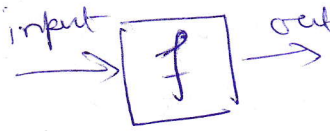
# Boolean Algebra

- Circuits in computers → input: either a 0 or a 1
  → output: also 0's and 1's.
- Circuits can be constructed using any basic element that has two different states.
  e.g. → switches that can be either in the on or the off position
  → optical devices that can either be lit or unlit.
- Basic rules of logic, first given by George Boole in 1854
- These were used to design circuits by Claude Shannon in 1938.
- These rules form the basis for Boolean algebra.
- Boolean algebra provides the operations of the rules for working with the set {0,1}.

- operations of a Circuit C
- BF f

## Steps to construct a circuit




- Write using $\cdot, +, -$

- Define the operation of a circuit by a Boolean fun. which specifies the value of output for each set of inputs
- Represent the Boolean fun. corresponding to a circuit by an expression built up using the basic operations of Boolean algebra.

## Three basic operations in Boolean algebra

- Complementation. (denoted with a bar) $\overline{0} = 1$, $\overline{1} = 0$
- the Boolean Sum (denoted by + or by OR) $1+1=1$, $1+0=1$, $0+1=1$, $0+0=0$
- the Boolean product (denoted by AND) $1 \cdot 1 = 1$, $1 \cdot 0 = 0$, $0 \cdot 1 = 0$, $0 \cdot 0 = 0$.

- the symbol $\cdot$ can be deleted if no danger of confusion, as in just writing algebraic products.

- Rules of precedence $\neg$ , $\cdot$ , $+$

  . all complements are computed first
  . followed by Boolean Products
  . followed by all Boolean sum.

- The complement, Boolean sum, and Boolean product correspond to the logical operations $\neg$, $\vee$, & $\wedge$ resp'ly.
  - 0 corresponds to F (false), 1 corresponds to T (true).
  - Equalities in Boolean algebra can be directly translated into equivalence of compound proposition.
  - Conversely, equivalence of compound propositions can be translated into equalities in Boolean algebra.

Example: find the value of $1 \cdot 0 + \overline{(0+1)}$

$1 \cdot 0 + \overline{(0+1)} = 1 \cdot 0 + \overline{1} = 1 \cdot 0 + 0 = 0 + 0 = 0$

$= 0 + 1$

$= 1$

Example: Translate $1 \cdot 0 + \overline{(0+1)} = 0$, ~~the equality~~ into a logical equivalence.

$(T \wedge F) \vee \neg (F \vee T)$

$\Leftrightarrow (T \wedge F) \vee (\neg F \wedge \neg T)$ $A \vee \bar{A} = -$

$\Leftrightarrow [(T \wedge F) \vee \neg F] \wedge [(T \wedge F) \vee \neg T]$

$\Leftrightarrow (T \vee \neg F) \wedge (F \vee \neg F) \wedge (T \vee T) \wedge (\neg T \vee F)$

$\Leftrightarrow (T \vee \neg F) \times F \wedge F \wedge (\neg T \vee F)$

$= F$

$\Leftrightarrow F$

**Example:** Translate the logical equivalence

$(T \wedge T) \vee \neg F \iff T$

into an identity in Boolean algebra.

$(T \wedge T) \vee \neg F \iff T$ is translated to

$1 \cdot 1 + \overline{0} = 1$

## Boolean expressions and Boolean functions

- $B = \{0, 1\}$
- $B^n = \{(x_1, x_2, \cdots, x_n) \mid x_i \in B, 1 \le i \le n\}$ → the set of all possible n-tuples of 0's & 1's.
- A variable $x$ is called a **Boolean variable** if it assumes values only from $B$.

  i.e. $x$ is either 0 or 1
- A function $f : B^n \to B$ is called a **Boolean fun^n of degree n**.

**Example:**

| $x$ | $y$ | $F(x, y)$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

→ a Boolean fun^n of degree 2

- **Boolean expression** (Recursive def^n) in the variables $x_1, x_2, \cdots, x_n$.

