

Embedded System Hardware - Processing -

Peter Marwedel
Informatik 12
TU Dortmund
Germany



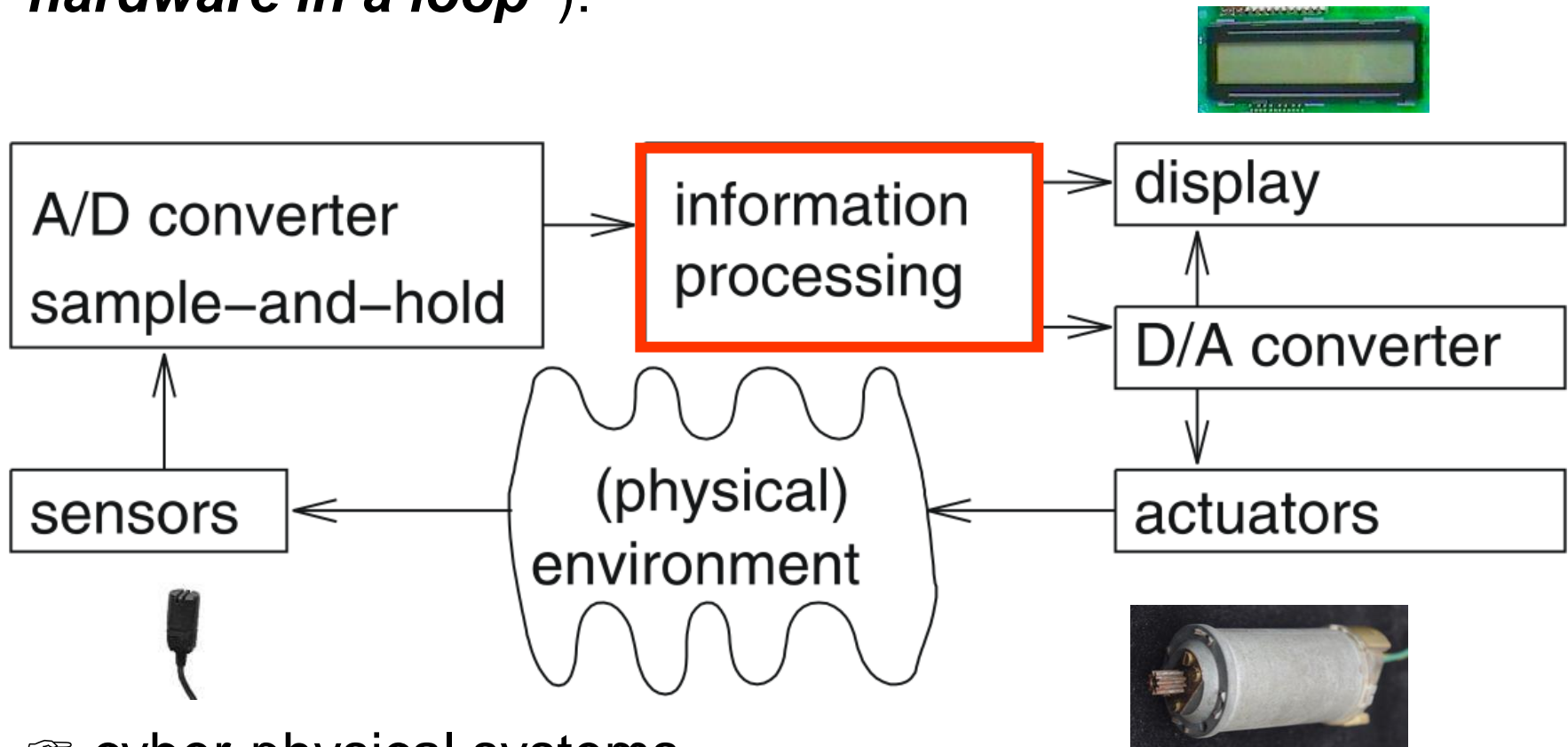
© Springer, 2010

2013年 11 月 12 日

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.

Embedded System Hardware

Embedded system hardware is frequently used in a loop (***“hardware in a loop”***):



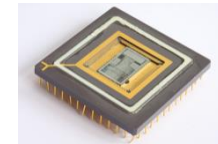
👉 cyber-physical systems

Efficiency:

slide from lecture 1 applied to processing

- CPS & ES must be **efficient**

- ➡ • Code-size efficient
(especially for systems on a chip)



- ➡ • Run-time efficient



- Weight efficient



- Cost efficient

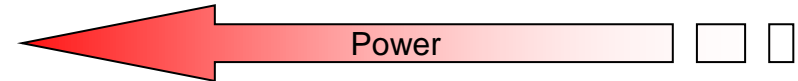


- ➡ • Energy efficient



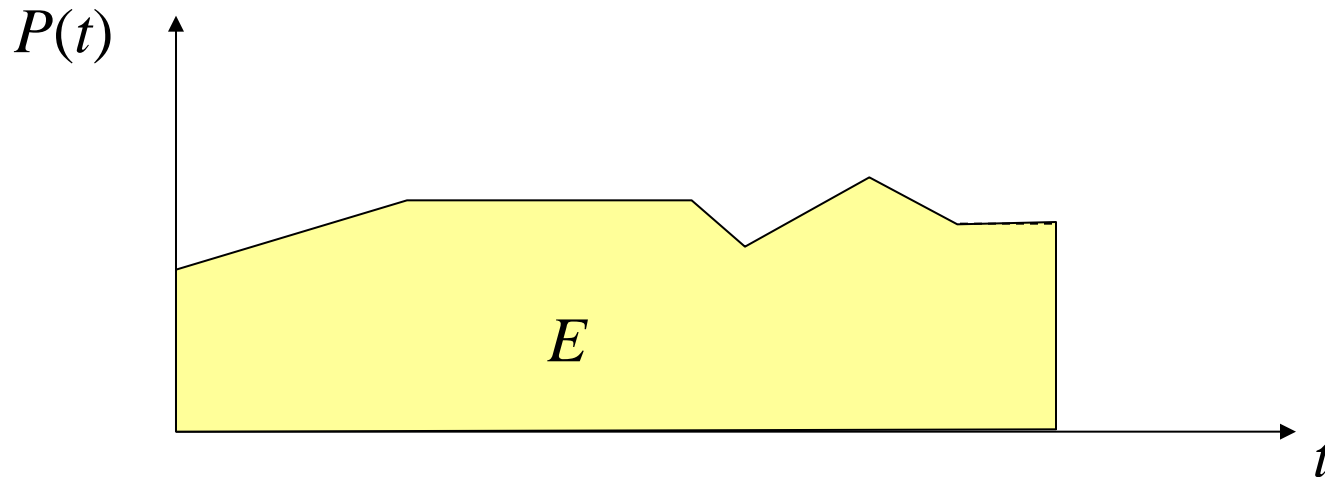
Why care about energy efficiency ?

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



Should we care about energy consumption or about power consumption?

$$E = \int P(t) dt$$



Both are closely related, but still different

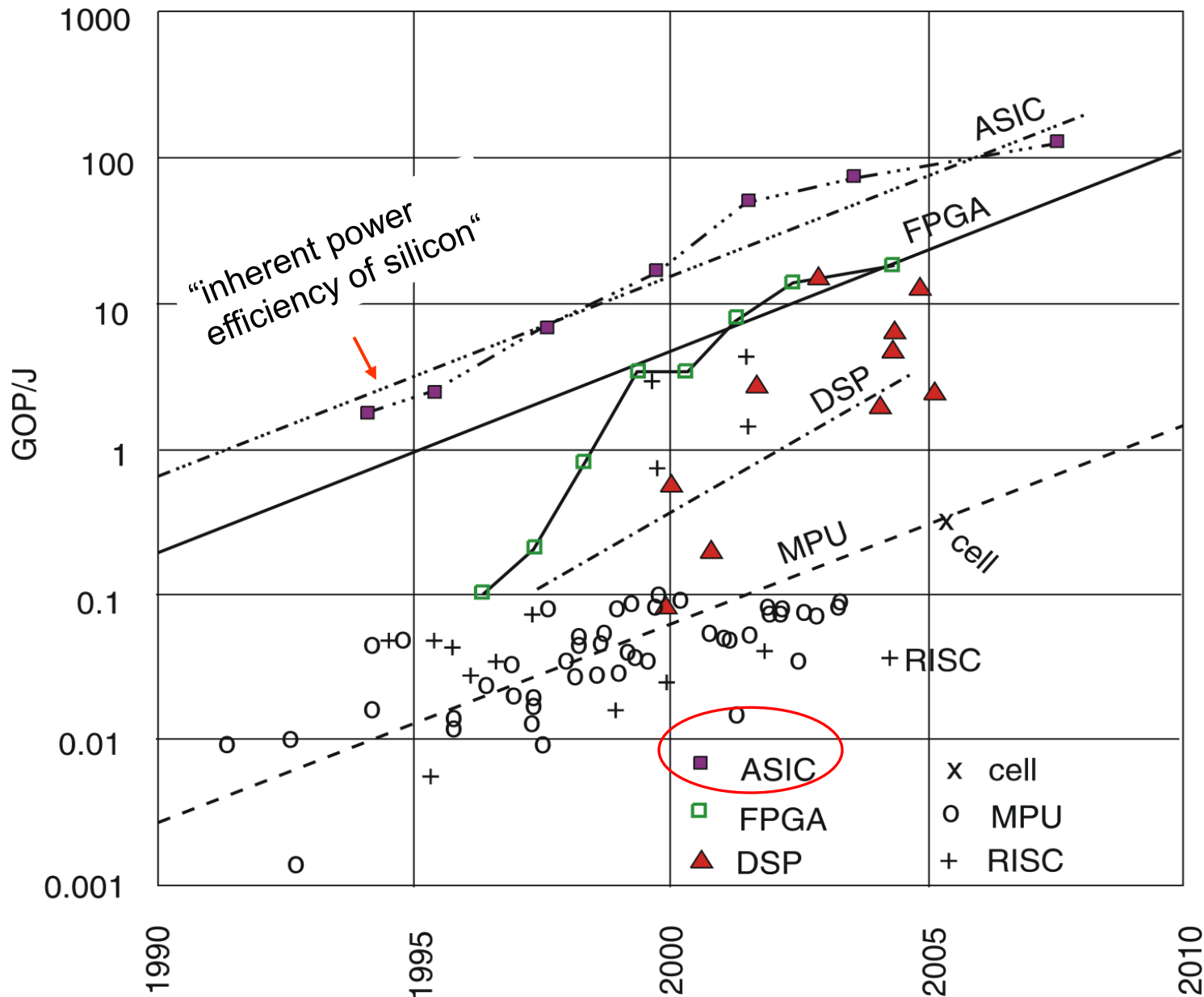
Should we care about energy consumption or about power consumption (2)?

- Minimizing **power consumption** important for
 - design of the power supply & regulators
 - dimensioning of interconnect, short term cooling
- Minimizing **energy consumption** important due to
 - restricted availability of energy (mobile systems)
 - cooling: high costs, limited space
 - thermal effects
 - dependability, long lifetimes



👉 **In general, we need to care about both**

Energy Efficiency of different target platforms

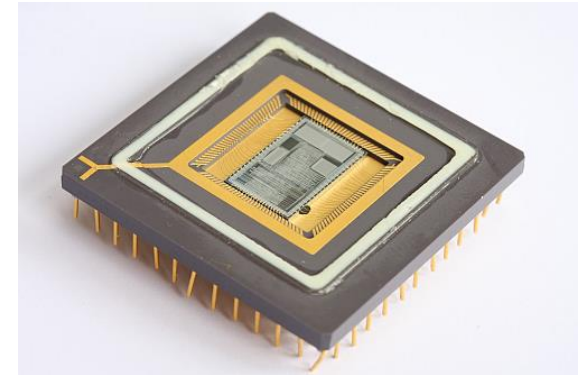


© Hugo De Man,
IMEC, Philips, 2007

Application Specific Circuits (ASICs) or Full Custom Circuits

Approach suffers from

- long design times,
- lack of flexibility
(changing standards) and
- high costs
(e.g. Mill. \$ mask costs).

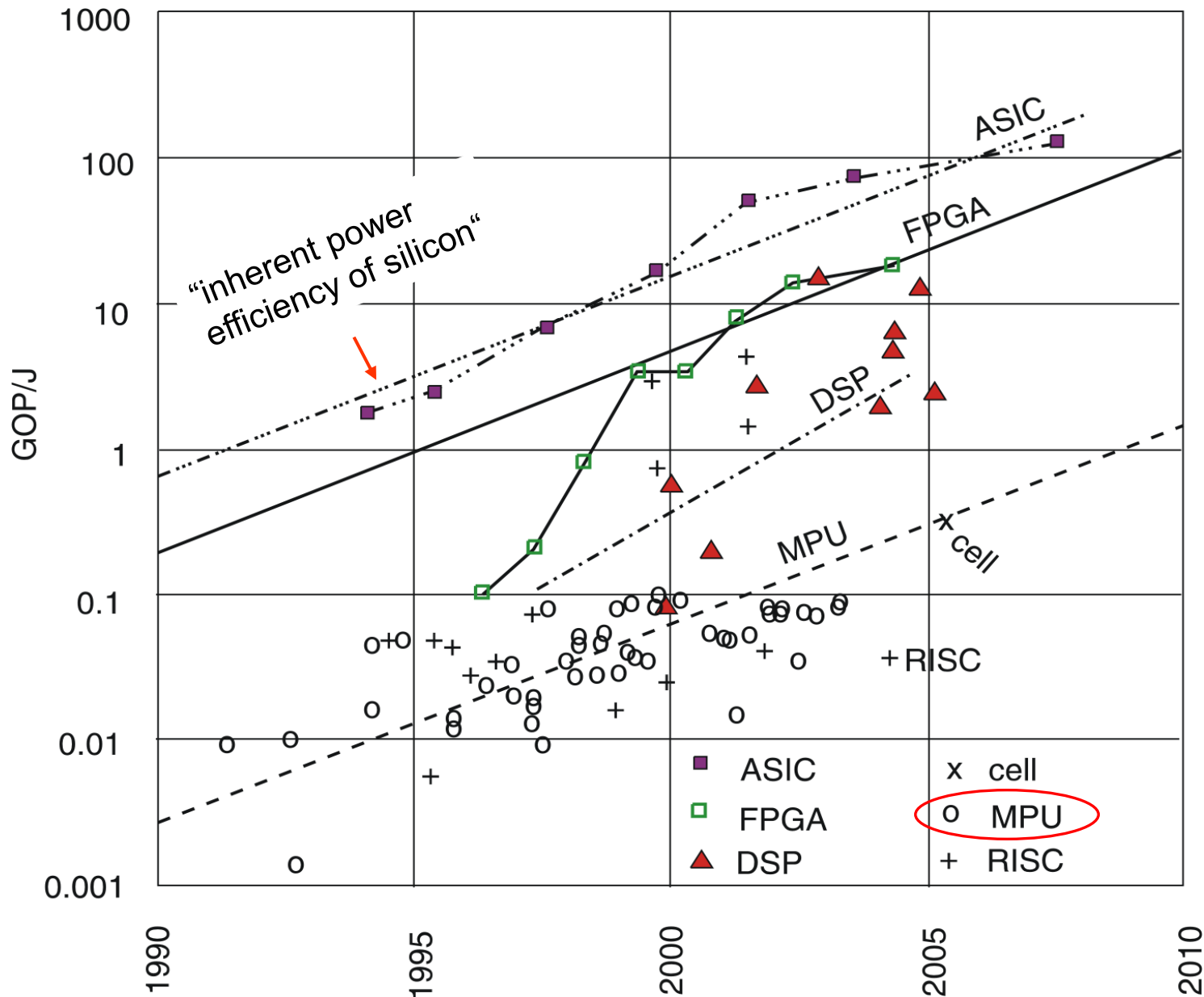


Custom-designed circuits necessary

- if ultimate speed or
- energy efficiency is the goal and
- large numbers can be sold.

