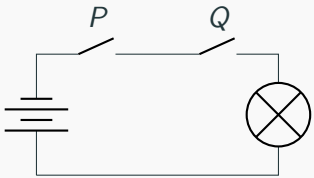
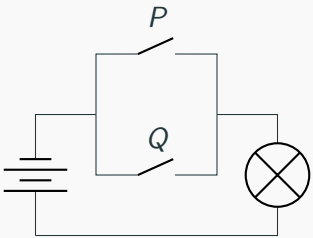
Application: Digital logic circuits

Logic and electric circuits



"in series"

P	*Q*	light
closed	*closed*	on
closed	*open*	off
open	*closed*	off
open	*open*	off



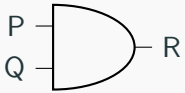
"in parallel"

Q	*Q*	light
closed	*closed*	on
closed	*open*	on
open	*closed*	on
open	*open*	off

Modern computers use logic gates

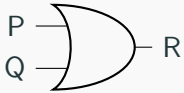
Basic logic gates

AND gate



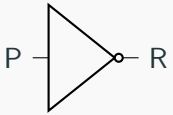
P	Q	R
1	1	1
1	0	0
0	1	0
0	0	0

OR gate



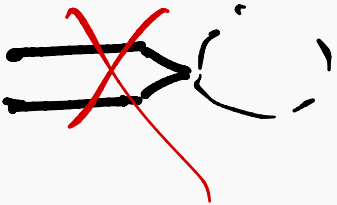
P	Q	R
1	1	1
1	0	1
0	1	1
0	0	0

NOT gate



P	R
1	0
0	1

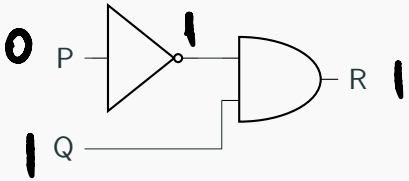
Rules for a combinatorial circuit



- Never combine two input wires.
- A single input wire can be split partway and used as input for two separate gates.
- An output wire can be used as input.
- *No output of a gate can eventually feed back into that gate.*

