

Embedded Hardware Fundamentals

Embedded Systems / Cyber-Physical System Fundamentals (ES/CPSF)

Peter Ulbrich

Course website

Winter Semester 2026/27

Recap: From Embedded Systems to Physical Loops

Embedded systems

Computation is part of an engineered product and serves its physical purpose.

Cyber-physical systems

Sensing, computation, and actuation close a feedback loop through the environment.

Design constraints

Timeliness, dependability, cost, and efficiency constrain the complete system.

Heating, lighting, engine control, power grids, and robots are examples from the last lecture.

- **Today we open the hardware part of that loop: what does the platform do to our signals, timing, and energy?**

Why Hardware Matters

Hardware connects software to the physical world. We follow a measurement from the sensor to a digital result, then examine its timing and energy cost.

Time

Caches, buses, pipelines, and interrupts shape execution time.

Resources

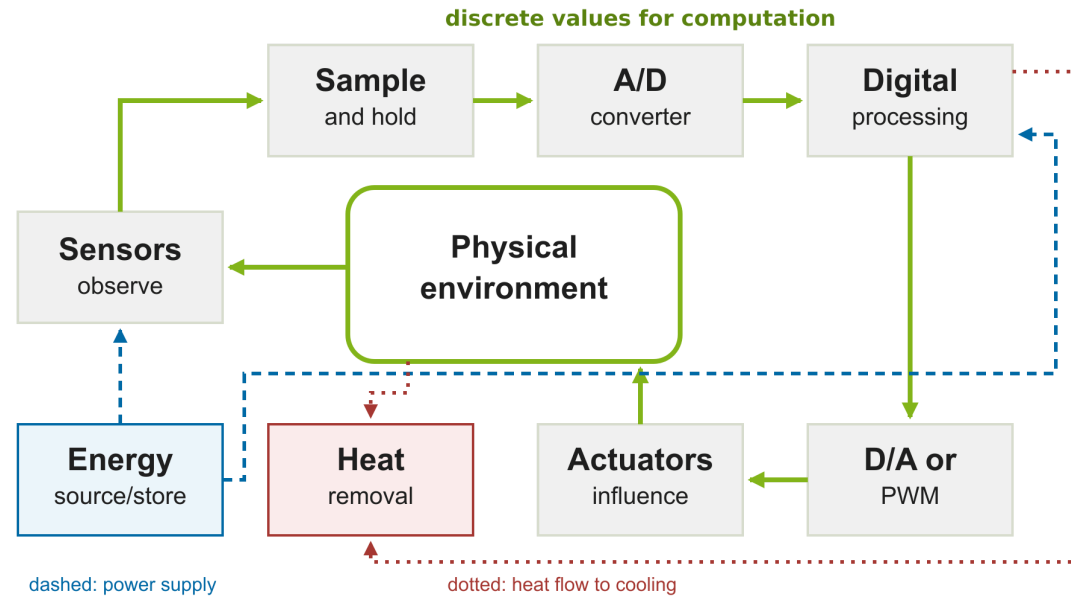
Memory, available energy, and component cost limit the software design.

Physics

Noise, calibration, heat, and physical failures affect correctness and reliability.

► Software assumptions must agree with the hardware platform and the physical process.

The Hardware Loop



Hardware loop after Marwedel / Chen. Solid green: signal/control flow; dashed blue: power supply; dotted red: heat removal.

Learning Objectives

By the end of this chapter, you should be able to:

- ▶ **Sensors:** Relate **sensor physics and readout** to quality, latency, and energy.
- ▶ **Signals:** Choose **sampling rate and analog filtering** to preserve the required bandwidth.
- ▶ **A/D conversion:** Calculate **resolution**; compare conversion time, power, and precision.
- ▶ **Processing:** Analyze **code density, timing, and arithmetic**, including fixed-point and saturation.
- ▶ **Energy:** Apply **CMOS, DVFS, and DPM models**; assess parallel and heterogeneous execution under deadlines.

Chapter 2: From Physics to Computation

Sensors

physical effects

Signals

time and bandwidth

A/D Conversion

values and codes

Processing

code and arithmetic

Energy

power and energy

- Follow one chain: physical measurement → sampled signal → digital code → useful computation → energy and timing constraints.

Sensors: Emergency Braking Assistance

An emergency braking assistant must detect an obstacle **early enough to react**.

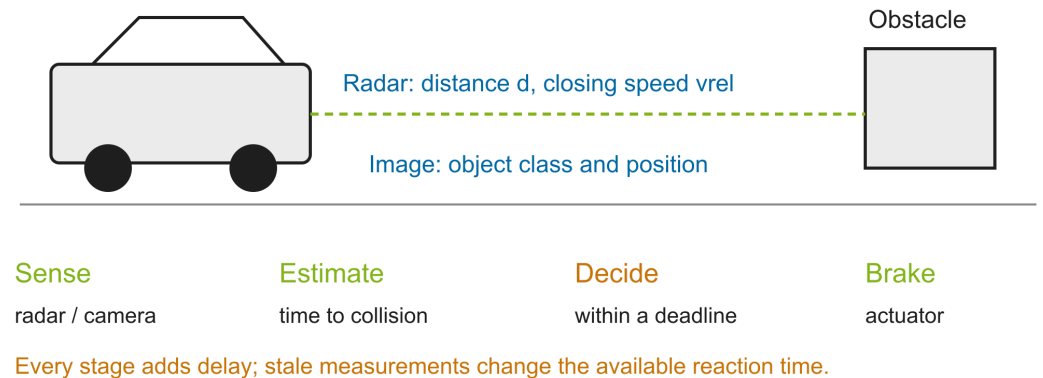
- **Radar:** distance and relative velocity from reflected radio waves.
- **Camera:** image data for object and lane recognition.
- **Wheel sensors:** motion estimates from magnetic or optical effects.

A sensor converts a physical quantity into an electrical signal; software interprets that signal.

- **Distance alone is not enough: measurement age, uncertainty, and relative speed affect the decision.**

Simplified teaching example, not a production braking algorithm. EBA: emergency braking assistance.

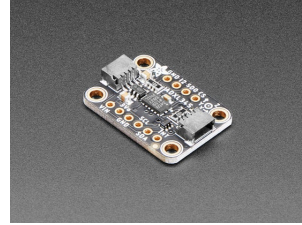
A measurement becomes a decision



Sensor Families

Motion

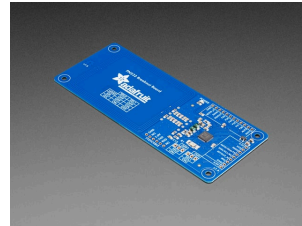
Acceleration and angular rate for navigation, balance, and control.



ADXL345: MEMS accelerometer

Optical

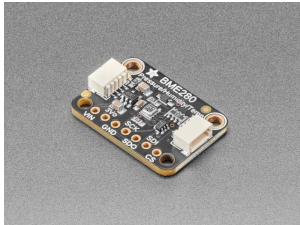
Light intensity and images for perception and measurement.



PN532: RFID / NFC reader

Environment

Pressure, temperature, and chemistry expose operating conditions.



BME280: environmental sensor

Automotive

Position, speed, and pressure support engine and safety functions.

Smart sensors

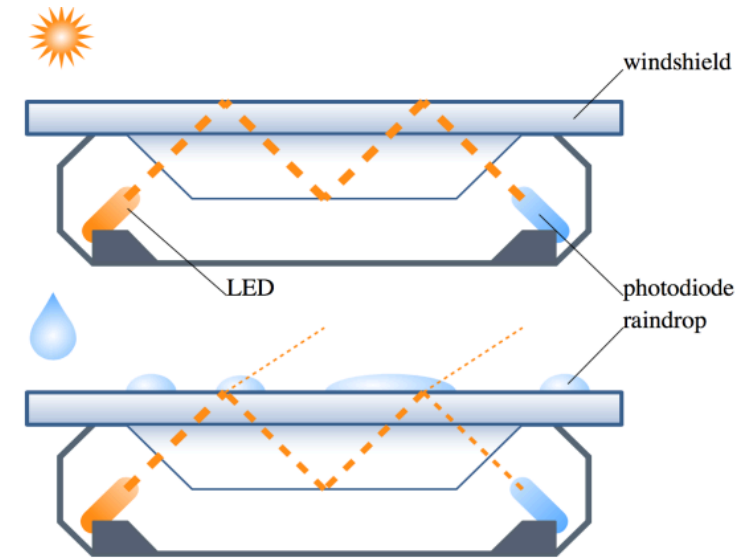
Integrated logic preprocesses data close to its physical source.

Hardware photographs: Adafruit ([ADXL345](#), [PN532](#), [BME280](#)). Breakout boards add power and interface circuitry around the sensor chip.

Rain Sensors

Infrared light enters the windshield at about **45 degrees**.

- Dry glass returns much of the light by internal reflection.
- Water changes the optical boundary: less light reaches the receiver.
- The wiper controller reacts to the measured intensity.



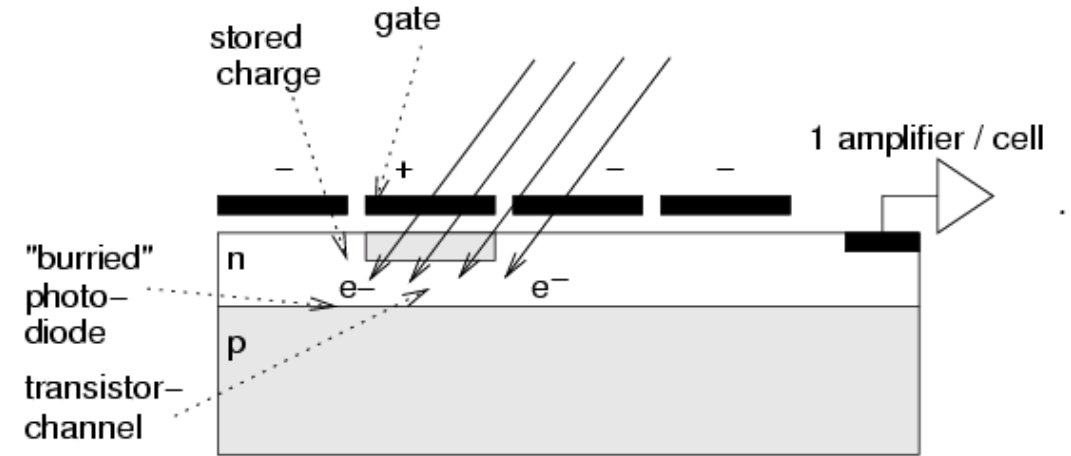
► **A physical effect becomes a measurement without directly measuring water volume.**

Chen / Marwedel 2024, Inputs p.8.

CCD Image Sensors

Photons create charge packets in pixel cells.

- Clocked gates transfer charge to the next cell.
- Sequential transfer brings charge to the readout amplifier.
- Readout and A/D conversion produce image data.

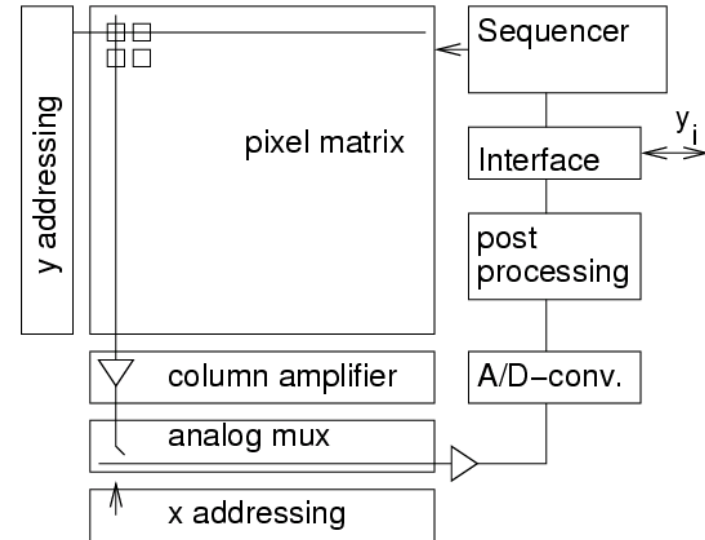


► The measured value is physically transported before it becomes a digital number.

CMOS Image Sensors

Standard CMOS production allows pixel circuitry and logic on one chip.

- Row/column addressing selects pixels.
- Amplifiers, multiplexers, and ADCs provide readout.
- Timing, interface logic, and preprocessing can be integrated.



► **Memory-like access and integration distinguish CMOS readout from serial CCD charge transfer.**

Chen / Marwedel 2024, Inputs p.10.

