ECE153a/253 Embedded Systems Class Overview

Forrest Brewer

Class Overview

- **What is an Embedded System?**
  - **Physical Constraints: Time, Cost, Power**
  - **Software Engineering in Real Time**
  - **Multiple Stimulus/Response loops**
- **Principles of Structured Design**
  - **Metrics and System Performance**
  - **Correctness and Design Costs**
  - **Specification, Modeling and Abstraction**

Class Logistics

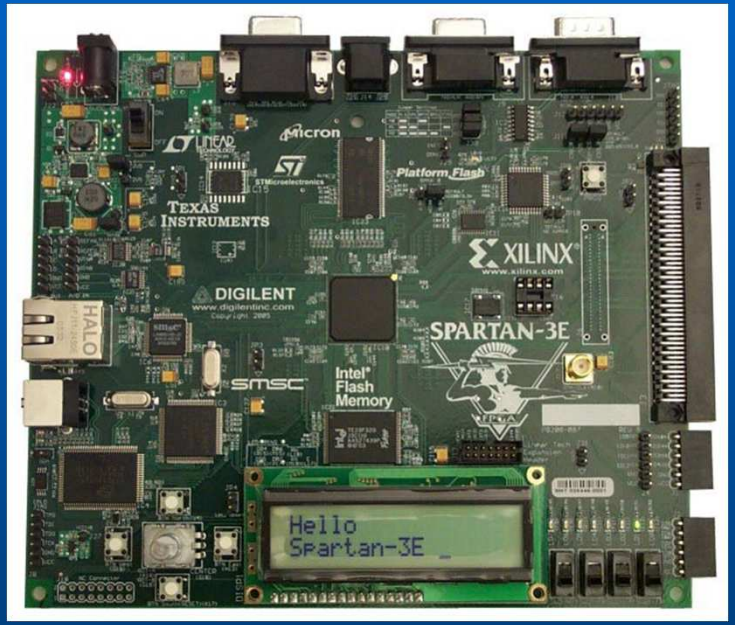
- 2 Weekly Lectures
  - Papers to read (no textbook --)
  - Ref: “Practical UML StateCharts in C/C++” Miro Samek
  - Weekly Exercises (Homework) 25%
  - Out Wednesday, due Wednesday before 3PM in box
- Recitation Section 10%
  - Graded only for undergrads, everyone should go!
- Final Project + Presentation 25%
  - Graduates – ‘open’ final project
  - Undergraduates – directed lab
- 4 orchestrated Labs 40%

Lab Location

- Linux or Windows can be used
  - Beware path issues if moving between platforms
  - You can install 14.3 on your own laptop
    - Need to use VPN to access license off campus
- 2-person teams need to obtain:
  - Digilent Spartan 3E Starter Board
    - $179 from http:/5/www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,792&Prod=S3EBOARD
    - PMOD SPI Microphone (digilent site)
    - Order asap! – labs start in 1 week
- ECI Lab (HFH 1st floor)
  - Xilinx 14.3 Suite
    - Licenses served from 2100@license.ece.ucsb.edu

Class Labs

- **Xilinx/Digilent Spartan 3e Card**
  - **MicroBlaze/PicoBlaze Processors**
  - **Verilog/VHDL Programmable Peripheral Devices**
  - **EDK/SDK tools (ECI)**
- **Each 2 person group responsible for own card!**
  - **Card: $179, microphone, < $200/team**



Graduate Section (ECE 253)

- Extra Readings (See Course Bibliography)
- Must Propose final project and have it approved. Final project must use soft platform (i.e. Xilinx or Altera FPGA – DE0-nano $79 is smaller/faster form factor – beware platform issues!) Groups of 1-2 only.
- Many possible sensors/control PMODS all 5/10 pin SPI
- Beware that SPI is a ‘loose’ standard, be prepared for at least 2 weeks to verify/validate interface to your hardware in addition to design/coding time. I2C seems more generally stable
- We cannot accept last minute platform changes

Syllabus 1

I. Overview of Embedded Systems

- What is an Embedded System?
  - Technology
    - CPU/FPGA/DSP/ASIC
  - Hardware and Software
  - Real-time, limited resources
- Computation Models and Abstractions
  - Why Abstract Models?
  - Models: Circuit, RTL, Threads, Tasks
  - Modeling Time

Syllabus 2

II. Finite-State Automata

- Overview of Finite Automata
- States (DFA/NFA)
- Sampling and Triggering
- Hierarchy and Concurrence (State Charts)
  - Modality
  - Decomposition
  - C implementation
- NFA Models and Scheduling

III. Process Models

- Kahn Process Models
- SDF

Syllabus 3

IV. Data-Flow and Scheduling

- Loops and Timed Behavior
- Constraints, ASAP, ALAP, Resources
- Exact and Heuristic Scheduling
- Real-Time Task Scheduling
  - Periodic and Priority Policies
  - Rate-Monotone and Deadline Scheduling
  - Priority Inversion
  - Preemption, Overhead and Context
- ILP optimization (IBM CPLEX, LP_SOLVE)

Syllabus 4

V. Real-World:

- Sensors
- Signal Sampling/Noise and Jitter
  - Conversion Issues
- Motors and Actuators
  - DC/Servo, Stepper, Pulse-Drive
- Intro to PID Control

