INF-741: Embedded systems programming for IoT

Prof. Edson Borin

Institute of Computing

University of Campinas

Embedded Systems

Agenda

Embedded systems overview

Design metrics

Technologies

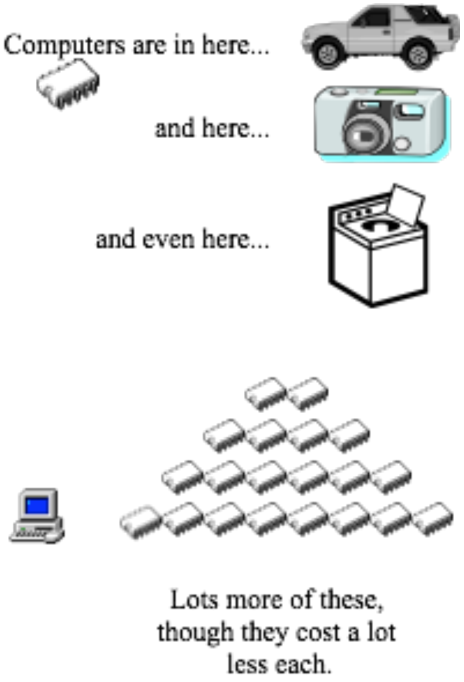
Embedded Systems Overview

- Computing systems are everywhere!
- Most of us think of “desktop” computers
  - Smartphones, Tablets, 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
- Billions of units produced yearly, versus millions of desktop units
- Perhaps 50 per household and per automobile



Embedded Systems Overview

Examples

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

WiFi routers
MPEG decoders
Network cards
Network switches
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 more...

Embedded Systems Overview

What is the definition of Embedded Systems?

Embedded Systems Overview

What is the definition of Embedded Systems?

Quiz on Moodle

Embedded Systems Overview

What is the definition of Embedded Systems?

Embedded Systems Overview

Several different definitions on books and on the web:

- An embedded system is a combination of computer hardware and software—and perhaps additional parts, either mechanical or electronic—designed to perform a dedicated function.
- Embedded systems are information processing systems embedded into enclosing products.
- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.
- An embedded system is a dedicated computer system designed for one or two specific functions. This system is embedded as a part of a complete device system that includes hardware, such as electrical and mechanical components.
- An embedded system is a computer system that does a particular task inside a machine or larger electrical system.
- An embedded system is any electronic system that uses a computer chip, but that is not a general-purpose workstation, desktop or laptop computer.
- An embedded system is a special-purpose system in which the computer is completely encapsulated by the device it controls.
- An embedded system is a computing system that does not look like a computer.

Embedded Systems Overview

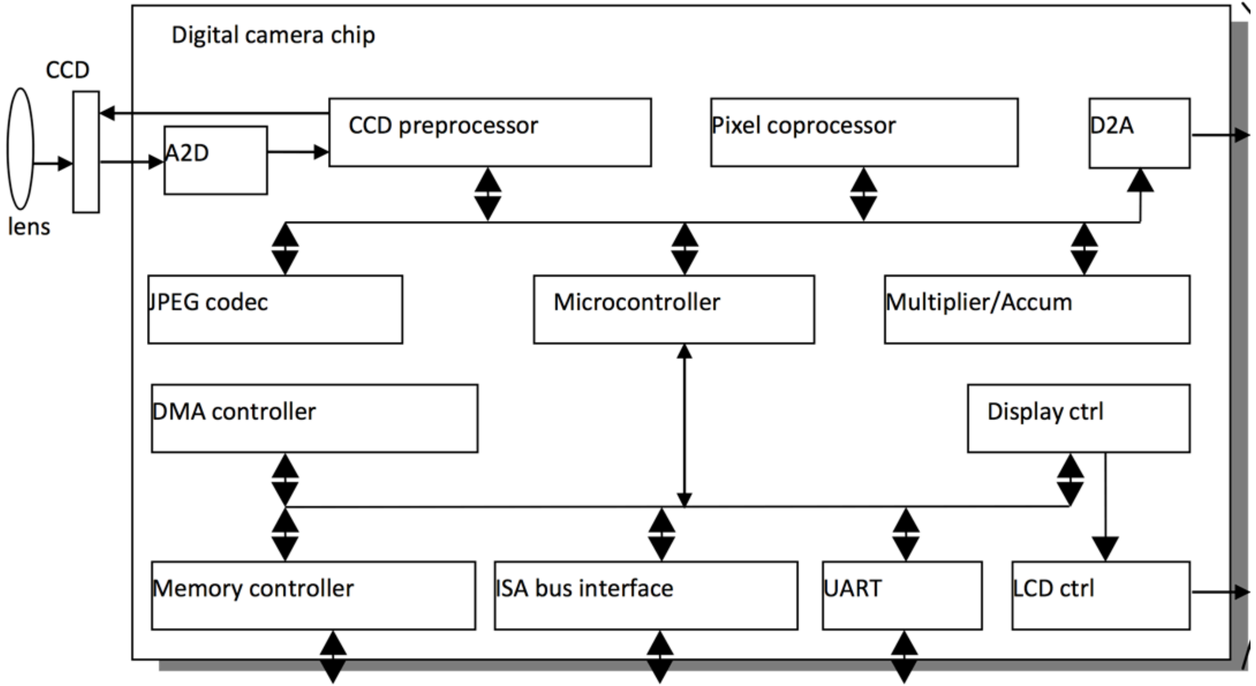
Common aspects on most of the definitions:

- **Information processing**; and
- **Dedicated** function / one or two specific function / particular task ...

