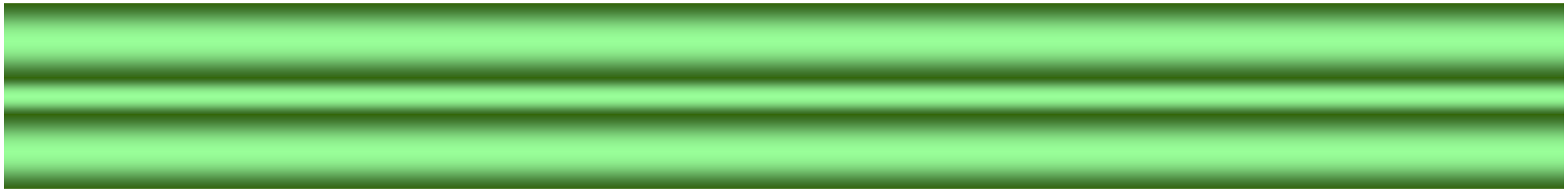
EE414 Embedded Systems

Ch 1. Introduction to Embedded Systems
Part 2/2



Byung Kook Kim
School of Electrical Engineering
Korea Advanced Institute of Science and Technology

Overview

- 1.1 Embedded Systems Overview
- 1.2 Design Challenge – Optimizing Design Metrics

Technologies

- 1.3 Processor Technologies
- 1.4 IC Technologies
- 1.5 Design Technologies
- 1.6 Trade-offs

1.3 Processor Technology

- Technology
  - A manner of accomplishing a task, especially using technical processes, methods, or knowledge.
- Three key technologies for embedded systems:
  - 1.3 Processor technology
  - 1.4 IC technology
  - 1.5 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
  - *Not equal to* (contains) general-purpose processor

  - General-purpose processor
  - Application-specific processor
  - Single-purpose processor

Processor technology

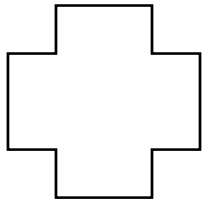
- Processors vary in their customization for the problem at hand
  - Summing function

