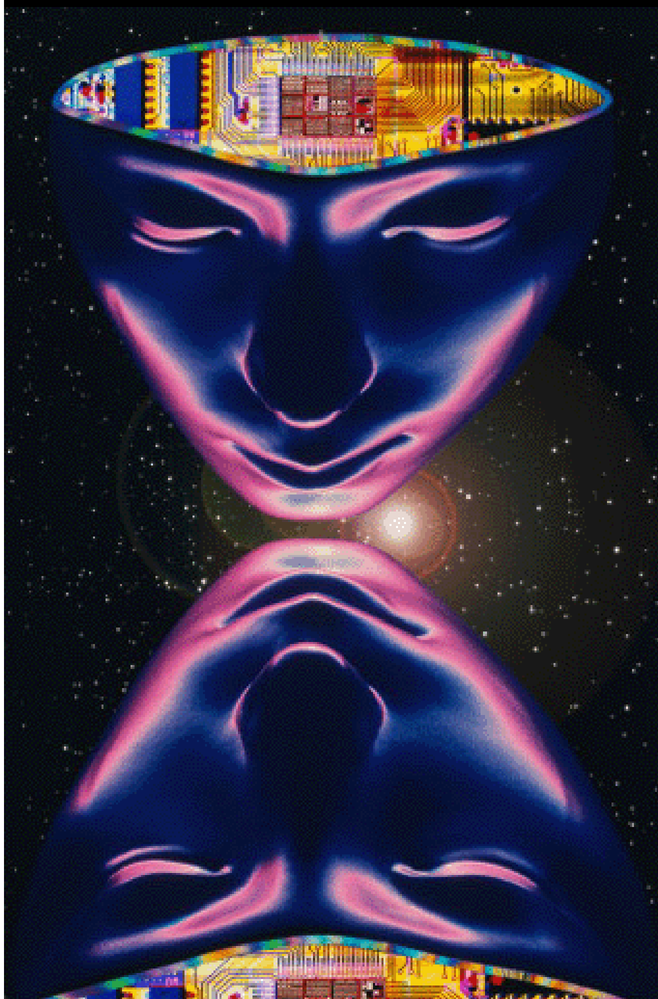


EE249
Embedded System Design: Models, Validation and Synthesis

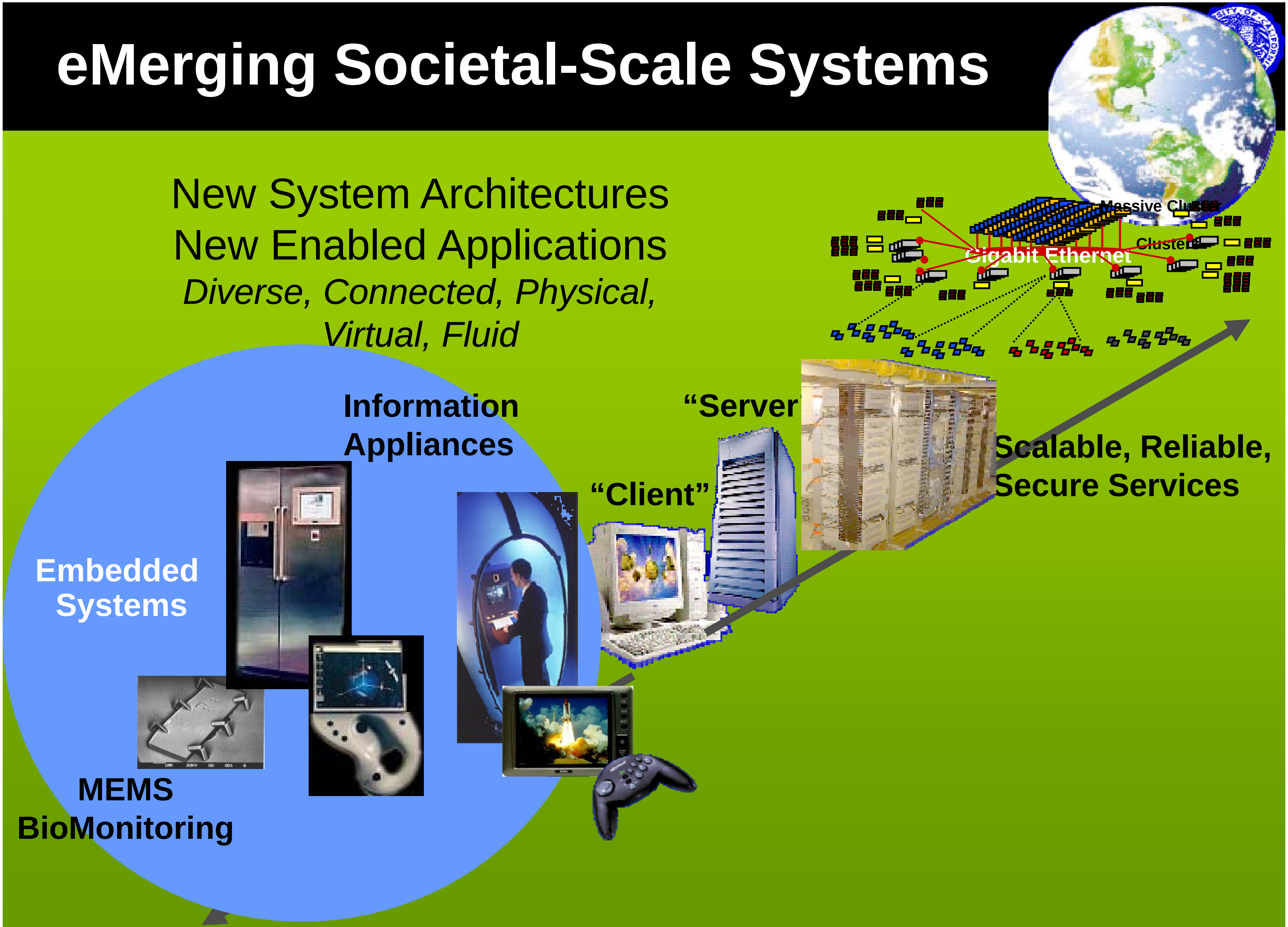
Alberto Sangiovanni-Vincentelli

"I believe we are now entering the Renaissance phase of the Information Age, where creativity and ideas are the new currency, and invention is a primary virtue, where technology truly has the power to transform lives, not just businesses, where technology can help us solve fundamental problems."

Carly Fiorina, CEO, Hewlett Packard Corporation

eMerging Societal-Scale Systems

New System Architectures
New Enabled Applications
Diverse, Connected, Physical, Virtual, Fluid



Embedded Software Systems

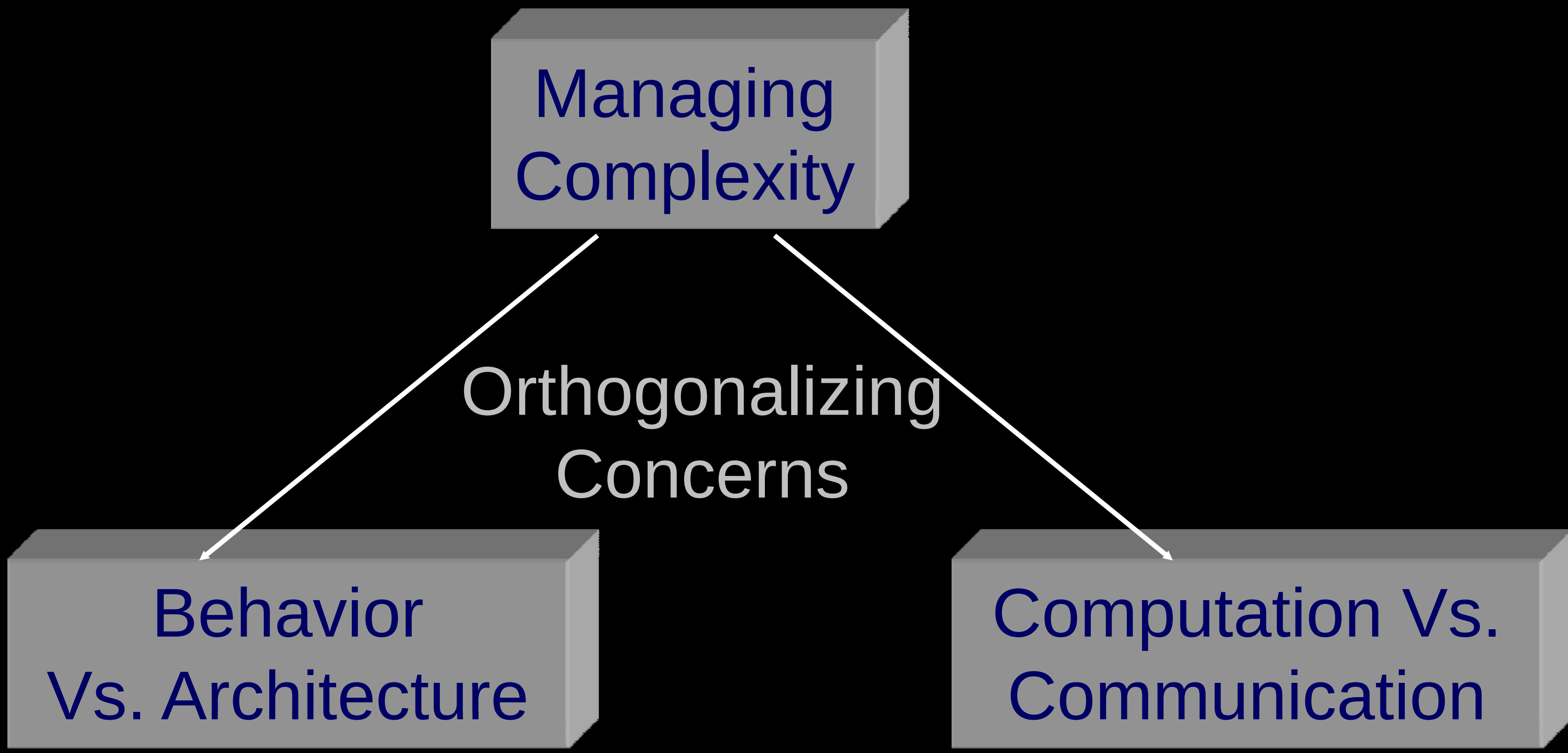
- Computational
  - but not first-and-foremost a computer
- Integral with physical processes
  - sensors, actuators
- Reactive
  - at the speed of the environment
- Heterogeneous
  - hardware/software, mixed architectures
- Networked
  - shared, adaptive

Source: Edward A. Lee

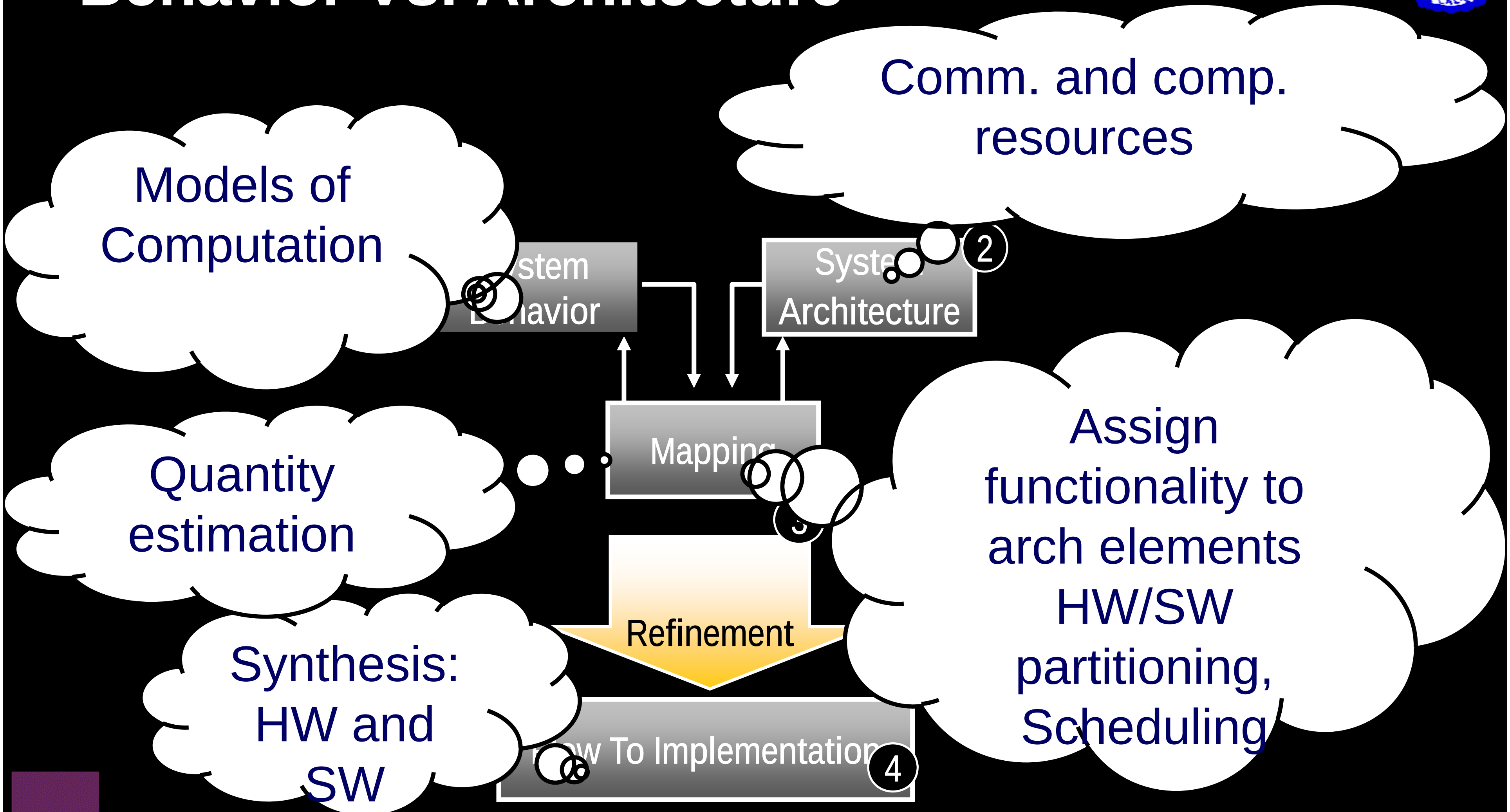
Observations

- We are on the middle of a revolution in the way electronics products are designed
- System design is the key (also for IC design!)
  – Start with the highest possible level of abstraction (e.g. control algorithms)
  – Establish properties at the right level
  – Use formal models
  – Leverage multiple "scientific" disciplines

Course overview



Behavior Vs. Architecture



- Polis (1990-1996)
- VCC (1996-2003)

Behavior Vs. Communication

- Clear separation between functionality and interaction model
- Maximize reuse in different environments, change only interaction model

Administration

- Office hours: *Alberto : Tu-Th 12:30pm-2pm or (better) by appointment (2-4882)*
- *Co-Instructor: Alessandro Pinto, apinto@eecs.berkeley.edu*
- Teaching Assistant:
  - **Qi Zhu,** zhuqi@eecs.berkeley.edu

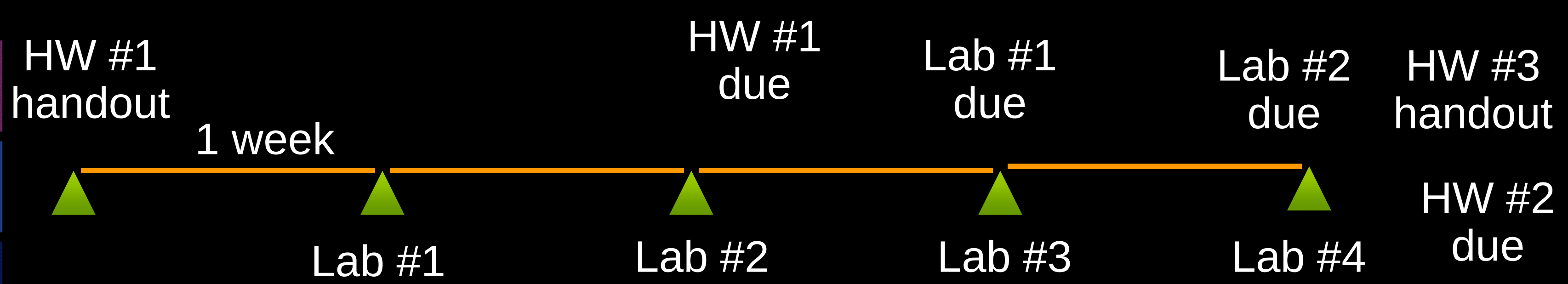
Grading

- Grading will be assigned on:
  - Homework (~30%)
  - Project (~50%)
  - Reading assignments (~10%)
  - Labs (10%)
- Bi-weekly homework.
  - HW #n is due the same day HW #n+1 is handed out

Schedule

Schedule is tight
Don’t fall behind!!!

