Digital Logic and Design

Combinational Circuits

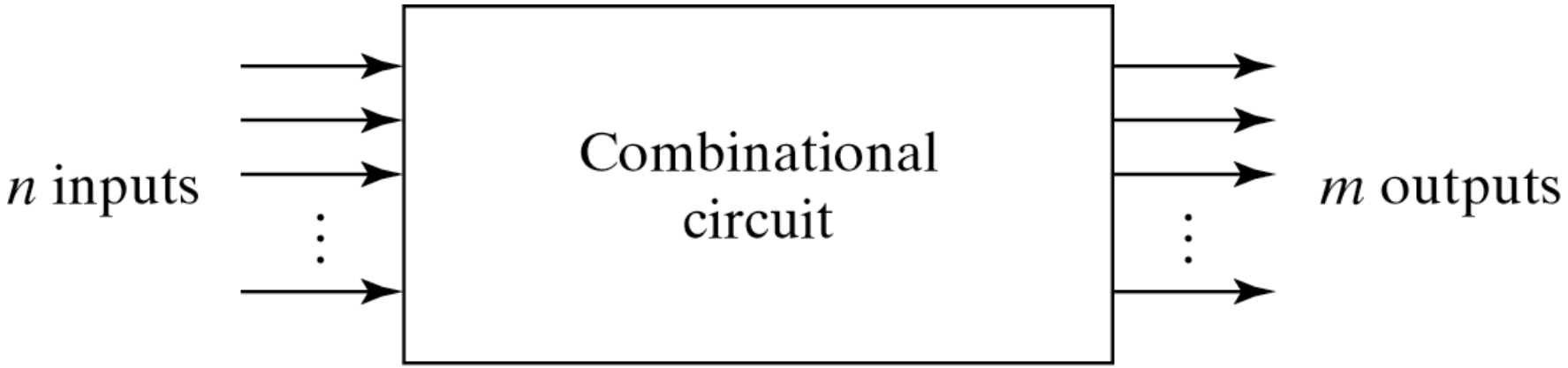
Half/Full adder, Subtractor, Decoder, Encoder, MUX, DeMUX etc.

Santosh Kumar Verma
Mentor-GeeksforGeeks

What is Combinational Circuits?

- A Combinational Circuit is a combination of Logic gates, the output depends upon the current value of the inputs.

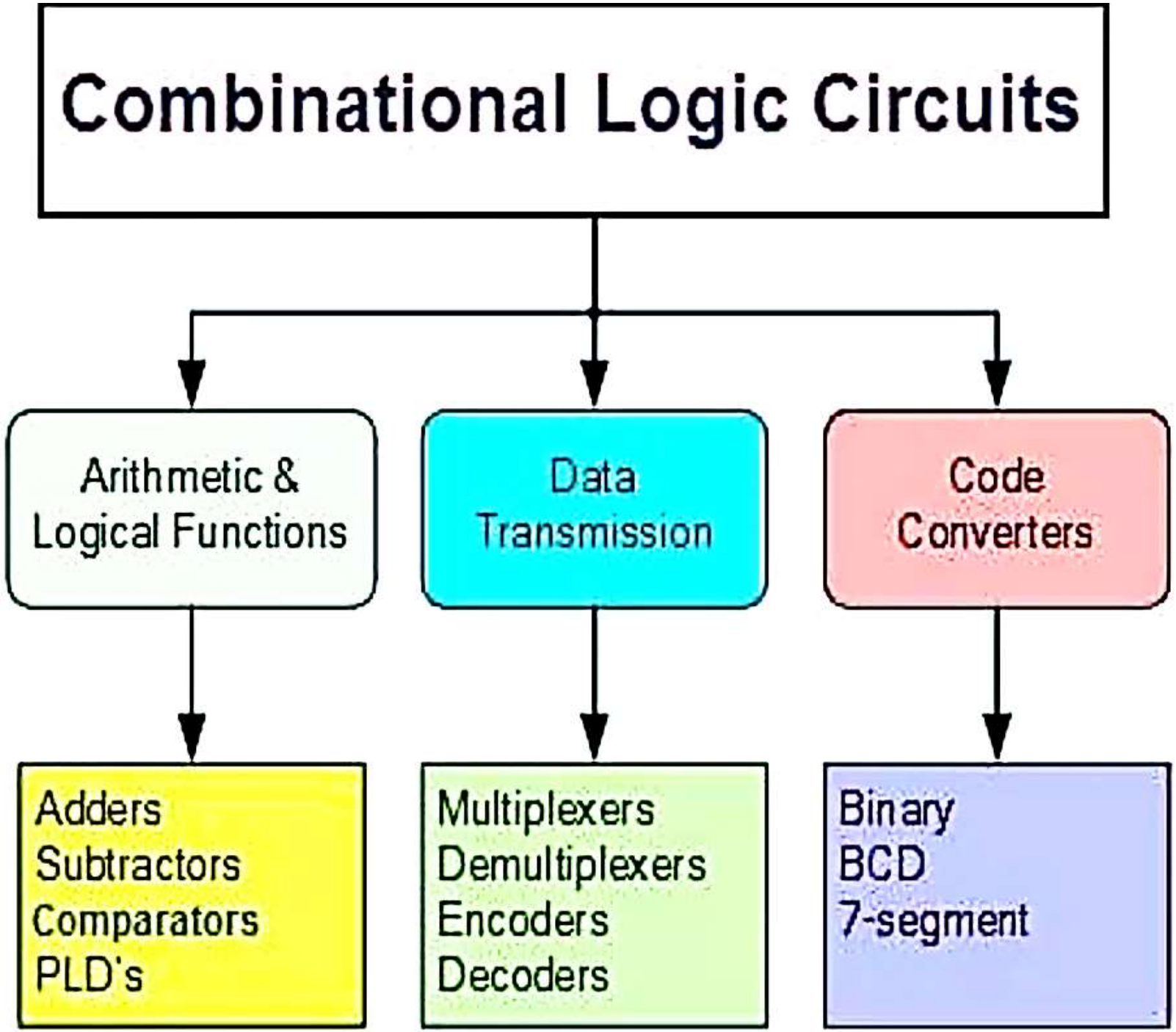
Combinational Circuits



Block Diagram of Combinational Circuit

Examples of Combinational Circuits

- Addition:
  - **Half Adder (HA)**
  - **Full Adder (FA)**
  - **BCD(Decimal) Adder**
- Subtraction:
  - **Half Subtractor**
  - **Full Subtractor**
- Multipliers:
- Comparator:
  - **Magnitude Comparator**
- Multiplexers
- Demultiplexers
- Encoders
- Decoders
- Converters
  - Binary to Gray Code
  - Gray to Binary Code
  - Binary to BCD Code


Combinational Logic Circuits
Arithmetic & Logical Functions
Data Transmission
Code Converters
Adders
Subtractors
Comparators
PLD's
Multiplexers
Demultiplexers
Encoders
Decoders
Binary
BCD
7-segment

Designing Combinational Circuits

In general we have to do following steps:

1. Problem description
2. Input/output of the circuit
3. Define truth table
4. Simplification for each output
5. Draw the circuit

Half Adder

- Adding two single-bit binary values, X, Y produces a sum S bit and a carry out C-out bit.
- This operation is called half addition and the circuit to realize it is called a half adder.

Half Adder Truth Table

Inputs		Outputs	
X	Y	S	C-out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

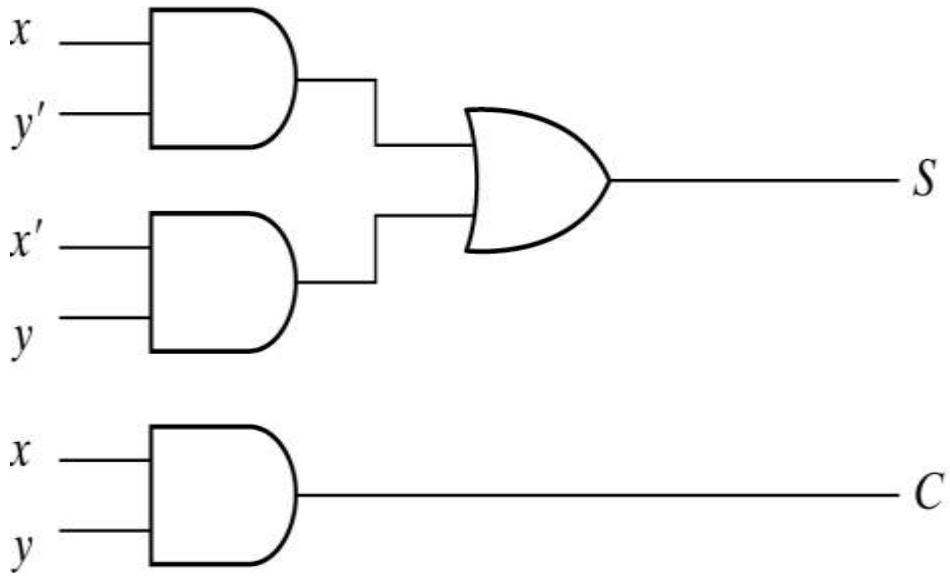
$S(X,Y) = \Sigma\ (1, 2)$

$S = X'Y + XY'$

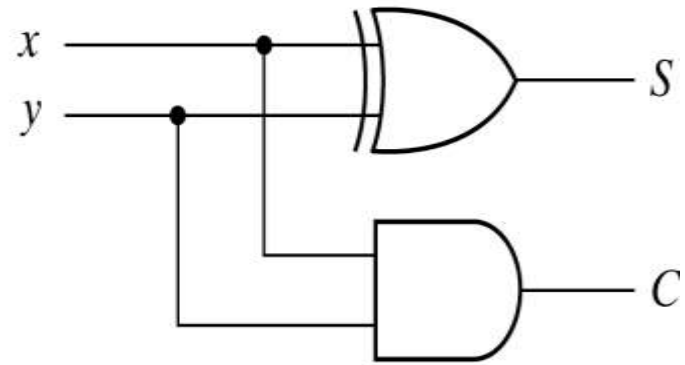
$S = X \oplus Y$

$C\text{-}out(x, y) = \Sigma\ (3)$

$C\text{-}out = XY$



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



(b) $S = x \oplus y$
$C = xy$

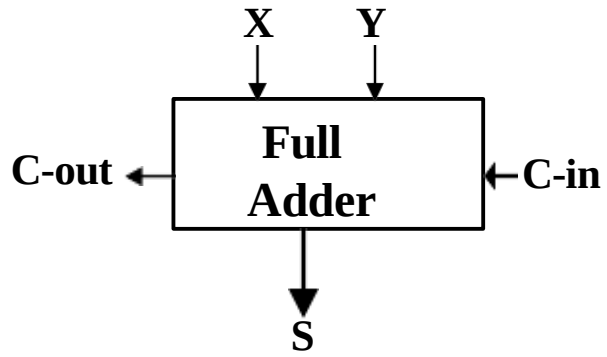
Implementation of Half-Adder

Full Adder

- Adding two single-bit binary values, X, Y with a carry input bit C-in produces a sum bit S and a carry out C-out bit.

Full Adder Truth Table

Inputs			Outputs	
X	Y	C-in	S	C-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Full Adder

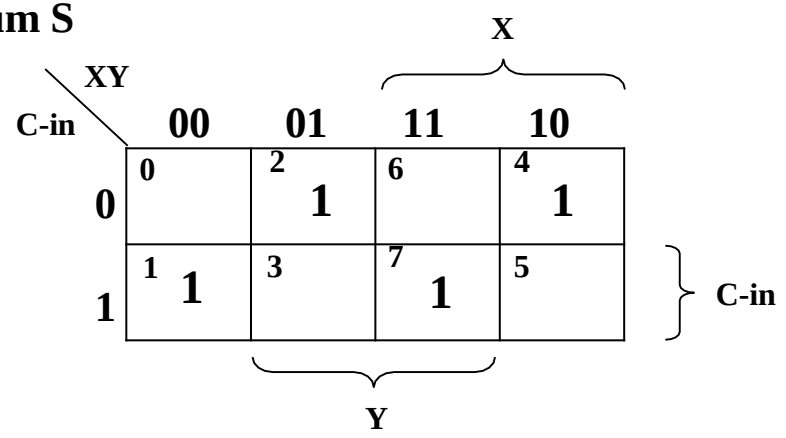
Full Adder Truth Table

Inputs			Outputs	
X	Y	C-in	S	C-out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S(X,Y, C\text{-in}) = \Sigma\ (1,2,4,7)$

$C\text{-out}(x, y, C\text{-in}) = \Sigma\ (3,5,6,7)$

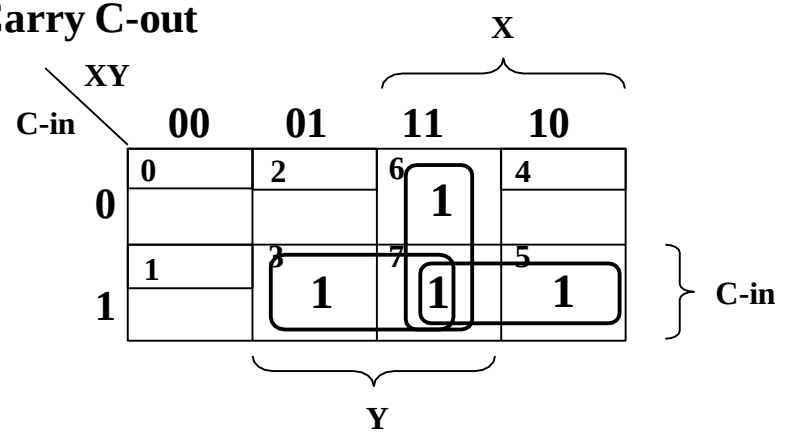
Sum S



$S = X'Y'(C\text{-in}) + XY'(C\text{-in})' + X'Y(C\text{-in})' + XY(C\text{-in})$

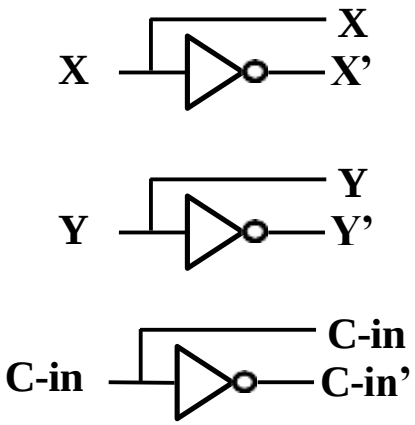
$S = X \oplus Y \oplus (C\text{-in})$

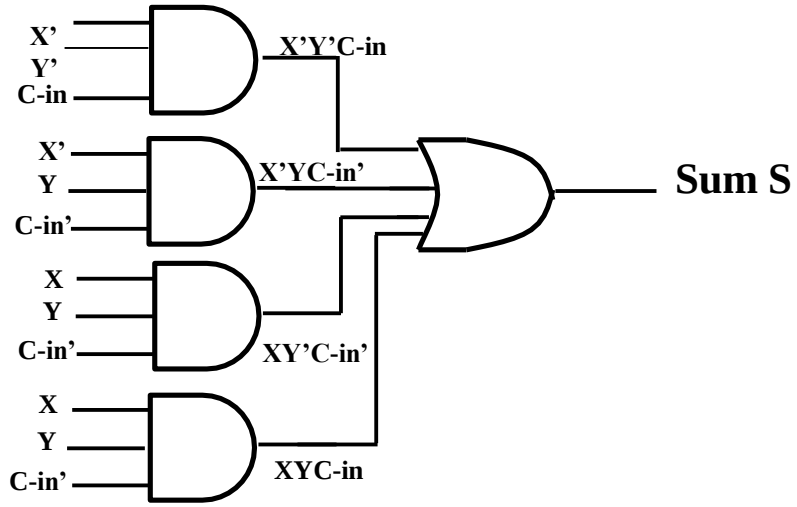
Carry C-out

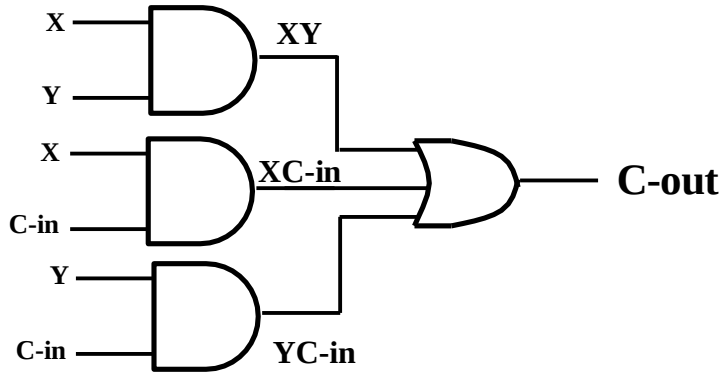


$C\text{-out} = XY + X(C\text{-in}) + Y(C\text{-in})$

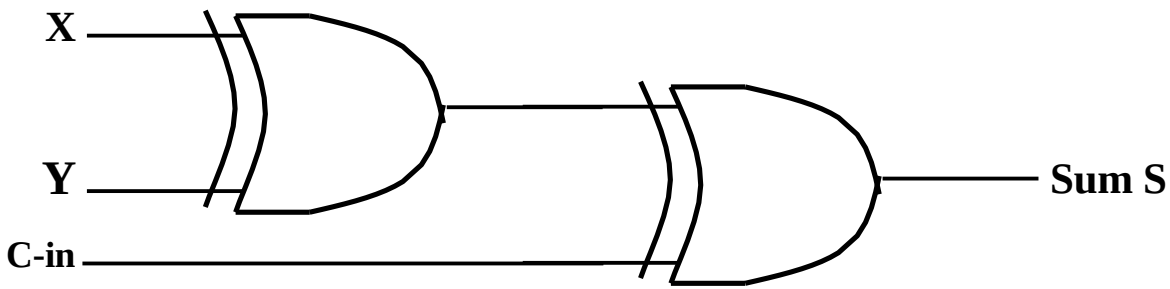
Full Adder Circuit Using AND-OR

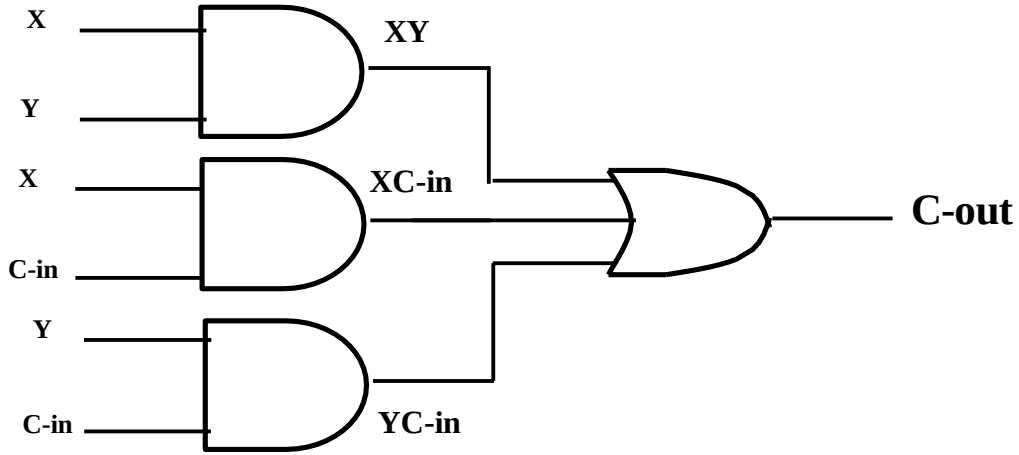




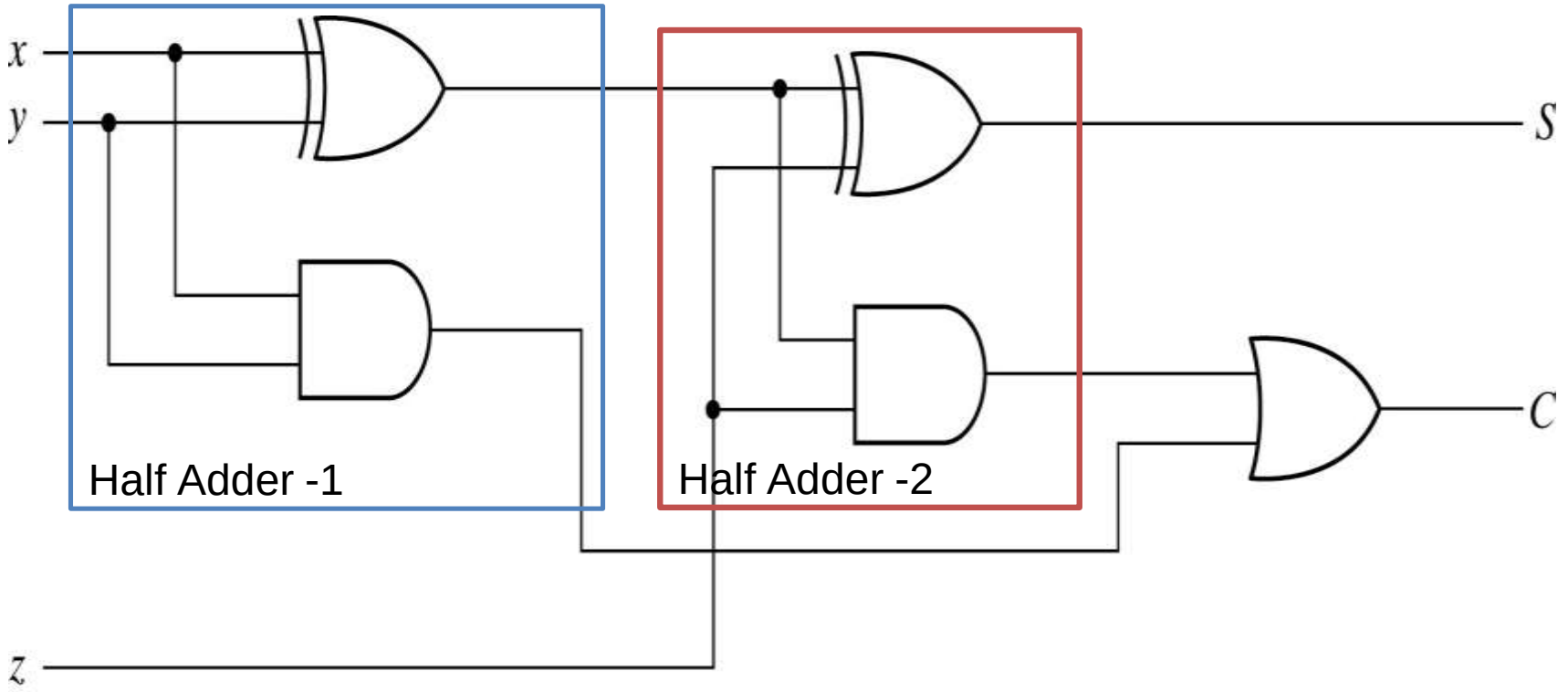


Full Adder Circuit Using Ex-OR





Full Adder Circuit Using two half - Adders



Implementation of Full Adder with Two Half Adders and an OR Gate

Binary adder

- Binary adder that produces the arithmetic sum of binary numbers can be constructed with full adders connected in cascade, with the output carry from each full adder is connected to the input carry of the next full adder in the chain

