**DeMorgan’s Theorem**

DeMorgan’s theorem allows the logic functions of NAND and NOR to be simplified. We do not have to go into in depth analysis of how DeMorgan arrived at his theorem, just be able to use the result which is surprisingly simple. The rules are as follows:

1. If you break a ‘bar’ change the sign underneath the break.
2. If you complete a ‘bar’ change the sign underneath where the bar is joined. (We will use this in Topic 1.2.4)

Let’s look at a couple of simple examples.

1. If we start with , DeMorgan’s theorem suggests that this can be written as . The ‘bar’ has been broken and the sign changed underneath the break in the bar. We can check this by looking at the truth table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **0** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **0** | **0** | **1** | **1** | **1** |
| **1** | **1** | **0** | **0** | **0** | **0** |

This shows that the two logic expressions are the same.

2. If we start with , DeMorgan’s theorem suggests that this can be written as . The ‘bar’ has been broken and the sign changed underneath the break in the bar. We can check this by looking at the truth table below.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| **0** | **0** | **1** | **1** | **1** | **1** |
| **0** | **1** | **1** | **0** | **0** | **0** |
| **1** | **0** | **0** | **1** | **0** | **0** |
| **1** | **1** | **0** | **0** | **0** | **0** |

This shows that the two logic expressions are the same.

The rule seems to work, according to the truth tables but we have only used very basic logic expressions here to prove the rule. If we looked at a more realistic problem then this would involve more terms and a more complex expression.

We will put a few of our new skills to the test by looking at a typical problem. You can work out some of the parts as we go through the example, but the correct solution will be provided at each stage so that you can check that you are on the correct path.Consider the following logic circuit.

**A**

**B**

**C**

**Q**

**D**

**Q** = …………………………………………………………….

Use the process we looked at earlier to derive a Boolean expression for the logic circuit shown above.

Remember to be very careful with the NAND gates.

The Boolean expression you should have arrived at is as follows:



This looks quite complicated but by using DeMorgan’s theorem we can make this a little bit easier. As a general ‘rule of thumb’ start from the top and work downwards. So the first thing that we have to do is to break to top ‘bar’ between the two brackets. This gives us the following:



If we look at the first term we can see that there is a double ‘bar’ over the expression. This means that the term is inverted and then inverted again, which will return the original state of the term. Therefore the **double inversion** as it is called can be removed.



We now apply DeMorgan’s theorem again to the second term to give the following:



Again we have a double inversion, applied only to the variable **C**, which can be removed to leave the final expression as:



If there were additional terms in the expression, this could now be simplified using the normal rules of Boolean algebra as discussed previously.

Now for a couple of examples for you to try!

**Exercise 5.** – Simplify the following expressions as much as possible.

1. 

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

2. 

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………

……………………………………………………………………………………………………………………………………