IC CHARACTERSTICS

- **Fan Out**
- **Propagation Delay**
- **Noise Margin**
- **Power Dissipation**

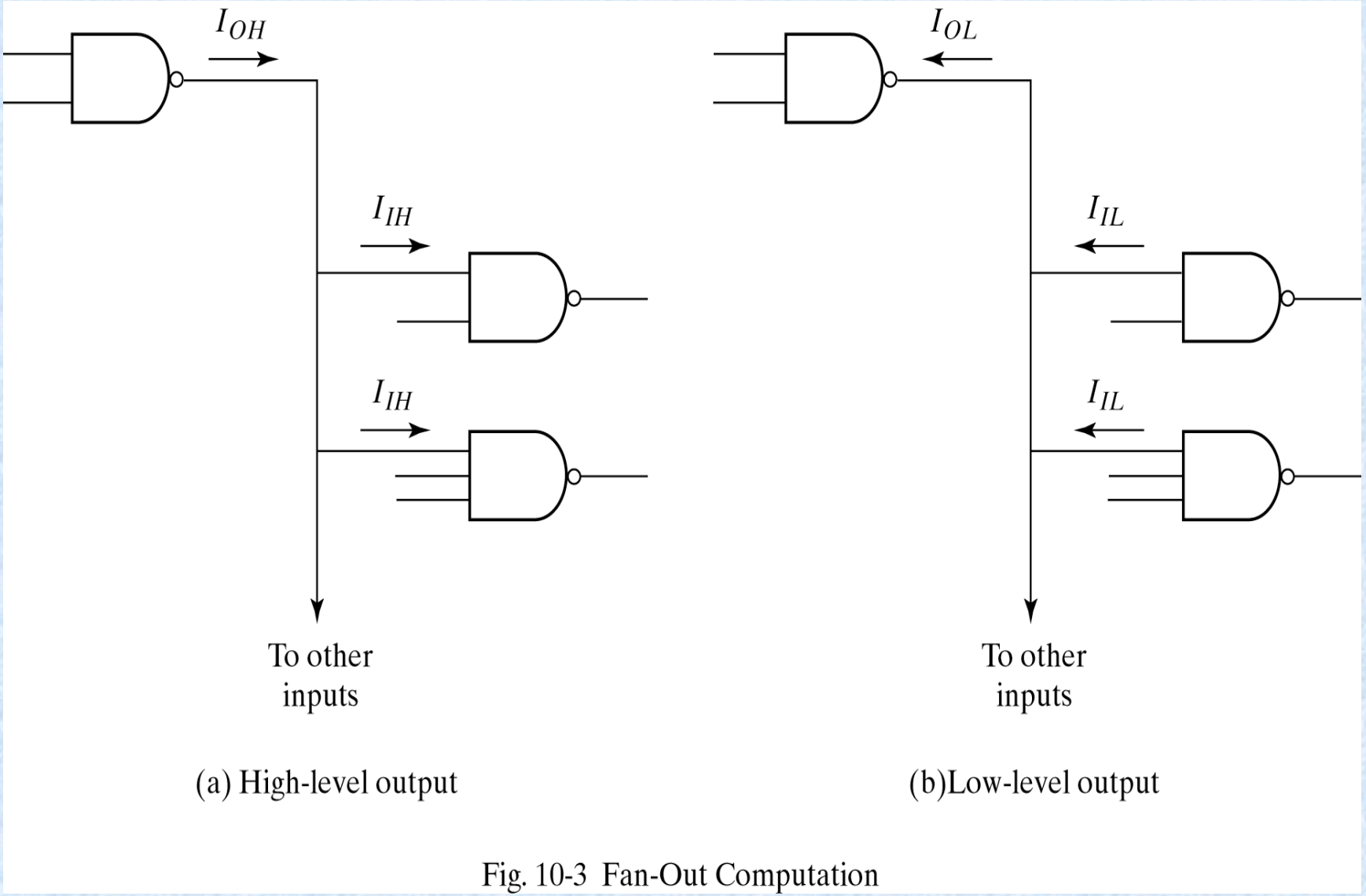


Fig. 10-3 Fan-Out Computation

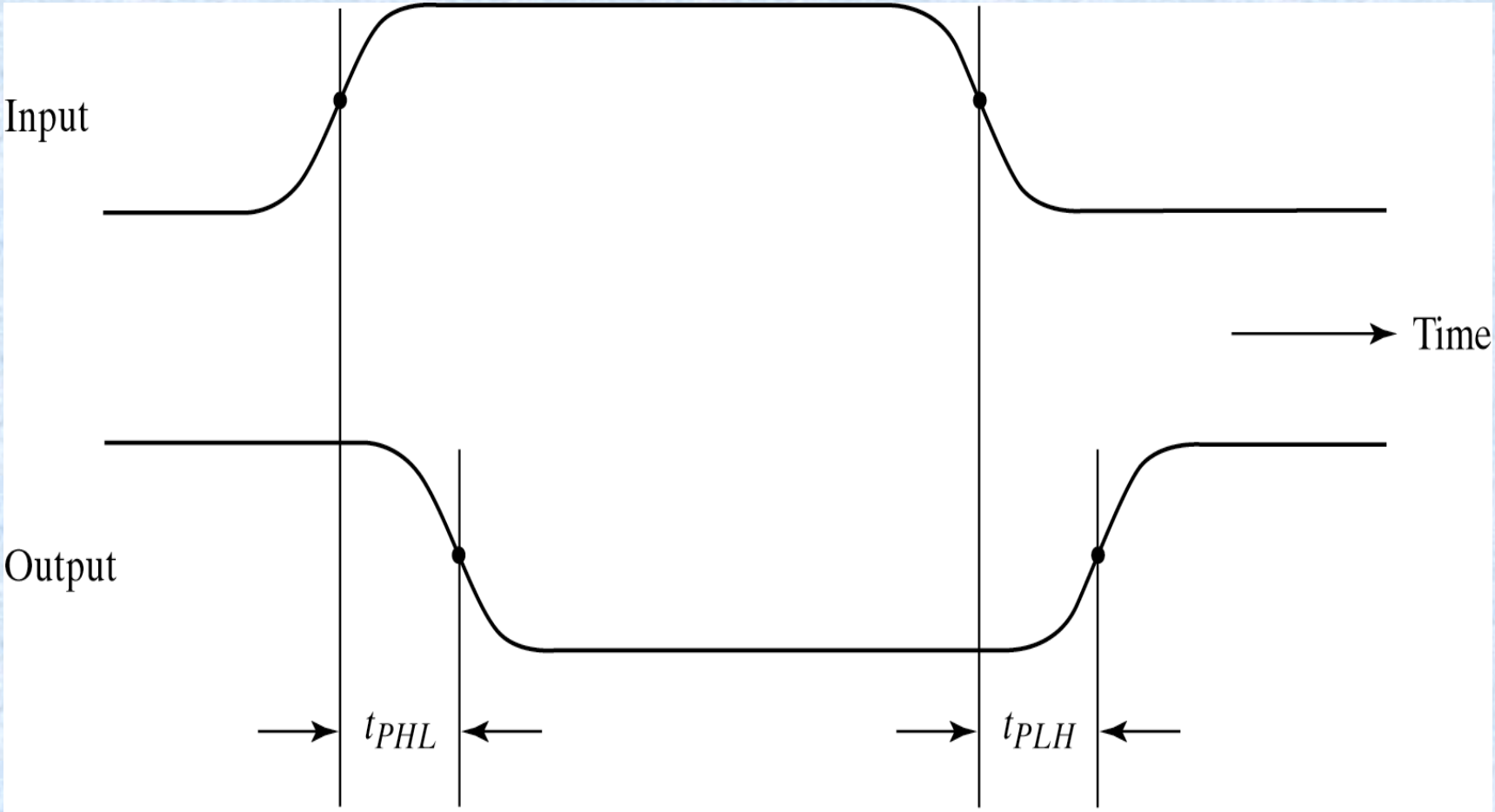