CCD and CMOS Image Sensors

Property	CCD	CMOS
Readout principle	Sequential charge transfer	Addressed pixels, often parallel ADCs
Timing / region of interest	Serial transfer limits readout speed	Parallel readout and ROI can reduce latency
Energy: typical modern designs	Charge transfer and external circuits add overhead	Integration often reduces power
On-chip processing	Usually external	Can share the sensor chip
Shutter / motion artifacts	Architecture and device dependent	Rolling or global shutter, device dependent
Typical strength	Specialized scientific imaging, e.g. EMCCD	Video, embedded vision, integrated cameras

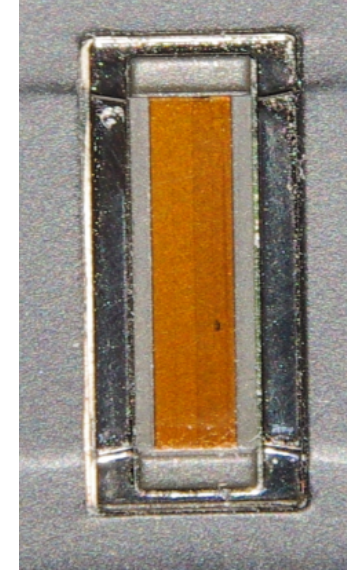
- Compare **end-to-end latency and energy per frame**, not just pixel count. Rankings remain device dependent.

Updated comparison: [Teledyne, CCD vs CMOS](#). Chen's power ranking is not a reliable rule for modern sensors.

Biometrical Sensors

Fingerprint sensors acquire a physical surface pattern using optical, capacitive, or thermal effects.

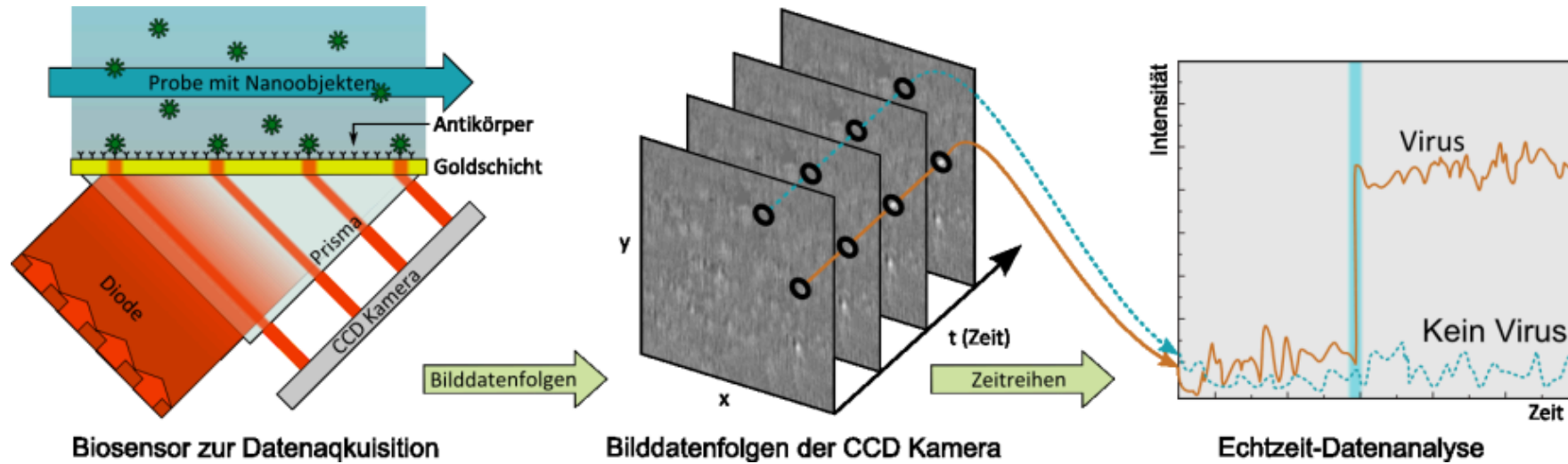
- Matching thresholds trade false acceptance against false rejection.
- Calibration, contamination, aging, and spoofing affect the system.



► **Sensor acquisition and the identity decision are separate parts of the system.**

Fingerprint sensor photograph: Marwedel 2010 / Chen Inputs p.12.

PAMONO: An Embedded Monitoring Pipeline



► **Binding at a coated surface** → **CCD image sequence** → **real-time detection**.

The embedded challenge: distinguish small optical changes from artifacts within limited **time**, **computing capacity**, and **energy**. Monitoring does not require an actuator.

Figure: Chen Inputs p.13 (labelled "PAMANO"); project name: PAMONO. *ISAS: optical detection and resource-optimized real-time analysis*, 2020.

Other Embedded Sensors

Sensor class	Example quantity	Typical embedded use
Pressure	tire, manifold, liquid pressure	safety monitoring, engine control
Proximity	distance, presence, touch	robotics, HMI, collision avoidance
Hall effect	magnetic field, wheel position	commutation, speed sensing, crankshaft position
Chemical	gas concentration, humidity, pollutants	environmental and industrial monitoring

- The common abstraction is simple, but the implementation physics determines accuracy, latency, failure modes, and cost.

Signals: What Must Sampling Preserve?

Sensors

physical effects

Signals

time and bandwidth

A/D Conversion

values and codes

Processing

code and arithmetic

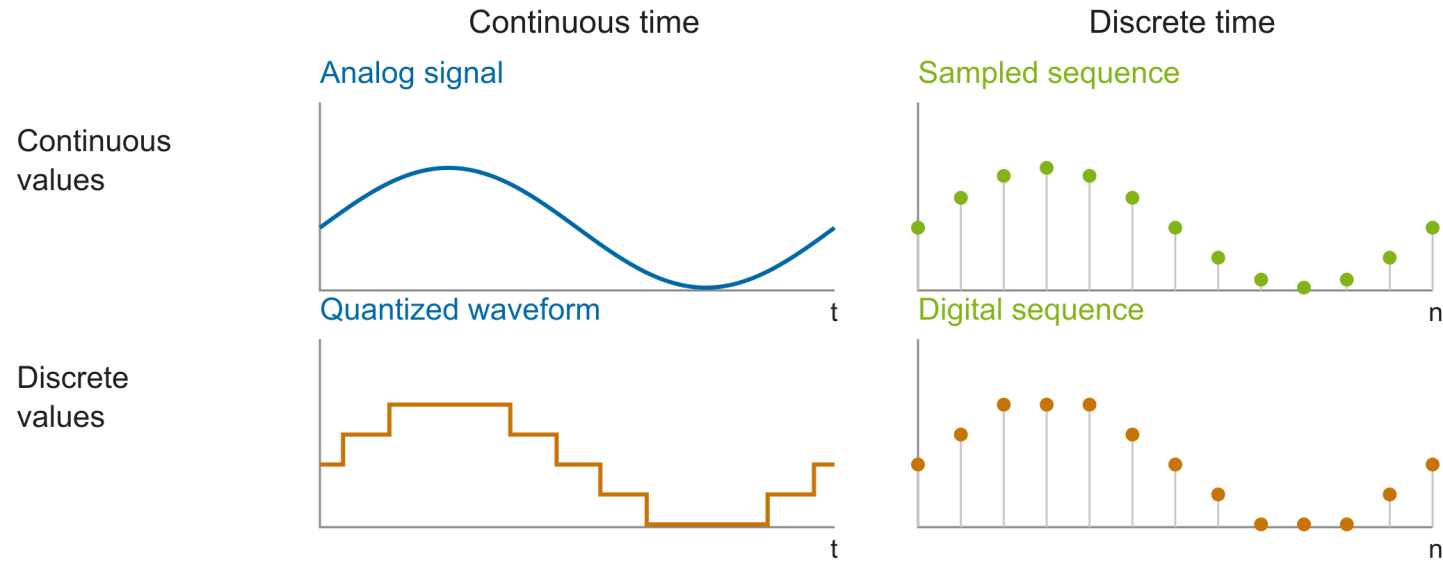
Energy

power and energy

- ▶ A sensor produces a signal. Before software can use it, we must preserve the relevant information while discretizing time.

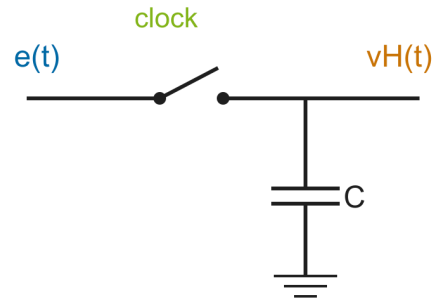
Signal Domains

A signal maps time to values: $s : D_T \rightarrow D_V$. **Sampling** discretizes time; **quantization** discretizes values.



Sample-and-Hold Circuits

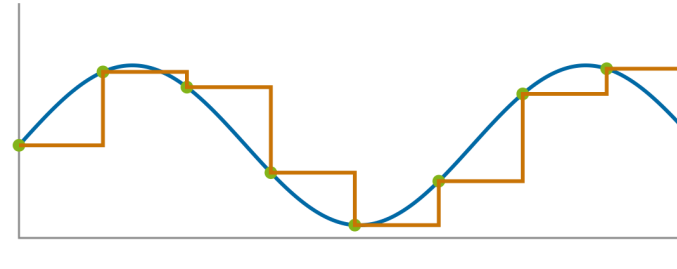
Clocked switch and storage capacitor



Track: switch closed; C follows input

Hold: switch open; C stores voltage

Analog input and held voltage



Blue: $e(t)$ Orange: ideal zero-order hold

Green: $h[n] = e(nT_s)$, continuous-valued samples

Short acquisition windows idealized as instants

$h[n] = e(nT_s)$ is a **discrete-time sequence**; the capacitor output remains an **analog voltage** held for conversion.

► Track: charge and settle. Hold: keep the input stable while the ADC determines its code.

Fourier Approximation

Can we reconstruct a continuous signal from samples? Fourier decomposition lets us reason about frequency components.

$$e'_K(t) = \sum_{k=1,3,\dots,K} \frac{4}{\pi k} \sin\left(\frac{2\pi t}{p_k}\right), \quad p_k = p_1/k$$

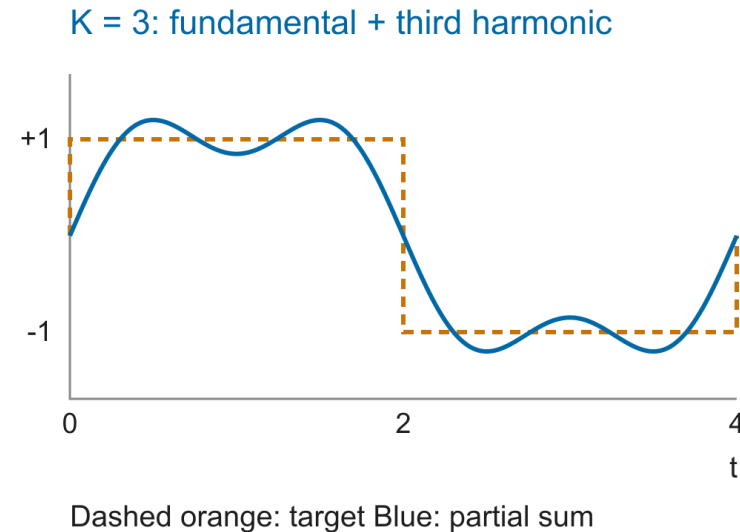
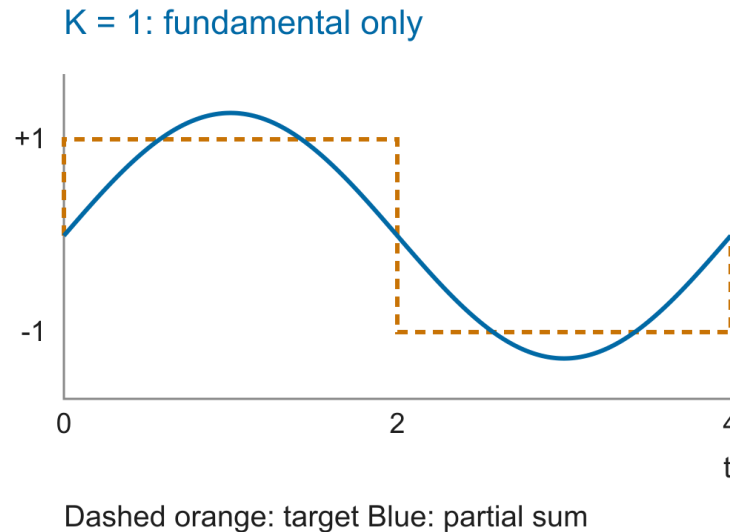
Term	Meaning in this example
$p_1 = 4, p_k = p_1/k$	Fundamental period and period of harmonic k
$4/(\pi k)$	Amplitude of the k th sine component
$k = 1, 3, \dots, K$	Odd harmonics of the symmetric square wave
$e'_K(t)$	Approximation retaining frequencies up to K/p_1

We increase **K from 1 through 3, 5, 7, 9 to 11**: more bandwidth gives sharper edges, but overshoot remains. An ideal square wave is not band-limited.

- Preserving sharp transitions requires preserving higher frequencies. Reconstruction is therefore a bandwidth question.

