COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

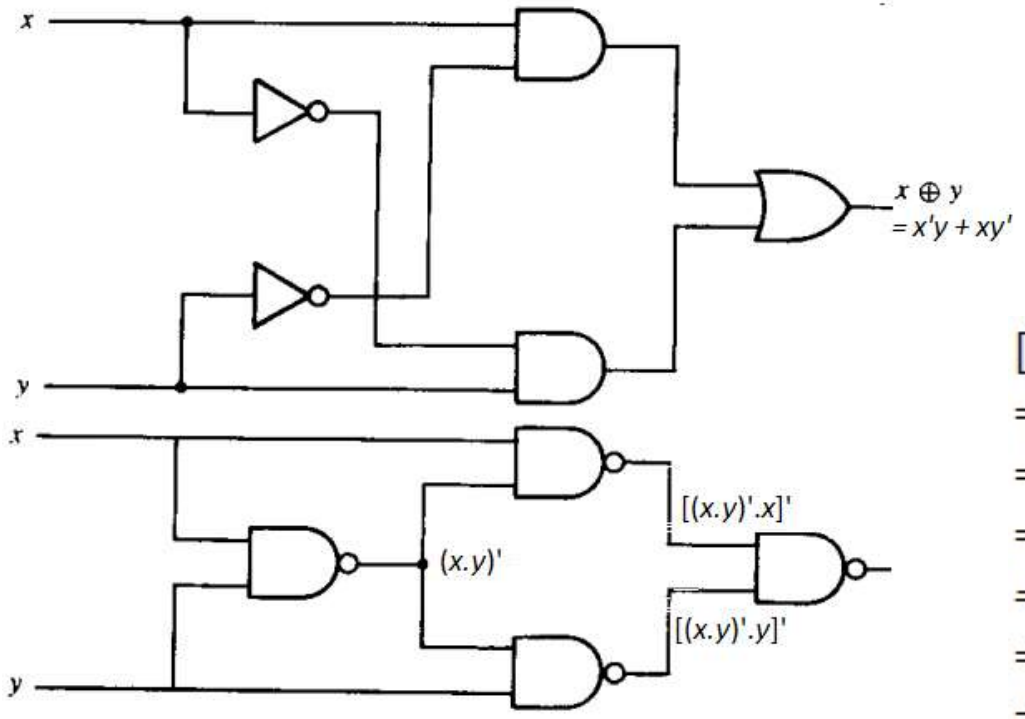
Exclusive OR and Equivalence Functions

Exclusive-OR and equivalence, denoted by $\oplus$ and $\odot$, respectively, are binary operations that perform the following Boolean functions:

$$x \oplus y = xy' + x'y$$
$$x \odot y = xy + x'y'$$

The two operations are the complements of each other. Each is commutative and associative. Because of these two properties, a function of three or more variables can be expressed without parentheses as follows:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$



$$[\{(x.y)'.x\}'.\{(x.y)'.y\}']'$$
$$= \{(x.y)'.x\} + \{(x.y)'.y\}$$
$$= \{(x' + y').x\} + \{(x' + y').y\}$$
$$= x.x' + x.y' + x'.y + y.y'$$
$$= 0 + x.y' + x'.y + 0$$
$$= x.y' + x'.y$$
$$= x \oplus y$$

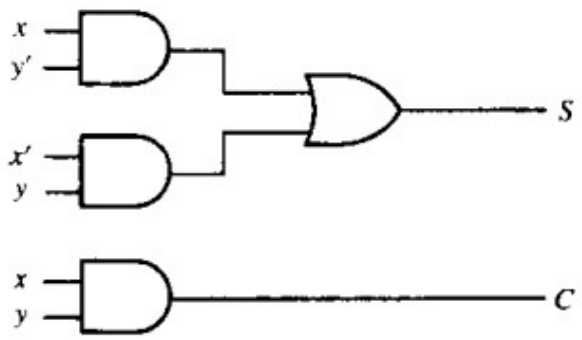
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Design of Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

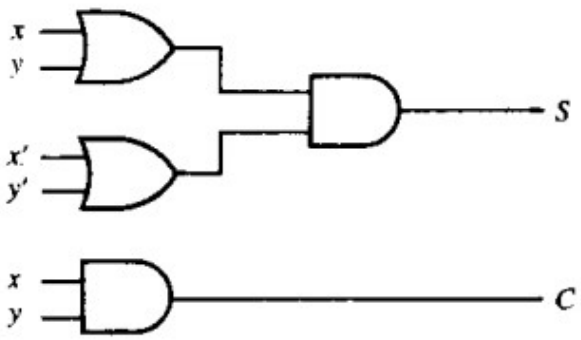
$S = x'y + xy'$

$C = xy$



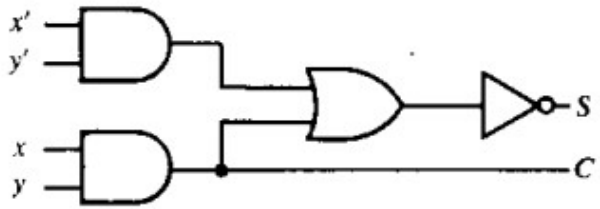
(a) $S = xy' + x'y$

$C = xy$



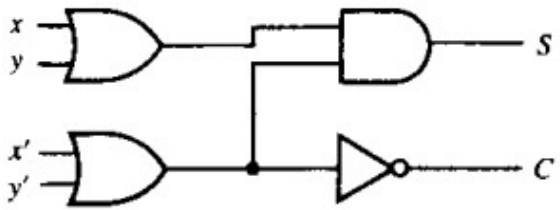
(b) $S = (x + y)(x' + y')$

$C = xy$



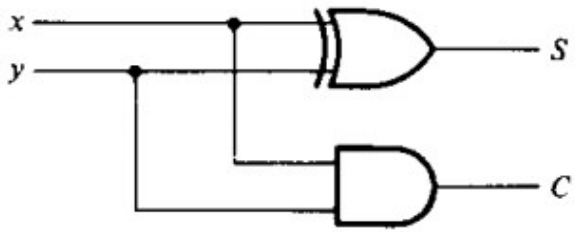
(c) $S = (C + x'y')'$

$C = xy$



(d) $S = (x + y)(x' + y')$

$C = (x' + y')'$



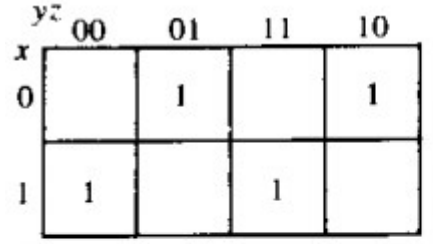
(e) $S = x \oplus y$

$C = xy$

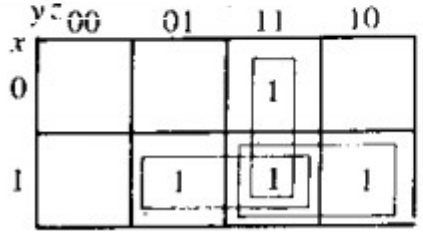
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Design of Full Adder

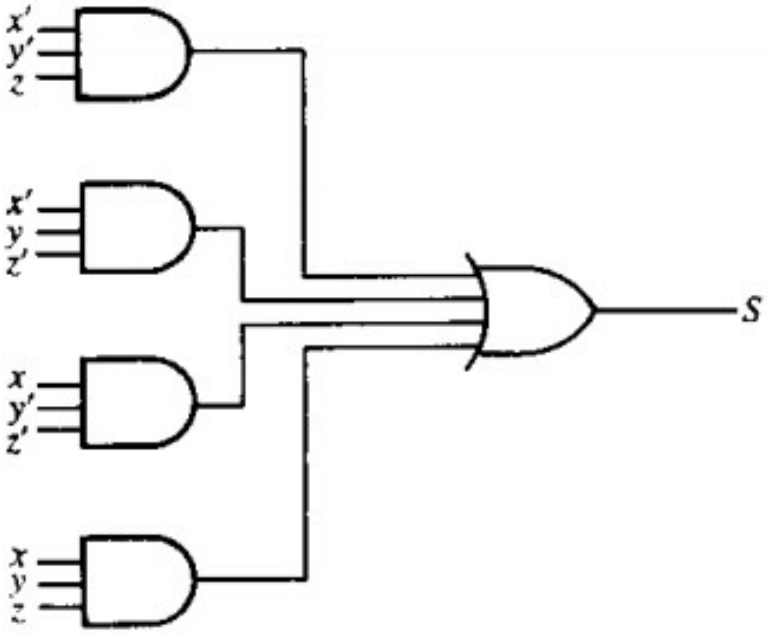
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

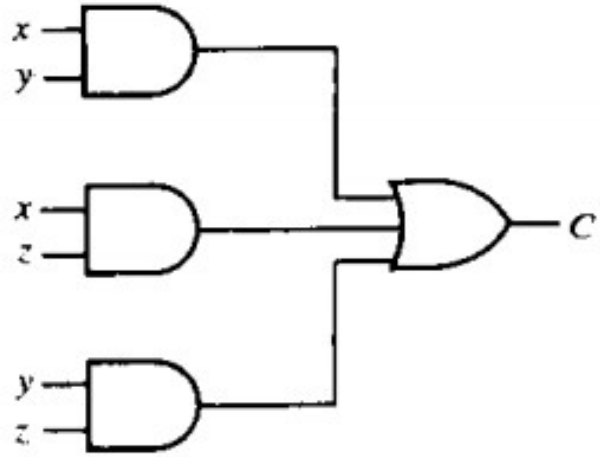


$S = x'y'z + x'yz' + xy'z' + xyz$



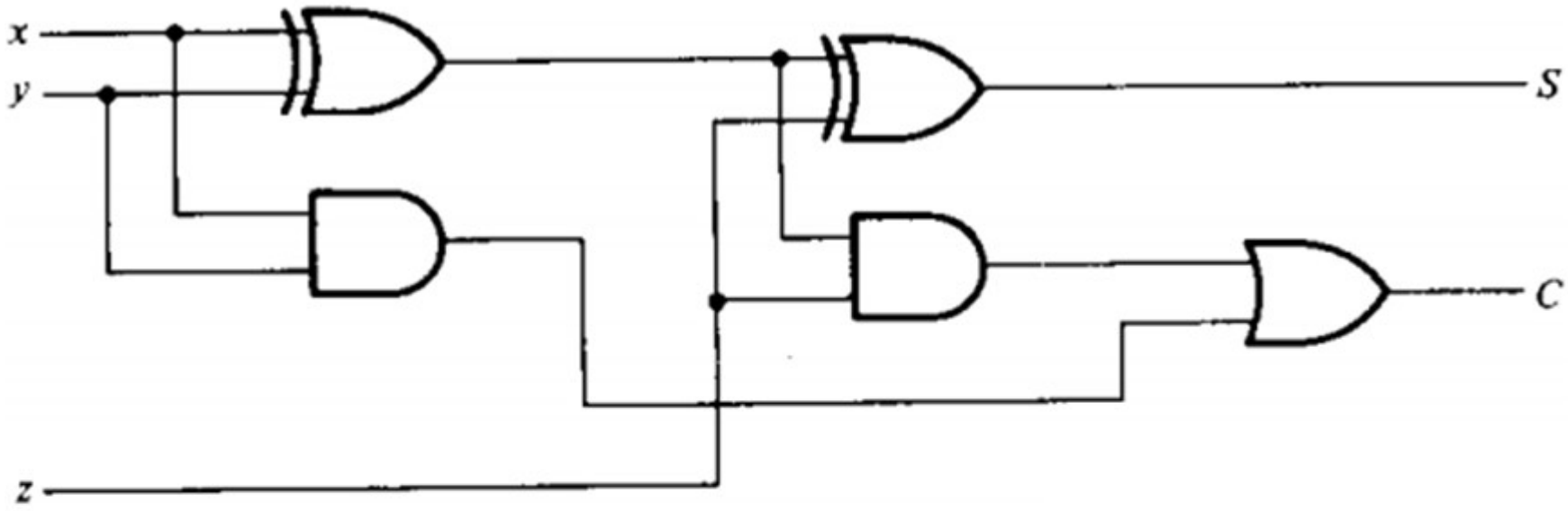
$C = xy + xz + yz$



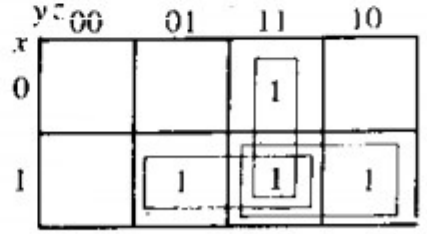


Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Full Adder with Two Half Adders and an OR Gate



$$S = z \oplus (x \oplus y)$$
$$= z'(xy' + x'y) + z(xy' + x'y)'$$
$$= z'(xy' + x'y) + z(xy + x'y')$$
$$= xy'z' + x'yz' + xyz + x'y'z$$
$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$



$C = xy + xz + yz$

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Design of Subtractor

Half-Subtractor

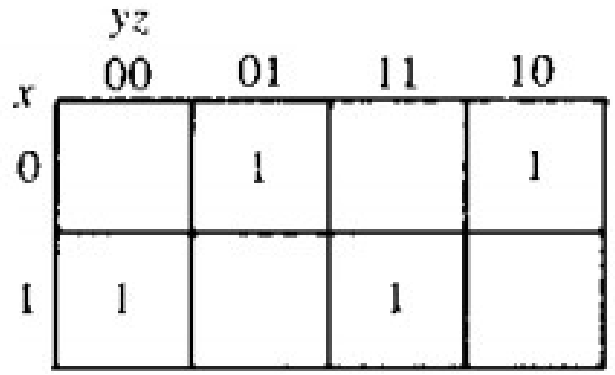
x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$$D = x'y + xy'$$