Fig. 10-4 Measurement of Propagation Delay

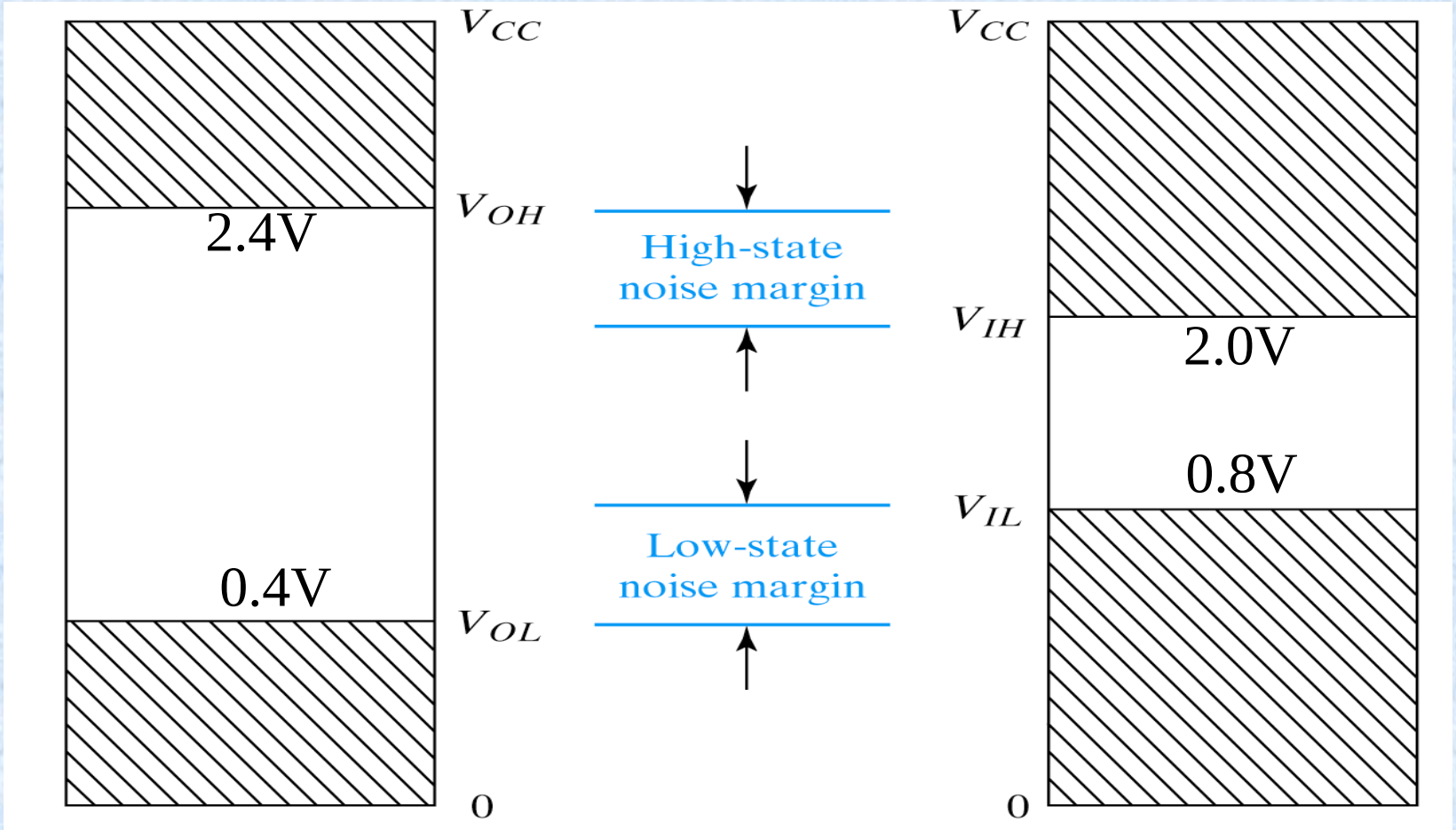
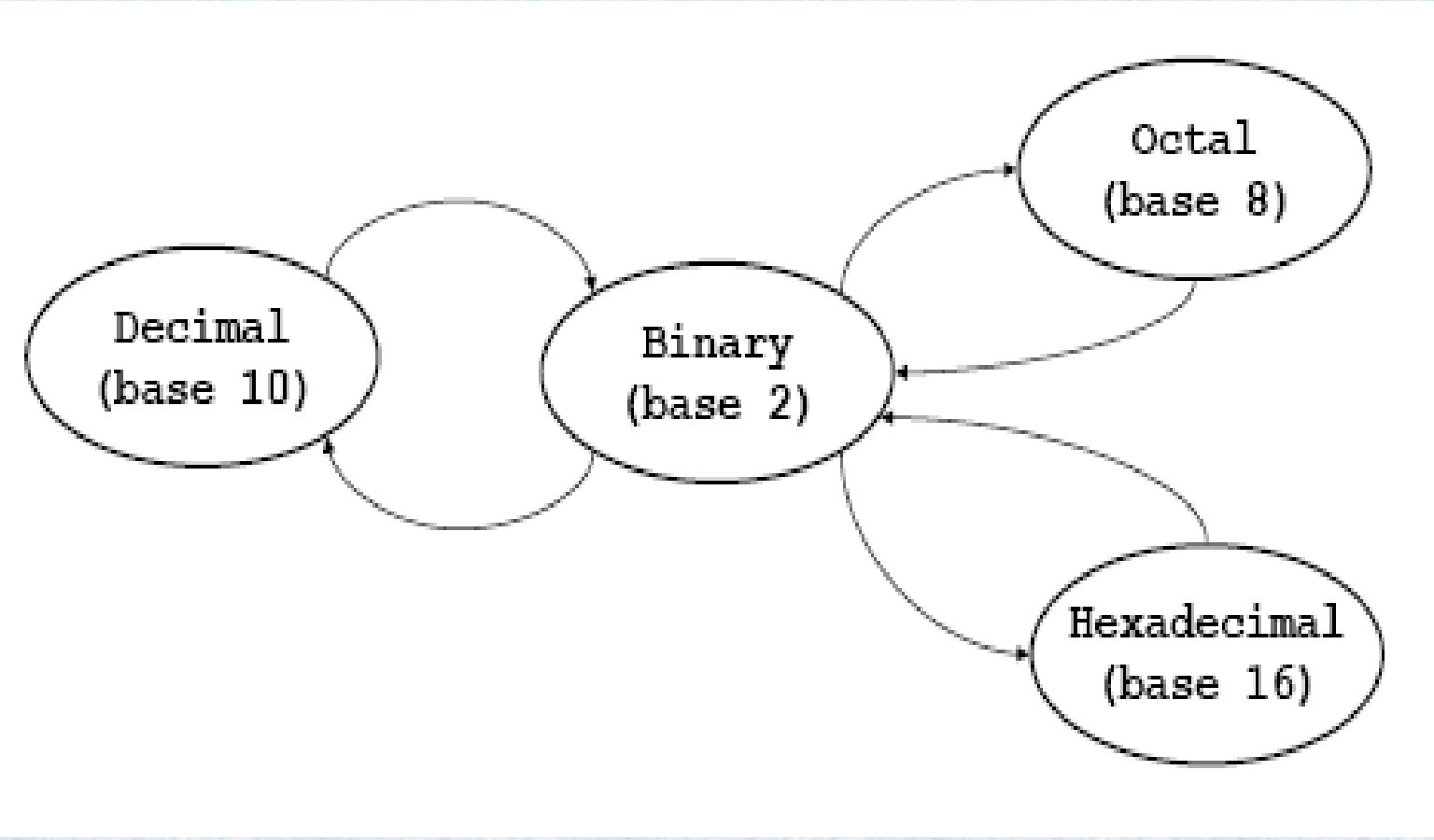


