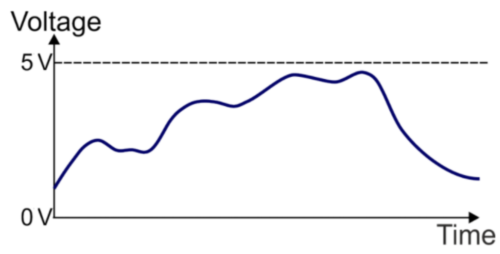
AS-1 Combinational Logic

Revision sheet

1 Signals

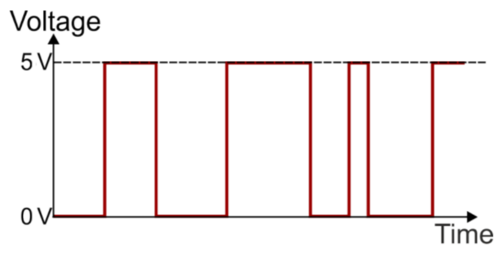
There are two types of signals.

1.1 Analogue Signal



Analogue signals can have any value between the minimum and maximum.

1.2 Digital Signals



Digital signals can either be on (logic 1 / +5V) or off (logic 0 / 0V).

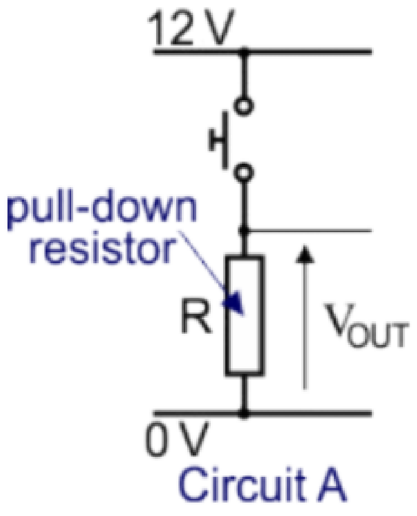
2 Digital Inputs and Outputs

There are a number of different digital input components: Push-To-Make button; Push-To-Break button; latching switch.

2.1 Push-To-Make Buttons

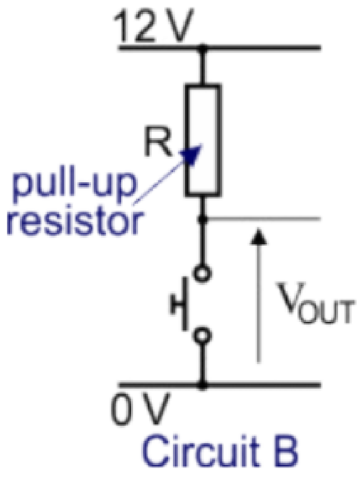
Buttons require a pull up or pull down resistor. This will make sure that when the button isn't pressed (in the case of PTM), the value outputted is an actual value, not undefined/floating.

2.1.1 Active High button



This is normally outputting a low signal until the button is pressed, then it outputs a high signal.

2.1.2 Active Low Button

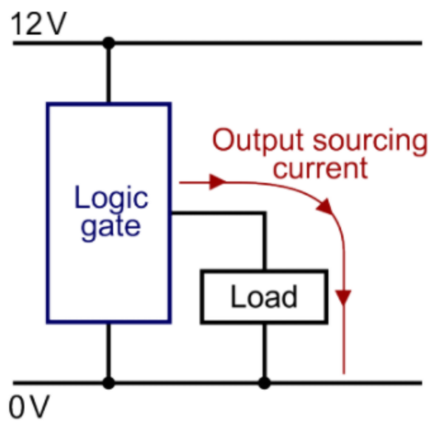


This is normally outputting a high signal until the button is pressed, then it outputs a low signal.

2.2 Sourcing and Sinking

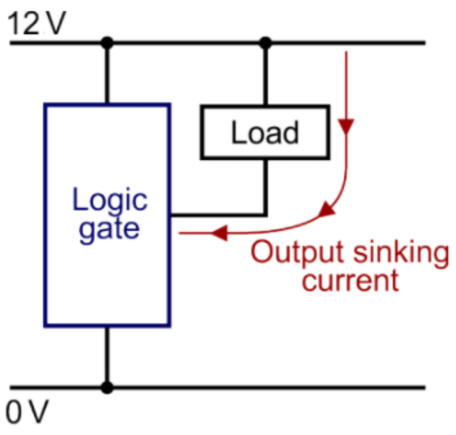
Logic gates can either source or sink current. This means they can either output current or take in current. This is extremely useful as it allows us to power active high and active low components off of the outputs of logic gates.

2.2.1 Sourcing



Current flows from the gate to the 0V rail. The gate is sourcing current.

2.2.2 Sinking



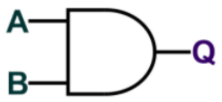
Current flows from the load and into the gate. The gate is sinking current.

3 Logic Gates and Truth Tables

For the most part, logic gates can have multiple inputs and one output. Pictured here are the smallest arrangement they come in. In real life, logic gates come in chips, generally bundled into sets of 4 or 6 in one chip.

