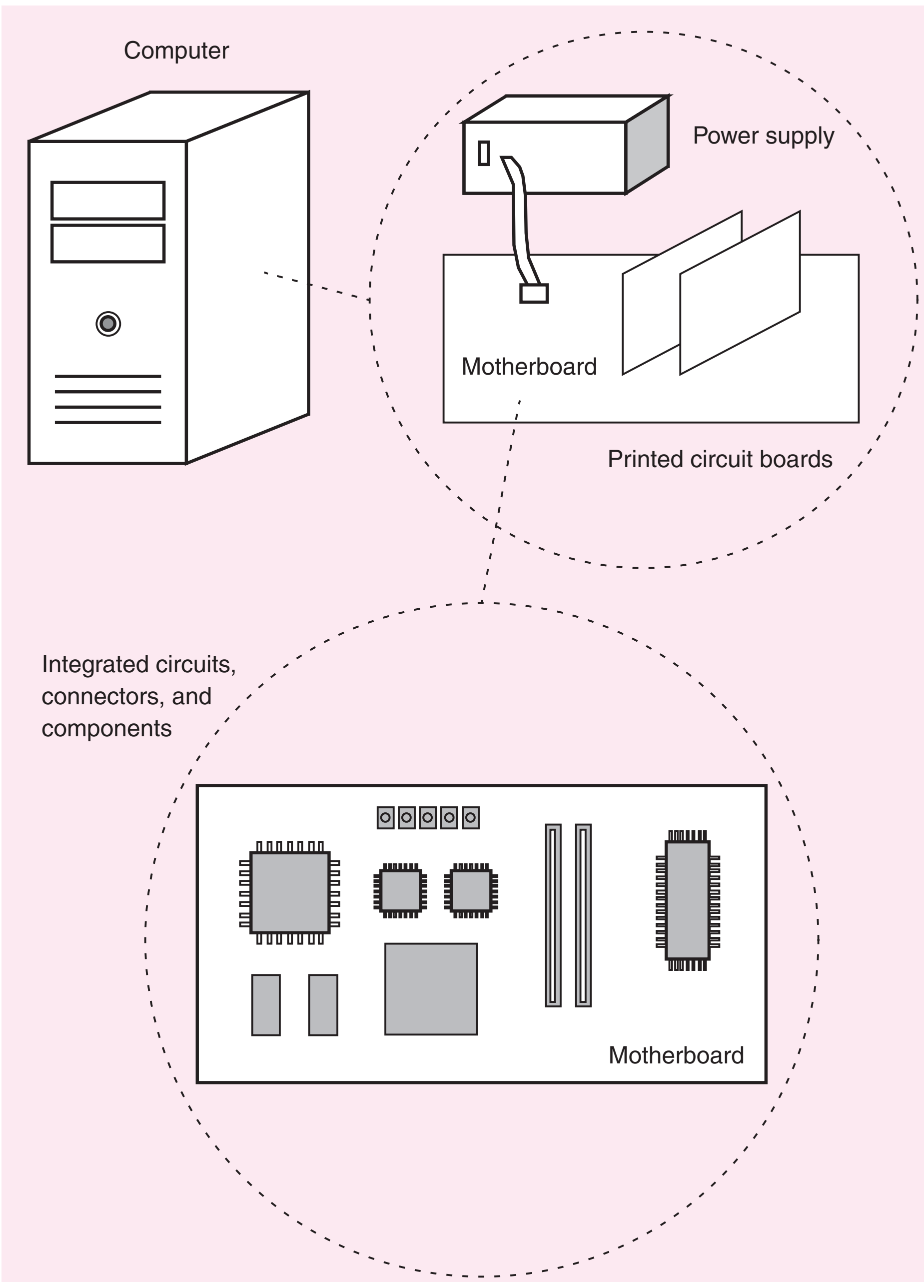
ECE/CS 3700
Digital System Design

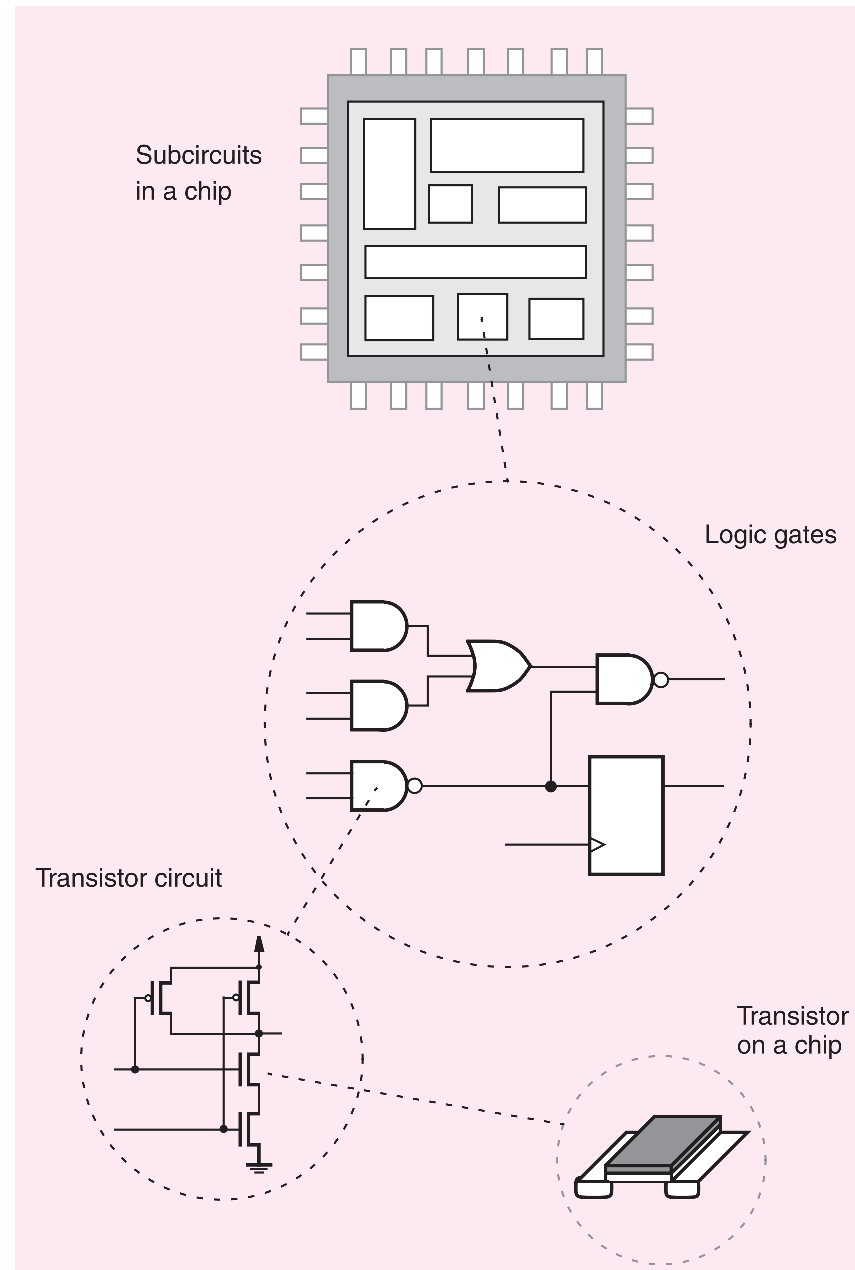
Lecture Slides for Chapters 1 & 2

THE UNIVERSITY OF UTAH

Priyank Kalla
Professor
Electrical & Computer Engineering

From Circuit to Logic and System Design





Learn Logic Design Fundamentals, as well as Modern Computer-Aided Design

- Logic Design with Boolean Algebra
- Hardware Description Languages (HDL)
  - We will study Verilog-HDL
  - Other HDLs exist: VHDL, others extensions
- Use of CAD tools

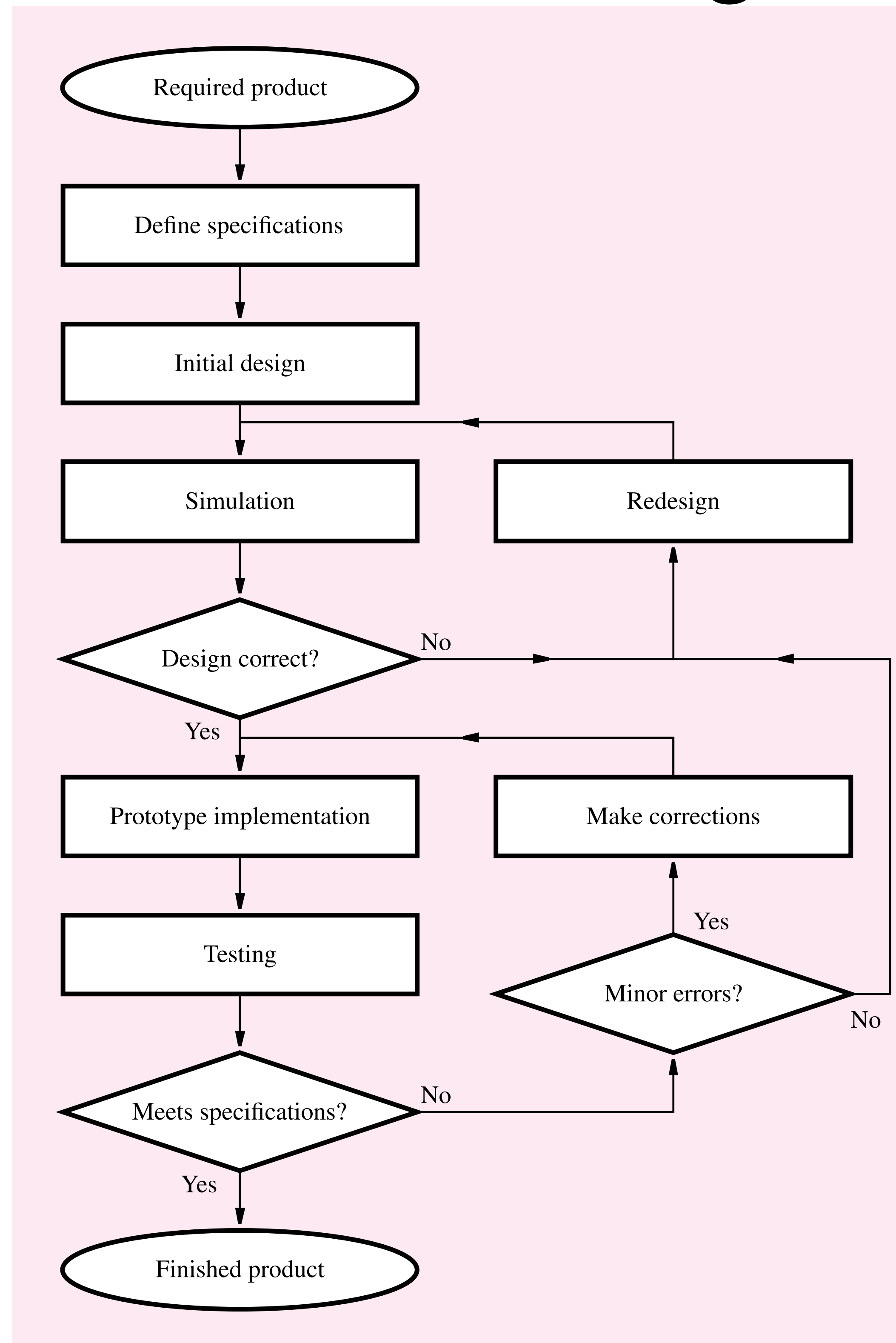
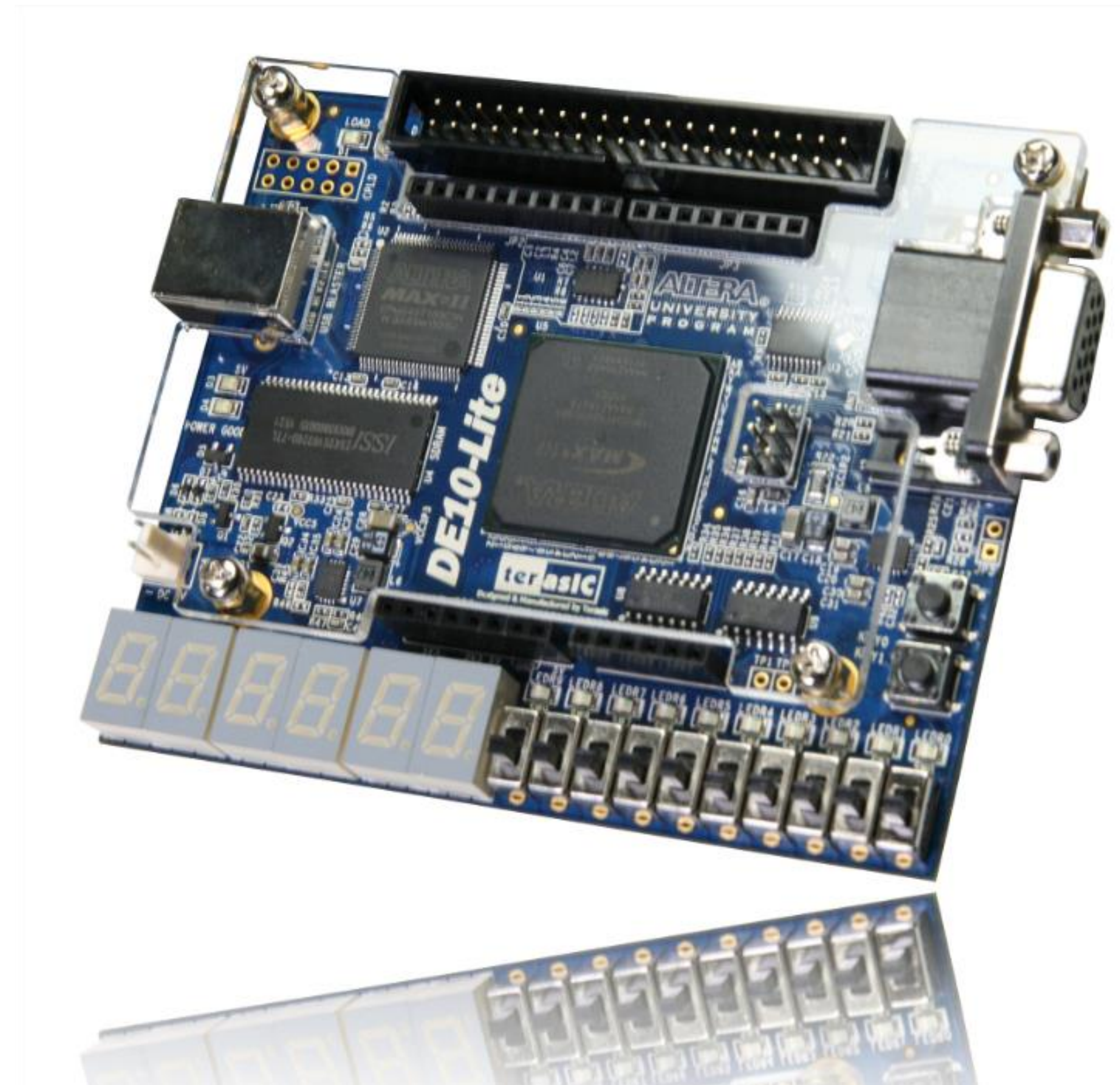