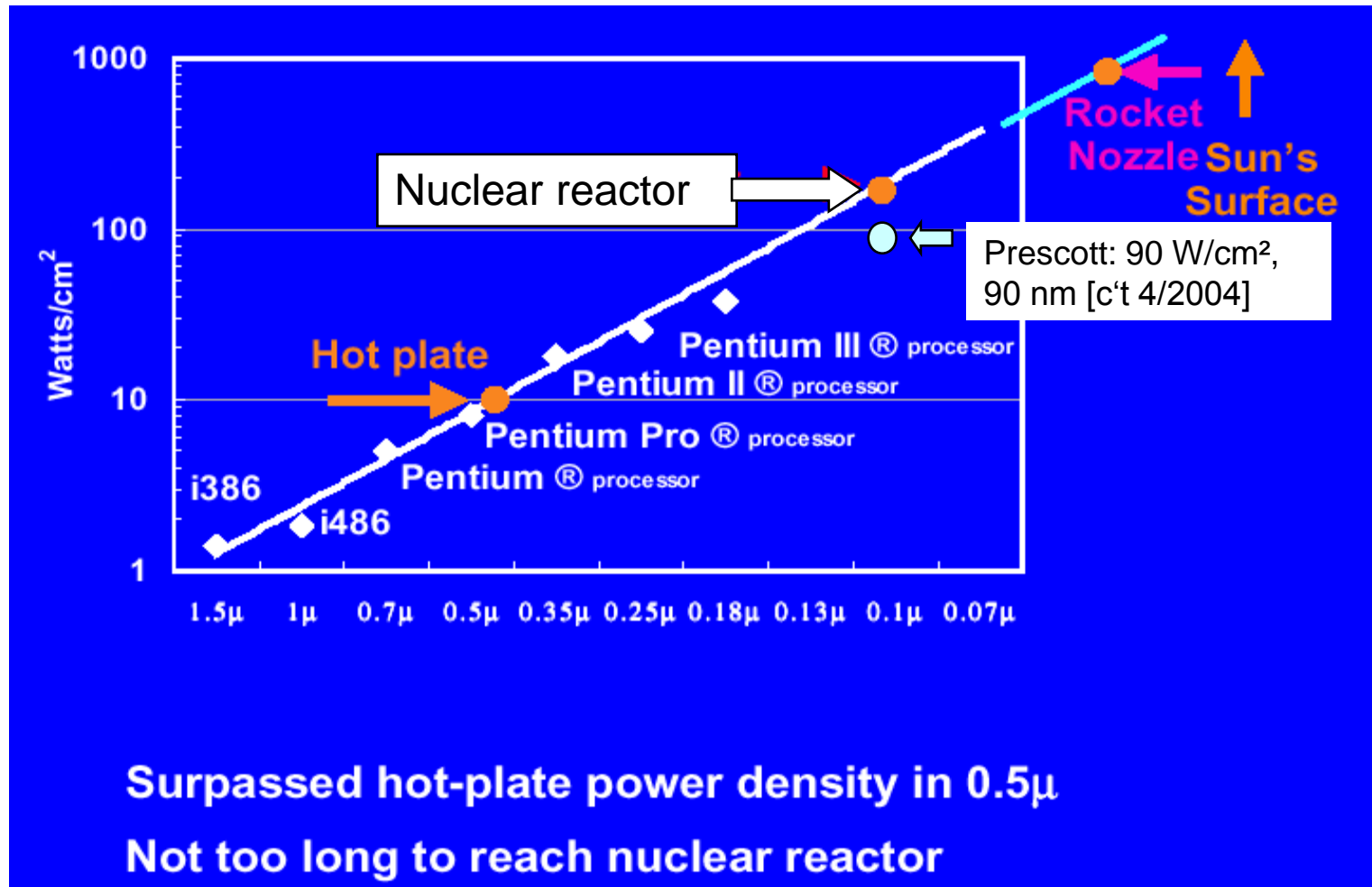
👉 HW synthesis not covered in this course, let's look at processors

Energy Efficiency of different target platforms



© Hugo De Man,
IMEC, Philips, 2007

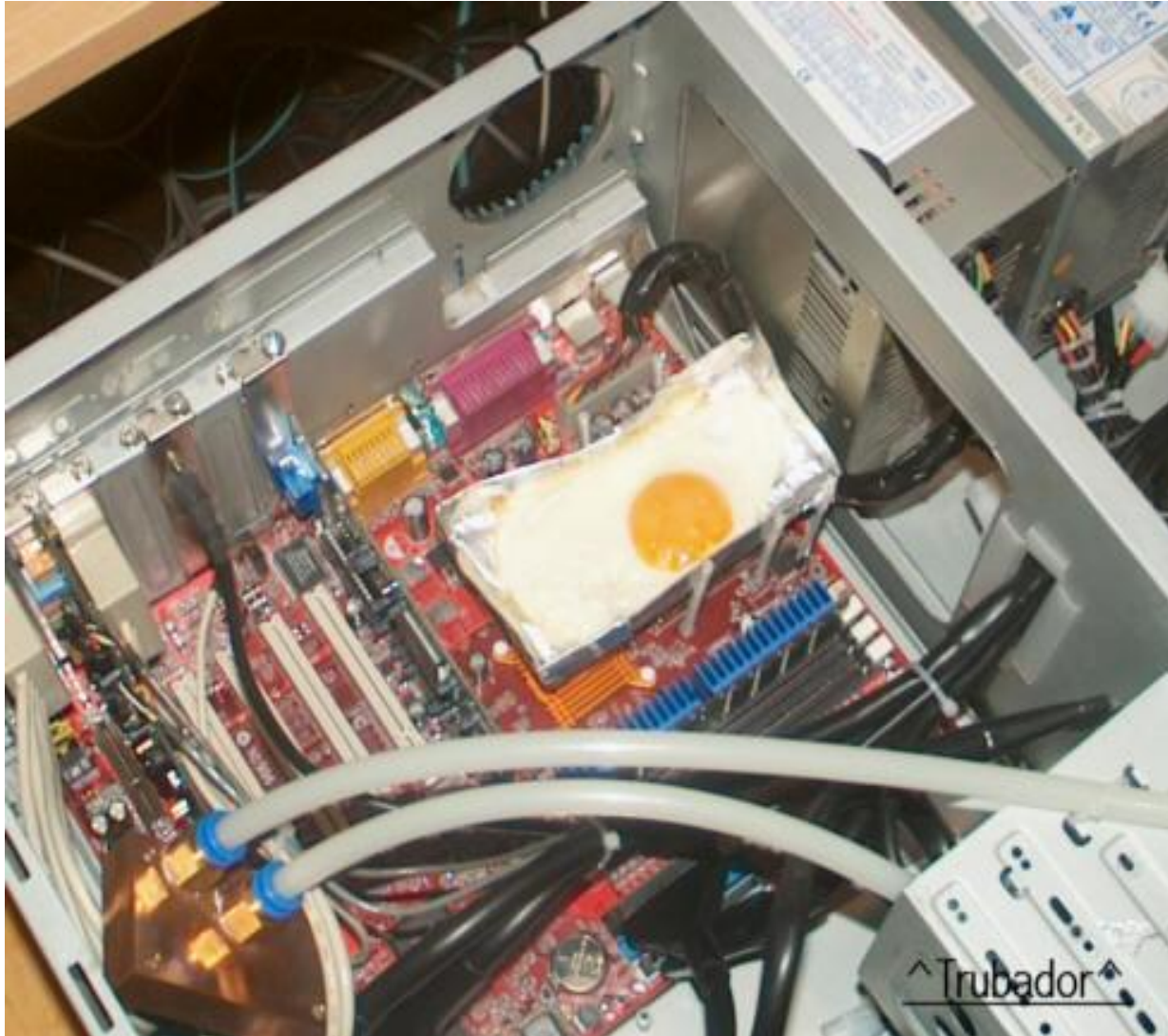
PCs: Problem: Power density increasing



© Intel
M. Pollack,
Micro-32

PCs: Surpassed hot (kitchen) plate ...?

Why not use it?

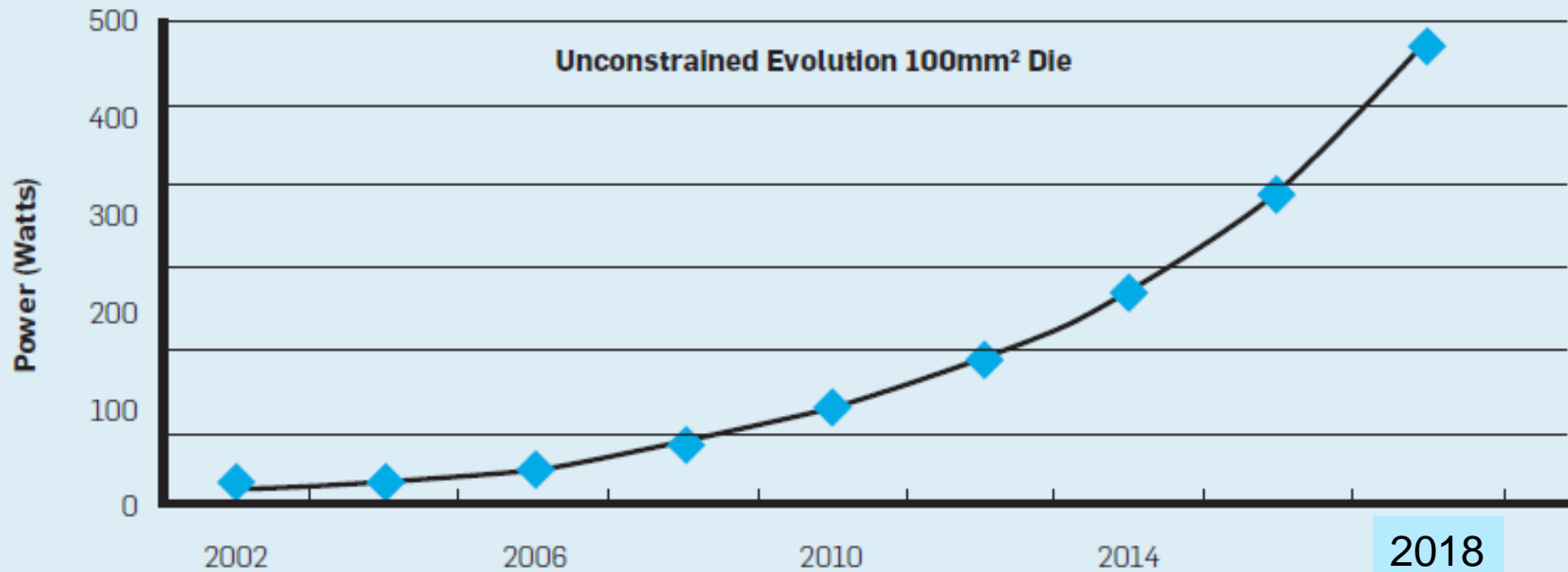


Strictly speaking, energy is not “consumed”, but converted from electrical energy into heat energy

http://www.phys.ncku.edu.tw/~htsu/humor/fry_egg.html

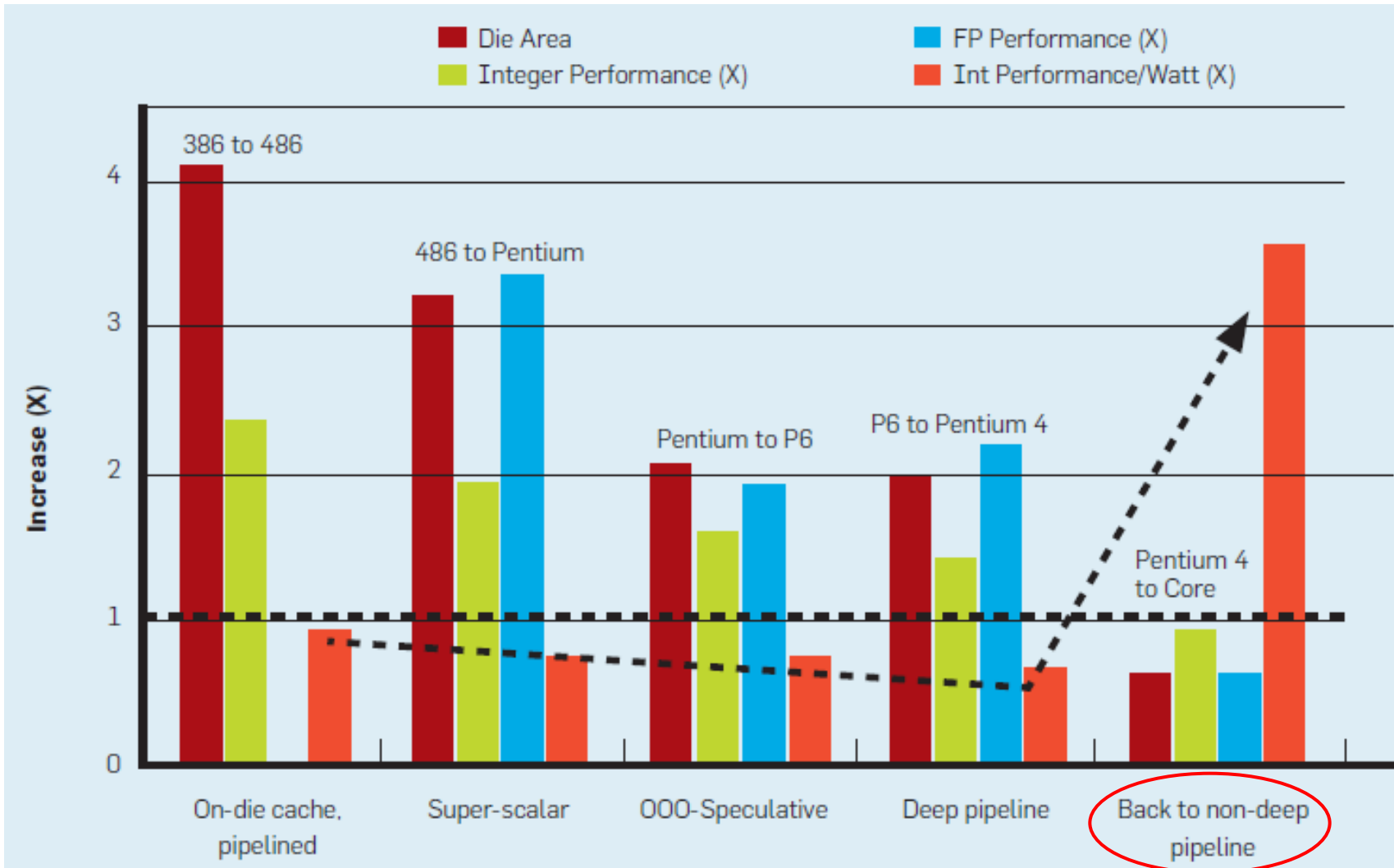
PCs: Just adding transistors would have resulted in this:

Figure 7. Unconstrained evolution of a microprocessor results in excessive power consumption.



