William Stallings Computer Organization and Architecture 10th Edition

Part One Overview

Chapter 1 organization, structure, architecture, function

Chapter 2 computer evolution and performance

Part Two Computer System

Chapter 3 A top level-view of computer function and interconnection

Chapter 4 Cache memory

Chapter 5 Internal memory

Chapter 6 External memory

Chapter 7 Input/output

Chapter 8 Operating system support

Part Three Arithmetic and Logic

Chapter 9 Number systems

Chapter 10 Computer arithmetic

Chapter 11 Digital logic

Part Four The Central Processing Unit

Chapter 12 Instruction sets: Characteristics and functions

Chapter 13 Instruction sets: Addressing modes and formats

Chapter 14 Processor structure and function

Chapter 15 Reduced instruction set computers

Chapter 16 Instruction-level parallelism and superscalars processors

Part Five Parallel Organization

Chapter 17 Parallel Processing

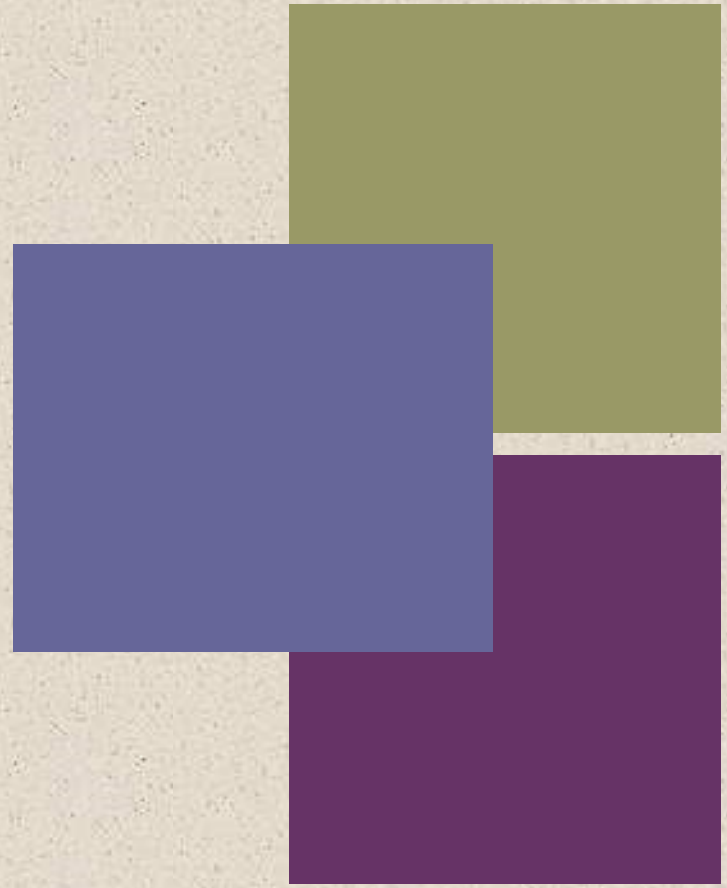
Chapter 18 Multicore computers

Chapter 19 Graphic Processing Unit

Part Six Control Unit

Chapter 20 Control unit operation

Chapter 21 Microprogrammed control



Chapter 11

Digital Logic

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

(a) Boolean Operators of Two Input Variables

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

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

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

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

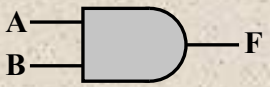
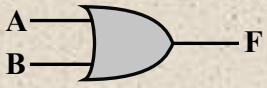
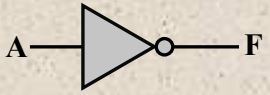
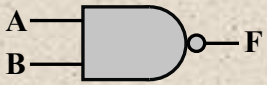
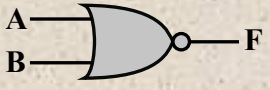
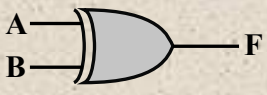
Name	Graphical Symbol	Algebraic Function	Truth Table
AND		F = A • B or F = AB	A B F 0 0 0 0 1 0 1 0 0 1 1 1
OR		F = A + B	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOT		F = $\overline{A}$ or F = A’	A F 0 1 1 0
NAND		F = $\overline{AB}$	A B F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		F = $\overline{A + B}$	A B F 0 0 1 0 1 0 1 0 0 1 1 0
XOR		F = A⊕B	A B F 0 0 0 0 1 1 1 0 1 1 1 0

Figure 11.1 Basic Logic Gates

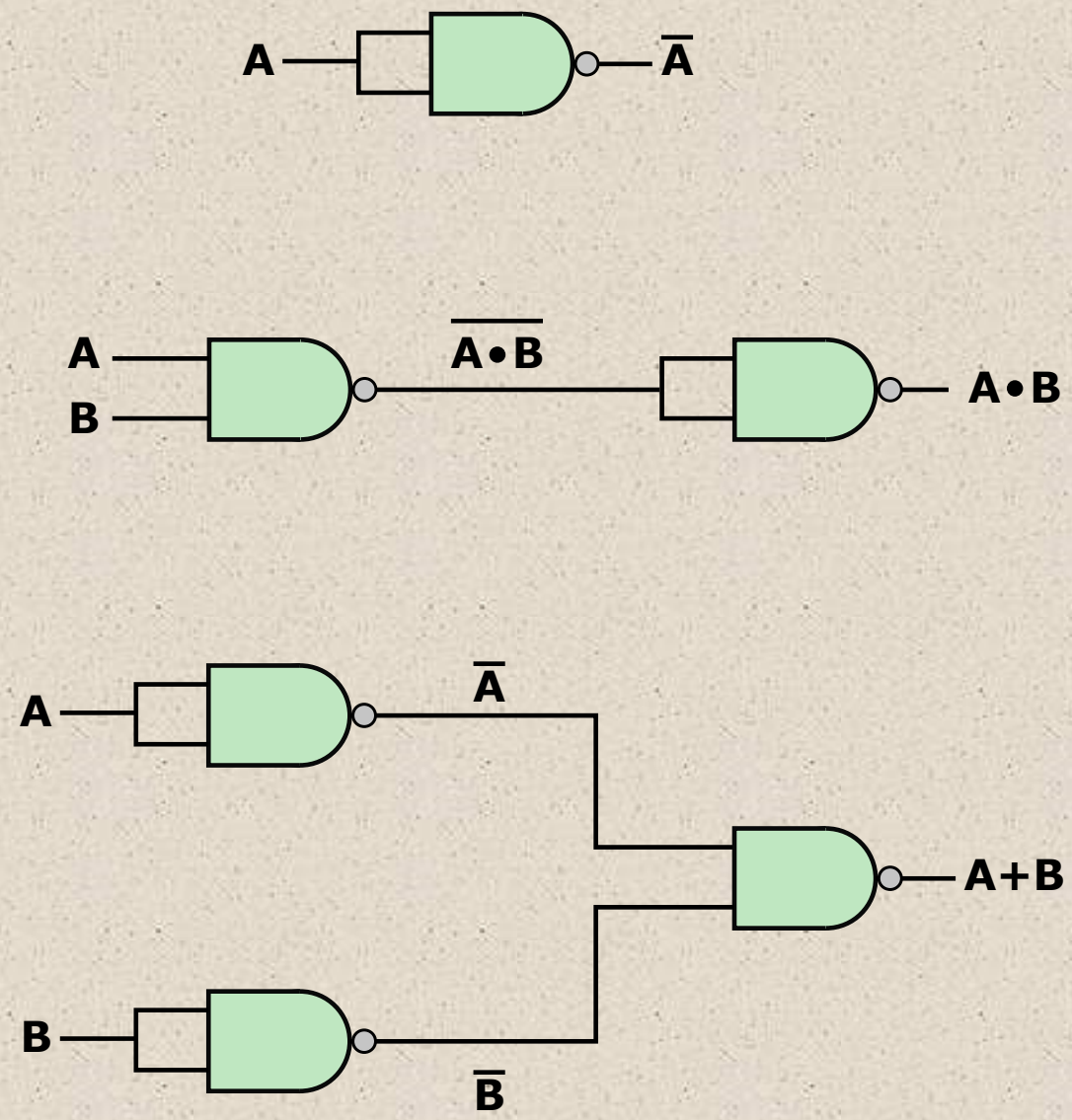


Figure 11.2 Some Uses of NAND Gates

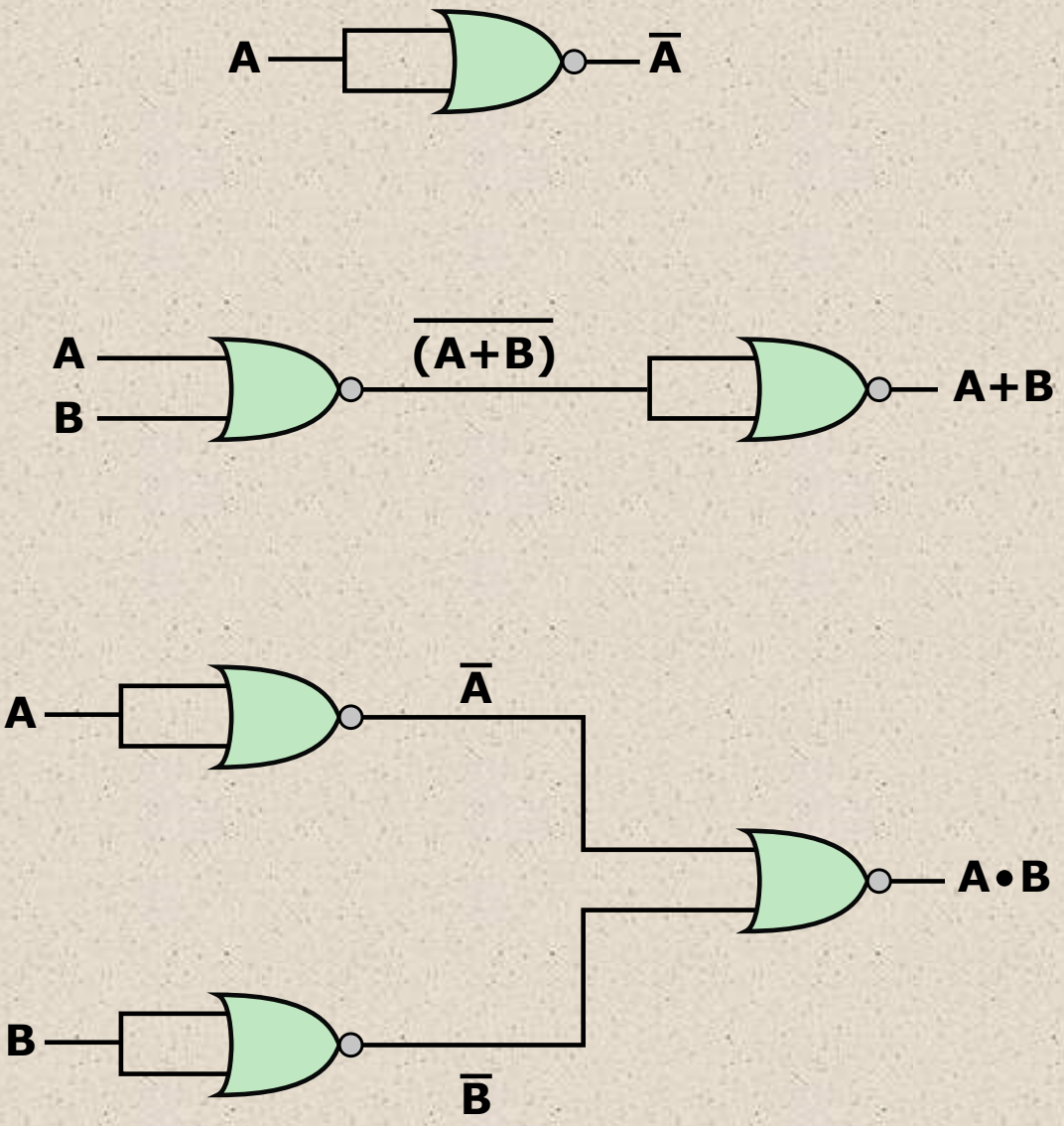
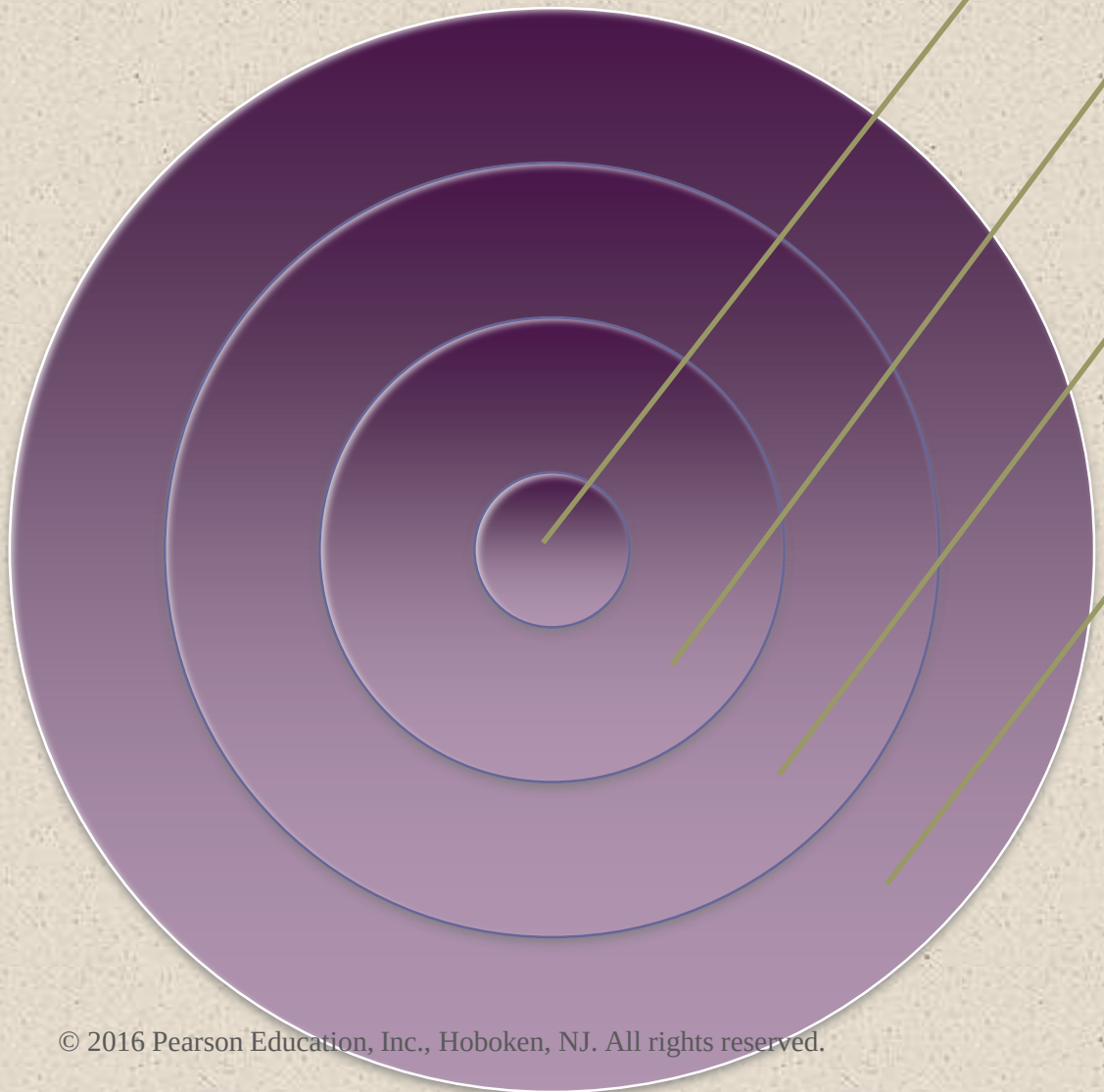


