



Boolean Logic

Student Booster Event

Starter: Writing Boolean expressions

Ava wants to go to a party. Her Dad says, "you can go to the party if you don't get any detentions this week and don't make a mess of your room".

Write a Boolean expression for this situation.

party = _____



Answer: Writing Boolean expressions

Ava wants to go to a party. Her Dad says, "you can go to the party if you don't get any detentions this week and don't make a mess of your room".

Write a Boolean expression for this situation.

party = NOT (detentions) AND NOT (mess)

which is the same as...

party = NOT (detentions OR mess)

party = detentions NOR mess



Learning objectives

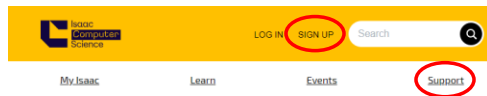
By the end of this 90-minute session, you will:

- Understand why computers use Boolean logic
- Recap all six logic gate symbols, their truth tables, and operator precedence
- Draw a circuit from a truth table or problem statement
- Write an expression from a circuit and vice versa
- Understand the binary half-adder circuit
- Recap Boolean algebra and simplify an expression
- Recap Karnaugh maps and simplify an expression



Getting Started on Isaac

To make the most of this session, you need an account on the Isaac website. Go to [IsaacComputerScience.org](https://isaac-computer-science.org) and click **Sign Up**. Follow the instructions. For help click *Support*.



If you can't do this today, that's fine, just use the handouts provided and sign up later.



Setting your preferences

Click **My Account** and fill in these fields:

I am studying !

A Level

▼

OCR

▼

☐ Show other content that is not for my selected exam board. !

Boolean logic notation

And (∧) Or (∨) Not (¬)

▼



Algebraic notation

A quick note on notation...

All these expressions mean the same thing →

OCR uses the symbols:

$\wedge \vee \neg$

AQA uses the symbols:

$\cdot + -$

$$Q = A \text{ AND } B$$

$$Q = A \wedge B \quad (\text{OCR})$$

$$Q = A \cdot B \quad (\text{AQA})$$

Check with your teacher you are using the correct notation.



"My theory of Logic is the thing by which I would desire to be remembered hereafter"



George Boole,
1815-1864



Boolean logic



Boole's "leap of logic" was to realise:

1. All statements of **fact** can be represented by **mathematical expressions**
2. These expressions can be **combined and manipulated** to deduce new facts

The rules became known as **Boolean Logic** or Boolean Algebra.

An example of a Boolean expression is:

NOT (detentions) AND NOT(mess)



Why do computers use Boolean?

Boolean logic can be performed with binary numbers. We can say that "1" means "true" and "0" means "false".

Then we can store **true** and **false** in the computer as the **bits 1 and 0**, then manipulate them just like other numbers.

Mum: "Did you tidy your room?"

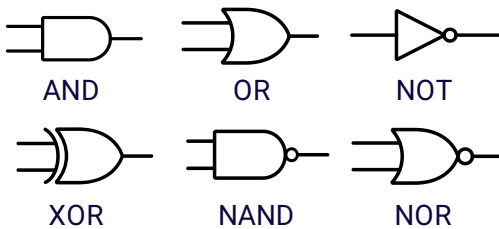
Me: "1"



Logic gates



Logic gates



More information at
isaaccomputerscience.org/concepts/sys_bool_logic_gates



AND gate



The output **Q** is 1 if both inputs **A** and **B** are 1.

We can show all the possible inputs and the expected output for each in a **truth table** →

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



AND gate



We can also write a **Boolean expression** for this circuit, it is:

$$Q = A \text{ AND } B$$

or

$$Q = A \wedge B \quad (\text{OCR})$$

$$Q = A \cdot B \quad (\text{AQA})$$

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



OR gate



The output **Q** is 1 if either input **A** or **B** is 1. Here are the truth table and expressions:

$$Q = A \text{ OR } B$$

$$Q = A \vee B \quad (\text{OCR})$$

$$Q = A + B \quad (\text{AQA})$$

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



NOT gate

Takes only one input and changes it to the opposite value (**inverts** it).

