



**EE249**

**Embedded System Design:  
Models, Validation and  
Synthesis**

**Alberto Sangiovanni Vincentelli**



Chiesa di S. Simeone Profeta  
1. Palazzo Lancellotti, 2. Abbatia della Famiglia, 3. Arco detto di Formi, 4. Abbatia e Chiesa parrocchiale di S. Simeone, 5. Palazzo già dei Ceri.



Copyright © The National Gallery 1998. All rights reserved

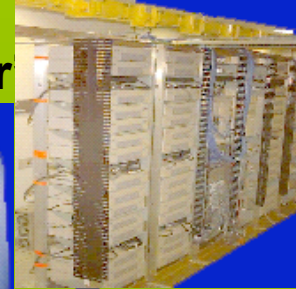
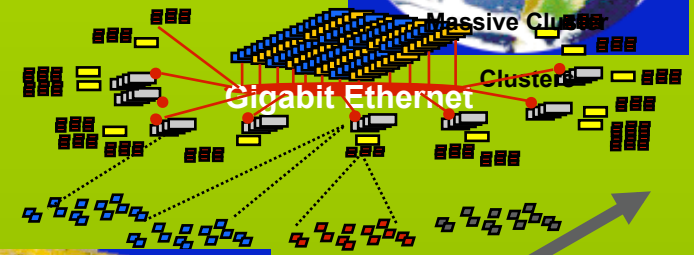
"I believe we are now entering the Renaissance phase of the Information Age, where creativity and ideas are the new currency, and invention is a primary virtue, where technology truly has the power to transform lives, not just businesses, where technology can help us solve fundamental problems."

Carly Fiorina, CEO, Hewlett Packard Corporation

# eMerging Societal-Scale Systems



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



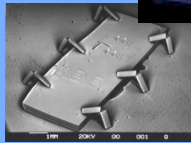
**Scalable, Reliable,  
Secure Services**

**Information  
Appliances**

**“Server”**

**“Client”**

**Embedded  
Systems**



**MEMS**

**BioMonitoring**



# Embedded 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



cellular phones

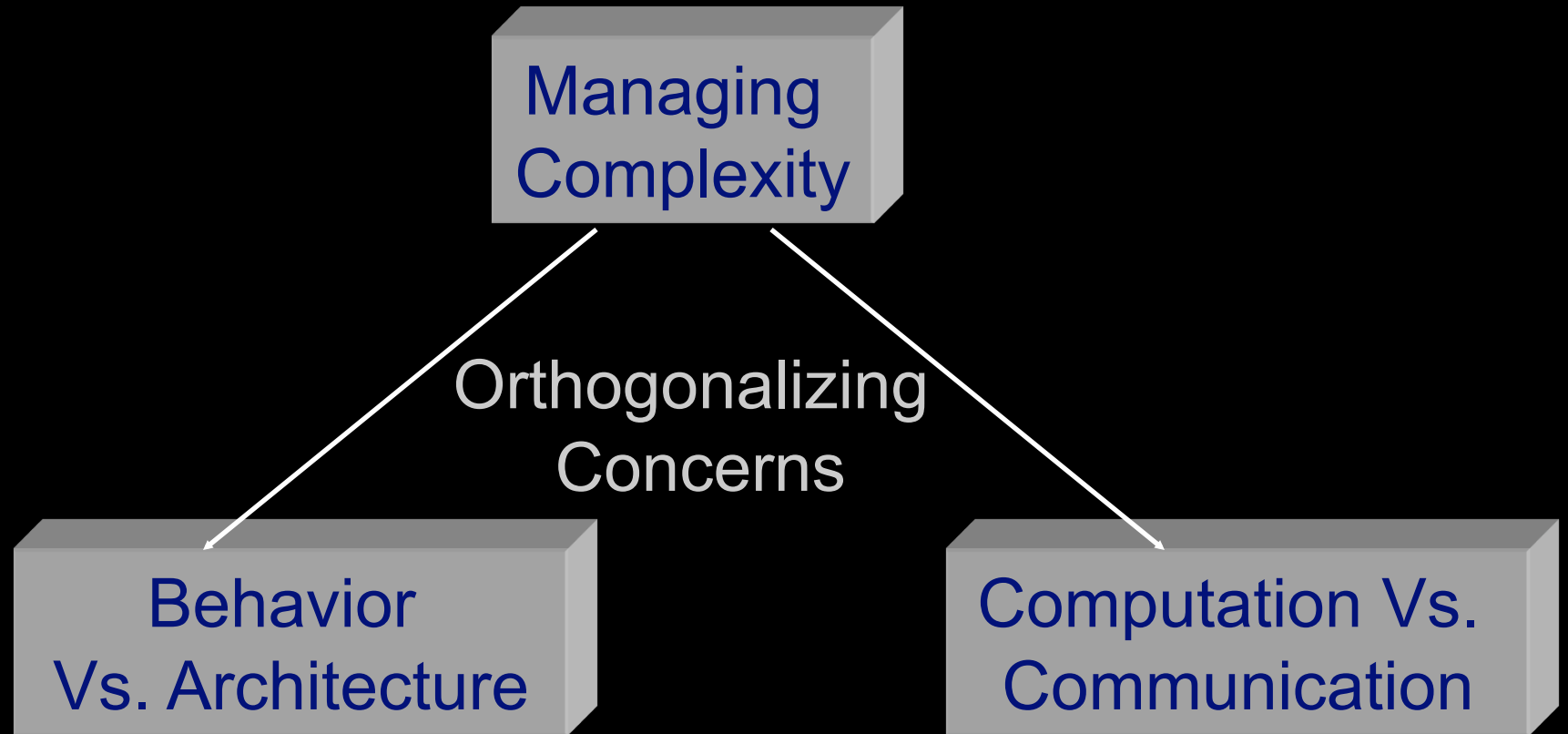


# Observations

- 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

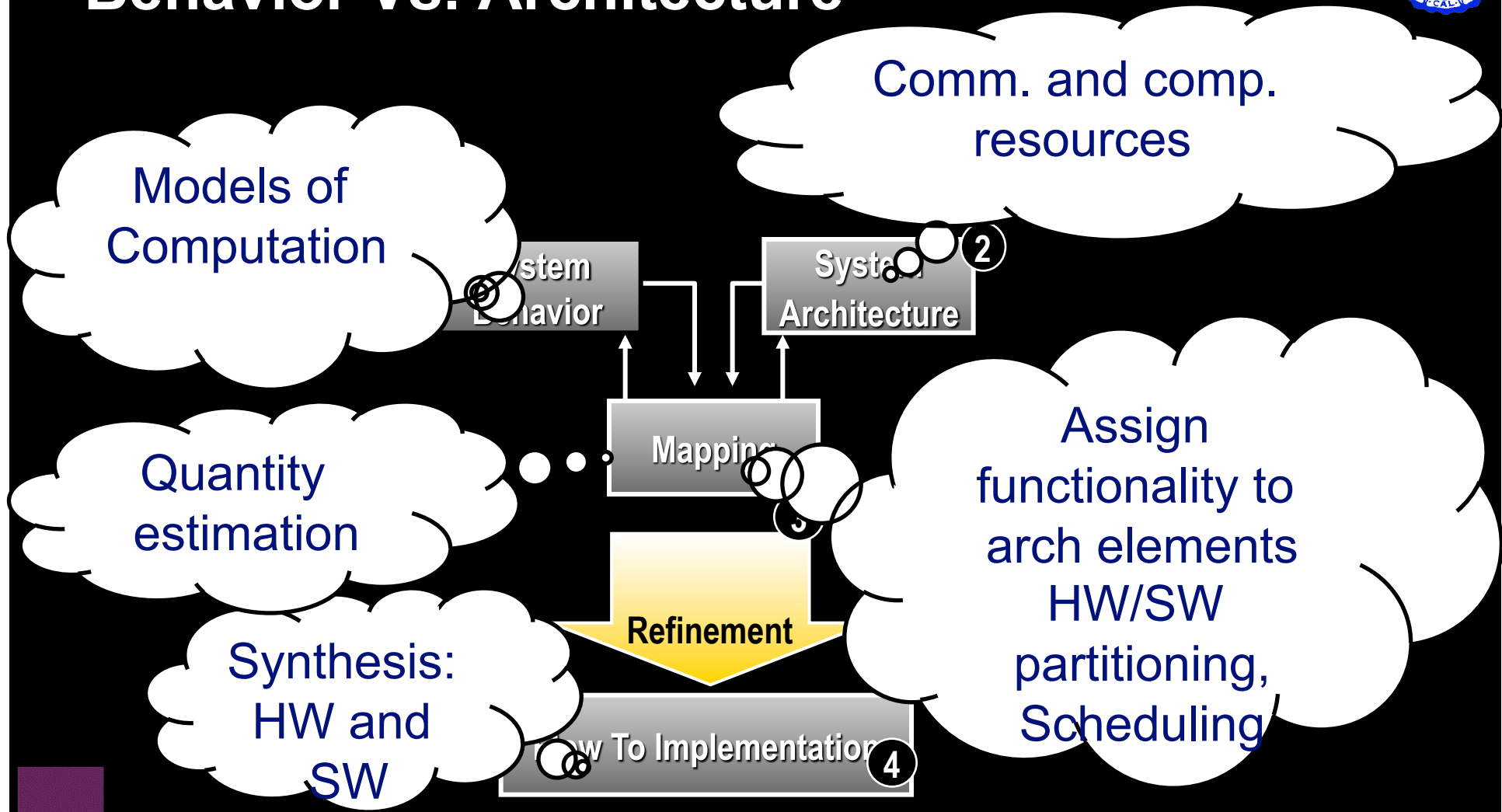


# Course overview





# Behavior Vs. Architecture



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





# 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)
- Teaching Assistant:
  - **Xuening Sun**, [xuening@eecs.berkeley.edu](mailto:xuening@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
- Discussion Session (Tu. 5-6)
  - Each student (possibly 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://chess.eecs.berkeley.edu/design/index.html>



# Outline of the course

<i>Part 1: Introduction</i>	Design complexity, Example of embedded systems, traditional design flow, Platform-Based Design
<i>Part 2: Functional modeling, analysis and simulation</i>	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
<i>Part 3: Architecture and performance abstraction</i>	Definition of architecture, examples. Distributed architecture, coordination, communication. Real time operating systems, scheduling of computation and communication.
<i>Part 4: Mapping</i>	Definition of mapping and synthesis. Software synthesis, quasi static scheduling. Behavioral synthesis. Communication Synthesis and communication-based design
<i>Part 5: Verification</i>	Validation vs Simulation. Verification of hybrid system. Interface automata and assume guarantee reasoning.
<i>Part 6: Applications</i>	Distributed Systems Avionics and Automotive: CAN, Flexray, Autosar Architecture, GM car architecture, Power generation subsystems in Airplanes, scheduling and timing analysis Building automation: BanNet, LonWorks, ZigBee with applications to energy efficiency and security



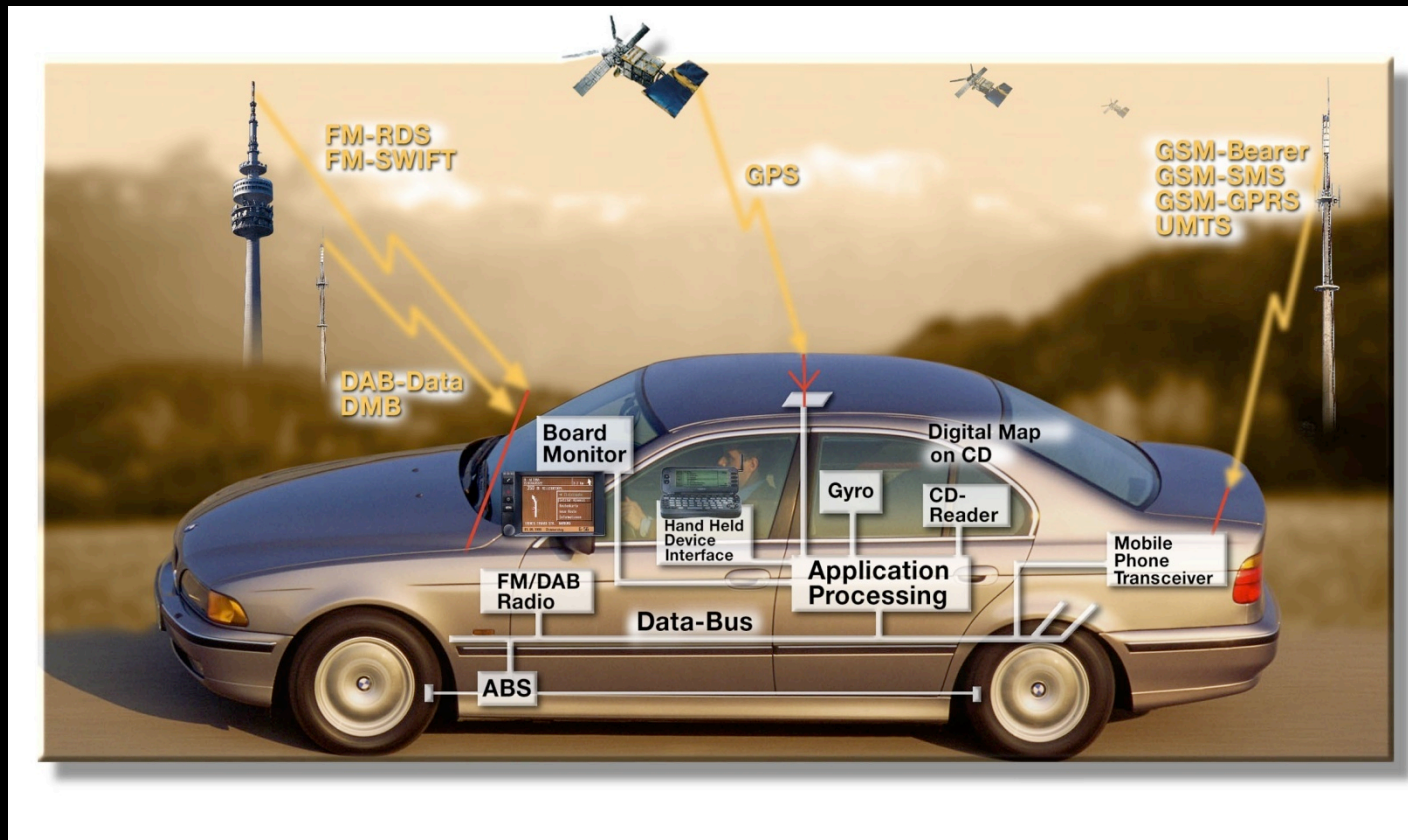
# Outline for the Introduction

- 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



TOYOTA



DAIMLERCHRYSLER



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

## Tier 1 Suppliers



DENSO

JOHNSON  
CONTROLS

BOSCH

DELPHI



90%+ of revenue  
from automotive

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

## IC Vendors



RENESAS



NEC

~15% of revenue  
from automotive

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

Source: Public financials, Gartner 2005

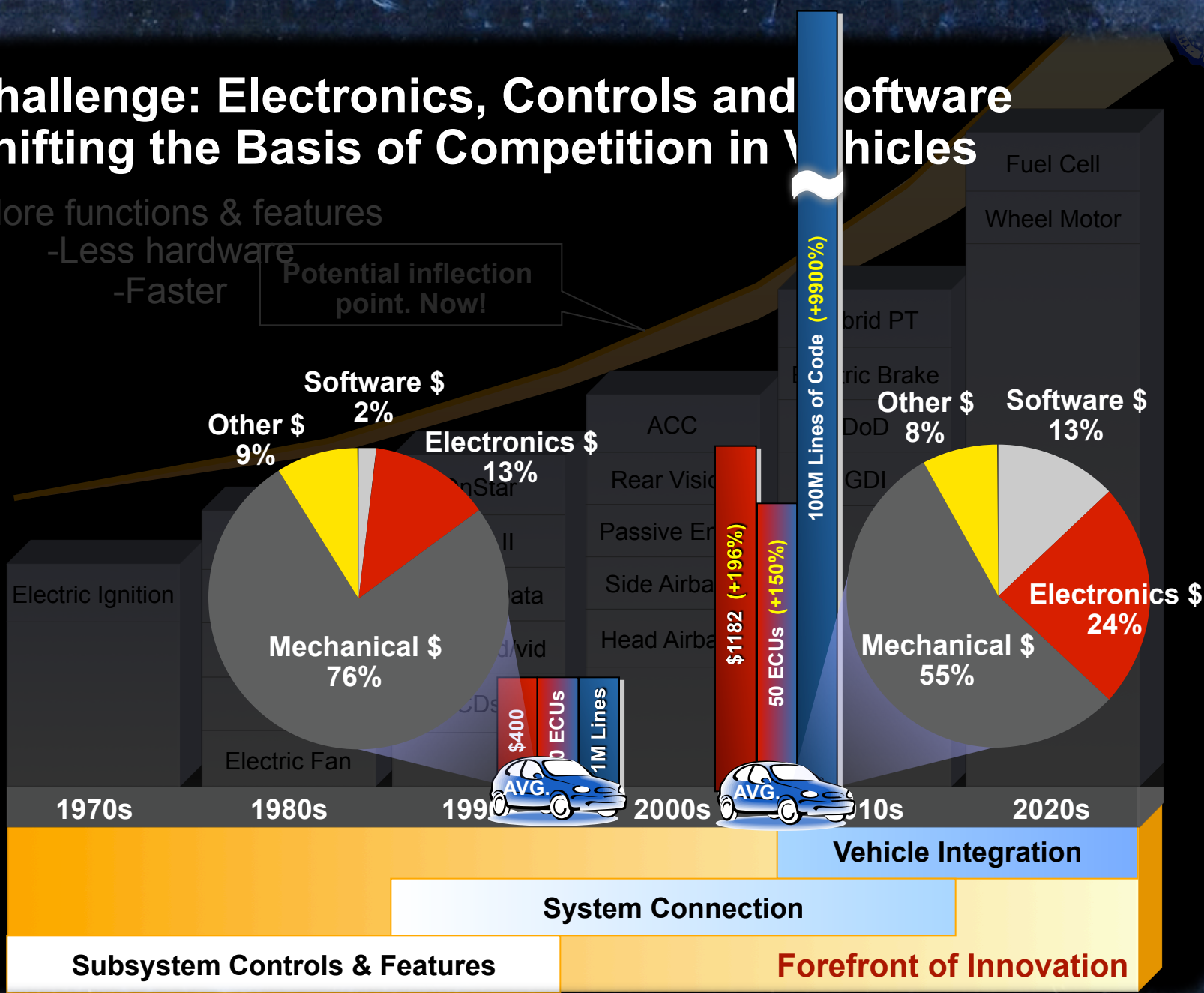


# Challenge: Electronics, Controls and Software Shifting the Basis of Competition in Vehicles

- More functions & features
- Less hardware
- Faster

Potential inflection point. Now!

