technische universität
dortmund

fakultät für informatik
informatik 12

Embedded System Hardware

Peter Marwedel
Informatik 12
TU Dortmund
Germany

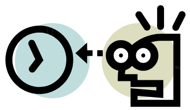
2012年 11 月 13 日

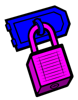
Motivation

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

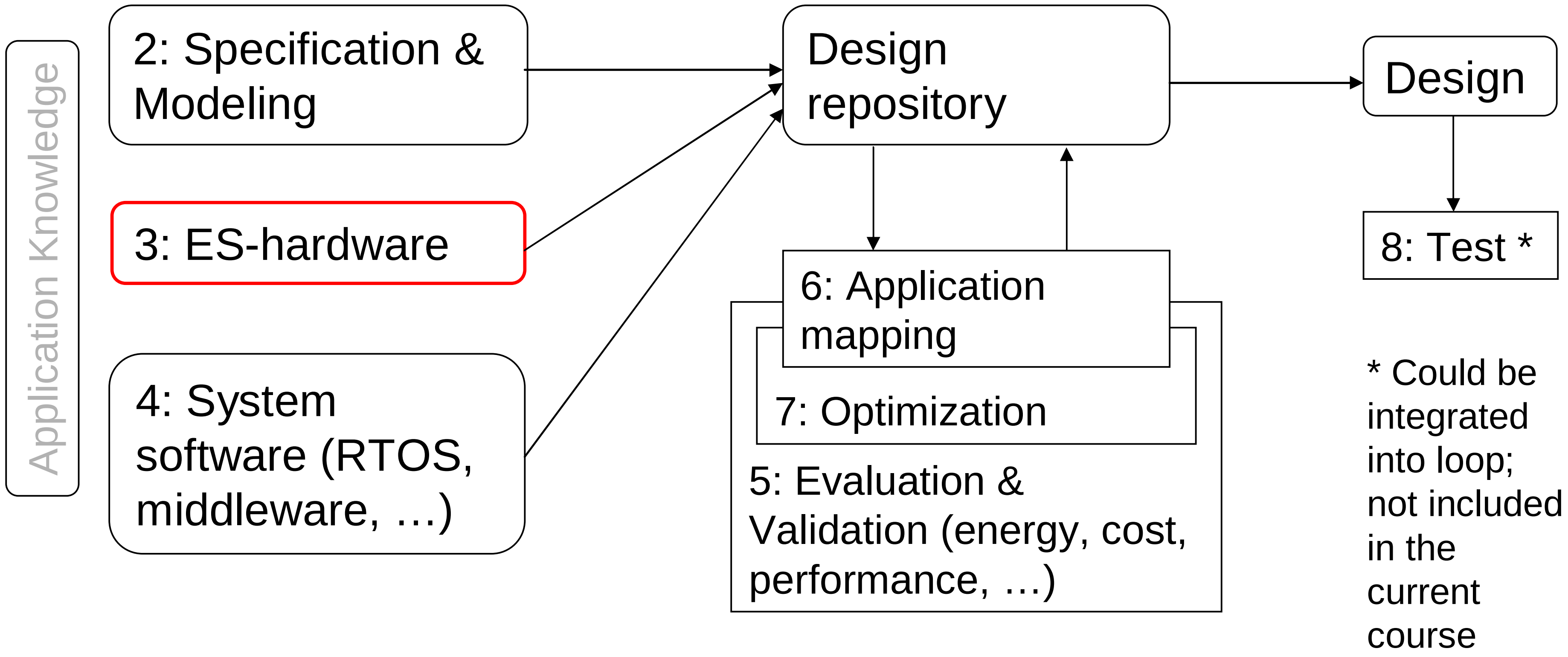
$$\int P \, dt$$

Reasons for considering hard- and software:

- Real-time behavior 
- Efficiency 

  - Energy
  - …
- Security 
- Reliability 
- …

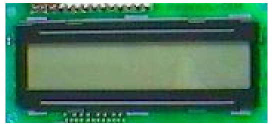
Structure of this course

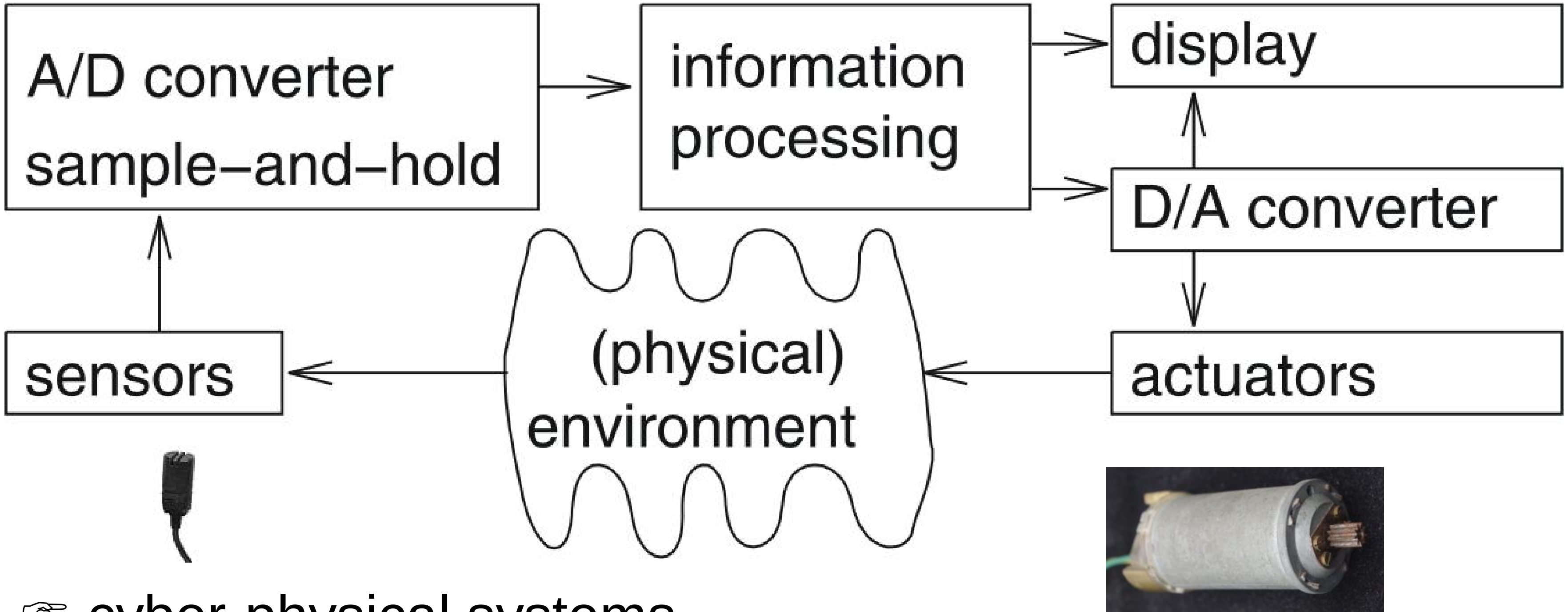


Generic loop: tool chains differ in the number and type of iterations
Numbers denote sequence of chapters

Embedded System Hardware

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

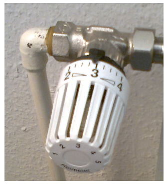




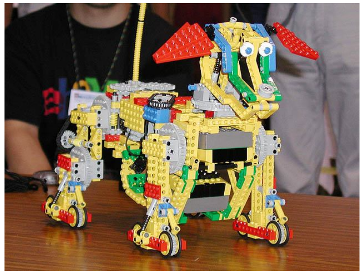
☞ cyber-physical systems

Many examples of such loops

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



© P. Marwedel, 2011



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.

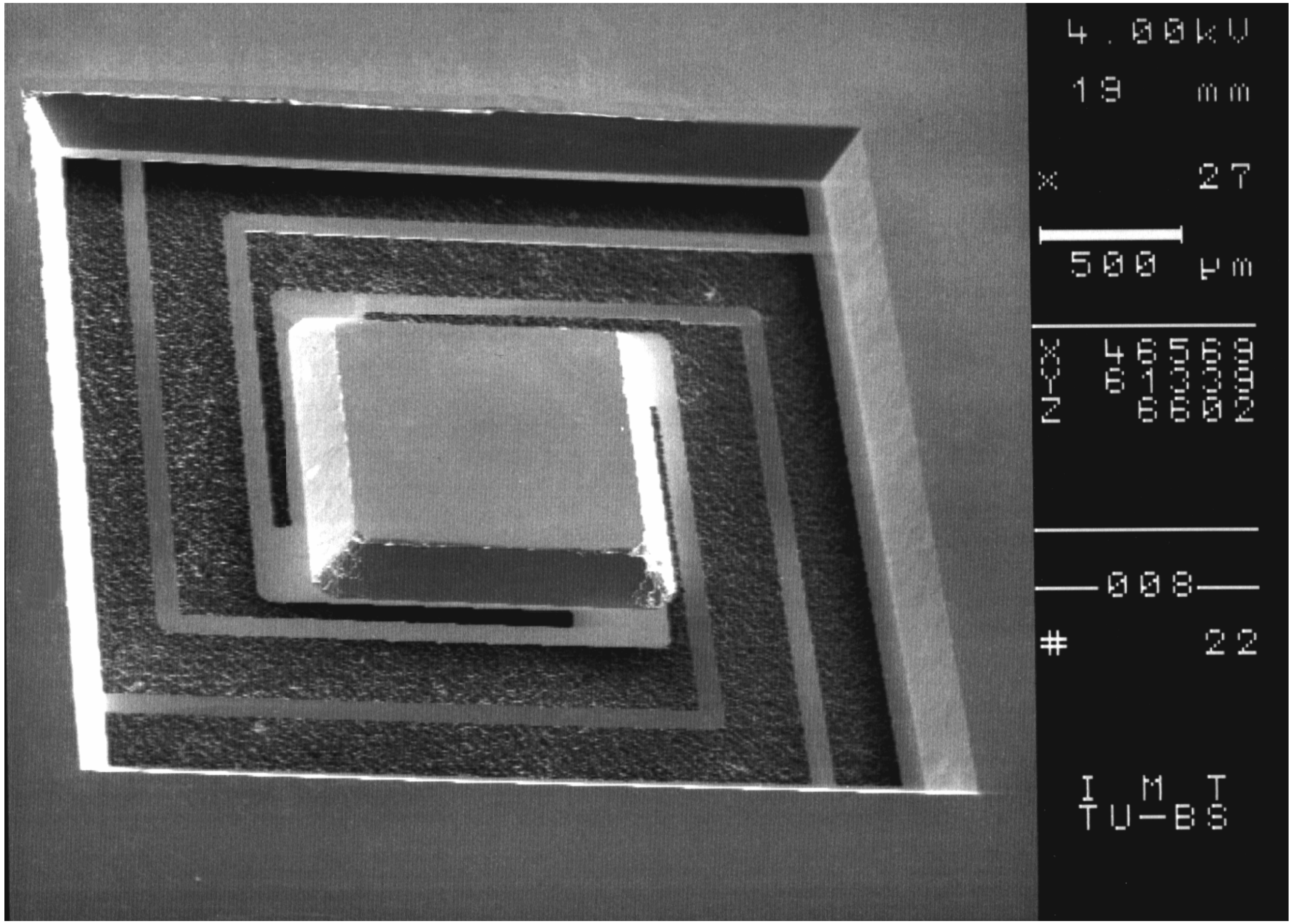
Many physical effects used for constructing sensors.

