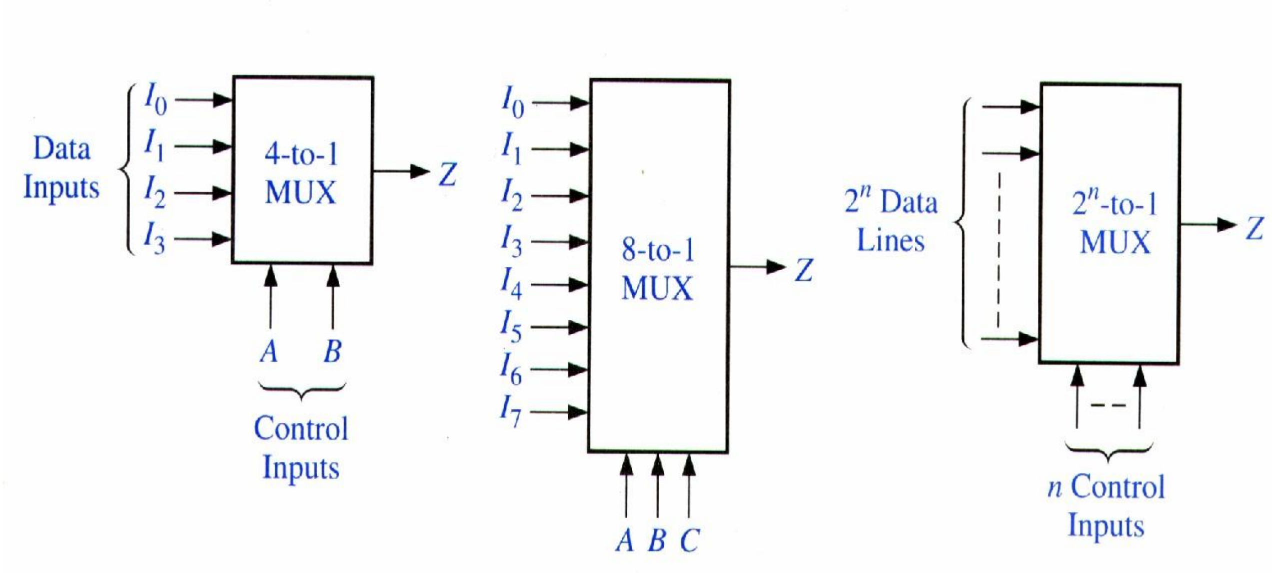
Lecture 9 Multiplexer, Decoder, and PLD

- SSI (small-scale integration)
  - NAND, NOR, NOT, flip flop etc
    - Gate count < 10.
- MSI (medium-scale integration)
  - Adders, multiplexers, decoders, registers, counters
    - Gate count < 100
- LSI
- VLSI (very large-scale integration
  - Memory chips, microprocessors

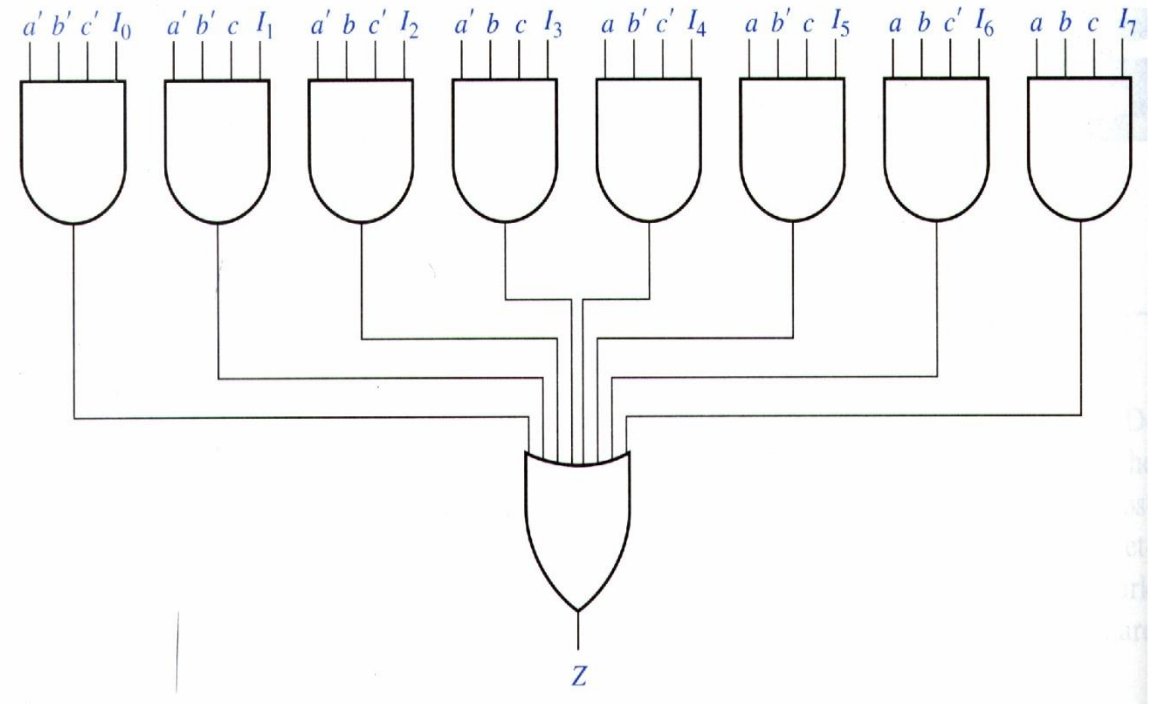
Multiplexers

- Multiplexers are selectors.
  - Z = A’B’I_0 + A’BI_1 + AB’I_2 + ABI_3. (4-to-1 MUX)



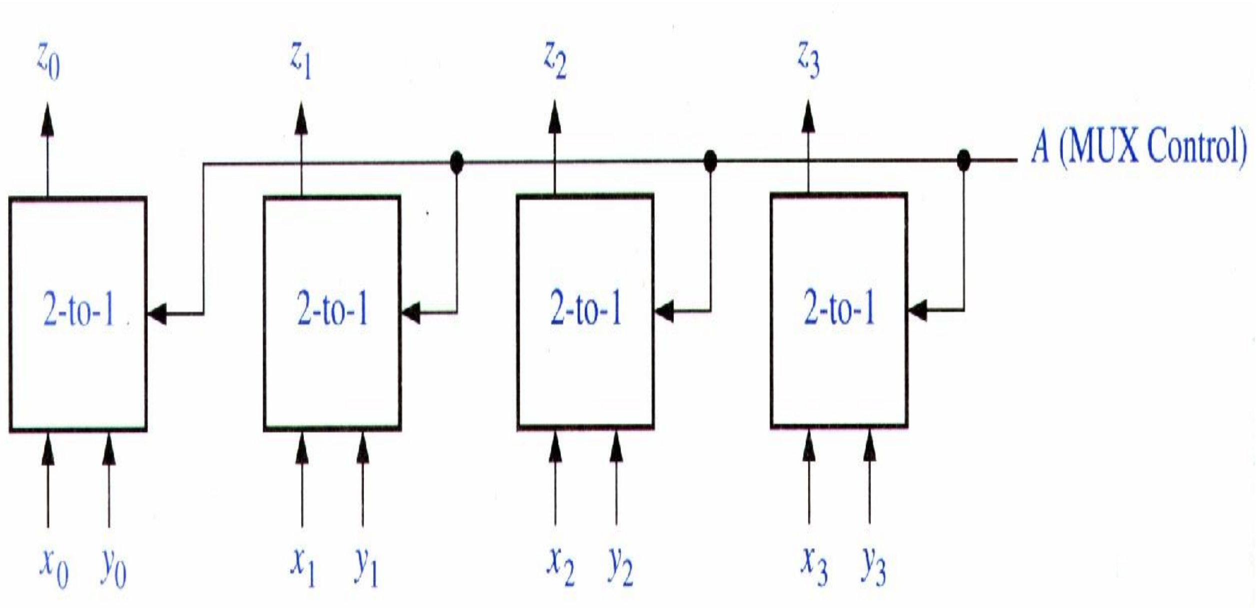
Logic Diagram for MUX

- 8-to-1 MUX



Application of MUX

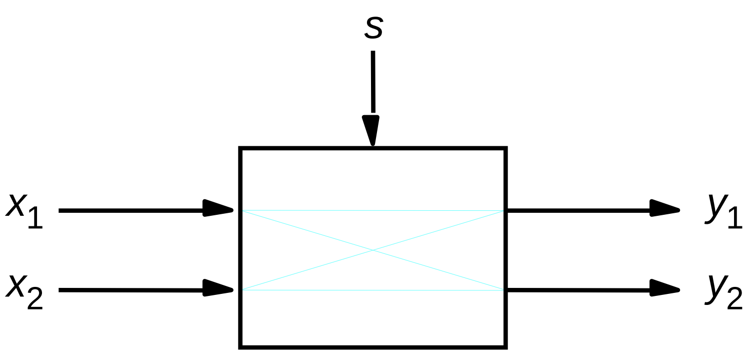
- Select data.



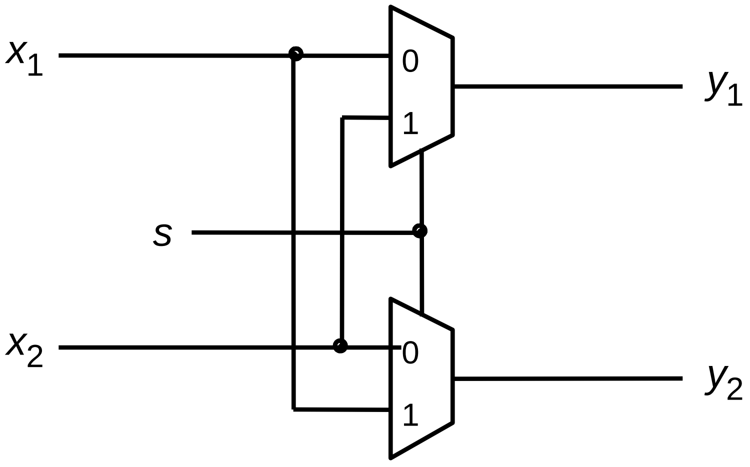
Application of MUX

Crossbar switch

- As s=0, x1 to y1, x2 to y2.
- As s=1, x1 to y2, x2 to y1.



