O/A/IB Computer Science
with
Shahbaz Tariq
0300 9204594

LOGIC GATES

Contents

Teaching and learning approaches

Introduction

This guide provides suggestions, ideas and activities for practitioners on the delivery of combinational logic, which forms one learning outcome of the Circuit Design unit in National 5 Practical Electronics.

The content also covers all the combinational logic topics required in Circuit Design at National 4 so this guide can also be used for this level but this will require more tutorial exercises to be developed.

Aims and objectives

This guidance is intended to support the practitioner in the delivery Practical Electronics: Circuit Design, Combinational Logic.
The materials provide guidance on the following three parts:

1. basic logic functions
2. converting truth tables to logic circuits and converting Boolean expressions to logic circuits
3. converting logic circuits to Boolean expressions.

Learners could be encouraged to work through the exercises both individually and in pairs.

Once the basic concepts have been learned, simulation packages and logic tutor boards could be used to enhance the learning and provide practical hands-on experience.

The guidance will support practitioners in giving the learners the opportunity to gain an understanding of the basic gates used in combinational logic, their truth tables and Boolean expression.

They could also be able to convert truth tables and Boolean expressions to logic circuits and convert logic circuits to Boolean expressions.

Introduction

The first part of the advice and guidance covers the basic logic gates and provides the ANSI 2 and the BS EN 60617 logic symbols, truth table and Boolean expression for each gate.

The second part describes how truth tables and Boolean expressions are converted into logic circuits.

The third part describes how logic circuits are converted into Boolean expressions.

Learners sometimes cannot see the relevance to real-life situations of combinational logic, so practitioners may make reference to how logic is used everyday, ie ‘I will buy you a coffee – not’ is an example of the NOT or invert logic function.

Practical examples of where logic is used could also be included, eg:

- Computers need combinational logic circuits to work.
- Modern cars have electronic control units (ECUs). These are small, powerful computers that control various functions within the car, such as the fuel management system.
- Televisions can have Freeview, which is a digital television signal that uses combinational logic.

Practitioners may wish to provide some other examples.

Suggested learning and teaching approaches

The skills and knowledge required will be gained as the learner progresses through each topic.

Monitoring progress

As each topic builds on the previous one, it is important that the learner is confident before moving on.

Learners therefore should be encouraged to take responsibility for their own progress so that they can work at their own pace and difficulties are identified as early as possible.

Suggested Learning and Teaching Approaches

These will be dependent on the facilities and expertise available in the centre but the following are suggested:

- worked examples for each topic
- tutorials to build up knowledge and understanding
- learner-centred building and testing on simulation software
- logic tutor-type boards individually and/or in pairs.

Resources

Some useful websites for basic combinational logic are:

http://www.play-hookey.com/digital/basic_gates.html
http://computer.howstuffworks.com/boolean1.htm
http://www.wisc-online.com/objects/ViewObject.aspx?ID=DIG1302
http://www.sci.brooklyn.cuny.edu/~goetz/projects/logic/combi.html
http://www.kpsec.freeuk.com/gates.htm

There are various software simulation packages available, such as Crocodile Clips, Electronic Workbench, MultiSim etc.

There are also a number of logic tutor systems available, for example the Feedback Logic tutor system.

It is suggested that learners use software simulation packages to build and test circuits. This is a safe method to use before they go on to use the more hands-on logic tutor systems.

Learners using simulation software can work either individually or in pairs. Circuits can be built and tested then saved. They can then be demonstrated at a later time.

The PowerPoint Presentation contains all the basic logic symbols and these can be used to make up further exercises if required.

Once the learners have gained experience using the simulation software, the circuits could be built and verified using logic tutor boards or whatever practical system the centre employs.

Because this is a practical learning environment, the relevant safety precautions must be observed.

A list of device numbers and pin connections for all the basic gates is provided in Appendix 1.

Basic logic functions

This section provides an introduction to the various logic gates AND, OR, NOT, XOR, NAND and NOR.

For each gate the logic symbol is provided along with the truth table and Boolean expression.

Both the ANSI 2 and the EN BS 60617 logic symbols are provided. However, after the tutorial questions in the first part of the document, only the ANSI 2 symbols will be used as they are more commonly used.

A short explanation of truth tables and Boolean algebra has also been included but will need further explanation or expanding during the lesson.

Learners often don't understand where the logic 1s and 0s for the inputs come from. A short refresher on binary arithmetic might be appropriate here so learners understand how to count from 0 to 3 and 0 to 7 in binary.

Another method sometimes used is to take the value of the binary column, 8, 4, 2, 1 etc, to determine the number of 0s then 1s together, ie starting from the top, the 8 column will have eight 0s followed by eight 1s, the 4 column will have four 0s followed by four 1s etc.

This is an easy method for producing the truth tables and allows the learner to concentrate on the output of the table rather than how to produce the input values.

In order to aid the learner's understanding, the concept for each logic gate, where possible, has been introduced using a simple electrical circuit using switches as the inputs and a light as the output. It would be beneficial to the learner if, during teaching, more examples could be given so reinforcing the idea that logic is used in all sorts of areas.

For example, a two-way lighting circuit used on staircases with one switch at the top and one at the bottom is an example of the XOR function.

The PowerPoint Presentation contains all the logic symbols used in these notes and these can be used to develop more tutorial exercises if required.

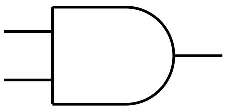
An internet search will provide a number of examples of simple uses of combinational logic circuits.

After the learners have gone through the tutorial exercises, the basic logic functions should be reinforced using simulation software such as Electronic workbench, MultiSim or whatever software package the centre uses to test logic functions.

This will provide added value to the learner’s knowledge.

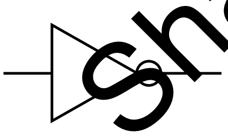
Tutorial

1. Identify the logic gate from the logic symbol.

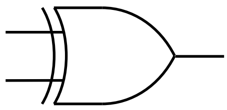


AND

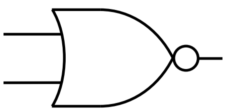
OR



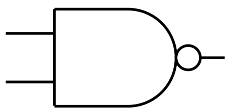
NOT



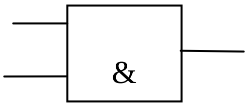
XOR



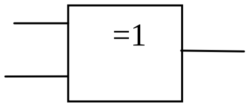
NOR



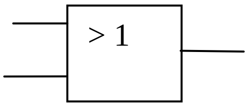
NAND



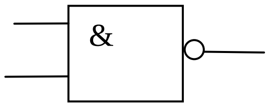
AND



XOR



OR



NAND

Learners could also be asked to write the Boolean expression for each gate.

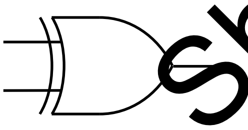
2. Draw the logic gate and write the Boolean expression described by the following truth tables.

B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	0

XOR

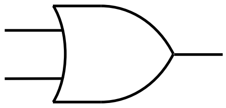
B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	1

OR



XOR logic symbol

O/P = A $\oplus$ B

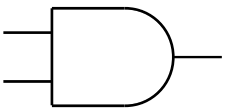


OR logic symbol

O/P = A + B

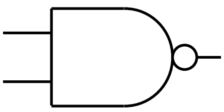
B	A	O/P
0	0	0
0	1	0
1	0	0
1	1	1

AND



B	A	O/P
0	0	1
0	1	1
1	0	1
1	1	0

NAND



AND logic symbol

O/P = A.B

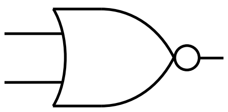
NAND logic symbol

O/P = $\overline{A.B}$

B	A	O/P
0	0	1
0	1	0
1	0	0
1	1	0

A	O/P
0	0
1	1

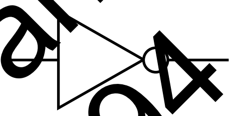
NOR



NOR logic symbol

O/P = $\overline{A + B}$

NOT



NOT logic symbol

O/P = $\overline{A}$

3. Determine the logic function represented by the following Boolean expressions:

O/P = $\overline{P}$

NOT

O/P = A.B.C

AND

O/P = P + Q

OR

O/P = $\overline{A.B}$

NAND

O/P = $\overline{A + B}$

NOR

O/P = $P \oplus Q$

XOR

Learners could also be asked to produce the truth table for each expression.

Converting from truth tables to logic circuits

This section builds on the knowledge of the basic gates by first taking the truth table and explaining that only lines on the truth table where the output is a logic 1 are required to design the logic circuit. This is because the circuit has to produce a logic 1 output for only these input conditions.

It could be highlighted that the inputs to the circuit are tied together on a truth table by the Boolean AND function. It could also be mentioned, that this AND function is called the 'product of sums' form.

Following on from this, all the lines on a truth table that have a logic 1 output are tied to each other by the Boolean OR function. This is called the 'sum of products' form. Boolean expressions produced from a truth table will always be in the sum of products form.

The practitioner could emphasise the following:

- Because the circuit has only one output, there can only be one logic gate producing this output. Because the Boolean expression is in the sum of products form, the output gate will be an OR gate if more than one line on the truth table is a logic 1. However, if only one line on the truth table is a logic 1 then the output gate will be an AND gate.

- The number of product terms, ie the number of lines on the truth table whose output is logic 1, also determines how many inputs there will be into the output gate.

- It doesn't matter what order the inputs are written in; A.B.C is exactly the same as C.A.B or B.C.A.

- A + B + C is the also same as B + A + C etc.

Practitioners should emphasise to learners that any type of electrical circuit diagram flows from left to right on the page, ie signals input on the left and output on the right. Also, when drawing logic circuits or any kind of electrical diagram, wiring must always be drawn either horizontally or vertically, never at an angle.

Once the learners have progressed through the course the circuits could then be constructed and tested first using a simulation package, then with a logic tutor.

When using the logic tutor system, learners should be instructed to always connect all unused inputs to logic 1. This prevents problems occurring and is also good practice.

Example

Design a logic circuit for the following truth table.

R	P	Q	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

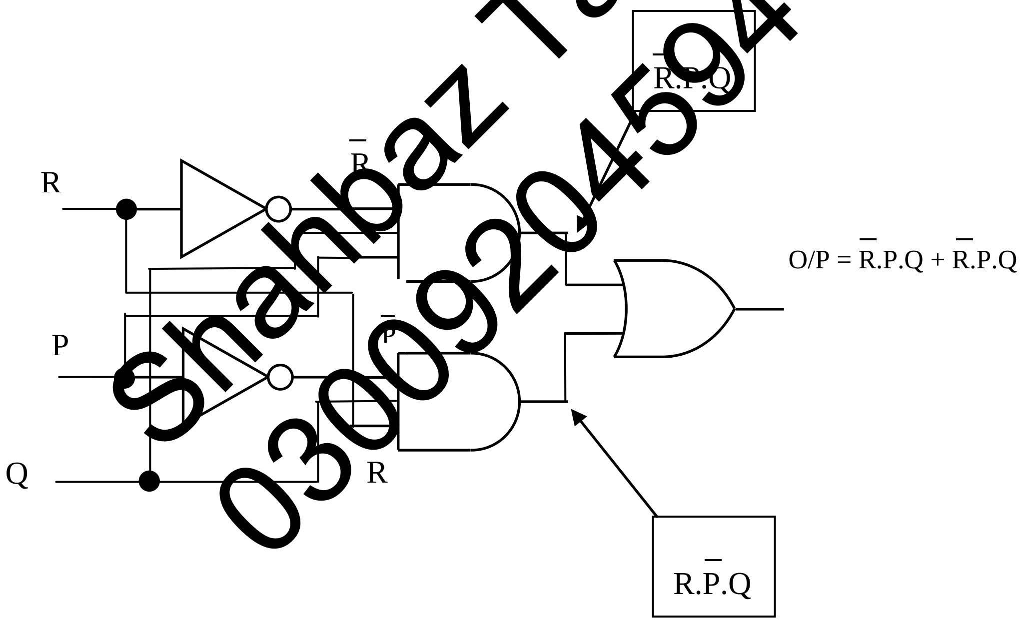
The two lines that have the output at logic 1 are $\bar{R}.P.Q$ and $R.\bar{P}.Q$.
These are the two products of sum expressions.

The output is therefore O/P = $\bar{R}.P.Q + R.\bar{P}.Q$

The output gate is an OR gate with two inputs. One input is $\bar{R}.P.Q$ and the other is R.P.Q.

The circuit can now be built up from the output gate back to the input.

Although the complete logic circuit is shown below, it should be built up in stages from the output gate back to demonstrate how it is constructed.



Tutorial: Logic circuits from truth tables

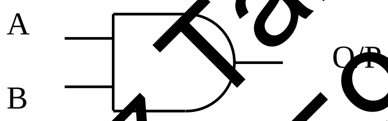
Draw the logic circuit described by the truth tables below:

1.

B	A	O/P
0	0	0
0	1	0
1	0	0
1	1	1

AND truth table

O/P = A.B

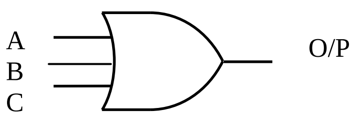


2.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

OR truth table

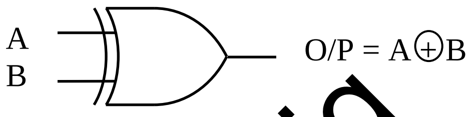
O/P = A + B + C



3.