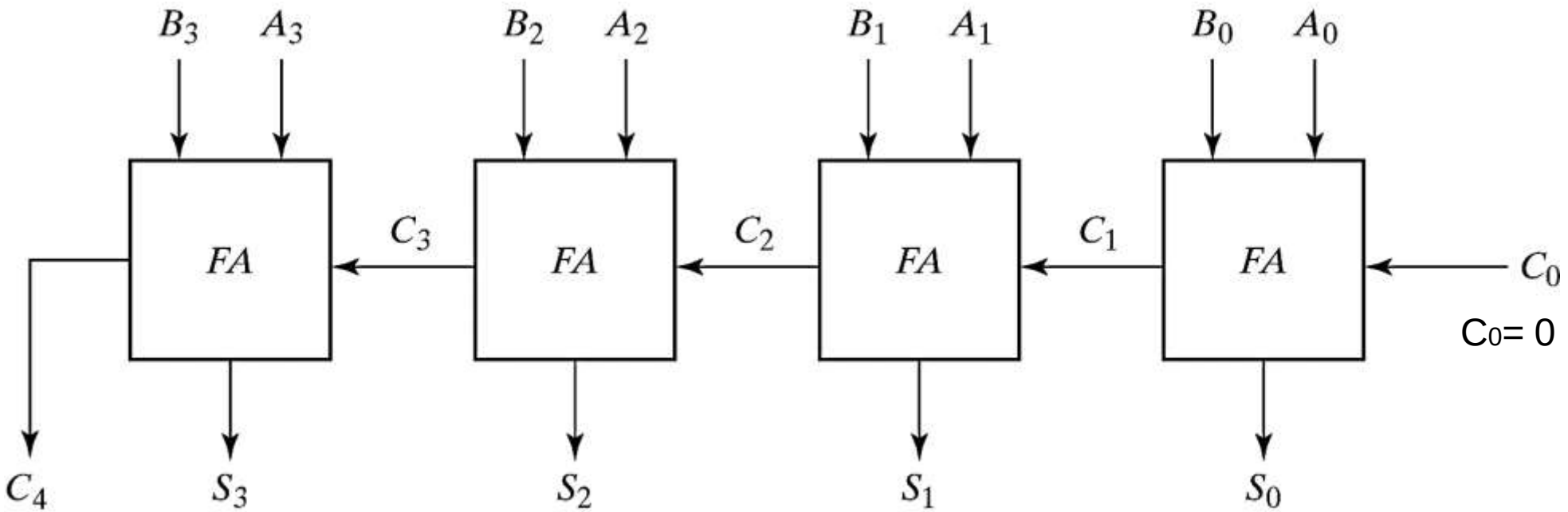
- Note that the input carry C_0 in the least significant position must be 0.

Binary Adder

- For example to add A= 1011 and B= 0011

subscript i:	3	2	1	0	
Input carry:	0	1	1	0	C_i
Augend:	1	0	1	1	A_i
Addend:	0	0	1	1	B_i
Sum:	1	1	1	0	S_i
Output carry:	0	0	1	1	C_{i+1}

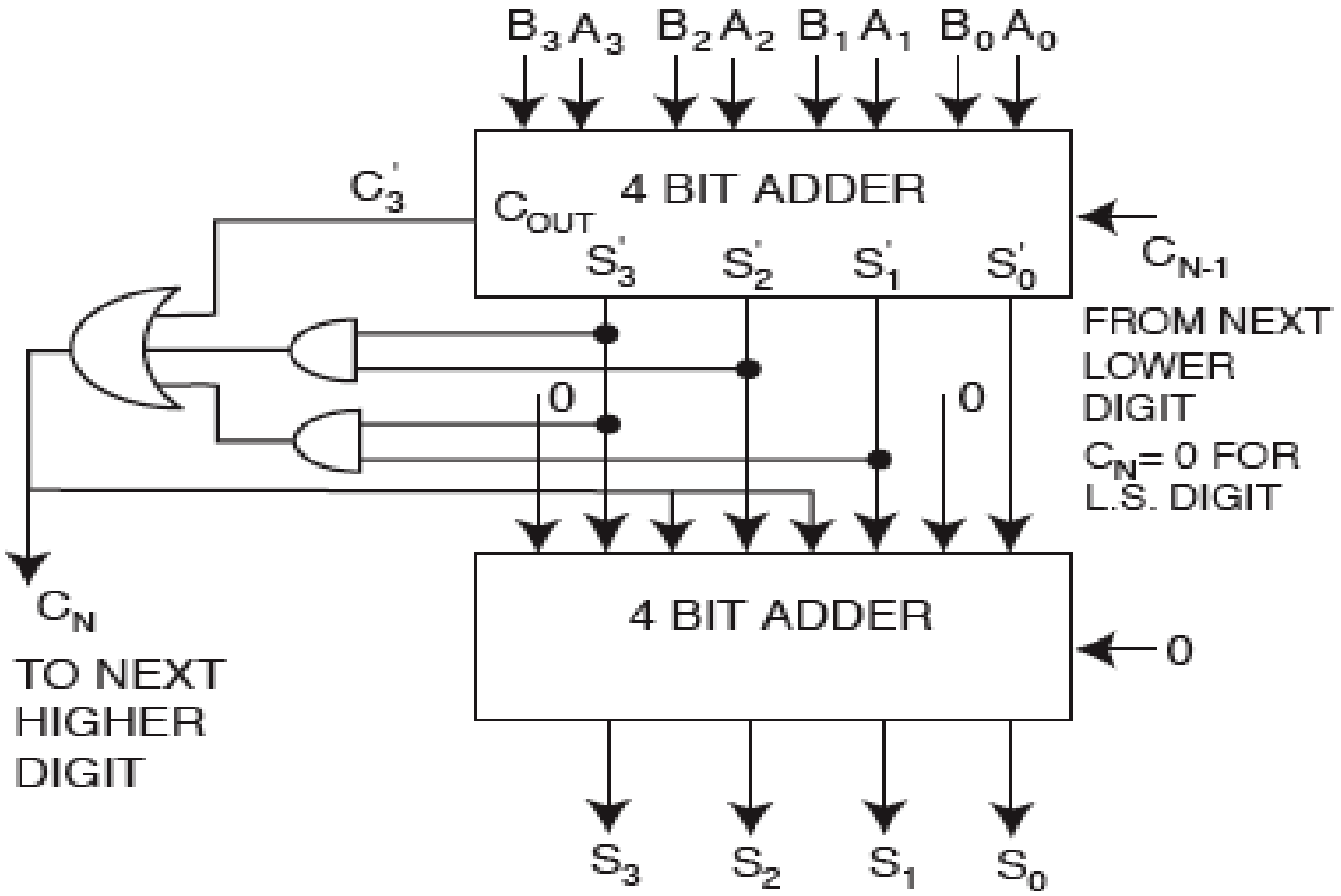
Binary Adder



4-Bit Adder

$C_0 = 0$

DECIMAL/BCD ADDER



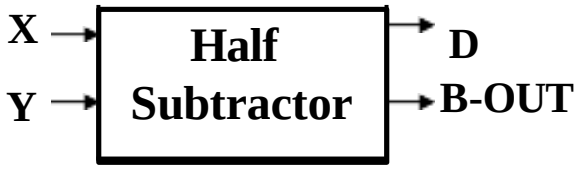
- ADD **0110** WHEN C_N=1
- ADD **0000** WHEN C_N=0

Half Subtractor

- Subtracting a single-bit binary value Y from anther X (I.e. X - Y) produces a difference bit D and a borrow out bit B-out.
- This operation is called half subtraction and the circuit to realize it is called a half subtractor.

Half Subtractor Truth Table

Inputs		Outputs	
X	Y	D	B-out
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

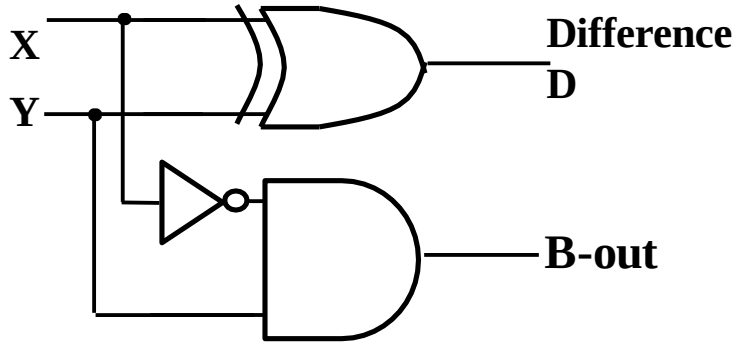


$$D(X,Y) = \Sigma\,(1,2) \quad D = X'Y + XY'$$

$$D = X \oplus Y$$

$$B\text{-}out(x, y, C\text{-}in) = \Sigma\,(1)$$

$$B\text{-}out = X'Y$$



Binary Arithmetic Operations Subtraction

- Two binary numbers are subtracted by subtracting each pair of bits together with borrowing, where needed.
- Subtraction Example:

		0 0 1 1 1 1 1 0 0	Borrow
X	229	1 1 1 0 0 1 0 1	
Y -	46	- 0 0 1 0 1 1 1 0	
	183	1 0 1 1 0 1 1 1	

Full Subtractor

- Subtracting two single-bit binary values, Y, B-in from a single-bit value X produces a difference bit D and a borrow out B-out bit. This is called full subtraction.

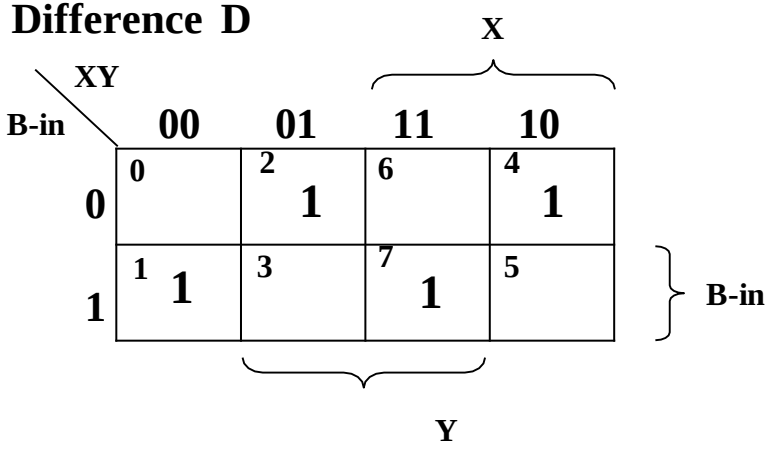
Full Subtractor Truth Table

Inputs			Outputs	
X	Y	B-in	D	B-out
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$S(X,Y, C\text{-in}) = \Sigma\ (1,2,4,7)$

$C\text{-out}(x, y, C\text{-in}) = \Sigma\ (1,2,3,7)$

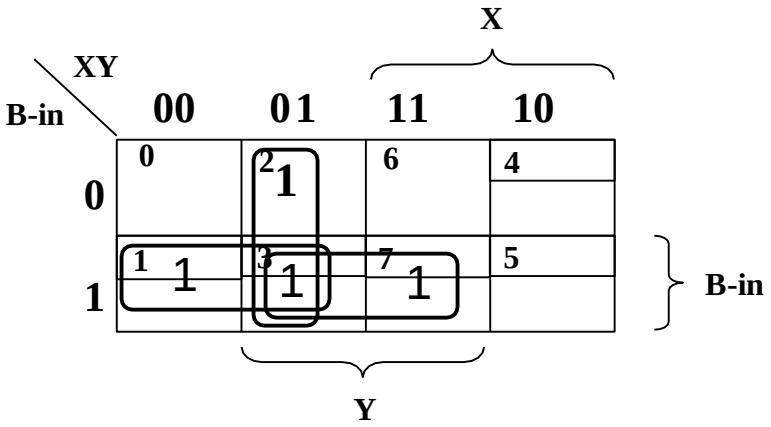
Difference D



$S = X'Y'(B\text{-in}) + XY'(B\text{-in})' + XY'(B\text{-in})' + XY(B\text{-in})$

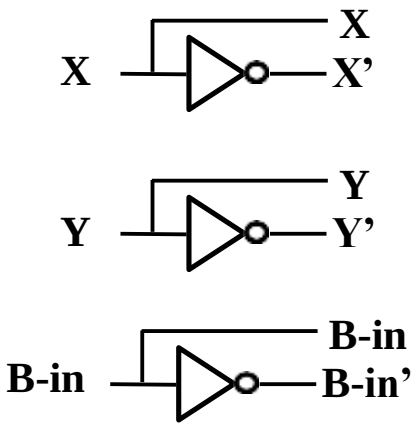
$S = X \oplus Y \oplus (C\text{-in})$

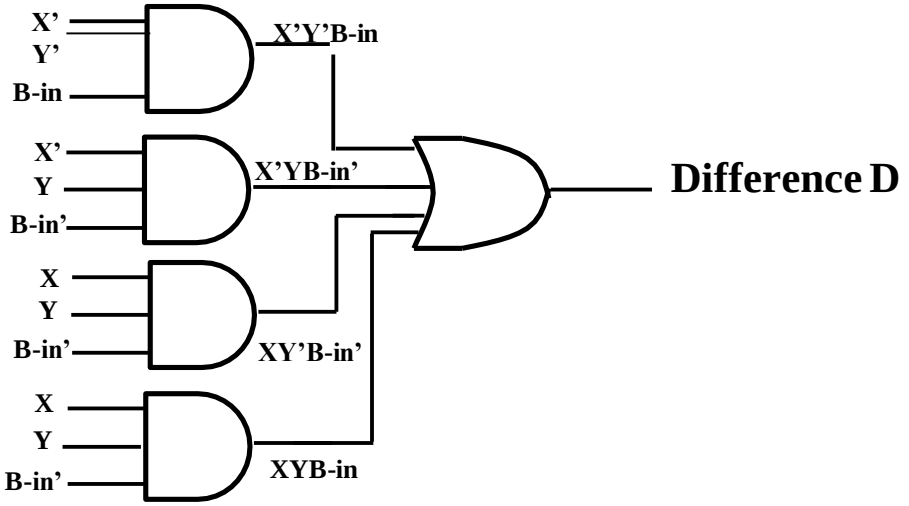
Borrow B-out

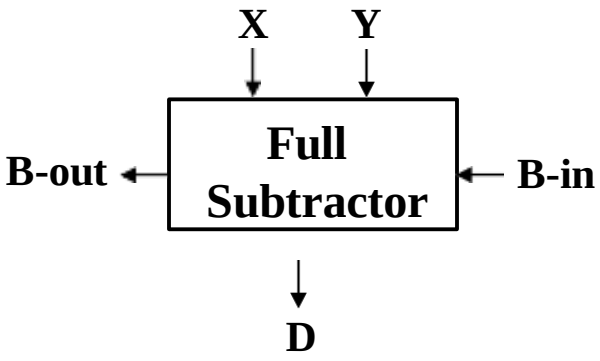


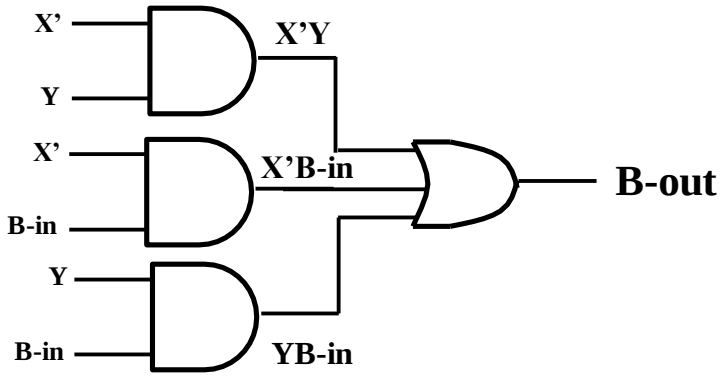
$B\text{-out} = X'Y + X'(B\text{-in}) + Y(B\text{-in})$

Full Subtractor Circuit Using AND-OR

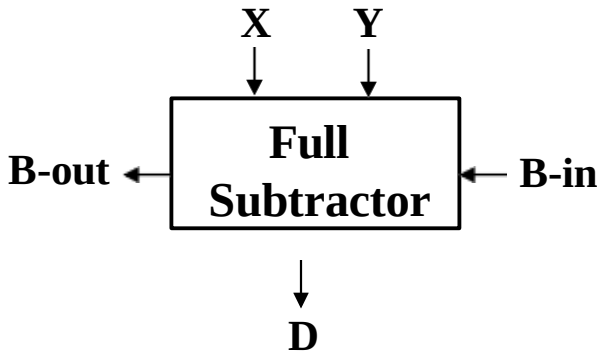


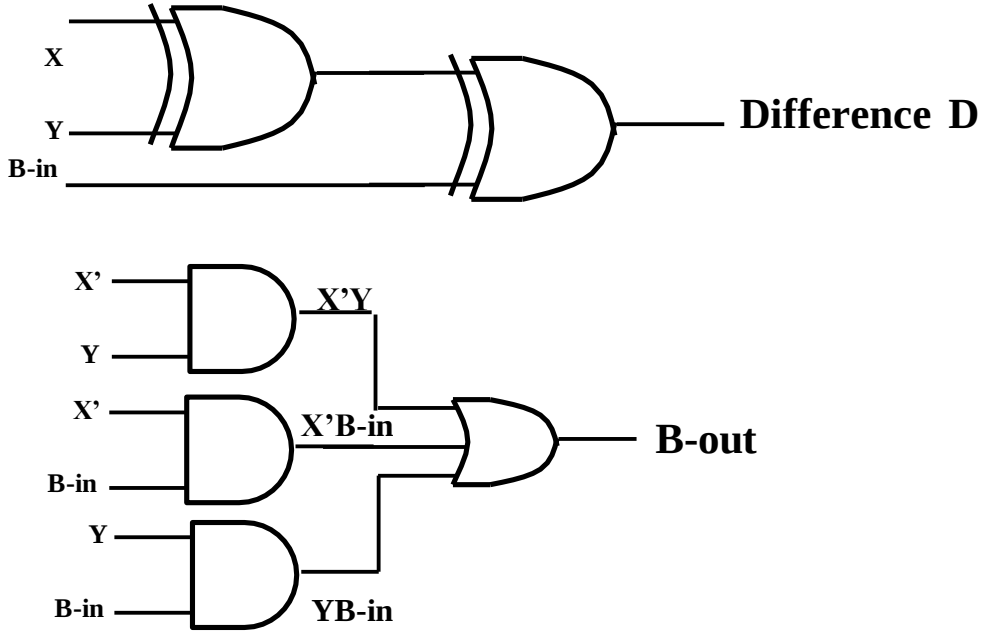






Full Subtractor Circuit Using XOR





n-bit Subtractors

An n-bit subtracor used to subtract an n-bit number Y from another n-bit number X (i.e X-Y) can be built in one of two ways:

- By using n bit full subtractors and connecting them in series, creating a borrow ripple subtractor:
  - **Each borrow out B-out from a full subtractor at position j is connected to the borrow in B-in of the full subtracor at the higher position j+1.**

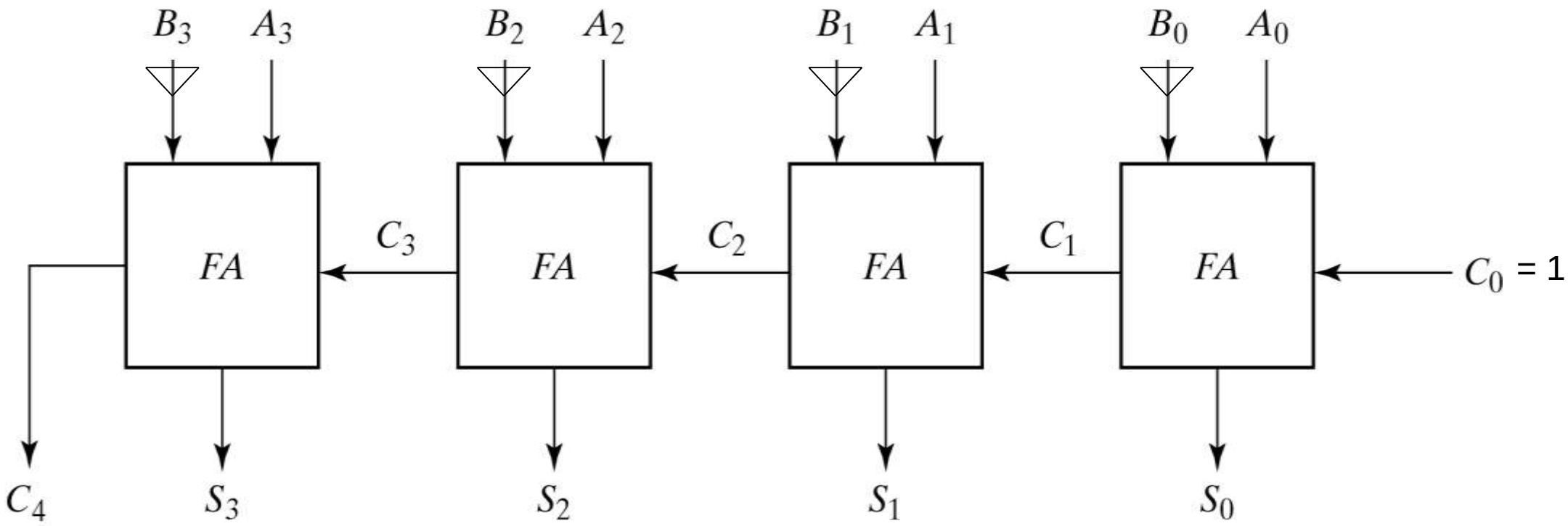
- By using an n-bit adder and n inverters:
  - **Find two's complement of Y by:**
    - Inverting all the bits of Y using the n inverters.
    - Adding 1 by setting the carry in of the least significant position to 1
  - **The original subtraction (X - Y) now becomes an addition of X to two's complement of Y using the n-bit adder.**

Binary Subtractor

- The subtrcation A – B can be done by taking the 2's complement of B and adding it to A because A- B = A + (-B)

- It means if we use the inveters to make 1's complement of B (connecting each Bi to an inverter) and then add 1 to the least significant bit (by setting carry C_0 to 1) of binary adder, then we can make a binary subtractor.

4 bit 2's complement Subtractor



4-Bit Adder

Adder Subtractor

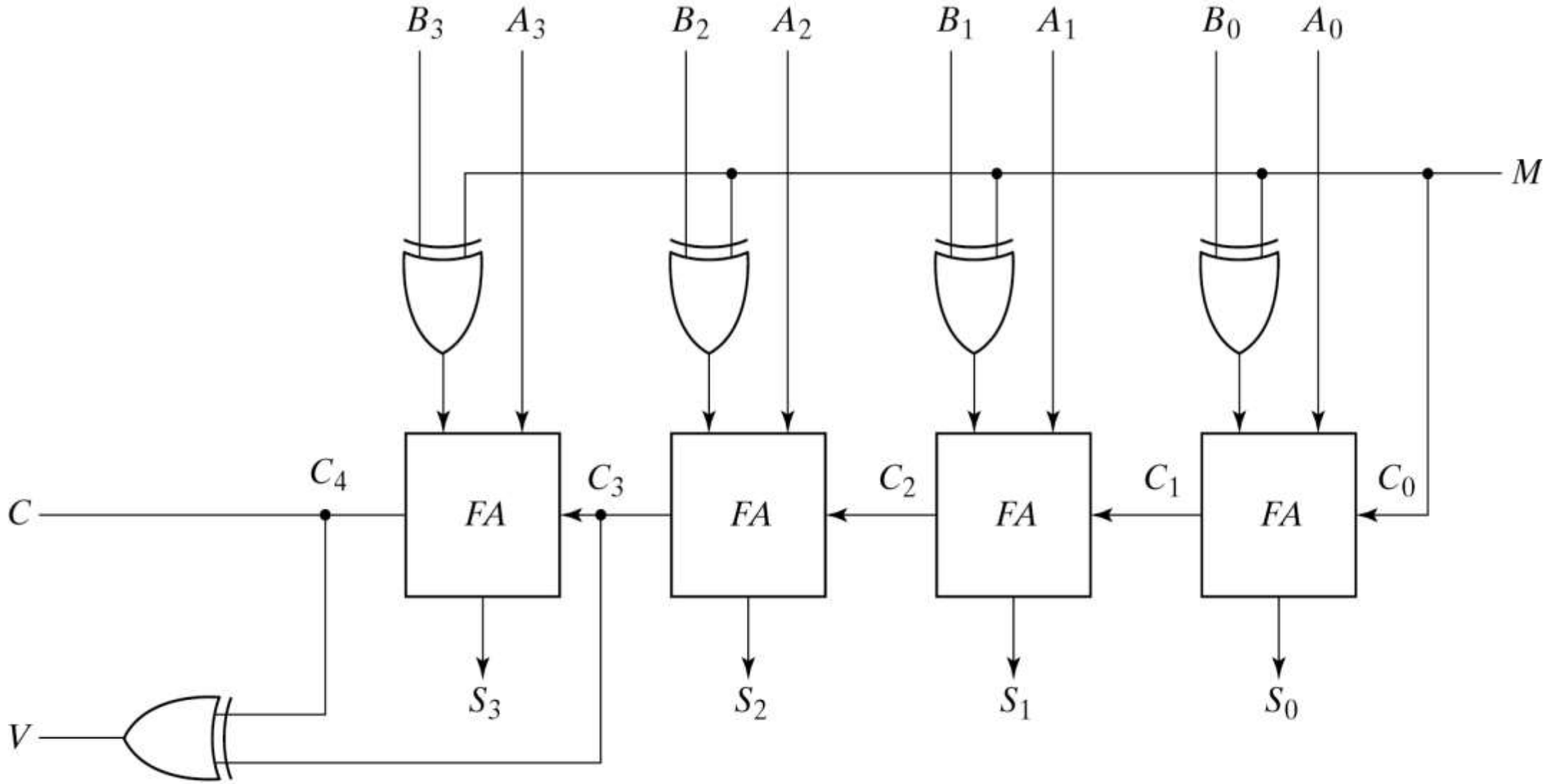
- The addition and subtraction can be combined into one circuit with one common binary adder (see next slide).

- The mode M controls the operation.

a) When M=0 the circuit is an adder when

b) M=1 the circuit is subtractor.

It can be don by using exclusive-OR for each Bi and M.
Note that $1 \oplus x = x'$ and $0 \oplus x = x$



4-Bit Adder Subtractor

Checking Overflow

- Note that in the previous slide if the numbers considered to be signed V detects overflow.

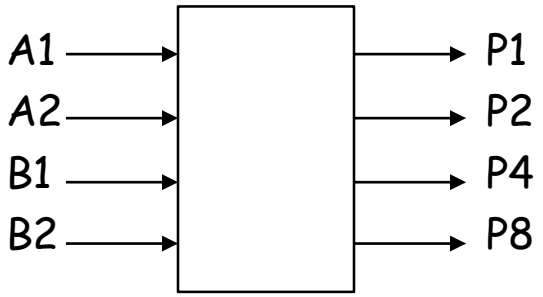
  a) V=0 means no overflow and

  b) V=1 means the result is wrong because of overflow

- Overflow can be happened when adding two numbers of the same sign (both negative or positive) and result can not be shown with the available bits.

- It can be detected by observing the carry into sign bit and carry out of sign bit position. If these two carries are not equal an overflow occurred. That is why these two carries are applied to exclusive-OR gate to generate V.

Design example: 2x2-bit multiplier

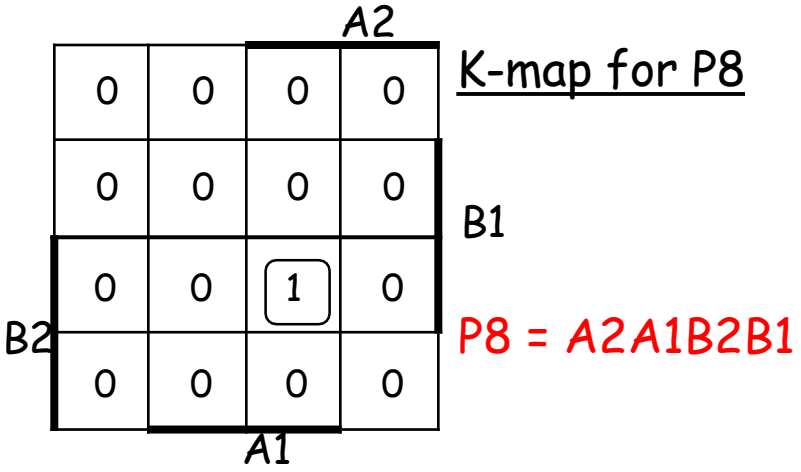


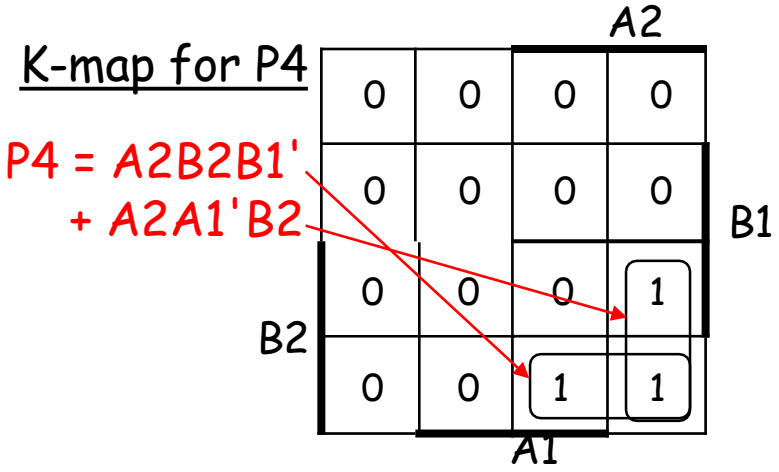
block diagram
and
truth table

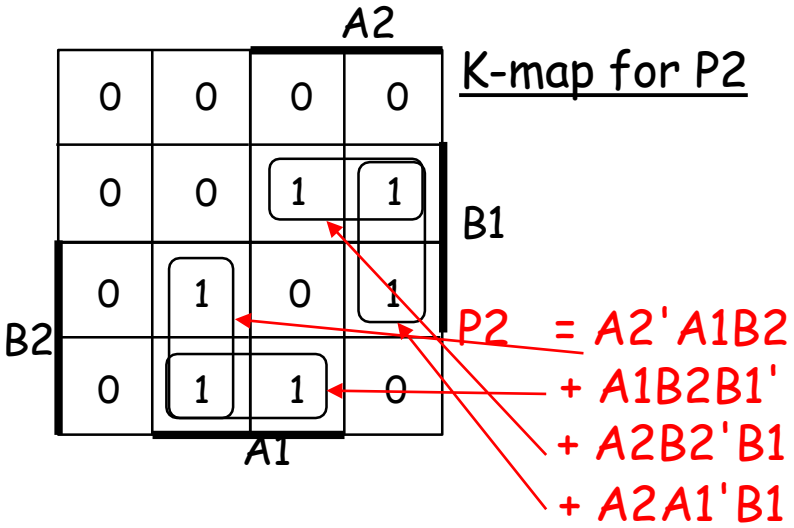
A2	A1	B2	B1	P8	P4	P2	P1
0	0	0	0	0	0	0	0
		0	1	0	0	0	0
		1	0	0	0	0	0
		1	1	0	0	0	0
0	1	0	0	0	0	0	0
		0	1	0	0	0	1
		1	0	0	0	1	0
		1	1	0	0	1	1
1	0	0	0	0	0	0	0
		0	1	0	0	1	0
		1	0	0	1	0	0
		1	1	0	1	1	0
1	1	0	0	0	0	0	0
		0	1	0	0	1	1
		1	0	0	1	1	0
		1	1	1	0	0	1

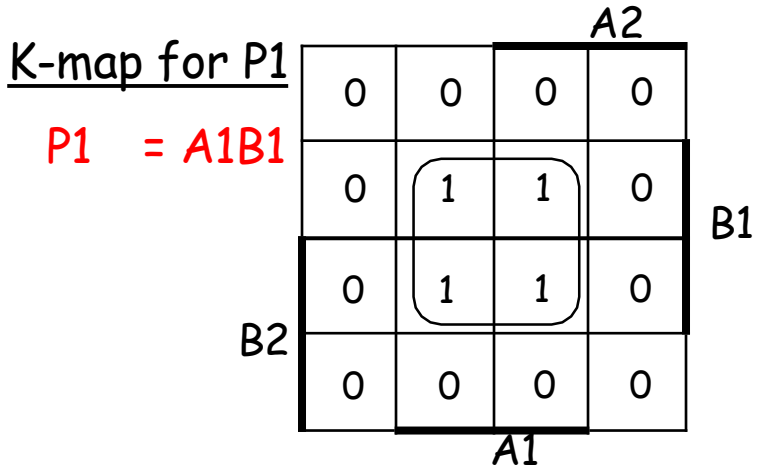
4-variable K-map for each
of the 4 output functions

Design example: 2x2-bit multiplier (cont'd)









Magnitude Comparator

- It is a combinational circuit that compares two numbers and determines their relative magnitude

- The output of comparator is usually 3 binary variables indicating: A>B

  A=B

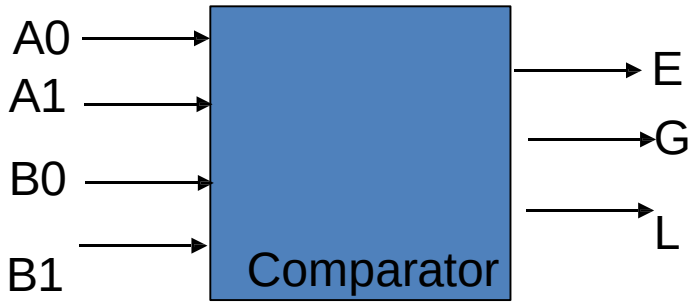
  A<B

- For example to design a comparator for 2 bit binary numbers A (A1A0) and B (B1B0) we do the following steps:

Comparators

- **For a 2-bit comparator we have four inputs A1A0 and B1B0 and three output E (is 1 if two numbers are equal), G (is 1 when A > B) and L (is 1 when A < B) If we use truth table and KMAP the result is**
- **E= A1'A0'B1'B0' + A1'A0B1'B0 + A1A0B1B0 + A1A'0B1B'0**

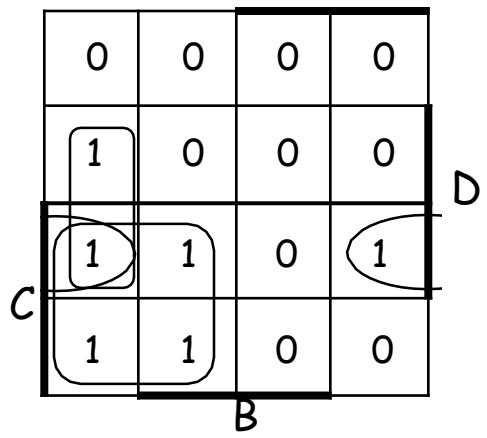
  or E=((A0 ⊕ B0) + (A1 ⊕ B1))' (see next slide)
- **G = A1B1' + A0B1'B0' + A1A0B0'**
- **L= A1'B1 + A1'A0'B0 + A0'B1B0**



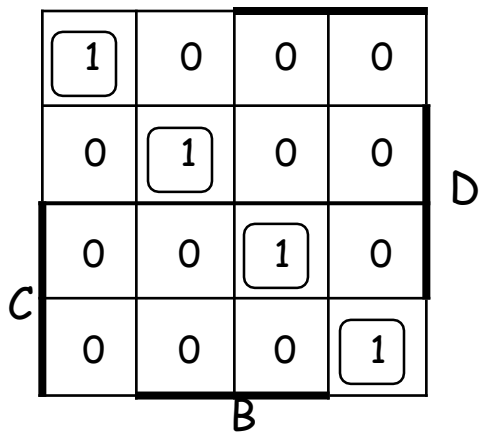
Truth Table for comparator

A0	A1	B0	B1	E	L	G
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1

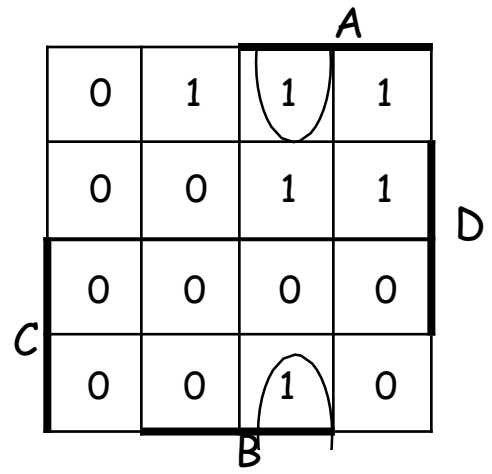
Design example: two-bit Comparator (cont'd)



K-map for LT



K-map for EQ



K-map for GT

LT = A' B' D + A' C + B' C D

EQ = A'B'C'D' + A'BC'D + ABCD + AB'CD' = (A xnor C) • (B xnor D)

GT = B C' D' + A C' + A B D'

Canonical Pof S vs minimal?

