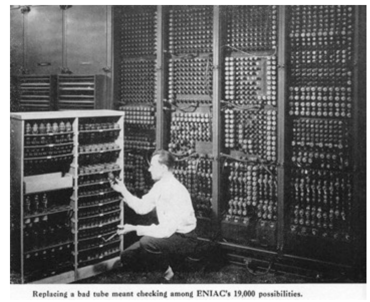
Introduction to Embedded Systems

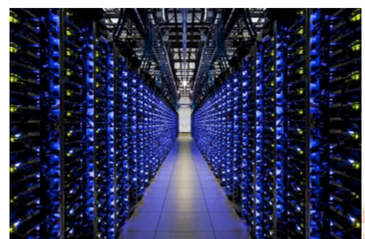
Lecture Outline

- What are ***embedded*** systems?
  - **NOT seen/noticed** unless something goes wrong
- Why do we care about embedded systems?
  - From iPhone to Boing 787, to space shuttle and satellites
- Typical Characteristics
- Constraints and design tradeoffs:
  - cost, performance, time etc.
- Design challenges & metrics
- Summary

General Computing Systems

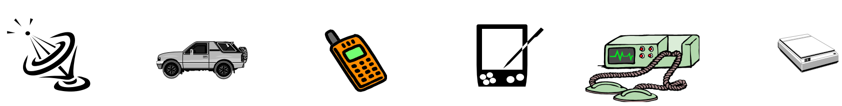
- Personal Computers (PCs)
  - Desktops
  - Laptops
- Mainframes
- Servers





Embedded (Computing) Systems

- Computing systems ***embedded*** within another devices/systems
- Computing system other than a desktop computer/ laptop
- Perhaps 50 per household and per automobile
- Billions of units produced yearly, versus millions of desktop units



Embedded Systems: Broad Range

- Pocket remote control RF transmitter
  - 100 KIPS, crush-proof, long battery life
  - Software optimized for size
- Industrial equipment controller
  - 1 MIPS, 1 MB memory; safety-critical
  - Software control loops
- Military signal processing
  - 1 GFLOPS, 1 GB/sec IO, 32 MB
  - Software for high performance

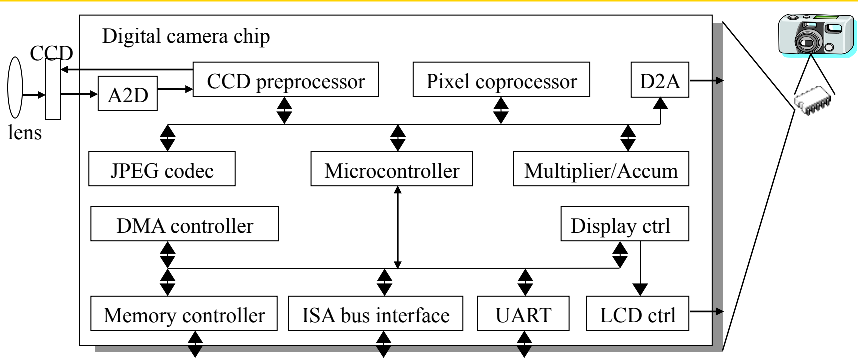
Embedded Systems: Components

- Hardware
  - Buttons, signal light and touch panel etc
  - **No** mouse, keyboard or screen
- Software
  - Dedicated software
  - Simple (or no) operating systems (compare to **Windows**)

Typical Characteristics (vs. PCs)

- **Single-Functioned**
  - A ***single*** or tightly knit set of functions
- **Tightly-Constrained**
  - Power, size, cost and reliability are important attributes
- **Reactive and real-time**
  - Continually reacts to changes in the system's environment
  - Must compute certain results in real-time without delay

Digital Camera: An Example



- Single-functioned – take photos (or movies)
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Definition: An Embedded System

Key points:

- Employs a combination of hardware & software to perform a ***single/specific*** function
- Part of a larger system (not a "computer")
- Works in a *reactive* and *time-constrained* environment.

- Software provides features and flexibility
- Hardware = {Processors, ASICs, Memory,...} is used for performance (& sometimes security)

Why are Embedded Systems Important?

- Market reasons **(dollar is important ☺)**
  - 90% of processors → "**non-computers**"
  - E.g., modern cars: tens of cpus, cost about $2,000/car
  - Market is around billions of $
- Engineering reasons
  - Devices can be controlled by a microprocessor
  - Why does a satellite need a Windows prompt ?
  - Does McDonald's POS terminal need MacOS?
- Embedded system designers
  - Broad knowledge: hardware, software, and combination of networking, control theory and signal processing

Typical Constraints: Size and Power

- Small **Size**, Low **Weight**
  - Handheld electronics: PDAs, cell phones, Laptops
  - Transportation and consumer: weight costs money
- Low **Power**
  - Longer battery life; more safe

