Challenges in Embedded Systems Research & Education

Philip Koopman

koopman@cmu.edu - http://www.ices.cmu.edu/koopman

Institute for Complex Engineered Systems

Carnegie Mellon

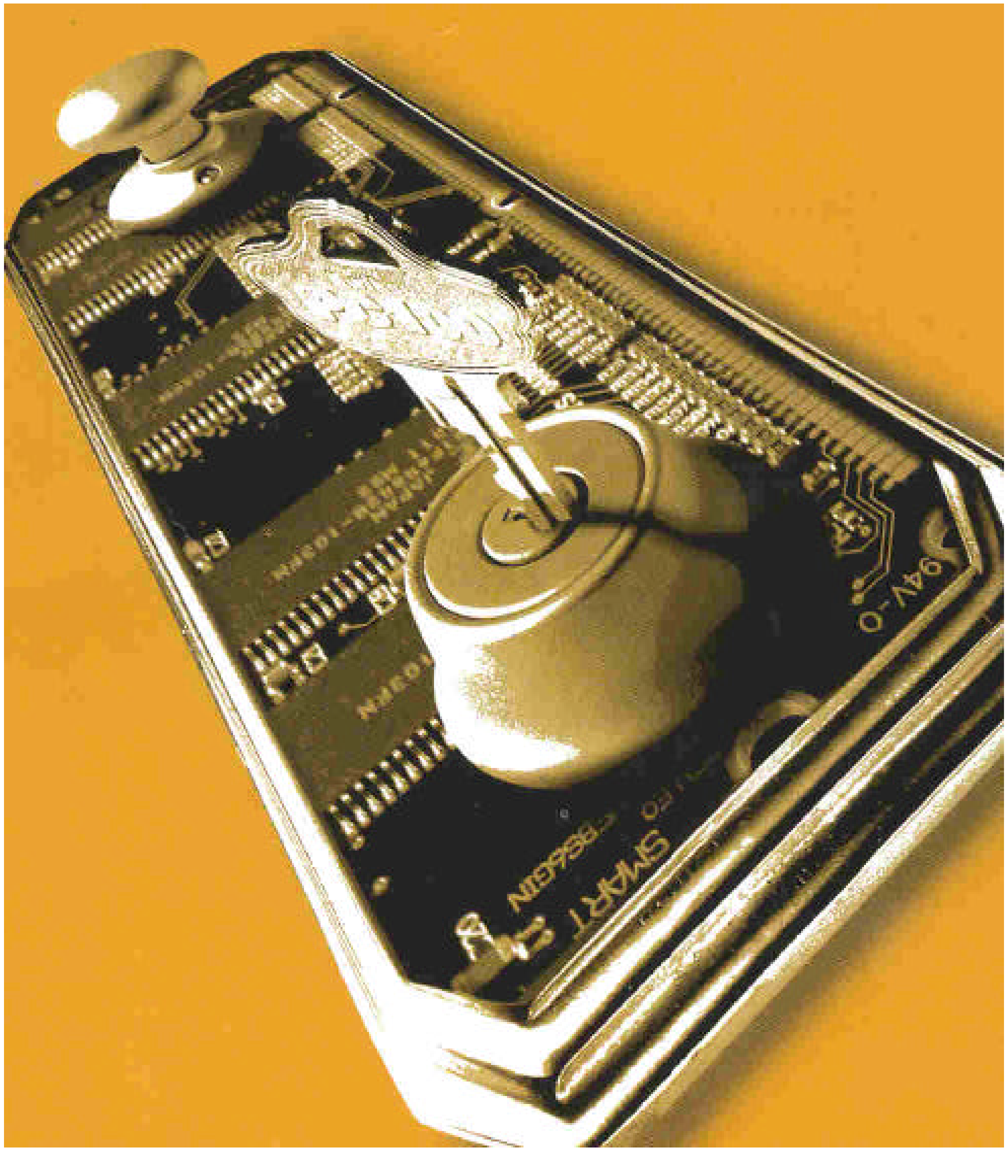
Electrical & Computer ENGINEERING

Circa 1980:

What in the world are you going to do with all those computers?

It's not as if you want one in every doorknob!

- Danny Hillis, circa 1980, as told by Guy Steele at 1996 CMU SCS commencement



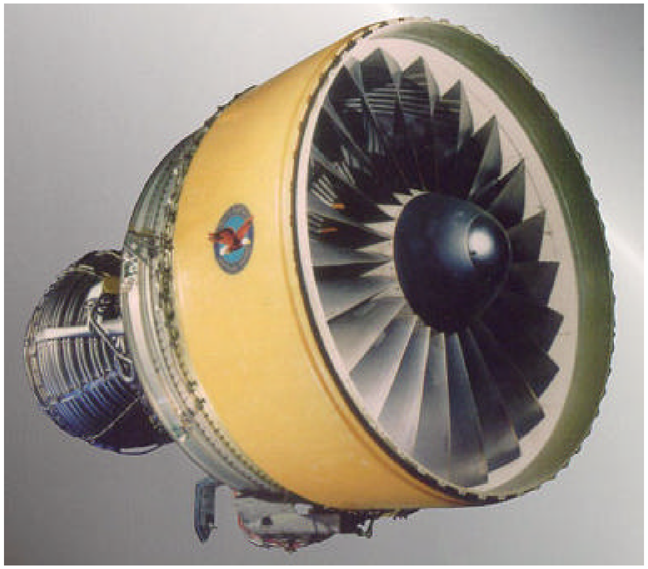
1981:

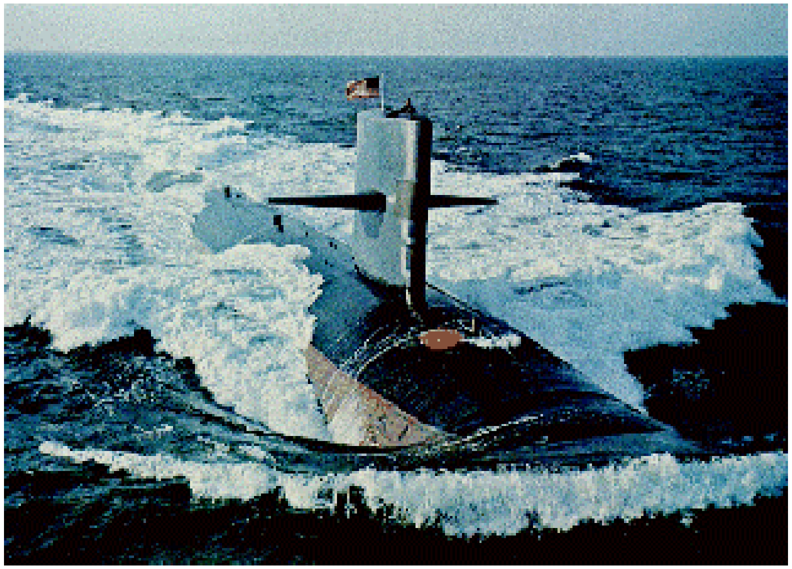
Atari 800 used by hotel control startup company

Overview

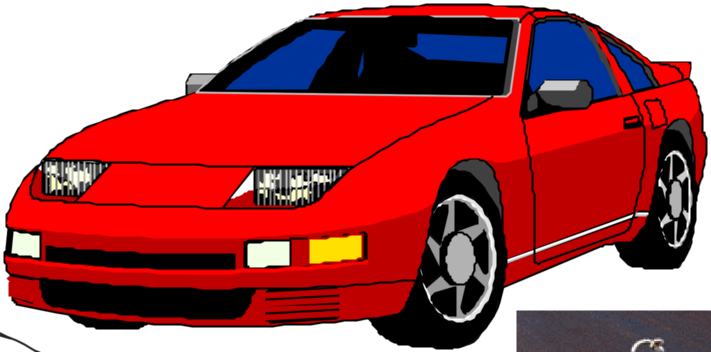
20 Years Later, What's Left To Research?

