

CSCE 313

Embedded Systems Design

2004/1/12

Lecture 1

Overview of Embedded Systems & Architecture

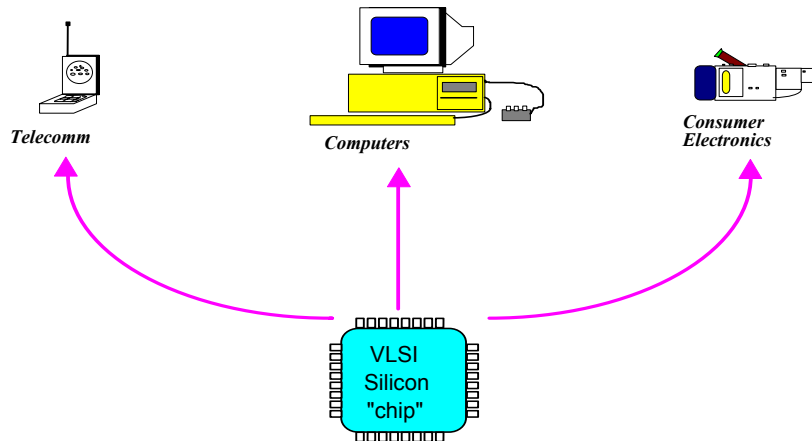
© 2003 Dr. James P. Davis

Some figures from Tanenbaum © 1999 Prentice Hall Publishers, Inc.

CSCE 313 Lecture 1 - Outline

- Introduction
 - ✓ Why this course is important to Computer Engineering.
 - ✓ Drivers for Embedded Systems.
 - ✓ Example: Wireless Communications.
 - ✓ The CPU is the Heart of an Embedded System.
- Computer Architecture
 - ✓ Computer Engineering Design Space (Y-chart).
 - ✓ Abstraction Levels and Representation Domains.
 - ✓ Organization of a Computing System.
 - ✓ Organization of the CPU.
- Course Drivers
 - ✓ The CPU architecture and programming model.
 - ✓ The layers of the “virtual machine” (the Instruction Set Architecture).
 - ✓ The program development process.
 - ✓ Operating at the Hardware-Software boundary.

Embedded Systems - The New Realities



"At the root of cascading changes of modern economic life...devaluing resources in technology, business and geopolitics...overcoming the constraints of material resources, the microchip has devalued most large accumulations of physical capital and made possible the launching of global economic enterprises...microchips find their value not in their substance but in their intellectual content: their design..."

George Gilder, Microcosm, 1989

© 2003 Dr. James P. Davis Page 3



Embedded Systems - Industry "Drivers"

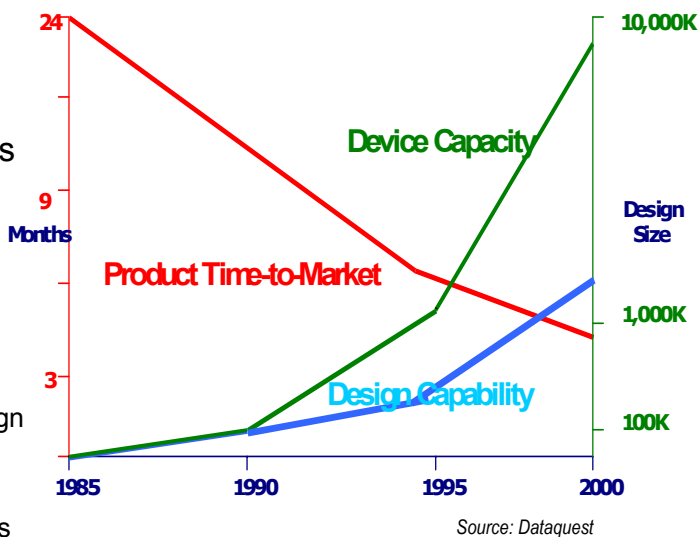
- Many market and technology factors coming together to create pressure on electronics product engineering organizations worldwide.
 - ✓ Increasing global competition and new markets.
 - ✓ Increasing rate of product innovations and new product introductions.
 - ✓ Decreasing time-to-market windows.
 - ✓ Decreasing "shelf life" for products in many categories.
 - ✓ Increasing pressure on competitive cost containment, profit margins.
 - ✓ Increasing convergence: integrated functionality in single electronics devices and product packages.
 - ✓ Increasing quality expectations: means for containing distribution and support costs.
 - ✓ Increasing innovation in silicon process technology and wafer scale integration densities, also in embedded software technologies.
- ✓ Increasing disparity: capacity of the underlying technologies versus capability of designers to manage increasing design complexity.



