Read-only Memory

Sparsh Mittal

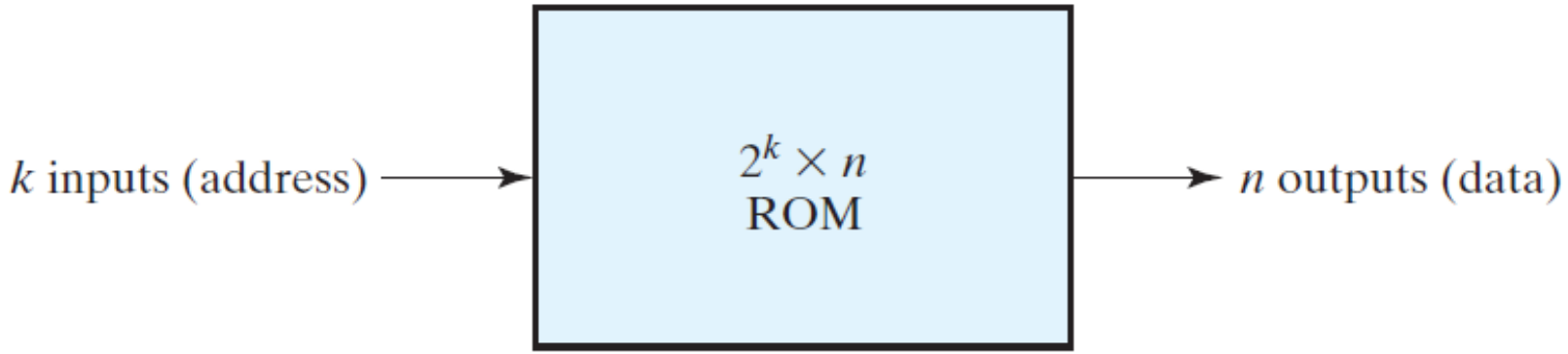
ROM stores permanent binary information.

The binary information must be specified by the designer and is then embedded in the unit to form the required interconnection pattern.

Once the pattern is established, it stays within the unit even when power is turned off and on again.

ROM does not have data inputs, because it does not have a write operation.

Block diagram of a ROM consisting of k inputs and n outputs.



Example: a 32 * 8 ROM

It has 32 words of 8 bits each.

There are five input lines to form numbers 0 to 31.

These lines are decoded into 32 distinct outputs using a 5 * 32 decoder.

The 32 outputs of the decoder are connected to each of the eight OR gates.

