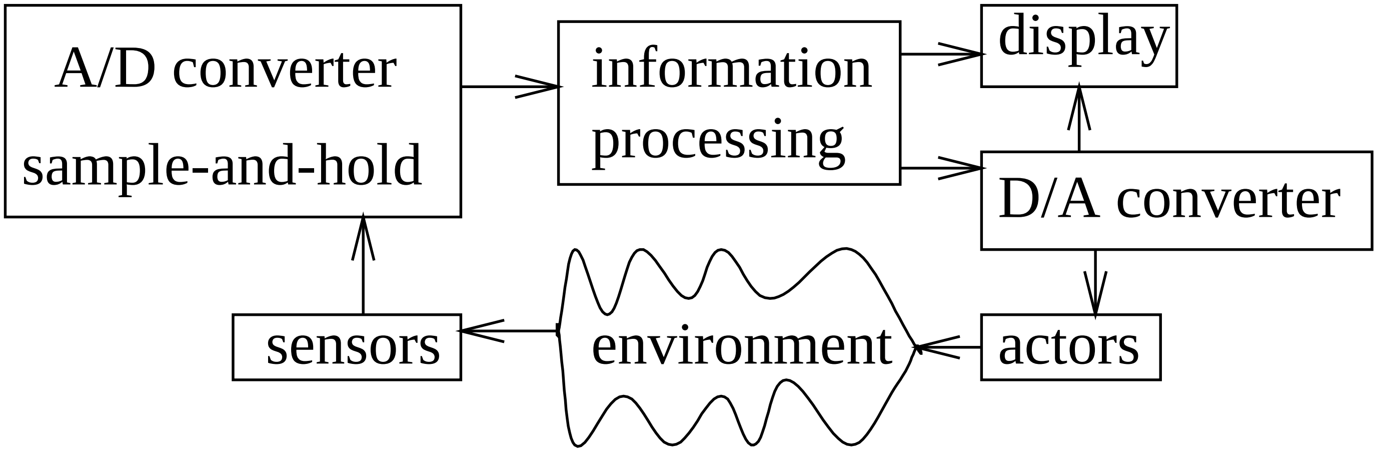
3. Embedded system hardware

Overview:



3.1 Sensors

Processing of physical data starts with capturing this data. Substantial effort and progress during recent years. Sensors can capture:

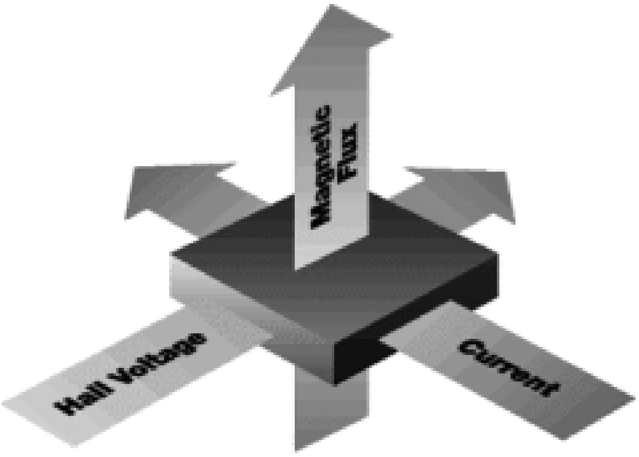
- length
- weight
- accelleration
- strength of fields
- temperature
- brightness

etc.

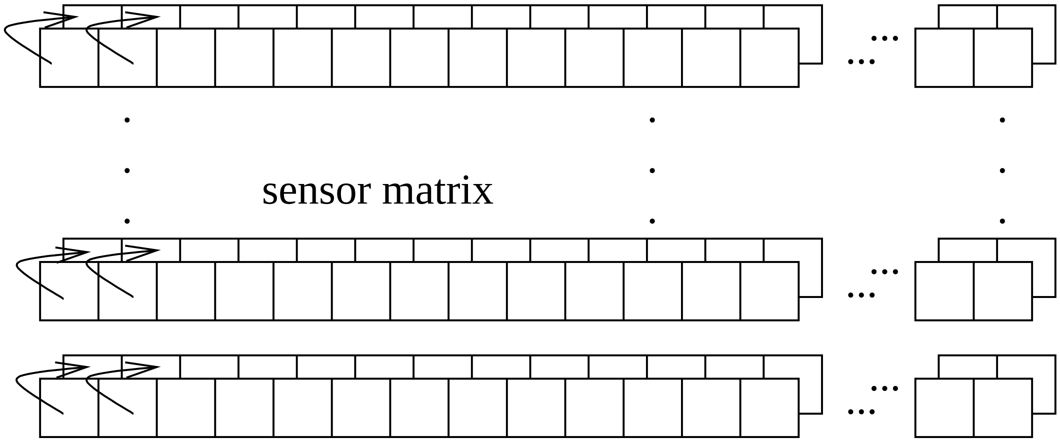
There are also sensors for chemicals.

Examples:

- **Induction sensors**
- **Hall effect sensors:**



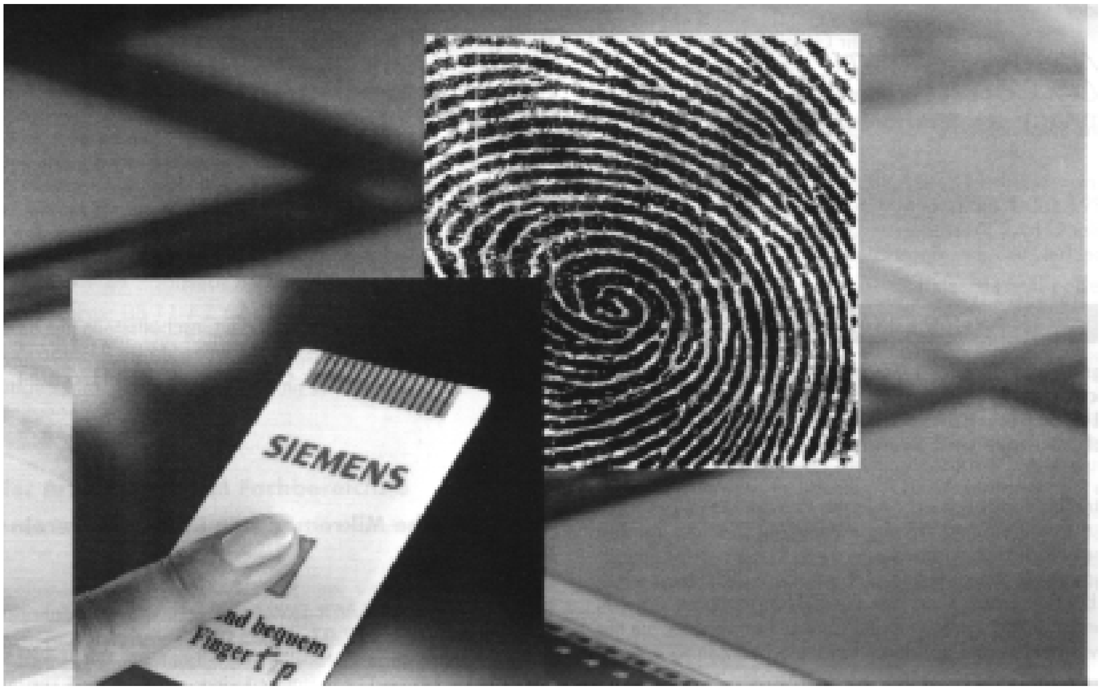
- **CCD cameras** (CCD=charge coupled devices), Example:



- **Rain sensors for wiper control**

> *Sensors multiply like rabbits ... Five human senses no longer sufficient for driving cars.* [ITT Automotive]

- **Light sensors for artificial eyes**
- **Engine control sensors**
- **Finger tip sensors**



256 × 256 elements, resolution: 500 dpi.

- Fingers have an effect on electrical field,
→ electrical image of finger tip patterns in the chip.

- 3D-image
→ photos of finger tip patterns detected.

- Computation of electrical resistance
→ wax copy cannot fool the system.

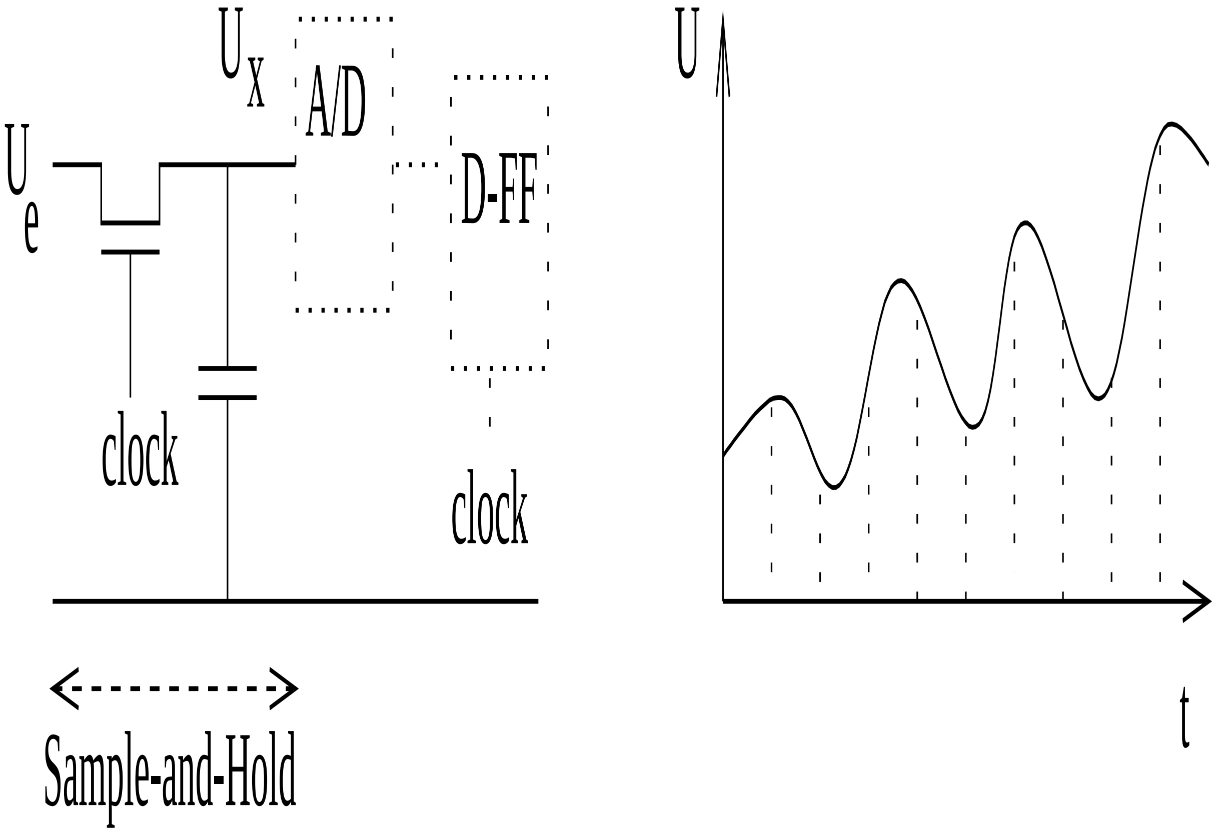
3.2 Sample-and-hold and discrete time

In this course:

restriction to digital information processing.

