OCR Computer Science A Level

1.4.3 Boolean Algebra

Concise Notes

Specification:

1.4.3 a)

- Define problems using Boolean logic

1.4.3 b)

- Manipulate Boolean expressions
  - Karnaugh maps to simplify Boolean expressions

1.4.3 c)

- Use the following rules to derive or simplify statements in Boolean algebra:
  - De Morgan's Laws
  - Distribution
  - Association
  - Commutation
  - Double negation

1.4.3 d)

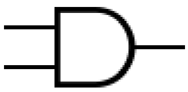
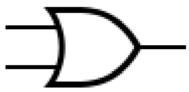
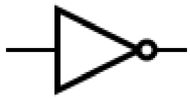
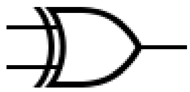
- Using logic gate diagrams and truth tables

1.4.3 e)

- The logic associated with D type flip flops, half and full adders

Logic Gate Diagrams and Truth Tables

- Problems can be defined using Boolean logic in Boolean equations
- A Boolean equation can equate to either True or False
- Four operations are used:

Operation	Conjunction	Disjunction	Negation	Exclusive Disjunction
Logic gate				
	AND	OR	NOT	XOR
Symbol	$\wedge$	$\vee$	$\neg$	$\veebar$

Truth tables

- A table showing every possible permutation of inputs to a logic gate and the corresponding output
- Inputs are usually labeled A, B, C etc
- 1 represents True, 0 represents False

Conjunction (AND)

- Applied to two literals (or inputs) to produce a single output
- Can be thought of as applying multiplication to its inputs
- Truth table shows $A \wedge B = Y$

AND

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Disjunction (OR)

- Operates on two literals and produces a single output
- Can be thought of as applying addition to its inputs
- As long as one input is True then the output is True
- Truth table shows $A \vee B = Y$

OR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Negation (NOT)

- Only applied to one literal
- Reverses the truth value of the input
- Truth table shows $\neg A = Y$

NOT

A	Y
0	1
1	0

Exclusive Disjunction (XOR)

- Also known as exclusive OR
- Similar to disjunction but differs when both inputs are True
- Only outputs True when exactly one input is True
- Otherwise output is False
- Truth table shows $A \veebar B = Y$

XOR

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Combining Boolean Operations

- Boolean equations are made by combining Boolean operators
- This is done in the same way that standard mathematical operators are combined
- Every boolean equation can be represented with a truth table

Manipulating Boolean Expressions

- Sometimes a long Boolean expression has the same truth table as another, shorter expression
- It tends to be desirable to use the shorter versions
- There are a variety of methods which can be used to simplify expressions

Karnaugh Maps

- Can be used to simplify Boolean expressions
- The tables are filled in corresponding to the expression's truth table
- Can be used for a truth table with two, three or four variables
- It's important that the values in the columns and rows are written using Gray code
- Columns and rows only ever differ by one bit, including wraparound
- To simplify a Boolean expression:
    - First write your truth table as a Karnaugh map
    - Then highlight all of the 1s in the map with a rectangle
    - The larger the rectangle you can highlight at once the better
    - Only groups of 1s with edges equal to a power of 2 (1, 2 or 4 in a row) can be highlighted, wraparound is included
    - Remove variables which change within these rectangles from the expression
    - Keep variables which do not change, but negate to become True if required

Simplifying Boolean Algebra

De Morgan's Laws

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

Distribution

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$
$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge (A \wedge C)$$
$$A \vee (B \vee C) \equiv (A \vee B) \vee (A \vee C)$$

Association

$$(A \wedge B) \wedge C \equiv A \wedge (B \wedge C) \equiv A \wedge B \wedge C$$
$$(A \vee B) \vee C \equiv A \vee (B \vee C) \equiv A \vee B \vee C$$

Commutation

$$A \vee B \equiv B \vee A$$
$$A \wedge B \equiv B \wedge A$$

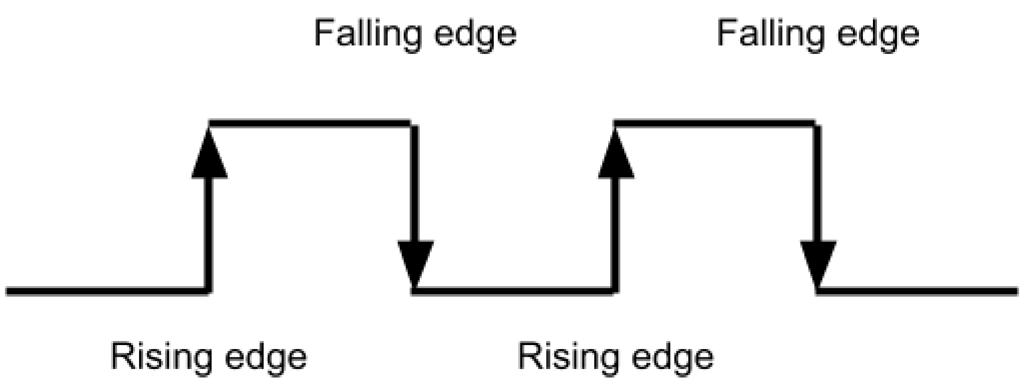
Double Negation

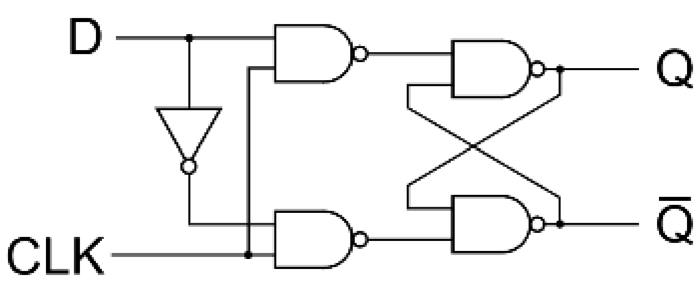
$$\neg\neg A \equiv A$$

Logic Circuits

D-Type Flip Flops

- Logic circuit which can store the value of one bit
- Two inputs, a control signal and a clock
  - A clock is a regular pulse generated by the CPU which is used to coordinate the computers' components
  - A clock pulse rises and falls as shown in the diagram
  - Edges can be classified rising or falling
  - The output of a D-type flip flop can only change at a rising edge, the start of a clock tick
- Logic circuit uses four NAND gates
- Updates the value of Q to the value of D whenever the clock (CLK) rises
- The value of Q is the stored value





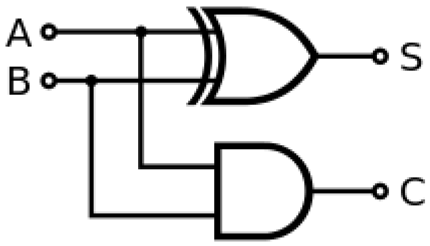
Adders

- A logic circuit which adds together the number of inputs which are True
- Outputs this number in binary

Half Adder

- Two inputs, A and B
- Two outputs, Sum and Carry
- Formed from two logic gates: AND and XOR
- When both A and B are False, both outputs are False
- When one of A or B is True, Sum (S) is True
- When both inputs are True, Carry (C) is True

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Full Adder

- Similar to a half adder, but has an additional input
- Allows carry in to be represented
- Formed from two XOR gates, two AND gates and an OR gate
- Can be chained together to form a ripple adder with many inputs

A	B	C_{in}	C_{out}	Sum
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