B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	0

O/P = A + B◯

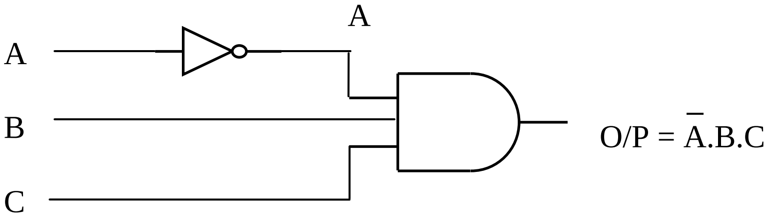


An XOR gate

4.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

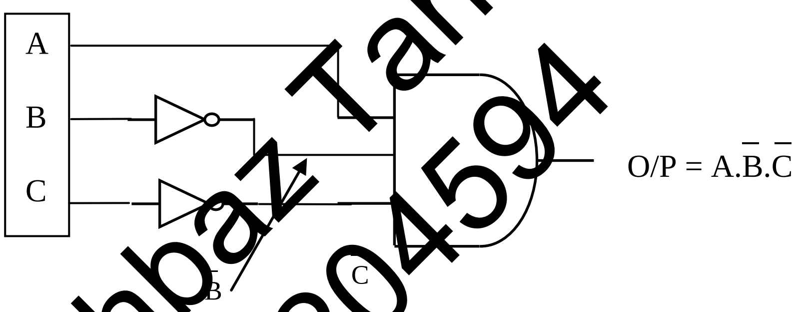
$O/P = \bar{A}.B.C$



5.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

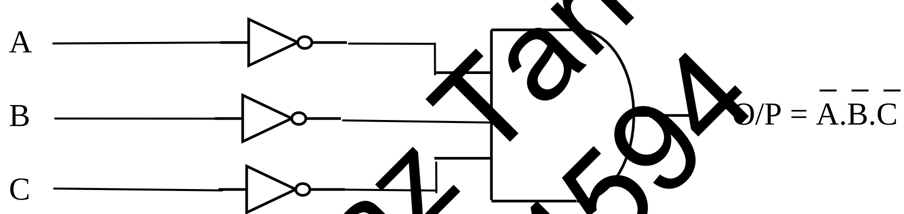
$$O/P = A.\bar{B}.\bar{C}$$



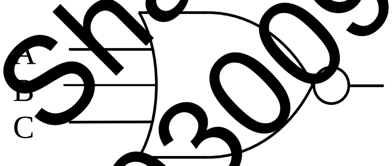
6.

C	B	A	O/P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$$O/P = \bar{A}.\bar{B}.\bar{C}$$



By looking at the truth table it can be seen that it is for a three-input NOR gate and could have been written as:



$$O/P = \overline{A + B + C}$$

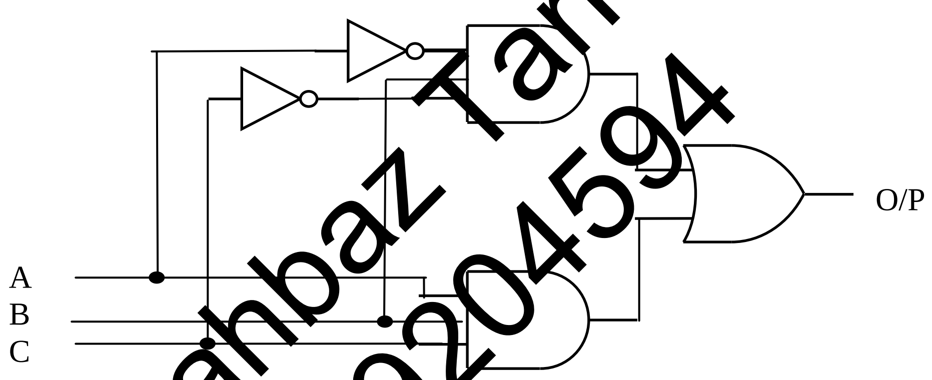
It should be pointed out that both circuits are correct but the NAND gate is used in a practical circuit as it uses fewer devices.

Although it is not part of this course, the above diagram illustrates one of De Morgan’s theorems in a practical circuit, ie move the inversion(s) from the input to the output and change the gate from AND to OR.

7.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

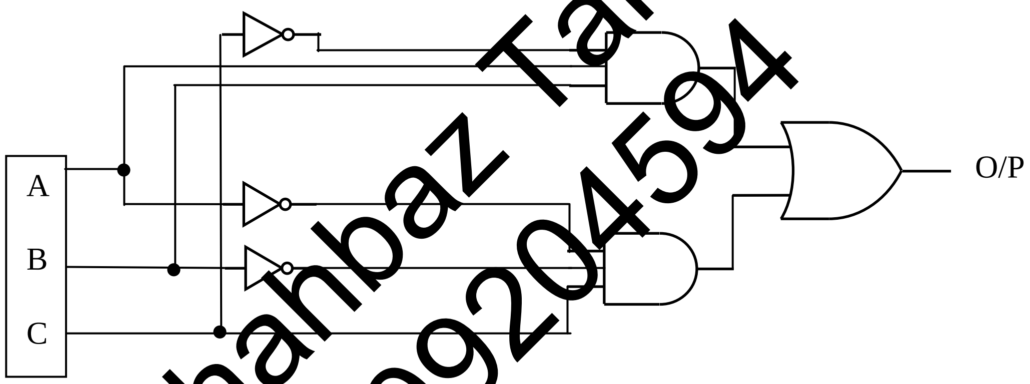
$$O/P = \bar{A}.B.\bar{C} + A.B.C$$



8.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

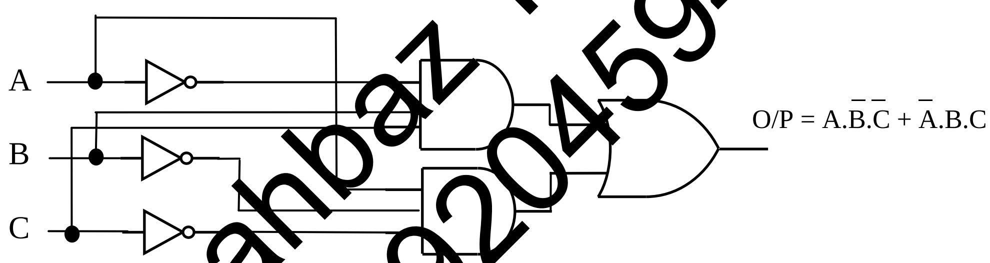
$$O/P = A.B.\overline{C} + \overline{A}.\overline{B}.C$$



9.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$O/P = A.\bar{B}.\bar{C} + \bar{A}.B.C$$

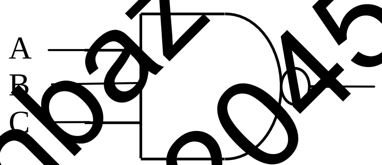


10.

R	Q	P	O/P
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$$O/P = \overline{A.B.C}$$

Note: It might be advisable to tell the learners to look closely at the truth table before designing the logic circuit.

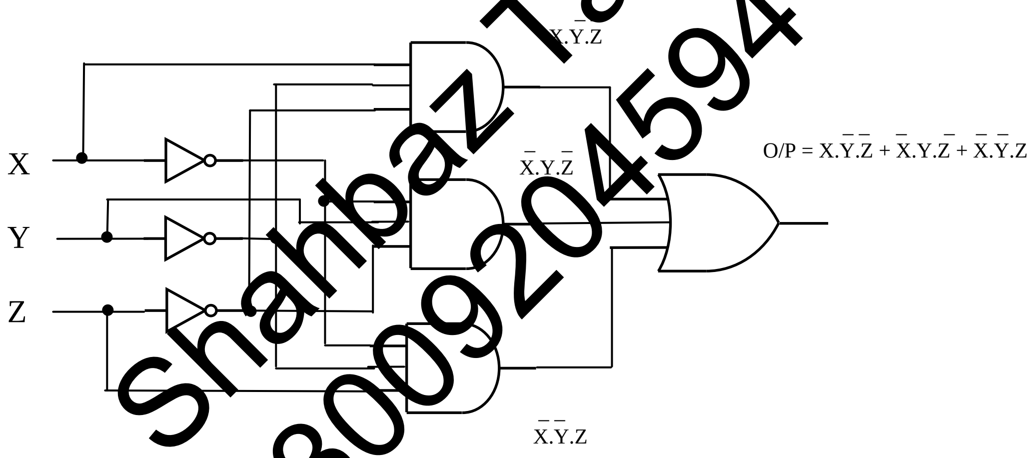


$O/P = \overline{A.B.C}$

11.

Z	Y	X	O/P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

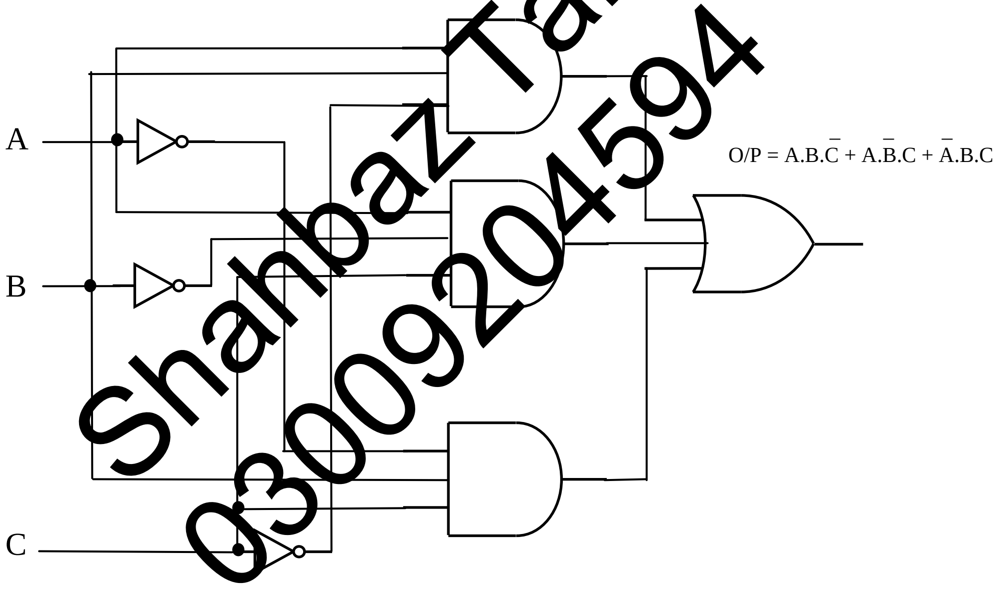
$$O/P = X.\bar{Y}.\bar{Z} + \bar{X}.Y.\bar{Z} + \bar{X}.\bar{Y}.Z$$



12.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$O/P = A.B.\overline{C} + A.\overline{B}.C + \overline{A}.B.C$$



Logic circuits from Boolean expressions

This technique is harder for learners as the output gate can be any of the basic gates. Learners often find difficulty in deciding what the output gate should be and the expression doesn't need to be very complex for this to happen.

Analysing expressions to deduce the output gate, without drawing the logic circuit, is good practice for learners and centres could produce a range of examples for this purpose.

Encouraging learners to use brackets, as they would for ordinary algebra, also greatly helps them to determine the output gate.

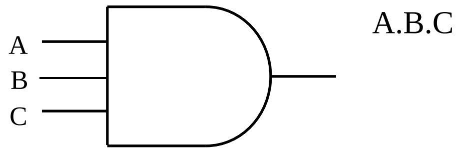
Again, learners should be taught to work back from the output gate, working out the logic for each branch back to the input. With practice, the logic diagrams will get tidier and clearer.

The learners should build and test the circuits using a simulation package. Logic tutor systems should then be used to build and test the circuits.

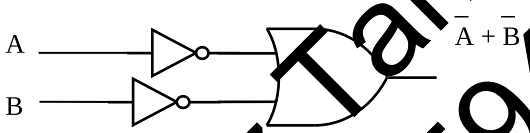
Logic circuits from Boolean expressions

Draw the logic circuit for each of the Boolean expressions given below.

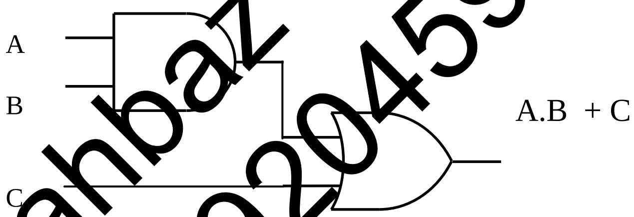
1. A.B.C



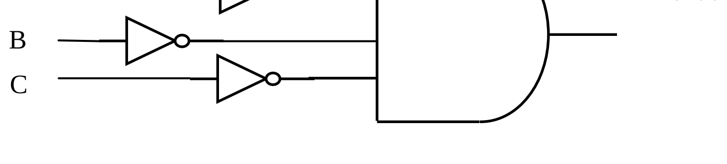
2. $\overline{A} + \overline{B}$



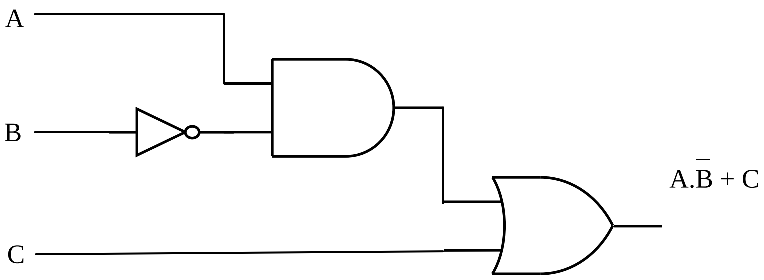
3. A.B + C



4. $\overline{A}.\overline{B}.\overline{C}$

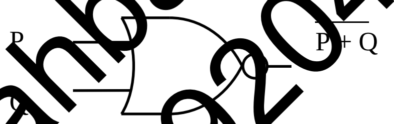


5. $A.\overline{B} + C$



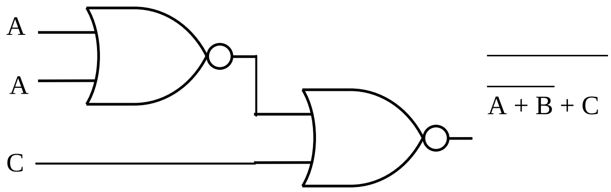
6. $\overline{P + Q}$

This is a two-input NOR gate, indicated by the bar above the whole expression.

