Chapter 1
Logic of
Compound Statement

1.1 Introduction

Logic

- The discipline that deals with the methods of reasoning.
- Provides rules and techniques for determining whether a given argument is valid.
- Use to prove theorems and verify the correctness of programs.

1.2 Logical Form and Logical Equivalence

Statement / Proposition

- A declarative sentence that is *either* true *or* false, but **NOT BOTH**.
- Usually represented by lowercase alphabet, such as *p*, *q*, *r*, *s* and so on.
- E.g. *p* : The sun is shining today.

  q : It is cold.
- *p*, *q*, *r*, *s* : propositional variables.

Example 1:

Which of the following are statements?

i)	$2 + 3 = 5$	Statement
ii)	The earth is round.	Statement
iii)	$3 - x = 5$	Not Statement
iv)	Do you speak English?	Not Statement
v)	Take two aspirins.	Not Statement

Compound Statements

Obtained by

a) *Negation* of a statement, $\sim p$ or $\neg p$

- "not p" or "it is not the case that p".
- Opposite value from p, if p is true, $\sim p$ is false; if p is false, $\sim p$ is true.
- From the previous e.g.,

  $\sim p$: The sun is not shining today.

  $\sim q$: It is not cold.

b) *Combine* two or more statements using following *Logical Connectives*.

i) Conjunction, $\wedge$

- Denoted as $p \wedge q$, "*p* and *q*".
- True when, and only when, BOTH *p* and *q* are true.
- If *either p or q* is false, or *both* are false, $p \wedge q$ is false.
- From the previous e.g.,

  $p \wedge q$: The sun is shining today <u>and</u> it is cold.

ii) Disjunctive

- Inclusive, $\vee$
  - Denoted as $p \vee q$, "*p* or *q*".
  - True when *at least* one of p or q is true and is false only when BOTH p and q are false.
- Exclusive, $\underline{\vee}$
  - Denoted as $p \underline{\vee} q$, $p \oplus q$, p XOR q.
  - True if one or the other but *not both* of the statements is true.

- E.g.

  $p \vee q$: The sun is shining, or it is cold.

  $p \underline{\vee} q$: The sun is shining today, or it is cold, but not both.

iii) NAND, { | }

- “and” followed by “not”, denoted as $p|q$.
- $p|q \equiv \sim(p \wedge q)$
- { | } : *Scheffer Stroke*.

iv) NOR {↓}

- "or" followed by "not", denoted as $p \downarrow q$.
- $p \downarrow q \equiv \sim (p \vee q)$
- {↓} : *Pierce Arrow*.

Note:

- Be careful for the order of operation.
- E.g.
  - $\sim p \wedge q$: conjunction of $\sim p$ and q
    (negation first, conjunction later)
  - $\sim(p \wedge q)$: negation of the conjunction of p and q
    (conjunction first, followed by negation)
- Strictly speaking, “not” is not a connective, since it does not join 2 statements, so $\sim p$ is not really a compound statement.

Example 2:

Form the negation, conjunction, and disjunctive for the following statements.

i) p: $2 + 3 > 1$

$\therefore \sim p$: $2 + 3 \leq 1$

ii) p: It is snowing.

q: I am cold.

$\therefore p \wedge q$: It is snowing AND I am cold.

iii) p: $2 < 3$.

q: $-5 > -8$

$\therefore\ p \wedge q$: $2 < 3$ AND $-5 > -8$

iv) p: 2 is a positive integer.

q: 2 is an odd number.

$\therefore\ p \vee q$: 2 is a positive integer OR 2 is an odd number.

v) p: Kelly ate the pie.

q: Charles ate the pie.

$\therefore\ p \underline{\vee}\ q$: Kelly ate the pie OR Charles ate the pie BUT NOT BOTH.

Statement Form / Propositional Form

- Expression that made up of statement variables (*p*, *q*, *r*) and logical connectives ($\wedge$, $\vee$, ~) that becomes a statement when actual statements are substituted for the component statement variable.

Truth Table

- A table displays the truth values of different compound statements. Use “0” or “F” for false statement; “1” or “T” for true statement.
- Use “0” or “F” for false statement; “1” or “T” for true statement.

Example 3:

Construct a truth table for $\sim p$, $p \wedge q$, $p \vee q$, $p \veebar q$, $p \mid q$, and $p \downarrow q$.

p	q	$\sim p$	$p \wedge q$	$p \vee q$	$p \veebar q$	$p \mid q$	$p \downarrow q$
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	1	0
1	1	0	1	1	0	0	0

Example 4:

Construct the truth table for

$(p \wedge q) \vee \sim(p \vee q)$.

p	q	$p \wedge q$	$p \vee q$	$\sim(p \vee q)$	$(p \wedge q) \vee \sim(p \vee q)$
0	0	0	0	1	1
0	1	0	1	0	0
1	0	0	1	0	0
1	1	1	1	0	1

Example 5:

Construct the truth table for $(p \downarrow q) \downarrow r$.

p	q	r	$p \vee q$	$p \downarrow q$	$(p \downarrow q) \vee r$	$(p \downarrow q) \downarrow r$
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	1	0	1	0	0	1
0	1	1	1	0	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

Logical Equivalence

- Two statement forms, P and Q, are logically equivalence if, and only if, they have identical truth values for each possible substitution of statements for their statement variables.
- Denoted as $P \equiv Q$ or $P \Leftrightarrow Q$.
- Statement P is true (respectively, false) if and only if the statement Q is true (respectively, false).

Steps to Test Two Statement Forms are Logically Equivalence

1) Construct the truth table for *P*.
2) Construct the truth table for *Q.*
3) Check each combination of truth values of the statement variables to see whether the truth value of *P* is the same as the truth value of *Q*.

a) If in *each* row the truth value of *P* is the same as the truth value of *Q*, then $P \equiv Q$.
b) If in *some* rows *P* has a different truth value from *Q*, then *P* and *Q* are not logically equivalence.

Example 6:

Show that $p \veebar q \equiv (p \vee q) \wedge \sim(p \wedge q)$.

p	q	$p \veebar q$	$p \vee q$	$p \wedge q$	$\sim(p \wedge q)$	$(p \vee q) \wedge \sim(p \wedge q)$
0	0	0	0	0	1	0
0	1	1	1	0	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	0

Example 7:

Check whether $\sim(p \wedge q)$ and $\sim p \wedge \sim q$ are logically equivalence or not.

p	q	$p \wedge q$	$\sim(p \wedge q)$	$\sim p$	$\sim q$	$\sim p \wedge \sim q$
0	0	0	1	1	1	1
0	1	0	1	1	0	0
1	0	0	1	0	1	0
1	1	1	0	0	0	0

They are not logically equivalent

Tautology

- A statement form that is always true regardless of the truth values of individual statements substituted for its statement variables.
- Denoted as *t*.

Example: The dog is always brown, or the dog is not brown

Contradiction

- A statement form that is always false regardless of the truth values of individual statements substituted for its statement variables.
- Denoted as *c*.

Example: He is king and he is not a king.

Law of the Logical Equivalences

- Given any statement variables *p*, *q*, and *r*, a tautology *t*, and a contradiction *c*, the following logical equivalences hold:

1) Commutative laws:

- $p \wedge q \equiv q \wedge p$
- $p \vee q \equiv q \vee p$

2) Associative laws:

- $(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
- $(p \vee q) \vee r \equiv p \vee (q \vee r)$

3) Distributive laws:

- $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

4) Identity laws:

- $p \wedge t \equiv p$ p AND tautology (true) is p
- $p \vee c \equiv p$ p OR contradiction (false) is p

5) Negation/Inverse laws:

- $p \vee {\sim}p \equiv t$ p OR NOT p is always true
- $p \wedge {\sim}p \equiv c$ p AND NOT p is always false

6) Double Negative law:

- ${\sim}({\sim}p) \equiv p$ NOT NOT p is p

7) Idempotent laws:

- $p \wedge p \equiv p$ p AND p is p
- $p \vee p \equiv p$ p OR p is p

8) De Morgan's laws:

- $\sim(p \wedge q) \equiv \sim p \vee \sim q$
- $\sim(p \vee q) \equiv \sim p \wedge \sim q$

If you don't have a cat and a dog, it means you don't have a cat, or you don't have a dog.

If you don't have a cat or a dog, it means you don't have a cat, and you don't have a dog.

9) Universal Bound / Domination laws:

- $p \vee t \equiv t$ p OR tautology (true) is true
- $p \wedge c \equiv c$ p AND contradiction (false) is false

10) Absorption laws:

- $p \vee (p \wedge q) \equiv p$ p OR p AND q is p, if P is false, p AND q is false
- $p \wedge (p \vee q) \equiv p$ p AND p OR q is p, if p is false, p AND (...) is false

11) Negation of *t* and *c* :

- $\sim t \equiv c$ Not tautology (true) is contradiction (false)
- $\sim c \equiv t$ Not contradiction (false) is tautalogy (true)

Example 8:

Use De Morgan's Law to write negation for the following statement:

Sam swims on Thursday and Kate plays tennis on Saturdays.

Solution:

Let *p*: Sam swims on Thursday,

and *q*: Kate plays tennis on Saturdays.

Given $p \wedge q$.

By De Morgan's law,

Negation statement:

Sam does not swims on Thursday or Kate does not plays tennis on Saturdays

Example 9:

Negate the following $(p \wedge q) \vee (p \vee q)$.

Solution:

$$
\begin{aligned}
\sim[(p \wedge q) \vee (p \vee q)] &\equiv \sim(p \wedge q) \wedge \sim(p \vee q) \text{ [DeMorg]} \\
&\equiv (\sim p \vee \sim q) \wedge (\sim p \wedge \sim q) \text{ [DeMorg]} \\
&\equiv (\sim p \vee \sim q) \wedge \sim p \wedge \sim q \quad \text{[Associative]} \\
&\equiv \sim p \wedge \sim q \text{ [Absorption of laws]}
\end{aligned}
$$

Example 10:

Prove Negation law and Universal Bound law.

Note: t is always true. c is always false

p	$\sim p$	$p \vee \sim p$	$p \wedge \sim p$	t	c	$p \vee t$	$p \wedge c$
0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0

Negation laws:

$p \vee \sim p \equiv t$

$p \wedge \sim p \equiv c$

Universal Bound laws:

$p \vee t \equiv t$

$p \wedge c \equiv c$

Example 11:

Use theorem above to simplify the logical equivalence:

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

Solution:

$\sim(\sim((p \vee q) \wedge r) \vee \sim q)$

1.3 Conditional Statements

- If p and q are statement variables, the conditional of q by p is "if p then q" or "p implies q".
- Denoted as $p \to q$.

- Truth values:

p	q	$p \to q$
0	0	1
0	1	1
1	0	0
1	1	1

Sort of a promise statement:
p = promise
q = delivery
If I promise, and I didn't deliver, I "broke" that promise.
If I didn't promise, it doesn't matter if I deliver or not.