Fig. 10-5 Signals for Evaluating Noise Margin

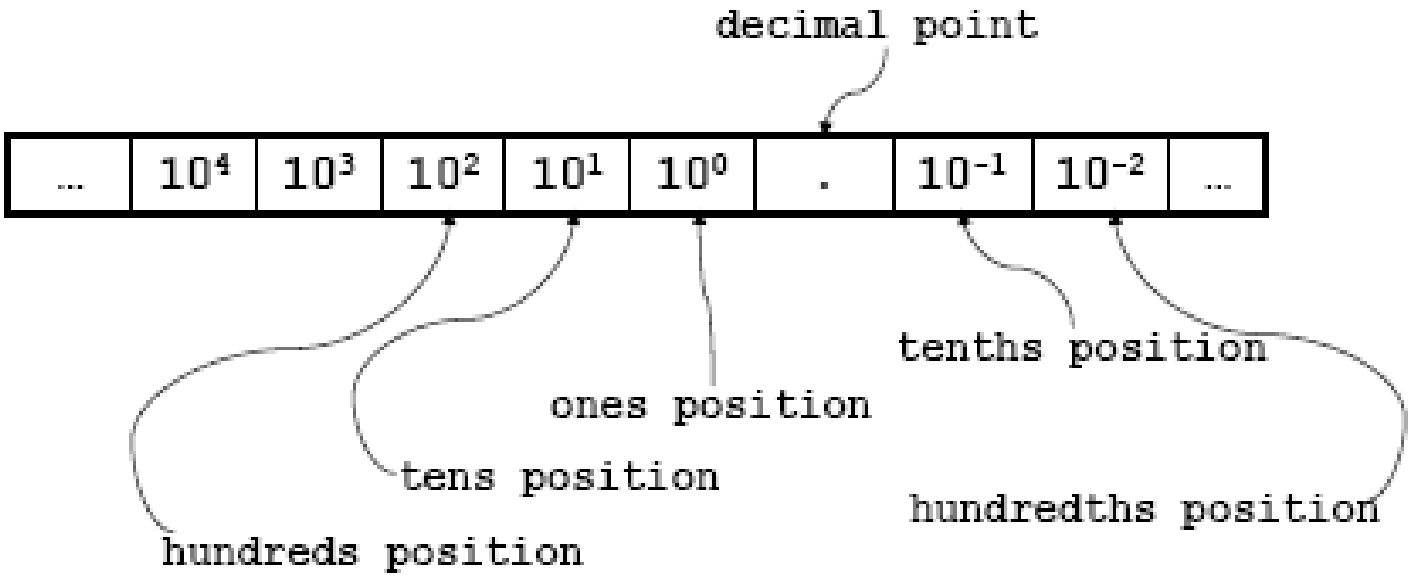
Power Dissipation (P_D)

- **Expressed in Milliwatts**
- **PD= V_{CC} * I_{CC}**
- **I_{CC}(avg)=(I_{CCH}+I_{CCL})/2**

