

# Digital System Design using Verilog

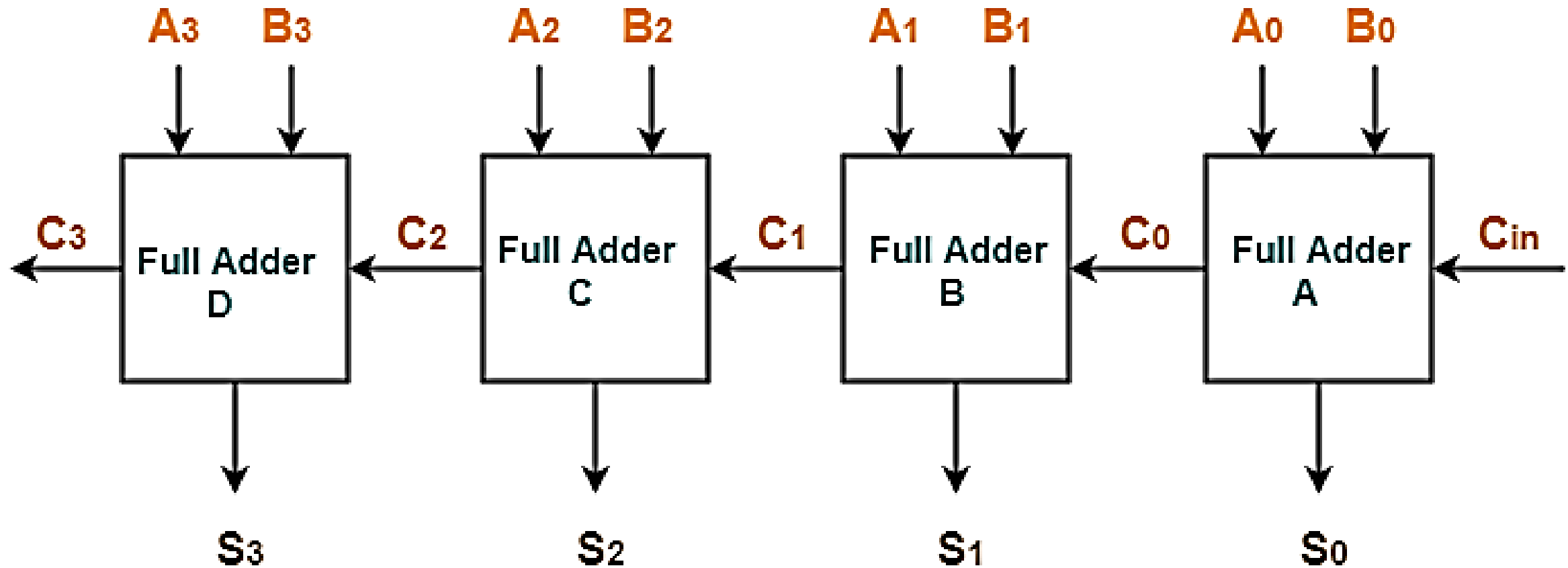
## 1BEC302

### CREDITS: 4

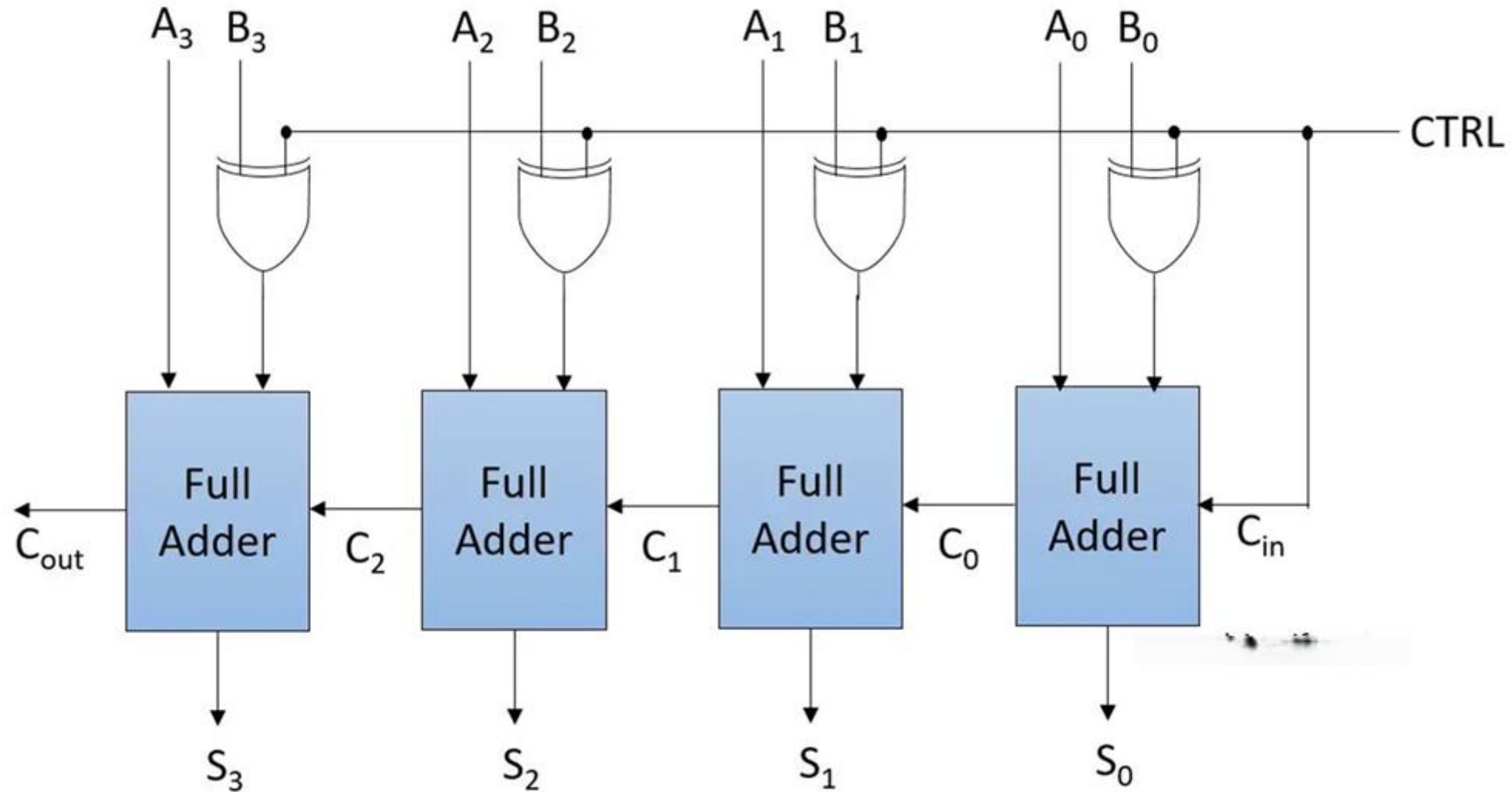
**Pradeep Kumar Y**  
Assistant Professor  
Department of ECE  
ATME College of Engineering  
Mysuru-570028