Value from Electronics & Software





# GM SAC Vehicular Electronics, Controls and Software Study

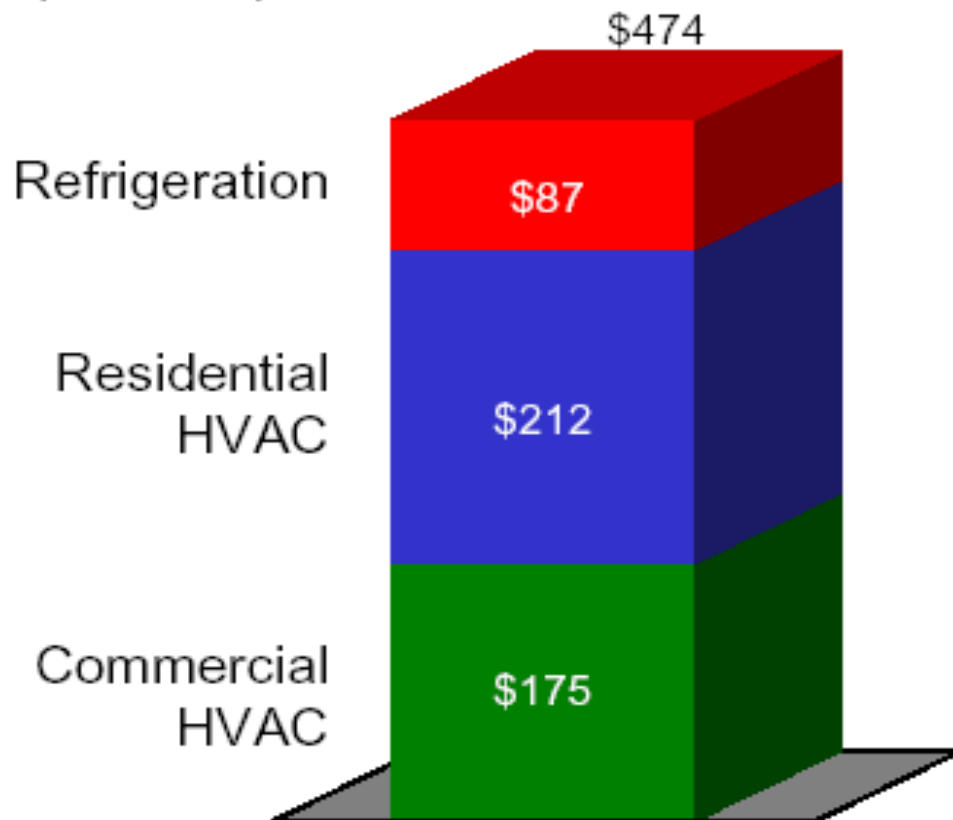
- Software content in automobiles could increase by 100 X over the next 5-6 years. Challenges will include:
  - Software system architecture
  - Partitioning for modularity & system reliability
  - Reuse
  - Standardization of interfaces



# CARRIER CONTROLS BUSINESS

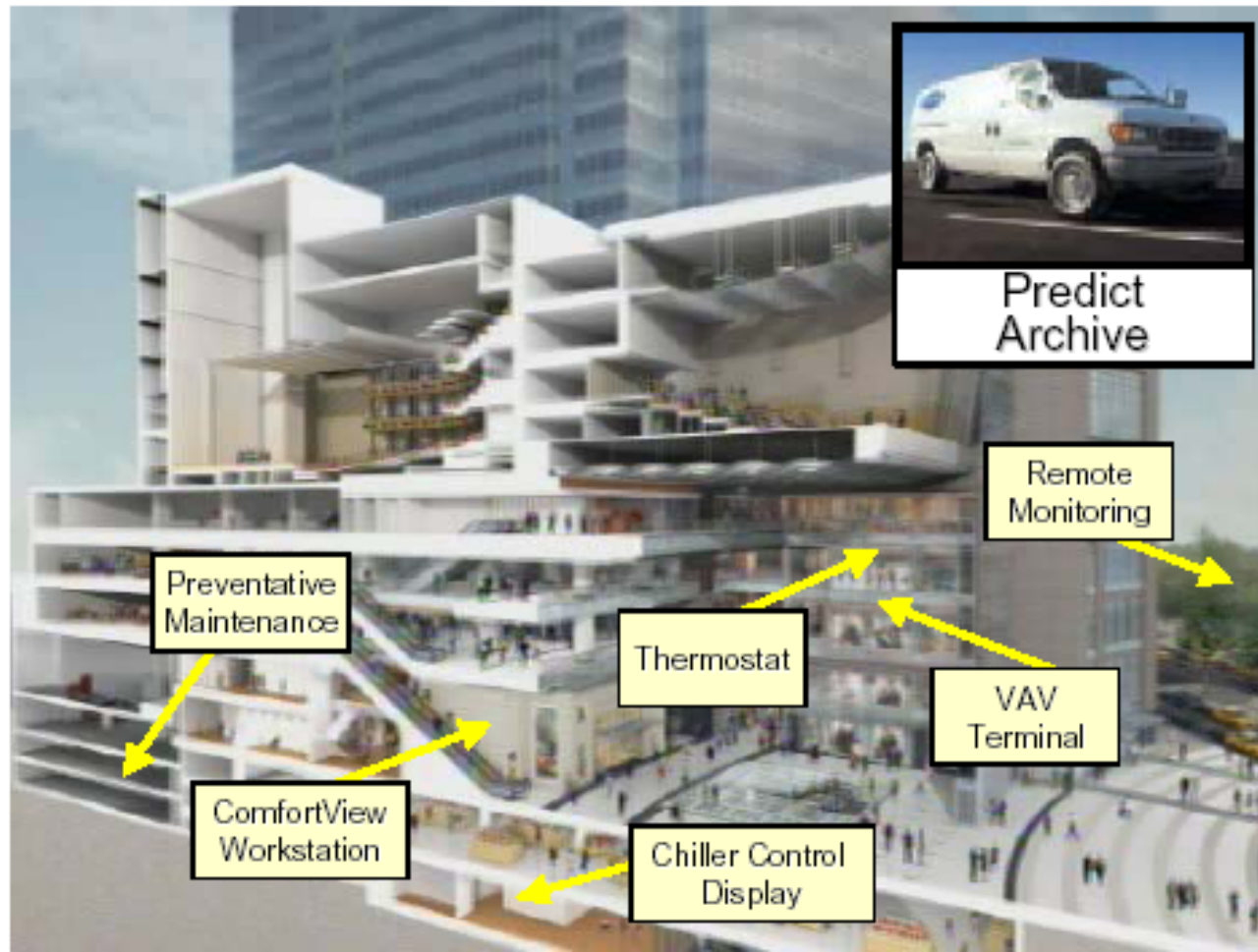
## Market segments

2001  
(\$ millions)



# FUNCTION OF CONTROLS

## Typical commercial HVAC application

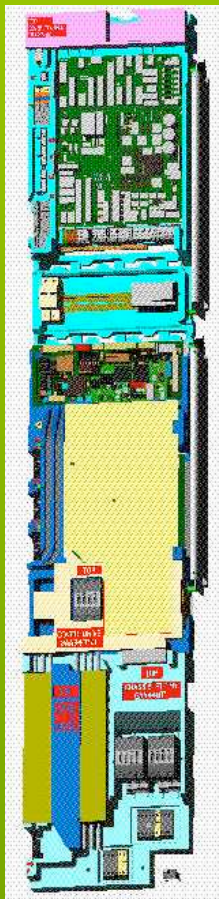


- Configure
- Sense
- Actuate
- Regulate
- Display
- Trend
- Diagnose
- Predict
- Archive

# OTIS Elevators

---

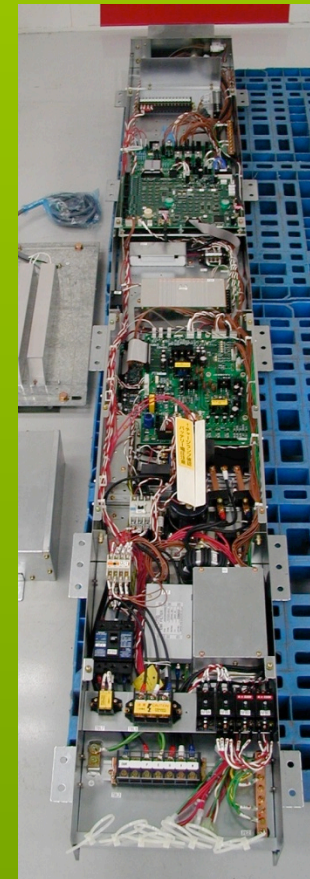
1. EN: GeN2-Cx



2. ANSI:  
Gen2/GEM



3. JIS:  
GeN2-JIS

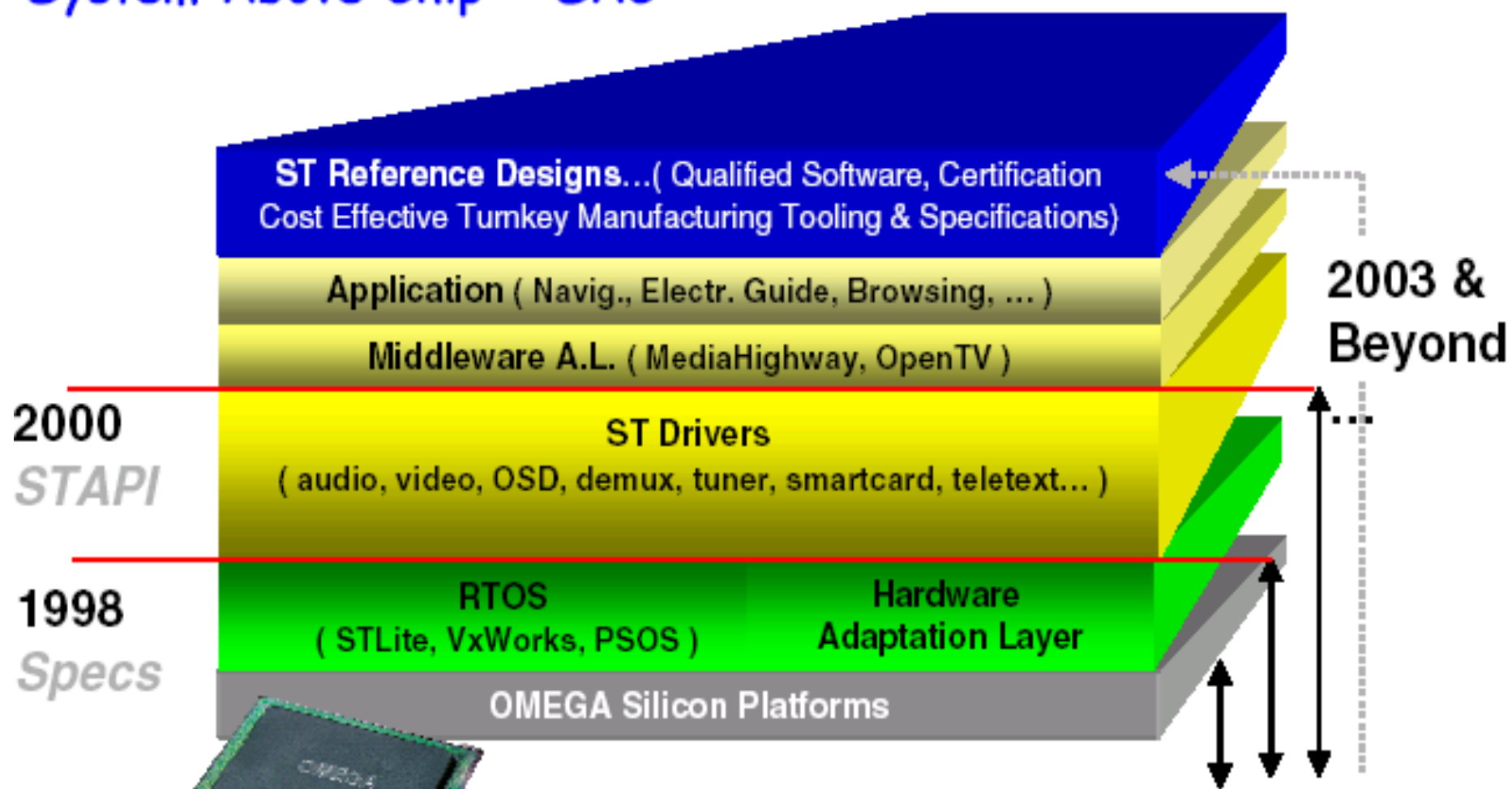


# Segments



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

CMG-Design

STMicroelectronics Confidential and Proprietary





# Consumer segments

## Common technology elements



Gaming

Bro

### 'Systems within systems'

Multimedia processors

Embedded  $\mu$ P

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**



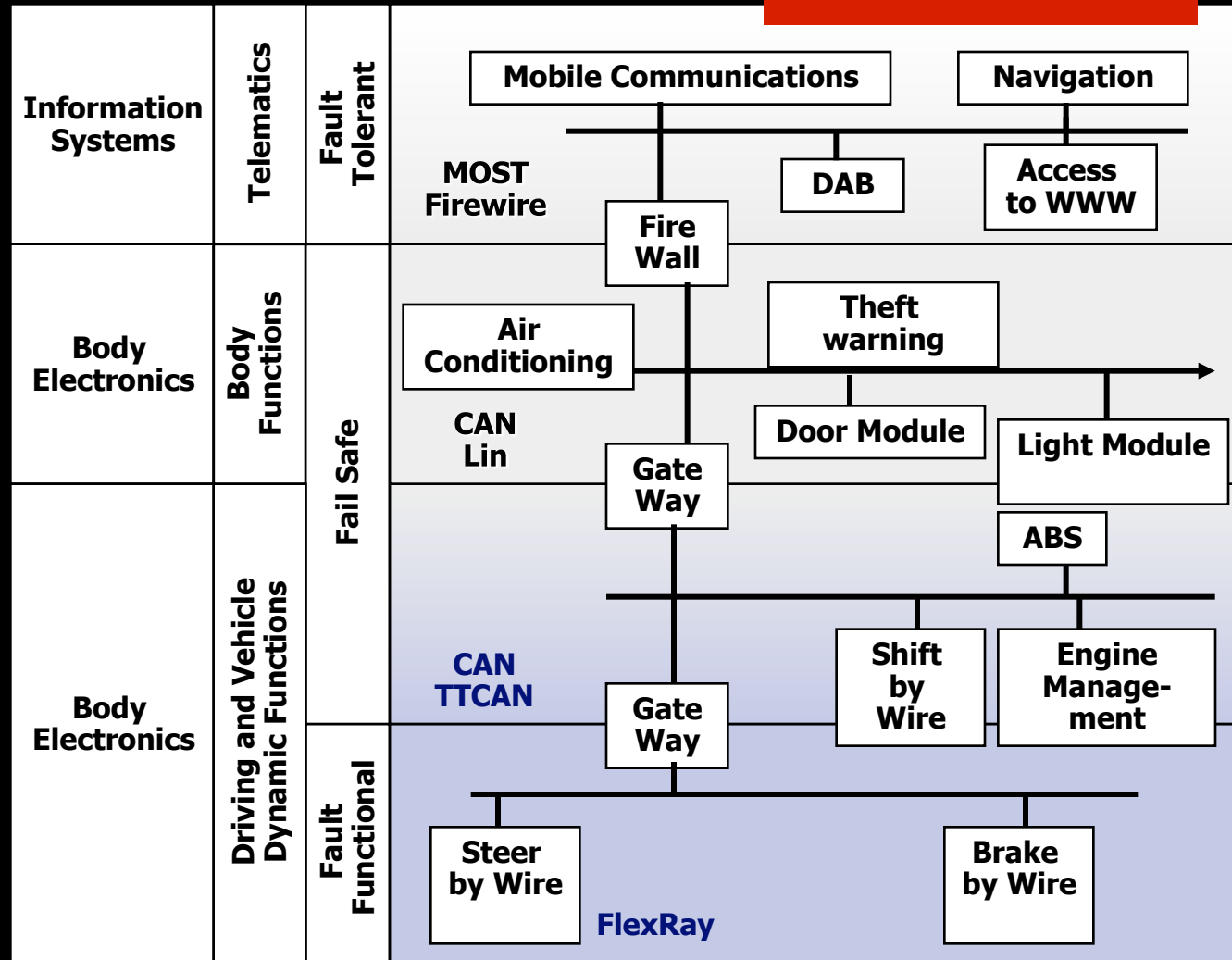
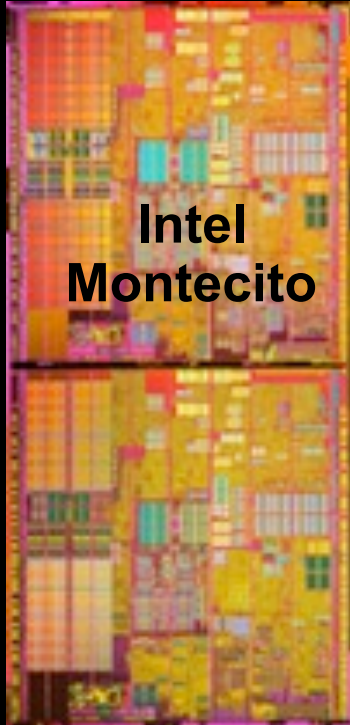
# Outline for the Introduction