- Labs (Th. 4-6):
  - Presentation of tools followed by hands-on tutorial and assignments (to turn in after 2 weeks, we might have to skip some labs….)
- Discussion Session (Tu. 5-6)
  - Each student (in groups of 2 people) will have to make one or more oral presentations during the class
- Last two weeks of class dedicated only to projects (usually due the 1st or 2nd week of Dec.)
- Auditors are OK but please register as P-NP (resources are assigned according to students…)



Links

- Class website
  - http://inst.eecs.berkeley.edu/~ee249
- But also
  - http://www.eecs.berkeley.edu/~apinto/esd/

Outline of the course

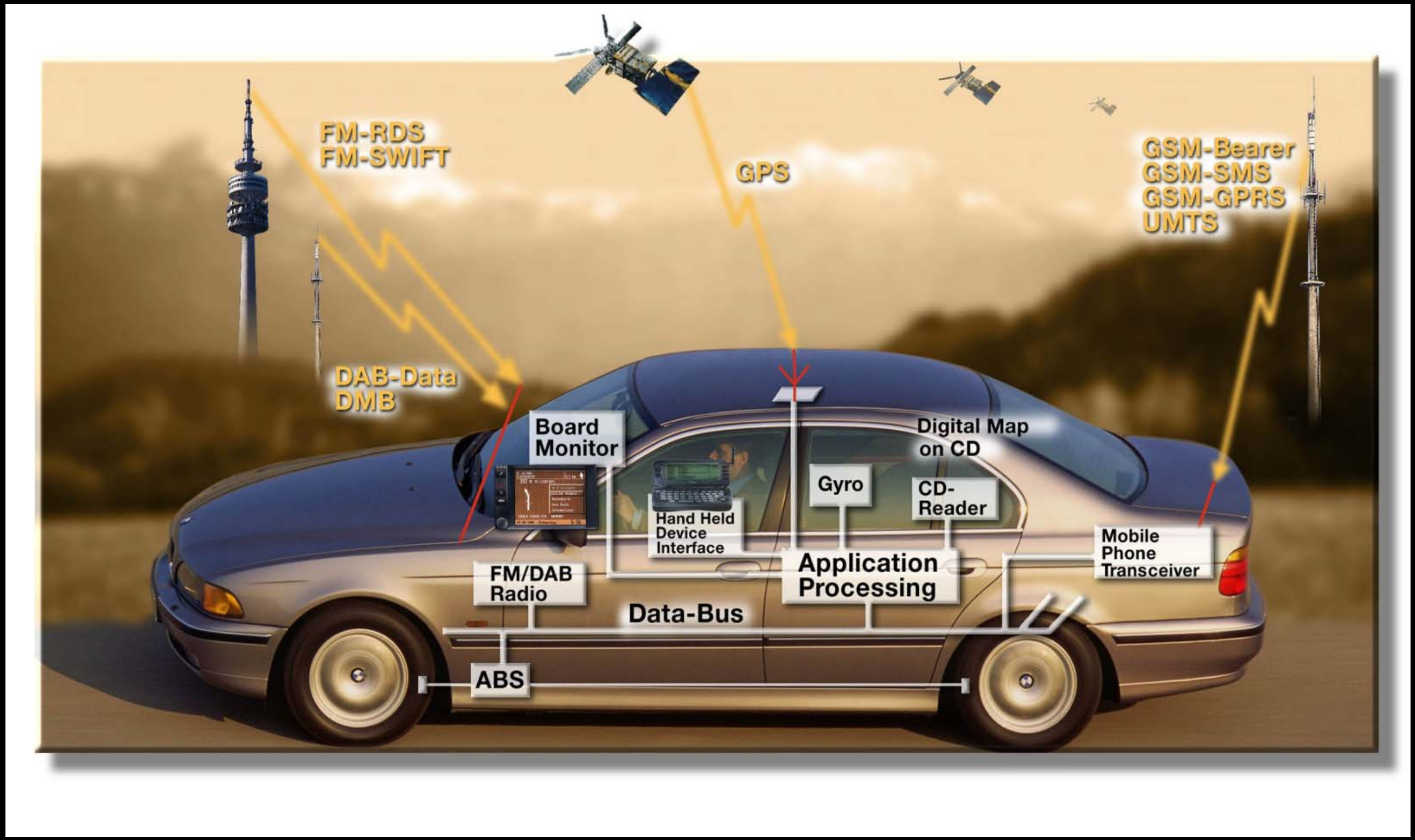
Part 1: Introduction	Design complexity, Example of embedded systems, traditional design flow, Platform-Based Design
Part 2: Functional modeling, analysis and simulation	Introduction to models of computation. Finite State Machines and Co-Design Finite State Machines, Kahn Process Networks, Data Flow, Petri Nets, Hybrid Systems. Unified frameworks: the Tagged Signal Model, Agent Algebra
Part 3: Architecture and performance abstraction	Definition of architecture, examples. Distributed architecture, coordination, communication. Real time operating systems, scheduling of computation and communication.
Part 4: Mapping	Definition of mapping and synthesis. Software synthesis, quasi static scheduling. Behavioral synthesis. Communication Synthesis and communication-based design
Part 5: Verification	Validation vs Simulation. Verification of hybrid system. Interface automata and assume guarantee reasoning.
Part 6: Applications	Automotive: CAN,Flexray, Auotosar Architecture, GM car architecture, scheduling and timing analysis Building automation: BanNet, LonWorks, ZigBee with applications to monitoring and security

Outline for the Introduction

- Examples of Embedded Systems
- Their Impact on Society
- Design Challenges
- Embedded Software and Control

Electronics and the Car

- More than 30% of the cost of a car is now in Electronics
- 90% of all innovations will be based on electronic systems



Automotive Industry
Three Levels of Players

Automakers

GM, Ford, TOYOTA, VW, HONDA, DAIMLERCHRYSLER

- 2005 Revenue: $1.1T
- CAGR 2.8% (2004-2010)

Tier 1 Suppliers

Visteon, DENSO, JOHNSON CONTROLS, BOSCH, DELPHI

90%+ of revenue from automotive

- 2004 Revenue ~$200B
- CAGR 5.4% (2004-2010)

IC Vendors

freescale semiconductor, RENESAS, ST, Infineon, NEC

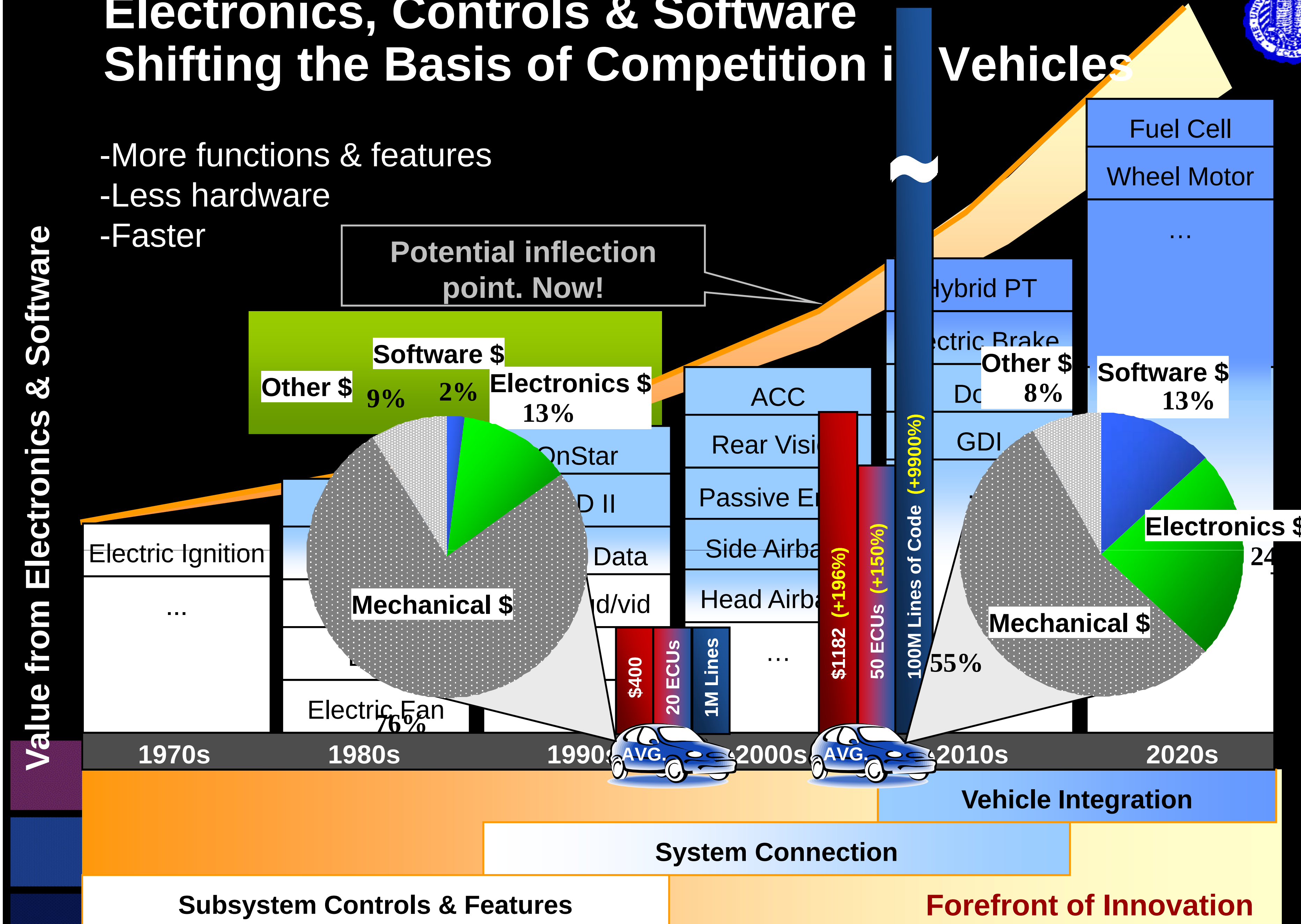
~15% of revenue from automotive

- 2005 revenue $17.4B
- CAGR 10% (2004-2010)

Source: Public financials, Gartner 2005

Electronics, Controls & Software Shifting the Basis of Competition in Vehicles

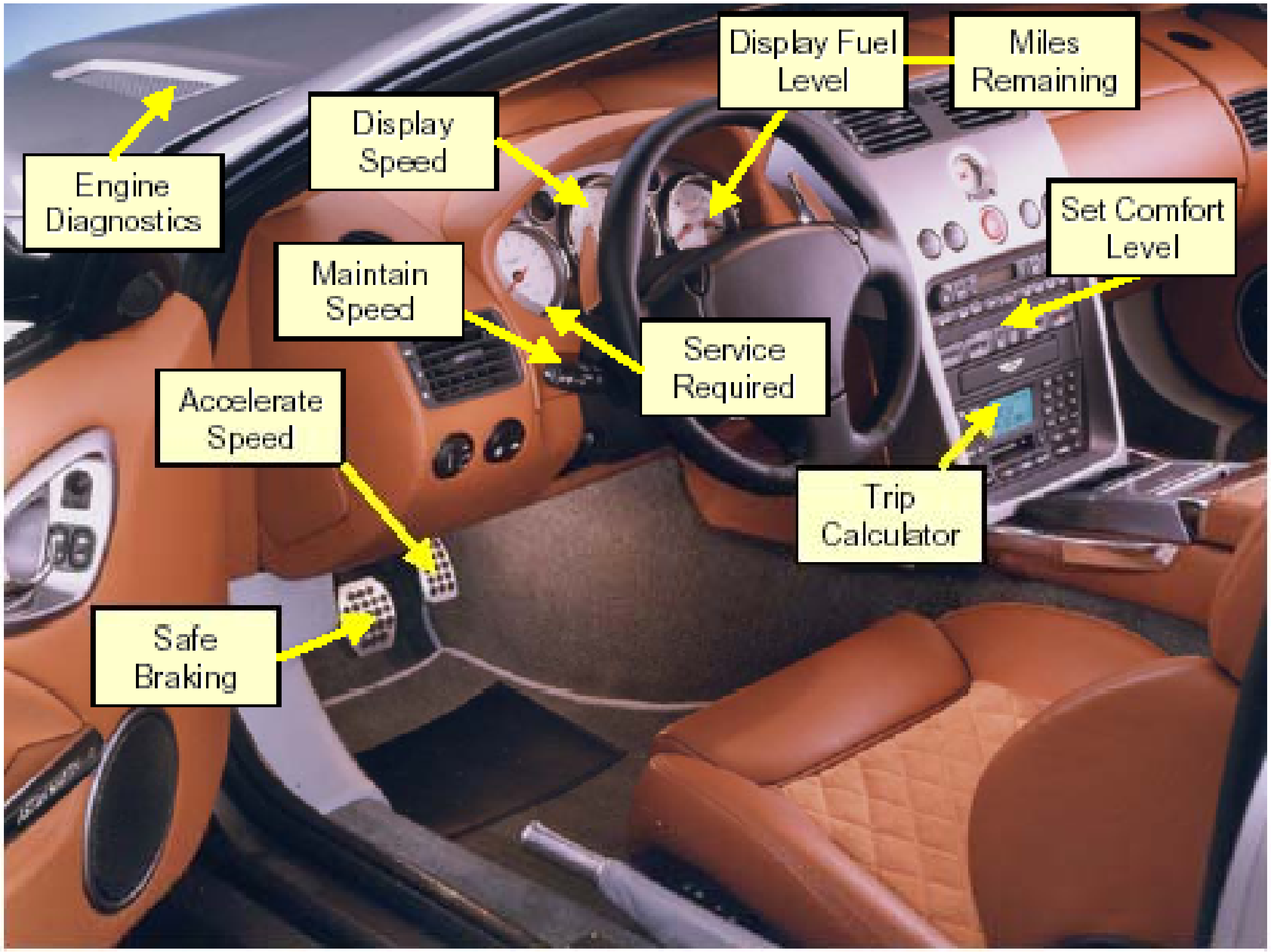
- -More functions & features
- -Less hardware
- -Faster



Source: Matt Tsien, GM

FUNCTION OF CONTROLS

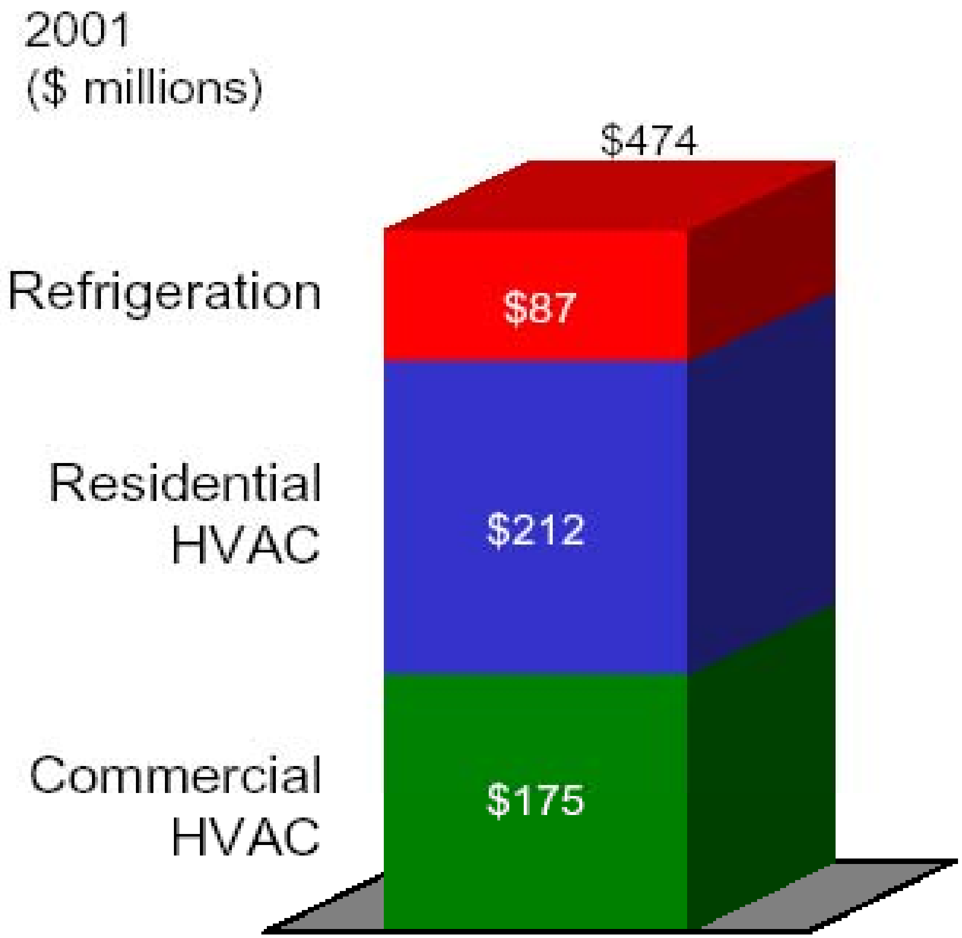
Typical minivan application



- Configure
- Sense
- Actuate
- Regulate
- Display
- Trend
- Diagnose
- Predict
- Archive