(a) A 2x2 crossbar switch

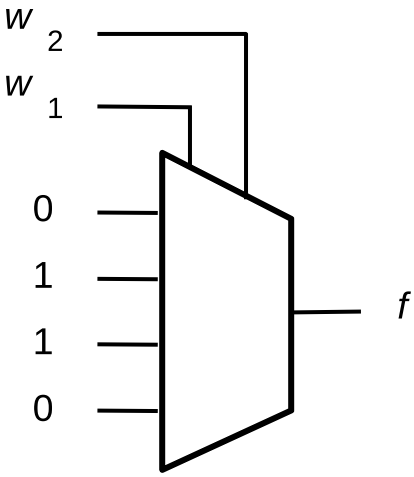


(b) Implementation using multiplexers

Using Mux for logic function, XOR

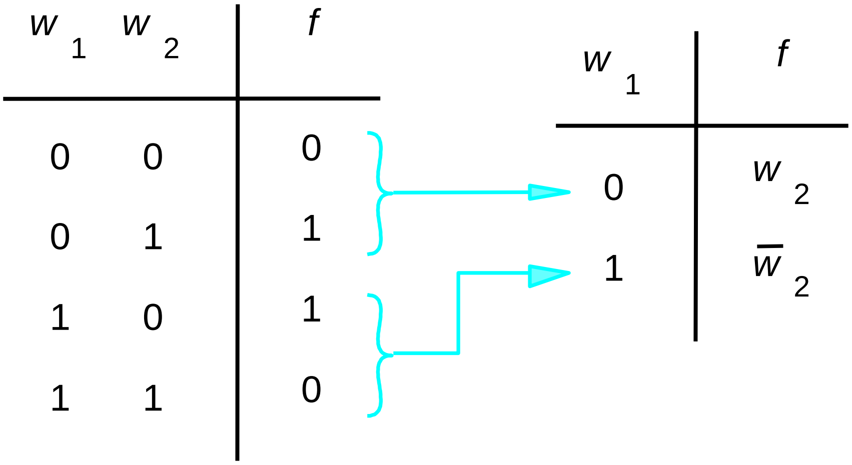
- f = w_1 xor w_2

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0



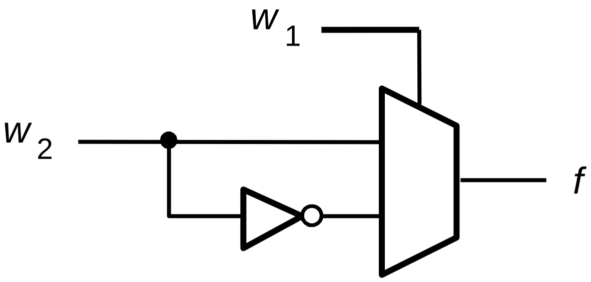
Better

- Modify the truth table.
- When $w_1 = 0$, $f = w_2$. Otherwise, $f = \sim w_2$



w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

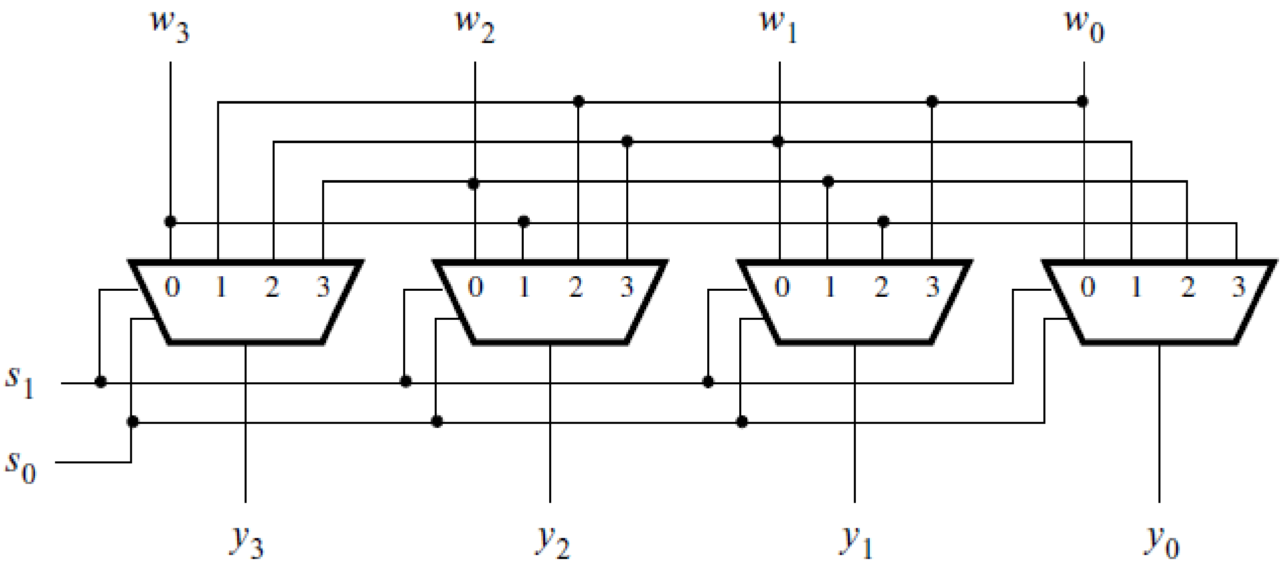
w_1	f
0	w_2
1	$\overline{w}_2$



Using Mux to Implement a barrel shifter

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

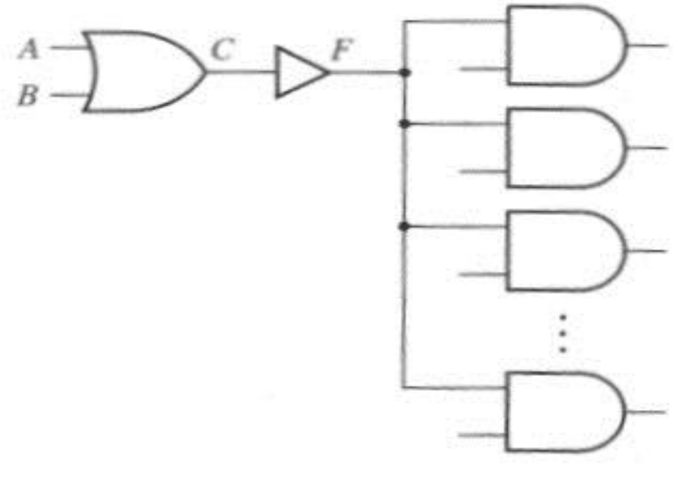
(a) Truth table

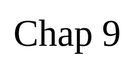


(b) Circuit

Buffers/Tri-state Buffer

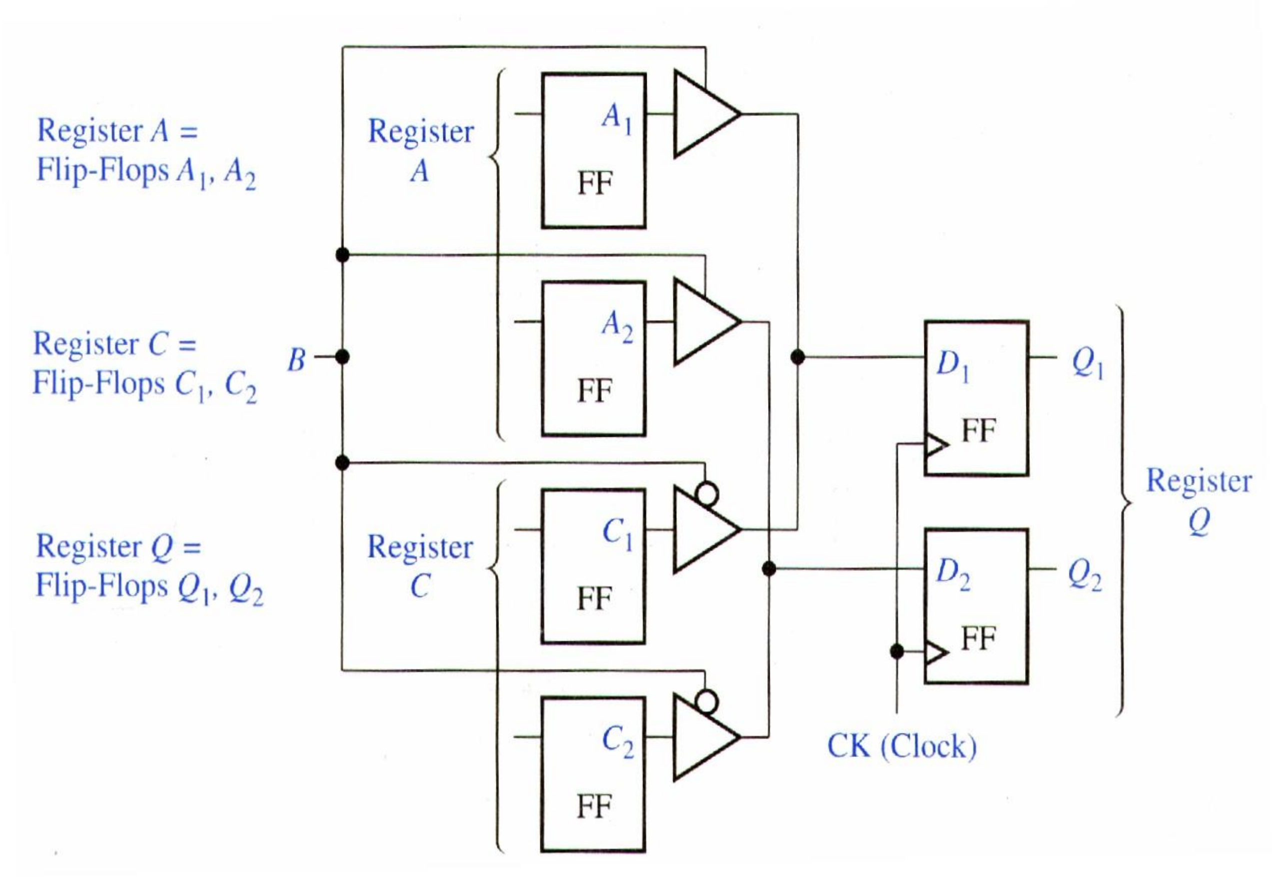
- Driving capability
- High impedance





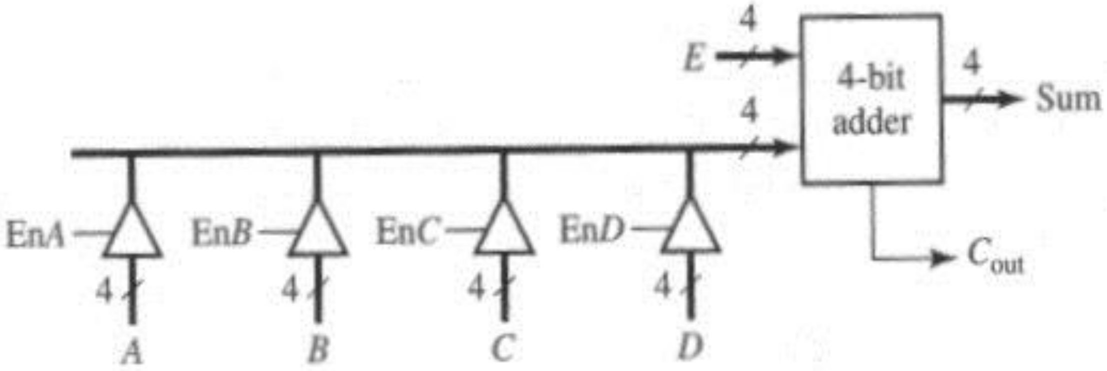
Tri-state Buffer Application

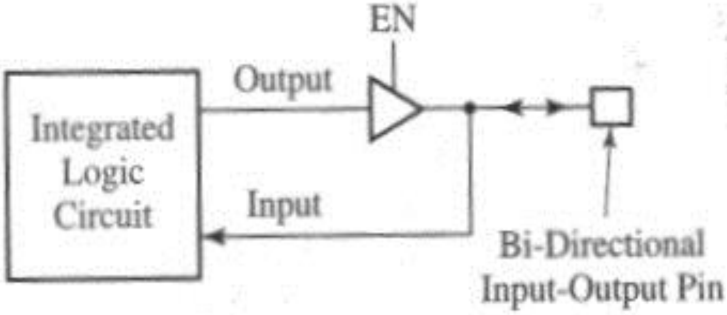
- Data transfer between registers



Tri-state Buffer Application (cont.)