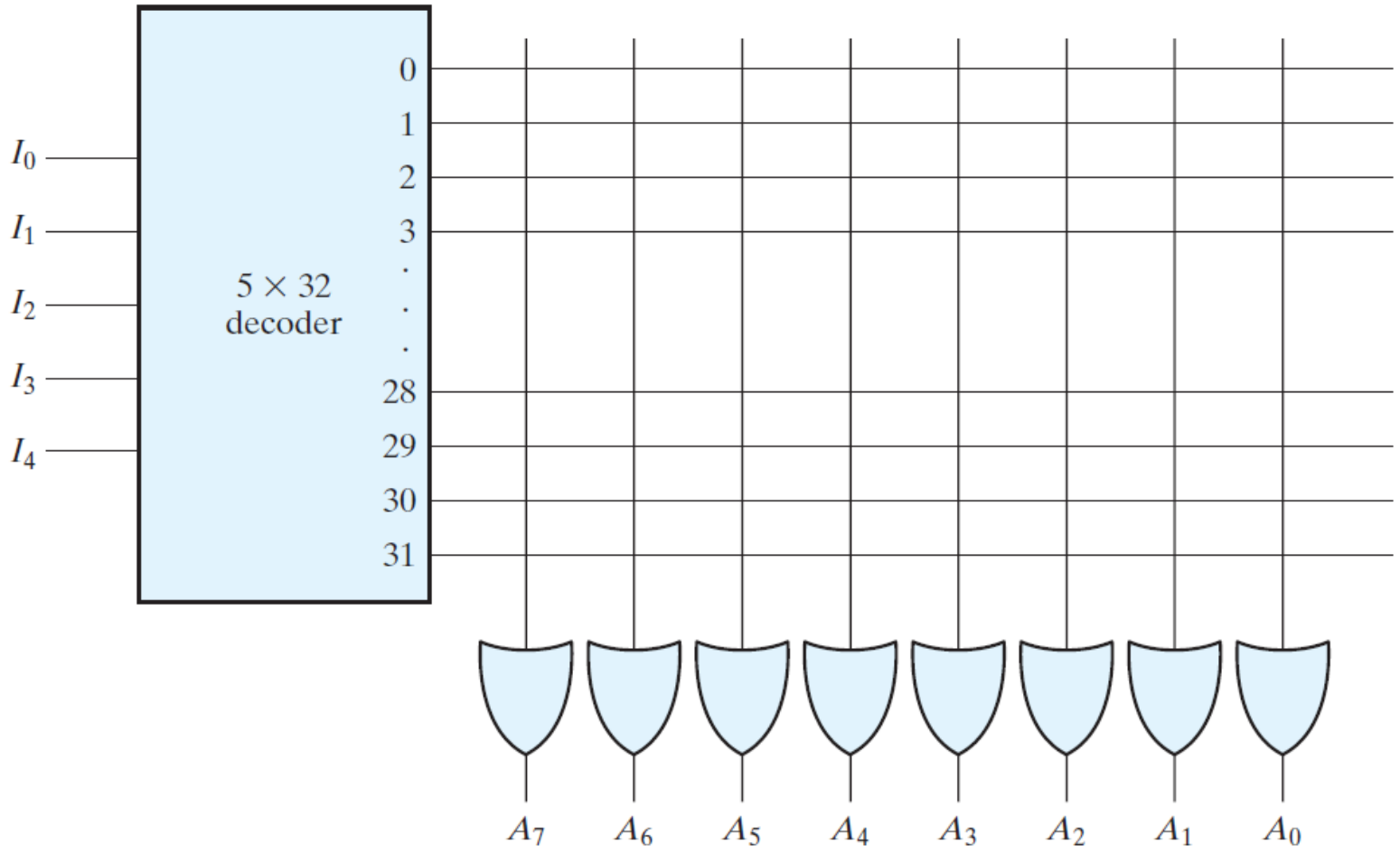


FIGURE 7.10
Internal logic of a 32 × 8 ROM

Example (continued)

Each OR gate must be considered as having 32 inputs.

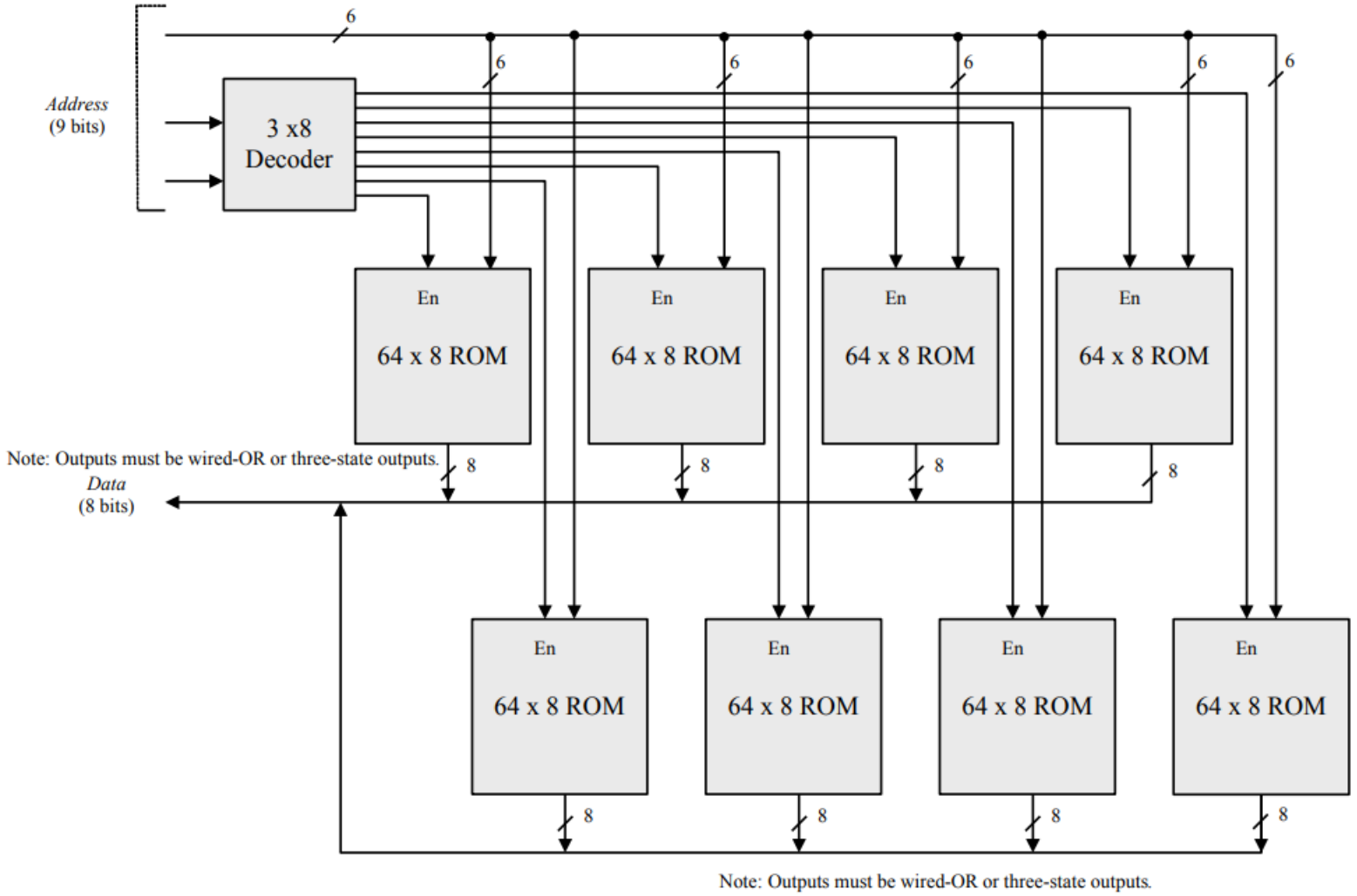
Each decoder output is connected to one of the inputs of each OR gate.