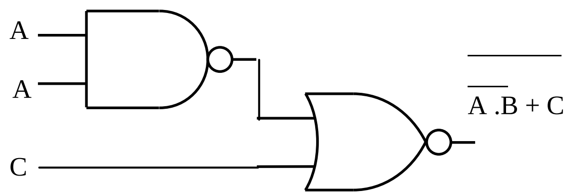


7. $\overline{\overline{A + B} + C}$

The output gate in this case is a NOR. The bar above the A + B groups them together as one input to the NOR gate.

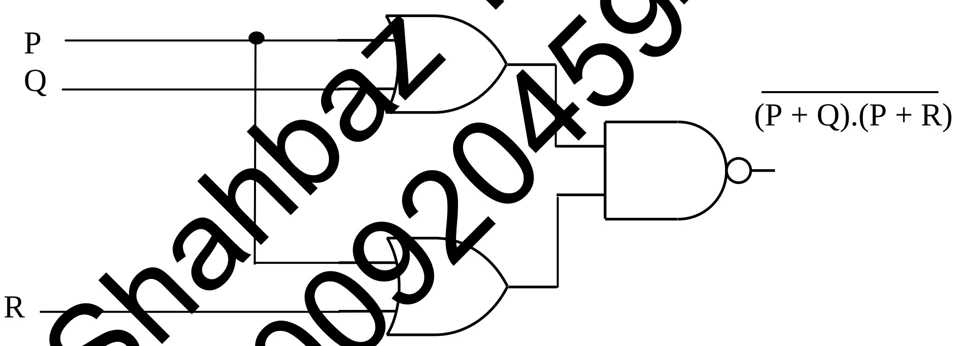


8. $\overline{\overline{A}.B + C}$

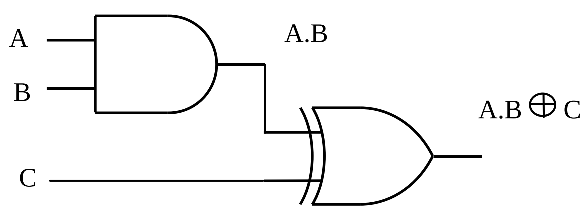


9. $\overline{(P + Q).(P + R)}$

The output gate is NAND.



10. $(A.B) \oplus C$



Boolean expressions and truth tables from logic circuits

Conversion of a logic circuit into a Boolean expression is very straightforward. Simply start at the input to the circuit and at the output of every gate write down on the logic diagram the logic expression for that gate based on the inputs.

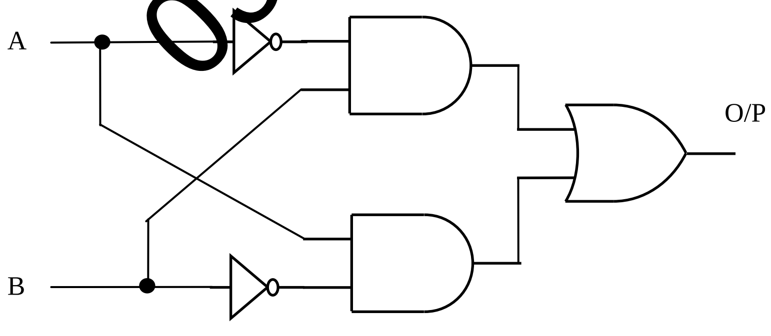
Continue this until the expression for the output gate has been derived.

Common mistakes:

- Learners do not pay attention to the type of gate.
- They mix up AND and OR functions.
- They often forget that a circle on the output of a gate means that there has been an inversion and there should be a bar over the whole of the output of that gate.
- They recognise that the gate has an inversion but put the bar on each of the inputs individually.
- They correctly derive the output of a gate but do not use the complete expression as the input to the next gate, ie they miss out the bar above an expression.

The importance of truth tables in determining the operation of logic circuits cannot be understated. It is often very difficult to understand the function of a logic circuit from the diagram or the Boolean expression. However, a truth table can help greatly in clarifying the function of the circuit.

For example, take the following logic circuit:



The output expression for this circuit is $A.\overline{B} + \overline{A}.B$.

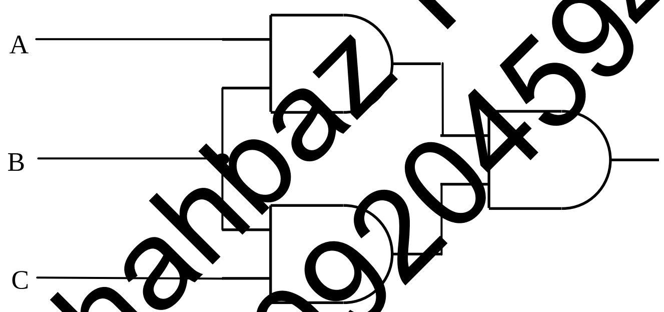
Anyone who has logic experience will recognise this as an XOR function, but learners may struggle to see this.

However, the truth table for the circuit is:

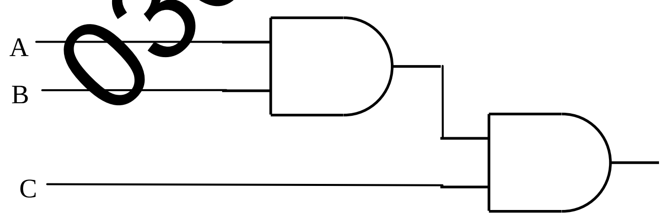
B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	0

Learners should now be able to see that this is the truth table for the XOR gate studied earlier.

The circuit in the diagram below uses two-input AND gates to provide a three-input AND gate.



This is one method of producing a three-input AND gate.

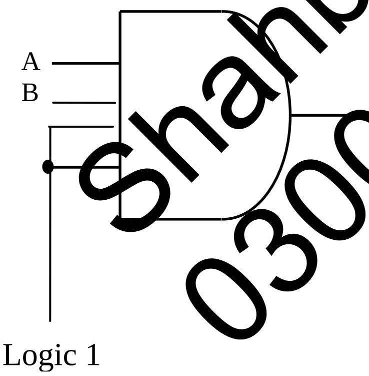


A truth table drawn for both circuits will show the logic function clearly:

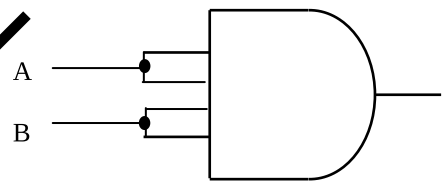
C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

This is the truth table for both circuits above.

Similarly, a three- or four-input gate can be used as a two-input device by using two of the inputs and tying the other two permanently to logic 1. Alternatively, it could be used by tying two inputs together as one input and doing the same with the other two:



Tying two inputs to logic 1



Using two inputs tied together for each input

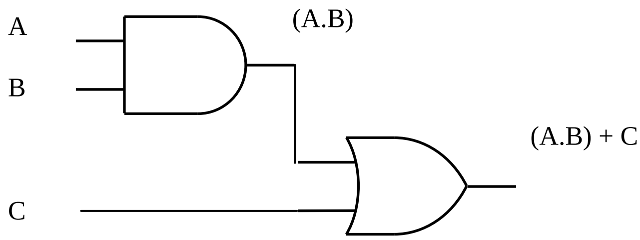
All the circuits in this section should be built and tested using the logic tutor boards.

Questions 2, 4 and 11 in the tutorial can be simplified. This should be highlighted to the learner.

Conversion of logic circuits to Boolean expressions

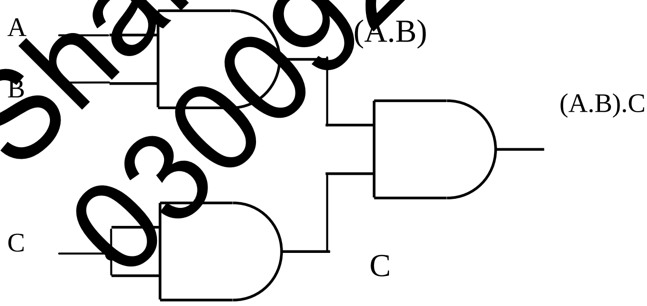
Derive the truth table and Boolean expression for the output of the following logic circuits.

1.



C	B	A	(A.B)	(A.B) + C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

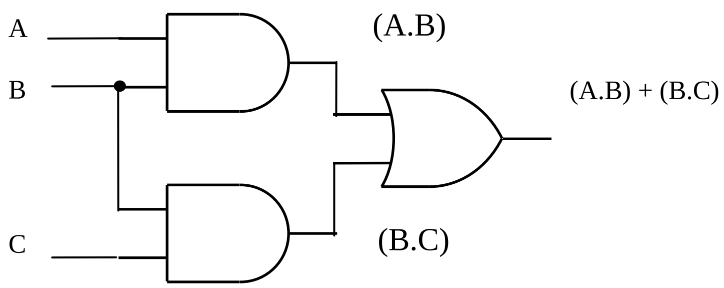
2.



C	B	A	(A.B)	(A.B). C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

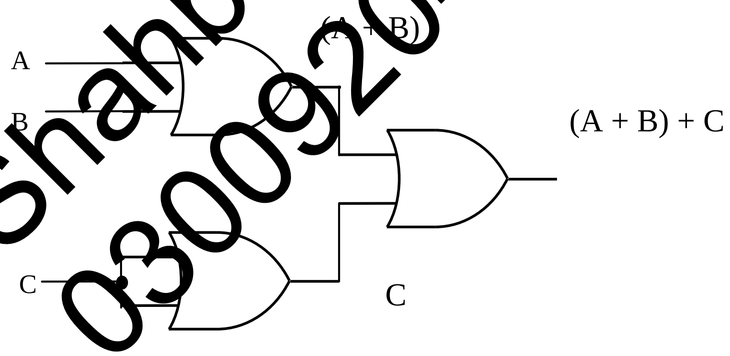
This circuit can be replaced by a three input AND gate

3.



C	B	A	(A.B)	(B.C)	(A.B) + (B.C)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	1	1	1

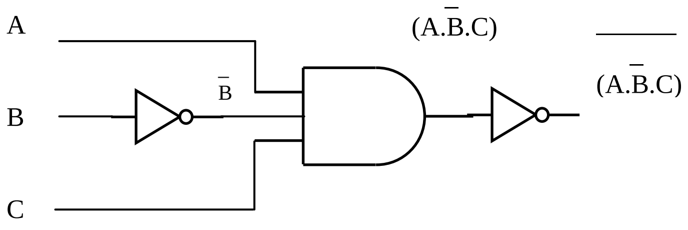
4.



C	B	A	(A + B)	(A.+ B) + C
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

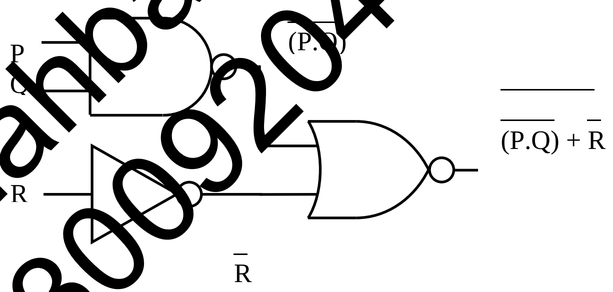
This circuit can be replaced by a three-input OR gate.

5.



C	B	A	$\bar{B}$	$A.\bar{B}.C$	$\overline{A.\bar{B}.C}$
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	[illegible]	1	0
1	1	0	0	0	1
1	1	1	0	0	1

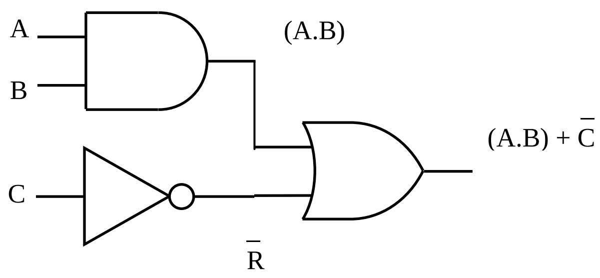
6.