Figure 1.3 The development process.

Field Programmable Gate Array (FPGA)

- Learn how to design logic circuits
- Design in Verilog, Simulate, validate correctness
- Synthesize the circuit and implement on an FPGA
- FPGA = reconfigurable hardware, excellent for prototyping
- 6 or 7 Lab assignments

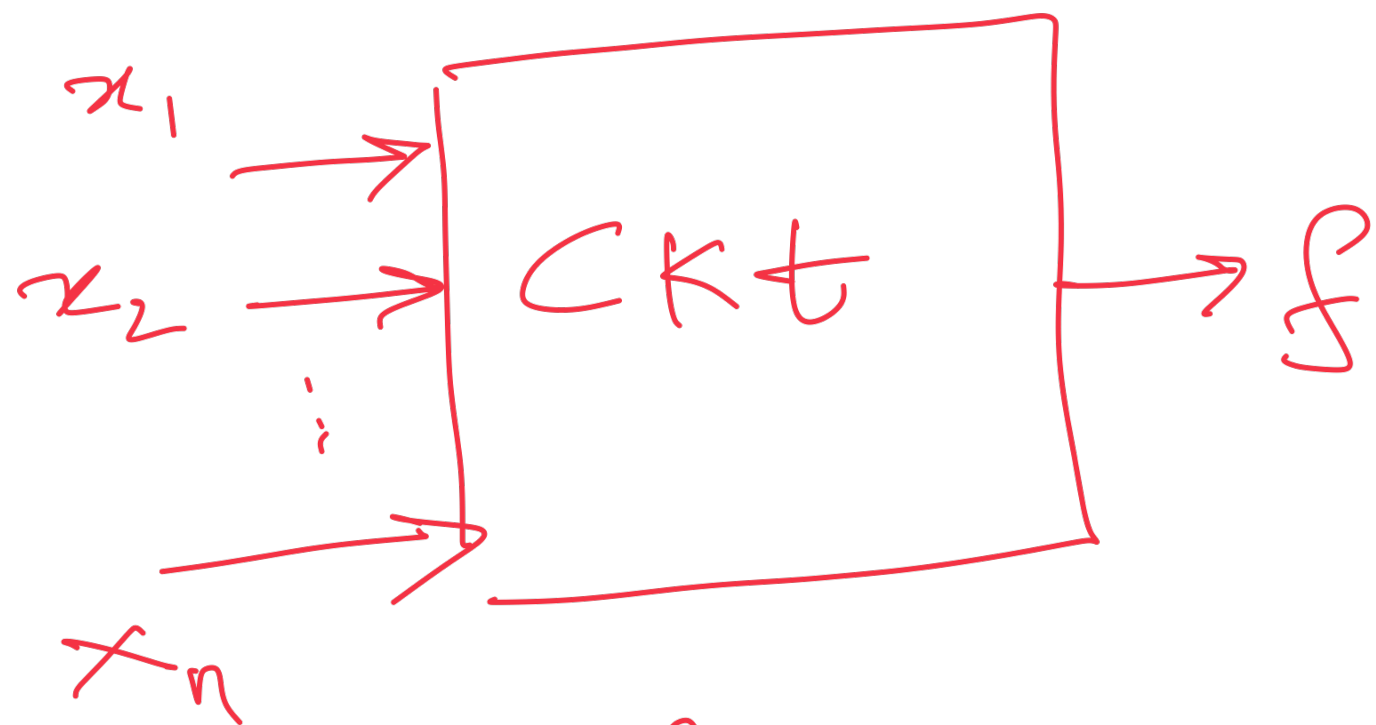


Chapter 2: Intro to Logic Circuits

- Boolean Algebra fundamentals
- Boolean functions and logical operations
- Boolean logic gates and circuits
- Logic Synthesis: Describe Boolean functions in the form of truth tables, and synthesize a logic circuit from it
- Perform logic optimization to reduce “cost of circuit implementation” — area, speed/delay, power, etc.

Boolean Algebra and Functions

- Algebra = set and operations on the elements of the set
- $\mathbb{B} = \{0,1\}$ is the Boolean domain
- Boolean function: $\mathbb{B}^n \to \mathbb{B}$
- Logic circuits implement Boolean functions

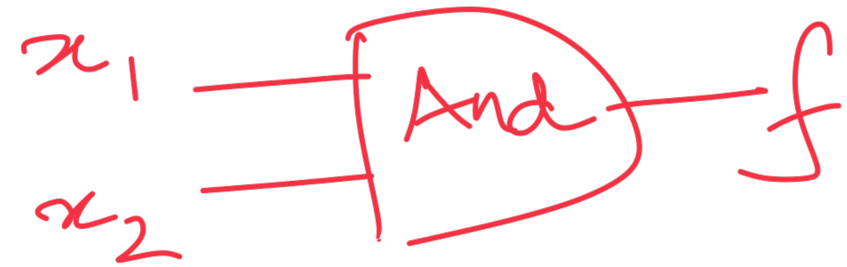


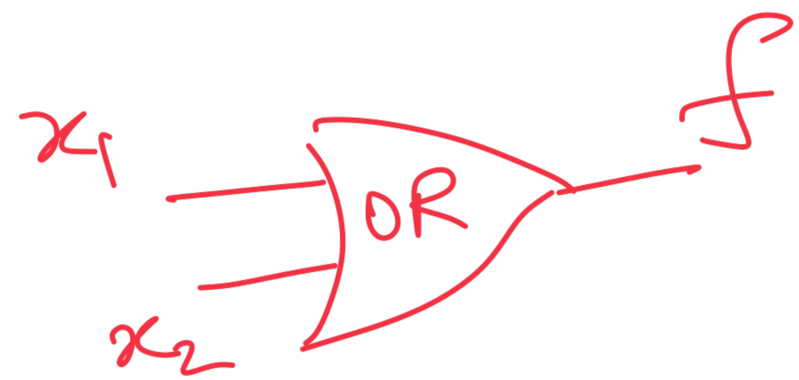
$f : B^n \to B$

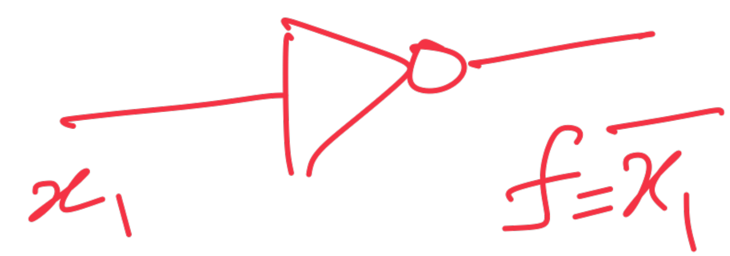
$f = 0$ or 1

Boolean functions: Truth Tables

- Simple Boolean functions: AND and OR functions
- Set complement: NOT function
- $AND : f = x_1 \cdot x_2 = x_1 \wedge x_2$
- $OR : f = x_1 + x_2 = x_1 \vee x_2$
- NOT: $f = \neg x_1 = x_1' = \overline{x_1}$







$0 \rightarrow 1$

$1 \rightarrow 0$

x_1	x_2	$x_1 \cdot x_2$	$x_1 + x_2$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
		AND	OR

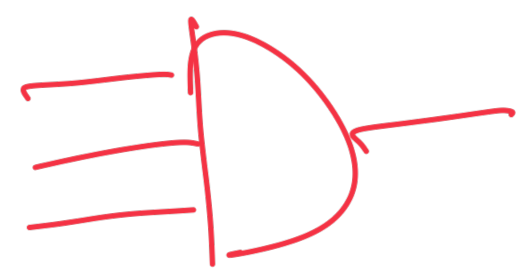
Boolean functions

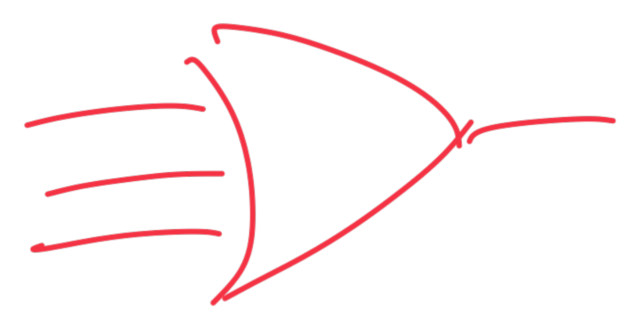
- Boolean functions: can have arbitrary inputs
- $AND: f = x_1 \cdot x_2 \cdot x_3$
- OR: $f = x_1 + x_2 + x_3$

n-inputs $= 2^n$ input possibilities

x_1	x_2	x_3	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 2.7 Three-input AND and OR operations.





