Embedded System Hardware

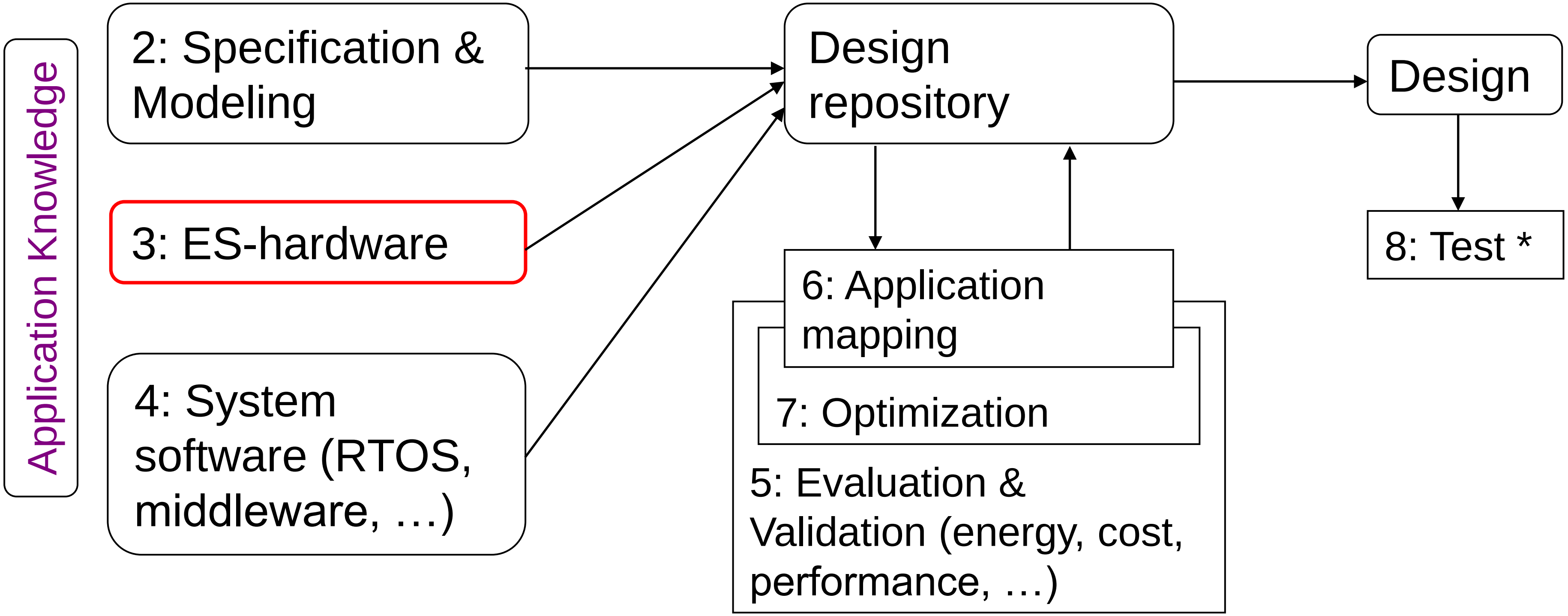
Sensors – Discretization – Information processing

2DT903– Embedded Systems – 5 hp

Hemant Ghayvat
Department of Computer Science and Media Technology
Hemant.ghayvat@lnu.se

Linnæus University

Structure of this course


Application Knowledge
2: Specification & Modeling
3: ES-hardware
4: System software (RTOS, middleware, …)
Design repository
Design
8: Test *
6: Application mapping
7: Optimization
5: Evaluation & Validation (energy, cost, performance, …)

Generic loop: tool chains differ in the number and type of iterations
Numbers denote sequence of chapters

Motivation

"The development of ES cannot ignore the underlying HW characteristics. Timing, memory usage, power consumption, and physical failures are important.“

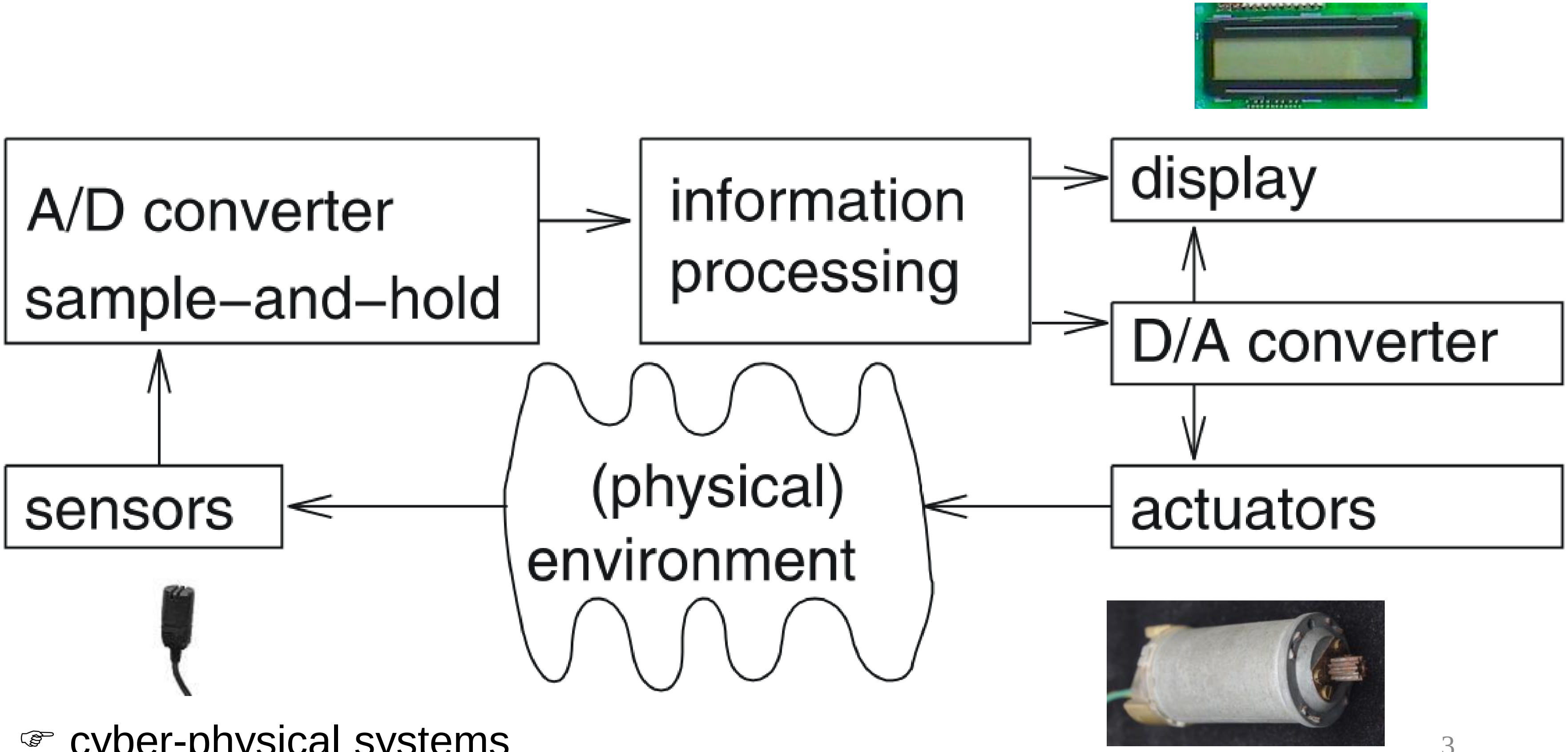
Reasons for considering hard- and software:

- 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

ANALOG-TO-DIGITAL CONVERSION

A digital signal is superior to an analog signal because it is more robust to noise and can easily be recovered, corrected and amplified. For this reason, the tendency today is to change an analog signal to digital data. It is best done by PCM (pulse code modulation).

PCM consists of three steps to digitize an analog signal:

a. Sampling

b. Quantization

c. Binary encoding

Before we sample, we have to filter the signal to limit the maximum frequency of the signal as it affects the sampling rate.

Filtering should ensure that we do not distort the signal, ie remove high frequency components that affect the signal shape.

Many examples of such loops

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

© P. Marwedel, 2011

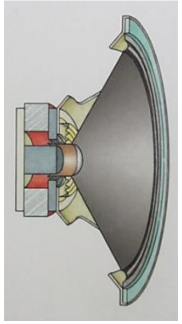
We live in an analog world

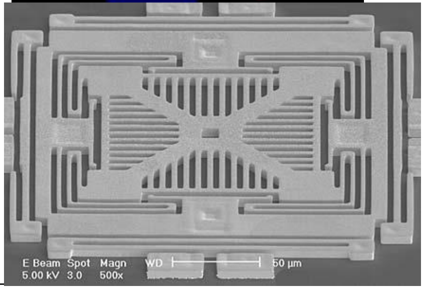
Everything in the physical world is an analog signal

- Sound, light, temperature, pressure

Need to convert into electrical signals

- Transducers: converts one type of energy to another
  - Electro-mechanical, Photonic, Electrical, …
- Examples
  - Microphone/speaker
  - Thermocouples
  - Accelerometers

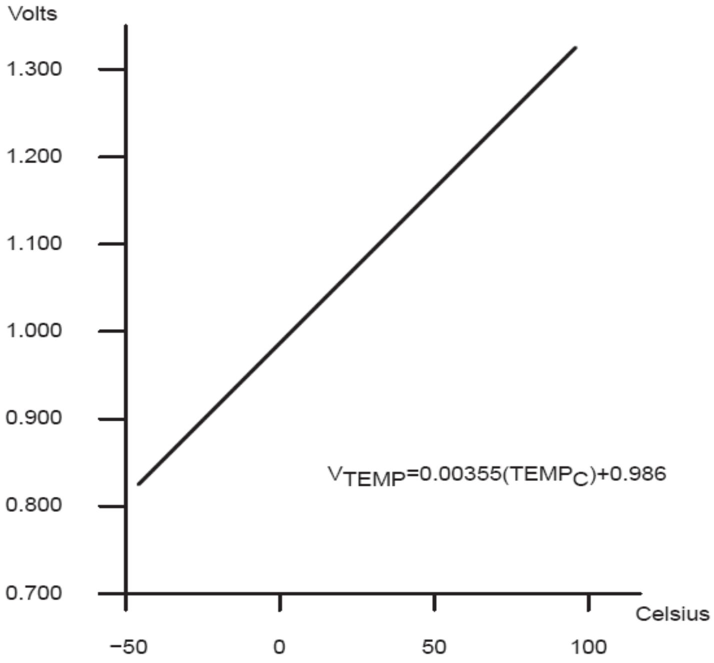




Transducers convert one form of energy into another

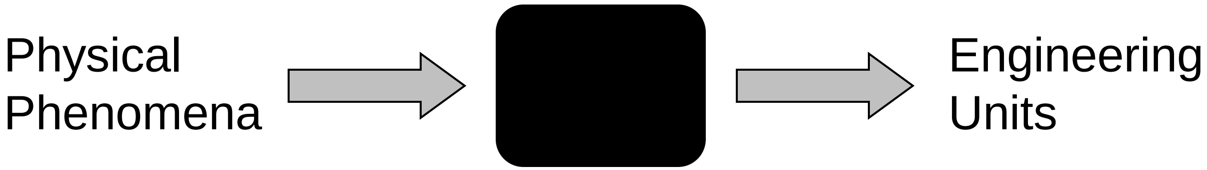
Transducers

- Allow us to convert physical phenomena to a voltage potential in a well-defined way.

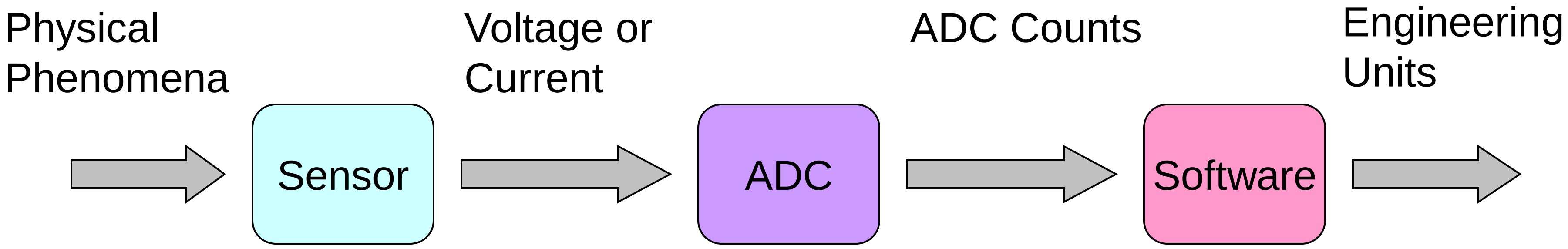


Going from analog to digital

What we want



How we have to get there



Sensors (Input)

- 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.

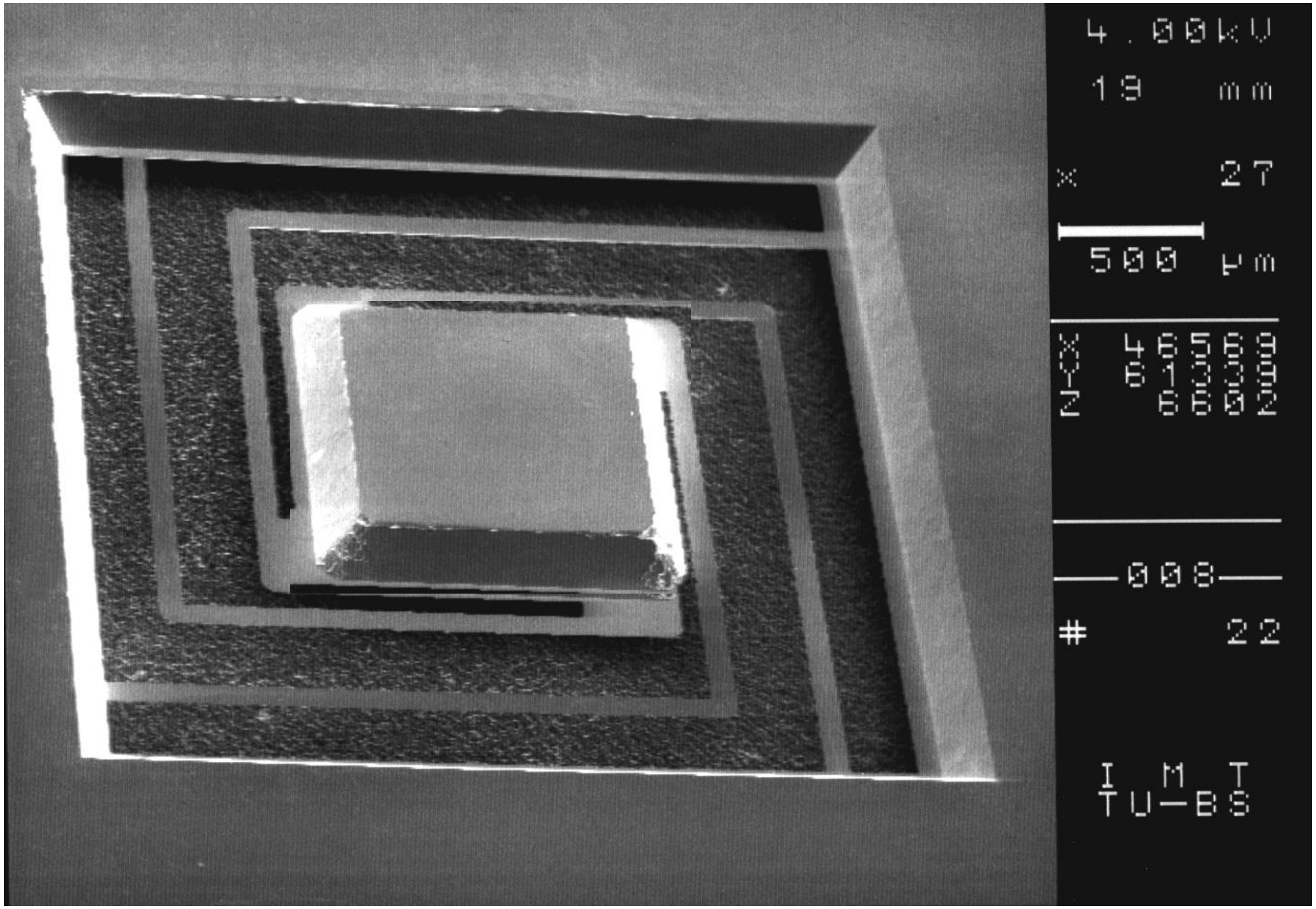
- Many physical effects used for constructing sensors.

  Examples:

  - law of induction (generat. of voltages in a magnetic field),
  - light-electric effects.

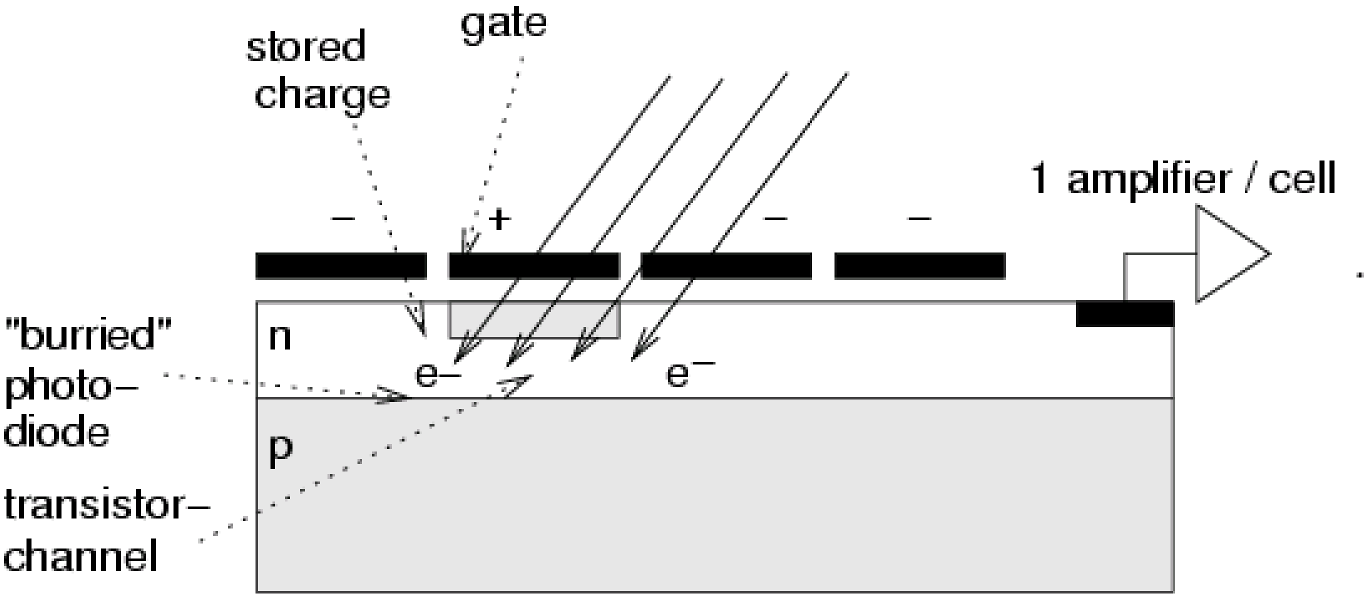
- Huge amount of sensors designed in recent years.