We write this as

$$Q = \text{NOT } A$$

$$Q = \neg A \quad (\text{OCR})$$

$$Q = \bar{A} \quad (\text{AQ A})$$



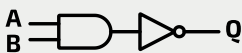
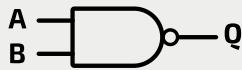
A	Q
0	1
1	0



NAND gate

The **NAND** gate combines the logic of an **AND** gate and a **NOT** gate.

It produces the **inverse** of the output of an **AND** gate...



NAND gate

This results in an output 1 if it's **NOT true that A and B are 1**

We write this as

$$Q = A \text{ NAND } B$$

$$Q = \neg (A \wedge B)$$

$$Q = \overline{A \cdot B}$$



A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



XOR gate



Outputs 1 if
**A or B but
not both** are 1.

Exclusive OR,
sometimes
written EOR.

$$Q = A \text{ XOR } B$$

$$Q = A \vee B$$

$$Q = A \oplus B$$

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

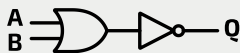


NOR gate

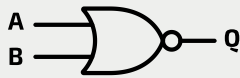


The **NOR** gate
combines the logic
of an **OR** gate and
a **NOT** gate.

It produces the
inverse of the
output of an **OR**
gate...



NOR gate



Outputs 1 if both
inputs are zero, i.e.
neither A NOR B
are 1.

$$Q = A \text{ NOR } B$$

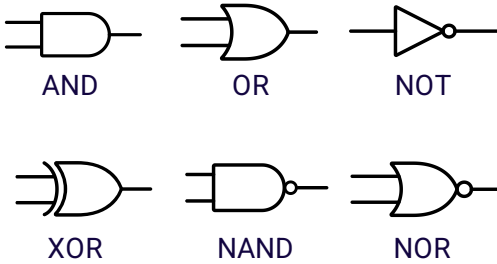
$$Q = \neg(A \vee B)$$

$$Q = A + B$$

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



Logic gates



Operator Precedence

If you studied GCSE CS you learned:

1. Brackets
2. NOT
3. AND
4. OR

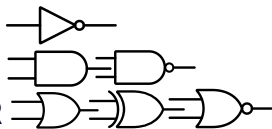
*Boole
Never
Ate
Olives*



Operator Precedence

At A-level we add the new gates:

1. Brackets
2. NOT
3. AND, NAND
4. OR, XOR, NOR



Activity 1: identify the gates

Open the Logic Gates topic here:

bit.ly/ialogicgates

Answer the questions at the bottom...



Related questions

[It's a gate \(A Level - P1\)](#)

[Truth table \(A Level - P1\)](#)

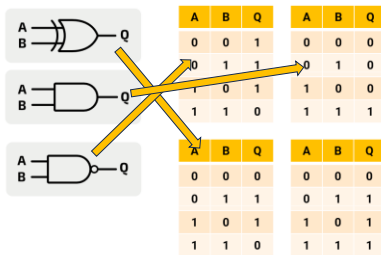
[Tables and gates \(A Level - P1\)](#)

[Truth tables \(A Level - P1\)](#)



Activity 1: solution

Match the gates to their truth tables:



Writing truth tables

When constructing a truth table, always ensure the inputs are in binary numerical order...

A	B	Q
0	0	
0	1	
1	0	
1	1	

A	B	C	Q
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Writing truth tables

Start with the rightmost input and fill with 0, 1, 0, 1, then pairs 0, 0, 1, 1, then fours and so on...

A	B	Q

A	B	C	Q

"To most people, the word
"information" suggests meaning.
To the communication engineer, it is
the problem of getting zeros and ones
from one point to another"



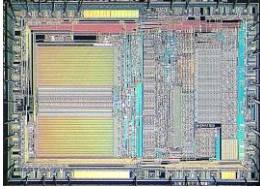
Claude Shannon,
1916 - 2001

Logic circuits

Logic circuits

Computer circuits are made up of millions of logic gates.

To illustrate the principles, we will consider simple circuits made up of just a few logic gates.