Examples:

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

Huge amount of sensors designed in recent years.

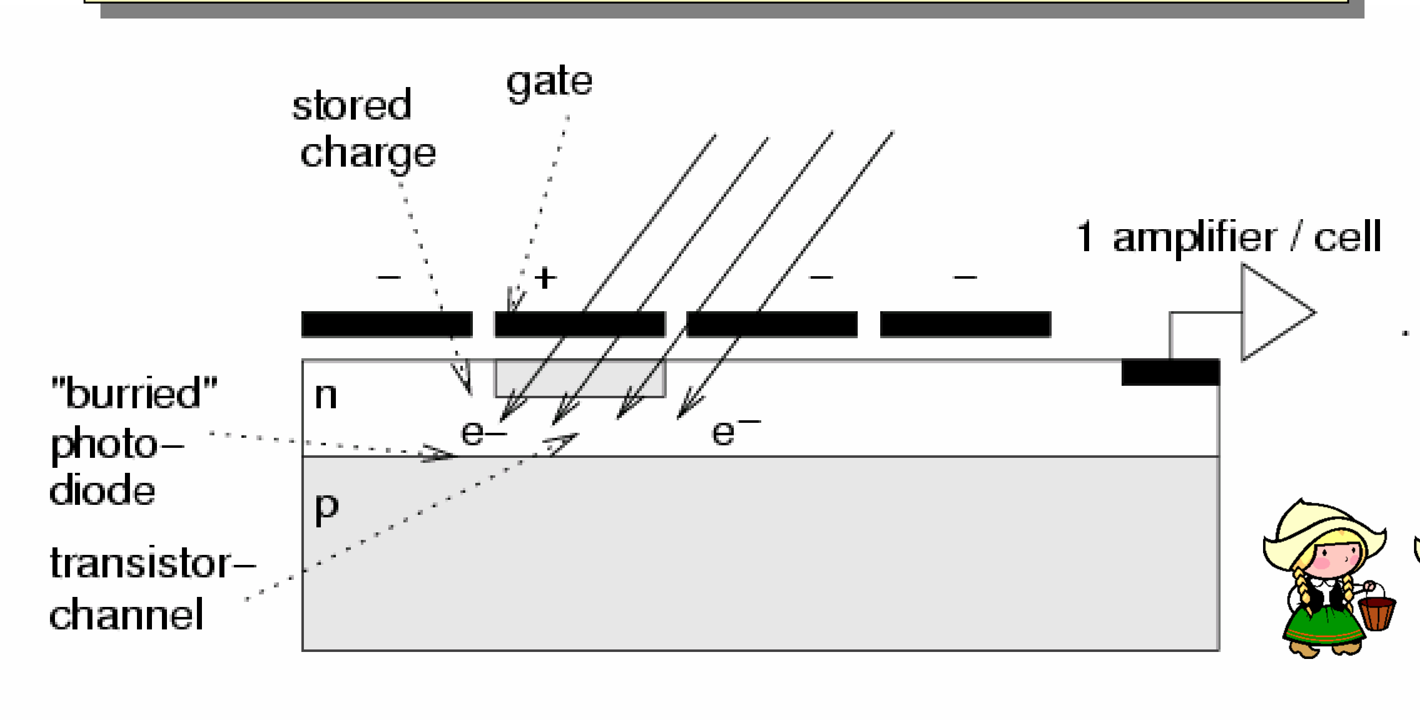
Example: Acceleration Sensor



Courtesy & ©: S. Bütgenbach, TU Braunschweig

Charge-coupled devices (CCD) image sensors

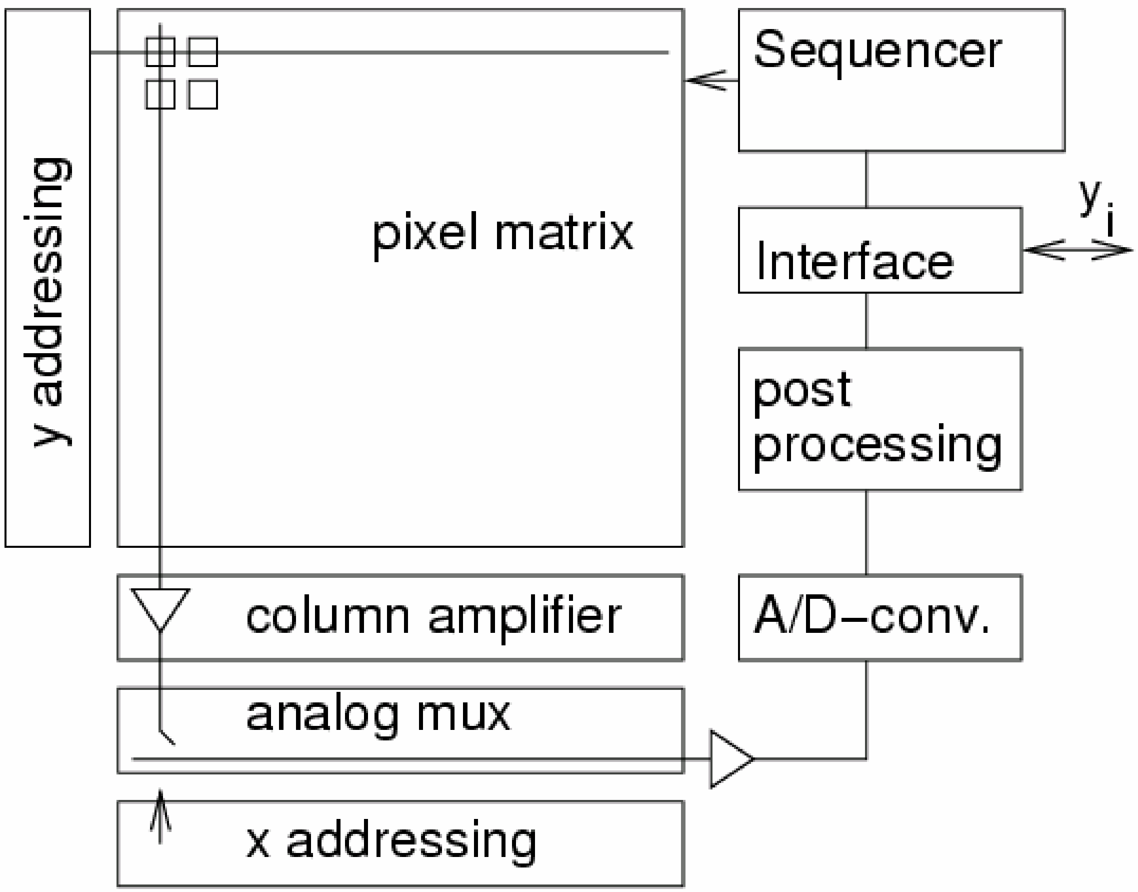
Based on charge transfer to next pixel cell



Corresponding to “bucket brigade device”
(German: “*Eimerkettenschaltung*”)

CMOS image sensors

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



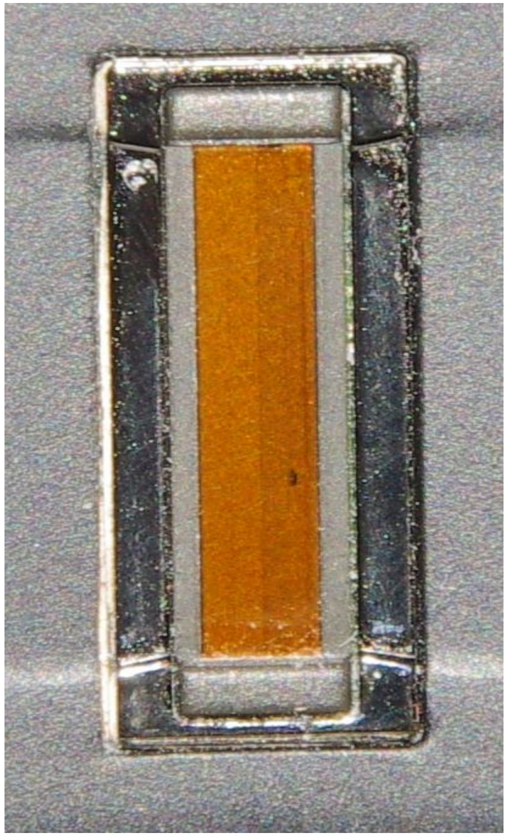
Comparison CCD/CMOS sensors

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)

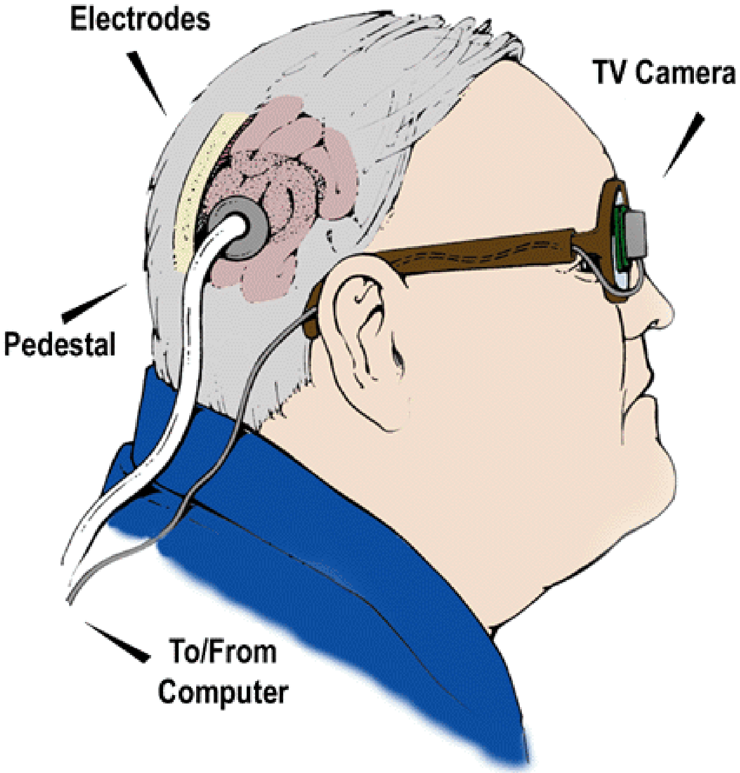
Example: Biometrical Sensors

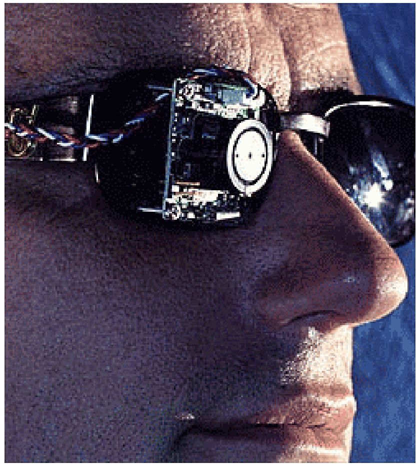
e.g.: Fingerprint sensor

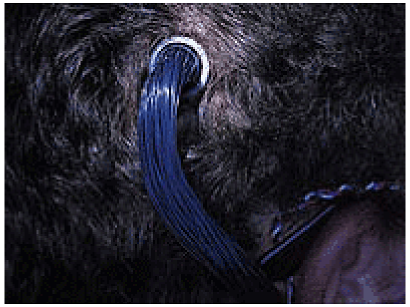


© P. Marwedel, 2010

Artificial eyes (1)