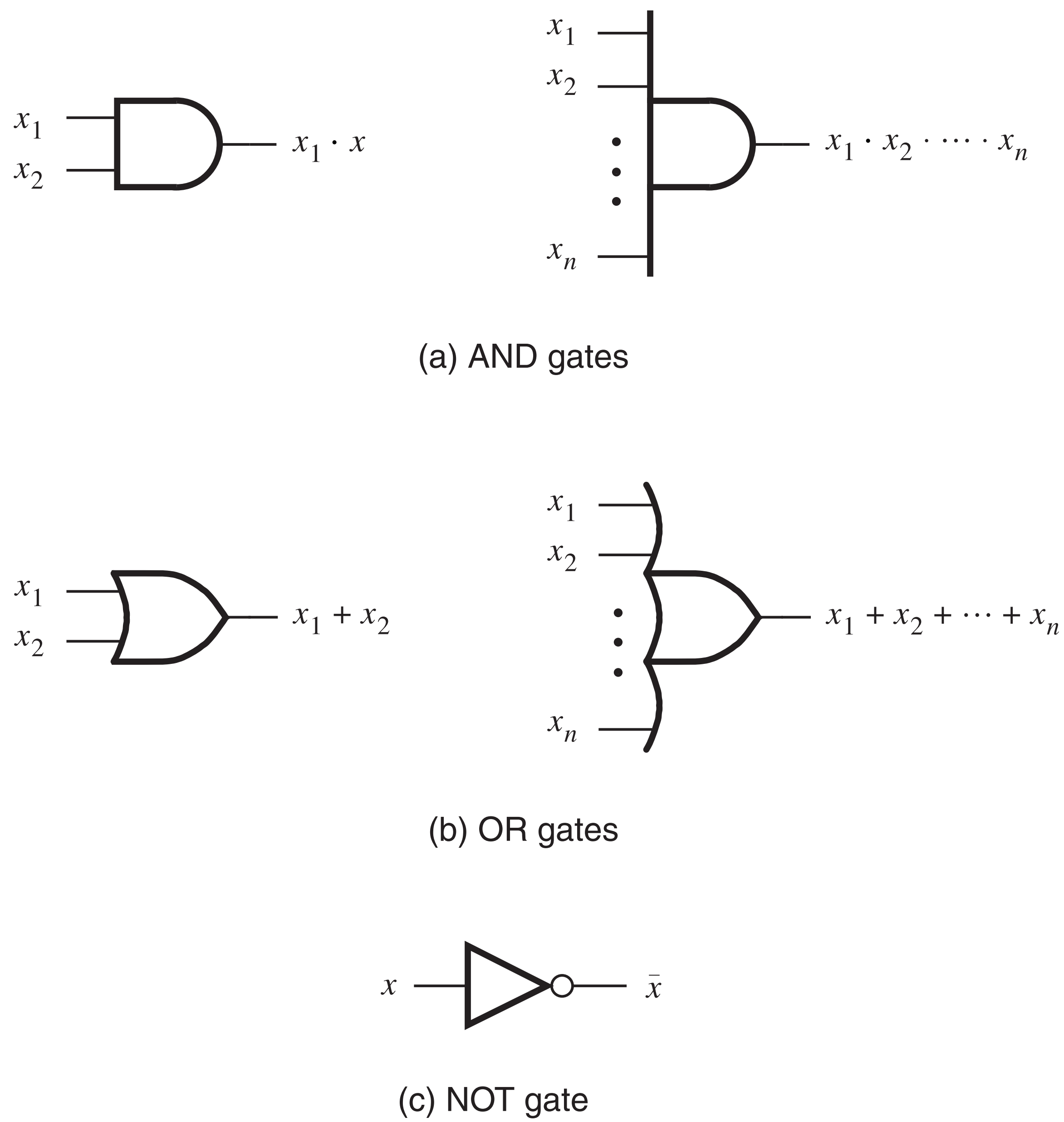
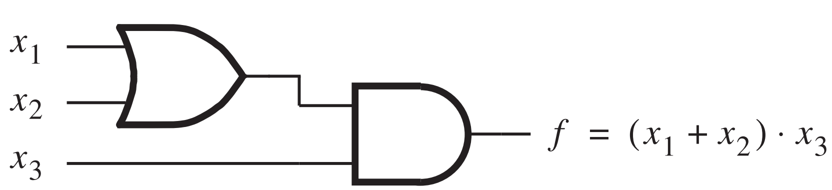
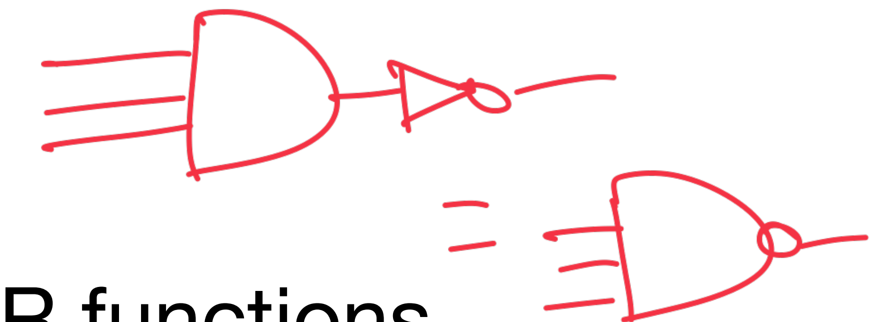


Figure 2.8 The basic gates.



NAND and NOR Functions

- We saw AND and OR functions
- Invert them, and you get NAND and NOR functions



			And	OR	Nand	NOR
x_1	x_2	x_3	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$	$x_1 \cdot x_2 \cdot x_3$	$x_1 + x_2 + x_3$
0	0	0	0	0	1	1
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	1	0	0

Figure 2.7 Three-input AND and OR operations.

Exclusive-OR (XOR) and XNOR functions

- XOR and XNOR functions in two variables
- XOR: the function is true when the inputs are mutually exclusive: denoted $f = x_1 \oplus x_2$
- XNOR: $f = \overline{x_1 \oplus x_2} = x_1 \overline{\oplus} x_2$

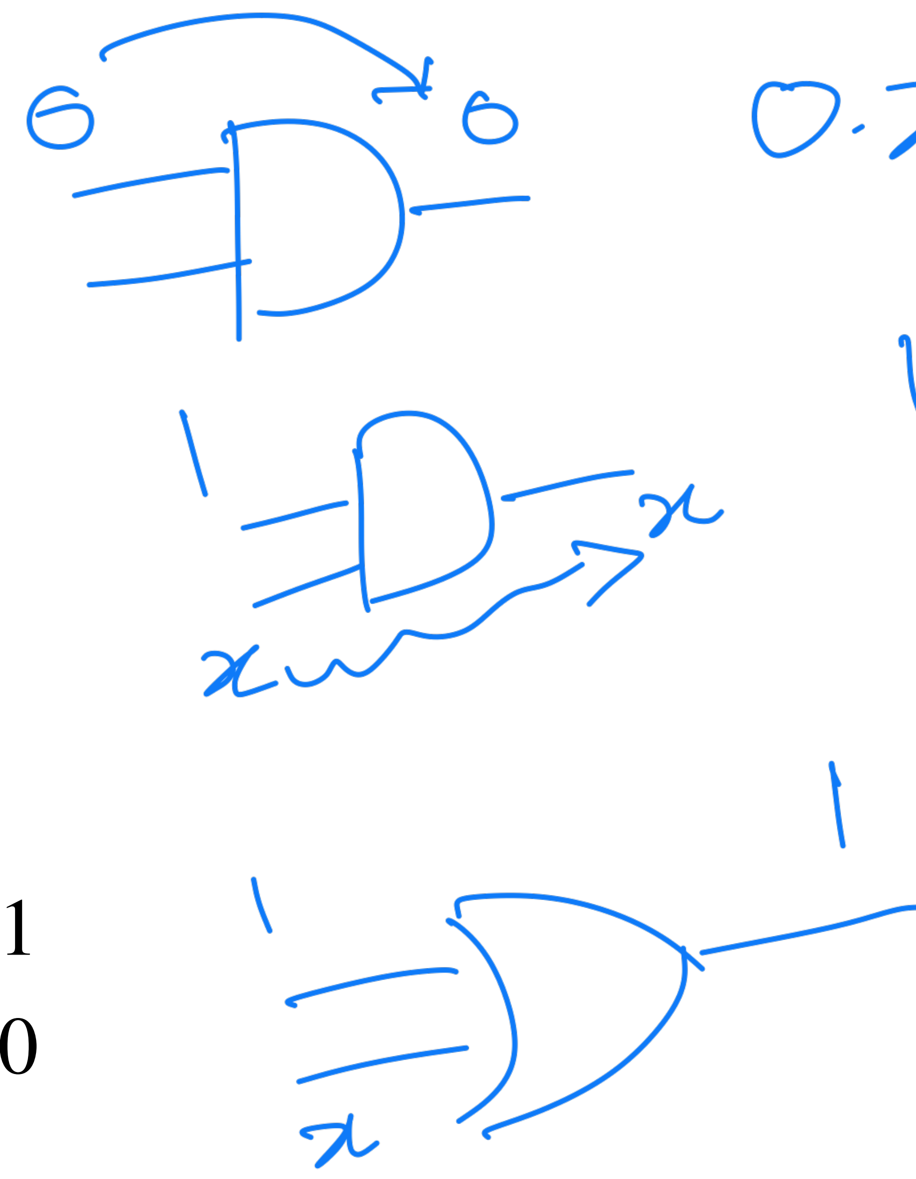
x_1	x_2	$x_1 \oplus x_2$	$x_1 \overline{\oplus} x_2$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

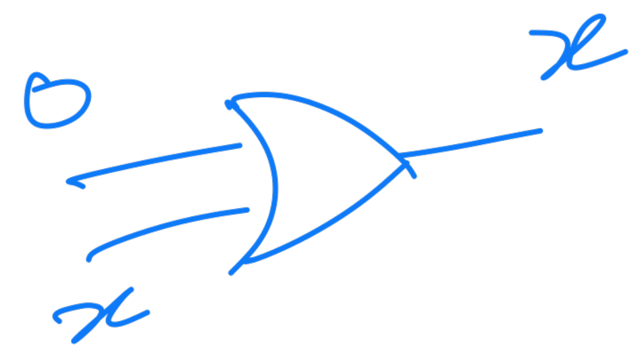
Boolean Algebra Axioms

$$\begin{array}{ll}
1a. & 0 \cdot 0 = 0 \\
1b. & 1 + 1 = 1 \\
2a. & 1 \cdot 1 = 1 \\
2b. & 0 + 0 = 0 \\
3a. & 0 \cdot 1 = 1 \cdot 0 = 0 \\
3b. & 1 + 0 = 0 + 1 = 1 \\
4a. & \text{If } x = 0, \text{ then } \overline{x} = 1 \\
4b. & \text{If } x = 1, \text{ then } \overline{x} = 0
\end{array}$$

