Technology Mapping

To reduce implementation cost and turnaround time, designers use gate-arrays. These gate-arrays contain only \( m \)-input NAND and NOR gates where \( m \) is usually 3. **Technology mapping** is the process where we convert a schematic (expression) with AND, OR, and NOT gates to NAND and NOR gates.

The conversion is based on the following rules:

1. \( xy = ((xy)' )' \)
2. \( x + y = ((x + y)' )' = (x' y' )' \)
3. \( xy = ((xy)' )' = (x' + y')' \)
4. \( x + y = ((x + y)' )' \)
5. \( x' = x \)

Replace AND and OR gates with NAND gates by using Rules 1 and 2.
Replace AND and OR gates with NOR gates by using Rules 3 and 4.
Eliminate double inverters whenever possible by using Rule 5.

Example: Derive the NAND and NOR implementations of the carry function \( c_{i+1} = x_i y_i + x_i c_i + y_i c_i \)

A better NOR implementation is to start with the product-of-sums expression \( c_{i+1} = (x_i + y_i) (x_i + c_i) (y_i + c_i) \)
Translation of Sum of Products and Product of Sums to NAND and NOR schematics:

<table>
<thead>
<tr>
<th>Form type</th>
<th>Standard form Implementation</th>
<th>NAND Implementation</th>
<th>NOR Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum of products</td>
<td><img src="image" alt="NAND Circuit" /></td>
<td><img src="image" alt="NAND Circuit" /></td>
<td><img src="image" alt="NOR Circuit" /></td>
</tr>
<tr>
<td>Product of sums</td>
<td><img src="image" alt="NAND Circuit" /></td>
<td><img src="image" alt="NAND Circuit" /></td>
<td><img src="image" alt="NOR Circuit" /></td>
</tr>
</tbody>
</table>

Four more identities:

\[
\begin{align*}
\text{\_\_\_} & = \quad \text{\_\_\_} & = \quad \text{\_\_\_]}
\end{align*}
\[
\begin{align*}
\text{\_\_\_} & = \quad \text{\_\_\_} & = \quad \text{\_\_\_]}
\end{align*}
\[
\begin{align*}
\text{\_\_\_} & = \quad \text{\_\_\_} \\
\text{\_\_\_} & = \quad \text{\_\_\_}
\end{align*}
\[
\begin{align*}
\text{\_\_\_} & = \quad \text{\_\_\_}
\end{align*}
\]
Example: Implement the following schematic using 2 and 3-input NAND gates:

**Rule 1:**

```
\[ \text{AND gate} = \text{NAND gate} \]
```

**Rule 2:**

```
\[ \text{OR gate} = \text{NAND gate} \]
```

**AND-OR implementation using only 2 & 3-input gates**

**Convert to NAND implementation**

**Optimized decomposition**

**NAND implementation**

**Optimized NAND implementation**

Max delay = 9.2 ns

Max delay = 6.4 ns
Example: Convert the expression $w'z' + z(w + y)$ into a logic schematic using any of the gates from the library defined in Tables 3.14, 3.15 and 3.16.

AND-OR implementation
delay=7.2ns, cost=28

NAND implementation
delay=5.2ns, cost=22

two possible conversions

Rule 1:

Rule 2:

Alternative A
delay=5.4ns, cost=20

Alternative B
delay=3.8ns, cost=20

Cost-optimized alternative B
delay=3.8ns, cost=18