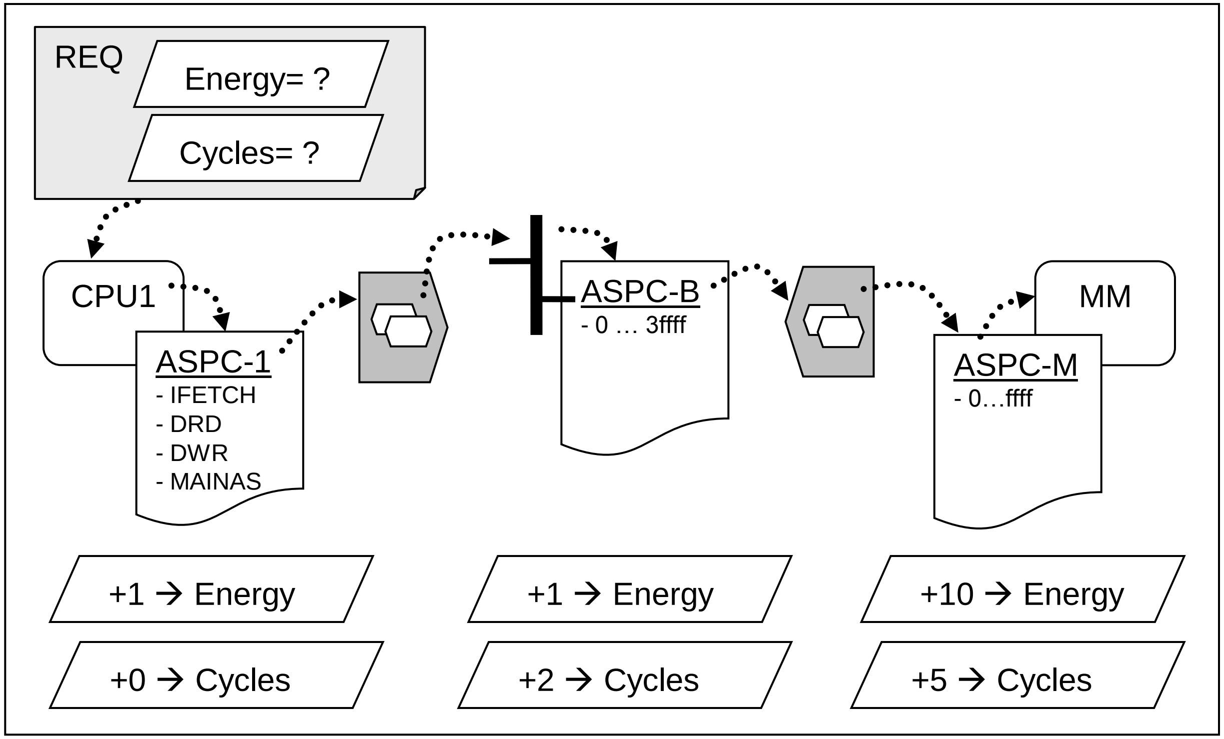


Memory architecture description @ MACCv2

- Query can include address, time stamp, value, …
- Query can request energy, delay, stored values
- Query processed along a chain of HW components, incl. busses, address translations etc., each adding delay & energy

