COMBINATIONAL LOGIC

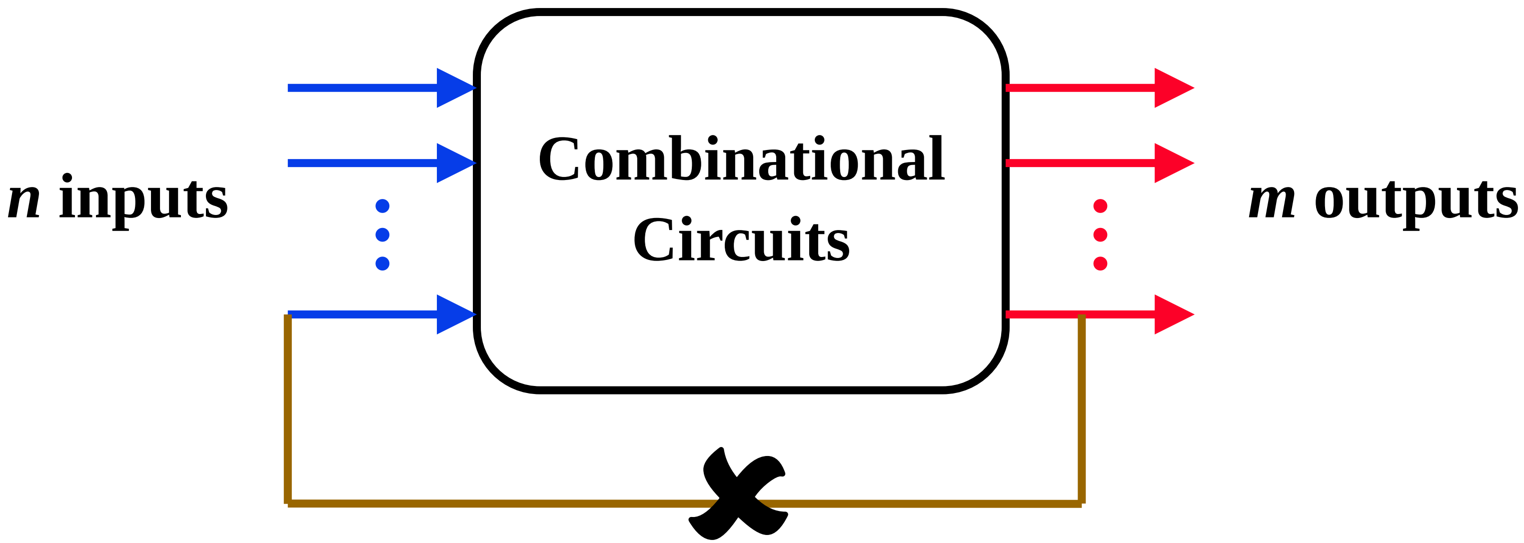
DIGITAL LOGIC DESIGN

Iqra Chaudhary (Lecturer CS dept. NUML)

Combinational Circuits

★ **Output is function of input only**

i.e. no feedback

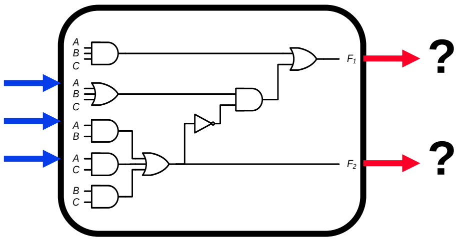


When input changes, output may change (after a delay)

Combinational Circuits

★ Analysis

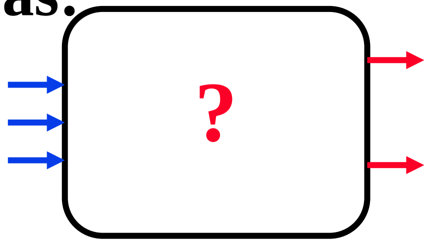
- **Given a circuit, find out its *function***



- **Function may be expressed as:**
  - ♦ **Boolean function/equation**
  - ♦ **Truth table**

