Embedded Systems

An introduction

Historical background

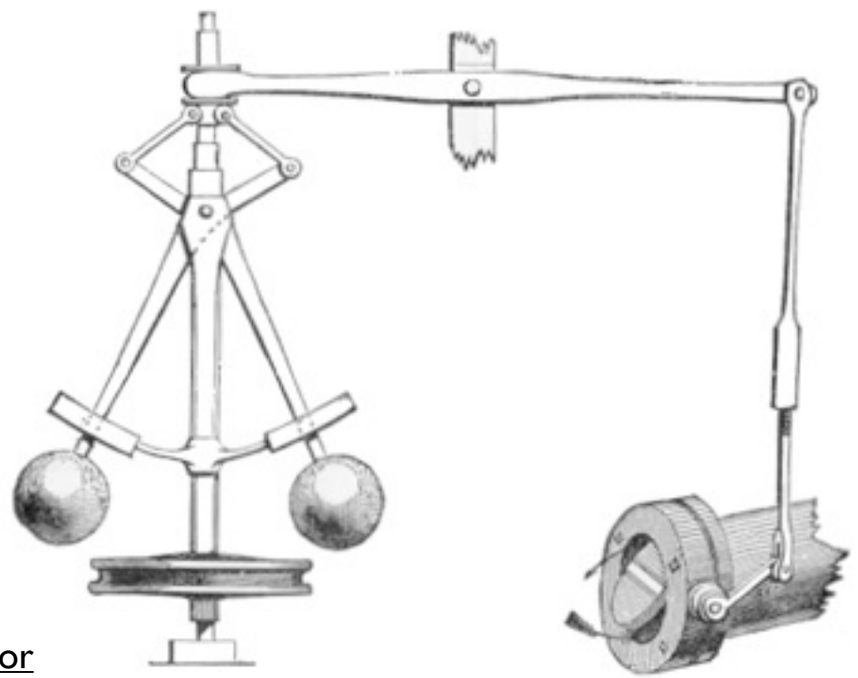
- Most machines that we rely on in our everyday life need some sort of regulating system (or control system) to achieve their goals.
- Although these control systems are nowadays often implemented as computer systems, they have been around way before the invention of computing

Historical background

- Example:
  - To keep constant the speed of a steam machine one needs a system that regulates the amount of steam sent to the pistons
  - When the machine slows down, more steam must be sent.
  - When the machine accelerates, less steam must be sent.

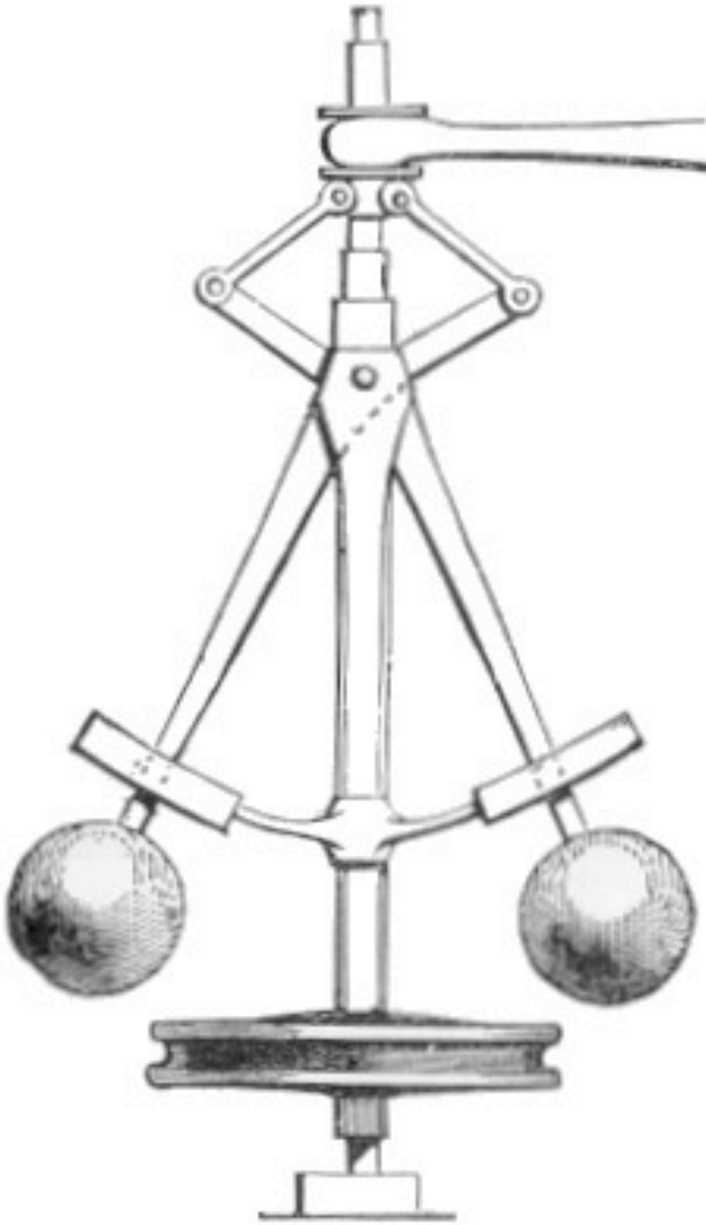
Historical background

- Example:
  - Can this be done automatically ?
  - Yes: the centrifugal governor (17th century)



source: http://en.wikipedia.org/wiki/Centrifugal_governor

Historical background

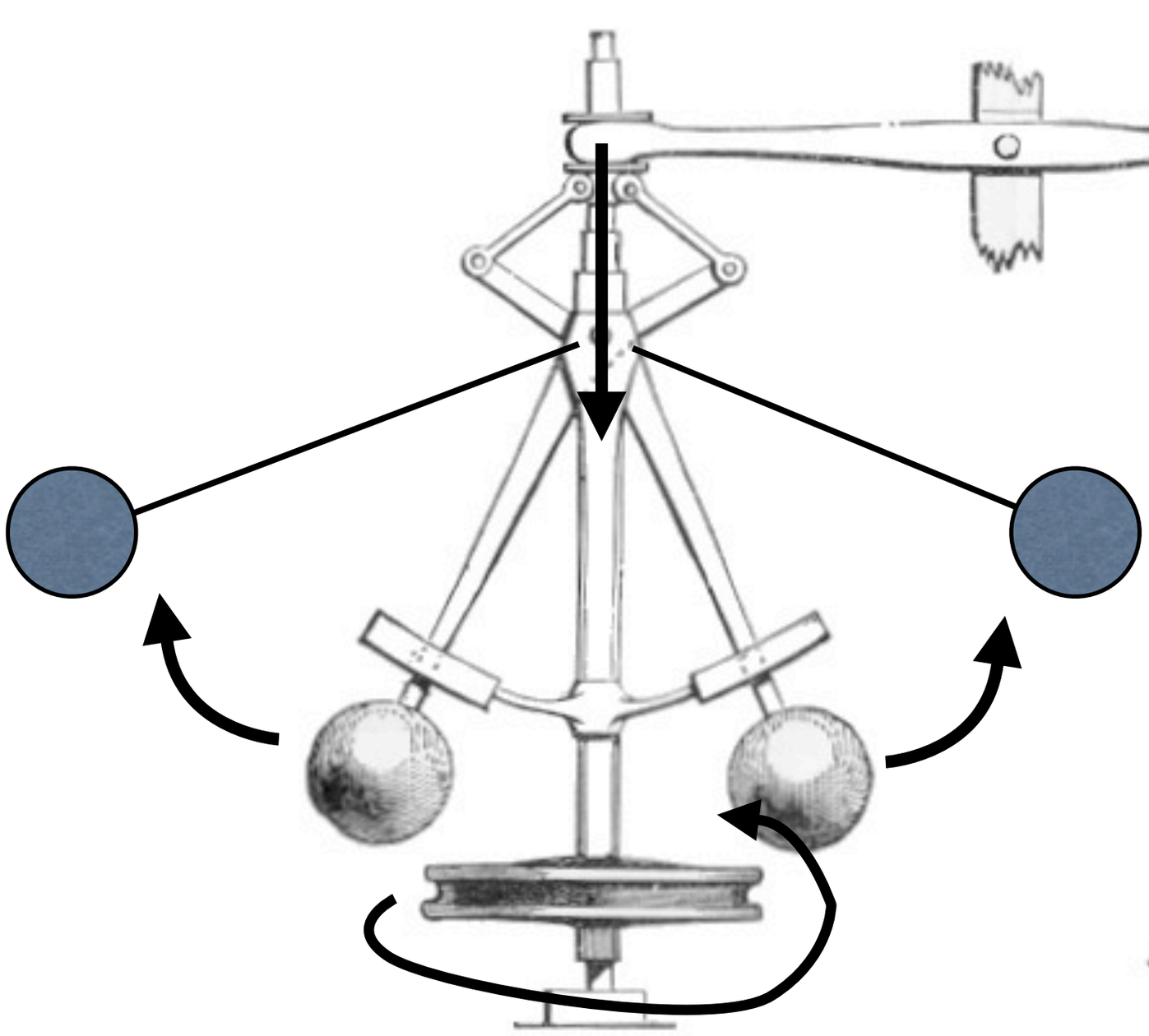


rod connected to the wheels

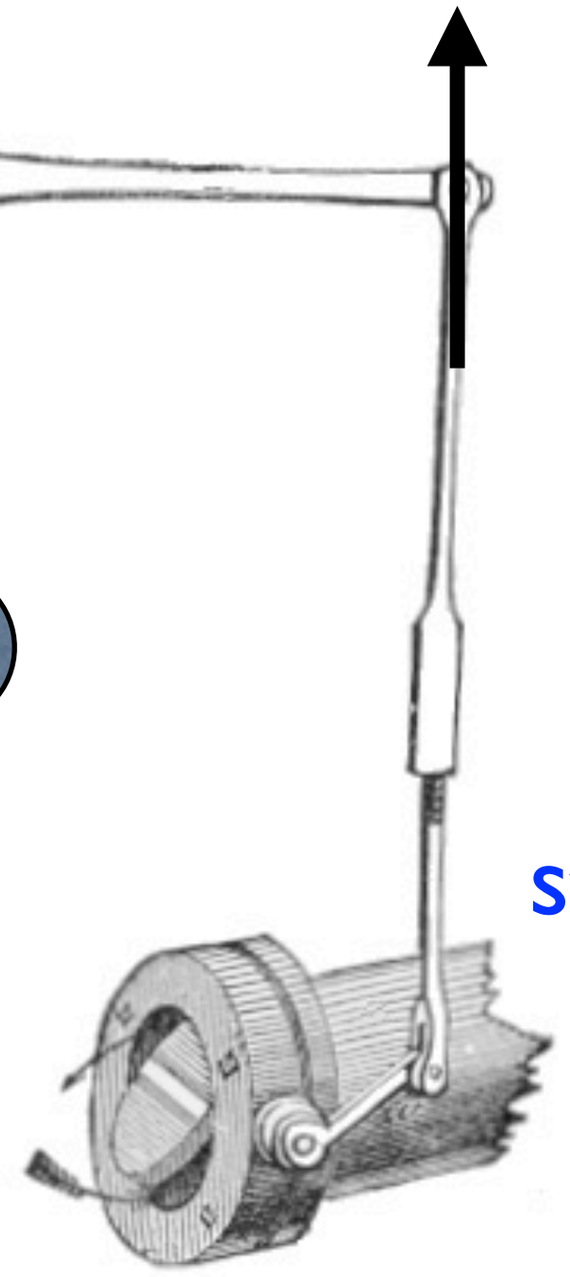
steam valve

source: http://en.wikipedia.org/wiki/Centrifugal_governor

Historical background



quick rotation

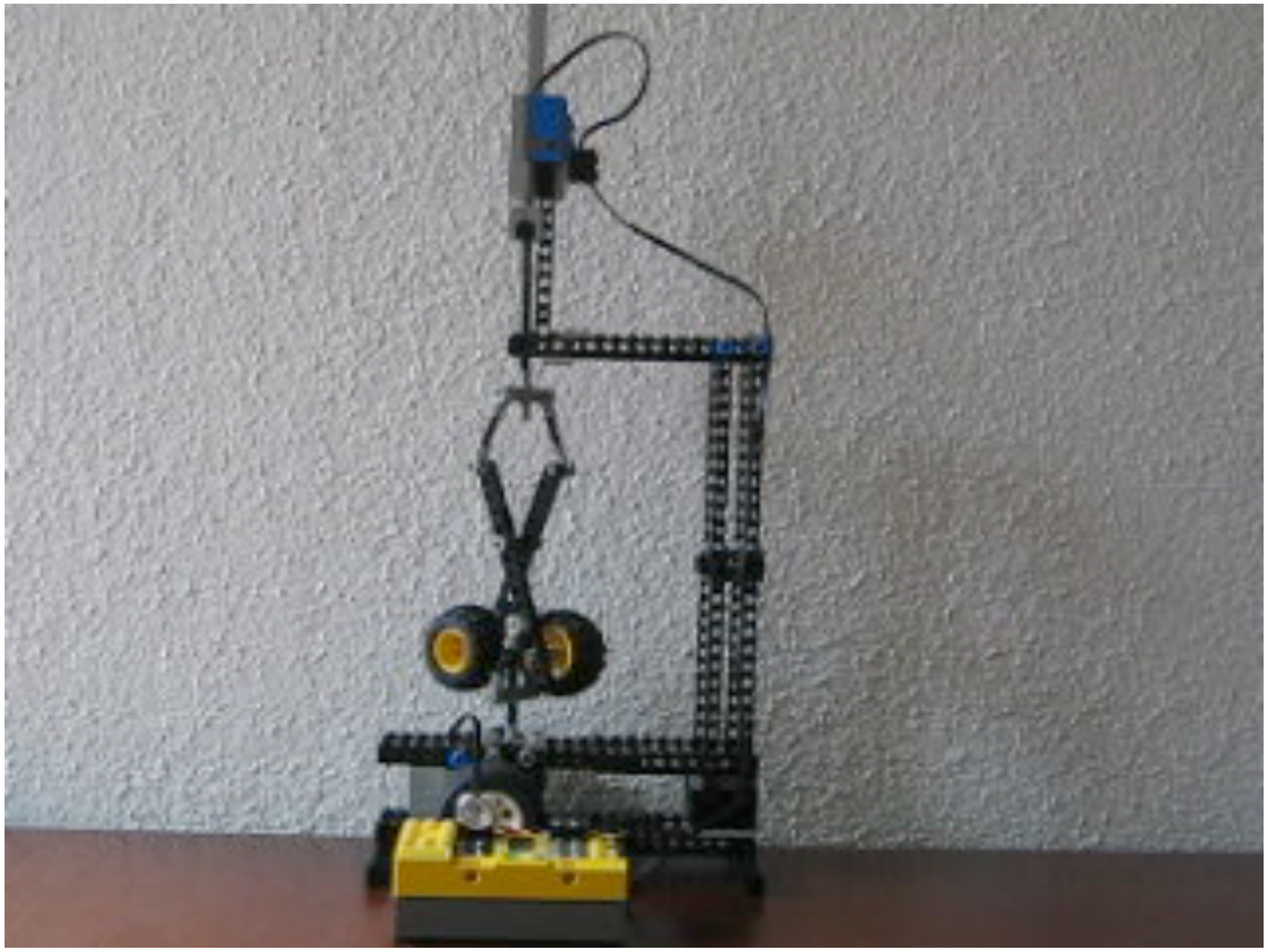


closing

steam valve

source: http://en.wikipedia.org/wiki/Centrifugal_governor

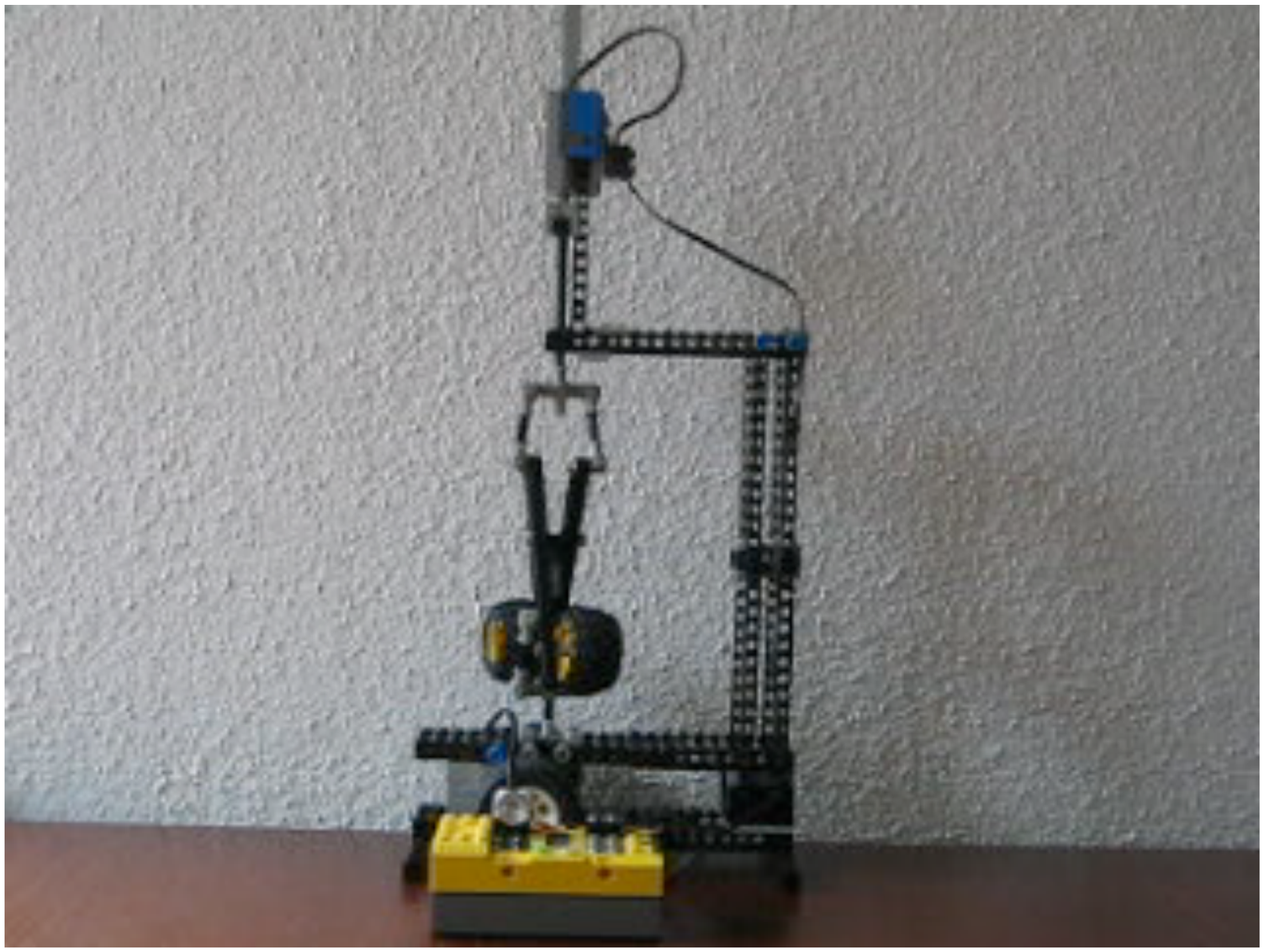
Historical background



source: http://www.legoengineering.com

Historical background

Calibrated version

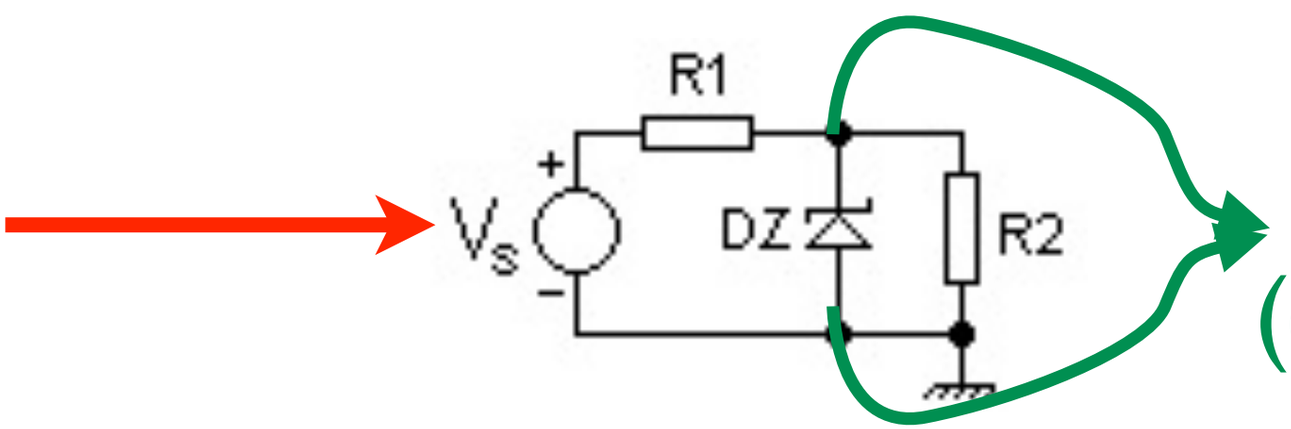


source: http://www.legoengineering.com

Historical background

- Such control systems can thus be implemented by purely mechanical means
- More recently, these systems have been implemented using electric and electronic devices

input voltage (varies)



output voltage (constant)

The voltage encodes a given measure

Historical background

- Since the early ‘50s, computer systems have developed more and more quickly and have become more and more pervasive
- Computer scientists have quickly realised that computers are not only «business machines»
- Computer = data manipulating device

Historical background

- It thus makes a lot of sense to use a computer as a part of a broader system, in order to...
  - replace an analog regulative device
  - perform a treatment/computation that would be too difficult to perform with an analog device.