R	Q	P	$\overline{P.Q}$	$\bar{R}$	$\overline{\overline{P.Q} + \bar{R}}$
0	0	0	1	1	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	0	0	1

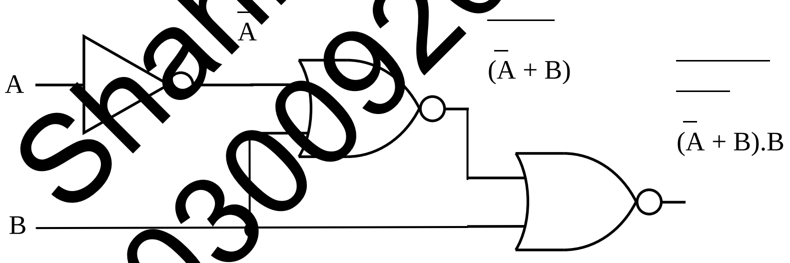
It is hard to see from the logic diagram or the Boolean expression, but the truth table shows that this circuit can be replaced by a three-input AND gate.

7.



C	B	A	(A.B)	$\overline{C}$	(A.B) + $\overline{C}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1

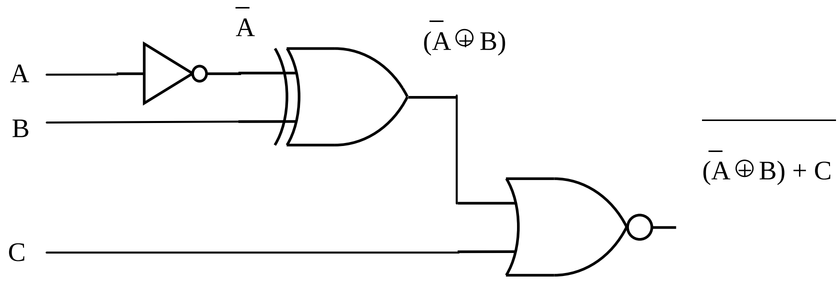
8.



B	A	$\overline{A}$	$\overline{\overline{A} + B}$	$\overline{(\overline{A} + B)} + B$	$\overline{\overline{(\overline{A} + B)} + B}$
0	0	1	0	0	1
0	1	0	1	1	0
1	0	1	0	1	0
1	1	0	0	1	0

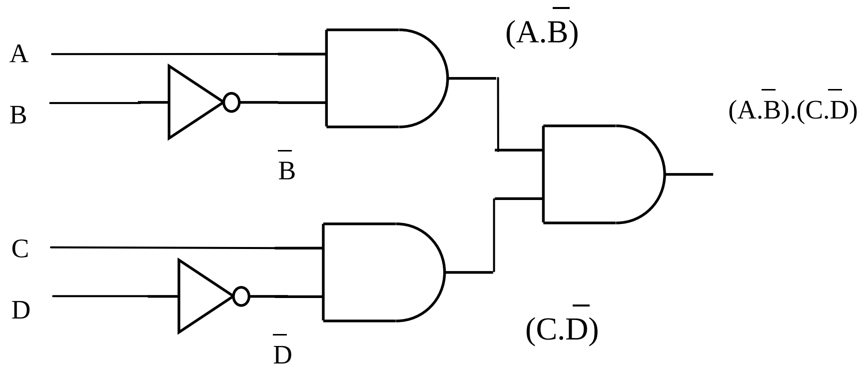
This circuit can be replaced by a two-input NOR gate.

9.



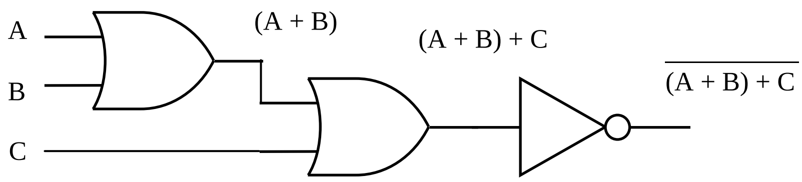
C	B	A	$\bar{A}$	$(\bar{A} \oplus B)$	$(\bar{A} \oplus .B) + C$	$\overline{(\bar{A} \oplus .B) + C}$
0	0	0	1	1	1	0
0	0	1	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	1	1	0

10.



D	C	B	A	$\bar{B}$	$\bar{D}$	$A.\bar{B}$	$C.\bar{D}$	$(A.\bar{B}).(C.\bar{D})$
0	0	0	0	1	1	0	0	0
0	0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0	0
0	0	1	1	0	1	0	0	0
0	1	0	0	1	1	0	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	0	1	0
0	1	1	1	0	1	0	1	0
1	0	0	0	1	0	0	0	0
1	0	0	1	1	0	1	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0
1	1	0	1	1	0	1	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0

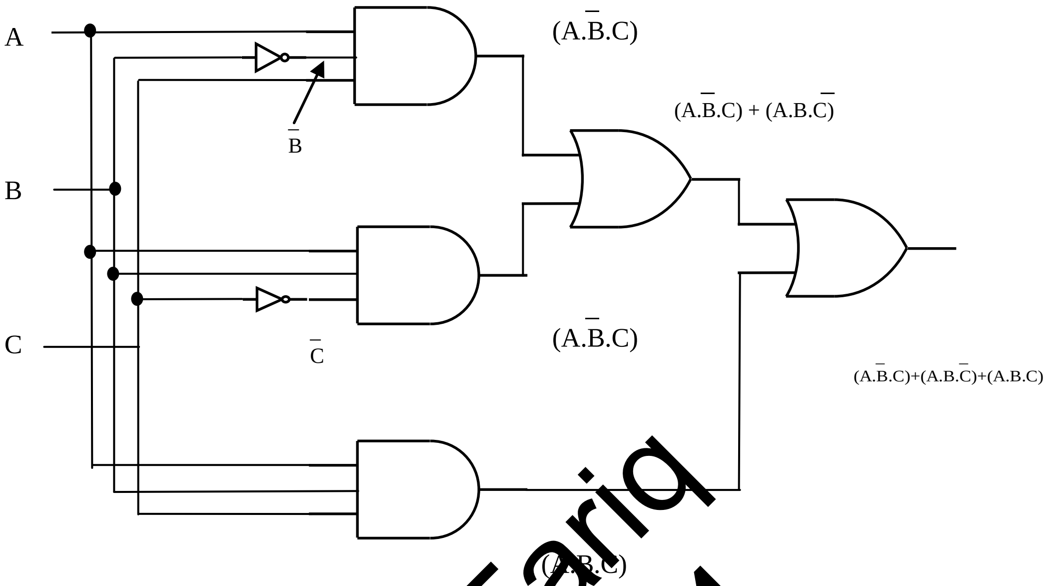
11.



C	B	A	(A + B)	(A + B) + C	$\overline{(A + B) + C}$
0	0	0	0	0	1
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	0

This circuit can be replaced by a three-input NOR gate.

12.



C	B	A	$\overline{B}$	$\overline{C}$	$(A.\overline{B}.C)$	$(A.B.\overline{C})$	(A.B.C)	$(A.\overline{B}.C) + (A.B.\overline{C})$	$(A.\overline{B}.C) + (A.B.\overline{C})$ (A.B.C)
0	0	0	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	1	1	0	1	0	1	0	1	1
1	0	0	1	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1	1
1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	0	1

Combinational logic circuits

Logic functions

Logical operations are used in everyday life. Statements such as ‘I will buy you a coffee – not’ are common. In this statement the person is saying something positive then making it into the opposite with the ‘not’ at the end of the sentence. Other logical operations are things such as:

- To use your computer system the computer must be switched on *and* the monitor must be switched on.

- To get out of the house you can exit by the front door or the back door.

Logic systems have only two states and they can be represented by various methods, ie high or low, up or down, 1 or 0, on or off, true or false etc.

Computer systems use logical operations and electrical circuits are used to produce these logic functions. Identification of the function is by a symbol and there are different symbols used for each of the basic logic functions.

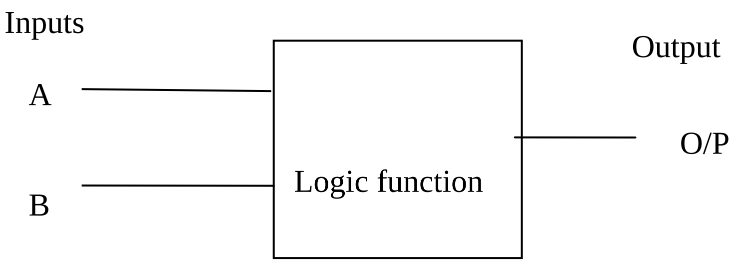
Two sets of symbols are used: ANSI symbols and IEC symbols. Both will be shown for the basic gates then the ANSI symbols will be used for the rest of the module.

All logic gates use electrical signals being either on or off to represent the logic states 1 and 0, ie a signal present is represented by logic 1 and a signal not present by logic 0.

The basic logic functions are termed gates, and gates are combined together to produce logic circuits.

The basic logical functions (gates) are AND, OR, NOT, XOR, NAND and NOR.

They can have one or more inputs but only one output.



A two-input logic function

All logical functions or circuits, from the simplest to the more complex, can be represented in various forms. Truth tables are one form, Boolean algebra is another.

Both of these describe the behaviour of the logic circuit.

Truth tables

A truth table is a table that lists all combinations of inputs along with their corresponding output. In combinational logic an input combination will always produce the same output condition irrespective of what has been present before, ie if an input of logic 1 on input A and logic 0 on input B produced an output of 1 it will always produce that output for that logic function.

An example of a truth table for a two-input logic circuit is shown below:

Input A	Input B	Output
0	0	0
0	1	1
1	0	0
1	1	1

The inputs for truth tables are simply a binary count from 0 to all 1s, for example the above truth table starts at input A = 0 and input B = 0 and counts up by one on every line until both inputs are at logic 1. The count in binary is therefore 00, 01, 10, 11 and the count in decimal is 0, 1, 2, 3.

If the logic function had three inputs then the count would start at 000 and go up to 111. In decimal this would be 0 to 7.

Boolean algebra

Boolean algebra is a mathematical method of representing logic functions and can be manipulated in a similar way to ordinary algebra.
It uses letters or names for the inputs A, B, C or P,Q etc. that are connected to the gate and connects them using special symbols to represent the logic function.

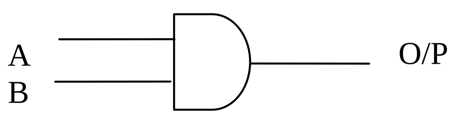
The basic logic gates are provided in the following pages with their truth tables and Boolean expressions.

AND gate

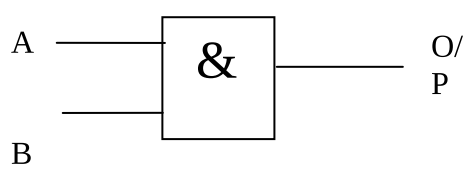
The easiest way to show the AND logic gate operation is by using an electrical circuit:

1 = switch on   0 = switch off

As can be seen, the lamp will only light when both switches are on.



ANSI two-input AND gate symbol



BS EN 60617 symbol for a two-input AND gate

The truth table below shows that the output y will be at logic 1 only when both the inputs are at logic 1.

A	B	O/P
0	0	0
0	1	0
1	0	0
1	1	1

Truth table for a two-input AND gate

Boolean expression

The other method to show the function of the circuit is to use Boolean algebra. This uses the names of the inputs connected with symbols that represent the function.

In the case of the AND function the Boolean operator is a full stop.

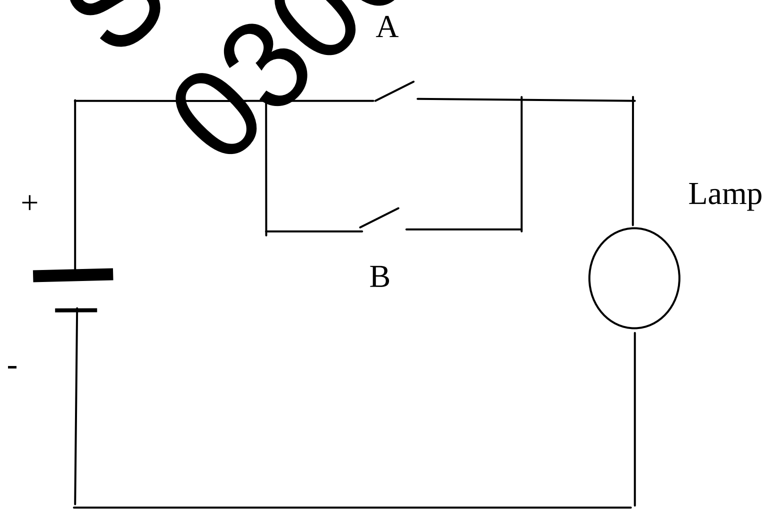
The Boolean expression for the two-input AND gate above is:

O/P = A.B

The output is logic 1 when A is at logic 1 *and* B is at logic 1.

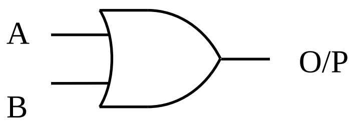
OR gate

The easiest way to show the OR logic function is by an electrical circuit.

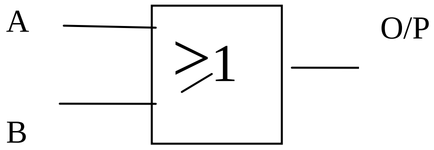


1 = switch on   0 = switch off

As can be seen, the lamp will light when either or both switches are on.



ANSI symbol for a two-input OR gate



BS EN 60617 symbol for a two-input OR gate

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	1

Truth table for a two-input OR function (gate)

Boolean expression

The other method to show the function of the circuit is to use Boolean algebra.