Known digital computers can process only discrete time series

→ discrete time; sample and hold-devices,



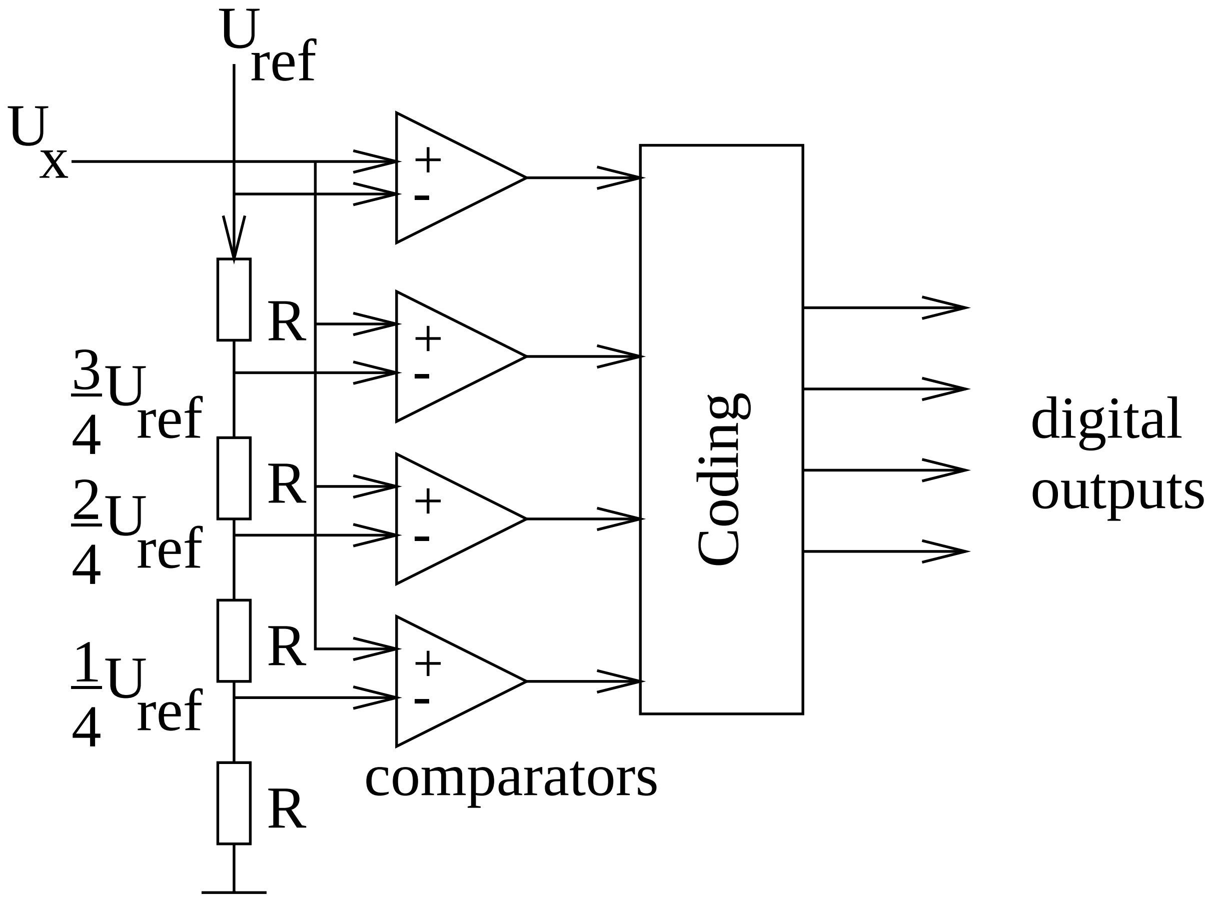
Ideal case: width of clock signal = 0,
full signal amplitude available at output

3.3 A/D Converters

Digital computers require digital form of physical values → A/D conversion.

Many methods with different speeds. Examples:

1. **Direct comparison**



Parallel comparison with reference voltages (n values):

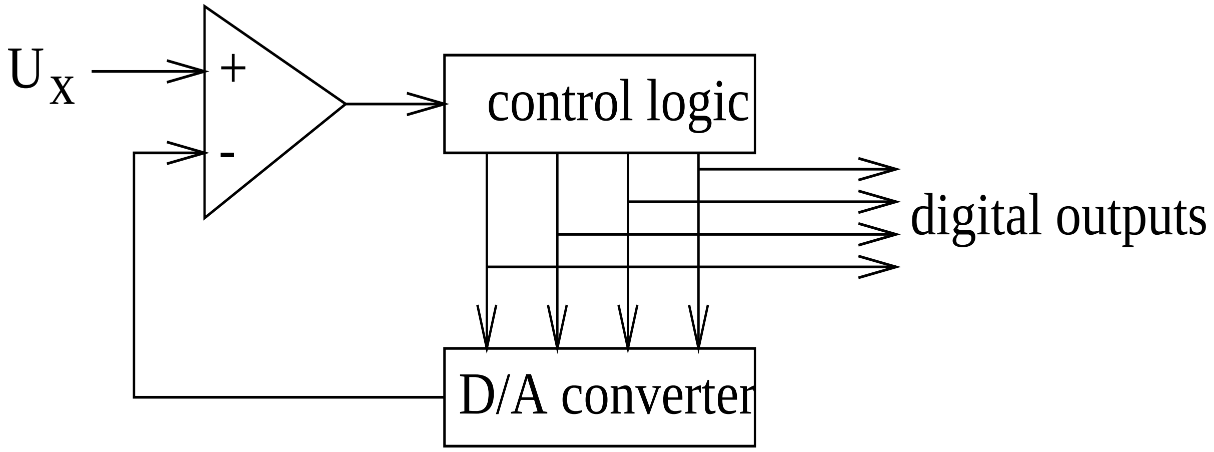
- Time complexity: $O(1)$.

- Hardware complexity: $n - 1$ comparators.

Fast; low resolution/accuracy.

2. **Successive approximation**

Schematic:



Key idea: binary search.

- Set MSB to ’1’

- If analog voltage too large: reset MSB

- Set MSB-1 to ’1’

- If analog voltage too large: reset MSB-1

- ...

Serial comparison for n values:

- Hardware complexity: $ld(n)$

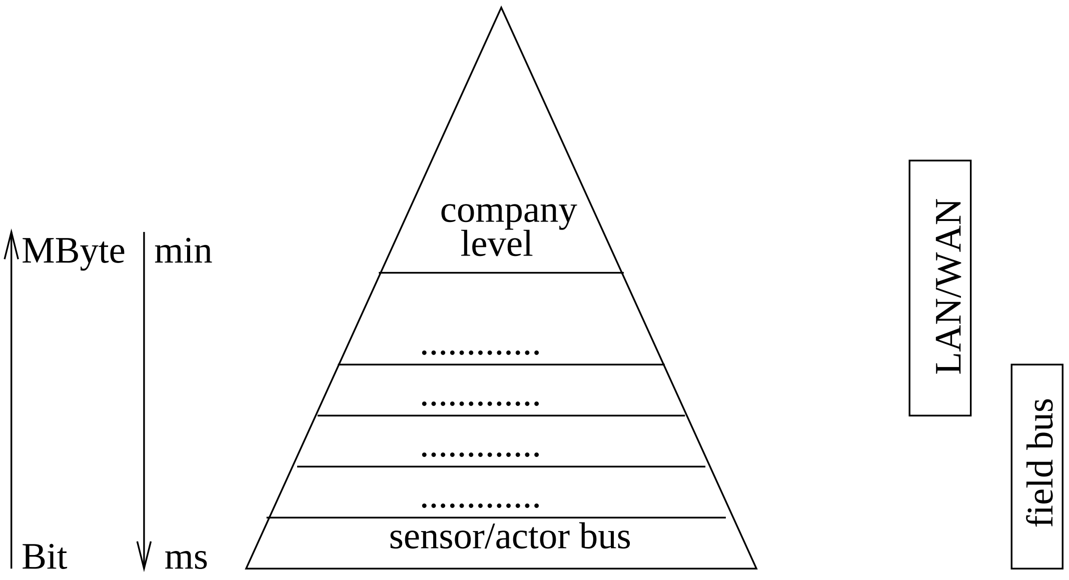
- Time complexity: $ld(n)$.