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



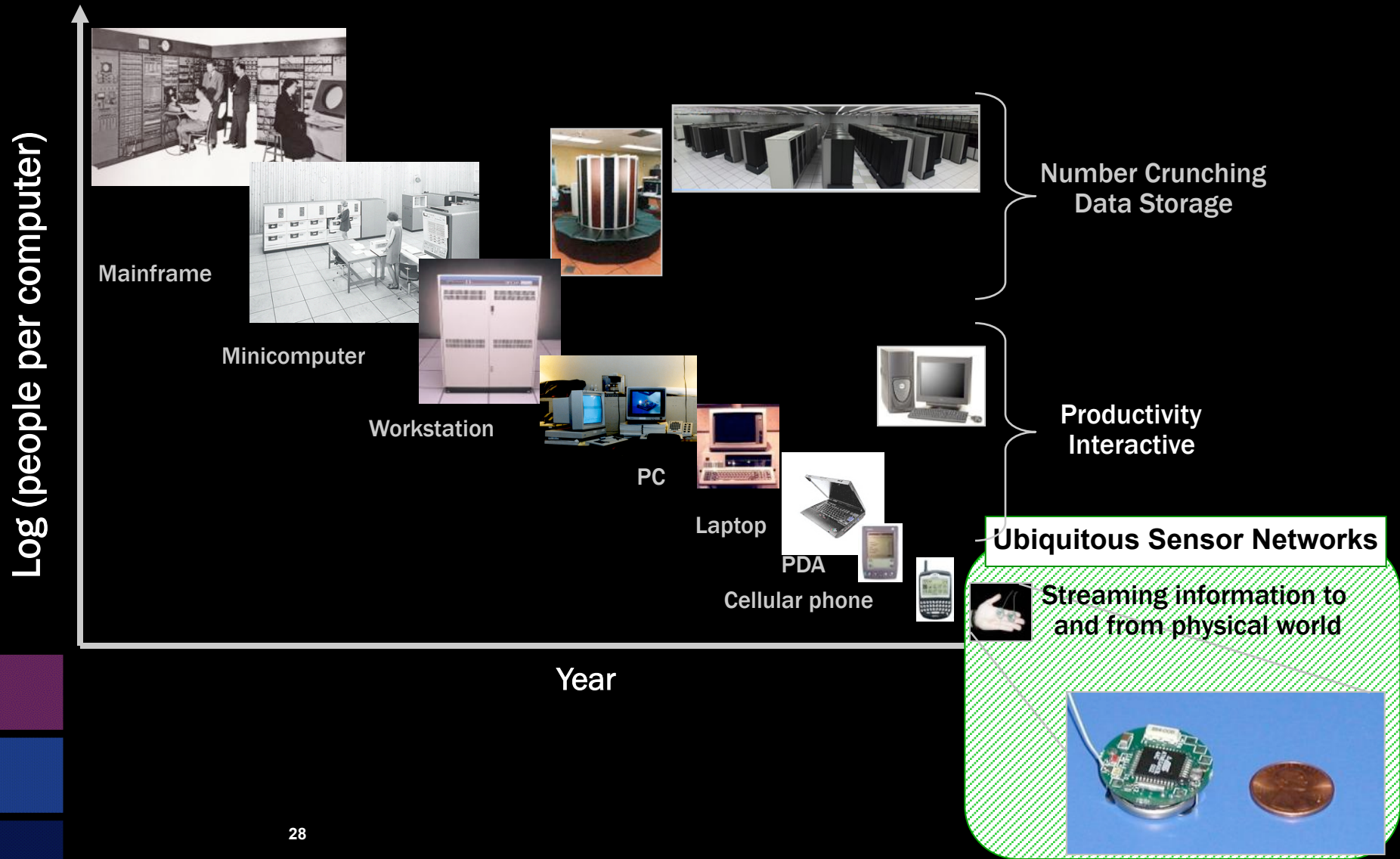
# Concurrency and Heterogeneity

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





# Challenge: The Physical Internet





# Exponentials Bound to Continue

[EE Times: Latest News](#)  
**Wireless is everywhere; ignore it at your peril**

[Bolaji Ojo](#)  
Page 1 of 2  
[EE Times](#)  
(01/07/2008 9:00 AM EST)

PRINT THIS STORY  
 SEND AS EMAIL  
 REPRINTS

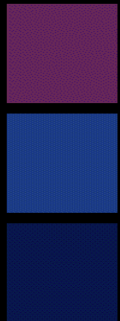
The search is over for the next killer app. It is wireless, it is all around you, and it will leave no sector of the global economy untouched.

EE Times,  
January 07, 2008

- 5 Billion people to be connected by 2015 (Source: NSN)
- The emergence of Web2.0
  - The “always connected” community network
- 7 trillion wireless devices serving 7 billion people in 2017 (Source: WirelessWorldResearchForum (WWRF))
  - 1000 wireless devices per person?



# The Emerging IT Scene





# The Technology Gradient: Computation



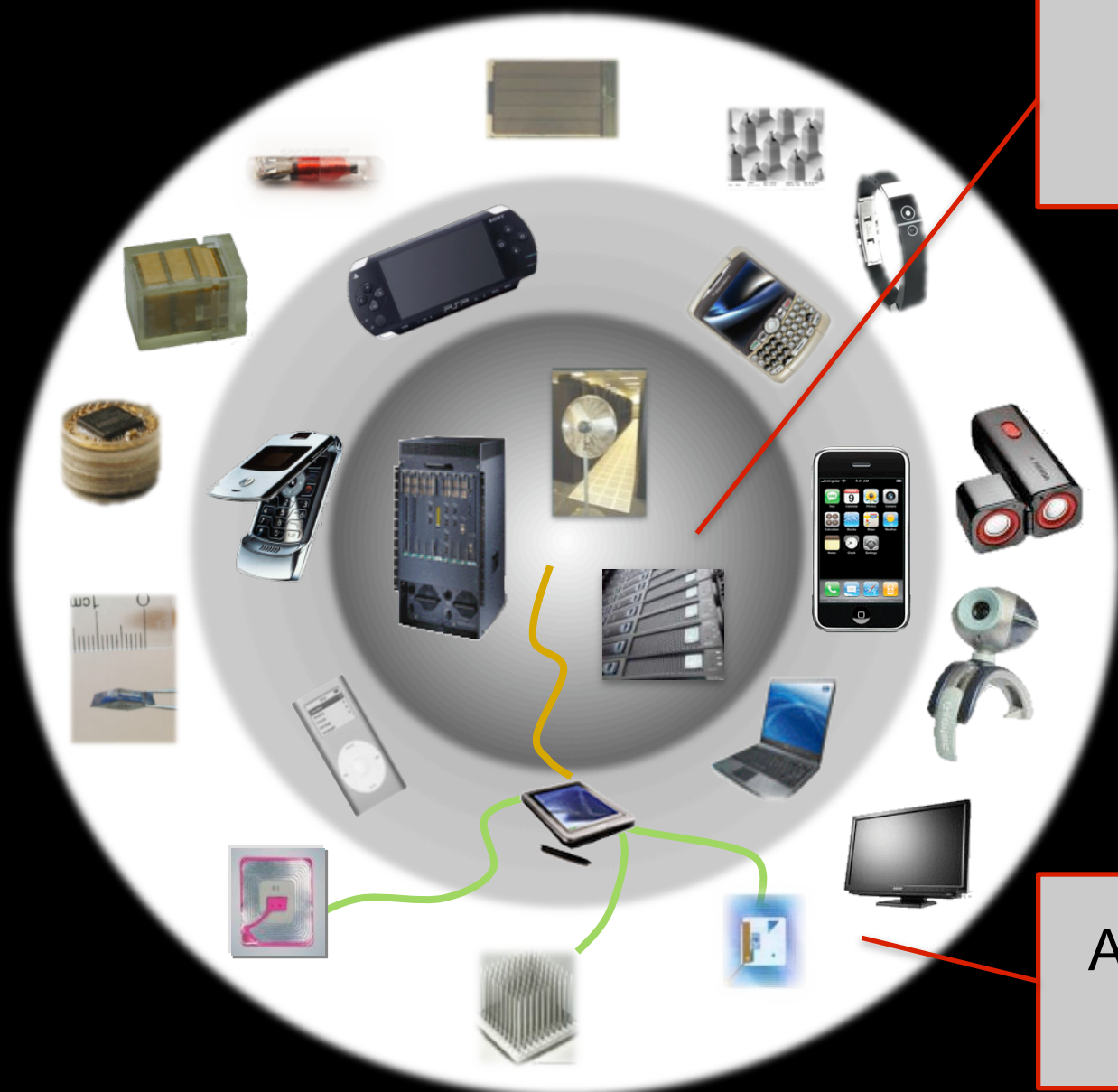
Driven by Moore's Law

Driven by "More Than Moore" and "Beyond Moore"





# The Technology Gradient: Communication



Mostly wired

Almost uniquely wireless







# Challenge: Power

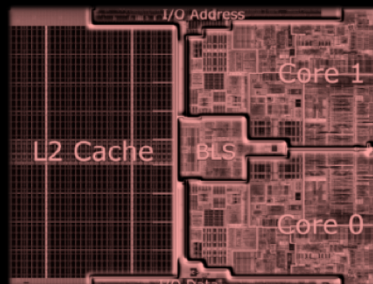
Energy = upper bound on the amount of available computation

- Total Energy of Milky Way Galaxy:  $10^{59}$  J
- Minimum switching energy for digital gate (1 electron@100 mV):  $1.6 \cdot 10^{-20}$  J (limited by thermal noise)
- Upper bound on number of digital operations:  $6 \cdot 10^{78}$
- Operations/year performed by 1 billion 100 MOPS computers:  $3 \cdot 10^{24}$
- Energy consumed in 180 years assuming a doubling of computational requirements every year.



# Challenge: Parallel Architectures

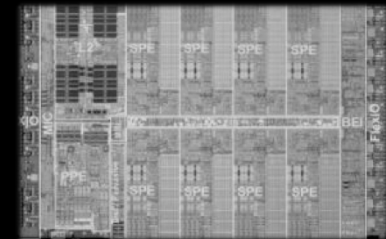
Scaling enabled integration of complex systems with hundreds of millions of devices on a single die



Intel KEROM dual core  
ISSCC 07, 290M trans.



SUN Niagara-2  
ISSCC 07, 500M trans.



IBM/Sony Cell  
ISSCC 05, 235M trans.



# Challenge: Design Chain Integration

## Automotive Industry

### Automakers



2005 Revenue  
\$1.1T

CAGR 2.8%  
(2004-2010)

### Tier 1 Suppliers



90%+ of  
revenue from  
automotive

2004 Revenue  
~\$200B

CAGR 5.4%  
(2004-2010)

### IC Vendors



~15% of  
revenue from  
automotive

2005 revenue  
\$17.4B

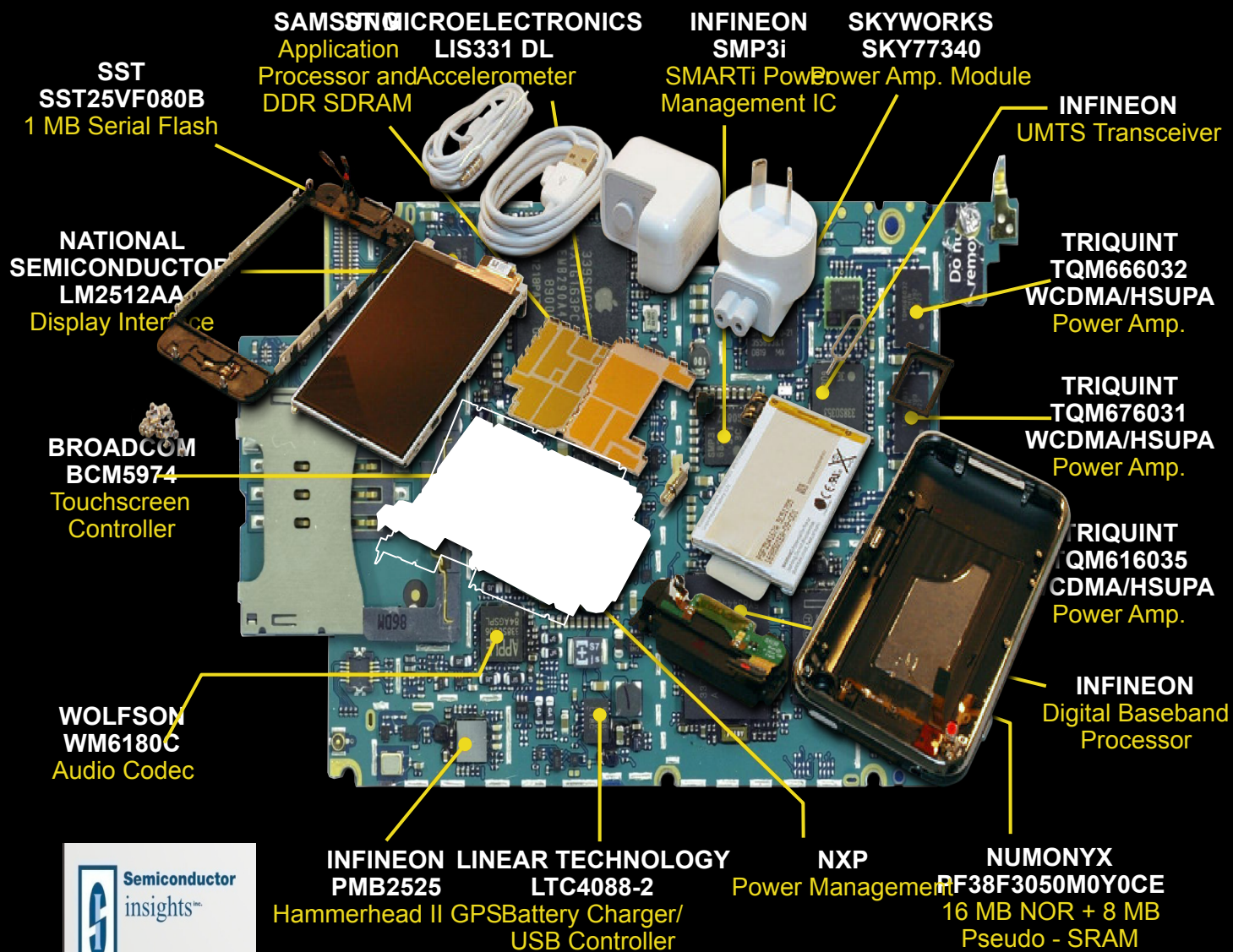
CAGR 10%  
(2004-2010)

Source: Public financials, Gartner 2005

EE249Fall09

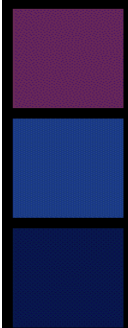
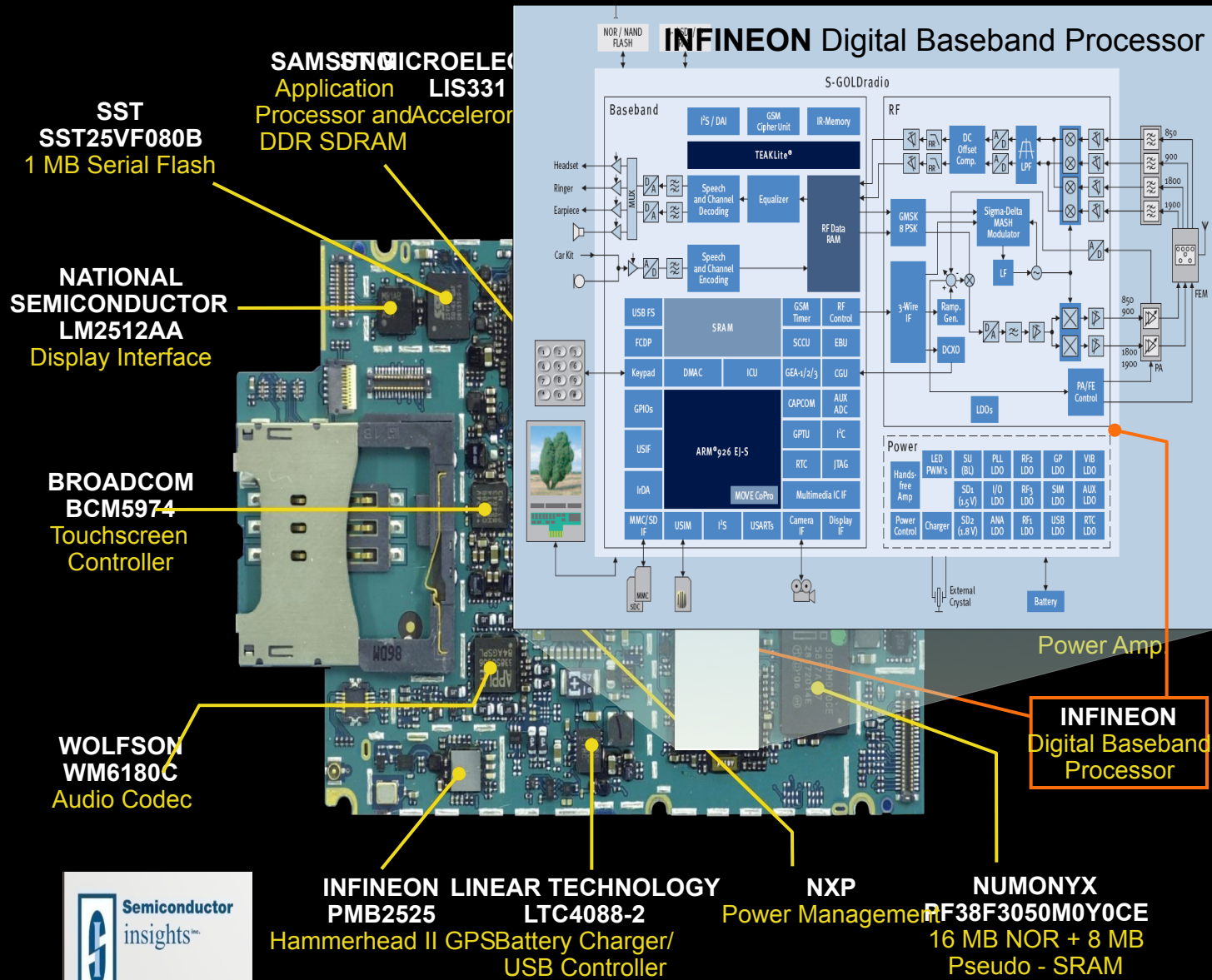


# Collaborating to Create the iPhone





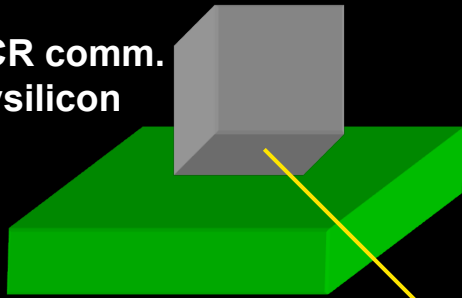
# Collaborating to Create the iPhone



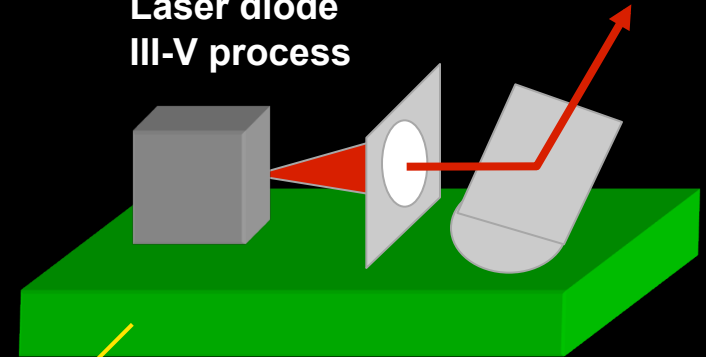


# Smart Dust

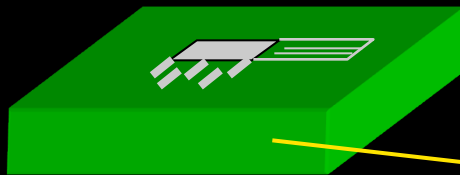
Passive CCR comm.  
MEMS/polysilicon



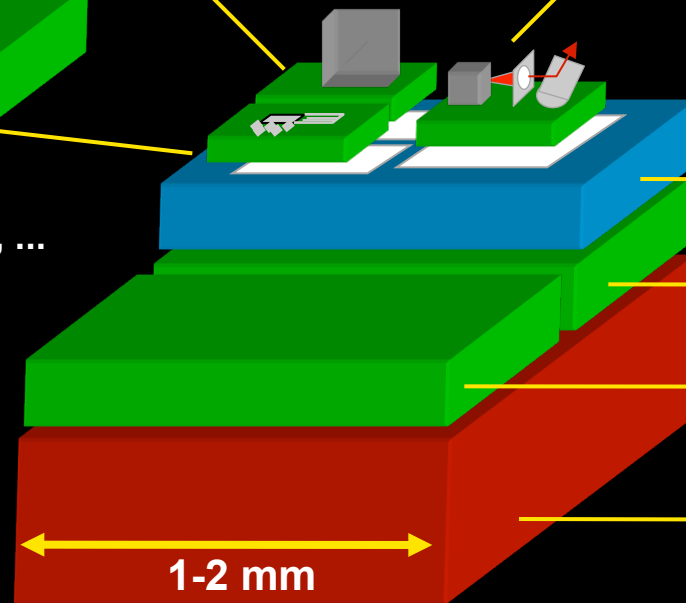
Laser diode  
III-V process



Active beam steering laser comm.  
MEMS/optical quality polysilicon



Sensor  
MEMS/bulk, surface, ...