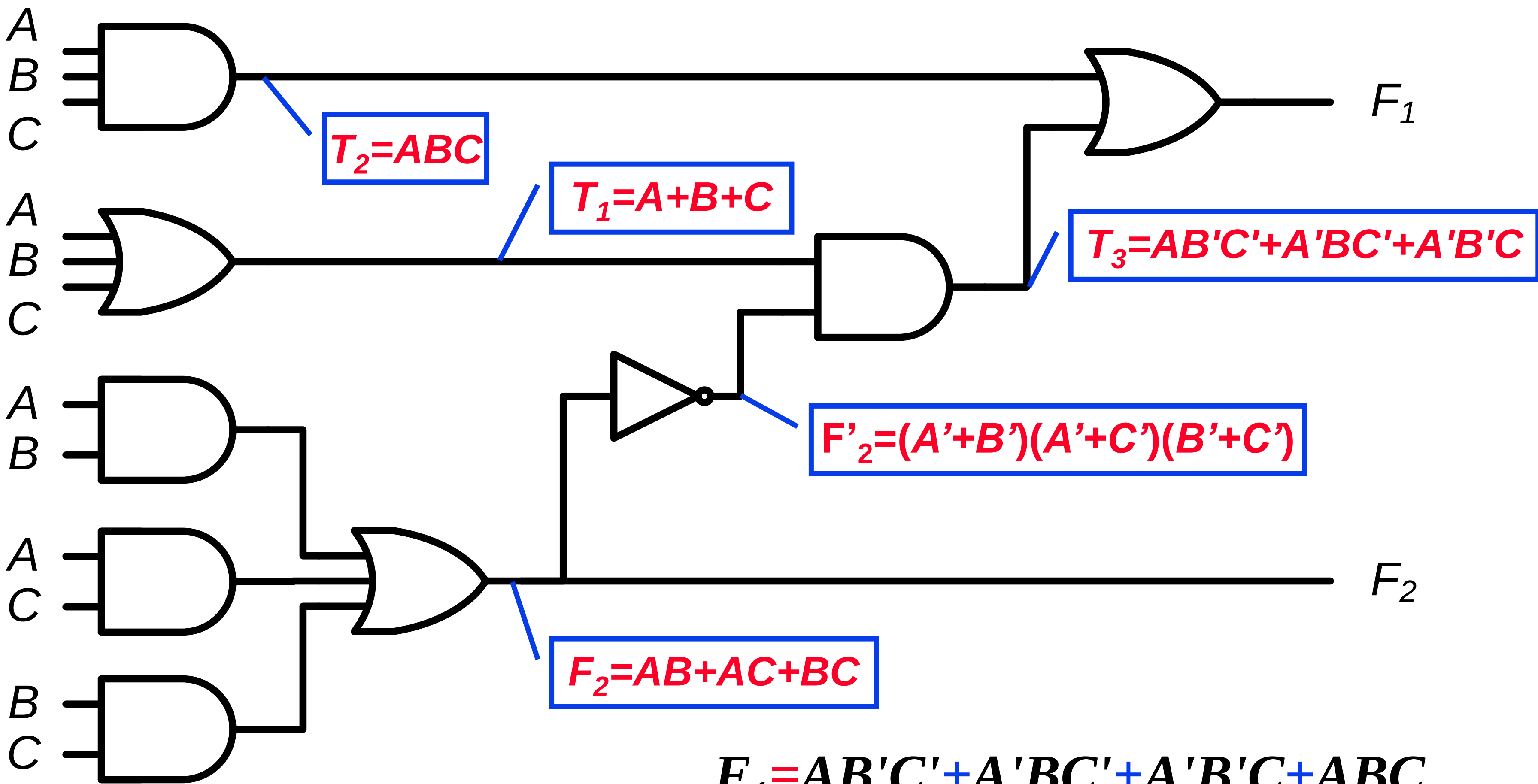
★ Design

- **Given a desired function/specification, determine its *circuit***
- **Function/specification may be expressed as:**
  - ♦ **Boolean function**
  - ♦ **Truth table**
  - ♦ **Statement**



Analysis Procedure

★ Boolean Expression Approach



$$F_1 = AB'C' + A'BC' + A'B'C + ABC$$

$$F_2 = AB + AC + BC$$

Boolean Expression Approach

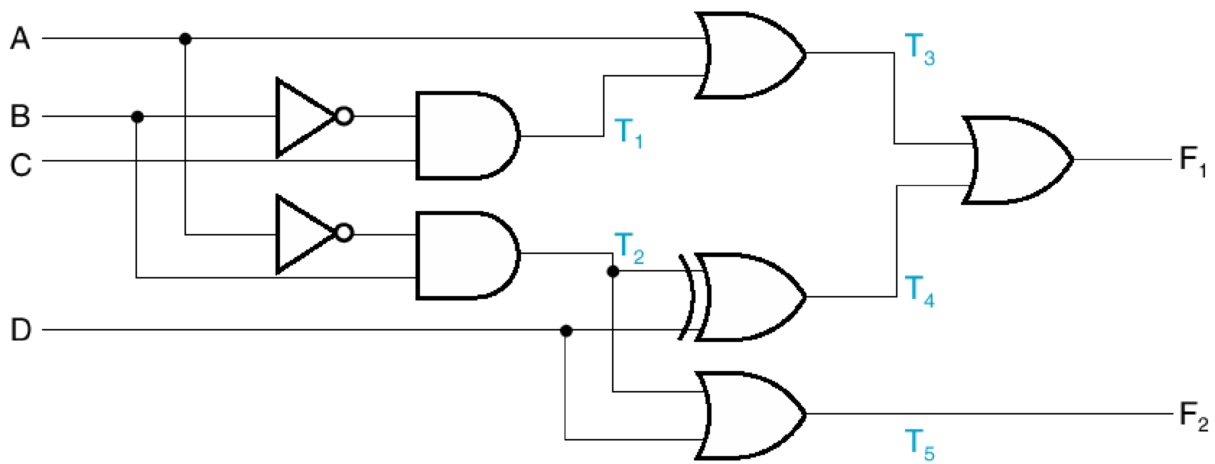
Label gate outputs of input variables

Determine Boolean functions or values

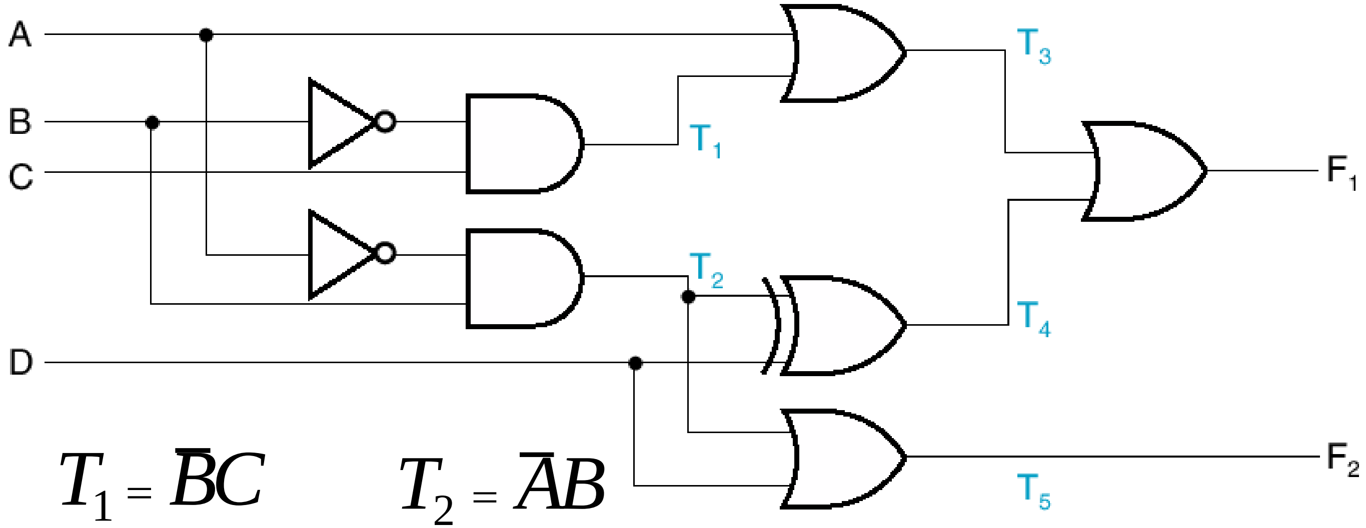
Label outputs of gates fed by previously labeled gates