Desired functionality

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

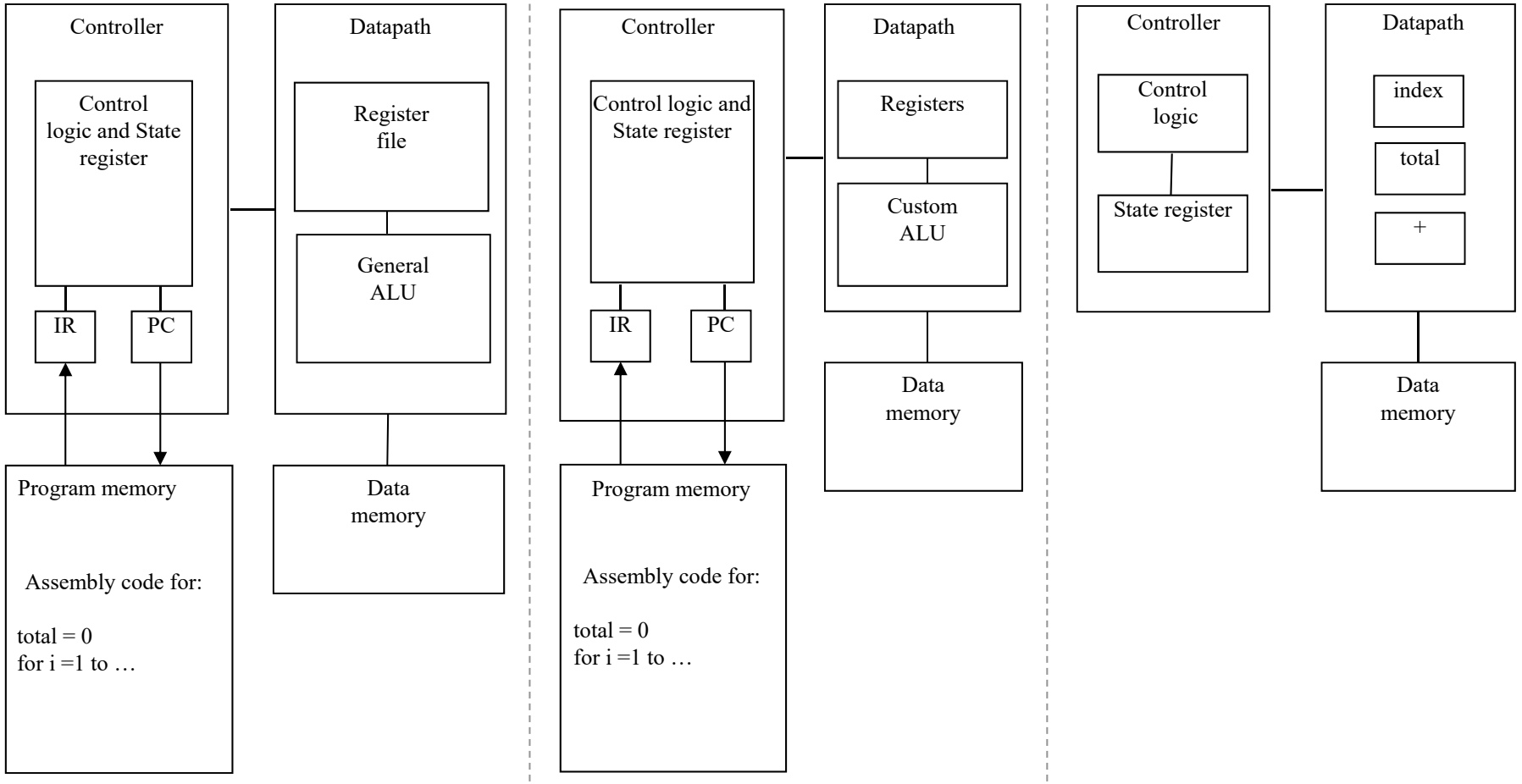
General-purpose processor

Application-specific processor



Single-purpose processor

Processor Technology



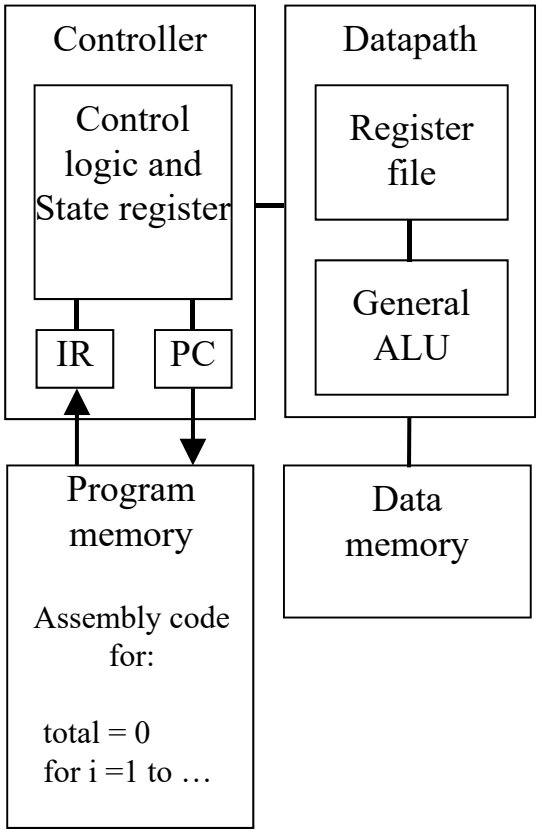
General-purpose ("software")

Application-specific

Single-purpose ("hardware")

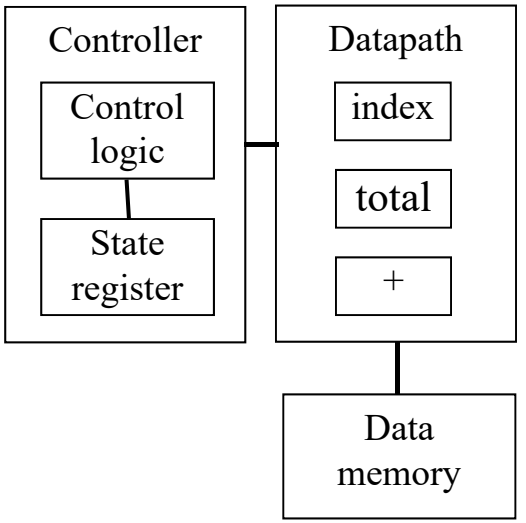
A. General-purpose processors

- Programmable device used in a variety of applications
  - Maximize the number of devices sold
  - Also known as "microprocessor": Central Processing Unit on a chip
- Features
  - Program memory
  - General datapath with large register file and general ALU
- User benefits
  - Programming the functionality (software):
  - Low time-to-market and NRE costs
  - High flexibility
- "Pentium" the most well-known, but there are hundreds of others.



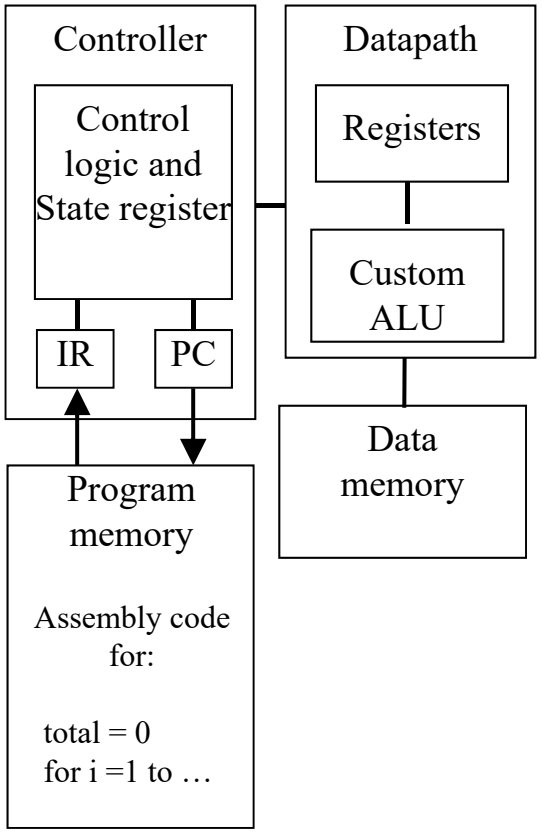
B. Single-purpose processors

- Digital circuit designed to execute exactly one program
  - a.k.a. coprocessor, accelerator or peripheral
  - JPEG codec
- Features
  - Contains only the components needed to execute a single program
  - No program memory
- Benefits
  - Fast, Low power, and Small size
  - Low unit cost for large volumes
- Drawbacks
  - High NRE cost
  - Low flexibility
  - High per-product cost for small volumes.



