B38DF
Computer Architecture and Embedded Systems

Alexander Belyaev

Heriot-Watt University
School of Engineering & Physical Sciences
Electrical, Electronic and Computer Engineering

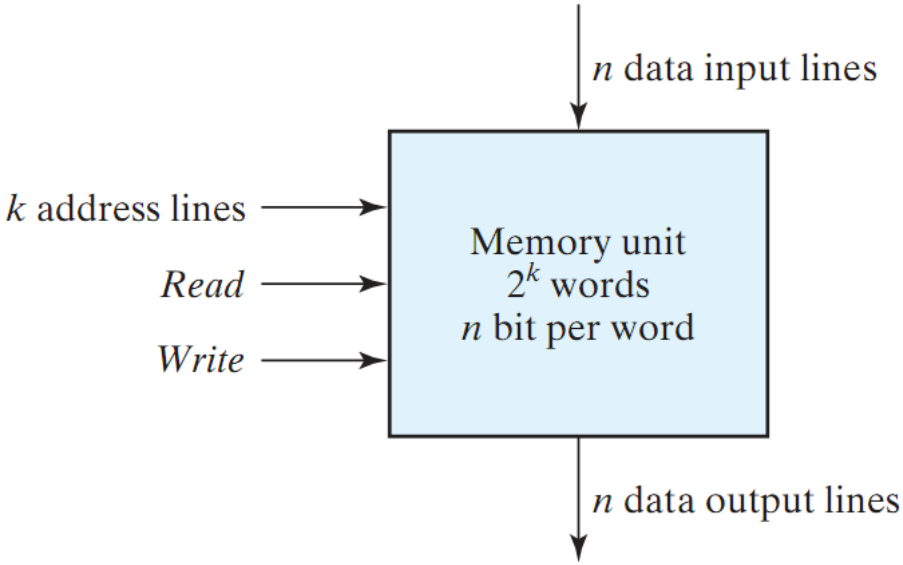
E-mail: a.belyaev@hw.ac.uk
Office: EM2.29

Based on the slides prepared by Dr. Mustafa Suphi Erden and Dr. Senthil Muthukumaraswamy

Memory

Memory: A collection of cells capable of storing binary information (1s or 0s) – in addition to electronic circuit for storing (writing) and retrieving (reading) information.

- n data lines (input/output)
- k address lines
- 2^k words (data unit)
- Read/Write Control
- Memory size = 2^k X n



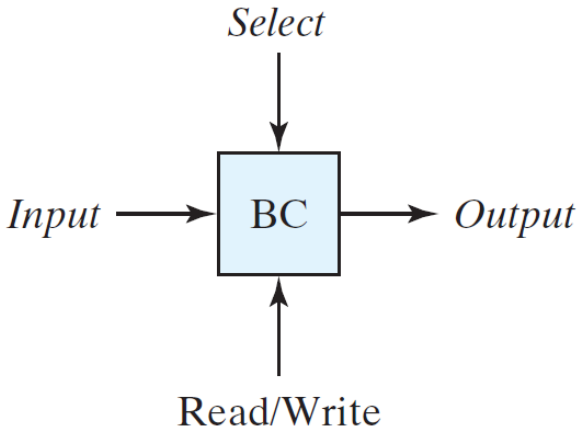
Memory (Cont.)

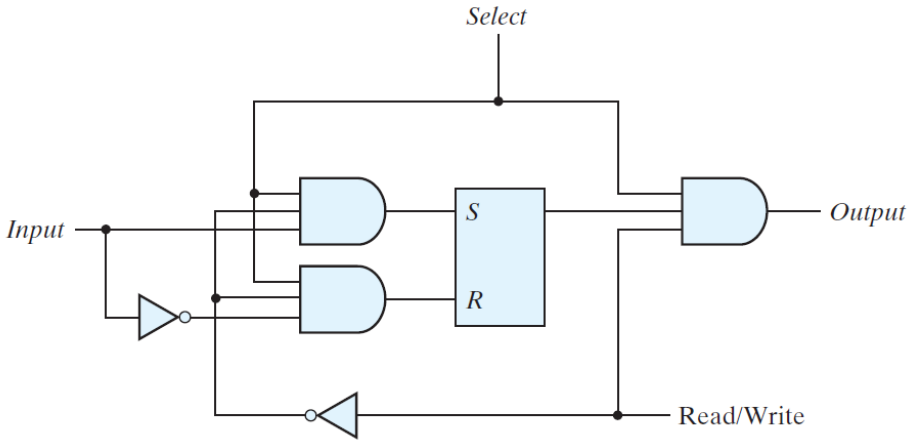
- Two Types of Memory:

- Random Access Memory (RAM):
  - Write/Read operations
  - **Volatile:** Data is lost when power is turned off

- Read Only Memory (ROM):
  - Read operation (no write)
  - **Non-Volatile:** Data is permanent.
  - PROM is programmable (allow special write)

RAM





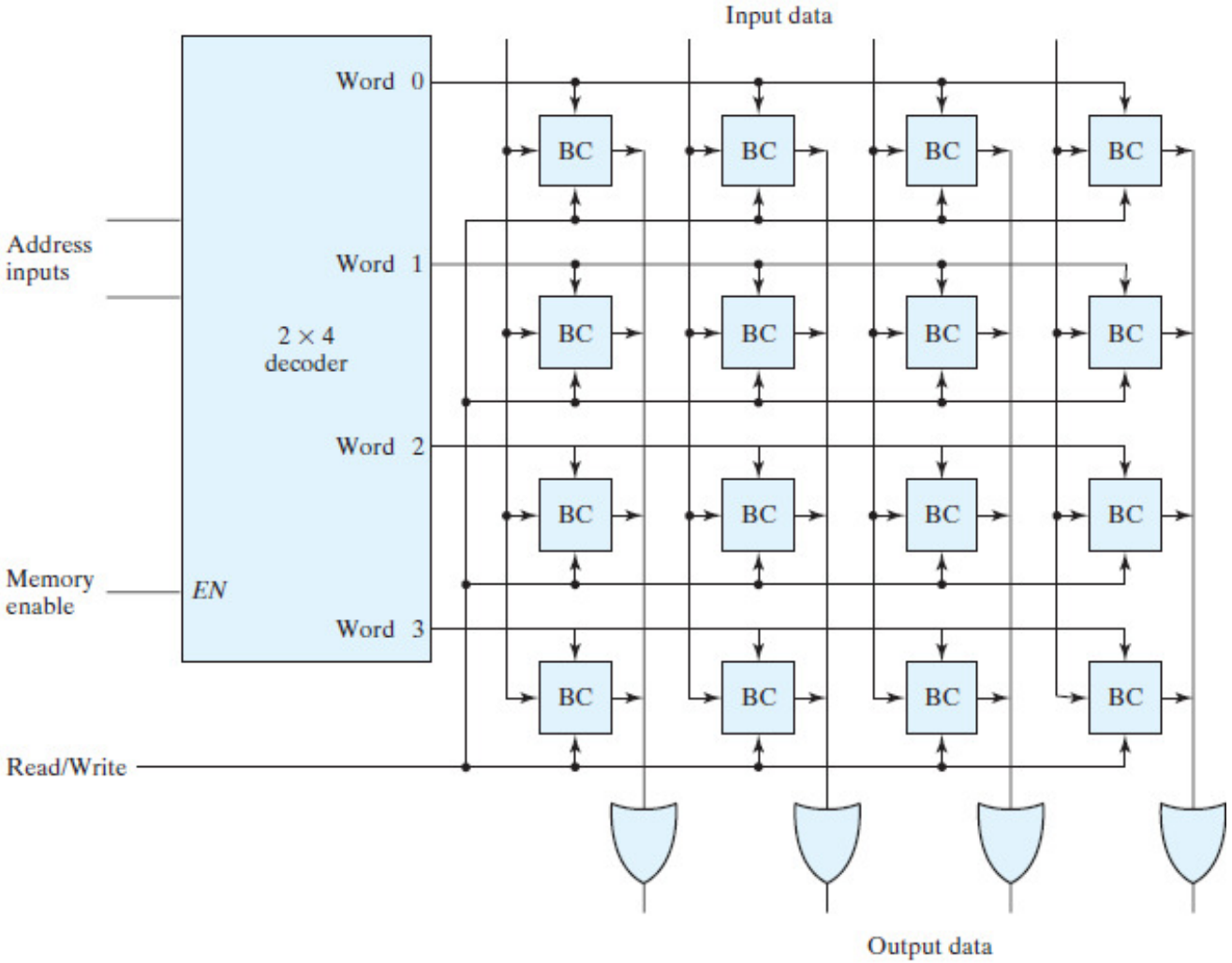
The binary cell (BC) stores one bit in its internal latch. The **select** input enables the cell for reading or writing, and the **read/write** input determines the operation of the cell when it is selected.

A **1** in the read/write input provides the **read** operation by forming a path from the latch to the output terminal.

A **0** in the read/write input provides the **write** operation by forming a path from the input terminal to the latch.

RAM

The logical construction of a small RAM. It consists of four words of four bits each and has a total of 16 binary cells. A memory with four words needs two address lines. The two address inputs go through a 2x4 decoder to select one of the four words.

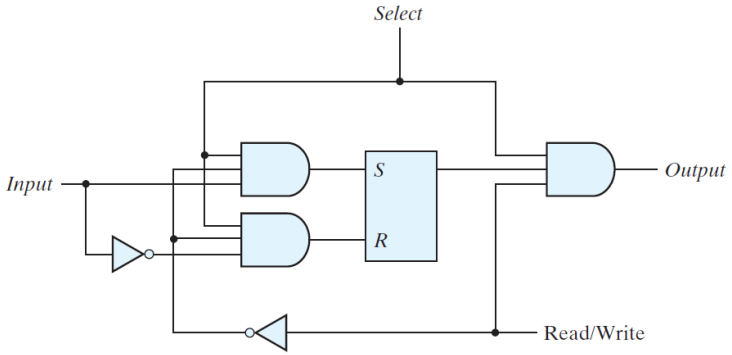


When the memory enable is 0, all outputs of the decoder are 0 and none of the memory words are selected. With the memory select at 1, one of the four words is selected, dictated by the value in the two address lines. Once a word has been selected, the read/write input determines the operation.

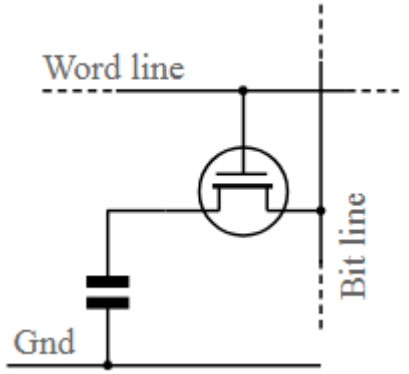
SRAM and DRAM

The **SRAM** (Static RAM) memory cell typically contains six transistors.

In order to build memories with higher density, it is necessary to reduce the number of transistors in a cell.

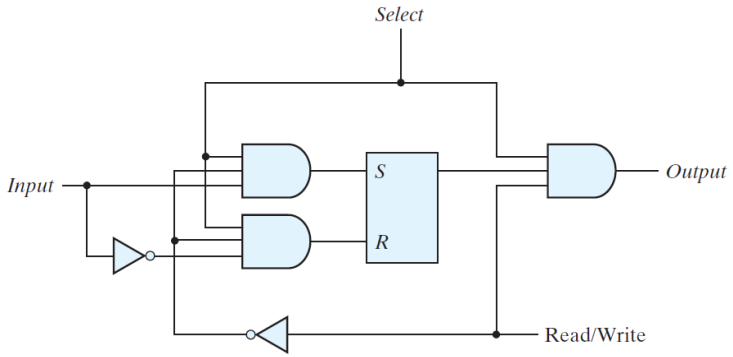


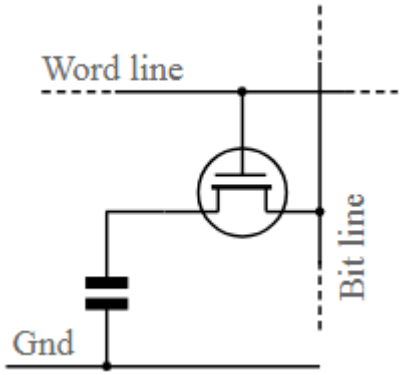
The **DRAM** (Dynamic RAM) cell contains a single MOS transistor and a capacitor. The charge stored on the capacitor discharges with time, and the memory cells must be periodically recharged by refreshing the memory.



Because of their simple cell structure, DRAMs typically have four times the density of SRAMs. This allows four times as much memory capacity to be placed on a given size of chip. The cost per bit of DRAM storage is three to four times less than that of SRAM storage. A further cost savings is realized because of the lower power requirement of DRAM cells. These advantages make DRAM the preferred technology for large memories in personal digital computers.

SRAM and DRAM





Dynamic RAM is constructed of tiny capacitors that leak electricity. DRAM requires a recharge every few milliseconds to maintain its data.

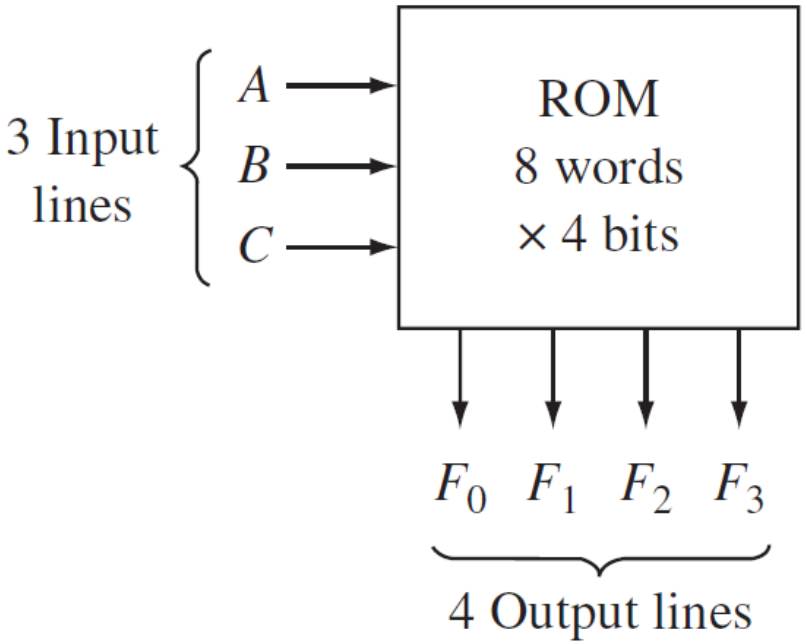
Static RAM technology, in contrast, holds its contents as long as power is available. SRAM consists of circuits similar to the D flip-flops.

SRAM is faster and much more expensive than DRAM; however, designers use DRAM because it is much denser (can store many bits per chip), uses less power, and generates less heat than SRAM.

For these reasons, both technologies are often used in combination: DRAM for main memory and SRAM for cache.

Read Only Memories (ROM)

A read-only memory (ROM) consists of an array of semiconductor devices that are interconnected to store an array of binary data. Once binary data is stored in the ROM, it can be read out whenever desired, but the data that is stored cannot be changed under normal operating conditions.



