# Boolean Algebra Worksheet 1

### Boolean Simplification

**AND** is called conjunction. It is represented by a dot (.). We are said to be ‘multiplying’ when we perform an AND in binary, i.e. 0x0 = 0, 1x0 = 0, etc.

**Exercise 1** – Fill in the table

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**OR** is called disjunction. It is represented by a plus sign (+). This is because we are said to be ‘adding’ when we perform an OR in binary, i.e. 0+0 = 0, 1+0 = 1, etc.

**Exercise 2** – Fill in the table

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Q** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**NOT** is represented by a line above the value (Ā).

**Exercise 3** – Fill in the table

|  |  |
| --- | --- |
| **A** | **Q** |
| 0 | 1 |
| 1 | 0 |

**XOR** is represented by the OR symbol inside a circle $⊕$. In the case where we have 1+1, XOR produces 0.

**NAND** is represented by $\overbar{(A.B)}$.

**NOR** is represented by $\overbar{(A+B)}$.

**Exercise 4** – Draw out the truth tables for the above logic gates

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **A⊕B** | **A.B** | $$\overbar{AB}$$ | **A+B** | $$\overbar{A+B}$$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |

RULE1 – Simple Boolean identities

1a. A.A = A

1b. A+A = A

1c. A+$\overbar{A}$ = 1

1d. A.$\overbar{A}$ = 0

1e. 1+A = 1

1f. 0+A = A

1g. 0.A = 0

1h. 1.A = A

**Exercise 5** – Fill out the table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | $$\overbar{A}$$ | **A.A** | **A+A** | **A+**$\overbar{A}$ | **A.**$\overbar{A}$ |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |

RULE2 – Commutative laws

The order of the statements in OR/AND does not change the outcome.

2a. A+B = B+A

2b. A.B = B.A

**Exercise 6** – Fill out the table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **A+B** | **B+A** | **A.B** | **B.A** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

RULE3 – Associative laws

3a. A+(B+C) = (A+B)+C

3b. A.(B.C) = (A.B).C

**Exercise 7** – Fill out the table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **B+C** | **A+(B+C)** | **A+B** | **(A+B)+C** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

RULE4 – Distributive laws

We can expand of contract a bracket just like in ‘normal’ maths. Notice how we also do this for ‘OR’ whereas we would usually not expand brackets for a + operator.

4a. A.(B+C) = (A.B)+(A.C)

4b. A+(B.C) = (A+B).(A+C)

**Exercise 8** – Fill out the table

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **B+C** | **A.(B+C)** | **A.B** | **A.C** | **(A.B)+(A.C)** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **A** | **B** | **C** | **B.C** | **A+(B.C)** | **A+B** | **A+C** | **(A+B).(A+C)** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

RULE5 – Double negation

If a value is doubly negated (two lines above it), it is the positive version of itself

$̿$ = A

**Exercise 9** – Fill out the table

|  |  |  |
| --- | --- | --- |
| **A** | $$\overbar{A}$$ | $$̿$$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

RULE6 – De Morgan’s Laws

De Morgan’s Laws are used to simplify Boolean equations so that you can build equations using only one sort of gate, usually NAND or NOR gates. You can build cheaper hardware this way.

The general idea of the laws are: If you break the bar, you have to change the sign.

**6a.**

$$\overbar{A.B}= \overbar{P}+\overbar{Q}$$

**Exercise 10** – Fill out the table to prove this is true

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P** | **Q** | **P.Q** | $$\overbar{P.Q}$$ | $$\overbar{P}$$ | $$\overbar{Q}$$ | $\overbar{P}$**+**$\overbar{Q}$ |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

**6b.**

$$\overbar{A+B}= \overbar{P}.\overbar{Q}$$

**Exercise 11** – Fill out the table to prove this is true

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P** | **Q** | **P+Q** | $$\overbar{P+Q}$$ | $$\overbar{P}$$ | $$\overbar{Q}$$ | $\overbar{P}$**.**$\overbar{Q}$ |
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

# Crib Sheet

Simple Boolean identities

|  |  |
| --- | --- |
| A.A = A A+A = AA+$\overbar{A}$ = 1A.$\overbar{A}$ = 01+A = 1 1.A = A0+A = A 0.A = 0 Commutative AB = BA A+B = B+A. Associative(A+B)+C = A+(B+C) A.(B.C) = (A.B).CDistributiveA.(B+C) = (A.B)+(A.C) A+(B.C) = (A+B).(A+C)Double negation$̿$ = A |   |

# Example Questions

Write the following Boolean expressions in their simplest forms

For each question also create a truth table to prove each example.

1. ($\overbar{A ̅.B ̅}$)

 Using De Morgan’s and double negation (Rules 6a and 5) produces $̿$ + $̿$

 The double negation rule (Rule 5) leaves us with AB

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

 Using distributive Rule 4a A.(B+$\overbar{B}$).

 Rule 1c states B+$\overbar{B}$=1. We are now left with A.1.

 Rule 1h states that A.1 = A

3. A.(B+1)

 Using Rule 1e B+1 = 1. Therefore, we are left with A.1. Rule 1h is A.1 = A.

4. $\overbar{(A.B)}$+$\overbar{(A+\overbar{B)}}$

 Using De Morgan’s Rule 6a $\left(\overbar{A}+\overbar{B}\right)+( \overbar{A}+B)$

 Then apply rule 1a $\overbar{A}+\overbar{A}=\overbar{A}$

 Which gives $\overbar{A}+B+\overbar{B} $but Rule 1c makes this $\overbar{A}+1$

 We can then apply rule 1e to give $\overbar{A}$ + 1 = 1

5. $\left(X+Y\right).(X+\overbar{Y})$

 Using distributive Rule 4b we get X + (Y + $\overbar{Y}$)

 Using Rule 1d this gives us X + 0

 Rule 1f says that X + 0 = X

6. $\overbar{\overbar{A}+\overbar{(B.A)}}$

 Using De Morgan’s Law $\overbar{\overbar{A}+\overbar{B}+\overbar{A}}$

 Using Rule 1a $\overbar{\overbar{A}+\overbar{B}}$

 Reapplying De Morgan’s Law gives $̿+̿$, and using double negation (Rule 5) gives A+B

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

 Using the distributive law 4a this gives A.($B.\overbar{C}+\overbar{C}))$

 We can then use the distributive rule again to give A.($\overbar{C}(B+1))$

 Because of Rule 1e we get A.$\overbar{C}(1))$

 The answer is therefore A.$\overbar{C}$

8. $B.(A+\overbar{A})$

 Using Rule 1c, B.(1) = B

9. A.B+B

 Using distributive law B.(A+1)

 Using Rule 1e B.(1) = B

10. $\overbar{B} .\overbar{\overbar{A}+\overbar{B}}$

 Using De Morgan’s Law $\overbar{B} .(A.B$)

 Using Rule 1d $\overbar{B}$.B.A = 0.A

 Using Rule 1g 0.A = 0