OF THE
DEPARTMENT OF
INFORMATION ENGINEERING

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Digital Hardware Implementation

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering

Academic Year 2023-2024

Purpose of the Lesson

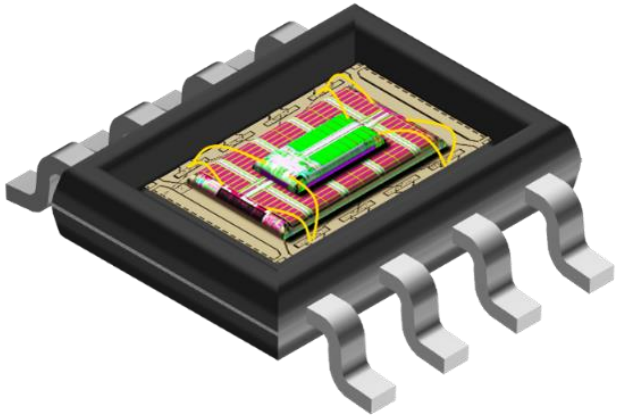
- Examine the different hardware implementations for digital circuits and the main technological parameters
- Overview of the solutions available for the implementation of programmable logic devices

Design Space

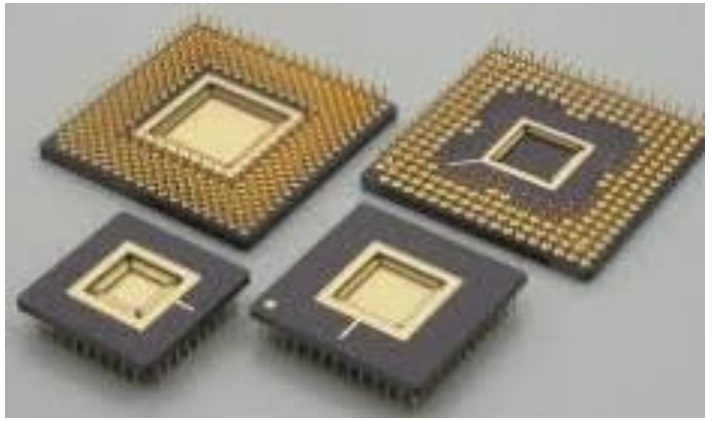
- The design space is the set of implementation technology (**primitive elements and the parameters characterizing them**) that have been chosen for a given system

Integrated Circuit

- An integrated circuit (IC) is a portion of silicon (chip) on which logic gates and memory elements are built, interconnected with each other inside the chip
  - The chip is then **mounted on a package** and special connections are welded from the chip to the external pins of the package (between a dozen and a few hundreds)
  - The characteristics of an IC are specified by the manufacturer in a document called **datasheet**



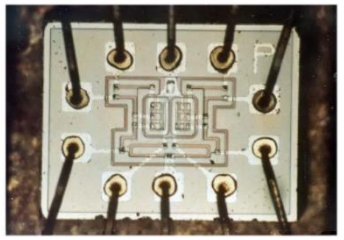
Source:www.embedded.com



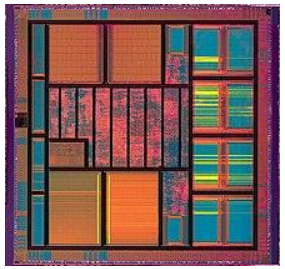
Source:www.ntktech.com

Integration Level

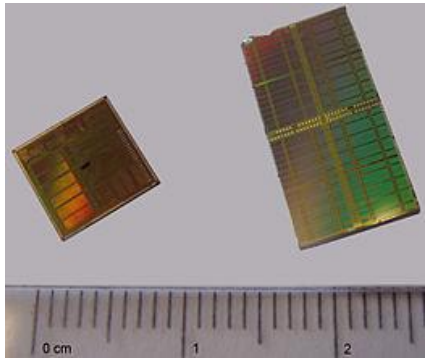
- With technology evolution, the level of IC integration has grown, with an **increasing number of logic gates on the same chip**
- Based on the number of logic gates, circuits are classified into
  - **Small-Scale Integrated (SSI) devices** : < 10 logic gates, inputs and outputs connected directly to the package pins
  - **Medium-Scale Integrated (MSI) devices**: 10-100 logic gates, perform specific elementary logic functions (e.g. 4-bit adder)
  - **Large-Scale Integrated (LSI) devices**: 100-10000 logic gates (e.g. memory or small microprocessor)
  - **Very-Large-Scale Integrated (VLSI) devices**: > 10,000 logic gates (e.g. complex microprocessor, digital signal processor)



IC (NOR) used on spacecraft Apollo (~ 1961)



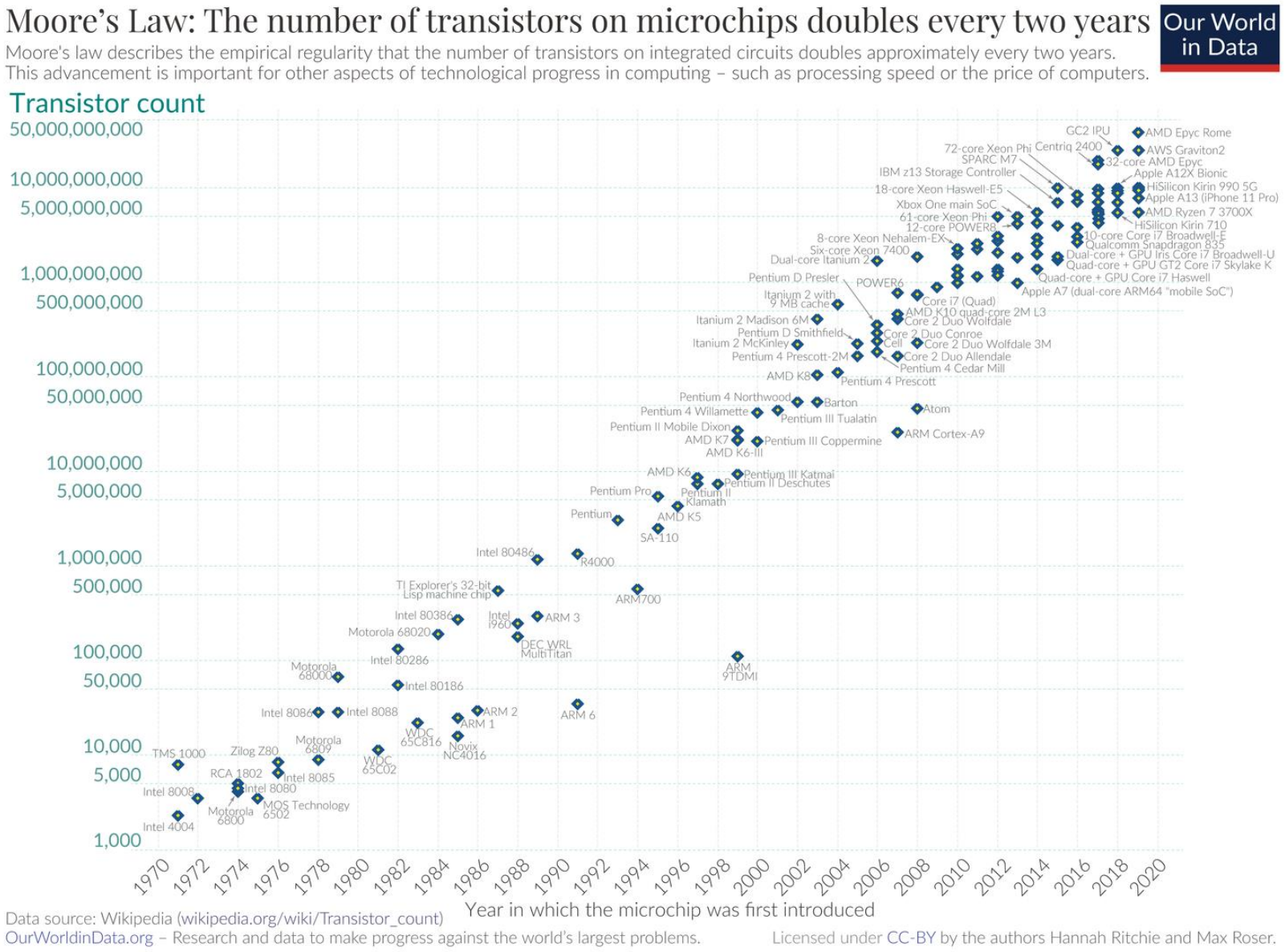
Atmel Diopsis 740 Dual-core DSP



Source: Wikipedia

Moore's Law

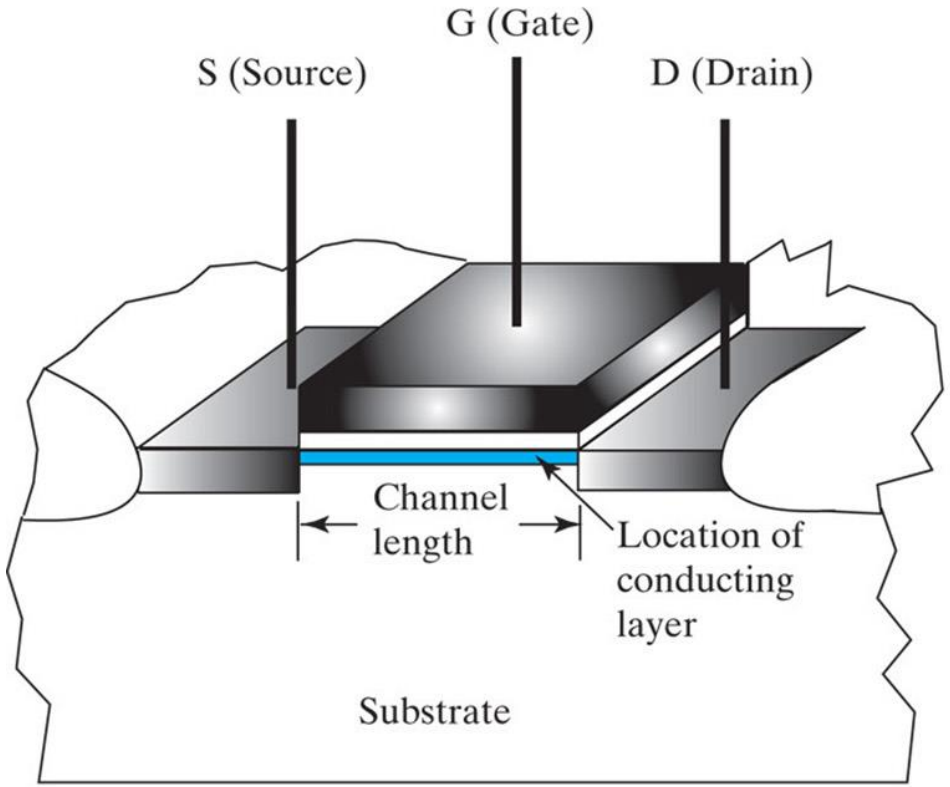
- In 1965, Gordon Moore (Fairchild Semiconductor and Intel founder) predicted that the **number of components per IC** doubles each year
- In 1975, the prediction was revised (doubling every two years)
- Since then, this prediction has been used in the semiconductor industry to guide planning and to set targets for research and development



