Component 1 | 1.4.3| Boolean algebra

Specification & learning objectives

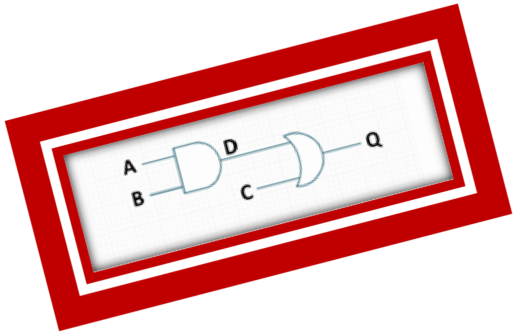
	A Level	Specification point description
	1.4.3a	Define problems using Boolean logic
	1.4.3b	Manipulate Boolean expressions, including the use of Karnaugh maps to simplify Boolean expressions
	1.4.3c	Use the following rules to derive or simplify statements in Boolean algebra: De Morgan's Laws, distribution, association, commutation, double negation
	1.4.3d	Using logic gate diagrams and truth tables
	1.4.3e	The logic associated with D type flip flops, half and full adders

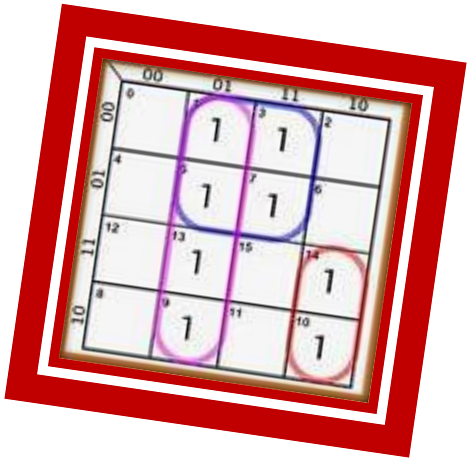
Resources

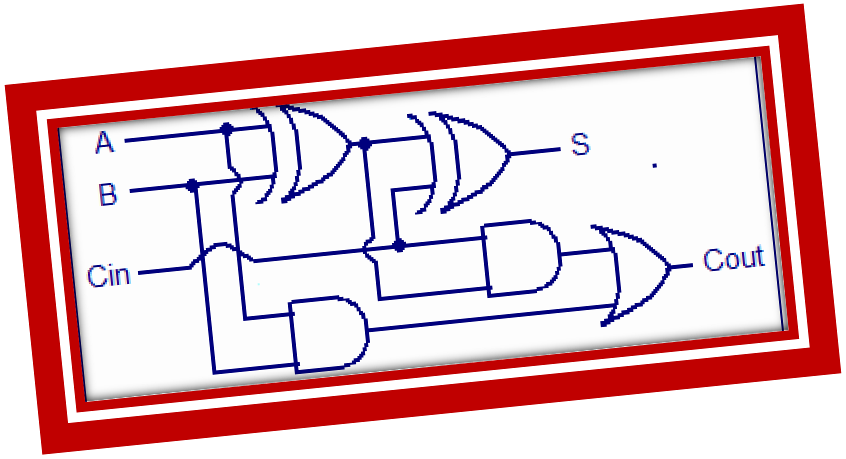
PG Online textbook page ref: 223-241

Hodder textbook page ref: 174-182

CraignDave videos for SLR 15

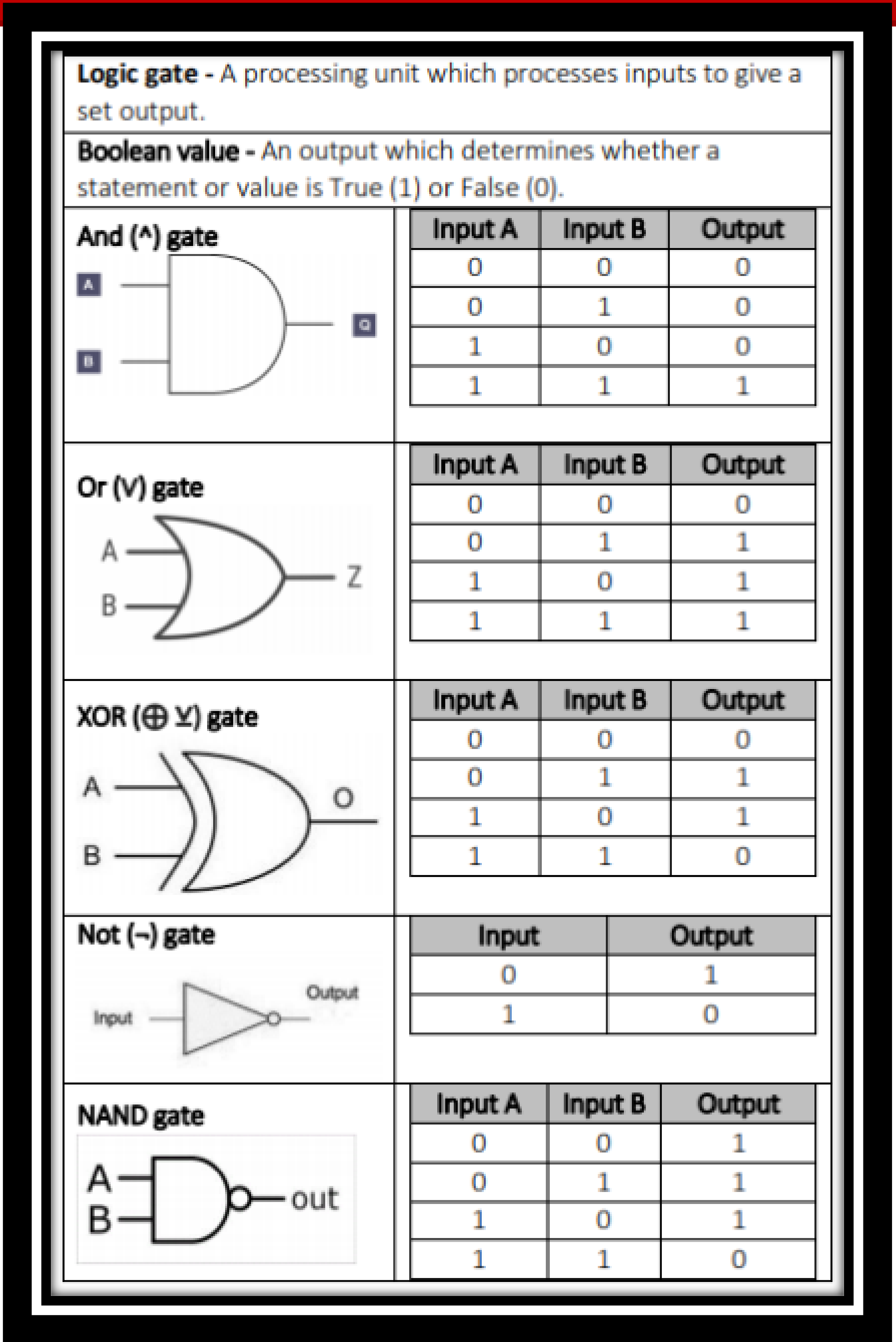






Component 1 | 1.4.3| Boolean algebra

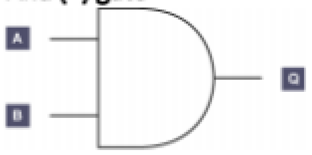
Key question: What are the Boolean operators and their associated logic gate symbols?



Logic gate - A processing unit which processes inputs to give a set output.

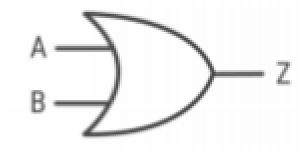
Boolean value - An output which determines whether a statement or value is True (1) or False (0).