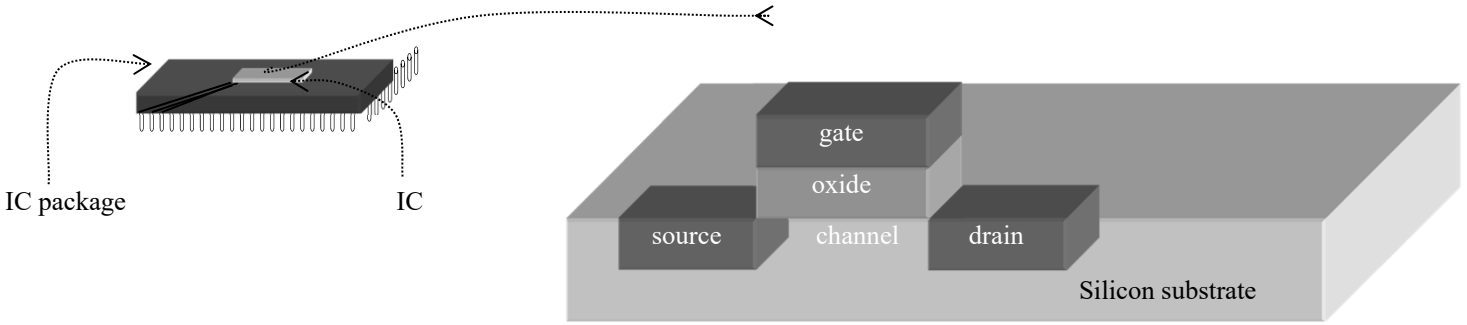
C. Application-specific [instruction-set] processors (ASIP)

- 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
- Types
  - Microcontrollers
  - Digital Signal Processors: Multiply-accumulate unit



1.4 IC technology

- The manner in which a digital (gate-level) implementation is mapped onto an IC (Integrated Circuit or chip)
  - IC technologies differ in their customization to a design
    - Most popular: CMOS
  - IC's consist of numerous layers (perhaps 10 or more)
    - IC technologies differ with respect to who builds each layer and when.
  - Layout
    - Feature size (submicron, nano).



IC technology (II)

- Three types of IC technologies:
  - A. Full-custom/VLSI (Very Large Scale Integration)
  - B. Semi-custom ASIC (Application Specific Integrated Circuit)
  - C. PLD (Programmable Logic Device)
- Independent from processor technology.

A. Full-custom/VLSI

- All layers are optimized for an embedded system's particular digital implementation
  - Designing down to transistor level
    - Placing transistors
    - Sizing transistors
    - Routing wires
  - Send mask spec to fabrication plant.
- Benefits
  - Excellent performance, small size, and low power.
- Drawbacks
  - Very high NRE cost (e.g., $300k), long time-to-market
- Used for high-volume or extremely performance-critical applications.

B. Semi-custom ASIC

- Lower layers are fully or partially built
  - Devices
    - Gate Array
    - Standard Cell: AND gate or AND-OR-INVERT
  - Designers are left with routing of wires and maybe placing some blocks (Mask design)
  - The most popular IC technology
- Benefits
  - Good performance and size
  - Less NRE cost than a full-custom implementation (perhaps $10k to $100k)
- Drawbacks
  - Still require weeks to months to develop

