18-540 Distributed Embedded Systems

Prof. Philip Koopman

Fall, 2000

Lecture: Mon/Wed 12:30-2:20 PM -- PH A18A

Recitations: Fridays 1:30-2:20 -- PH A18C

Recommended Text: Kopetz, *Real-Time Systems: design principles for distributed embedded applications*, Kluwer Academic Publishers

Carnegie Mellon

18-540 Distributed Embedded Systems

- **Based on lecture notes & practitioner-oriented papers**
  - Book offers additional info, but is not testable beyond lecture coverage
- **Course objectives detailed on web pages**
  - System Engineering
    - Requirements, design, verification/validation, certification, management-lite
  - System Architecture
    - Modeling/Abstraction, Design Methodology, Business Issues
  - Embedded Systems
    - Design Issues, scheduling, time, distributed implementations, performance
  - Embedded Networks
    - Protocol mechanisms, performance, CAN, TTP, embedded Internet
  - Critical Systems
    - Basic Techniques (FMEA), software safety, network safety, certification, ethics, testing, graceful degradation
  - Case Studies
    - Elevator as lab project, guest speakers and other discussions

Policies

- **See http://www.ece.cmu.edu/~ece540 for official versions**

- **Grading: straight scale A≥90; B≥80; C≥70; D≥65**
  - Three in-class tests: 20% each (no final exam)
  - Homeworks: 15%; lowest one dropped; due at 4 PM
  - Course project: 25% (a few bonus points are competitive)
  - Homework & project late penalty: grade is multiplied by $0.9^{\#days\_late}$

- **Slightly unusual policy on cheating**
  - Discussion of homework & project problems is acceptable and encouraged
    - But no direct copying
    - Information must stay in your brain for at least 5 minutes before you put it down on your own paper
  - Researching techniques is good, but cite source if beyond class coverage
  - Test are closed book; one 8.5”x11” sheet (2 sides) of paper notes permitted, bring your own calculator and pencils

- **Office hours: after every lecture and as posted**

Classroom Protocol

- **Please arrive on time; lecture begins promptly**
  - I also promise to end on time
  - Please put extra handouts in pile by door for the few latecomers
- **Questions are encouraged**
  - If you don't understand, ask, because probably other students are struggling too
  - Sometimes a long answer will be deferred to recitation or office hours
  - Philosophical questions are welcome in remaining time at end of class
- **There is no way to cover everything**
  - Embedded systems is a huge area; we will cover things I've found to be core topics in industry
  - I'm electing to cover fundamentals rather than latest fad topics (little emphasis on internet toaster ovens in this course)
  - There is a "digging deeper" section for each lecture if you want to expand what you are learning

1
Embedded Systems in the Real World

18-540 Distributed Embedded Systems

Philip Koopman

August 28, 2000

Required Reading: **Tennenhouse, "Proactive Computing"**

Carnegie Mellon

Assignments

- **Reading for this lecture**
  - *Required:* Tennenhouse, “Proactive Computing”

- **Reading for next lecture**
  - *Required:* Koopman, “Embedded System Design Issues: the rest of the story”
  - *Recommended:* Kopetz Chapter 1
  - *Required:* Lecture is always on-line ahead of time as a preview

- **Homework #0 due via e-mail Friday 9/1 at 4 PM**
  - (Most other homeworks will be hard-copy; projects will be on-line submissions)

- **Project #1 due Wednesday 9/13 at 4 PM**
  - Assemble project groups by this Friday per HW #0
  - Waiting until the night before is a bad idea

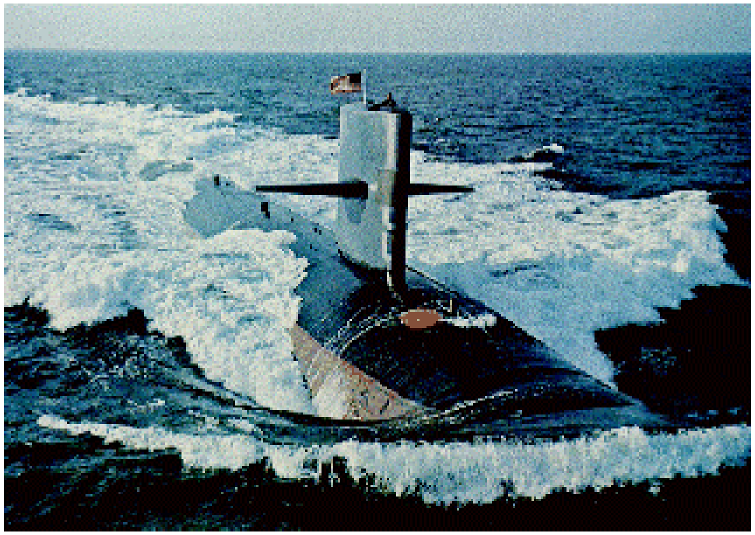
Where Are We Now?