LT and GT are similar (flip A/C and B/D)

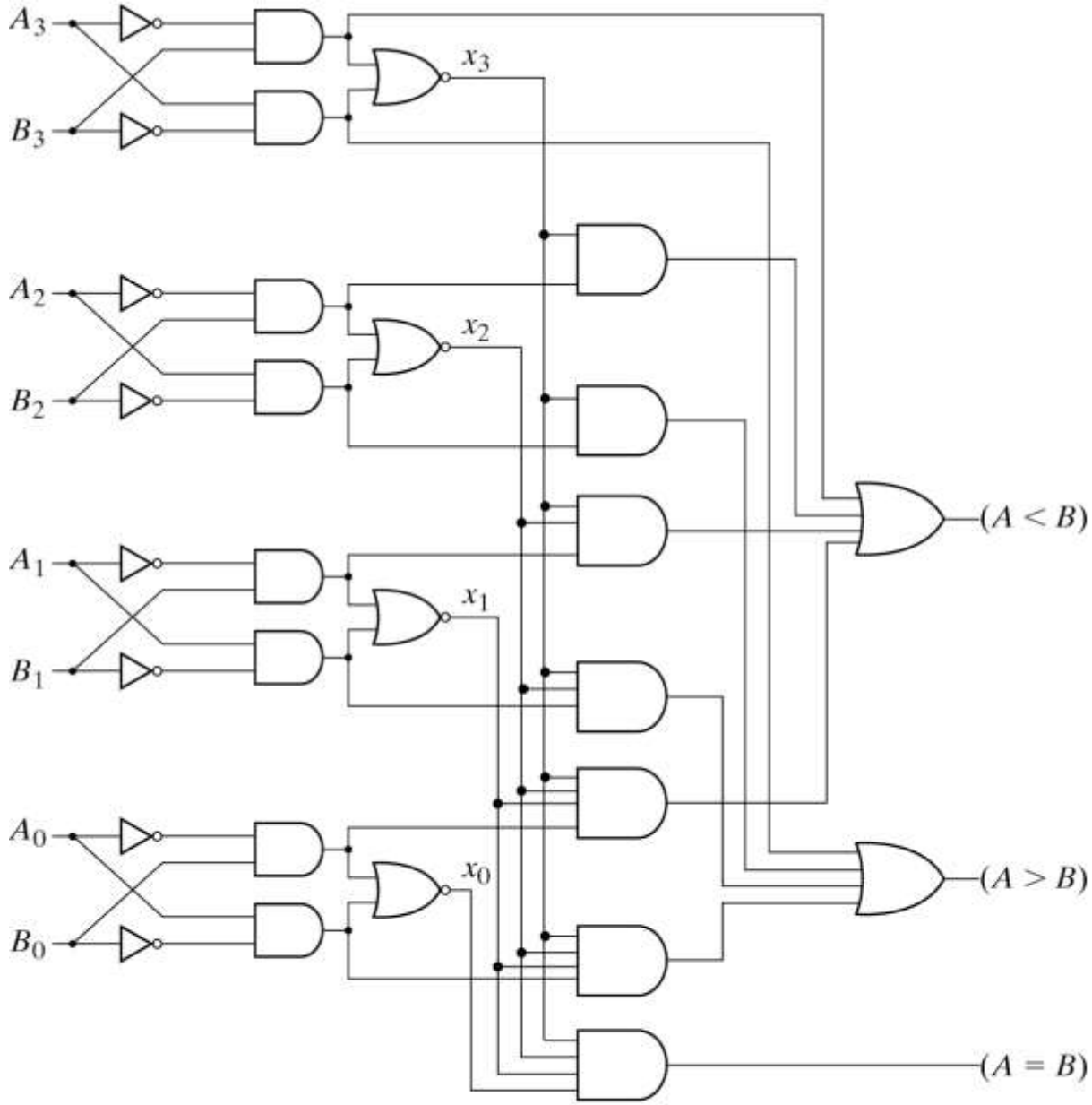
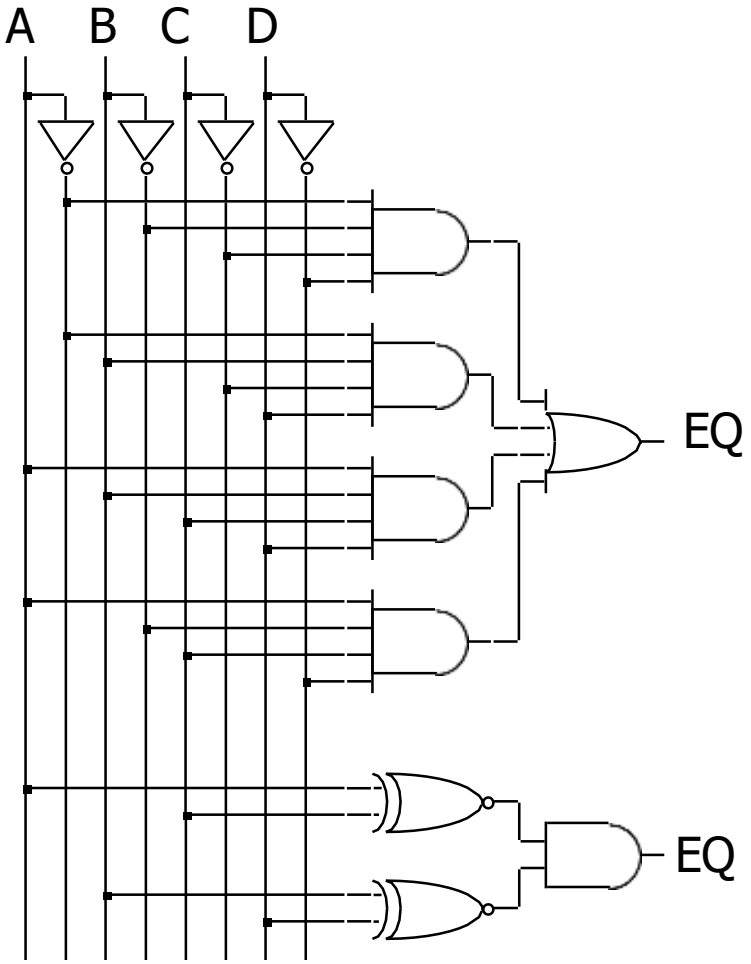
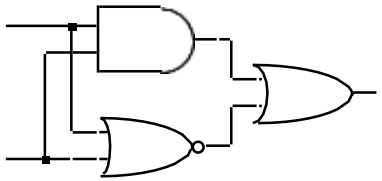


Fig. 4-17 4-Bit Magnitude Comparator

Design example: two-bit comparator (cont'd)



two alternative implementations of EQ with and without XOR



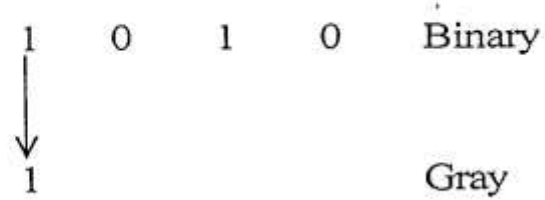
XNOR is implemented with at least 3 simple gates

Binary to Gray Code Converter

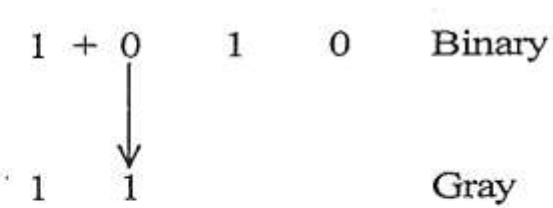
The Gray Code is unweighted and is not an arithmetic code: that is, there are no specific weights asssigned to the bit positions. The important feature of the Gray Code is that it exhibits only a single bit change from one code word to the next in sequence. This property is important in many applications, such as shaft position encoders, where error susceptibility increases with the number of bit changes between adjacent numbers in a sequence.

For instance - Let us convert the binary number 1010 to Gray Code.

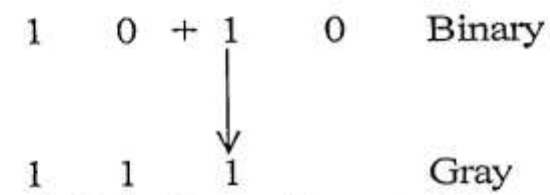
Step 1 - The left most Gray digit is the same as the left most binary



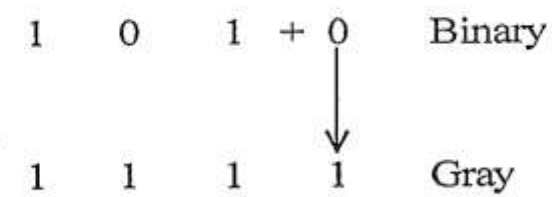
Step 2 - Add the left most binary digit to the adjacent one.



Step 3 - Add the next adjacent pair



Step 4 -Add the last adjacent pair



The conversion is now complete and the Gray Code is 1111.

Steps to design the converter

Design a converter by the following procedures:

a. Write down the truth table of both input and output bits of the converter.

b. Apply Karnaugh Map to look for the minimized logic expression for the output bits.

c. Then design the logical circuit.

Truth Table

Decimal	Binary				Gray			
	A	B	C	D	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

K-Map for each bit of Gray code

For practical consideration code conversions are made for each bit.

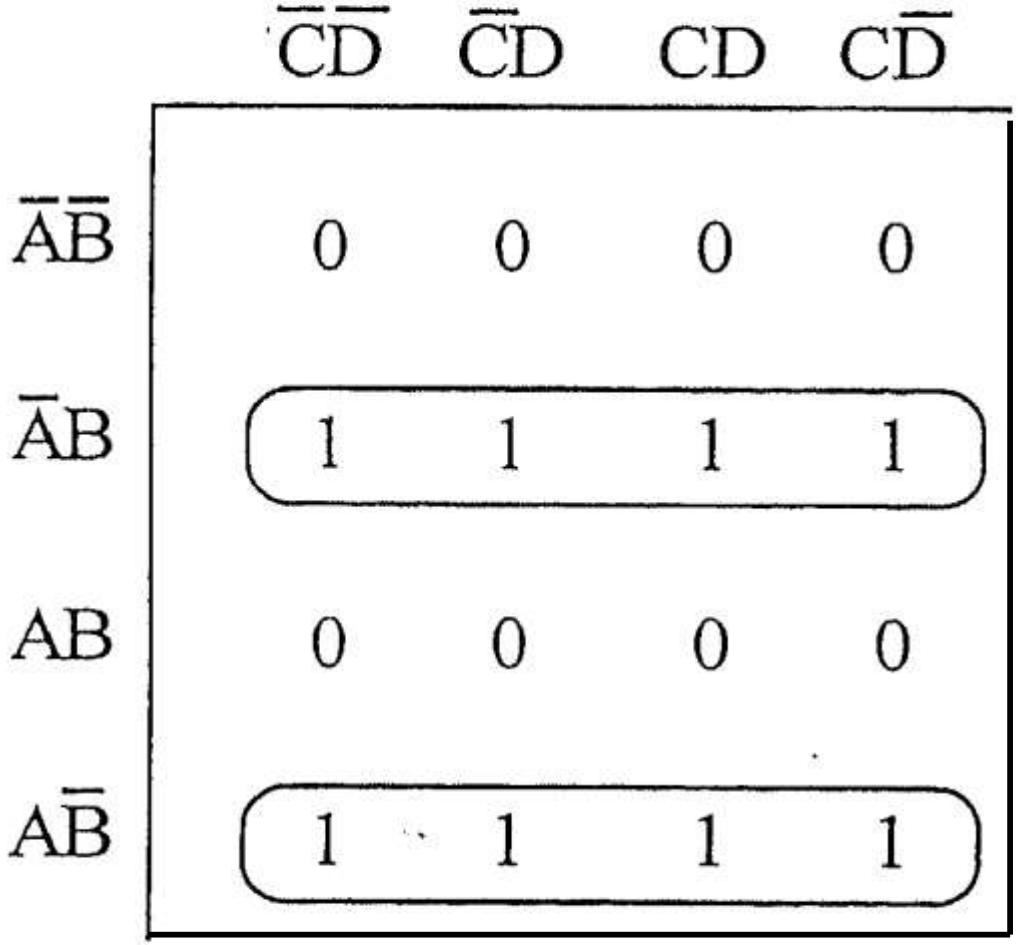
(a) For Y_3, the Karnaugh map can draw as follow.

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	0
$\overline{A}B$	0	0	0	0
AB	1	1	1	1
$A\overline{B}$	1	1	1	1

(a)

By minimization, $Y_3 = A$

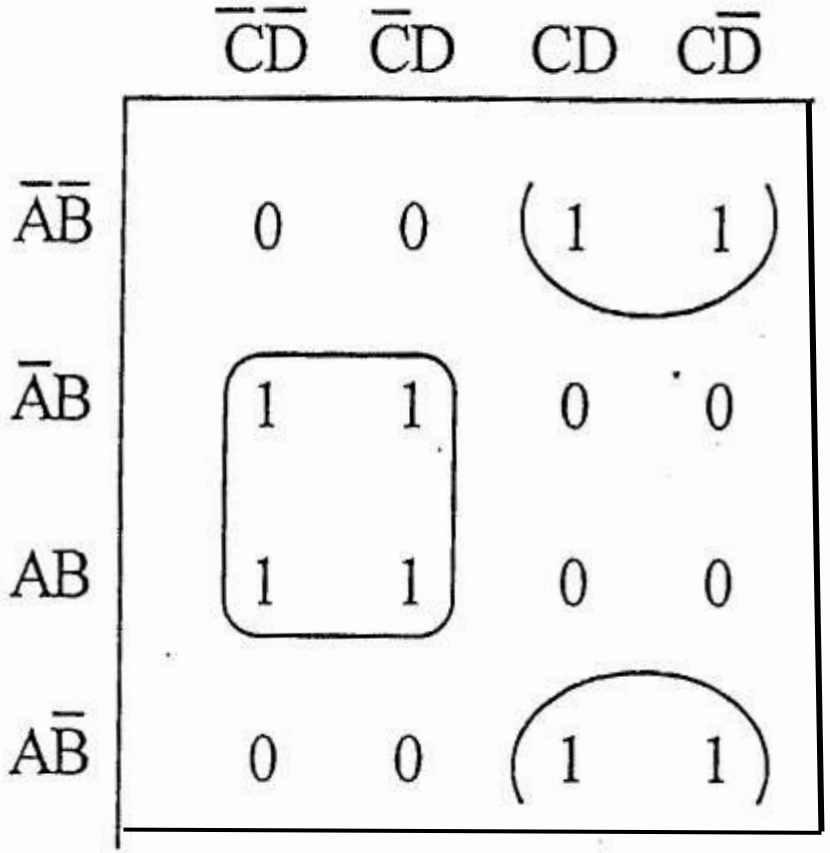
(b) For Y_2, the Karnaugh map can draw as follow.



(b)

By minimization, $Y_2 = A \oplus B$

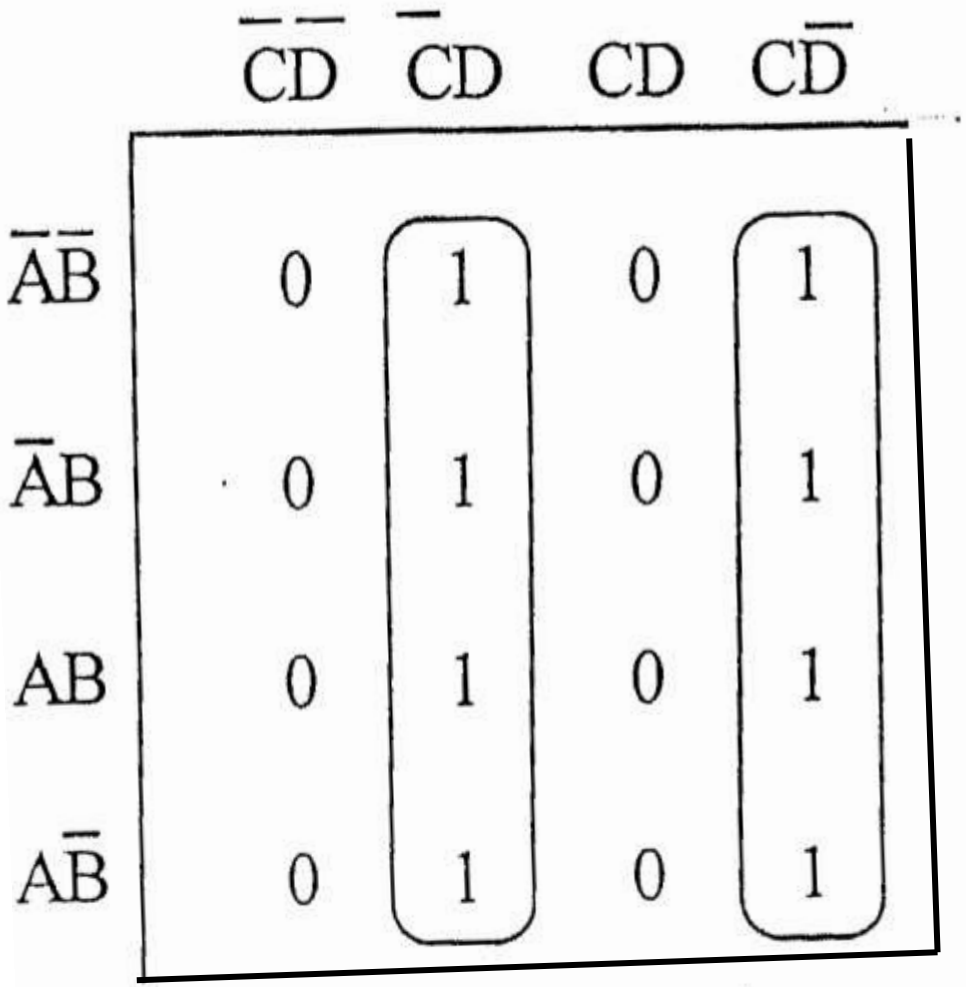
(c) For Y_1, the Karnaugh map can draw as follow.



(a)

By minimization, $Y_1 = B \oplus C$

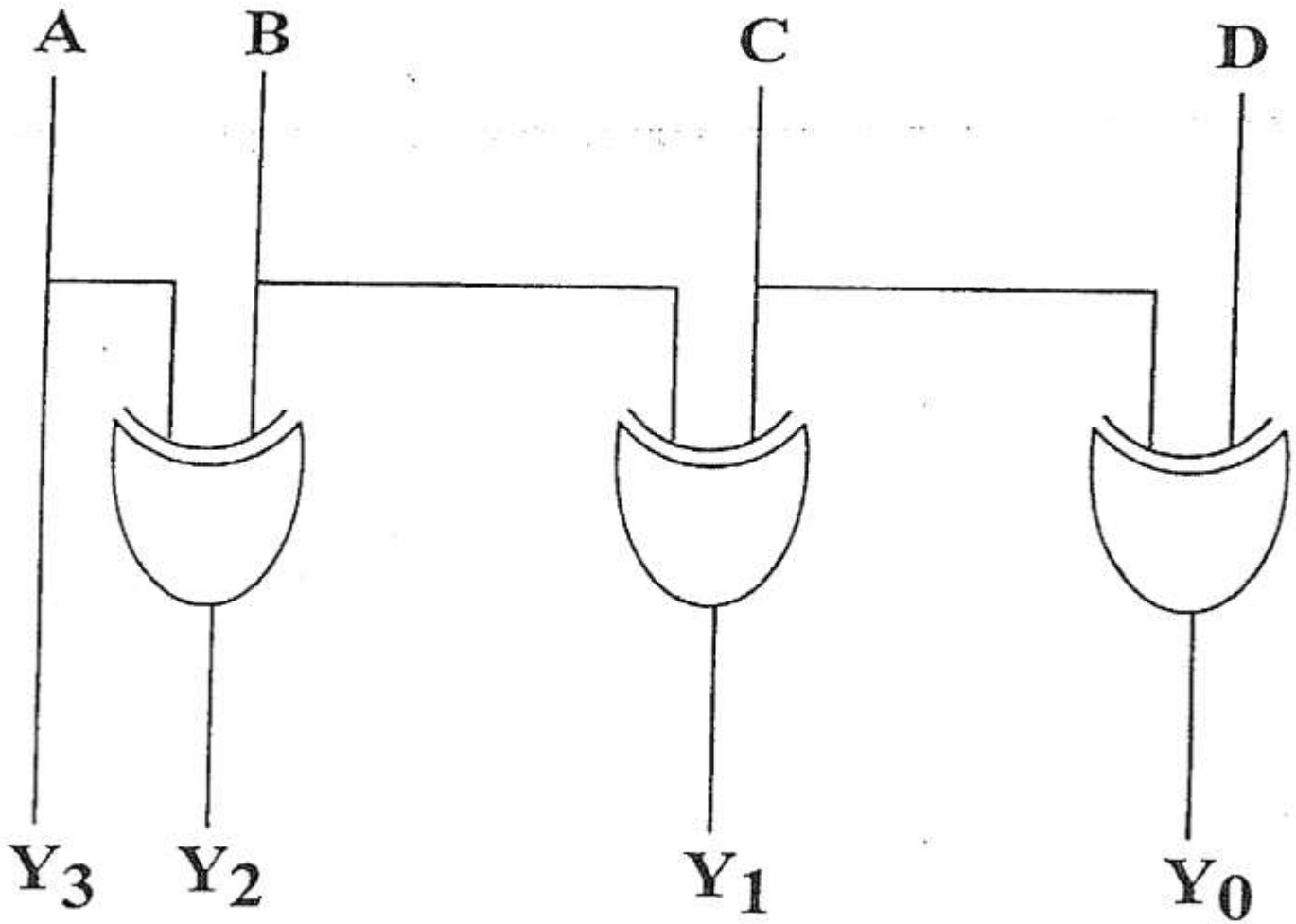
(d) For Y_0, the Karnaugh map can draw as follow.



