1. Boolean Notation for Logic Circuits

1.1 Introduction

We have already seen two distinct ways of representing a digital logic circuit.

- We can draw a circuit diagram, and we can construct a truth table.

Both of these representations have their uses:

- if you are intending to actually build the circuit, the diagram would be your starting point.
- Alternatively, if you want to see exactly what the circuit would do in all possible situations, you would use a truth table.

However, as an approach to communicating about circuits and their functionality, both representations have their problems.

- Drawing a circuit diagram is complex and the diagrams can get quite unwieldy, once you go beyond very simple functionality.
- Furthermore, it can take a certain amount of effort to see exactly what the circuit does.
- On the other hand, although the truth table tells you absolutely everything about the *behaviour* of a circuit, it doesn't tell you too much about how you might *construct* it using logic gates.

It seems that we need another representation for our circuit diagrams, one which:

- Is concise, allowing us to communicate it quickly and with less chance of introducing errors.
- Gives some indication of the behaviour of the circuit, if we read it properly.
- Gives us an indication of how the circuit could be constructed.

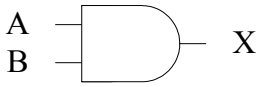
We will subsequently see that the notation we introduce in this class has other uses as well as the three that are listed above.

1.2 Boolean Notation

Boolean notation is founded on the fact that each of the basic gates has a symbol. Other symbols are built up from these.

1.2.1 AND Gate

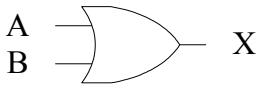
The symbol for an AND gate is a dot : *A.B*



Sometimes we omit the dot, so the symbol then becomes simply *AB*. Strictly speaking, the diagram above would be represented *X = A.B* or *X = AB*.

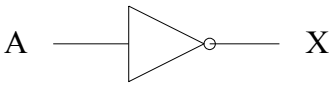
1.2.2 OR Gate

The symbol for an OR gate is a + sign : *X = A+B*



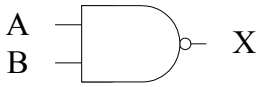
1.2.3 NOT Gate

The symbol for a NOT gate is a line over the input term: $X = \overline{A}$



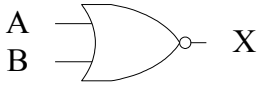
1.2.4 NAND Gate

The symbol for a NAND gate is a combination of the symbols for the AND gate and the NOT gate: $X = \overline{A.B}$ or $X = \overline{AB}$



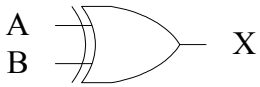
1.2.5 NOR Gate

The symbol for a NOR gate is a combination of the symbols for the OR gate and the NOT gate: $X = \overline{A + B}$



1.2.6 XOR Gate

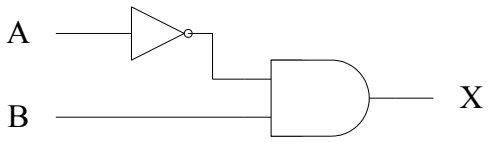
The symbol for an XOR gate is a + sign in a circle : $X = A \oplus B$



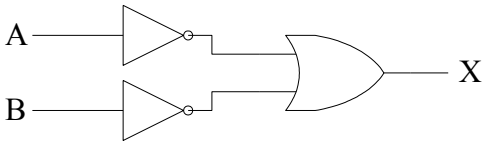
1.3 More Complex Circuits

The boolean symbols can be built up into *expressions* to describe more complex circuits.

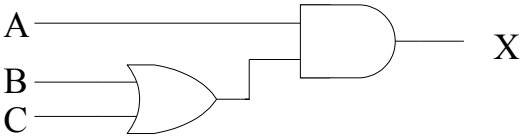
$X = \overline{A}.B$



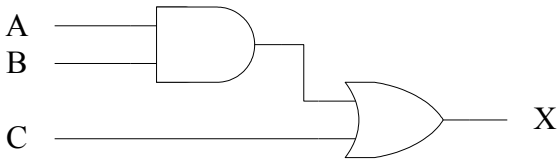
$X = \overline{A} + \overline{B}$



$X = A.(B + C)$

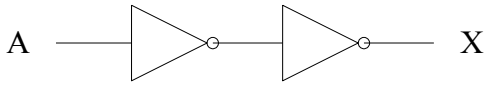


$X = AB + C$

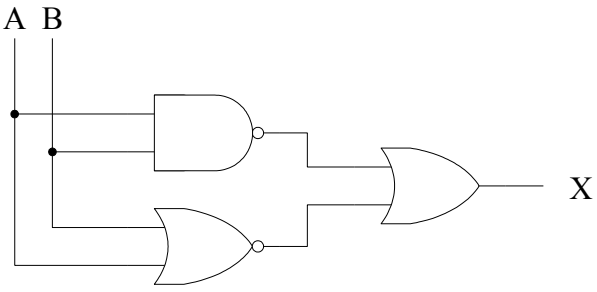


(Note the difference from the preceding example).