$0 \cdot x = 0$

$1 \cdot x = x$



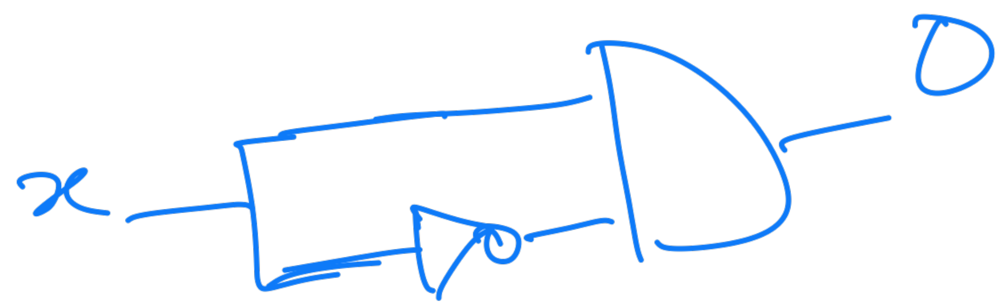


NOT operators

$$\overline{(\overline{x})} = x$$

$$x \cdot x = x$$

$$x + x = x$$



$$x \cdot \overline{x} = 0$$

$$x + \overline{x} = 1$$

"redundancies in logic design"

Design Problem: Going from a Specification to a circuit implementation

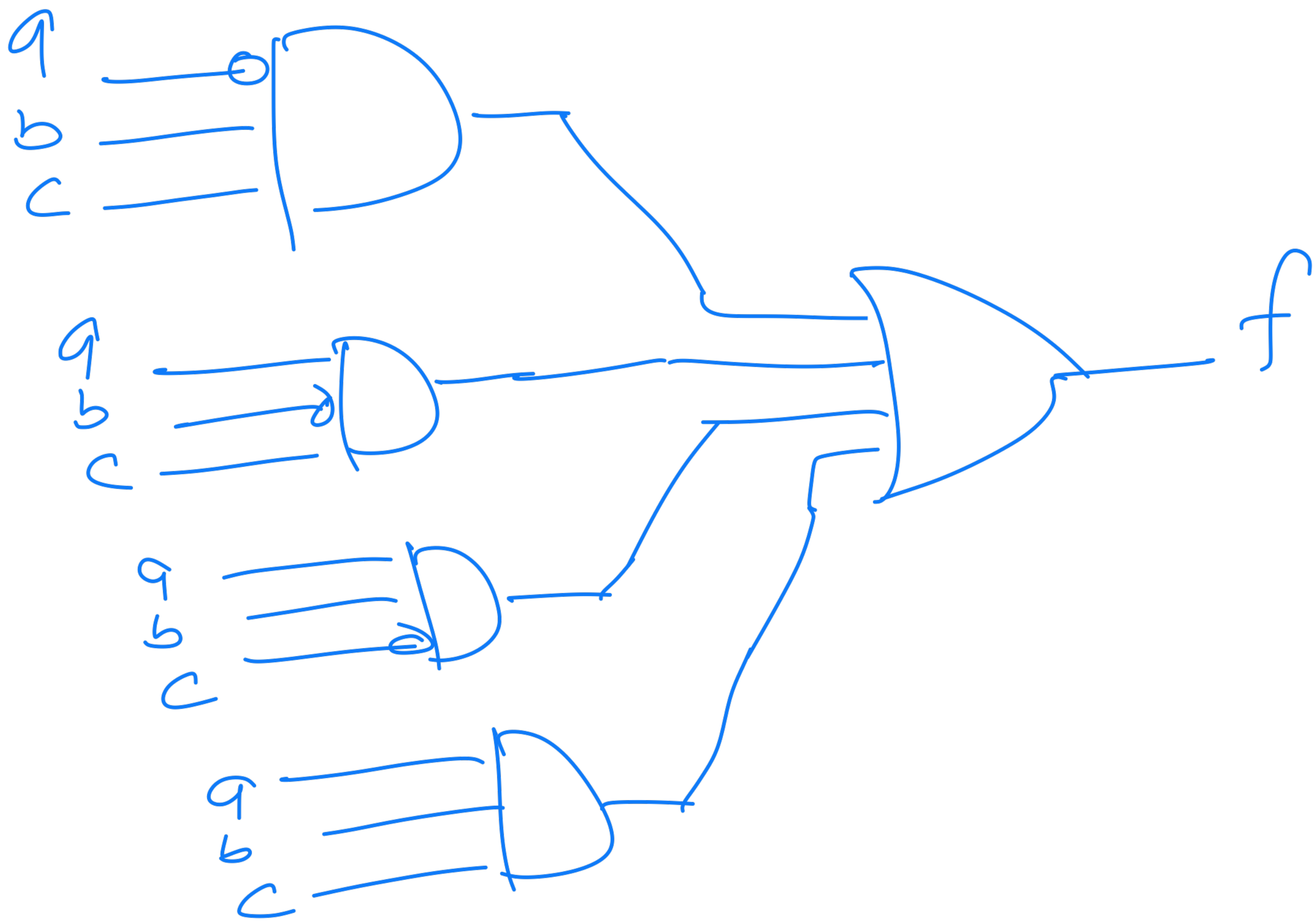
- Design a circuit with three inputs a, b, c and one output f
- Function f = TRUE when majority of inputs are TRUE, FALSE otherwise
- First job = write a truth table
- Collect the product terms (input product) that evaluates f = 1
- SUM (OR) of all these product terms

Truth Table

a	b	c	f
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	1
1	1	1	1

$$\bar{a}\cdot b\cdot c + a\cdot\bar{b}\cdot c + a\cdot b\cdot\bar{c} + a\cdot b\cdot c = f$$

$f = \bar{a}bc + a\bar{b}c + ab\bar{c} + abc$



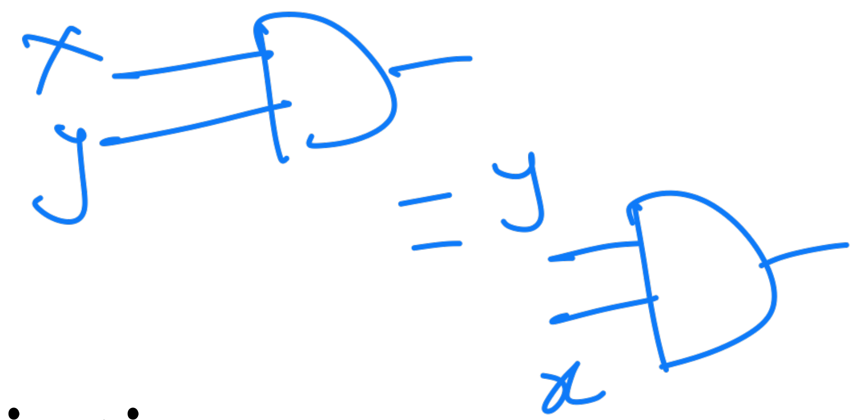
later on: $f = ab + ac + bc$

Boolean Algebra Properties

10a. $x \cdot y = y \cdot x$

10b. $x + y = y + x$

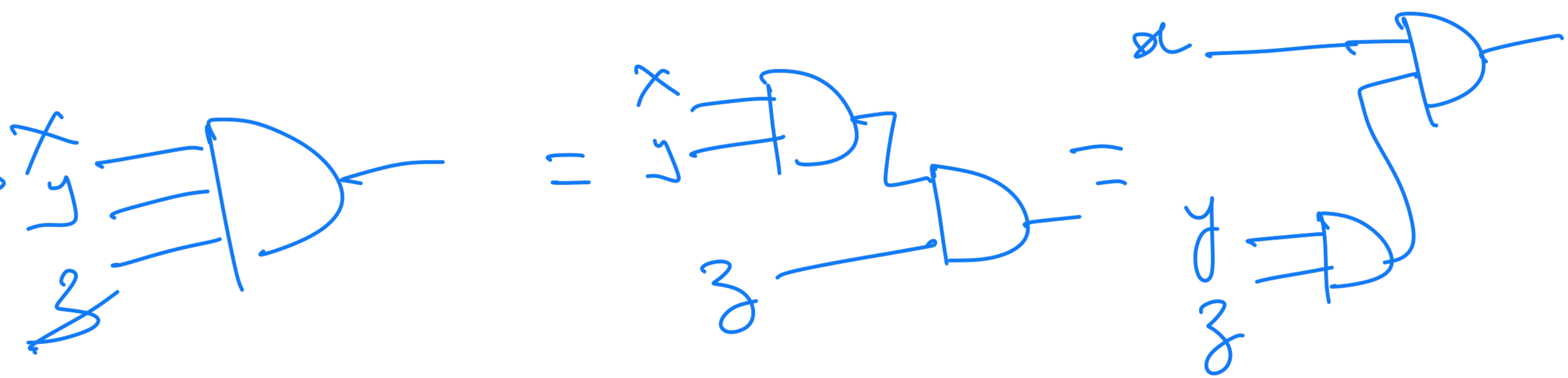
Commutative



11a. $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

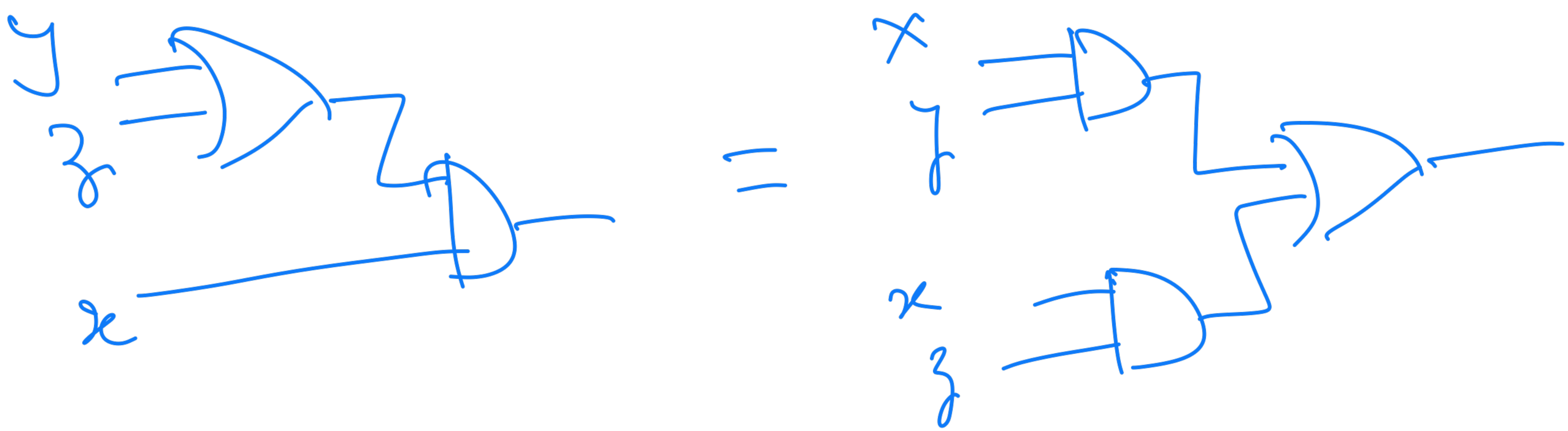
11b. $x + (y + z) = (x + y) + z$

Associative



Boolean Algebra Properties

12*a*. $x \cdot (y + z) = x \cdot y + x \cdot z$ *Distributive*



Different structure same function.

