Chapter 1

Introduction

Contents:

- ASIP DSP: Application Specific Instruction set Processor for Digital Signal Processing
- The goal of Chapter 1: To understand why to learn ASIP DSP and how to learn ASIP DSP
- Motivations

DSP: The key technology

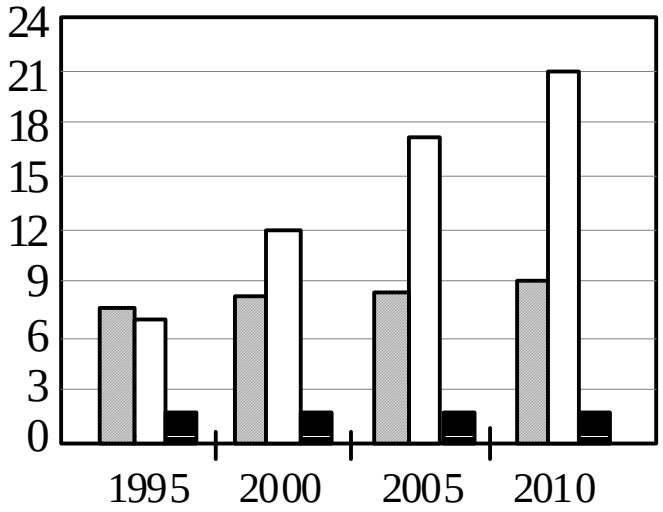
for

- Communication
- Home electronics
- Healthcare
- Industrial control
- Transportation
- Environment
- Test
- Scientific applications
- Military/Defense

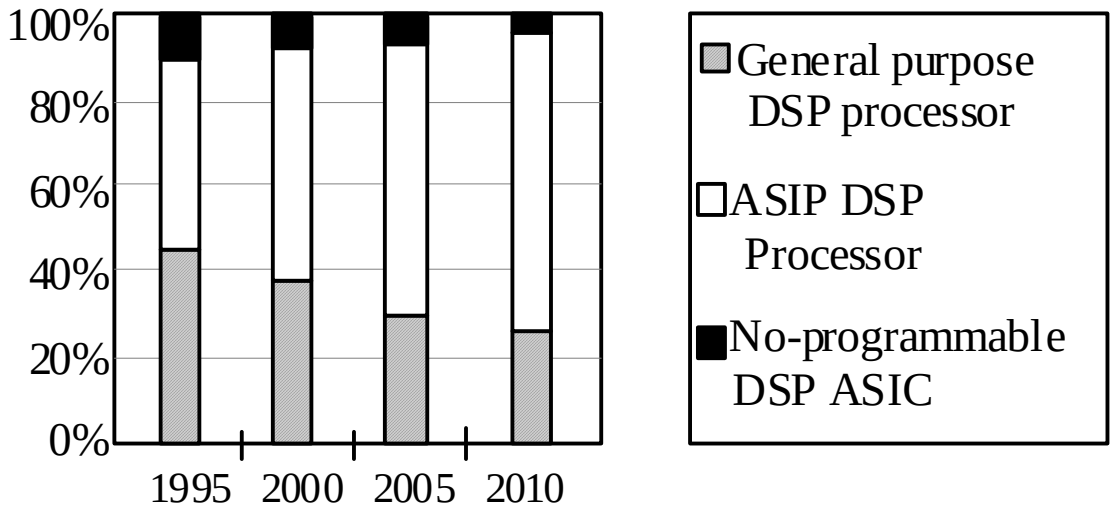
gives

- Function
- Quality
- Efficiency
- Reliability
- Fulfill your dream

The market of ASIP DSP



(a). DSP sales (Billion Dollar)



(b). DSP sales (market shares)

- The ASIP DSP market keeps increasing
- The DSP architectures are diverging for different application domains

DSP processor

- **Embedded system (ES) is the center part of most electronics products!**
- **Four essential parts in an ES**
  - **The DSP subsystem**
  - **The MCU subsystem**
  - **The memory subsystem**
  - **Peripherals, analog devices, sensors**
- **Want to know how a 3G phone, PDA, DVB…… works inside in detail?!**

Introduction to DSP

Basic concepts

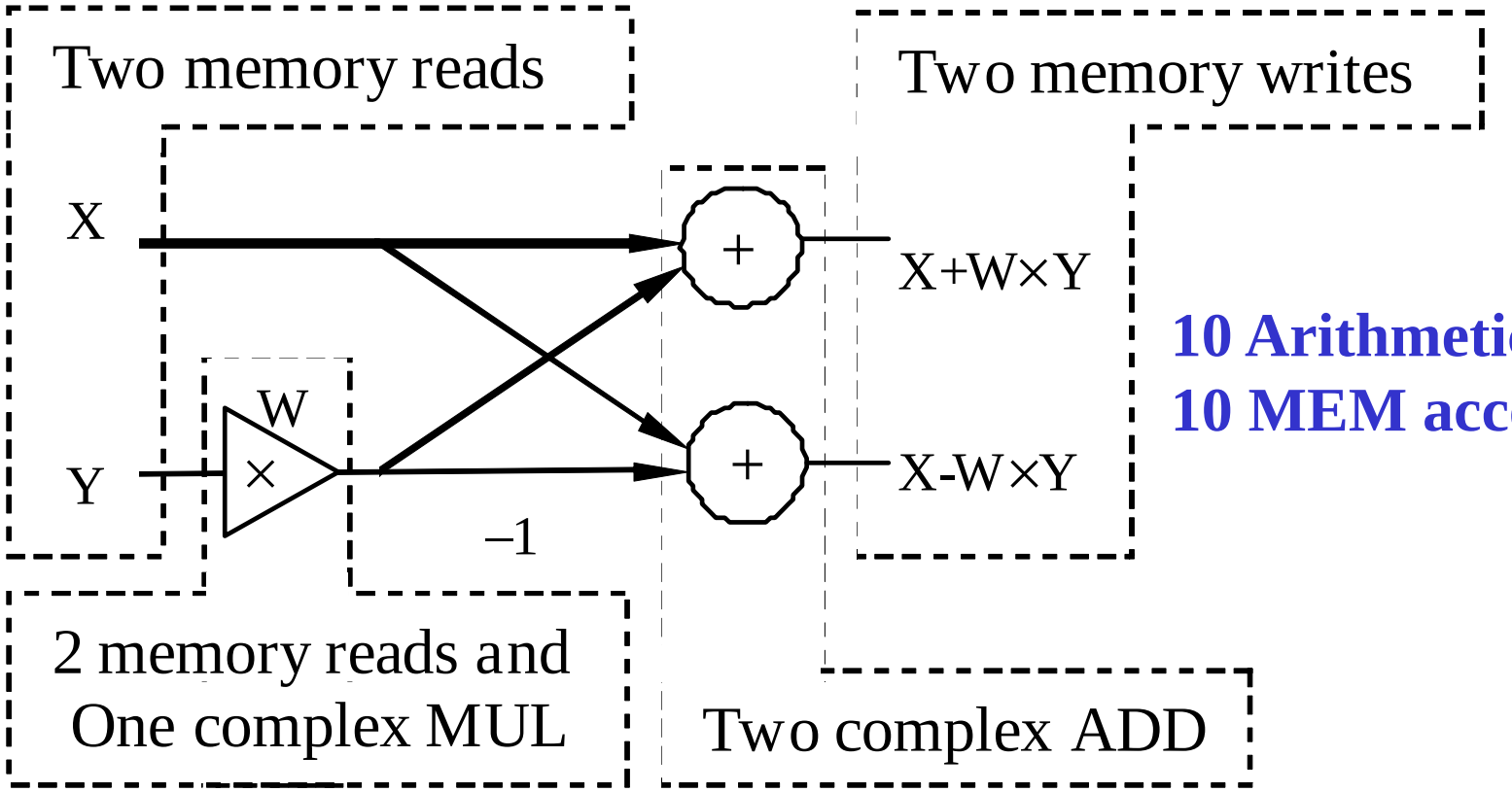
- **Algorithms**
  - ***mathematical representation of a task a DSP processor executes***
  - a group of ordered arithmetic operations formed, but does not specify how that arithmetic is implemented.
- **Arithmetic**
  - ***specified as a basic step of computing operations***
  - Basic arithmetic, for example: addition, subtraction, multiplication, division
  - Finite length arithmetic: such as guarding, saturation, truncation, and rounding

Costs of algorithms

- It is very important to understand the cost and types of costs of algorithms to be executed in an ASIP DSP
- The cost of arithmetic/logic computing
  - Number of arithmetic computing
  - Not yet include the cost of addressing computing
  - The number of memory access can be exposed
- The cost of data memory and registers
  - The data memory cost in included, the register cost is not yet included.

