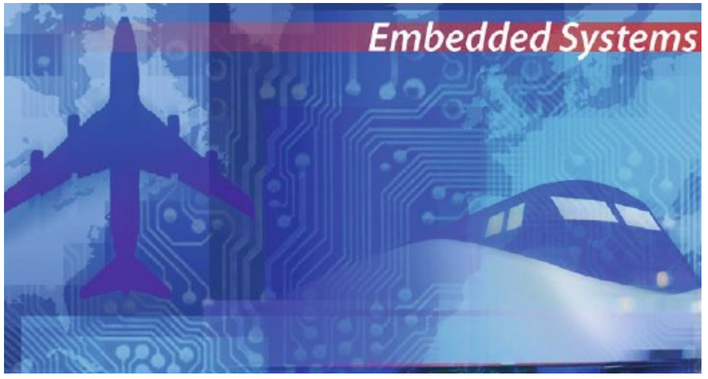
Sharif University of Technology
Department of Computer Science and Engineering

Lec. 4.1:
Embedded System Hardware



M. Ansari

Spring 2023

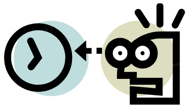
According to Peter Marwedel's Lectures

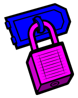
Motivation

(see lecture 1): "*The development of ES cannot ignore the underlying HW characteristics. Timing, memory usage, power consumption, and physical failures are important*."

$$\int P \, dt$$

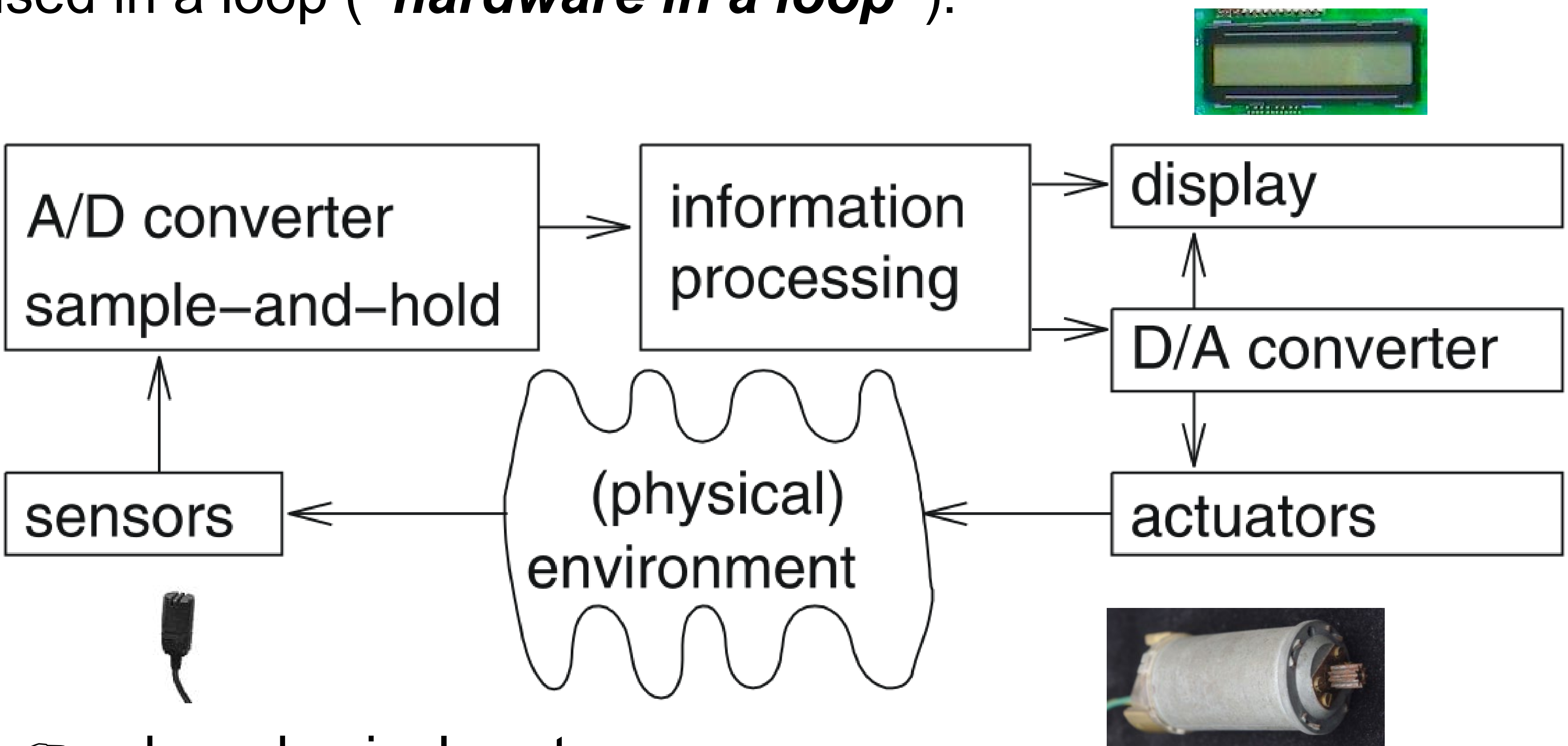
Reasons for considering hard- and software:

- Real-time behavior 
- Efficiency 

  - Energy
  - …
- Security 
- Reliability 
- …

Embedded System Hardware

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

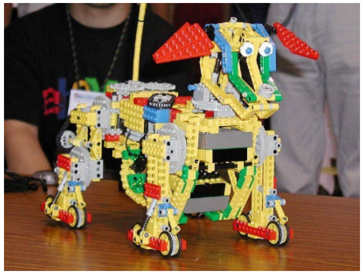


☞ cyber-physical systems

Many examples of such loops

- Heating
- Lights
- Engine control
- Power supply
- …
- Robots

© P. Marwedel, 2011



Embedded System Hardware
- Sensors-

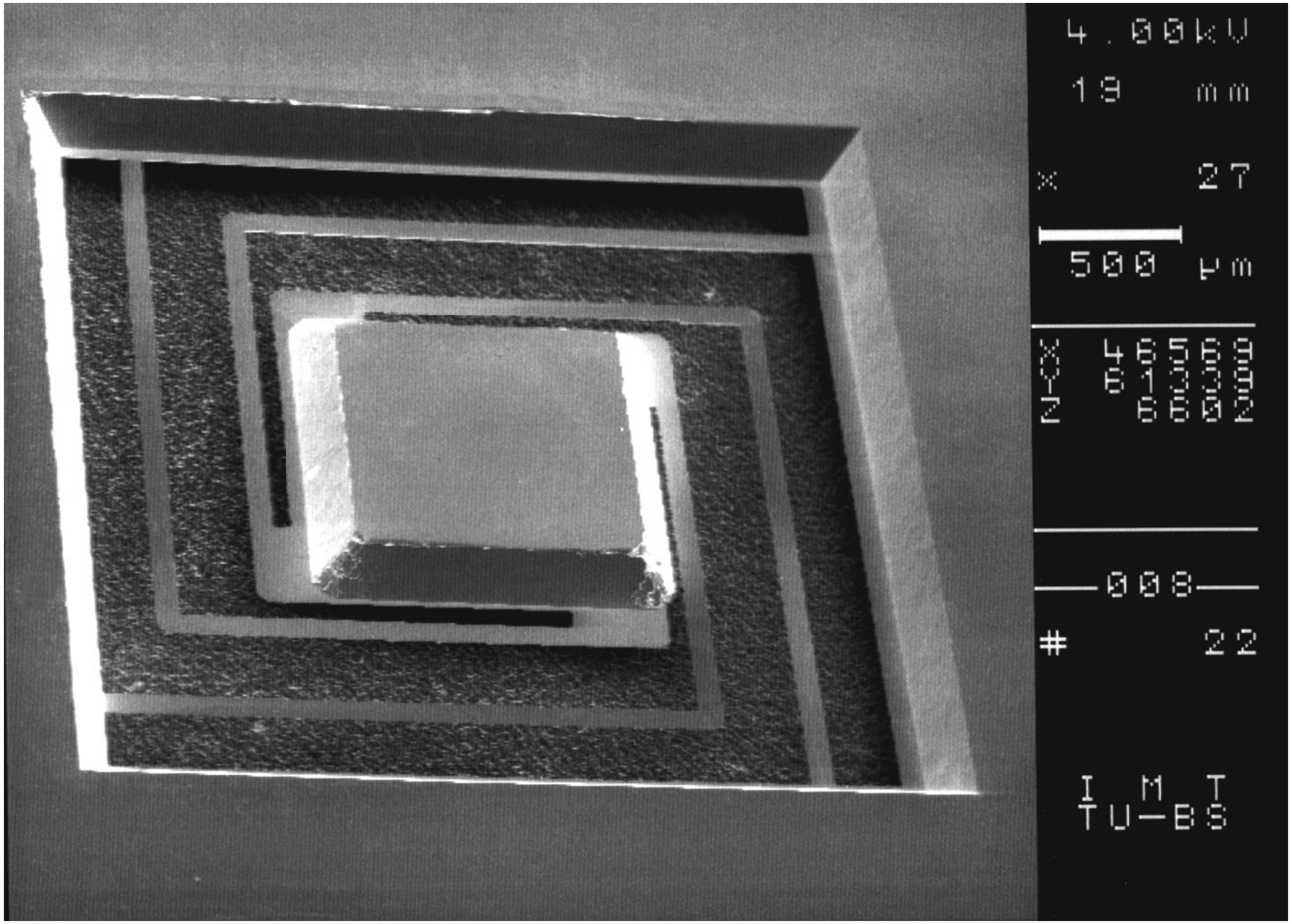
Sensors

- Processing of physical data starts with capturing this data. Sensors can be designed for virtually every physical and chemical quantity, including
  - weight, velocity, acceleration, electrical current, voltage, temperatures, and
  - chemical compounds.
- Many physical effects used for constructing sensors.

Examples:

  - law of induction (generat. of voltages in a magnetic field),
  - light-electric effects.
- Huge amount of sensors designed in recent years.

Example: Acceleration Sensor



Courtesy & ©: S. Bütgenbach, TU Braunschweig

Charge-coupled devices (CCD) image sensors

- CCD technology is optimized for optical applications.
- In CCD technology, charges must be transferred from one pixel to the next until they can finally be read out at an array boundary.
- This sequential charge transfer also gave CCDs their name. For CCD sensors, interfacing is more complex.
- Several application areas for CCDs have disappeared, but they are still used in areas such as scientific image acquisition.

CMOS image sensors

- The architecture of CMOS sensor arrays is similar to that of standard memories: individual pixels can be randomly addressed and read out.
- CMOS sensors use standard CMOS technology for integrated circuits.
- CMOS sensors require only a single standard supply voltage and interfacing in general is easy.
- CMOS-based sensors can be cheap.
- Based on standard production process for CMOS chips, allows integration with other components.

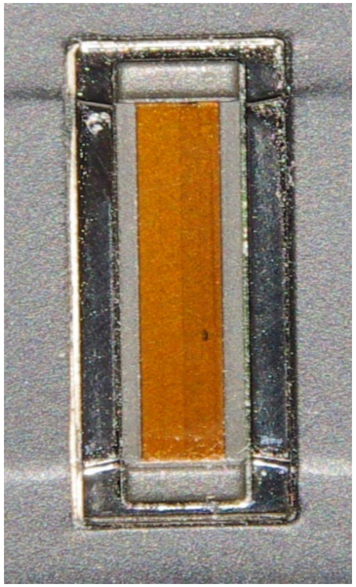
Comparison CCD/CMOS sensors

Property	**CCD**	**CMOS**
Technology optimized for	Optics	VLSI technology
Technology	Special	Standard
Smart sensors	No, no logic on chip	Logic elements on chip
Access	Serial	Random
Size	Limited	Can be large
Power consumption	Low	Larger
Video mode	Possibly too slow	ok
Applications	Situation is changing over the years	

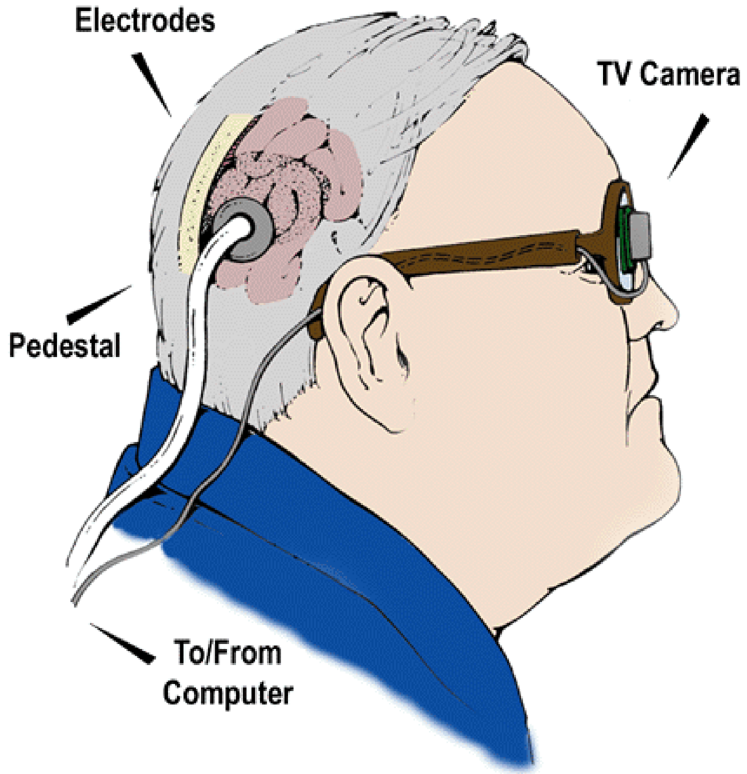
Example: Biometrical Sensors

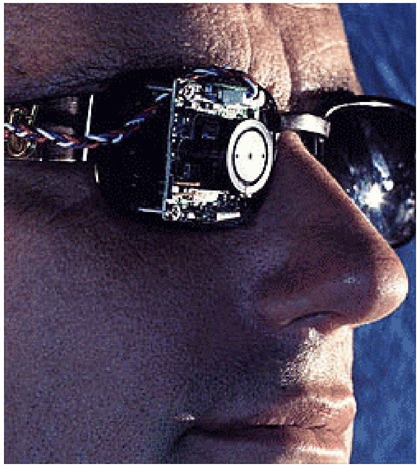
- Demands for higher security standards as well as the need to protect mobile and removable equipment have led to an increased interest in authentication.
  - e.g.: Fingerprint sensor
- False accepts as well as false rejects are an inherent problem of biometric authentication.

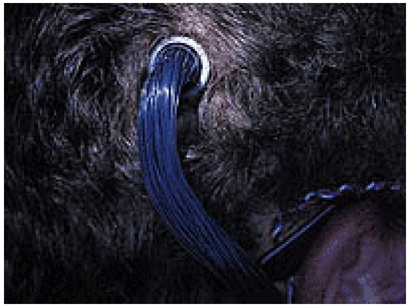
© P. Marwedel, 2010



Artificial eyes (1)







© Dobelle Institute
(was at www.dobelle.com)

Show movie from www.dobelle.com
(e.g. blind person driving a car)

Artificial eyes (2)

- Translation into sound
  [http://www.seeingwithsound.com/etumble.htm]

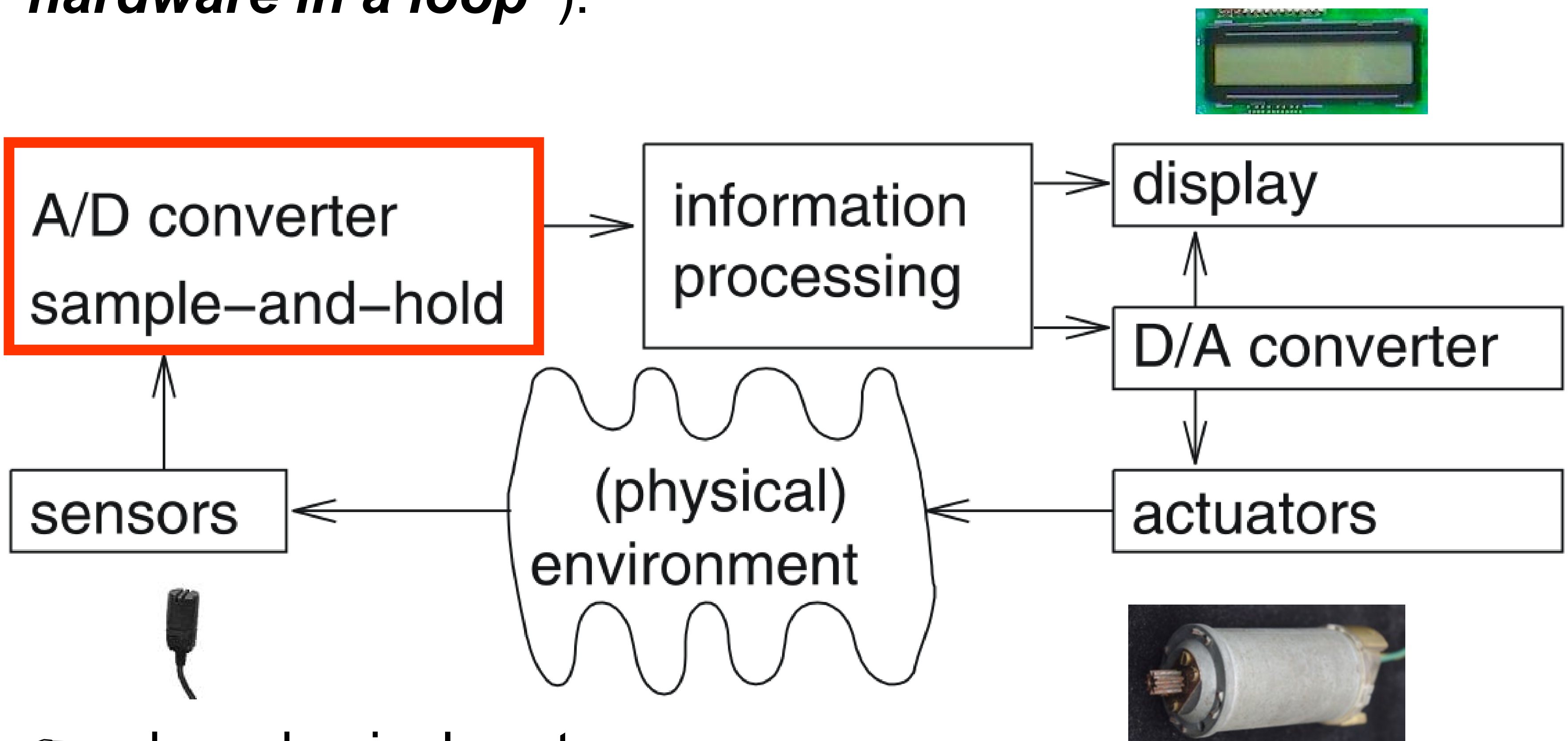
Other sensors

- Rain sensors for wiper control
  (“Sensors multiply like rabbits“ [ITT automotive])
- Pressure sensors
- Proximity sensors
- Engine control sensors

Embedded System Hardware
- A/D Converter-

Embedded System Hardware

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



☞ cyber-physical systems

Signals

Sensors generate *signals*

Definition: a **signal** s is a mapping

from the time domain D_T to a value domain D_V:

$$s : D_T \rightarrow D_V$$

D_T: continuous or discrete time domain

D_V: continuous or discrete value domain.

