Redstone circuits/Logic

< Redstone circuits
v t e

This article is about a specific category of redstone circuits. For other circuits, see redstone circuit.

A logic gate can be thought of as a simple device that returns a number of *outputs*, determined by the pattern of *inputs and rules* that the logic gate follows. For example, if both inputs in an AND gate are in the 'true'/'on'/'powered'/'1' state, then the gate returns 'true'/'on'/'powered'/'1'.

There are many different kinds of logic gates, each of which can be implemented with many different designs. Each design has various advantages and disadvantages, such as size, complexity, speed, maintenance overhead, or cost. The various sections describe different designs for each gate type.

Contents

Concepts

The output of each logic circuit reflects the state of its inputs at all times (though possibly with some delay incurred by the circuit).

Swapping inputs

For most of these gates, A and B can be swapped without changing the output.

Swapping the inputs of the IMPLIES gate *does* affect its output, and the NOT gate has only one input.

Stacking inputs

The AND, OR, and XOR gates can each be used in arrays to perform their operation on more than two inputs, by combining two inputs at a time, then combining the results with each other and/or other inputs. For these gates, the order in which the inputs are combined doesn't matter.

When an XOR gate is combined in this way, its output is on when an *odd* number of inputs is on.

Choosing a logic gate

When unsure which logic gate to use, try building a table like the one down below but with just one row of outputs. List the known inputs coming in and the possible combinations of power, and for each combination write down what the output should be for the structure to work. Then compare that to the table on the right and see which gate matches the desired outputs.

If the output needs to change when the input is stable, or needs to be remembered after the input has ended, the player may also need to look at pulse circuits or memory circuits.

Logic gate outputs
Shows the output (red) of each gate, for each combination of inputs A and B (green).

A	**On**	**On**	**Off**	**Off**	**Question Answered**
B	**On**	**Off**	**On**	**Off**	
A AND B	ON	Off	Off	Off	Are A and B on?
NOT (A IMPLIES B)	Off	ON	Off	Off	Is A on and B off?
NOT (B IMPLIES A)	Off	Off	ON	Off	Is B on and A off?
A NOR B	Off	Off	Off	ON	Are both inputs off?
A	ON	ON	Off	Off	Is A on?
A XOR B	Off	ON	ON	Off	Are the inputs different?
NOT A	Off	Off	ON	ON	Is A off?
A XNOR B	ON	Off	Off	ON	Are the inputs the same?
B	ON	Off	ON	Off	Is B on?
NOT B	Off	ON	Off	ON	Is B off?
A NAND B	Off	ON	ON	ON	Is either input off?
A IMPLIES B	ON	Off	ON	ON	If A is on, is B also on?
B IMPLIES A	ON	ON	Off	ON	If B is on, is A also on?
A OR B	ON	ON	ON	Off	Is either input on?

Logic gate

A **logic gate** is a basic logic circuit.

NOT gate

A **NOT gate** ($\overline{A}$), also known as an inverter, is a gate used when an opposite output is wanted from the input given. For instance, when the switch, or input, is set to "on", the output toggles to "off", and when the switch is toggled to "off", the output toggles to "on".

NOT gate outputs

A	On	Off
NOT A	Off	ON

Torch Inverter

1-wide, flat (horizontal only), silent, tileable
circuit delay: 1 tick

The torch inverter is the most commonly used NOT gate, due to its small size, versatility, and easy construction.

One drawback of the torch inverter is that it "burns out" if run on a clock cycle faster than a 3-clock (3 ticks on, 3 ticks off). A burnt out torch inverter turns on after it receives a block update.

Subtraction Inverter

flat, silent
circuit delay: 1 tick

The subtraction inverter offers little advantage over the torch inverter except that it can run on a 2-clock cycle without burning out. Faster clocks do not work though — the comparator simply doesn't react to them.

Variations: The powered lever can be replaced with another always-on power component (e.g., redstone torch, block of redstone), or with a full container if a power component would be inconvenient in that location.

The repeater is required to ensure the input signal is strong enough to overcome the comparator's rear source, but can be removed in a number of ways. If the input power level is known (because the circuit design is fixed, so it can be calculated), the repeater can be removed by replacing the powered lever with a container, which produces the same power level. Alternatively, the repeater can be removed if the output continues to a length of redstone wire that reduces the subtracted signal enough that the signal is inverted eventually.

Instant Inverter

instant
circuit delay: 0 ticks

The instant inverter is a basic building block of larger instant circuits.

The "ground" version has the largest volume, but is shorter and fits easily with flatter

circuits. The 1-wide version is the smaller in total volume and 2-tileable.