Understanding the cost

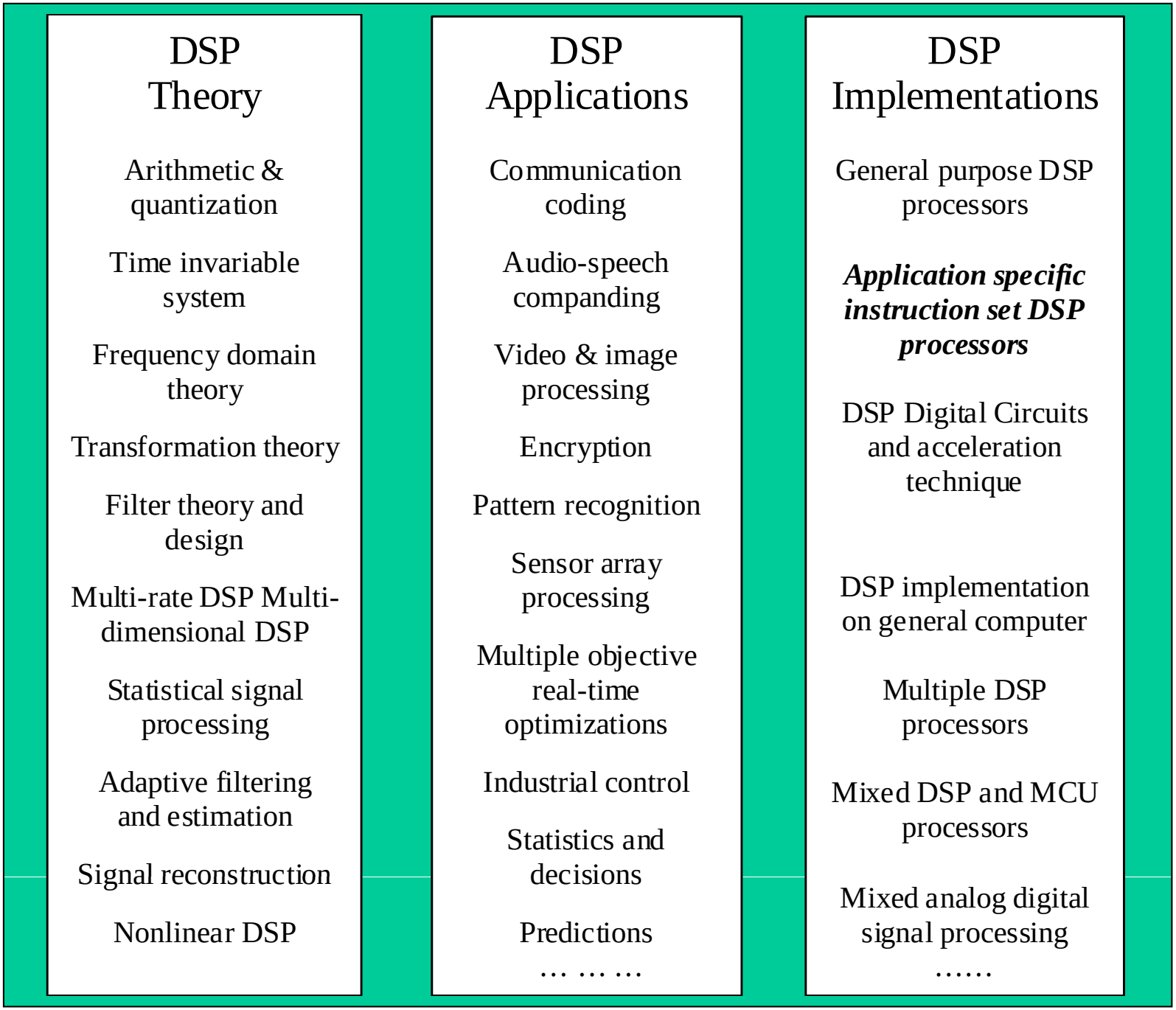
- For example: DIT butterfly in radix-2 FFT



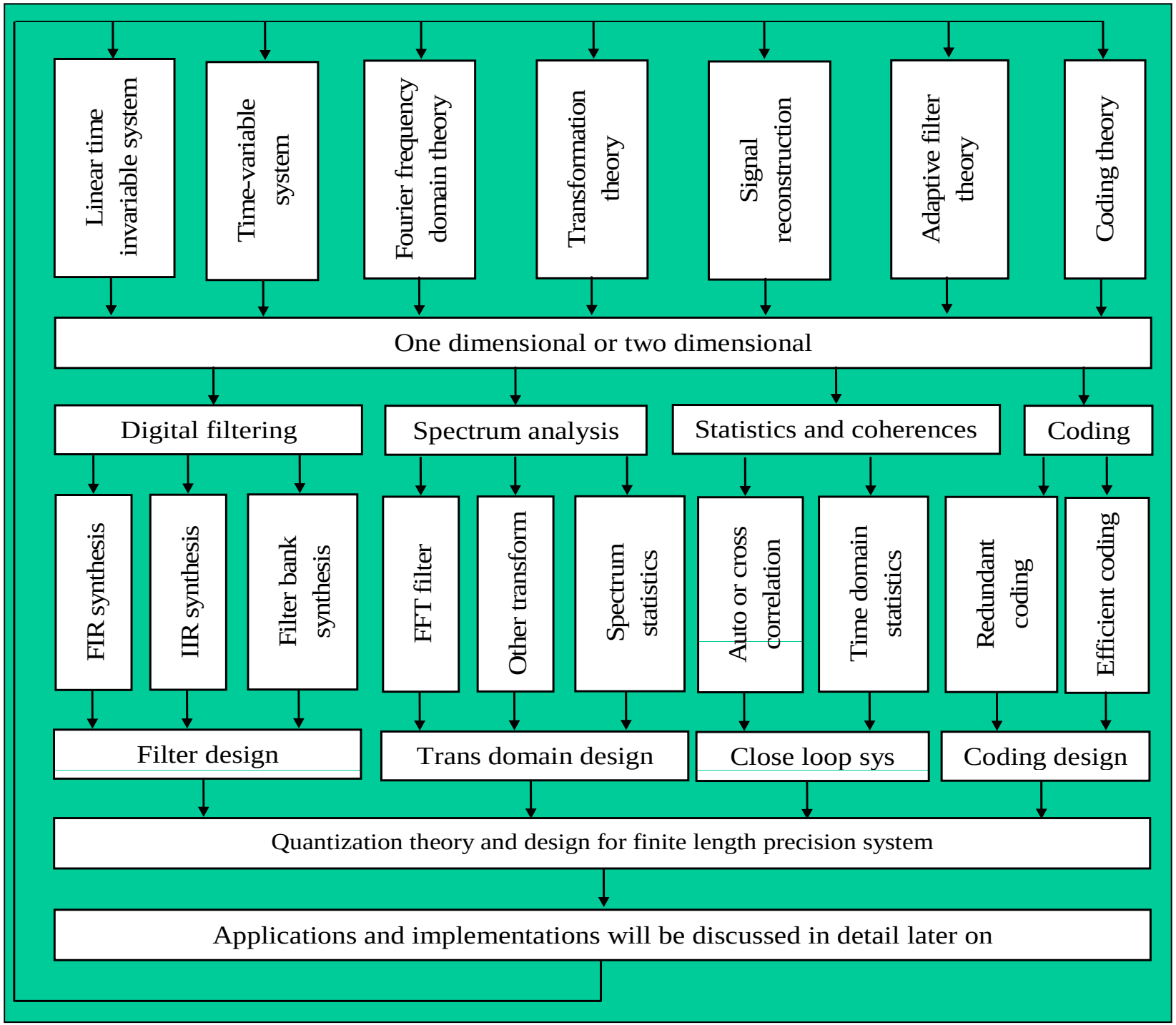
Definitions of DSP

- DSP theory
  - The application of theoretical mathematics
  - The basic way of modeling signal and systems
- DSP applications
  - Use DSP theory for a certain tasks
  - For example: A base-band specification, a CODEC
- DSP implementations
  - The industrial activities: to implement applications into either hardware or firmware (as *HdS*)