- **Where we're going today:**
  - General discussion of embedded systems
    (they're not the same as desktop or "general purpose" computers)

- **Where we're going next:**
  - Details of embedded+real-time+control systems
  - Elevator as a detailed example (basis of course project)

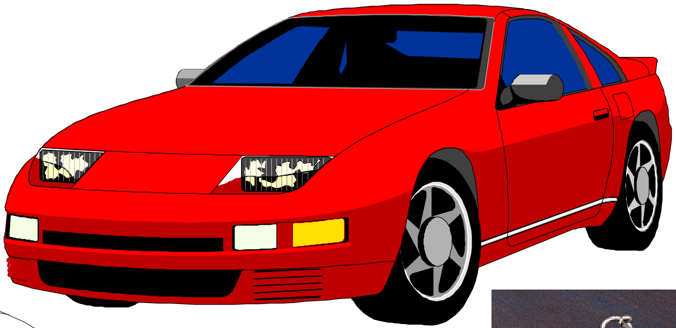
Preview

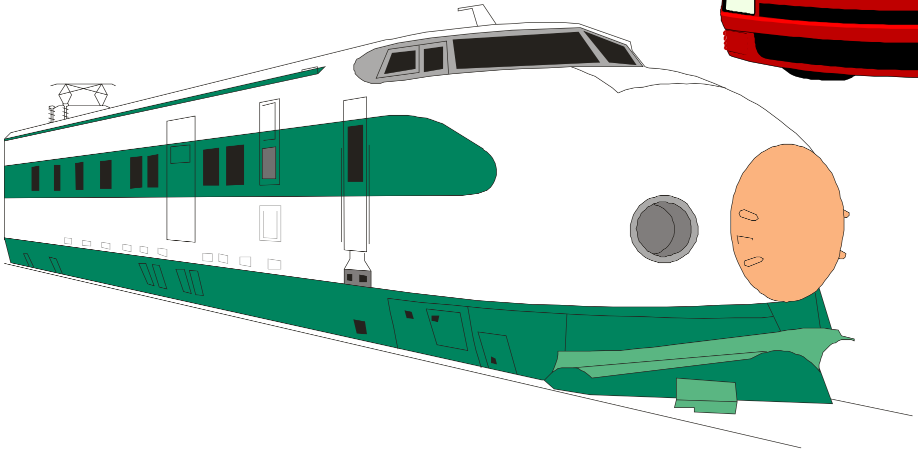
- **What is an embedded system?**
  - More than just a computer
- **What makes them different?**
  - Real time operation
  - Many sets of constraints on designs
- **What embedded system designers need to know**
  - The big picture
  - Skills required to “play” in this area



Embedded System =

Computers Inside a Product





Definition of an Embedded Computer

- **Computer purchased as part of some other piece of equipment**
  - Typically dedicated software (may be user-customizable)
  - Often replaces previously electromechanical components
  - Often no “real” keyboard
  - Often limited display or no general-purpose display device

- **But, every system is unique -- there are always exceptions**

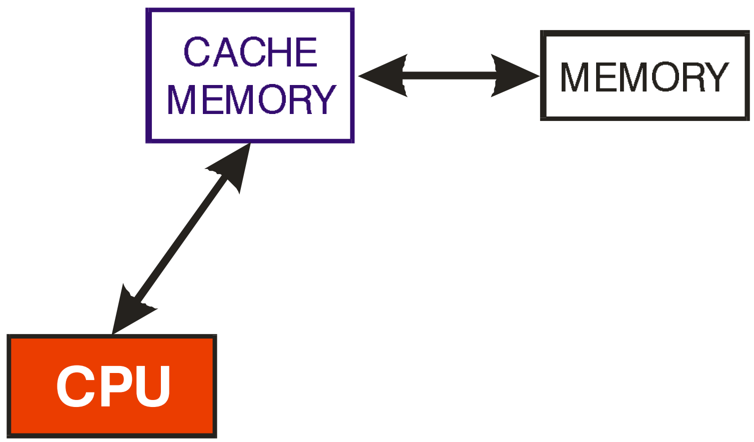
- **Course scope focuses on distributed embedded systems, and not other embedded areas such as:**
  - Military systems: Radar, Sonar, Command & Control
  - Consumer electronics: set-top boxes, digital cameras
  - Telecommunications/DSP: cell phones, central office switches
  - Robotics

