Unit 7  Multi-Level Gate Circuits/
NAND and NOR Gates
7.1 Multi-Level Gate Circuits
The maximum number of gates cascaded in series between a circuit input and the output is referred to as the number of levels of gates.

- ANR-OR circuit → A level of AND gates followed by a OR at the output
- OR-AND circuit → A level of OR gates followed by a AND at the output
- OR-AND-OR circuit → A level of OR gates followed by a level of AND gates followed by OR gate at the output

A function written in SOP form or in POS form corresponds to a two-level gate circuit.

Inverters which are connected directly to input variables will not be counted when determining the # of levels.
Example: A four-level realization with 6 gates and 13 gate inputs
Another realization of 3 levels of gates. There are six gates and 19 gate inputs in total.

\[ Z = AB(D + E) + C(D + E) + ABFG + CFG + H \]

* The same gate can be used for both appearances of \((D + E)\).

(a) Level 1

(b) Level 2

Level 3
Example: Find a circuit of AND and OR gates to realize

\[ f(a, b, c, d) = \sum m(1, 5, 6, 10, 13, 14) \]

By Karnaugh map, \( f = a'c'd + bc'd + bcd' + acd' \) (7-1)
Factoring (7-1) yields $f = c'd(a' + b) + cd'(a + b)$  (7-2)

Both realizations use 5 gates, but the later one has fewer inputs with one more level of gate delays.
An alternative realization in POS form: obtained from the 0’s on the Karnaugh map

\[ f' = c'd' + ab'c' + cd + a'b'c \]  \hspace{1cm} (7-3)

\[ f = (c+d)(a'+b+c)(c'+d')(a+b+c') \]  \hspace{1cm} (7-4)
- Partially multiplying out (7-4) using \((X+Y)(X+Z) = X+YZ\):
  \[ f = [c+d(a'+b)][c'+d'(a+b)] \quad (7-5) \]
  \[ = (c+a'd+bd)(c'+ad'+bd') \quad (7-6) \]
- Eq. (7-6) leads to a 3-level AND-OR-AND circuit
Summaries:

- If an expression for $f'$ has $n$-levels, the complement of that expression is an $n$-level expression of $f$.
- To realize $f$ as an $n$-level circuit with an AND-gate output, one procedure is to find an $n$-level expression for $f'$ with an OR operation at the output and then complement the expression for $f'$. 
7.2 NAND and NOR Gates
- NAND and NOR gates are frequently used because they are generally faster and use fewer components than AND or OR gates.
- Any logic function can be implemented using only NAND or only NOR gates.
- An n-input NAND gate is

\[ F = (X_1X_2\cdots X_n)' = X_1' + X_2' + \cdots + X_n' \]
Similarly, an n-input NOR gate is

\[ F = (X_1 + X_2 + \cdots + X_n)' = X_1'X_2'\cdots X_n' \]

A set of function is said to be functionally complete if any Boolean function can be expressed in terms of this set of operations, \textit{e.g.} AND, OR and NOT.

Any set of logic gates which can realize AND, OR, and NOT is also functionally complete.
E.g. AND and NOT form a functionally complete set of gates, since

\[(X'Y')' = X + Y\]

NAND is also functionally complete

\[(A'B')' = A + B\]
7.3 Design of Two-Level Circuits Using NAND and NOR Gates
A two-level circuit composed of AND and OR gates is easily converted to a circuit composed of NAND gates or NOR gates. E.g. converting from a minimum SOP

\[ F = A + BC' + B'CD = [(A + BC' + B'CD)'][ \]

(by 7-11) (7-14)

\[ = [A' \cdot (BC')' \cdot (B'CD)'][ \]

(by 7-12) (7-15)

\[ = [A' \cdot (B' + C) \cdot (B + C' + D')'][ \]

(by 7-12) (7-16)

\[ = A + (B' + C)' + (B + C' + D')' \]

(7-13) : AND-OR

(7-14) : NAND-NAND

(7-15) : OR-NAND

(7-16) : NOR-OR
Logic Design  Unit 7 Multi-Level Gate Circuits  Sau-Hsuan Wu

\[ F = A + B' + B'CD \]  (7-13)

\[ F = A + (B' + C') + (B + C' + D') \]  (7-16)

\[ F = (A' \cdot (B' + C') \cdot (B + C' + D'))' \]  (7-15)

\[ F = [A' \cdot (B'C') \cdot (B'CD)]' \]  (7-14)
Obtaining a two-level circuit containing only NOR gates should start with the minimum POS for F, instead of SOP.

**E.g.**

\[
F = (A + B + C)(A + B' + C')(A + C' + D)
\]

\[
= \{ [(A + B + C)(A + B' + C')(A + C' + D)]' \}'
\]

\[
= (A + B + C)' + (A + B' + C')' + (A + C' + D)'
\]

(by 7-12) \hspace{1cm} (7-19)

\[
= (A'B'C' + A'BC + A'CD')'
\]

(by 7-11) \hspace{1cm} (7-20)

\[
= (A'B'C')' \cdot (A'BC)' \cdot (A'CD')'
\]

(by 7-11) \hspace{1cm} (7-21)

- (7-18) : OR-AND
- (7-19) : NOR-NOR
- (7-20) : AND-NOR
- (7-21) : NAND-AND
Logic Design

Unit 7 Multi-Level Gate Circuits

Sau-Hsuan Wu

\[ F = (A + B + C)' \cdot (A' + B' + C') \cdot (A + C' + D) \] (7-18)
Two of the most commonly used circuits are the NAND-NAND and the NOR-NOR.