C. PLD (Programmable Logic Device)

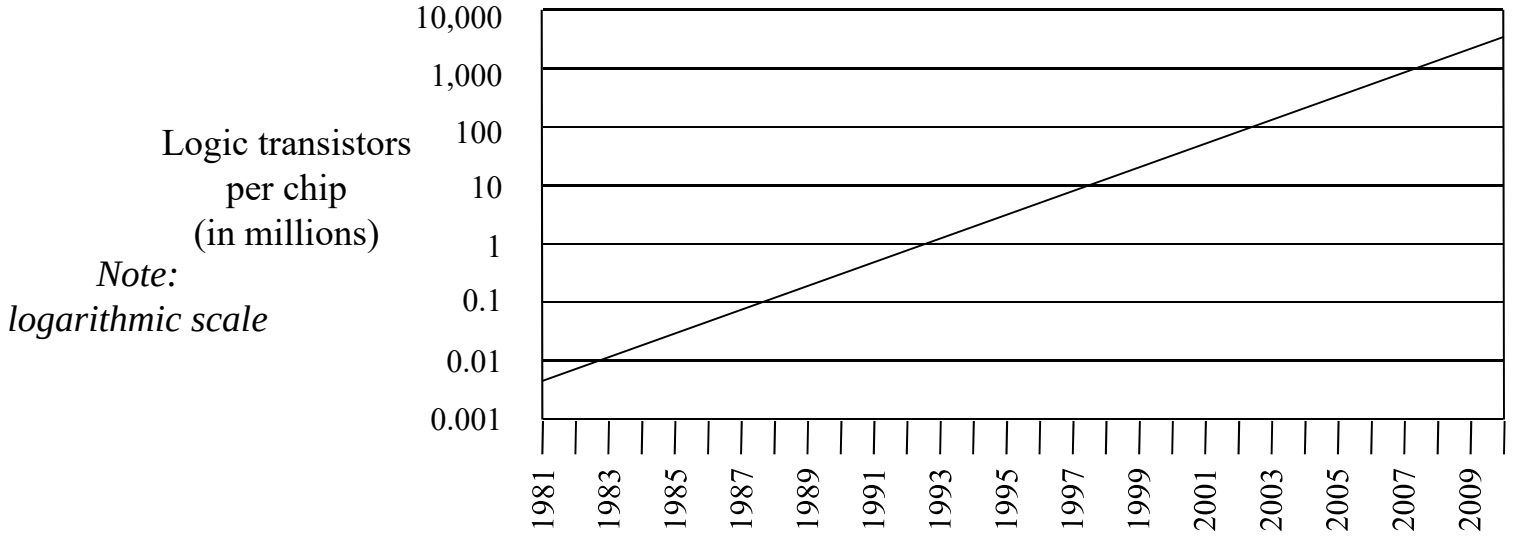
- All layers already exist
  - Designers can purchase an IC
  - Programmable circuit: Connections on the IC are either created or destroyed to implement desired functionality
- Devices
  - Programmable Logic Array (PLA)
    - Programmable array of AND gates and programmable array of OR gates
  - Programmable Array Logic (PAL)
    - One programmable array
  - Field-Programmable Gate Array (FPGA)
    - More general connectivity among blocks. Very popular
- Benefits
  - Very low NRE costs, almost instant IC availability
- Drawbacks
  - Bigger, expensive (perhaps $30 per unit), power hungry, slower.

Moore's law

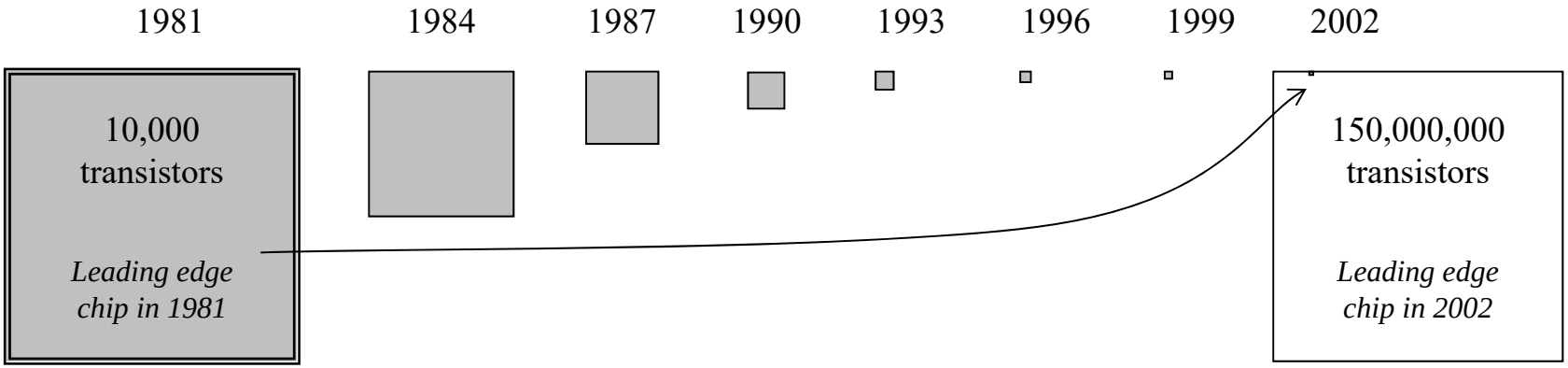
TRENDS – **Moore's Law**

- 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



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
- Low-cost high-performance embedded systems proliferate.

