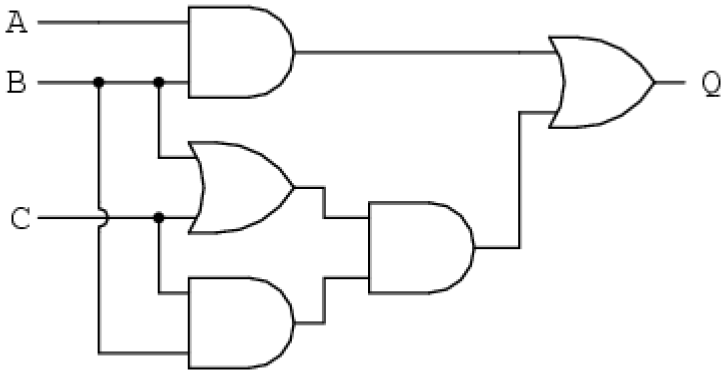
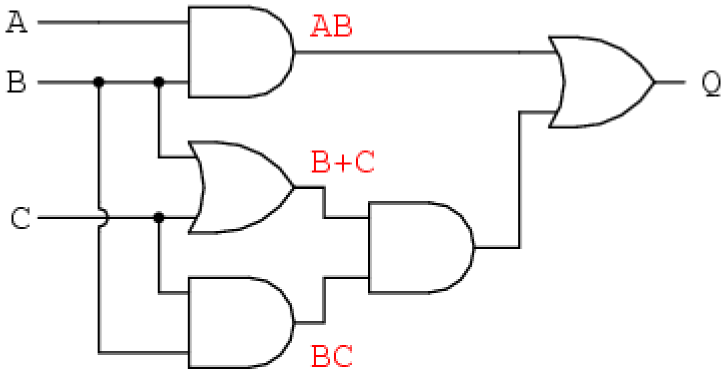
Circuit Simplification Example

Let's begin with a semiconductor gate circuit in need of simplification. The "A," "B," and "C" input signals are assumed to be provided from switches, sensors, or perhaps other gate circuits. Where these signals originate is of no concern in the task of gate reduction.

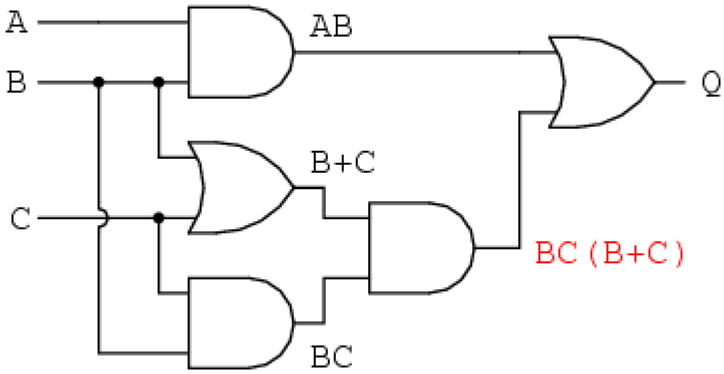


How to Write a Boolean Expression to Simplify Circuits

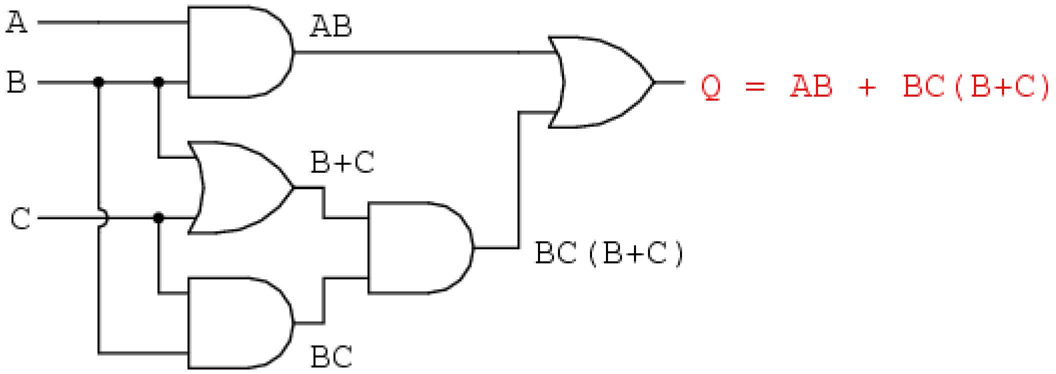
Our first step in simplification must be to write a Boolean expression for this circuit. This task is easily performed step by step if we start by writing sub-expressions at the output of each gate, corresponding to the respective input signals for each gate. Remember that OR gates are equivalent to Boolean addition, while AND gates are equivalent to Boolean multiplication. For example, I'll write sub-expressions at the outputs of the first three gates:



. . . then another sub-expression for the next gate:



Finally, the output (“Q”) is seen to be equal to the expression AB + BC(B + C):



Now that we have a Boolean expression to work with, we need to apply the rules of Boolean algebra to reduce the expression to its simplest form (simplest defined as requiring the fewest gates to implement):

$AB + BC(B + C)$

↓ Distributing terms

$AB + BBC + BCC$

↓ Applying identity **AA = A** to 2nd and 3rd terms

$AB + BC + BC$

↓ Applying identity **A + A = A** to 2nd and 3rd terms

$AB + BC$

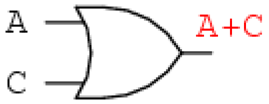
↓ Factoring **B** out of terms

$B(A + C)$

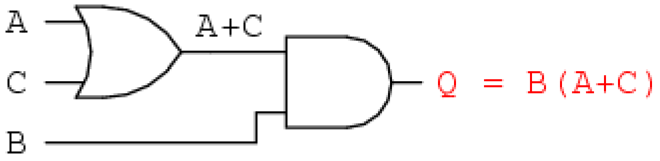
The final expression, B(A + C), is much simpler than the original, yet performs the same function. If you would like to verify this, you may generate a truth table for both expressions and determine Q's status (the circuits' output) for all eight logic-state combinations of A, B, and C, for both circuits. The two truth tables should be identical.

Generating Schematic Diagrams from Boolean Expressions

Now, we must generate a schematic diagram from this Boolean expression. To do this, evaluate the expression, following proper mathematical order of operations (multiplication before addition, operations inside parentheses before anything else), and draw gates for each step. Remember again that OR gates are equivalent to Boolean addition, while AND gates are equivalent to Boolean multiplication. In this case, we would begin with the sub-expression "A + C", which is an OR gate:



The next step in evaluating the expression "B(A + C)" is to multiply (AND gate) the signal B by the output of the previous gate (A + C):



Obviously, this circuit is much simpler than the original, having only two logic gates instead of five. Such component reduction results in higher operating speed (less delay time from input signal transition to output signal transition), less power consumption, less cost, and greater reliability.