Introduction to Embedded Systems

Sandeep K. Shukla, IIT Kanpur
Objectives

• Introduction to embedded systems
• Embedded system components
  • Hardware
  • Software
• Embedded system Design Issues
• Trends and Directions
Communications

Hand-held GPS Units

Telematics System for Automobiles

Y. Williams  
Csci-339, Spring 2002

Slide credit Y Williams, GWU
Robotics Control

Spider robot – constructed with LEGO Mindstorms Components

Y. Williams

Csci-339, Spring 2002

Introduction to Embedded Systems

Slide credit Y Williams, GWU
More examples

Smart Toys

Slide credit Y Williams, GWU

Introduction to Embedded Systems
Cruise Missile Guidance
Relevanz der Elektronik im Automobil
Komplexität des Phaeton: CAN vernetzte Steuergeräte

Slide credit S. Kowalewski Aachen University

Introduction to Embedded Systems
Definition

• “Any sort of device which includes a programmable computer but itself is not intended to be a general-purpose computer”
  • Marilyn Wolf
Definition

Embedded System = Computers Inside a Product

Slide credit P Koopman, CMU

Introduction to Embedded Systems
Embedded systems overview

- Computing systems are everywhere
- Most of us think of “desktop” computers
  - PC’s
  - Laptops
  - Mainframes
  - Servers
- But there’s another type of computing system
  - Far more common...

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000
Embedded systems overview

- Embedded computing systems
  - Computing systems embedded within electronic devices
  - Hard to define. 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

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000*
A “short list” of embedded systems

- Anti-lock brakes
- Auto-focus cameras
- Automatic teller machines
- Automatic toll systems
- Automatic transmission
- Avionic systems
- Battery chargers
- Camcorders
- Cell phones
- Cell-phone base stations
- Cordless phones
- Cruise control
- Curbside check-in systems
- Digital cameras
- Disk drives
- Electronic card readers
- Electronic instruments
- Electronic toys/games
- Factory control
- Fax machines
- Fingerprint identifiers
- Home security systems
- Life-support systems
- Medical testing systems
- Modems
- MPEG decoders
- Network cards
- Network switches/routers
- On-board navigation
- Pagers
- Photocopiers
- Point-of-sale systems
- Portable video games
- Printers
- Satellite phones
- Scanners
- Smart ovens/dishwashers
- Speech recognizers
- Stereo systems
- Teleconferencing systems
- Televisions
- Temperature controllers
- Theft tracking systems
- TV set-top boxes
- VCR’s, DVD players
- Video game consoles
- Video phones
- Washers and dryers

And the list goes on and on

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000*
How many do we use?

- Average middle-class American home has 40 to 50 embedded processors in it
  - Microwave, washer, dryer, dishwasher, TV, VCR, stereo, hair dryer, coffee maker, remote control, humidifier, heater, toys, etc.
- Luxury cars have over 80 embedded processors
  - Brakes, steering, windows, locks, ignition, dashboard displays, transmission, mirrors, etc.
- Personal computers have over 10 embedded processors
  - Graphics accelerator, mouse, keyboard, hard-drive, CD-ROM, bus interface, network card, etc.

- Mike Schulte
Types of Embedded Systems

Four General Embedded System Types

- **General Computing**
  - Applications similar to desktop computing, but in an embedded package
  - Video games, set-top boxes, wearable computers, automatic tellers

- **Control Systems**
  - Closed-loop feedback control of real-time system
  - Vehicle engines, chemical processes, nuclear power, flight control

- **Signal Processing**
  - Computations involving large data streams
  - Radar, Sonar, video compression

- **Communication & Networking**
  - Switching and information transmission
  - Telephone system, Internet

*Slide credit P Koopman, CMU*
Cyber Physical Systems vs. Embedded Systems

Two different main application areas

User

Embedding System

Embedded System

Environment

Product automation
Embedding system = product
Examples:
• Automotive Electronics
• Avionics
• Health Care Systems

Production automation
Embedding system = production system
Examples:
• Manufacturing Control
• Chemical Process Control
• Logistics