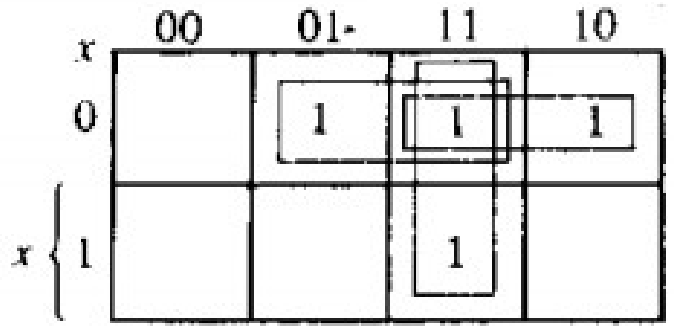
$$B = x'y$$

Full-Subtractor

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



$$D = x'y'z + x'yz' + xy'z' + xyz$$



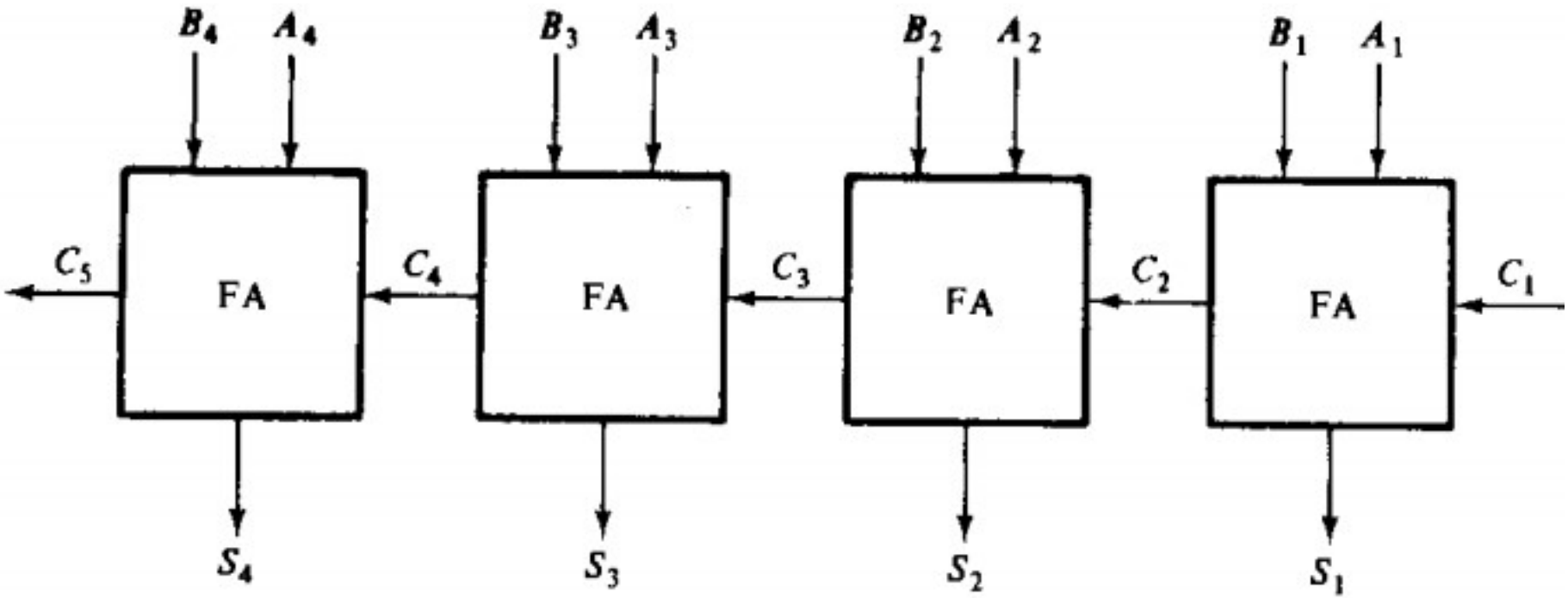
$$B = x'y + x'z + yz$$

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

Parallel Adder

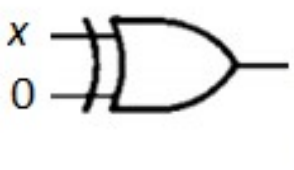


	C_5	C_4	C_3	C_2	C_1	←	C
		A_4	A_3	A_2	A_1	←	A
+		B_4	B_3	B_2	B_1	←	B
		S_4	S_3	S_2	S_1	←	S

	1	0	1	1		←	C
		1	0	1	1	←	A
+		1	0	0	1	←	B
		0	1	0	0	←	S

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

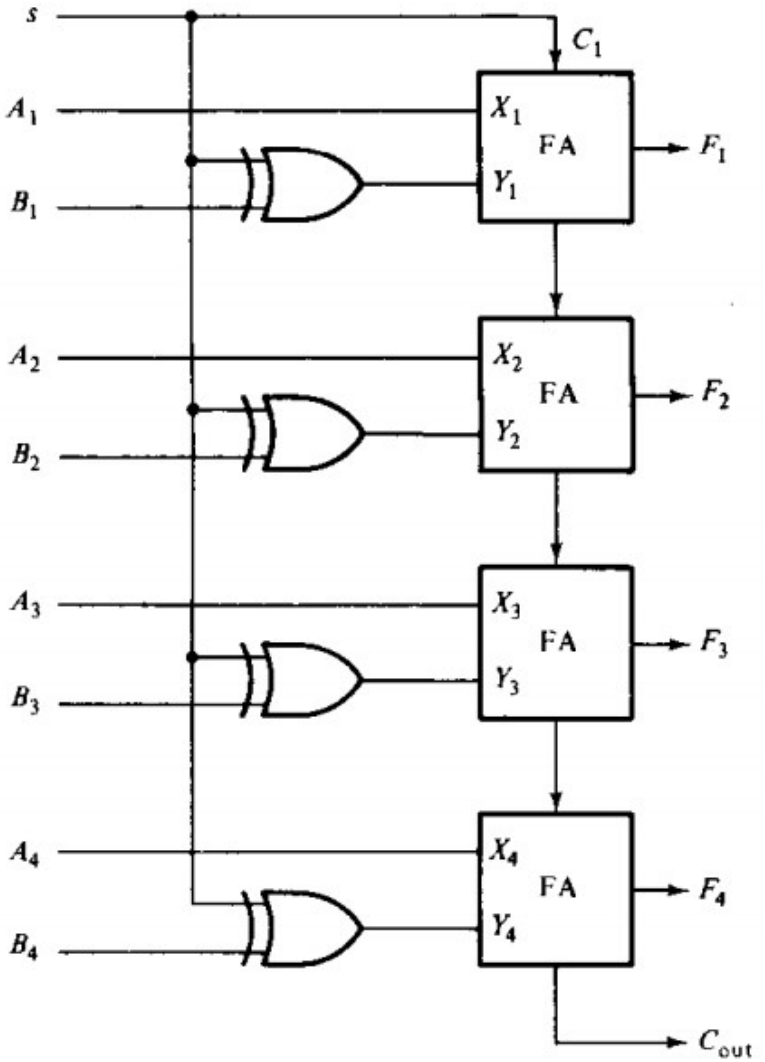
4-bit Adder/Subtractor



$$xy' + x'y$$
$$= x.1 + x'.0$$
$$= x$$

$$xy' + x'y$$
$$= x.0 + x'.1$$
$$= x'$$

$$A + B' + 1 = A - B$$



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD to Excess-3 Code Converter

Input BCD				Output 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

$$z = D'$$

$$y = CD + C'D' = CD + (C + D)'$$

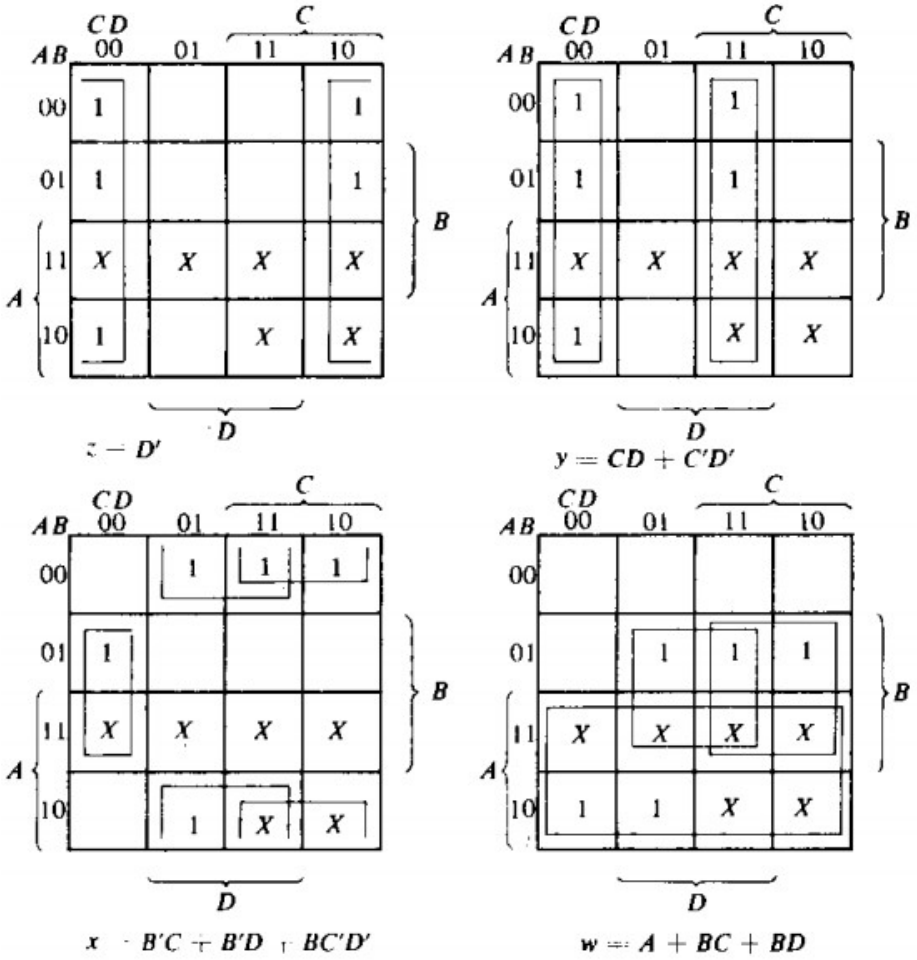
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$

$$= B'(C + D) + B(C + D)'$$

$$w = A + BC + BD = A + B(C + D)$$

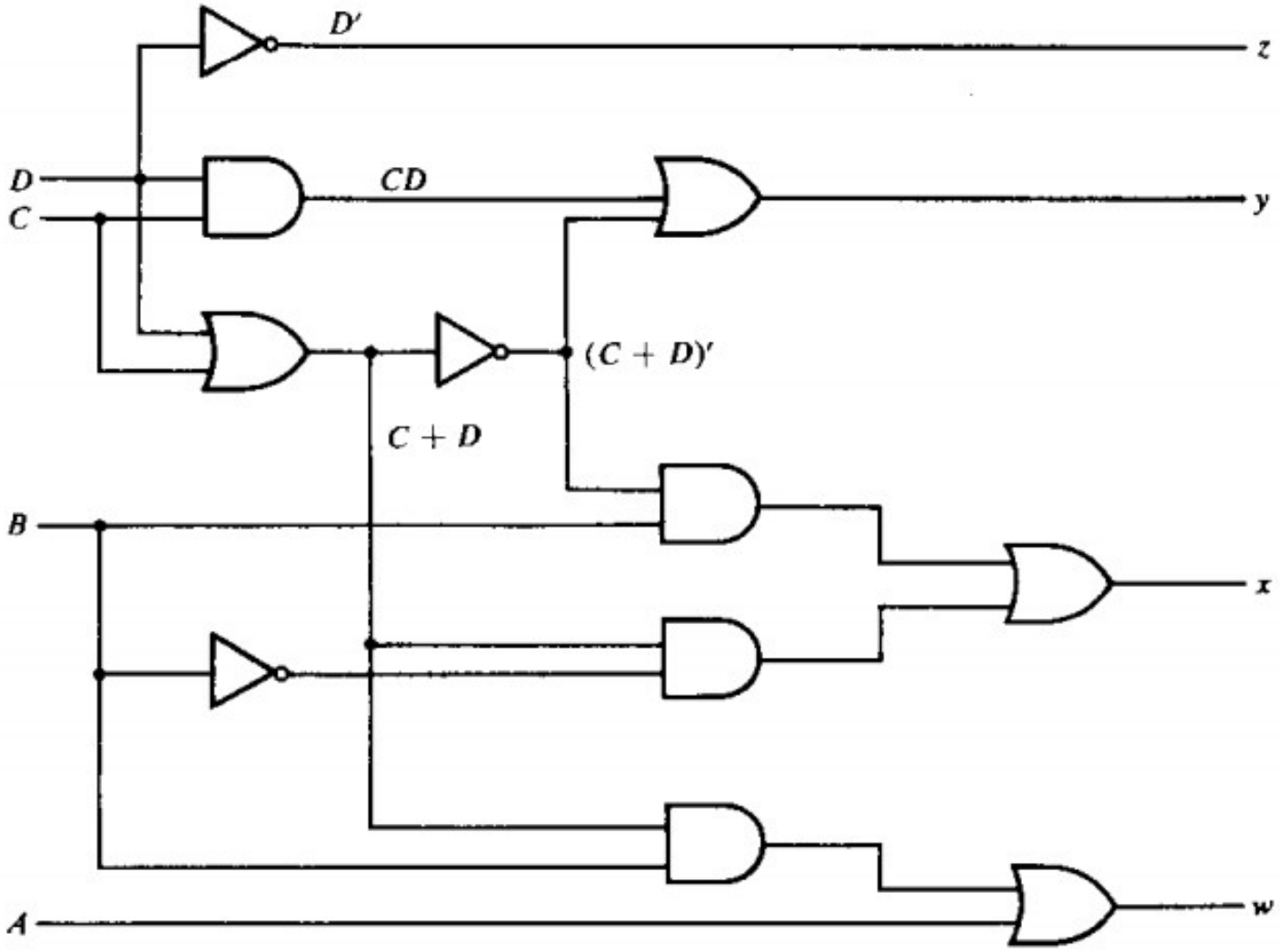
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD to Excess-3 Code Converter (contd.)



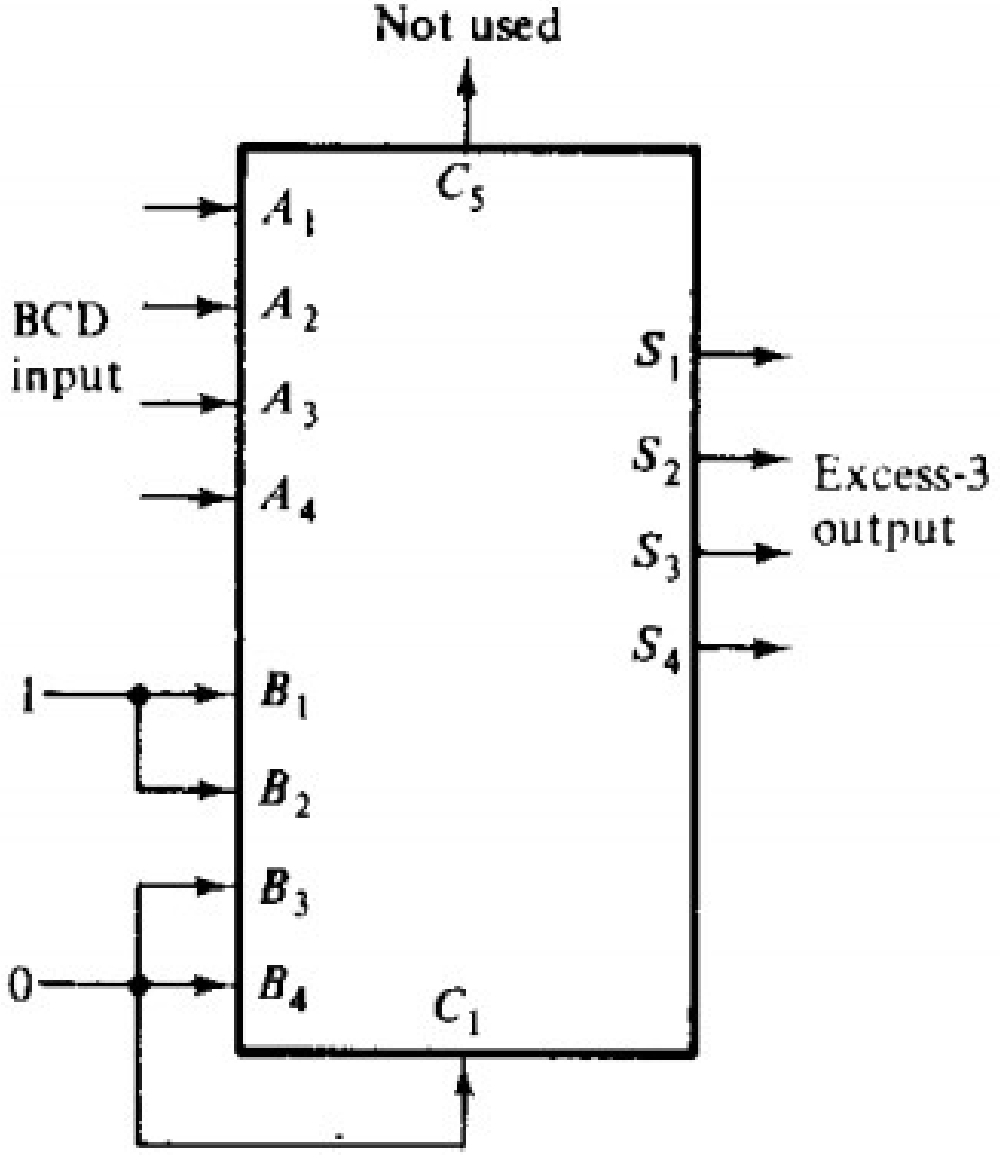
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD to Excess-3 Code Converter (contd.)



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD to Excess-3 Code Converter (contd.)