Syllabus 5

VI. Tricks of the Trade

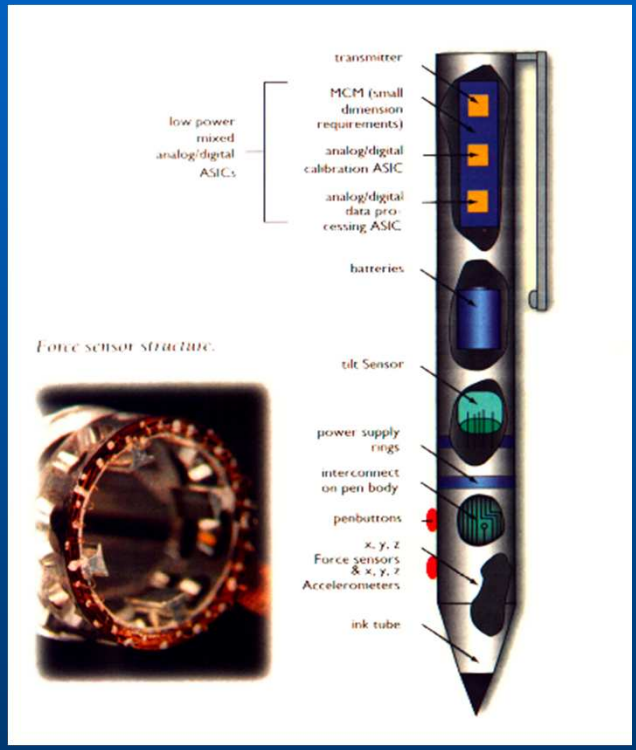
- A. Data Representation
- B. Representation and Computation of Functions
- C. Speculation, Pipelining, Systolic Computation, Trace Optimization
- D. Real-Time Reprise

Embedded Systems

- **Computing systems performing specific tasks within a framework of real-world constraints**
  - **Automotive: ECS, ABS; Aircraft**
  - **Network Appliances: Routers, Modems**
  - **Cell Phones, PDA, Mouse, E-Star Power**
  - **Printers, Hand Mixers, Toasters, Tires!**
- **Microprocessors are Ubiquitous**

Embedded Systems Everywhere!

Tire Pressure Sender



SmartPen

Characteristics of Embedded Systems

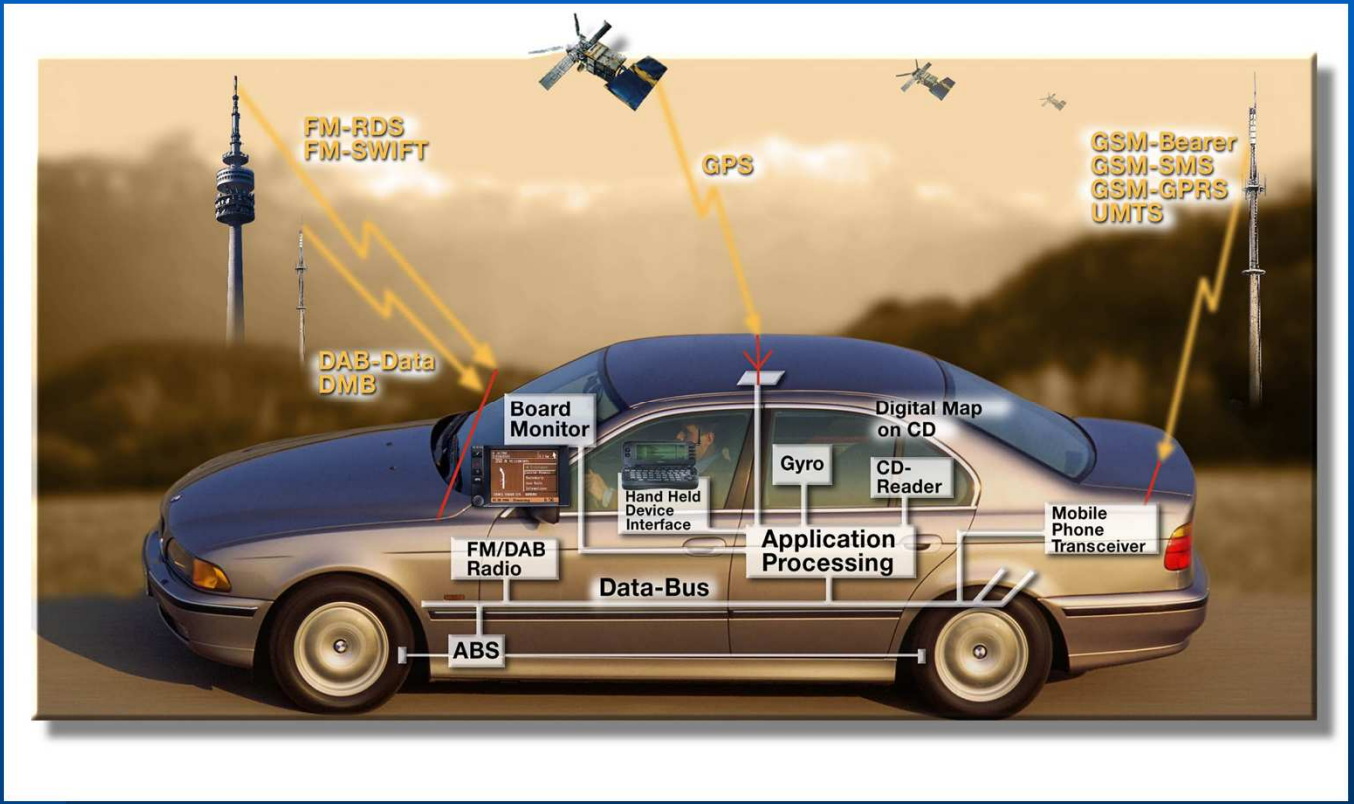
- **Part of larger system**
  - **_not_ a "computer with keyboard, display, etc."**
- **HW & SW application-specific – not G.P.**
  - **application is known a-priori**
  - **definition and development concurrent**
- **Interact (sense, manipulate, communicate) with the external world**
- **Never terminate (ideally)**
- **Operation is _time constrained_: latency, throughput**
- **Other constraints: power, size, weight, heat, reliability**
- **Increasingly high-performance (DSP) & networked**

Slide courtesy of Mani Srivastava®

Why Embed Computers?