Notes:

- Alternatively, $p \rightarrow q$ can also said as

  i) if p then q If mars is flat, then I am king

  ii) p is sufficient for q

  iii) p is a sufficient condition for q

  iv) p only if q p is true only if q is true

  v) q is necessary for p

  vi) q is a necessary condition for p

  vii) q if p

  viii) q unless $\sim p$

- $p \to q \equiv \sim p \vee q$
  (can show using truth table)

p	q	$p \to q$	$\sim p$	$\sim p \vee q$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	1	0	1

Example 12:

contradiction

Construct a truth table for $(p \wedge \sim q) \to c$.

p	q	$\sim q$	$p \wedge \sim q$	c	$(p \wedge \sim q) \to c$
0	0	1	0	0	1
0	1	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1

Example 13:

Let p: I weigh more than 120 pounds.

q: I shall enroll in an exercise class.

Translate $p \rightarrow q$ in English sentence.

Solution:

IF I weigh more than 120 pounds, THEN I shall enroll in an exercise class

Example 14:

Note: $\Leftrightarrow$ = $\equiv$ = $\leftrightarrow$

Show that $\sim((p \vee q) \rightarrow r) \Leftrightarrow (p \vee q) \wedge \sim r$.

p	q	r	$\sim r$	$p \vee q$	$(p \vee q) \rightarrow r$	$\sim((p \vee q) \rightarrow r)$	$(p \vee q) \wedge \sim r$
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
1	0	0	1	1	0	1	1
1	0	1	0	1	1	0	0
1	1	0	1	1	0	1	1
1	1	1	0	1	1	0	0

same

Negation of a Conditional Statement

- The negation of "if p then q" is logically equivalent to "p and not q".
- $\sim(p \to q) \equiv p \wedge \sim q$

p = promise, q = deliver

$p \to q$ is a promise.

- If not delivered then false.

$\sim(p \to q)$ is a "broken promise".

- If delivered then false.
- If not promised also false, because its not a promise to begin with
- Only way to false is to promise (p) AND not deliver (~q)

Example 15:

Prove the above statement:

i) $\sim(p \to q) \equiv \sim(\sim p \vee q)$

Alternative representation of implication

$\equiv p \vee \sim q$

De Morgan's, Double negation

ii) Truth table:

p	q	$p \to q$	$\sim(p \to q)$	$\sim q$	$p \wedge \sim q$
0	0	1	0	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	1	1	0	0	0

same

Contrapositive of a Conditional Statement

- The contrapositive of a conditional statement of the form "if p then q" is "if $\sim q$ then $\sim p$".
- The contrapositive for $p \rightarrow q \equiv \sim q \rightarrow \sim p$.
- The conditional statement is logically equivalent to its contrapositive, i.e. $p \rightarrow q \equiv \sim q \rightarrow \sim p$.

Example 16:

Prove the above contrapositive statement.

p	q	$p \to q$	$\sim q$	$\sim p$	$\sim q \to \sim p$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	1	1	0	0	1

same

Converse of a Conditional Statement

- The converse of $p \rightarrow q$ (if p then q) is $q \rightarrow p$ (if q then p).
- The conditional statement is not logically equivalent to its converse.

Converse of a Conditional Statement

- The converse of $p \to q$ (if p then q) is $q \to p$ (if q then p).
- The conditional statement is not logically equivalent to its converse.

If I deliver, doesn't mean I promise

Inverse of a Conditional Statement

- The inverse of $p \to q$ (if p then q) is $\sim p \to \sim q$ (if $\sim p$ then $\sim q$).
- The conditional statement is not logically equivalent to its inverse.

If I didn't promise, doesn't mean I didn't deliver

Example 17:

Write the contrapositive, converse, and inverse for the following implication.

“If today is Labour Day, then tomorrow is Thursday.”

Solution:

Let p: Today is Labour Day.

q: Tomorrow is Thursday.

Given $p \to q$.

Contrapositive ($\sim q \to \sim p$): If tomorrow is NOT thursday, then today is NOT labour day.

Solution:

- Converse:
- If tomorrow is Thursday, then today is Labour Day.

- Inverse:

If today is not Labour Day, then tomorrow is not Thursday.

Notes:

- A conditional statement and its *converse* are NOT logically equivalent.
- A conditional statement and its *inverse* are NOT logically equivalent.

- The converse and the inverse of a conditional statement are logically equivalent to each other, e.g. for $p \rightarrow q$,

  converse: $q \rightarrow p$

  inverse: $\sim p \rightarrow \sim q$

  According to contrapositive,

  the contrapositive of $q \rightarrow p$ is $\sim p \rightarrow \sim q$,

  therefore $q \rightarrow p \equiv \sim p \rightarrow \sim q$.

Biconditional

- The biconditional of p and q is "p, if and only if, q" means both p and q have the same truth values.
- Denoted as $p \leftrightarrow q$.

p	q	$p \leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

- "*p*, if and only if (iff), *q*" means the same as saying both "*p* if *q*" and "*p* only if *q*".
- $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$
- Consequently, *p* is a necessary and sufficient condition for *q*.

Only-if Biconditional

- *p* only if *q* means

  i) if *p* then *q*

  ii) if not *q* then not *p*

Notes:

- p only if q is not same as p if q,
- Order of operations:

  1) $\sim$

  2) $\wedge$, $\vee$

  3) $\rightarrow$, $\leftrightarrow$
- In informal language, simple conditionals are often used to mean biconditionals.

1.4 Duality

- Let s be a statement consists of $\wedge$ and $\vee$, then the duality of the statement s, s^d, is a statement that obtained when $\wedge$ is replaced by $\vee$, $\vee$ is replaced by $\wedge$, t is replaced by c, and c is replaced by t.
- E.g. if s: $p \vee \sim p$, then s^d: $p \wedge \sim p$.

Notes:

- The statement $\sim p$ is remain unchanged when s is changed to s^{d}.
- The dual of a statement p is p, and the dual of $\sim p$ is $\sim p$.