And (^) gate



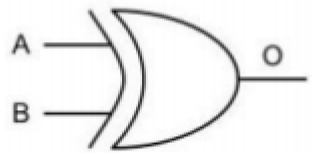
Input A	Input B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Or (V) gate



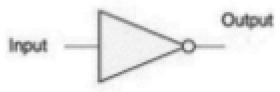
Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

XOR (⊕ ⊻) gate



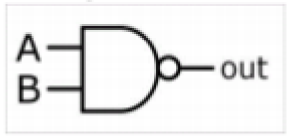
Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

Not (¬) gate



Input	Output
0	1
1	0

NAND gate



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

Component 1 | 1.4.3| Boolean algebra

Key question: How do you translate a logic gate diagram into its associated truth table and Boolean expression and vice versa?

(A ∧ D) ∨ (B ∧ D) ∨ (A ∧ ¬B ∧ C ∧ D)

(A ∧ D)

CD / AB	00	01	11	10
00				
01				
11		1	1	
10		1	1	

These two rows represent A being true

These two columns represent D being true

Where the rows overlap is the parts which represent A ^ D

(B ∧ D)

CD / AB	00	01	11	10
00				
01		1	1	
11		1	1	
10		1	1	

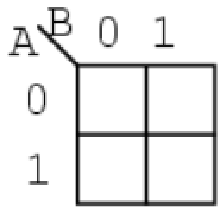
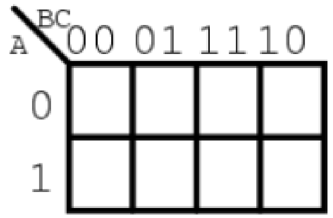
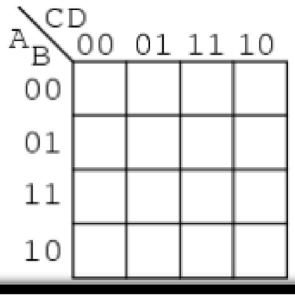
(A ∧ ¬B ∧ C ∧ D)

CD / AB	00	01	11	10
00				
01				
11			1	
10			1	

This can be simplified to (B ^ D) V (A ^ D)

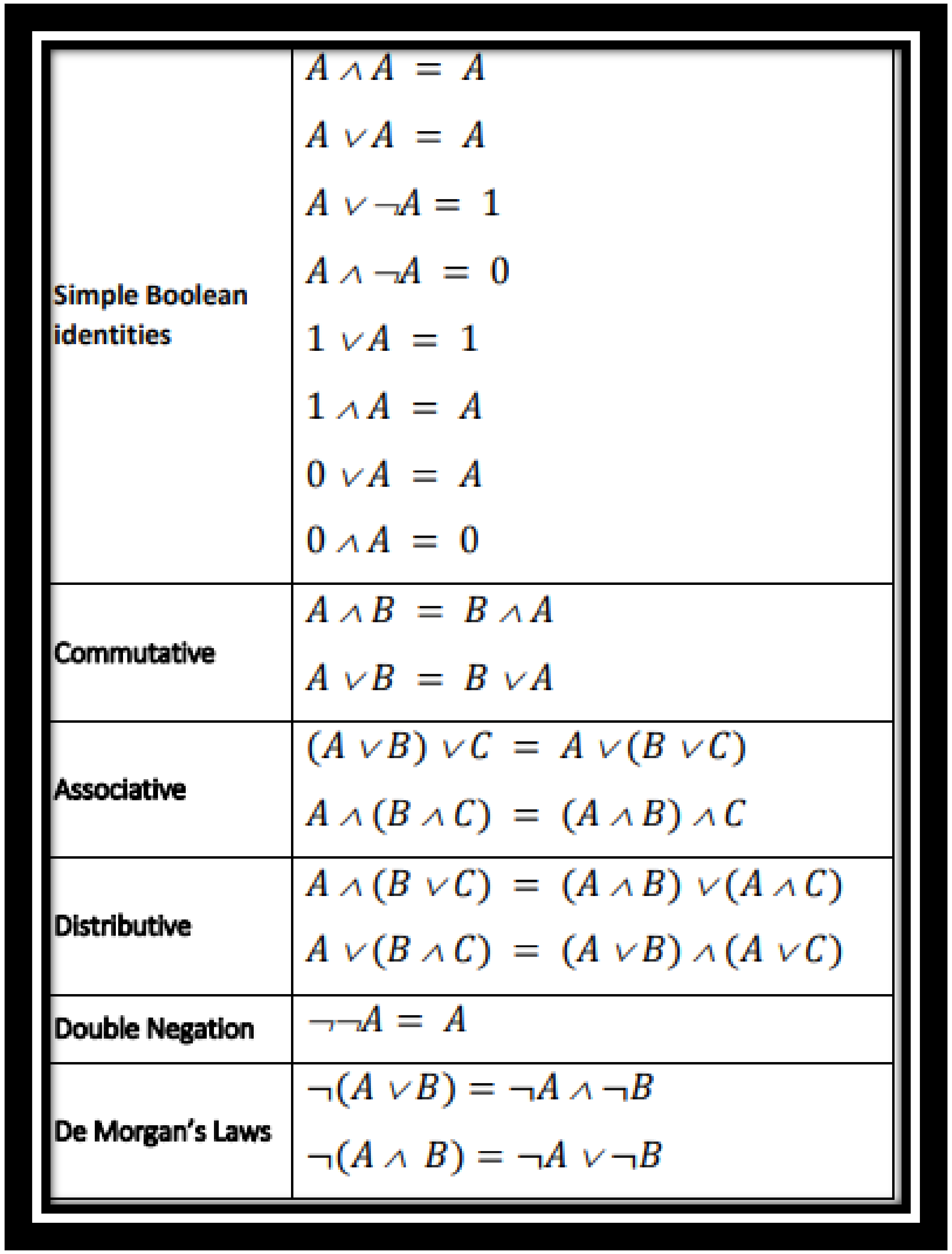
CD / AB	00	01	11	10
00				
01		1	1	
11		1	1	
10		1	1	