Figure 11.3 Some Uses of NOR Gates

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

Table 11.3
A Boolean Function of Three Variables

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

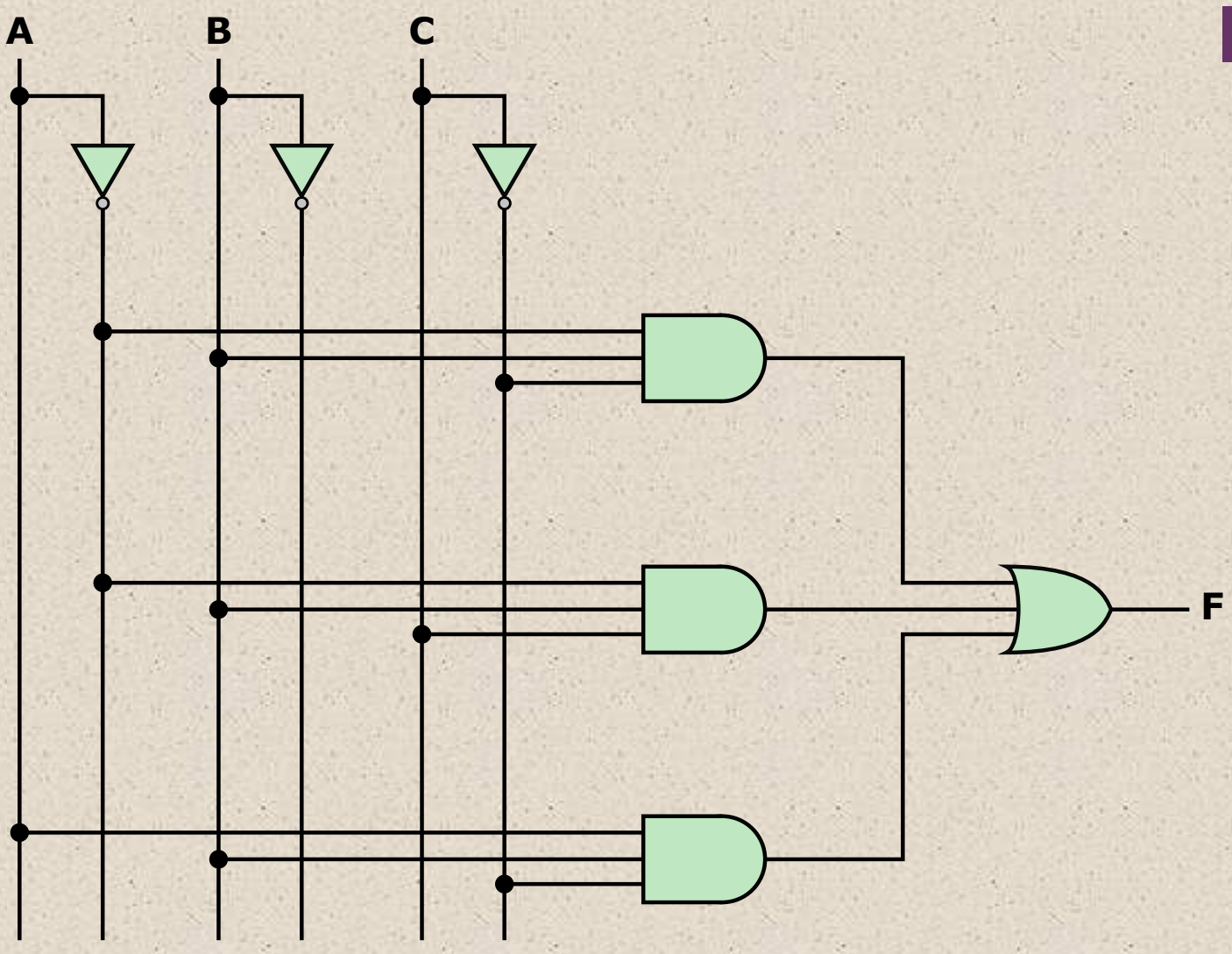


Figure 11.4 Sum-of-Products Implementation of Table 11.3

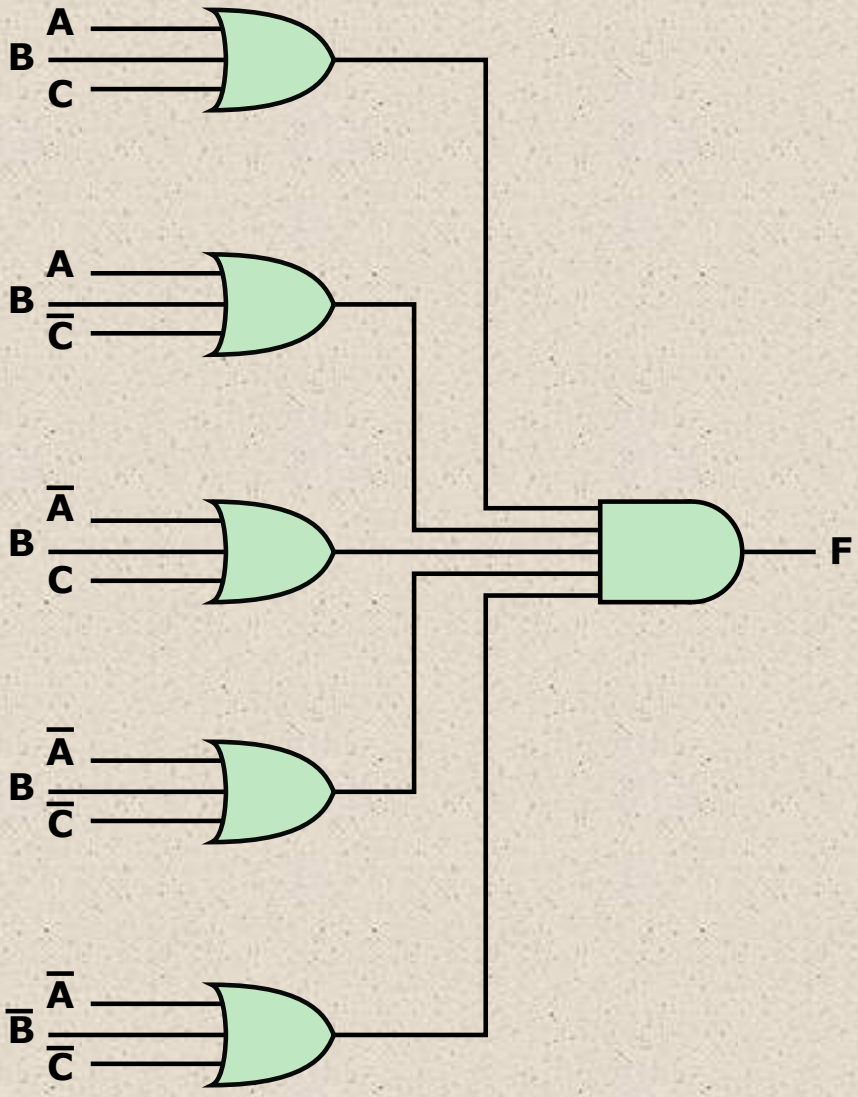


Figure 11.5 Product-of-Sums Implementation of Table 11.3

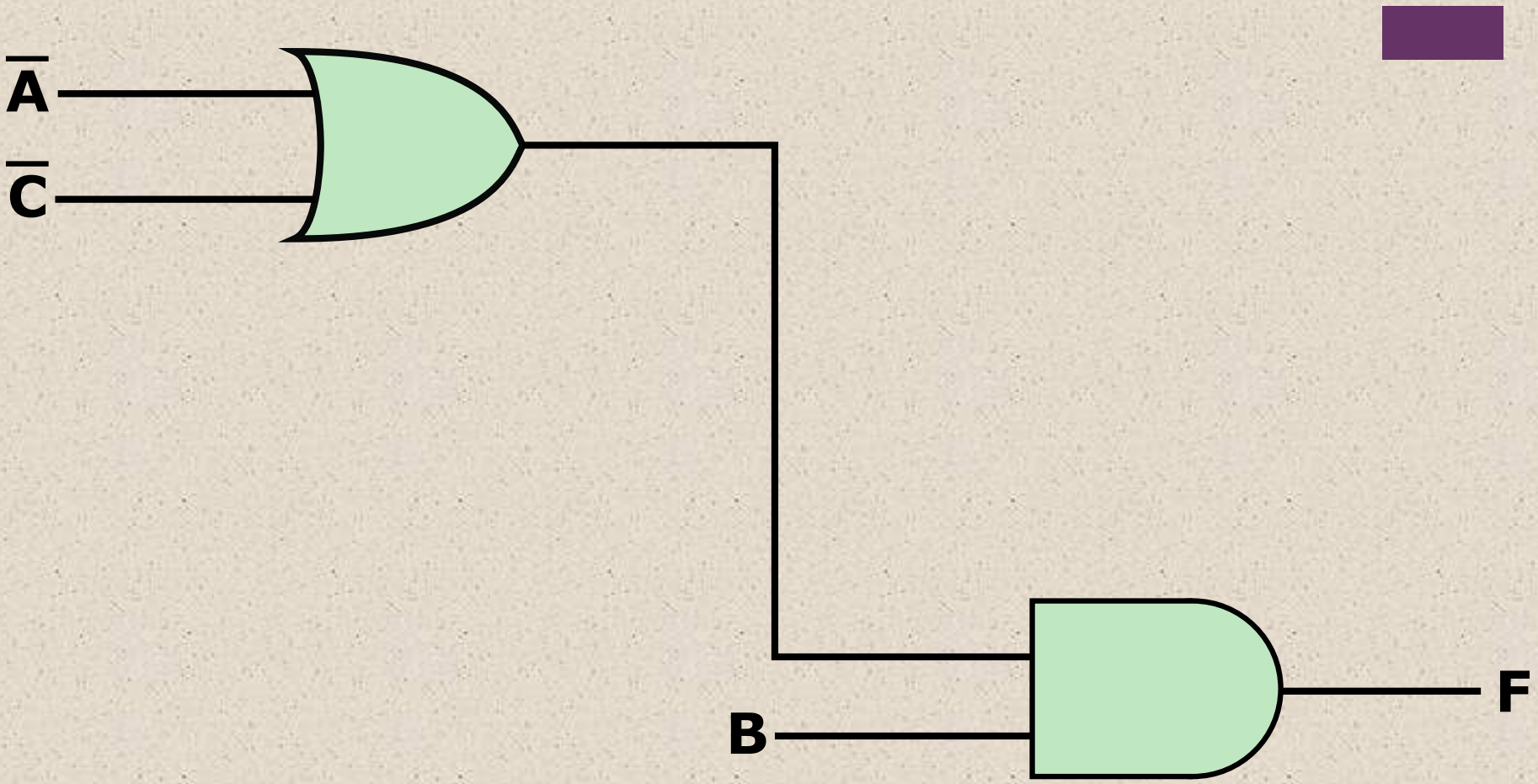


Figure 11.6 Simplified Implementation of Table 11.3

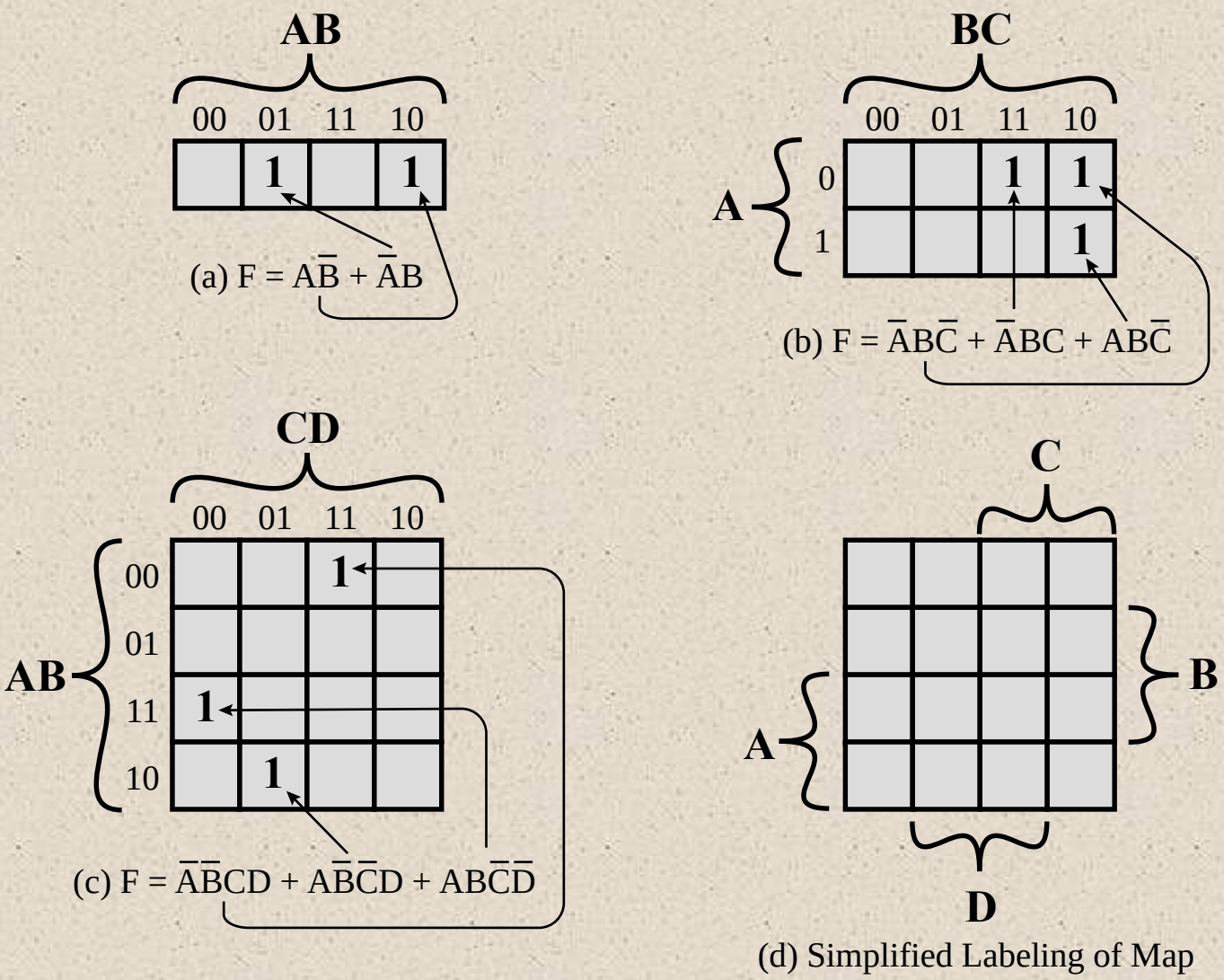
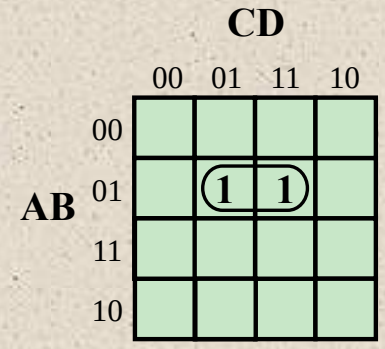
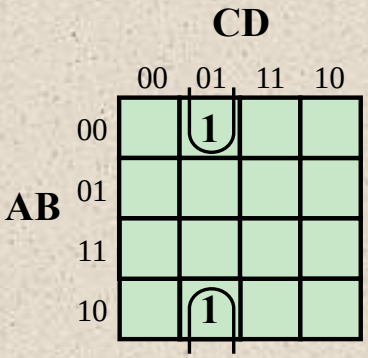