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD Adder

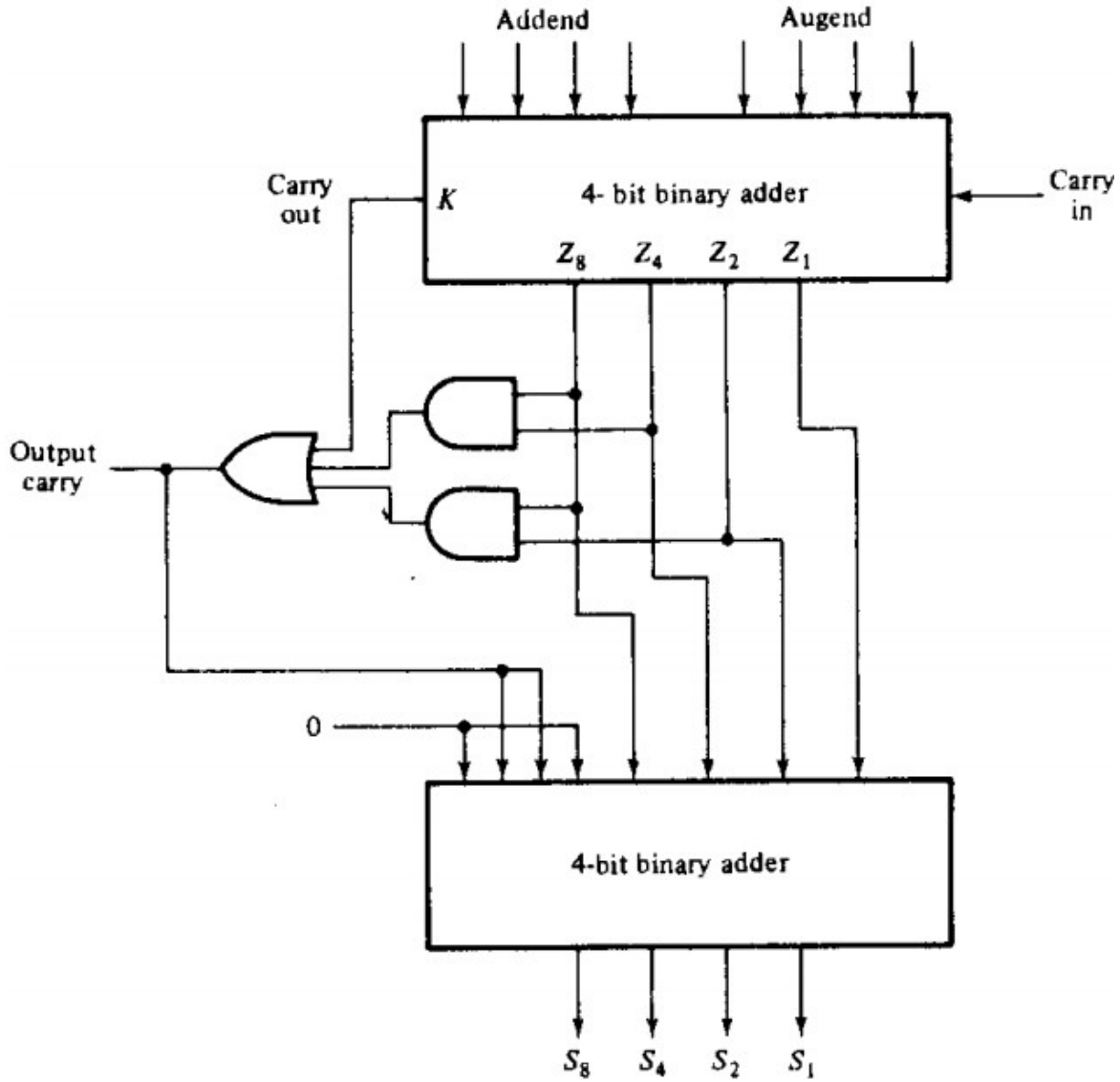
$$C = K + Z_8Z_4 + Z_8Z_2$$

Binary sum					BCD sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD Adder (contd.)

$$C = K + Z_8Z_4 + Z_8Z_2$$



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Magnitude Comparator

$$A = A_3A_2A_1A_0$$

$$B = B_3B_2B_1B_0$$

$$x_i = A_iB_i + A'_iB'_i \qquad i = 0, 1, 2, 3$$

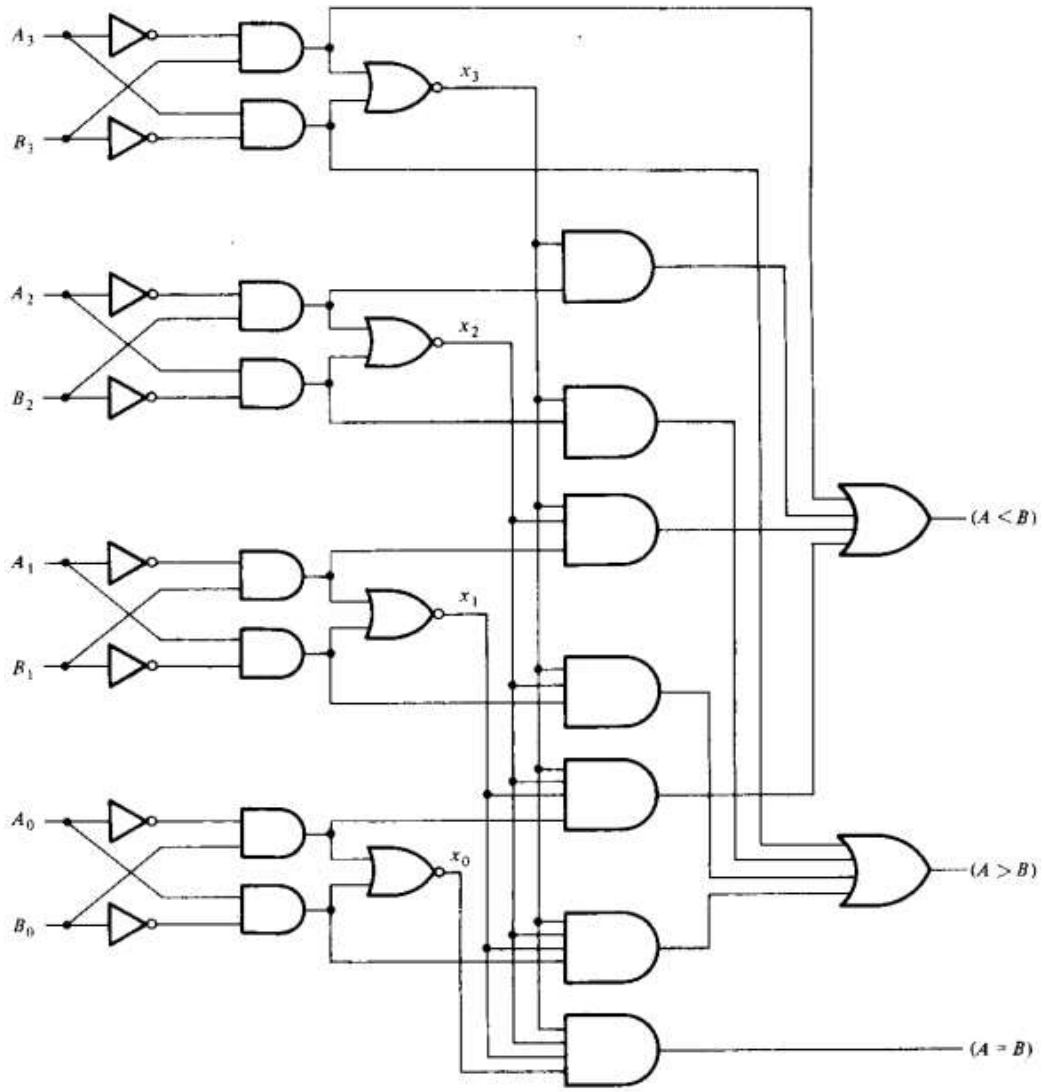
$$(A = B) = x_3x_2x_1x_0$$

$$(A > B) = A_3B'_3 + x_3A_2B'_2 + x_3x_2A_1B'_1 + x_3x_2x_1A_0B'_0$$

$$(A < B) = A'_3B_3 + x_3A'_2B_2 + x_3x_2A'_1B_1 + x_3x_2x_1A'_0B_0$$

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Magnitude Comparator (contd.)



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

Decoders

Discrete quantities of information are represented in digital systems with binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of the coded information. A *decoder* is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.

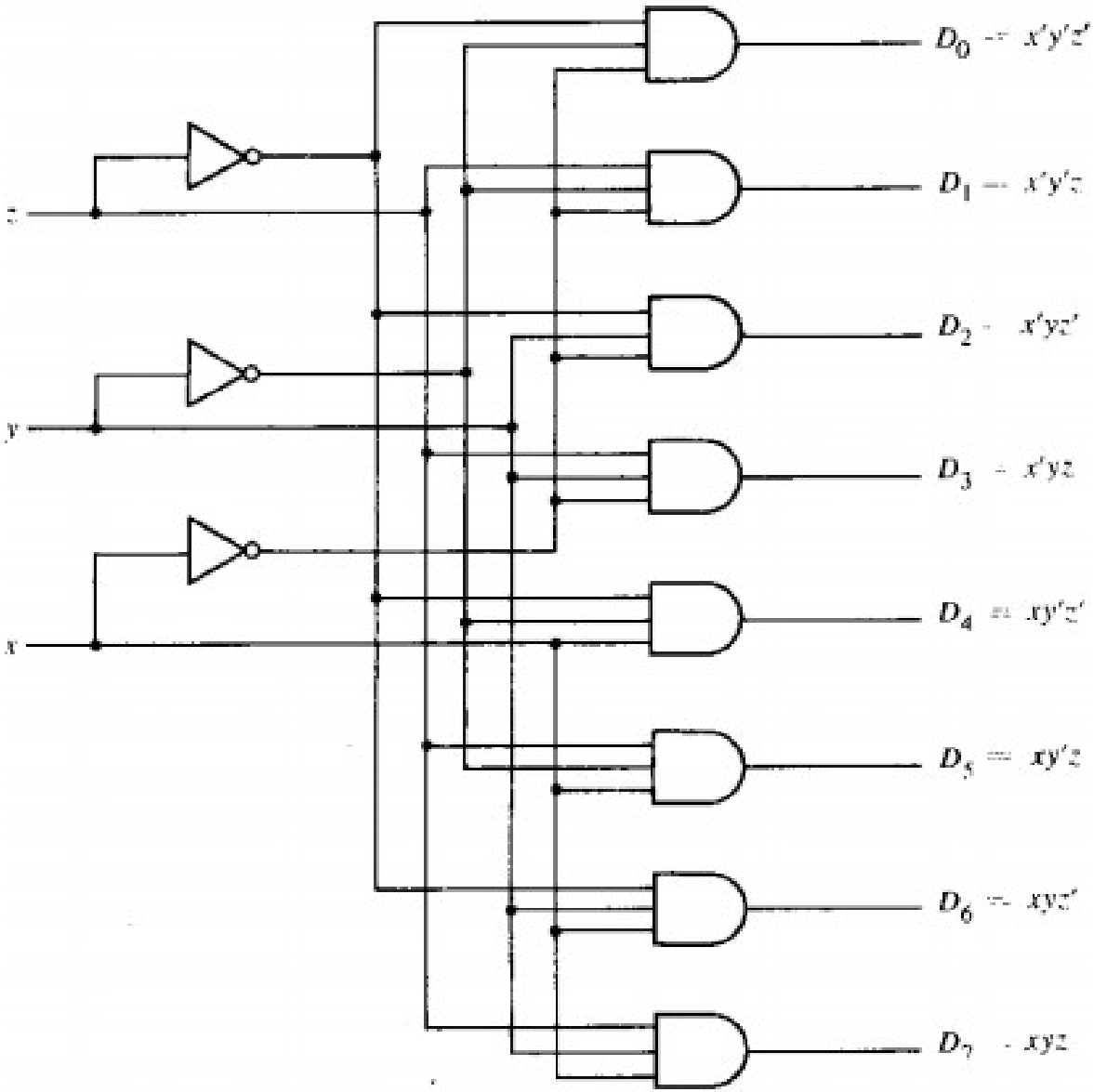
The decoders presented here are called n-to-m line decoders where $m \leq 2^n$. Their purpose is to generate the 2^n (or less) minterms of n input variables.

Truth table of a 3-to-8 line decoder

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_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

Reference: M. Morris Mano, "Digital Logic and Computer Design", PHI

3-to-8 Line Decoder



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD-to-Decimal Decoder

wx \ yz	00	01	11	10
00	D_0	D_1	D_3	D_2
01	D_4	D_5	D_7	D_6
11	X	X	X	X
10	D_8	D_9	X	X