Slide credit S. Kowalewski Aachen University

Introduction to Embedded Systems
Typical Cyber Physical Embedded Systems

- Are designed to be observed (through sensors) and control something (through actuators)

E.g. air condition senses room temperature and maintains it at set temperature via thermostat.
Embedded System Block Diagram

Slide credit Y Williams, GWU
Processors

- Microprocessors for PCs
- Embedded processors or Microcontrollers for embedded systems
  - Often with lower clock speeds
  - Integrated with memory and
  - I/O devices e.g. A/D D/A PWM CAN
  - Higher environmental specs
Microcontrollers dominates processor market

Worldwide Device Shipments (000s)

Source: IC Insights

Source and Copyright: IDCandec, 2012

Introduction to Embedded Systems
There are so many microcontrollers in the world.
Types of Embedded Processors

- **Computational micros (32- or 64-bit datapaths)**
  - CPU of workstations, PCs, or high-end portable devices (PDAs)
  - x86, PA-RISC, PowerPC, SPARC, etc.
- **Embedded general purpose micros (32-bit datapaths)**
  - Designed for a wide range of embedded applications
  - Often scaled-down version of computational micros
  - ARM, PowerPC, MIPS, x86, 68K, etc.
- **Microcontrollers (4-, 8-, or 16-bit datapaths)**
  - Integrate processing unit, memory, I/O buses, and peripherals
  - Often low-cost, high-volume devices
- **Domain-specific processors (datapath size varies greatly)**
  - Designed for a particular application domain
  - Digital signal processors, multimedia processors, graphics processors, network processors, security processors, etc.

*Slide credit - Mike Schulte*
Processor Sales Data

Microprocessor Unit Sales
All types, all markets worldwide

Source: WSTS

Slide credit - Mike Schulte

Introduction to Embedded Systems
Processor Market

Source: UBM Tech report
Trend in Programmable Logic Usage in Embedded Systems

![Bar chart showing the trend in programmable logic usage from 2010 to 2014.]

- **Yes**
  - 2010: 32%
  - 2011: 31%
  - 2012: 35%
  - 2013: 38%
  - 2014: 42%

- **No**
  - 2010: 68%
  - 2011: 69%
  - 2012: 65%
  - 2013: 62%
  - 2014: 58%

**Note:** The sample sizes (N) for each year are as follows:
- 2010: 1,540
- 2011: 1,870
- 2012: 1,669
- 2013: 2,073
- 2014: 1,295
Growing Demand

- Embedded processors account for:
  - Over 97% of total processors sold
  - Over 60% of total sales from processors
- Sales expected to increase by roughly 15% each year

Source: Gartner

Slide credit - Mike Schulte
Some common characteristics of embedded systems

• Single-functioned
  ▫ Executes a single program, repeatedly
• Tightly-constrained
  ▫ Low cost, low power, small, fast, etc.
• Reactive and real-time
  ▫ Continually reacts to changes in the system’s environment
  ▫ Must compute certain results in real-time without delay

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000
Characteristics of Embedded Systems

- Application-specific functionality – specialized for one or one class of applications
- Deadline constrained operation – system may have to perform its function(s) within specific time periods to achieve successful results
- Resource challenged – systems typically are configured with a modest set of resources to meet the performance objectives
- Power efficient – many systems are battery-powered and must conserve power to maximize the usable life of the system.
- Form factor – many systems are light weight and low volume to be used as components in host systems
- Manufacturable – usually small and inexpensive to manufacture based on the size and low complexity of the hardware.

*Slide credit Y William, GWU*
Design with focus on Application

- Technology is not the end; it is the means
  - the goal is solving (highly constrained) problems!

“IT SURE WOULD BE MORE WORK WITHOUT COMPUTERS,” SAYS A SOYBEAN FARMER WHO RELIES ON HIGH-TECH HELP FOR HARVESTING.

Harvesting beans and data. Ted Sander, 52, a farmer from Moberly, Mo., uses an onboard computer to create maps that show which plots need more fertilizer, herbicide or pesticide.