Less complex than direct comparison, but slower.

3.4 Communication

3.4.1 Requirements

Hierarchy of communications:



Top levels:

- large volumes of data, no real-time requirements.

Lower levels:

- small volumes of data, hard real-time requirements.

Key requirements:

- robust (electrically and mechanically)
- fault tolerant, fast maintenance
- real time behaviour; event driven communication
- adequate speed
- economical; centralized power supply

3.4.2 Basic techniques

Electrical robustness:

- Shielding of cables.

- Optical cables; ....

- Differential (symmetrical) signalling

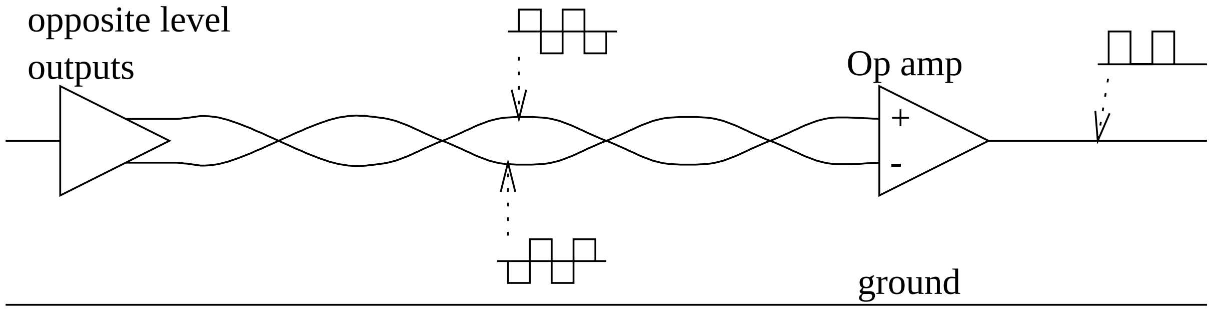
Symmetrical vs. asymmetrical signalling = differential vs. single ended signalling

- Single ended signalling: voltages with respect to ground used for all signals.
- Differential signalling:

  Voltages with respect to second signal wire used.

  Second wire carries opposite level.

  Voltage at input of Op-Amp positive → '1', otherwise '0'.



Used in combination with twisted pairs.

Most noise added to both wires.

Advantages:

– Subtraction removes most of the noise
– Changes of voltage levels have no effect
– Reduced importance of ground wiring
– Higher speed.

Disadvantages:

– Requires negative voltages
– Increased number of wires and connectors

Applications:

– *differential SCSI* (see below)
– *token ring*-networks,
– RS 422,
– ISDN,
– *shielded/unshielded twisted pair* cables (STP- / UTP-cables),
– studio quality, analog audio wires

Other basic techniqes:

- **Fault tolerance:** error detecting and error correcting bus protocolls
- **Privacy**: encryption, virtually private networks
- **Real time behaviour:**

  CSMA/CD (Ethernet) cannot be used
  (no guaranteed response time).

  Alternatives:

  - token rings
  - token busses
  - CSMA/CA (CA = collision avoidance):

    Each partner gets an ID (priority).

    After each bus transfer, all partners try to set their ID on the bus (wired-OR IDs).

    Partners detecting higher IDs disconnect themselves from the bus.

    Highest priority partner gets guaranteed response time, others only if this partner gives them a chance.

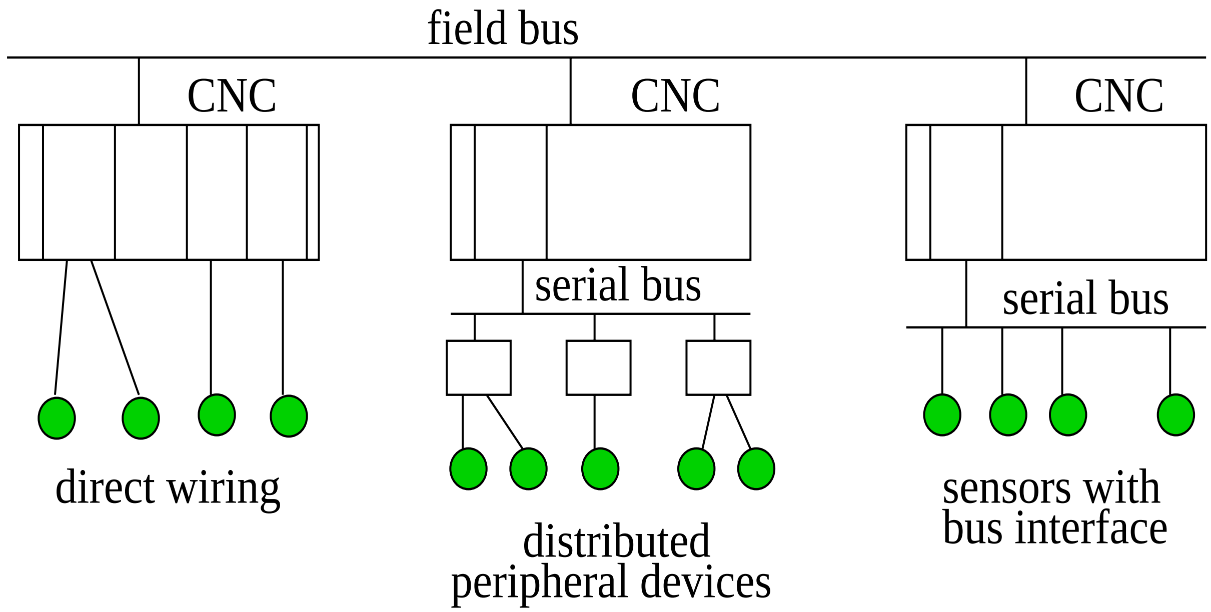
2.4.3 Sensor/actor busses

Principles

Connecting sensors/actors.