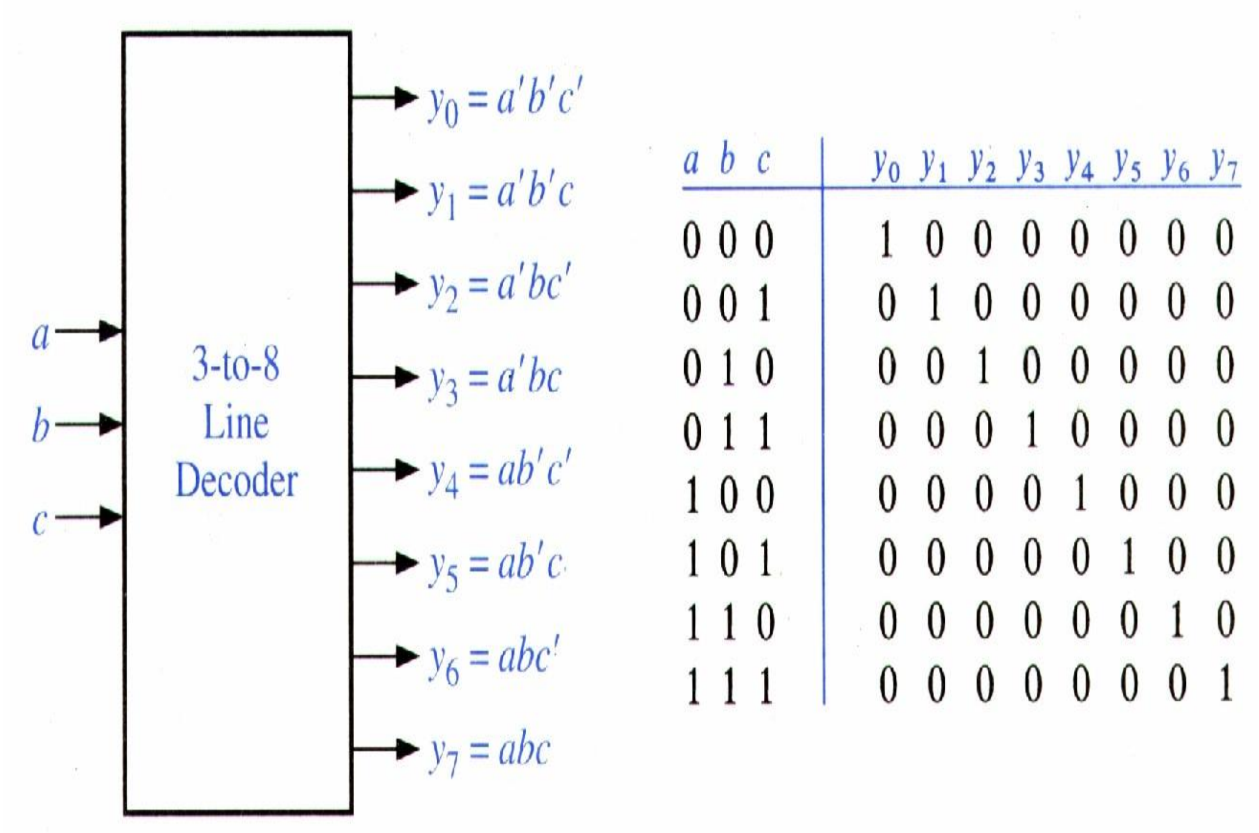
- Source selection





Decoder

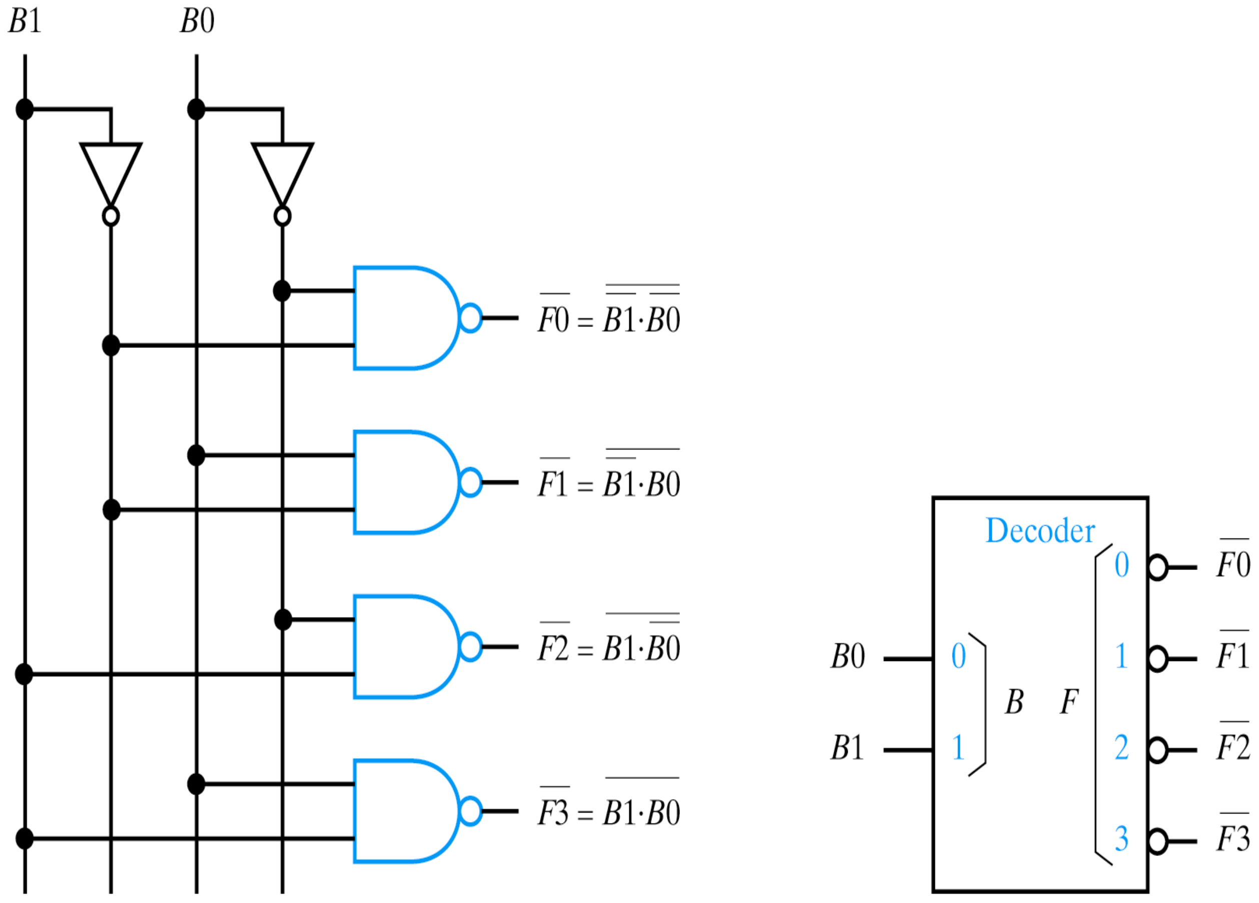
- 3-to-8 line decoder
  - An n-to-2^n decoder generates all 2^n minterms of the n input variables.



a	b	c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

More Decoder

- 2-to-4 decoder: minterm generator

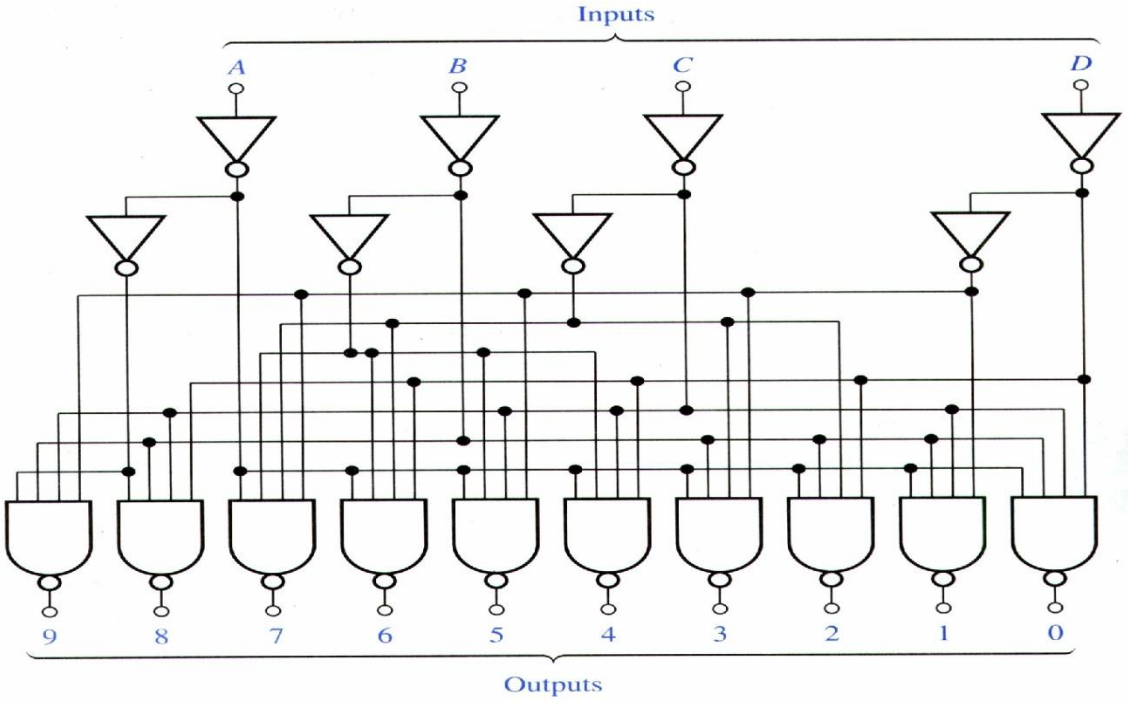

B1
B0
$\overline{F0} = \overline{\overline{B1}\cdot\overline{B0}}$
$\overline{F1} = \overline{\overline{B1}\cdot B0}$
$\overline{F2} = \overline{B1\cdot\overline{B0}}$
$\overline{F3} = \overline{B1\cdot B0}$
Decoder
B0
B1
0
1
B
F
0
1
2
3
$\overline{F0}$
$\overline{F1}$
$\overline{F2}$
$\overline{F3}$

(a) (b)

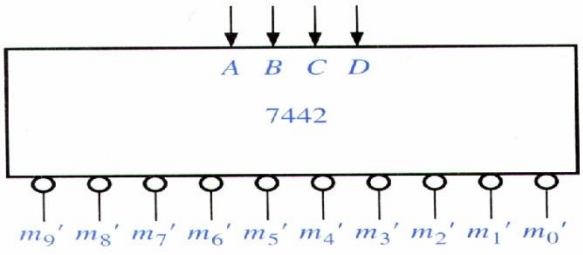
Figure 3.5.3 Decoder: (a) gate-level circuit diagram, (b) logic symbol.

Decoder (cont.)

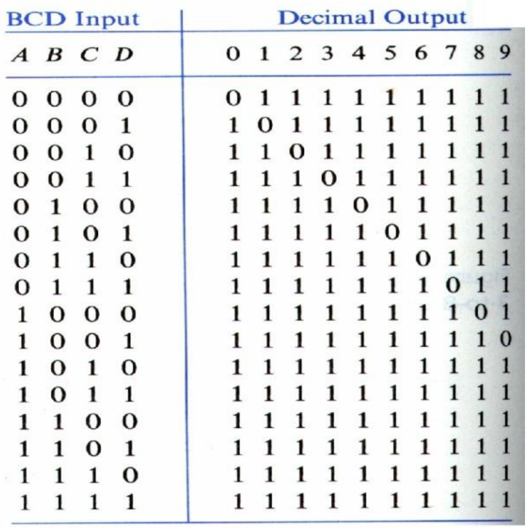
- 4-to-10 line decoder



(a) Logic diagram


A B C D
7442
m_9' m_8' m_7' m_6' m_5' m_4' m_3' m_2' m_1' m_0'

(b) Block diagram

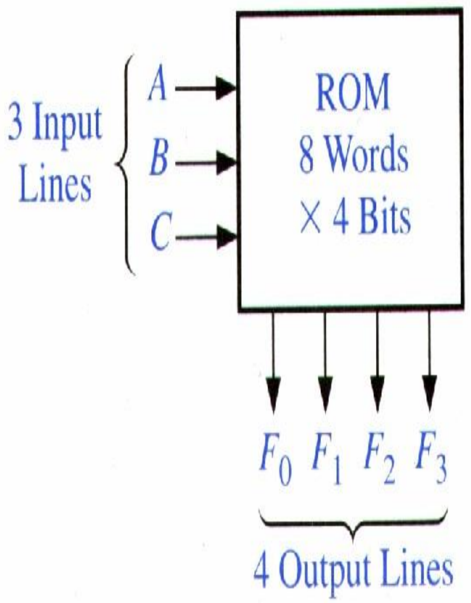


BCD Input				Decimal Output									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

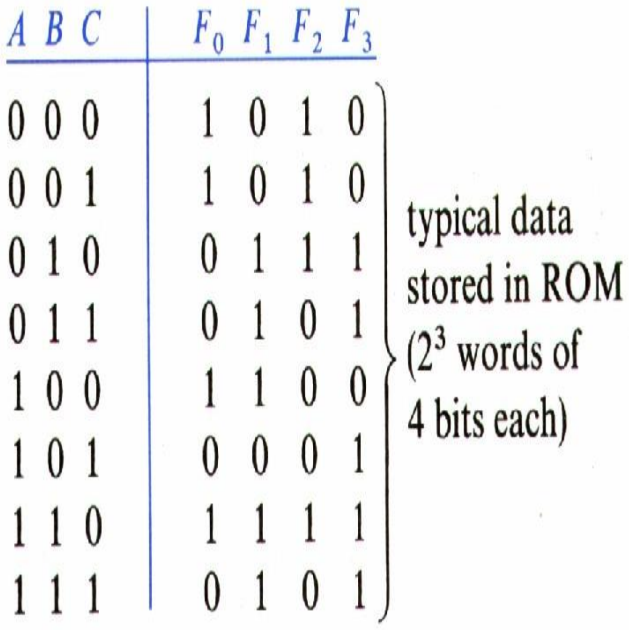
(c) Truth table

ROM

- Read-only memory: stored data can not be changed under normal operating conditions.