Slide credit – P Koopman, CMU

Introduction to Embedded Systems
Design Constraints

- Small Size, Low Weight
  - Hand-held electronics
  - Transportation applications -- weight costs money
- Low Power
  - Battery power for 8+ hours (laptops often last only 2 hours)
  - Limited cooling may limit power even if AC power available
- Harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
- Safety-critical operation
  - Must function correctly
  - Must not function incorrectly
- Extreme cost sensitivity
  - $.05 adds up over 1,000,000 units

*Slide credit – P Koopman, CMU*
Design Challenges

• Does it really work?
  ▫ Is the specification correct?
  ▫ Does the implementation meet the spec?
  ▫ How do we test for real-time characteristics?
  ▫ How do we test on real data?

• How do we work on the system?
  ▫ Observability, controllability?
  ▫ What is our development platform?

More importantly – optimizing design metrics!!
Design Metrics

- **Common metrics**
  - **Unit cost**: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - **NRE cost (Non-Recurring Engineering cost)**: The one-time monetary cost of designing the system
  - **Size**: the physical space required by the system
  - **Performance**: the execution time or throughput of the system
  - **Power**: the amount of power consumed by the system
  - **Flexibility**: the ability to change the functionality of the system without incurring heavy NRE cost

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction*
Design Metrics

• Common metrics (continued)
  • **Time-to-prototype**: the time needed to build a working version of the system
  • **Time-to-market**: the time required to develop a system to the point that it can be released and sold to customers
  • **Maintainability**: the ability to modify the system after its initial release
  • **Correctness, safety, many more**

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction*
Trade-off in Design Metrics

- Expertise with both software and hardware is needed to optimize design metrics
  - Not just a hardware or software expert, as is common
  - A designer must be comfortable with various technologies in order to choose the best for a given application and constraints

---

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction
Time-to-market: a demanding design metric

- Time required to develop a product to the point it can be sold to customers
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction
Losses due to delayed market entry

- **Simplified revenue model**
  - Product life = 2W, peak at W
  - Time of market entry defines a triangle, representing market penetration
  - Triangle area equals revenue

- **Loss**
  - The difference between the on-time and delayed triangle areas

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction*
Other Design Considerations

• Dependability
  ▫ **Reliability**: probability of system working correctly provided that it worked at time $t=0$
  ▫ **Maintainability**: probability of system working correctly $d$ time units after error occurred. [Some systems require no maintenance throughout their operating lives (e.g. electric kettles, computer keyboards), while some may need it such as mobile phones and airplane flight control (software upgrade)]
Other Design Considerations

- **Dependability**
  - Availability: probability of system working at time $t$
  - Safety
  - Security

Basically, critical applications have to operate correctly at all time e.g. airplane flight control computer. This includes both **hardware and software** aspects.
Example of System Fault

Finite Precision Can Lead to Disaster

Example: Failure of Patriot Missile (1991 Feb. 25)
Source: http://www.math.psu.edu/dna/455.f96/disasters.html

American Patriot Missile battery in Dharian, Saudi Arabia, failed to intercept incoming Iraqi Scud missile.
The Scud struck an American Army barracks, killing 28.

Cause, per GAO/IMTEC-92-26 report: “software problem”
(inaccurate calculation of the time since boot)

Specifics of the problem: time in tenths of second
as measured by the system’s internal clock
was multiplied by 1/10 to get the time in seconds
Internal registers were 24 bits wide
1/10 = 0.0001 1001 1001 1001 1001 100 (chopped to 24 b)
Error ≥ 0.1100 1100 × 2⁻²³ ≥ 9.5 × 10⁻⁸
Error in 100-hr operation period
≥ 9.5 × 10⁻⁸ × 100 × 60 × 60 × 10 = 0.34 s
Distance traveled by Scud = (0.34 s) × (1676 m/s) ≥ 570 m
This put the Scud outside the Patriot’s “range gate”

Ironically, the fact that the bad time calculation
had been improved in some (but not all) code parts
contributed to the problem,
since it meant that inaccuracies did not cancel out

Slide credit B. Pahami
Other Design Considerations

- Operating environment
  Some engine Electronic Control Units (ECUs) in cars are located under the bonnets. So they have to work at high temperature, as well as dusty and wet environment.

