Problems Facing Embedded Systems

Philip Koopman

koopman@cmu.edu - (412) 268-5225 - http://www.ices.cmu.edu/koopman

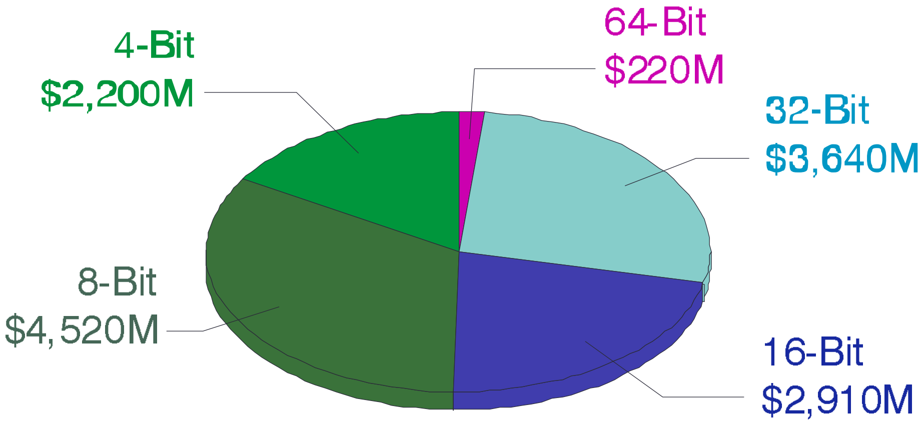
Institute for Complex Engineered Systems

Carnegie Mellon

Electrical & Computer ENGINEERING

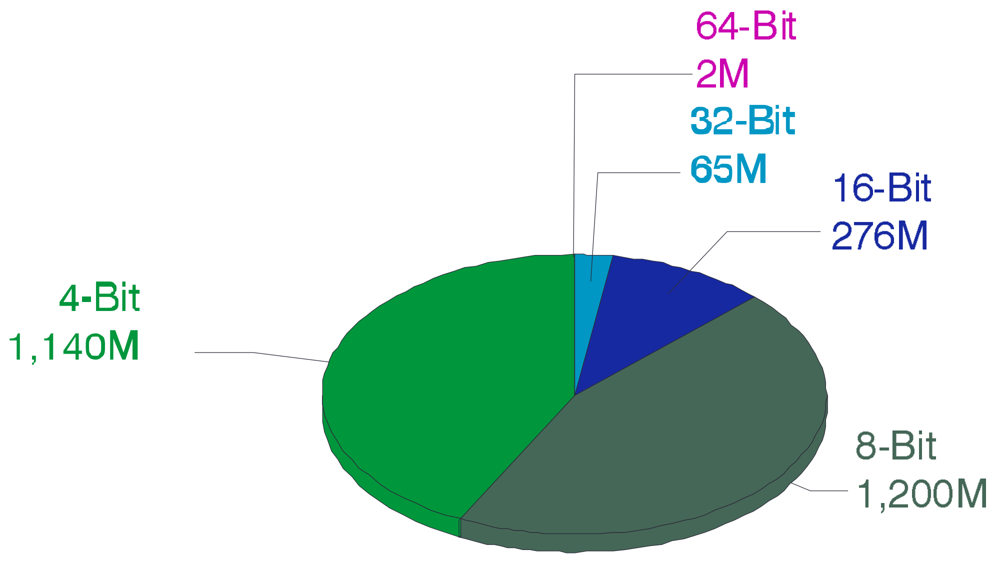
Embedded System Context

- **Don’t think in terms of just cost or just performance -- think in terms of how much you get for:**
  - $1 chip (on-chip memory only) -- most of the market
  - $10 chip (with one RAM/ROM combo chip) -- much of the market
  - $100 chip (with DRAM + 1 boot flash chip) -- a tiny piece of the market