Determine Boolean function or values

Repeat 2 until done



Boolean Expression Approach



$T_1 = \overline{B}C$ $\quad$ $T_2 = \overline{A}B$

$$T_3 = A + T_1 = A + \overline{B}C$$

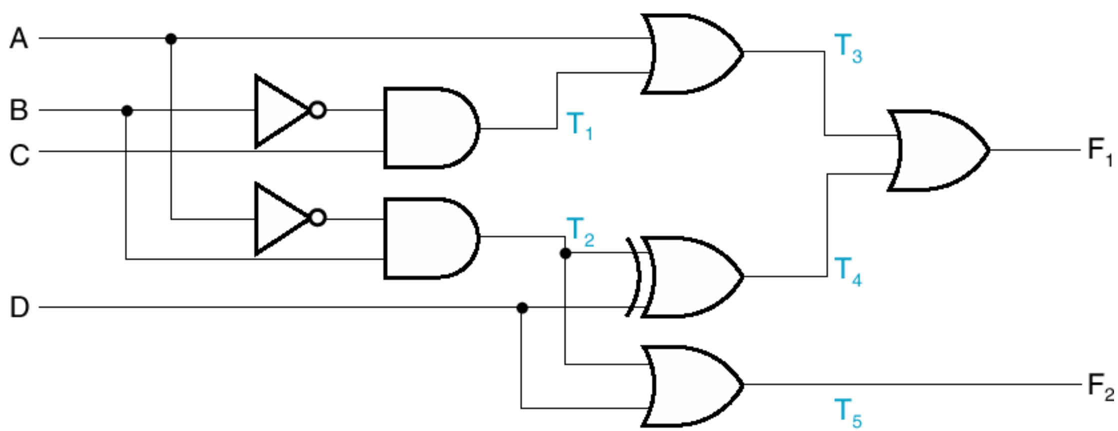
Boolean Expression Approach :Cont .. Analysis Example

$$T_4 = T_2 \oplus D = (A\bar{B}) \oplus D = A\bar{B}\bar{D} + \bar{A}D + BD$$

$$T_5 = T_2 + D = A\bar{B} + D$$

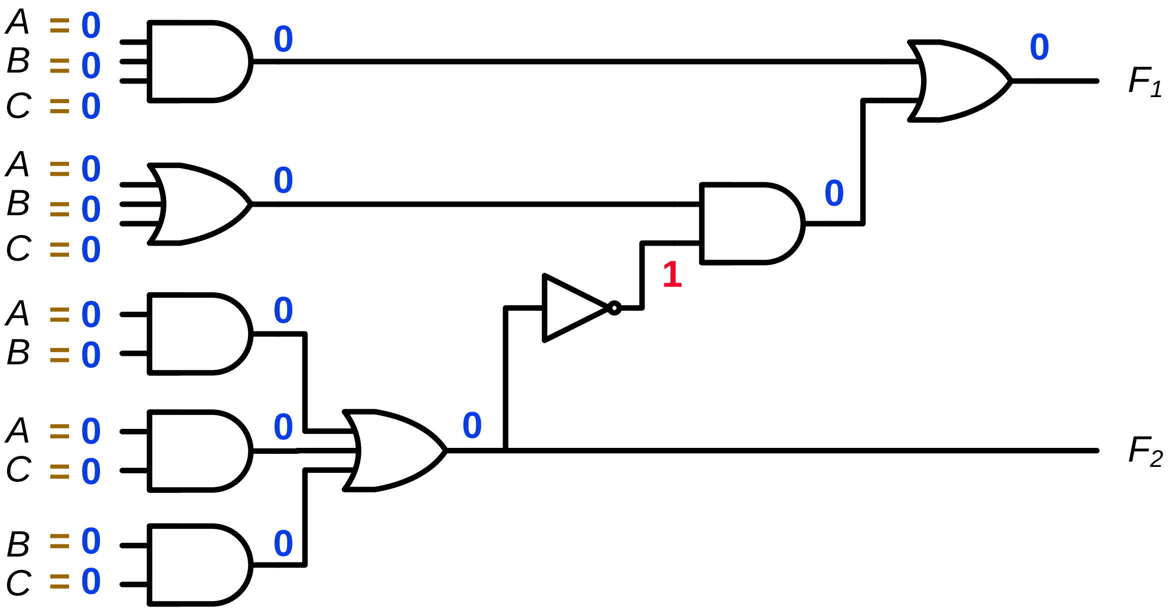
$$F_1 = T_3 + T_4 = A + \bar{B}C + A\bar{B}\bar{D} + \bar{A}D + BD$$