Discretization of time

Digital computers require discrete sequences of physical values

$$s : D_T \rightarrow D_V$$

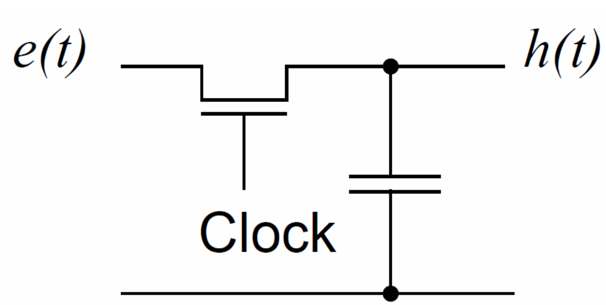
Discrete time domain

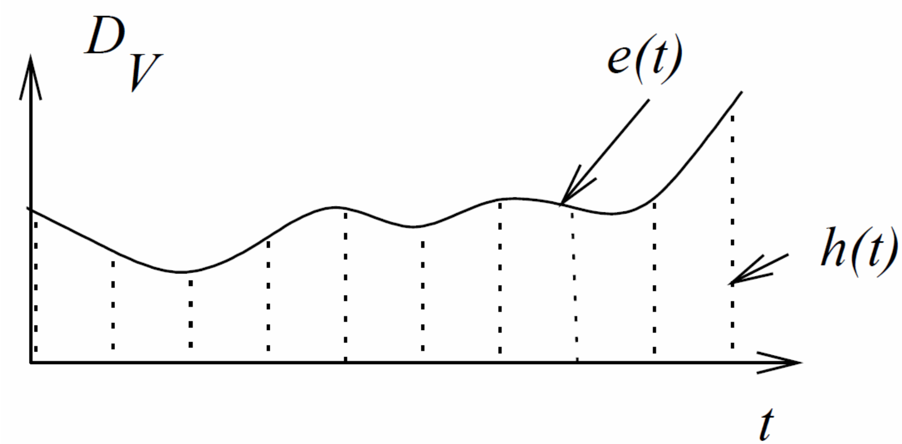
☞ Sample-and-hold circuits

Sample-and-hold circuits

Clocked transistor + capacitor;

Capacitor stores sequence values





$e(t)$ is a mapping $\mathbb{R} \rightarrow \mathbb{R}$

$h(t)$ is a **sequence** of values or a mapping $\mathbb{Z} \rightarrow \mathbb{R}$

Discretization of values: A/D-converters

Digital computers require digital form of physical values

$$s: D_T \rightarrow D_V$$

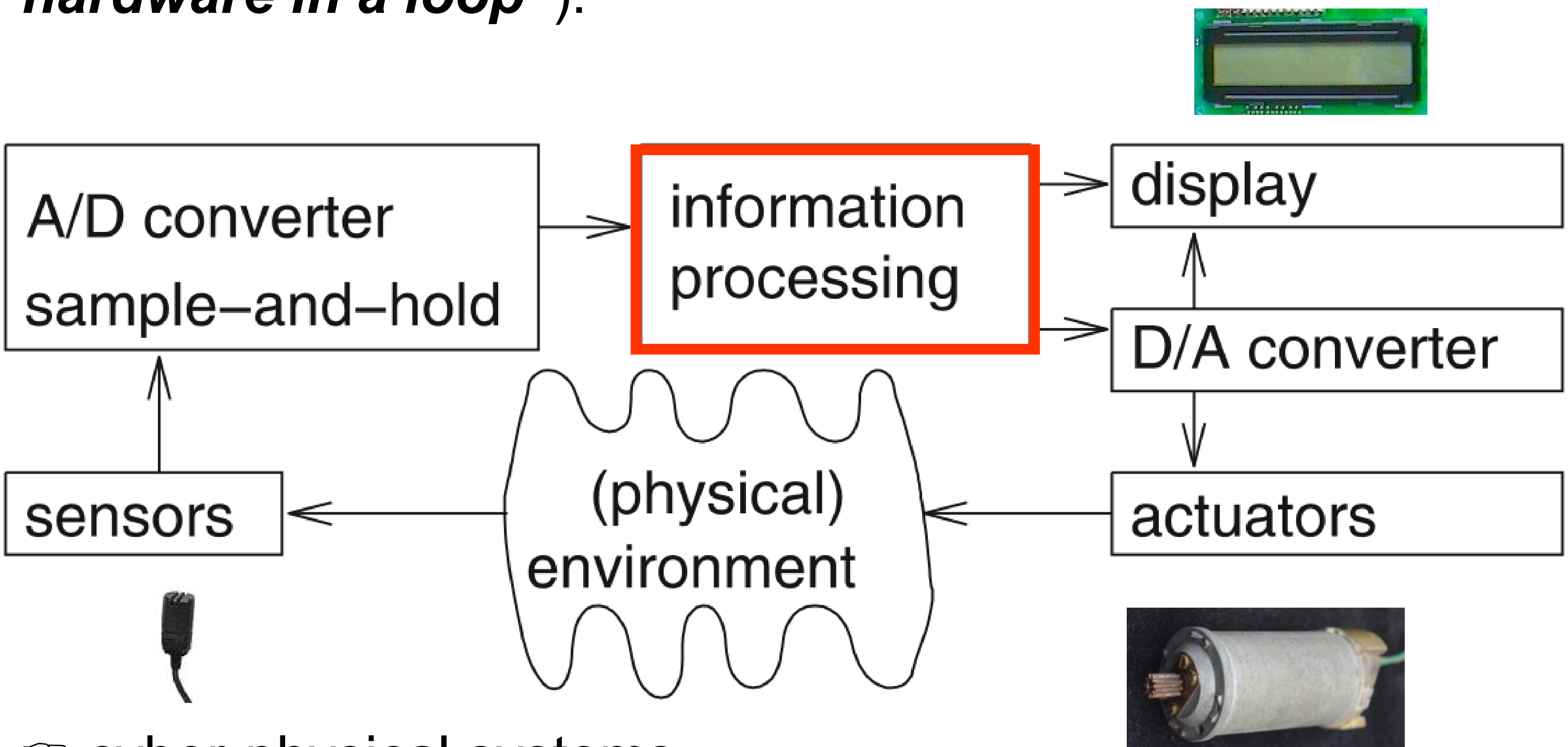
Discrete value domain

☞A/D-conversion; many methods with different speeds.

Embedded System Hardware
- Information Processing-

Embedded System Hardware

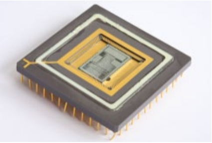
Embedded system hardware is frequently used in a loop (***“hardware in a loop“***):



☞ 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

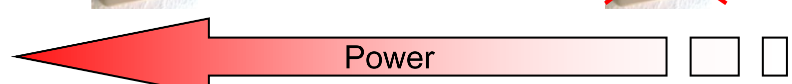
  - Cost efficient

  - Energy efficient

© Graphics: Microsoft, P. Marwedel, M. Engel, 2011

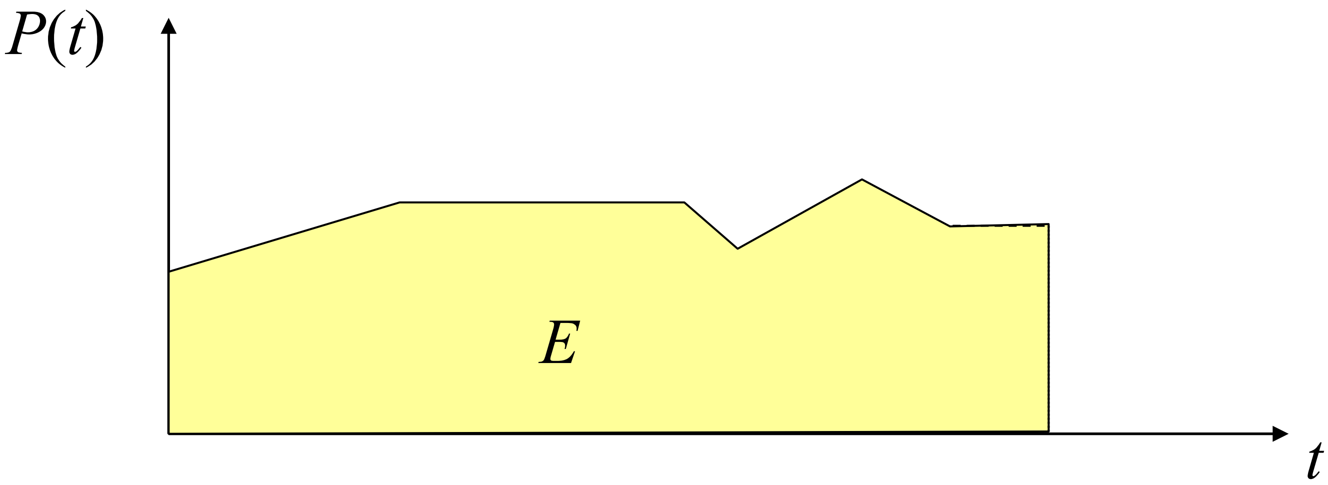
Why care about energy efficiency ?

	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	☐	☑	☑



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

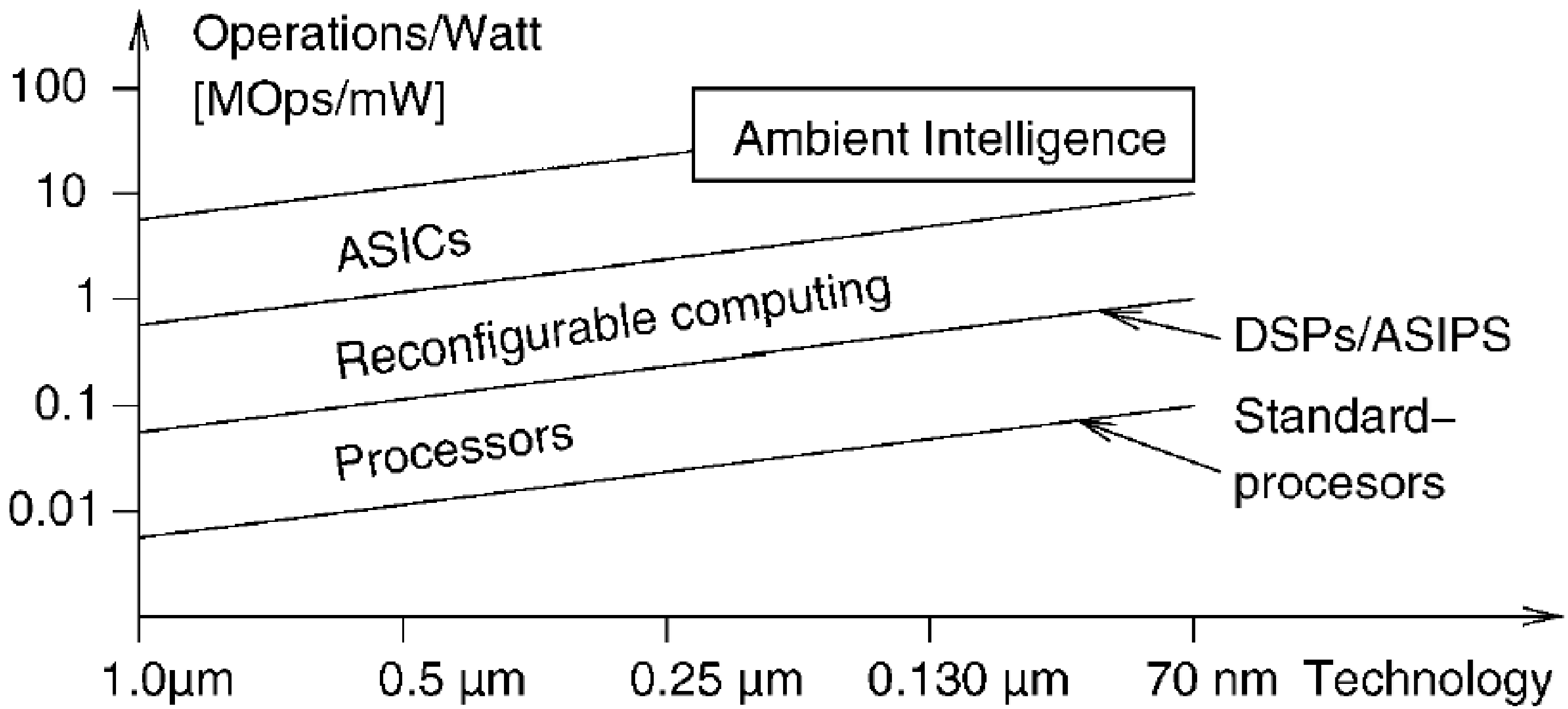
- Minimizing **energy consumption** important due to
  - restricted availability of energy (mobile systems)
  - cooling: high costs, limited space
  - thermal effects
  - dependability, long lifetimes

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

Energy and Power Consumption

- Power consumption
  - Size of the power supply
  - Design of the voltage regulators
  - Dimensioning of the interconnect
  - Short term cooling
- Energy consumption
  - Mobile applications
    - Battery life time