Behavior (i.e., how it works): An instant inverter has two functional elements, and a piston, or pistons, that activate them:

1. a constant power source with output that can be instantly powered off (but powering it on takes time): either a redstone block that ceases to provide power as soon as piston starts moving it (within the same tick the piston receives or loses power) or a solid block in front of a powered repeater or comparator, powering redstone dust; as soon as the block starts moving the dust is unpowered.
2. a signal line with output that can be instantly powered on but not necessarily off, its input delayed by and sustained for 2 ticks. The "instant on" is achieved by the dust-cut technique: a solid block placed against edge of a block over which a redstone line is running, disconnects that line from line below. Start of motion of that block instantly reconnects the line and provides power. The delay is achieved by running the input through a 2 tick repeater, two torches or similar means. That means, when power appears on input, the block moved by piston is able to cut the line before signal passes through the delay. With input unpowered, the output is instantly activated and the line still provides power "stored" in the repeater for two ticks, which time is sufficient to reactivate the constant power source from previous point.
3. Piston, or pistons, to move the block/blocks activating the elements from point 1 or 2.

Schematic gallery: NOT gate

OR gate

An **OR gate** ($A \vee B$) is a gate that uses two or more inputs and whenever any input is "on", the output is also "on". The only time the output is "off" is when all inputs are "off". Note that since the OR operation is associative and commutative, OR gates can be combined freely: The player can compare huge numbers of inputs by using small OR gates to collect groups of inputs, then comparing their results with more OR gates. The result does not depend on the arrangement of the inputs, or on which ones were combined first.

OR gate outputs

A	**On**	**On**	**Off**	**Off**
B	**On**	**Off**	**On**	**Off**
A OR B	ON	ON	ON	Off

The simplest version of the OR gate is design **A**: merely a wire connecting all inputs and outputs. However, this causes the inputs to become "compromised", so that they can be used only in this OR gate. The introduction's example, using a solid block instead of wire, does not suffer the same hazard.

If players need to use the inputs elsewhere, the inputs need to be "isolated", by passing them through a block as above, or a device such as a torch or repeater. Torches yield version **B**. Note that this is in fact a NOR gate with an inverter on the output.

Version **C** isolates the inputs with repeaters. It can be expanded horizontally up to 15 inputs. Besides the isolated inputs, it is one tick faster than **B**. Additional repeaters can be used to add new groups of inputs, or to strengthen the output signal. This design is more expensive, as each

repeater costs 3 redstone dust to craft (along with smooth stone).

Version **D** is a 1-wide version designed for vertical use, such as in walls. The repeater serves to isolate the outputs from the inputs. This version can take only two inputs, though of course the inputs can be stacked with multiple gates.

Version **E** utilizes the properties of light-transparent blocks: glowstone, and upside-down stairs or slabs. These send signals up, but not down. It is expandable, like design **C**.

Schematic gallery: OR gate

NOR gate

A **NOR gate** ($A \downarrow B$ or $\overline{A \vee B}$) is the opposite of the OR gate. Whenever at least one switch is toggled to "on", the output is toggled to "off". The only time the output is "on" is when all inputs are toggled to "off". This gate also uses two or more inputs.

NOR gate outputs

A	**On**	**On**	**Off**	**Off**
B	**On**	**Off**	**On**	**Off**
A NOR B	Off	Off	Off	ON

All logic gates can be made from some combinations of the NOR gate.

In *Minecraft*, NOR is a basic logic gate, implemented by a torch with two or more inputs. (A torch with 1 input is the NOT gate, and with no inputs is the TRUE gate, that is, a power source.)

A torch can easily accommodate 3 mutually isolated inputs, as in design **A**. Design **B** goes to greater lengths to squeeze in a fourth input. If more inputs are necessary, it is simplest to use OR gates to combine them, then use an inverter (NOT) at the end. It is also possible to combine OR and NOR gates, by using the inversion of OR gates as inputs to NOR gates.

For inverted output when A is OFF, use redstone torch for B and result is:

NOR gate outputs with inverted B

A	**On**	**On**	**Off**	**Off**
B	**On**	**Off**	**On**	**Off**
A NOR B	Off	Off	Off	ON

Schematic gallery: NOR gate

AND gate

An **AND gate** ($A \wedge B$) is used with two or more switches or other inputs. The output is toggled to "on" only when all inputs are "on". Otherwise, the output remains "off".

