NAND & NOR Implementation

By : Ali Mustafa
NAND and NOR Implementations

• NAND & NOR gates are universal gates.
• Digital circuit are frequently constructed with NAND or NOR gates rather than AND and OR gates.
• NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families. (covering AND, OR, NOT)
DeMorgan’s Law

Recall the DeMorgan’s Law:

\[(a + b)' = a' \cdot b'\] \hspace{2cm} \[(a \cdot b)' = a' + b'\]

\[a + b = (a' \cdot b')'\] \hspace{2cm} \[(a \cdot b) = (a' + b')'\]
Universal Gate – NAND

I will demonstrate

• The basic function of the NAND gate.

• How a NAND gate can be used to replace an AND gate, an OR gate, or an INVERTER gate.

• How a logic circuit implemented with AOI logic gates can be re-implemented using only NAND gates.

• That using a single gate type, in this case NAND, will reduce the number of integrated circuits (IC) required to implement a logic circuit.

AOI Logic

More ICs = More $$

NAND Logic

Less ICs = Less $$
NAND Gate

\[ Z = \overline{X Y} = \overline{X} + \overline{Y} \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>
NAND Gate as an Inverter Gate

\[ X \cdot X = X \] (Before Bubble)

\[ Z = \overline{X} \]

<table>
<thead>
<tr>
<th>X</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Equivalent to Inverter
NAND Gate as an AND Gate

\[
X \quad Y \quad \overline{X \cdot Y} \quad Z = \overline{X \cdot Y} = \overline{X} \cdot \overline{Y}
\]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Equivalent to AND Gate
NAND Gate as an OR Gate

\[ Z = \overline{X \overline{Y}} = \overline{X} + \overline{Y} = X + Y \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</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>1</td>
</tr>
</tbody>
</table>

Equivalent to OR Gate
NAND Gate Equivalent to AOI Gates

AND

OR

INVERTER
Process for NAND Implementation

1. If starting from a logic expression, implement the design with AOI logic.

2. In the AOI implementation, identify and replace every AND, OR, and INVERTER gate with its NAND equivalent.

3. Redraw the circuit.

4. Identify and eliminate any double inversions (i.e., back-to-back inverters).

5. Redraw the final circuit.
NAND Implementation

Example:

Design a NAND Logic Circuit that is equivalent to the AOI circuit shown below.

\[ z = B \overline{C} + A \overline{C} \]
NAND Implementation

Solution – Step 2

Identify and replace every AND, OR, and INVERTER gate with its NAND equivalent.
Redraw the circuit.
NAND Implementation
Solution – Step 4

Identify and eliminate any double inversions.
NAND Implementation
Solution – Step 5

Redraw the circuit.
Proof of Equivalence

\[ Z = B \overline{C} + \overline{A} \overline{C} \]
## AOI vs. NAND

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Gates</th>
<th>Gate / IC</th>
<th># ICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS04</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>74LS08</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>74LS32</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total Number of ICs →** 3

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Gates</th>
<th>Gate / IC</th>
<th># ICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS00</td>
<td>4</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**Total Number of ICs →** 1
Summary - NAND Universality

- Inverter
  - $A \rightarrow \overline{A}$

- AND gate
  - $A \& B \rightarrow AB$

- OR gate
  - $A \oplus B \rightarrow A + B$

- NOR gate
  - $A \oplus B \rightarrow \overline{A + B}$
Self Task

• Implement the following Boolean express using only NAND gates

\[ Y = ABC' + ABC + A'BC \]
\[ Y = AB + CDE + F \]
\[ Y = \sum (0,1,5) \]
Universal Gate – NOR

Now I will demonstrate...

• The basic function of the **NOR** gate.

• How an **NOR** gate can be using to replace an **AND** gate, an **OR** gate or an **INVERTER** gate.

• How a logic circuit implemented with **AOI** logic gates could be re-implemented using only **NOR** gates.

• That using a single gate type, in this case **NOR**, will reduce the number of integrated circuits (IC) required to implement a logic circuit.

 AOI Logic

More ICs = More $$

NOR Logic

Less ICs = Less $$
NOR Gate

\[ Z = \overline{X + Y} = \overline{X} \cdot \overline{Y} \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
NOR Gate as an Inverter Gate

\[ X + X = X \quad \text{(Before Bubble)} \]

\[ Z = \overline{X} \]

\[ \begin{array}{c|c}
X & Z \\
\hline
0 & 1 \\
1 & 0 \\
\end{array} \]

Equivalent to Inverter
NOR Gate as an OR Gate

\[
X \lor Y = \overline{X + Y} = X + Y
\]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</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>1</td>
</tr>
</tbody>
</table>

Equivalent to OR Gate
NOR Gate as an AND Gate

Z = \overline{X} \overline{Y} = \overline{X Y} = X \overline{Y}

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Equivalent to AND Gate
NOR Gate Equivalent of AOI Gates

AND

OR

INVERTER
Process for NOR Implementation

1. If starting from a logic expression, implement the design with AOI logic.

2. In the AOI implementation, identify and replace every AND, OR, and INVERTER gate with its NOR equivalent.

3. Redraw the circuit.

4. Identify and eliminate any double inversions. (i.e. back-to-back inverters)

5. Redraw the final circuit.
**NOR Implementation**

*Example:*

Design a NOR Logic Circuit that is equivalent to the AOI circuit shown below.