## **Module 2: Logic Design with MSI Components & PLDs**

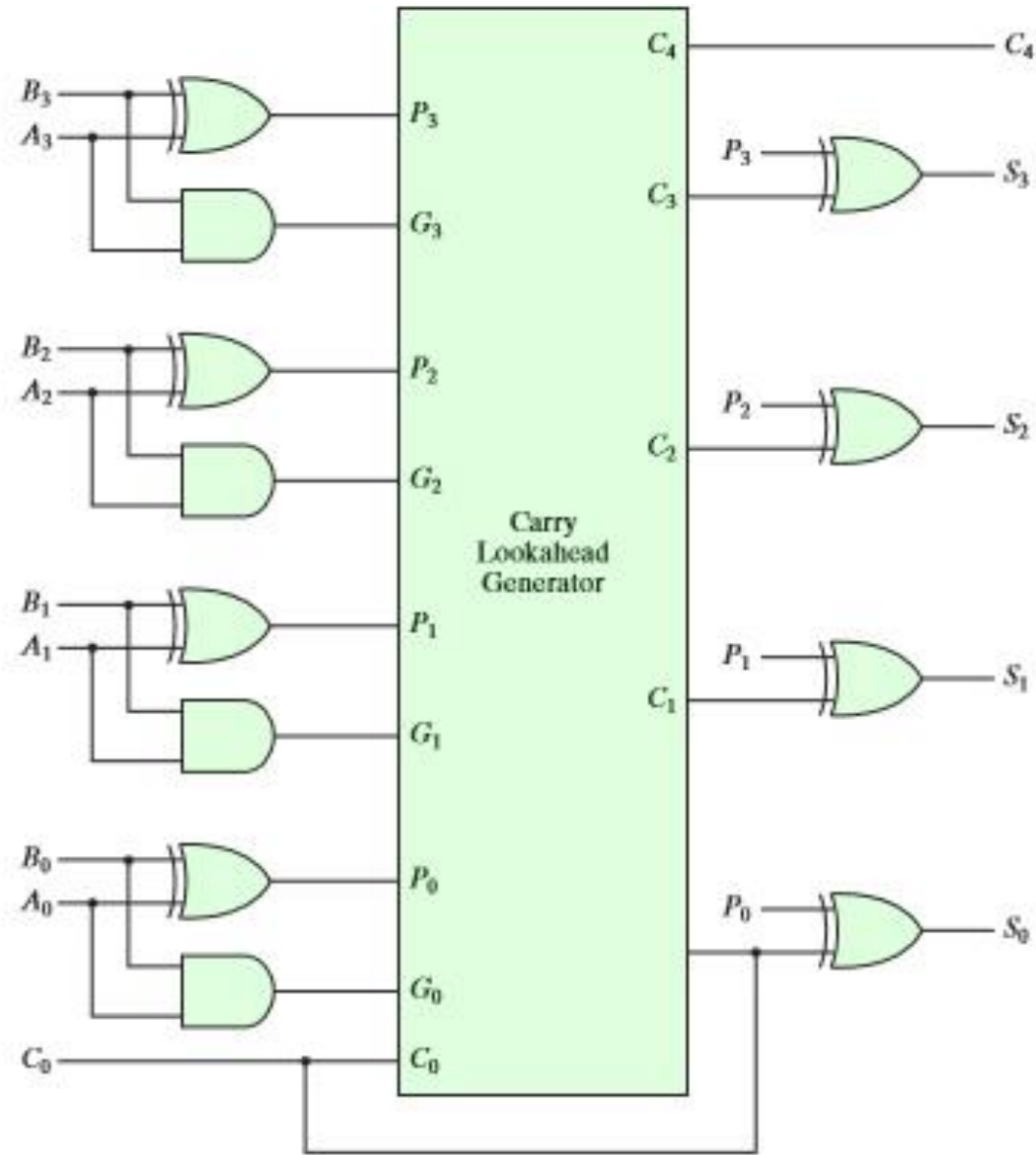
1. Binary Adders and Subtractors
2. Comparators
3. Decoders and Encoders
4. Multiplexers
5. Introduction to Programmable Logic Devices (PLDs)



## 4-bit Ripple Carry Adder



## 4-Bit Adder Subtractor



## Programmable Logic Devices (PLDs):

- ❑ PLDs are **digital devices** used to implement **combinational and sequential logic circuits**, where the logic function can be **programmed** by the user.
- ❑ Unlike fixed-function logic ICs, PLDs can be **configured to perform a variety of logic operations** after manufacturing.
- ❑ **Types of PLDs:**
  1. **PROM (Programmable Read-Only Memory)** – Used for implementing simple combinational logic.
  2. **PAL (Programmable Array Logic)** – Has a **programmable AND** array and a **fixed OR** array.
  3. **PLA (Programmable Logic Array)** – Both **AND and OR arrays** are programmable.
  4. **CPLD (Complex PLD)** – Contains multiple PAL/PLA blocks on a single chip.
  5. **FPGA (Field Programmable Gate Array)** – Most advanced PLD with configurable logic blocks and interconnections

