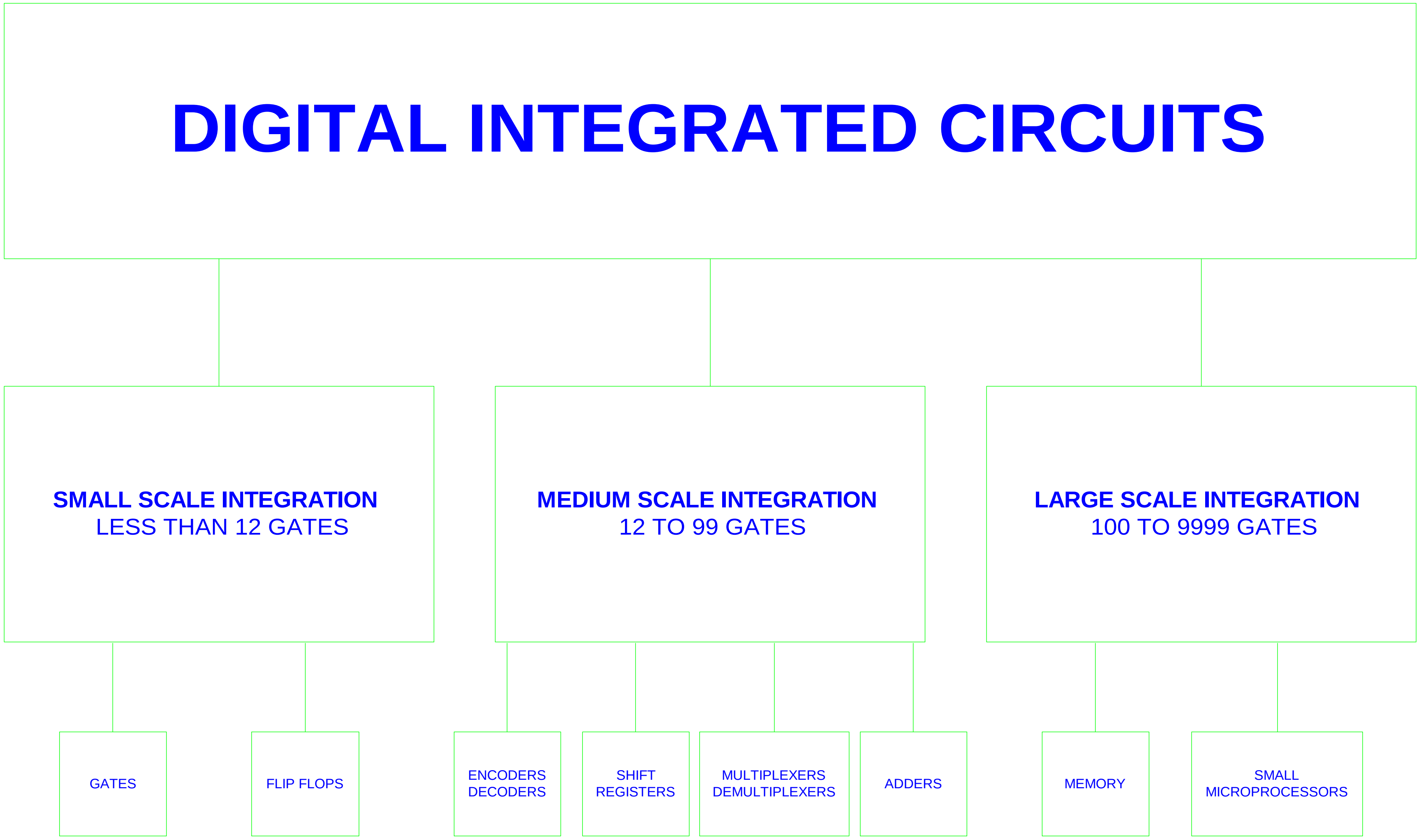
Digital Electronics and Microprocessors

Class 15

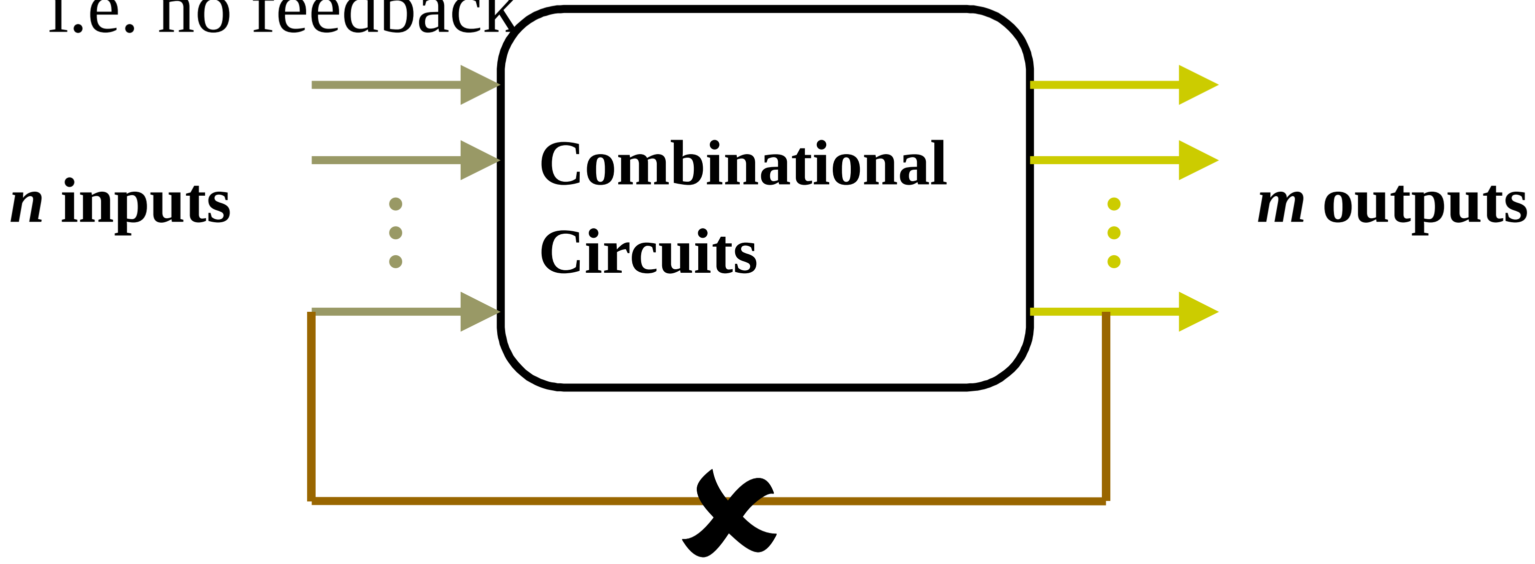
CHHAYADEVI BHAMARE


DIGITAL INTEGRATED CIRCUITS
SMALL SCALE INTEGRATION
LESS THAN 12 GATES
MEDIUM SCALE INTEGRATION
12 TO 99 GATES
LARGE SCALE INTEGRATION
100 TO 9999 GATES
GATES
FLIP FLOPS
ENCODERS DECODERS
SHIFT REGISTERS
MULTIPLEXERS DEMULTIPLEXERS
ADDERS
MEMORY
SMALL MICROPROCESSORS

Combinational Circuits

- Output is function of input only

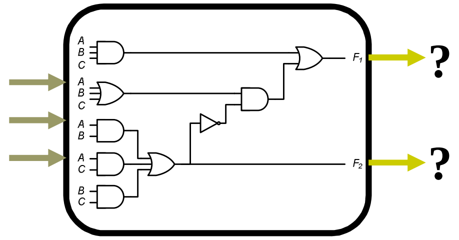
  i.e. no feedback



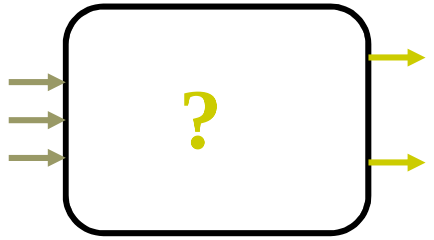
When input changes, output may change (after a delay)

Combinational Circuits

- Analysis
  - Given a circuit, find out its *function*
  - Function may be expressed as:
    - Boolean function
    - Truth table