  (i) $0, 1, x_1, x_2, \cdots, x_n$ are Boolean expressions

  (ii) if $E_1$ and $E_2$ are Boolean expressions, then $E_1$, $E_1 E_2$ and $E_1 + E_2$ are Boolean expressions

- each Boolean ~~fun.~~ expression represents a Boolean fun
- each Boolean fun. can be represented by a Boolean expression.

not unique (in CNF, in DFN etc.)

Example: find the values of the Boolean fun. represented by $F(x,y,z) = xy + \bar{z}$

| $x$ | $y$ | $z$ | $xy$ | $\bar{z}$ | $F(x,y,z) = xy + \bar{z}$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 → |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 → |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 → |
| 0 | 0 | 0 | 0 | 1 | 1 |

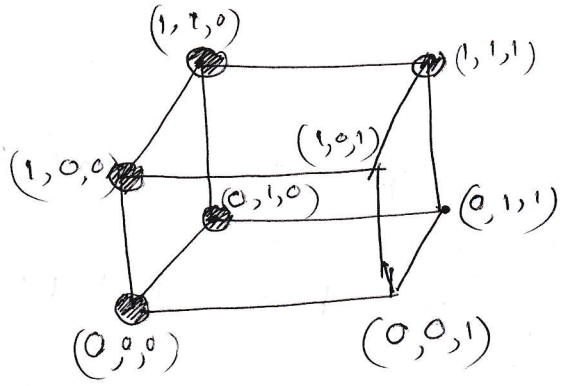
→ FAD p-DNF

$F = xyz + xyz' + xy'z' + x'yz' + x'y'z'$

$F = (x'+y+z')(x+y'+z')(x+y+z')$

p-CNF

- Graphically, a Boolean fun. distinguishes the vertices of the n-cube that corresponds to n-tuples of bits where the fun. has value 1




$F^d = (x'+y+z')(x'+y+z)(x+y'+z)(x+y+z')$

p-CNF

$(F^d)^d = \bar{F} = xy'z + \bar{x}yz + \bar{x}y\bar{z}$

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

- $f(x,y,z) = xy + \bar{z} : B^3 \to B$ can be represented by distinguishing the vertices that correspond to the ~~3-tuples~~ five 3-tuples $(0,0,0), (1,0,0), (1,1,0), (1,1,1)$ & $(0,1,0)$ where $f(x,y,z) = 1$.

- Boolean fun. $F$ and $G$ of $n$ variables are equal iff $F(b_1, b_2, \cdots, b_n) = G(b_1, b_2, \cdots, b_n)$ whenever $b_1, b_2, \cdots, b_n \in B = \{0, 1\}$.

- Two Boolean expressions that represent the same fun. are called equivalent.

e.g. $xy$, $xy+0$, $xy \cdot 1$ are equivalent.

- $\overline{F}$ → The Complement of the Boolean fun. $F$.

$$\overline{F}(x_1, x_2, \cdots, x_n) = \overline{F(x_1, x_2, \cdots, x_n)}$$

- $F+G$ → The Boolean sum
  $F \cdot G$ → The Boolean product
  } $F, G$ are Boolean fun. of degree $n$.

$$(F+G)(x_1, x_2, \cdots, x_n) = F(x_1, \cdots, x_n) + G(x_1, \cdots, x_n)$$

$$FG(x_1, \cdots, x_n) = F(x_1, \cdots, x_n)\, G(x_1, \cdots, x_n)$$

- # of different Boolean fun. of degree $n$
  $= 2^{2^n}$.

- $f(x_1, x_2, \cdots, x_n)$ a Boolean fun. of degree $n$ $\quad n=2$
  # of binary $n$-tuples $(x_1, x_2, \cdots, x_n)$ is $2^n$. $\quad 2^n = 4$
  - $f$ has value 0 or 1 corresponding to each of these $n$-tuples $(x_1, x_2, \cdots, x_n)$

$f : B^n \to B \Rightarrow$ A fixed $n$-tuple ↓ gives either 0 or 1. i.e. 2 possible $f$.