## Programmable Logic Devices (PLDs):

❑ Programming is done using technologies like **fuse links, EEPROM, or SRAM.**

❑ **Advantages:**

Reduces circuit size and complexity

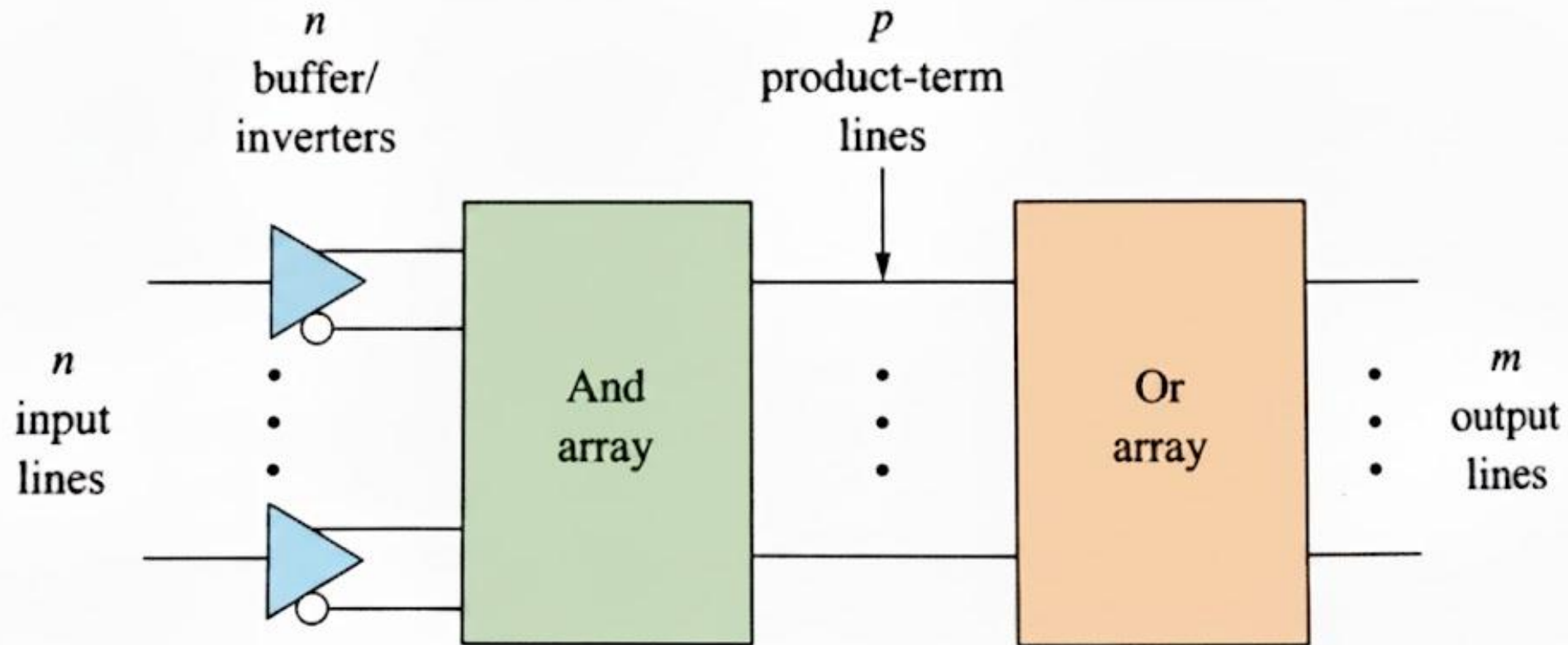
Increases design flexibility and reusability

Faster prototyping and easier design changes

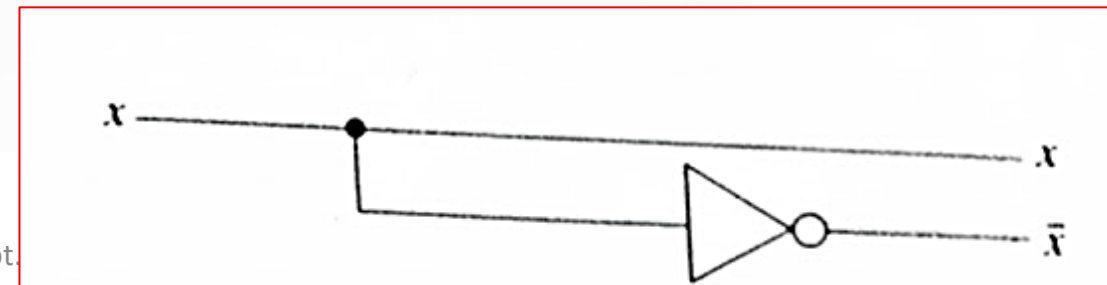
❑ **Applications:**

Used in **digital system design, signal processing, embedded systems, and communication systems.**

# Programmable Logic Devices (PLDs):



General structure of PLDs.



# Programmable Logic Devices (PLDs):

## Input Stage:

- Inputs to the PLD are applied through **buffer/inverter circuits**.
- Each input provides both **true** and **complemented** forms.
- Buffers also provide **sufficient drive capability** for large fan-out to multiple gates.

## AND-Array:

- Accepts **n input variables** and their complements.
- Generates **p product terms (minterms)**.
- Acts as the foundation for forming logical combinations.

## OR-Array:

- Combines the **product terms** to form **m sum-of-products (SOP) expressions**.
- Produces the **final logic outputs** of the PLD.