Since each OR gate has 32 input connections and there are 8 OR gates, the ROM contains 32 * 8 = 256 internal connections.

Problem – 7.15

Using 64 × 8 ROM chips with an enable input, construct a 512 × 8 ROM with eight chips and a decoder.

Solution – 7.15

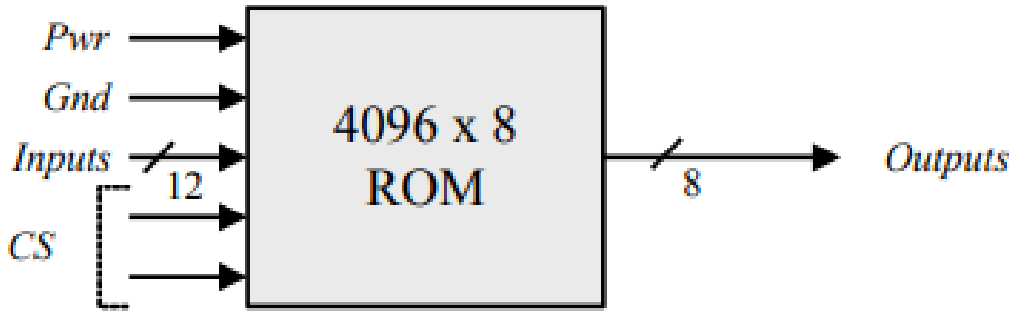


Problem – 7.16

A ROM chip of 4,096 × 8 bits has two chip select inputs and operates from a 5-V power supply. How many pins are needed for the integrated circuit package? Draw a block diagram, and label all input and output terminals in the ROM.

Solution – 7.16

Note: $4096 = 2^{12}$



16 inputs + 8 outputs requires a 24-pin IC.

Programmable connections

- The 256 intersections are programmable.
- A programmable connection (also called crosspoint) between two lines is logically equivalent to a switch that can be altered to be either
  - closed (means that the two lines are connected)
  - or open (meaning that the two lines are disconnected).
- Various physical devices are used to implement crosspoint switches.
- Simplest: use a fuse that normally connects the two points, but is opened or “blown” by applying a high-voltage pulse into the fuse.

Truth table for a ROM

The binary storage of a ROM is specified by a truth table that shows the word content in each address. An example truth table is here.

Table 7.3
ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮							⋮			
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Truth table

It shows the five inputs under which are listed all 32 addresses.

Each address stores a word of 8 bits, which is listed in the outputs columns.

This table shows only first four and last four words in the ROM.