An All-Too-Common View of Computing

- Measured by: Performance

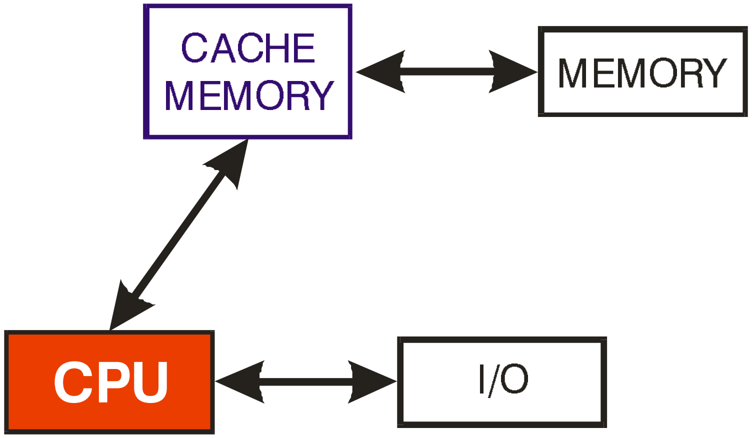
An Advanced Computer Engineer’s View

- **Measured by: Performance**
  - Compilers matter too…



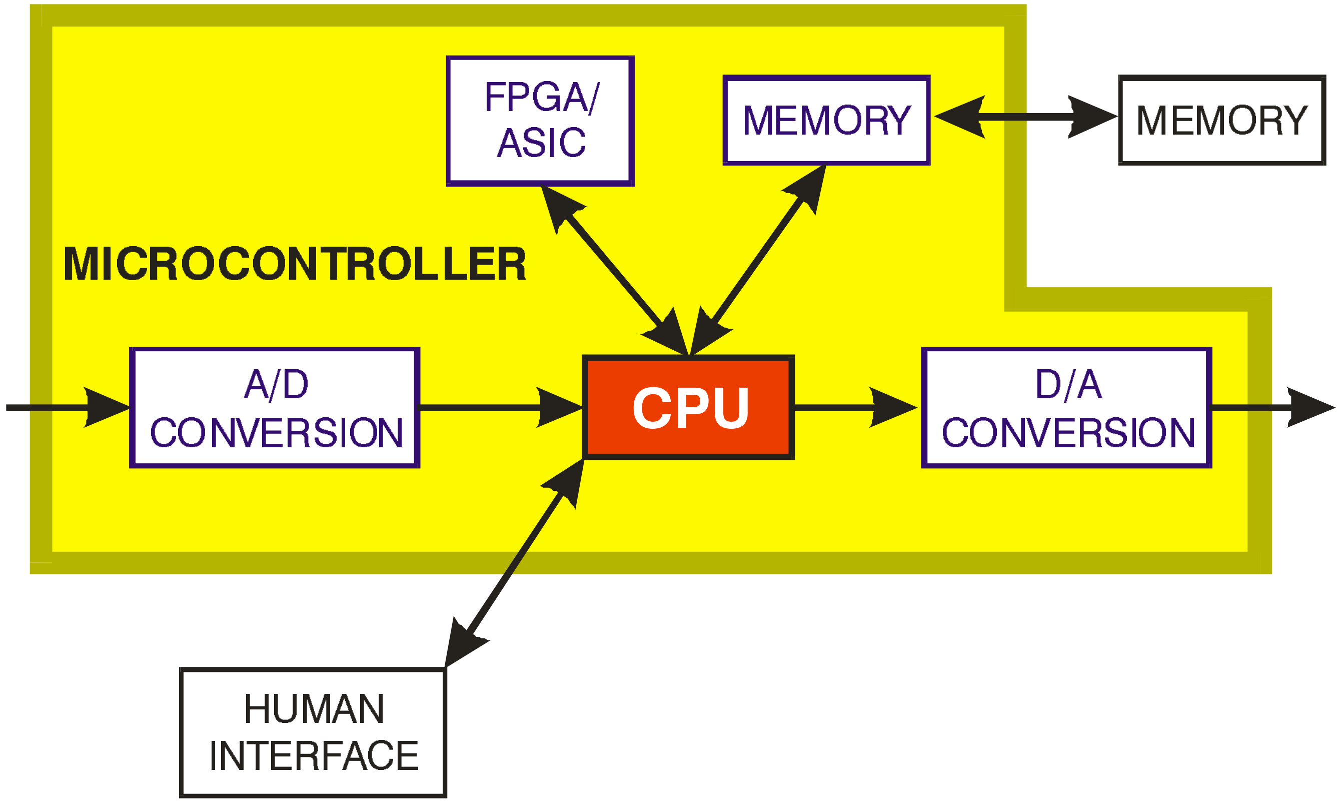
An Enlightened Computer Engineer's View

- **Measured by: Performance, Cost**
  - Compilers & OS matter



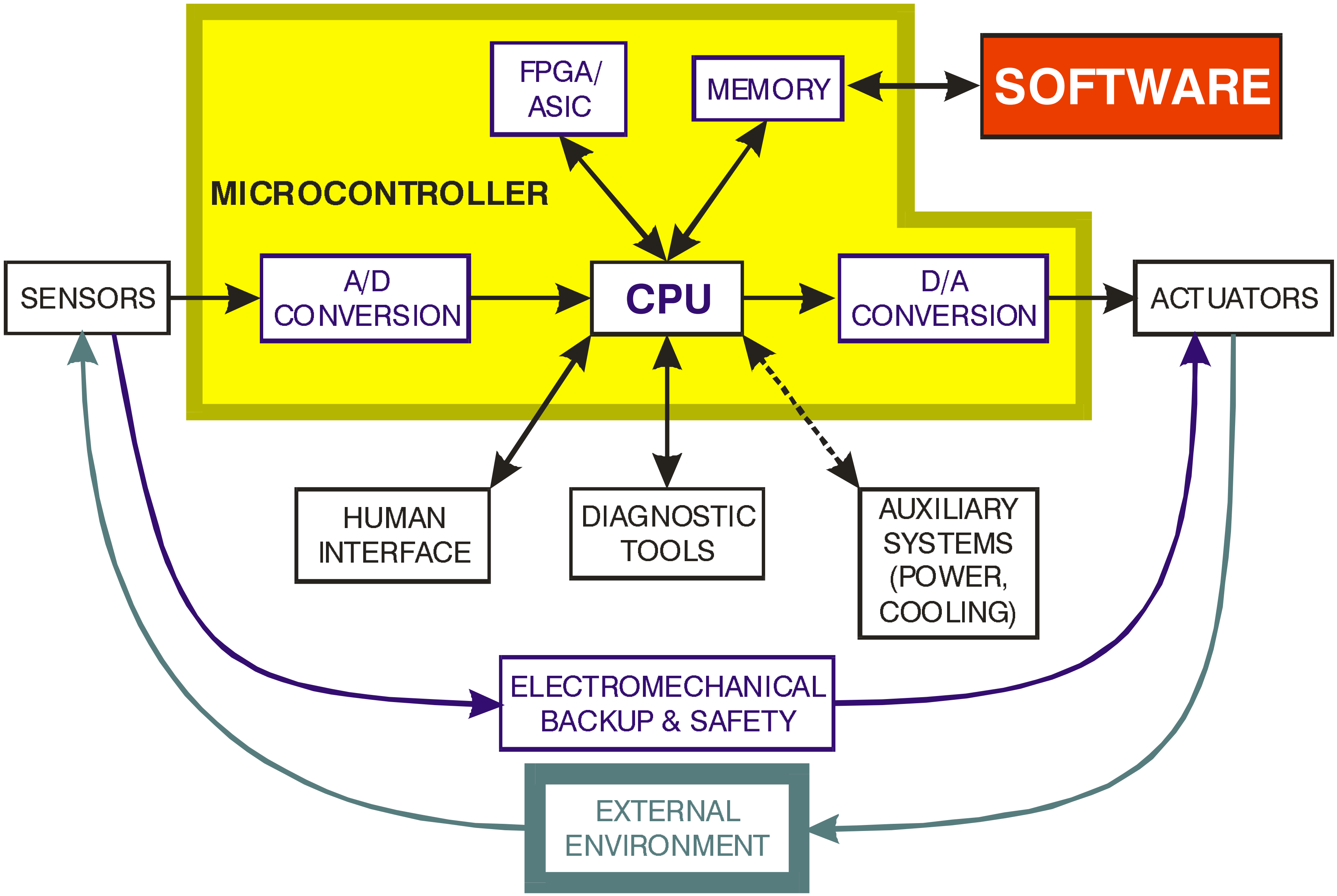
An Embedded Computer Designer's View

