4 Canonical and standard forms

- A Boolean (logic) function can be expressed in a variety of algebraic forms. For example

$$y = c \cdot \bar{a} + c \cdot b = c(\bar{a} + b) = c(\bar{c} + b + \bar{a})$$

- Each algebraic form entails specific gate implementation.
- A Boolean function can be uniquely described by its **truth table**, or in one of the **canonical forms**.
- **Two dual canonical forms** of a Boolean function are available:
  - The sum of minterms (SoM) form
  - The product of maxterms (PoM) form.
- A **minterm is a product of all variables** taken either in their direct or complemented form
- A **maxterm is a sum of all variables** taken either in their direct or complemented form

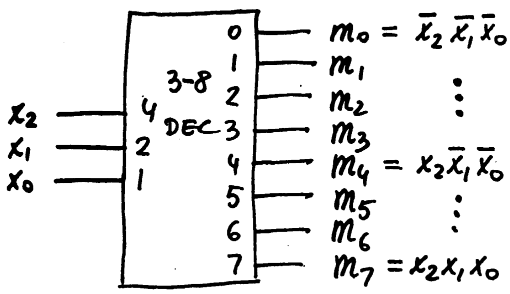
4.1 Minterms. n-to-2^n Decoders

- Consider, for example, all possible logic products of three variables $\mathbf{x} = (x_2, x_1, x_0)$
- There are $2^3 = 8$ different **minterms** that can be written in the form $m_i = \tilde{x}_2 \cdot \tilde{x}_1 \cdot \tilde{x}_0$ where $\tilde{x}$ represents either variable x or its complement $\bar{x}$

All three-variable minterms are listed in the following table:

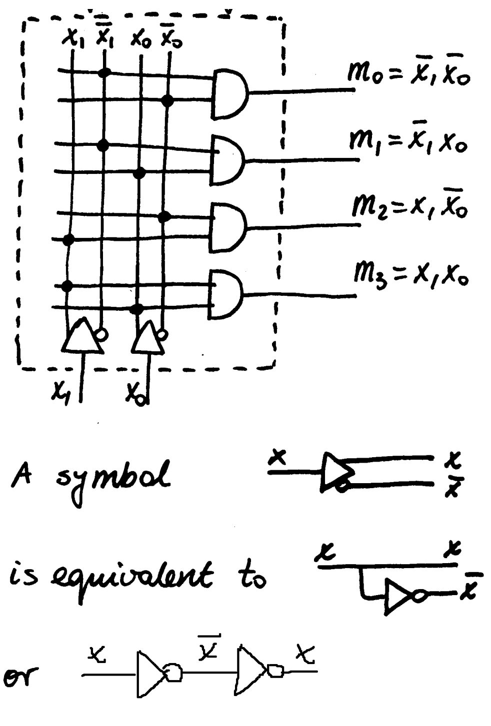
$\mathbf{x}$	x_2	x_1	x_0	Minterms	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	0	$m_0 = \bar{x}_2 \cdot \bar{x}_1 \cdot \bar{x}_0$	1	0	0	0	0	0	0	0
1	0	0	1	$m_1 = \bar{x}_2 \cdot \bar{x}_1 \cdot x_0$	0	1	0	0	0	0	0	0
2	0	1	0	$m_2 = \bar{x}_2 \cdot x_1 \cdot \bar{x}_0$	0	0	1	0	0	0	0	0
3	0	1	1	$m_3 = \bar{x}_2 \cdot x_1 \cdot x_0$	0	0	0	1	0	0	0	0
4	1	0	0	$m_4 = x_2 \cdot \bar{x}_1 \cdot \bar{x}_0$	0	0	0	0	1	0	0	0
5	1	0	1	$m_5 = x_2 \cdot \bar{x}_1 \cdot x_0$	0	0	0	0	0	1	0	0
6	1	1	0	$m_6 = x_2 \cdot x_1 \cdot \bar{x}_0$	0	0	0	0	0	0	1	0
7	1	1	1	$m_7 = x_2 \cdot x_1 \cdot x_0$	0	0	0	0	0	0	0	1

The logic circuit that generates all minterms is called an n-to-2^n decoder:



Example: Logic structure of a 2-to-4 decoder

x	x_1	x_0	Minterms	m_0	m_1	m_2	m_3
0	0	0	$m_0 = \bar{x}_1 \cdot \bar{x}_0$	1	0	0	0
1	0	1	$m_1 = \bar{x}_1 \cdot x_0$	0	1	0	0
2	1	0	$m_2 = x_1 \cdot \bar{x}_0$	0	0	1	0
3	1	1	$m_3 = x_1 \cdot x_0$	0	0	0	1