(a) Block diagram

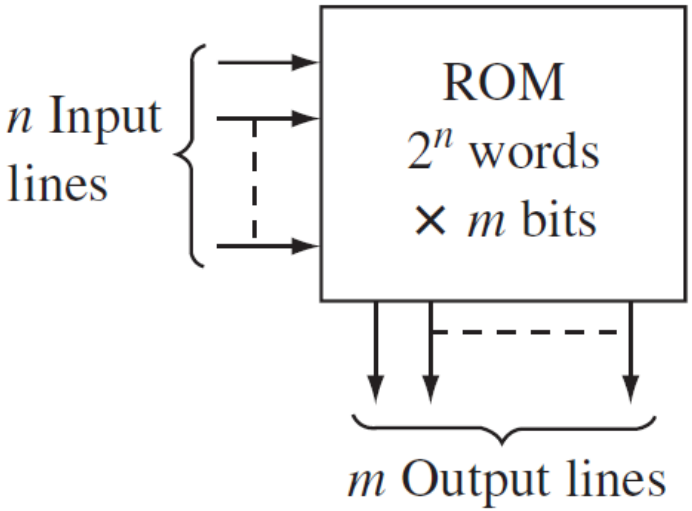
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Typical data stored in ROM (2^3 words of 4 bits each)

(b) Truth table for ROM

Read Only Memories (ROM)

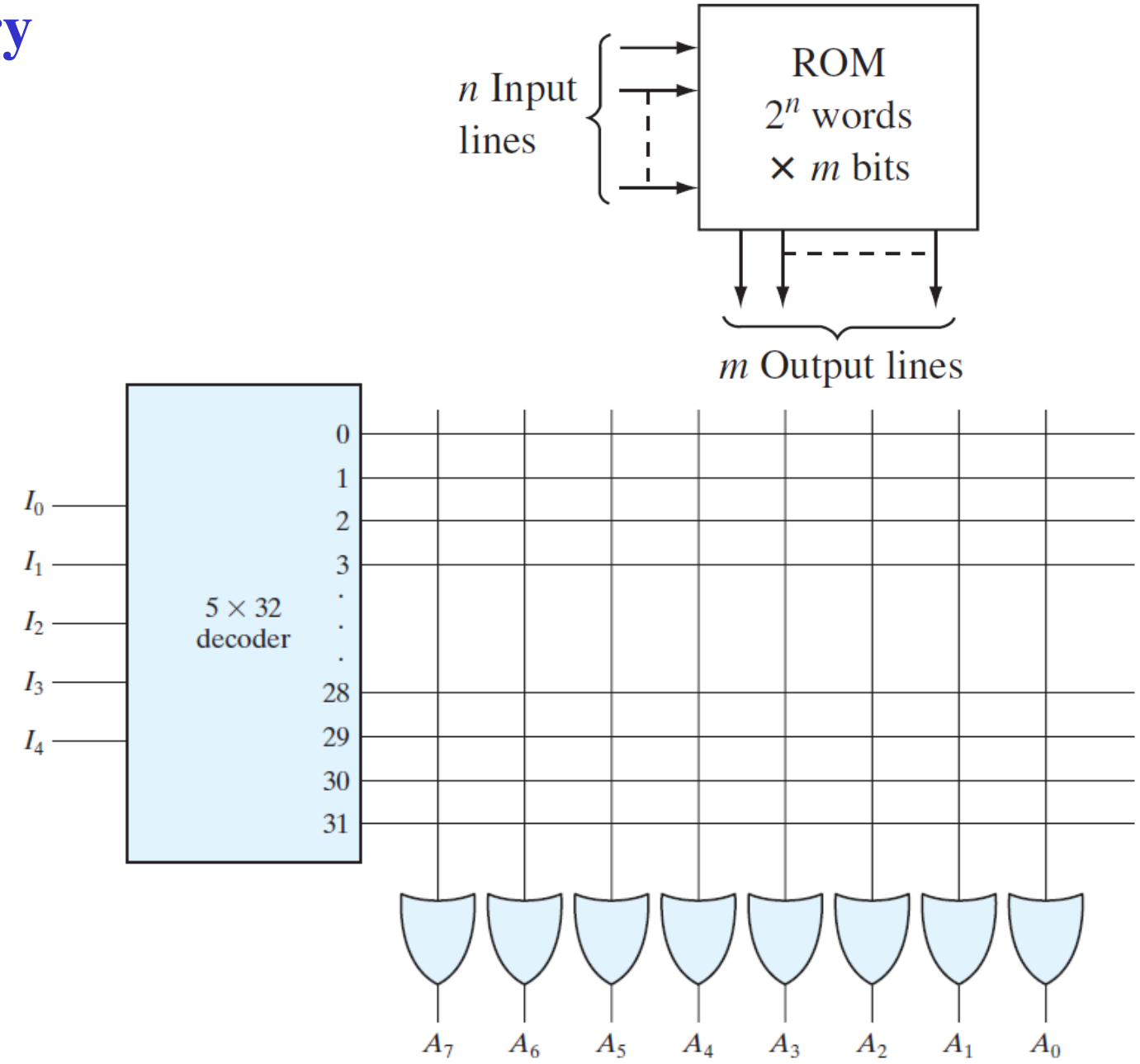
A ROM that has n input lines and m output lines contains an array of 2^n words, and each word is m bits long. The input lines serve as an address to select one of the 2^n words. When an input combination is applied to the ROM, the pattern of 0s and 1s stored in the corresponding word in the memory appears at the output lines.



n Input Variables	m Output Variables
00 · · · 00	100 · · · 110
00 · · · 01	010 · · · 111
00 · · · 10	101 · · · 101
00 · · · 11	110 · · · 010
⋮	⋮
11 · · · 00	001 · · · 011
11 · · · 01	110 · · · 110
11 · · · 10	011 · · · 000
11 · · · 11	111 · · · 101

Typical data array stored in ROM (2^n words of m bits each)

Read Only Memory (ROM)

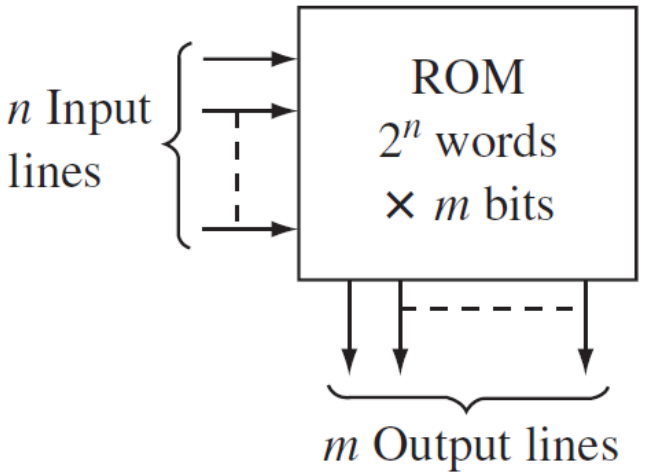


32x8 ROM

The 256 intersections are programmable. A programmable connection between two lines is logically equivalent to a switch that can be altered to be either closed (meaning that the two lines are connected) or open (meaning that the two lines are disconnected).

Read Only Memory (ROM)

A ROM basically consists of a decoder and a memory array. When a pattern of n 0s and 1s is applied to the decoder inputs, exactly one of the 2^n decoder outputs is 1. This decoder output line selects one of the words in the memory array, and the bit pattern stored in this word is transferred to the memory output lines.

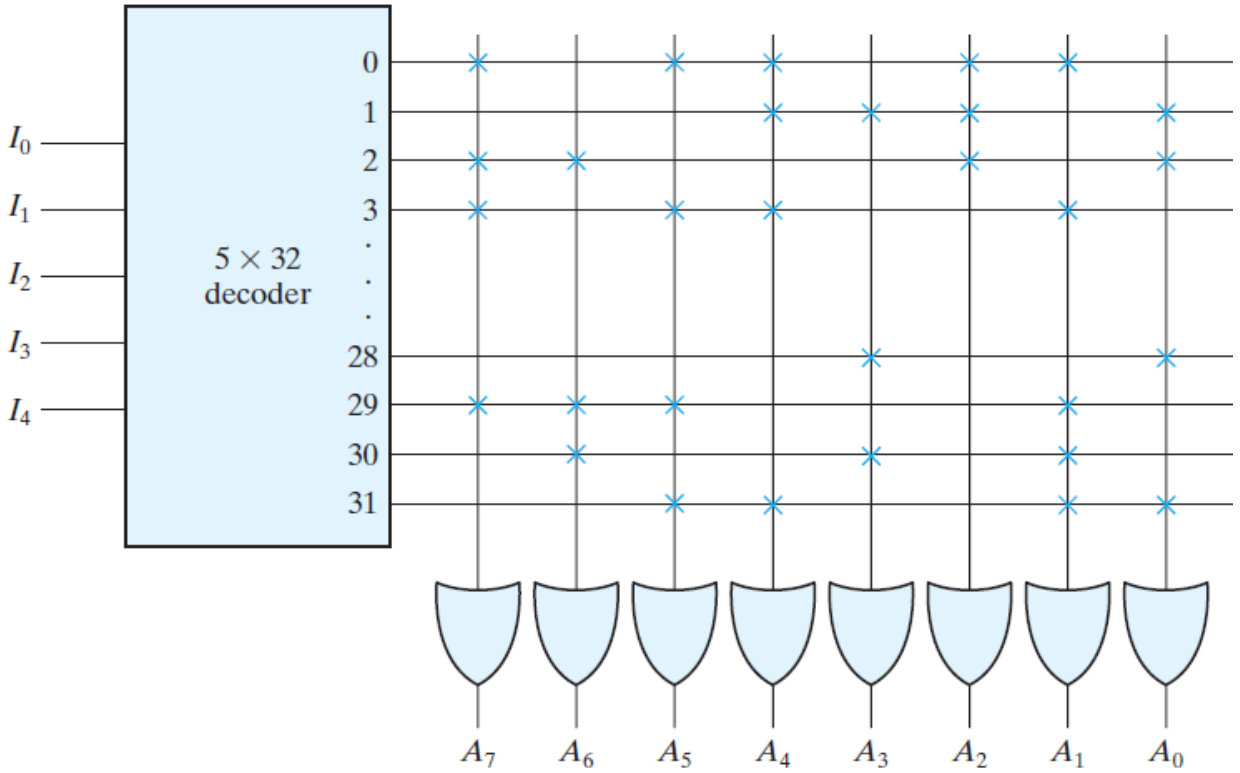


A ROM can implement any combinational circuit. Essentially, if the outputs for all combinations of inputs are stored in the ROM, the outputs can be "looked up" in the table stored in the ROM. The ROM method is also called the **Look-Up Table (LUT)** method for this reason.

Read Only Memory (ROM)

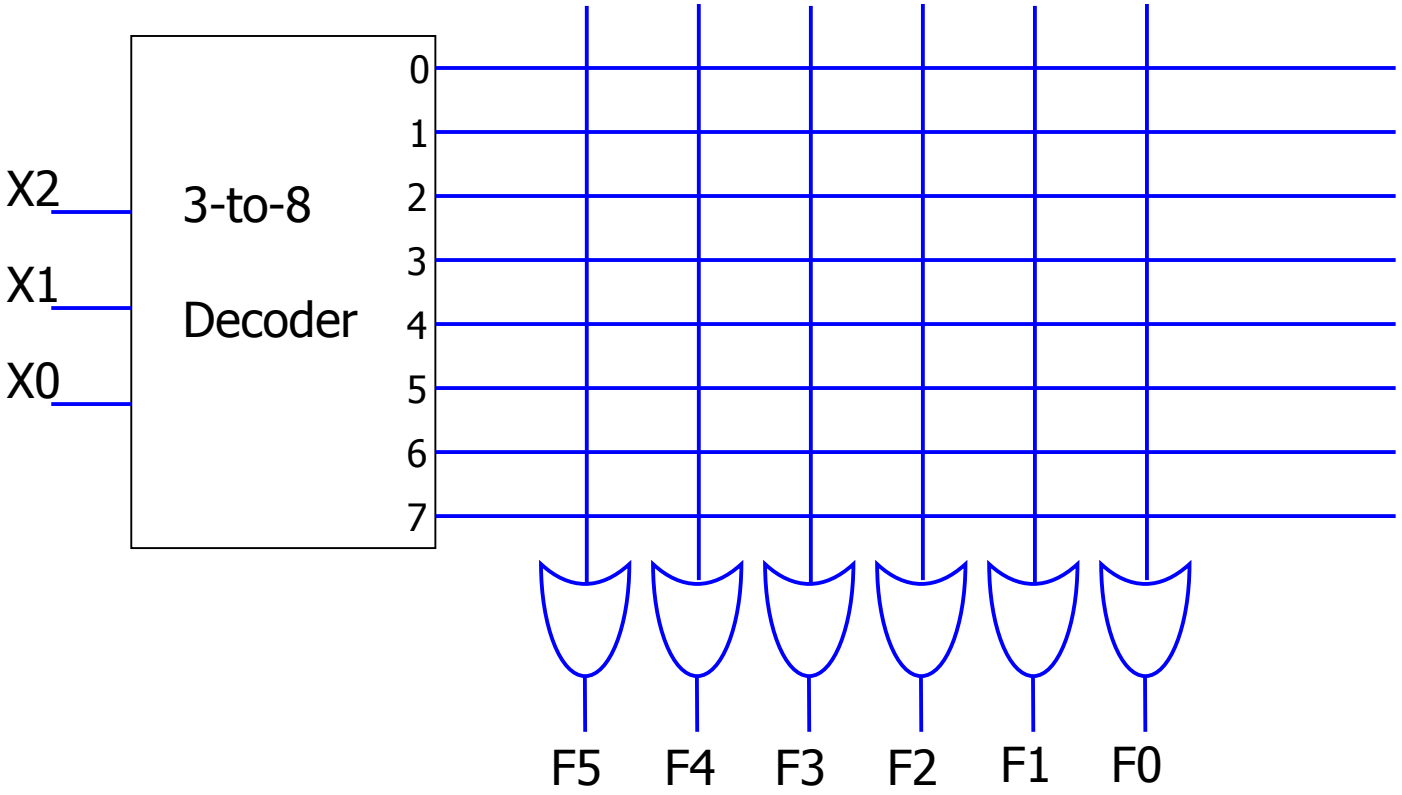
ROM Truth Table (Partial)

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



Read Only Memory (ROM): Square Lookup Table

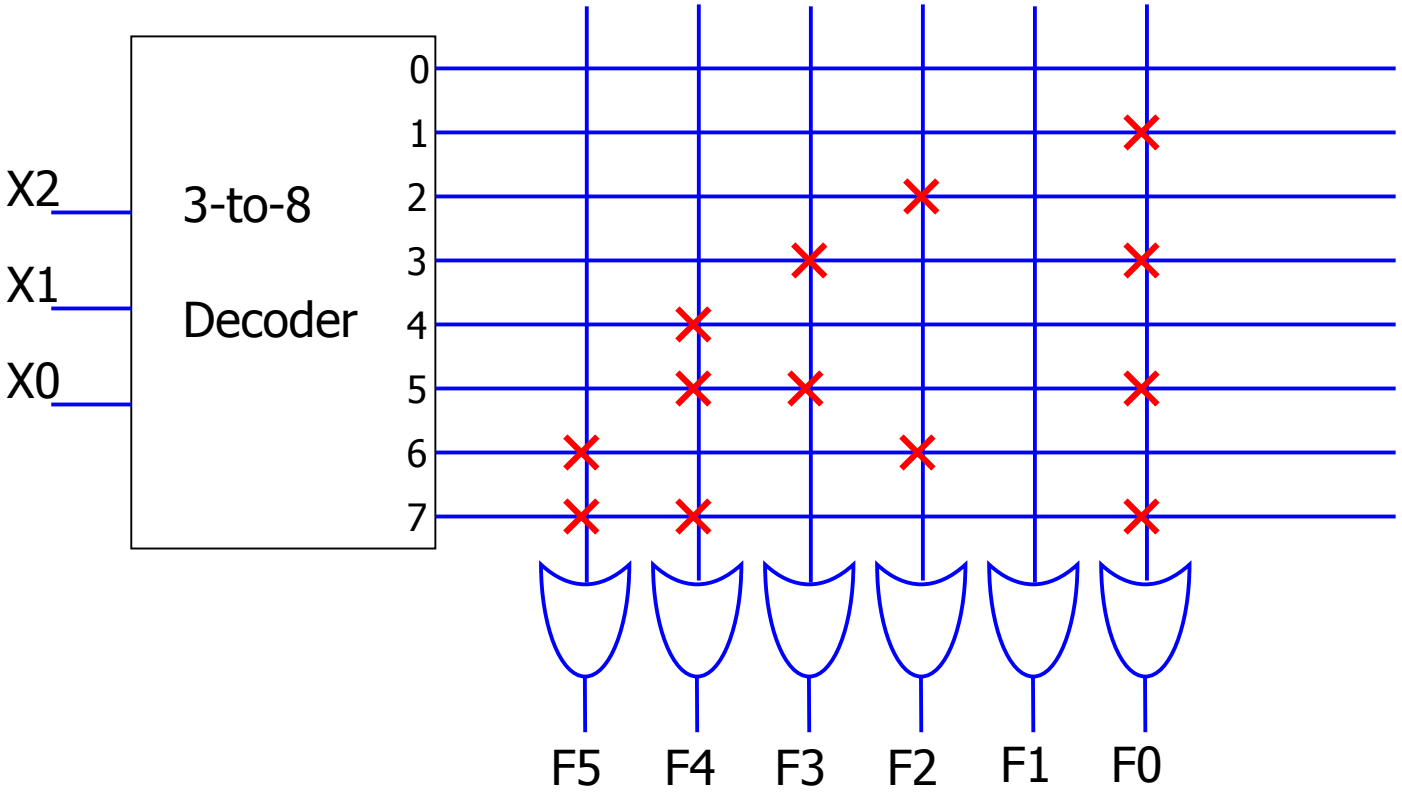
X	$F(X)=X^2$
000	000000
001	000001
010	000100
011	001001
100	010000
101	011001
110	100100
111	110001



