INTRODUCTION TO DIGITAL SYSTEMS

- DESCRIPTION AND DESIGN OF DIGITAL SYSTEMS
- FORMAL BASIS: SWITCHING ALGEBRA
- IMPLEMENTATION: MODULES (ICs) AND NETWORKS
- IMPLEMENTATION OF ALGORITHMS IN “HARDWARE”
- COURSE EMPHASIS: CONCEPTS, ANALYSIS AND DESIGN
- Follow-on courses:
  - Computer Architecture CS151B;
  - Digital Design - Advanced Topics CS 151C
  - Digital Lab CS152A; Computer Architecture Lab CS 152B

OVERVIEW

- WHAT IS A DIGITAL SYSTEM?
- HOW IT DIFFERS FROM AN ANALOG SYSTEM?
- WHY ARE DIGITAL SYSTEMS IMPORTANT?
- BASIC TYPES OF DIGITAL SYSTEMS: COMBINATIONAL AND SEQUENTIAL
- SPECIFICATION AND IMPLEMENTATION OF DIGITAL SYSTEMS
- ANALYSIS AND DESIGN OF DIGITAL SYSTEMS
- DESIGN PROCESS AND CAD TOOLS

WHAT IS DIGITAL?

DIGITAL SYSTEMS

– inputs and outputs:
finite number of discrete values

ANALOG SYSTEMS

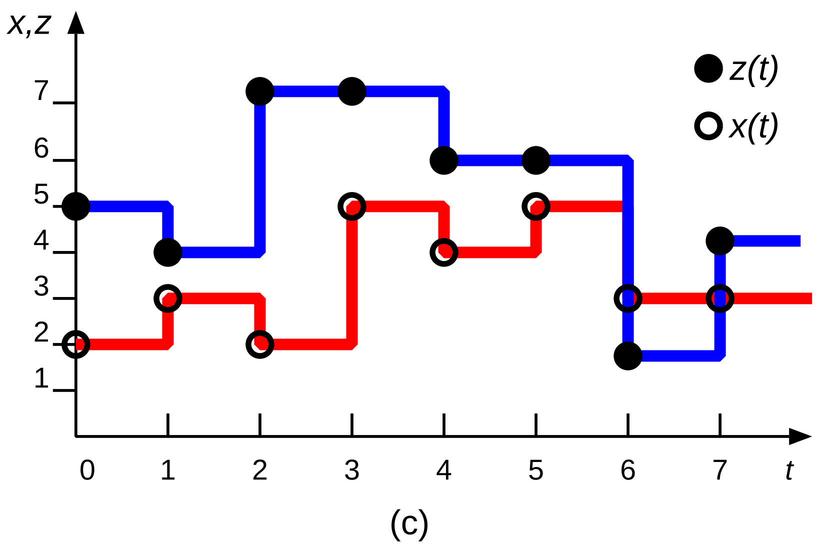
– inputs and output values:
from a continuous (infinite) set

Example: digital vs. analog scale for measuring weights

MAIN USES OF DIGITAL SYSTEMS:

- INFORMATION PROCESSING (text, audio, visual, video)
- TRANSMISSION (communication)
- STORAGE

SYSTEM AND SIGNALS



t	0	1	2	3	4	5	6	7
$x(t)$	2	3	2	5	4	5	3	3
$z(t)$	5	4	7	7	6	6	2	4

(d)

Figure 1.1: System S: a) Block diagram. b) Analog I/O signals. c) Digital I/O signals. d) I/O sequence pair.

WHY DIGITAL

1. FOR BOTH NUMERICAL AND NONNUMERICAL INFORMATION PROCESSING
2. INFORMATION PROCESSING CAN USE A GENERAL-PURPOSE SYSTEM (a *computer*)
3. DIGITAL REPRESENTATION:

   – vector of signals with just two values (*binary signals*)

   Example:

digit	0	1	2	3	4	5	6	7	8	9
vector	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

   – All signals binary

   – Simple devices to process binary signals:

   (SWITCHES with two STATES: open and closed).

WHY DIGITAL (cont.)