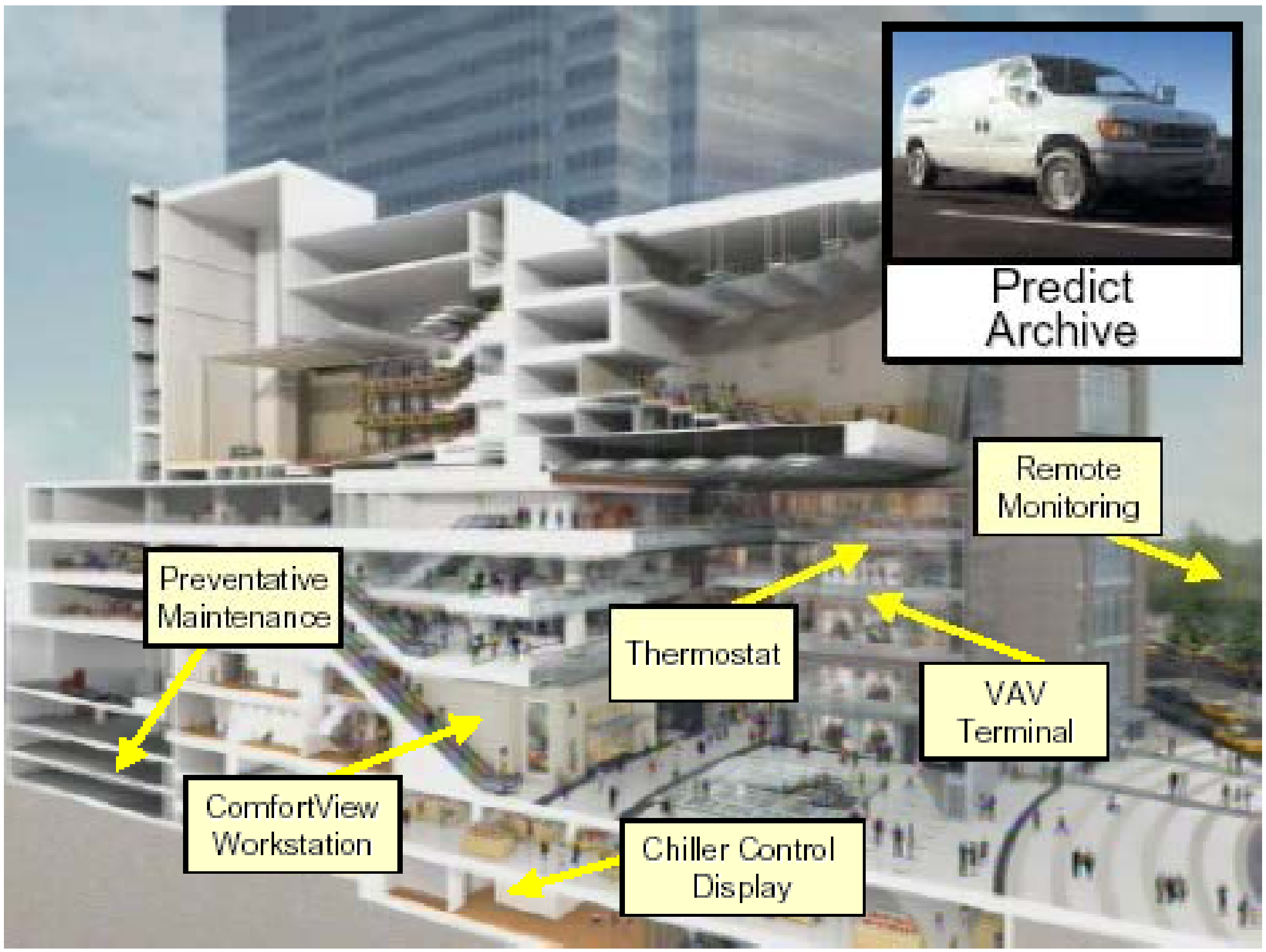
CARRIER CONTROLS BUSINESS

Market segments



FUNCTION OF CONTROLS

Typical commercial HVAC application



Configure

Sense

Actuate

Regulate

Display

Trend

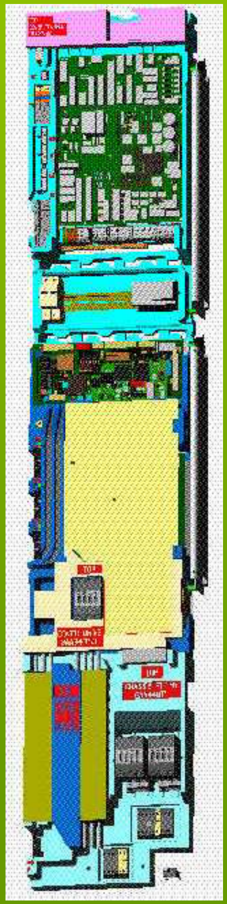
Diagnose

Predict

Archive

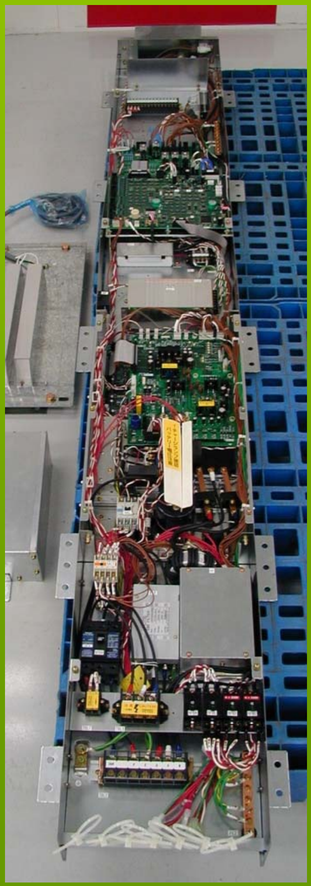
OTIS Elevators

1. EN: GeN2-Cx



2. ANSI:
Gen2/GEM

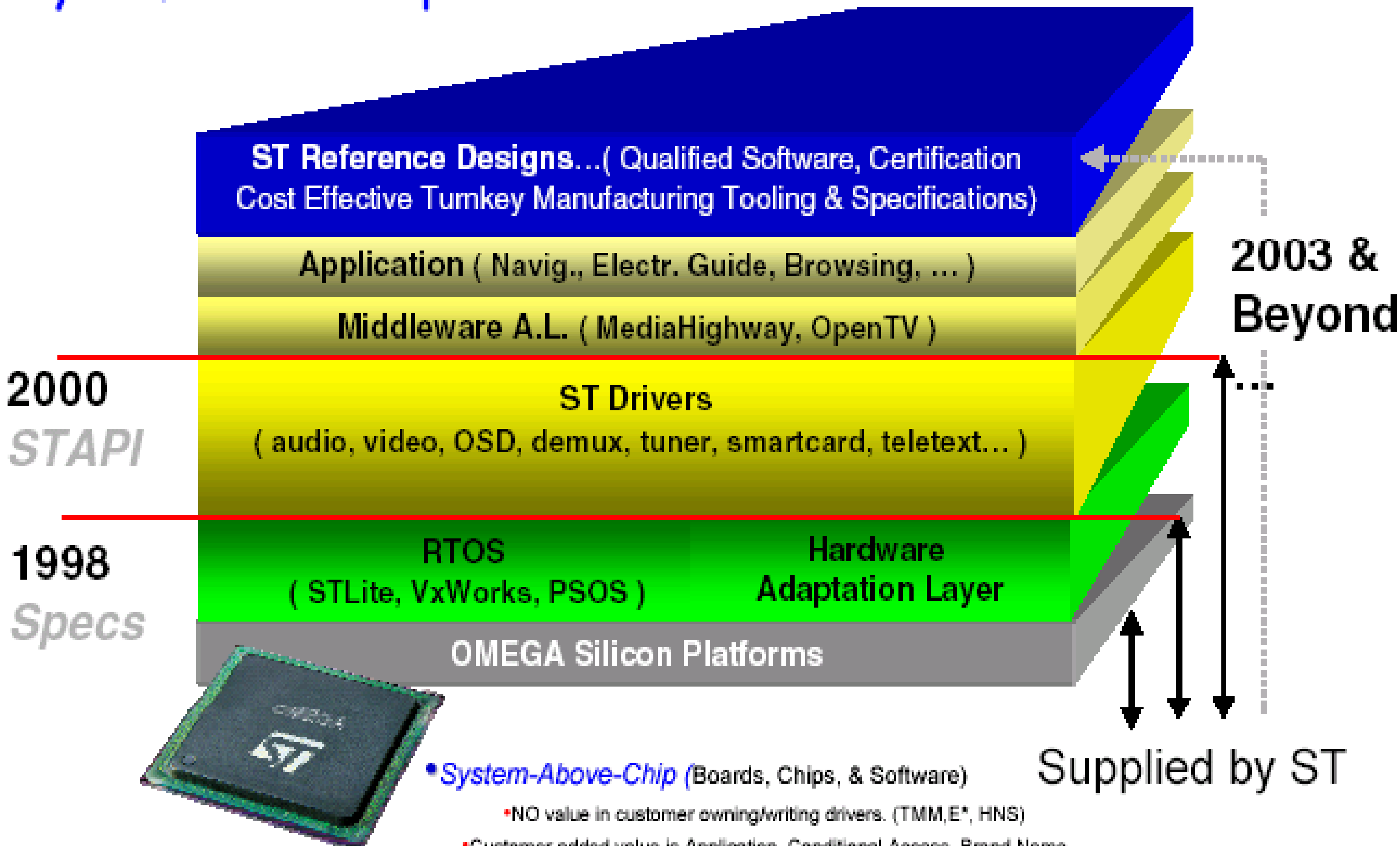
3. JIS:
GeN2-JIS



Segments

Attribute	Type 1	Type 2	Type 3
Stops/Rise	< 20 stops Opportunity: < 6 stops (20m)	< 64 stops	< 128 stops
Group Size	Simplex	1 – 8 cars	1 – 8 cars
Speed	< 4m/s <= .75 m/s (ANSI)	< 4 m/s	< 15 m/s
Op Features	Basic	Advanced	Hi-End Dispatch
Motion Features	Basic Perf. Basic FM	Limited Perf. Advanced FM	Advanced Perf. Advanced FM
Code	EN, ANSI, JIS	EN, ANSI, JIS	EN, ANSI, JIS
Remote Service	Yes	Yes	Yes
Price Sensitivity	High	High, Med	Med
Market	Utility	Utility, Design	Design

System Above Chip - SAC



- *System-Above-Chip* (Boards, Chips, & Software)
  - NO value in customer owning/writing drivers. (TMM,E*, HNS)
  - Customer added value is Application, Conditional Access, Brand Name
- ST supplies the complete base system BELOW MIDDLEWARE to save time to market

Consumer segments
Common technology elements

aming

Bro

‘Systems within systems’

- Multimedia processors
- Embedded µP
- Wireless connectivity
  Baseband processing, RF transceivers
- Power Amps
- Flat panel displays
- Digital signal processor technologies
- VOIP

Locality

Internet

e-commerce

Auto electronics

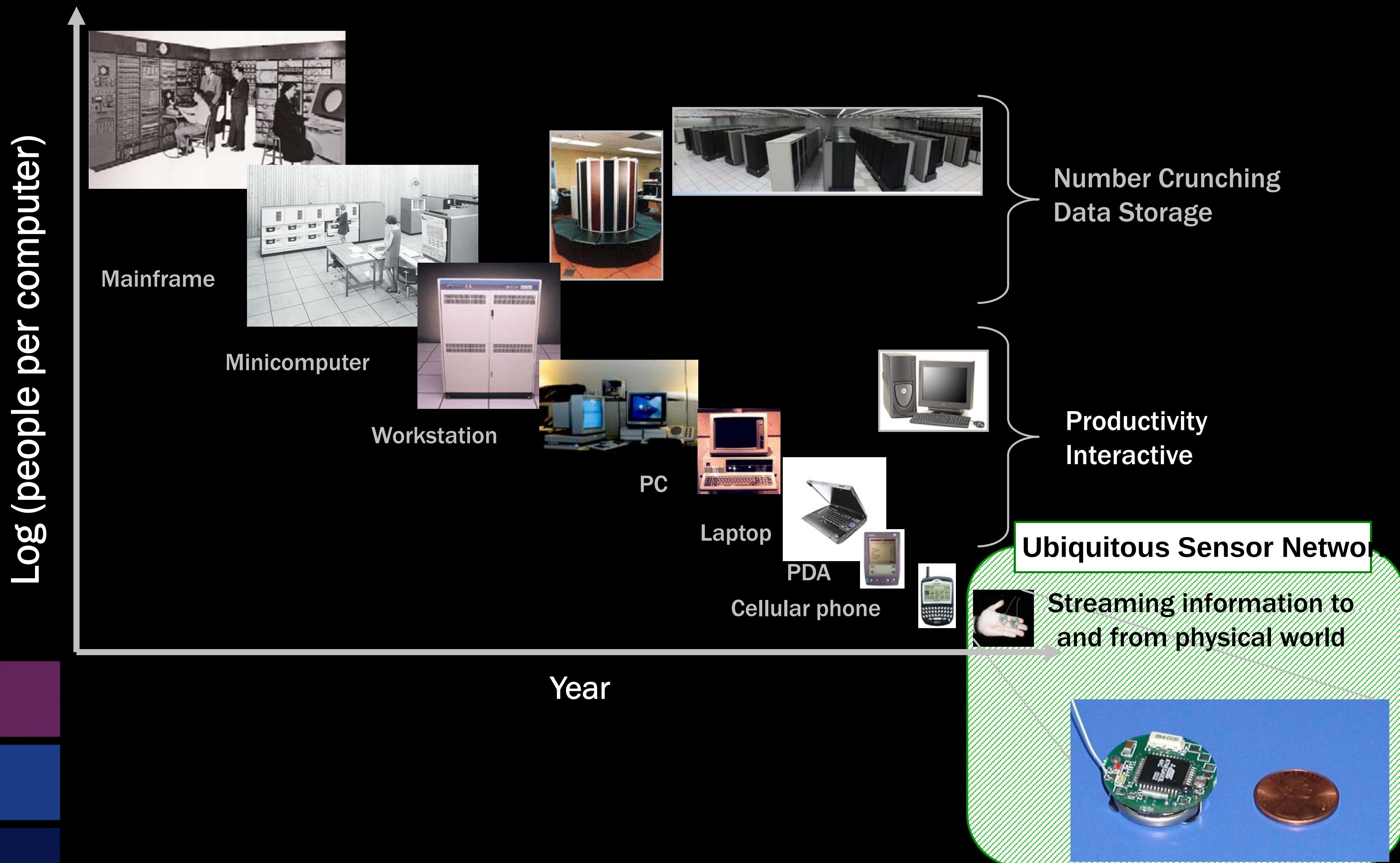
Common Situation in Industry

- Different hardware devices and architectures
- Increased complexity
- Non-standard tools and design processes
- Redundant development efforts
- Increased R&D and sustaining costs
- Lack of standardization results in greater quality risks
- Customer confusion

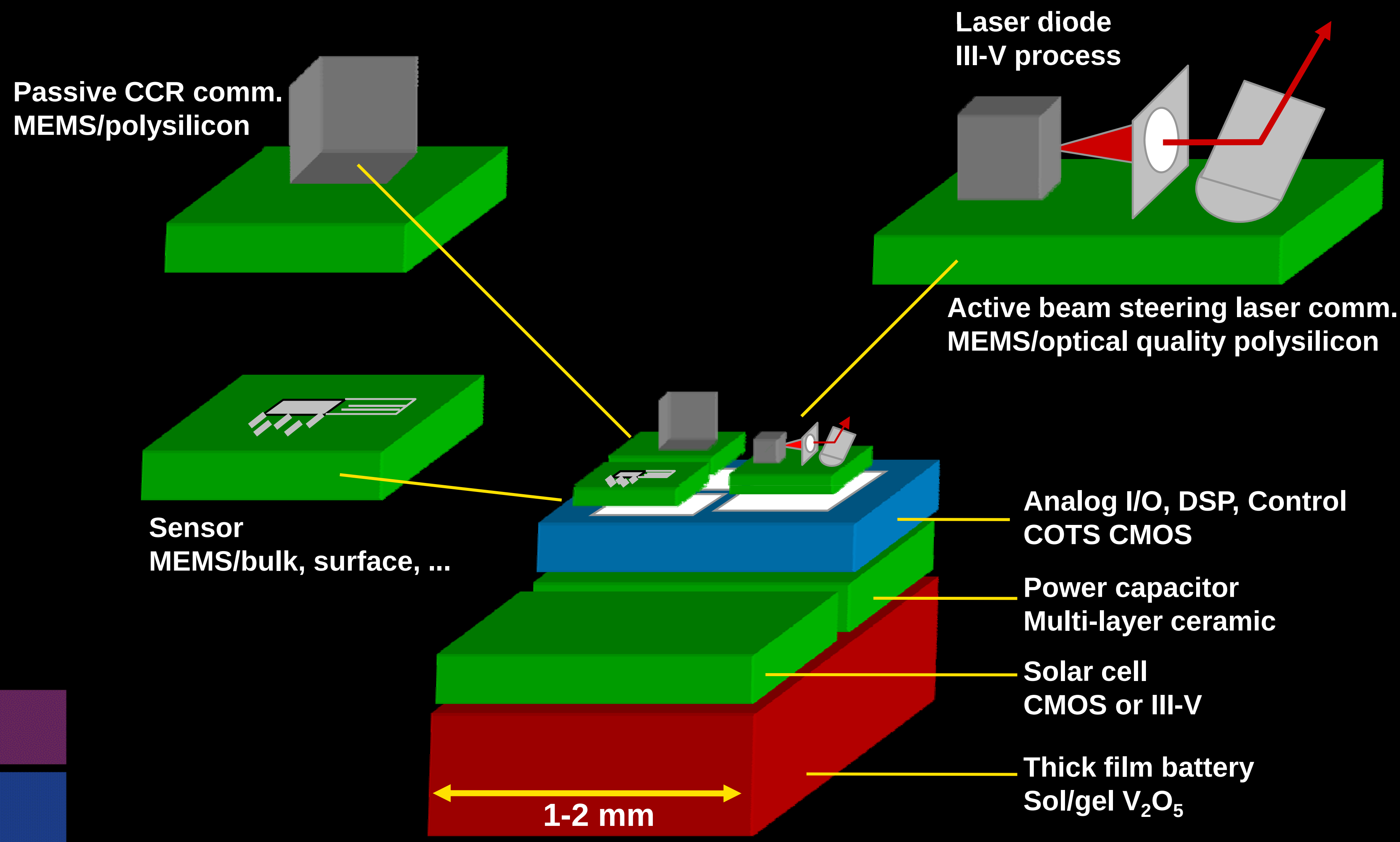
Outline for the Introduction

- Examples of Embedded Systems
- The Future of Embedded Systems and Their Impact on Society
- Design Challenges
- Embedded Software and Control