The symbol for the OR operator is a plus (+) sign, not to be confused with the addition sign.

The Boolean expression for the two-input OR gate above is:

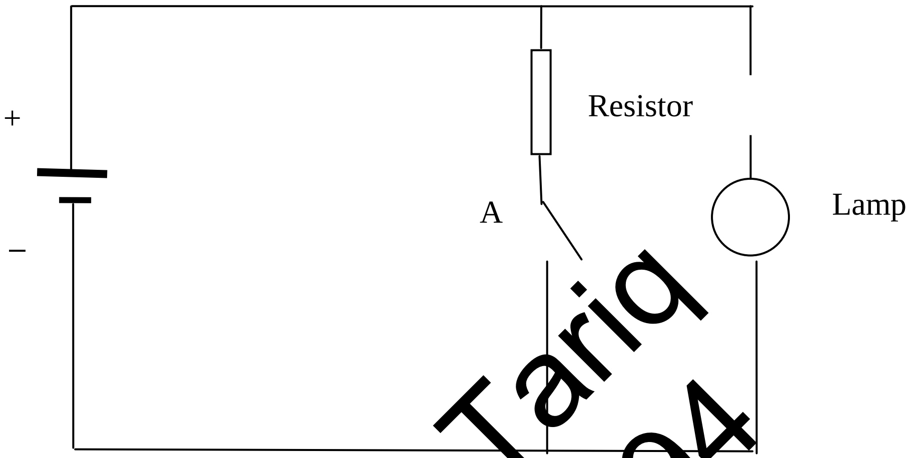
O/P = A + B

This is read as O/P = 1 when A is 1 *or* B is 1 *or* both are 1.

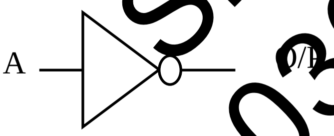
This is more correctly termed the inclusive OR function as it includes all the conditions: A or B or both.

NOT gate

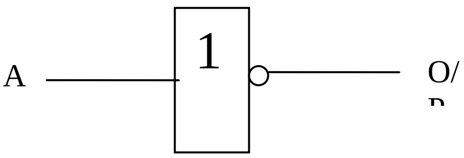
The electrical circuit for the NOT function is shown below. It is also called an inverter gate.



When the switch is off (logic 0) the lamp is on, ie logic 1.
When the switch is on (logic 1) the lamp is off, ie logic 0.
The output is therefore the opposite of the input.



ANSI symbol for a NOT gate



BS EN 60617 symbol for a NOT gate

Notice the bubble on the output of the NOT gate. This bubble signifies that an inversion has taken place (ie a logic 0 becomes a logic 1 and a logic 1 becomes a logic 0).

A	O/P
0	1
1	0

Truth table for a NOT gate

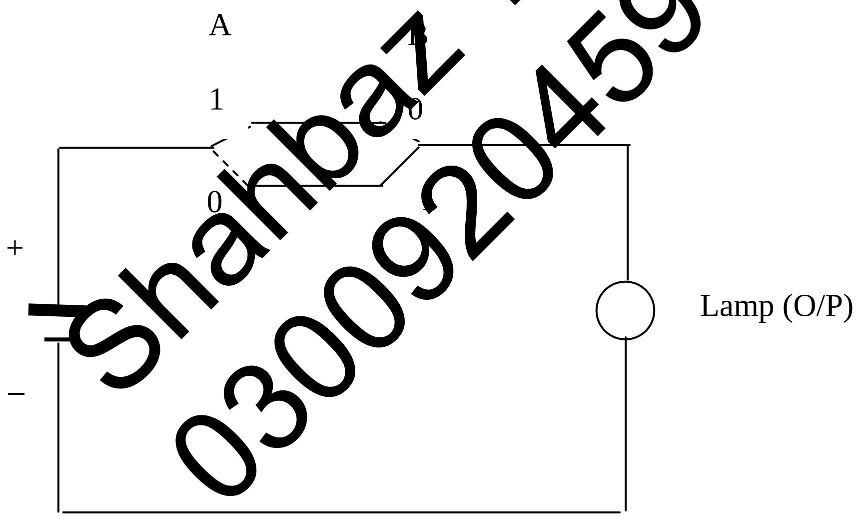
Boolean expression

The Boolean expression for a NOT operator is obtained by placing a line called a bar above the function:

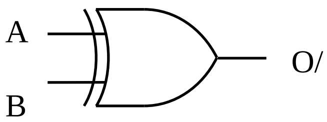
$O/P = \overline{A}$

This is read as the output = not A, ie the output is the opposite of the input A.

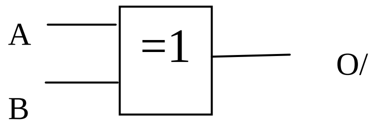
Exclusive OR function (XOR)



As can be seen, the lamp will light when switch A is in position 1 *and* switch B is in position 0 *or* when switch A is in position 0 *and* switch B is in position 1.



ANSI symbol for an exclusive (XOR) OR gate



BS EN 60617 symbol for an exclusive (XOR) OR gate

A	B	O/P
0	0	0
0	1	1
1	0	1
1	1	0

Truth table for a XOR function (gate)

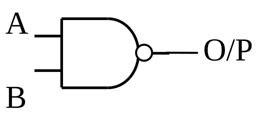
Boolean expression

In the case of the XOR function the Boolean operator is a plus sign inside a circle:

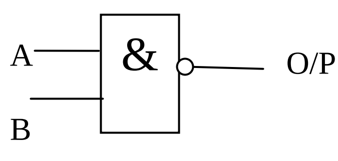
O/P = A $\oplus$ B

The NOT AND or NAND gate

This is an AND gate with the output inverted.



ANSI symbol for a two-input NAND gate



BS EN 60617 symbol for a two-input NAND gate

Note the bubble, which indicates an inversion.

A	B	O/P
0	0	1
0	1	1
1	0	1
1	1	0

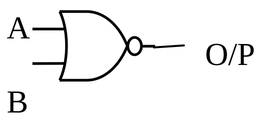
Truth table for a two-input NAND gate

The Boolean expression takes the output of the AND gate and inverts it:

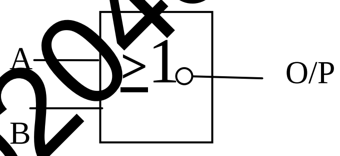
O/P = $\overline{A.B}$

NOT OR or NOR gate

This is an OR gate with the output inverted.



ANSI symbol for a two-input NOR gate



BS EN 60617 symbol for a two-inout NOR gate

Again, note the bubble, which indicates an inversion.

A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	0

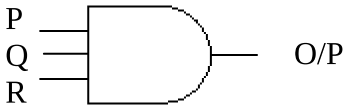
Truth table for a two-input NOR gate

The Boolean expression takes the output from an OR gate and inverts it:

O/P = $\overline{A + B}$

The above logic gates have been shown as having only two inputs. However, gates are available with more than two inputs, but the function remains the same. For example, an AND gate with three inputs will only produce a logic 1 output when all three inputs are at logic 1, as shown below.

P	Q	R	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1



The Boolean expression for this gate is y = P.Q.R

Summary of logic gates

An AND gate will only produce a logic 1 output when all the inputs are at logic 1.

An OR gate will produce an output of logic 1 when any or all the inputs are at logic 1.

A NOT gate produces an output that is the opposite of its input.

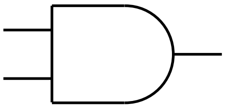
An XOR gate will only produce an output of logic 1 when both inputs are different.

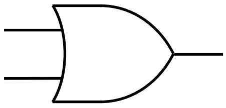
A NAND gate will only produce a logic 0 output when all the inputs are at logic 1.

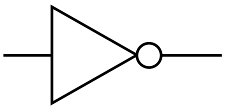
A NOR gate will only produce a logic 1 output when all the inputs are at logic 0.

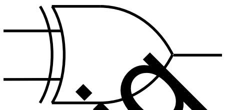
Tutorial Logic Gates

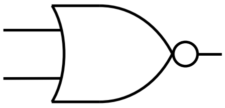
1. Identify the logic gate from the logic symbol.

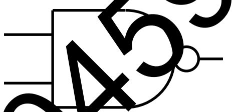


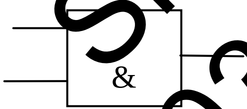


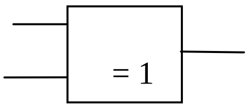


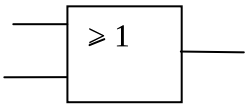


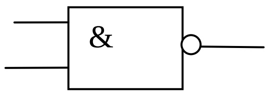












--

2. Draw the logic gate and write the Boolean expression described by the following truth tables.

B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	0

B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	1

B	A	O/P
0	0	0
0	1	0
1	0	0
1	1	1

B	A	O/P
0	0	1
0	1	1
1	0	1
1	1	0

B	A	O/P
0	0	1
0	1	0
1	0	0
1	1	0

A	O/P
0	0
1	1

3. Determine the logic function represented by the following Boolean expressions:

O/P = $\overline{P}$

O/P = A.B.C

O/P = P + Q

O/P = $\overline{A.B}$

O/P = $\overline{A + B}$

O/P = $P \oplus Q$

Converting logic circuits to Boolean expressions and truth tables

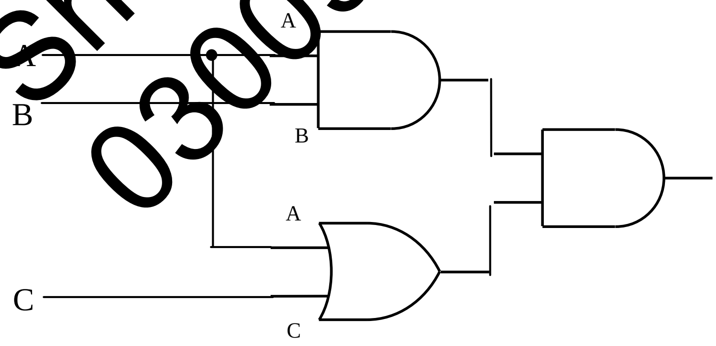
Converting truth tables or Boolean expressions to logic circuits involved first identifying the output gate and the number of inputs to that gate, then working each of these inputs in turn back to the input signals using the Boolean expression for the output of each gate.

This time the logic circuit is known and the Boolean expression must be derived from this logic circuit.

The technique for doing this is the opposite of that used to determine the logic circuit. Instead of working from the output back to the input, the Boolean expression is derived by working from the input to the output. This is done for each logical path in the circuit.

Example 1

Derive the Boolean expression for the following logic circuit.



By looking at the circuit, it can be seen that there are two logical paths from the input signals to the output.

One path is through the top AND gate and the other path is through the bottom OR gate.

Work out for one of these gates, eg the AND gate, what its inputs are and what the output expression is.

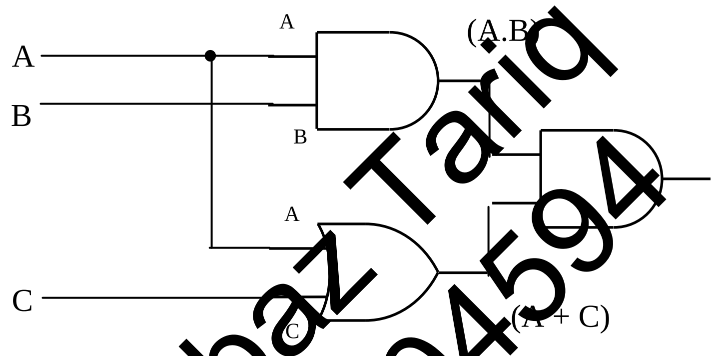
The inputs are A and B, and it is an AND gate so the output is (A.B).

Now repeat this for the OR gate.

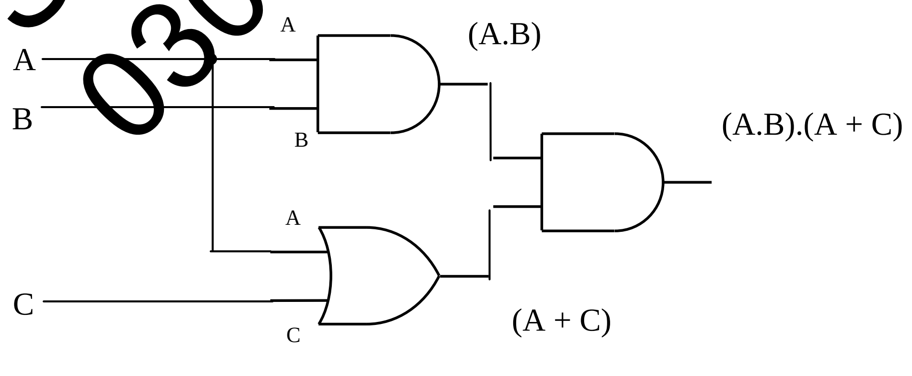
The inputs are A and C, and it is an OR gate so the output is (A + C).

Write the output expression for the two gates on the respective outputs, as shown in the diagram below.

It is sometimes a good idea to enclose the inputs to a logic gate in brackets. This keeps the output expression each gate linked together.



Use these two outputs, which are also the inputs to the last AND gate, to determine the output expression for the AND gate. This is also the Boolean function for the complete circuit.



It is often useful to produce the truth table for the circuit as the function of the circuit can then be determined.