In reality, the usual implementation is a NOR gate with inverted inputs ($\overline{\overline{A} \vee \overline{B}}$). Taking the inputs A and B, the first two torches (at the top and bottom of Schematic (A) below) invert them into $\overline{A}$ and $\overline{B}$. The redstone wire between these torches serves as an OR gate, and is therefore in state $\overline{A} \vee \overline{B}$, which can be interpreted as $\overline{A \wedge B}$ by De Morgan's Law. Finally, the third torch (the center-right one) applies a NOT to that statement; thus it becomes $\overline{\overline{A} \vee \overline{B}} = \overline{\overline{A \wedge B}} = A \wedge B$.

AND gate outputs

A	On	On	Off	Off
B	On	Off	On	Off
A AND B	ON	Off	Off	Off

A 3-input AND gate is shown, but, like OR gates, AND gates can be freely "ganged", combining groups of inputs and then combining the results.

A typical use for an AND gate would be to build a locking mechanism for a door, requiring both the activating button and the lock (typically a lever) to be on.

Piston AND gates act similarly to a "tri-state buffer", in which input B acts like a switch, connecting or disconnecting input A from the rest of the circuit. Such designs have one input feeding a circuit, which is opened or closed by a sticky piston driven by the other input. The difference from real-life tri-state buffers is that one cannot drive a low current in Minecraft.

Schematic gallery: AND gate

NAND gate

A **NAND gate** ($A \uparrow B$ or $\overline{A \wedge B}$) turns the output off only when both inputs are on, the reverse of an AND gate. All logic gates can be made from NAND gates. As with NOR, large numbers of inputs are probably best handled by stacking up AND gates, then inverting the result. By De Morgan's Law, $\overline{A \wedge B}$ is identical to $\overline{A} + \overline{B}$.

NAND gate outputs

A	On	On	Off	Off
B	On	Off	On	Off
A NAND B	Off	ON	ON	ON

All logic gates can be made from some combinations of the NAND gate.

Schematic gallery: NAND gate

XOR gate

An **XOR gate** ($A \veebar B$, $A \nleftrightarrow B$ or $A \oplus B$) is a gate that uses two inputs and the output is toggled to "on" when one switch is "on" and one switch is "off". XOR is pronounced "zor" or "exor", a shortening of "exclusive or", because each input is mutually exclusive with the output. It is useful for controlling a mechanism from multiple locations. Because of these properties, XOR gates are commonly found in complex redstone circuits. In some cases, it is possible to get an OR gate output and an AND gate output on different channels. Design F is composed of AND gates,

XOR gate outputs

A	On	On	Off	Off
B	On	Off	On	Off
A XOR B	Off	ON	ON	Off

OR gates and NOT gates. The whole circuit is $\overline{(A \wedge B) \vee \overline{A}} \vee \overline{(A \wedge B) \vee \overline{B}}$, which can be further simplified into $(\overline{A} \wedge B) \vee (A \wedge \overline{B})$ (or, equivalently, $(A \vee B) \wedge \overline{A \wedge B}$).

A useful feature is that an XOR (or XNOR) gate always changes its output when one of its inputs changes, hence it is useful for controlling a mechanism from multiple locations. When controls (such as levers) are combined in an XOR gate, toggling any control toggles the XOR gate's output (like a light bulb controlled by two light switches — players can flip *either* one to turn the light on or off, or *either* of which can always open or close a door, or turn some other device on or off.

Like AND and OR gates, XOR gates can freely be "stacked" together, with gates gathering groups of inputs and their outputs being gathered in turn. The result of XORing more than two inputs is called "parity" — the result is 1 if and only if an odd number of inputs are 1.

Design **D** is tiny, but useful if players want the levers to be fixed to the circuit. The shaded block indicates the block the levers and the lit torch are attached to, along with the block that one is resting on.

Design **F** is the most widely used of the torch-only designs, but newer components can do much better. Design **H** uses pistons, and is both faster and more compact.

Beyond torches and pistons, various diodes can be used to produce fairly compact and cheap XOR gates. Design **I** can have its input repeaters coming in from either side or underneath, changing its size accordingly to fit tight spaces. Design **J** uses transparent blocks for a cheaper option.

Schematic gallery: XOR gate

The introduction of the comparator allows for several variations of a new design, the "subtraction XOR gate", which is flat, fast and silent (also easy to remember). The cons in Survival mode is that making comparators requires the access to the Nether to obtain nether quartz.

Each input is the same distance to the rear and side of the comparator closest to it, suppressing its own signal there, but travels farther to get to the side of the further comparator, so doesn't suppress its signal in the further comparator. Only if both inputs are on do both comparators get suppressed by a side input.

