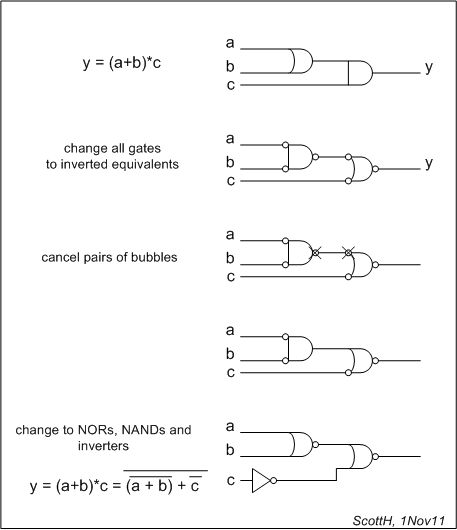
**Computer Architecture**

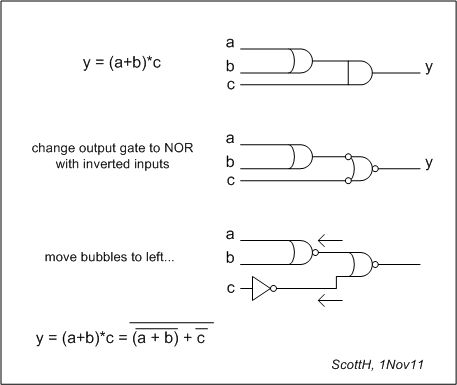
**NAND logic**Using just NAND gates, all other logic gates can be constructed. This means the NAND gate is functionally complete. A NOR gate is also functionally complete, but are easier and cheaper to make, so are used more often.

One way to convert a logic gate into its logically equivalent NAND construction, is to draw out the truth table, get it into conjunctive normal form (ANDs of ORs) and then simplify down. Since it will be comprised of NOT, OR and AND (a functionally complete set), the individual gates can then be converted to NANDs. However there are often simpler versions. Alternatively a formula can be converted into NAND form (¬A + ¬B) using rules such as de Morgen.

Can also convert from a circuit made up of positive logic (ORs and ANDs) by playing “the bubble game”. This uses de Morgen’s laws to convert to NAND and NOR. There are 4 “rules”:

* You can change ANDs or ORs to (N)ANDs and (N)ORs with bubbles on all terminals.
* You can "push" a bubble from the output back to the inputs, making them all inverted.
* You can "push" bubbles from all inputs through to the output, inverting the output.
* Two bubbles on a line cancel.

Example:

Or alternatively, if only the output gate is changed…

|  |  |  |  |
| --- | --- | --- | --- |
| **Gate** | **Truth table** | **NAND equivalent circuit** | **NOT, AND equivalent formula** |
| NAND ANSI Labelled.svgNAND | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | | - | - |
| NOT  NOT ANSI Labelled.svg | |  |  | | --- | --- | | **A** | **Q** | | 0 | 1 | | 1 | 0 | | NOT from NAND.svg  A NAND A | ¬(A.A) |
| AND  AND ANSI Labelled.svg | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | AND from NAND.svg  (A NAND B) NAND (A NAND B) | ¬(  ¬(A.B) . ¬(A.B)  ) |
| OR  OR ANSI Labelled.svg | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | | OR from NAND.svg(A NAND A) NAND (B NAND B) | ¬(  ¬(A.A) . ¬(B.B)  ) |
| NOR  NOR ANSI Labelled.svg | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 | | NOR from NAND.svg((A N A) N (B N B))N((A N A) N (B N B)) | ¬{  ¬[¬(A.A).¬(B.B)].  ¬[¬(A.A).¬(B.B)]} |
| XOR  XOR ANSI Labelled.svg | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | | XOR from NAND.svg(A N (A N B)) N (B N (A N B)) | ¬{  ¬[ A . ¬(A.B)]  .  ¬[B . ¬(A.B)]  } |
| XNOR  XNOR ANSI Labelled.svg | |  |  |  | | --- | --- | --- | | **A** | **B** | **Q** | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | (A N (A N B)) N (B N (A N B))XNOR from NAND.svg  N  (A N (A N B)) N (B N (A N B)) | ¬(  ¬{  ¬[ A . ¬(A.B)]  .  ¬[B . ¬(A.B)]  } . ¬{  ¬[ A . ¬(A.B)]  .  ¬[B . ¬(A.B)]  }  ) |
| https://upload.wikimedia.org/wikipedia/commons/thumb/3/39/Multiplexer_2-to-1.svg/175px-Multiplexer_2-to-1.svg.png2-1 Multiplexer | |  |  |  |  | | --- | --- | --- | --- | | **Io** | **I1** | **A** | **Q** | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 | | 2-to-1 multiplexer circuit  (Io N A) N ( (A N A) N I1) | ¬(  ¬(Io . A)  .  (¬ (¬A . A) . I1)  ) |
| http://sub.allaboutcircuits.com/images/04465.png1-2 Demultiplexer | |  |  |  | | --- | --- | --- | | **Sel** | **D0** | **D1** | | 0 | I | 0 | | 1 | 0 | I |  |  |  |  |  | | --- | --- | --- | --- | | **I** | **A** | **D0** | **D1** | | 0 | 0 | 0 | 0 | | 0 | 1 | 0 | 1 | | 1 | 0 | 0 | 0 | | 1 | 1 | 1 | 0 | | DEMUX GateA/D0 = (I N (S N S)) N ( (I N (S N S))  B/D1 = (I N S) N (I N S) | A/D0 = ¬(¬(I.¬(S.S))  . (I.¬S.S))  B/D1 = ¬(¬(I.S).¬(IS)) |