MODULE 14: Selected Topics 2

Lecture 14.1
Embedded Systems

Prepared By:

- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech

Invent the Future®

Lecture 14.1 Objectives

- Differentiate between embedded and general-purpose computing systems
- Describe alternatives for embedded systems hardware platforms, including microcontrollers, “systems on a chip,” configurable hardware, and custom-designed hardware
- Describe key features of software for embedded systems

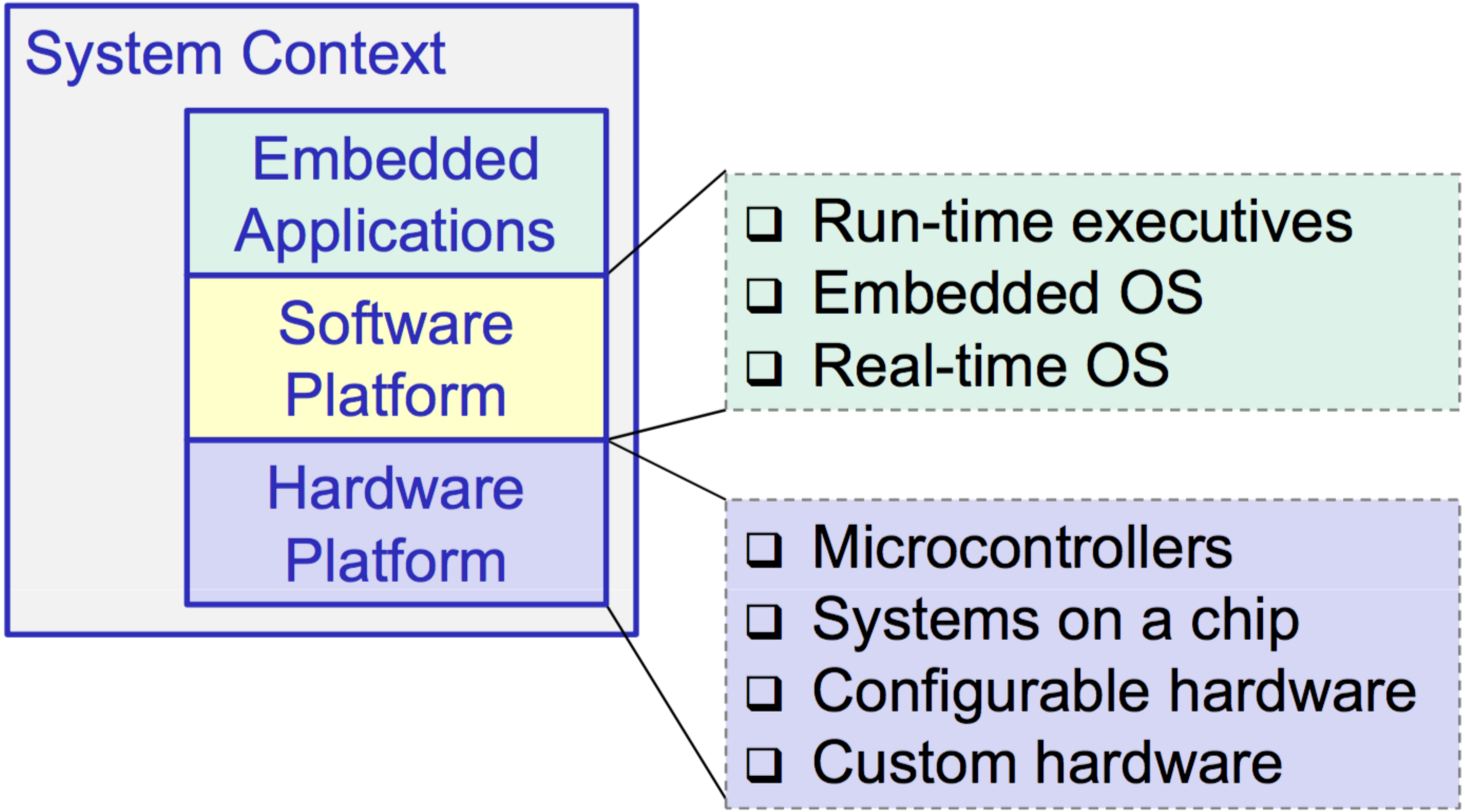
Whither General-Purpose Computing?