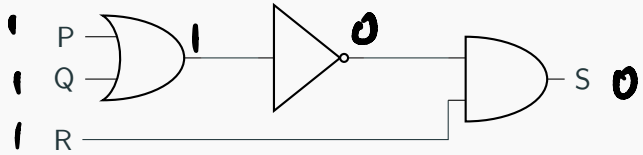
Determining output for a given circuit

Input signals: $P = 0$ and $Q = 1$



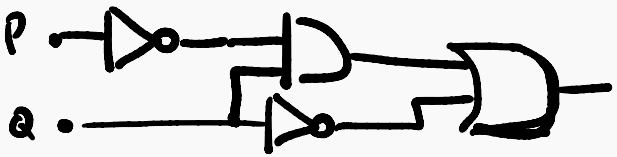
Boolean expression

Input signals: $P = 1$, $Q = 1$ and $R = 1$

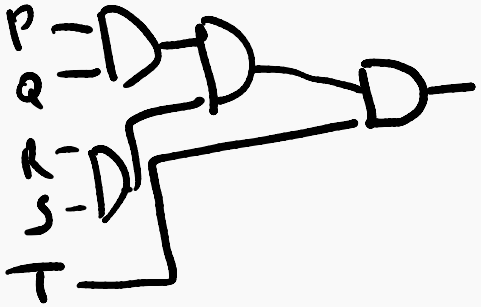


Constructing circuits for Boolean expressions

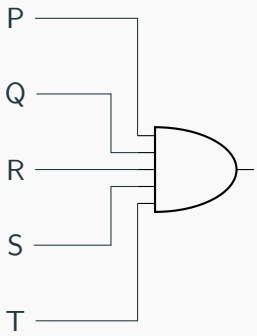
- $(\neg P \wedge Q) \vee \neg Q$

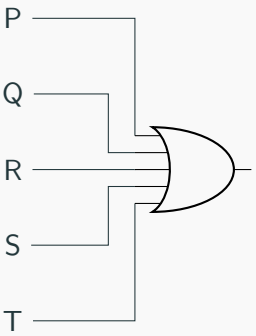


- $((P \wedge Q) \wedge (R \wedge S)) \wedge T$



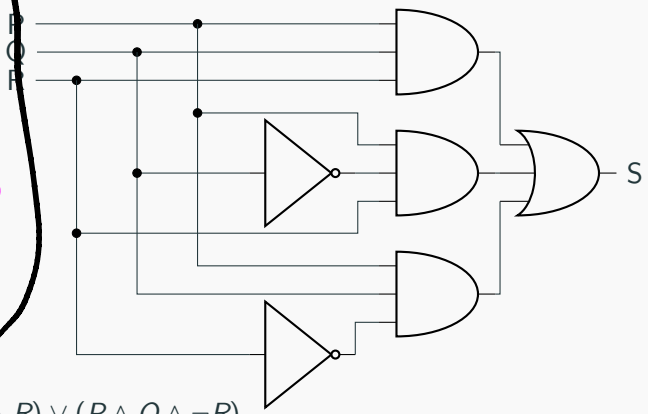
Multi-input AND and OR gates





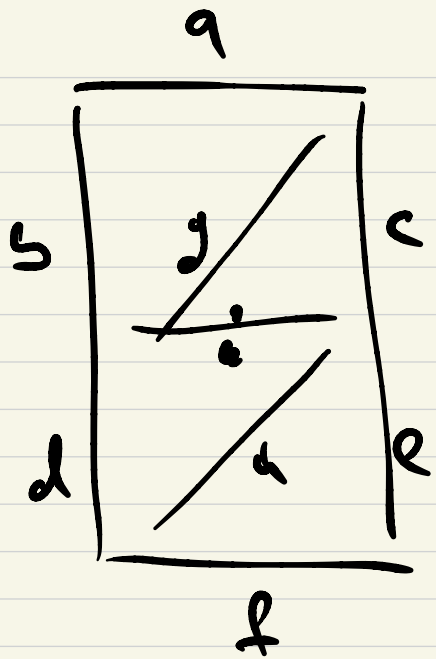
Designing a circuit for a given input/output table

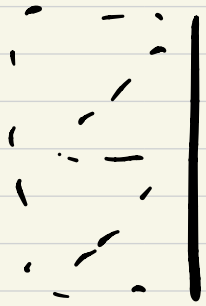
Input			Output
P	**Q**	**R**	**S**
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0



$(P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge \neg R)$

disjunctive normal form (DNF)





c = high

e = high

b = high

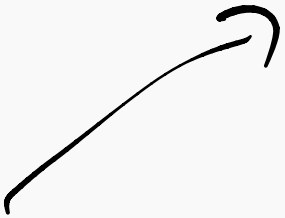
i = high

c = high

e = high

Another example

Input			Output
P	**Q**	**R**	**S**
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	0

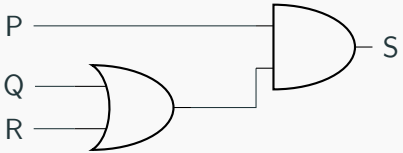


$(P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee$
$(\neg P \wedge Q \wedge R) \vee (\neg P \wedge \neg Q \wedge \neg R)$

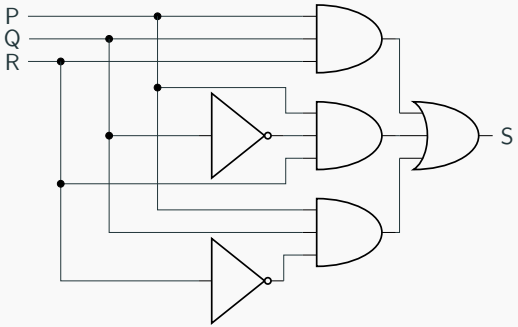
Reopen the first case

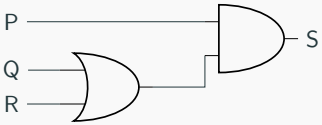
Input			Output
P	**Q**	**R**	**S**
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

$P \wedge (Q \vee R)$



Circuit equivalence





- Two digital circuits are equivalent if they produce the same output given the same inputs.

Logical equivalence

Definition Two formulas P and Q are called equivalent if they have the same truth value under every possible interpretation. In other words, P and Q are equivalent if $I(P) = I(Q)$ for every interpretation I. This is denoted by

$$P \equiv Q.$$

Example:

$$(P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge \neg R) \equiv P \wedge (Q \vee R)$$

On logical equivalence

Theorem The relation $\equiv$ is an equivalence relation on $\mathcal{P}$.

