CHAPTER 9

MULTIPLEXERS, DECODERS, AND PROGRAMMABLE LOGIC DEVICES

Contents

Objectives

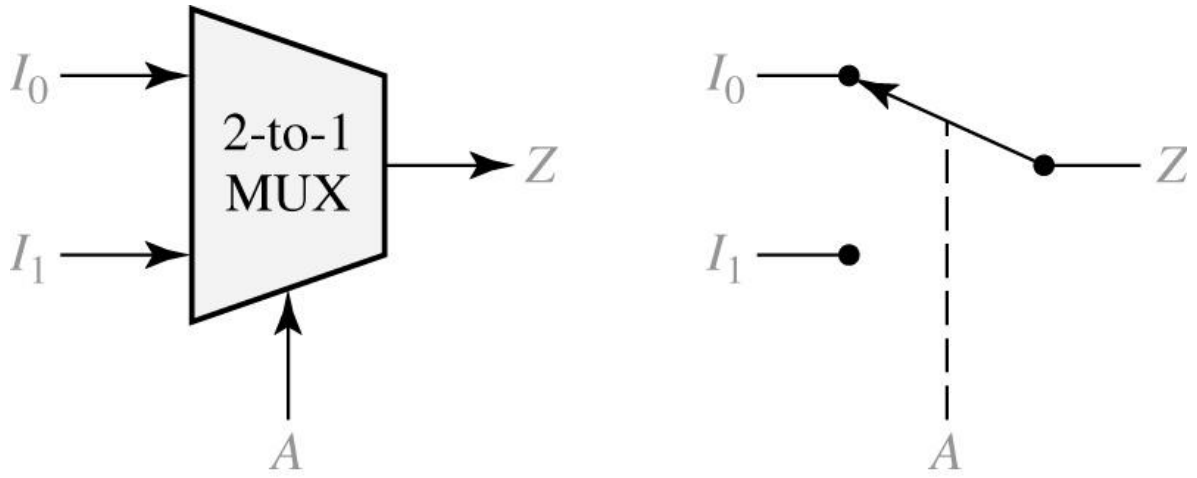
1. Explain the function of a multiplexer. Implement a multiplexer using gates.
2. Explain the operation of three-state buffers. Determine the resulting output when three-state buffers outputs are connected together. Use three-state buffers to multiplex signals onto a bus.
3. Explain the operation of a decoder and encoder. Use a decoder with added gates to implement a set of logic functions. Implement a decoder or priority encoder using gates.
4. Explain the operation of a read-only memory (ROM). Use a ROM to implement a set of logic functions.
5. Explain the operation of a programmable logic array (PLA). Use a PLA to implement a set of logic functions. Given a PLA table or an internal connection diagram for a PLA, determine the logic functions realized.
6. Explain the operation of a programmable array logic device (PAL). Determine the programming pattern required to realize a set of logic function with a PAL.
7. Explain the operation of a complex programmable logic device (CPLD) and a field programmable gate array (FPGA).
8. Use Shannon's expansion theorem to decompose a switching function.

9.1 Introduction

- **Multiplexer, Decoder, encoder. Three-state Buffer**
- **ROMs**
- **PLD**
- **PLA**
- **CPLD**
- **FPGA**

9.2 Multiplexers

Fig 9-1. 2-to-1 Multiplexer and Switch Analog

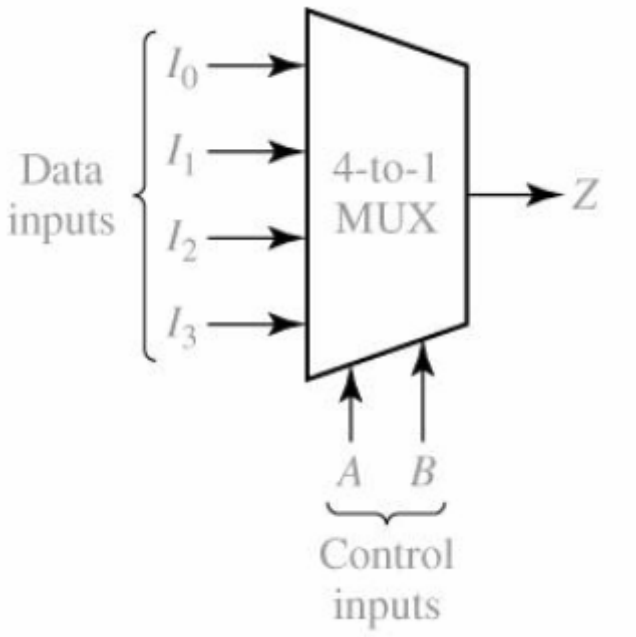


logic equation for the 2-to-1 MUX

$$Z = A' I_0 + A I_1$$

9.2 Multiplexers

Fig 9-2. Multiplexer (1)

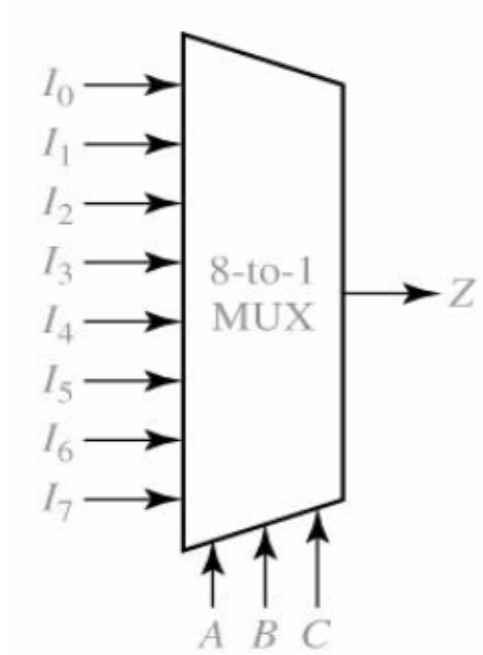


logic equation for the 4-to-1 MUX

$$Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$$

9.2 Multiplexers

Fig 9-2. Multiplexer (2)

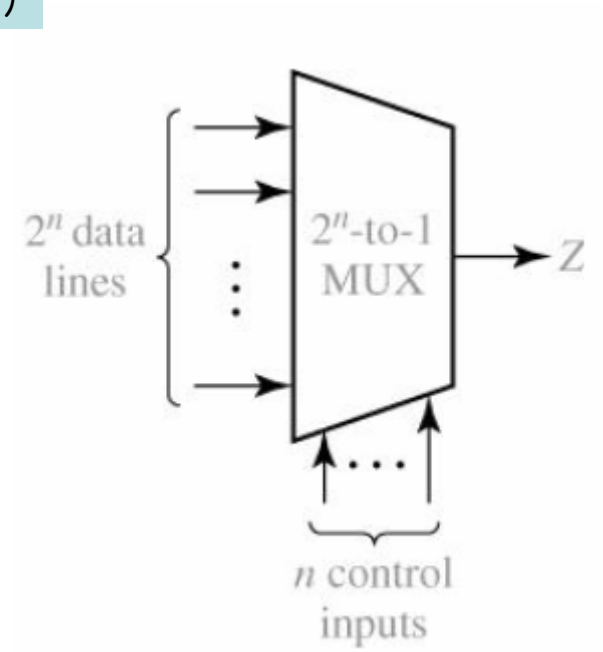


logic equation for the 8 - to - 1 MUX

$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3$$
$$+ AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

9.2 Multiplexers

Fig 9-2. Multiplexer (3)

