Logic Gates

Table of contents

The 7 basic gate types 1
More than two inputs 2
Universal (complete) gates 3
Gates on ICs 3

Every logic gate has one or more inputs and exactly one output, and can be described four equivalent ways: **logic symbol, truth table, Boolean expression, timing diagram** (see timing-diagrams[1] for the last of these, and for real-gate propagation delay / rise / fall times).

Gates sit at the bottom of the abstraction stack — the **digital logic level**, level 0 (see 2026-07-27-course-intro-bits-bytes-binary[2]). A digital circuit only ever carries **two** logic levels: logic '0' is usually a small voltage (around 0 V) and logic '1' a larger one (e.g. 0.8 V to 5 V, depending on the **logic family** — the type and size of the transistors the gate is built from). Because inputs and outputs have only these two states, they can equivalently be called *true* and *false*.

The 7 basic gate types

Gate	Behaviour	Truth table ($A, B \to X$)
NOT (inverter)	Inverts the input	`0→1`, `1→0`
AND	Output is HIGH when **all** inputs are HIGH	00→0, 01→0, 10→0, 11→1
OR	Output is HIGH when **at least one** input is HIGH	00→0, 01→1, 10→1, 11→1
NAND	Output is HIGH when **at least one** input is LOW	00→1, 01→1, 10→1, 11→0

[1] courses/csse2010/timing-diagrams.pdf
[2] courses/csse2010/2026-07-27-course-intro-bits-bytes-binary.pdf

Gate	Behaviour	Truth table ($A, B \to X$)
NOR	Output is HIGH when **all** inputs are LOW	00→1, 01→0, 10→0, 11→0
XOR	Output is HIGH when **exactly one** input is HIGH	00→0, 01→1, 10→1, 11→0
XNOR	Output is HIGH when the inputs are **the same**	00→1, 01→0, 10→0, 11→1

Useful to remember: **XOR is the odd function**, **XNOR is the even function**.

Deriving the inverting gates from the base gates

The quickest way to fill in a NAND/NOR/XNOR truth table is not to memorise it, but to write the underlying AND/OR/XOR column first and then invert it row by row — the "N" prefix is exactly one inversion on the output:

A	B	AND	**NAND**	OR	**NOR**	XOR	**XNOR**
0	0	0	**1**	0	**1**	0	**1**
0	1	0	**1**	1	**0**	1	**0**
1	0	0	**1**	1	**0**	1	**0**
1	1	1	**0**	1	**0**	0	**1**

More than two inputs

A gate of n inputs has a truth table of 2^n rows — two inputs give $2 \times 2 = 4$ rows, three inputs give 8. The "at least one / all" wordings above carry over unchanged to more inputs; the XOR/XNOR wordings do **not**, and must be restated in terms of parity:

- **XOR** output is HIGH when an **odd number** of inputs are HIGH (the *odd function*). For two inputs this collapses to "exactly one", which is why the 2-input wording works.
- **XNOR** output is HIGH when an **even number** of inputs are HIGH (the *even function*).

A	B	C	3-input AND	**3-input NAND**	**3-input XOR**
0	0	0	0	**1**	**0**
0	0	1	0	**1**	**1**
0	1	0	0	**1**	**1**
0	1	1	0	**1**	**0**
1	0	0	0	**1**	**1**

A	B	C	3-input AND	**3-input NAND**	**3-input XOR**
1	0	1	0	**1**	**0**
1	1	0	0	**1**	**0**
1	1	1	1	**0**	**1**

Note that the kit of parts used for practicals has **no** 3-or-more-input gate chips (see device-pinouts[3]) — a 3-input function has to be built from 2-input gates.

Universal (complete) gates

All circuits can be constructed from **NAND-only** or **NOR-only** gates — these are called **complete** (or universal) gates, because NOT, AND, and OR can each be built purely from one gate type:

- **NOT**: a NAND (or NOR) gate with both inputs tied together — $\overline{A \cdot A} = \bar{A}$, and $\overline{A + A} = \bar{A}$.
- **AND from NAND**: a NAND followed by a second NAND wired as an inverter. **AND from NOR**: invert both inputs first, then NOR them ($\overline{\bar{A} + \bar{B}} = AB$, by De Morgan — see boolean-algebra[4]).
- **OR from NOR**: a NOR followed by a NOR-as-inverter. **OR from NAND**: invert both inputs first, then NAND them ($\overline{\bar{A} \cdot \bar{B}} = A + B$).

