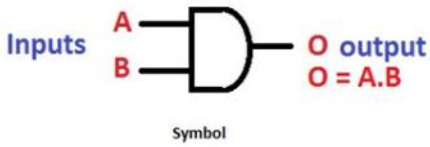
TRUTH TABLE FOR BASIC LOGIC GATE

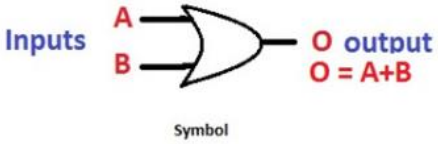
AND GATE:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



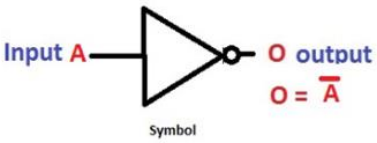
OR GATE:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



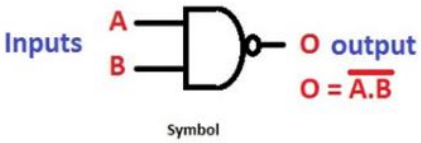
NOT GATE:

A	NOT A
0	1
1	0



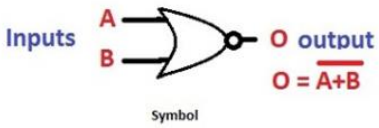
NAND GATE:

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0



NOR GATE:

1	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0



Symbol

Combining Gates to Form Complex Circuits

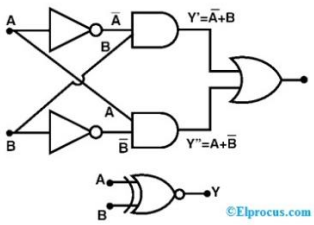
Logic gates can be combined to create complex circuits that perform advanced functions. For example:

- **AND + NOT** can create a NAND gate.
- **OR + NOT** can create a NOR gate.
- Combining multiple gates can create adders, multiplexers, and even entire processors.

The key idea is to use the outputs of one gate as inputs to another, building up the desired functionality step by step.

The XOR (exclusive OR) function outputs 1 when the inputs are different and 0 when they are the same. Its truth table is:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Ex-OR Logic Gates Formation

Step 1: Express XOR in Terms of Basic Gates

The XOR function can be expressed as:

$$A \oplus B = (A \cdot \overline{B}) + (\overline{A} \cdot B)$$

This means:

- AND A with NOT B.
- AND NOT A with B.
- OR the two results together.

Step 2: Implement Using NAND Gates

Since NAND is a universal gate, we can use it to create NOT, AND, and OR gates:

1. **NOT Gate**: Connect both inputs of a NAND gate together.
$$\overline{A} = \text{NAND}(A, A)$$
2. **AND Gate**: Use a NAND gate followed by a NOT gate.
$$A \cdot B = \overline{\text{NAND}(A, B)}$$
3. **OR Gate**: Use De Morgan's laws to express OR in terms of NAND.
$$A + B = \overline{\overline{A} \cdot \overline{B}} = \text{NAND}(\overline{A}, \overline{B})$$

Step 3: Construct the XOR Circuit

Using the above, the XOR circuit can be built as follows:

1. Compute $\overline{A}$ and $\overline{B}$ using two NAND gates.
2. Compute $A \cdot \overline{B}$ and $\overline{A} \cdot B$ using two more NAND gates.
3. Combine the results using a final NAND gate to achieve the OR operation.

Final Circuit

The circuit will require **4 NAND gates**:

1. NAND A and A to get $\overline{A}$.
2. NAND B and B to get $\overline{B}$.
3. NAND A and $\overline{B}$ to get $A \cdot \overline{B}$.
4. NAND $\overline{A}$ and B to get $\overline{A} \cdot B$.
5. NAND the results of steps 3 and 4 to get $A \oplus B$.

VERIFICATION:

You can verify this circuit by attempting all possible input combinations and verifying that the output is right based on the XOR truth table.

Summary

By breaking down the XOR operation into fundamental operations and using NAND gates to obtain NOT, AND, and OR, we can design a circuit to do XOR using only NAND gates. This shows the power of universal gates in building complex logic circuits.

REFRENCE: https://www.elprocus.com/basic-logic-gates-with-truth-tables/