Real time behaviour and cost very important.

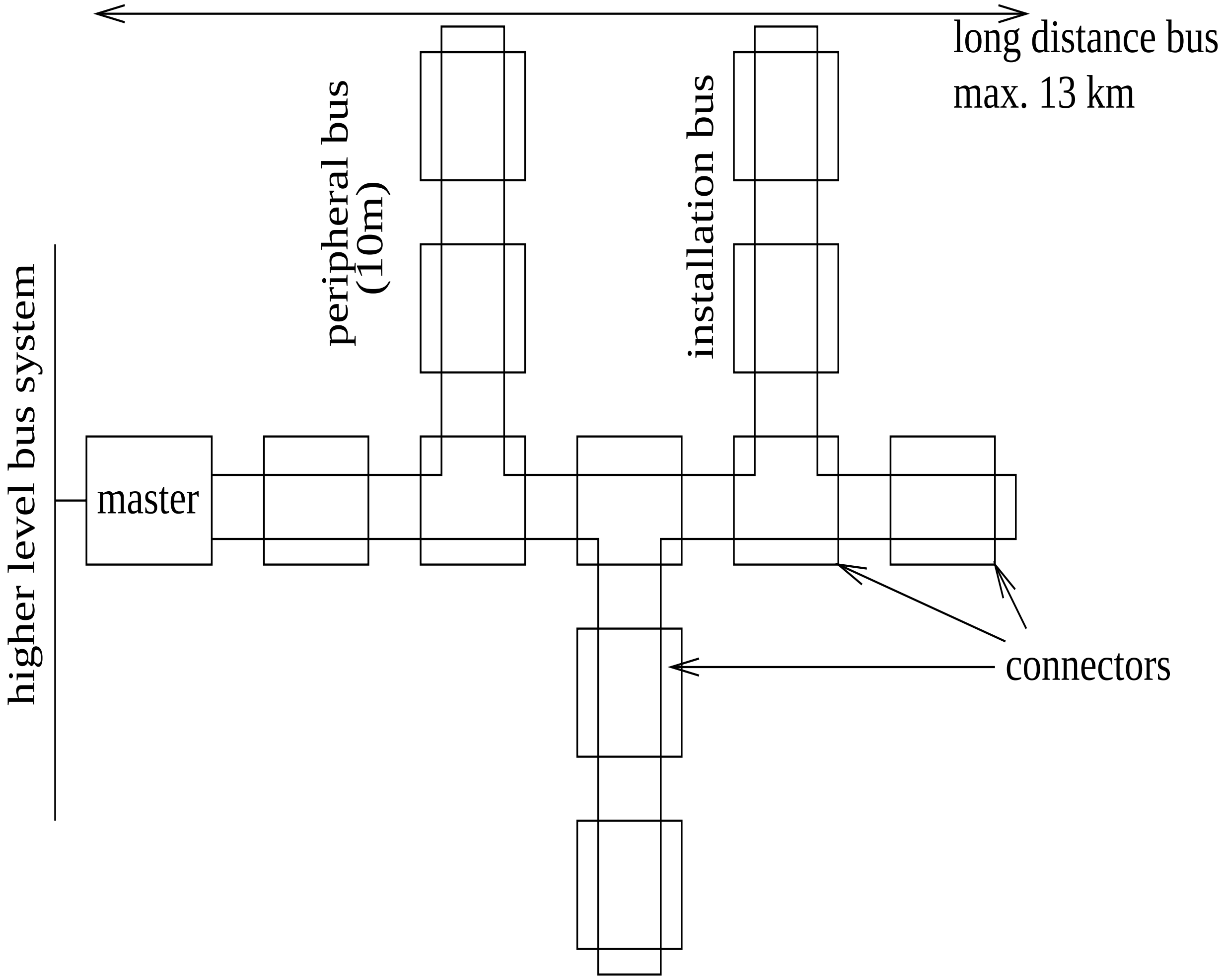
3 different techniques:



1. Direct connections:
   simple sensors, many wires.
2. Special interface components:
   simple sensors, less wires.
3. Sensors with bus interface:
   expensive and flexible.

Examples of sensor/actor busses

- **Interbus-S**



  - bus arbitration: time multiplex

- **Sensor/Actor-Interface ASI**
  - Includes power supply of 3.1 Amps max.
  - Using standard power line cables.
  - bus arbitration: polling

3.4.4 Field busses

More powerful/expensive than sensor interfaces.

Serial busses preferred. Examples:

- **Process Field Bus (Profibus)**

  - National standard.

  - bus arbitration: token passing

  - 9,6 kBit/s (1.200 m) to 500 kBit/s (200 m).

  - Too slow to be used for hard time constraints.

- **Controller Area Network (CAN)**

  - Design by Bosch and Intel in 1981; used in cars and other equipment.

  - Bus access method: CSMA/CA

  - Twisted pair wiring

- **European Installation Bus (EIB)**

  - Designed for smart buildings.

  - bus access method CSMA/CA

  - low data rate

- **IEEE 488**, designed for laboratory equipment.

- **MAP**

  Designed for factory automation; token bus.

2.5 Information processing

Large number of possible hardware components.

Examples:

2.5.1 Processors

Key advantages

- Flexibility
- Reuse of standard components

> *At the chip level, embedded chips include microcontrollers and microprocessors. Microcontrollers are the true workhorses of the embedded family. They are the original 'embedded chips' and include those first employed as controllers in elevators and thermostates.* [EEDesign]

> *... the New York Times has estimated that the average American comes into contact with about 60 microprocessors every day ...* [Camposano]

Recent BMWs: > 100 Processors.

Microcontrollers

Scope: control dominated applications

Typical: Intel 8051:

1. 8-Bit CPU, optimized for control applications
2. Support for Boolean data types
3. instruction address space: 64 kBytes
4. data address space: 64 kBytes
5. 4 k bytes program memory on chip
6. 128 bytes data memory on chip
7. 32 I/O lines
8. Two counters on chip
9. Asynchronous receiver/transmitter (UART) on chip
10. 5 interrupt vectors
11. clock generation on chip
12. many variations

1,2,5,6,7,8,9 and 11 typical for microcontrollers.

