Multilevel Gate Networks

- maximum # of gates cascaded in series between an input and output is the # of levels of gates
  - example: sum of products form is a two level gate network
- we do not count inverter gates
- the number of levels can be increased by factoring the sum of products expression for AND-OR network, or by multiplying out some terms in the product of sum expression for OR-AND network
- # levels of gates is proportional to total propagation delay through logic
- however sometimes by increasing the # of levels one can decrease the # of gates and gate inputs
- one can determine the # of levels, # of gates, and # of gate inputs from inspection.
- Example
  - find AND and OR networks with 2 and 3 levels of gates (that minimizes # of gates, and gate inputs) for \( f(a,b,c,d)=m(1,5,6,10,13,14) \)
  - we must look at both AND-OR networks and OR-AND networks for 2 level gates
  1. simplify \( f \) using karnaugh map
  2. show the 2 level AND-OR network
  3. factor the sum o products expression for \( f \)
  4. show the 3 level OR-AND-OR network from 3.
  5. now using 0's on karnaugh map give the product of sums expression of \( f \).
  6. show the 2 level OR-AND network.
  7. Partially multiply out the expression of \( f \) from 5. to get a 3 level AND-OR-AND network
  8. show the AND-OR-AND network
  9. now answer the question by identifying the best 2 level and 3 level networks.
- If an expression for \( f \) has \( n \) levels the complement \( f \) also has \( n \) levels

Other types of logic gates

- 3 - input NAND gate
  - \( f=(abc)'=a'+b'+c' \)
- 3 - input NOR gate
  - \( f=(a+b+c)'=a'b'c' \)
- any logic function can be implemented using only nand or nor gates.

  - Majority gate
    - \( f=1 \) if majority of inputs (odd #) are 1, \( f_M=bc+ac+ab \)
  - Minority gate
    - \( f=1 \) if minority of inputs (odd #) are 1, \( f_m=(b'+c')(a'+c')(a'+b') \)

Functionally complete set of gates

- a set of logic operations is functionally complete if any boolean function can be expressed in terms of this set of operations

<table>
<thead>
<tr>
<th>Functionally Complete Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND, OR, NOT</td>
</tr>
</tbody>
</table>
AND, NOT

NAND

3 input minority gate

- to show this, write out minimum sum of products expression (make sure it includes complement in one of the expression otherwise it can't be functionally complete)
- then attempt to realize AND or OR.

\[ (x'y')' = x + y \] (AND, NOT functions can realize OR function).

**Design of 2 level NAND, NOR networks**

- can convert AND-OR to network composed of NAND or NOR using \( F = (F')' \) and deMorgan's law \( (x + y)' = x'y', (xy)' = x' + y' \)
- example

<table>
<thead>
<tr>
<th>( a + b'(c + b')d' )</th>
<th>AND-OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ((a + b'(c + b')d')') )</td>
<td></td>
</tr>
<tr>
<td>( (a'(b'c')(b'c'd')') )</td>
<td>NAND-NAND</td>
</tr>
<tr>
<td>( (a'(b' + c)(b + c' + d'))' )</td>
<td>OR-NAND</td>
</tr>
<tr>
<td>( a + (b' + c') + (b + c' + d')' )</td>
<td>NOR-OR</td>
</tr>
</tbody>
</table>

- for NOR-NOR network start with minimum product of sum form

<table>
<thead>
<tr>
<th>( (a + b + c)(a + b' + c')(a + c' + d) )</th>
<th>OR-AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ((a + b + c)' + (a + b' + c')' + (a + c' + d)')' )</td>
<td>NOR-NOR</td>
</tr>
<tr>
<td>( (a'b'c')(a'bc)'(a'cd')' )</td>
<td>NAND-AND</td>
</tr>
</tbody>
</table>

**Design of Multilevel NAND and NOR networks**

1. simplify function to be realized
2. design multilevel AND, OR network where output gate is OR (AND), cannot connect AND -> AND, OR -> OR
3. # the levels where output gate is level 1
4. replace all gates with NAND(NOR). invert literals that are input to levels 1, 3, 5, ...