Analog I/O, DSP, Control  
COTS CMOS

Power capacitor  
Multi-layer ceramic

Solar cell  
CMOS or III-V

Thick film battery  
Sol/gel  $V_2O_5$

1-2 mm

Source: K. Pister, Berkeley

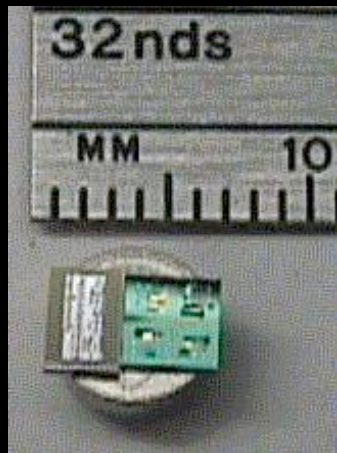
EE249Fall09



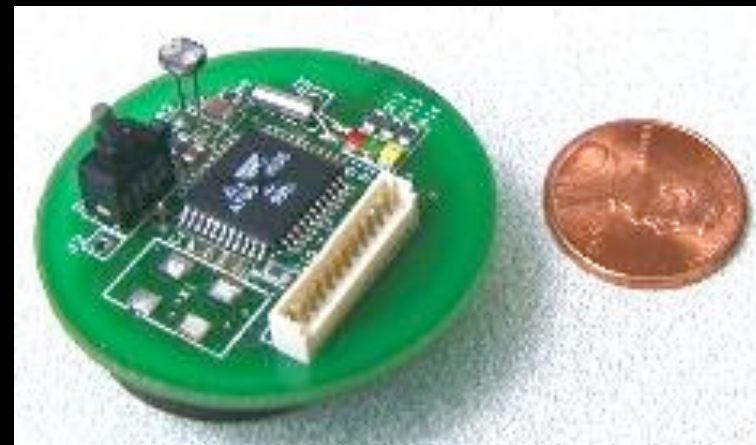
# Wireless Sensor Networks

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<sup>1</sup>



Berkeley Mote<sup>1</sup>



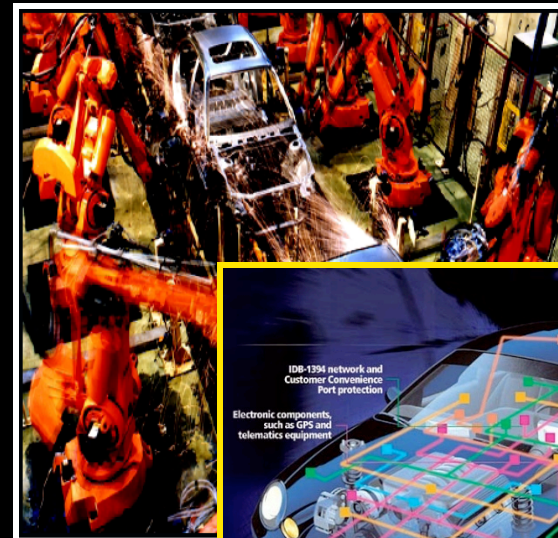
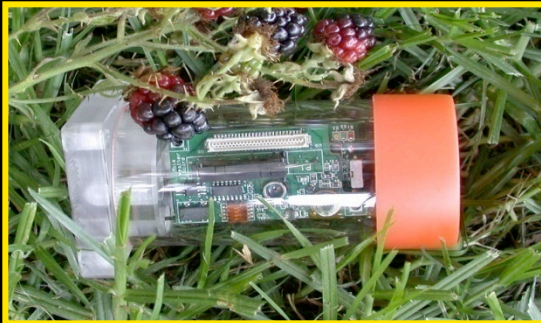
<sup>1</sup>From Pister et al., *Berkeley Smart Dust Project*



# Creating a Whole New World of Applications

From Monitoring

To Automation



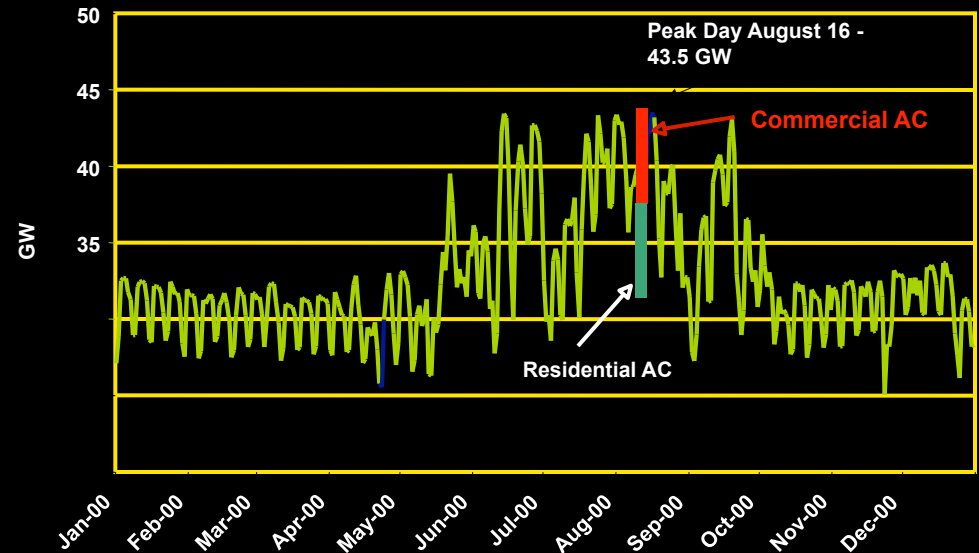




# Energy Management and Conservation

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

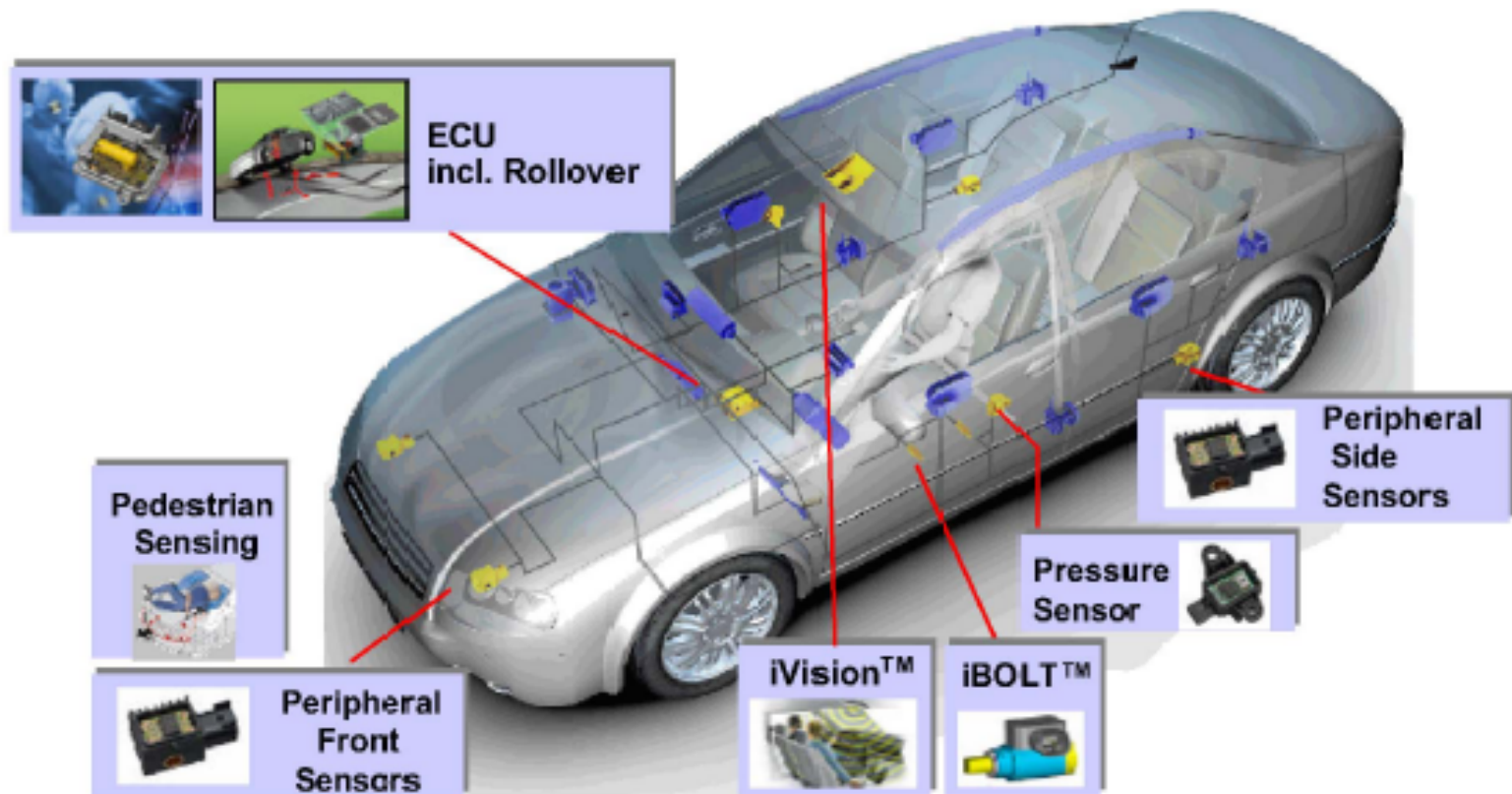
Cal ISO Daily Peak Loads  
January 1, 2000 - December 31, 2000



- 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



# Occupant Safety Systems Portfolio



Automotive Electronics

14

Confidential | AE/EE | 07/04/2005 | © Robert Bosch GmbH reserves all rights, even in the event of industrial property rights. We reserve all rights of disposal such as copying and passing on to third parties.

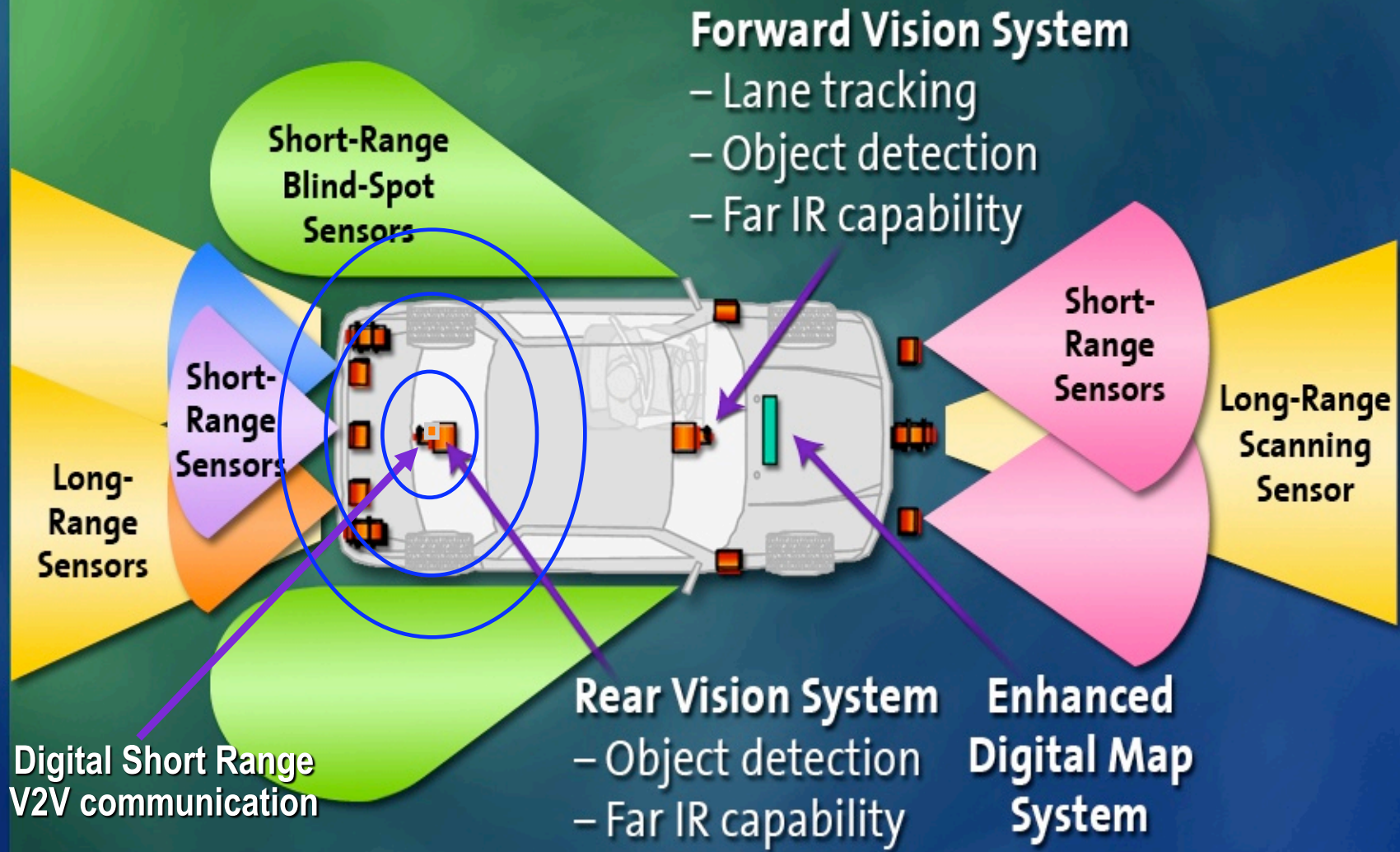


**BOSCH**



# 360° Safety with Integrated Sensor Strategy

## The refuse-to-rotate car!





# The Tire of the Future

New materials: enhanced performances, reduced rolling resistance, lower noise, reduced puncture risk, nanotechnologies, new compounds, new tread design, “self sealing” technologies.

New design technologies: virtual engineering for reducing time to market & engineering costs.

New electronics technologies inside the tire: pressure monitoring, friction, slip, tire consumption, contact force, “health” check-up information extraction & transmission....