To make things a bit easier, the outputs from all the logic gates should be shown on the truth table, as demonstrated below.

Inputs A, B and C to the circuit.

This is the output of the circuit. The inputs are the outputs of the AND and OR gates given in the previous two columns.

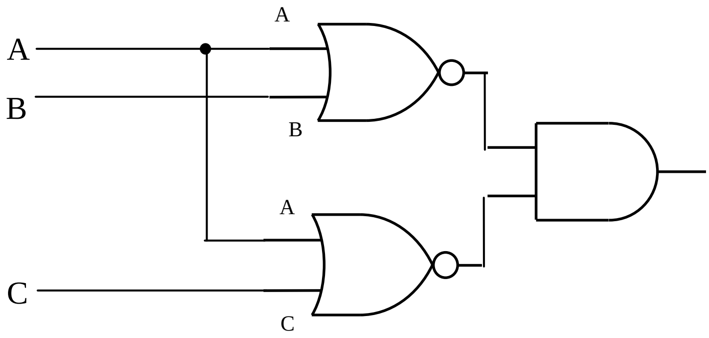
C	B	A	A.B	A + C	O/P = (A.B).(A + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	1	0
1	1	0	0	1	0
1	1	1	1	1	1

This is the output from the AND gate that has A and B as its inputs.
Remember, with an AND gate all the inputs need to be at logic 1 to produce a logic 1 on the output of the gate.

This is the output from the OR gate that has A and C as its inputs.
Remember with an OR gate any input at logic 1 produces a logic 1 on the output of the gate.

Example 2

Derive the truth table and Boolean expression for the following logic circuit:



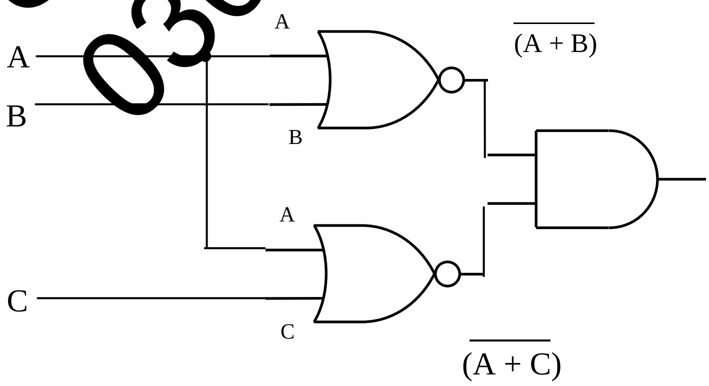
This circuit is very similar to the one in Example 1 but is being used to show how an output with a bar above it is handled.

First identify all the paths from the input to the output and for each gate in the path. Derive its output expression, remembering to enclose the outputs of each gate in brackets.

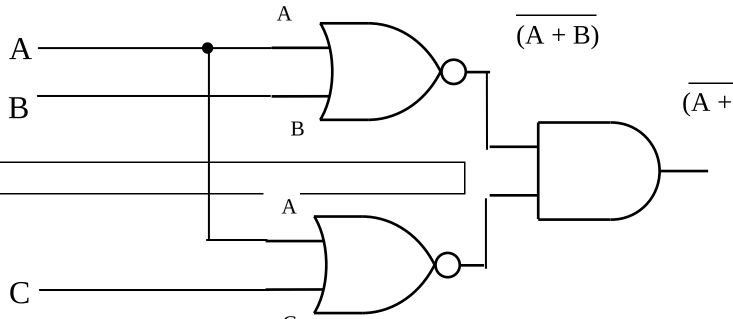
The top gate is a NOR gate, ie an OR gate with its output inverted so the whole of the output will have a bar above it to show this inversion.

The output will be $\overline{(A + B)}$

The output for the bottom NOR gate will be $\overline{(A + C)}$.



The output of the last AND gate will be $\overline{(A + B)}.\overline{(A + C)}$



The truth table for the circuit is:

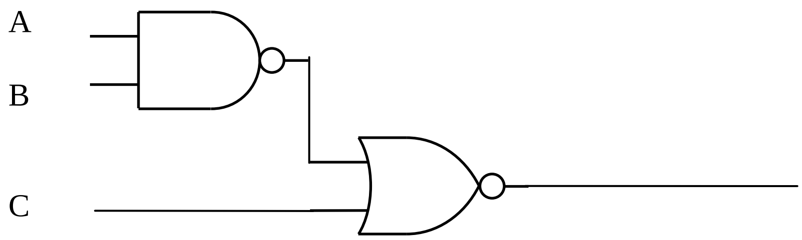
C	B	A	$\overline{A+B}$	$\overline{A + C}$	O/P = $\overline{(A+B)}.\overline{(A + C)}$
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Remember $\overline{A + B}$ and $\overline{A + C}$ are NOR gates so the output will only be logic 1 when both the inputs are logic 0.

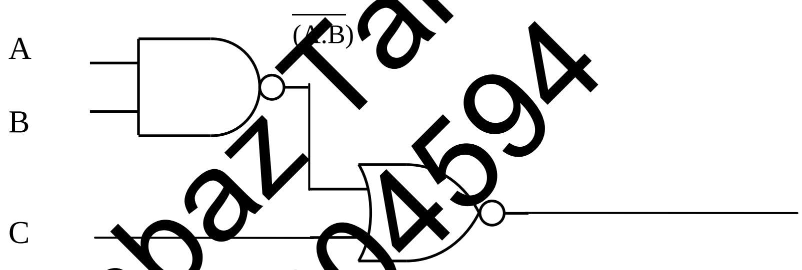
The output is an AND gate so the output will be logic 1 when all inputs are logic 1.

Example 3

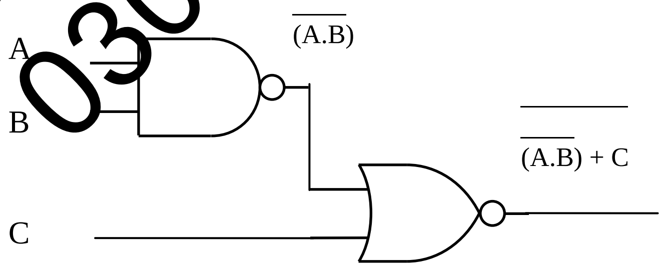
Derive the truth table and Boolean expression for the following logic circuit:



Write the Boolean expression for the output of the NAND gate:



As there is only one gate connected to the output NOR gate, the output diagram can now be written as:



The output expression is therefore $\overline{\overline{(A.B)} + C}$

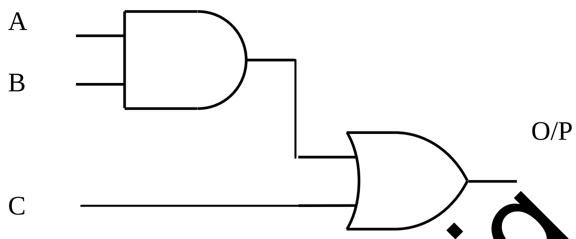
The truth table for this circuit is given below:

C	B	A	$\overline{A.B}$	$\overline{(\overline{A.B}) + C}$
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	0

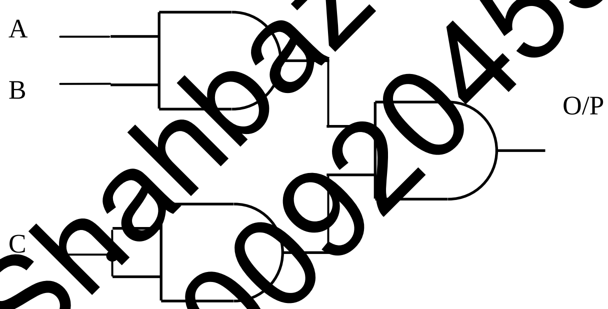
Tutorial: Conversion of logic circuits to Boolean expressions

Derive the truth table and Boolean expression for the output of the following logic circuits.

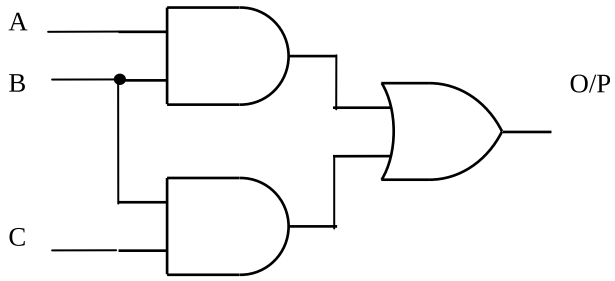
1.



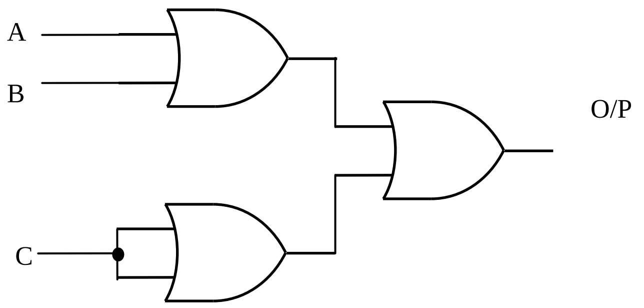
2.



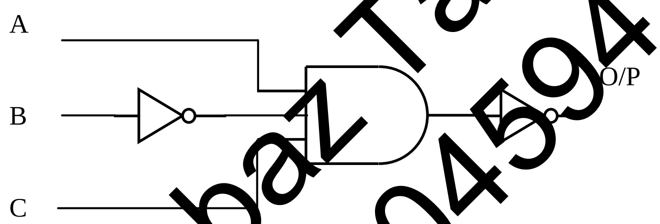
3.



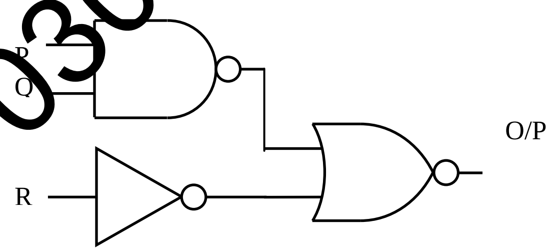
4.



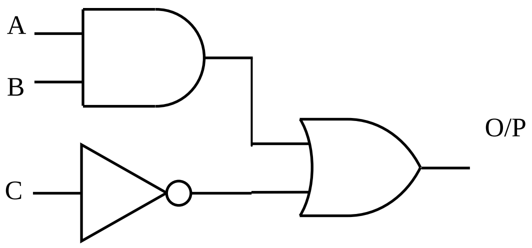
5.



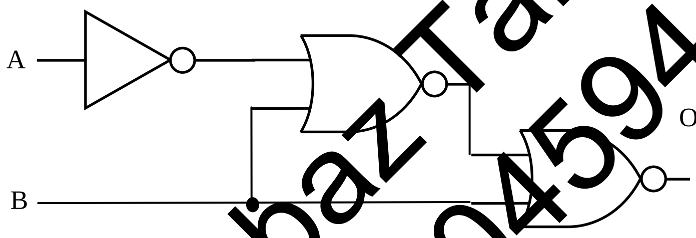
6.



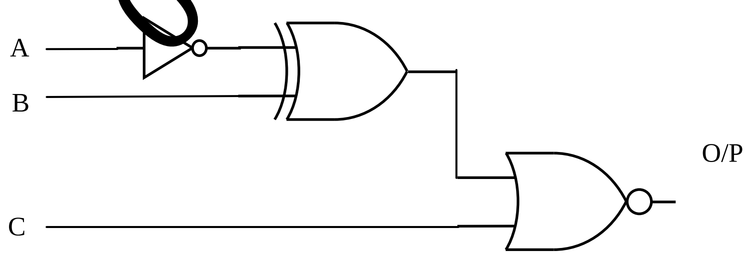
7.



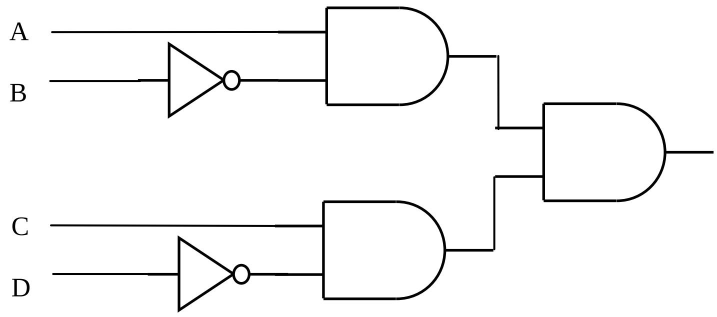
8.



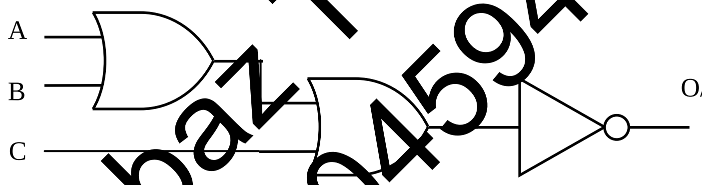
9.