Key question: How can Karnaugh maps be used to simplify Boolean expressions?

Karnaugh Map	A Karnaugh map provides an alternative way of simplifying Boolean expressions which is often easier than using Boolean algebra for those involving up to 4 variables. Groupings are in groups of 1, 2, 4 and can go through walls and corners.
Two variable problem	
Three variable problem	
Four variable problem	

Component 1 | 1.4.3| Boolean algebra

Key question: What are the rules for simplifying Boolean expressions?

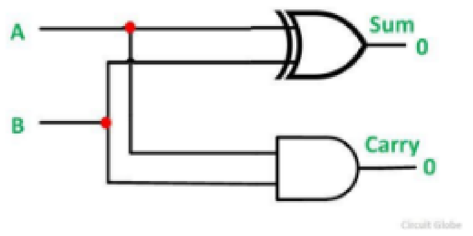


Simple Boolean identities	$A \wedge A = A$
	$A \vee A = A$
	$A \vee \neg A = 1$
	$A \wedge \neg A = 0$
	$1 \vee A = 1$
	$1 \wedge A = A$
	$0 \vee A = A$
	$0 \wedge A = 0$
Commutative	$A \wedge B = B \wedge A$
	$A \vee B = B \vee A$
Associative	$(A \vee B) \vee C = A \vee (B \vee C)$
	$A \wedge (B \wedge C) = (A \wedge B) \wedge C$
Distributive	$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
	$A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
Double Negation	$\neg\neg A = A$
De Morgan's Laws	$\neg(A \vee B) = \neg A \wedge \neg B$
	$\neg(A \wedge B) = \neg A \vee \neg B$

Component 1 | 1.4.3| Boolean algebra

Key Circuits

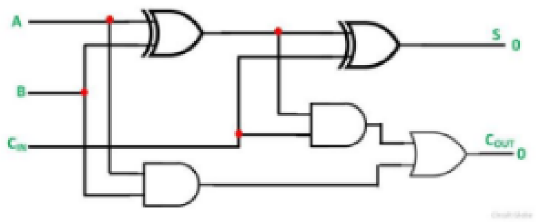
Half adder circuit



A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

This circuit returns a 1 in the sum output when only one input is on, and a 1 in the carry when both inputs are 1. This mimics the rules of binary addition.