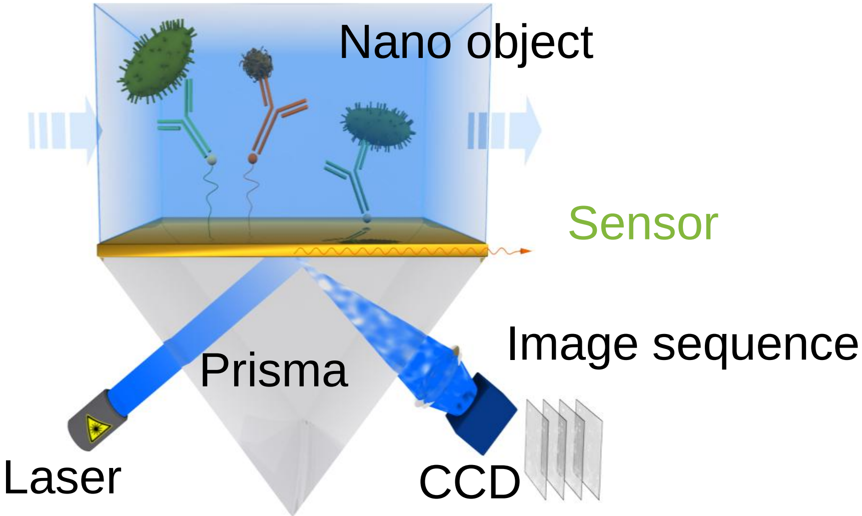
www.mnemee.org

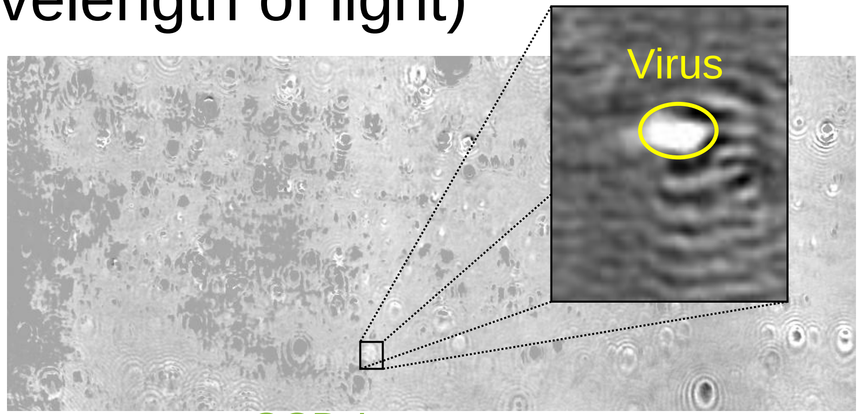
- API query to model simplifies integration into compiler
- External XML representation



SFB 876: Resource-efficient machine learning
Project B2: Resource-efficient virus detection

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



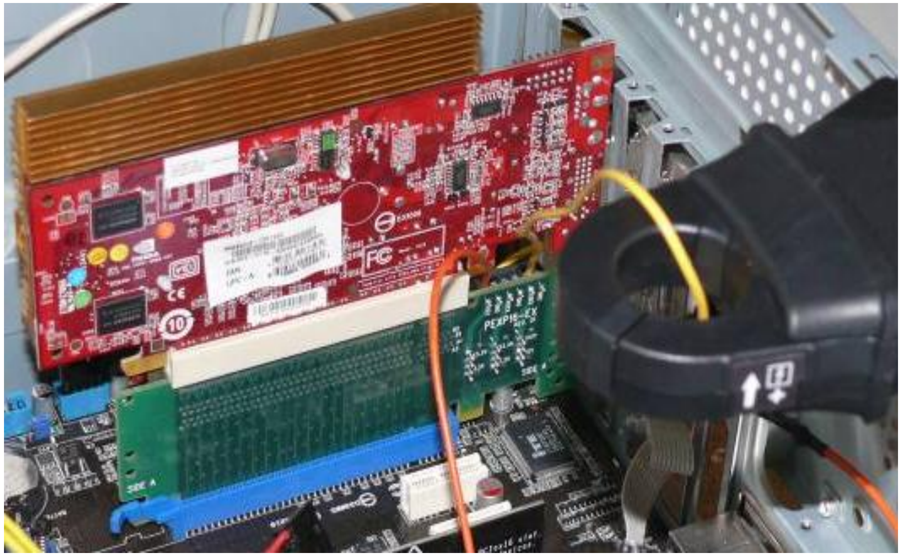


CCD-Image

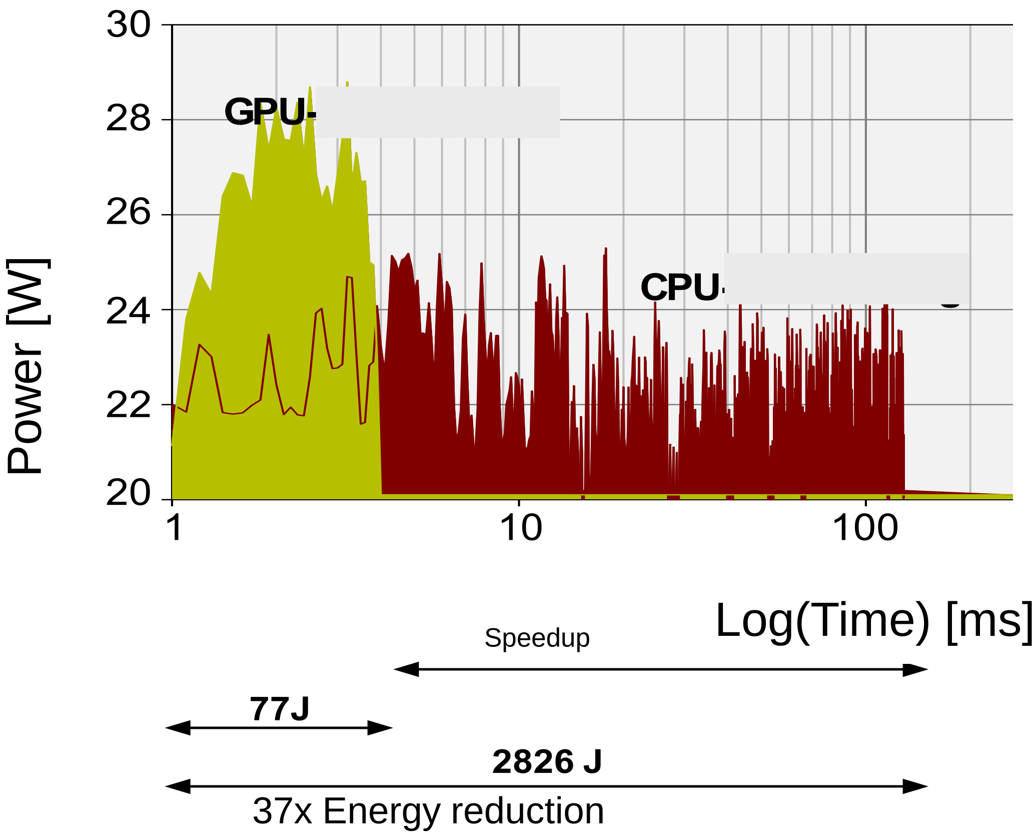
Project goal

The project focuses on studying the tradeoffs between energy, runtime, memory size and detection quality

☞ start with an efficient implementation: an efficient graphics processor, compute energy consumption.



Energy-efficient execution on graphics processor (GPU)



Future work:

Considering tradeoff between energy consumption and detection quality

Outline

- Motivation
  - Where is energy converted into heat?
- Energy models (during use phase)
- Energy optimization (during use phase)
  - Standard optimizations
  - Exploitation of scratch pad memories in embedded computing
  - Memory-architecture aware design framework
  - Energy-efficiency of graphics processors
- Future work
- Summary

Future work

- Completion of the Mnemee tool chain
- Energy-efficient design for the PAMONO sensor
- Comparison of GPGPU approach with FPGA-based approach
- Multi-objective optimization
(also considering WCET, ACET, code size, reliability)
- Early estimation of required energy, allowing algorithm designers to estimate the required energy early
- Tradeoffs between result quality and energy consumption

Summary

- For energy-efficient designs, we have to consider
  - Production
  - and use.
- Energy models can be based on
  - Real hardware (precise, but HW has to be available)
  - Models (no HW required, but imprecise)
- Optimizations possible at several levels
  - Device level/chip production
  - Circuit level
  - Compiler/tool chain level
  - Run-time support (e.g. DPM)
- Some time until industrial tools become available