Boolean Algebra Properties

$13a. \quad x + x \cdot y = x$ *Absorption*

$$x \cdot 1 + x \cdot y = x(1+y) = x \cdot 1 = x$$

$13b. \quad x \cdot (x + y) = x$

$$= x \cdot x + x \cdot y$$

$$= x + xy$$

$$= x$$

DeMorgan's Law

- Break the line and change the sign

$$\overline{x+y} = \bar{x} \cdot \bar{y}$$

- Make the line and change the sign

$$\overline{x \cdot y} = \bar{x} + \bar{y}$$

x	y	$x \cdot y$	$\overline{x \cdot y}$	$\bar{x}$	$\bar{y}$	$\bar{x} + \bar{y}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

LHS | RHS

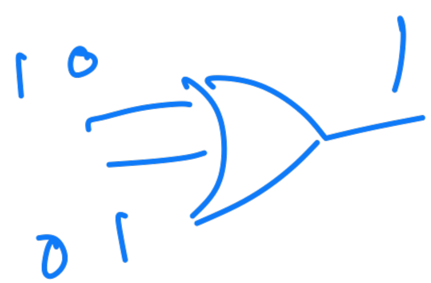
Logic Simplification

- Simplify $(x \cdot y + x \cdot y')$ and $(x + y)(x + y')$

$x(y + y')$

$= x \cdot (1)$

$= x$



$x \cdot x + x\bar{y} + xy + y \cdot \bar{y}$

$= x + x\bar{y} + xy + 0$

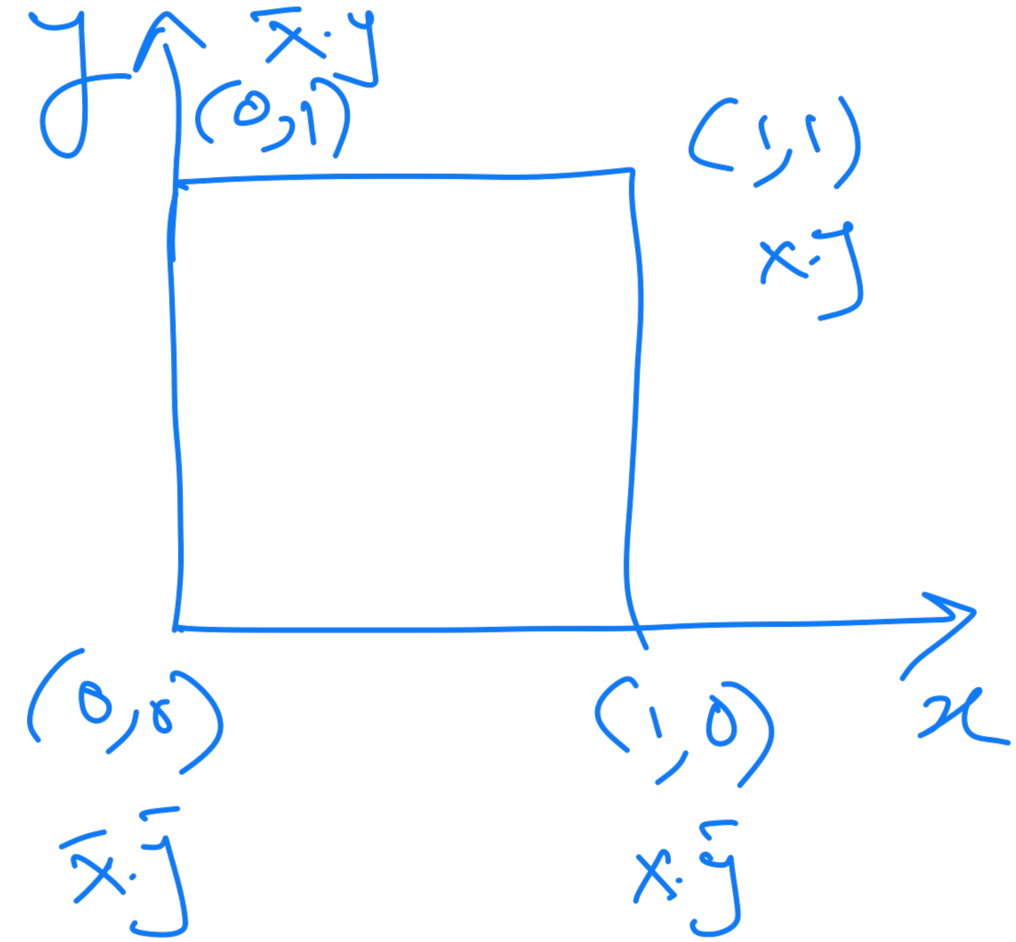
$= x(1 + \bar{y} + y)$

$= x(\underbrace{1 + \text{anything}}_{= 1})$

$= x$

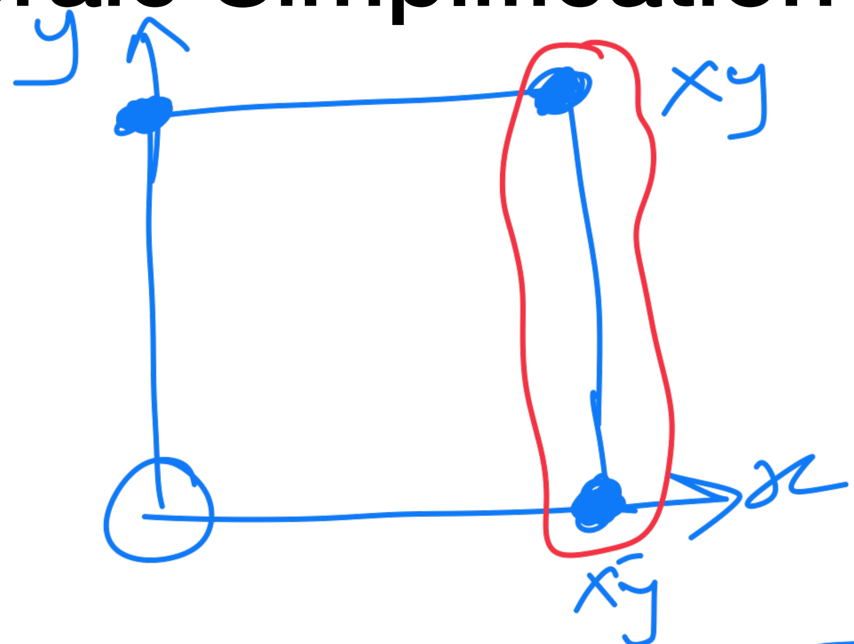
Looking at Algebraic Simplification more Formally

- Consider an n-dimensional Boolean space
- Boolean variable: co-ordinate of the space
- Literal: occurrence of a variable: x or x'
- Minterm: product of all literals, denotes a point in the Boolean space
    - ON-set minterm = minterm where $f = 1$
    - OFF-set minterm = minterm where $f = 0$
- Cube is a product of literals that represents a point or a set of points in the design space
    - Hamming distance: how many bits are changing between any 2 points?



Looking at Algebraic Simplification more Formally

x	y	OR
0	0	0
0	1	1
1	0	1
1	1	1



- Consider the OR function
- $f = xy' + x'y + xy \overset{??}{\rightarrow} x + y$

● = on-set point

○ = off set "

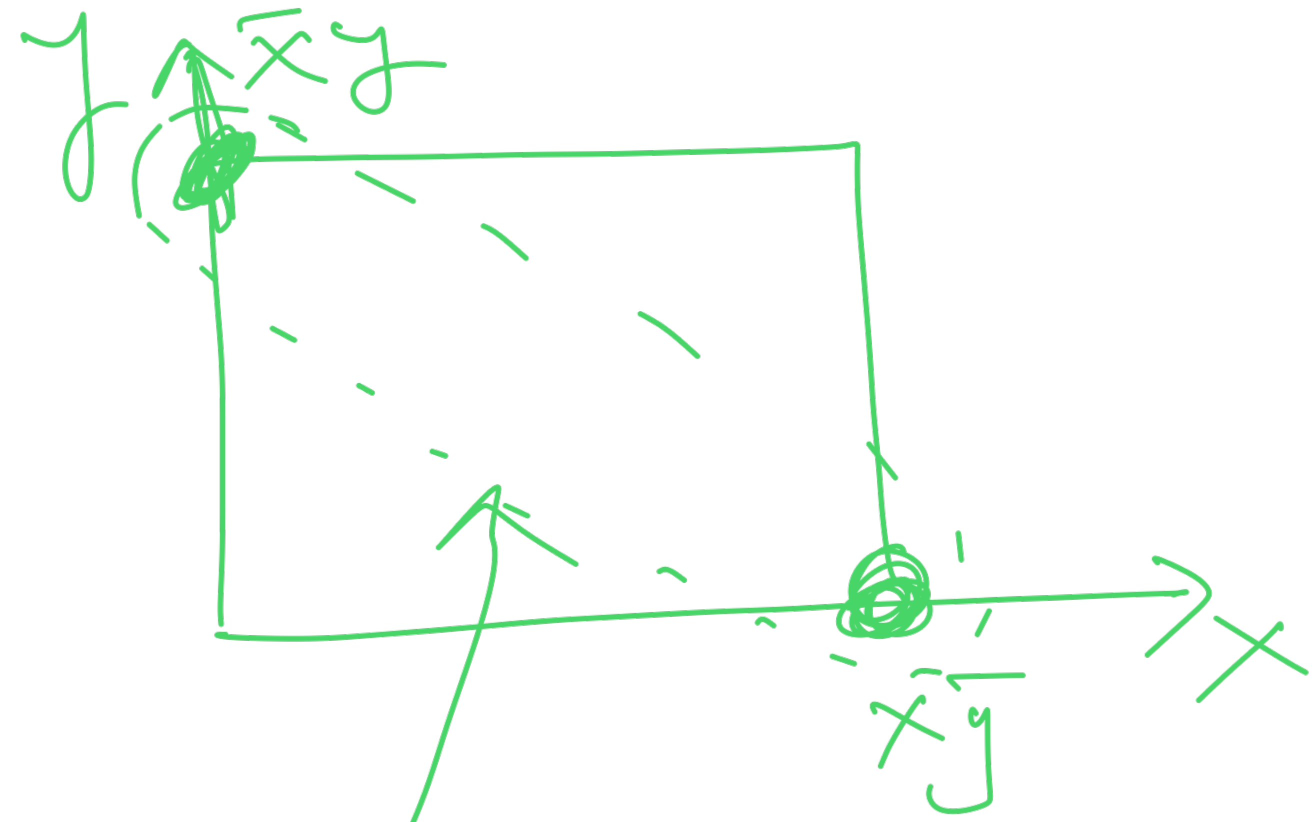
$$f = x\bar{y} + \bar{x}y + xy$$

$$= \bar{x}y + x(y + \bar{y}) + xy$$

$$= \bar{x}y + x(1) + xy$$

$$= y + x = x + y$$

Looking at Algebraic Simplification more Formally



- Consider the XOR function
- $f = x \oplus y = xy' + x'y$
- No simplification?