Impact of PUs on Energy and Power Consumption

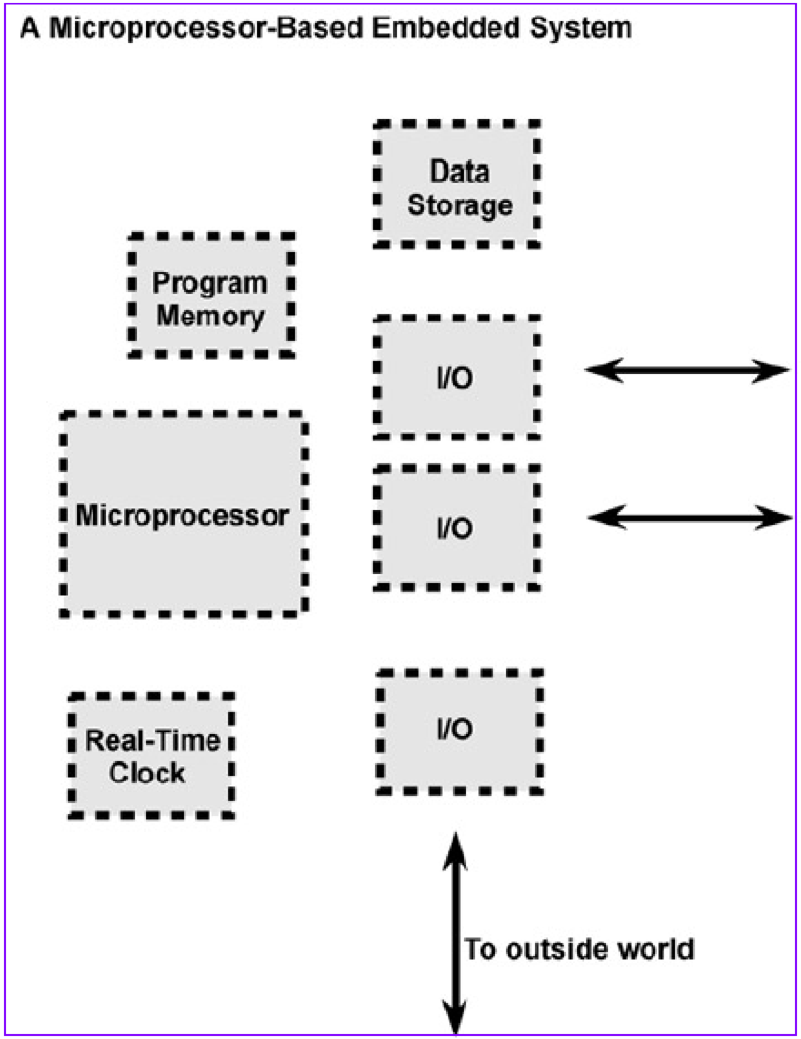


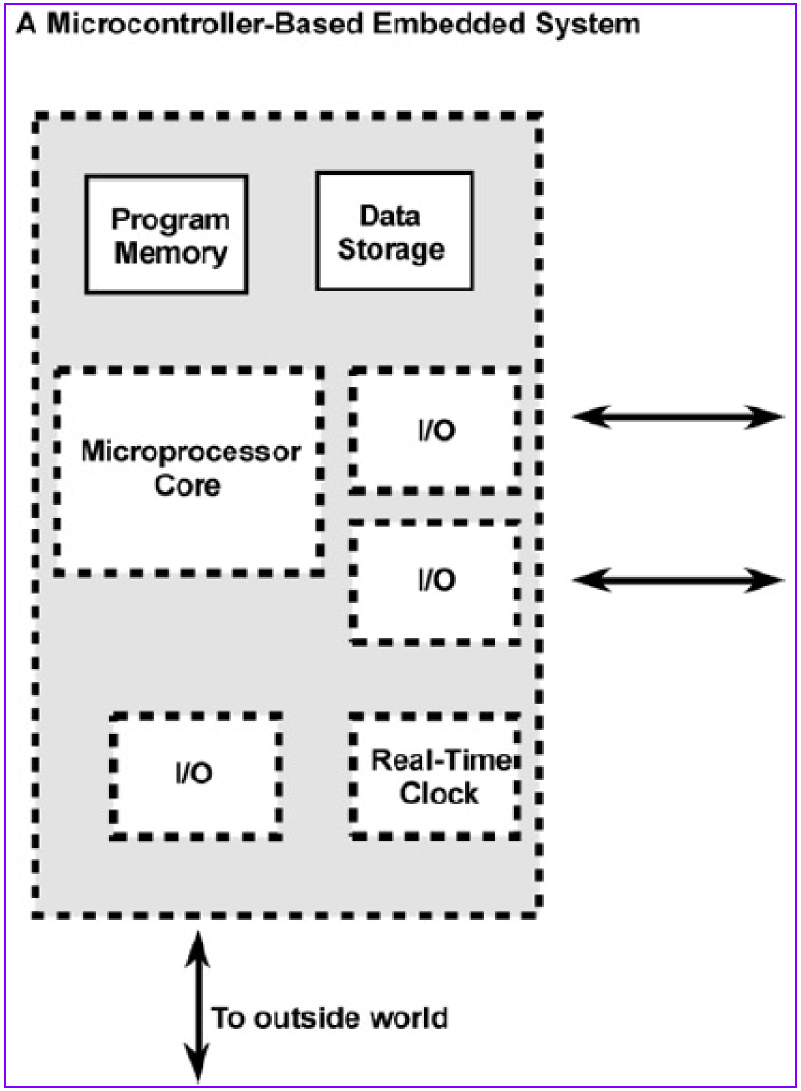
Embedded Processors vs. PCs

- Embedded processors do not need to be instruction set compatible with PCs.

- Efficiency
  - Energy efficiency
  - Code-size efficiency
  - Run-time efficiency (e.g., ASIPs)

Microprocessor VS. Microcontroller





Microcontrollers

- **Microcontrollers have become so prevalent and even dominate the entire embedded world.**
  - Lower cost: One part replaces many parts.
  - More reliable: Fewer packages, fewer interconnects.
  - Better performance: System components are optimized for their environment, Signals can stay on the chip.

Core-Based Microcontrollers

- 8086 processor
  - 80186 family of devices

- Motorola’s 68000 and 68020
  - 68300 family of devices

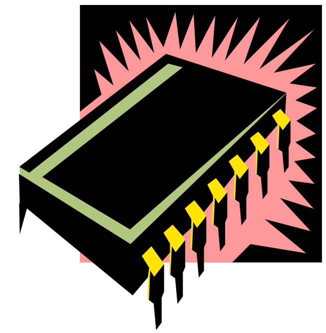
Benefit $\Rightarrow$ Cost Reduction

Embedded System Hardware
- Memory-

Memory

Memories?

Oops!
Memories!

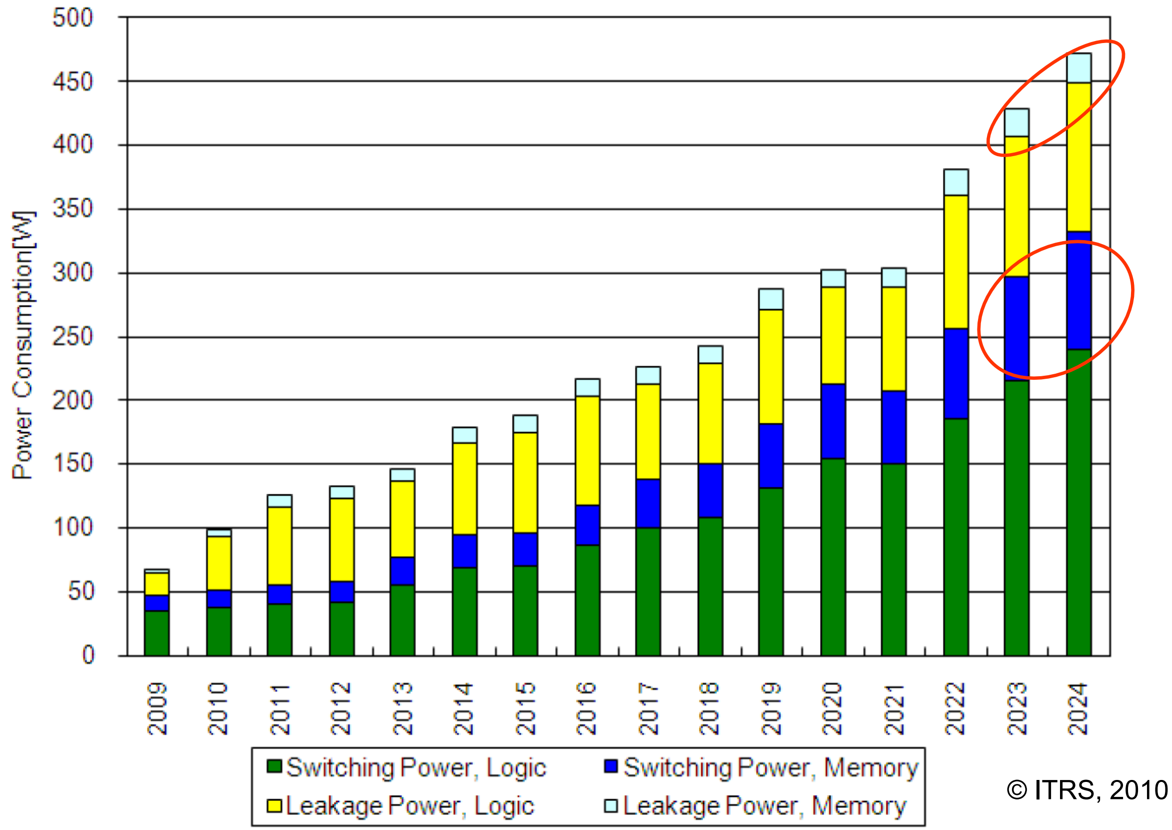


For the memory, efficiency is again a concern:

- capacity
- energy efficiency
- speed (latency and throughput); predictable timing
- size
- cost
- other attributes (volatile vs. persistent, etc)

Where is the power consumed?
- Stationary systems -

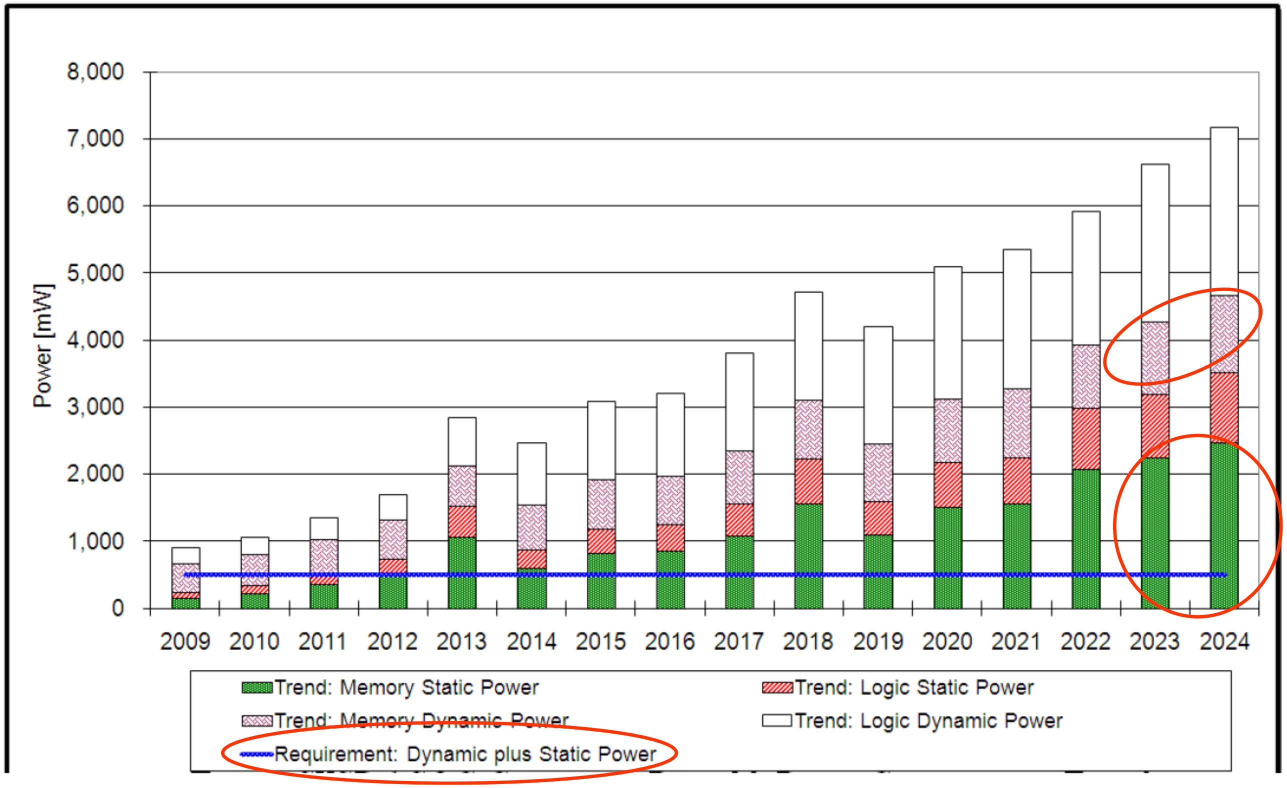
- According to *International Technology Roadmap for Semi-conductors* (ITRS), 2010 update, [www.itrs.net]



- Switching power, logic dominating
- Overall power consumption a nightmare for environmentalists

Where is the power consumed?
- Consumer portable systems -

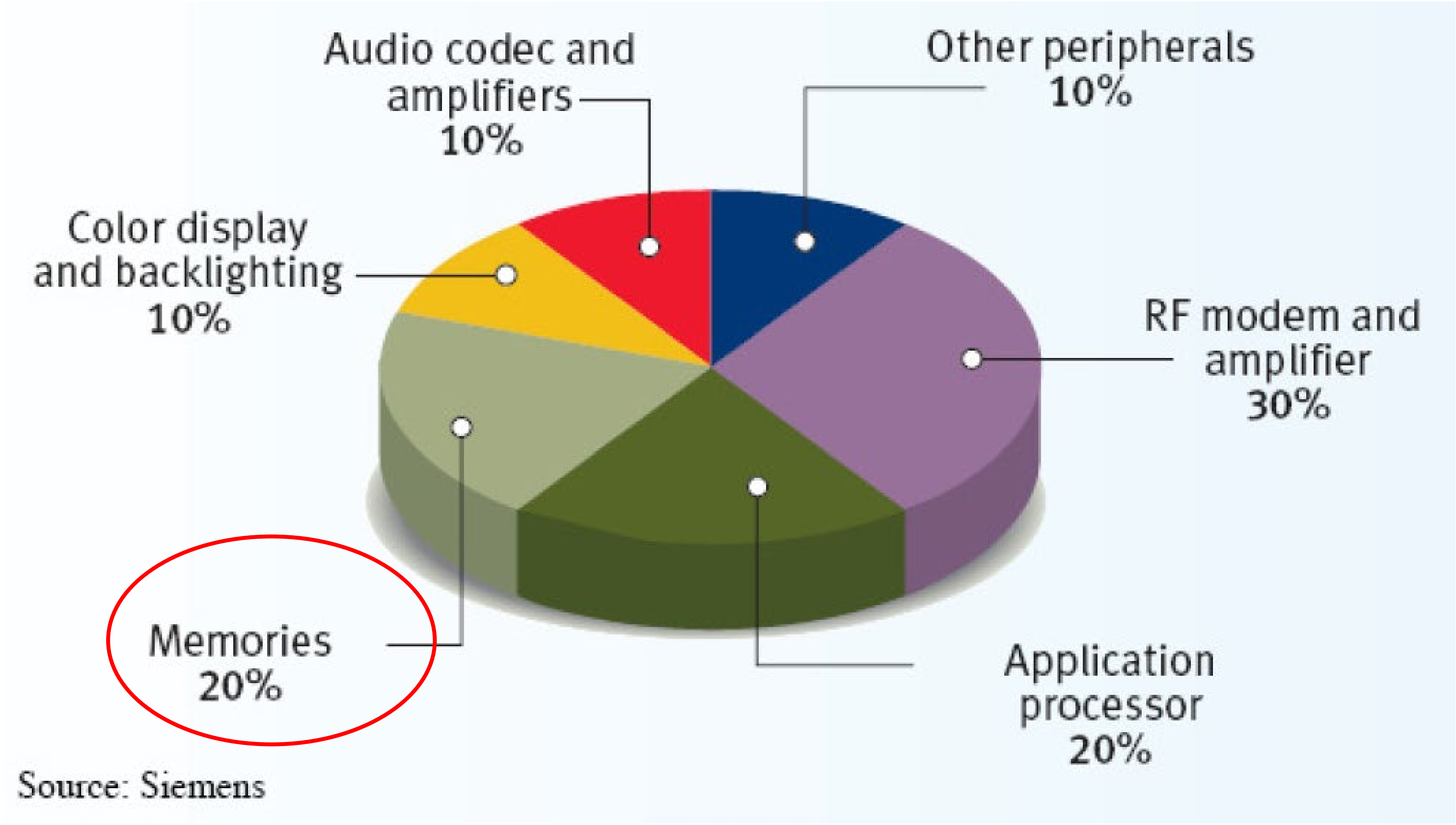
- According to *International Technology Roadmap for Semi-conductors* (ITRS), 2010 update, [www.itrs.net]
- Based on current trends



© ITRS, 2010

- Memory and logic, static and dynamic relevant
- Following current trends will violate maximum power constraint (0.5-1 W).

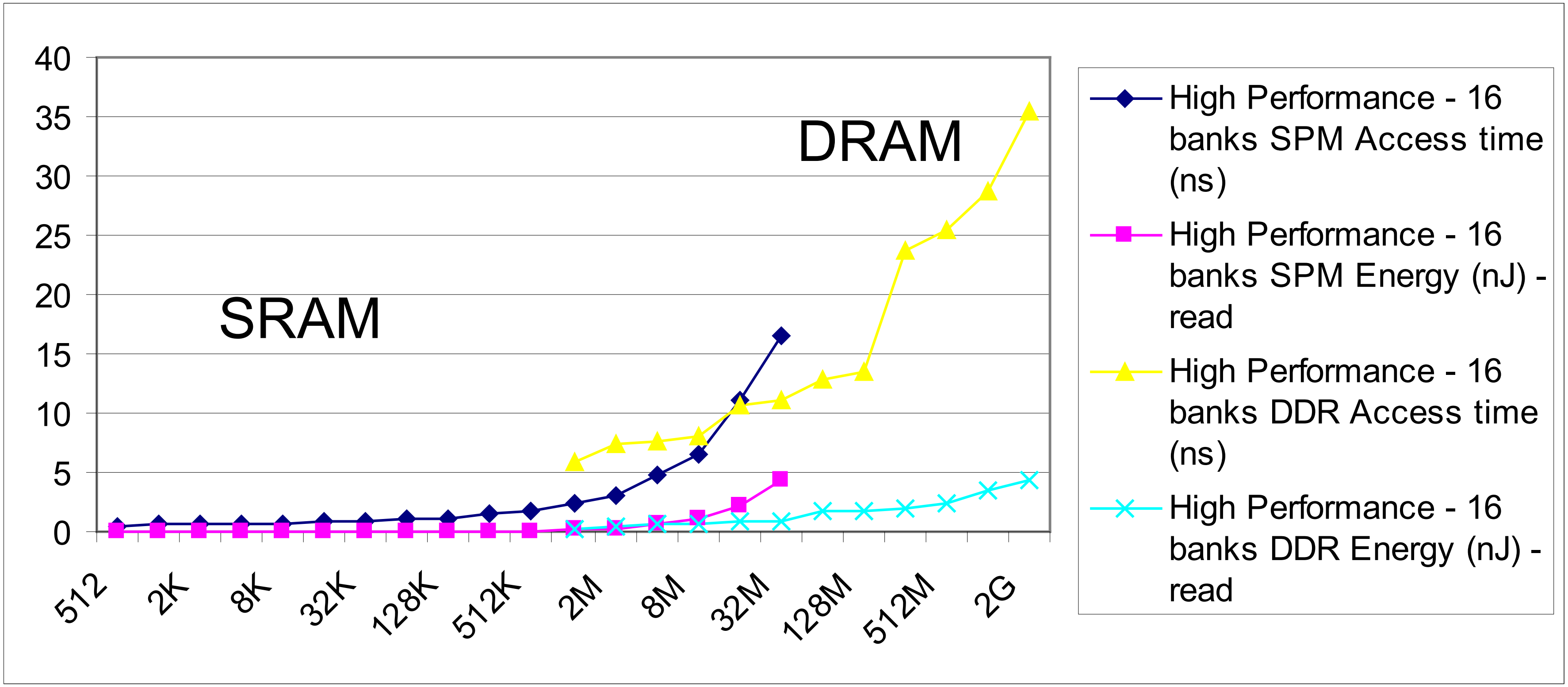
Memory energy significant even if we take display and RF of mobile device into account



[O. Vargas (Infineon Technologies): Minimum power consumption in mobile-phone memory subsystems; Pennwell Portable Design - September 2005;] Thanks to Thorsten Koch (Nokia/ Univ. Dortmund) for providing this source.