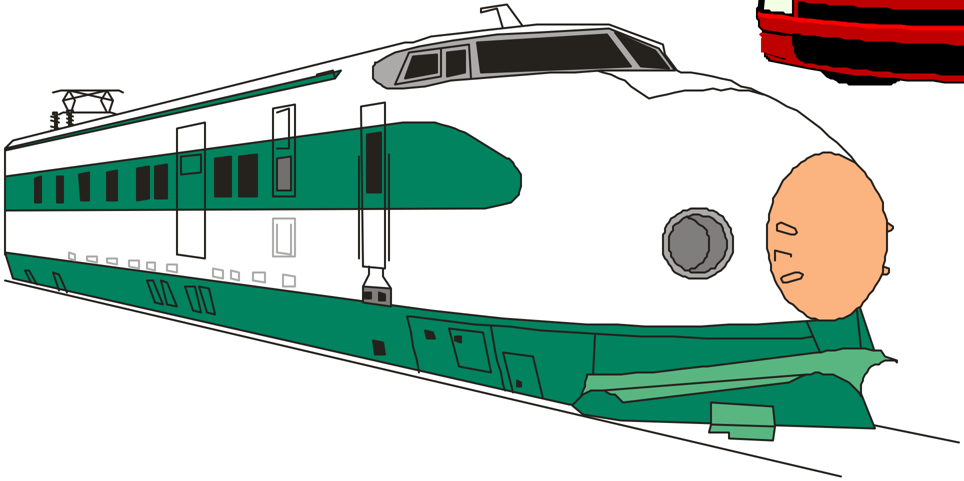
- **What's an embedded system?**
- **Why can't you just design them like desktop systems?**
  - Or, how to succeed in a research project and find out you were asking the wrong question
- **What's coming next?**
  - It's not only stranger than we imagine,
    It's probably stranger than we *can* imagine.
- **What does it take to do good embedded system research?**
  - What about good embedded system education?





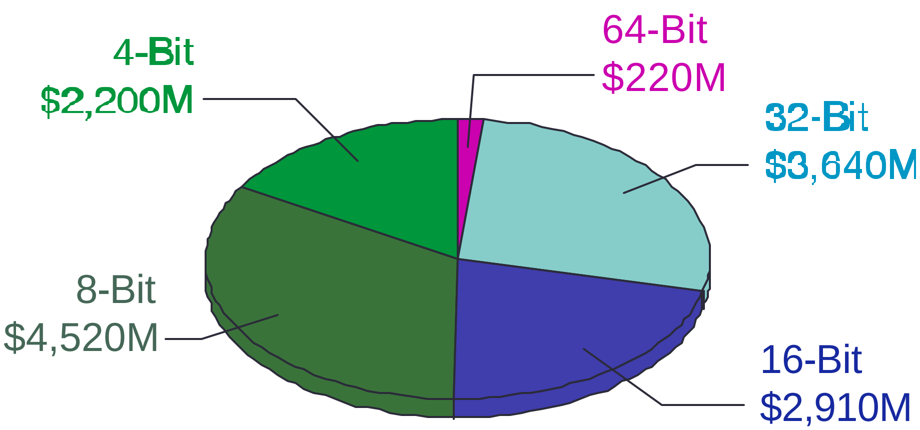
Embedded System =
Computers Inside a Product