3.1 AND Gate

This only outputs when all inputs are logic high.



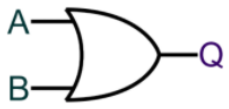
B	A	Q
0	0	0
0	1	0
1	0	0
1	1	1

Table 1: Truth Table for AND Gate

Boolean Expression: $Q = A \cdot B$

3.2 OR Gate

This outputs 1 when either or both are logic high.



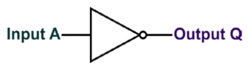
B	A	Q
0	0	0
0	1	1
1	0	1
1	1	1

Table 2: Truth Table for OR Gate

Boolean Expression: $Q = A + B$

3.3 NOT Gate

This is a single input gate. It inverts the signal in.



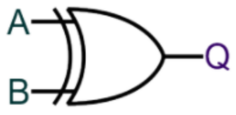
A	Q
0	1
1	0

Table 3: Truth Table for NOT Gate

Boolean Expression: $Q = \overline{A}$

3.4 XOR Gate

This outputs only when input A or B is logic high - eXclusiveOR.



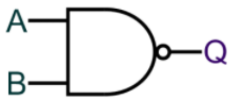
B	A	Q
0	0	0
0	1	1
1	0	1
1	1	0

Table 4: Truth Table for XOR Gate

Boolean Expression: $Q = A \oplus B$

3.5 NAND Gate

This is the same as an AND gate, except it inverts the output.



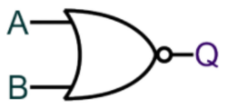
B	A	Q
0	0	1
0	1	1
1	0	1
1	1	0

Table 5: Truth Table for NAND Gate

Boolean Expression: $Q = \overline{A \cdot B}$

3.6 NOR Gate

This only outputs logic high if both inputs are logic 0 - NotOR. This makes it useful for checking if something is 0.



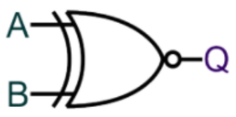
B	A	Q
0	0	1
0	1	0
1	0	0
1	1	0

Table 6: Truth Table for NOR Gate

Boolean Expression: $Q = \overline{A + B}$

3.7 XNOR Gate

This outputs if all inputs are logic 0 or if all inputs are logic 1 - eXclusiveNotOR. This is useful for checking if the inputs are the same.



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

Table 7: Truth Table for XNOR Gate

Boolean Expression: $Q = \overline{A \oplus B}$

4 Manually Simplifying Boolean Algebra

4.1 Boolean Identities

$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$
$\overline{A \oplus B} = A \cdot B + \overline{A} \cdot \overline{B}$

4.2 Single Variable Rules

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A = A$$
$$A \cdot \overline{A} = 0$$
$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A = A$$
$$A + \overline{A} = 1$$

4.3 Rules

4.3.1 Commutative

The order doesn't matter.

$$A \cdot B = B \cdot A$$
$$A + B = B + A$$

4.3.2 Associative

Some can be re-grouped to make simplification easier.

$$
\begin{aligned}
A+(B+C) &= (A+B)+C \\
&= A+B+C
\end{aligned}
$$

$$
\begin{aligned}
A\cdot(B\cdot C) &= (A\cdot B)\cdot C \\
&= A\cdot B\cdot C
\end{aligned}
$$

4.3.3 Distributive

$$
\begin{aligned}
A\cdot B + A\cdot C &= A(B+C) \\
C\cdot(B\cdot A + D) &= C\cdot B\cdot A + C\cdot D \\
(A\cdot B)\cdot(C\cdot D) &= A\cdot C + A\cdot D + \\
&\quad B\cdot C + B\cdot D \\
A+(B\cdot C) &= (A+B)\cdot(A+C)
\end{aligned}
$$

4.3.4 Absorption / Redundancy

$$
\begin{aligned}
A + A\cdot B &= A \\
A + \overline{A}\cdot B &= A + B
\end{aligned}
$$

4.3.5 Double Inversion

$$\overline{\overline{A}} = A$$

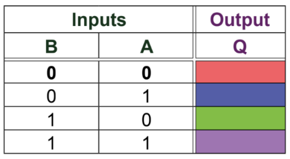
4.4 deMorgan's Theorem

Break the bar, change the sign.

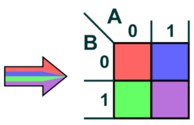
$$
\begin{aligned}
\overline{A\cdot B} &= \overline{A}+\overline{B} \\
\overline{A+B} &= \overline{A}\cdot\overline{B}
\end{aligned}
$$

5 Karnaugh Maps

Karnaugh maps can be used to simplify boolean algebra expressions. In their simplest form, they have a grid of two across and two down. This is the size which would be needed to simplify a boolean expression for a single logic gate.