Energy consumption and access times of memories

Example CACTI: Scratchpad (SRAM) vs. DRAM (DDR2):

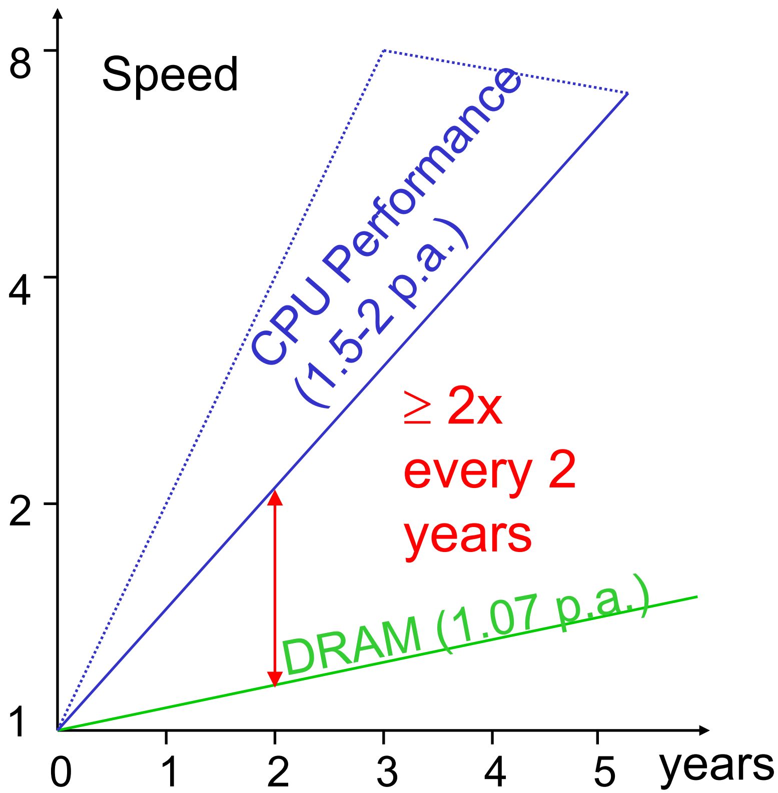


16 bit read; size in bytes;
65 nm for SRAM, 80 nm for DRAM

Source: Olivera Jovanovic, TU Dortmund, 2011

Trends for the Speeds

Speed gap between processor and main DRAM increases



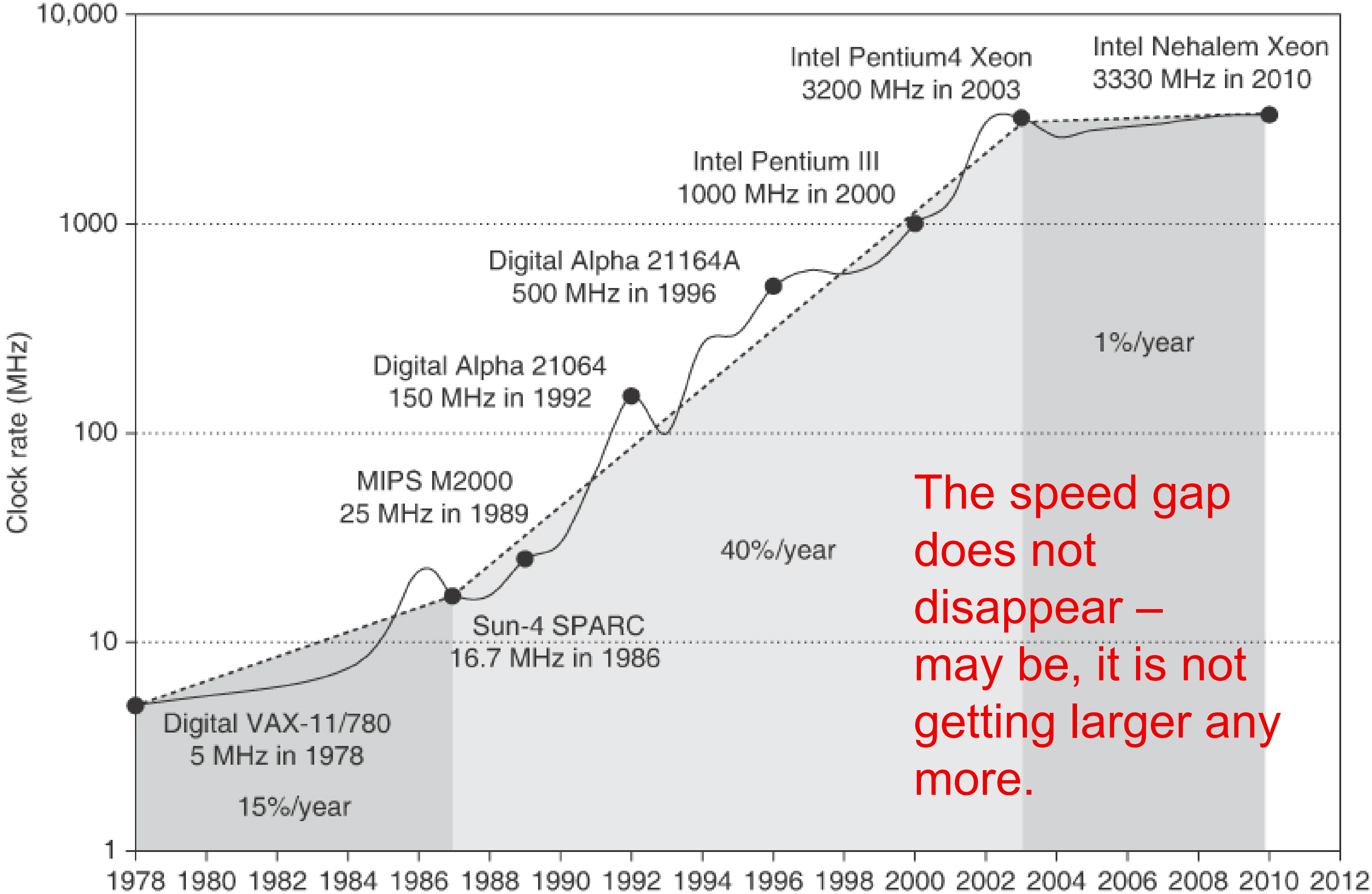
Similar problems also for embedded systems & MPSoCs

☞ Memory access times >> processor cycle times

☞ "Memory wall" problem

[P. Machanik: Approaches to Addressing the Memory Wall, TR Nov. 2002, U. Brisbane]

However, clock speed increases have come to a halt

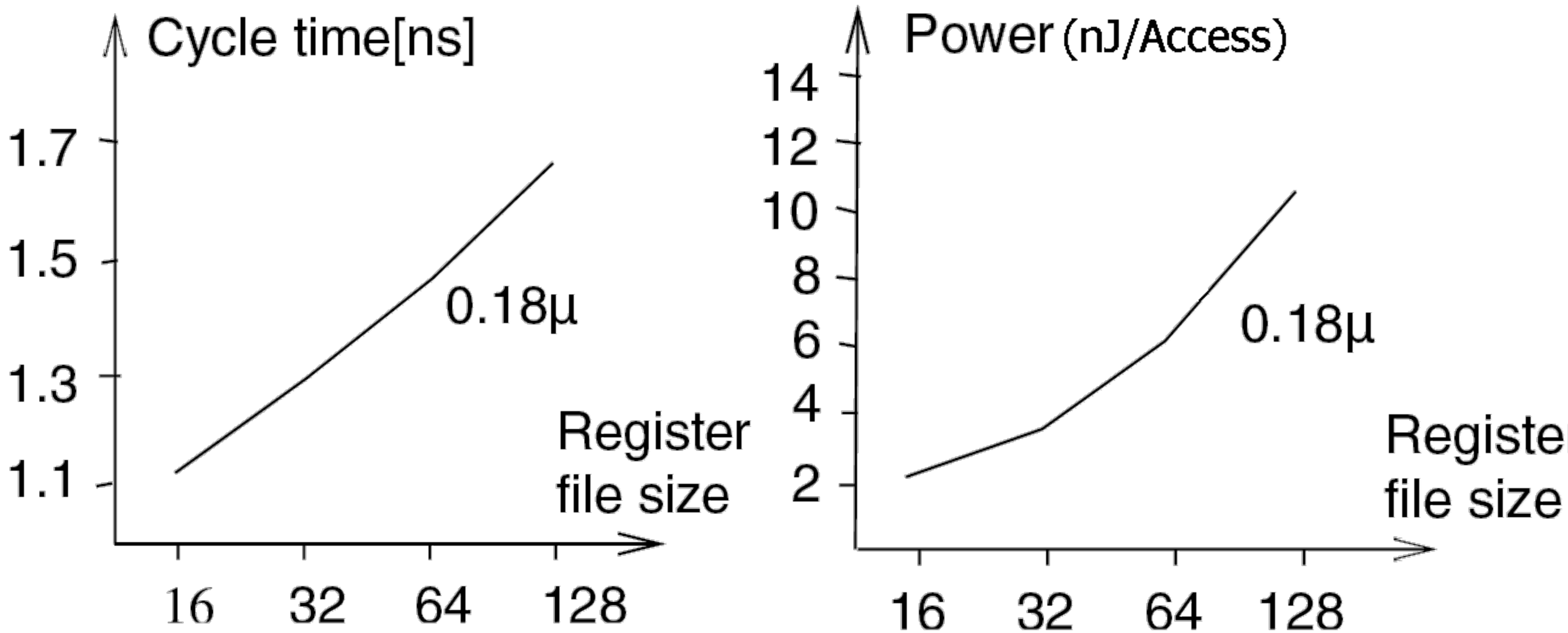


The speed gap does not disappear – may be, it is not getting larger any more.

Copyright © 2011, Elsevier Inc. All rights Reserved.

[Hennessy/Patterson: Computer Architecture, 5th ed., 2011]

Impact of MEM Size

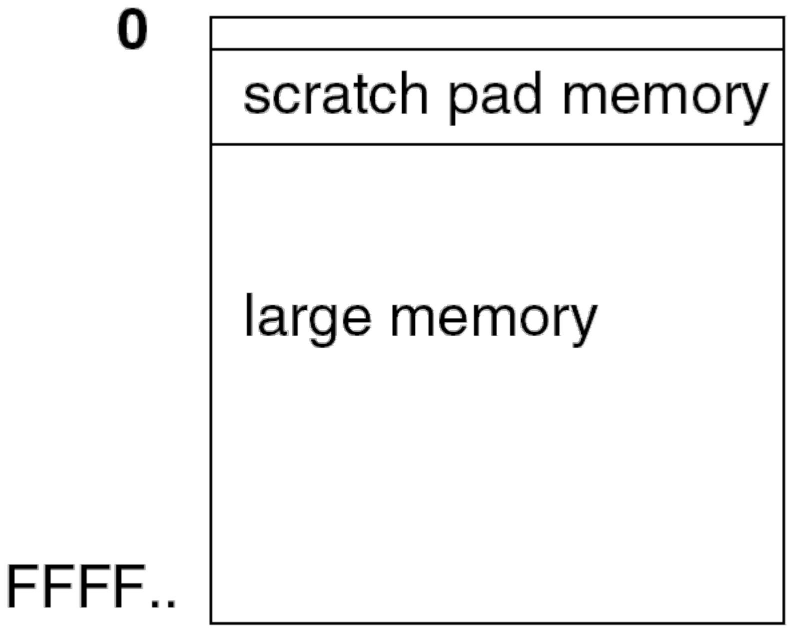


Disadvantages of Caches

- The predictability of the real-time performance of caches is frequently low.
  - Serious problems in scheduling

Solution: Scratch Pad Memory

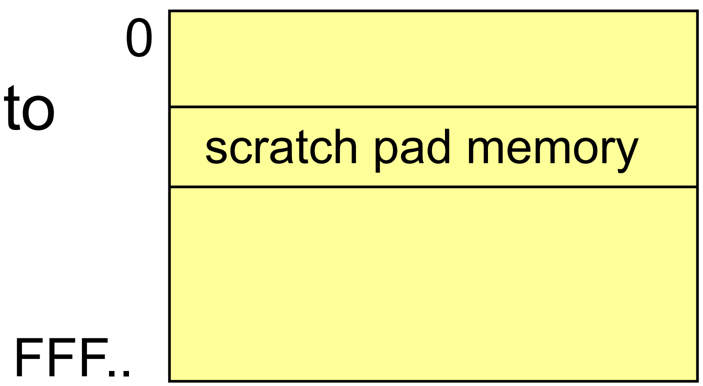
- SPM vs. Cache
  - The only difference is that Caches are transparent while SPMs are not.



Hierarchical memories using scratch pad memories (SPM)

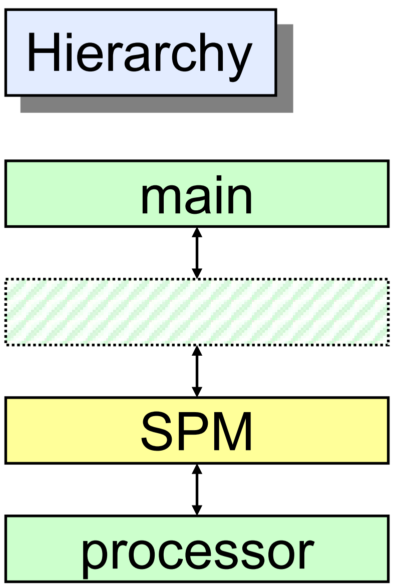
SPM is a small, physically separate memory mapped into the address space

Address space

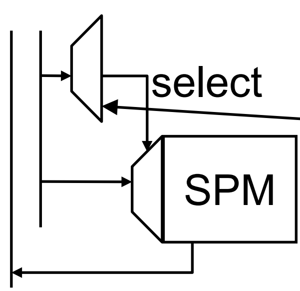


Examples:

- Most ARM cores allow tightly coupled memories
- IBM Cell
- Infineon TriCore
- Many multi-cores, due to high costs of coherent caches



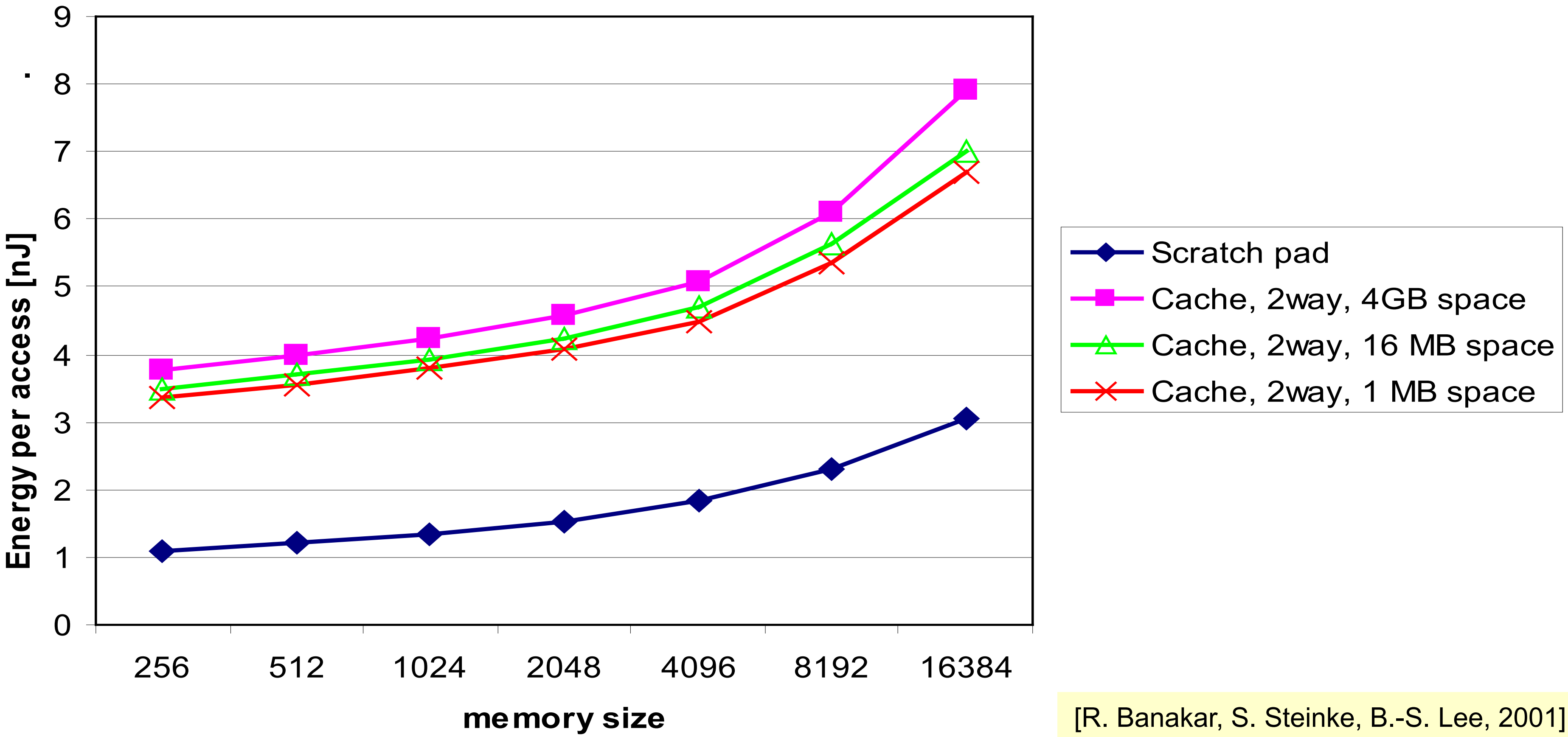
no tag memory



Selection is by an appropriate address decoder (simple!)