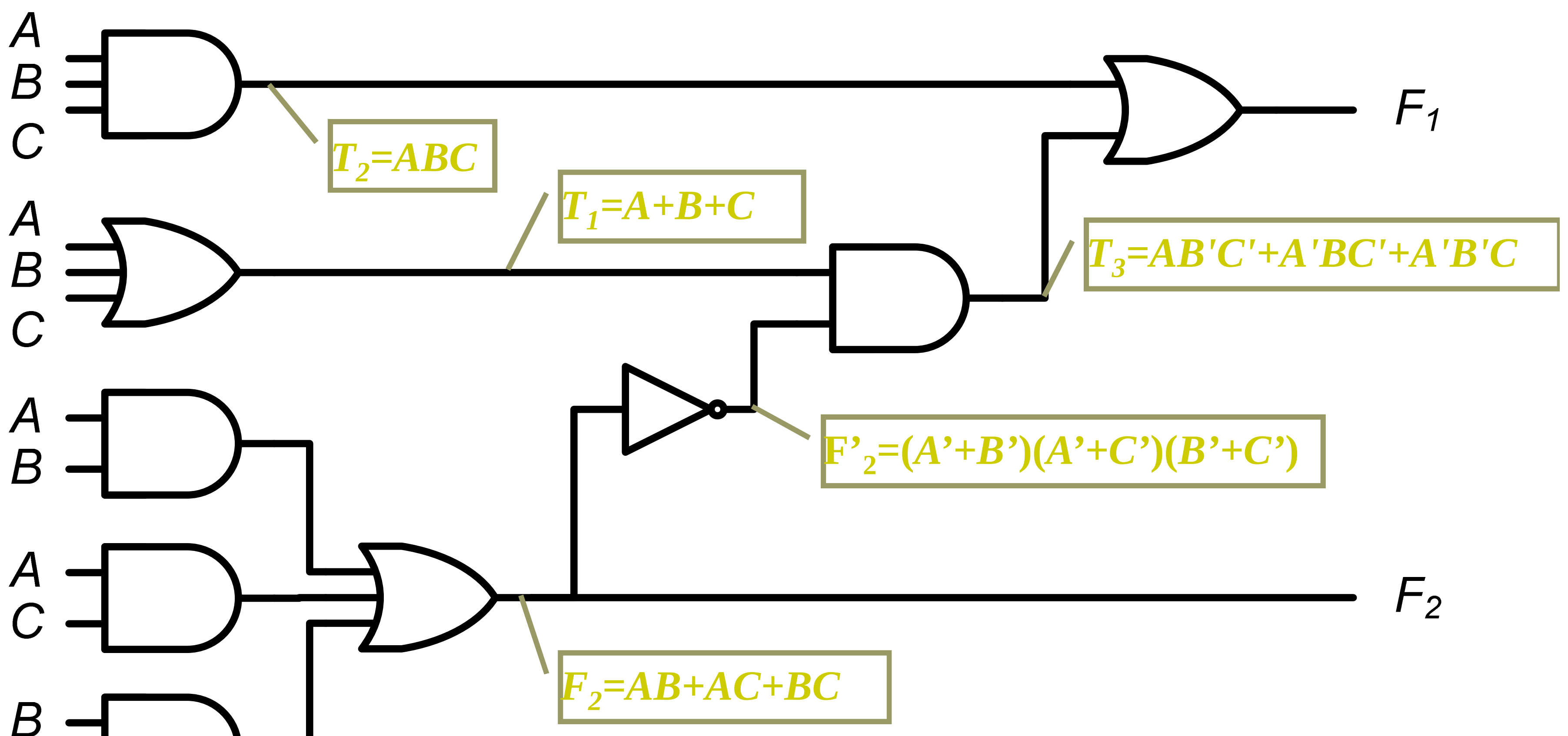


- Design
  - Given a desired function, determine its *circuit*
  - Function may be expressed as:
    - Boolean function
    - Truth table