$13,490M Total

1994 Worldwide
Microcontroller Revenue
($Million U.S.)



2,683M Total

1994 Worldwide
Microcontroller Units
(Million Devices)

Approximated from EE Times,
March 20, 1995
Source: The Information Architects

Different Systems Have Different Problems

- **Near-desktop systems (set-top box; wearable computer; *etc*.)**
  - Time to market
  - Cost

- **Embedded control systems (elevators, aircraft, factories)**
  - Real-time determinacy (architecture) & predictability (compiler)
  - Off-the-shelf RTOS (Real Time Operating System)
  - Software development problems
  - Cost

- **Tiny embedded systems (rice cookers, *etc*.)**
  - Cost
  - Cost
  - Compilers/runtime on a $1 chip
  - Cost

Relative Importance

#1 - Cost

- **Cost + performance** often matters more than performance
- ("Cost" includes issues such as power, size, weight too)

#2 - Time to Market

- (Debugability is an important factor)

#3 - Predictability/Determinacy

- It is important to pick a fast enough processor for worst case
- Is this really debugability in the performance space?

…

#943 - Instruction Level Parallelism

- Does ILP make sense on an 8051? That is still much of the market
- Most embedded systems use older CPU designs (how many MIPS do you need in a toaster oven?)

Technology Buzz (Embedded Control)

- **Windows CE *vs.* other RTOSs**
  - Remember the phrase “nobody every got fired for buying from IBM?”
  - Lots of companies are thinking about this; maybe with Win CE 3.0 we’ll see more widespread adoption
  - Potentially gives opportunities for Non-Intel CPU designs
- **Java**
  - Most are not really talking about this seriously (at this point)
  - But there’s plenty of Hype!
- **UML/design tools**
  - Design methods often matter most (SW is the problem, not HW)
- **CORBA / DCOM**
  - Distributed object technology is coming
  - What does the HW need to do to make it viable on low-end systems?

Skepticism

- **Networked Everything In A House**
  - CPU is one thing; getting a cheap network connection is another
  - Even a $1 wireless port connection is a lot in a $15 toaster
  - Who wants to debug their house? We can’t even set VCR time now…

Does Java Matter?

- **Maybe, but…**
  - It’s too big
  - Configuration control of applets would be a nightmare for ordinary folks
  - The most numerous low-end systems are still written in assembly language
- **The biggest problem is software development**
  - Language choice is a second-order effect on productivity

Does Reconfigurable Hardware Matter

- **Possibly**

- **Currently a move toward flash memory instead of masked ROM**
  - EVEN on very large volume applications
    - Frequent requirements/design changes
    - Ability to perform field bug patches if recalls occur
    - "Just-in-time" programming + standard parts reduces inventory costs
  - So, maybe reconfigurable hardware matters in the future

- **Is it really just another form of "software"?**
  - Reconfigurable hardware is about having hardware replace software
  - But the other half of the equation is if you have a fast processor, software can replace hardware (*e.g.*, "software serial port")