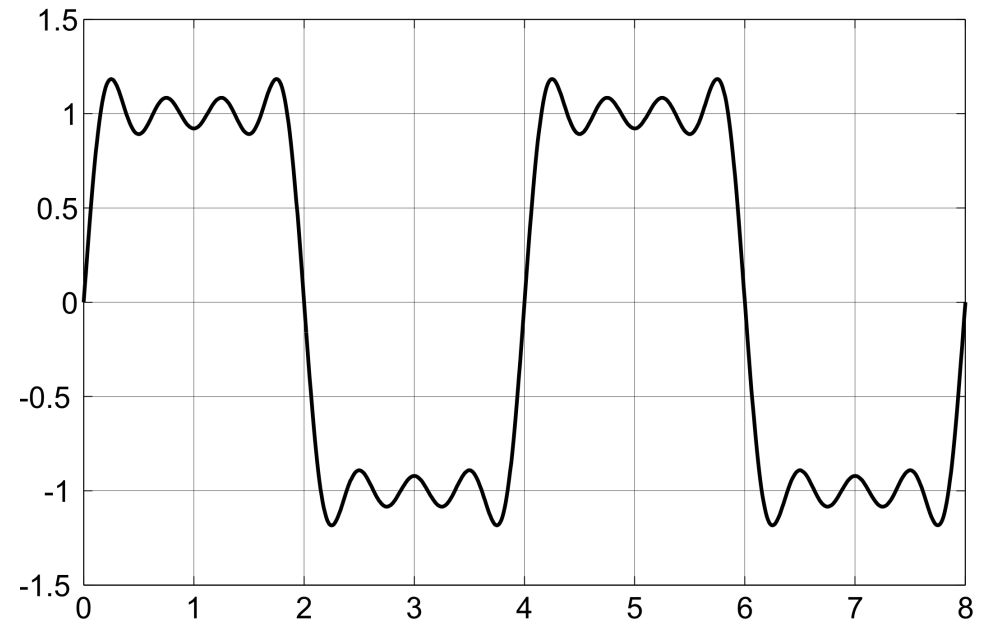
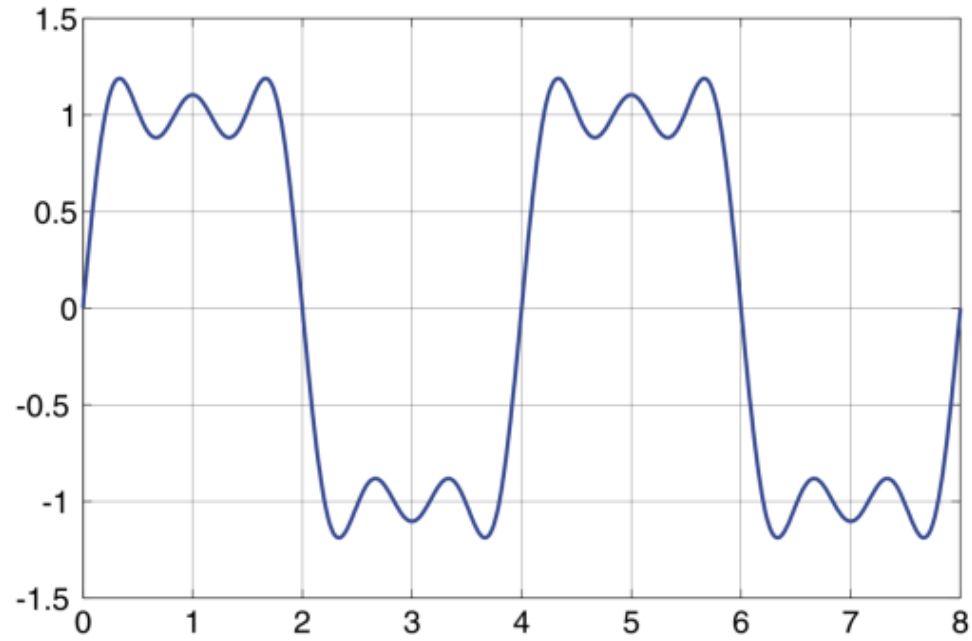
Fourier Approximation: $K=1$ and $K=3$

The **target is the square wave** in both plots. We build it as a sum of sine waves, beginning with the fundamental.



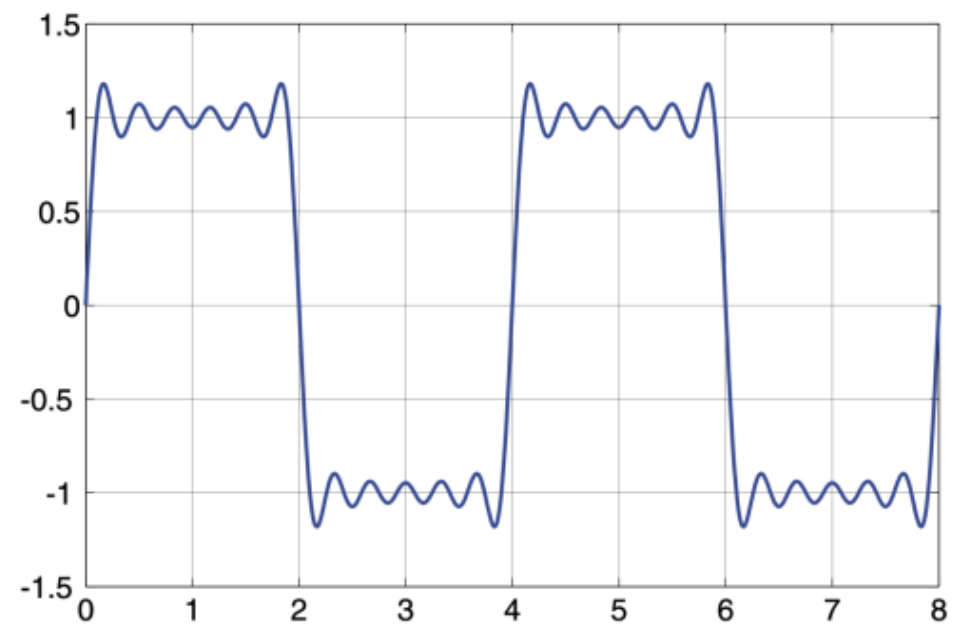
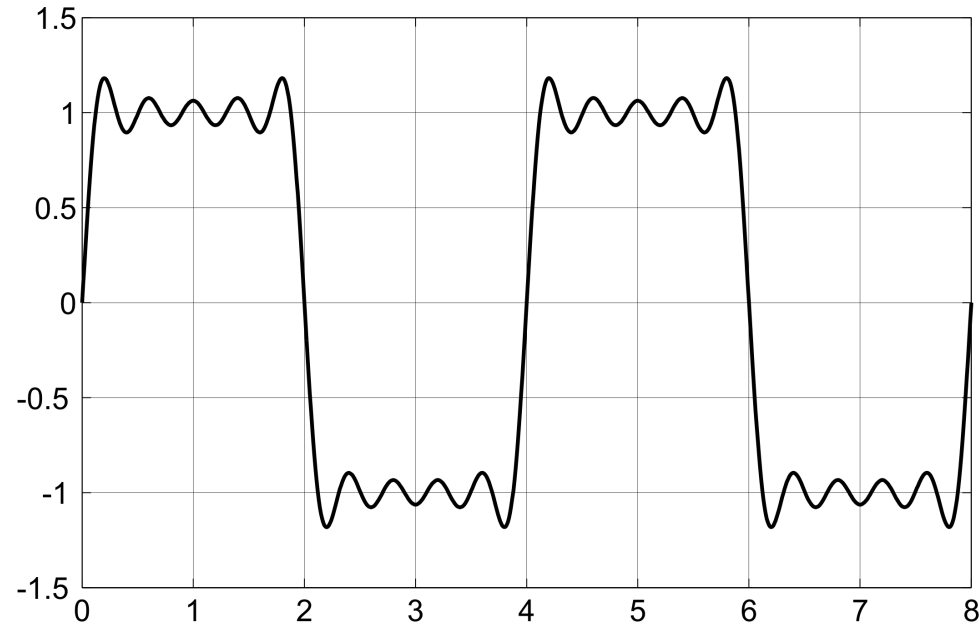
- Left: only $k = 1$. Right: add $k = 3$ with one third of the amplitude. The sum starts to resemble the same target square wave.

Fourier Approximation: $K=5$ and $K=7$



Chen / Marwedel 2024, Inputs, PDF p.20; original PowerPoint image.

Fourier Approximation: $K=9$ and $K=11$



Chen / Marwedel 2024, Inputs, PDF p.21; original PowerPoint image.

Linear Transformations

Why Fourier helps: a **linear operation** acts on each frequency component separately; its results can then be added.

Additivity

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

Also $Tr(ae) = aTr(e)$: additivity and homogeneity define linearity.

Consequence

For ideal sampling $S(e)[n] = e(nT_s)$:

$$S(\sum_k a_k e_k) = \sum_k a_k S(e_k).$$

So first ask whether one sine wave can be recovered.

- **Next: different sine components can produce the same samples. The sampling theorem states when this ambiguity is excluded.**

Tr : transformation; e, e_1, e_2, e_k : input signals; a, a_k : scalar weights; k : component index. S : ideal sampling operator; n : integer sample index; T_s : sampling period, so $S(e)[n] = e(nT_s)$.

Aliasing

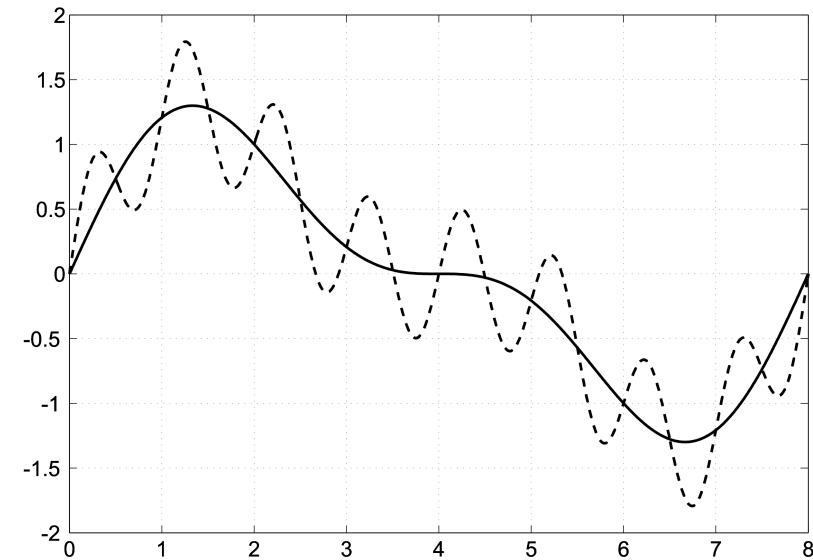
The obstacle to reconstruction: **different continuous signals can give identical samples.**

$$e_3(t) = \sin(2\pi t/8) + 0.5 \sin(2\pi t/4)$$

$$e_4(t) = e_3(t) + 0.5 \sin(2\pi t)$$

At integer times $t = n$, $\sin(2\pi n) = 0$: both continuous signals give the same samples.

Plot: solid e_3 , dashed e_4 ; the integer times are the sampling instants.



- The added 1 Hz component disappears at $f_s = 1$ Hz. Without a bandwidth limit, reconstruction is not unique.

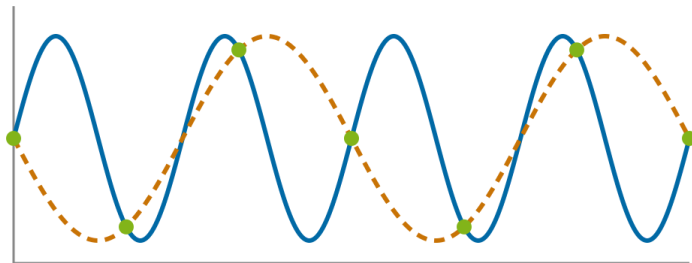
t : time in seconds; n : integer sample index; e_3, e_4 : continuous-time signals. $f_s = 1/T_s$: sampling rate; T_s : sampling period. f_{max} : highest input frequency (1 Hz for e_4 here); $f_s/2$: Nyquist limit.

Sampling Theorem: Exact Recovery Is Possible

$$f_s > 2f_{max}$$

A signal with no components above f_{max} can be reconstructed exactly from its ideal, uniformly spaced samples. $f_s/2$ is the Nyquist limit.

$f_s = 1.5 f_0$: ambiguous

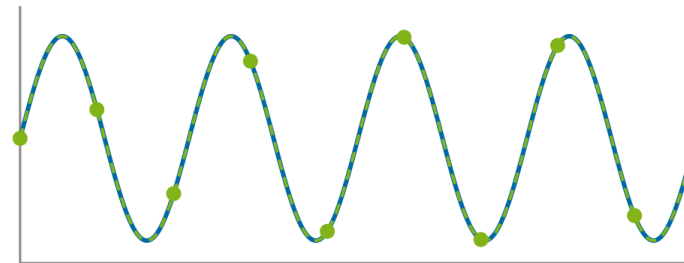


Time in periods of the original sine

Same points also fit a $0.5 f_0$ sine

Blue: original Green dots: samples Dashed: alias / ideal reconstruction

$f_s = 2.2 f_0$: recoverable



Time in periods of the original sine

Ideal reconstruction matches the original

f_s : sampling rate; f_{max} : highest input frequency; f_0 : sine frequency in the plots. Ideal reconstruction uses the complete sample sequence, not straight-line interpolation. Here $f_{max} = f_0$; practical filters require margin.