$x_i$ column | $f$ column.

$2^n$ rows { | $2^n$ rows { 1 → 0/1, 2 → 0/1, ⋮, $2^n$ → 0/1

# such $f$ columns $= 2^{2^n}$

$\vec{x} = (x_1, x_2, \cdots, x_n)$

$f$ column $\Rightarrow$ $\vec{x_1}$ | $\vec{x_2}$ | ... | $\vec{x_{2^n}}$

↓ ↓ ↓
$f$ $f$ $f$

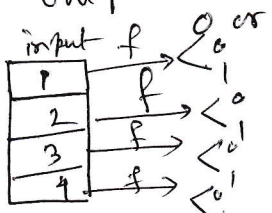
# The Boolean functions of degree 2

| $x$ | $y$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ | $F_9$ | $F_{10}$ | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | $F_{15}$ | $F_{16}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

## Identities of Boolean Algebra

1. $\overline{\overline{x}} = x$ (law of double complement)
2. $x + x = x$ (Idempotent laws)
   $x \cdot x = x$
3. $x + 0 = x$ (Identity laws)
   $x \cdot 1 = x$
4. $x + 1 = 1$ (Domination laws)
   $x \cdot 0 = 0$
5. $x + y = y + x$ (Commutative laws)
   $xy = yx$
6. $x + (y+z) = (x+y) + z$ (Associative laws)
   $x(yz) = (xy)z$
7. $x + yz = (x+y)(x+z)$ (Distributive laws)
   $x(y+z) = xy + xz$
8. $\overline{(xy)} = \bar{x} + \bar{y}$ (DeMorgan's laws)
   $\overline{(x+y)} = \bar{x}\,\bar{y}$
9. $x + xy = x$ (Absorption laws)
   $x(x+y) = x$
10. $x + \bar{x} = 1$ (Unit property), 11. $x\bar{x} = 0$ (Zero property)

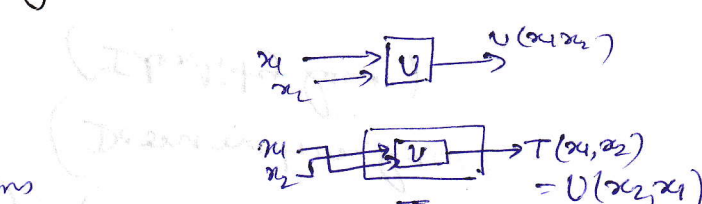
- Two variable n=
- ~~2 = 4 tuple~~
- # of 2 tuples = $2^n$ =
- input → f → 0 or 1
- i.e. each of 2-tupl as input of f outputs eith




↓ column is $2^n$ tuple each ent either 0 or 1 $2^n$ such column

- # of Boolean fns of Degree n

| n | # $(2^{2^n})$ |
|---|---|
| 1 | 4 |
| 2 | 16 → 12 patterns |
| 3 | 256 → 80 patterns to remember |
| 4 | 65,536 → 3984 patterns → 222 patterns (using other symmetries) |
| 5 | 4,294,967,296 → 37,333,248 |
| 6 | 18,446,744,073,709,551,616. |




to realize switching fns of 4 variables

Example: Show that the distributive law

$x(y+z) = xy + xz$ is valid.

| x | y | z | y+z | xy | xz | x(y+z) | xy+xz |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

- The above identities are particularly useful in simplifying the degree of circuits.
- Each identity can be proved using a truth table as above
- Each identity can be obtained by making the appropriate translations into the logical equivalences
  - each Boolean sum is a disjunction (replace 0 by F & 1 by T)
  - each Boolean product is a conjunction

**Example:** Translate the distributive law

$$x + yz = (x+y)(x+z)$$

into a logical equivalence.

**Soln:**

- change each Boolean variable into a propositional variable.

  $x, y, z$ to $p, q, r$ respectively.