$D_0 = w'.x'.y'.z'$

$D_1 = w'.x'.y'.z$

$D_2 = x'.y.z'$

$D_3 = x'.y.z$

$D_4 = x.y'.z'$

$D_5 = x.y'.z$

$D_6 = x.y.z'$

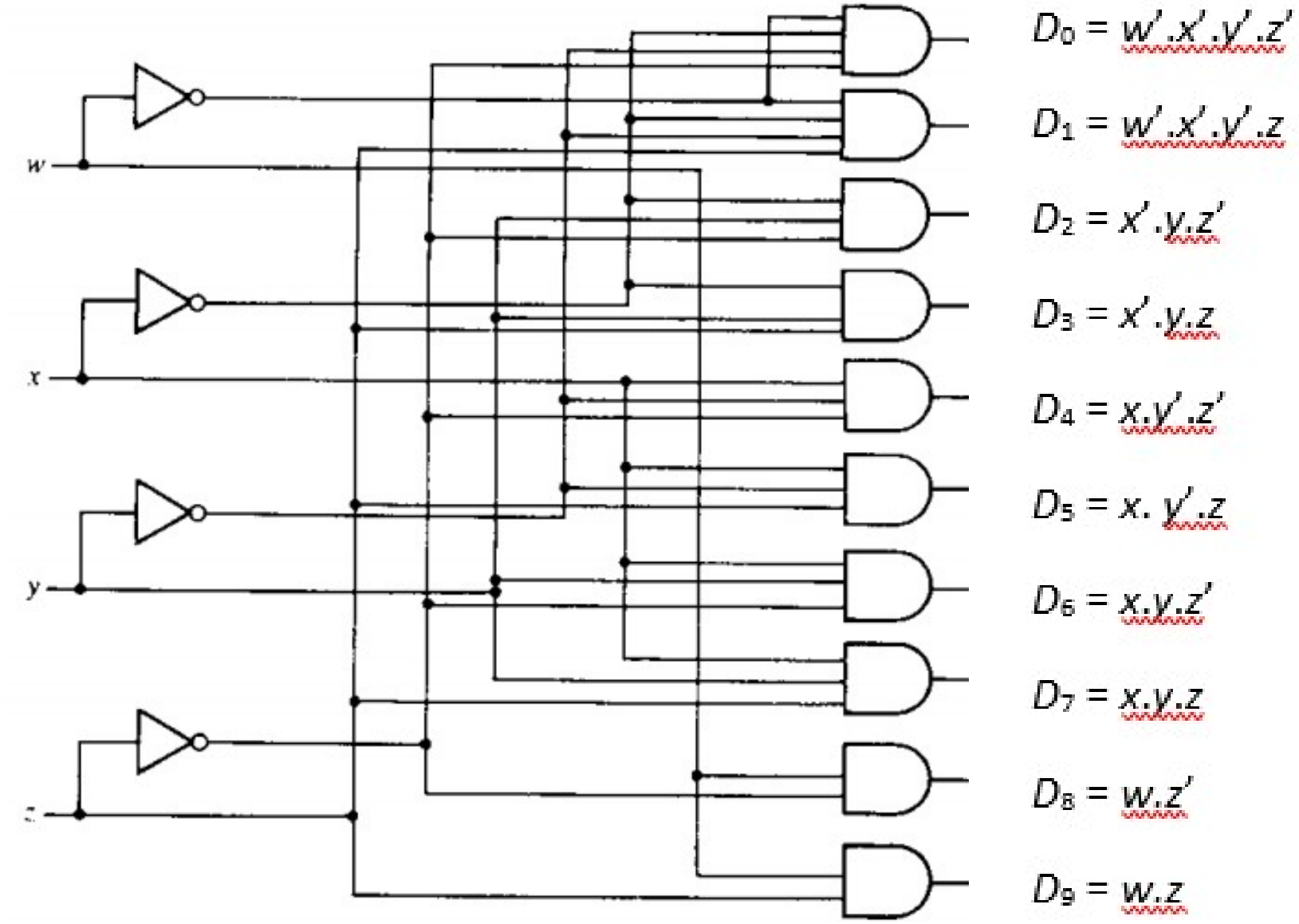
$D_7 = x.y.z$

$D_8 = w.z'$

$D_9 = w.z$

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

BCD-to-Decimal Decoder Logic Diagram

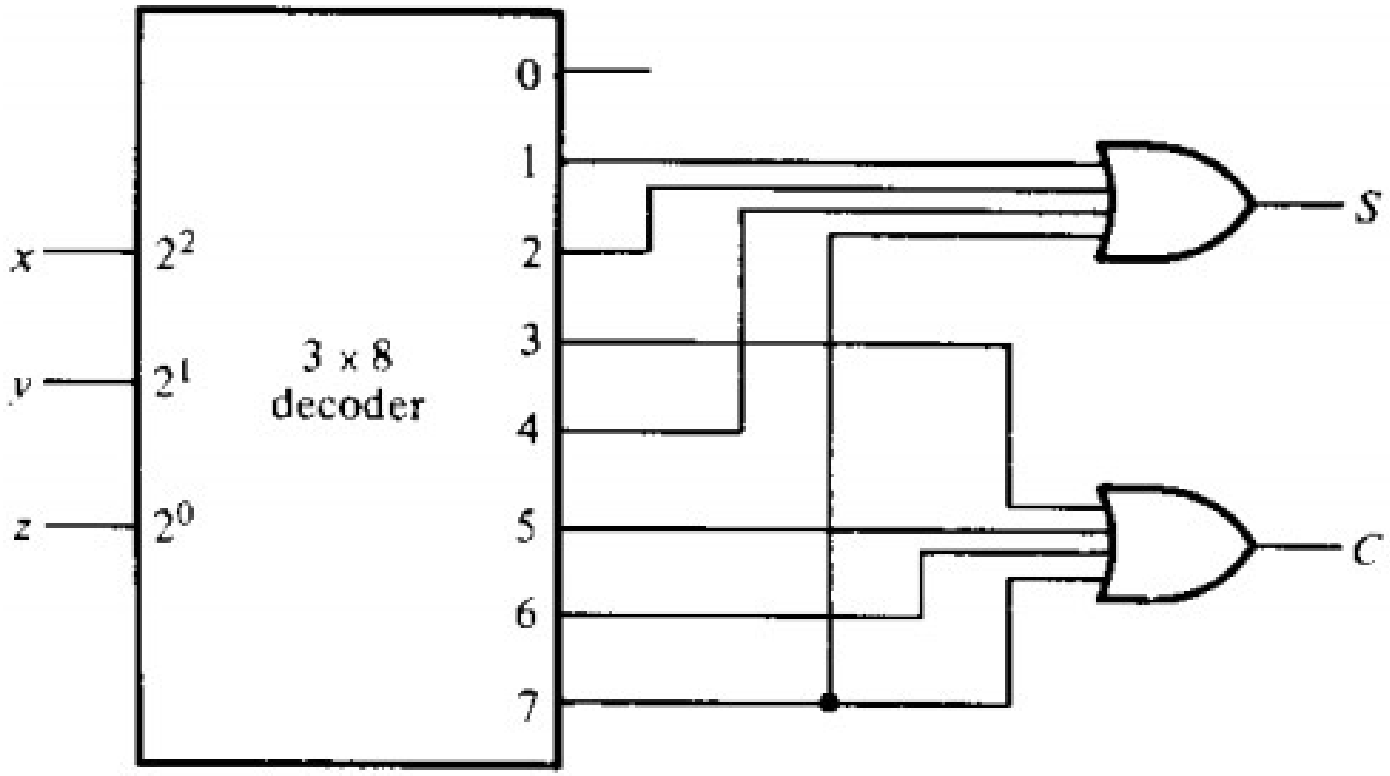


Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Realization of Boolean Functions using Decoders

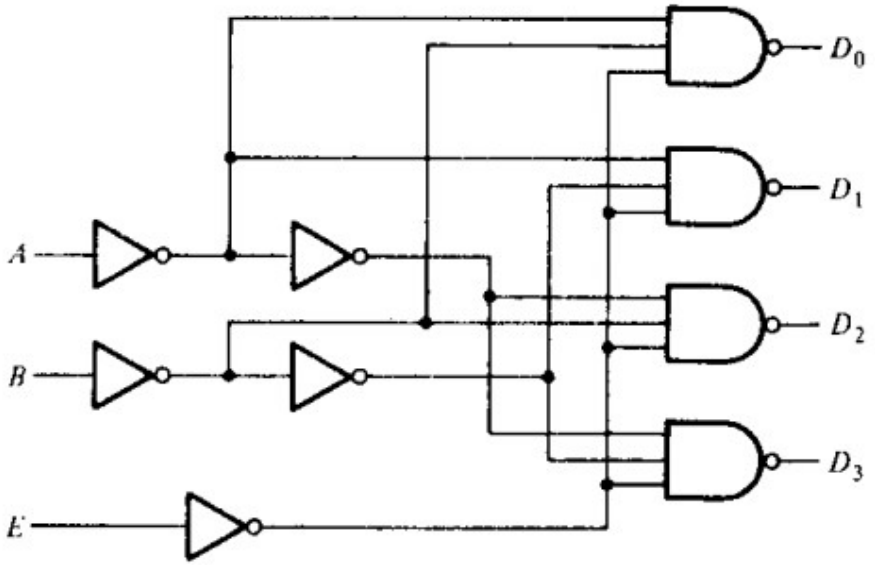
From the truth table of the full-adder, we obtain the functions for this combinational circuit in sum of minterms:

$$S(x, y, z) = \Sigma(1, 2, 4, 7)$$
$$C(x, y, z,) = \Sigma(3, 5, 6, 7)$$

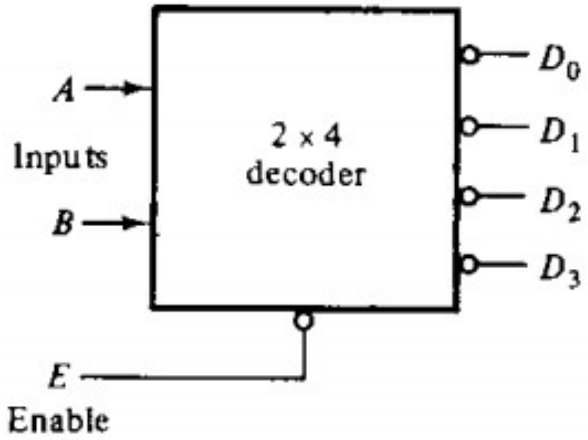


Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

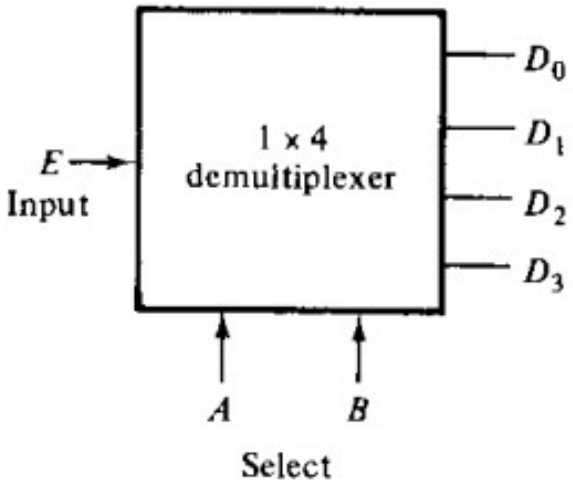
DeMultiplexer



E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



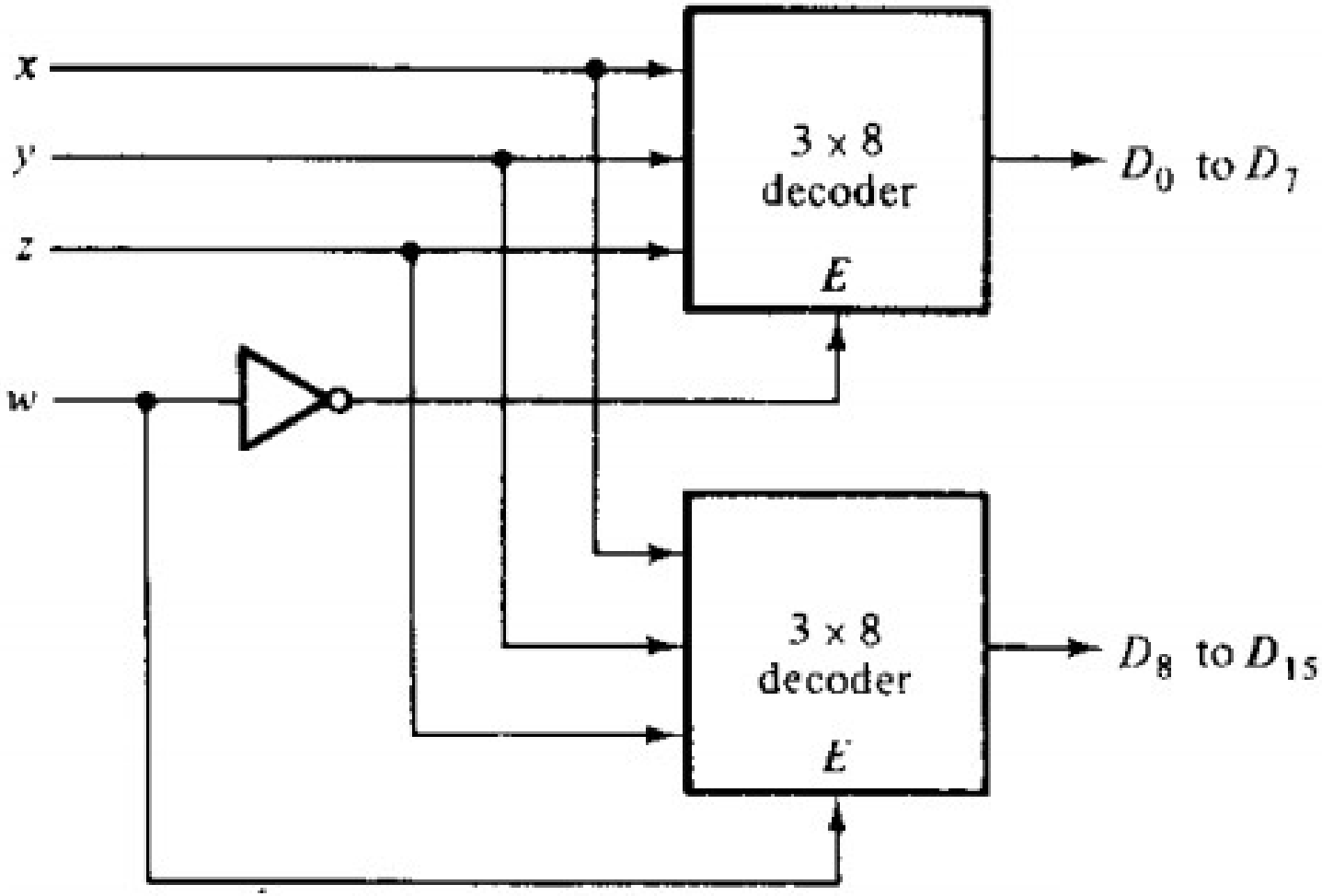
(a) Decoder with enable



(b) Demultiplexer

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Higher Order Decoders



4 × 16 decoder constructed with two 3 × 8 decoders

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

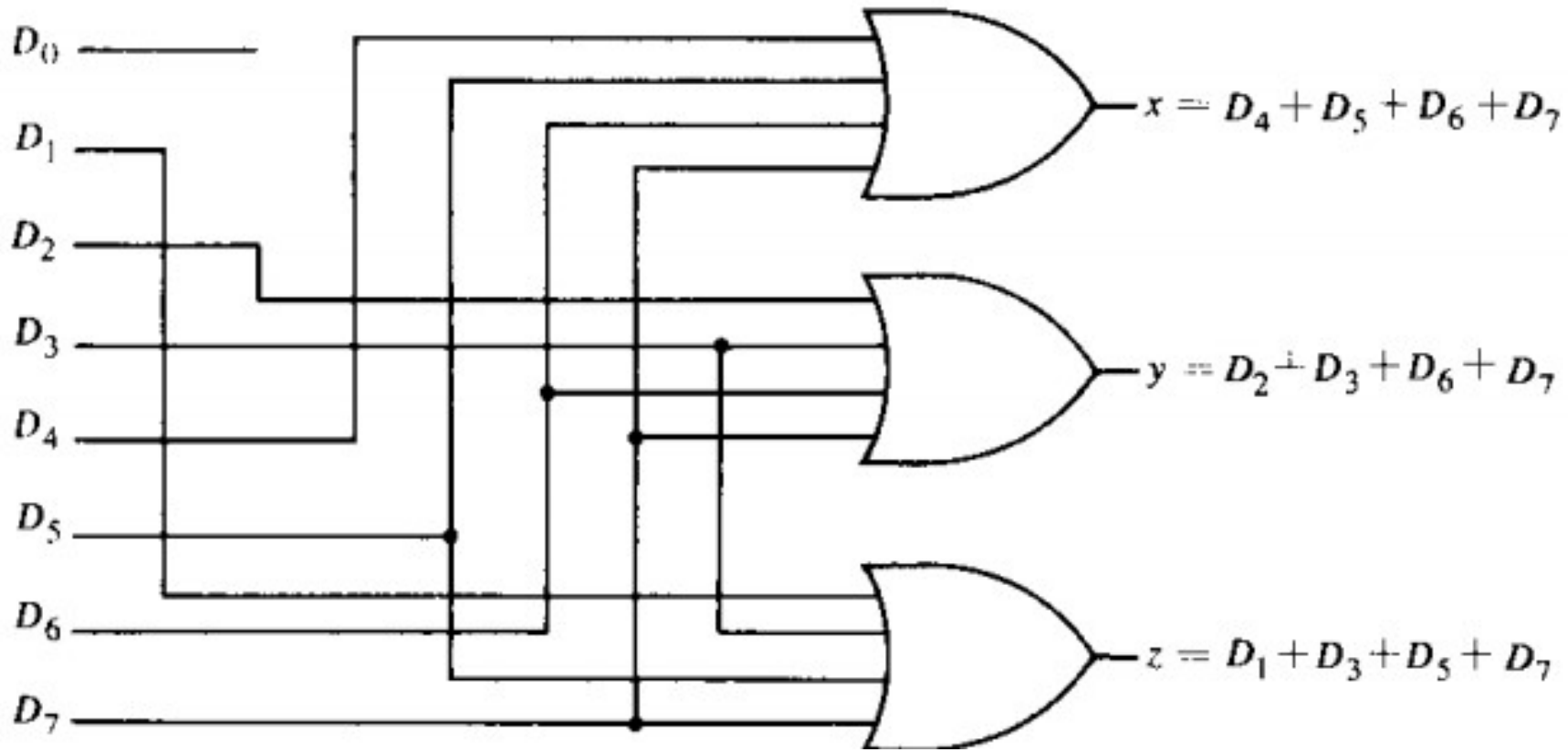
Encoders

Truth table of octal-to-binary encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Octal to Binary Encoder