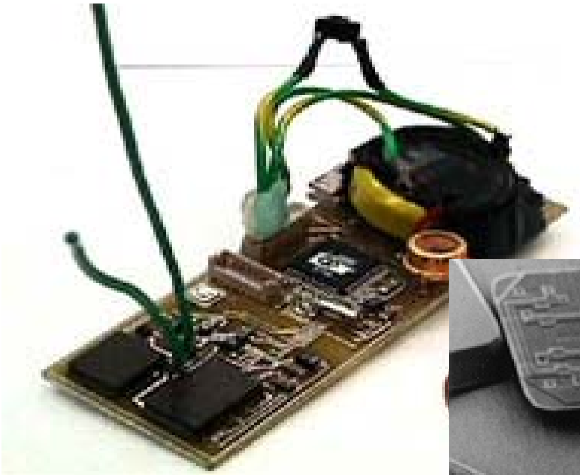
The Physical Internet



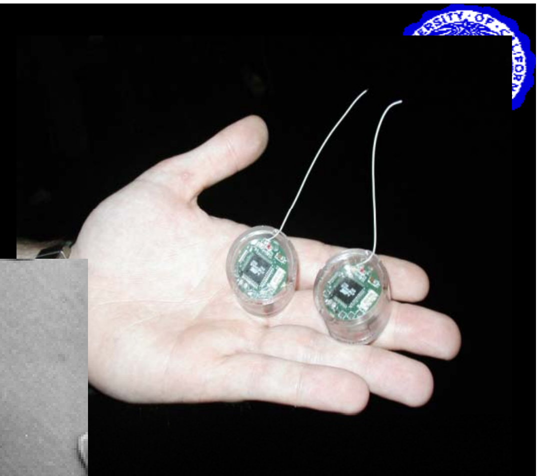
Smart Dust



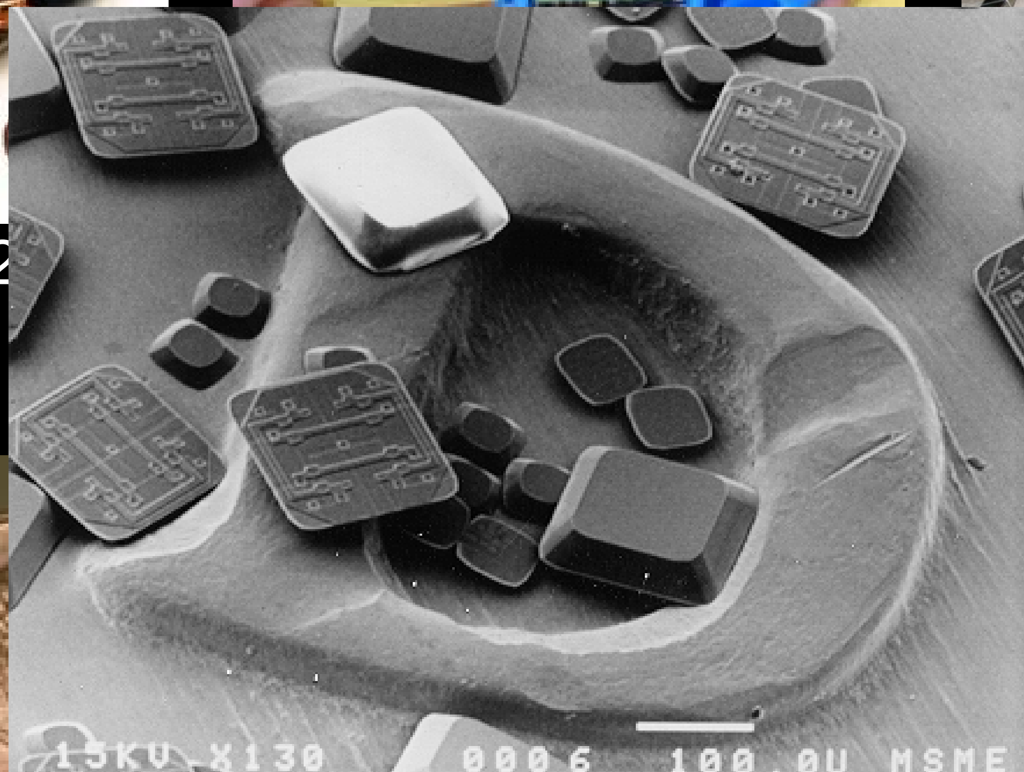
Source: K. Pister, Berkeley



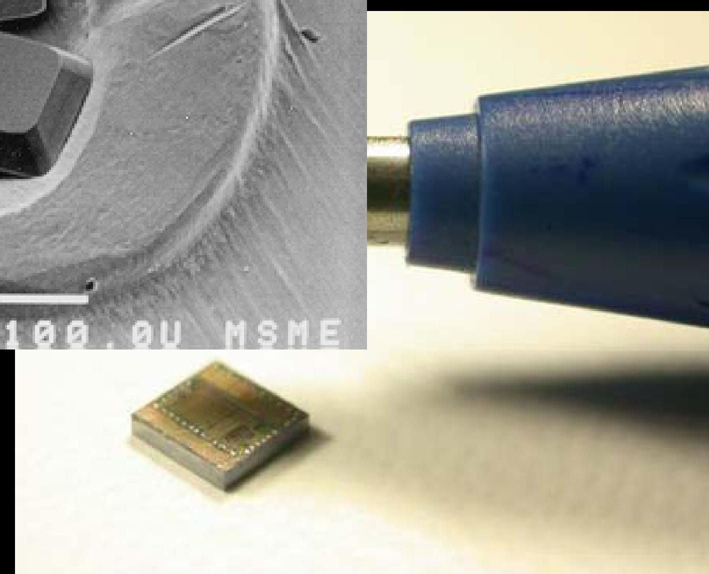
February 2



August 2001

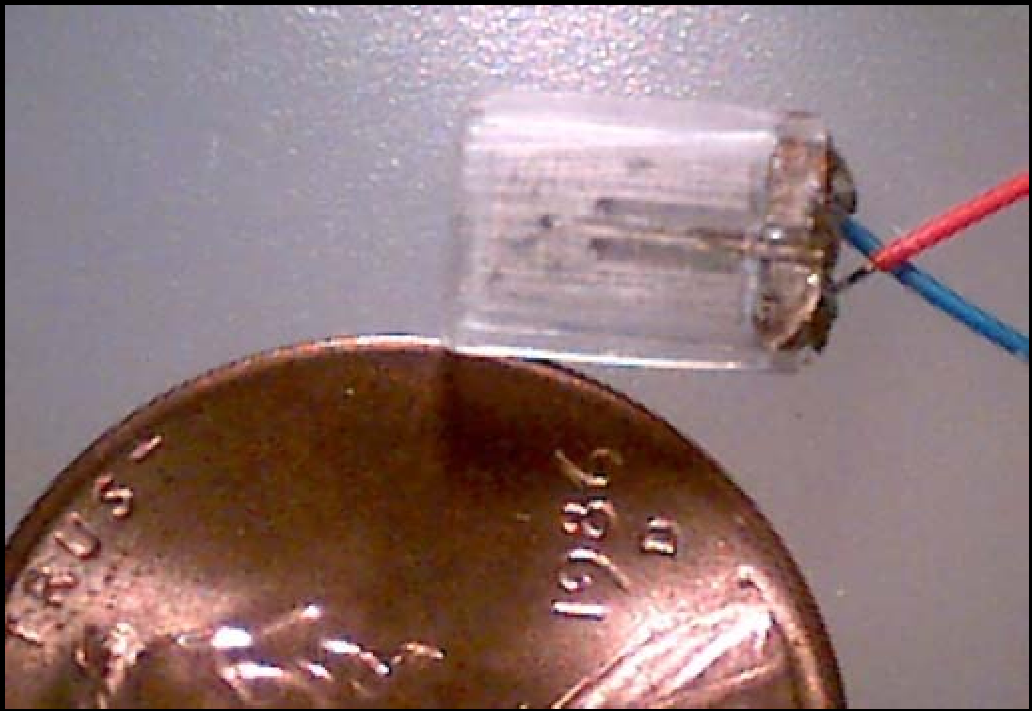


February 2002



February 2003

Energy Scavenging: Vibration

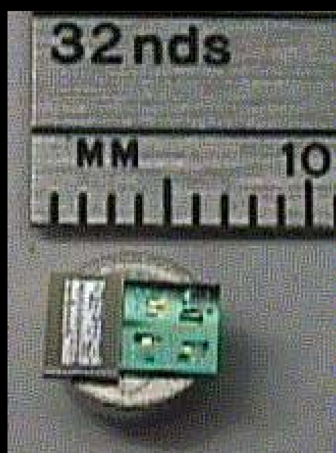


Source: P. Wright, Berkeley

Wireless Sensor Networks

The use of wireless networks of embedded computers "could well dwarf previous milestones in the information revolution" - National Research Council Report: Embedded, Everywhere", 2001.

Berkeley Dust Mote[1]



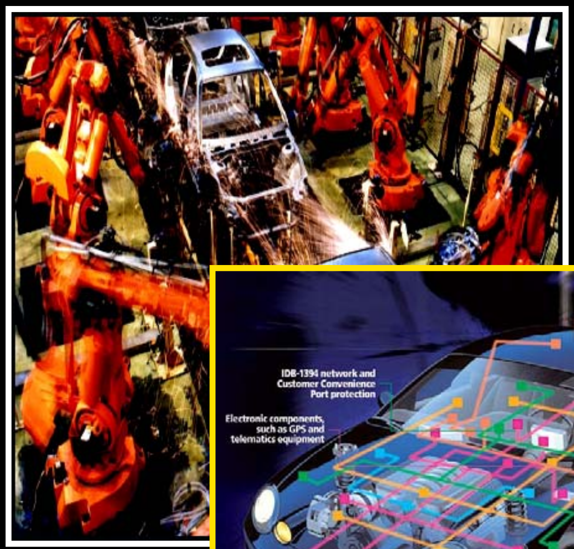
Berkeley Mote[1]

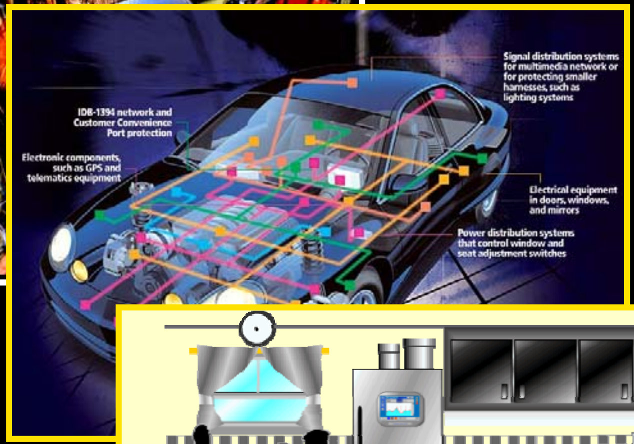
[1]**From Pister *et al.*, *Berkeley Smart Dust Project***

Creating a Whole New World of Applications

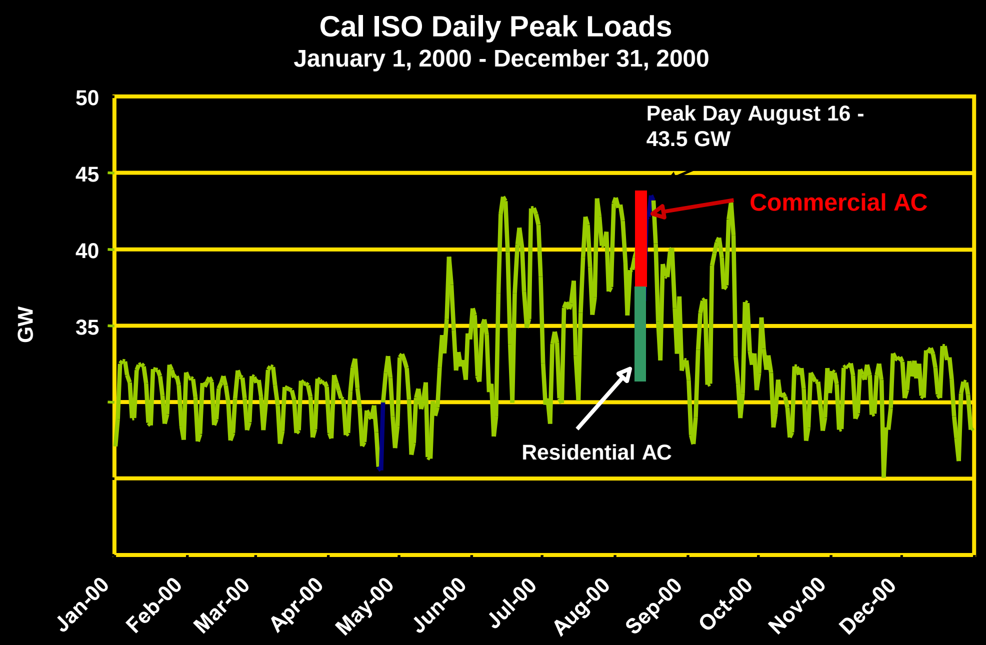
From Monitoring

To Automation



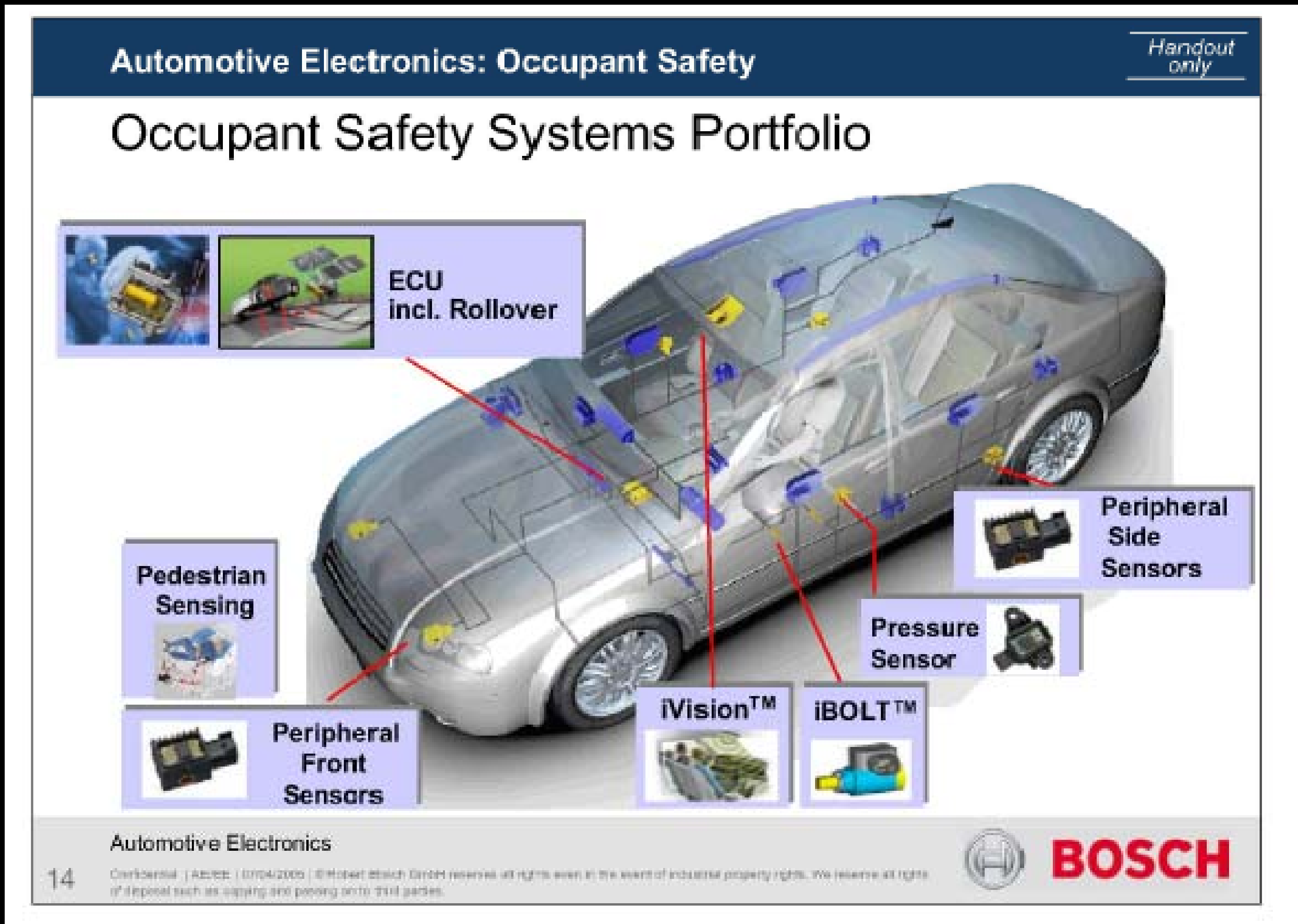


Energy Management and Conservation

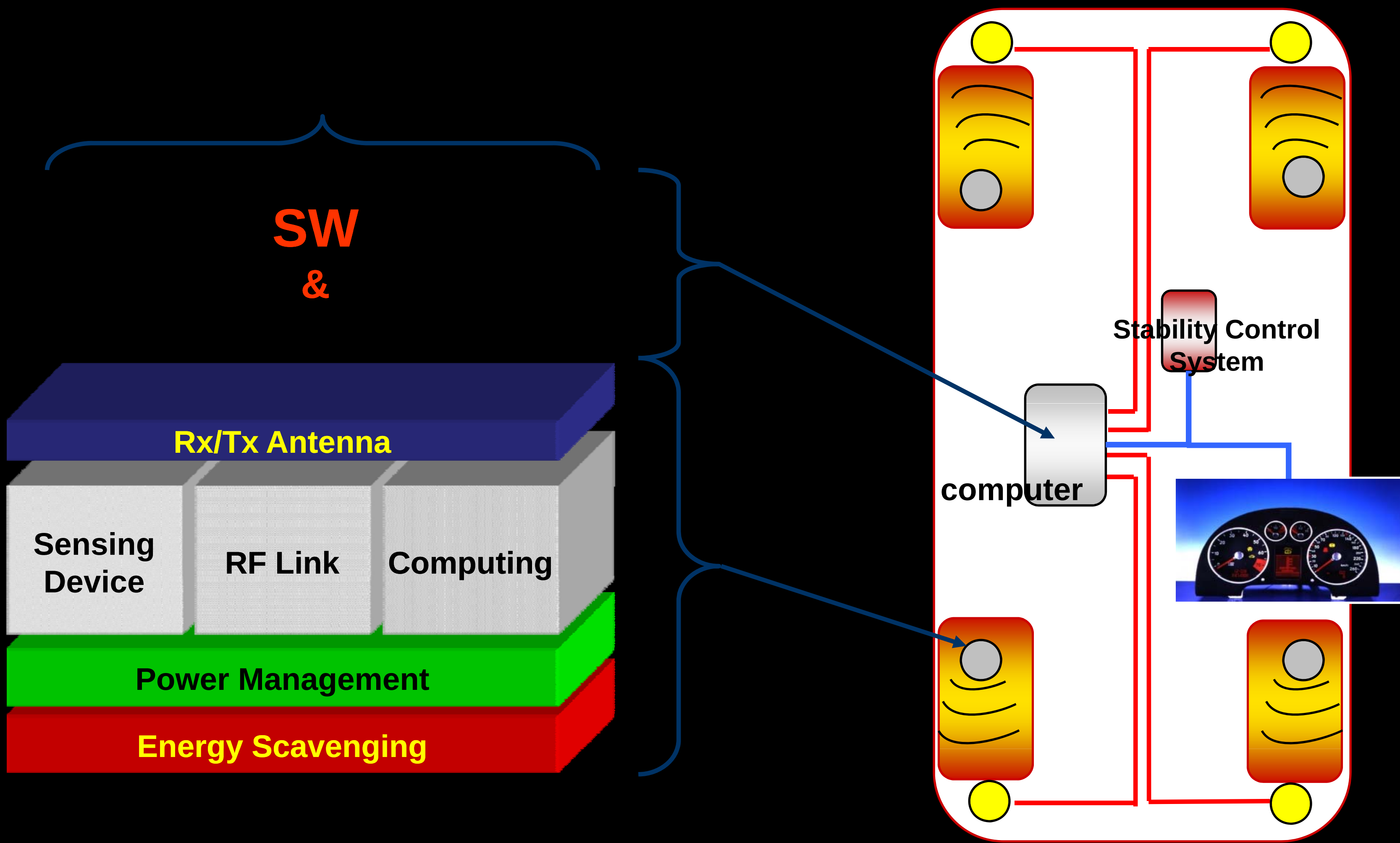


Demand response:
Make energy prices dependent upon time-of-use

- Advanced thermostats operate on required level of comfort, energy cost, weather forecast and distributed measurements to offload peak times
- Appliances are energy and cost aware


Automotive Electronics: Occupant Safety
Handout only
Occupant Safety Systems Portfolio
ECU incl. Rollover
Pedestrian Sensing
Peripheral Front Sensors
iVision™
iBOLT™
Pressure Sensor
Peripheral Side Sensors
Automotive Electronics
14
BOSCH

Tire to Vehicle



Industrial Plants

Monitoring:
Vibrations, Temperature,
Humidity, Position, Logistics

Current solution:
Wired Infrastructure

Future solution:
WIRELESS

Wireless advantages:
Reduce cabling
Enhance flexibility
Easy to deploy
Higher safety
Decreased maintenance costs

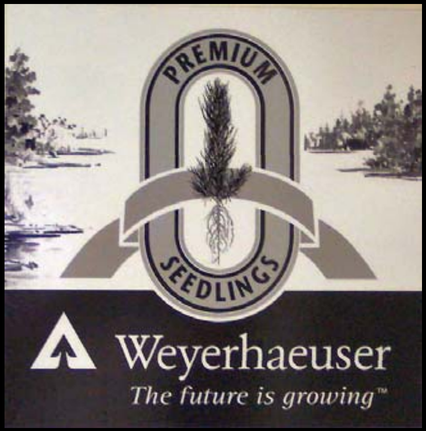
Temperature Tracking

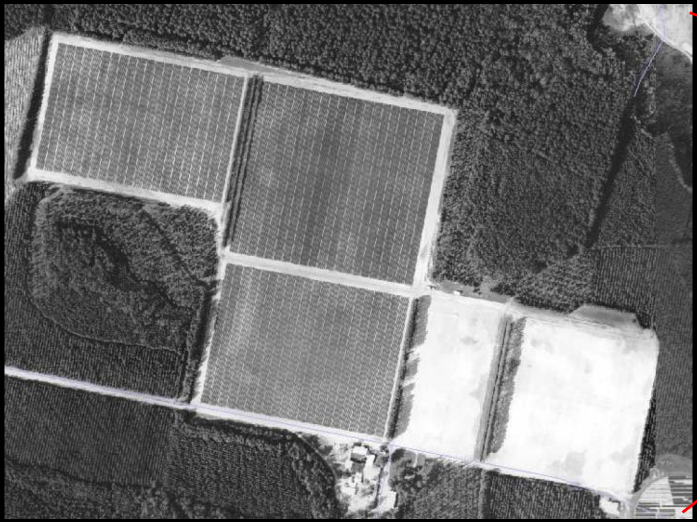
- No or little real-time data on assets, environment, or activity
  - Inventory/supply management
    - Pharmaceutical
    - Foods
  - Automated meter reading