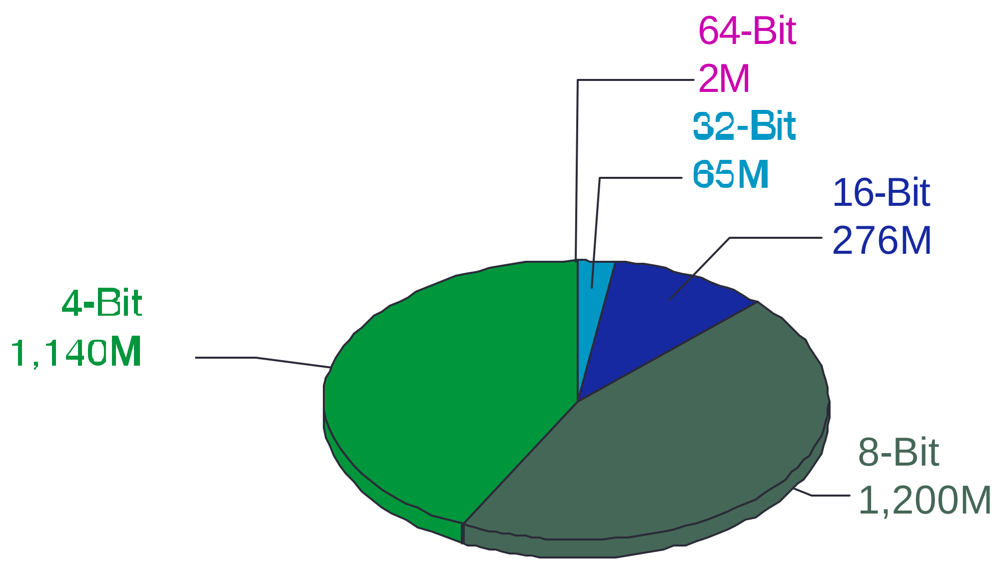
Embedded System Context

- **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

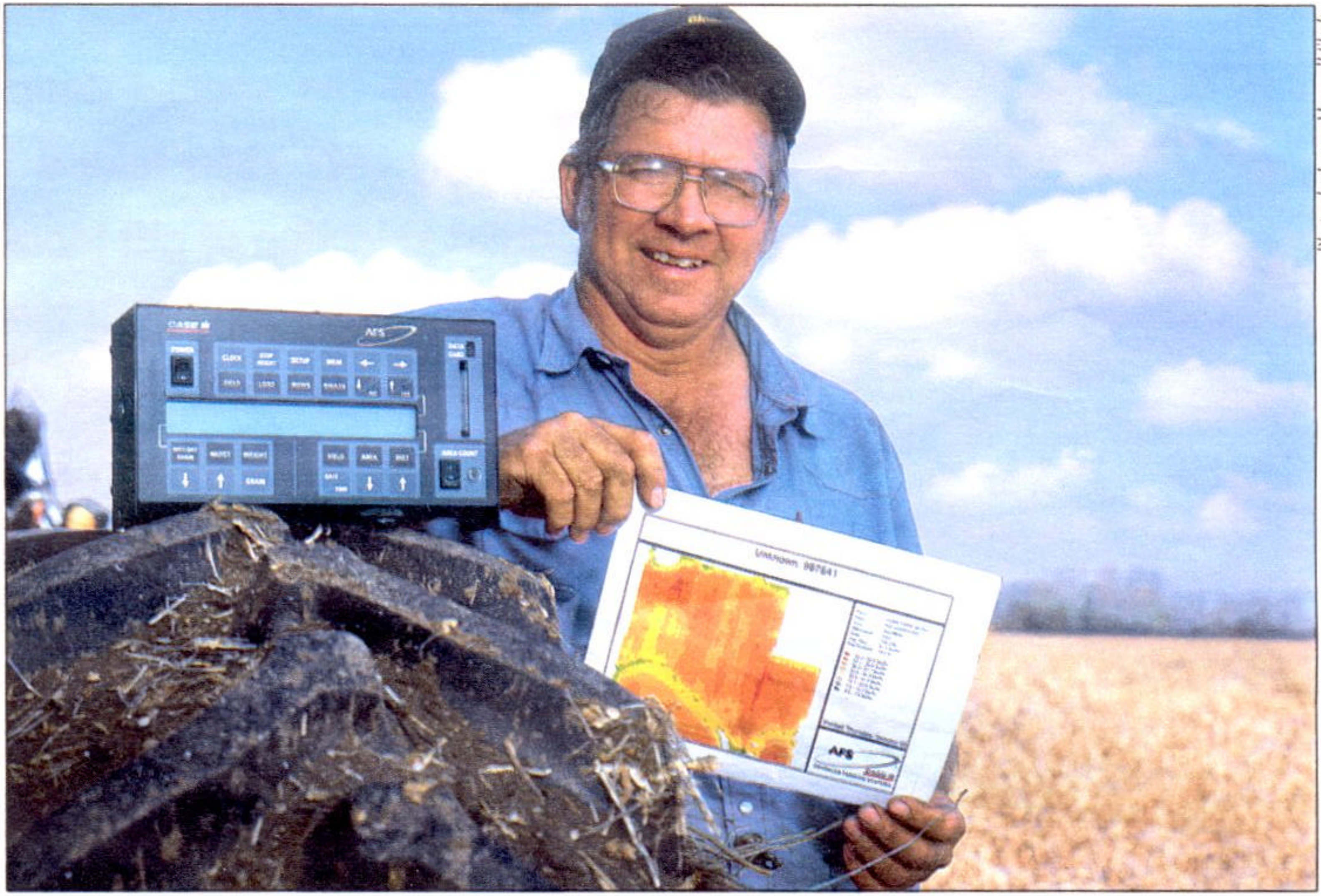
1994 Worldwide
Microcontroller Units
(Million Devices)

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

It's About the Applications, Not the Technology

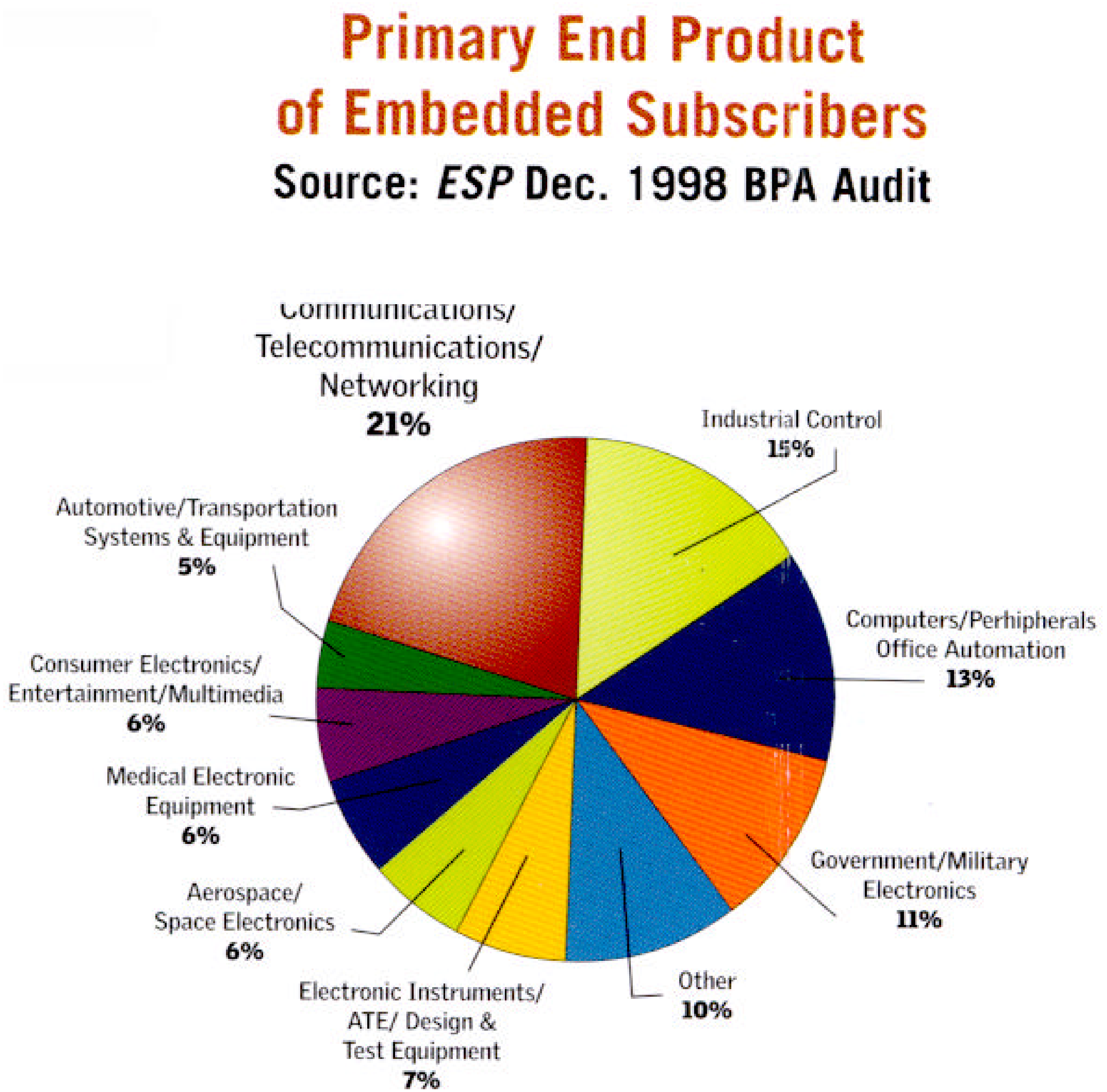
- **Technology is not the end; it is the means**
  – the goal is solving (highly constrained) problems!

"IT SURE WOULD BE MORE WORK WITHOUT COMPUTERS," SAYS A SOYBEAN FARMER WHO RELIES ON HIGH-TECH HELP FOR HARVESTING.



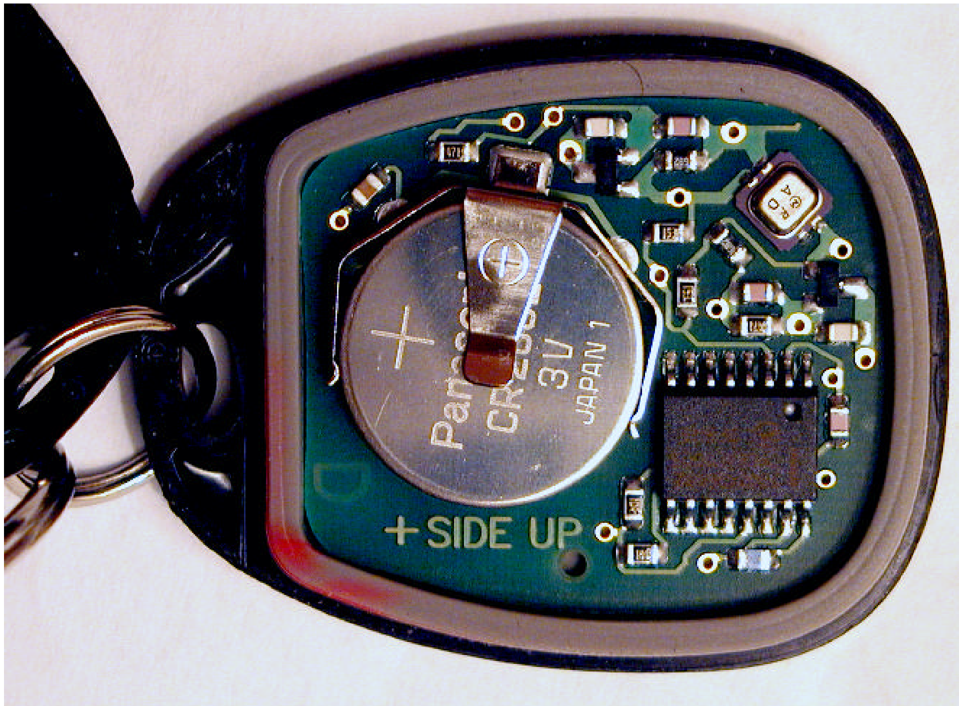
HARVESTING BEANS AND DATA. Ted Sander, 52, a farmer from Moberly, Mo., uses an onboard computer to create maps that show which plots need more fertilizer, herbicide or pesticide.