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Priority Encoder

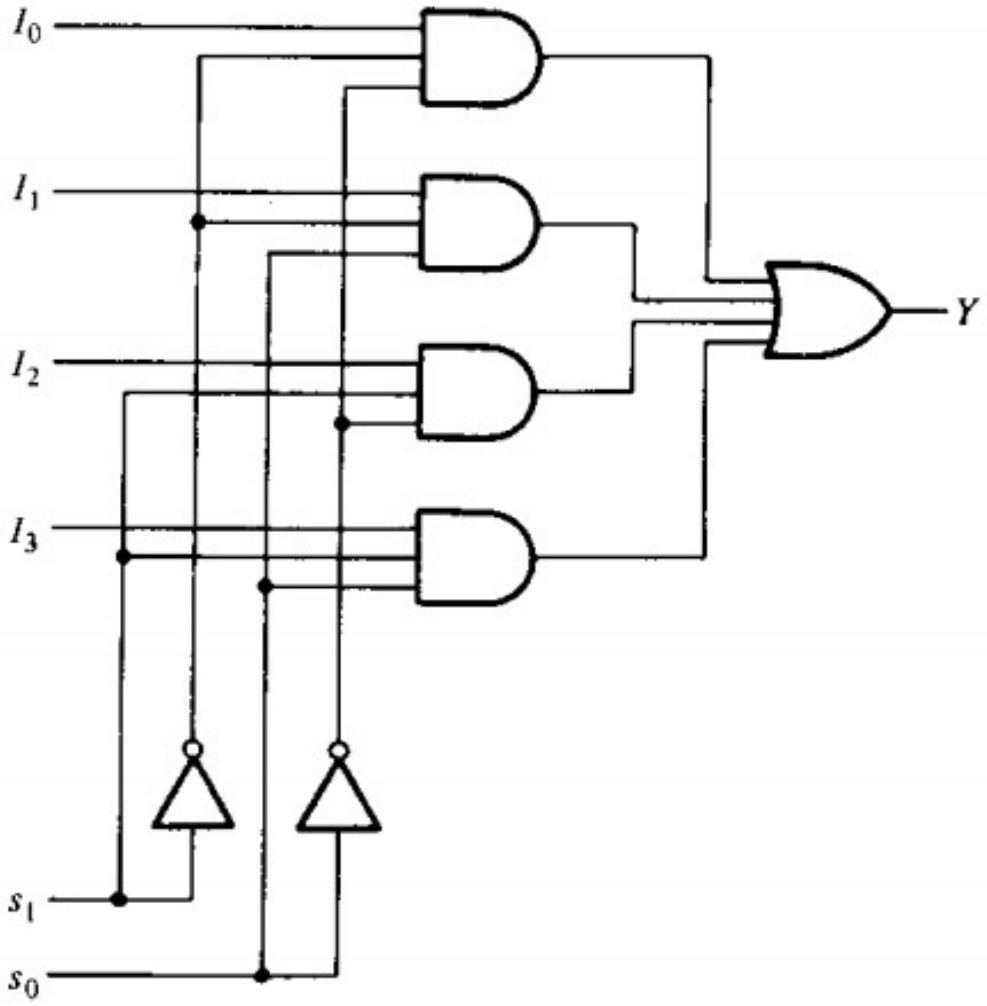
Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
x	1	0	0	0	0	0	0	0	0	1
x	x	1	0	0	0	0	0	0	1	0
x	x	x	1	0	0	0	0	0	1	1
x	x	x	x	1	0	0	0	1	0	0
x	x	x	x	x	1	0	0	1	0	1
x	x	x	x	x	x	1	0	1	1	0
x	x	x	x	x	x	x	1	1	1	1

$$x = D_4\, D_5'\, D_6'\, D_7' + D_5\, D_6'\, D_7' + D_6\, D_7' + D_7$$

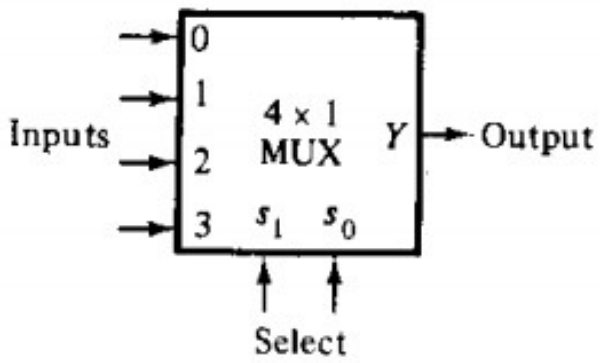
$$y = D_2\, D_3'\, D_4'\, D_5'\, D_6'\, D_7' + D_3\, D_4'\, D_5'\, D_6'\, D_7' + D_6\, D_7' + D_7$$

$$z = D_1\, D_2'\, D_3'\, D_4'\, D_5'\, D_6'\, D_7' + D_3\, D_4'\, D_5'\, D_6'\, D_7' + D_5\, D_6'\, D_7' + D_7$$

Multiplexers



(a) Logic diagram



(c) Block diagram

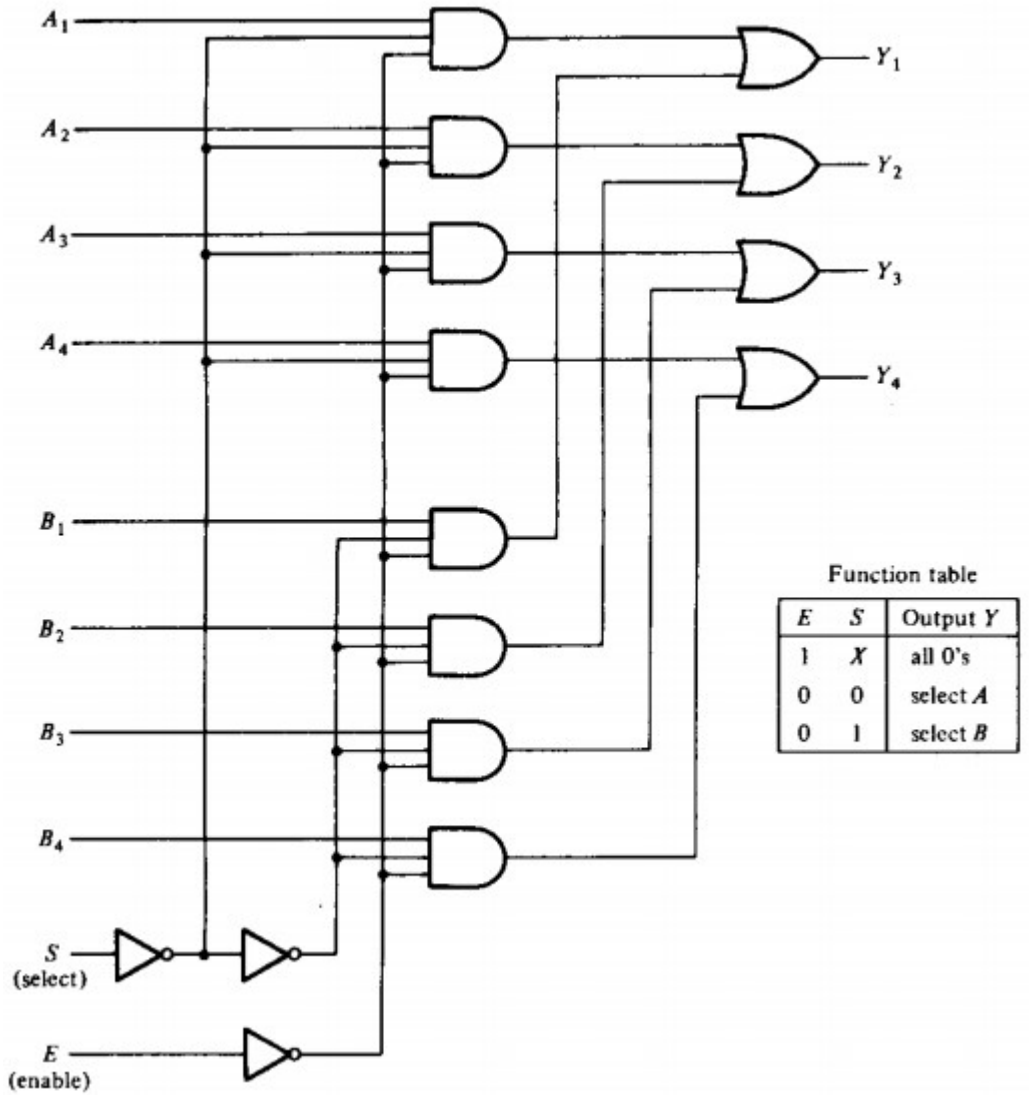
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

A 4-to-1 line multiplexer

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Quadruple 2-to-1 Line Multiplexers



Function table

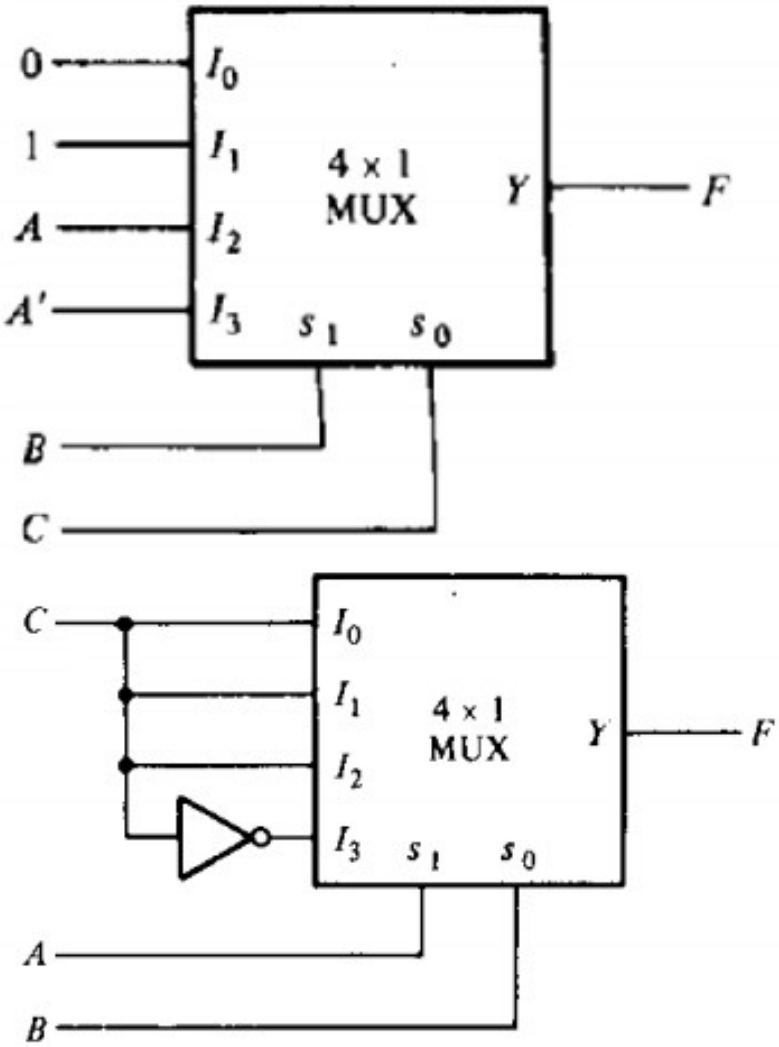
E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Function Implementation Using Multiplexers

$$F(A, B, C) = \Sigma(1, 3, 5, 6)$$

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

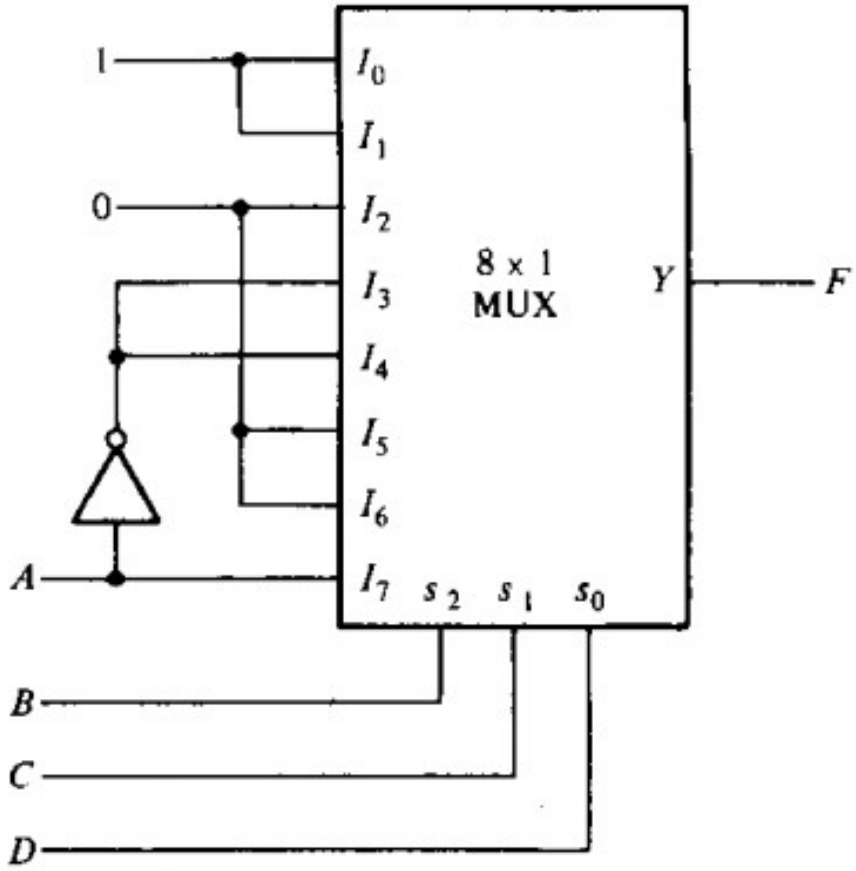


Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Function Implementation Using Multiplexers

$$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$$

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



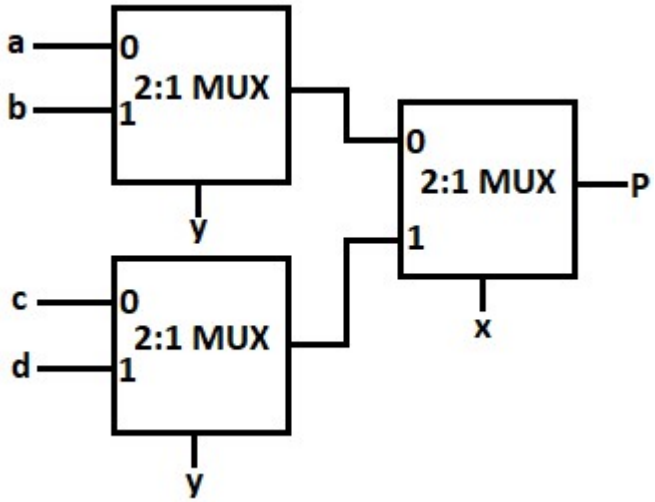
- Only Canonical SOP form functions are implemented
- One function per multiplexer
- No external gate required

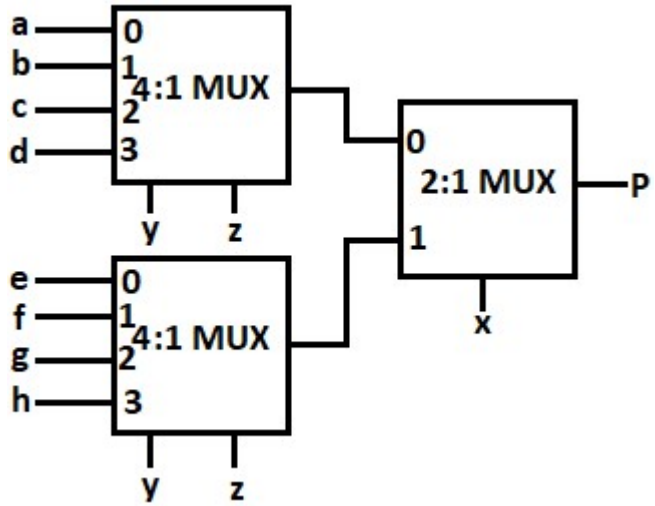
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

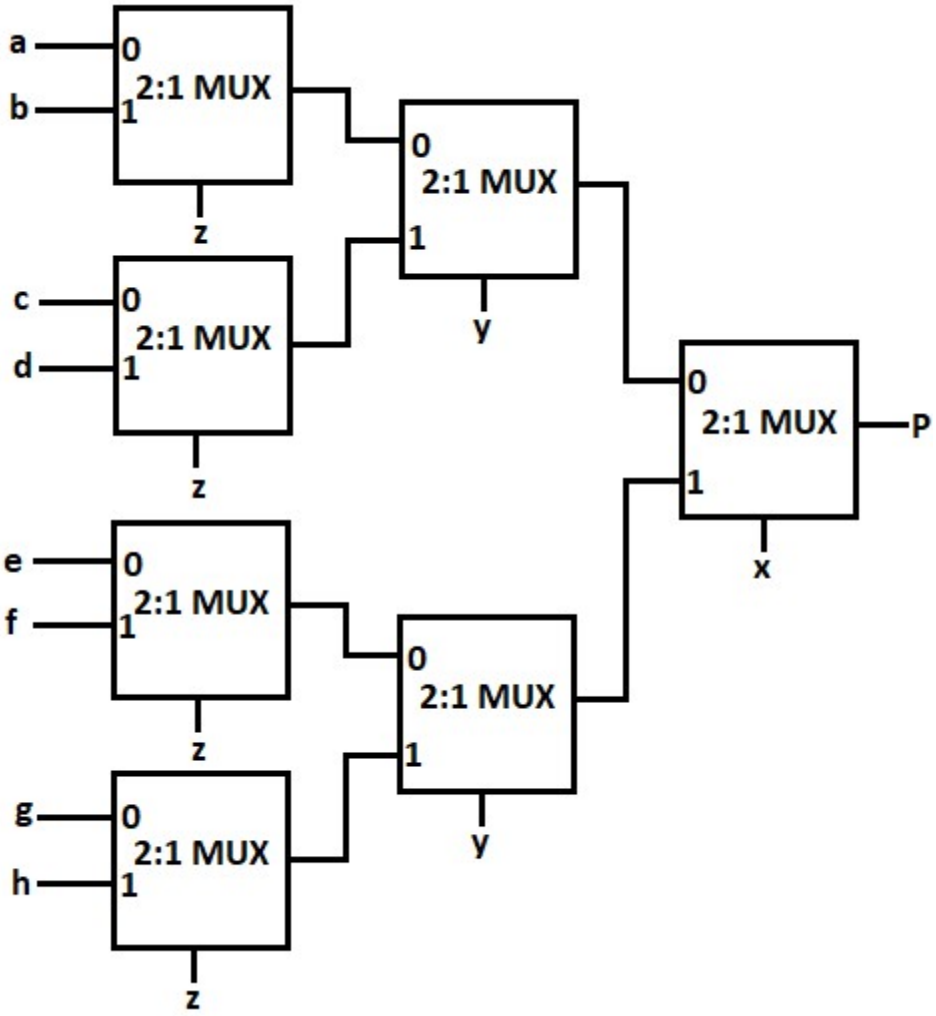
COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