S. Borkar, A. Chien: The future of microprocessors, *Communications of the ACM*, May 2011

Keep it simple, stupid (KISS)

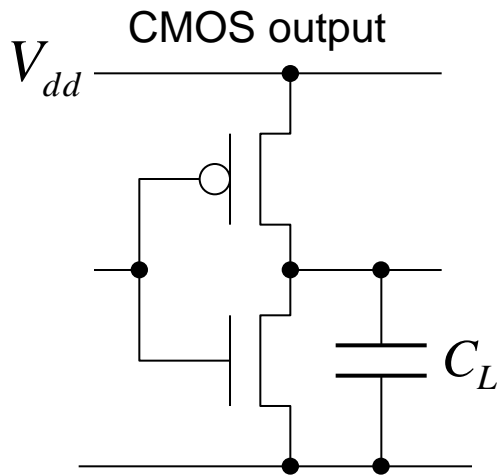


S. Borkar, A. Chien: The future of microprocessors, *Communications of the ACM*, May 2011

Prerequisite:

Static and dynamic power consumption

- Dynamic power consumption: Power consumption caused by charging capacitors when logic levels are switched.



$$P = \alpha C_L V_{dd}^2 f \text{ with}$$

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

f : clock frequency

➡ Decreasing V_{dd} reduces P quadratically

- Static power consumption (caused by leakage current): power consumed in the absence of clock signals
- Leakage becoming more important due to smaller devices

How to make systems energy efficient: Fundamentals of dynamic voltage scaling (DVS)

Power consumption of CMOS circuits (ignoring leakage):

$$P = \alpha C_L V_{dd}^2 f \text{ with}$$

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

f : clock frequency

Delay for CMOS circuits:

$$\tau = k C_L \frac{V_{dd}}{(V_{dd} - V_t)^2} \text{ with}$$

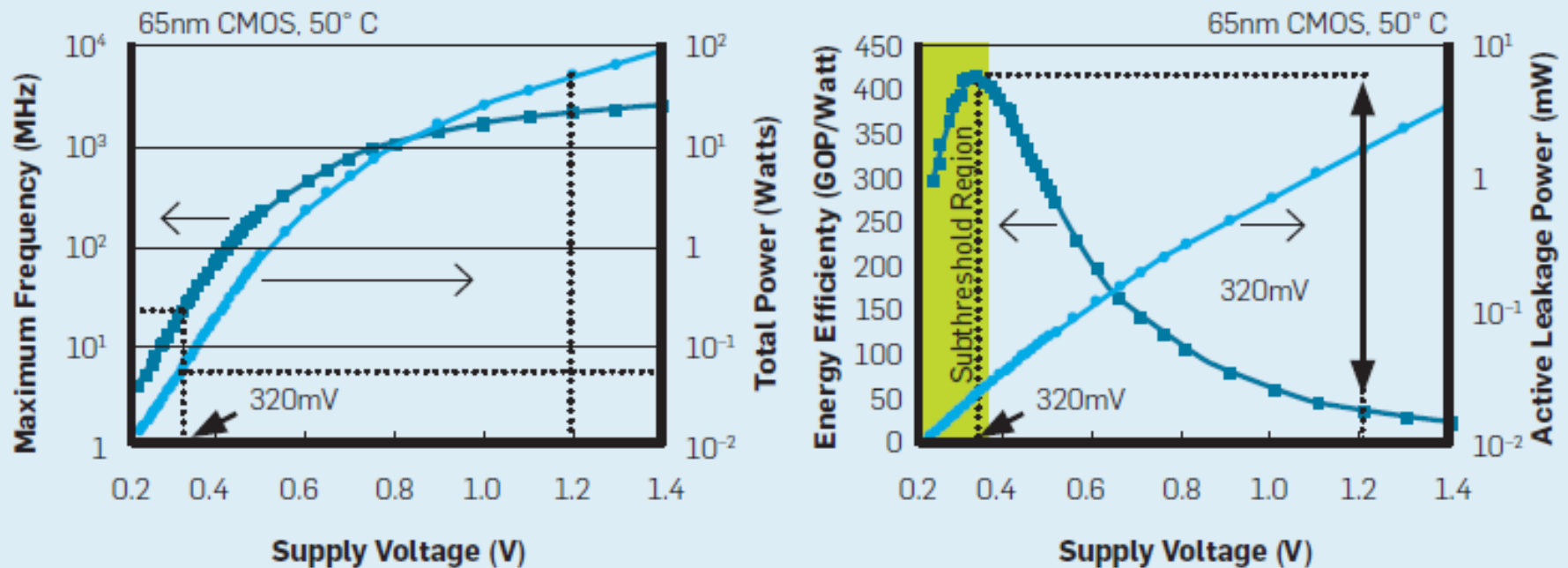
V_t : threshold voltage

($V_t < V_{dd}$)

☞ Decreasing V_{dd} reduces P quadratically,
while the run-time of algorithms is only linearly increased

Voltage scaling: Example

Figure 13. Improving energy efficiency through voltage scaling.

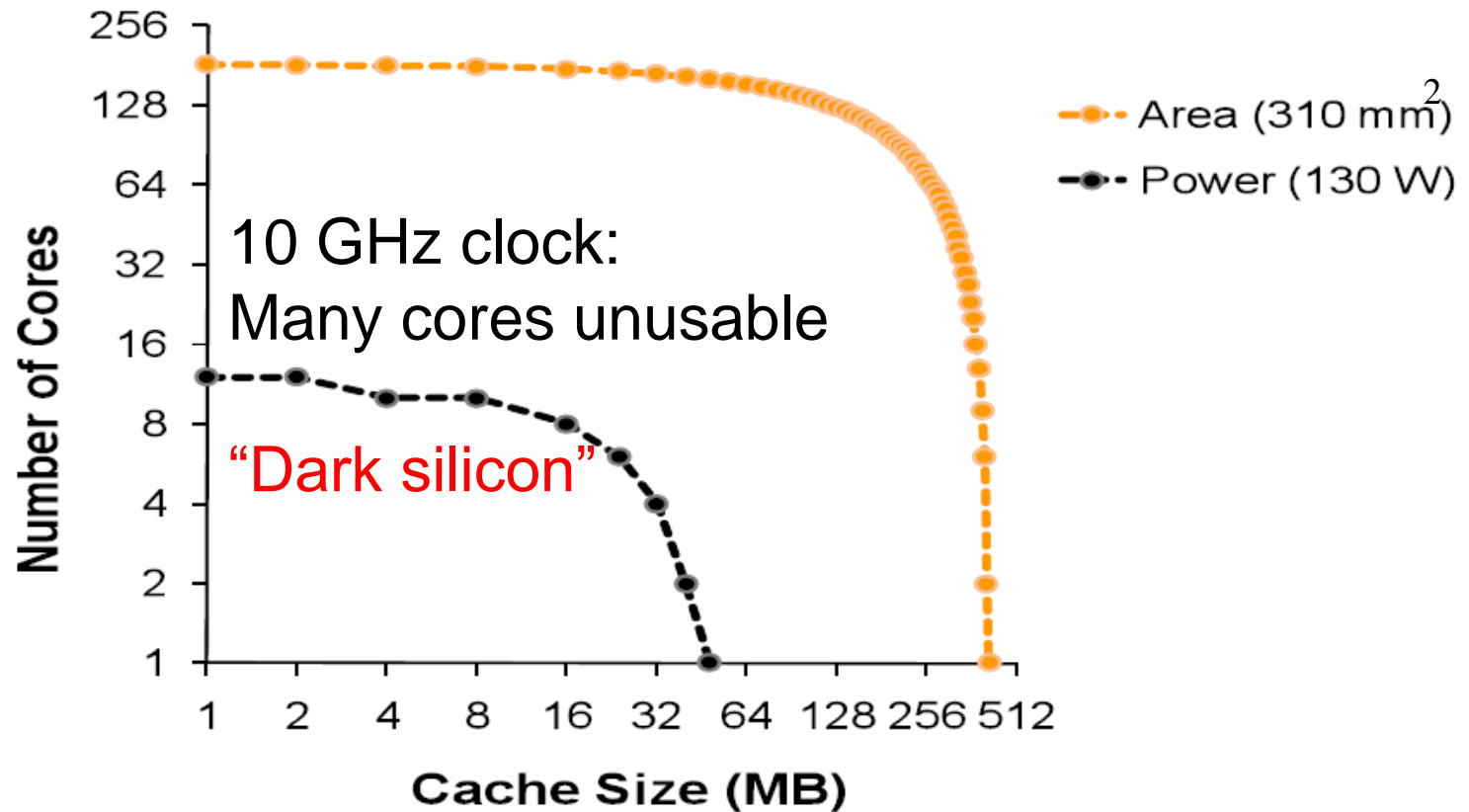


© ACM, 2011

S. Borkar, A. Chien: The future of microprocessors, *Communications of the ACM*, May 2011

Area vs. Power Envelope (22nm)

(For
servers)



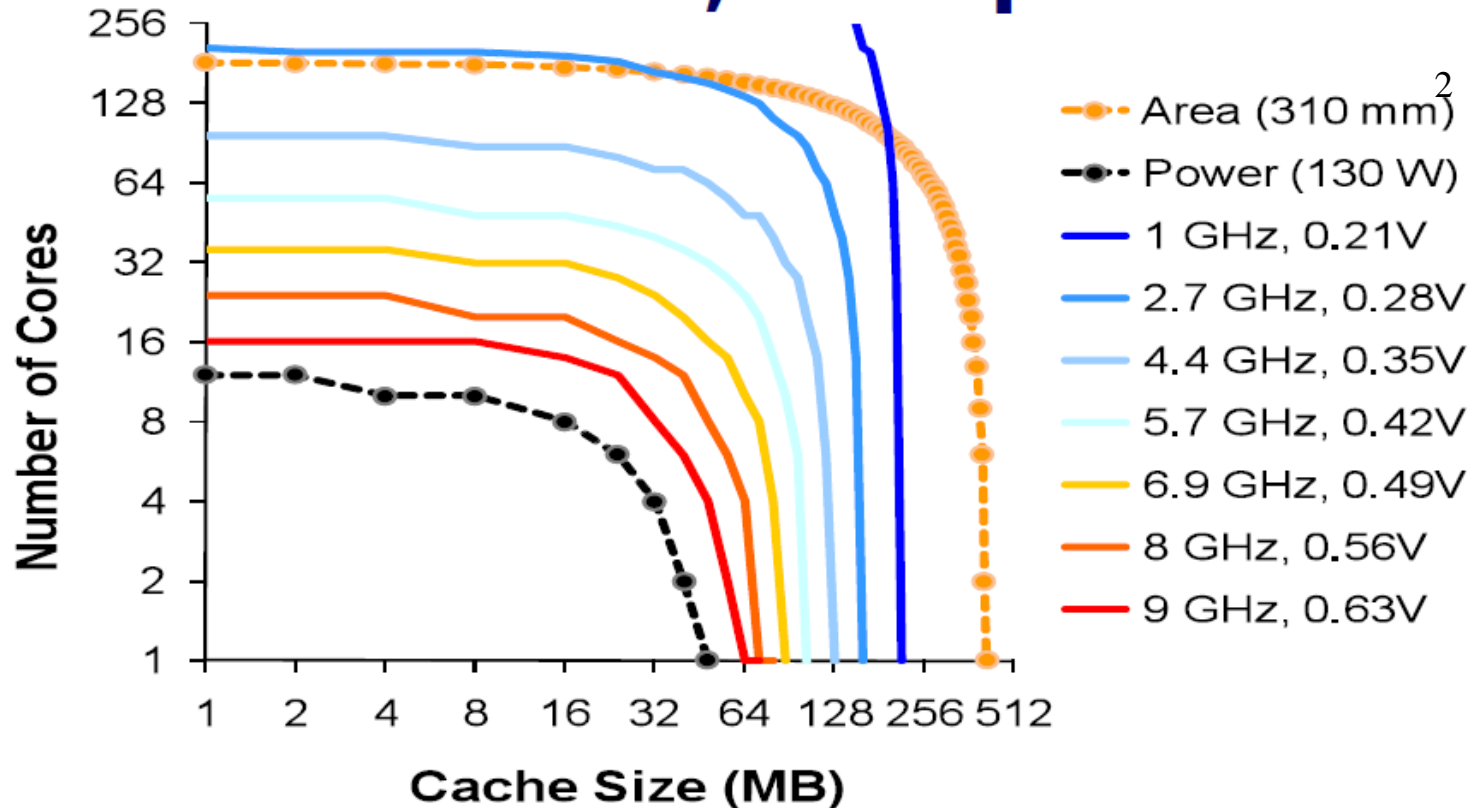
✓ Good news: can fit hundreds of cores

✗ Can not use them all at highest speed

© 2010 Babak Falsafi

Of course one could pack more slower cores, cheaper cache

(For servers)



- Result: a performance/power trade-off
- Assuming bandwidth is unlimited

© 2010 Babak Falsafi

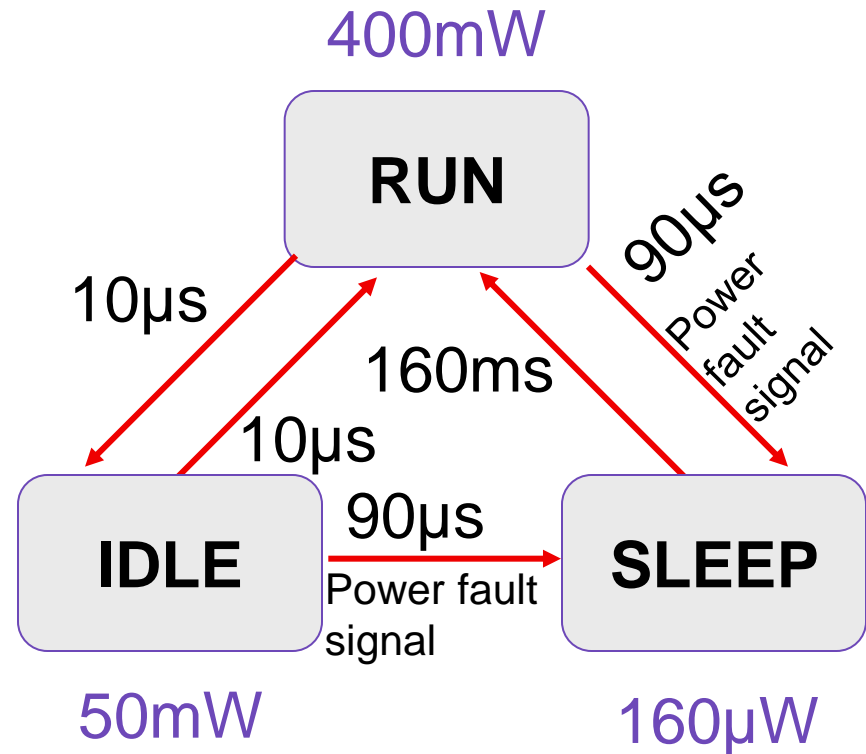
Dynamic power management (DPM)