Figure 11.7 The Use of Karnaugh Maps to Represent Boolean Functions

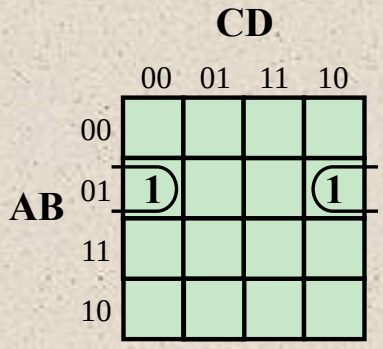
Karnaugh Maps



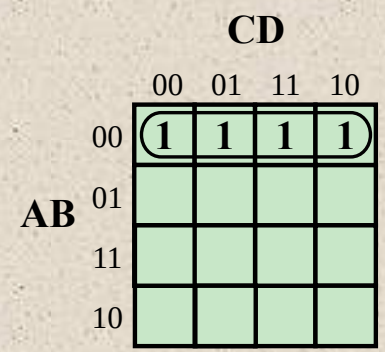
(a) $\bar{A}BD$



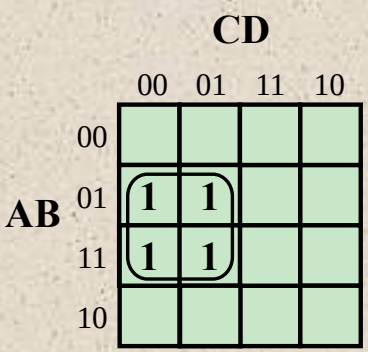
(b) $\bar{B}\bar{C}D$



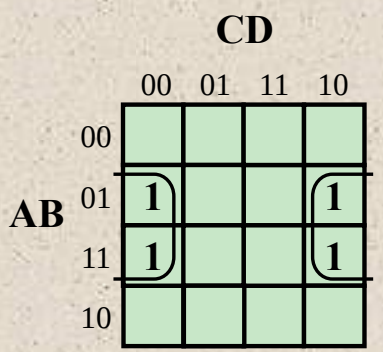
(c) $\bar{A}B\bar{D}$



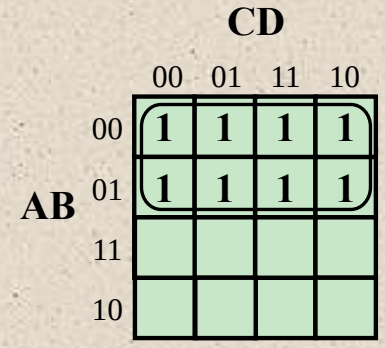
(d) $\bar{A}\bar{B}$



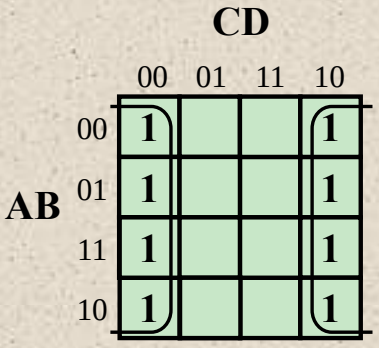
(e) $B\bar{C}$



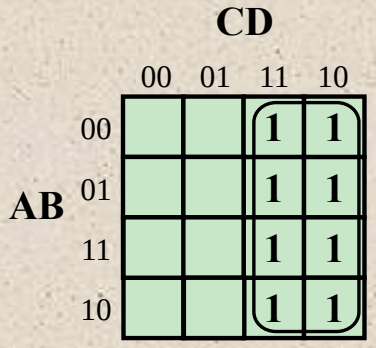
(f) $B\bar{D}$



(g) $\bar{A}$



(h) $\bar{D}$



(i) C

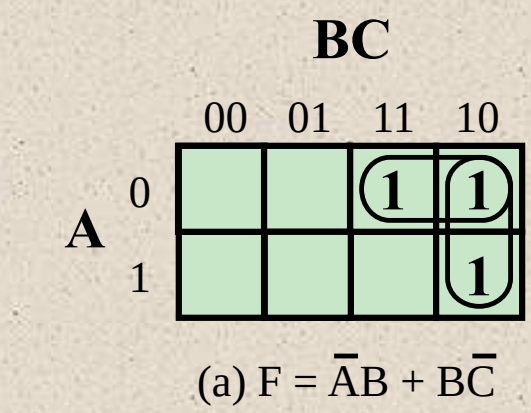


Figure 11.9 Overlapping Groups

Table 11.4
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
	1	0	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

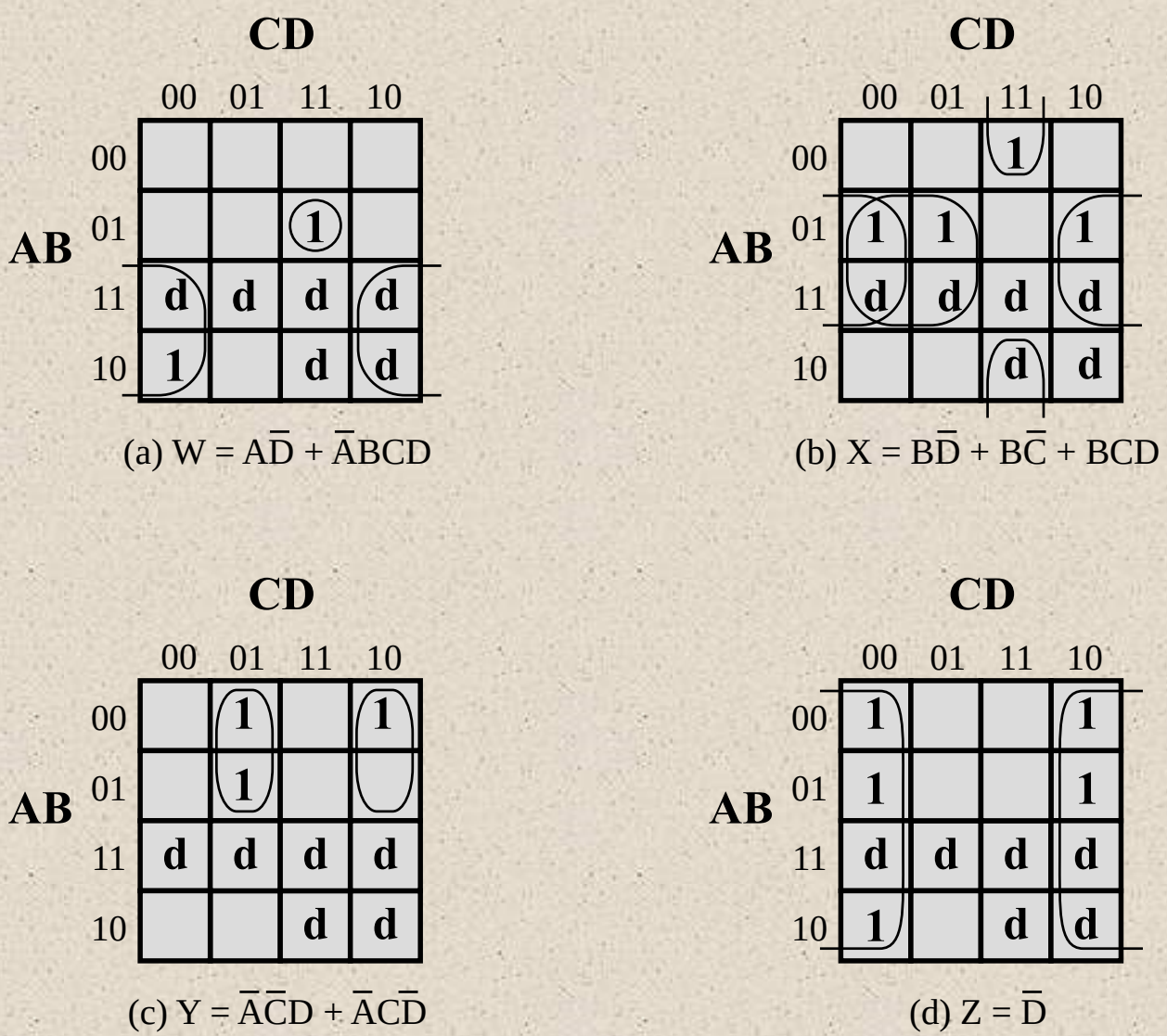


Figure 11.10 Karnaugh Maps for the Incrementer

Table 11.5
First Stage of Quine-McCluskey Method

(for F = ABCD + AB D + AB + A CD + BCD + BC + B D + D)

Product Term	Index	A	B	C	D	
$\bar{A}\bar{B}\bar{C}D$	1	0	0	0	1	✓
$\bar{A}B\bar{C}D$	5	0	1	0	1	✓
$\bar{A}BC\bar{D}$	6	0	1	1	0	✓
$AB\bar{C}\bar{D}$	12	1	1	0	0	✓
$\bar{A}BCD$	7	0	1	1	1	✓
$A\bar{B}CD$	11	1	0	1	1	✓
$AB\bar{C}D$	13	1	1	0	1	✓
$ABCD$	15	1	1	1	1	✓

Table 11.6
Last Stage of Quine-McCluskey Method

(for F = ABCD + AB D + AB + A CD + BCD + BC + B D + D)

	ABCD	AB$\bar{C}$D	AB$\bar{C}\bar{D}$	A$\bar{B}$CD	$\bar{A}$BCD	$\bar{A}$BC$\bar{D}$	$\bar{A}$B$\bar{C}$D	□□□D
BD	X	X			X		X	
$\bar{A}\bar{C}$D							[X]	Ä
$\bar{A}$BC					[X]	Ä		
AB$\bar{C}$		[X]	Ä					
ACD	[X]			Ä				