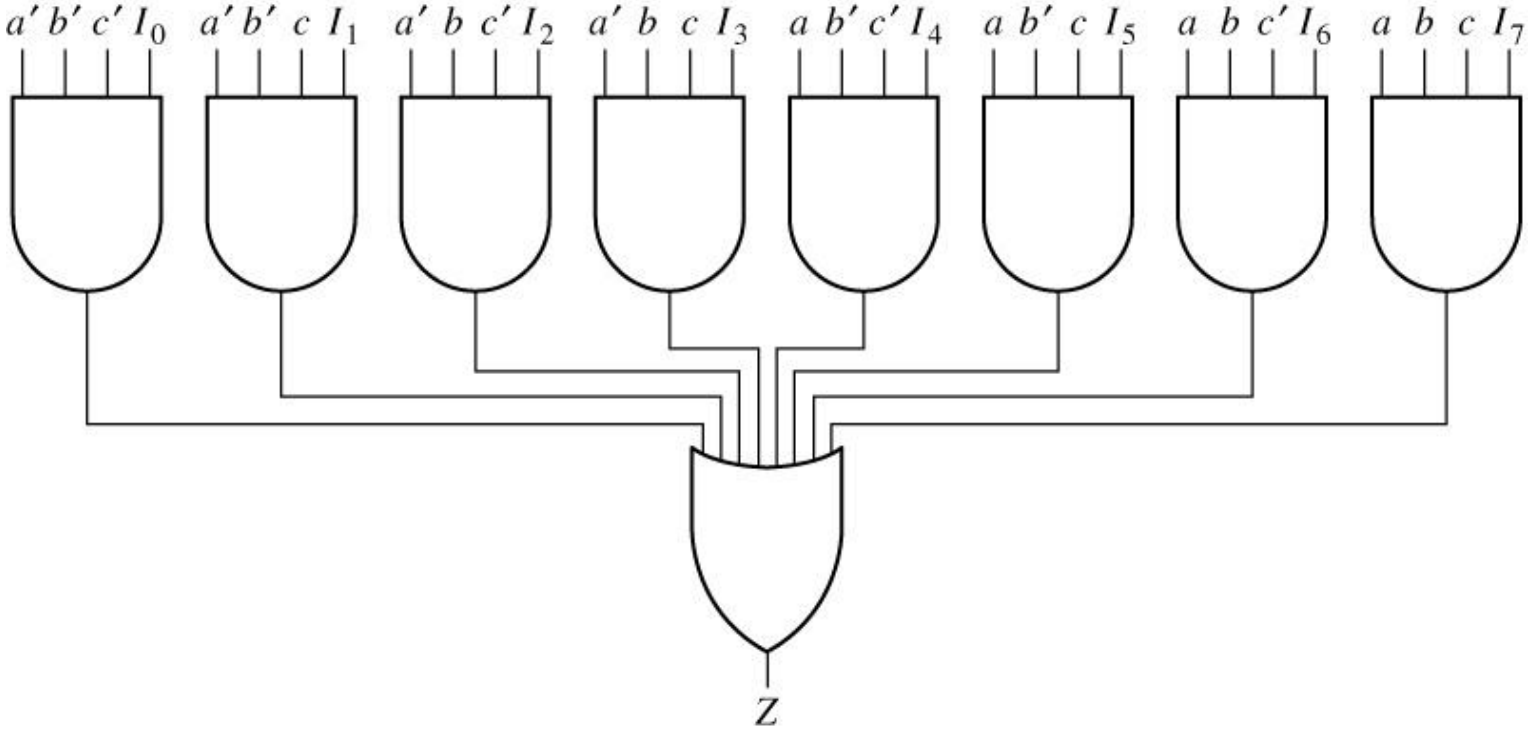


logic equation for the 2^n-to-1 MUX

$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

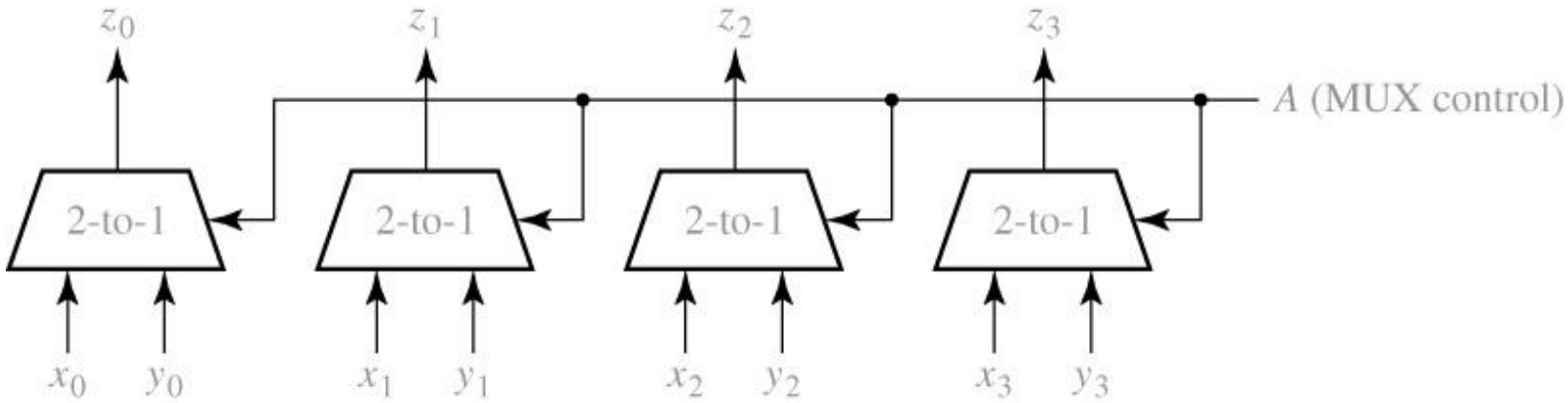
9.2 Multiplexers

Fig 9-3. Logic Diagram for 8-to-1 MUX



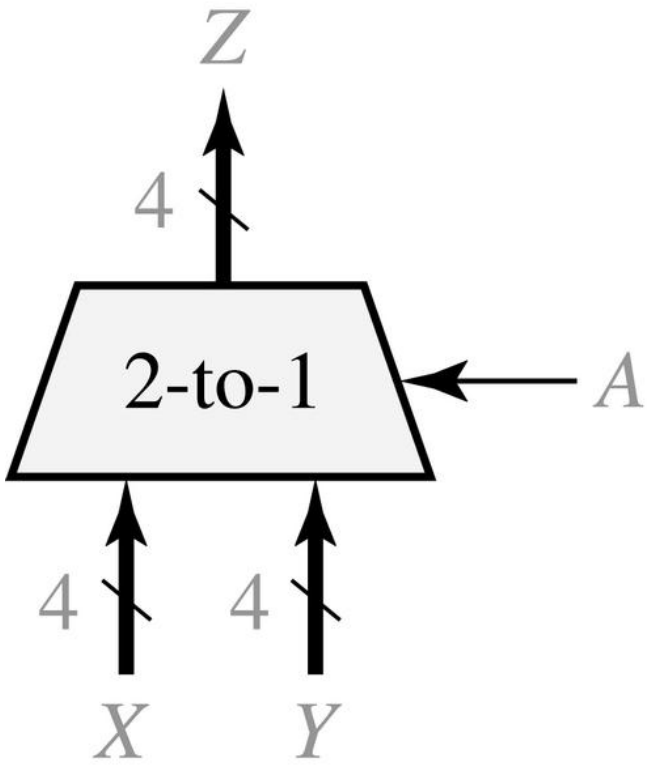
9.2 Multiplexers

Fig 9-4. Quad Multiplexer Used to Select Data



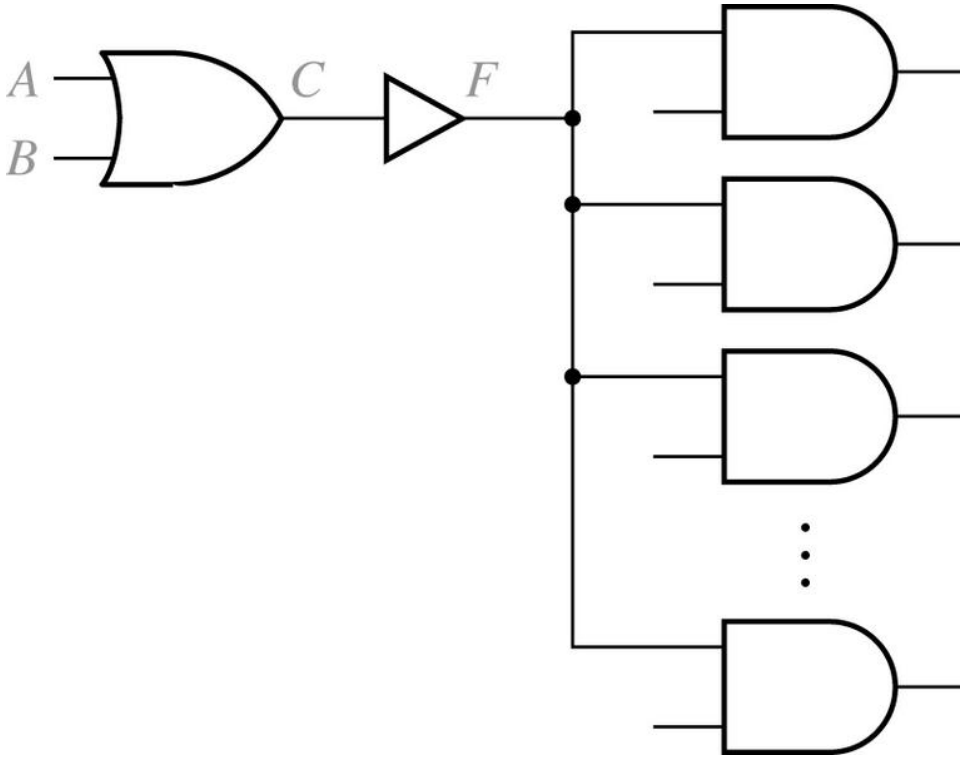
9.2 Multiplexers

Fig 9-5. Quad Multiplexer with Bus Inputs and Output



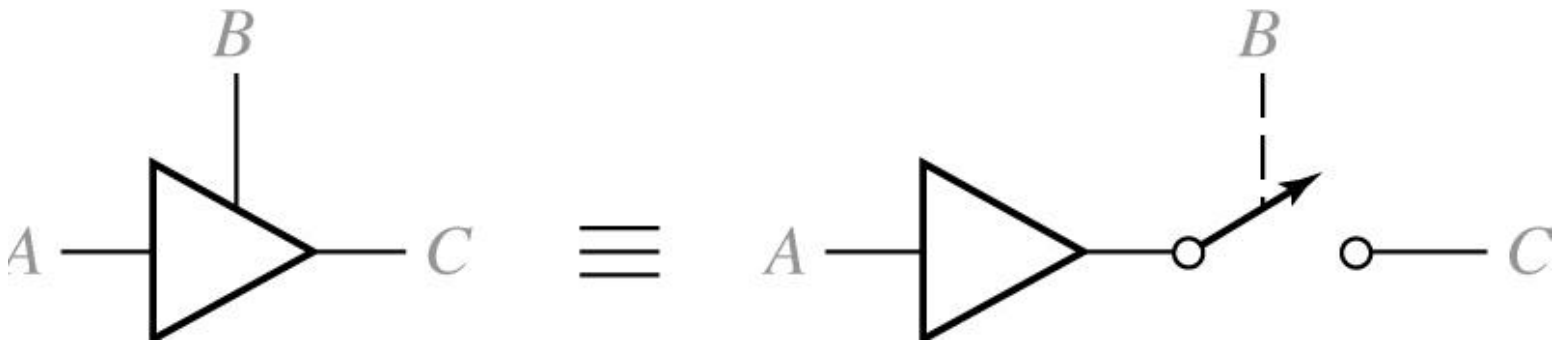
9.3 Three-State Buffers

Fig 9-6. Gate Circuit with Added Buffer



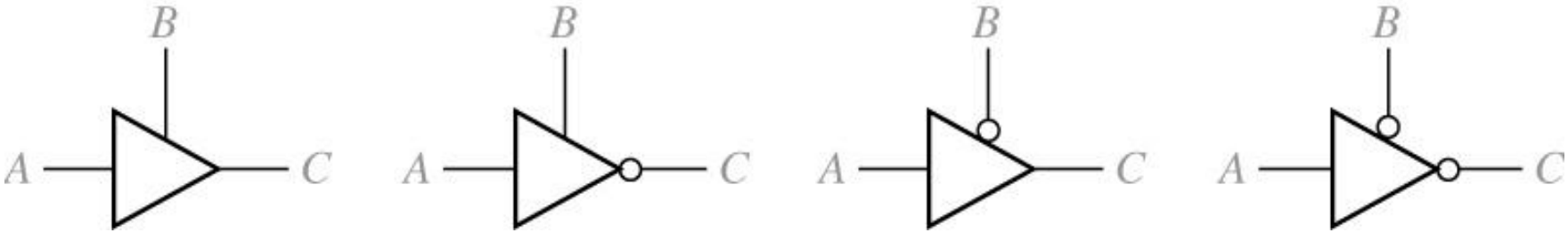
9.3 Three-State Buffers

Fig 9-7. Three-State Buffer



9.3 Three-State Buffers

Fig 9-8. Four Kinds of Three-State Buffers



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

(c)

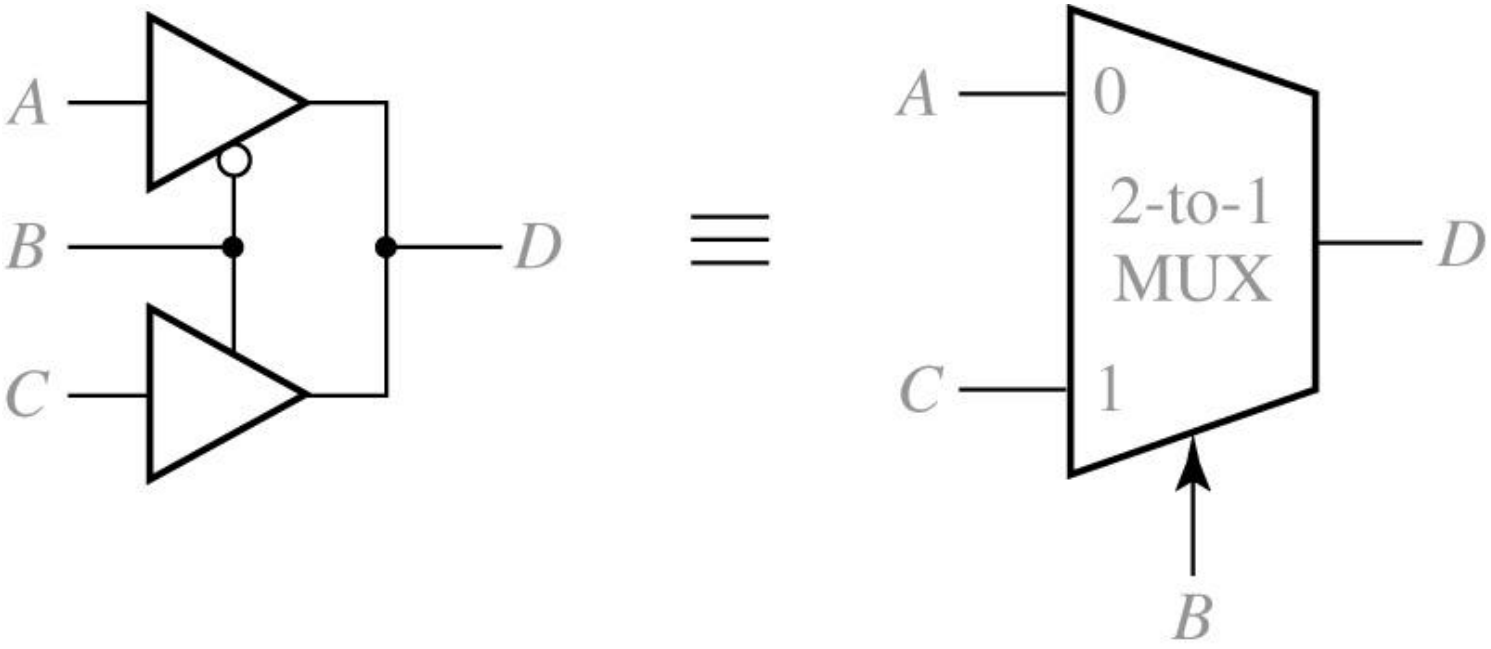
B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

(d)