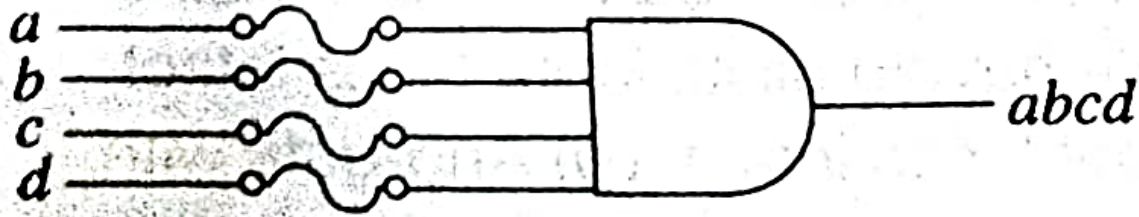
## Programmability:

- **One or both arrays (AND/OR)** can be made programmable.
- Enables designers to **customize logic connections** for specific Boolean functions

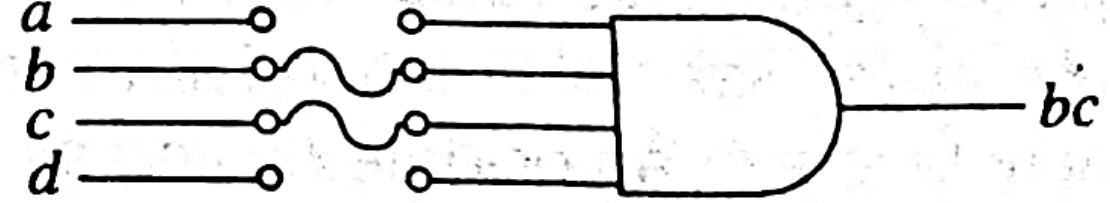
## Programmable Logic Devices (PLDs):

| Device                                         | And-array    | Or-array     |
|------------------------------------------------|--------------|--------------|
| <b>PROM</b><br>(Programmable Read-Only Memory) | Fixed        | Programmable |
| <b>PLA</b><br>(Programmable Logic Array)       | Programmable | Programmable |
| <b>PAL</b><br>(Programmable Array Logic)       | Programmable | Fixed        |

## Programmable Logic Devices (PLDs):



(a)



(b)

# Programmable Logic Devices (PLDs):

## Programmable Arrays in PLDs

### •Concept:

- In a programmable array, **connections to each gate can be modified.**
- Each input of a gate is connected through a **fuse.**

### •Fuse Programming Example:

- Example: To generate product term **abcd**, all fuses are intact.
- To generate **bc**, fuses for inputs **a** and **d** are **blown (removed).**

### •Programming Process:

- Programming is a **hardware operation** performed using **specialized equipment (programmers).**
- Determines which fuses remain connected or are blown.

# Programmable Logic Devices (PLDs):

## Types of Programmable Devices

### 1. Fuse-Programmable Devices:

**One-time programmable** — fuses are permanently blown.

Used where the logic design is final and fixed.

### 2. Erasable Programmable Devices (EPLDs):

Can be **erased and reprogrammed**.

Erasing done using **ultraviolet (UV) light** or **electrical signals**.

Enables design modifications and reuse.

# Programmable Logic Devices (PLDs):

## Types of Programmable Devices

### 3. Field Programmable Devices (FPDs):

**User-programmable** directly using a programmer.

Used widely for prototyping and custom logic development.

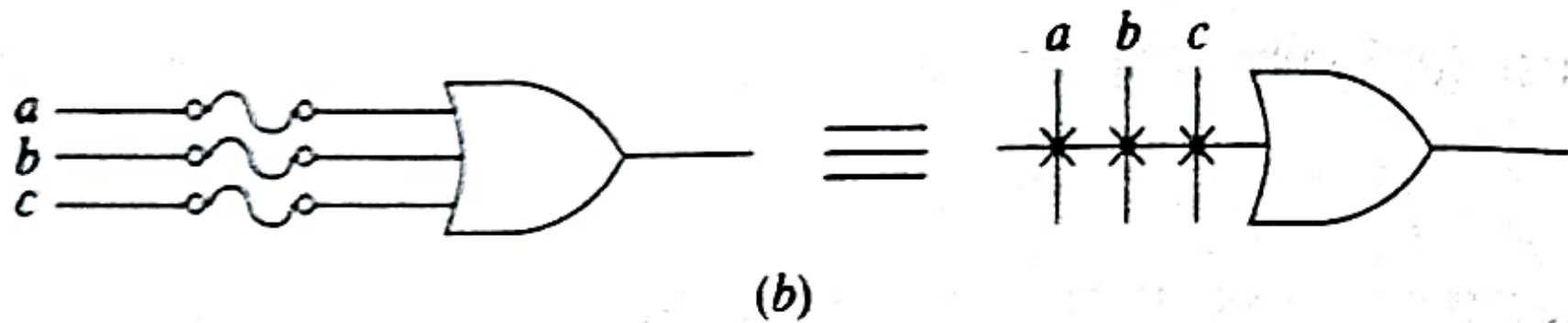
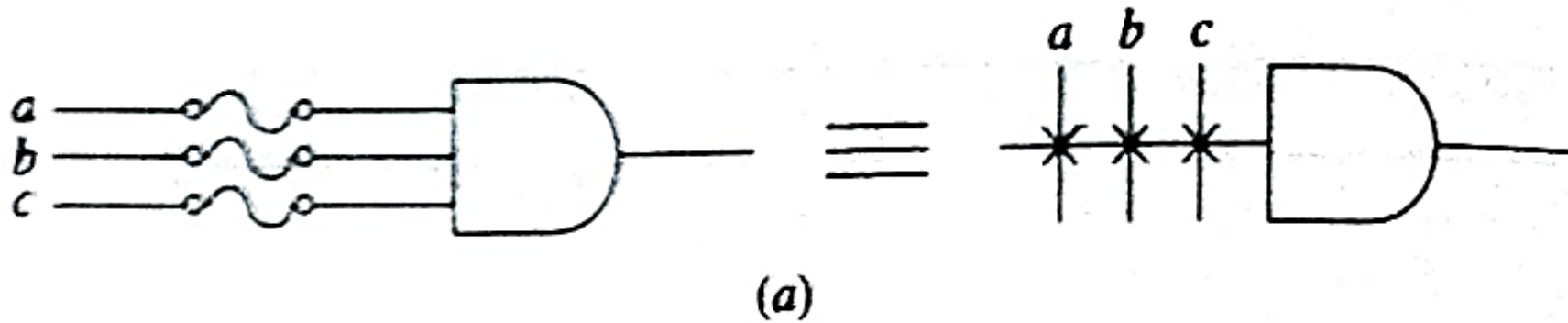
### 4. Mask Programmable Devices:

User provides connection data to the **manufacturer**.

