EEE8068 Real Time Computer Systems
Topic 1: Introduction

Dr. Bystrov, Dr. Koelmans

School of Electrical Electronic and Computer Engineering

University of Newcastle upon Tyne

Aims

- To study main concepts and characteristics of embedded systems
- Familiarisation with platforms, interfaces, protocols and standards
- Familiarisation with common components of embedded systems
- To gain experience of design

Intended learning outcomes

- Understanding of the real-time aspect of design
- Ability to choose the right platform and its configuration for the design
- Understanding of software drivers and firmware for microcontroller design
- Understanding and skills in using embedded operating systems
- Completing practicals
- Understanding of economics of embedded system design

Rationale

The formal lectures cover a set of hardware and software aspects needed to build a basic embedded system
Practicals develop skills and help to understand deeper the theory.

- Theory: FSM model, Petri net model, concurrency, arbitration, communication modelling, ACM, various types of real-time scheduling, their models and characteristics.
- Embedded design: ARM platform, cross compilation of source code in C, implementation of a real-time example with scheduling.

Assessment

- 50% Closed book exam
- 50% Coursework
  - Report on practical design 40%
  - Seminar presentation 10%

Books

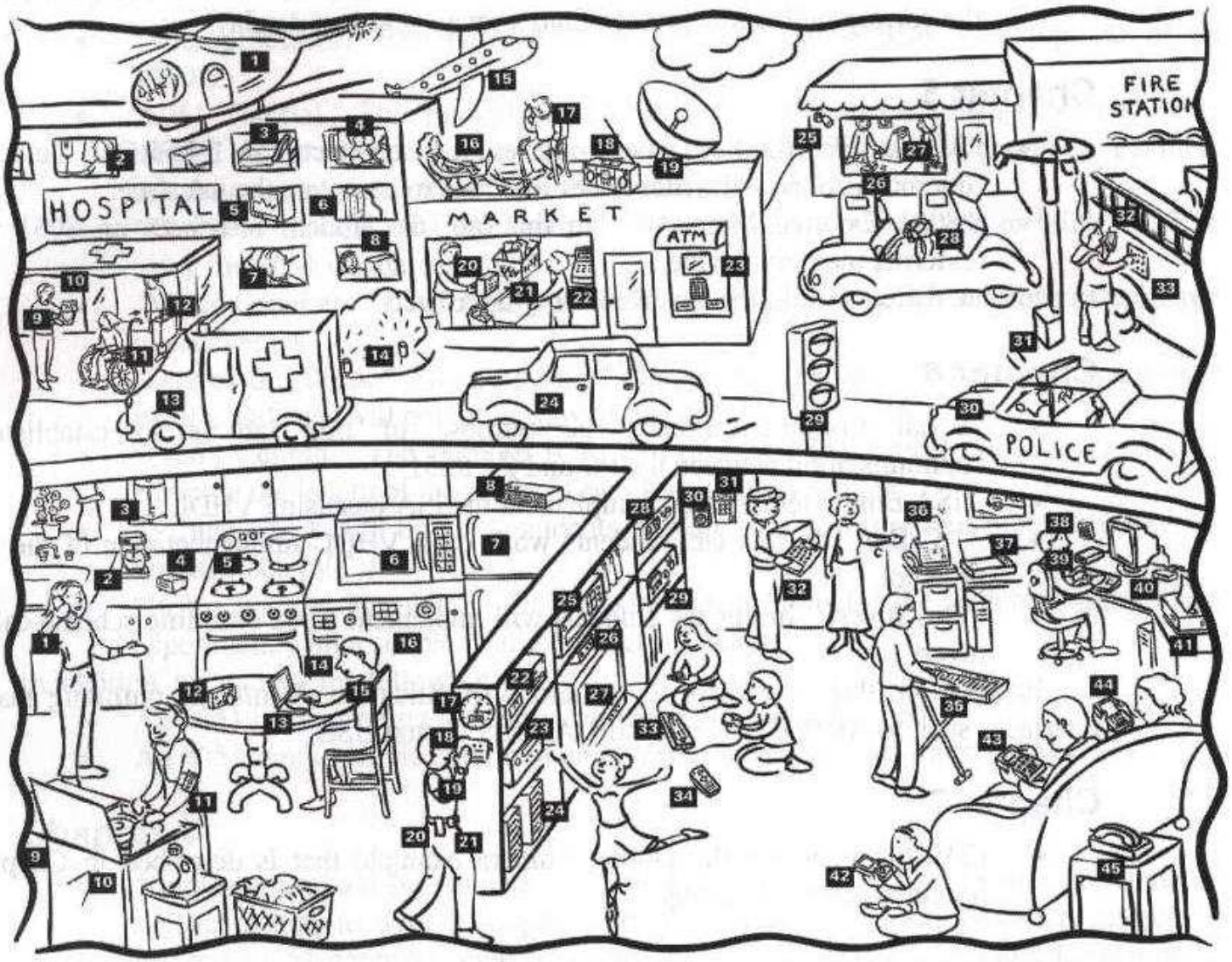
1. **Buttazzo, G.: “Hard Real-Time Computing Systems”, Springer, 2004 (highly recommended!)**
2. **Williams, R.: “Real-Time systems Development”, Elsevier, 2006**
3. **Web: “International Technology Roadmap for Semiconductors”**
4. Yakovlev, A. , Gomes, L., Lavagno, L. ed.: “Hardware design and Petri nets”, Kluwer, 2000.
5. Jerraya A.A., Mermet, J. ed.: "System-Level Synthesis", Kluwer, 1999.