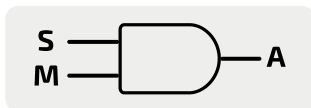
Expression from a circuit: intruder alarm



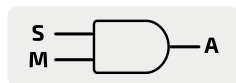
An intruder alarm circuit has two inputs S and M. The alarm will sound if the system is armed ($S = 1$) and motion is detected ($M = 1$).

There is one output, an alarm A, which will sound if output is 1.

What gate(s) do we need for this circuit?



Expression from a circuit

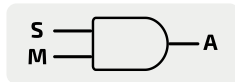


What should the truth table look like for the intruder alarm?

S	M	A
0	0	
0	1	
1	0	
1	1	



Expression from a circuit



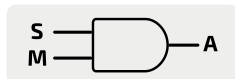
What should the truth table look like for the intruder alarm?

The same as an AND gate.

S	M	A
0	0	0
0	1	0
1	0	0
1	1	1



Expression from a circuit



What is the Boolean expression?

$A = S \text{ AND } M$

$A = S \wedge M$ (OCR)

$A = S \cdot M$ (AQA)

S	M	A
0	0	0
0	1	0
1	0	0
1	1	1



Circuit from a scenario: fire and intruder alarm



The system has been upgraded to detect motion only at night, and a new **temperature sensor** has been added to detect a fire.

The alarm will now sound:

- if motion is detected **and** it is between 20:00 and 06:00 hours, **or...**
- ...if the temperature exceeds 30 degrees C, whatever time it is.

Question: what gates are needed in this circuit?
Type in the chat all the gates needed.



Activity 2: circuit from a scenario



The alarm will sound:
if motion is detected **and** it is between 20:00 and 06:00 hours, **or**...
...if the temperature exceeds 30 degrees C, whatever time it is.

Inputs are labelled T, M and C

- T will be set to 1 if temperature exceeds 30C, else 0
- M will be 1 if motion is detected, else 0
- C will be 1 if time between 20:00 and 06:00, else 0

There is one output A:

- Alarm A will sound if it receives a 1.

Question: draw the logic circuit for this scenario.

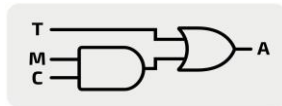


Circuit from a scenario

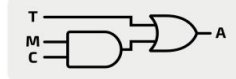


Solution: plug the variables into the question, and identify the operations (and/or/not):

- if motion is detected (**M**) **and** it is between 20:00 and 06:00 hours (**C**), **or**...
- ...if the temperature exceeds 30 degrees C, whatever time it is (**T**)



Expression from a circuit



Question: what is the Boolean expression for this circuit? Type in the chat.

$$A = (M \text{ AND } C) \text{ OR } T$$

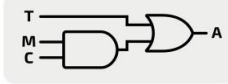
$$A = M \text{ AND } C \text{ OR } T$$

$$A = M \wedge C \vee T \quad (\text{OCR})$$

$$A = M \cdot C + T \quad (\text{AQA})$$



Truth table from a circuit

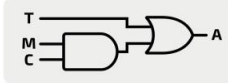


To construct the Truth Table for this circuit, first we calculate M AND C, then OR this with T.

T	M	C	M AND C	A
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	



Truth table from a circuit



To construct the Truth Table for this circuit, first we calculate M AND C, then OR this with T.

T	M	C	M AND C	A
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	1	1

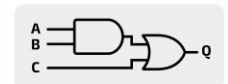


Circuit from an expression

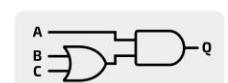
Question: Which of these logic circuits correctly implements this Boolean expression?

$$Q = A \wedge B \vee C \quad (\text{OCR})$$

$$Q = A \cdot B + C \quad (\text{AQA})$$



Option A

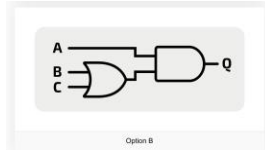


Option B

Expression from a circuit

This is circuit B.

Question: what is its Boolean expression?