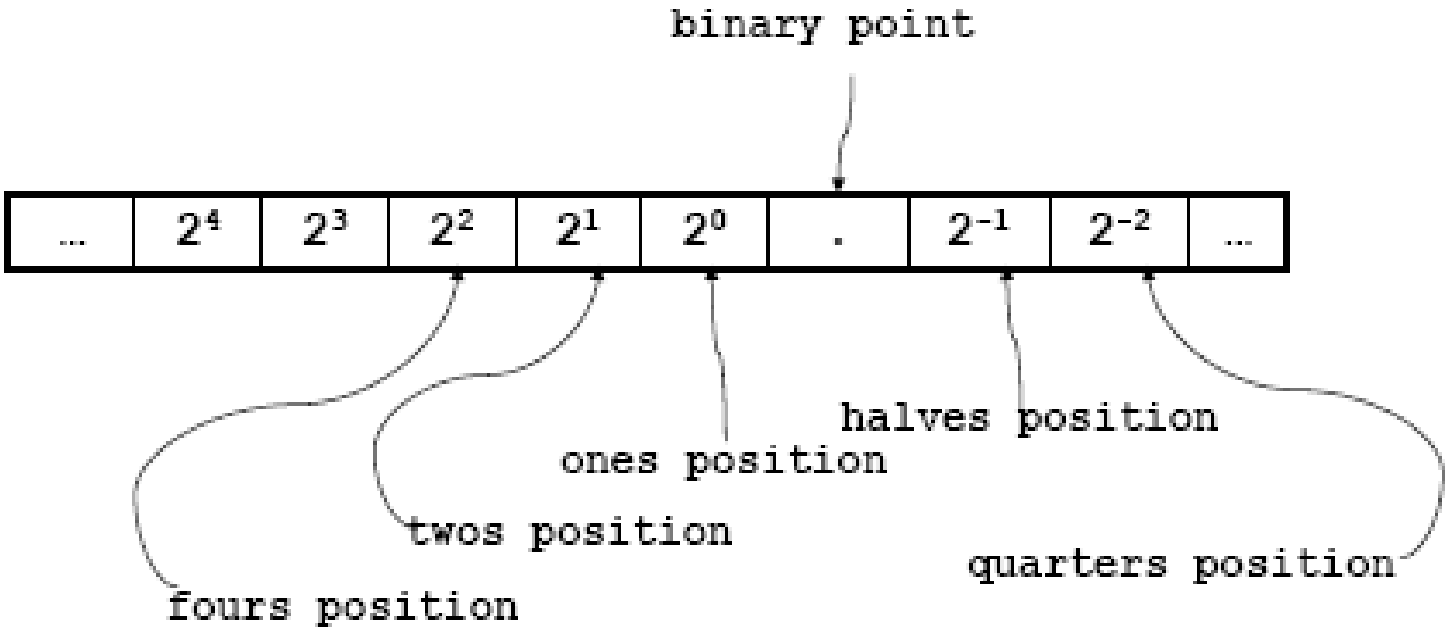
Number Systems and Codes



Decimal Positional System
(Base 10 or radix 10)



Binary Positional System
(Base 2 or radix 2)



Example

Decimal Example

$$346.17_{10} = (3\times10^{2}) + (4\times10^{1}) + (6\times10^{0}) + (1\times10^{-1}) + (7\times10^{-2})$$
$$= 300 + 40 + 6 + 0.1 + 0.07$$

Binary Example

$$1101.01_{2} = 1\times2^{3} + 1\times2^{2} + 0\times2^{1} + 1\times2^{0} + 0\times2^{-1} + 1\times2^{-2}$$
$$= 8 + 4 + 0 + 1 + 0 + .25$$
$$= 13.25_{10}$$

- **Diminished Radix Complement**

  Number N in base r having n digits

  (r-1)'s complement $= (r^n - 1) - N$

- **Radix Complement**

  Number N in base r having n digits

  (r)'s complement $= (r^n - N)$

Ex: base 2: 2's complement, 1's complement

base10: 10's compliment, 9's compliment

Some possible binary codes are illustrated below.

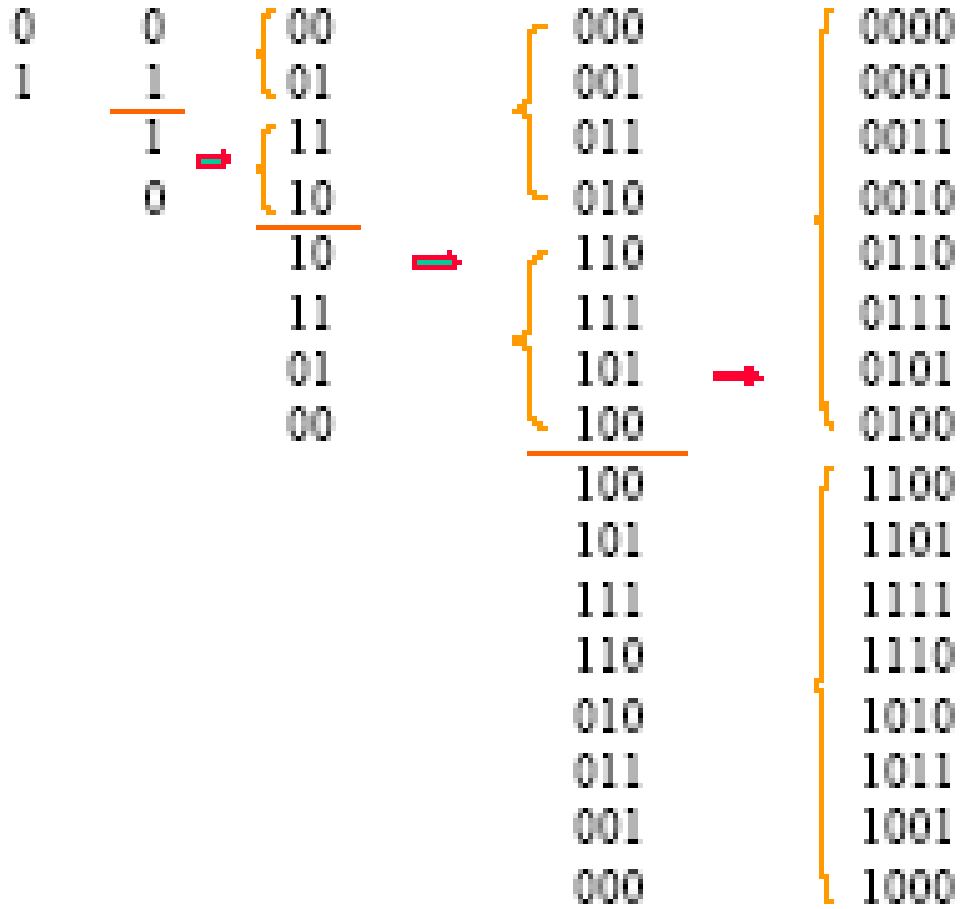
Decimal digits	BCD (8421)	Excess-3	84-2-1	2421	Bi-quinary (5043210)
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

The Reflected Code

The advantage of the reflected code over pure binary numbers is that a number in the reflected code changes by only one bit as it proceeds from one number to the next. The reflected code is also known as the *Gray* code.