- Computing has moved to many platforms other than general-purpose computers
- Beyond general-purpose computing
  - Embedded computing – computing in a system
  - Mobile computing – handheld and similar multi-function devices
  - Sensor networks – integrated sensing and networking
  - Ubiquitous and pervasive computing – computing fades into the background and is “ambient”
  - Cyber-physical systems – tightly conjoin the cyber (computing, communications, and control) with physical environments and systems

Embedded Computing Drivers

- Low-cost processors
- Increasing capabilities on a single chip
  - Processing
  - Memory
  - Communications
- Motivation for flexible products
  - Configuration at design time through software and programmable logic
- Need for autonomic and adaptive systems
  - Self-configuration during operation

Embedded Systems



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Differentiate between embedded and general-purpose computing systems

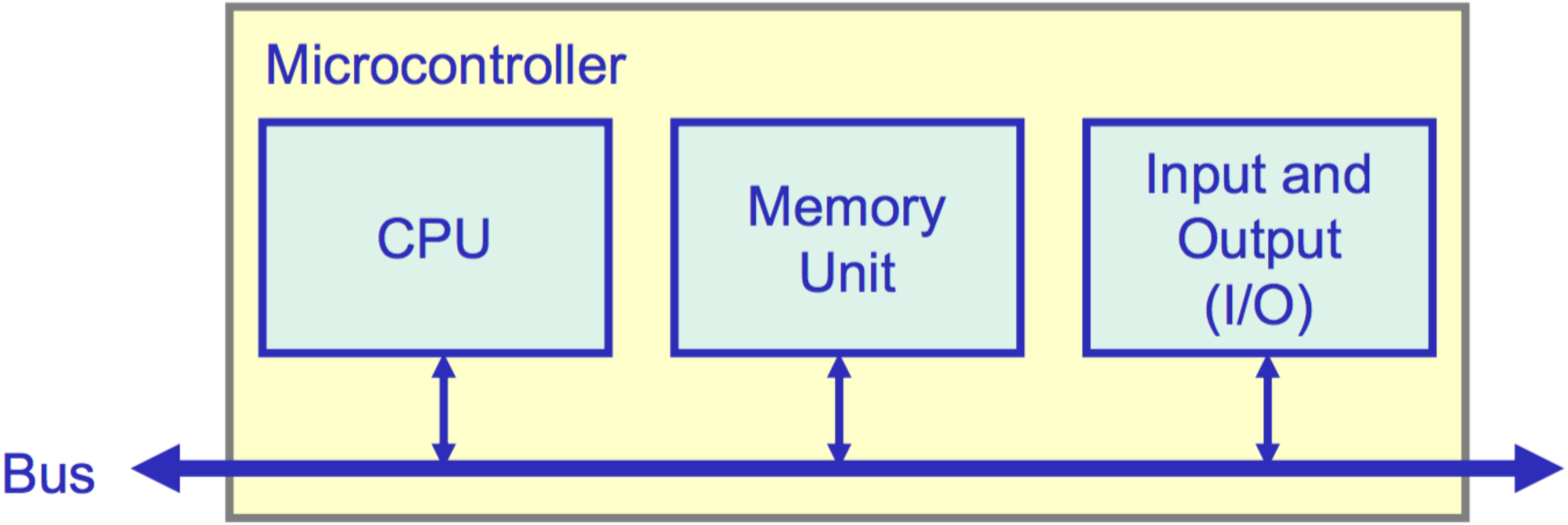
If you have any difficulties, please review the lecture video before continuing.

Microcontrollers

- A microcontroller is a low-cost, highly integrated, single-chip computer for embedded applications
- Single-chip and integration of other functions, such as memory and input/output, reduces system cost
- Used in systems that we do not normally think of as a computer
- For example, the “PIC” in the PIC microcontroller stands for “Peripheral Interface Controller”

Microcontroller Features

- The fundamental difference between a microcontroller and a microprocessor is in packaging
- Microcontroller integrates CPU, memory, and I/O
- Microprocessor provides a CPU, but relies on external memory and I/O controllers



Microcontroller Features (cont’d)

- Microcontrollers are typically designed with other features that differ from general microprocessors
- Built-in I/O control functions, e.g., serial or parallel port
- Analog-to-digital or digital-to-analog converters
- Built-in timers and counters
- Multiple interrupt sources
- Watchdog timers to monitor system sanity
- Special instructions for bit-level control
- Low-power sleep modes
- Built-in electrically erasable programmable read-only memory and read-only memory

Microcontroller Features (cont’d 2)