[1] is available both electronically and on paper from the library

Optional reading

1. Balarin, F. et. al.: "Hardware-Software Co-design of Embedded Systems. The POLIS Approach", Kluwer, 1997.
2. Vahid, F., Givargis, T.: “Embedded system design : a unified hardware/software introduction”, Wiley, 2002.
3. Wolf, W.: "Computers as Components. Principles of Embedded Computing System Design", Morgan Kaufmann Publishers, 2001.

Introduction



What is an embedded system?

Definition: "nearly any computing unit not used as a desktop computer" (**Vahid & Givargis 2002**)

A better definition (Gupta 2002):

- employs a combination of hardware+software to perform a specific function;
- is a part of a larger system that may not be a "computer";
- works in a reactive and time-constrained environment.

My definituion: All modern electronics apart from primitive components are Embedded Systems.

General characteristics

- Latency limit
- Event-driven scheduling
- Time-driven scheduling
- Low-level programming
- SW tightly coupled to special HW
- Dedicated specialised functions
- Computer inside a control loop
- Multi-tasking
- Continuous running
- Various specific metrics: safety-critical app., security, power constraints, variability-tolerant design, etc.

Software/hardware co-design

- Software provides features and flexibility;
- Hardware (processors, ASICs, memory, accelerators, etc.) is used for performance, fault-tolerance and security.

Example of DSP:

- DSP code
- processor cores
- memory
- application-specific gates
- analogue I/O

(Gupta 2002)

Driving forces

- Computing technologies are proliferating to non-computing domains
  - portable units with significant data & control ops.
    - medical instrumentation & imaging, information appliances...
- Increasing need for product personalisation
  - Application domains and user differentiation
  - Device programmability
- Competitive pressures of "commodity markets"
  - Time to market (TTM), Time to money (TT$),
    - Product lines,
    - Product differentiation, etc.

Market facts – ES revolition

- Market size for embedded systems: $31 B (2002)
  - General-purpose computing: $46.5 B
- Microcontroller market: $4.6 B, +18% AGR
  - versus +10% AGP for GP processors

Market watchers tell us that:

- ECS market will surpass computing market before this decade is out (2009)

Design challenges (1)

Read ITRS – a very important home task!

- System complexity
  - Higher-level abstraction and spec.
  - Dynamism and softness
  - System-level reuse
  - Design space exploration and system-level estimation
- Power consumption
  - Energy-performance-flexibility trade-offs
  - Novel data transfer and storage techniques

Design challenges (2)

- Integration of heterogeneous technologies
  - Co-design
    - HW-SW
    - chip-package-board
    - fixed-reprogrammable
  - Non-scalability of analogue circuits
  - Top-down implementation planning
    - digital, AMS, RF, MEMS, SW

Design challenges (3)