(a) block diagram



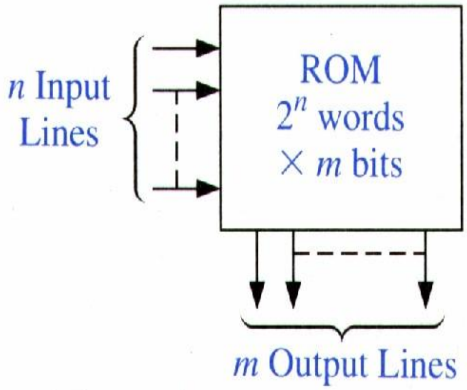
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

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

(b) truth table for ROM

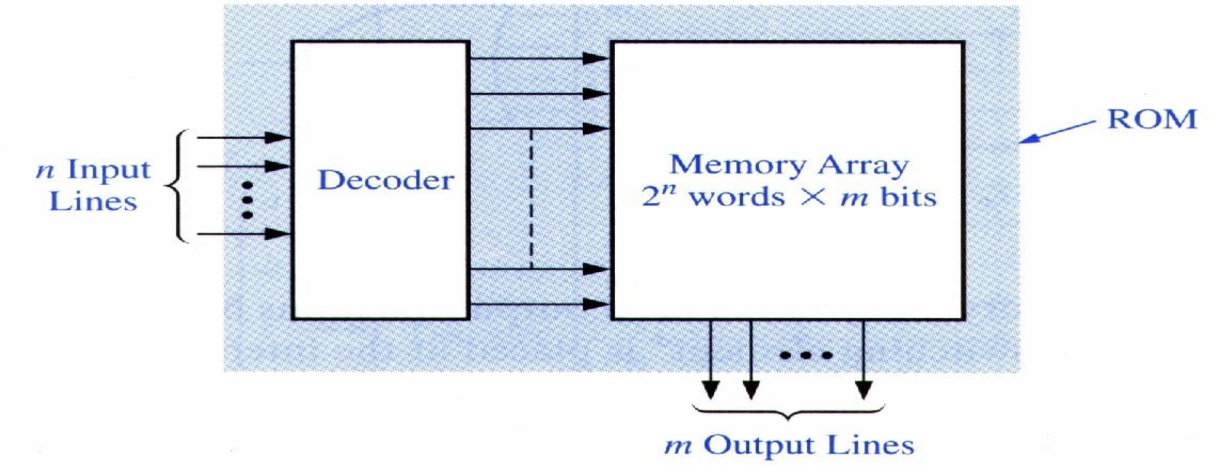
ROM (cont.)

- ROM size = 2^n x m bits.



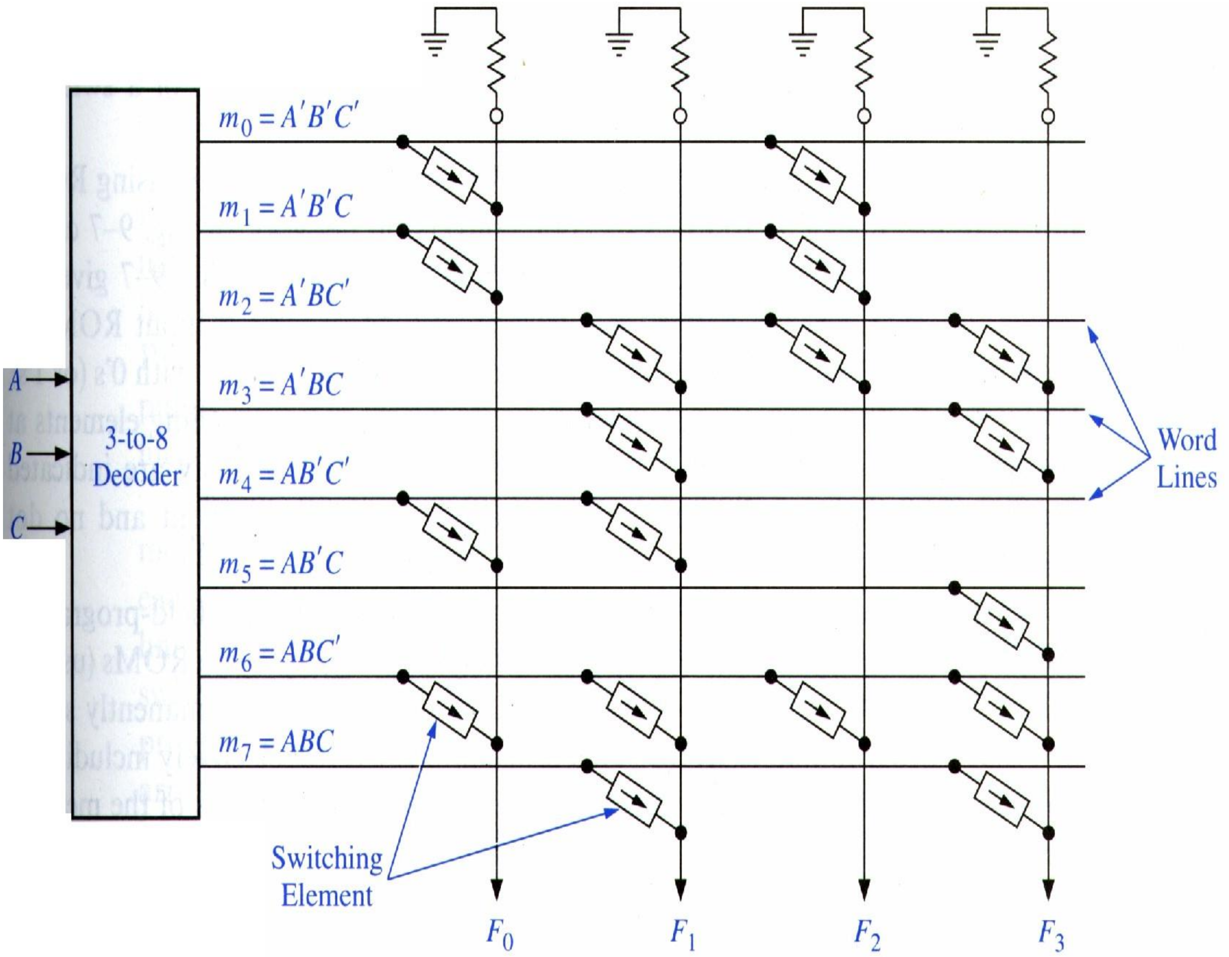
n Input Variables	m Output Variables
00 ··· 00	100 ··· 110
00 ··· 01	010 ··· 111
00 ··· 10	101 ··· 101
00 ··· 11	110 ··· 010
⋮	⋮
11 ··· 00	001 ··· 011
11 ··· 01	110 ··· 110
11 ··· 10	011 ··· 000
11 ··· 11	111 ··· 101

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



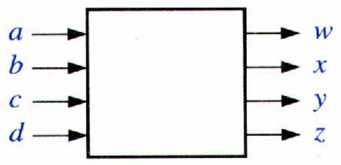
ROM Basic Structure

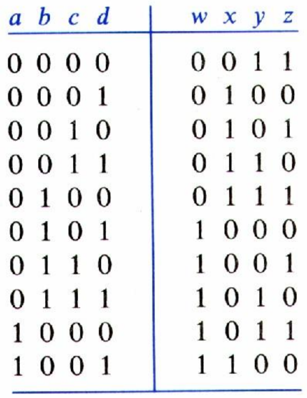
- 8-word x 4 bit ROM
  - $F_0 = m_0 + m_1 + m_4 + m_6$



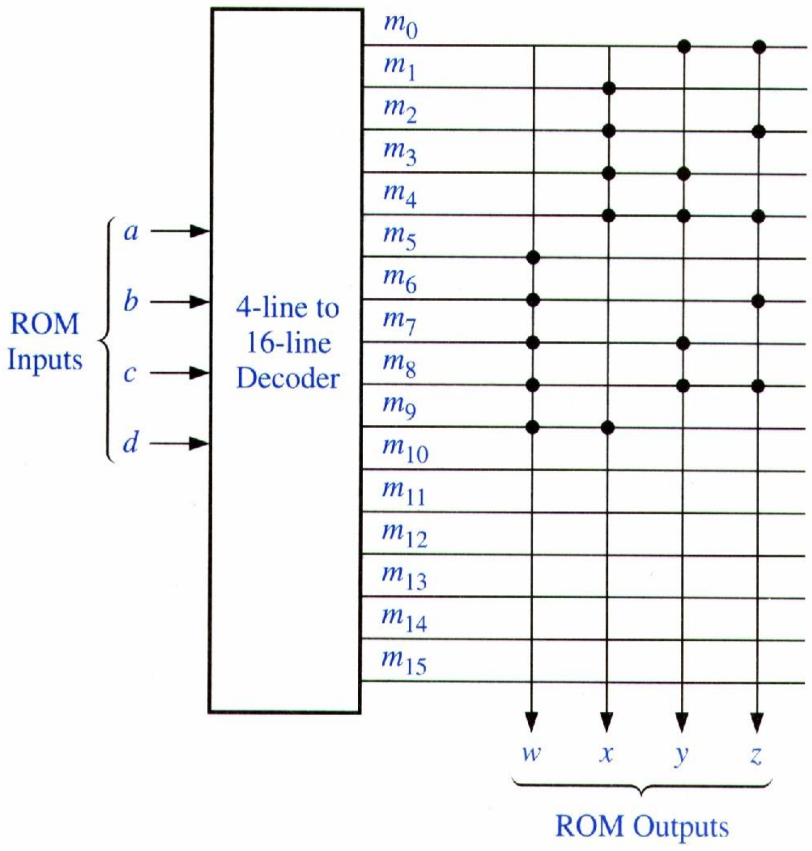
Multiple-Output Network Using ROM

- BCD to Excess-3 code





a	b	c	d	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

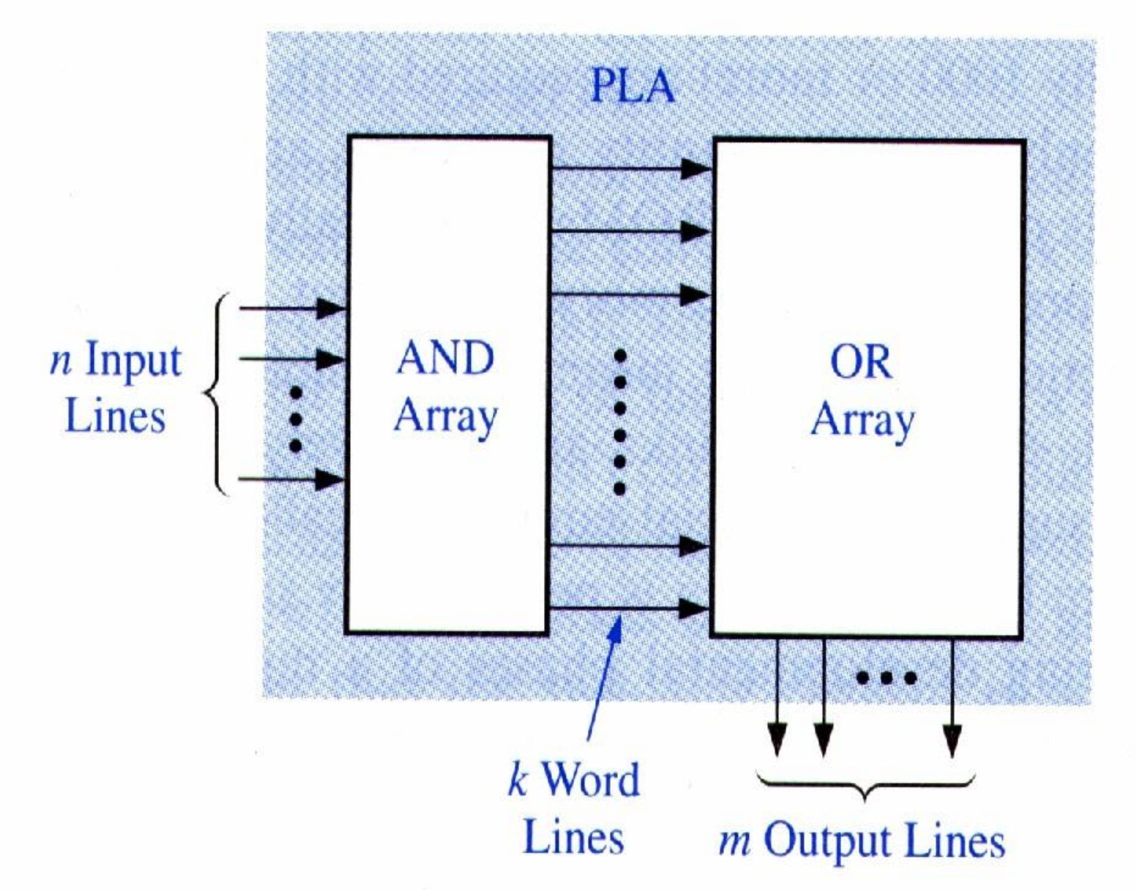


Types of ROM

- Mask-programmable ROM
  - Contents are stored during fabrication.
- Field-programmable ROM
  - PROM: programmable ROM
    - Fusible link (PROM programmer)
  - EPROM: erasable PROM
    - Use ultraviolet light for erasure.
  - EEPROM: electrically erasable PROM.
    - Flash memory

Programmable Logic Devices

- PLA (programmable logic array)
  - Realize m functions of n variables.