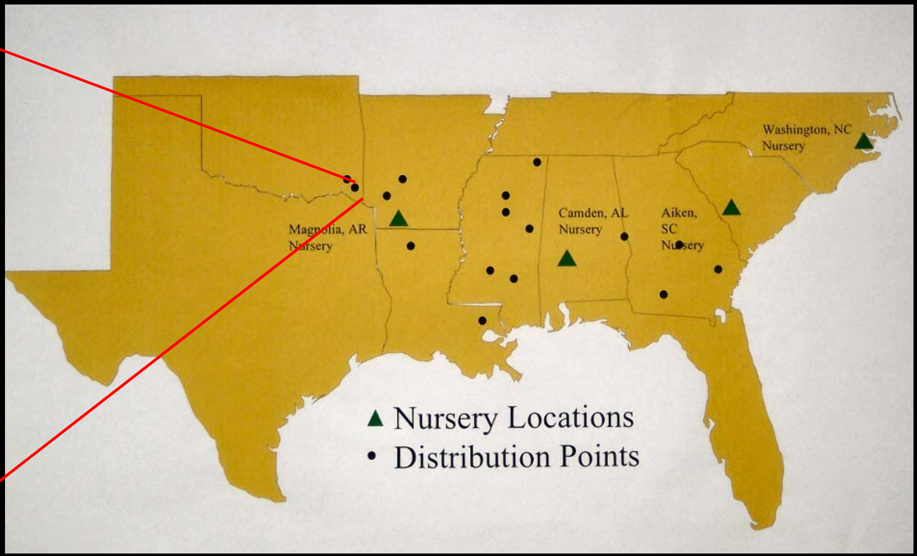
Source: Xbow

Weyerhaeuser 20 Million Seed Management

- **Task:**
  - Manage 20 million fast growing seeds annually
- **Issue:**
  - Seed dormancy depends on a complex combination of water, light, temperature, gasses, mechanical restrictions, seed coats, and hormone structures







Source: Xbow

Tree Growth Rate Variability

- **Old Method**
- Trust nature
- Monitor local atmospheric conditions

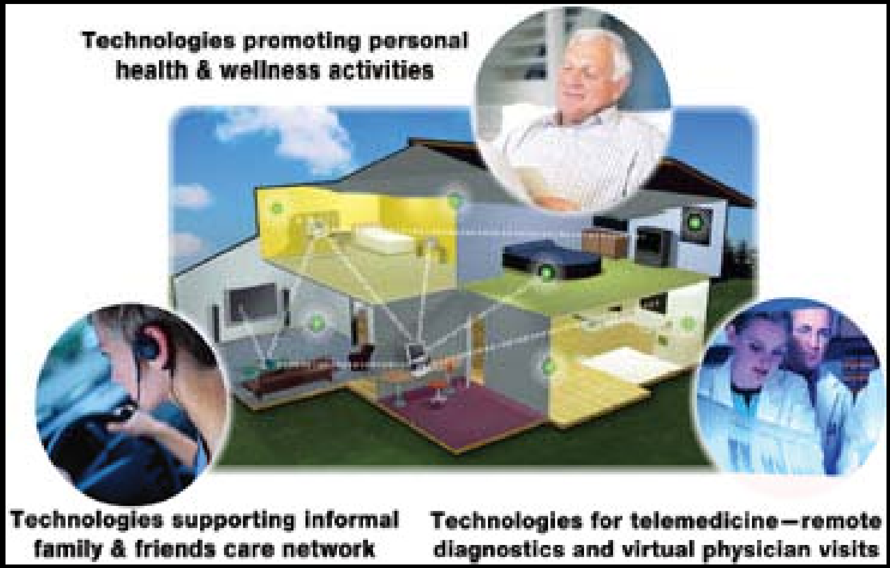
- **Sensor Network Way:**
- Monitor soil temperature and moisture at various locations
- Adjust irrigation schedule accordingly

Preventative Maintenance Program on Oil Tankers

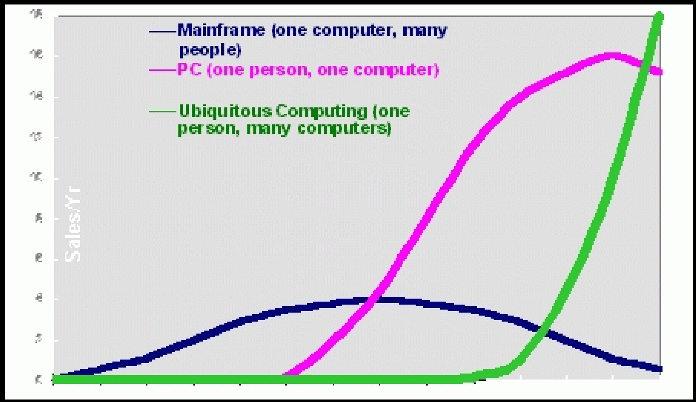
- The task:
  - Engine monitoring is critical for both keeping the ship operational and complying with insurance policy.
- Old Methods
  - Manually record vibration profile with data loggers.
  - Post process data for engine health and diagnostics.

Source: Xbow

Personal Themes



- Data driven, remote feedback control
- Government or industry mandates
- Personal computing themes
  - Ubiquitous computing
  - Safety
  - Convenience
  - Health and performance
  - Entertainment



Source: Xbow

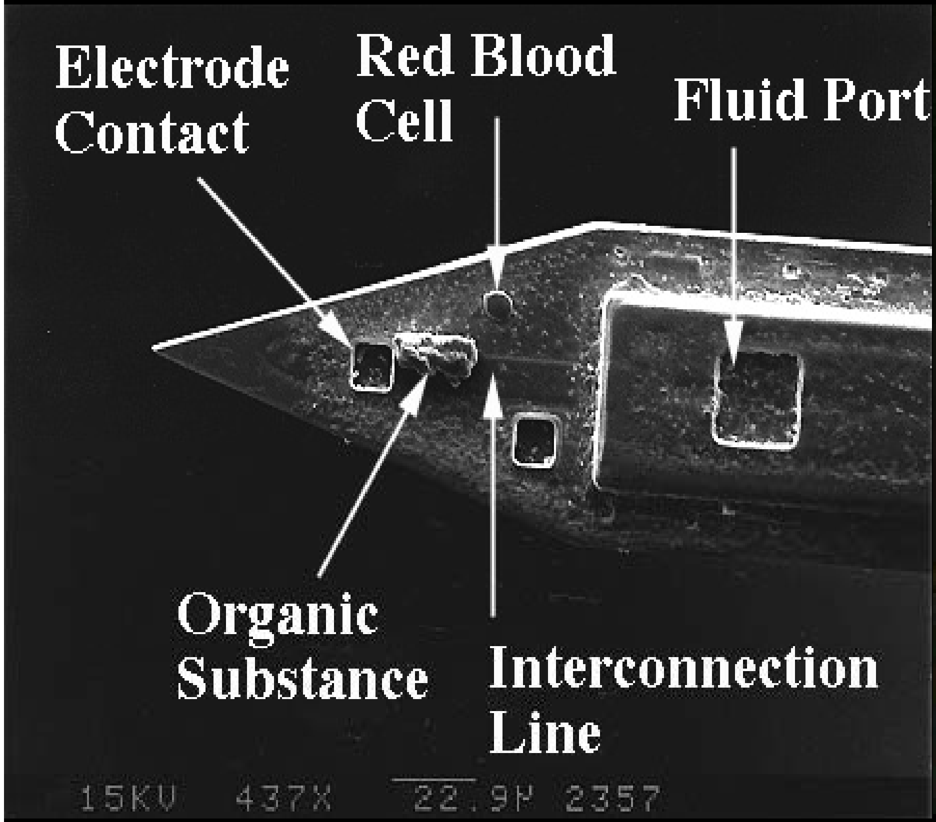
Applications

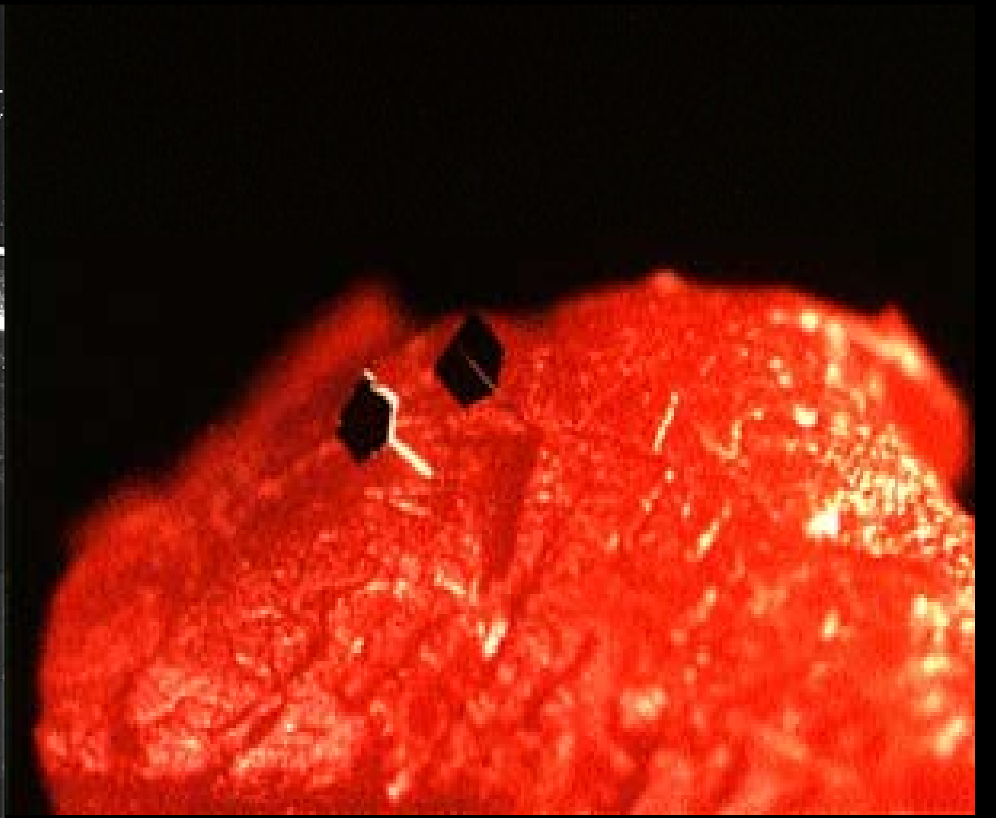
Distributed Bio-monitoring

- Wristband bio-monitors for chronic illness and the elderly
- Monitored remotely 24x7x365
- Emergency response and potential remote drug delivery
- Cardiac Arrest
  - Raise out-of-hospital survival rate from 6% to 20% => save 60K lives/year

Silicon-Processed Micro-needles

bio-medical **porterhouse (New-York)**





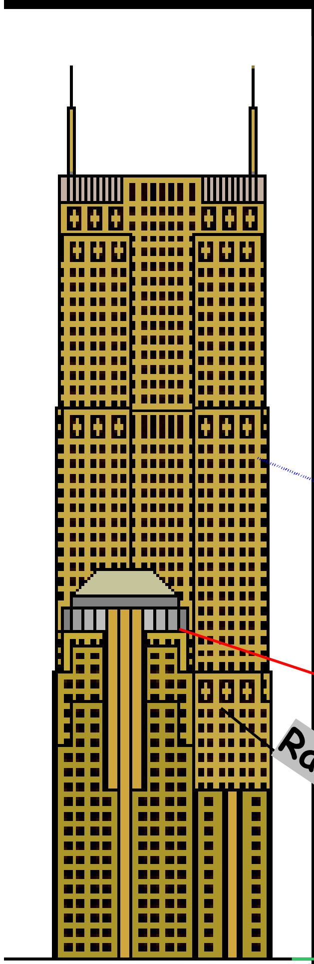
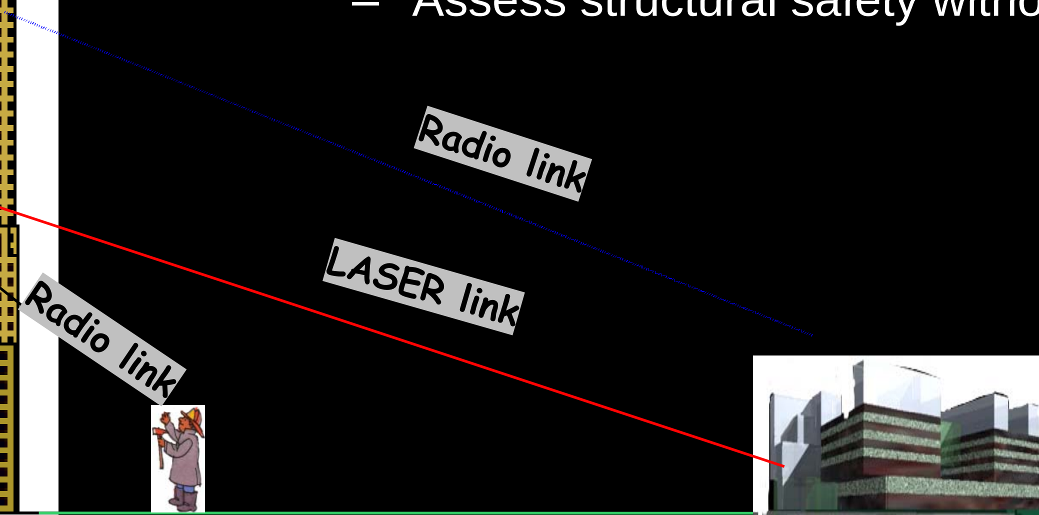
Applications

Disaster Mitigation (natural and otherwise)

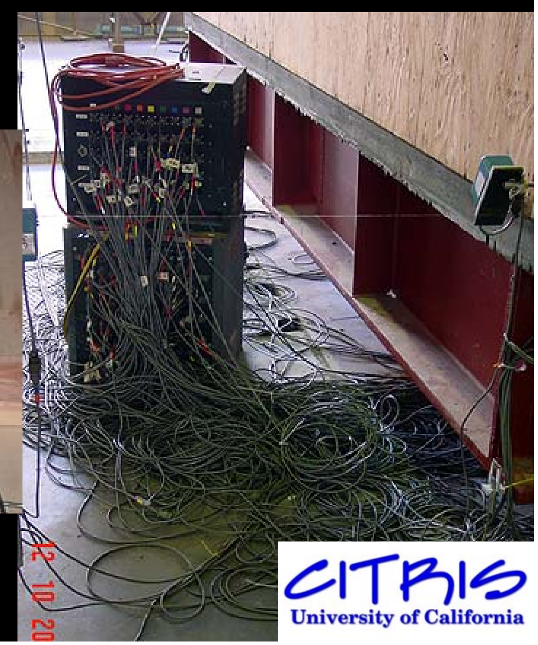
- Monitor buildings, bridges, lifeline systems to assess damage after disaster
- Provide efficient, personalized responses
- Must function at maximum performance under very difficult circumstances

What is Disaster Response?

- **Sensors installed near critical structural points**
- **Sensor measure motion, distinguish normal deterioration and serious damage**
- **Sensors report location, kinematics of damage during and after an extreme event**
  - Guide emergency personnel
  - Assess structural safety without deconstructing building

Seismic Monitoring of Buildings: Before CITRIS



Seismic Monitoring of Buildings: With CITRIS Wireless Motes

CITRIS
University of California

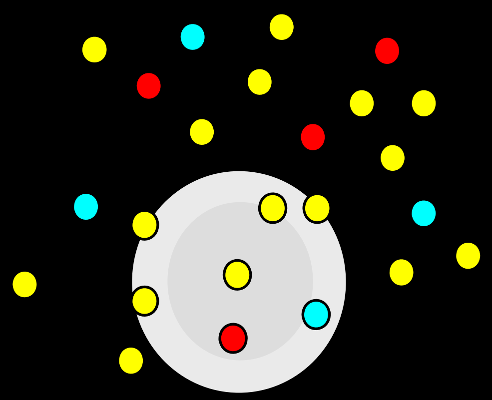
Stability of Masada North Face:

The Foundations of King Herod's Palace

Discussion

- What are the most challenging aspects of these applications (and how does a company make money) ?
  - Interaction mechanisms: sensors, actuators, wireless networks
  - Reliability and survivability
  - Infrastructure
  - Services
  - Legislation
  - ……

Picoradio Sensor Networks (BWRC)



- Control Environmental parameters (temperature, humidity…)
- Minimize Power consumption
- Cheap (<0.5$) and small (< 1 cm^3)
- Large numbers of nodes — between 0.05 and 1 nodes/m^2
- Limited operation range of network — maximum 50-100 m
- Low data rates per node — 1-10 bits/sec average
- Low mobility (at least 90% of the nodes stationary)

- Key challenges
  - Satisfy tight performance and cost constraints (especially power consumption)
  - Identify Layers of Abstraction (Protocol Stack)
  - Develop distributed algorithms (e.g. locationing, routing) for ubiquitous computing applications
  - Design Embedded System Platform to implement Protocol Stack efficiently


Government Operations
Gas & Oil Storage and Delivery
Emergency Services
Water Supply Systems
Critical Infrastructures
Telecommunications
Banking & Finance
Electrical Energy
Transportation

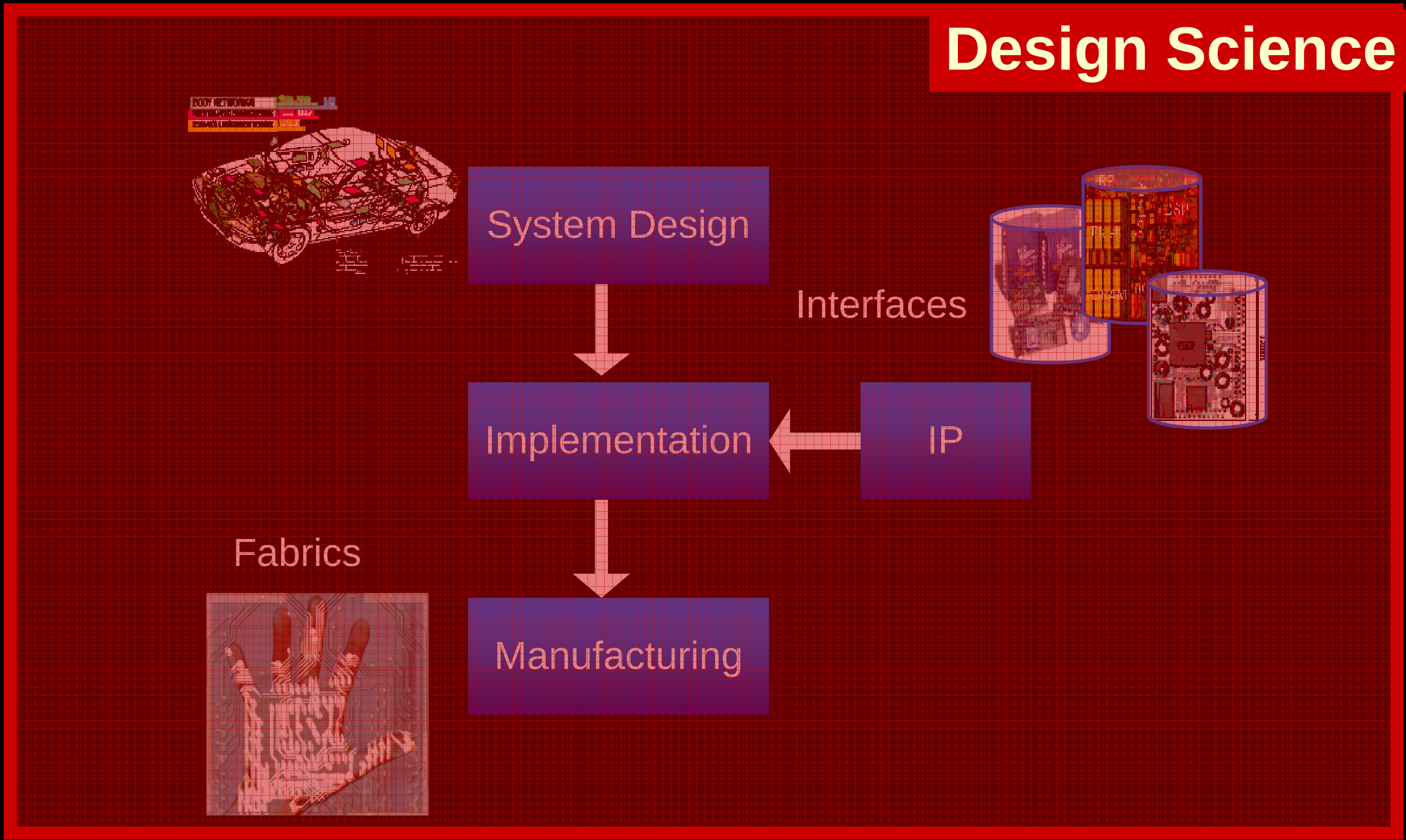
Secure Network Embedded SystEms (SENSE)