Generation of Gray Codes:

an n-bits code is generated by reflecting the (n-1)-bit code.



Reflected Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	2
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Four-bit reflected code

- Boolean expressions must be evaluated with the following order of operator precedence
  - parentheses
  - NOT
  - AND
  - OR

Example:

$$F = (A(\overline{C + \overline{B}D}) + \overline{B\overline{C}})\overline{E}$$

$$F = \underbrace{\left(\underbrace{A \underbrace{\left(\overline{C + \underbrace{\overline{B}D}} \right)}} + \overline{\underbrace{B\overline{C}}} \right)} \overline{E}$$

$X + 0 = X$	$X \cdot 1 = X$	Identity
$X + 1 = 1$	$X \cdot 0 = 0$	
$X + X = X$	$X \cdot X = X$	Idempotent Law
$X + X' = 1$	$X \cdot X' = 0$	Complement
$(X')' = X$		Involution Law
$X + Y = Y + X$	$XY = YX$	Commutativity
$X + (Y + Z) = (X + Y) + Z$	$X(YZ) = (XY)Z$	Associativity
$X(Y + Z) = XY + XZ$	$X + YZ = (X + Y)(X + Z)$	Distributivity
$X + XY = X$	$X(X + Y) = X$	Absorption Law
$X + X'Y = X + Y$	$X(X' + Y) = XY$	Simplification
$(X + Y)' = X'Y'$	$(XY)' = X' + Y'$	DeMorgan's Law
$XY + X'Z + YZ = XY + X'Z$	$(X + Y)(X' + Z)(Y + Z) = (X + Y)(X' + Z)$	Consensus Theorem

- Example: Simplify the following expression

$$F = BC + B\overline{C} + BA$$

- Simplification

$$F = B(C + \overline{C}) + BA$$

$$F = B \cdot 1 + BA$$

$$F = B(1 + A)$$

$$F = B$$

- Example: Simplify the following expression

$$F = A + \overline{A}B + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}E$$

- Example: Simplify the following expression

$$F = A + \bar{A}B + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}E$$

- Simplification

$$F = A + \bar{A}(B + \bar{B}C + \bar{B}\bar{C}D + \bar{B}\bar{C}\bar{D}E)$$

$$F = A + B + \bar{B}C + \bar{B}\bar{C}D + \bar{B}\bar{C}\bar{D}E$$

$$F = A + B + \bar{B}(C + \bar{C}D + \bar{C}\bar{D}E)$$

$$F = A + B + C + \bar{C}D + \bar{C}\bar{D}E$$

$$F = A + B + C + \bar{C}(D + \bar{D}E)$$

$$F = A + B + C + D + \bar{D}E$$

$$F = A + B + C + D + E$$

Complementing a truth table

- The complement of a function should output 0 when the original function outputs 1, and vice versa.
- In a truth table, we can just exchange 0 and 1 in the output column.
  - On the left is a truth table for f(x,y,z) = (x + y')z + x'.
  - On the right is the table for the complement of f, denoted f'(x,y,z).

x	y	z	f(x,y,z)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

x	y	z	f'(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Complementing an expression

- To complement an expression, you can use DeMorgan's Laws to keep "pushing" the NOT operator inwards, all the way to the literals.

$$f(x,y,z) = (x + y')z + x'$$

$$\begin{aligned} f'(x,y,z) &= ((x + y')z + x')' && \text{[complementing both sides]} \\ &= ((x + y')z)' \bullet (x')' && \text{[because } (x + y)' = x'y' \text{]} \\ &= ((x + y')' + z') \bullet x && \text{[} (xy)' = x' + y' \text{, and } (x')' = x \text{]} \\ &= (x'y + z') \bullet x && \text{[} (x + y)' = x'y' \text{ again]} \end{aligned}$$

- Another clever method of complementing an expression is to take the dual of the expression, and then complement each literal.
  - The dual of $(x + y')z + x'$ is $(xy' + z) \bullet x'$.
  - Complementing each literal yields $(x'y + z') \bullet x$.
  - So $f'(x,y,z) = (x'y + z') \bullet x$.

- **Complement of a Function.**

  F = x’yz’ + x’y’z

- **Complement of a Function.**

  F = x'yz' + x'y'z

  dual of F is (x'+y+z')(x'+y'+z)

  complement each literal

  - (x+y'+z)(x+y+z') = F'

Sum of products expressions

- There are many equivalent ways to write a function, but some forms turn out to be more useful than others.
- A sum of products or SOP expression consists of:
  - One or more terms *summed* (OR'ed) together.
  - Each of those terms is a *product of literals*.