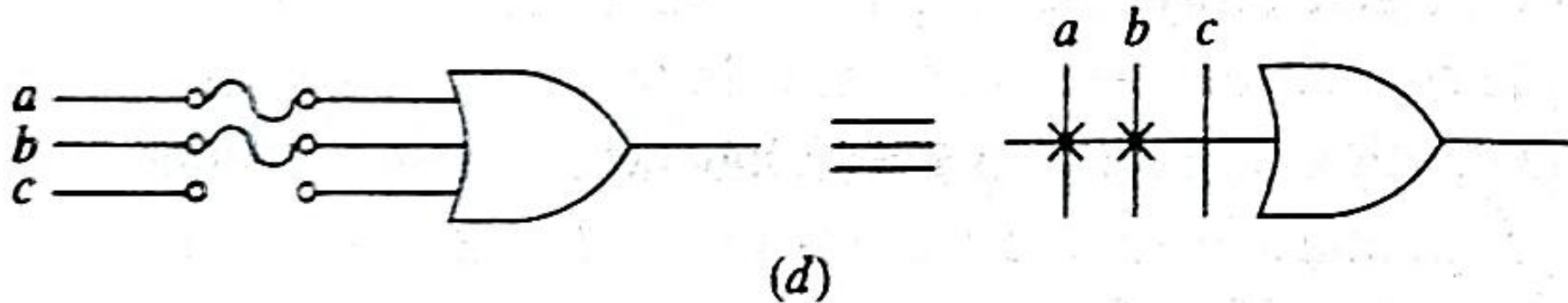
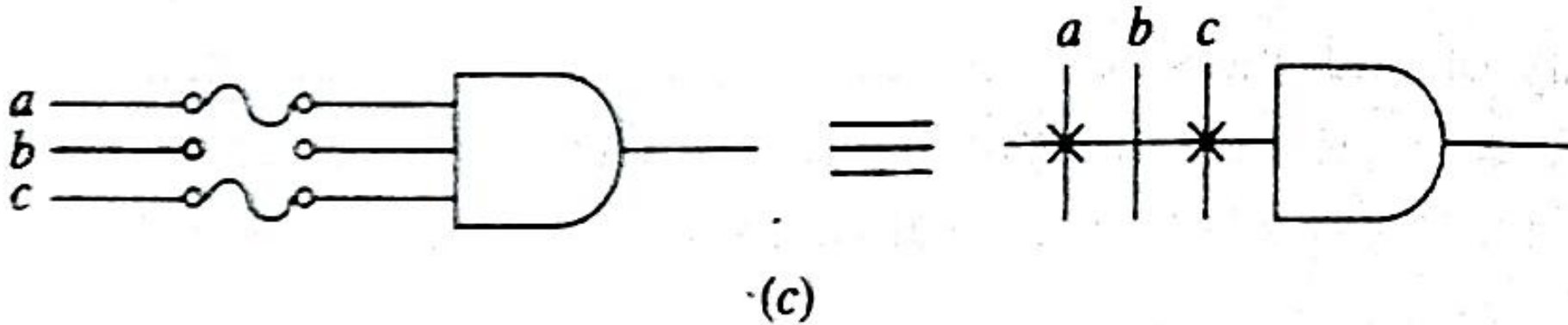
Final connections are made using a **mask layer** during fabrication.

Suitable for **mass production** of a fixed design.

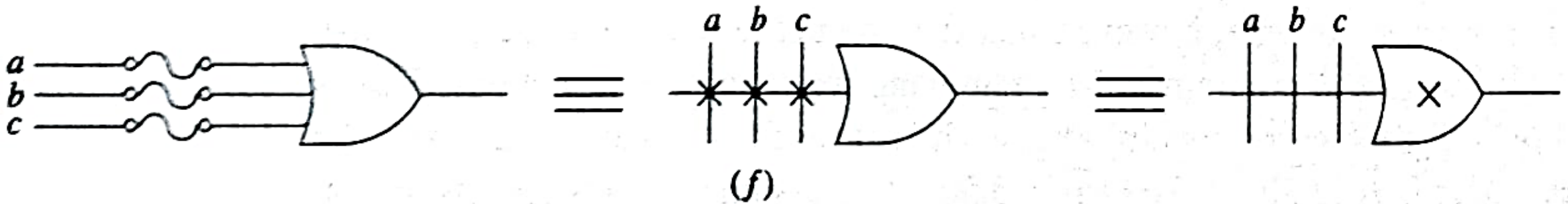
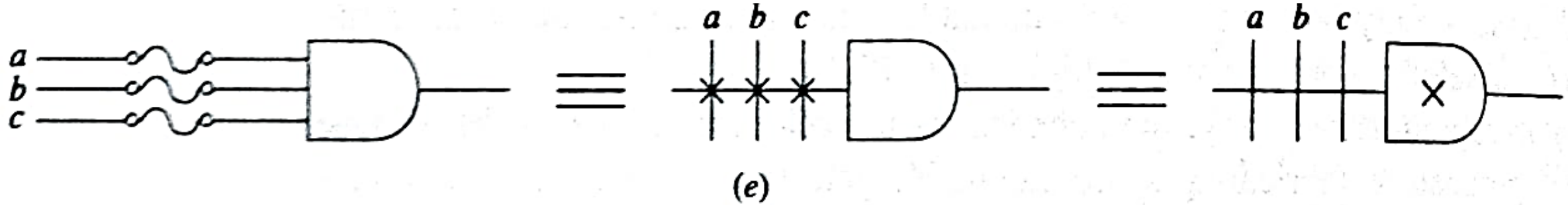
# Programmable Logic Devices (PLDs):