$$F_2 = T_5 = AB + D$$



Analysis Procedure

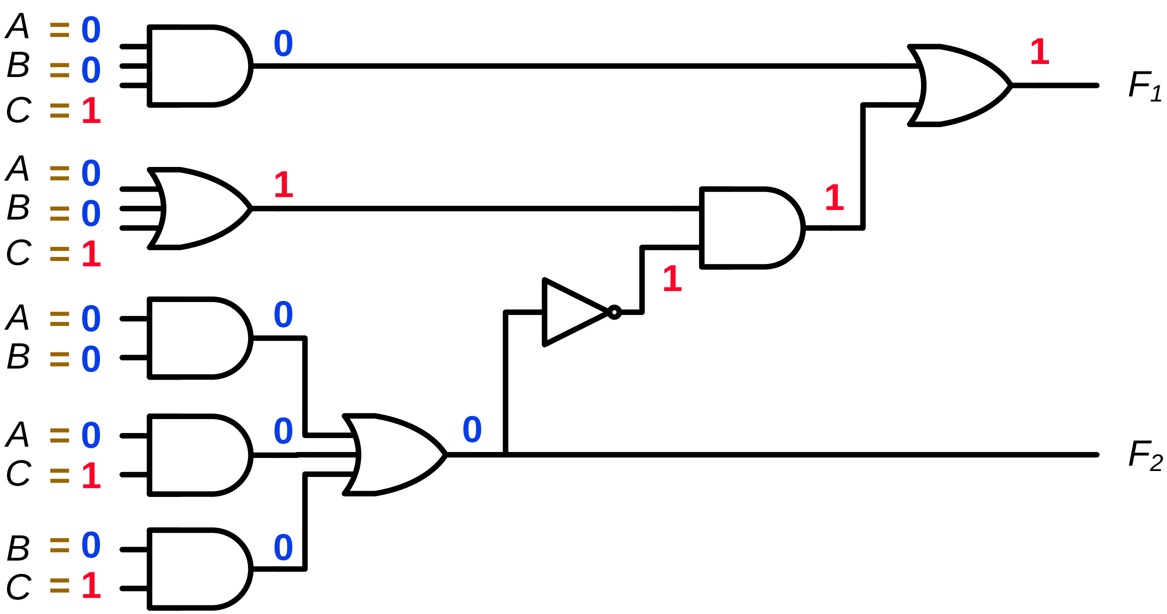
★ Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0

Analysis Procedure

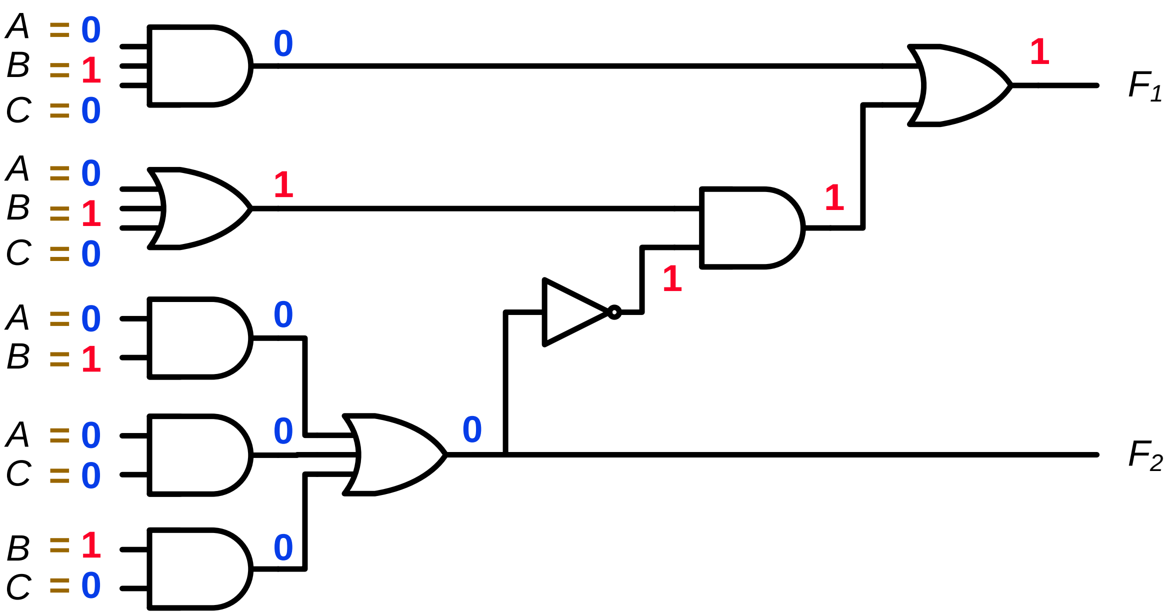
★ Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0