- Embedded software
  - SW-SW codesign onto highly reprogrammable platforms
  - System capture and abstraction
  - New automation from high-level to HW-SW (incl. SW synthesis)
  - Formal verification for SW
  - HW-SW co-verification
- Verification, test and culture
  - Integration-oriented verification and test architectures
  - Divergent design practices and cultures
- Communication-centric design

Common design metrics (1)

Unit cost: fabrication cost of a single copy of the system, excluding NRE cost

NRE cost: (Non-Recurring Engineering cost) one-time monetary cost of designing the system

Total cost: NRE_cost + Unit_cost * Number_of_units

Per product cost: NRE_cost/Number_of_units + Unit_cost

Time-to-prototype: time to develop a working version

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

Common design metrics (2)

Size: physical dimensions or volume

Performance: execution time, throughput

Power: amount of power consumed

Flexibility: the ability to change the functionality of the system without incurring heavy NRE costs

Correctness, Safety, Security, etc.

Design metric competition

- Improving one metric may worsen others
- Expertise is needed
  - system-level,
  - software,
  - hardware,
  - communications, coding, encryption, etc.
- Trade-off examples – Home Task
  - performance–size,
  - performance–power,
  - size–power,
  - optimality–NRE cost,
  - NRE cost– time-to-market, etc.

Software/hardware co-design

- Software provides features and flexibility;
- Hardware (processors, ASICs, memory, accelerators, etc.) is used for performance, fault-tolerance and security.

Example of DSP:

- DSP code – SW
- Computing system – HW that executes the SW
  - processor core(s) + memory
- application-specific gates – HW accel., interfaces
- analogue I/O – analogue HW

What is SW/HW co-design (1)

- Traditional approach
  - SW and HW partitioning is decided at an **early stage**, and designs proceed separately from then onward.
- CAD today addresses synthesis problems at a purely hardware level:
  - efficient techniques for data-path and control synthesis down to silicon.
- ECS hardware often uses standard IC/IP components
  - μP, DSP cores, network and bus interfaces, etc.

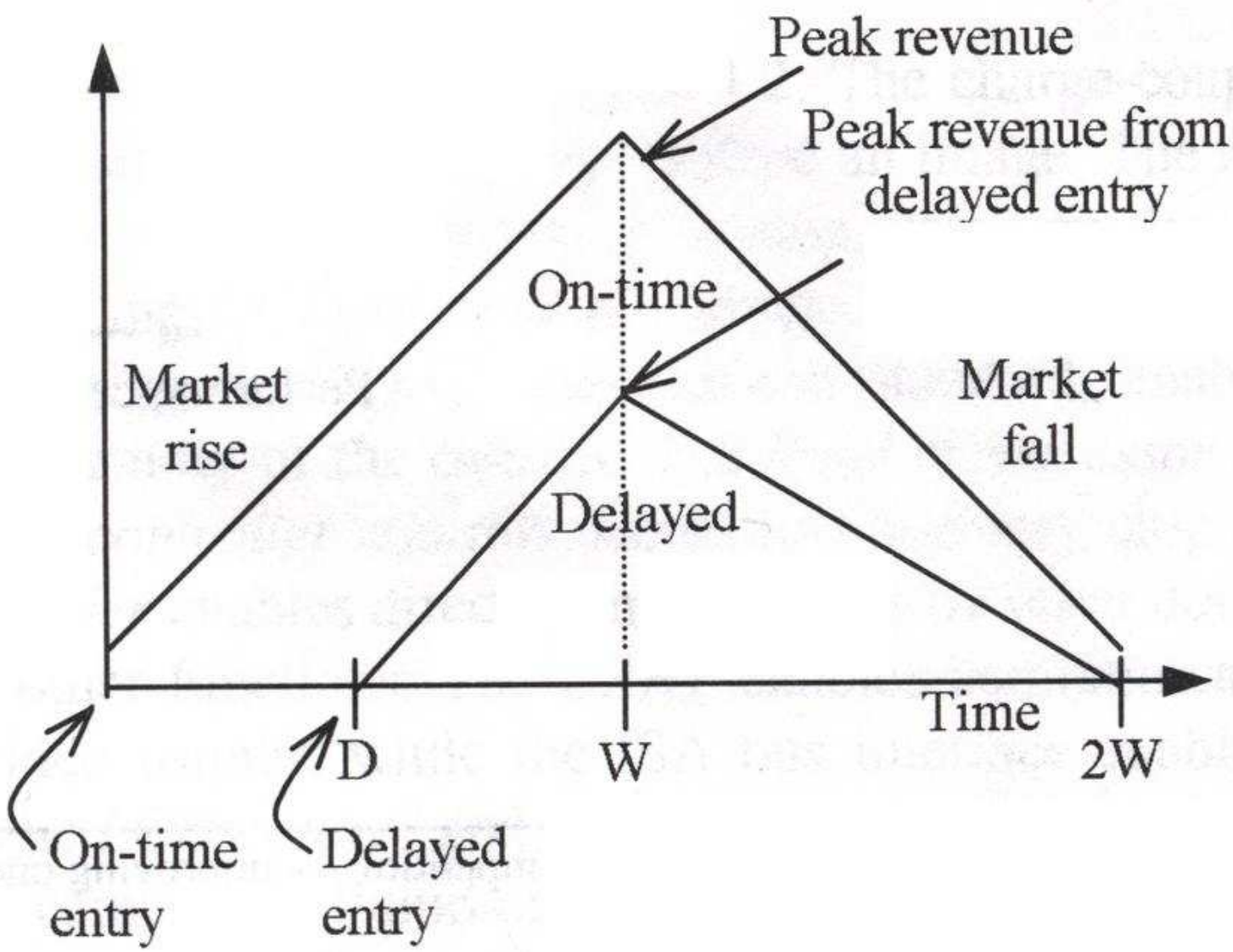
What is SW/HW co-design (2)