Typical Constraints: Safety and Cost

- **Safety** critical operations
  - Must function correctly in harsh environment
  - Heat, vibration, shock
  - Power fluctuations, RF interference, lightning
  - Water, corrosion, physical abuse
  - Out space and radiations
- **Cost** sensitivity
  - **NRE (Non-Recurring Engineering cost):** One-time monetary cost of designing and developing a new system
  - **Unit cost**: the monetary cost of manufacturing each copy of the system (excluding NRE cost)

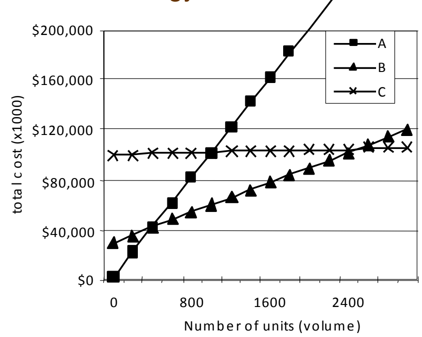
Cost: NRE and Unit Cost vs. Number

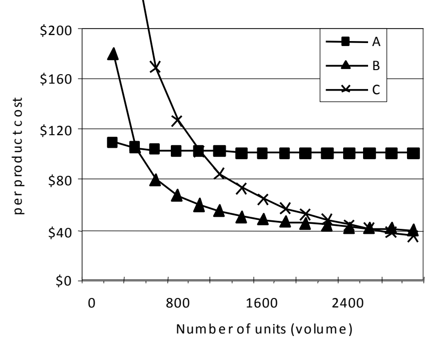
- Costs:
  - **NRE cost (Non-Recurring Engineering cost):** The one-time monetary cost of designing the system
  - **Unit cost**: the monetary cost of manufacturing each copy of the system, excluding NRE cost
  - *total cost = NRE cost + unit cost * # of units*
  - *per-product cost = total cost / # of units*
    = (NRE cost / # of units) + unit cost
- Example
  - NRE=$2000, unit=$100
  - For 10 units
    - ✓ total cost = $2000 + 10*$100 = $3000
    - ✓ per-product cost = $2000/10 + $100 = $300

Amortizing NRE cost over all the units

NRE and Unit Cost

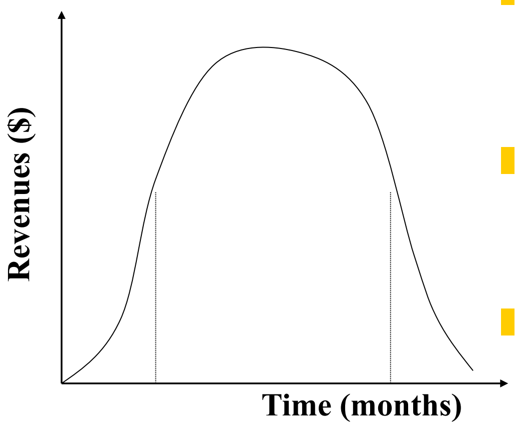
- Compare technologies by costs -- best depends on quantity
  - Technology A: NRE=$2,000, unit=$100
  - Technology B: NRE=$30,000, unit=$30
  - Technology C: NRE=$100,000, unit=$2





But, must also consider ***time-to-market***

Time-to-Market: A Demanding Metric



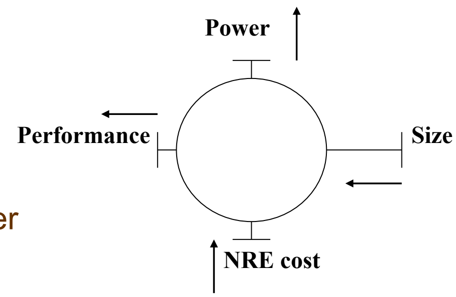
- Time required to develop a product to the point it can be sold to customers
- **Market window**
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly

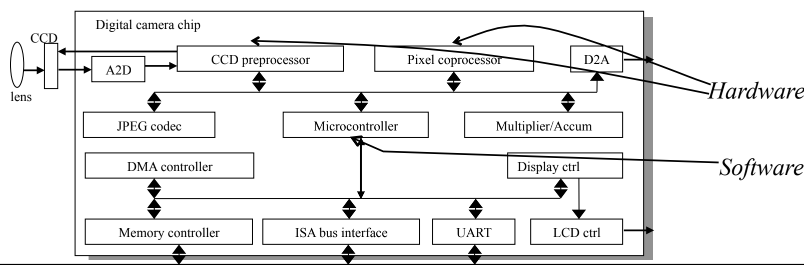
Typical Constraints (cont.)