To program ROM, we will blow fuse links accordingly.

For example, the table specifies the eight-bit word 10110010 for permanent storage at address 3.

The four 0's in the word are programmed by blowing the fuse links between output 3 of the decoder and the inputs of the OR gates associated with outputs *A*6, *A*3, *A*2, and *A*0.

X shows temporary connection

● shows permanent connection

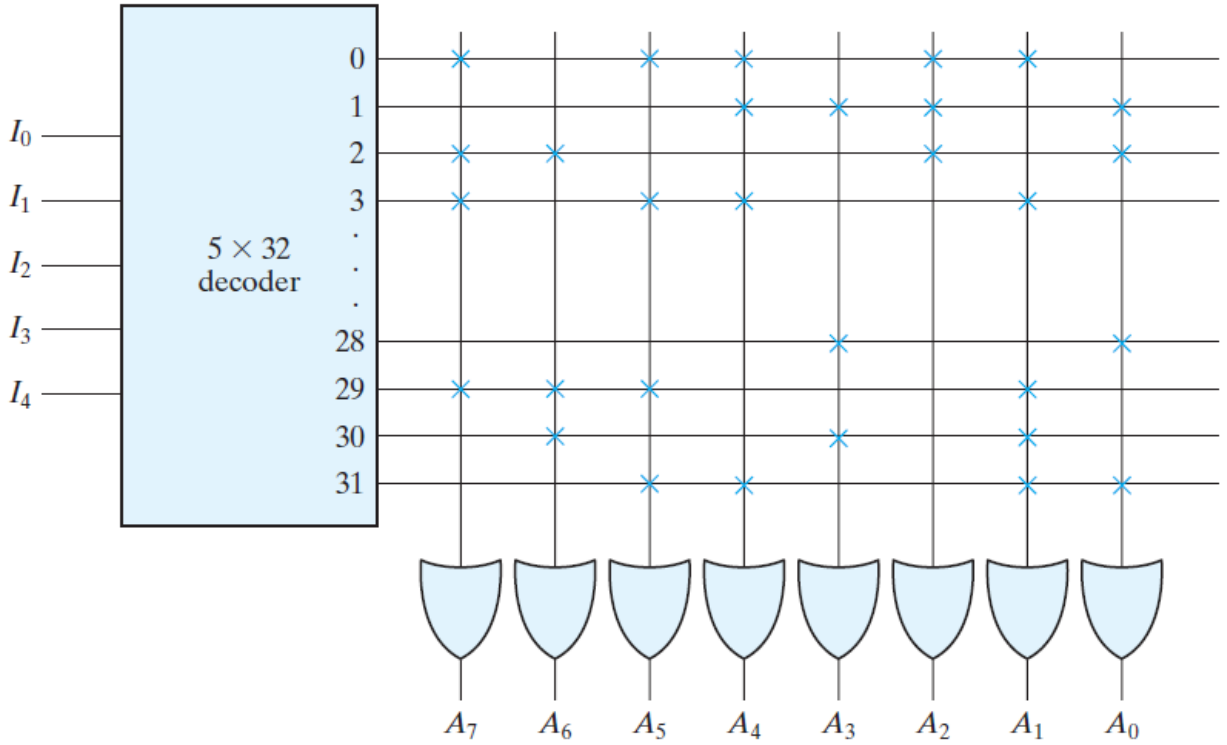


FIGURE 7.11
Programming the ROM according to Table 7.3

When the input of the ROM is 00011, all the outputs of the decoder are 0 except for output 3, which is at logic 1. The signal equivalent to logic 1 at decoder output 3 propagates through the connections to the OR gate outputs of A_7, A_5, A_4, and A_1.

The other four outputs remain at 0. The result is that the stored word 10110010 is applied to the eight data outputs.

Combinational Circuit Implementation

A decoder generates the 2^k minterms of the k input variables.

By inserting OR gates to sum the minterms of Boolean functions, we were able to generate any desired combinational circuit.

ROM includes both the decoder and the OR gates within a single device to form a minterm generator.

By choosing connections for those minterms which are included in the function, the ROM outputs can be programmed to represent the Boolean functions of the output variables in a combinational circuit.

Example: Design a combinational circuit using a ROM.

The circuit accepts a three-bit number and outputs a binary number equal to the square of the input number.

