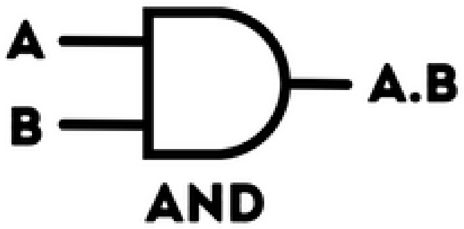
Logic Circuits

YORK COLLEGE & UNIVERSITY CENTRE

AND gate

- An AND gate uses two inputs to generate one output
- The output is 1 (TRUE) only if both inputs are 1 (TRUE)
- AND gates are also known as a conjunction
- AND has the notation **A.B** / **AB** / **A*B** / **A AND B**

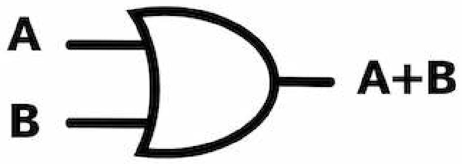


2 input AND Gate

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

- An OR gate uses two inputs to generate one output
- The output is 1 (TRUE) if either of the inputs are 1 (TRUE)
- OR gates are also known as a disjunction
- OR gates have the notation **A+B** / **A OR B**

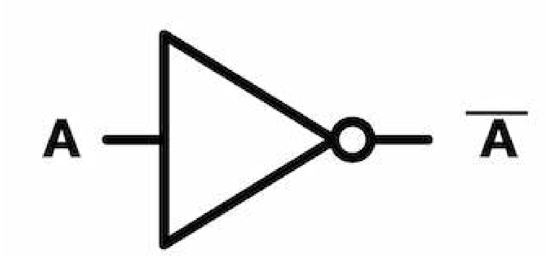


2 input OR gate

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate

- A NOT gate takes 1 input to generate 1 output
- The output is the inverse of the input
- NOT gates are also known as the negation or an inverter
- NOT gates have the notation **Ā** / **~A** / **NOT A** / **A’**

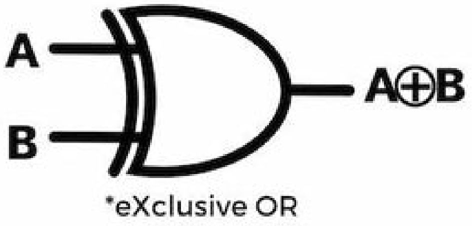


2 input NOT gate

A	Ā
0	1
1	0

XOR gate

- An XOR gate takes 2 inputs to generate 1 output
- It’s very similar to an OR gate but only outputs true if one of the two inputs are true not both
- XOR gates are also known as an exclusive disjunction
- XOR gates have the notation **A⊕B** / **AXORB**

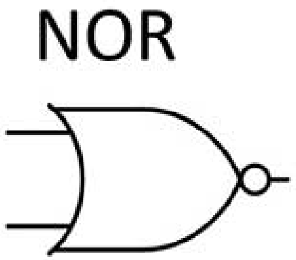


2 input XOR gate

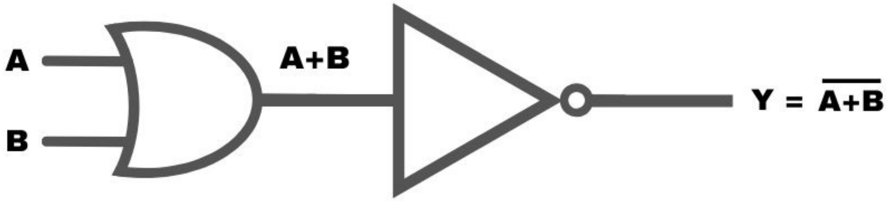
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

NOR gate

- A NOR gate takes 2 inputs to generate 1 output
- It's the combination of an OR gate and a NOT gate
- NOR gates have the notation **(A+B)'** / $\overline{\mathbf{A+B}}$ / **ANORB**

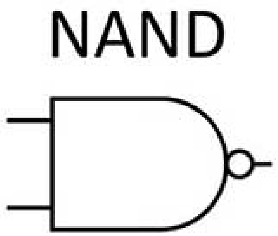


A	B	Output
0	0	1
1	0	0
0	1	0
1	1	0

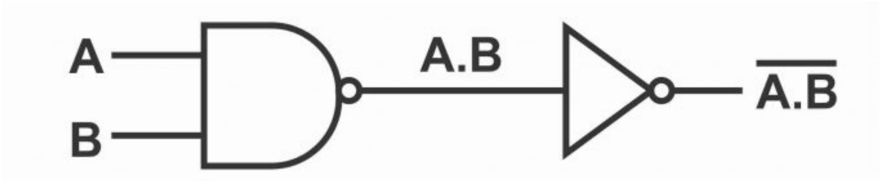


NAND gate

- A NAND gate takes 2 inputs to generate 1 output
- It’s the combination of an AND gate and a NOT gate
- AND gates have the notation **(A.B)’** / $\overline{\mathbf{AB}}$ **/ ANANDB**