Definitions of DSP



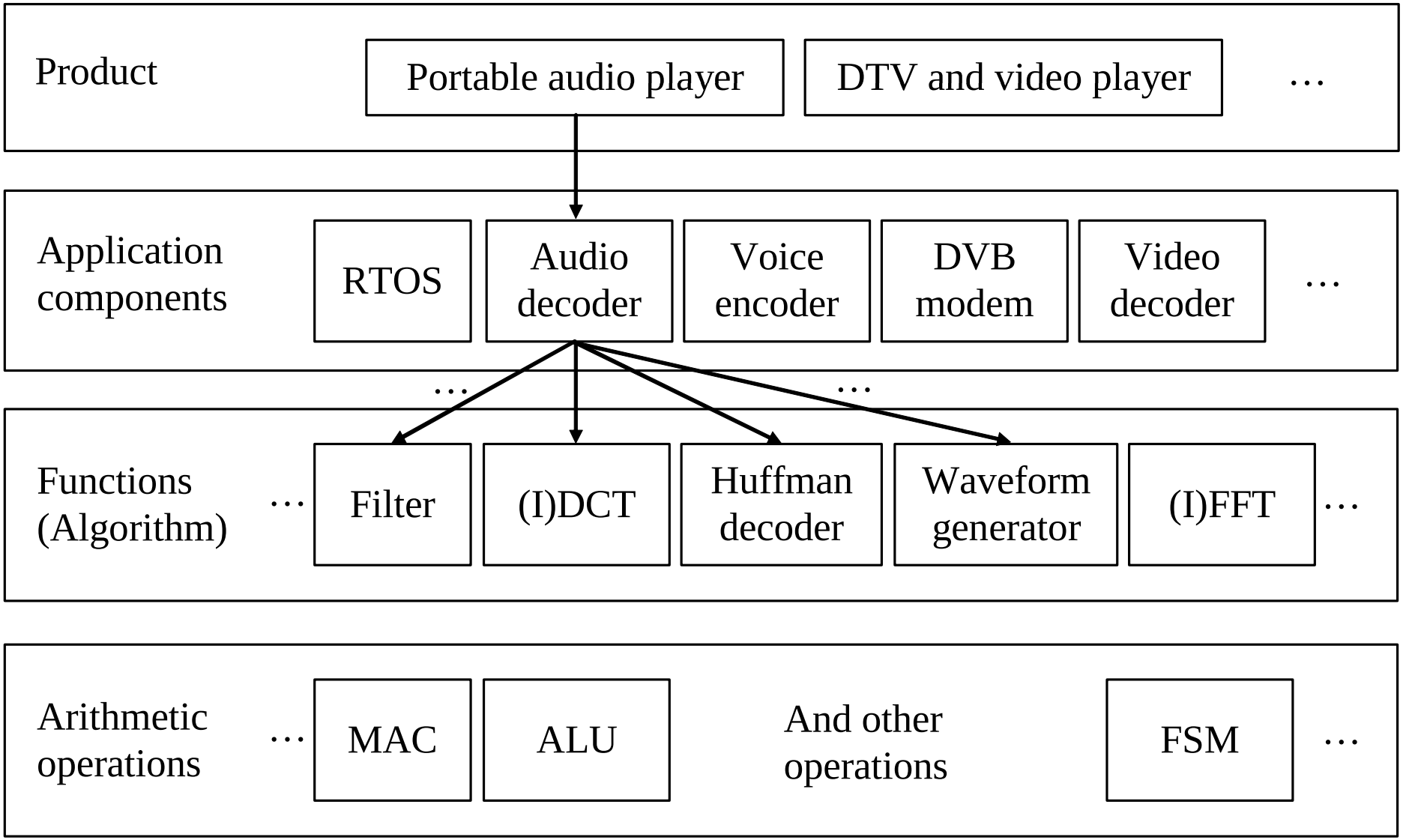
An out-look of DSP theory



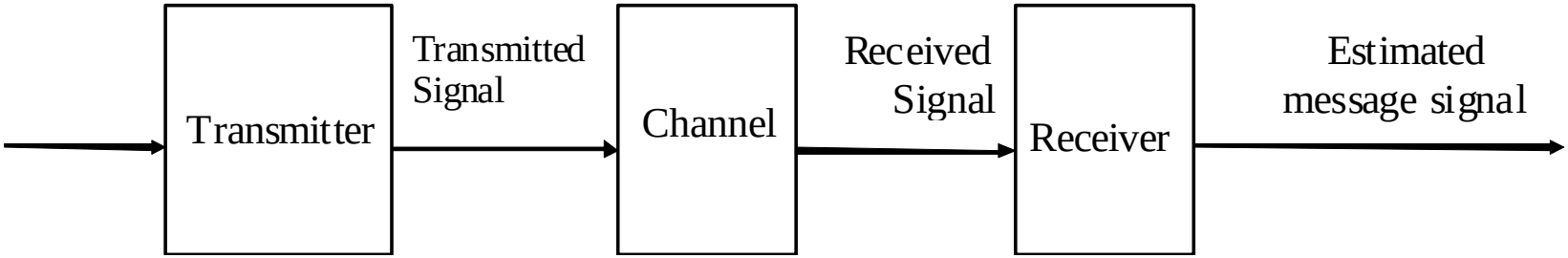
Understand Applications

- Classify applications
  - Applications following standards
  - Applications do not follow standards
- Real-time or not real-time applications
  - Real time: Finish the processing within a period