- EMI (Electromagnetic Interference)
Real-Time Consideration

- Correct operation of real-time systems means:
  - Working correctly (functionally correct)
  - Producing outputs in time!

- i.e. correct result at the right time
Hard Real-time

• System designed to meet all deadlines
• A missed deadline is a design flaw
• For examples: ABS brake, nuclear reactor monitoring system
• System hardware (over) designed for worst-case performance
• System software rigorously tested
• Formal proofs used to guarantee timing correctness

Slide credit – T Givargis
Firm Real-time

• System designed to meet all deadlines, but occasional missed deadline is allowed
  ▫ Sometimes statistically quantified (e.g. 5% misses)
• For examples: multimedia systems
• System hardware designed for average case performance
• System software tested under average (ideal) conditions

*Slide credit – T Givargis*
Soft Real-time

• System designed to meet as many deadlines as possible
  ▫ Best effort to complete within specified time, but may be late
• For examples: network switch or router
• System hardware designed for average case performance
• System software tested under averaged (ideal) conditions

*Slide credit – T Givargis*
Real-time Systems Deadlines

**Deadlines**

- **Deadline**: maximum time before a task must complete

  - The *profit* associated with execution of a task after the deadline:
    - **Hard** deadline: negative
    - **Firm** deadline: 0 (either make it or just don’t do it)
    - **Soft** deadline: decreasing with time
Levels of System Design

- requirements
- specification
- architecture
- component design
- system integration
Traditional Embedded System Design Approach

- Decide on the hardware
- Give the chip to the software people.
- Software programmer must make software ‘fit’ on the chip and only use that hardware’s capabilities.

*Slide credit - W. McUmber, MSU*
Problems with Increased Complexity

- Systems are becoming more and more complex.
- Harder to think about total design.
- Harder to fix ‘bugs.’
- Harder to maintain systems over time.
- Therefore, the traditional development process has to change,

*Slide credit - W. McUmber, MSU*
Design with Time Constraint

- In embedded electronics, the total design cycle must decrease.
- Historically, design for automotive electronic systems takes 3-5 years to develop.
- Must be reduced to a 1-3 year development cycle.
- Must still be reliable and safe.

Possible Ways to Do

• Need to keep design process abstract for a longer period of time.
• Decomposable hierarchy (object-oriented).
• Reuse previous designs:
  ▫ When a design changes, reuse similar sections.
  ▫ Don’t throw away last year’s design and start from scratch!
• Automated verification systems.

*Slide credit - W. McUmber, MSU*
Levels of Embedded System Design

- **Specification**
  - Design productivity increases with the level of abstraction
  - The task of functional verification is very difficult at low abstraction levels

- **Implementation**
  - Efficient implementations require to exploit the low-level features of the target architecture

*Slide credit – Ingo Sander*
Design Abstraction

- Start of design process
  - overall functionality has to be understood and captured
  - system components have to be identified
  - details are not important yet
  - system shall be modeled at a high abstraction level

- Implementation Phase
  - Implementation details are important to fine-tune the design
  - Low abstraction level is needed

Slide credit – Ingo Sander

Introduction to Embedded Systems
Abstraction Levels

- It is important to work on the right level of abstraction.
- The higher the level of abstraction, the shorter the design time.
- The lower the level of abstraction, the more details can be fine-tuned.

Slide credit – Ingo Sander

Introduction to Embedded Systems
Abstraction Levels

System Level

- A/B Computer
- RADAR
- C/D
- IMU

Chip Level

- RAM
- Parallel Port
- UART
- Custom Interface

Slide credit – Ingo Sander

Introduction to Embedded Systems
Abstraction Levels

Slide credit – Ingo Sander

Introduction to Embedded Systems
Abstraction Level

Slide credit – Ingo Sander

Introduction to Embedded Systems
Hardware vs Software

- Many functions can be done by **software** on a general purpose microprocessor **OR** by **hardware** on an application specific ICs (ASICS)
- For examples: game console graphic, PWM, PID control
- Leads to Hardware/Software Co-design concept
Hardware or Software?

• Where to place functionality?
  ▫ ex: A Sort algorithm
    • Faster in hardware, but more expensive.
    • More flexible in software but slower.
    • Other examples?