© 2003 Dr. James P. Davis Page 4

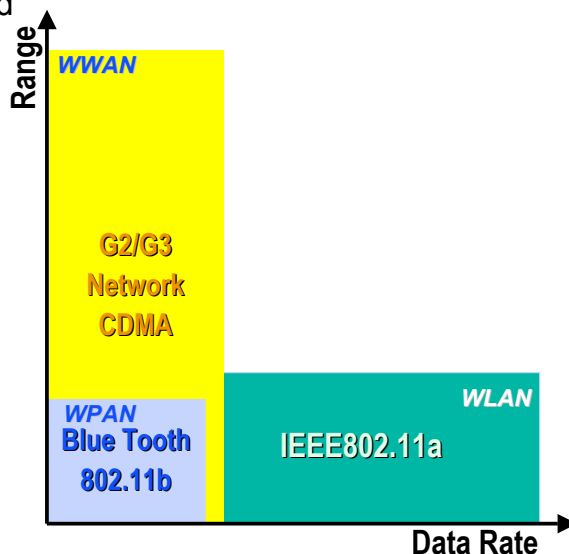
The Capacity vs. Capability Gap

- Increasing capacity of the technology:
 - ✓ The rate of new technology and associated silicon process changes has continued to follow Moore's Law.
- The capability of designers and design teams to use this capacity isn't keeping up.
 - ✓ The Capacity versus Capability Gap is widening.
 - ✓ Each set of technology and process changes requires designers to manage ever more complexity in the design process.
 - ✓ New architectures, abstractions, methods and tools are required to address this increasing complexity.



Example – Wireless Communications

- The market is seeking product technology options to cover different geography ranges and data rates.
 - ✓ Bluetooth – **WPAN**.
 - ✓ IEEE 802.11 - **WLAN**.
 - ✓ 2/3G Network – **WWAN**.
- The opportunity for creating value chains encompassing product offerings, distribution and new service offerings hinges on the ability to get low cost solutions to market quickly.
 - ✓ Deliver content to wireless handheld devices.
 - ✓ Function convergence in the handset and at the base station.
 - ✓ Requires large cross-functional design teams in varied disciplines.

