7.1 Multi-Level Gate Circuits

Level of a circuit = maximum number of gates in series between input and output

Sum-of-Products: 2 level circuit (AND-OR)
Product-of-Sums: 2 level circuit (OR-AND)

Ex:

\[ (ABCD + EF + G)H + ABCD \]

Why does level matter?
1. Multi-level may have fewer gates
2. Limited number of inputs per gate (i.e., limited fanin).
3. Limited number of loads (gates) connected to the output of a gate (limited fanout).
4. Delay: a change in the input variable value takes time to induce a change in the output value (propagation delay). A circuit can't run faster than its slowest path.

Ex: Suppose each gate has a propagation delay of 100 ps. The slowest delay for the circuit for \( Z \) above takes 400 ps.
7.2 NAND and NOR Gates

A logic gate (or gates) is/are functionally complete if they are able to generate any Boolean function.

Operators for minterm expansions are functionally complete (AND, OR, NOT)
Operators for maxterm expansions are functionally complete (OR, AND, NOT)

AND is not functionally complete \((A \cdot A \neq A')\) \(\{\text{can't generate inverse}\}\)
OR is not functionally complete \((A + A \neq A')\) \(\{\text{can't generate two input operations}\}\)

NAND and NOR are functionally complete.

Difficult to prove directly, but easier to say that we can generate any function by its minterm/maxterm expansion using AND, OR, NOT, and then implement these using NAND/NOR:

**NOT:**

**AND:**

**OR:**

\(\therefore\) NAND/NOR are functionally complete.
Moving Between Different Types of Two Level Circuits

Starting with Sum-of-Products:

A
B
C
D

AND-OR

A'
B'
C'
D'

NOR-OR

A
B
C
D

NAND-NAND

A'
B'
C'
D'

OR-NAND

Starting with Product-of-Sums:

A
B
C
D

OR-AND

A'
B'
C'
D'

NAND-AND

A
B
C
D

NOR-NOR

A'
B'
C'
D'

AND-NOR
Ex: Multi-level NAND

```
A
\|--
    |    \                     OR-AND-OR form
    |  c    \                     
B  |    \                     
   |  c    \                     
   |  d    \                     
   |  z    
   \---

A'  |    \                     convert using
B'  |  c    \                     Demorgan's Law and
   |  d    \                     X'' = X identity
   |    \                     
   |  z    
   \---
```

Ex: Find minimum 3 level NOR circuit that implements

\[ f = a'b' + abd + acd \]

<table>
<thead>
<tr>
<th>cd</th>
<th>db</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>1 0</td>
</tr>
<tr>
<td>01</td>
<td>1 0</td>
</tr>
<tr>
<td>11</td>
<td>1 1</td>
</tr>
<tr>
<td>10</td>
<td>1 0</td>
</tr>
</tbody>
</table>

Want NOR-NOR-NOR circuit,

1) Find AND-OR-AND circuit \( \rightarrow \) find OR-AND circuit then
factor using \((x+y)(x+z) = x+yz\)

To find OR-AND (Product-of-Sums), use K-map for \( f' \):

\[ f' = a'b + ad' + ab'c' \]

\[ f = f'' = (a+b')(a'+d)(a'+b+c) \]

\[ x = a' \quad y = d \quad z = b+c \]

\[ f = (a+b')(a'+d(b+c)) \quad 4 \text{ levels} \]

\[ = (a+b')(a'+db+dc) \quad 3 \text{ levels} \]
Replace AND with NOR equivalents (De Morgan's Laws)