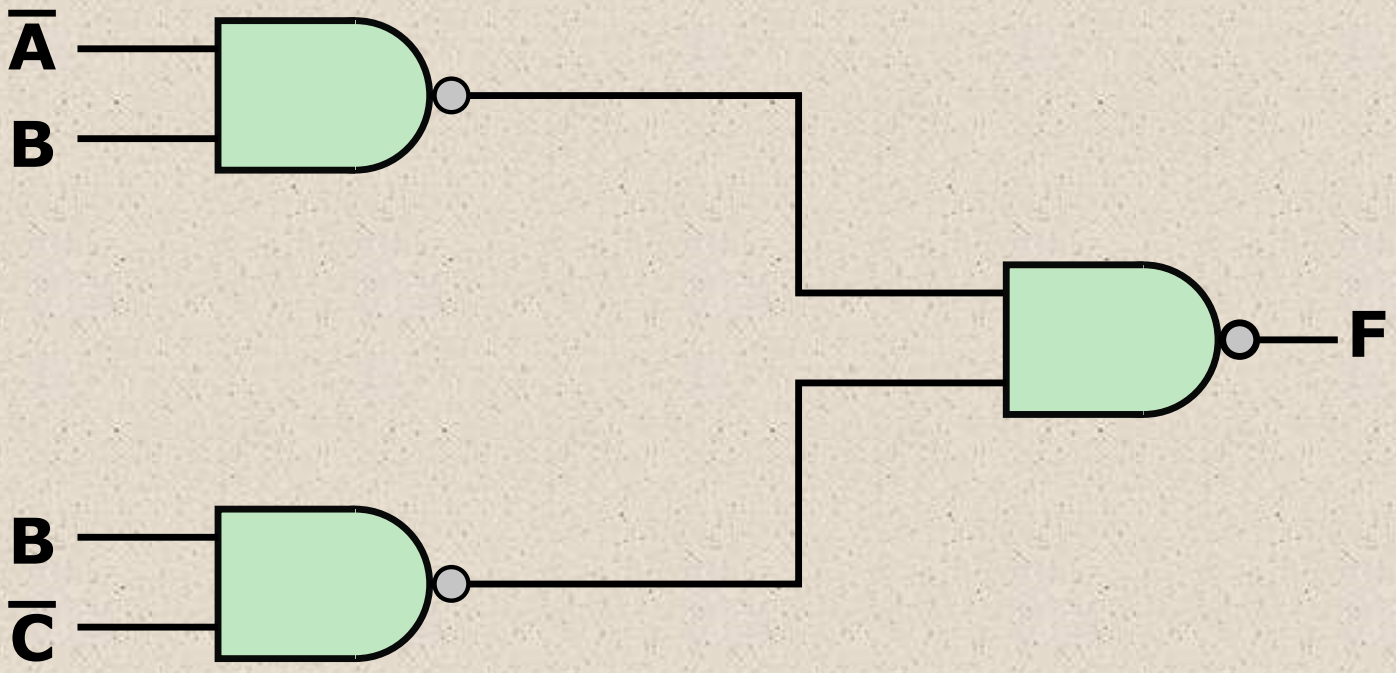


Figure 11.11 NAND Implementation of Table 11.3

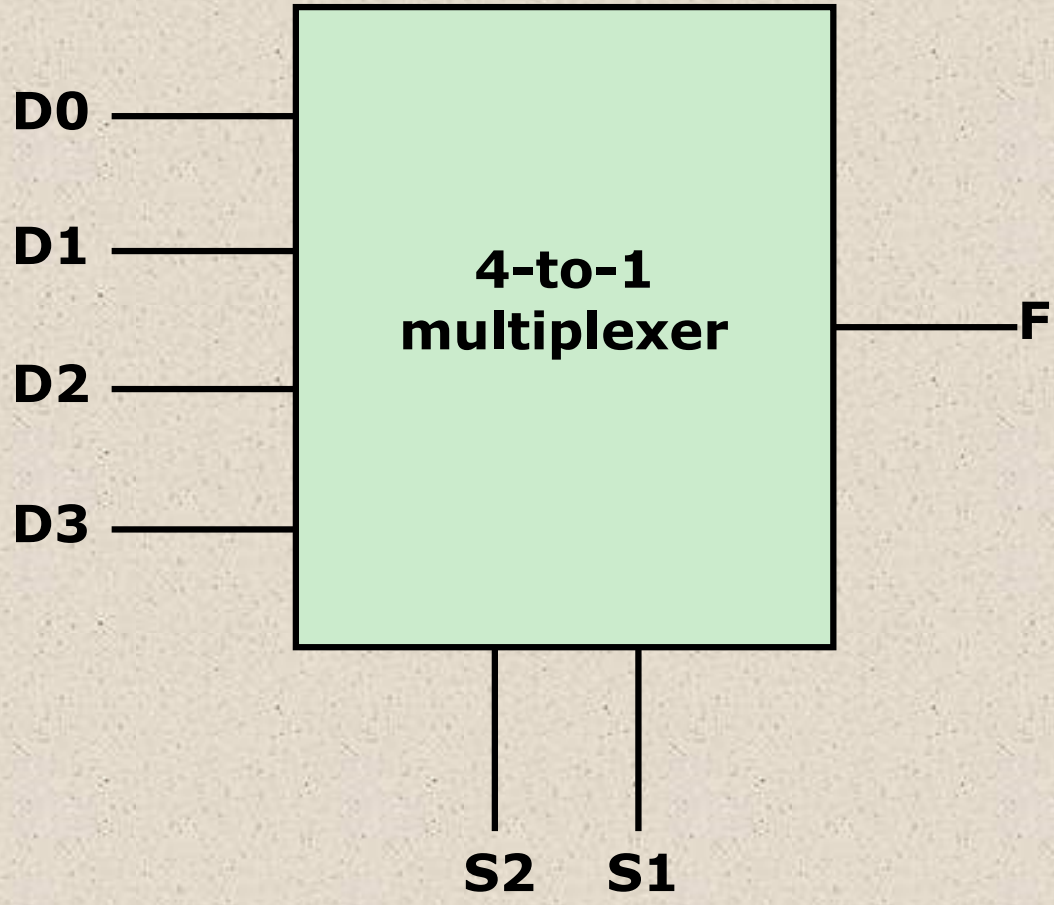


Figure 11.12 4-to-1 Multiplexer Representation

Table 11.7
4-to-1 Multiplexer Truth Table

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

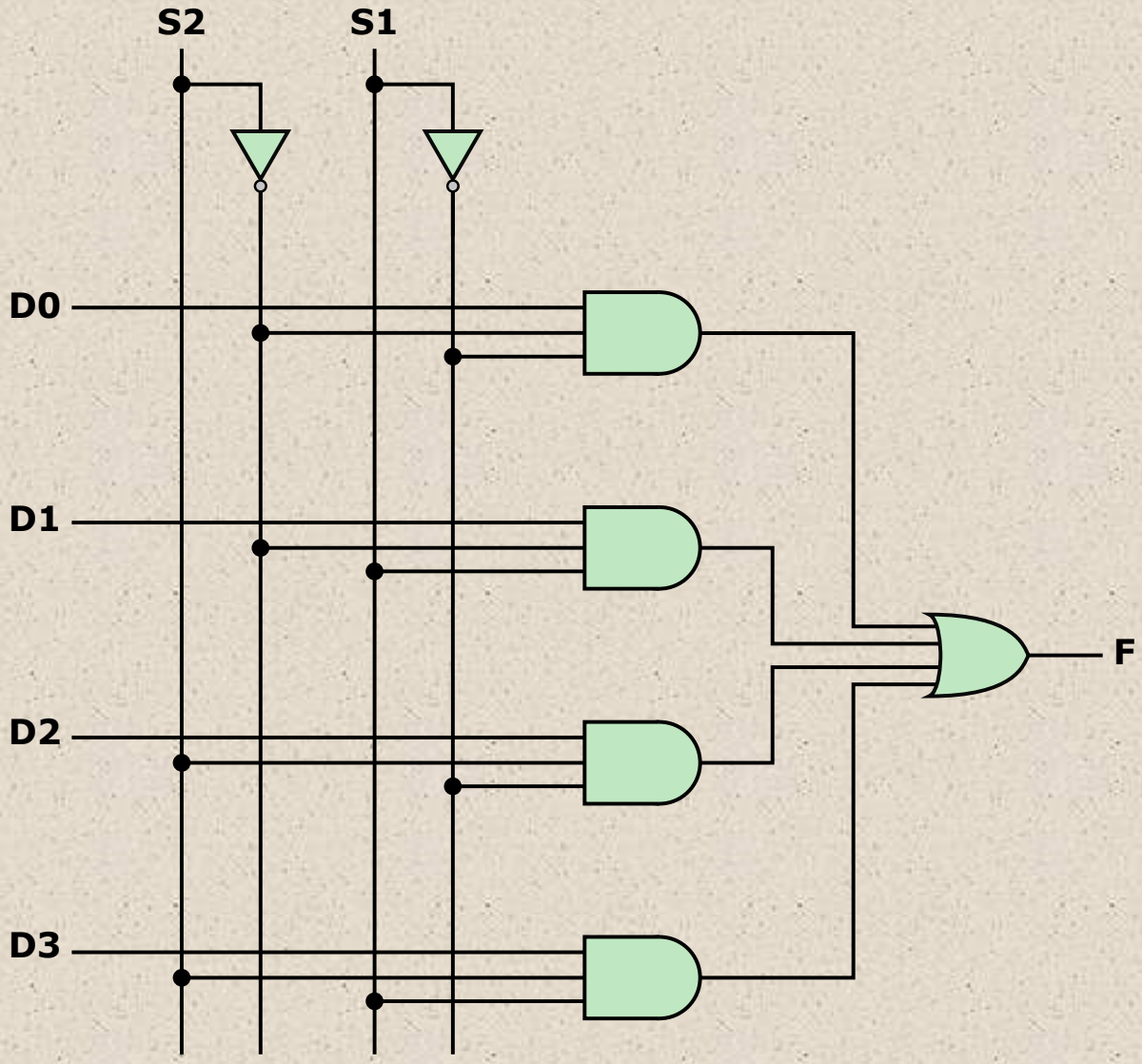


Figure 11.13 Multiplexer Implementation

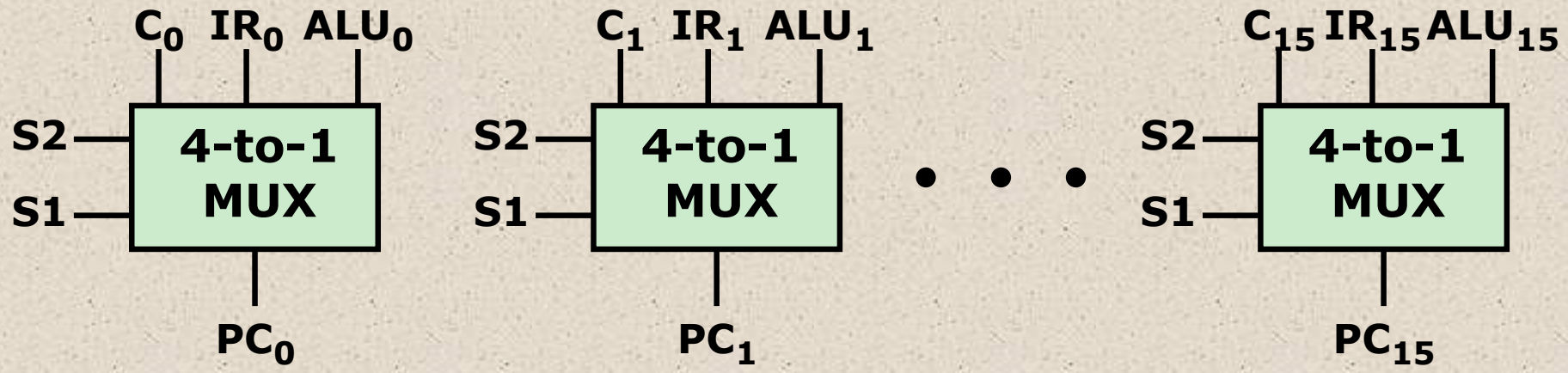


Figure 11.14 Multiplexer Input to Program Counter

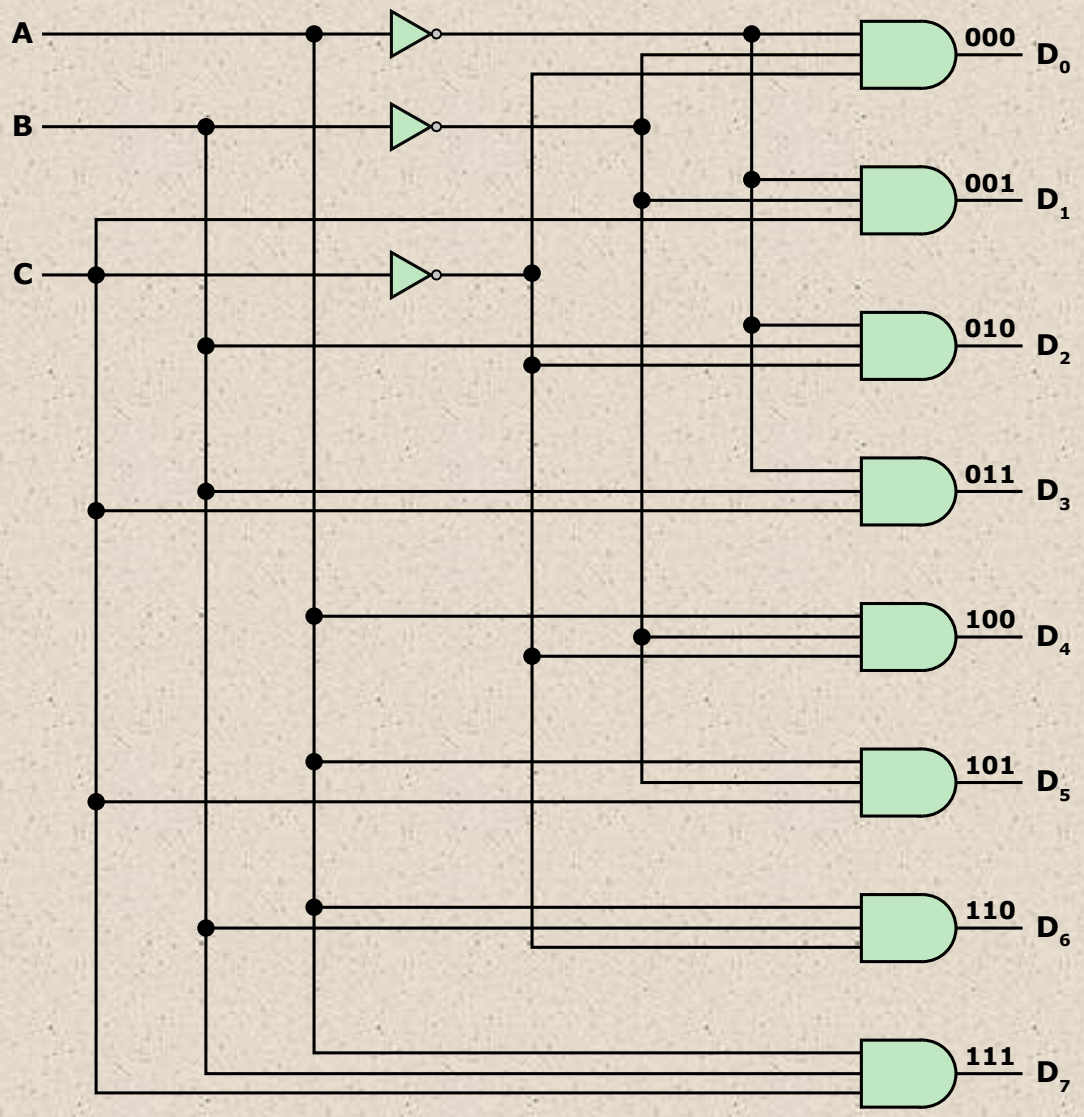


Figure 11.15 Decoder with 3 Inputs and $2^3 = 8$ Outputs

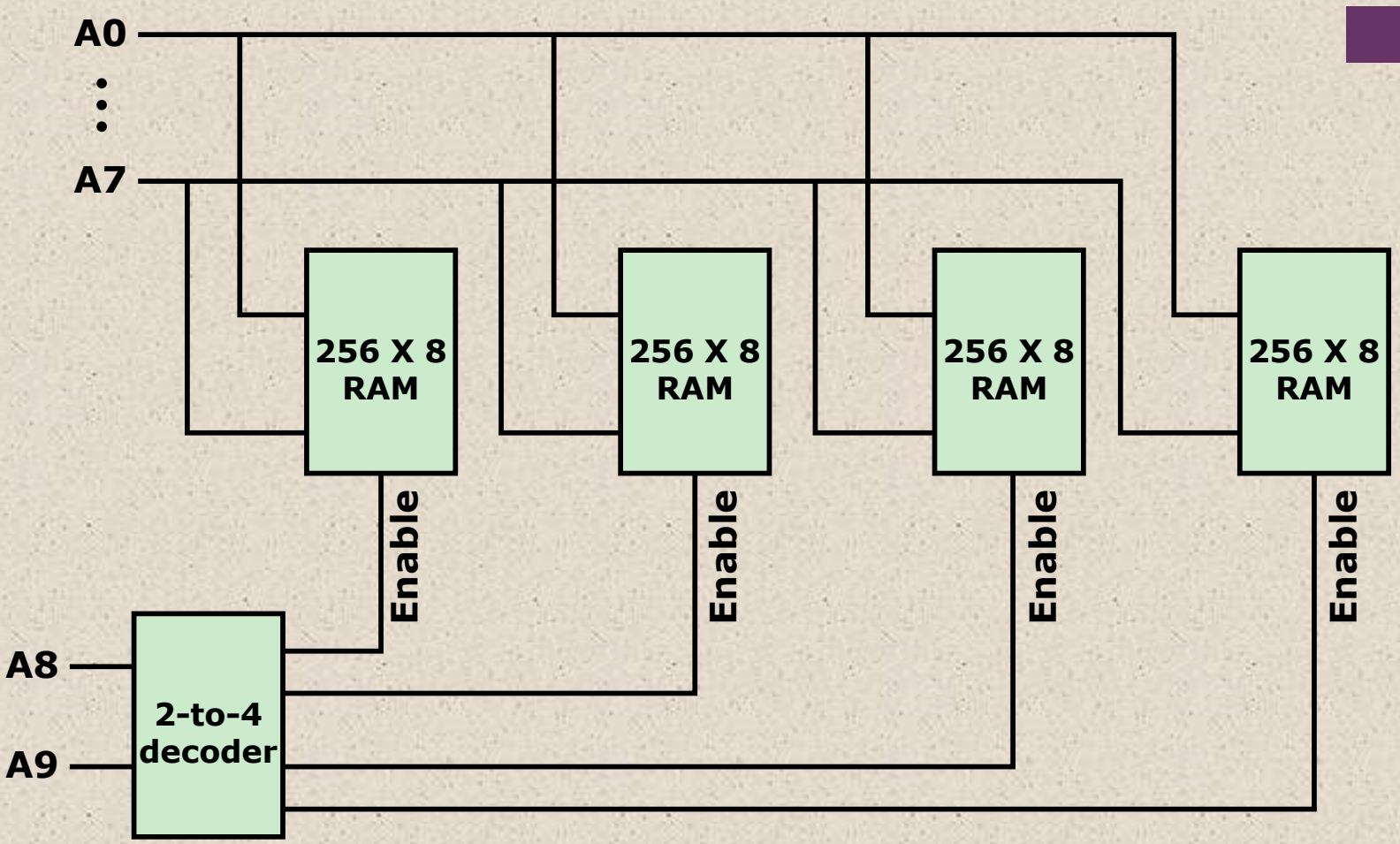


Figure 11.16 Address Decoding

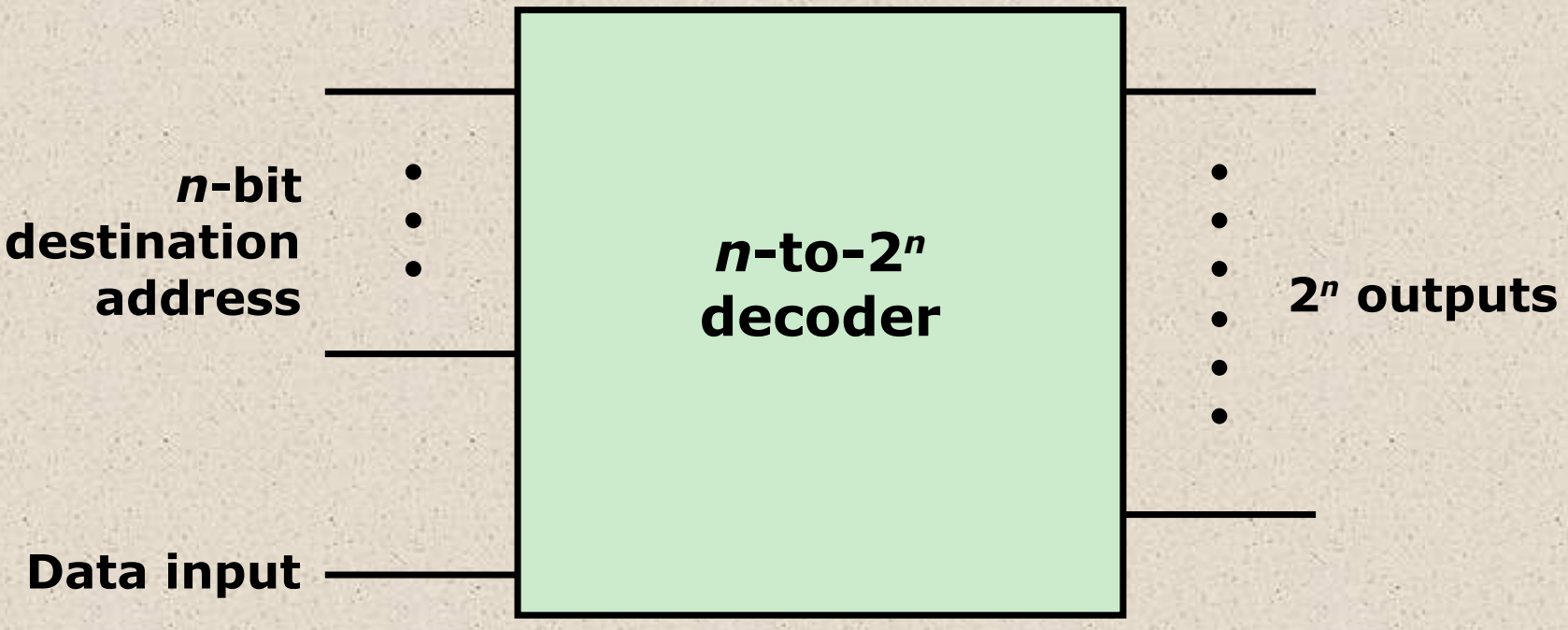


Figure 11.17 Implementation of a Demultiplexer Using a Decoder

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

Table 11.8
Truth Table for a ROM

Input				Output			
X_1	X_2	X_3	X_4	Z_1	Z_2	Z_3	Z_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

