Sistemas Embarcados

Fabiano Hessel
hessel@inf.pucrs.br
http://www.inf.pucrs.br/~hessel

Outline

- The embedded computing space.
- Platforms: system-on-chip, networks.
- Architectures, applications, methodologies.
- Standards-based design.
  - Multiple standards.

Introduction

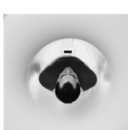
- What are embedded systems?
- Challenges in embedded computing system design.
- Design methodologies.

Definition

- Embedded system: any device that includes a programmable computer but is not itself a general-purpose computer.
- Take advantage of application characteristics to optimize the design

Example embedded computing systems


Motorola


Siemens

Apple


BMW

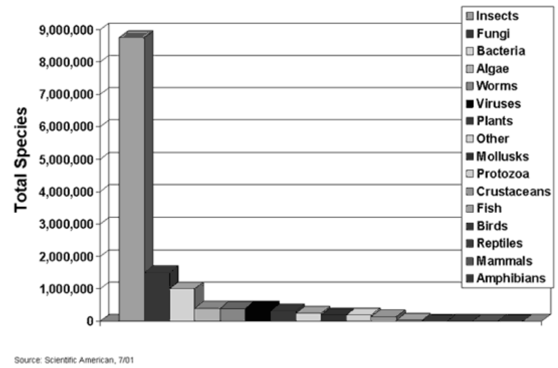
Early history

- Automobiles used microprocessor-based engine controllers starting in 1970's.
  - Control fuel/air mixture, engine timing, etc.
  - Multiple modes of operation: warm-up, cruise, hill climbing, etc.
  - Provides lower emissions, better fuel efficiency.

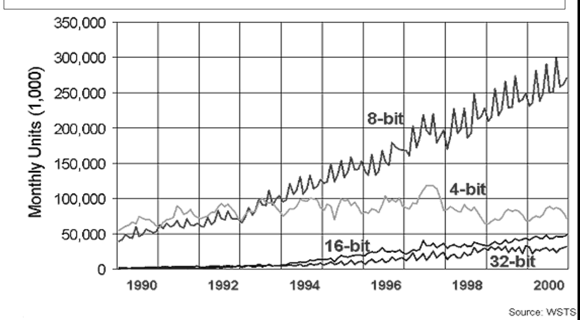
Microprocessor varieties

- Microcontroller: includes I/O devices, on-board memory.
- Digital signal processor (DSP): microprocessor optimized for digital signal processing.
- Typical embedded word sizes: 8-bit, 16-bit, 32-bit.

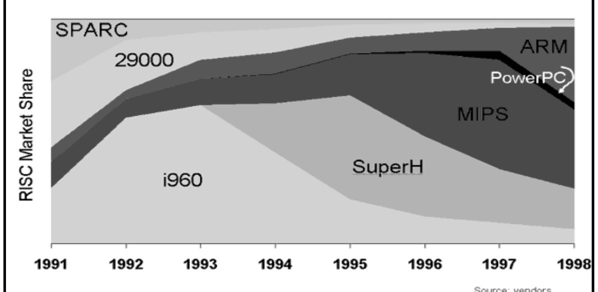
All Life on Earth Is Insects…



As vendas dos microprocessadores Pentium da Intel representam apenas cerca de 2% do mercado de processadores:



A grande diversidade de aplicações justifica a grande variedade de processadores para sistemas embarcados existentes.



Application examples

- Simple control: front panel of microwave oven, etc.
- Canon EOS 3 has three microprocessors.
  - 32-bit RISC CPU runs autofocus and eye control systems.
- Analog TV: channel selection, etc.
- Digital TV: programmable CPUs + hardwired logic.

Application Examples, contd.

- Personal digital assistant (PDA).
- Printer.
- Cell phone.
- Automobile: engine, brakes, dash, etc.
- Television.
- Household appliances.
- PC keyboard (scans keys).

Multiprocessor systems-on-chips

- Roughly speaking, system-on-chip with at least two processors.
- Usually heterogeneous multiprocessor:
  - CPUs, DSPs, etc.
  - Hardwired accelerators.
  - Mixed-signal front end.