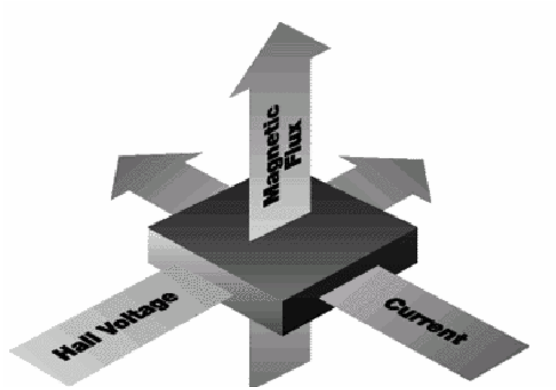
Show movie from www.dobelle.com
(e.g. blind person driving a car)

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
- Hall effect sensors



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.

technische universität
dortmund

fakultät für informatik
informatik 12

Discretization

Peter Marwedel
Informatik 12
TU Dortmund
Germany

Discretization of time

Digital computers require discrete sequences of physical values

$$s : D_T \rightarrow D_V$$

↑

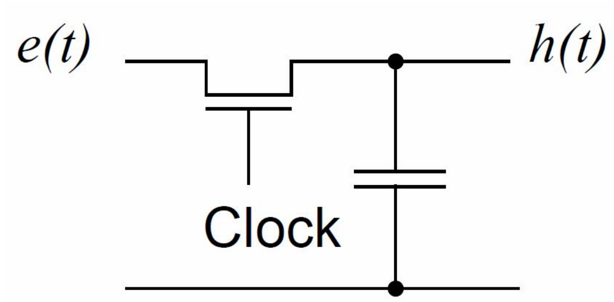
Discrete time domain

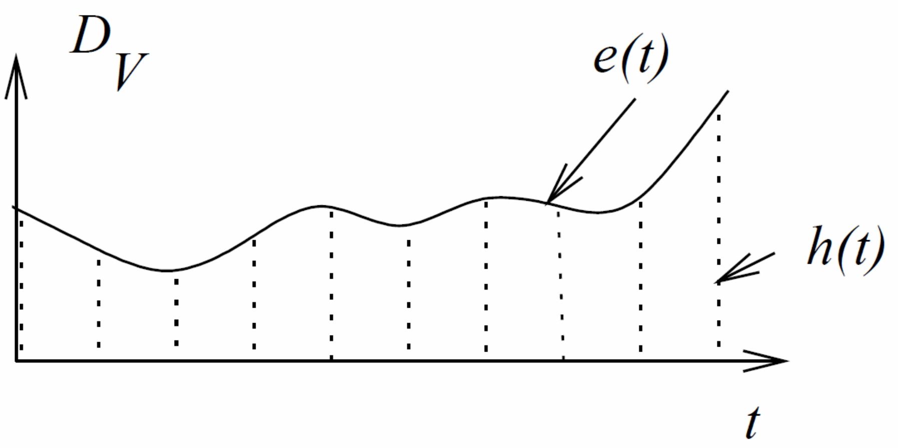
☞ Sample-and-hold circuits

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 $\mathbb{Z} \rightarrow \mathbb{R}$

Do we lose information due to sampling?

Would we be able to reconstruct input signals from the sampled signals?

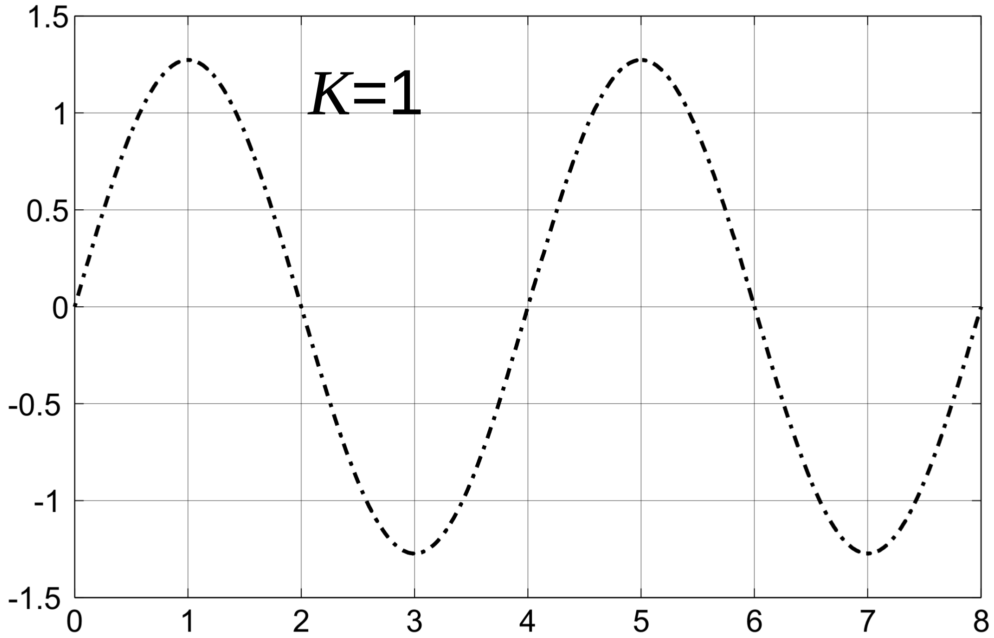
☞ approximation of signals by sine waves.

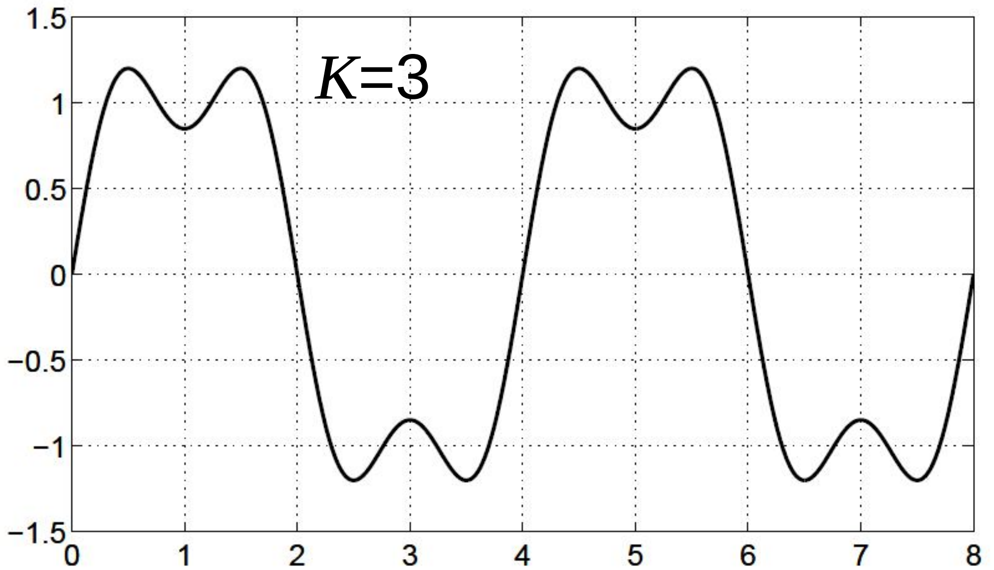
Approximation of a square wave (1)

Target: square wave with period p_1=4

$$e'_K(t) = \sum_{k=1,3,5,..}^{K} \frac{4}{\pi k} \sin\left(\frac{2\pi\, t}{p_k}\right)$$

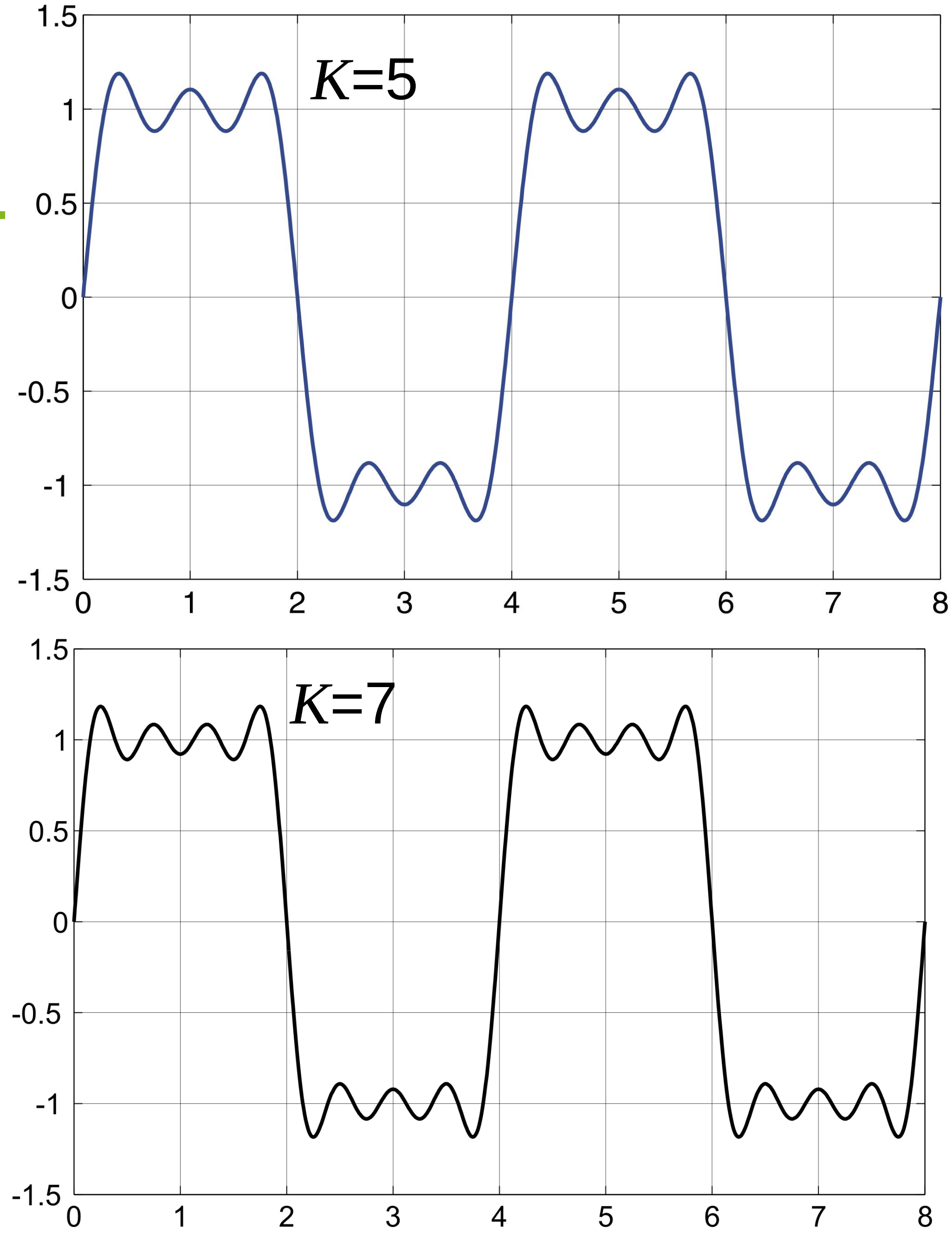
with $\forall k$: $p_k = p_1/k$: periods of contributions to e'