Example: STRONGARM SA1100

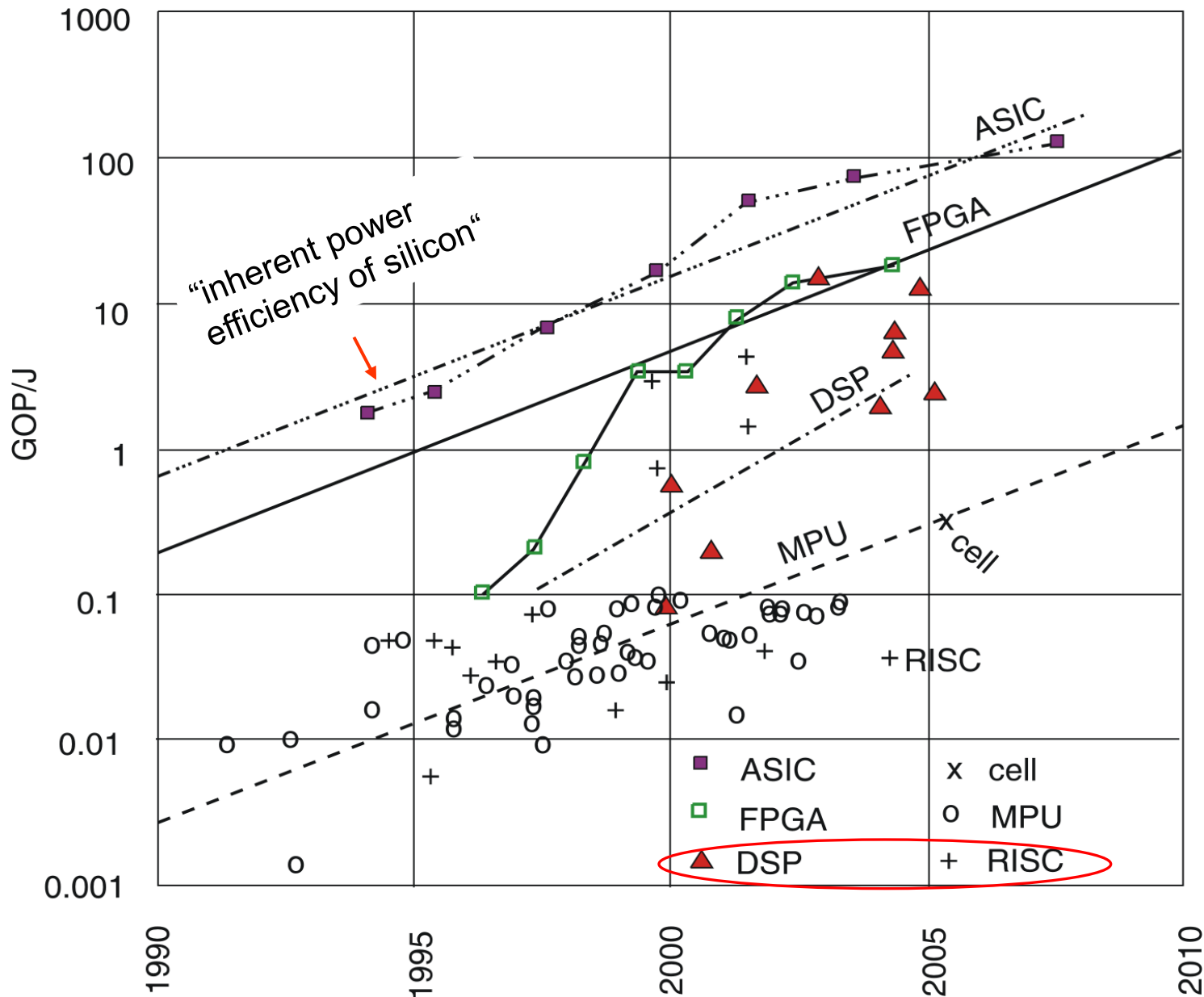
RUN: operational

IDLE: a SW routine may stop the CPU when not in use, while monitoring interrupts

SLEEP: Shutdown of on-chip activity



Energy Efficiency of different target platforms



© Hugo De Man,
IMEC, Philips, 2007

Low voltage, parallel operation more efficient than high voltage, sequential operation

Basic equations

Power:

$$P \sim V_{DD}^2,$$

Maximum clock frequency:

$$f \sim V_{DD},$$

Energy to run a program:

$$E = P \times t, \text{ with: } t = \text{runtime (fixed)}$$

Time to run a program:

$$t \sim 1/f$$

Changes due to parallel processing, with β operations per clock:

Clock frequency reduced to:

$$f' = f / \beta,$$

Voltage can be reduced to:

$$V_{DD}' = V_{DD} / \beta,$$

Power for parallel processing:

$$P^\circ = P / \beta^2 \text{ per operation,}$$

Power for β operations per clock:

$$P' = \beta \times P^\circ = P / \beta,$$

Time to run a program is still:

$$t' = t,$$

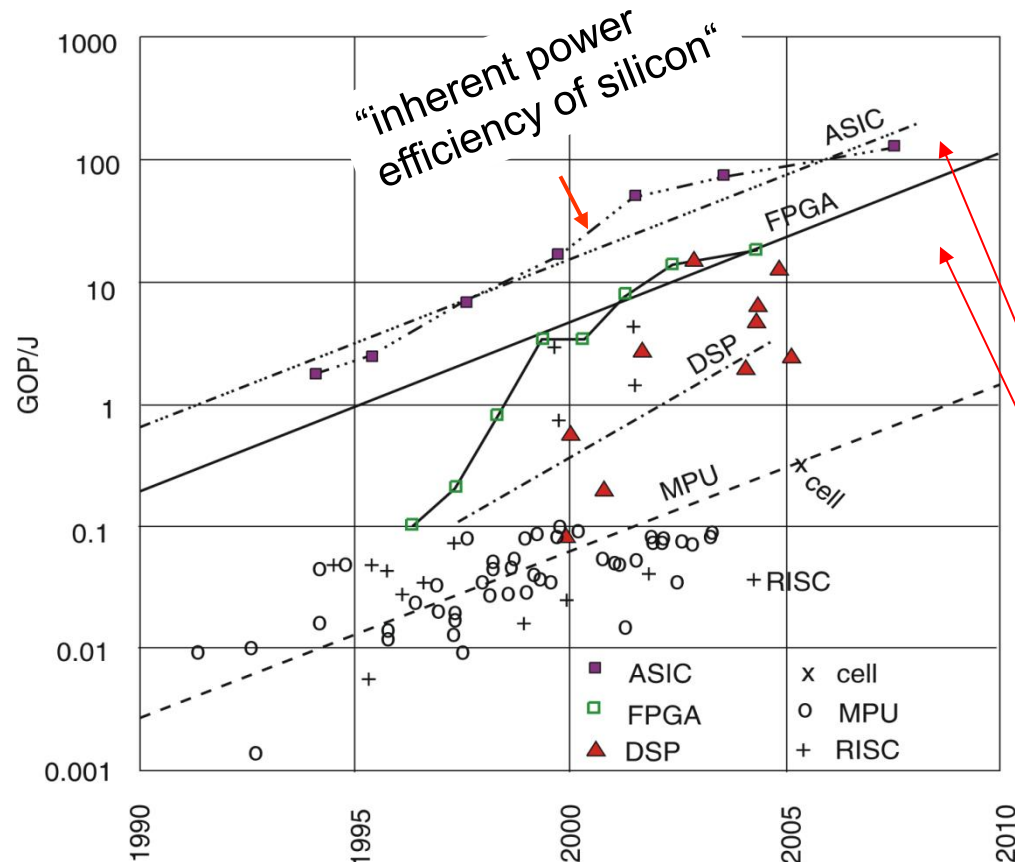
Energy required to run program:

$$E' = P' \times t = E / \beta$$

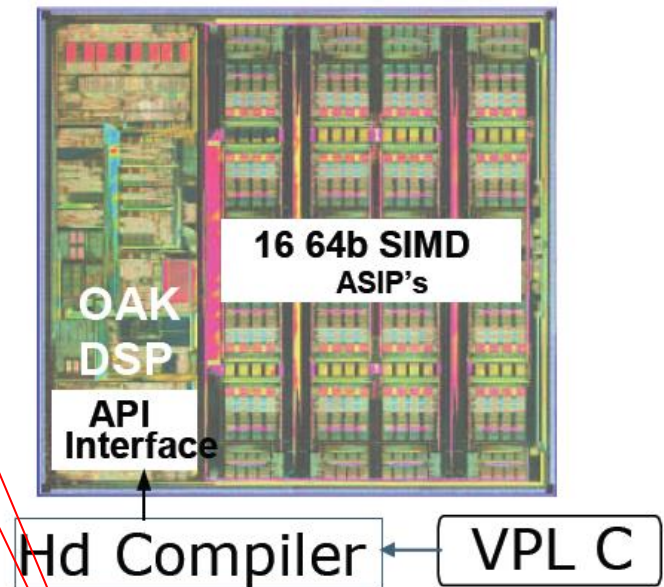
➡ Argument in favour of voltage scaling,
and parallel processing

Rough
approximations!

More energy-efficient architectures: Domain- and application specific



VIP for car mirrors Infineon

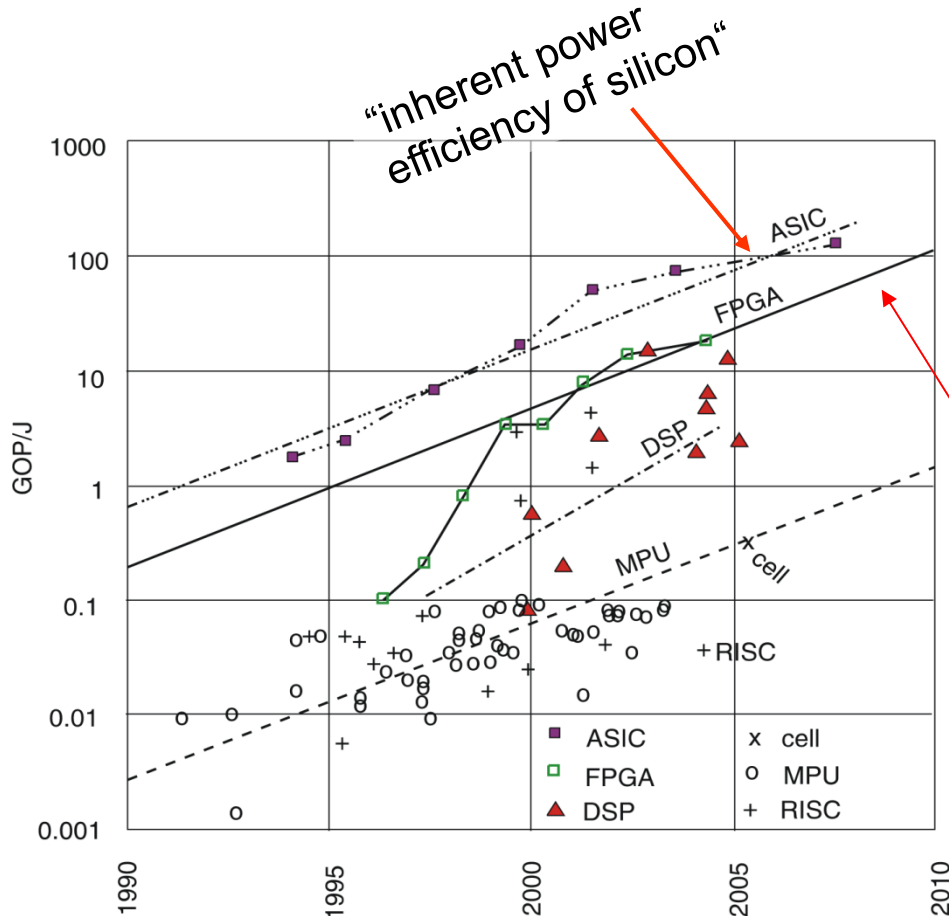


200MHz , 0.76 Watt
100Gops @ 8b
25Gops @ 32b

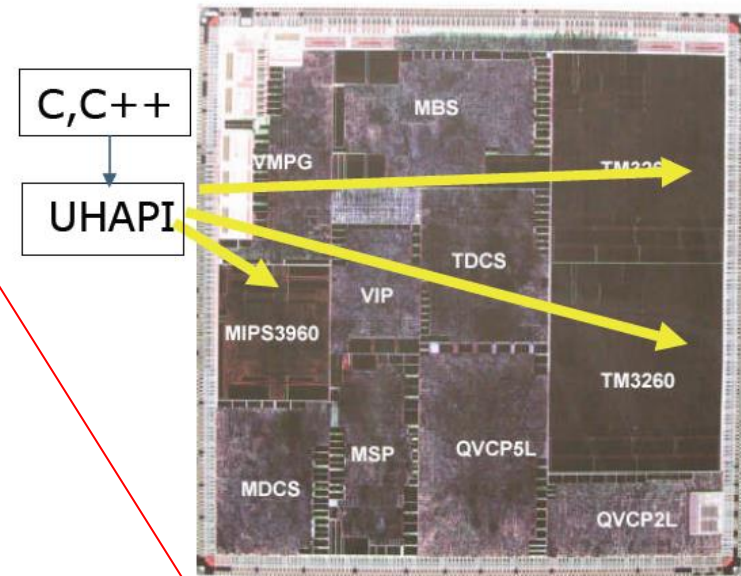
Close to power
efficiency of silicon

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

Energy-efficient architectures: Domain- and application specific



Nexperia Digital Video Platform NXP



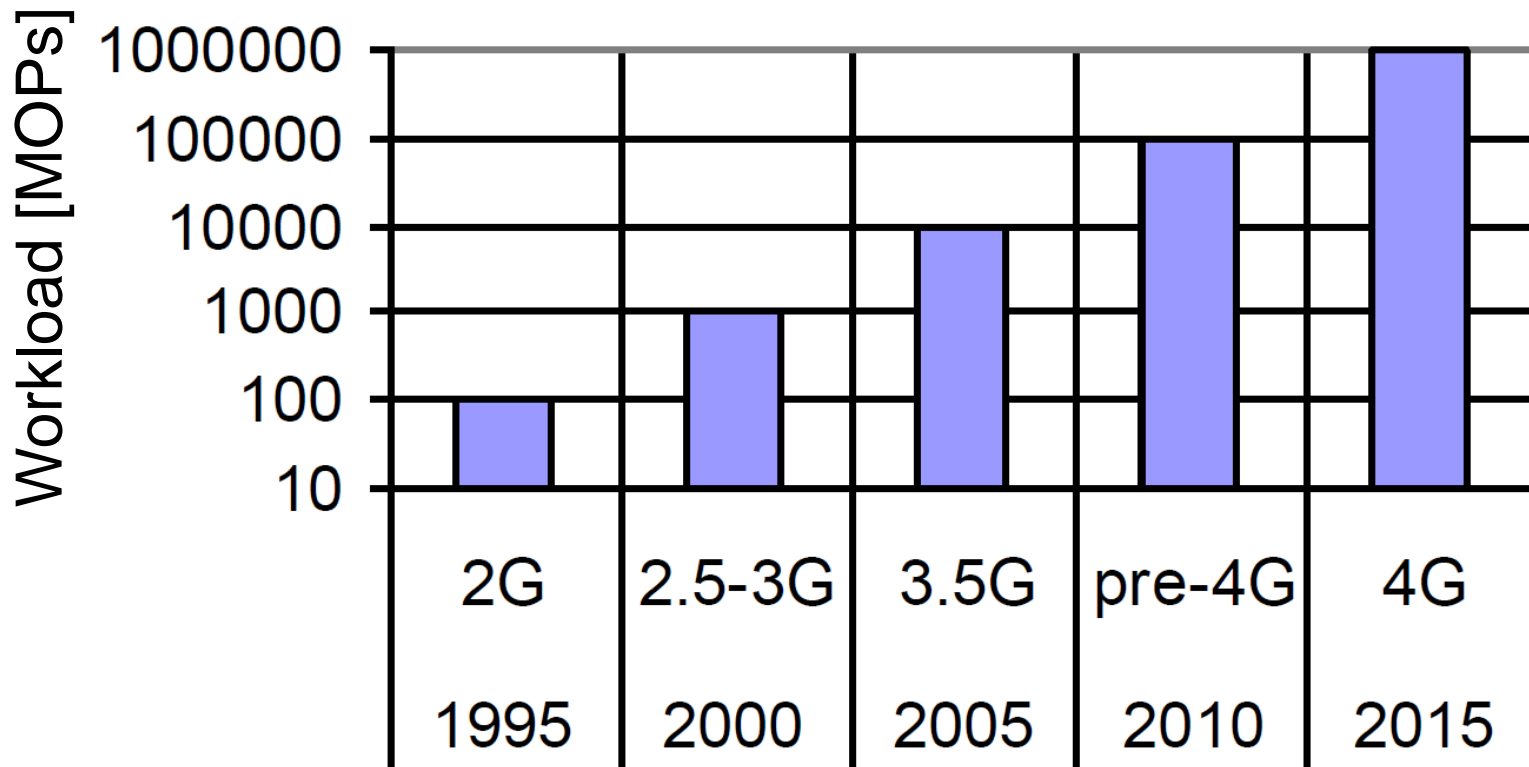
**1 MIPS, 2 Trimedia
60 coproc, 250 RAM's
266MHz, 1.5 watt 100 Gops**

