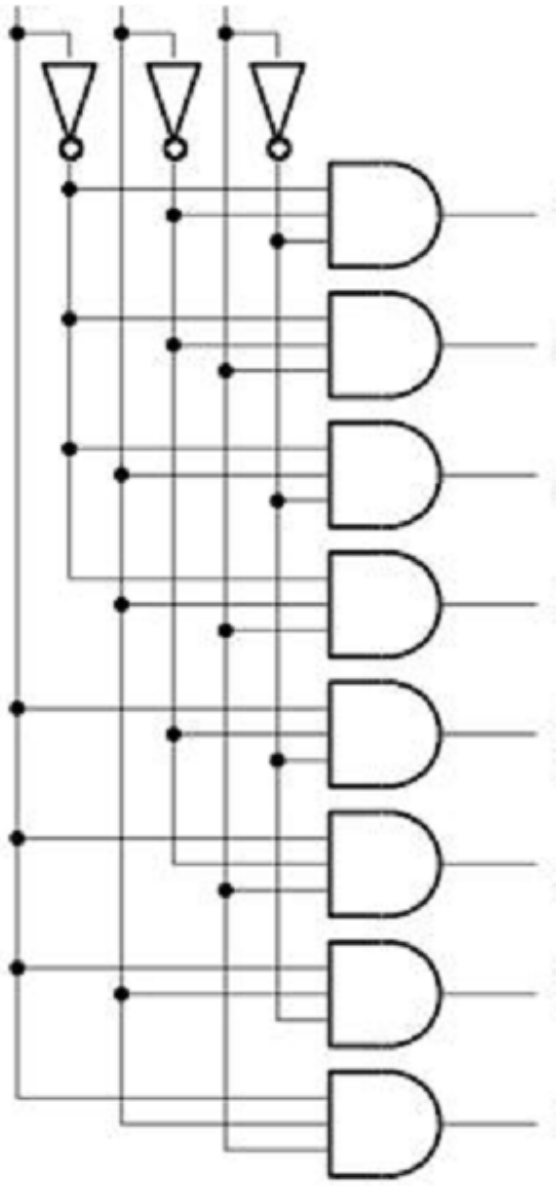
DECODERS

A *decoder* is a combinational circuit that converts binary information from *n* input lines to a maximum of 2^n unique output lines. The purpose of decoders is to generate the 2^n minterms of *n* input variables.

3-to-8 line decoder:



– logic diagram –

- The three inputs (x, y, z) are decoded into eight outputs (D_0 –D_7), each output representing one of the minterms of the three input variables.
- Only one output can be equal to 1 at any time.

Boolean Function Implementation using Decoders:

Since any Boolean function can be expressed in sum-of-minters form, a decoder can be used to generate the minterms and an OR gate to form the sum.

Ex: Implement the following functions with a 3×8 decoder and minimum number of external gates.

$F_1(x, y, z) = \sum m\ (0, 2, 4)$
$F_2(x, y, z) = \sum m\ (0, 5, 6, 7)$
$F_3(x, y, z) = \sum m\ (1, 6, 7)$

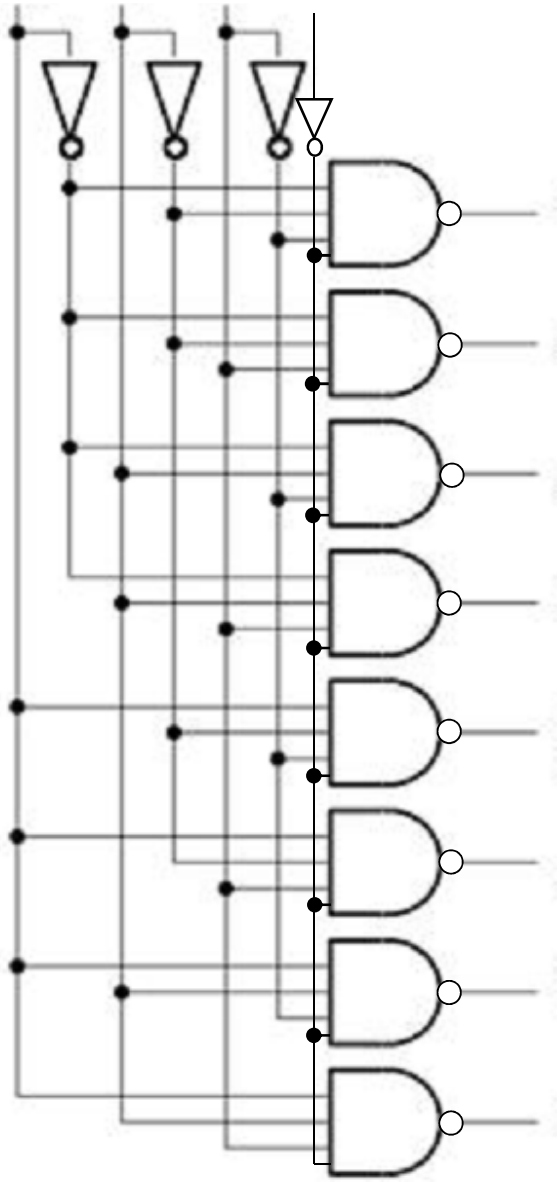
Ex: Implement the full-adder with a decoder.

Some IC decoders are constructed with NAND gates by replacing AND gates;

- ⇨ minterms are generated in their complemented form.

Also, most IC decoders include one or more enable inputs for making the decoder ACTIVE or INACTIVE. The decoder will operate if the enable is activated.

3-to-8 line decoder constructed with NAND gates and an enable input:



─ logic diagram ─

When E=1, the decoder is disabled and all outputs are 1.
When E=0, the decoder is enabled.

Ex: Implement the following functions with a 3×8 decoder constructed with NAND gates and minimum number of 2-input external gates.

$F_1(A, B, C) = \sum m\ (2, 4, 7)$
$F_2(A, B, C) = \sum m\ (0, 3)$
$F_3(A, B, C) = \sum m\ (0, 2, 3, 4, 7)$

Ex: Implement the full-adder with a decoder constructed with NAND gates using:

a) multi-input external gates
b) 2-input external gates

Decoders can be constructed together to form a larger decoder circuit.

Ex: Construct a 4×16 decoder using two 3×8 decoders with active high enable inputs.

Ex: Implement F(A, B, C, D) = $\sum$ m (1, 3, 4, 5, 6, 9, 12, 14, 15) by using 3×8 decoder(s).