- **Time-to-prototype**
  - The time needed to build a working version of the system
- **Performance**
  - Response/execution time
  - Throughput
- **Flexibility and maintainability**
  - Ability to change the functionality of a system without incurring heavy NRE cost

Optimization: Tradeoffs

- Improving one metric may worsen others
- Expertise with both **software and hardware** is needed to optimize design metrics
- **Various technologies** in order to choose the best for a given application and constraints





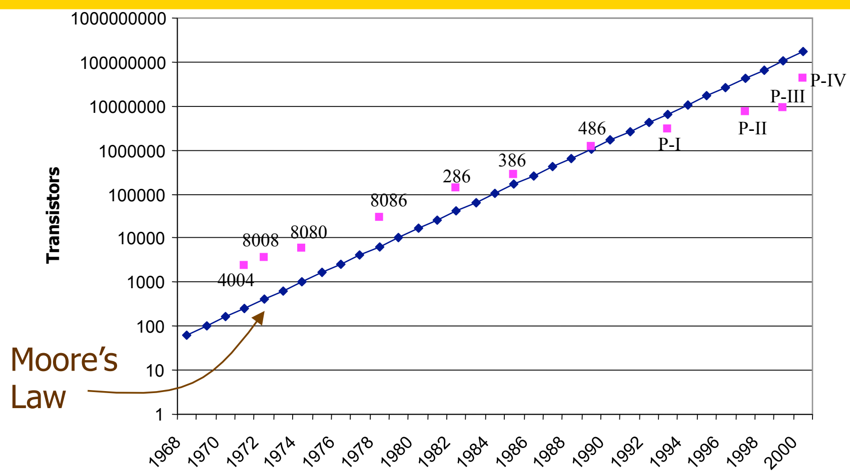
Performance Design Metrics

- Widely-used measures of system (widely-abused)
  - **Clock frequency:** instructions per second
  - E.g., digital camera – 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: improve on concurrency**
  - Tasks per second, e.g. Camera A processes 4 images/second
  - Camera B may process 8 images per second (by capturing a new image while previous image is being stored).
  - *Speedup* of B over A: S =8/4 = 2

Trends in Embedded Systems

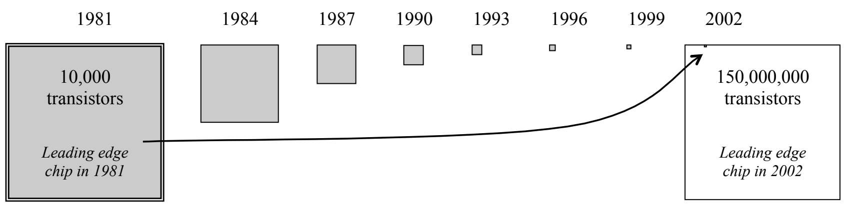
- Increasing code size
  - average code size: 16-64KB in 1992, 64K-512KB in 1996
  - migration from hand (assembly) coding to high-level languages
- Reuse of hardware and software components
  - processors (micro-controllers, DSPs)
  - software components (drivers)
  - Proprietary designs
- Increasing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

Moore's Law: Transistors on a Chip



IC transistor capacity has doubled roughly every 18 months for the past several decades

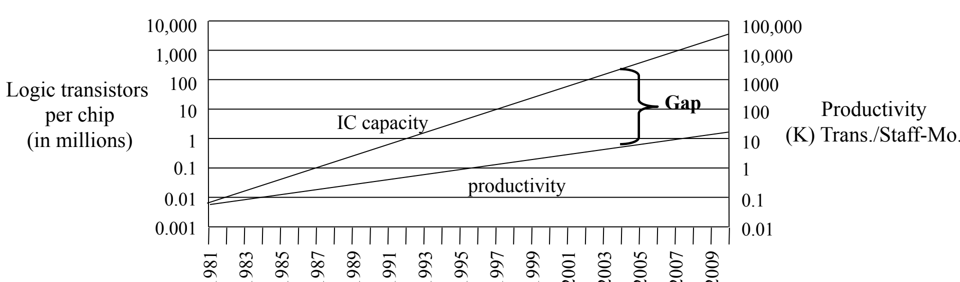
Graphical Illustration of Moore's law



- Something that doubles frequently grows more quickly than most people realize!
  - A 2002 chip can hold about 15,000 1981 chips inside itself
  - underestimation the pyramid schemes

Design Productivity Gap

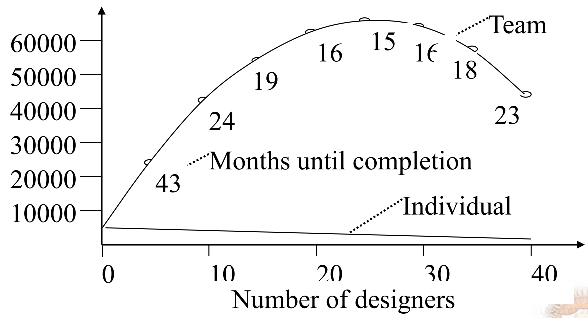
- 1981 leading edge chip required 100 designer months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30,000 designer months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M



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
- 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

- What is an embedded system?
  - More than just a computer it's a complete ***system***
- Why do we care about embedded systems?
  - From iPhone to Boing 787, to space shuttle and satellite
- Typical Characteristics
  - Single-function, tightly-constrained, real-time and reactive
- What makes embedded systems different?
  - Many constraints on designs: Size, Power, Cost, and Performance etc.
- Design challenges & metrics