Analysis Procedure

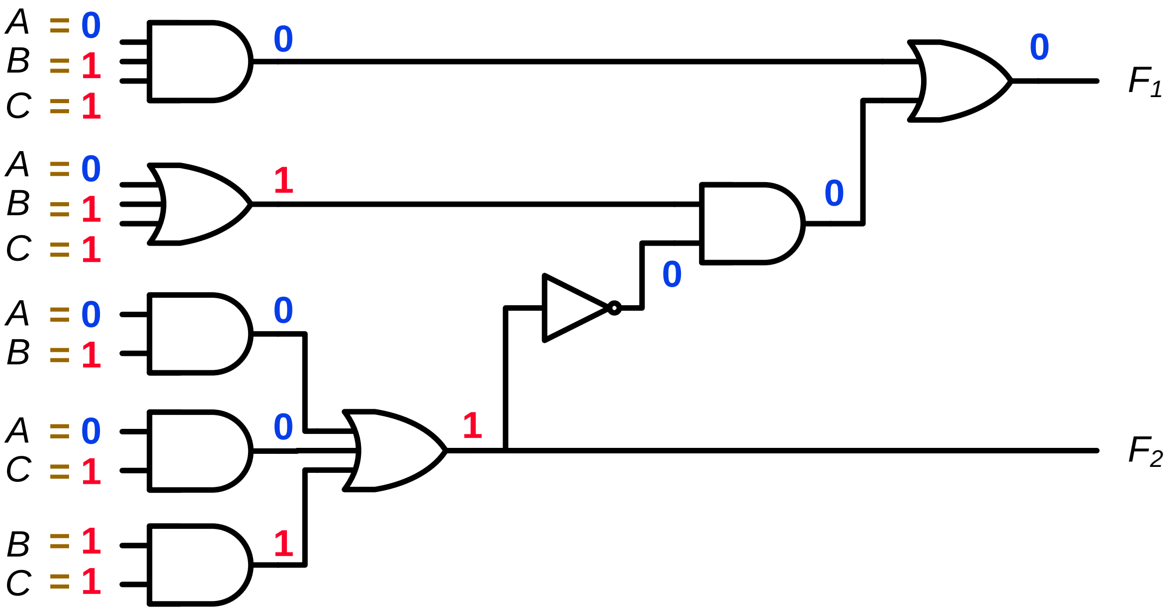
★ Truth Table Approach



A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

Analysis Procedure

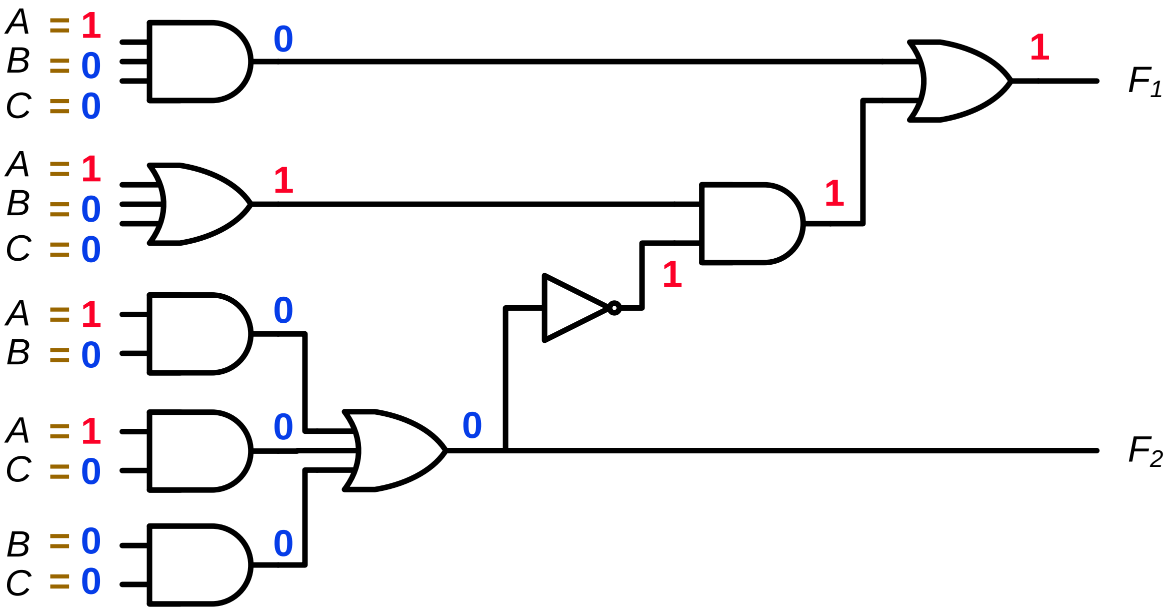
★ Truth Table Approach



A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1

Analysis Procedure

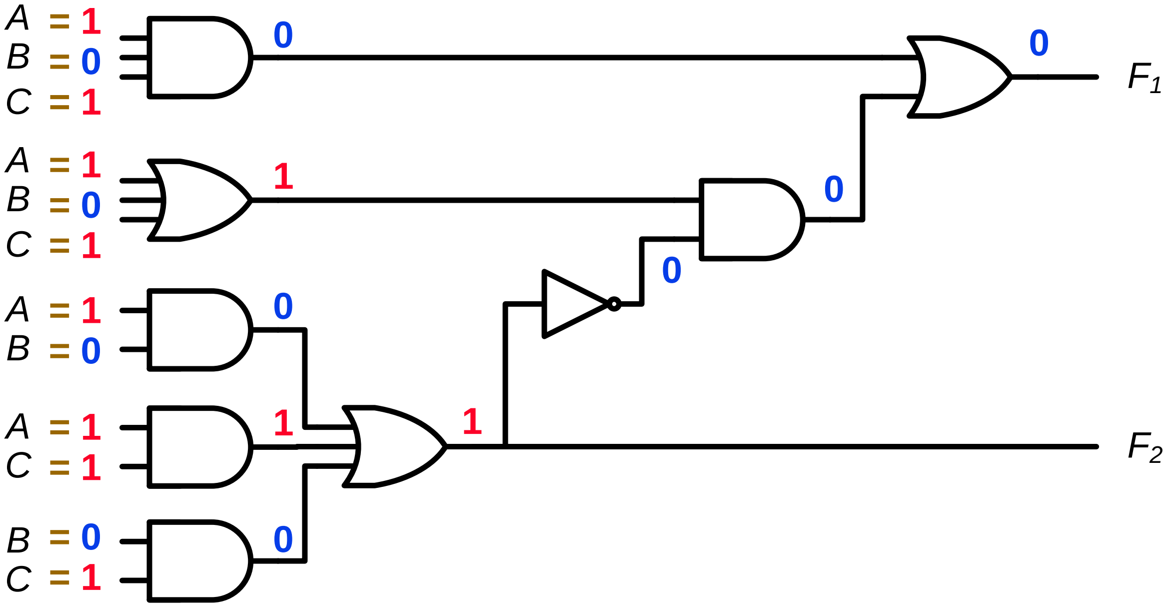
★ Truth Table Approach



A B C	F_1	F_2
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

Analysis Procedure

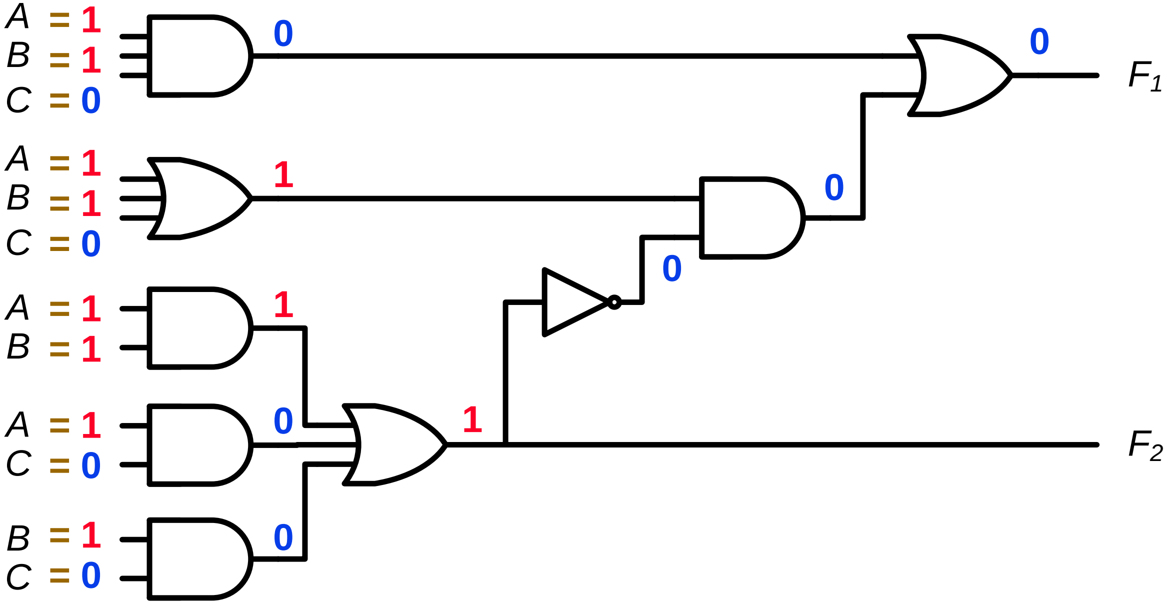
★ Truth Table Approach



A	B	C	F_1	F_2
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

Analysis Procedure

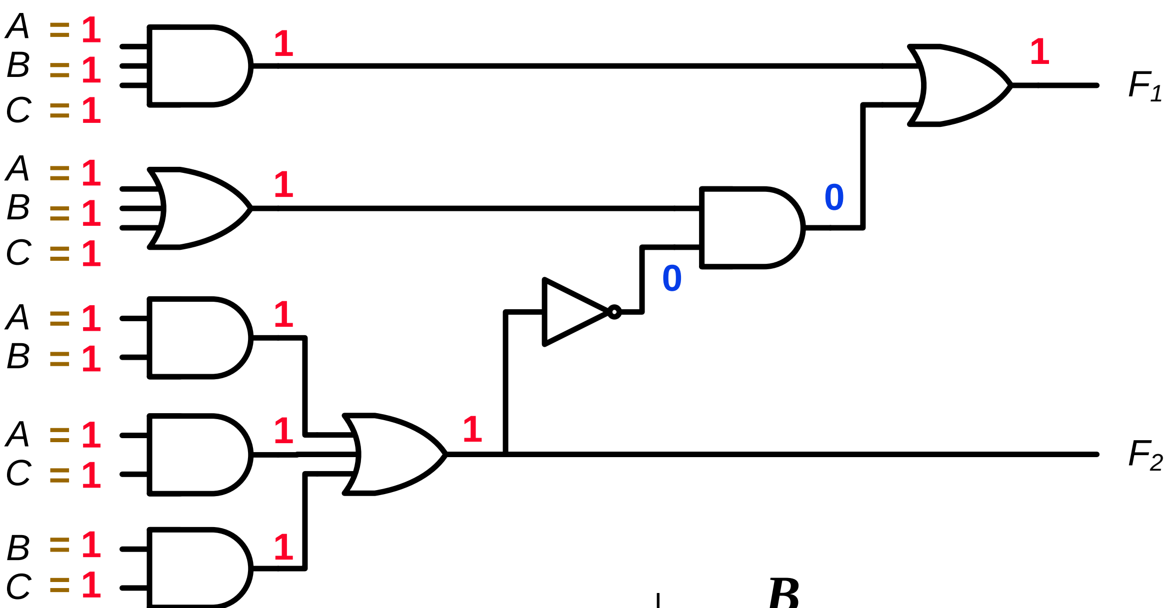