Analysis Procedure

- Boolean Expression Approach

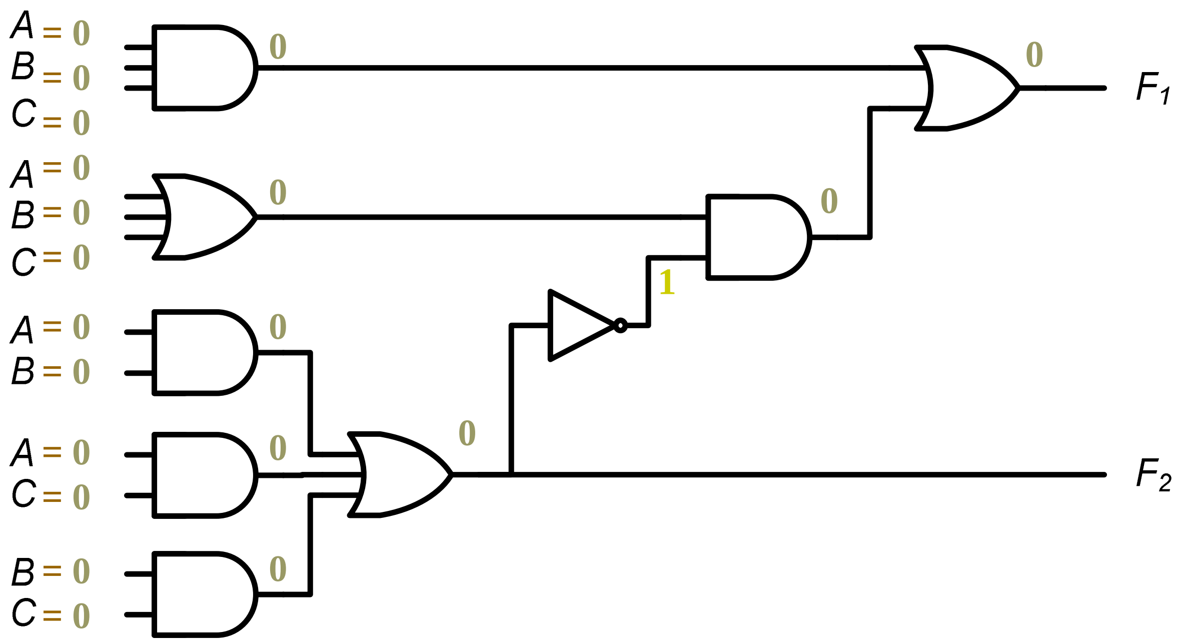


$F_1=AB'C'+A'BC'+A'B'C+ABC$

$F_2=AB+AC+BC$

Analysis Procedure

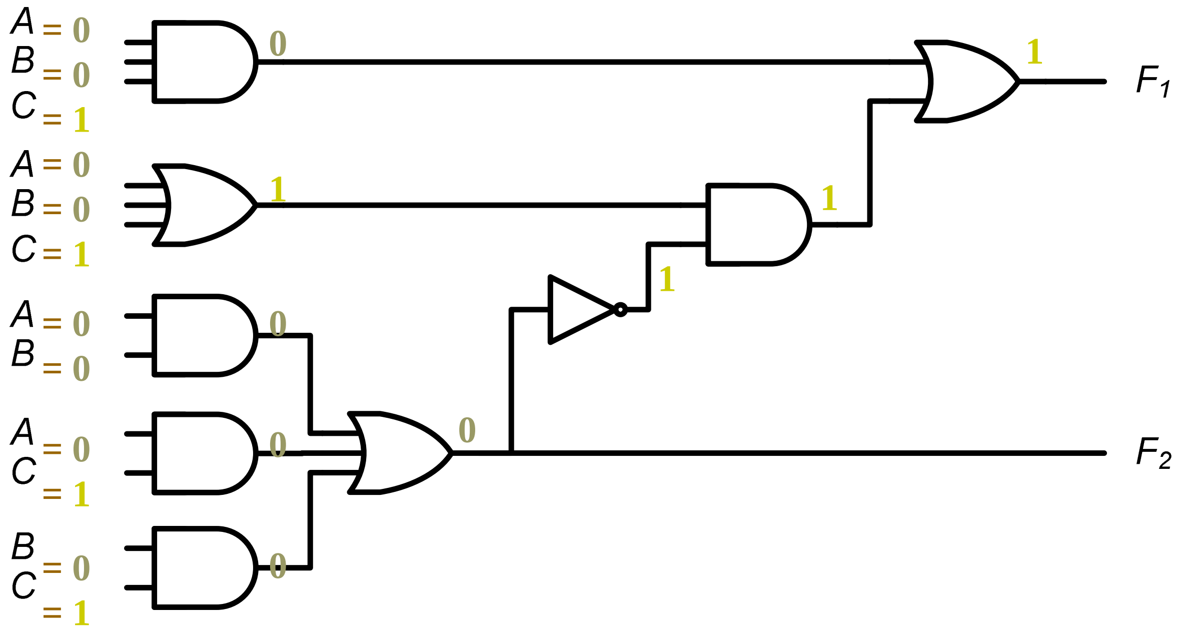
- Truth Table Approach



A B C	F_1	F_2
0 0 0	0	0

Analysis Procedure

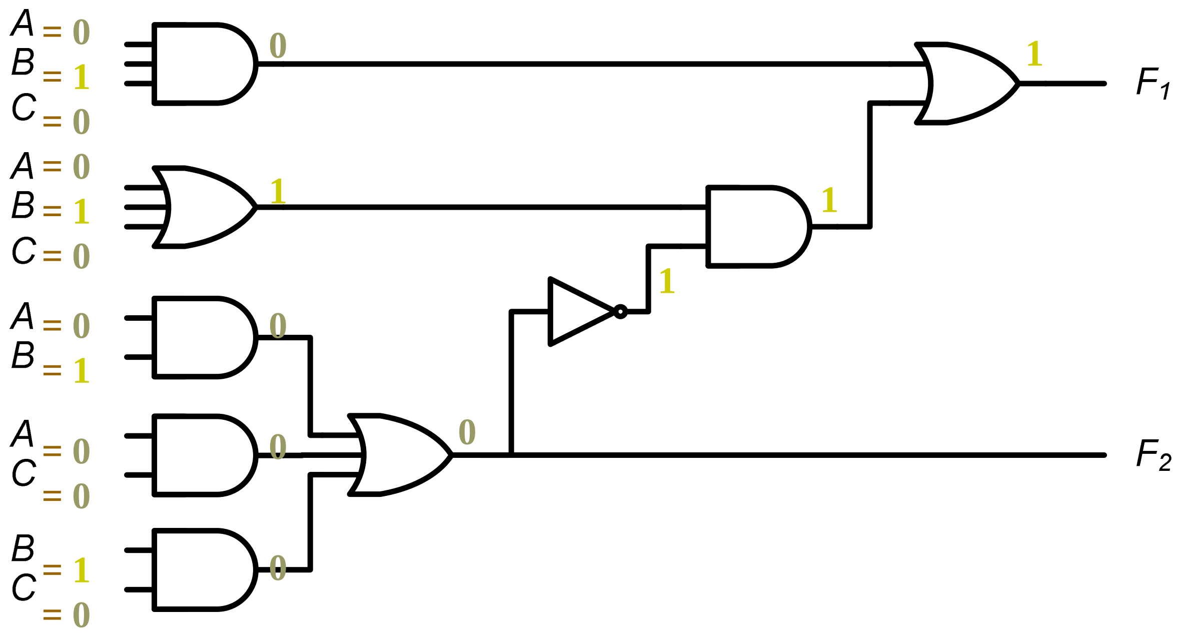
- Truth Table Approach



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

Analysis Procedure

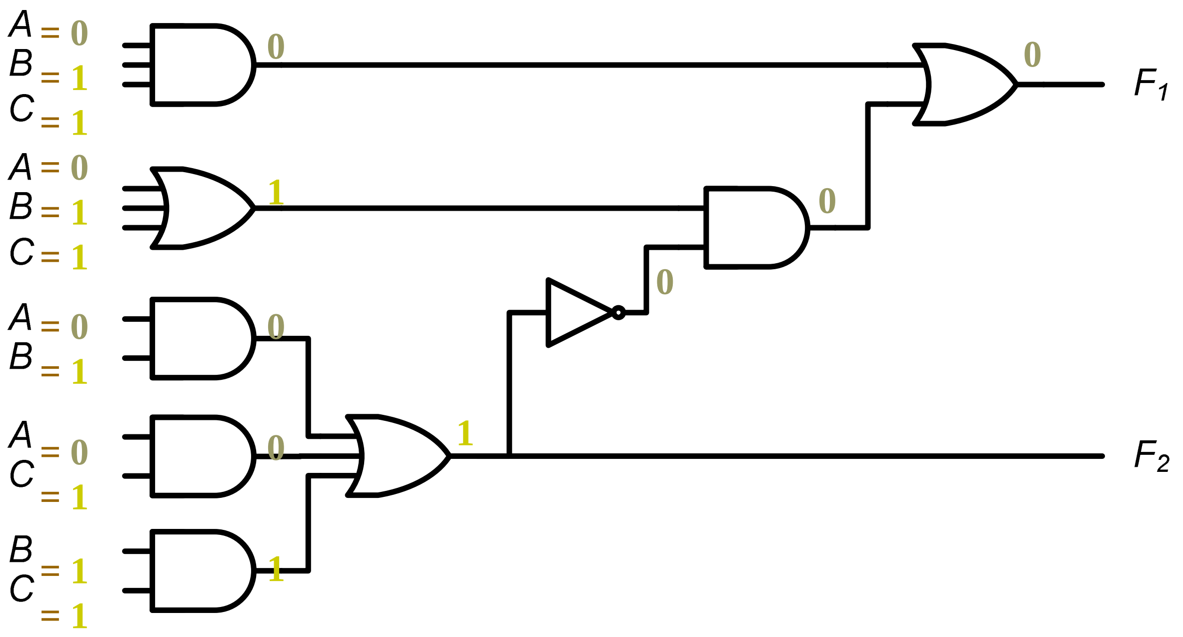
- Truth Table Approach



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

Analysis Procedure

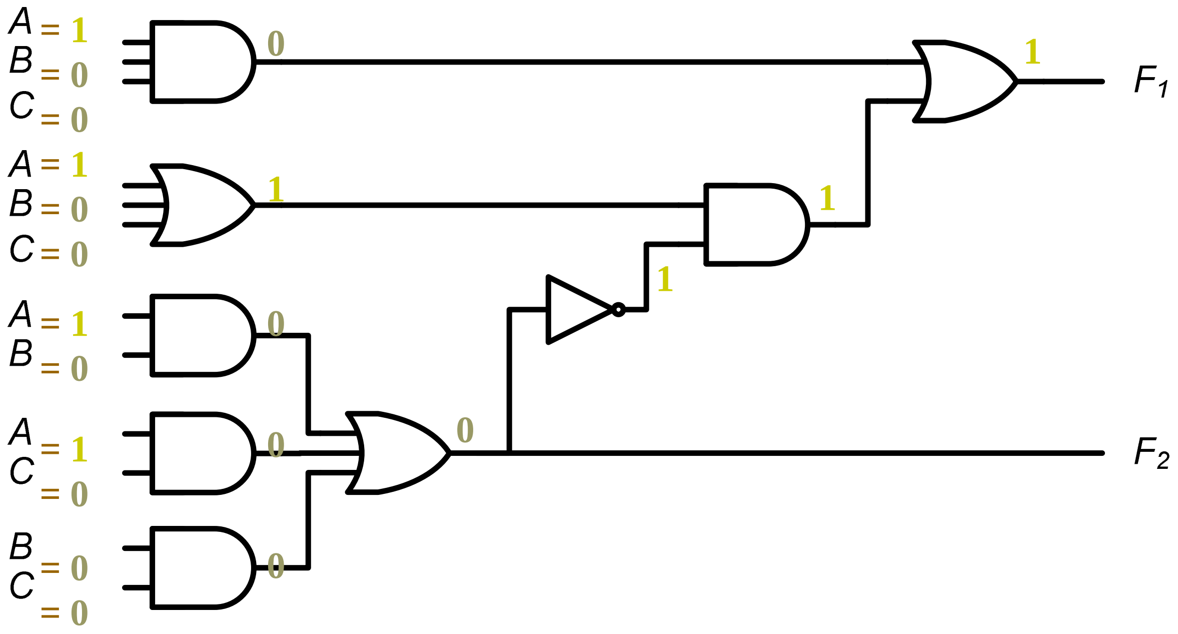
- Truth Table Approach



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

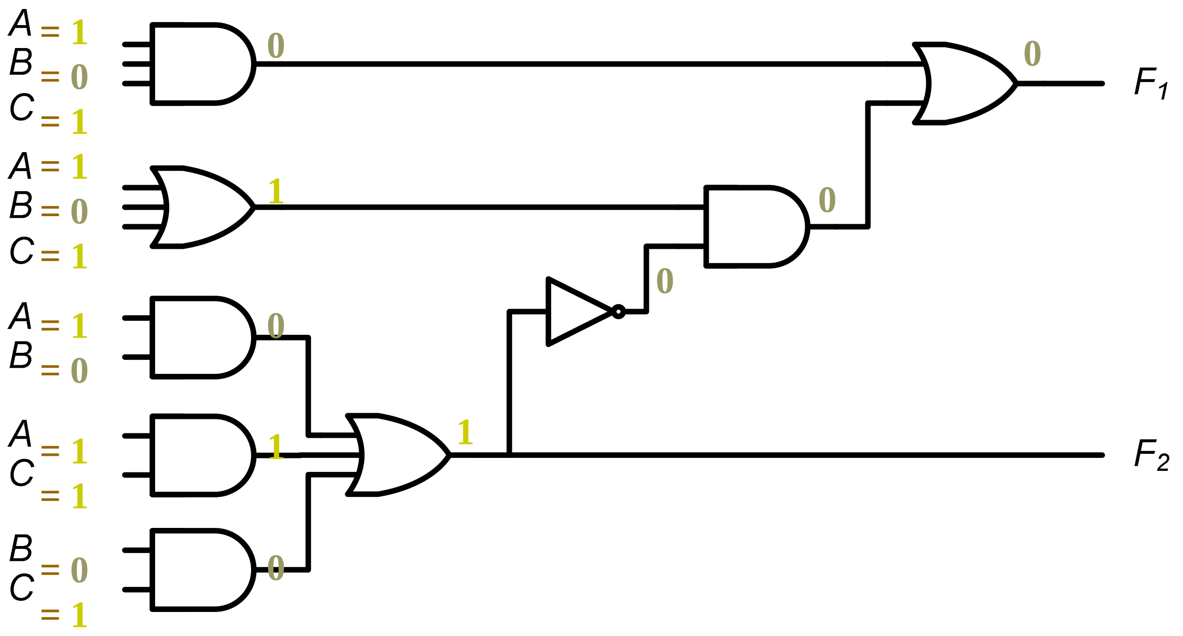
- Truth Table Approach



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

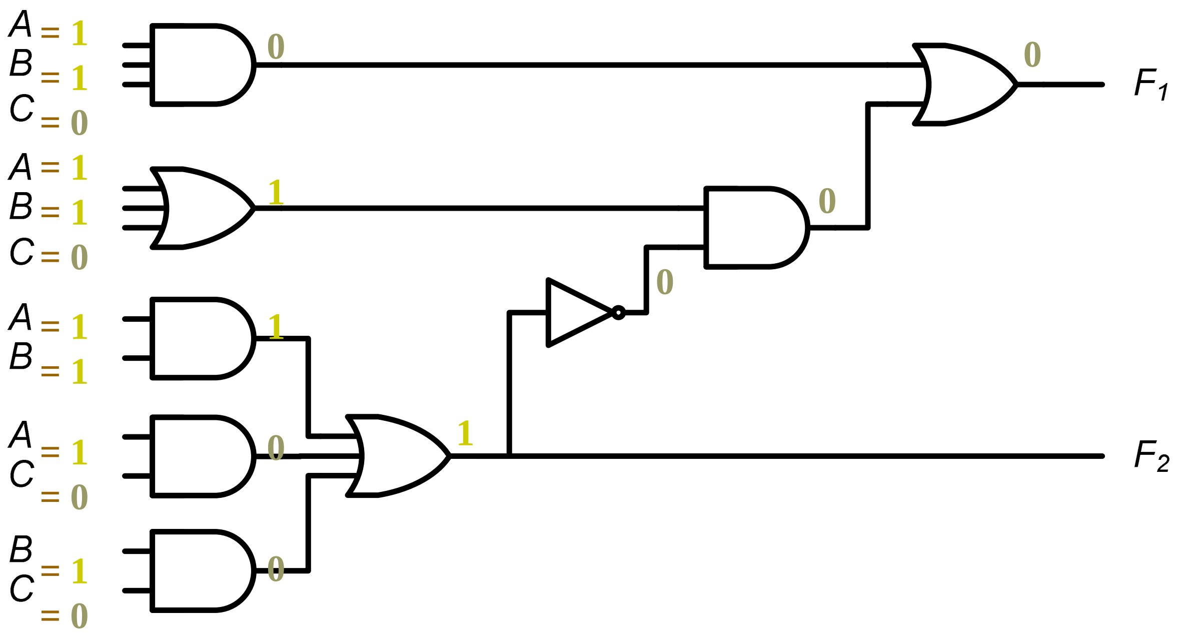
- Truth Table Approach



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

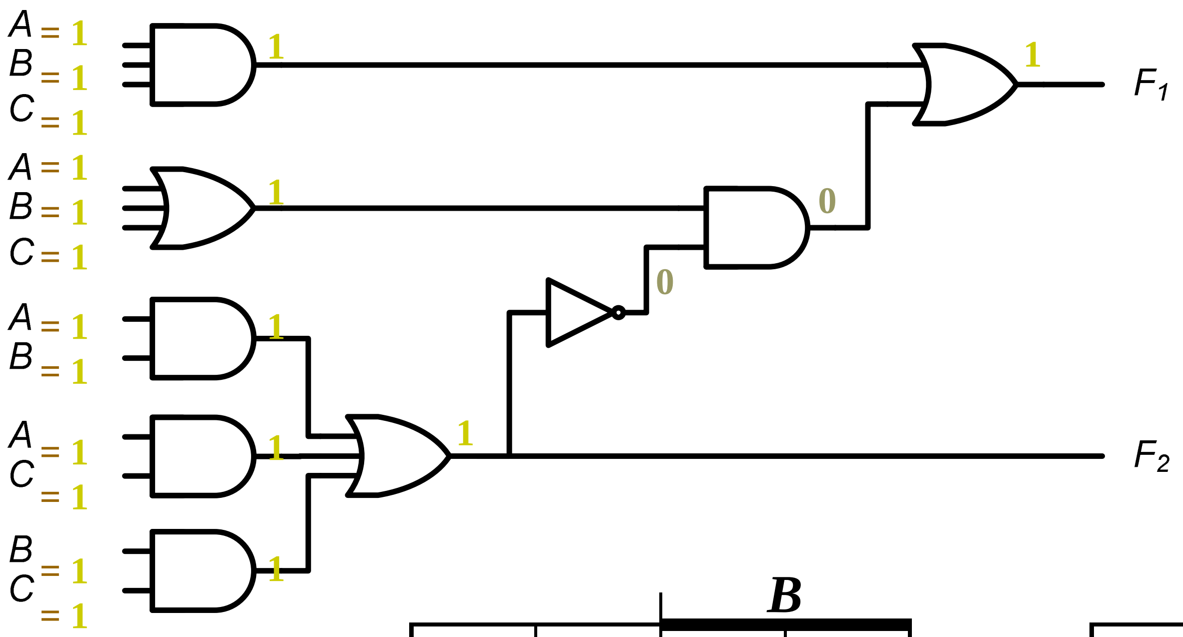
- Truth Table Approach



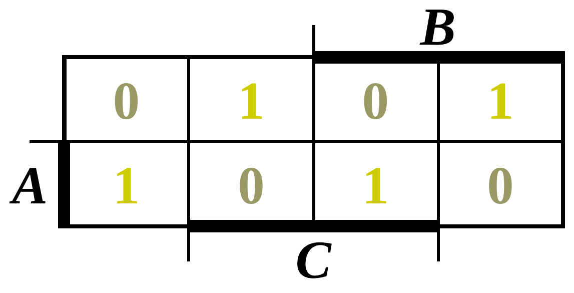
A B C	F_1	F_2
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	0	1
1 0 0	1	0
1 0 1	0	1
1 1 0	0	1

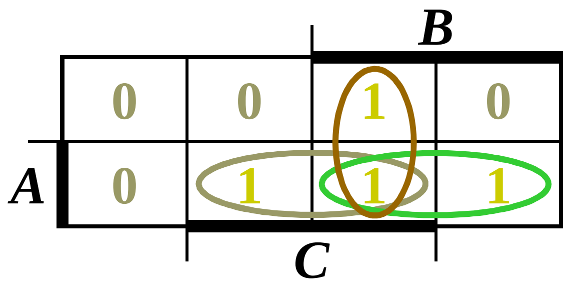
Analysis Procedure