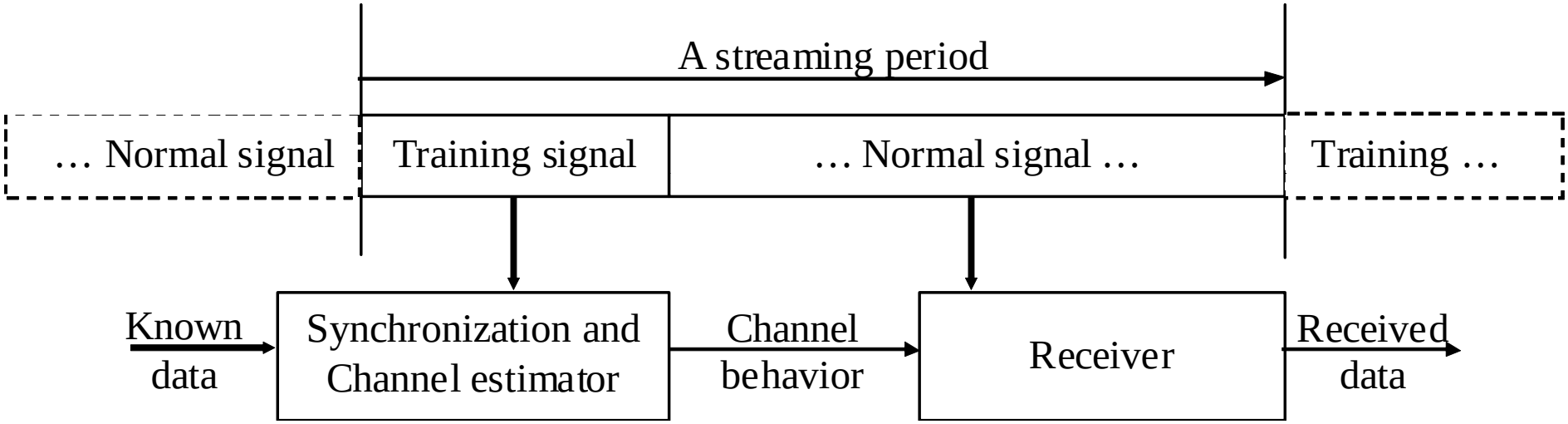
Understand Applications



Application example: Communication

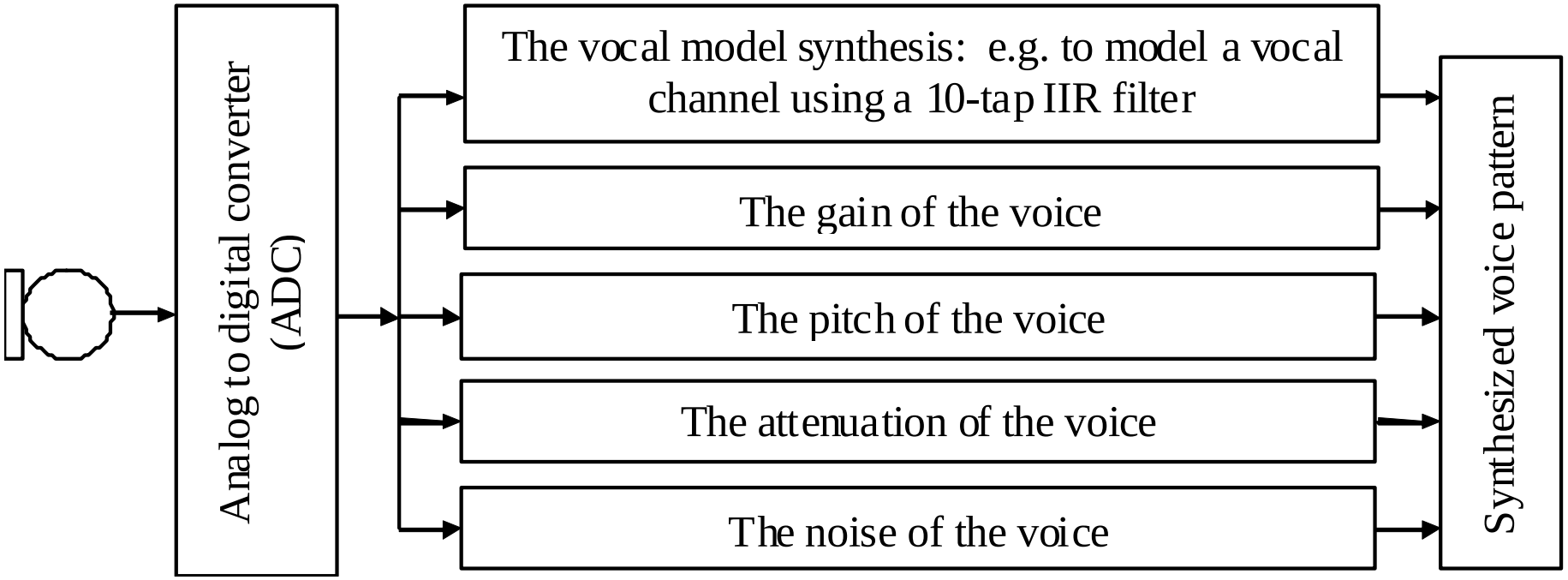


Streaming signal between a transceiver pair



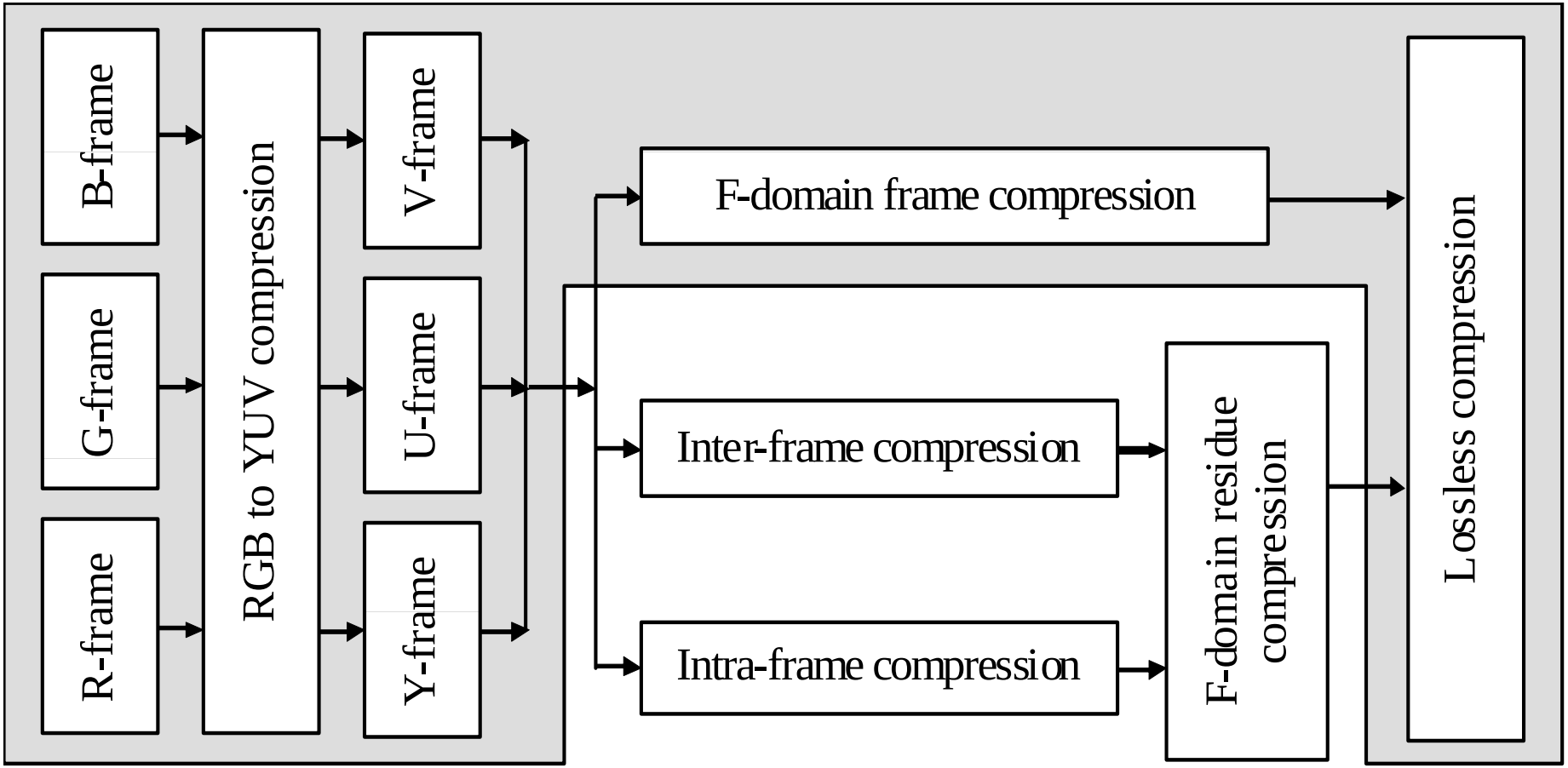
Recovering data from noise channel

Application example: Voice encoder



Can compress voice data from 104 kbits/s to 1.2 kbits/s

Application example: Image and video compression



Introduction to DSP Implementations

Examples of DSP implementations

- In a desk top computer – not the focus today
  - Window based for example DVD player, Audio player
- In general purpose DSP processors
  - Flexible instruction set, for many applications
  - High end; enhanced; low power low end
- In application specific instruction set processors
  - Specific instruction set, flexible in a limited area
- In an application specific integrated circuit
  - Not programmable