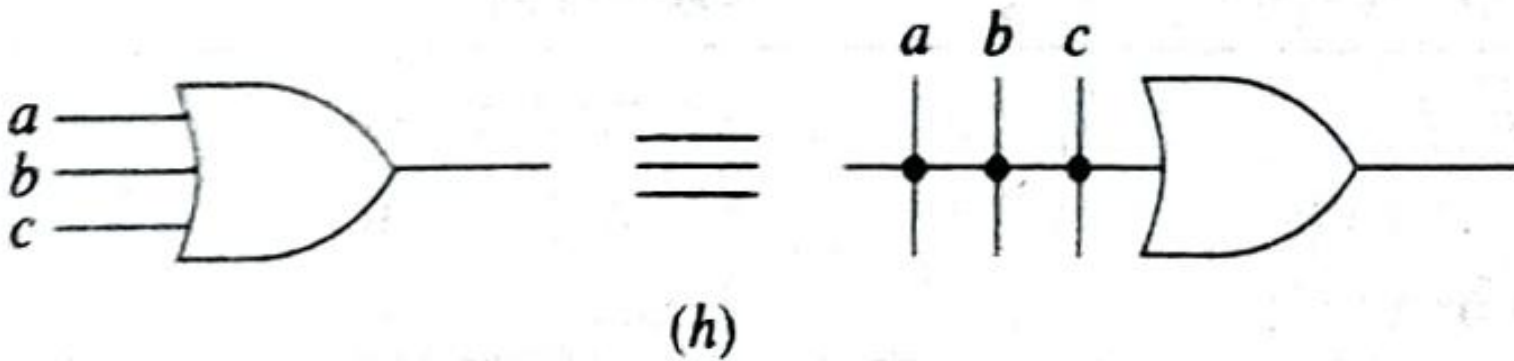
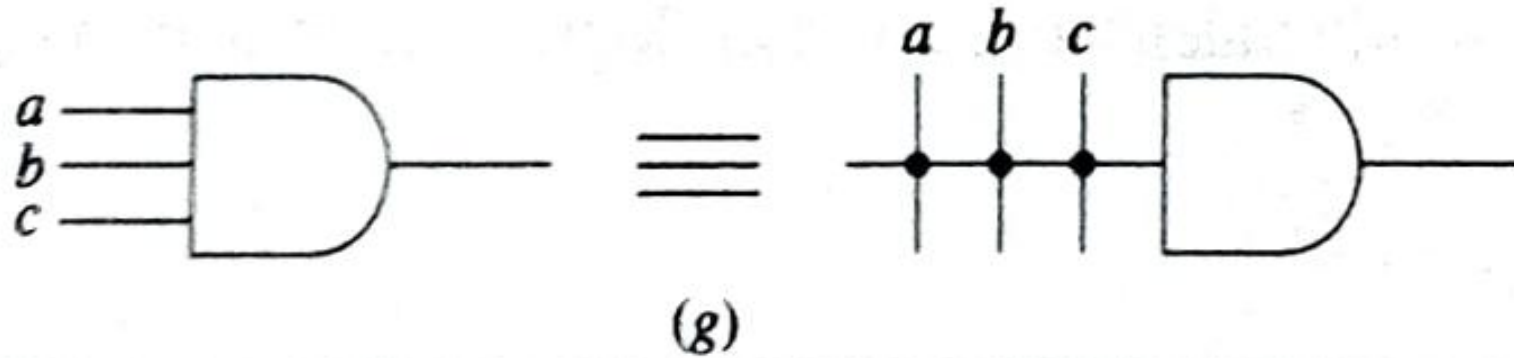
# Programmable Logic Devices (PLDs):



# Programmable Logic Devices (PLDs):

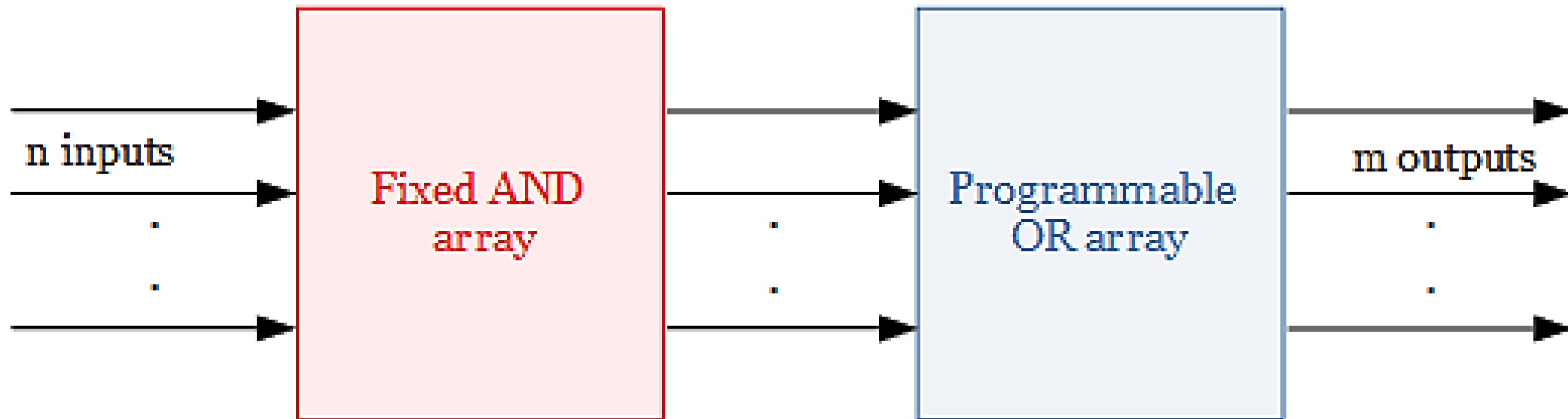


# Programmable Logic Devices (PLDs):



# Programmable Read-Only Memories (PROMs)

- Consists of an **AND-array** (decoder) and a **programmable OR-array** (memory array).
- AND-array with buffer/inverters acts as an **n-to- $2^n$  line decoder** (minterm generator).
- OR-array collects outputs through **programmable fusible links**.



# Programmable Read-Only Memories (PROMs)

## •Operation:

- **Inputs (address lines):**  $n$
- **Outputs (bit lines):**  $m$
- PROM is characterized as a  **$2^n \times m$  PROM**.
- Each of the  $2^n$  decoder outputs connects to every OR-gate via fusible links.

## •Programming:

- **Fixed AND-array, programmable OR-array.**
- Logic functions are implemented by **selectively blowing fuses**.
- Fuse presence = logic connection; blown fuse = disconnection.

