Akdeniz University
Computer Engineering Department

CSE206 Computer Organization

Week11: Digital Logic

Assoc.Prof.Dr. Taner Danışman

tdanisman@akdeniz.edu.tr

Course program (Textbook: Stalling 10th Edt.)

Week 1	19-Feb-24	Introduction	Ch1
Week 2	26-Feb-24	Computer Evolution	Ch2
Week 3	4-Mar-24	Computer Systems	Ch3
Week 4	11-Mar-24	Cache Memory, Direct Cache Mapping	Ch4
Week 5	18-Mar-24	Associative and Set Associative Mapping	Ch4
Week 6	25-Mar-24	Internal Memory, External Memory, I/O	Ch5-Ch6-Ch7
Week 7	1-Apr-24	Number Systems, Computer Arithmetic	Ch9-Ch10
Week 8	8-Apr-24	Holiday	
Week 9	15-Apr-24	Number Systems, Computer Arithmetic	Ch11
Week 10	26-Apr-24	Midterm	Ch1-Ch11
Week 11	29-Apr-24	Digital Logic	Ch11
Week 12	6-May-24	Instruction Sets	Ch12
Week 13	13-May-24	Addressing Modes	Ch13
Week 14	20-May-24	Processor Structure and Function	Ch14
Week 15	27-May-24	Assembly Language (TextBook : Assembly Language for x86 Processors)	Kip Irvine

Boolean Algebra

- Mathematical discipline used to design and analyze the behavior of the digital circuitry in digital computers and other digital systems

- Named after **George Boole**
  - English mathematician
  - Proposed basic principles of the algebra in 1854

- Claude Shannon suggested Boolean algebra could be used to solve problems in relay-switching circuit design

- Is a convenient tool:
  - Analysis
    - It is an economical way of describing the function of digital circuitry
  - Design
    - Given a desired function, Boolean algebra can be applied to develop a simplified implementation of that function

Boolean Variables and Operations

- Makes use of variables and operations
  - Are logical
  - A variable may take on the value 1 (TRUE) or 0 (FALSE)
  - Basic logical operations are AND, OR, and NOT
- AND
  - Yields true (binary value 1) if and only if both of its operands are true
  - In the absence of parentheses the AND operation takes precedence over the OR operation
  - When no ambiguity will occur the AND operation is represented by simple concatenation instead of the dot operator
- OR
  - Yields true if either or both of its operands are true
- NOT
  - Inverts the value of its operand

Table 11.1 Boolean Operators

P	Q	NOT P $(\overline{P})$	P AND Q $(P \cdot Q)$	P OR Q $(P + Q)$	P NAND Q $(\overline{P \cdot Q})$	P NOR Q $(\overline{P + Q})$	P XOR Q $(P \oplus Q)$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

(a) Boolean Operators of Two Input Variables

Operation	Expression	Output = 1 if
AND	$A \cdot B \cdot \ldots$	All of the set $\{A, B, \ldots\}$ are 1.
OR	$A + B + \ldots$	Any of the set $\{A, B, \ldots\}$ are 1.
NAND	$\overline{A \cdot B \cdot \ldots}$	Any of the set $\{A, B, \ldots\}$ are 0.
NOR	$\overline{A + B + \ldots}$	All of the set $\{A, B, \ldots\}$ are 0.
XOR	$A \oplus B \oplus \ldots$	The set $\{A, B, \ldots\}$ contains an odd number of ones.

(b) Boolean Operators Extended to More than Two Inputs (A, B, . . .)

Table 11.2 Basic Identities of Boolean Algebra

Basic Postulates

$A \cdot B = B \cdot A$	$A + B = B + A$	Commutative Laws
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$	Distributive Laws
$1 \cdot A = A$	$0 + A = A$	Identity Elements
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$	Inverse Elements

Other Identities

$0 \cdot A = 0$	$1 + A = 1$	
$A \cdot A = A$	$A + A = A$	
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	Associative Laws
$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$	DeMorgan's Theorem

Table 11.2   Basic Identities of Boolean Algebra

Basic Logic Gates

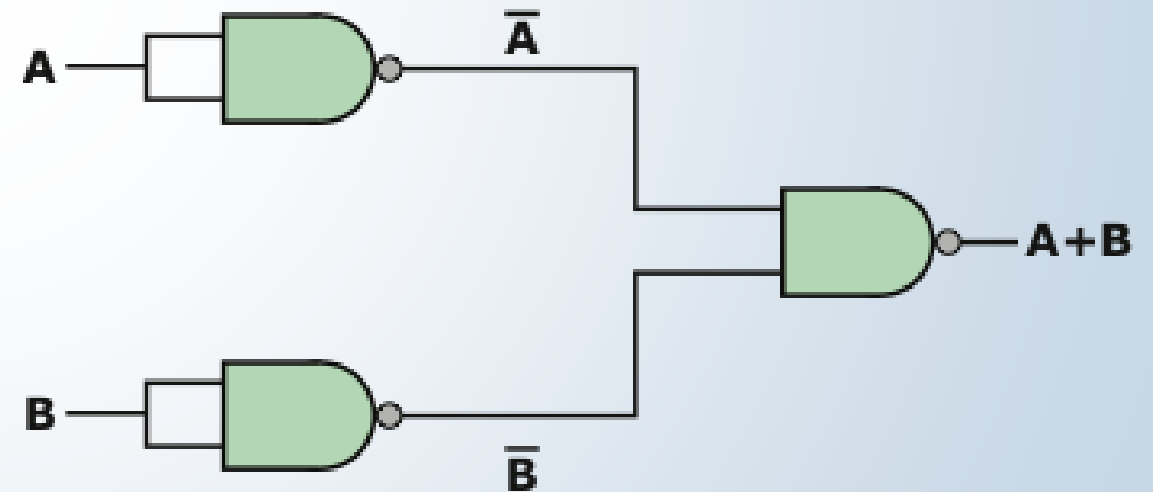| Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

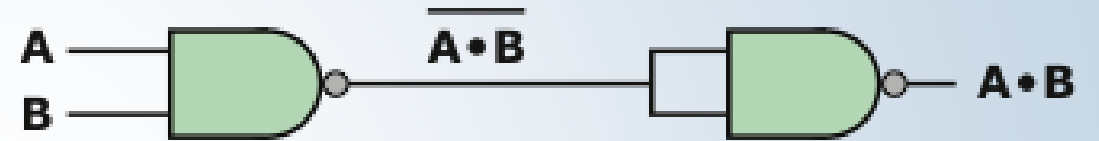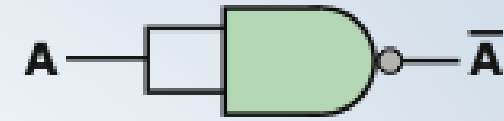# Uses of NAND Gates

Figure 11.2 Some Uses of NAND Gates
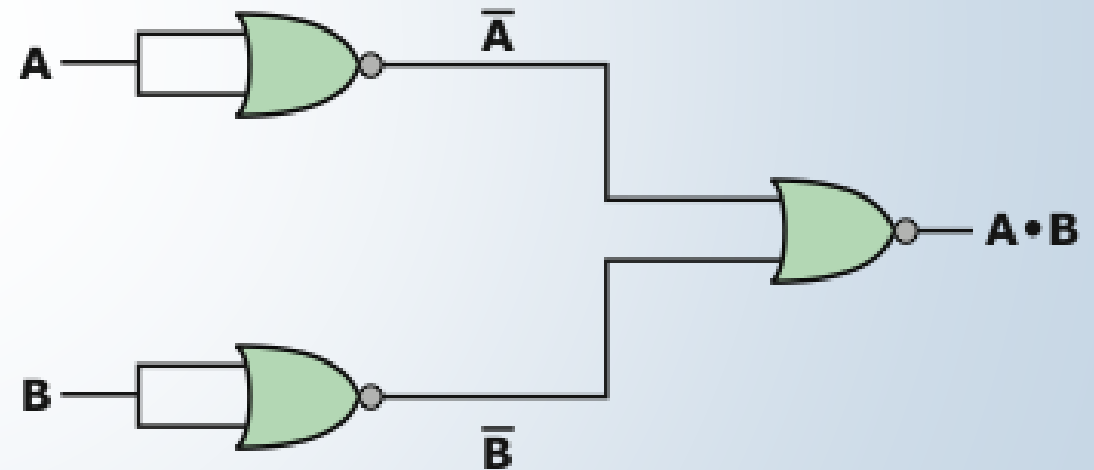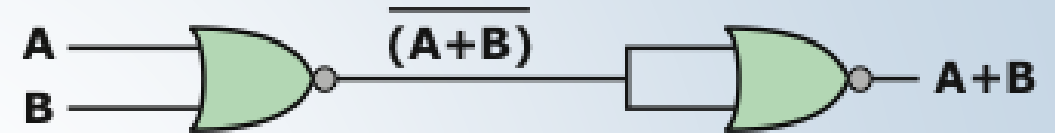
Uses of NOR Gates

Figure 11.3   Some Uses of NOR Gates

Combinational Circuit

- An interconnected set of gates whose output at any time is a function only of the input at that time

- The appearance of the input is followed almost immediately by the appearance of the output, with only gate delays

- Consists of *n* binary inputs and *m* binary outputs

- Can be defined in three ways:

  - Truth table

    - For each of the 2^n possible combinations of input signals, the binary value of each of the *m* output signals is listed

  - Graphical symbols

    - The interconnected layout of gates is depicted

  - Boolean equations

    - Each output signal is expressed as a Boolean function of its input signals

Boolean Function of Three Variables

There are three combinations of input values that cause F to be 1, and if any one of these combinations occurs, the result is 1. This form of expression, for self- evident reasons, is known as the **sum of products (SOP)** form.