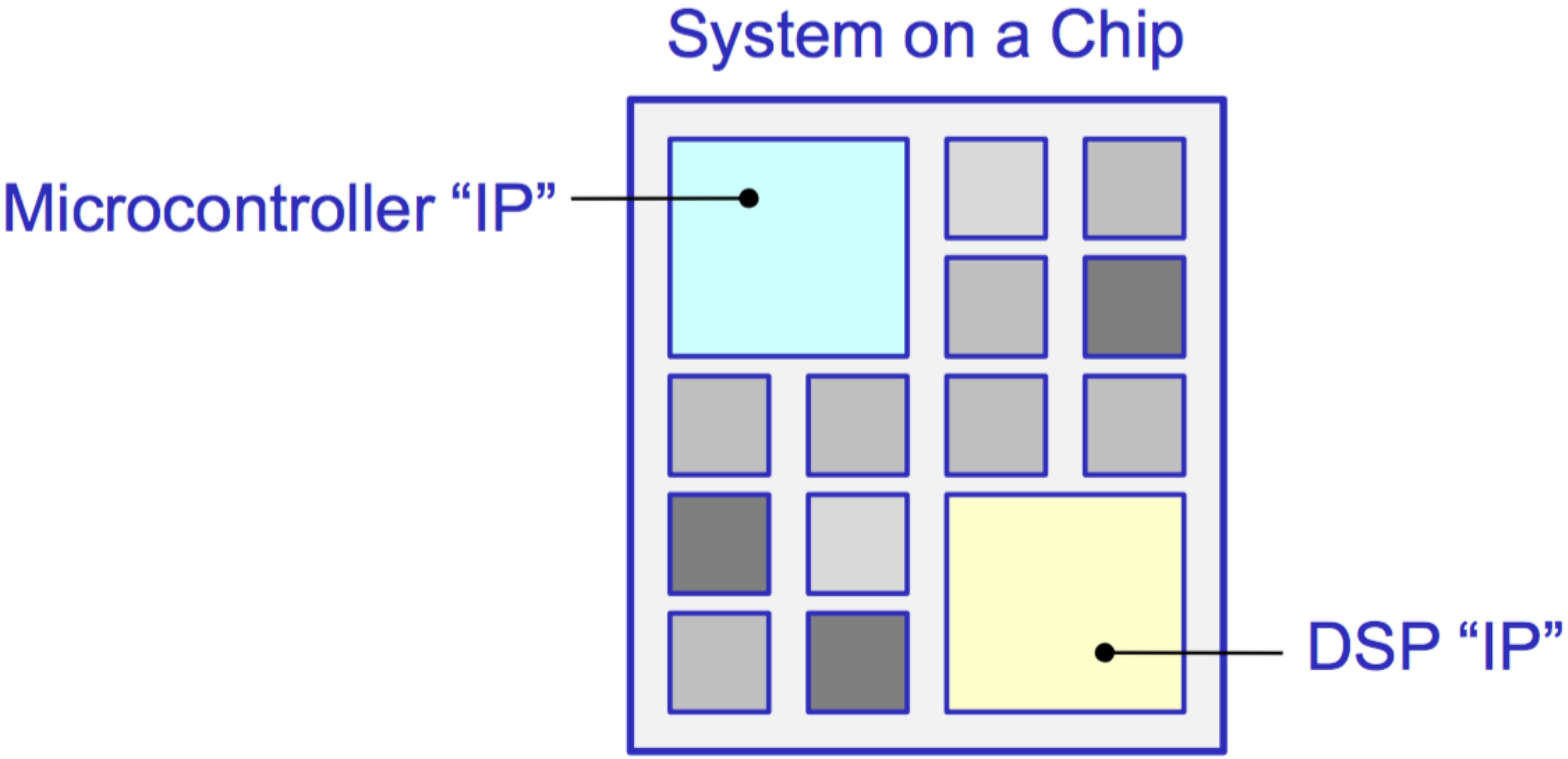
- Microcontrollers typically don’t have high-performance computing features
  - Floating point support
  - Large address spaces
  - Advanced memory systems supporting cache and virtual memory
- Digital signal processors (DSPs) as microcontrollers
  - Combine some features of microcontrollers with fast arithmetic, especially for multiply-accumulate operations for signal processing

System on a Chip

- The system on a chip (SoC) concept takes the microcontroller concept even further by integrating additional functionality onto an integrated circuit
- Approaches
  - Domain specific, e.g., microcontroller plus multimedia decoding (MPEG, etc.) for a variety of entertainment applications
  - Application specific, e.g., microcontroller plus multimedia decoding plus specific peripherals for a specific class of entertainment product
  - Product specific as part of a customized device

System on a Chip (cont'd)

- Microcontroller and other designs can be provided as protected "intellectual property" (IP) files
- Designs are then integrated into a programmable or custom integrated circuit



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Describe alternatives for embedded systems hardware platforms, including microcontrollers and “systems on a chip”

If you have any difficulties, please review the lecture video before continuing.