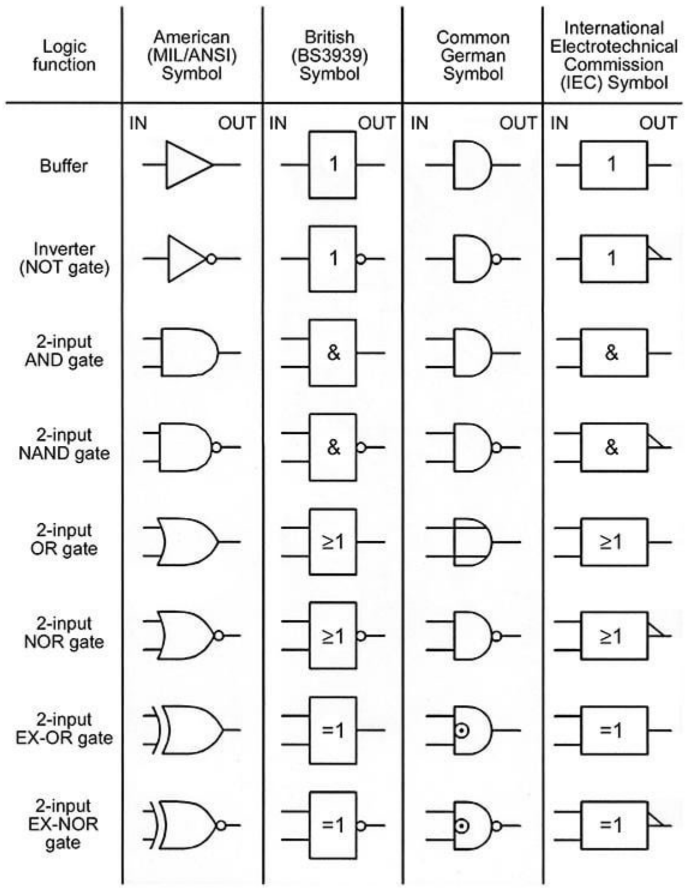


A	B	Output
0	0	1
1	0	1
0	1	1
1	1	0



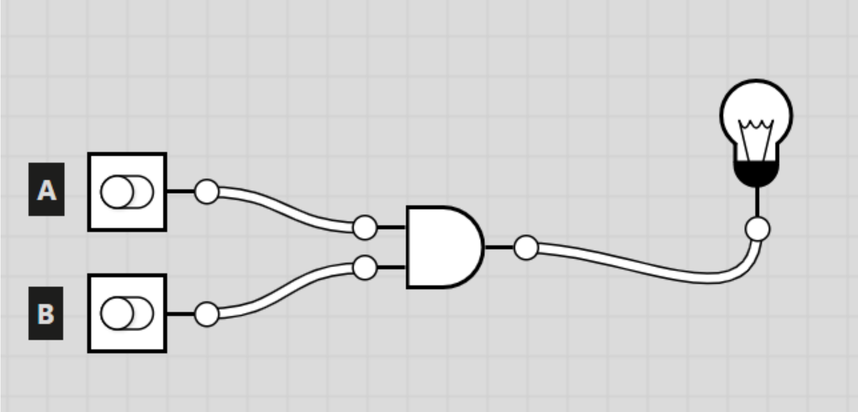
Symbol Standards

- So far, we have looked at the American standard symbols
- The American standard has become the international standard
- Both BS and IEC are deprecated and thus aren’t used commonly



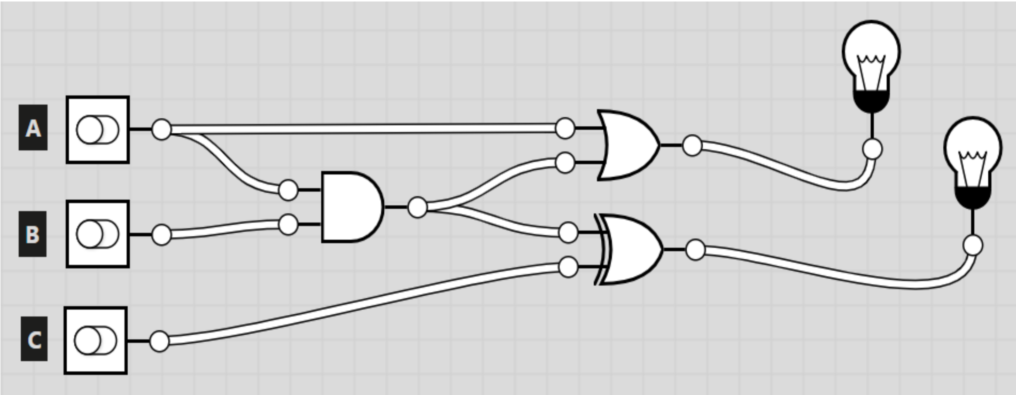
Truth Tables

- Truth Tables help us to understand the output of a binary logic circuit
- They systematically lists all possible combinations of truth values (true or false, often represented as 1 and 0) for given input variables