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Table 11.3 A Boolean Function of Three Variables
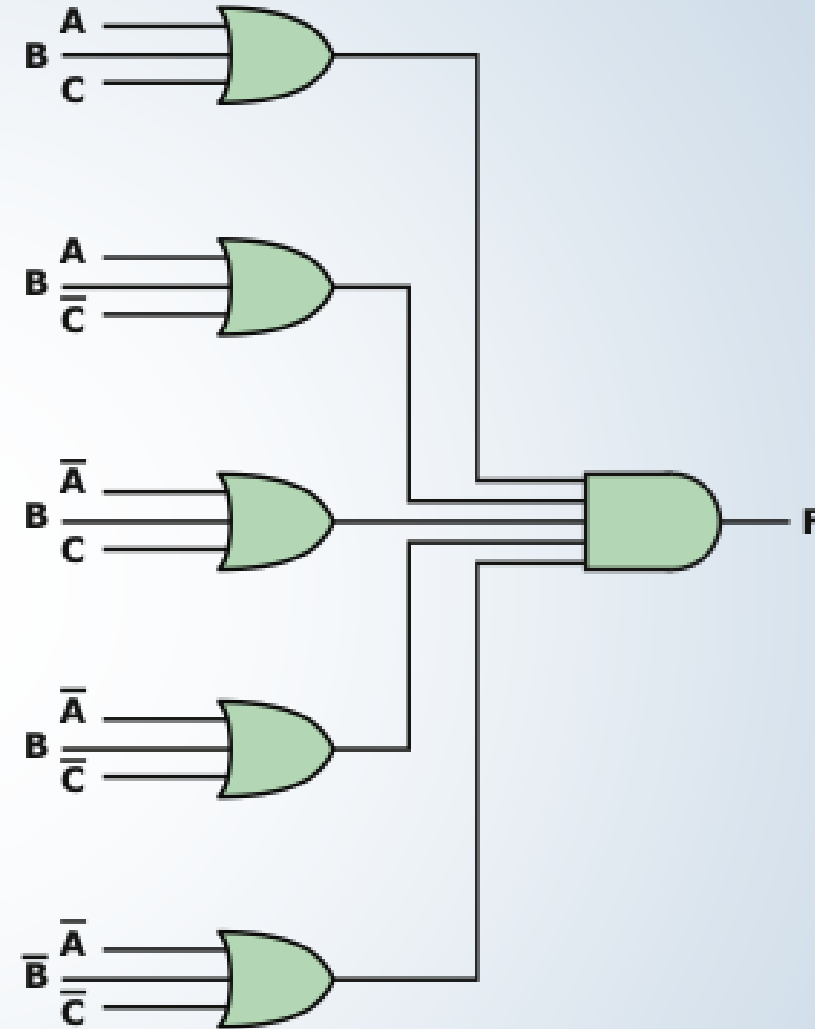
Product-of-Sums Implementation of Table 11.3

Figure 11.5 Product-of-Sums Implementation of Table 11.3
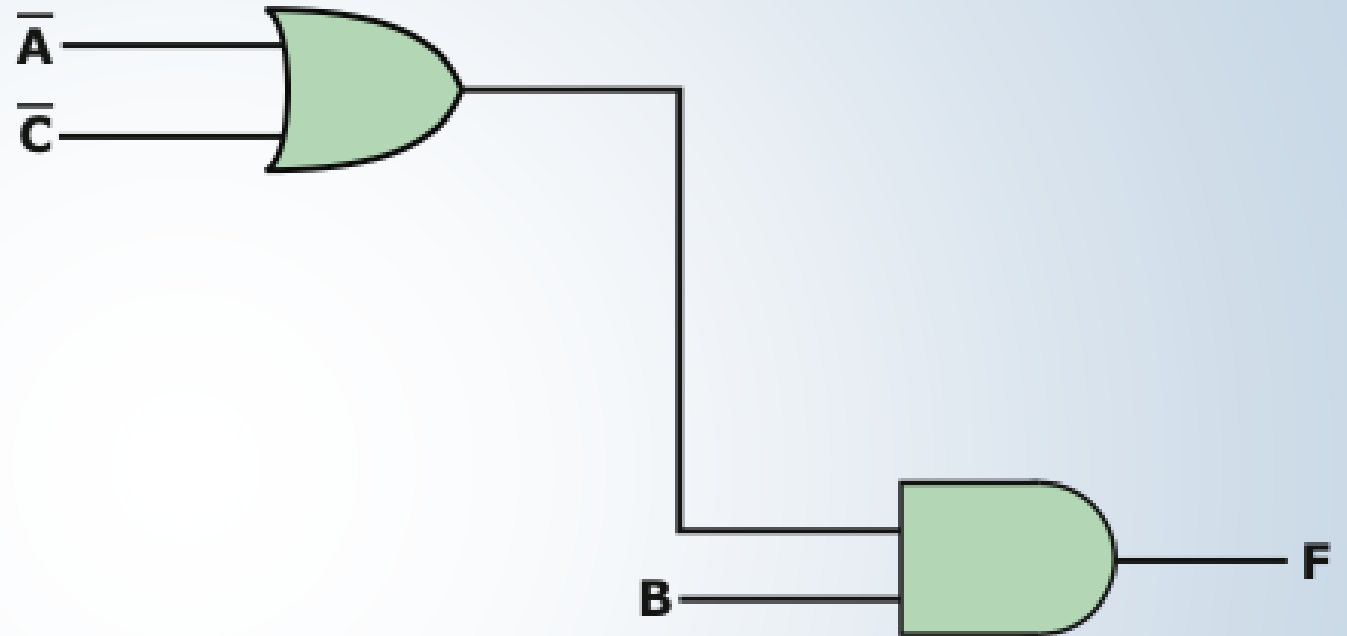
Algebraic Simplification

Figure 11.6   Simplified Implementation of Table 11.3

- Involves the application of the identities of Table 11.2 to reduce the Boolean expression to one with fewer elements

Karnaugh Maps

■ A convenient way of representing a Boolean function of a small number (up to four) of variables
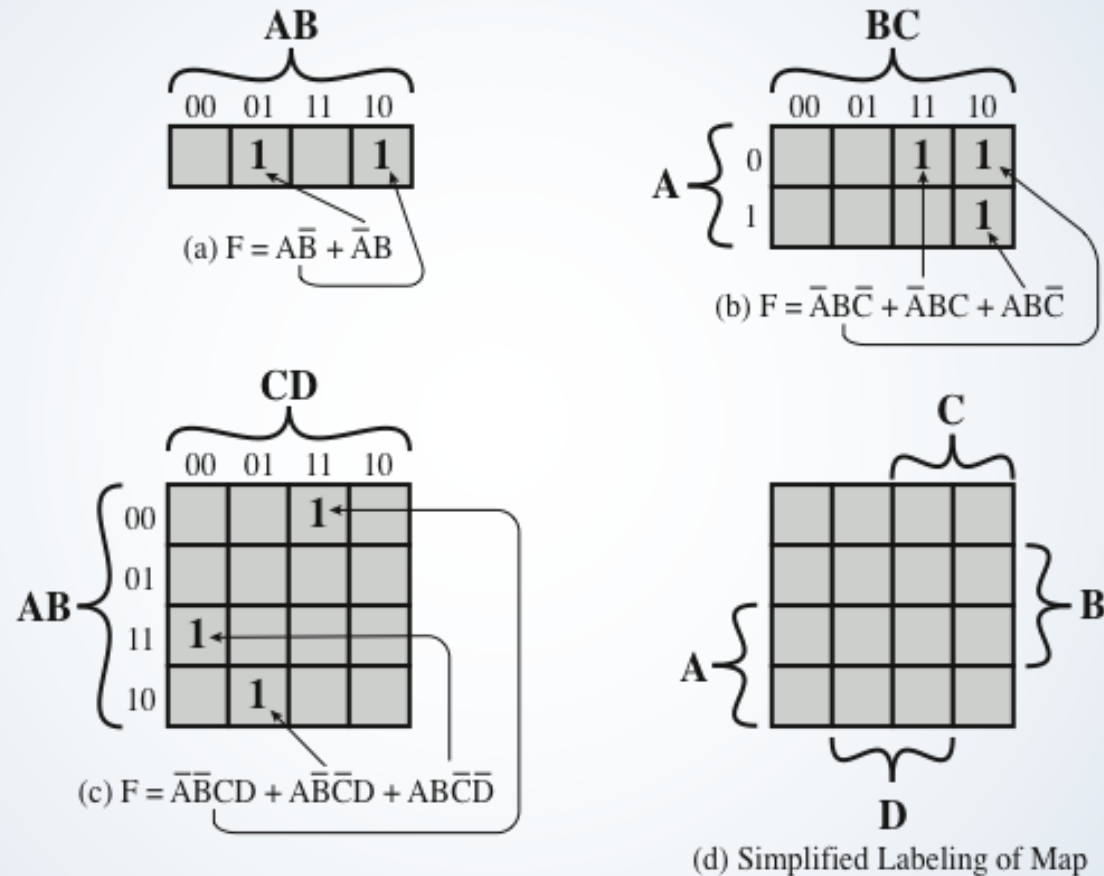