1.5 Design Technology

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

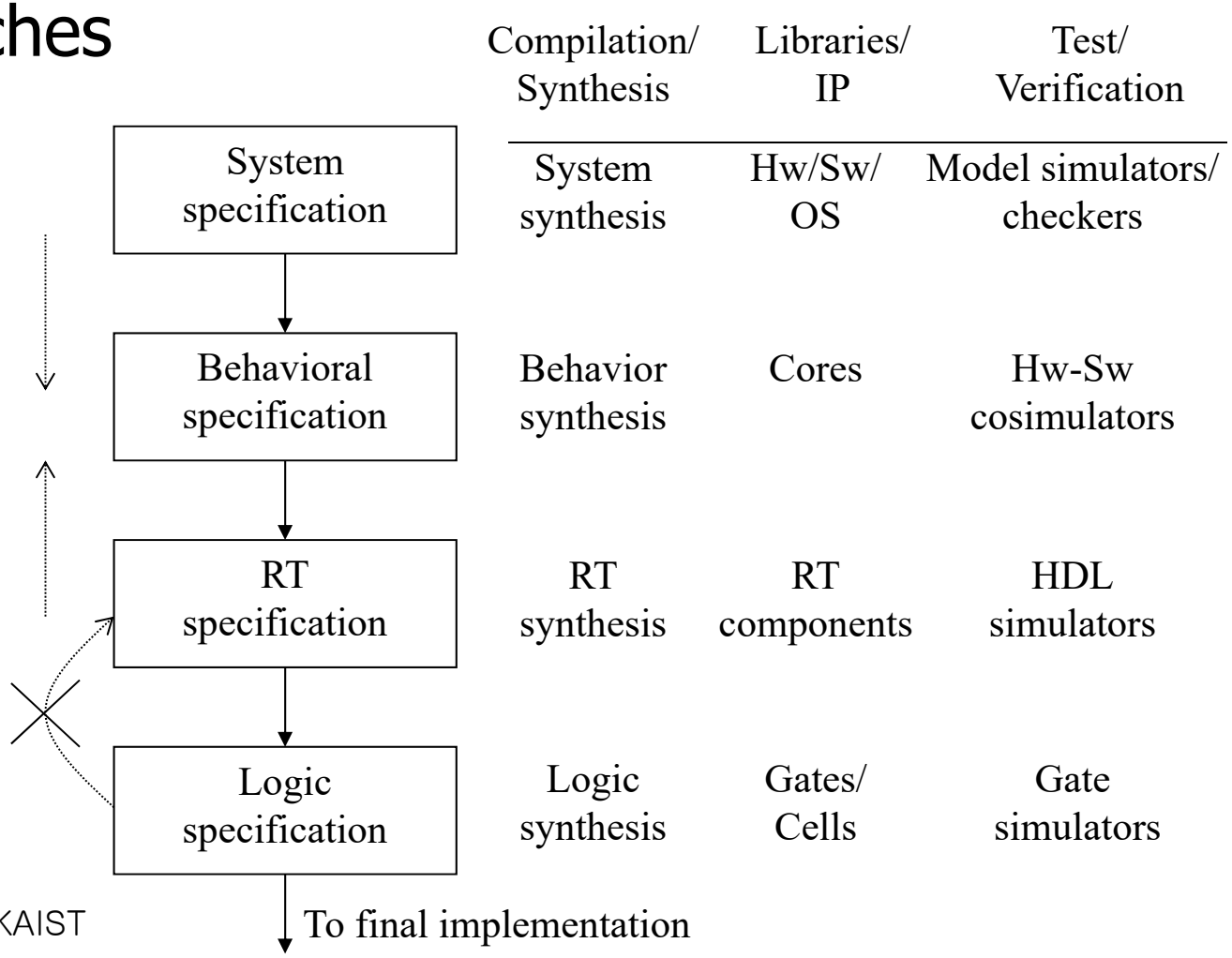
- **Top-down design process**: popular
  - Refines the system through several abstraction levels
  - System level
    - System specification: in C, English
    - Distributing into several processors.
  - Behavioral specification
    - Behavior of each processor.
  - Register Transfer (RT) specification
    - Assembly code level or Register transfer level
  - Logic specification
    - Boolean equations.

Design Technology (II)

- **Design Approaches** – Increased productivity
  - 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.

Design Technology (III)

- Ideal top-down design process & Design approaches



A. Compilation/Synthesis

- Specify desired functionality in an abstract manner
- Automatically generates lower-level implementation details.
  - Improve productivity by reducing the amount of details

- Tools
  - Logic synthesis tools: Boolean expr. -> Connection of logic gates (netlist)
  - Register-transfer (RT) synthesis tool: FSM, RT -> Datapath & controller of Boolean eq.
  - Behavioral synthesis tool: Sequential -> FSM, RT
    - Software compiler: Sequential program -> Assembly code (RT)
  - System synthesis tool: System spec. -> Sequential programs.