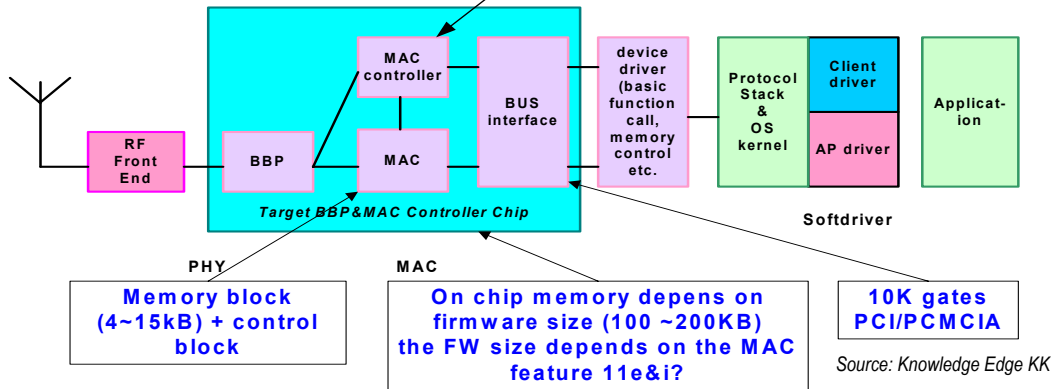


Source: Knowledge Edge KK

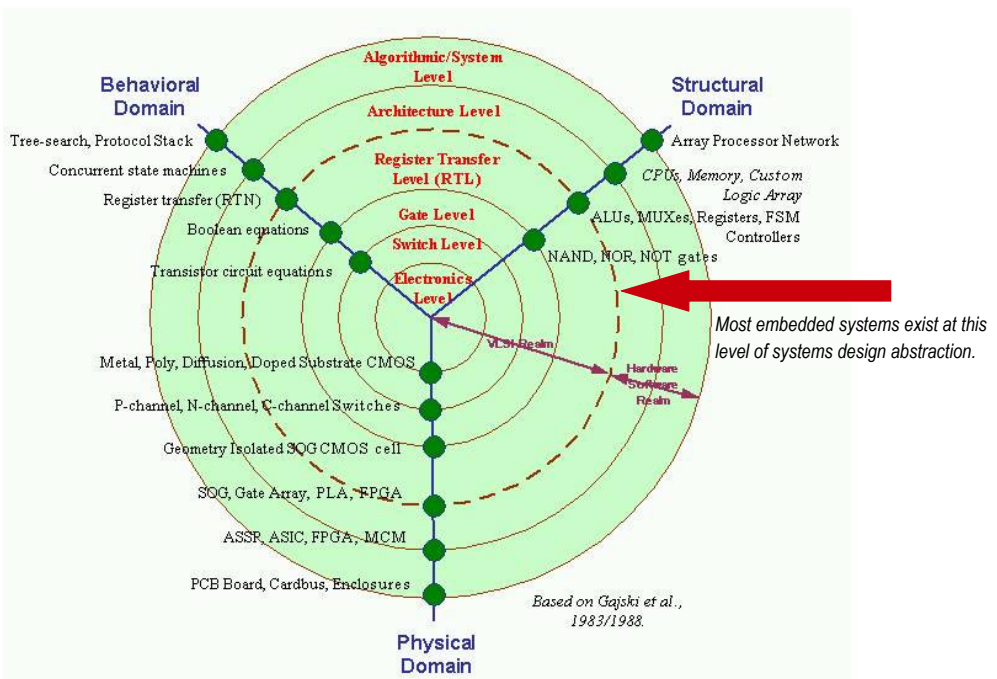


Embedded System – 802.11 WLAN NIC

CPU core	Die area	Peak Power Consumption (mW/MHz)	Memory System	Clock frequency & MIPS performance
ARM9E / ARM946E-S cached processor with tightly coupled memory interfaces	4.9mm ² on 0.18µm estimated size with 16KB instruction & 4KB data caches and no TCMs Using Artisan cell library & RAM compiler	1.1 mW/MHz @ 1.8V (estimated)	Selectable I & D cache sizes: 0, 4K, 8K... 1M Selectable I & D TCM sizes: 0, 4K, 8K... 1M	150MHz on TSMC 0.18µm (worst case) 230MHz on TSMC 0.18µm (typical)

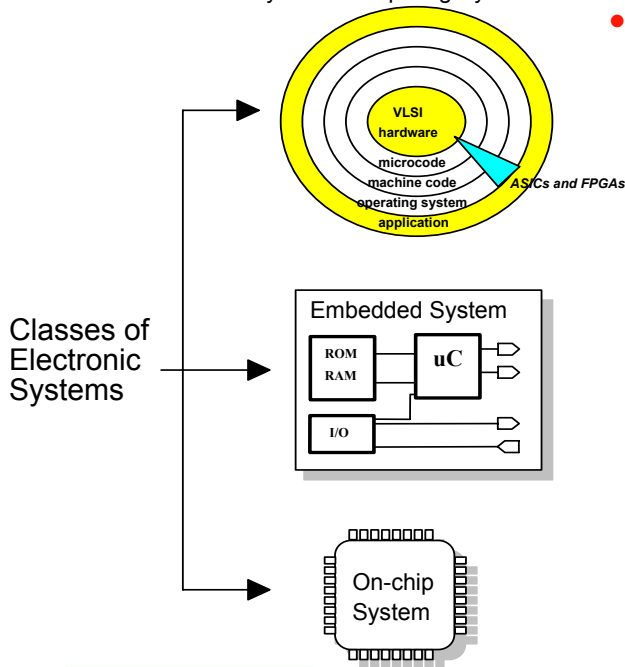


Computing System Design Space (Y-chart)