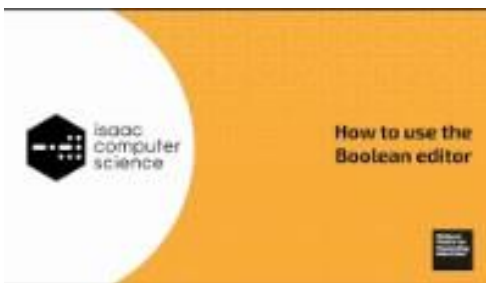


$$Q = A \wedge (B \vee C) \quad (\text{OCR})$$

$$Q = A \cdot (B + C) \quad (\text{QAQA})$$



Interlude: The Isaac Boolean Editor



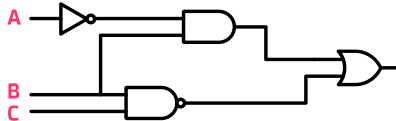
[youtube/4EckK8HNwRM](https://www.youtube.com/watch?v=4EckK8HNwRM)



Activity 3: Expression from a circuit



Attempt the Isaac question here: bit.ly/ialogic4



Write the Boolean expression for this circuit.

$$\neg A \wedge B \vee \neg (B \wedge C) \quad (\text{OCR})$$

$$\bar{A} \cdot B + (\bar{B} \cdot \bar{C}) \quad (\text{QAQA})$$



There are 10 types of people in the world, those who understand binary, and those who don't.

Professor Ian Stewart,
b.1945



Useful circuits



Useful circuits

We have seen already how logic circuits can implement logical decisions (such as "if" and "while" statements in high-level programs).

The other function the ALU performs is arithmetic. Let's look at **binary arithmetic** now and consider how we can perform it with logic gates...

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$



Binary half adder

A **binary half adder** circuit adds two single-bit binary numbers together using these rules →

Note the last row results in a **carry bit**.

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10



Binary half adder

We can draw a truth table for these rules.

Let's call the two input bits A and B and the outputs C (carry) and S (sum).

0	+	0	=	0
0	+	1	=	1
1	+	0	=	1
1	+	1	=	10

INPUTS		OUTPUTS	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Binary half adder

What logic circuit would result in this truth table?

First consider output C, the carry bit, what gate is this?



AND gate

INPUTS		OUTPUTS	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Binary half adder

Now consider output S, the sum bit.
What gate would produce this output?

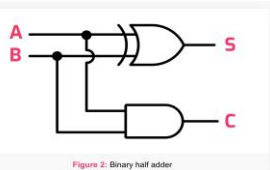


INPUTS		OUTPUTS	
A	B	C	S
0	0		0
0	1		1
1	0		1
1	1		0

Binary half adder

The complete half-adder circuit is therefore:

INPUTS		OUTPUTS	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

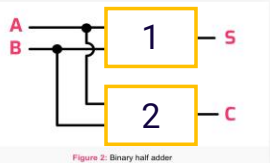


Activity 4: Half adder

Complete the half-adder circuit diagram on the handout (or on paper).

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

INPUTS		OUTPUTS	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

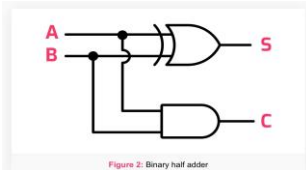


Solution: Half adder

This is the complete half adder circuit.

$$\begin{array}{r} 1 \\ + 1 \\ \hline 10 \end{array}$$

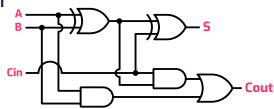
INPUTS		OUTPUTS	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Useful circuits

See isaaccomputerscience.org/concepts/sys_bool_useful_circuits later for more useful circuits including:

- Binary equivalence tester
- Binary full adder (below)
- Four-bit binary adder
- Sequential circuits
- Flip-flops



The moving power of
mathematical invention is not
reasoning but imagination.

Augustus de Morgan,
1806 - 1871

Boolean algebra – a sample

We will look at some of the rules of Boolean Algebra.
All the rules can be found at bit.ly/iabooleanalgebra

Note: The next few slides use OCR-style notation.
Skip forward for AQA.