Typical characteristics of embedded systems

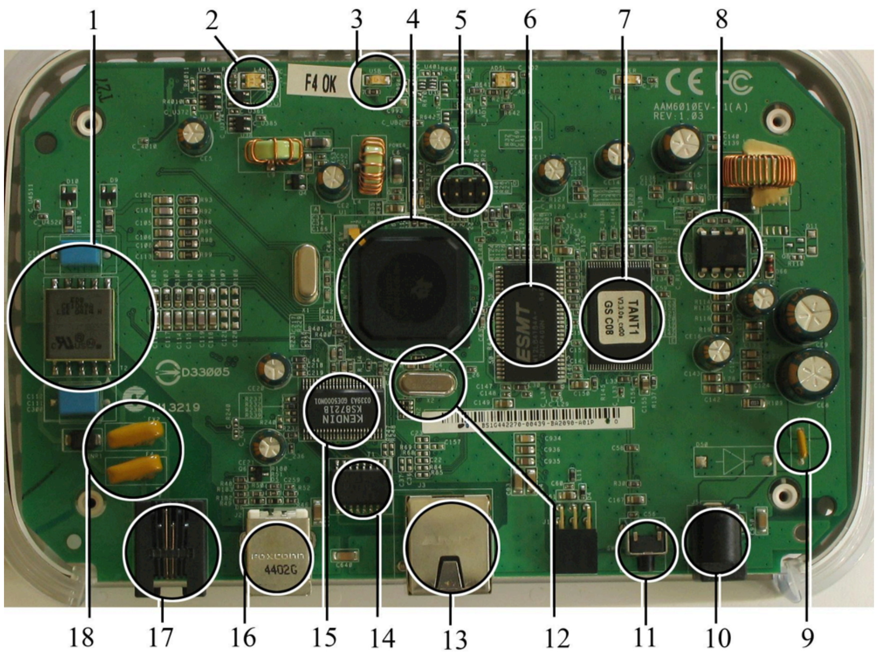
- Single-functioned
  - Executes a single program, repeatedly
- Tightly-constrained
  - Low cost, low power, small, fast, etc.
- Reactive and real-time
  - Continually reacts to changes in the system’s environment
  - Must compute certain results in real-time without delay

Embedded system example: 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 example: ADSL modem/router



1 Telephone decoupling electronics (for ADSL)
2 Multicolour LED (displaying network status)
3 Single colour LED (displaying USB status)
4 Main processor,
5 JTAG (Joint Test Action Group) test and programming port
6 RAM,
7 Flash memory: ESMT M12L64164A 8 MB chip
8 Power supply regulator
9 Main power supply fuse
10 Power connector
11 Reset button
12 Quartz crystal
13 Ethernet port
14 Ethernet transformer,
15 Delta LF8505 KS8721B Ethernet PHY transceiver
16 USB port
17 Telephone (RJ11) port
18 Telephone connector fuses

Agenda

Embedded systems overview

Design metrics

Technologies

Design Metrics

- Obvious design goal:
  - Construct an implementation with desired functionality
- Design metric
  - A measurable feature of a system’s implementation
- Key design challenge:
  - Simultaneously optimize numerous design metrics

Common Design Metrics

- **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
- **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

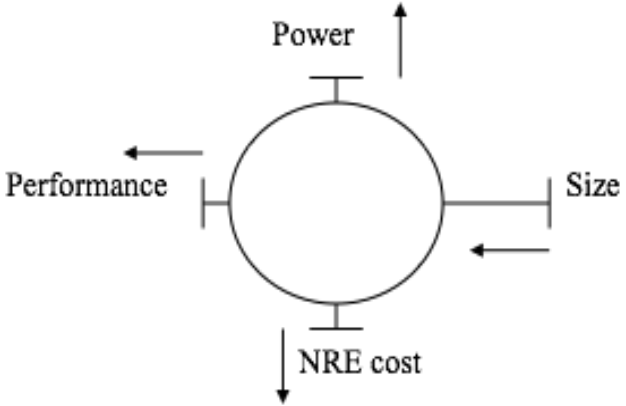
Common Design Metrics

- **Flexibility**: the ability to change the functionality of the system without incurring heavy NRE cost
- **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 design metric may affect (worsen) others

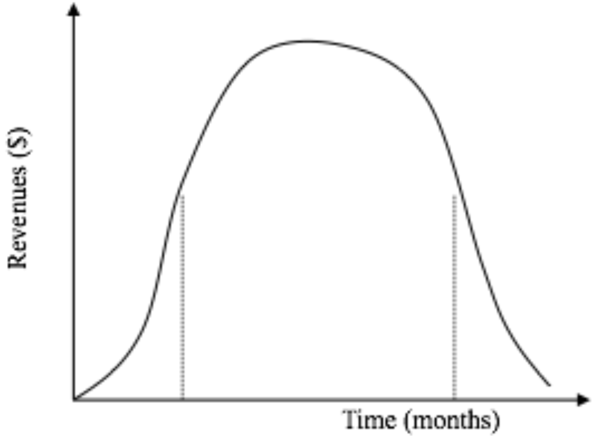
- Ex: Reducing number of transistors may improve size, however, it may affect performance.
- This phenomenon may be compared to a wheel with numerous pins. If you push one pin, such as size, then the other pins pop out.



