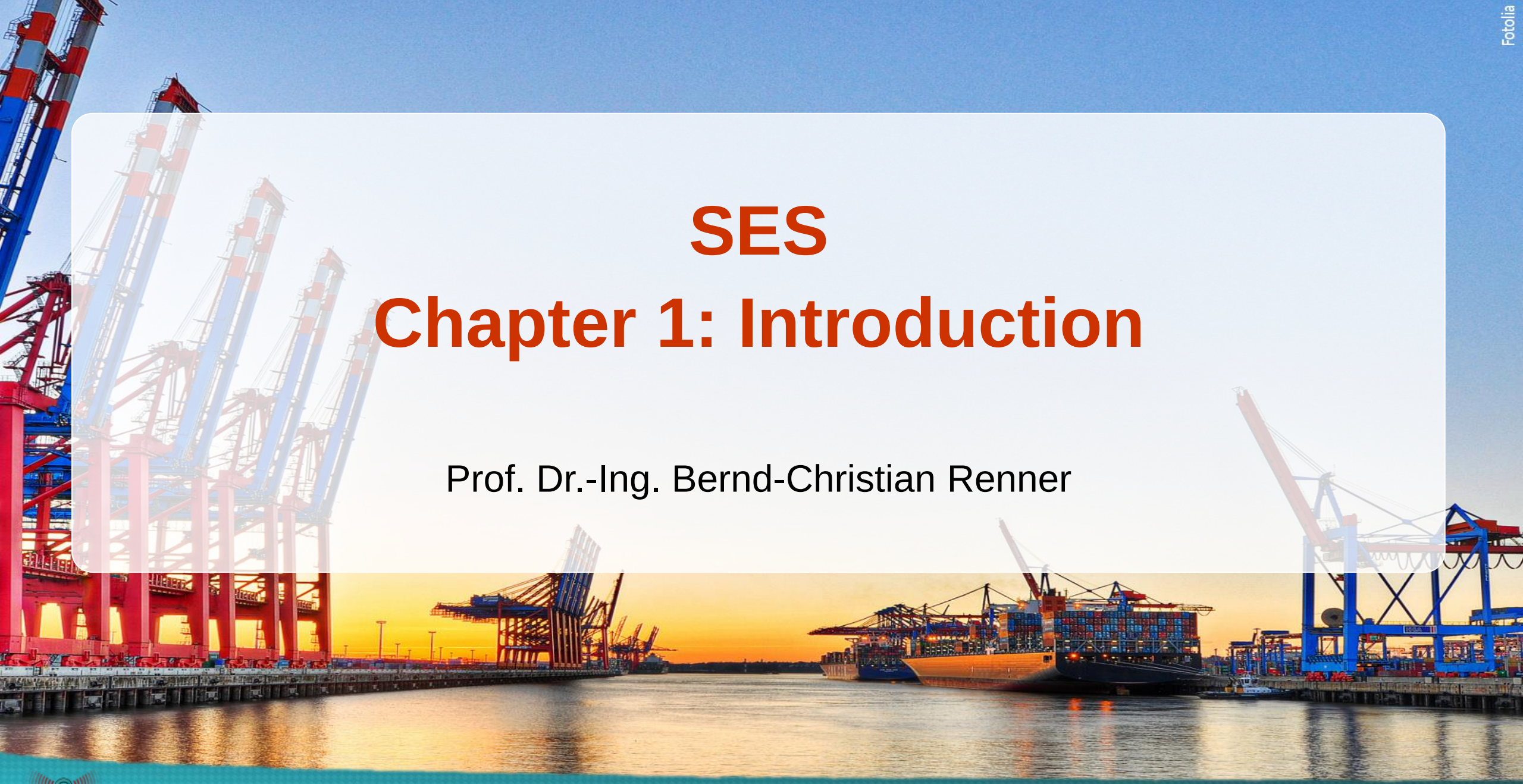


SES
Chapter 1: Introduction

Prof. Dr.-Ing. Bernd-Christian Renner

Institute smartPORT
Hamburg University of Technology
TUHH

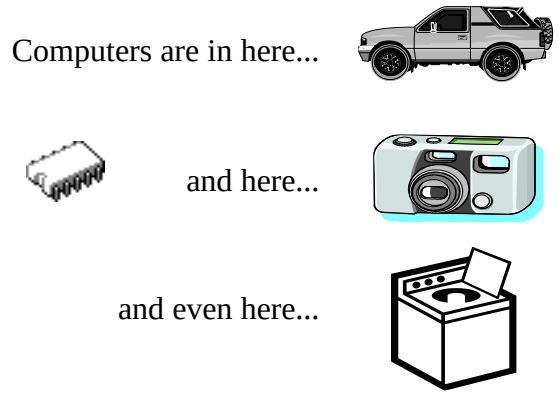
Contents

1

Overview

Embedded Computing Systems

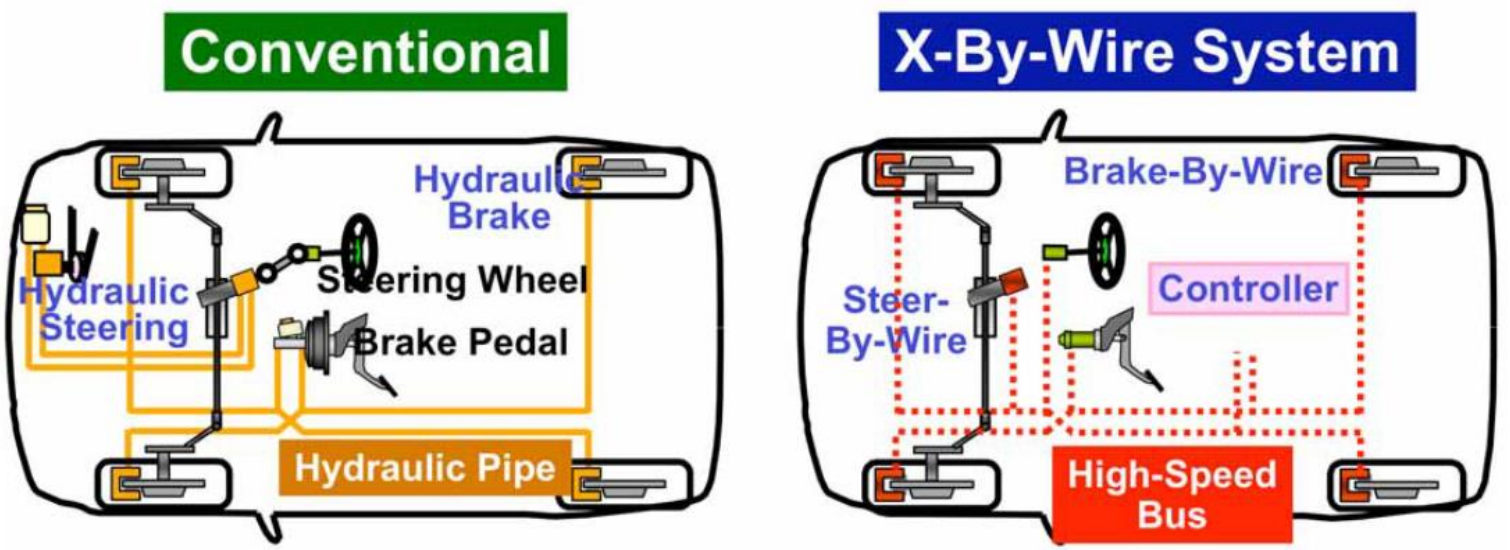
- Computing systems embedded into a larger product interacting with outside world
- Nearly any computing system other than a desktop computer
- Billions of units produced yearly, versus millions of desktop units
- Perhaps 50 per household and per automobile



Sources of images (if not specified otherwise):
Embedded System Design, F. Vahid and T. Givargis

Characteristics of Embedded Systems

- Consist of small, computerized parts within a larger device that serves a more general purpose
- Are designed to do some specific task, rather than be a general-purpose computer for multiple tasks
- Can have real-time performance constraints that must be met, for reasons such as safety and usability



Source: Hitachi Res. Lab.

Characteristics of Embedded Systems

- **Timeliness**
  Reaction time is specified by environment
  - Modern processors used in PCs or laptops **cannot** estimate duration of computation due to caching, dynamic dispatch, branch prediction
- **Concurrency**
  Embedded systems must simultaneously react to stimulus from several sensors, and retain timely control over actuators
  - Classic tools for managing concurrency (threads, processes, semaphores) are not directly suitable for embedded software
- **Liveness**
  Programs must not terminate or block waiting for events that will never occur
- **Dynamic properties**
  Classic interfaces do not describe dynamic properties such as
  - method init() must be called at least 5 ms before the fire() method
  - method x() must be invoked every 10 ms

Classification of Computing Systems

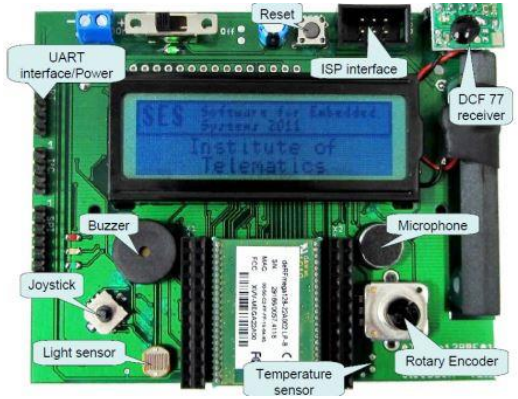
- **Transformational system**
  Input precedes processing, processing precedes output, after which the program terminates (data driven)
  - Example: Accounting, tax calculation
- **Interactive system**
  React with environment at their own speed, they can delay system execution, best effort style (user driven)
  - Example: Database server, Web-Server
- **Reactive System**
  React continuously to their environment at the speed of environment, they do not terminate (control/event driven)
  - Example: Engine control

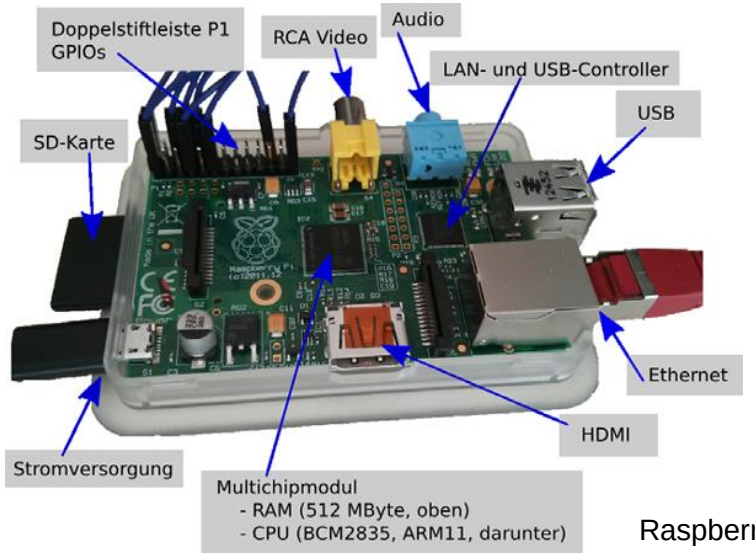
Real-time Computing

- Real-time computing
  - Hardware/software systems that are subject to a "real-time constraint“, i.e., operational deadlines for reaction to event
  - A real-time deadline must be met, regardless of system load
    - Example: Anti-lock brakes on a car: Real-time constraint is the time in which brakes must be released to prevent wheel from locking
- By contrast, a non-real-time system is one for which there is no deadline, even if fast response or high performance is desired or preferred
- Not all embedded systems are real-time systems

Types of Embedded Systems

- Deeply embedded systems (this course)
  - A few tasks
  - Simple hardware
  - 8 or 16 Bit microcontroller
  - Single program
- Open embedded systems
  - Several complex tasks
  - 32 Bit, multicore
  - Embedded operating system

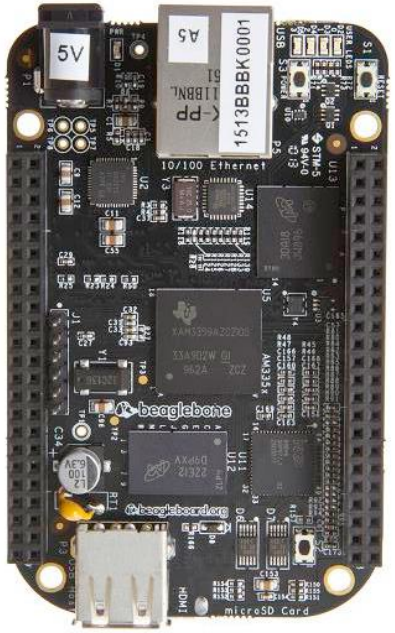




Raspberry Pi

Cheap Single-Board Computers Available

Raspberry Pi 2



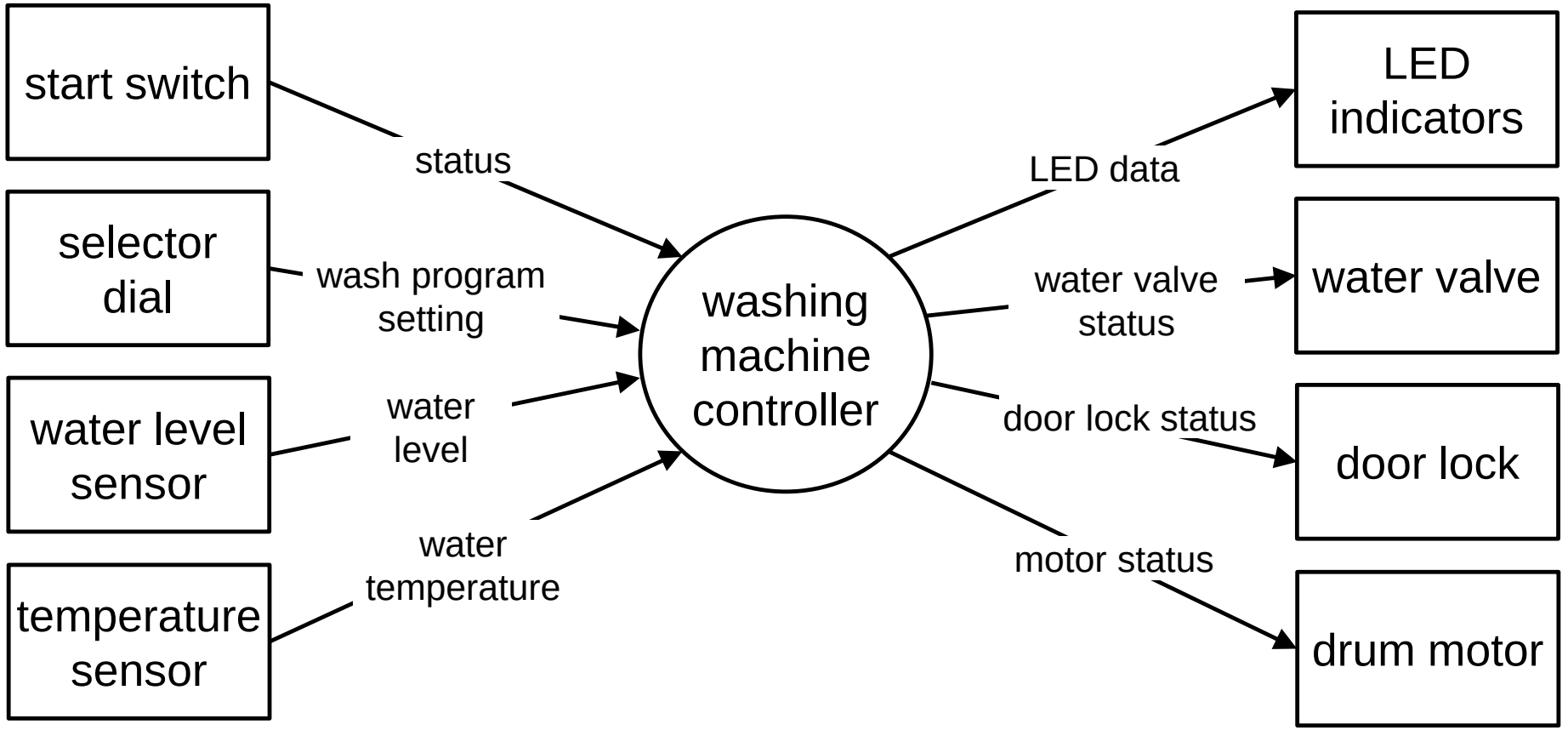
BeagleBoard

Arduino

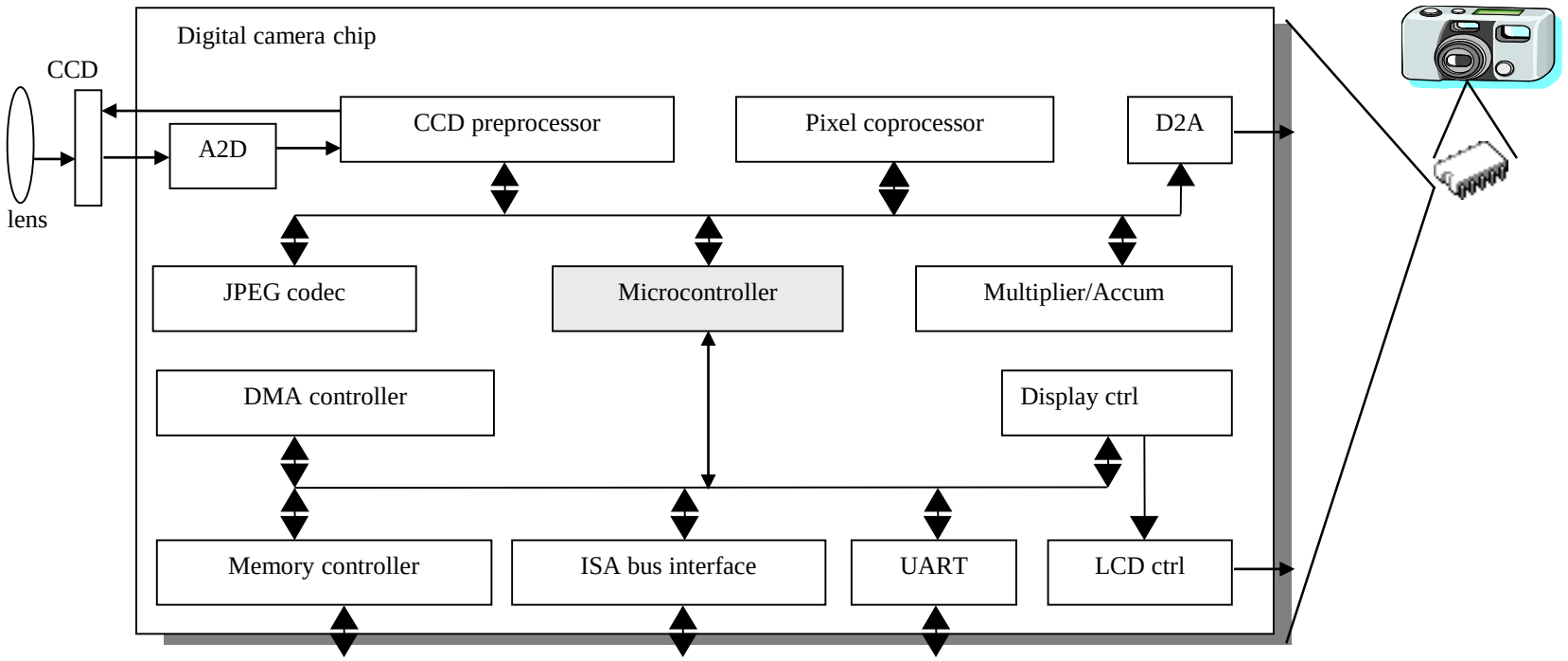
Summary

- Embedded systems are
  - Single-functioned
    - Execute a single program, repeatedly
  - Tightly-constrained
    - Low cost, low power, small, etc.
  - Reactive and real-time
    - Continually react to changes in system's environment
    - Must compute certain results in real-time without delay
- Predictability is key property
- Correctness is even more important than usual
  - "correctness" is not an abstract concept
  - "but I assumed that the hardware worked correctly" is no excuse
    - Over a long time and over a large range of conditions, it simply doesn't

Embedded System I: Washing Machine



Embedded System II: Digital Camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Embedded System III: A Coffee Maker

2

Safety & Embedded Software

Fatal Failure

- Patriot Missile Failure (Feb. 25, 1991)
  - A Patriot missile defense system operating in Saudi Arabia (operation desert storm) failed to track and intercept an incoming Scud which subsequently hit an US army barrack, killing 28 soldiers, 98 wounded
  - Cause:
    - Inaccurate calculation of time since boot due to **computer arithmetic errors**. Time in 100 ms as measured by system's internal clock was multiplied by 1/10 to produce time in seconds
    - Calculation was performed using a 24 bit fixed point register. In particular, value 1/10, which has a non-terminating binary expansion, was chopped at 24 bits after radix point
    - Small chopping error, when multiplied by large number led to a significant error
    - During accident this error was so large that Scud travelled over 500 m in this time

Safety & Embedded Software

- Therac-25 radiation therapy machine
  - Involved in at least six accidents between 1985 and 1987, in which patients were given massive overdoses of radiation
  - Cause: Software interlock could fail due to a race condition, then a one-byte counter in a testing routine frequently overflowed
- Other incidents
  - Honda recalls nearly 350K Odyssey minivans over unintended braking
  - Software fault causes 4 million GM cars to be affected by an issue that prevents airbags from deploying.
  - Boing 737 Max crashes, software problem (?)
- Are self-driving cars safe?

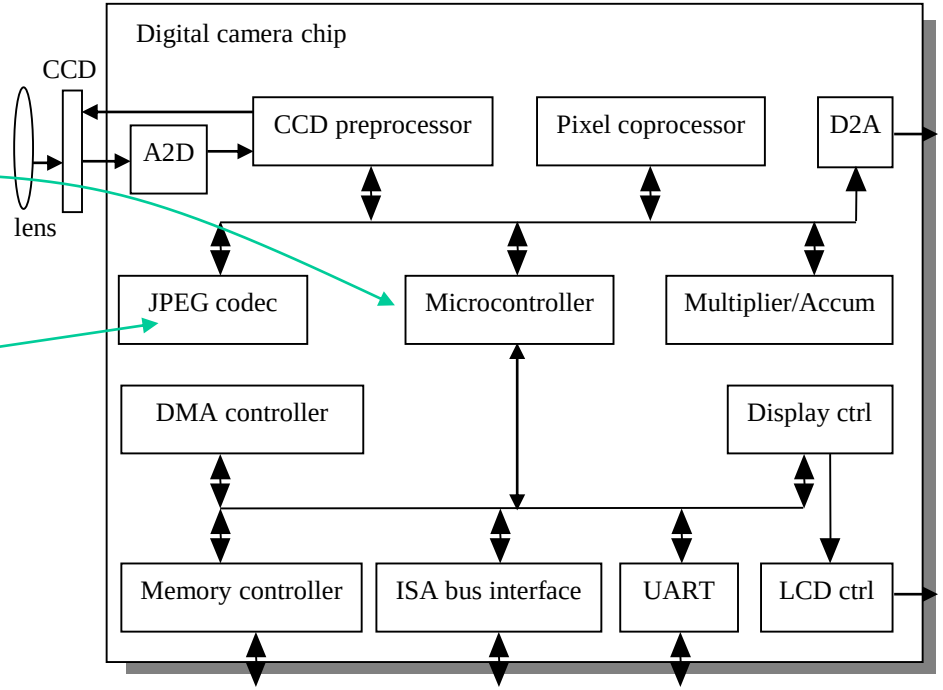
Source: Google

3

Processor Technology

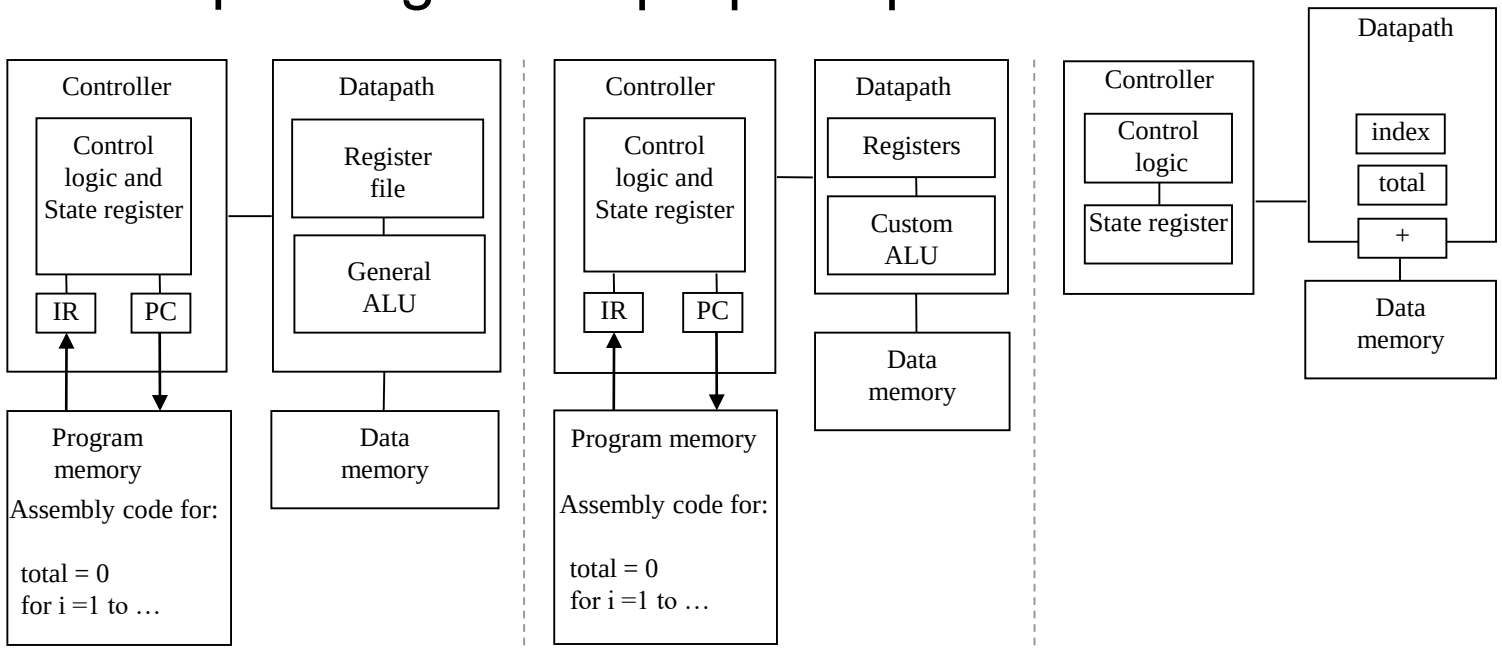
Architecture

- Processor
  - Digital circuit that performs computation tasks
  - Controller and datapath
  - General-purpose: variety of computation tasks
  - Single-purpose: one particular computation task
  - Custom single-purpose: non-standard task
- A custom single-purpose processor may be
  - Fast, small, low power
  - But, high NRE costs, longer time-to-market, less flexible



Processor Technology

- Architecture of computation engine used to implement a system's desired functionality
- Processor does not have to be programmable
  - "Processor" not equal to general-purpose processor

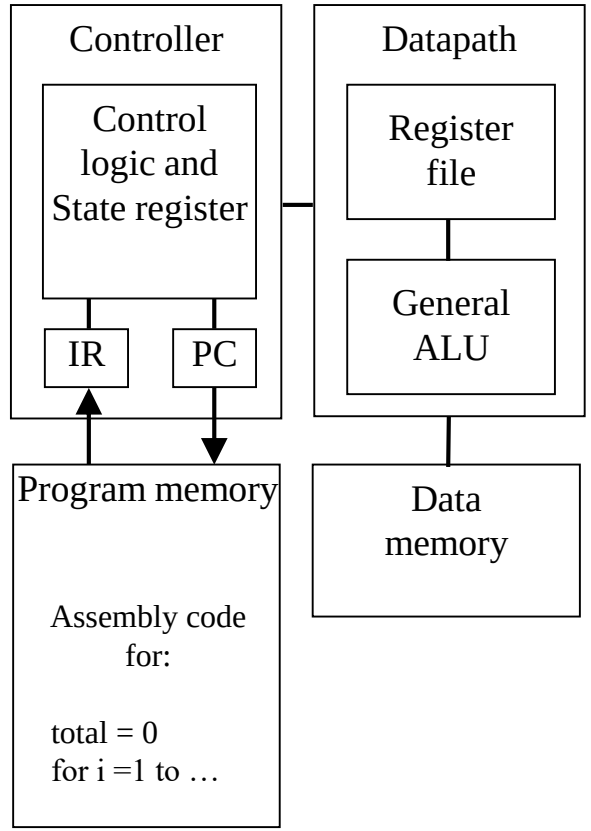


Processor Technology

- Datapath:
  - collection of functional units, such as ALUs or multipliers, that perform data processing operations
- Controller:
  - regulates the interaction between the datapath and memory
- Register file:
  - array of processor registers in controller
- Single purpose processor
  - datapath contains only essential components for program
- Application specific instruction set processor (ASIP):
  - data path is optimized, special functional units are added for common operations

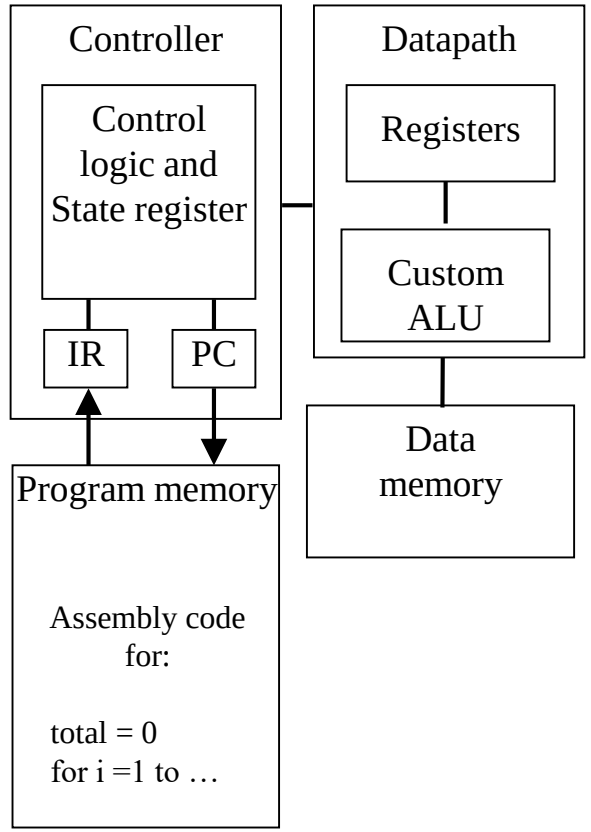
General-purpose Processors

- Programmable device used in a variety of applications
  - Also known as “microprocessor”
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Low time-to-market and NRE costs
  - High flexibility
- “Pentium” the most well-known, x86-compatible, produced by Intel
  - But there are hundreds of others!



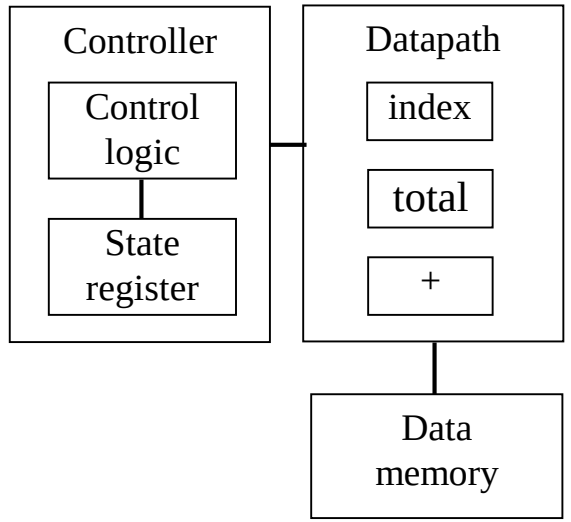
Application-specific Processors (ASIP)

- Programmable processor optimized for a particular class of applications having common characteristics
  - Compromise between general-purpose and single-purpose processors
- Features
  - Program memory
  - Optimized datapath
  - Special functional units
- Benefits
  - Some flexibility, good performance, size and power
- Example
  - Fast Fourier Transformation



Single-purpose Processors

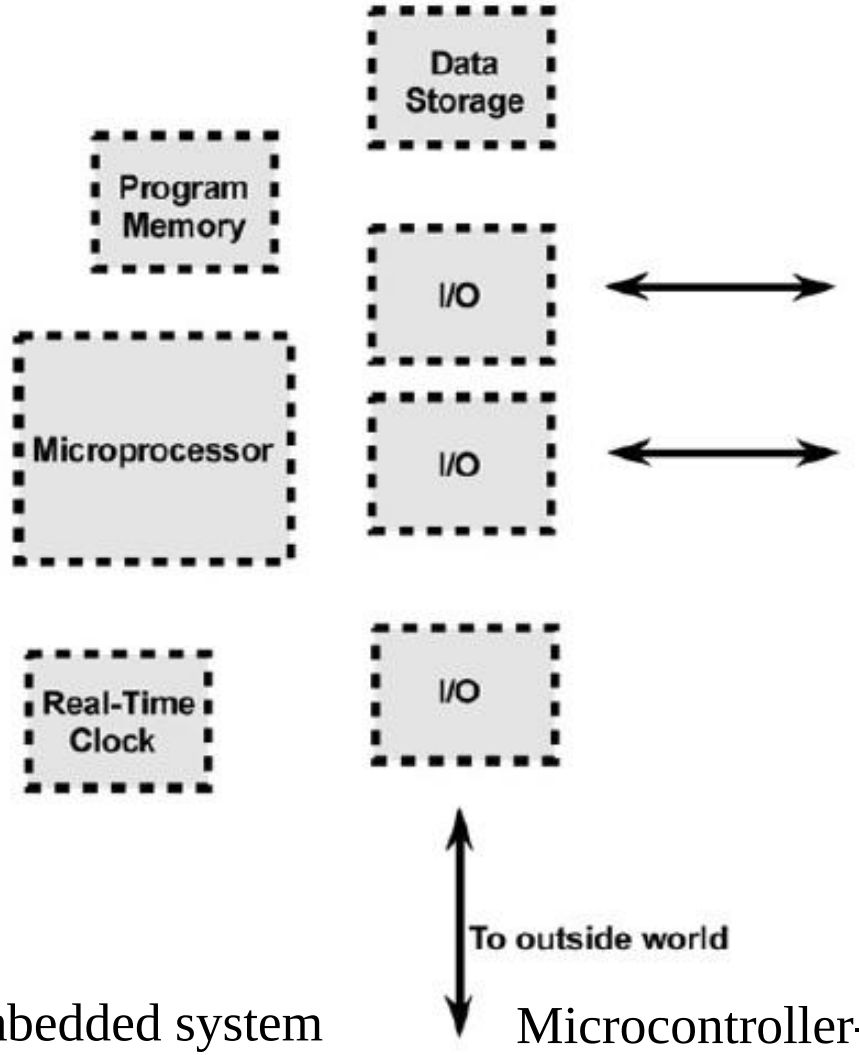
- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
- Features
  - Contains only components needed to execute a single program
  - No program memory
- Benefits
  - Fast
  - Low power
  - Small size



Microprocessor vs Microcontroller

- **Microprocessor**
  CPU without any additional peripheral or support devices
  - CPU and various I/O functions are packaged as separate ICs
- **Microcontroller**
  Designed to need a minimum complement of external parts
  - many, if not all, of I/O functions are integrated into the same package with CPU
- Most embedded systems use microcontrollers instead of microprocessors

Microprocessor vs. Microcontroller

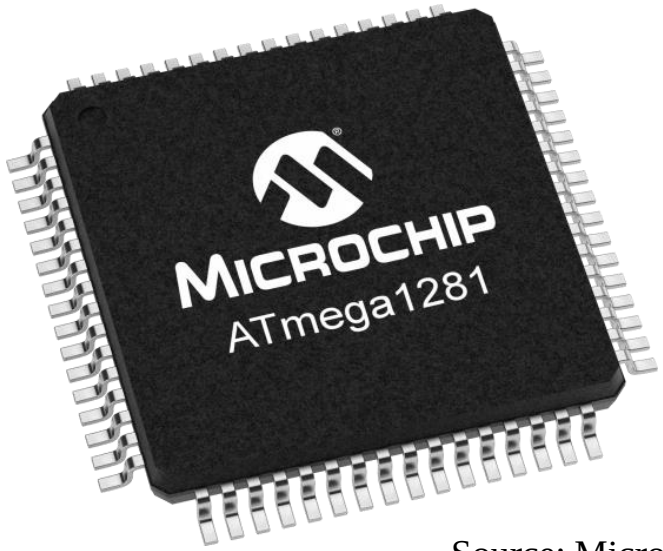


Microprocessor-based embedded system

Microcontroller-based embedded system

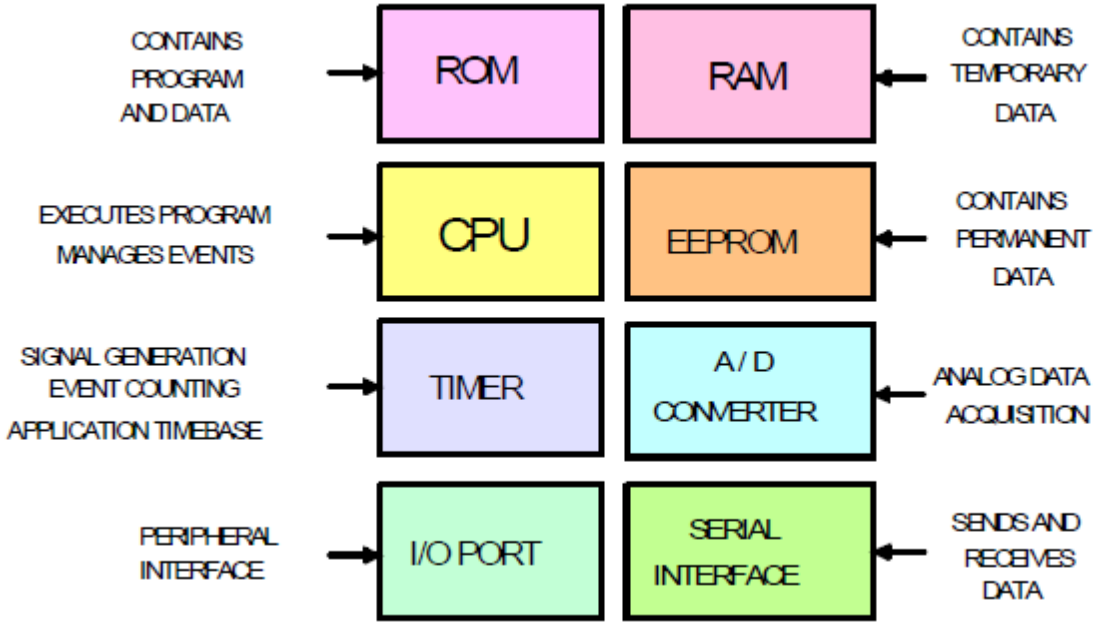
Microprocessor vs. Microcontroller

- Advantages of microcontroller's higher level of integration
  - Lower cost
    One part replaces many parts
  - More reliable
    Fewer packages, fewer interconnects
  - Better performance
    System components are optimized for their environment
  - Faster
    Signals can stay on chip
  - Lower RF signature