CMOS Technology

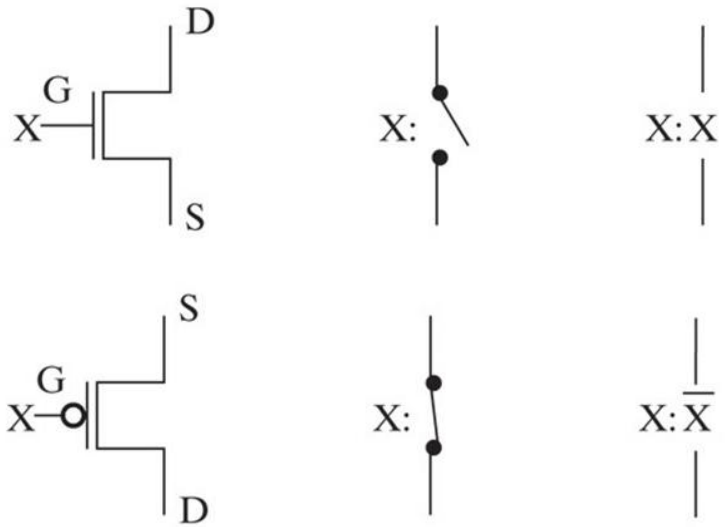
- Today the most commonly used technology for the implementation of integrated circuits is CMOS technology (Complementary Metal Oxide Semiconductor)
- The basic element of CMOS technology is the MOS (Metal Oxide Semiconductor) transistor, made from materials such as silicon, with properties located between conductors and insulators

Typical dimensions for the channel length of a modern transistor are in the order of 10 nm (10^{-9} m)



Transistors and Switch Model

- In **digital systems** and in a **simplified treatment**, transistors can be treated as switches: if a switch is open (OFF) there is no path through the circuit (a signal does not pass), if it is closed (ON) there is a path (a signal passes)
  - This first-order approximation ignores the physical details and captures only the **logical behavior**
- CMOS technology uses 2 types of transistors, having different behavior as they are made with different materials. The switch has 3 terminals (Gate, Source, and Drain). The position of the switch is controlled by the logic value at the Gate terminal
  - N-channel MOS: **OFF** (switch open) if the **logic value at the Gate is low**, **ON** (switch closed) if the **logic value at the Gate is high**
  - P-channel MOS: **OFF** (switch open) if the **logic value at the Gate is high**, **ON** (switch closed) if the **logic value at the Gate is low**



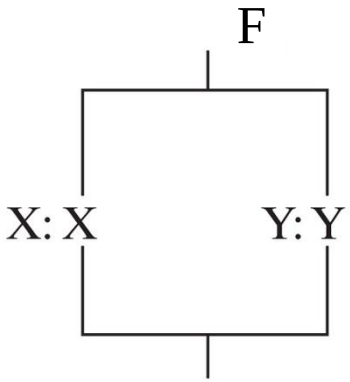
N-MOS: Path exists between S and D if X=1; path does not exist if X=0

P-MOS: Path exists between S and D if X=0; path does not exist if X=1

Circuit of Switches

- A **digital circuit** is made of transistors that are modeled as switches: a circuit implements a function F if there is a **path through the circuit when F is equal to '1' and there is no path when F = '0'**
  - Example 1: circuit with n-channel transistors

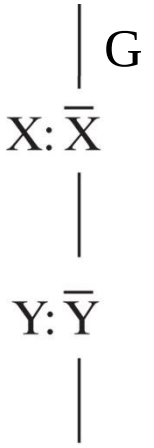
Switches in series provide AND gates, switches in parallel provide OR gates



F = '1' if
X = 1 **OR** Y = 1
$F = X + Y$

N-MOS: Path exists if X=1; path does not exist if X=0

  - Example 2: circuit with p-channel transistors

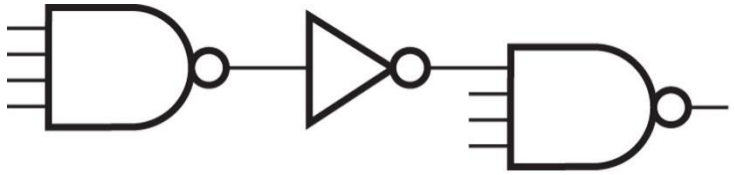


G = '1' if
X = 0 **AND** Y = 0
$G = \overline{X} \cdot \overline{Y} = \overline{X + Y}$

P-MOS: Path exists if X=0; path does not exist if X=1

Technology Parameters

- **Fan-in**: number of inputs available on a logic gate
  - The fan-in is typically less than 4-5 for primitive logic gates in high-speed circuits: to create gates with more than 5 inputs, multiple gates (with lower fan-in) connected together are used
  - Example:



7-input NAND gate is made with two 4-input NAND gates and one inverter

- **Fan-out**: quantifies the number of standard loads driven by the output of the gate, i.e. what can be connected to the output of a gate without degrading the performance of the gate itself
  - Typically, the fan-out is measured in terms of number of standard loads (= minimum-size inverters)
  - Example: a minimum-size inverter connected to the output of a gate constitutes one standard load. If a gate drives 6 minimum-size inverters, it has a fan-out equal to 6 standard loads

Technology Parameters

- **Noise margin**: maximum external noise superimposed on the input signal that does not cause an undesirable change of the circuit output
- **Propagation delay**: time required for the change in logic value of a signal to propagate from the input to the output of a gate. The speed of a circuit made of multiple gates is inversely proportional to the propagation delay of the slowest gate
- **Power dissipation**: power consumed by a gate. Since power is dissipated in the form of heat, the specification of the maximum consumption of a circuit is a function of the operating temperature. For battery-powered systems, the power consumption determines the lifetime of the battery

Technology Parameters: the Cost

- The **cost** of a gate is the contribution to the total cost of the integrated circuit that contains the gate itself
- The cost is directly proportional to the occupied **silicon area**. It depends on the **number of transistors, size of transistors**, **and on the connections** (wiring)
- The costs for the design of an integrated circuit are divided into
  - **Non-Recurring Engineering (NRE) costs**: one-time costs (design, verification, test, ...) that are amortized for large production volumes
  - **Production costs**: direct costs, which must be incurred for each unit chip produced (materials, labor, energy, ...)
  - For small production volumes, NRE costs dominate the production costs for each unit

Programmable Logic Devices (PLD)

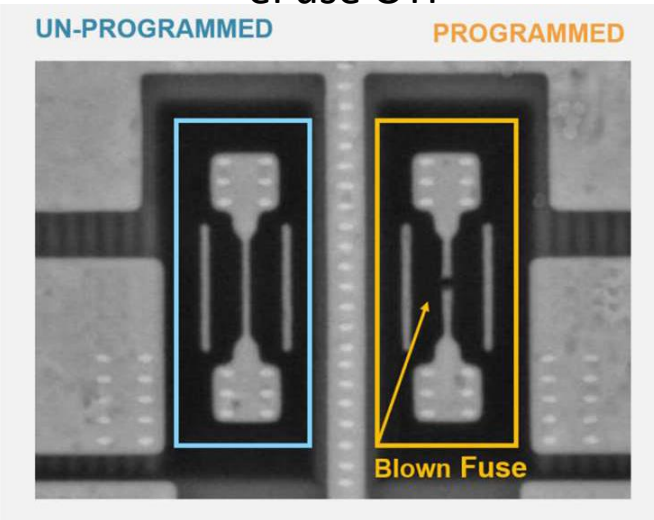
- So far, we have talked about fixed implementations, that is, technologies manufactured with one or more integrated circuits that implement predefined, non-modifiable logic functions
- In contrast, **programmable logic devices** (**PLD**) are integrated circuits manufactured with structures that can be **configured by the user to perform different logic functions**
  - The **configuration** procedure **"changes" the hardware**, determining the particular logic function that will be implemented in the PLD
  - Configuration procedure ≠ software programming procedure in a microcontroller!
- We will see 4 categories of PLD:
  - Read-Only Memory (**ROM**)
  - Programmable Logic Array (**PLA**)
  - Programmable Array Logic (**PAL**)
  - Field Programmable Gate Array (**FPGA**)