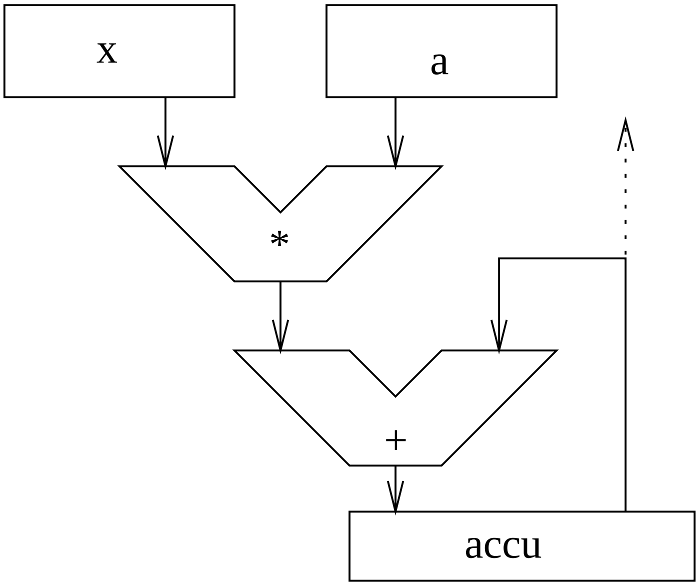
Embedded processors

Key feature of embedded processors: efficiency.

→ Support for application-domain specific instruction sets.

Example: support for digital signal processing

Sample data path of DSP processor:



$$y(i) = \sum_{j=0}^{n} x(i-j) * a(j)$$

Architecture allows for single cycle computation of updated sum.

Includes:

- specialized registers (e.g. accu)

- limited parallelism

- special multiply/add instruction

General register file would require more instruction bits, longer cycle time and more power.

General properties of DSP processors

- **(limited) parallelism**

  E.g., parallel moves

  Application: compilation of

  $$y_i = \sum_{j=0}^{n} x_{i-j} * a_j$$

  into:

  $k = i - 1, l = 1, W = x_i, Z = a_0;$

  $loop : acc = acc + W * Z, W = x_k, Z = a_l,$
  $k = k - 1, l = l + 1, IF...GOTO\ loop$

  Loop will be single instruction.

- **special addressing modes**

  Modulo addressing for implementing ring buffer

  ...

- **heterogenous register sets**:

  See previous slide

- **multiply/accumulate instructions**:

  See previous slide

- **saturating arithmetic:**

  Returns largest/smallest numer in case of overflow/underflow.

  Examples (natural numbers):

	0111
+	1001
wrap-around arithmetic	0000
saturating arithmetic	1111

	0111
-	1001
wrap-around arithmetic	1110
saturating arithmetic	0000

- **Real time capability**:

  No caches, no virtual memory.

- **Many MIPS/Watt**

Techniques for high code density:

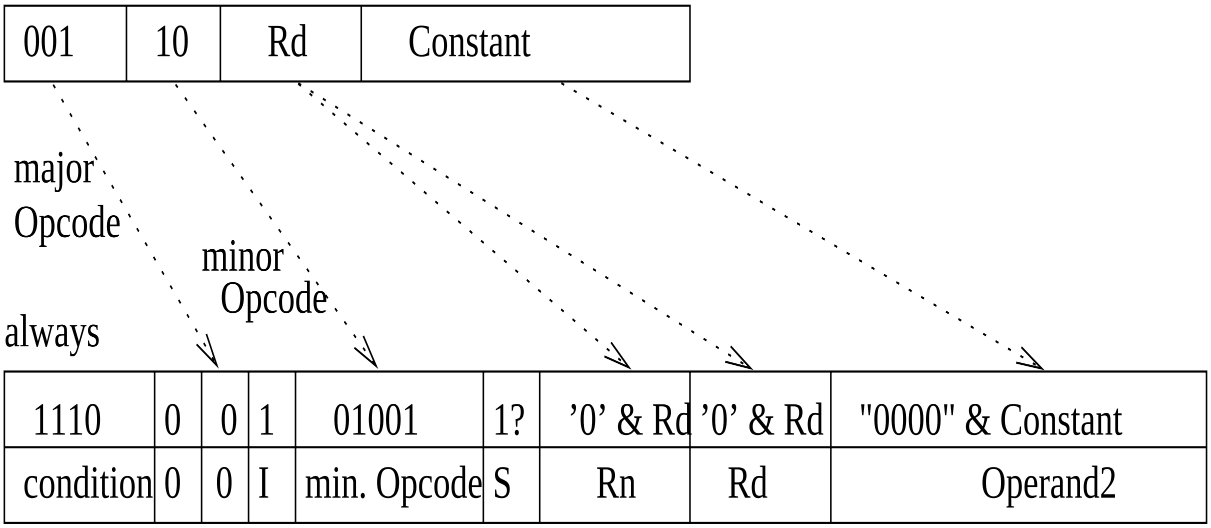
1. **CISC** processors

   Example: Motorola MC 680x0.

2. **Compression with 16-bit instruction set**

   Example: ARM with thumb instructions set

   Dynamic expansion of 16-bit external instructions into 32-bit internal instructions.



001	10	Rd	Constant

major Opcode

minor Opcode

always

1110	0	0	1	01001	1?	’0’ & Rd	’0’ & Rd	"0000" & Constant
condition	0	0	I	min. Opcode	S	Rn	Rd	Operand2

   Processor with 16-bit memory almost as fast as processor with 32-bit memory, but much more cost-effective.