Implementation procedure for PLA

- Prepare the truth table based on your system
- Write the Boolean expression in SOP (sum of product) form.
- Obtain the minimum SOP form to reduce the number of product terms to a minimum.
- Decide the input connection of the AND matrix for generating the required product term.
- Then decide the input connections of OR matrix to generate the sum terms.
- Decide the connections of invert matrix.
- Program the PLA.

Programmable Symbology

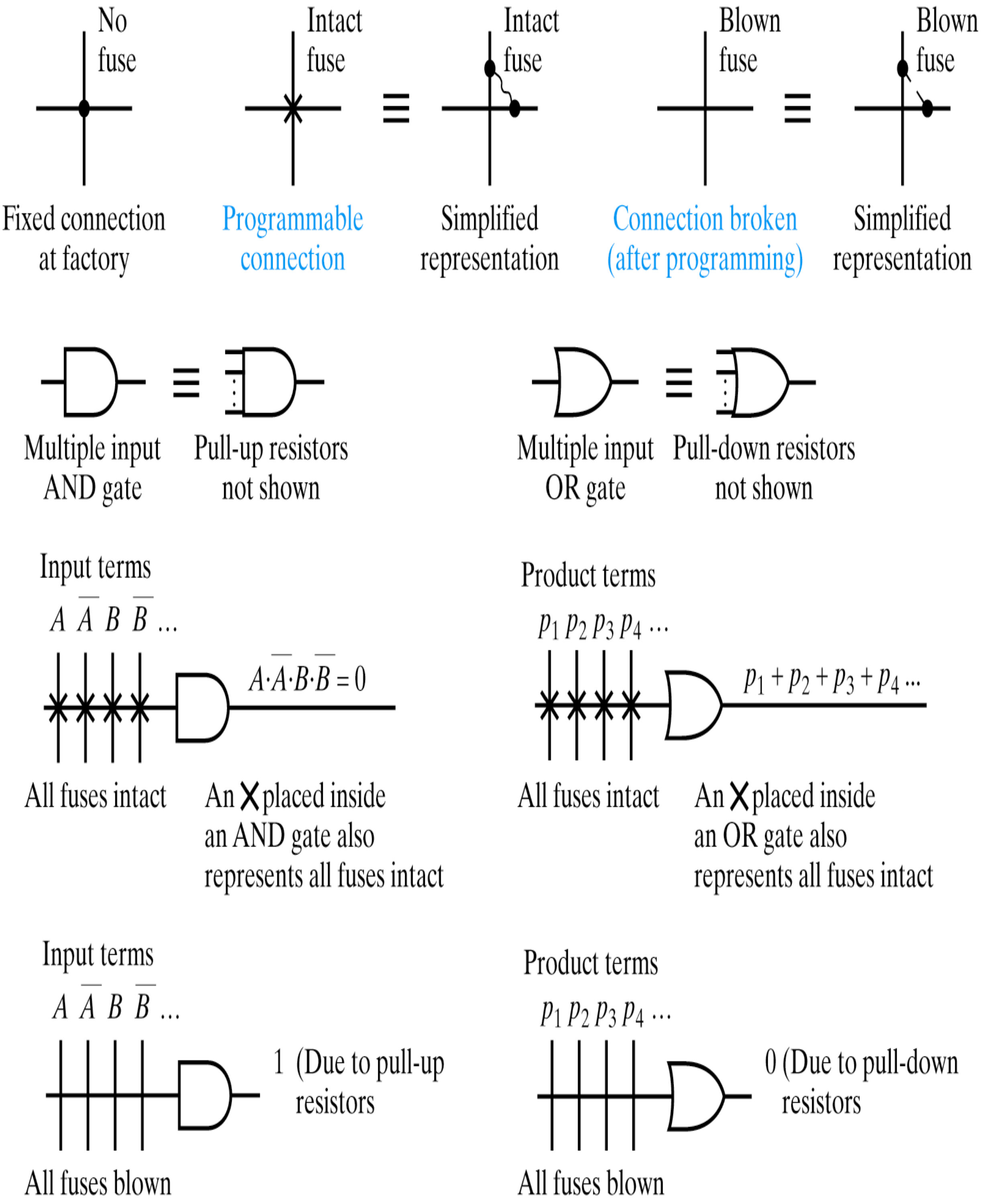
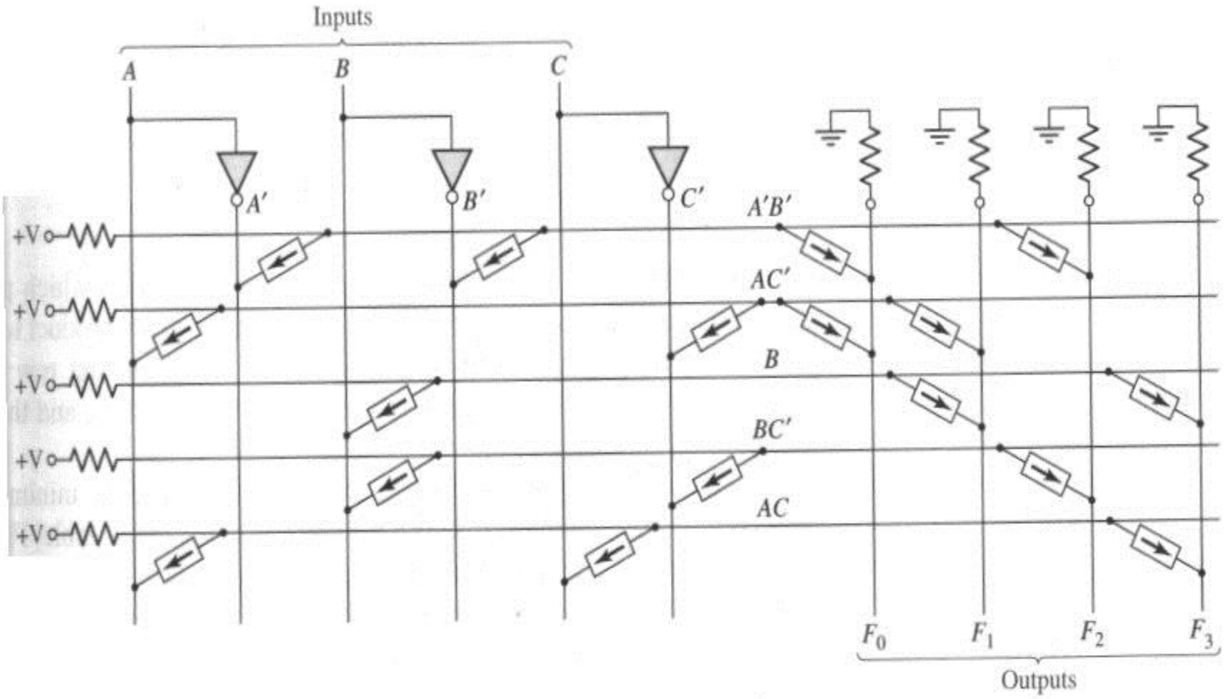

No fuse
Intact fuse
Intact fuse
Blown fuse
Blown fuse
≡
≡
Fixed connection at factory
Programmable connection
Simplified representation
Connection broken (after programming)
Simplified representation
≡
≡
Multiple input AND gate
Pull-up resistors not shown
Multiple input OR gate
Pull-down resistors not shown
Input terms
$A\ \bar{A}\ B\ \bar{B}\ \ldots$
$A \cdot \bar{A} \cdot B \cdot \bar{B} = 0$
All fuses intact
An X placed inside an AND gate also represents all fuses intact
Product terms
$p_1\ p_2\ p_3\ p_4\ \ldots$
$p_1 + p_2 + p_3 + p_4 \ldots$
All fuses intact
An X placed inside an OR gate also represents all fuses intact
Input terms
$A\ \bar{A}\ B\ \bar{B}\ \ldots$
1 (Due to pull-up resistors
All fuses blown
Product terms
$p_1\ p_2\ p_3\ p_4\ \ldots$
0 (Due to pull-down resistors
All fuses blown

Figure 3.6.2 Programmable symbology summary for the different types of PLDs.

PLA

- PLA 3 inputs, 5 product terms, 4 outputs.

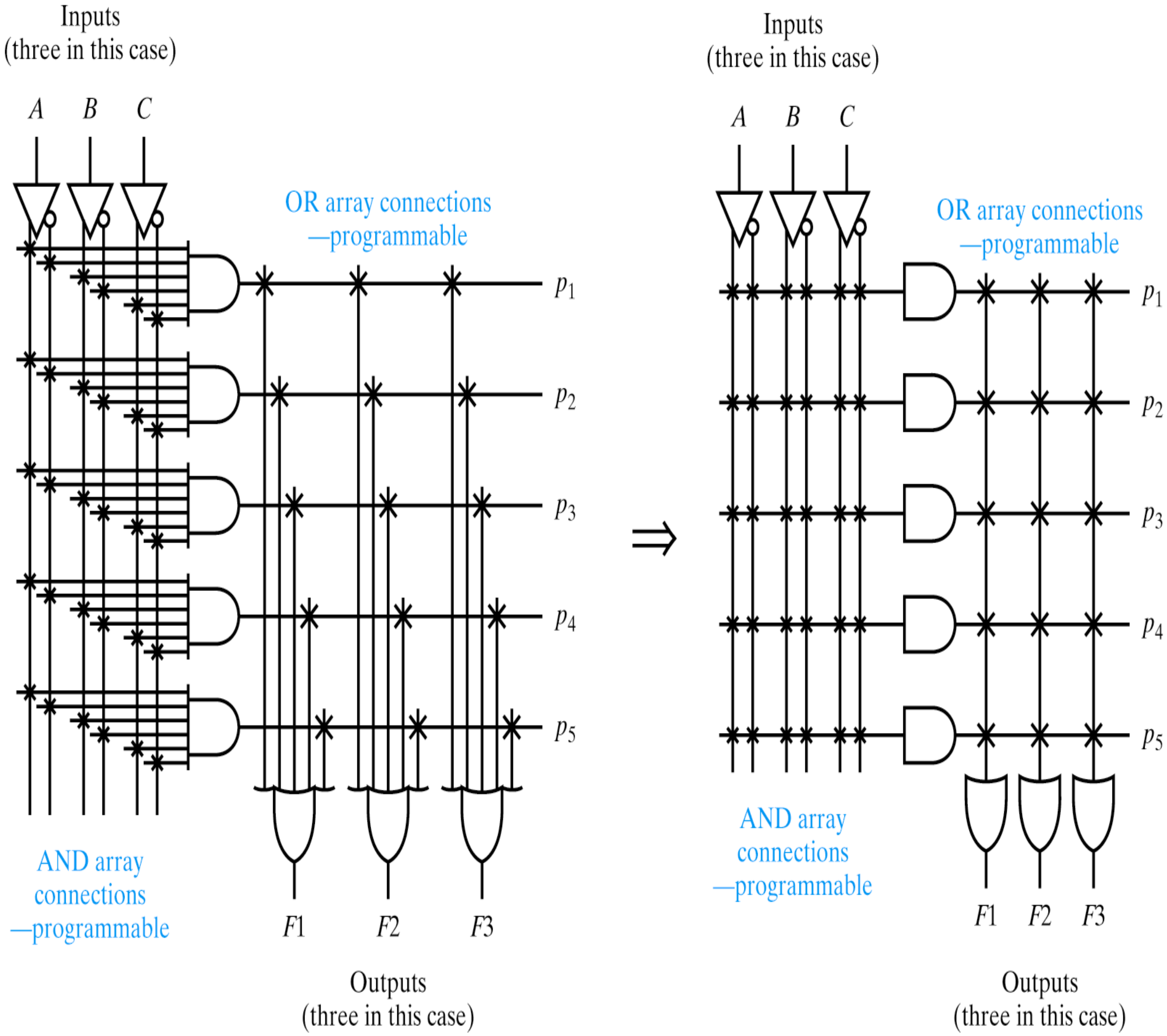


Product Term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
$A'B'$	0	0	–	1	0	1	0
AC'	1	–	0	1	1	0	0
B	–	1	–	0	1	0	1
BC'	–	1	0	0	0	1	0
AC	1	–	1	0	0	0	1