$X = \overline{\overline{A}}$

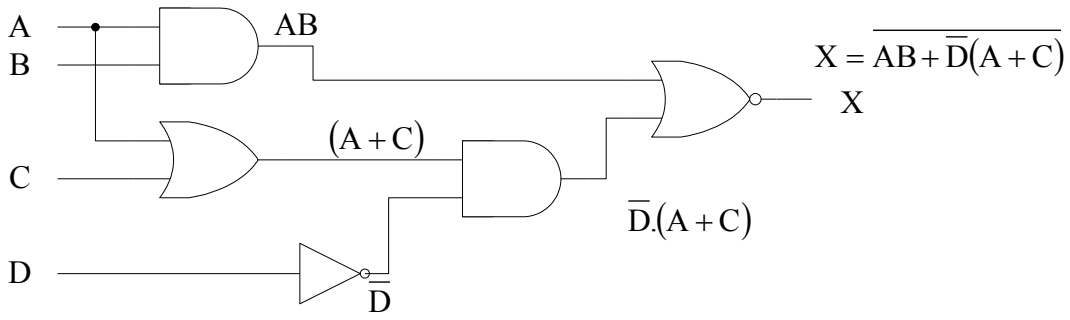


$X = \overline{AB} + \left(\overline{A + B}\right)$



When creating truth tables, we have used ***intermediate*** values to make finding the output easier.

We can use a similar technique to make it easier to derive the boolean expression of a circuit.

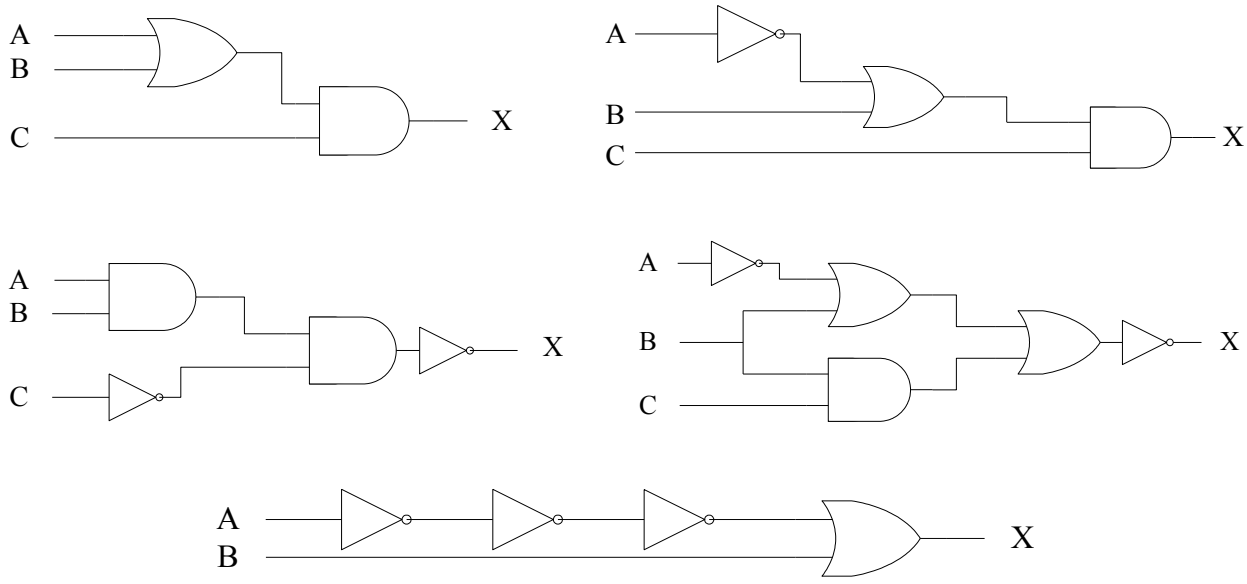


1.4 Exercises

1. Draw a circuit which has two inputs feeding into an AND gate and the output of the AND gate feeding into a NOT gate.
   - Label the inputs A and B, and label the output (after the NOT gate) X.
   - Draw a truth table for the circuit.

2. Draw a circuit which has two inputs feeding into an OR gate and the output of the OR gate feeding into a NOT gate.
   - Label the inputs A and B, and label the output (after the NOT gate) X.
   - Draw a truth table for the circuit.

3. Draw the truth tables and give the *Boolean expressions* for the circuits shown below.



4. Using truth tables prove the following:

   1. $A(B+C) = AB + AC$
   2. $A + AB = A$
   3. $A + \overline{A}B = A + B$
   4. $\overline{AB} = \overline{A} + \overline{B}$
   5. $\overline{A}.\overline{B} \neq \overline{AB}$
   6. $\overline{A}.\overline{B} = \overline{A + B}$
   7. $(A+B)(A+C) = A + BC$

LogiSim

This is a Digital logic Simulator which allows you to construct digital logic circuits and run them in a variety of ways. We shall be using it over the next few weeks.

1) Your tutor will instruct you as to the location of the package on the network.
2) Open the application, *if you click on the Help tab, there is an item for the Tutorial*.

5. Complete the tutorial; this will also give you the solution to ***SAQ 1.2*** in the lecture notes.

6. Using **Logisim**, construct and test the simple circuits in Ex 3

7. Using **Logisim** draw circuit diagrams for the following expressions, and check that the circuits output the correct values for all inputs:

   1. $A + BC$
   2. $AB + CD$
   3. $\overline{A}.\overline{B}$ (hint: this is *not* a ***NAND*** gate!)
   4. $\overline{\overline{A} + \overline{B}}$
   5. $Z = AB + CD + EF$ (hint: use a three input ***OR*** gate)
   6. $Y = (\overline{A} + B)(C + D)$
   7. $Y = A.(\overline{B} + C) + C.(\overline{BD})$
   8. $X = \overline{(\overline{\overline{A} + \overline{B}})(C + \overline{D})\overline{AB}.(\overline{A} + \overline{C} + \overline{D})}$