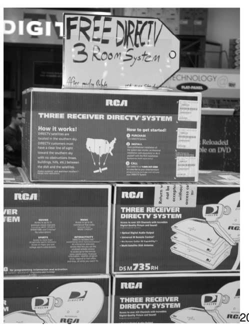
Embedded Computing © 2005 Wayne Wolf

Consumer electronics categories

	2001	2002	2003	2004
Satellite TV	$1.18	$1.12	$1.48	$1.89
DVR (40E6)	0.14	0.57	0.18	0.54
DVD	2.1	2.43	2.7	2.46
Set-top Internet	0.20	0.12	0.63	0.341
PC (120E6)	12.96	12.61	15.58	17.2

Embedded Computing © 2005 Wayne Wolf Wall Street Journal/EIA

Consumer electronics prices



Best Buy November 2003:

Characteristics of embedded systems

- Very high performance, Sophisticated functionality.
  - Vision + compression + speech + networking all on the same platform.
- Multiple task, heterogeneous.
- Real-time.
- Often low power.
- Highly reliable.
  - I reboot my piano every 4 months, my PC every day.
- Designed to tight deadlines by small teams.
- Low manufacturing cost.

Embedded Computing © 2005 Wayne Wolf

Real-time operation

- Must finish operations by deadlines.
  - Hard real time: missing deadline causes failure.
  - Soft real time: missing deadline results in degraded performance.
- Many systems are multi-rate: must handle operations at widely varying rates.

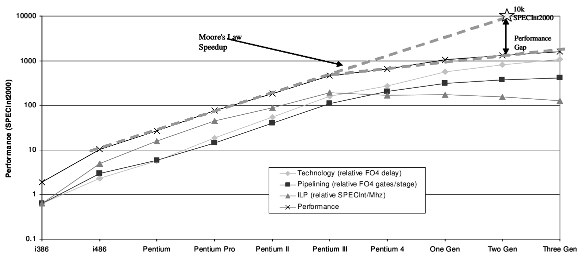
Embedded Computing © 2005 Wayne Wolf

Mudge et al: Mobile supercomputing

- Future mobile platform:
  - Speech recognition.
  - Cryptography.
  - Augmented reality.
  - Typical applications (email, etc.).
- Requires 16x 2 GHz Pentium 4.
- Peak power must not exceed 75 mW.
  - Assumes 5% battery improvement per year.

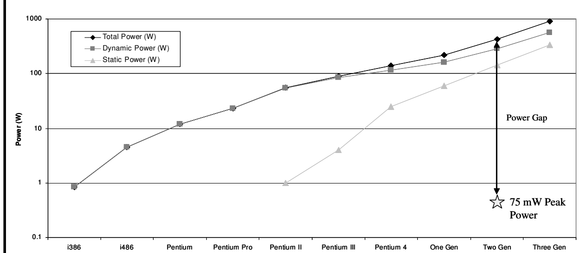
Embedded Computing © 2005 Wayne Wolf

Mudge et al: Performance trends for desktop processors



© 2004 IEEE Computer Society

Mudge et al: Power trends for desktop processors



© 2004 IEEE Computer Society

Non-functional requirements

- Many embedded systems are mass-market items that must have low manufacturing costs.
  - Limited memory, microprocessor power, etc.
- Power consumption is critical in battery-powered devices.
  - Excessive power consumption increases system cost even in wall-powered devices.

Design teams

- Often designed by a small team of designers.
- Often must meet tight deadlines.
  - 6 month market window is common.
  - Can't miss back-to-school window for calculator.

Challenges in embedded system design

- How much hardware do we need?
  - How big is the CPU? Memory?
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
- How do we minimize power?
  - Turn off unnecessary logic? Reduce memory accesses?

Challenges, etc.

- Does it really work?
  - Is the specification correct?
  - Does the implementation meet the spec?
  - How do we test for real-time characteristics?
  - How do we test on real data?
- How do we work on the system?
  - Observability, controllability?
  - What is our development platform?

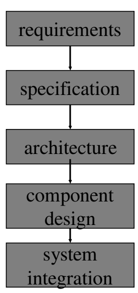
Design methodologies

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn’t skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - help automate methodology steps;
  - keep track of the methodology itself.