Output contains 6 bits

Read Only Memory (ROM): Square Lookup Table

X	$F(X)=X^2$
000	000000
001	000001
010	000100
011	001001
100	010000
101	011001
110	100100
111	110001

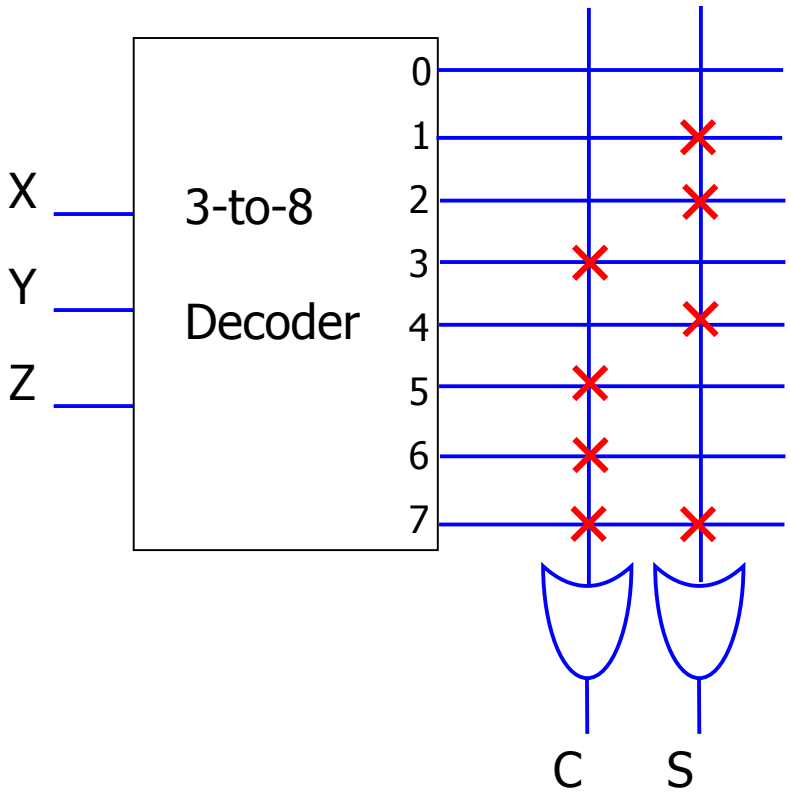


Output contains 6 bits

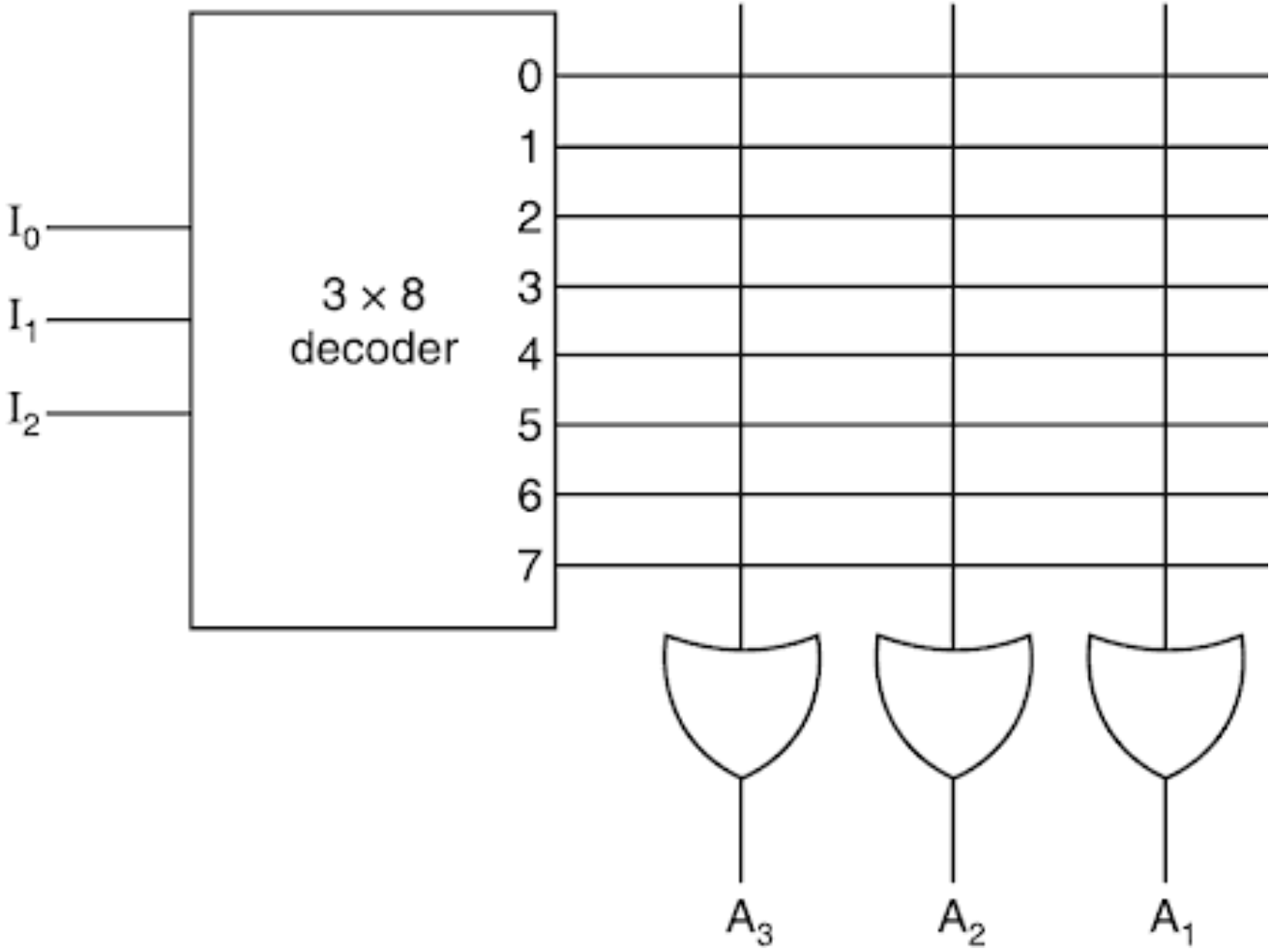
Read Only Memory (ROM): A Full Adder

Truth Table of Full Adder

Inputs			Outputs	
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

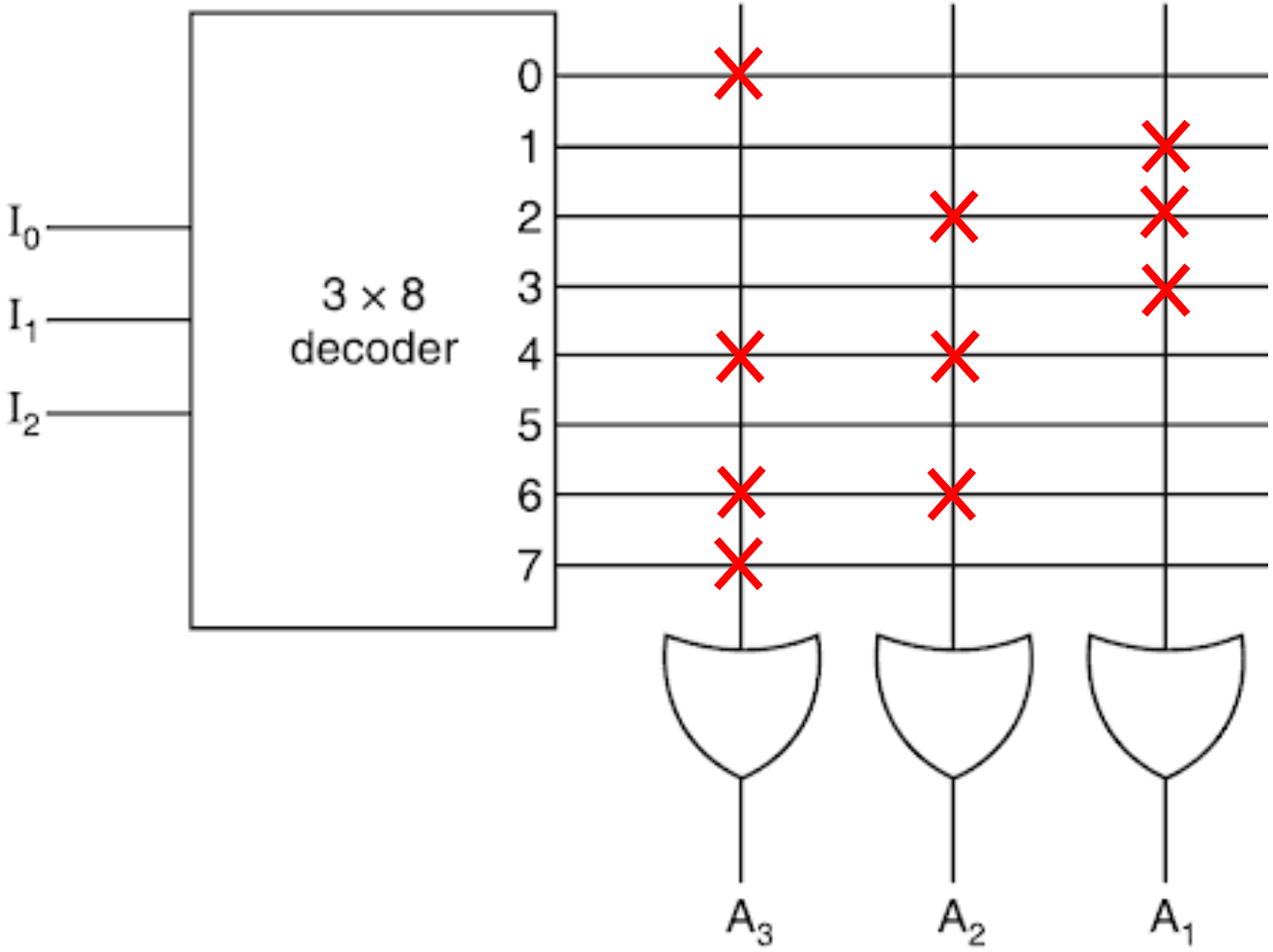


Read Only Memory (ROM): Another example



$$A_1 = \sum m(1,2,3)$$
$$A_2 = \sum m(2,4,6)$$
$$A_3 = \sum m(0,4,6,7)$$

Read Only Memory (ROM): Another example



$$A_1 = \sum m(1,2,3)$$
$$A_2 = \sum m(2,4,6)$$
$$A_3 = \sum m(0,4,6,7)$$

Types of ROMs

The required paths in a ROM may be programmed in four different ways.

Mask programming is done by the semiconductor company during the last fabrication process of the unit. The procedure for fabricating a ROM requires that the customer fill out the truth table he or she wishes the ROM to satisfy. The truth table may be submitted in a special form provided by the manufacturer or in a specified format on a computer output medium. The manufacturer makes the corresponding mask for the paths to produce the 1's and 0's according to the customer's truth table. Mask programming is economical only if a large quantity of the same ROM configuration is to be ordered.

Programmable ROM (PROM). When ordered, PROM units contain all the fuses intact, giving all 1's in the bits of the stored words. The fuses in the PROM are blown by the application of a high-voltage pulse to the device through a special pin. A blown fuse defines a binary 0 state and an intact fuse gives a binary 1 state. Special instruments called PROM programmers are available commercially to facilitate the procedure. In any case, all procedures for programming ROMs are hardware procedures, even though the word *programming* is used.

Types of ROMs

The required paths in a ROM may be programmed in four different ways.

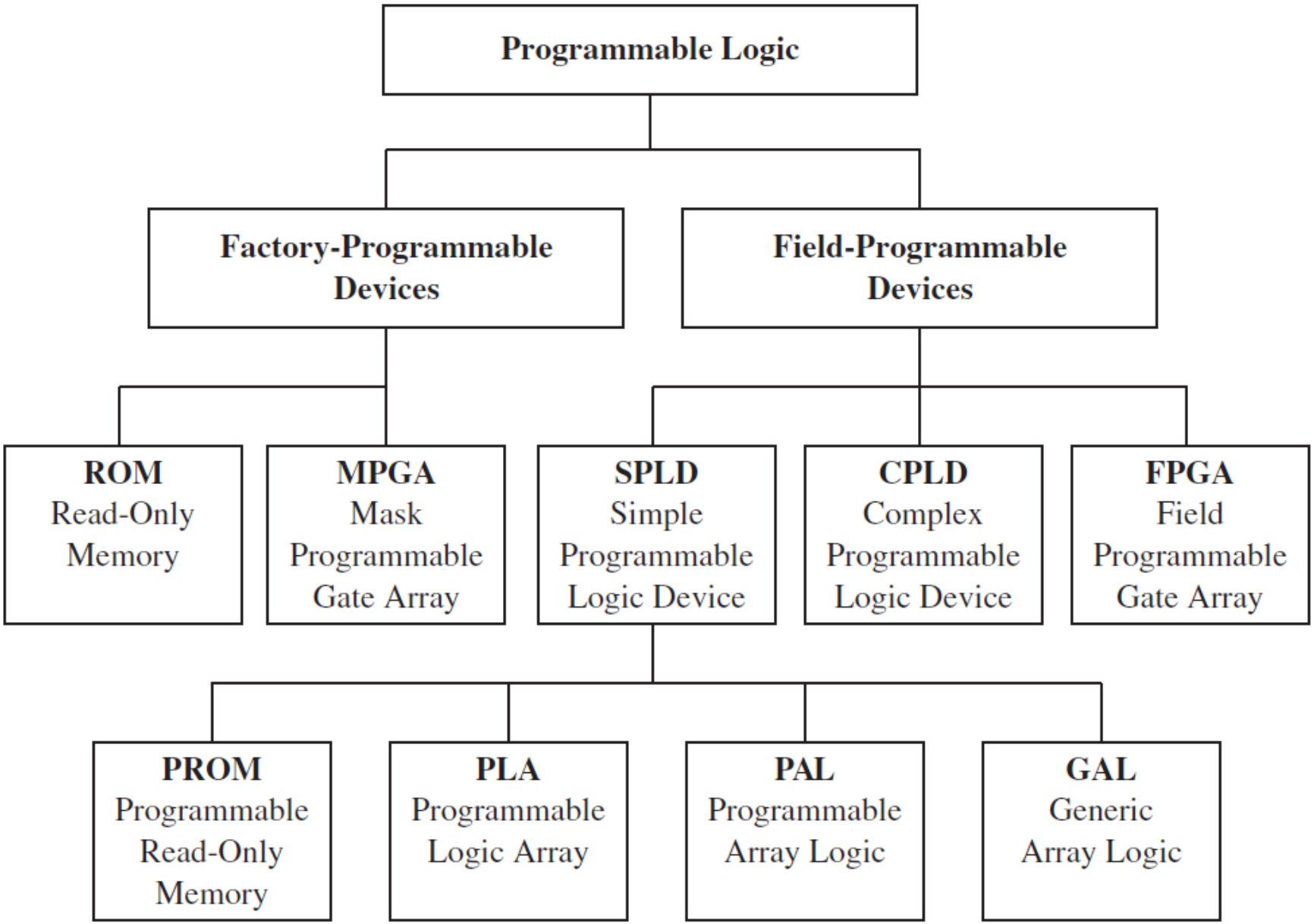
PROMs is irreversible, and once programmed, the fixed pattern is permanent and cannot be altered. Once a bit pattern has been established, the unit must be discarded if the bit pattern is to be changed.