- Networked embedded systems and distributed control creates a new generation of future applications: new infrastructures
- We need to think about how to prevent the introduction of vulnerabilities via this exciting technology
- Security, Networking, Embedded Systems

Outline for the Introduction

- Examples of Embedded Systems
- Their Impact on Society
- Design Challenges
- Embedded Software and Control

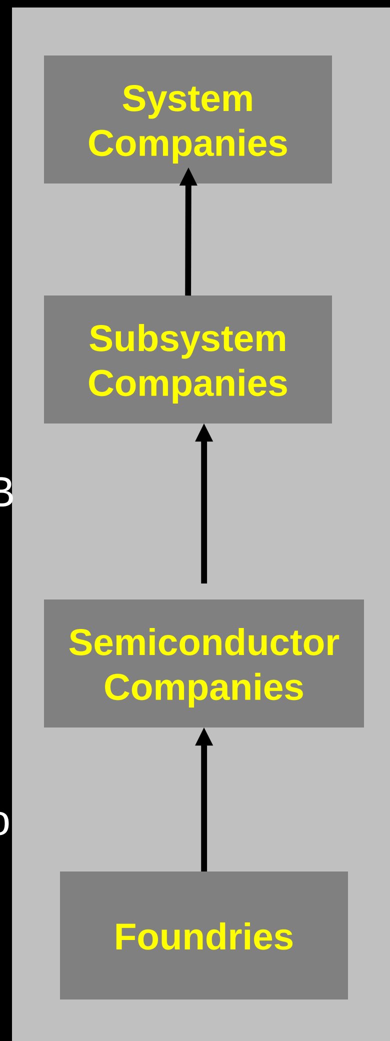
Opportunity:
Electronic Systems Design Chain



Disaggregation:
Complex Design Chain Management

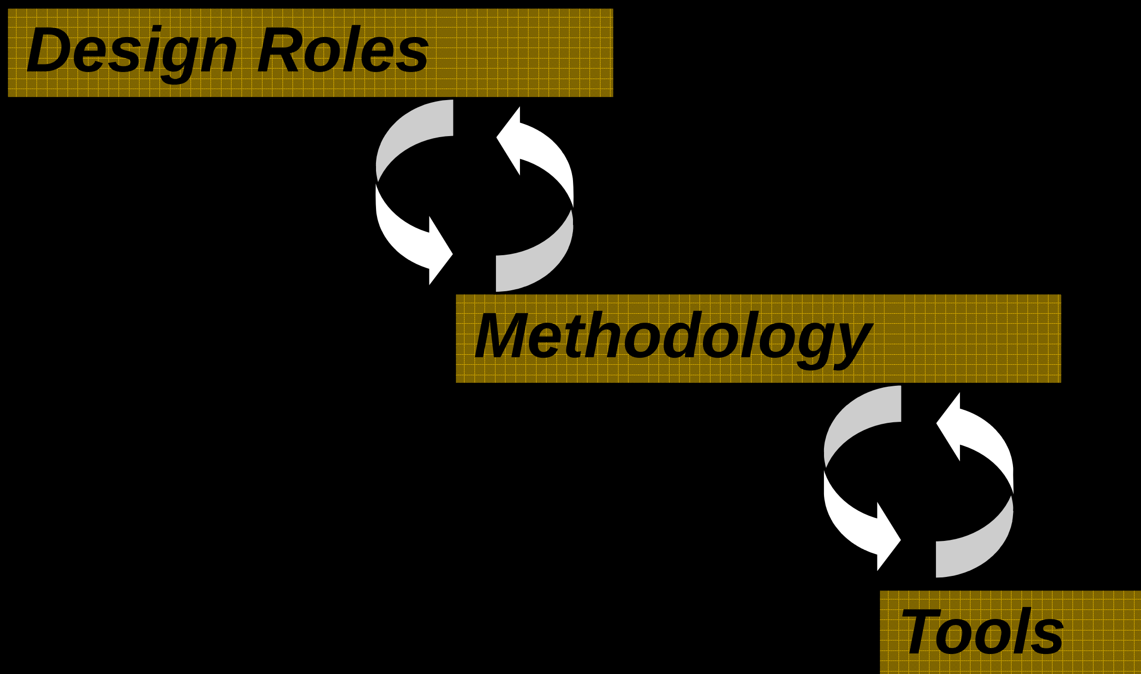
Supply Chain
- Movement of tangible goods from sources to end market
- Supply Chain Management is $3.8B market projected to be $20B in 2005

Design Chain
- Movement of technology (IP and knowledge) from sources to end market
- Design Chain Management is an untapped market

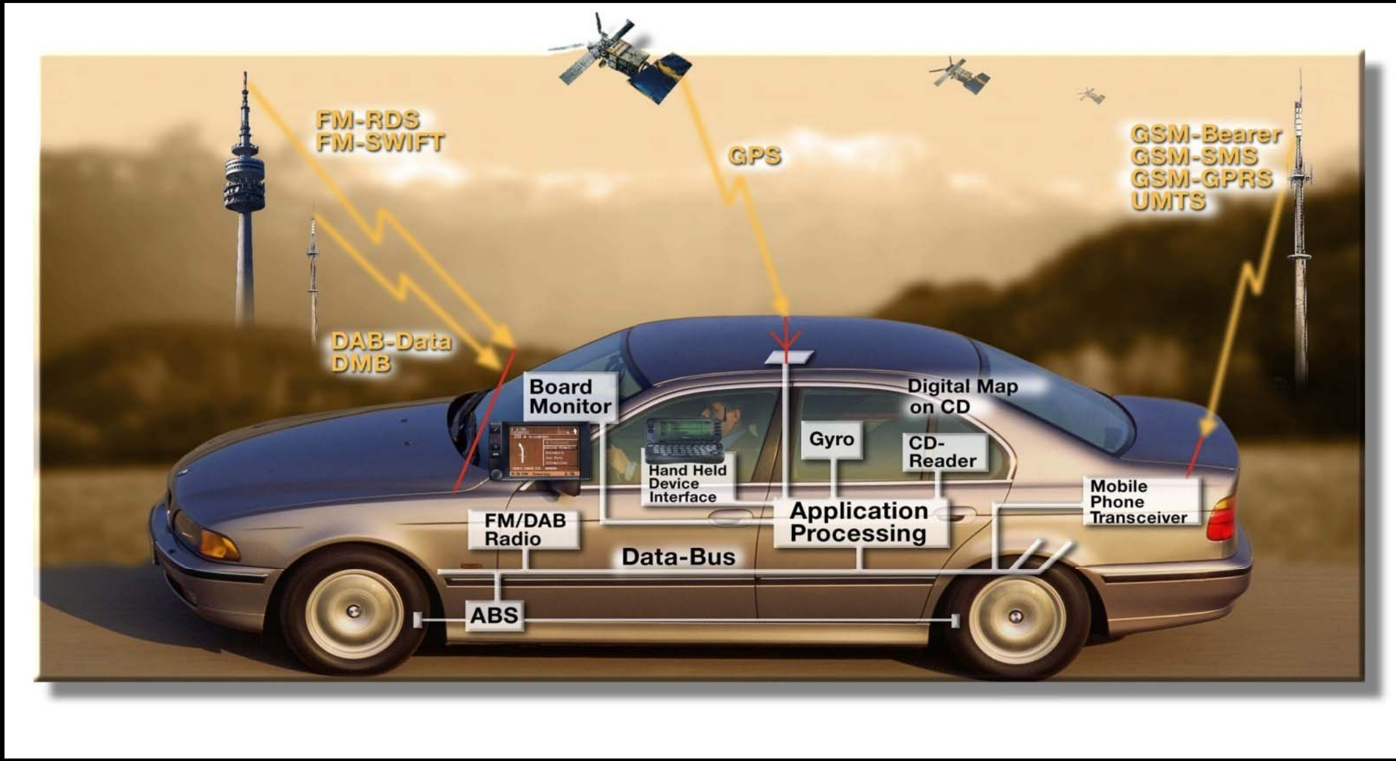


- EDA
- Design Services
- Embedded Software
- Software Development Tools
- Processor & Hardware IP
- IC Packaging & Test
- Process & Yield Services
- Mechanical CAD
- Contract Manufacturing
- System Test Equipment
- Fabrication Equipment

Supply Chain:
Design Roles-> Methodology->Tools

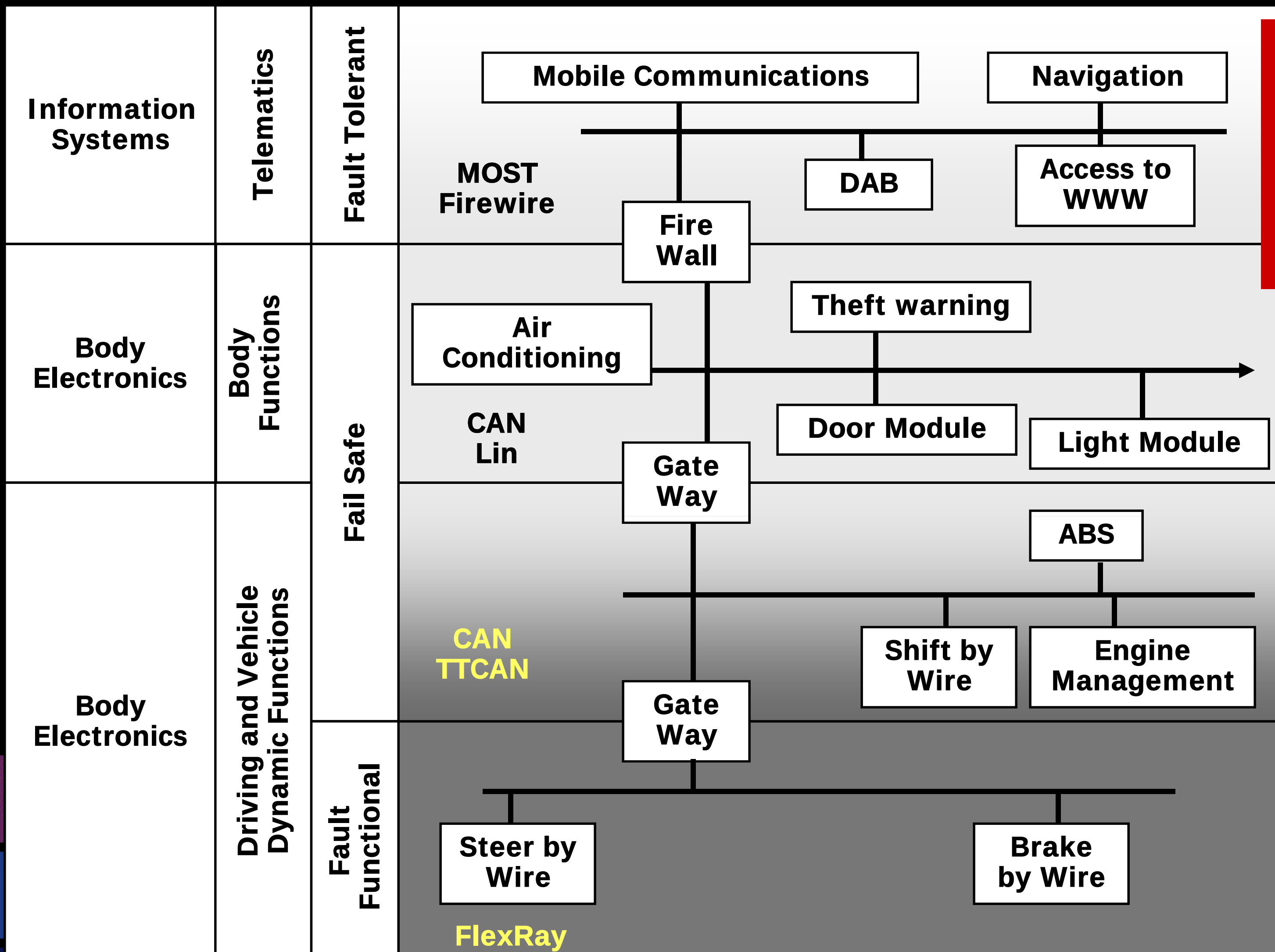


Automotive Supply Chain:
Car Manufacturers



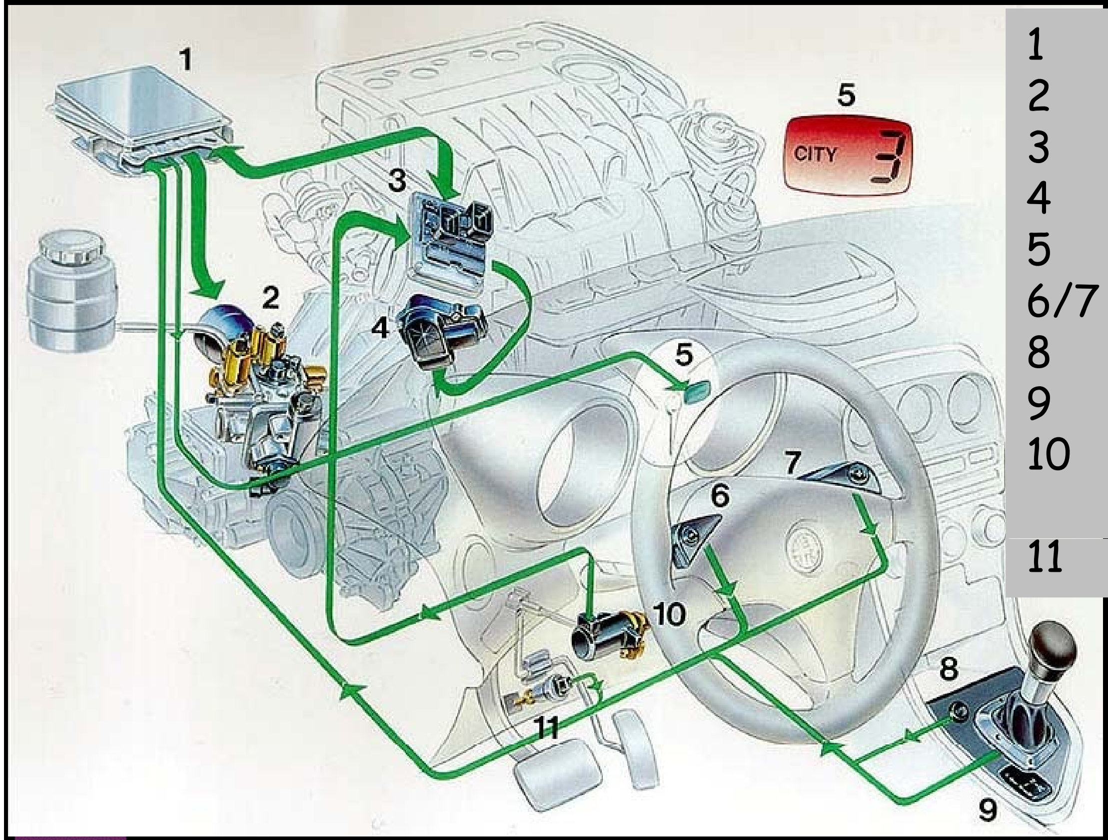
Product Specification & Architecture Definition
(e.g., determination of Protocols and Communication standards)
System Partitioning and Subsystem Specification
Critical Software Development
System Integration

Electronics for the Car: A Distributed System



Today, more than 80 Microprocessors and millions of lines of code

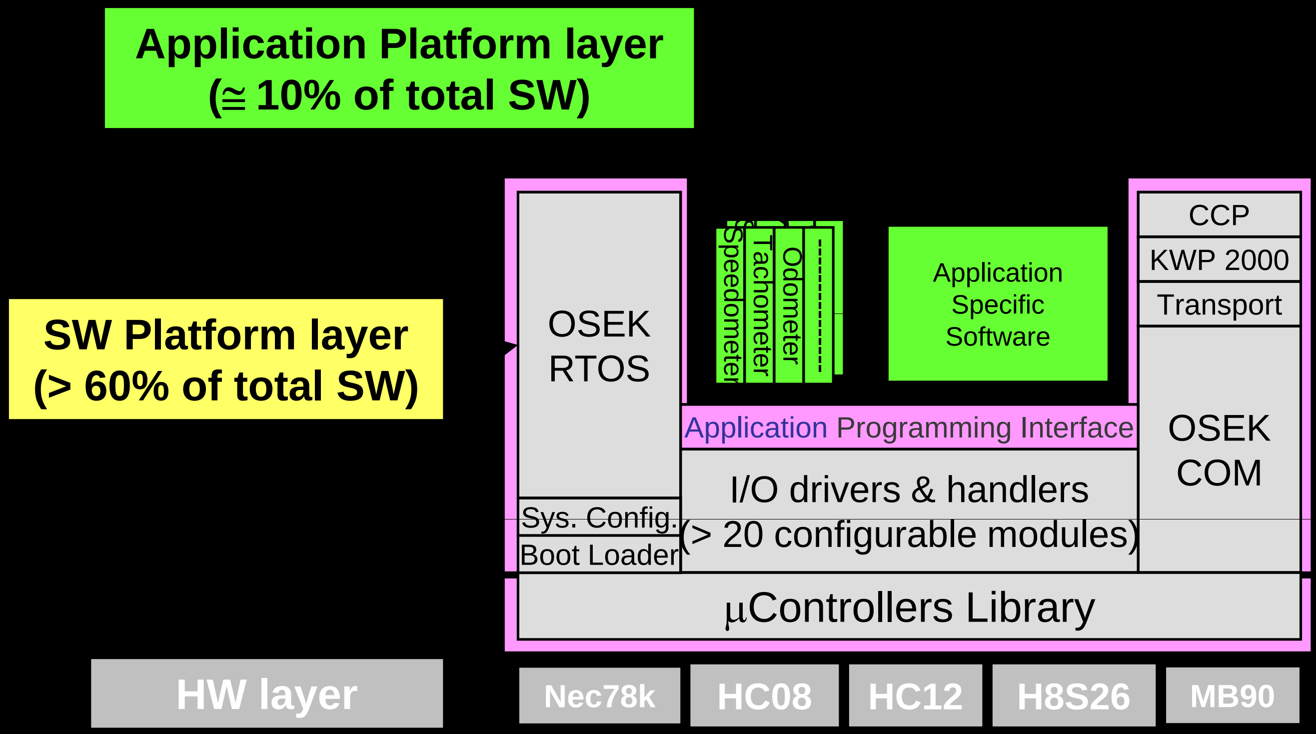
Automotive Supply Chain:
Tier 1 Subsystem Providers