There Are Many Application Areas



Typical Embedded System Constraints

- **Small Size, Low Weight**
  - Hand-held electronics
  - Transportation applications -- weight costs money
- **Low Power**
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- **Harsh environment**
  - Power fluctuations, RF interference, lightning
  - Heat, vibration, shock
  - Water, corrosion, physical abuse
- **Safety-critical operation**
  - Must function correctly
  - Must *not* function *in*correctly
- **Extreme cost sensitivity**
  - $.05 adds up over 1,000,000 units



Why Can’t You Design Embedded Systems Just Like Desktop Systems?

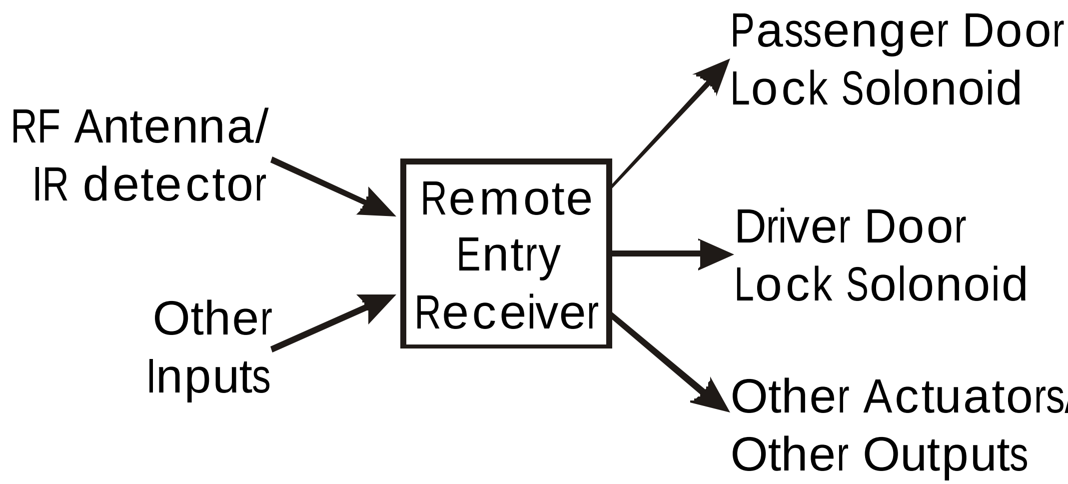
Case Study: Synthesize A Remote Entry Receiver

- **Use Fidelity: a commercial schematic synthesis tool**
  - Replicate a real automotive product design
  - Assess viability in real-world embedded system design environment

- **Note: already we are diverging from the research mainstream**
  - Most embedded system research is about chip synthesis, BUT
    most real embedded system design is about *component composition*
  - Fidelity was chosen because it is a design-by-composition tool

What's A Remote Entry Receiver?

- **RF receiver for door locks, trunk (boot), latch, *etc.***
  - 8-bit microcontroller
  - Outputs and inputs vary in:
    - Current capacity
    - Signal type
  - Very cost constrained, but must satisfy goals for:
    - Power consumption
    - Performance @ 5 MHz
    - Lifetime
    - Warranty period reliability



- **Newer functions:**
  - Transmissions encrypted
  - Monitors tire pressure
  - "Panic" alarm feature

The Experiment

- **Automotive business driven by 2-week responses to Quote Requests**
  - Engineer gets 2 weeks to estimate price
  - Bid lost if too high
  - Business gets 3 years to lose money if too low

- **Wouldn’t it be nice if you could do an optimized design in a few hours?**
  - Optimal component selection for price
  - Guaranteed to meet all constraints
  - Generates input to PCB layout tools

- **Wouldn’t it be nice if you could re-design monthly for cost savings?**
  - But, can a CAD tool really match super-macho embedded system engineers?