We use the Symbol Z to represent high-impedance state

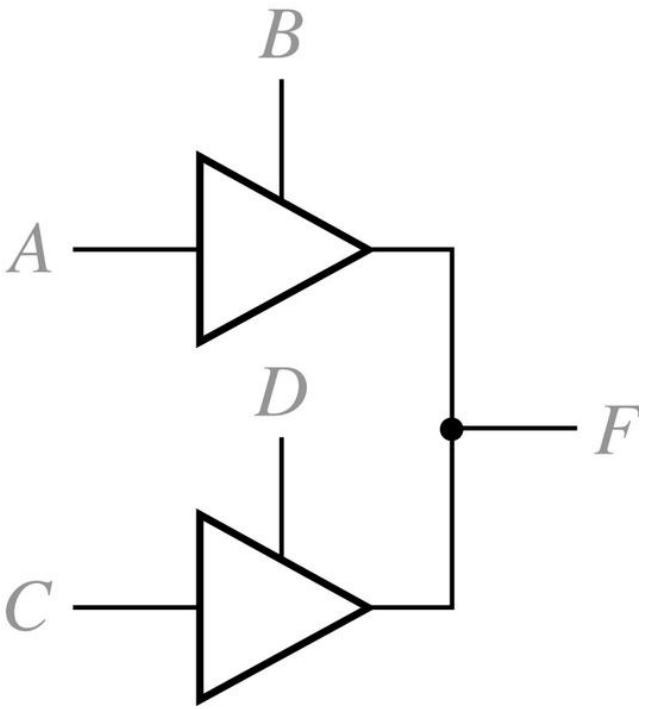
9.3 Three-State Buffers

Fig 9-9. Data Selection Using Three-State Buffers



9.3 Three-State Buffers

Fig 9-10. Circuit with Two Three-State Buffers

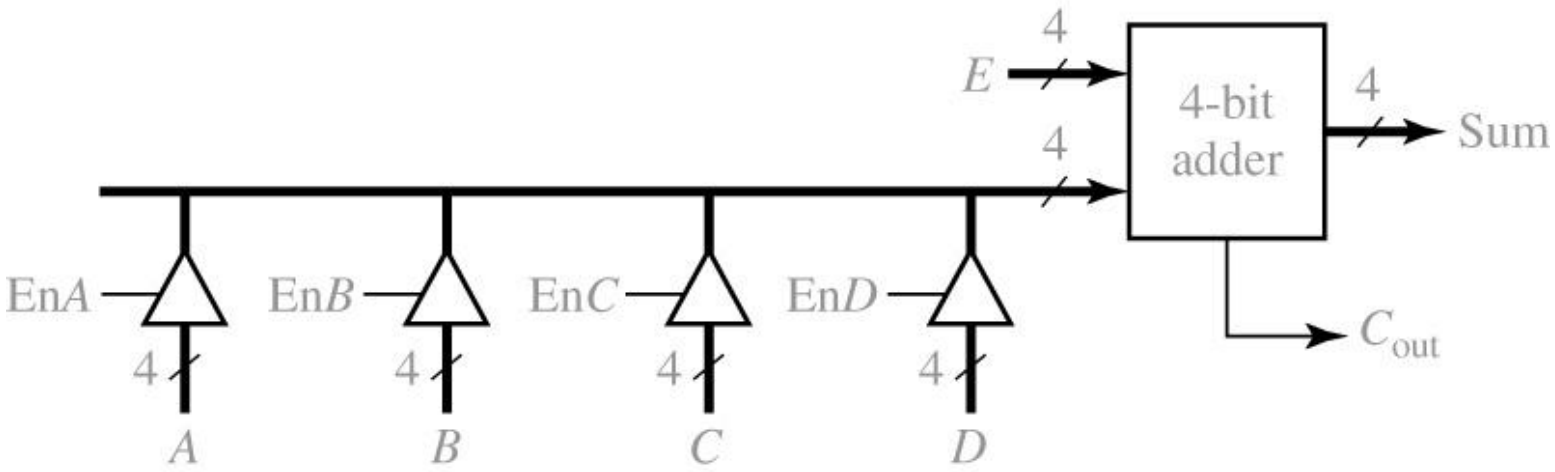


	S2			
S1	X	0	1	Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

X = Unknown

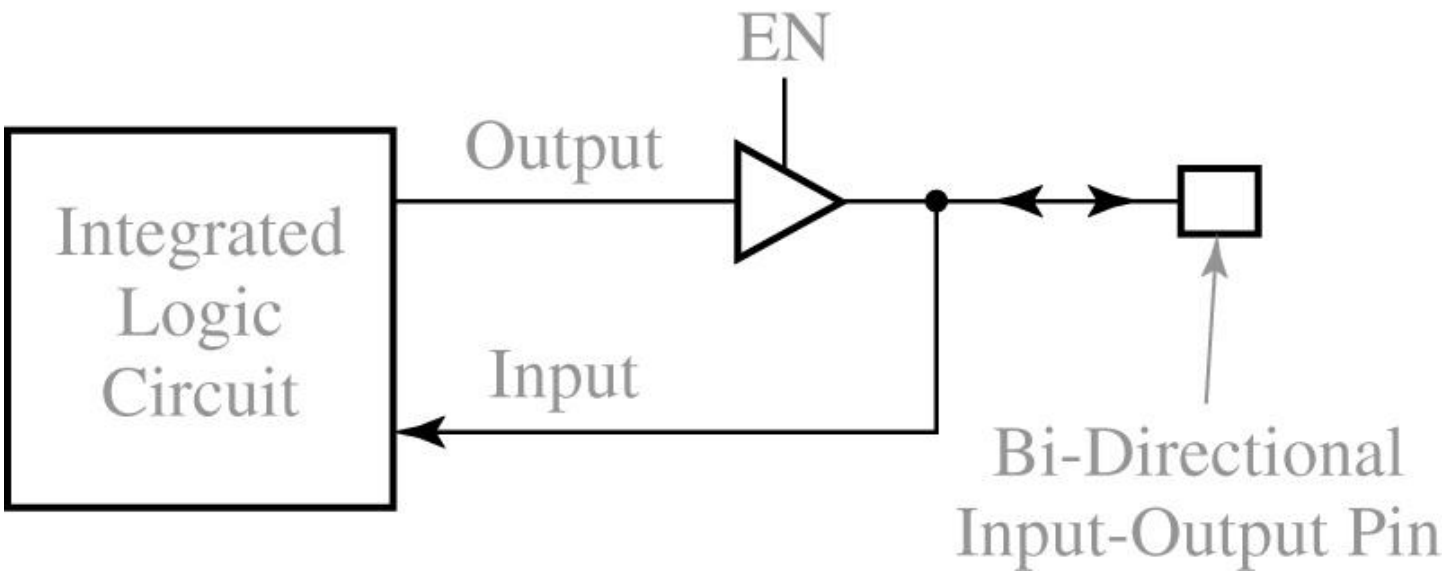
9.3 Three-State Buffers

Fig 9-11. 4-Bit Adder with Four Sources for One Operand



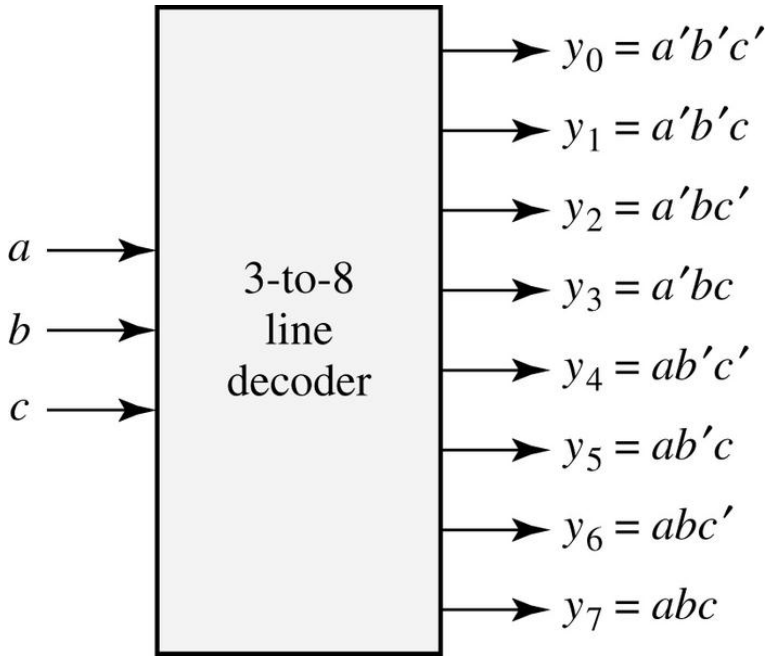
9.3 Three-State Buffers

Fig 9-12. Integrated Circuit with Bi-Directional Input/Output Pin



9.4 Decoders and Encoders

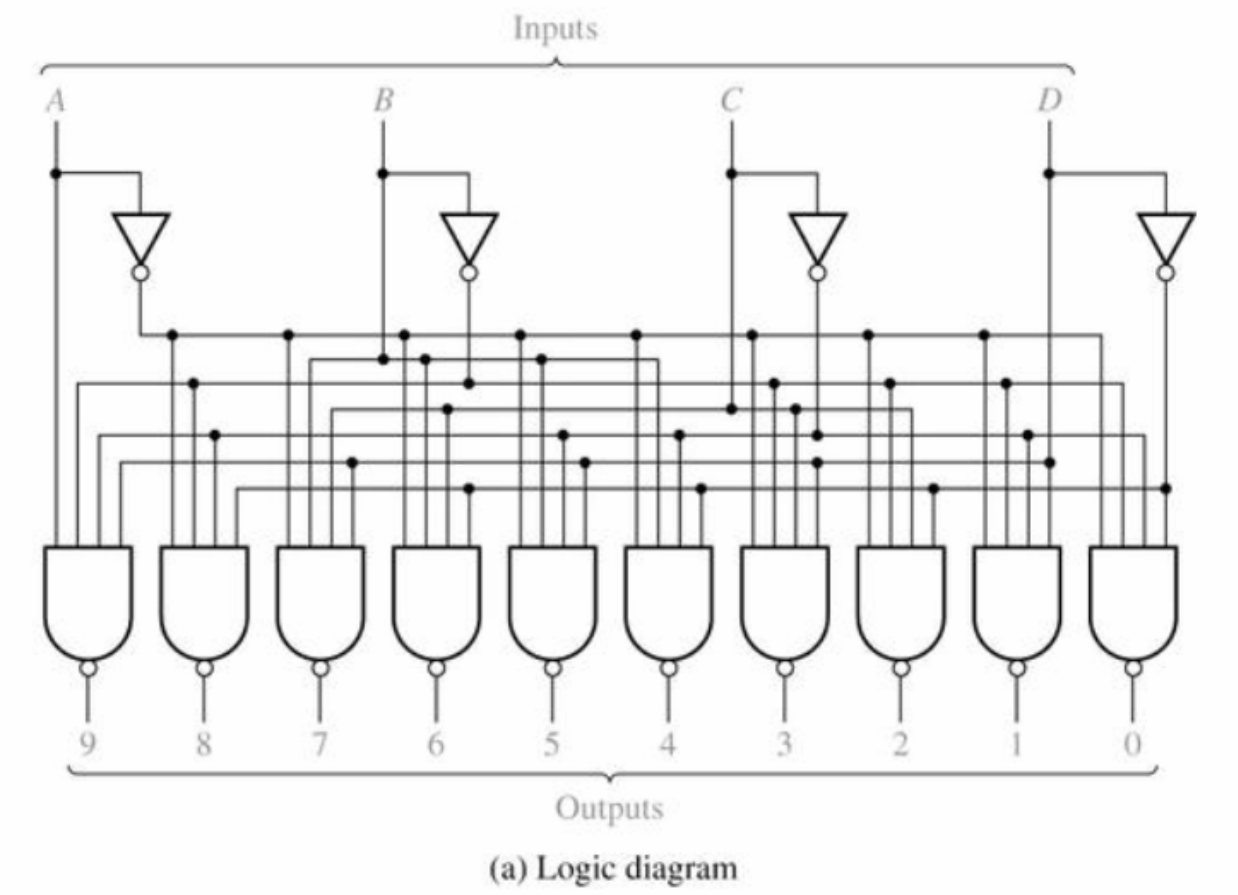
Fig 9-13. 3-to-8 Line Decoder



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

9.4 Decoders and Encoders

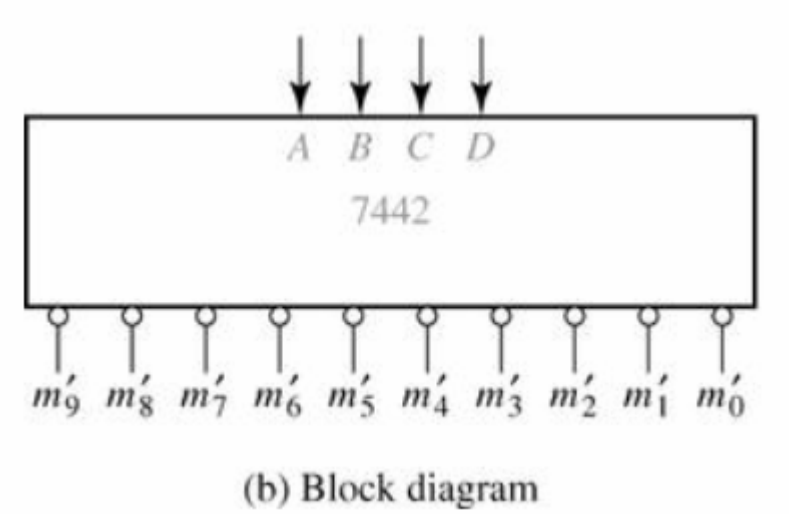
Fig 9-14. A 4-to-10 Line Decoder (1)



(a) Logic diagram

9.4 Decoders and Encoders

Fig 9-14. A 4-to-10 Line Decoder (2)