Close to power
efficiency of silicon

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

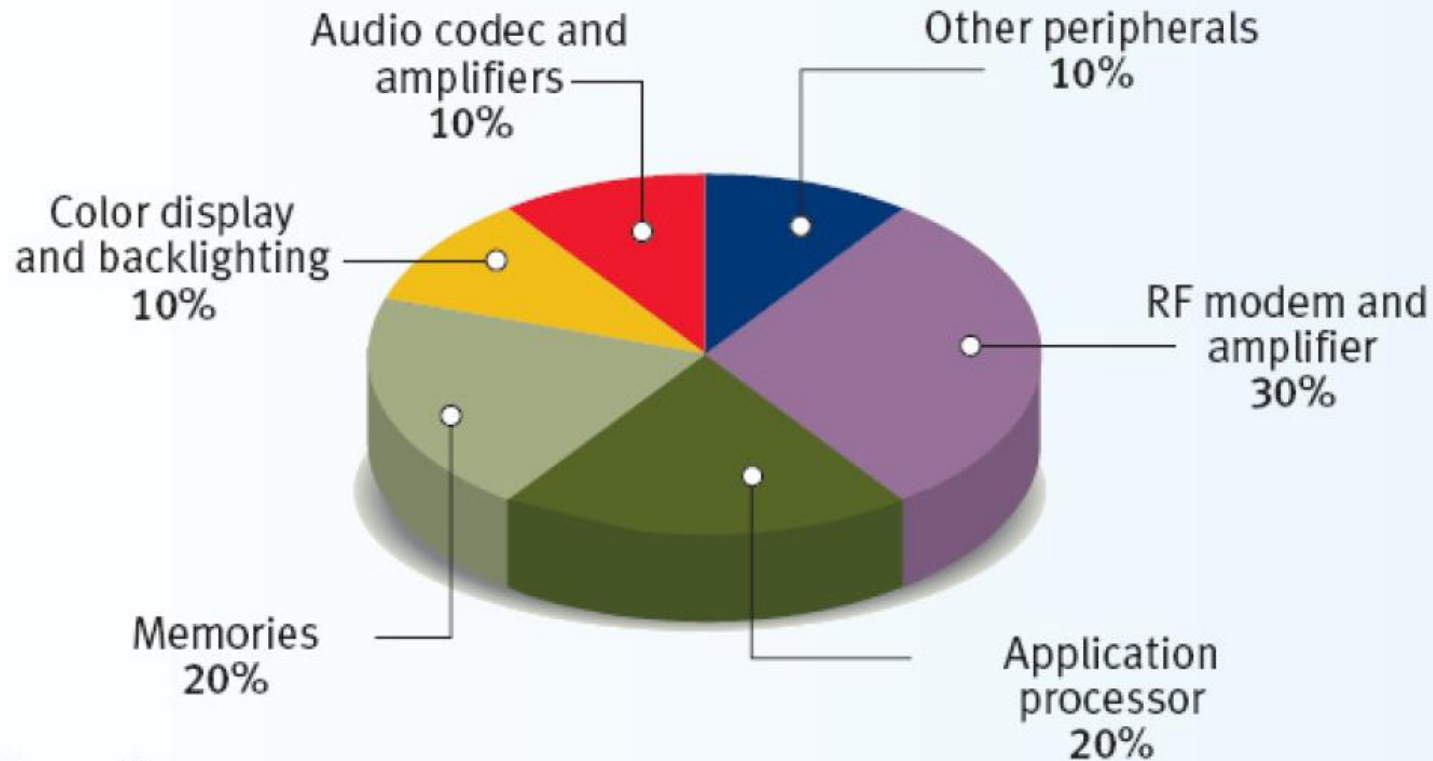
Mobile phones: Increasing performance requirements

C.H. van Berkel: Multi-Core for Mobile Phones, DATE, 2009;



Many more instances of the power/energy problem

Mobile phones: Where does the power go?



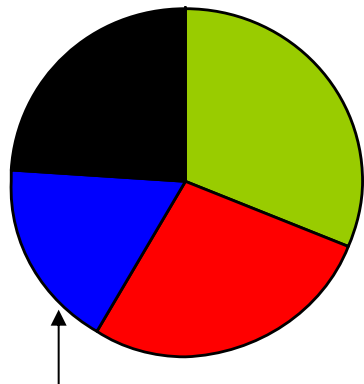
Source: Siemens

[O. Vargias: Minimum power consumption in mobile-phone memory subsystems; Pennwell Portable Design - September 2005;]

- **It not just I/O, don't ignore processing!**

Mobile phones: Where does the power go? (2)

Mobile phone use, breakdown by type of computation



With special purpose HW!

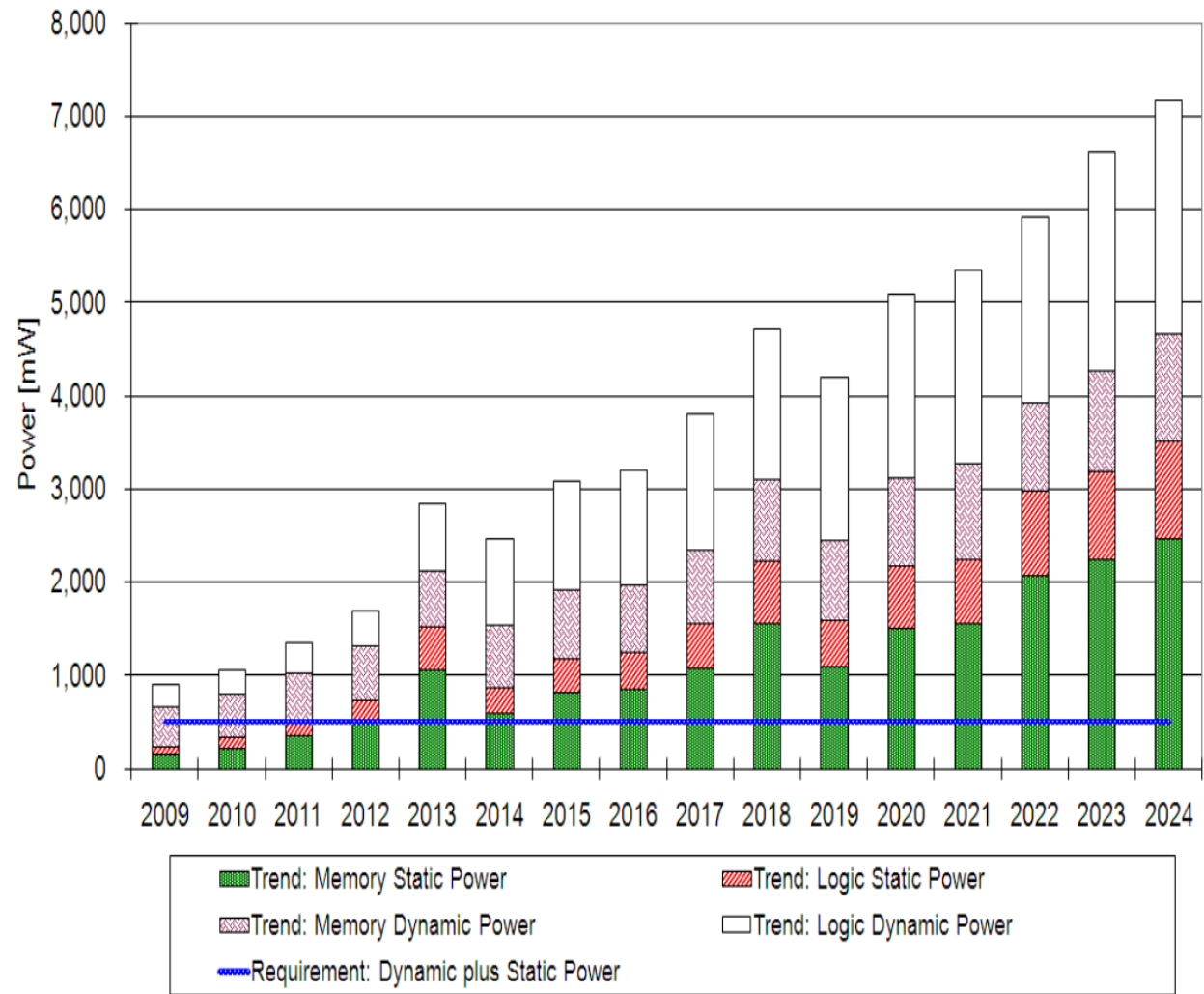
- Graphics (geometry processing, rasterization, pixel shading)
- Media (display & camera processing, video (de)coding)
- Radio (front-end, demodulation, decoding, protocol)
- Application (user interface, browsing, ...)

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

☞ During use, all components & computations relevant

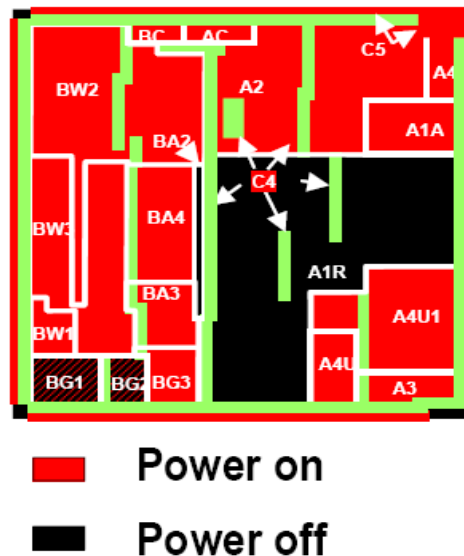
Mobile phones: Where is the energy consumed?

- According to *International Technology Roadmap for Semiconductors* (ITRS), 2010 update, [www.itrs.net]
- Current trends ➡ violation of 0.5-1 W constraint for small mobiles; large mobiles: ~ 7 W



Energy-efficient architectures: Heterogeneous processors

(2)Telephony (W-CDMA)



Baseband part	Control	ON
	W-CDMA	ON
	GSM	ON / OFF
Application part	System-domain	ON
	Realtime-domain	OFF
Measured Leakage Current (@ Room Temp, 1.2V)		407 μ A

<http://www.mpsoc-forum.org/2007/slides/Hattori.pdf>

☞ **“Dark silicon”** (not all silicon can be powered at the same time, due to current, power or temperature constraints)