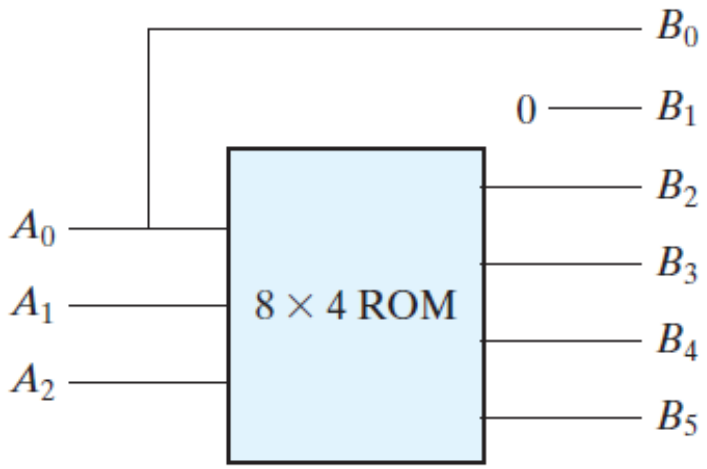
Steps: Derive the truth table of the combinational circuit

Table 7.4
Truth Table for Circuit of Example 7.1

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

We actually need to generate only four outputs with the ROM; the other two are readily obtained.

The minimum size of ROM needed must have three inputs and four outputs. Three inputs specify eight words, so the ROM must be of size 8 * 4.



(a) Block diagram

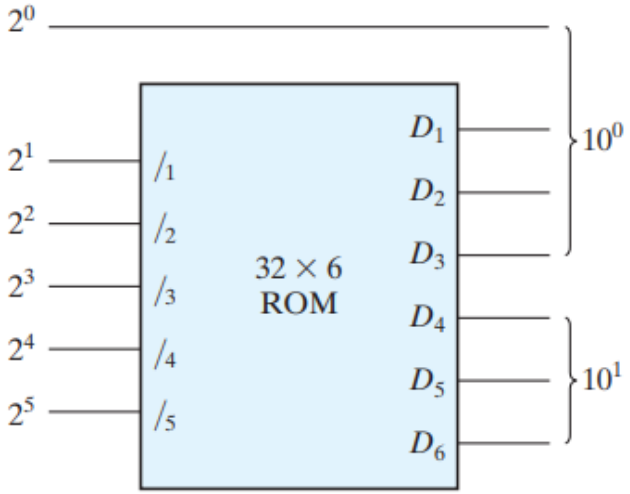
A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

FIGURE 7.12
ROM implementation of Example 7.1

Problem – 7.17

The 32 × 6 ROM, together with the 20 line, as shown in the figure, converts a six-bit binary number to its corresponding two-digit BCD number. For example, binary 100001 converts to BCD 011 0011 (decimal 33). Specify the truth table for the ROM.



Solution – 7.17

Input Address	Output of ROM			
$I_5 I_4 I_3 I_2 I_1$	$D_6 D_5 D_4$	$D_3 D_2 D_1$	D_0 (2^0)	Decimal
0 0 0 0 0	0 0 0	0 0 0	0, 1	0, 1
0 0 0 0 1	0 0 0	0 0 1	0, 1	2, 3
…	…	…	…	…
…	…	…	…	…
0 1 0 0 0	0 0 1	0 1 1	0, 1	16, 17
0 1 0 0 1	0 0 1	1 0 0	0, 1	18, 19
…	…	…	…	…
…	…	…	…	…
1 1 1 1 0	1 1 0	0 0 0	0, 1	60, 61
1 1 1 1 1	1 1 0	0 0 1	0, 1	62, 63

Problem – 7.18

Specify the size of a ROM (number of words and number of bits per word) that will accommodate the truth table for the following combinational circuit components:

(a) a binary multiplier that multiplies two 4-bit binary words,

(b) a 4-bit adder–subtractor,

(c) a quadruple two-to-one-line multiplexer with common select and enable inputs, and

(d) a BCD-to-seven-segment decoder with an enable input.