Proof

- $\equiv$ is reflexive, since, trivially, $I(P) = I(P)$ for every interpretation I.
- $\equiv$ is transitive, since $P \equiv Q$ and $Q \equiv R$ implies $P \equiv R$.
- $\equiv$ is symmetric, since $P \equiv Q$ implies $Q \equiv P$.

Simplifying propositional formulae

Exercises:

- $(P \Rightarrow Q) \equiv (\neg P \vee Q)$
- $\neg(P \Rightarrow Q) \equiv (P \wedge \neg Q)$
- $(P \Leftrightarrow Q) \equiv ((P \Rightarrow Q) \wedge (Q \Rightarrow P))$
- $(P \Leftrightarrow Q) \equiv (\neg P \Leftrightarrow \neg Q)$
- $(P \wedge (P \vee Q)) \equiv P$
- $\neg(P \vee (\neg P \wedge Q)) \equiv (\neg P \wedge \neg Q)$

Useful equivalences

The following equivalences can be checked by truth tables:

- Associative laws:

$$(P \vee (Q \vee R)) \equiv ((P \vee Q) \vee R),$$

$$(P \wedge (Q \wedge R)) \equiv ((P \wedge Q) \wedge R);$$

- Commutative laws:

$$(P \vee Q) \equiv (Q \vee P), \quad (P \wedge Q) \equiv (Q \wedge P);$$

- Identity laws:

$$(P \vee \bot) \equiv P, \ (P \vee \top) \equiv \top, \ (P \wedge \top) \equiv P, \ (P \wedge \bot) \equiv \bot;$$

- Distributive laws:

$$(P \wedge (Q \vee R)) \equiv ((P \wedge Q) \vee (P \wedge R))$$

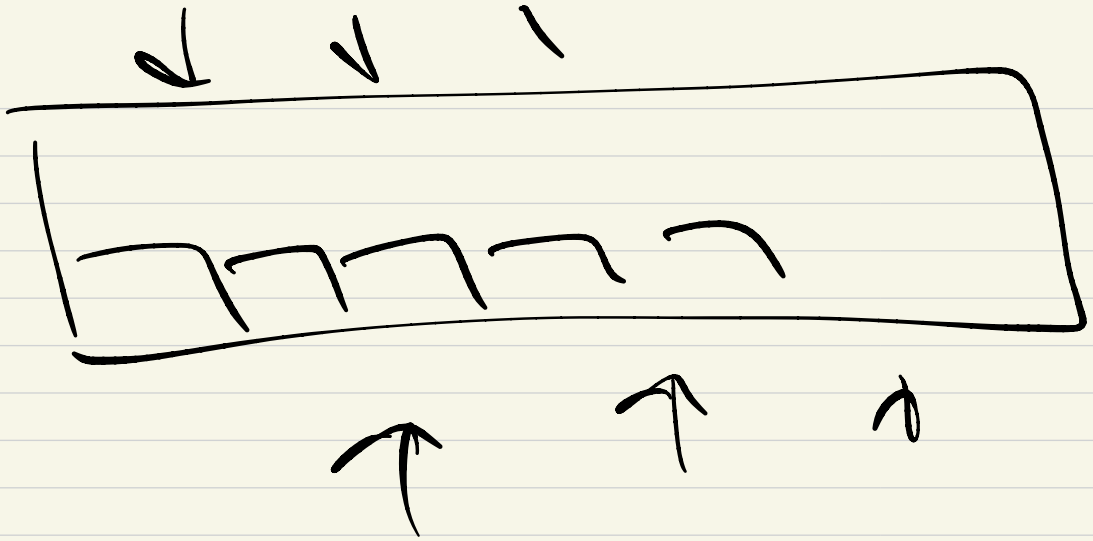
$$(P \vee (Q \wedge R)) \equiv ((P \vee Q) \wedge (P \vee R));$$

- Complement laws:

$$P \vee \neg P \equiv \top,\ \neg\top \equiv \bot,\ \neg\neg P \equiv P, P \wedge \neg P \equiv \bot,\ \neg\bot \equiv \top;$$

- De Morgan's laws:

$$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q),\ \neg(P \wedge Q) \equiv (\neg P \vee \neg Q).$$