- **Measured by: Cost, I/O connections, Memory Size, Performance**



An Embedded Control System Designer's View

- Measured by: Cost, Time-to-market, Cost, Functionality, Cost & Cost.

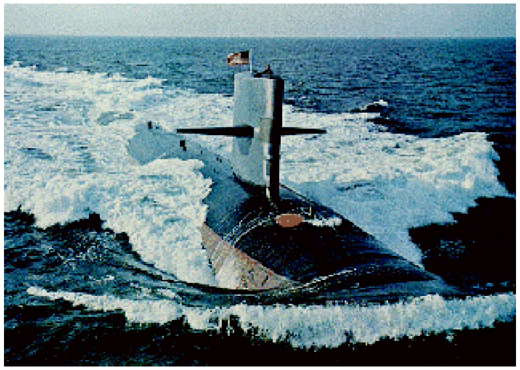


Three Embedded Examples

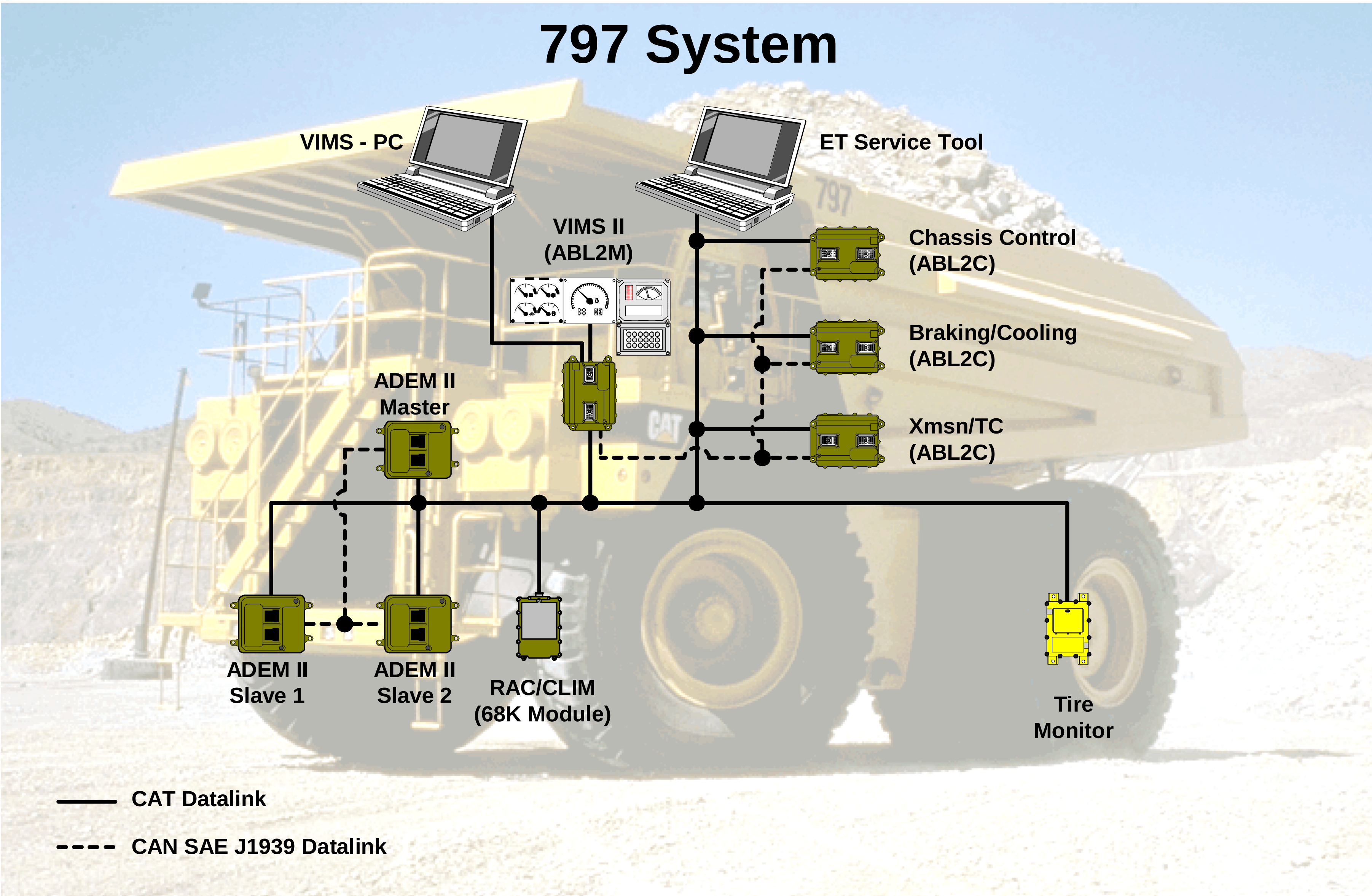
- **Pocket remote control RF transmitter**
  - 100 KIPS, water/crush-proof, small, 5-year battery life
  - Software hand-crafted for small size (less than 1 KB)