Example: Acceleration Sensor



Charge-coupled devices (CCD) image sensors

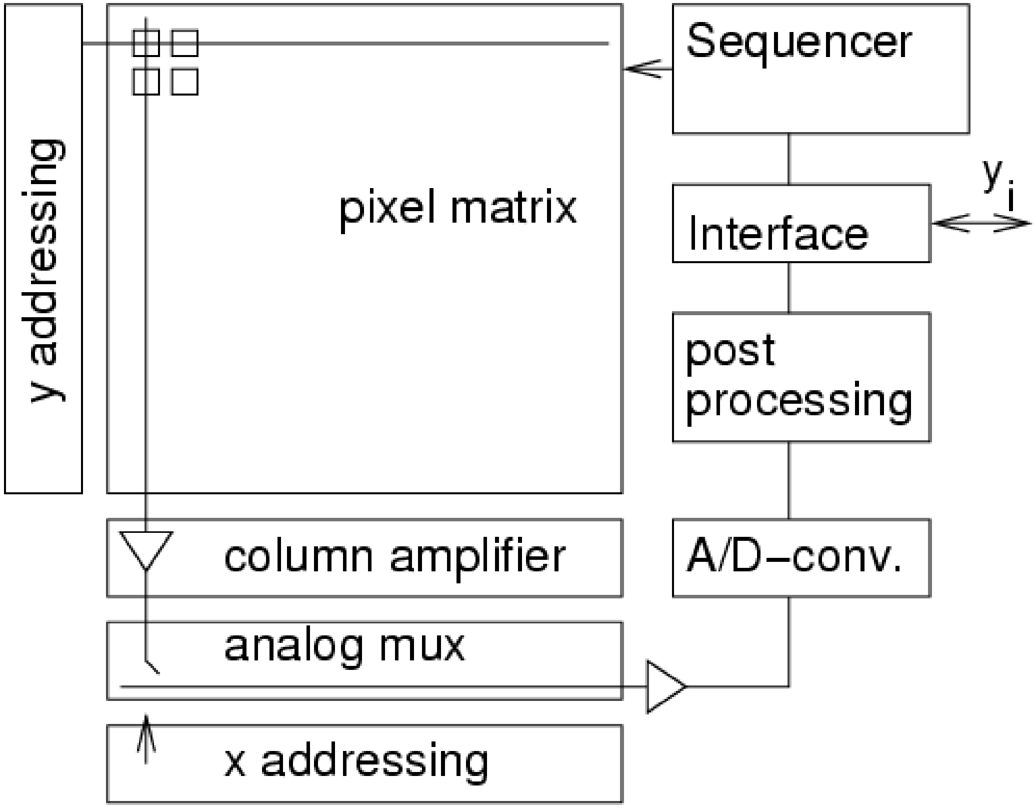
Based on charge transfer to next pixel cell



Corresponding to “bucket brigade device”

CMOS image sensors

Based on standard production process for CMOS chips, allows integration with other components.



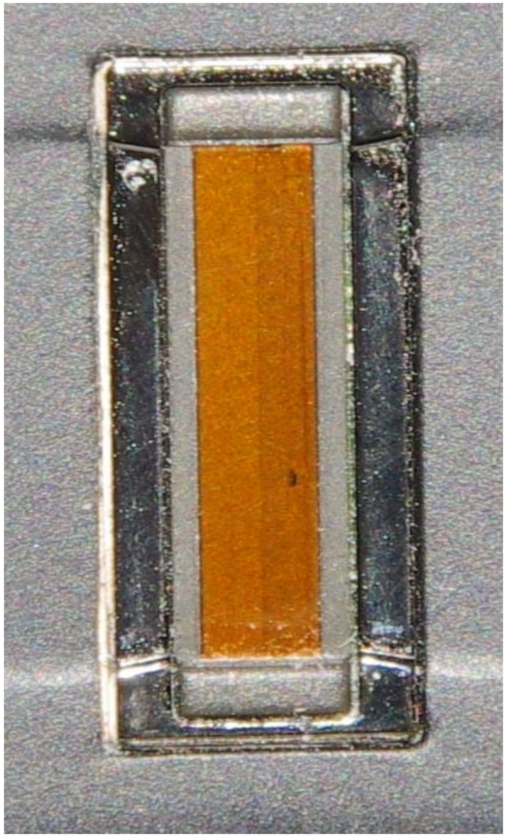
Comparison CCD/CMOS sensors

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	

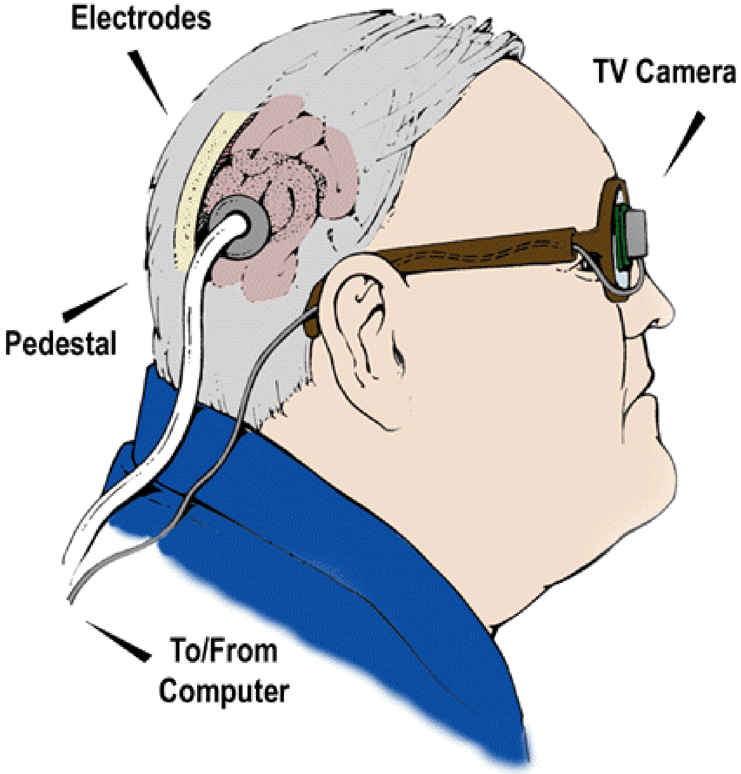
See also B. Diericks: CMOS image sensor concepts. Photonics West 2000 Short course (Web)

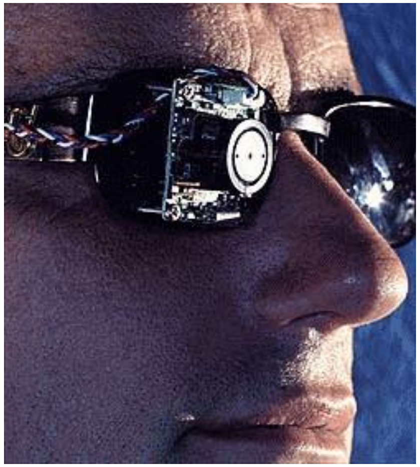
Example: Biometrical Sensors

e.g.: Fingerprint sensor



Artificial eyes (1)





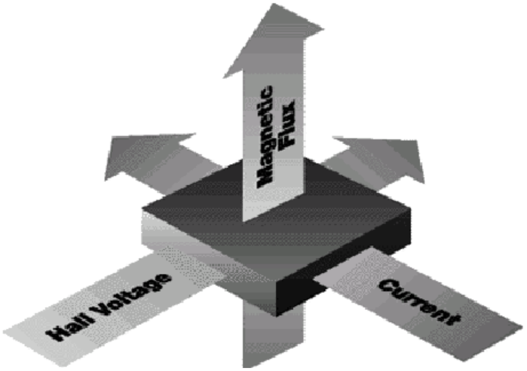
© Dobelle Institute
(was at www.dobelle.com)

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
- Hall effect sensors



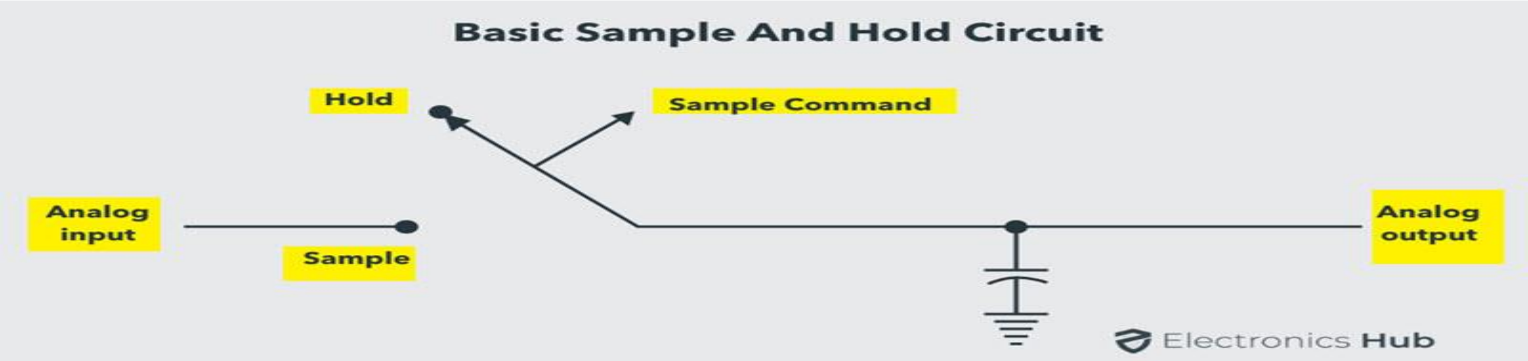
Discretization of time

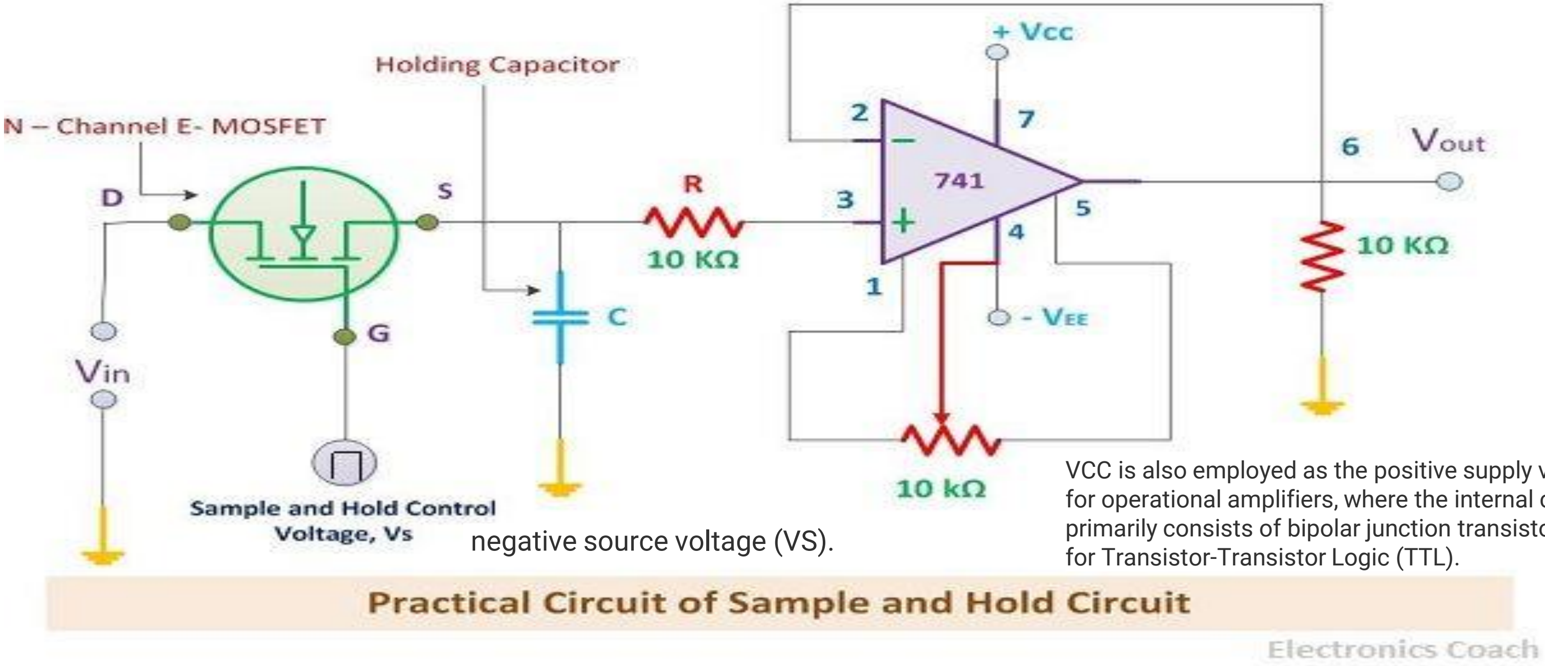
Digital computers require discrete sequences of physical values

$$s : \underset{\substack{\uparrow \\ \text{Discrete time domain}}}{D_T} \rightarrow D_V$$

☞ Sample-and-hold circuits

Sample-and-hold circuits

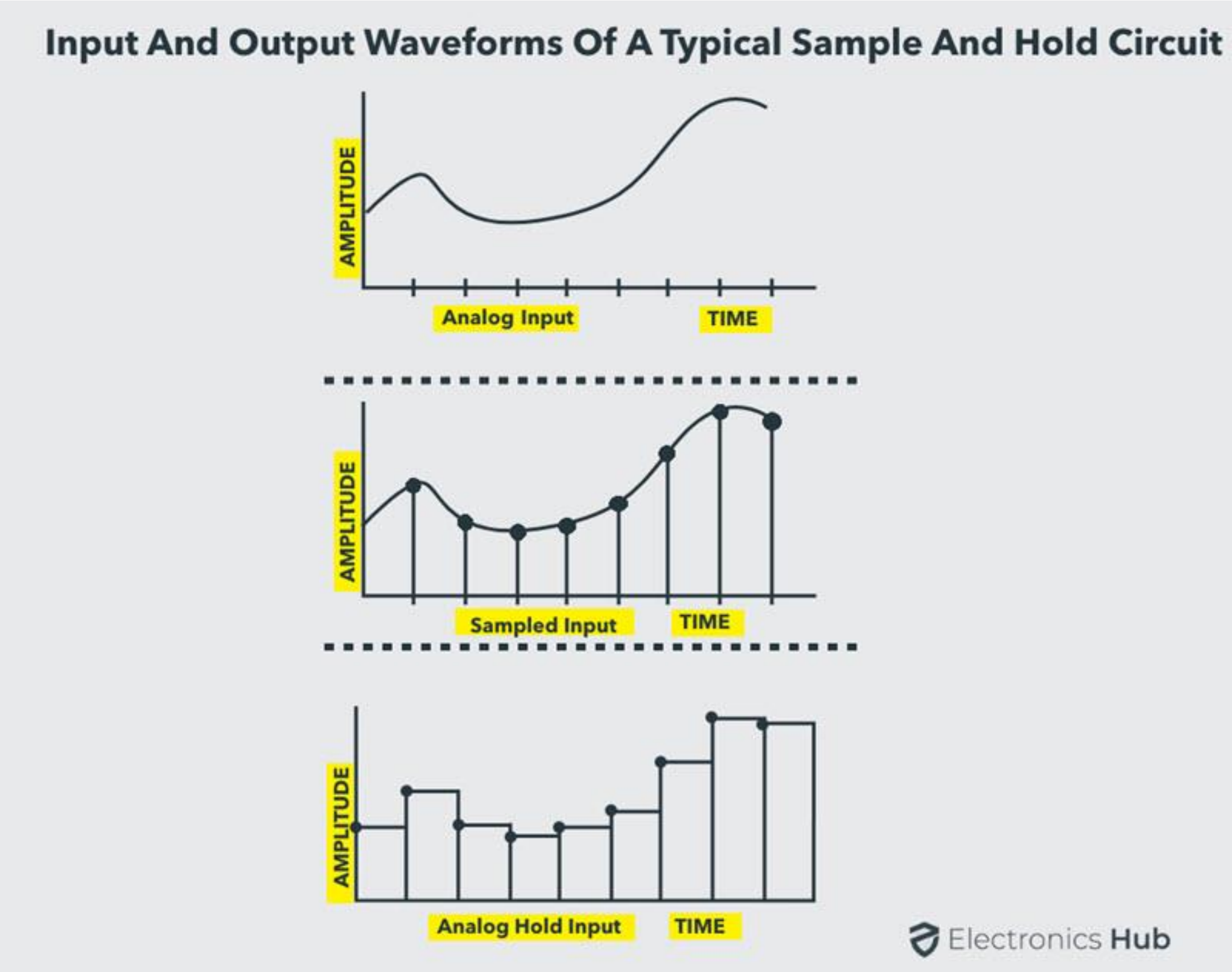




negative source voltage (VS).

VCC is also employed as the positive supply voltage for operational amplifiers, where the internal circuitry primarily consists of bipolar junction transistors, and for Transistor-Transistor Logic (TTL).

Sample-and-hold circuits

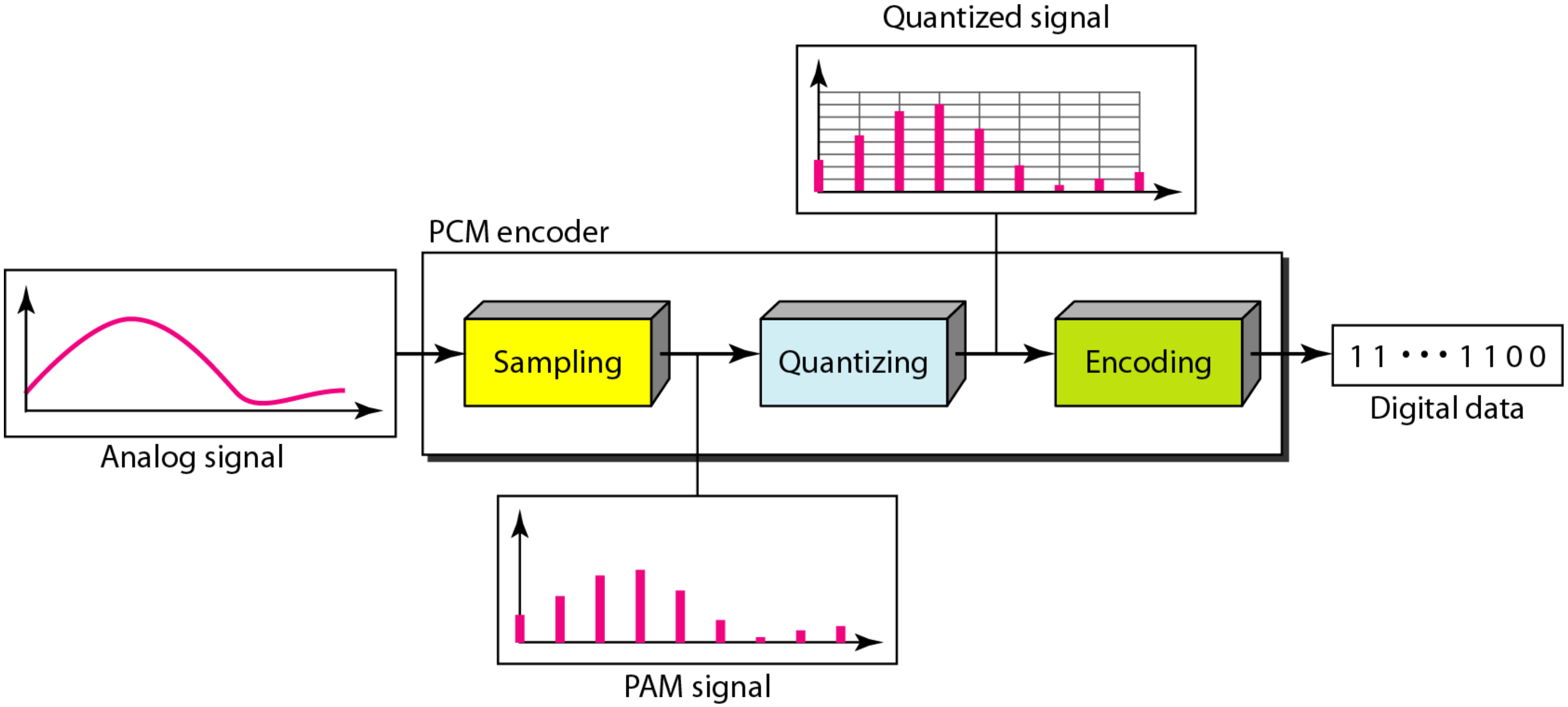


Do we lose information due to sampling?

Careful consideration of the sampling rate and quantization process is necessary to ensure that the digital signal accurately represents the original analog signal.