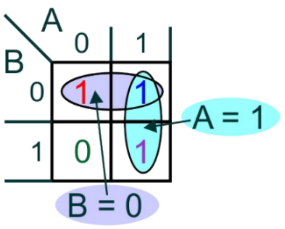


Inputs		Output
B	A	Q
0	0	🟥
0	1	🟦
1	0	🟩
1	1	🟪



5.1 Simplification using a Karnaugh Map

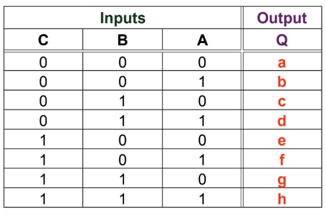
Karnaugh maps allow us to simplify boolean expressions graphically. This works by identifying groups of 2, 4 or 8 neighbouring cells containing logic 1 (these can also wrap around the edge of the table); finding the term common to each groups then combining the terms together using the OR operator. Every cell containing logic 1 must be included in at least one group.



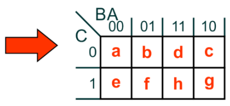
In the example above, the top right cell is in two groups, this is okay as long as each group as at least one of the cells in the group is only in one group. The cells in the blue group are in the A=1 column, this means the term common to them is A. The cells in the purple group are in the B=0 row, this means the term common to them is $\overline{B}$. Combining the blue and purple group gives us $Q = \overline{B} + A$.

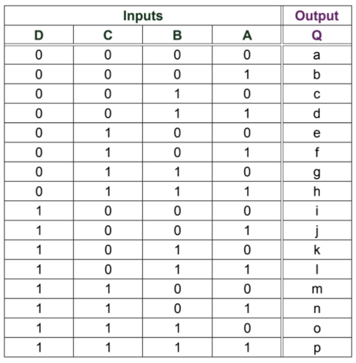
5.2 Bigger Karnaugh Maps

Karnaugh maps can have many more inputs. Shown below are three-input and four-input Karnaugh maps.

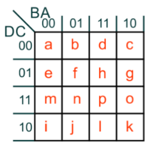


Inputs			Output
C	B	A	Q
0	0	0	a
0	0	1	b
0	1	0	c
0	1	1	d
1	0	0	e
1	0	1	f
1	1	0	g
1	1	1	h



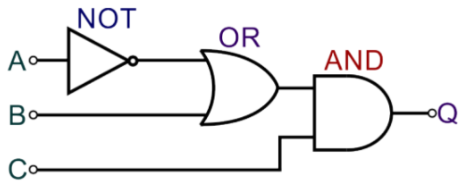


Inputs				Output
D	C	B	A	Q
0	0	0	0	a
0	0	0	1	b
0	0	1	0	c
0	0	1	1	d
0	1	0	0	e
0	1	0	1	f
0	1	1	0	g
0	1	1	1	h
1	0	0	0	i
1	0	0	1	j
1	0	1	0	k
1	0	1	1	l
1	1	0	0	m
1	1	0	1	n
1	1	1	0	o
1	1	1	1	p

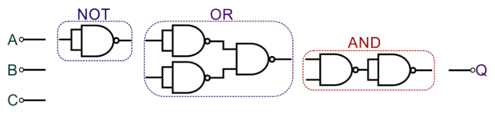


6 NAND Gate Conversion

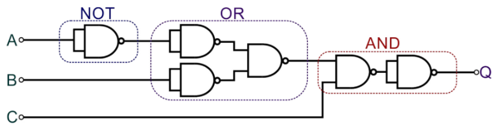
Any logic gate can be constructed from NAND Gates. This is good as it means the number of chips needed for a circuit can be reduced as you would only need NAND gates. The example below shows the stages to go through to convert a system which uses lots of different logic gates into a NAND Gate simplified system.
This is the logic system which is going to be converted:



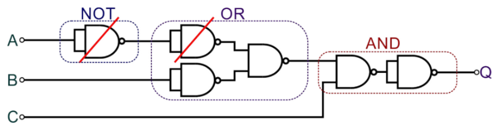
1. Replace each gate with its NAND equivalent.



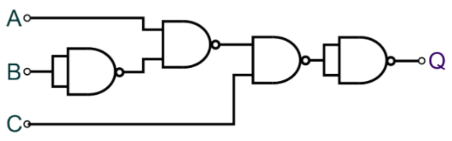
2. Connect the equivalents together



3. Cross out redundant gates (gates where there are two single inputs in a row).

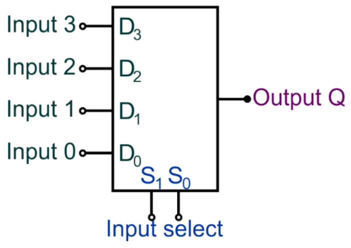


The final circuit is shown below:

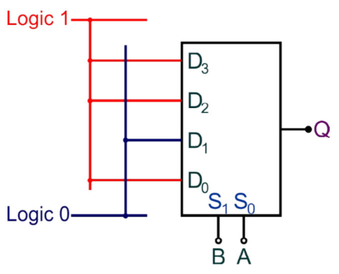


7 Multiplexers

Multiplexers have several inputs and one output. You can select which input is sent to the output.



Multiplexers are connected to logic 1 and 0 as shown below:



Multiplexers are useful for complicated logic systems where a large number of logic gates would otherwise be required.