- **Industrial equipment controller (e.g., elevator; jet engine)**
  - 1-10 MIPS for 1 to 10 CPUs, 1 - 8 MB memory
  - Safety-critical software; real-time control loops

- **Military signal processing (e.g., Radar/Sonar)**
  - 1 GFLOPS, 1 GB/sec I/O, 32 MB memory
  - Software hand-crafted for high performance



Embedded + Distributed – Caterpillar 797



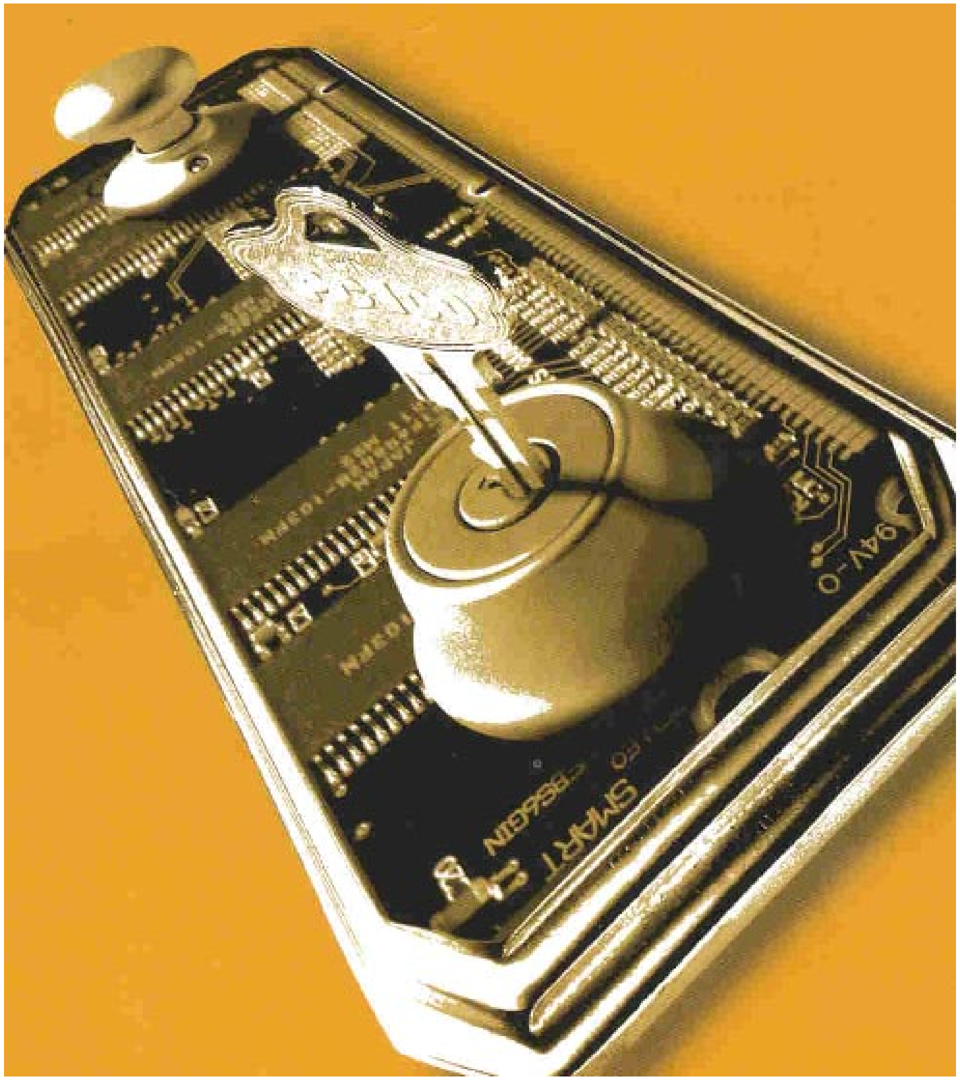
797sys.vsd
6-18-98
dab/jwf
Warning: All paper copies of this document are uncontrolled

A Customer View

- Reduced Cost
- Increased Functionality
- Improved Performance
- Increased Overall Dependability
  - (Debatable, but can be true)

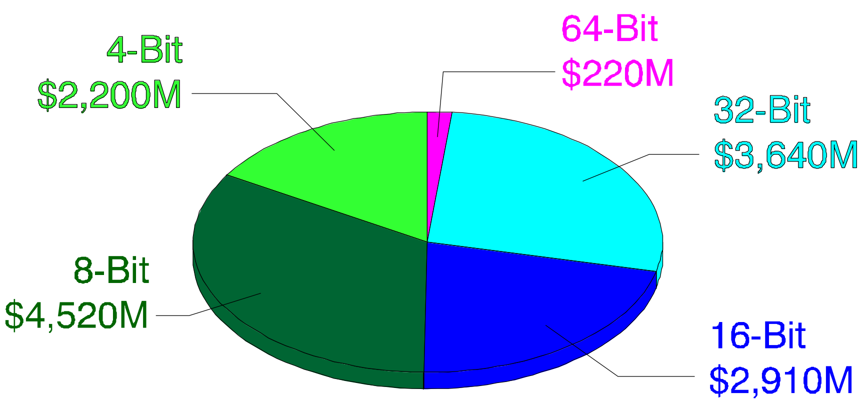
What in the world are you going to do with all those computers? It's not as if you want one in every doorknob!