Categories of Digital Systems Design

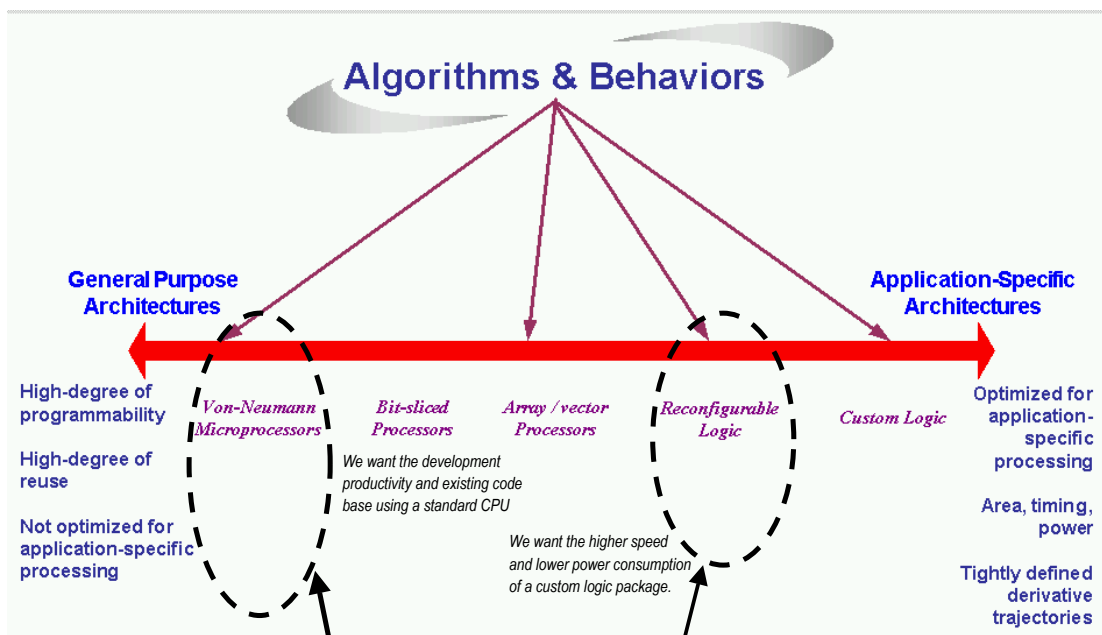
"Layered" Computing System



- Electronic systems today are of different types, depending on the (1) function, (2) application.
 - ✓ Computing system: CPU and hardware executing O/S, with applications running on top of O/S.
 - ✓ Embedded system: fixed-functions, instruction-based micro-controller with both hardware and software components.
 - ✓ On-chip system: complete control and data functions implemented in "custom logic" VLSI package.
 - ✓ On-chip systems can be components of embedded systems, which can be part of a layered "virtual" machine.
 - ✓ All these systems **do** interface with the outside "analog" world.



Continuum of Embedded System Types

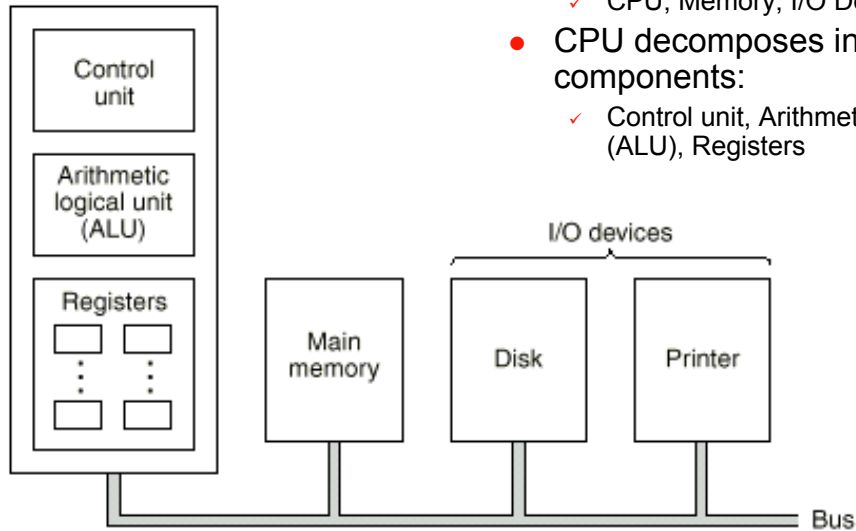


- Industry trends have us focusing on the use of CPU cores and EEPROM/Flash memory packaged with other logic in reconfigurable FPGA devices for small form-factor products (e.g., PDAs).



