Boolean Algebra – Axioms

Boolean Algebra

- Boolean Algebra:
  - algebra over 2 elements: {0,1}
  - 3 operators: AND (∧), OR (∨), NOT (¬)
- AND Operation:
  - a ∧ b = 1 iff a=1 and b=1
- OR Operation:
  - a ∨ b = 1 if a=1 or b=1
- NOT Operation:
  - ¬a = 1 iff a=0

a	b	a&b	a|b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

a	~a
0	1

Axioms of Boolean Algebra

- *Axioms: a set of mathematical statements that we assert to be true.*
- Identity:
  - $1 \wedge x = x$
  - $0 \vee x = x$
- Annihilation:
  - $0 \wedge x = 0$
  - $1 \vee x = 1$
- Negation:
  - $\neg 0 = 1$
  - $\neg 1 = 0$

Boolean Algebra - Theorems

Useful Boolean Properties
(all can be derived from the axioms)

Communicative	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
Associative	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \vee (y \vee z) = (x \vee y) \vee z$
Distributive	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
Idempotence	$x \wedge x = x$	$x \vee x = x$
Complementation	$x \wedge \neg x = 0$	$x \vee \neg x = 1$

Useful Boolean Properties
(all can be derived from the axioms)

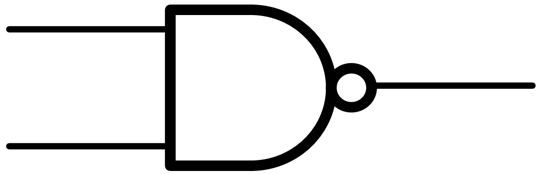
Absorption	$x \wedge (x \vee y) = x$	$x \vee (x \wedge y) = x$
Combining	$(x \wedge y) \vee (x \wedge \neg y) = x$	$(x \vee y) \wedge (x \vee \neg y) = x$
DeMorgan's	$\neg(x \wedge y) = \neg x \vee \neg y$	$\neg(x \vee y) = \neg x \wedge \neg y$
Consensus	$(x \wedge y) \vee (\neg x \wedge z) \vee (y \wedge z)) = (x \wedge y) \vee (\neg x \wedge z)$	$(x \vee y) \wedge (\neg x \vee z) \wedge (y \vee z)) = (x \vee y) \wedge (\neg x \vee z)$

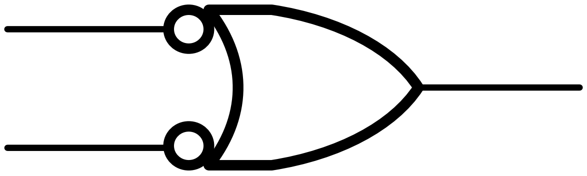
DeMorgan's Law

- $\neg(x \wedge y) = \neg x \vee \neg y$ $\neg(x \vee y) = \neg x \wedge \neg y$
- Proof by perfect induction

x	y	$\neg(x \wedge y)$	$\neg x \vee \neg y$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

DeMorgan Graphically





Applying Boolean Properties

$f(a,b,c) = (a \wedge c) \vee (a \wedge b \wedge c) \vee (\neg a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c)$

$$\begin{aligned} f(a,b,c) = {} & (a \wedge c) \vee (a \wedge b \wedge c) \vee \\ & (\neg a \wedge b \wedge c) \vee (a \wedge b \wedge c) \vee \\ & (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c) \end{aligned}$$

← *Apply Absorption Property*

$$\begin{aligned} f(a,b,c) = {} & (a \wedge c) \vee \\ & (\neg a \wedge b \wedge c) \vee (a \wedge b \wedge c) \vee \\ & (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c) \end{aligned}$$

← *Apply Combining Property*

$f(a,b,c) = (a \wedge c) \vee (b \wedge c) \vee (a \wedge b)$

Boolean Algebra – Dual function

Dual Functions

$$f(a,b) = (a \wedge b) \vee (b \wedge c), \tag{3.7}$$

then

$$f^D(a,b) = (a \vee b) \wedge (b \vee c). \tag{3.8}$$

A very useful property of duals is that the dual of a function applied to the complement of the input variables equals the complement of the function. That is:

$$f^D(\bar{a}, \bar{b}, \ldots) = \overline{f(a, b, \ldots)}.$$

Example: Finding the dual of a function

Give the dual of the following un-simplified function:

$$f(x, y) = (1 \wedge x) \vee (0 \vee \neg y)$$

Summary

- Learned the basics of Boolean algebra.
  - Axioms (identity, annihilation, negation)
  - Properties (commutative, associative ... DeMorgan's theorem)
  - Theorems (consensus, absorption, uniting ...)
- Duality