Higher Order MUX using Lower Order MUXes

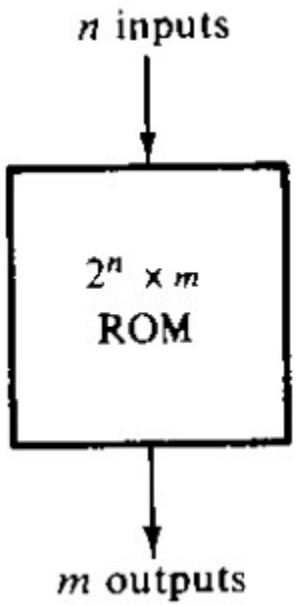






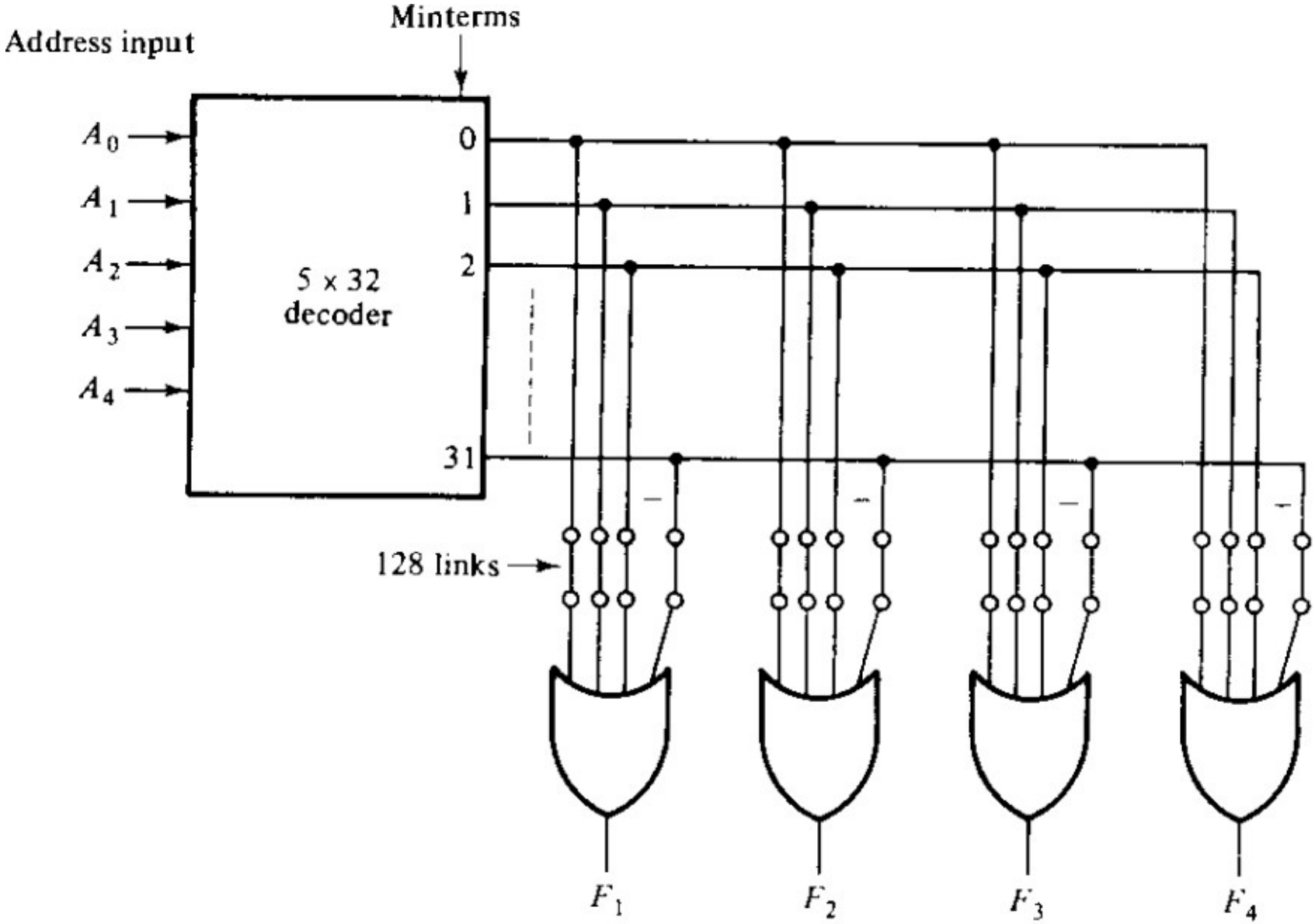
Read Only Memory (ROM)

- Decoder followed by Encoder
- Non-volatile



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

32 × 4 ROM



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

COC2070 – Digital Logic and System Design

Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

ROM Applications

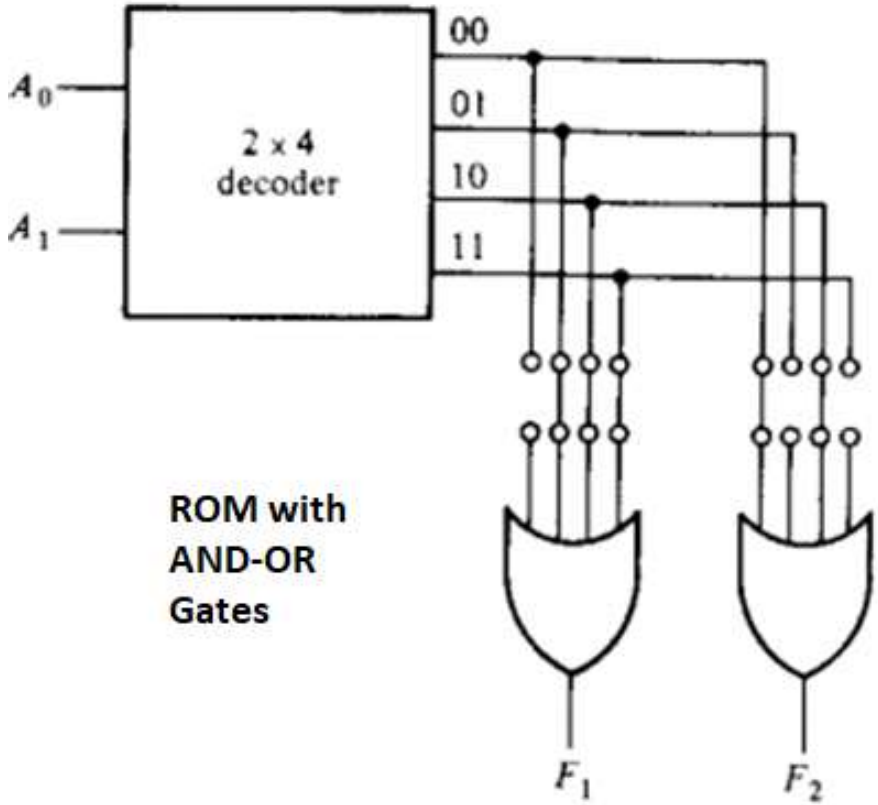
- Memory which is non-volatile
- Function Implementation
  - Many functions without any external gate
- Code Converter
  - Decoder followed by encoder
- Look up table
  - e.g. Multiplication tables, log tables, trigonometric tables, etc.
- Waveform Generation
  - Irregular but periodic waveforms
- Boot ROM as it is non-volatile

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

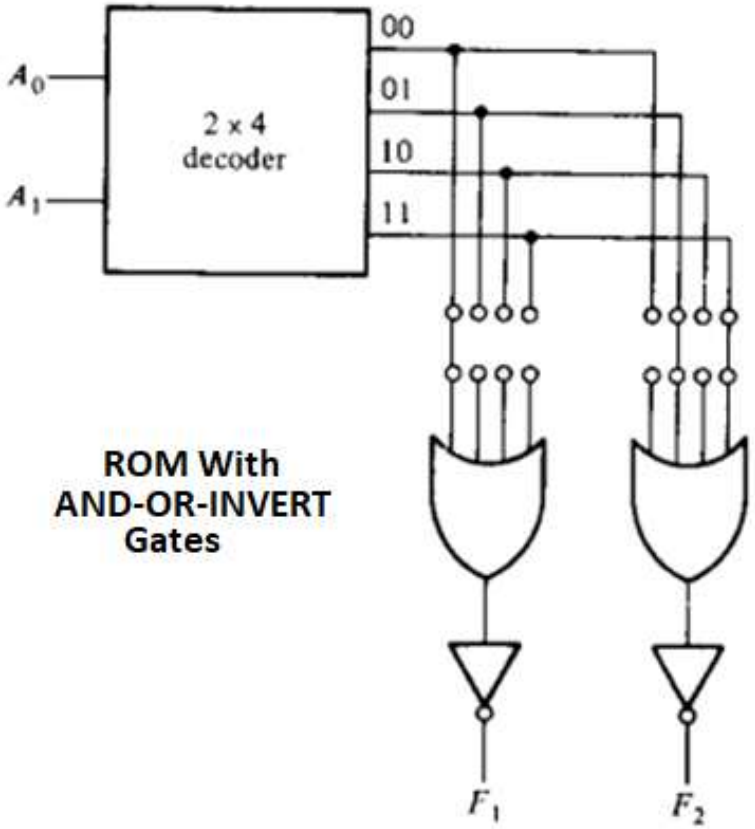
Function Implementation using ROM

$$F_1(A_1, A_0) = \Sigma(1, 2, 3)$$
$$F_2(A_1, A_0) = \Sigma(0, 2)$$

A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0



ROM with AND-OR Gates

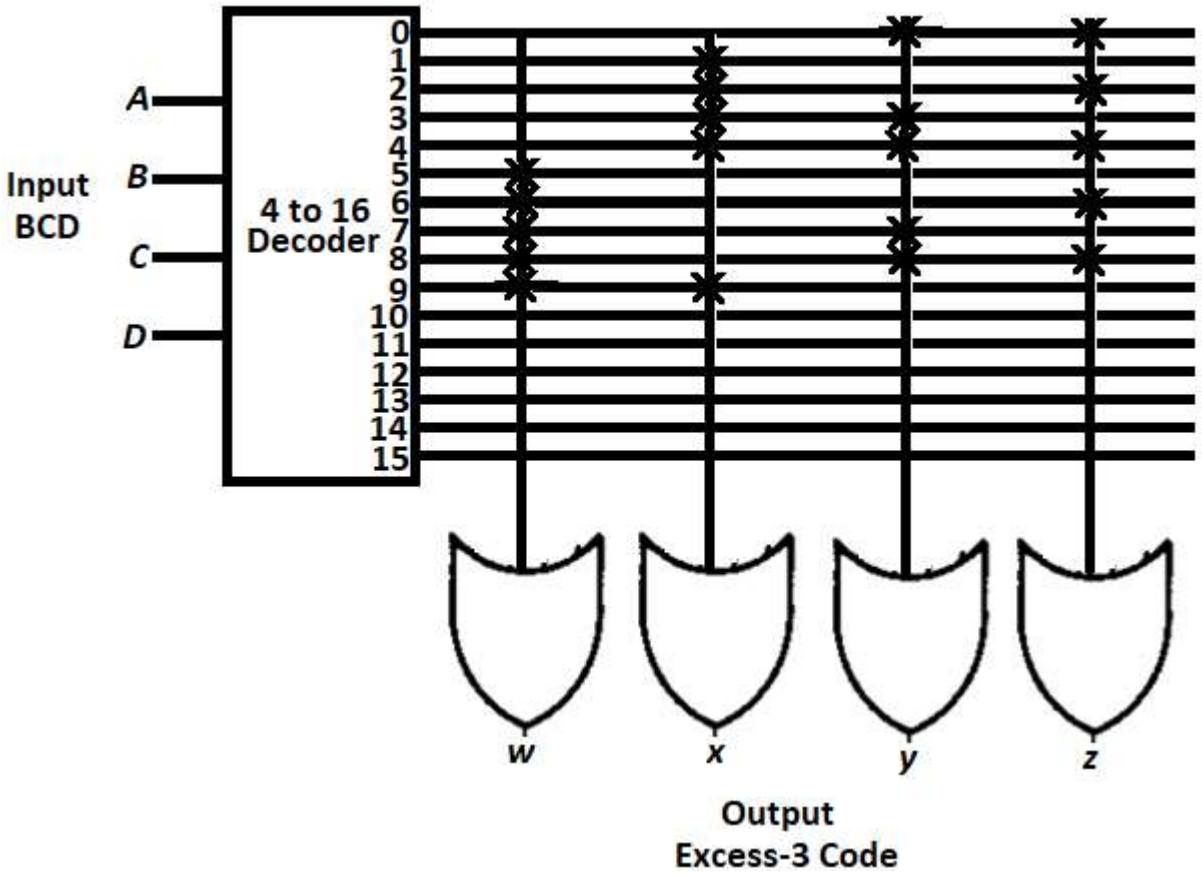


ROM With AND-OR-INVERT Gates

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

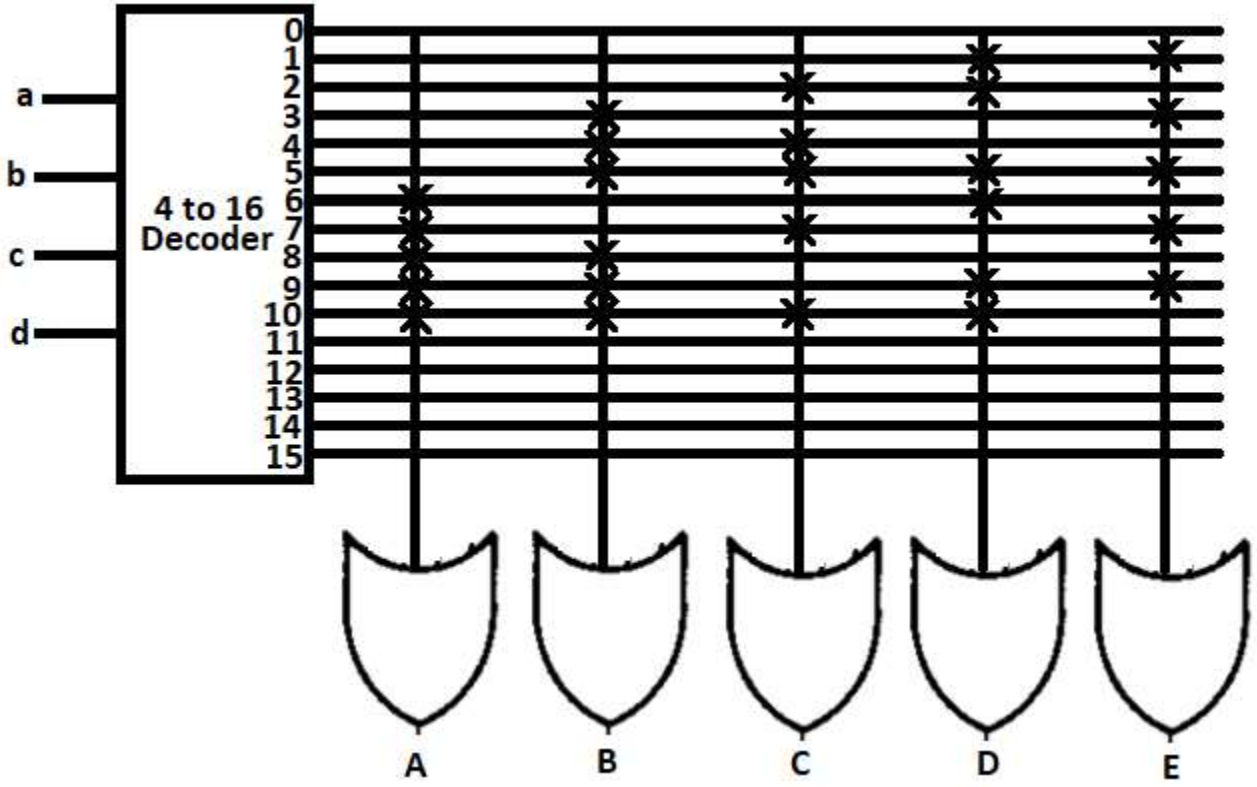
BCD to Excess-3 Code Converter using ROM