$$F_0 = A'B' + AC'$$
$$F_1 = AC' + B$$
$$F_2 = A'B' + BC'$$
$$F_3 = B + AC$$

PLA

- PLA circuit



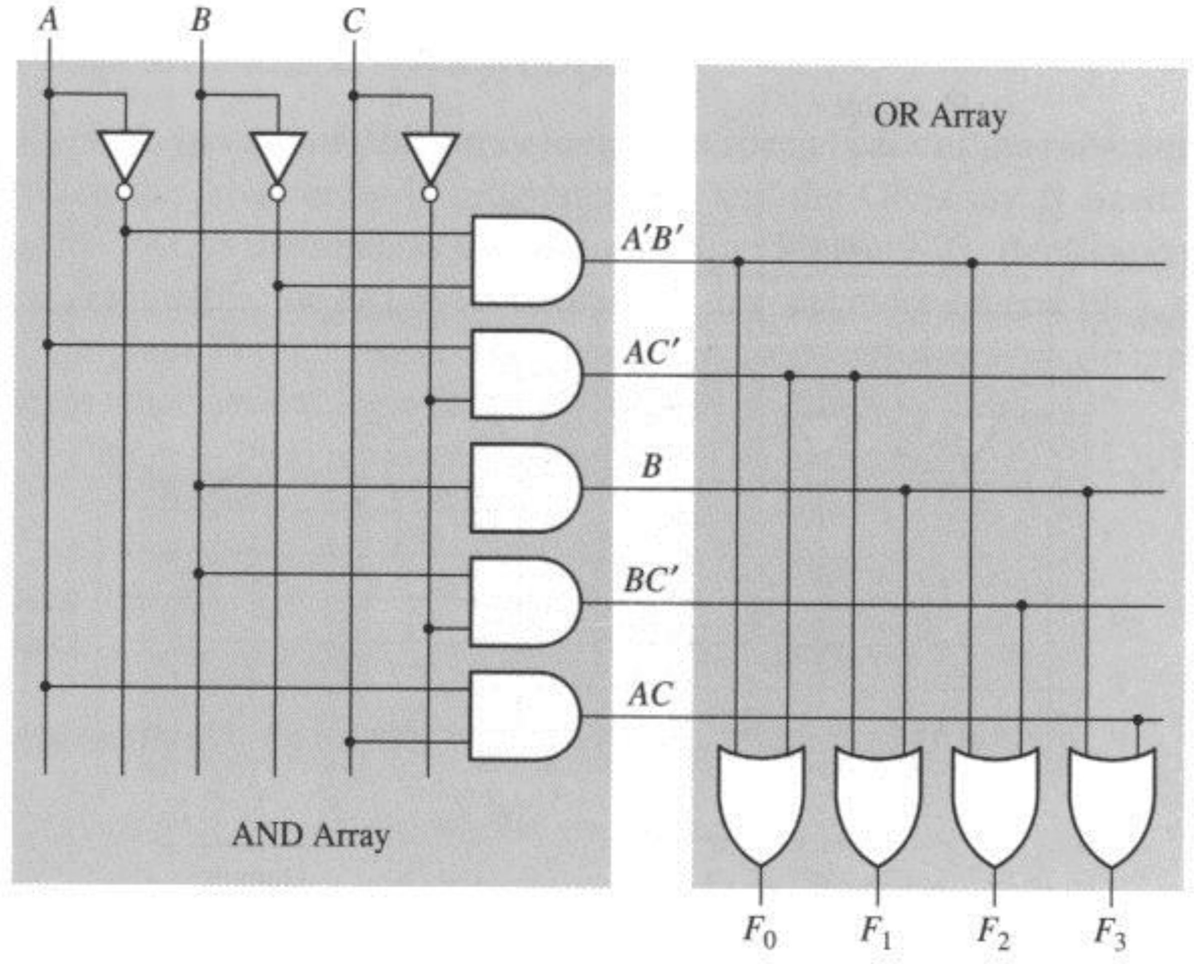
(a)

(b)

Figure 3.6.4 PLA circuit: (a) circuit representation using gates, (b) simplified circuit representation using gates.

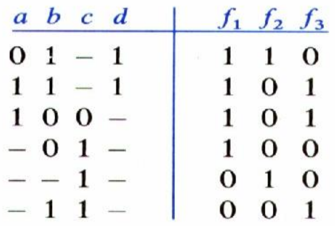
PLA (cont.)

- PLA 3 inputs, 5 product terms, 4 outputs. AND-OR array.



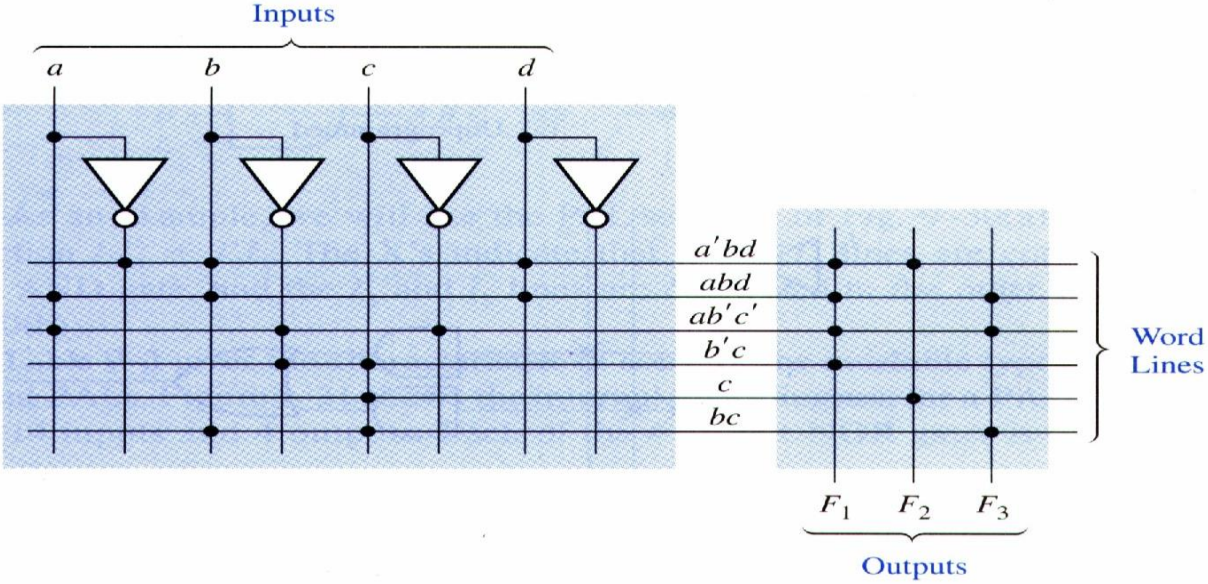
PLA Application

- f_1 = a’bd + abd + ab’c’ + b’c
- f_2 = c + a’bd
- f_3 = bc + ab’c’ + abd
    - If abcd = 0111, 1st, 5th, 6th rows are selected. f_1 = 1 +0+0, f_2 = 1+1+0, etc.



a	b	c	d	f_1	f_2	f_3
0	1	–	1	1	1	0
1	1	–	1	1	0	1
1	0	0	–	1	0	1
–	0	1	–	1	0	0
–	–	1	–	0	1	0
–	1	1	–	0	0	1

(a) PLA table

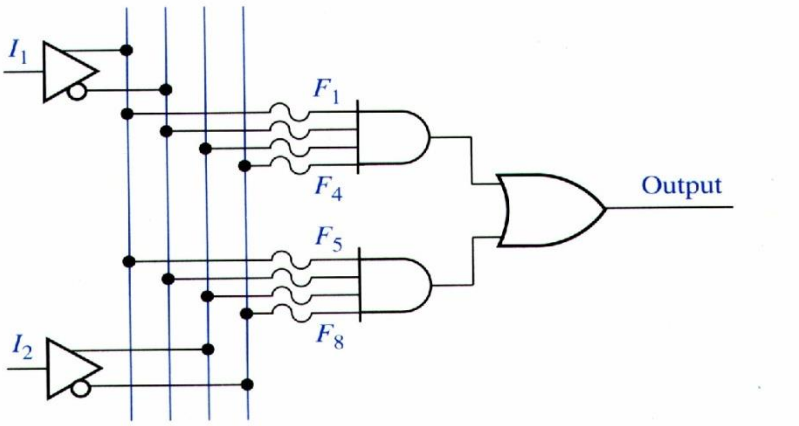


(b) PLA structure

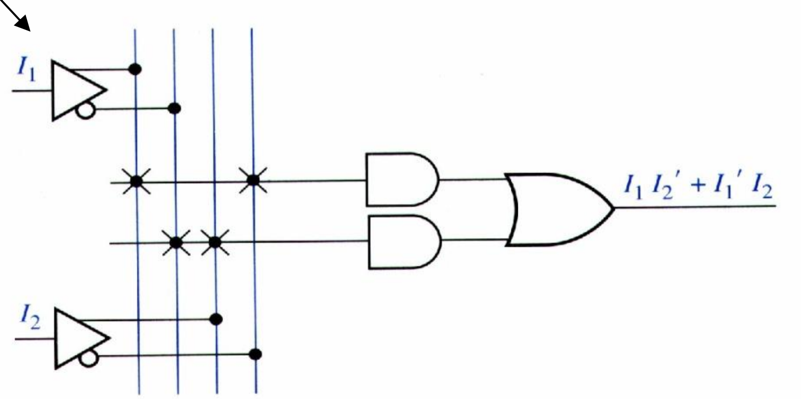
Programmable Array Logic (PAL)

- AND array is programmable. OR array is fixed.
- Use input buffer to drive many ANDs.