- **Enablers**
  - **New behaviors and applications (GPS, PDA, Wii)**
- **Cost!**
  - **Intelligent Control: workaround manufacturing flaws (Auto ABS), replace antiquated controllers (Toaster), combine functions (Cell Phone)**
- **Remote Sensing and Control**
  - **Expanding Human perception and effectiveness**

Embedded Automotive



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

Slide courtesy of Alberto Sangiovanni-Vincentelli®

Why do we care? Some Market Tidbits...

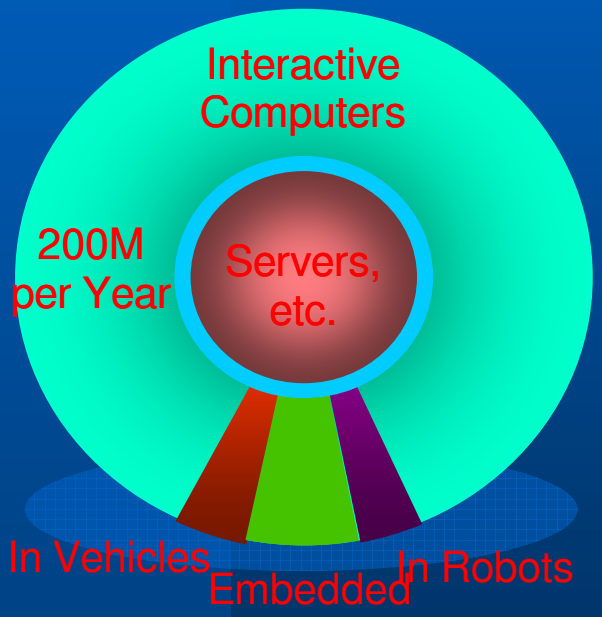
- **Specialized devices and appliances replacing PCs**
  - **variety of forms: set-top boxes, fixed-screen phones, smart mobile phones, PDAs, NCs, etc.**
  - **In 1997, 96% of internet access devices sold in the US were PCs, by 2004, shipments far exceeded PCs**
  - **2009 Tire Pressure Sensors: 60-70M/yr, Smart cards 100-200M/yr**
- **Traditional Products**
  - **dependent on computation systems**
  - **Modern cars: ~100 processors running complex software**

Slide courtesy of Mani Srivastava®

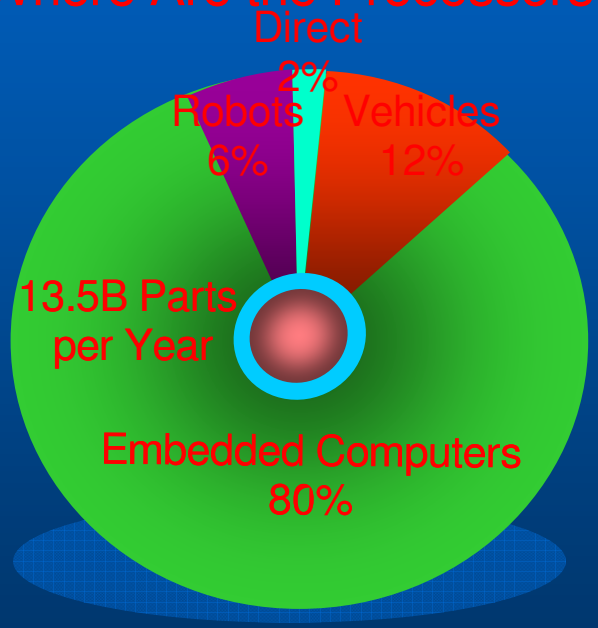
Where are the CPUs?

Estimated 98% of 8 Billion CPUs produced in 2000 used for embedded apps

Where Has CS Focused?



Where Are the Processors?



Look for the CPUs...the Opportunities Will Follow!

Source: DARPA/Intel (Tennenhouse)

