NAND logic

From Wikipedia, the free encyclopedia

Because the NAND function has functional completeness all logic systems can be converted into NAND gates. This is also true for NOR gates. In principle, any combinatorial logic function can be realized with enough NAND gates.

Contents

NAND

A NAND gate is an inverted AND gate. It has the following truth table:

Q = NOT(**A** AND **B**)

Truth Table

Input A	Input B		Output Q
0	0		1
0	1		1
1	0		1
1	1		0

Making other gates by using NAND gates

A NAND gate is a universal gate, meaning that any other gate can be represented as a combination of NAND gates.

NOT

A NOT gate is made by joining the inputs of a NAND gate together. Since a NAND gate is equivalent to an AND gate followed by a NOT gate, joining the inputs of a NAND gate leaves only the NOT gate.

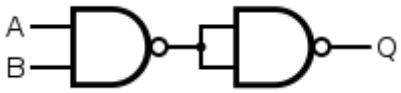
Desired NOT Gate	**NAND Construction**
	
Q = NOT(**A**)	= **A** NAND **A**

Truth Table

Input A		**Output Q**
0		1
1		0

AND

An AND gate is made by following a NAND gate with a NOT gate as shown below. This gives a NOT NAND, i.e. AND.

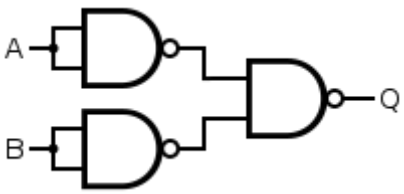
Desired AND Gate	**NAND Construction**
	
Q = **A** AND **B**	= (**A** NAND **B**) NAND (**A** NAND **B**)

Truth Table

Input A	**Input B**		**Output Q**
0	0		0
0	1		0
1	0		0
1	1		1

OR

If the truth table for a NAND gate is examined or by applying De Morgan's Laws, it can be seen that if any of the inputs are 0, then the output will be 1. To be an OR gate, however, the output must be 1 if any input is 1. Therefore, if the inputs are inverted, any high input will trigger a high output.

Desired OR Gate	**NAND Construction**
	
Q = **A** OR **B**	= (**A** NAND **A**) NAND (**B** NAND **B**)

Truth Table

Input A	**Input B**		**Output Q**
0	0		0

0	1		1
1	0		1
1	1		1

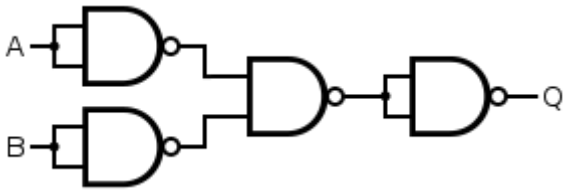
NOR

A NOR gate is simply an inverted OR gate. Output is high when neither input A nor input B is high.

Desired NOR Gate

Q = **A** NOR **B**

NAND Construction



= [(**A** NAND **A**) NAND (**B** NAND **B**)]
NAND
[(**A** NAND **A**) NAND (**B** NAND **B**)]

Truth Table

Input A	**Input B**		**Output Q**
0	0		1
0	1		0
1	0		0
1	1		0

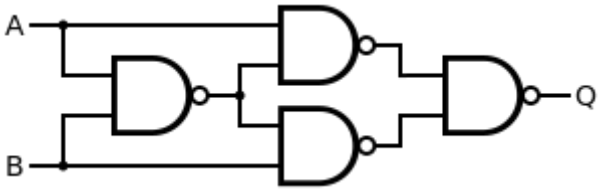
XOR

An XOR gate is constructed similarly to an OR gate, except with an additional NAND gate inserted such that if both inputs are high, the inputs to the final NAND gate will also be high, and the output will be low. This construction has a propagation delay three times that of a single NAND gate and uses four gates.

Desired XOR Gate

Q = **A** XOR **B**

NAND Construction



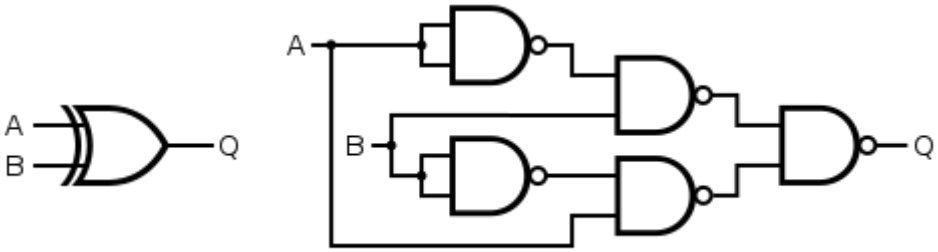
= [**A** NAND (**A** NAND **B**)] NAND
[**B** NAND (**A** NAND **B**)]

Truth Table

Input A	**Input B**		**Output Q**
0	0		0
0	1		1
1	0		1
1	1		0

Alternatively, the B-input of the XNOR gate with the 3-gate propagation delay can be inverted. This construction uses five gates instead of four.