Configurable Hardware

- Configurable hardware presents a compromise
  - Faster for many functions than a microcontroller or microprocessor, but generally harder to design functionality
  - Much simpler to design and much cheaper to produce for small to moderate quantities than a custom-designed application-specific integrated circuit (ASIC)
- Configuration may be done at:
  - Design time (most common)
  - Run-time using the “configurable computing” model (less common)

Forms of Configurable Hardware

- Programmable Array Logic (PAL)
  - Early form of configurable hardware
  - Highly structured and constrained
  - Logic controlled by making or breaking connections
  - Can be programmed by a manufacturer (for moderate to large quantities) or by the developer (for development and small quantities)
  - Some versions allow reprogramming
  - Some versions incorporate flip-flops to store state
- Programmable Logic Arrays (PLAs)
  - Similar to PALs, but more flexible

Forms of Configurable Hardware (cont'd)

- Field-Programmable Gate Arrays (FPGAs)
  - Widely used today
  - Logic controlled by look-up tables (programmable using bit files loaded into the device)
  - Consist of programmable combinational and sequential logic elements and a programmable interconnection
  - Intellectual property designs available of specific processors and other functions (for SoC)
  - Some versions incorporate a microcontroller core (also for SoC)

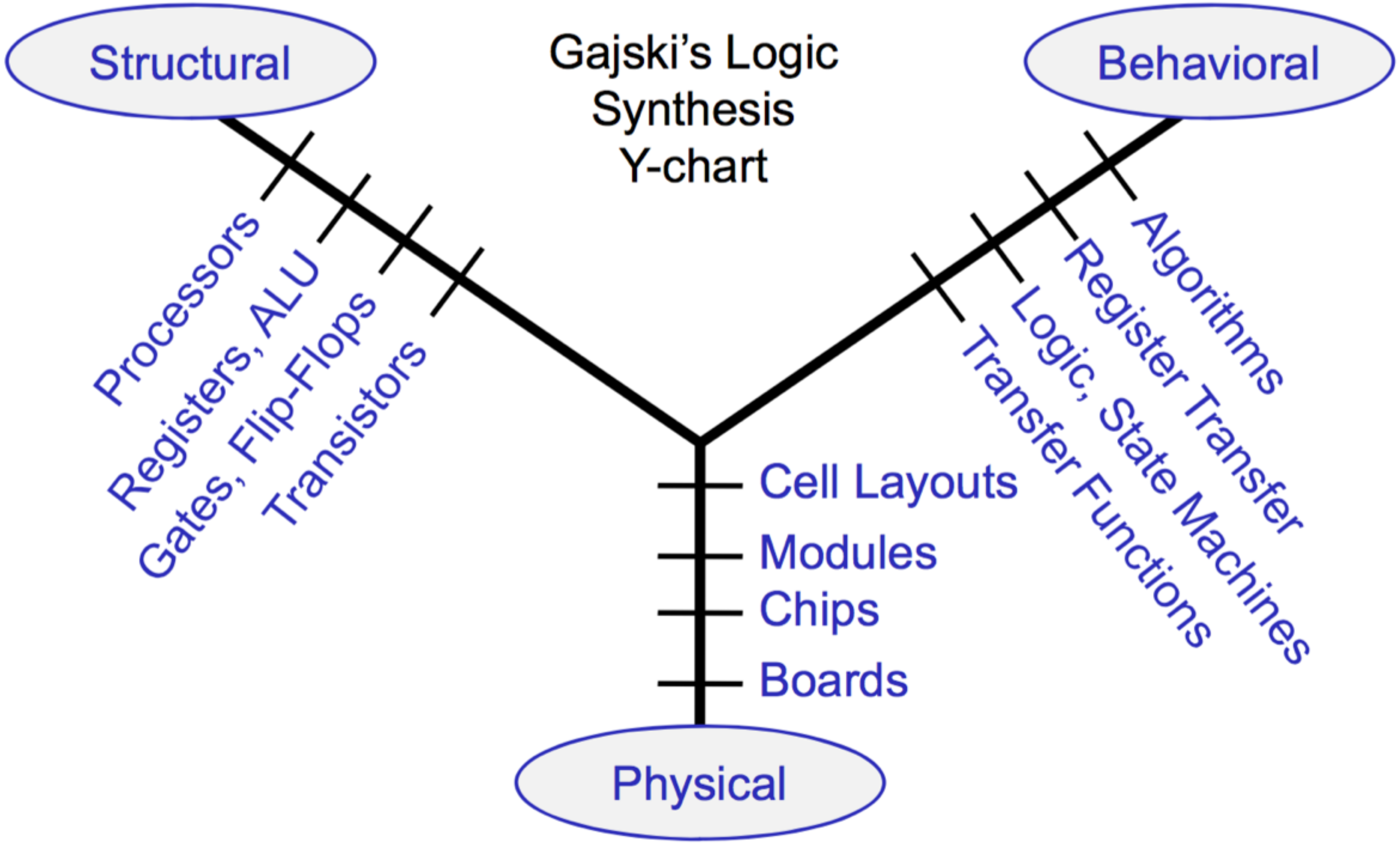
Custom-Designed Embedded Hardware

- Application-specific integrated circuits (ASICs) offer the best performance, but the economics are extreme
  - Extremely high non-recurring engineering (NRE) cost for design and manufacturing set up (often above $1M)
  - Recurring cost per device can be extremely low (often measured in cents rather than dollars)
- Thus, ASICs are suitable for extremely high-volume products
- Design challenge may be reduced by the use of “IP” designs for parts of the system, such as a microcontroller core or common function (e.g., for communication or multimedia)

Designing Embedded Hardware

- Three categories of tasks for designing embedded hardware
  - Behavioral design – what does the system do?
  - Structural design – what components and subsystems are used to realize the desired behavior
  - Physical design – how are the components realized as a physical system
- Models and design and analysis tools support these different types of tasks

Designing Embedded Hardware (cont'd)



CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Describe alternatives for embedded systems hardware platforms, including configurable hardware and custom-designed hardware

If you have any difficulties, please review the lecture video before continuing.