Design Metrics: Time-to-market

Time-to-market: the time required to develop a system to the point that it can be and sold to customers

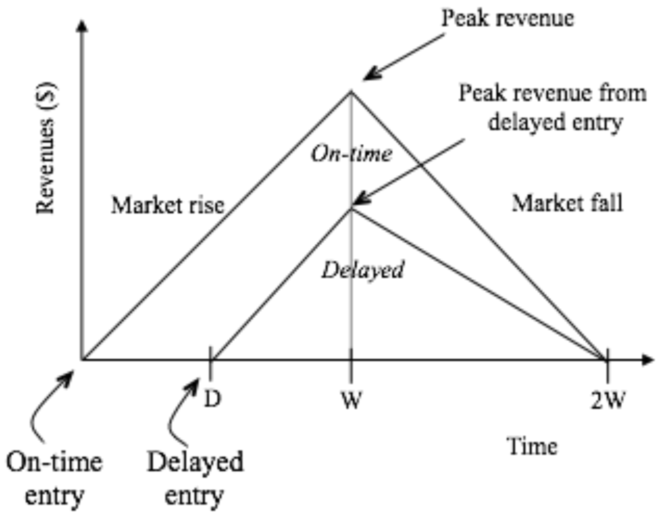
- Market window
  - Period during which the product would have highest sales
- Average time-to-market constraint is about 8 months
- Delays can be costly



Design Metrics: Time-to-market

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



Design Metrics: Time-to-market

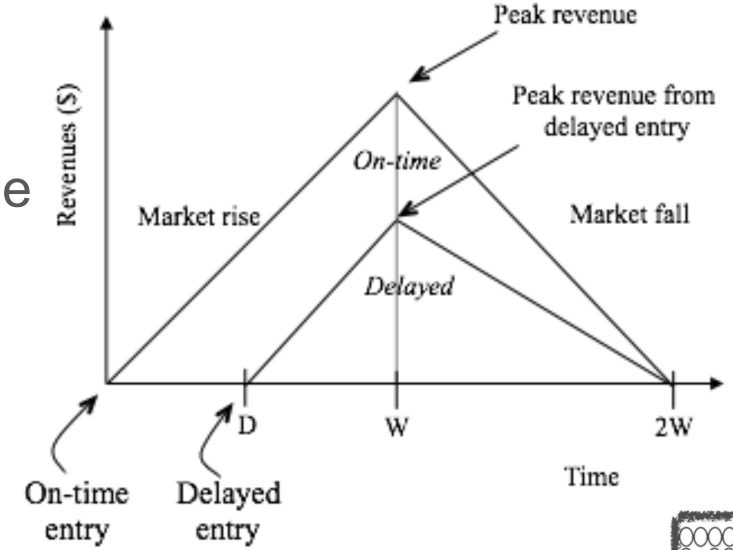
Quiz on Moodle

Assuming:

- Percentage revenue loss = ((On-time - Delayed) / On-time) * 100%
- Market rise angle = 45º

Consider a product whose lifetime is 52 weeks (W = 26). Compute the percentage revenue loss when:

A) D = 4 weeks
B) D = 10 weeks



Design Metrics: Non-Recurring Engineering cost

Design cost Manufacturing cost

Total cost = $\underbrace{\text{NRE cost}}_{}$ + unit cost * # of units

Per-production cost = Total cost / # of units

= (NRE cost / # of units) + unit cost

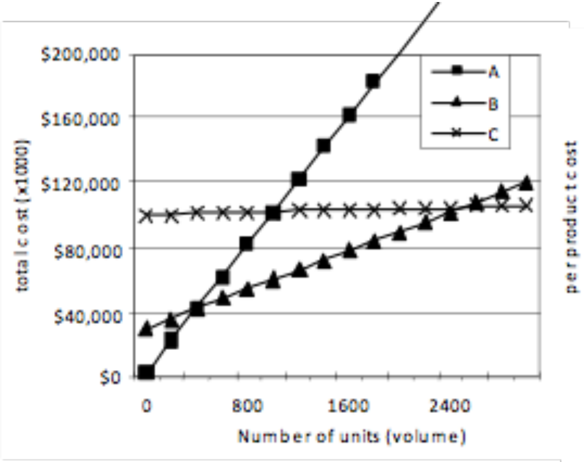
Example:

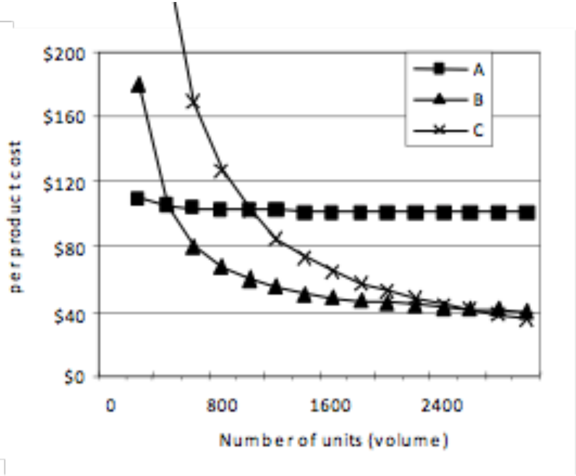
- NRE=$2000, unit=$100
- 10 units:
  - total cost = $2000 + 10*$100 = $3000
  - per-product cost = $2000/10 + $100 = $300

Design Metrics: Non-Recurring Engineering 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



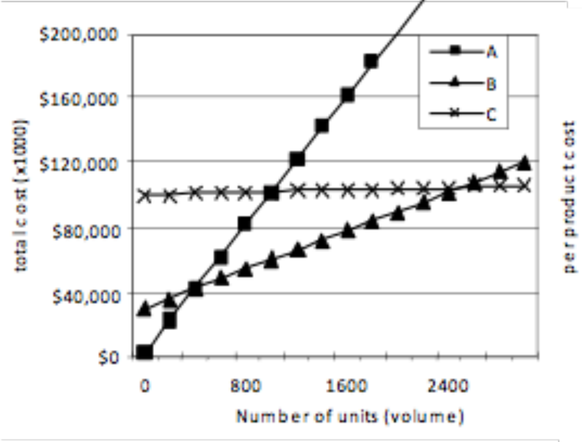


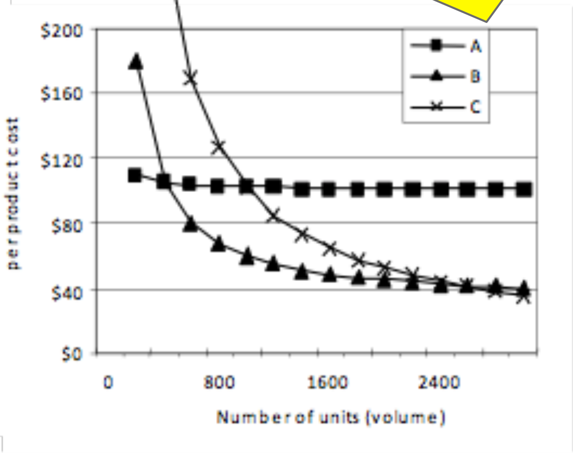
Design Metrics: Non-Recurring Engineering cost

Compare technologies by costs -- best depends

- Technology A: NRE=\$2,000, unit=\$
- Technology B: NRE=\$30,000, unit=\$
- Technology C: NRE=\$100,000, unit=\$

Technology C may look cheaper for larger volumes, but the designer must also consider the time-to-market!





Design Metrics: Performance

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

Design Metrics: Performance

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

Agenda

Embedded systems overview

Design metrics

Technologies

Technologies

Technology

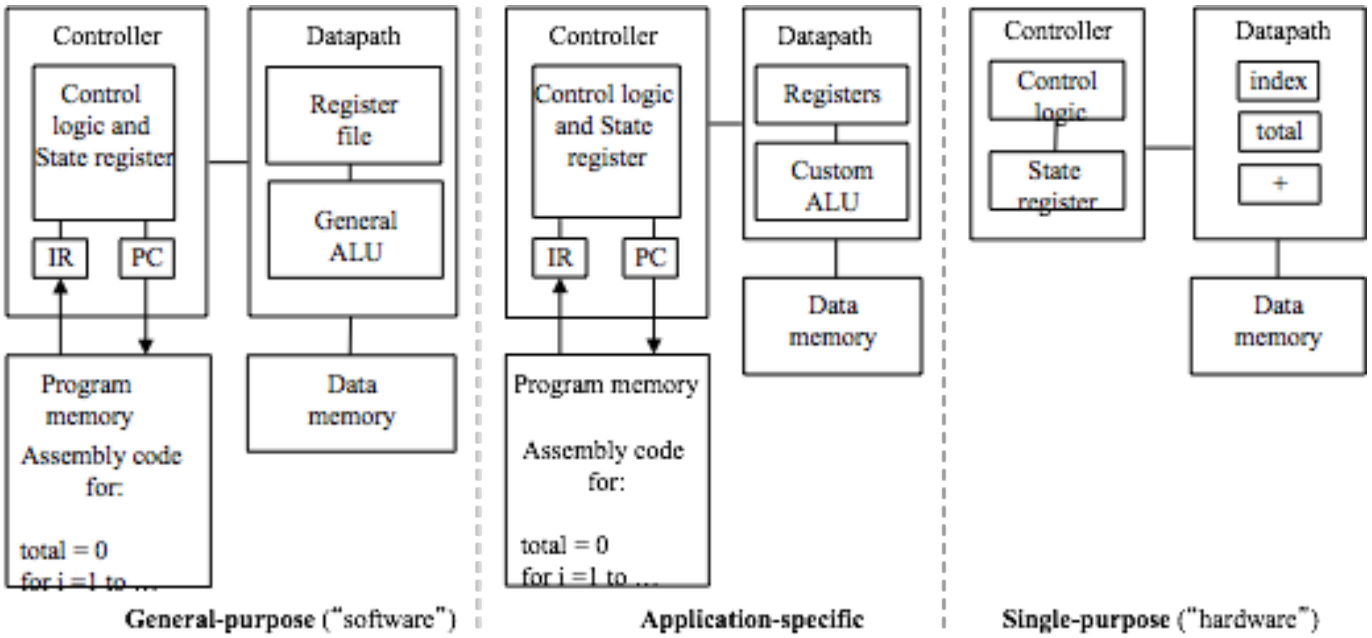
- A manner of accomplishing a task, especially using technical processes, methods, or knowledge

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.