Solution – 7.18

(a)	8 inputs	8 outputs	$2^8 \times 8$	256×8 ROM
(b)	9 inputs	5 outputs	$2^9 \times 5$	512×5 ROM
(c)	10 inputs	4 outputs	$2^{10} \times 4$	1024×4 ROM
(d)	5 inputs	7 outputs	$2^5 \times 7$	32×7 ROM

Problem – 7.20

Tabulate the truth table for an 8 × 4 ROM that implements the Boolean functions

$$A(x, y, z) = (0, 3, 4, 6)$$
$$B(x, y, z) = (0, 1, 4, 7)$$
$$C(x, y, z) = (1, 5)$$
$$D(x, y, z) = (0, 1, 3, 5, 7)$$

Considering now the ROM as a memory. Specify the memory contents at addresses 1 and 4.

Solution – 7.20

Inputs	Outputs
x y z	*A, B, C, D*
0 0 0	1 1 0 1
0 0 1	0 1 1 1
0 1 0	0 0 0 0
0 1 1	1 0 0 1
1 0 0	1 1 0 0
1 0 1	0 0 1 1
1 1 0	1 0 0 0
1 1 1	0 1 0 1

$M[001] = 0111$

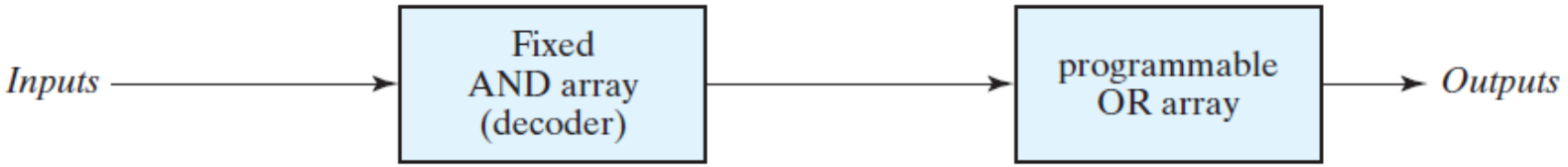
$M[100] = 1100$

PLDs

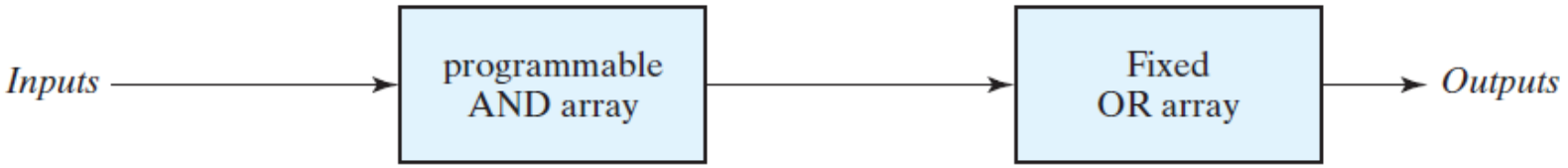
The programmable ROM (PROM) is a combinational programmable logic device (PLD)

A PLD is an IC with programmable gates divided into an AND array and an OR array to provide an AND–OR sum-of-product implementation.

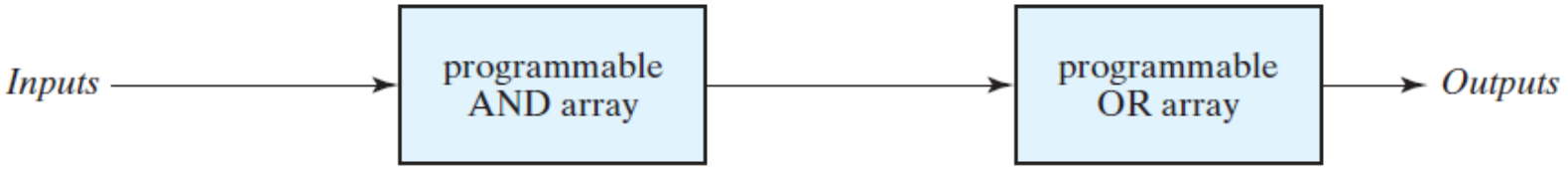
There are three major types of combinational PLDs, differing in the placement of the programmable connections in the AND–OR array.



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

FIGURE 7.13
Basic configuration of three PLDs

3 types of PLDs

1. The PROM has a fixed AND array constructed as a decoder and a programmable OR array.

The programmable OR gates implement the Boolean functions in sum-of-minterms form.

2. The PAL has a programmable AND array and a fixed OR array.

The AND gates are programmed to provide the product terms for the Boolean functions, which are logically summed in each OR gate.

3. The most flexible PLD is the PLA, in which both the AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum-of-products implementation.