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



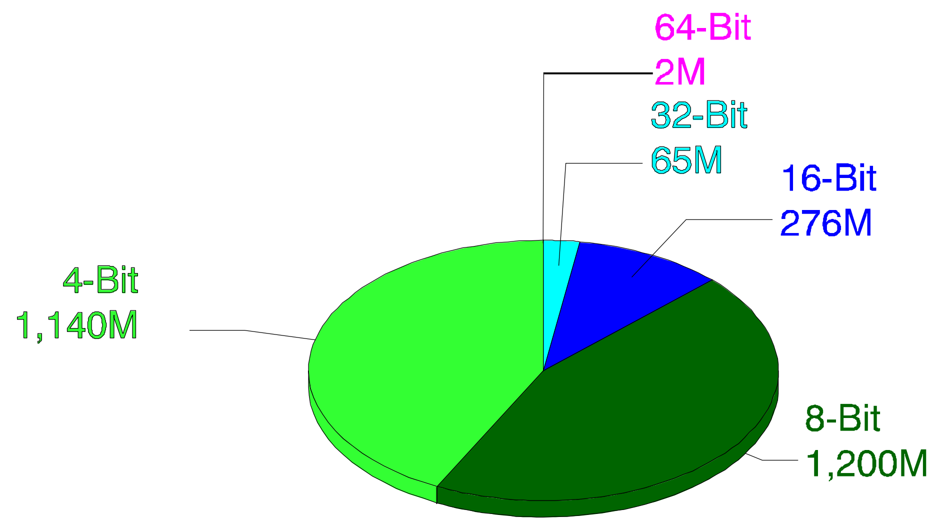
Small Computers Rule The Marketplace

- **~80 Million PCs vs. ~3 Billion Embedded CPUs Annually in 1995**
  - 150 Million PCs and 7.5 Billion embedded CPUs + in 2000



$13,490M Total

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

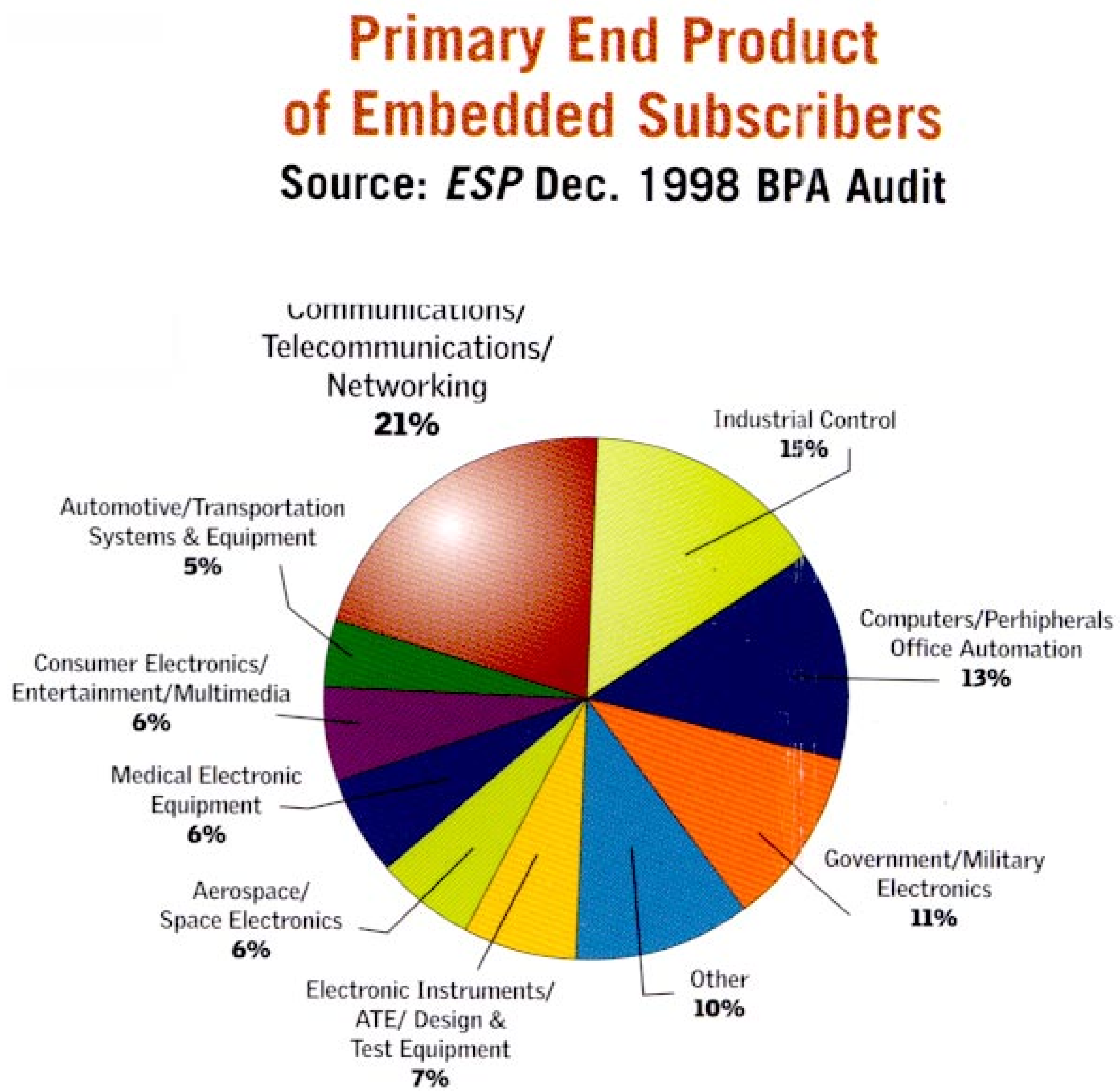


2,683M Total

1994 Worldwide
Microcontroller Units
(Million Devices)

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

There Are Many Application Areas



Four General Embedded System Types

- **General Computing**
  - Applications similar to desktop computing, but in an embedded package
  - Video games, set-top boxes, wearable computers, automatic tellers
- **Signal Processing**
  - Computations involving large data streams
  - Radar, Sonar, video compression
- **Communication & Networking**
  - Switching and information transmission
  - Telephone system, Internet
- **Control Systems**
  - Closed-loop feedback control of real-time system
  - Vehicle engines, chemical processes, nuclear power, flight control

