Embedded System Hardware

Jian-Jia Chen
(Slides are based on Peter Marwedel)
Informatik 12
TU Dortmund
Germany

2018年 10 月 24 日

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

(see lecture 1): "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
  - ...
- Reliability
- ...

\[ \int P \, dt \]
Structure of this course

2: Specification & Modeling
3: ES-hardware
4: System software (RTOS, middleware, …)

Design repository

6: Application mapping
7: Optimization
5: Evaluation & Validation (energy, cost, performance, …)

Design

8: Test

* Could be integrated into loop

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"):

© JJ Chen and P. Marwedel, Informatik 12, 2018
Many examples of such loops

- Heating
- Lights
- Engine control
- Power grids
- …
- Robots
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),
- Photoelectric effects.

Huge amount of sensors designed in recent years.
Rain Sensors

An infrared light is beamed at a 45-degree angle into the windshield from the interior — if the glass is wet, less light makes it back to the sensor, and the wipers turn on.
Charge-coupled devices (CCD) image sensors

Based on charge transfer to next pixel cell
CMOS image sensors

Based on standard production process for CMOS chips, allows integration with other components.
Comparison CCD/CMOS sensors

<table>
<thead>
<tr>
<th>Property</th>
<th>CCD</th>
<th>CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology optimized for</td>
<td>Optics</td>
<td>VLSI technology</td>
</tr>
<tr>
<td>Cost</td>
<td>Higher</td>
<td>Lower</td>
</tr>
<tr>
<td>Smart sensors</td>
<td>No, no logic on chip</td>
<td>Logic elements on chip</td>
</tr>
<tr>
<td>Access</td>
<td>Serial</td>
<td>Random</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Low</td>
<td>Larger</td>
</tr>
<tr>
<td>Video mode</td>
<td>Possibly too slow</td>
<td>ok</td>
</tr>
<tr>
<td>Applications</td>
<td>Situation is changing over the years</td>
<td></td>
</tr>
</tbody>
</table>
Example: Biometrical Sensors

e.g.: Fingerprint sensor

© P. Marwedel, 2010
PAMANO Sensor
Other sensors

- 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.
Discretization

Jian-Jia Chen
(Slides are based on Peter Marwedel)
Informatik 12
TU Dortmund
Germany

These slides use Microsoft clip arts. Microsoft copyright restrictions apply.
Sample-and-hold circuits

Clocked transistor + capacitor;
Capacitor stores sequence values

\[ e(t) \] is a mapping \( \mathbb{R} \to \mathbb{R} \)

\[ h(t) \] is a **sequence** of values or a mapping \( \mathbb{Z} \to \mathbb{R} \)
Do we lose information due to sampling?

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

Approximation of signals by sine and cosine 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)

\[ e'_K(t) = \sum_{k=1,3,5,\ldots}^{K} \frac{4}{\pi k} \sin\left(\frac{2\pi t}{4/k}\right) \]
Approximation of a square wave (3)

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

\[ p_s < \frac{1}{2} p_N \] 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.
Anti-aliasing filter

A filter is needed to remove high frequencies

![Diagram of anti-aliasing filter](image)

Ideal filter

Realizable filter

\[ g(t) \]

\[ e(t) \]

\[ f_s/2 \]

\[ f_s \]

\[ e_4(t) \text{ changed into } e_3(t) \]
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.
Flash A/D converter

Encoding of voltage intervals

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:

$Q = \frac{V_{ref}}{4}$ 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

Effective number of bits at bandwidth

- Integrating types (used in multimeters)
- Delta-Sigma (using single bit D/A-converters; common for high quality audio equipments) [http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html]
- Pipeline (Pipelined flash converters)
- Successive Approximation
- Folding

[Gielen et al., DAC 2003]

(http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html)
Quantization Noise

Assuming “rounding” (truncating) towards 0

\[ h(t) \]

\[ w(t) \]

\[ w(t) - h(t) \]
Summary

Hardware in a loop

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