Erasable PROM (EPROM) can be restructured to the initial state even though it has been programmed previously. When the EPROM is placed under a special ultraviolet light for a given length of time, the short wave radiation discharges the internal floating gates that serve as the programmed connections. After erasure, the EPROM returns to its initial state and can be reprogrammed to a new set of values.

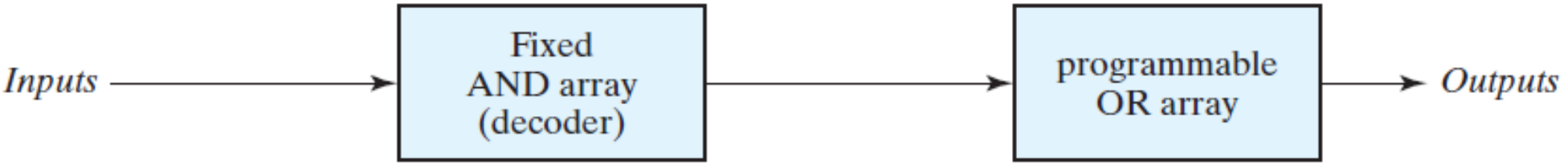
Electrically erasable PROM (EEPROM or E^2PROM) is like the EPROM, except that the previously programmed connections can be erased with an electrical signal instead of ultraviolet light. The advantage is that the device can be erased without removing it from its socket.

Flash memory devices are similar to EEPROMs. Like EEPROMs, flash memories are subject to fatigue, typically having about 10^5 block erase cycles.

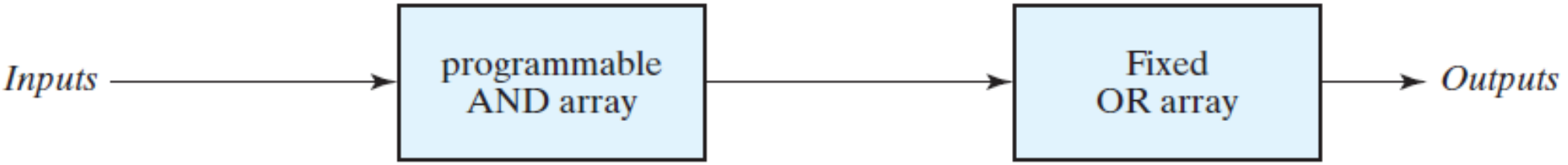
Programmable Logic Devices



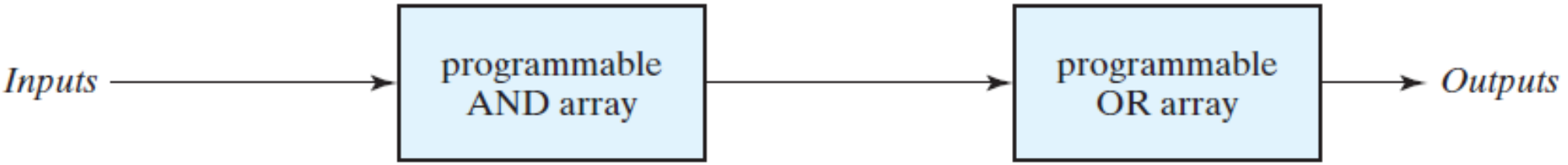
Programmable Logic Devices



(a) Programmable read-only memory (PROM)



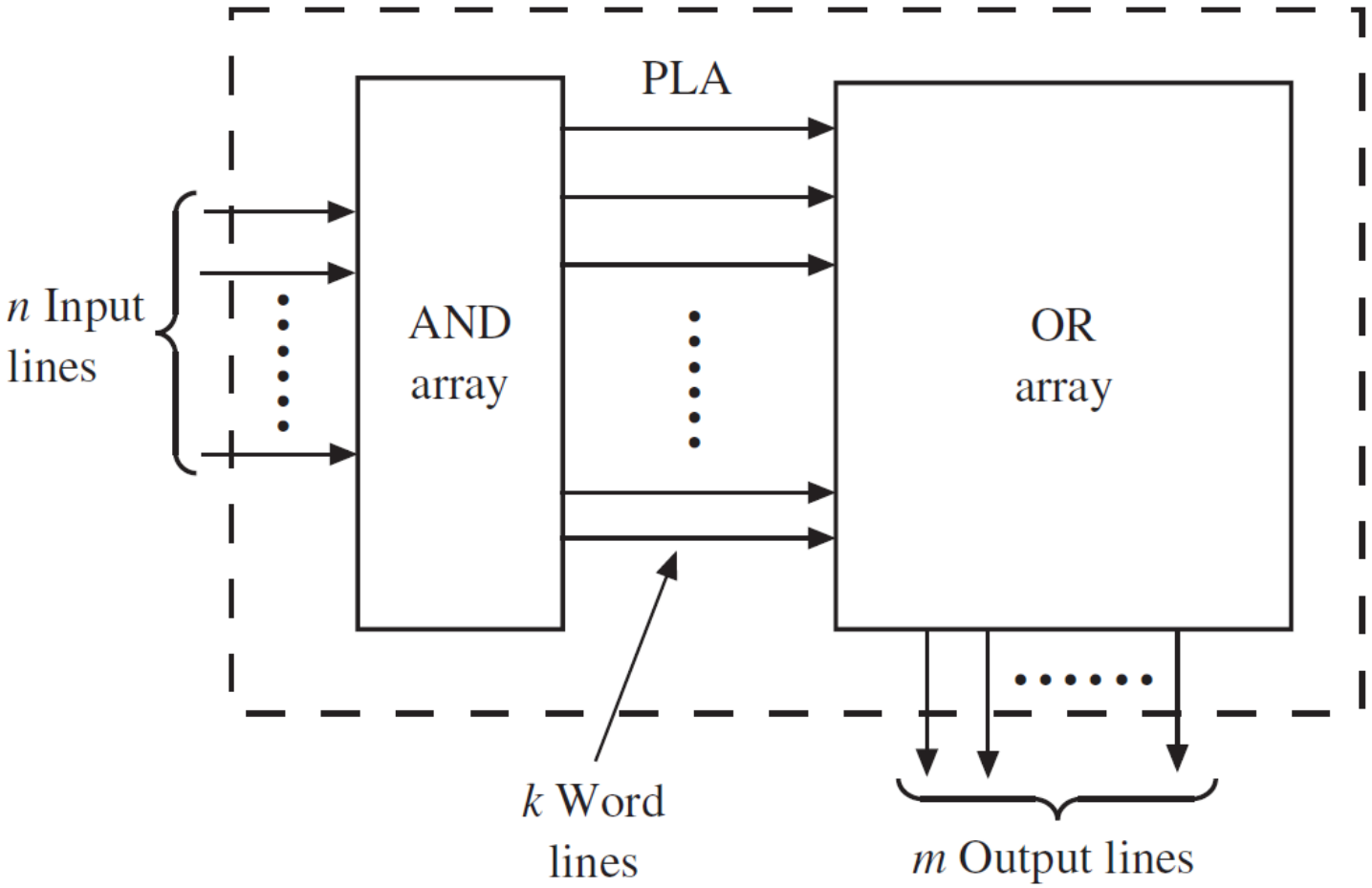
(b) Programmable array logic (PAL)



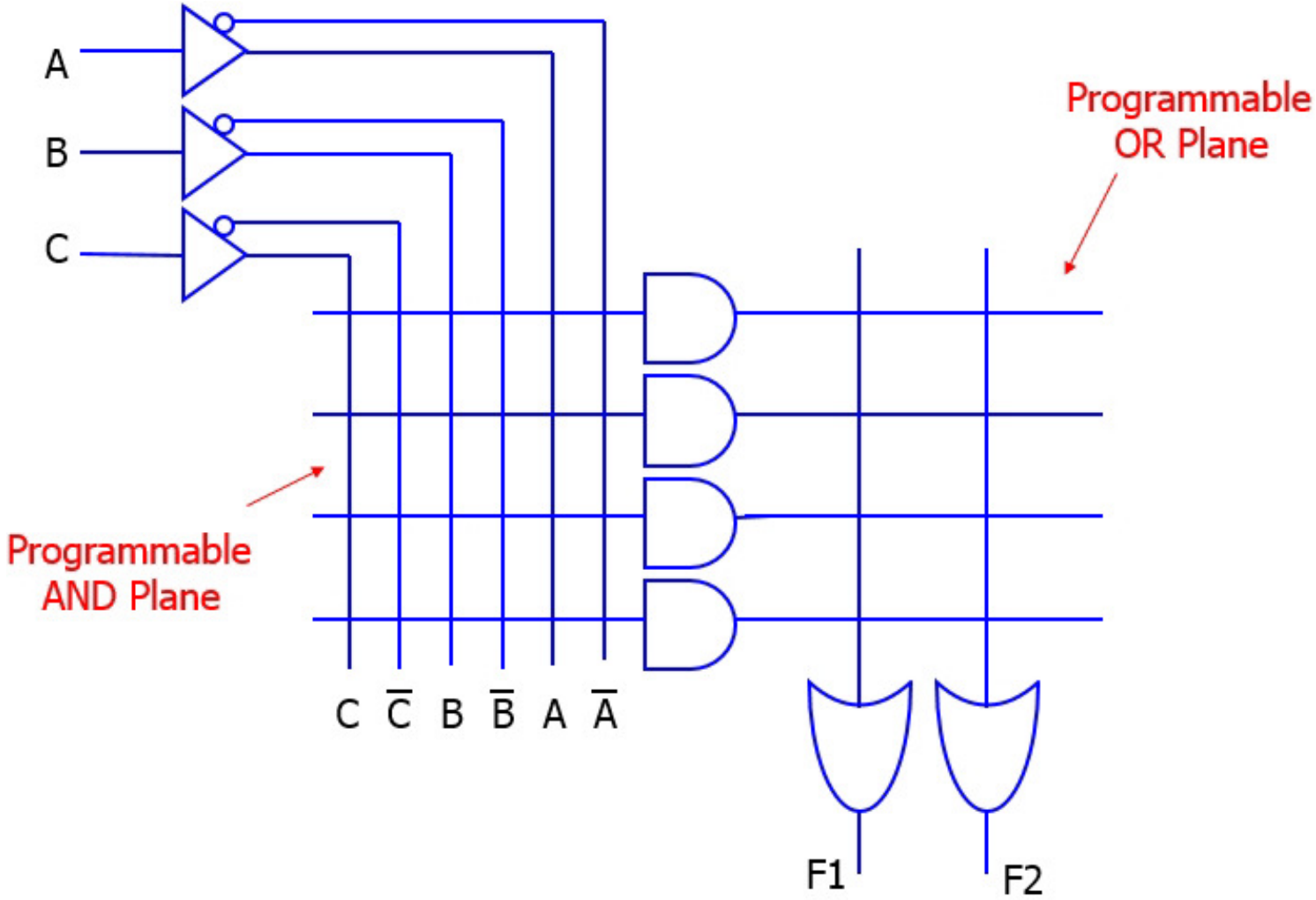
(c) Programmable logic array (PLA)

Programmable Logic Array (PLA)

A PLA with n inputs and m outputs can realize m functions of n variables.

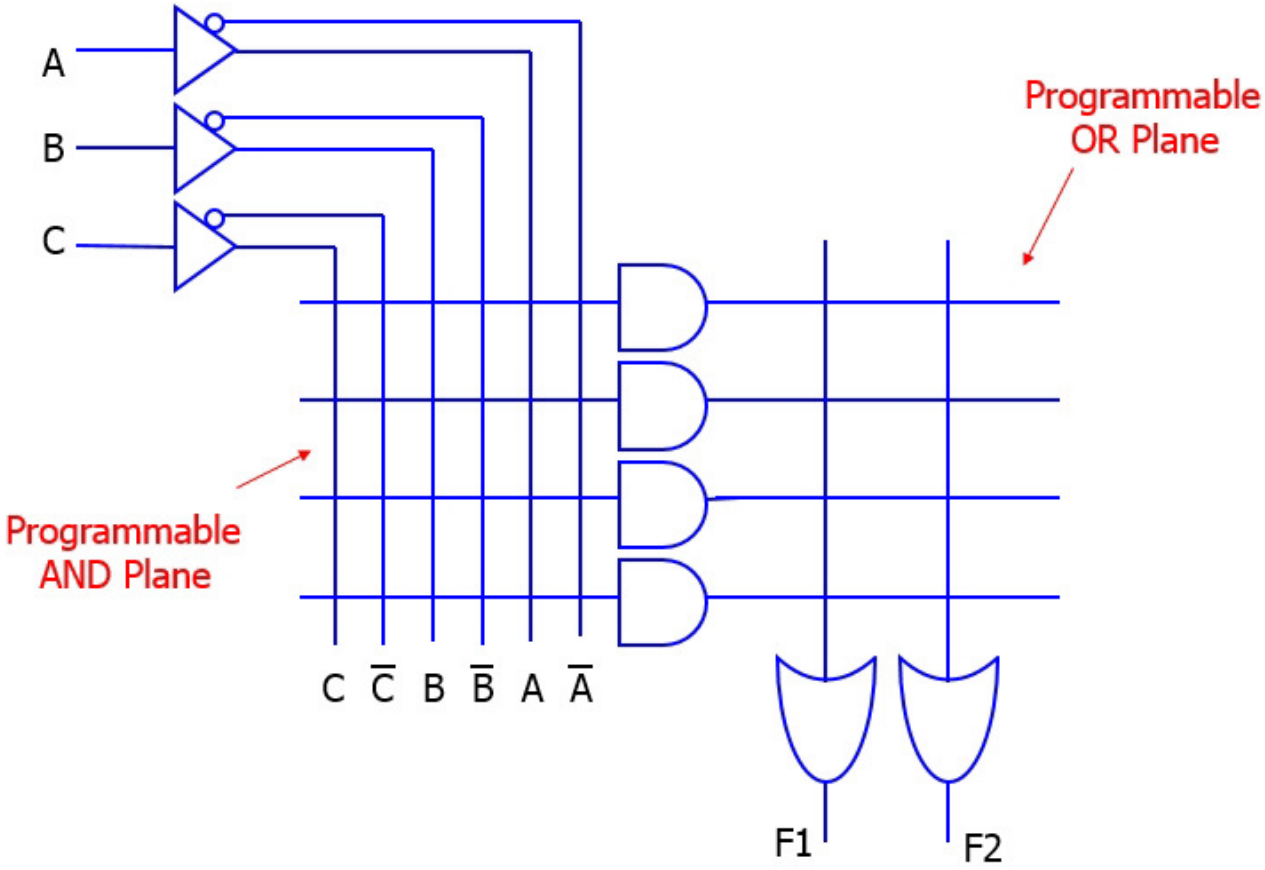


Programmable Logic Array (PLA)