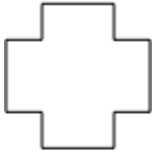
Processor Technology

Processors vary in their customization for the problem at hand

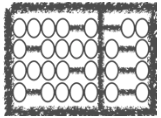
```
total = 0
for i = 1 to N  loop
   total += M[i]
end loop
```

General-purpose processor

Application-specific processor



Single purpose processor



Processor Technology: General-purpose processor

Programmable device used in a variety of applications

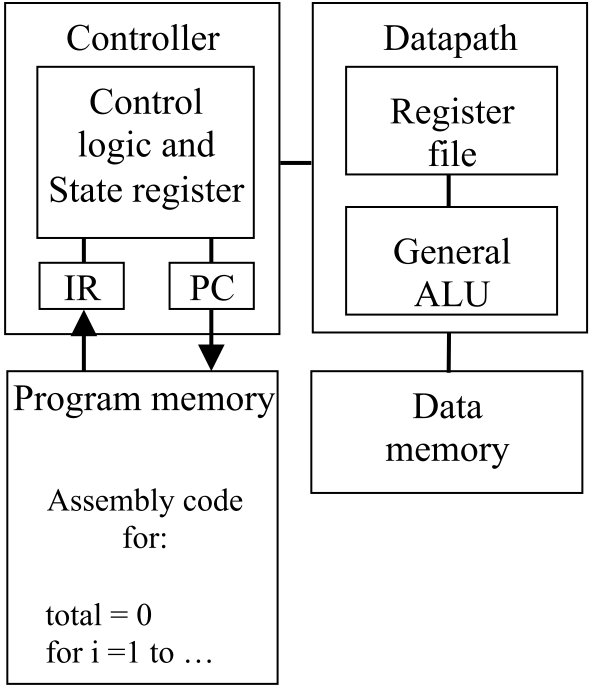
- Also known as "microprocessor"

Features

- Program memory
- General datapath with large register file and general ALU

User benefits

- Low time-to-market and NRE costs
- High flexibility



Processor Technology: Single-purpose processor

Digital circuit designed to execute exactly one program

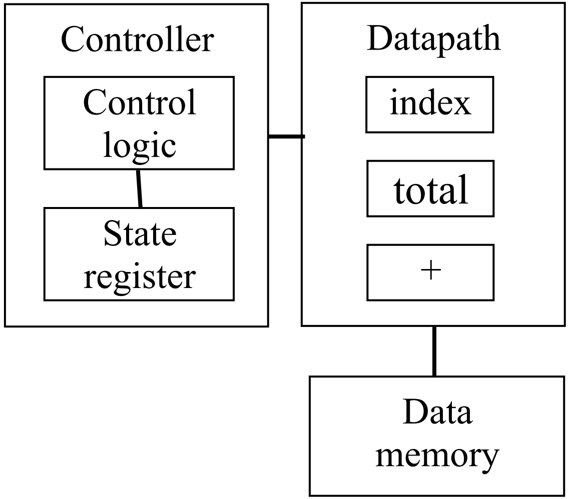
- a.k.a. coprocessor, accelerator or peripheral

Features

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

Benefits

- Fast
- Low power
- Small size



Processor Technology: Application-specific processor

Programmable processor optimized for a particular class of applications having common characteristics

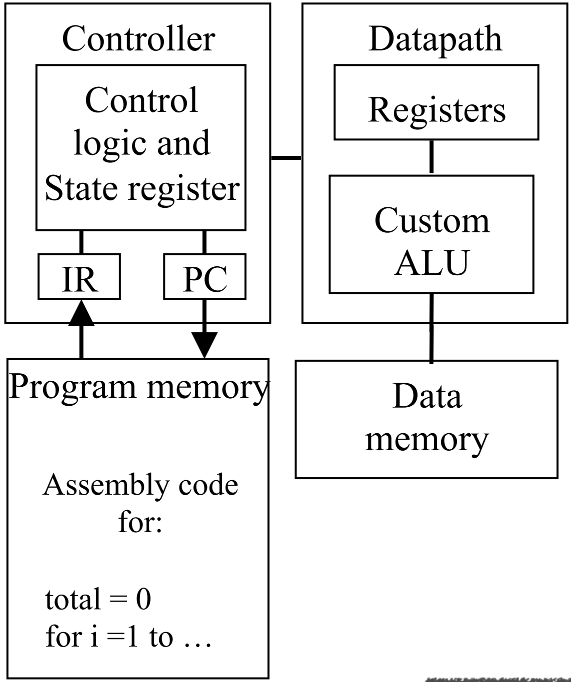
- Compromise between general-purpose and single-purpose processors