Programmable Implementation Technologies

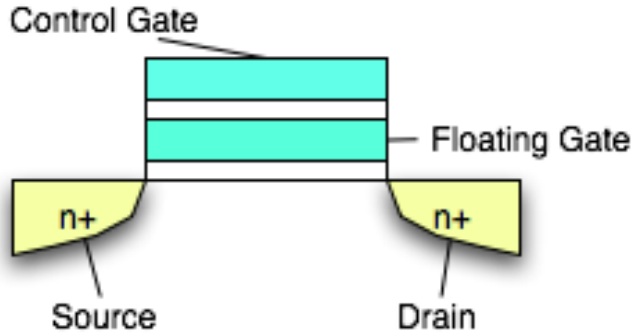
1) Technologies involving an **irreversible change** in their structure at the time of configuration. They can be configured only once
   - Fuses/anti-fuses: connections between the logic elements are eliminated/created selectively by applying high electrical voltages
   - Mask programming: connections are made by the manufacturer during the last phase of the chip manufacturing process

2) Technologies whose configuration determines a **reversible change**. They can be reconfigured multiple times
   - Based on switches connected to volatile memory elements (Static Random Access Memory, SRAM). Programming information is lost when power is removed
   - Based on the control of transistor switching, uses transistors with 'configurable' threshold. Typically non-volatile, they maintain programming even in the absence of power. MOSFETs with programmable threshold voltage are based on Floating gate (FG) technology

eFuse OTP



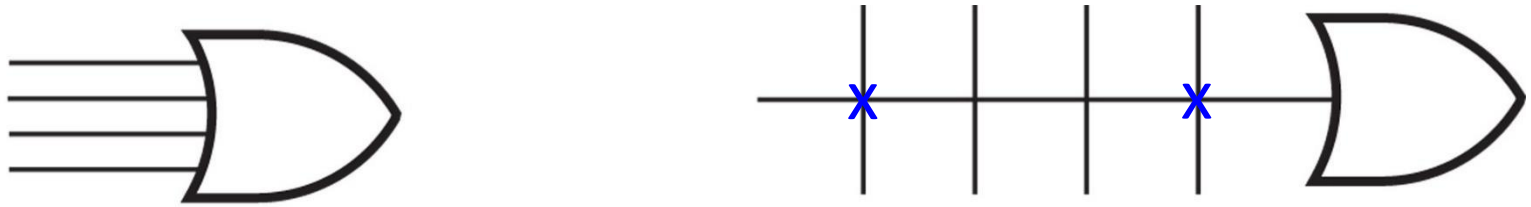
https://www.pufsecurity.com/technology/otp/



https://it.wikipedia.org/wiki/Floating_Gate_MOSFET

Symbols

- Logic gates in a PLD can have a large number of inputs
  - To indicate logic gates with high fan-in in programmable logic devices we will use compact symbols:



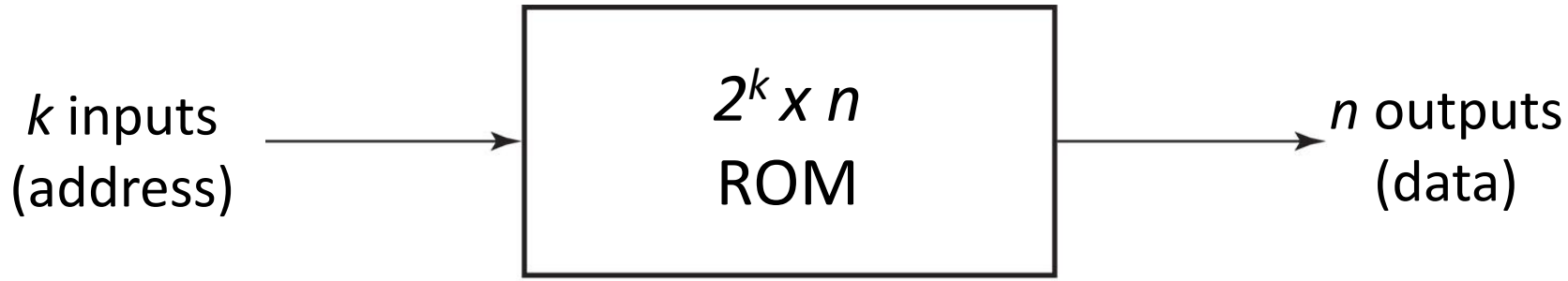
(a) Conventional symbol (b) Array logic symbol

  - The presence of an x at the intersection of two lines indicates that there is connection, the absence of x indicates that there is no connection

Read-Only Memory (ROM)

- A ROM (**Read Only Memory**) stores information **permanently**, even when power is turned off (**non-volatile memories**)
- Programmable ROMs (PROMs) can be erased and reprogrammed through special processes, which are performed more rarely than the normal processes of reading and writing in the remaining classes of memories (RAM: Random Access Memory)
- A ROM can be made with different technologies
  - **ROM**: mask programmed (programmed by the manufacturer in the foundry)
  - **PROM**: fuse or anti-fuse (programmed by the user only once)
  - **EPROM** (Erasable and Programmable): erasable by ultraviolet rays, it can be erased and reprogrammed by the user
  - **EEPROM or E^{2}PROM** (Electrically Erasable and Programmable): electrically erasable by the user and reprogrammable
  - **Flash memory**: evolution of E^{2}PROM, uses transistors with 'configurable' threshold (erasable in blocks)
- The choice of technology depends on production volumes, type of use (e.g. reprogramming frequency) and performance (e.g. speed)

ROM: Block Diagram



- The **input** contains the **address** of the data
- The **output** contains the **data** bit (word) selected by the address provided at the input
- A ROM can store 2^k **words of** n **bits**
- Being a **read-only memory**, there is no possibility to write on it, i.e. a ROM does not have input data

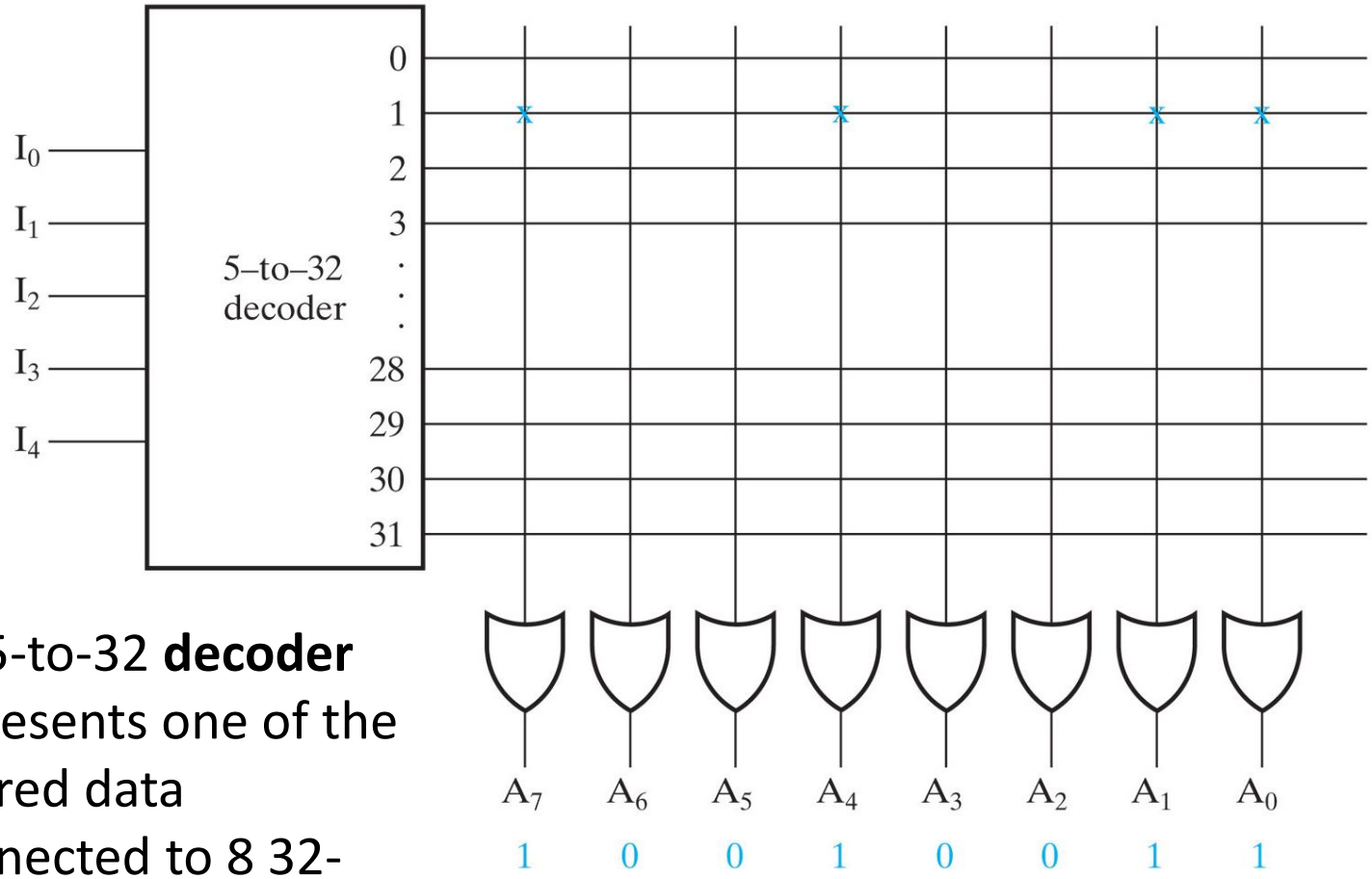
Conceptual Idea of a Memory

Memory Address		Memory Contents
Binary	Decimal	
0000000000	0	10110101 01011100
0000000001	1	10101011 10001001
0000000010	2	00001101 01000110
	.	.
	.	.
	.	.
	.	.
	.	.
1111111101	1021	10011101 00010101
1111111110	1022	00001101 00011110
1111111111	1023	11011110 00100100

FIGURE 7-2
Contents of a 1024 × 16 Memory

32 x 8 ROM: Example (32 8-bit Words)