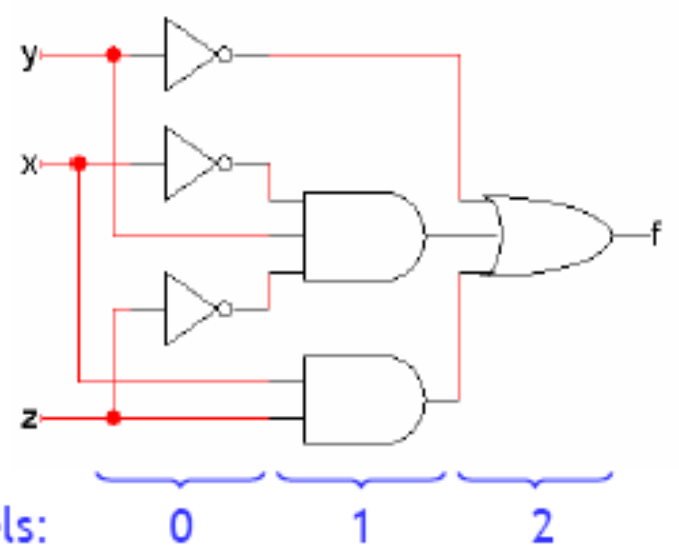
$$f(x, y, z) = y' + x'yz' + xz$$

Sum of products expressions

- There are many equivalent ways to write a function, but some forms turn out to be more useful than others.
- A sum of products or SOP expression consists of:
  - One or more terms *summed* (OR'ed) together.
  - Each of those terms is a *product of literals*.

$$f(x, y, z) = y' + x'yz' + xz$$

- Sum of products expressions can be implemented with two-level circuits.



Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with n input variables has 2^n possible minterms.

Minterms

- A minterm is a special product of literals, in which each input variable appears exactly once.
- A function with *n* input variables has 2^n possible minterms.
- For instance, a three-variable function f(x,y,z) has 8 possible minterms:

x'y'z'	x'y'z	x'y z'	x'y z
x y'z'	x y'z	x y z'	x y z

- Each minterm is true for exactly one combination of inputs.

Minterm	True when	Shorthand
x'y'z'	xyz = 000	m_0
x'y'z	xyz = 001	m_1
x'y z'	xyz = 010	m_2
x'y z	xyz = 011	m_3
x y'z'	xyz = 100	m_4
x y'z	xyz = 101	m_5
x y z'	xyz = 110	m_6
x y z	xyz = 111	m_7

Sum of minterms expressions

- A sum of minterms is a special kind of sum of products.
- Every function can be written as a *unique* sum of minterms expression.
- A truth table for a function can be rewritten as a sum of minterms just by finding the table rows where the function output is 1.

Sum of minterms expressions

- A sum of minterms is a special kind of sum of products.
- Every function can be written as a *unique* sum of minterms expression.
- A truth table for a function can be rewritten as a sum of minterms just by finding the table rows where the function output is 1.

x	y	z	C(x,y,z)	C'(x,y,z)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$C = x'yz + xy'z + xyz' + xyz$
$= m_3 + m_5 + m_6 + m_7$
$= \Sigma m(3,5,6,7)$

$C' = x'y'z' + x'y'z + x'yz' + xy'z'$
$= m_0 + m_1 + m_2 + m_4$
$= \Sigma m(0,1,2,4)$

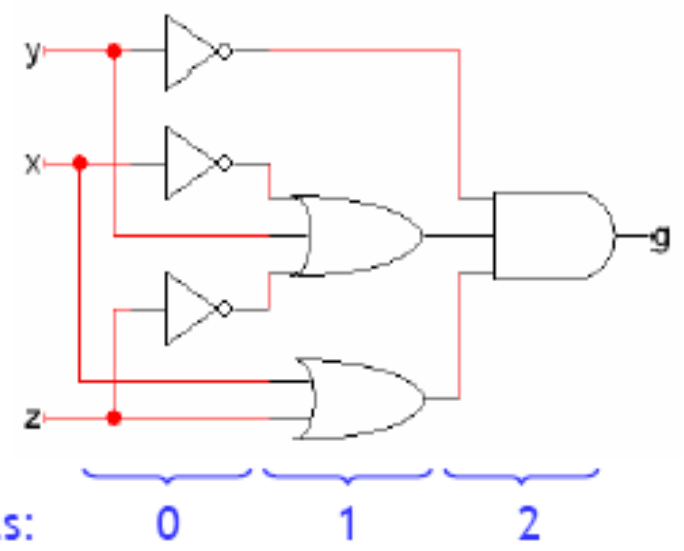
C' contains all the minterms *not* in C, and vice versa.

Product of sums expressions

- As you might expect, we can work with the duals of these ideas too.
- A product of sums or POS consists of:
  - One or more terms *multiplied* (AND'ed) together.
  - Each of those terms is a *sum of literals*.

$$g(x, y, z) = y'(x' + y + z')(x + z)$$

- Products of sums can also be implemented with two-level circuits.