Input buffer



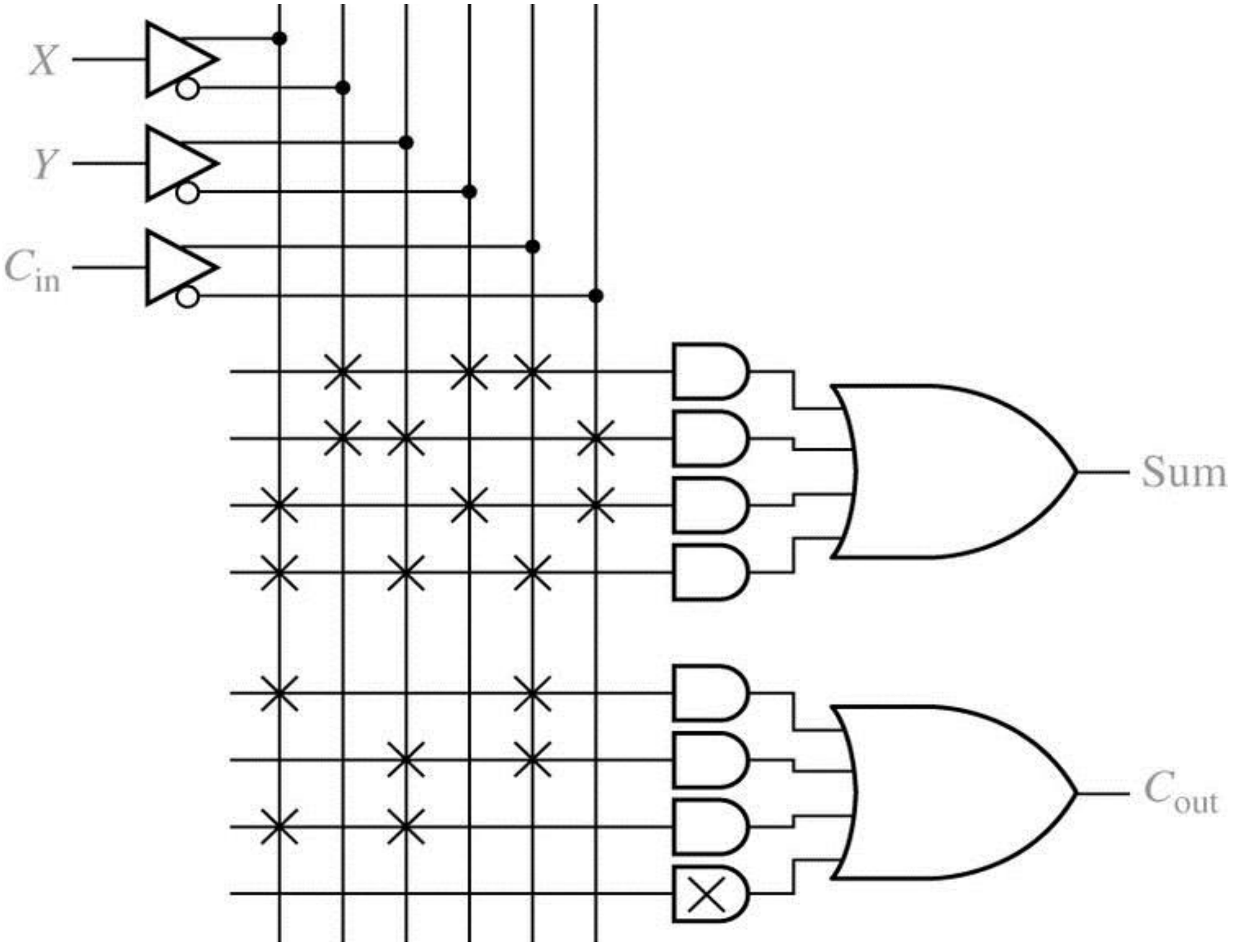
(a) Unprogrammed



(b) Programmed

Programmable Array Logic (PAL)

- Full Adder using a PAL



Why prefers PLA rather than ROM

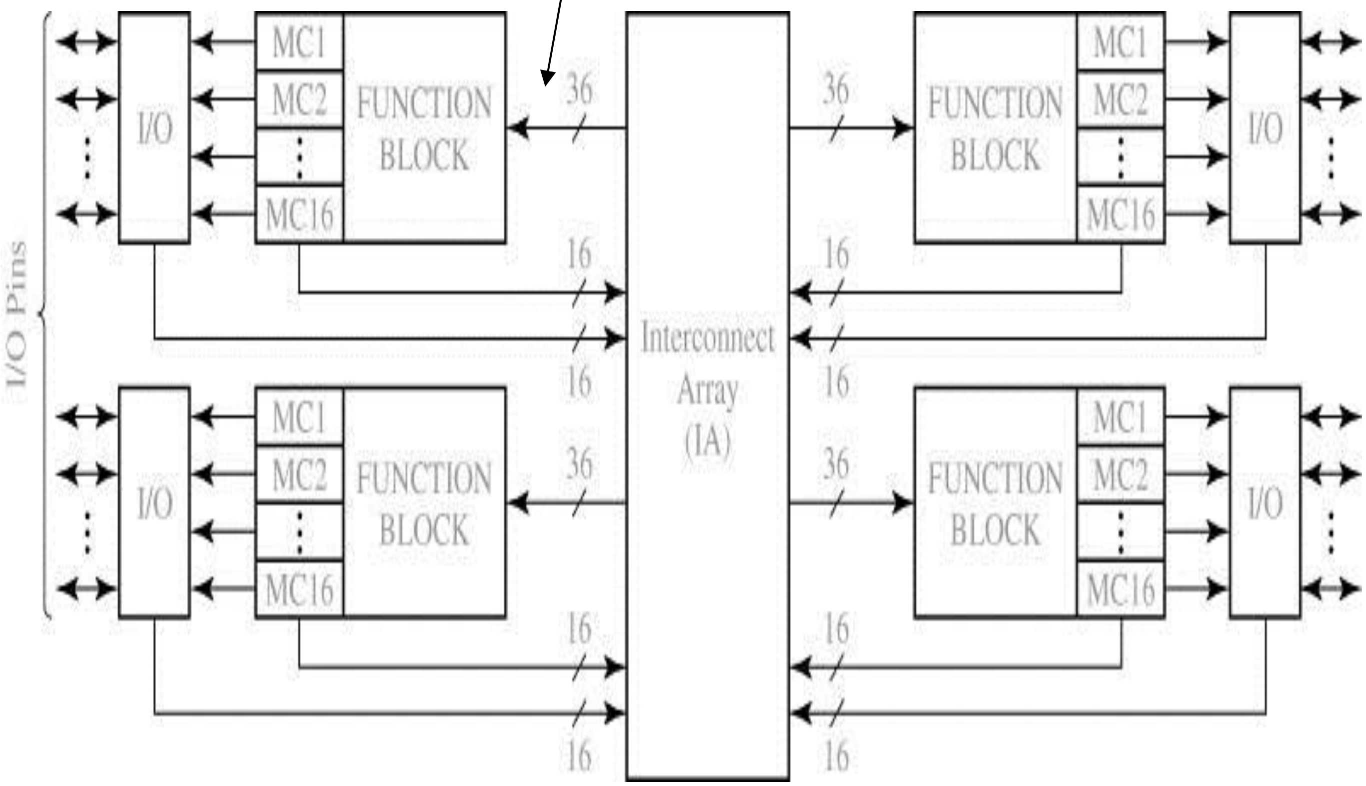
- A combinational circuit may occasionally have don't-care conditions. When implemented with a ROM, a don't care condition becomes an address input that will never occur.
- The words at the don't care address need not be programmed and may be left in their original state(all 0's or all 1's).
- The has the result that all the bit patterns available in ROM are used, but for the don’t care addresses you really can do without them.

What Programmable Means

- "programmable" does not indicate that all PLAs are field-programmable.
- Many are mask-programmed during manufacture in the same manner as a mask ROM.
  - This is particularly true of PLAs that are embedded in more complex and numerous integrated circuits such as microprocessors.
- PLAs that can be programmed after manufacture are called FPGA (Field-programmable gate array)

Complex Programming Logic Devices

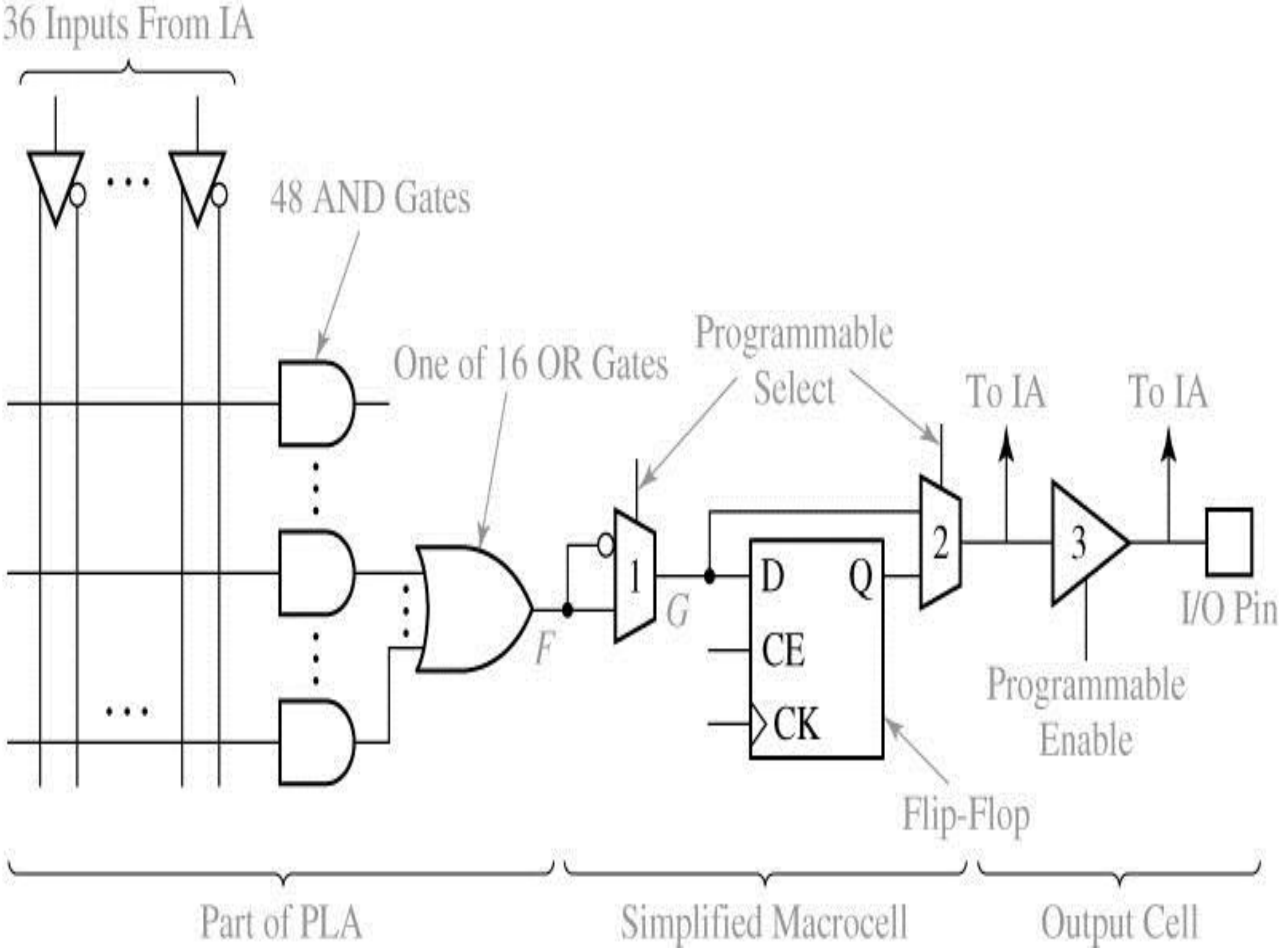
- Xilinx XCR3064XL CPLD
  - Function block (16 macrocells)= PLA
  - Macrocell = a flip flop + multiplexers
  - IA routes signals