Design goals

- Performance.
  - Overall speed, deadlines.
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

Levels of abstraction



Top-down vs. bottom-up

- Top-down design:
  - start from most abstract description;
  - work to most detailed.
- Bottom-up design:
  - work from small components to big system.
- Real design uses both techniques.

Stepwise refinement

- At each level of abstraction, we must:
  - analyze the design to determine characteristics of the current state of the design;
  - refine the design to add detail.

Requirements

- Plain language description of what the user wants and expects to get.
- May be developed in several ways:
  - talking directly to customers;
  - talking to marketing representatives;
  - providing prototypes to users for comment.

Functional vs. non-functional requirements

- Functional requirements:
  - output as a function of input.
- Non-functional requirements:
  - time required to compute output;
  - size, weight, etc.;
  - power consumption;
  - reliability;
  - etc.

GPS moving map needs

- Functionality: For automotive use. Show major roads and landmarks.
- User interface: At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- Performance: Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- Cost: $500 street price = approx. $100 cost of goods sold.

GPS moving map needs, cont'd.

- Physical size/weight: Should fit in hand.
- Power consumption: Should run for 8 hours on four AA batteries.

Specification

- A more precise description of the system:
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- May include functional and non-functional elements.
- May be executable or may be in mathematical form for proofs.

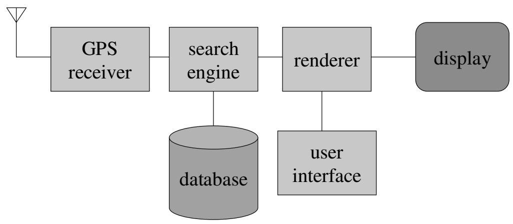
GPS specification

- Should include:
  - What is received from GPS;
  - map data;
  - user interface;
  - operations required to satisfy user requests;
  - background operations needed to keep the system running.

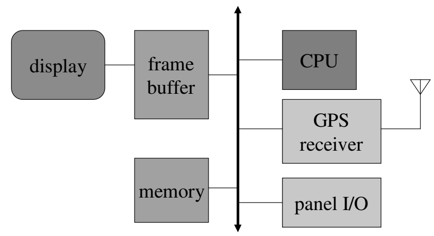
Architecture design

- What major components go satisfying the specification?
- Hardware components:
  - CPUs, peripherals, etc.
- Software components:
  - major programs and their operations.
- Must take into account functional and non-functional specifications.

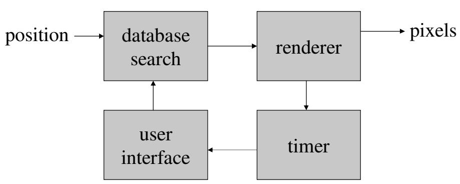
GPS moving map block diagram



GPS moving map hardware architecture



GPS moving map software architecture



Designing hardware and software components

- Must spend time architecting the system before you start coding.
- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.

System integration

- Put together the components.
  - Many bugs appear only at this stage.
- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible.

Summary

- Embedded computers are all around us.
  - Many systems have complex embedded hardware and software.
- Embedded systems pose many design challenges: design time, deadlines, power, etc.
- Design methodologies help us manage the design process.

Platforms

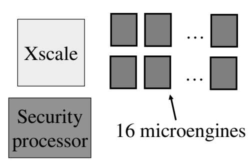
- An architecture that is designed for an application domain:
  - Can be used in several products.
  - Allows customization.
- Platforms are often customized for their target audience.
- Platforms spread out development costs over more products.
- Some people hope for a single universal platform…

Why multiple platforms?

- People still care about cost.
- People care about power consumption.
- Sufficiently general solutions don't fit on one chip.

Intel IXP2850 network processor

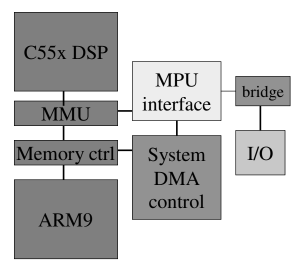
- Packet processing, control processing, security.
- Software development environment includes simulator.



TI OMAP

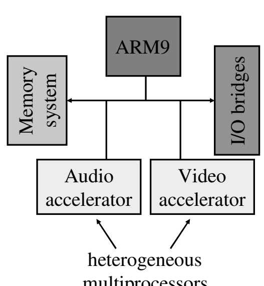
- Targets communications, multimedia.
- Multiprocessor with DSP, RISC.

OMAP 5910:



ST Nomadik

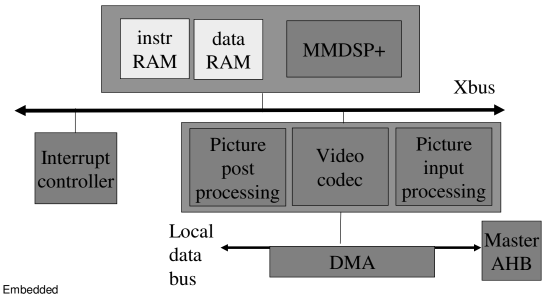
- Targets mobile multimedia.
- A multiprocessor-of-multiprocessors.



ST MMDSP+

- Embedded processor core used in multiple chips:
  - Runs at 175 MHz.
  - 1 cycle per instruction.
  - 2-level instruction cache.
  - 16/24-bit fixed point.
  - 32-bit floating point.
  - C programmed
  - Fully synthesizable.

Nomadik video accelerator



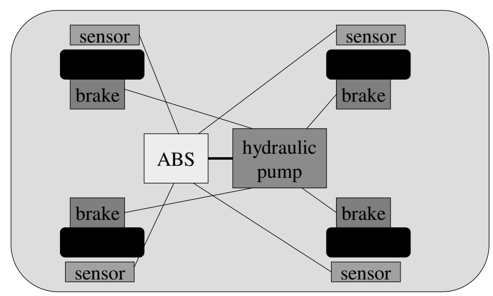
Automotive embedded systems

- Today's high-end automobile may have 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.

BMW 850i brake and stability control system

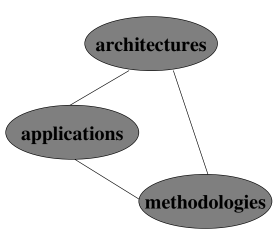
- Anti-lock brake system (ABS): pumps brakes to reduce skidding.
- Automatic stability control (ASC+T): controls engine to improve stability.
- ABS and ASC+T communicate.
  - ABS was introduced first---needed to interface to existing ABS module.

BMW 850i, cont'd.



The eternal triangle

- Hardware and software architectures determine capabilities.
- Applications guide design decisions.
- Methodologies allow repeatable, predictable design.



Observations and implications

- A little domain knowledge helps a lot.
- The architectural design space is large and chunky.
  - Less synthesis, more analysis.
- IP components must be adapted to play together.
  - Configurable IP, wrappers.
  - Supporting tools (compilers, etc.) must be adaptable.

Software in consumer devices (ST)

- Modern audio standards (Dolby, MP3, etc.):
- Modern video standards (MPEG-2, DV, etc.):

- 1 million lines of code.
- 2 million lines of code and counting.

Software and MPSoC design

- The MPSoC must run the application.
  - Design verification must include the software running on the hardware.
- May not know all possible code at design time.
  - Limits design characterization.
  - Must provide programming environment.

MPSoCs and standards

- Standards enable large markets.
  - MPSoCs need large markets to justify chip development costs, reduce manufacturing overhead.
- MPSoCs provide benefits:
  - Low power.
  - High performance.
- Meeting the standard requires effort:
  - Platform must allow multiple implementations.
  - Standard is complex and hard to implement.

Design challenges in standards-driven markets

- Design and verify methods within the standard.
  - Standards allow differentiation.
- Design and verify methods outside the standard's scope.
  - User interface, etc.
- Design and verify interfaces.
  - Within standard, connection to extra-standard elements.

Standards-based systems

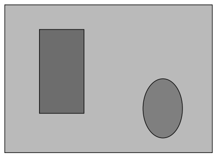
- Reference implementation forms a basis for product.
  - Port to platform.
  - Enhance performance, features.
- Want to minimize unnecessary changes to the software.
- Must make some changes to the software.