Anti-Aliasing Filters

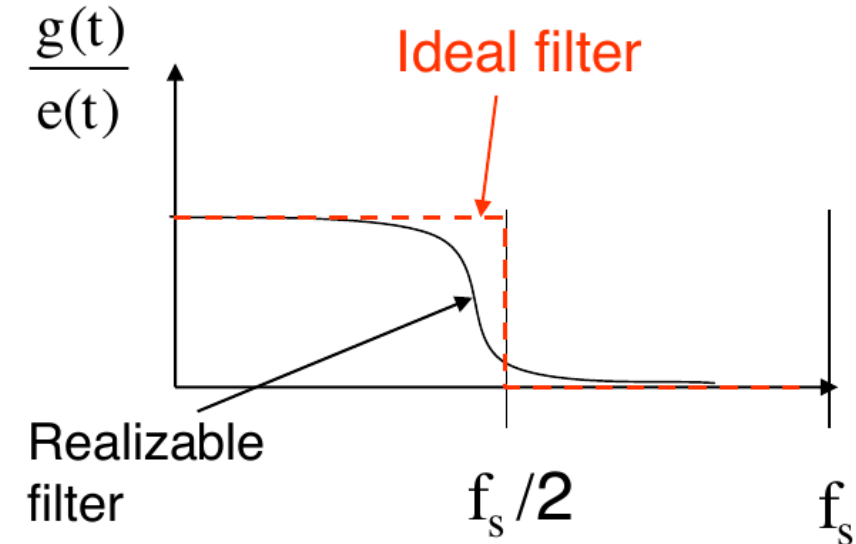
The analog filter precedes sample-and-hold:

$$e(t) \rightarrow g(t) \rightarrow h[n].$$

- **Ideal:** remove frequencies at and above $f_s/2$.
- **Realizable:** finite transition band and imperfect attenuation require margin below $f_s/2$.
- A later digital filter cannot undo frequencies already aliased.

Without the analog filter, components above $f_s/2$ **violate the theorem's bandwidth assumption** and fold into the useful band.

► **Sensor bandwidth, filter response, and sampling rate must be designed together.**



Aliasing in Images

The same theorem applies in space: pixels are samples. Fine structures above half the spatial sampling rate violate the bandwidth condition and appear as false coarse patterns.



- ▶ **Left: original. Right: subsampled without filtering. Camera and rendering pipelines also need anti-aliasing.**

Chen / Marwedel 2024, Inputs p.26.

A/D Conversion: From Voltage to Codes

Sensors

physical effects

Signals

time and bandwidth

A/D Conversion

values and codes

Processing

code and arithmetic

Energy

power and energy

- Sampling chooses when to observe. A/D conversion now chooses which numerical values we can represent.

A/D Conversion and Resolution

An ADC maps an analog input voltage to a discrete output code.

$$Q = \frac{V_{FSR}}{n}, \quad n = 2^b$$

- V_{FSR} : full-scale voltage range; n : voltage intervals; b : output bits.
 - Q : voltage step between neighboring codes. Example: $5 \text{ V} / 256 = 19.53 \text{ mV}$.
 - **Design choices:** resolution, conversion latency / throughput, area, power, and cost.
- Time discretization and value discretization solve different problems.

Successive Approximation

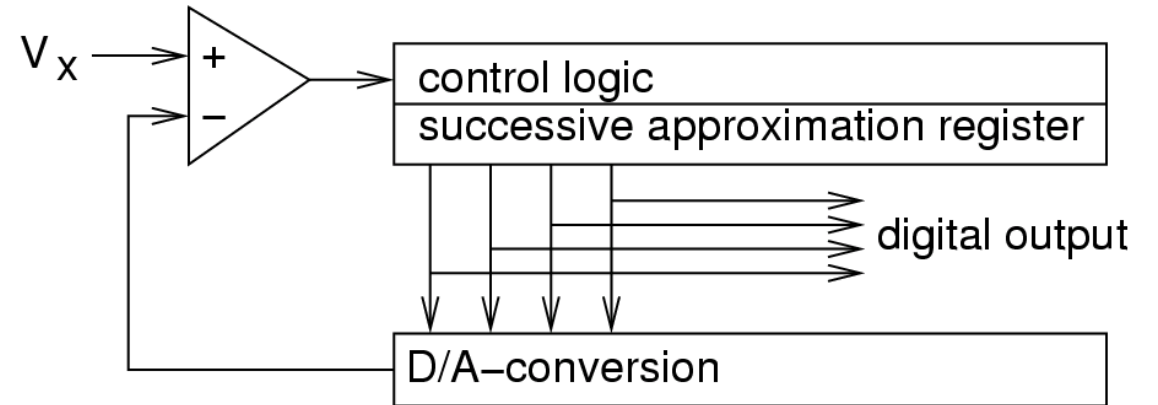
SAR is a common microcontroller ADC

architecture: one comparator is reused for a binary search, from MSB to LSB.

1. Set the next trial bit; the DAC generates a reference voltage.
2. Compare the reference with the held input.
3. Keep or clear the bit, then continue.

For $n = 2^b$ levels: $b = \log_2 n$ rounds. The input must remain held while the DAC and comparator settle at each step.

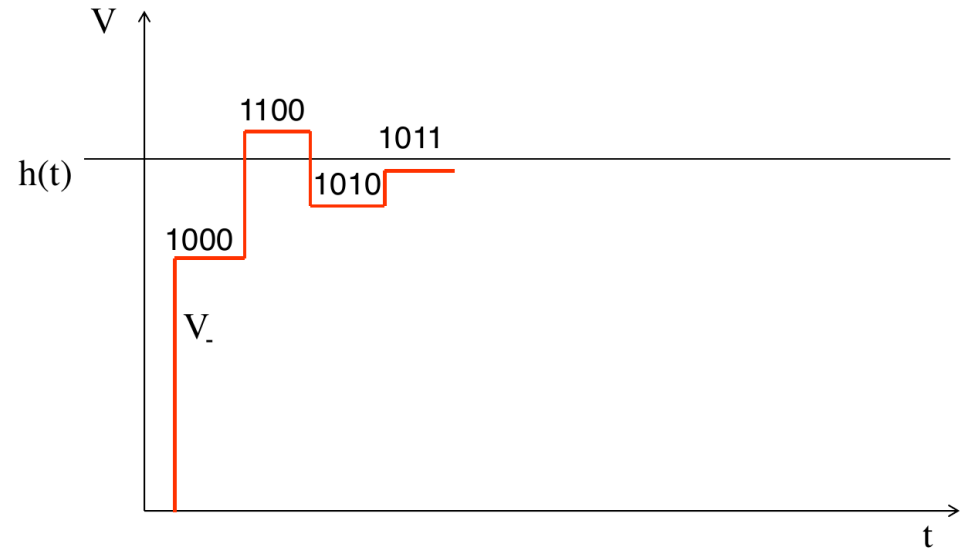
► Start with the embedded default: moderate speed, useful resolution, and low power. Flash trades much more hardware for faster conversion.



Successive Approximation Example

Four-bit example: $V_{in} = 0.70V_{FSR}$; trial voltage $code/16 \cdot V_{FSR}$.

Trial	Full scale	Decision
1000	0.5000	keep bit 3
1100	0.7500	clear bit 2
1010	0.6250	keep bit 1
1011	0.6875	keep bit 0

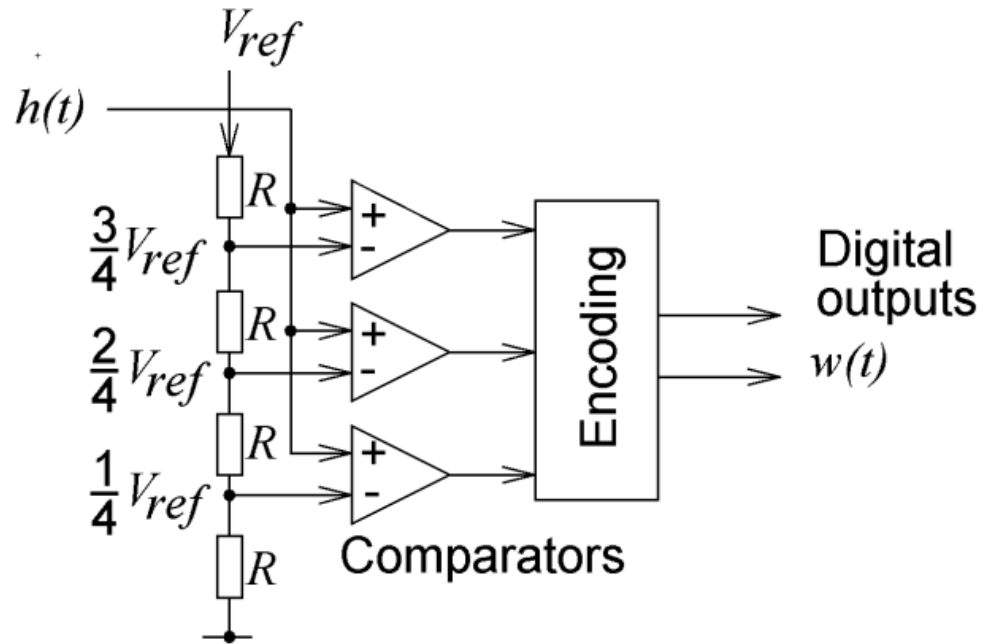


► **Four sequential comparisons resolve the interval.**

Chen / Marwedel 2024, Inputs p.33; native worked example uses the same decision sequence.

Flash A/D Converter

Can we replace SAR's **sequential decisions** with **parallel comparisons**?



Input interval	Comparator pattern	Encoded output
$0 \leq V_{in} < V_1$	none active	00
$V_1 \leq V_{in} < V_2$	lowest active	01
$V_2 \leq V_{in} < V_3$	two active	10
$V_3 \leq V_{in}$	three active	11

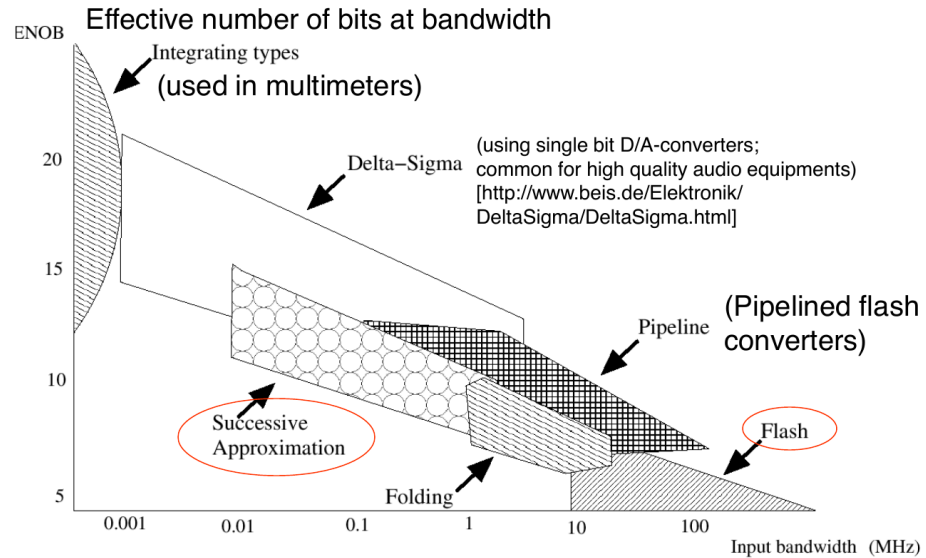
- One parallel threshold comparison, then encoding: $2^b - 1$ comparators. Shorter conversion, but more area and power; acquisition and input settling still take time.

ADC Application Areas

Converter family	Strength	Typical application area
SAR	sequential binary search; balanced speed and power	microcontrollers, control, sensing
Flash	parallel comparisons; very fast, many comparators	high-speed acquisition
Pipeline	stages overlap successive samples; high throughput	video, RF, oscilloscopes
Delta-sigma	oversampled feedback loop; noise shaping and digital filtering	audio, precision measurement
Integrating / dual-slope	average voltage over time; slow, strong noise rejection	precision voltmeters and multimeters

Delta-sigma: A coarse quantizer runs in a fast feedback loop that pushes quantization noise out of the signal band. A digital low-pass filter and downsampling produce the multi-bit output, at the cost of bandwidth and filter latency.

ADC Bandwidth and Effective Resolution

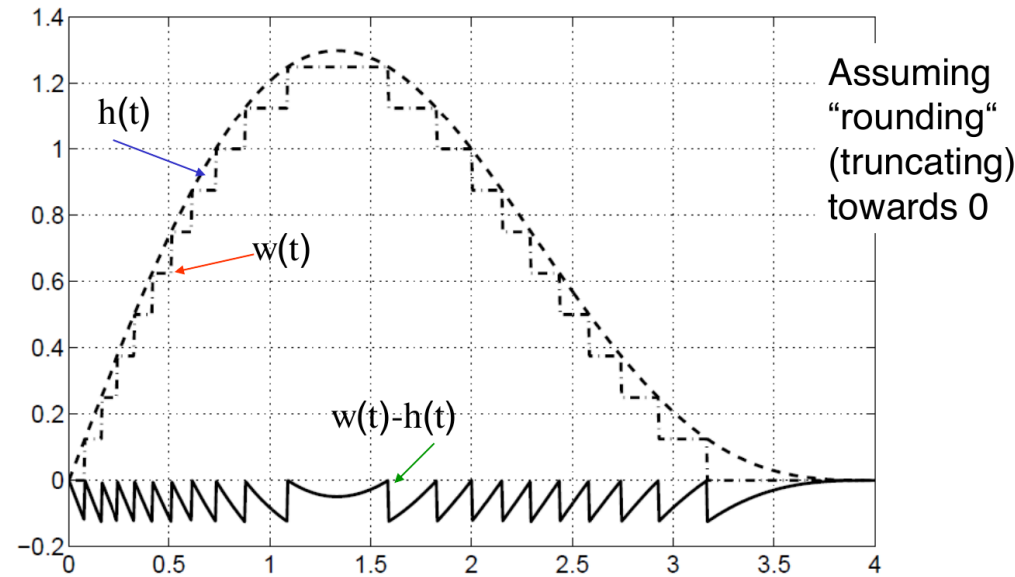


Pipeline: coarse conversion → DAC subtraction → amplified residue to the next stage. Stages work on **different samples at once**: high throughput, but pipeline latency.