(b)

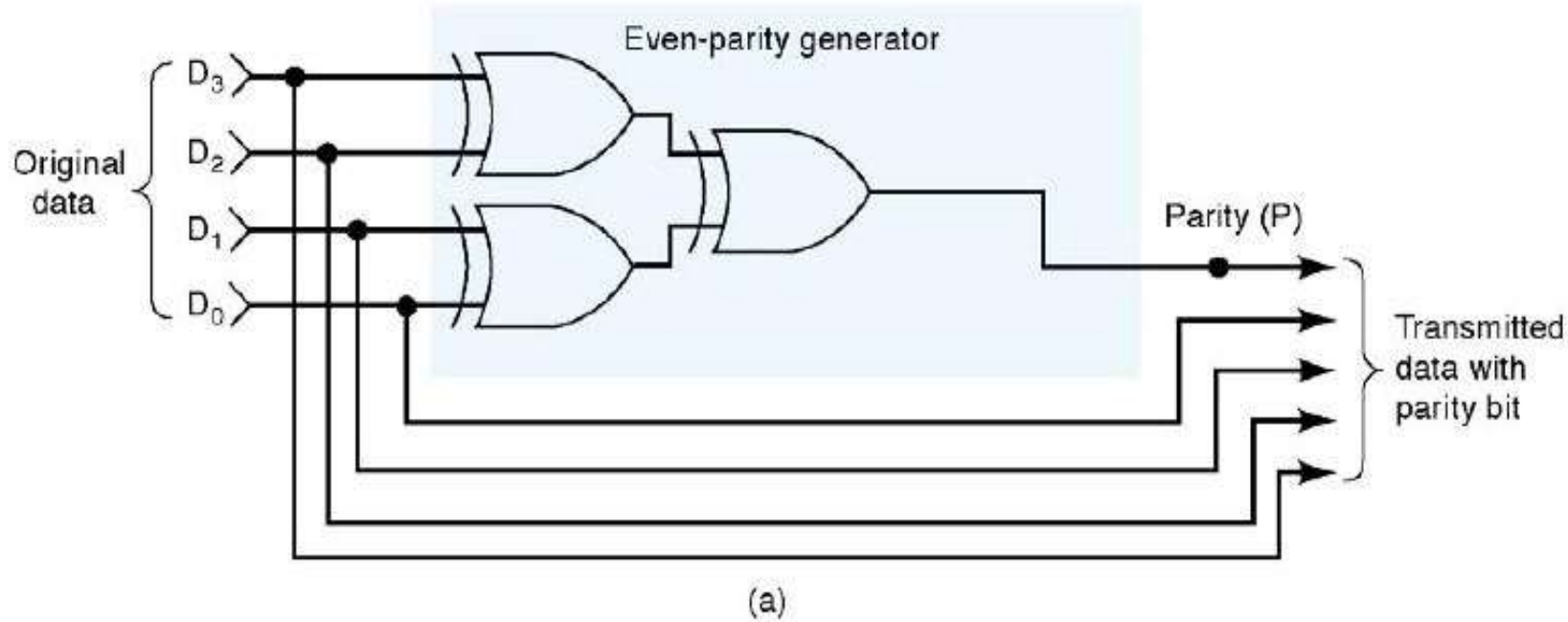
By minimization, $Y_0 = C \oplus D$

Binary Input

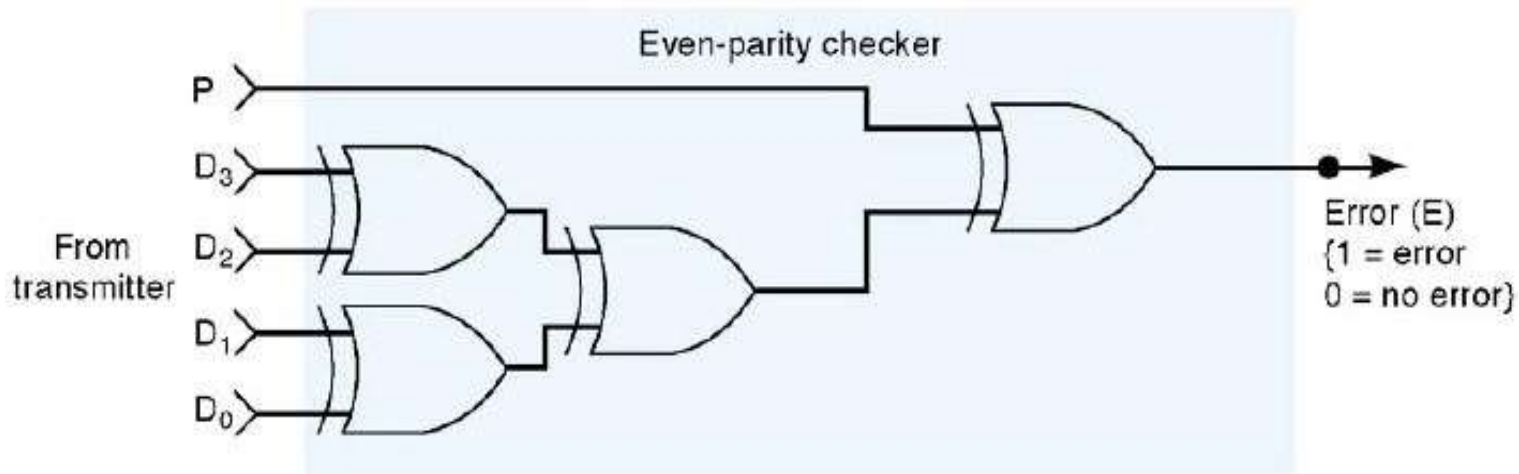


Gray Code Output

Parity Generator and Checker



(a)



(b)

- A transmitter can attach a parity bit to a set of data bits before transmitting the data bits to a receiver. The receiver will detect any single bit errors that may have occurred during the transmission.

- In figure (a) the set of data to be transmitted is applied to the parity-generator circuit, which produces the even-parity bit, P, at its output. This parity bit is transmitted to the receiver along with the original data bits, making a total of five bits.

- In figure (b) these five bits (data+parity) enter the receiver's parity-checker circuit, which produces an error output, E that indicates whether or not a single-bit error has occurred.

Mux/Demux

MULTIPLEXER (aka DATA SELECTOR)- circuit that can select one of a several inputs and pass the logic level of that input to the output.

DEMULTIPLEXER (aka DATA DISTRIBUTOR)- circuit that depending on the status of its select inputs will channel its data input to one of several outputs.

SELECT INPUTS (aka ADDRESS LINES)- used by the mux to determine which data inputs will be switched to the output.

$$if\ 2^N\ input\ lines = N\ select\ lines$$

A Multiplexer (MUX)

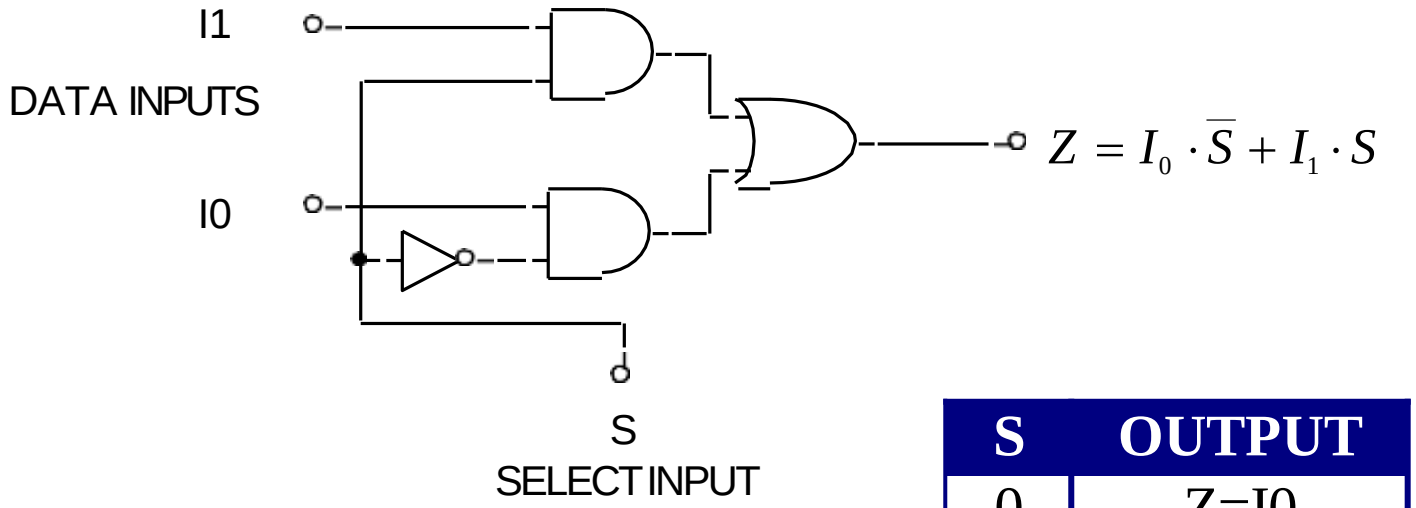
Consider an integer ‘m’, which is constrained by the following relation:

$\mathbf{m = 2^n}$, where m and n are both integers.

- **A m-to-1 Multiplexer** has
  - m Inputs: $I_0, I_1, I_2, \ldots\ldots\ldots\ldots I_{(m-1)}$
  - one Output: Y
  - n Control inputs: $S_0, S_1, S_2, \ldots\ldots S_{(n-1)}$
  - One (or more) Enable input(s)

such that Y may be equal to one of the inputs, depending upon the control inputs.

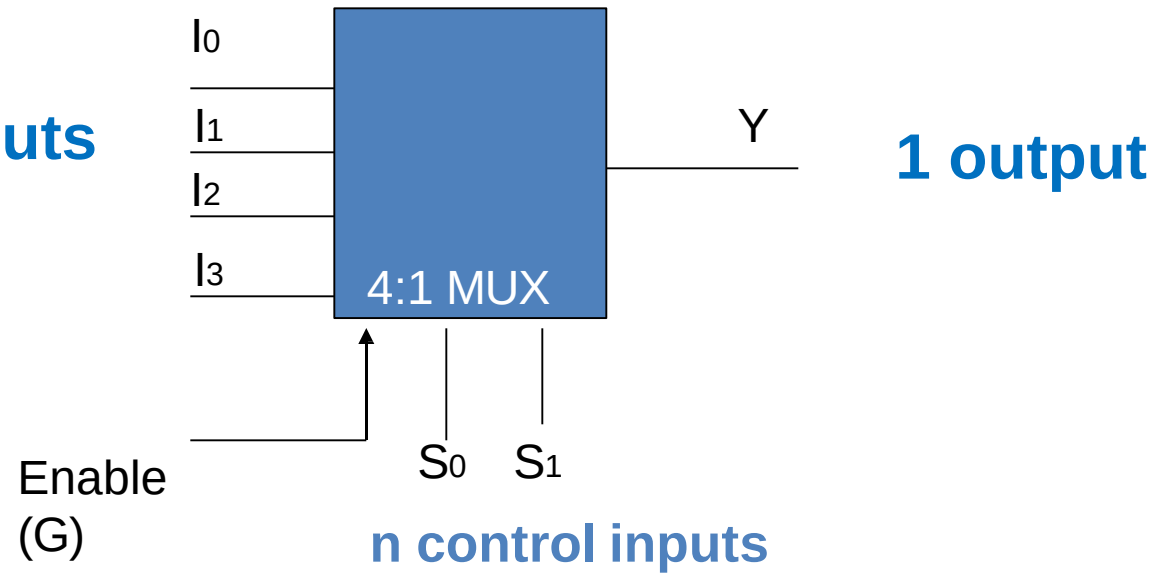
BASIC TWO-INPUT MULTIPLEXER



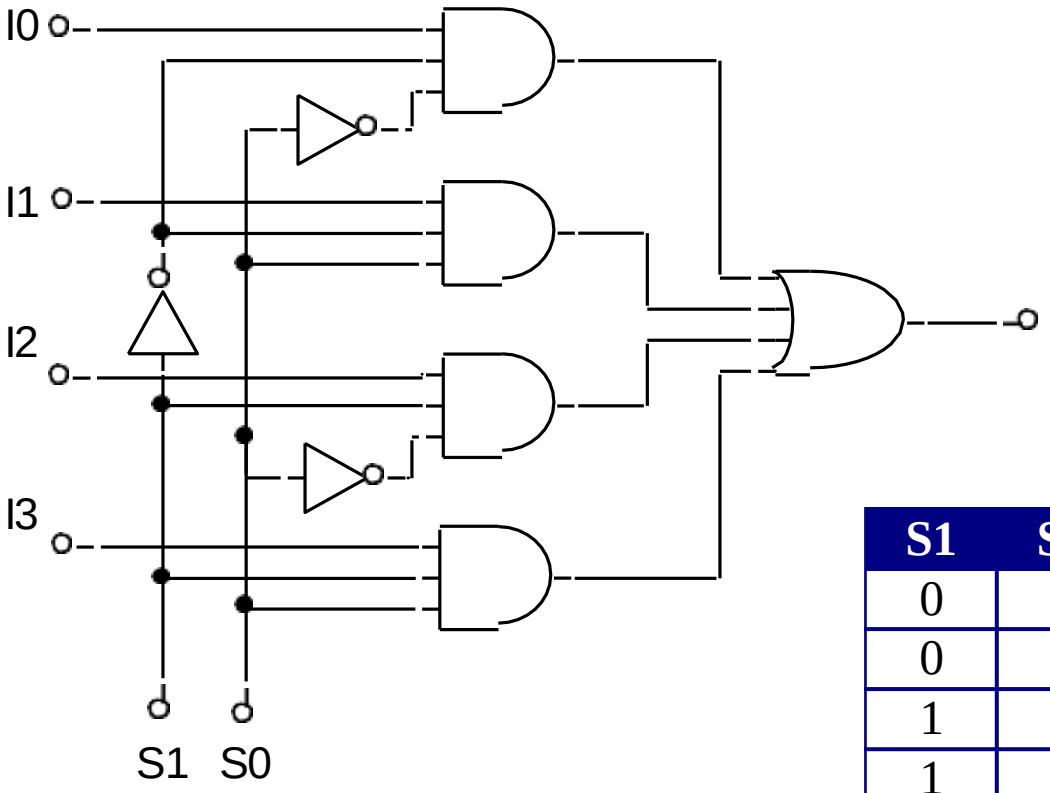
S	OUTPUT
0	Z=I0
1	Z=I1

Example: A 4-to-1 Multiplexer

A 4-to-1 Multiplexer:



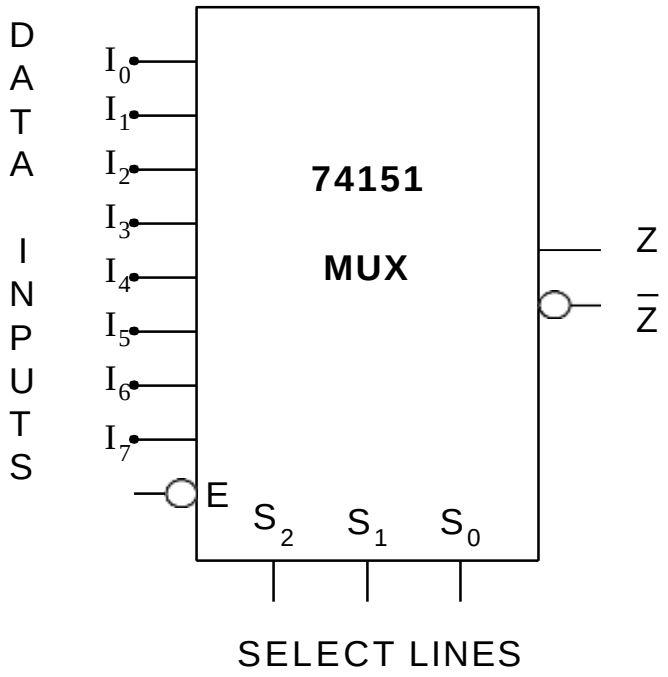
FOUR-INPUT MULTIPLEXER



S1	S0	OUTPUT
0	0	Z=I0
0	1	Z=I1
1	0	Z=I2
1	1	Z=I3

MULTIPLEXER LOGIC DIAGRAM

- Takes one of many inputs and funnels it to an output Z.
- Take the selector lines convert to a decimal number and this is the input funneled to the output.
- Strobe is active low enable

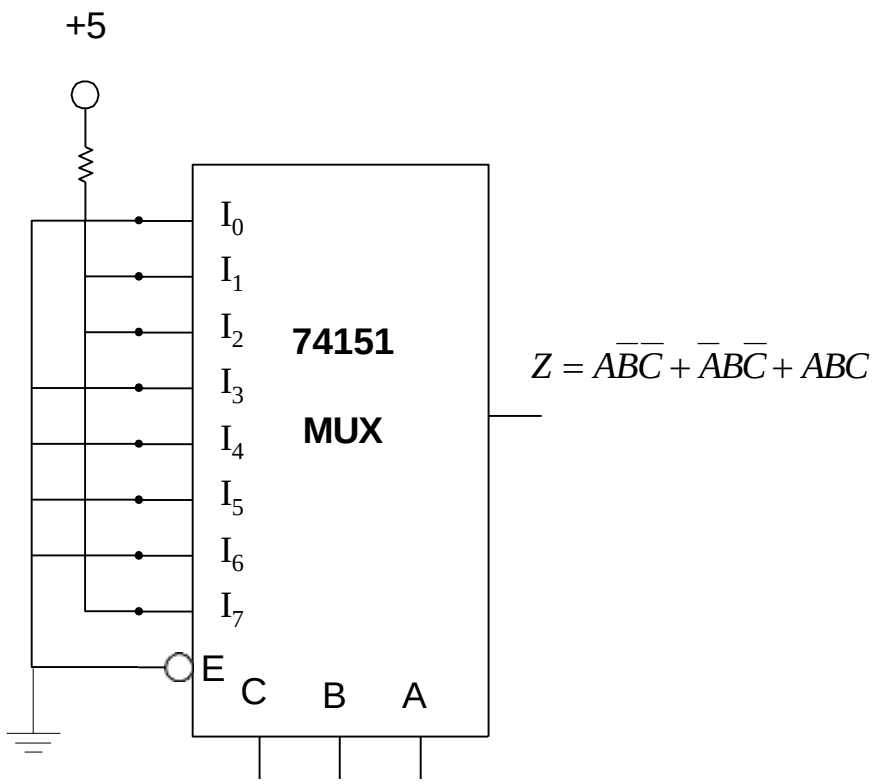


S2	S1	S0	E	Z
0	0	0	0	I0
0	0	1	0	I1
0	1	0	0	I2
0	1	1	0	I3
1	0	0	0	I4
1	0	1	0	I5
1	1	0	0	I6
1	1	1	0	I7

MULTIPLEXER APPLICATIONS

- Data routing
- Parallel-to-serial conversion
- Operation sequencing
- Implement logic function of a truth table