- **Fidelty promised it could do all that**
  - So, let’s see if it really can

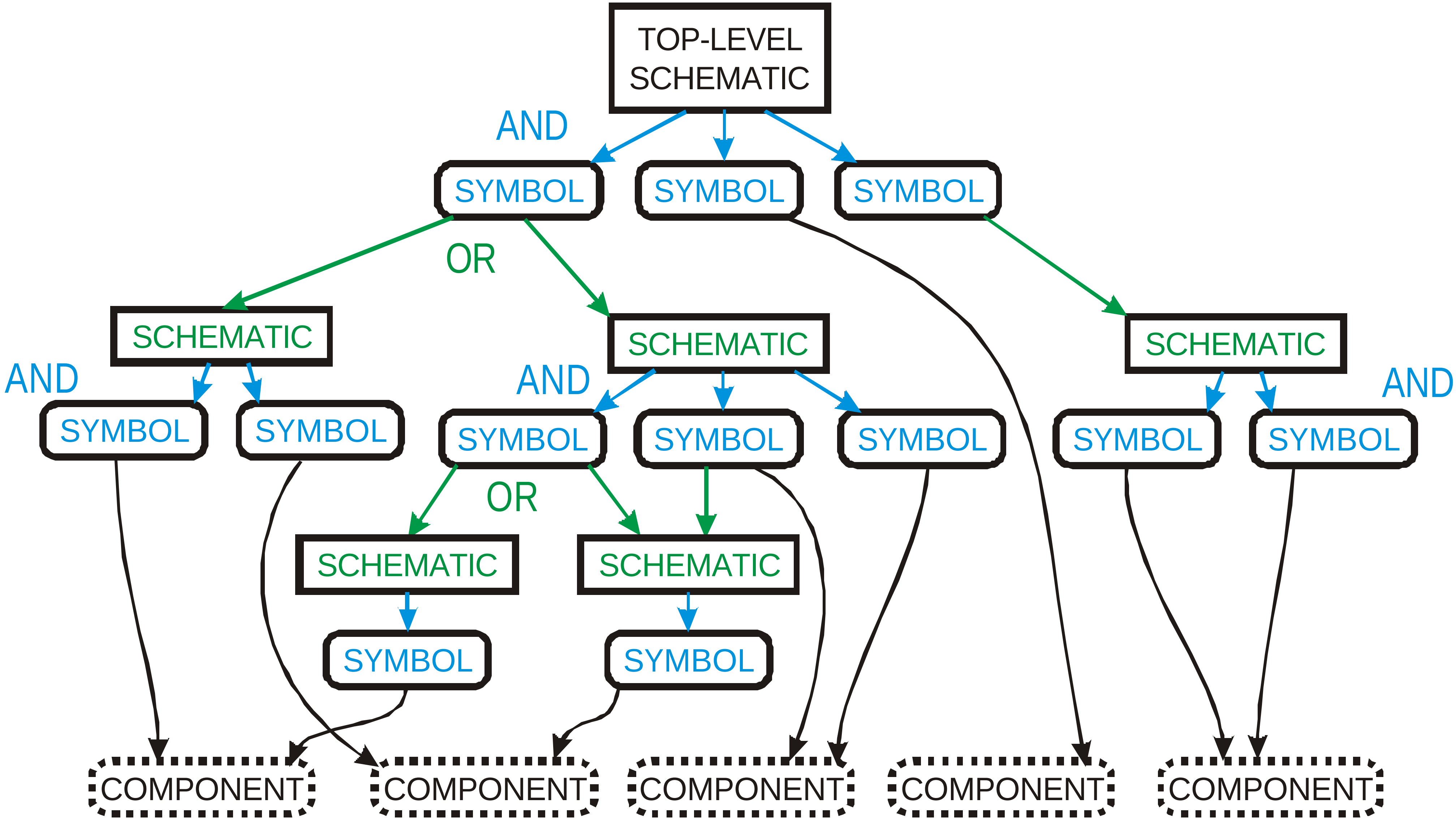
Fidelity Tool Details

- **Design-by-composition tool from Omniview, Inc.**
  - Commercialization of Carnegie Mellon Micon tool
  - Designed to automated PC motherboard synthesis, and it's good at that
  - Arbitrary synthesis from equations is not the point (it's not Verilog/VHDL)

- **Schematic hierarchy in Mentor Graphics tool set used**
  - Each "symbol" can link to *multiple* child "schematics"/(components)
  - Exactly *one* such schematic is used in any given design instance

Fidelity Design Representation

- **Represents all known components/subsystems**
  - Searches for optimal combination that meets constraints

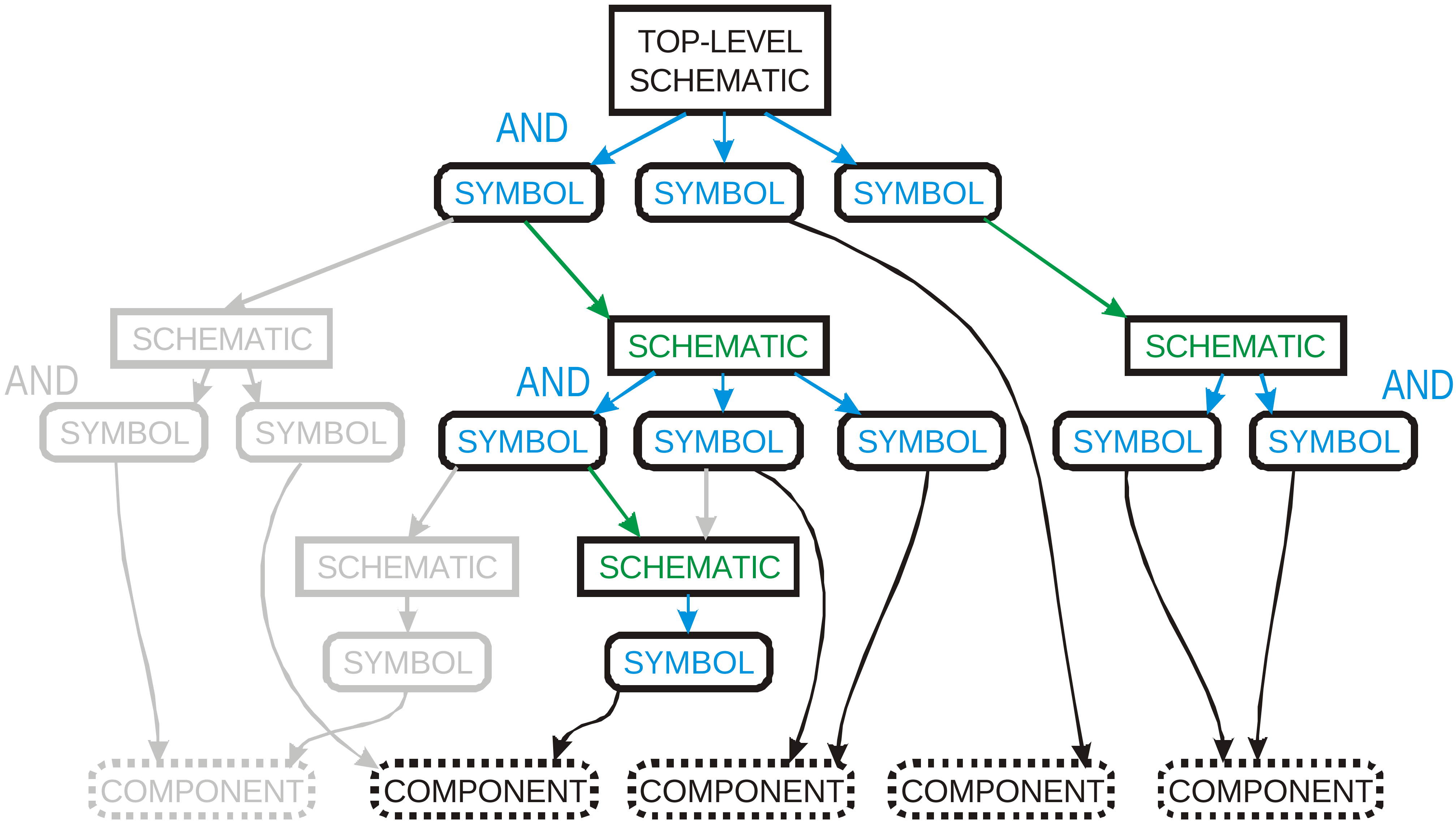


Design Constraints *etc.*

- **“Design equations” communicate constraints within hierarchy**
  - Values or value ranges can state power, signal, voltage requirements
  - Interval arithmetic inequalities can specify analog circuit parameters
- **Global constraints can be used to filter designs**
  - Power
  - Cost
  - One or two other user-defined global constraints

Fidelity Design Result

- **Select optimal set of schematics (design options) given constraints**
  - Picks exactly one schematic/component per symbol



Did It Work?

- **Yes, it was able to find optimal design points**
  - Reproduced hand-done designs using component database
  - Used design-by-selection, which was required (synthesized designs undesirable because of NRE and lead time issues)
- **But it was not able to meet all the other requirements!**
  - Additional engineering constraints
  - Business constraints
  - Cultural issues

Lessons Learned: Electronic Design

- **Digital, analog, and power components**
  - There is often only one digital component (a microcontroller)

  Embedded designs interface to an analog world!

- **Digital design vs. digital component selection**
  - Standard components are used for cost, flexibility & cycle time
  - Digital design consists of selecting a microcontroller, not IC synthesis

  Selecting components may be more important than synthesizing them.