A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

More Complex Truth Tables



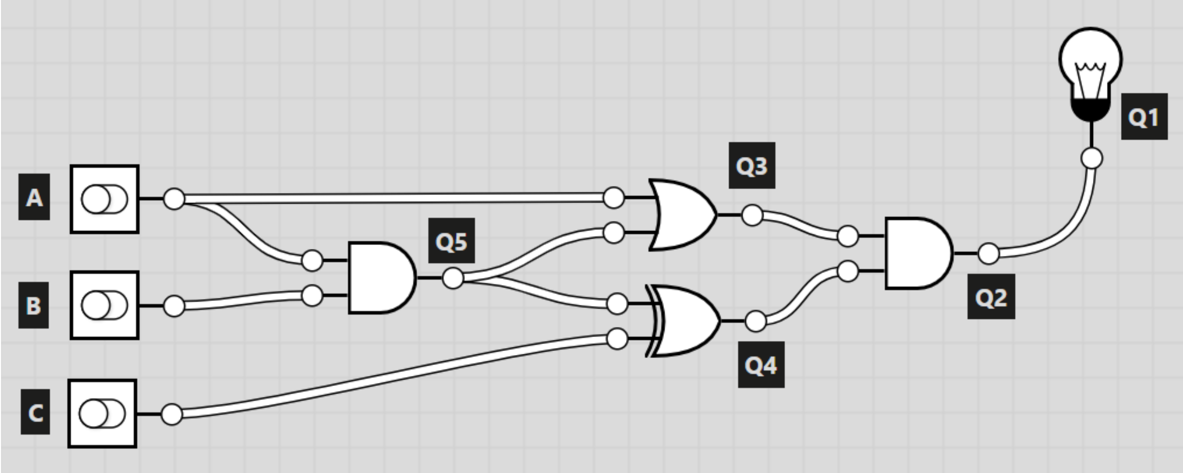
A	B	C	A.B	A+(A.B)	C⊕(A.B)
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	1	1
1	1	1	1	1	0

More complex truth table simplified

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

Binary Algebra

- Binary Algebra uses binary notation to understand a Binary logic circuit
- We can trace backwards through our binary logic circuit to write out our notation
- We can then put in our numbers to get a quick answer to the outcome



Q1 = Q2 = Q3.Q4

Q3 = A+Q5

Q5 = A.B

Q3 = A+(A.B)

Q4 = Q5⊕C

Q4 = (A.B)⊕C

Q1 = (A+(A.B)).((A.B)⊕C)

Binary Algebra – Finding the Outcome

Q1 = (A+(A.B)).((A.B)⨁C)

A = 1

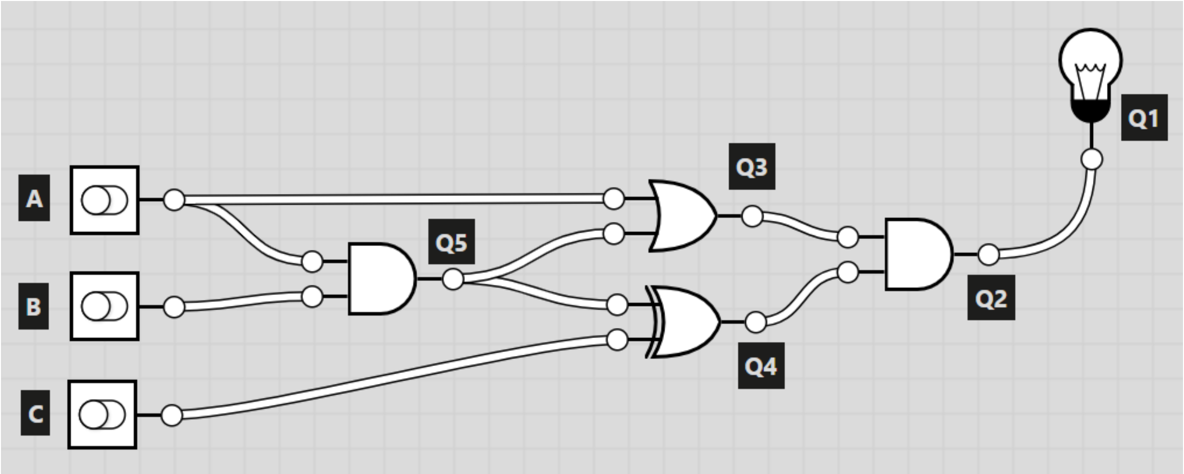
B = 0

C = 1

(1+(1*0))*((1*0)⨁1)

(1+0)*(0⨁1)

1*1 = 1



Simulating Binary Logic Circuits

https://logic.ly/

Classwork