A **2^k x n ROM** consists of a k-to-2^k decoder and n OR gates

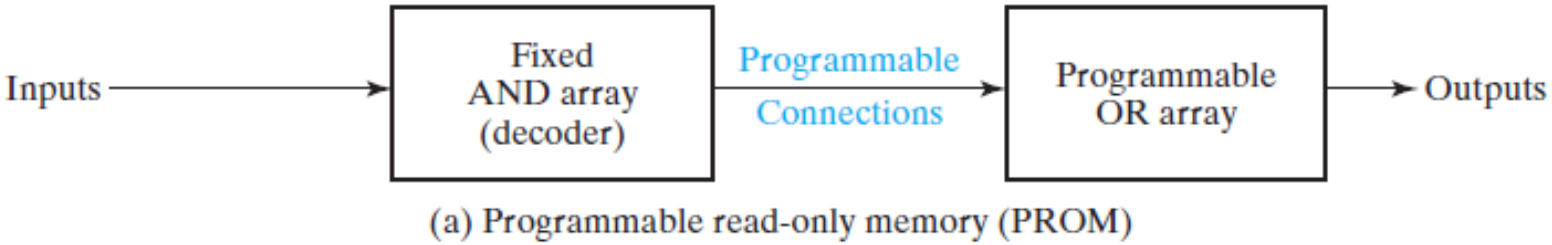


Fixed AND plane (decoder)

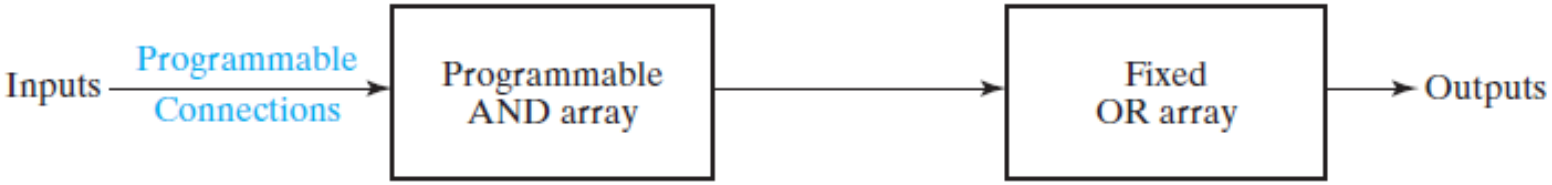
- Input (5 address bits) goes to a 5-to-32 **decoder**
- Each output of the decoder represents one of the 32 possible addresses of the stored data
- The 32 decoder outputs are connected to 8 32-input **programmable OR gates** (x indicates the presence of a connection)
- In the example, address 1 of the memory contains the word "10010011"

Programmable 32-input OR plane

Programmable Logic Devices (PLD)



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL) device

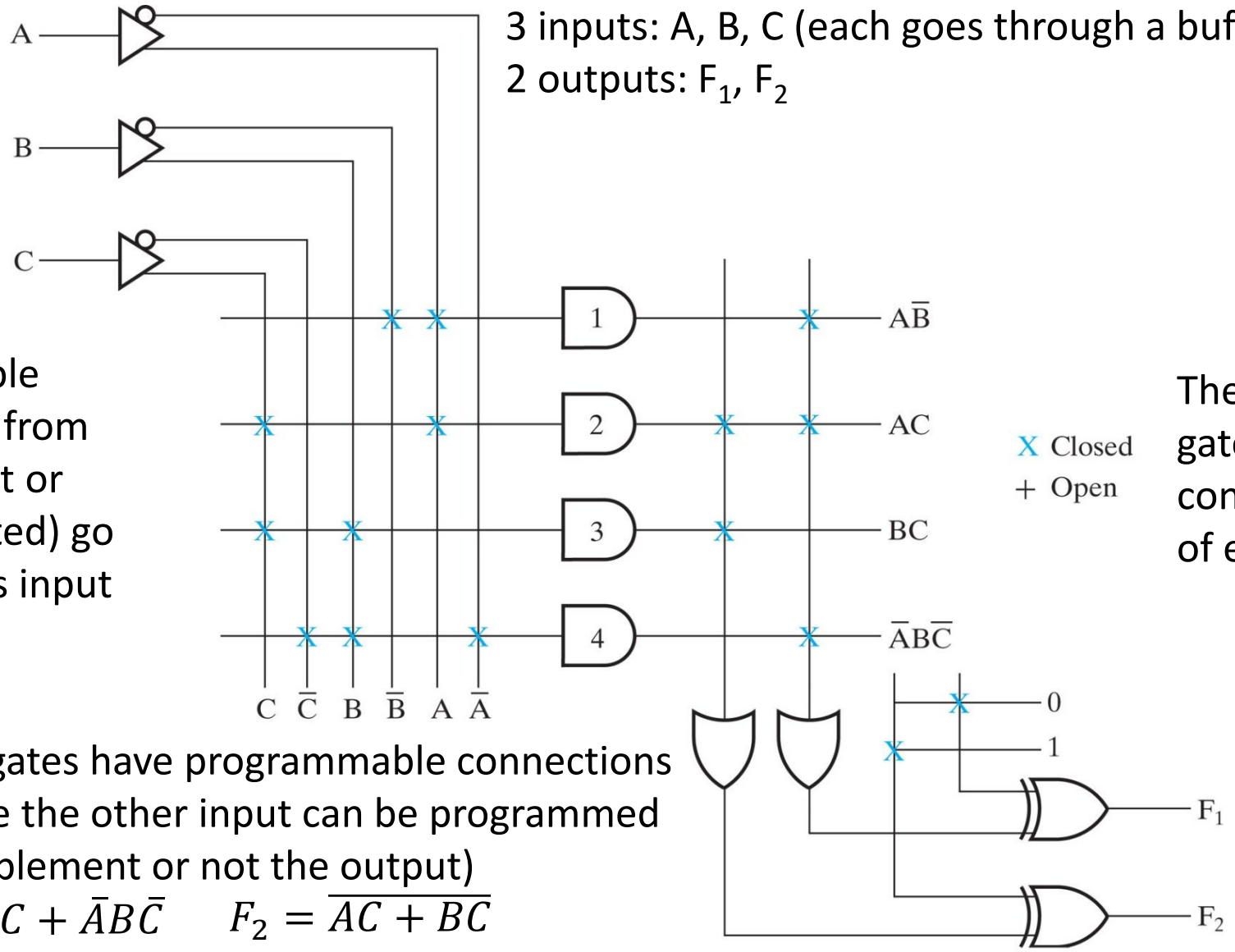
(c) Programmable logic array (PLA) device

Programmable Logic Array (PLA)

- It is a PLD similar to a ROM, but the decoder is replaced by an array of **programmable AND gates that generate the product terms of the input variables**
- The product terms are then **selectively connected to OR gates**, to implement the sum of minterms
- The difference with a ROM is that it does not generate all the minterms corresponding to the input variables, but only a programmable subset of them
- Commercial PLAs contain a very large number of gates and connections (otherwise they would not be cost effective)

PLA: Example

3 inputs: A, B, C (each goes through a buffer and an inverter)
2 outputs: F_1, F_2



Programmable connections from inputs (direct or complemented) go to AND gates input

The outputs of the AND gates have programmable connections to the input of each OR gate

The outputs of OR gates have programmable connections to XOR gates, where the other input can be programmed to '0' or '1' (to complement or not the output)

$$F_1 = A\bar{B} + AC + \bar{A}B\bar{C} \qquad F_2 = \overline{AC + BC}$$

PLA: Implementation of a Combinational Circuit

- The information needed to program a PLA to implement a given logic function is
  - Closed connections between the inputs (direct or complemented) and the AND gates
  - Closed connections between AND gates and OR gates
  - Whether the final sum of products is direct or complemented
- To design a PLA with minimum size, it is important to **minimize** (via Boolean simplifications) **the number of terms produced**, that is, sharing the product terms produced between the outputs
  - The number of literals in each mintermit is less important, since all the input variables, direct and complemented, are available
- For the design of combinational circuits using PLA, **optimization of two-level functions on multiple outputs** algorithms are often exploited

PLA: Example of Circuit Implementation

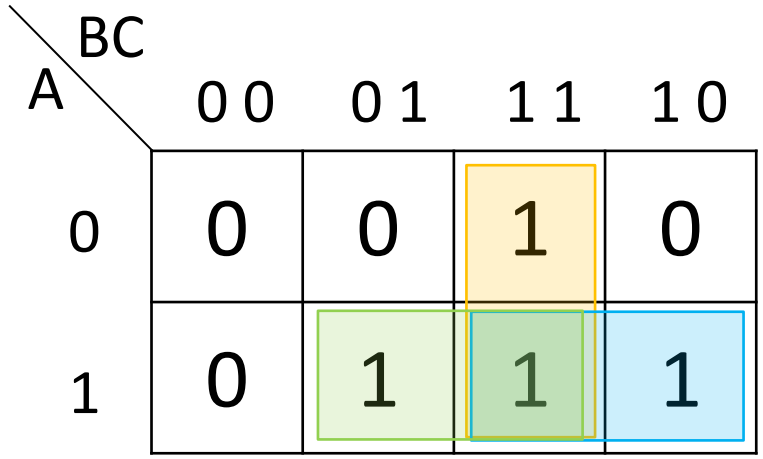
- Implement the following logic functions via a PLA

$$F_1(A, B, C) = \sum m(3,5,6,7)$$

$$F_2(A, B, C) = \sum m(1,2,3,7)$$

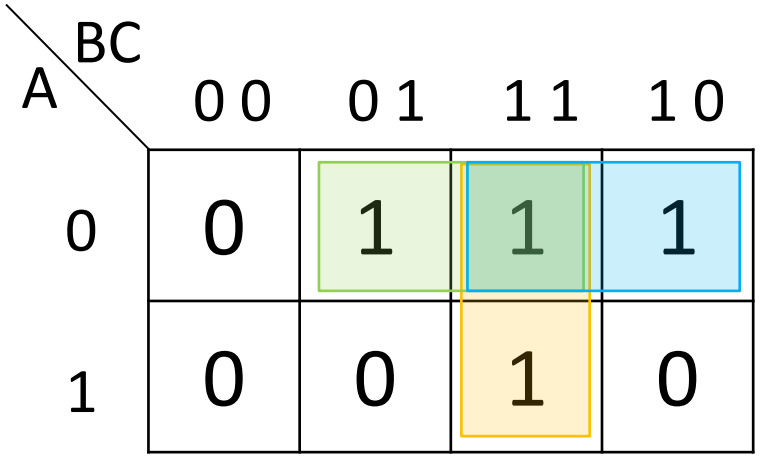
PLA: Example of Circuit Implementation

$$F_1(A,B,C) = \sum m(3,5,6,7) \qquad F_2(A,B,C) = \sum m(1,2,3,7)$$



$$F_1(A,B,C) = BC + AC + AB$$

5 product terms
(1 shared between
F_1 and F_2)



$$F_2(A,B,C) = BC + \bar{A}B + \bar{A}C$$

PLA: Example of Circuit Implementation

$$F_1(A,B,C) = \sum m(3,5,6,7) \qquad F_2(A,B,C) = \sum m(1,2,3,7)$$

A \ BC	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

Let's consider $\overline{F_1}$:

$$\overline{F_1(A,B,C)} = \bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C}$$