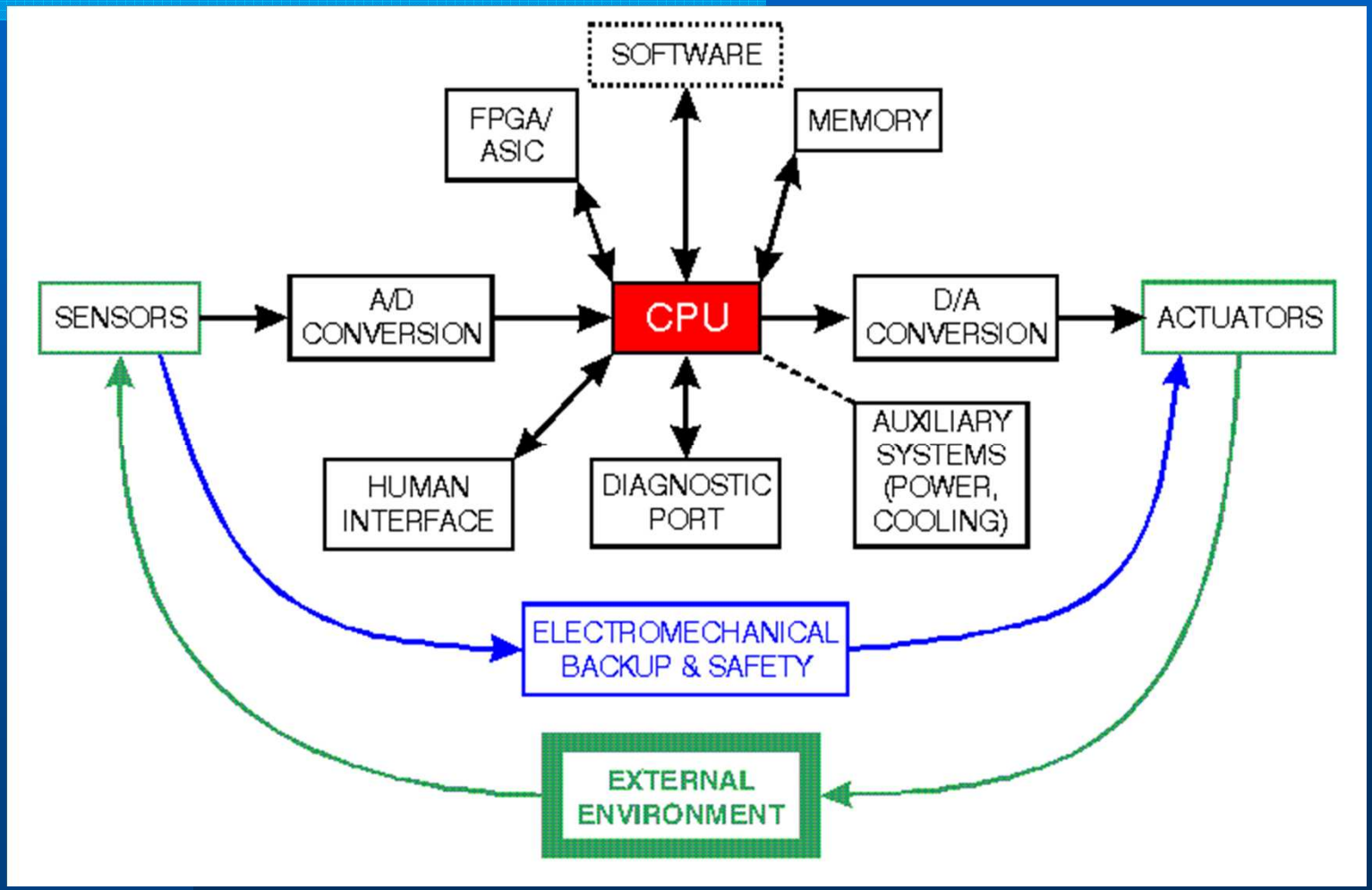
Design Issues

- **Complex Systems!**
  - **How to get it working?**
    - **(on time, on budget)**
    - **Need for abstraction and design reuse**
  - **How to test it?**
    - **Unforeseen Behaviors (Air Bus!)**
- **Real Time Physical Embedding**
  - **Does it meet constraints?**
  - **Design Budgeting: Power, Size, Cost, Reliability**
  - **What are the exploitable design options?**

Technology

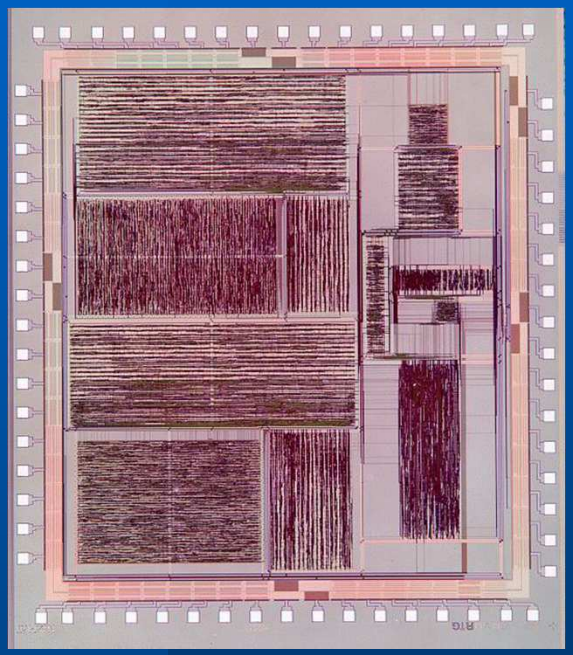
- **Integrated Processors**
  - **Nearly free in production**
- **A/D, D/A, Sampling**
  - **Interface Analog world to cheap computing**
- **MEMS, NEMS, Opto-Devices**
  - **Miniature Direct Coupling to Real World**
- **Batteries, Solar Cells, Vibration Scavenging, Thermal Gradient Cell**
  - **Computation and Communication Power**

"Traditional" Software Embedded Systems = CPU + RTOS



Slide courtesy of Mani Srivastava®

ASIC Hardware Embedded System



ASIC Features

- **Area: 4.6 mm x 5.1 mm**
- **Speed: 20 MHz @ 10 Mcps**
- **Technology: HP 0.5 μm**
- **Power: 16 mW - 120 mW (mode dependent) @ 20 MHz, 3.3 V**
- **Avg. Acquisition Time: 10 μs to 300 μs**