- Co-design of the future
  - A flexible design strategy, wherein the **HW/SW designs proceed in parallel**, with feedback and interaction occurring between the two as the design progresses.
  - Final HW/SW partition/allocation is made **after** evaluating trade-offs and performance of options.
  - Task: identify the associated problems.
  - Home task: read the ITRS and search the web.

Co-design as a metric

- Goal: Improved time to market (for a given function)
  - with acceptable performance
    - (speed, power, cost, reliability, ...)
  - and product differentiation
- **Traditional** productivity metric
  - Gates delivered per day
- **New** productivity metric of relevance
  - Functionality delivered in a market window
- Requires better ways to design systems that have a heterogeneous mix of hardware and software subsystems.
  - => New design paradigms, techniques, & tools.

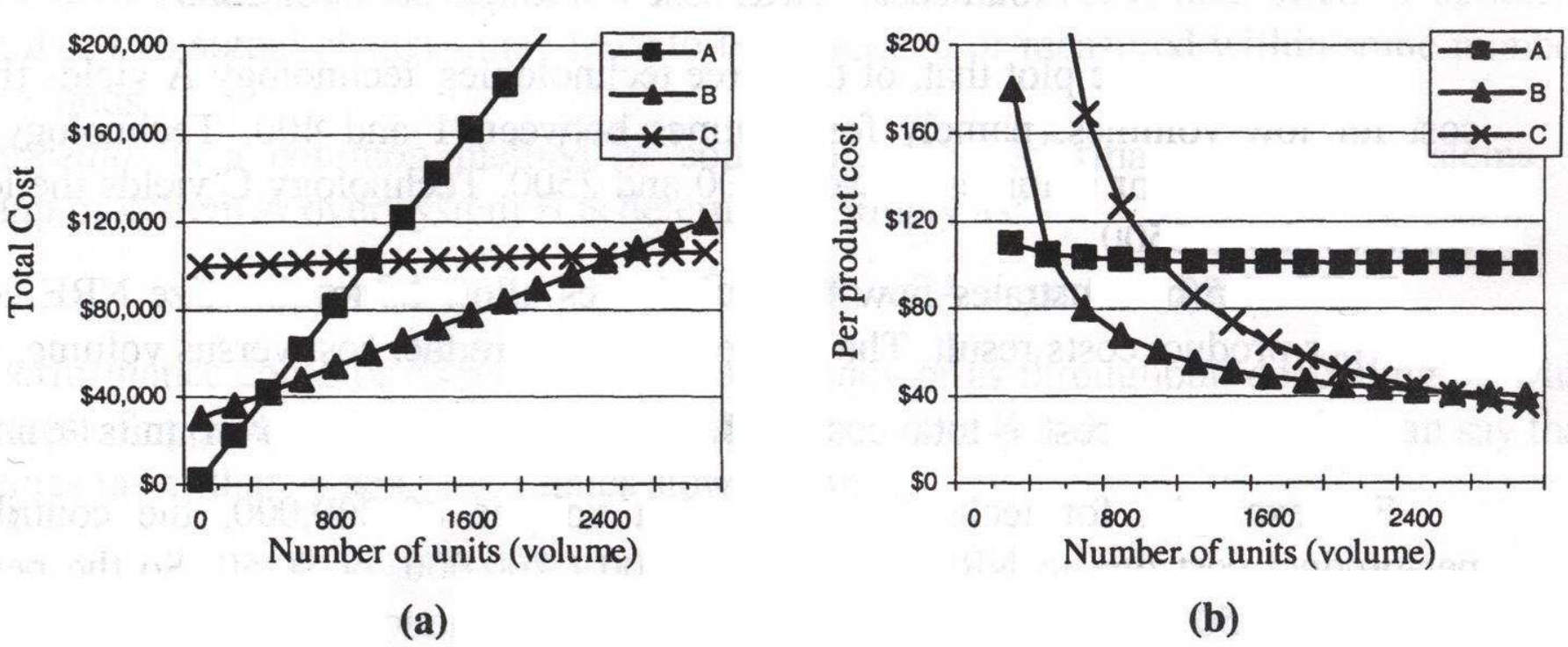
Time-to-market metric (simplified model)