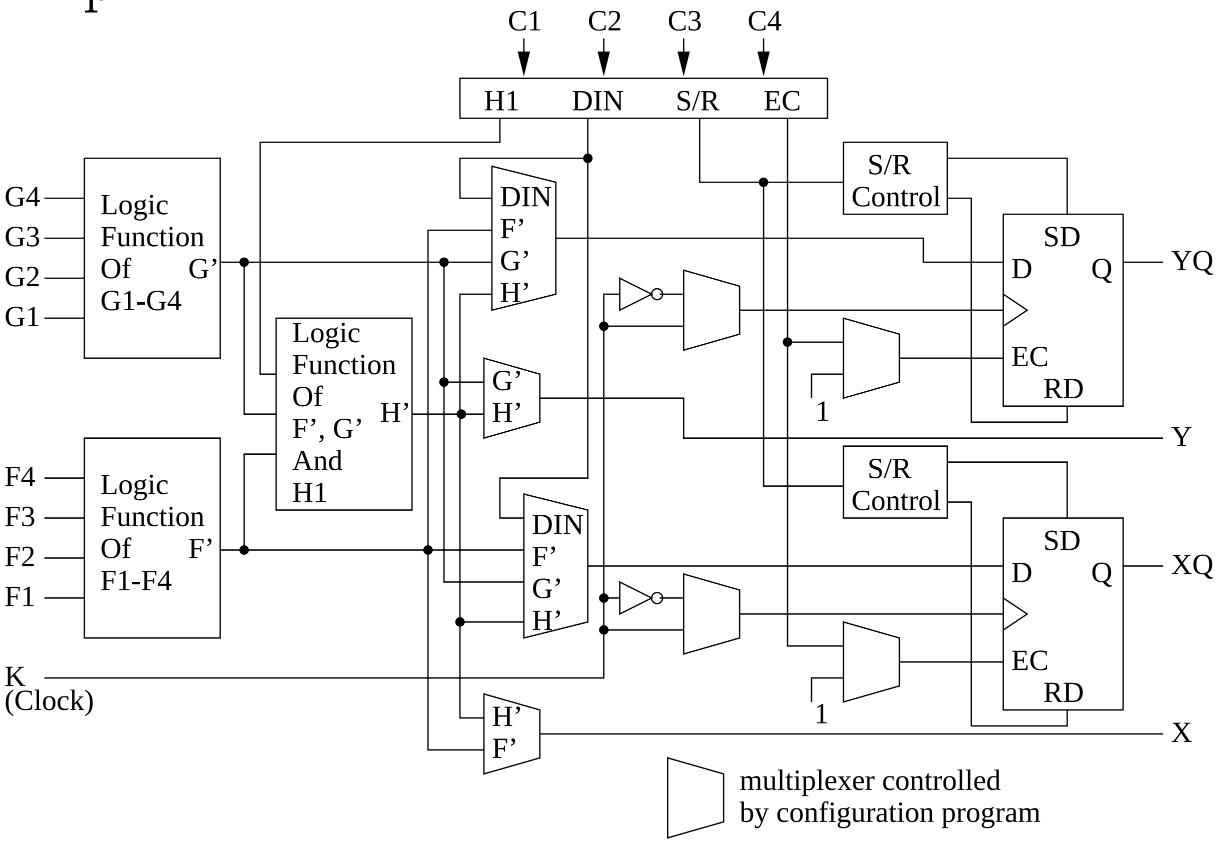
3. **Cache-based decompression**
4. **predicated execution**
5. ...

5.2.2 Field Programmable Gate Arrays (FPGAs)

Pre-fabricated hardware components.

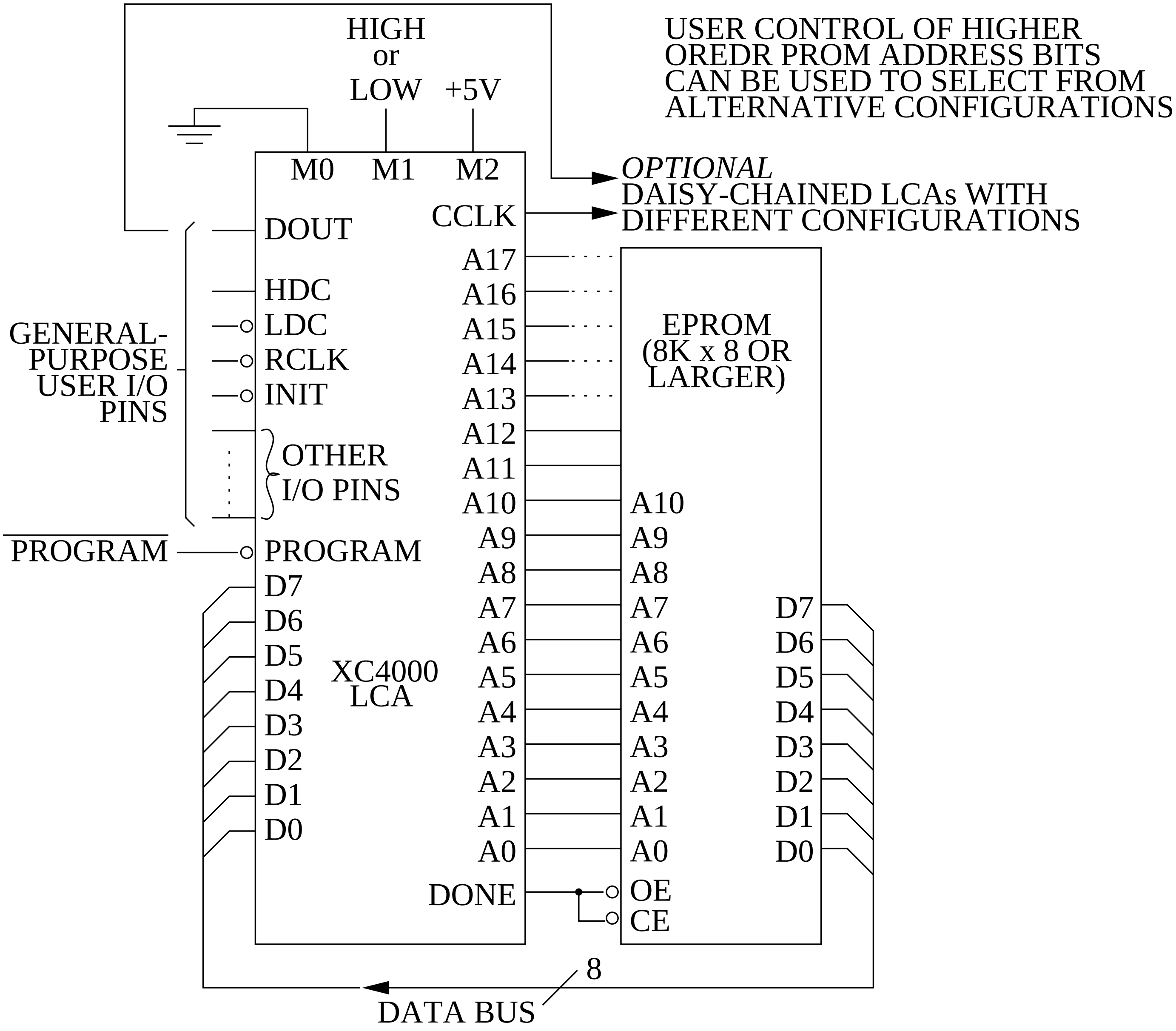
Blocks (CLBs) can be configured to perform any (computable) function.