4.2 The Sum-of-Minterms (SoM) canonical form of a logic function

- Any logic function y of n variables can be expressed as the **logic sum of** products of **minterms** and the respective values of the function, that is:

$$y = f(x_{n-1}, \ldots, x_0) = \sum_{i=0}^{2^n-1} y_i \cdot m_i \tag{4.1}$$

- It is clearly equivalent to the sum of minterms for which the values of the function are 1, say, $y_j = 1$

$$y = f(x_{n-1}, \ldots, x_0) = \sum_{\text{for all } j \text{ such that } y_j = 1} m_j \tag{4.2}$$

Example of a 3-variable function

x	x_2	x_1	x_0	y	m_j
0	0	0	0	0	
1	0	0	1	0	
2	0	1	0	1	m_2
3	0	1	1	0	
4	1	0	0	1	m_4
5	1	0	1	1	m_5
6	1	1	0	0	
7	1	1	1	1	m_7

$$\begin{aligned} y &= 0 \cdot m_0 + 0 \cdot m_1 + 1 \cdot m_2 + 0 \cdot m_3 + 1 \cdot m_4 + 1 \cdot m_5 + 0 \cdot m_6 + 1 \cdot m_7 \\ &= m_2 + m_4 + m_5 + m_7 \\ &= \sum(2, 4, 5, 7) \quad \text{— a commonly used short notation} \\ &= \bar{x}_2 \cdot x_1 \cdot \bar{x}_0 + x_2 \cdot \bar{x}_1 \cdot \bar{x}_0 + x_2 \cdot \bar{x}_1 \cdot x_0 + x_2 \cdot x_1 \cdot x_0 \end{aligned}$$

4.3 Decoder-based implementation of a logic function. Solution 1.

Any logic function can be implemented using a decoder and a logic OR gate.

4.4 Maxterms

- A logic sum (OR) of all variables taken in their direct or complemented form is called a **Maxterm**, M_i.
- A maxterm is a complement of an equivalent minterm

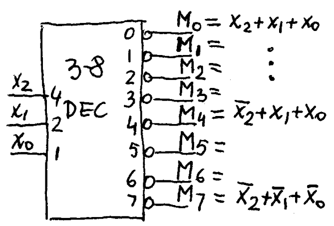
$$M_i = \overline{m}_i = \overline{\tilde{x}_{n-1} \cdot \ldots \tilde{x}_0} = \bar{\tilde{x}}_{n-1} + \ldots \bar{\tilde{x}}_0$$

where $\bar{\tilde{x}}$ represents either variable x or its complement $\bar{x}$

For example, all three-variable Maxterms are listed in the following table:

x	x_2	x_1	x_0	Maxterms	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	0	$M_0 = x_2 + x_1 + x_0$	0	1	1	1	1	1	1	1
1	0	0	1	$M_1 = x_2 + x_1 + \bar{x}_0$	1	0	1	1	1	1	1	1
2	0	1	0	$M_2 = x_2 + \bar{x}_1 + x_0$	1	1	0	1	1	1	1	1
3	0	1	1	$M_3 = x_2 + \bar{x}_1 + \bar{x}_0$	1	1	1	0	1	1	1	1
4	1	0	0	$M_4 = \bar{x}_2 + x_1 + x_0$	1	1	1	1	0	1	1	1
5	1	0	1	$M_5 = \bar{x}_2 + x_1 + \bar{x}_0$	1	1	1	1	1	0	1	1
6	1	1	0	$M_6 = \bar{x}_2 + \bar{x}_1 + x_0$	1	1	1	1	1	1	0	1
7	1	1	1	$M_7 = \bar{x}_2 + \bar{x}_1 + \bar{x}_0$	1	1	1	1	1	1	1	0

The logic circuit that generates all Maxterms is also called an n-to-2^n decoder:



Note that decoder producing maxterms has circles at its outputs

4.5 The Product-of-Maxterms (PoM) canonical form of a logic function

- Using the duality principle we can re-write eqn (4.1) in the following form:
- Any logic function y of n variables can be expressed as the **logic product of** sums of **maxterms** and the respective values of the function, that is:

$$y = f(x_{n-1}, \ldots, x_0) = \prod_{i=0}^{2^n-1} (y_i + M_i) \tag{4.3}$$

- It is clearly equivalent to the product of Maxterms for which the values of the function are 0, say, $y_j = 0$

$$y = f(x_{n-1}, \ldots, x_0) = \prod_{\text{for all } j \text{ such that } y_j=0} M_j \tag{4.4}$$

Example of a 3-variable function

x	x_2	x_1	x_0	y	m_j
0	0	0	0	0	M_0
1	0	0	1	0	M_1
2	0	1	0	1	
3	0	1	1	0	M_3
4	1	0	0	1	
5	1	0	1	1	
6	1	1	0	0	M_6
7	1	1	1	1	

$$\begin{aligned} y &= (0+M_0)(0+M_1)(1+M_2)(0+M_3)(1+M_4)(1+M_5)(0+M_6)(1+M_7) \\ &= M_0 \cdot M_1 \cdot M_3 \cdot M_6 \\ &= \prod(0,1,3,6) \quad \text{— a commonly used short notation} \\ &= (x_2+x_1+x_0)(x_2+x_1+\bar{x}_0)(x_2+\bar{x}_1+\bar{x}_0)(\bar{x}_2+\bar{x}_1+x_0) \end{aligned}$$

4.6 More on Decoder-based implementation of a logic function.

Using AND, NOR, OR and NAND gates.