The two rows are De Morgan's law drawn twice: whenever you push an inversion through a gate, AND and OR swap. NAND and NOR are preferred in practice because they are **easier to build from transistors** than AND/OR directly.

Gates on ICs

Gates are sold packaged several to an **integrated circuit** (IC). The **74HCT00** is four independent 2-input NAND gates in one 14-pin DIP package.

- **Vcc** = power (e.g. 5 V), **GND** = ground (0 V). One Vcc/GND pair supplies all four gates.
- Physically: pin spacing is 0.1" along the package and 0.3" across it; the chip is about 15 mm long.

[3] courses/csse2010/device-pinouts.pdf
[4] courses/csse2010/boolean-algebra.pdf

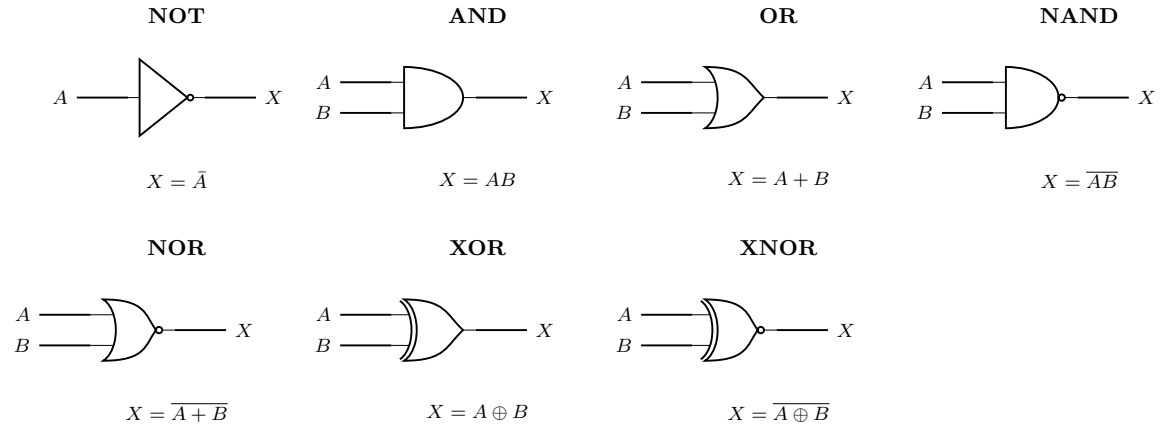

Figure 1: The seven basic gates, with the distinguishing symbol and Boolean expression for each. NOT, AND and OR are the three base shapes; NAND, NOR and XNOR are those same bodies with an inversion bubble on the output, and XOR/XNOR add a second arc across the back of the OR body.