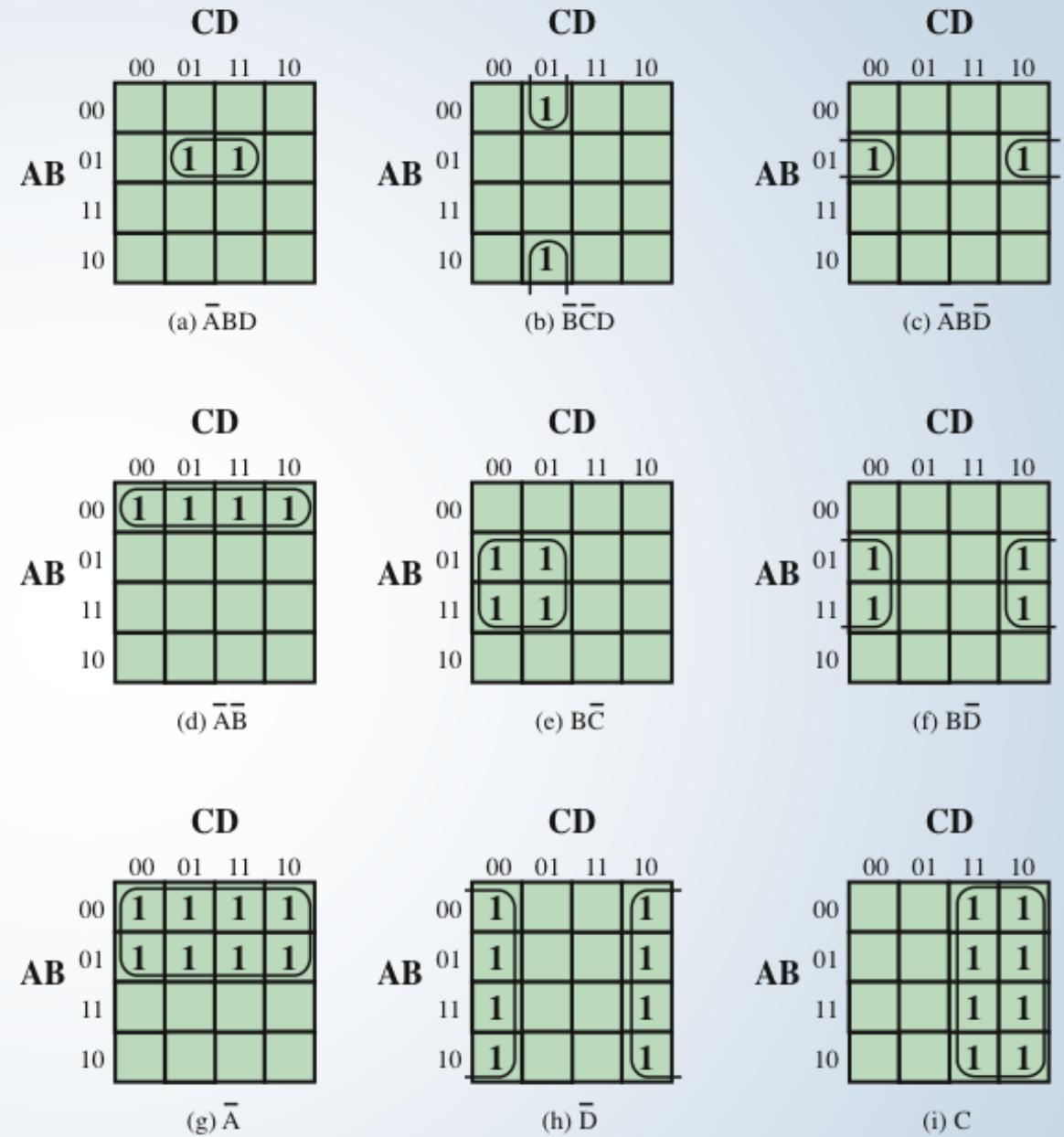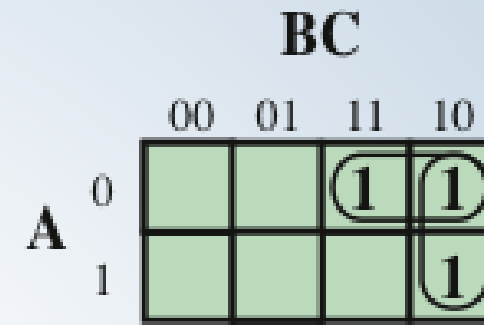Figure 11.7 The Use of Karnaugh Maps to Represent Boolean Functions

Karnaugh Maps

Figure 11.8 Example Use of Karnaugh Maps

Karnaugh Maps
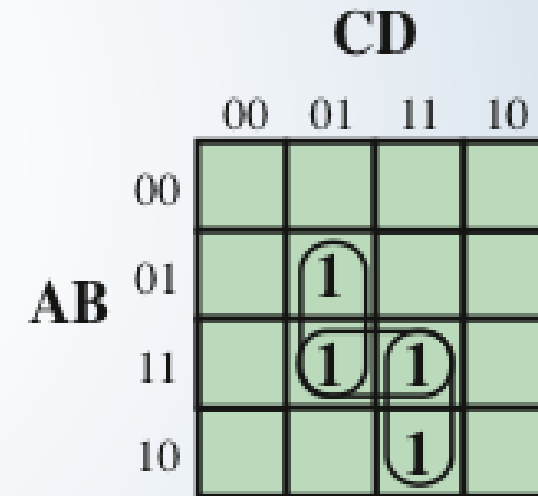
Overlapping

BC

	00	01	11	10
A 0			1	1
1				1

(a) $F = \overline{A}B + B\overline{C}$

Groups

CD

	00	01	11	10
AB 00				
01		1		
11	1	1		
10		1		

(b) $F = B\overline{C}D + ACD$

Figure 11.9 Overlapping Groups

Truth Table for the One-Digit Packed Decimal Incrementer

	Input					Output			
Number	A	B	C	D	Number	W	X	Y	Z
0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	2	0	0	1	0
2	0	0	1	0	3	0	0	1	1
3	0	0	1	1	4	0	1	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	6	0	1	1	0
6	0	1	1	0	7	0	1	1	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	9	1	0	0	1
9	1	0	0	1	0	0	0	0	0
Don't care condition	1	0	1	0		d	d	d	d
	0	1	1	1		d	d	d	d
	1	1	0	0		d	d	d	d
	1	1	0	1		d	d	d	d
	1	1	1	0		d	d	d	d
	1	1	1	1		d	d	d	d

Table 11.4   Truth Table for the One-Digit Packed Decimal Incrementer

The Incrementer

CD

00 01 11 10

AB 01

(a) $W = A\overline{D} + \overline{A}BCD$

CD

00 01 11 10

AB 01

(b) $X = B\overline{D} + B\overline{C} + BCD$

CD

00 01 11 10

AB 01

(c) $Y = \overline{A}\,\overline{C}D + \overline{A}C\overline{D}$

CD

00 01 11 10

AB 01
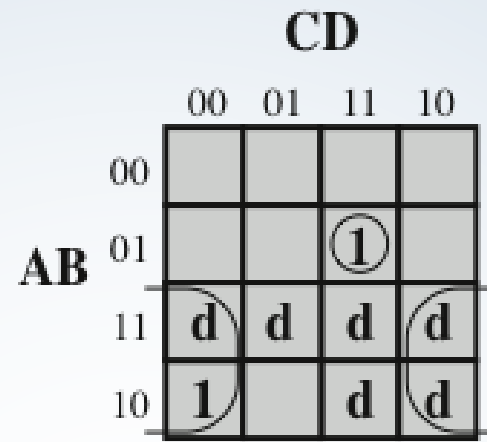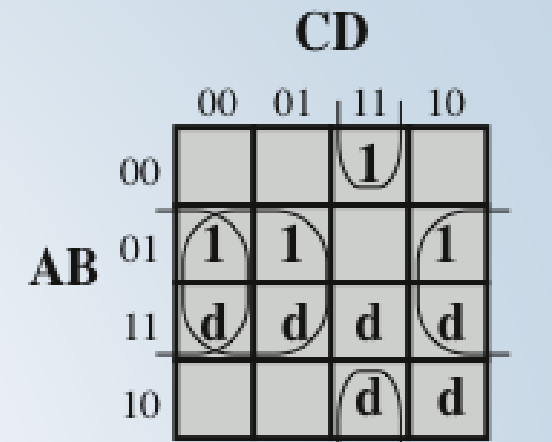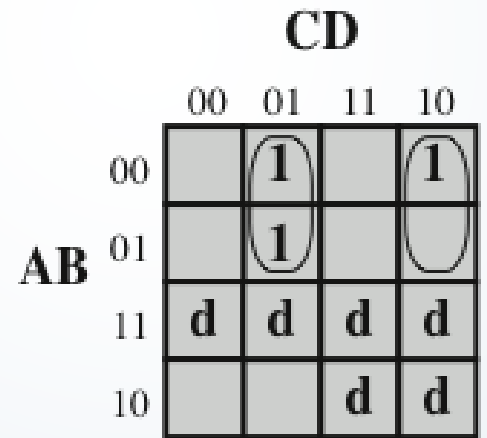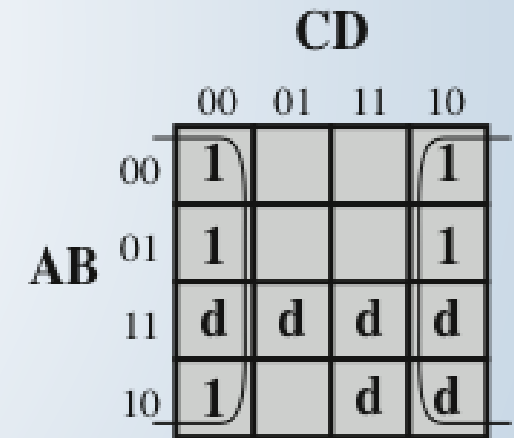
(d) $Z = \overline{D}$

Figure 11.10   Karnaugh Maps for the Incrementer

NAND and NOR Implementations

➡ It is sometimes desirable to implement a Boolean function solely with NAND gates or solely with NOR gates.