Associative
Absorption
Identity
Commutative
DeMorgan



Boolean Algebra – OCR



Identities – OCR

These are basic building blocks of Boolean algebra.

They are mostly obvious, such as $A \vee \neg A = 1$

In this example, "raining or not raining" must be true, because it must be one or the other!

$A \vee \neg A$	$= 1$
$A \wedge \neg A$	$= 0$
$\neg \neg A$	$= A$
$A \wedge 0$	$= 0$
$A \wedge 1$	$= A$
$A \vee 1$	$= 1$



Identities – OCR

Question: simplify:

$$(A \wedge B) \wedge \neg (A \wedge B)$$

Answer: using

$$A \wedge \neg A = 0$$

this simplifies to:

0

$$A \vee \neg A = 1$$

$$A \wedge \neg A = 0$$

$$\neg \neg A = A$$

$$A \wedge 0 = 0$$

$$A \wedge 1 = A$$

$$A \vee 1 = 1$$



Absorption law – OCR

$$X \vee (X \wedge Y) = X$$

$$X \wedge (X \vee Y) = X$$

If you "AND/OR" a variable with a term that is dependent on that variable, the variable **absorbs** that term. Consider an example:

raining OR (raining AND windy)
= raining



Commutative and Associative laws – OCR

$$X \wedge Y = Y \wedge X$$

$$X \vee Y = Y \vee X$$

$$X \wedge (Y \wedge Z) = (X \wedge Y) \wedge Z$$

$$X \vee (Y \vee Z) = (X \vee Y) \vee Z$$

Variables and brackets can be rearranged as long as the operators are the same.

saturday OR (sunday OR bank-holiday)
= sunday OR saturday OR bank-holiday



Distributive law – OCR

$$\begin{aligned} X \wedge (Y \vee Z) &= (X \wedge Y) \vee (X \wedge Z) \\ X \vee (Y \wedge Z) &= (X \vee Y) \wedge (X \vee Z) \end{aligned}$$

This operation is equivalent to expanding the brackets or factorisation in normal algebra.

Question: use the distributive law to simplify
raining AND (windy OR cold)

$$= (\text{raining AND windy}) \text{ OR } (\text{raining AND cold})$$



De Morgan's laws – OCR

$$\begin{aligned} \neg (X \wedge Y) &= \neg X \vee \neg Y \\ \neg (X \vee Y) &= \neg X \wedge \neg Y \end{aligned}$$

For each of De Morgan's laws, remember "drive in negation and flip the operator".

Question: use De Morgan's law to simplify
NOT (raining AND windy)

$$= \text{NOT raining OR NOT windy}$$



Activity 5: simplifying a Boolean expression – OCR



Simplify $(A \wedge \neg B) \vee B$

$$= B \vee (A \wedge \neg B)$$

< commutative

$$= (B \vee A) \wedge (B \vee \neg B)$$

< distributive

$$= (B \vee A) \wedge 1$$

< identity $A \vee \neg A = 1$

$$= (B \vee A)$$

< identity $A \wedge 1 = A$

For help and more worked examples
see Isaac at bit.ly/iasimplify

Boolean Algebra – AQA

Identities – AQA

These are basic building blocks of Boolean algebra.

They are mostly obvious, such as $A + \bar{A} = 1$

In this example, "raining or not raining" must be true, because it must be one or the other!

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

$$\bar{\bar{A}} = A$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A + 1 = 1$$

Identities – AQA

These can be used to simplify Boolean expressions.