- 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



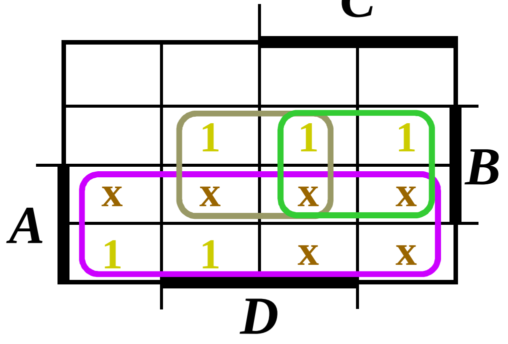


$F_1 = AB'C' + A'BC' + A'B'C + ABC$ $F_2 = AB + AC + BC$

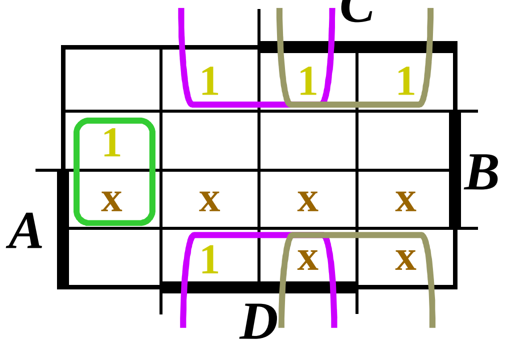
Design Procedure

- BCD-to-Excess 3 Converter

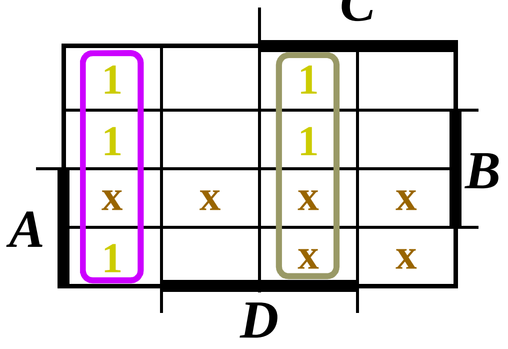
A B C D	w x y z
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0
1 0 1 0	x x x x
1 0 1 1	x x x x
1 1 0 0	x x x x
1 1 0 1	x x x x
1 1 1 0	x x x x
1 1 1 1	x x x x



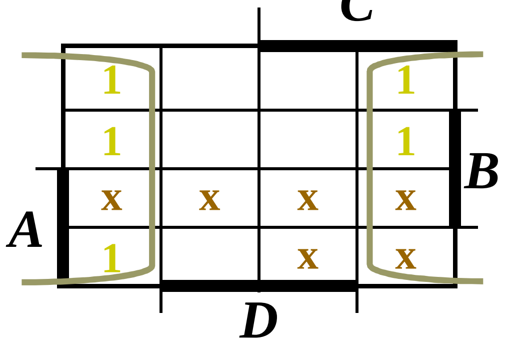
w = *A*+*BC*+*BD*



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



y = *C'D'*+*CD*

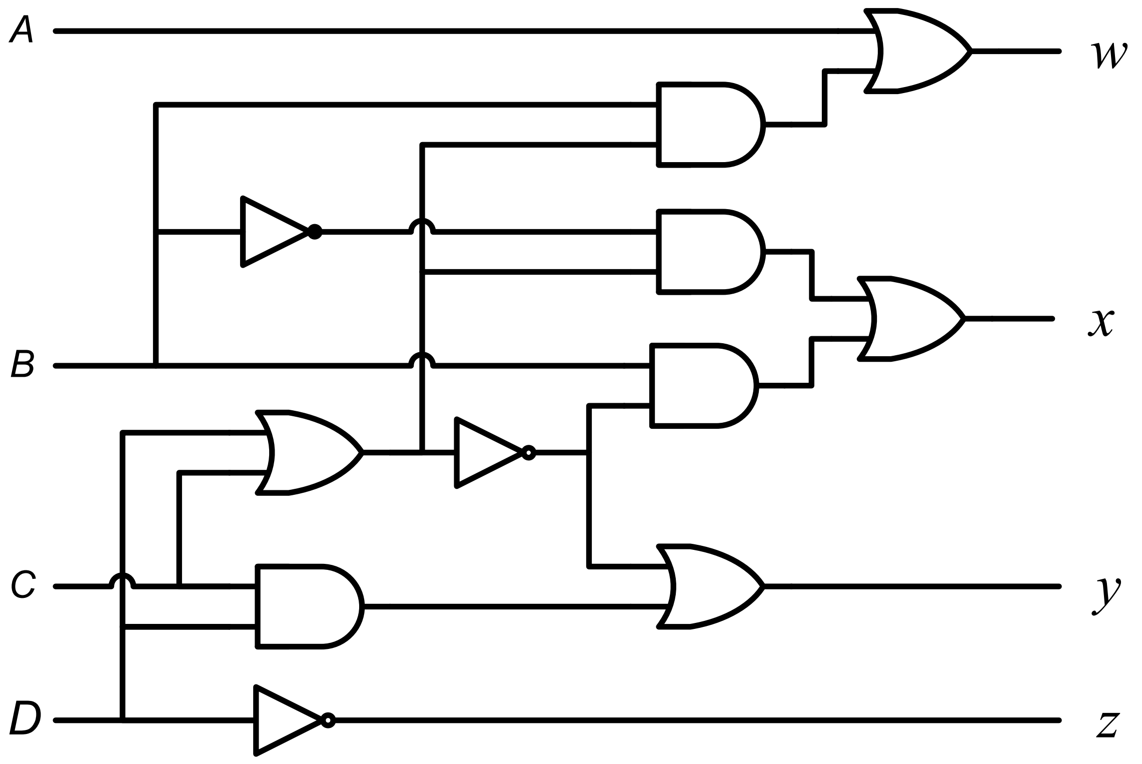


z = *D'*

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



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

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

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

$z = D'$

MSI logic circuits(Chapter 9 of T1)

- Digital systems obtain data and information continuously operated on in some manner:
  - *Decoding/encoding.*
  - *Multiplexing/demultiplexing,.*
  - *Comparison; Code conversion;*
- These and other operations have been facilitated by the availability of numerous ICs in the MSI (medium-scale-integration) category.