Programmable Logic Array (PLA): An example

$F_1(A,B,C) = \sum m(0,2,4,7)$ $F_2(A,B,C) = \sum m(0,1,2,7)$



A \ BC	00	01	11	10
0	1			1
1	1		1	

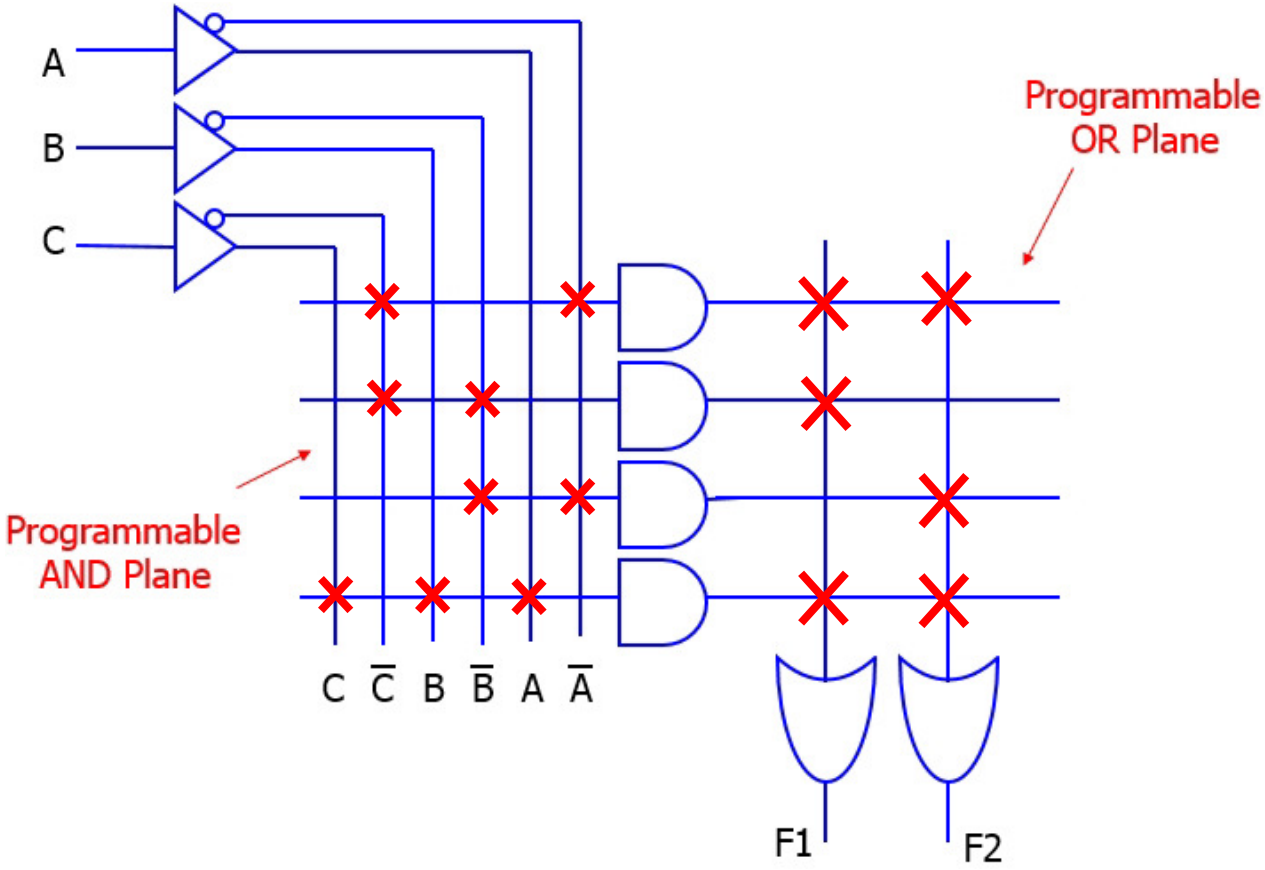
$$\sum m(0,2,4,7) = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$$

A \ BC	00	01	11	10
0	1	1		1
1			1	

$$\sum m(0,1,2,7) = \bar{A}\bar{B} + \bar{A}\bar{C} + ABC$$

Programmable Logic Array (PLA): An example

$F_1(A,B,C) = \sum m(0,2,4,7)$ $F_2(A,B,C) = \sum m(0,1,2,7)$



A \ BC	00	01	11	10
0	1			1
1	1		1	

$$\sum m(0,2,4,7) = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$$

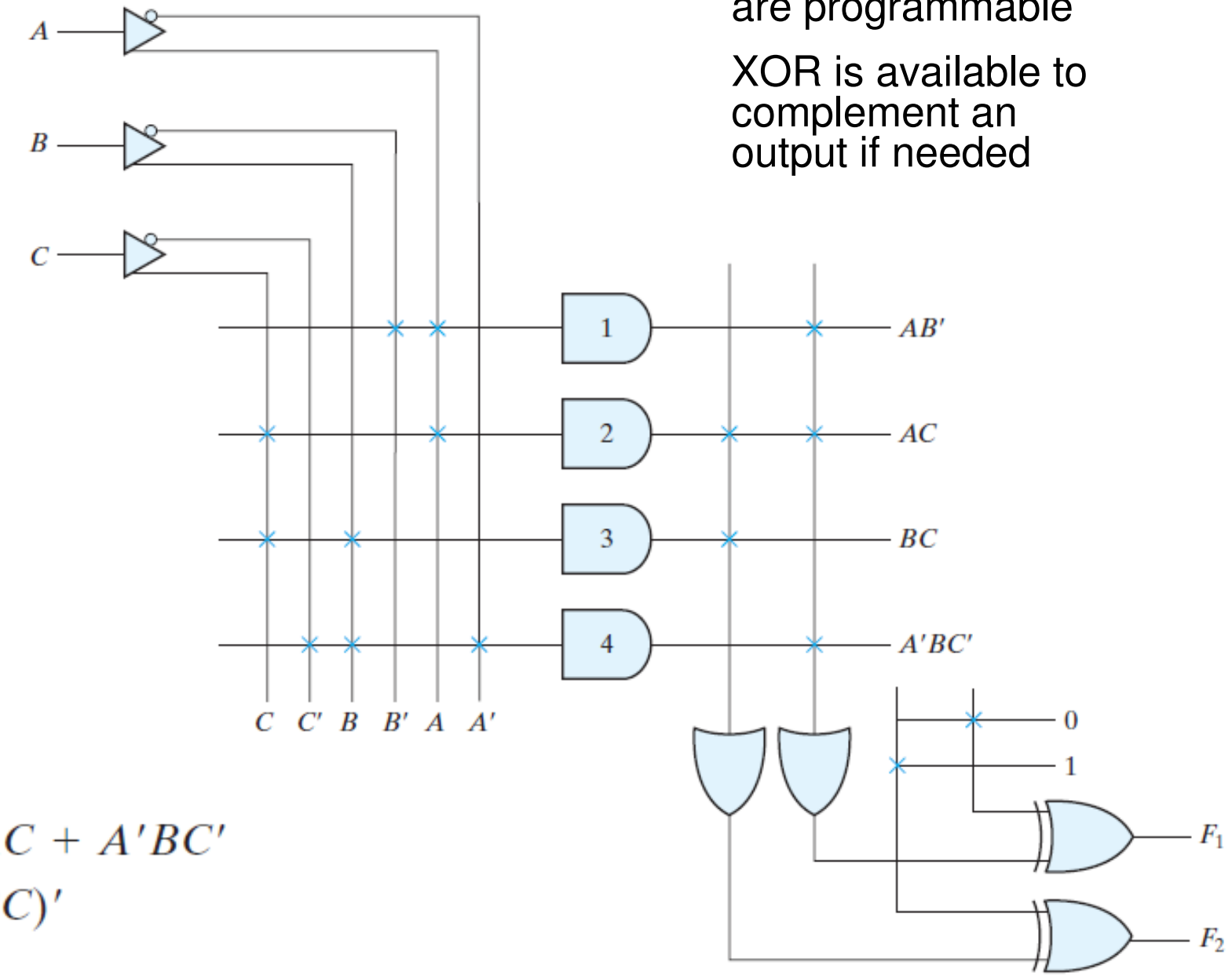
A \ BC	00	01	11	10
0	1	1		1
1			1	

$$\sum m(0,1,2,7) = \bar{A}\bar{B} + \bar{A}\bar{C} + ABC$$

Programmable Logic Array (PLA)

AND and OR arrays are programmable

XOR is available to complement an output if needed

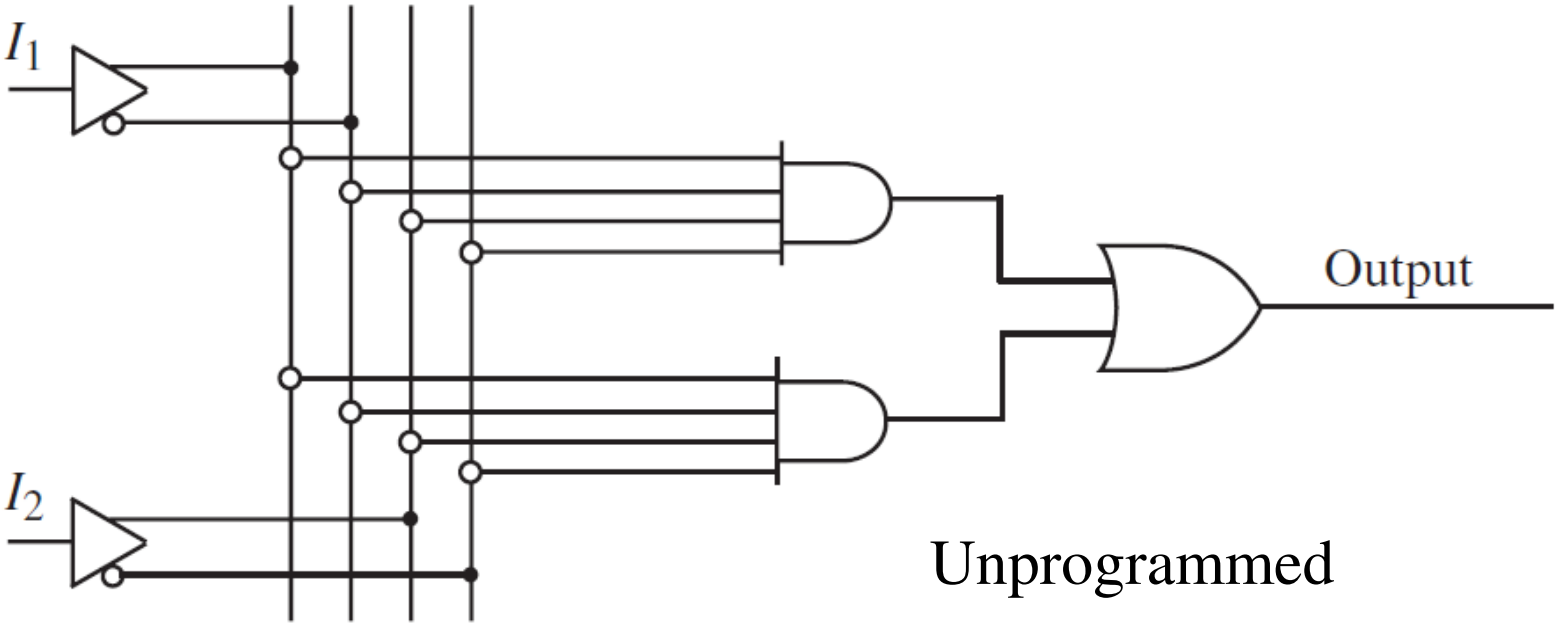


$$F_1 = AB' + AC + A'BC'$$

$$F_2 = (AC + BC)'$$

Programmable Array Logic (PAL)

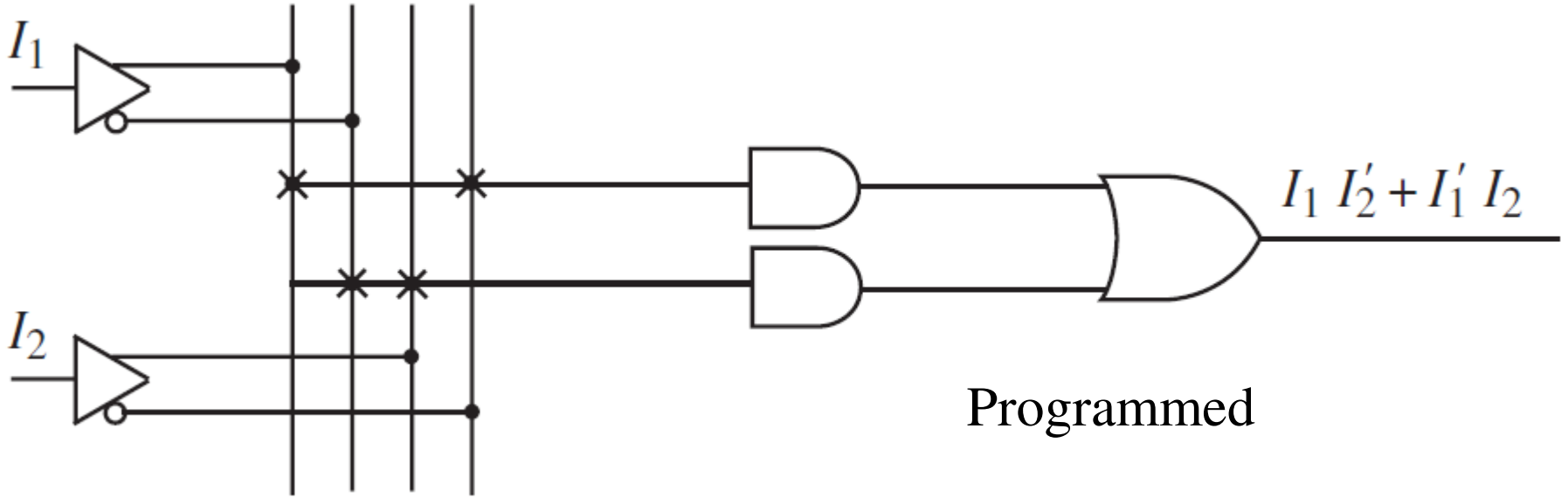
The PAL (programmable array logic) is a special case of the programmable logic array in which the AND array is programmable and the OR array is fixed. The basic structure of the PAL is the same as the PLA. Because only the AND array is programmable, the PAL is less expensive than the more general PLA, and the PAL is easier to program. For this reason, logic designers frequently use PALs to replace individual logic gates when several logic functions must be realized.