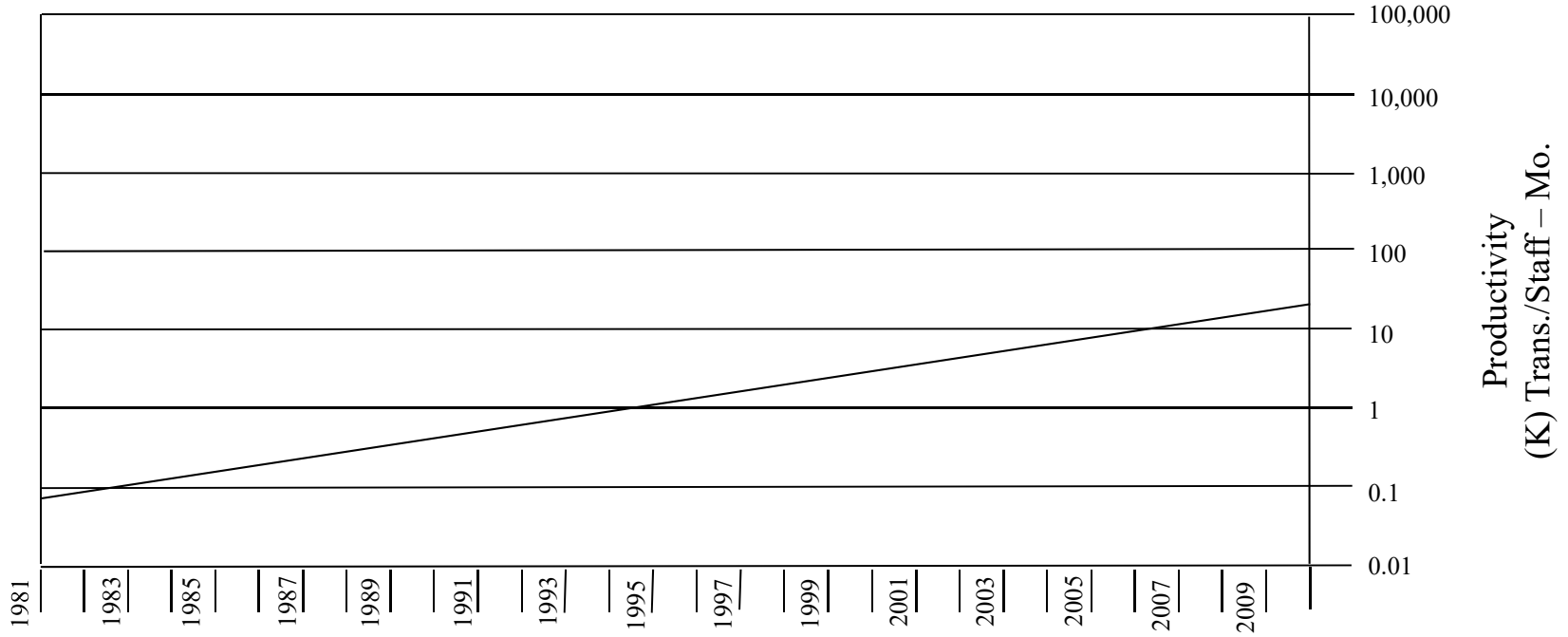
B. Libraries/IP

- Library: Reuse of preexisting implementations
  - Improve productivity (Find rather than design)
- Libraries
  - Logic level library: Layout of logic gates and cells
  - RT-level library: Layout of RT components
  - Behavioral-level library: Components, such as compression, bus, processor etc.
  - System level library: Complete systems solving particular problems.
- Intellectual Property (IP)
  - Cores in an intellectual form
  - Copy protected.

C. Test/Verification

- Ensure that functionality to be correct.
- Tools
  - Simulation
    - Gate level simulator
    - RT level (HDL: Hardware Description Language)
      - Provide output waveforms, given input waveforms.
    - Processor level: HW & SW co-verification.
    - System level: model simulator, model checker.
  - Formal verification
    - Growing in popularity.

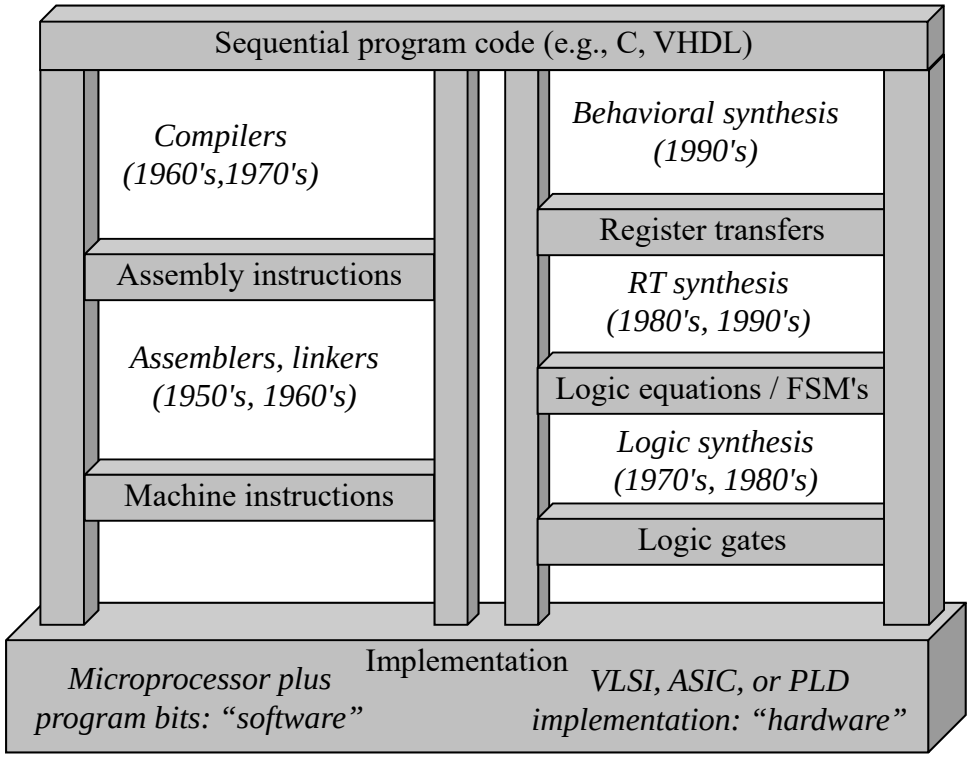
Design productivity exponential increase



- Exponential increase over the past few decades.
  - 1981: 100 TRs/month
  - 2002: 5000 TRs/month.