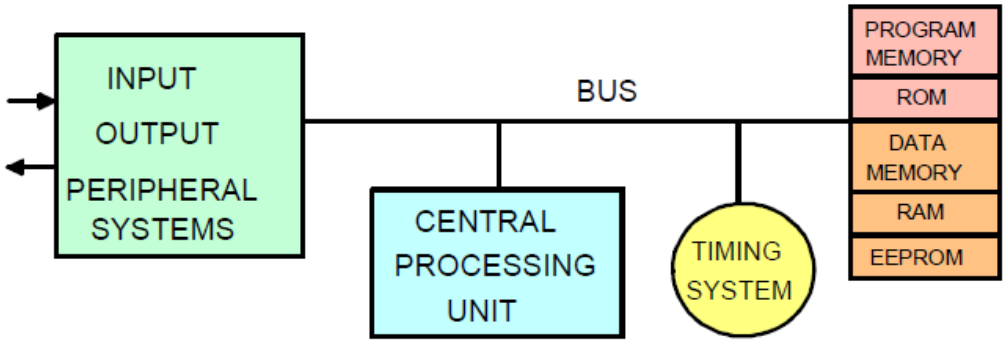


Source: Microchip

Typical Microcontroller



Block Diagram

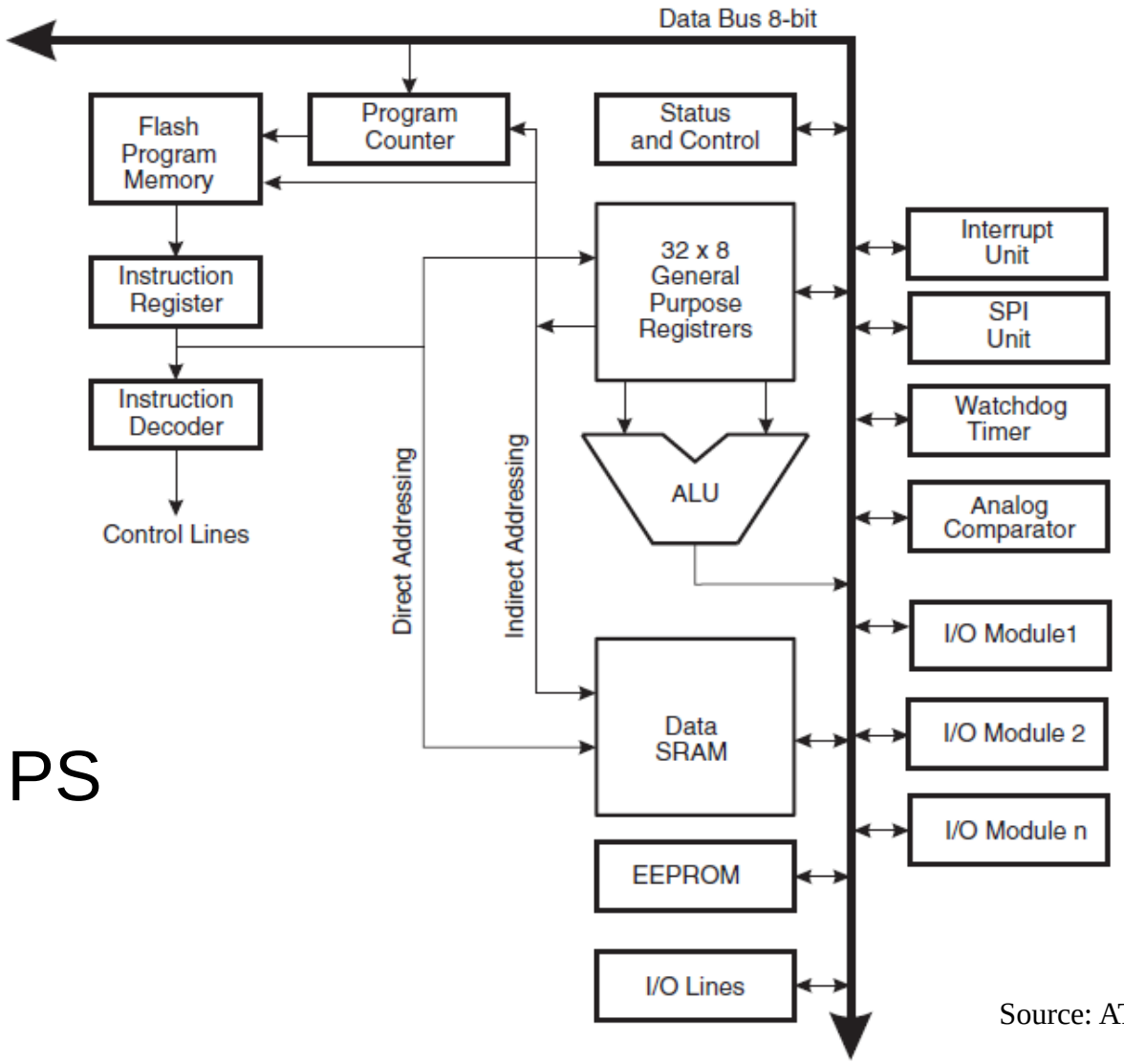


Block Organization

Source: STMicroelectronics ST AN887

ATmega 1281

- Low-power CMOS 8-bit microcontroller
- Based on the Atmel AVR enhanced RISC architecture
- Throughputs approaching 1 MIPS per MHz
- 32 general purpose working registers



Source: ATMEL

Common ASIP Digital Signal Processor

- Digital Signal Processor (DSP) is a microprocessor whose internal CPU has been optimized for use in applications involving discrete-time signal processing
  - Finite impulse response (FIR) filters
  - Fast Fourier transform (FFT)
- In addition to standard microprocessor instructions, DSPs usually support a set of specialized instructions, like multiply-and-accumulate (MAC), to perform common signal-processing computations quickly
- Some modern microprocessors offer certain DSP instructions

Digital Signal Processor

- Signal processing applications
  - Large amounts of digitized data, often streaming
  - Data transformations must be applied in real time
  - e.g., cell-phone voice filter, digital TV, music synthesizer
- DSP features
  - Several instruction execution units
  - Multiple-accumulate single-cycle instruction
    - $a \leftarrow a + (b \cdot c)$
  - Efficient vector operations – e.g., add two arrays
    - Vector ALUs, loop buffers, etc.
    - Hardware modulo addressing, allowing circular buffers to be implemented without having to constantly test for wrapping
    - Typical operations: Finite Impulse Response Filter, Discrete Fourier Transform, Discrete Cosine Transform, Convolution

Mobile base station
Massive signal processing several processing tasks per connected mobile phone, based on DSPs

Trend: Even More Customized ASIPs

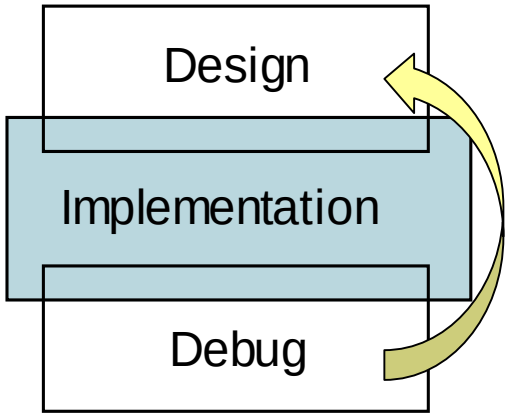
- In past, microprocessors were acquired as chips
- Today, we increasingly acquire a processor as Intellectual Property (IP)
  - e.g., synthesizable VHDL model
- Opportunity to add a custom datapath hardware and a few custom instructions, or delete a few instructions
  - Significant performance, power and size impacts
  - Problem: need compiler for customized ASIP
    - One solution: automatic compiler/debugger generation

4

The Process of Development

Development Process

- Design
  - Requirements Engineering: Functional and non-functional requirements
    - Which functions? How fast? …
  - Specification
  - Software design
    - Top down: Components, their function and relations
    - Complex functions replaced by more primitive functions
    - Refine components into more concrete realizations
- Implementation
  - Bottom up: Assemble primitives into more complex blocks
- Debug
  - Defensive programming
  - Design to make debugging easier