4 product terms (2 shared between F_1 and F_2), with 2 non-prime implicants

A \ BC	0 0	0 1	1 1	1 0
0	0	1	1	1
1	0	0	1	0

Let's consider F_2:

$$F_2(A,B,C) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + BC$$

=> Using two non-prime implicants,
sharing between F_1 and F_2 is optimized

Programmable Array Logic (PAL)

- It is a PLD with **a programmable array of AND gates and a fixed array of OR gates**
- PAL is less flexible than PLA (where both OR and AND gate arrays are programmable), but has a less complex structure, so design is easier
  - Unlike PAL, product terms (AND gates) cannot be shared between different OR gates
- For the design of combinational circuits using PAL, **optimization of two-level functions on single output** algorithms are used
- Typically a commercial PAL contains at least 8 inputs and 8 outputs, otherwise it would not be cost-effective

PAL: Example of Circuit Implementation

- Implement the following logic functions through a PAL

$$W(A,B,C,D) = \sum m\,(2,12,13)$$

$$X(A,B,C,D) = \sum m\,(7,8,9,10,11,12,13,14,15)$$

$$Y(A,B,C,D) = \sum m\,(0,2,3,4,5,6,7,8,10,11,15)$$

$$Z(A,B,C,D) = \sum m\,(1,2,8,12,13)$$

PAL: Example of Circuit Implementation

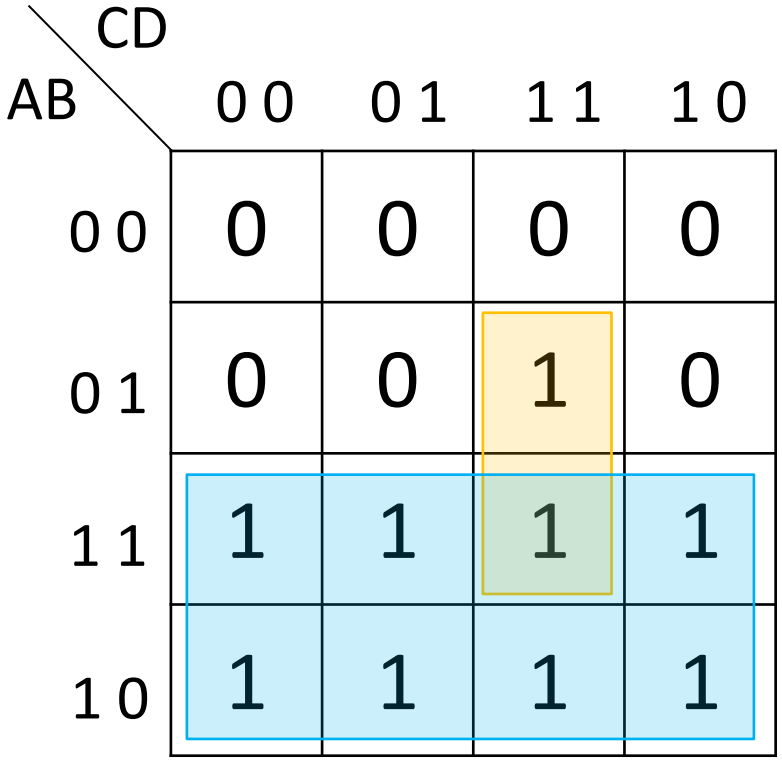
$$W(A,B,C,D) = \sum m\,(2,12,13)$$

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	0	1
0 1	0	0	0	0
1 1	1	1	0	0
1 0	0	0	0	0

$$W = AB\bar{C} + \bar{A}\bar{B}C\bar{D}$$

PAL: Example of Circuit Implementation

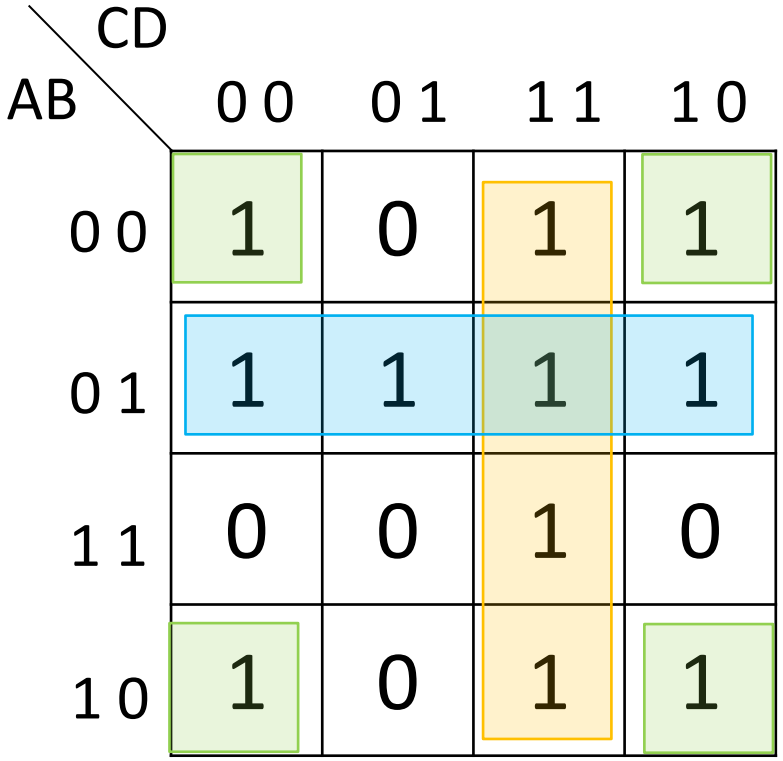
$$X(A,B,C,D) = \sum m\,(7,8,9,10,11,12,13,14,15)$$



$$X = A + BCD$$

PAL: Example of Circuit Implementation

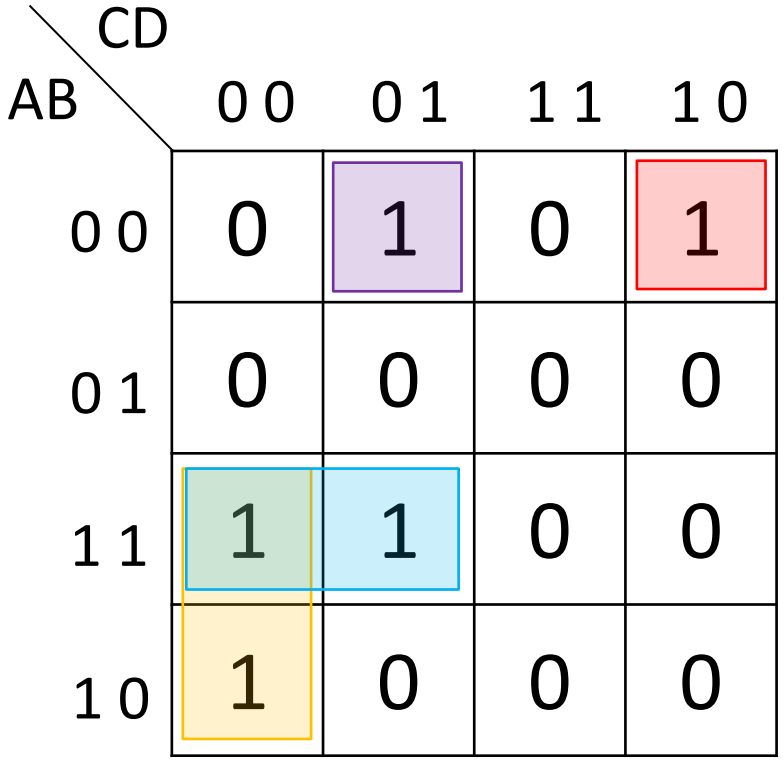
$$Y(A,B,C,D) = \sum m\,(0,2,3,4,5,6,7,8,10,11,15)$$