☞ approximation of signals by sine waves.

Components of PCM encoder



Sampling

Analog signal is sampled every T_S secs.

T_s is referred to as the sampling interval.

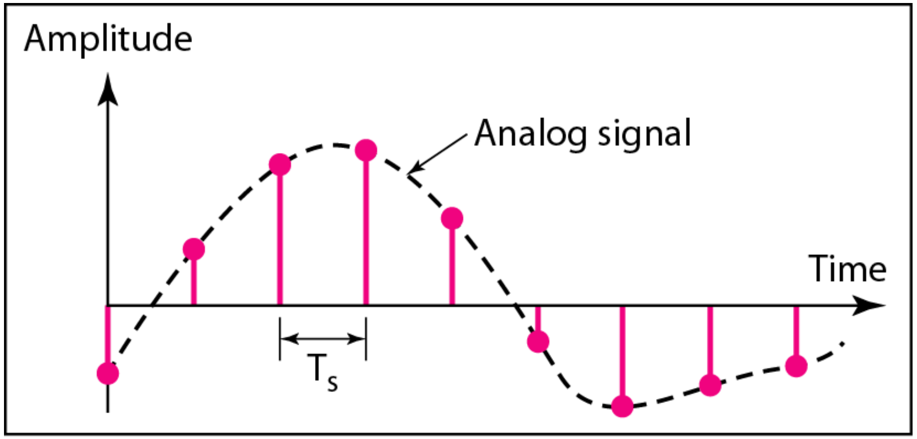
$f_s = 1/T_s$ is called the sampling rate or sampling frequency.

There are 3 sampling methods:

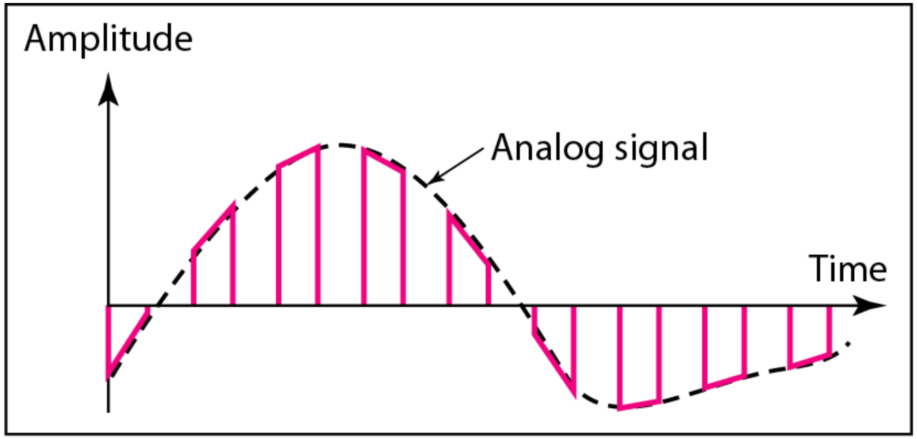
- Ideal - an impulse at each sampling instant
- Natural - a pulse of short width with varying amplitude
- Flattop - sample and hold, like natural but with single amplitude value

The process is referred to as pulse amplitude modulation PAM and the outcome is a signal with analog (non integer) values

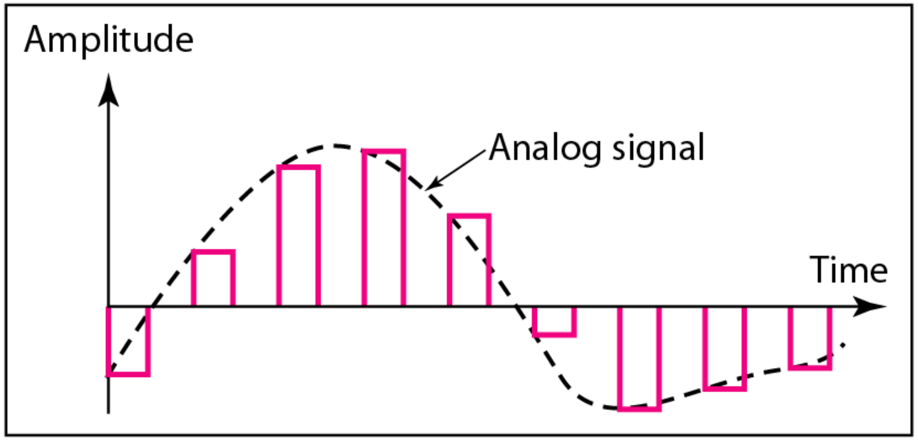
Three different sampling methods for PCM



a. Ideal sampling



b. Natural sampling

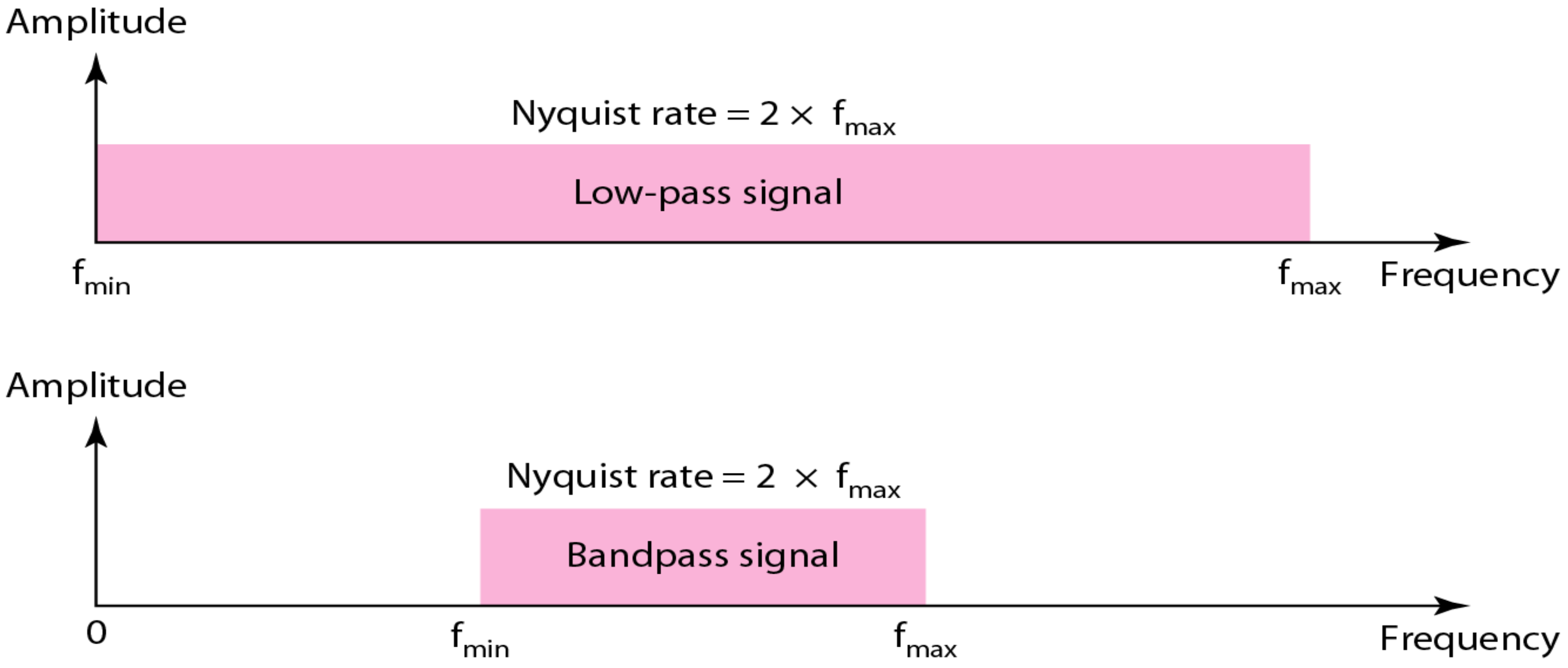


c. Flat-top sampling

Linnæus University

Nyquist theorem (VVIMP)

According to the Nyquist theorem, the sampling rate must be at least 2 times the highest frequency contained in the signal.



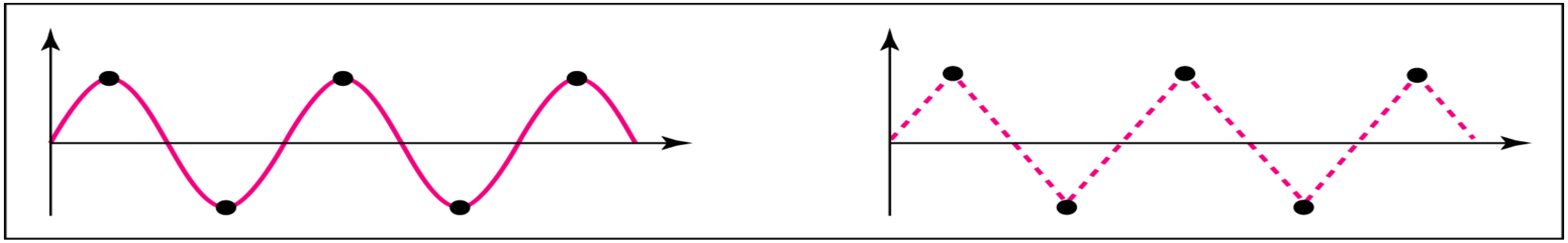
Nyquist sampling rate for low-pass and bandpass signals

Difference between Sampling, Over-Samping and Under Sampling

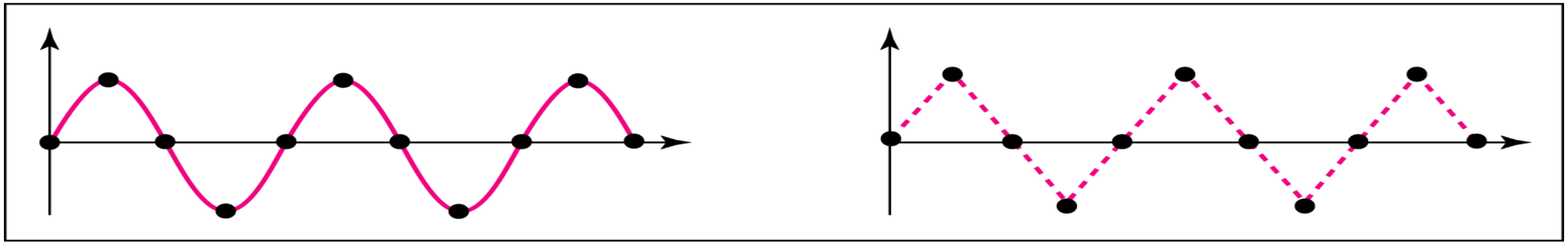
For an intuitive example of the Nyquist theorem, let us sample a simple sine wave at three sampling rates: f_s = 4f (2 times the Nyquist rate), f_s = 2f (Nyquist rate), and f_s = f (one-half the Nyquist rate). Figure shows the sampling and the subsequent recovery of the signal.

It can be seen that sampling at the Nyquist rate can create a good approximation of the original sine wave (part a). Oversampling in part b can also create the same approximation, but it is redundant and unnecessary. Sampling below the Nyquist rate (part c) does not produce a signal that looks like the original sine wave.

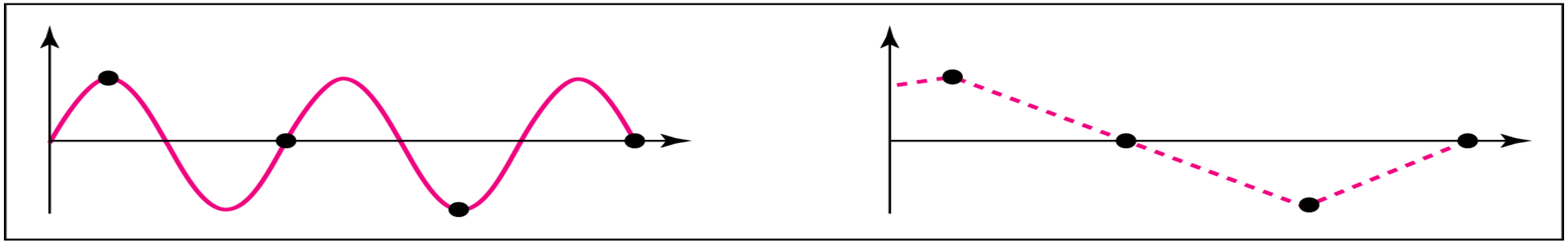
Recovery of a sampled sine wave for different sampling rates



a. Nyquist rate sampling: $f_s = 2\,f$



b. Oversampling: $f_s = 4\,f$



c. Undersampling: $f_s = f$

Quantization

Sampling results in a series of pulses of varying amplitude values ranging between two limits: a min and a max.

The amplitude values are infinite between the two limits.

We need to map the *infinite* amplitude values onto a finite set of known values.

This is achieved by dividing the distance between min and max into L zones, each of height Δ.

$$\Delta = (\max - \min)/L$$

Quantization Levels

The midpoint of each zone is assigned a value from 0 to L-1 (resulting in L values)

Each sample falling in a zone is then approximated to the value of the midpoint.

Quantization zones:

Assume we have a voltage signal with amplitutes V_{min}=-20V and V_{max}=+20V.

We want to use L=8 quantization levels.

Zone width Δ = (20 - -20)/8 = 5

The 8 zones are: -20 to -15, -15 to -10, -10 to -5, -5 to 0, 0 to +5, +5 to +10, +10 to +15, +15 to +20

The midpoints are: -17.5, -12.5, -7.5, -2.5, 2.5, 7.5, 12.5, 17.5

Quantization Levels

Assigning Codes to Zones:

Each zone is then assigned a binary code.

The number of bits required to encode the zones, or the number of bits per sample as it is commonly referred to, is obtained as follows:

$$n_b = \log_2 L$$

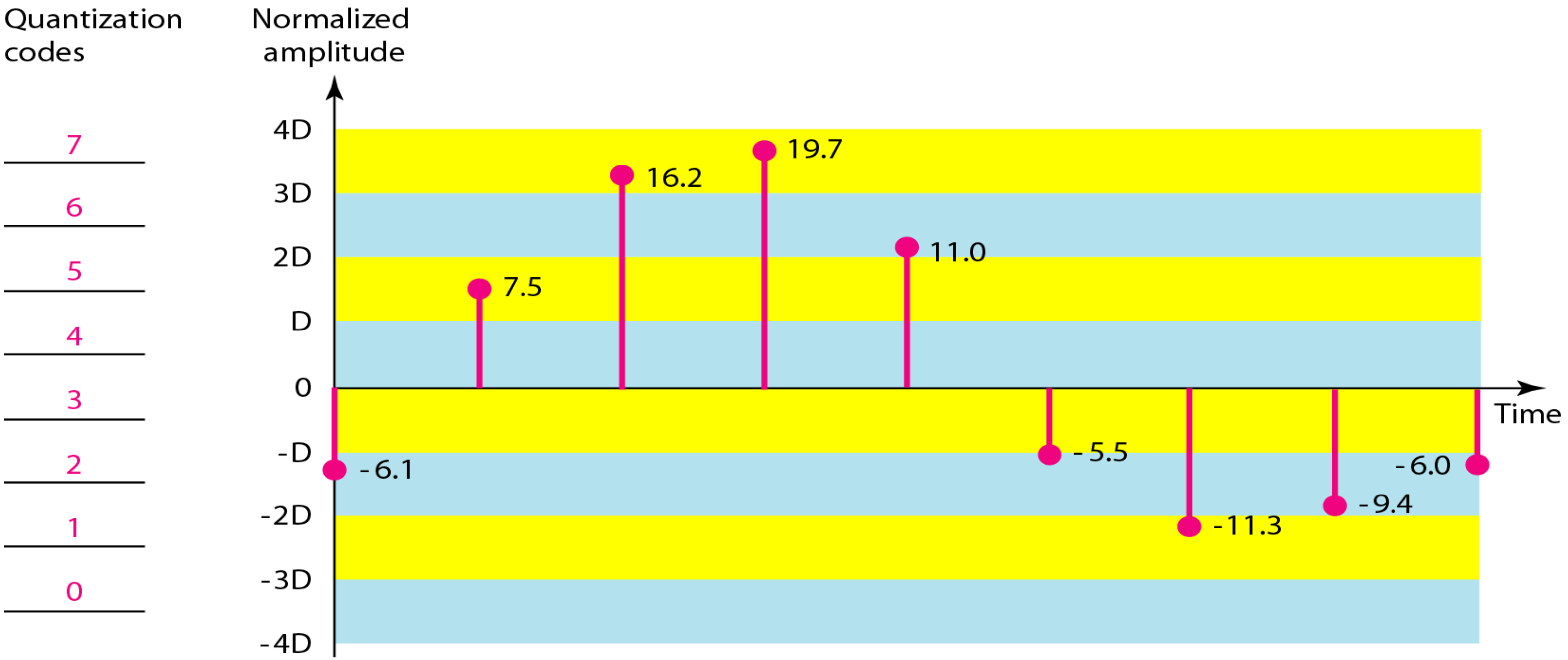
Given our example, $n_b = 3$

The 8 zone (or level) codes are therefore: 000, 001, 010, 011, 100, 101, 110, and 111

Assigning codes to zones:

- 000 will refer to zone -20 to -15
- 001 to zone -15 to -10, etc.

Quantization and encoding of a sampled signal



Normalized PAM values	-1.22	1.50	3.24	3.94	2.20	-1.10	-2.26	-1.88	-1.20
Normalized quantized values	-1.50	1.50	3.50	3.50	2.50	-1.50	-2.50	-1.50	-1.50
Normalized error	-0.38	0	+0.26	-0.44	+0.30	-0.40	-0.24	+0.38	-0.30
Quantization code	2	5	7	7	6	2	1	2	2
Encoded words	010	101	111	111	110	010	001	010	010

Quantization Error

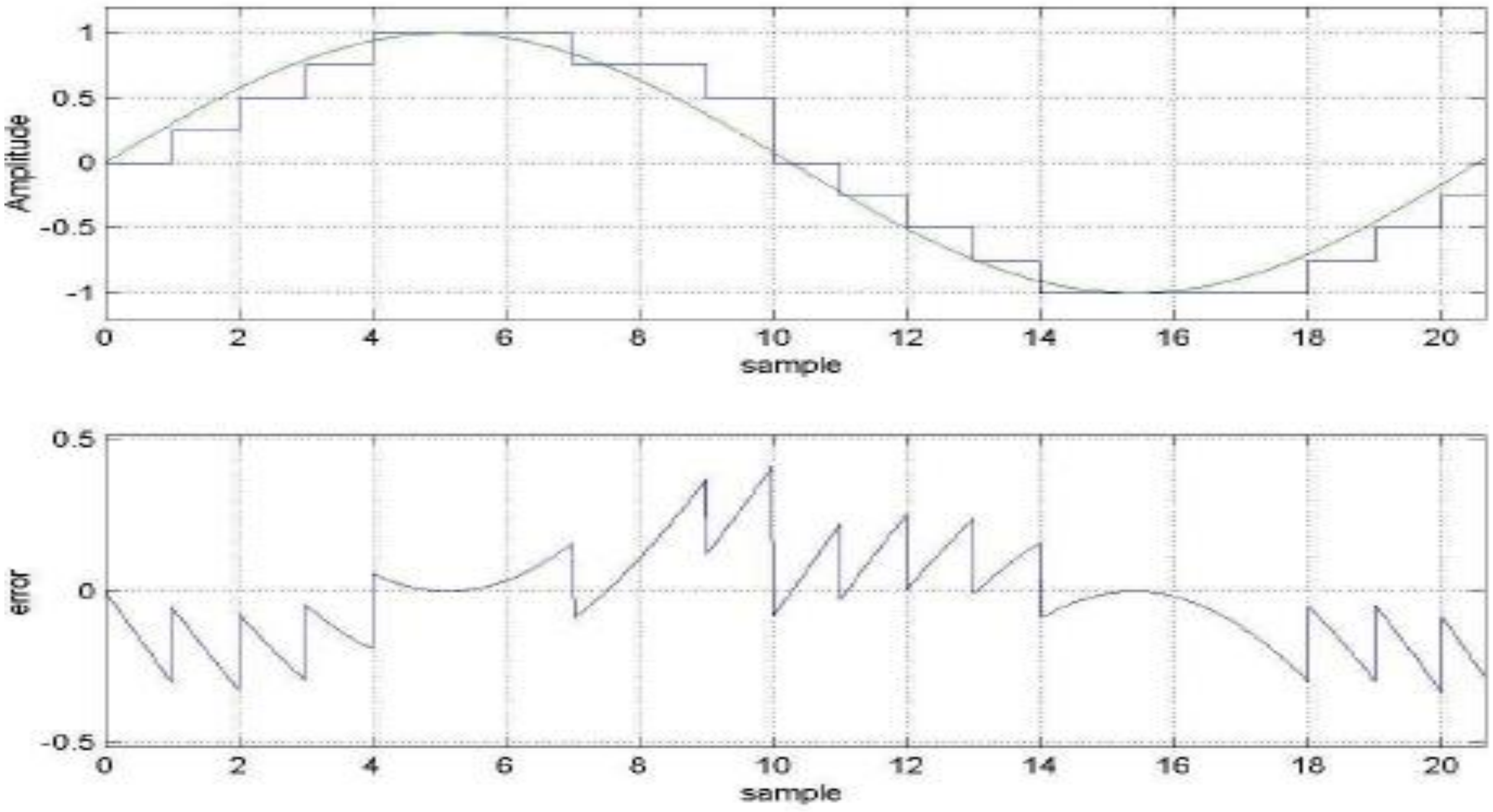
-When a signal is quantized, we introduce an error - the coded signal is an approximation of the actual amplitude value.