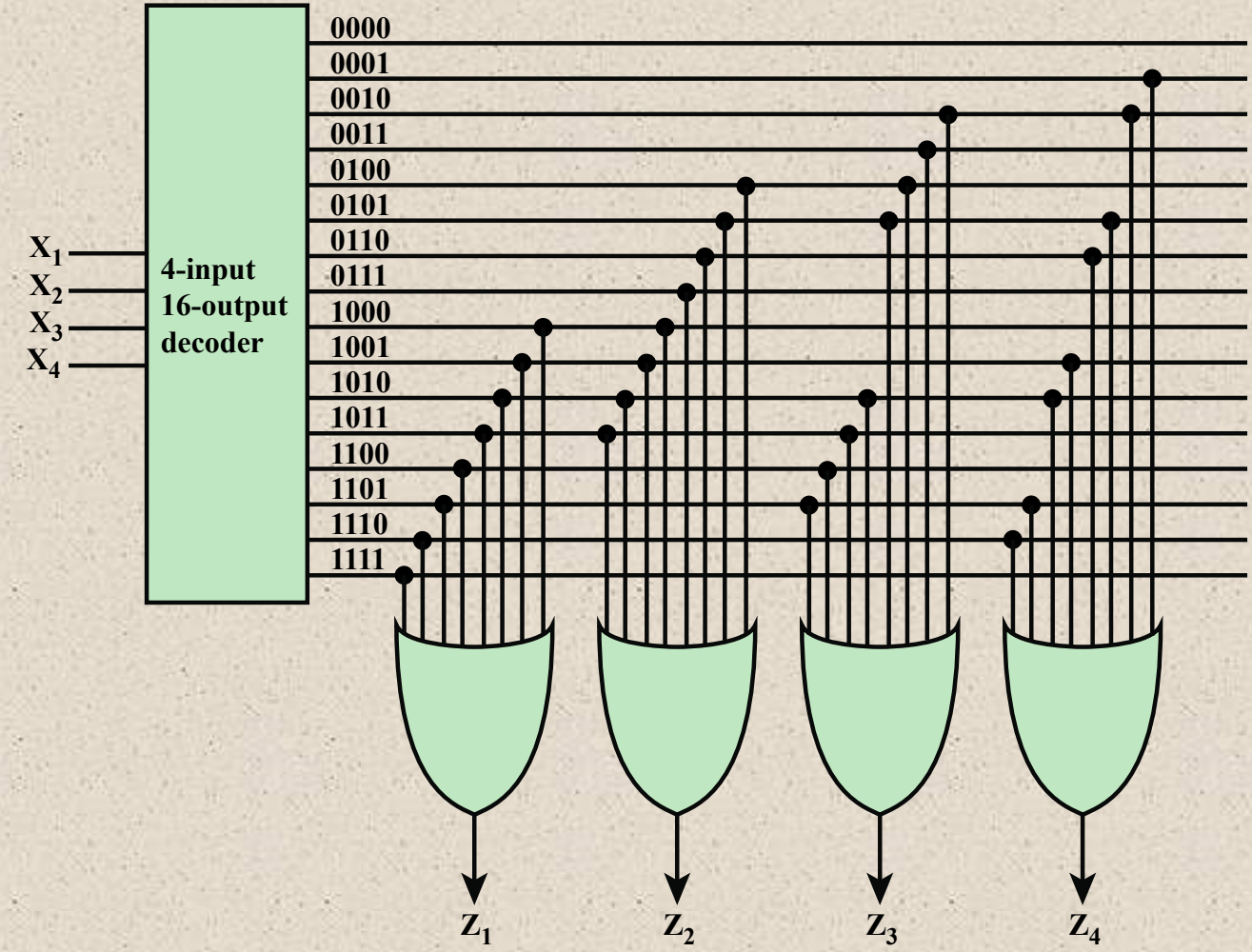


Figure 11.18 A 64-Bit ROM

Table 11.9
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

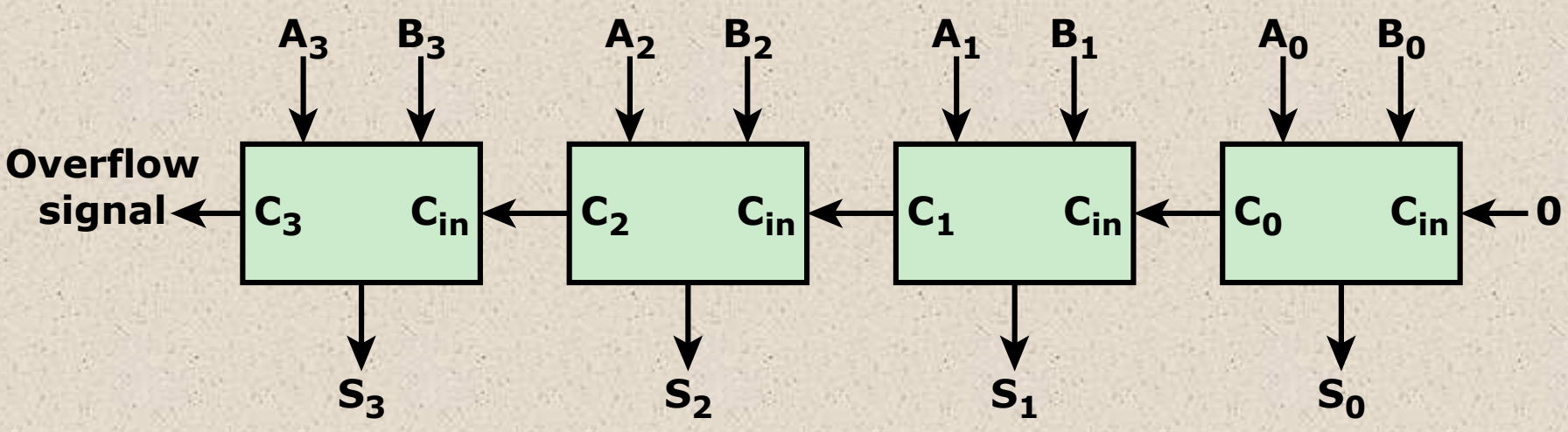


Figure 11.19 4-Bit Adder

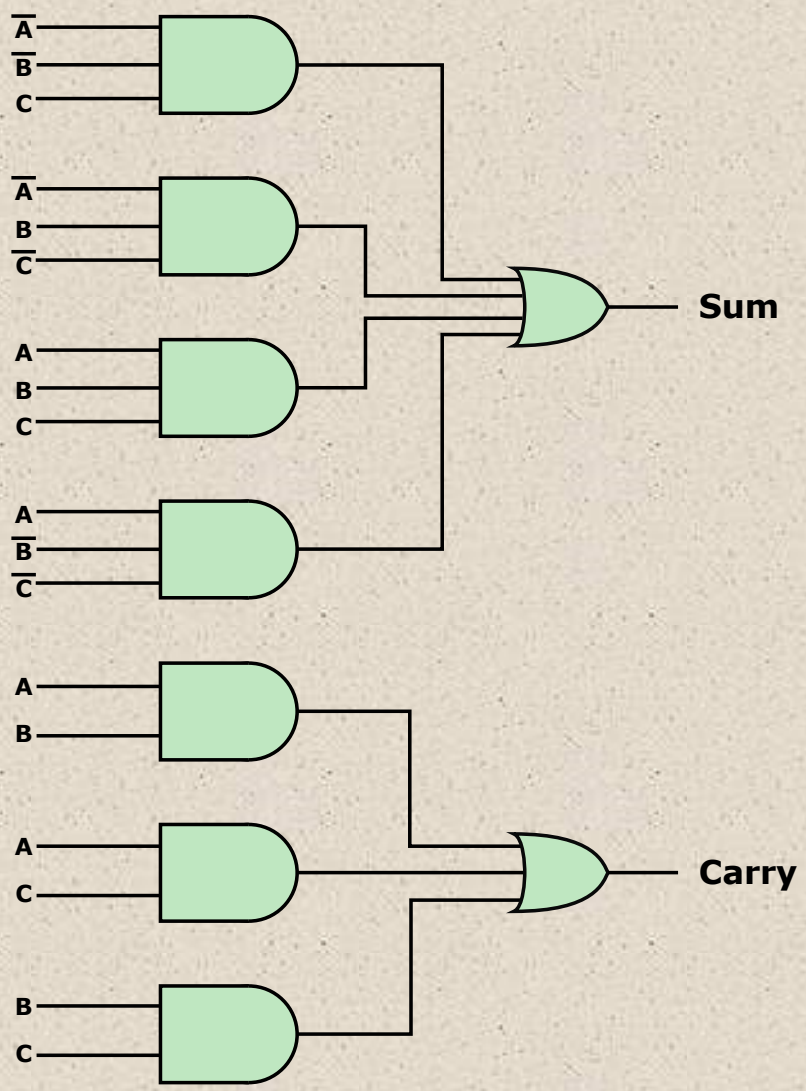


Figure 11.20 Implementation of an Adder

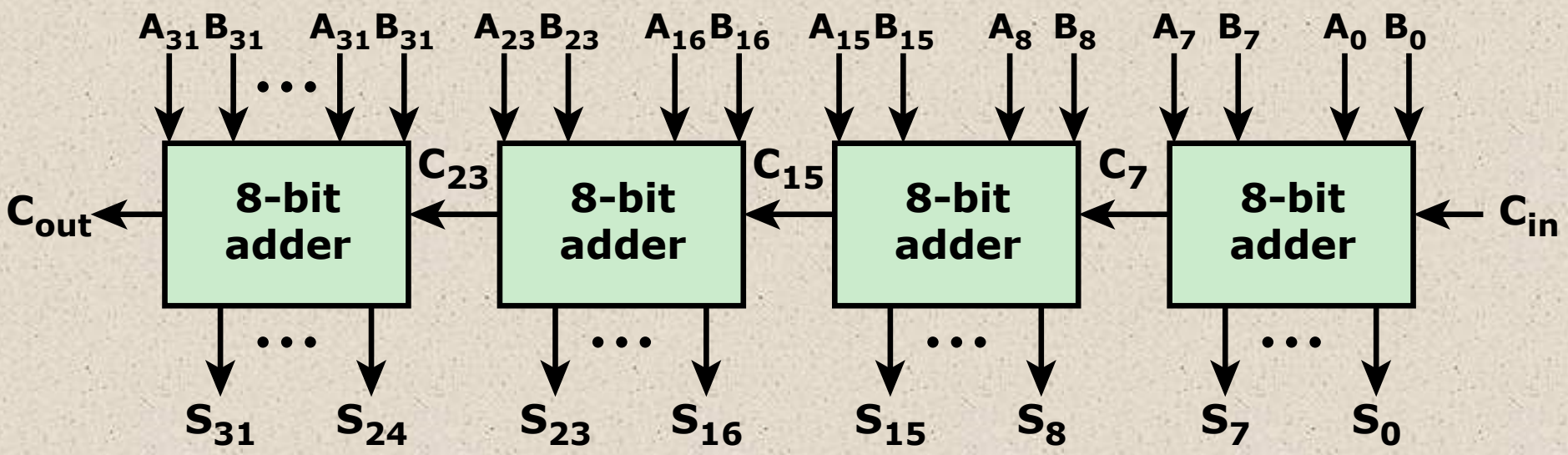
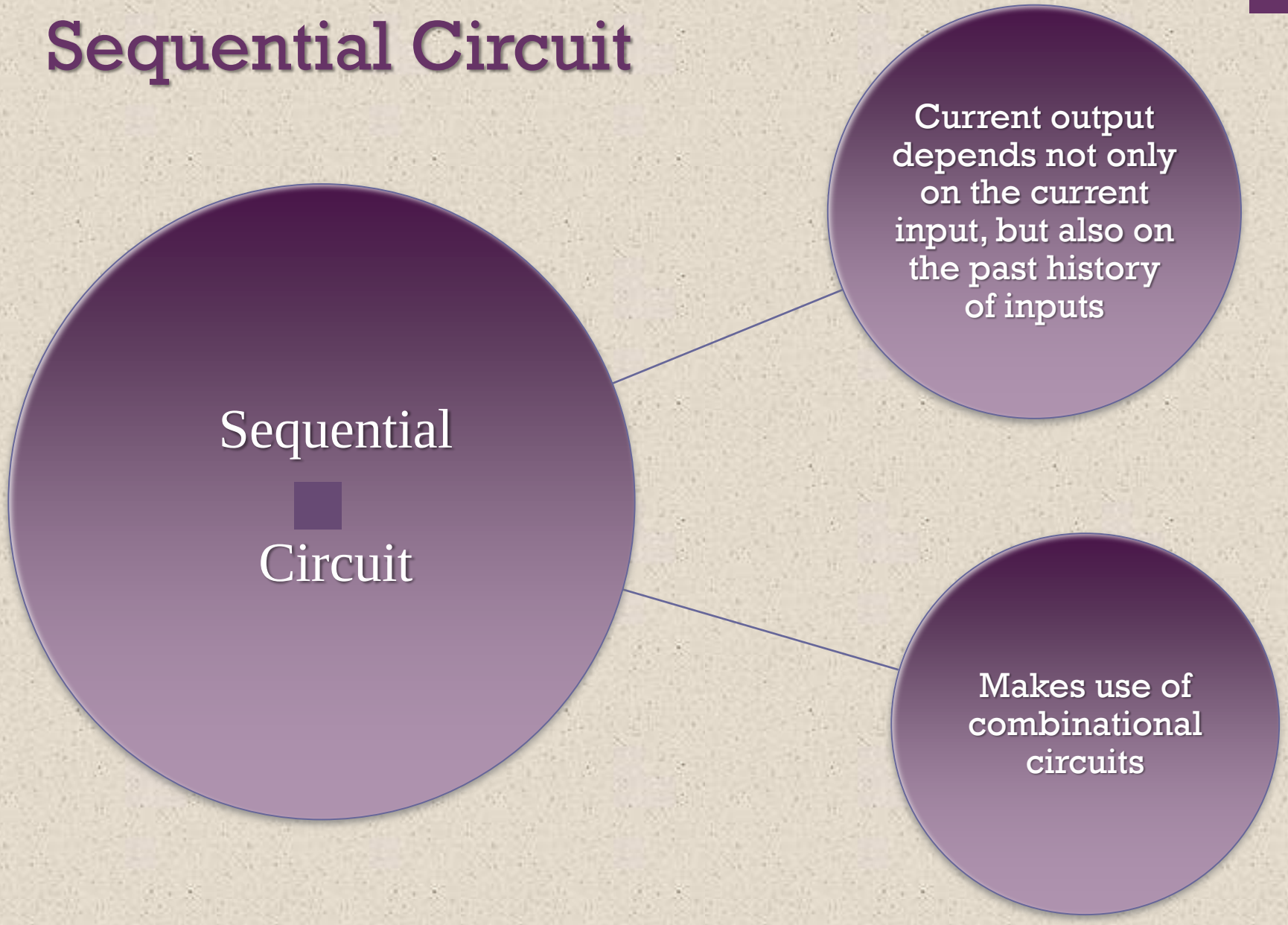


Figure 11.21 Construction of a 32-Bit Adder Using 8-Bit Adders

Sequential Circuit