Characteristics of reference implementations

- The specification does not describe hardware or software.
  - The spec is in the domain of signal processing, etc.
- Designed for and tested on workstations.
  - Infinite memory.
  - Poor cache behavior.
  - Single process.
  - Limited real-time behavior.
- The executable spec misrepresents some system properties:
  - Error handling.
  - Buffer management.

H.264 motion estimation, cont'd.

- Multiple reference frames increases accuracy.
  - Handles occlusion.
- Once again, receiver is more complex.



Why are standards so complex?

- Algorithm designers like to design algorithms.
  - Standards are complex.
- Standards bodies must embody competing interests, ideas in their standards.

MPEG Tampere meeting

Design refinement

- Bad news:
  - hard to learn the platform in order to change it.

Worldwide shipping by UPS ...

roughly US$ 50 for CD and US$ 100 for paper copy

(1500 pages, heavy!)

Bluetooth.com

- Good news:
  - an existing design can be measured, analyzed, and refined.

Four types of people

- Algorithms people.
  - Don't like programming.
  - Don't know that hardware exists.
- Software people.
  - Don't like hardware.
- Hardware people.
  - Tolerate software.
  - Don't know applications exist.
- Managers.
  - Don't know anything.
  - Don't do anything.

Example: MPEG-2 codec

- One of the reference MPEG-2 codecs.
  - Simple algorithms.
- Designed for workstation operation.
- Implementers must port to chosen platform.
  - Limited memory.
  - Limited CPU.

MPEG-2 porting challenges

- Codec uses a mixture of buffering strategies.
  - Some buffers are statically allocated.
  - Some buffers are allocated from the heap.
- May need to change number representation.
  - Integer, double-precision, etc.
- Error messages use Unix methods.

Example: H.264 codec

- Reference encoder is 700,000 lines of C code.
  - Uses simple algorithms.
- Supports a wide range of:
  - Display sizes.
  - Features.

H.264 porting challenges

- Figure out what code is of interest.
  - Large call graph.
- May need to change number representation.
  - Integer, double-precision, etc.
- Buffer management.
  - Buffer allocation takes up over 50% of CPU time.

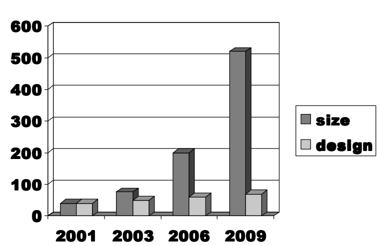
Multiple standards

- Many MPSoCs must implement multiple standards:
  - Communications.
  - Networking.
  - Multimedia.
  - Security.
- Requires running a lot of different types of algorithms.
  - Good case for specialization, co-design, configurable CPUs, etc.
  - Need some general-purpose computers for load sharing, compatibility.

Platforms, standards, and MPSoCs

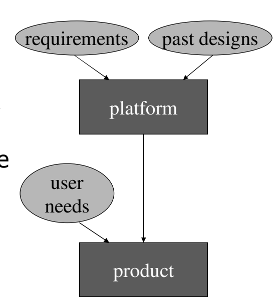
- A platform allows multiple variations of a system.
  - Well-suited to standards.
- Programmability is key to platform-based design.

The design productivity gap



Two phases of platform-based design

- Semiconductor house designs the platform.
  - Requirements may come from standards, systems houses.
- Systems house uses the platform.
  - May need to start design before chip is available.



Challenges in platform-based design

- Don't have the full application.
  - Must estimate characteristics of part of the application.
- Must determine the appropriate level of programmability.
  - Programmability often costs in area, power.
- Must provide programming tools along with the chip.

Transaction-level modeling is not enough

- The MPSoC must run the complete application.
  - Implementing transactions is necessary but not sufficient.
- Transactions are relatively short term.
- SoCs have a lot of state in memory.
  - Need to thoroughly exercise that state over a long period.

Summary

- Chip designers are now system designers.
  - Must deal with hardware and software.
- Today's applications are complex.
  - Reference implementations must be optimized, extended.
- Platforms present challenges for:
  - Hardware designers---characterization, optimization.
  - Software designers---performance/power evaluation, debugging.