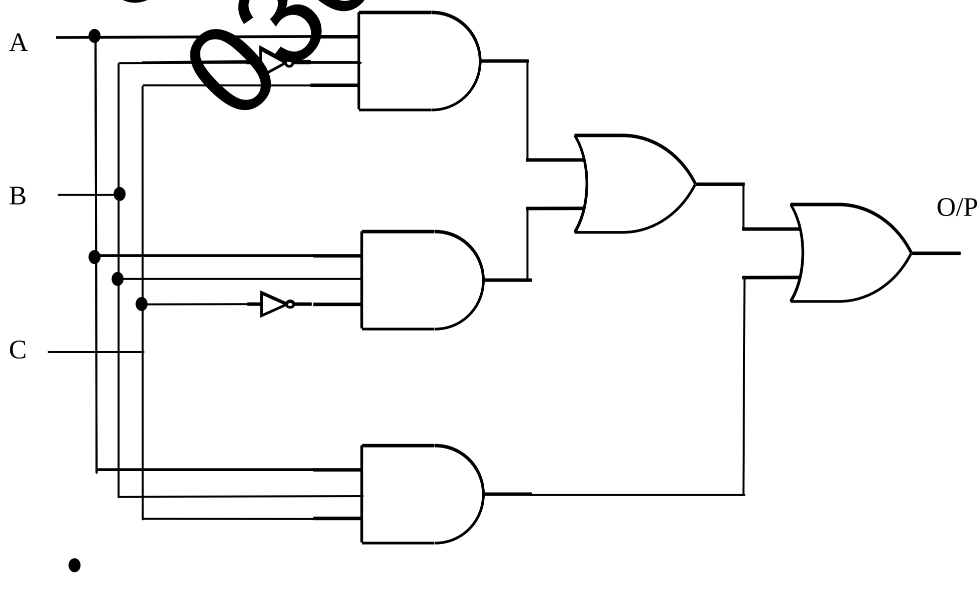
10.



11.



12.



Converting Boolean expressions to logic circuits

Logic circuits can also be designed from Boolean expressions.

Often designers work out how the circuit should perform using Boolean algebra. This will produce a Boolean expression that describes the operation of the logic circuit.

As was seen earlier with truth tables, the output gate is either an OR gate or an AND gate, and the Boolean expression is always in the sum of products form.

With Boolean expressions, however, it's not so simple. The output gate could be any of the basic gates and the output could be in the sum of products (OR or NOR gate) form or it could be in the product of sums form. This form is where each input to the final gate is connected with an AND gate (or NAND) so the output gate is an AND gate (or NAND), eg.

O/P = A + B.A + C

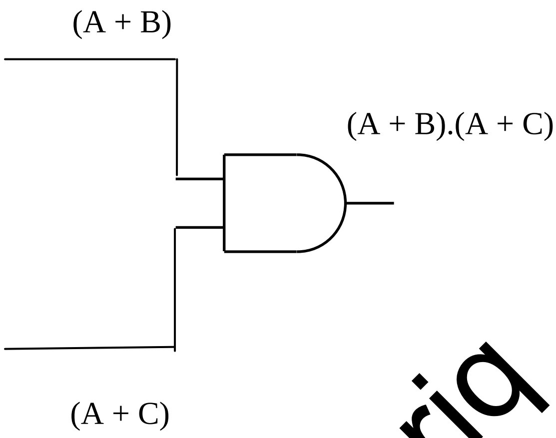
This is not very clear. However, if brackets are used, exactly the same as in ordinary algebra, the expression becomes much clearer:

O/P = (A + B).(A + C)

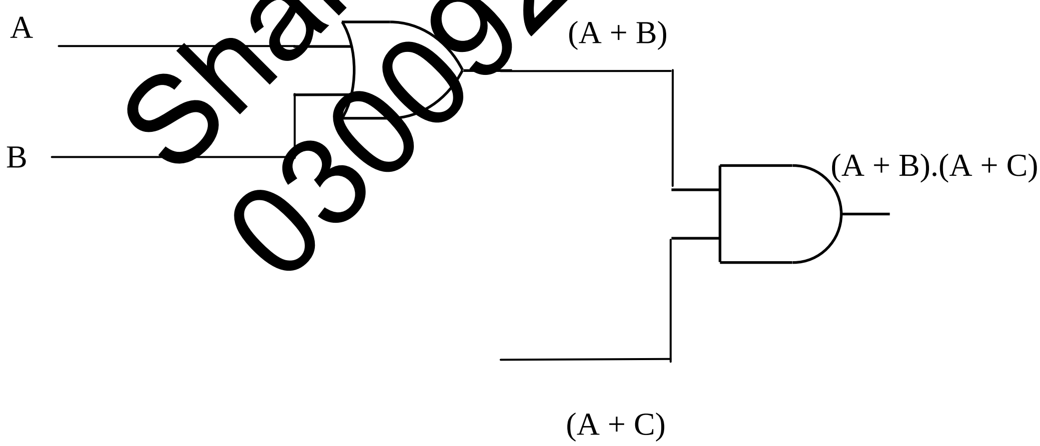
In this case the output gate is an AND gate with two inputs: (A + B) and (A + C).

The circuit diagram is drawn using exactly the same method as for the truth tables, ie draw the output gate then work each input of that gate back to the input of the circuit.

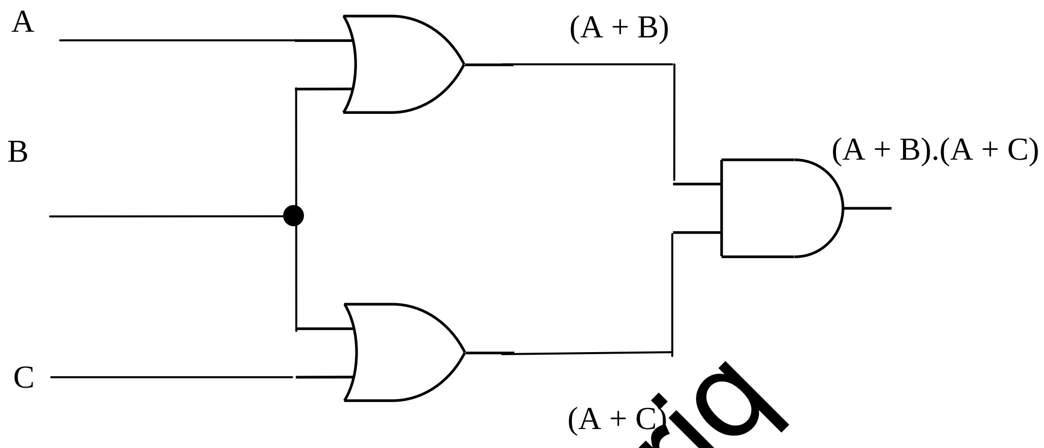
Step 1
Draw the output gate.



Step 2
Take each input in turn and determine the gate and its inputs.
First complete (A + B).



Step 3
Complete (A + C).



Final circuit for (A + B).(A + C)

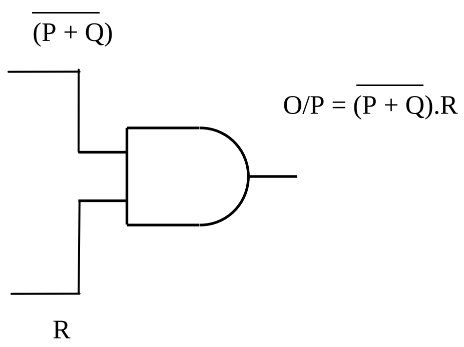
Example 1

Design the logic circuit for the following Boolean expression.

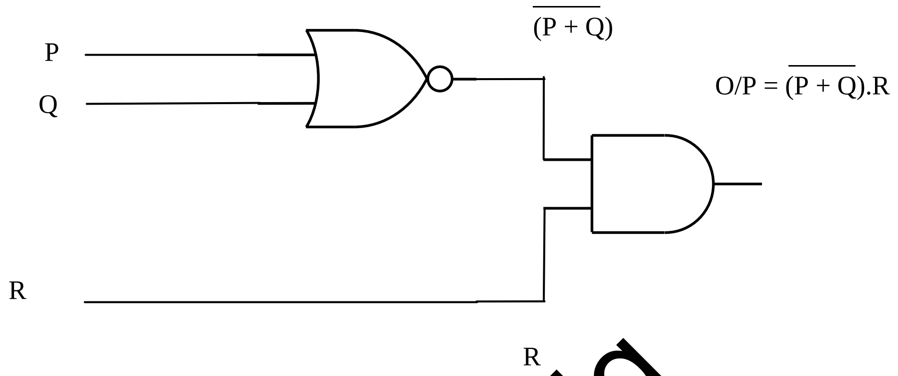
O/P = $\overline{(P + Q)}$.R

The output gate in this example is an AND gate with two inputs: $\overline{(P + Q)}$ and R.

Step 1
Draw the output gate and show its inputs.



Step 2
Take each input in turn and determine the gate and its inputs.



Example 2

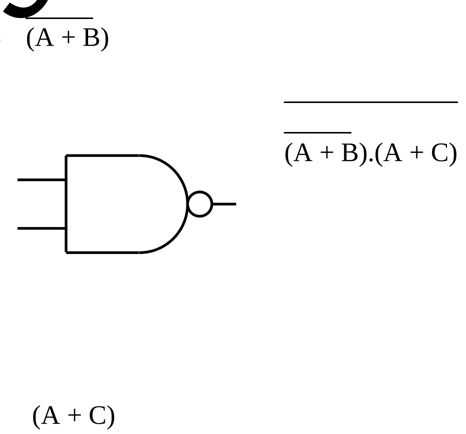
Design the logic circuit for the following Boolean expression:

$$O/P = \overline{\overline{(A + B)}.\ (A + C)}$$

At first glance, this looks like an AND gate output, but a closer inspection shows that there is a bar above the whole expression. This means that the output gate is a NAND gate with two inputs:

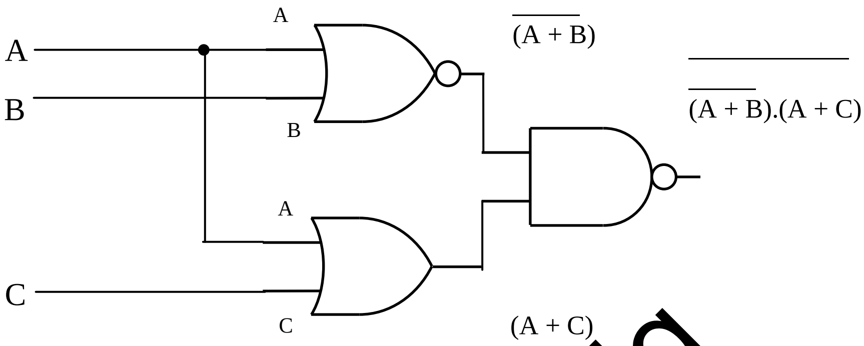
$\overline{(A + B)}$, ie a NOR gate, and $(A + C)$, ie an OR gate.

Step 1
Draw the output gate showing its inputs.



Step 2

Take each input in turn and determine the gate and its inputs.



Tutorial: Logic circuits from Boolean expressions

Draw the logic circuit for each of the Boolean expressions given below.

1. $A.B.C$

2. $\overline{A} + \overline{B}$

3. $A.B + C$

4. $\overline{A}.\overline{B}.\overline{C}$

5. $\overline{A}.B + C$

6. $\overline{P + Q}$

BOOLEAN EXPRESSIONS TO LOGIC CIRCUITS

7. $\overline{\overline{A + B} + C}$

8. $\overline{\overline{A.B} + C}$

9. $\overline{(P + Q).(P + R)}$

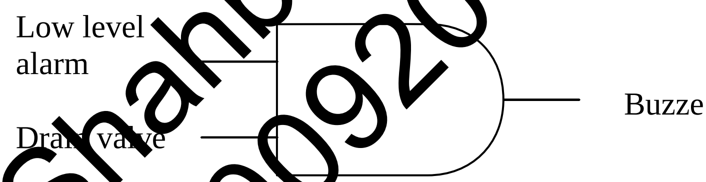
10. (A.B) O ⊕ C

Converting truth tables to logic circuits

Logic gates can be used either individually or combined with other logic gates to produce circuits that provide some function.

For example, a circuit is required that will sound an alarm if the water in a tank drops to a level that activates the low water level alarm and the drain valve is open. This is a simple AND function with one input from the low water alarm and the other from the drain valve.

The low water level alarm signal is at logic 1 when the alarm is activated. The drain valve signal will be at logic 1 when the drain valve is open.



The truth table is:

Low water level	Drain valve	Buzzer
0	0	0
0	1	0
1	0	0
1	1	1

The Boolean expression for this circuit is:

alarm = low water level • drain valve

ie the alarm will sound when the water trips the low level alarm *and* the drain valve is open.

Example 1

A machine for cutting metal will only activate the cutter when the safety guard is in position, the operator holds the safety handle and the footswitch is activated.

This is another AND function, with the safety guard as one input, the safety handle as another input and the footswitch as the third input.

The truth table is:

Guard	Safety handle	Footswitch	Cutter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

The cutter will only operate when the guard is in place, the safety handle is held and the footswitch is activated.

The Boolean expression is cutter = guard.safety handle.footswitch.

Designers of logic circuits are sometimes presented with either a truth table or a Boolean expression of what is required. From this the designer must develop the logic circuit.

Logic design from a truth table

Circuit designers often use truth tables as the specification for a logic circuit.

A truth table is a table that shows all combinations of inputs and the corresponding output for each input combination. The only outputs that need to be considered are the ones that have logic 1 as the output. Logic 0 outputs do not get used.

If an input (eg input A) is at logic 1 then that input is simply called A (or P, or Q etc).

However, if the input to a logic gate is $\overline{A}$ (or $\overline{P}$, or $\overline{R}$ etc.) then this is the opposite of the input signal and is produced by using an inverter gate in the circuit to change A to $\overline{A}$.