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.

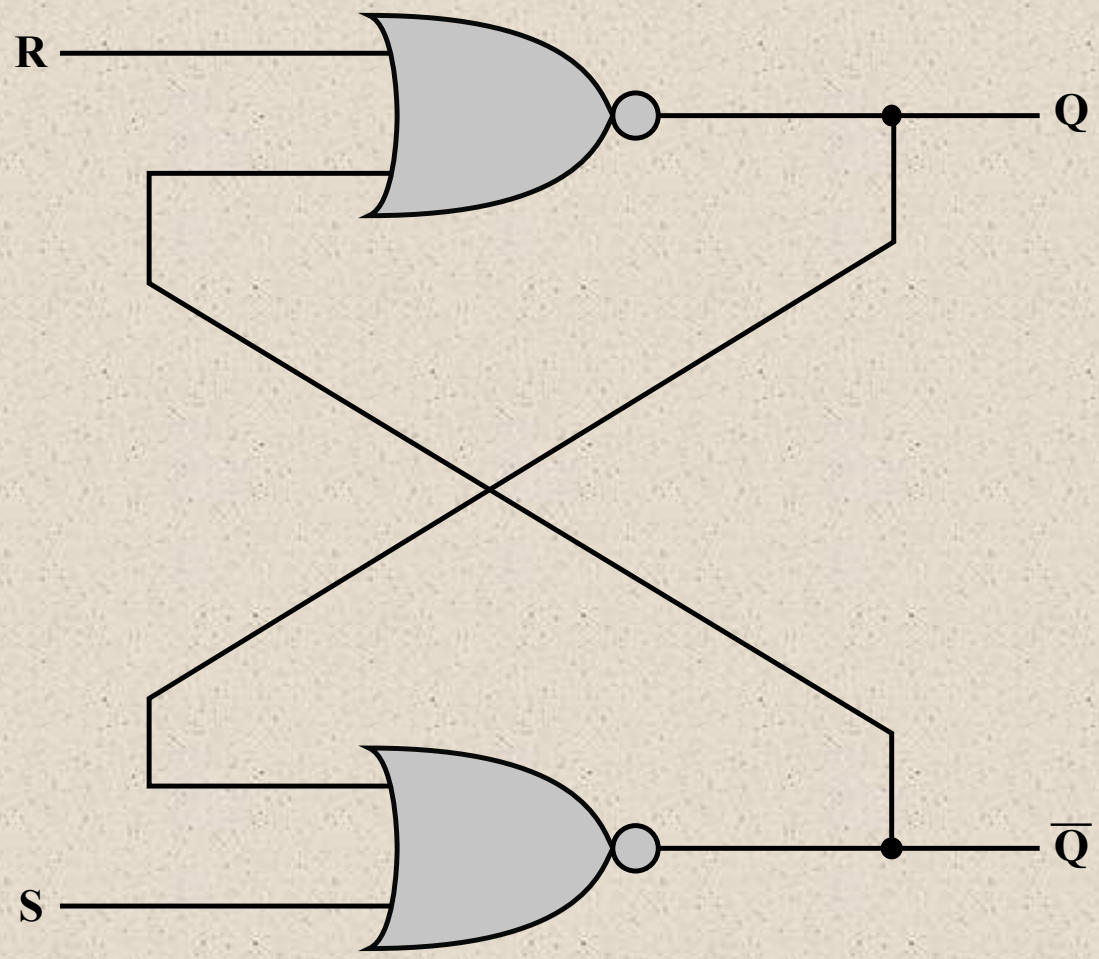


Figure 11.22 The S-R Latch Implemented with NOR Gates

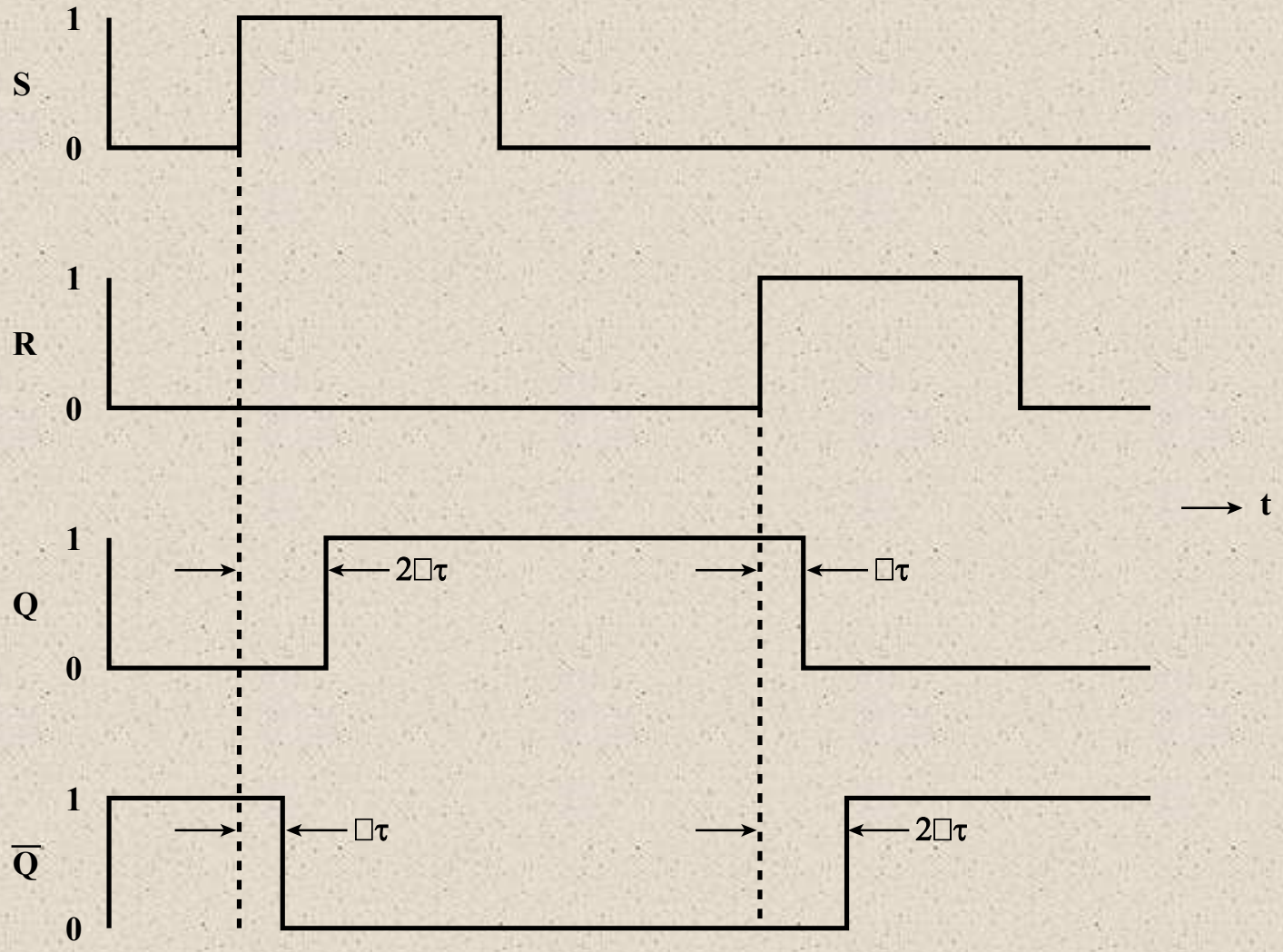


Figure 11.23 NOR S-R Latch Timing Diagram

Table 11.10 The S-R Latch

(a) Characteristic Table

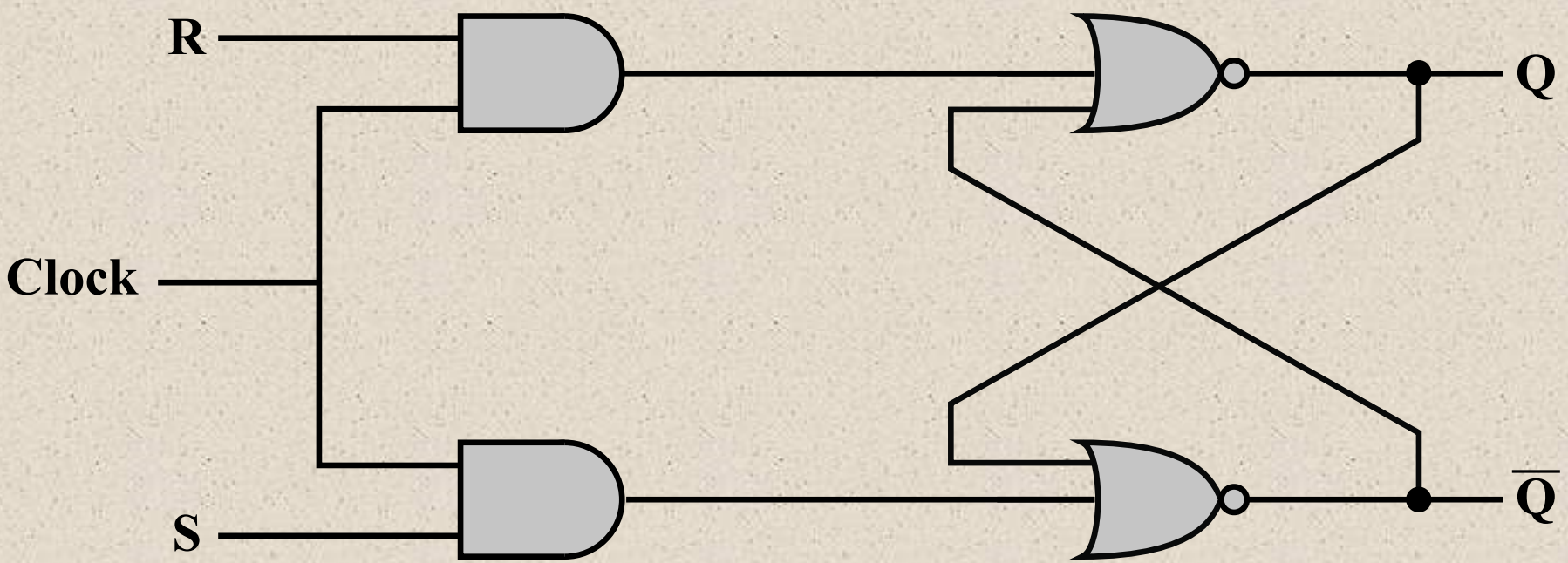
Current Inputs SR	Current State Q_n	Next State Q_{n+1}
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	—
11	1	—