-The difference between actual and coded value (midpoint) is referred to as the quantization error.

-The more zones, the smaller **Δ** which results in **smaller errors**.

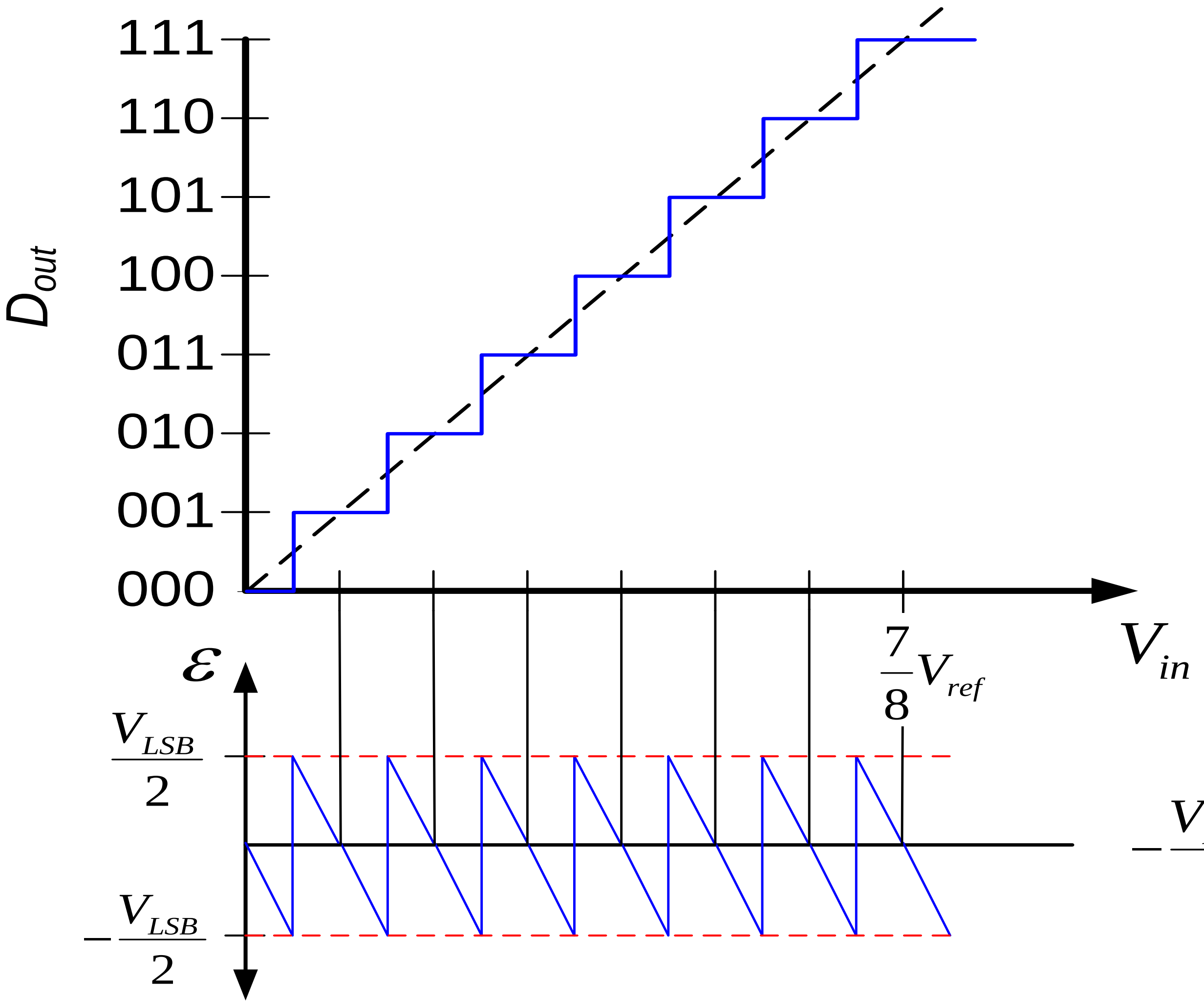
-BUT, the more zones the more bits required to encode the samples -> higher bit rate

Quantization Error



Eugenio Di Gioia, *Sigma-Delta-A/D-Wandler*, 2007

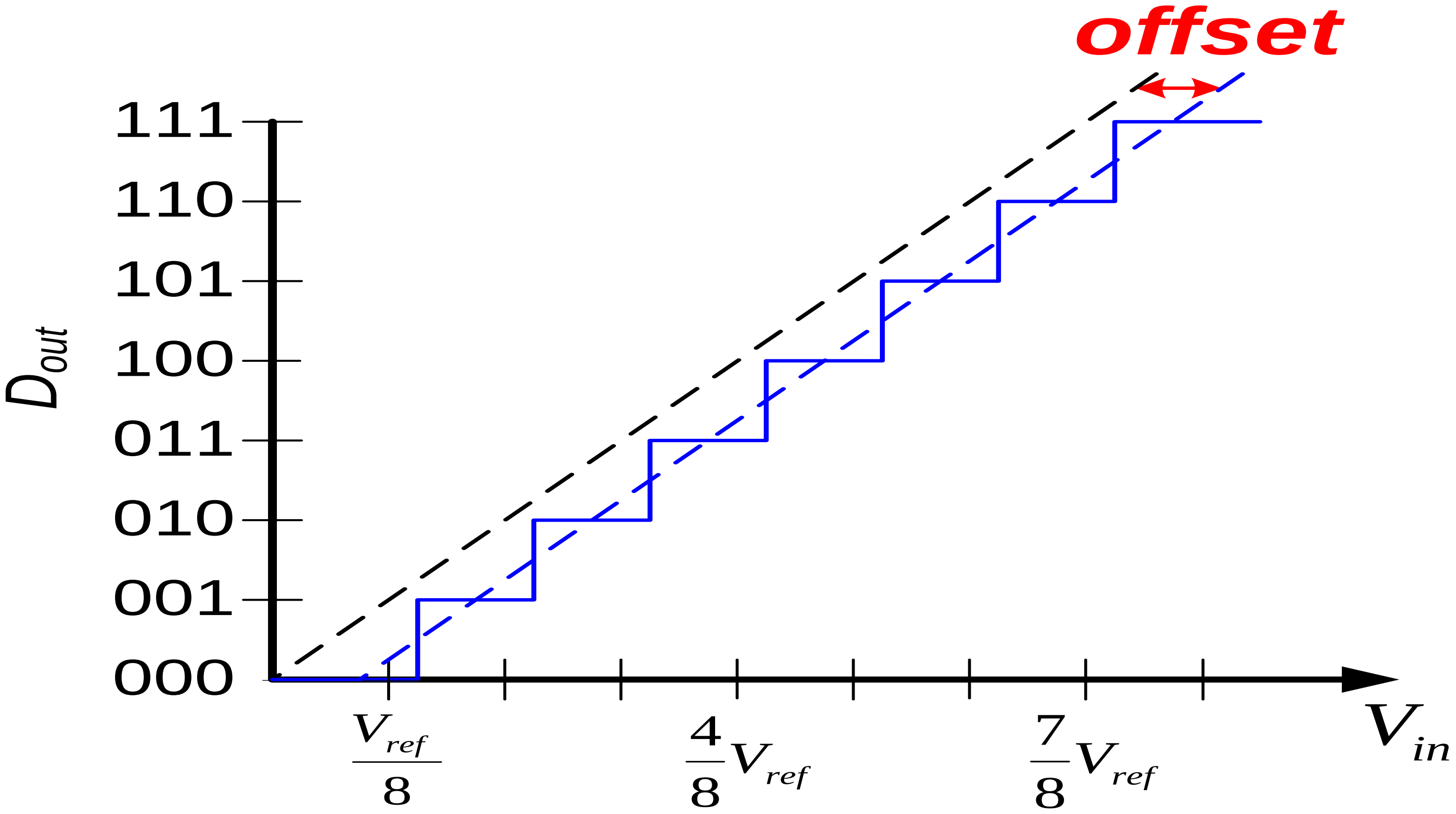
Quantization Error ε



$$-\frac{V_{LSB}}{2} \leq \varepsilon < \frac{V_{LSB}}{2}$$

Sill 2008, @copyright

Offset Error

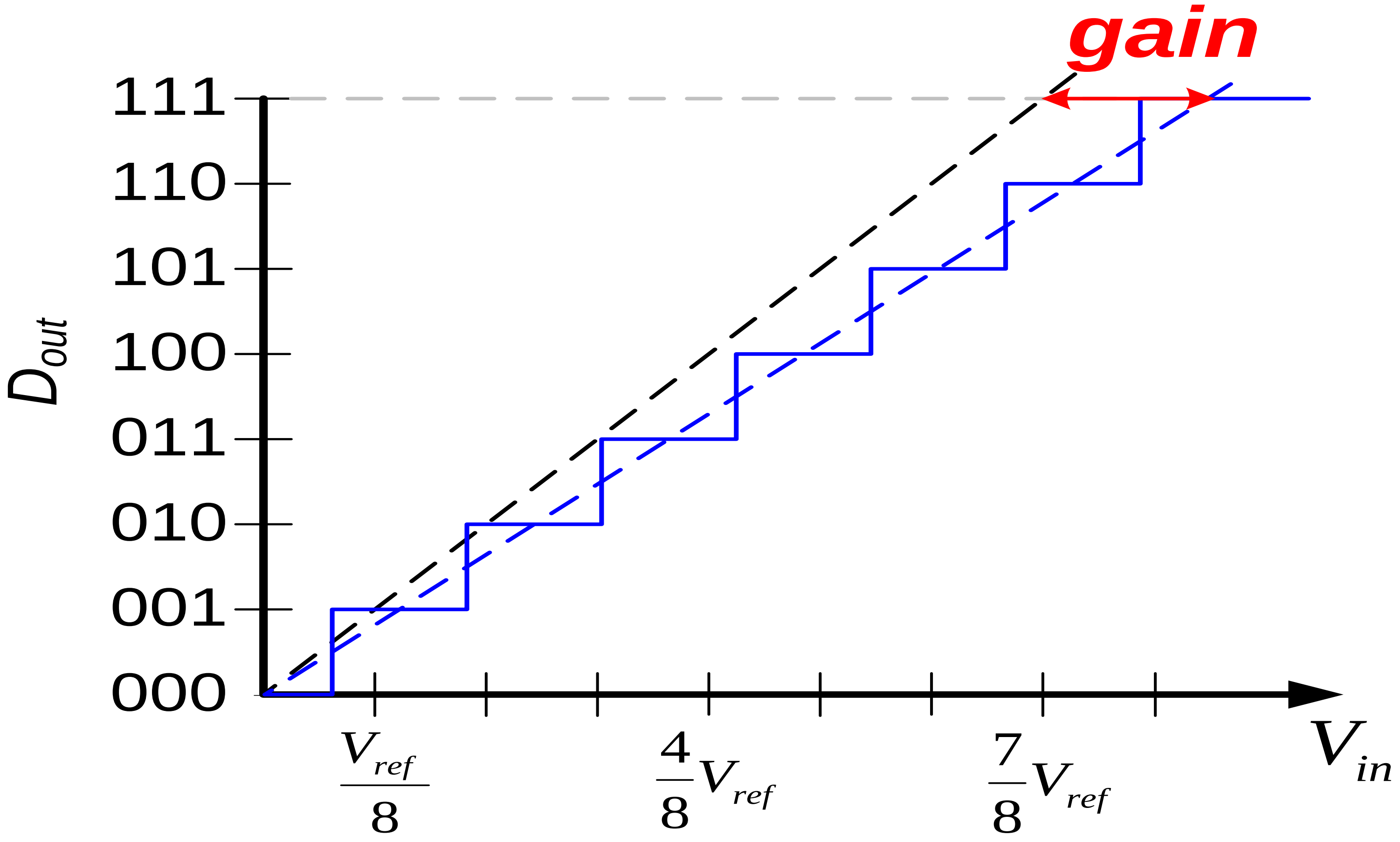


Parallel shift of the whole curve

E.g. caused by difference in ground line voltages

Sill 2008, @copyright

Gain Error

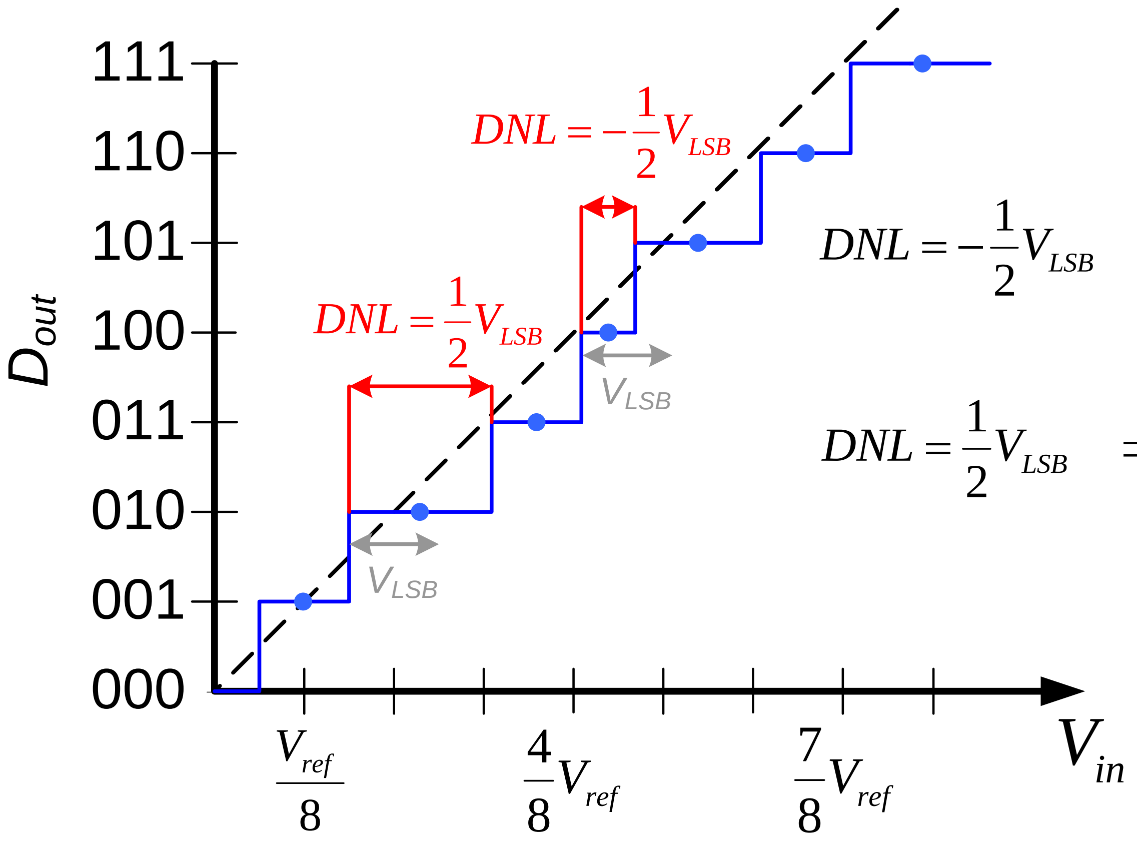


Corresponds to too small or to large but equal Δ

E.g. caused by too small or too large V_{ref}

Sill 2008, @copyright

Differential Non-Linearity (DNL)



$$DNL = -\frac{1}{2}V_{LSB} \quad \Rightarrow \quad \Delta V = \frac{1}{2}V_{LSB}$$

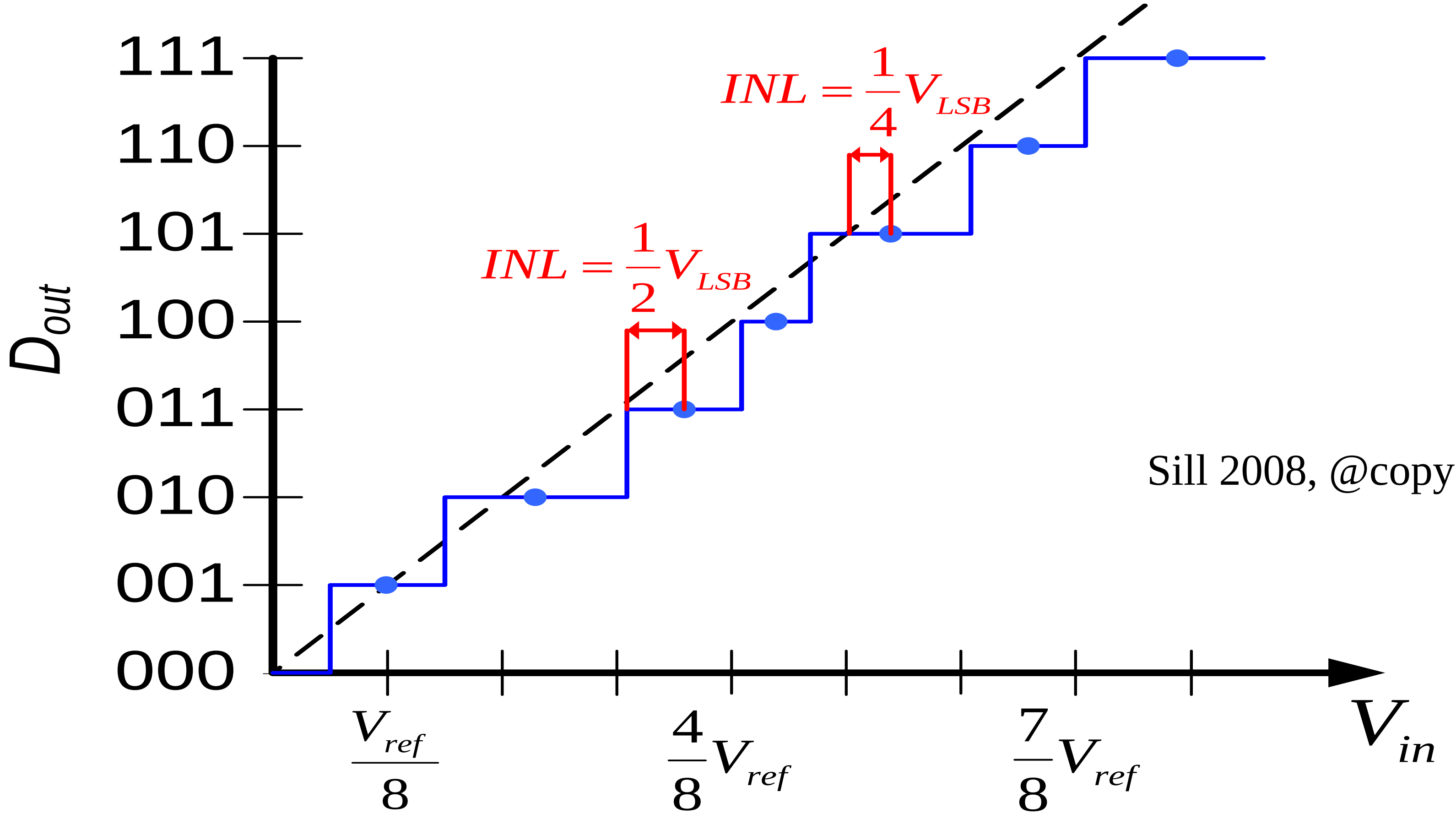
$$DNL = \frac{1}{2}V_{LSB} \quad \Rightarrow \quad \Delta V = 1.5V_{LSB}$$