LOGIC FUNCTION GENERATION



$$Z = A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$$

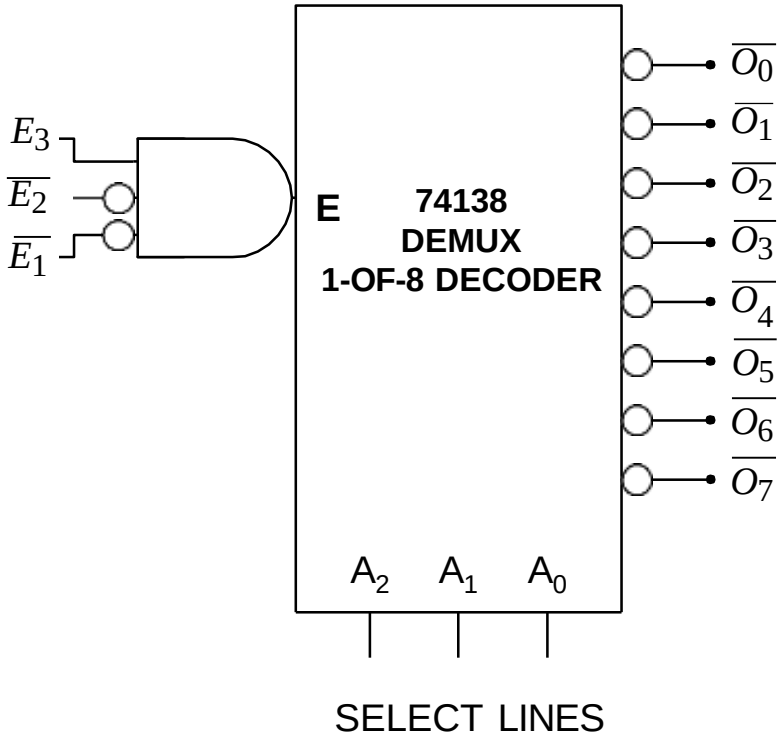
C	B	A	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

DEMULTIPLEXER

DEMULTIPLEXER LOGIC DIAGRAM

- **Logic circuit that depending on the status of its select inputs will funnel its data input to one of several data outputs.**

- **Separate enable inputs (useful for cascading decoders) into AND gate which must be high to enable the decoder outputs.**

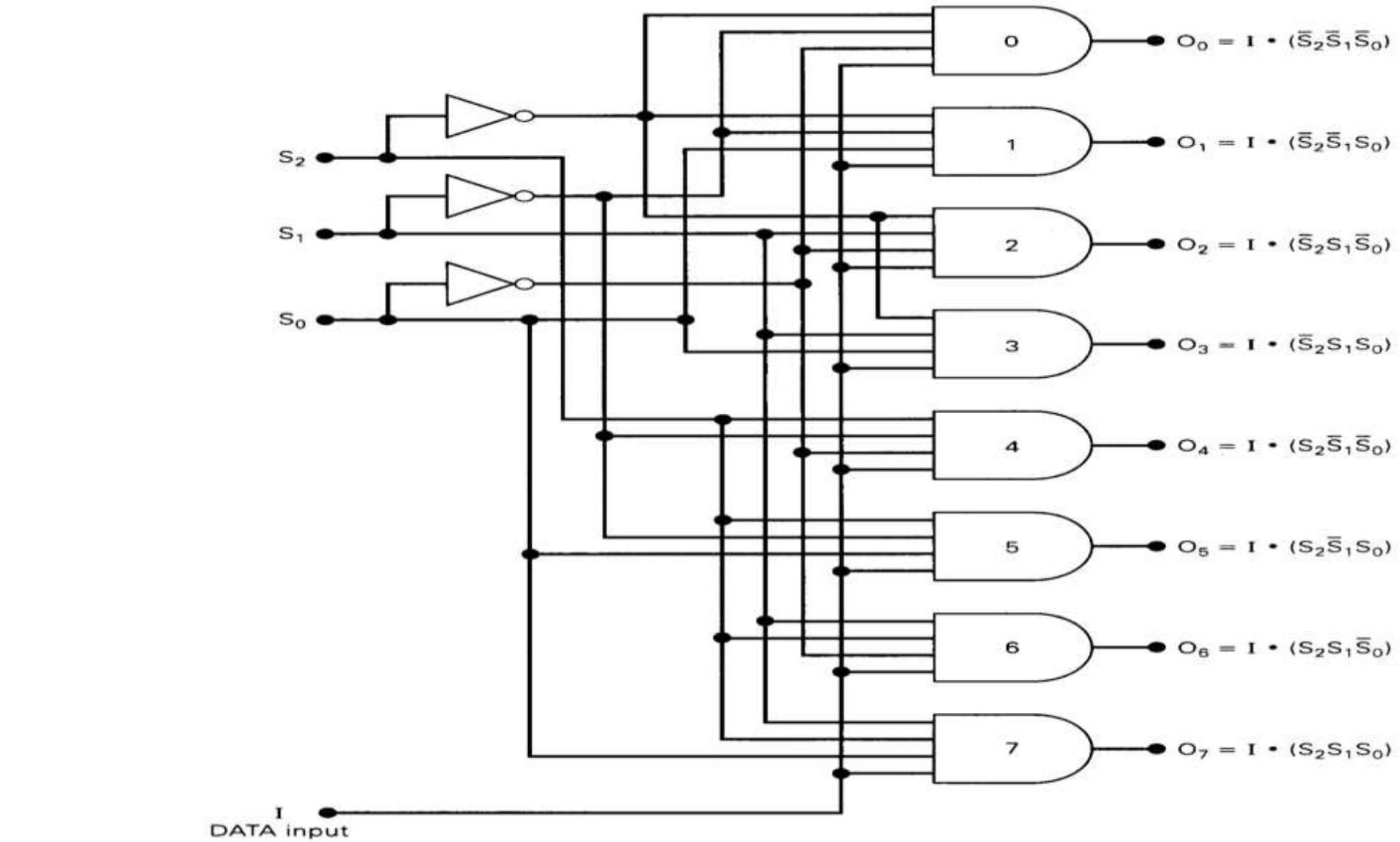


$\overline{E_1}$	$\overline{E_2}$	E_3	*OUTPUTS*
0	0	1	RESPOND TO INPUT CODE A2A1A0
1	X	X	DISABLED –ALL HIGH
X	1	X	DISABLED –ALL HIGH
X	X	0	DISABLED –ALL HIGH

DEMULTIPLEXER

SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

DEMULTIPLEXER



DECODER

- A decoder is a logic circuit that accepts a set of inputs that represents a binary number and activates only the output that corresponds to the input number.

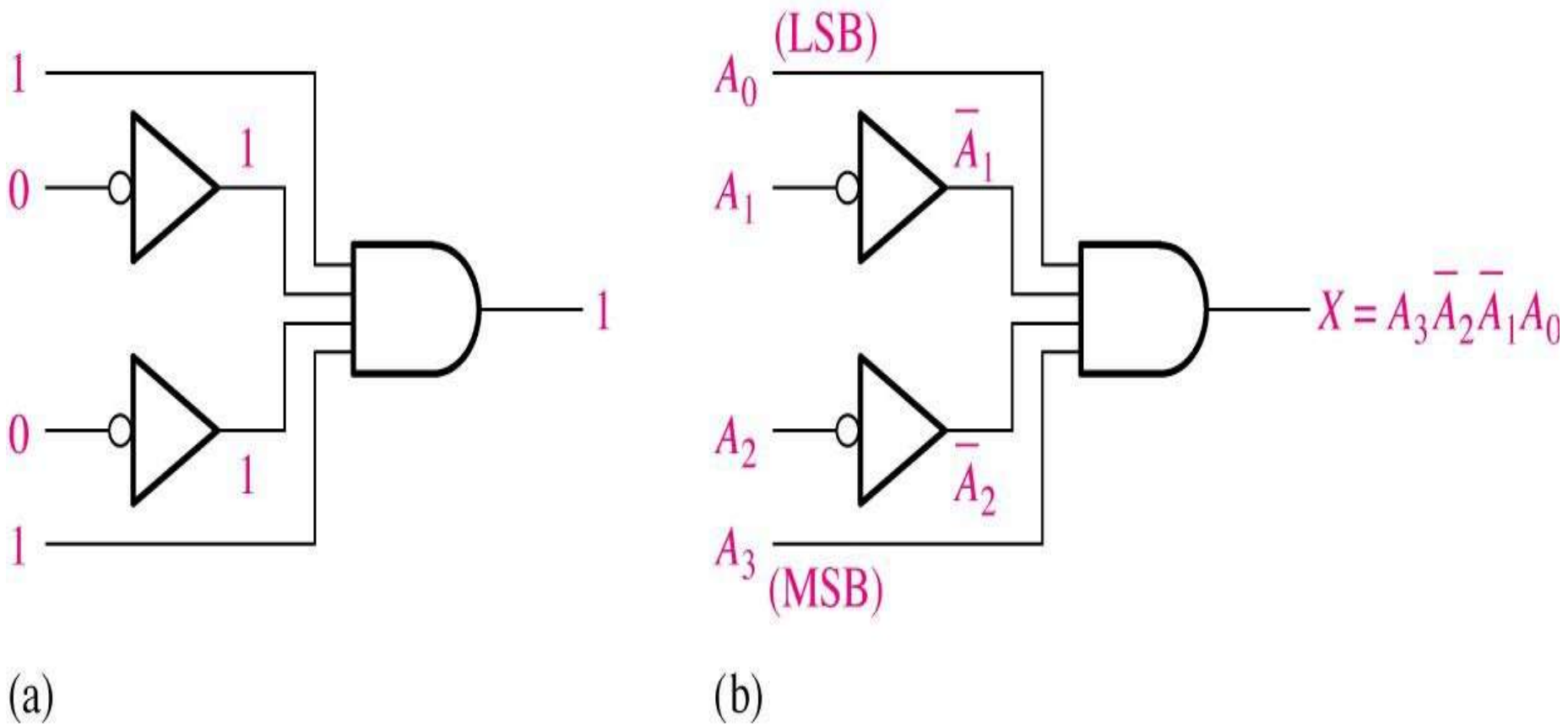
- In other words, a decoder circuit looks at its inputs, determines which binary number is present there, and activates the one output that corresponds to that number ; all other outputs remain inactive

In its general form, a decoder has N input lines to handle N bits and form one to 2^N output lines to indicate the presence of one or more N-bit combinations.

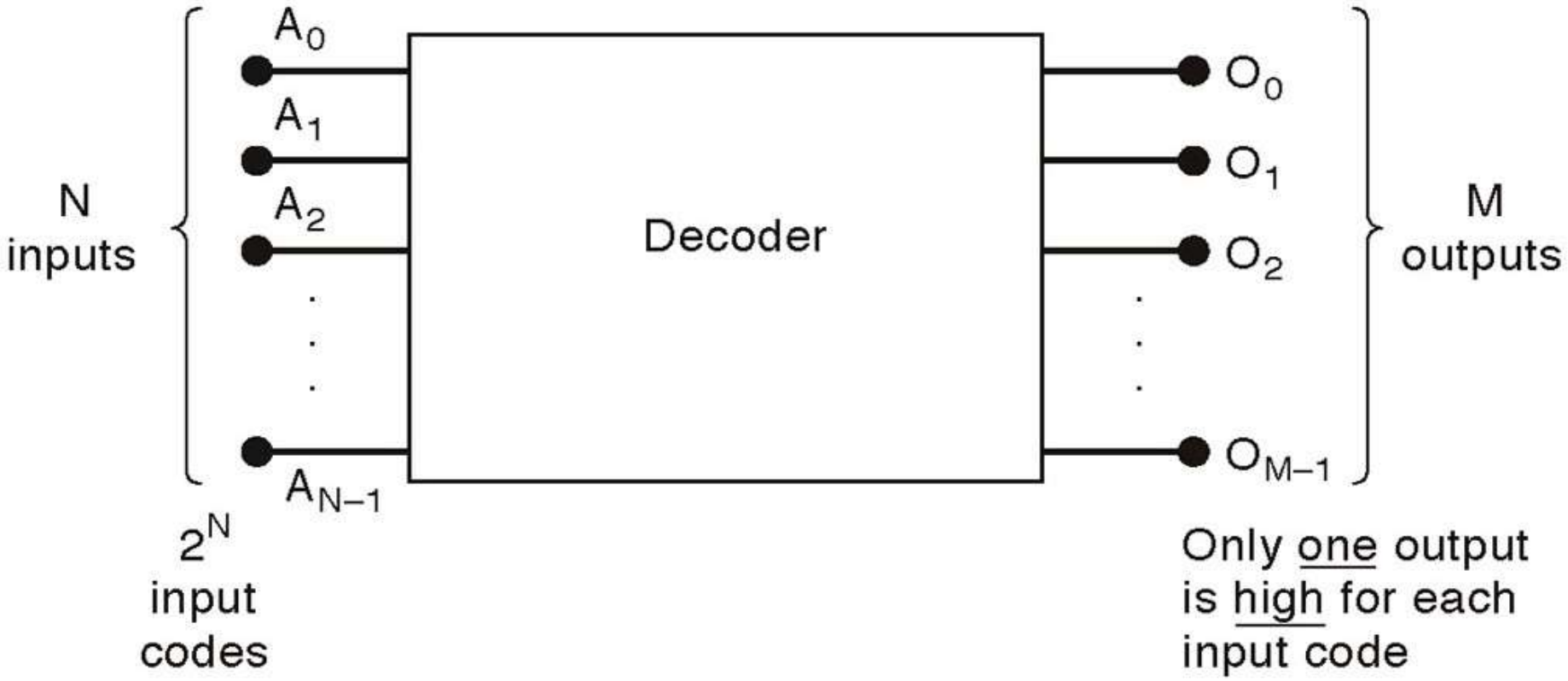
The basic binary function

- An AND gate can be used as the basic decoding element because it produces a HIGH output only when all inputs are HIGH

Decoding logic for the binary code 1001 with an active-HIGH output.



General decoder diagram



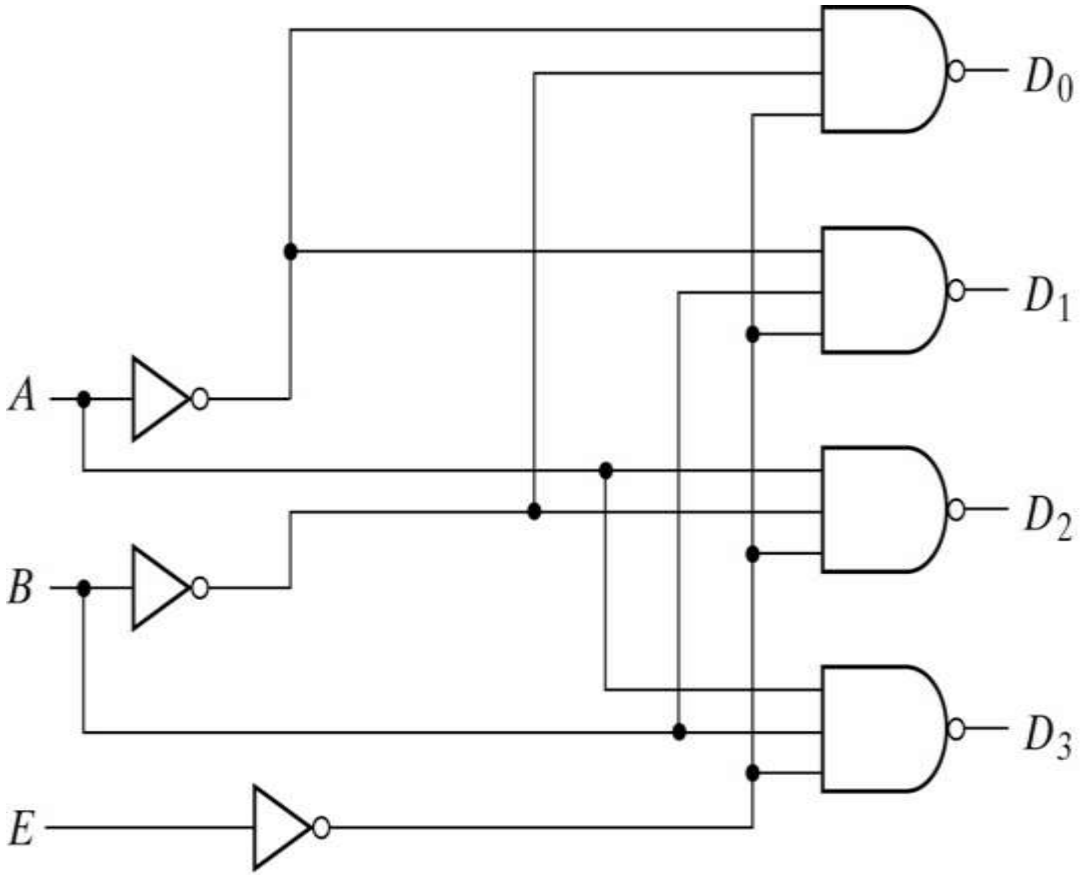
There are 2^N possible input combinations, from A_0 to A_{N-1}.

For each of these input combinations only one of the *M* outputs will be active *HIGH* (1), all the other outputs are *LOW* (0).

- If an active-LOW output (74138, one of the output will low and the rest will be high) is required for each decoded number, the entire decoder can be implemented with
    1. NAND gates
    2. Inverters

- If an active-HIGH output (74139, one of the output will high and the rest will be low) is required for each decoded number, the entire decoder can be implemented with
    1. AND gates
    2. Inverters

2-to-4-Line Decoder
(with Enable input)-Active LOW output (1)...



(a) Logic diagram

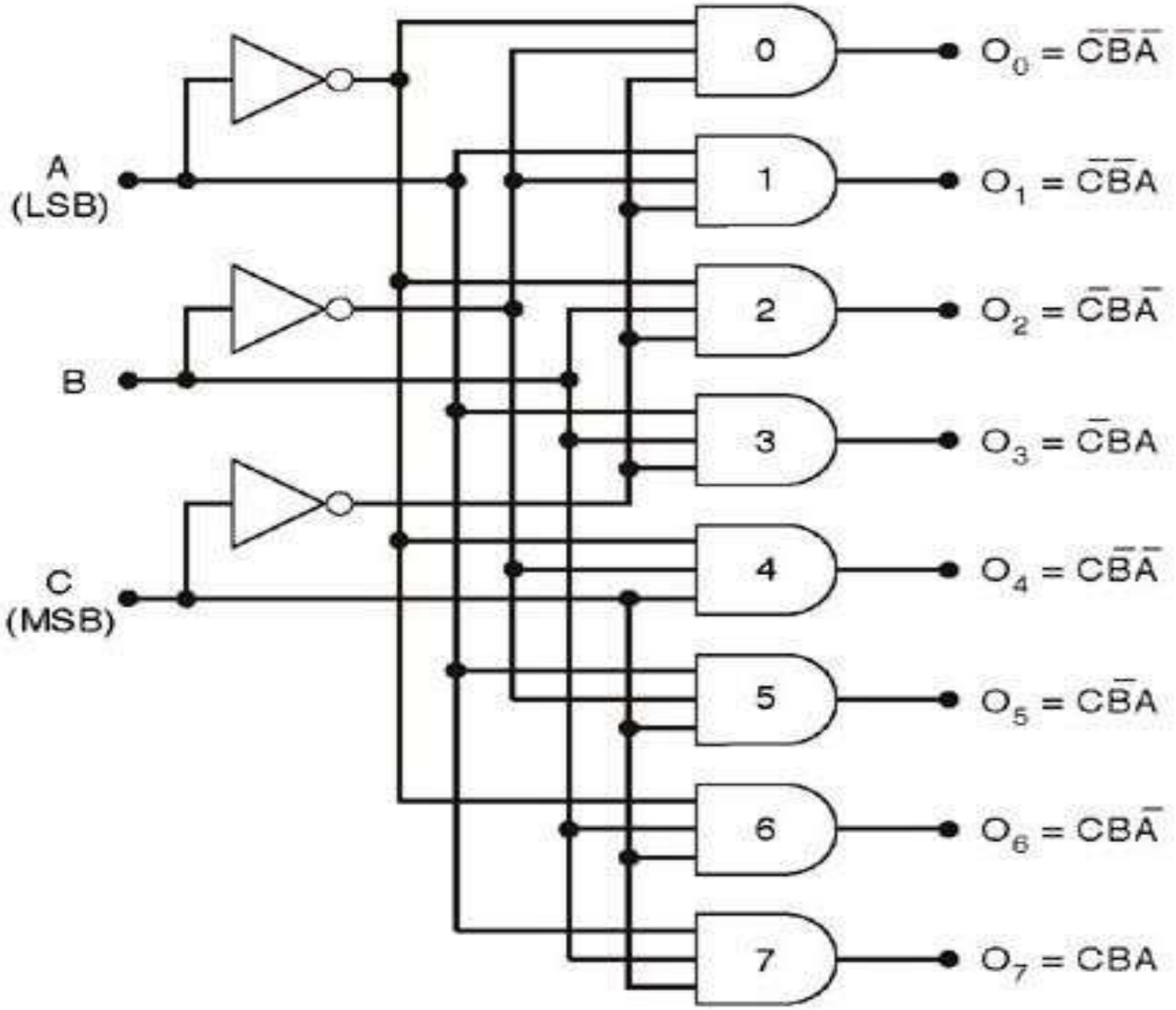
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

2-to-4-Line Decoder (with Enable input)-Active LOW output (2)

- The circuit operates with complemented outputs and a complement enable input. The decoder is enabled when E is equal to 0.
- Only one output can be equal to 0 at any given time, all other outputs are equal to 1.
- The output whose value is equal to 0 represents the minterm selected by inputs A and B
- The circuit is disabled when E is equal to 1.