**The Tire as an Intelligent Sensor!**



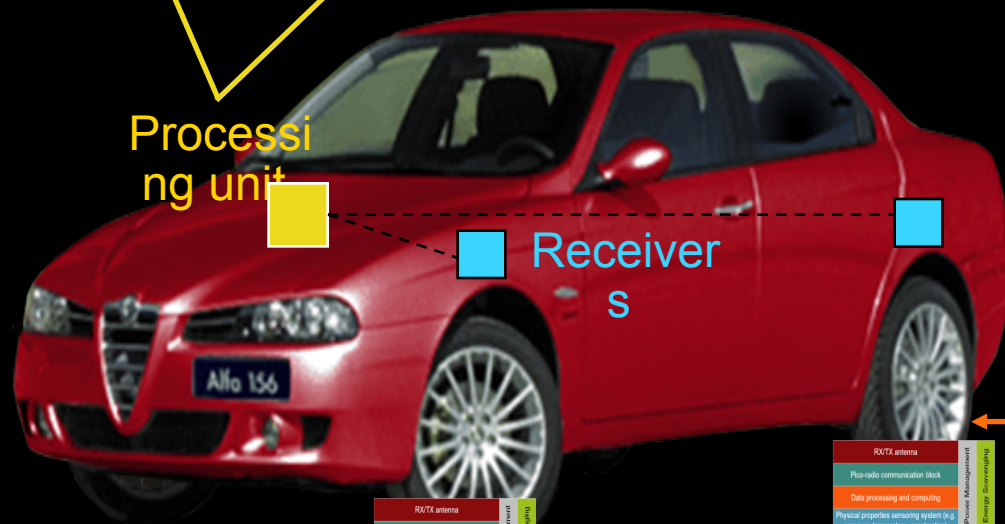
# Cyber™ Tyre Intelligent Tire System

Vehicle dynamics control system

User Applications

Processing unit

Receivers

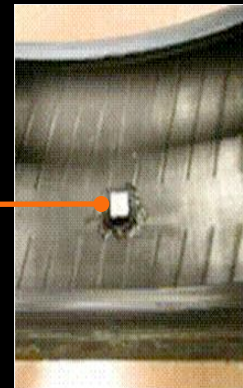


RX/TX antenna  
Pico-radio communication block  
Data processing and computing  
Physical properties sensing system (e.g. pressure, temperature, acceleration)

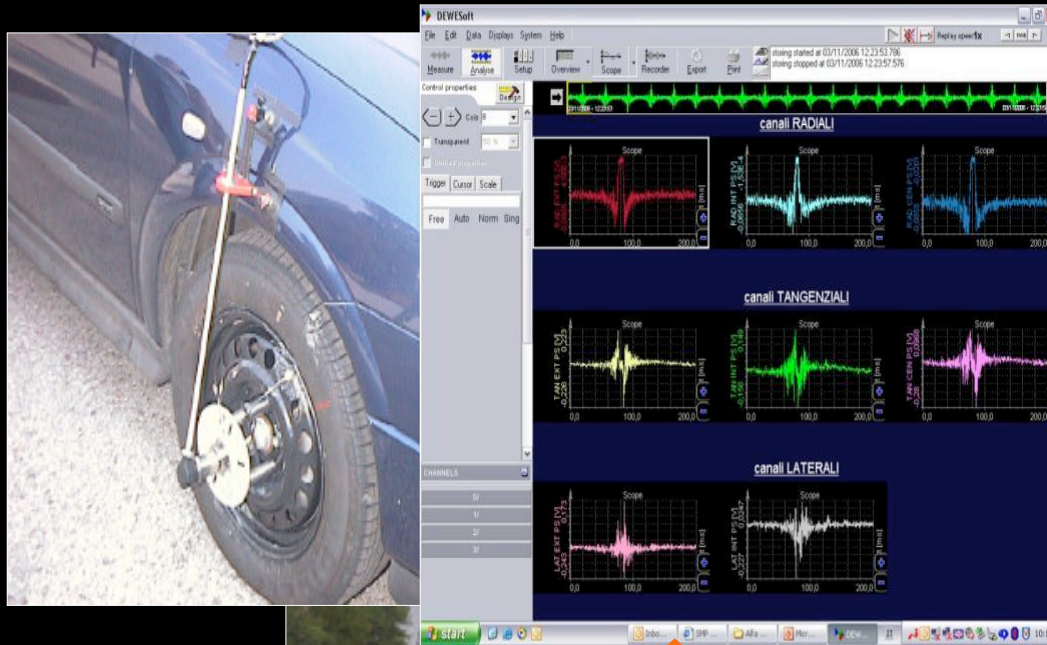
Cyber™Tyre

RX/TX antenna  
Pico-radio communication block  
Data processing and computing  
Physical properties sensing system (e.g. pressure, temperature, acceleration)

Cyber™Tyre



# Experimental Tests



Tyre inside

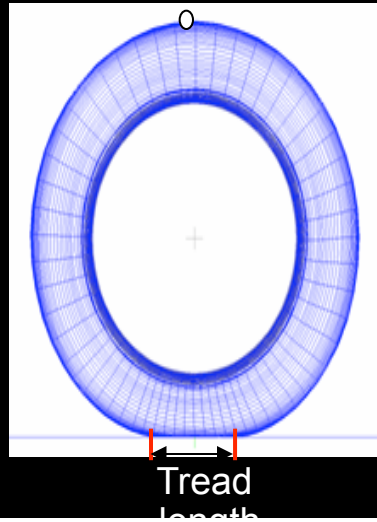
Accelerometers



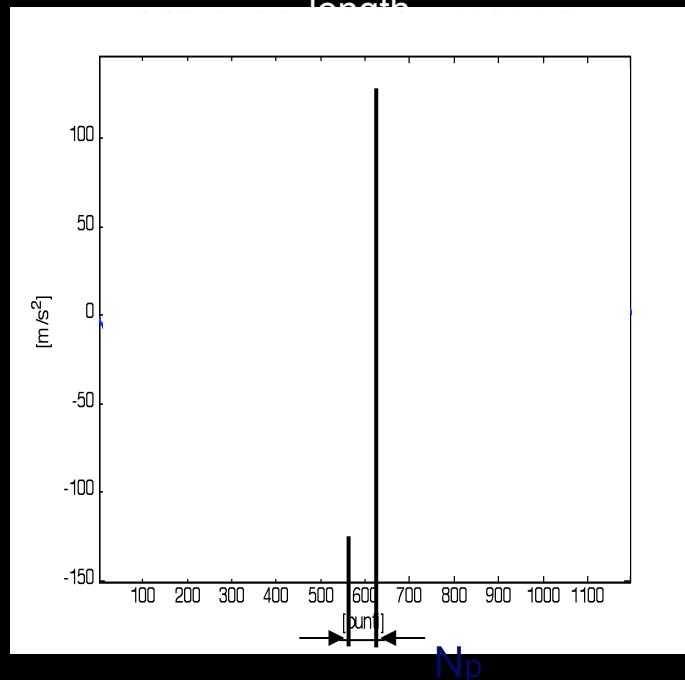
## Wide database

- Different tires
- Different sensor positioning
- Different speeds
- Different tracks
  - Steering pad
  - Straight line
  - Braking
  - Acceleration
  - ...
- Different conditions
  - Dry
  - Wet
  - Ice

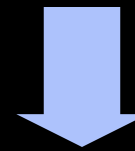
# Tread Length Estimation



- Minimum of the tangential component signal: tread area entry
- Maximum of the tangential component signal: tread area exit



$$PL = N_p / f_c \cdot \omega \cdot R_{rot}$$



PL : tread length

$R_{rot}$  : rolling radius

$\omega$  : angular speed

$f_c$  : sampling rate



# Cyber™ Tyre Development Partners

**Politecnico di Milano**  
Feature Extraction  
Kinematics pre-conditioner

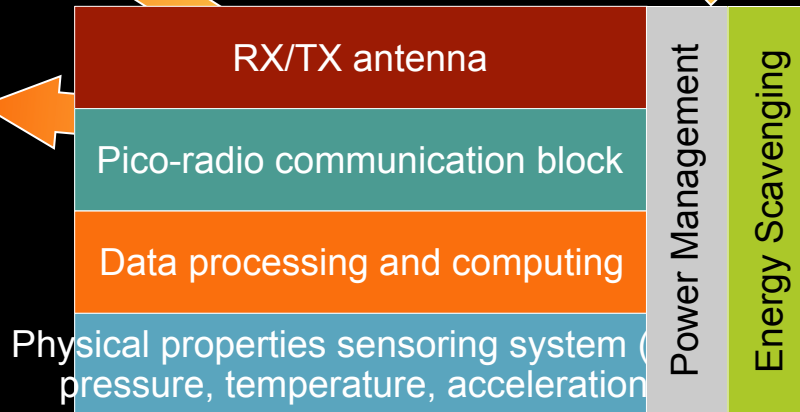
**Politecnico di Torino**  
Prototype Verification, Integration,  
Engineering Support

**Valtronic Technologies**

**UMC**  
IP and chip manufacturing

assembly, SA packaging technologies

**University of California, Berkeley**  
Ultra low power radio



**Encrea**

Breakthrough energy supply and power management technologies

Advanced new communication protocols

**Accent S.p.A.**  
acquisition, processing and advanced architectural technologies





# Industrial Plants

## Monitoring:

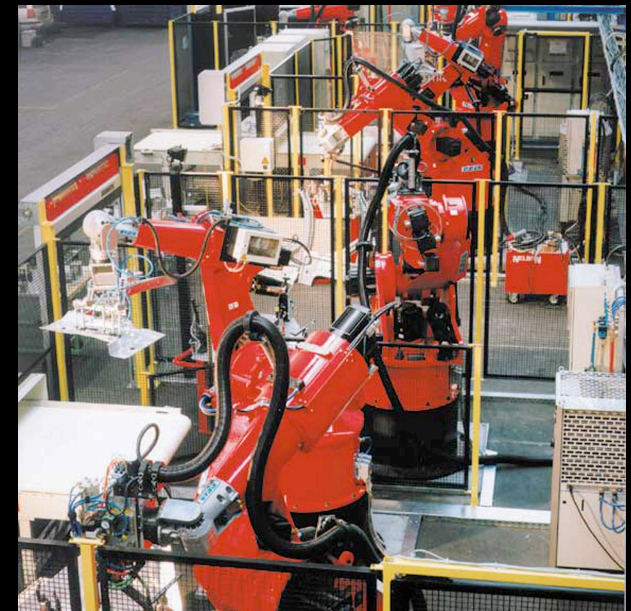
Vibrations, Temperature,  
Humidity, Position, Logistics

**Current solution:  
Wired Infrastructure**

**Future solution:  
WIRELESS**

## Wireless advantages:

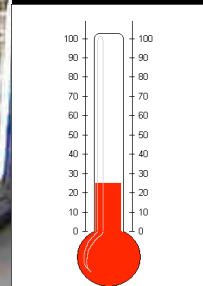
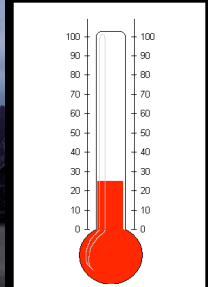
Reduce cabling  
Enhance flexibility  
Easy to deploy  
Higher safety  
Decreased maintenance costs





# Temperature Tracking

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





# Preventative Maintenance Program on Oil Tankers

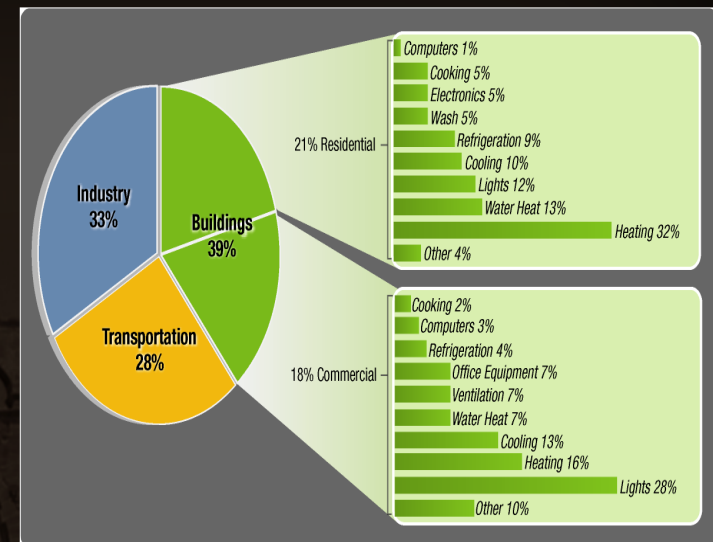
- The task:
  - Engine monitoring is critical for both keeping the ship operational and complying with insurance policy.
- Old Methods
  - Manually record vibration profile with data loggers.
  - Post process data for engine health and diagnostics.



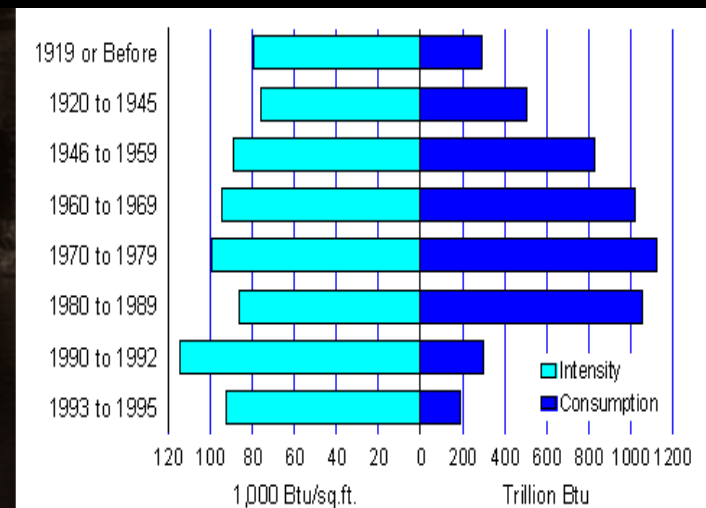
# Building Energy Demand Challenge

- Buildings consume
  - 39% of total U.S. energy
  - 71% of U.S. electricity
  - 54% of U.S. natural gas
- Buildings produce 48% of U.S. carbon emissions
- Commercial building annual energy bill: \$120 billion
- The only energy end-use sector showing growth in energy intensity
  - 17% growth 1985 - 2000
  - 1.7% growth projected through 2025

## Energy Breakdown by Sector



## Energy Intensity by Year Constructed



Energy Information Administration  
1995 Commercial Buildings Energy Consumption Survey

Sources: Ryan and Nicholls 2004, USGBC, U

# Systems of Systems Approach to Energy Efficiency

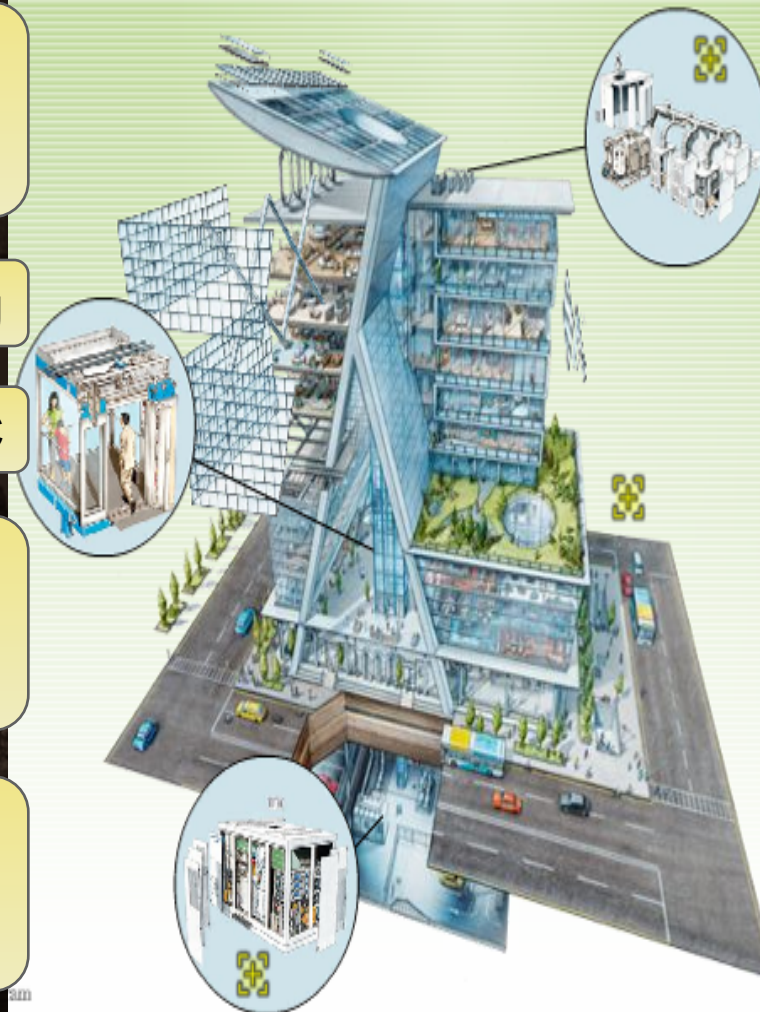
Buildings Design  
Energy and Economic  
Analysis

Windows and Lighting

HVAC

Domestic/International  
Policies, Regulation,  
Standards, Markets

Demonstrations,  
Benchmarking, Operations  
and Maintenance



Natural Ventilation,  
Indoor Environment

Networks,  
Communications,  
Performance Database

Sensors, Controls,  
Performance Metrics

Power Delivery and  
Demand Response

Building Materials,  
Misc. Equipment

**Integration: *The Whole is Greater than the Sum of the Parts***

# Building Systems Integration Challenges

## Complex\* interconnections among building components

- HETEROGENEITY

- Components do not necessarily have mathematically similar structures
- May involve different scales in time or space

- SIZE

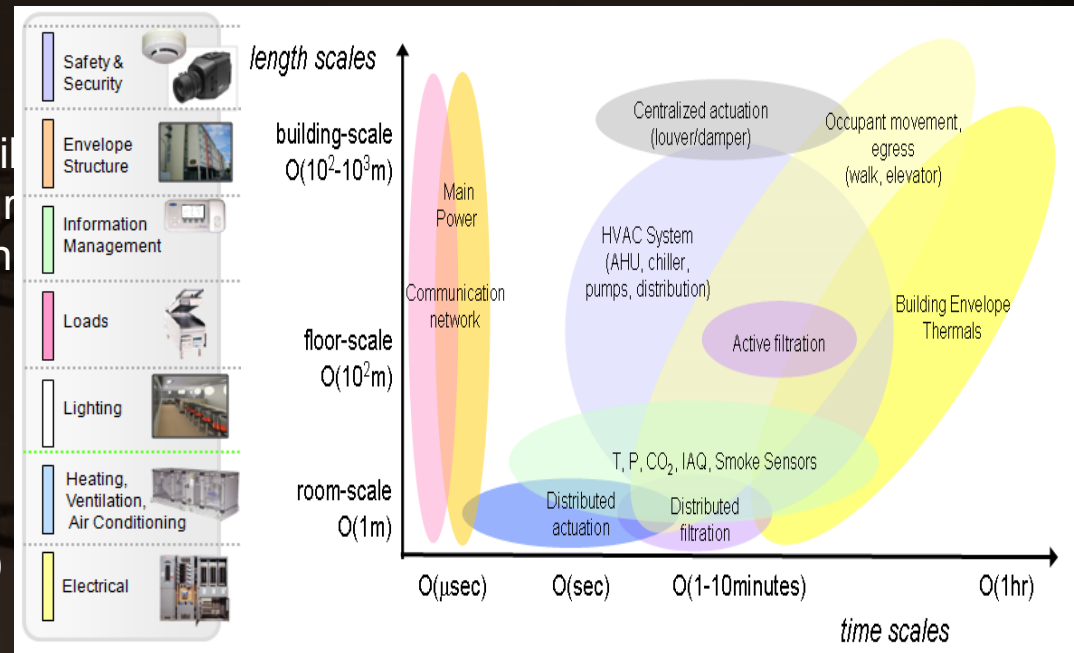
- The number of components may be large/enormous

- DISTRIBUTED NETWORKED SYSTEMS

- Components can be connected in a variety of ways, most often nonlinearly and/or via a network
- Local and system wide phenomena may depend on each other in complicated ways