Historical background

- Advantages of using a computer:
  - greater flexibility
    - the program can be easily modified
    - code can be re-used
  - nowadays computer deliver very high computing power
  - usually more compact

Historical background

Control systems

Computers

Embedded systems

Historical background

- One of the first example of embedded system goes back to the ‘60s
- It was the computer on board the spaceships of the Apollo NASA program.
- That computer could control, in real time, actual flight parameters and would adapt the spaceship course
- It was an interactive computer

Historical background

User Interface

NO CPU: 4000+ integ. circuits with 3 NOR gates each

32 Kb RAM

72 Kb ROM

2MHz

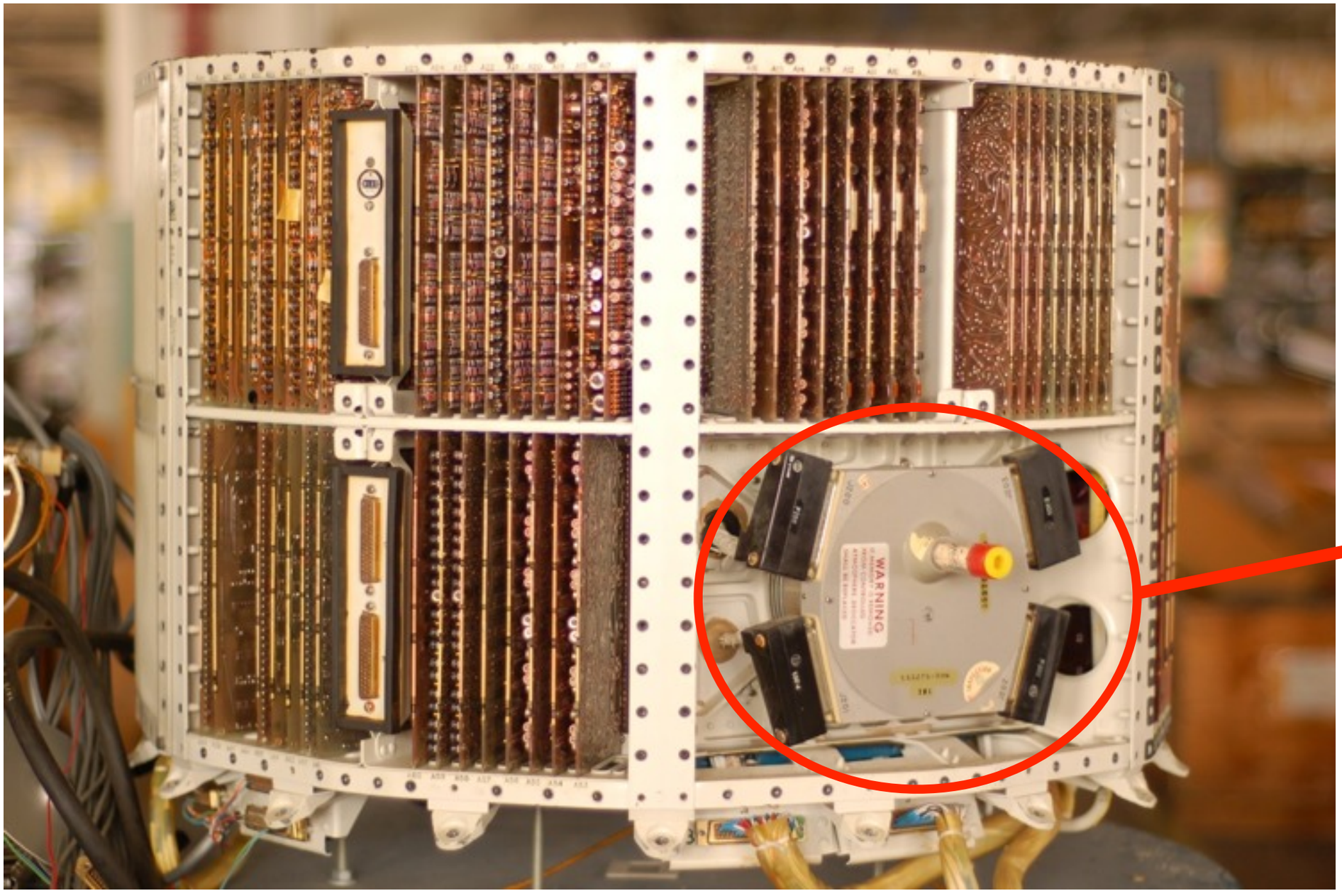
Assembly language (11 instructions)

source: wikipedia

Historical background

- The first mass produced embedded system is the Autonetics D-17.
- It served as a control system for US nuclear weapons LGM-30 Minuteman
- Produced from 1962

Historical background



The HD was used as primary memory !

source: wikipedia

Historical background

- Since then, embedded systems have (hopefully) become more diverse
  - They have enabled the huge development of the consumer electronics market
    - mobile phones, MP3s, etc
- Embedded systems are also widely used in the industry
  - Production lines control, etc

Historical background

- These last years, we have witnessed a significant convergence between consumer electronics and computer systems:
  - The Microsoft XBox is just a PC wrapped up as a gaming system
  - Desktop PCs (and Macs...) can easily be turned into *media-centers* to replace a traditional hi-fi audio system

Embedded Systems

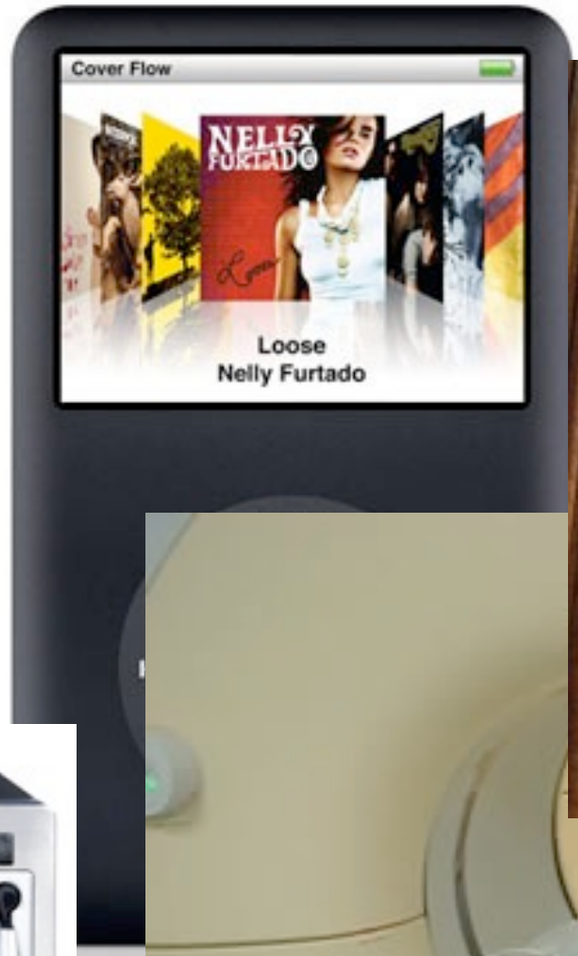
Embedded Systems

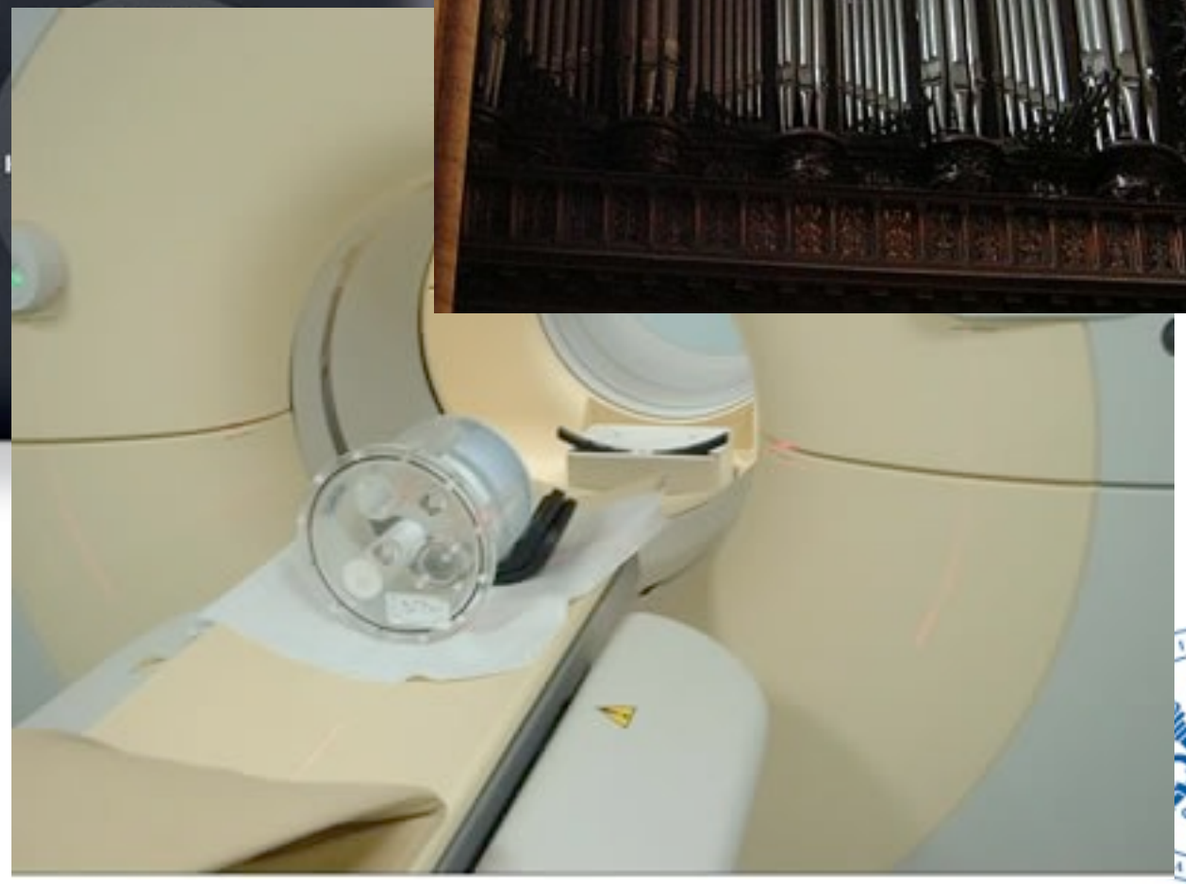
Embedded Systems

Embedded Systems

Embedded Systems

Embedded Systems





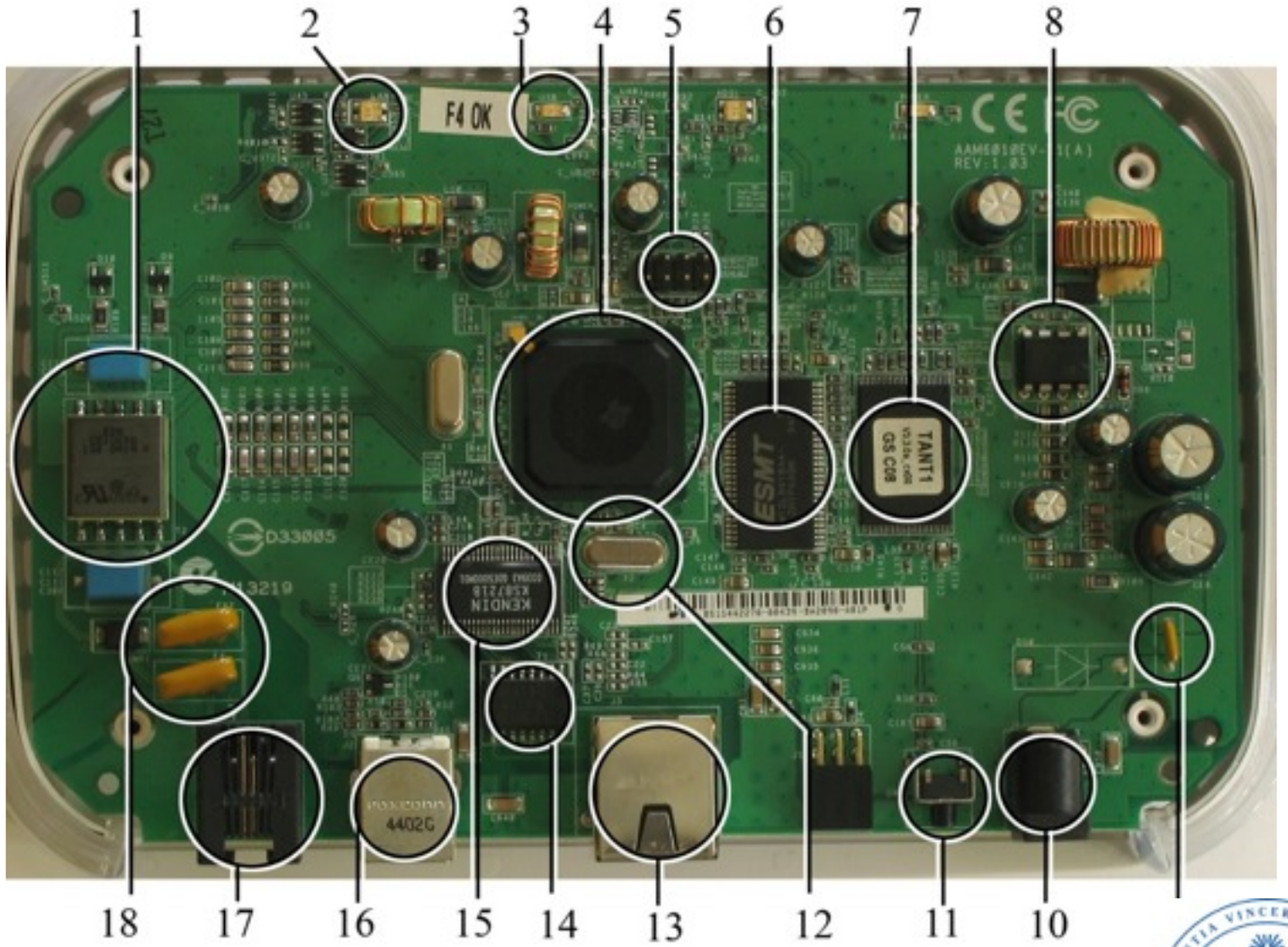
Definition

- No unanimous definition
- Embedded system = Computer system designed to perform one or several dedicated functions.
- For instance, the control system of an underground train has been designed solely for that purpose. It cannot be used to run a word processor

Example

- Modem / router Netgear ADSL

1. Telephone decoupling electronics (for ADSL).
2. & 3. LEDs
4. Main processor (TNETD7300GDU)
6. RAM, 8 MB chip.
7. Flash memory,
8. Power supply regulator.
10. Power connector.
11. Reset button.
12. Quartz crystal.
13. Ethernet port.
16. USB port.
17. Telephone (RJ11) port.



source: wikipedia

Characteristics

- Tight constraints:
  - Low power CPU
  - Low memory (primary and secundary -- if any)
  - Few I/Os
  - ...

Characteristics

- Increased demand on reliability
  - ES are hard to debug
  - ES are present in critical applications !
- The constraints are usually complex (real-time...)

Characteristics

- Increased demand on reliability
  - ES are hard to debug
  - ES are present in critical applications !

- The constraints are usually complex (real-time...)

Prominence

- The ES market is huge and much more important than that of personal computers
- Figures in 1997:
  - 30 millions CPUs sold in personal computers
  - 3 billions CPUs sold in ES

Market

- World ES market (million $)

	2004	2008	2013 (pred.)
Software	1 641	2 200	2 900
Hardware	44 229	89 800	109 600
Total	45 873	92 000	112 500

source: "Future of embedded systems technology" BCC report ITF016A and ITF016C

Prominence

- The ES market is significantly driven by consumer electronics
- In 2004, it has been estimated that 42% of the embedded OS sales are for consumer electronics

source: http://cordis.europa.eu/ist/embedded/facts_figures.htm

Importance

- Embedded systems make it for an increasing larger share in the final value of the product

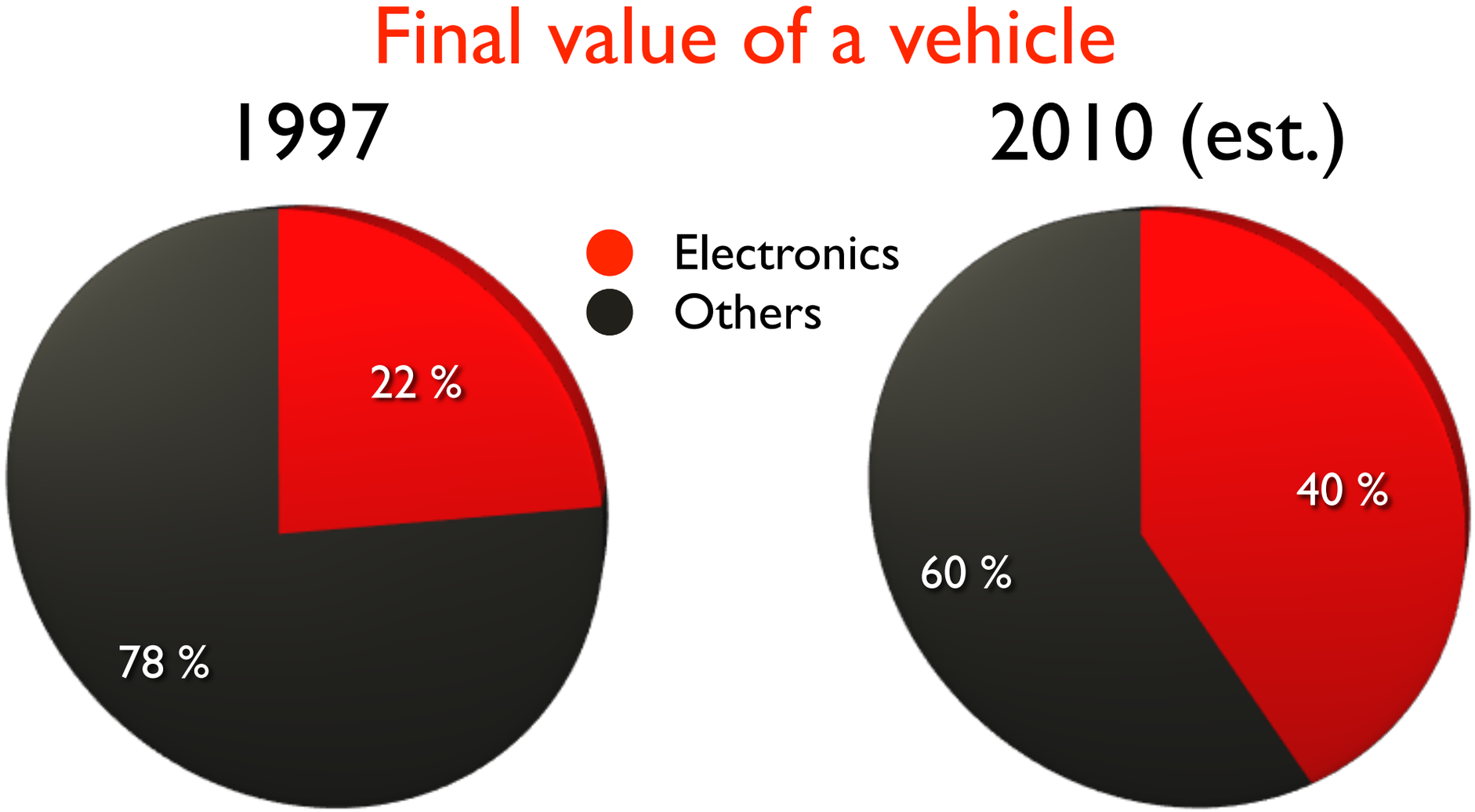
Type	% value of the final product	Type	% value of the final product
Telecom.	37	Industrial app.	22
Consumer electronics	41	Medical app.	33

source: http://cordis.europa.eu/ist/embedded/facts_figures.htm

Prominence

- This is particularly true in the automotive business.

Final value of a vehicle



source: http://cordis.europa.eu/ist/embedded/facts_figures.htm

Constraints

Limited resources

- Unlike desktop systems, ES design has to take into account several curbs on the available resources:
  - costs: have to be kept as low as possible for mass marketing
  - size: some ES have to fit into a few sq. inches
  - energy: many ES run on batteries

Limited resources

- These resources strongly limit the features of the system
- But other constraints come from the difficulty of developing software for ES:
  - few I/Os: no keyboard, no screen, hardly a serial port...

Reliability issues

- ES that control a critical system tolerate no failure !
  - This is a hard to reach goal, therefore several laws and regulations exist:
    - In Europe, a physical link is still compulsory between the driving wheel and the steering system of a car
    - In Belgium, computer control is banned from nuclear power plants

Reliability issues

- Reliability is also an issue for non-critical systems that are hard to update
  - No ‘windows update’ for your DVD player...
  - Thus, the product has to meet high quality standards as soon as it exits the factory !

Example

Ariane 5 maiden flight, 1996

source: http://en.wikipedia.org/wiki/Ariane_5

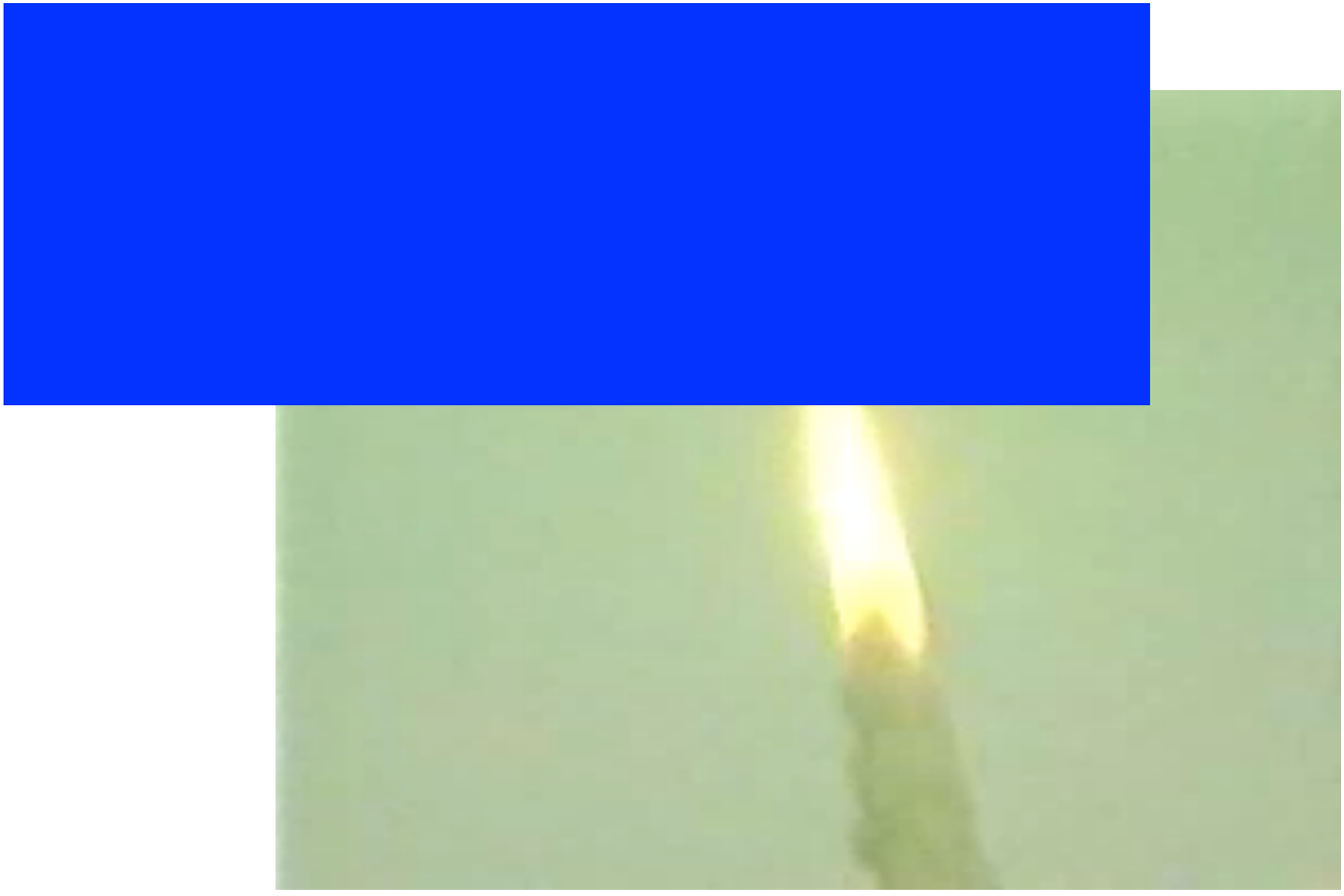
Example

Ariane 5 maiden flight, 1996

source: http://en.wikipedia.org/wiki/Ariane_5

Example

Ariane 5 maiden flight, 1996

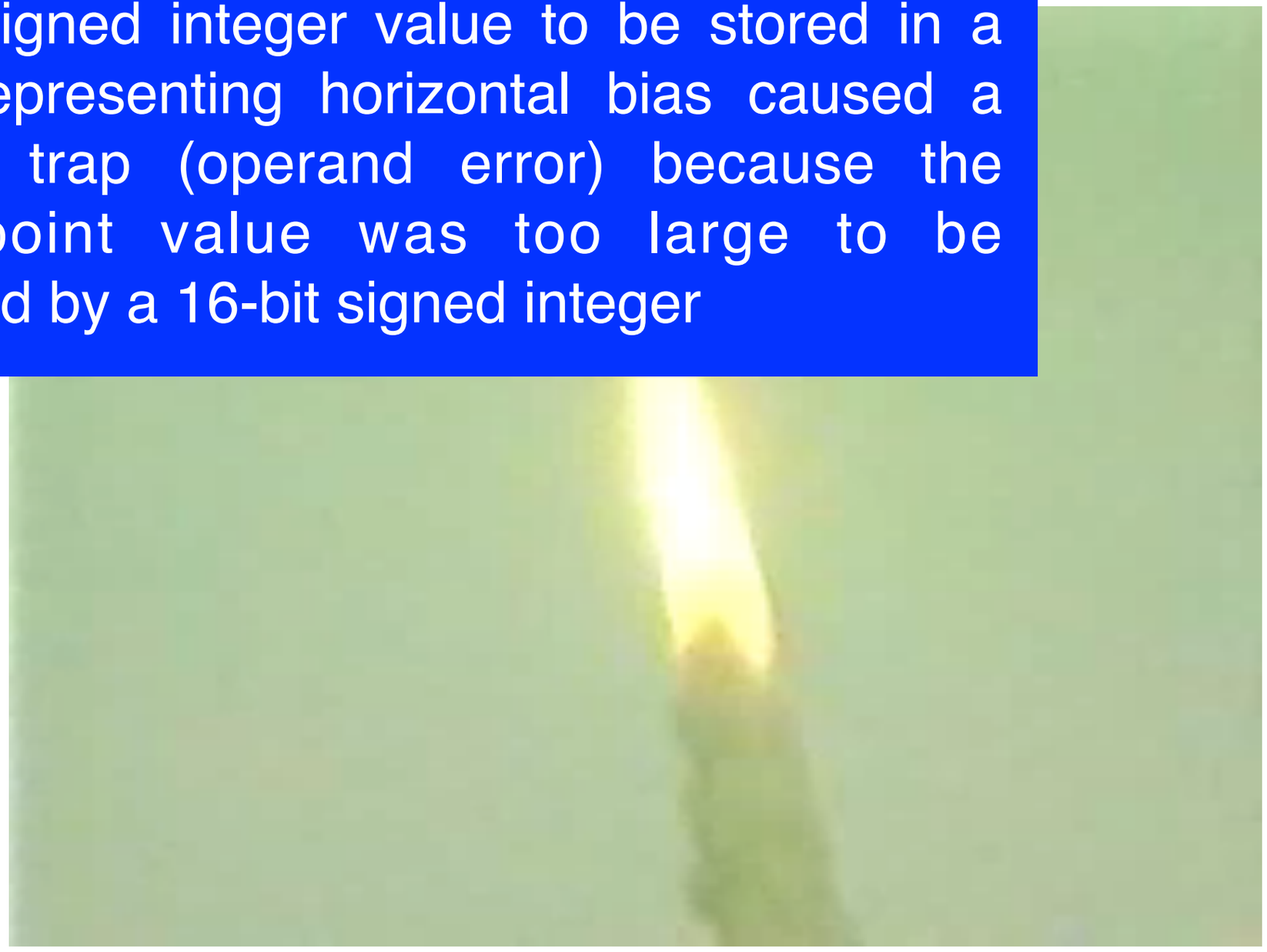


source: http://en.wikipedia.org/wiki/Ariane_5

Example

Ariane 5 maiden flight, 1996

A data conversion from 64-bit floating point value to 16-bit signed integer value to be stored in a variable representing horizontal bias caused a processor trap (operand error) because the floating point value was too large to be represented by a 16-bit signed integer



source: http://en.wikipedia.org/wiki/Ariane_5

Example

Ariane 5 maiden flight, 1996

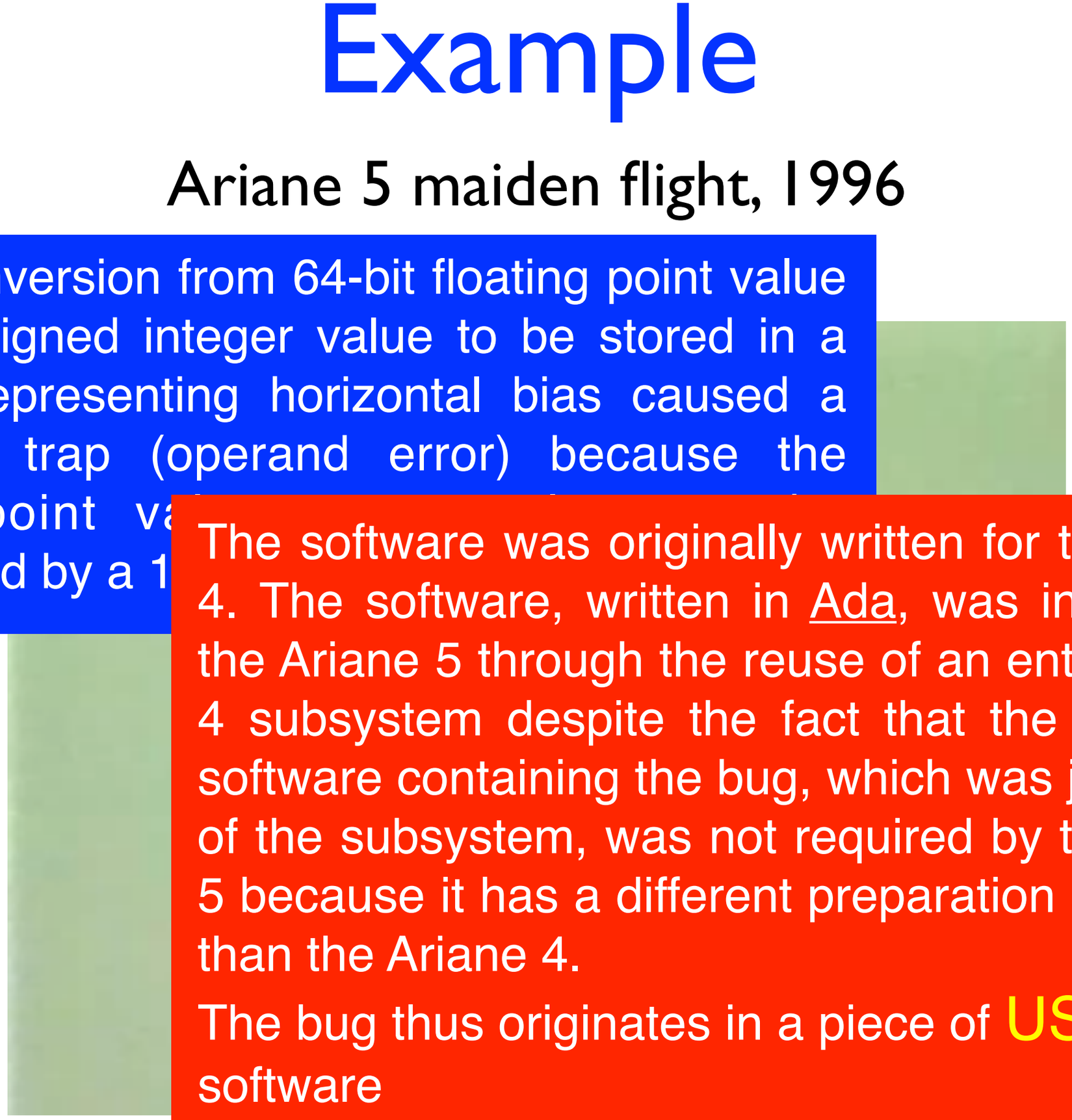
A data conversion from 64-bit floating point value to 16-bit signed integer value to be stored in a variable representing horizontal bias caused a processor trap (operand error) because the floating point va
represented by a 1

source: http://en.wikipedia.org/wiki/Ariane_5

Example

Ariane 5 maiden flight, 1996

A data conversion from 64-bit floating point value to 16-bit signed integer value to be stored in a variable representing horizontal bias caused a processor trap (operand error) because the floating point va
represented by a 1



The software was originally written for the Ariane 4. The software, written in Ada, was included in the Ariane 5 through the reuse of an entire Ariane 4 subsystem despite the fact that the particular software containing the bug, which was just a part of the subsystem, was not required by the Ariane 5 because it has a different preparation sequence than the Ariane 4.

The bug thus originates in a piece of USELESS software

source: http://en.wikipedia.org/wiki/Ariane_5

Example

- Qantas flight 72 - Airbus 330 - 303 accident on October, 7th, 2008.

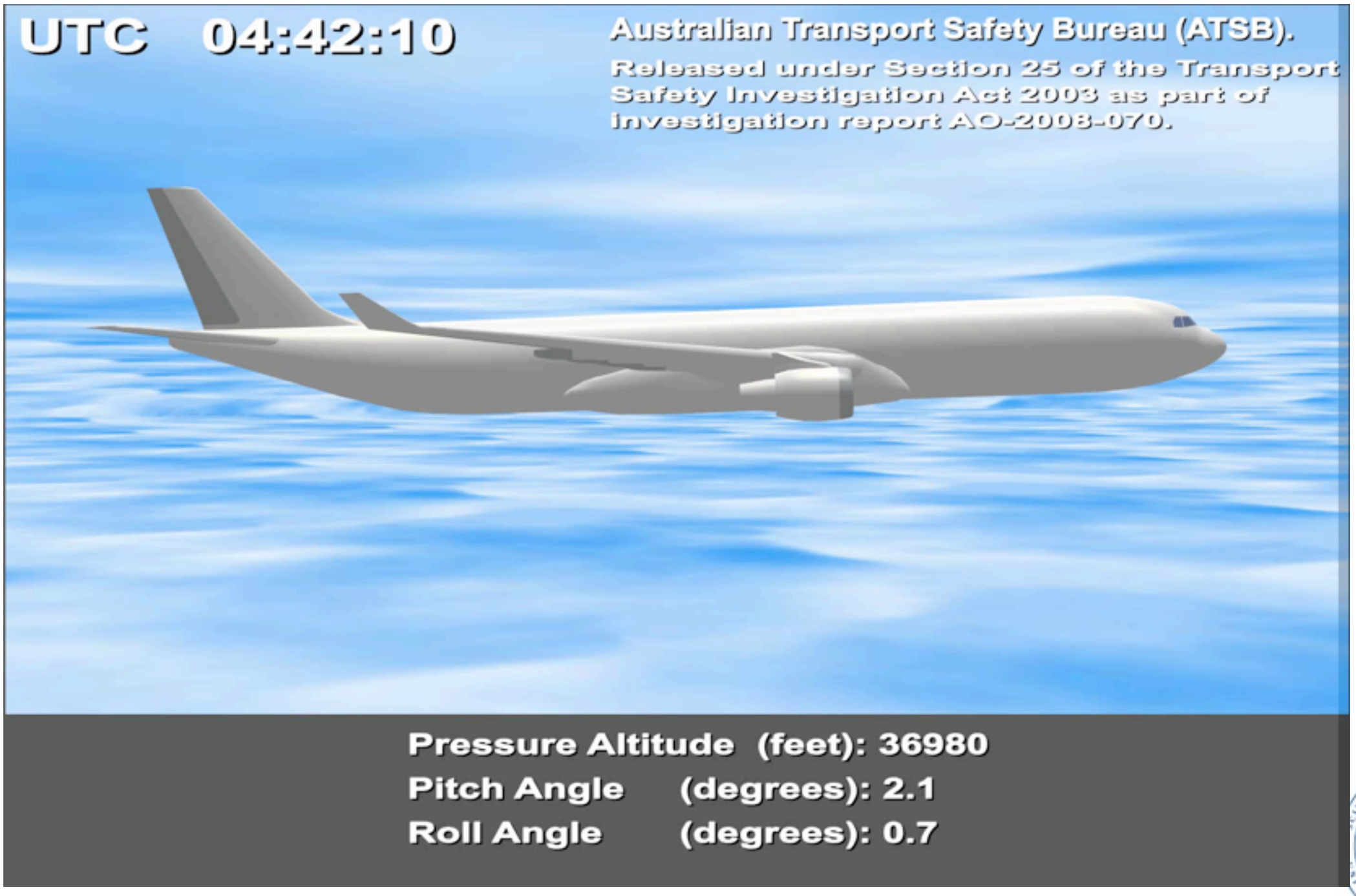
On 7 October 2008, an Airbus A330-303 aircraft, registered VH-QPA and operated as Qantas flight 72, departed Singapore on a scheduled passenger transport service to Perth, Western Australia. While the aircraft was in cruise at 37,000 ft, one of the aircraft's three air data inertial reference units (ADIRUs) started outputting intermittent, incorrect values (spikes) on all flight parameters to other aircraft systems. Two minutes later, in response to spikes in angle of attack (AOA) data, the aircraft's flight control primary computers (FCPCs) commanded the aircraft to pitch down. At least 110 of the 303 passengers and nine of the 12 crew members were injured; 12 of the occupants were seriously injured and another 39 received hospital medical treatment.

Source: Australian Govnmt Transport Safety Bureau
http://www.atsb.gov.au/publications/investigation_reports/2008/aair/ao-2008-070.aspx

Each of the intermittent data spikes was probably generated when the LTN-101 ADIRU's central processor unit (CPU) module **combined the data value from one parameter with the label for another parameter**.

Source: Australian Govnmt Transport Safety Bureau
http://www.atsb.gov.au/publications/investigation_reports/2008/aair/ao-2008-070.aspx

Example


UTC 04:42:10
Australian Transport Safety Bureau (ATSB).
Released under Section 25 of the Transport Safety Investigation Act 2003 as part of investigation report AO-2008-070.
Pressure Altitude (feet): 36980
Pitch Angle (degrees): 2.1
Roll Angle (degrees): 0.7

Verification and synthesis techniques

- Many formal verification techniques exist
  - They allow to prove (in a mathematical sense) that a software respects a given requirement
- Researchers have also proposed synthesis techniques to automatically build (parts of) software, in such a way that it respects given correctness criteria.

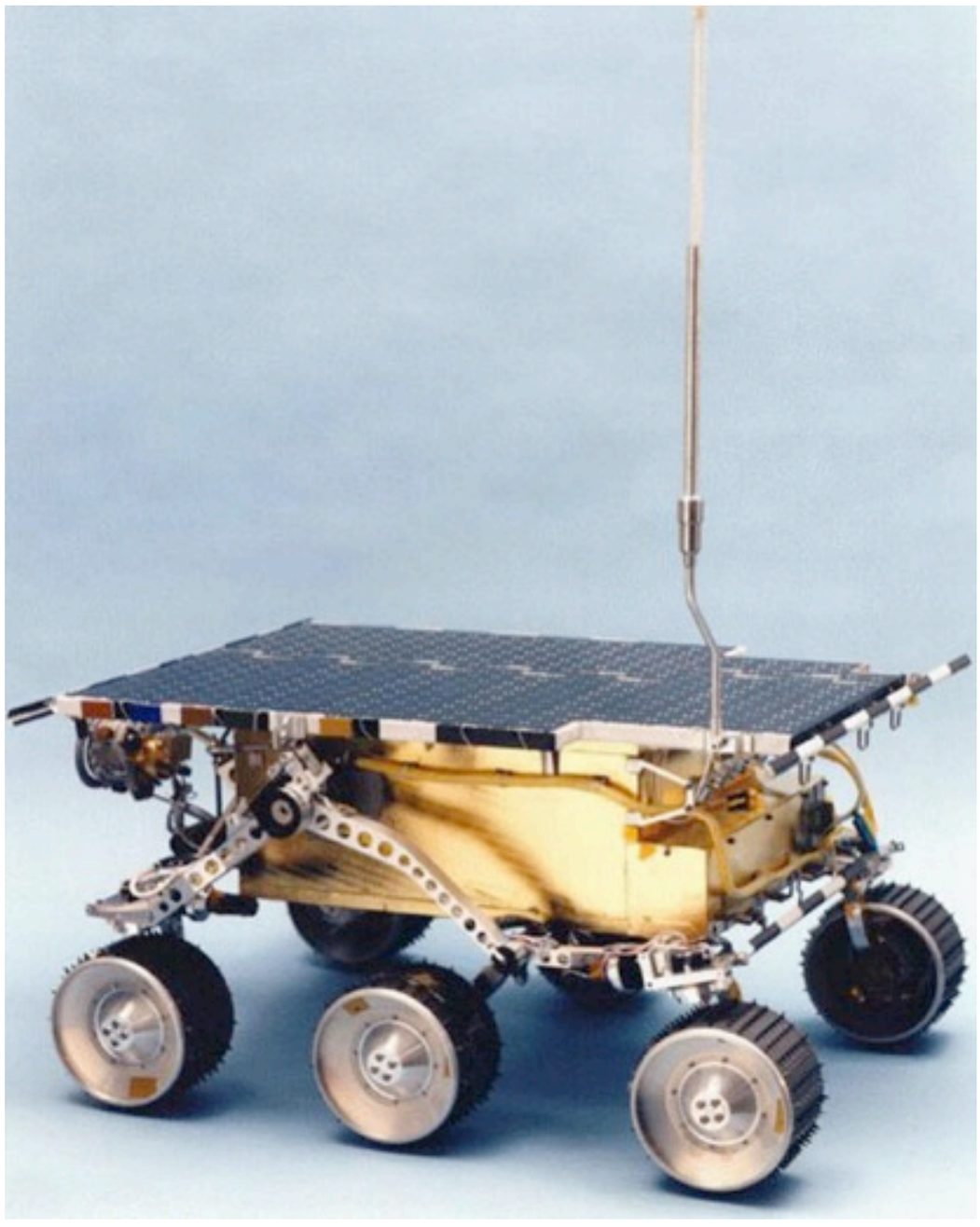
Real time

- Many ES should enforce real time constraints
  - Real time constraints are constraints about the response time of the system's tasks
  - When a task starts, it should produce results within a certain deadline, relative to its starting time.

Real time

- Example:
  - A plane autopilot has to send orders to the ailerons, to control the plane attitude
  - These orders have to «arrive on time»
  - In case of lateral wind gust, the autopilot has to compensate within 100 ms to ensure the aircraft stability
  - The tasks that computes the response to wind change has thus a maximal response time, that must be enforced, whatever the load of the compute system that runs it

Real time case study



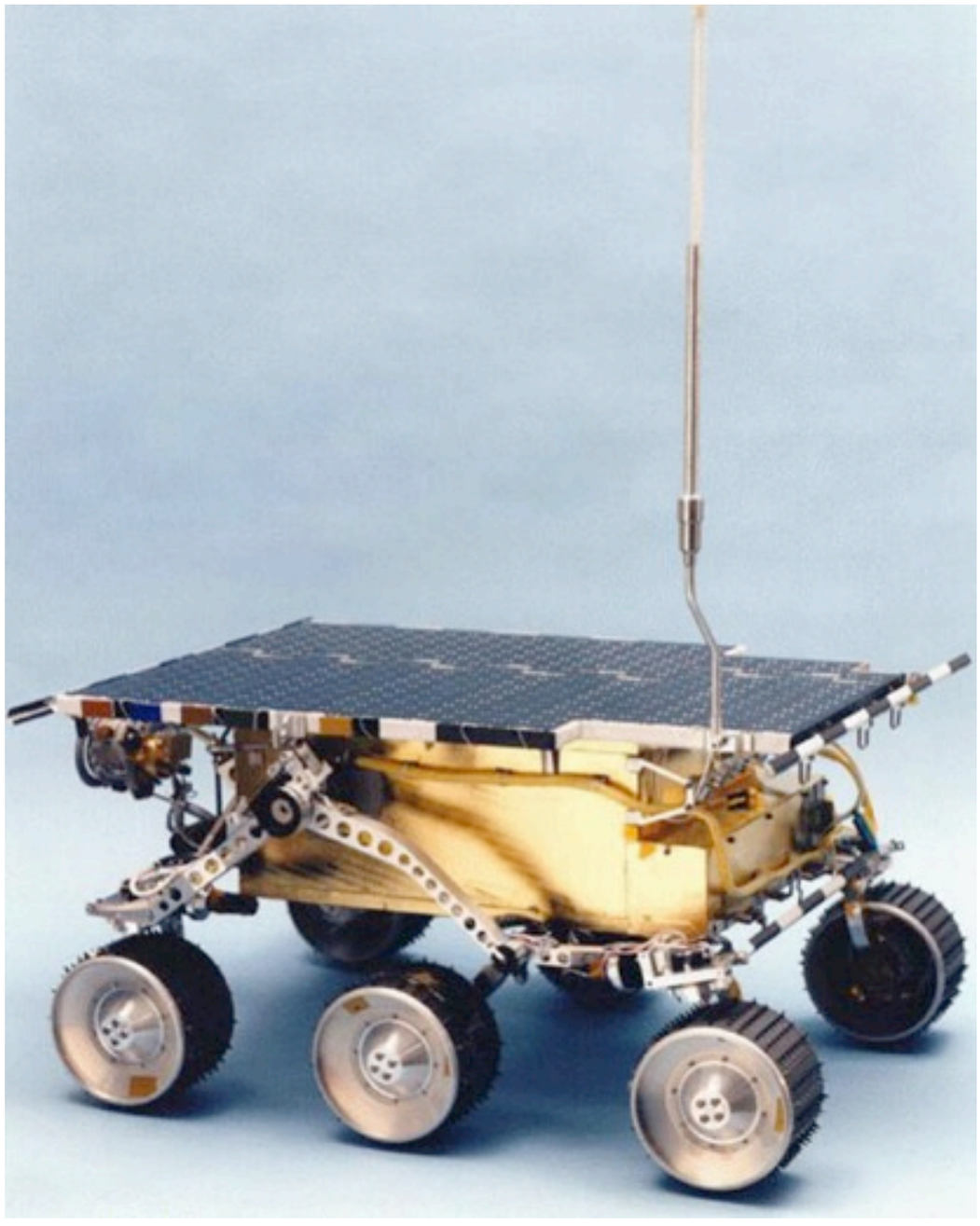
Sojourner

Robot sent to Mars by NASA in 1996 (pathfinder project)

Embedded computer with Intel 80C85 CPU (8 bits) and 512 Kb RAM

source: NASA

Real time case study

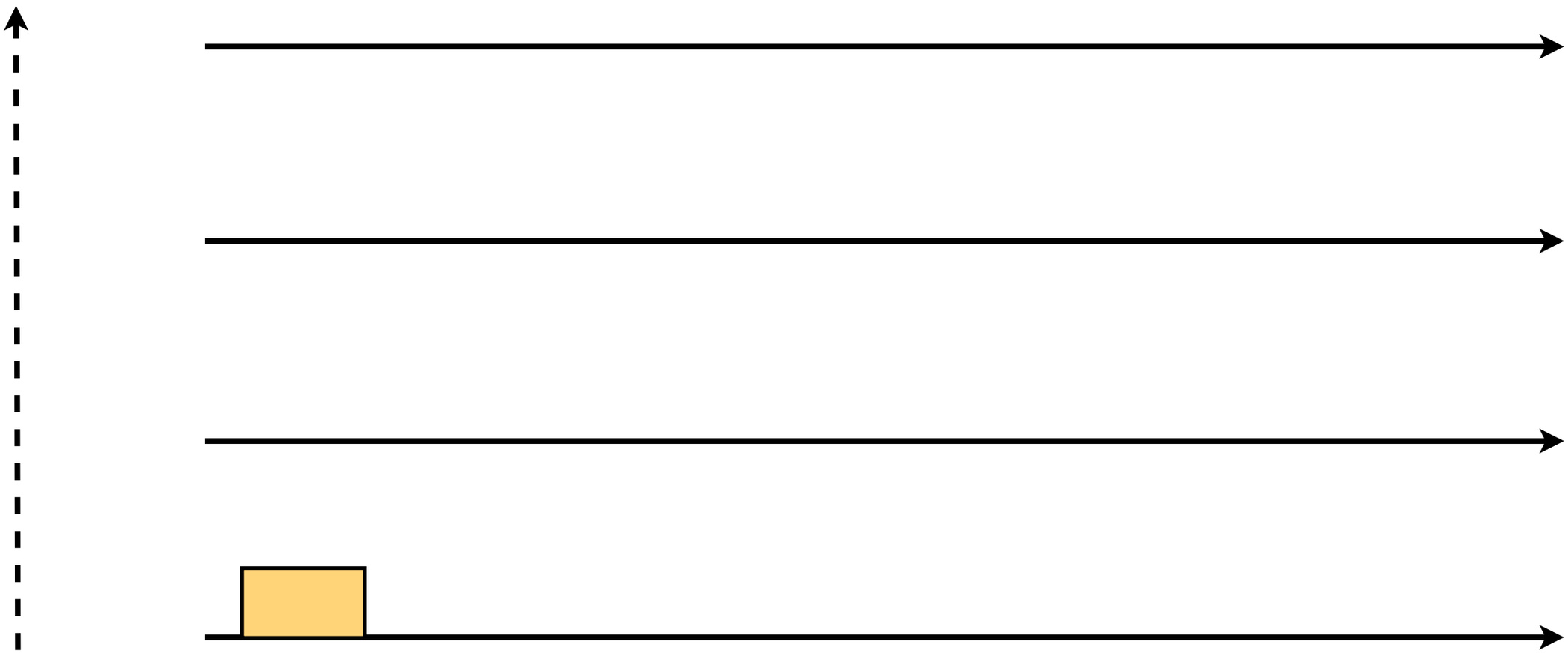


During the mission, this embedded system has suffered a priority inversion phenomenon

source: NASA

Real time case study

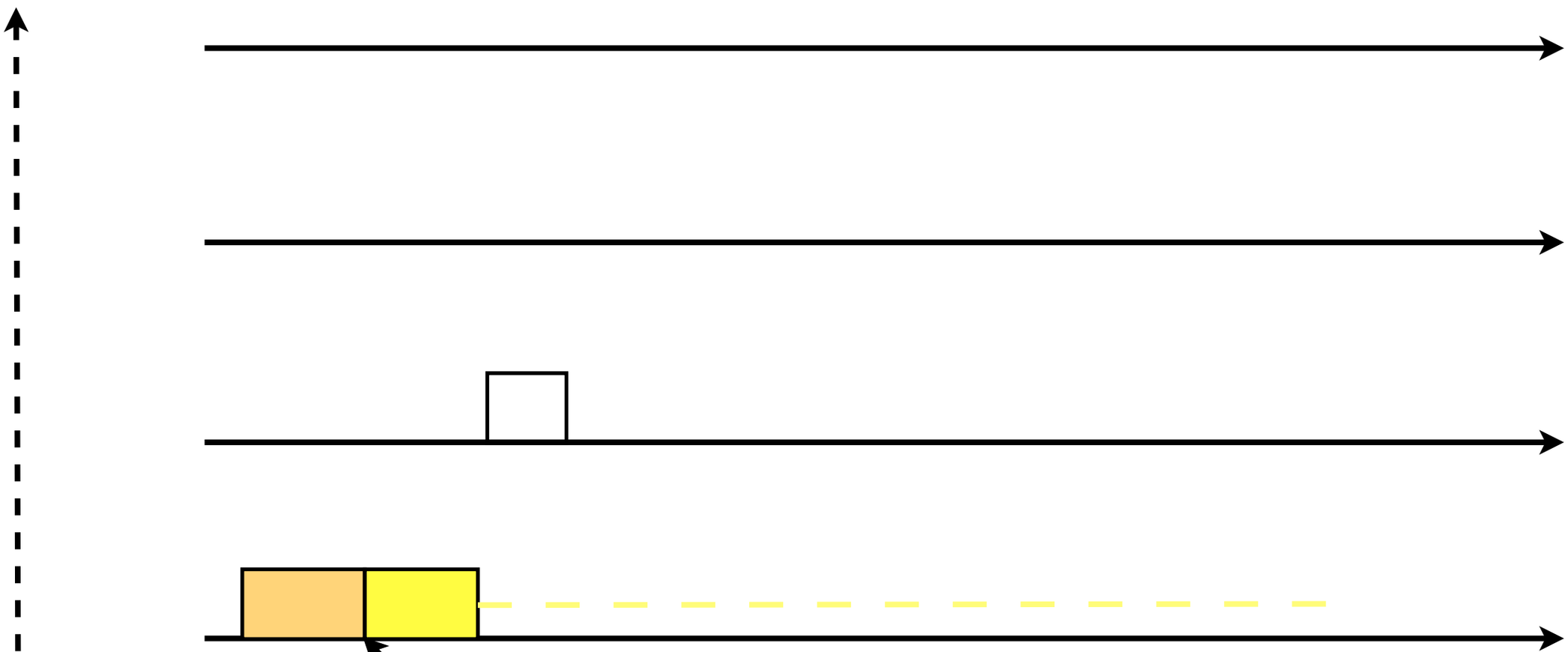
high prio.



low prio.

Real time case study

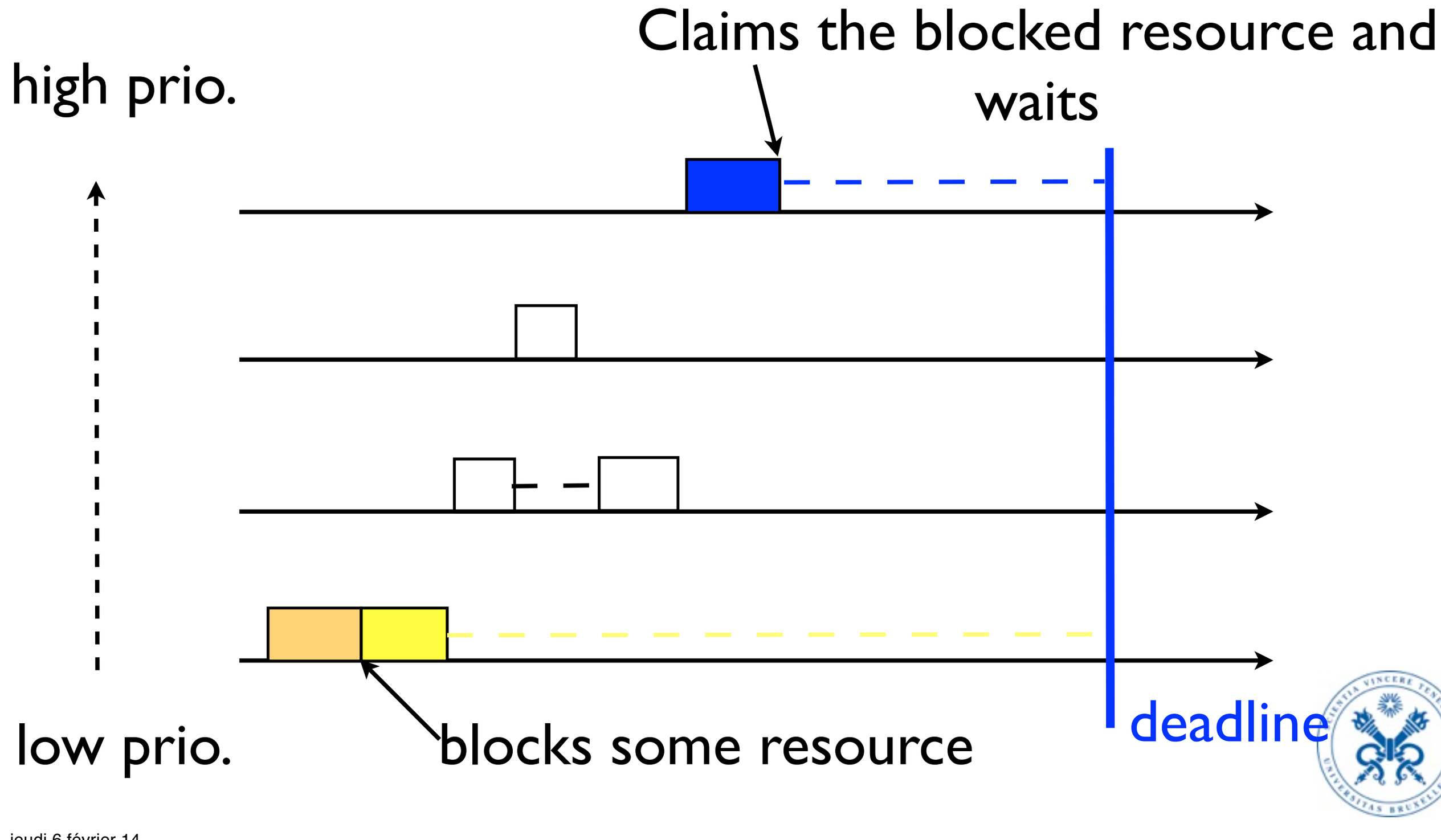
high prio.



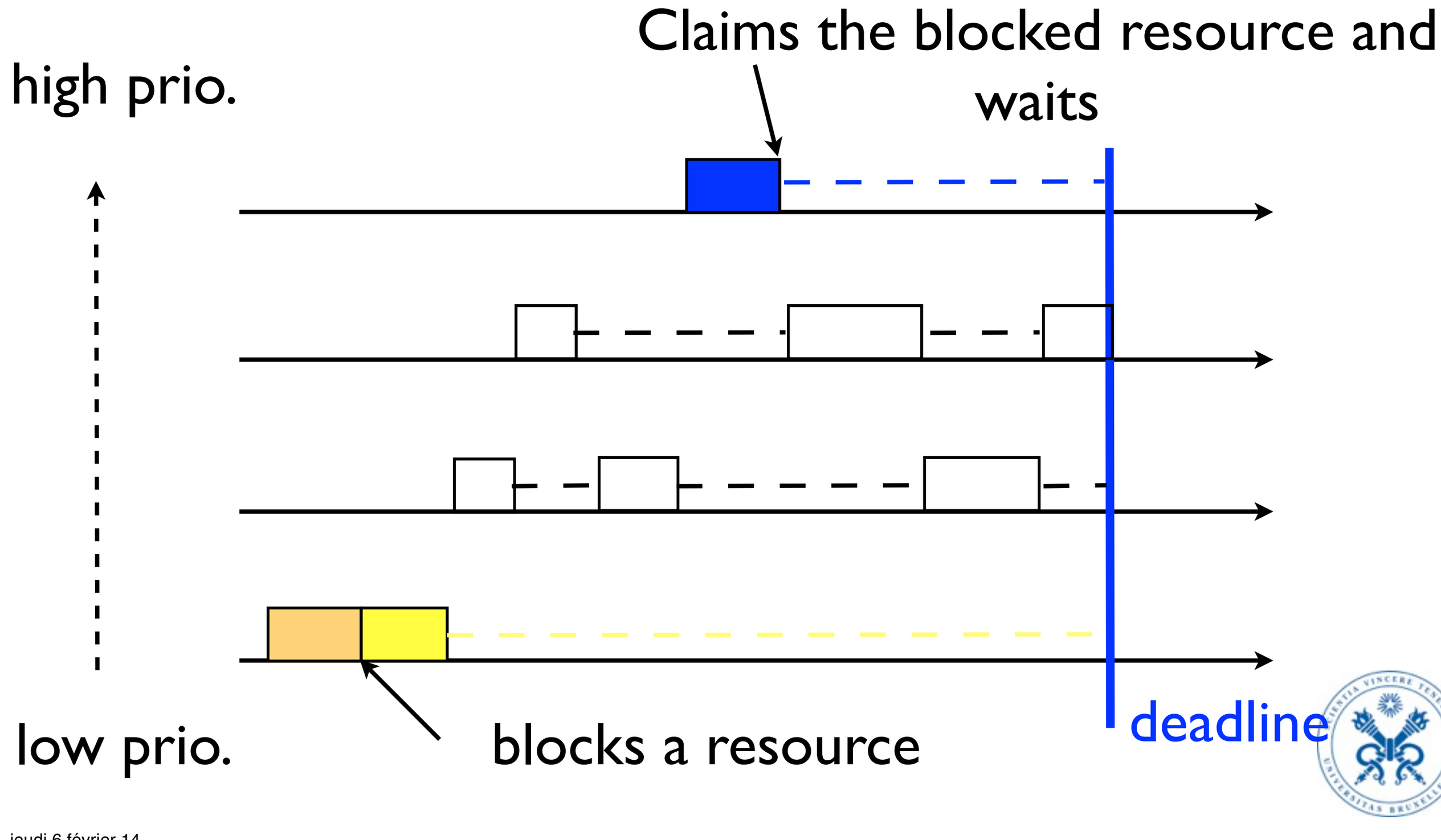
low prio.

blocks some resource

Real time case study



Real time case study



Security & privacy

- Some embedded systems manipulate sensitive data
- As more and more ES are more and amore open and powerful, they are also more exposed to attacks

Security & privacy

- Example:
  - A medical testing device is implemented on top of a Windows XP box
  - This device is connected to a database records the results of the tests, transmitting them via the local network of the hospital

Security & privacy

- Example:
  - For security (?) reasons, the user is forbidden from accessing the software underlying the system. That means that no security updates can be applied to the OS...
  - The device is compromised by a hacker who gains access to the whole database
  - See: http://www.networkworld.com/weblogs/security/005694.html

Security & privacy

- Example (2):
  - STIB has recently introduced a new ticketing system called MOBIB
  - It is base on a RFID card that can transmit data from short distance
  - Reading the card content is thus possible without any physical contact between the card and the reader

Security & privacy

C.R.I. COM (2007-2008) N° 108

I.V. COM (2007-2008) Nr. 108

**PARLEMENT
DE LA RÉGION DE
BRUXELLES-CAPITALE**

**BRUSSELS
HOOFDSTEDELIJK
PARLEMENT**

**Compte rendu intégral
des interpellations et
des questions orales**

**Integraal verslag
van de interpellaties en
mondelinge vragen**

•••

Les trajets ne sont pas enregistrés au niveau de la carte Mobib, mais ils le sont au niveau du valideur, qui mémorise l'heure, la date et la ou les lignes utilisées. Ceci permet à la STIB, après remontée de l'ensemble des données de validation, d'ajuster de manière optimale l'offre à la demande. Il s'agit aussi d'une garantie permettant d'éviter les vérifications d'itinéraire par les contrôleurs. Les données anonymes sont ensuite généralisées dans le système.

Ministry of transport statement

Security & privacy

C.R.I. COM (2007-2008) N° 108 I.V. COM (2007-2008) Nr. 108

Les trajets ne sont pas enregistrés au niveau de la carte Mobib, mais ils le sont au niveau du valideur, qui mémorise l'heure, la date et la ou les lignes utilisées. Ceci permet à la STIB, après remontée de l'ensemble des données de validation, d'ajuster de manière optimale l'offre à la demande. Il s'agit aussi d'une garantie permettant d'éviter les vérifications d'itinéraire par les contrôleurs. Les données anonymes sont ensuite généralisées dans le système.

manière optimale l'offre à la demande. Il s'agit aussi d'une garantie permettant d'éviter les vérifications d'itinéraire par les contrôleurs. Les données anonymes sont ensuite généralisées dans le système.

Ministry of transport statement

Security & privacy

- A team of researchers at UCL has analyse the content of a MOBIB card:

Due to the lack of clear information around the MOBIB, the Information Security Group (GSI) decided to analyzed the content of the MOBIB. It showed that the card contains among others the following data:

- First and last name of the holder.
- Birthdate of the holder.
- ZipCode of the holder.
- Last three validations of the holder (date, time, bus line, bus stop, subway station, etc.)
- Some other technical data

All these data are stored without encryption !

source: http://www.uclouvain.be/sites/security/mobib.html

Security & privacy

- A team of researchers at UCL has analyse the content of a MOBIB card:

Due to the
Informatio
the MOBI
following c
• First and
• Birthdate
• ZipCode
• Last thre
subway sta
• Some ot

ntent of
e
stop,

Quoting STIB's spokeperson:
«Il y a des données qui peuvent apparaître en clair, à condition d'avoir les logiciels et le matériel approprié, mais ce n'est pas en clair, il faut avoir un terrible matériel.»
http://bugbrother.blog.lemonde.fr/2009/01/13/la-bonne-blague-de-la-ratp-belge/

All these data are stored without encryption !

source: http://www.uclouvain.be/sites/security/mobib.html

MOBIB

see: http://www.youtube.com/watch?v=P-YQ6wT0Y48

RFID

- Many controversies have emerged around the introduction of RFID technology, for several different reasons:
  - Information stored in the chip are most often unencrypted. Chips that implement security mechanisms exist but are more expensive.

RFID

- Many controversies have emerged around the introduction of RFID technology, for several different reasons:

- The user is usually not aware of the content of the chip. Moreover, deactivating a chip a making it unreadable is not easy.

RFID

- Problems related to the weak cryptographic power of RFID tend to limit their applications.
- However many governments want to introduce RFID chips in id. documents to "raise security levels"
- Many of these systems have already been compromised and it turns out that it is easier to copy and RFID chip than a traditional paper-based passport.

RFID

source: http://freeworld.thc.org/thc-epassport/

Metrics

- Several constraints that have an impact on the implementation choices can be quantified
  - they are called metrics
- It is a rather common concept in software engineering

Metrics - examples

- Non recuring costs: are the development costs that are not repeated every time an unit of the product is manufactured
- Unit cost: The cost of manufacturing one unit of the product (without non-recuring costs)
- Size
- Performance: several measures can be used (response time, flops,...)

Metrics - examples

- Energy consumption
- *Time to prototype*: Time needed to build a working prototype
- *Time to market*: Time needed to develop a system so that it can be released on the market and sold

Metrics

- Other constraints such as flexibility, safety, security, correctness, etc... are hard to quantify and we do not include them in the metrics.

Metrics - contradictions

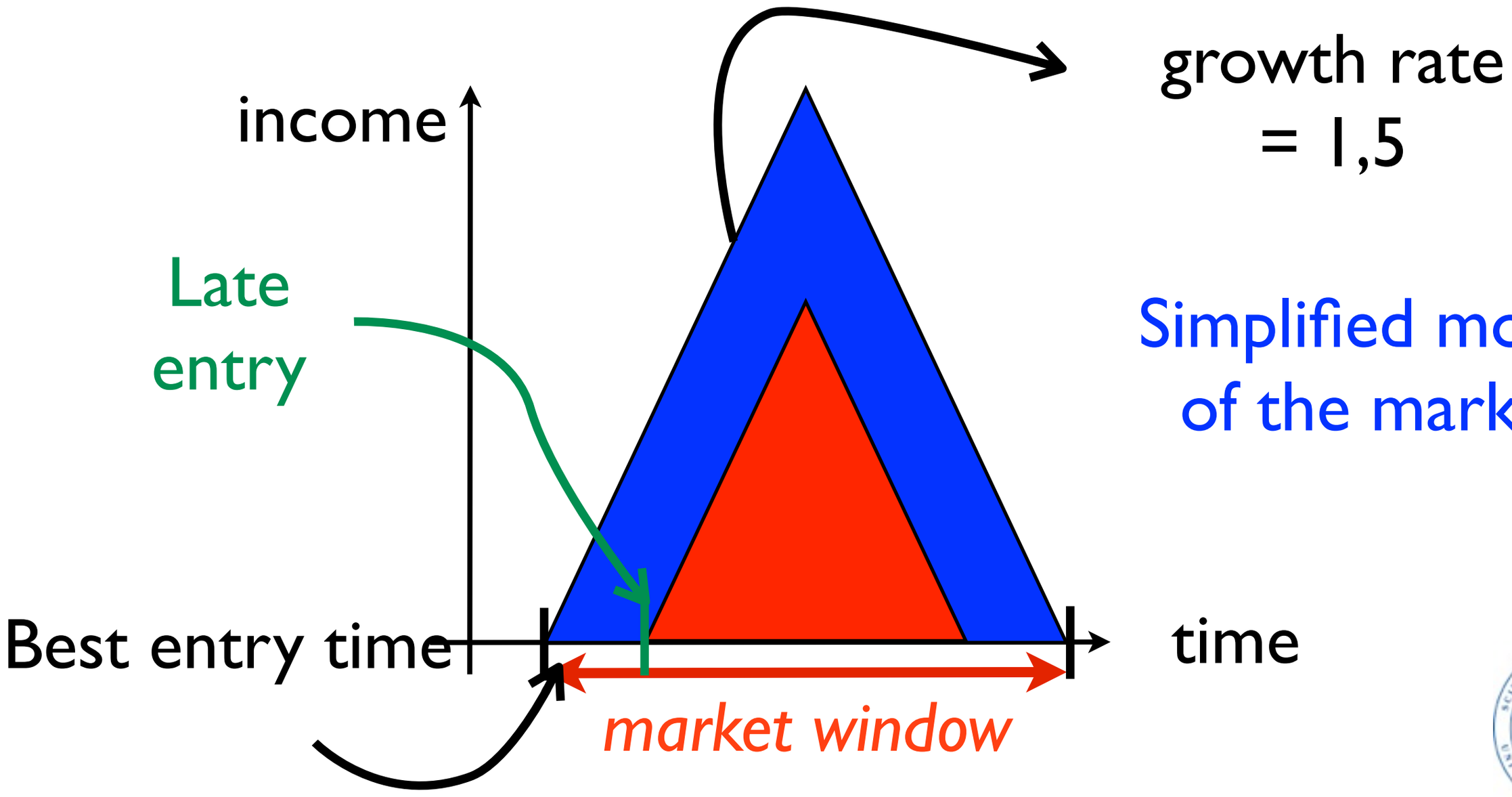
- Some metrics compete with each others
- For instance, with a lower investment in development, one lowers the non-recurring costs, but might augment the *time to market*...

Time to market

- This metric is most important from the economical point of view, in particular in the consumer electronics market
- It is related to the best time to introduce a product on the market:
  - too soon, the product will not meet a customer demand (eg: Newton d'Apple) and will not sell
  - too late, the competitors will surely occupy the market already, and the product will yield poor revenue

Time to market

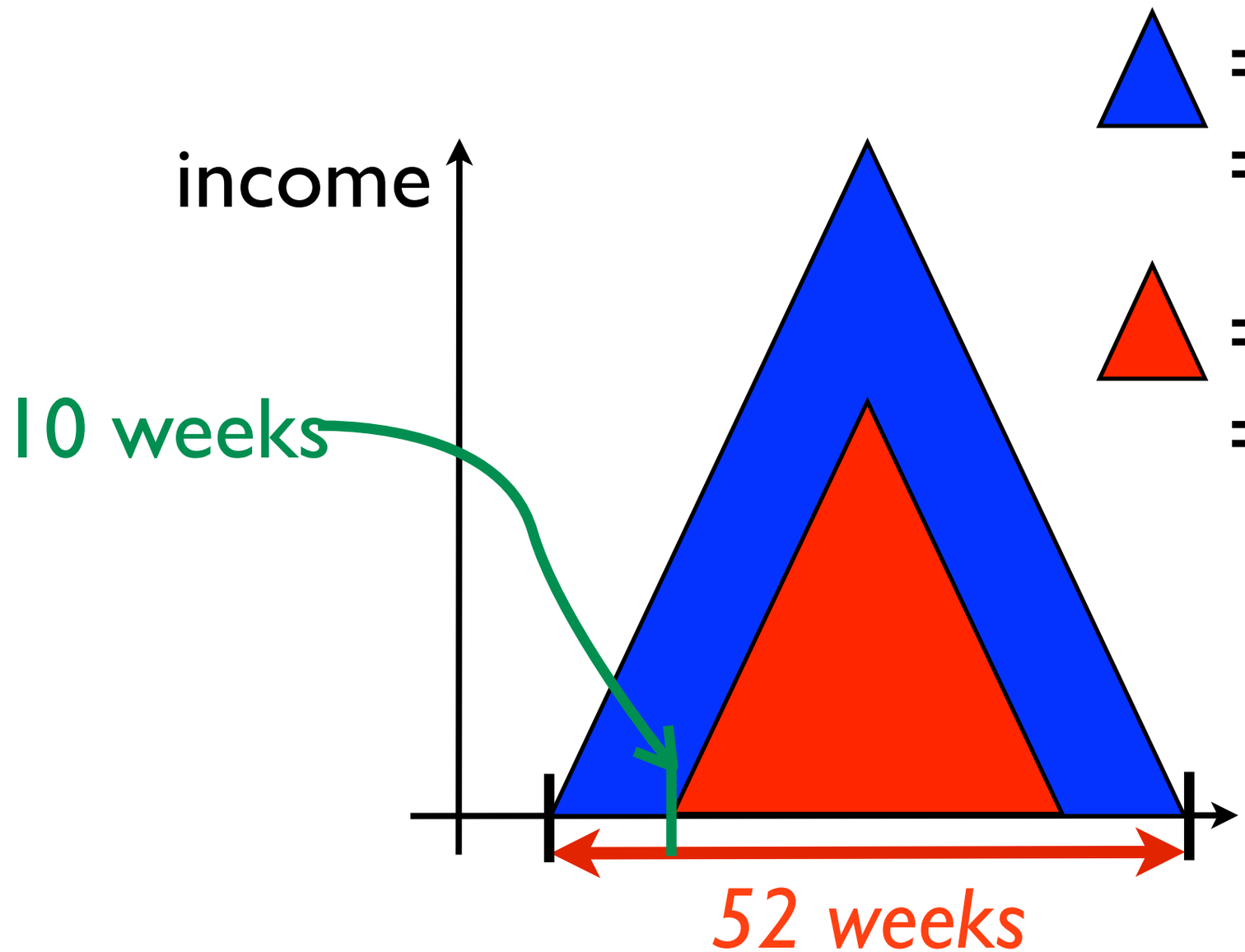
- Delays might have a big impact



Simplified model of the market

Time to market

- Delays might have a big impact



= (52 * 26 * 1,5)/2
=1014

= (32 * 16 * 1,5)/2
=384

A 60% loss !!!

Costs

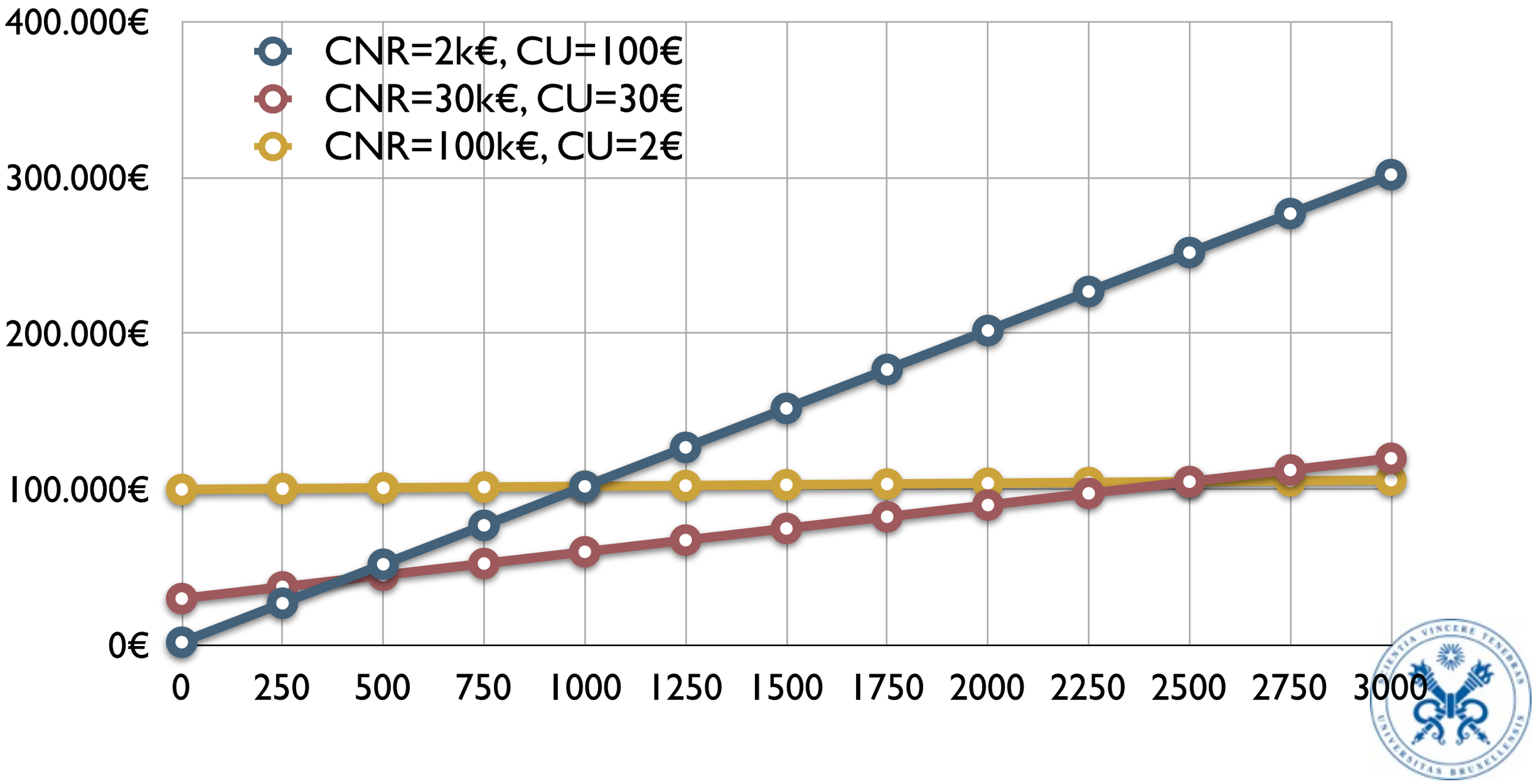
- Costs metrics also have an impact on development choices
  - Most often non-recuring costs and units costs will compte with each other:
    - Usually, a larger investment in the development yields an easier and cheaper to produce system

Costs

- Compromises between different choices must often be done to minimise the (total cost)/unity value
  - this value is the most important to determine the benefit
  - (total cast) / unity =
    (NRC + number produced * UC)/
    number produced
    where NRC = non-recurring costs
    and UC = unit cost

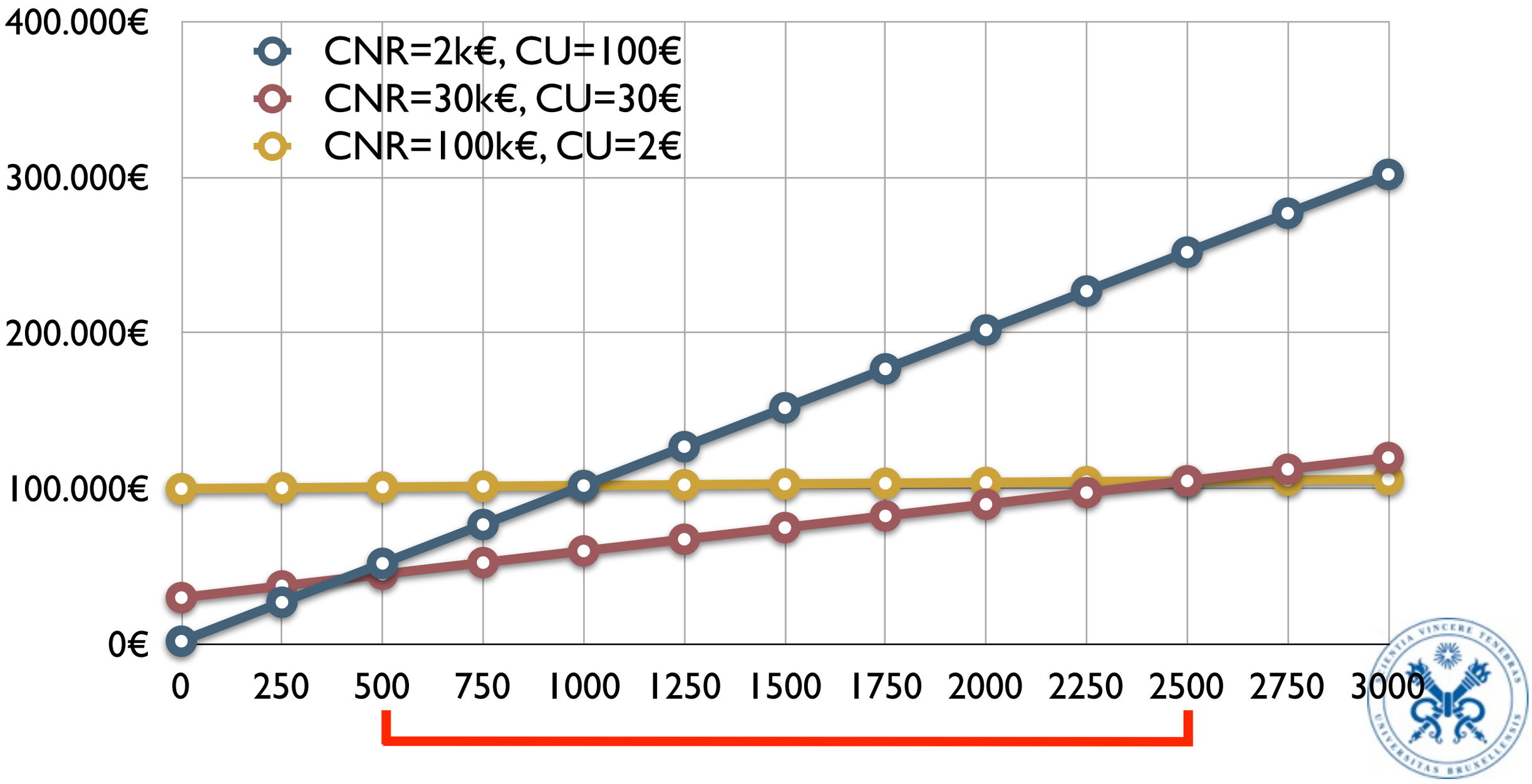
Costs

Example: three different technologies
Total cost vs number of units



Costs

If a production of 500 to 2500 units is planned technology n°2 yields the lowest unit cost



Implementation

Which choices ?

- An embedded system is made up of *software* and *hardware*
- A given feature of the ES can often be implemented either by *software*, or by *hardware*
- Deciding what to implement in hardware and what in software is called partitioning

Hardware / Software

- Example:
  - We want to design a digital camera with that can:
    - save the pictures as JPEG files
    - display a GUI (menus) to set up the camera parameters

Hardware / Software

- JPEG compression can be implemented as a specialised hardware (chip):
  - it receives the raw data from the sensor
  - it produces a compressed version of the picture

Hardware / Software

- JPEG compression can be implemented as a specialised hardware (chip):
  - Advantages:
    - Quick
    - Such chips are readily available from manufacturers
  - Drawbacks:
    - not flexible

Hardware / Software

- The GUI is best implemented as a piece of *software*
  - We can use a high-level language to program the system, and rely on existing libraries to generate the menus
  - The camera contains an all-purpose CPU that executes this program and communicates with the different components of the camera

Hardware / Software

- The GUI is best implemented as a piece of *software*
- Flexibility maters most for the GUI
- Speed is not so important here as long as it stays reasonable

Hardware / Software

- Once the partioning *hardware/software* is completed, several questions remain:
  - How to realise the *hardware* ?
  - How to execute the *software* ?

Hardware

Memory

- OTP ROM, EPROM, EEPROM, Flash memory, RAM,...
- Let's tour these different technologies !

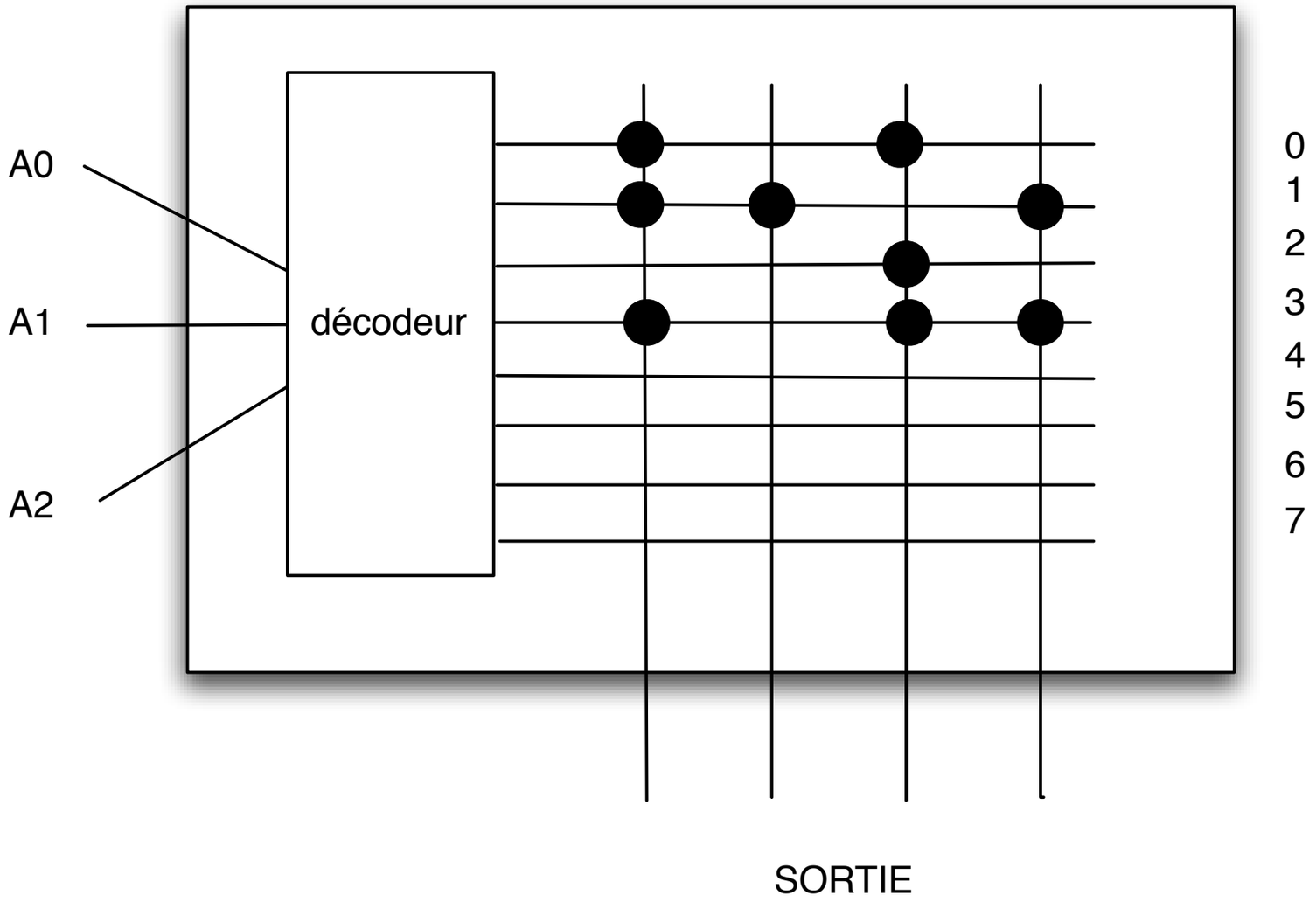
ROM

- ROM = Read Only Memory
- Data in ROM cannot be modified and is kept when the computer is turned off
  - *cannot be modified*: should read «cannot easily be modified»

ROM

● = connexion
= bit = 1

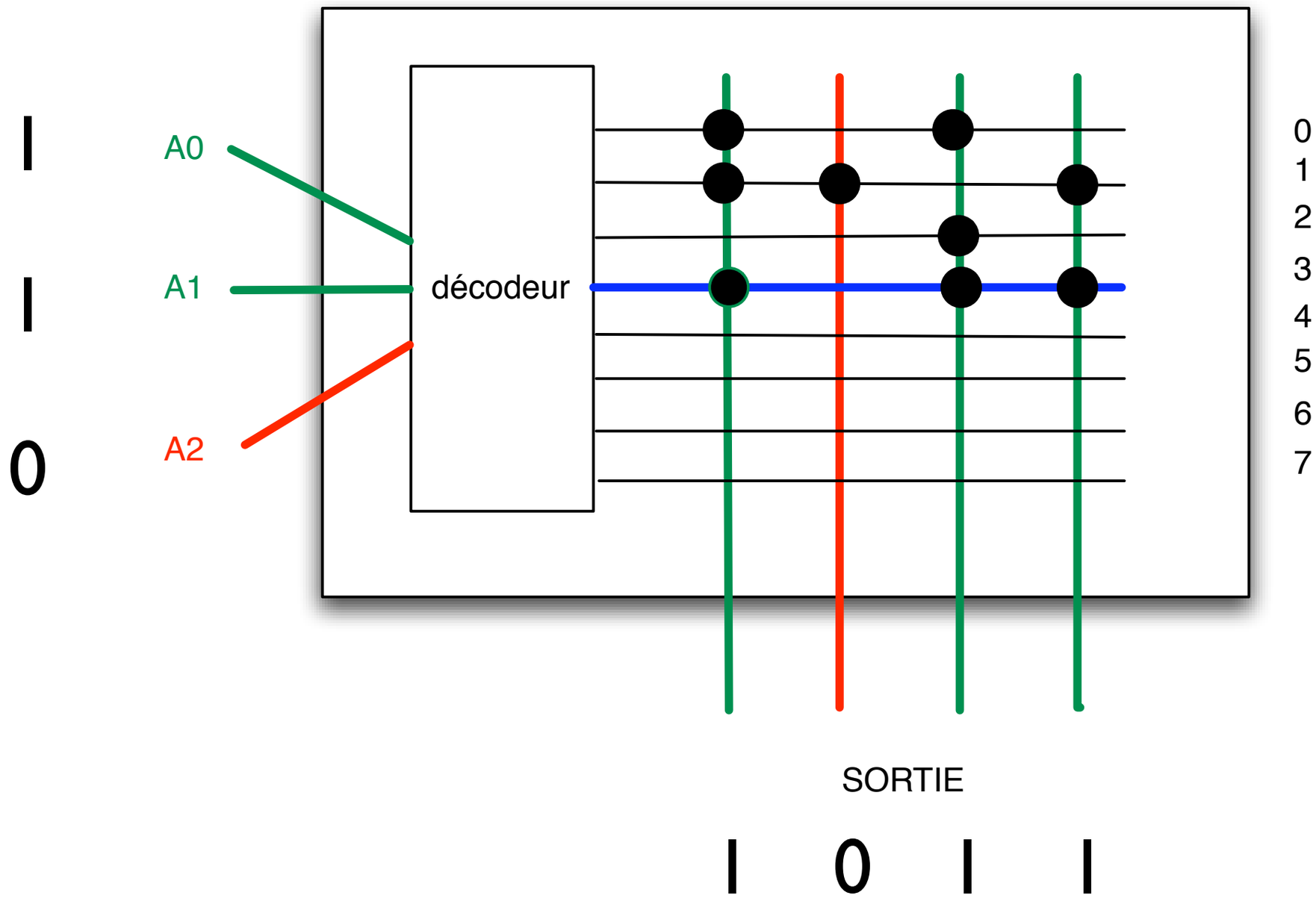
Address



1 word per line

Example: contains 8 words of 4 bits (3 bits addresses)

ROM

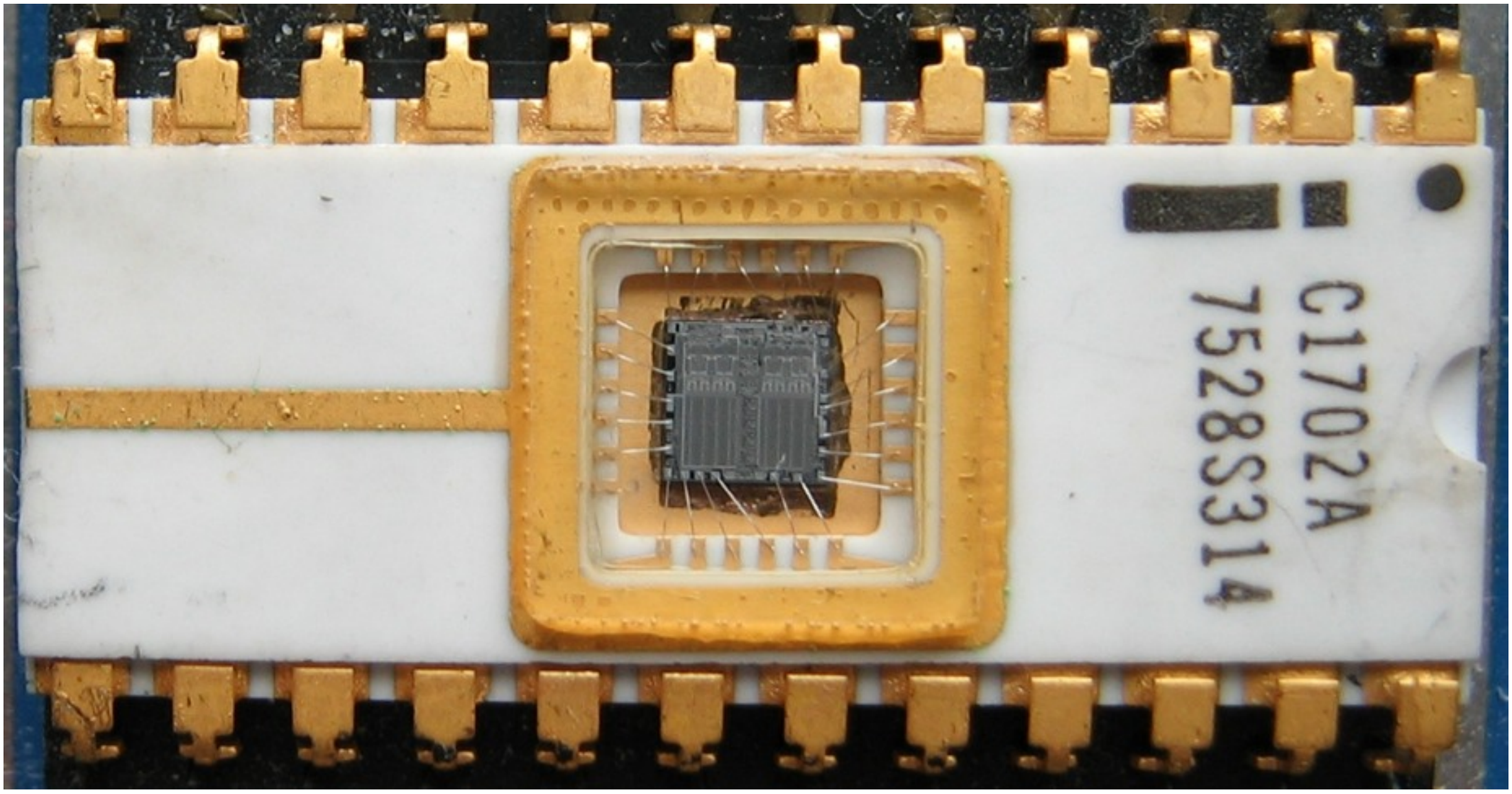


Example: contains 8 words of 4 bits (3 bits addresses)

ROM

- How actual connections are realised depends on the technology:
  - OTP ROM = *One-Time Programmable ROM*: Connections are made "*in silico*" by the manufacturer. No way to modify the data
  - EPROM = *Erasable Programmable ROM*: connections are carried out by applying a high voltage (25 volts). EPROMS can be erased by exposing then to ultraviolet light

EPROM



source: wikipedia

EEPROM

- EEPROM = *Electrically Erasable Programmable ROM*: as an EPROM but erasing can be done electrically (early '80s)
- l'EEPROM can be used instead of classical RAM provided the erasing circuit is included in the device
- Drawback: erasing (and writing) are slow

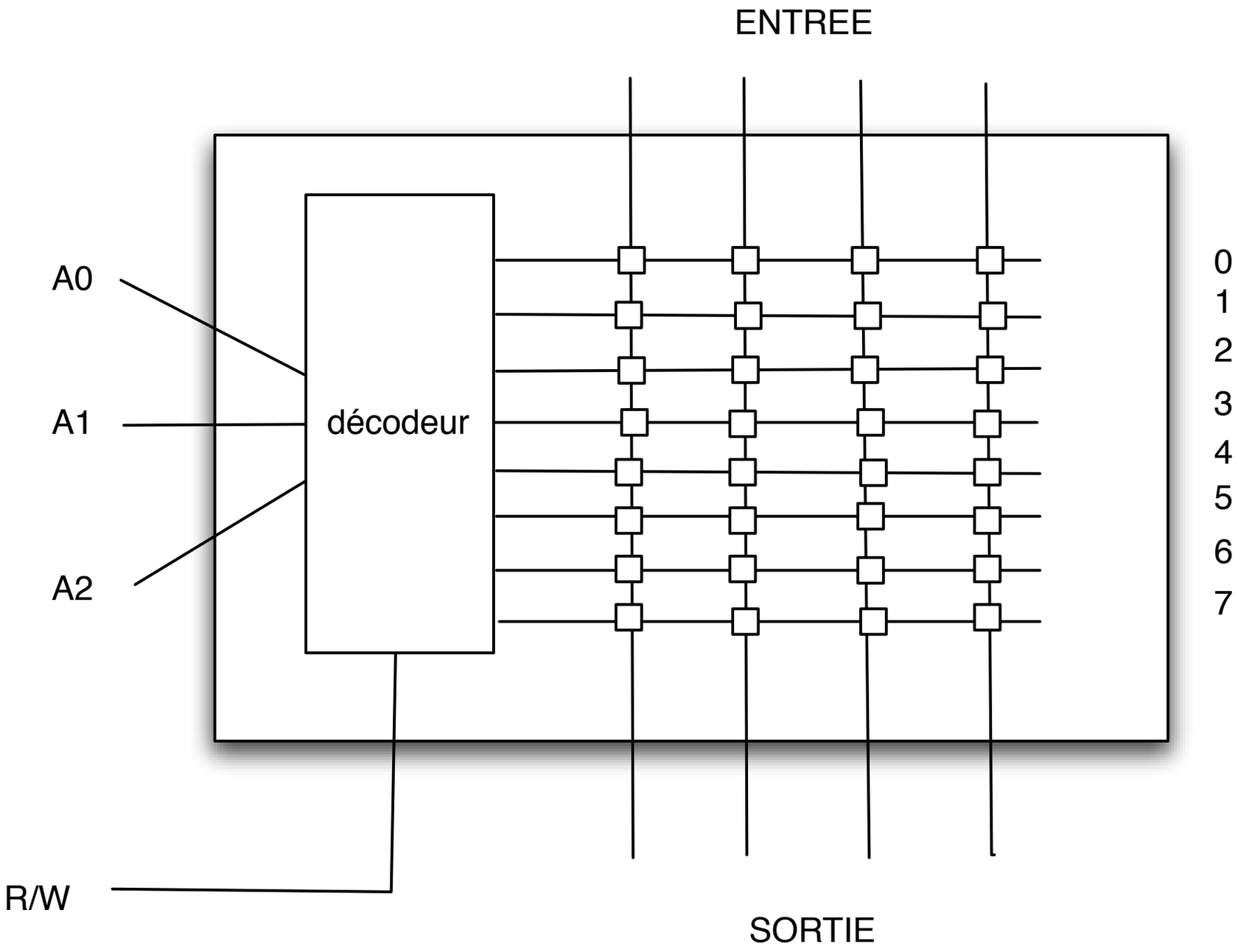
Flash memory

- Follow the same principle as EEPROM but erasing is quicker
- To maintain efficiency, large blocks are erased at a time
- Drawback: still slow when writing blocks of data that are much smaller than the block size
  - Can be solved with an efficient caching policy

ROM vs RAM

- Why are Flash memory and EEPROM regarded as ROM ?
  - Although writing and reading are possible, they are not symmetrical: reading is more efficient and easier than writing
- In RAM, writing and reading are symmetrical

RAM



□ = 1 cell = 1 bit

Hardware

- How to implement a feature of the system as hardware ?
  - easy way: re-use some ready-made chip...
  - depending on the feature, this is not always possible !

Custom hardware

- Three main technologies exist to build custom integrated circuits:
  - VLSI : the IC is built from transistors
  - ASIC : the IC is built from pre-defined blocks
  - PLD = programmable IC

VLSI

- VLSI = *Very Large Scale Integration*
  - The designer draws a map that specifies the position of each transistor and its connections to the others
  - Drawing the map is done in three steps:
    - Placement of the transistors
    - Routing of the connections
    - Choosing the size of the IC

VLSI

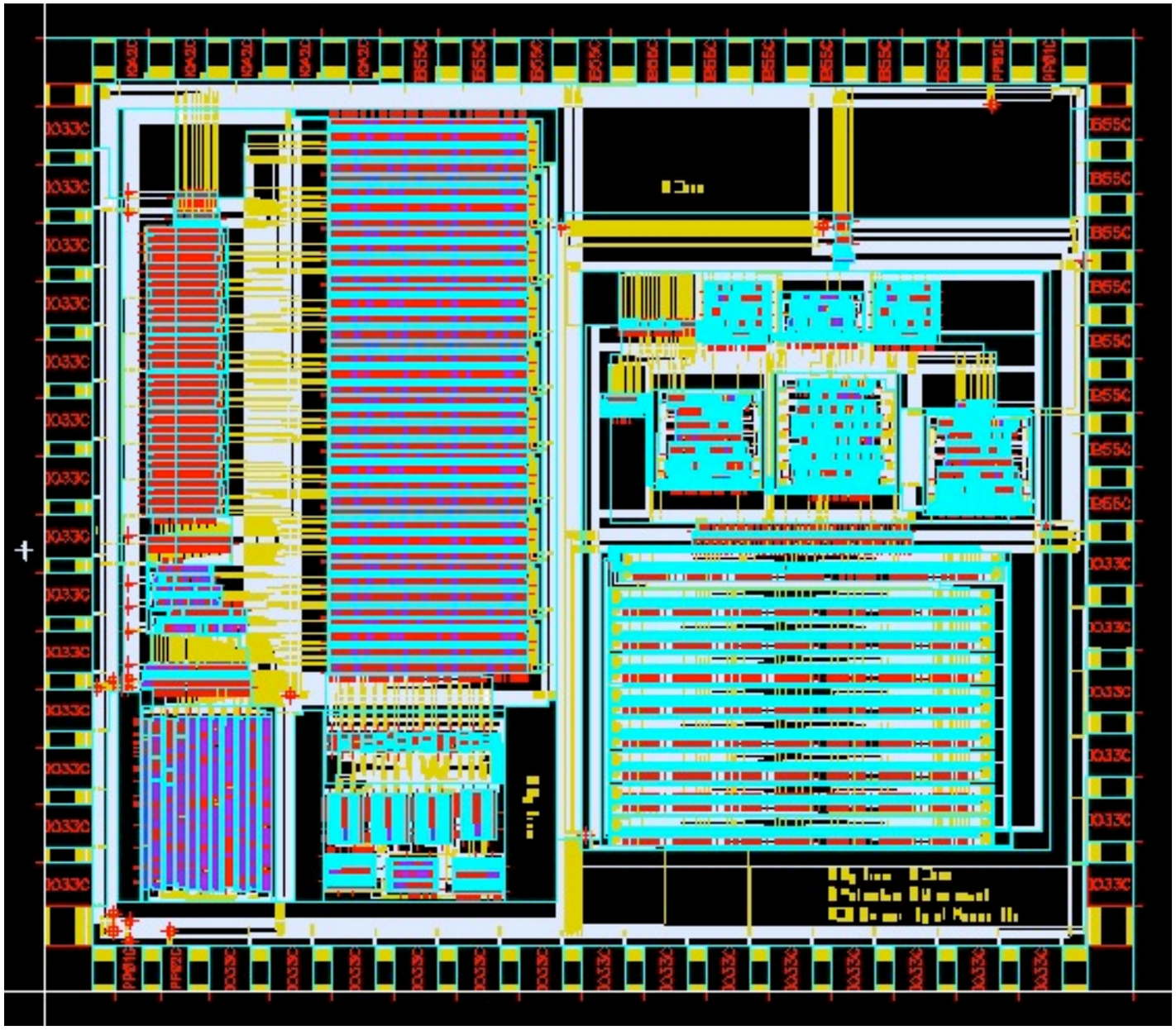
- Designing a VLSI is a long, costly and complex procedure. It must account for several physical constraints (heat, electrical issues,...)

- *In fine*, the designer send to the manufacturer masks describing each layer of silicon and copper.

- The NRC is very high, but the unit cost is very low

VLSI

Mask example



source: http://www.staff.ncl.ac.uk/alex.yakovlev/home.formal/

VLSI

- Advantages:
  - Unit cost very low
  - Allow very compact designs
- Drawbacks:
  - NRC and time to market very high
  - Requires non-trivial engineering skills
  - Avoiding mistakes in the design is of the utmost importance !

ASIC

- ASIC = *Application Specific Integrated Circuit*
  - The designer works with an IC that already contains logic components
  - Designing an ASIC consists in properly connecting the pre-existing components to obtain the desired functionality

ASIC

- Depending on the ASIC model, the logical components can be of different types:
  - Logic gates: AND, OR...
    - More advanced logic functions called «cells»:
      - NAND, NOR, multiplexer
    - Sometimes the designer can even chose the position of the cells on the ASIC

ASIC

- Advantages:
  - Unit cost low
  - Easier to design than a VLSI, while still offering great flexibility

- Drawbacks:
  - NRC and time to market still high
  - Important engineering skills still required
  - Avoiding mistakes in the design is of the utmost importance !

PLD

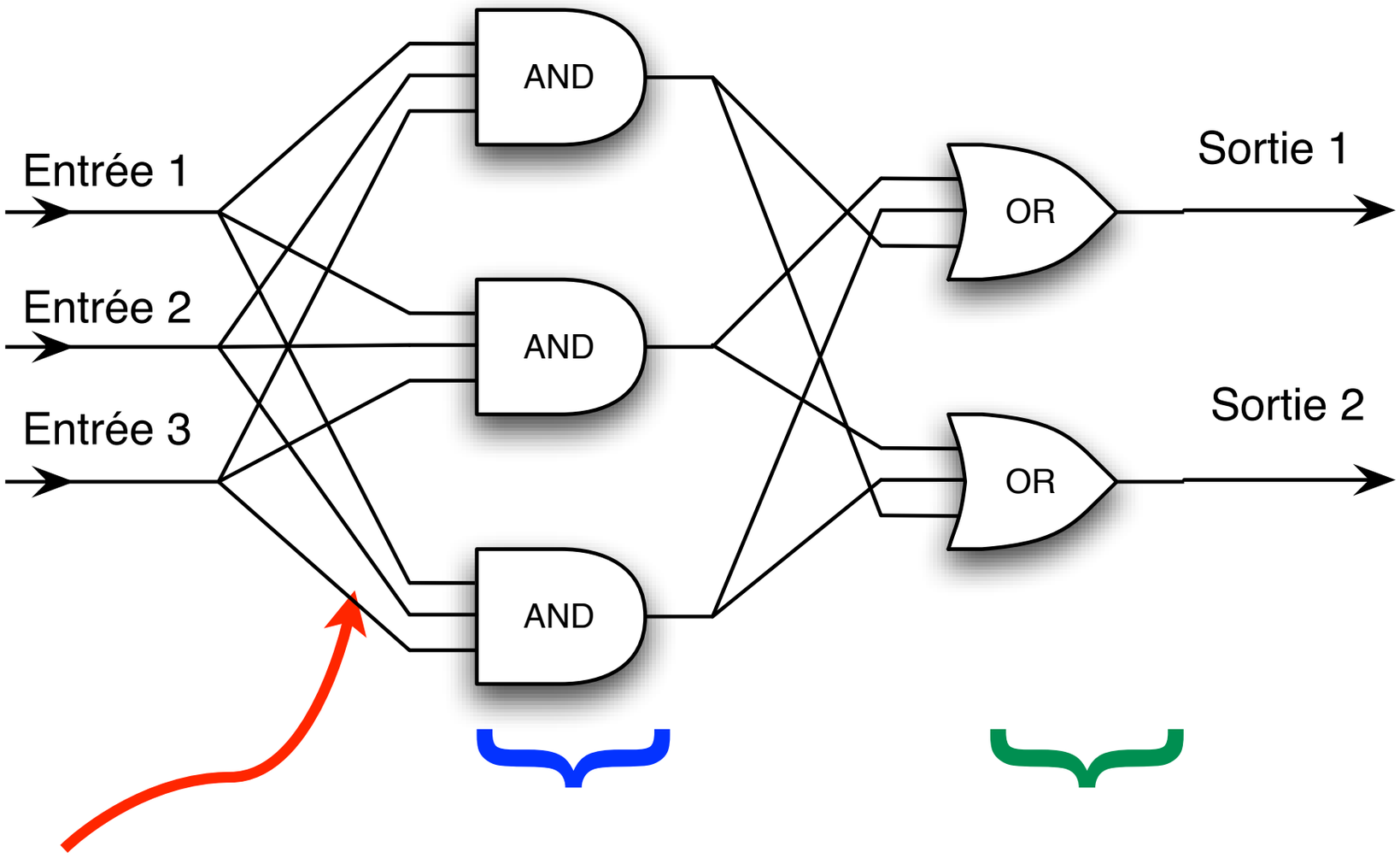
- An important drawback of VLSI / ASIC is that they are hard to adjust once the system has been designed
  - Trial-and-error is definitely not the good methodology !
    - Simulators exist but their reliability is questionable

PLD

- The PLD (*Programmable Logic Device*) technology allow to re-program the IC «on the field»
- One buys the PLD «as is» and then configure it to obtain the desired functionality

PLD

Example of PLA (= simplest PLD)

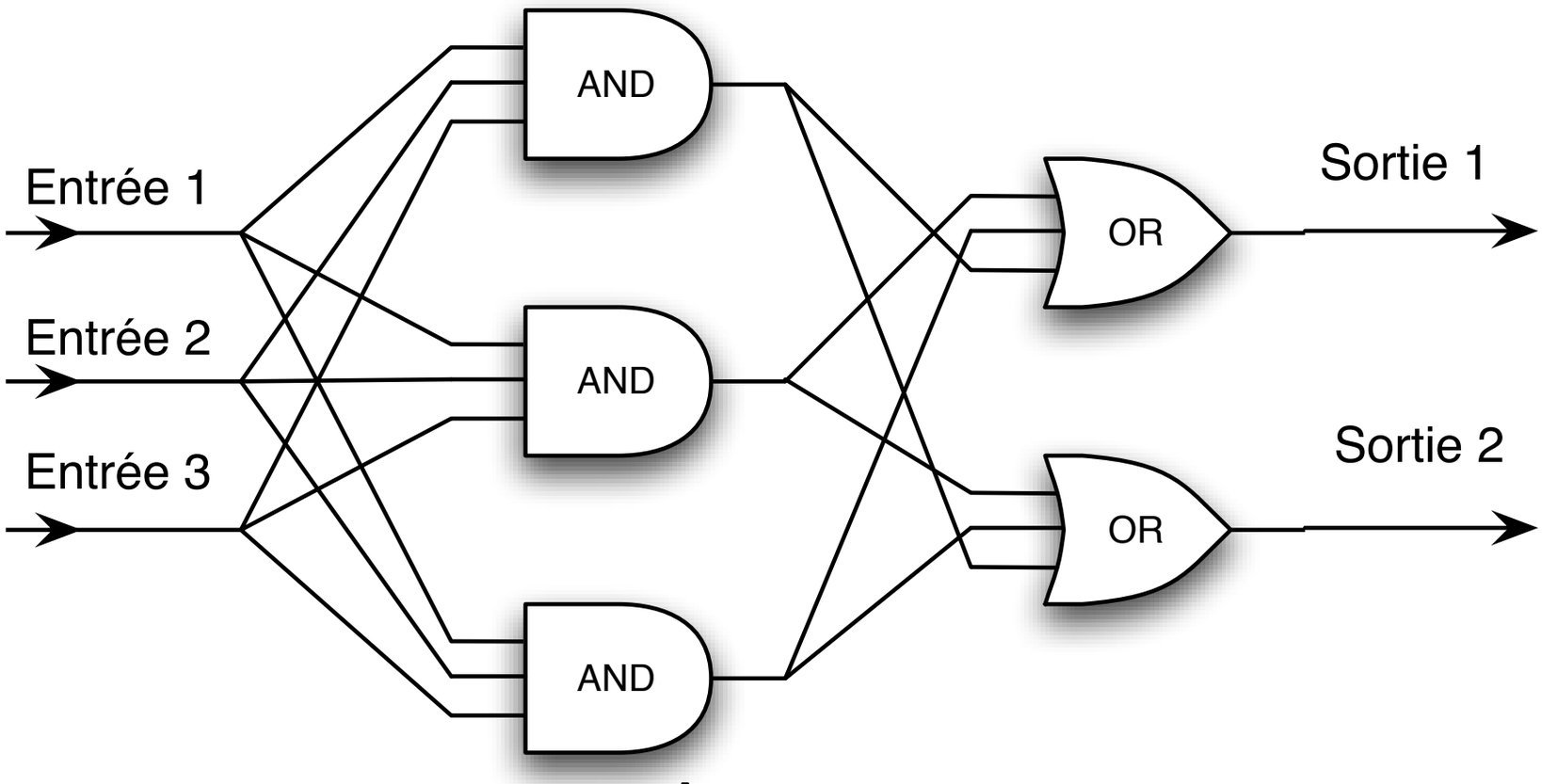


Each link can be cut off

Stage 1
AND gates

Stage 2
OR gates

PLD



As is:

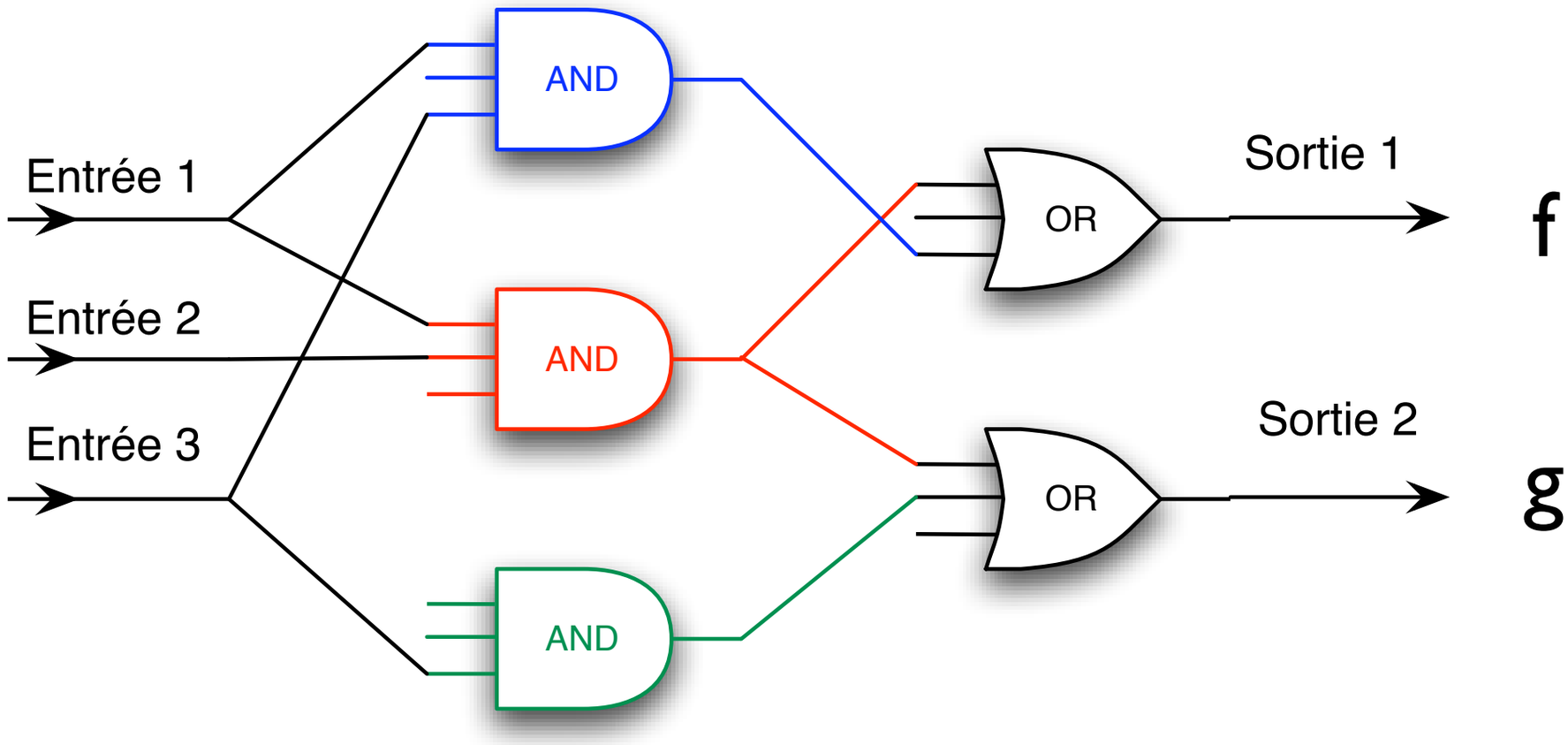
$s_1 = (e_1 \,\&\, e_2 \,\&\, e_3) \mid (e_1 \,\&\, e_2 \,\&\, e_3) \mid (e_1 \,\&\, e_2 \,\&\, e_3)$

$s_2 = (e_1 \,\&\, e_2 \,\&\, e_3) \mid (e_1 \,\&\, e_2 \,\&\, e_3) \mid (e_1 \,\&\, e_2 \,\&\, e_3)$

by cutting off certain connections,

the PLA can be configured

PLD



This PLD computes the two functions:

$f(e_1, e_2, e_3) = e_1 \,\&\, (e_2 \mid e_3) = (e_1 \,\&\, e_3) \mid (e_1 \,\&\, e_2)$

$g(e_1, e_2, e_3) = e_3 \mid (e_1 \,\&\, e_2)$

PLD

- In practice, real PLA's:
  - contain thousands of OR and AND gates
  - also contain negations (for the inputs and outputs)

PLD

- In the case of PLAs, the configuration step is done once and for all by the manufacturer
- PLAs are thus very clos to ASICs except that it is simpler and cheaper to cut off links than to create links between cells

PLD

- Some PLD even allow the user to configure and configure again the device
  - They are programable "*on the field*"
    - e.g. : FPGA = *Field Programmable Gate Array*
- The configuration is stored in an (E)EPROM or Flash

PLD ≠ CPU

- It is important to bear in mind the difference between a PLD and a CPU !
  - PLD = logic circuit than can be configured
    - The configuration is not a program. It is the description of a Boolean function.
  - CPU = logic circuit that executes a program, instruction by instruction (with loops, ifs...)

