Chapter 1: Introduction
Outline

• Embedded systems overview
  – What are they?

• Design challenge – optimizing design metrics

• Technologies
  – Processor technologies
  – IC technologies
  – Design technologies
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...
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

Computers are in here...
and here...
and even here...
Lots more of these, though they cost a lot less each.
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
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
An embedded system example -- a 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
Design challenge – optimizing design metrics

• Obvious design goal:
  – Construct an implementation with desired functionality

• Key design challenge:
  – Simultaneously optimize numerous design metrics

• Design metric
  – A measurable feature of a system’s implementation
  – Optimizing design metrics is a key challenge
Design challenge – optimizing 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
Design challenge – optimizing 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**
Design metric competition -- improving one may worsen others

- 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
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
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
Losses due to delayed market entry (cont.)

- Area = 1/2 * base * height
  - On-time = 1/2 * 2W * W
  - Delayed = 1/2 * (W-D+W)*(W-D)
- Percentage revenue loss = \( \frac{D(3W-D)}{2W^2} \)*100%
- Try some examples
  - Lifetime 2W=52 wks, delay D=4 wks
    - \( \frac{4*(3*26 –4)/2*26^2}{} = 22\%
  - Lifetime 2W=52 wks, delay D=10 wks
    - \( \frac{10*(3*26 –10)/2*26^2}{} = 50\%
  - Delays are costly!
NRE and unit cost metrics

- Costs:
  - 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
  - Total cost = NRE cost + unit cost * # of units
  - Per-product cost = total cost / # of units
  = \( \frac{NRE\ cost}{#\ of\ units} + \text{unit cost} \)

- Example
  - NRE=$2000, unit=$100
  - For 10 units
  - total cost = $2000 + 10*$100 = $3000
  - per-product cost = \( \frac{2000}{10} + 100 = 300 \)

  Amortizing NRE cost over the units results in an additional $200 per unit
NRE and unit cost metrics

- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2

- But, must also consider time-to-market
The performance design metric

- Widely-used measure of system, widely-abused
  - Clock frequency, instructions per second – not good measures
  - Digital camera example – a user cares about how fast it processes images, not
    clock speed or instructions per second

- Latency (response time)
  - Time between task start and end
  - e.g., Camera’s A and B process images in 0.25 seconds

- Throughput
  - Tasks per second, e.g. Camera A processes 4 images per second
  - Throughput can be more than latency seems to imply due to concurrency, e.g.
    Camera B may process 8 images per second (by capturing a new image while
    previous image is being stored).

- Speedup of B over S = B’s performance / A’s performance
  - Throughput speedup = 8/4 = 2
Three key embedded system technologies

• Technology
  – A manner of accomplishing a task, especially using technical processes, methods, or knowledge

• Three key technologies for embedded systems
  – Processor technology
  – IC technology
  – Design technology
Processor technology

- The architecture of the 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

- Processors vary in their customization for the problem at hand

```
total = 0
for i = 1 to N loop
    total += M[i]
end loop
```

Desired functionality

General-purpose processor

Application-specific processor

Single-purpose processor
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
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
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

Assembly code for:
```
total = 0
for i = 1 to ...
```
IC technology

- The manner in which a digital (gate-level) implementation is mapped onto an IC
  - IC: Integrated circuit, or “chip”
  - IC technologies differ in their customization to a design
  - IC’s consist of numerous layers (perhaps 10 or more)
    - IC technologies differ with respect to who builds each layer and when
IC technology

• Three types of IC technologies
  – Full-custom/VLSI
  – Semi-custom ASIC (gate array and standard cell)
  – PLD (Programmable Logic Device)
Full-custom/VLSI

• All layers are optimized for an embedded system’s particular digital implementation
  – Placing transistors
  – Sizing transistors
  – Routing wires

• Benefits
  – Excellent performance, small size, low power

• Drawbacks
  – High NRE cost (e.g., $300k), long time-to-market
Semi-custom

• Lower layers are fully or partially built
  – Designers are left with routing of wires and maybe placing some blocks