- **Incremental design updates**
  - Want minimum manufacturing disruption for updates, not complete redesign
  - Ideally, all design changes are 100% in software

  Redesign needs to limit scope of changes, not seek perfect optimality

Lessons Learned: System Design

- **Design margin & customer variation**
  - Some customers want it “cheap”, others want it “good”
  - Customer-specific input protection circuits, *etc.* (need product families)
    - This was easily handled with design equations
    - Variations also occurred per country of sale per manufacturer
  - ASICs undesirable; customer changes requirements several times/year

  Designs must be tailored and change regularly; investment in ASICs is sometimes impractical

- **Clock speed limitations**
  - Receiver CPU limited to 5 MHz by RFI concerns (RF interference)
  - Transmitter limited to 1 MHz(!)
  - Cryptographic algorithms were tailored to minimize clock cycles & memory

  Faster raw clock rates may not help at all due to RFI & power limitations

Lessons Learned: Business & Process

- **Lifecycle component cost is more complex than quantity-1 cost:**
  - Volume-purchasing discounts
  - Cost of purchasing dept. time for each component type
  - Cost of component qualification
  - Cost of vendor qualification
  - Cost of component database maintenance
  - Cost of logistics (spare parts, warehousing, *etc.*)
  - Limited number of component bins on pick&place equipment

  Use minimum number of component types across all products.

- **System certification and lifecycle costs can dominate**
  - All changes must be vetted by customer (warranty cost concerns)
  - Many changes must undergo FCC recertification
  - Many changes require a new shake&bake life test

  Weigh potential benefits against validation & certification costs;

  Don't underestimate cost of recertifying a critical system for a "minor" change

More Business & Process Lessons

- **CAD tool proficiency matters**
  - Engineers assigned to products, not engineering functions
  - CAD tools have a steep learning curve; expertise evaporates clearly
  - Elite corps of CAD experts isn't viable due to turnover, cost

  Complex digital CAD tools may not be viable in many situations

- **Model & library database maintenance**
  - Who updates the price information?
  - Companies use internal part numbers, requiring format & number translation
  - Who polices database quality?
    - Do you want to go bankrupt because someone mis-typed a component price?

  Infrastructure costs can be significant when using design tools

- **Legacy designs & understandability**
  - Deep hierarchies for decoupling design issues don't print well
  - Archives are all on paper (for good reason)

  CAD designs still have to be printed for long-term records

Cultural Issues

- **Compelling advantage required to change current practices**
  - If they can build products today, why should they change?
  - "Engineers are free" paradox - why buy them a $50K tool?

  Compelling advantage required. In this case design-to-quote cycle time was a very good incentive.

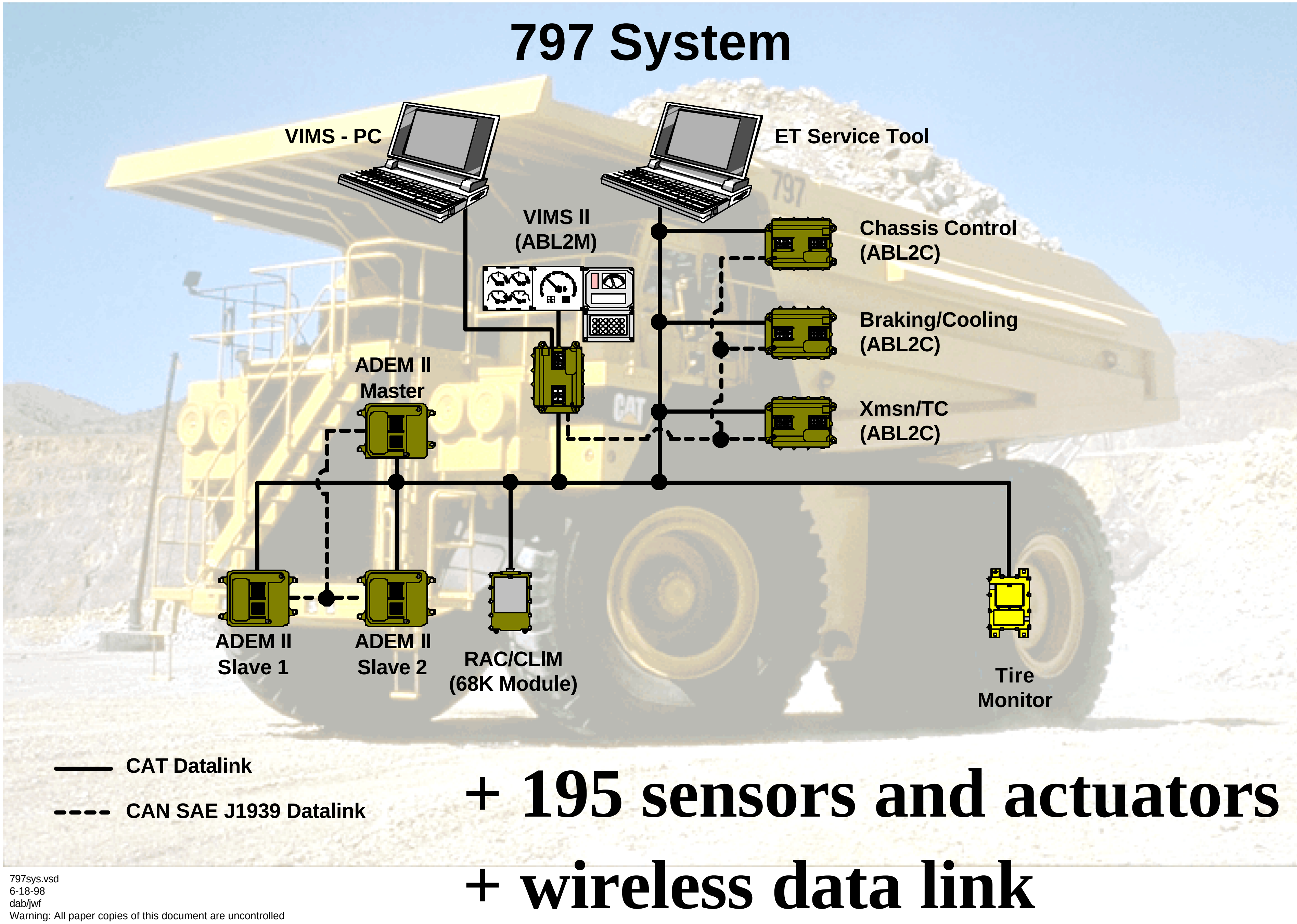
- **Computer culture vs. "metal-bending" cultures**
  - Non-computer engineers may not appreciate (or even believe in) simulation-based design methods
  - Computers are a small part of embedded systems (weight, size, to some degree cost)
    - But, some companies are waking up to the fact that their main cost is bending software instead of metal.
  - It's the system that matters, not the whizziness of the technology (usually)

  Things we take for granted become major battles in embedded applications

What Does The Future Look Like?

Today:

Embedded + Distributed – Caterpillar 797



+ 195 sensors and actuators

+ wireless data link

Tomorrow: Embedded Computers *Everywhere*

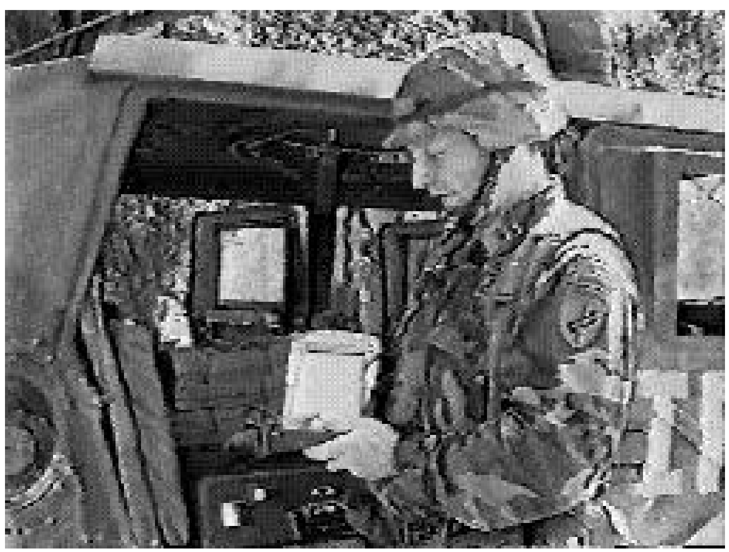
- **Sewing Machines**

- **Transportation**
- **Consumer Electronics**
- **Concrete (sensors)**
- **Clothing(?)**