Modeling HW and HW/SW Systems

- Models are important for designing embedded hardware (HW) and embedded hardware/software (HW/SW) systems
- Type of models and languages
  - Hardware description languages (HDL) – for HW system design
  - System design languages – specifically to support HW/SW system co-design
- Examples
  - VHSIC* Hardware Design Language (VHDL)
  - Verilog
  - System-C
  - SpecC

(* Very High-Speed Integrated Circuit)

Embedded Software Platforms

- Embedded applications may be built on:
  - Hardware only – for simple applications
  - A run-time executive – simple system software
  - An operating system – often with special properties

Embedded Operating Systems

- Reduced "footprint"
  - Embedded systems need to use a minimal amount of memory for program storage
  - An embedded OS may be configured to include only those parts needed for a particular application
  - Example: remove code for TCP/IP if the application does not use the TCP/IP protocol stack
- Memory organization and management
  - Programs are usually stored in read-only memory (not in read/write memory as in a general-purpose system)
  - Systems are usually disk-less
  - Memory resources are precious

Embedded Operating Systems (cont'd)

- An embedded OS interacts extensively with hardware
  - Rich structure for polling and interrupts
  - Mechanisms to reduce interrupt latency
  - Hierarchy of interrupts and nested servicing of interrupts
- Many applications require "real-time" processing
  - Certain tasks must be done within certain deadlines ("hard" or "soft")
  - A real-time operating system (RTOS) has scheduling facilities to manage priorities and attempt to meet deadlines

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Describe key features of software for embedded systems

If you have any difficulties, please review the lecture video before continuing.

Summary

- General-purpose computing is only one (small) part of the overall computing industry; embedded computing is already very important and is becoming more important
- Cost, form factor, energy use, and other requirements differ significantly for embedded systems and lead to different design tradeoffs than in general-purpose systems
- Microcontrollers, system on a chip (SoC), configurable hardware (e.g., FPGAs), and application-specific integrated circuits (ASICs) are used for embedded hardware platforms
- Reduced “footprint,” memory organization, interrupt management, and real-time operation are all features of embedded software platforms

MODULE 14: Selected Topics 2

Lecture 14.1
Embedded Systems

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech

VirginiaTech
Invent the Future®