Folding: a nonlinear stage folds the input into a smaller range; decision bits retain which branch was taken. The folded residue is processed further, avoiding a full flash comparator bank.

Application map: Chen / Marwedel, 2024, Hardware Inputs, p.34. ENOB is effective resolution, not merely output-code width.

Quantization Noise

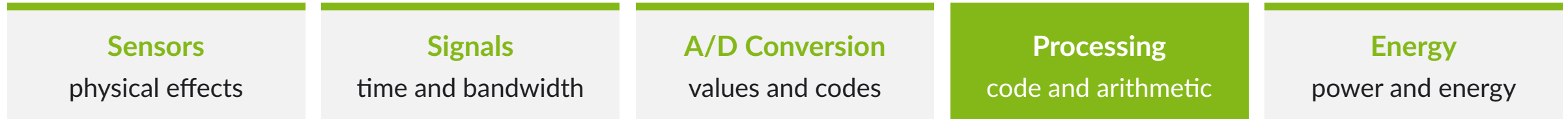


For quantization error $q = w - h$: rounding gives $|q| \leq Q/2$ within range. Truncation toward zero has a sign-dependent bias and $|q| < Q$.

More bits reduce the step size. More speed often costs power, area, or resolution.

Chen Inputs p.35 shows truncation toward zero, not nearest-value rounding.

Processing: Efficient and Predictable Computation



- The input is now digital. Processing must deliver the required result within memory, arithmetic, and timing constraints.

Processing Requirements

Requirement	What to measure	Design consequence
Code size	bytes, memory accesses	fit memory; reduce fetch cost
Run time	cycles per input sample	exploit repeated operations
Energy / power	joules per result / watts	battery lifetime / cooling
Real time	worst-case response time	meet the physical deadline

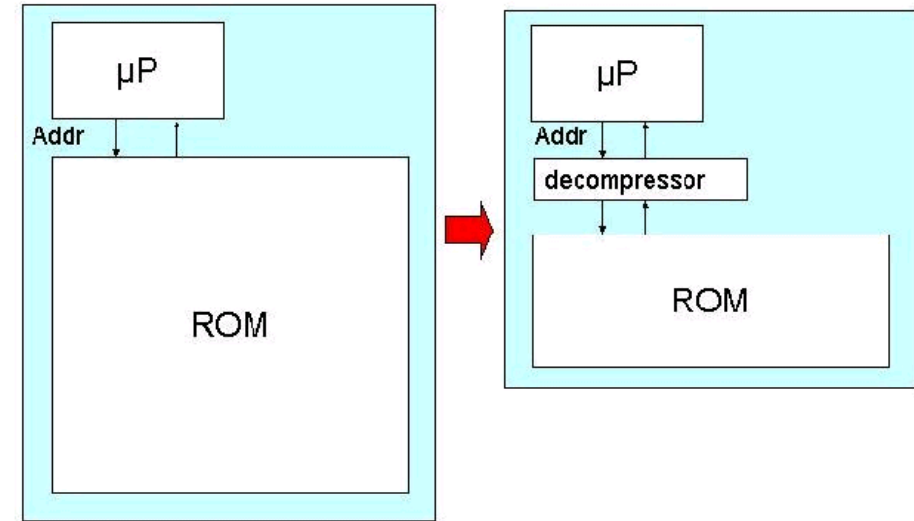
Weight, packaging, and production cost further constrain embedded designs.

- For the braking assistant: a fast average is not enough. The result must arrive **before the deadline**, within the available **memory and energy**.

Code-Size Efficiency

Larger code memories raise cost and fetch energy.

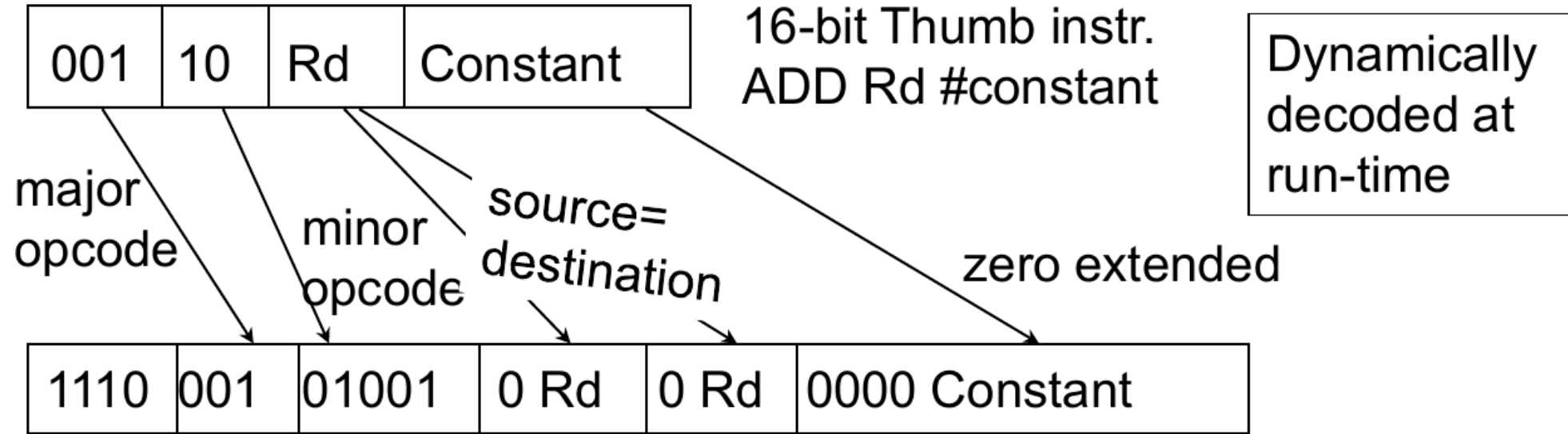
- Store a compressed encoding and decompress on fetch.
- The decoder must be fast, simple, and energy-efficient.
- Compiler, assembler, and hardware must agree on the encoding.



► **Code density changes the memory system as well as the processor.**

Chen Processing p.5.

ARM Thumb as Code-Size Technique



Classic Thumb encodes instructions in 16 bits. An immediate ADD reuses the destination as source; decoding expands opcode/register fields and zero-extends the constant.

Reported for classic Thumb: **65-70% code size**, **130% performance with 8/16-bit memory**, **85% with 32-bit memory**, relative to full-width ARM.

ARM / Gupta via Chen Processing p.6; implementation-dependent results, not a general claim for current ARM cores.

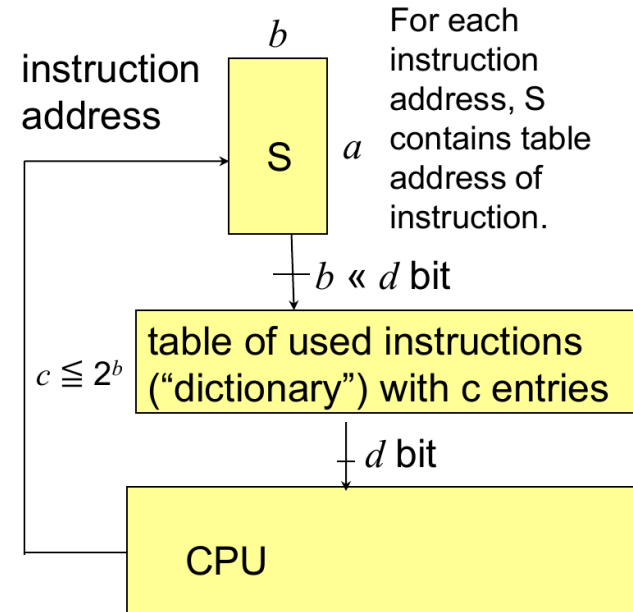
Dictionary-Based Code Compression

Dictionary coders replace frequent instruction sequences by shorter references into a dictionary.

Design variables

- a : number of instruction addresses or references
- b : reference width
- c : number of dictionary entries
- **Addressability:** $c \leq 2^b$ and $b \ll d$
- d : original instruction width
- **Useful if:** $a \cdot b + c \cdot d < a \cdot d$

Dictionary: Two-Level Instruction Fetch



For each instruction address, S contains table address of instruction.

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

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

Hopefully, $axb + \text{small} \times d < axd$.

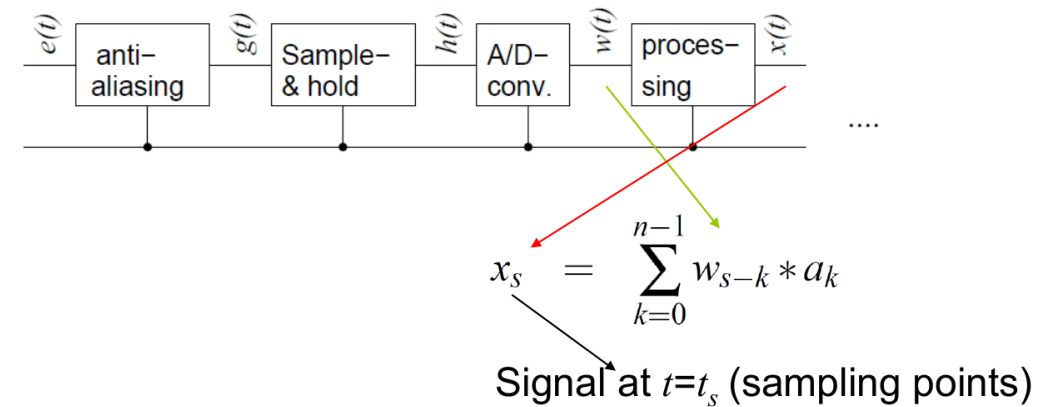
Called nanoprogramming in the Motorola 68000.

DSP Filtering as Architecture Driver

Each input sample triggers N multiply-accumulate operations (MACs). DSP hardware combines multiplication, accumulation, and address updates.

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

At sample rate f_s , the filter needs Nf_s MACs/s. With no buffering, its worst-case processing time must fit within $T_s = 1/f_s$.



w_s : input samples; x_s : output; a_k : coefficients, newest to oldest sample; N : number of taps; s : sample index; k : tap index; f_s : sample rate; T_s : sample period. Source: Marwedel (2021), Chapter 3.3.2, pp.150-152; Chen Processing p.9.

Power and Energy as Platform Budgets

The design quantities from the processing overview become concrete limits: **watts constrain supply and cooling**; **joules constrain how long it can run**.

Design quantity	Factory controller	Electric car	Wireless sensor
Power [W]	supply and cooling capacity	power-converter and thermal limits	radio peaks and supply limits
Energy [J]	energy cost per result	energy available between charges	battery or harvested energy
Duty cycle	active and idle workload phases	driving and parked operation	sampling, transmit, sleep intervals

- **First meet the timing requirement; then choose an operating point and duty cycle within the power and energy budgets. The models follow in the energy section.**

Adapted from Marwedel (2021), Chapter 1.3, Table 1.1, p.13, also used in Chen Processing p.10. The table above states platform constraints rather than reproducing the original checklist.

Real-Time Capability

Shared resources

Caches, buses, and memory controllers introduce interference; unified caches create instruction/data conflicts.

Speculation

Branch prediction and out-of-order execution can make timing harder to bound.

Pipelines

Stalls, flushes, and data-dependent instruction times create execution-time variation.

Interrupts

Asynchronous events can disturb otherwise regular execution.

Refresh and I/O

Memory refreshes and unpredictable multiprocessor communication add timing effects.

Rule of thumb

Use features only when their worst-case impact can be analyzed tightly.

WCET: worst-case execution time. Here we identify hardware effects on that bound; detailed timing analysis follows later. Next: numerical behavior, a separate requirement on the computed result. Marwedel (2021), Chapter 3.3.2, pp.153-154; Chen Processing p.11.

Saturating Arithmetic

Meeting a deadline does not guarantee a meaningful numerical result. Overflow can wrap around or clamp to the representable range: **unsigned four-bit values**, $a = 7$, $b = 9$.

Operation	Intermediate	Four-bit result
Wrap-around sum	$10000_2 = 16$	$0000_2 = 0$
Saturating sum	$10000_2 = 16$	$1111_2 = 15$
Exact average	$16/2$	$1000_2 = 8$
Wrap, then shift	$0000_2/2$	$0000_2 = 0$
Saturate, then shift	$1111_2/2$	$0111_2 = 7$

- Saturation avoids wrap-around jumps; hardware support also avoids a software overflow handler. Exact averaging still requires a wider intermediate.