Question: simplify: _____

$$(A \cdot B) \cdot (\bar{A} \cdot B)$$

Answer: using

$$A \cdot \bar{A} = 0$$

this simplifies to:

$$A + \bar{A} = 1$$

$$A \cdot \bar{A} = 0$$

$$\bar{\bar{A}} = A$$

$$A \cdot 0 = 0$$

$$A \cdot 1 = A$$

$$A + 1 = 1$$

Absorption law – AQA

$$X + (X \cdot Y) = X$$

$$X \cdot (X + Y) = X$$

If you "AND/OR" a variable with a term that is dependent on that variable, the variable **absorbs** that term. Consider an example:

raining OR (**raining** AND **windy**)
= **raining**



Commutative and Associative laws – AQA

$$X \cdot Y = Y \cdot X$$

$$X + Y = Y + X$$

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

$$X + (Y + Z) = (X + Y) + Z$$

Variables and brackets can be rearranged as long as the operators are the same.

saturday OR (**sunday** OR **bank-holiday**)
= **sunday** OR **saturday** OR **bank-holiday**



Distributive law – AQA

$$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

$$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$

This operation is equivalent to expanding the brackets or factorisation in normal algebra.

Question: use the distributive law to simplify

raining AND (**windy** OR **cold**)
= (**raining** AND **windy**) OR
(**raining** AND **cold**)



De Morgan's laws – AQA

$$\overline{X \cdot Y} = \overline{X} + \overline{Y}$$

$$\overline{X + Y} = \overline{X} \cdot \overline{Y}$$

For each of De Morgan's laws, remember "drive in negation and flip the operator".

Question: use De Morgan's law to simplify

NOT (raining AND windy)

= NOT raining OR NOT windy



Activity 5: simplifying a Boolean expression – AQA



Simplify $(A \cdot \overline{B}) + B$

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

< commutative

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

< distributive

$$= (B + A) \cdot 1$$

< identity $A + \overline{A} = 1$

$$= (B + A)$$

< identity $A \cdot 1 = A$

For help and more worked examples see Isaac at bit.ly/iasimplify



Karnaugh maps



Karnaugh maps overview

A Karnaugh map is just a truth table rearranged to enable us to see patterns. Let's look at the truth table and Karnaugh map for...

$$A \vee (A \wedge B) \quad (\text{OCR})$$

$$A + (A \cdot B) \quad (\text{AQA})$$

A	B	$A \vee (A \wedge B)$
0	0	0
0	1	0
1	0	1
1	1	1

A \ B	0	1
0	0	0
1	0	1



Karnaugh maps overview

Now we have our Karnaugh map, we can look for a group of 1s together, making up a power of two (2, 4 or 8 "1"s), in a contiguous rectangle.

$$A \vee (A \wedge B) \quad (\text{OCR})$$

$$A + (A \cdot B) \quad (\text{AQA})$$

A	B	$A \vee (A \wedge B)$
0	0	0
0	1	0
1	0	1
1	1	1

A \ B	0	1
0	0	0
1	0	1

A = 1



Activity 6: Simplifying Karnaugh maps



Attempt the question here:

bit.ly/iaboolq9

Answer:

In this group, A = 1 (A is true)

In this group, B = 1 (B is true)

So output is 1 when A is 1 or B is 1:

$$A \vee B \quad (\text{OCR})$$

$$A + B \quad (\text{AQA})$$

A \ B	0	1
0	0	0
1	1	1

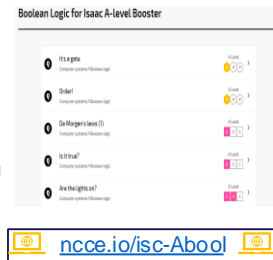


You have 3 minutes!



Isaac Gameboard practice

- If you want more Boolean logic practice, then try this gameboard.
- You will need to sign in to **Isaac Computer Science** or register for a free account if not done already.



Learning objectives – recap

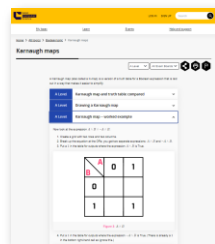
You should now be able to:

- Understand why computers use Boolean logic
- Recap all six gate symbols, their truth tables, and operator precedence
- Draw a circuit from a truth table or problem statement
- Write an expression from a circuit and vice versa
- Understand the binary half-adder circuit
- Recap Boolean algebra and simplify an expression
- Recap Karnaugh maps and simplify an expression



Isaac Computer Science

Don't forget, IsaacComputerScience.org is your totally free online textbook and revision aid. It contains the entire GCSE and A-level syllabus and hundreds of high-quality self-assessment questions.



Thank you