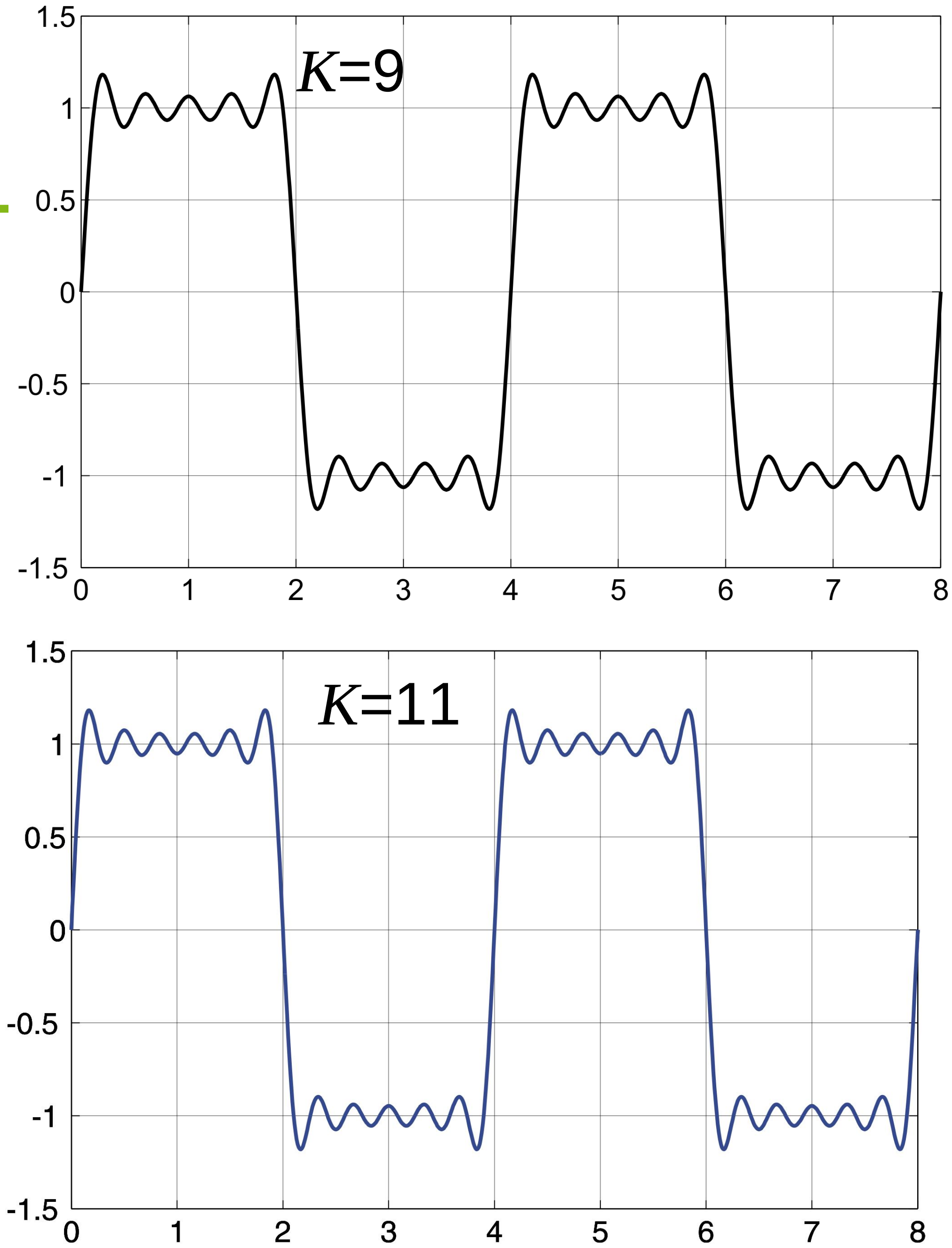
Approximation of a square wave (2)

$$e'_K(t) = \sum_{k=1,3,5,..}^{K} \frac{4}{\pi k} \sin\left(\frac{2\pi\, t}{4/k}\right)$$



Approximation of a square wave (3)

$$e'_K(t) = \sum_{k=1,3,5,..}^{K} \frac{4}{\pi k} \sin\left(\frac{2\pi\, t}{4/k}\right)$$



Linear transformations

Let $e_1(t)$ and $e_2(t)$ be signals

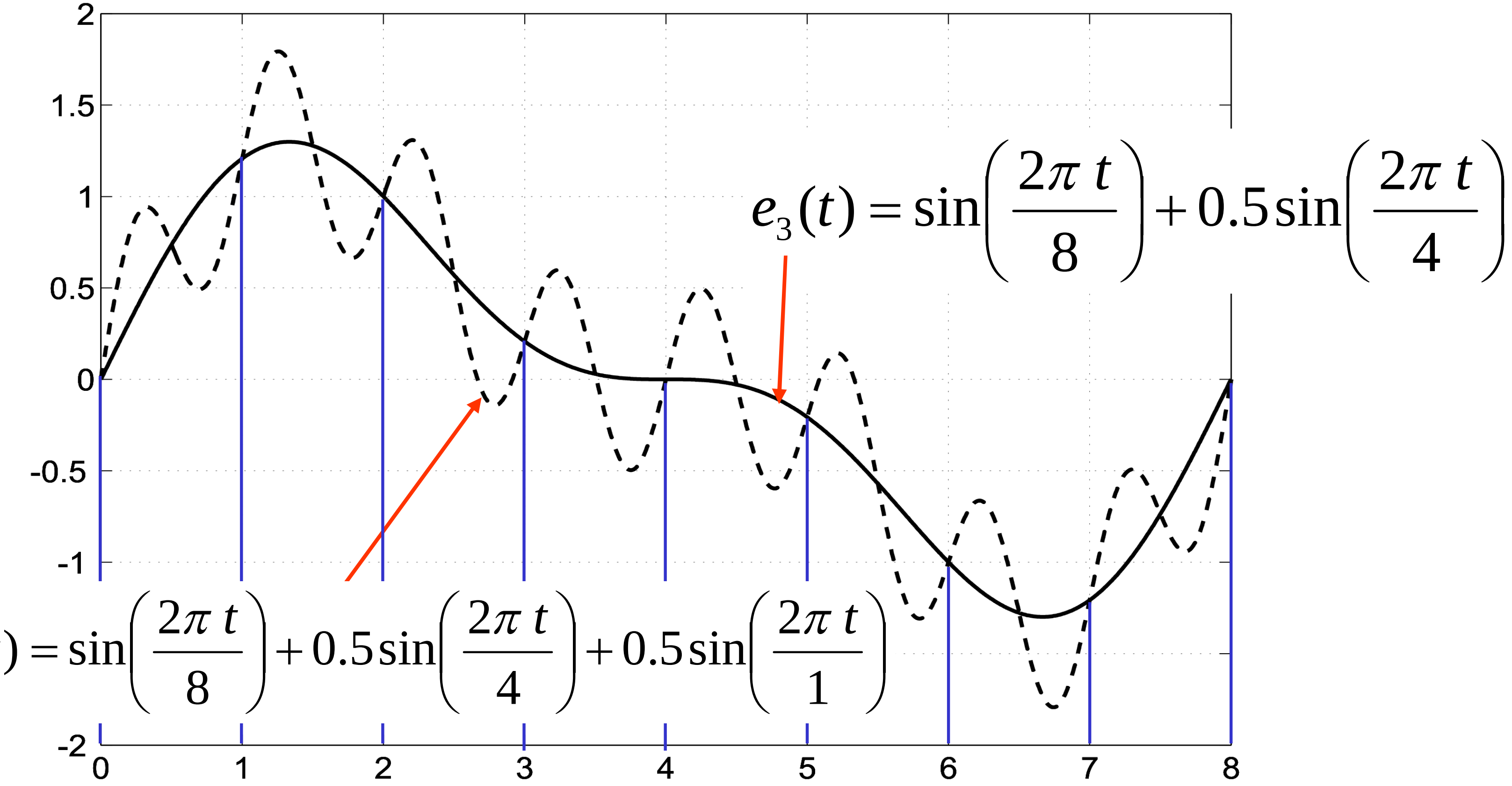
Definition: A transformation Tr of signals is linear iff

$$Tr(e_1 + e_2) = Tr(e_1) + Tr(e_2)$$

In the following, we will consider linear transformations.

☞ We consider sums of sine waves instead of the original signals.