★ Truth Table Approach



A	B	C	F_1	F_2
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

Analysis Procedure

★ Truth Table Approach



A	B	C	F_1	F_2
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

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

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

$F_1 = AB'C' + A'BC' + A'B'C + ABC$ $\qquad$ $F_2 = AB + AC + BC$

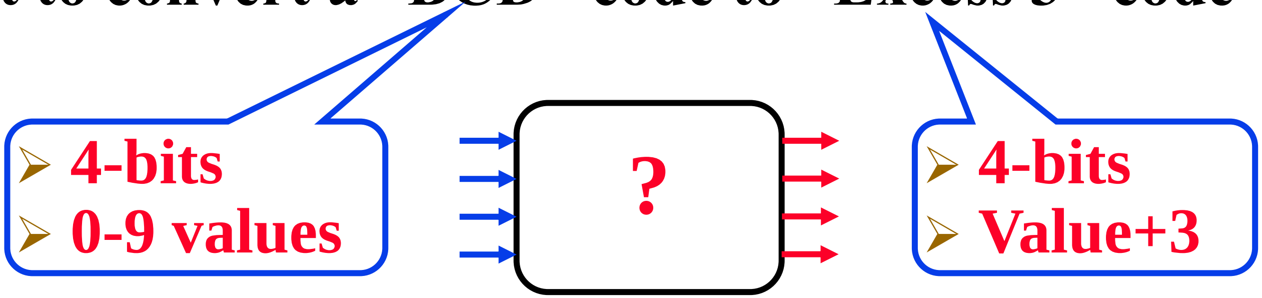
Design Procedure

★ **Given a problem statement:**

- **Determine the number of *inputs* and *outputs***
- **Derive the truth table**
- **Simplify the Boolean expression for each output**
- **Produce the required circuit**

Example 1:

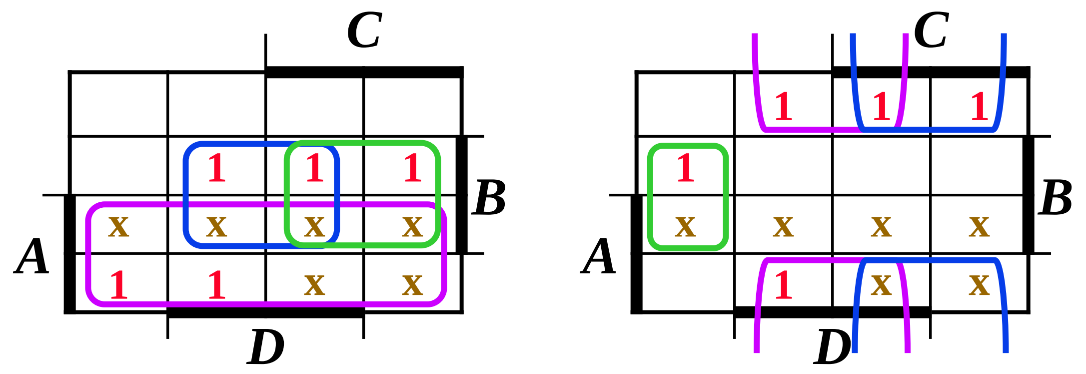
Design a circuit to convert a “BCD” code to “Excess 3” code



Design Procedure

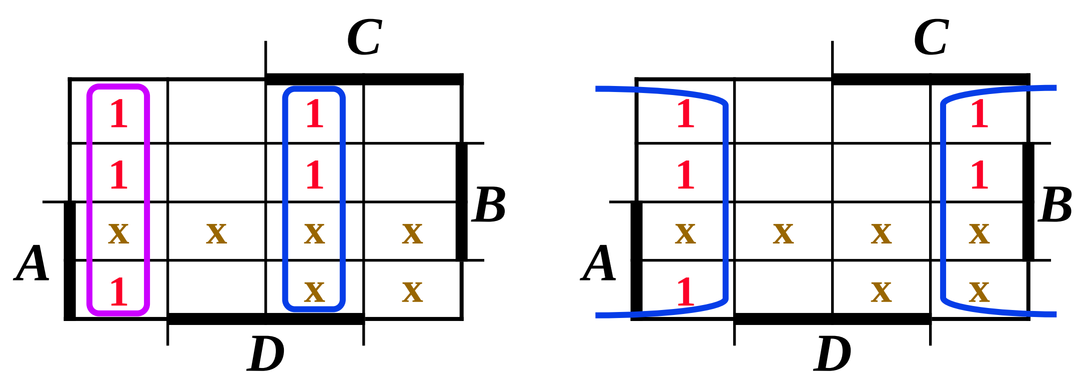
★ BCD-to-Excess 3 Converter

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
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



$w = A + BC + BD$

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



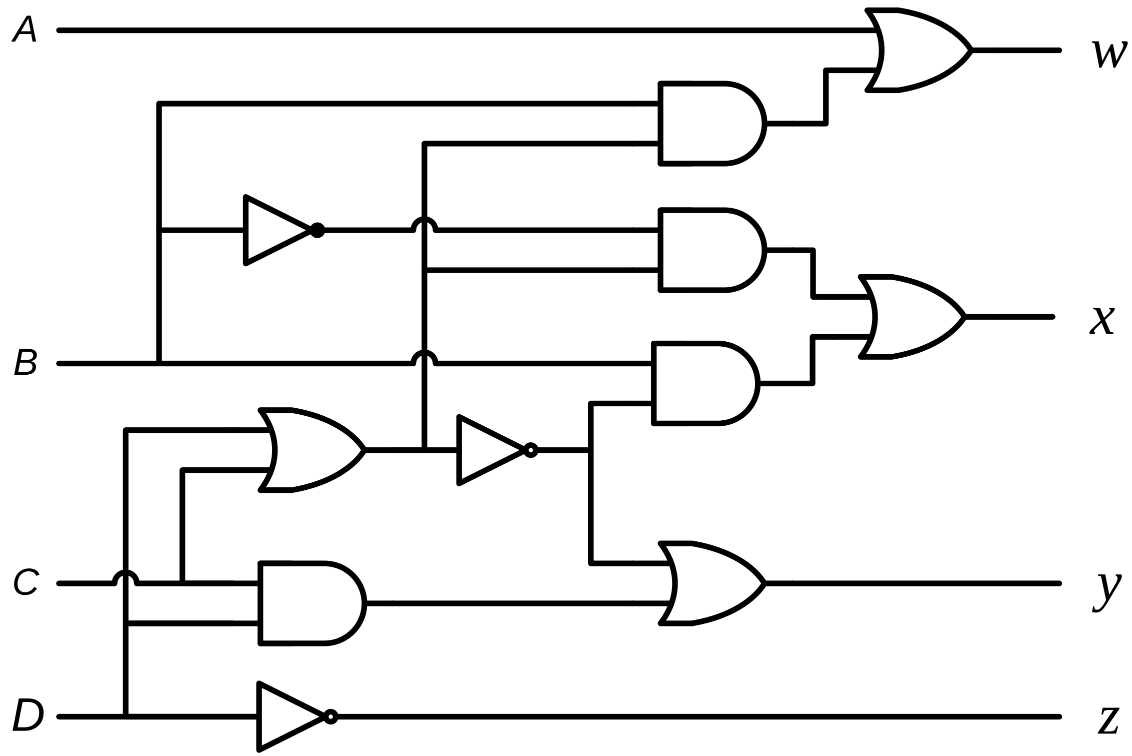
$y = C'D' + CD$

$z = D'$

Design Procedure

★ BCD-to-Excess 3 Converter

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
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