Deviation of ΔV from V_{LSB} value (**in V_{LSB}**)

Defined after removing of gain

E.g. Caused by mismatch of the reference elements

Integral Non-Linearity (INL)



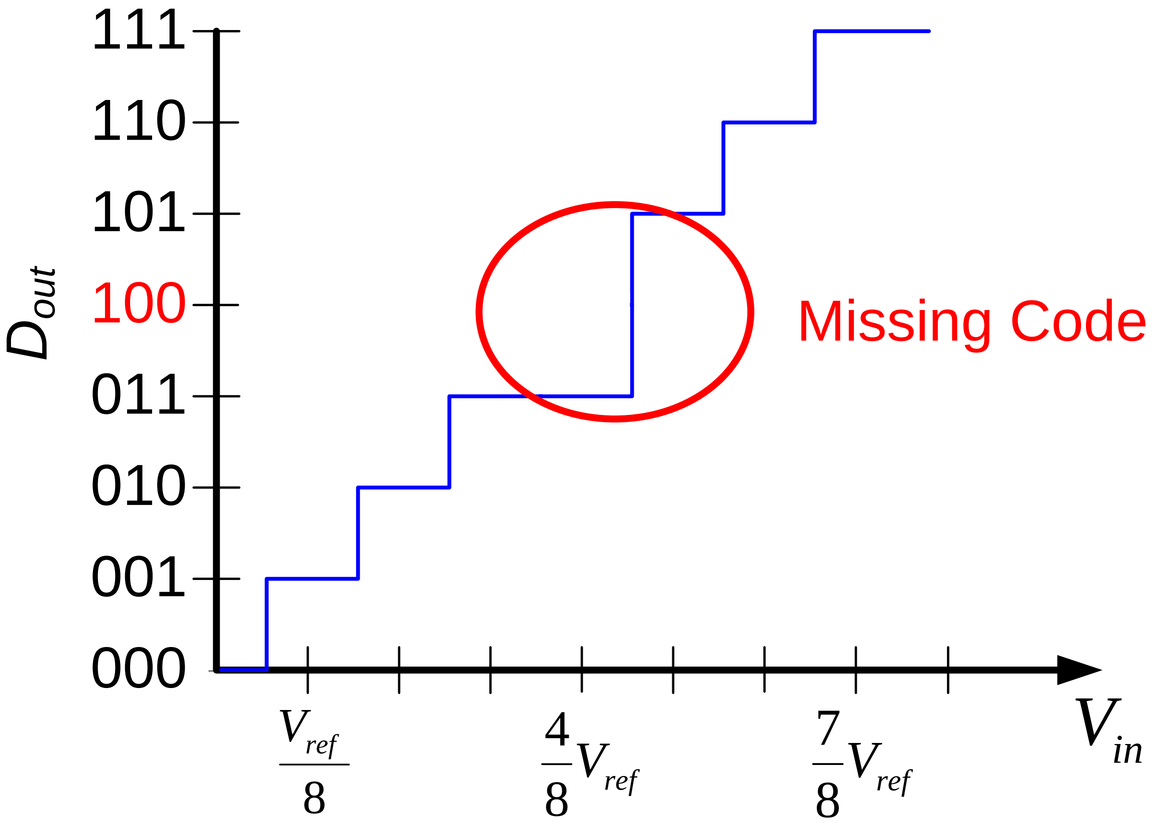
Sill 2008, @copyright

Deviation from the straight line (best-fit or end-point) (**in V_{LSB}**)

Defined after removing of gain and offset

E.g. caused by mismatch of the reference elements

Missing Codes

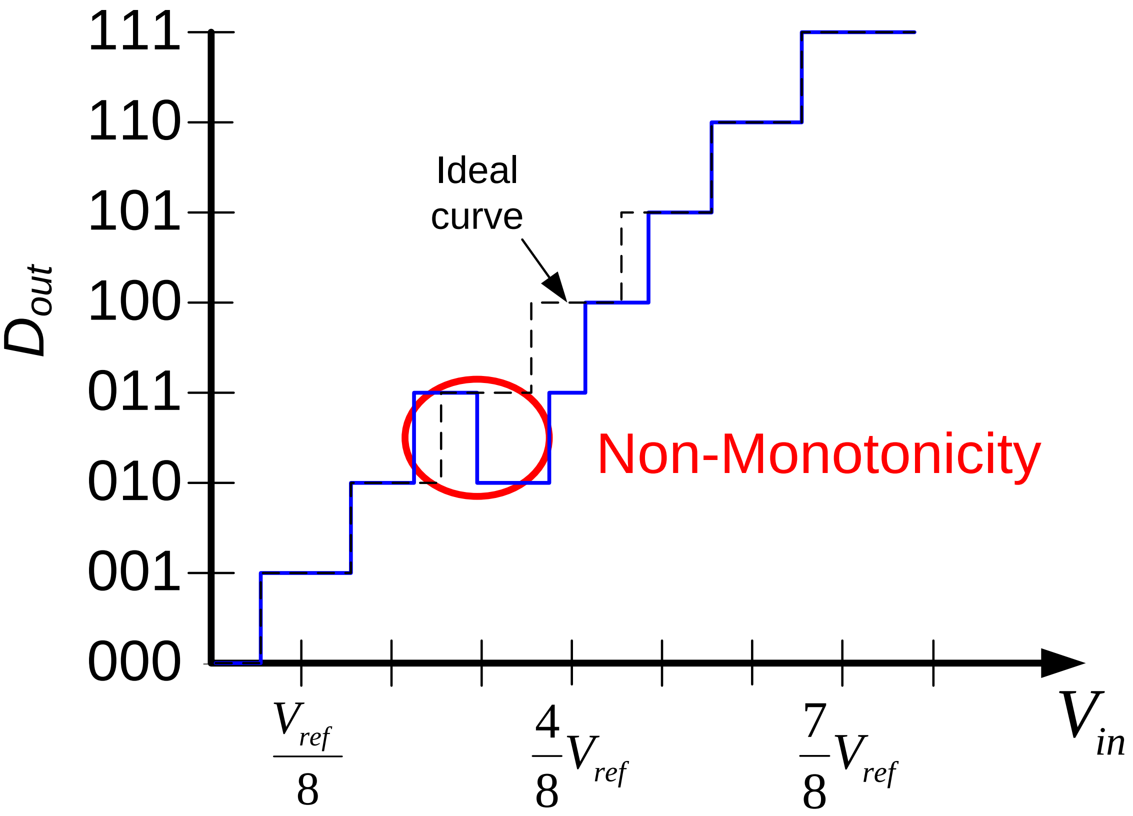


Sill 2008, @copyright

Some bit combinations never appear

Occurs, if maximum DNL > 1 V_{LSB} or maximum INL > 0.5 V_{LSB}

Non-Monotonicity



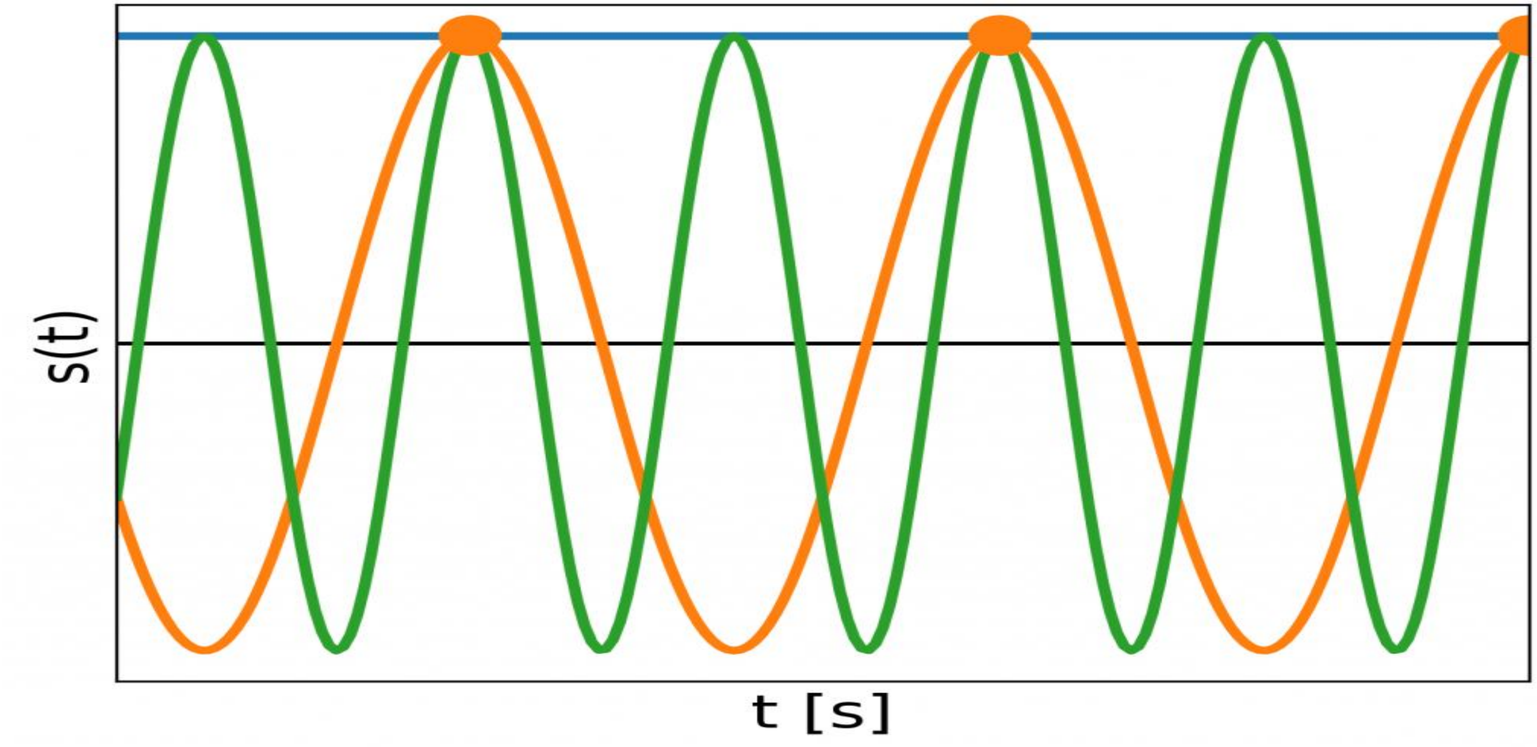
Sill 2008, @copyright

Lower conversion result for a higher input voltage

Includes that same conversion may result from two separate voltage ranges

Linnæus University

Aliasing



Too small sampling rate f_{samp} (under-sampling) can lead to aliasing (= frequency of reconstructed signal is to low)

Nyquist criterion:

- f_{samp} more than two times higher than highest frequency component f_{in} of input signal: $f_{samp} > 2 \cdot f_{in}$

Aliasing

☞ Reconstruction impossible, if not sampling frequently enough

How frequently do we have to sample?

Nyquist criterion (sampling theory):

Aliasing can be avoided if we restrict the frequencies of the incoming signal to less than half of the sampling rate.

$p_s < ½\, p_N$ where p_N is the period of the "fastest" sine wave

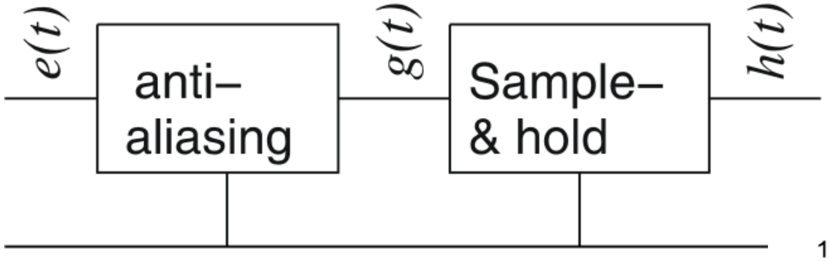
or $f_s > 2\, f_N$ where f_N is the frequency of the "fastest" sine wave

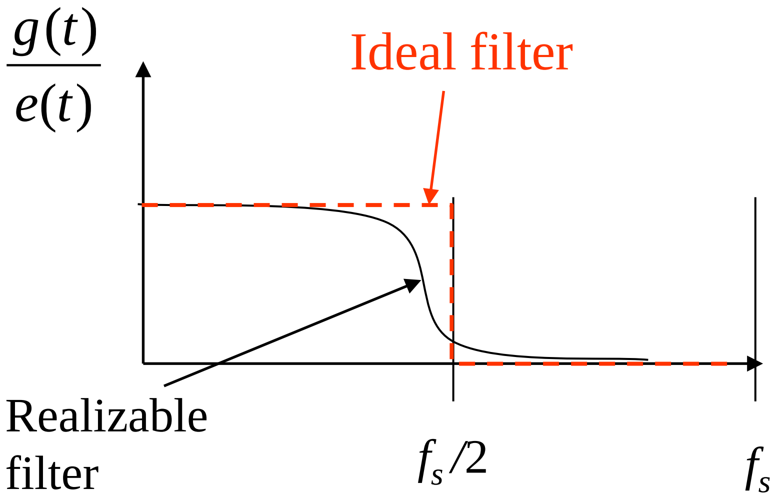
f_N is called the **Nyquist frequency**, f_s is the **sampling rate**.

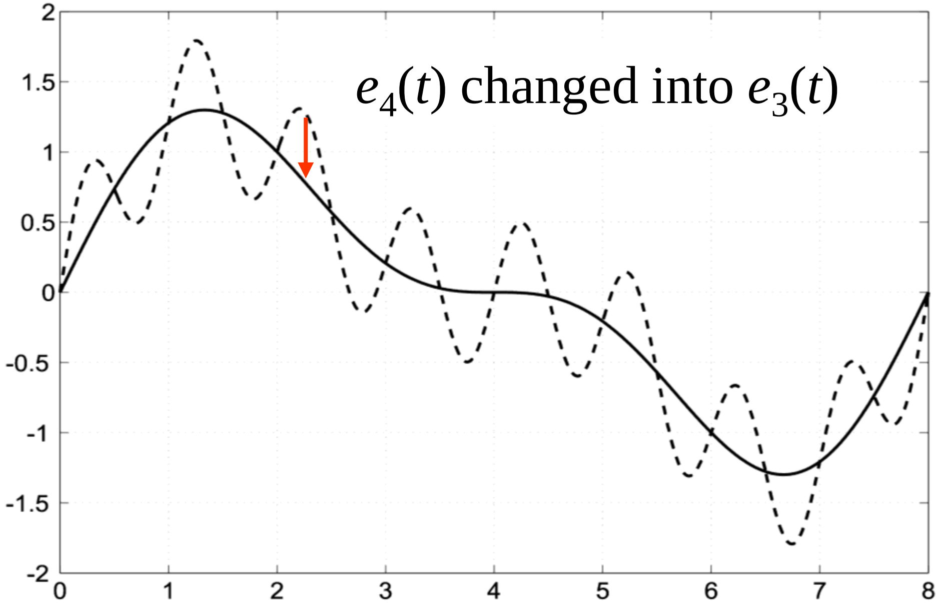
See e.g. [Oppenheim/Schafer, 2009]

Anti-aliasing filter

A filter is needed to remove high frequencies



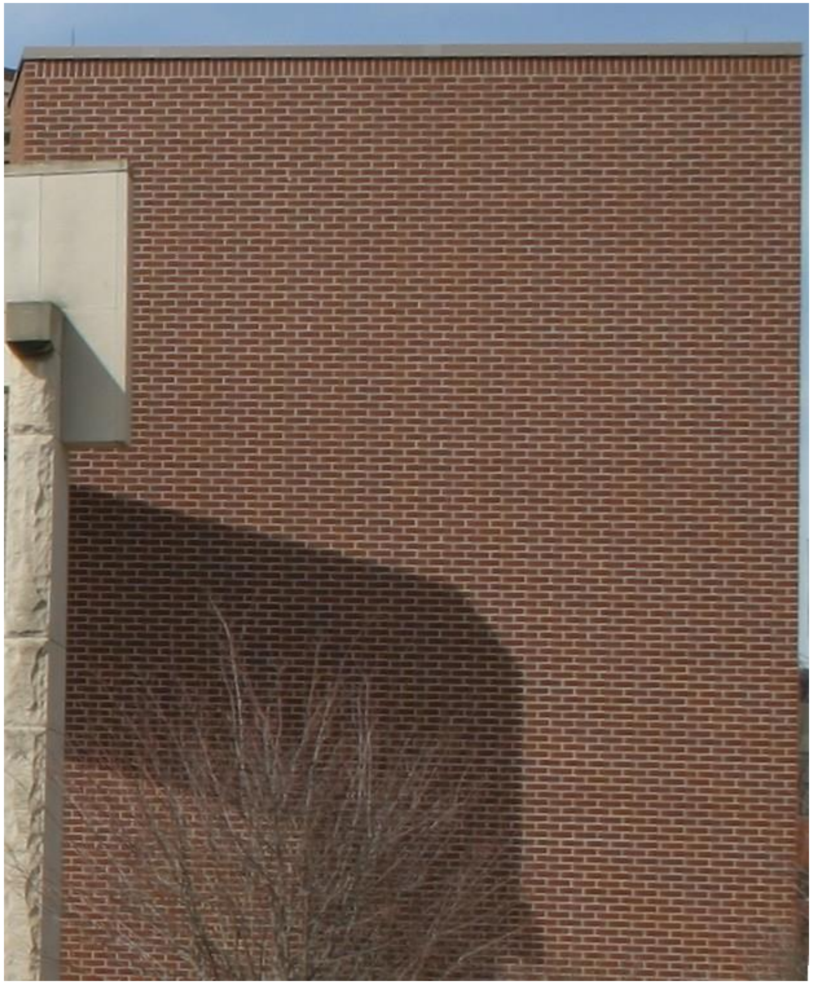




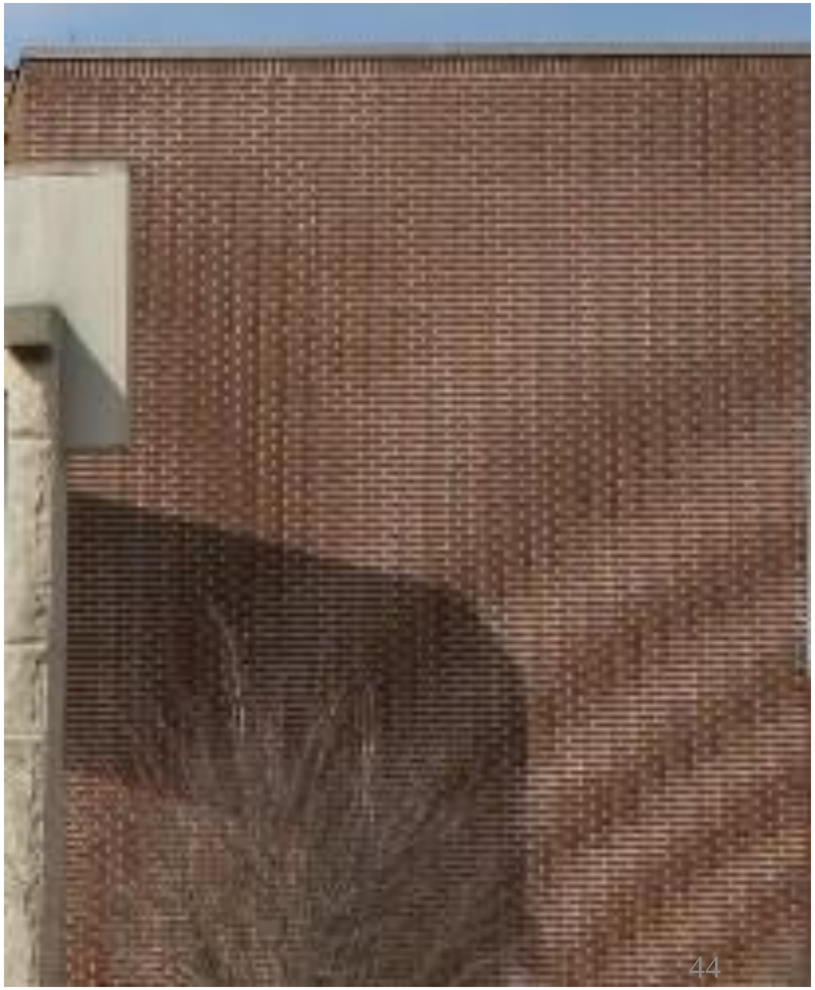
$e_4(t)$ changed into $e_3(t)$

Examples of aliasing in computer graphics

Original



Sub-sampled, no filtering



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.