On a desk top or laptop

- Real time applications:
  - DVD decoder or MP3 decoder for example
  - SIMD: Single Instruction Multiple Data
  - Pentium II MMX (MultiMedia eXtensions) instruction
  - Pentium III SSE instruction (Streaming SIMD extension)
  - Accelerated SIMD instructions on packed integer and/or packed floating-point data elements contained in the 64-bit MMX or the 128-bit XMM registers.
  - Enables increased performance on a wide variety of multimedia and communications applications.
- Special support from the operating system

COTS DSP processors

- Not designed for specific users.
- Even through a general purpose, it has its own feature:
  - High end application: C6X
  - Low end application: C20
  - In between high end and low end: C55
- The instruction set architecture must be general.
- The Peripheral and interfaces must be comprehensive.
- No way to reach very low power, no way to reach high speed, and no way to reach very low cost.
- If the volume is not high or requirement is very high, a general purpose digital signal processor is preferred.
- Main development: software design
- Hardware design: limited to peripheral design

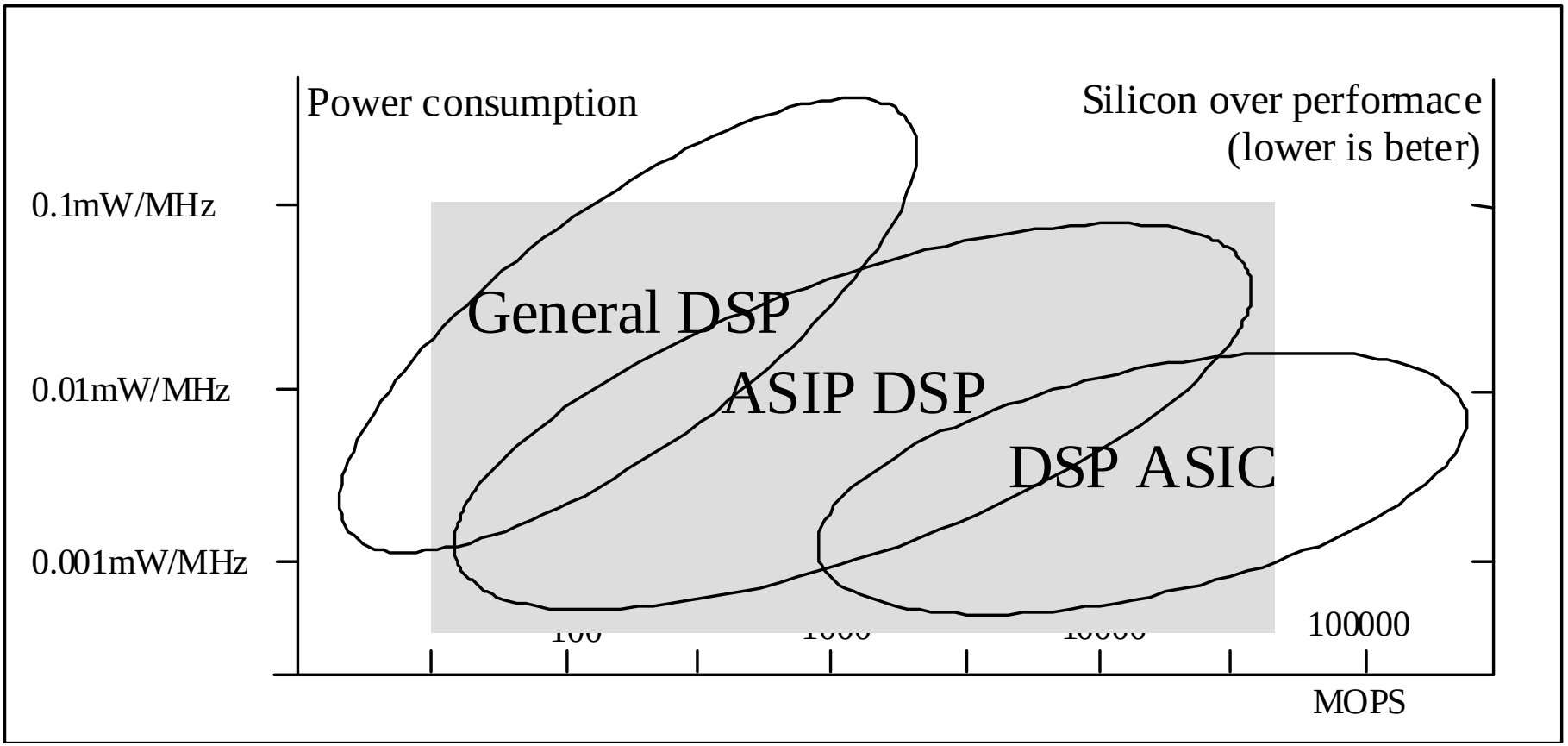
DSP ASIP

- **ASIP: Application Specific Instruction set Processor**
- **DSP ASIP: The highest volume** DSP processors
  - for a group or a kind of embedded applications
- Has its own strong and domain specific features.
- A special instruction set and its architecture.
- **It is the focus of the course!**
- Low power, high speed, and low cost.
- For example:
  - Base-band, voice processor, image processor
  - video processor, processor for motor control, for a sensor,…

Non-programmable DSP ASIC

- Non-program (might or should be configurable).
- Arithmetic / algorithm mapping to circuit.
- Very low power, very high speed, low silicon cost.
- Not flexible or not enough flexible
- Need (low cost) experts knowing both HW and SW
- Applications:
  - for extremely low power and/or high speed
  - when flexibility is not required at all
  - When the verification cost is low

Trade off of DSP solutions



DSP Architecture

- Program memory: part of control path
- Programmable FSM: consists of a PC and an instruction decoder
- Data memory and its addressing:
- Data processing unit: ALU, RF, MAC, and accelerated units
- Input/output unit: memory bus and peripherals are included
- Figure 1.19

DSP Firmware

- Firmware is fixed software running in an electronic product (device)
- Executable binary code is developed in 4 steps
  - Design source code
  - Develop HW dependent source code
  - Translate it to assembly code
  - Assemble/link/debug binary code
- Figure 1.20

Fixed-point DSP

- Cost: less than a half compared to a standard floating-point DSP
- Quantization error: minimized
- Dynamic range: maximized
- Functional firmware and data quality control firmware
- Figs. 1.21 and 1.22
  - main task flow,
  - measurement flow, and
  - scaling flow

Embedded Systems

- A keyboard controller
- A mouse controller
- A wireless modem
- A mobile phone
- A digital camera
- A digital TV
- Embedded systems are application specific
- Embedded system design covers almost all activities in the area of EE
- Just focus on DSP subsystems

DSP in an embedded systems

- Fig. 1.23: a general embedded system including a DSP subsystem
- Divided into four subsystems
  - MCU: OS, connection protocols, Java programs, human-machine interface, and HW management
  - ASIP DSP subsystem including accelerators
  - Memory subsystems
  - Peripherals

Basics of embedded computing

- Embedded computing can implemented with
  - ASIC
  - HW/SW (processors + accelerators), and SW
- Embedded SW
  - OS
  - SW for real-time computing
  - SW for best-effort computing
- Specific DSP platform for real-time SW
- Static scheduling can be use for embedded SW because of its complexity and dependency can be known before runtime.

What is the focus of the book

- There are many people working on DSP math!
- However, 90% activities in industry is the issues of implementation!
- The cost of different implementations
  - What is the kernel computing cost
  - What is the cost of memory access
  - What is so called “miscellaneous cost”
    - Program flow,
    - Control and dependency induced cost
    - Overhead of parallelization

How to measure the cost of an implementation

- Clock cycles
  - MHz per task
- Silicon cost
  - HW cost an arithmetic units
  - Memory cost
- Profiling – before the design
- Benchmarking – after the design

Cost of algorithm and memories

- Take a filter as an example
- We do not care how to get coefficients
- We do care the cost of FIR and IIR

$$y(n) = \sum_{k=0}^{K-1} a_k x(n-k) \qquad y(n) = \sum_{k=0}^{K-1} a_k x(n-k) - \sum_{l=1}^{L-1} b_l y(n-l)$$

- What is the cycle cost to run it?
- What is the data and code memory cost?