➡ Although this may not be the minimum-gate implementation, it has the advantage of **regularity**, which can simplify the manufacturing process.
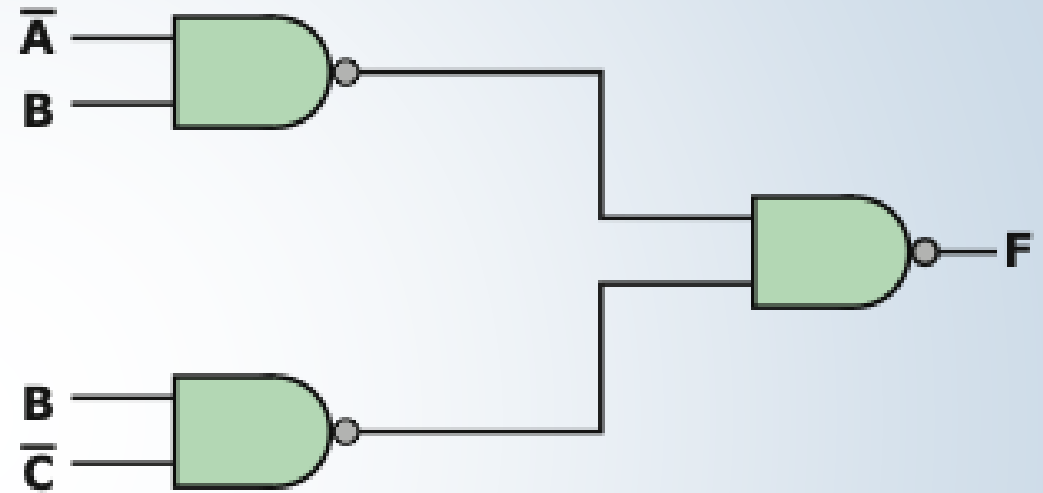
Figure 11.11 NAND Implementation of Table 11.3

Multiplexers

- Connect multiple inputs to a single output

- At any time, one of the inputs is selected to be passed to the output.

- To select one of the four possible inputs, a 2-bit selection code is needed, and this is implemented as two select lines labeled S1 and S2.
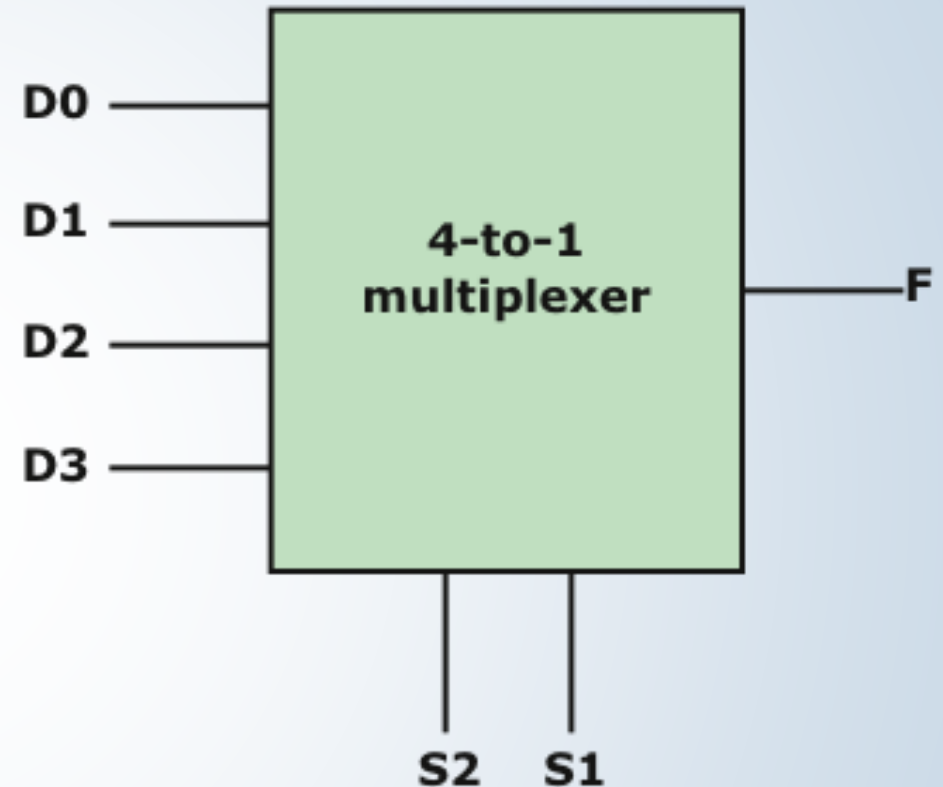
Figure 11.12 4-to-1 Multiplexer Representation

4-to-1
Multiplexer Truth Table

S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Table 11.7 4-to-1 Multiplexer Truth Table

Decoders

- Combinational circuits with a number of output lines, only one of which is asserted at any time
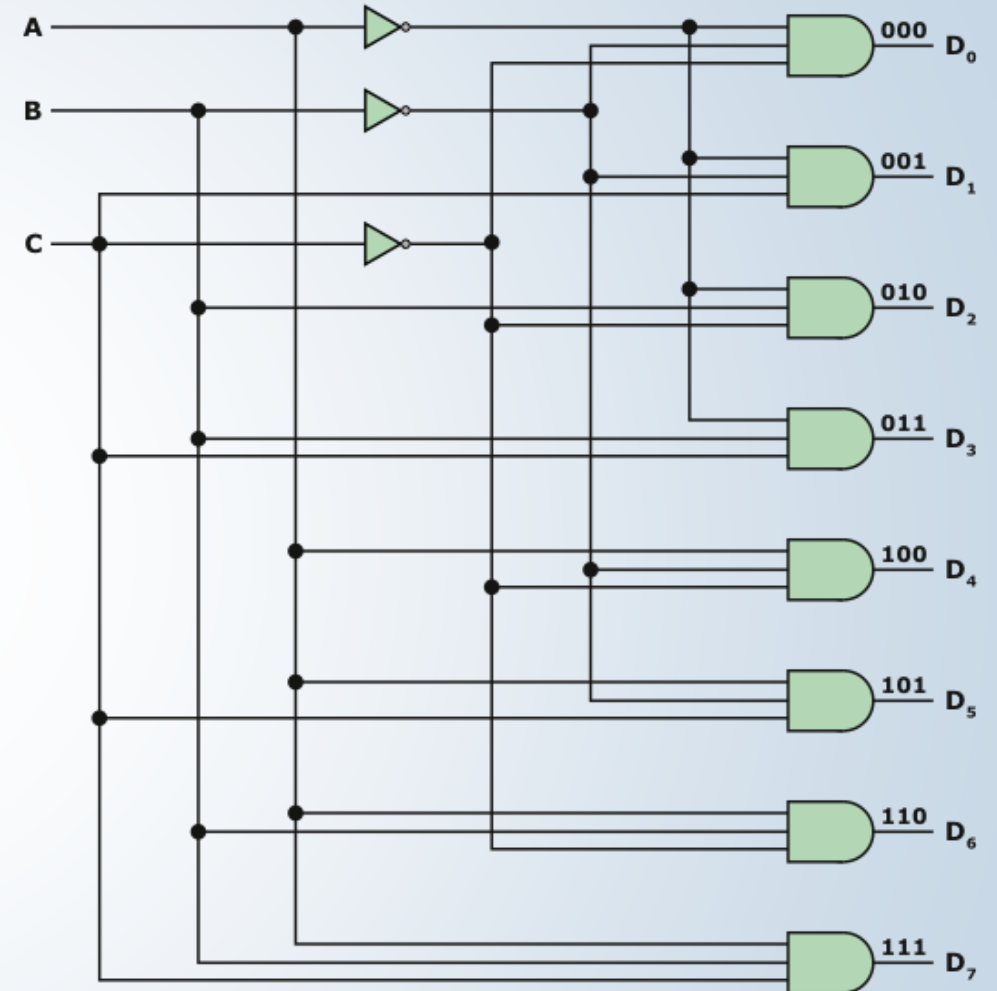
Figure 11.15 Decoder with 3 Inputs and $2^3 = 8$ Outputs

Address Decoding

Address	Chip
0000-00FF	0
0100-01FF	1
0200-02FF	2
0300-03FF	3
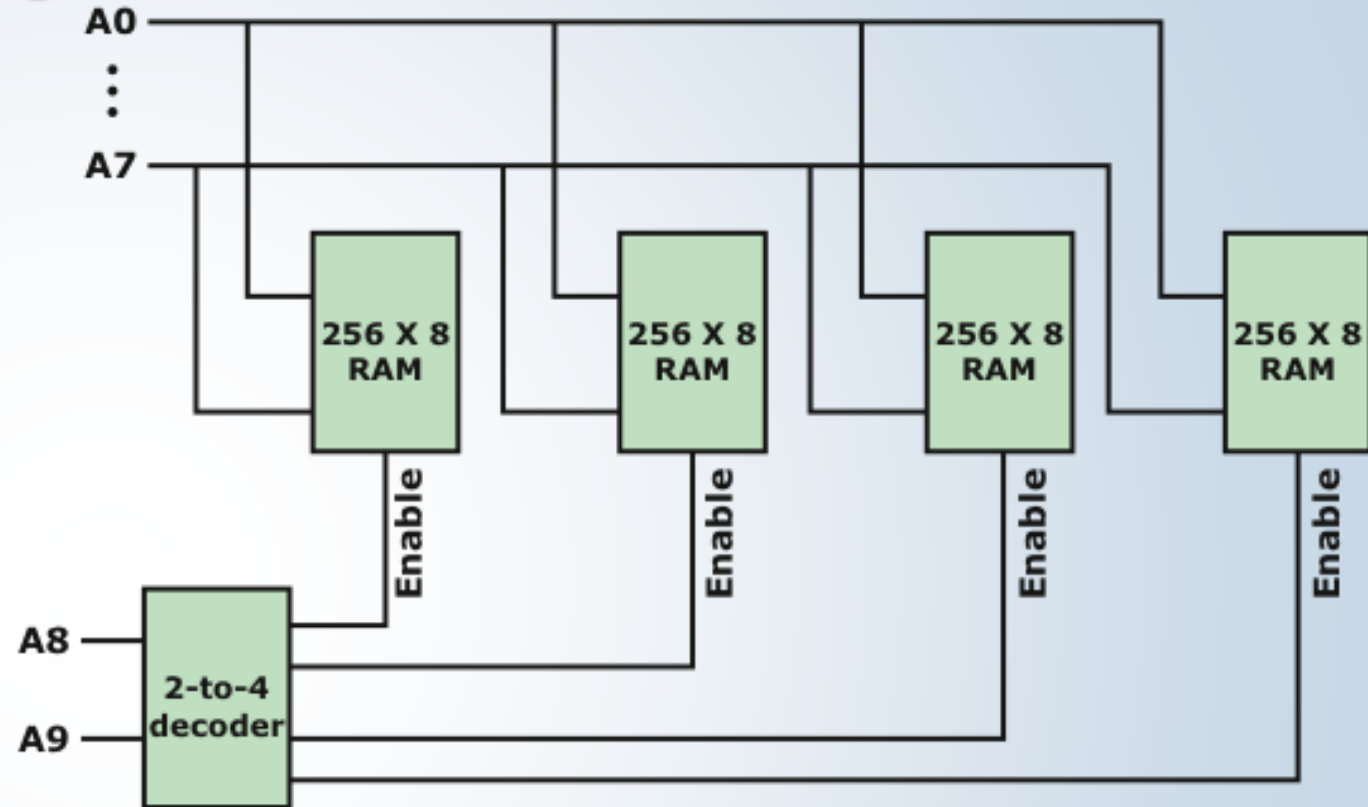
1K-byte memory using four 256 * 8-bit RAM chips