Features

- Program memory
- Optimized datapath
- Special functional units

Benefits

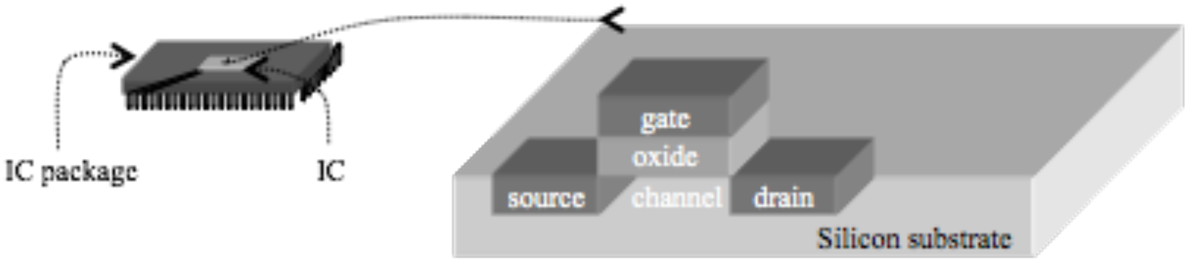
- Some flexibility, good performance, size and power



Integrated Circuit 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



Integrated Circuit Technology

Three types of IC technologies

- Full-custom/VLSI
- Semi-custom ASIC (gate array and standard cell)
- PLD (Programmable Logic Device)

Integrated Circuit Technology: 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

Integrated Circuit Technology: 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

Integrated Circuit Technology: PLD

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

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.

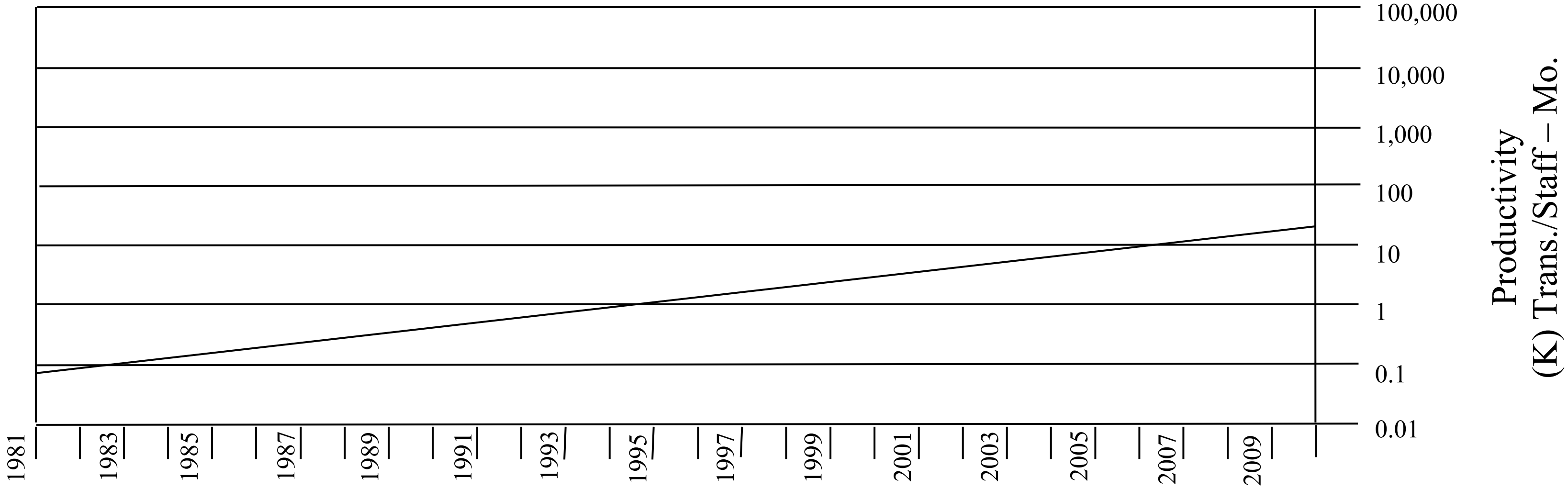
Test/Verification: Ensures correct functionality at each level, thus reducing costly iterations between levels.

	Compilation/ Synthesis	Libraries/ IP	Test/ Verification
System specification	System synthesis	Hw/Sw/ OS	Model simulat./ checkers
Behavioral specification	Behavior synthesis	Cores	Hw-Sw cosimulators
RT specification	RT synthesis	RT components	HDL simulators
Logic specification	Logic synthesis	Gates/ Cells	Gate simulators