Example 18:

Write the dual of the statement

s: $(p \wedge \sim q) \vee (r \wedge t)$.

s^d = (p ∨ ~q) ∧ (r ∨ c)

Duality Principle

- If $s \Leftrightarrow r$, then $s^d \Leftrightarrow r^d$.

Example 19:

Prove the absorption law: $p \vee (p \wedge q) \equiv p$, and write its dual.

- According to duality principle, the dual is $p \wedge (p \vee q) \equiv p$

1.5 Normal Forms

Disjunctive Normal Form

- Comparison of two logical expressions would be convenient if both expressions can be converted into some standard form.

- The normal forms are the statements that are expressed as

  i) the disjunctive of several terms, each of which is the conjunction of some simple statements and the negations of some simple statements, such as

  $(p \wedge q) \vee (p \wedge \sim q) \vee (\sim p \wedge q \wedge r)$; or

ii) the conjunction of several terms, each of which is the disjunction of some simple statements and the negations of some simple statements, such as

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

- Use "product" for conjunction, and "sum" for disjunction.
- An *elementary product*: conjunction of some simple statement variables and the negations of some simple statement variables, such as p, $p \wedge q$, $\sim q \wedge p \wedge \sim p$, $q \wedge \sim p$.
- An *elementary sum*: disjunction of some simple statement variables and the negations of some simple statement variables, such as p, $\sim q$, $p \vee q$, $p \vee \sim q \vee r$, $\sim p \vee \sim r$.

- A part of an elementary sum or product, which is itself an elementary sum or product, is called a *factor* of the original elementary sum or product.
- E.g. for $\sim q \wedge p \wedge \sim p$; $\sim q$, $p \wedge \sim p$, $\sim q \wedge p$ are some factors for the elementary sum.
- An expression, which is equivalent to a given logical expression that consists of a sum of elementary products, is called a *disjunctive normal form* of the given expression.

Procedure to obtain a disjunctive normal form:

1) Change $\rightarrow$ and $\leftrightarrow$ using equivalent expressions in terms of $\wedge$, $\vee$ and ~.

2) For ~ which appears in front of a compound statement enclosed in parentheses, use De Morgan's laws to bring in the ~.

3) If the expression is still not in disjunctive normal form because of some parts are products of sums, use distributive laws repeatedly to convert products of sums into sums of products.

Example 20:

Express $p \wedge (p \to q)$ in disjunctive normal form.

Solution:

$p \wedge (p \to q) \equiv$ p ∧ ~p ∨ p ∧ q (after simplification)

Notes:

- To simplify workings, write pq for $p \wedge q$, $p + q$ for $p \vee q$, $\overline{p}$ for $\sim p$, 1 for t, 0 for c.
- Then the working above may be written as

Notes:

- To simplify workings, write pq for $p \wedge q$, $p + q$ for $p \vee q$, $\overline{p}$ for $\sim p$, 1 for t, 0 for c.
- Then the working above may be written as

$p\overline{p} + pq$.

Example 21:

Write $\sim[(p \vee q) \leftrightarrow (p \wedge q)]$ in disjunctive normal form.

$\sim[(p \vee q) \leftrightarrow (p \wedge q)]$

$\equiv \sim[((p \vee q) \rightarrow (p \wedge q)) \wedge ((p \wedge q) \rightarrow (p \vee q))]$

$\equiv \sim[(\sim(p \vee q) \vee (p \wedge q)) \wedge (\sim(p \wedge q) \vee (p \vee q))]$

$\equiv [\sim(\sim(p \vee q)) \vee (p \wedge q)] \vee \sim[(\sim(p \wedge q) \vee (p \vee q))]$

$\equiv [(p \vee q) \wedge \sim(p \wedge q)] \vee [(p \wedge q) \wedge \sim(p \vee q)]$

$\equiv$ Refer to PDF file

Conjunctive Normal Form

- A similar procedure can be used to convert a given expression into a product of elementary sums.
- An expression, which is equivalent to a given expression that consists of a product of elementary sums, is called a *conjunctive normal form* of the given expression.

Example 22:

Obtain a conjunctive normal form for

i) $p \wedge (p \rightarrow q)$

ii) $\sim(p \vee q) \leftrightarrow (p \wedge q)$

$\equiv [\sim(p \vee q) \rightarrow (p \wedge q)] \wedge [(p \wedge q) \rightarrow \sim(p \vee q)]$

$\equiv [(p \vee q) \vee (p \wedge q)] \wedge [\sim(p \wedge q) \vee \sim(p \vee q)]$

$\equiv [(p \vee q) \vee (p \wedge q)] \wedge [(\sim p \vee \sim q) \vee (\sim p \wedge \sim q)]$

$\equiv$ Refer to PDF file

- A given expression is identically true, i.e. a tautology, if every elementary sum in its conjunctive normal form is identically true.
- The following rules apply to elementary sums and products:

  1) A necessary and sufficient condition for an elementary product to be identically false is that it contains at least one pair of factors in which one is the negation of the other.

  2) A necessary and sufficient condition for an elementary sum to be identically true is that it contains at least one pair of factors in which one is the negation of the other.

Example 23:

Obtain a conjunctive normal form of

$q \vee (p \wedge {\sim}q) \vee ({\sim}p \wedge {\sim}q)$ and show that this expression is a tautology.