Cost of FIR and IIR

- FIR

$$y(n) = \sum_{k=0}^{K-1} a_k x(n-k)$$

  - K multiplications
  - 2K memory accesses
  - K accumulations
  - 4K operations

- IIR

$$y(n) = \sum_{k=0}^{K-1} a_k x(n-k) - \sum_{l=1}^{L-1} b_l y(n-l)$$

  - (K+L-1) multiplications
  - 2(K+L-1) memory accesses
  - (K+L-2) accumulations
  - 4(K+L-1)-1 operations

Design Flow

Design methodology

- A general design methodology for an electronic product is the development process for a system from conceptualization to manufacturing including modeling, implementation, and verification.
- The process consists of a set of specific tasks with a particular order to be executed and a set of CAD tools to be used during the execution of each task.

HW design flows in general

- Description or synthesis.
  - –Two basic ways of translations
    - capture–and–simulate
    - describe–and –synthesis

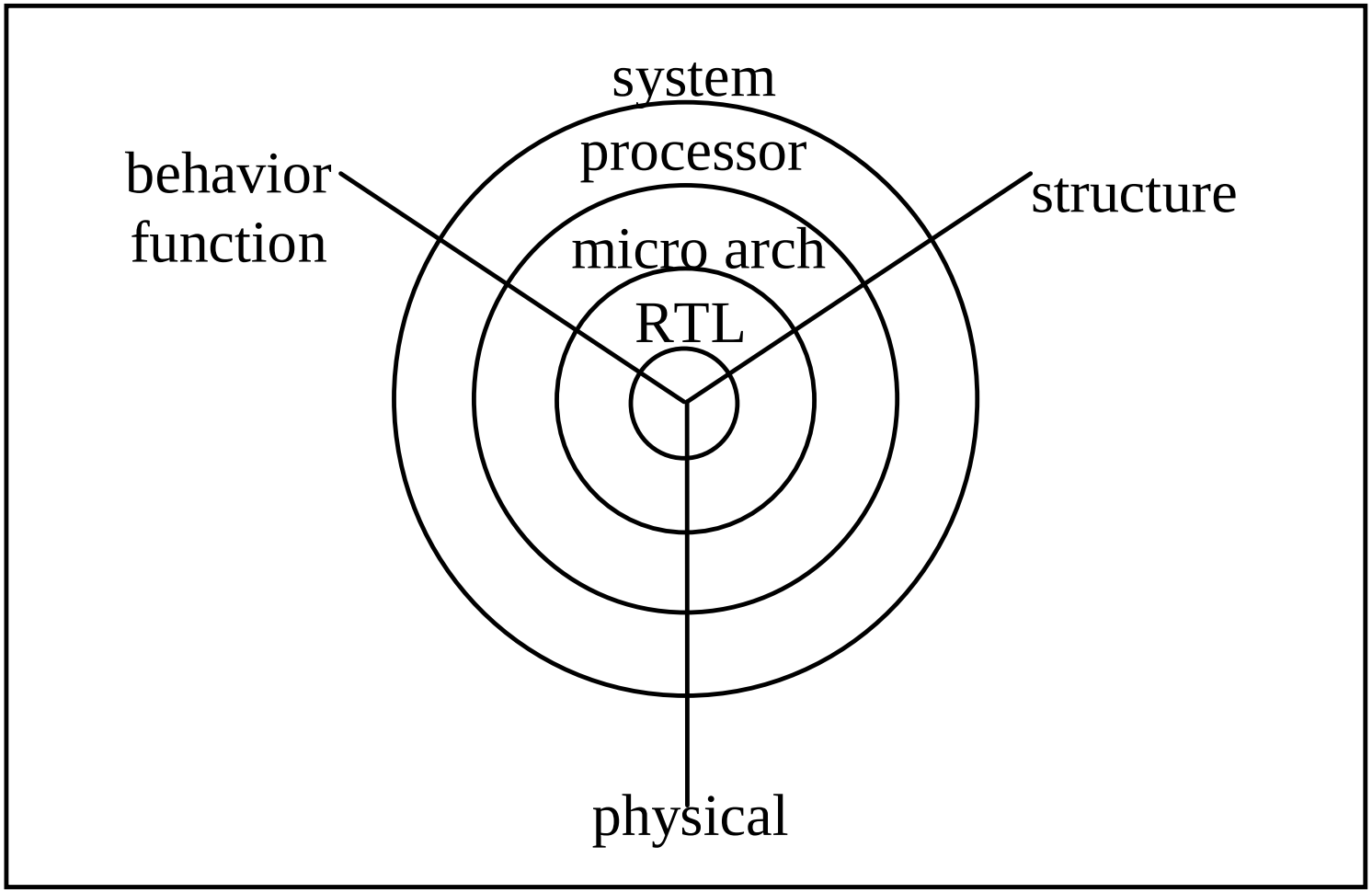
Capture–and–simulate

- Dominant methodology 60s to 80s.
  - Describe the system on every level
  - Prove the description by simulation.
- SW and HW design separated as a system gap
  - SW designers tested algorithms, wrote requirements
  - This specification to HW designers
- Do not know whether design fits the specification until the gate level design was captured and simulated

Describe–and –synthesis

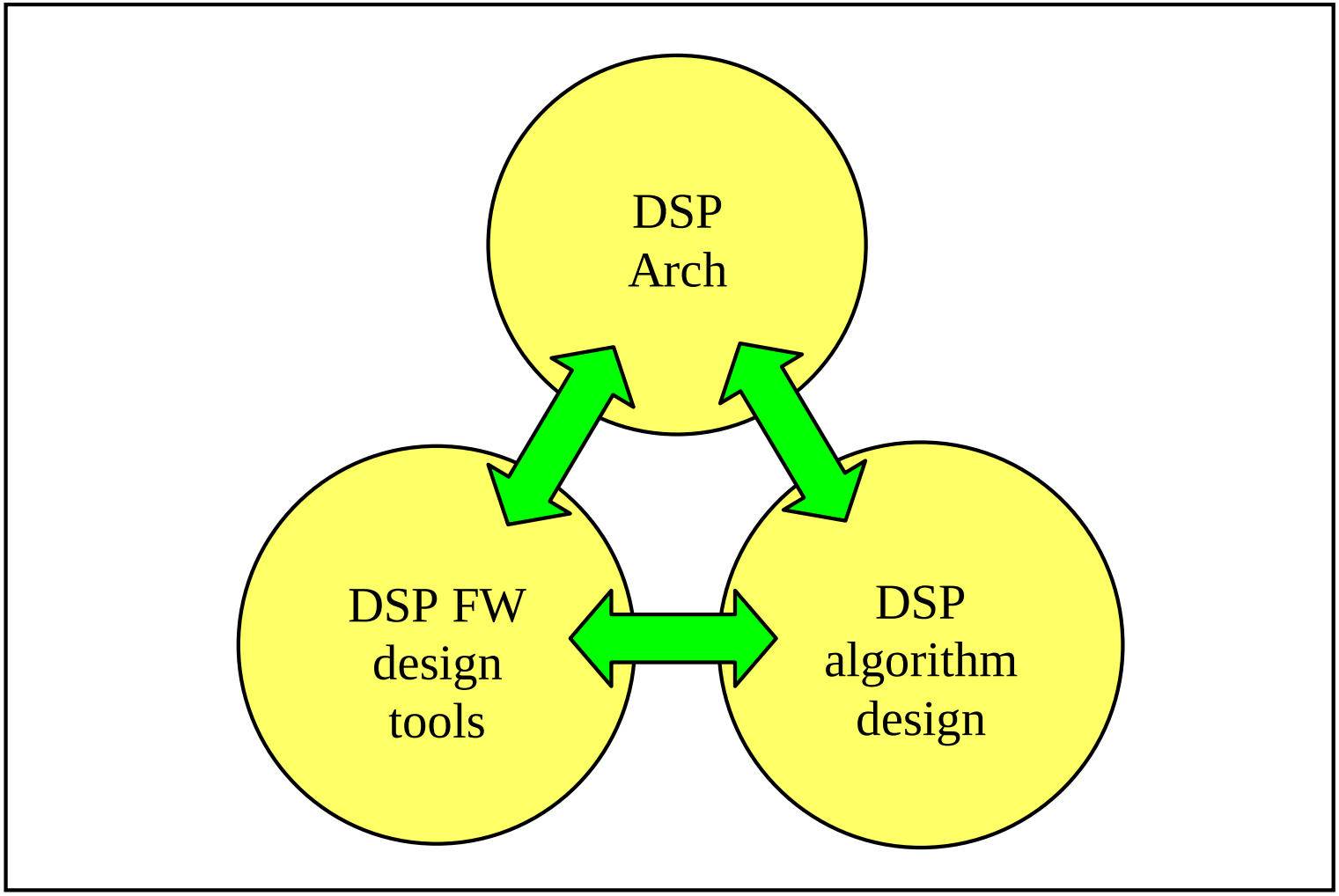
- Late 1980s logic synthesis.
- Higher level synthesis late 1990s.
  - the nightmare
- behavior synthesis
  - Too many choices / freedoms
  - not be the dominant design methodology in recent years.
  - Cannot optimize HW multiplexing, performance, power
- The lower level, the logic synthesis
  - translates RTL into gate level through logic synthesis
  - well developed and well used.