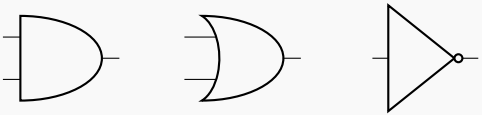
Boolean functions of arity 2

P	Q	$\bot$	$\wedge$	$\neg(P \Rightarrow Q)$	P	$\neg(Q \Rightarrow P)$	Q	$\neg(P \Leftrightarrow Q)$	$\vee$
1	1	0	1	0	1	0	1	0	1
1	0	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	0	0

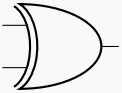
P	Q	$\neg(P \vee Q)$	$P \Leftrightarrow Q$	$\neg Q$	$Q \Rightarrow P$	$\neg P$	$P \Rightarrow Q$	$\neg(P \wedge Q)$	$\top$
1	1	0	1	0	1	0	1	0	1
1	0	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	1	1	1

Logic gates

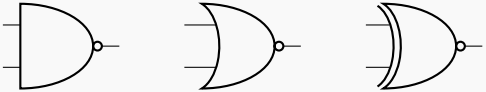
- AND, OR, NOT



- XOR



- NAND, NOR, XNOR



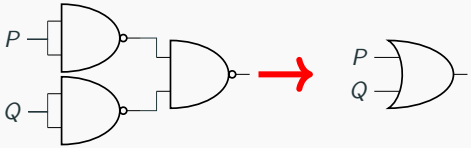
Universality of NAND and NOR

- NAND (AKA Sheffer stroke) $P \mid Q = \neg(P \wedge Q)$
- and NOR (AKA Pierce arrow) $P \downarrow Q = \neg(P \vee Q)$

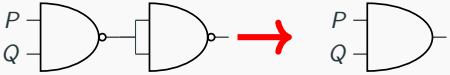
are universal:

$\neg P \equiv P \mid P$

$P \vee Q \equiv (P \mid P) \mid (Q \mid Q)$



$P \wedge Q \equiv (P \mid Q) \mid (P \mid Q)$

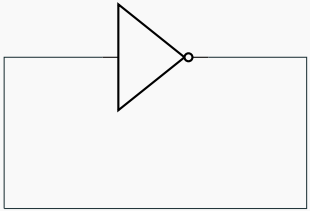


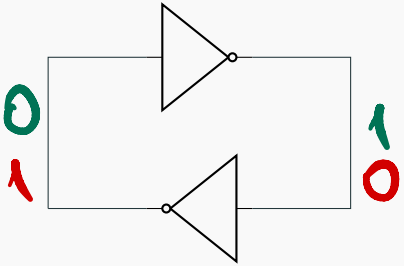
A bit on sequential circuits

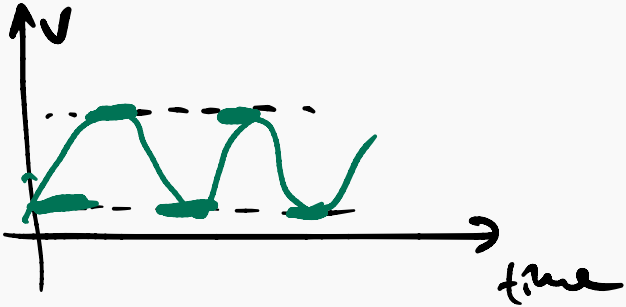
Rules for a sequential circuit

- Never combine two input wires.
- A single input wire can be split partway and used as input for two separate gates.
- An output wire can be used as input.
- An output of a gate **can** eventually feed back into that gate.

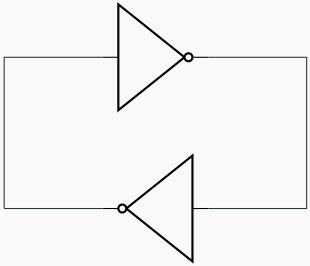
What happens here?



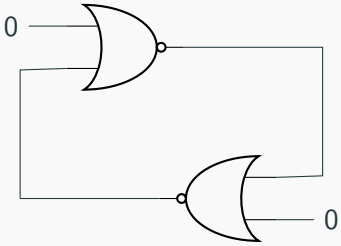




Same behaviour



is same as



NOR gate

Set/Reset flip-flop circuit

AKA latch