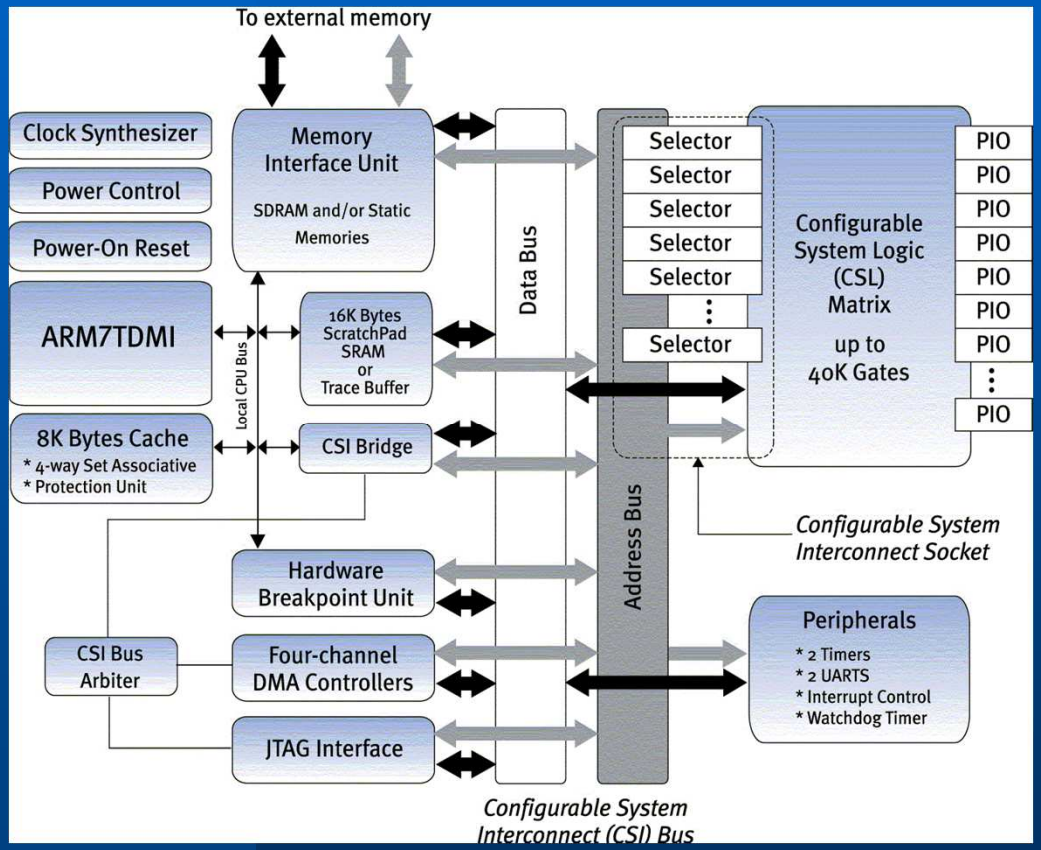
- **A direct sequence spread spectrum (DSSS) receiver ASIC**

Slide courtesy of Mani Srivastava®

ASIC Issues

- **Good:**
  - **High performance custom peripherals**
  - **Multiple heterogenous Cores**
  - **Integrated A/D, D/A, timers …**
  - **Low Cost, High performance on-chip communication**
  - **Low Part Cost in Volume**
- **Bad:**
  - **Very expensive ($5-$75M/design)**
  - **Very High Risk (Several total failure points)**
  - **Potentially impossible to Program even if working!**

Reconfigurable SoC



Triscend's A7 CSoC

Other Examples

Atmel's FPSLIC (AVR + FPGA)

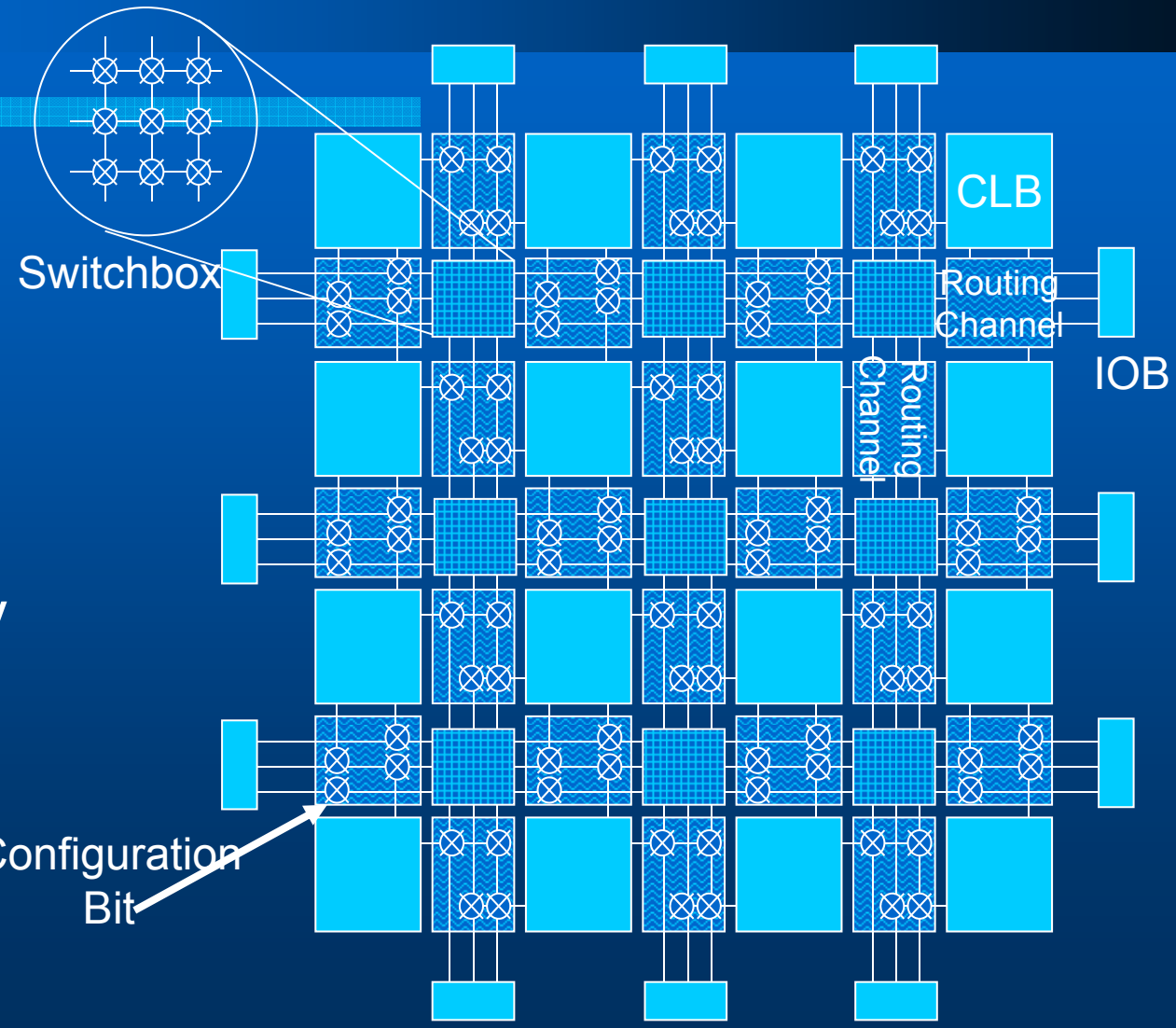
Altera's Nios (configurable RISC on a PLD)