3-8 line decoder (active-HIGH)

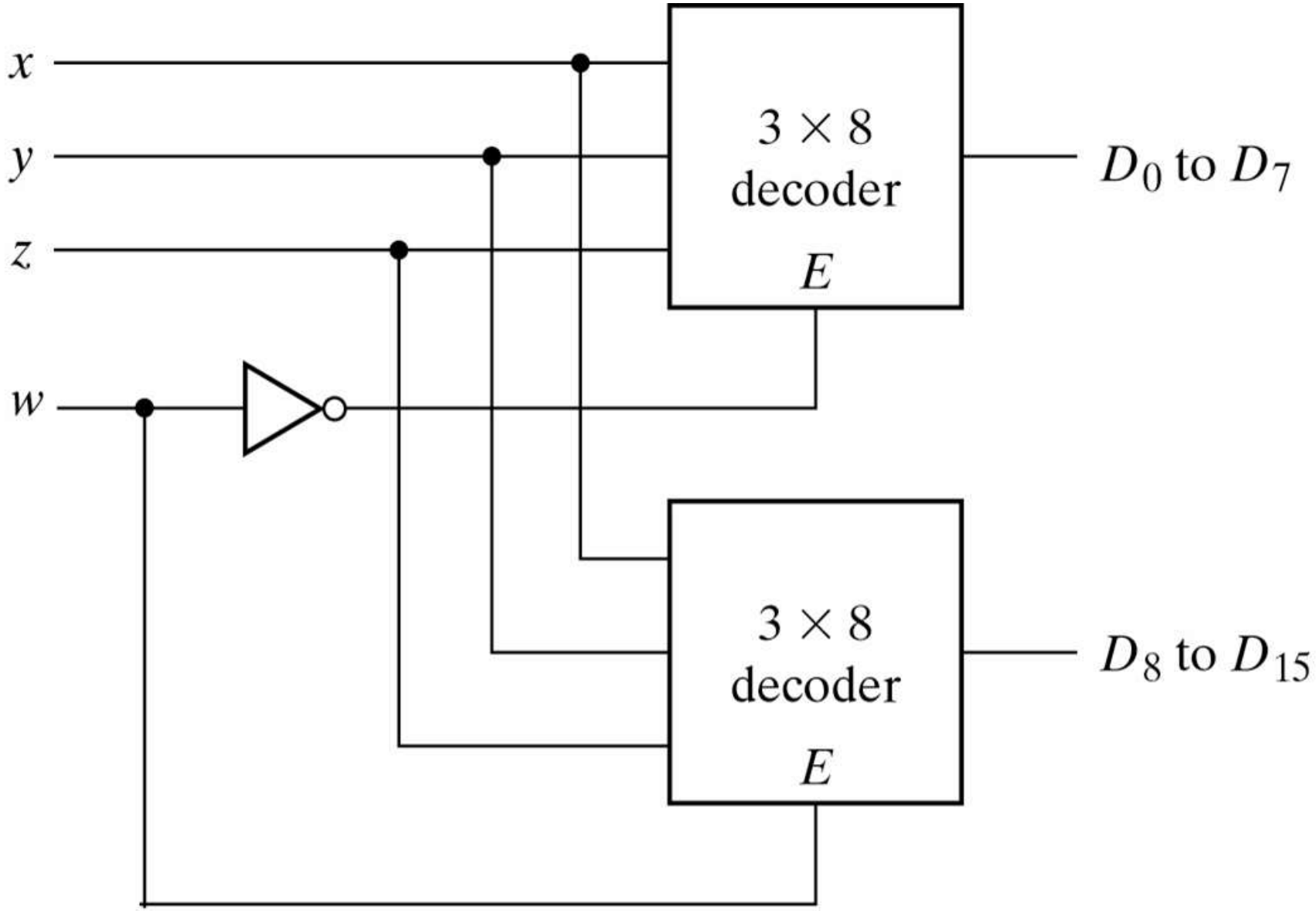


C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

- This decoder can be referred to in several ways. It can be called a 3-line-to-8-line decoder, because it has three input lines and eight output lines.

- It could also be called a binary-octal decoder or converters because it takes a three bit binary input code and activates the one of the eight outputs corresponding to that code. It is also referred to as a 1-of-8 decoder, because only 1 of the 8 outputs is activated at one time.

4-line-to-16 line Decoder constructed with two 3-line-to-8 line decoders (1)...

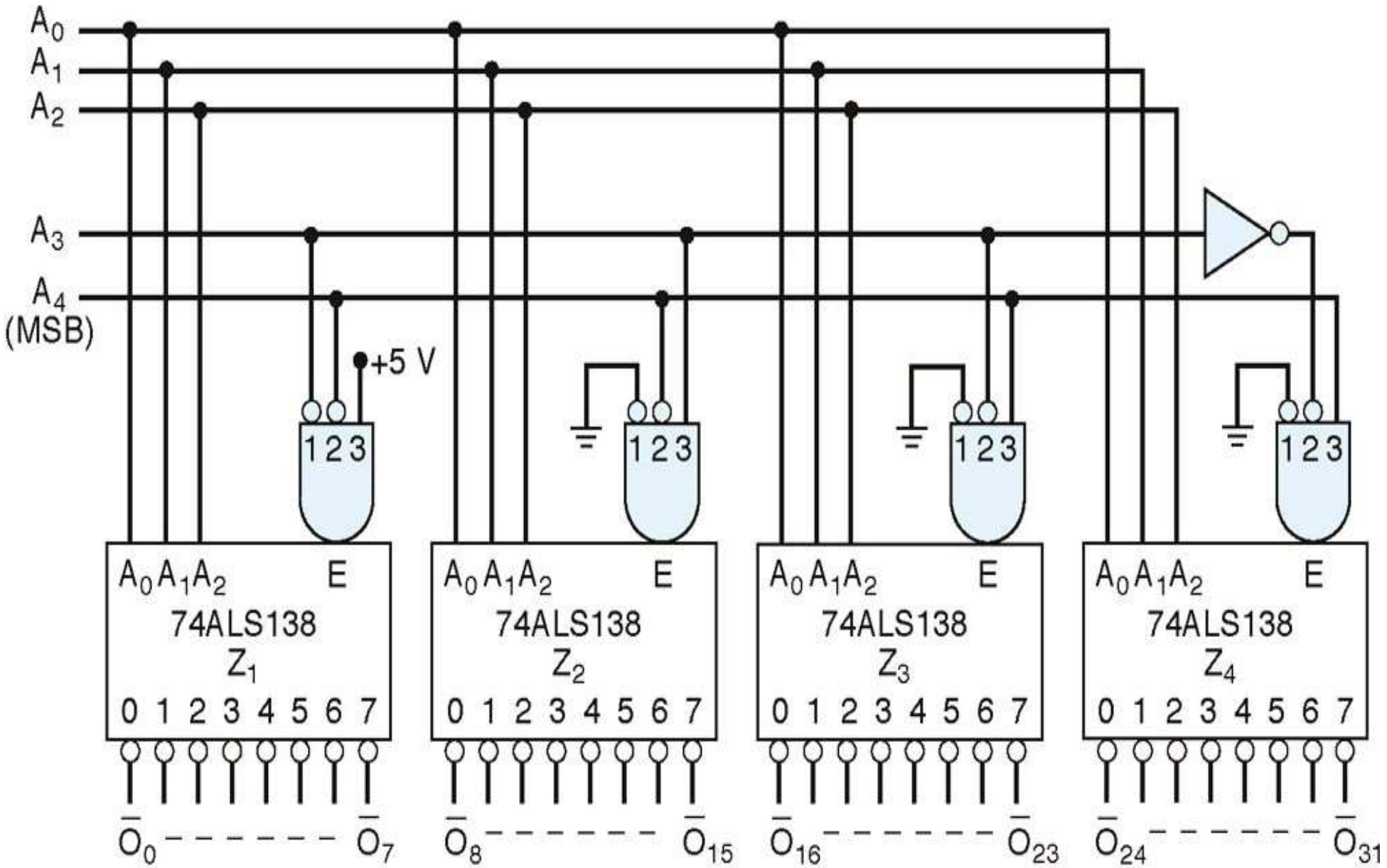


4-line-to-16 line Decoder constructed with two 3-line-to-8 line decoders (2)

- When w=0, the top decoder is enabled and the other is disabled. The bottom decoder outputs are all 0's , and the top eight outputs generate min-terms 0000 to 0111.

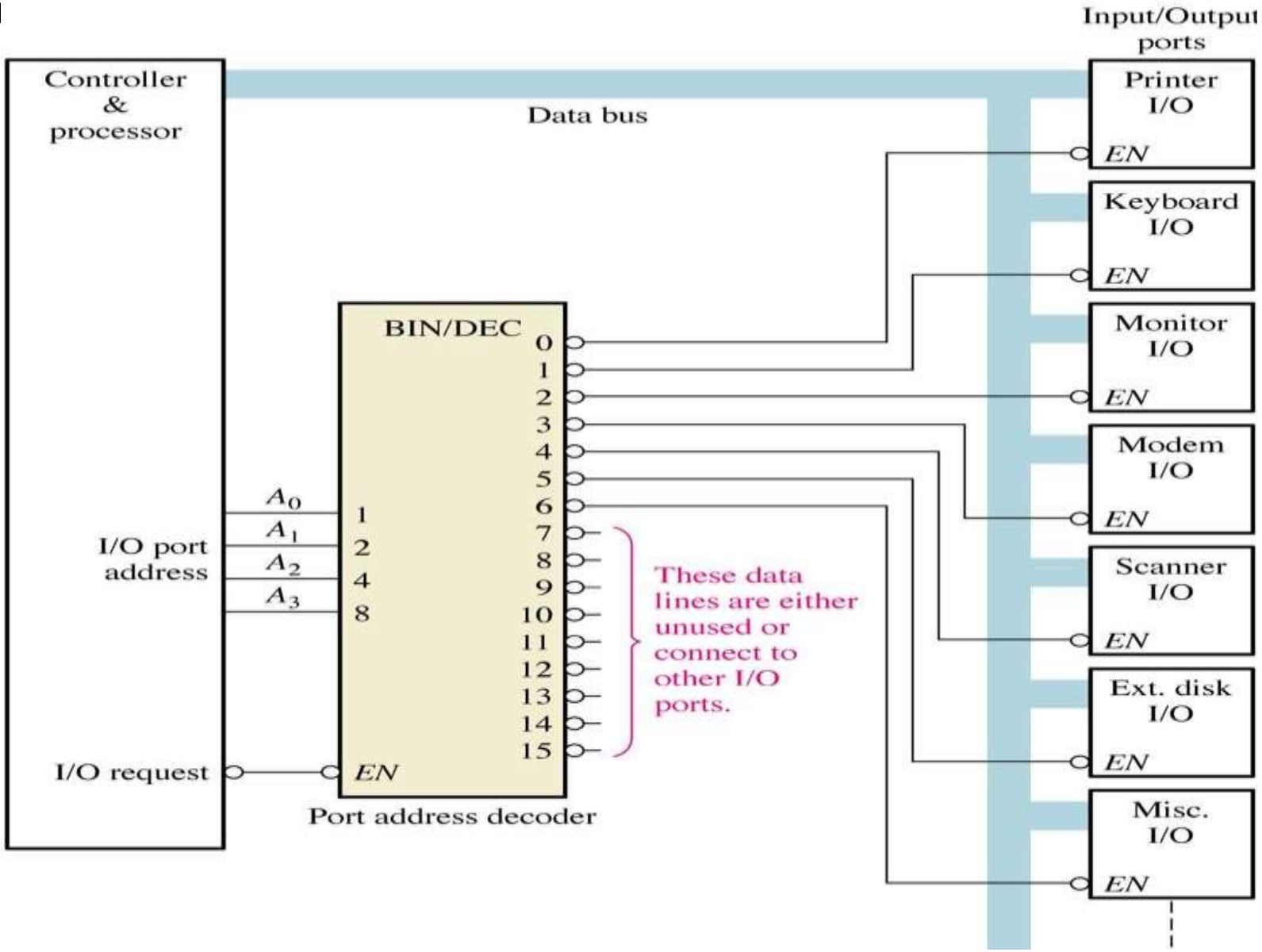
- When w=1, the enable conditions are reversed. The bottom decoder outputs generate min- terms 1000 to 1111, while the outputs of the top decoder are all 0's.

Example of a 5 to 32 Bit Decoder



Application example

A simplified computer I/O port system with a port address decoder with only four address lines shown



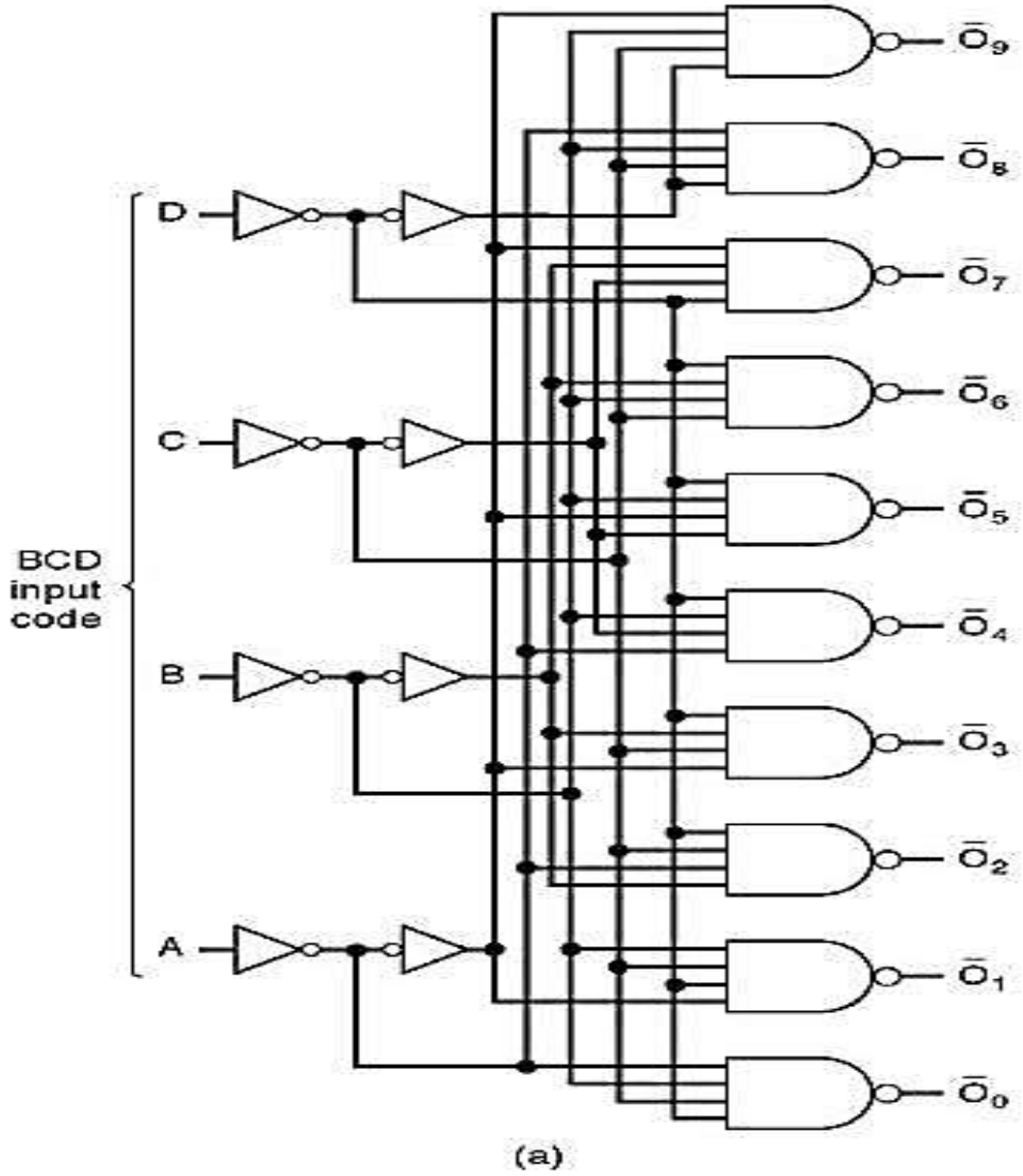
BCD -to- Decimal decoders

- The BCD-to-decimal decoder converts each BCD code into one of Ten Positional decimal digit indications. It is frequently referred as a 4-line -to- 10 line decoder.

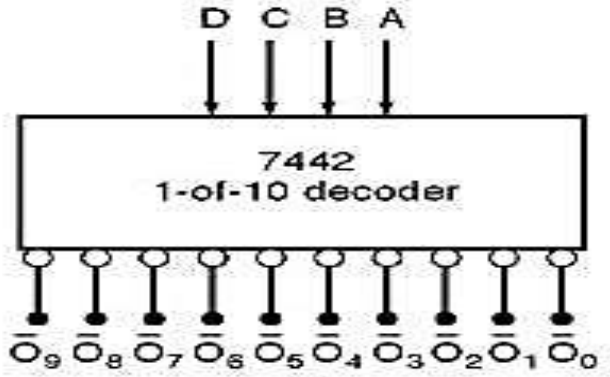
- The method of implementation is that only ten decoding gates are required because the BCD code represents only the ten decimal digits 0 through 9.