Addition example: Marwedel (2021), Chapter 3.3.2, Table 3.1, p.153. Averaging extension: Chen (2024), Processing, slide 12. Saturation limits the stored result; it does not make the intermediate sum exact.

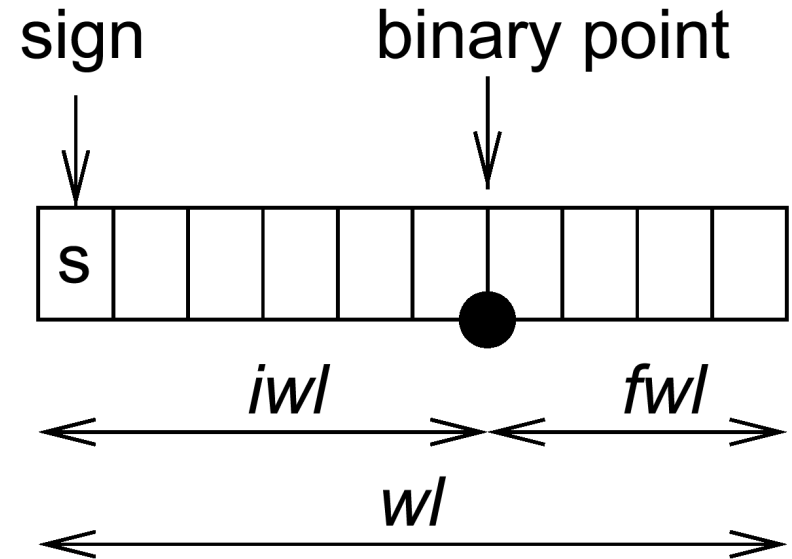
Fixed-Point Arithmetic

With f fractional bits, stored integer X represents $x = X2^{-f}$.

For a signed word, $w = 1 + i + f$: sign, integer, and fractional fields.

In the diagram, $iwl = 1 + i$ includes the sign bit, and $fwl = f$.

- Multiply: $Z = (X \cdot Y) \gg f$.
- Divide: $Z = (X \ll f)/Y$.
- Use wide intermediates; check overflow, rounding, and division by zero.



► Shifts restore the binary point; they cannot recover already lost information.

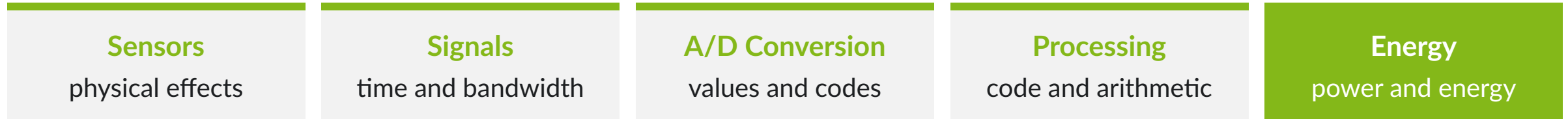
Fixed-Point Scaling Example

With $f = 4$: $1.5 \leftrightarrow X = 24$ and $0.5 \leftrightarrow Y = 8$.

Stage	Integer value	Scale / represented value
Multiply	$24 \cdot 8 = 192$	2^{-8} : $192/256 = 0.75$
Restore format	$192 \gg 4 = 12$	2^{-4} : $12/16 = 0.75$
Divide	$(24 \ll 4)/8 = 48$	2^{-4} : $48/16 = 3.0$

- Shifts restore the binary point; they do not repair information already lost through overflow or truncation.

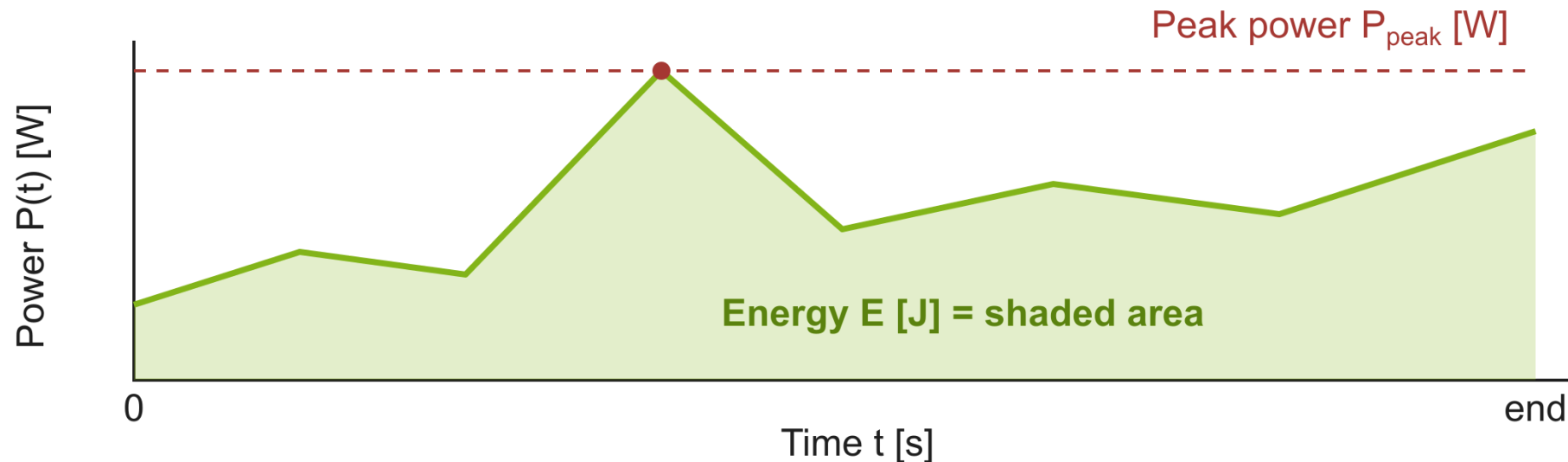
Energy: Completing the Hardware Budget



- An efficient computation is only useful if the platform can supply its energy and remove its heat.

Energy and Power

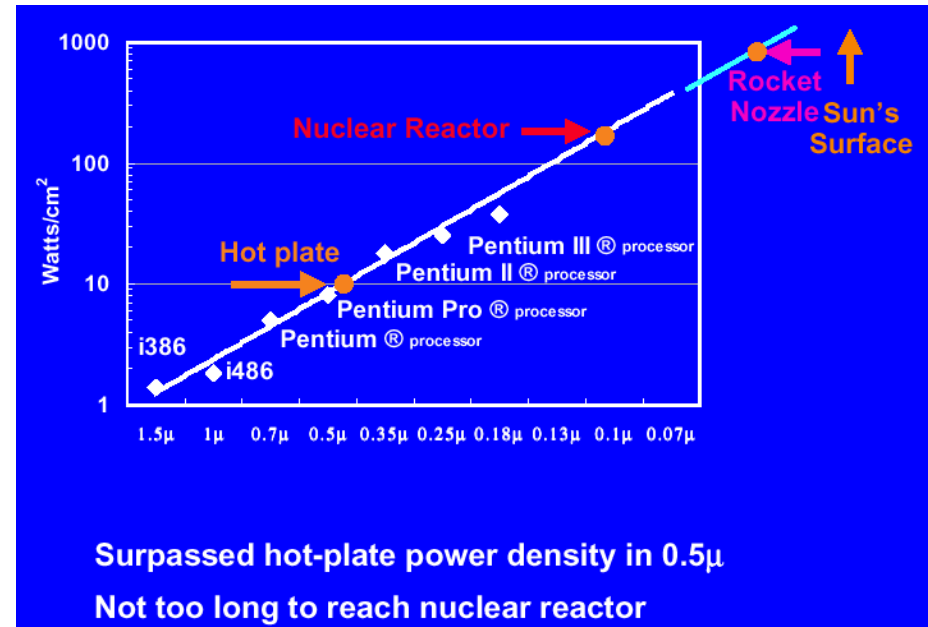
Power $P(t)$ is measured in watts. Energy $E = \int P(t) dt$, measured in joules, is the area under the power curve over the execution time t .



- Peak power P_{peak} affects supply sizing; energy E affects battery life. Lower power P can still mean higher energy E if runtime increases enough.

Power Density and Cooling Pressure

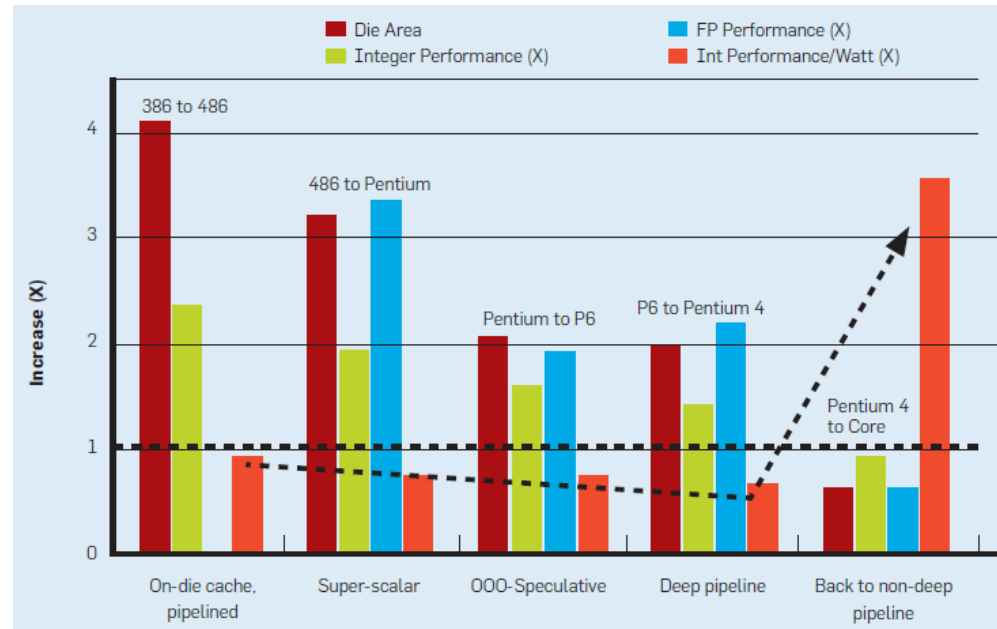
Power density P/A measures heat generation per chip area A . Cooling limits sustained power P , especially in sealed or passively cooled products.



Intel / Pollack via Chen Processing PDF p.16. Logarithmic vertical axis: P/A in W/cm^2 . The graphic illustrates a scaling concern, not a current forecast. Prescott (90 nm, 2004): about 90 W/cm^2 .

Simpler Cores and Cooling

More complex cores can increase chip area and power P faster than performance. The chart compares relative gains in area, performance, and performance per watt.

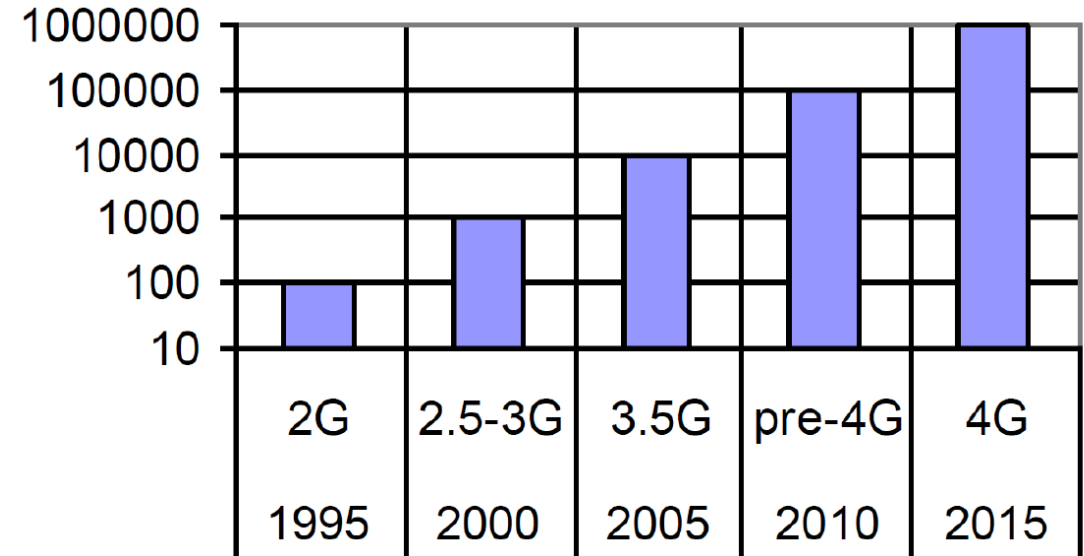


Borkar / Chien, CACM 2011; Chen Processing p.20. Vertical axis: multiplicative increase (1 = unchanged); each group compares successive processor generations. More elaborate cooling adds cost, volume, and energy demand.

Mobile Performance and Power

The chart spans 2G (1995) to 4G (2015).

- Richer codecs, graphics, and radio protocols increase compute and memory demands.
- Workload rises by orders of magnitude on the logarithmic MOPS scale.
- Sustained power remains limited by enclosure, cooling, and battery.