Why not just use a cache? (SPM vs. Cache)

❖ Energy for parallel access of sets, in comparators, muxes.



[R. Banakar, S. Steinke, B.-S. Lee, 2001]

SPM vs. Cache

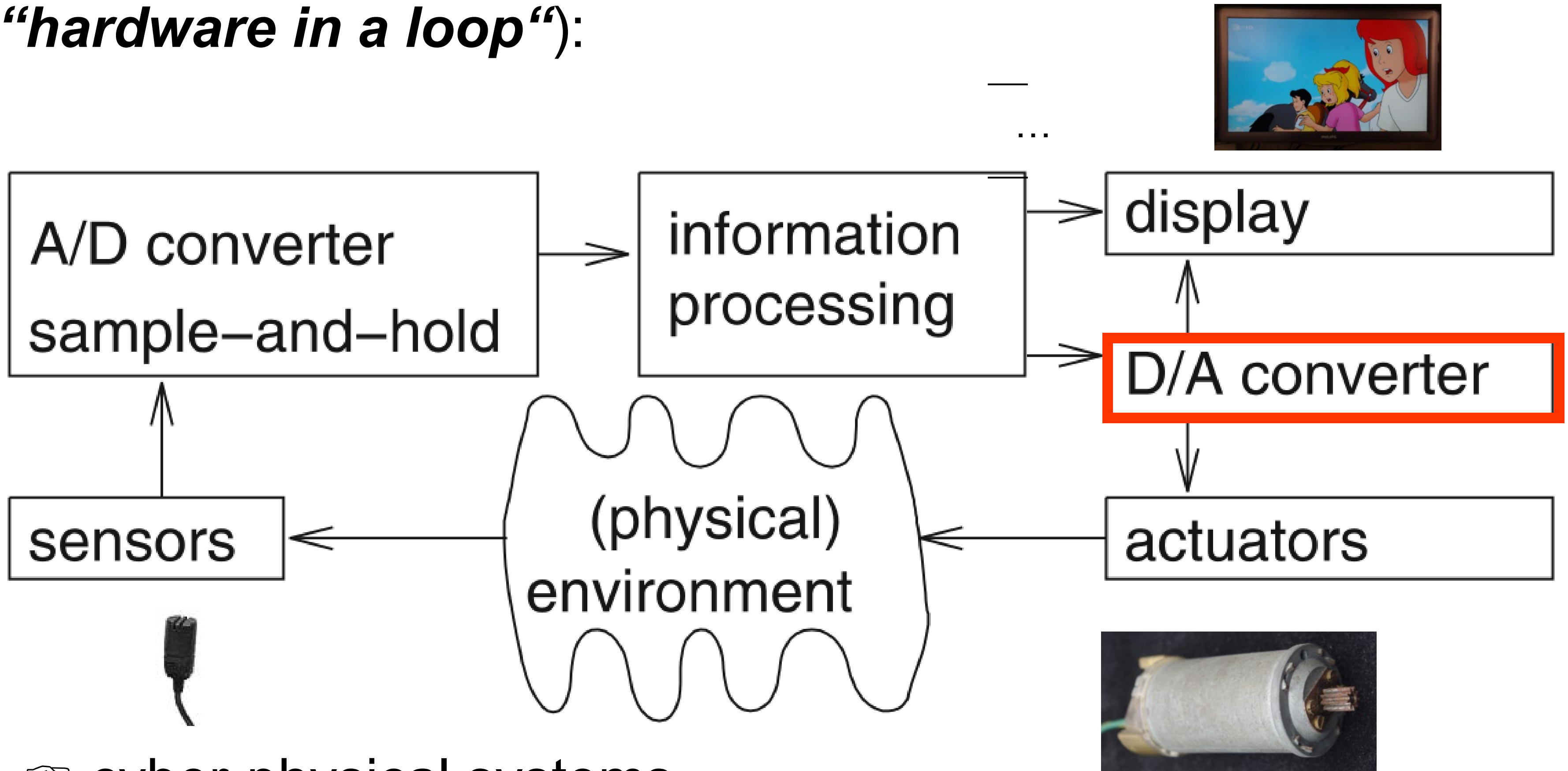
- SPMs are more power/energy-efficient than caches.
  - There is no cache controller hardware

- From a programming point of view:
  - The usage of SPMs is harder than caches.

Embedded System Hardware
- D/A Converter-

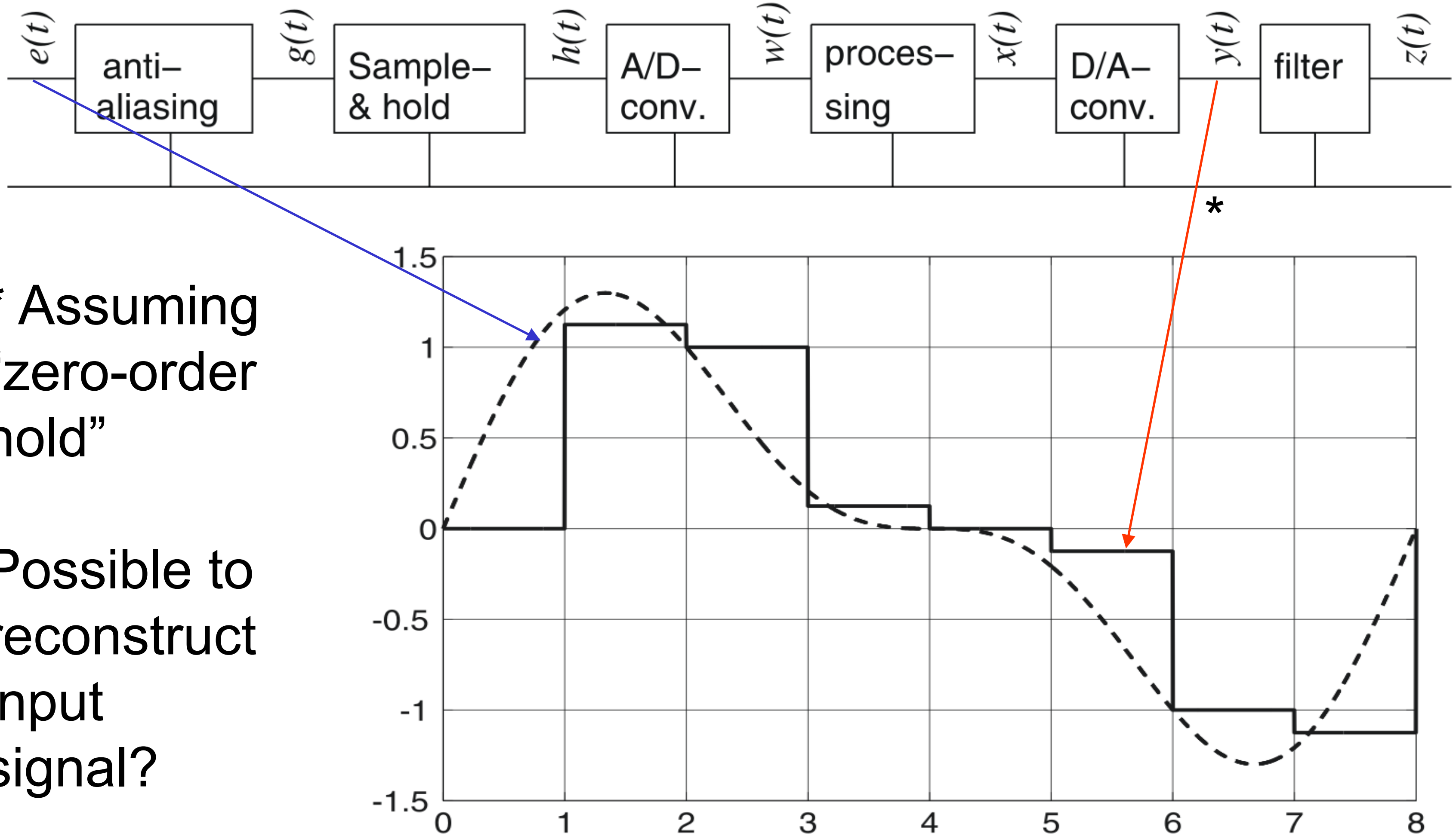
Embedded System Hardware

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



☞ cyber-physical systems

Output generated from signal $e_3(t)$



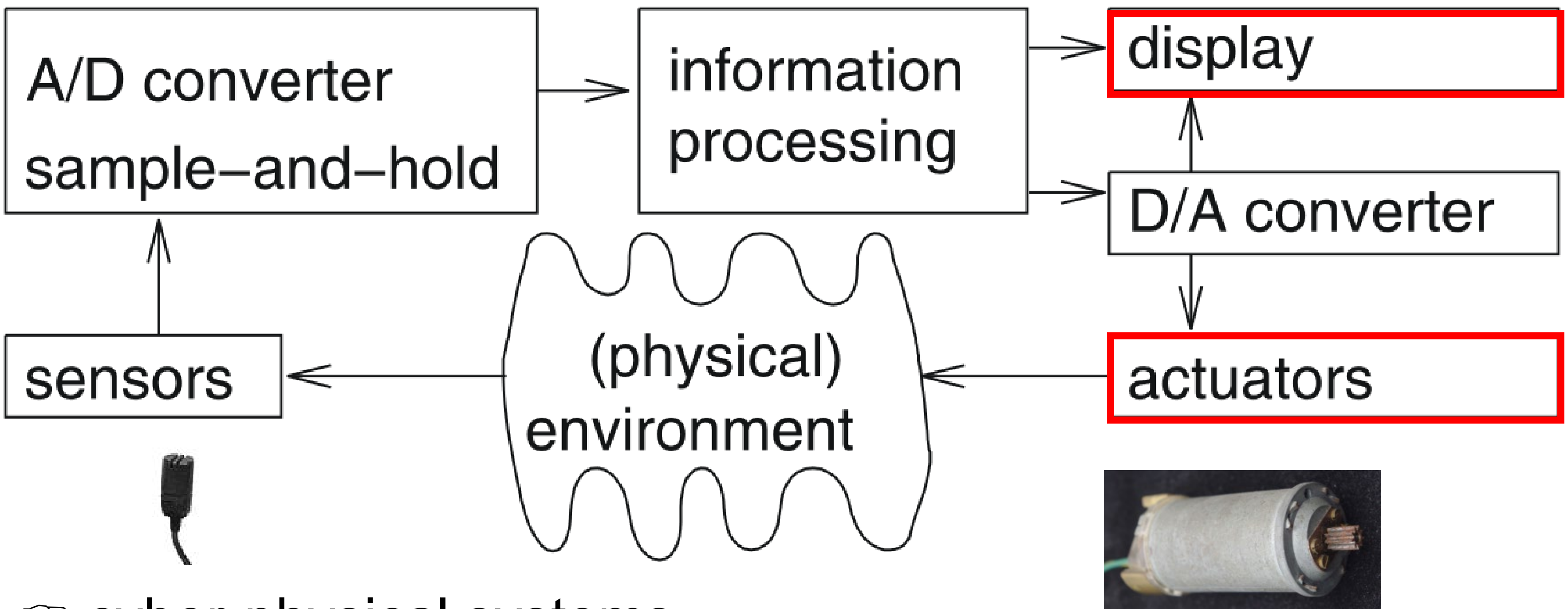
* Assuming "zero-order hold"

Possible to reconstruct input signal?

Embedded System Hardware
- Actuators and Displays-

Embedded System Hardware

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



☞ cyber-physical systems

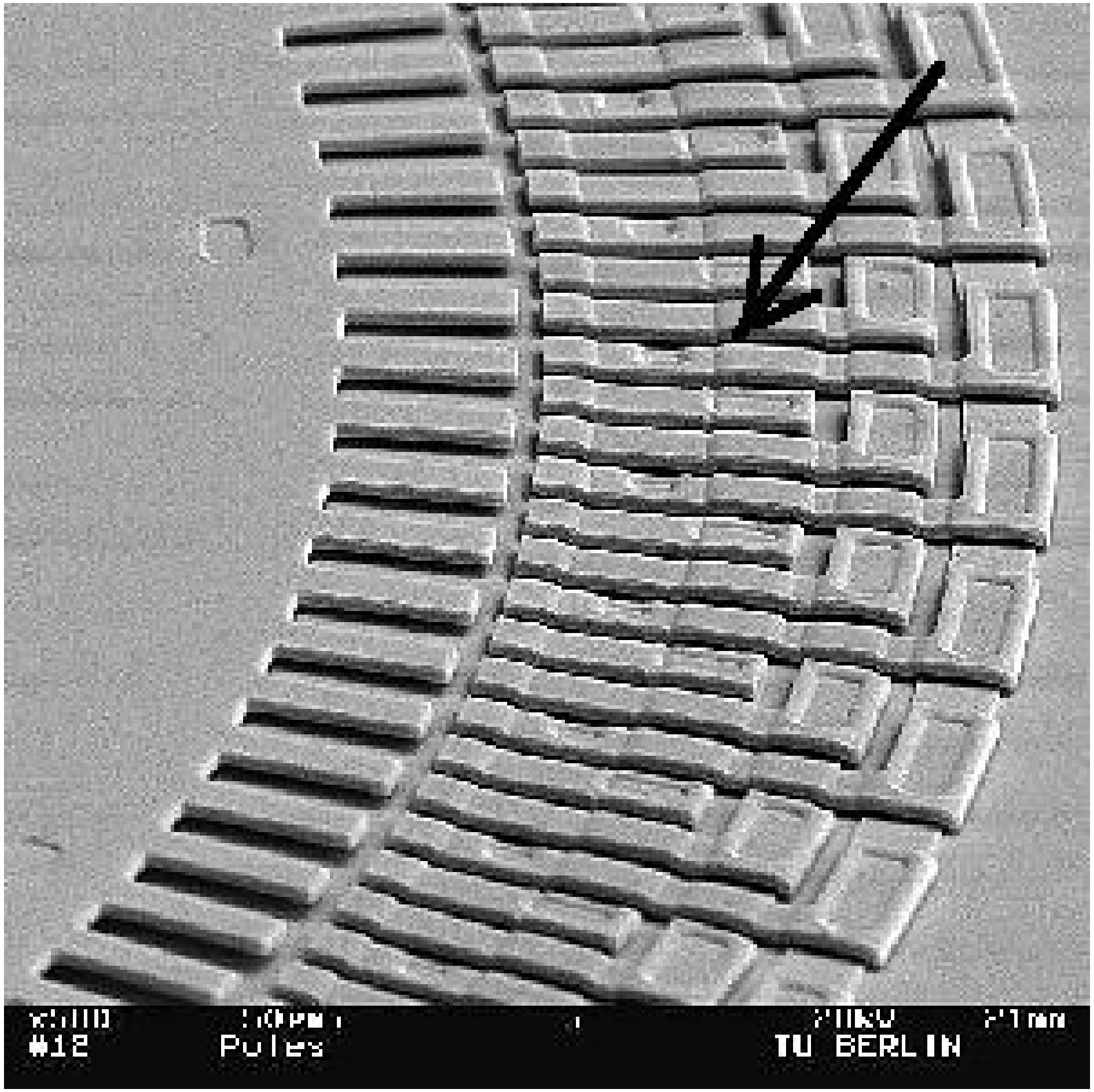
Displays

- Display technology is an area which is extremely important.
- A large amount of information exists on this technology.
- Major research and development efforts lead to:
  - New display technology such as organic displays.
    - Organic displays are emitting light and can be fabricated with very high densities. In contrast to LCDs, they do not need backlight and polarizing filters. Major changes are therefore expected in these markets.