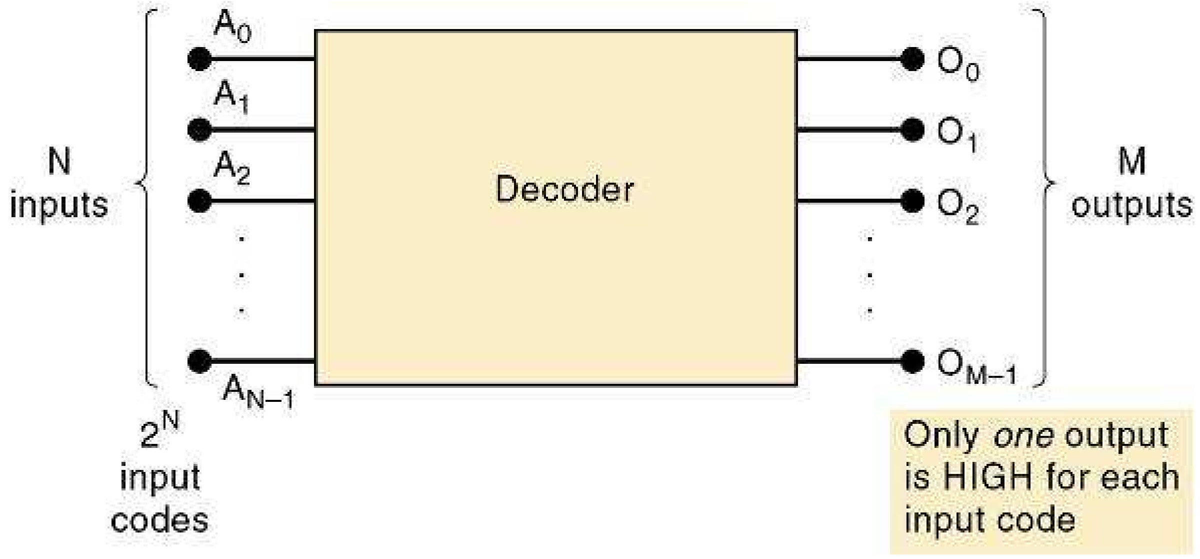
Decoders

- Decoders are used when an output or a group of outputs is to be activated only on the occurrence of a specific combination of input levels.
  - Often provided by outputs of a counter or a register.

Decoders

- A **decoder** accepts a set of inputs that represents a binary number—activating only the output that corresponds to the input number.

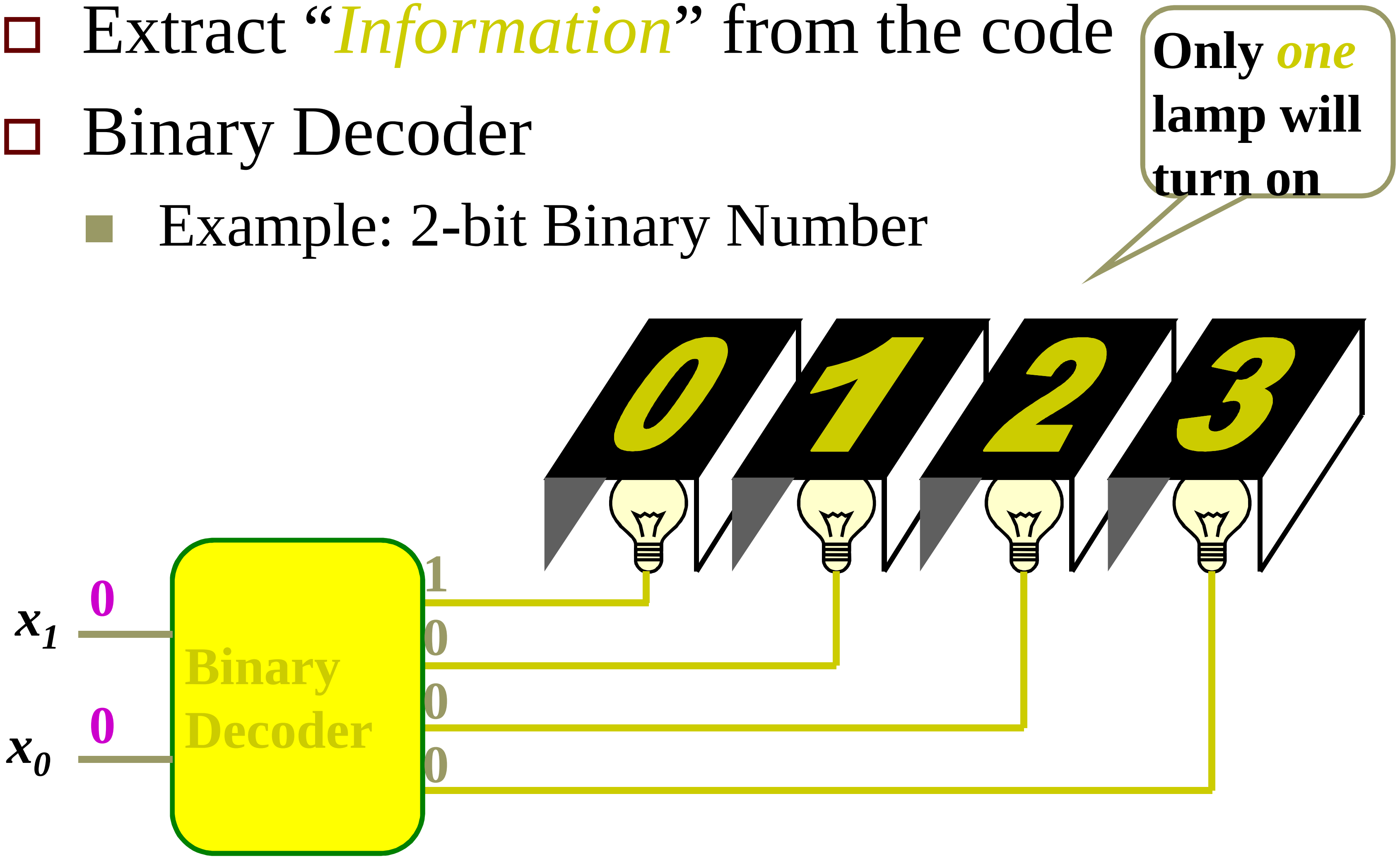
For each of these input combinations, only one of the *M* outputs will be active (HIGH); all the other outputs are LOW.



Many decoders are designed to produce active-LOW outputs, where only the selected output is LOW while all others are HIGH.

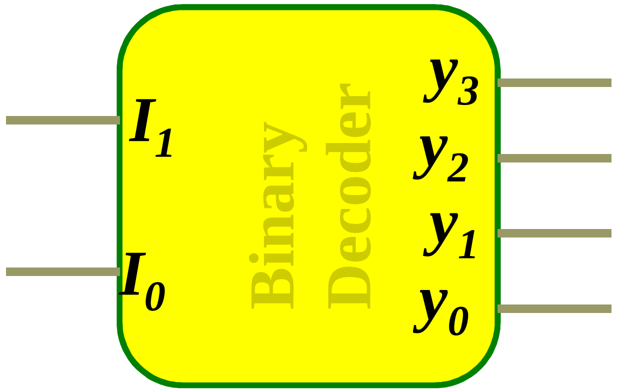
Decoders

- Extract "*Information*" from the code
- Binary Decoder
  - Example: 2-bit Binary Number