Figure 11.16 Address Decoding

Read-Only Memory (ROM)

- Memory that is implemented with combinational circuits
  - Combinational circuits are often referred to as "memoryless" circuits because their output depends only on their current input and no history of prior inputs is retained
- Memory unit that performs only the read operation
  - Binary information stored in a ROM is permanent and is created during the fabrication process
  - A given input to the ROM (address lines) always produces the same output (data lines)
  - Because the outputs are a function only of the present inputs, ROM is a **combinational circuit**

Binary Addition Truth Tables

(a) Single-Bit Addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(b) Addition with Carry Input

C_{in}	A	B	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 11.9 Binary Addition Truth Tables

4-Bit Adder
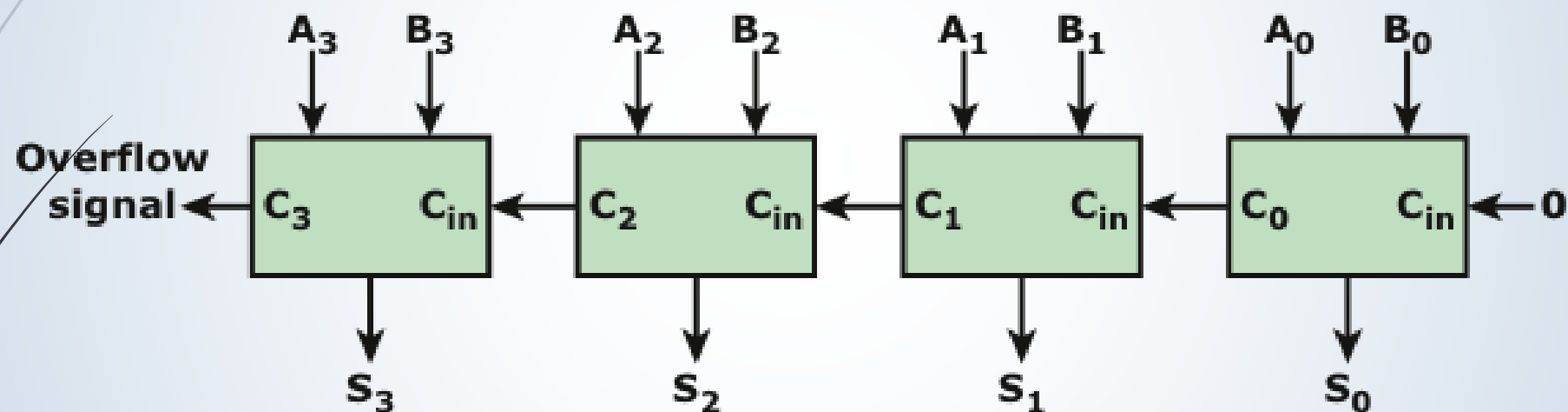
Figure 11.19 4-Bit Adder

Implementation of an Adder
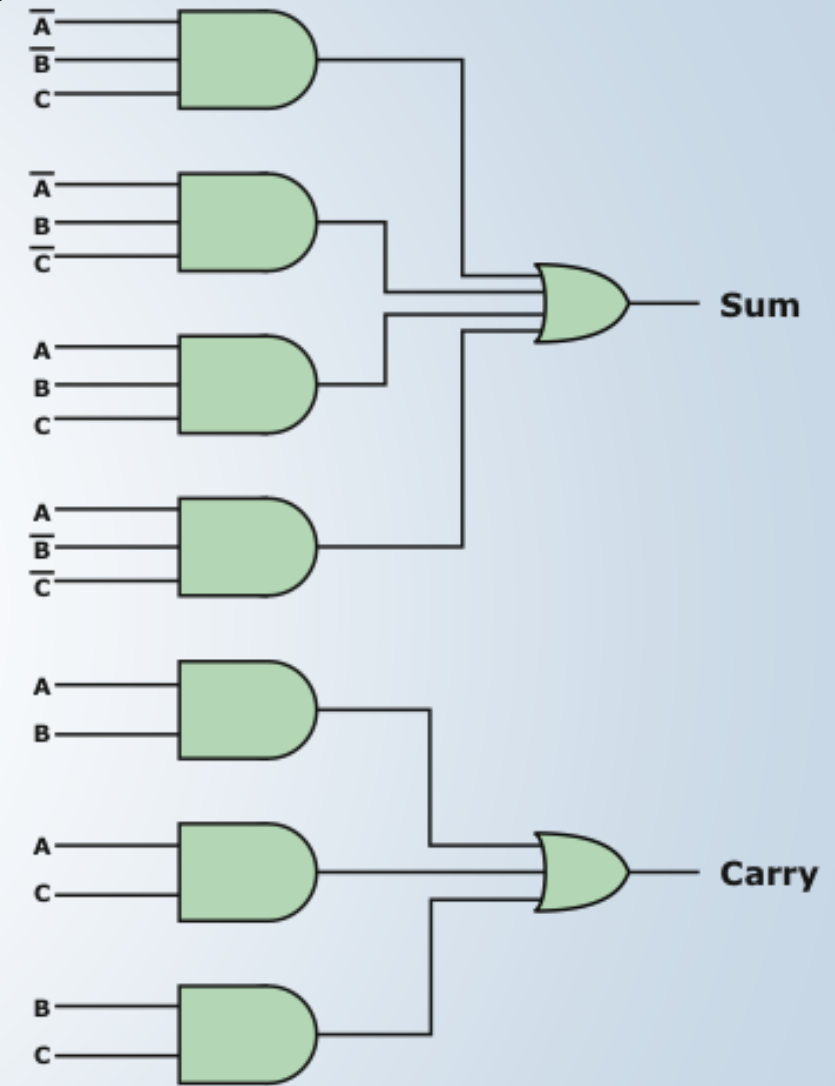
Figure 11.20 Implementation of an Adder