2 hamming distance

$x\bar{y} + \bar{x}y$ = no combination nor factorization

Apply DeMorgan's Laws: XOR/XNOR

- Consider the XOR function
- $f = x \oplus y = xy' + x'y$

  $\bar{f} = \overline{xy' + x'y}$

  $= \overline{xy'} \cdot \overline{x'y}$

  $= (x' + y)(x + y')$
- $= x'x + x'y' + xy + yy'$

  $= x'y' + xy = x \overline{\oplus} y$

x_1	x_2	$x_1 \oplus x_2$	$x_1 \overline{\oplus} x_2$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

"Sum-of-product" form Boolean function.

product = And gates

sum = OR-gates

Interesting 3-variable Boolean functions

- 3-variable XOR, Majority Function, Multiplexors
- We've already seen the majority function
- 3-var XOR: $(x \oplus y) \oplus z$

x	y	z	Majority	XOR
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$\bar{x}(y \oplus z) + x(y\, \overline{\oplus z})$$

$$= \bar{x} \cdot A + x \cdot \bar{A}$$

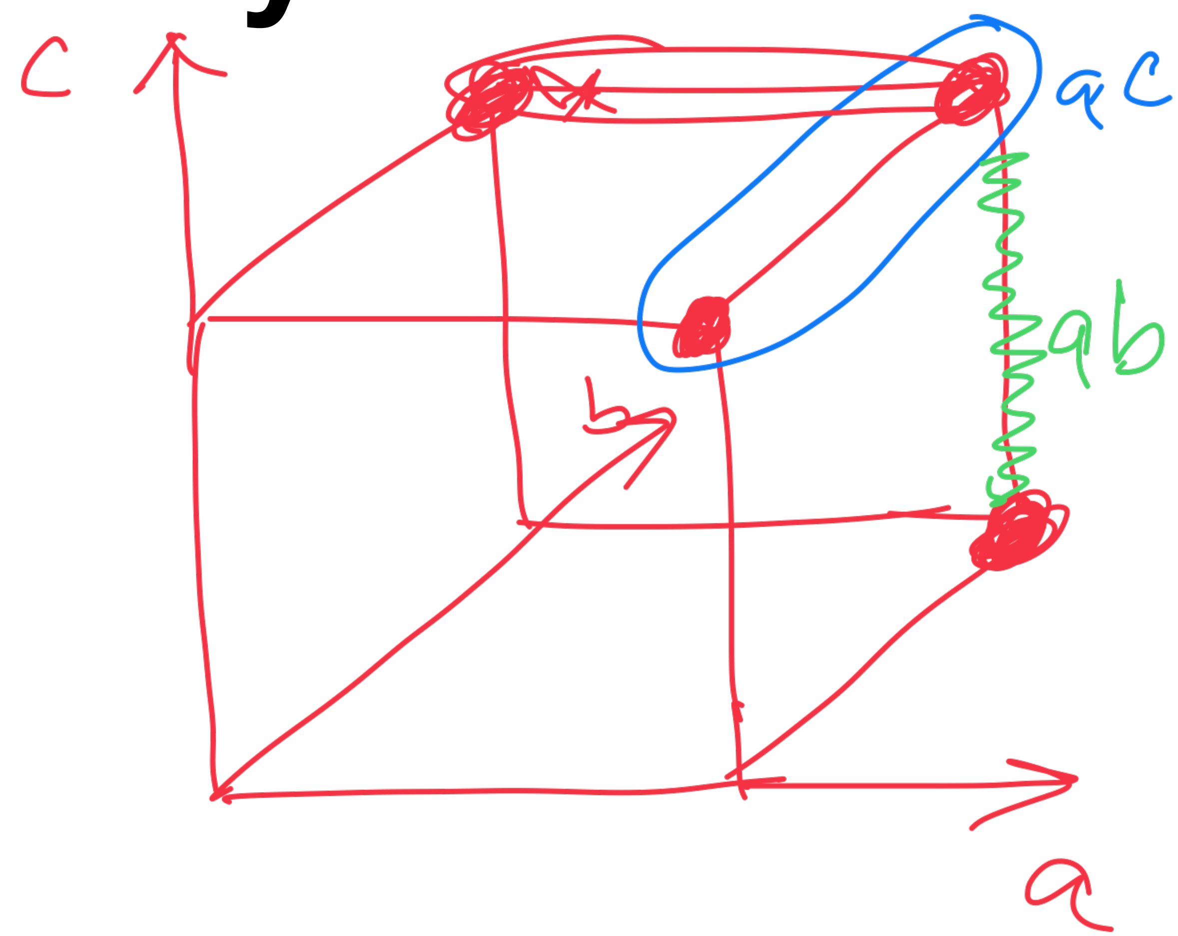
$$= x \oplus A$$

$$= x \oplus y \oplus z$$

Simplify the Majority Function

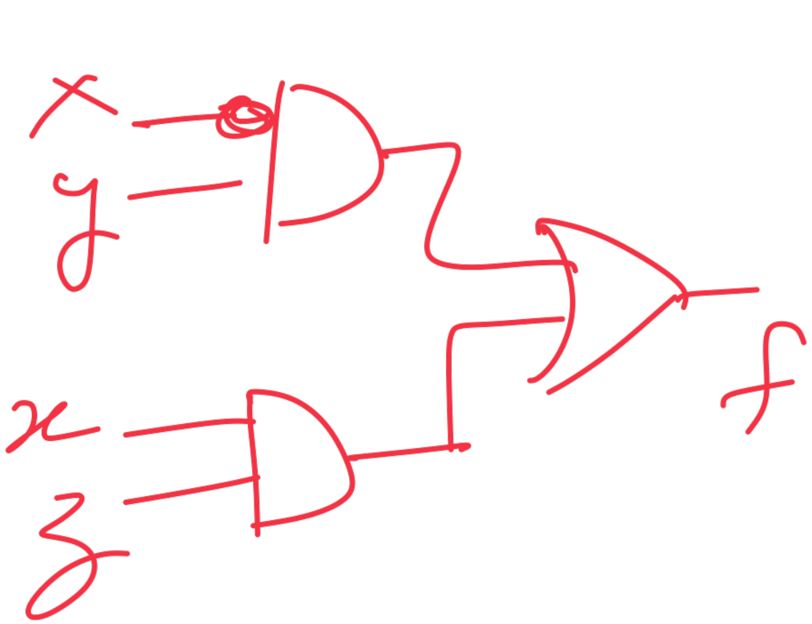
- $f = a'bc + ab'c + abc' + abc$
- $= ab + bc + ac$

$$= ac(b+\bar{b}) + ab(c+\bar{c}) + bc(a+\bar{a}) = ab + ac + bc$$

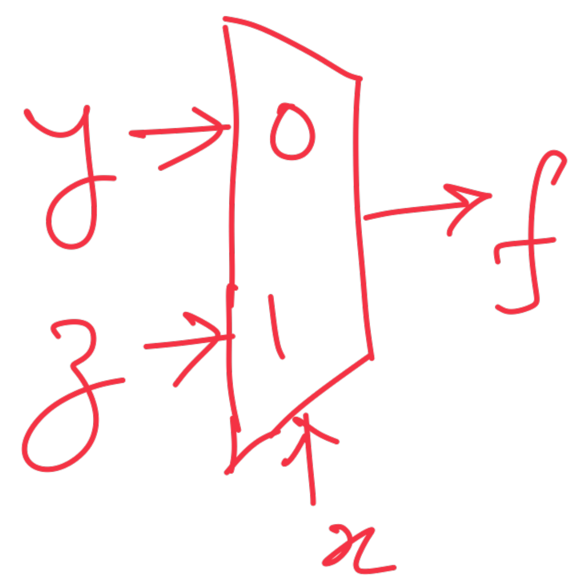


Interesting 3-variable Boolean functions

- Multiplexors = MUX(x, y, z) $= \bar{x}y + xz$
- When x=0, f = y
- When x = 1, f = z



x	y	z			Mux
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	1	1



Use of Multiplexors

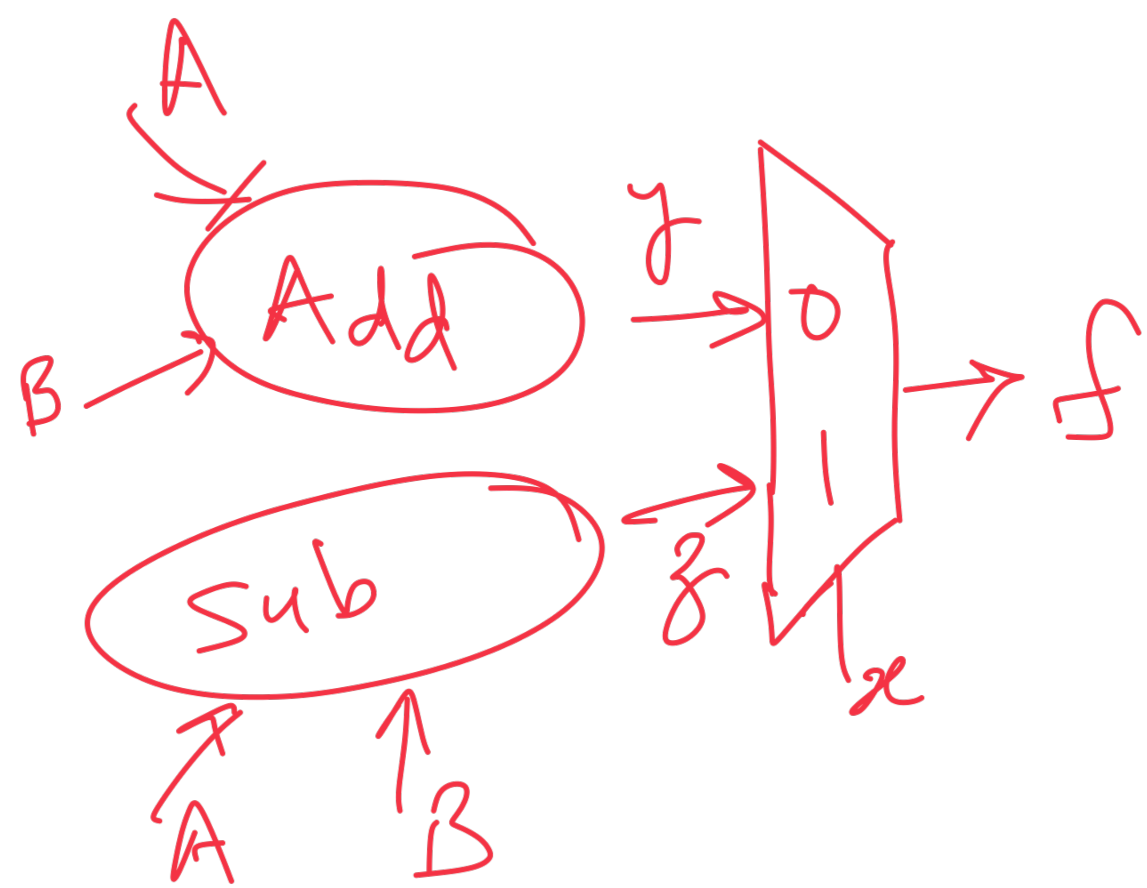
- MUX: multiplexer, multiplexor = multiple xor
- MUXes are everywhere