Full adder circuit



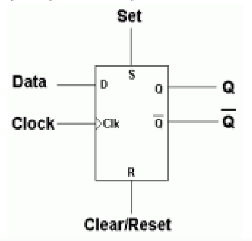
This circuit is used to ensure that the carry value is brought forward. It is made by linking 2 half adder circuits. A series of these circuits allows a computer to add binary numbers.

A	B	C IN	C OUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Key Circuits

Flip Flop

Created from several NAND (or AND and OR) gates and is effectively 1 single bit of memory. To store 8 bits, 8 flip flops are required.

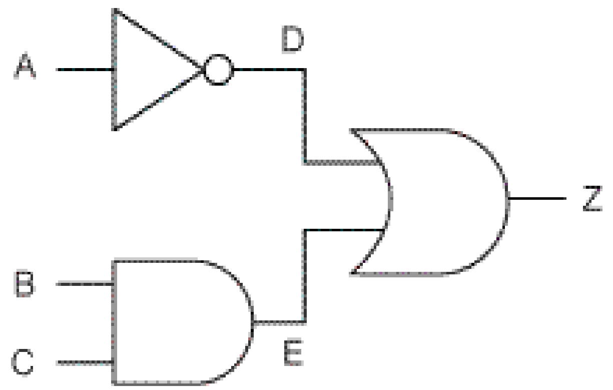


Typical exam questions

1. Simplify the following expression: **[6]**

A∧B∧C∨A∧¬B∧C∨A∧B∧¬C

2. Complete the truth table for the following logic diagram. **[8]**



A	B	C	D	E	Z

3. State the three different logic gates used in the logic diagram above. **[3]**

Component 1 | 1.4.3| Boolean algebra

Target: ☐ Overall grade: ☐

Minimum expectations & learning outcomes

☐	Terms 187-199 from your A Level Key Terminology should be included and formatted.
☐	You must show the general rules for simplification of Boolean expressions.
☐	You must include annotated examples of a two, three and four input Karnaugh Map, and show how it is used to simplify a Boolean expression.
☐	You must include an example of each of the following rules and show clearly how they are used to help simply Boolean statements: De Morgan's Laws, distribution, association, commutation, double negation.
☐	You must show a series of logic gate diagrams and their associated truth tables.
☐	You must illustrate D type flip flops, half and full adders explaining where they are used in a CPU.
☐	Answer the exam questions.

Feedback

Breadth	Depth	Presentation	Understanding
☐ All	☐ Analysed	☐ Excellent	☐ Excellent
☐ Most	☐ Explained	☐ Good	☐ Good
☐ Some	☐ Described	☐ Fair	☐ Fair
☐ Few	☐ Identified	☐ Poor	☐ Poor

Comment & action required

Reflection & Revision checklist

Confidence	Clarification
☹ 😐 ☺	Candidates should be familiar with AND, OR, NOT and XOR. Candidates should be familiar with the logic of each Boolean operator, and the truth tables.
☹ 😐 ☺	Candidates should be able to construct logic gate diagrams from a Boolean expression and vice versa.
☹ 😐 ☺	Candidates should be able to construct truth tables from Boolean expressions and logic gate diagrams.
☹ 😐 ☺	Candidates should have an understanding that Boolean expressions can be simplified and should have experience of simplifying expressions using Karnaugh maps.
☹ 😐 ☺	Candidates should be able to create, complete and interpret Karnaugh maps to simplify Boolean expressions.
☹ 😐 ☺	Candidates should be aware of the given De Morgan's laws and should be able to apply these to a Boolean statement.
☹ 😐 ☺	Candidates should have experience of manipulating and simplifying Boolean statements using these rules of distribution, commutation, association and double negation.
☹ 😐 ☺	Candidates need to understand the purpose and principles of D type flip flops and how and where they are used in a computer. They should be able to recognise how they can be triggered by a clock pulse.
	Candidates are not expected to memorise the logic gates that make up a D-type flip flop.
☹ 😐 ☺	Candidates need to understand the purpose and function of an adder circuit, and the difference between a half and full adder. They should be able to recognise and draw the logic gates and truth tables for full and half adders.