(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

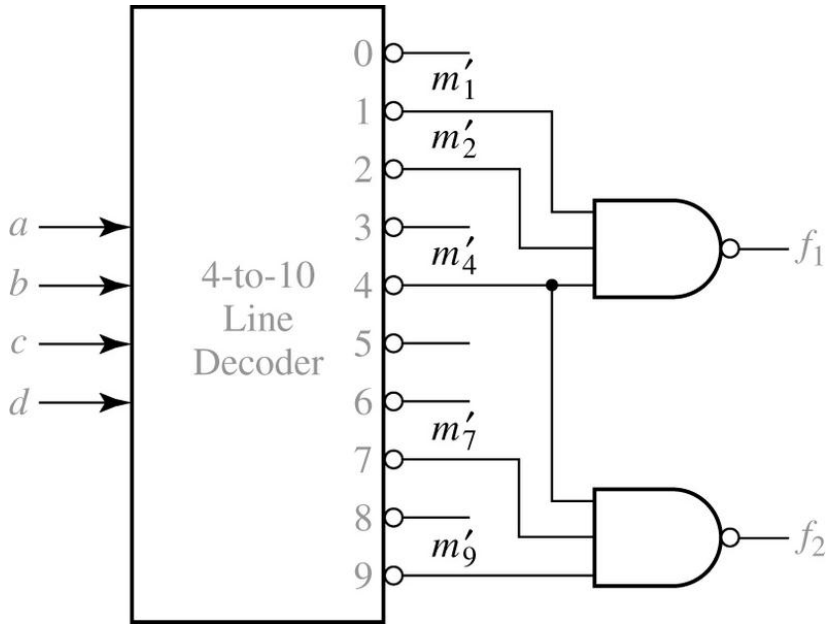
9.4 Decoders and Encoders

Fig 9-15. Realization of a Multiple-Output Circuit Using a Decoder

$$y_i = m_i, \qquad i = 0 \text{ to } 2^n - 1 \qquad \text{(noninverted outputs)}$$

or

$$y_i = m_i' = M_i, \quad i = 0 \text{ to } 2^n - 1 \qquad \text{(inverted outputs)}$$

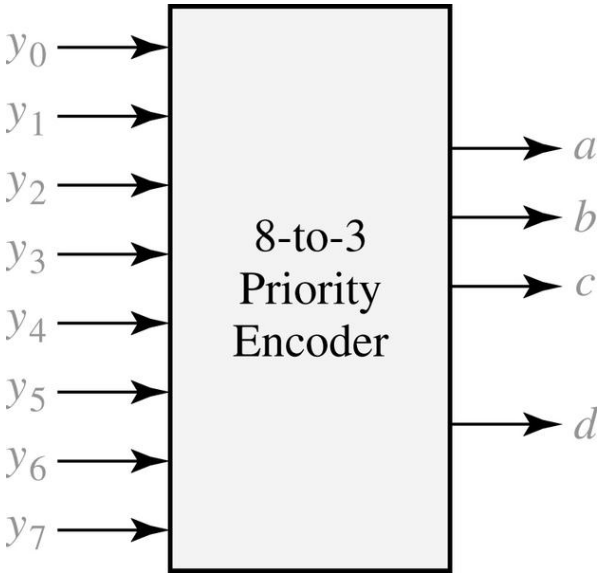


$$f_1(a,b,c,d) = m_1 + m_2 + m_4$$
$$= (m_1' m_2' m_4')'$$

$$f_2(a,b,c,d) = m_4 + m_7 + m_9$$
$$= (m_4' m_7' m_9')'$$

9.4 Decoders and Encoders

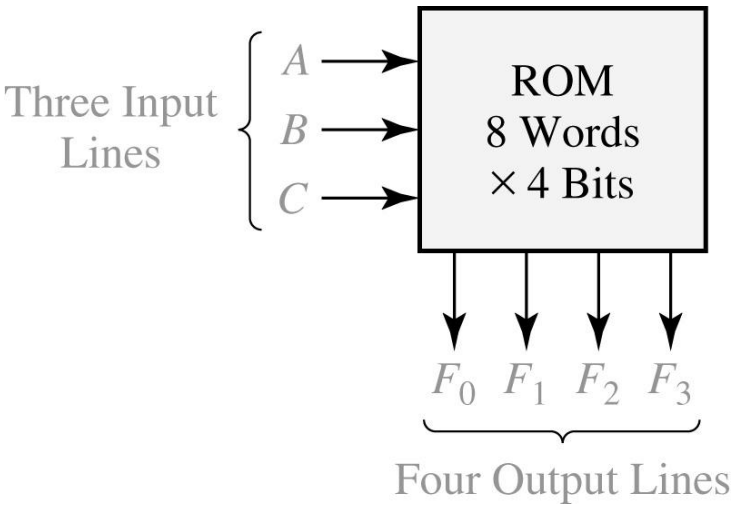
Fig 9-16. 8-to-3 Priority Encoder



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

9.5 Read-Only Memories

Fig 9-17. An 8-Word x 4-Bit ROM



(a) Block diagram

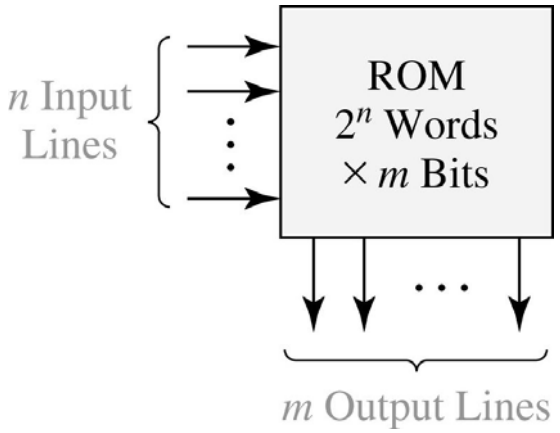
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

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

(b) Truth table for ROM

9.5 Read-Only Memories

Fig 9-18. Read-Only Memory with *n* Inputs and *m* Outputs

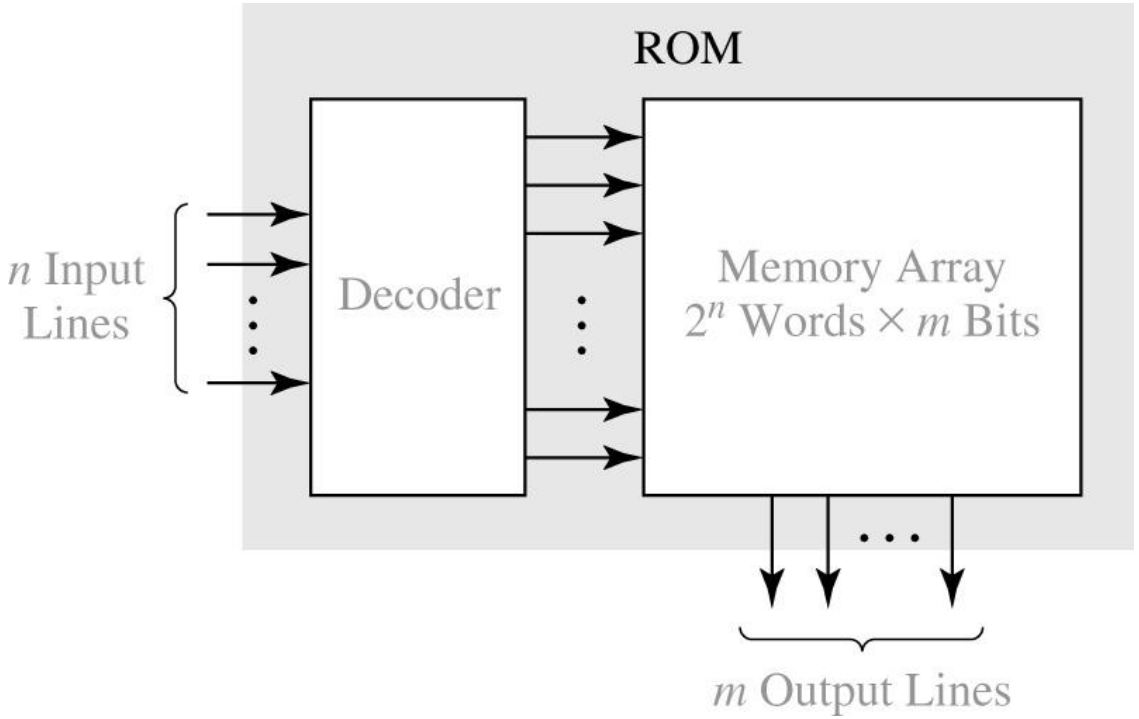


n input Variables	*m* output Variables
00 ···· 00	100 ···· 110
00 ···· 01	010 ···· 111
00 ···· 10	101 ···· 101
00 ···· 11	110 ···· 010
.	. .
.	. .
11 ···· 00	001 ···· 011
11 ···· 01	110 ···· 110
11 ···· 10	011 ···· 000
11 ···· 11	111 ···· 101