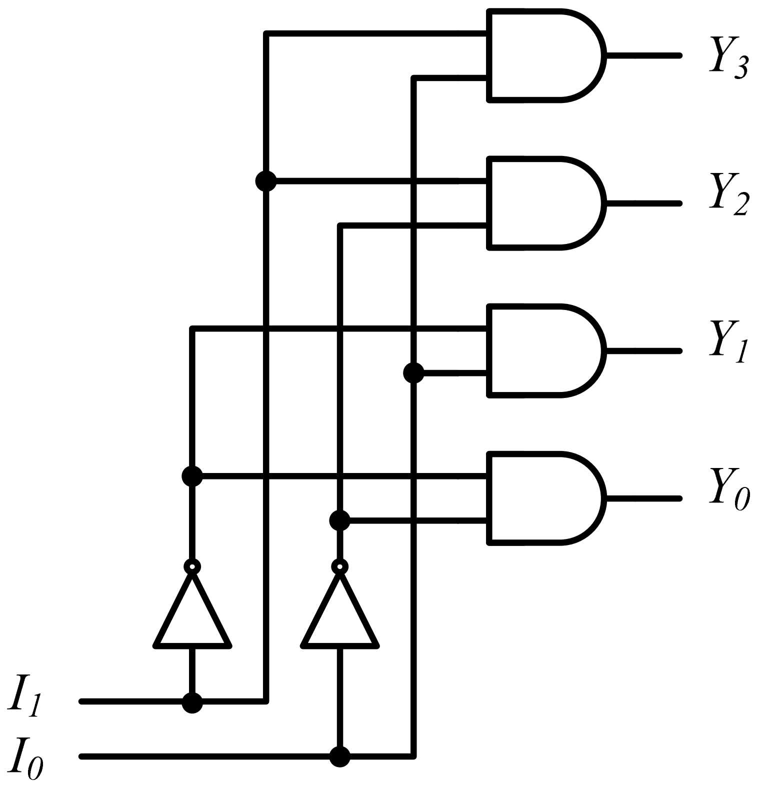


Decoders

- 2-to-4 Line Decoder



I_1 I_0	Y_3	Y_2	Y_1	Y_0
0 0	0	0	0	1
0 1	0	0	1	0
1 0	0	1	0	0
1 1	1	0	0	0

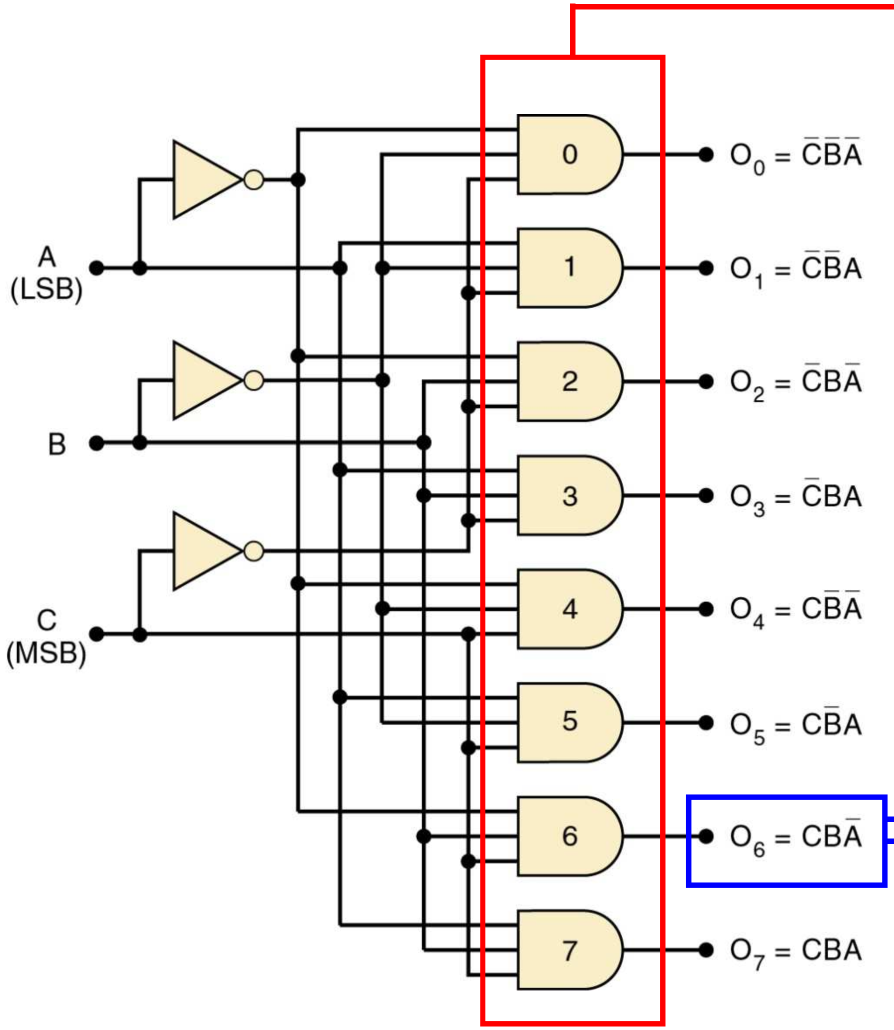


$Y_3 = I_1\, I_0$ $\qquad$ $Y_2 = I_1\, \bar{I}_0$

$Y_1 = \bar{I}_1\, I_0$ $\qquad$ $Y_0 = \bar{I}_1\, \bar{I}_0$

Decoders

Circuitry for a decoder with three inputs and 8 outputs.


A (LSB)
B
C (MSB)
0
1
2
3
4
5
6
7
$O_0 = \overline{C}\overline{B}\overline{A}$
$O_1 = \overline{C}\overline{B}A$
$O_2 = \overline{C}B\overline{A}$
$O_3 = \overline{C}BA$
$O_4 = C\overline{B}\overline{A}$
$O_5 = C\overline{B}A$
$O_6 = CB\overline{A}$
$O_7 = CBA$

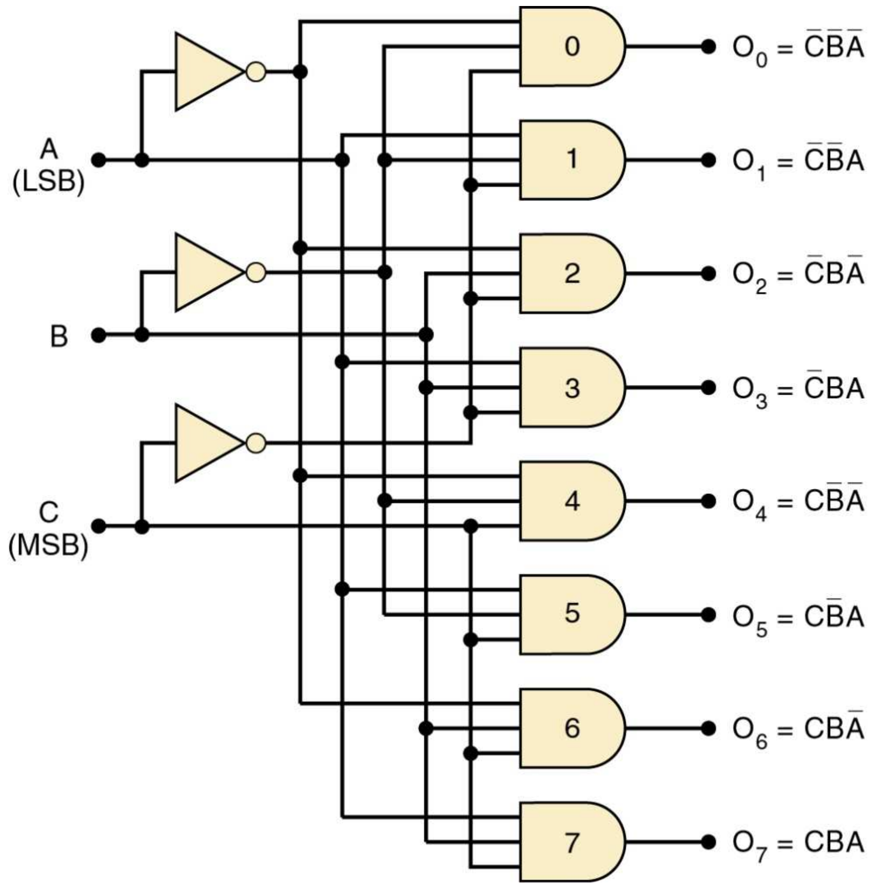
It uses all **AND** gates, so outputs are active-HIGH

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

Output O6 goes HIGH only when *CBA* 110

Decoders

Circuitry for a decoder with three inputs and 8 outputs.



This can be called a *3-line-to-8-line decoder*—it has three input lines and eight output lines.

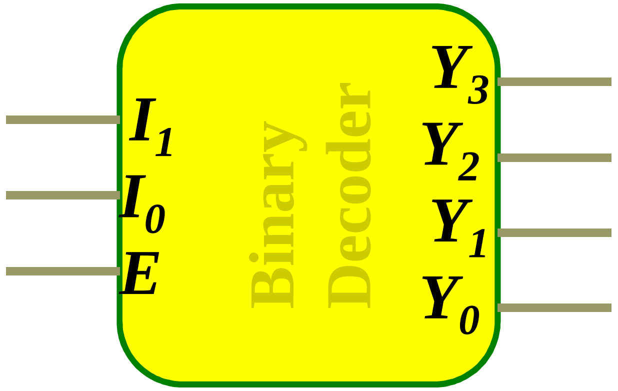
Also referred to as a *1-of-8 decoder*—only 1 of the 8 outputs is activated at one time.

Decoders

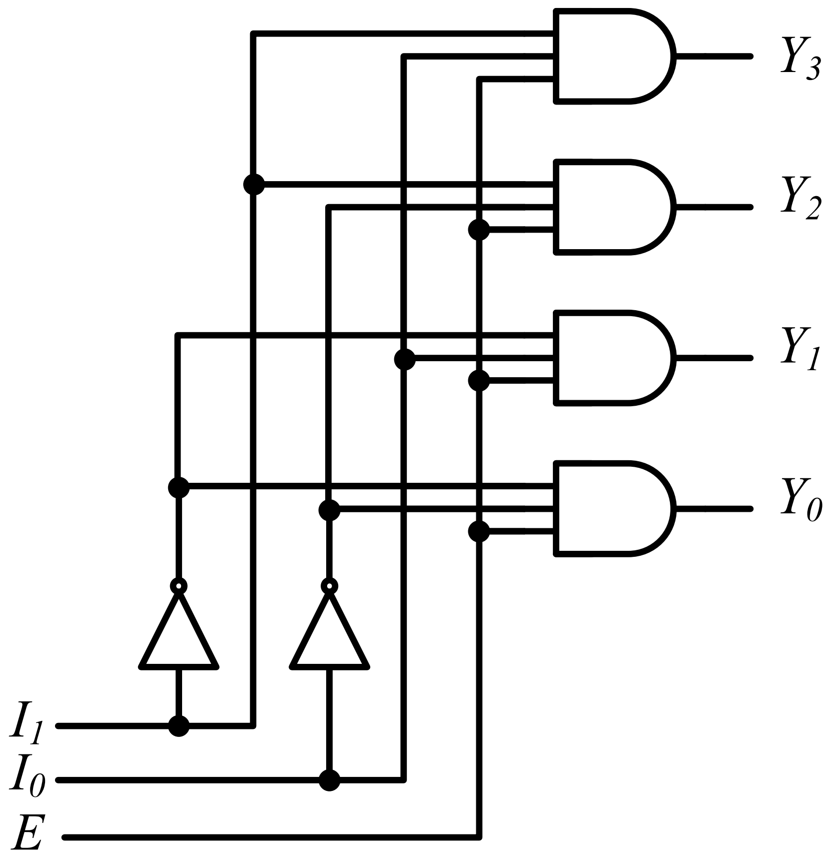
- Some decoders have one or more enable inputs used to control the operation of the decoder.
  - The decoder is enabled only if *ENABLE* is HIGH.
- With common *ENABLE* line connected to a fourth input of each gate:
  - If *ENABLE* is HIGH, the decoder functions normally.
    - *A*, *B*, *C* input will determine which output is HIGH.
  - If *ENABLE* is LOW, *all* outputs will be forced LOW.
    - *Regardless* of the levels at the *A*, *B*, *C* inputs.