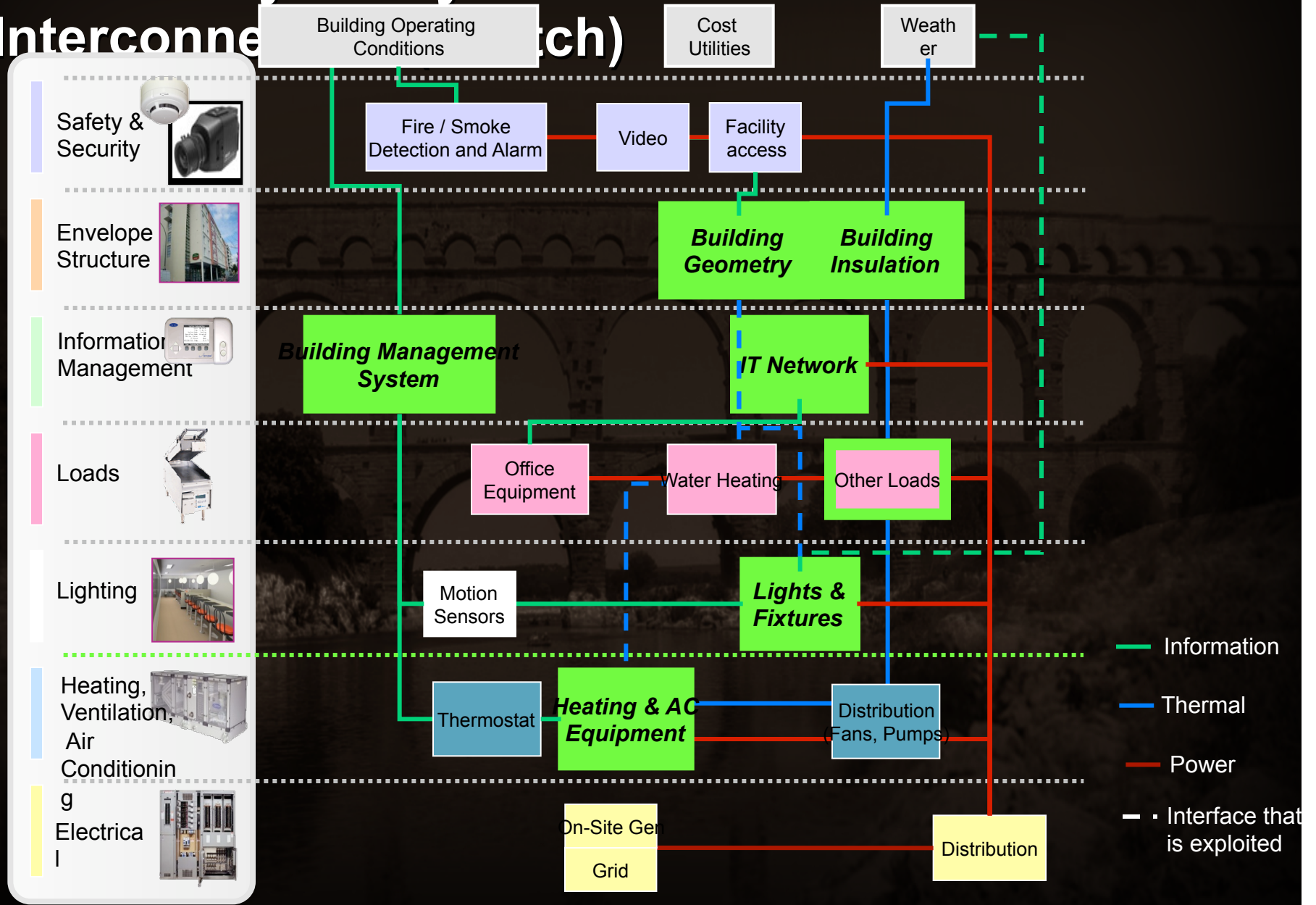
- EMERGING BEHAVIOR IN COMPOSITION

- Overall system behavior can be difficult to predict from the behavior of individual components. May evolve along qualitatively different pathways that may display great sensitivity to small perturbations at any stage



\* D.L. Brown, J. Bell, D. Estep, W. Gropp, B. Hendrickson, S. Keller-McNulty, D. Keyes, J. T. Oden and L. Petzold, Applied Mathematics at the U.S. Department of Energy: Past, Present and a View to the Future, DOE Report, LLNL-TR-401536, May 2008.

# Full Facility Subsystems and their Interconnections (ch)



# Engineering Tomorrow's Designs

## Synthetic Biology

The creation of novel biological functions and tools by  
modifying  
or integrating well-characterized biological  
components into  
higher-order systems using mathematical modeling  
to direct  
the construction towards the desired end product.

*"Building life from the ground up" (Jay Keasling, UCB)*

Keynote presentation, World Congress on Industrial Biotechnology and  
Bioprocessing,  
March 2007.

### **Development of foundational technologies:**

Tools for hiding information and managing complexity

Core components that can be used in combination reliably



# Pioneering Synthetic Biology

## ENGINEERING LIFE: Building a FAB for Biology

BY THE BIO FAB GROUP\*

\*David Baker, George Church, Jim Collins,  
Drew Endy, Joseph Jacobson, Jay Keasling,  
Paul Modrich, Christina Smolke and Ron Weiss

Principles and practices learned  
from engineering successes can  
help transform biotechnology  
from a specialized craft into  
a mature industry

### Moving from ad-hoc to structured design

[Reference: Scientific American, June 2006]



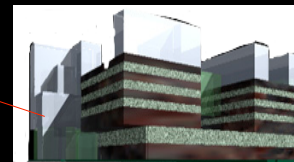
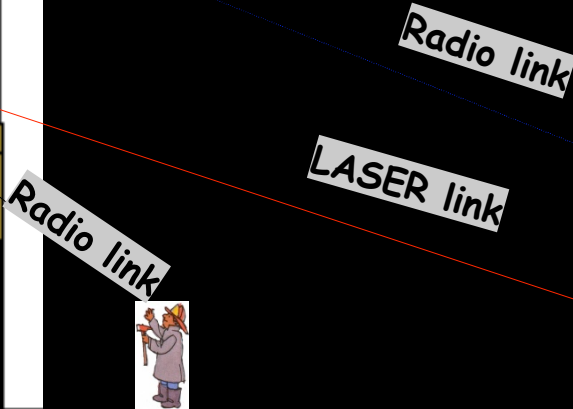
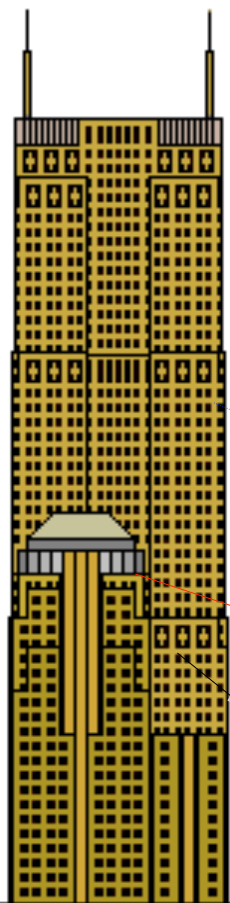
# Applications

## 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





# 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
  - .....

**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

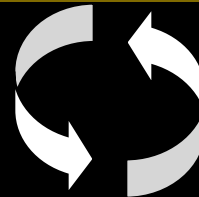


# Supply Chain: Design Roles-> Methodology->Tools

*Design Roles*



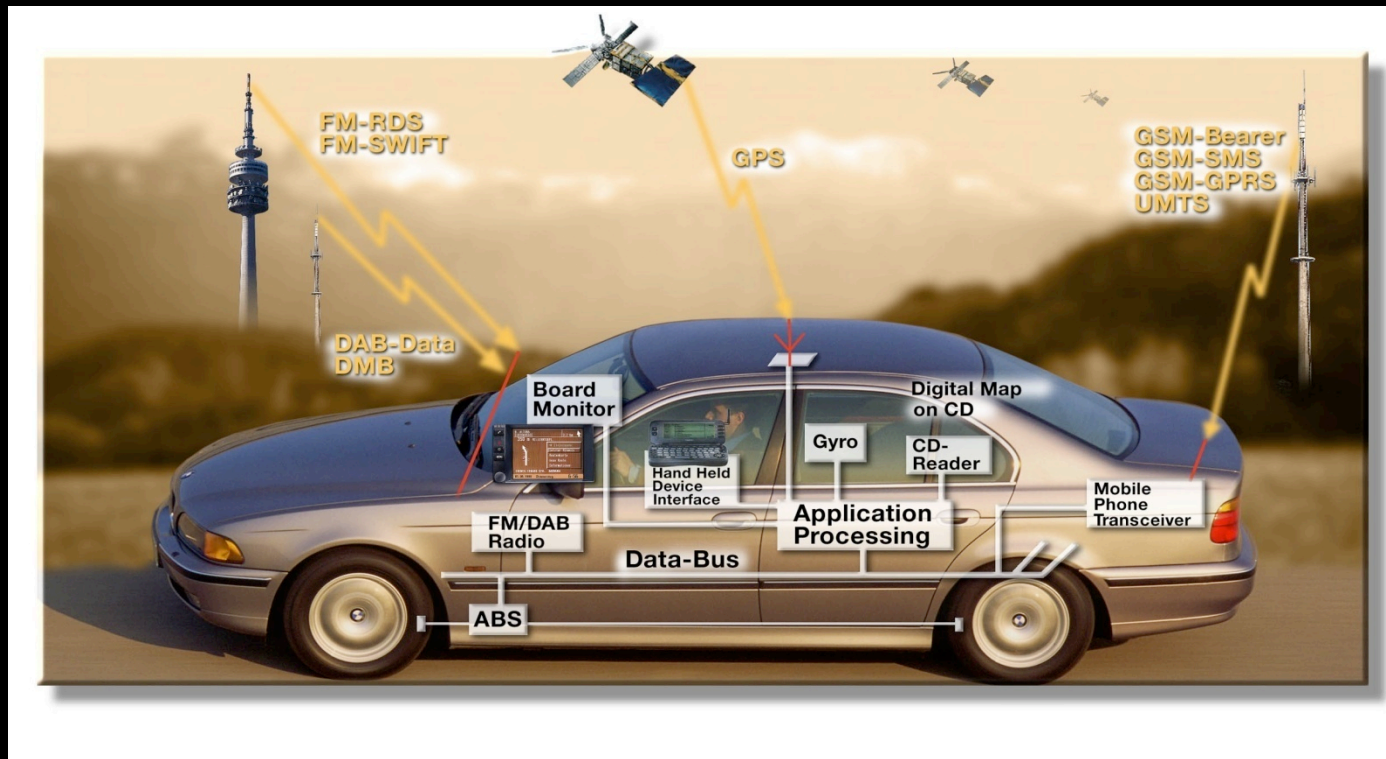
*Methodology*



*Tools*



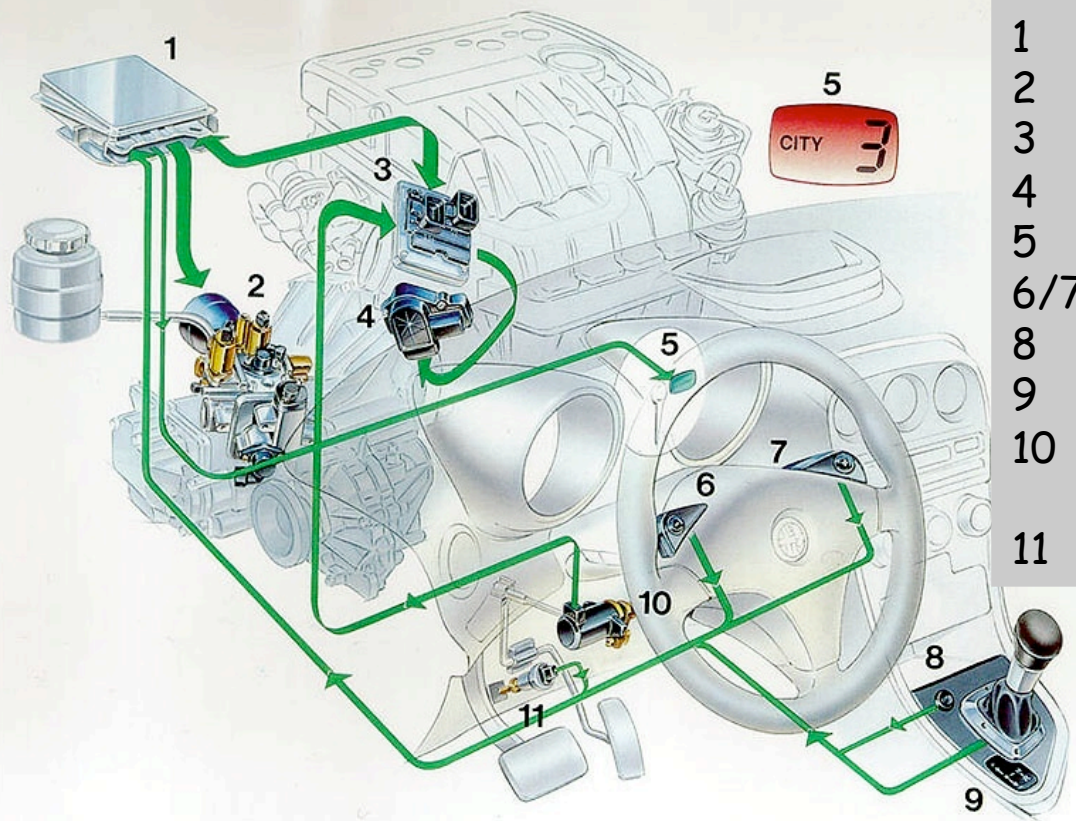
# Automotive Supply Chain: Car Manufacturers



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



# 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



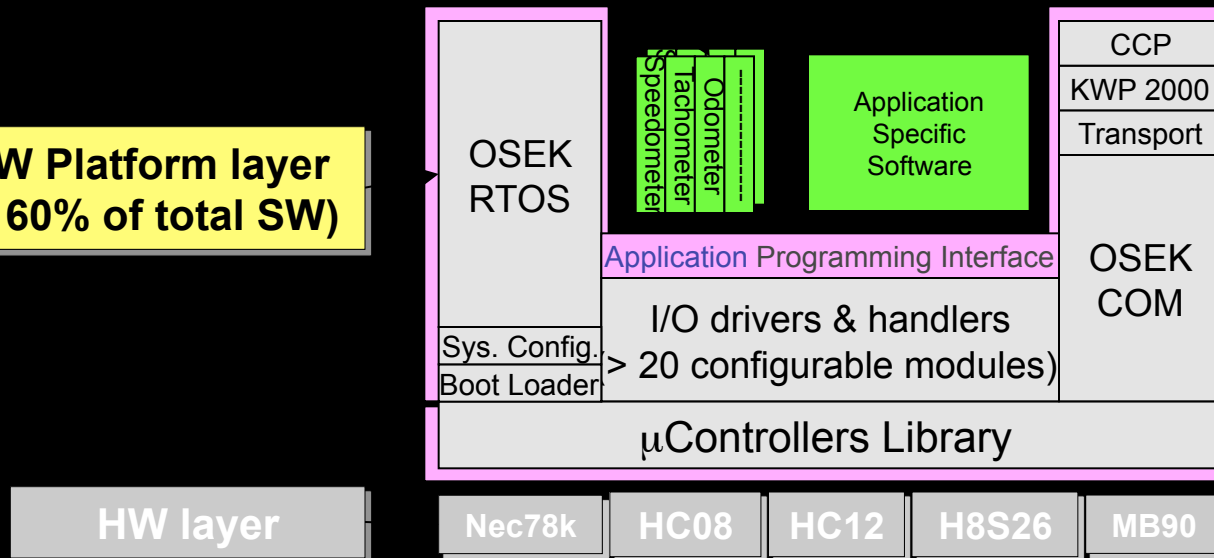
- Subsystem Partitioning
- Subsystem Integration
- Software Design: Control Algorithms, Data Processing
- Physical Implementation and Production



# Automotive Supply Chain: Subsystem Providers

Application Platform layer  
(≅ 10% of total SW)

SW Platform layer  
(> 60% of total SW)



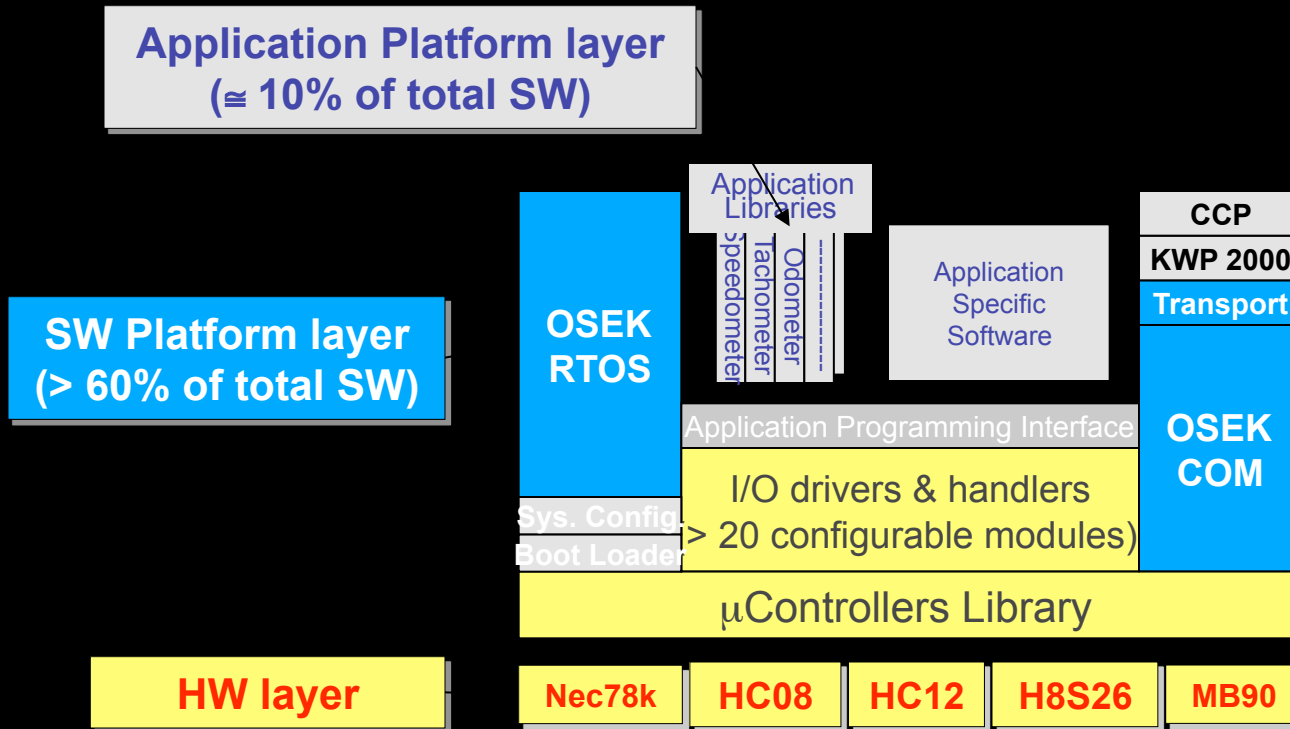
Platform Integration  
Software Design

“firmware” and “glue software”