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

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

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

$z = D'$

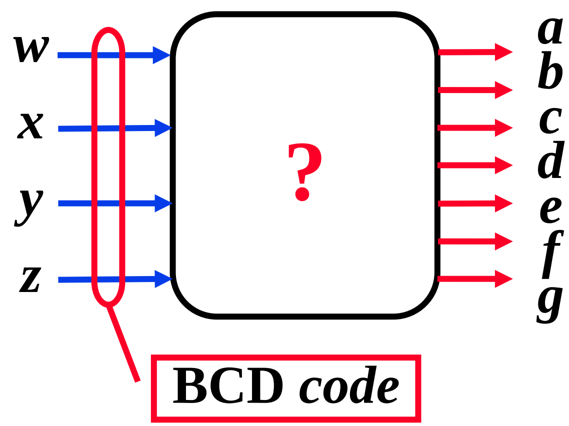
Example 2:
Seven-Segment Decoder

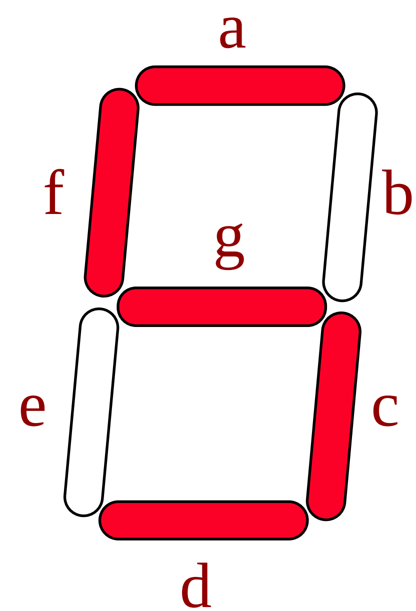
(a) Segment designation

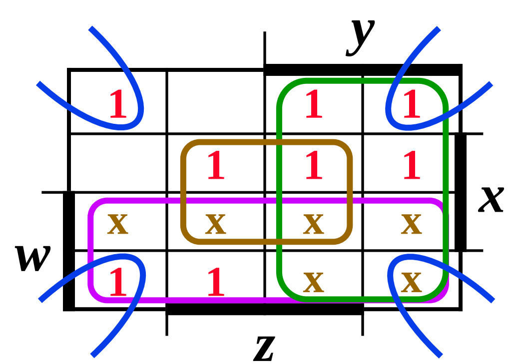
(b) Numerical designation for display

Example 2:
Seven-Segment Decoder

w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	x	x	x	x	x	x	x
1	0	1	1	x	x	x	x	x	x	x
1	1	0	0	x	x	x	x	x	x	x
1	1	0	1	x	x	x	x	x	x	x
1	1	1	0	x	x	x	x	x	x	x
1	1	1	1	x	x	x	x	x	x	x







$a = w + y + xz + x'z'$

$b = \ldots$
$c = \ldots$
$d = \ldots$

Example 3: Implement circuit that can convert 3bit gray code into binarycode

B2 = Σ m (4, 5, 6, 7)

B1 = Σ m (2, 3, 4, 5)

B0 = Σ m (1, 2, 4, 7)

	G1'G0' 00	G1'G0 01	G1G0 11	G1G0' 10
G2' 0	(0)	(1)	(3)	(2)
G2 1	1 (4)	1 (5)	1 (7)	1 (6)

$B_2 = G_2$

	G1'G0' 00	G1'G0 01	G1G0 11	G1G0' 10
G2' 0	(0)	(1)	1 (3)	1 (2)
G2 1	1 (4)	1 (5)	(7)	(6)

B1=G2'G1+G2G1'

$B_1 = G_2 \oplus G_1$

	G1'G0' 00	G1'G0 01	G1G0 11	G1G0' 10
G2' 0	(0)	1 (1)	(3)	1 (2)
G2 1	1 (4)	(5)	1 (7)	(6)

$B_0 = G_2 \oplus G_1 \oplus G_0$

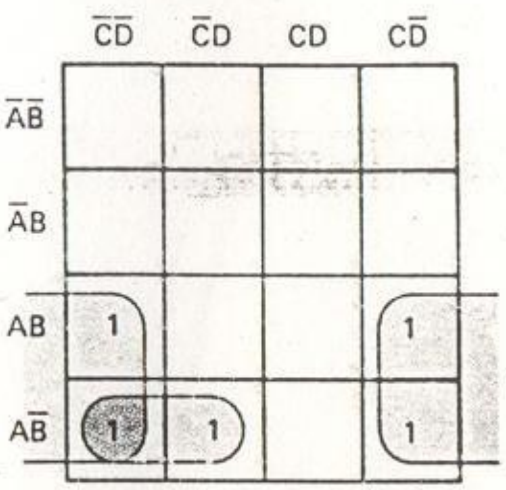
Decimal Equivalent	Gray Code			Binary Output		
	G2	G1	G0	B2	B1	B0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
3	0	1	1	0	1	0
2	0	1	0	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1
5	1	0	1	1	1	0
4	1	0	0	1	1	1

Example 4: Implement a binary to gray converter

Decimal Equivalent	Binary			Gray Code		
	B2	B1	B0	G2	G1	G0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

Example 5: System Design Application

Design a circuit that can be built using an AOI and inverters that will output a HIGH (1) whenever the 4-bit hexadecimal input is an odd number from 0 to 9.



Odd number = $A\overline{D} + A\overline{B}\,\overline{C}$

Hex Truth Table Used to Determine the Equation for Odd Numbers[a] from 0 to 9

D	C	B	A	DEC	
0	0	0	0	0	
0	0	0	1	1	$\leftarrow A\overline{B}\,\overline{C}\,\overline{D}$
0	0	1	0	2	
0	0	1	1	3	$\leftarrow AB\overline{C}\,\overline{D}$
0	1	0	0	4	
0	1	0	1	5	$\leftarrow A\overline{B}C\overline{D}$
0	1	1	0	6	
0	1	1	1	7	$\leftarrow ABC\overline{D}$
1	0	0	0	8	
1	0	0	1	9	$\leftarrow A\overline{B}\,\overline{C}D$

[a] Odd number $= A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + A\overline{B}C\overline{D} + ABC\overline{D} + A\overline{B}\,\overline{C}D$.

Example 5: System Design Application

implementation of the "odd-number decoder" using an AOI