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

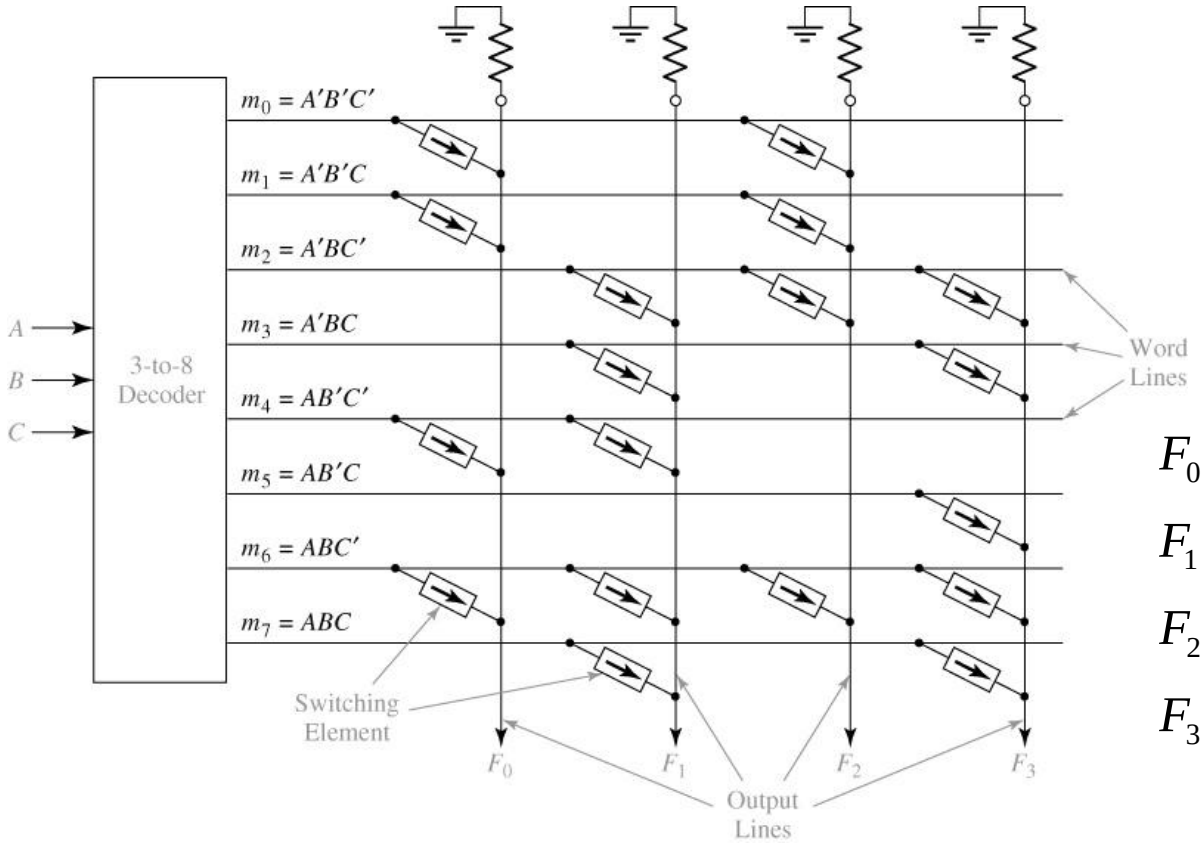
9.5 Read-Only Memories

Fig 9-19. Basic ROM Structure



9.5 Read-Only Memories

Fig 9-20. An 8-Word x 4-Bit ROM



$$F_0 = \sum m(0,1,4,6) = A'B' + AC'$$

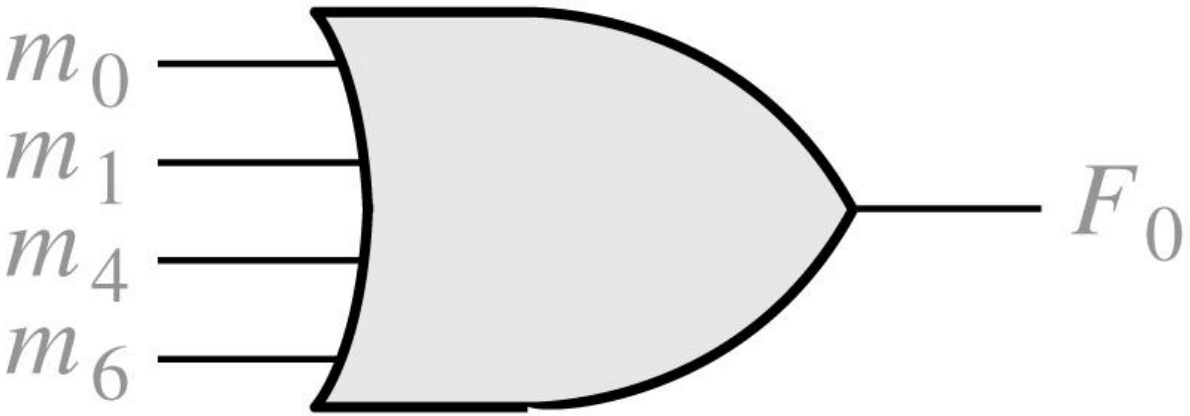
$$F_1 = \sum m(2,3,4,6,7) = B + AC'$$

$$F_2 = \sum m(0,1,2,6) = A'B' + BC'$$

$$F_3 = \sum m(2,3,5,6,7) = AC + B$$

9.5 Read-Only Memories

Fig 9-21. Equivalent OR Gate for F_0

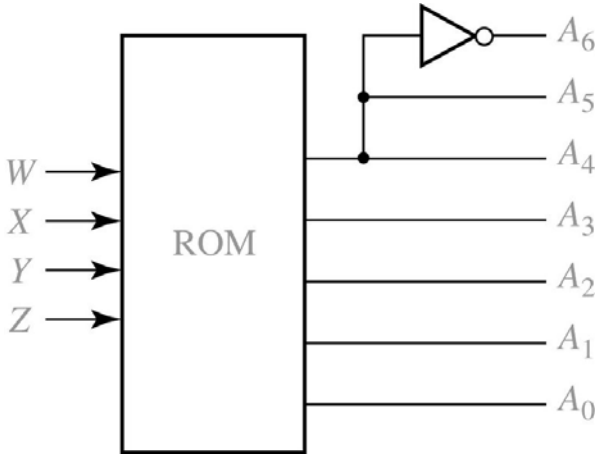


$$F_0 = \sum m(0,1,4,6) = A'B' + AC'$$

9.5 Read-Only Memories

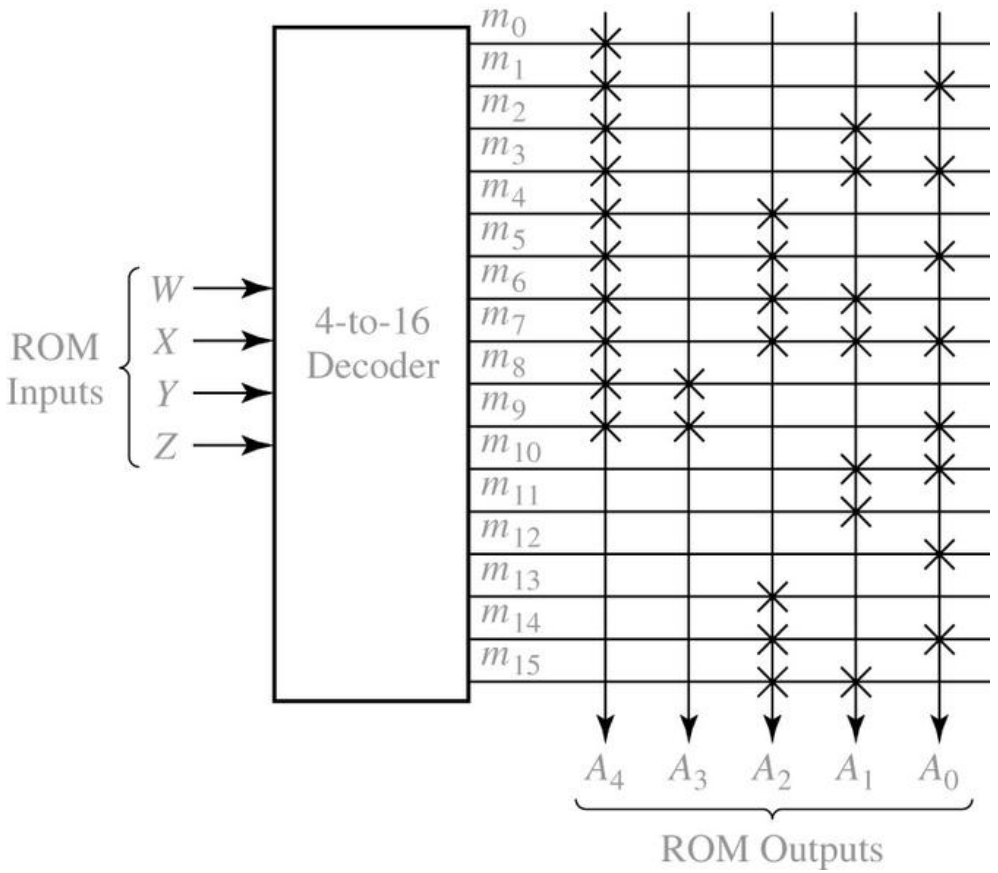
Fig 9-22. Hexadecimal to ASCII Code Converter

Input				Hex Digit	ASCII Code for Hex Digit						
W	X	Y	Z		A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	1	1	0	0	0	1
0	0	1	0	2	0	1	1	0	0	1	0
0	0	1	1	3	0	1	1	0	0	1	1
0	1	0	0	4	0	1	1	0	1	0	0
0	1	0	1	5	0	1	1	0	1	0	1
0	1	1	0	6	0	1	1	0	1	1	0
0	1	1	1	7	0	1	1	0	1	1	1
1	0	0	0	8	0	1	1	1	0	0	0
1	0	0	1	9	0	1	1	1	0	0	1
1	0	1	0	A	1	0	0	0	0	0	1
1	0	1	1	B	1	0	0	0	0	1	0
1	1	0	0	C	1	0	0	0	0	1	1
1	1	0	1	D	1	0	0	0	1	0	0
1	1	1	0	E	1	0	0	0	1	0	1
1	1	1	1	F	1	0	0	0	1	1	0