“Application”



# Automotive Supply Chain: Platform & IP Providers



- “Software” platform
- “Hardware” platform

RTOS and communication layer

Hardware and IO drivers



# Outline for the Introduction

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



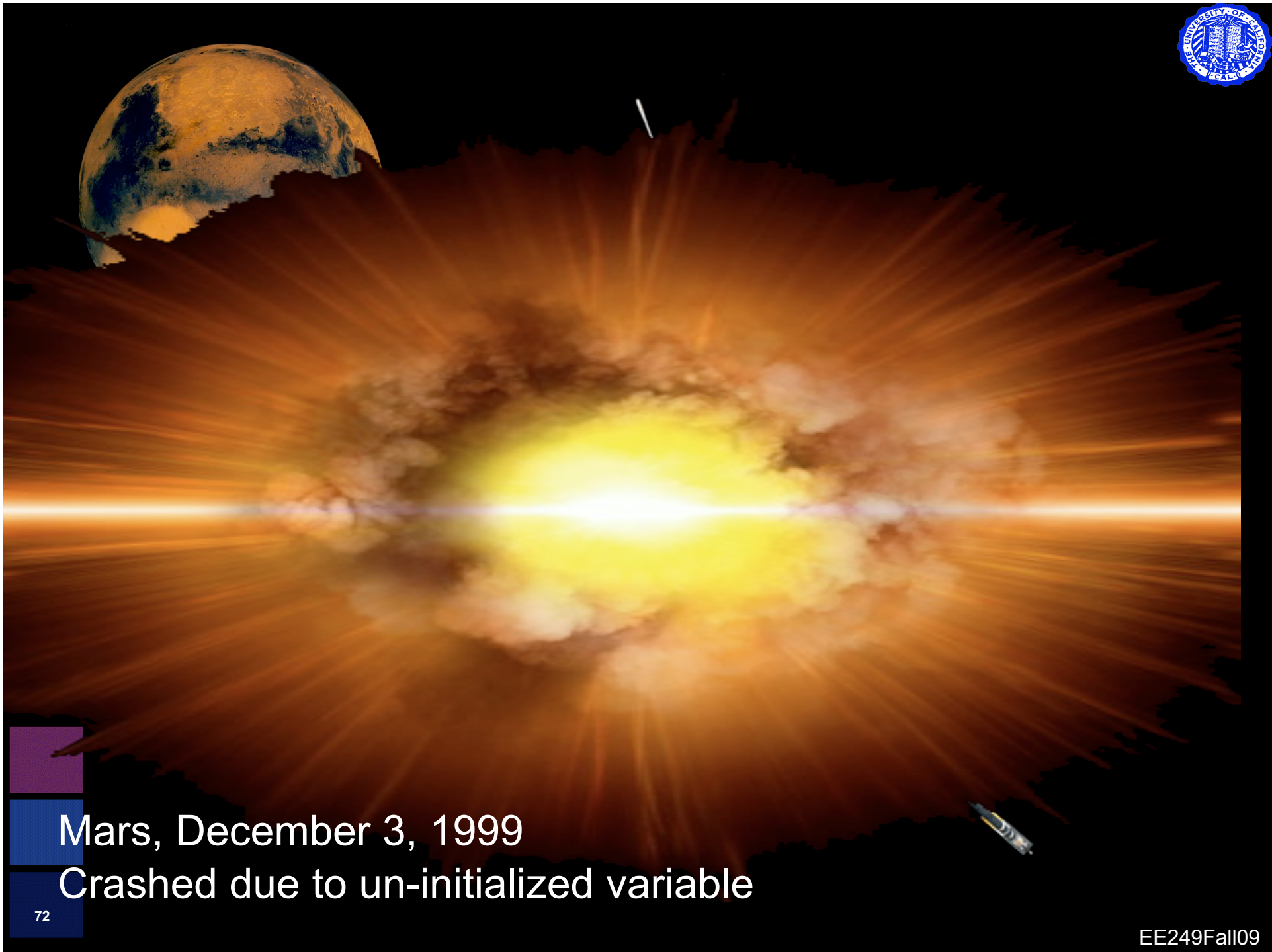
# How Safe is Our Real-Time Software?





# Computing for Embedded Systems





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


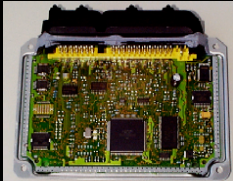






**\$4 billion development effort  
40-50% system integration & validation cost**



# Complexity, Quality, & Time To Market today

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

\* C++ CODE



Intelligence. Drives You Safe.

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

EE249Fall09



# Software Bugs Cost \$59.5 Billion a Year

INFOWORLD, JUNE 28, 2002 – BY PAUL KRILL

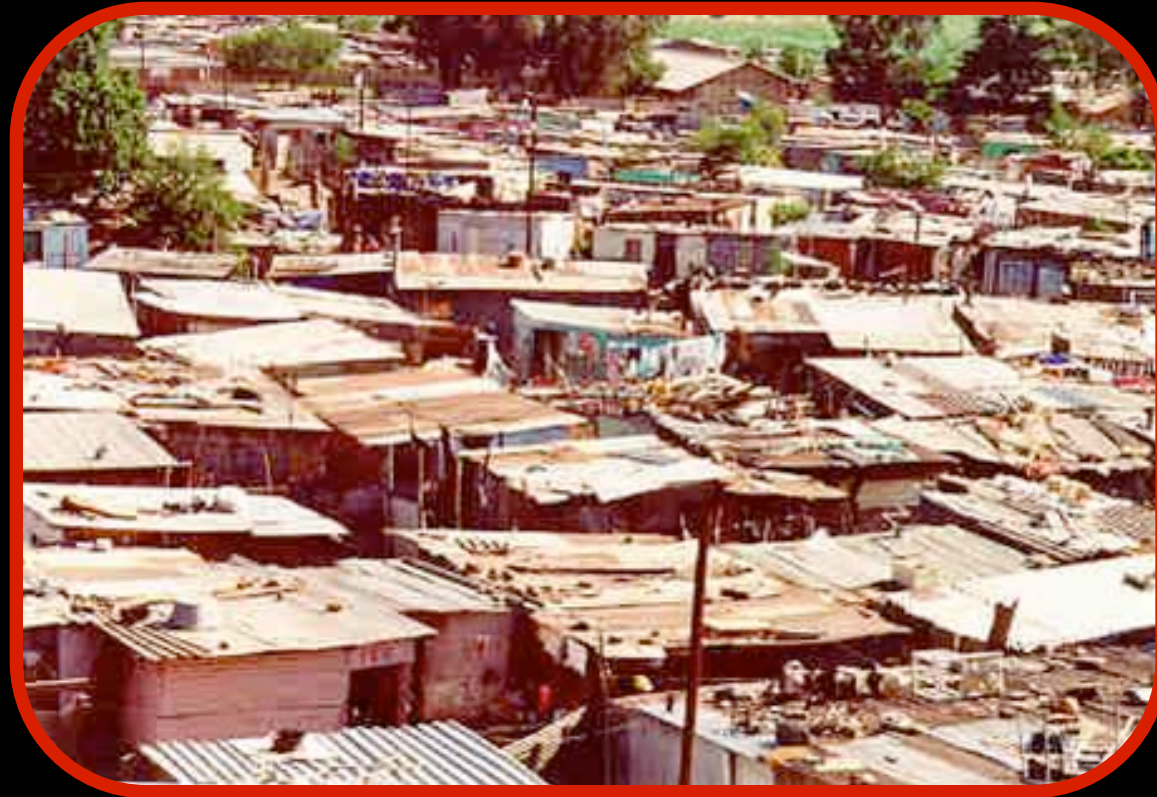
## Software bugs cost \$59.5 billion a year, study says

Software bugs cost the U.S. economy an estimated \$59.5 billion per year, or 0.6 percent of the gross domestic product, according to a newly released study by the U.S. Department of Commerce National Institute of Standards and Technology (NIST). In a statement released on Friday, NIST said more than half the costs are borne by software users and the remainder by software developers and vendors.

Additionally, the study found that although errors cannot be removed, more than a third of the costs, or an estimated \$22.2 billion, could be eliminated by improved testing that enables earlier and more effective identification and removal of defects. Currently, more than half of errors are not found until “downstream” in the development process or during post-sale use of software, according to NIST.



# Embedded Software Architecture Today





# We Live in an Imperfect World!

PAGE 14 – SUNDAY, FEBRUARY 6, 2005 – THE NEW YORK TIMES (by Tim Moran)

## What's Bugging the High-Tech Car?

On a hot summer trip to Cape Cod, the Mills family minivan did a peculiar thing. After an hour on the road, it began to bake the children. Mom and Dad were cool and comfortable up front, but heat was blasting into the rear of the van and it could not be turned off.

Fortunately for the Mills children, their father — W. Nathaniel Mills III, an expert on computer networking at I.B.M. — is persistent. When three dealership visits, days of waiting and the cumbersome replacement of mechanical parts failed to fix the problem, he took the van out and drove it until the oven fired up again. Then he rushed to the mechanic to look for a software error.

Additionally, the study found that although errors cannot be removed, more than a "It took two minutes for them to hook up their diagnostic tool and find the fault," said Mr. Mills, senior technical staff member at I.B.M.'s T.J. Watson Research Center in Hawthorne, N.Y. "I can almost see the software code; a sensor was bad."

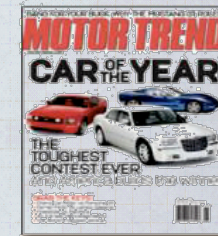
Indeed, the high-tech comfort of the confused the 2001 sending freezing loyal van up, third billion, c



## NHTSA To Probe Reports Of Sudden Engine Stalls In Prius Hybrids

The National Highway Traffic Safety Administration said yesterday it is investigating reports that a software problem can cause the engine of Toyota's Prius hybrid to stall without warning at highway speeds. No accidents have been reported thus far.

NHTSA has received 33 reports of stalling in Prius cars from model years 2004 and 2005, according to the agency's initial report. More than 85 percent of the cars that stalled did so at speeds between 35 and 65 miles per hour.





# How is Embedded Software Different from Ordinary Software?

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



# 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  $\infty$  registers





# 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*



# Examples of the Standard Trick

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

*Summary: abstract and hide complexity of resources*



# 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



# 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



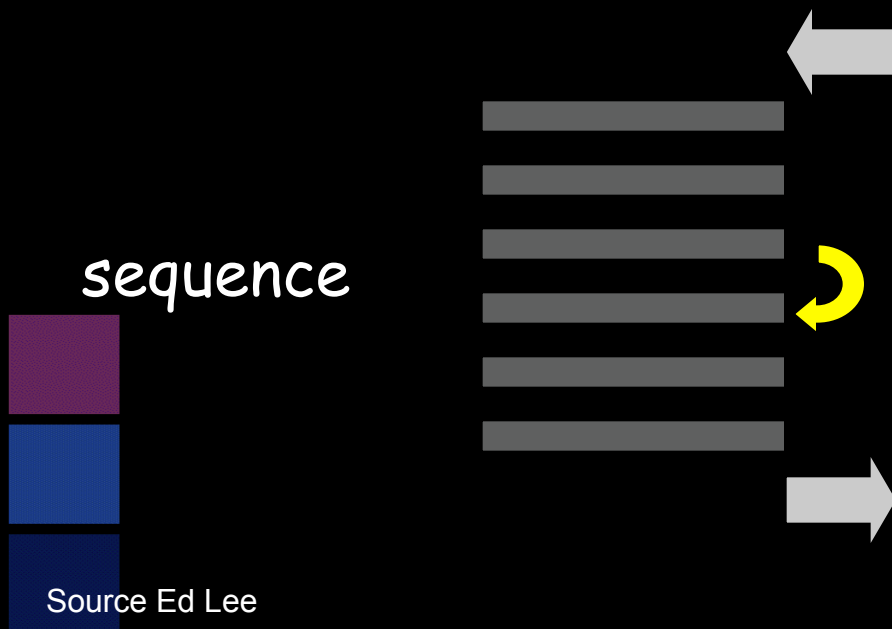
# Another Traditional Systems Science - Computation, Languages, and Semantics



Alan Turing

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

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

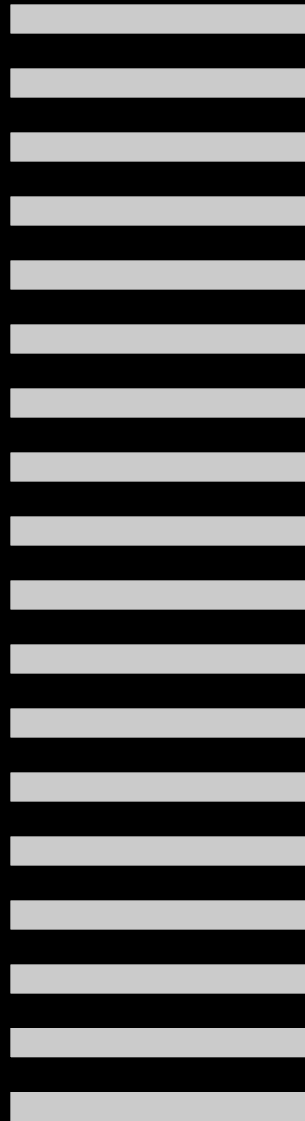




# Processes and Process Calculi

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

incoming message →



← outgoing message

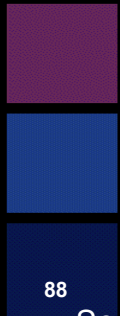
In prevailing software practice, processes are sequences of external interactions (total orders).

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.*

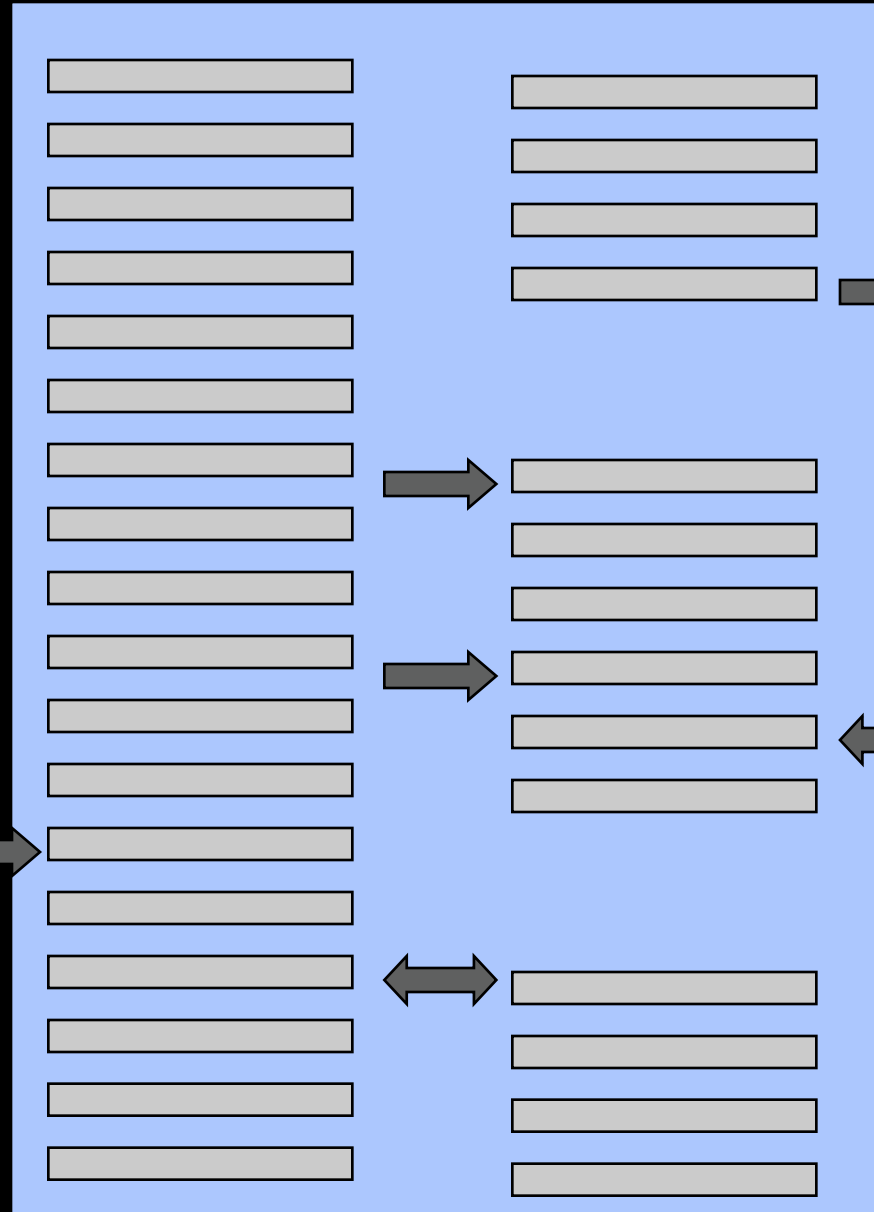




# Interacting Processes – Not Compositional

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.





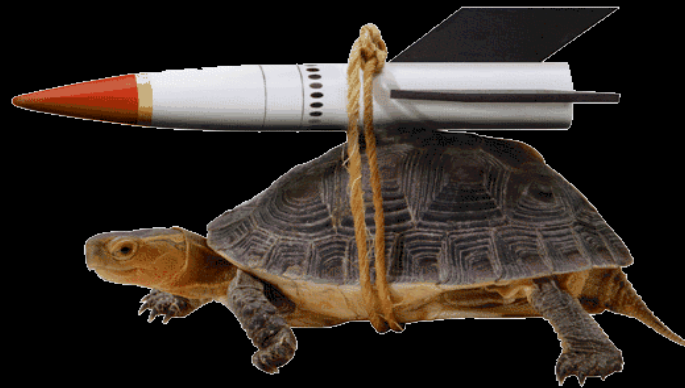
# Compositionality



Non-compositional formalisms lead to very awkward architectures.