ADC components:

•Analog input: This is the input signal that is to be converted into a digital signal.

•Comparator: This compares the analog input voltage with the voltage produced by the digital-to-analog converter (DAC).

•Register: This holds the digital output of the ADC. It is initially set to zero and is updated during the conversion process.’

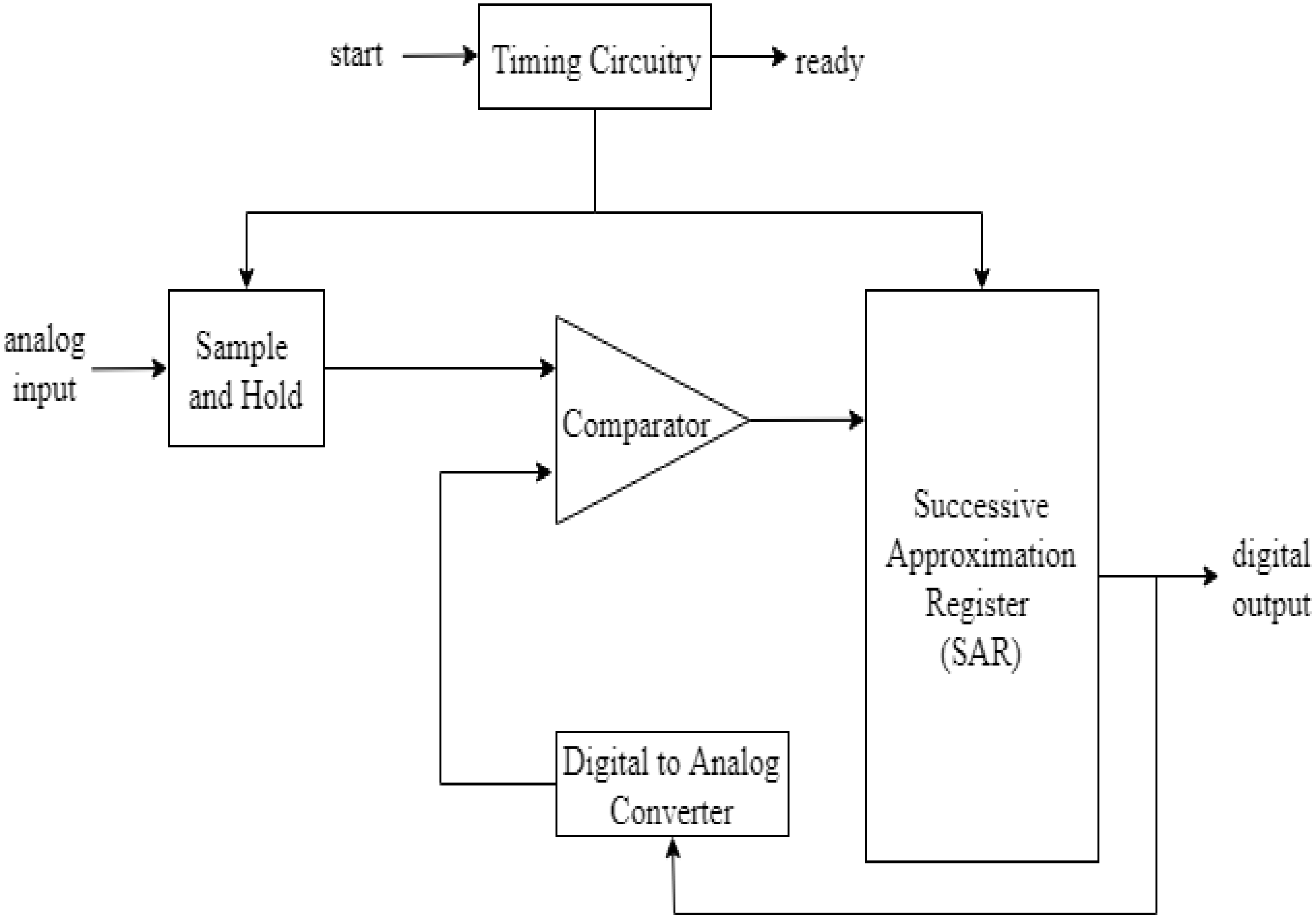
•DAC: This produces a voltage that is compared with the analog input voltage. The DAC output is controlled by the register contents.

•Clock: This provides the timing for the ADC operation.

SUCCESSIVE APPROXIMATION ADC

- Widely used
- Similar to Counter type ADC except that ,a SAR is used
- SAR acts as programmable Up/Down counter
- Completion of conversion , triggered by a change in the state of the comparator
- Much faster than the counter type

SUCCESSIVE APPROXIMATION ADC



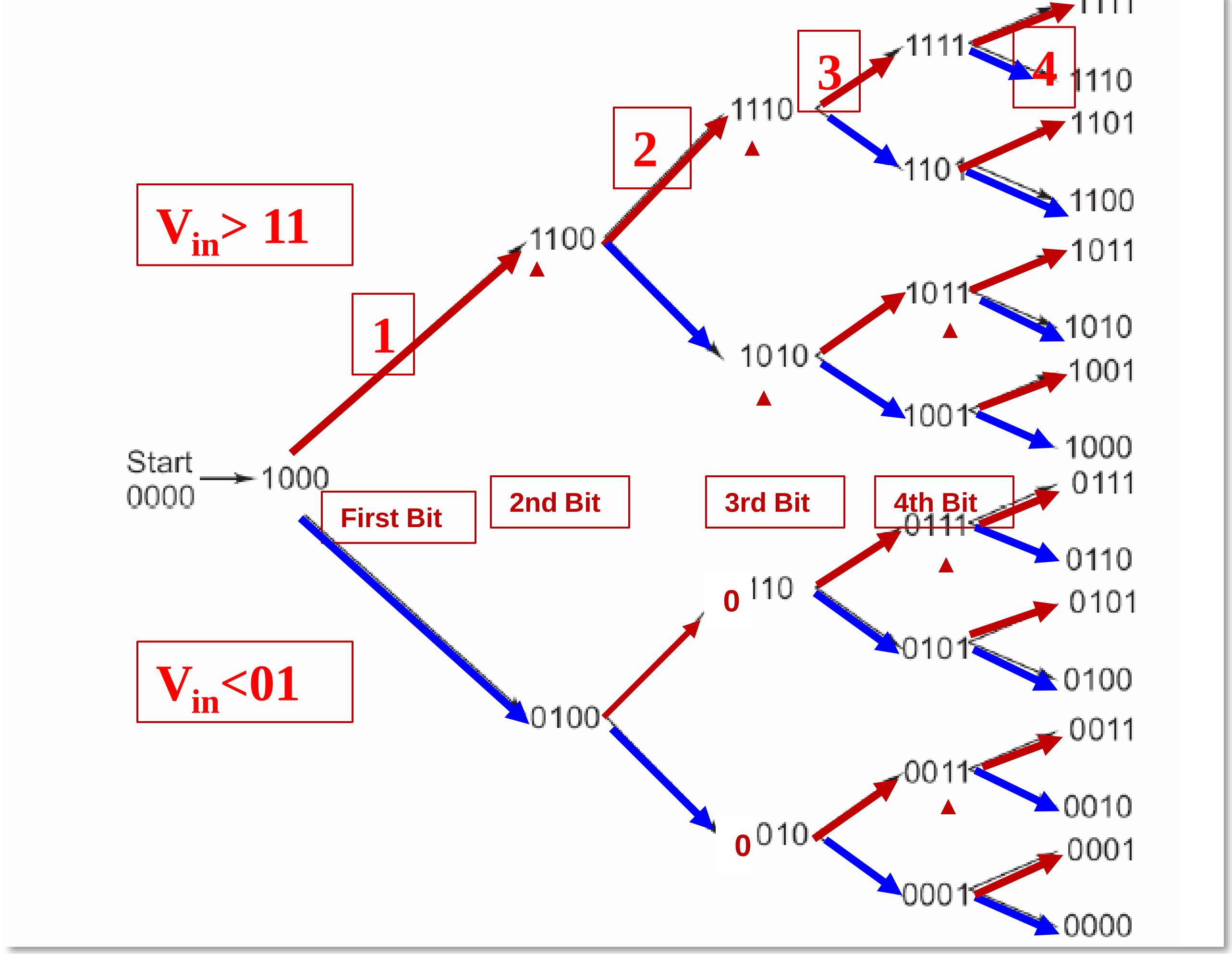
Implements Binary search algorithm

- **Initially, DAC input set to midscale (MSB =1)**
- **$V_{IN} > V_{DAC}$, MSB remains 1. Next bit is set to 1**
- **$V_{IN} < V_{DAC}$, MSB set to 0. Next bit is set to 1**
- **MSB's remain same after each conversion and next 3 bits are processed. Then next 2 bits, first 2 remaining same etc.**
- **Algorithm is repeated until LSB.**

DAC [input] = ADC [output]

N cycles required for N-bit Conversion.

Typical accuracy levels of 8 to 12 bits.



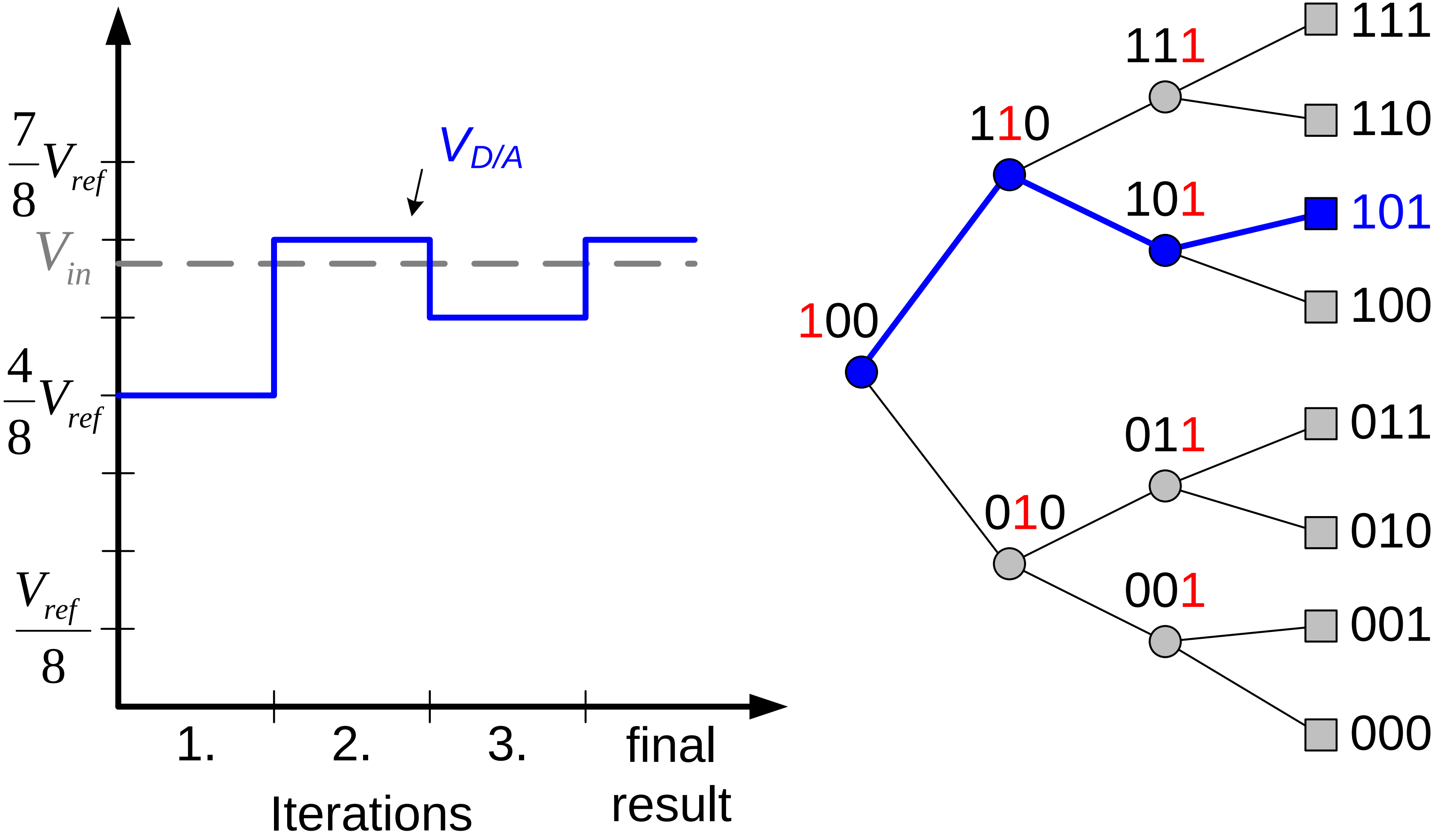
Upper arrow 1

Lower arrow 0

Next bit is always changed to ‘1’

1. Start conversion pulse will set all zeros in SAR.
2. MSB is set to 1 and others remaining 0’s (1/2 the input voltage). DAC output is compared with unknown voltage. *(1000)*
3. If unknown voltage is higher, the bit under comparison is **retained 1** and the next bit is made 1.
4. If unknown voltage is less, the bit under comparison is made 0 and the next bit is made 1.
5. MSB’s remain same after each conversion and next 3 bits are processed. Then next 2 bits, first 2 remaining same etc.
6. Then comparison moves to next bit and process continues till last bit.

Successive Approximation ADC



Analog Digital Converter

Errors and ADCs

Figures and some text from:

- Understanding analog to digital converter specifications. By Len Staller
- http://www.embedded.com/showArticle.jhtml?articleID=60403334

Key concept here is that the specification provides *worst case* values.

Errors

Once again: Errors in a specification are worst case.

- So if you have an (Integral nonlinearity) INL of ±.25 LSB, you "know" that the device will never have more than .25 LSB error from its ideal value.
- That of course assumes you are operating within the specification
  - Temperature, input voltage, input current available, etc.

INL and DNL are the ones we expect you to work with

- Should know what full-scale error is

Signal to noise ratio

$$\text{signal to noise ratio (SNR) [db]} = 20\log_{10}\left(\frac{\text{effective signal voltage}}{\text{effective noise voltage}}\right)$$

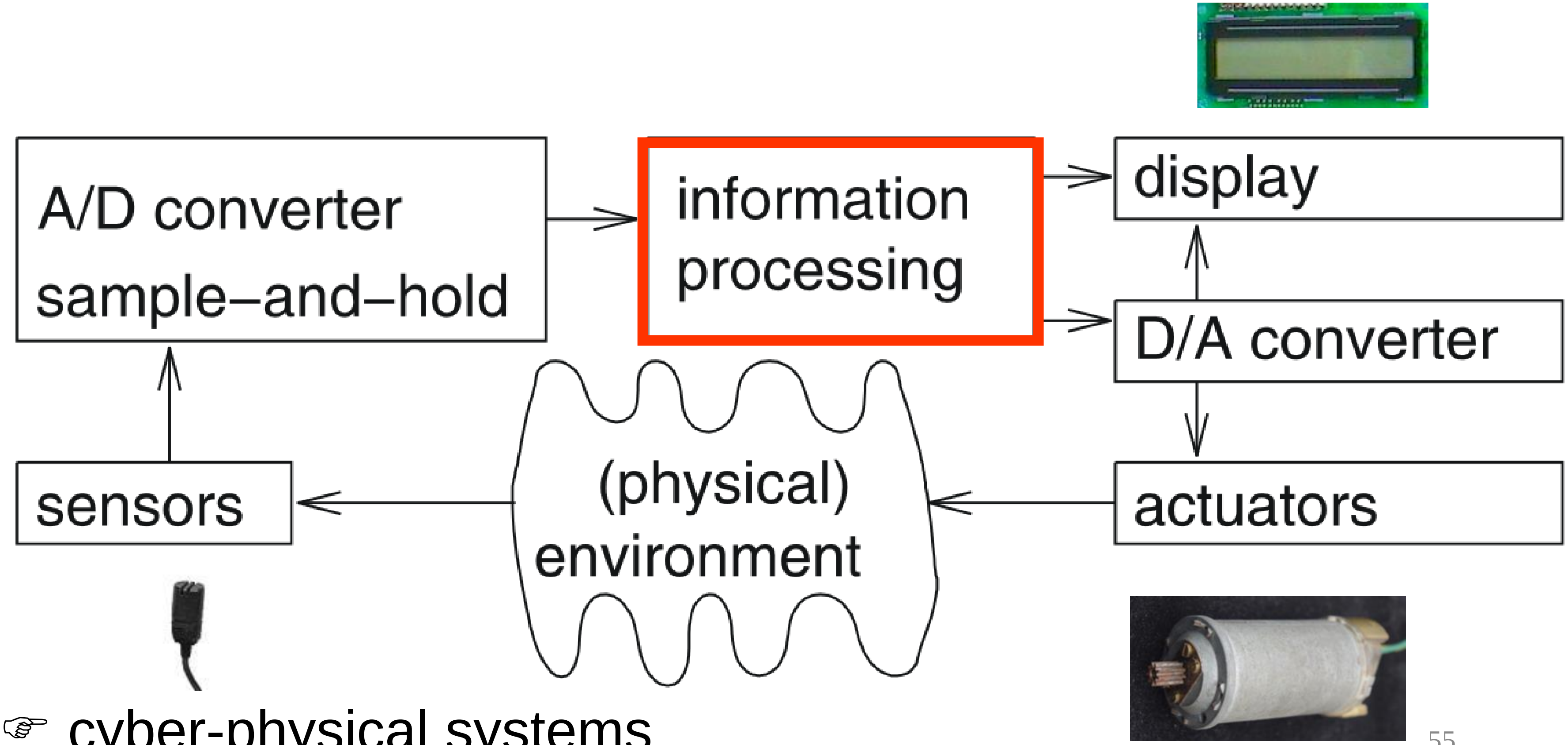
e.g.: 20 $\log_{10}(2)$=6.02 decibels

Signal to noise for ideal *n*-bit converter : *n* * 6.02 + 1.76 [dB]
e.g:
98.1 db for 16-bit converter,
160 db for 24-bit converter

Additional noise for non-ideal converters

Embedded System Hardware

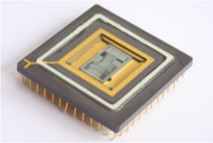
Embedded system hardware is frequently used in a loop (***“hardware in a loop“***):



☞ cyber-physical systems

Efficiency

– CPS & ES must be **efficient**



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

➡ ❖ Run-time efficient

❖ Weight efficient

❖ Cost efficient

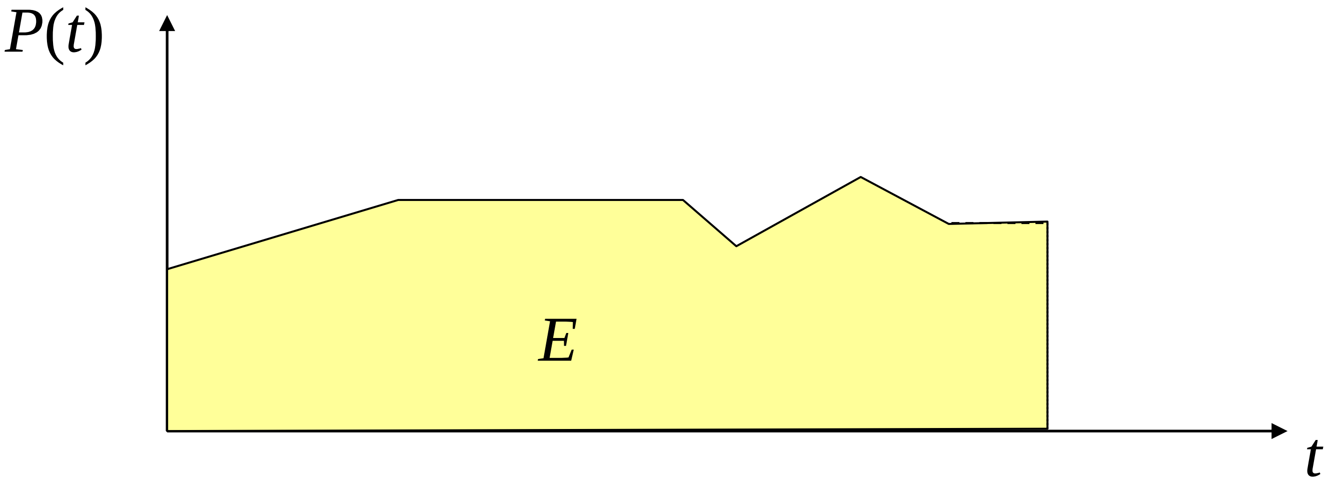
➡ ❖ Energy efficient

Why care about energy efficiency ?