HW Design methodology



Y-chart: proposed by Prof. Daniel Gajski

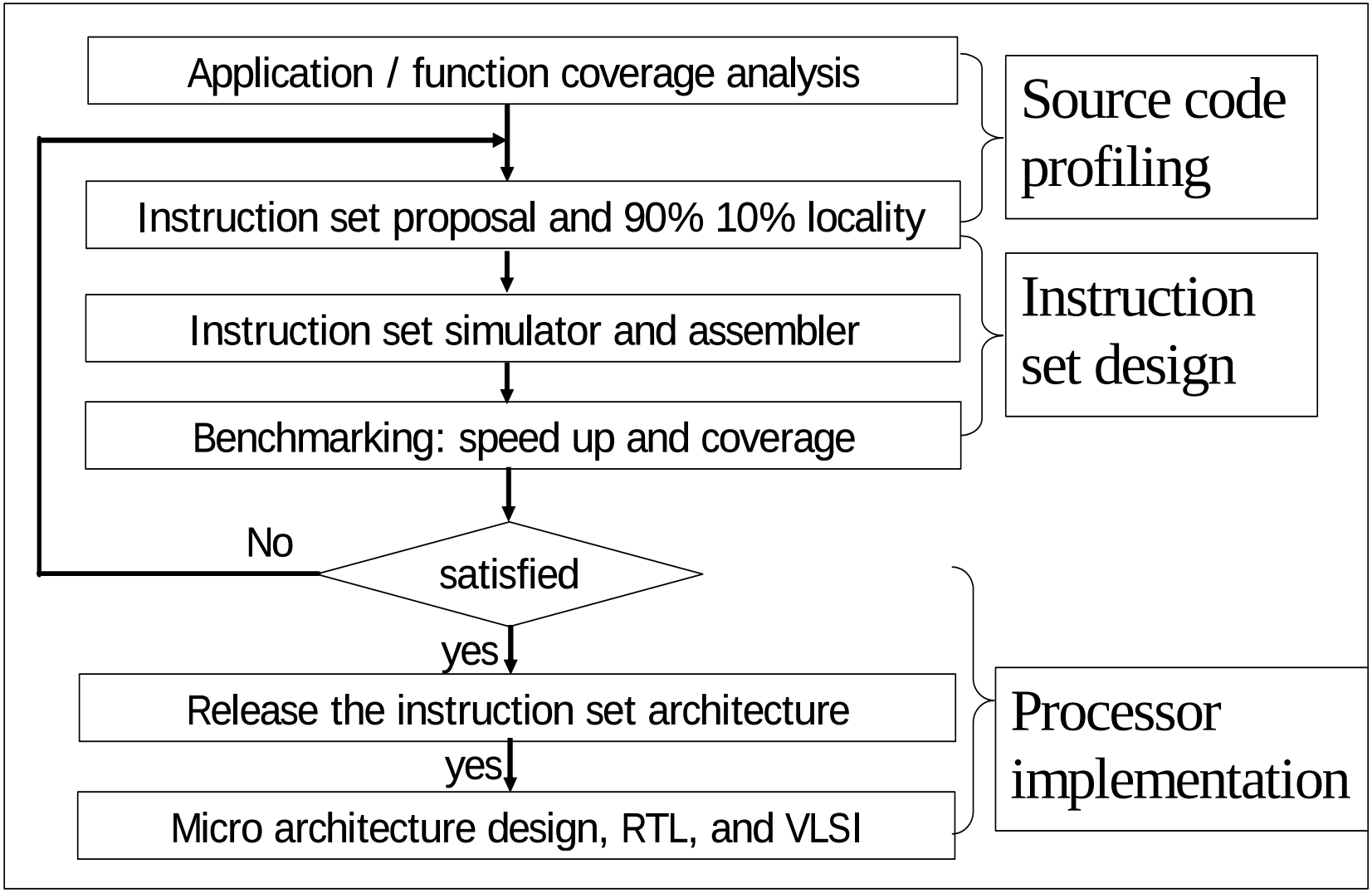
ASIP Design flow



Instruction set design

- Complicated
  - Difficult to claim which is the best
- Tradeoff among many parameters including
  - Performance, Functional coverage, Flexibility, Power consumption, Silicon cost, Design time

ASIP Design flow (chap 4)



Application analysis

- Application coverage should be specified first
  - Understanding standards
  - Partitioning
- Then translated into functional (algorithmic) coverage
  - Difficult to add extra features for future usages

ASIP ISA design

- Code profiling (chap 6)
  - which function should be accelerated?
- Instruction set design (chap 7)
  - 10% - 90% code locality rule
- SW development tool chain (chap 8)
  - Compiler, assembler, simulator
- Evaluation of an instruction set (chap 9)

ASIP architecture design

- DSP architecture (chap 3)
  - Datapath, control, bus, memory
  - ISA
- DSP micro-architecture (chap 10)
  - Functional modules (alu, rf, mac, mux..)
  - Specification for RTL coding
  - Microoperations