Input of function block

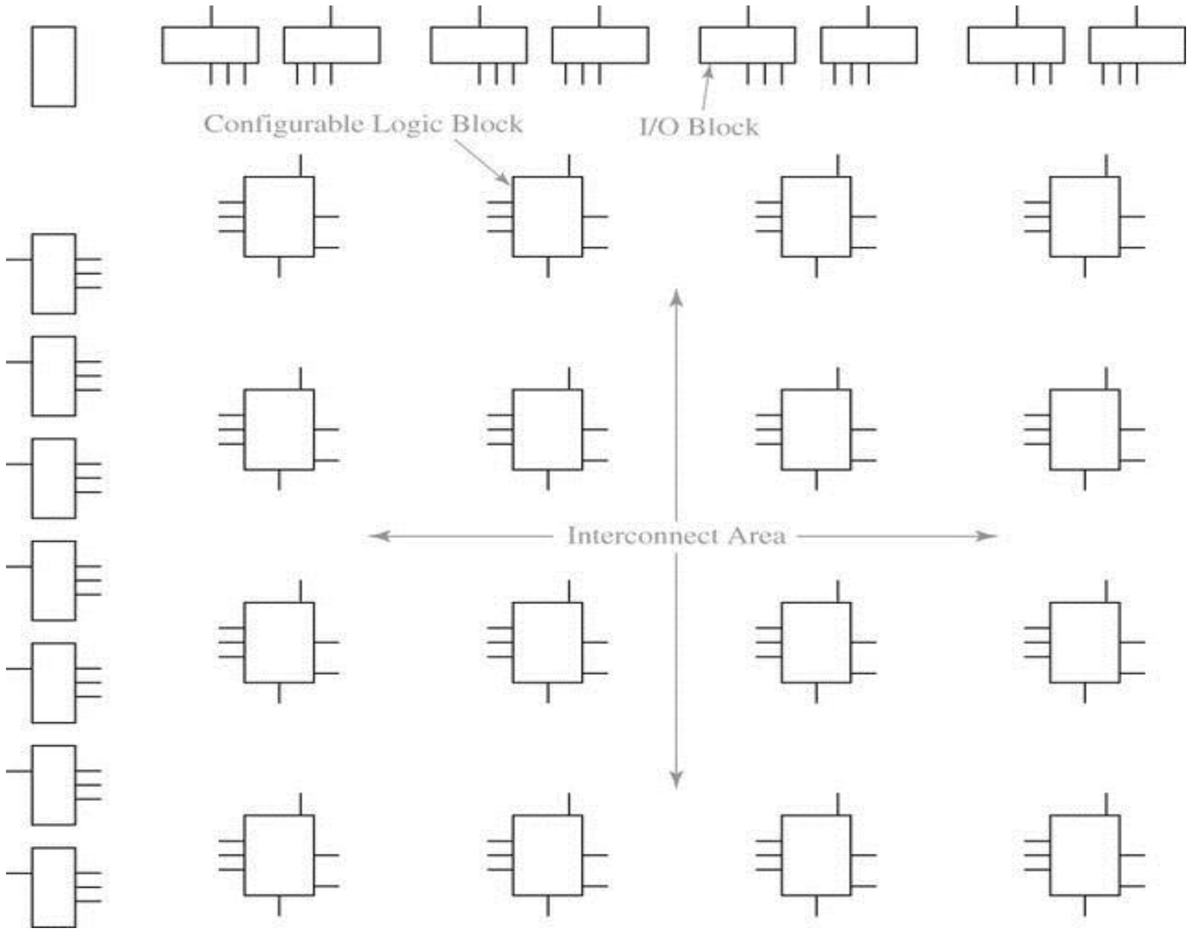
Function Block and MC

- Signal from PLA -> marcocell -> I/O pin
- Use CAD tool to fit the design into the PLD.



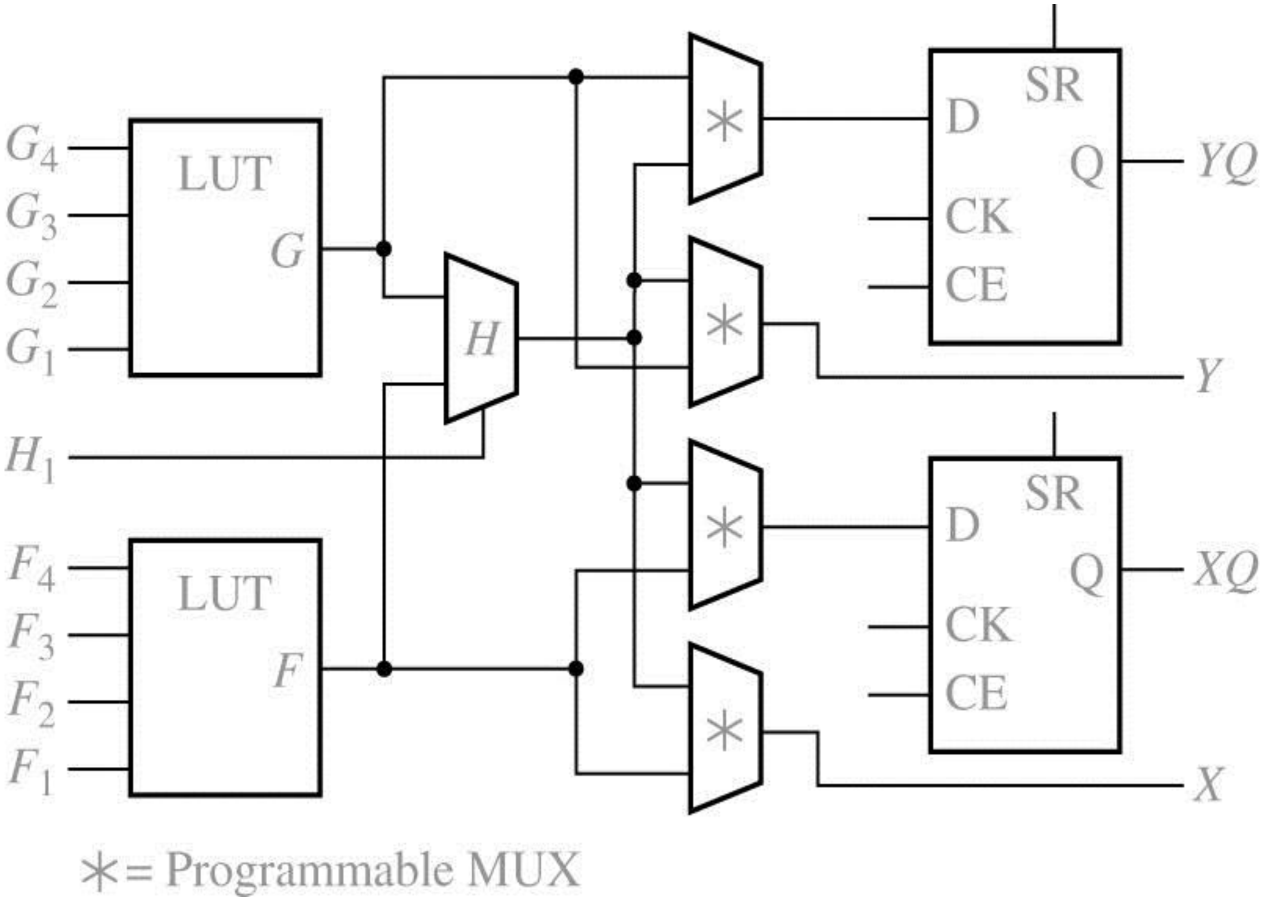
Field Programmable Gate Arrays (FPGA)

- Logic cell: configurable logic blocks (CLBs)
- Input/Output blocks (I/O blocks)



Configurable Logic Block

- Inside a CLB: function generators (LUT), FFs, and MUXs
- LUT: lookup table (truth table) is a reprogrammable ROM (16 1-bit words)



A Lookup Table (LUT)

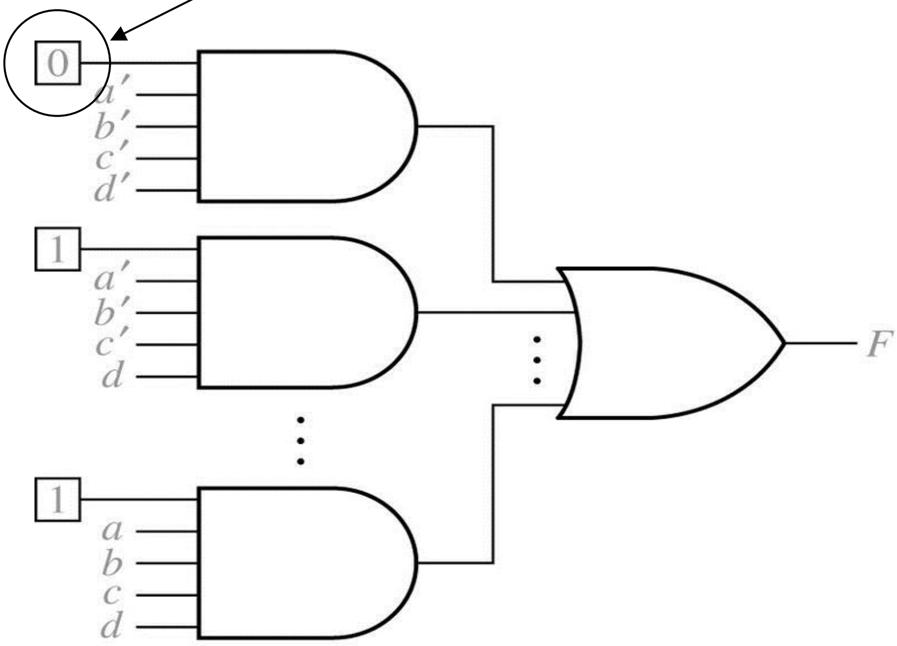
- If we want F = abc (one minterm)
  - 1110 (and F=1) + 1111 (and F=1)
- Or if we want F = a’b’c’d’ + a’b’cd’ + …abcd. (15 minterms)
- Require a single function generator. Program the LUT table to get what we want.

To be implemented

a b c d	F
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
…..	
1 1 1 1	1

This is a 4-variable function generator.

Bit stored in the LUT.



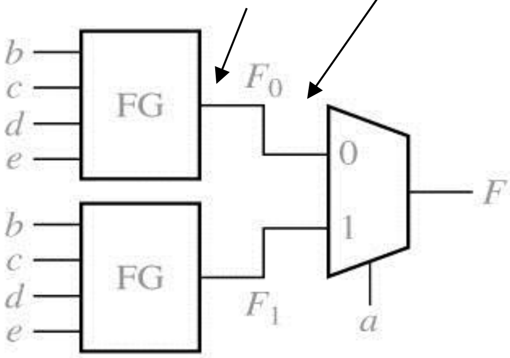
Shannon's Expansion Theorem

- What if # of variables > 4 variables

$$
\begin{aligned}
f(x_1, x_2, \ldots x_n) &= x_i' f(x_1, x_2, .. x_{i-1}, 0, x_{i+1}, \ldots x_n) + \\
&\quad x_i \, f(x_1, x_2, .. x_{i-1}, 1, x_{i+1}, \ldots x_n) \\
&= x_i' f_0 + x_i f_1
\end{aligned}
$$

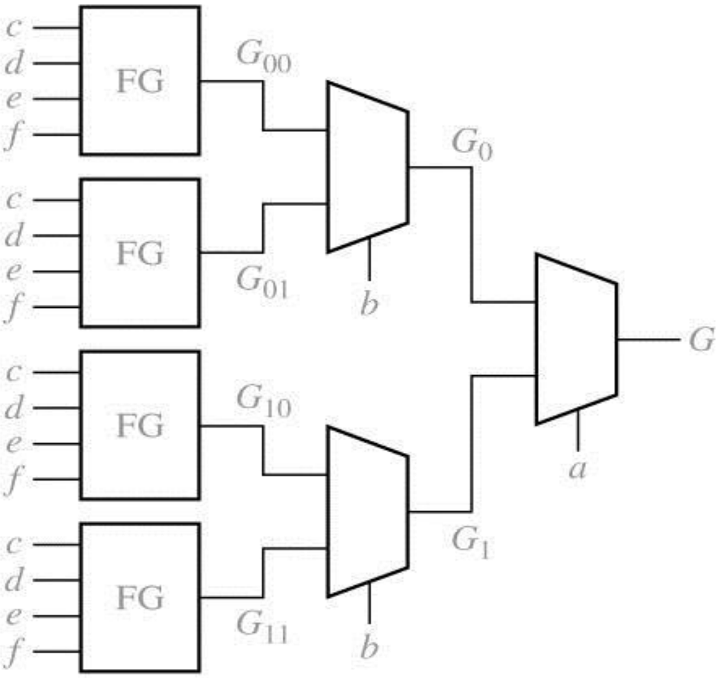
$$f(a, b, c, d, e) = a' f(0, b, c, d, e) + a f(1, b, c, d, e)$$

Let a = 0, what lefts are terms with b, c, d, e



2-1 MUX $F = a'I_1 + a I_2$

(a) 5-variable function



(b) 6-variable function