Actuators

- Huge variety of actuators and output devices, impossible to present all of them.
  - Motor as an example:

Actuators (2)



Courtesy and ©: E. Obermeier, MAT, TU Berlin

http://www.piezomotor.se/pages/PWtechnology.html

http://www.elliptec.com/fileadmin/elliptec/User/Produkte/Elliptec_Motor/Elliptecmotor_How_it_works.h

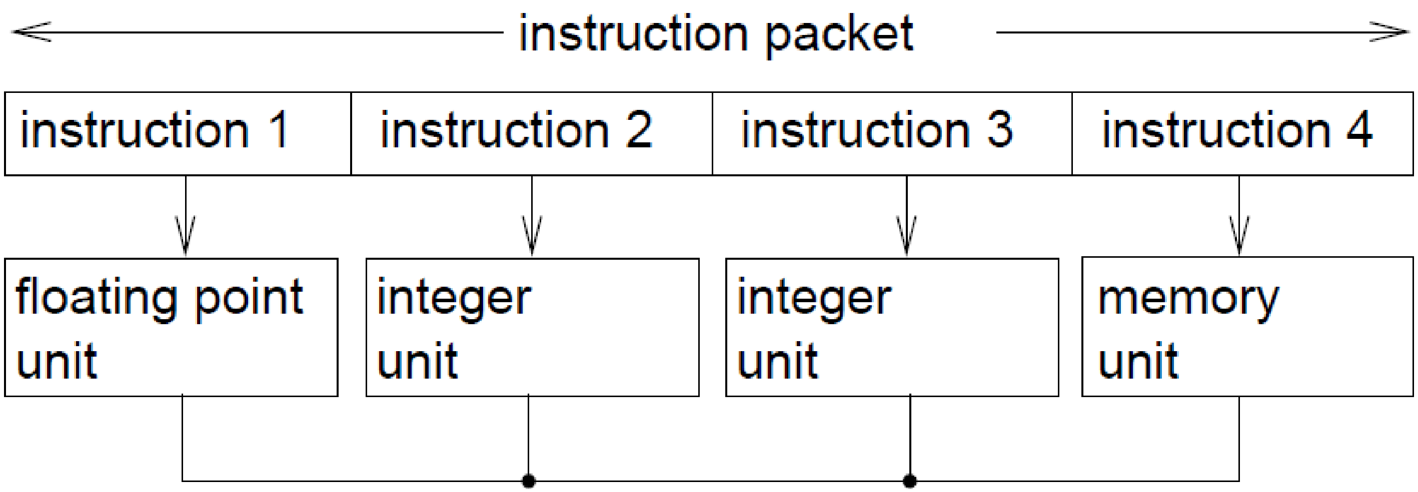
Summary

- Embedded Systems Hardware
  - Sensors
  - Information Processing
    - Processing
    - Memory
  - A/D and D/A Convertor
  - Actuators and Displays

Embedded System Hardware
Others
- Idea of very long instruction word (VLIW) computers-

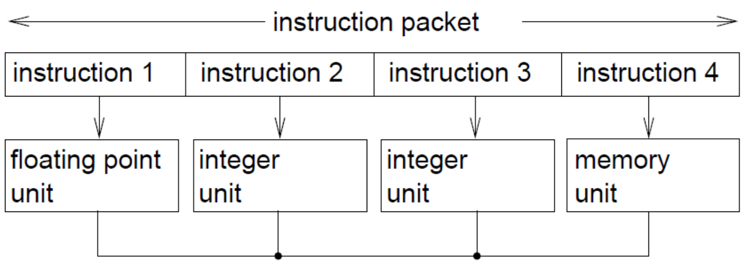
Key idea of very long instruction word (VLIW) computers (1)

- Instructions included in long instruction packets.
- Instruction packets are assumed to be executed in parallel.
- Fixed association of packet bits with functional units.



- Compiler is assumed to generate these “parallel” packets

Key idea of very long instruction word (VLIW) computers (2)



- Complexity of finding parallelism is moved from the hardware (RISC/CISC processors) to the compiler;
- Ideally, this avoids the overhead (silicon, energy, ..) of identifying parallelism at run-time.

☞ A lot of expectations into VLIW machines

- However, possibly low code efficiency, due to many NOPs

☞ Explicitly parallel instruction set computers (EPICs) are an extension of VLIW architectures: parallelism detected by compiler, but no need to encode parallelism in 1 word.

EPIC: TMS 320C6xxx as an example

1 Bit per instruction encodes end of parallel exec.

31 … 0	31 … 0	31 … 0	31 … 0	31 … 0	31 … 0	31 … 0
0	1	1	0	1	1	0
Instr. A	Instr. B	Instr. C	Instr. D	Instr. E	Instr. F	Instr. G

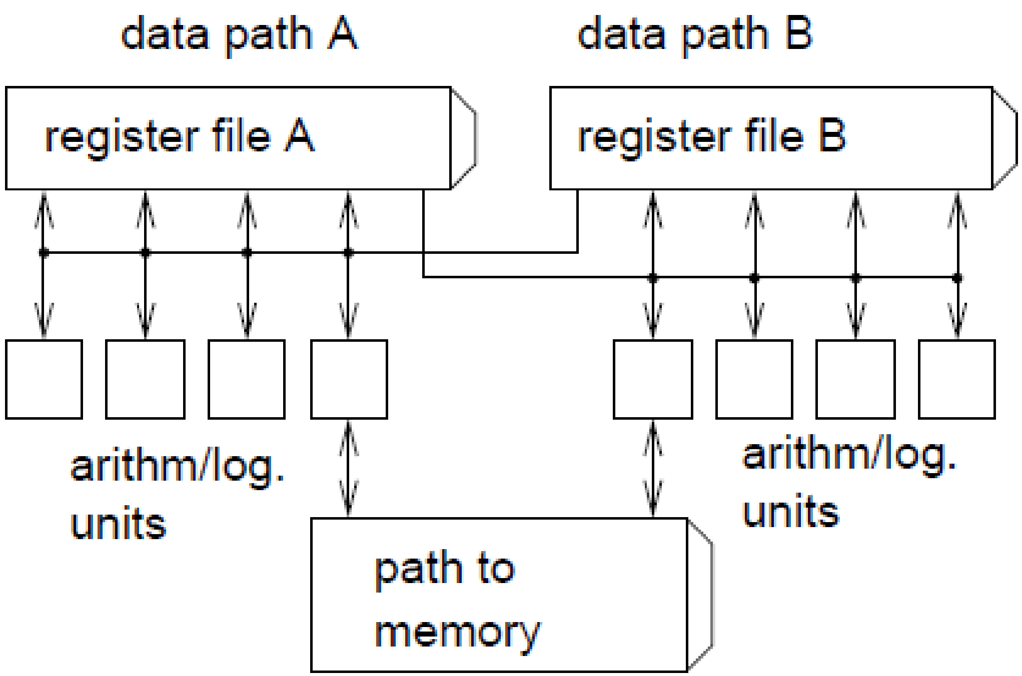
Cycle	Instruction		
1	A		
2	B	C	D
3	E	F	G

Instructions B, C and D use disjoint functional units, cross paths and other data path resources. The same is also true for E, F and G.

Partitioned register files

- Many memory ports are required to supply enough operands per cycle.
- Memories with many ports are expensive.

☞ Registers are partitioned into (typically 2) sets, e.g. for TI C6xxx:



Parallel execution cannot span several packets ☞ IA64

More encoding flexibility with IA-64 Itanium

3 instructions per **bundle:**

127 … 0

instruc 1	instruc 2	instruc 3	template

There are 5 instruction types:

- A: common ALU instructions
- I: more special integer instructions (e.g. shifts)
- M: Memory instructions
- F: floating point instructions
- B: branches

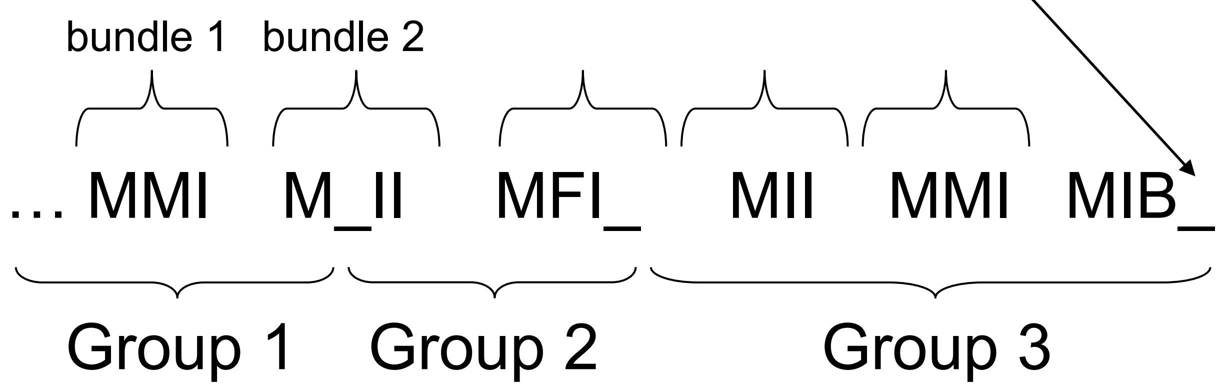
Instruction grouping information

The following combinations can be encoded in templates:

- MII, MMI, MFI, MIB, MMB, MFB, MMF, MBB, BBB, MLX
  with LX = *move 64-bit immediate* encoded in 2 slots

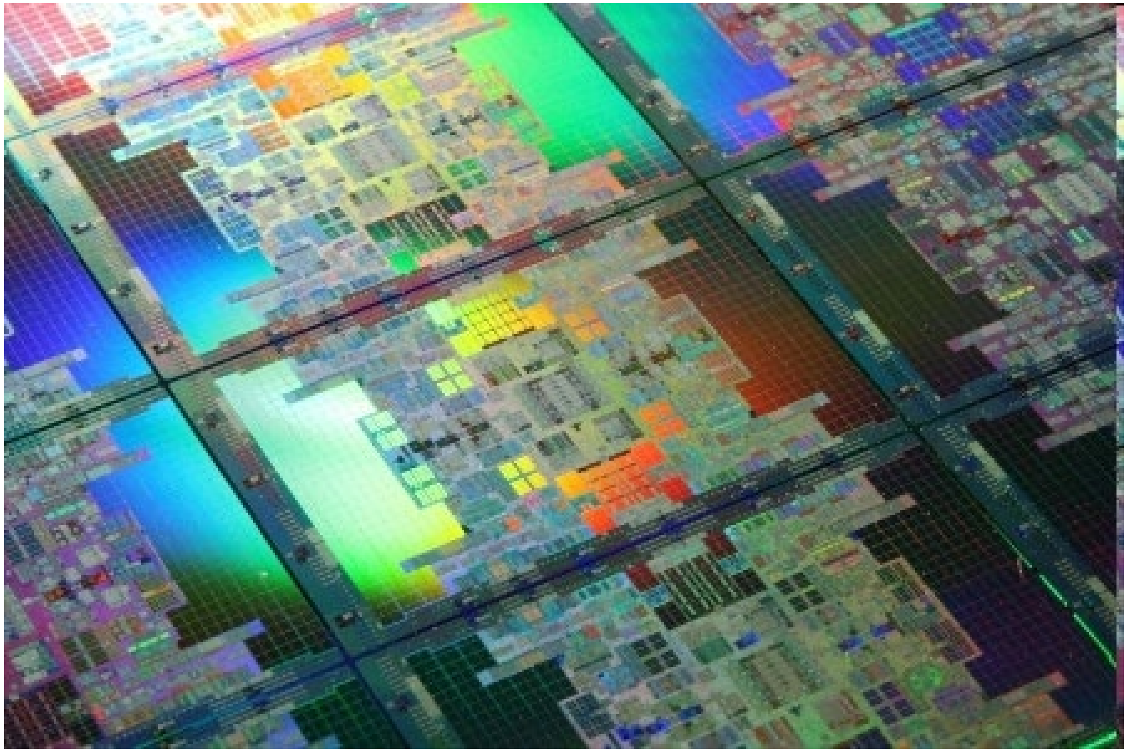
Templates and instruction types

End of parallel execution called **stops.**
Stops are denoted by underscores.
Example:



Very restricted placement of stops within bundle.
Parallel execution within groups possible.
Parallel execution can span several bundles

Itanium® 9300 (Tukwila), 2010

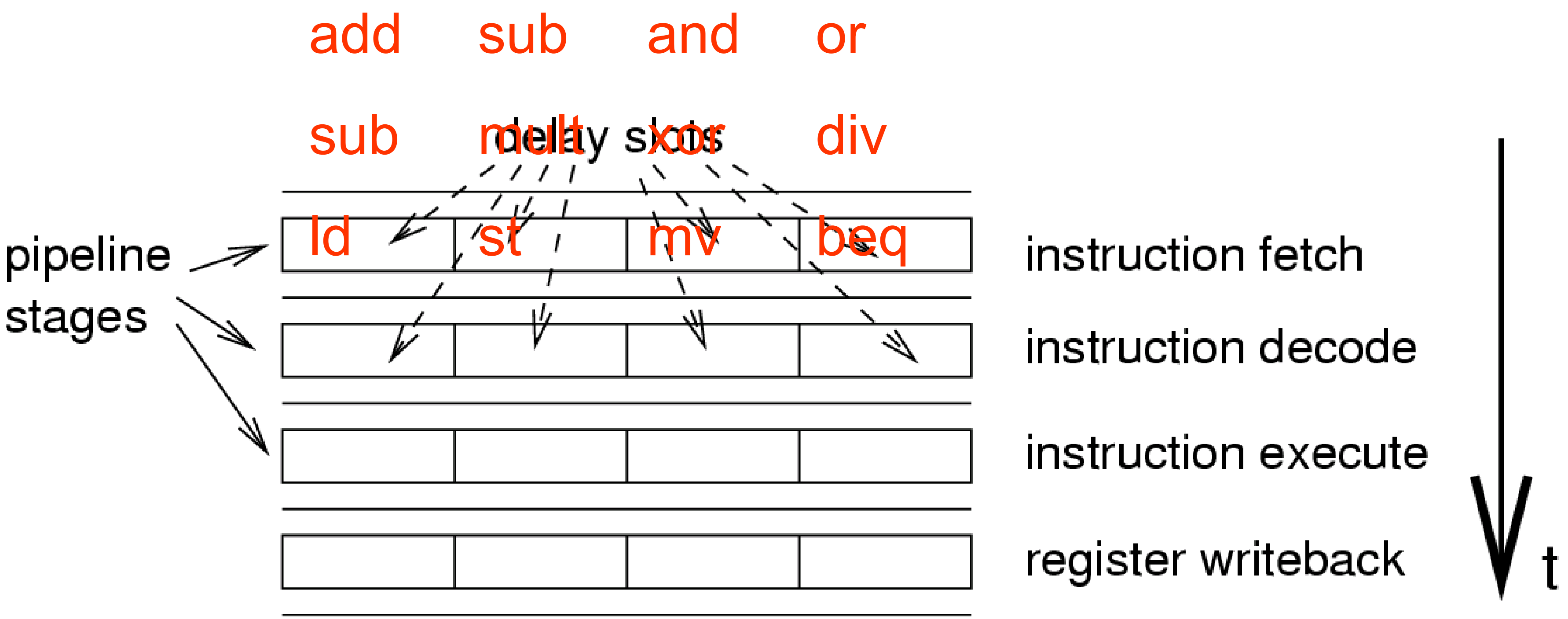


- 2 G transistors
- 4 cores
- 8-fold hyper-threading/core
- 1.5 GHz at 1.3V

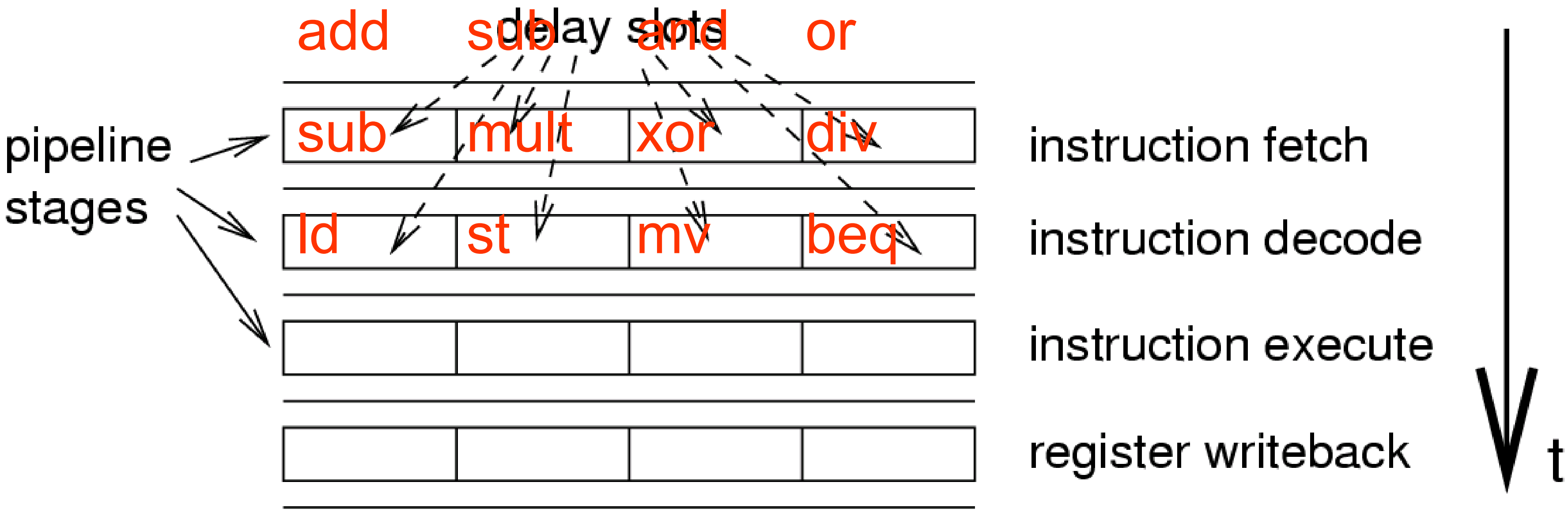
[http://www.intel.com/cd/corporate/pressroom/emea/deu/442093.htm]