Unprogrammed

Programmable Array Logic (PAL)

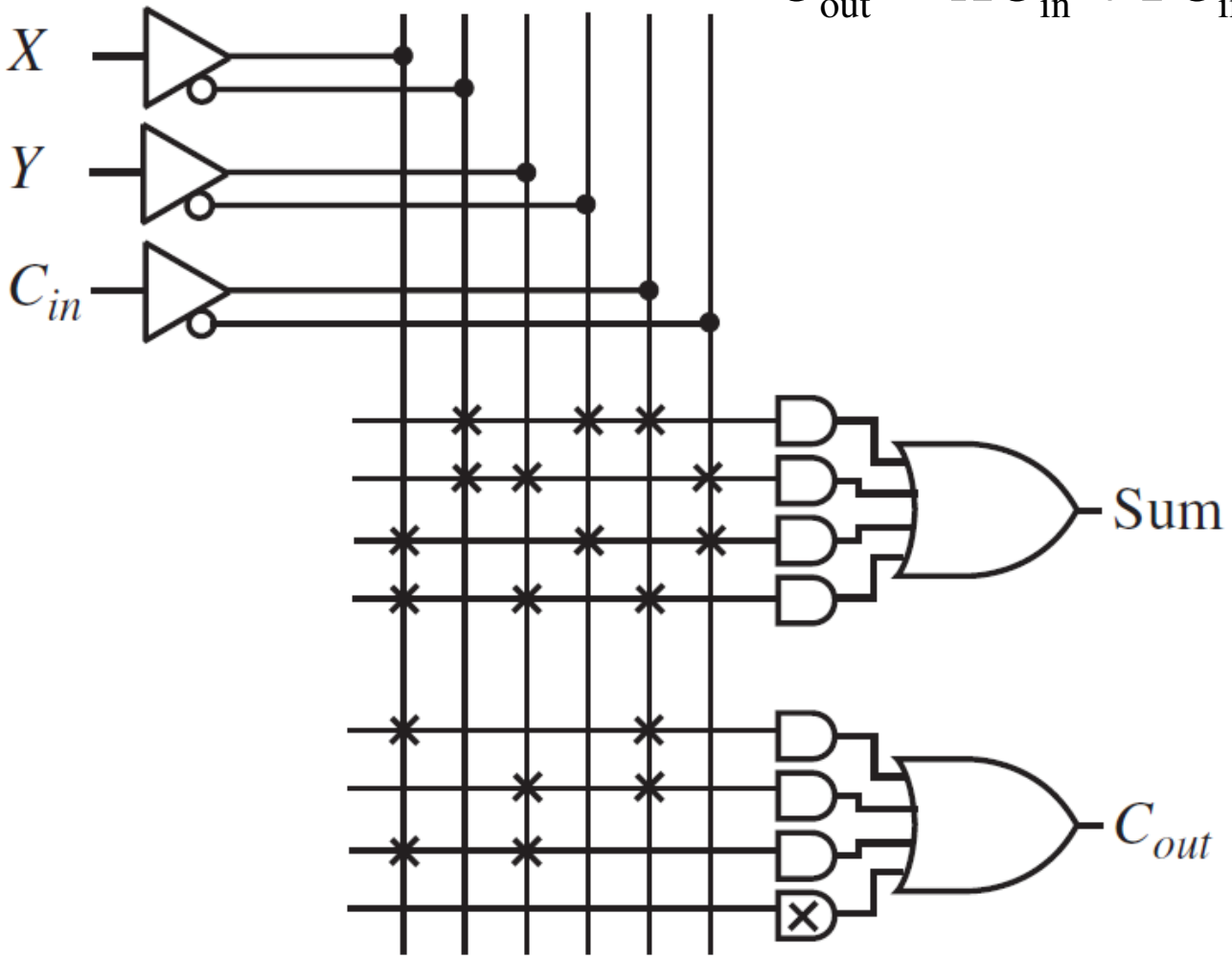
The PAL (programmable array logic) is a special case of the programmable logic array in which the AND array is programmable and the OR array is fixed. The basic structure of the PAL is the same as the PLA. Because only the AND array is programmable, the PAL is less expensive than the more general PLA, and the PAL is easier to program. For this reason, logic designers frequently use PALs to replace individual logic gates when several logic functions must be realized.



Programmable Array Logic (PAL)

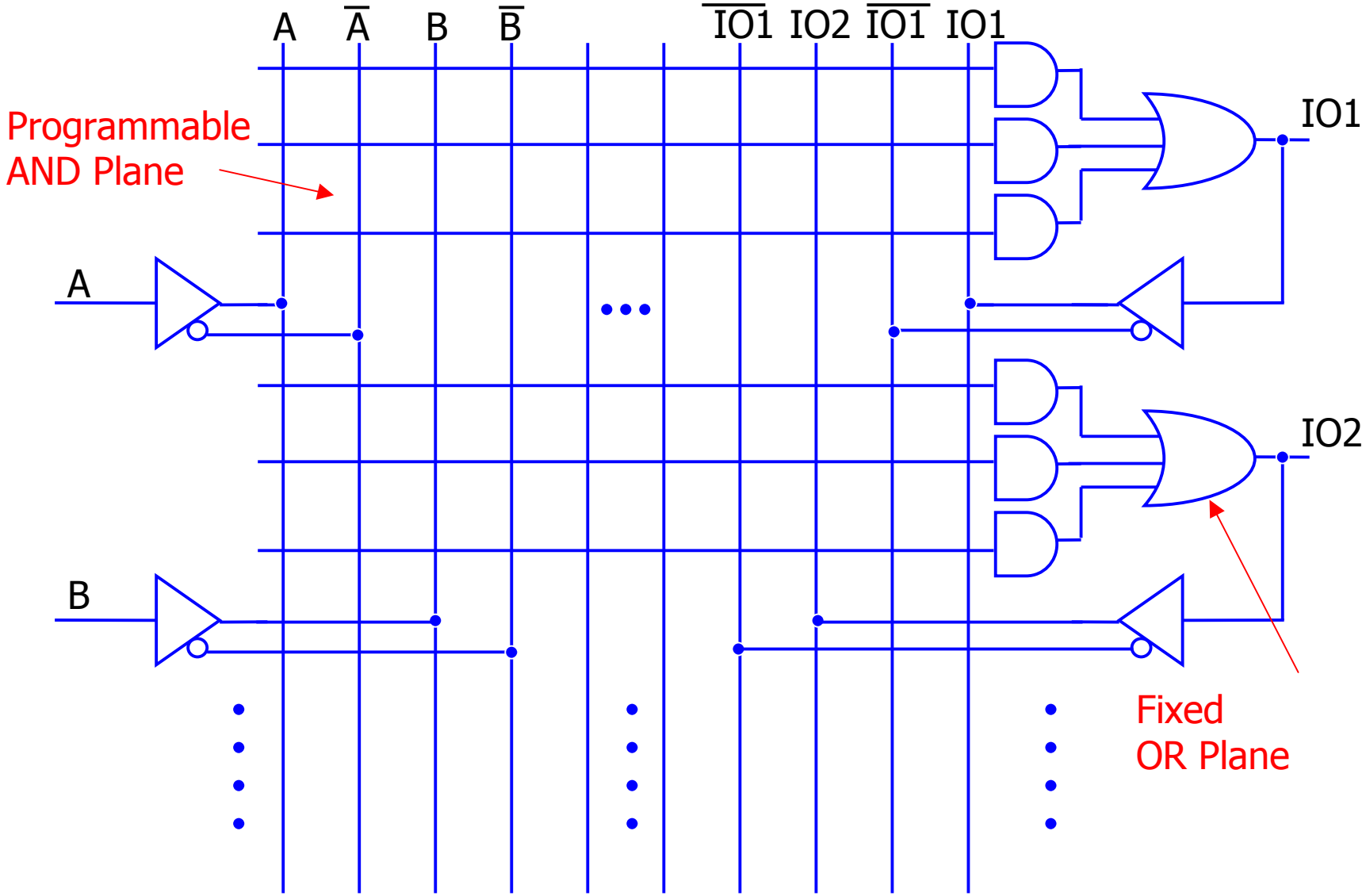
$$\text{Sum} = X'Y'C_{\text{in}} + X'YC'_{\text{in}} + XY'C'_{\text{in}} + XYC_{\text{in}}$$

$$C_{\text{out}} = XC_{\text{in}} + YC_{\text{in}} + XY$$

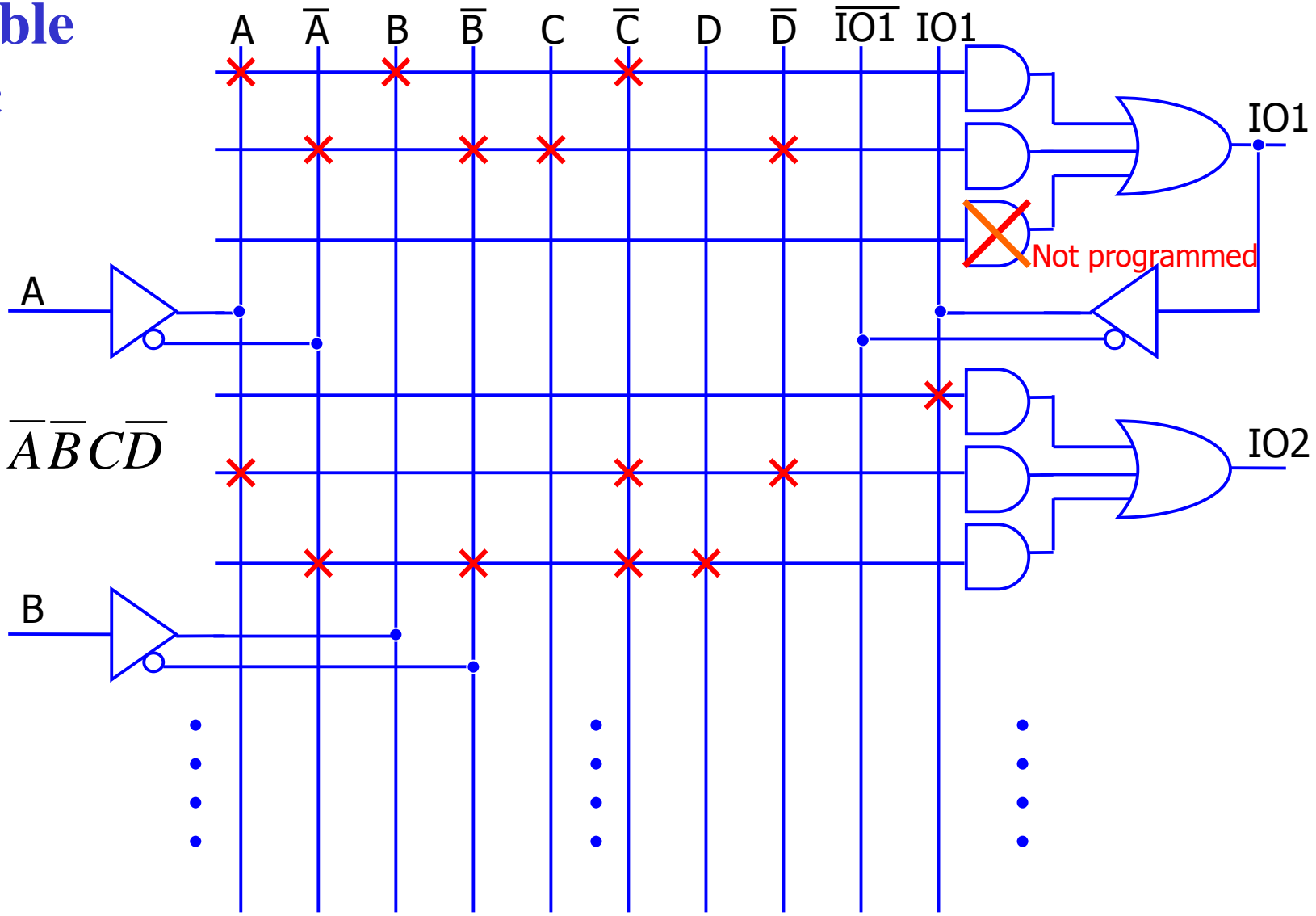


Programmable Array Logic (PAL)

A fixed OR array and a programmable AND array.



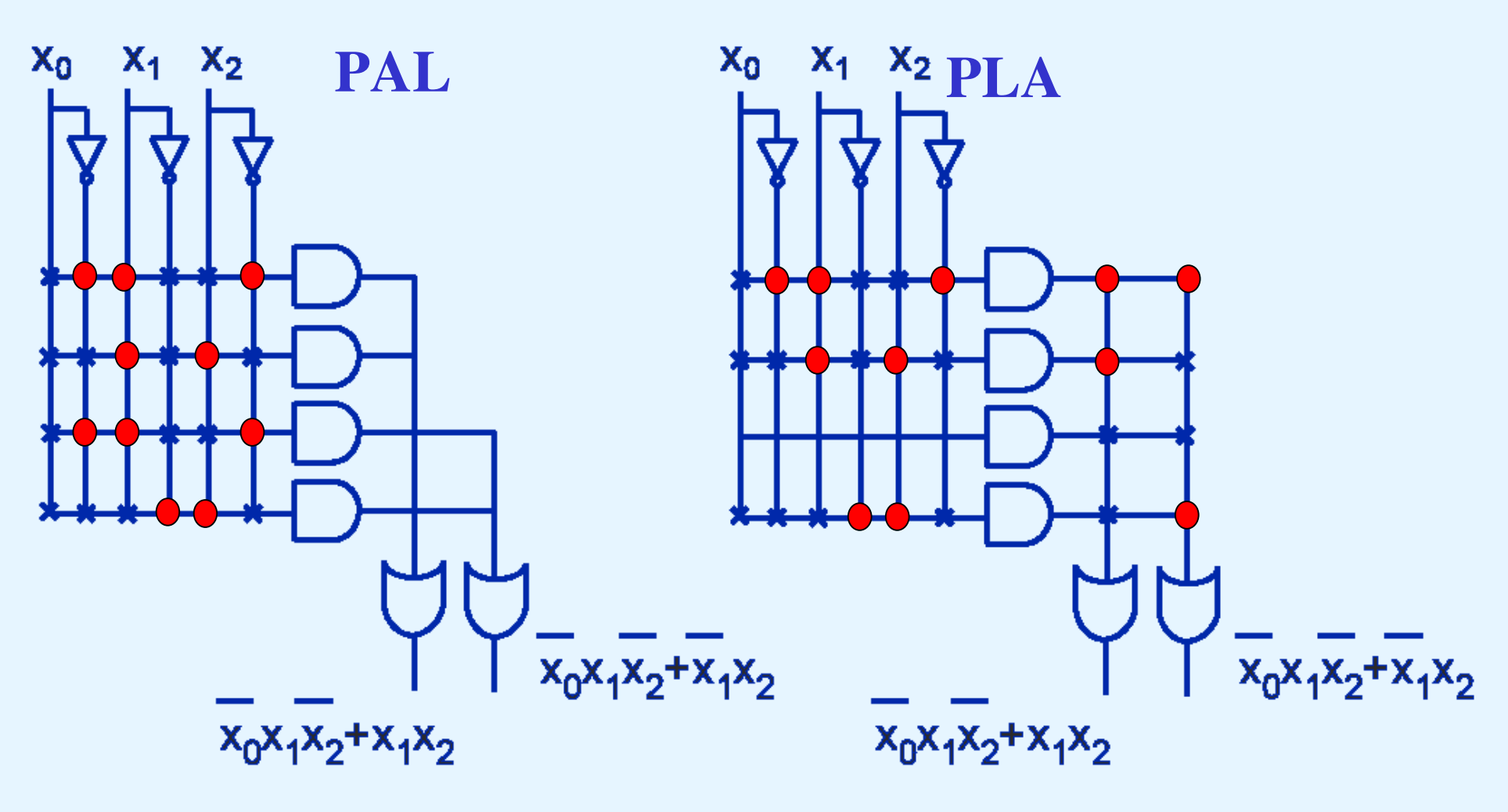
Programmable Array Logic (PAL)



$$IO1 = AB\overline{C} + \overline{A}\,\overline{B}C\overline{D}$$

$$IO2 = IO1 + A\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D = AB\overline{C} + \overline{A}\,\overline{B}C\overline{D} + A\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D$$

PAL vs. PLA

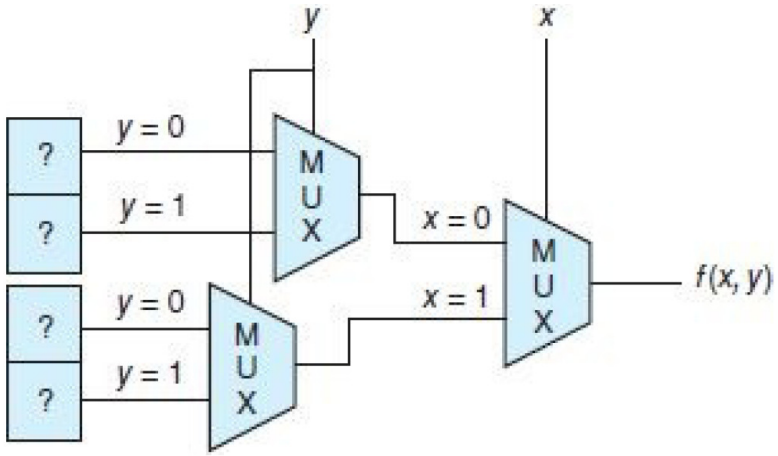


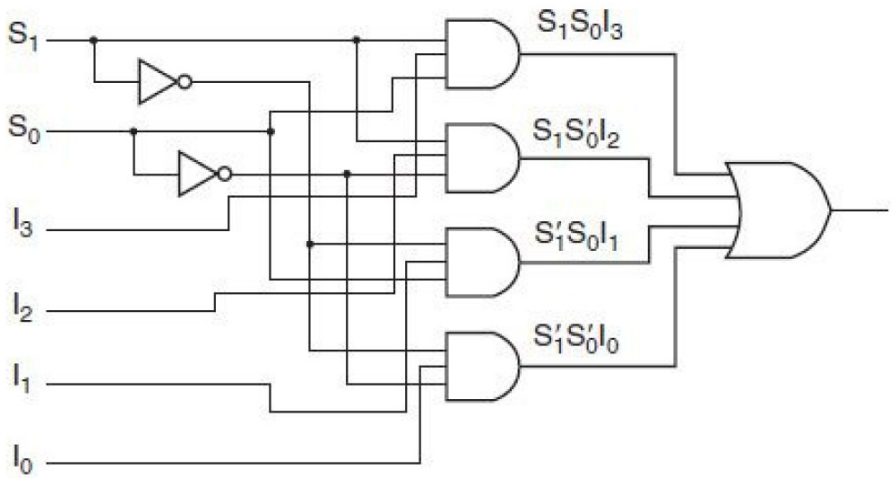
PLAs are more flexible than PALs, but they also tend to be slower and more costly than PALs.