$$Y = \bar{A}B + CD + \bar{B}\bar{D}$$

PAL: Example of Circuit Implementation

$$Z(A,B,C,D) = \sum m\,(1,2,8,12,13)$$

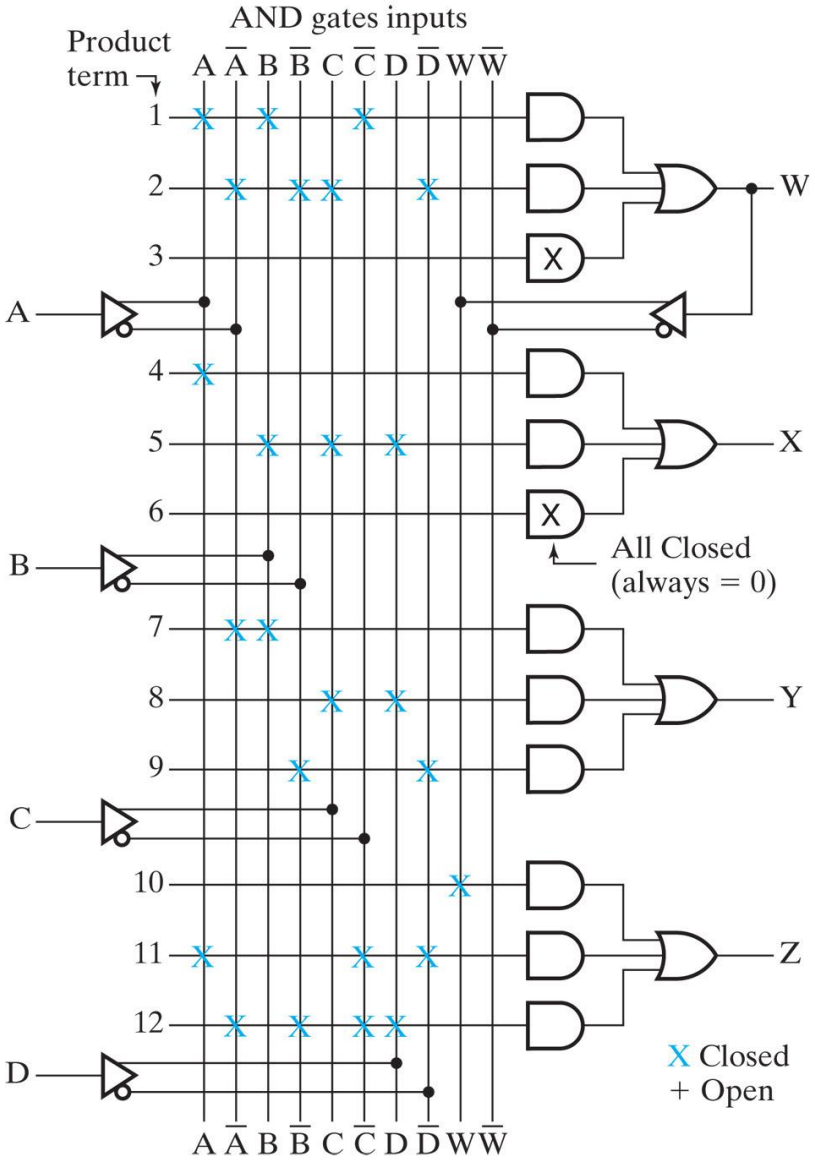


$$Z = AB\bar{C} + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D}$$

$$W = AB\bar{C} + \bar{A}\bar{B}C\bar{D}$$

$$\Rightarrow Z = W + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$

PAL: Example of Circuit Implementation



$$W = AB\bar{C} + \bar{A}\bar{B}C\bar{D}$$

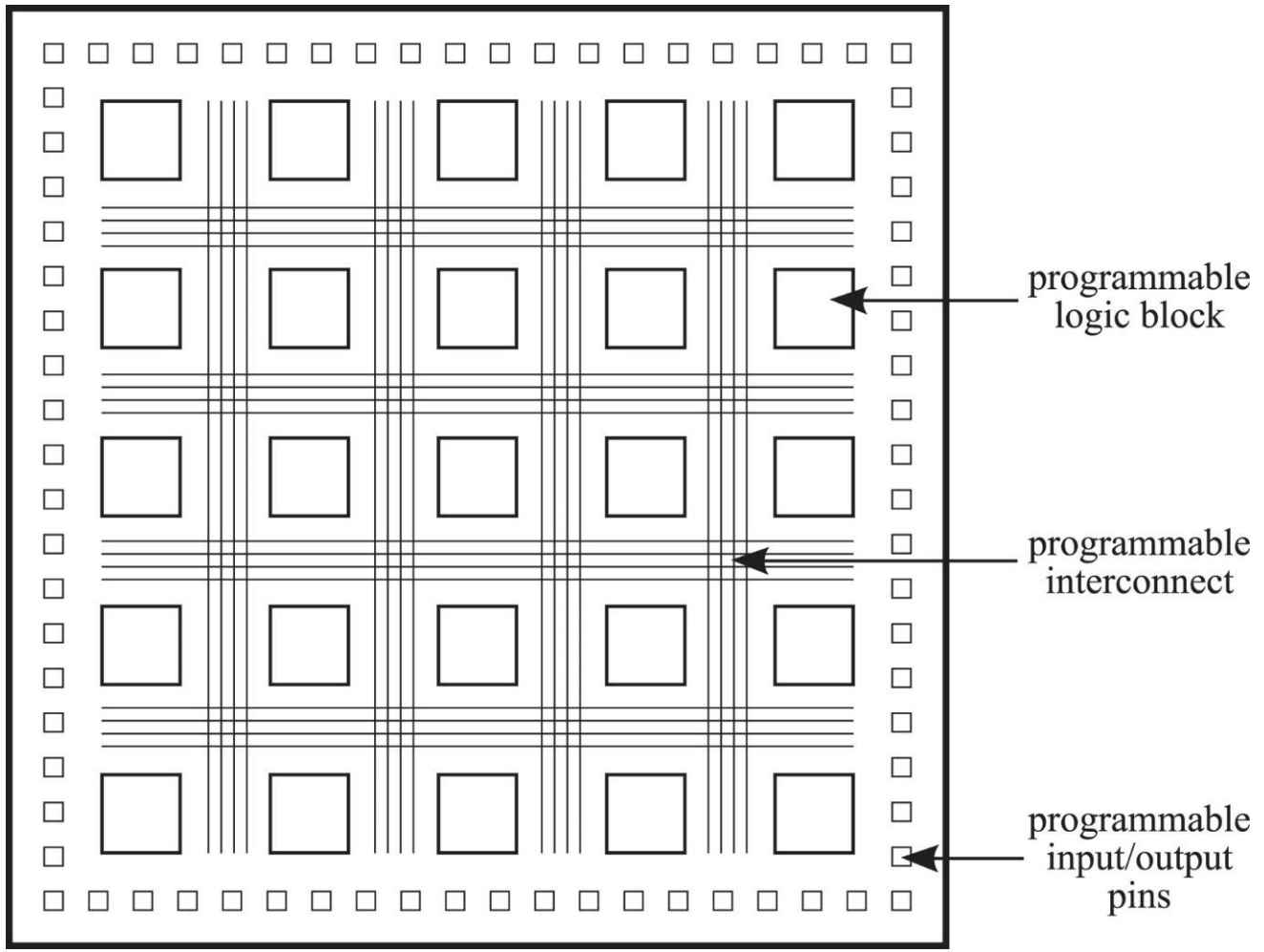
$$X = A + BCD$$

$$Y = \bar{A}B + CD + \bar{B}\bar{D}$$

$$Z = W + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$$

Field Programmable Gate Array (FPGA)

- FPGAs are the most popular type of PLDs currently available
- There are different FPGAs on the market, with a variety of features, but most of the solutions have the following elements in common
  - **Programmable logic blocks**
  - **Programmable interconnections**
  - **Programmable I/O pins**

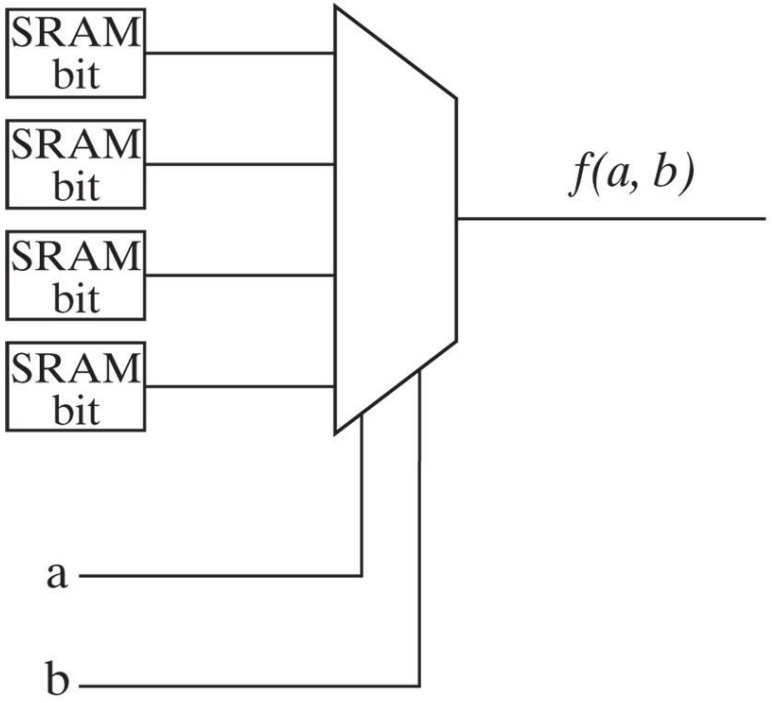


Field Programmable Gate Array (FPGA)

- In addition to these configurable elements, FPGAs can contain **specialized blocks of dedicated logic** (memories, components for arithmetic calculation, microprocessors)
- The advantages of using FPGAs instead of other programmable devices are the **availability of configurable logic blocks and flip-flops**, and **ease of reconfiguration**
- The information on the implemented circuit is contained in the **configuration memory**, which can be volatile (SRAM) or non-volatile (Flash, fuse/anti-fuse)
- To program an FPGA one needs to specify the value of the configuration bits for the logic blocks, routing (interconnects), and input/output pins