Input BCD				Output 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



Look-up Table Example: Multiplication Table of 3

Input (Multiplication Factor)		Output (Product)	
Decimal	Binary (a,b,c,d)	Binary (*A,B,C,D,E*)	Decimal
1	0 0 0 1	0 0 0 1 1	3
2	0 0 1 0	0 0 1 1 0	6
3	0 0 1 1	0 1 0 0 1	9
4	0 1 0 0	0 1 1 0 0	12
5	0 1 0 1	0 1 1 1 1	15
6	0 1 1 0	1 0 0 1 0	18
7	0 1 1 1	1 0 1 0 1	21
8	1 0 0 0	1 1 0 0 0	24
9	1 0 0 1	1 1 0 1 1	27
10	1 0 1 0	1 1 1 1 0	30



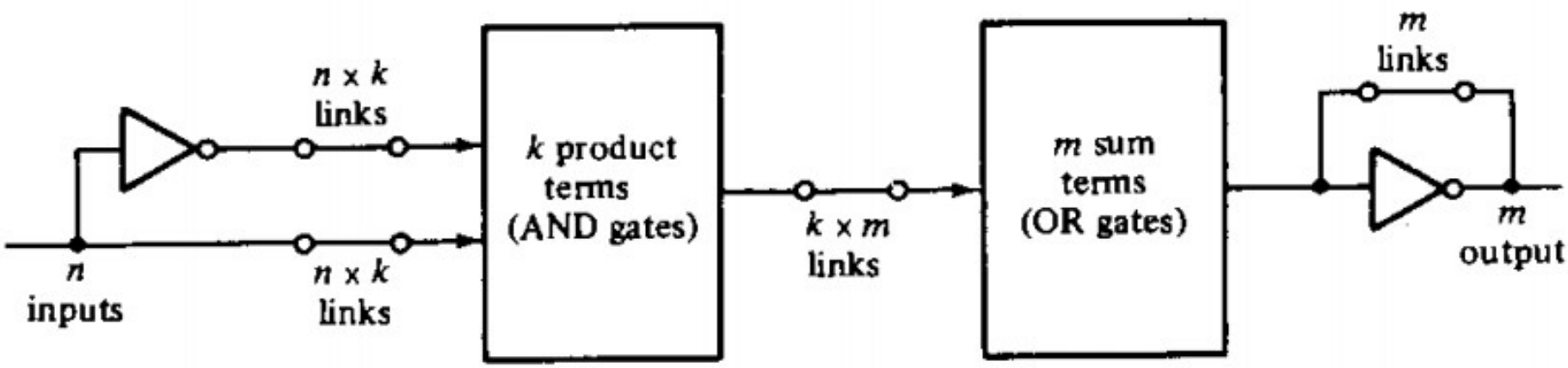
Types of ROM

- ROM : Read Only Memory
  - Programmed in the factory
- PROM: Programmable Read Only Memory
  - May be programmed in the field
- EPROM: Erasable Programmable Read Only Memory
  - Erasure through Ultraviolet exposure
- EEPROM or E^2PROM: Electrically Erasable Programmable Read Only Memory
  - Erasure also through electrical signal
- Versatility keeps going up and so is the cost

Programmable Logic Array (PLA)

- Both AND array as well as OR array is programmable

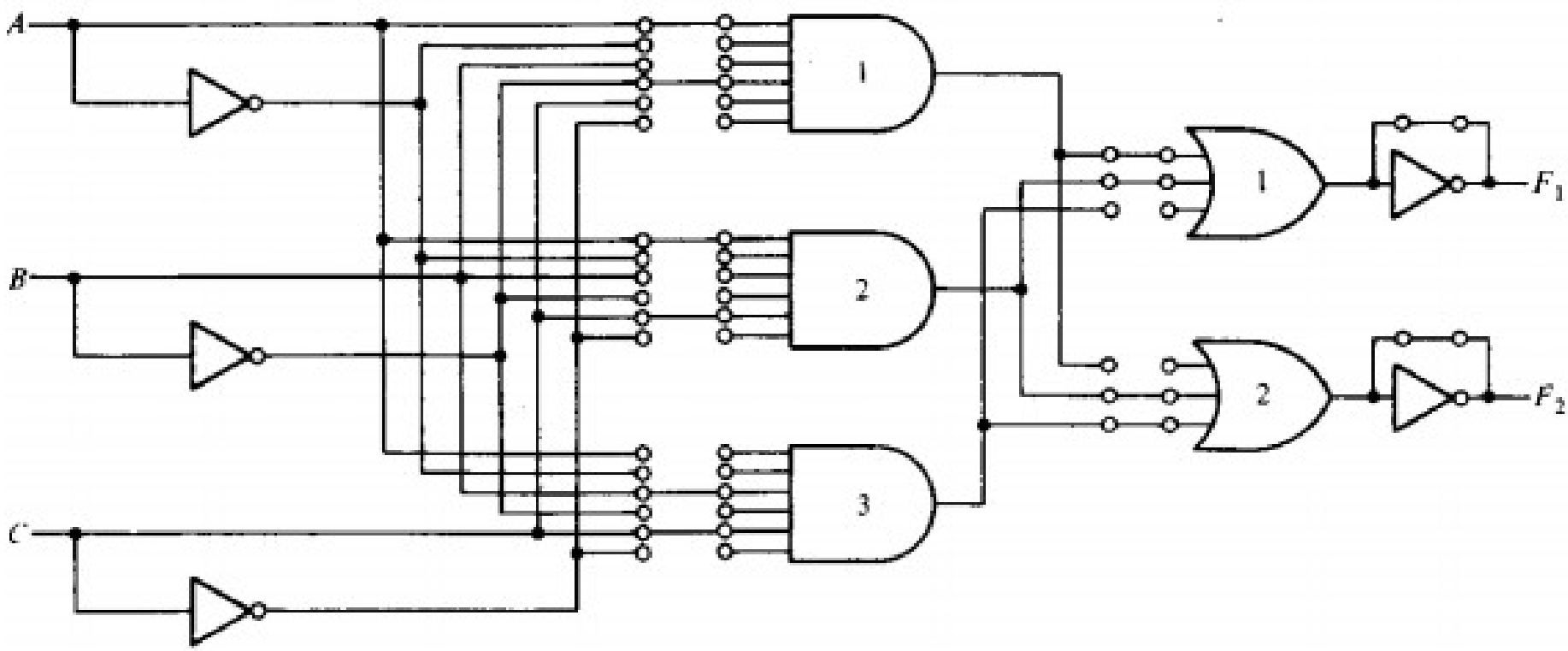
Block Diagram of PLA



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Example of PLA

$$F_1 = AB' + AC\ , \qquad F_2 = AC + BC$$



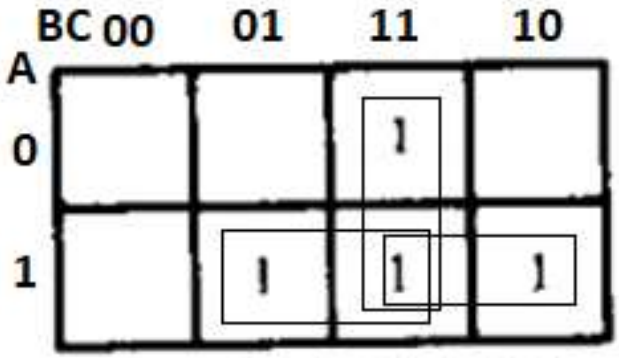
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

COC2070 – Digital Logic and System Design

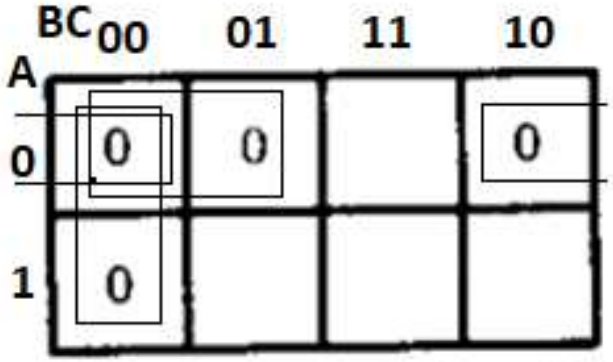
Prof. M. M. Sufyan Beg
Department of Computer Engineering
Z. H. College of Engineering & Technology
Aligarh Muslim University, India

PLA Implementation

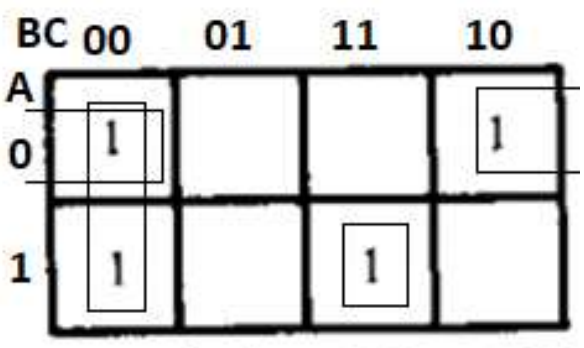
$$F_1(A, B, C) = \Sigma(3, 5, 6, 7), F_2(A, B, C) = \Sigma(0, 2, 4, 7)$$



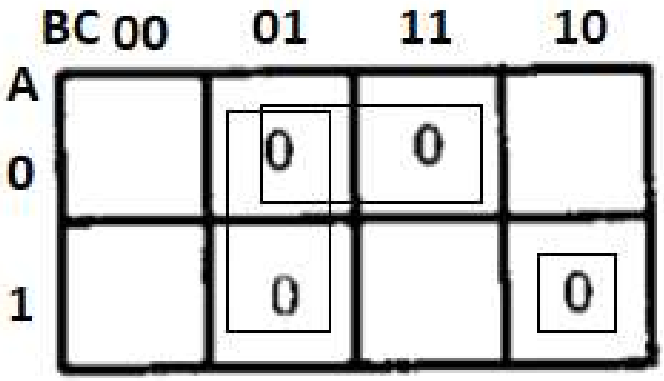
$F_1 = AC + AB + BC$



$F_1' = B'C' + A'C' + A'B'$



$F_2 = B'C' + A'C' + ABC$



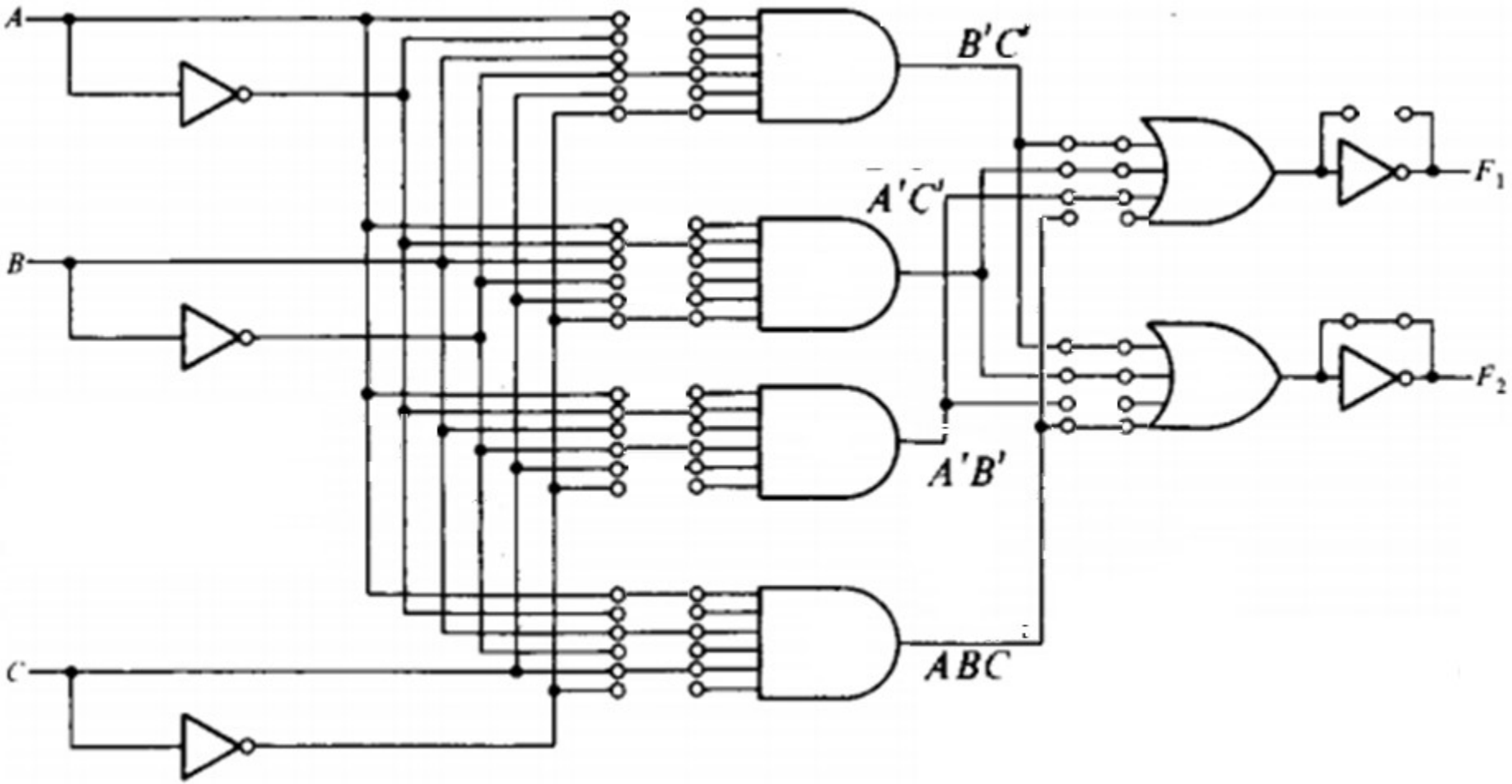
$F_2' = B'C + A'C + ABC'$

PLA program table

	Product term	Inputs A	B	C	Output F_1	F_2
$B'C'$	1	–	0	0	1	1
$A'C'$	2	0	–	0	1	1
$A'B'$	3	0	0	-	1	–
ABC	4	1	1	1	–	1
					C	T	T/C

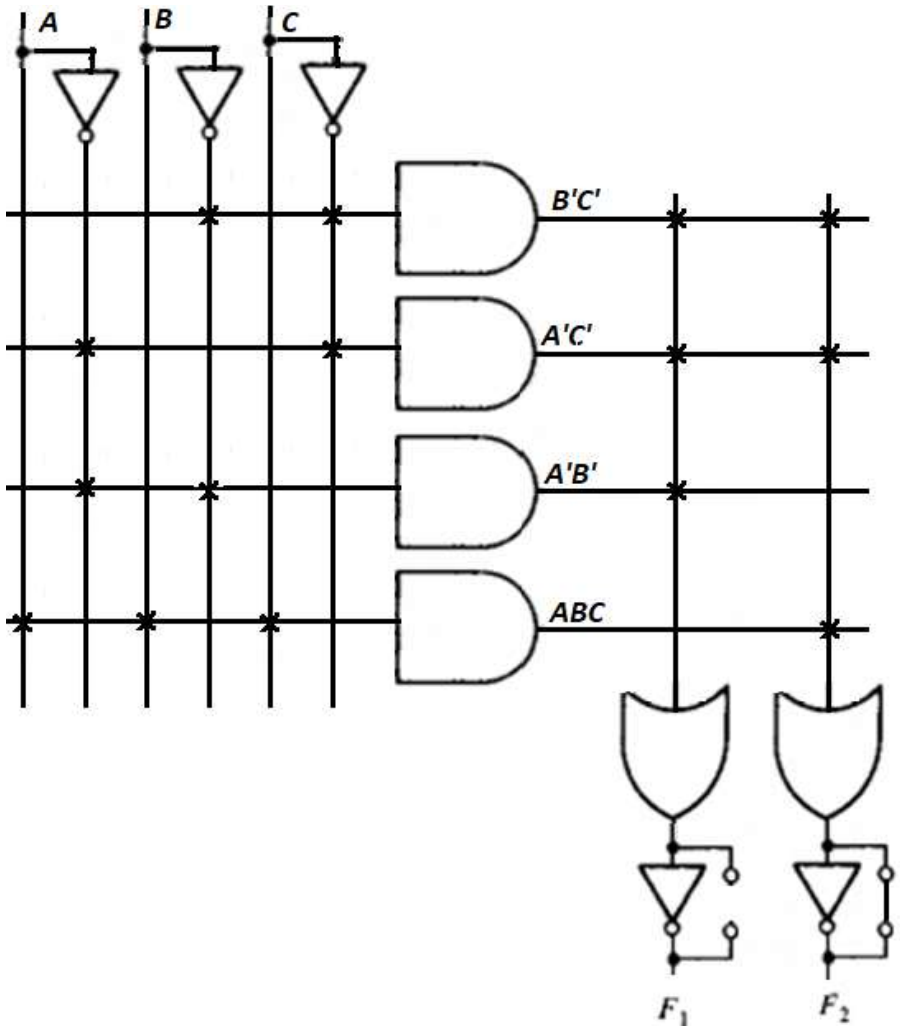
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