Maxterms

- A maxterm is a *sum* of literals where each input variable appears once.
- A function with *n* input variables has 2^n possible maxterms.
- For instance, a function with three variables x, y and z has 8 possible maxterms:

x + y + z x + y + z' x + y'+ z x + y'+ z'
x'+ y + z x'+ y + z' x'+ y'+ z x'+ y'+ z'

- Each maxterm is *false* for exactly one combination of inputs.

Maxterm	False when	Shorthand
x + y + z	xyz = 000	M_0
x + y + z'	xyz = 001	M_1
x + y'+ z	xyz = 010	M_2
x + y'+ z'	xyz = 011	M_3
x'+ y + z	xyz = 100	M_4
x'+ y + z'	xyz = 101	M_5
x'+ y'+ z	xyz = 110	M_6
x'+ y'+ z'	xyz = 111	M_7

Product of maxterms expressions

- Every function can also be written as a unique product of maxterms.
- A truth table for a function can be rewritten as a product of maxterms just by finding the table rows where the function output is 0.

x	y	z	C(x,y,z)	C'(x,y,z)
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$C = (x + y + z)(x + y + z')$
$(x + y' + z)(x' + y + z)$
$= M_0 M_1 M_2 M_4$
$= \prod M(0,1,2,4)$

$C' = (x + y' + z')(x' + y + z')$
$(x' + y' + z)(x' + y' + z')$
$= M_3 M_5 M_6 M_7$
$= \prod M(3,5,6,7)$

C' contains all the maxterms *not* in C, and vice versa.

- Now we've seen two different ways to write the function C, as a sum of minterms $\Sigma m(3,5,6,7)$ and as a product of maxterms $\Pi M(0,1,2,4)$.
- Notice the product term includes maxterm numbers whose corresponding minterms do not appear in the sum expression.

x	y	z	C(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$C = x'yz + xy'z + xyz' + xyz$
$= m_3 + m_5 + m_6 + m_7$
$= \Sigma m(3,5,6,7)$

$C = (x + y + z)(x + y + z')$
$(x + y' + z)(x' + y + z)$
$= M_0 M_1 M_2 M_4$
$= \Pi M(0,1,2,4)$

The relationship

- Every minterm m_i is the *complement* of its corresponding maxterm M_i.

Minterm		True when
(m_0)	x'y'z'	xyz = 000
(m_1)	x'y'z	xyz = 001
(m_2)	x'y z'	xyz = 010
(m_3)	x'y z	xyz = 011
(m_4)	x y'z'	xyz = 100
(m_5)	x y'z	xyz = 101
(m_6)	x y z'	xyz = 110
(m_7)	x y z	xyz = 111

Maxterm		False when
(M_0)	x + y + z	xyz = 000
(M_1)	x + y + z'	xyz = 001
(M_2)	x + y'+ z	xyz = 010
(M_3)	x + y'+ z'	xyz = 011
(M_4)	x'+ y + z	xyz = 100
(M_5)	x'+ y + z'	xyz = 101
(M_6)	x'+ y'+ z	xyz = 110
(M_7)	x'+ y'+ z'	xyz = 111

- For example, $m_4' = M_4$ because $(xy'z')' = x' + y + z$.

Converting between standard forms

- We can convert sums of minterms to products of maxterms algebraically.

$C = \Sigma m(3,5,6,7)$

$C' = \Sigma m(0,1,2,4)$ [C' contains the minterms not in C]
$= m_0 + m_1 + m_2 + m_4$

$(C')' = (m_0 + m_1 + m_2 + m_4)'$ [complementing both sides]
$C = m_0' \, m_1' \, m_2' \, m_4'$ [DeMorgan's law]
$= M_0 \, M_1 \, M_2 \, M_4$ [from the previous page]
$= \Pi M(0,1,2,4)$

- The easy way is to replace minterms with maxterms, using the maxterm numbers that do not appear in the sum of minterms.

$C = \Sigma m(3,5,6,7)$
$= \Pi M(0,1,2,4)$

- Example

$$F(A, B, C) = AB + \overline{B}(\overline{A} + \overline{C}) = AB + \overline{A}\,\overline{B} + \overline{B}\,\overline{C}$$
$$= AB(C + \overline{C}) + \overline{A}\,\overline{B}(C + \overline{C}) + (A + \overline{A})\overline{B}\,\overline{C}$$
$$= \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$$
$$= \sum m(0, 1, 4, 6, 7)$$

A	B	C	F	
0	0	0	1	← 0
0	0	1	1	← 1
0	1	0	0	
0	1	1	0	
1	0	0	1	← 4
1	0	1	0	
1	1	0	1	← 6
1	1	1	1	← 7

Minterms listed as 1s in Truth Table

- **Canonical Form**

  - Boolean Expression expressed in sum of minterms or product of maxterms

- **Standard form**

  F1 = y’ +xy + x’yz’

- **F = (AB + CD)(A’B’+C’D’)**