	Relevant during use?		
Execution platform	Plugged	Uncharged periods	Unplugged
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

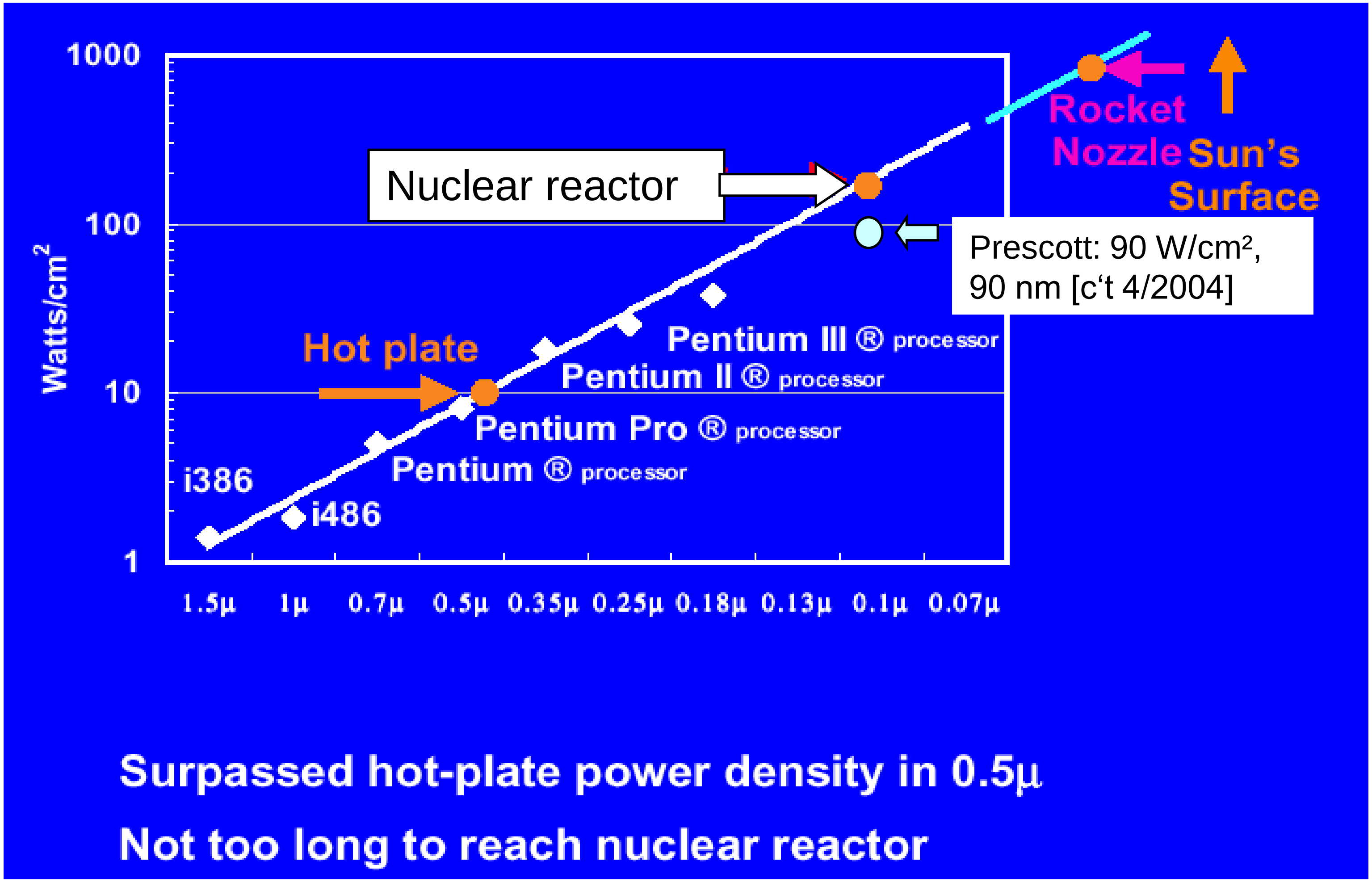
- design of the power supply & regulators
- dimensioning of interconnect, short term cooling

❑ 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**

PCs: Problem: Power density increasing



Nuclear reactor

Prescott: 90 W/cm², 90 nm [c't 4/2004]

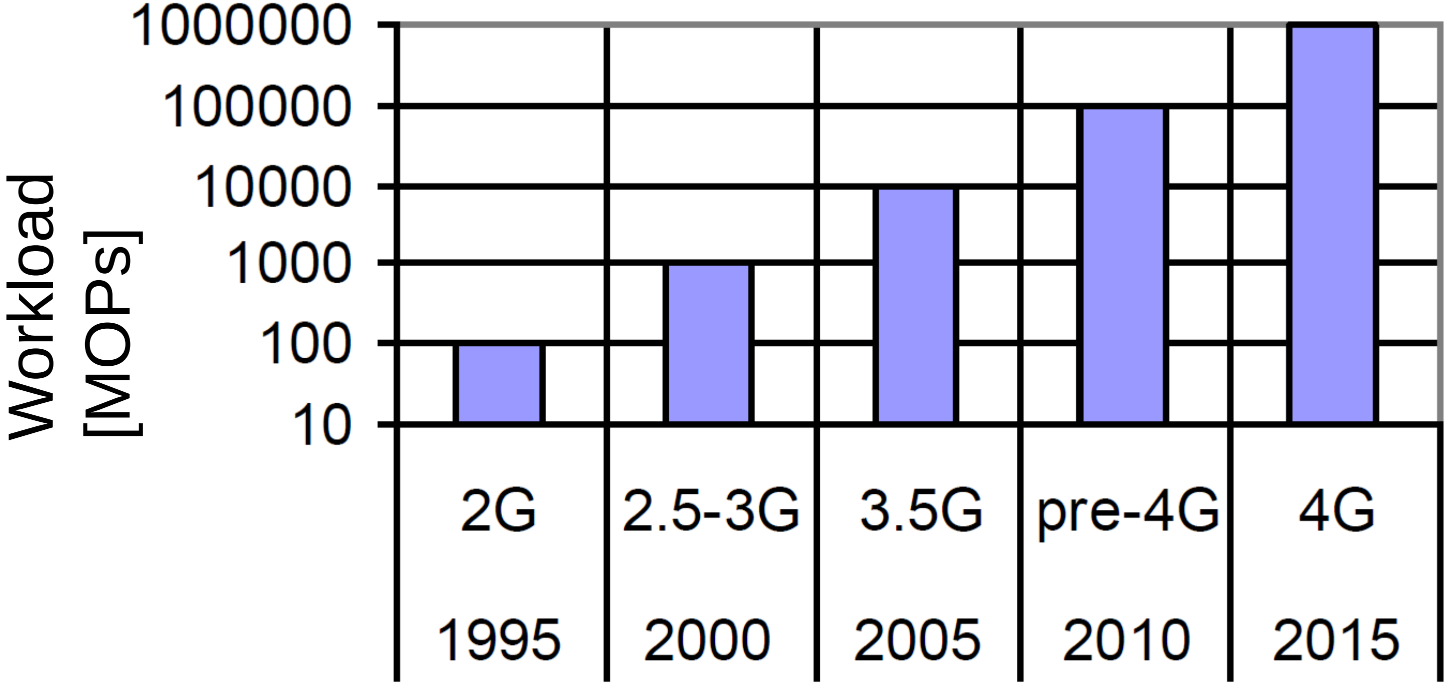
Surpassed hot-plate power density in 0.5µ

Not too long to reach nuclear reactor

© Intel
M. Pollack,
Micro-32

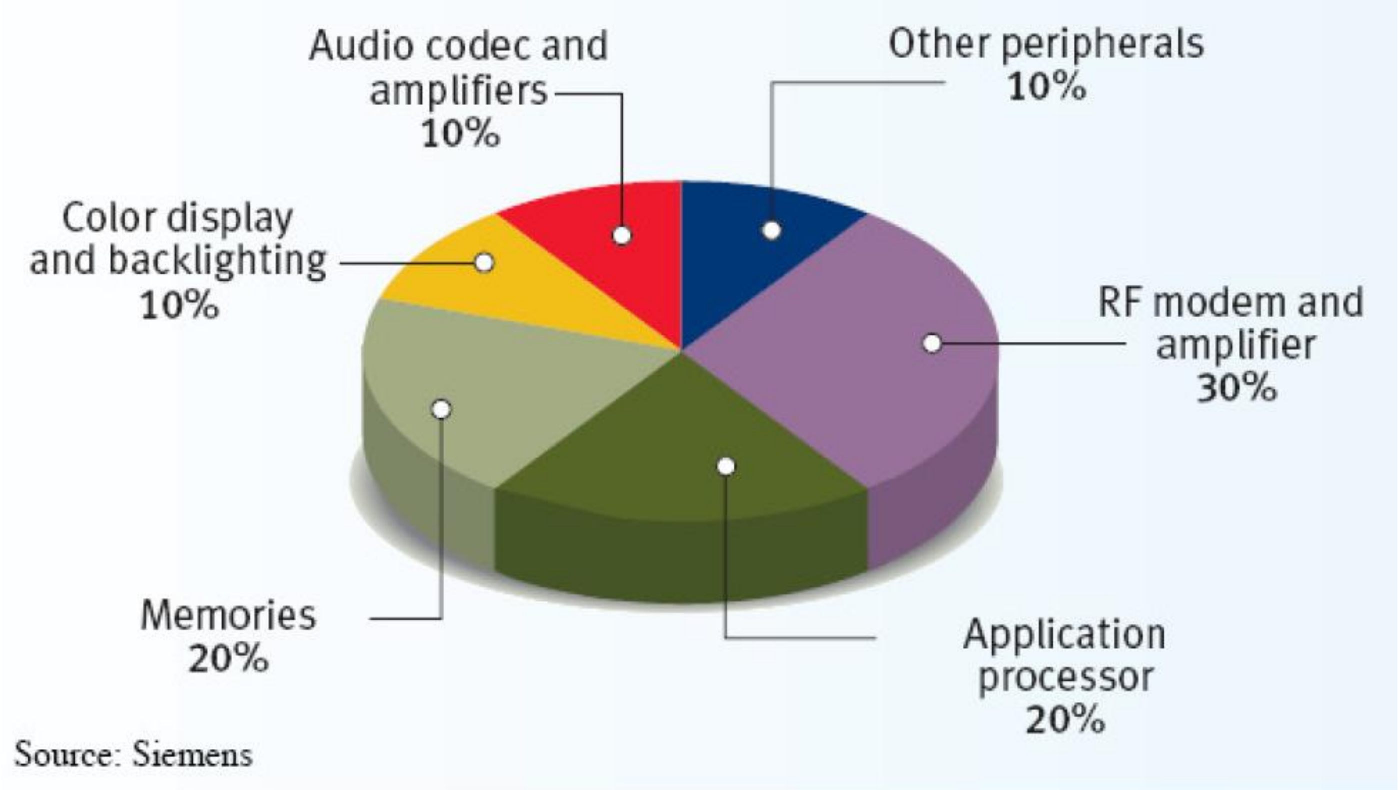
Mobile phones: Increasing performance requirements

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



Many more instances of the power/energy problem (Memory operations per second MOPS)

Mobile phones: Where does the power go?



Source: Siemens

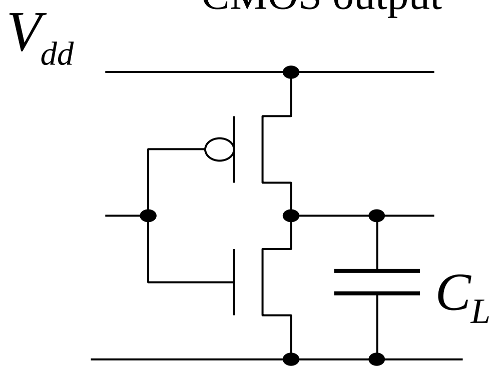
- **It not just I/O, don’t ignore processing!**

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

Static and dynamic power consumption

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

CMOS output



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

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

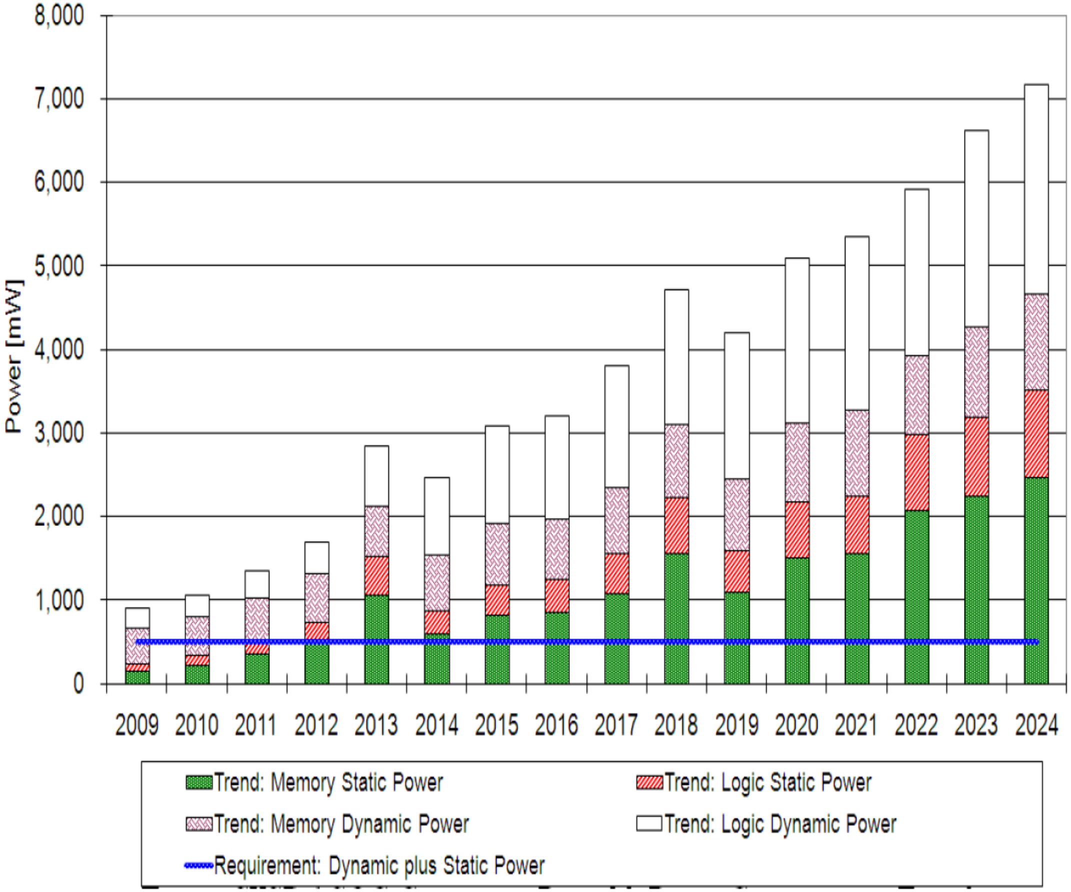
f : clock frequency

☞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

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



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

Power consumption of CMOS circuits (ignoring leakage):

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

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

f : clock frequency

Delay for CMOS circuits:

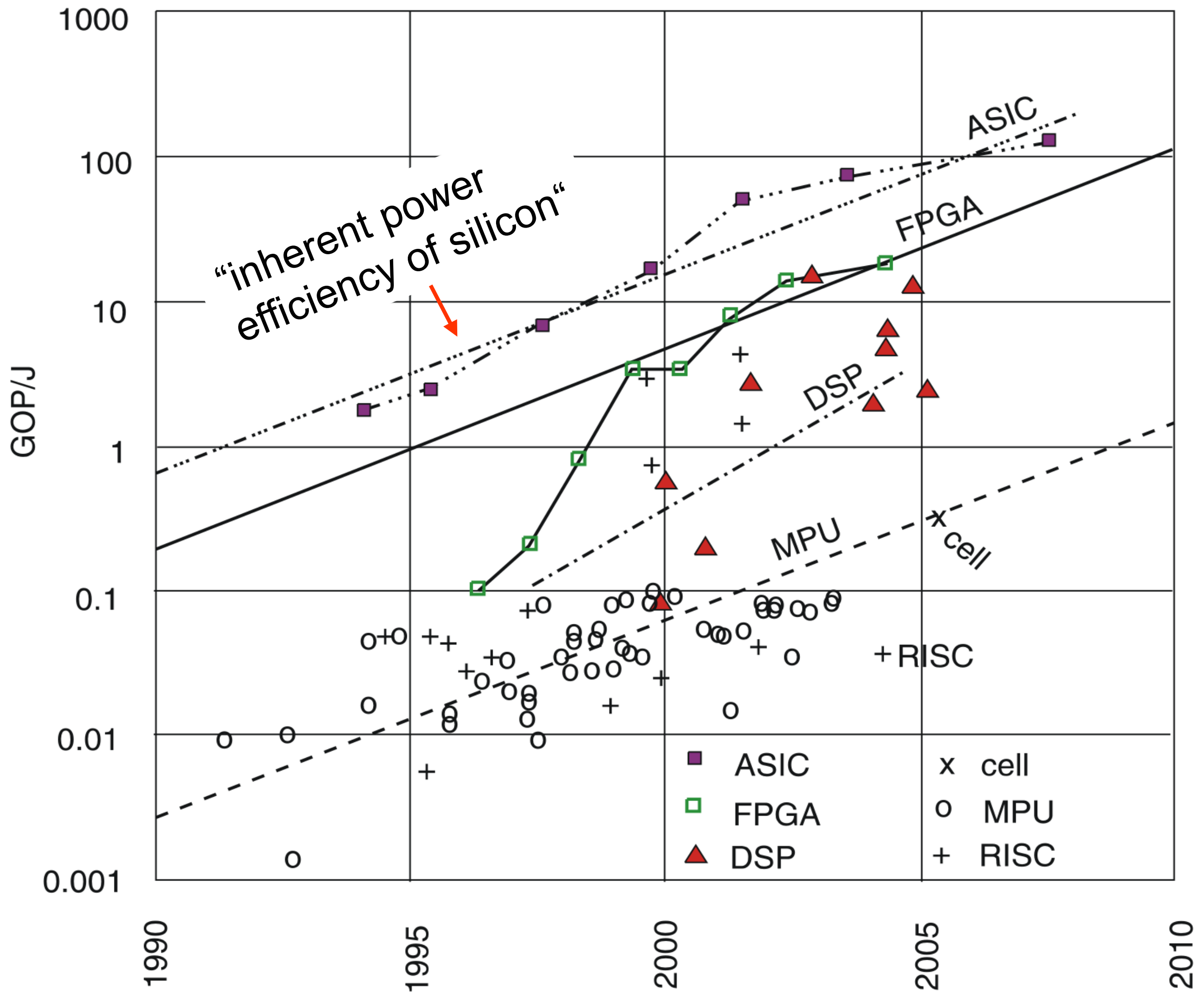
$\tau = k \, C_L \frac{V_{dd}}{(V_{dd} - V_t)^2}$ with