4.7 Standard forms

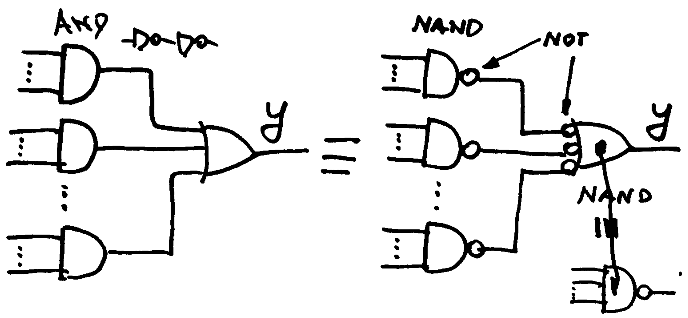
- Implementations of logic functions based on canonical forms are called **two-level implementations** (inverters of input variables are not counted)
- It is often possible to simplify canonical forms into standard forms:
- The Sum-of-Minterms (SoM) form can be simplified into a Sum-of-Products (SoP) form.
- The Product-of-Maxterm (PoM) form can be simplified into a Product-of-Sums (PoS) form

Examples:

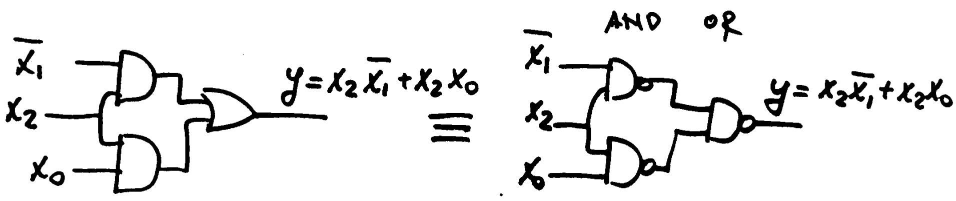
$$
\begin{aligned}
y_1 &= a \cdot \bar{b} + \bar{a} \cdot b \cdot c \\
y_2 &= (a + \bar{b})(\bar{a} + b + c)
\end{aligned}
$$

4.8 NAND and NOR based implementations

NAND implementation can be easily obtained from the **Sum-of-Products** form.

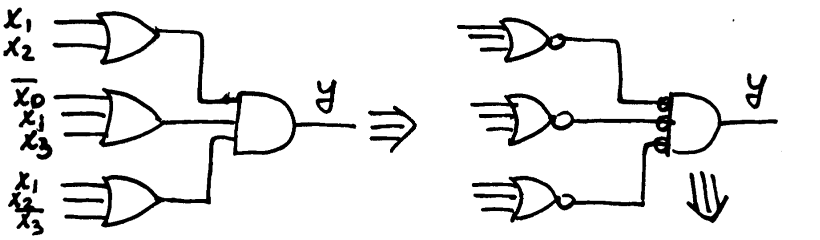


Example



NOR implementation can be easily obtained from the **Product-of-Sums** form.

EXAMPLE



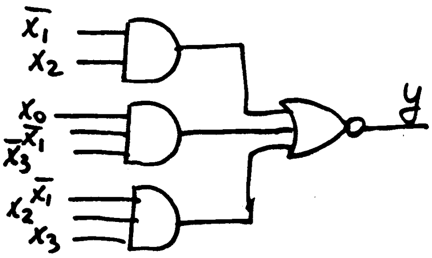
$$y = (x_1 + x_2)(\bar{x}_0 + x_1 + x_3)(x_1 + x_2 + \bar{x}_3)$$

Product-of-Sums form can also be implemented as an **Inverted-Sum-of-Products** form

EXAMPLE

$$y = (x_1 + \bar{x}_2)(\bar{x}_0 + x_1 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3)$$

$$= \overline{\bar{x}_1 \cdot x_2 + x_0 \cdot \bar{x}_1 \cdot \bar{x}_3 + \bar{x}_1 \cdot x_2 \cdot x_3}$$



Sum-of-Products form can also be implemented as an **Inverted-Product-of-Sums** form

EXAMPLE

$$y = x_2\bar{x}_1 + \bar{x}_2 x_0 = \overline{(\bar{x}_2 + x_1)(x_2 + \bar{x}_0)}$$