4. DIGITAL SIGNALS INSENSITIVE TO VARIATIONS OF COMPONENT PARAMETER VALUES

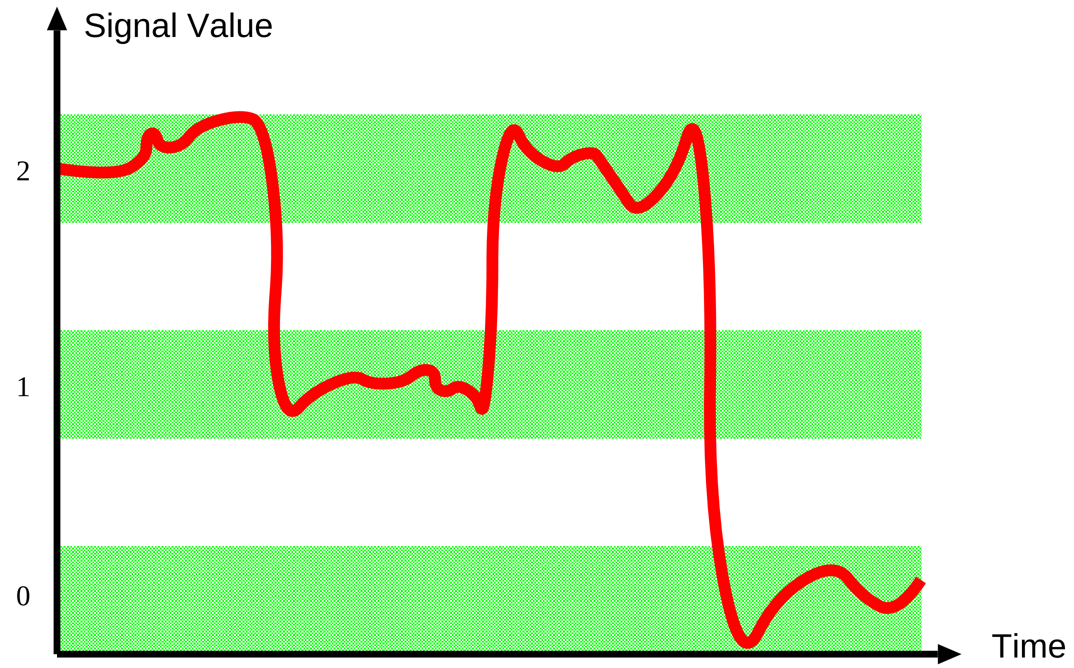


Figure 1.2: Separation of digital signal values.

WHY DIGITAL (cont.)

5. Numerical digital systems can be made MORE ACCURATE by simply increasing the number of digits used in the representation.

6. PHENOMENAL ADVANCES OF MICROELECTRONICS TECHNOLOGY:

   - Possible to fabricate extremely complex digital systems, which are small, fast, and cheap

   - Digital systems built as *integrated circuits* composed of a large number of very simple devices

WHY DIGITAL (cont.)

7. DIFFERENT IMPLEMENTATIONS OF SYSTEMS WHICH TRADE-OFF SPEED AND AMOUNT OF HARDWARE (COST)

Example:

– add two integers represented by six decimal digits

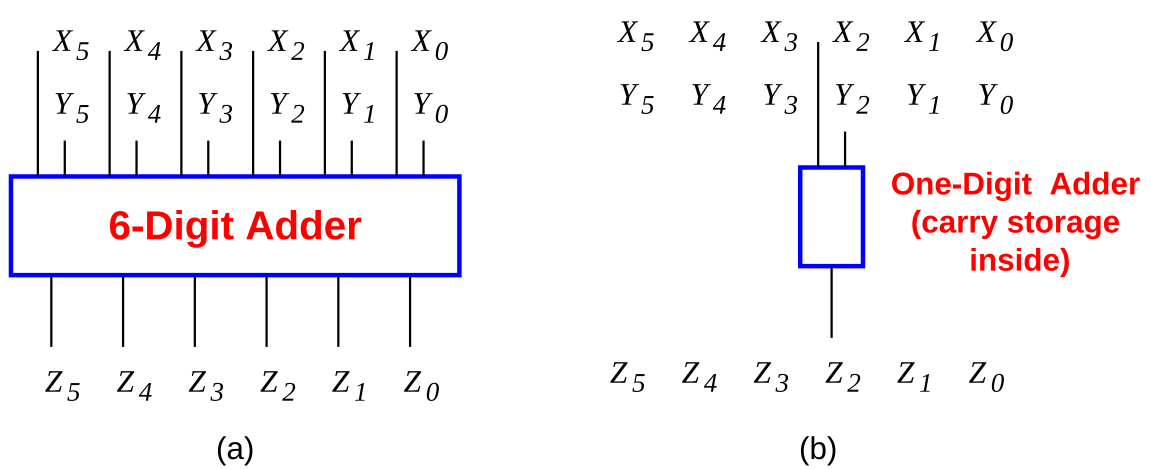


Figure 1.3: Six-digit adder: a) Parallel implementation. b) Serial implementation.