V_t : threshhold voltage

($V_t <$ than V_{dd})

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

Energy Efficiency of different target platforms

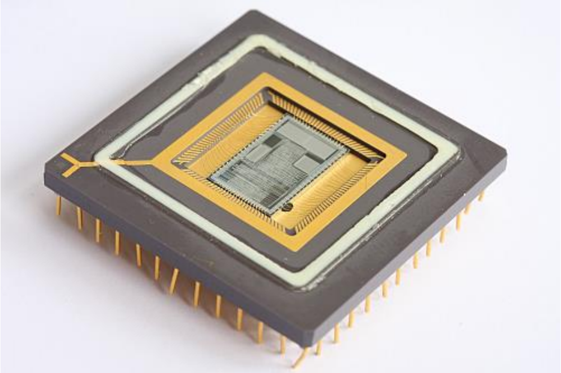


"inherent power efficiency of silicon"

© Hugo De Man, IMEC, Philips, 2007

Application Specific Circuits (ASICS) or Full Custom Circuits

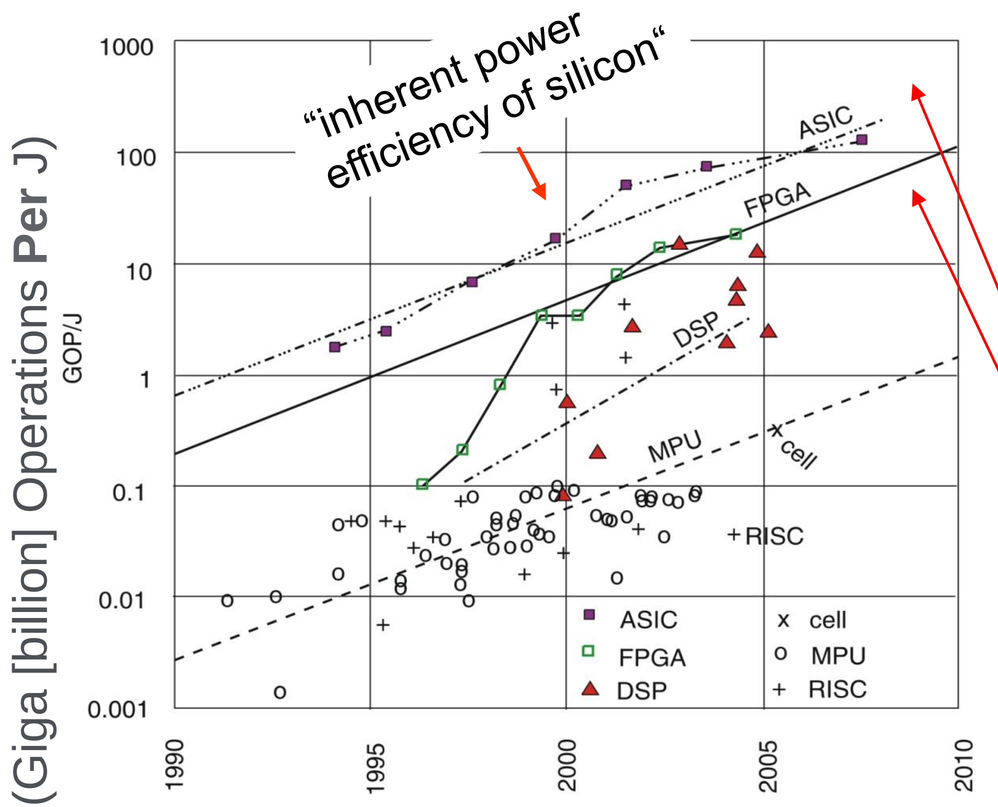
- ❑ Approach suffers from
  - ▪ long design times,
  - ▪ lack of flexibility (changing standards) and
  - ▪ high costs (e.g. Mill. $ mask costs)



- ❑ 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

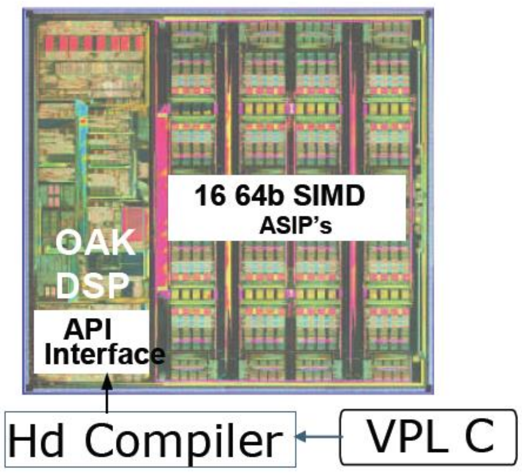
More energy-efficient architectures: Domain- and application specific



"inherent 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

VIP for car mirrors Infineon

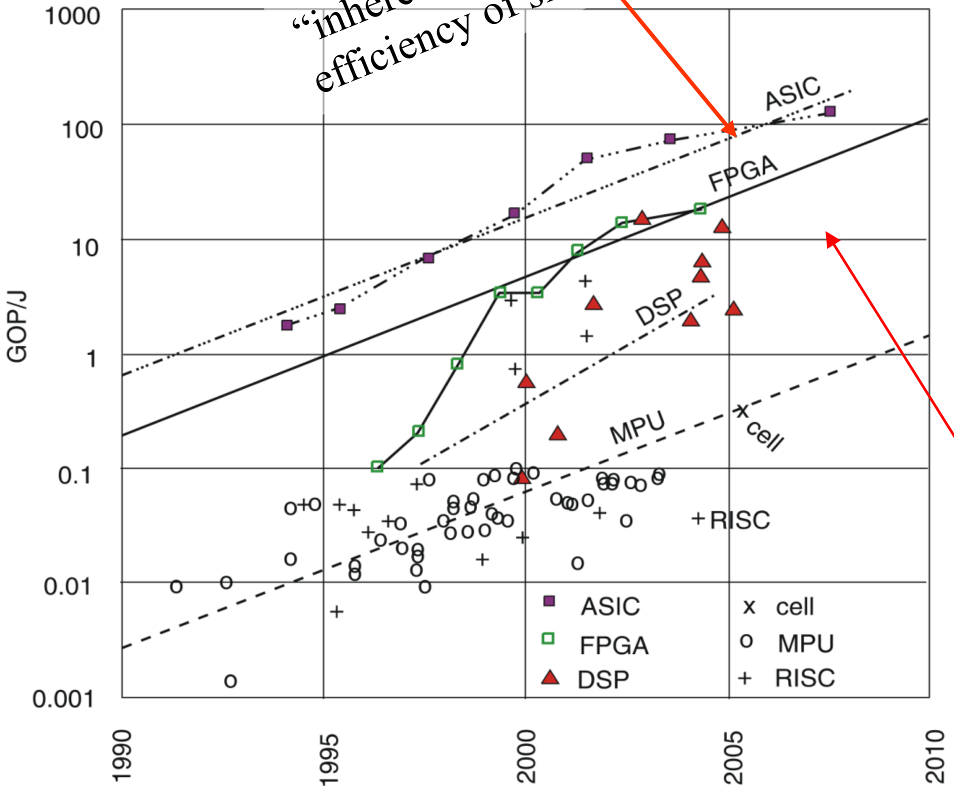


200MHz , 0.76 Watt
100Gops @ 8b
25Gops @ 32b

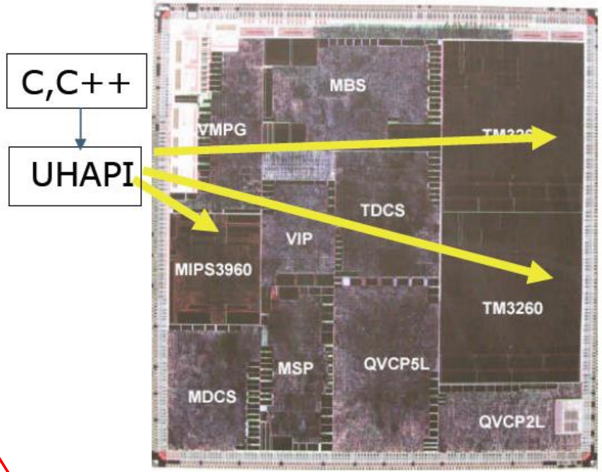
Close to power efficiency of silicon

Energy-efficient architectures:
Domain- and application specific

“‘inherent power efficiency of silicon“



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

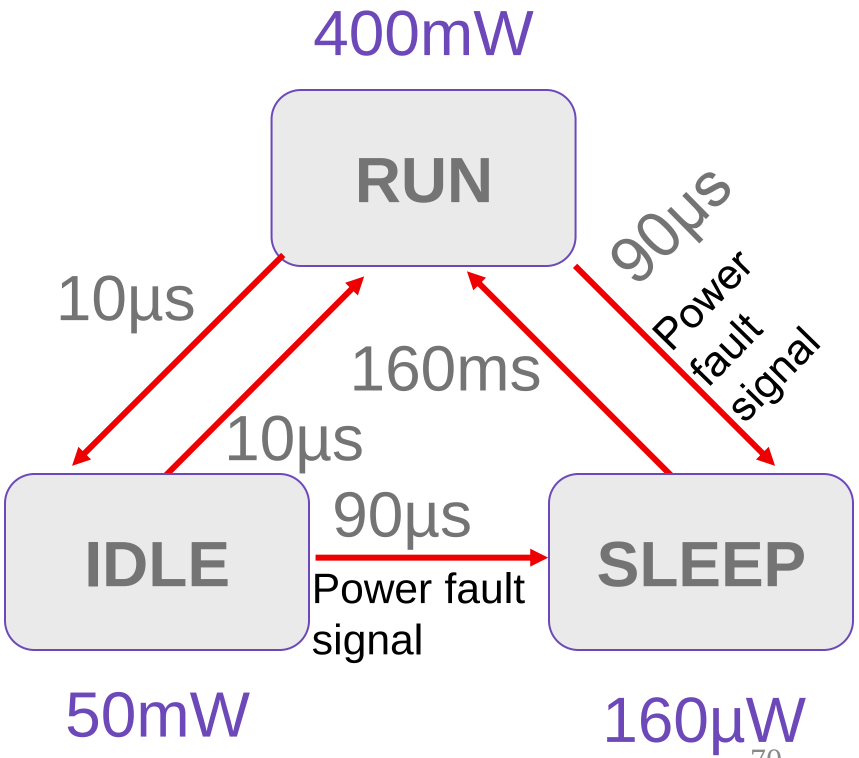
Dynamic power management (DPM)

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



Summary

Hardware in a loop

- ❑ Sensors
- ❑ Discretization
    - • Sample-and-hold circuits
        - - Aliasing (and how to avoid it)
        - - Nyquist criterion
    - • A/D-converters
        - - Quantization noise
- ❑ Information Processing
    - • Energy Efficiency

References

[Base slide]Pao-Ann Hsiung, 熊博安 Graduate Institute of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621, Taiwan, R.O.C.

[Boehm73] Boehm, B.W. “Software and its Impact: A Quantitative Assessment,” *Datamation,* May 1973, p. 48-59.

[Buchenrieder93] Buchenrieder, K., “Codesign and Concurrent Engineering”, Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 85-86

[Buck94] Buck, J., et al., “Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems,” *International Journal of Computer Simulation,* Vol. 4, April 1994, pp. 155-182.

[Chiodo92] Chiod0, M., A. Sangiovanni-Vincentelli, “Design Methods for Reactive Real-time Systems Codesign,” *International Workshop on Hardware/Software Codesign,* Estes Park, Colorado, September 1992.

[Chiodo94] Chiodo, M., P. Giusto, A. Jurecska, M. Marelli, H. C. Hsieh, A. Sangiovanni-Vincentelli, L. Lavagno, “Hardware-Software Codesign of Embedded Systems,” *IEEE Micro,* August, 1994, pp. 26-36; © IEEE 1994.

[Chou95] P. Chou, R. Ortega, G. Borriello, “The Chinook hardware/software Co-design System,” *Proceedings ISSS,* Cannes, France, 1995, pp. 22-27.

[DeMicheli93] De Micheli, G., “*Extending CAD Tools and Techniques*”, Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 84

[DeMicheli94] De Micheli, G., “Computer-Aided Hardware-Software Codesign”, *IEEE Micro,* August, 1994, pp. 10-16

[DeMichelli97] De Micheli, G., R. K. Gupta, “Hardware/Software Co-Design,” *Proceedings of the IEEE,* Vol. 85, No. 3, March 1997, pp. 349-365.

[Ernst93] Ernst, R., J. Henkel, T. Benner, “*Hardware-Software Cosynthesis for Micro-controllers*”, *IEEE Design and Test*, December, 1993, pp. 64-75

[Franke91] Franke, D.W., M.K. Purvis. “Hardware/Software Codesign: A Perspective,” *Proceedings of the 13th International Conference on Software Engineering,* May 13-16, 1991, p. 344-352; © IEEE 1991

References (Cont.)

Copyright ã 1995-1999 SCRA, Hardware/Software Codesign Overview RASSP Education & Facilitation Program Module 14

Proceedings of the International Workshops / Symposium on HW/SW Codesign, 1993 ~ 2001 (ACM Press & IEEE CS Press).

Hardware Software Co-design of Embedded Systems, F. Balarin, Chiodo, et al., Kluwer Academic Publishers, May 1997.

Co-synthesis of Hardware and Software for Embedded Systems, R. Gupta, Kluwer Academic Publishers, 1995.

Special Issue of the Proceedings of the IEEE on Hardware Software Co-design edited by G. De Micheli, Vol. 85, No. 3, March 1997.

POLIS and Ptolemy tools introduction materials and manuals

[Gajski94] Gajski, D. D., F. Vahid, S. Narayan, J. Gong, *Specification and Design of Embedded Systems*, Prentice Hall, Englewood Cliffs, N J, 07632, 1994

[Gupta92] Gupta, R.K., C.N. Coelho, Jr., G.D. Micheli. “Synthesis and Simulation of Digital Systems Containing Interactive Hardware and Software Components,” *29th Design Automation Conference*, June 1992, p.225-230.

[Gupta93] Gupta, R.K., G. DeMicheli, “Hardware-Software Cosynthesis for Digital Systems,” *IEEE Design and Test*, September 1993, p.29-40; © IEEE 1993.

[Hermann94] Hermann, D., J. Henkel, R. Ernst, “An approach to the estimation of adapted Cost Parameters in the COSYMA System*”, 3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 100-107

[Hood94] Hood, W., C. Myers, "RASSP: Viewpoint from a Prime Developer," *Proceedings 1st Annual RASSP Conference*, Aug. 1994.

[IEEE] All referenced IEEE material is used with permission.

[Ismail95] T. Ismail, A. Jerraya, “Synthesis Steps and Design Models for Codesign,” *IEEE Computer*, no. 2, pp. 44-52, Feb 1995.

[Kalavade93] A. Kalavade, E. Lee, “A Hardware-Software Co-design Methodology for DSP Applications,” *IEEE Design and Test*, vol. 10, no. 3, pp. 16-28, Sept. 1993.

References (Cont.)

[Parker84] Parker, A.C., “Automated Synthesis of Digital Systems,” *IEEE Design and Test,*, November 1984, p. 75-81.

[RASSP94] *Proceedings of the 1st RASSP Conference*, Aug. 15-18, 1994.

[Rozenblit94] Rozenblit, J. and K. Buchenrieder (editors). Codesign Computer -Aided Software/Hardware Engineering, IEEE Press, Piscataway, NJ, 1994; © IEEE 1994.

[Smith86] Smith, C.U., R.R. Gross. “Technology Transfer between VLSI Design and Software Engineering: CAD Tools and Design Methodologies,” *Proceedings of the IEEE*, Vol. 74, No. 6, June 1986, p.875-885.

[Srivastava91] M. B. Srivastava, R. W. Broderson, “Rapid prototyping of Hardware and Software in a Unified Framework,” *Proceedings ICCAD*, 1991, pp. 152-155.

[Subrahmanyam93] Subrahmanyam, P. A., “Hardware-Software Codesign -- Cautious optimism for the future”, Hot Topics, *IEEE Computer*, R. D. Williams, ed., January, 1993, pp. 84

[Tanenbaum87] Tanenbaum, A.S., *Operating Systems: Design and Implementation*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1987.

[Terry90] Terry, C. “Concurrent Hardware and Software Design Benefits Embedded Systems,” *EDN*, July 1990, p. 148-154.

[Thimbleby88] Thimbleby, H. “Delaying Commitment,” *IEEE Software*, Vol. 5, No. 3, May 1988, p. 78-86.

[Thomas93] Thomas, D.E., J.K. Adams, H. Schmitt, “A Model and Methodology for Hardware-Software Codesign,” *IEEE Design and Test*, September 1993, p.6-15; © IEEE 1993.

[Turn78] Turn, R., “Hardware-Software Tradeoffs in Reliable Software Development,” *11th Annual Asilomar Conference on Circuits, Systems, and Computers*, 1978, p.282-288.

[Vahid94] Vahid, F., J. Gong, D. D. Gajski, “*A Binary Constraint Search Algorithm for Minimizing Hardware During Hardware/Software Partitioning”,* 3rd International Conference on Hardware/Software Codesign, Grenoble, France, Sepetember22-24, 1994, pp. 214-219

[Wolf94] Wolf, W.H. “Hardware-Software Codesign of Embedded Systems,” *Proceedings of the IEEE*, Vol. 82, No.7, July 1994, p.965-989.

References (Cont.)

Additional Reading:

Aylor, J.H. et al., "The Integration of Performance and Functional Modeling in VHDL" in *Performance and Fault Modeling with VHDL*, J. Schoen, ed., Prentice-Hall, Englewood Cliffs, N.J., 1992.

D'Ambrosio, J. G., X. Hu, "Configuration-level Hardware-Software Partitioning for Real-time Embedded Systems", *3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 34-41

Eles, P., Z. Peng, A. Doboli, "VHDL System-Level Specification and Partitioning in a Hardware-Software Cosynthesis Environment", *3rd International Conference on Hardware/Software codesign*, Grenoble, France, September 22-24, 1994, pp. 49-55

Gupta, R.K., G. DeMicheli, "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design and Test*, September 1993, p.29-40.

Richards, M., Gadient, A., Frank, G., eds. *Rapid Prototyping of Application Specific Signal Processors*, Kluwer Academic Publishers, Norwell, MA, 1997

Schultz, S.E., "An Overview of System Design," *ASIC and EDA*, January 1993, p.12-21.

Thomas, D. E, J. K. Adams, H. Schmit, "A Model and Methodology for Hardware-Software Codesign", *IEEE Design and Test*, September, 1993, pp. 6-15

Zurcher, F.W., B. Randell, "Iterative Multi-level Modeling - A Methodology for Computer System Design," *Proceedings IFIP Congress '68,* Edinburgh, Scotland, August 1968, p.867-871.

Lnu.se