ARM's big.LITTLE as an example

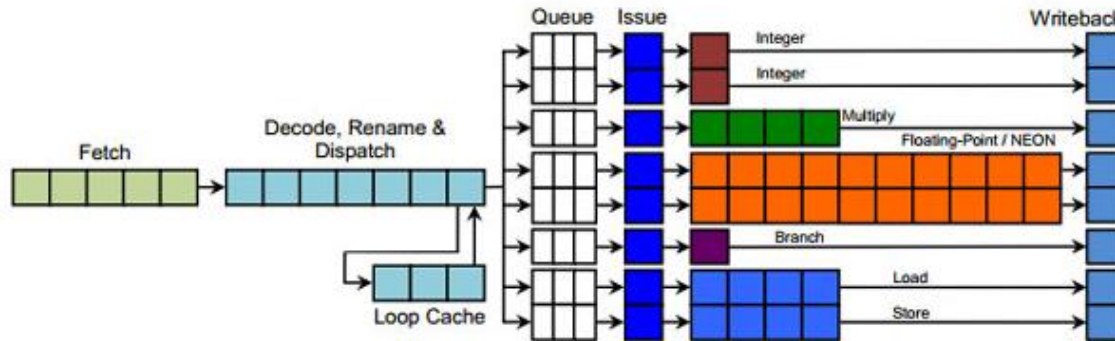


Figure 2 Cortex-A15 Pipeline

Used in
Samsung S4

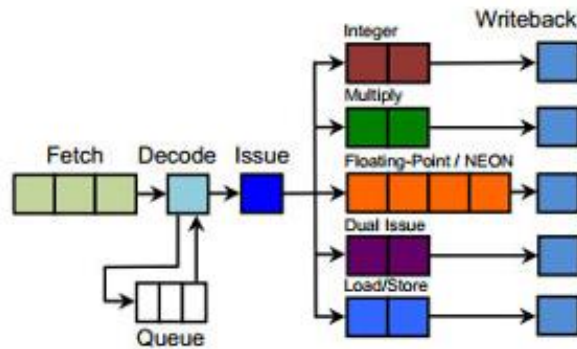
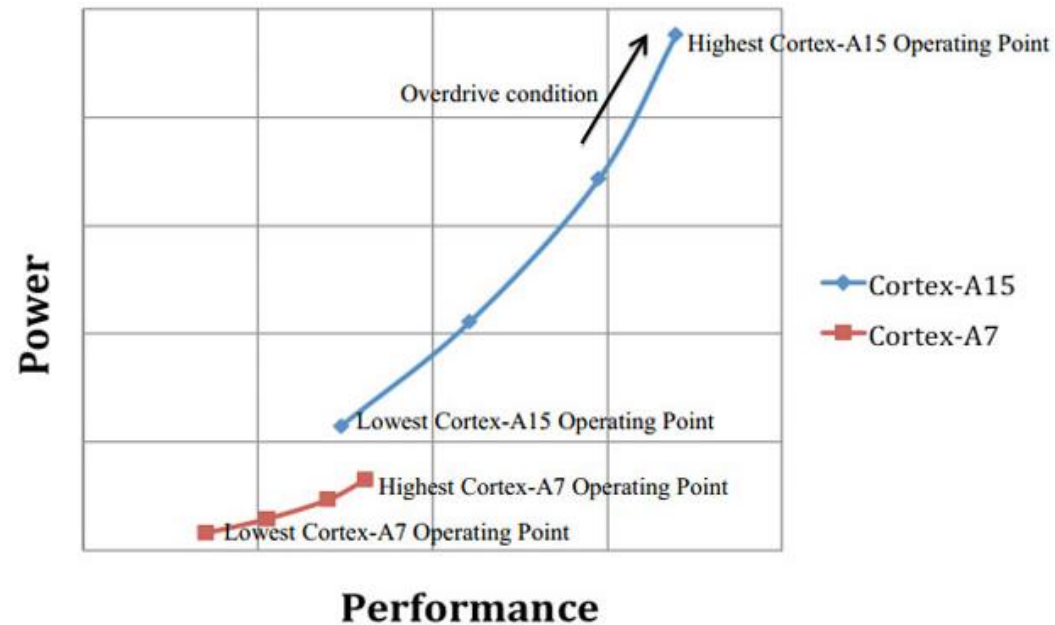
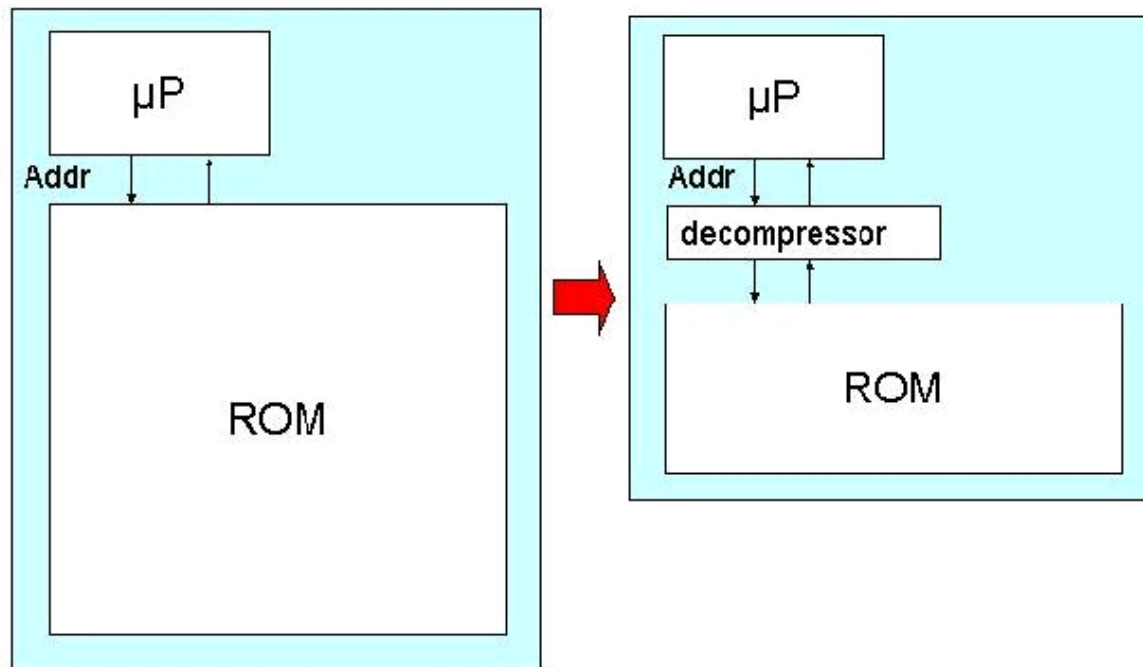


Figure 1 Cortex-A7 Pipeline



Key requirement #2: Code-size efficiency

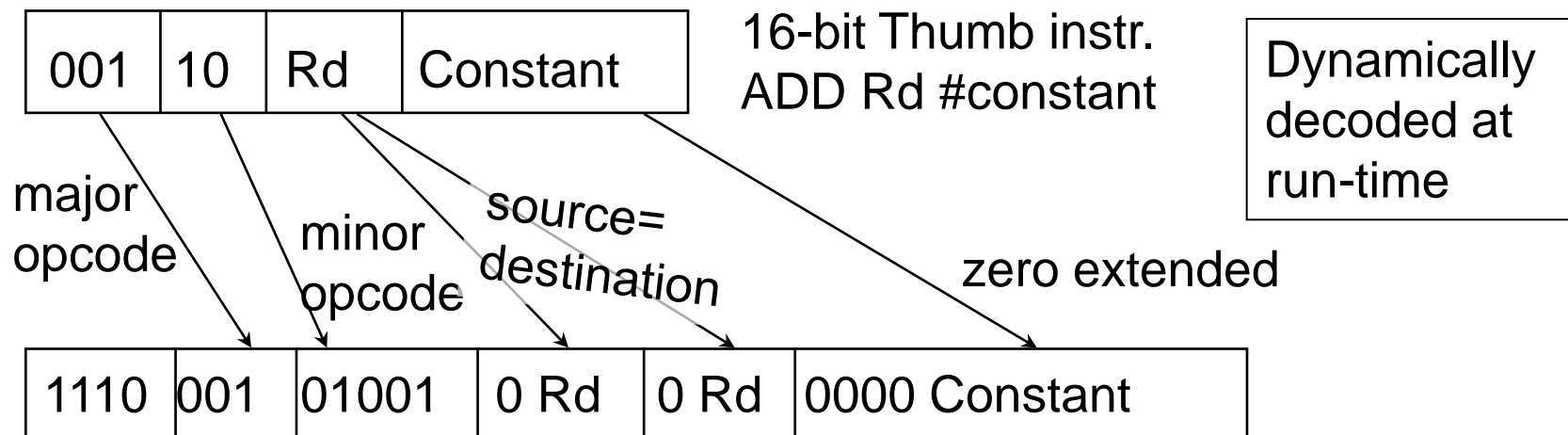
- Overview: <http://www-perso.iro.umontreal.ca/~latendre/codeCompression/codeCompression/node1.html>
- **Compression techniques:** key idea



Code-size efficiency

■ Compression techniques (continued):

- 2nd instruction set, e.g. ARM Thumb instruction set:



- Reduction to 65-70 % of original code size
- 130% of ARM performance with 8/16 bit memory
- 85% of ARM performance with 32-bit memory

Same approach for LSI TinyRisc, ...

Requires support by compiler, assembler etc.

Dictionary approach, two level control store (indirect addressing of instructions)

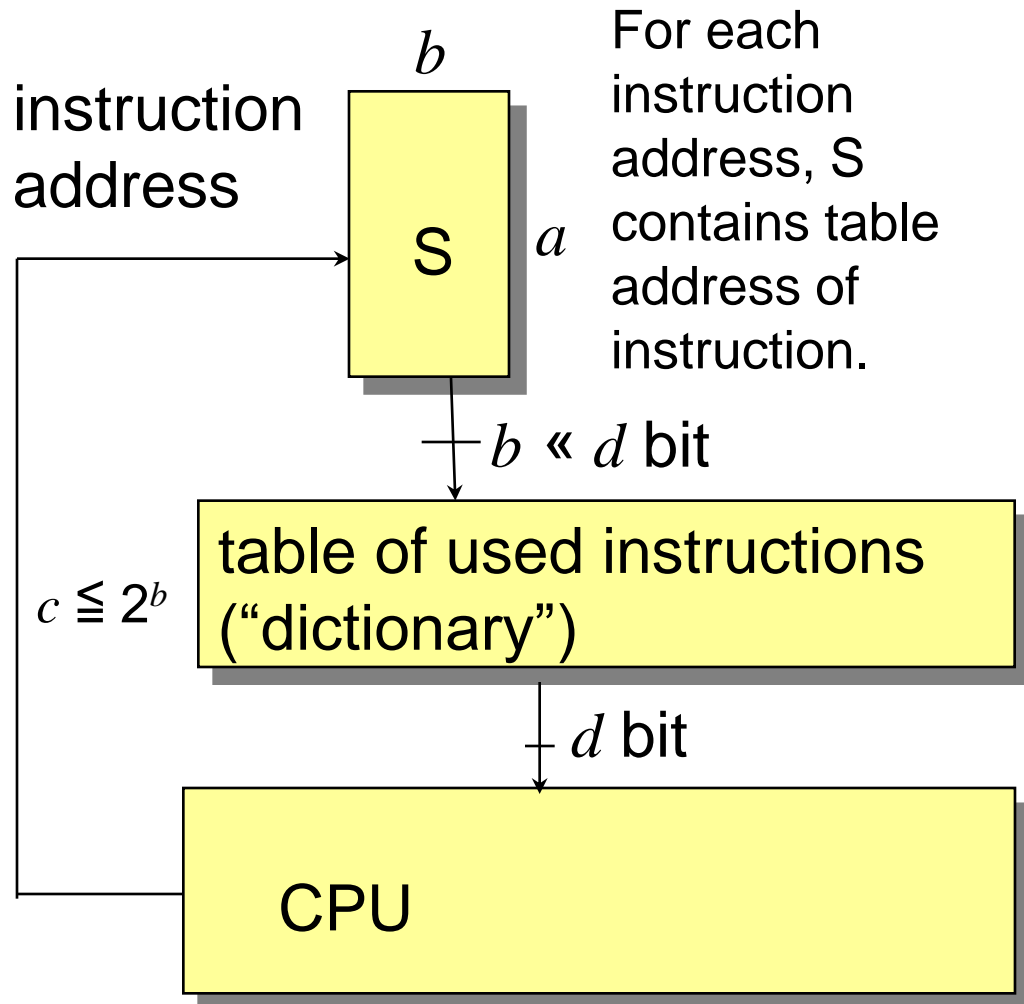
“Dictionary-based coding schemes cover a wide range of various coders and compressors.

Their common feature is that the methods use some kind of a dictionary that contains parts of the input sequence which frequently appear.

The encoded sequence in turn contains references to the dictionary elements rather than containing these over and over.”

[Á. Beszédés et al.: Survey of Code size Reduction Methods, Survey of Code-Size Reduction Methods, *ACM Computing Surveys*, Vol. 35, Sept. 2003, pp 223-267]

Key idea (for d bit instructions)



Uncompressed storage of a d -bit-wide instructions requires $a \times d$ bits.

In compressed code, each instruction pattern is stored only once.

Hopefully, $a \times b + c \times d < a \times d$.

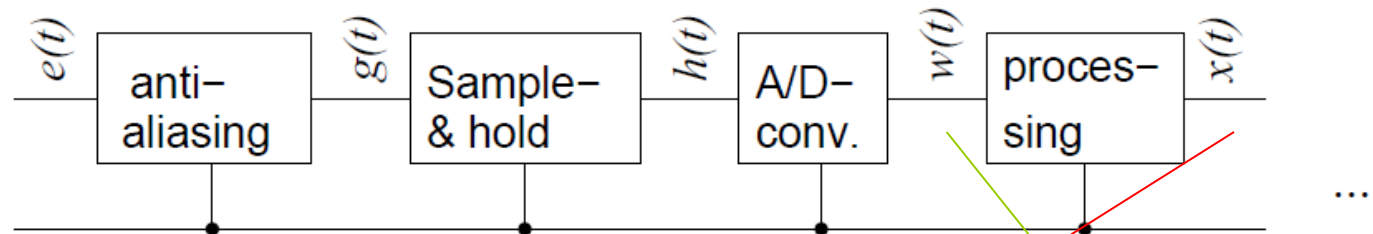
Called nanoprogramming in the Motorola 68000.

small

Key requirement #3: Run-time efficiency

- Domain-oriented architectures -

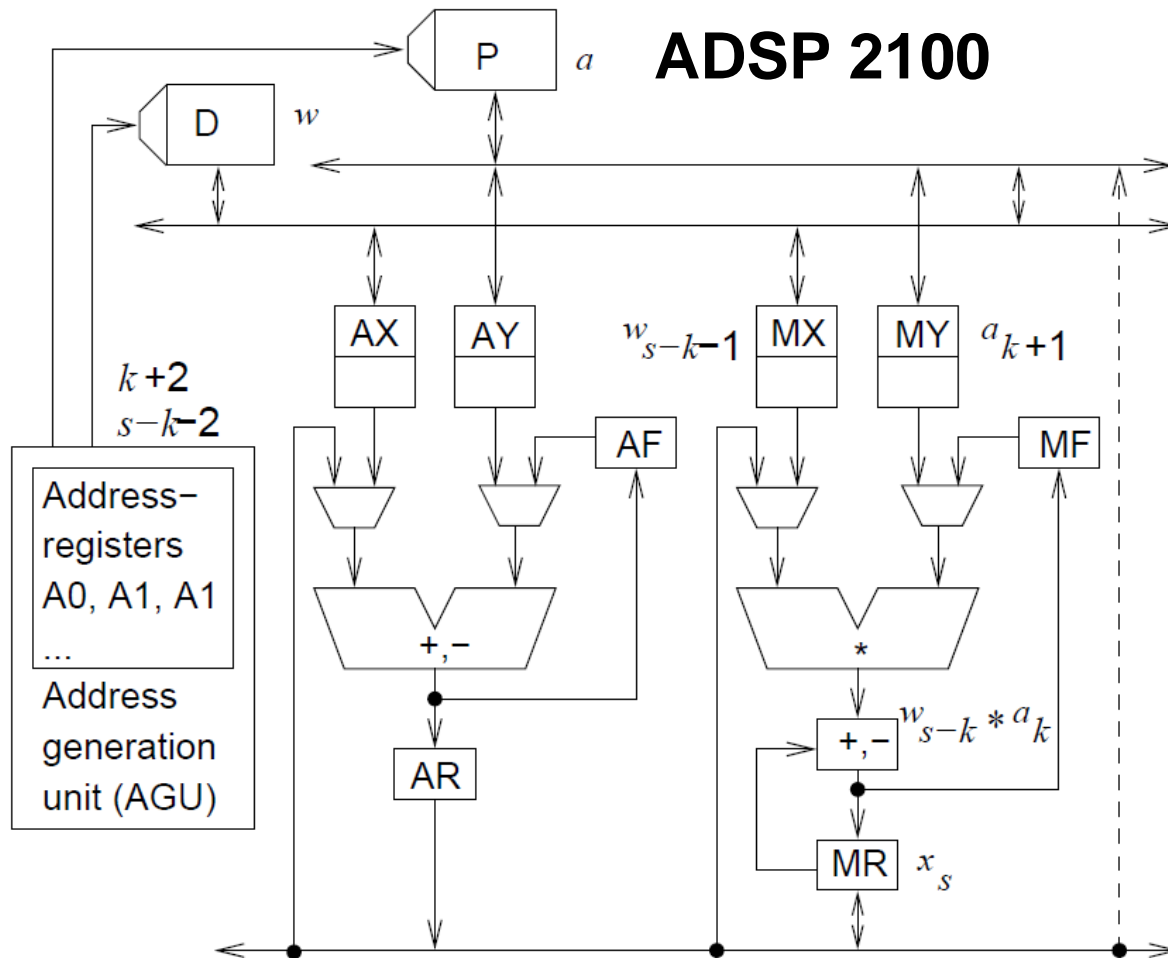
Example: Filtering in Digital signal processing (DSP)



$$x_s = \sum_{k=0}^{n-1} w_{s-k} * a_k$$

Signal at $t=t_s$ (sampling points)