Types of Embedded System Functions

- **Control Laws**
  - PID control
  - Fuzzy logic, …
- **Sequencing logic**
  - Finite state machines
  - Switching modes between control laws
- **Signal processing**
  - Multimedia data compression
  - Digital filtering
- **Application-specific interfacing**
  - Buttons, bells, lights,…
  - High-speed I/O
- **Fault response**
  - Detection & reconfiguration
  - Diagnosis

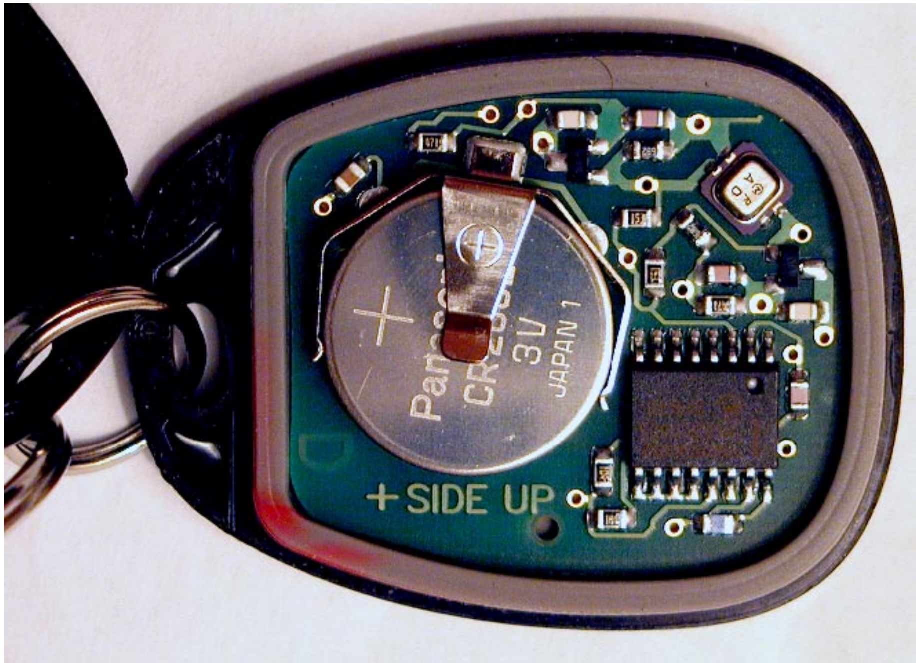
Distinctive Embedded System Attributes

- **Reactive: computations occur in response to external events**
  - Periodic events (*e.g.*, rotating machinery and control loops)
  - Aperiodic events (*e.g.*, button closures)

- **Real Time: correctness is partially a function of time**
  - Hard real time
    - Absolute deadline, beyond which answer is useless
    - (May include minimum time as well as maximum time)
  - Soft real time
    - Approximate deadline
    - Utility of answer degrades with time difference from deadline
  - In general Real Time != "Real Fast"

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



Embedded System Design World-View

- **A complex set of tradeoffs**
  - Optimize for more than just speed
  - Consider more than just the computer
  - Take into account more than just initial product design

Multi-Objective

- Dependability
- Affordability
- Safety
- Security
- Scalability
- Timeliness

×

Multi-Discipline

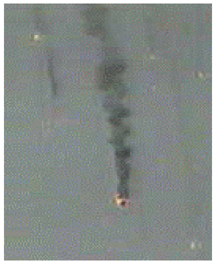
- Electronic Hardware
- Software
- Mechanical Hardware
- Control Algorithms
- Humans
- Society/Institutions