Construction of a 32-Bit Adder Using 8-Bit Adders
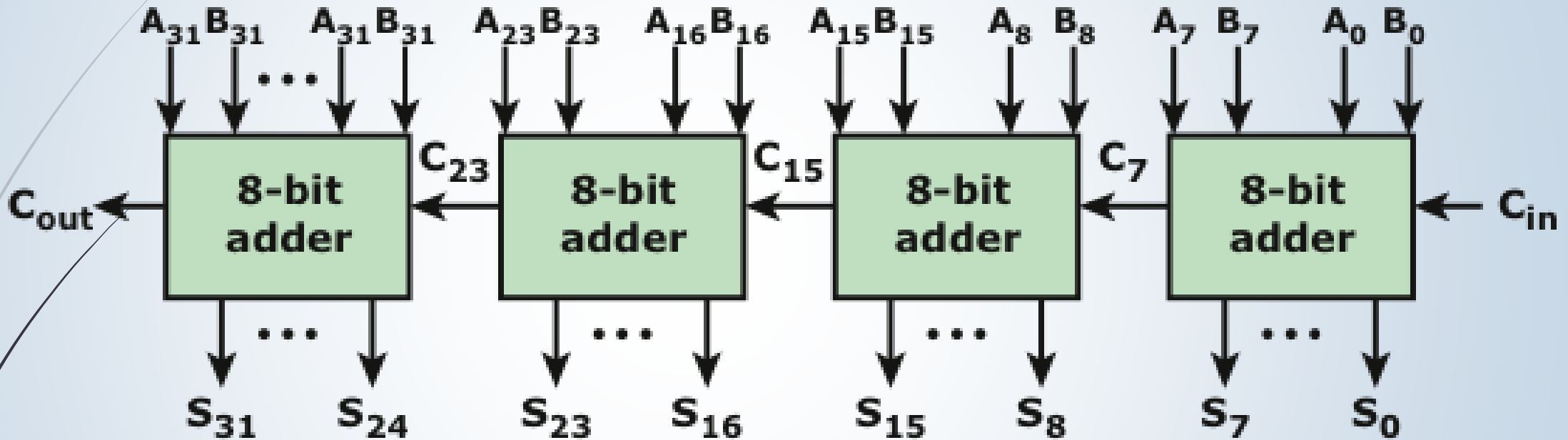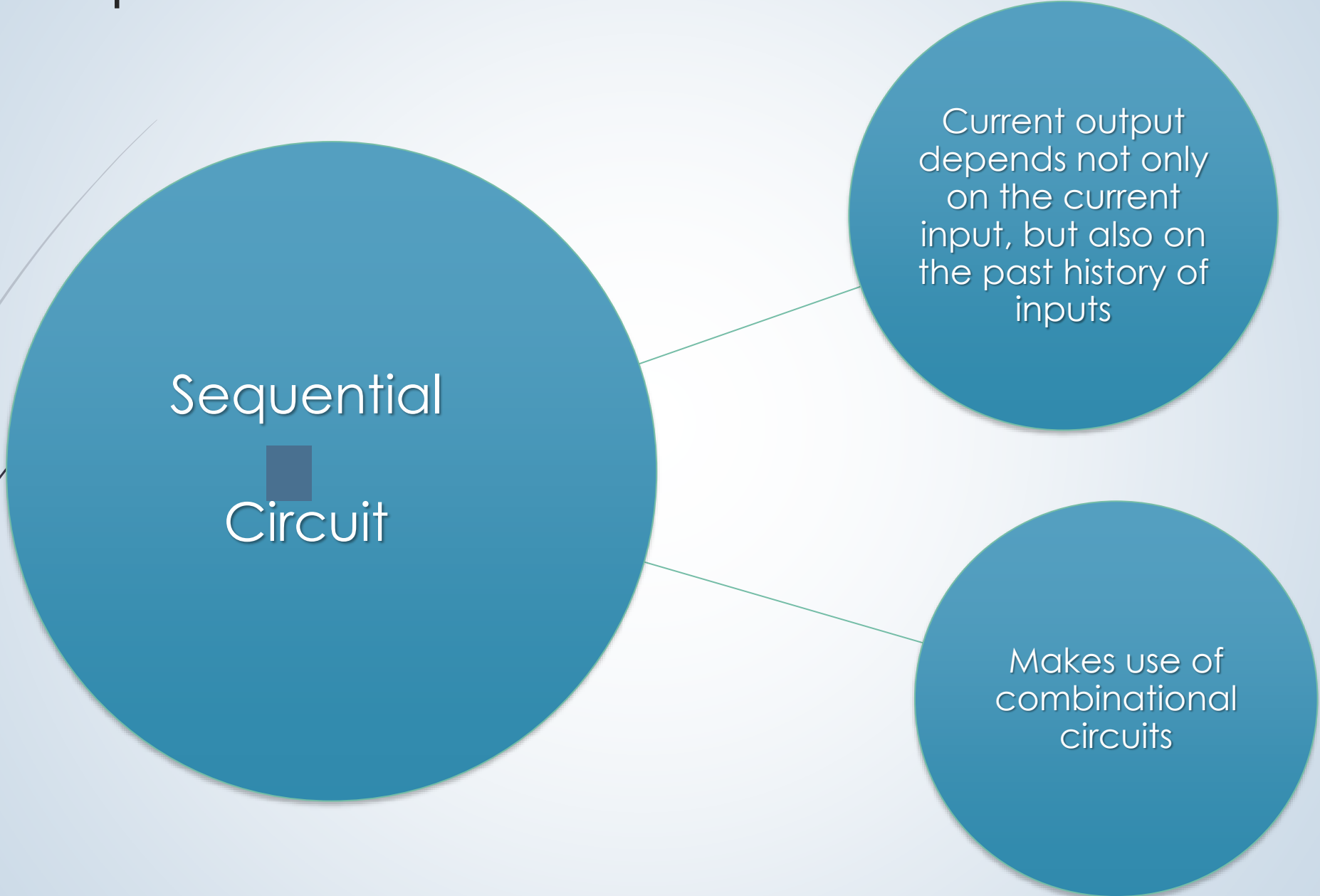
Figure 11.21 Construction of a 32-Bit Adder Using 8-Bit Adders

Sequential Circuit

Sequential Circuit

Current output depends not only on the current input, but also on the past history of inputs

Makes use of combinational circuits

Flip-Flops

- Simplest form of sequential circuit

- There are a variety of flip-flops, all of which share two properties:

  1. The flip-flop is a bistable device. It exists in one of two states and, **in the absence of input, remains in that state**. Thus, the flip-flop can function as a 1-bit memory.

  2. The flip-flop has two outputs, which are always the complements of each other.
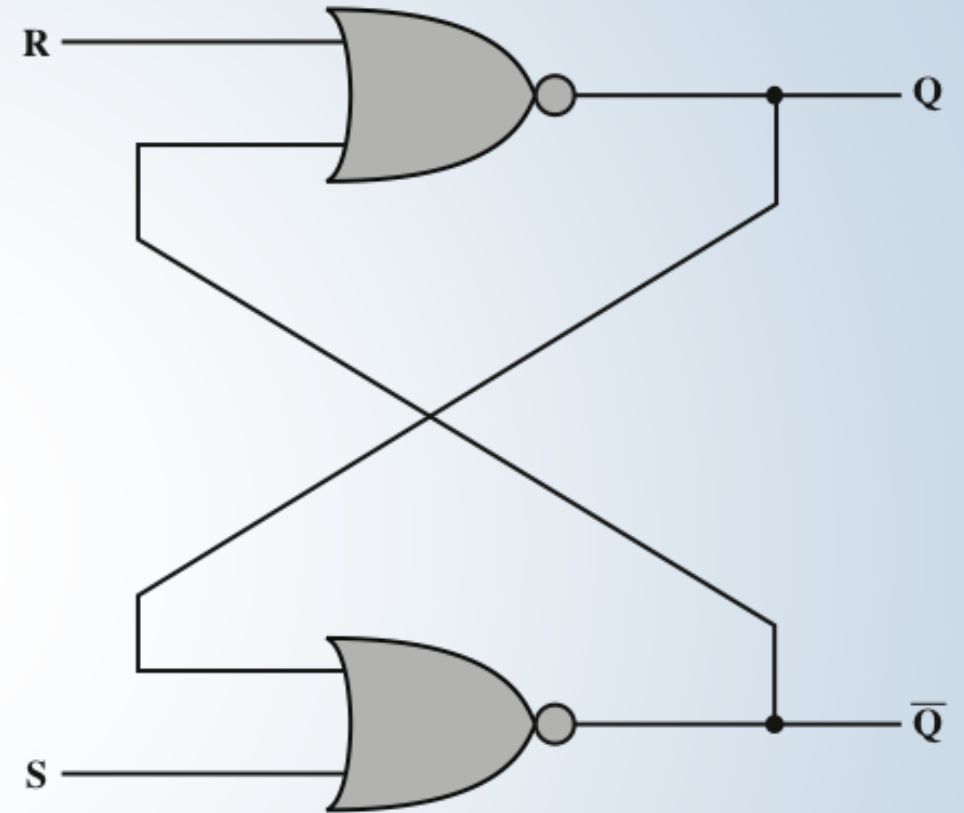
The S-R Latch