1	Transmission ECU
2	Actuation group
3	Engine ECU
4	DBW
5	Active shift display
6/7	Up/Down buttons
8	City mode button
9	Up/Down lever
10	Accelerator pedal position sensor
11	Brake switch

- Subsystem Partitioning
- Subsystem Integration
- Software Design: Control Algorithms, Data Processing
- Physical Implementation and Production

Automotive Supply Chain: Subsystem Providers

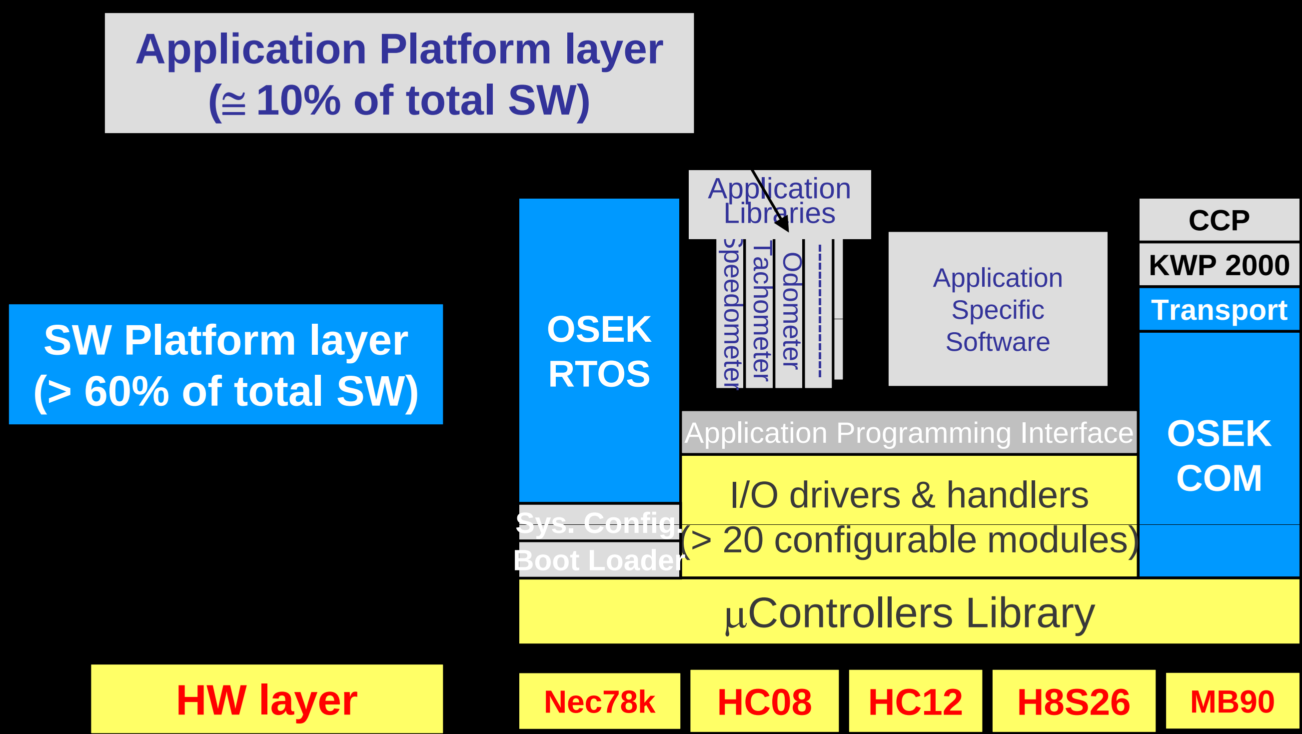


- Platform Integration
- Software Design

"firmware" and "glue software"

"Application"

Automotive Supply Chain: Platform & IP Providers



- “Software” platform — RTOS and communication layer
- “Hardware” platform — Hardware and IO drivers

Outline for the Introduction

- Examples of Embedded Systems
- Their Impact on Society
- Design Challenges
- Embedded Software and Control

How Safe is Our Real-Time Software?

Computing for Embedded Systems

Mars, December 3, 1999
Crashed due to un-initialized variable


$4 billion development effort
40-50% system integration & validation cost

Complexity, Quality, & Time To Market today

	PWT UNIT 	BODY GATEWAY 	INSTRUMENT CLUSTER 	TELEMATIC UNIT 
Memory	256 Kb	128 Kb	184 Kb	8 Mb
Lines Of Code	50.000	30.000	45.000	300.000
Productivity	6 Lines/Day	10 Lines/Day	6 Lines/Day	10 Lines/Day*
Residual Defect Rate @ End Of Dev	3000 Ppm	2500 ppm	2000ppm	1000 ppm
Changing Rate	3 Years	2 Years	1 Year	< 1 Year
Dev. Effort	40 Man-yr	12 Man-yr	30 Man-yr	200 Man-yr
Validation Time	5 Months	1 Month	2 Months	2 Months
Time To Market	24 Months	18 Months	12 Months	< 12 Months

*** C^{++} CODE**

FABIO ROMEO, Magneti-Marelli
DAC, Las Vegas, June 20th, 2001

How is Embedded Software Different from Ordinary Software?

- It has to work
- One or more (very) limited resources
  - Registers
  - RAM
  - Bandwidth
  - Time

 Alex Aiken slide

Devil's Advocate

- So what's different?
- All software works with limited resources
- We have compiler technology to deal with it
  - Various forms of program analysis

Example: Registers

- All machines have only a few registers
- Compiler uses the registers as best as it can
  - *Spills* the remaining values to main memory
  - Manages transfers to and from registers
- The programmer feels she has 1 registers

 Alex Aiken slide

The Standard Trick

- This idea generalizes
- For scarce resource X
  - Manage X as best as we can
  - If we need more, fall back to secondary strategy
  - Give the programmer a nice abstraction

The Standard Trick

- This idea generalizes
- For scarce resource X
  - Manage X as best we can
  - *Any correct heuristic is OK, no matter how complex*
  - If we need more, fall back to secondary strategy
  - *Focus on average case behavior*
  - *Give the programmer a nice abstraction*

 Alex Aiken slide

Examples of the Standard Trick

- Compilers
  - Register allocation
  - Dynamic memory management
- OS
  - Virtual memory
  - Caches

Summary: abstract and hide complexity of resources

 Alex Aiken slide

What's Wrong with This?

- Embedded systems have limited resources
- Meaning hard limits
  - Cannot use more time
  - Cannot use more registers
- The compiler must either
  - Produce code within these limits
  - Report failure
- The standard trick is anathema to embedded systems
  - Can't hide resources

 Alex Aiken slide

Revisiting the Assumptions

- *Any correct heuristic is OK, no matter how complex*
  - Embedded programmer must understand reasons for failure
  - Feedback must be relatively straightforward
- *Focus on average case behavior*
  - Embedded compiler must reason about the worst case
  - Cannot improve average case at expense of worst case
- *Give the programmer a nice abstraction*
  - Still need abstractions, but likely different ones

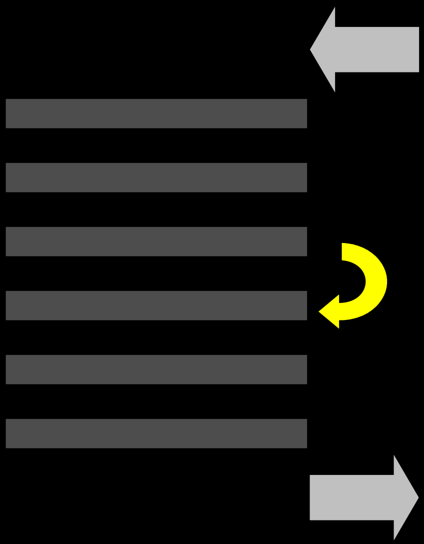
 Alex Aiken slide

Another Traditional Systems Science - Computation, Languages, and Semantics

Alan Turing

Everything "computable" can be given by a terminating sequential program.

- Functions on bit patterns
- Time is irrelevant
- Non-terminating programs are defective



sequence

Current fashion – Pay Attention to "Non-functional properties"

- Time
- Security
- Fault tolerance
- Power consumption
- Memory management

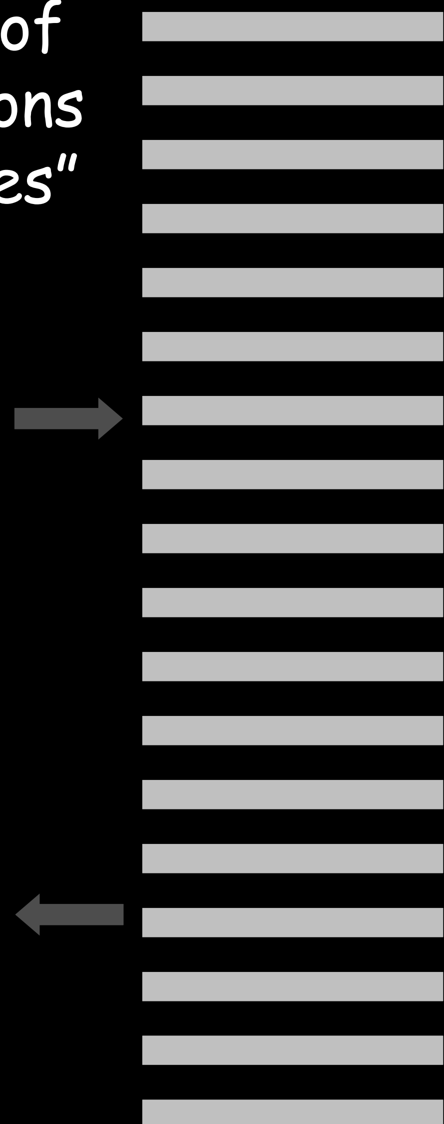
But the formulation of the question is very telling:

> How is it that *when* a braking system applies the brakes is any less a *function* of the braking system than *how much* braking it applies?

Processes and Process Calculi

Infinite sequences of state transformations are called "processes" or "threads"

incoming message



outgoing message

In prevailing software practice, processes are sequences of external interactions (total orders).

And messaging protocols are combined in ad hoc ways.

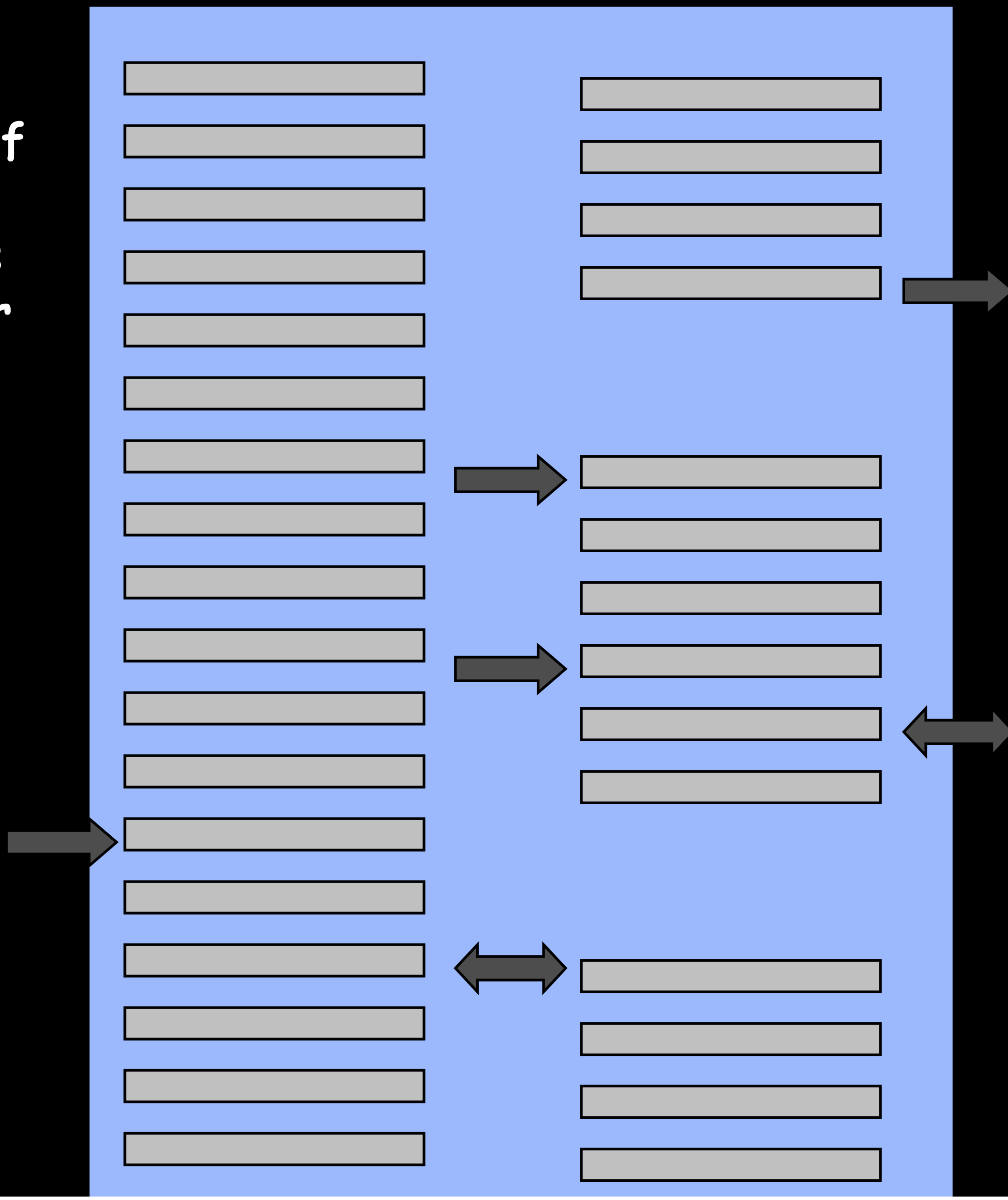
Interacting Processes – Concurrency as Afterthought

Software realizing these interactions is written at a very low level (e.g., semaphores). *Very* hard to get it right.

Interacting Processes – Not Compositional

An aggregation of processes is not a process (a total order of external interactions). What is it?

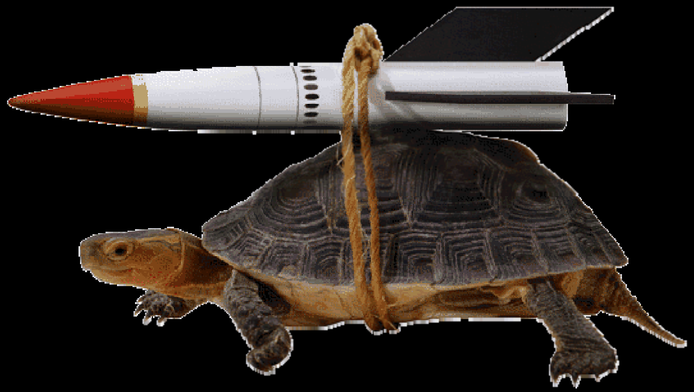
Many software failures are due to this ill-defined composition.



Compositionality

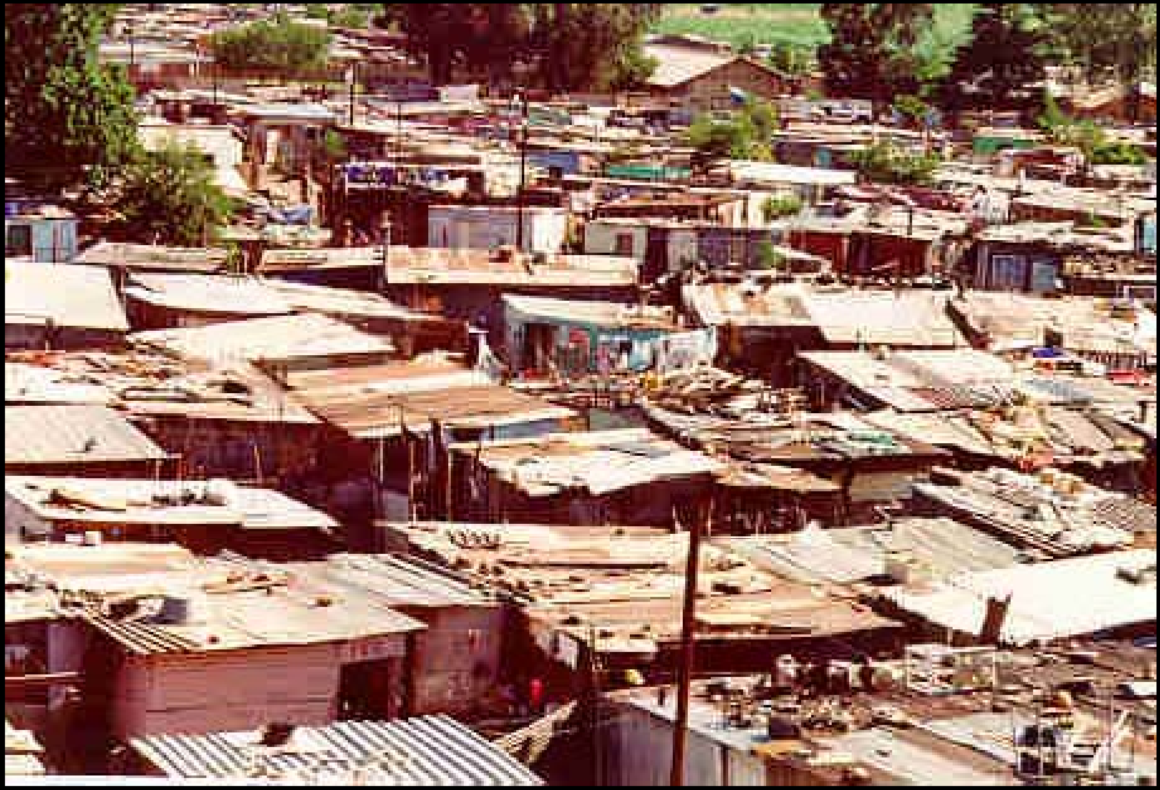
Non-compositional formalisms lead to very awkward architectures.

What About Real Time?



"Make it faster!"