1.6 Trade-offs

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

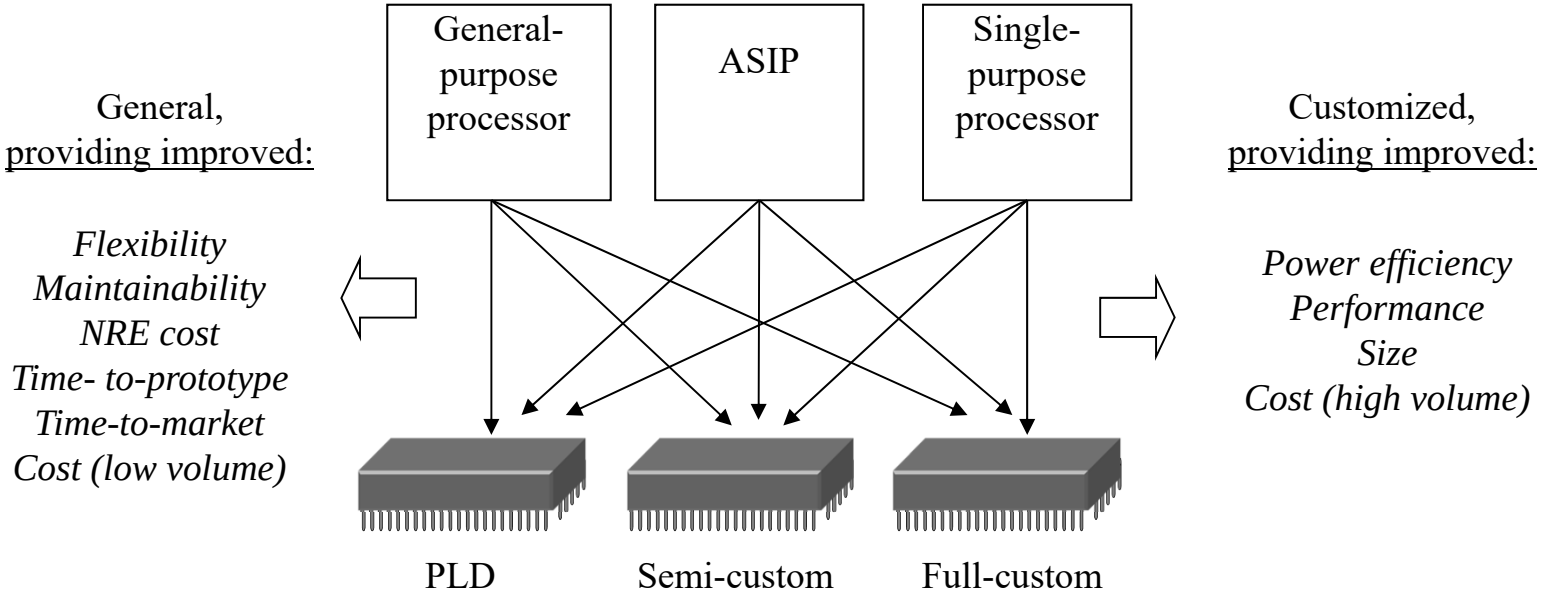


- Hardware/software "codesign"
  - The co-design ladder ->
  - Unified view

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.

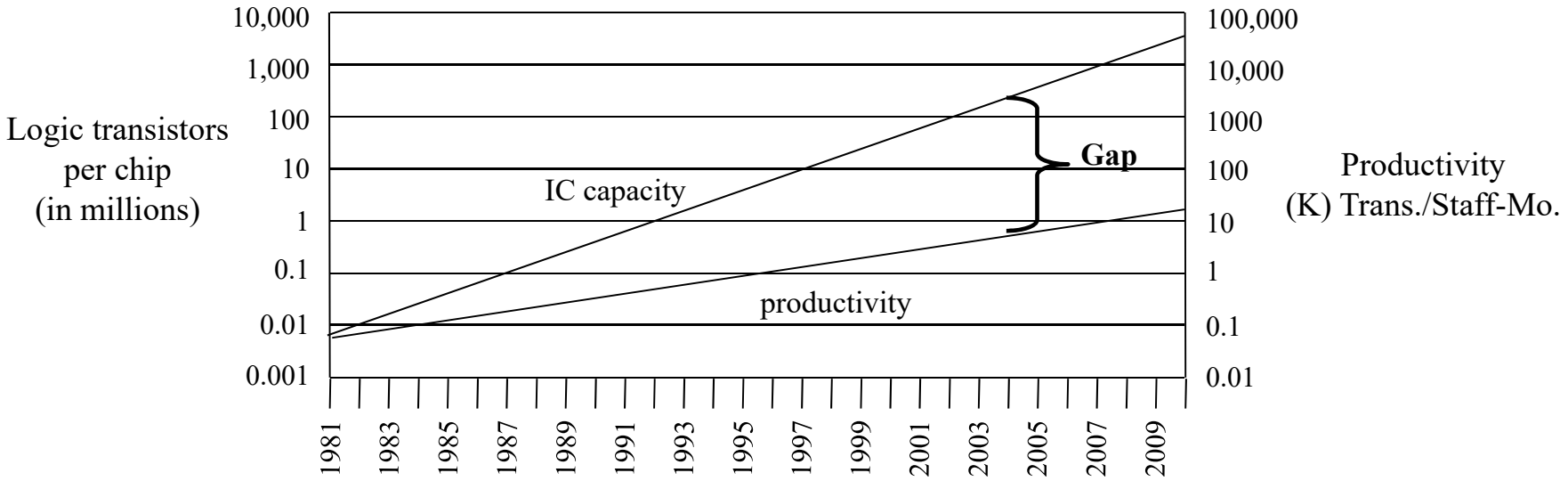
Independence of processor and IC technologies