© Intel, 2010

Large # of delay slots, a problem of VLIW processors

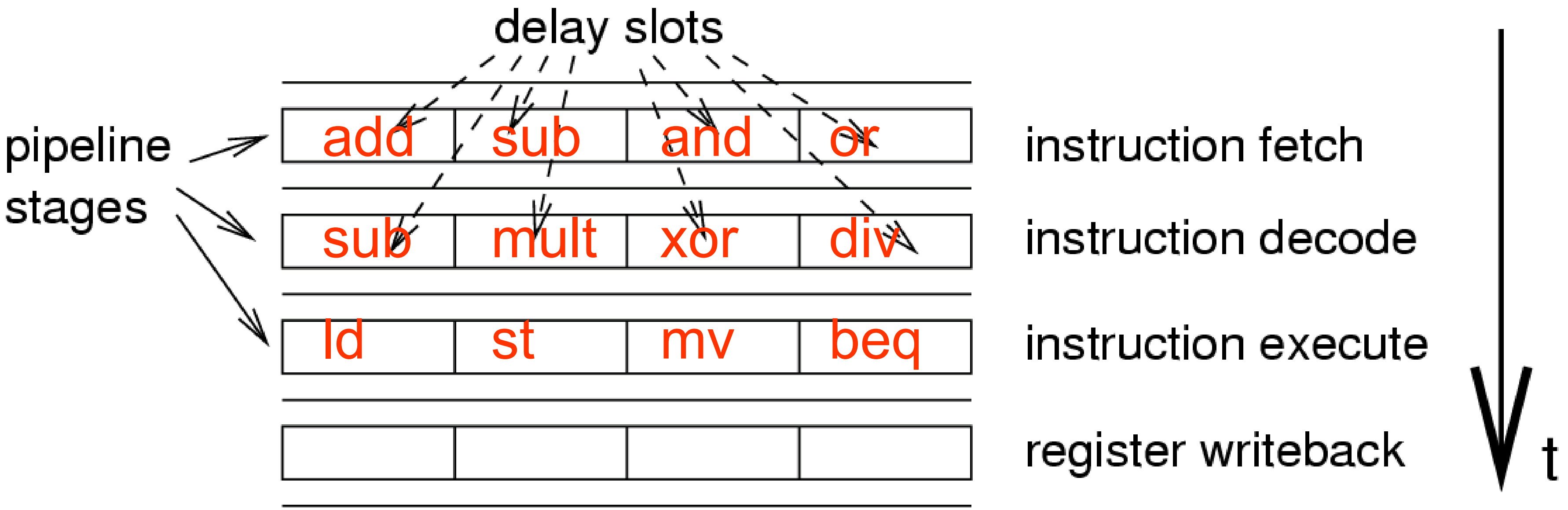


Large # of delay slots, a problem of VLIW processors



Large # of delay slots,
a problem of VLIW processors

The execution of many instructions has been started before it is realized that a branch was required.



Nullifying those instructions would waste compute power

☞ Executing those instructions is declared a feature, not a bug.
☞ How to fill all “delay slots“ with useful instructions?
☞ Avoid branches wherever possible.

Predicated execution: Implementing IF-statements “branch-free“

Conditional Instruction “[c] I“ consists of:

- condition c (some expression involving condition code regs)
- instruction I

c = true ☞ I executed
c = false ☞ NOP

Predicated execution: Implementing IF-statements “branch-free“: TI C6xxx

```
if (c)
{ a = x + y;
  b = x + z;
}
else
{ a = x - y;
  b = x - z;
}
```

Conditional branch

```
         [c] B L1
             NOP 5
             B L2
             NOP 4
             SUB x,y,a
          || SUB x,z,b
L1:          ADD x,y,a
          || ADD x,z,b
L2:
```

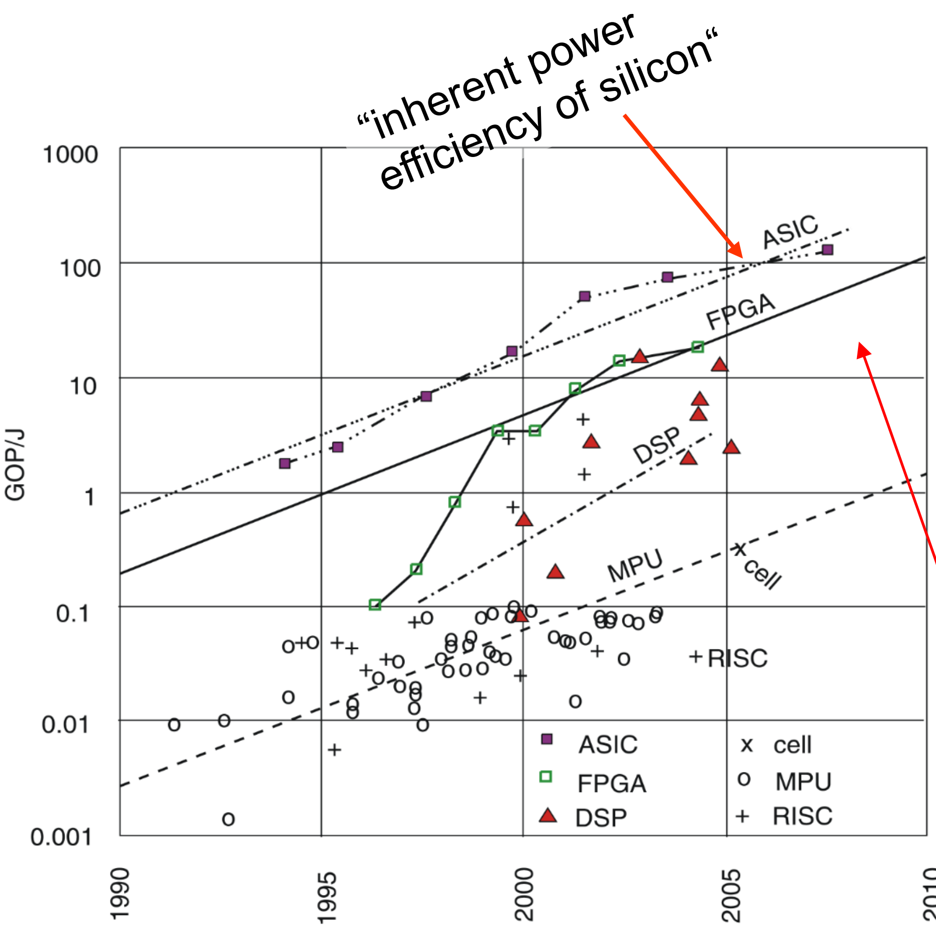
max. 12 cycles

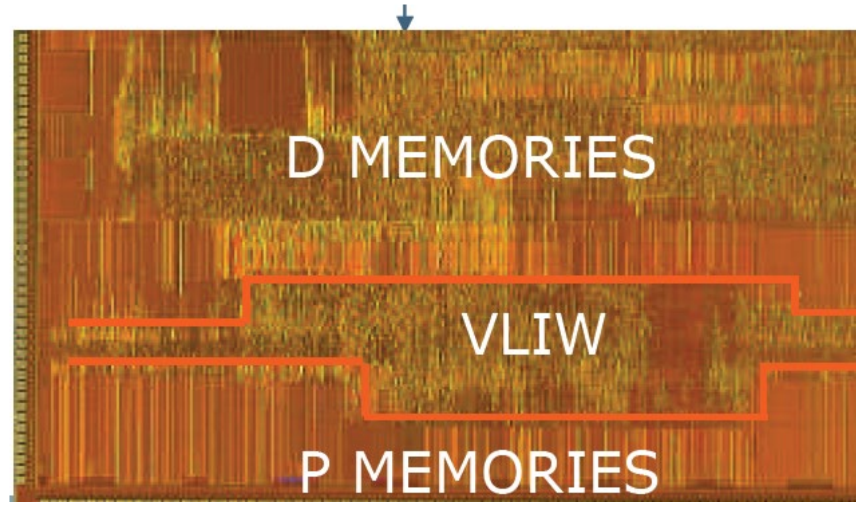
Predicated execution

```
   [c]  ADD x,y,a
|| [c]  ADD x,z,b
|| [!c] SUB  x,y,a
|| [!c] SUB  x,z,b
```

1 cycle

Energy efficiency reached with VLIW processors





© Silicon Hive
41 Issue VLIW for SDR
130 nm, 1,2 V, 6,5mm², 16 bit
30 operations/cycle (OFDM)
150 MHz, 190 mW (incl. SRAMs)
24 GOPs/W, ~1/5 IPE

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

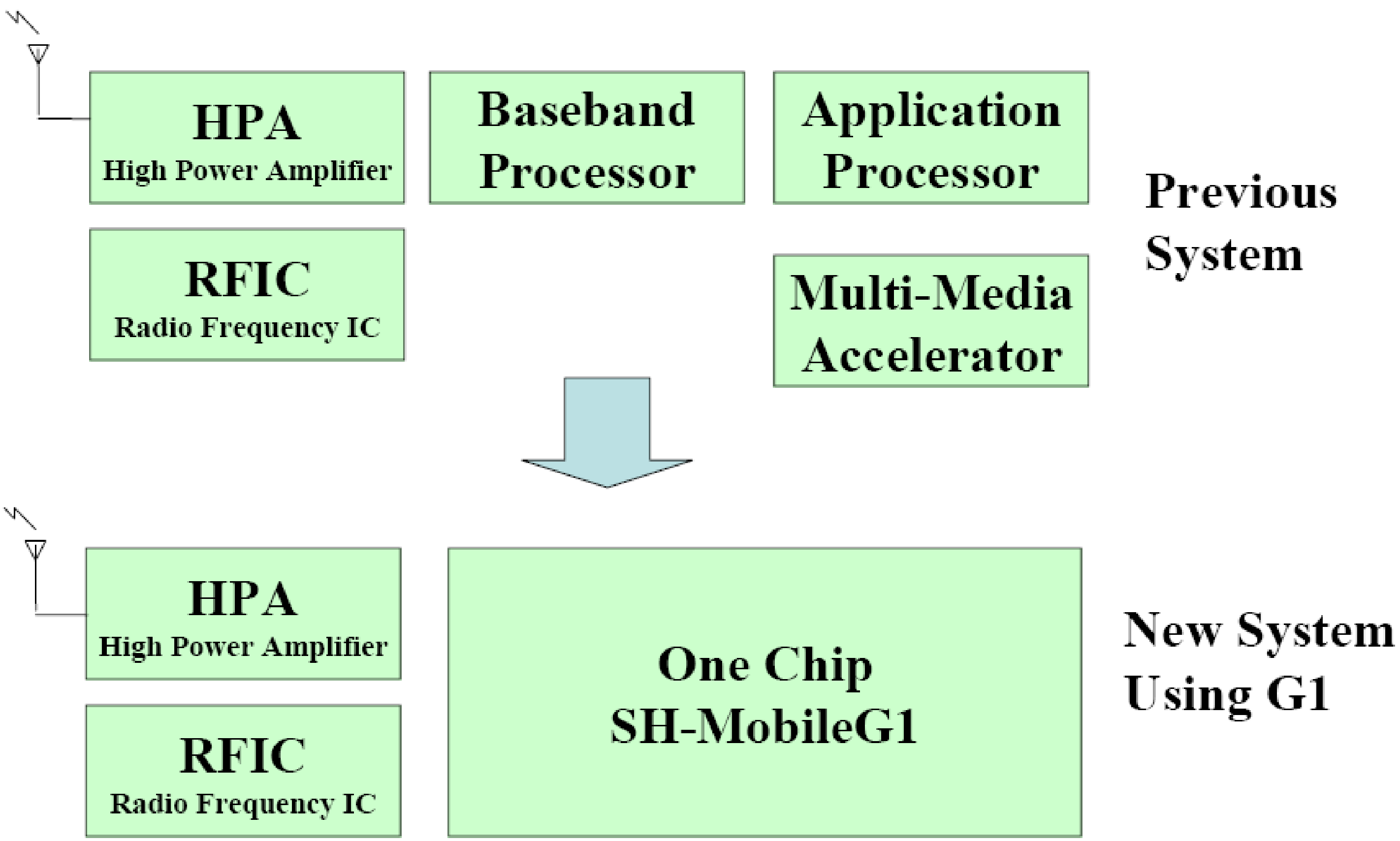
Microcontrollers
- MHS 80C51 as an example -

- 8-bit CPU optimised for control applications
- Extensive Boolean processing capabilities
- 64 k Program Memory address space
- 64 k Data Memory address space
- 4 k bytes of on chip Program Memory
- 128 bytes of on chip data RAM
- 32 bi-directional and individually addressable I/O lines
- Two 16-bit timers/counters
- Full duplex UART
- 6 sources/5-vector interrupt structure with 2 priority levels
- On chip clock oscillators
- Very popular CPU with many different variations

Features 4 Embedded Systems

Trend: multiprocessor systems-on-a-chip (MPSoCs)

3G Multi-Media Cellular Phone System



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

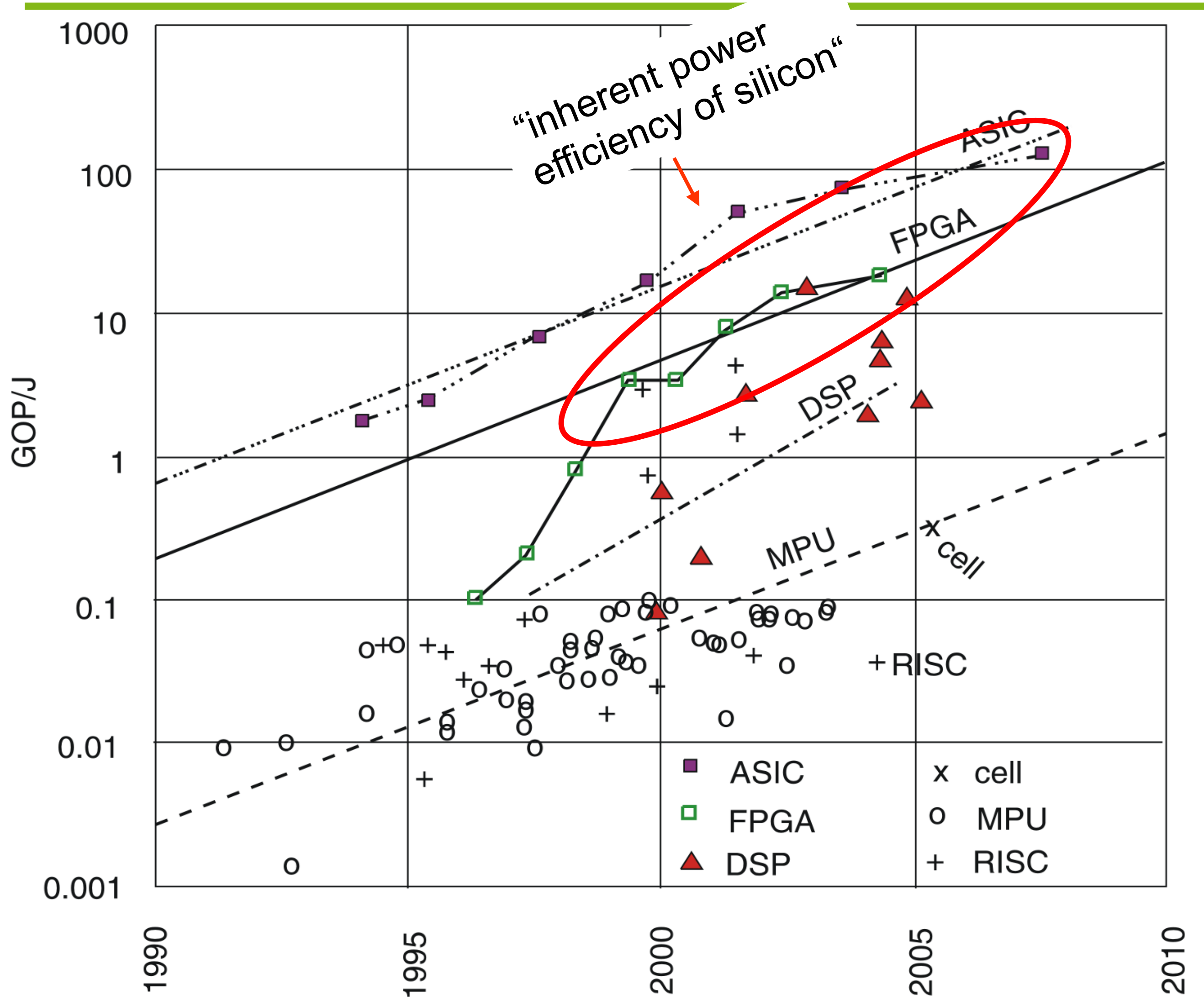
©2007. Renesas Technology Corp., All rights reserved.

Everywhere you imagine. RENESAS

Embedded System Hardware

- Reconfigurable Hardware -

Energy Efficiency of FPGAs



© Hugo De Man, IMEC, Philips, 2007

Reconfigurable Logic

Custom HW may be too expensive, SW too slow.