- **Home Appliances**

Computer Fridge

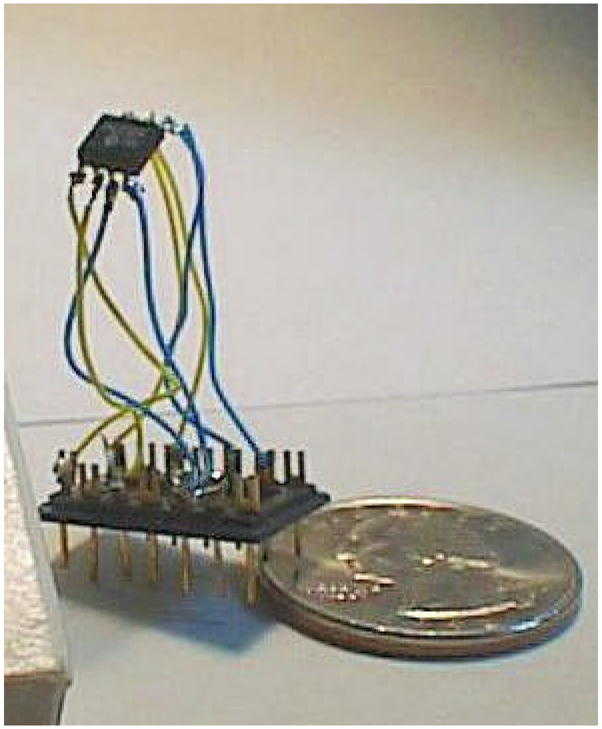
- **Communications & Translation**



The Future(?)

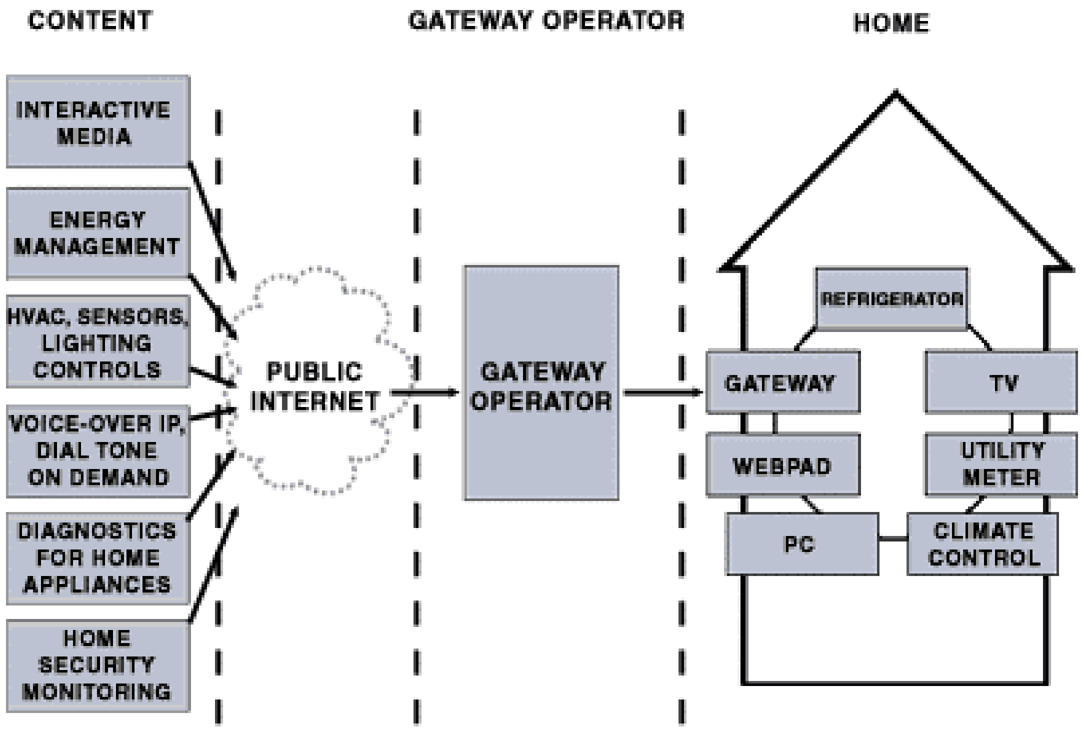
- Every time I hear a far fetched idea, I can find a web page with a photo of a prototype or product

Embedded web server



Digital Frying Pan

Sun’s Version of the Wired House



HOME GATEWAY USAGE SCENARIO

- **Will people adopt this other than as a toy?**
  - Will the same people who can’t set time on a VCR be able to debug their house?

- **If we can make the system readily accessible, reliable, affordable, …the possibilities are almost endless**

Would *You* Drive A Car In Which:

"THE SOFTWARE is provided 'AS IS' and with all faults. THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY, AND EFFORT (INCLUDING LACK OF NEGLIGENCE) IS WITH YOU."

(You will.)

- **Virtually all embedded OS vendors are requiring end-user licenses with liability waivers (and they're already legally binding in some states!)**

Research & Education

Educational Issues

- **Embedded *system* engineers are more generalists in an age of specialization**
  - Multi-disciplinary tradeoffs, often with design team size of 1 engineer

- **Need education way beyond traditional A/D, D/A, and assembly:**
  - Real time operating systems & scheduling
  - System design methodologies (requirements / design / test / *etc.*)
    - Many engineers need software/system engineering literacy
  - Distributed systems & distributed networks
    - Entirely different set of tradeoffs for embedded than for "regular" networks
  - Architectural approaches to distributed systems
  - Critical system design (dependability, safety)
  - Human/computer interfaces
  - Specialty skills: low power, design for particular constraints

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 targeting a $1 chip
  - Time to market
  - Cost

Relative Embedded System 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?

#4 - Security

- Do you want someone hacking your digital wallet?

…

#837 - 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?)

Pressing Research Topics

- **System level tradeoffs.**
  "System" =
  - Digital hardware + Analog hardware
  - Software
  - People/operators
  - Mechanical components
  - Life cycle support/logistics -- *trade off from transistors to business process*