However, that is true only if the inputs are the same power level (or at least not different by more than 1), otherwise one signal could overwhelm the other's attempt to suppress its signal. If this circuit is sure to receive inputs of the same power level (because the system it's part of has been designed that way), then the "basic" version can be used. Otherwise, some method should be used to ensure the inputs are equal — for example, with repeaters (the "repeated" version) or with torches (the "inverted" version).

Schematic gallery: subtraction XOR gate

XNOR gate

XNOR gate outputs

A	On	On	Off	Off
B	On	Off	On	Off
A XNOR B	ON	Off	Off	ON

An **XNOR gate** ($A \leftrightarrow B$ or $A \odot B$) is the opposite of an XOR gate. This is commonly referred to as "if and only if" ("iff" [*sic (https://en.wikipedia.org/wiki/Sic)*] for short), "bi-conditional", or "equivalence". It uses two inputs. When both switches are in the same state (both switches are "on" or both switches are "off"), then the output is toggled to "on". Otherwise, if the switches differ, the output is toggled to "off". Similar to the XOR gate, when either input changes, the output changes.

An XNOR gate can be built by inverting either the output, or *one* input, of an XOR gate.

Design **A** is a pure-torch design. If external input isn't needed, the back-facing torches can be replaced with levers, yielding **B**. Design **F** is larger but highlights the logic, while **I** is an inverted variant of XOR gate **H**. Note that the output inverter can also be placed in line with the rest of the gate, or even in a pit attached to one of the output redstone's support blocks.

Schematic gallery: XNOR gate

IMPLY gate

IMPLY gate outputs

A	On	On	Off	Off
B	On	Off	On	Off
A IMPLIES B	ON	Off	ON	ON

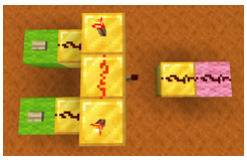
An **IMPLY gate** ($A \rightarrow B$) turns on either if both inputs are on, or if the first input is off. Unlike the other gates here, the inputs are not interchangeable; it is not commutative. This represents material implication or a conditional statement, "if *A* then *B*", or "*A* implies *B*". The output is off only if the antecedent *A* is true, but the consequent *B* is false. It is the logical equivalent of B ∨ ¬A, and the mathematical equivalent of A ≤ B.

Design **C** has a speed of 2 ticks if output is 1, but 1 tick if the output is 0. Similarly, the other designs take 1 tick if the output is 0, but are immediate (and not isolated) if the output is 1. If the player must synchronize (or isolate) the output, consider placing a 1-tick repeater in front of the "fast" input (input A for **C**, input B for the others).

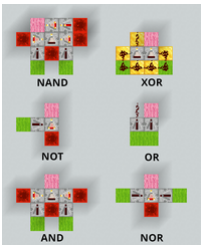
Note if the IMPLY gate powers a block that needs to be powered again to be activated (e.g. dropper, dispenser) then either, A has to be on, or A and B, as it is already on. The logic may opposingly apply to NIMPLY gates.

Schematic gallery: IMPLY gate

Gallery



AND gate example.



Some Compact Redstone Logic Circuits.

Videos

See also

- Logic gates on Wikipedia

Navigation

Redstone [hide]

Redstone circuits & tutorials [hide]

Redstone basics Redstone components · Redstone mechanics · Conductivity

Redstone circuits Clock circuits · Logic circuits · Memory circuits · Miscellaneous circuits · Pulse circuits · Transmission circuits

Featured tutorials Advanced redstone circuits · Block update detector (BUD) · Comparator update detector (CUD) · Flying machine · Mechanisms · Piston circuits · Quasi-connectivity · Redstone music

Redstone components [show]

Tutorials [hide]

Introductory [show]

General [show]

Challenges [show]

Constructions [show]

Farming [show]

Mechanisms [hide]

Various devices Redstone (Circuits · Components · Dust · Mechanics · Tips) · Automatic respawn anchor recharger · Combination locks · Command block · Crafter · Flying machines · Hopper · TNT · Item sorting · Item transportation · Mechanisms · Observer stabilizer · Randomizers · Redstone music (Doorbell) · Rube Goldberg machine · Shulker box storage

Detectors Block update detector · Comparator update detector · Daylight detector

Minecarts Train station · Minecarts (Storage · Storage system)

Traps Snow golems · TNT cannons · Trapdoor uses · Trap design · Traps

Pistons Piston uses · Piston circuits · Quasi-Connectivity · Zero-ticking · Instant repeaters

Tripwire Tripwire techniques

Advanced redstone Advanced redstone circuits · Arithmetic logic · Calculator · Command stats · Hourly clock · Morse code · Printer · Redstone computers · Redstone telegraph · 360 degree ender pearl cannon

Multiplayer [show]

Technical [show]

Outdated [show]