TI's OMAP (ARM Cortex+ Custom GPU+ TI DSP)

Slide courtesy of Mani Srivastava®

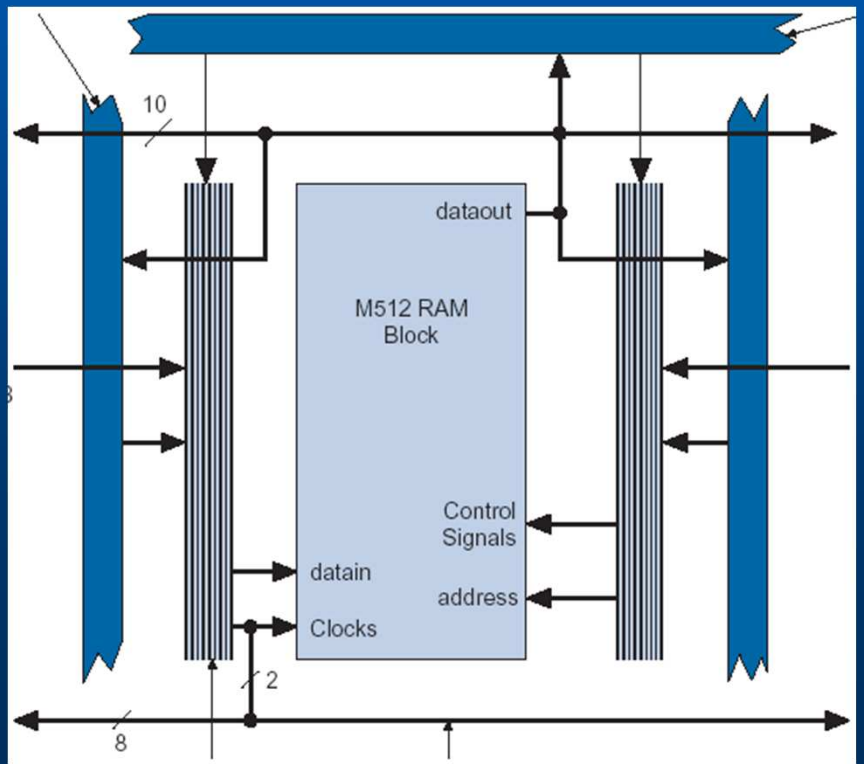
FPGA

FPGA advantage: performance of parallel hardware with the flexibility of software

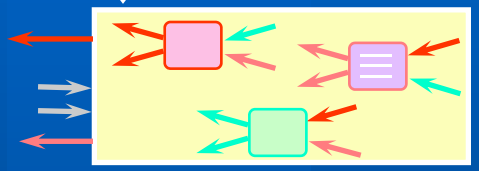


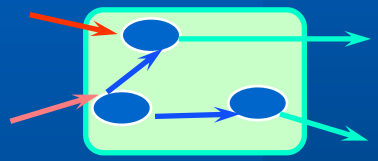
FPGA Embedded RAM

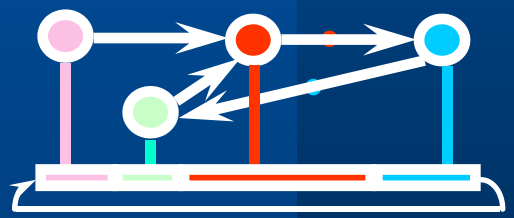
- Xilinx – Block SelectRAM
  - 18Kb dual-port RAM arranged in columns
- Altera – TriMatrix Dual-Port RAM
  - M512 – 512 x 1
  - M4K – 4096 x 1
  - M-RAM – 64K x 8

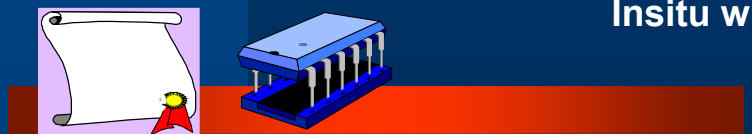


Embedded System Design Flow









Modeling

the system, and algorithms involved;

Refining (or "partitioning")

the function into smaller, interacting pieces

Test Bench Design

Abstractions from the Design, Communication, Storage, and Computation resources

HW-SW partitioning: Allocation

(1) HW

(2) SW

Determine power and performance bounds

Scheduling

When are functions executed

Resource Arbitration

Mapping (Implementing)