Software

Executing software

- The software has to be executed on a CPU:
  - Which CPU ?
  - How ?
    - The CPU executes a single program
    - An OS is used to allow the execution of multiple programs that share resources

Microcontroler

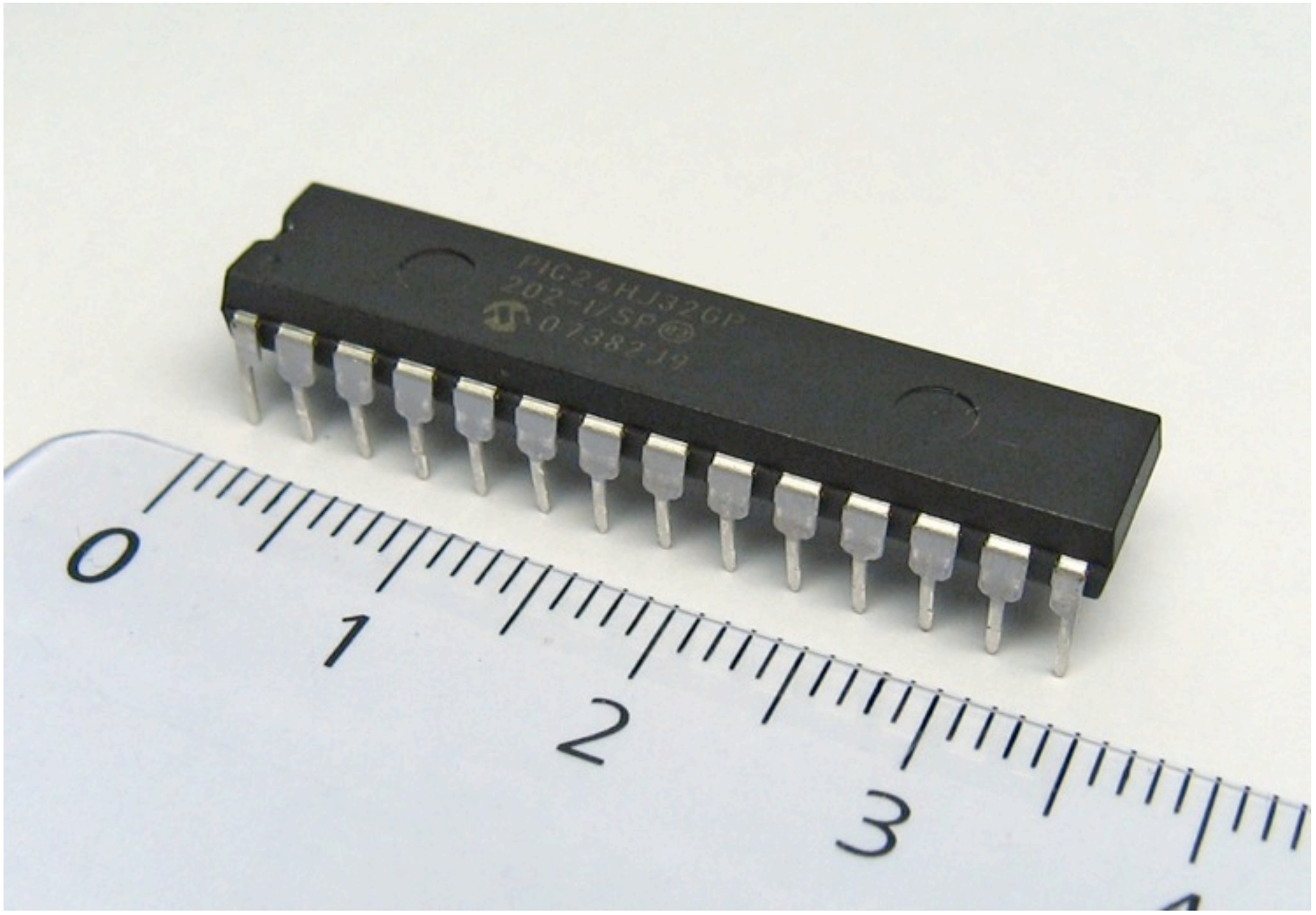
- In the no-OS case, a microcontroler is usually sufficient
- A microcontroler is a simplified CPU, whose chip usually includes other components (unlike a CPU) :
  - RAM, ROM, I/O...

Example

- A widespread family of microcontrolers is the PIC developed by Microchip
- Hundreds of different versions of PIC can be purchased:
  - 8 bits, 16 bits, 32 bits
  - various housings
  - the price goes from a few cents to several euros...

Example - PIC

PIC24



source: wikipedia

Exemple - PIC

- PICs boast a Harvard architecture: there are separate memories for code and data, both in the same IC
  - data is stored in RAM
  - code is stored in EEPROM or Flash
- The instruction set is tiny. Most instructions execute in one cycle
- Programmable interruptions
- C as favourite programming language

Microcontrolers

- Microcontrolers usually consume less than CPUs
- Most microcontrolers have no MMU
  - which prevents them from running several OSes

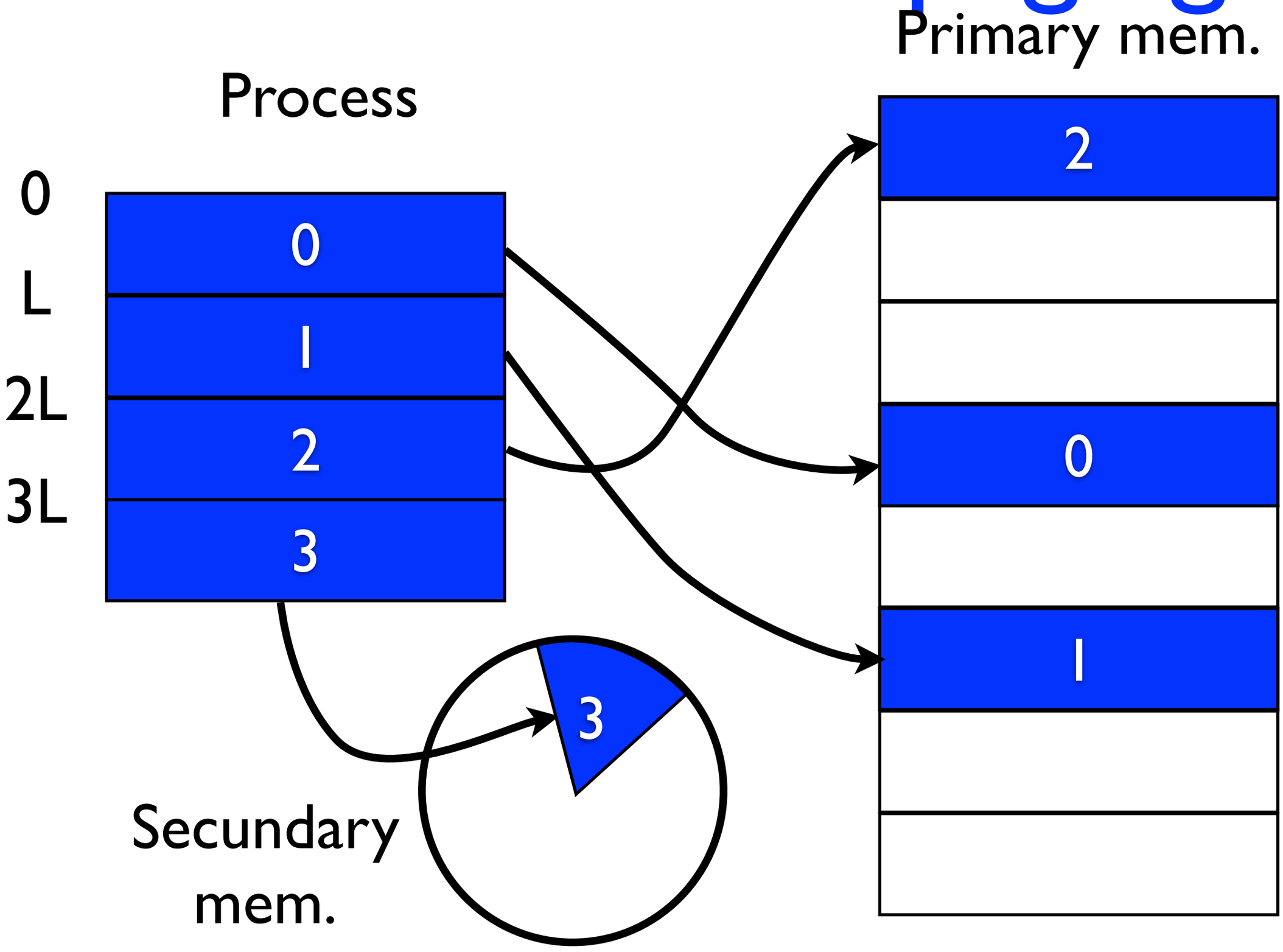
On-demand paging

- In modern computers, memory is managed by (on-demand) paging
- Main ideas:
  - Addresses in the programs are relative to the first program word
  - Processes are divided into pages of equal length
  - Memory is divided into page frames, of the same length

On-demand paging

- Necessary pages are loaded to primary memory, into the page frames, but:
  - not contiguously
  - not in the same order as in the process
- Unused pages are stored in the secundary memory

On-demand paging



MMU

- Addresses in the process are called virtual addresses
  - The process «sees» only virtual addresses and behaves as if it was loaded contiguously at address 0, with nearly unlimited memory available
- Addresses in the mémoire are called real addresses
  - The real address that corresponds to a virtual address depends on the position of the pages in memory

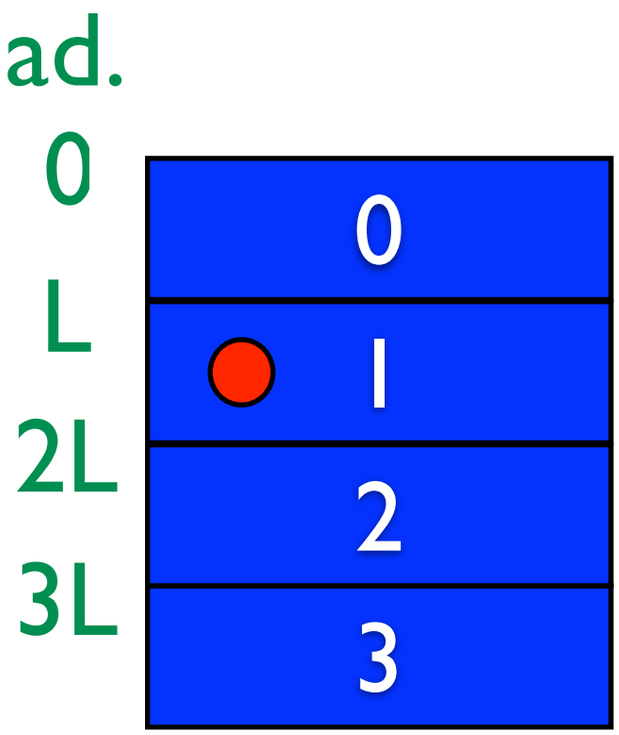
MMU

- Since the position of the pages changes over time, one needs an efficient address translation mechanism

- The MMU is a specialised hardware that implements it
- It relies on a page table that tells it where each page loaded

MMU

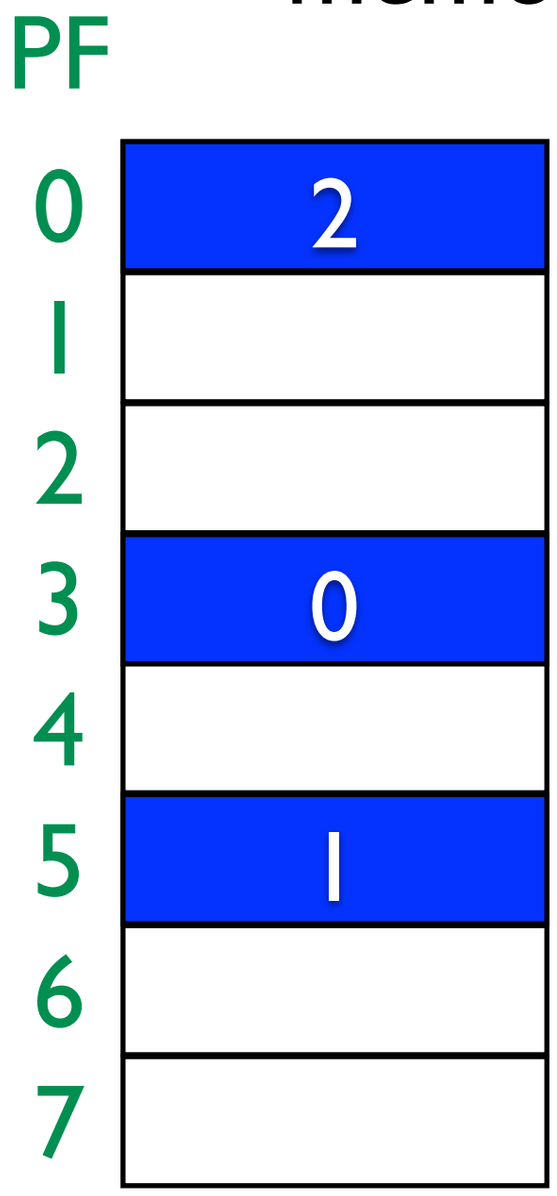
Process



Page table

Page	PF
0	3
1	5
2	0
3	-

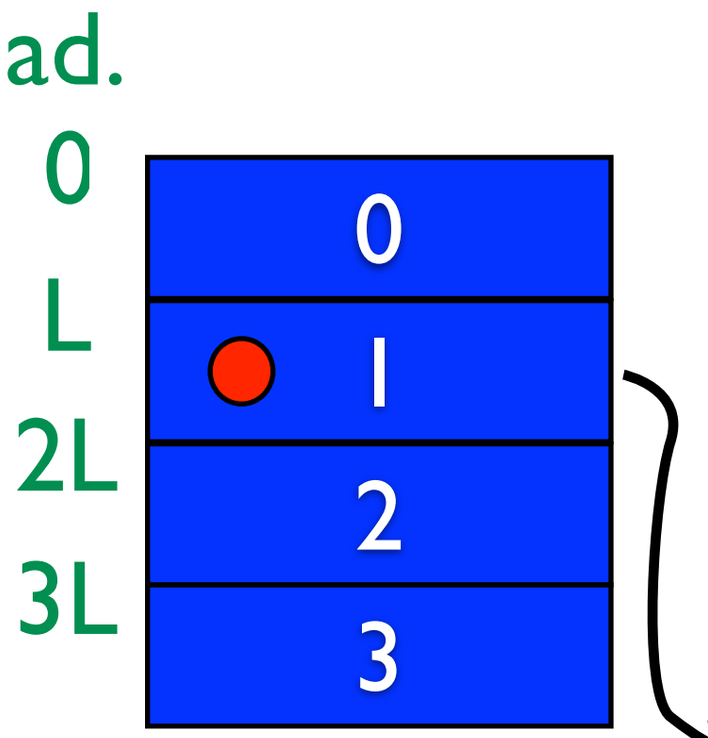
Main memory



Any VA $L \leq a < 2L$ is in page $a/L = 1$, at an offset $a \bmod L$ from the beginning of the page

MMU

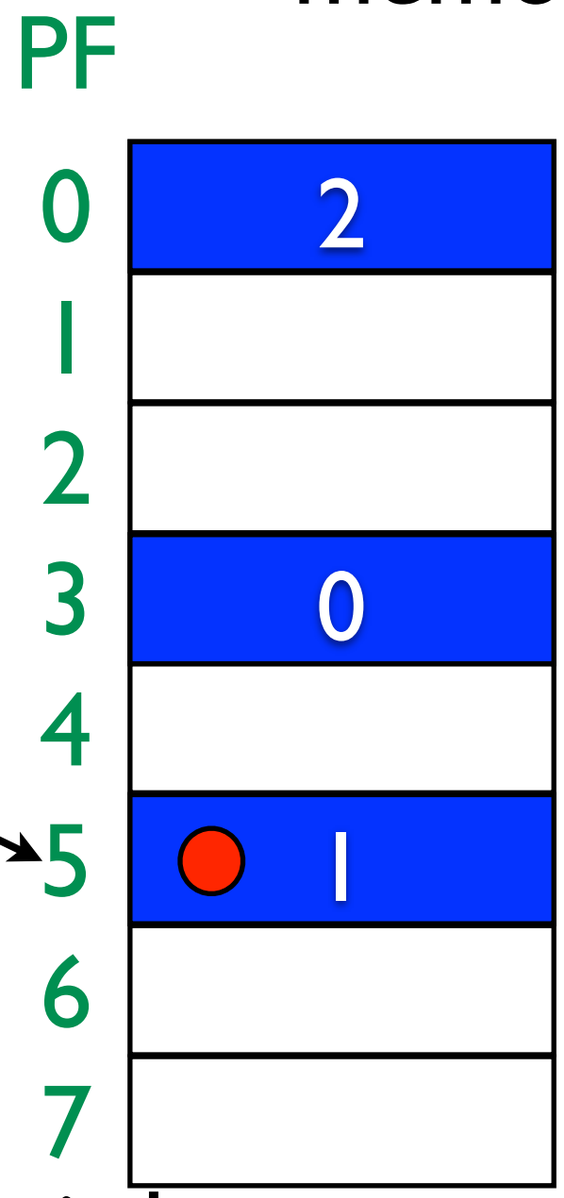
Process



Page table

Page	PF
0	3
1	5
2	0
3	-

Main memory



The corresponding real address i thus:

address of PF 5 + a mod L =

5L + a mod L

Embedded Os

- If the ES has sufficient computing resources, an OS can be used
- Advantages:
  - Easier to develop (almost like on a desktop machine)
  - Hardware drivers...

Embedded OS

- Advantages:
  - Easier to re-use code (maybe from a non-embedded project)
  - Real time OSes can deal with real time constraints

Embedded OS

- Drawbacks:
  - Operating systems consume resources:
    - CPU
    - RAM
    - boot time

Which CPU ?

- Depends on the features and the requested computing power
- Most makes (Intel, Motorola,...) ship embedded versions of their CPUs:
  - Low power consumption
  - Old models (e.g. Celeron for Intel)

Which CPU ?

- However some CPU families are entirely devoted to ESs
- A good example it the ARM family
- ARM CPUs are pretty common in ES of all kind
- In 2006, 2,45 billions ARM CPUs have been sold (source: ARM)

ARM

- ARM CPUs are certainly the most common 32 bits CPUs all over the world
- Some applications: Palm, Nokia and Sony Ericsson phones, Zune, Blackberry, Psion, Lego NXT, Game Boy advance, Nintentdo DS, Apple Newton,...

ARM

- The ARM Holdings company does not produce the CPUs
- It sells designs and licenses manufacturers to produce the CPUs
- The design can be adapted to fit individual needs, but all ARM CPUs share a common architecture (same instruction set).
- Main manufacturers: Marvell (ex Intel), Toshiba, Samsung...