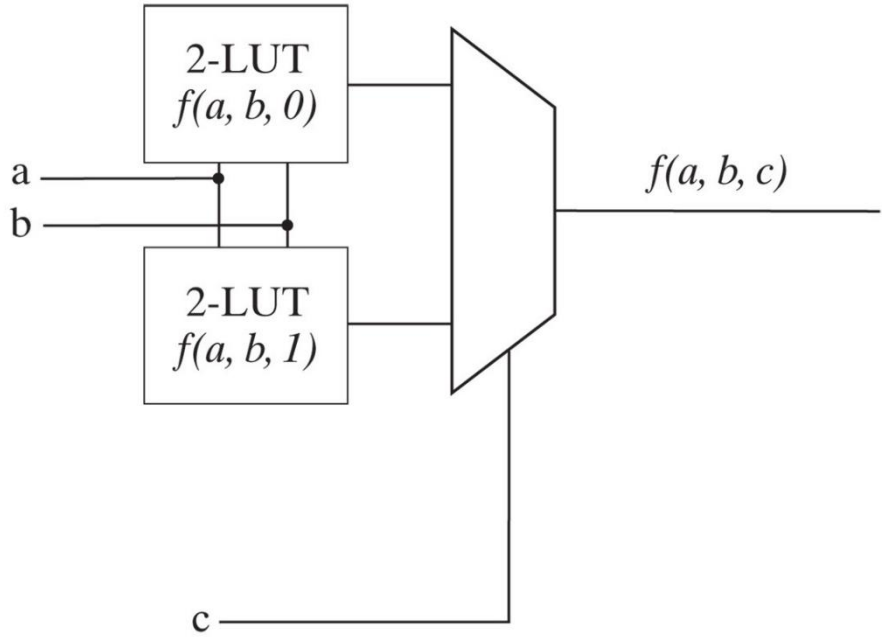
Programmable Logic Blocks

- In FPGAs, blocks with combinational and sequential logic are available. The blocks can be configured by the user to perform the desired functions
- In many FPGA families the programmable logic blocks contain **Look-UpTable (LUT)**, i.e. memories of size 2^kx 1 implementing the **truth table of a combinational function of k variables**



2-input LUT (a, b):
By appropriately setting the FPGA configuration memory bits to '0' or '1' (data input of mux), the desired logic function can be implemented (out of the possible 16 Boolean functions with 2 input variables)

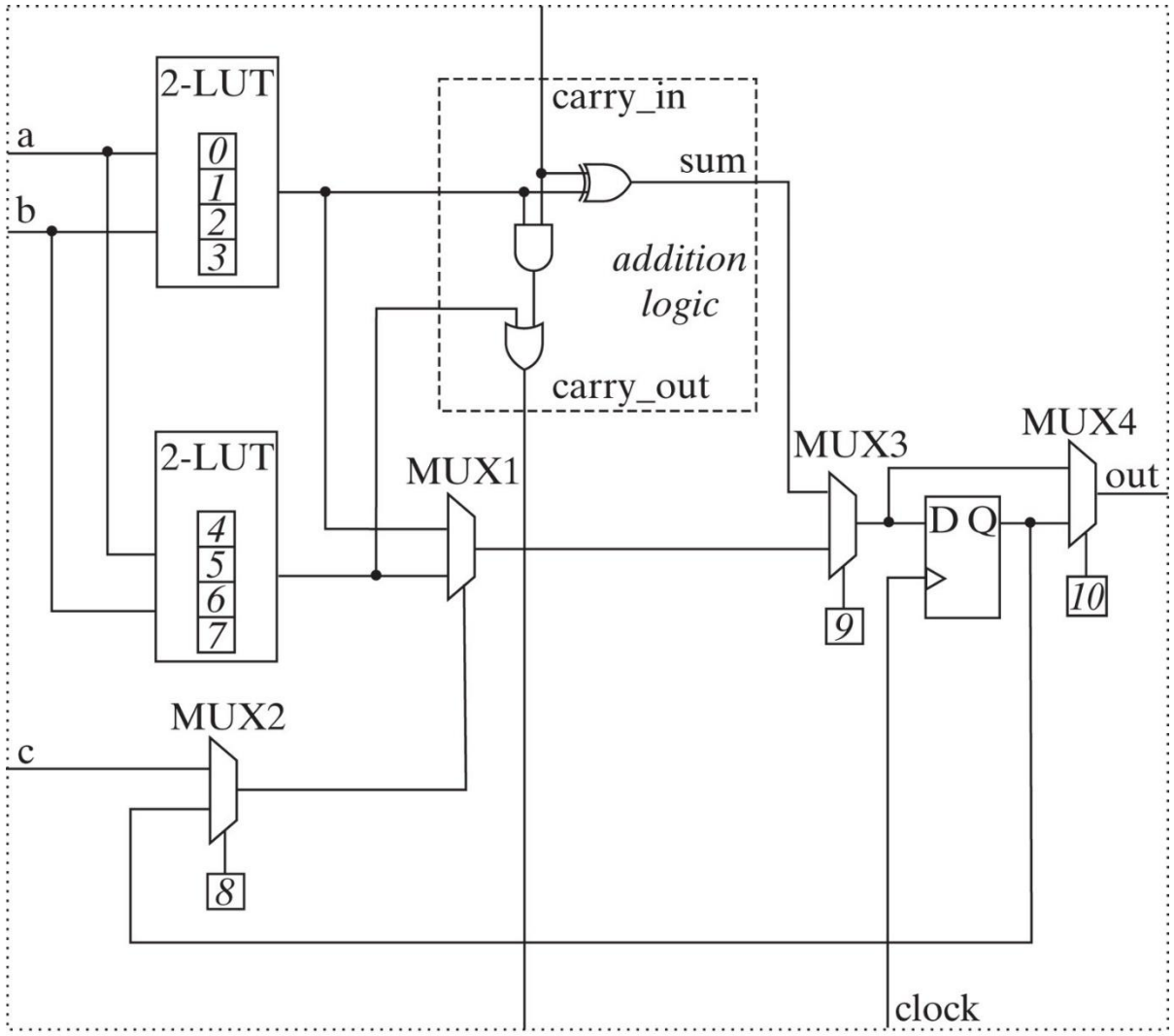
Programmable Logic Blocks



3-input LUT (a, b, c):
If we connect a 2-input LUTs with a 2-to-1 multiplexer, we get a 3-input LUT (a, b, c)

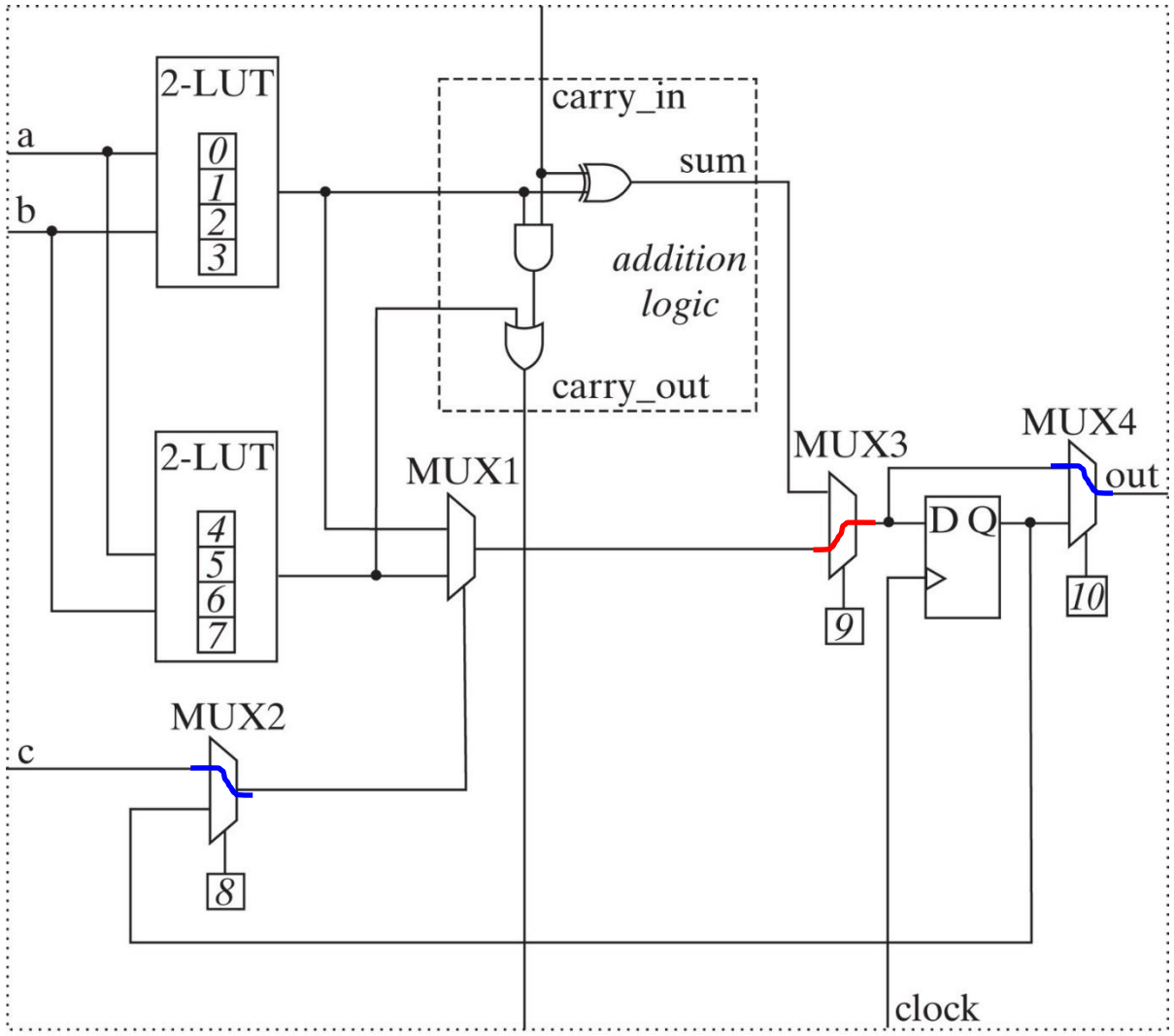
- In addition to LUTs, programmable logic blocks also contain **other elements** (**flip-flops, multiplexers, arithmetic blocks**, etc.) to implement a variety of logic functions