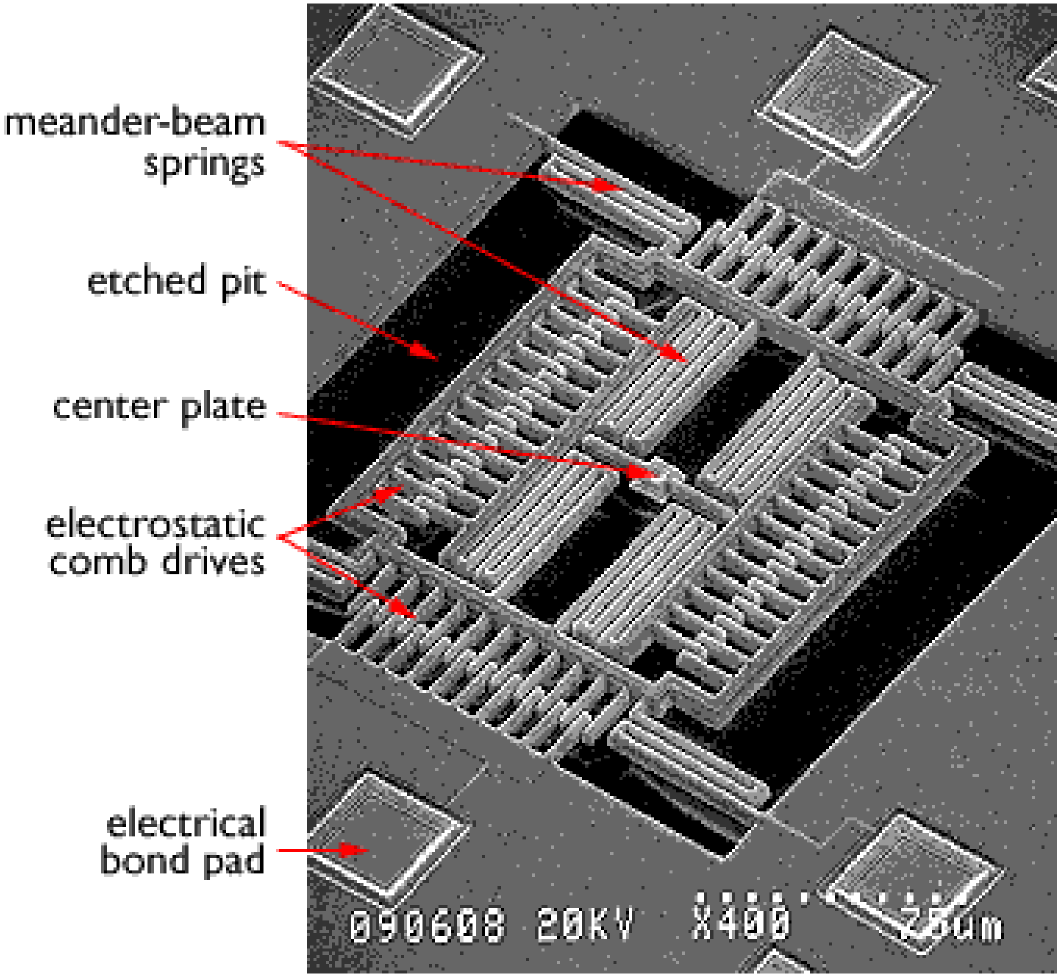
What Are We (Researchers) Missing?

- **Dependability -- we can’t even put a number on it yet**
  - Everyday embedded applications are indirectly mission/safety-critical
    - Pager outage shuts down hospital
    - Incorrect GPS position can sink a ship
  - Design defects (SW + HW) are becoming the biggest culprit
    - Throwing redundant hardware at the problem is an obsolete approach
  - But, we can’t afford to apply current critical-system components & design techniques
    - (and, those approaches don’t even work all that well anyway)
- **Low-end systems**
  - The big problem is not CPU design, it is dealing with *complexity* on a system level
  - Deep, multi-disciplinary tradeoffs -- transistors to business process

New Applications/Problems

- **Very Low Power (wearables; stand-alone devices)**
  - Battery operation for days, not hours
  - Thermal dissipation will be limited by small surface area

- **MEMS-based devices**
  - Micro-Electro-Mechanical Systems
  - In the future, “system-level integration” includes electro-mechanical I/O



Challenge Areas

- **Increase integration levels (including Analog)**
  - Hardware + Software + I/O + Storage co-design -- smallest total chip cost
  - Ultra-fast CPUs or programmable logic are part of the equation
  - It is total system cost that matters most
    - Resist the temptation to optimize the CPU and shove problems off-chip

- **Help solve the ongoing "software crisis"**
  - Speed definitely helps
  - But HW has bugs -- it can be part of the problem
  - HW/SW combined design approaches using standard/customizable parts

- **Biggest opportunity**
  - Nobody cares if their car engine controller is "Intel Inside" (yet…)