Combine the speed of HW with the flexibility of SW

- ☞HW with programmable functions and interconnect.
- ☞Use of configurable hardware;
  common form: field programmable gate arrays (FPGAs)

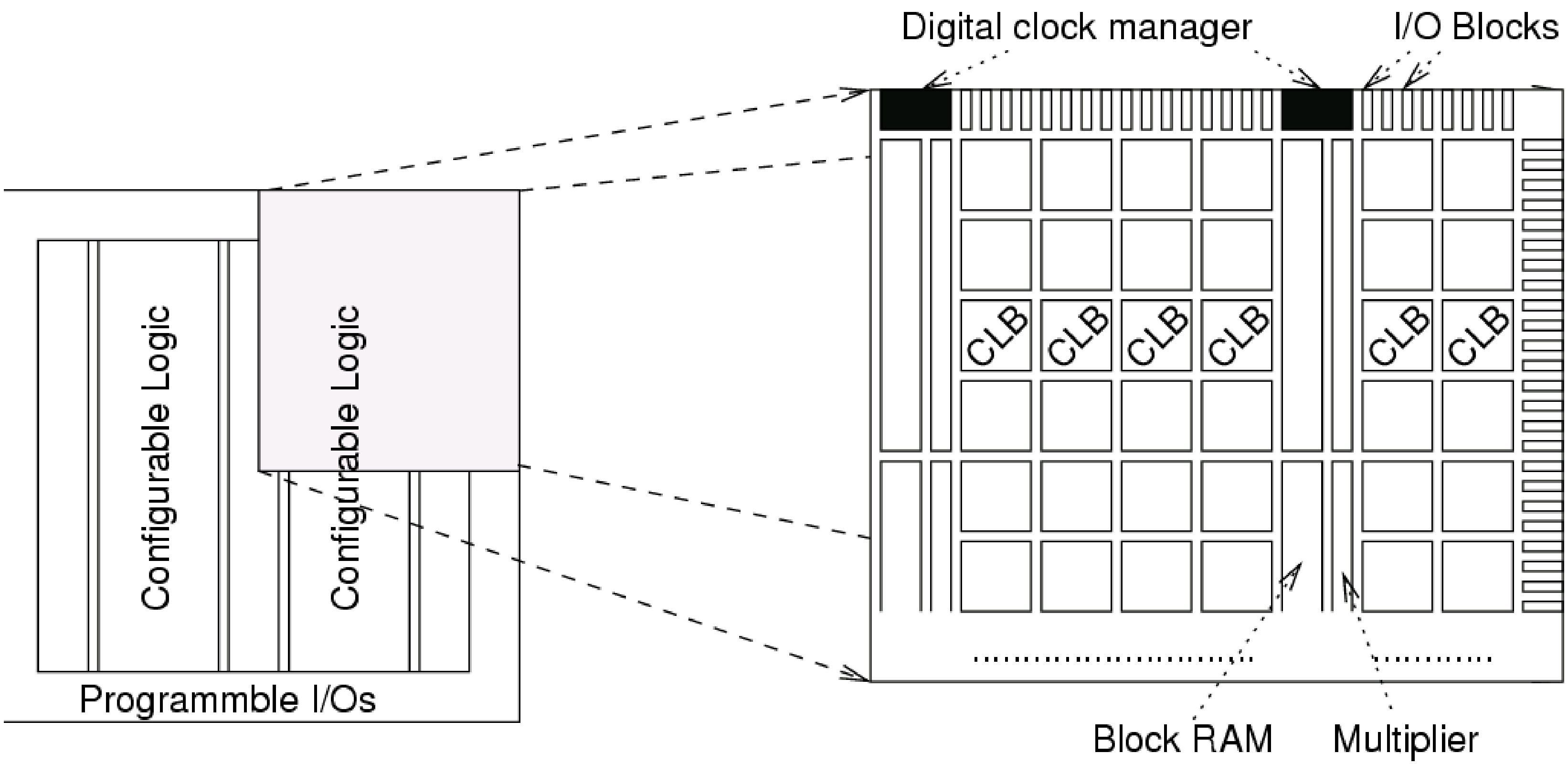
Applications:

- algorithms like de/encryption,
- pattern matching in bioinformatics,
- high speed event filtering (high energy physics),
- high speed special purpose hardware.

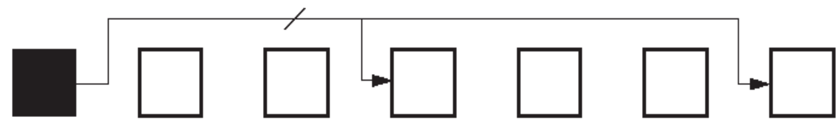
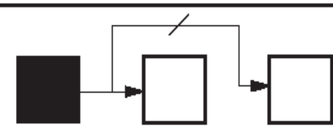
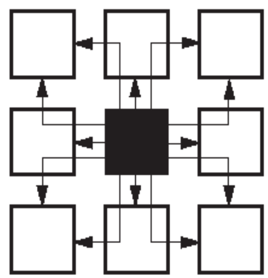
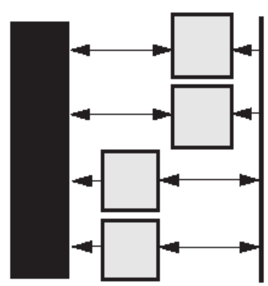
Very popular devices from

- XILINX, Actel, Altera and others

Floor-plan of VIRTEX II FPGAs



Interconnect for Virtex II

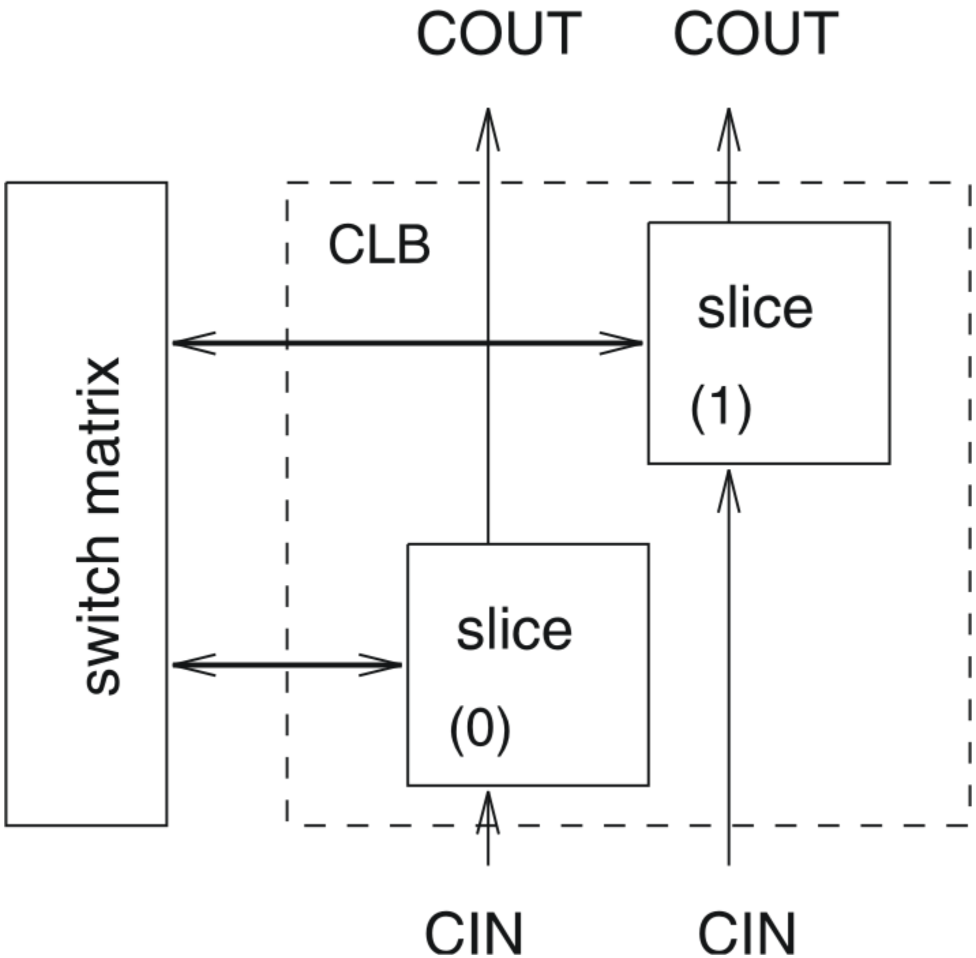
24 Horizontal Long Lines 24 Vertical Long Lines	
120 Horizontal Hex Lines 120 Vertical Hex Lines	
40 Horizontal Double Lines 40 Vertical Double Lines	
16 Direct Connections (total in all four directions)	
8 Fast Connects	

Hierarchical Routing Resources;

More recent: Virtex 5, 6, 7

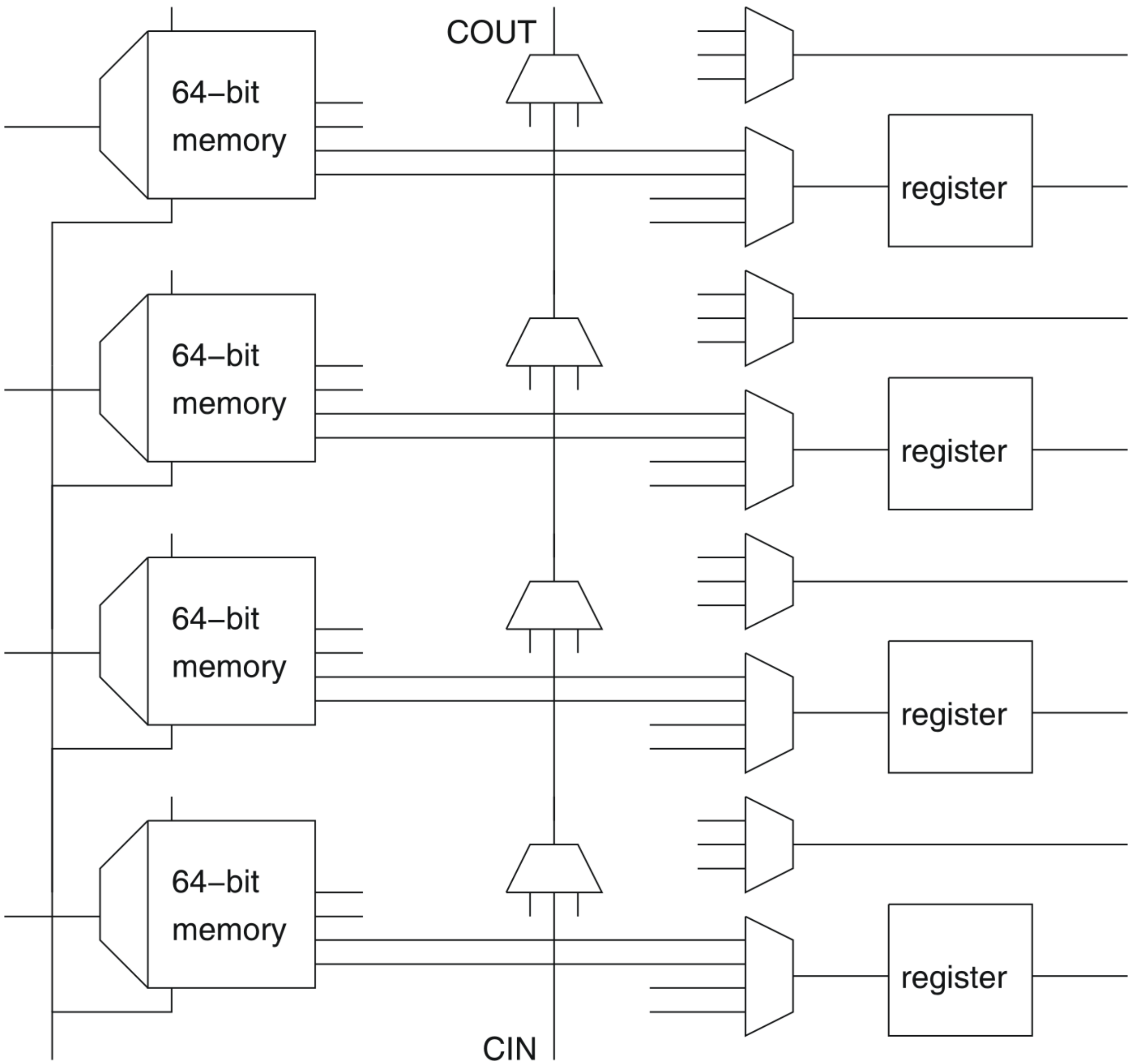
no routing plan found for Virtex 7.

Virtex 7 Configurable Logic Block (CLB)



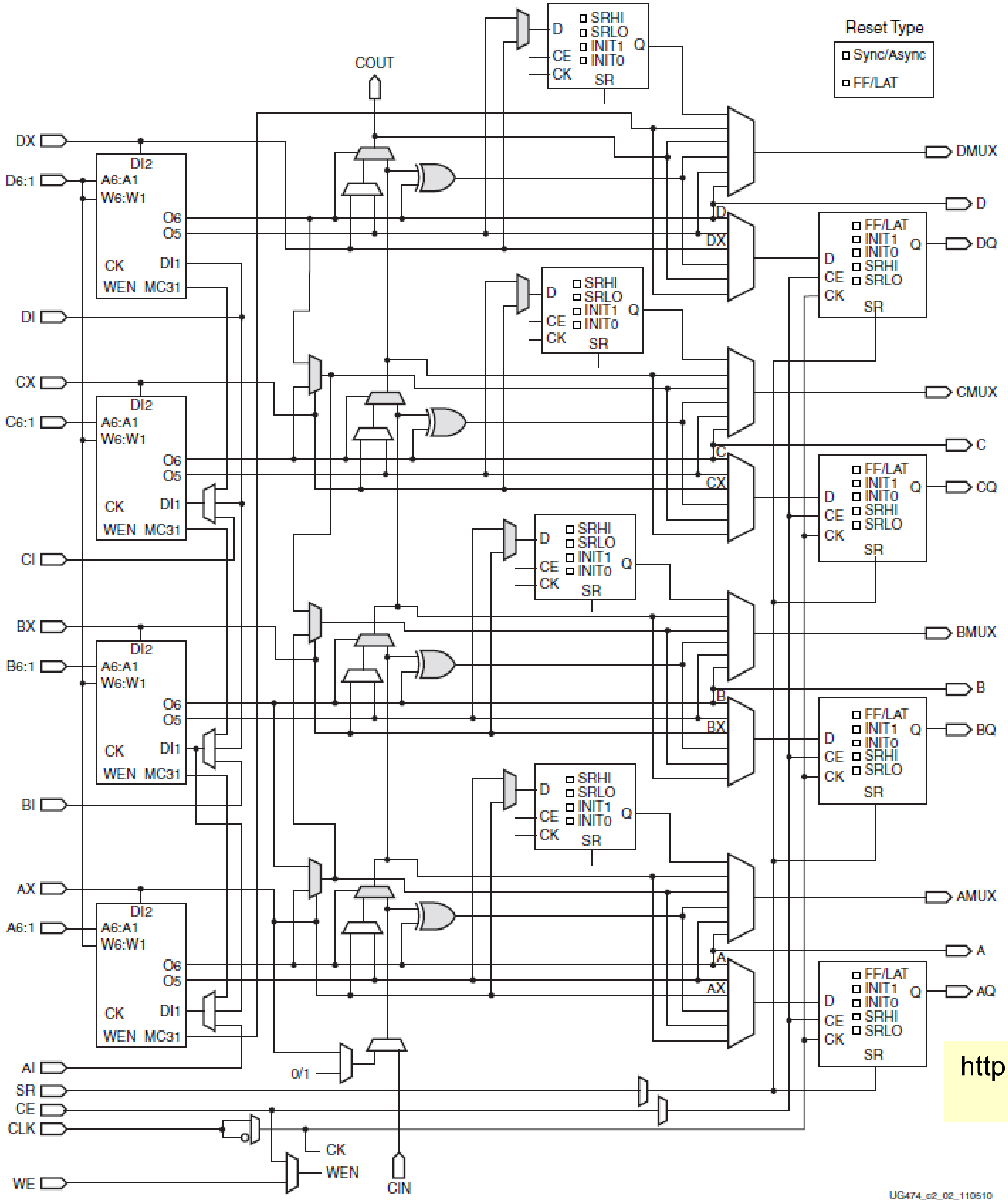
http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

Virtex 7 Slice (simplified)



Memories typically used as look-up tables to implement any Boolean function of ≤ 6 variables.

Processors typically implemented as "soft cores" (microblaze)



Virtex 7 SliceM

SliceM supports using memories for storing data and as shift registers

http://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

© Xilinx

Resources available in Virtex 7 devices (max)

Device	XC7V2000T
Logic cells	1,954,560
Slices	305,400
Max distributed RAM [Kb]	21,550
DSP slices	2,160
Block RAM blocks 18 Kb	2,584
Block RAM blocks 36 Kb	1,292
Block RAM blocks Max [Kb]	46,512
PCIe	4
GTX Transceivers	36
A/D converters	1
User I/O	1,200

[http://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

Virtex-7 FPGAs

Maximum Capability	Virtex-7 T	Virtex-7 XT	Virtex-7 HT
Logic density	1,995 k	1,139 k	864 k
Peak transceiver speed	12.5 Gb/s (GTX)	13.1 Gb/s (GTH)	28.05 Gb/s (GTZ)
Peak bi-directional bandwidth	0.9 Tb/s	2.515 Tb/s	2.784 Tb/s
DSP throughput (symmetric filter)	2,756 G MACS	5,314 GMACS	5,053 GMACS
Block RAM	46.5 Mb	85 Mb	64.4 Mb
I/O pins	1,200	1,100	700

http://www.xilinx.com/products/silicon-devices/fpga/virtex-7/index.htm