Delayed market example

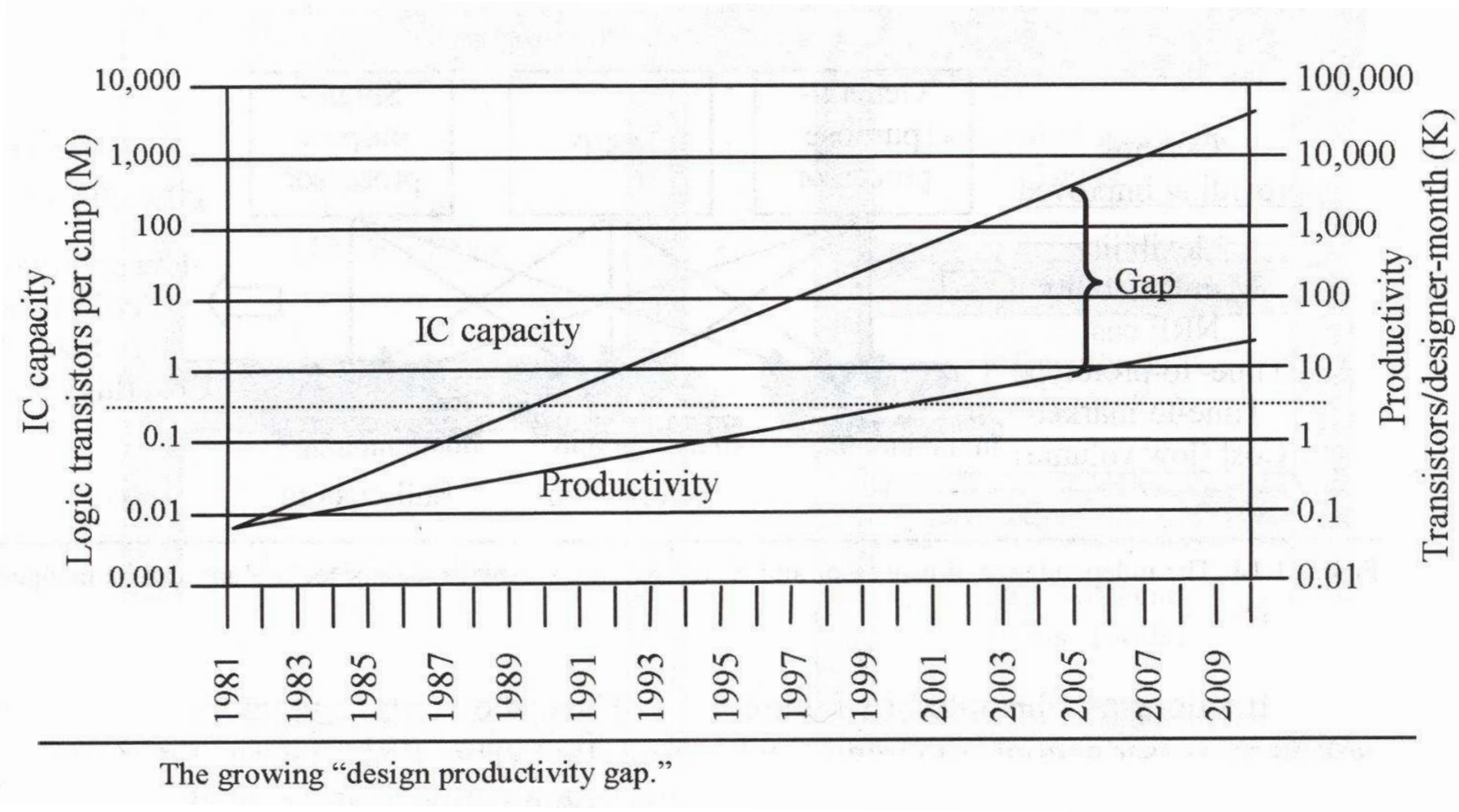
- Total (integral) revenues are the areas under the graphs (triangle areas)
- Please work out the formula for the percentage revenue loss.
- Calculate the loss for
  - lifetime $2W = 52$ weeks;
  - delay $M = 10$ weeks.
- Plot the percentage revenue loss as a function of delay