SUMMARY

- DIGITAL REPRESENTATION AND PROCESSING METHODS WIDELY USED
- EXTRAORDINARY PROGRESS IN DIGITAL TECHNOLOGY AND USE
- INDISPENSABLE IN MODERN SOCIETY
- NEW APPLICATIONS FUELED BY THE DEVELOPMENT OF COMPUTER TECHNOLOGY
- KNOWLEDGE ABOUT THE DESIGN AND USE OF DIGITAL SYSTEMS REQUIRED IN A LARGE VARIETY OF HUMAN ACTIVITIES

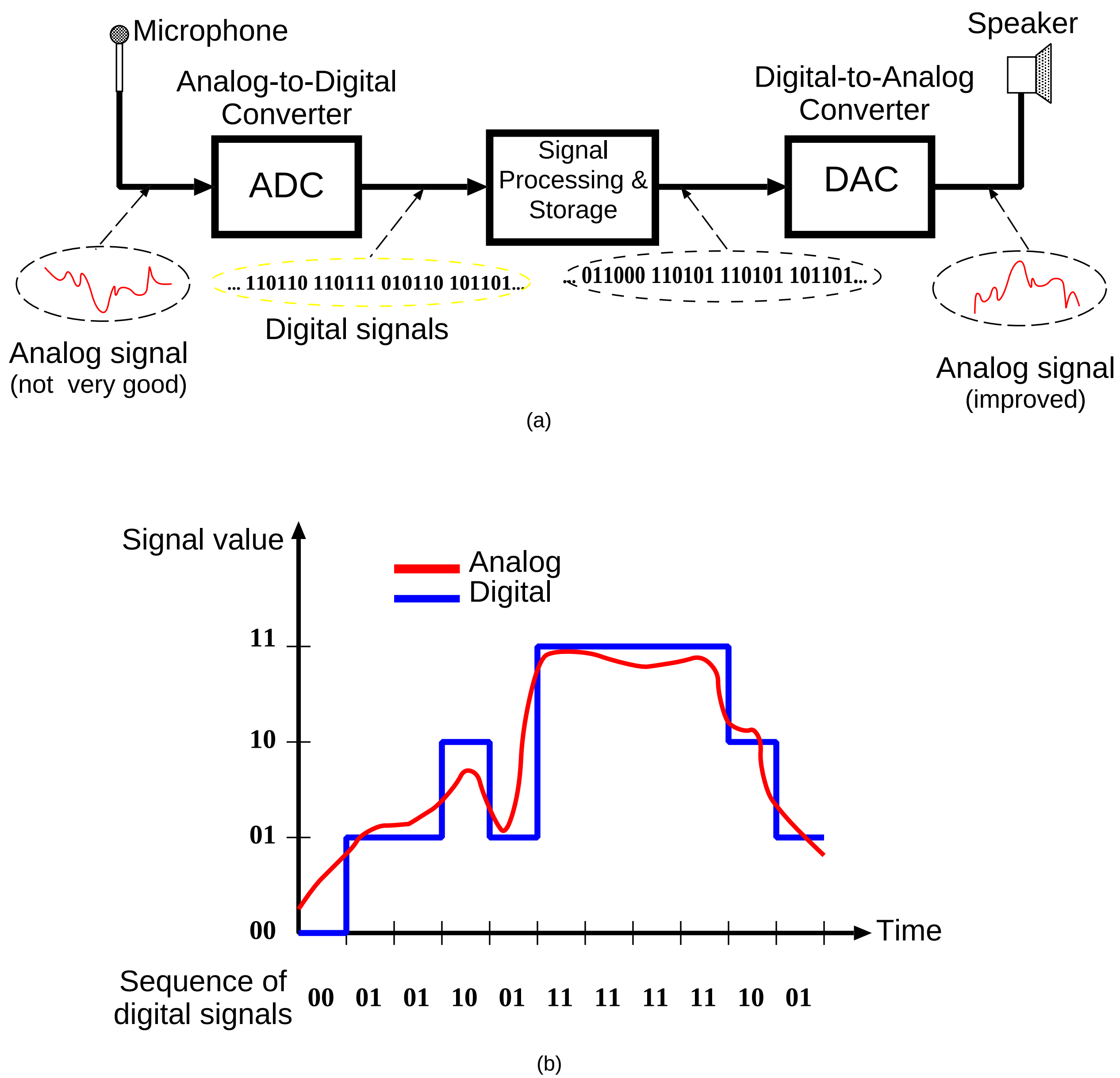


Figure 1.4: a) A system with analog and digital signals. b) Analog-to-digital conversion.

COMBINATIONAL AND SEQUENTIAL SYSTEMS