PLA Implementation (continued)



$$F_1 = (B'C' + A'C' + A'B')' \,,\; F_2 = B'C' + A'C' + ABC$$

Line Diagram of the PLA



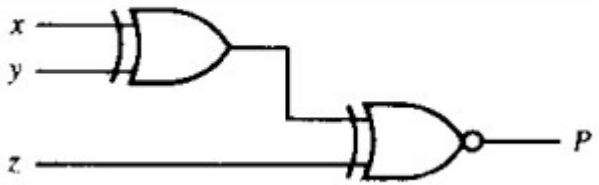
$$F_1 = (B'C' + A'C' + A'B')' \, , \, F_2 = B'C' + A'C' + ABC$$

Parity Generator/Checker

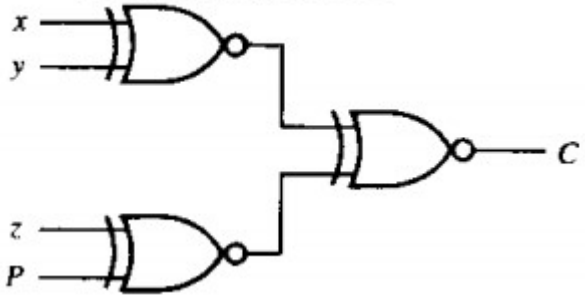
Odd-parity generation

Three-bit message			Parity bit generated
x	y	z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

3-bit odd parity generator



4-bit odd parity checker

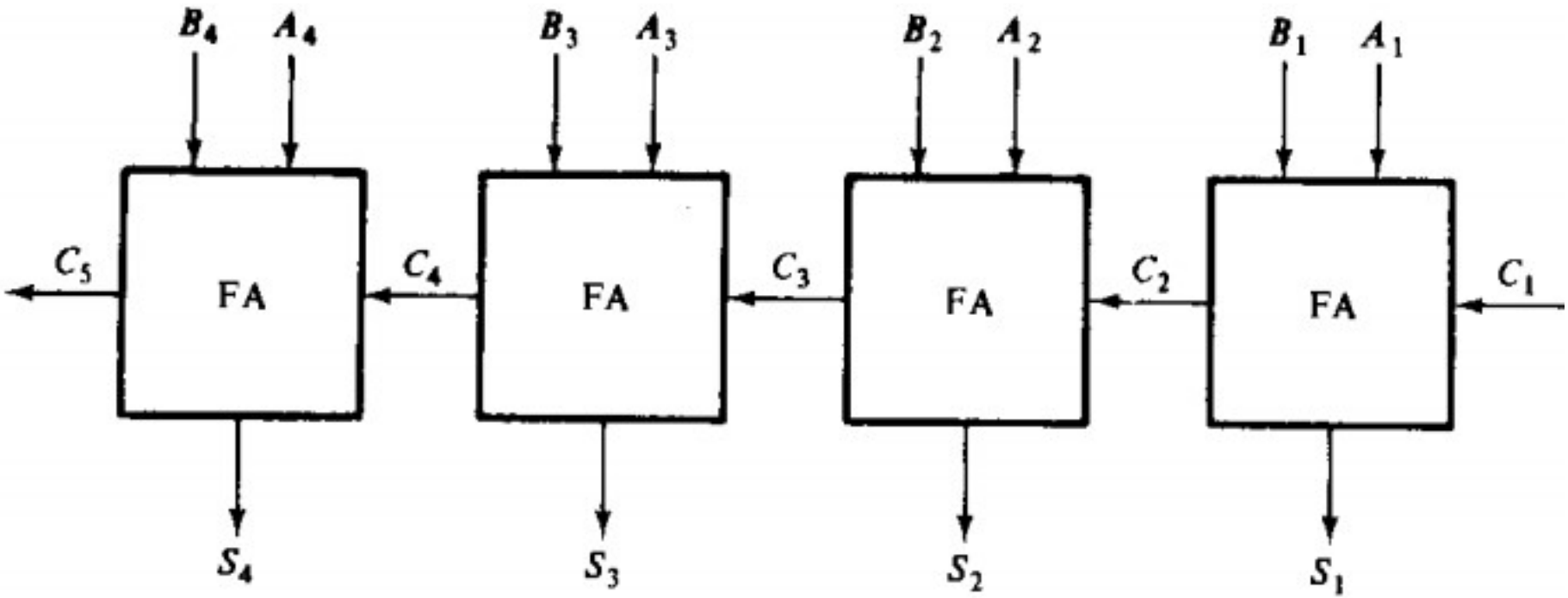


Odd-parity check

Four-bits received				Parity-error check
x	y	z	P	C
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Parallel Adder (Ripple Adder)

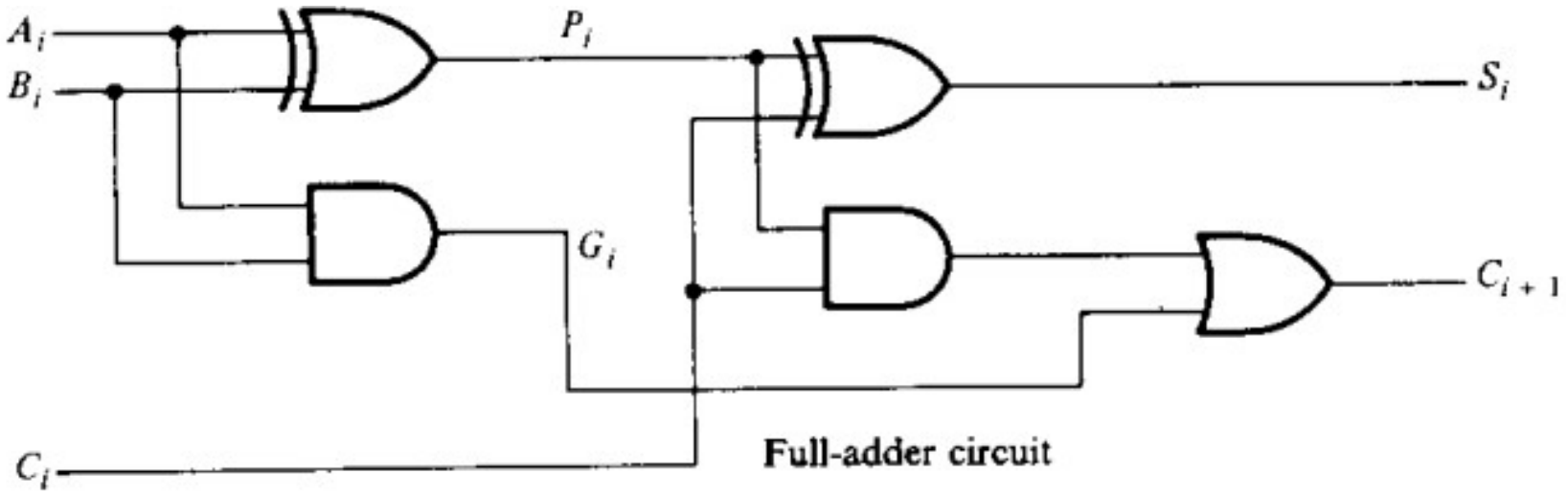


	C_5	C_4	C_3	C_2	C_1	←	C
		A_4	A_3	A_2	A_1	←	A
+		B_4	B_3	B_2	B_1	←	B
		S_4	S_3	S_2	S_1	←	S

	1	0	1	1		←	C
		1	0	1	1	←	A
+		1	0	0	1	←	B
	0	1	0	0		←	S

Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Carry Look Ahead Adder



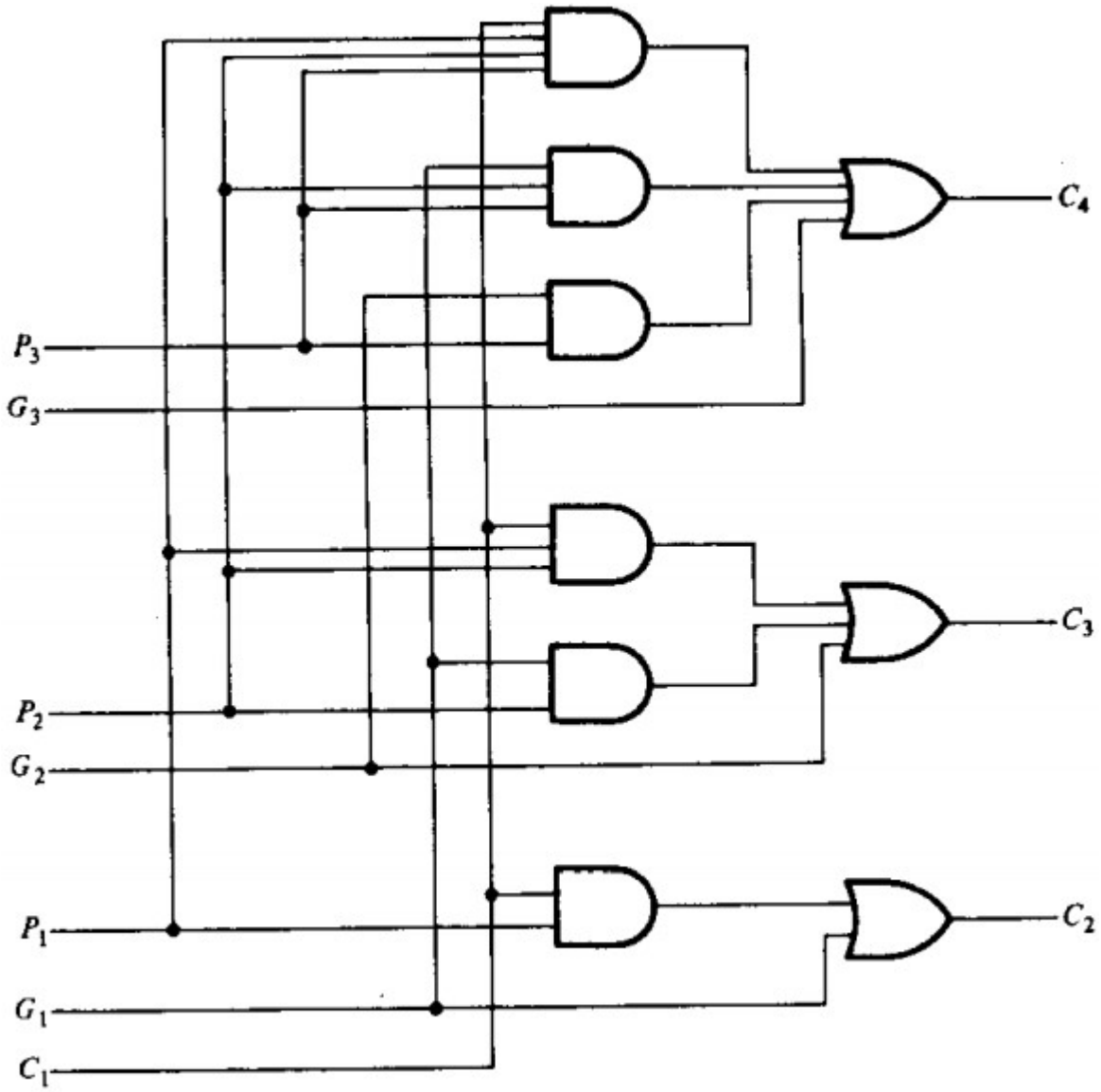
Full-adder circuit

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

$$C_2 = G_1 + P_1 C_1$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$$
$$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$$

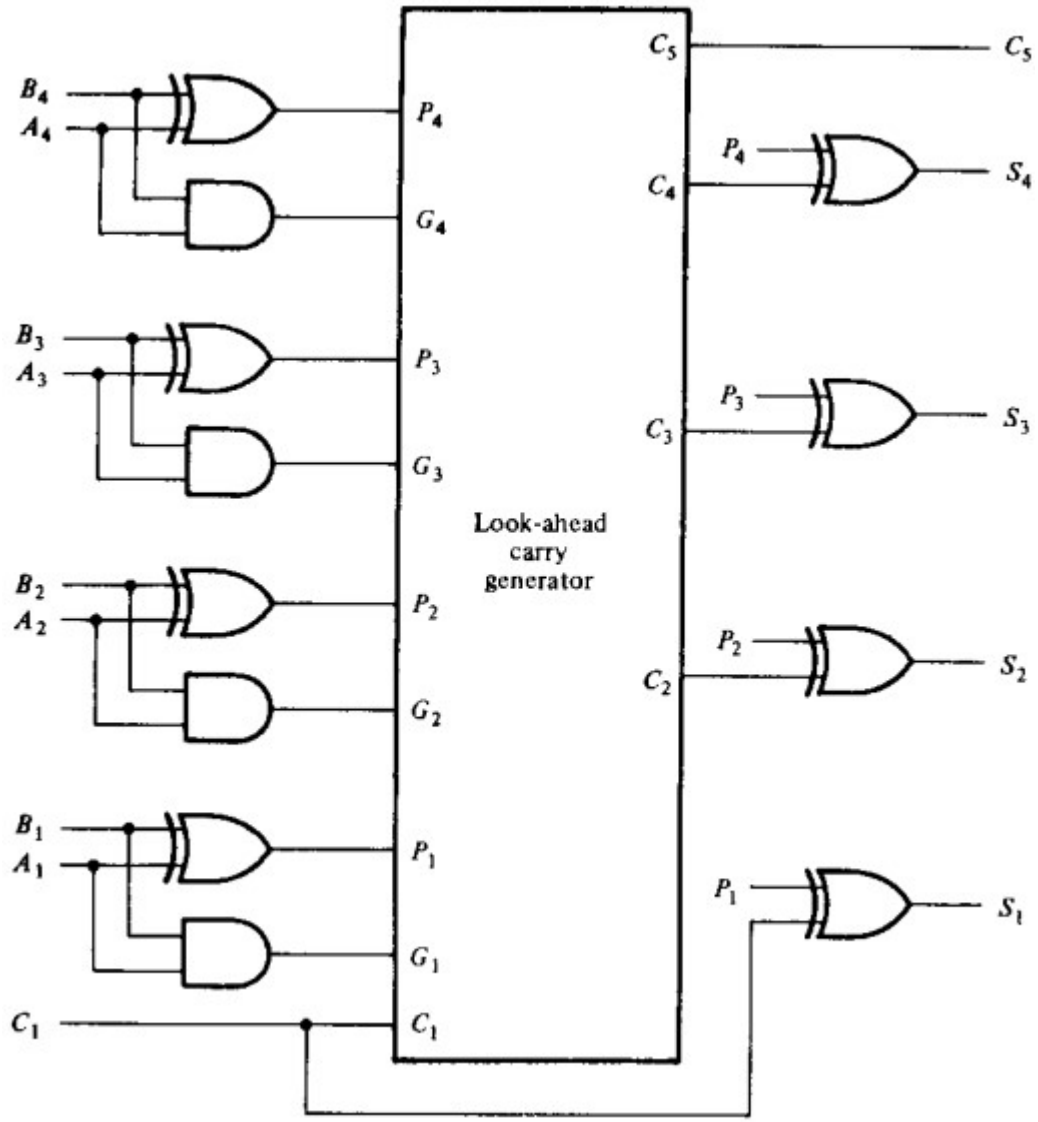
Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Logic Diagram of a Carry Look Ahead Generator



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

4-bit Full Adders with Look Ahead Carry



Reference: M. Morris Mano, “Digital Logic and Computer Design”, PHI

Unit-II Completed

UNIT II.
COMBINATIONAL LOGIC
Hardware aspect of arithmetic logic functions, Half-Adder, Full-Adder, Binary Adder/Subtractor, Parallel Adder, Magnitude Comparator, Demultiplexer, Multiplexer, Parity Checker/Generator, ROM, etc.

Course Outcome

2. Analyse and synthesize digital combinational units.