Boolean Algebra – Normal forms

Normal Form

Simplified Sum-of-Products form

$f(a, b, c) = (a \wedge c) \vee (b \wedge c) \vee (a \wedge b)$

Canonical (standard) Sum-of-Products form

$f(a, b, c) =$

Shannon expansion

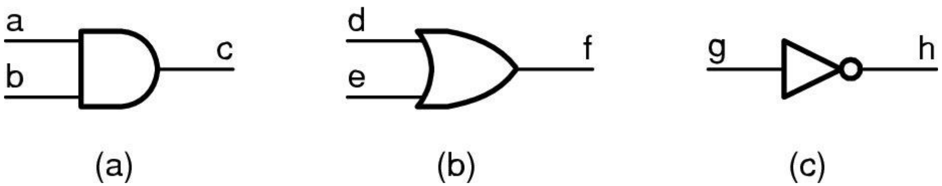
Example

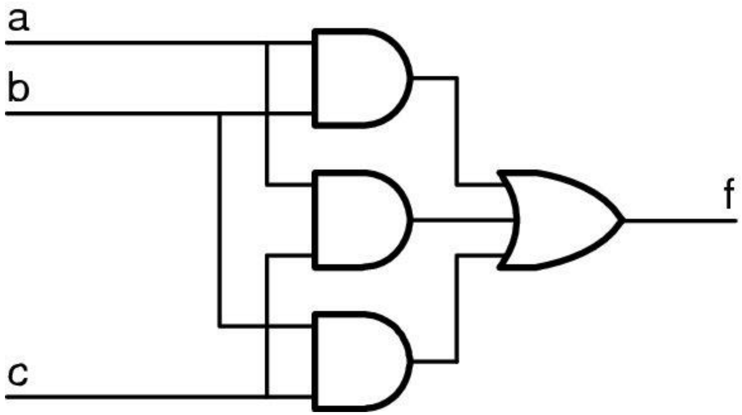
Write the following equation in normal form:

$$f(a, b, c) = a \vee (b \wedge c)$$

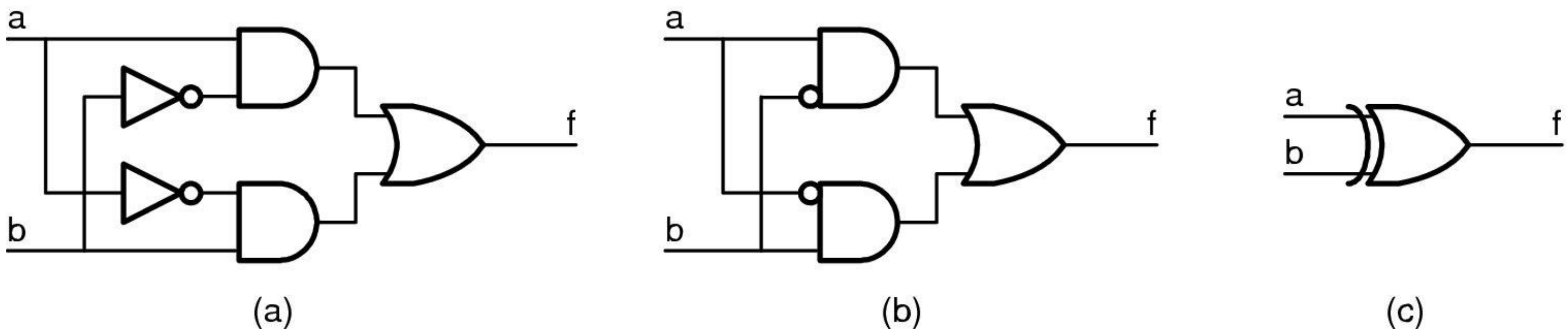
Boolean Algebra – Logic diagram

From Equations to Gates

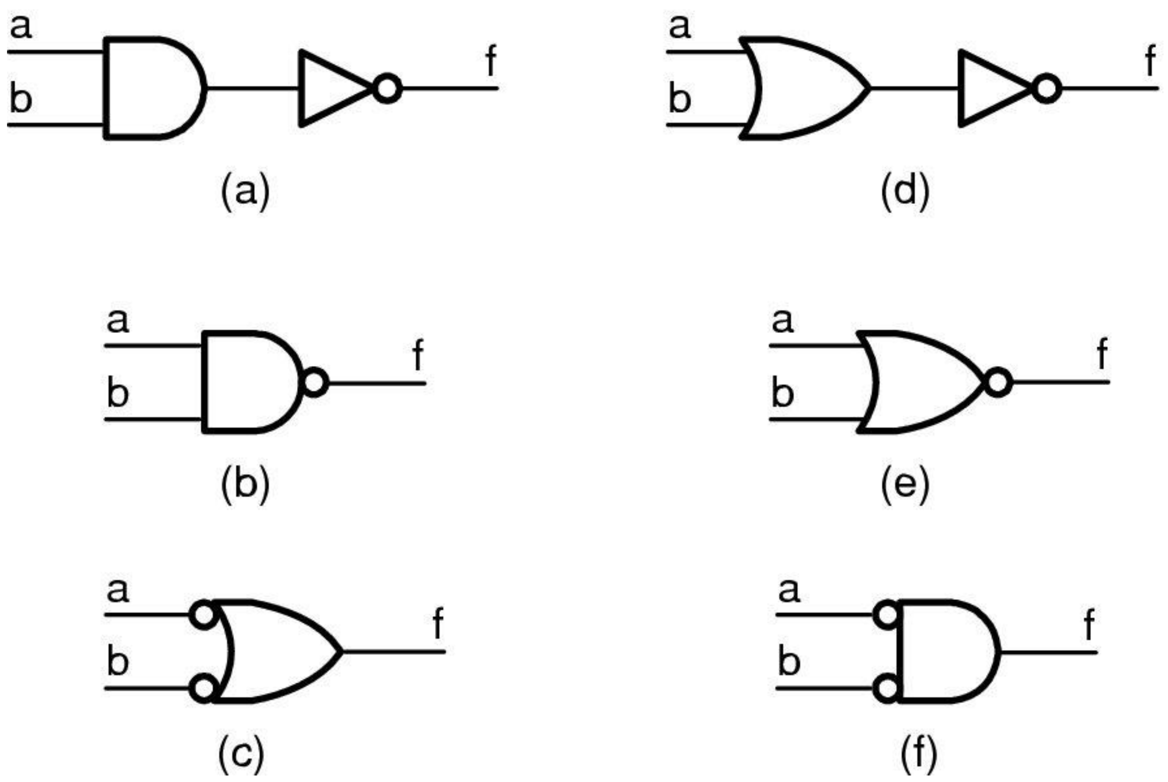




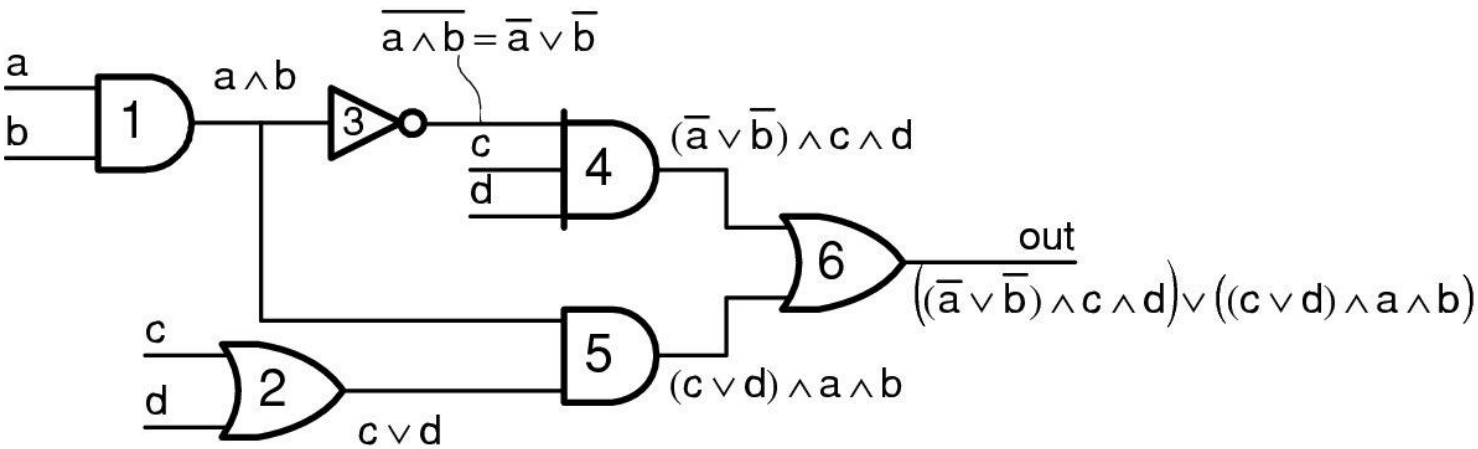
Exclusive-or function



NAND and NOR gates



From Gates to Equations



Example: From equation to schematic

Draw the schematic for a three-input majority function that only uses NAND gates:

Boolean Algebra – Verilog expression

Boolean Expressions in Verilog

```
module Majority(a, b, c, out) ;
  input a, b, c ;
  output out ;

  wire out ;

  assign out = (a & b)|(a & c)|(b & c) ;
endmodule
```

```
module test ;
  reg [2:0] count ;      // input - three bit counter
  wire out ;             // output of majority

  // instantiate the gate
  Majority m(count[0],count[1],count[2],out) ;

  // generate all eight input patterns
  initial begin
    count = 3'b000 ;
    repeat (8) begin
      #100
      $display("in = %b, out = %b",count,out) ;
      count = count + 3'b001 ;
    end
  end
endmodule
```

```
in = 000, out = 0
in = 001, out = 0
in = 010, out = 0
in = 011, out = 1
in = 100, out = 0
in = 101, out = 1
in = 110, out = 1
in = 111, out = 1
```

Summary

- Normal forms
- Logic diagram
- Verilog using `assign` statement