Computer Structure & Organization-1

Central processing unit (CPU)



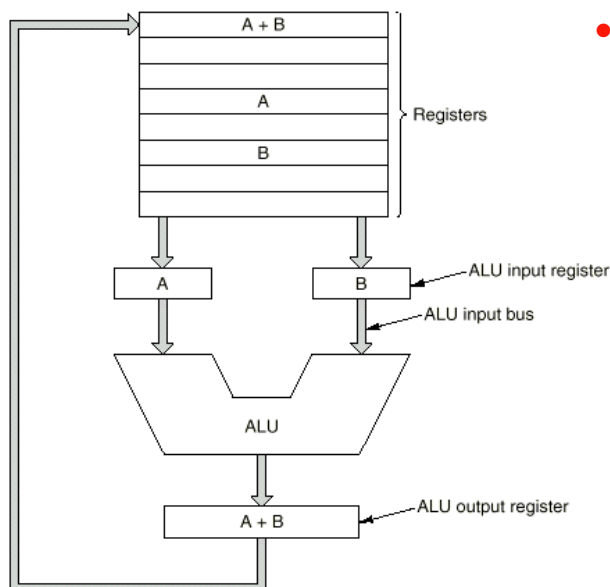
Source: Tanenbaum, 4th ed. © 1999, Prentice-Hall



© 2003 Dr. James P. Davis Page 11

- Basic systems components of Computer:
 - ✓ CPU, Memory, I/O Devices, Bus
- CPU decomposes into basic components:
 - ✓ Control unit, Arithmetic Logic Unit (ALU), Registers

Computer Structure & Organization-2



Source: Tanenbaum, 4th ed. © 1999, Prentice-Hall

- Von Neumann computer architecture:
 - ✓ Developed in the late 1940s by John Von Neumann (Princeton U.)
 - ✓ Combined the elements of *stored program machine*: one machine could run many programs.
 - ✓ Program instructions stored in memory and fetched, in sequence, to be executed by CPU.
 - ✓ Results of execution stored in Registers, later written back to memory.

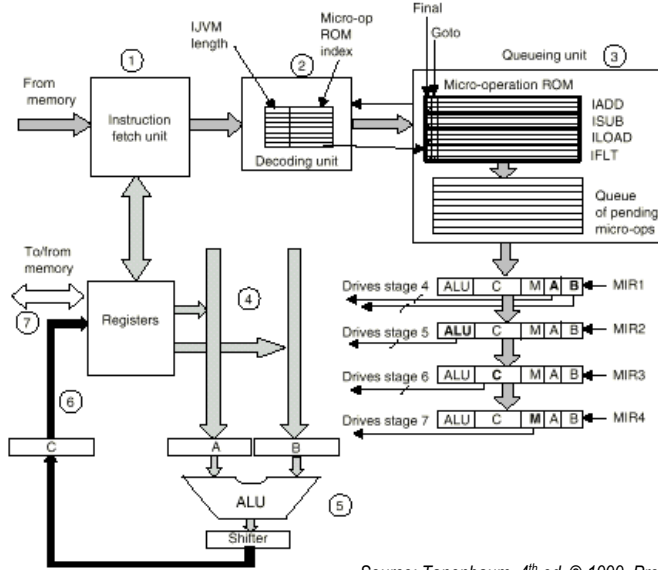


© 2003 Dr. James P. Davis Page 12

Computer Structure & Organization-3

- Instruction Sequencing

- ✓ This CPU has the sequencing of data path operations by one or more state machines
- ✓ The example shown is the data path for a small CPU, where micro-operations based on program instructions are decoded and staged to execute multi-cycle instructions out of memory.
- ✓ This example also used pipelining (discussed on next slide).