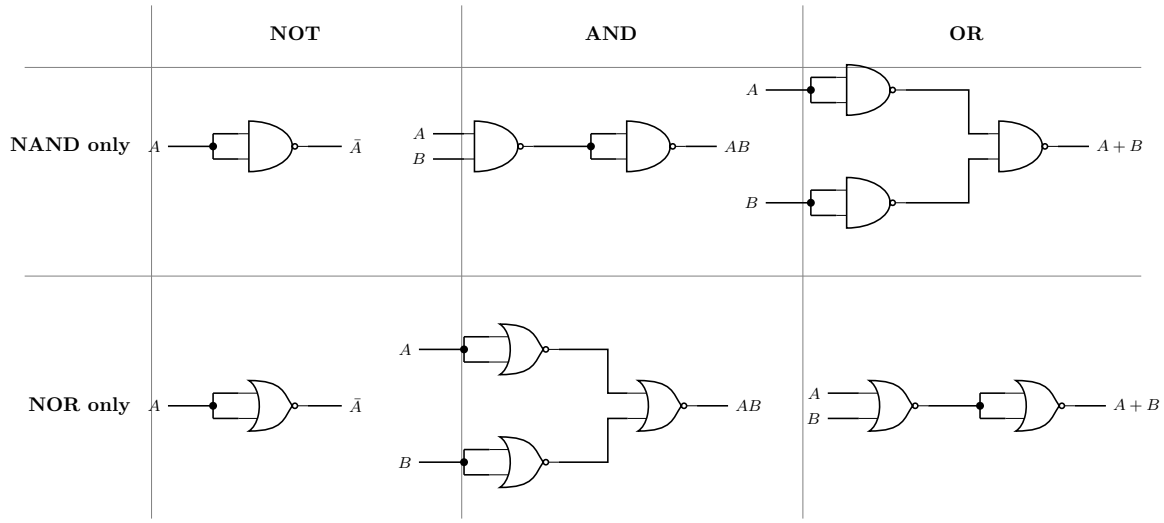

Figure 2: Why NAND and NOR are complete gates: NOT, AND and OR built from each one alone. Tying both inputs of a NAND or NOR together turns it into an inverter, and everything else follows — AND is NAND then invert, OR is invert-both-inputs then NAND (and the NOR row is the mirror image of that).

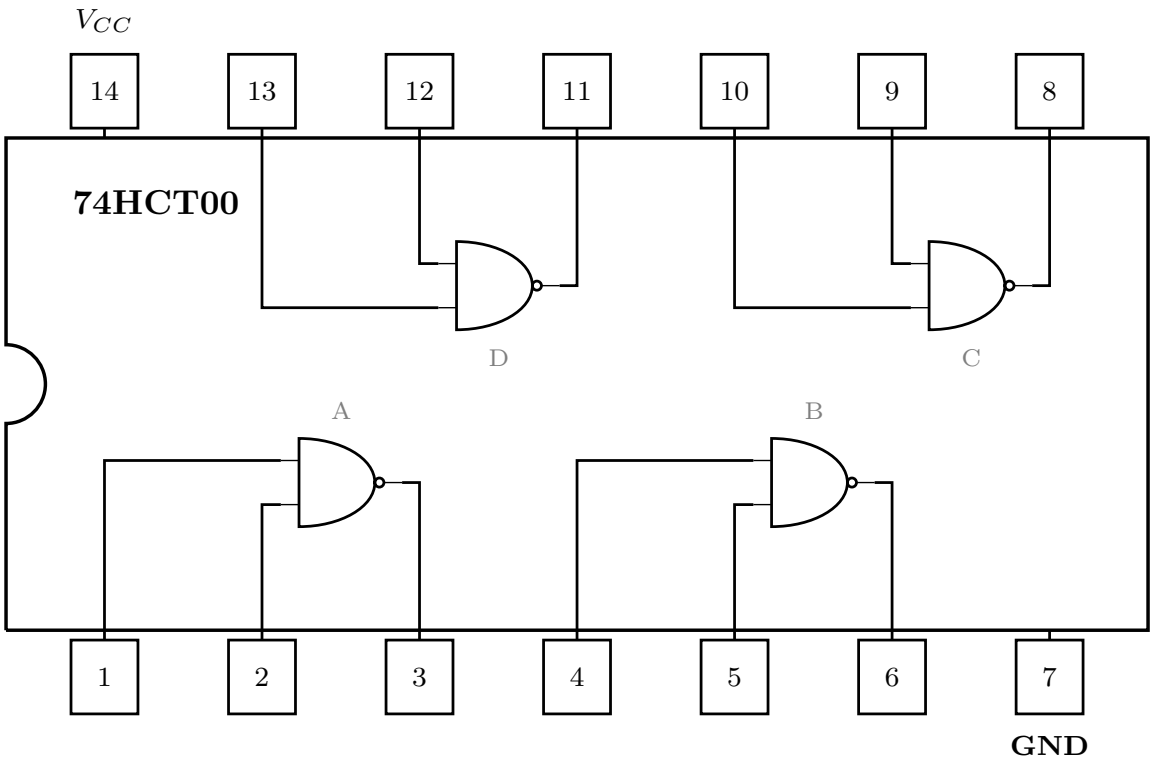

Figure 3: Inside the 74HCT00: four independent 2-input NAND gates in a 14-pin package, with the pin each gate terminal is wired to. Pin 14 is Vcc (power, e.g. 5 V) and pin 7 is GND (0 V) — shared by all four gates, so nothing works if either is left unconnected. Gate letters A-D are the labels a schematic uses as U1:A … U1:D.

- The drawing above is the chip's **internal schematic**, not a picture of the part — see circuit-schematics[5] for how a chip like this is drawn (and labelled `U1:A` … `U1:D`) on a schematic you can actually build from, and device-pinouts[6] for the pinouts of every chip in the course kit.

[5] courses/csse2010/circuit-schematics.pdf
[6] courses/csse2010/device-pinouts.pdf