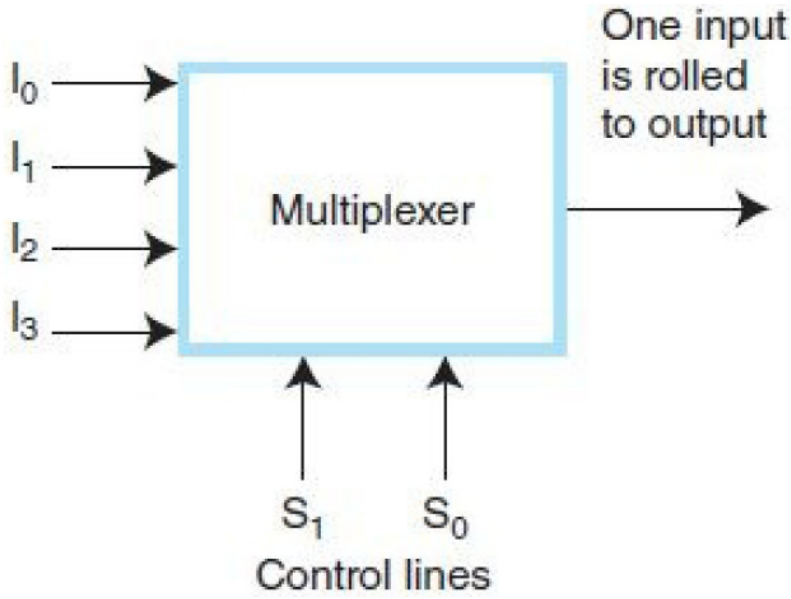
Field Programmable Gate Array (FPGA)

Provides programmable logic using lookup tables instead of changing the wiring of the chip.

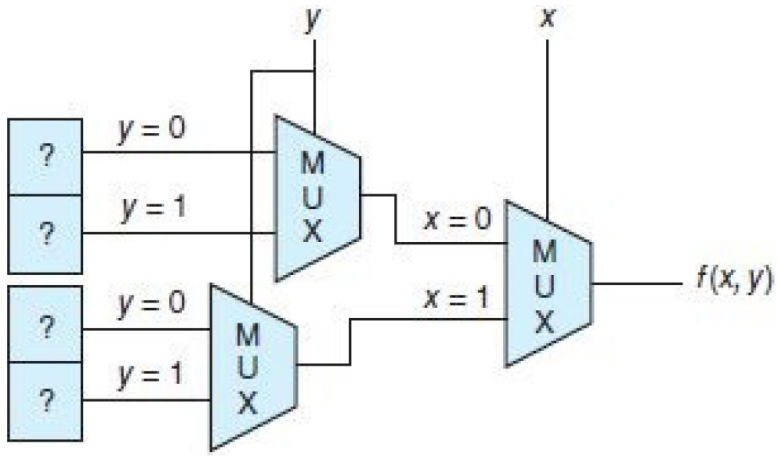
A typical FPGA logic element consists of memory cells and multiplexers. The inputs of multiplexers are selected according to to the values of of the logic function inputs, x and y,



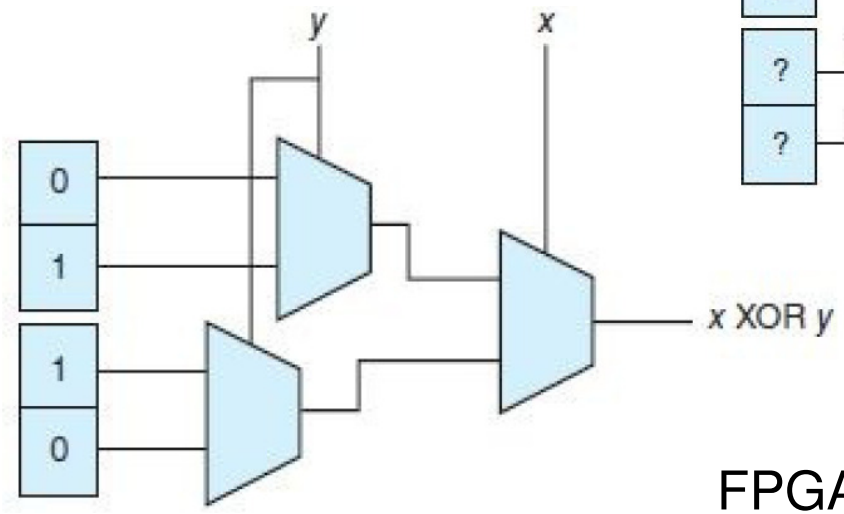




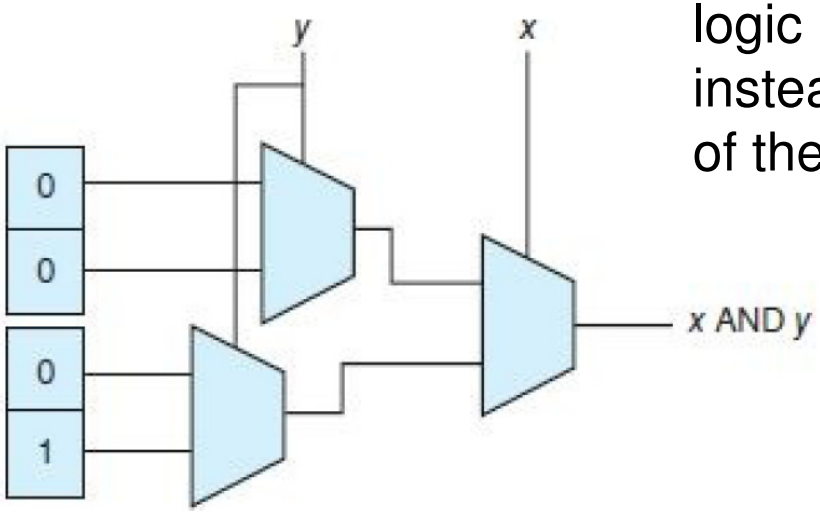
Field-Programmable Gate Array (FPGA)



x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0



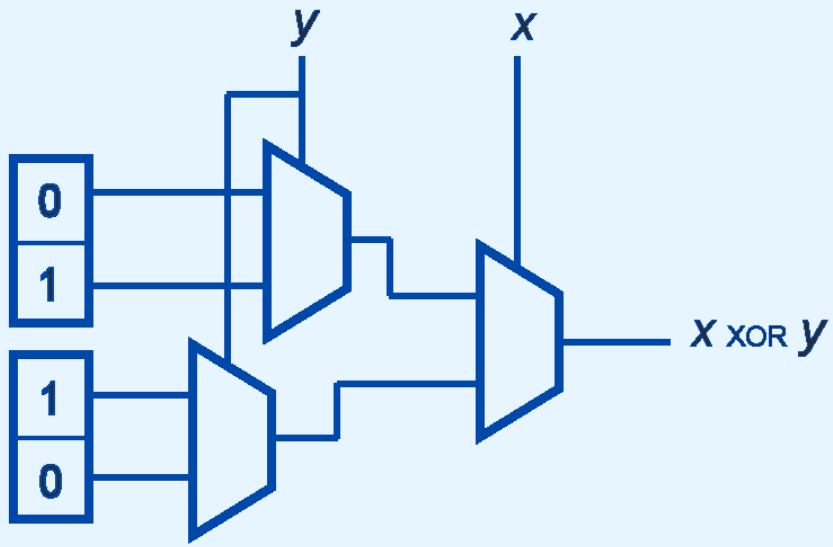
x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



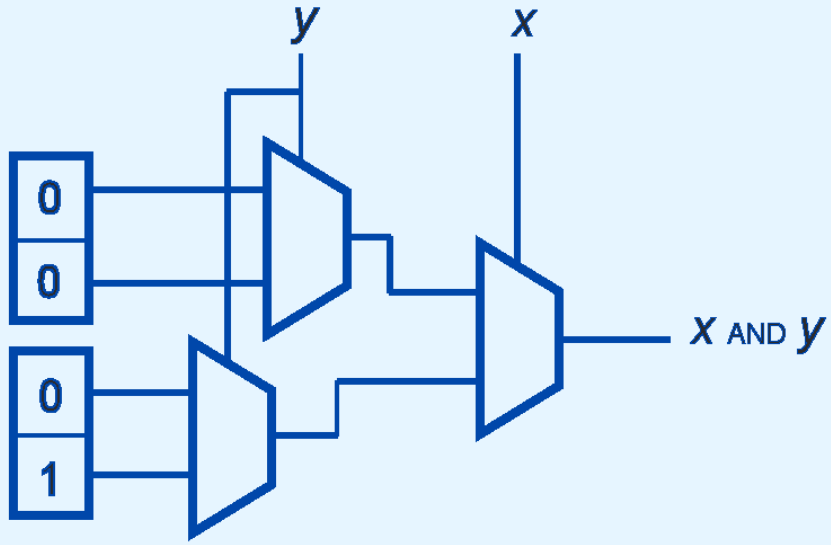
FPGA provides programmable logic using lookup tables instead of changing the wiring of the chip.

Field-Programmable Gate Array (FPGA)