# What About Real Time?



“Make it faster!”

## First Challenge on the Cyber Side: Real-Time and Power-aware Software

***Correct execution of a program in C, C#, Java, Haskell, etc. has nothing to do with how long it takes to do anything. All our computation and networking abstractions are built on this premise.***



Timing of programs is not repeatable, except at very coarse granularity.

Programmers have to step outside the programming abstractions to specify timing and power behavior.

## Second Challenge on the Cyber Side: Concurrency

Threads dominate concurrent software.

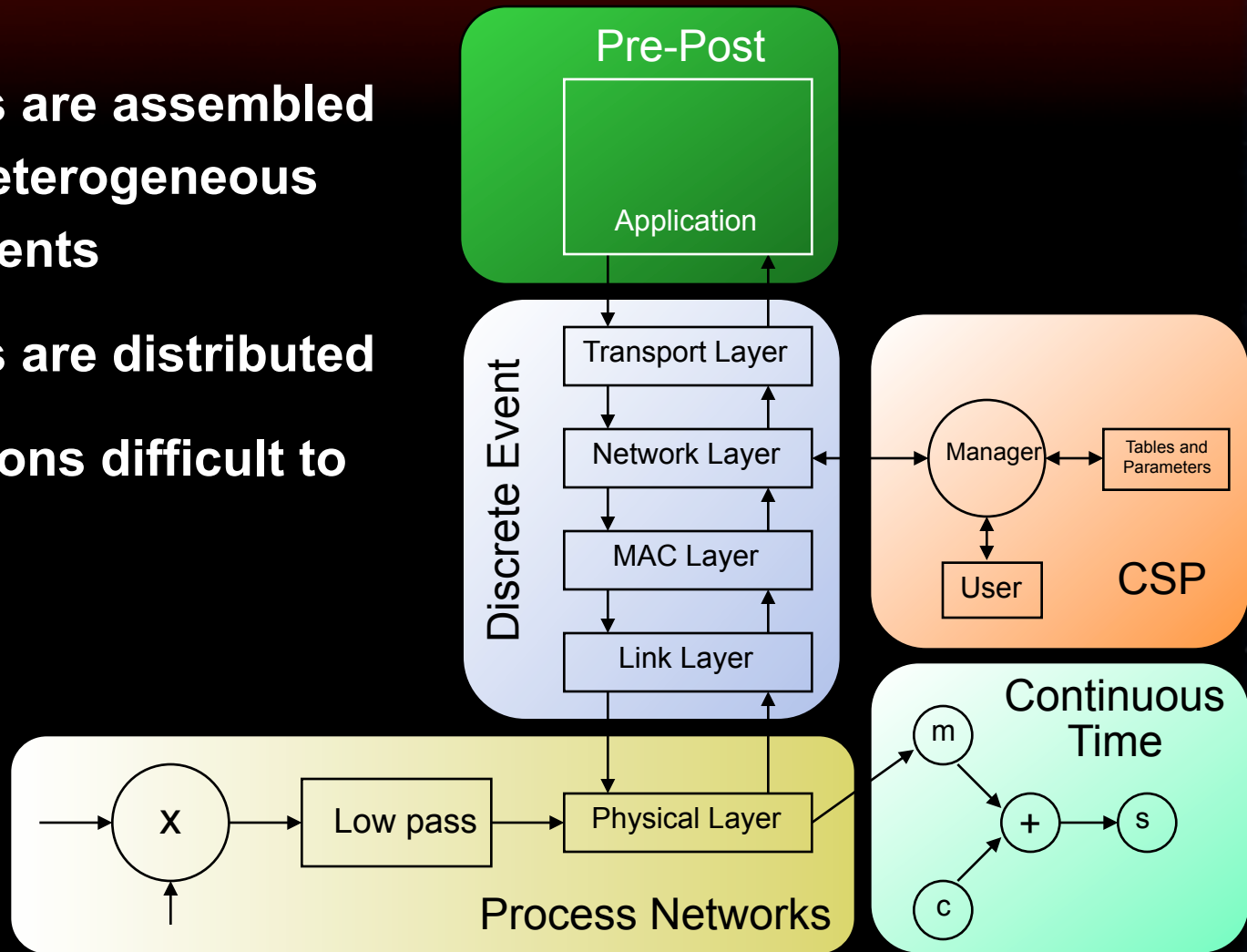
- **Threads**: Sequential computation with shared memory.
- **Interrupts**: Threads started by the hardware.

Incomprehensible interactions between threads are the sources of many problems:

- **Deadlock**
- **Priority inversion**
- **Scheduling anomalies**
- **Nondeterminism**
- **Buffer overruns**
- **System crashes**

# Common Features

- Systems are assembled out of heterogeneous components
- Systems are distributed
- Interactions difficult to define



# The Intellectual Agenda

To create a modern computational systems science and systems design practice with

- Concurrency
- Composability
- Time
- Hierarchy
- Heterogeneity
- Resource constraints
- Verifiability
- Understandability



# Chess: Center for Hybrid and Embedded Software Systems

## Principal Investigators

- Thomas Henzinger (EPFL)
- Edward A. Lee (Berkeley)
- Alberto Sangiovanni-Vincentelli (Berkeley)
- Shankar Sastry (Berkeley)
- Janos Sztipanovits (Vanderbilt)
- Claire Tomlin (Berkeley)



This center, founded in 2002, blends systems theorists and application domain experts with software technologists and computer scientists.

## Executive Director

- Christopher Brooks

## Associated Faculty

- David Auslander (Berkeley, ME)
- Ahmad Bahai (Berkeley)
- Ruzena Bajcsy (Berkeley)
- Gautam Biswas (Vanderbilt)
- Ras Bodik (Berkeley, CS)
- Bella Bollobas (Memphis)
- Karl Hedrick (Berkeley, ME)
- Gabor Karsai (Vanderbilt)
- Kurt Keutzer (Berkeley)
- George Nacula (Berkeley, CS)
- Koushik Sen (Berkeley, CS)
- Sanjit Seshia (Berkeley)
- Jonathan Sprinkle (Arizona)
- Masayoshi Tomizuka (Berkeley, ME)
- Pravin Varaiya (Berkeley)

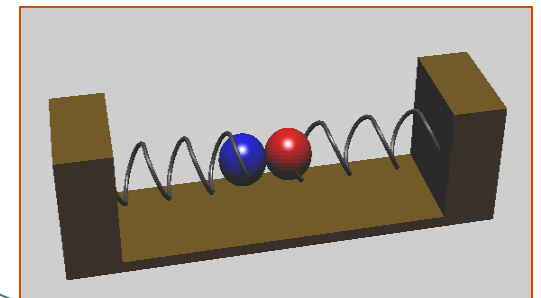
## *the Berkeley directors of Chess*

### Some Research Projects

- Precision-timed (PRET) machines
- Distributed real-time computing
- Systems of systems
- Theoretical foundations of CPS
- Hybrid systems
- Design technologies
- Verification
- Intelligent control
- Modeling and simulation

## Applications

- Building systems
- Automotive
- Synthetic biology
- Medical systems
- Instrumentation
- Factory automation
- Avionics



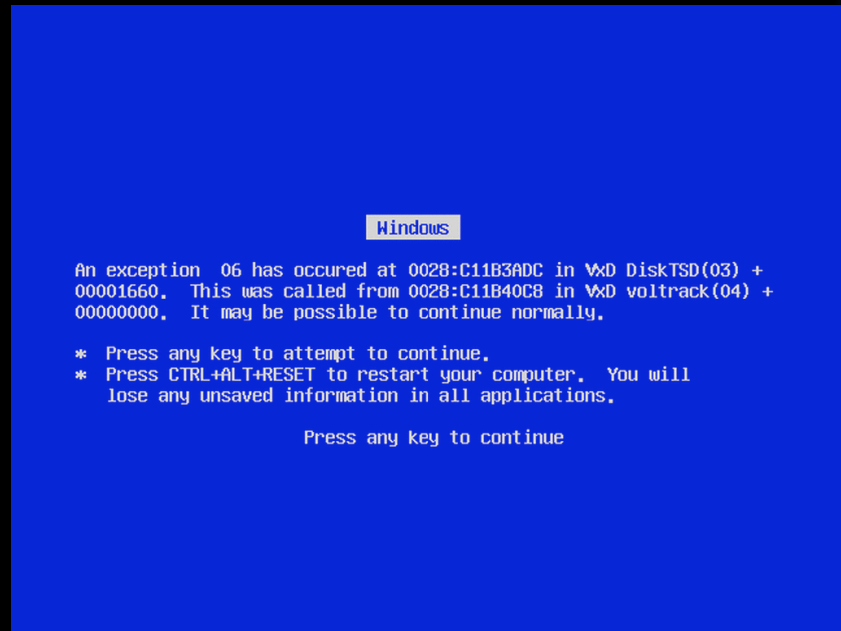


# Why can't we make Software Reliable?



Uptime: 125 years

Source: T. Henzinger



# Why can't we make Software reliable?

## Engineering

Theories of estimation.  
Theories of robustness.

R

## Computer Science

Theories of correctness.

B

Source: T. Henzinger

# Why can't we make Software reliable?

## Engineering

Theories of estimation.  
Theories of robustness.

*Goal: build reliable systems.*

## Computer Science

Theories of correctness.

*Temptation: programs are  
mathematical objects; hence we  
want to prove them correct.*

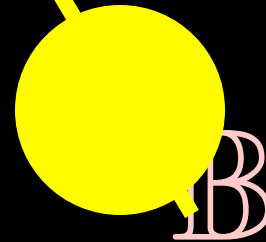
# The CHESS Premise:

The pendulum has swung too far

Engineering

Computer Science

R



B

Source: T. Henzinger

# The CHESS Premise:

The pendulum has swung too far

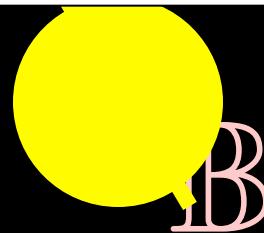
Engineering

Computer Science

Embedded Systems are a perfect playground to readjust the pendulum.

R

Physicality



Computation

Source: T. Henzinger

## Execution constraints

CPU speed  
power  
failure rates

## Reaction constraints

deadlines  
throughput  
jitter

Embedded  
Systems

## Computation

algorithms  
protocols  
reuse

Embedded System Design is  
generalized hardware design  
(e.g. System C).

### Execution constraints

CPU speed  
power  
failure rates

### Reaction constraints

deadlines  
throughput  
jitter

Embedded  
Systems

### Computation

algorithms  
protocols  
reuse

## Execution constraints

CPU speed  
power  
failure rates

## Reaction constraints

deadlines  
throughput  
jitter

Embedded  
Systems

## Computation

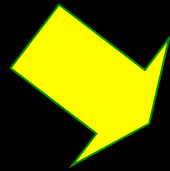
algorithms  
protocols  
reuse

Embedded System Design is  
generalized control design  
(e.g. Matlab Simulink).



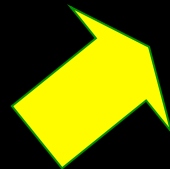
## Execution constraints

CPU speed  
power  
failure rates

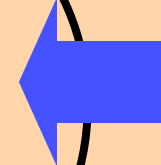


## Reaction constraints

deadlines  
throughput  
jitter



Embedded  
Systems



## Computation

algorithms  
protocols  
reuse

Embedded System Design is  
generalized software design  
(e.g. RT Java).

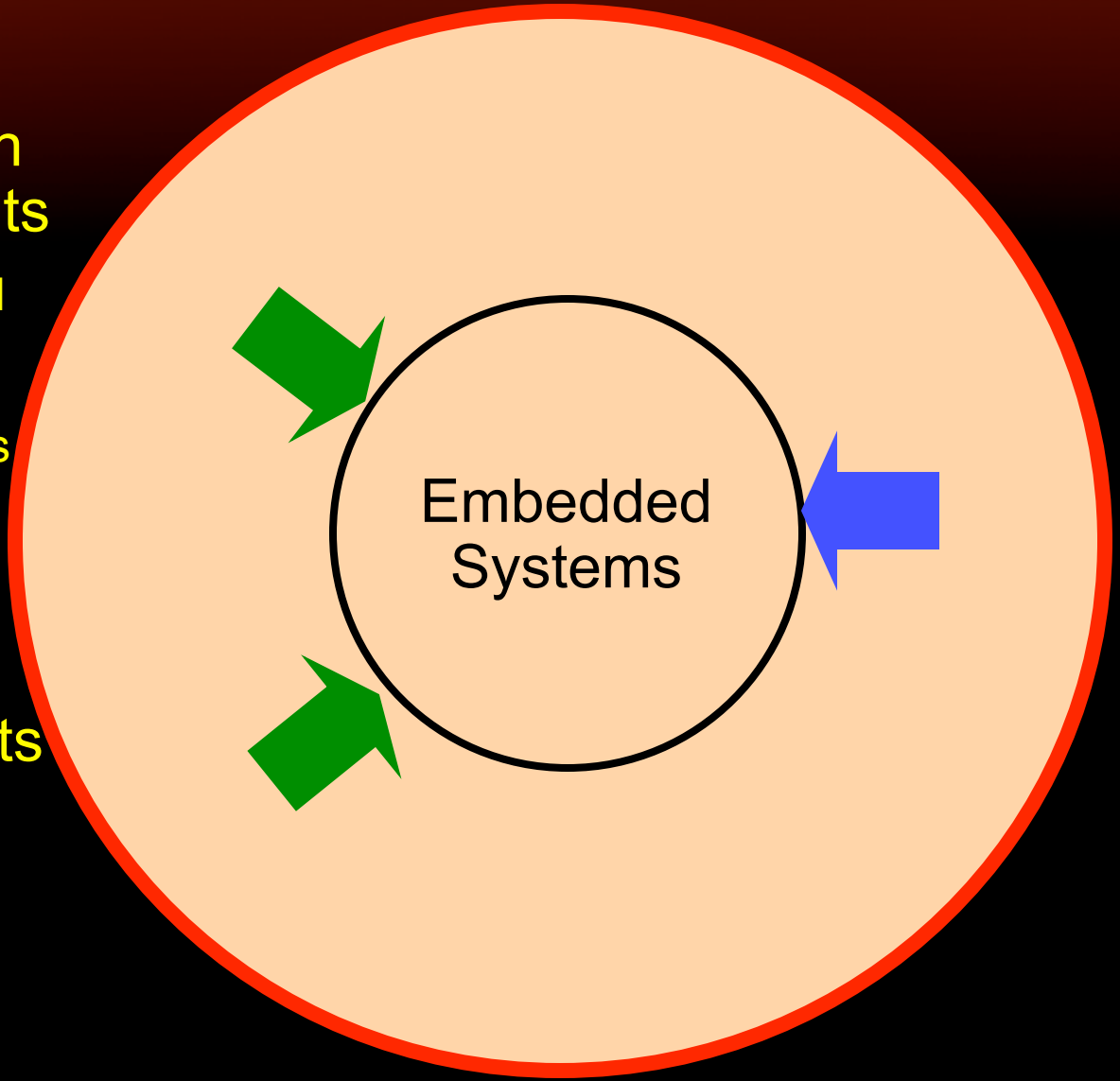
## Execution constraints

- CPU speed
- power
- failure rates

## Reaction constraints

- Deadlines
- throughput
- jitter

Embedded Systems

A diagram illustrating the constraints and computation for Embedded Systems. At the center is a black circle containing the text "Embedded Systems". This central circle is surrounded by a larger, light-orange circle with a thick red border. To the left of the central circle, two green arrows point towards it. To the right, a blue arrow points towards it. The background is dark blue with a textured, stone-like appearance.

## Computation

- Algorithms
- protocols
- reuse

Source: T. Henzinger

# The CHES Challenge

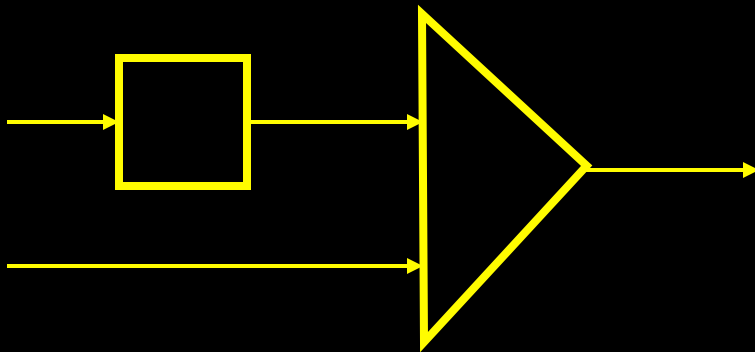
We need a new formal foundation for  
embedded systems, which systematically  
and even-handedly re-marries  
computation and **physicality**.

Source: T. Henzinger

# Integration of the Two Cultures

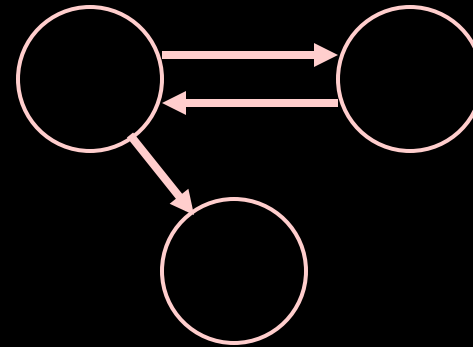
## Engineering

Component model: transfer function  
Composition: parallel  
Connection: data flow



## Computer Science

Component model: subroutine  
Composition: sequential  
Connection: control flow



[Hybrid Systems; Ptolemy; Metropolis; Metamodels]

# Integration of the Two Cultures

## **Equational Models**

Strengths:

Concurrency  
Quantitative constraints  
(time, power, QoS)

Tool support:

Best-effort design  
Optimization

## **Abstract-Machine Models**

Dynamic change  
Complexity theory

Worst-case analysis  
Constraint satisfaction

Engineers must understand both complexities and trade-offs .

Source: T. Henzinger

# The Embedded Software SCIENCE Dilemma



110 Raffaello Sanzio, The Athens School



# Software Architecture Today





# Software Architecture Tomorrow?