NRE and Unit Cost design metrics



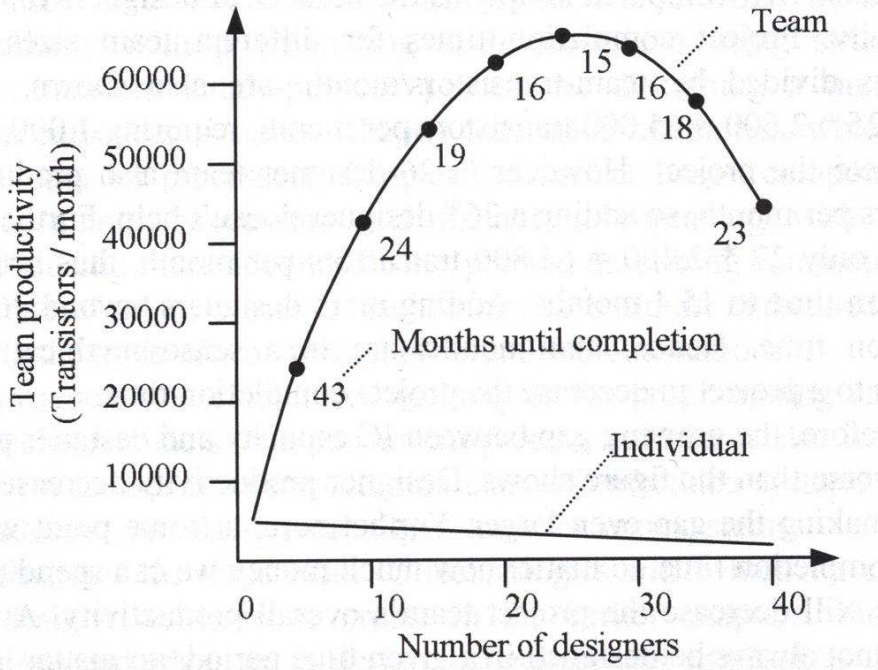
Costs for technologies A, B, and C as a function of volume: (a) total cost, (b) per-product cost.