Figure 11.22 The S-R Latch Implemented with NOR Gates

NOR S-R Latch Timing Diagram

Figure 11.22   The S-R Latch Implemented with NOR Gates
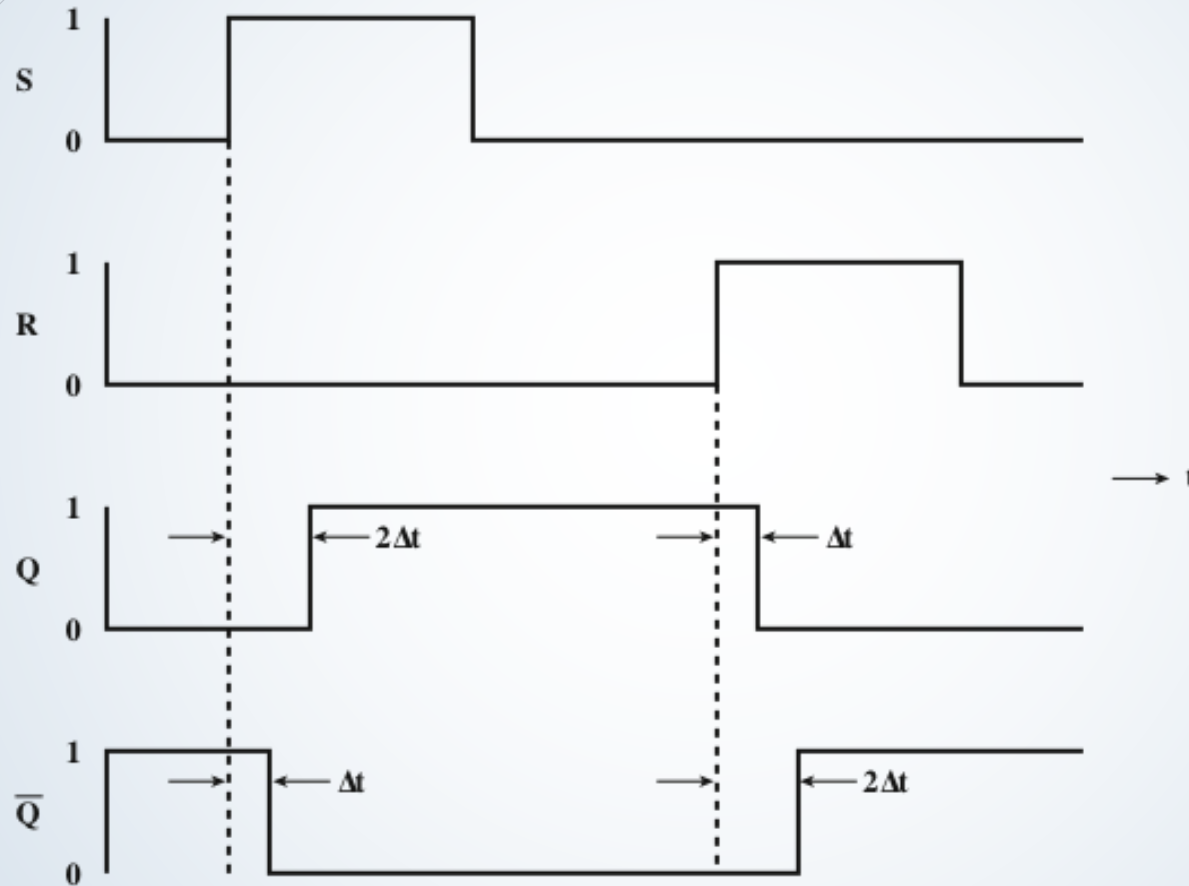
Figure 11.23   NOR S-R Latch Timing Diagram
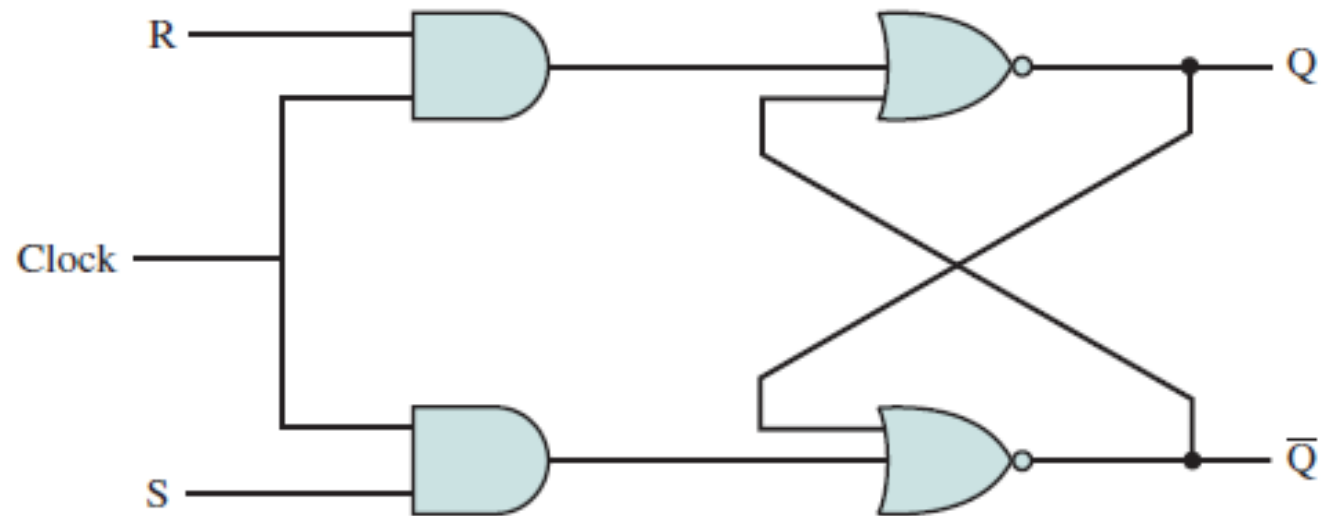
Figure 11.24   Clocked S–R Flip-Flop
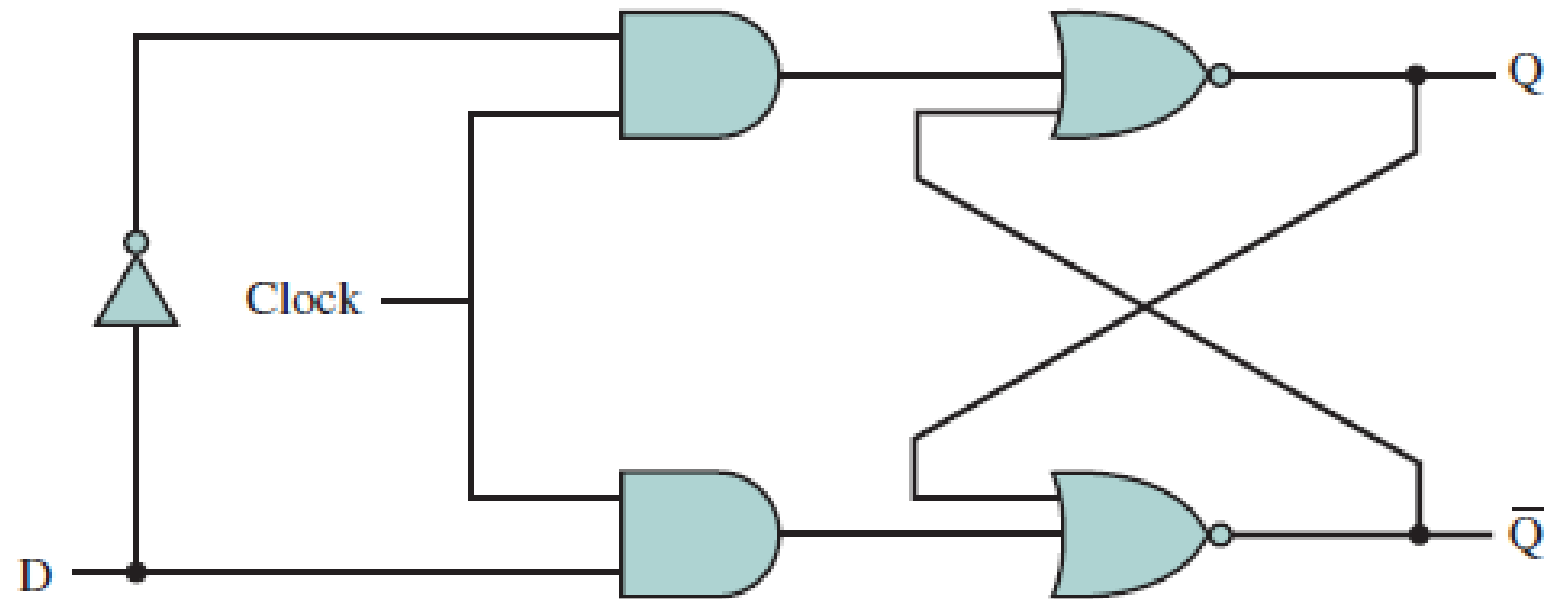
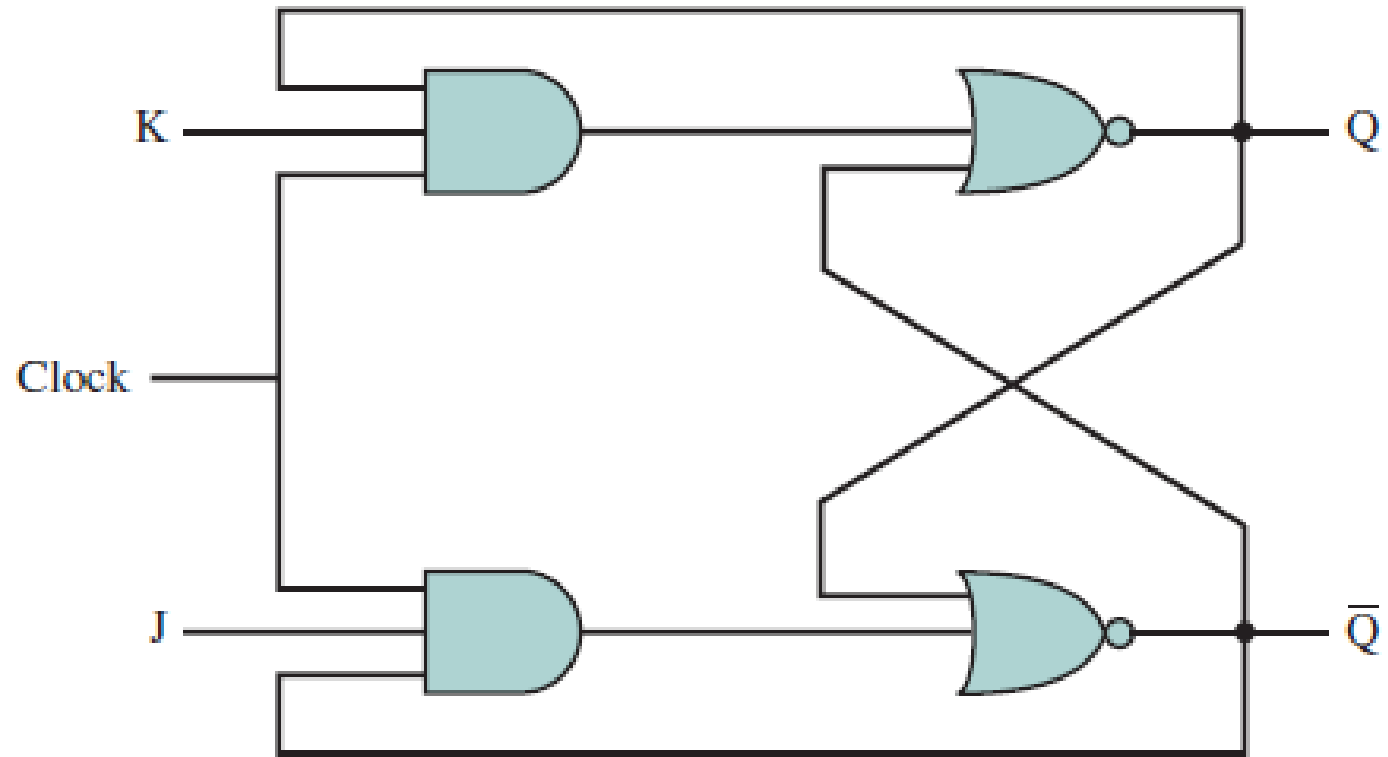**Figure 11.25   D Flip-Flop**

Figure 11.26   J–K Flip-Flop

Basic Flip Flops
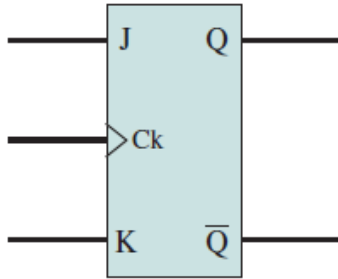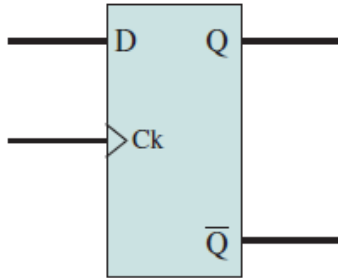
Name	Graphical Symbol	Truth Table		
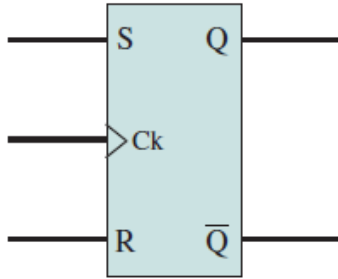
Name	Graphical Symbol	Truth Table