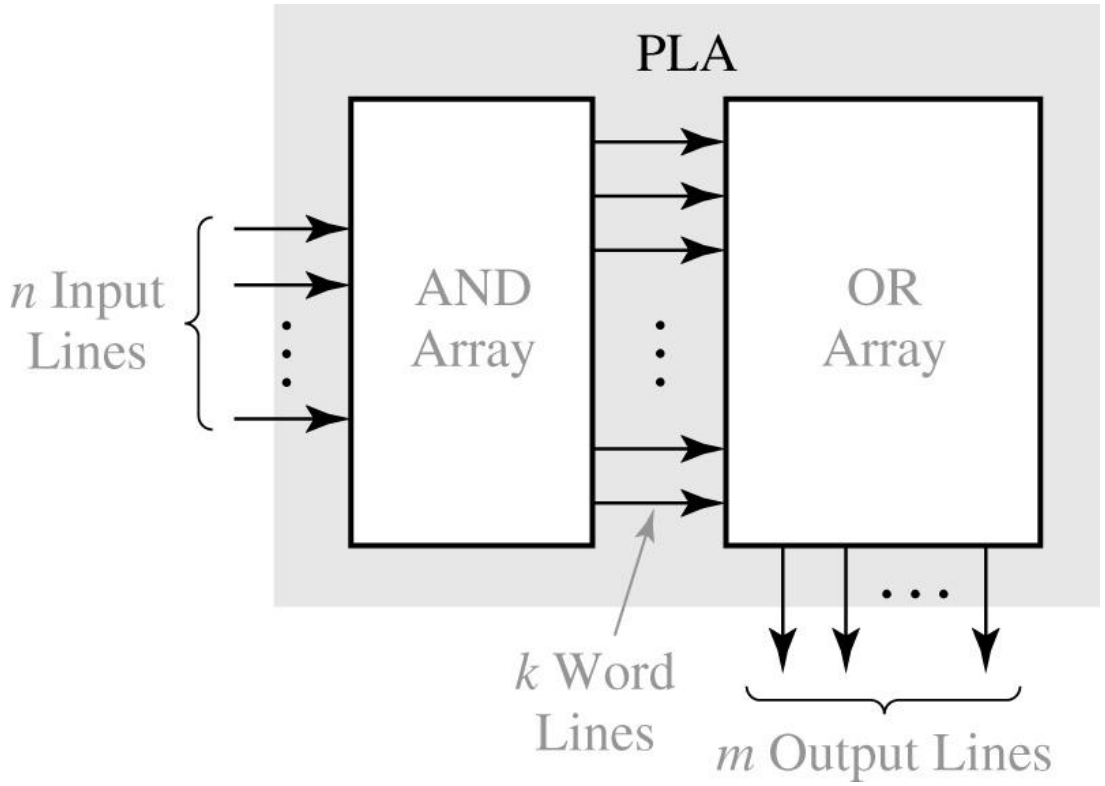
9.5 Read-Only Memories

Fig 9-23. ROM Realization of Code Converter



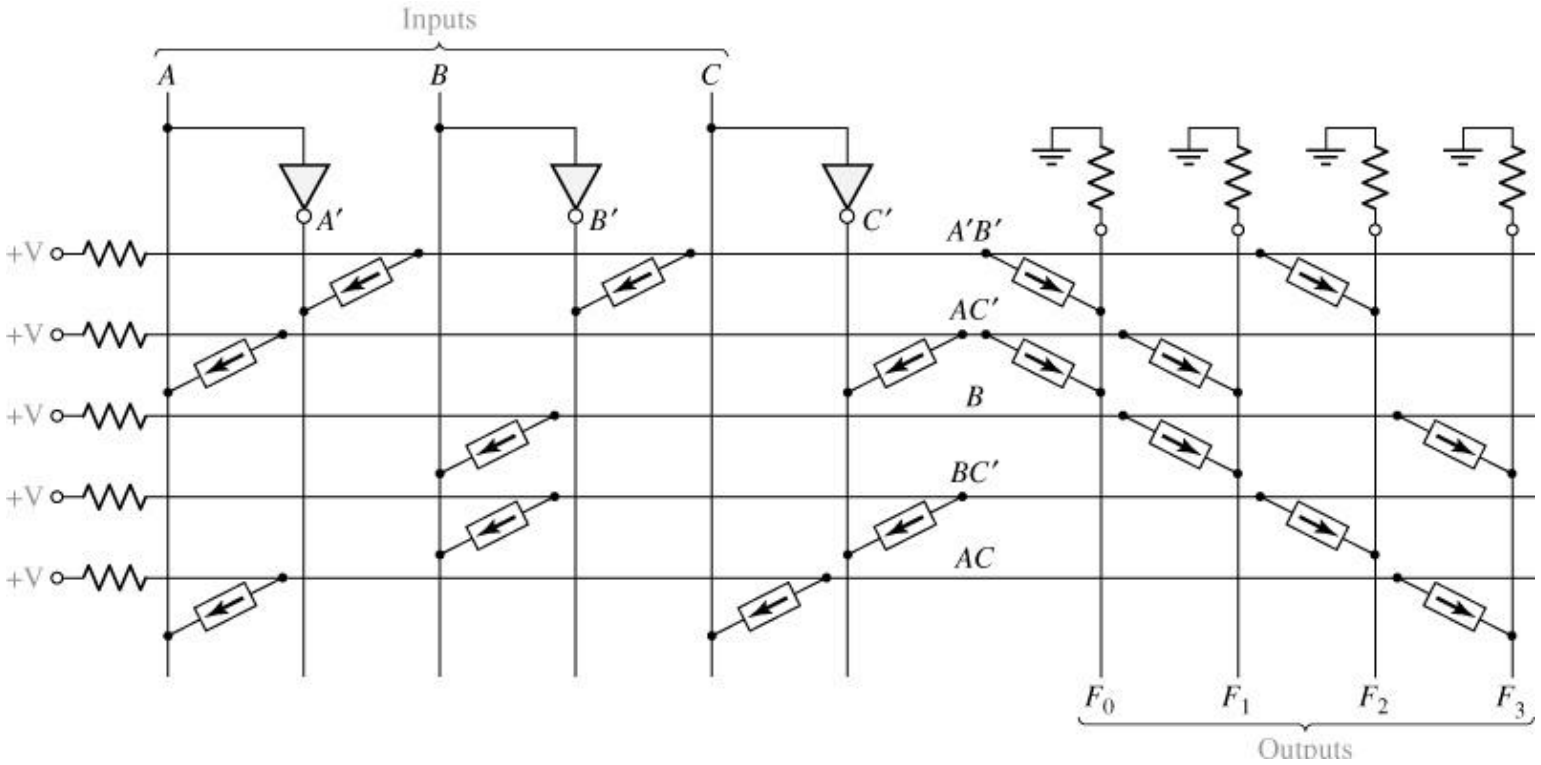
9.6 Programmable Logic Devices

Fig 9-24. Programmable Logic Array Structure



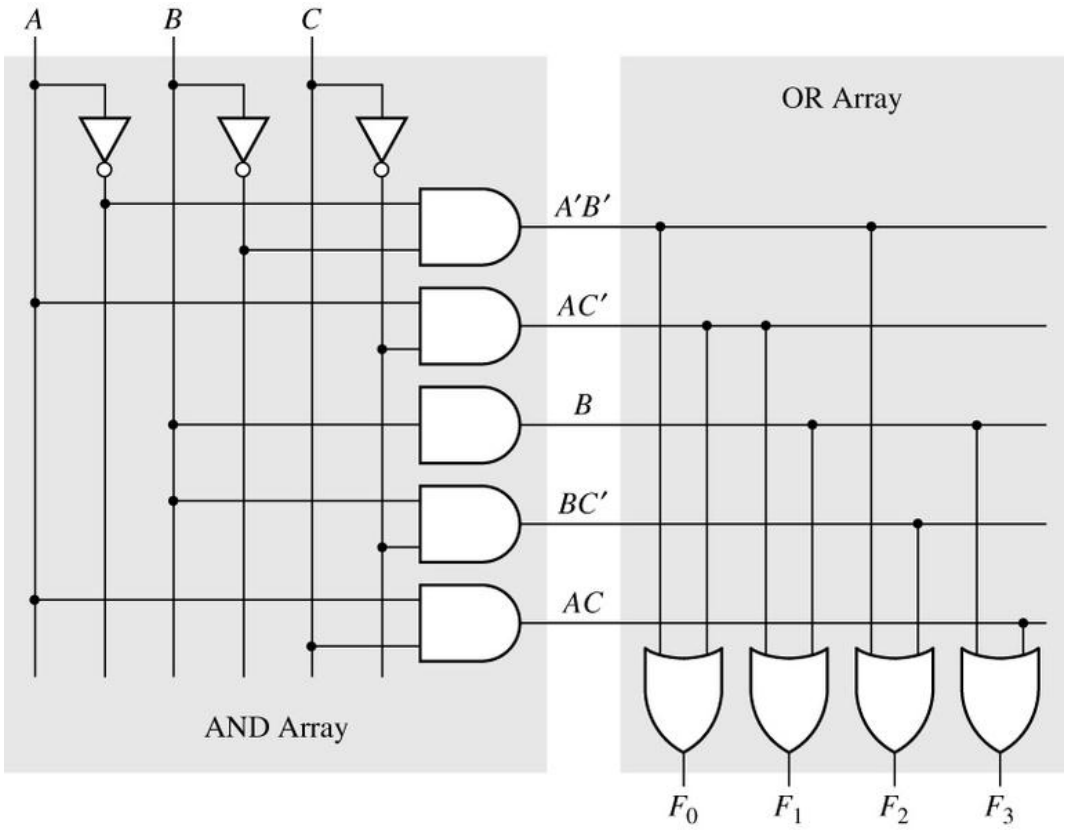
9.6 Programmable Logic Devices

Fig 9-25. PLA with Three Inputs, Five Product Terms, and Four Outputs



9.6 Programmable Logic Devices

Fig 9-26. AND-OR Array Equivalent to Figure 9-25



9.6 Programmable Logic Devices

Table 9-1. PLA Table for Figure 9-25

Product Term	Inputs			Outputs			
	A	B	C	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

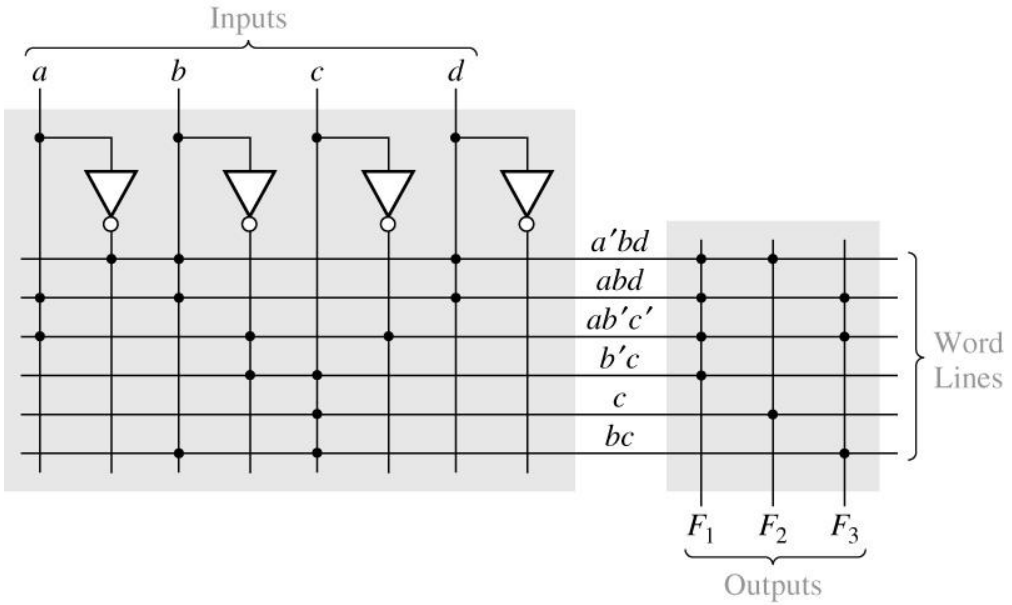
$$F_0 = A'B' + AC'$$
$$F_1 = AC' + B$$
$$F_2 = A'B' + BC'$$
$$F_3 = B + AC$$

9.6 Programmable Logic Devices

Fig 9-27. PLA Realization of Equations (7-23b)

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

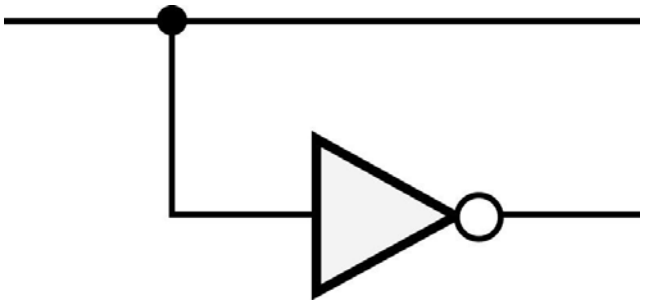


(b) PLA structure

9.6 Programmable Logic Devices

Programmable Array Logic

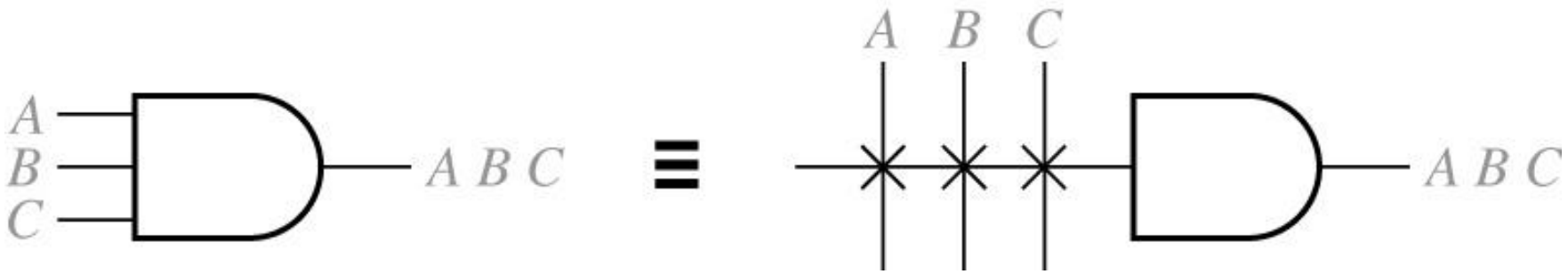
The symbol of Figure 9-28(a)



logically equal

9.6 Programmable Logic Devices

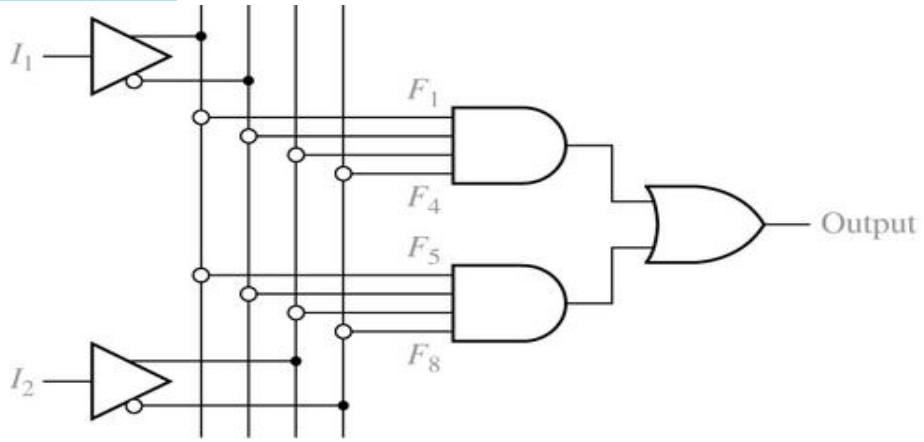
Programmable Array Logic



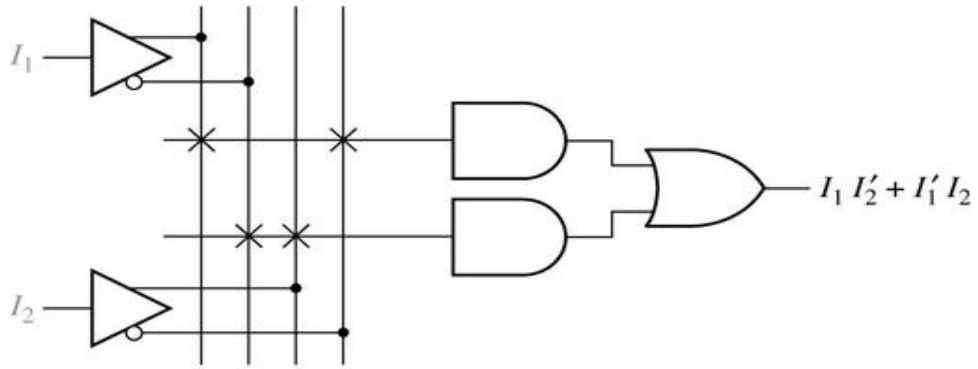
Connections to the AND gate inputs in a PAL