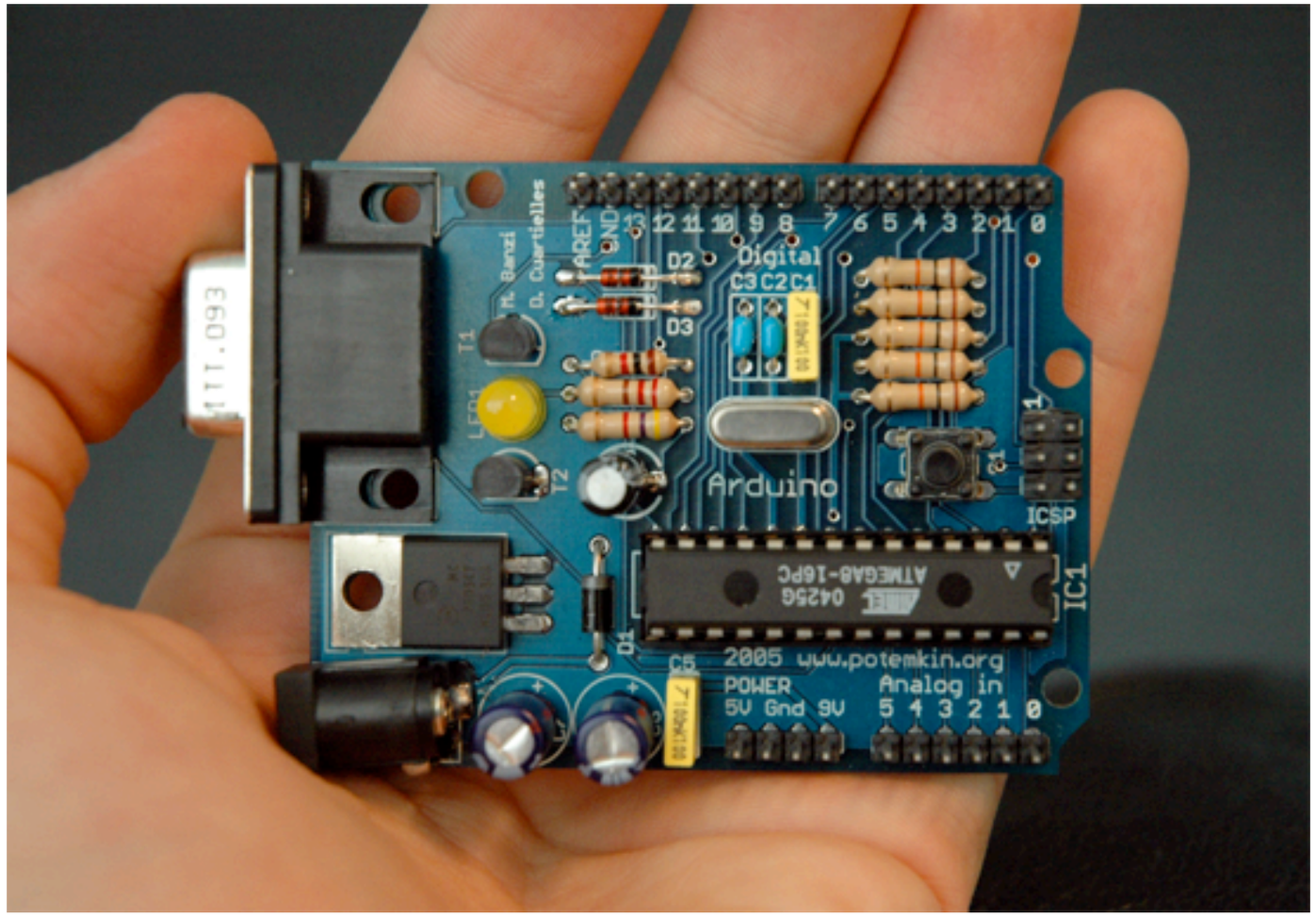
source: Building embedded Linux Systems

ARM

- Linux can run on most ARM CPUs and supports more than 3,300 ARM-based computer
- see: http://www.arm.linux.org.uk/developer/machines/

Arduino

- We have ordered several Arduino boards
  - Atmel microprocessor
- http://www.arduino.cc/



Versatile Board

- In a practical, we'll used QEmu, a CPU emulator to emulate this ARM-based board
- see: http://bellard.org/qemu/
- We'll see how to install Linux on that system, and how to port applications on it.

Versatile


smc91x.c: v1.1, sep 22 2004 by Nicolas Pitre <nico@cam.org>
eth0: SMC91C11xFD (rev 1) at c8868000 IRQ 25 [nowait]
eth0: Ethernet addr: 52:54:00:12:34:56
Loading iSCSI transport class v2.0-870.
PCI: enabling device 0000:00:0c.0 (0140 -> 0143)
sym0: <895a> rev 0x0 at pci 0000:00:0c.0 irq 27
sym0: No NVRAM, ID 7, Fast-40, LVD, parity checking
sym0: SCSI BUS has been reset.
scsi0 : sym-2.2.3
scsi 0:0:2:0: CD-ROM QEMU QEMU CD-ROM 0.9. PQ: 0 ANSI: 3
target0:0:2: tagged command queuing enabled, command queue depth 16.
target0:0:2: Beginning Domain Validation
target0:0:2: Domain Validation skipping write tests
target0:0:2: Ending Domain Validation
Driver 'sd' needs updating - please use bus_type methods
mice: PS/2 mouse device common for all mice
i2c-parport-light: adapter type unspecified
TCP cubic registered
NET: Registered protocol family 17
SCTP: Hash tables configured (established 4096 bind 8192)
input: AT Raw Set 2 keyboard as /class/input/input0
VFS: Cannot open root device "<NULL>" or unknown-block(0,0)
Please append a correct "root=" boot option; here are the available partitions:
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)

Which OS ?

- DOS
  - Sometimes sufficient for very simple applications
    - e.g.: old «bancontact» terminals
  - + Monotask system: easily predictable
  - + Low resources
  - + Free versions exist (freedos)

Which OS ?

- DOS
  - - Not secured
    - no protected memory
    - single user: no file permissions, no login,...
  - - Restricted OS

Which OS ?

- Microsoft Windows:
  - More frequent than you'd think !
    cfr. http://www.windowsfordevices.com/
    - e.g.: common in photo printing booths or display signs
  - old versions: 3.1 (was still supported for ES up to Nov 2008), 95, XP
  - e.g.: entertainment systems on board virgin atlantic flights used to run windows 3.1...

Which OS ?

- GNU/Linux
  - + Completely customisable everything can be modified and recompiled from scratch
  - + Very flexible as far as resources are concerned:
    - 50Mb of secondary memory is sufficient...

Which OS ?

- GNU/Linux
  - + multitask and multiuser
  - + “*Free as in free speach and free beer*”
  - + real time support (in some versions)
  - + Lots of libraries available
  - - Needs an MMU (otherwise µCLinux)

Real time Linux

- Several versions are available
- Completely free versions:
  - RTLinux http://fsmlabs.com/community/
  - RTAI - Real Time Application Interface http://www.aero.polimi.it/~rtai/
- Commercial versions:
  - RTLinuPro, Montavista, LynuxWorks,...

Which OS ?

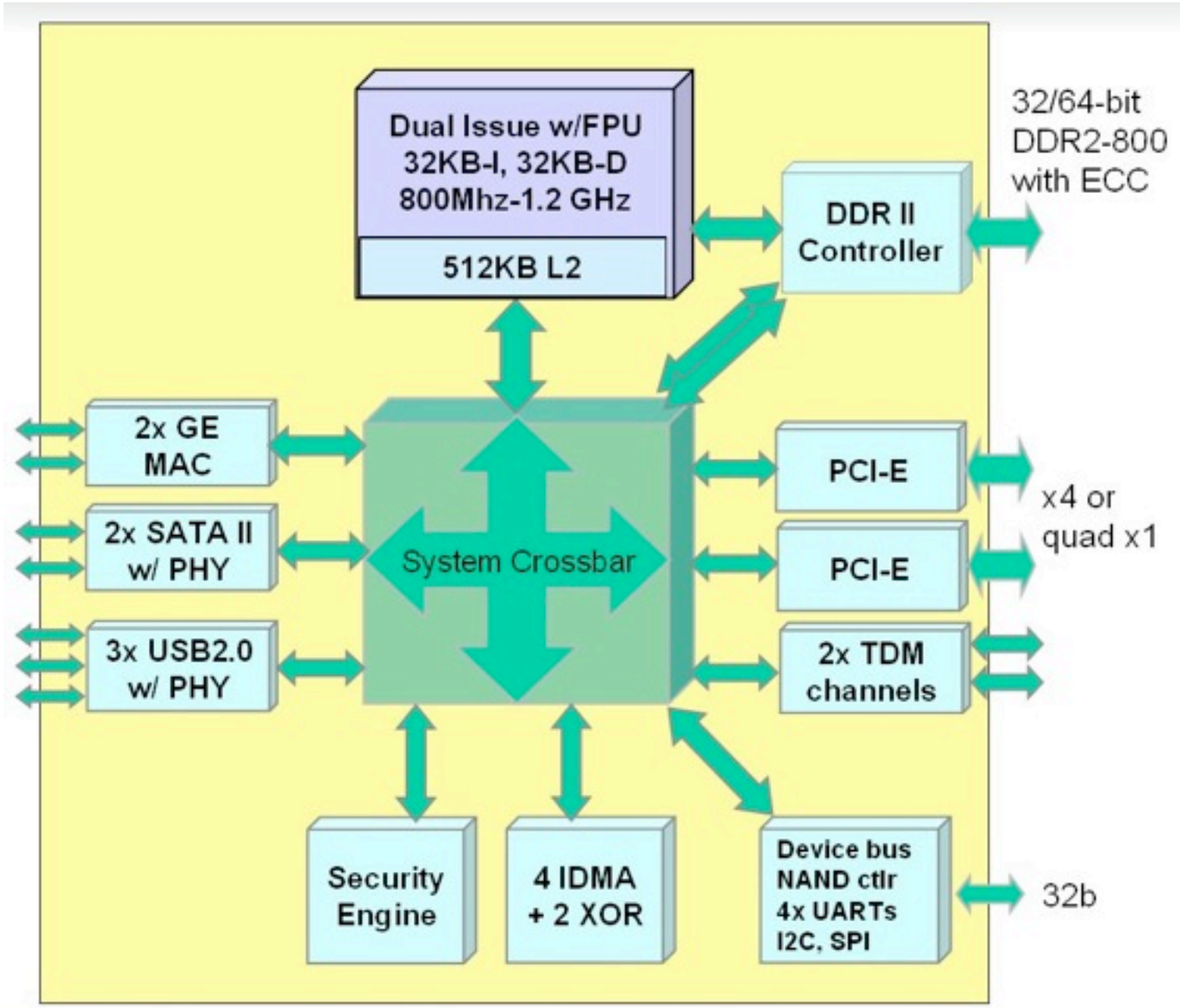
- A dedicated OS:
  - QNX: real time
  - PalmOS / WebOS
  - Windows CE
  - Symbian (ARM CPUs)
- Some of them are not all-purpose (eg: PalmOS)

System on a chip

- It is even possible to have a whole system fitted into a single chip
- This technology is called "*System on a Chip*"

System on a chip

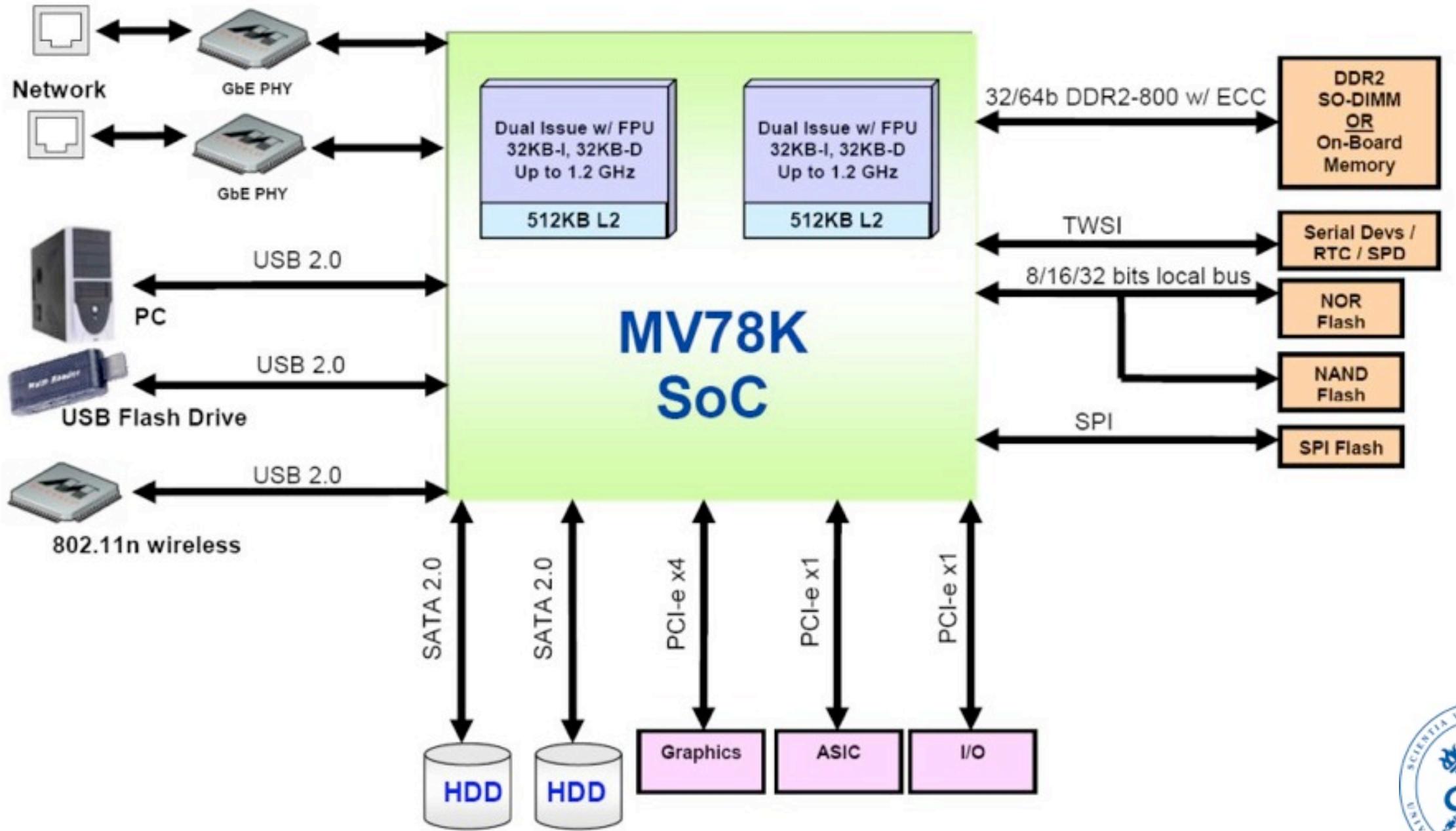
Example: Marvell discovery chip (CPU ARM)



source: http://www.linuxdevices.com/news/NS6658204257.html

System on a Chip

Example: Marvell discovery chip (CPU ARM)



source: http://www.linuxdevices.com/news/NS6658204257.html

Developing an ES application

Development

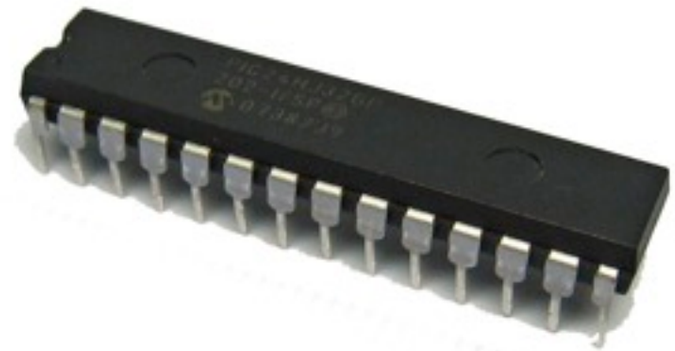
- The development process of an ES application is somewhat different from the development of a desktop application
- One of the main differences: developing is usually carried out on a machine that is quite different from the hardware that will run the code

Developing - Example

Development machine

Intel PC workstation

Target machine



PIC microcontroler

Cross compilers

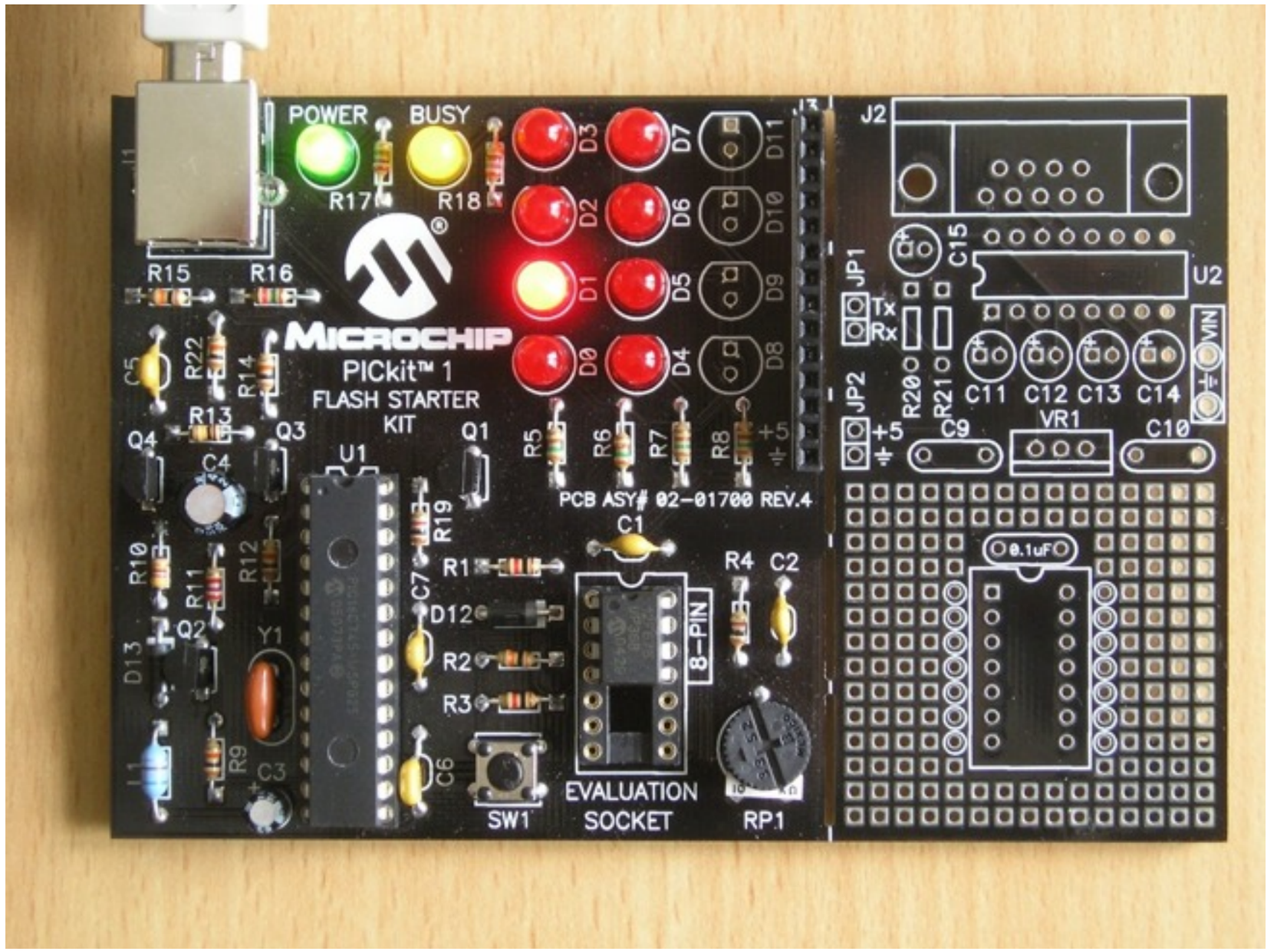
- The most important tool is the cross compiler
  - In general, a compiler handles code in a language L, runs on a machine D and produces a binary for a machine H
  - Usually: D = H
  - For a cross compiler: D ≠ H
  - e.g.: g++, L = C++, D = linux i386, H = linux ARM

Loading the binary

- Loading and executing the binary on the ES also differs from the traditional case
- The technicalities depend on the target platform
- e.g.: Flash memory: one can use an USB card reader
- e.g.: PIC: one needs a dedicated board

Loading the binary

Example: PICkit1 board from Microchip



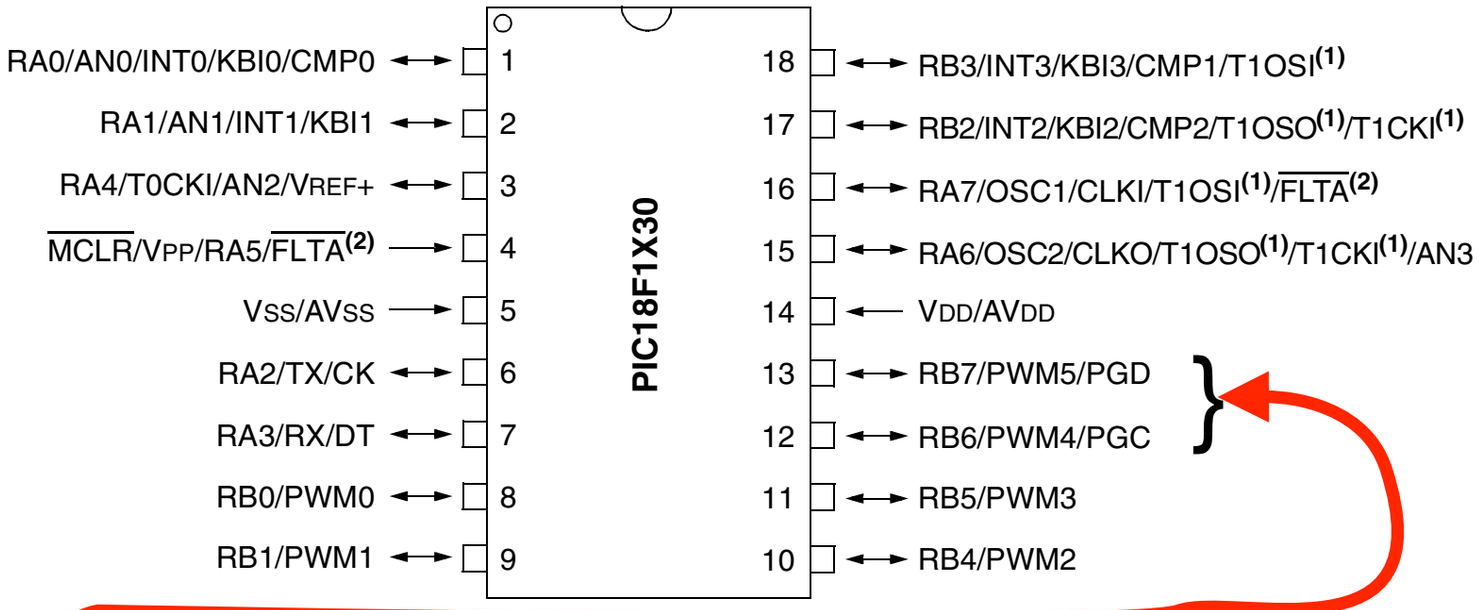
Debugging

- Since the target machine and the development machines are different, debugging is also an issue
  - For microcontrolers: some allow "*in-circuit*" debugging
    - Allows to access registers and internal variables as the microcontroler executes

Debugging

Example: PIC18

18-Pin PDIP, SOIC



Debugging

- One can also use simulators or virtual machines
  - that are more or less accurate ?
- These technologies allow to test and debug the system on a traditional workstation

Contents and objectives of the course

Content and objectives

- We'll be mainly concerned by the software aspects of ES:
- We'll review tools to:
  - ensure that the system is reliable
  - ensure control objectives
  - take real time constraints into account

Content and objectives

- We'll see that we can rely on formal models of the system to guarantee these constraints are met
- = Model centric design
- In some cases we won't have to write code: it will be automatically generated from the model !

Content and objectives

- We will also briefly review some technological aspects to get familiar with ES «in the wild» and motivate our theoretical discussions

Syllabus

- Modeling and verifying with SCADE
- Control theory
- Game theory with applications to controller synthesis
- Real time scheduling with Linux
- Embedded Linux
- One additional topic ?