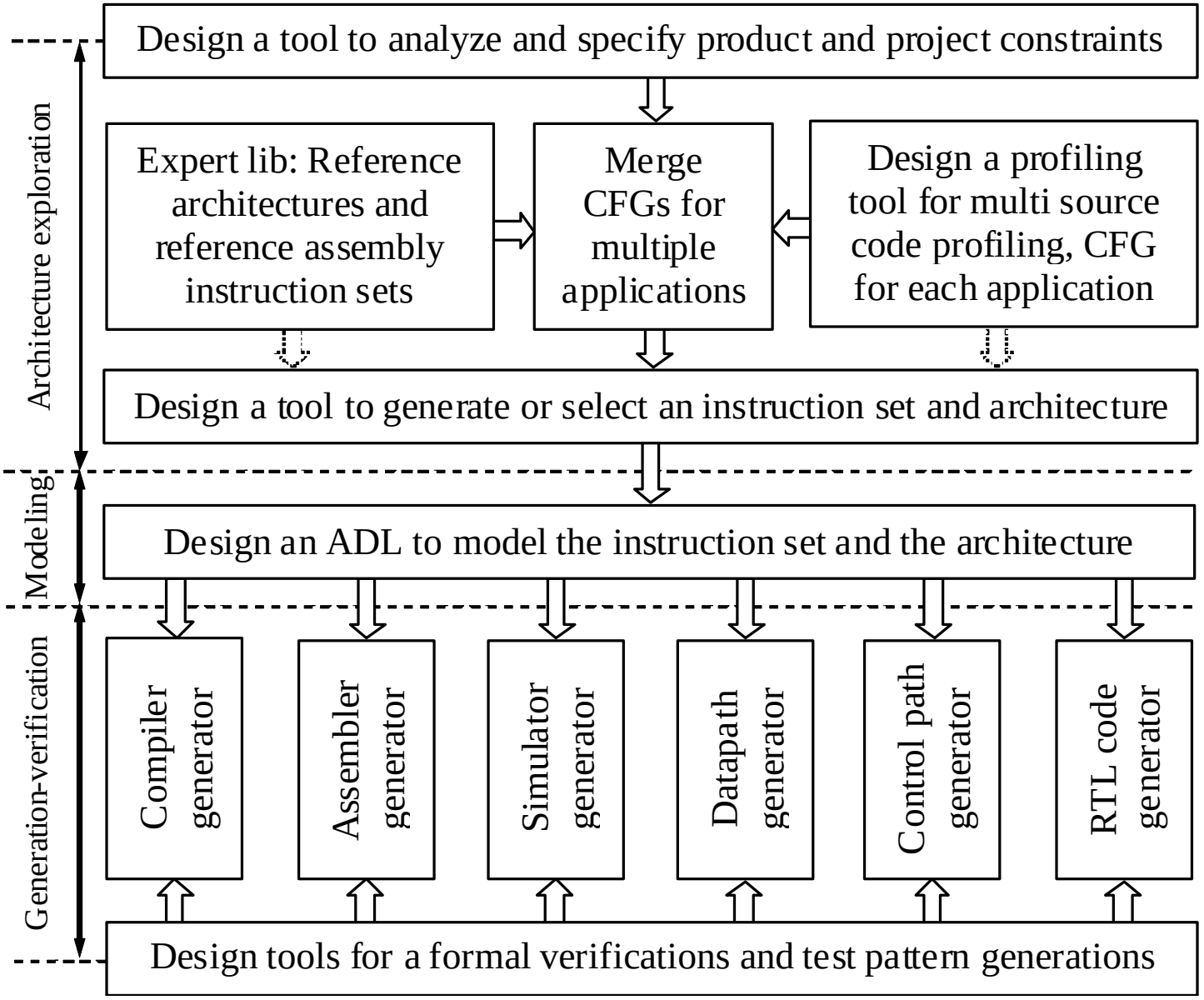
ASIP architecture design

- Register file (chap 11)
- ALU (chap 12)
- MAC (chap 13)
- Control path (chap 14)
- Memory subsystem (chap 15)
- Peripherals (chap 16)

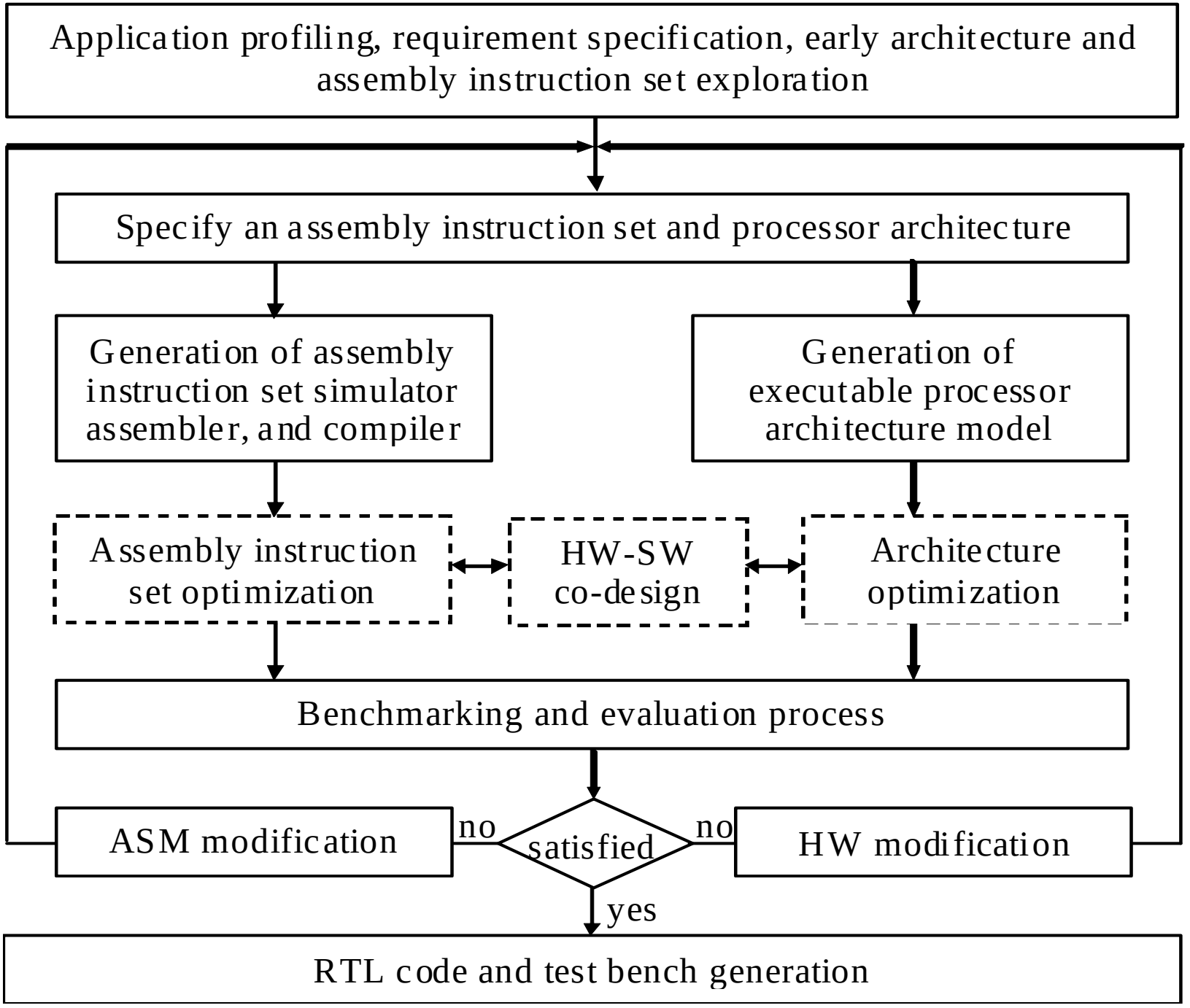
ASIP Design flow automation

1. Architecture exploration (selecting or generating an architecture and IS according to the application analyses).
2. Specify an ADL (Architecture Description Language) to model the instruction set and architecture
3. Generations and verifications of tools and architectures

Tool researchers' view on ASIP



Designers' view on ASIP



Available ASIP Design tools

Tool or solution	Profiling and architecture exploring	Compiler generator	Assembler generator	Simulator generator	Cycle accurate architecture behavior model generator	Instruction set and architecture Optimizer	Datapath RTL code generator	Control path RTL code generator	Limited by an architecture
MIMOLA				yes			yes	yes	no
Cathedral-II	yes			yes	yes		yes	yes	no
Target	yes	yes	yes	yes	yes		yes	yes	no
ARC		yes	yes	yes	yes		yes	yes	limited
Tensilica	yes	yes	yes	yes	yes	yes	yes	yes	limited
LISA	yes	yes	yes	yes	yes		yes	yes	limited
MESCAL	yes		yes	yes	yes		yes	yes	no
PEAS-III			yes	yes	yes		yes	yes	limited
NoGAP	yes		yes	yes	yes		yes	yes	no

Review

- The scope of DSP is huge:
  - Theory, applications, implementations.
- The focus is hardware implementation of ASIP for DSP applications.
- Background knowledge was reviewed.
- Suggested to read references books