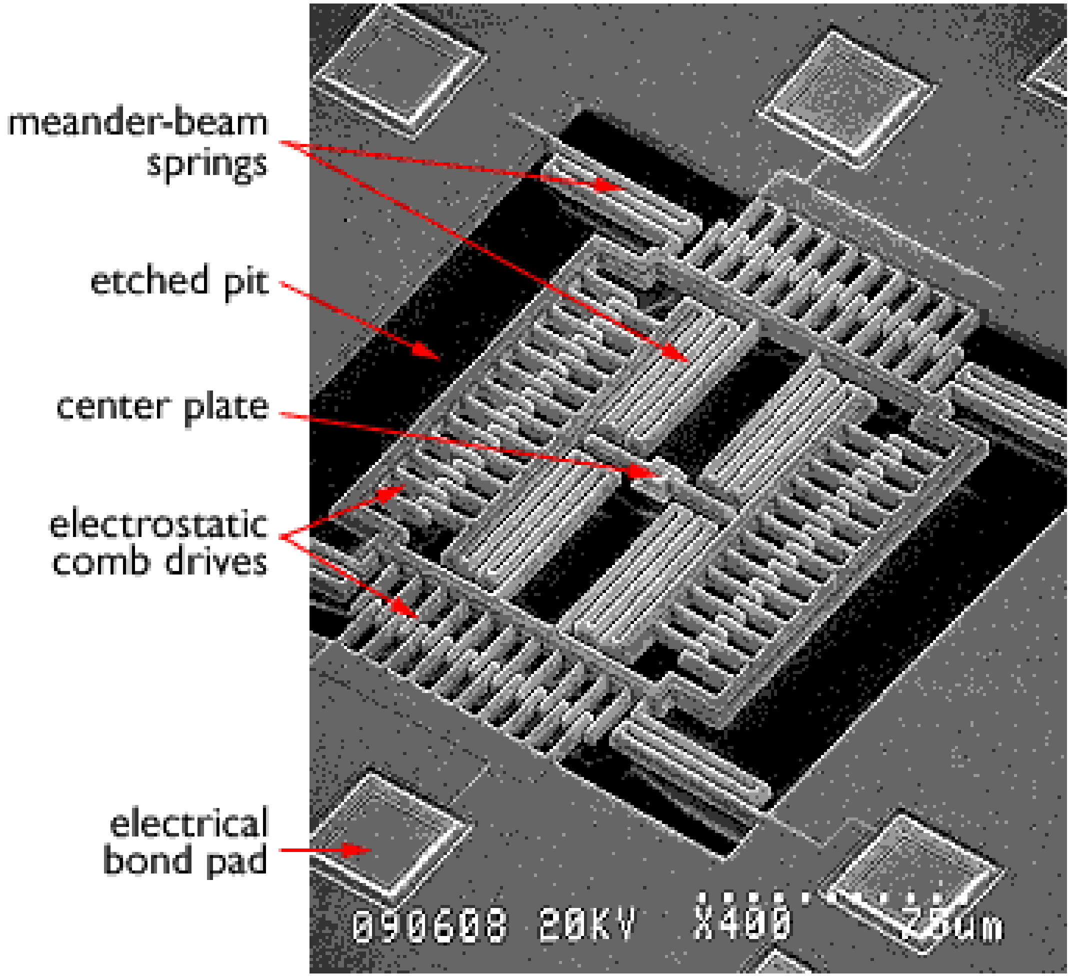
- **Affordable dependability**
  - How can we trust our lives to a $1 microcontroller? (we will…)
  - How can we get a clue about making dependable software for less than $1M

- **Design for embedded constraints**
  - Hard real time
  - Harsh environments
  - Low cost security
  - Low power
  - Small memory footprints
  - *etc.*

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



RoSES: Robust Self-Configuring Embedded Systems

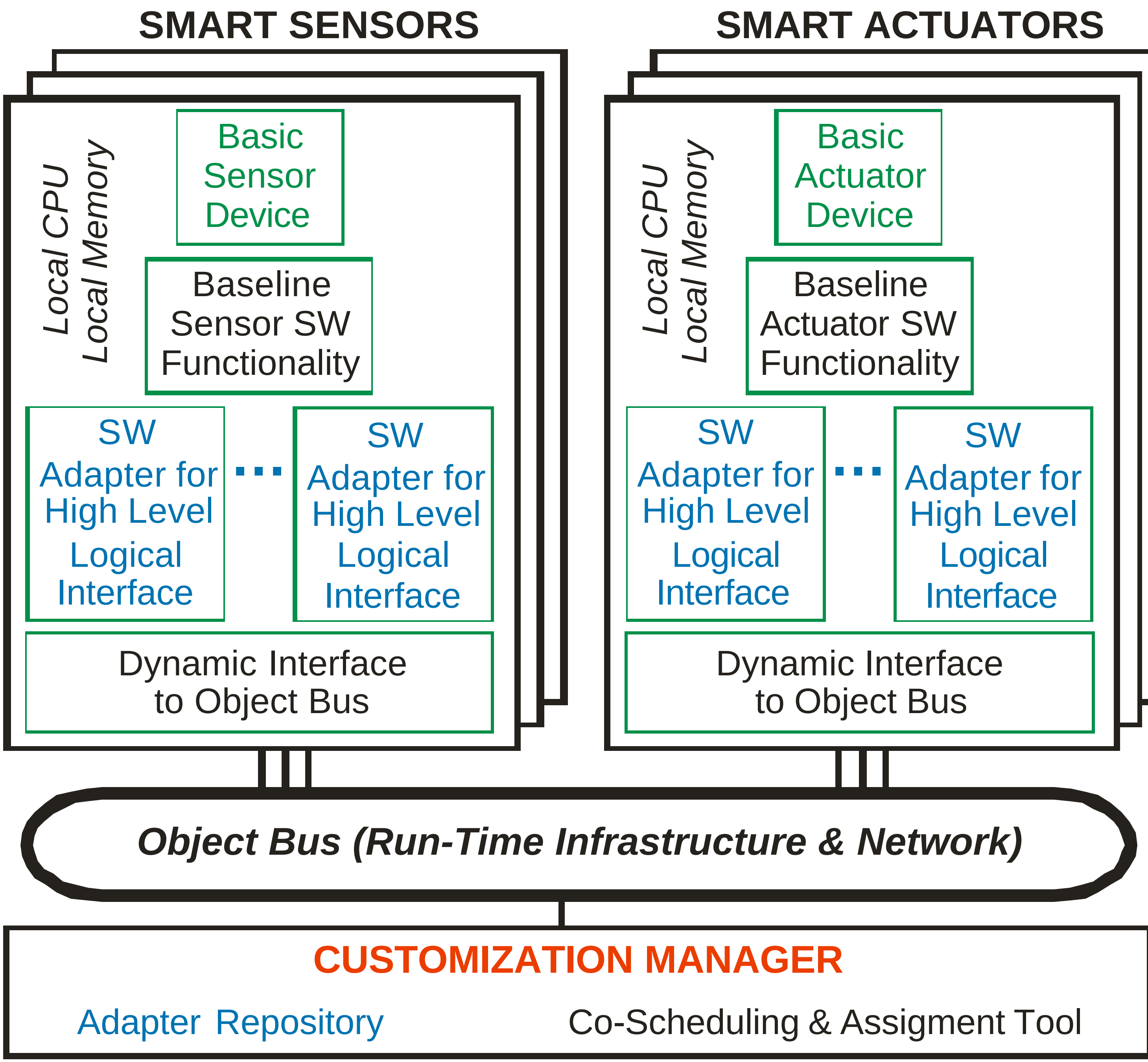
- **Product families + automatic reconfiguration =**
  - Operation with failed components
  - Automatic integration of inexact spares
  - Automatic integration of upgrades
  - Fine-grain product family capability

- **Potential Impact:**
  - Logical component interfaces + configuration mgr.
  - Fine-grain software component run-time support
  - Architectures that are naturally resilient

- **First demos in late 2001**

RoSES

Generic RoSES System Architecture



Conclusions

- **What's an embedded system?**
  - Contains computers that interact with the real world
  - Pretty soon, it may be everything!
- **Why can't you just design them like desktop systems?**
  - Design constraints can be much tighter (cost, size, power, speed, …)
  - Life cycle effects are far more important than the disposable PC market
  - Software can kill people in these systems
- **What about embedded system research & education?**
  - It's about the *system*!
  - Requires broad perspective, multidisciplinary tradeoffs, and attention to the "ilities"