Aliasing



$$e_3(t) = \sin\left(\frac{2\pi t}{8}\right) + 0.5\sin\left(\frac{2\pi t}{4}\right)$$

$$e_4(t) = \sin\left(\frac{2\pi t}{8}\right) + 0.5\sin\left(\frac{2\pi t}{4}\right) + 0.5\sin\left(\frac{2\pi t}{1}\right)$$

Periods of p=8,4,1

Indistinguishable if sampled at integer times, p_s=1

Aliasing (2)

☞ 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

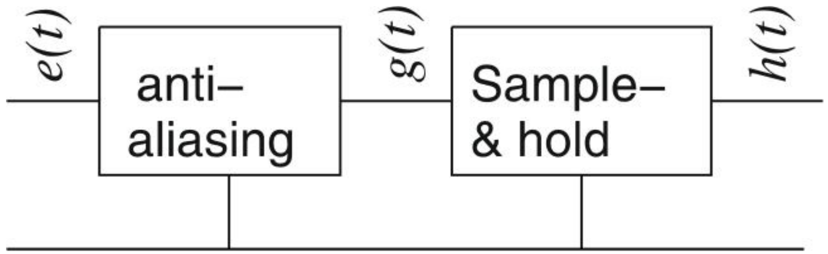
or $f_s > 2\ f_N$ where f_N is the frequency of the "fastest" sine wave

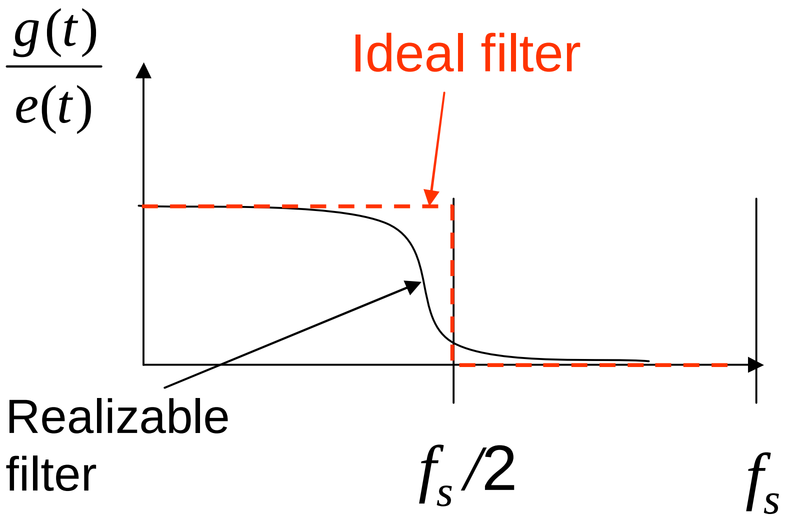
f_N is called the **Nyquist frequency**, f_s is the **sampling rate**.

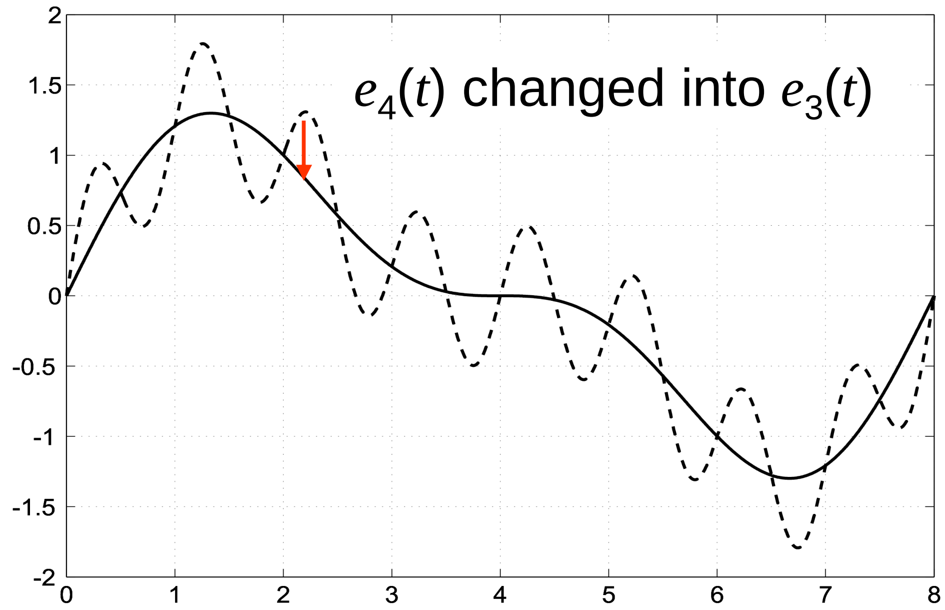
See e.g. [Oppenheim/Schafer, 2009]

Anti-aliasing filter

A filter is needed to remove high frequencies

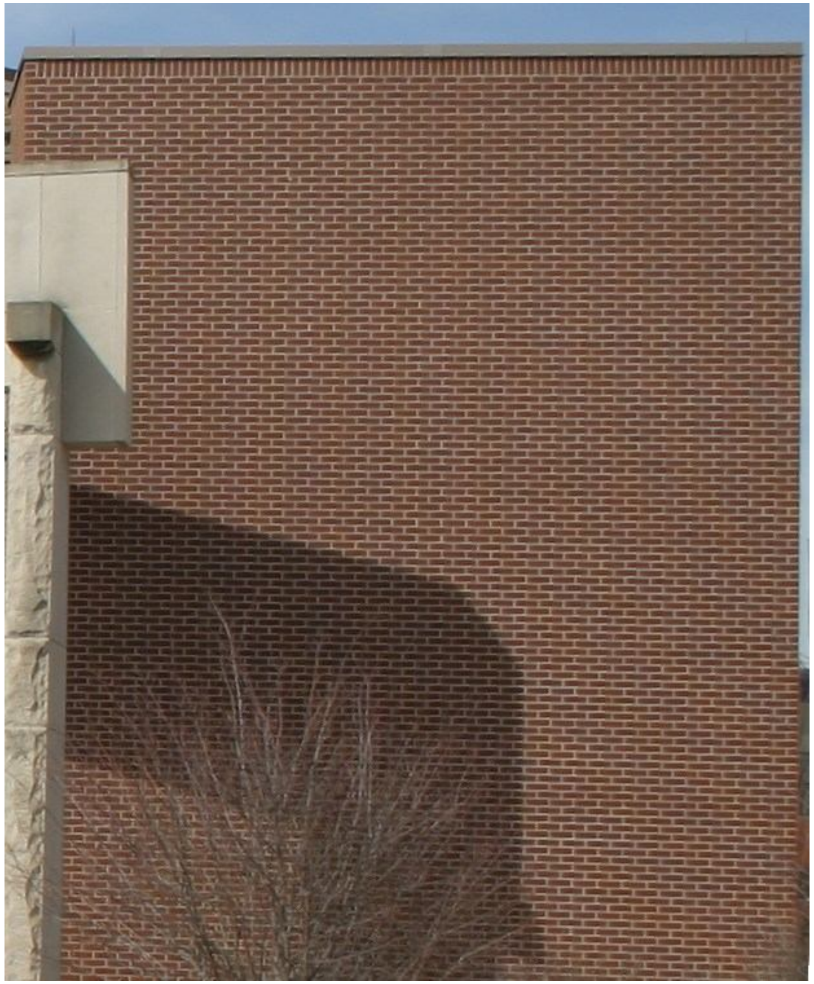






Examples of aliasing in computer graphics

Original

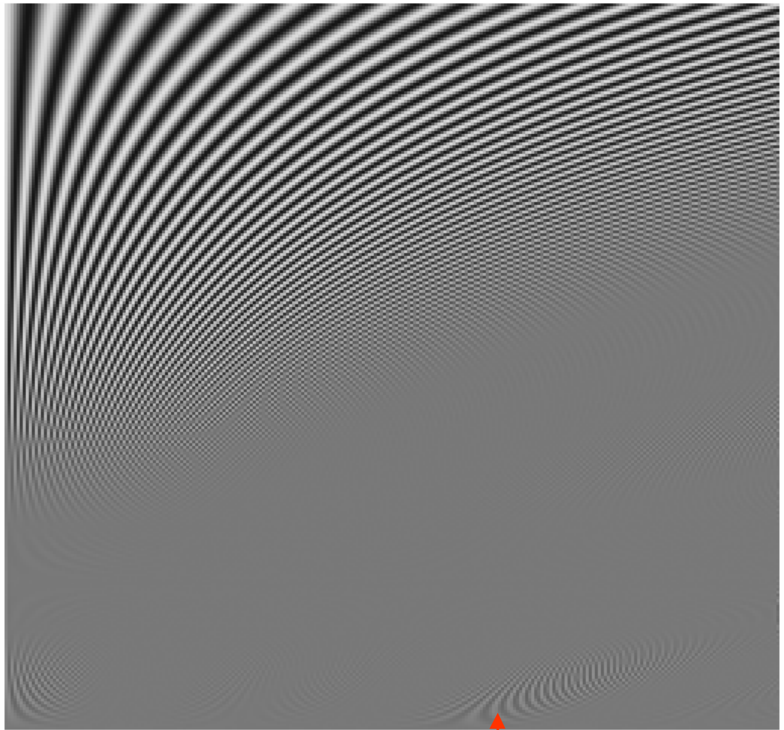


Sub-sampled, no filtering

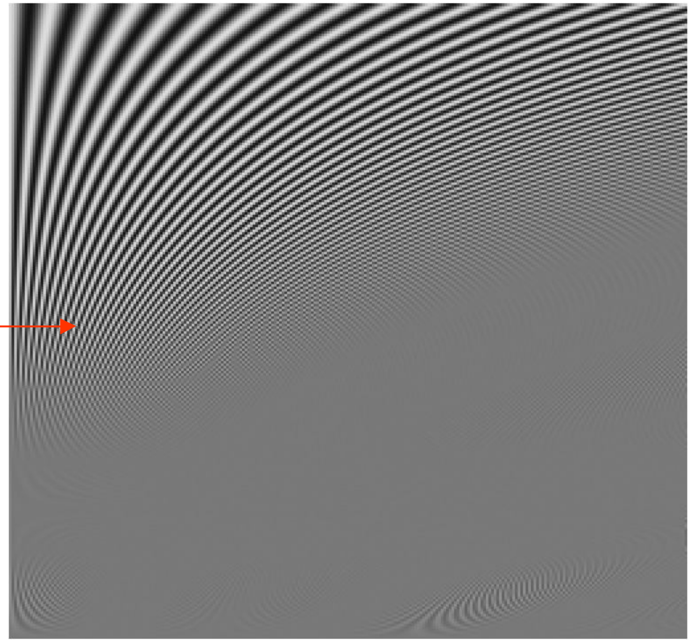
http://en.wikipedia.org/wiki/Image:Moire_pattern_of_bricks_small.jpg

Examples of aliasing in computer graphics (2)

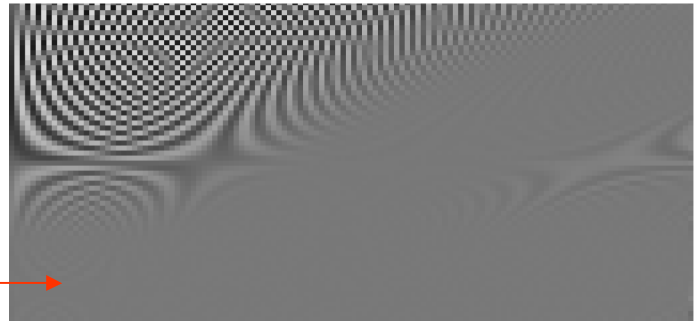
Original (pdf screen copy)