Design productivity gap



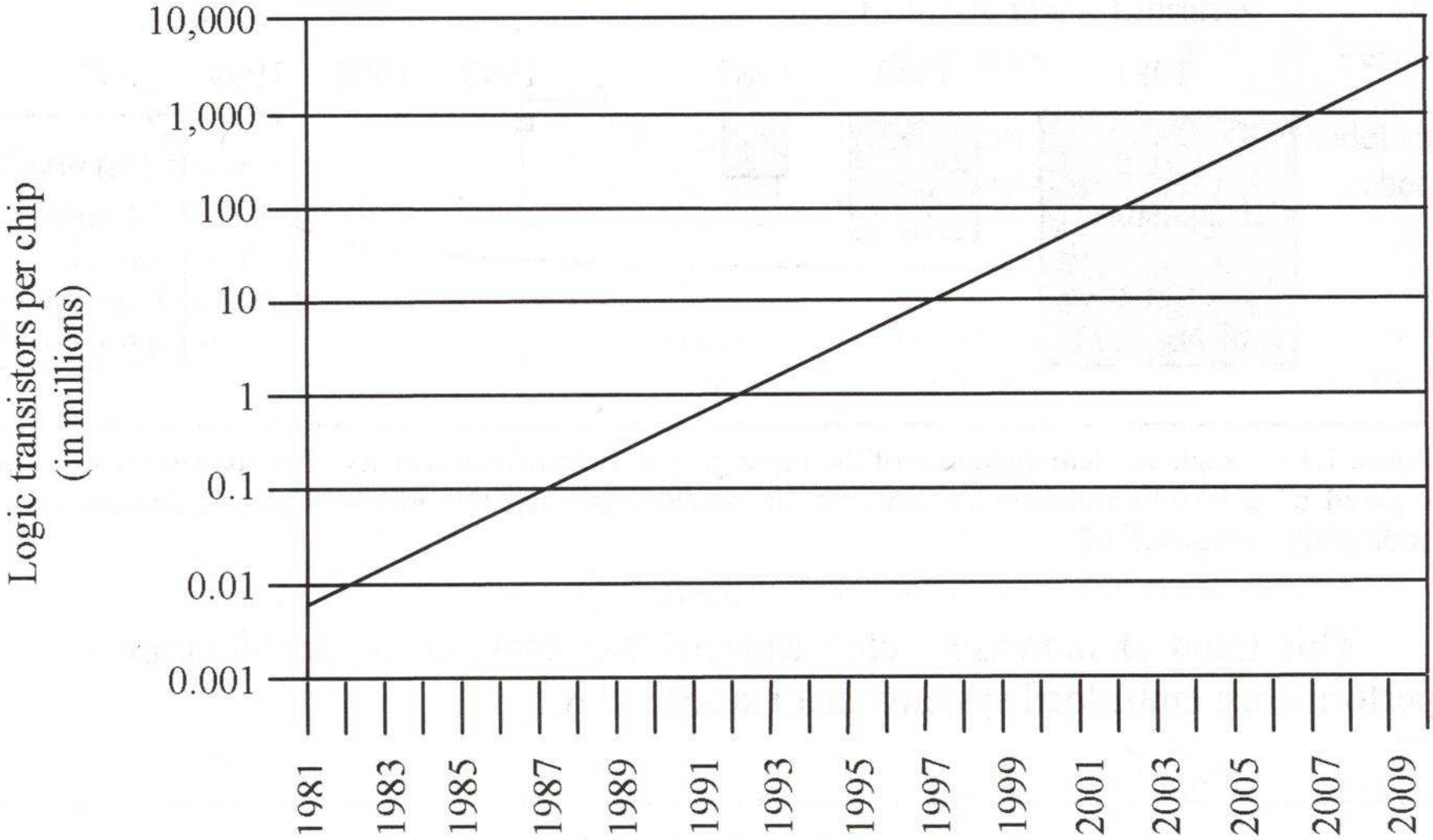
The growing "design productivity gap."

The gap is bigger than it seems...



- "The Mythical Man-Month" by Frederick Brooks 1975
- Scaling up the group "hits the wall" at some point
- After that point adding new men does NOT increase overall productivity
- Individual productivity goes down

… and keeps increasing



"Moore's Law": IC capacity increases exponentially

Other metrics

Which metrics are affected by changing processor types?
Processor types:

- General purpose
  - controller – program memory
  - datapath – ALU – data memory
- Application-specific
  - controller – program memory – spec. instructions
  - spec. datapath – custom ALU – data memory
- Single purpose
  - FSM controller
  - spec. datapath – single purpose ALU – data memory

Conclusions

- Embedded systems, including real-time, are everywhere and represent all electronic applications
- Everything above is an Introduction – the really important and complex stuff will follow
- Attendance is important
- Finding the literature sources is extremely important
- This is an advanced course – please help us to analyse your needs as a learner, so we could support you to a maximum extent
- Technical content of the introduction covered a definition of an embedded systems and metrics