- Each of these decoding functions is implemented with NAND gates to provide active -LOW outputs. If an active HIGH output is required, AND gates are used for decoding

Logic diagram of BCD – decimal decoder (Active LOW output)



(a)

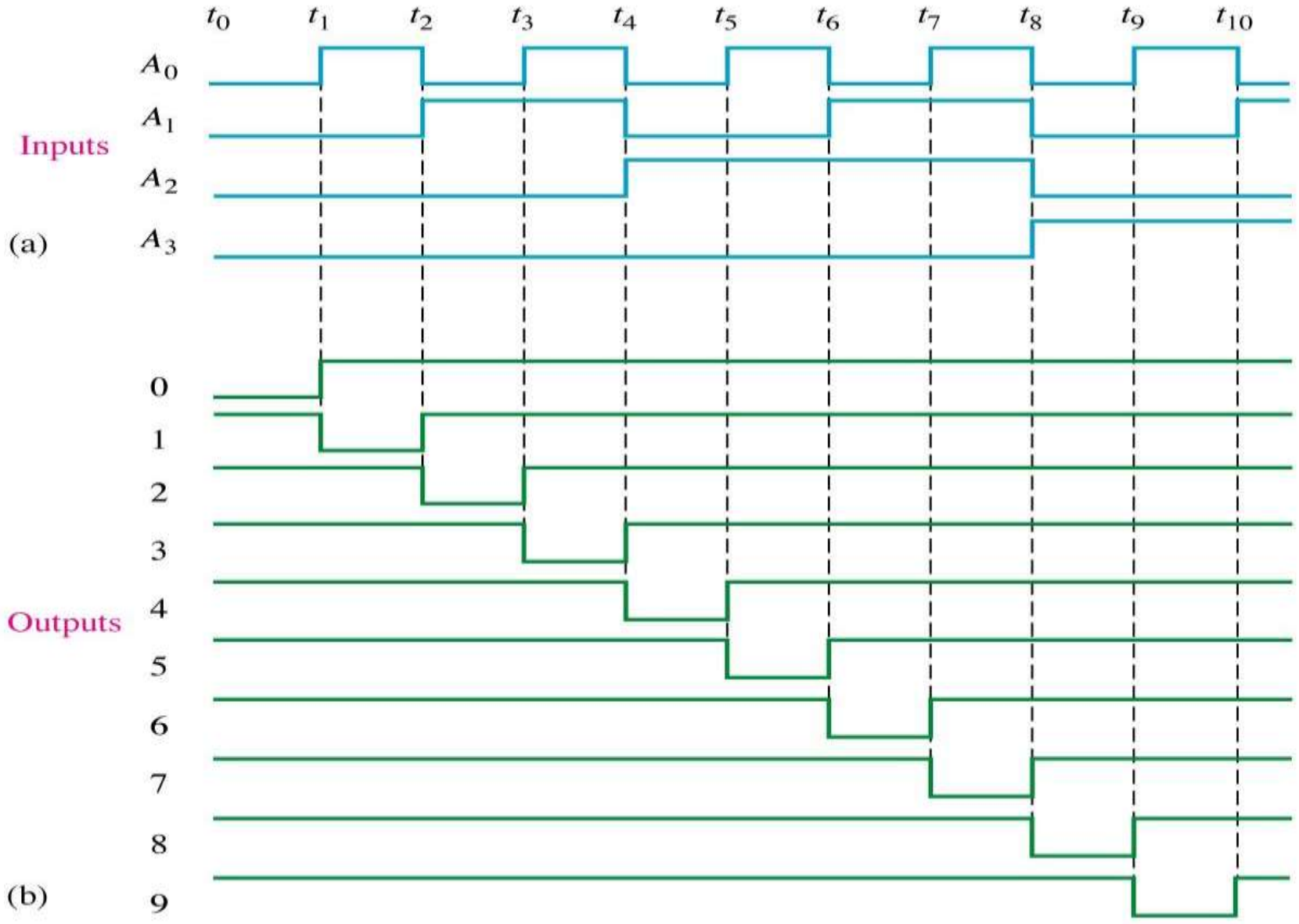


(b)

Inputs				
D	C	B	A	Active Output
L	L	L	L	$\bar{O}_0$
L	L	L	H	$\bar{O}_1$
L	L	H	L	$\bar{O}_2$
L	L	H	H	$\bar{O}_3$
L	H	L	L	$\bar{O}_4$
L	H	L	H	$\bar{O}_5$
L	H	H	L	$\bar{O}_6$
L	H	H	H	$\bar{O}_7$
H	L	L	L	$\bar{O}_8$
H	L	L	H	$\bar{O}_9$
H	L	H	L	None
H	L	H	H	None
H	H	L	L	None
H	H	L	H	None
H	H	H	L	None
H	H	H	H	None

H = HIGH Voltage Level
L = LOW Voltage Level

(c)



Output Waveform for BCD Decoder

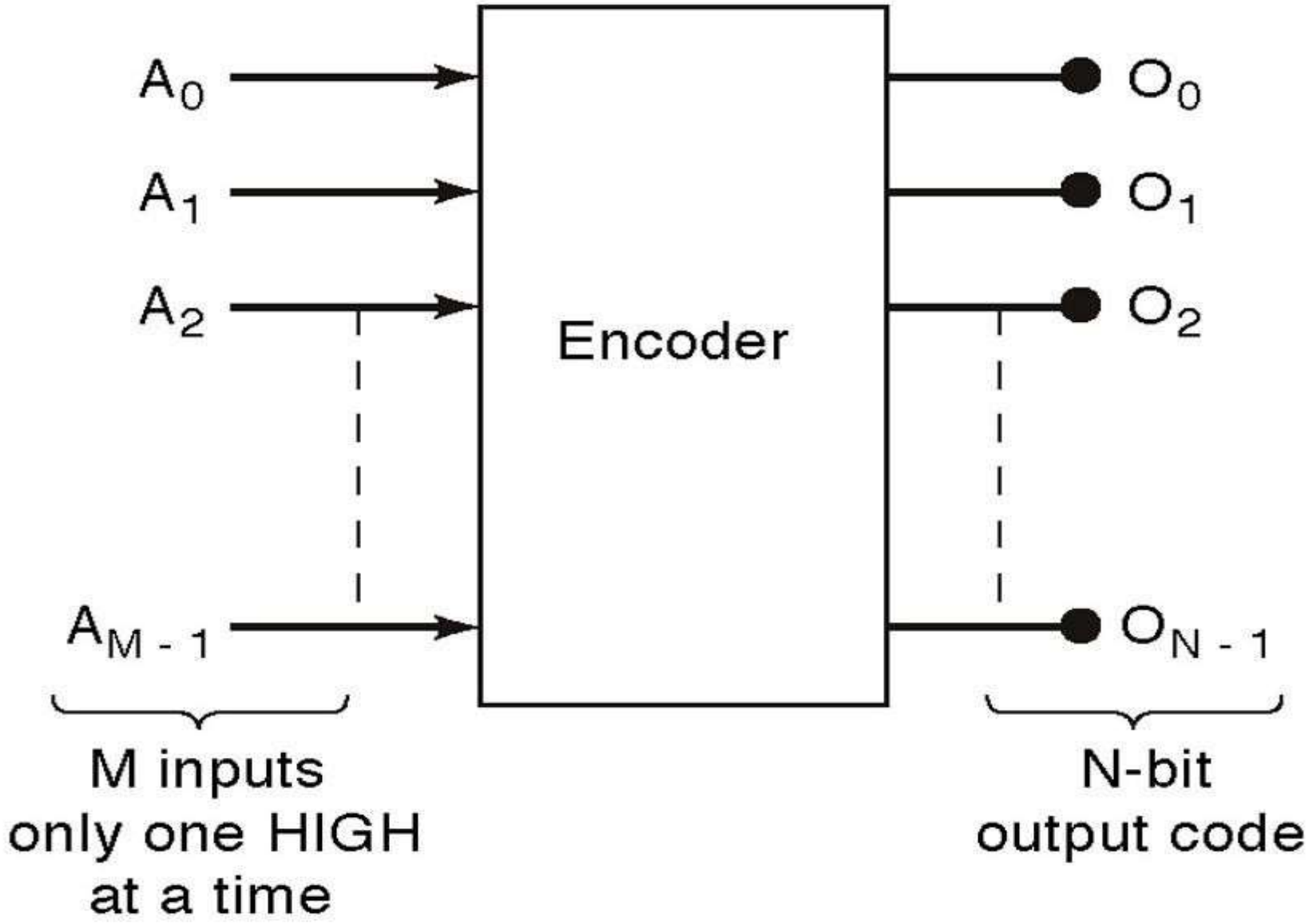
Encoder

- An encoder is a combinational logic circuit that essentially performs a "reverse" of decoder functions.
- An encoder accepts an active level on one of its inputs, representing digit, such as a decimal or octal digits, and converts it to a coded output such as BCD or binary.
- Encoders can also be devised to encode various symbols and alphabetic characters.
- The process of converting from familiar symbols or numbers to a coded format is called ***encoding***.

- Most decoders accept an input code and produce a HIGH (or a LOW) at one and only one output line. In otherworlds , a decoder identifies, recognizes, or detects a particular code. The opposite of this decoding process is called encoding and is performed by a logic circuit called an encoder.

- An encoder has a number of input lines, only one of which input is activated at a given time and produces an N-bit output code,depending on which input is activated.

General encoder diagram

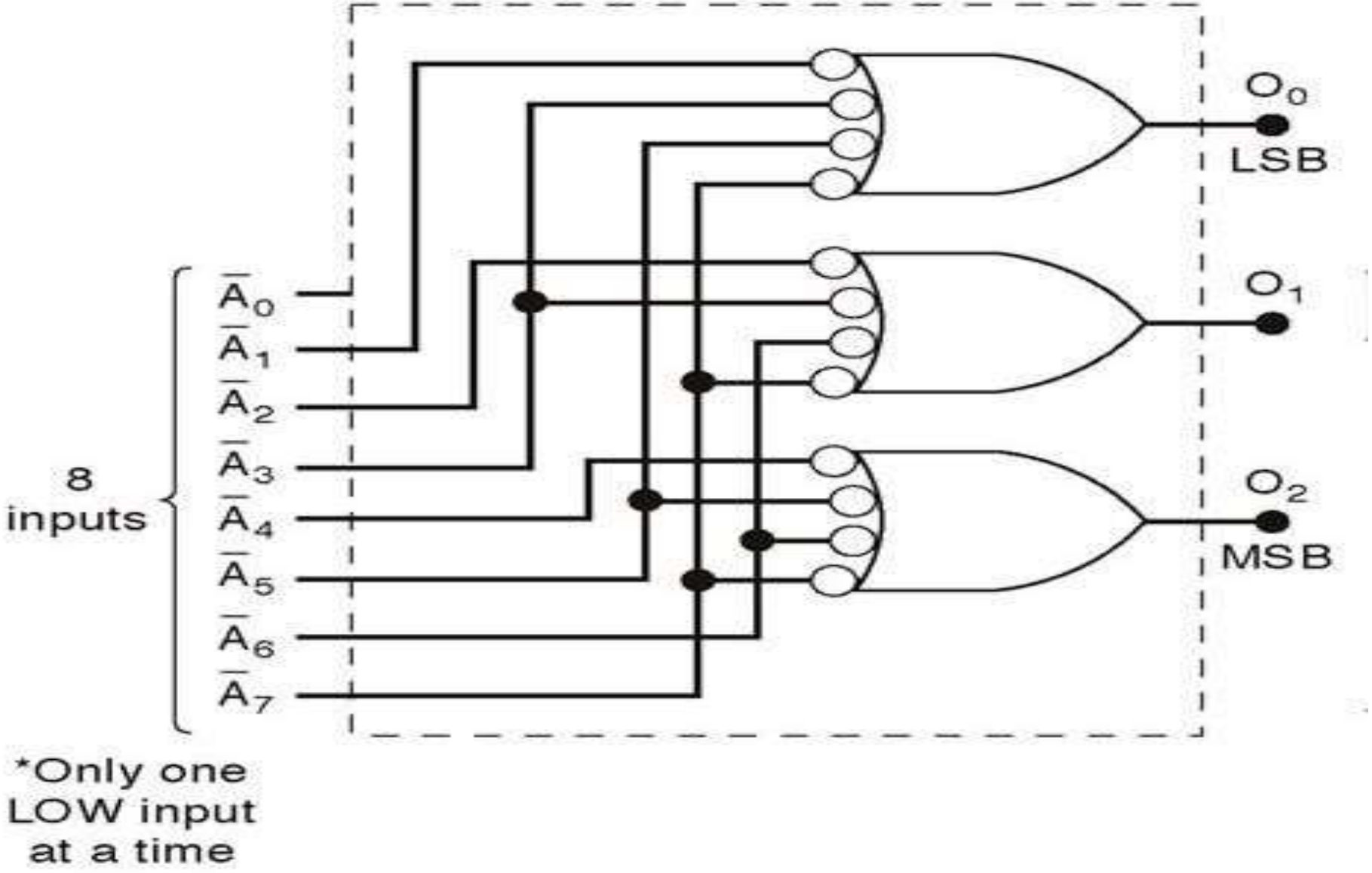


Truth table for octal-to binary encoder [8-line- 3-line]

Inputs								Outputs		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

A low at any single input will produce the output binary code corresponding to that input. For instance , a low at A_3' will produce O_2 =0, O_1=1 and O_0 =1, which is binary code for 3. A_o' is not connected to the logic gates because the encoder outputs always be normally at 0000 when none of the inputs is LOW

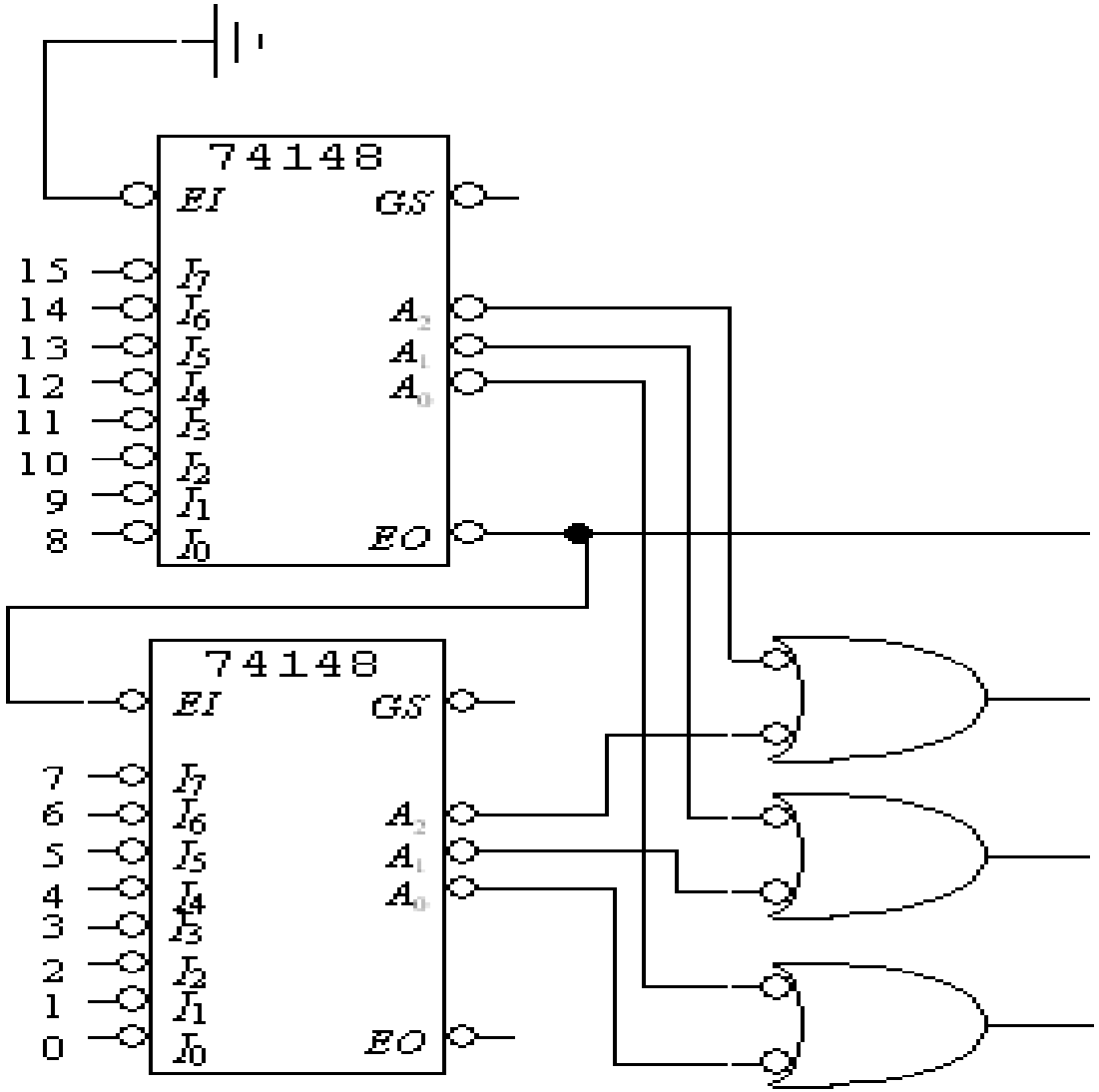
Logic circuit for octal-to binary encoder [8-line- 3-line]



The 16 –to–4 Encoder

The 74LS148 can be expanded to a 16–line–to–4–line encoder by connecting the *EO* of the higher–order encoder to the *EI* of the lower–order encoder and negative–ORing the corresponding binary outputs as shown

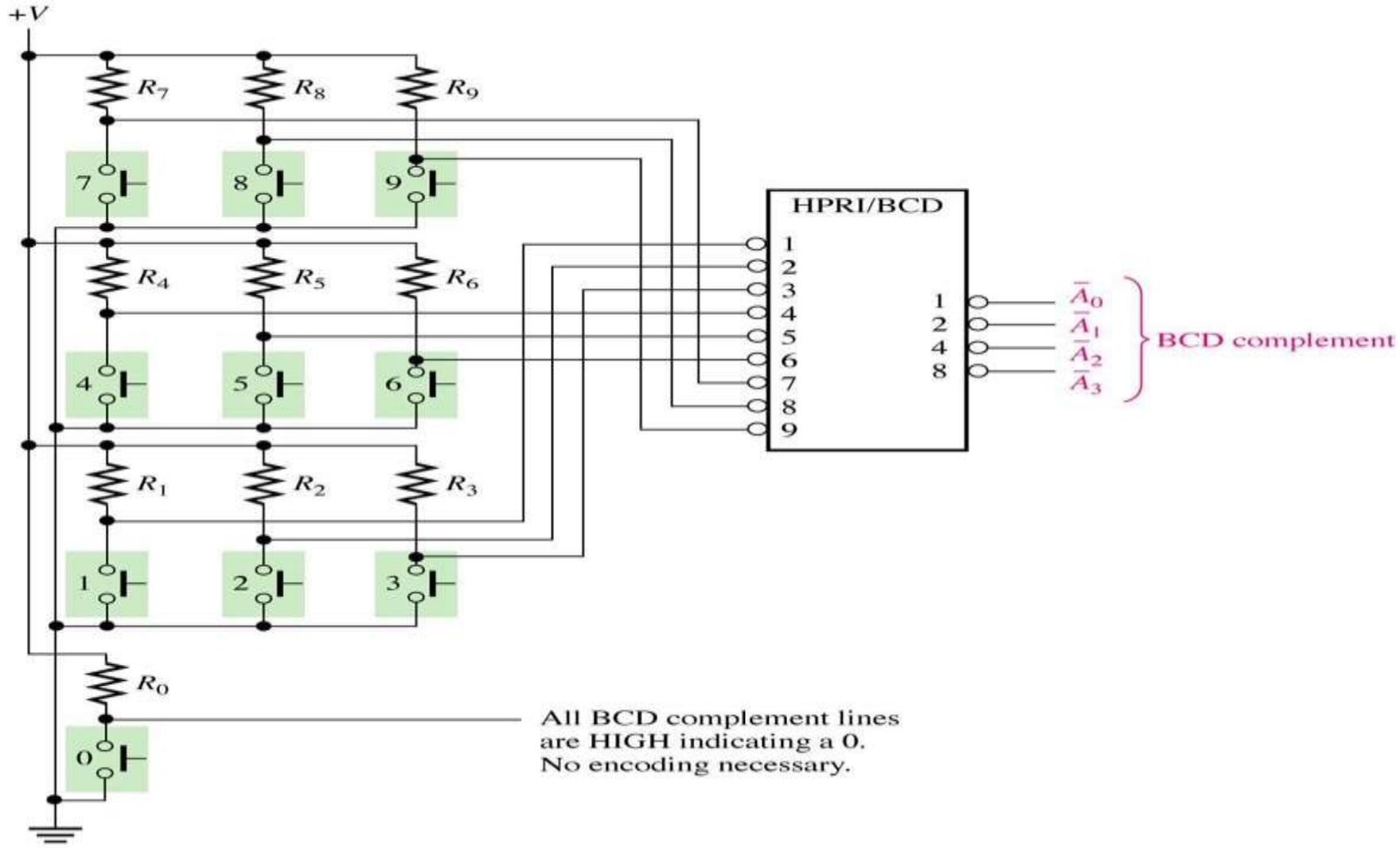
The 16 –to–4 Encoder



A 16–line–to–4–line encoder using 74LS148s and external logic.

Application example

A simplified keyboard encoder.



- When one of the keys is pressed, the decimal digit is encoded to the corresponding BCD code
- The keys are represented by 10 push-button switches, each with a **pull-up resistor** to V+. The pull-up resistor ensures that the line is HIGH when a key is not depressed.
- When a key is depressed, the line is connected to ground, and a LOW is applied to the corresponding encoder input.
- The zero key is not connected because the BCD output represents zero when none of the other keys is depressed
- The BCD complement output of the encoder goes into a storage device, and each successive BCD code is stored until the entire number has been entered

Assignment

Design a single encoder for following functions.

F1 = Σm(1, 3, 7, 15)

F2 = Σm(4,6,8,10)