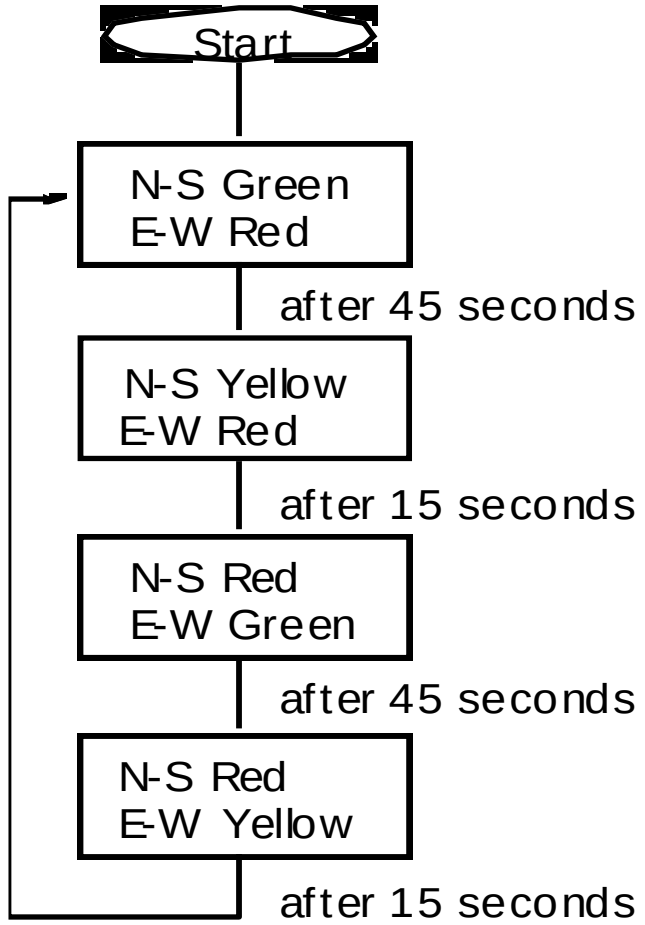


Example: Traffic Light Controller

- Functional requirements
  - Lights point in directions N, S, E, W
  - Illuminates same lights N as S and E as W
  - Cycles through sequence GREEN-YELLOW-RED
  - N-S and E-W never GREEN or YELLOW at the same time
  - Stay GREEN for 45 seconds, yellow for 15, red for 60
- Non-functional requirements
  - speed: compute changes in under 100 ms
  - power: consume less than 20 watts
  - area: implementation in less than 20 cm^2
  - cost: less than 10 € in manufacturing costs

The Art of Design

- English language specification
  - easy to write, but not precise and subject to ambiguity
- Functional description
  - more precise specification
  - flow charts, program fragments
- Structural description
  - complex components decomposed into compositions of less complex components
- Physical description
  - design in terms of most primitive building blocks



What Can Go Wrong

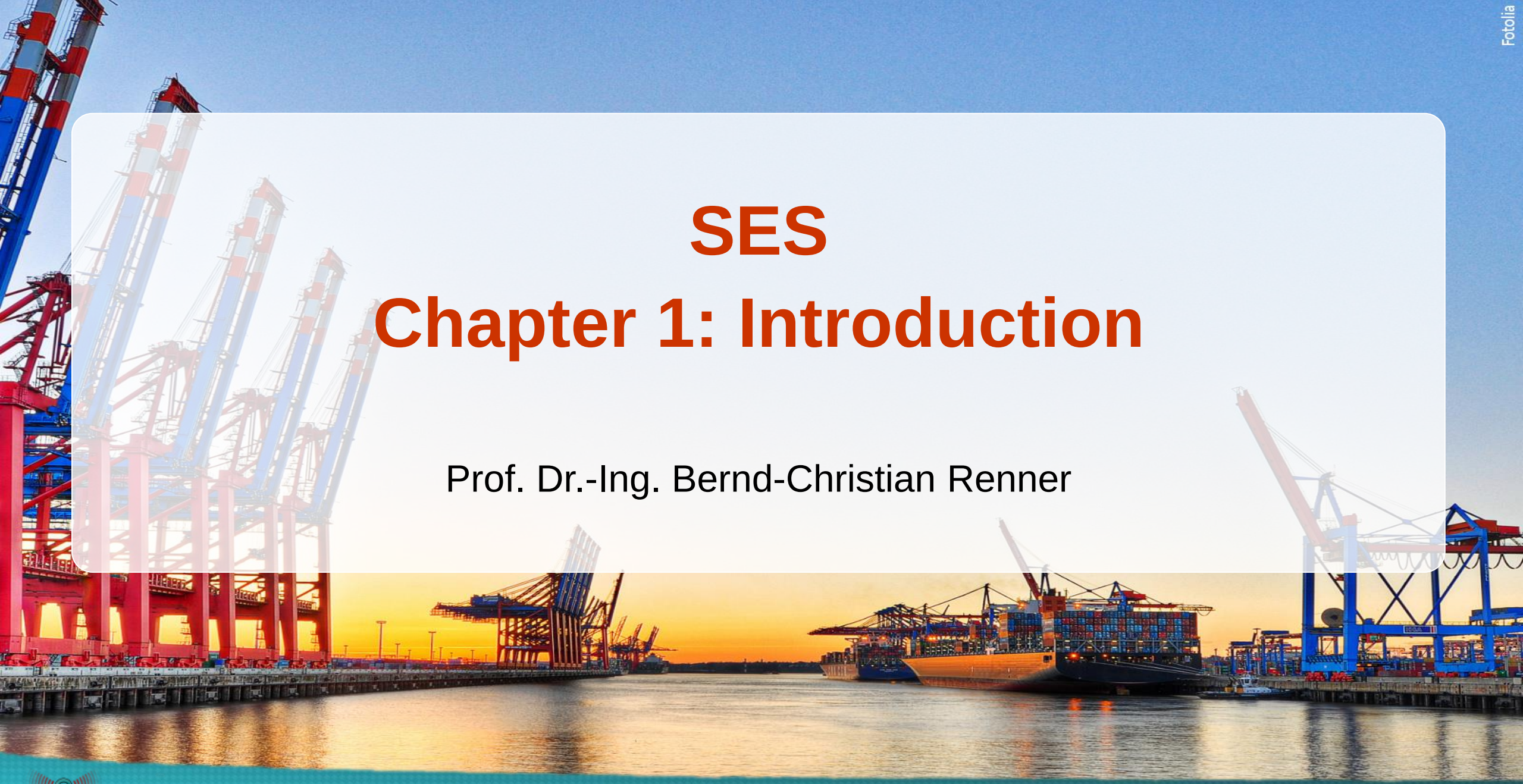
- Design Flaws
  - Design does not meet functional specification
  - Logic design is incorrect (wrong function implemented)
  - Misinterpretation or corner cases ignored
- Implementation Flaws
  - Individual modules function correctly but their compositions do not
  - Misunderstanding of interface and timing behavior
  - Misuse of hardware interface
  - Non-functional requirements are not met
- Component Flaws
  - Are all components logically correct and correctly wired?
  - Not all hardware components are guaranteed to work!

Debugging the System

- Debugging Skills:
  - Improving the testability of the design
  - Formulating a testing plan and choosing test cases
  - Hypothesizing about the cause of the problem
  - Isolating portions of the implementation for testing
  - Effective use of laboratory instruments for troubleshooting

Trends in Embedded Systems

- Increasing code size
  - average code size: 16-64KB in 1992, 64K-512KB in 1996
  - migration from hand (assembly) coding to high-level languages
- Reuse of hardware and software components
  - processors (microcontrollers, DSPs)
  - software components (drivers)
- Increasing integration and system complexity
  - integration of RF, DSP, network interfaces
  - 32-bit processors



SES
Chapter 1: Introduction

Prof. Dr.-Ing. Bernd-Christian Renner

Institute smartPORT
Hamburg University of Technology
TUHH