xor $= \bar{x}y + x\bar{y}$

mux $\quad \bar{x}y + xz$

$x = 0 \quad f = A + B$

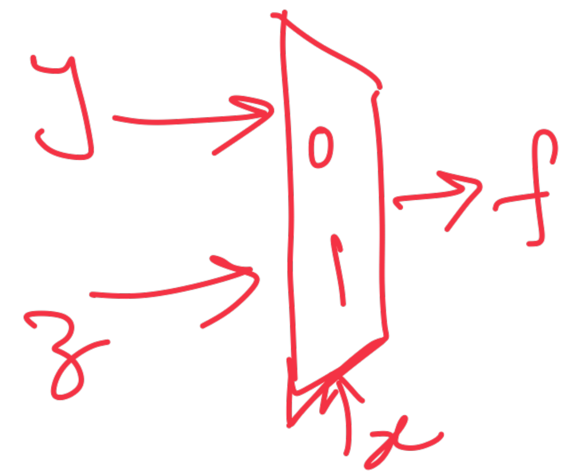
$x = 1 \quad f = A - B$

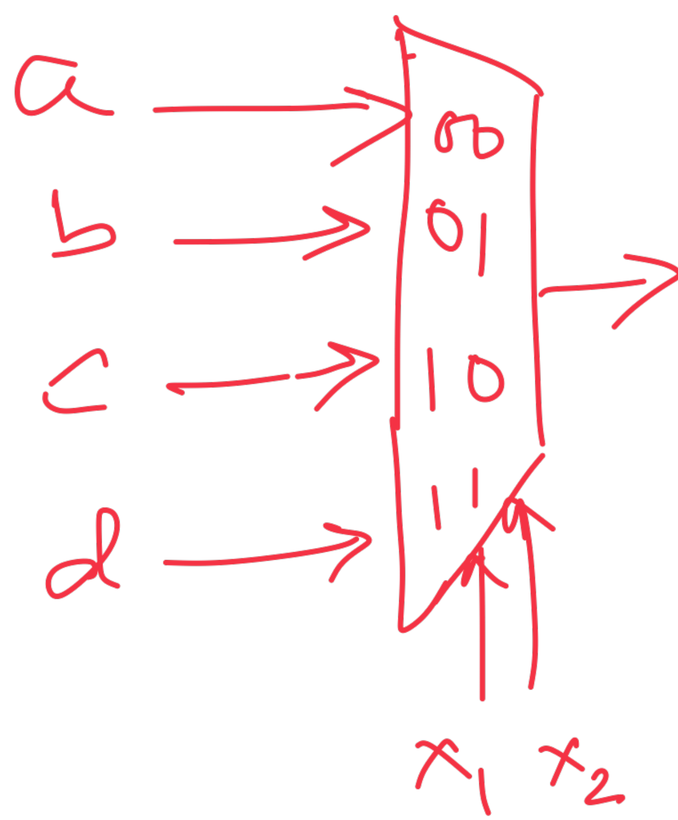


Mux = <u>if-then-else</u>

Use of Multiplexors

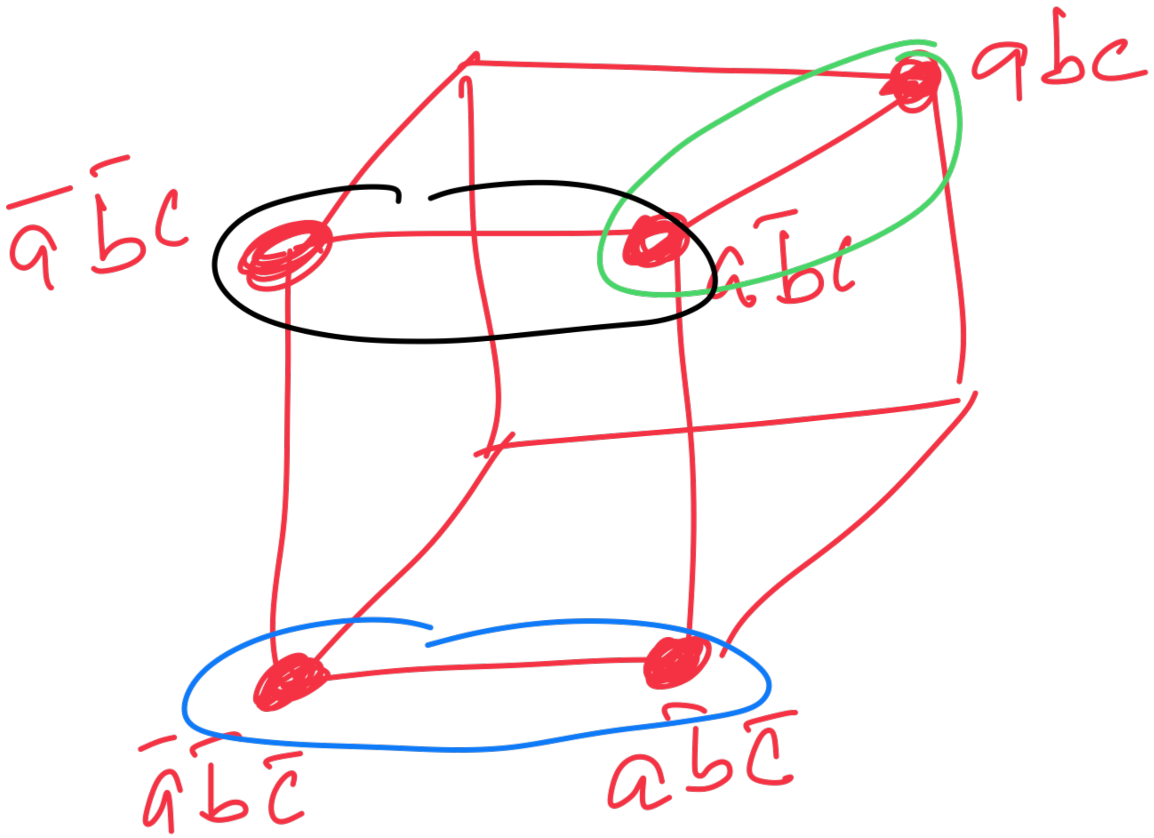
- One-bit control, 2 data inputs
- 2-bit control, 4 data inputs
- N-bit control, 2^N data inputs





$$f = \bar{x}_1 \bar{x}_2 \cdot a + \bar{x}_1 \cdot x_2 \cdot b + x_1 \cdot \bar{x}_2 \cdot c + x_1 x_2 d$$

Logic Optimization: One more example



$$f = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + \bar{a}\bar{b}c + a\bar{b}c + abc$$

$$= ac + \bar{b}c + \bar{b}\bar{c}$$

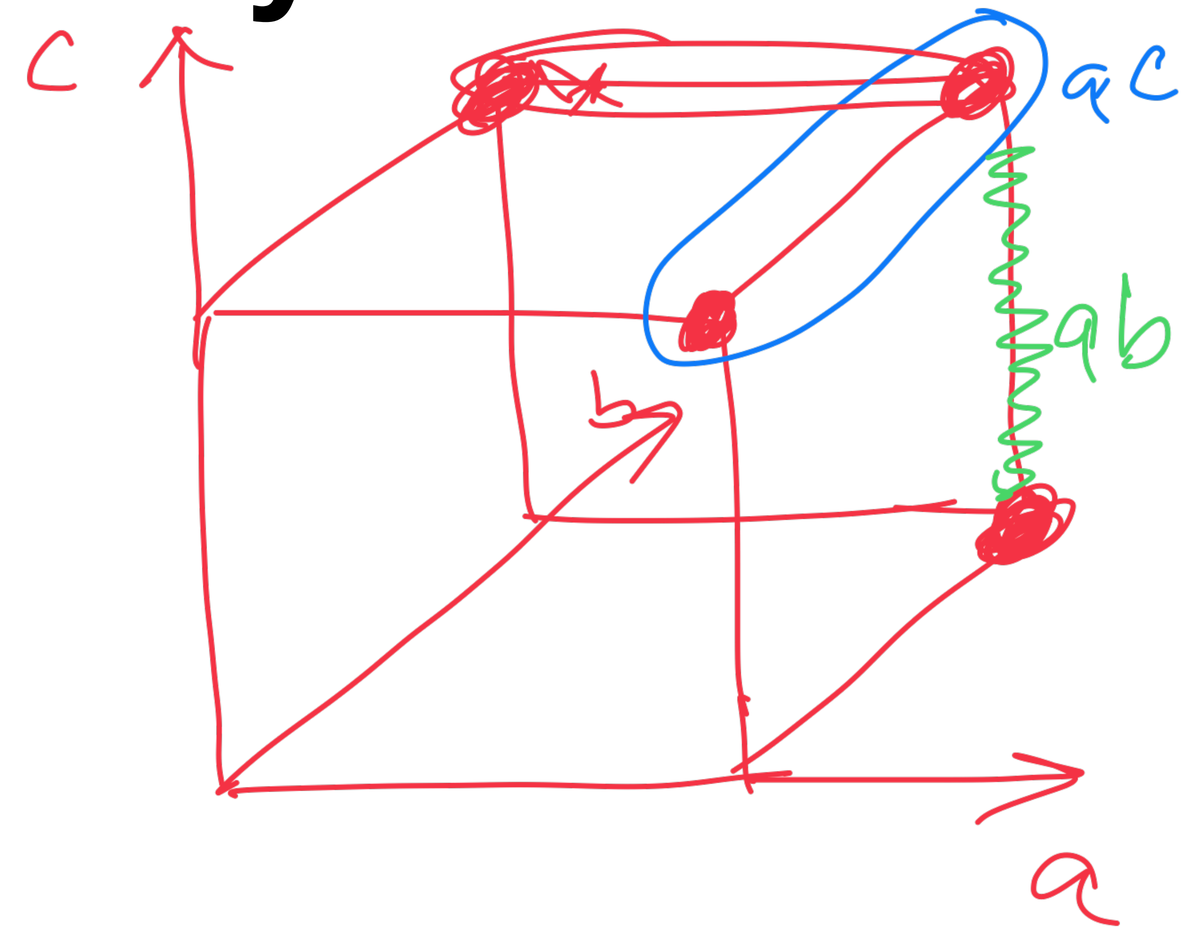
$$= ac + \bar{b}$$

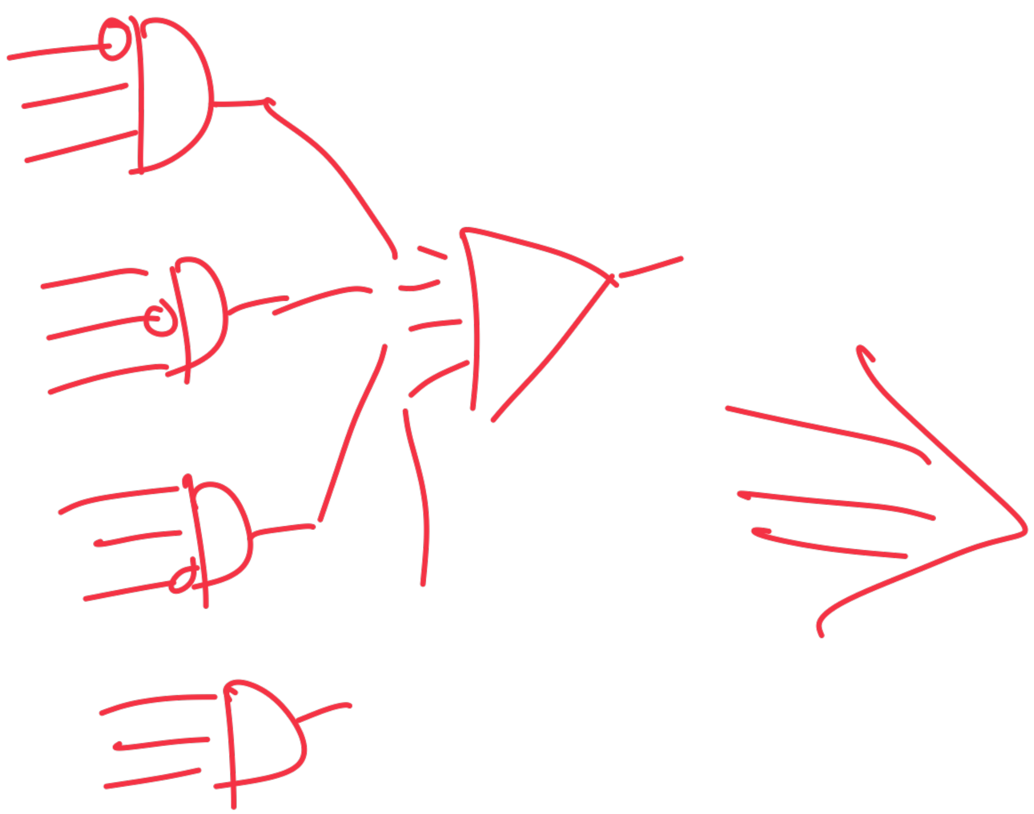
Sum of Product (SOP) form of Boolean functions