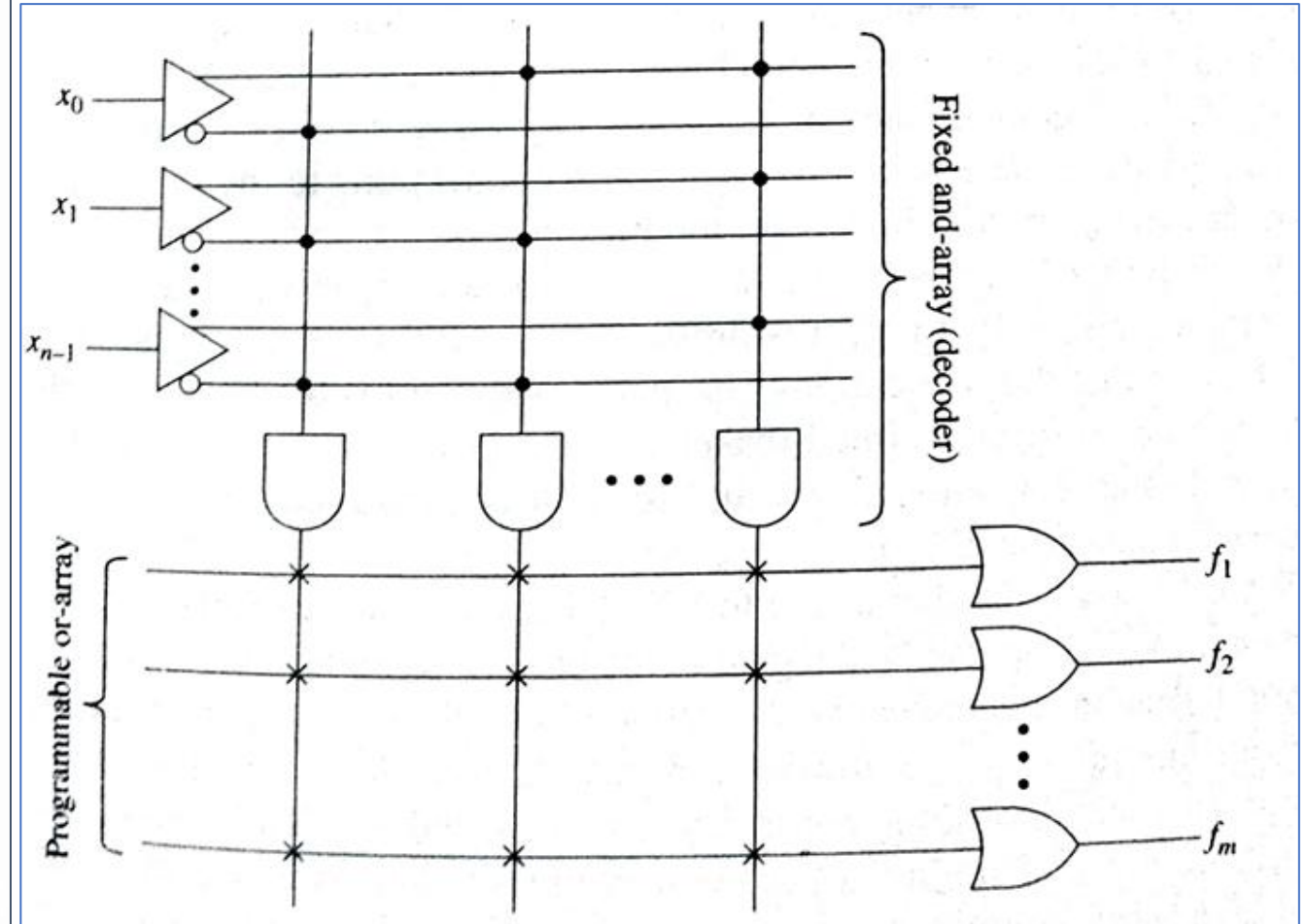
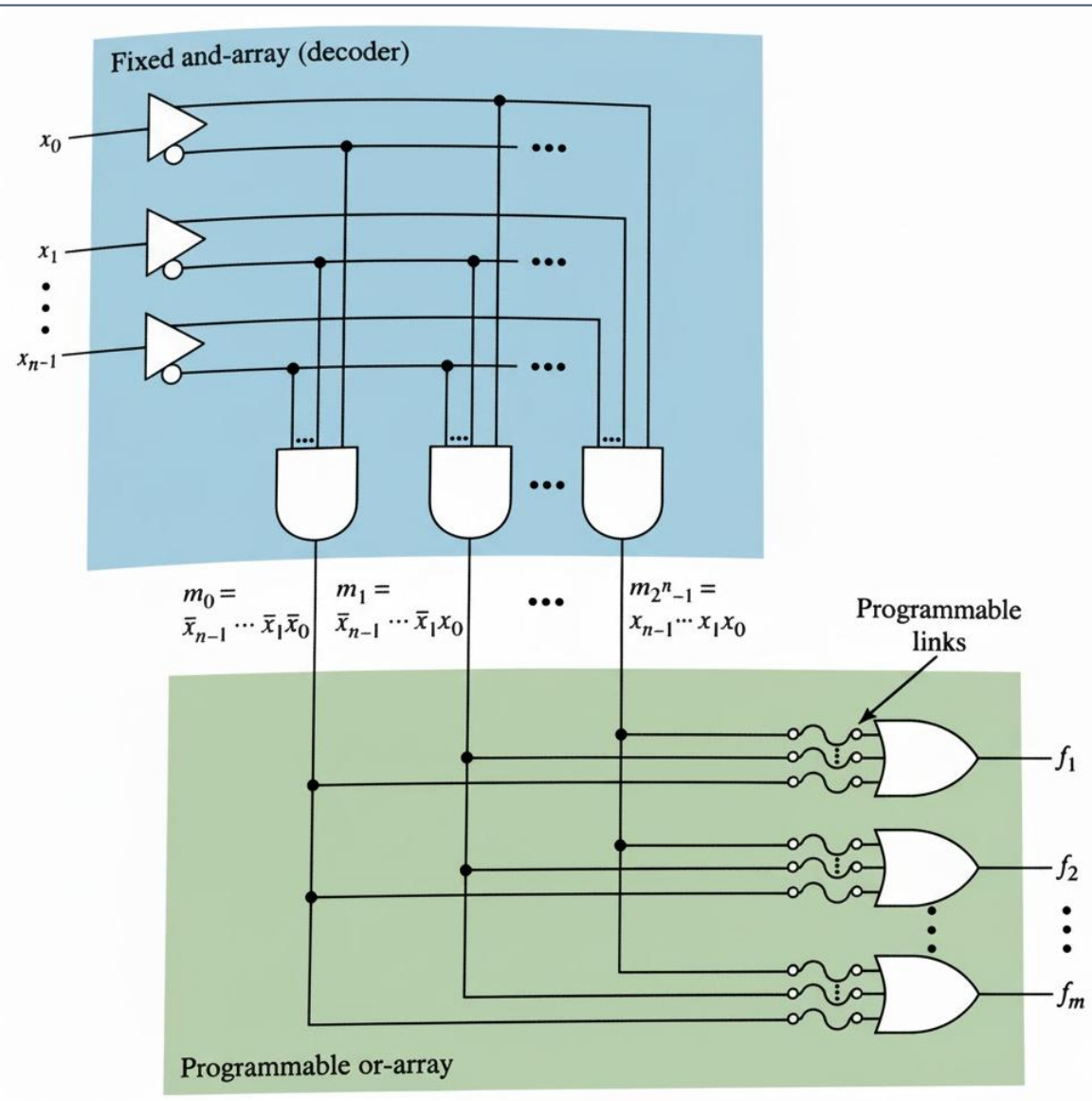
## •Realization:

- Boolean expressions or truth tables are directly mapped to PROM structure.
- The decoder produces all minterms; OR gates combine them as per required functions.

## •Applications:

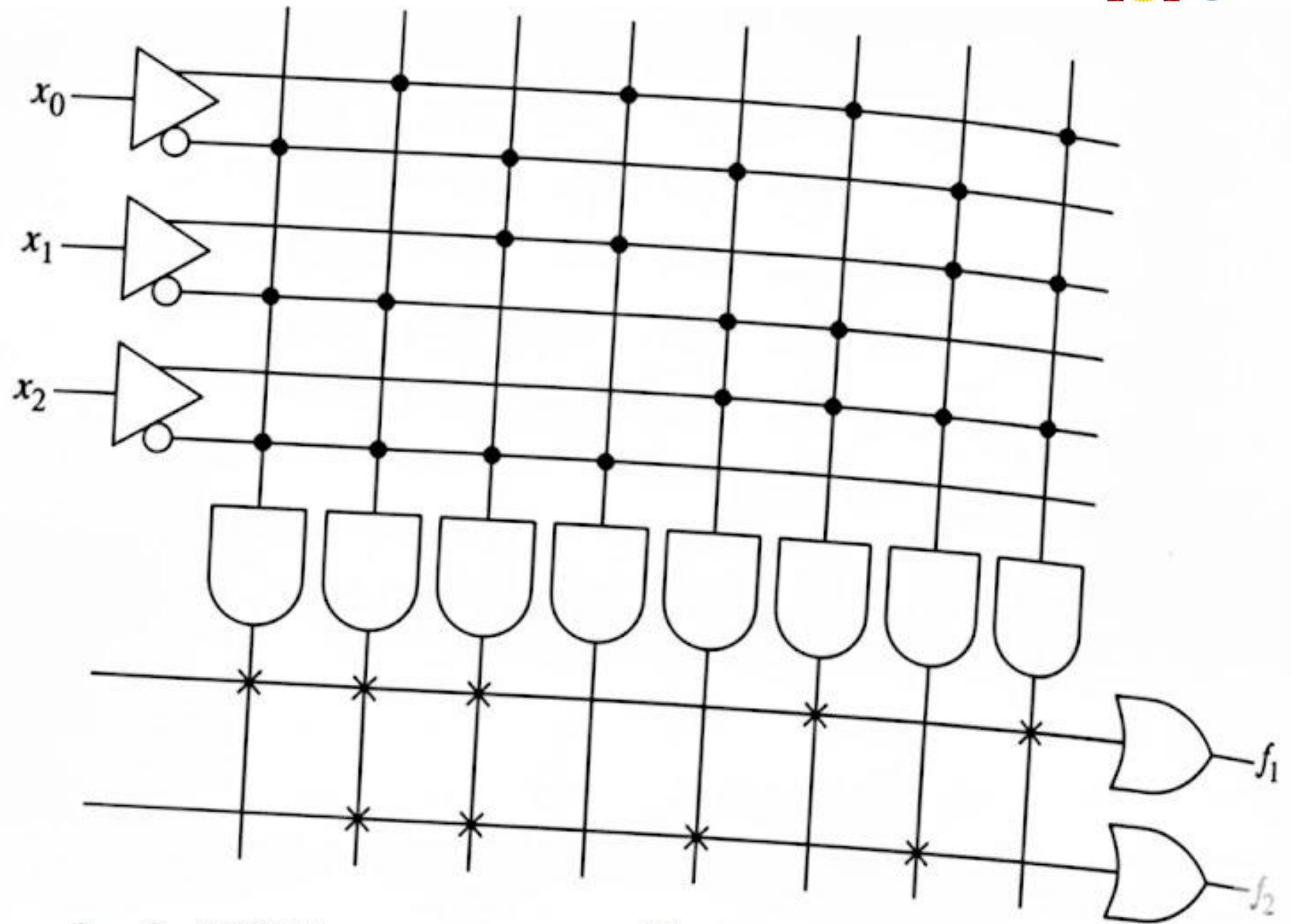
- Code conversion
- Character generation (bit pattern generation)
- <sup>9/6/2026</sup> Lookup tables for arithmetic or logic functions

# Programmable Read-Only Memories (PROMs)



| $x_2$ | $x_1$ | $x_0$ | $f_1 \ f_2$ |   |
|-------|-------|-------|-------------|---|
| 0     | 0     | 0     | 1           | 0 |
| 0     | 0     | 1     | 1           | 1 |
| 0     | 1     | 0     | 1           | 1 |
| 0     | 1     | 1     | 0           | 0 |
| 1     | 0     | 0     | 0           | 1 |
| 1     | 0     | 1     | 1           | 0 |
| 1     | 1     | 0     | 0           | 1 |
| 1     | 1     | 1     | 1           | 0 |

(a)



(b)

Figure 5.54 An example of a PROM