Filtered & sub-sampled



Sub-sampled, no filtering



Impact of rasterization

http://www.niirs10.com/Resources/ Reference Documents/Accuracy in Digital Image Processing.pdf

Discretization of values: A/D-converters

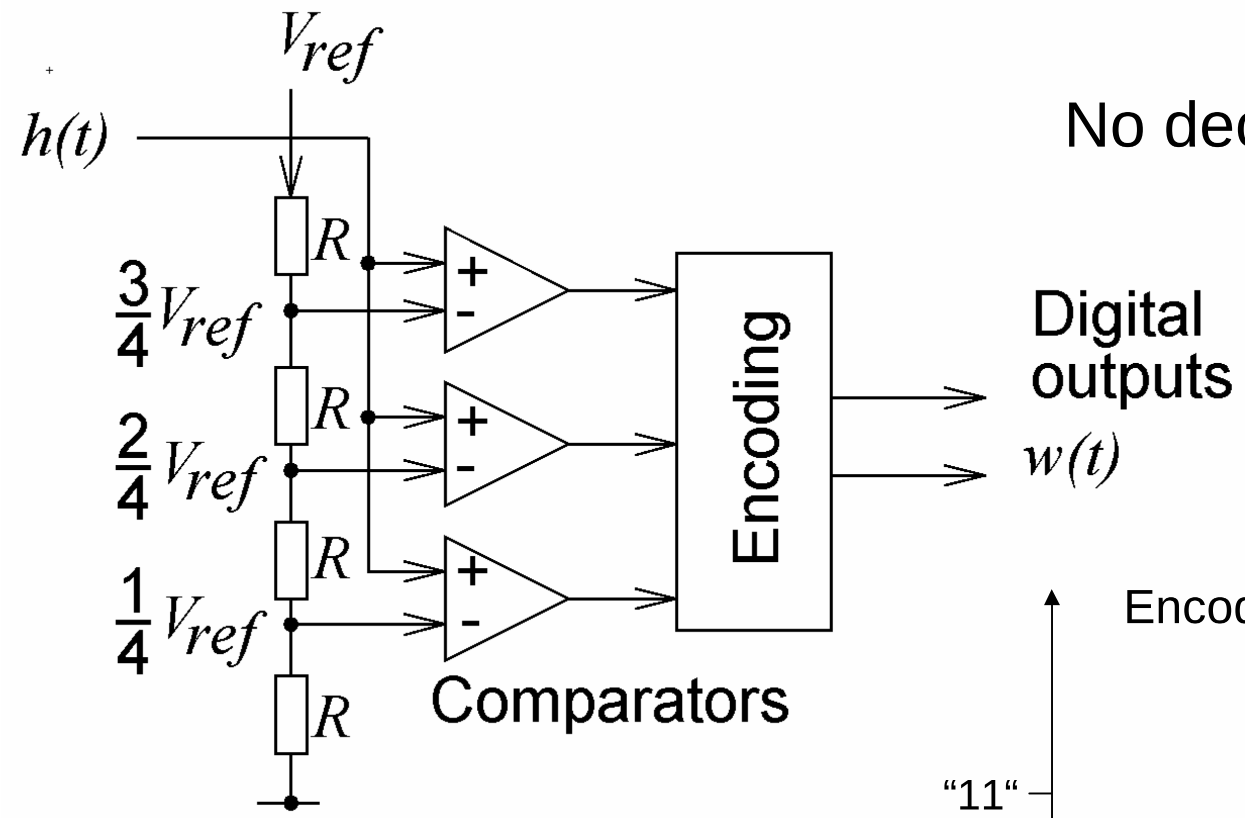
Digital computers require digital form of physical values

$$s: D_T \rightarrow D_V$$

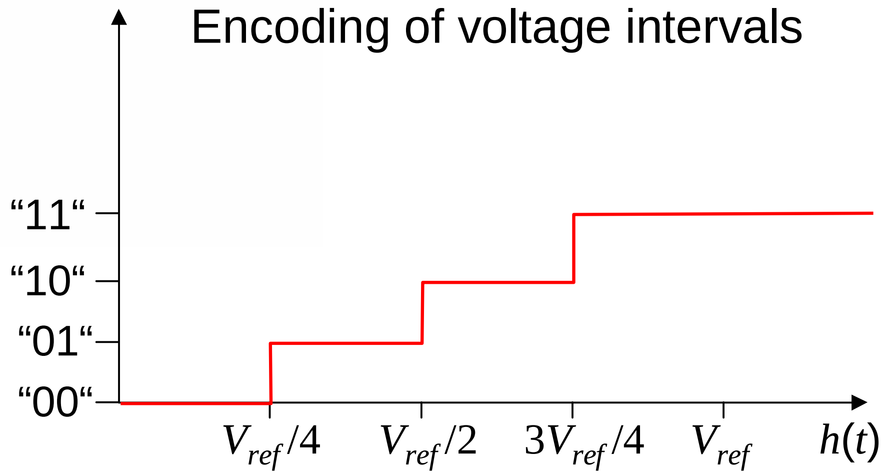
↑ Discrete value domain

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

Flash A/D converter



No decoding of $h(t) > V_{ref}$



Resolution

- Resolution (in bits): number of bits produced
- Resolution Q (in volts): difference between two input voltages causing the output to be incremented by 1

$$Q = \frac{V_{FSR}}{n}$$ with

Q: resolution in volts per step
V_{FSR}: difference between largest and smallest voltage
n: number of voltage intervals

Example:
$Q = V_{ref}/4$ for the previous slide

Resolution and speed of Flash A/D-converter

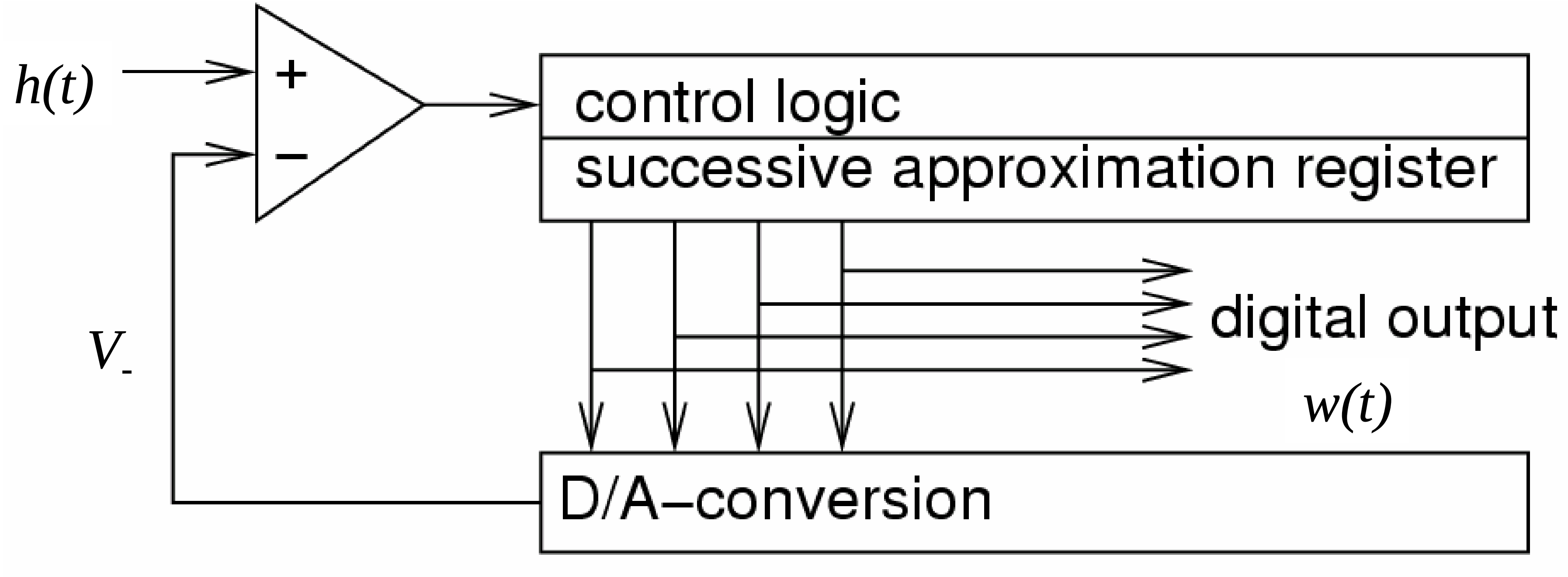
Parallel comparison with reference voltage

Speed: $O(1)$

Hardware complexity: $O(n)$

Applications: e.g. in video processing

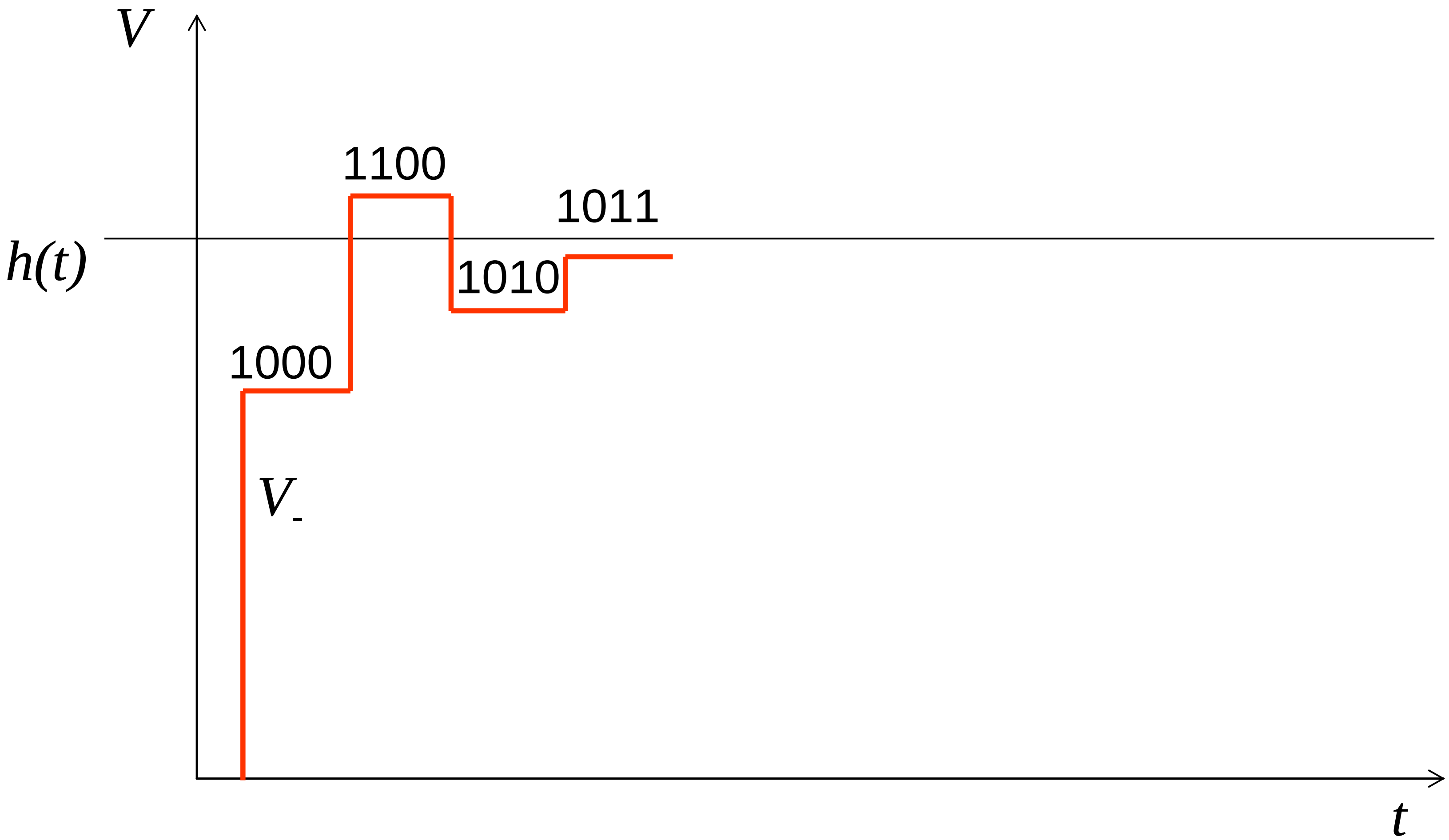
Higher resolution: Successive approximation