- change each Boolean sum into a disjunction & each Boolean product into a conjunction.

$$p \vee (q \wedge r) \iff (p \vee q) \wedge (p \vee r).$$

↳ one of the distributive laws for propositional logic.

**Example:** Prove the absorption law

$$x(x+y) = x$$

using the other identities of Boolean algebra.

**Soln:**

$x(x+y) = xx + xy$ (dist. law)

↙or $= x + xy$ (~~ident~~ idempotent law).

$(x+0)(x+y)$

$= x + 0 \cdot y$

$= x + 0$

$= x$

$= \cancel{x+0}$

$= x \cdot 1 + xy$ (Identity law)

$= x(1+y)$ (Dist. law)

$= x \cdot 1$ (Domination law)

$= x$ (Identity law)

## Duality

- The dual of a Boolean expression is obtained by interchanging Boolean sums & Boolean products & interchanging 0's and 1's.

Example: $x(y+0)$ dual is $x+(y\cdot 1)$

$\bar{x}\cdot 1+(\bar{y}+z)$ dual is $(\bar{x}+0)\cdot(\bar{y}\cdot z)$

- The dual of a Boolean fun$^n$.

Same Boolean fun$^n$ $\xrightarrow{\text{can}}$ may have different Boolean expressions

F represented by a Boolean expression

is the fun$^n$. represented by the dual of this expression.

- Dual of F, denoted by $F^d$, does not depend on the particular Boolean expression used to represent F.

if $F_1 \Leftrightarrow F_2$

then $F_1^d \Leftrightarrow F_2^d$.

## Duality principle

An identity between functions represented by Boolean expressions remain valid when the duals of both sides of the identity are taken. This result is called duality principle and is very useful for obtaining new identities.

Example: Construct an identity from the absorption law $x(x+y)=x$ by taking duals.

Sol$^n$: $x+xy=x$ is the dual of $x(x+y)=x$ which is also an absorption law.

## Boolean algebra (Abstract defn).

A Boolean algebra is a set B with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation $^-$ such that these properties hold for all $x, y, z$ in B:

$$\left.\begin{aligned} x \vee 0 &= x \\ x \wedge 1 &= x \end{aligned}\right\} \text{ Identity laws}$$

$$\left.\begin{aligned} x \vee \bar{x} &= 1 \\ x \wedge \bar{x} &= 0 \end{aligned}\right\} \text{ Complement laws}$$

$$\left.\begin{aligned} (x \vee y) \vee z &= x \vee (y \vee z) \\ (x \wedge y) \wedge z &= x \wedge (y \wedge z) \end{aligned}\right\} \text{ Associative laws.}$$

$$\left.\begin{aligned} x \vee y &= y \vee x \\ x \wedge y &= y \wedge x \end{aligned}\right\} \text{ Commutative laws}$$

$$\left.\begin{aligned} x \vee (y \wedge z) &= (x \vee y) \wedge (x \vee z) \\ x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) \end{aligned}\right\} \text{ Distributive laws.}$$

- It is possible to prove many other laws that hold for Boolean algebra, such as idempotent & domination laws.

**Example:** i) $B = \{0, 1\}$ with $\overset{(+)}{OR}$ and $\overset{(\cdot)}{AND}$ operations & the complement $\overset{(-)}{\hat{}}$ operation.
→ Satisfies all ten properties of a Boolean algebra.

ii) The set of propositions in variables, with $\vee$ and $\wedge$ operators, F and T, and the negation operator
→ Satisfies all ten properties of a Boolean algebra

iii) The set of subsets of a universal set U with union & intersection operations, the empty set ($\phi$) & the universal set (U) & the set complementation operator → also a Boolean algebra

- To prove each of Boolean expressions, propositions, and sets is a Boolean algebra, we need only to prove results about abstract Boolean algebras

Example: lattice excluded.

- Boolean algebras may also be defined using the notion of lattice.

- A lattice is a partially ordered set in which every pair of elements $x, y$ has a least upper bound, denoted by $lub(x,y)$ & a greatest lower bound denoted by $glb(x,y)$.