Solution:

$$q \vee (p \wedge \sim q) \vee (\sim p \wedge \sim q)$$

$$\equiv q \vee [(p \wedge \sim q)] \vee (\sim p \wedge \sim q)]$$

$$\equiv q \vee [(p \vee \sim p) \wedge \sim q]$$

$$\equiv [q \vee (p \vee \sim p)] \wedge (q \vee \sim q)$$

$\equiv$ Refer to PDF file

Minterms and Maxterms

- The disjunctive normal form and the conjunctive normal form of a given expression are not unique, i.e. there may be several normal forms that are equivalent to each other. minterm = CNF/PoS, maxterm = INF/SoP
- For example, $p \vee (q \wedge r) \equiv$

Minterms and Maxterms

- The disjunctive normal form and the conjunctive normal form of a given expression are not unique, i.e. there may be several normal forms that are equivalent to each other.
- For example, $p \vee (q \wedge r) \equiv$ Refer to PDF file

- Comparison would be easier if the expressions can be converted into some standard form that does not have many variations, preferably that is unique for each expression.
- *Minterm* or *Boolean conjunction* of several simple statement variables p, q, r,… is a product of p, q, r, … or their negations, formed in such a way that each variable appears exactly once either as itself or its negation.

- For *n* variables, there are 2^{n} possible minterms.
- E.g. for 2 variables *p* and *q*, the 4 minterms are

  pq, p~q, ~pq, ~p~q

<u>Principal Disjunctive Normal Form (pdnf)</u>

- Any expression that obtained by commuting the factors in the expressions above is not included in the list, as it would be equivalent to one of the minterms above.

The truth table for these minterms is as below:

minterms: PoS

p	q	$\sim p$	$\sim q$	$p \wedge q$	$p \wedge \sim q$	$\sim p \wedge q$	$\sim p \wedge \sim q$
0	0	1	1	0	0	0	1
0	1	1	0	0	0	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	0	0	0

- From the truth table,

  1) each minterm has the truth value 1(T) for exactly one combination of the truth values of p and q.

  2) no two minterms are equivalent.

- *Principal disjunctive normal form* (pdnf) or *sum-of-product canonical form*: an equivalent expression consisting of disjunctions of minterms only.
- To obtain pdnf,

1) For every truth value 1(T) in the truth table of the given expression, select the minterm which also has the truth value 1(T) for the same combination of the truth values of p and q.

2) Take the disjunction of these minterms.

Then the disjunction will be equivalent to the given expression.

Example 24:

Construct a truth table for $p \to q$, $p \vee q$ and $\sim(p \wedge q)$. Then obtain the principal disjunctive normal forms of these expressions.

p	q	$p \wedge q$	$p \wedge \sim q$	$\sim p \wedge q$	$\sim p \wedge \sim q$	$p \rightarrow q$	$p \vee q$	$\sim(p \wedge q)$
0	0	0	0	0	1	1	0	1
0	1	0	0	1	0	1	1	1
1	0	0	1	0	0	0	1	1
1	1	1	0	0	0	1	1	0

$$p \to q \equiv$$

$$\equiv$$

$$p \vee q \equiv$$

$$\equiv$$

$$\sim(p \wedge q) \equiv$$

$$\equiv$$

Notes:

- The number of minterms appearing in the normal form is the same as the number of entries with the truth value 1 in the truth table of the given expression.
- Every expression which is not a contradiction has an equivalent principal disjunctive normal form. Such a normal form is unique, except for the rearrangements of the factors in the disjunctions as well as in each of the minterms.

- A certain order in which the variables appear in the minterms as well as a definite order in which the minterms appear in the disjunction may impose. Then, the given two equivalent expressions must have identical principal disjunctive normal forms.
- Then for 3 variables p, q, and r, the 8 minterms are

- To obtain pdnf without constructing the truth table, first obtain a disjunctive normal form as before. Any elementary product which is a contradiction is dropped.
- Minterms are obtained in the disjunctions by introducing the missing factors. For example, if *p* is missing in a product, introduce $p \vee {\sim}p$, and then expand the term by the distributive law.
- Repeated minterms in the disjunctions are deleted.

Example 25:

Obtain the principal disjunctive normal forms of

i) $\sim p \vee q$

already in disjunctive normal form but not yet in PDNF

Example 25:

Obtain the principal disjunctive normal forms of

i) $\sim p \vee q \equiv [\sim p \wedge (q \vee \sim q)] \vee [(p \vee \sim p) \wedge q]$

(introduce missing factors)

Example 25:

Obtain the principal disjunctive normal forms of

i) $\sim p \vee q \equiv [\sim p \wedge (q \vee \sim q)] \vee [(p \vee \sim p) \wedge q]$

(introduce missing factors)

$\equiv (\sim p \wedge q) \vee (\sim p \wedge \sim q) \vee (p \wedge q) \vee (\sim p \wedge q)$

ii) $(p \wedge q) \vee (\sim p \wedge r) \vee (q \wedge r)$

$$\equiv pq + \overline{p}r + qr$$

Example 26:

Show that the following are equivalent expressions:

i) $p \vee (p \wedge q) \Leftrightarrow p$

ii) $p \vee (\sim p \wedge q) \Leftrightarrow p \vee q$

Example 27:

Obtain the principal disjunctive normal form of $p \to [(p \to q) \wedge \sim(\sim q \vee \sim p)]$.

Solution:

$p \to [(p \to q) \wedge \sim(\sim q \vee \sim p)]$

$= \sim p \vee [(\sim p \vee q) \wedge (q \wedge p)]$

$=$

Principal Conjunctive Normal Form

- For a given number of variables, a *maxterm* consists of disjunctions in which each variable or its negation, but not both, appears only once.
- The maxterms are the duals of minterms.
- Each of the maxterms has the truth value 0 (F) for exactly one combination of the truth values of the variables. Different maxterms have the truth value 0 for different combinations of the truth values of the variables.