Example 2

Consider the following truth table.

B	A	O/P
0	0	0
0	1	1
1	0	0
1	1	0

There is only one condition where the output is at logic 1: when input A is at logic 1 and input B is at logic 0.

It is important to note that the inputs to the circuit are A and B, and the output is O/P.

Input A is A but input B, because it is at logic 0, is $\overline{B}$. ie B not $\overline{B}$.A.

B	A	O/P
0	0	0
0	1	1
1	0	0
1	1	0

In algebra 2 × 3 is the same as 3 × 2 and this is also true in Boolean algebra:

$\overline{B}$.A is exactly the same as A.$\overline{B}$

As there is only one output in the truth table at logic 1, the Boolean expression for the truth table is:

O/P = $\overline{B}$.A

This could also have been written as O/P = $\overline{A}$.B.

All the inputs for a particular output are connected together using an AND function, as shown above.

TRUTH TABLES TO LOGIC CIRCUITS

All the circuits we will look at in this course will have two or more inputs, but only one output. This means there will be a single logic gate producing the output.

This logic gate brings all the parts of either the truth table or the Boolean expression together to a single output.

If we are producing a logic circuit from a truth table, then the output gate will always be an OR gate (or NOR) if there is more than one line in the truth table at logic 1.

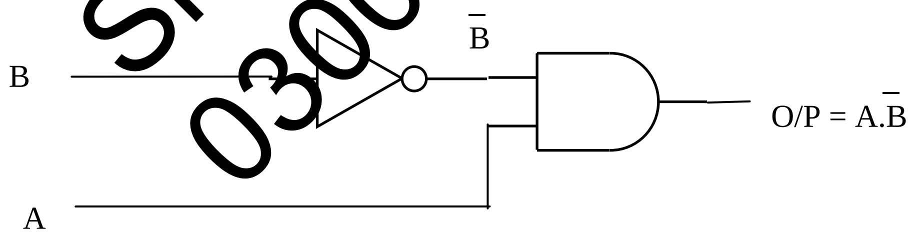
If there is only one line on the truth table at logic 1 then the output gate will be an AND gate (or NAND).

Consider the example above. The output is given by $A.\overline{B}$. This is called the product of sums form, ie all inputs are connected by an AND gate.

By looking at the expression we can see the there are two inputs, A and B, and the gate is an AND gate, signified by the ‘ **.** ’ operator.

The circuit will consist of an AND gate with two inputs, A and $\overline{B}$.

$\overline{B}$ is simply input B passed through an inverter gate as shown below:



This is the logic circuit described by the truth table.

Example 2

Draw the logic diagram for the truth table given below.

R	Q	P	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Step 1
Write the Boolean expression for all lines where the output is logic 1.

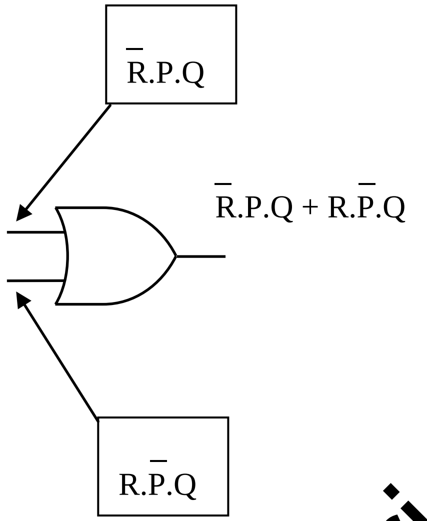
In this case there are two lines where the output is logic 1.

R	Q	P	O/P	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{R}.Q.P$
1	0	0	0	
1	0	1	1	$R.\overline{Q}.P$
1	1	0	0	
1	1	1	0	

$O/P = \overline{R}.Q.P + R.\overline{Q}.P$

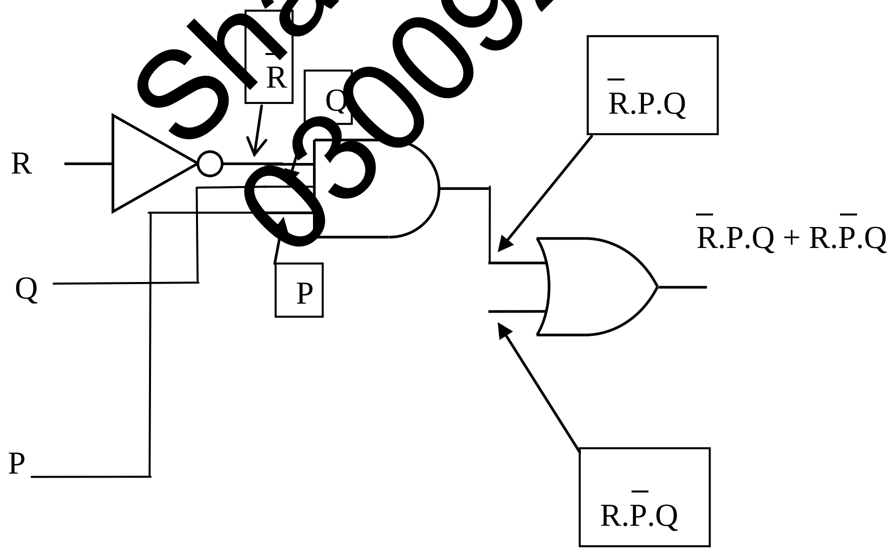
This expression is called the sum of products form, in which the inputs for each line are connected by an AND function (product of sums) and the expression for each line is connected by an OR function.

The output gate is an OR gate with one input $\overline{R}.P.Q$ and the other input $R.\overline{P}.Q$.

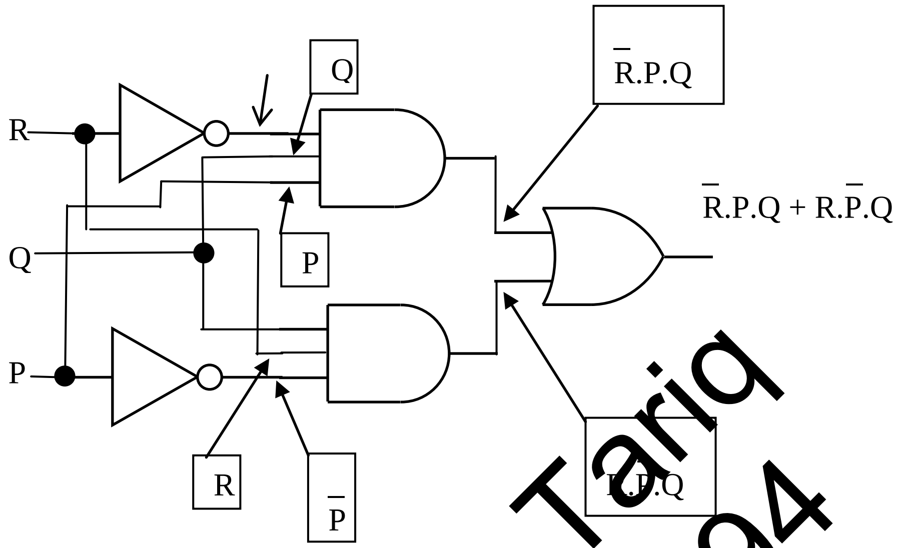


Now take each input in turn and determine the gate and its inputs.

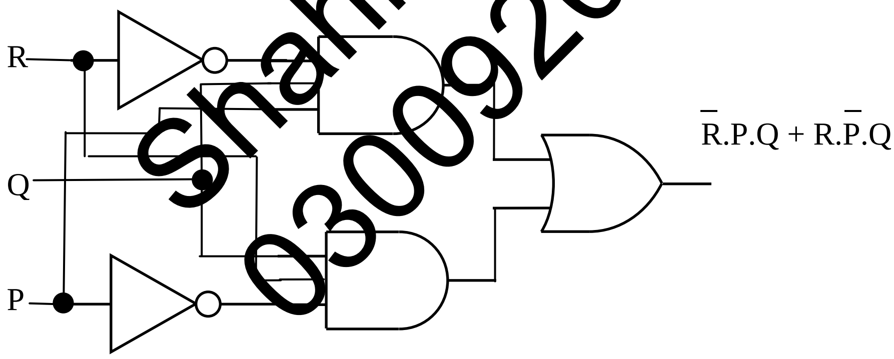
The input $\overline{R}$.P.Q is another AND gate with inputs $\overline{R}$, P and Q as shown below. Input R is put through an inverter gate to produce the required input to the AND gate.



Now complete the circuit for $\overline{R}.P.Q$.



The circuit is shown below without any labels for clarity.



Tutorial: Logic circuits from truth tables

Draw the logic circuit described by the truth tables below.

1.

B	A	O/P
0	0	0
0	1	0
1	0	0
1	1	1

2.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3.

B	A	O/P
0	0	0
0	1	1
1	0	1
1	1	0

4.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

5.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

6.

C	B	A	O/P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

7.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

8.

C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

9.

C	B	A	O/P
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

10.

R	Q	P	O/P
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

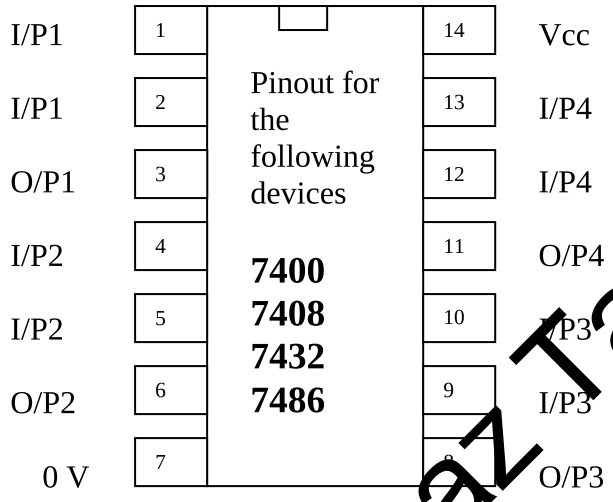
11.

Z	Y	X	O/P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

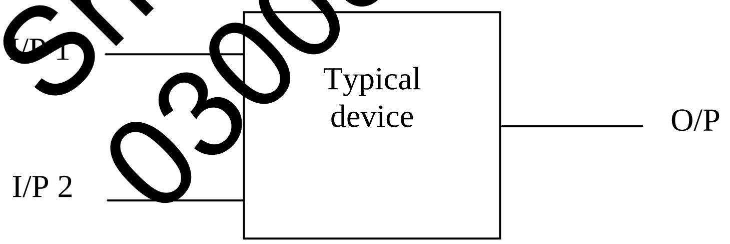
12.

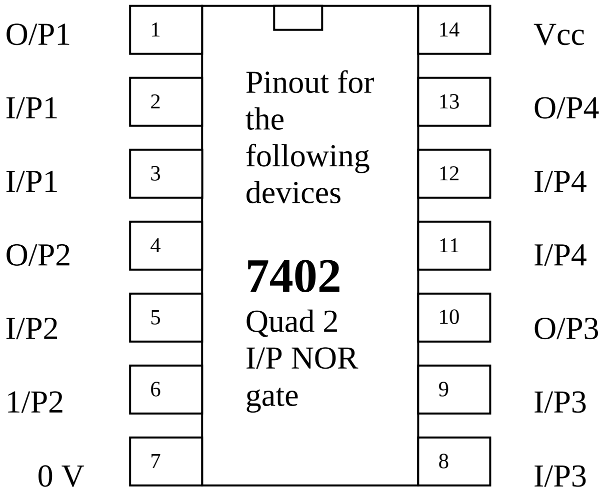
C	B	A	O/P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

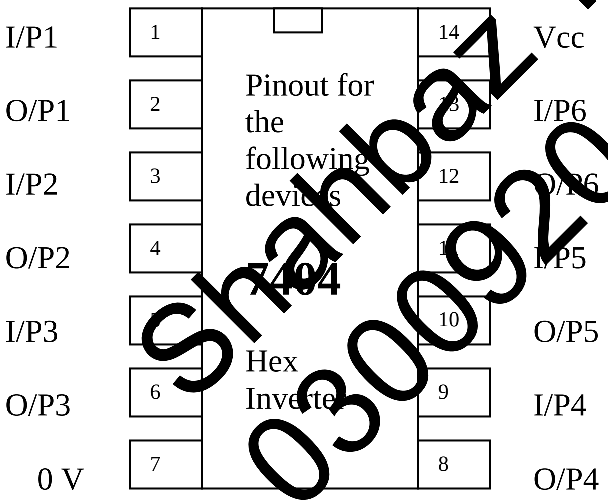
Appendix 1



7400 – quad 2 I/P NAND gate
7408 – quad 2 I/P AND gate
7432 – quad 2 I/P OR gate
7486 – quad 2 I/P XOR gate




O/P1
I/P1
I/P1
O/P2
I/P2
1/P2
0 V
1
2
3
4
5
6
7
Pinout for the following devices
7402
Quad 2 I/P NOR gate
14
13
12
11
10
9
8
Vcc
O/P4
I/P4
I/P4
O/P3
I/P3
I/P3


I/P1
O/P1
I/P2
O/P2
I/P3
O/P3
0 V
1
2
3
4
6
7
Pinout for the following devices
7404
Hex Inverter
14
12
10
9
8
Vcc
I/P6
O/P6
I/P5
O/P5
I/P4
O/P4