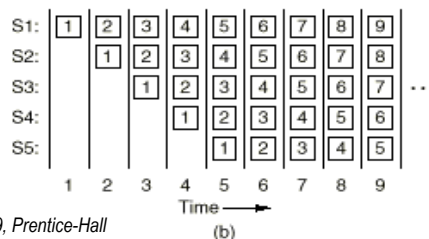
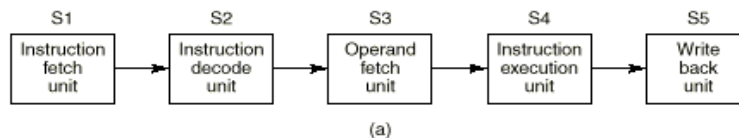
Source: Tanenbaum, 4th ed. © 1999, Prentice-Hall



Computer Structure & Organization-4

- N-Stage Pipeline

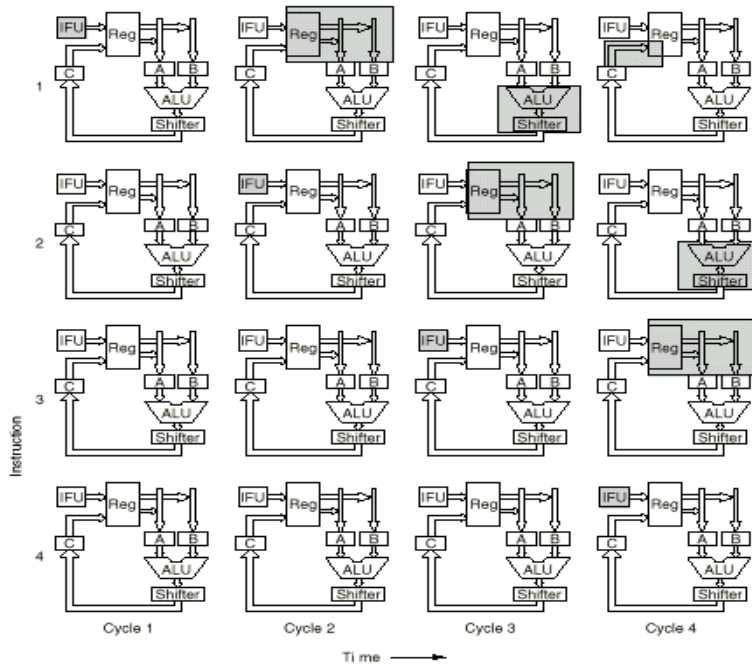
- ✓ Pipeline allows serial processing, in sequence, of instructions or data elements. Each n-element in the pipeline processes its task, then passes the element to the stage n+1 in the pipeline.
- ✓ Design structures that use pipelining: CPU Instruction Fetch Unit (IFU), Digital filters (e.g., FIR, IIR).



Source: Tanenbaum, 4th ed. © 1999, Prentice-Hall



Computer Structure & Organization-5



Source: Tanenbaum, 4th ed. © 1999, Prentice-Hall

- Pipelining Instructions
 - The sequence of figures show how pipelining works in the control path.
 - The control pipelining is the Instruction Fetch, Decode, Execute cycle used in all CPU architectures.
 - Each stage of the control pipeline is buffered by registers that provide setup of data.
 - The different stages of the pipeline also use handshaking.

© 2003 Dr. James P. Davis Page 15

Summary - What You'll Get in This Course

- This course is about analysis, architecture and programming of digital embedded systems.
 - ✓ We assume the presence of a CPU, and therefore our primary task is to implement functionality using the CPU's Instruction Set.
 - ✓ We are concerned about efficiency—in terms of execution speed of instructions, and in the amount of memory we consume—to get a specific programming task accomplished.
- You will learn development process, assembler program design methods, and program verification framework, allowing you to take an embedded systems application, analyze it, and implement it.
 - ✓ Iterative enhancement, top-down/bottom-up, stepwise refinement (stables of software engineering discipline)
 - ✓ Additional heuristics in relating properties of the CPU's computer architecture (instruction set, bus structure, interrupt handling, etc.) to the embedded systems application, to make the systems planning process more effective.
- You will learn how to evaluate the “goodness” of your embedded systems programs based on the use of tools, and by collecting and analyzing program execution data to tune your programs.
 - ✓ We'll use the WISM68K emulator/debugger, and the 68KMB hardware lab apparatus as our tools.



© 2003 Dr. James P. Davis Page 16