

Notes: Combinatorial Logic Circuits

A. Why?

Combinatorial logic circuits correspond to pure functions on boolean values (functions without a changeable internal memory state), so they're ubiquitous in logical and arithmetic calculations and decision-making.

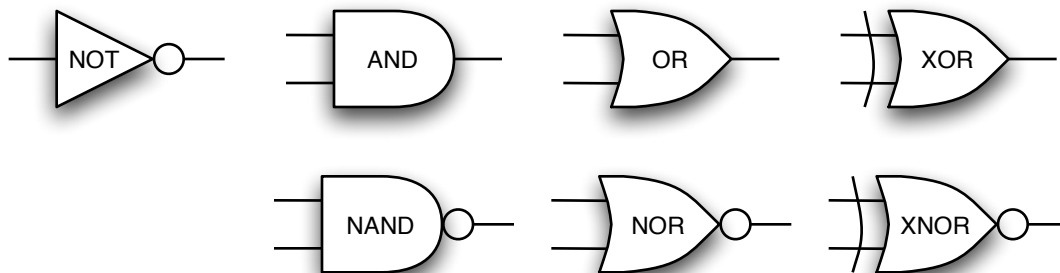
B. Outcomes

By the end of the class you should

- Know the symbols for the usual gates
- Know what a combinatorial logic circuit is.
- Know what decoders and multiplexers (muxes) do (and that they're combinatorial logic circuits).
- Know what half adders and full adders do.

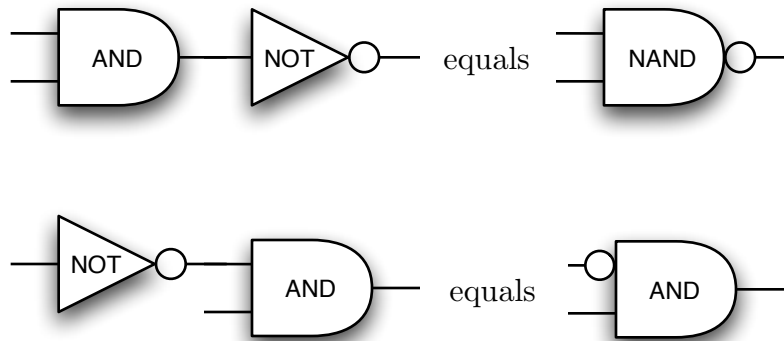
C. Gate Diagrams

- Symbols for gates:



- Note little circle indicates negation.
- XNOR (“eks-nore”) is logical equivalence (i.e., iff).
- Can have AND/NAND, OR/NOR gates with > 2 inputs: Do n-way AND or OR, negate if necessary.
- Straightforward to combine gates to calculate simple boolean expressions:
 - Examples:
 - $P \text{ OR } \text{NOT } Q$
 - $(\text{NOT } P \text{ OR } \text{NOT } Q) \text{ AND } (P \text{ OR } Q)$
 - Path through graph gets longer when the expression gets more deeply nested.
 - Example: $P \text{ AND } (Q \text{ OR } (R \text{ AND } (S \text{ OR } T)))$

- Abbreviate NOT gates



D. Truth Tables to Logic Gates

- Any truth table with n inputs and m outputs can be implemented using logic gates.
 - (To make sure the circuit is combinatorial, the inputs and outputs must be distinct — no input is also an output.)
- Treat each output separately:
 - Each input row of a table corresponds to the AND of the inputs (some perhaps NOTted)
 - The output is the OR of ANDs where the output rows are 1.
 - Example below: $R = \text{NOT } P \text{ AND } Q \text{ OR } P \text{ AND } Q$; $S = \text{NOT } P \text{ AND NOT } Q \text{ OR NOT } P \text{ AND } Q \text{ OR } P \text{ AND NOT } Q$

P	Q	Corresponding Expression	Output R	Output S
0	0	NOT P AND NOT Q	0	1
0	1	NOT P AND Q	1	1
1	0	P AND NOT Q	0	1
1	1	P AND Q	1	0

- Resulting expression can be implemented using AND, NOT, and OR.
 - Might exist implementations with fewer gates.
- Alternative notation for AND, OR, NOT: juxtapositioning, plus sign, overbar
 - E.g. NOT P OR NOT Q is written $\overline{P} + \overline{Q}$; NOT P AND NOT Q is written $\overline{P} \overline{Q}$, and NOT (P AND Q) is written \overline{PQ} .

E. Logic Gates to Truth Table

- A logic gate diagram without loops can be translated into a boolean function by working bottom-up from the inputs to the outputs.

Ended here

F. Some Standard Circuits

- Decoder: n -bit input (binary number), 2^n outputs (named $00\dots0$ to $11\dots1$); the one output specified by the input will be 1, the others will be 0.
- Multiplexer (“mux”) works the other way: 2^n data inputs (named $00\dots0$ to $11\dots1$) and an n -bit “select signal” input. The one output equals the named input.
- Half adder: Consider adding 1 bit + 1 bit to get a 2-bit output ($0+0 = 00$, $0+1 = 1+0 = 01$, $1+1 = 10$); the left bit is the carry out.
- Full adder: For unsigned addition of two binary numbers, there’s also a carry in to worry about ($0+0+0 = 00$, $0+0+1 = 0+1+0 = 1+0+0 = 01$, $0+1+1 = 1+0+1 = 1+1+0 = 10$, $1+1+1 = 11$).
 - (A half adder is a full adder with carry in = 0.)
 - To add two n -bit numbers, you need n copies of a full adder.