- Given two elements $x$ and $y$ of $L$, we can define two operations $\vee$ and $\wedge$ on pairs of elements of $L$ by

$$x \vee y = lub(x,y) \quad \text{and} \quad x \wedge y = glb(x,y)$$

- for a lattice $L$ to be a Boolean algebra, it must be
  (i) Complemented
  (ii) distributive

- for a lattice to be complemented, it must have a least element 0 and a greatest element 1.

- for every element $x$ of the lattice, there must exist an element $\bar{x}$ s.t.

$$x \vee \bar{x} = 1$$
$$x \wedge \bar{x} = 0$$

- for a lattice to be distributive, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
for every $x, y, z$ in $L$.

# Representing Boolean functions

practically important
Two problems ~~addressed~~ in circuit design

~~How to represent a Boolean fun~~
How to find a Boolean expression that represents a a given Boolean fun.

Is there a smaller set of operators that can be used to represent all Boolean fun.?

Given the values of a Boolean fun., how to find a Boolean expression that represents this Boolean fun.

↓

Every Boolean fun. can be represented by a Boolean sum of products of the variables & their complements

↓

Every Boolean fun. can be represented using three Boolean operators, $-$, $+$ and $\cdot$.

↓

all Boolean fun can be represented by ~~only~~ using only one operator.

Illustrative Example:

| x | y | z | F | G |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1- |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1- | 0 |
| 1 | 1 | 0 | 0 | 1- |
| 1 | 1 | 1 | 0 | 0 |

$F = x\bar{y}z$

$G = \bar{x}yz + x\bar{y}z.$

disjunctive normal form
or
Sum of product form.

## Procedure of constructing a Boolean expression representing a function with given values

- Why it works: Each combination of the values of the variables for which the function has the value 1 leads to a Boolean product of the variables or their complements.

**Def$^n$:** A literal is a Boolean variable or its complement.

A minterm of the $n$ Boolean variables $x_1, x_2, \dots, x_n$ is a Boolean product $y_1 y_2 \cdots y_n$ where

$$y_i = x_i \text{ or } y_i = \bar{x}_i.$$

- Hence, a minterm is a product of $n$ literals, with one literal for each variable.
- A minterm has the value 1 for one and only one combination of values of its variables.
- More precisely, the minterm $y_1 y_2 \cdots y_n$ is 1 iff each $y_i = 1$, and this occurs iff

$$x_i = \begin{cases} 1 & \text{when } y_i = x_i \\ 0 & \text{when } y_i = \bar{x}_i \end{cases}$$

**Example:** minterm that equals 1 if $x_1 = x_3 = 0$ & $x_2 = x_4 = x_5 = 1$.

$$\bar{x}_1 x_2 \bar{x}_3 x_4 x_5.$$

- Boolean sum of distinct minterms

$m_{i_1} + m_{i_2} + \cdots + m_{i_k}$ — yields a Boolean expression with a specified set of values.

has value → 1 if $m_{i_l} = 1$ for at least one $i_l \in \{i_1, i_2, \ldots, i_k\}$

~~if all other $m_i = 0$~~

~~for $i \in \{i_1, i_2, \ldots, i_k\}$ $\forall i$~~

→ 0 if $m_i = 0$ $\forall i \in \{i_1, i_2, \ldots, i_k\}$.

→ Boolean fun$^n$.

- Given a Boolean fun$^n$, ~~form a Boolea~~

  - form a Boolean sum of minterms that has the value 1 when this Boolean fun$^n$ has the value 1, and has the value 0 when this Boolean fun$^n$ has the value 0.

  - The minterms in this Boolean sum correspond to those combinations of values for which the fun$^n$ has the value 1.

→ DNF or sum-of-product expansion of the Boolean fun$^n$.

Example — Sum-of-product ~~form~~ expansion of the fun. $F(x,y,z) = (x+y)\bar{z}$

Approach 1 (Using Boolean identities)

$$F(x,y,z) = (x+y)\bar{z}$$
$$= x\bar{z} + y\bar{z} \quad \text{(Dist.)}$$
$$= x\bar{z}(y+\bar{y}) + y\bar{z}(x+\bar{x}) \quad \text{(Unit)}$$
$$= xy\bar{z} + x\bar{y}\bar{z} + xy\bar{z} + \bar{x}y\bar{z} \quad \text{(Dist.)}$$
$$= xy\bar{z} + x\bar{y}\bar{z} + \bar{x}y\bar{z}. \quad \text{(Idempotent)}$$

Approach 2

| $x$ | $y$ | $z$ | $x+y$ | $\bar{z}$ | $F=(x+y)\bar{z}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 ✓ |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 ✓ |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 ✓ |
| 1 | 1 | 1 | 1 | 0 | 0 |

$$f(x,y,z) = \bar{x}y\bar{z} + x\bar{y}\bar{z} + xy\bar{z}.$$

DNF

$\neg(x \wedge x)$

- Also possible to find a Boolean expression that represents a Boolean fun. by taking a Boolean product of Boolean sum.

→ conjunctive normal form (CNF) or product-of-sum expansion of the fun.

CNF can be obtained from DNF by taking duals

## Functional Completeness

- Every Boolean fun. can be expressed as a Boolean sum of minterms.
- Each minterm is the Boolean product of Boolean variables or their complements.

→ every Boolean fun. can be represented using the set of operators $\{\cdot, +, \bar{}\}$.

→ we say $\{\cdot, +, \bar{}\}$ is functionally complete.

Q> Can we get a smaller set of functionally complete operators?

- $x+y = \overline{\bar{x}\cdot\bar{y}}$ ⇒ $\{\cdot, \bar{}\}$ functionally complete.
- $xy = \overline{\bar{x}+\bar{y}}$ ⇒ $\{+, \bar{}\}$ functionally complete.
- But $\{+, \cdot\}$ is not functionally complete as it is impossible to express the Boolean fun.

$$F(x) = \bar{x}$$

using + & · operators.

Q> Can we find a smaller set of functionally complete operators? → A set containing only one operator?

YES
- NAND operator ↑
- NOR operator ↓

Both the sets $\{\uparrow\}$ & $\{\downarrow\}$ are functionally complete.

## NAND ($\uparrow$)

$1 \uparrow 1 = 0,\ 1 \uparrow 0 = 0 \uparrow 1 = 0 \uparrow 0 = 1$

Not AND

| x | y | x.y | $x \uparrow y$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## NOR ($\downarrow$)

$1 \downarrow 1 = 0 = 1 \downarrow 0 = 0 \downarrow 1,\quad 0 \downarrow 0 = 1$

Not OR

| x | y | x+y | $x \downarrow y$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 |

## To prove $\{\uparrow\}$ is functionally complete.

- $\{\cdot, \bar{}\}$ is functionally complete, we need to show that both $\cdot$ & $\bar{}$ can be expressed using just $\uparrow$ operator.

$\bar{x} = x \uparrow x$

$xy = (x \uparrow y) \uparrow (x \uparrow y)$

| x | x | x.x | $x \uparrow x$ | $\bar{x}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

| x | y | $x \uparrow y$ | $x \uparrow y$ | $(x \uparrow y) \uparrow (x \uparrow y)$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

**Exercise** Prove that $\{\downarrow\}$ is functionally complete.

## Logic Gates

- Boolean algebra is used to model the circuits of electronic devices.
- each input & each output $\in \{0, 1\}$.
- Computer or other electronic device, is made up of a number of circuits.
- each circuit can be designed using the rules of Boolean algebra.
- Gates: Basic elements of circuits.

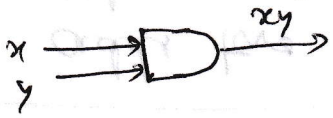
~~<-- implements~~

- each type of gate implements a Boolean operation
- Circuits are designed using different gates & applying the rules of Boolean algebra to perform variety of tasks.
- Combinational circuits / Gating network
  - give outputs that depend only on the input, not on the current state of the circuit.
  - has no memory capabilities.
  - Uses mainly three types of elements.