(1) software, (2) Hardware,

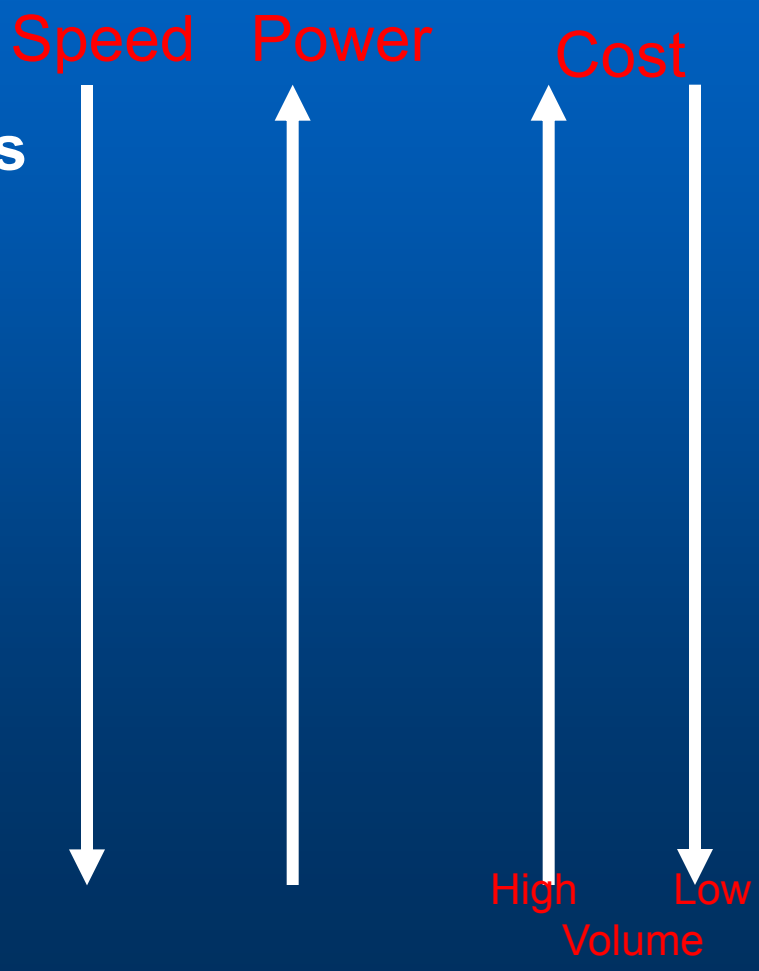
(3) Coherence/Communication

Testing/Validation

Insitu with Design

Many Implementation Choices

- Microprocessors
- Domain-specific processors
  - DSP
  - Network processors
  - Microcontrollers
- ASIPs
- Reconfigurable SoC
- FPGA
- GateArray
- ASIC
- Full-Custom (COTS)



Slide courtesy of Mani Srivastava®

Hardware vs. Software Modules

- **Hardware = functionality implemented in custom architecture**
- **Software = functionality implemented stored program**
- **Key differences:**
  - **Configurability, Adaptability**
  - **Process Time Multiplexing**
    - **software modules time multiplexed on a processor**
    - **hardware modules often mapped to dedicated hardware**
  - **Concurrency**
    - **processors have serial "thread of control"**
    - **dedicated hardware has concurrent activity**

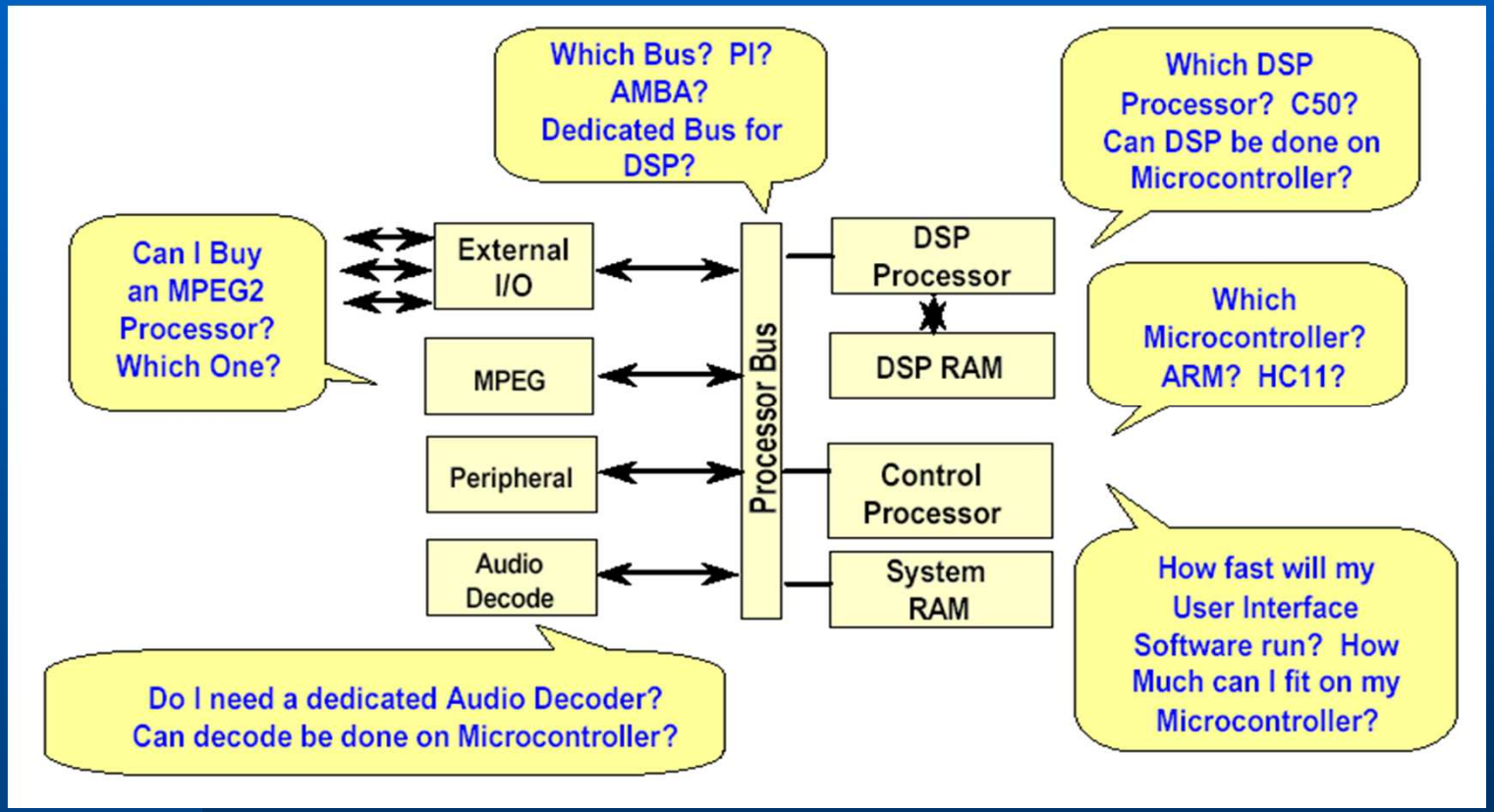
Slide courtesy of Mani Srivastava®

Hardware-Software Architecture

- A significant part of the system design problem is deciding which parts are software, and which are hardware
- Issues:
  - Cost of development
  - System performance
  - Upgrade potential
  - Sales/Implementation Volume
  - Availability of IP
  - Development Time-Line

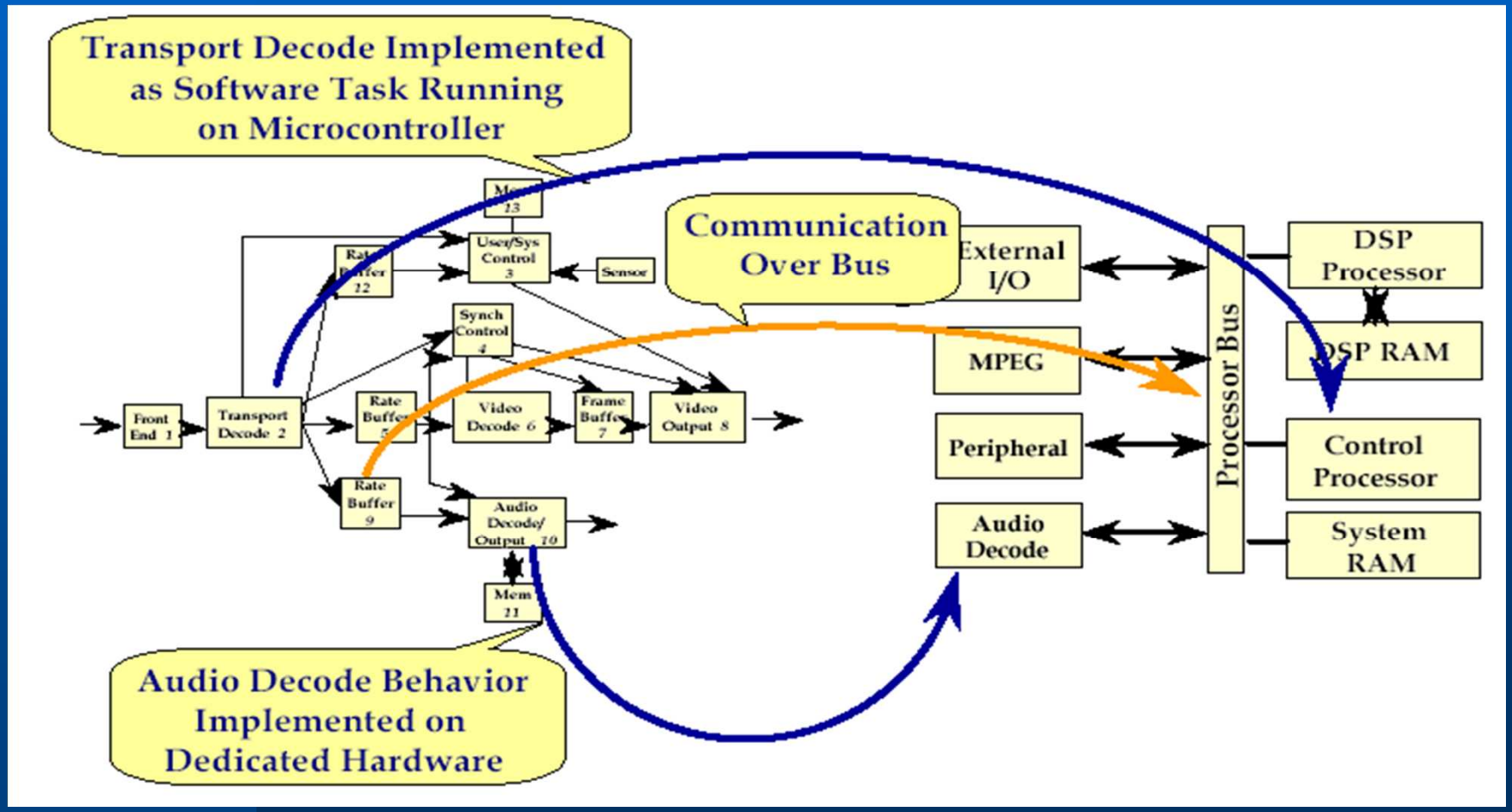
Slide courtesy of Mani Srivastava®

IP-based Design



Slide courtesy of Mani Srivastava®

Map Behavior to Architecture



Metrics / Models / Abstractions and Systematic Analysis

To make progress, we need objective means to evaluate heterogeneous collections of components, to test and validate correct operation and models to abstract system complexity.

- **Models of Computation**
  - **Provide metrics on component and system performance**
    - **Time, Throughput, Power?**
  - **Abstract models as bounds on system behavior**
    - **Provable or at least Simulation bound on system correctness**
  - **Unfortunately– no complete, encompassing, provable model exists**
    - **Lots of small provable sub-models: FSM, eFSM, SDL, Kahn Processes**
    - **Lots of encompassing but undecidable models: RT-FSM, General Modal Logic**
- **In search of set of generally useful abstractions**
  - **Embedded Systems a bit like software compilers in Lisp/Fortran era**

This Class

- Focus on tightly constrained environments and high performance demands
  - Multi-kHz sample rates
  - Multiple Interrupt sources
  - Relatively small memory, realistic bus environment
- Will use HFSM approach to organize execution and task dispatch (1-2kbytes)
- Labs focus on software side of interface (hardware issues in 153b).
- Conceptual understanding of broader aspects of real-time systems.
  - Scheduling, Priority Inversion, Atomic Code Blocks