S–R

S — Q
Ck
R — $\overline{Q}$

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	–

J–K

J — Q
Ck
K — $\overline{Q}$

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

D

D — Q
Ck
$\overline{Q}$

D	Q_{n+1}
0	0
1	1

Figure 11.27 Basic Flip-Flops
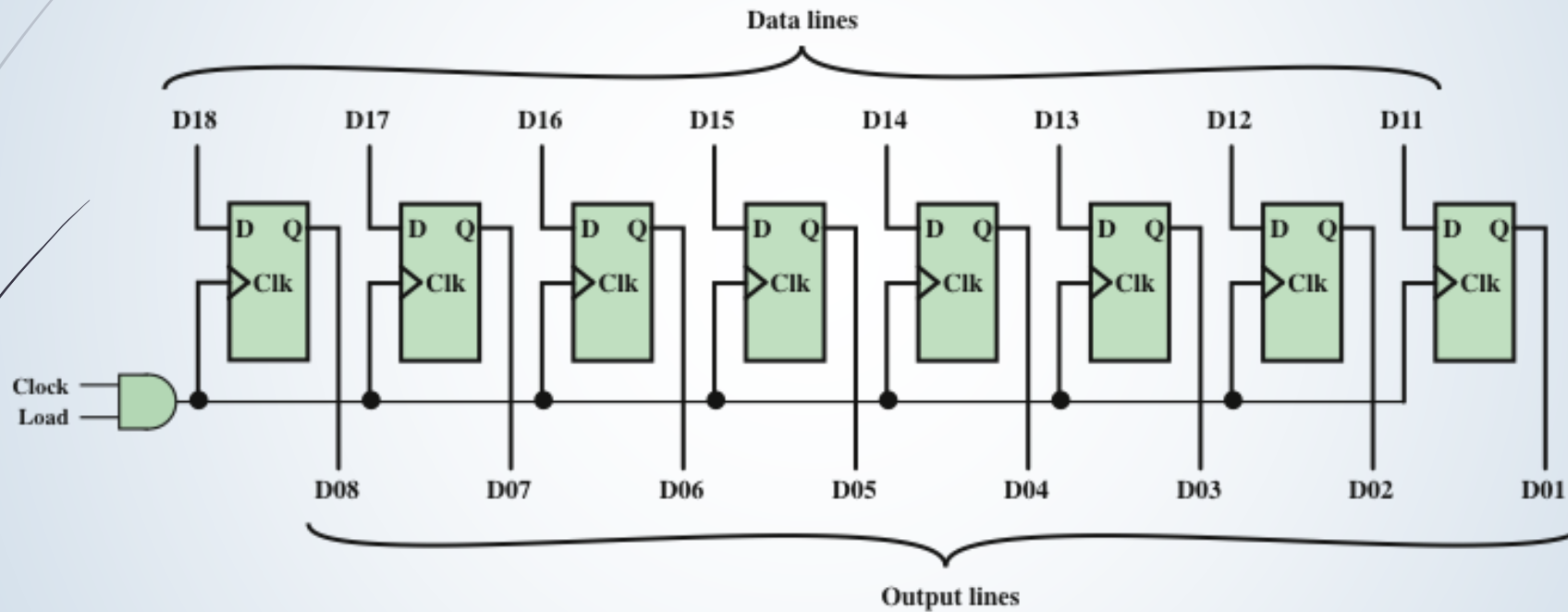
Parallel Register

Figure 11.28   8-Bit Parallel Register
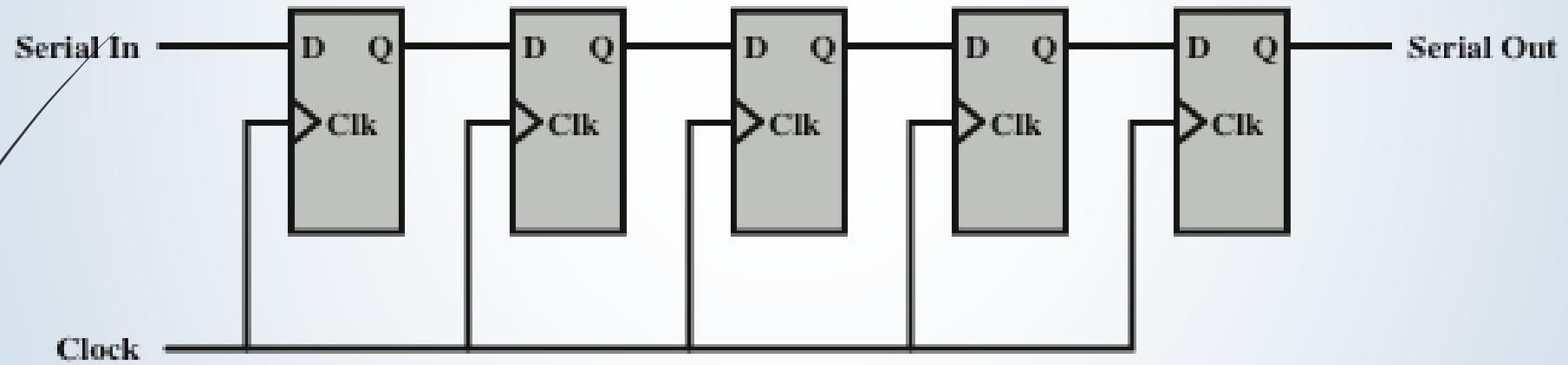
5-Bit Shift Register

Figure 11.29   5-Bit Shift Register

Counter

- A register whose value is easily incremented by 1 modulo the capacity of the register

- After the maximum value is achieved the next increment sets the counter value to 0

- An example of a counter in the CPU is the program counter

- Can be designated as:

  - Asynchronous

    - Relatively slow because the output of one flip-flop triggers a change in the status of the next flip-flop

  - Synchronous

    - All of the flip-flops change state at the same time

    - Because it is faster it is the kind used in CPUs
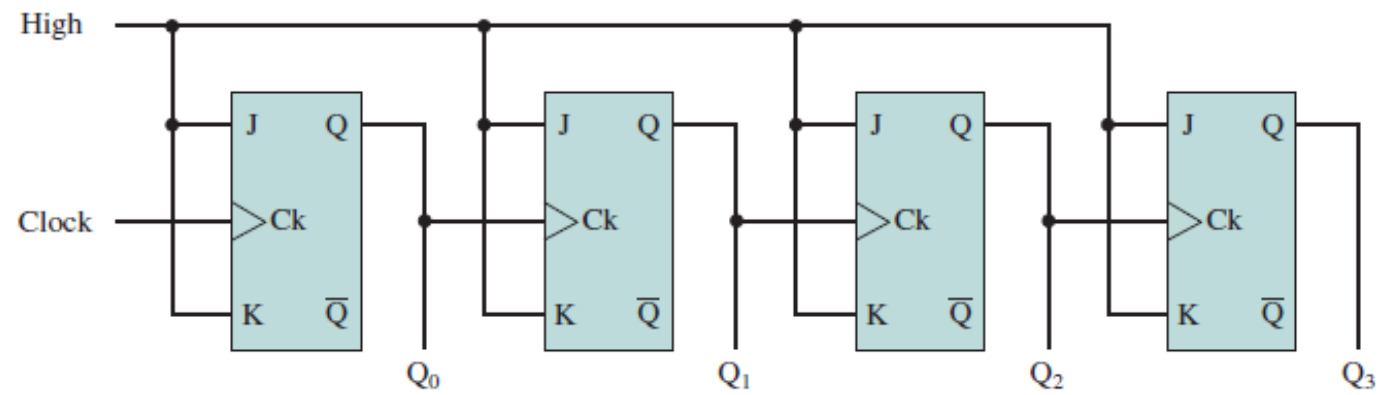
(a) Sequential circuit
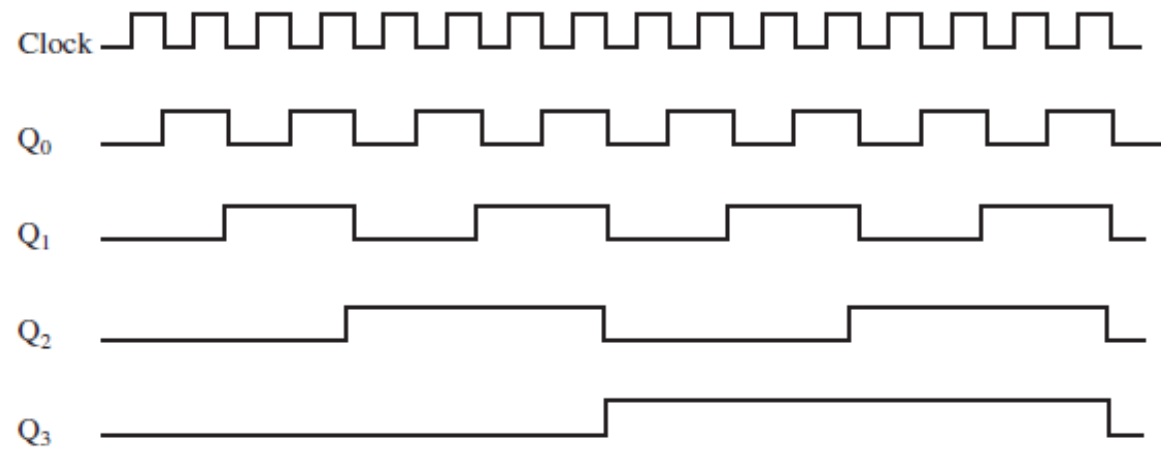
(b) Timing diagram

Figure 11.30 Ripple Counter