x	y	x XOR y
0	0	0
0	1	1
1	0	1
1	1	0

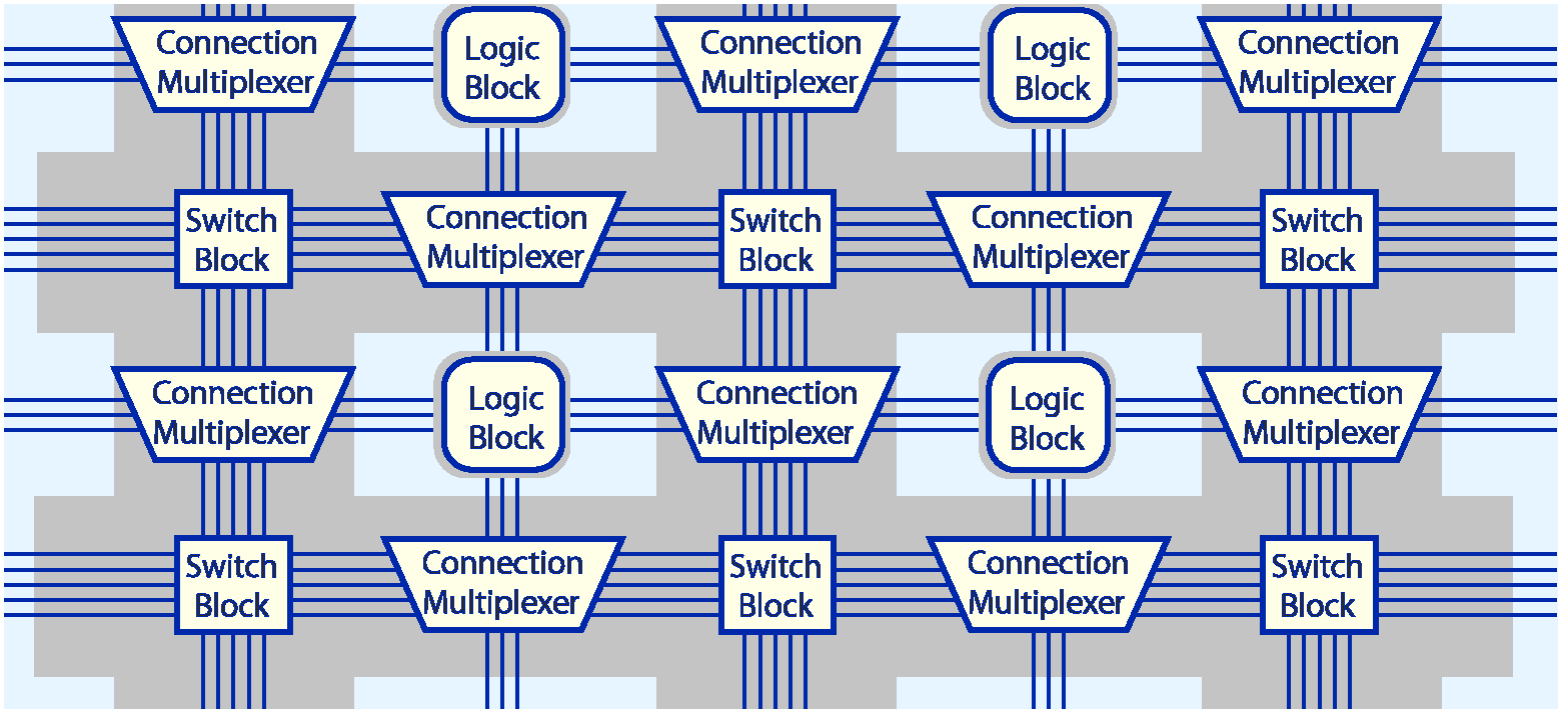


x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

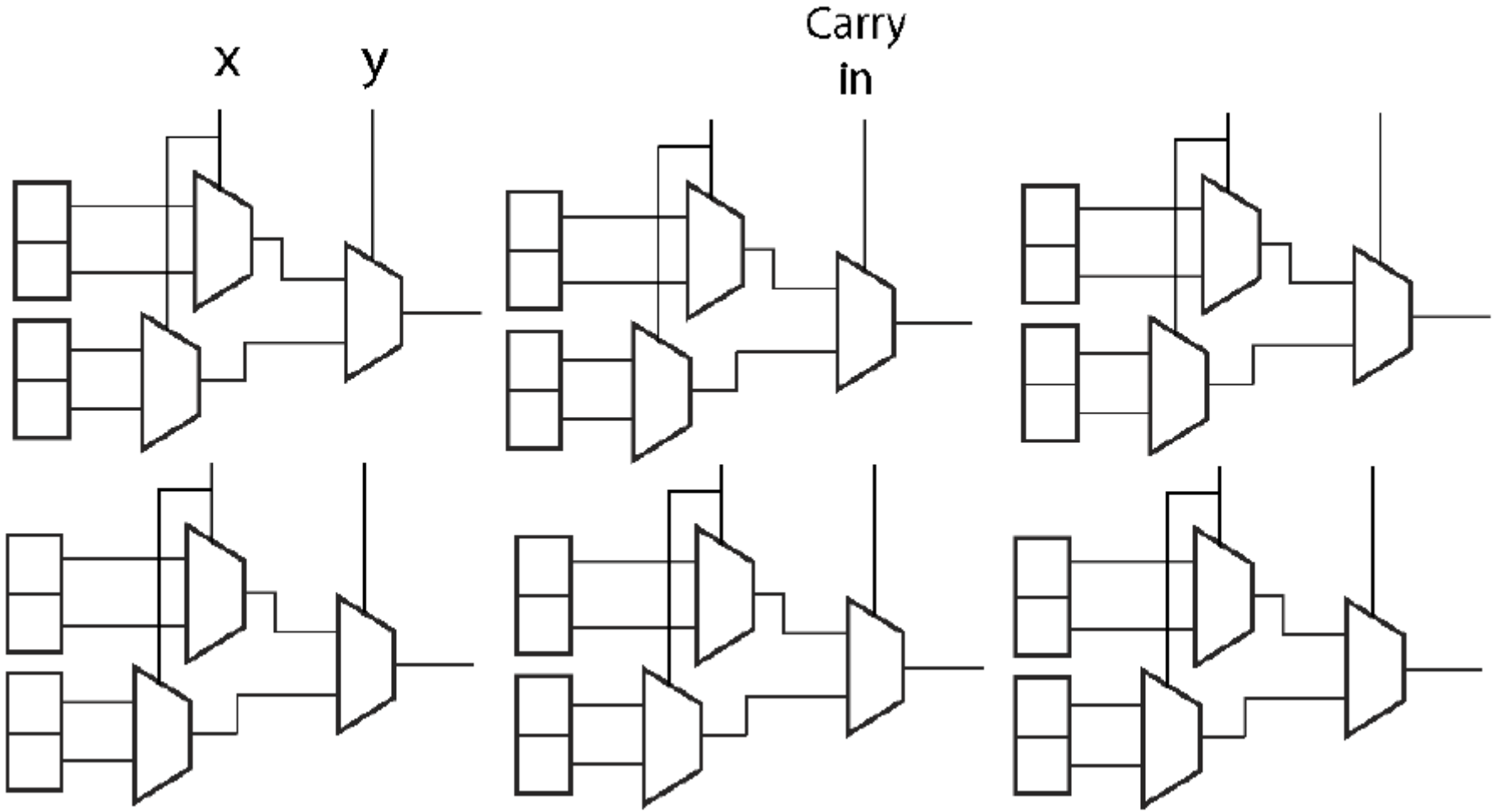


Field-Programmable Gate Array (FPGA)

FPGAs typically consist of blocks of logic elements interconnected by switches and multiplexers in an “island” configuration.

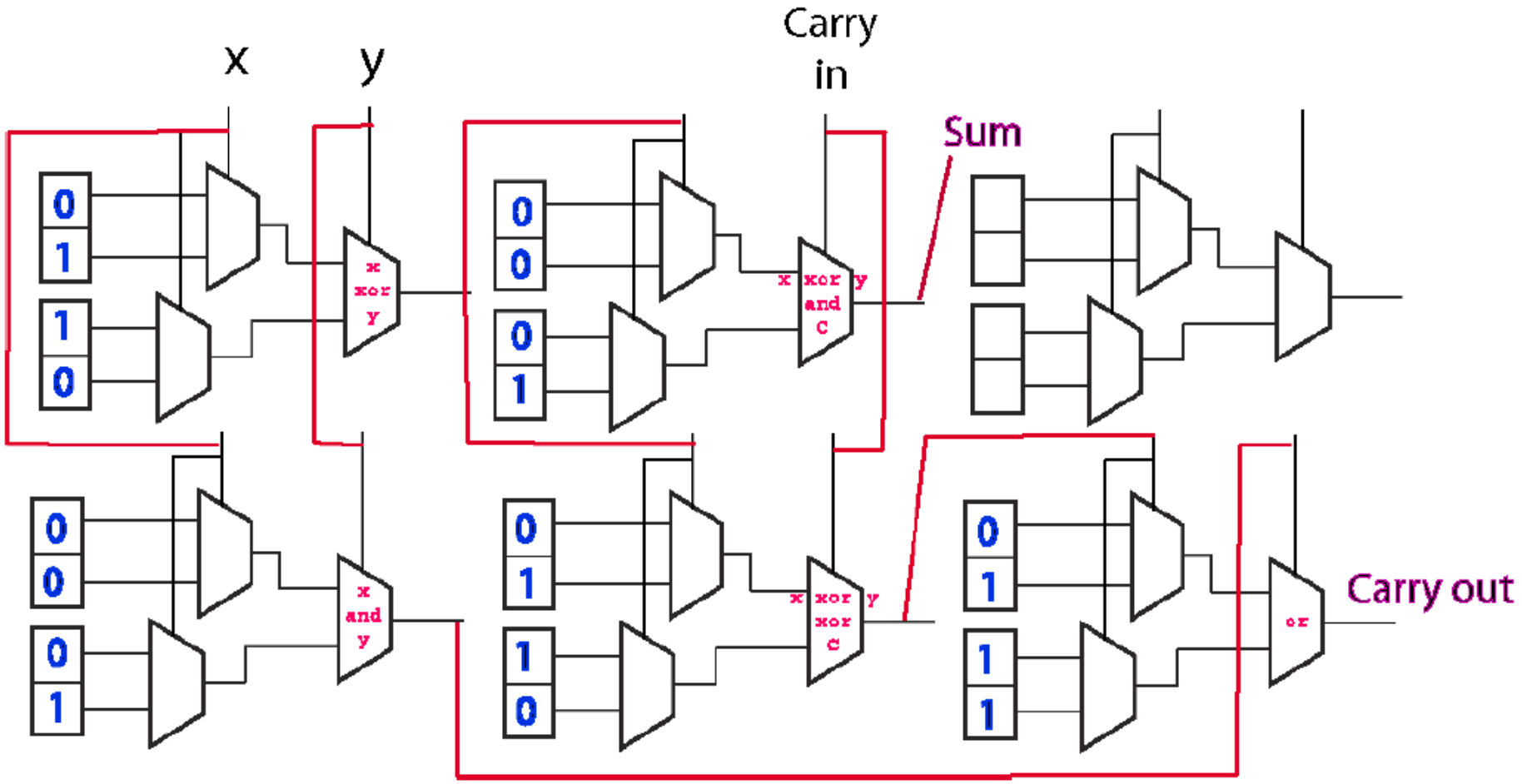


Field-Programmable Gate Array (FPGA)

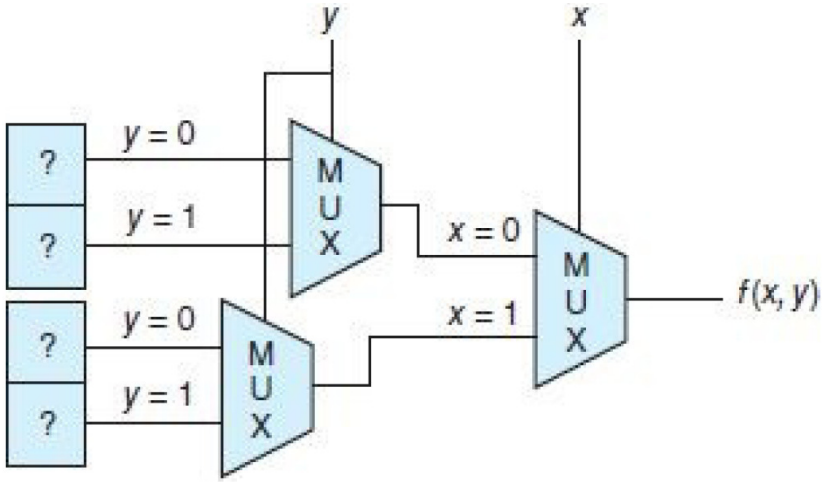


Field-Programmable Gate Array (FPGA)

FPGA is used to implement a full adder:

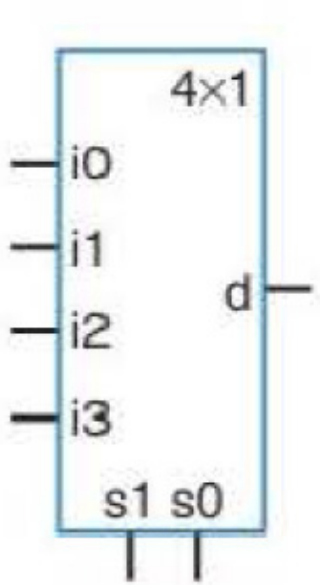


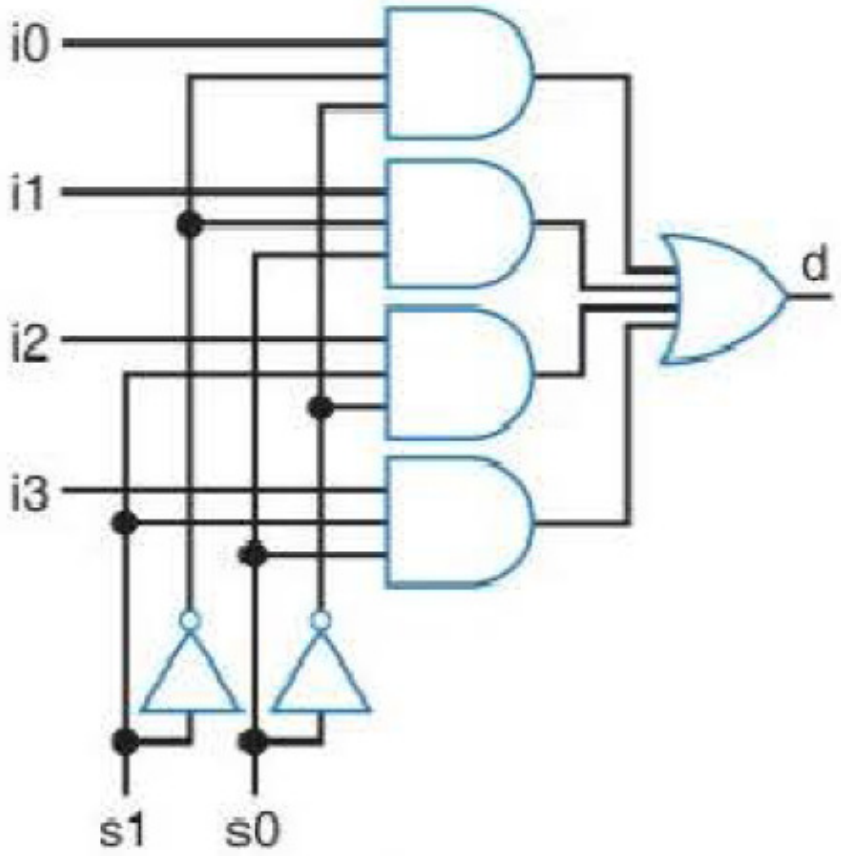
Field-Programmable Gate Array (FPGA)



- FPGAs provide a multitude of logic functions at a reasonable cost with acceptable performance in many applications.
- Because they can be reprogrammed as many times as needed, FPGAs are especially **useful for prototyping customized circuit designs**.
- In fact, **FPGAs can even be programmed to reprogram themselves!** Because of this, FPGA-based systems are considered by many to be the first of a new breed of computers that can dynamically reconfigure themselves in response to variations in workload or component failures.

FPGA with 4-to-1 multiplexers





s_1	s_0	d
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

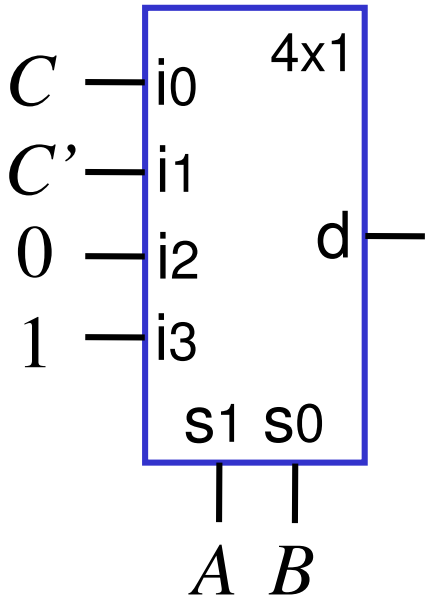
In the most naïve method, a 4-to-1 multiplexer can generate any 2-input function.

If inverted inputs can be provided, a 4-to-1 multiplexer can generate any 3-input function.

FPGA with 4-to-1 multiplexers

Assume that we want to implement the function $F = A'B'C + A'BC' + AB$ using an FPGA with programmable logic blocks consisting of 4-to-1 multiplexers

A	B	C	F	MUX input
0	0	0	0	C
0	0	1	1	C
0	1	0	1	C'
0	1	1	0	C'
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

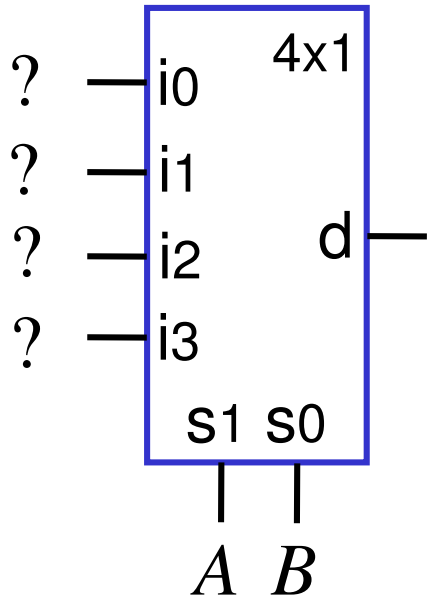


s_1	s_0	d
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

FPGA with 4-to-1 multiplexers

Implement the function given by the truth table shown below using an FPGA with programmable logic blocks consisting of 4-to-1 multiplexers. Assume that the inputs and their complements are available.

A	B	C	F	MUX input
0	0	0	0	
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

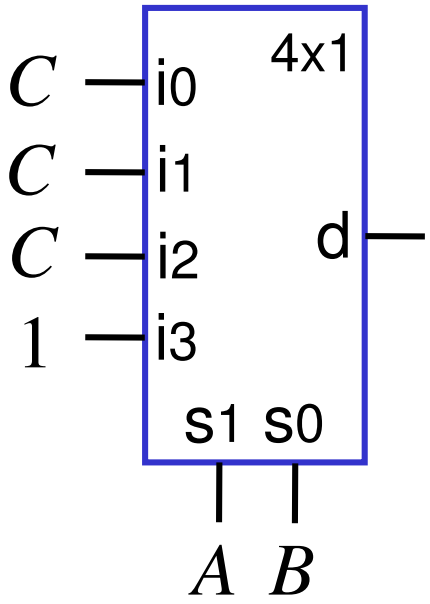


s_1	s_0	d
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3

FPGA with 4-to-1 multiplexers

Implement the function given by the truth table shown below using an FPGA with programmable logic blocks consisting of 4-to-1 multiplexers. Assume that the inputs and their complements are available.

A	B	C	F	MUX input
0	0	0	0	C
0	0	1	1	C
0	1	0	0	C
0	1	1	1	C
1	0	0	0	C
1	0	1	1	C
1	1	0	1	1
1	1	1	1	1



s_1	s_0	d
0	0	i_0
0	1	i_1
1	0	i_2
1	1	i_3