Decoders

- "*Enable*" Control



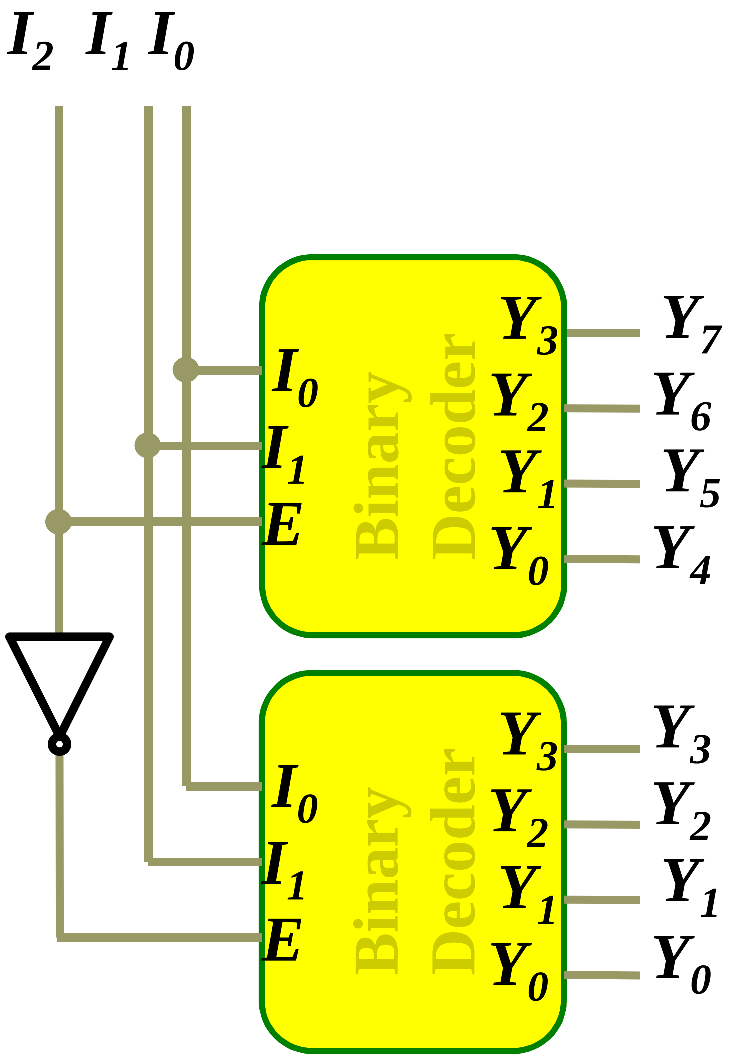
E	I_1 I_0	Y_3 Y_2 Y_1 Y_0
0	x x	0 0 0 0
1	0 0	0 0 0 1
1	0 1	0 0 1 0
1	1 0	0 1 0 0
1	1 1	1 0 0 0



Decoders

- Expansion

I_2	I_1	I_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_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

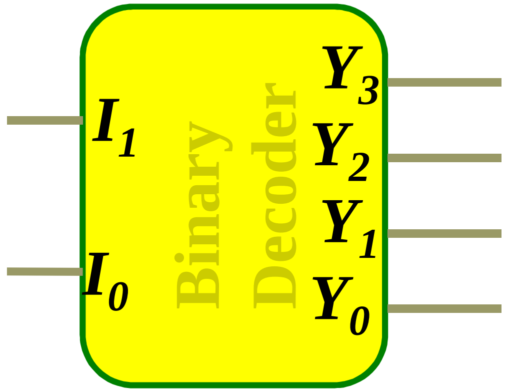


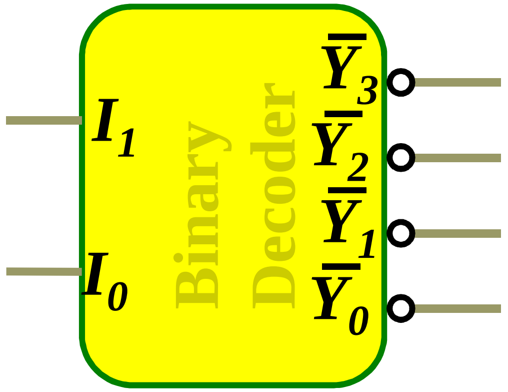
Decoders

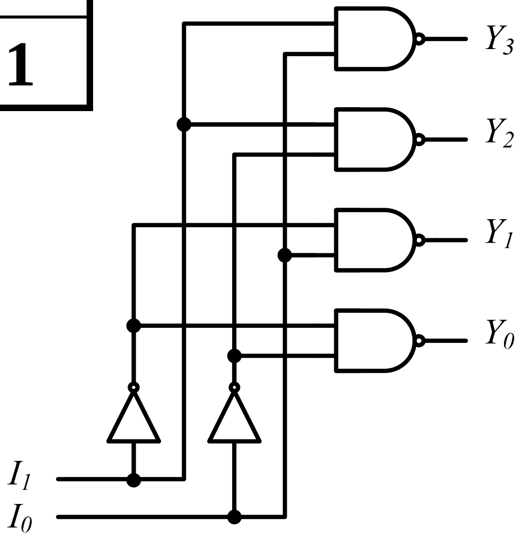
- Active-High / Active-Low

I_1 I_0	Y_3	Y_2	Y_1	Y_0
0 0	0	0	0	1
0 1	0	0	1	0
1 0	0	1	0	0
1 1	1	0	0	0

I_1 I_0	Y_3	Y_2	Y_1	Y_0
0 0	1	1	1	0
0 1	1	1	0	1
1 0	1	0	1	1
1 1	0	1	1	1







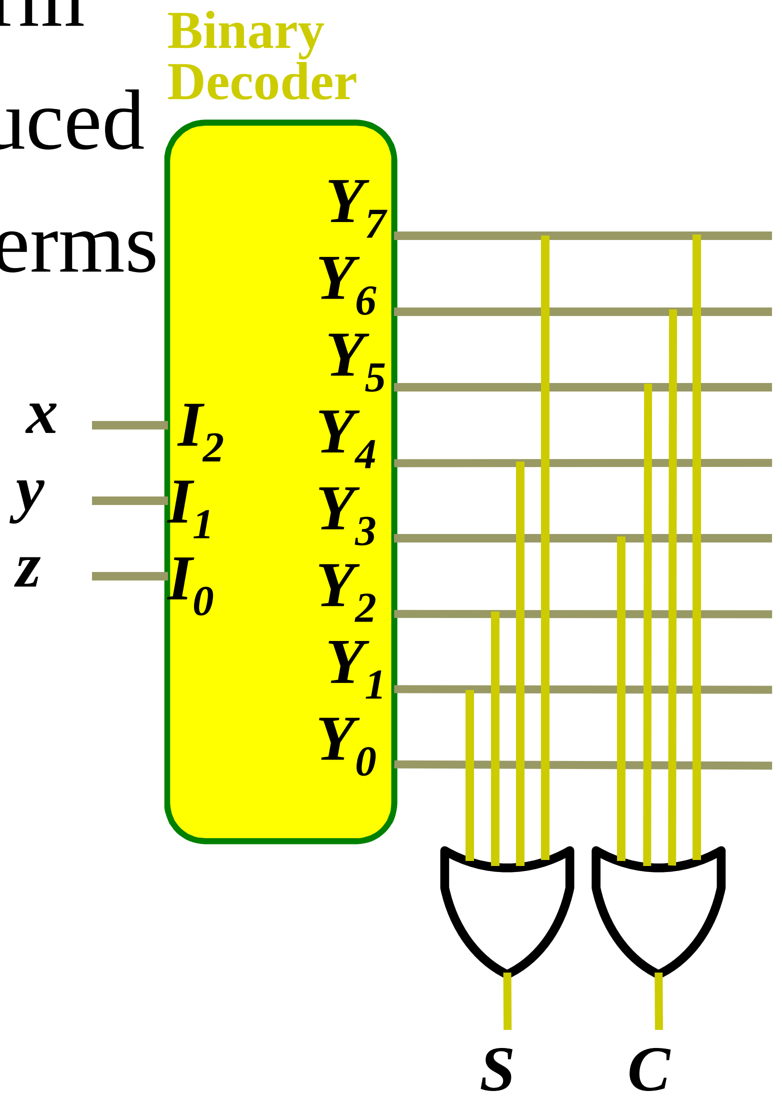
Implementation Using Decoders

- Each output is a minterm
- All minterms are produced
- Sum the required minterms