×

Life Cycle

- Requirements
- Design
- Manufacturing
- Deployment
- Logistics
- Retirement

Mission-Critical Applications Require Robustness



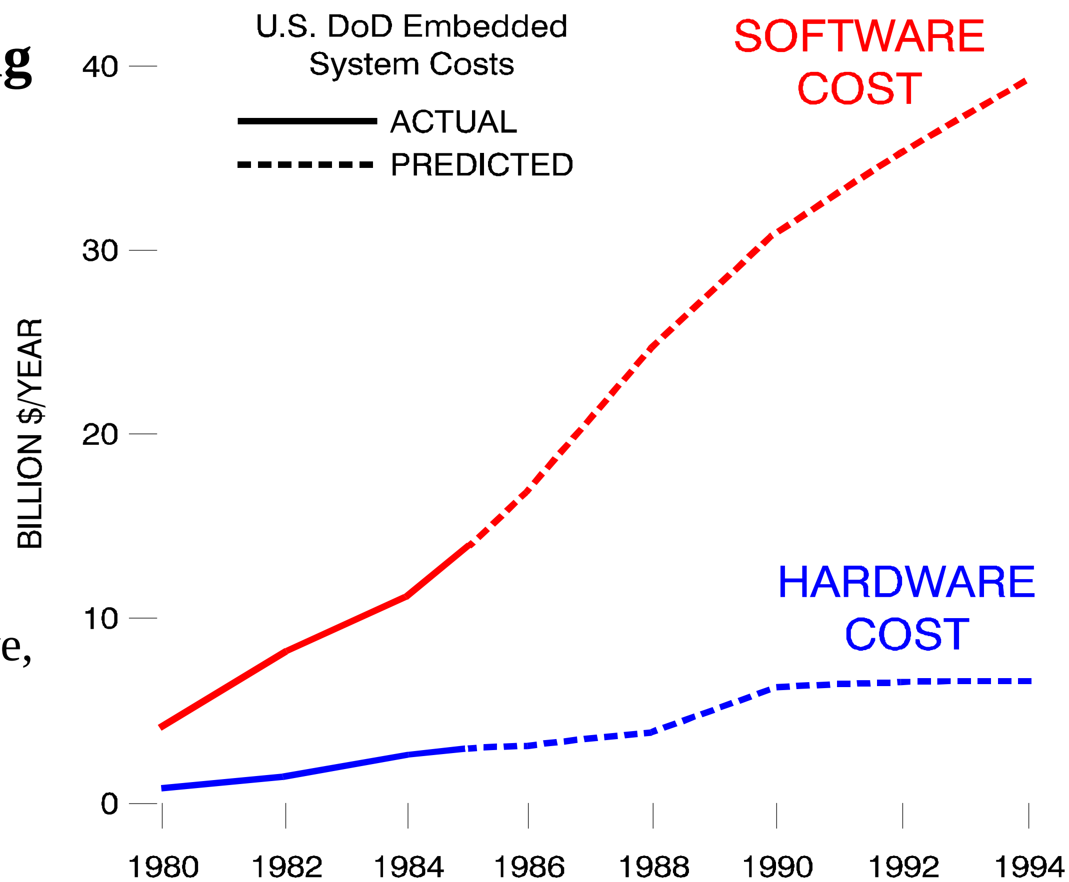
- **June, 1996 loss of inaugural flight**
  - Lost $400 million scientific payload (the rocket was extra)
- **Efforts to reduce system costs led to the failure**
  - Re-use of Inertial Reference System software from Ariane 4
  - Improperly handled exception caused by variable overflow during new flight profile (that wasn't simulated because of cost/schedule)
    - 64-bit float converted to 16-bit int assumed not to overflow
    - Exception caused dual hardware shutdown (because it was assumed software doesn't fail)

- **What really happened here?**
  - The narrow view: it was a software bug -- fix it
  - The broad view: the loss was caused by a lack of system robustness in an exceptional (unanticipated) situation

- **Many embedded systems must be *robust***

Software Drives Designs

- **Hardware is mostly a recurring cost**
  - Cost proportional to number of units manufactured
- **Software is a "one-time" non-recurring engineering design cost (NRE)**
  - Paid for "only once"
    - But bug fixes may be expensive, or impossible
  - Cost is related to complexity & number of functions
  - Market pressures lead to feature creep
  - ***SOFTWARE Is Not FREE!!!!!***

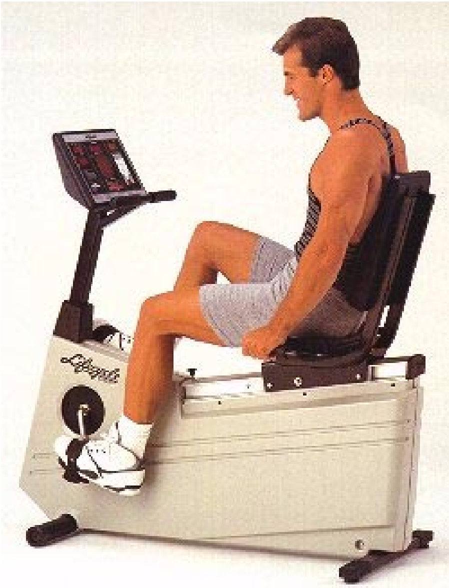


Source: *Software Requirements: objects, functions, states; Davis, 1993.*

Life-Cycle Concerns Figure Prominently

- **"Let's use a CAD system to re-synthesize designs for cost optimization"**
  - Automatically use whatever components are cheap that month
  - Would permit quick responses to bids for new variants
  - Track record of working fine for PC motherboards

- **Why wouldn't it work for an automotive application?**
  - Embedded system had more analog than digital -- mostly digital synthesis tool
  - Cost of re-certification for safety, FCC, warrantee repair rate
  - Design optimized for running power, not idle power
    - Car batteries must last a month in a parking lot
  - Parts cost didn't take into account life-cycle concerns
    - Price breaks for large quantities
    - Inventory, spares, end-of-life buy costs
  - Tool didn't put designs on a single sheet of paper
    - Archive system paper-based -- how else do you read 20-year-old files?



Generic Embedded System Designer Skill Set

- **Appreciation for multi-disciplinary nature of design**
  - System skills;   system = HW + SW + …
  - Understanding of engineering beyond digital logic
  - Ability to take a project from specification through production
- **Communication & teamwork skills**
  - Work with other disciplines, manufacturing, marketing
  - Work with customers to understand the real problem being solved
  - Make a good presentation; even better -- write “trade rag” articles
- **And, by the way, technical skills too…**
  - Low level: Microcontrollers, FPGA/ASIC, assembly language, A/D, D/A
  - High level: Object-oriented Design, C/C++, Real Time Operating Systems, Critical System design
  - Meta level: Creative solutions to highly constrained problems
  - Likely in the future: Unified Modeling Language, embedded networks
  - Uncertain future: Java, Windows CE

Review

- **What is an embedded system?**
  - More than just a computer -- it’s a system
- **What makes embedded systems different?**
  - Many sets of constraints on designs
  - Four general types:
    - General Purpose
    - Signal Processing
    - Communications
    - Control *(distributed control is focus of this course)*
- **What embedded system designers need to know**
  - **Multi-objective:** cost, dependability, performance, *etc.*
  - **Multi-discipline:** hardware, software, electromechanical, *etc.*
  - **Life cycle:** specification, design, prototyping, deployment, support, retirement
  - **Critical systems:** would you trust your own life to your product?

- **Next Lecture: Embedded/Real Time Fundamentals**