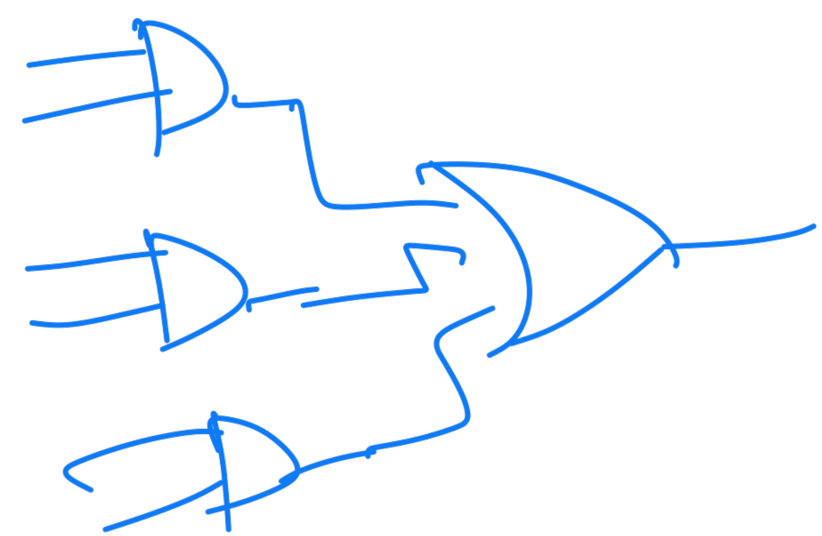
- $F(x_1, \ldots, x_n) =$ sum of ON-set minterms
- Simplify ON-set minterms into "larger cubes": combine minterms that are one hamming distance apart, and keep on combining them as much as possible
    - Algebraically: factorize and simplify
- Identify a minimum number of largest cubes that cover all the ON-set minterms. [Often called a "minimum cover" of F]
- "Smallest" cover exists, find it!
- Larger cubes = smaller AND gates = fewer transistors = fewer literals
- Minimum number of cubes = smallest OR gate
- Smaller area, faster circuit (less *RC* to charge/discharge)
- SOP form = two-level logic: one-level of AND gates, and one-level of (possibly big!) OR-gate. [Ignore the level corresponding to inverters)

Simplify the Majority Function

- $f = a'bc + ab'c + abc' + abc$
- $= ab + bc + ac$







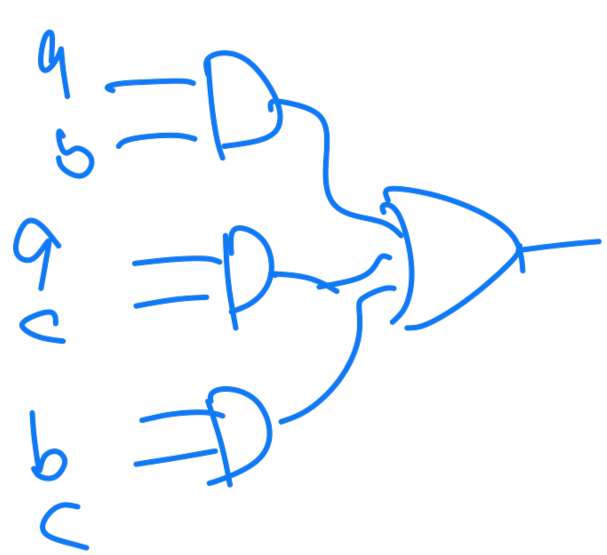
SOP-form, contd.

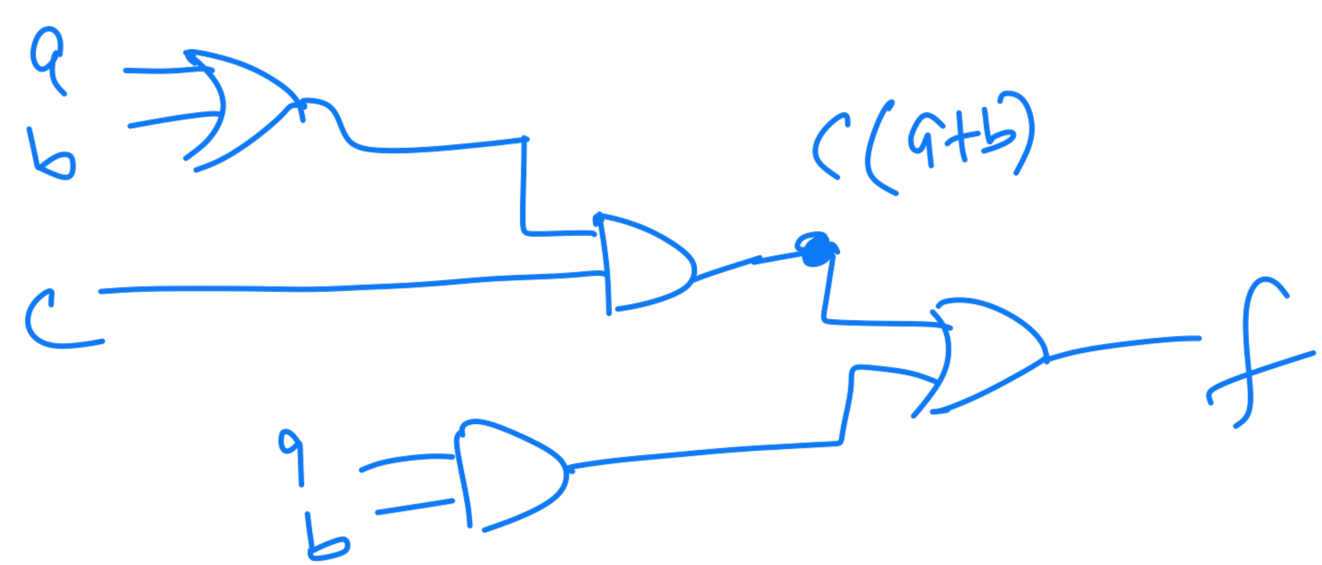
- Given $F(x_1, x_2, x_3, \ldots)$, with variable order $x_1, x_2, x_3 \ldots$.
- F can be specified as a sum of ON-set minterms
- E.g., majority function:
    - $F(x_1, x_2, x_3) = \sum (m_3, m_5, m_6, m_7)$
- Self-study assignment for you: Sec 2.6.1, product of sum forms, and Maxterms

Row number	x_1	x_2	x_3	Minterm
0	0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$
1	0	0	1	$m_1 = \bar{x}_1\bar{x}_2 x_3$
2	0	1	0	$m_2 = \bar{x}_1 x_2\bar{x}_3$
3	0	1	1	$m_3 = \bar{x}_1 x_2 x_3$
4	1	0	0	$m_4 = x_1\bar{x}_2\bar{x}_3$
5	1	0	1	$m_5 = x_1\bar{x}_2 x_3$
6	1	1	0	$m_6 = x_1 x_2\bar{x}_3$
7	1	1	1	$m_7 = x_1 x_2 x_3$

Two-level logic versus multi-level logic

- $f = ab + ac + bc$ = SOP form = 2-level
  - Objective: minimum number of largest cubes = minimum cover
- Factorize: f = ab + c(a+b) = factored form ≠ SOP form
  - Multi-level logic: minimize the number of “literals”
- Some technologies are suitable for 2-level logic (such as PLAs), whereas others are suitable for multi-level logic (contemporary CMOS technologies)
  - We’ll study this a little later, in appendix B, in the textbook!
- Multi-level logic optimization often utilizes 2-level optimization techniques, so study of 2-level SOP form minimization is a must!





With more variables, Logic simplification becomes infeasible using algebraic/symbolic manipulation. We need algorithmic techniques, which we'll study a bit later…

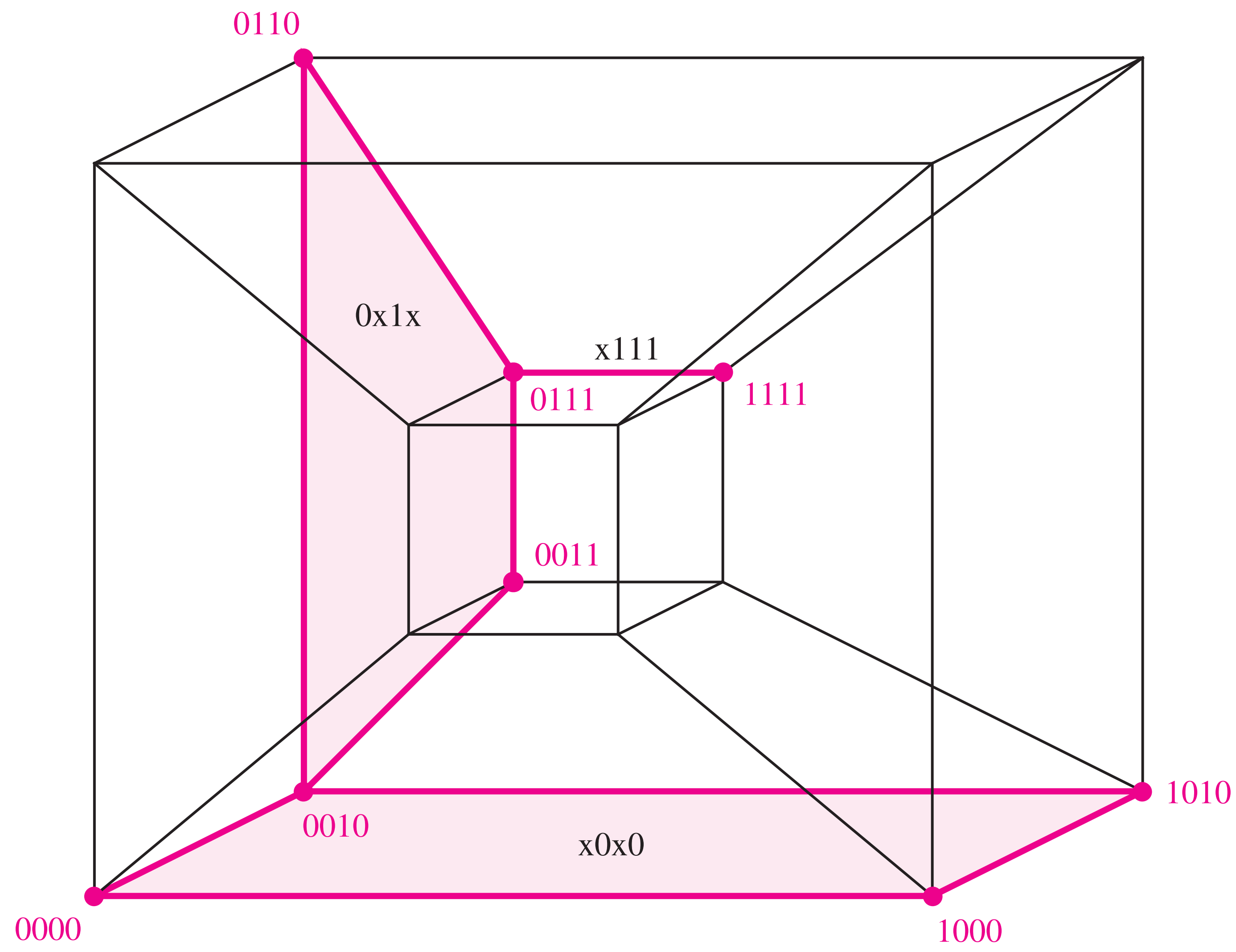


Figure 8.18 Representation of function f_3 from Figure 2.54

A 4-dimensional cube