Logical completeness and 2-level design

So far,

NOT, AND, OR are logically complete: we can implement any Boolean function with them.

Question: are there any other sets of logically complete functions?
**NAND function**

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(A B)'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Symbol:

\[ A \quad (A B)' \quad B \]

To show NAND function is logically complete, we will show it can implement NOT, AND, OR.
* NOT function:

* AND function:
* OR function:
**NANO implementation of SOP form**

Example:

\[
\begin{array}{ccc}
A & \rightarrow & F = ? \\
B & \rightarrow & \\
C & \rightarrow & \\
\end{array}
\]

Using **NANO implementation** for **AND, OR**:

Using **involution property** we can simplify:
**NOR function**

Truth table:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>(A+B)'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

* NOT function: \( x' = (x+x)' \) ( ? )

Symbol:

\[
\begin{array}{c}
A \\
\downarrow \\
\uparrow \\
B \\
\end{array} \quad (A+B)'
\]
* AND function: \[xy = \left[(xy)']'\right] (?)
= \left[x'y'\right] (?)

* OR function: \[x+y = \left[(x+y)']'\right] (?)
* NOR implementation of POS form

Example:

```
x
\downarrow
\downarrow
y
z
```

\[ G = ? \]
2-level networks

A circuit is 2-level if there are at most 2 gates between any input and the output.

Every function can be represented in SOP form, thus it can be implemented with a 2-level (AND-OR) network or 2-level NAND network.

Similarly, every function can be represented in POS form, so it can be implemented with a 2-level (OR-AND) network or a 2-level NOR network.
Exclusive-OR (XOR)

For a 2-input XOR, output is 1 if inputs are different.

For an n-input XOR, output is 1 if inputs have odd number of 1’s.

XOR can be used as controlled inverter: