Lecture 23: Boolean Function & Circuits

Jingjin Yu | Computer Science @ Rutgers

December 1, 2020

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Algorithmic Robotics & Control Lab

Outline

- Lecture 21-22 review
- Boolean Functions
  - Boolean algebra
  - Boolean functions
  - Duality
- Representing Boolean functions
  - Minterms and sum-of-product
  - Functional completeness
- Logic gates and Boolean circuits

- A repeating note: **make sure you read the textbook**

L21-22: Primality & FTA

- **Def^{n}**: An integer $p > 1$ is a **prime** if its only positive factors are 1 and p. Otherwise, p is a composite
- **Lemma 1**. If $n > 1$ is a composite, then n has a prime factor of no more than $\sqrt{n}$.
- **Ex**: For 101, only need to check $2, 3, 5, 7$ $(< \sqrt{101})$
- **Theorem 2**. (Fundamental Theorem of Arithmetic). Every integer > 1 can be written **uniquely** as a prime or as the product of two or more primes where the prime factors are written in non-decreasing size.
- **Ex**: $1024 = 2^{10}$, $100 = 2^2 5^2$, 1500450271 is a prime

L21-22: Prime Factorization Algorithm

- **Algorithm**. (Prime Factorization). To factorize n or determine it is a prime, let $n_1 = n$. Starting with $i = 1$, do
  1) Let the prime numbers $\leq \sqrt{n_1}$ be $p_1 = 2, p_2 = 3, \ldots$
  2) Test whether $p_j \mid n_i$ starting from $j = 1,2, \ldots$ two possibilities
  3) If for some j, $p_j \mid n_i$, let $n_{i+1} = \frac{n_i}{p_j}$ and $q_i = p_j$. Go to 1) with $i = i + 1$
  4) Let $q_i = n_i$. All the q_i collected so far are the prime factors

- **Ex**: Factor $n = 126$

L21-22: Finding All Primes $< n$

- **Algorithm**. (Sieve of Eratosthenes). List all primes $p_1 = 2, p_2 = 3, \ldots, p_k$ less than $\sqrt{n}$. Then starting with $p_1 = 2$, remove all numbers less than n divisible by p_1, except p_1. Repeat until p_k is done
- **Ex**: Find all primes less than 40

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40

- Properties of primes (e.g., Goldbach's conjecture)

L21-22: GCD and LCM

- **Def^{n}**: Let $a \neq b$ be integers. The largest d such that $d \mid a$ and $d \mid b$ is the **greatest common divisor** of a and b, or $\gcd(a, b)$.
- **Ex**: $\gcd(35, 28) = 7$, $\gcd(5, 22) = 1$

- **Def^{n}**: Integers a and b are **relatively prime** if $\gcd(a, b) = 1$.
- **Ex**: 5 and 22 are relative primes.

- **Def^{n}**: The **least common multiple** of positive integers a and b is the smallest positive integer that is divisible by a and b, denoted as $\operatorname{lcm}(a, b)$.
- **Ex**: $\operatorname{lcm}(3, 6) = 6$, $\operatorname{lcm}(4, 5) = 20$
- **Theorem 6.** For positive integers a and b,

$$ab = \gcd(a, b) \cdot \operatorname{lcm}(a, b)$$

L21-22: Euclid's Algorithm

- **Lemma 7**. Let $a = bq + r$ where a, b, q, and r are integers. Then $\gcd(a, b) = \gcd(b, r)$
  - Proof: $d = \gcd(a, b) \Rightarrow d|a, d|b \Rightarrow d|-bq \Rightarrow d|(a - bq) \Rightarrow d|r$
- **Algorithm**. Let $r_0 = a > r_1 = b$
  - $r_0 = r_1 q_1 + r_2, 0 < r_2 < r_1$
  - $r_1 = r_2 q_2 + r_3, 0 < r_3 < r_2$
  - …
  - $r_{n-2} = r_{n-1}\, q_{n-1} + r_n, 0 < r_n < r_{n-1}$
  - $r_{n-1} = r_n q_n$
  - $\gcd(a, b) = \gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{n-1}, r_n) = r_n$
- **Ex**: Compute $\gcd(662, 414)$

L21-22: Euclid's Lemma

- **Theorem 8**: If a and b are positive integers, then there exists integers s, t such that $\gcd(a, b) = sa + tb$
  - A constructive proof via Euclid's algorithm

- **Corollary 9**: If $\gcd(a, b) = 1$, then $\exists s, t, sa + tb = 1$.
- **Lemma 10**. (**Euclid's lemma**). Let p be a prime. For integers a and b, if $p \mid ab$, then $p \mid a$ or $p \mid b$

- Using Euclid's lemma, we can prove the uniqueness statement in the Fundamental Theorem of Arithmetic

Boolean Algebra

- Basic Boolean values: 1 (True) and 0 (False)
- Operations
  - Completement: $\overline{0} = 1, \overline{1} = 0$
  - "Boolean Sum" (OR): $1 + 1 = 1, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0$
  - "Boolean Product" (AND): $1 \cdot 1 = 1, 1 \cdot 0 = 0, 0 \cdot 1 = 0, 0 \cdot 0 = 0$
    - May omit the "·" when there is no confusion, e.g., $(0 + 1)(1 + 1)$

- **Ex** (Boolean Algebra): $1 \cdot 0 + \overline{(0 + 1)}$

$0 + \overline{1} = 0$

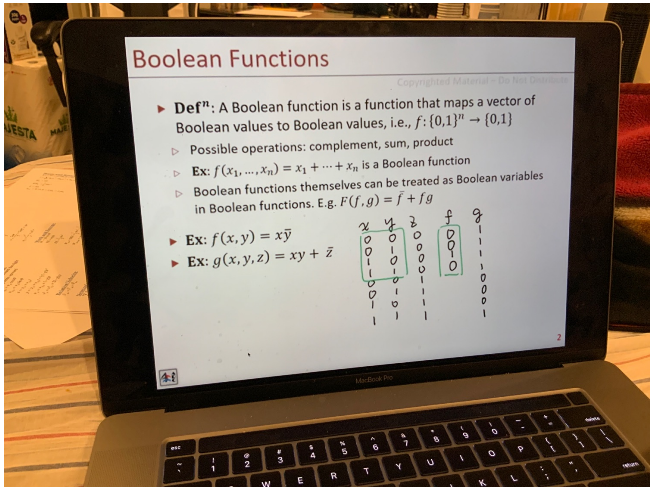
$0 + 0$

- Translating to logical equivalence

Boolean Functions

- **Def^n**: A Boolean function is a function that maps a vector of Boolean values to Boolean values, i.e., $f: \{0,1\}^n \rightarrow \{0,1\}$
  - Possible operations: complement, sum, product
  - **Ex**: $f(x_1, \ldots, x_n) = x_1 + \cdots + x_n$ is a Boolean function
  - Boolean functions themselves can be treated as Boolean variables in Boolean functions. E.g. $F(f, g) = \bar{f} + fg$

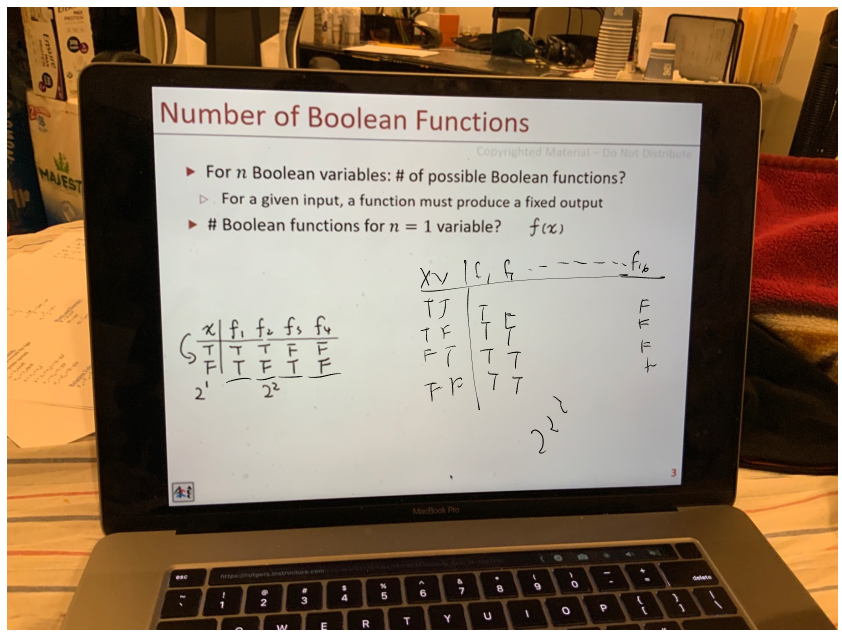
- **Ex**: $f(x, y) = x\bar{y}$
- **Ex**: $g(x, y, z) = xy + \bar{z}$



- **Ex**: $F(x, y, z) = f(x, y)g(x, yz) = x\bar{y}(xy + \bar{z})$

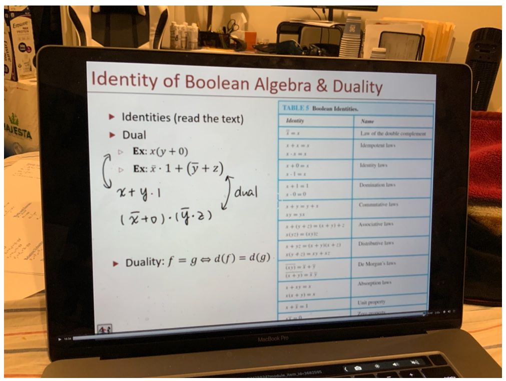
Number of Boolean Functions

- For n Boolean variables: # of possible Boolean functions?
  - For a given input, a function must produce a fixed output
- # Boolean functions for $n = 1$ variable? $f(x)$
- # Boolean functions for $n = 2$ variable? $f(y)$
- In general, 2^{2^n}



Identity of Boolean Algebra & Duality

- Identities (read the text)
- Dual
  - **Ex**: $x(y+0)$
  - **Ex**: $x(y+0)$



- Duality: $f = g \Leftrightarrow d(f) = d(g)$

TABLE 5 Boolean Identities.

Identity	*Name*
$\bar{\bar{x}} = x$	Law of the double complement
$x + x = x$ $x \cdot x = x$	Idempotent laws
$x + 0 = x$ $x \cdot 1 = x$	Identity laws
$x + 1 = 1$ $x \cdot 0 = 0$	Domination laws
$x + y = y + x$ $xy = yx$	Commutative laws
$x + (y + z) = (x + y) + z$ $x(yz) = (xy)z$	Associative laws
$x + yz = (x + y)(x + z)$ $x(y + z) = xy + xz$	Distributive laws
$\overline{(xy)} = \bar{x} + \bar{y}$ $\overline{(x + y)} = \bar{x}\,\bar{y}$	De Morgan's laws
$x + xy = x$ $x(x + y) = x$	Absorption laws
$x + \bar{x} = 1$	Unit property
$x\bar{x} = 0$	Zero property

Sum-of-Product Expansion

- **Question**: Given inputs and outputs of Boolean functions, how can we find the corresponding function?
  - Using **sum-of-product**!

	TABLE 1				
	x	y	z	F	G
1	1	1	1	0	0
2	1	1	0	0	1
3	1	0	1	1	0
4	1	0	0	0	0
5	0	1	1	0	0
6	0	1	0	0	1
7	0	0	1	0	0
8	0	0	0	0	0

- **Def^n**: A **literal** for a Boolean variable x is x or $\bar{x}$.
- **Def^n**: A **minterm** of Boolean variables $x_1, \ldots, x_n$ is a **Boolean product** $y_1 \ldots y_n$ where y_i is a literal for the variable x_i . That is, $y_i = x_i$ or $y_i = \overline{x_i}$.
  - **Ex**: Determine when the minterm $\overline{x_1} x_2 \overline{x_3 x_4} x_5$ takes value 1 and 0.

- **Def^n**: A **sum-of-product** Boolean function is composed of sums of minterms
- Next: Representing Boolean functions as sum-of-products

Boolean Functions as Sum-of-Products

- We want to get an expression of $F(x_1, x_2, \ldots, x_n)$ given entries of the Boolean function

$$x_1^1, x_2^1, \ldots, x_n^1, F^1$$
$$x_1^2, x_2^2, \ldots, x_n^2, F^2$$
$$\ldots \ldots$$
$$x_1^m, x_2^m, \ldots, x_n^m, F^m$$

TABLE 1

	x	y	z	F	G
1	1	1	1	0	0
2	1	1	0	0	1
3	1	0	1	1	0
4	1	0	0	0	0
5	0	1	1	0	0
6	0	1	0	0	1
7	0	0	1	0	0
8	0	0	0	0	0

- **Algorithm**: Set $F = 0$. For each entry $1 \le k \le m$ of function F, if $F^k = 1$, add an additive minterm to F such that if $x_i^k = 1$, the literal x_i^k is added to the minterm; otherwise, the literal $\overline{x_i^k}$ is added to the minterm.

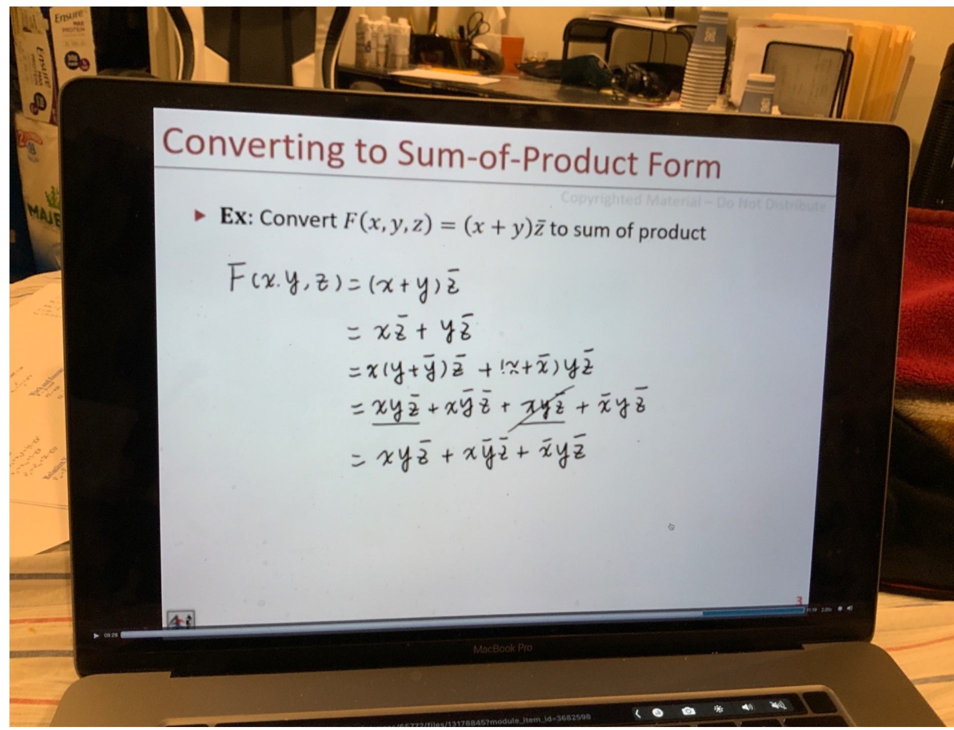
$F = x\overline{y}z$

Converting to Sum-of-Product Form

- **Ex**: Convert $F(x, y, z) = (x + y)\bar{z}$ to sum of product

$$F(x, y, z) = (x+y)\bar{z}$$

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



Functional Completeness

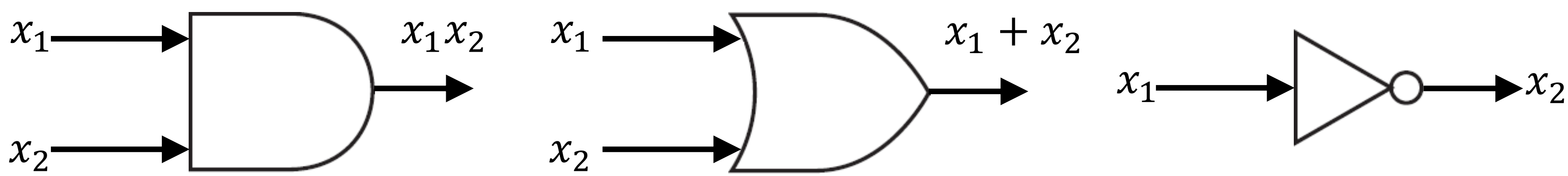
- $\mathbf{Def}^{n}$: A set of Boolean operators is functionally complete if Boolean functions constructed with these operators over n variables can represent all 2^{2^n} possible Boolean functions.

- $\mathbf{Theorem}.$ $\{+,\ \cdot\ ,\ \overline{\ \ }\ \}$ is functionally complete.
  - Proof: every Boolean function can be represented as sum-of-products.

- Other functionally complete operators
  - NAND
  - NOR

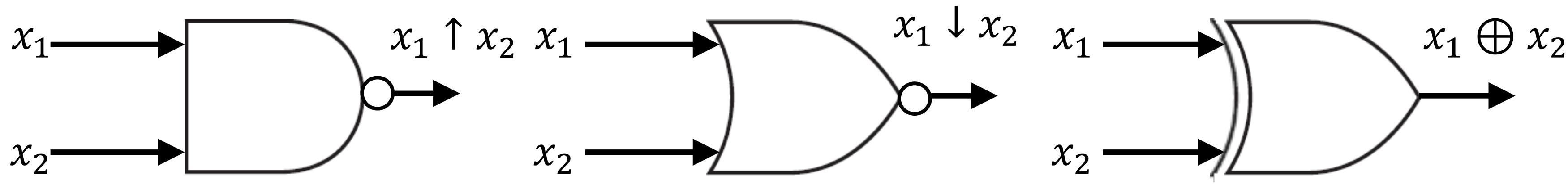
Logic Gates

- NOT (invertor), OR, and AND gates.



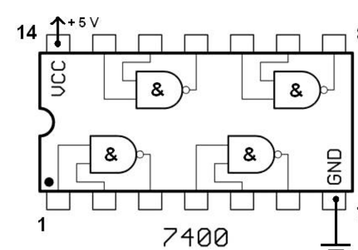
AND OR NOT

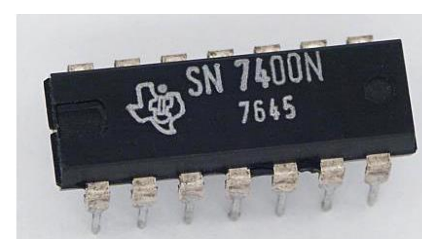
- NAND, NOR, and XOR gates



NAND NOR XOR

- Each gate is a Boolean function

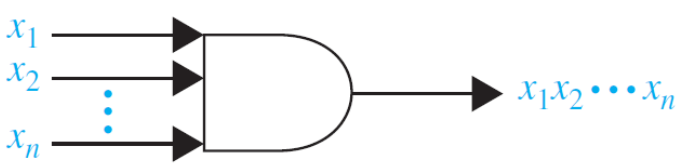




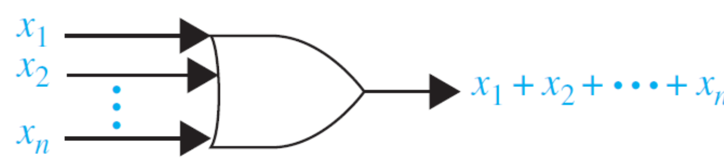
Boolean Circuits

- These “circuit gates” act like actual gates, in a sense
  - There is a trigger that “opens” a gate periodically
  - The specified operation (AND, OR, NOT, …) then happens
- In modern computers, these gates open/close a few trillion times a second, giving us GHz chips.

- Multiple input AND and OR gates



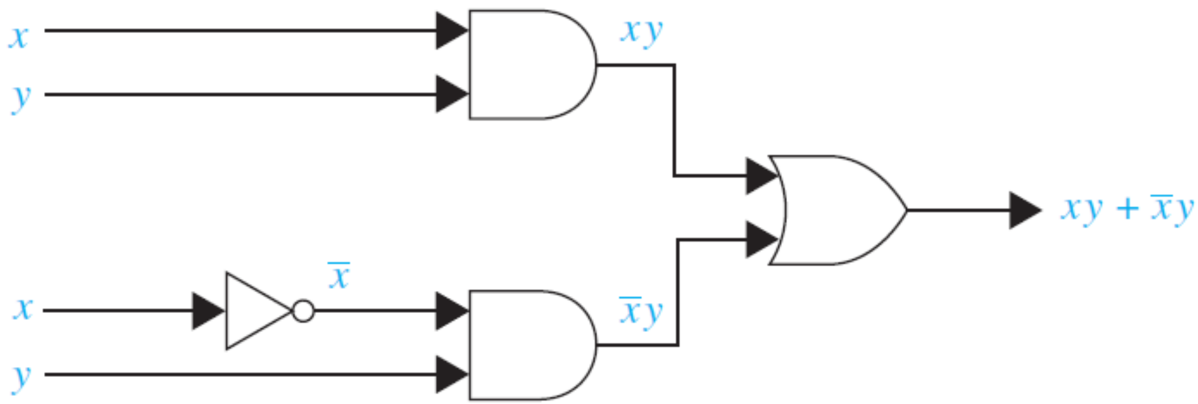
Multiple AND gate

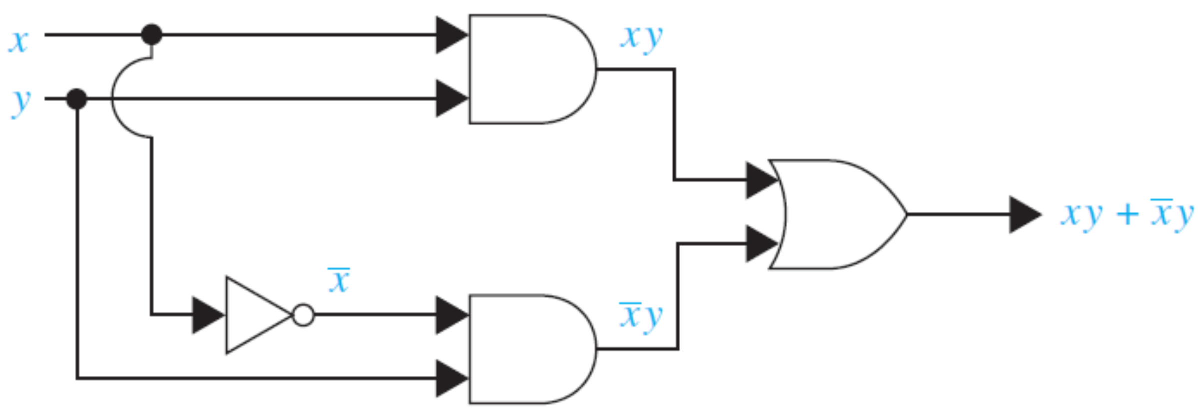


Multiple OR gate

Non-Uniqueness of Boolean Circuits

- **Ex:** $xy + \bar{x}y$

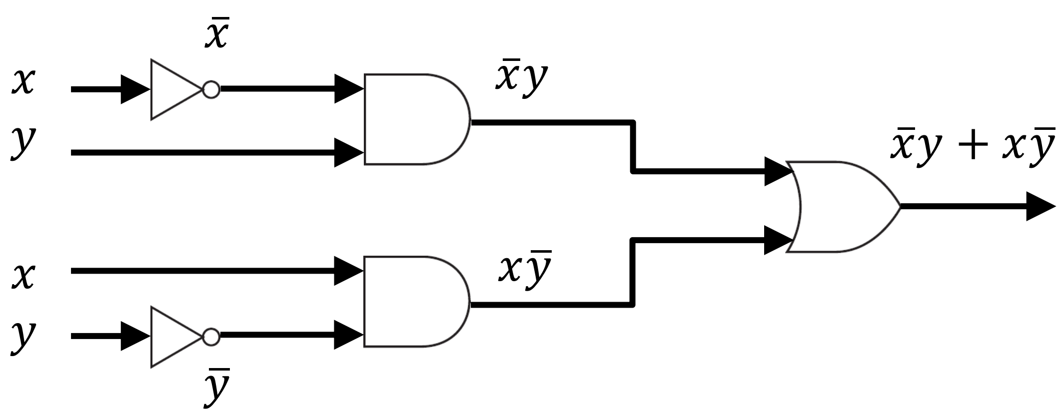




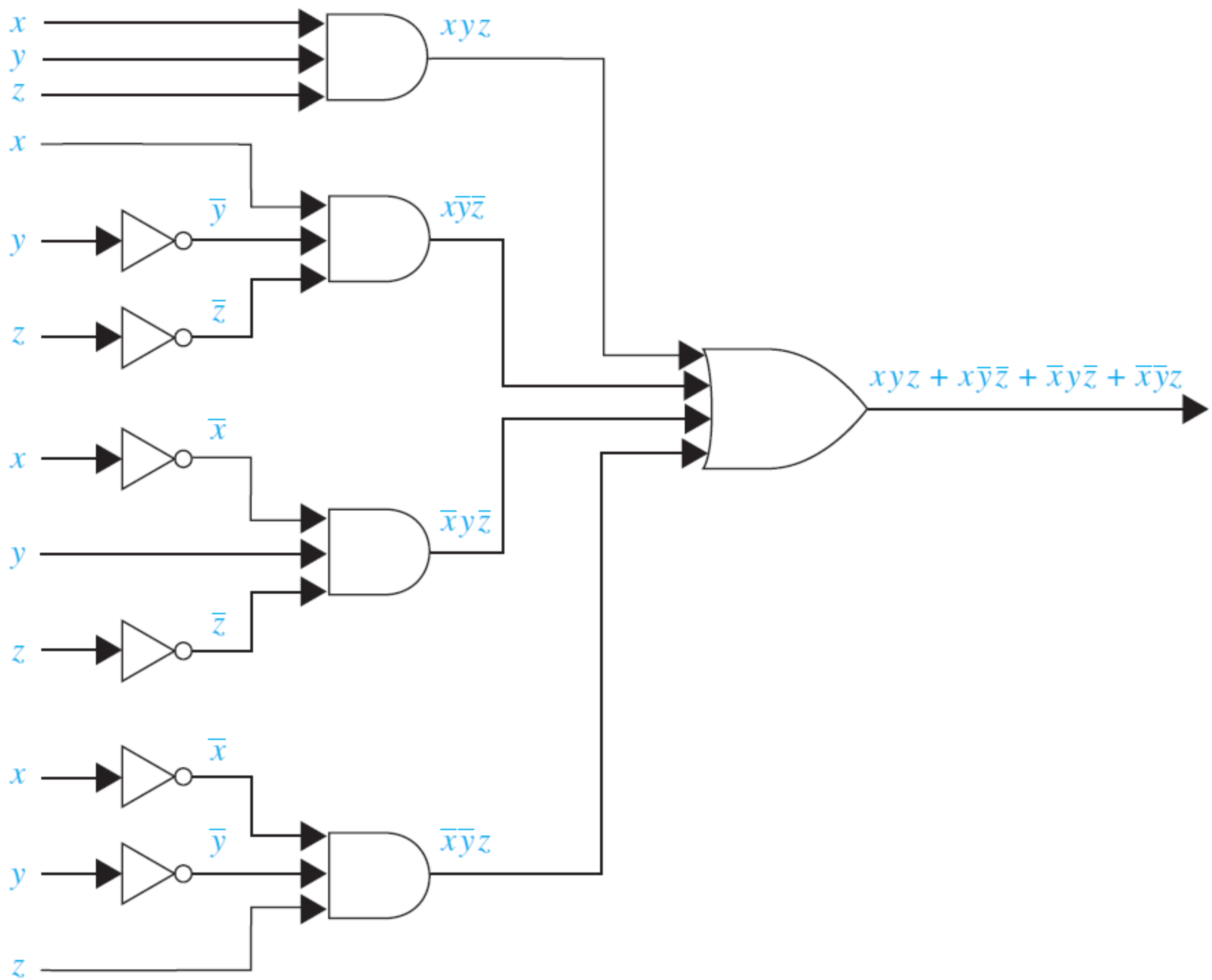
Designing a Two-Way Switch

- Task: Designing a two-way light switch
  - Input: switches x, y, which can be on (1) or off (0)
  - Output: light on/off as a Boolean function $F(x, y)$

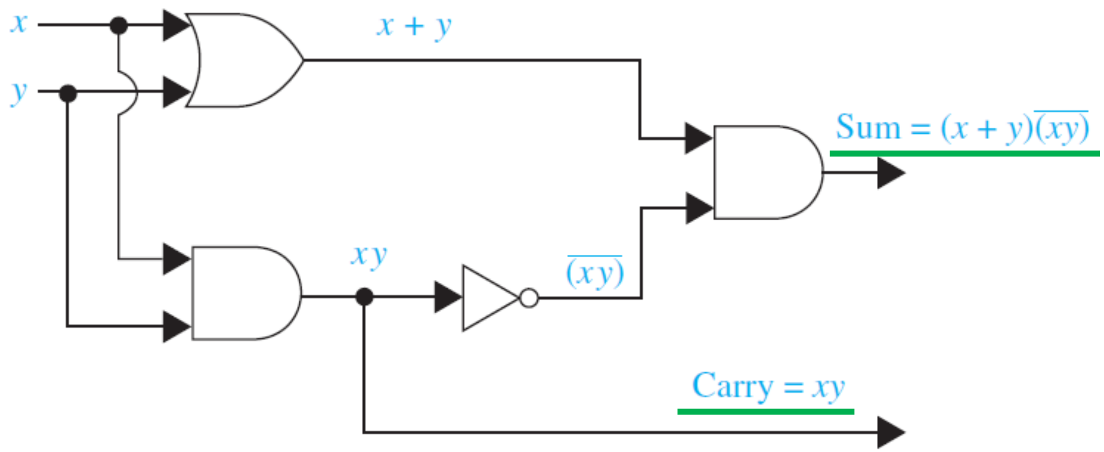
- Steps:
  - Assume that $x = 0$ means switch x is off. Same for y. [different from the textbook]
  - Assume when $x = y = 0$, $F(x, y) = 0$, light off.
  - From here, two possibilities: $x = 0, y = 1$ or $x = 1, y = 0$, $F(x, y) = 1$, light on.
  - From here, two possibilities: $x = y = 0$ or $x = y = 1$, light off.
  - We get $F(x, y) = 1$ when $x = 0, y = 1$ or $x = 1, y = 0$.
  - Using the sum-of-product construction, $F(x, y) = x\bar{y} + \bar{x}y$
  - Circuit:



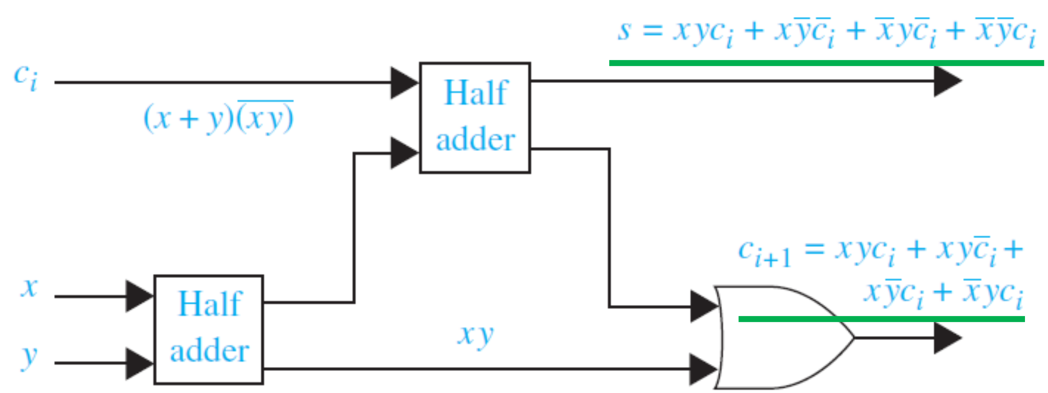
A Three-Way Switch



Half-Adder and Full-Adder



Half adder



Full adder