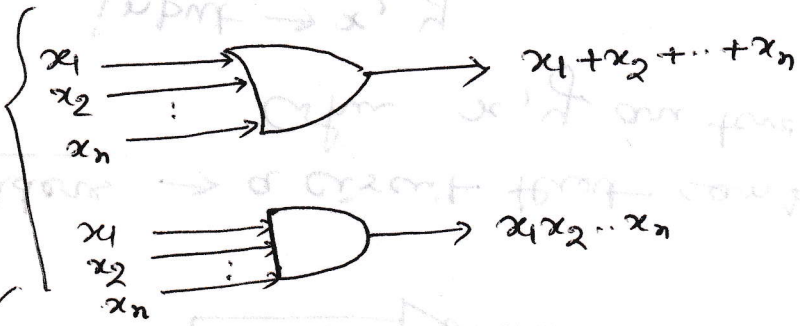
- Inverter


- OR gate


- AND gate




. Permitting multiple inputs to OR & AND gates.

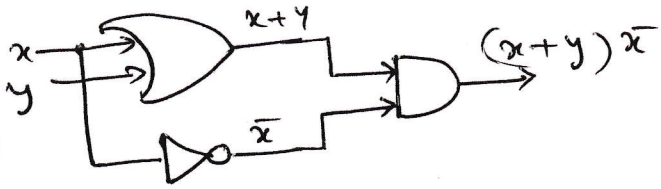



↳ Gates with n-inputs.

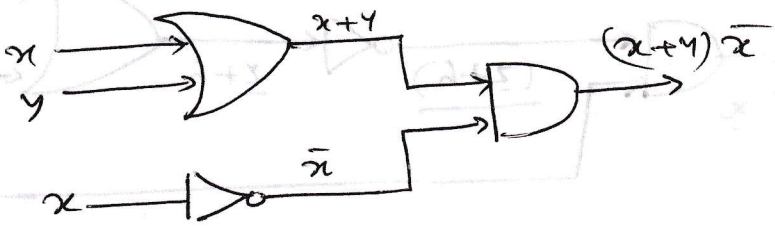
Example (Combinations of Gates).