• Must be able to explore these various trade-offs:
  ▫ Cost.
  ▫ Speed.
  ▫ Reliability.
  ▫ Form (size, weight, and power constraints.)
Hardware vs Software

Introduction to Embedded Systems

Slide credit - Mike Schulte

Programmability and Flexibility

Power/Performance

Embedded Application-Specific Processors

Embedded Domain-Specific Processors

Graphics Processors
DSP Processors
Network Processors

General-Purpose Processors

Workstations
Personal Computers

FFT Processors
MPEG Processors
FIR Processors

Graphics Processors
DSP Processors
Network Processors

Embedded Application-Specific Processors

Embedded Domain-Specific Processors

General-Purpose Processors

Workstations
Personal Computers
Hardware vs Software

Slide credit – Ingo Sander

Introduction to Embedded Systems
Introduction to Embedded Systems

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, but there are hundreds of others

---

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000*
Single-purpose processors

- **Digital circuit designed to execute exactly one program**
  - a.k.a. coprocessor, accelerator or peripheral

- **Features**
  - Contains only the components needed to execute a single program
  - No program memory

- **Benefits**
  - Fast
  - Low power
  - Small size

*Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000*
Application-specific processors

- **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

- **DSP**

Slide credit Vahid/Givargis, Embedded Systems Design: A Unified Hardware/Software Introduction, 2000
FPGA Architecture

FPGA layout with Configurable Logic Blocks (CLB) and I/O Blocks (IOB) (credit: Katz’s Contemporary Logic Design)

Programmable switch at wiring intersection (credit: www.wikipedia.com)

Typical CLB (credit: www.wikipedia.com)
Highly constrained products tend to use application specific processors
- Many mobile phones (power & size constrained) contain ARM chips
- Hi-Fi (high performance & time constrained) contain DSP chips
Hardware is mostly a recurring cost
- Cost proportional to number of units manufactured

Software is a “one-time” non-recurring engineering design cost (NRE)
- Paid for “only once”
  - But bug fixes may be expensive, or impossible
- Cost is related to complexity & number of functions
- Market pressures lead to feature creep

*SOFTWARE Is Not FREE!!!!!

---

**Introduction to Embedded Systems**
Disciplines Used in Embedded System Design

- The design of embedded systems draws upon several disparate disciplines in CS and EE.
- Application domain (Signal processing, ...)
- Software engineering (Programming Languages, Compilers)
  - For implementing the software components; this can be a major component of several systems.
- VLSI (computer aided) design
  - For implementing the custom and semi-custom hardware components.
- Parallel/Distributed system design
  - since many embedded systems and multiprocessors are structured as a network of communicating processors; the network may be loosely coupled or tightly coupled.
- Real-time systems (Hard- & soft- real time systems)

Slide credit – R Gupta, UC Irvine

Introduction to 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 (micro-controllers, DSPs)
  - software components (drivers)
- Increasing integration and system complexity
  - integration of RF, DSP, network interfaces
  - 32-bit processors, IO processors (I2O)

*Structured design and composition methods are essential.*

*Slide credit – R Gupta, UC Irvine*
Future Embedded Systems

- Every time I hear a far fetched idea, I can find a web page with a photo of a prototype or product
Future Embedded Systems

- **Will people adopt this other than as a toy?**
  - Will the same people who can’t set time on a VCR be able to debug their house?

- **If we can make the system readily accessible, reliable, affordable,**
  **...the possibilities are almost endless**

*Slide credit – P Koopman, CMU
Introduction to Embedded Systems*
Observations on Future Embedded Systems

• More complexity (people expect more functions and higher performance from their electronic products)
• This leads to more complex software
• Which requires better design process
• More importantly, thorough testing for safety critical systems (diagnostics codes of engine ECUs is half of its total software codes)
Research in Embedded Systems

- Hardware – to improve performance (sensors and actuators), verification, etc.
- Software – reusability, testing, verification, OS, etc.
- Network – higher connectivity between systems (e.g. smart homes link many systems together, standardised protocols, etc.
- Security – protection against attacks
- Design – improved methodology, more automation, formal verification