Software Architecture Today



Design “Practice”

Design Science:
Build upon Solid Foundations

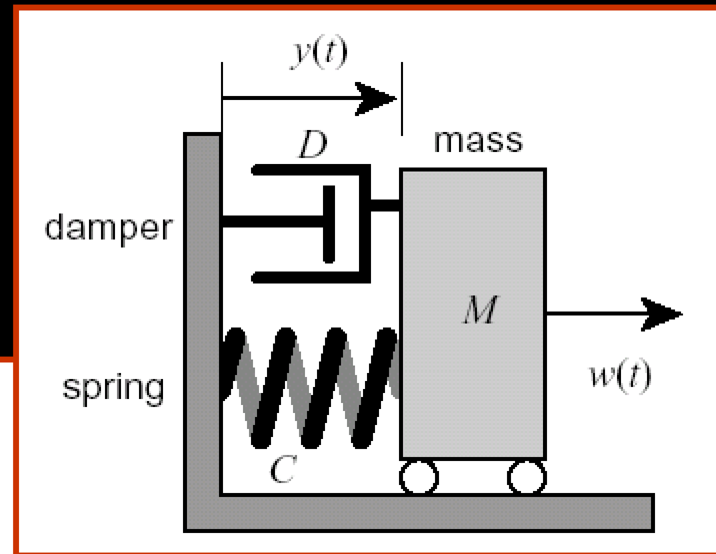
Software Architecture Tomorrow?

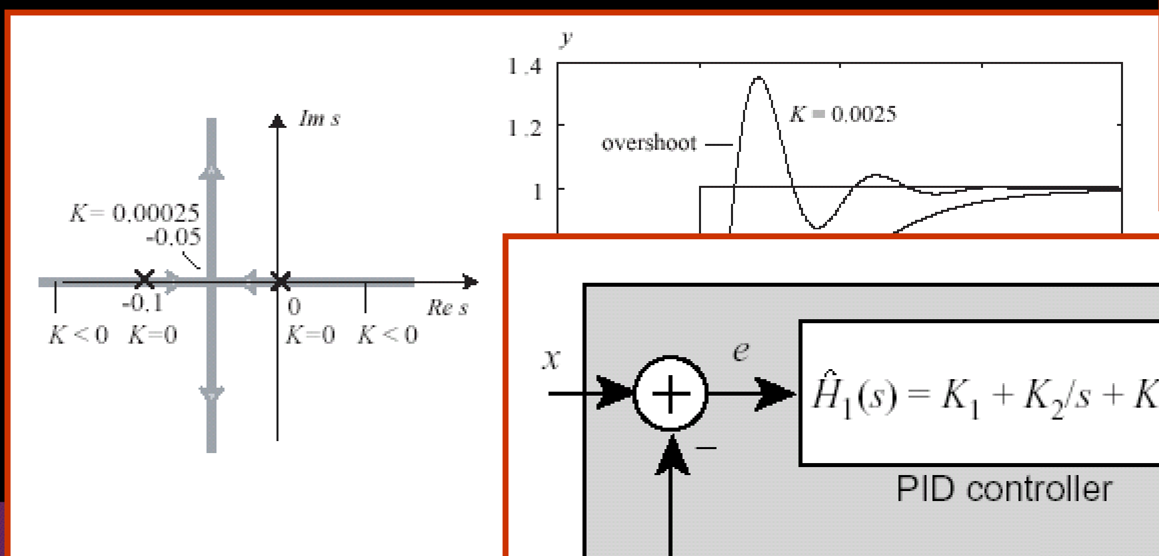
The Goal (CHESS Project)

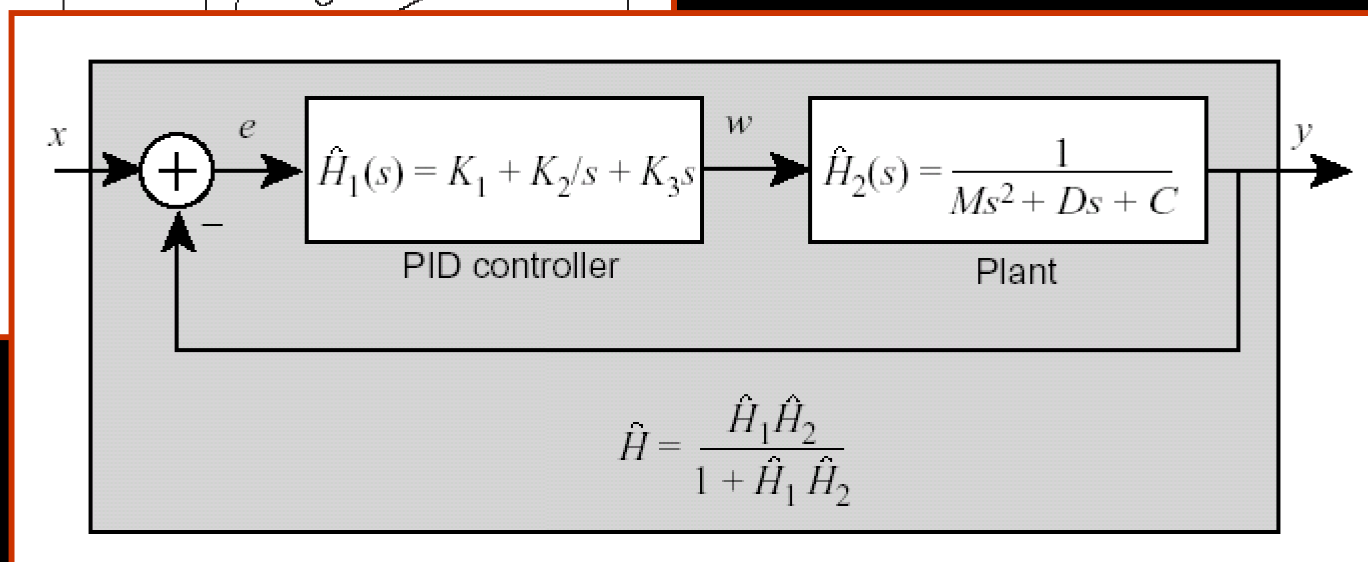
- To create a modern computational systems science and systems design practice with
  - Concurrency
  - Composability
  - Time
  - Hierarchy
  - Heterogeneity
  - Resource constraints
  - Verifiability
  - Understandability

A Traditional Systems Science – Feedback Control Systems

- Models of continuous-time dynamics
- Stability analysis
- But not accurate for software controllers

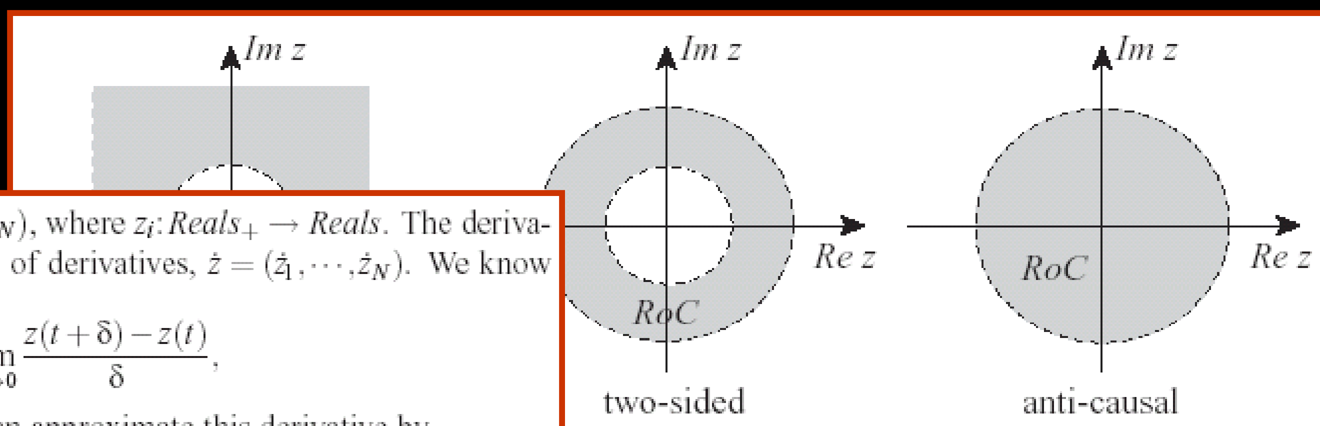






Discretized Model –
A Step Towards Software

- **Numerical integration techniques provided ways to get from the continuous idealizations to computable algorithms.**
- **Discrete-time signal processing techniques offer the same sophisticated stability analysis as continuous-time methods.**
- **But it's *still* not accurate for software controllers**



In general, z is an N-tuple, $z = (z_1, \cdots, z_N)$, where z_i: $Reals_+ \to Reals$. The derivative of an N-tuple is simply the N-tuple of derivatives, $\dot{z} = (\dot{z}_1, \cdots, \dot{z}_N)$. We know from calculus that

$$\dot{z}(t) = \frac{dz}{dt} = \lim_{\delta \to 0} \frac{z(t+\delta) - z(t)}{\delta},$$

and so, if $\delta > 0$ is a small number, we can approximate this derivative by

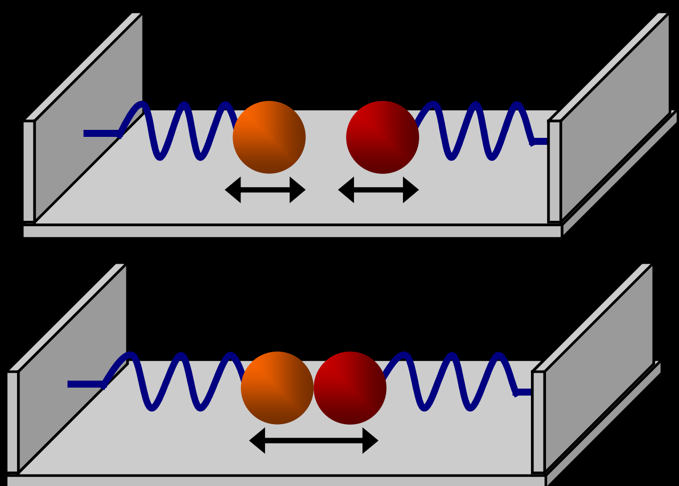
$$\dot{z}(t) \approx \frac{z(t+\delta) - z(t)}{\delta}.$$

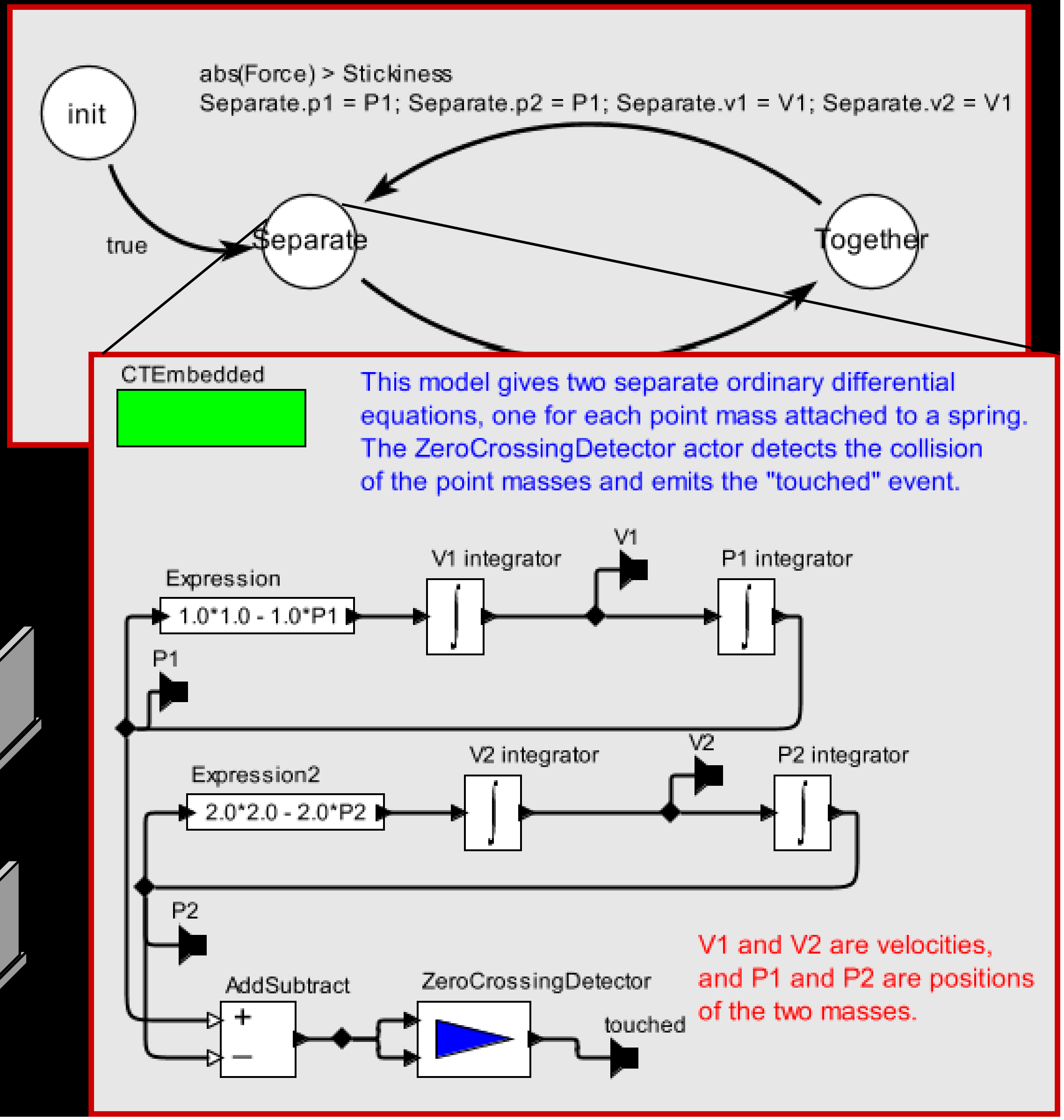
Using this for the derivative in the left-hand side of (5.50) we get

$$z(t+\delta) - z(t) = \delta g(z(t), v(t)). \quad (5.51)$$

Hybrid Systems –
Reconciliation of Continuous & Discrete

- **But it's *still* not accurate for software controllers**

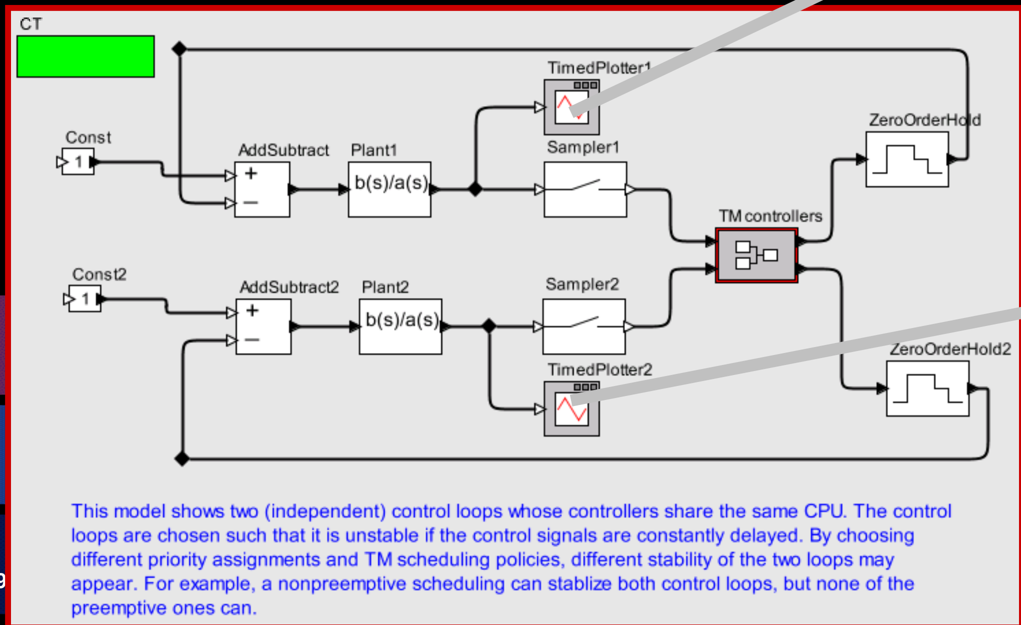


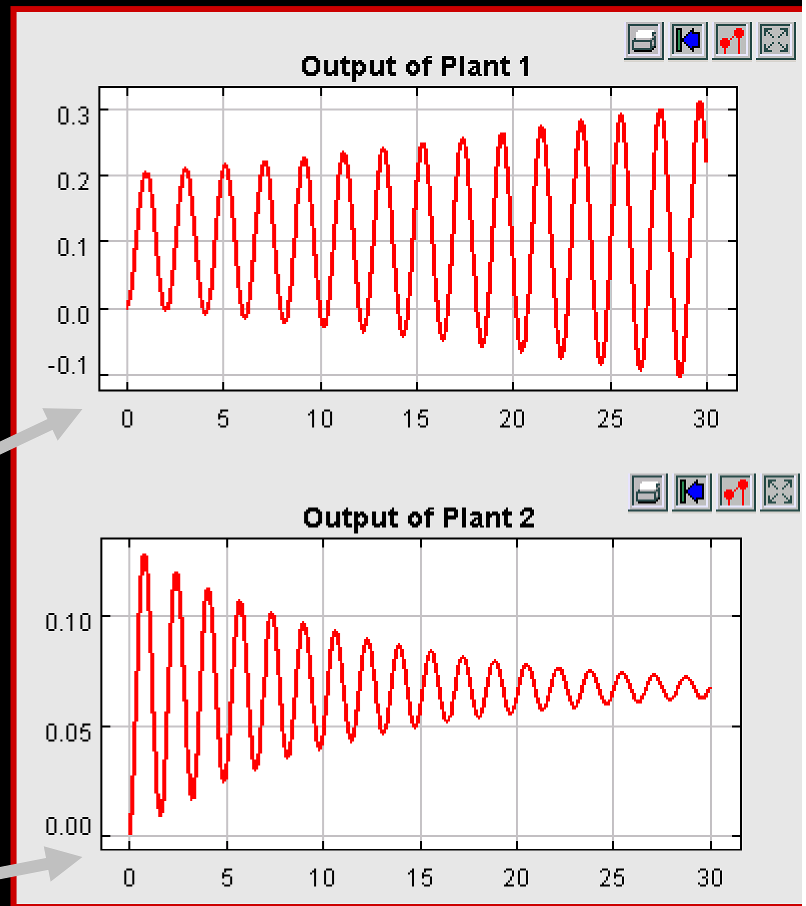


Timing in Software is More Complex Than What the Theory Deals With

An example (Jie Liu) models two controllers sharing a CPU under an RTOS. Under preemptive multitasking, only one can be made stable (depending on the relative priorities). Under non-preemptive multitasking, both can be made stable.

Where is the theory for this?





Foundational Theory Research …

- The science of computation has systematically abstracted away the physical world. The science of physical systems has systematically ignored computational limitations.
  Embedded software systems, however, engage the physical world in a computational manner.

- It is time to construct a Hybrid Systems Science that is simultaneously computational and physical.
  Time, concurrency, robustness, continuums, and resource management must be remarried to computation.