Filtering in digital signal processing



$$x_s = \sum_{k=0}^{n-1} w_{s-k} * a_k$$

```
-- outer loop over
-- sampling times ts
{ MR:=0; A1:=1; A2:=s-1;
  MX:=w[s]; MY:=a[0];
  for (k=0; k <= (n-1); k++)
  { MR:=MR + MX * MY;
    MX:=w[A2]; MY:=a[A1];
    A1++; A2--;
  }
  x[s]:=MR;
}
```

Maps nicely

DSP-Processors: multiply/accumulate (MAC) and zero-overhead loop (ZOL) instructions

```
MR:=0; A1:=1; A2:=s-1; MX:=w[s]; MY:=a[0];
```

```
for ( k:=0 <= n-1)
```

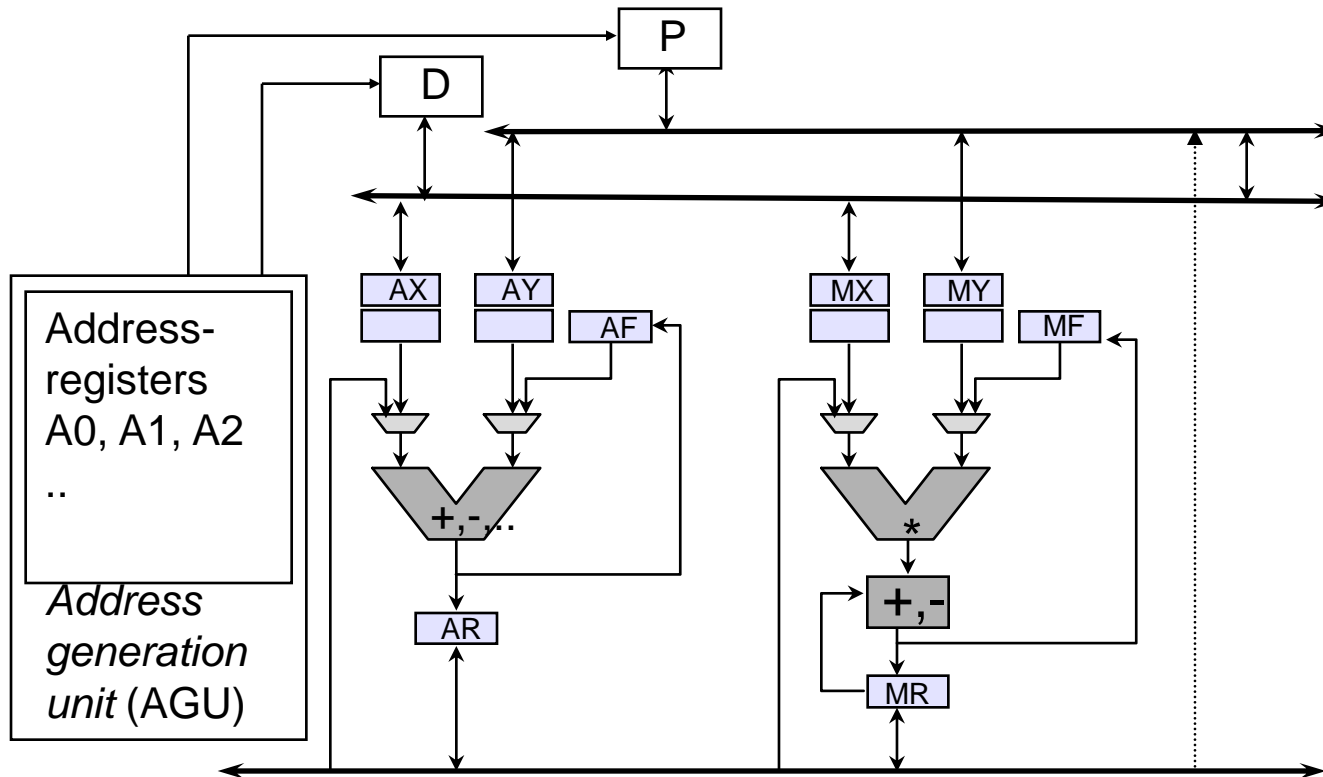
```
{MR:=MR+MX*MY; MY:=a[A1]; MX:=w[A2]; A1++; A2--}
```

Multiply/accumulate (MAC) instruction

Zero-overhead loop (ZOL) instruction preceding MAC instruction.
Loop testing done in parallel to MAC operations.

Heterogeneous registers

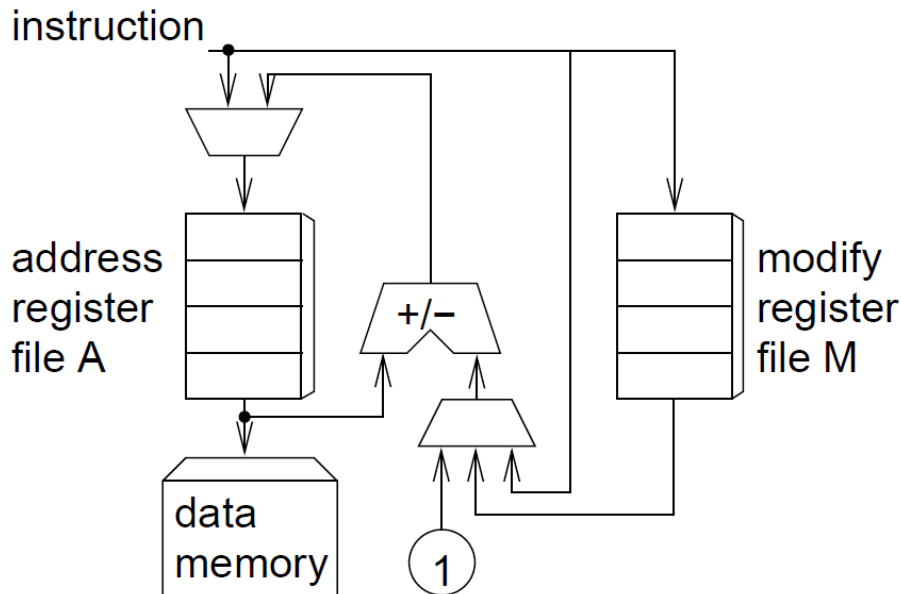
Example (ADSP 210x):



Different functionality of registers An, AX, AY, AF, MX, MY, MF, MR

Separate address generation units (AGUs)

Example (ADSP 210x):

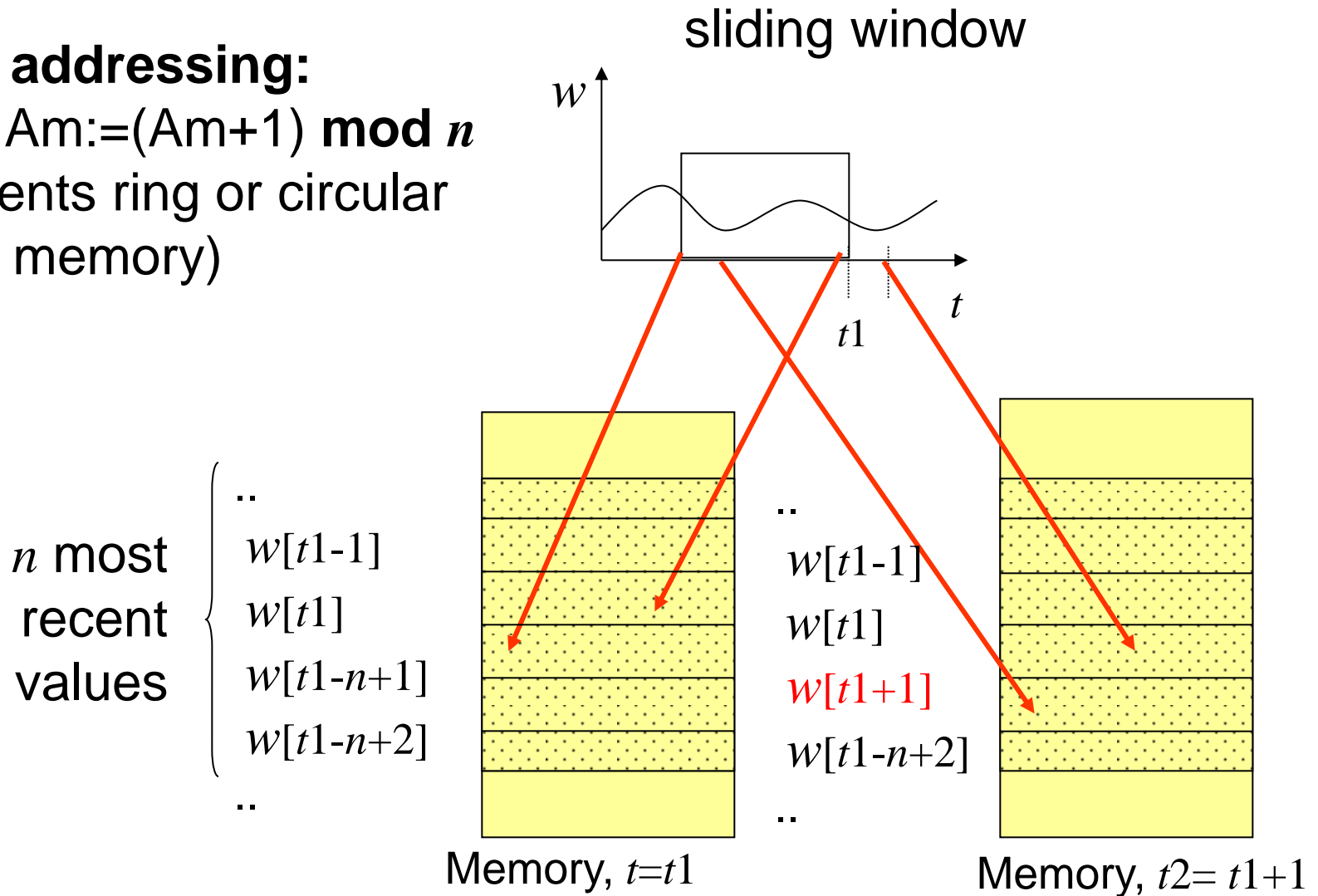


- Data memory can only be fetched with address contained in A,
 - but this can be done in parallel with operation in main data path (takes effectively 0 time).
 - $A := A \pm 1$ also takes 0 time,
 - same for $A := A \pm M$;
 - $A := \text{<immediate in instruction>}$ requires extra instruction
- ➡ Minimize load immediates
- ➡ Optimization in optimization chapter

Modulo addressing

Modulo addressing:

$A_{m++} \equiv A_m := (A_{m+1}) \bmod n$
(implements ring or circular buffer in memory)



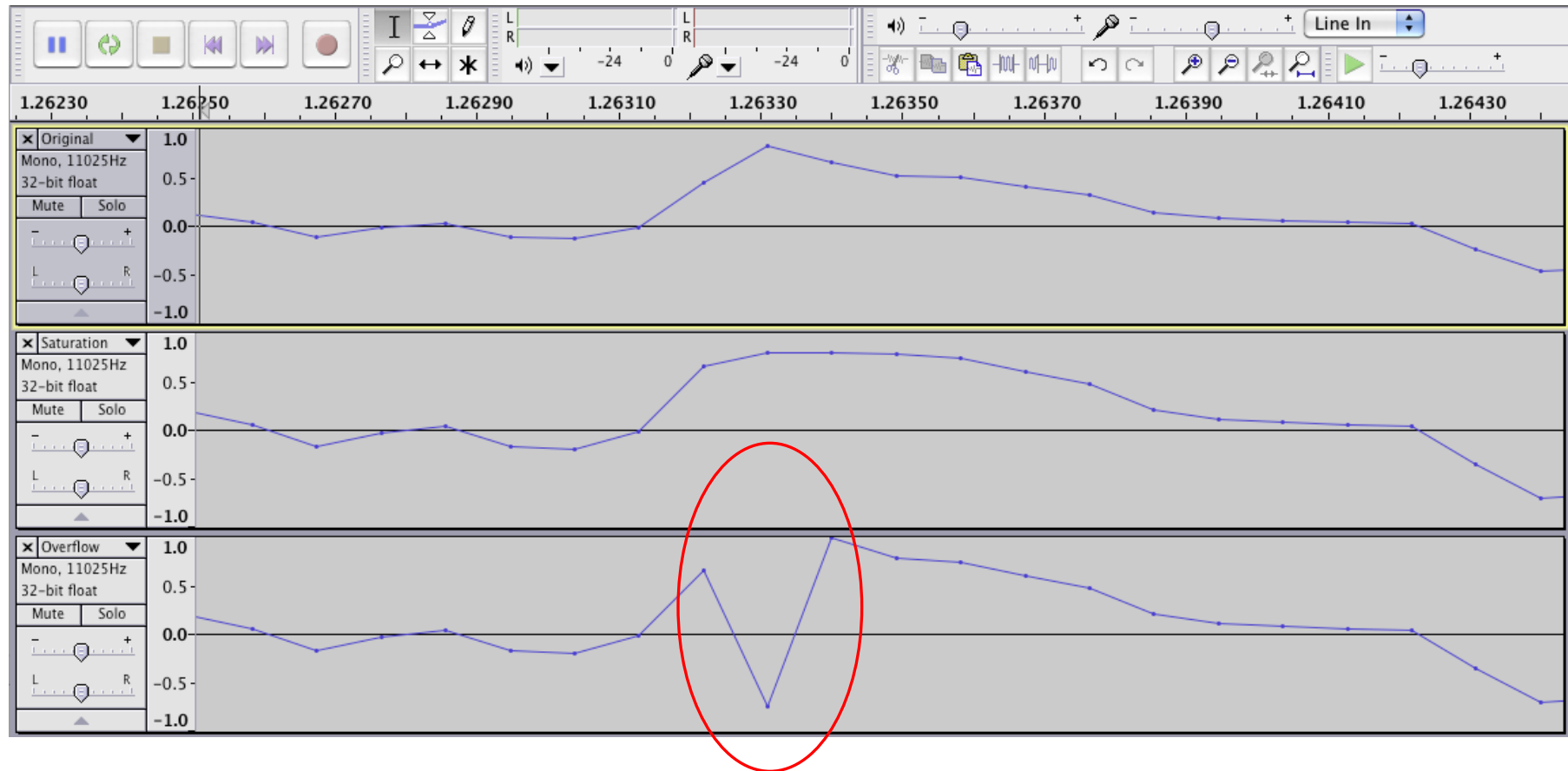
Saturating arithmetic

- Returns largest/smallest number in case of over/underflows
- Example:

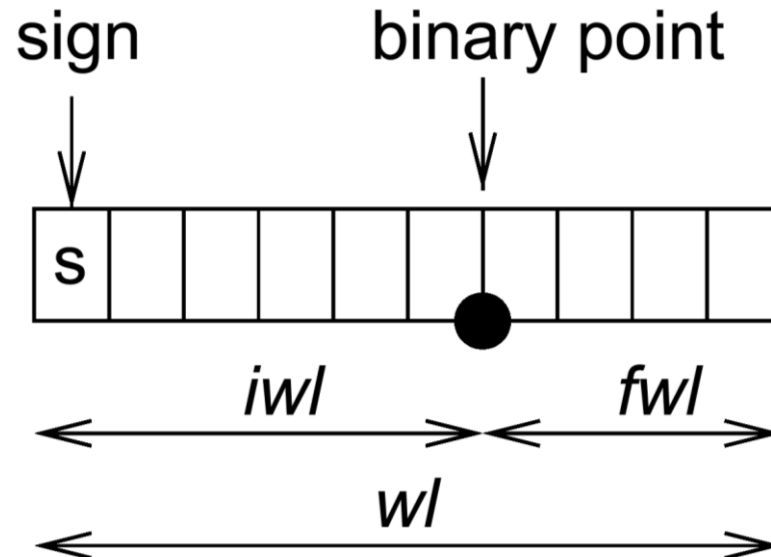
a		0111
b	+	1001
standard wrap around arithmetic		(1)0000
saturating arithmetic		1111
(a+b)/2: correct		1000
wrap around arithmetic		0000
saturating arithmetic + shifted		0111 “almost correct”

- Appropriate for DSP/multimedia applications:
 - No timeliness of results if interrupts are generated for overflows
 - Precise values less important
 - Wrap around arithmetic would be worse.