Desired Gate **NAND Construction**



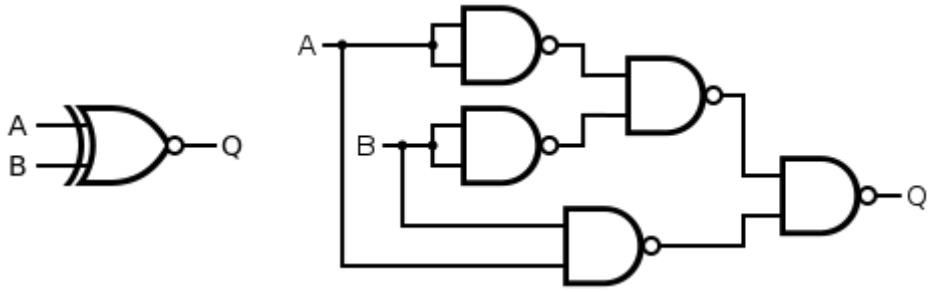
Q = **A** XOR **B**

= [**B** NAND (**A** NAND **A**)] NAND
[**A** NAND (**B** NAND **B**)]

XNOR

An XNOR gate is made by connecting the output of 3 NAND gates (connected as an OR gate) and the output of a NAND gate to the respective inputs of a NAND gate. This construction entails a propagation delay three times that of a single NAND gate and uses five gates.

Desired XNOR Gate **NAND Construction**



Q = **A** XNOR **B**

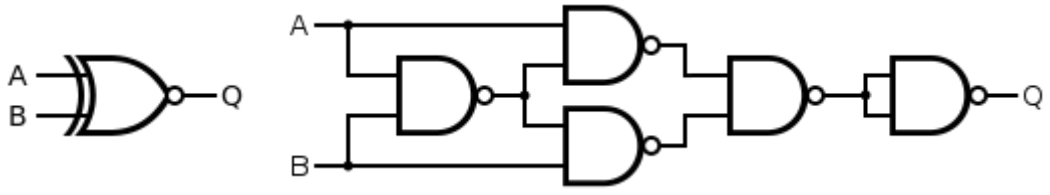
= [(**A** NAND **A**) NAND (**B** NAND **B**)]
NAND
(**A** NAND **B**)

Truth Table

Input A	Input B		Output Q
0	0		1
0	1		0
1	0		0
1	1		1

Alternatively, the 4-gate version of the XOR gate can be used with an inverter. This construction has a propagation delay four times (instead of three times) that of a single NAND gate.

Desired Gate **NAND Construction**



Q = **A** XNOR **B**

= { [**A** NAND (**A** NAND **B**)] NAND
[**B** NAND (**A** NAND **B**)] } NAND

{ [**A** NAND (**A** NAND **B**)]
NAND [**B** NAND (**A** NAND **B**)] }

MUX

A multiplexer or a MUX gate is a three-input gate that uses one of the inputs, called "selection bits", to select and output one of the other two inputs, called "data bits".[1]

Desired MUX Gate	NAND Construction
Q = [**A** AND NOT (**S**)] OR (**B** AND **S**)	= NOT{ NOT[**A** AND NOT(**S** AND **S**)] AND NOT(**B** AND **S**) }

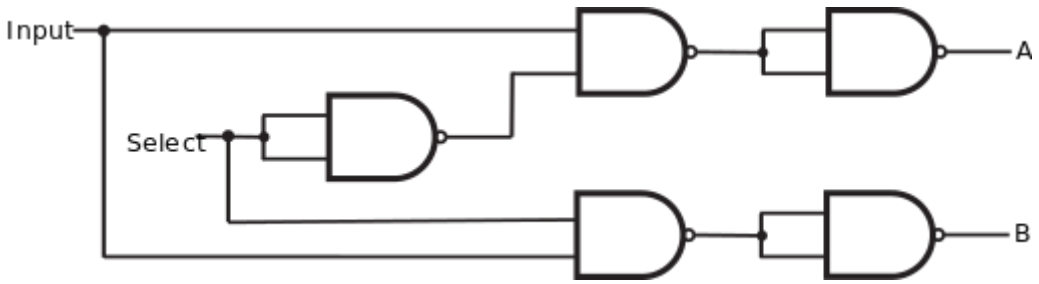
Truth Table

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

DEMUX

A demultiplexer performs the opposite function of a multiplexer: It takes a single input and channels it to one of two possible outputs according to a selector bit that specifies which output to choose.[1]

Desired DEMUX Gate	NAND Construction



Truth Table

Select		A	B
0		in	0
1		0	in

See also

- NOR logic. Like NAND gates, NOR gates are also universal gates.

- Functional Completeness

External links

- TTL NAND and AND gates (http://www.allaboutcircuits.com/vol_4/chpt_3/5.html) - All About Circuits
- Steps to Derive XOR from NAND gate. (http://www.fullchipdesign.com/drive_xor_nand_interview.htm)
- NAND Gate (http://teahlab.com/nand_gate/), Demonstrate an interactive simulation of the NAND Gate circuit created with Teahlab's simulator.

References

1. Nisan, N. & Schocken, S., 2005. In: From NAND to Tetris: Building a Modern Computer from First Principles. s.l.:The MIT Press, p. 20. Available at: http://www.nand2tetris.org/chapters/chapter%2001.pdf

Lancaster, Don (1974). *TTL Cookbook* (1st ed.). Indianapolis, IN: Howard W Sams. pp. 126–135. ISBN 0-672-21035-5.

Retrieved from "https://en.wikipedia.org/w/index.php?title=NAND_logic&oldid=800811813"

- This page was last edited on 15 September 2017, at 20:36.
- Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.