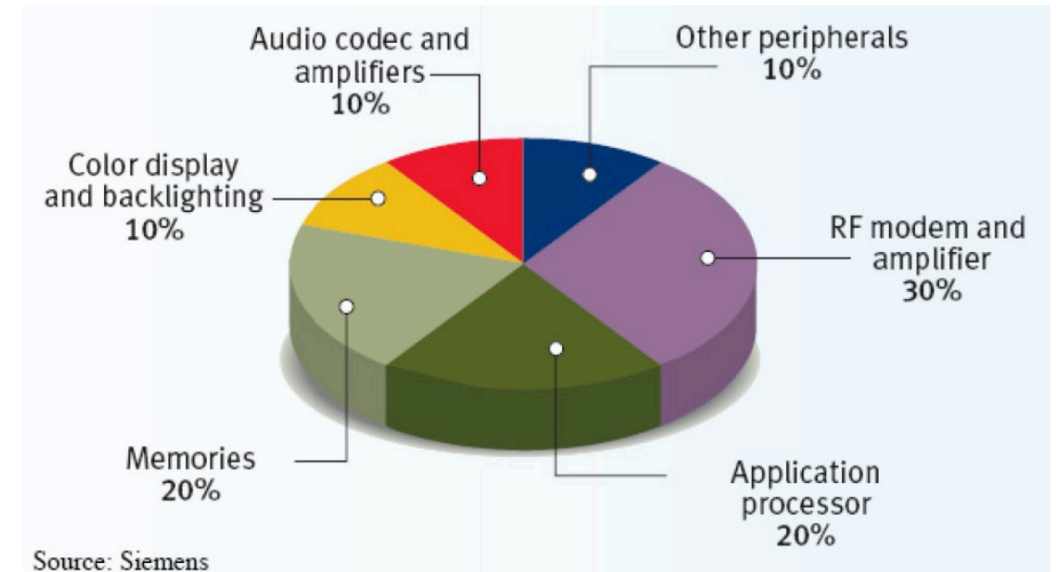


► **Useful performance per joule matters alongside responsiveness.**

Van Berkel, DATE 2009; later years in this source are projections.

Mobile Power Breakdown (2005)

Component	Share
RF modem / amplifier	30%
Application processor	20%
Memories	20%
Display / backlight	10%
Audio codec / amplifier	10%
Other peripherals	10%



► **Optimizing only the CPU leaves major platform consumers untouched.**

Siemens / Vargas 2005; not a current phone profile.

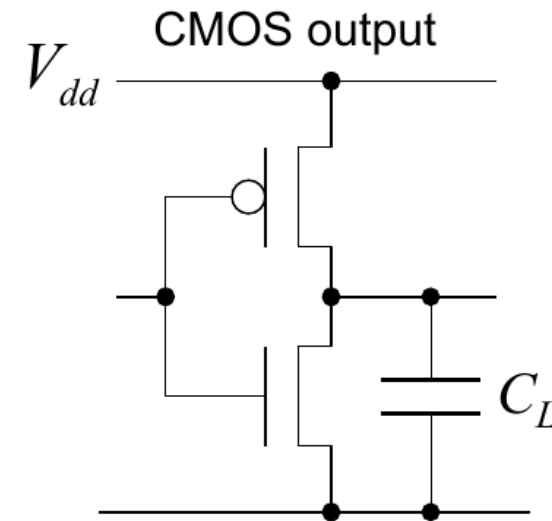
CMOS Power

$$P_{dyn} = \alpha C_L V_{dd}^2 f$$

- **Load C_L :** more capacitance must be charged and discharged.
- **Voltage V_{dd}^2 :** switching energy grows quadratically with voltage.
- **Activity α and clock f :** more switching events per second increase dynamic power P_{dyn} .

Leakage adds static power P_{static} even without switching:

$$P_{total} = P_{dyn} + P_{static}, \quad P_{static} \approx V_{dd} I_{leak}$$



► **Lowering frequency alone does not remove leakage; longer runtime can increase static energy.**

P_{dyn} , P_{static} , P_{total} : dynamic, static, and total power [W]; α : switching activity; C_L : load capacitance [F]; V_{dd} : supply voltage [V]; f : clock frequency [Hz]; I_{leak} : leakage current [A]. Source: Chen Processing p.23; Marwedel (2021), Chapter 3.3.2.

CMOS Delay and DVS

Voltage scaling reduces dynamic power quadratically, but it also slows the circuit.

Dynamic power

$$P_{dyn} = \alpha C_L V_{dd}^2 f$$

CMOS delay model

$\tau = k \frac{C_L V_{dd}}{(V_{dd} - V_t)^2}$, where V_t is threshold voltage and $V_{dd} > V_t$.

- DVS is a trade between lower voltage, lower power, slower switching, and potentially longer execution time.

P_{dyn} : dynamic power [W]; α : switching activity; C_L : load capacitance [F]; V_{dd} : supply voltage [V]; f : clock frequency [Hz]; τ : circuit delay [s]; V_t : threshold voltage [V]; k : technology-dependent proportionality constant. Source: Marwedel (2021), Chapter 3.3.2, Eqs.3.14-3.15.

DVFS: Idealized Scaling

Assume fixed cycle count, negligible leakage, and clock speed approximately proportional to voltage. Scale voltage and frequency by r , with $0 < r < 1$.

$$P'_{dyn} = r^3 P_{dyn}, \quad t' = t/r, \quad E'_{dyn} = r^2 E_{dyn}$$

For $r = 1/2$: dynamic power becomes $1/8$, runtime doubles, and dynamic energy becomes $1/4$.

- The cubic power claim is conditional. The CMOS delay relation is nonlinear near threshold, and static energy grows with runtime.

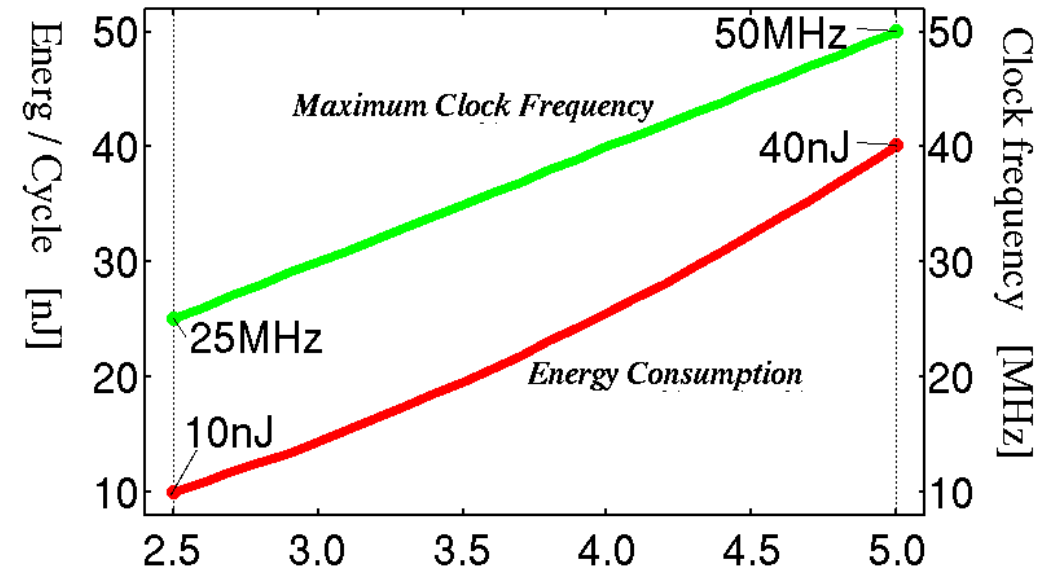
Voltage/Frequency Scaling Example

Yasuura's example relates voltage to maximum clock speed and energy per cycle.

Supply	Maximum speed	Energy/cycle
2.5 V	25 MHz	10 nJ
5.0 V	50 MHz	40 nJ

For equal cycle count, halving speed permits about one quarter of the dynamic energy in this example.

- Use lower voltage only when timing permits; leakage and transition costs require separate accounting.



Yasuura 2000 via Chen Processing p.26.

Abstract Power Model

CMOS core model

$$P(s) = P_{dynamic}(s) + P_{static}$$

Example parameters

$$\alpha = 1.76 \text{ W/GHz}^3, \gamma = 3, \beta = 0.5 \text{ W}.$$

Frequency approximation

$$P(s) = \alpha s^\gamma + \beta, \text{ with } \gamma > 1 \text{ and static power } \beta.$$

Model boundary

s is clock frequency. Memory and peripherals require additional terms in a whole-system model.

With s in GHz, $P(0.8) \approx 1.40 \text{ W}$ and $P(1.6) \approx 7.71 \text{ W}$. The model coefficient α here is distinct from switching activity in the CMOS equation.

Power Model: Frequency Dependence

CMOS-core Power Model

$$P(s) = P_{\text{dynamic}}(s) + P_{\text{static}}$$

Considering that:

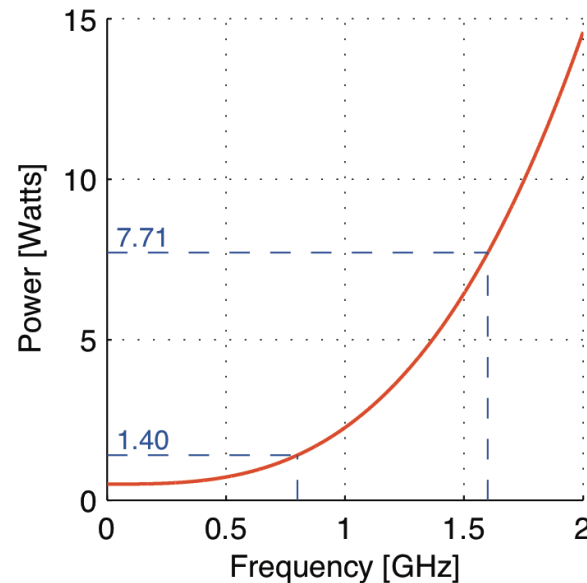
$$P_{\text{dynamic}}(s) = C_{\text{eff}} V_{dd}^2 s$$

$$s \propto \frac{(V_{dd} - V_t)^2}{V_{dd}}$$

We can approximate to:

$$P(s) = \alpha s^\gamma + \beta$$

Figure: $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$ and $\beta = 0.5 \text{ Watts}$



The graph shows core power $P(s)$ as clock frequency s changes:

$$P(s) = \alpha s^\gamma + \beta$$

- Static power $\beta = 0.5 \text{ W}$ remains while the core is powered.
- With $\gamma = 3$, the dynamic term αs^3 grows steeply: voltage must rise to support higher clock speeds.
- Doubling s from 0.8 to 1.6 GHz raises total power P from 1.40 to 7.71 W, not merely by a factor of two.

Abstract Energy Model

For a task requiring Δc cycles at clock frequency s , execution time is $t = \Delta c/s$ and task energy is $E(s) = P(s)t$.

Task energy

$$E(s) = (\alpha s^\gamma + \beta) \frac{\Delta c}{s}$$

Energy-minimizing frequency

$$s_{crit} = \left(\frac{\beta}{(\gamma-1)\alpha} \right)^{1/\gamma}$$

- ▶ Below s_{crit} , longer execution increases static energy more than it saves dynamic energy. For the example parameters, $s_{crit} \approx 0.52$ GHz.

Energy Model: Critical Frequency

Energy Consumption

$$E(s) = (\alpha s^\gamma + \beta) \frac{\Delta c}{s}$$

Critical Frequency:

$$s_{\text{crit}} = \sqrt[\gamma]{\frac{\beta}{(\gamma - 1)\alpha}}$$

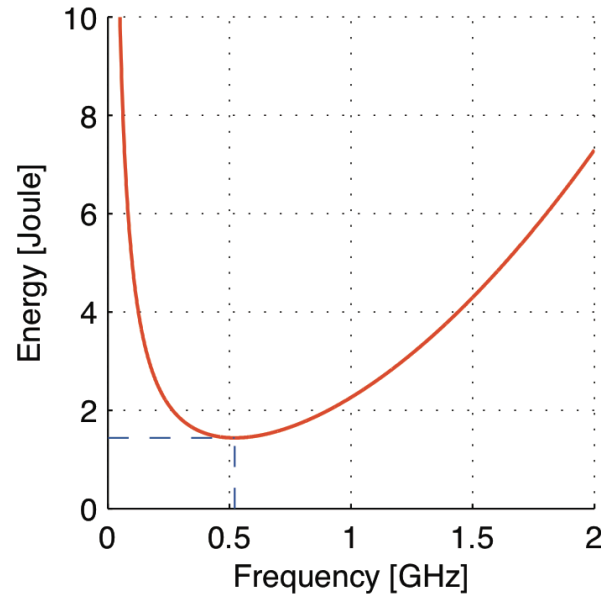


Figure: $\alpha = 1.76 \frac{\text{Watts}}{\text{GHz}^3}$, $\gamma = 3$,
 $\beta = 0.5 \text{ Watts}$ and $\Delta c = 10^9$ cycles

For the same work $\Delta c = 10^9$ cycles, task energy $E(s)$ includes both power $P(s)$ and execution time $t = \Delta c/s$.