Example



Fixed-point arithmetic



Shifting required after multiplications and divisions in order to maintain binary point.

Real-time capability

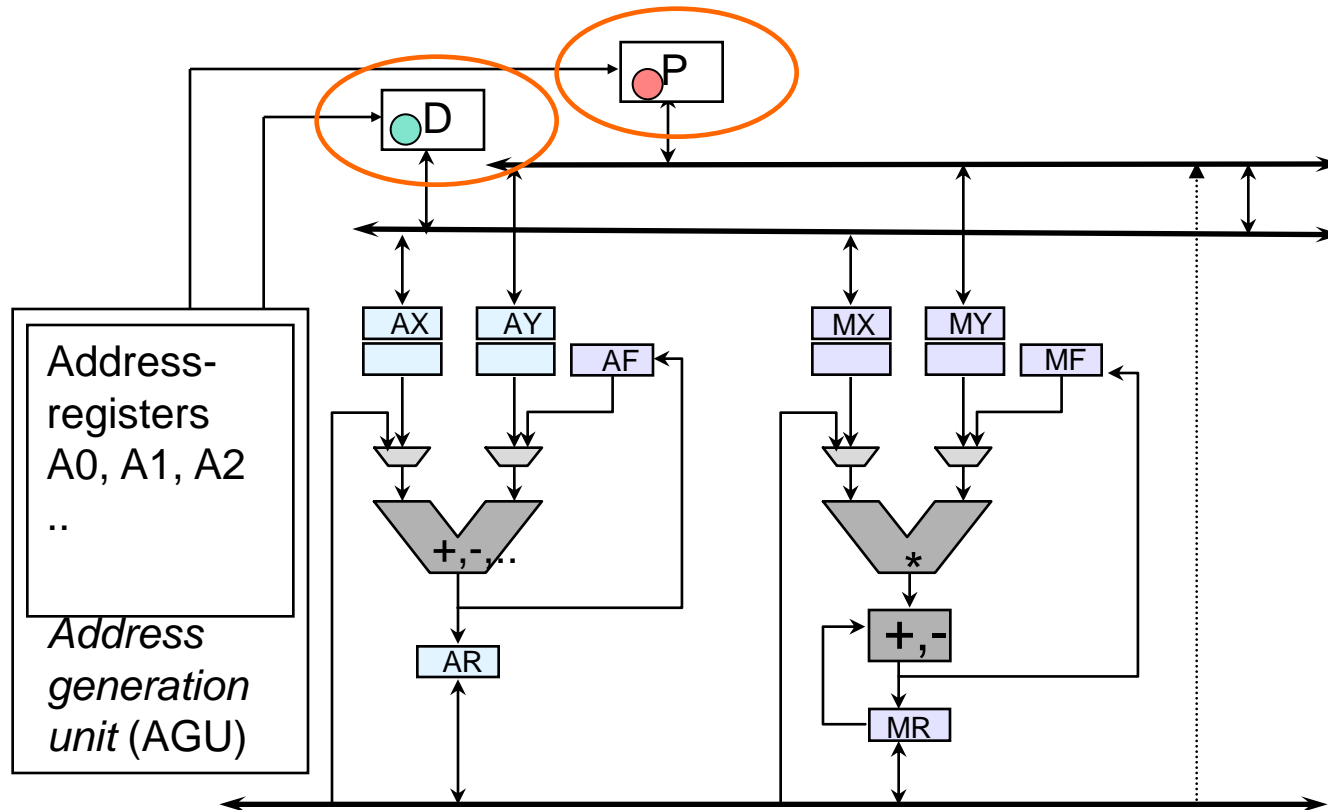
■ Timing behavior has to be predictable

Features that cause problems:

- Unpredictable access to shared resources
 - Caches with difficult to predict replacement strategies
 - Unified caches (conflicts between instructions and data)
 - Pipelines with difficult to predict stall cycles ("bubbles")
 - Unpredictable communication times for multiprocessors
- Branch prediction, speculative execution
- Interrupts that are possible any time
- Memory refreshes that are possible any time
- Instructions that have data-dependent execution times

👉 Trying to avoid as many of these as possible.

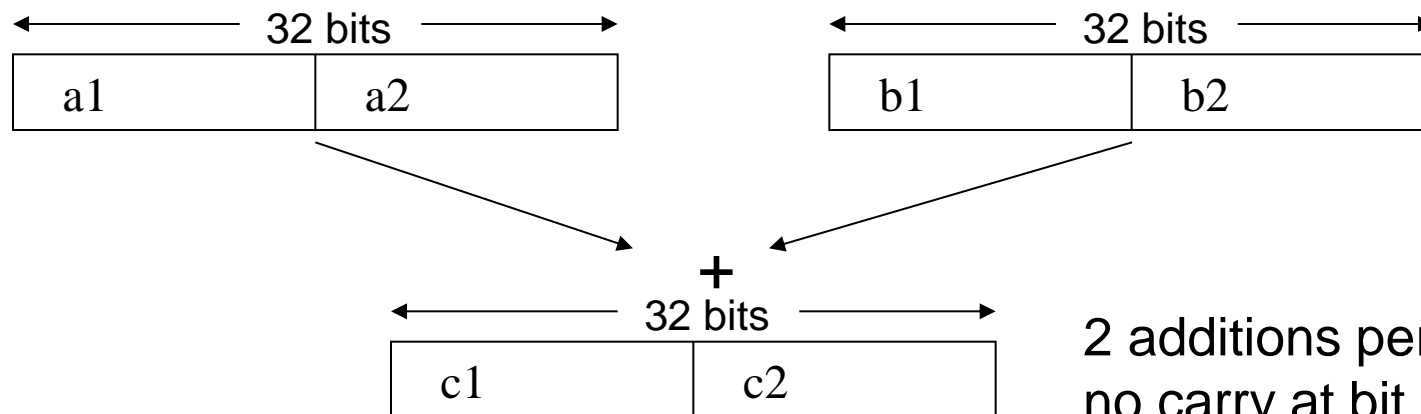
Multiple memory banks or memories



Simplifies parallel fetches

Multimedia-Instructions, Short vector extensions, Streaming extensions, SIMD instructions

- Multimedia instructions exploit that many registers, adders etc are quite wide (32/64 bit), whereas most multimedia data types are narrow
- 👉 2-8 values can be stored per register and added. E.g.:



2 additions per instruction;
no carry at bit 16

- Cheap way of using parallelism
- 👉 SSE instruction set extensions, SIMD instructions

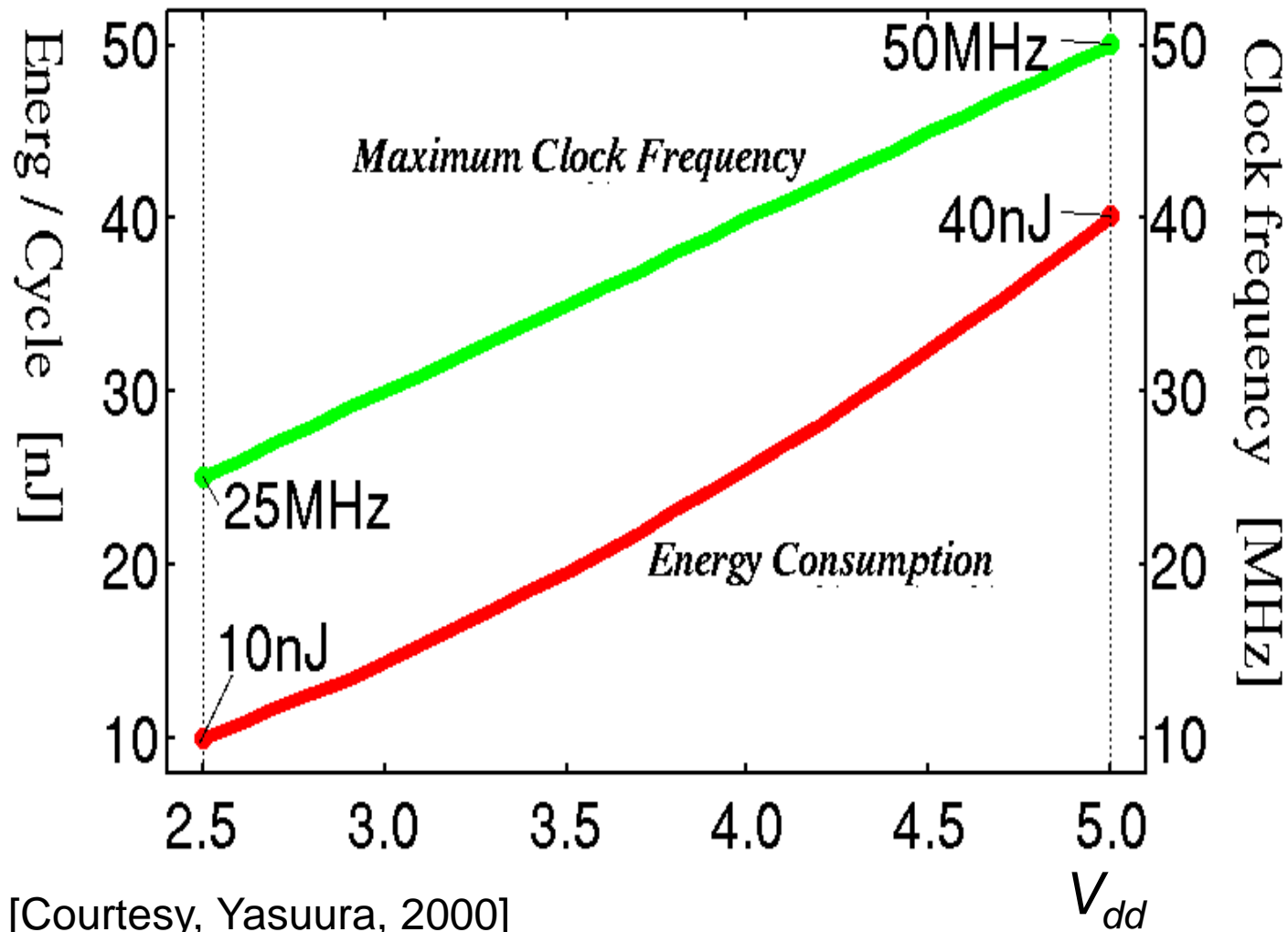
Summary

Hardware in a loop

- Sensors
- Discretization
- Information processing
 - Importance of energy efficiency
 - Special purpose HW very expensive
 - Energy efficiency of processors
 - Code size efficiency
 - Run-time efficiency
 - MPSoCs
- D/A converters
- Actuators

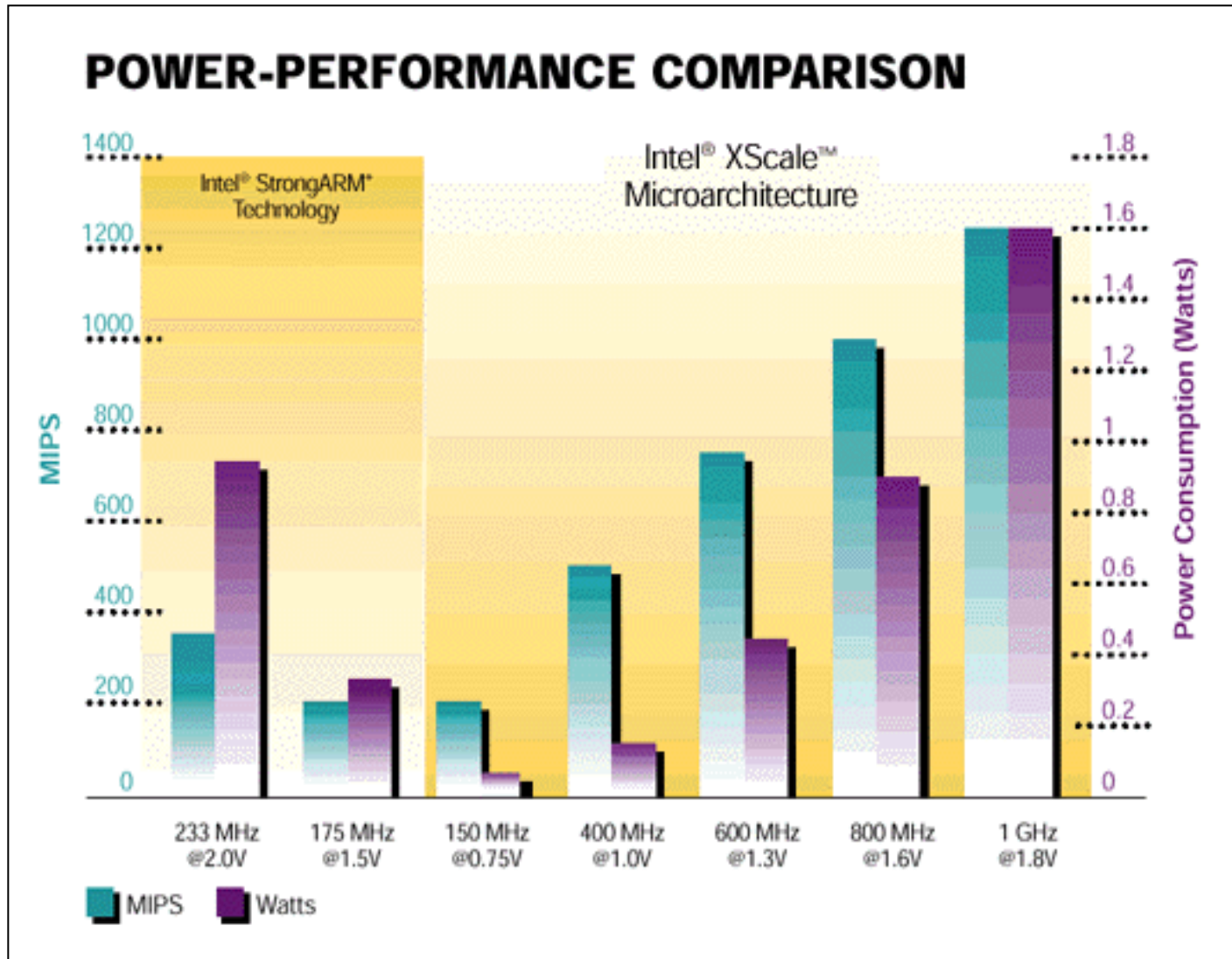
SPARES

Voltage scaling: Example



[Courtesy, Yasuura, 2000]

Variable-voltage/frequency example: INTEL Xscale



From Intel's Web Site