(b) Simplified Characteristic Table

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	—

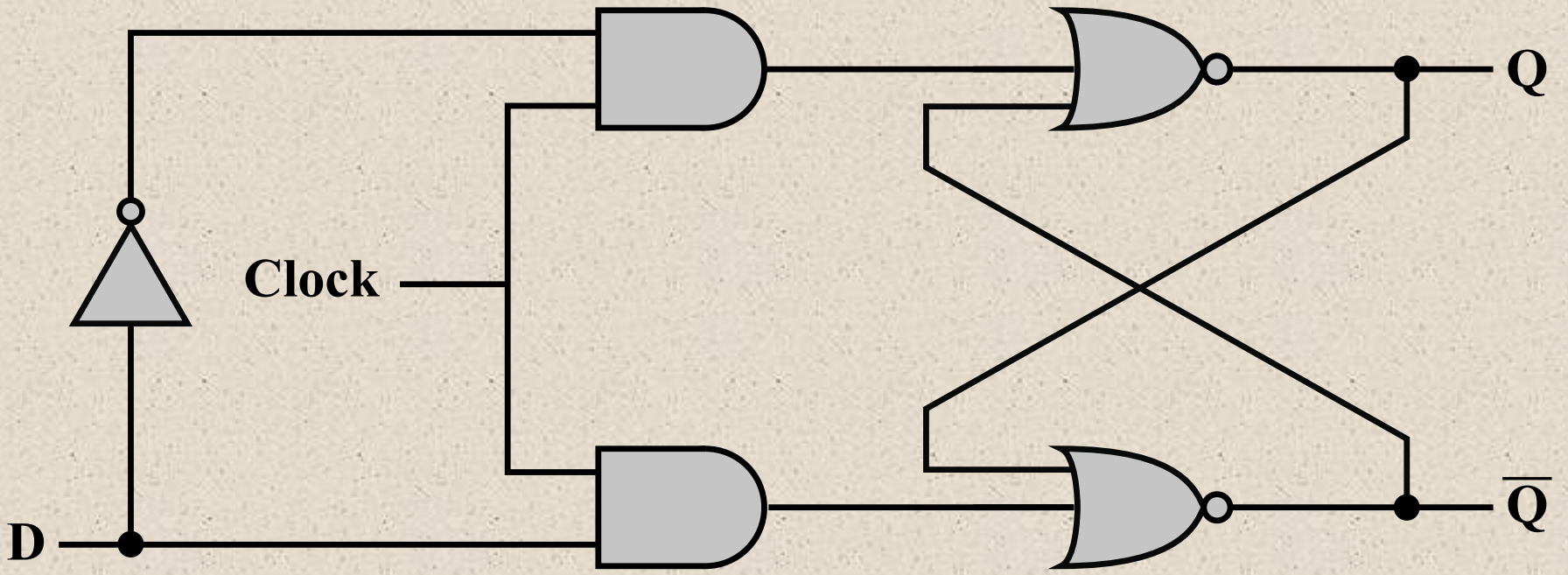
(c) Response to Series of Inputs

t	0	1	2	3	4	5	6	7	8	9
S	1	0	0	0	0	0	0	0	1	0
R	0	0	0	1	0	0	1	0	0	0
Q_{n+1}	1	1	1	0	0	0	0	0	1	1



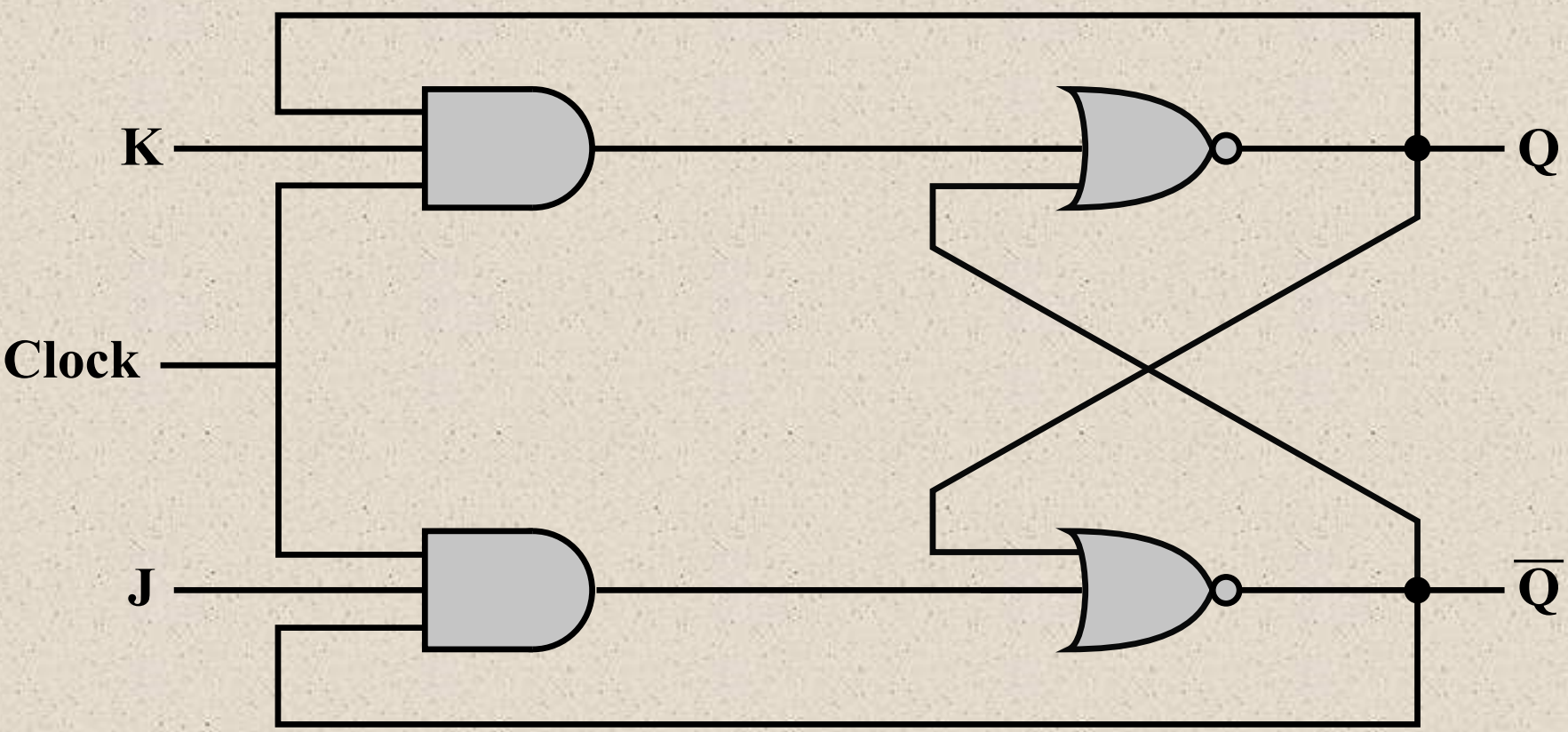
Asynchronous operation for dealing time delay

Figure 11.24 Clocked S-R Flip Flop



Data flip-flop (RAM; time delay)

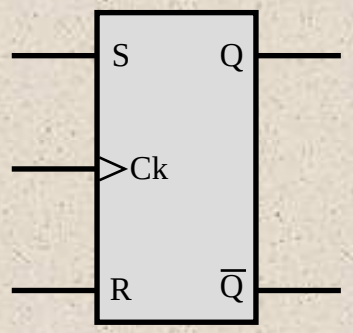
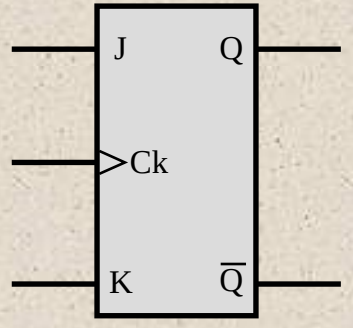
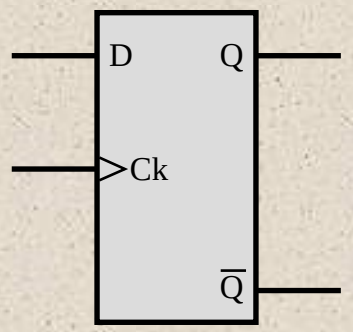
Figure 11.25 D Flip Flop



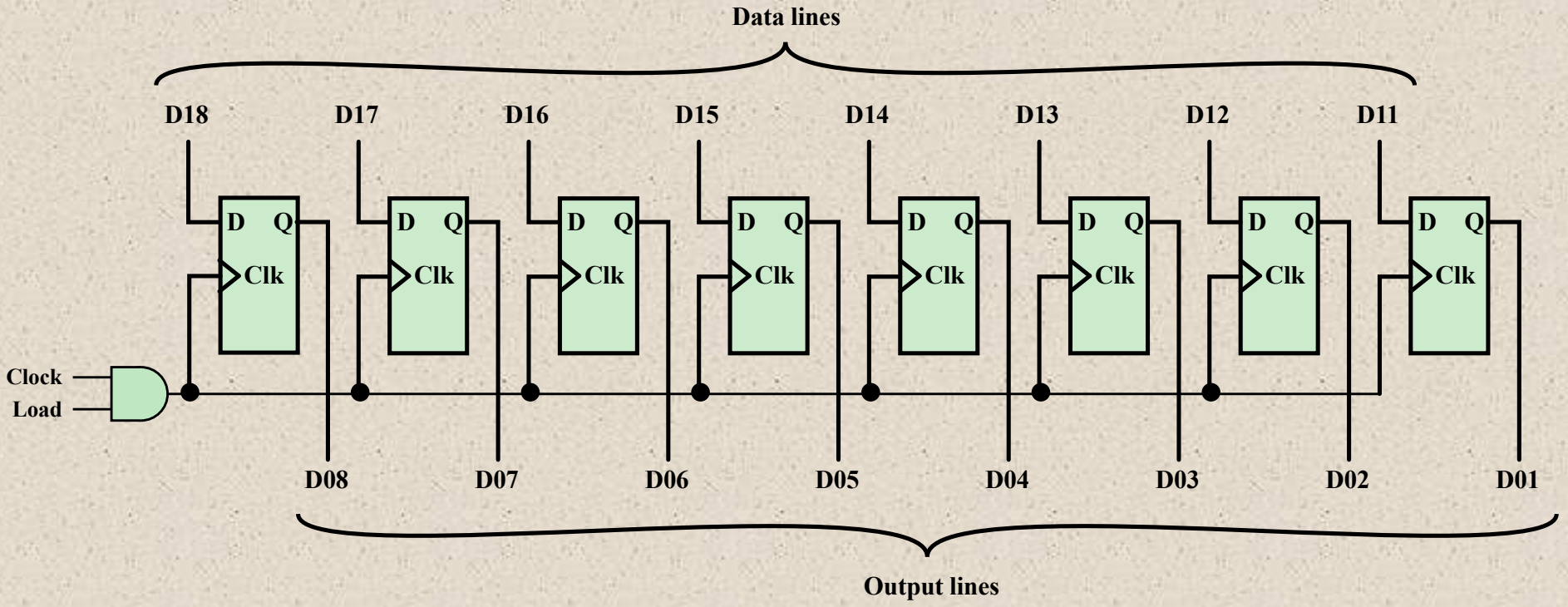
Gate implementation

Figure 11.26 J-K Flip Flop

Basic Flip-Flops

Name	Graphical Symbol	Truth Table
S-R		S R Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 –
J-K		J K Q_{n+1} 0 0 Q_n 0 1 0 1 0 1 1 1 $\overline{Q_n}$
D		D Q_{n+1} 0 0 1 1

A set of 1-bit memories



Output of multiplexers

Figure 11.28 8-Bit Parallel Register

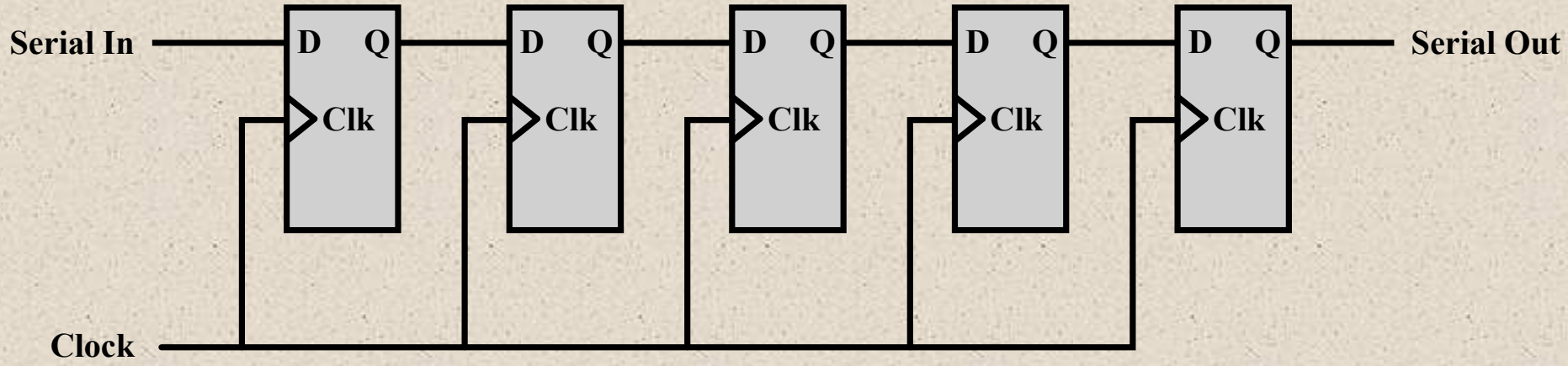


Figure 11.29 5-Bit Shift Register

Interface to serial I/O devices; Logical shift; Rotate functions

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

Ripple counter

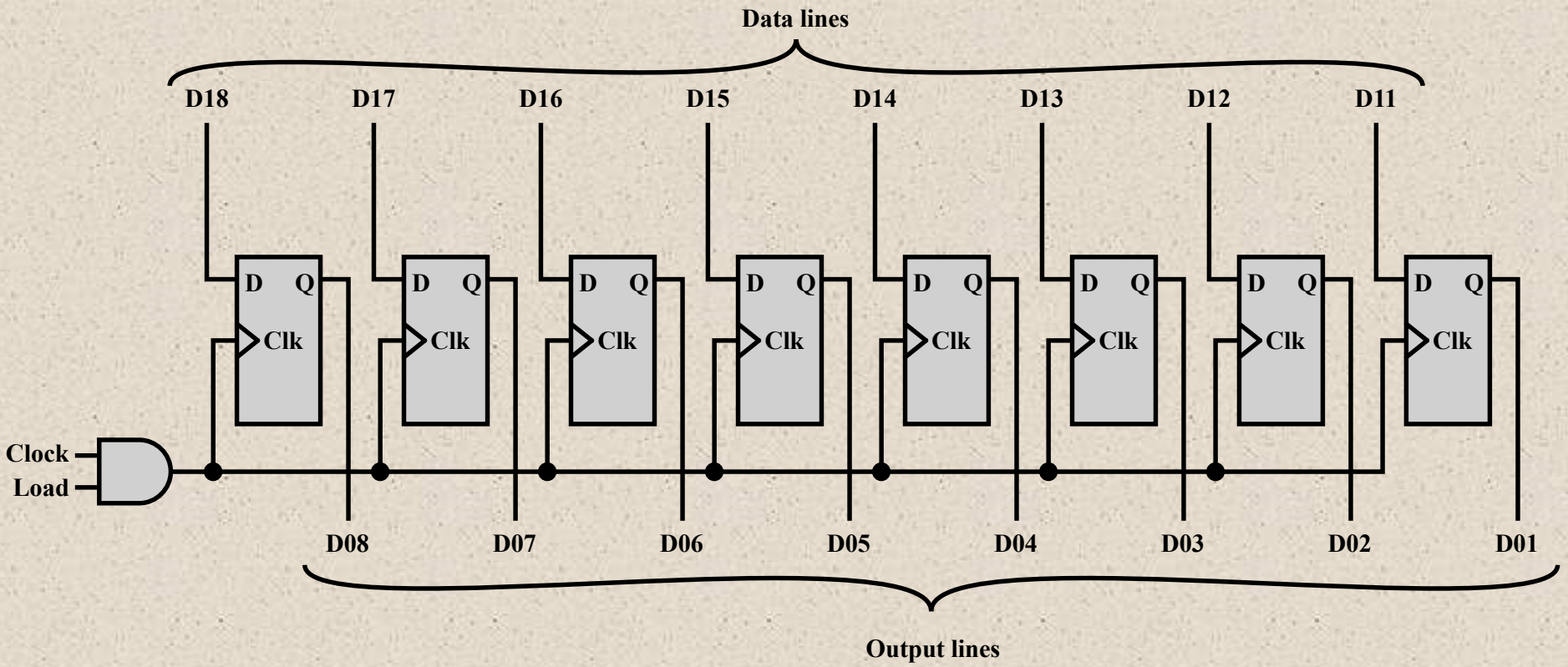


Figure 11.30 8-Bit Parallel Register

Design of a Synchronous Counter

(a) Truth table

C	B	A	Jc	Kc	Jb	Kb	Ja	Ka
0	0	0	0	d	0	d	1	d
0	0	1	0	d	1	d	d	1
0	1	0	0	d	d	0	1	d
0	1	1	1	d	d	1	d	1
1	0	0	d	0	0	d	1	d
1	0	1	d	0	1	d	d	1
1	1	0	d	0	d	0	1	d
1	1	1	d	1	d	1	d	1