Construct circuit that produce

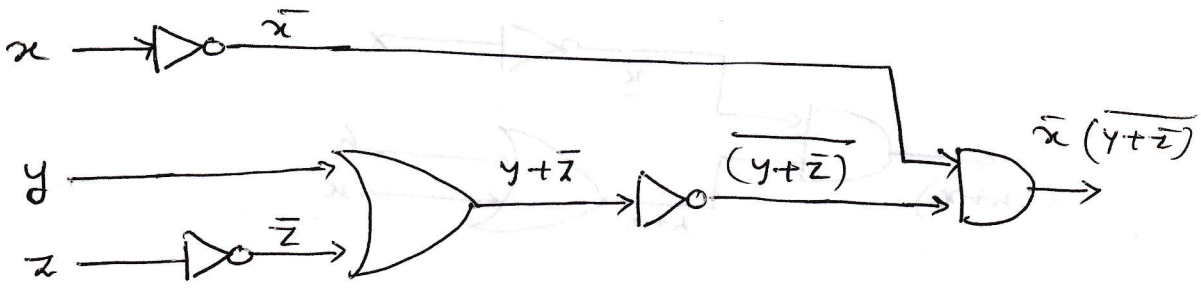
a) $(x+y)\bar{x}$



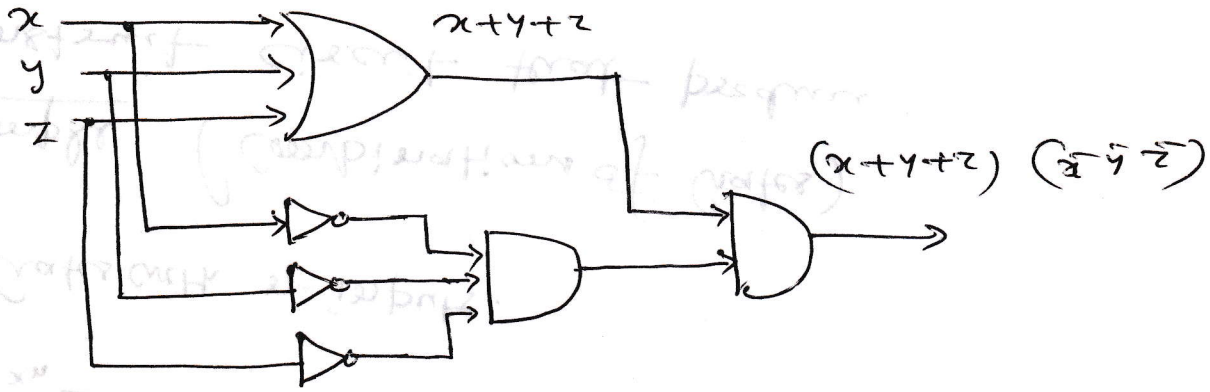

or



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




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




**Adders** → a circuit that can be used to find $x+y$, when $x, y$ are two bits.

input → $x$, $y$
output → $s$, $c$ $\quad s$ = sum, $c$ = carry

multiple output circuit → more than one output

---

Half-adder → it adds two bits, without considering a carry from a previous addition.

| $x$ | $y$ | $s$ | $c$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

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

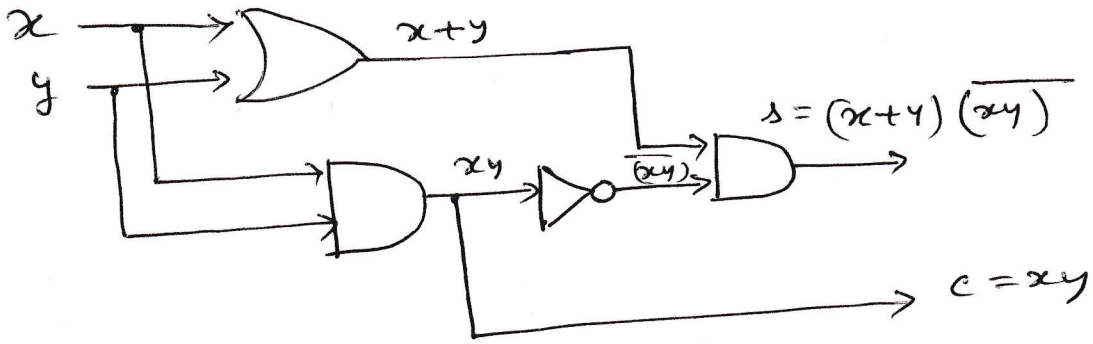
as $(x+y)(\bar{x}+\bar{y})$
$= x\bar{x} + xy + y\bar{x} + y\bar{y}$
$= xy + y\bar{x}$

$$c = xy$$

## The Half adder

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

$$c = xy$$




**Full adder** → computes the sum bit of the carry bit when two bits & a carry bit are added.

inputs → x, y, carry $c_i$

outputs → sum bit s, the new carry $c_{i+1}$.

| x | y | $c_i$ | s | $c_{i+1}$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$$s = \bar{x}\bar{y}c_i + \bar{x}y\bar{c}_i + x\bar{y}\bar{c}_i + xyc_i$$

$$c_{i+1} = \bar{x}\bar{y}c_i + x\bar{y}c_i + \bar{x}yc_i + xyc_i$$

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

($\bar{x}y + x\bar{y}$: Half adder sum)

$$s = (\bar{x}\bar{y} + xy)c_i + (\bar{x}y + x\bar{y})\bar{c}_i$$

$$= \overline{(\bar{x}y + x\bar{y})}\,c_i + (\bar{x}y + x\bar{y})\bar{c}_i$$

($\bar{x}y + x\bar{y}$: Half adder sum)