Procedure for designing a min 2-level NAND-NAND circuit:

- Find a minimum SOP for $F$.
- Draw the corresponding two-level AND-OR circuit.
- Replacing all gates with NAND gates.

(a) Before transformation

(b) After transformation
Procedure for designing a min 2-level NOR-NOR circuit

- Find a minimum POS for F
- Draw the corresponding two-level OR-AND circuit
- Replace all gates with NOR gates
7.4 Design of Multi-Level NAND- and NOR-Gate Circuits
The following procedure may be used to design multi-level NAND-gate circuits:

- Simplify the switching function to be realized.
- Design a multi-level circuit of AND and OR gates.
- The output gate must be a OR gate.
- AND-gate outputs cannot be used as AND-gate inputs; OR-gate outputs cannot be used as OR-gates inputs.
- Replace all gates with NAND gates.

The procedure for the design of multi-level NOR-gate circuits is exactly the same as for NAND-gate circuits except that the output gate of the circuit must be an AND gate, and all gates are replaced with NOR gates.
Logic Design

Unit 7 Multi-Level Gate Circuits

Sau-Hsuan Wu

(a) AND-OR network

(b) NAND network

Department of Communication Engineering, NCTU
7.5 Circuit Conversion Using Alternative Gate Symbols
- Alternative representations for an inverter

- Alternative representations for AND, OR, NAND and NOR gates

\[
egin{align*}
AB &= (A' + B')' \\
A + B &= (A'B')' \\
(AB)' &= A' + B' \\
(A + B)' &= A'B'
\end{align*}
\]
(a) NAND gate network

\[ Z = (A' + B) C + F' + DE \]

(b) Alternate form for NAND gate network

(c) Equivalent AND-OR network
The procedure for converting AND-OR circuit to a NAND or NOR circuit

- Convert all AND gates to NAND gates by adding an inversion bubble at the output
- Convert all OR gates to NAND gates by adding inversion bubbles at the inputs
- Whenever an inverted output drives an inverted input, these two inversions cancel
- Whenever a noninverted gate output drives an inverted gate input or vice versa, insert an inverter so that the bubble will cancel
(a) AND-OR network

(b) First step in NAND conversion

(c) Completed conversion
Example

(a) Circuit with OR and AND gates

Double inversion cancels

(b) Equivalent circuit with NOR gates
7.6 Design of Two-Level, Multiple-Output Circuits
Solution of digital design problems often requires the realization of several functions of the same variables. The use of some gates in common between two or more functions sometimes leads to a more economical circuit.

E.g. we have

<table>
<thead>
<tr>
<th>CD</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

\[ F_1 = A \overline{B} + \overline{A}C \overline{D} \]

\[ F_2 = \overline{AB} + \overline{A} \overline{B} \overline{C} \overline{D} \]

\[ F_3 = \overline{A}BC + \overline{A} \overline{B} \overline{C} \overline{D} \]
The individual realizations to them are

\[ F_1 = AB + ACD \]

\[ F_2 = ABC' + CD \]

\[ F_3 = A'CD + AB \]
Observe that the term ACD is necessary for the realization of $F_1$ and that A’CD is necessary for $F_3$. If replacing CD in $F_2$ by A’CD + ACD, the realization of CD is unnecessary.
In realizing multiple-output circuits, the use of a minimum sum of prime implicants for each function does not necessarily lead to a minimum cost solution.

When designing multiple-output circuits, try to minimize the total number of gates required.

E.g.

\[
\begin{align*}
  f_1 &= \sum m(2,3,5,7,8,9,10,11,13,15) \\
  f_2 &= \sum m(2,3,5,6,7,10,11,14,15) \\
  f_3 &= \sum m(6,7,8,9,13,14,15)
\end{align*}
\]
The corresponding Karnaugh maps are

\[ f_1 = bd + b'c + ab' \]
\[ f_2 = c + a'bd \]
\[ f_3 = bc + ab'c' + \begin{cases} 
    abd \\
    or \\
    ac'd 
\end{cases} \]

10 gates, 25 gate inputs
By inspection, we can see that

- $a'bd$ from $f_2$, $abd$ from $f_3$ and $ab'c'$ from $f_3$ can be used in $f_1$. Replacing $bd$ with $a'bd + abd$, the gate needed to realize $bd$ can be eliminated.
- $m_{10}$ and $m_{11}$ in $f_1$ are already covered by $b'c$, and $ab'c'$ from $f_3$ can be used to cover $m_8$ and $m_9$, thus $ab'$ being eliminated.
- The minimum solution is therefore

$$f_1 = a'bd + abd + ab'c' + b'c$$
$$f_2 = c + a'bd$$
$$f_3 = bc + ab'c' + abd$$

eight gates

22 gate inputs
Determination of essential prime implicants for multiple-output realization

- The prime implicants essential to an individual function may not be essential to the multiple-output realization
- $bd$ is an essential prime implicant of $f_1$ but not of all $f$'s
When searching for a prime implicant to an multiple-output realization,

- Check each 1 which do not appear on the other function maps

Example 1

- $c'd$ is essential to $f_1$, $bd'$ is essential to $f_2$
- $abd$ is not essential since it appears on both maps
Example 2

- $a'd'$ and $a'bc'$ are essential to $f_1$
- $bd'$ and $a'b'c$ are essential to $f_2$