(b) Karnaugh maps

Jc = BA C

C \ BA	00	01	11	10
0			1	
1	d	d	d	d

Kc = BA C

C \ BA	00	01	11	10
0	d	d	d	d
1			1	

Jb = A C

C \ BA	00	01	11	10
0		1	d	d
1		1	d	d

Kb = A C

C \ BA	00	01	11	10
0	d	d	1	
1	d	d	1	

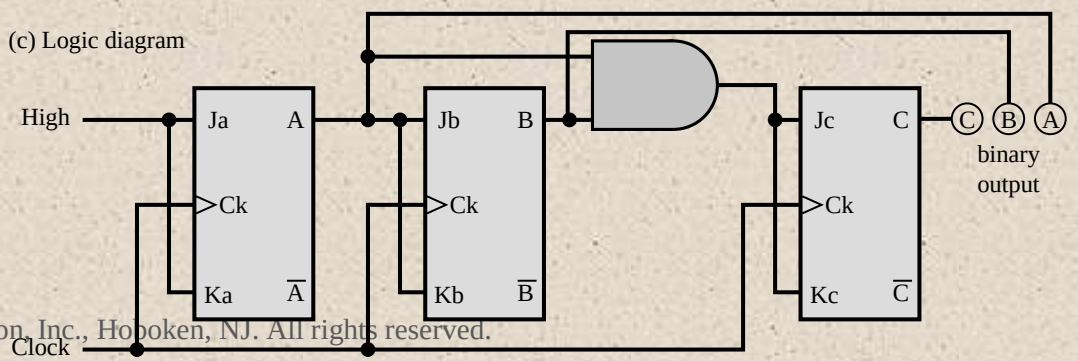
Ja = 1 C

C \ BA	00	01	11	10
0	1	d	d	1
1	1	d	d	1

Ka = 1 C

C \ BA	00	01	11	10
0	d	1	1	d
1	d	1	1	d

(c) Logic diagram



Programmable Logic Device (PLD)

A general term that refers to any type of integrated circuit used for implementing digital hardware, where the chip can be configured by the end user to realize different designs. Programming of such a device often involves placing the chip into a special programming unit, but some chips can also be configured "in-system". Also referred to as a field-programmable device (FPD).

Programmable Logic Array (PLA)

A relatively small PLD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable.

Programmable Array Logic (PAL)

A relatively small PLD that has a programmable AND-plane followed by a fixed OR-plane.

Simple PLD (SPLD)

A PLA or PAL.

Complex PLD (CPLD)

A more complex PLD that consists of an arrangement of multiple SPLD-like blocks on a single chip.

Field-Programmable Gate Array (FPGA)

A PLD featuring a general structure that allows very high logic capacity. Whereas CPLDs feature logic resources with a wide number of inputs (AND planes), FPGAs offer more narrow logic resources. FPGAs also offer a higher ratio of flip-flops to logic resources than do CPLDs.

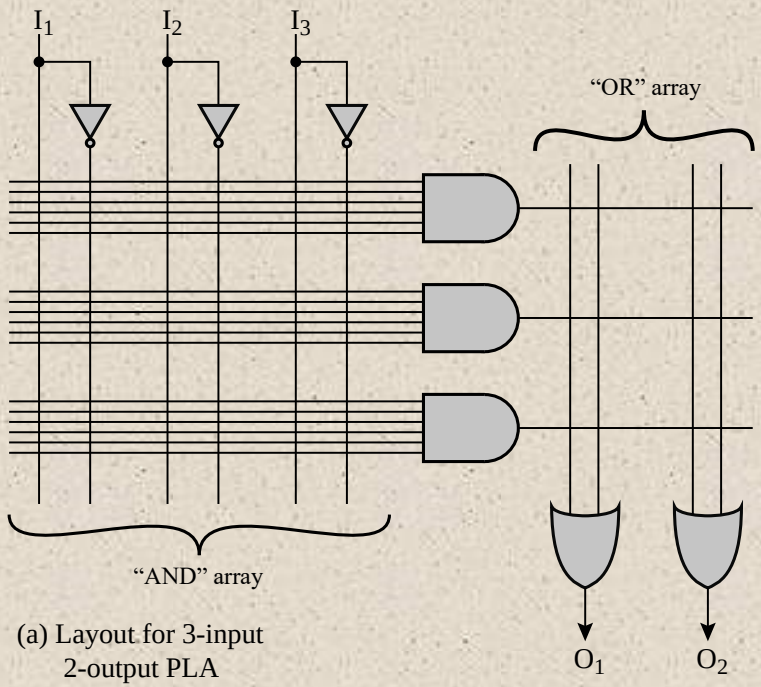
Logic Block

A relatively small circuit block that is replicated in an array in an FPD. When a circuit is implemented in an FPD, it is first decomposed into smaller sub-circuits that can each be mapped into a logic block. The term logic block is mostly used in the context of FPGAs, but it could also refer to a block of circuitry in a CPLD.

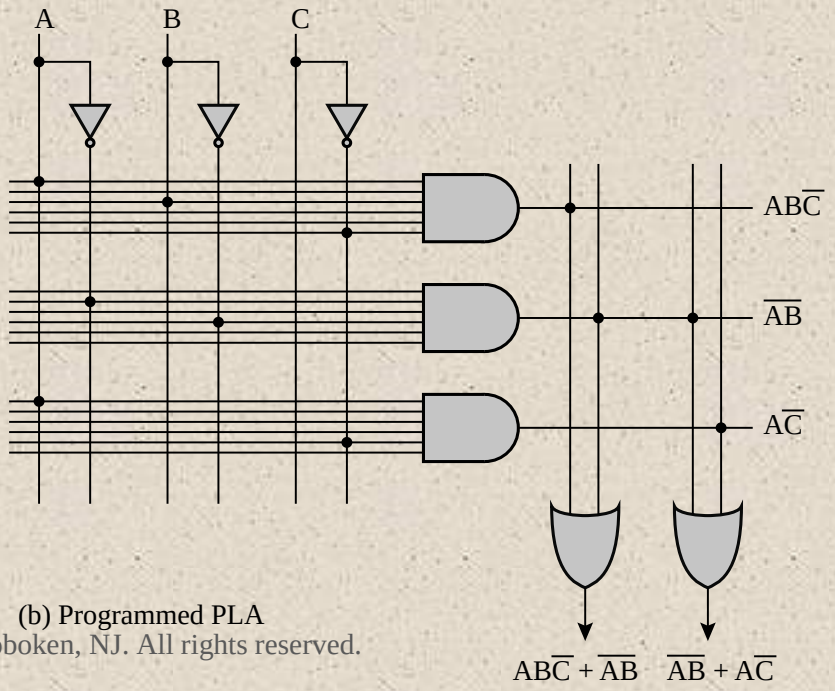
Table 11.11

PLD Terminology

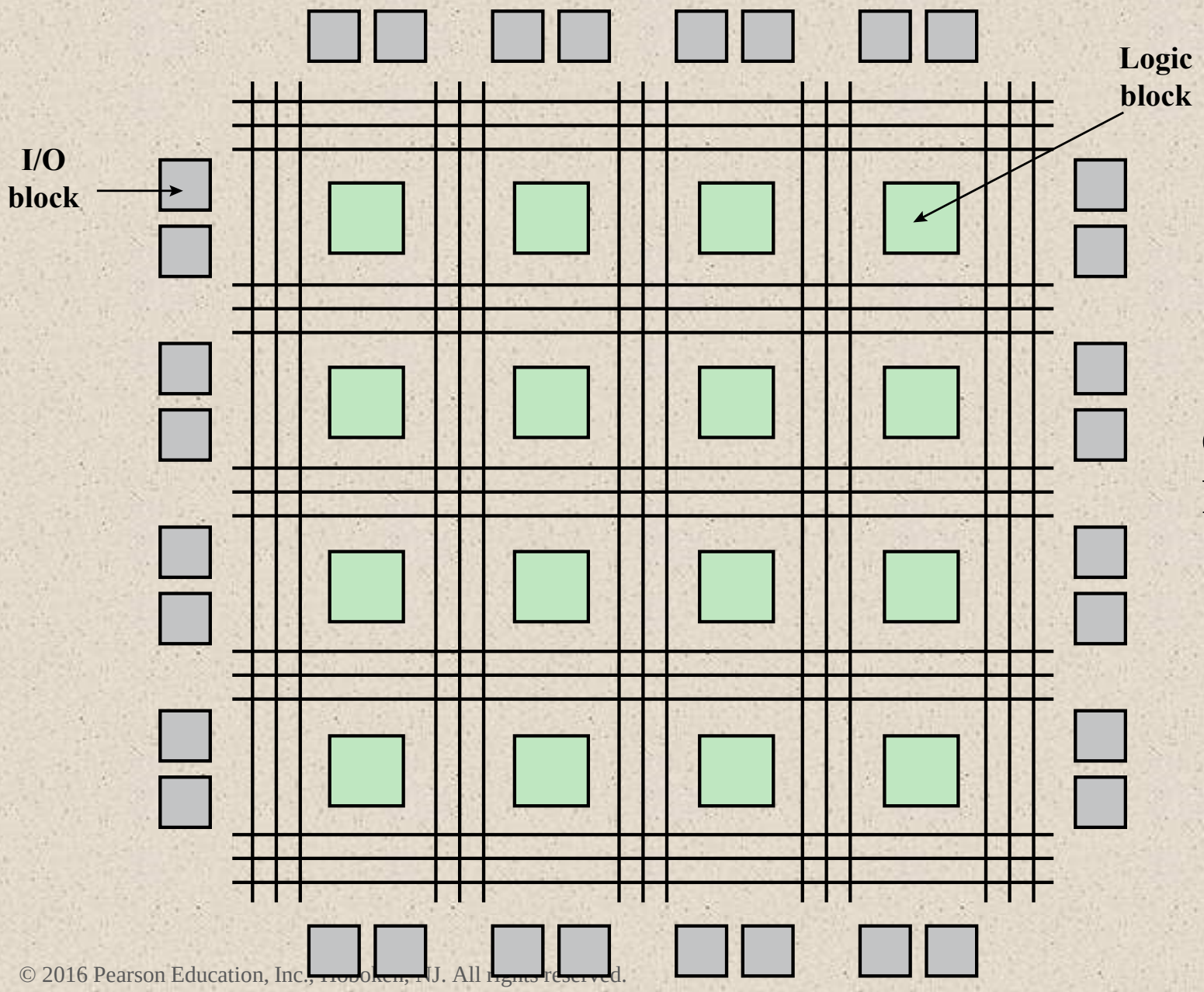
Programmable Logic Array



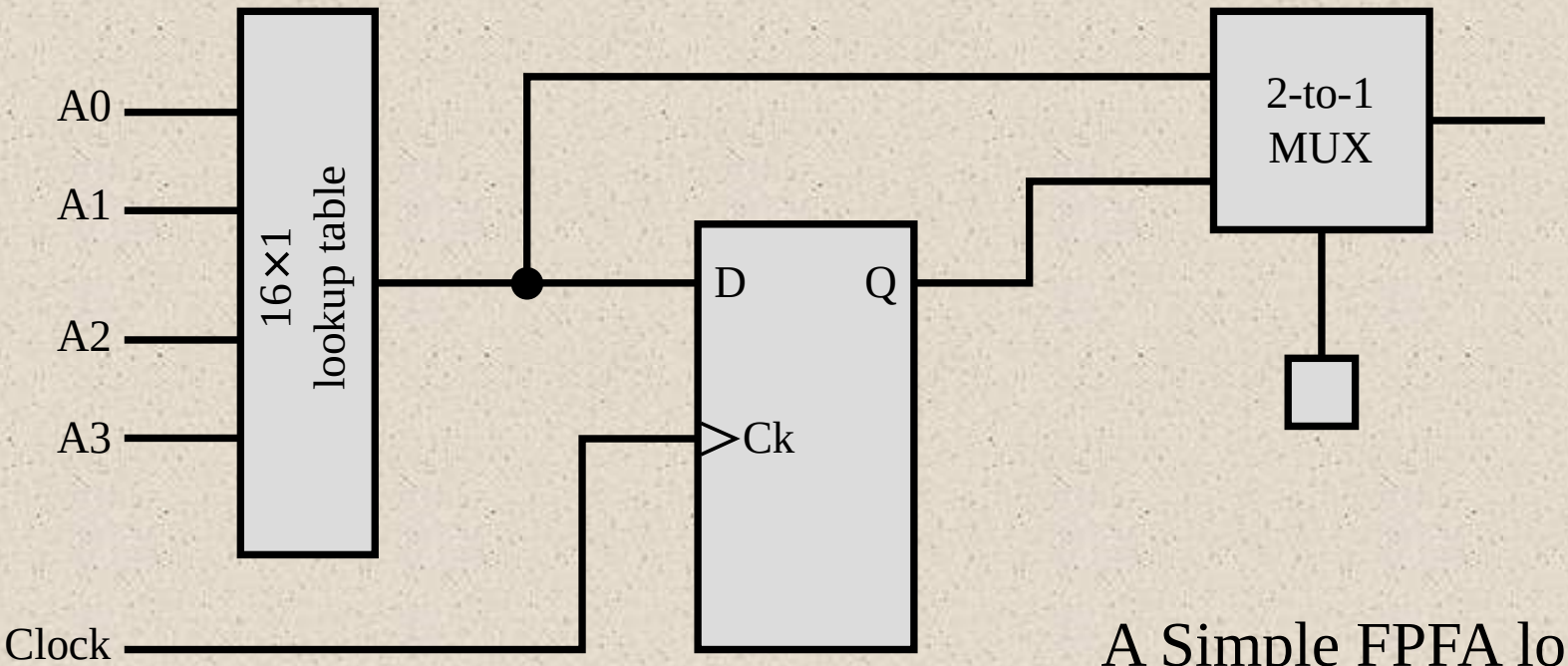
(a) Layout for 3-input 2-output PLA



(b) Programmed PLA



Structure of an FPGA



A Simple FPFA logic block

Summary

Chapter 11

- Boolean Algebra
- Gates
- Combinational Circuits
  - Implementation of Boolean Functions
  - Multiplexers
  - Decoders
  - Read-Only-Memory
  - Adders
- Sequential Circuits
  - Flip-Flops
  - Registers
  - Counters
- Programmable Logic Devices
  - Programmable Logic Array
  - Field-Programmable Gate Array