Programmable Logic Blocks: Example



- MUX1 + 2 2-LUT: implement a combinational function of 3 variables f (a, b, x), configuration bits 0-7 determine the truth table of the function
- MUX2 (config. bit 8) selects if the third input variable of f is c or the FF output
- MUX3 (config. bit 9) selects one between the output of MUX 1 and the bit sum of the adder as a FF input
- MUX4 (config. bit 10) selects one of the combinational block and the FF output as the output of the programmable block
- Addition logic: logic block to implement a 1-bit full adder by configuring the upper 2-LUT to be the function $f(a,b) = a \oplus b$ and the lower 2-LUT to be the function $f(a, b) = ab$. Then the *sum* signal is equal to $a \oplus b \oplus carry\_in$ and the *carry_out* signal is equal to $ab + carry\_in(a \oplus b)$

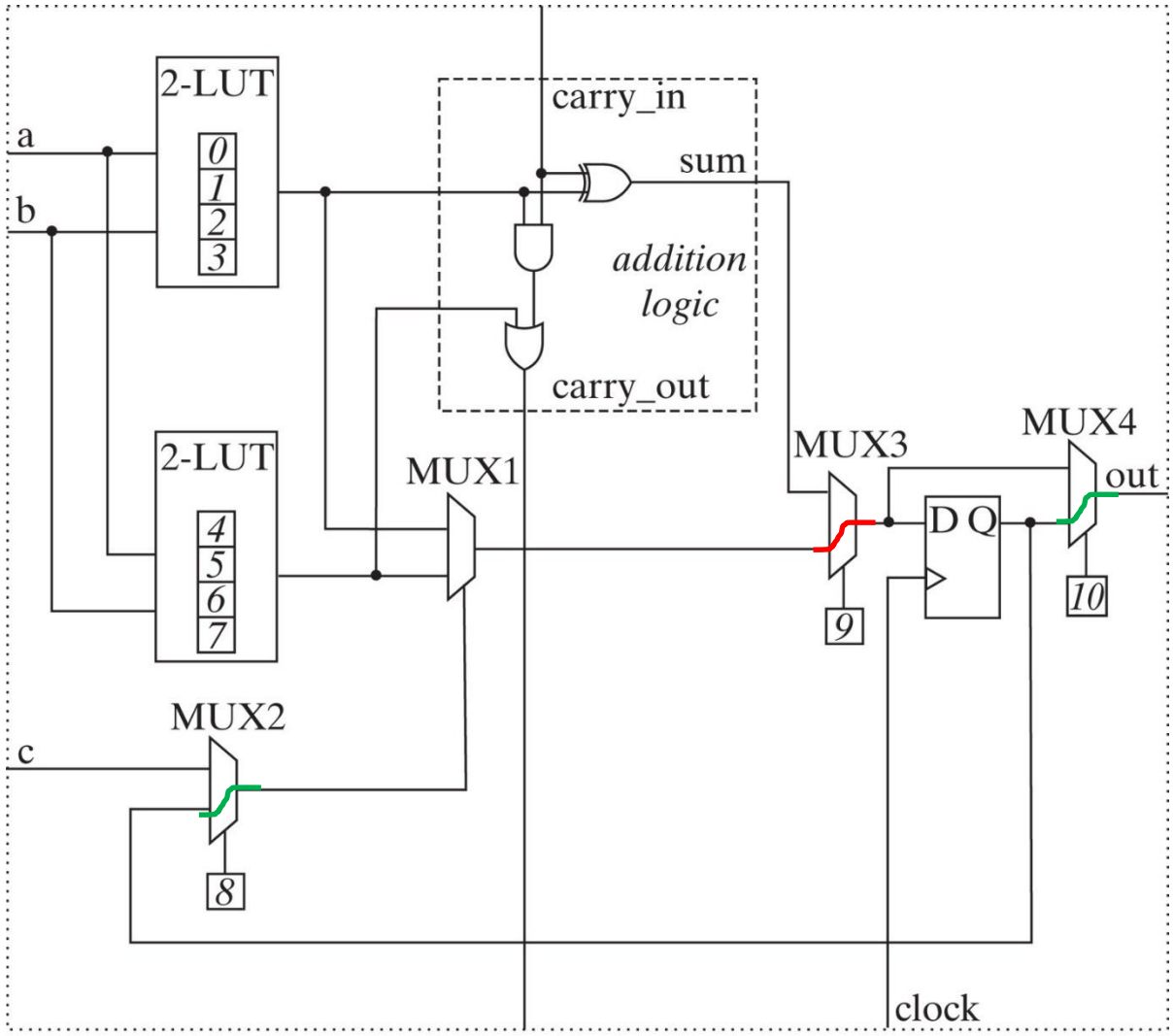
Programmable Logic Blocks: Example



Suppose that MUX3 (config. bit 9) is programmed to select the output of MUX1. Depending on how MUX2 and MUX4 are configured, the output out can be for example

1) Any combinatorial function of inputs a, b, c (if config. bit 8 and config. bit 10 = '0')

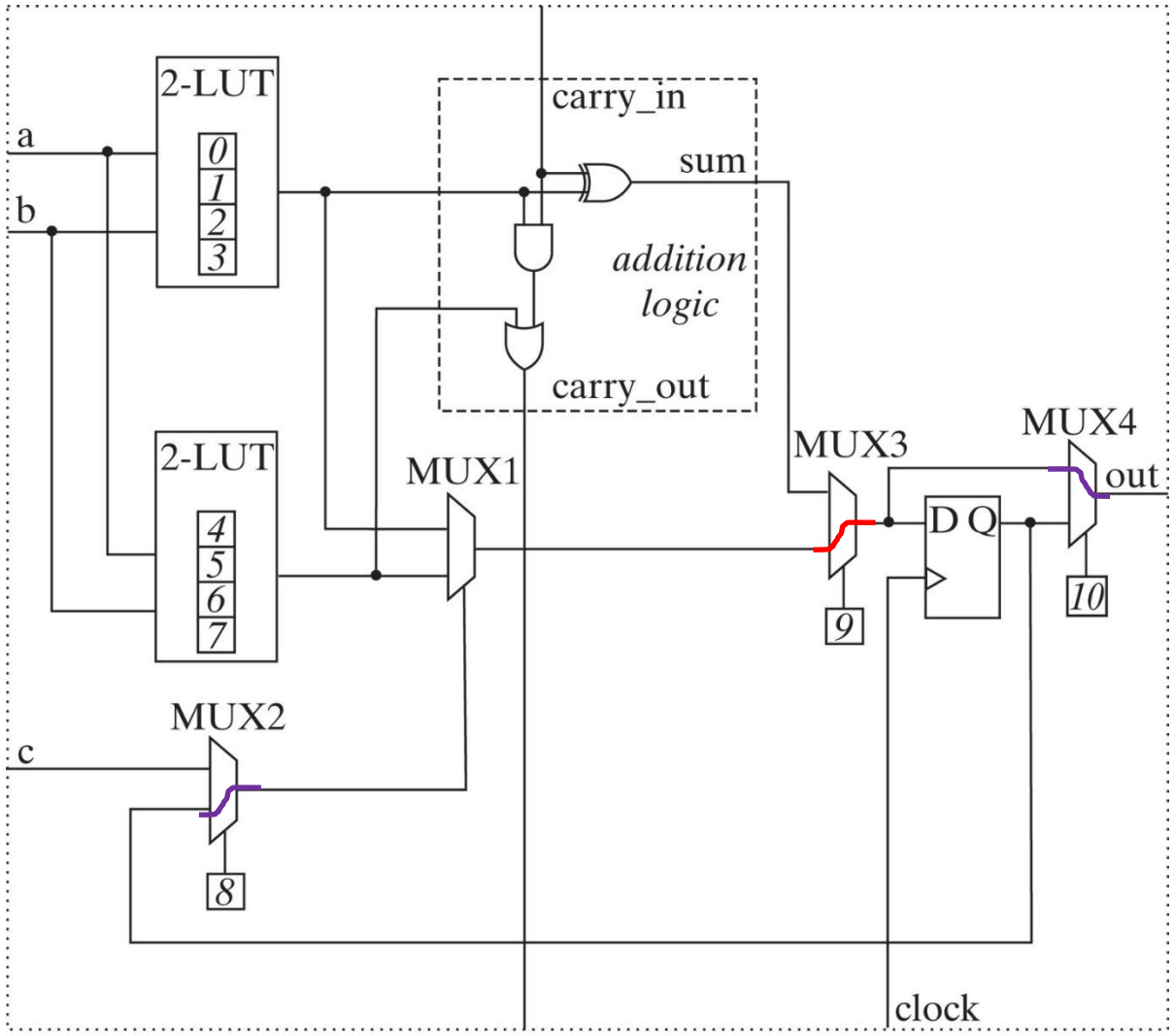
Programmable Logic Blocks: Example



Suppose that MUX3 (config. bit 9) is programmed to select the output of MUX1. Depending on how MUX2 and MUX4 are configured, the output out can be for example

2) A Moore machine (if config. bit 8 and config. bit 10 = '1', the output depends only on the present state and not on the inputs)

Programmable Logic Blocks: Example



Suppose that MUX3 (config. bit 9) is programmed to select the output of MUX1. Depending on how MUX2 and MUX4 are configured, the output out can be for example

3) A Mealy machine (if config. bit 8 = '1' and config. bit 10 = '0', the output depends not only on the state, but also on the inputs)

Disclaimer

Figures from*Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano |Kime| Martin