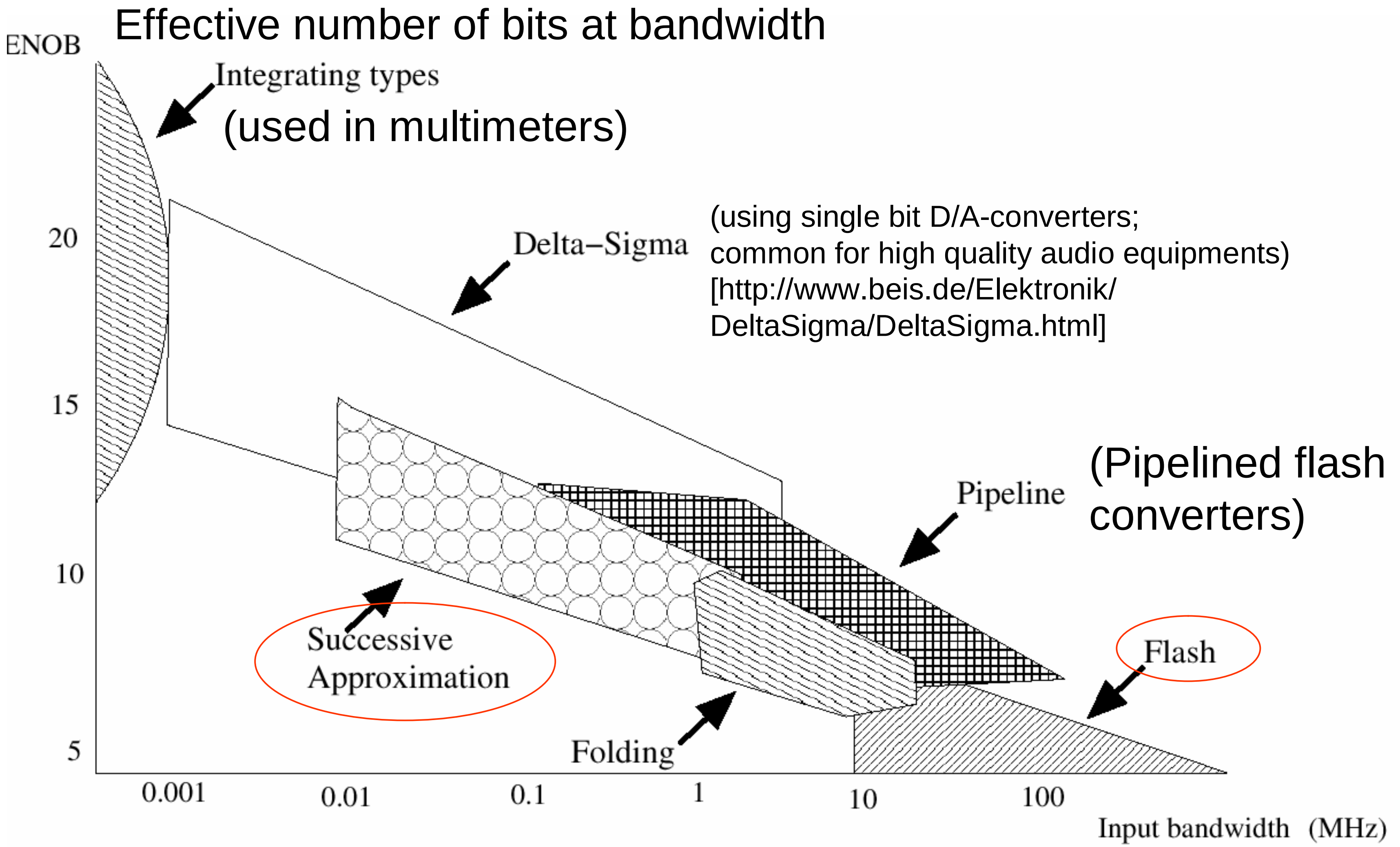
Key idea: binary search:
Set MSB='1'
if too large: reset MSB
Set MSB-1='1'
if too large: reset MSB-1

Speed: $O(\log_2(n))$
Hardware complexity: $O(\log_2(n))$
with n= # of distinguished voltage levels;
slow, but high precision possible.

Successive approximation (2)