- DIGITAL SYSTEMS - TWO CLASSES:
- *COMBINATIONAL SYSTEMS*

$$z(t) = F(x(t))$$

– no memory, the output does not depend on previous inputs

- *SEQUENTIAL SYSTEMS*

$$z(t) = F(x(0,t))$$

$x(0,t)$: input sequence from time 0 to time t

– $z(t)$ depends also on previous inputs - the system has MEMORY

COMBINATIONAL AND SEQUENTIAL SYSTEMS (cont.)

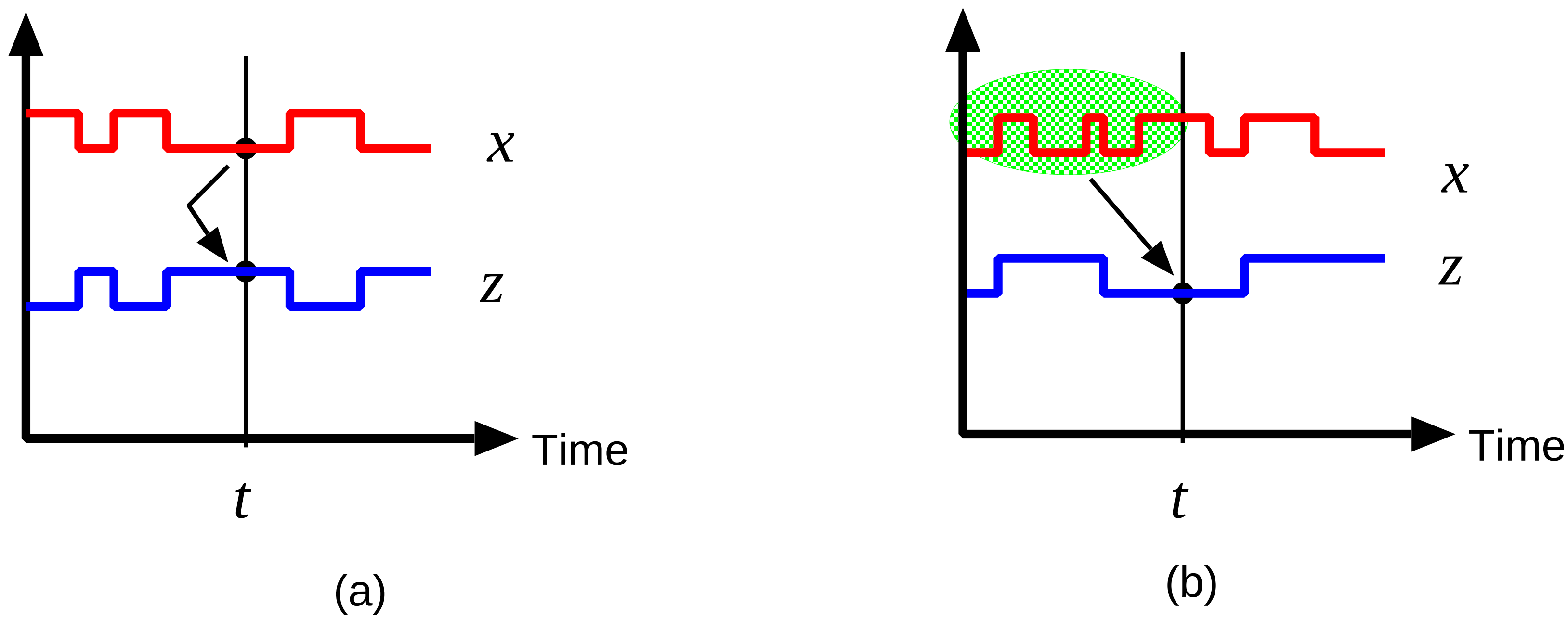


Figure 1.5: Input-output functions for: a) Combinational system; b) Sequential system.