9.6 Programmable Logic Devices

Fig 9-28. PAL Segment



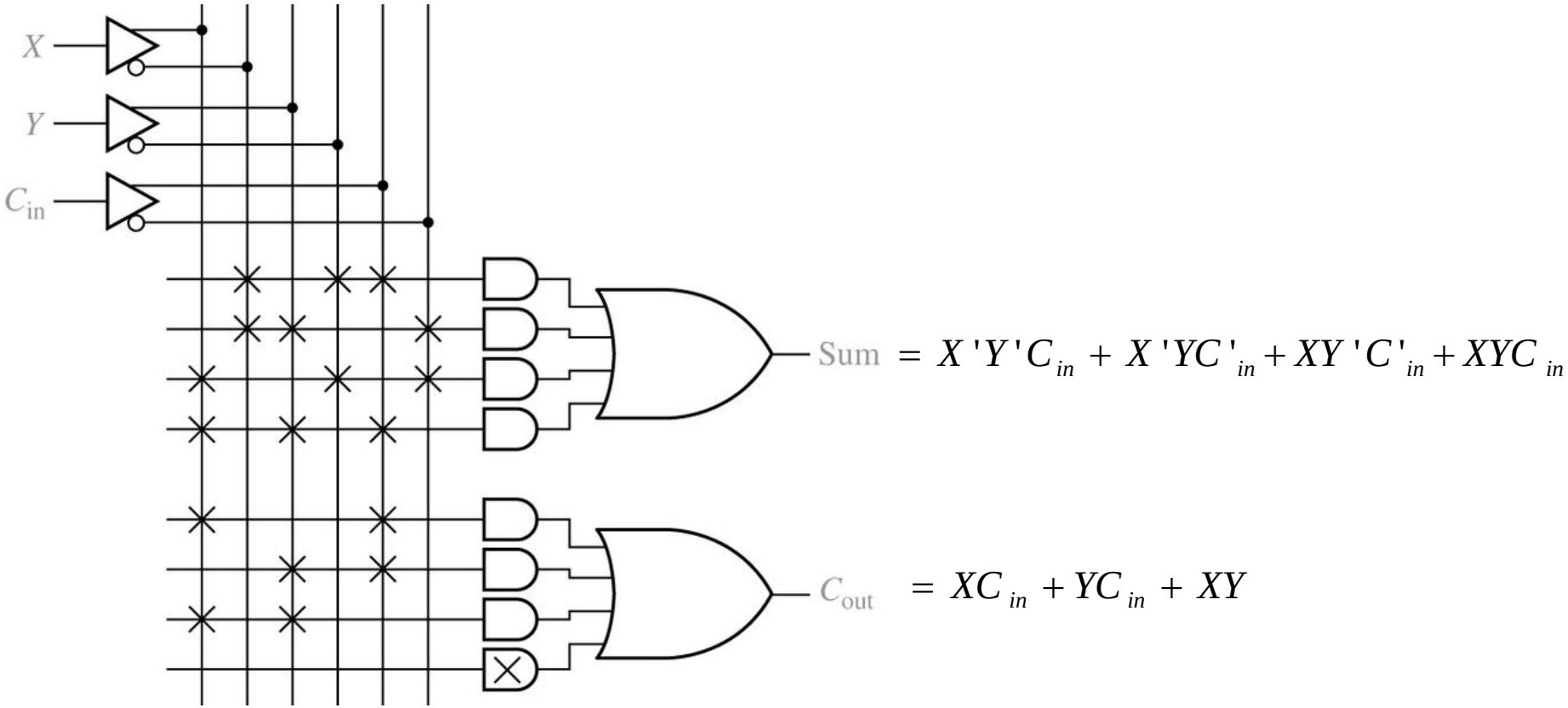
(a) Unprogrammed



(b) Programmed

9.6 Programmable Logic Devices

Fig 9-29. Implementation of a Full Adder Using a PAL

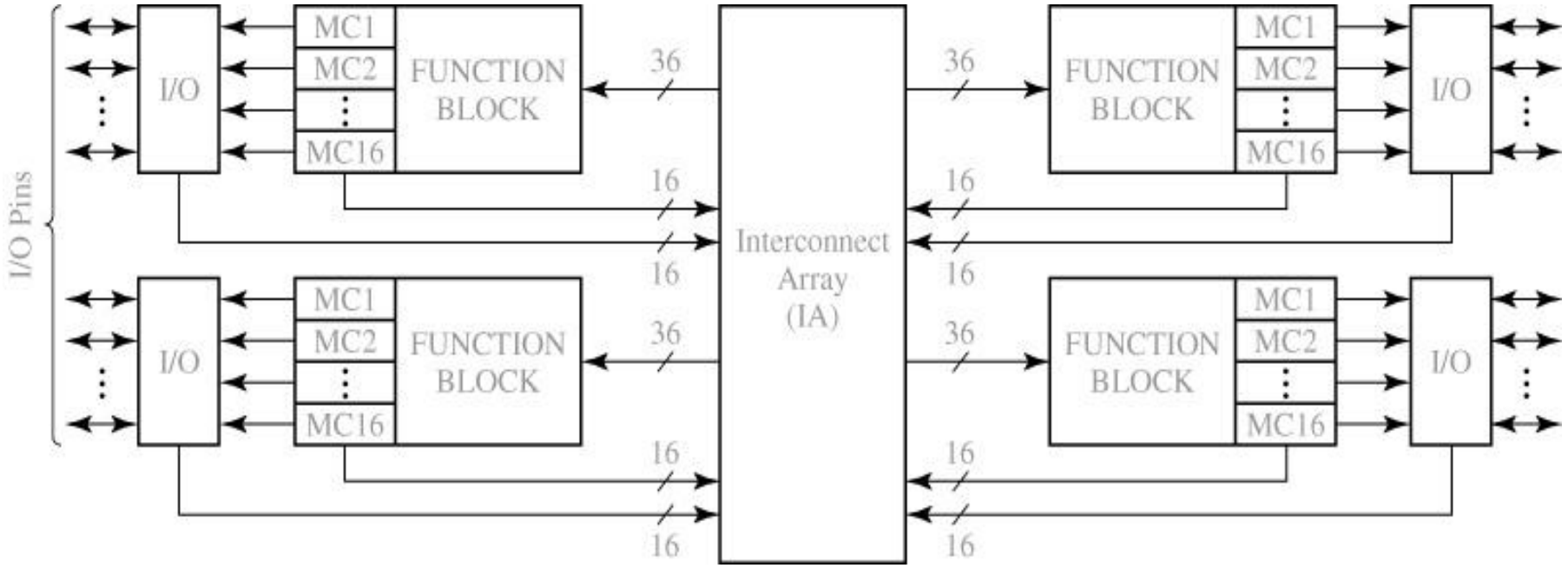


$$\text{Sum} = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in}$$

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

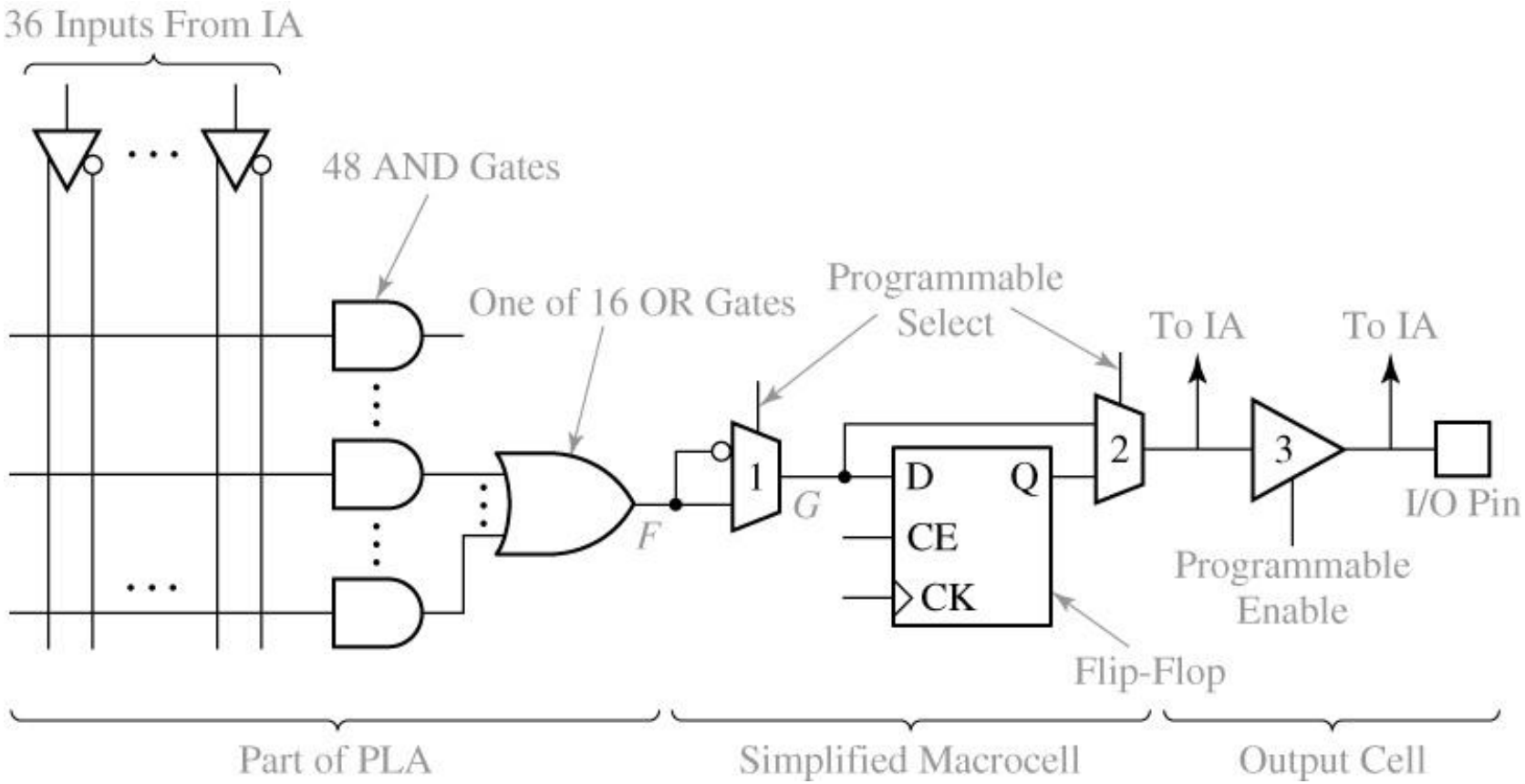
9.7 Complex Programmable Logic Devices

Fig 9-30. Architecture of Xilinx XCR3064XL CPLD



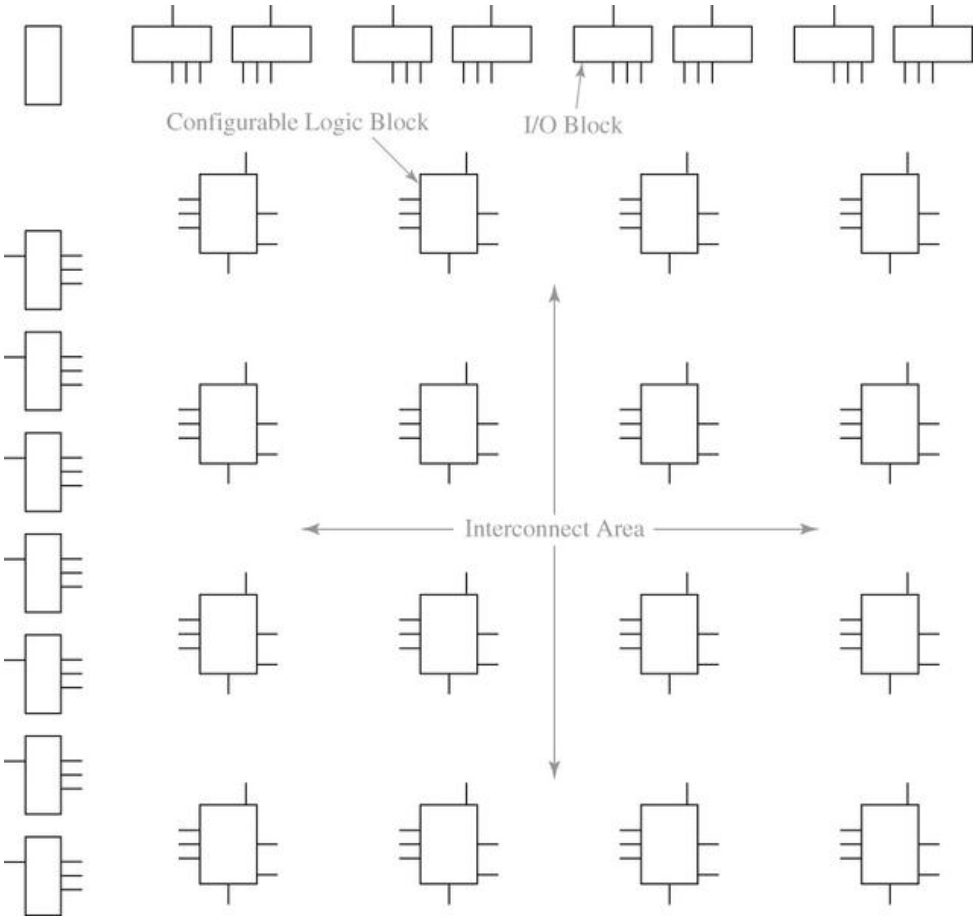
9.7 Complex Programmable Logic Devices

Fig 9-31. CPLD Function Block and Macrocell (A Simplified Version of XCR3064XL)