• Benefits
  – Good performance, good size, less NRE cost than a full-custom implementation (perhaps $10k to $100k)

• Drawbacks
  – Still require weeks to months to develop
PLD (Programmable Logic Device)

- All layers already exist
  - Designers can purchase an IC
  - Connections on the IC are either created or destroyed to implement desired functionality
  - Field-Programmable Gate Array (FPGA) very popular

- Benefits
  - Low NRE costs, almost instant IC availability

- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower
Moore’s law

- The most important trend in embedded systems
  - Predicted in 1965 by Intel co-founder Gordon Moore
  
  **IC transistor capacity has doubled roughly every 18 months for the past several decades**

![Graph of IC transistor capacity over time](image)
Moore’s law

- Wow
  - This growth rate is hard to imagine, most people underestimate
  - How many ancestors do you have from 20 generations ago
    - i.e., roughly how many people alive in the 1500’s did it take to make you?
    - \(2^{20}\) = more than 1 million people
  - (This underestimation is the key to pyramid schemes!)
Graphical illustration of Moore’s law

- Something that doubles frequently grows more quickly than most people realize!
  - A 2002 chip can hold about 15,000 1981 chips inside itself
**Design Technology**

- The manner in which we convert our concept of desired system functionality into an implementation

**Compilation/Synthesis:** Automates exploration and insertion of implementation details for lower level.

**Libraries/IP:** Incorporates pre-designed implementation from lower abstraction level into higher level.

**Test/Verification:** Ensures correct functionality at each level, thus reducing costly iterations between levels.

---

**System specification**

**Behavioral specification**

**RT specification**

**Logic specification**

To final implementation

<table>
<thead>
<tr>
<th>Compilation/Synthesis</th>
<th>Libraries/IP</th>
<th>Test/Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>System synthesis</td>
<td>Hw/Sw/OS</td>
<td>Model simulat./checkers</td>
</tr>
<tr>
<td>Behavioral synthesis</td>
<td>Cores</td>
<td>Hw-Sw cosimulators</td>
</tr>
<tr>
<td>RT synthesis</td>
<td>Components</td>
<td>HDL simulators</td>
</tr>
<tr>
<td>Logic synthesis</td>
<td>Gates/Cells</td>
<td>Gate simulators</td>
</tr>
</tbody>
</table>

---

*Embedded Systems Design: A Unified Hardware/Software Introduction, (c) 2000 Vahid/Givargis*
Design productivity exponential increase

- Exponential increase over the past few decades
The co-design ladder

- **In the past:**
  - Hardware and software design technologies were very different
  - Recent maturation of synthesis enables a unified view of hardware and software
- **Hardware/software “codesign”**

*The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.*
Independence of processor and IC technologies

- Basic tradeoff
  - General vs. custom
  - With respect to processor technology or IC technology
  - The two technologies are independent
Design productivity gap

- While designer productivity has grown at an impressive rate over the past decades, the rate of improvement has not kept pace with chip capacity.
Design productivity gap

- 1981 leading edge chip required 100 designer months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30,000 designer months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M
The mythical man-month

- The situation is even worse than the productivity gap indicates
- In theory, adding designers to team reduces project completion time
- In reality, productivity per designer decreases due to complexities of team management and communication
- In the software community, known as “the mythical man-month” (Brooks 1975)
- At some point, can actually lengthen project completion time! (“Too many cooks”)

- 1M transistors, 1 designer=5000 trans/month
- Each additional designer reduces for 100 trans/month
- So 2 designers produce 4900 trans/month each
Summary

- Embedded systems are everywhere
- Key challenge: optimization of design metrics
  - Design metrics compete with one another
- A unified view of hardware and software is necessary to improve productivity
- Three key technologies
  - Processor: general-purpose, application-specific, single-purpose
  - IC: Full-custom, semi-custom, PLD
  - Design: Compilation/synthesis, libraries/IP, test/verification