EXAMPLE 1.1: SEQUENTIAL SYSTEM

- INPUT x with VALUES 0,1, or 2
- OUTPUT z with VALUES 0 or 1
- FUNCTION:

$$z(t) = \begin{cases} 1 \textbf{ if } & (x(0), x(1), \ldots, x(t)) \text{ has} \\ & \quad \text{even } 2' \text{s and odd } 1' \text{s} \\ 0 \textbf{ otherwise} & \end{cases}$$

- AN INPUT-OUTPUT PAIR:

t	0	1	2	3	4	5	6	7	8	9	10	11
x	1	2	2	0	1	2	0	0	0	2	1	1
z	1	0	1	1	0	0	0	0	0	0	1	0

EXAMPLE 1.2: COMBINATIONAL SYSTEM

- INPUT $x(t)$ with values from the set of letters (upper and lower case)
- INPUT $y(t)$ with values 0 and 1
- FUNCTION:
  - change $x(t)$ to opposite case when $y(t) = 1$
  - leave it unchanged when $y(t) = 0$
- AN INPUT-OUTPUT PAIR:

t	0	1	2	3	4	5	6
x	E	X	A	M	P	L	E
y	0	1	0	0	0	1	0
z	E	x	A	M	P	l	E

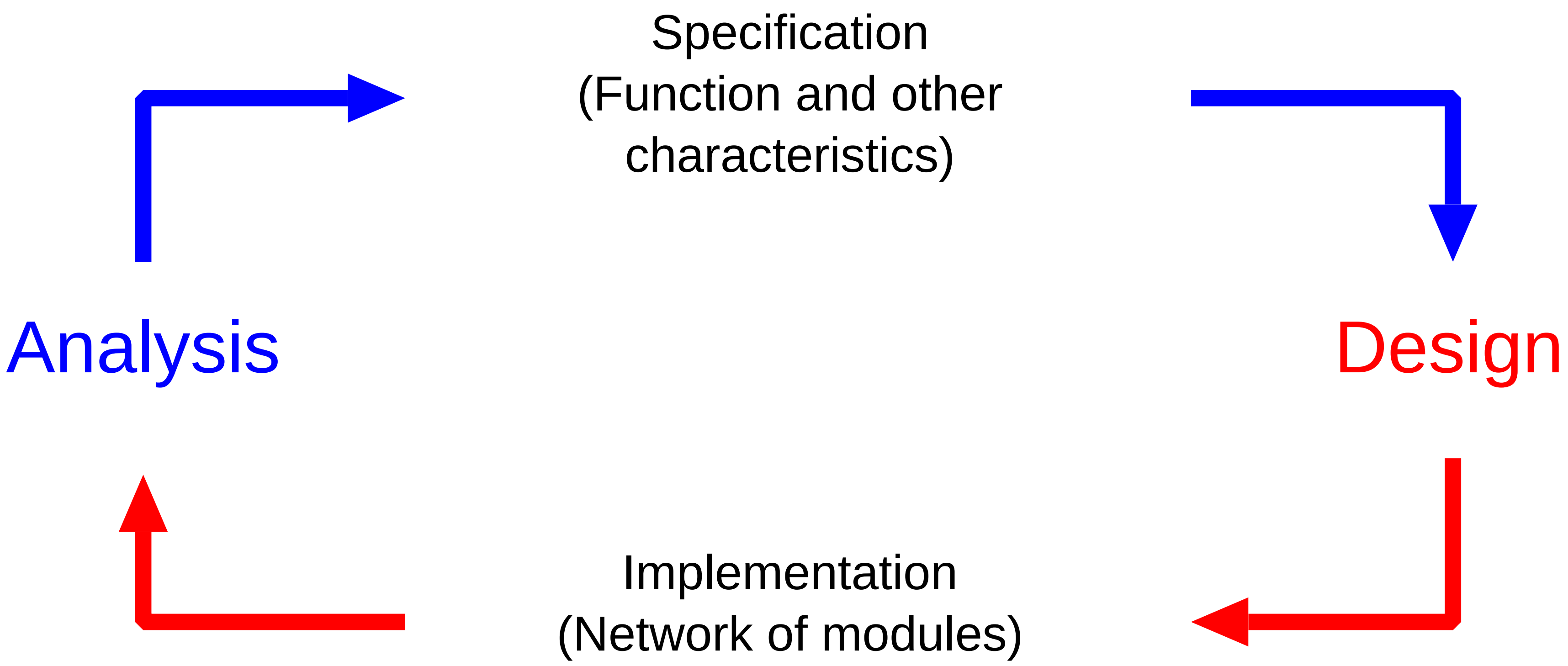


Figure 1.6: Relationship among system specification and implementation.

SPECIFICATION AND IMPLEMENTATION (cont.)

SPECIFICATION of a system describes its function.

Objective:

- to use the system as a component in more complex systems; and
- to serve as the basis for the implementation of the system by a network of simpler components.

SPECIFICATION LEVELS

- HIGH-LEVEL
- BINARY-LEVEL
- ALGORITHMIC-LEVEL
- Spec. of combinational systems: Chapter 2
- Spec. of sequential systems: Chapter 7
- Spec. of algorithmic systems: Chapter 13

IMPLEMENTATION

As a *DIGITAL NETWORK* – interconnection of modules

- SEVERAL LEVELS depending on the complexity of the primitive modules
  - from very simple *gates* to complex *processors*
- Need for *HIERARCHICAL IMPLEMENTATION*
- PHYSICAL LEVEL: interconnection of electronic elements such as transistors, resistors, and so on (Chapter 3).