- **High s :** the dynamic power term dominates; executing faster costs more energy E .
- **Low s :** longer runtime t increases leakage energy βt .
- The minimum is near $s_{\text{crit}} = 0.52 \text{ GHz}$, with $E \approx 1.44 \text{ J}$. Slower is not always more efficient.

A deadline D also requires $s \geq \Delta c/D$; the energy minimum may be too slow.

Critical Frequency: Derivation

$$E(s) = \Delta c \left(\alpha s^{\gamma-1} + \frac{\beta}{s} \right)$$

$$\frac{dE}{ds} = \Delta c \left((\gamma - 1) \alpha s^{\gamma-2} - \frac{\beta}{s^2} \right) = 0$$

Thus $(\gamma - 1) \alpha s^{\gamma} = \beta$. For the source example $\Delta c = 10^9$ cycles, the optimum is about 0.52 GHz and 1.44 J.

- With a deadline D , feasible frequency also requires $s \geq \Delta c / D$. Choose the energy optimum within the supported, deadline-feasible operating points.

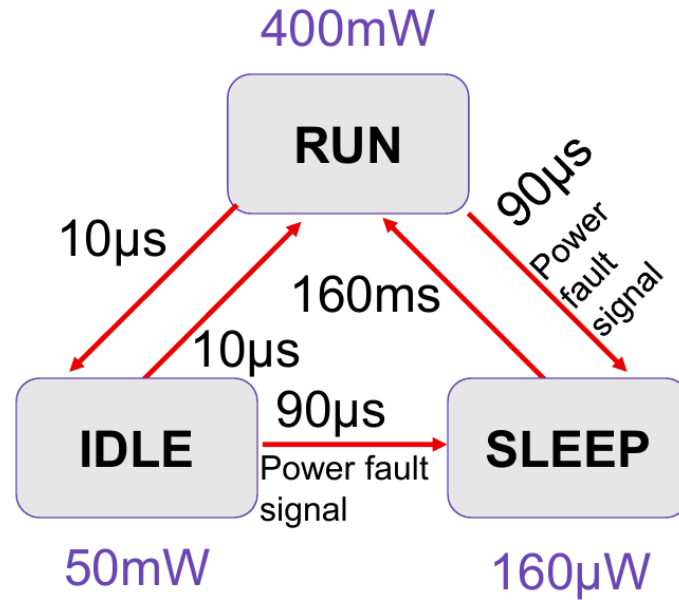
Dynamic Power Management

State	SA-1100 example	Design meaning
RUN	about 400 mW	useful computation
IDLE	about 50 mW	CPU stopped by software, interrupts monitored
SLEEP	about 160 uW	on-chip activity shut down

- Power states form an automaton with transition delays and transition energy, not just a list of numbers.

For an idle interval, compare $E_{transition}$ against the saved energy $(P_{idle} - P_{sleep})t$. Sleep pays off only beyond the break-even time and within the wakeup deadline.

SA-1100: Power-State Automaton



Chen / Marwedel 2024, Processing, PDF p.29.

Low-Voltage Parallel Operation

Lower voltage and parallel operation can consume less energy than high-voltage sequential operation, if the workload exposes enough parallelism.

Idealized assumptions

$$P \sim V_{DD}^2 s, \text{ with speed } s \sim V_{DD}$$

Per parallel unit $P'_{unit} = P/M^3$; multiplying by M gives total $P' = P/M^2$. Runtime remains $t' = t$ under ideal workload splitting.

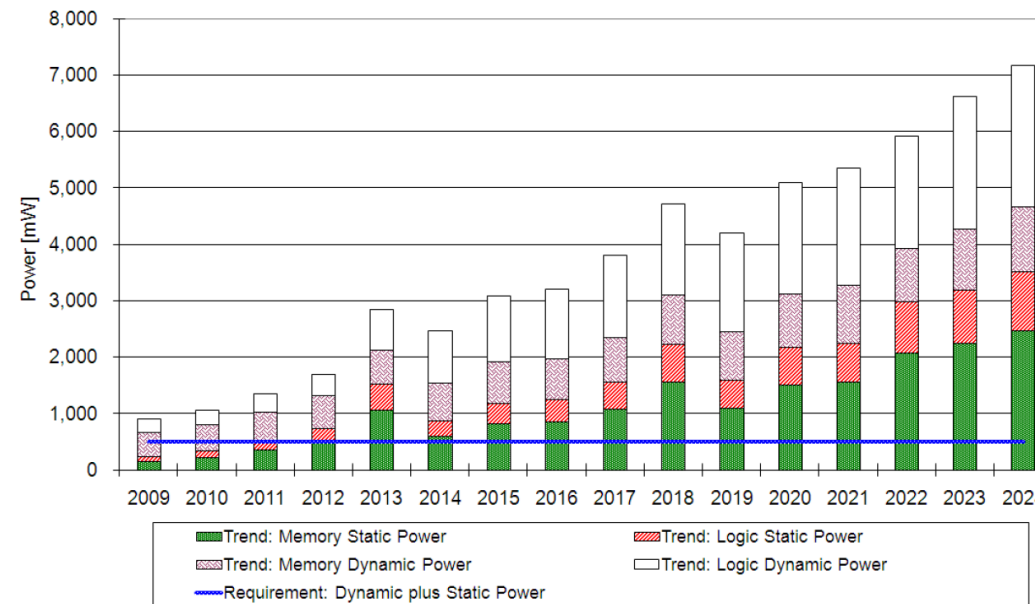
With M parallel units

$$s' = s/M, V'_{DD} = V_{DD}/M, P' = P/M^2, \\ E' = E/M^2$$

- This assumes equal throughput, perfect splitting, negligible leakage/communication costs, and a linear voltage/speed relation. It is a rough model, not a universal scaling law.

Mobile Energy Budget

The ITRS 2010 projection compares component power P with a **0.5-1 W** budget for small mobiles; larger mobiles were discussed around **7 W**.



ITRS 2010 / Chen Processing p.31. Projection from 2010; processor, memory, radios, display and platform budget must be considered together.

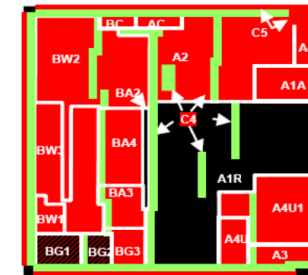
Heterogeneous Processing and Power Gating

A heterogeneous SoC can combine specialized processing and independently powered domains.

In Chen's W-CDMA example:

- Control and W-CDMA baseband are ON.
- GSM is mode-dependent; the application system domain is ON, its real-time domain OFF.
- Reported leakage: **407 microamperes**, room temperature, **1.2 V**.

(2)Telephony (W-CDMA)



■ Power on
■ Power off

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

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

MPSoC '07

Everywhere you imagine. **RENESAS**

► Power only the domains needed by the active use case.

Hattori / Renesas 2007; Chen Processing p.32.

big.LITTLE: Cortex-A15 and Cortex-A7 Pipelines

A complex Cortex-A15 provides higher peak performance; a simpler Cortex-A7 targets efficient execution at modest demand.

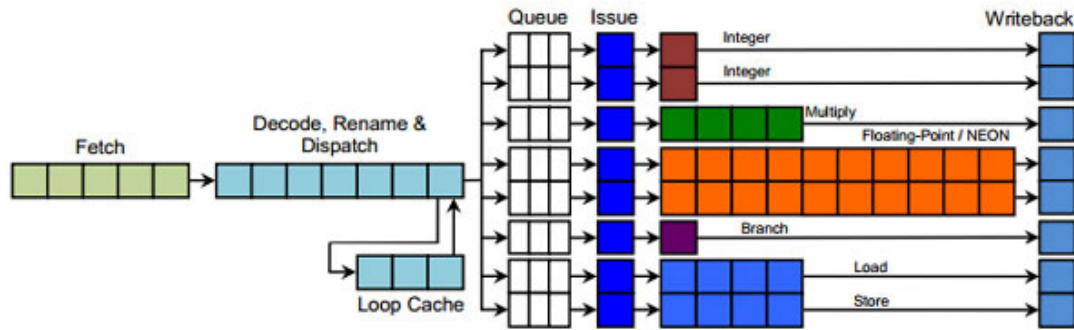


Figure 2 Cortex-A15 Pipeline

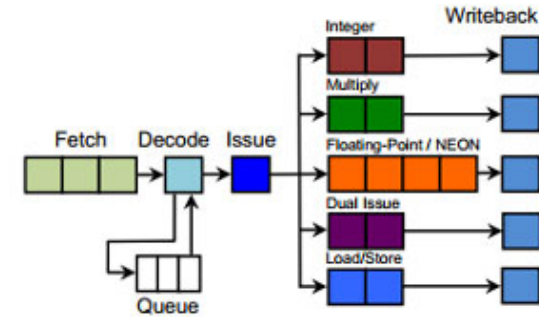
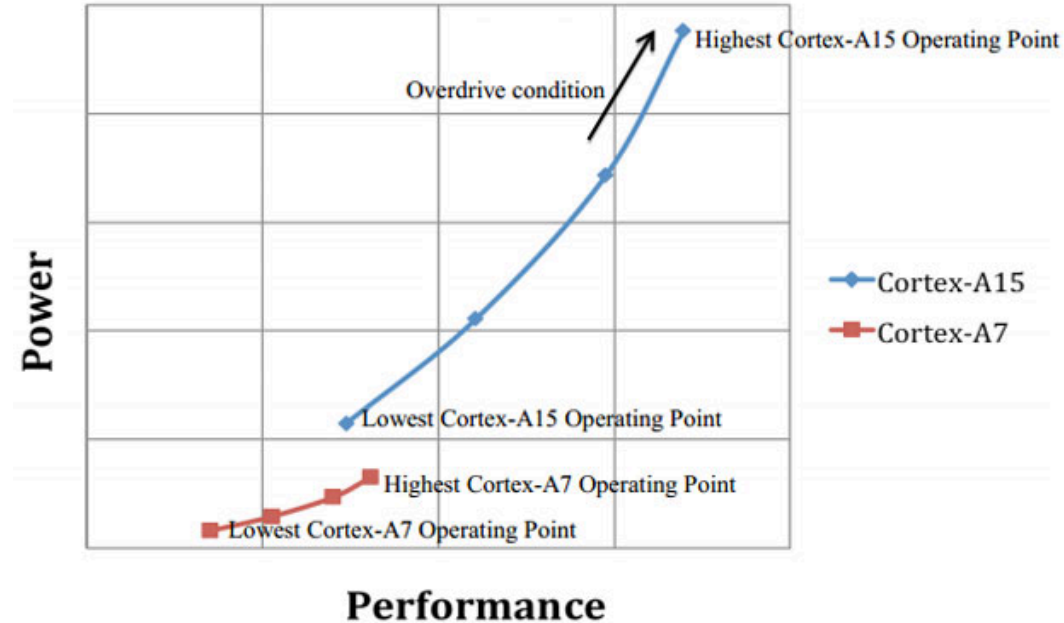


Figure 1 Cortex-A7 Pipeline

Left: Cortex-A15. Right: Cortex-A7.

ARM 2013; Chen Processing, PDF p.33.

big.LITTLE: Power and Performance



ARM 2013; Chen Processing, PDF p.33.

- The scheduler must match workload phases to suitable cores while respecting latency and thermals. The example platform is Samsung Galaxy S4.

Chapter 2 Summary

Interface

Sensors, sample-and-hold circuits, filters, and A/D converters form the input boundary. Output conversion and actuators follow in a later chapter.

Processing

Processor choice balances flexibility, efficiency, timing predictability, cost, and updateability.

Signals

Sampling handles time, quantization handles values, and aliasing appears when the sampling design ignores frequency content.

Energy

Power, energy, heat, voltage, frequency, and sleep states must be designed together.

Self Check

Explain

Why does an anti-aliasing filter sit before the sample-and-hold circuit?

Compare

When would you choose flash ADC over successive approximation?

Apply

Name three hardware features that make worst-case execution time harder to predict.

Calculate

If an ADC has $V_{FSR} = 5V$ and 256 intervals, what is Q ?

Reason

Why can a lower-power design still consume more energy?

Design

Sketch one embedded product and identify its sensors, processing unit, actuator, energy source, and heat path.

References

Reading in Marwedel

Fourth edition (2021)

- Hardware loop → Chapter 3.1
- Sensors → Chapter 3.2.1
- Sampling; sample-and-hold → Chapter 3.2.2
- Fourier and aliasing → Chapter 3.2.3
- Sampling theorem → Chapter 3.6.2
- ADCs and quantization → Chapter 3.2.4
- Code density, DSP, arithmetic → Chapter 3.3.2
- CMOS power; multicore efficiency → Chapters 3.3.2, 3.7.3
- Energy/power models (further reading) → Chapter 5.4
- DVFS; DPM → Chapters 7.4.1; 7.4.2

Bibliography

1. P. Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. 4th ed., Springer, 2021. [Open-access book](#).
2. A. P. Chandrakasan, S. Sheng, R. W. Brodersen. "Low-Power CMOS Digital Design." *IEEE Journal of Solid-State Circuits* 27(4), 1992.
3. W. Kester. *ADC Architectures VI: Folding ADCs*. Analog Devices, Tutorial MT-025, Rev. A, 2008. [Tutorial](#).