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



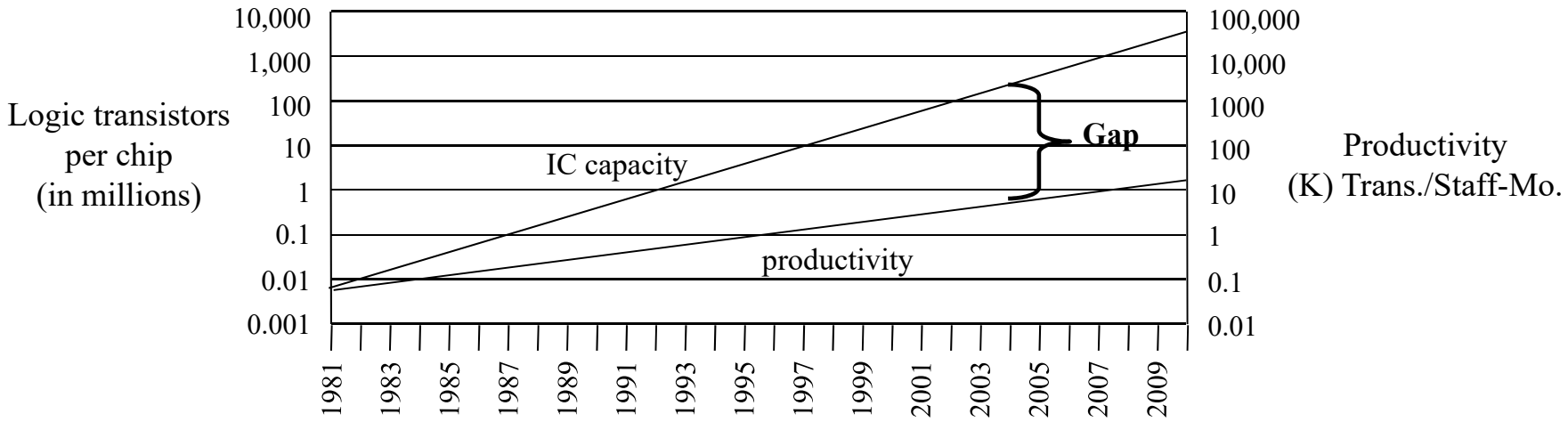
Design productivity gap

- 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 (II)

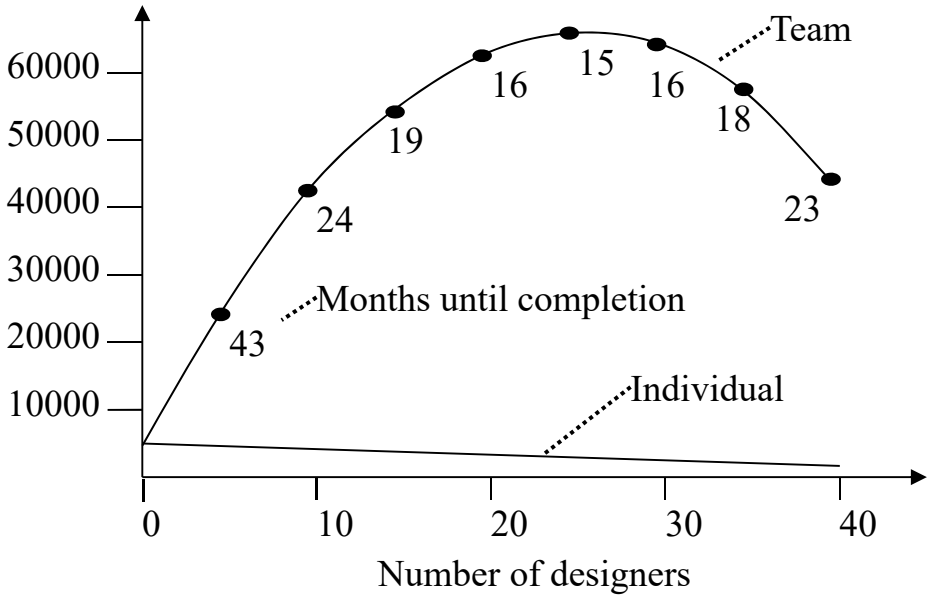
- 1981 leading edge chip required 100 designer months
  - 10,000 transistors / 100 transistors/month
- 2002 leading edge chip requires 30,000 designer months
  - 150,000,000 / 5000 transistors/month
- Designer cost increase from $1M to $300M



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

- Three key technologies
  - Processor: general-purpose, application-specific, single-purpose
  - IC: Full-custom, semi-custom, PLD
  - Design: Compilation/synthesis, libraries/IP, test/verification

- A unified view of hardware and software is necessary to improve productivity
  - Hardware and software codesign.

References

- [1] Frank Vahid, “Embedded system design: A unified hardware/software introduction”, John Wiley & Sons, 2002.