To final implementation

Design Technology: Increase in productivity

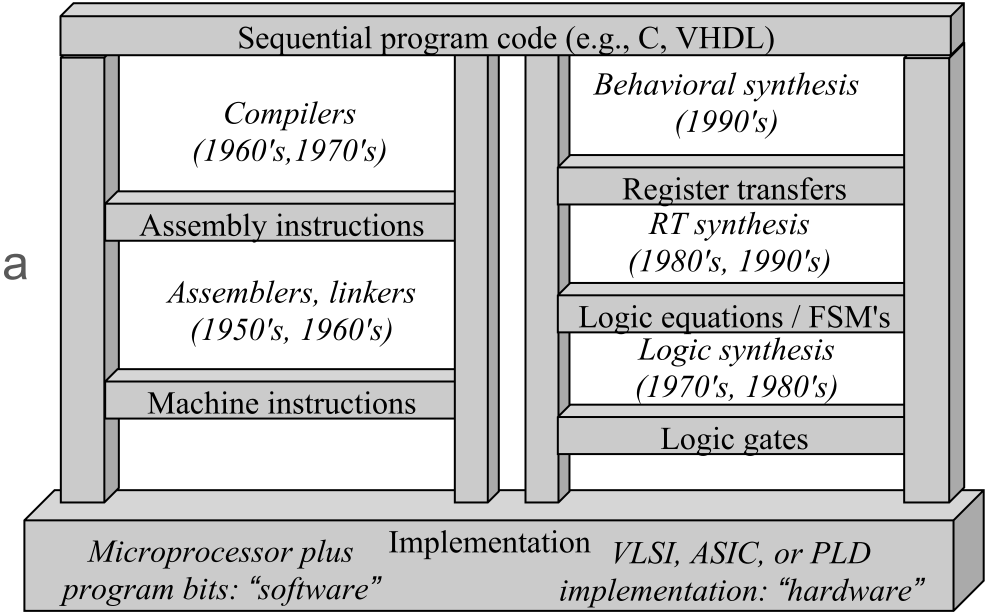
Productivity has increased exponentially over the past few decades.



Design Technology: HW/SW co-design

In the past:

- Hardware and software design technologies were very different
- Maturation of synthesis enabled a unified view of hardware and software
- Hardware/software "codesign"

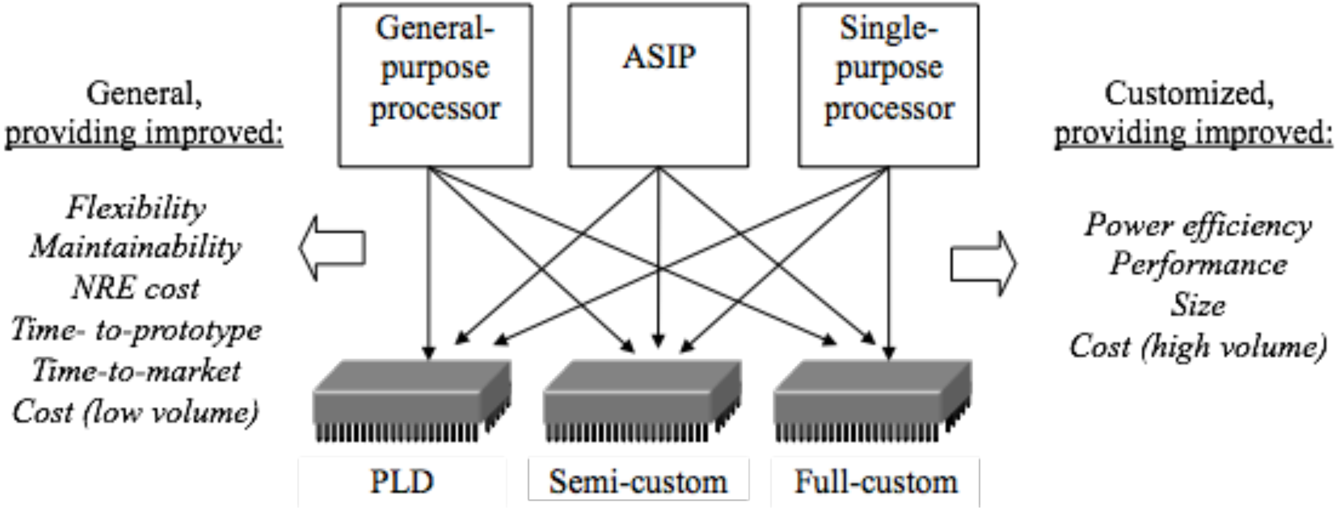


The choice of hardware versus software for a particular function is simply a tradeoff among various design metrics, like performance, power, size, NRE cost, and especially flexibility; there is no fundamental difference between what hardware or software can implement.