$$z = B \overline{C} + A \overline{C}$$
NOR Implementation
Solution – Step 2

Identify and replace every AND, OR, and INVERTER gate with its NAND equivalent.
NOR Implementation

Solution – Step 3

Redraw Circuit.
NOR Implementation

Solution – Step 4

Identify and eliminate any double inversions.
NOR Implementation

Solution – Step 5

Redraw Circuit.
# AOI vs NOR

## Circuit Diagrams

### AOI Circuit

- Inputs: A, B, C
- Output: Z

### NOR Circuit

- Inputs: A, B, C
- Output: Z

## IC Summary

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Gates</th>
<th>Gate / IC</th>
<th># ICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS04</td>
<td>1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>74LS08</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>74LS32</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>Number of ICs</strong></td>
<td><strong>3</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IC Type</th>
<th>Gates</th>
<th>Gate / IC</th>
<th># ICs</th>
</tr>
</thead>
<tbody>
<tr>
<td>74LS02</td>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total Number of ICs</strong></td>
<td><strong>2</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

33
Summary NOR - Universality

- **Inverter**
  - $A \rightarrow \overline{A}$

- **OR gate**
  - $A + B$

- **AND gate**
  - $A \land B$

- **NAND gate**
  - $\overline{A \land B}$
Self Task

• Implement the following Boolean expression using only NOR gates

\[ Y = (A' + B + C)(A + B)D \]
\[ Y = \prod (0,1,2,4,6) \]
Conversion to NAND Implementation

• Minimized expressions are AND-OR combinations
  –Two illustrations for NAND gates
    • AND-invert
    • Invert-OR

• Key observation: two “bubbles” eliminate each other
  • Two bubbles equal straight wire
• How to generate a sum of minterms using NAND?
  –Use AND-invert for minterms
  –Use invert-OR for sum
Conversion to NAND Implementation

- **Sum of minterms**

- **Replace AND with AND-invert and OR with invert-OR**
  - Still same circuit!

- **Replace AND-invert and invert-OR with NAND**
  - $F = \left( (AB)'(CD)' \right)'$
  - $= AB + CD$
NAND Example

- Function $F = \Sigma(1,2,3,4,5,7)$ Minimize and implement with NAND

**Solution**

Draw K-Map 1st

```
<table>
<thead>
<tr>
<th>x'y'</th>
<th>yz</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

$F = xy' + x'y + z$

**Implementation**
Multilevel NAND circuits

• Multilevel circuits conversion rules:
  1. Convert all AND gates to NAND with AND-invert symbols
  2. Convert all OR gates to NAND with invert-OR symbols
  3. Check all bubbles in diagram. For every bubble that is not compensated by another bubble, insert inverter.

• Example

![Diagram showing conversion of multilevel circuits from AND-OR gates to NAND gates]
Self Task

- Simplify the equation by simple gates & NAND

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

• Implement the following Boolean express using only NAND gates

\[ Y = ABC' + ABC + A'BC \]
\[ Y = AB + CDE + F \]
\[ Y = \sum (0,1,5) \]
Logic Operation with NOR gate

• NOR can also replace NOT, AND, OR

Inverter  \[ x \rightarrow x' \]

OR  \[ x \rightarrow y \rightarrow x + y \]

AND  \[ x \rightarrow y \rightarrow (x' + y')' = xy \]

• Two representations of NOR:
  – OR-invert and invert-AND

(a) OR-invert  \[ x \rightarrow y \rightarrow z \rightarrow (x + y + z)' \]

(b) Invert-AND  \[ x \rightarrow y \rightarrow z \rightarrow x'y'z' = (x + y + z)' \]
Converting to NOR Implementations

- Same rules as for NAND implementations

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

With NOR

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

• Implement the following Boolean expression using only NOR gates

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

\[ Y = \prod (0,1,2,4,6) \]
Ex-OR/NOR function

X-OR

\[
x \oplus y = x y' + x' y
\]

X-NOR

\[
(x \oplus y)' = x y + x' y'
\]
Ex-OR implementations

- In fig.(b), the first NAND gate performs the operation \((xy)' = (x' + y')\).

\[ xy' + x'y = x \oplus y \]

(a) With AND-OR-NOT gates

(b) With NAND gates

Exclusive-OR Implementations
Odd function

- Boolean expression of three-variable of the XOR:
  \[ A \oplus B \oplus C = (AB' + A'B)C' + (AB + A'B')C = AB'C' + A'BC' + ABC + A'B'C = \Sigma(1, 2, 4, 7) \]

Map for a Three-variable Exclusive-OR Function

(a) Odd function
\[ F = A \oplus B \oplus C \]

(a) Even function
\[ F = (A \oplus B \oplus C)' \]
Odd and Even functions

• The 3-input odd function is implemented by means of 2-input exclusive-OR gates.

\[ A \oplus B \oplus C = (AB' + A'B)C' + (AB + A'B')C \]

(a) 3-input odd function

(b) 3-input even function

Logic Diagram of Odd and Even Functions
Applications of Ex- OR/NOR

- Comparator
- Binary to Gray code convertor
- Adder & Subtractor circuits
- Parity generator
- And etc ...