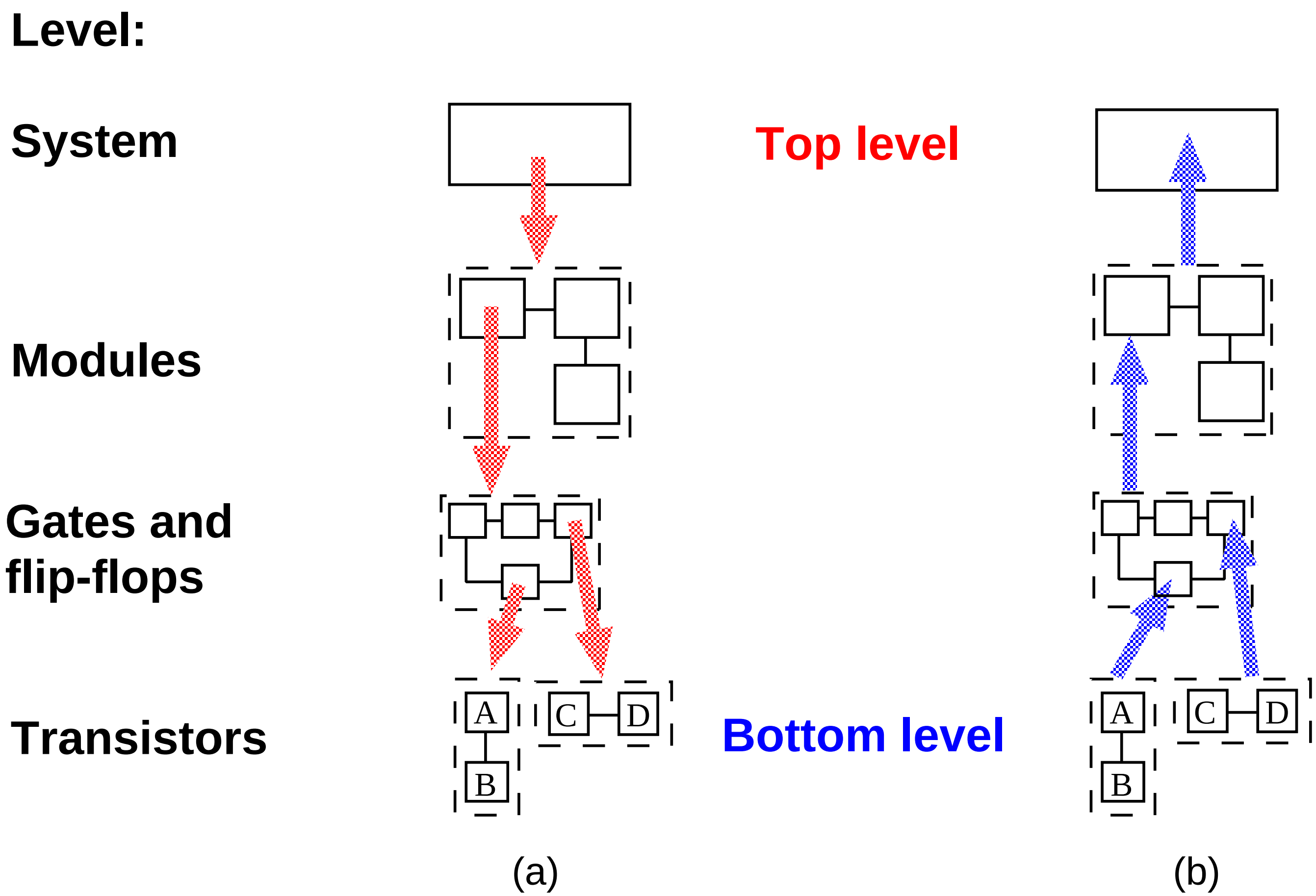


Figure 1.7: Hierarchical implementation: a) Top-down approach. b) Bottom-up approach.

COMMENTS ON IMPLEMENTATION

- Modules as conceptual entities
  – to simplify description of an implementation
- Standard modules of several levels of complexity
  – to facilitate design of large numbers of different systems
- Rules for interconnection of modules
- Why separation of the specification of a system from its implementation?
  – to shield the description from irrelevant implementation details;
  – to allow choosing an implementation from different alternatives,

- IMPLEMENTATION OF COMBINATIONAL SYSTEMS
  - at the gate level: Chapter 5 (and 6 - not covered in this course)
  - at the module level: Chapters 9, 10, and 12
- IMPLEMENTATION OF SEQUENTIAL SYSTEMS
  - elementary: Chapter 8
  - more complex: Chapters 11 and 12
- IMPLEMENTATION OF ALGORITHMIC SYSTEMS: Chapters 13-15

STRUCTURED ANALYSIS AND DESIGN

- *ANALYSIS*:

  * get specification from an implementation

- *DESIGN*:

  * obtain an implementation that satisfies the specification

- Use of MULTILEVEL APPROACH necessary
- The TOP-DOWN and BOTTOM-UP approaches
- A combination of the two approaches

LEVELS OF AN IMPLEMENTATION: MODULE, LOGICAL, PHYSICAL

Problem: Compute sum $Z(t)$ of $t+1$ inputs $X(i)$

$$Z(t) = \sum_{i=0}^{t} X(i) \quad (function)$$

$$Z(i) = Z(i-1) + X(i) \quad (algorithm)$$

$$Z(-1) = 0$$

- Sequential algorithm – sequential implementation of *datapath*
- One input and one output per cycle
- One operation: addition (operator: ADD); *control* not shown
- Two variables: input $X(i)$ and running sum $Z(i)$ (Registers: RX and RY)

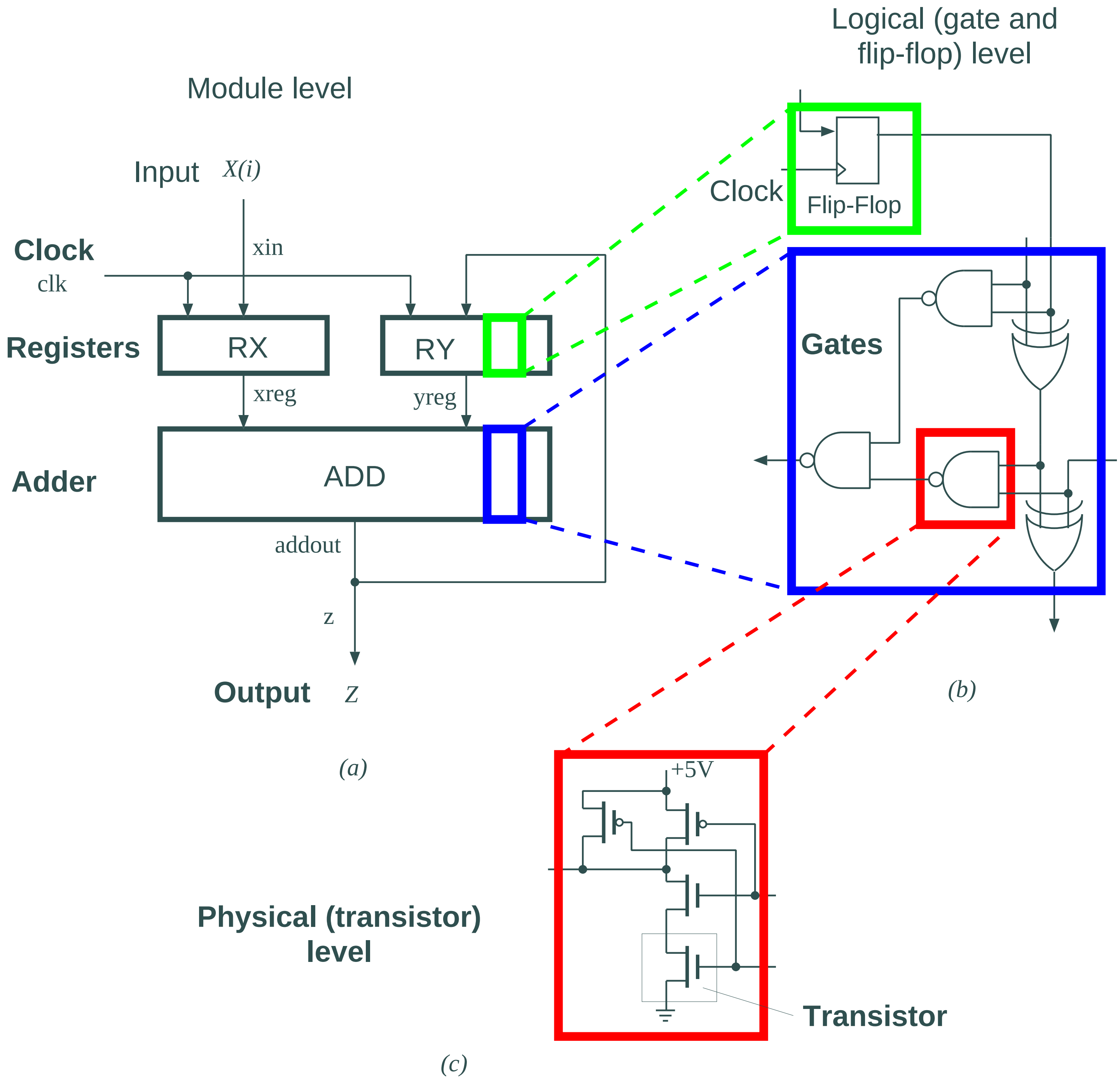


Figure 1.8: Digital system: a) module level; b) logical level; and c) physical level.

COMPUTER-AIDED DESIGN TOOLS

- Design of digital systems an involved and laborious process
- Various computer-aided design (CAD) tools available
- Main types of CAD tools support the main phases of digital design:

  (i) description (specification),

  (ii) design (synthesis) including various optimizations to reduce cost and improve performance, and

  (iii) checking of the design with respect to its specification.
- The design phases typically require several passes

CAD (cont.)

DESCRIPTION of digital systems for design purposes

- At a high-level, use hardware-description language (HDL)
- At the binary level, use HDLs to describe the system structure
- Editors used to produce HDL programs
- Graphical forms - *logic diagrams* also used for structure

μVHDL description

```
USE WORK.ALL;
ENTITY sample_system IS
  PORT (xin: IN  BIT_VECTOR;
        z  : OUT BIT_VECTOR;
        clk: IN  BIT      );
END sample_system;

ARCHITECTURE structural OF sample_system IS
  SIGNAL xreg, yreg, addout: BIT_VECTOR(7 DOWNTO 0);

BEGIN
  RX: ENTITY BitReg8 PORT MAP(xin,xreg,clk);
  RY: ENTITY BitReg8 PORT MAP(addout,yreg,clk);
 ADD: ENTITY Adder   PORT MAP(xreg,yreg,addout);
      z <= addout;
END structural;
```

Figure 1.9: μVHDL-based description of a system.

CAD (cont.)

SYNTHESIS AND OPTIMIZATION

– Semi-automated

SIMULATION tools generate behavior of a system for given input

– *Logic simulation*

– *Timing simulation*