- For a given expression, an equivalent expression consisting of conjunctions of the maxterms only is called its *principal conjunctive normal form*, or the *product-of-sums canonical form*.
- Every logical expression that is not a tautology has an equivalent principal conjunctive normal form, which is unique except for the rearrangement of the factors in the maxterms as well as in their conjunctions.

- If the principal disjunctive (conjunctive) normal form of a given expression A is known, then the principal disjunctive (conjunctive) normal form of $\sim A$ will consist of the disjunctive (conjunctive) of the remaining minterms (maxterms) which do not appear in the principal disjunctive (conjunctive) normal form of A.
- From $A \equiv \sim\sim A$, the pdnf (pcnf) of A can be obtained by repeated applications of De Morgan's laws to the pcnf (pdnf) of $\sim A$.

Example 28:

Let A represent $(\sim p \to r) \wedge (q \leftrightarrow p)$. Obtain the pcnf of A, and of $\sim A$. Deduce the pdnf of A.

$$A \equiv (\sim p \to r) \wedge (q \leftrightarrow p)$$
$$\equiv (p \vee r) \wedge [(q \to p) \wedge (p \to q)]$$
$$\equiv (p \vee r) \wedge (\sim q \vee p) \wedge (\sim p \vee q)$$

PCNF of *A*

$\equiv (p \vee q \vee r) \wedge (p \vee {\sim}q \vee r) \wedge (p \vee {\sim}q \vee {\sim}r)$

$\wedge ({\sim}p \vee q \vee r) \wedge ({\sim}p \vee q \vee {\sim}r)$

(5 maxterms)

Notes:

- The pcnf of a given expression can be written based on its truth table.
- The maxterms included correspond to the truth value 0 (F) in the truth table of that expression.
- The maxterms are written down by including the variable of its truth value is 0 (F) and its negation if the value is 1 (T).
- This is opposite to writing the minterms in pdnf.

Example 29:

Construct a truth table for the expression *A* in the previous example, and write its principal disjunctive normal form and the principal conjunctive normal form.

p	q	r	$\sim p$	$\sim p \to r$	$q \leftrightarrow p$	$(\sim p \to r) \wedge (q \leftrightarrow p)$
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	1	0	1	0	0	0
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	1	0	1	0	0
1	1	0	0	1	1	1
1	1	1	0	1	1	1

PDNF of *A*

PCNF of *A*

Notes:

- For a tautology, all the minterms are present and there is no maxterm.
- For a contradiction, all the maxterms are present and there is no minterm.

1.6 Valid and Invalid Arguments

Argument

- A sequence of statement.
- All statements except the final one are called premises / assumptions / hypothesis.
- Final statement is called conclusion.
- The symbol "$\therefore$", read as "therefore", place just before the conclusion.

Validity

- An argument form is *valid* when no matter what particular statements are substituted for the statement variables in its premises, if the resulting premises are all *true* then the conclusion is also true.

Note:

- The truth of conclusion is said to be *inferred* or *deducted* from the truth of the premises when an argument is valid and its premises are true.

Steps to Test Validity of an Argument

1) Identify the premises and conclusion of the argument.
2) Construct a truth table showing the truth values of all premises and the conclusion.
3) Find the row (critical row) in which all the premises are true.
4) In each critical row, determine the conclusion
a) if in each critical row the conclusion is also true, then the argument form is valid.
b) if there is *at least* ONE critical row in which the conclusion is false, the argument form is invalid.

Example 30:

Show that the following argument form is valid:

$$\begin{array}{l} p \vee (q \vee r) \\ \sim r \\ \hline \therefore\ p \vee q \end{array}$$

Solution:

p	q	r	$q \vee r$	$p \vee (q \vee r)$	$\sim r$	$p \vee q$
0	0	0	0	0	1	0
0	0	1	1	1	0	0
0	1	0	1	1	1	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	1
1	1	1	1	1	0	1

Since for all the critical rows, the conclusion is true, the argument form is valid.

Example 31:

Show that the following argument form is valid:

$$\begin{array}{l} p \vee q \\ \sim p \vee q \\ \sim r \\ \hline \therefore\ q \end{array}$$

Solution:

p	q	r	$p \vee q$	$\sim p$	$\sim p \vee q$	$\sim r$
0	0	0	0	1	1	1
0	0	1	0	1	1	0
0	1 √	0	1	1	1	1
0	1	1	1	1	1	0
1	0	0	1	0	1	1
1	0	1	1	0	1	0
1	1 √	0	1	0	1	1
1	1	1	1	0	1	0

Since for all the critical rows, the conclusion is true, the argument form is valid.

Example 32:

Show that the following argument form is invalid:

$$p \to q \lor \sim r$$

$$q \to p \land r$$

$$\therefore\ p \to r$$

Solution:

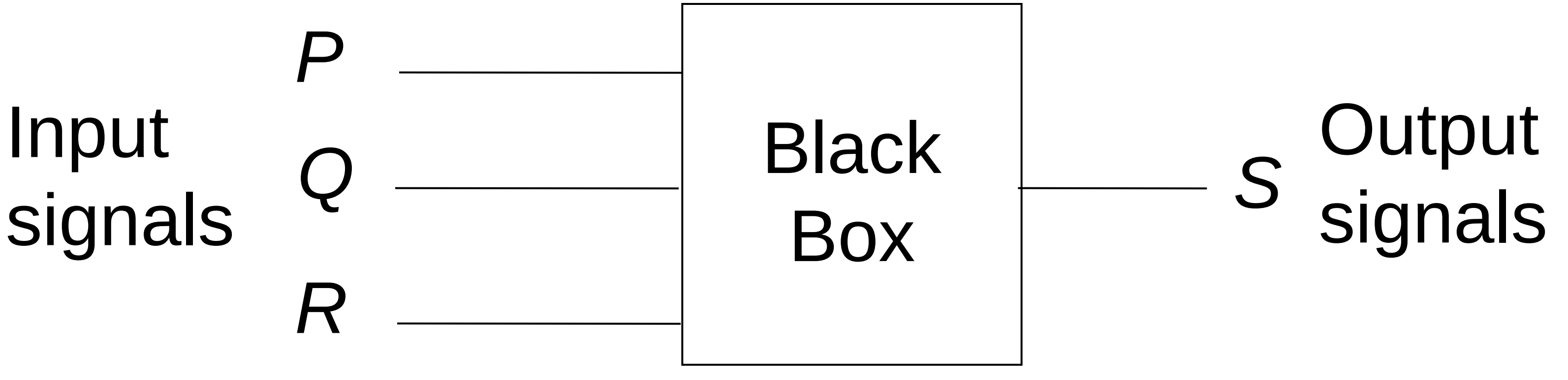
p	q	r	$\sim r$	$q \vee \sim r$	$p \to q \vee \sim r$	$p \wedge r$	$q \to p \wedge r$	$p \to r$
0	0	0	1	1	1	0	1	1
0	0	1	0	0	1	0	1	1
0	1	0	1	1	1	0	0	1
0	1	1	0	1	1	0	0	1
1	0	0	1	1	1	0	1	0
1	0	1	0	0	0	0	1	1
1	1	0	1	1	1	0	0	0
1	1	1	0	1	0	1	1	1

Since for all the critical rows, the conclusion is NOT ALWAYS true, the argument form is invalid.

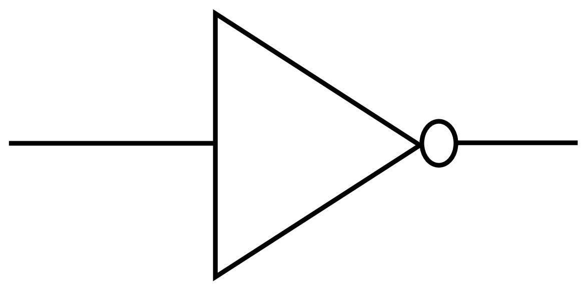
1.7 Application 1: Digital Logical Circuits

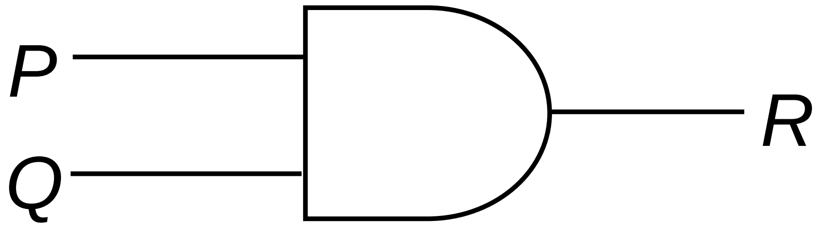
- The basic electronic components of a digital system.
- The symbol “0” and “1”, which denote false and true respectively, are called binary digits or bits.

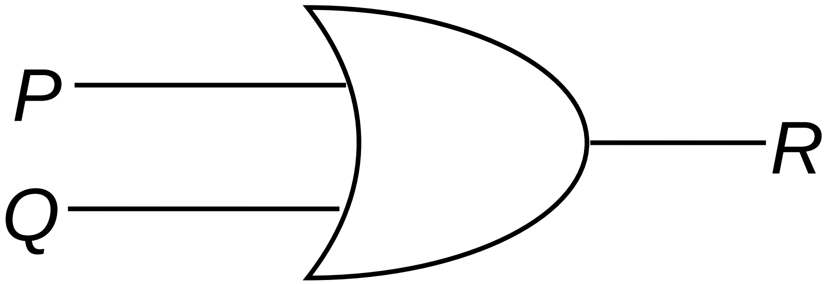
Black Box and Gates

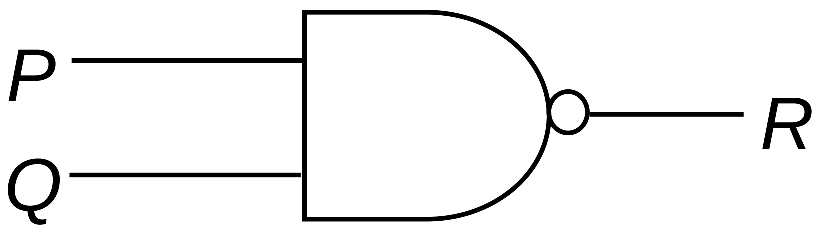


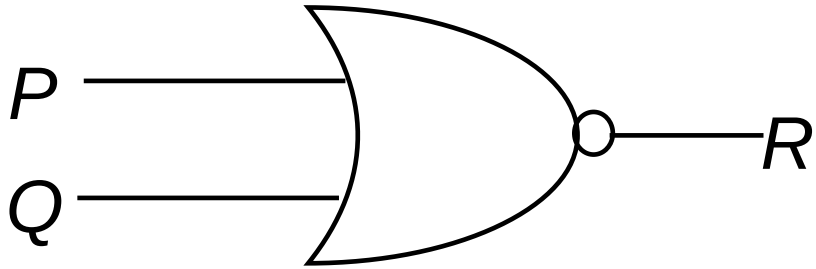
- The output of black box is completely specified by constructing an input / output table, that lists all its possible input signals together with their corresponding output signals.
- An efficient method for designing more complicated circuits is to build them by connecting less complicated black box circuits using gates as below,

<table>
<tr><th>Type of gate</th><th>Symbolic representation</th><th>Action</th></tr>
<tr><td>NOT-gate</td><td>P  Q</td><td><table><tr><th>Input, P</th><th>Output, Q</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td></tr>
</table>