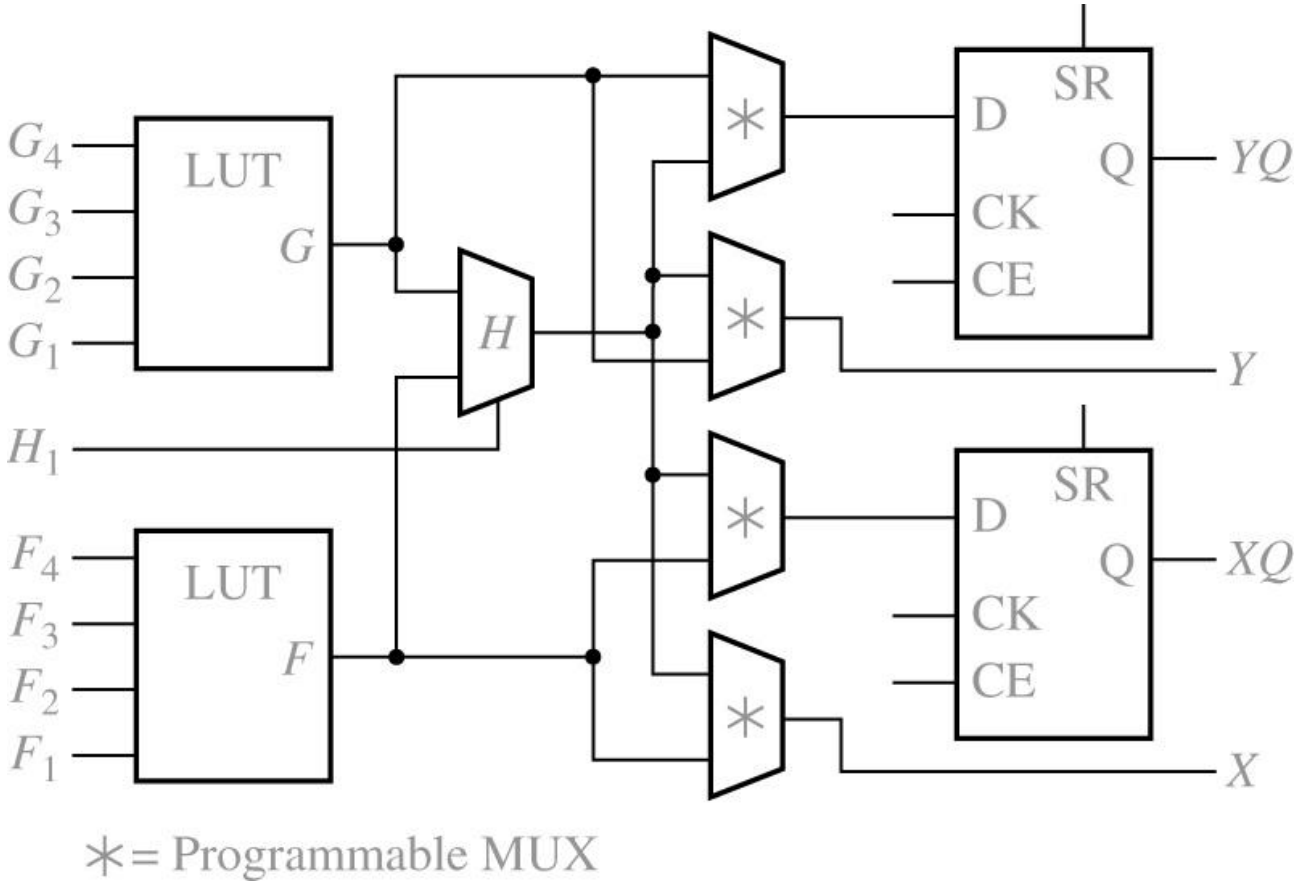
9.8 Field Programmable Gate Arrays

Fig 9-32. Equivalent OR Gate for F_0



9.8 Field Programmable Gate Arrays

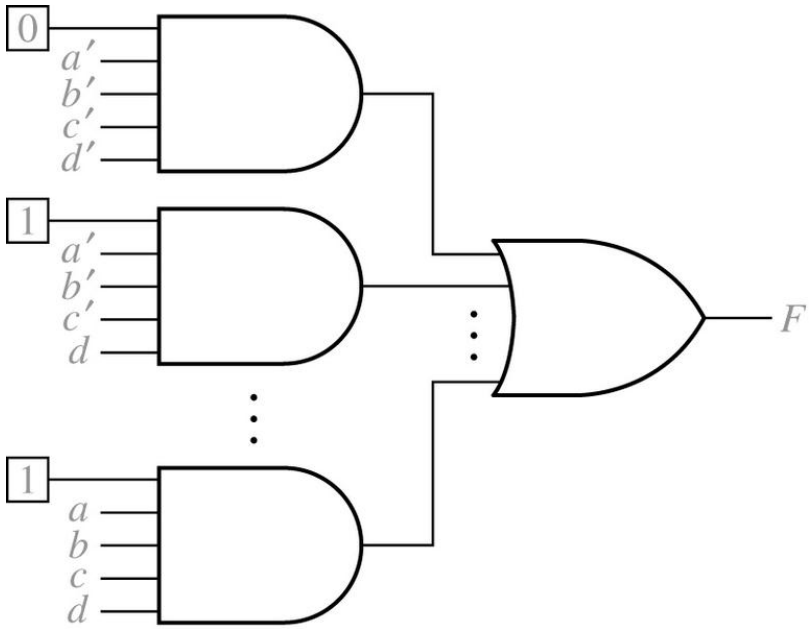
Fig 9-33. Simplified Configurable Logic Block (CLB)



9.8 Field Programmable Gate Arrays

Fig 9-34. Implementation of a Lookup Table (LUT)

a	b	c	d	f
0	0	0	0	0
0	0	0	1	1
.	.	.	.	.
.	.	.	.	.
1	1	1	1	1



$$F = a'b'c'd' + a'b'cd + a'bc'd + a'bcd' + ab'c'd + ab'cd' + abc'd' + abcd$$

9.8 Field Programmable Gate Arrays

Decomposition if switching Functions

$$f(a,b,c,d) = a'\, f(0,b,c,d) + af(1,b,c,d) = a'\, f_0 + af_1$$

$$\begin{aligned} f(a,b,c,d) &= c'd'+a'b'c+bcd+ac' \\ &= a'(c'd'+b'c+bcd)+a(c'd'+bcd+c') \\ &= a'(c'd'+b'c+cd)+a(c'+bd) = a'\, f_0 + af_1 \end{aligned}$$

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

9.8 Field Programmable Gate Arrays

Decomposition if switching Functions

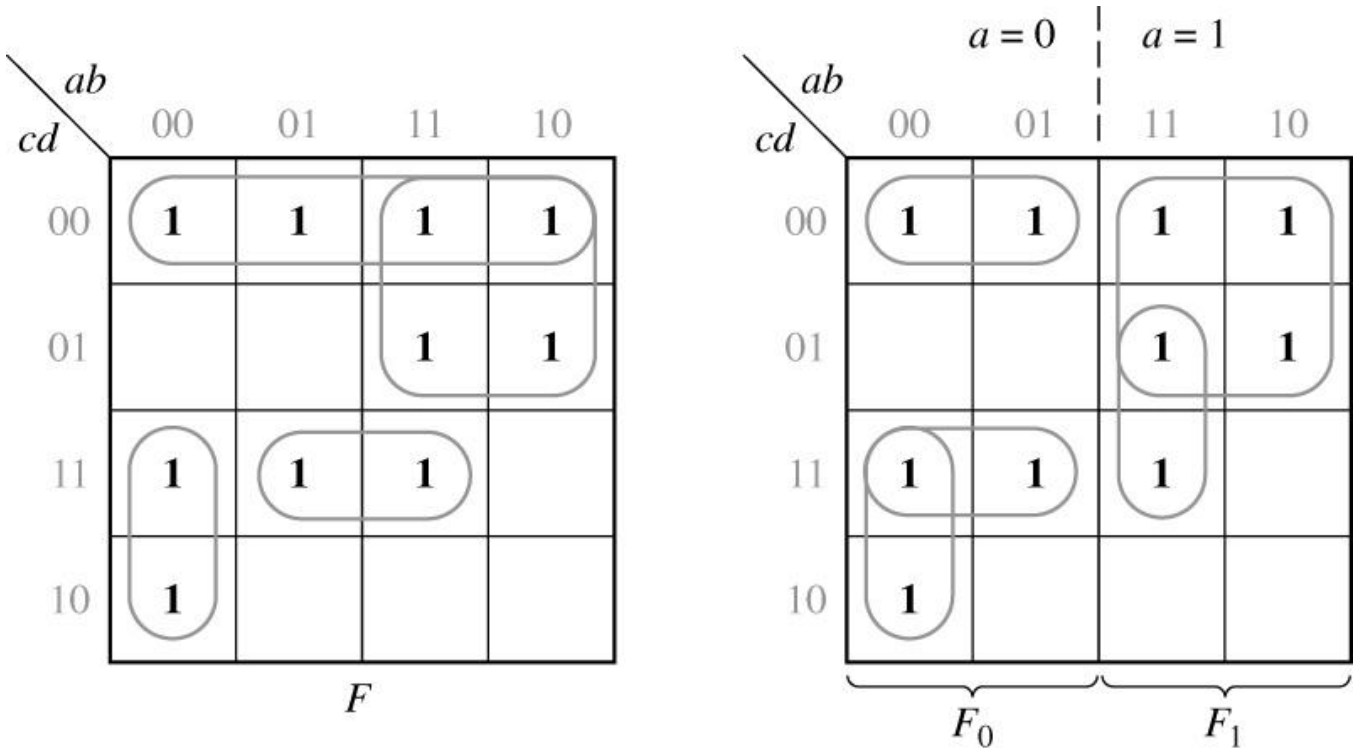
$$f(a,b,c,d,e) = a'\, f(0,b,c,d,e) + af(1,b,c,d,e) = a'\, f_0 + af_1$$

$$G(a,b,c,d,e,\, f) = a'G(0,b,c,d,e,\, f) + aG(1,b,c,d,e,\, f) = a'G_0 + aG_1$$
$$G_0 = b'G(0,0,c,d,e,\, f) + bG(0,1,c,d,e,\, f) = b'G_{00} + bG_{01}$$
$$G_1 = b'G(1,0,c,d,e,\, f) + bG(1,1,c,d,e,\, f) = b'G_{10} + bG_{11}$$

$$G(a,b,c,d,e,\, f) = a'b'G_{00} + a'bG_{01} + ab'G_{10} + abG_{11}$$

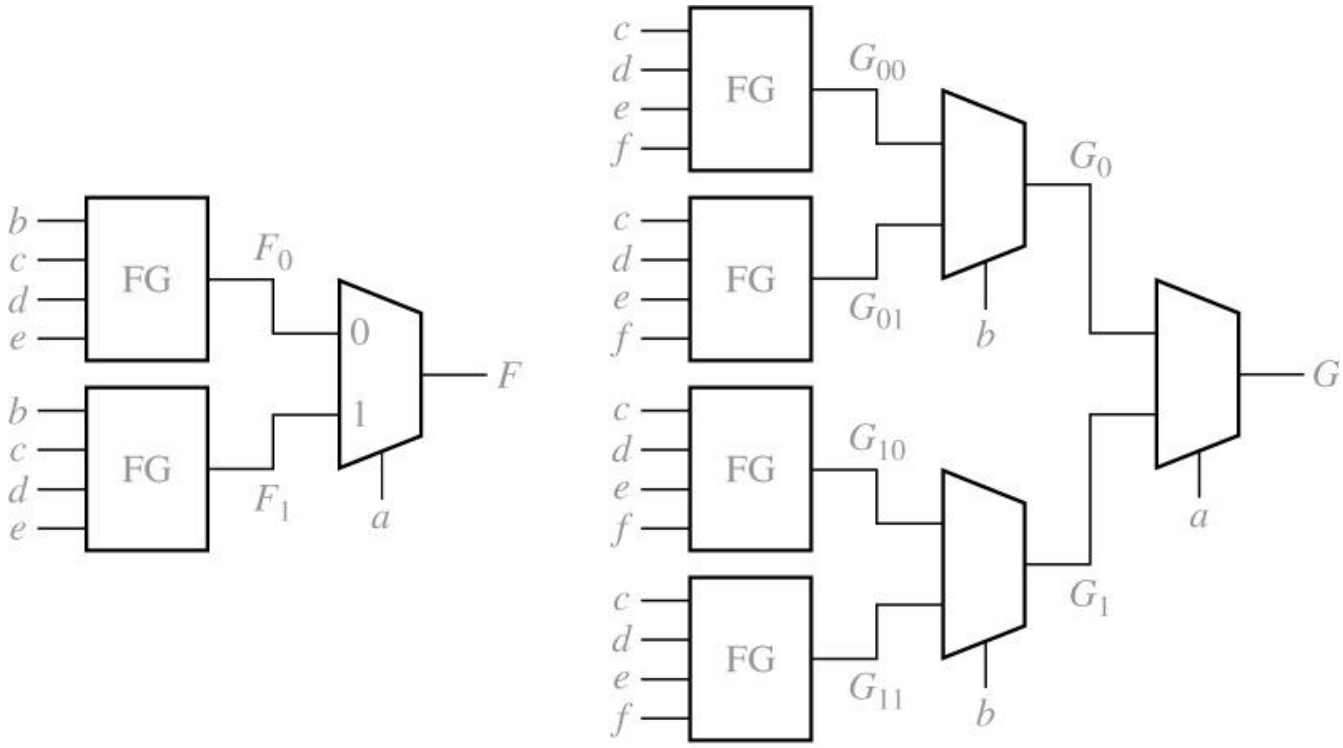
9.8 Field Programmable Gate Arrays

Fig 9-35. Function Expansion Using a Karnaugh Map



9.8 Field Programmable Gate Arrays

Fig 9-36. Realization of Five- and Six-Variable Functions with Function Generators



(a) 5-variable function

(b) 6-variable function