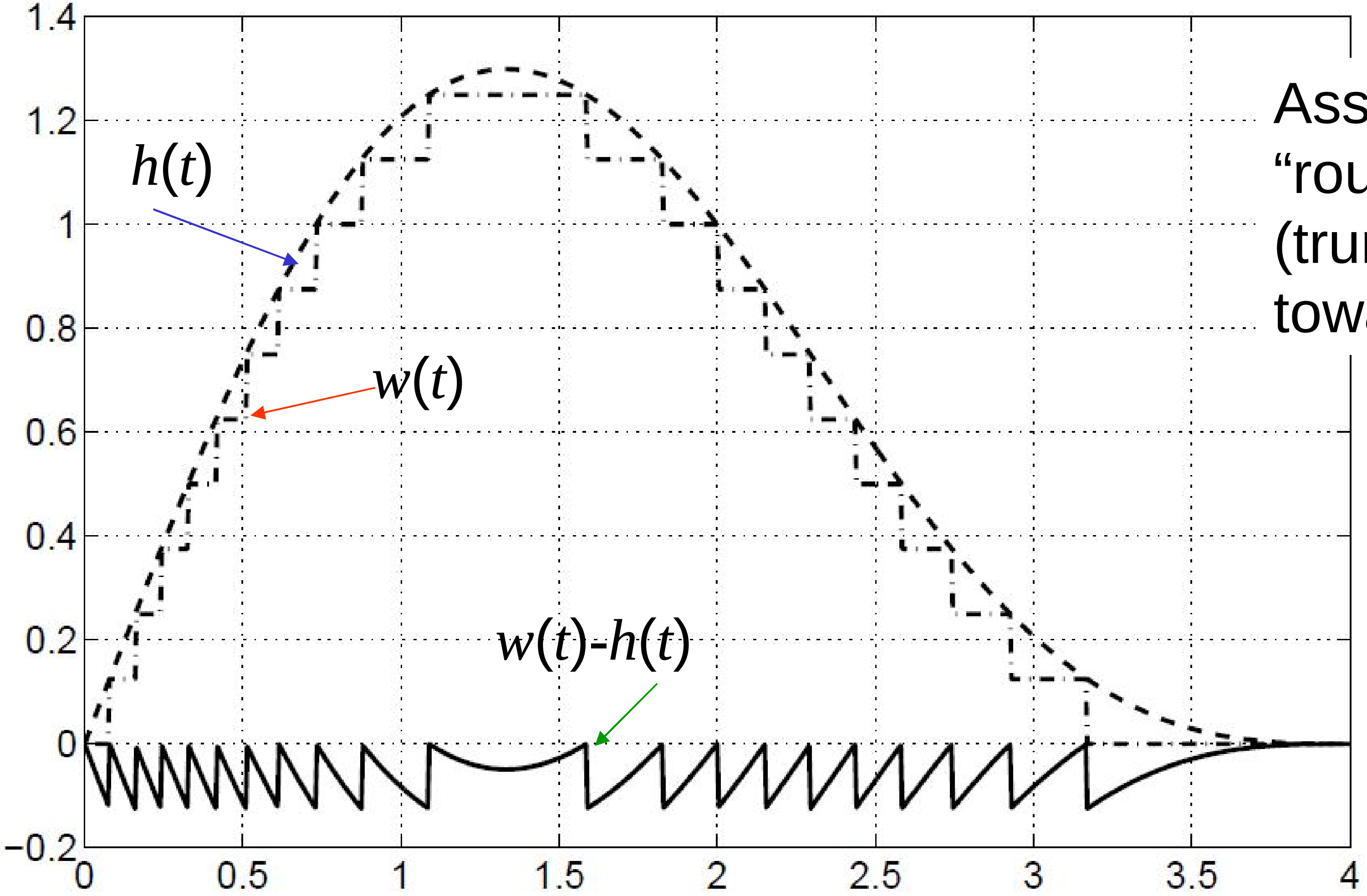


Application areas for flash and successive approximation converters



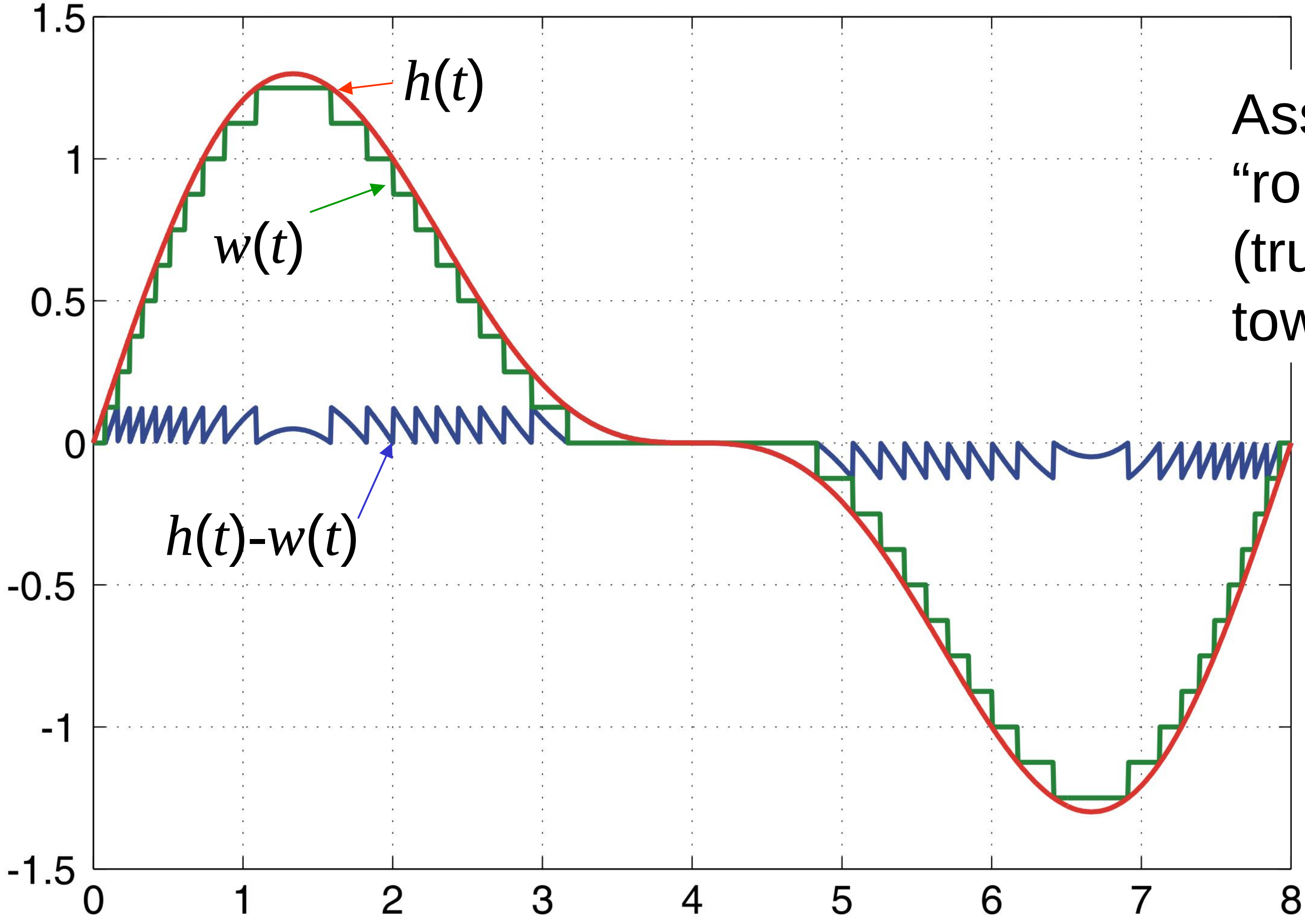
[Gielen et al., DAC 2003]

Quantization Noise



Assuming “rounding“ (truncating) towards 0

Quantization Noise



Assuming “rounding“ (truncating) towards 0

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

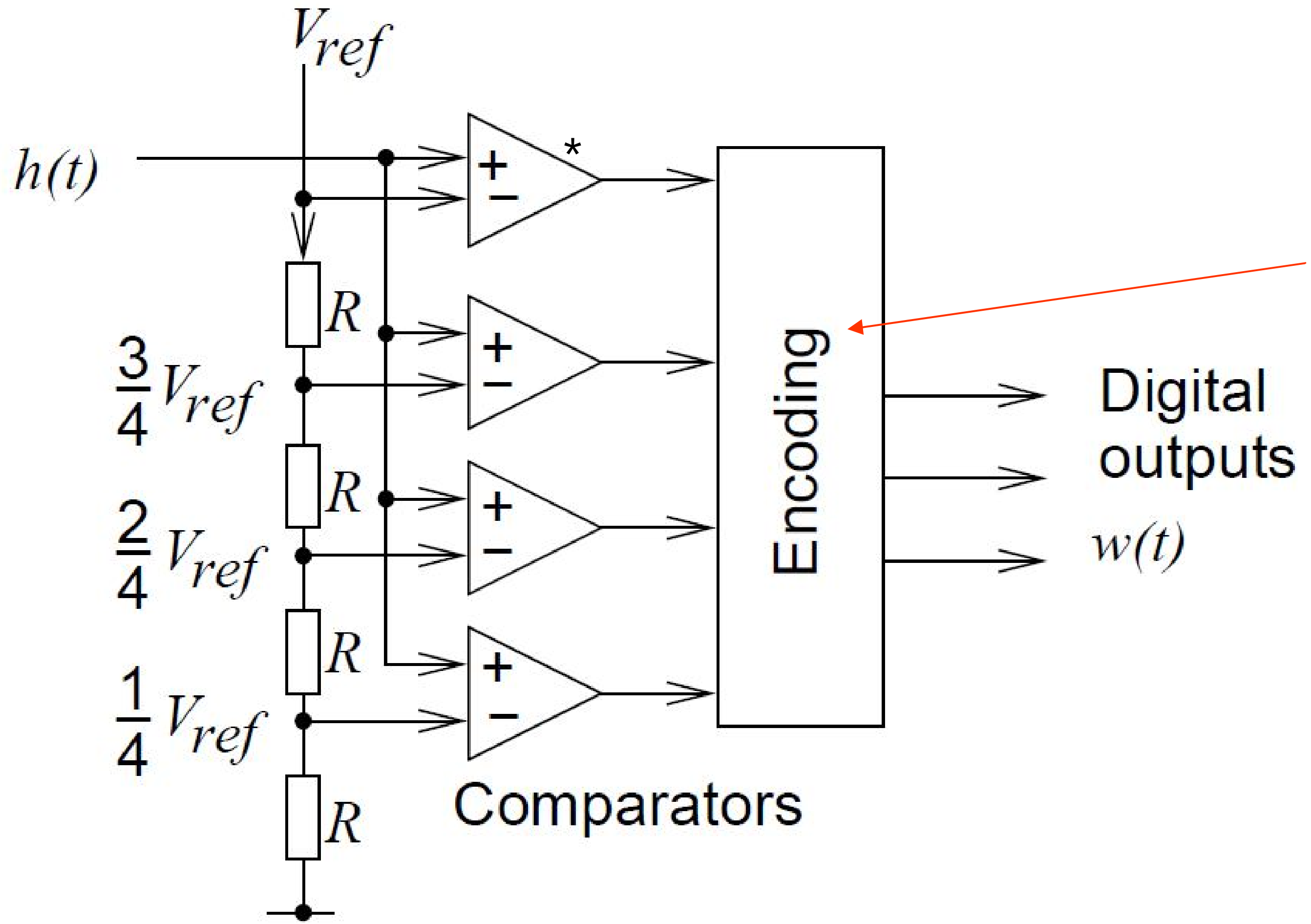
Summary

Hardware in a loop

- Sensors
- Discretization
    - Sample-and-hold circuits
        - Aliasing (and how to avoid it)
        - Nyquist criterion
    - A/D-converters
        - Quantization noise

SPARE SLIDES

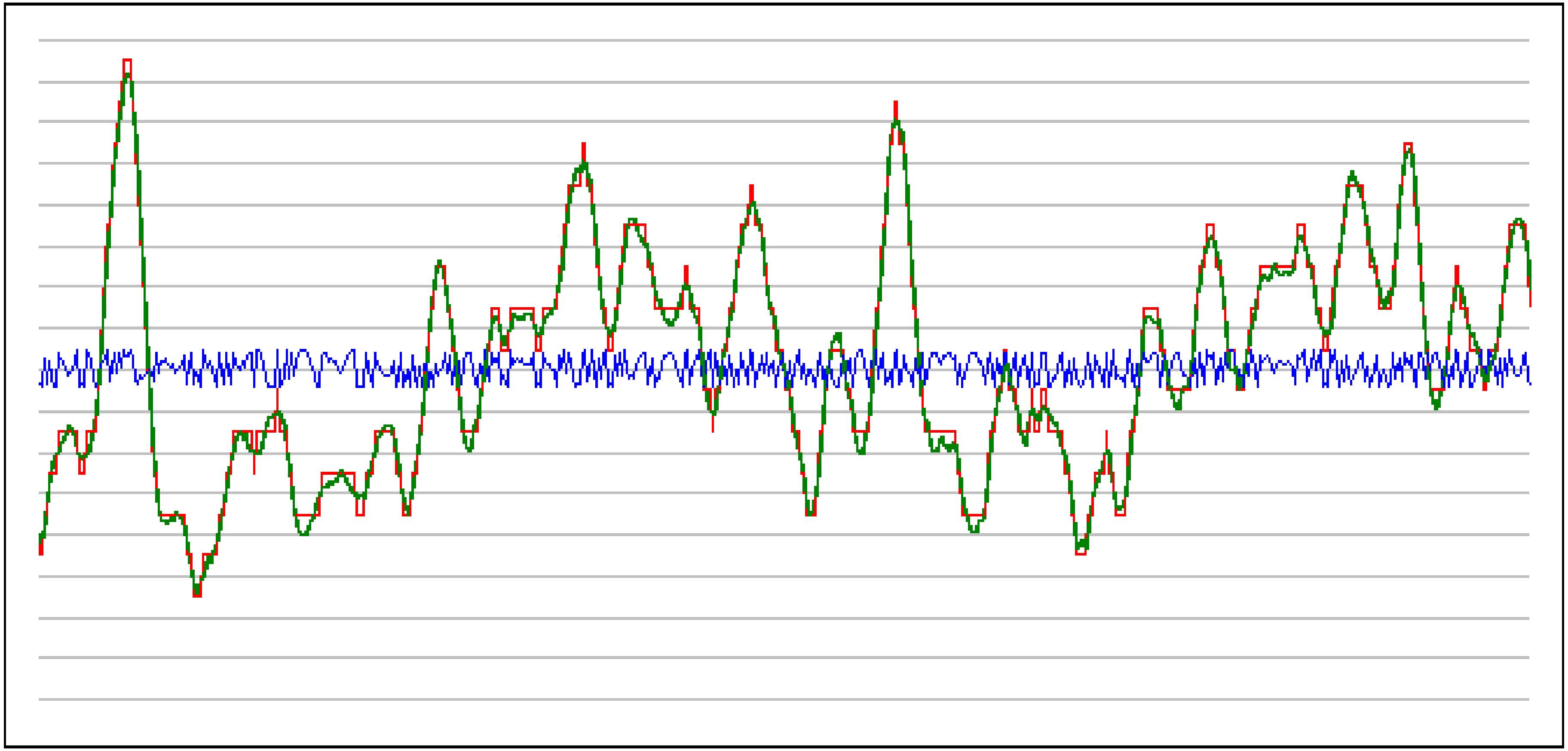
Flash A/D converter



Encodes input number of most significant ‘1’ as an unsigned number, e.g.
“1111” -> “100”,
“0111” -> “011”,
“0011” -> “010”,
“0001” -> “001”,
“0000” -> “000”
(Priority encoder).

* Frequently, the case $h(t) > V_{ref}$ would not be decoded

Quantization noise for audio signal



Source: [http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html]