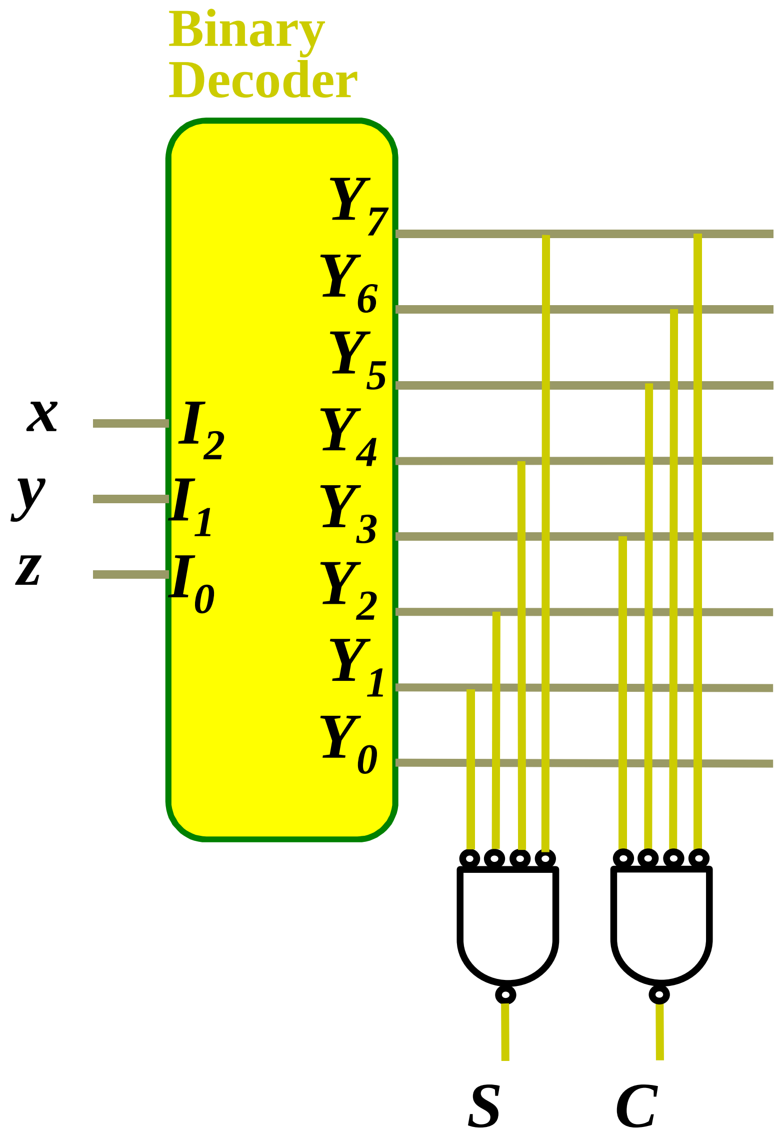
Example: Full Adder

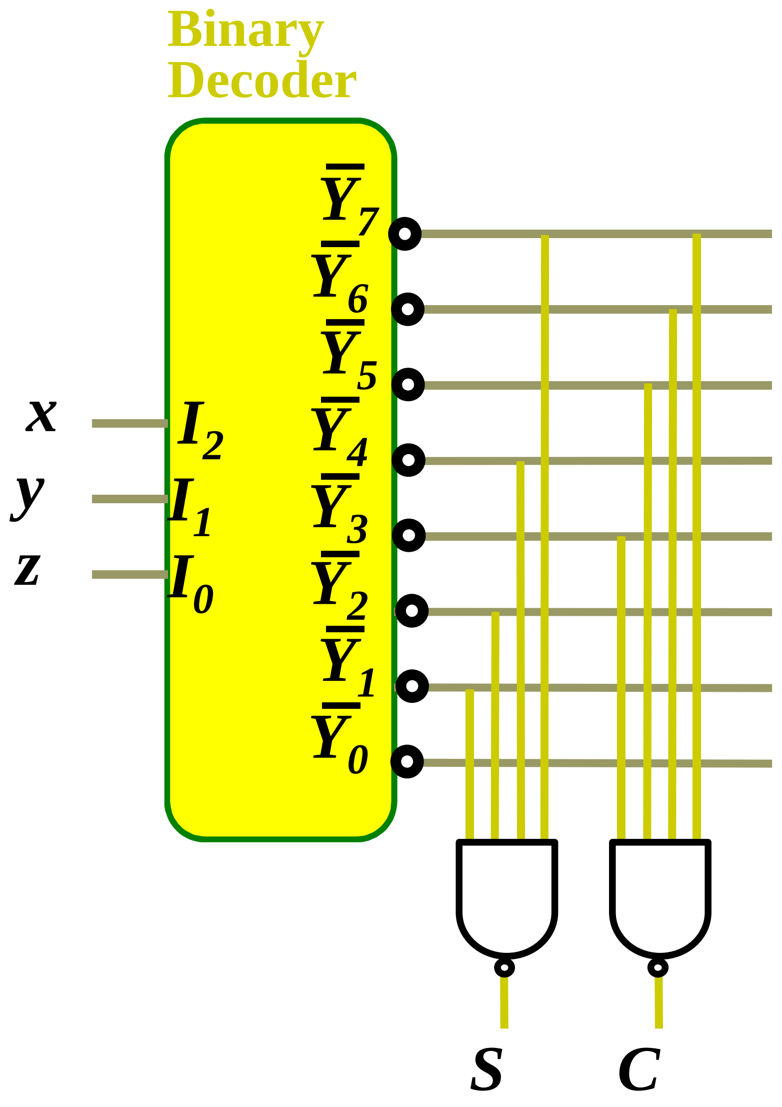
$S(x, y, z) = \sum(1, 2, 4, 7)$

$C(x, y, z) = \sum(3, 5, 6, 7)$



Implementation Using Decoders





Example:-Design a magnitude comparator circuit for 2-bit binary numbers *A=A1A0 and B=B1B0. The* outputs are *F, G, and H, where F is 1 if A>B, G is 1 if A=B, and H is 1 if A<B.*

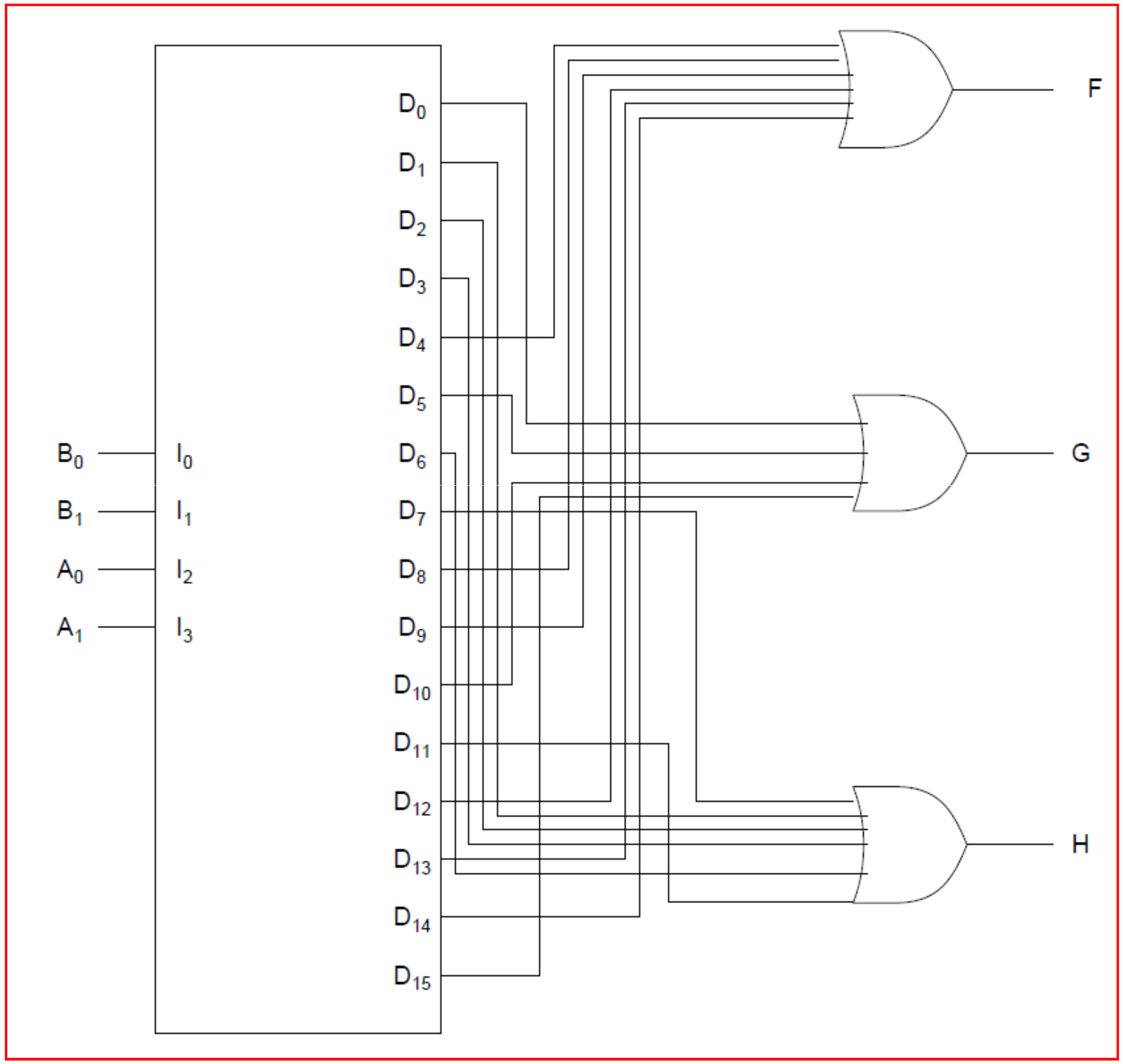
A_1	A_0	B_1	B_0	F(A>B)	G(A=B)	H(A<B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

$F = \Sigma(4,8,9,12,13,14)$

$G=\Sigma(0,5,10,15)$

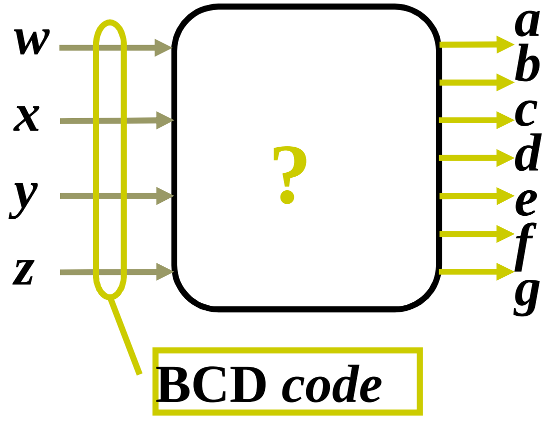
$H=\Sigma(1,2,3,6,7,11)$

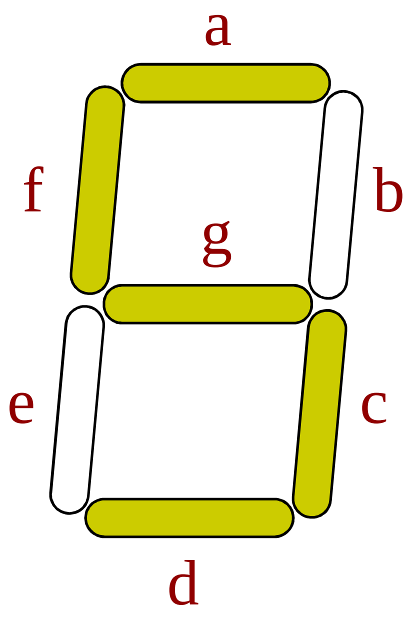
Implement your comparator design using a 4-to-16 line decoder

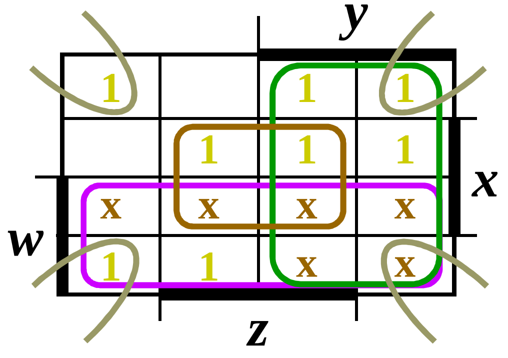


Seven-Segment Decoder

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 = . . .

c = . . .

d = . . .