Independence of processor and IC technologies

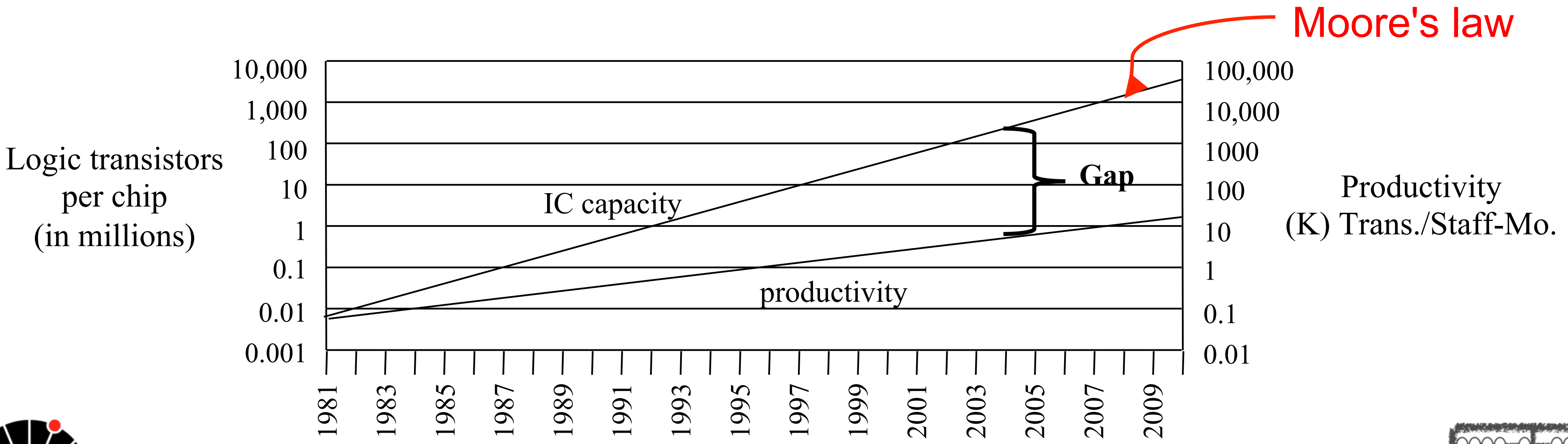
Basic tradeoff

- 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

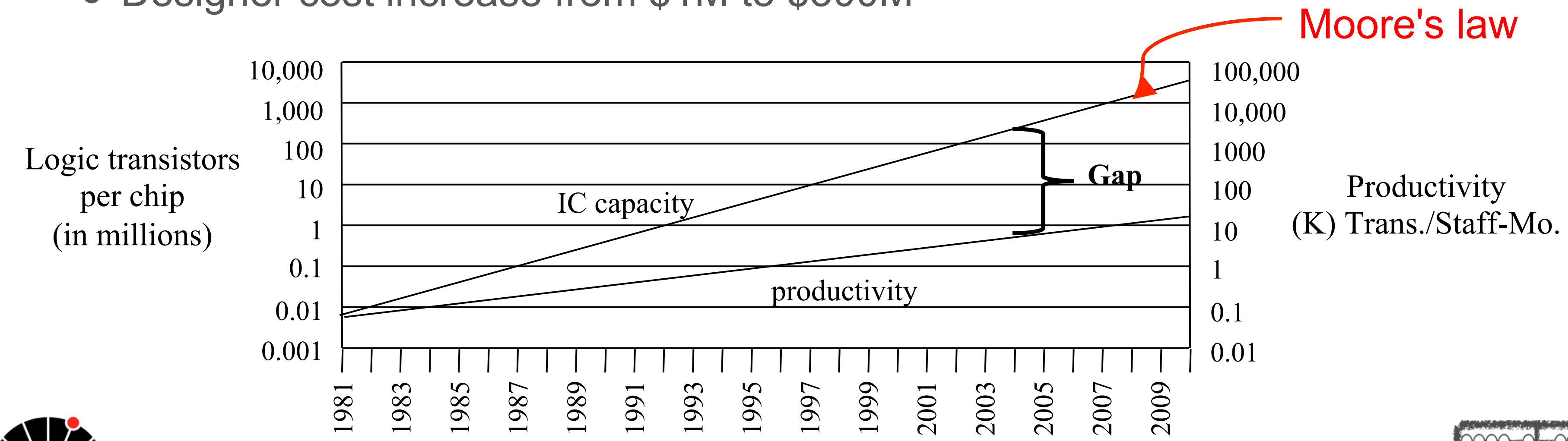


Design productivity gap

1981 leading edge chip required 100 designer months

2002 leading edge chip requires 30,000 designer months

- Designer cost increase from $1M to $300M



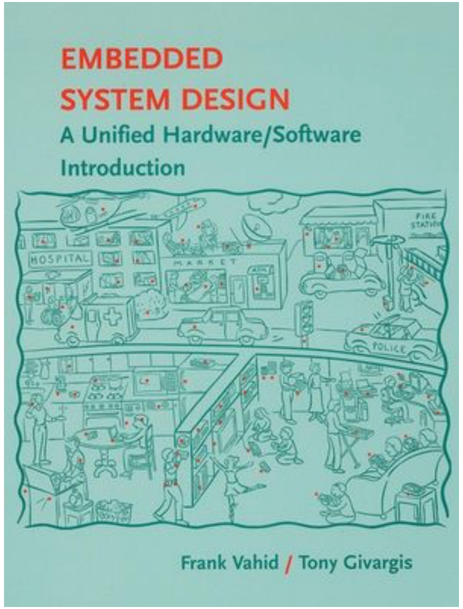
Summary

- Embedded systems are everywhere
- Key challenge: optimization of design metrics
- Design metrics compete with one another
- Three key technologies
  - Processor: general-purpose, application-specific, single-purpose
  - IC: Full-custom, semi-custom, PLD
  - Design: Compilation/synthesis, libraries/IP, test/verification

Reading for next class

Chapter 1: Introduction

Embedded System Design: A Unified Hardware/ Software Introduction
Frank Vahid and Tony Givargis
John Wiley & Sons;
ISBN: 0471386782. Copyright (c) 2002.



INF-741: Embedded systems programming for IoT

Prof. Edson Borin

Institute of Computing

University of Campinas