Example:



Interconnections: Configurable MOS switches

Configuration using EPROMs



5.2.3 PCs

Increasingly used for many applications
(main reason: low cost).

Problems:

- Robust PCs required (industrials PCs, IPs)
- Standard operating system has to be extended/ replaced by real time operating system.

5.2.4 Application specific integrated circuits (ASICs)

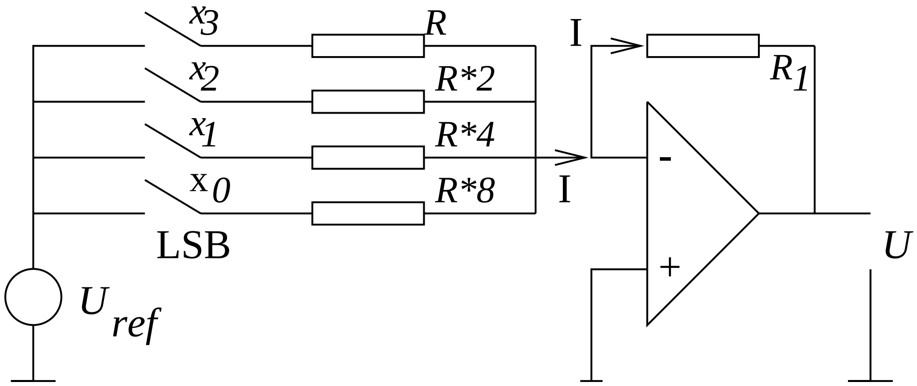
High costs, long turn-around time.

5.2.5 Computerized numerical control (CNC)

Low level programmability.

3.6 D/A Converters

Schematic (Example):



Ideal Op-Amp: voltage between inputs = 0

$$U = R_1 * I$$

I equal to current flowing from the left:

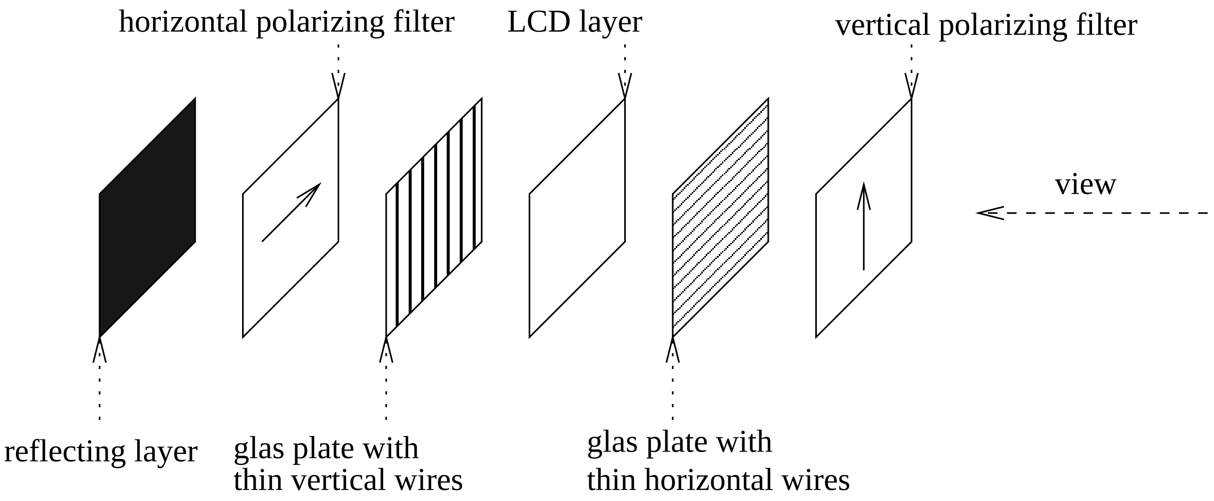
$$\begin{aligned}
I &= x_3 \cdot \frac{U_{ref}}{R} + x_2 \cdot \frac{U_{ref}}{2R} + x_1 \cdot \frac{U_{ref}}{4R} + x_0 \cdot \frac{U_{ref}}{8R} \\
I &= \frac{U_{ref}}{R} \cdot \sum_{i=0}^{3} x_i * 2^{i-3} \\
U &= U_{ref} * \frac{R_1}{R} * \sum_{i=0}^{3} x_i * 2^{i-3} \\
&= U_{ref} * \frac{R_1}{R} * \frac{nat(x)}{8}
\end{aligned}$$

Output voltage proportional to value coded in x. Very fast. Accuracy depends on resistors.

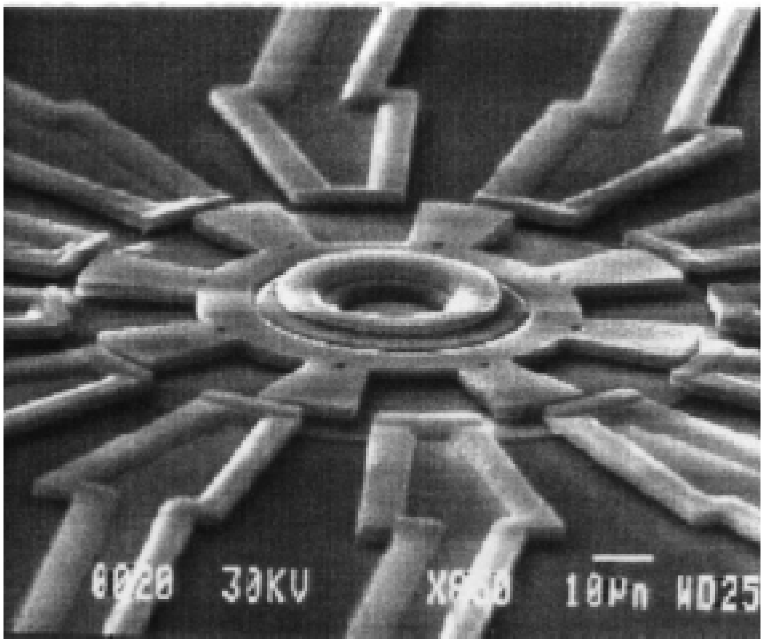
3.7 Output

Large amount of output devives. Examples:

- **Displays**, example (TFT-Display):



- **Actors**, Example (©MCNC):



→ micromachinery.