<table>
<tr><th>Type of gate</th><th>Symbolic representation</th><th>Action</th></tr>
<tr><td>AND-gate</td><td>
</td><td><table>
<tr><th colspan="2">Input</th><th rowspan="2">Output, R</th></tr>
<tr><th>P</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Type of gate</th><th>Symbolic representation</th><th>Action</th></tr>
<tr><td>OR-gate</td><td>
</td><td>
<table>
<tr><th colspan="2">Input</th><th rowspan="2">Output, R</th></tr>
<tr><th>P</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Type of gate</th><th>Symbolic representation</th><th>Action</th></tr>
<tr><td>NAND-gate</td><td>
</td><td>
<table>
<tr><th colspan="2">Input</th><th rowspan="2">Output, R</th></tr>
<tr><th>P</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Type of gate</th><th>Symbolic representation</th><th>Action</th></tr>
<tr>
<td>NOR-gate</td>
<td>
</td>
<td>
<table>
<tr><th colspan="2">Input</th><th rowspan="2">Output, R</th></tr>
<tr><th>P</th><th>Q</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

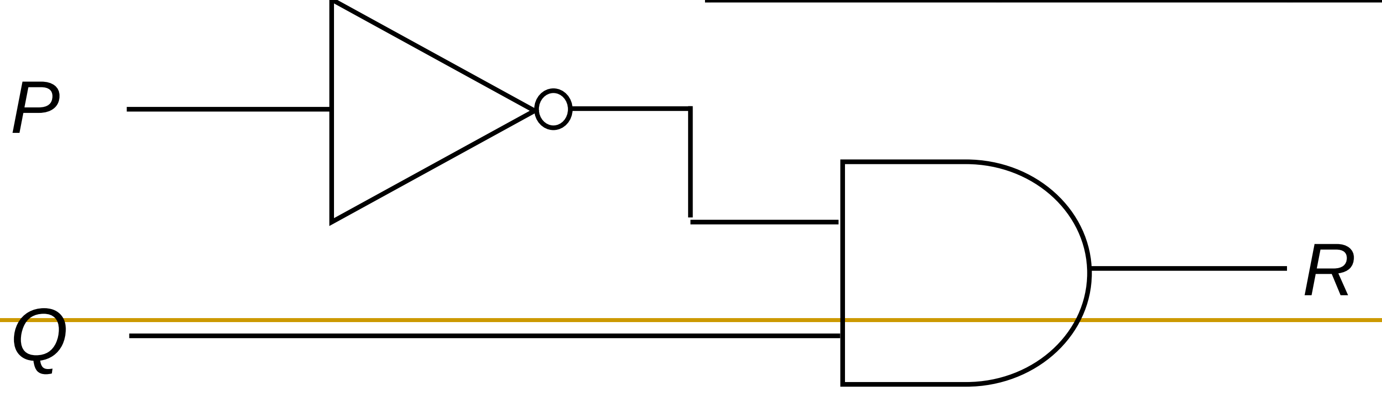
Notes:

- The action of the Not-, AND-, and OR- gates on signals correspond exactly to those of the logical connectives $\sim$, $\wedge$, and $\vee$ on statements.
- Never combine two input wires.
- A signal input wire can be split halfway and used as input for two separate gates.
- An output wire can be used as input.
- No output of a gate can eventually feed back into that gate.

Example 33:

Constructing the input / output table for the given circuit.

Input		Output
P	*Q*	*R*
0	0	0
0	1	1
1	0	0
1	1	0

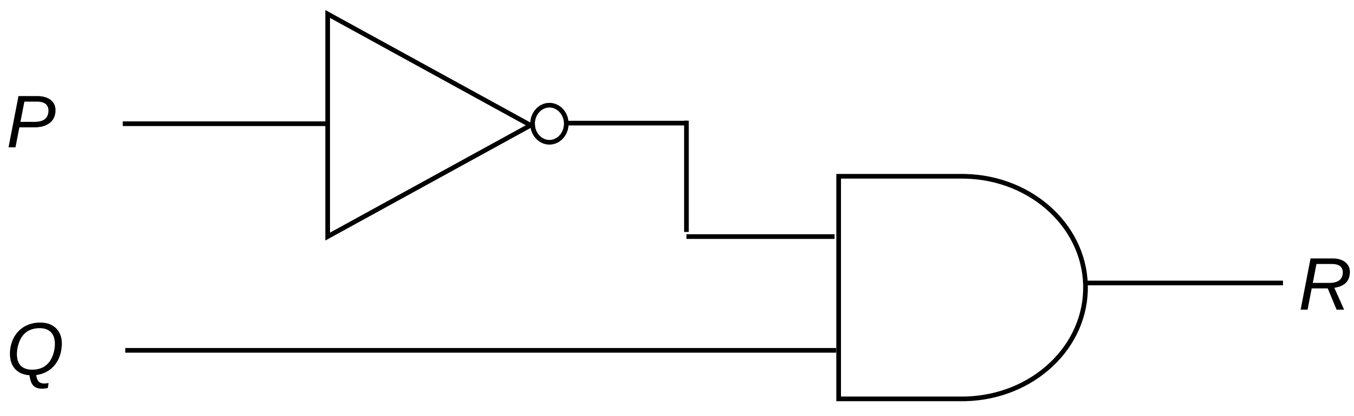


The Boolean Expression Corresponding to a Circuit

- The variables of an input signal (0 and 1) are called as Boolean variables.
- Expression composed of Boolean variables and the connectives $\sim$, $\wedge$, and $\vee$ is called a Boolean expression.

Example 34:

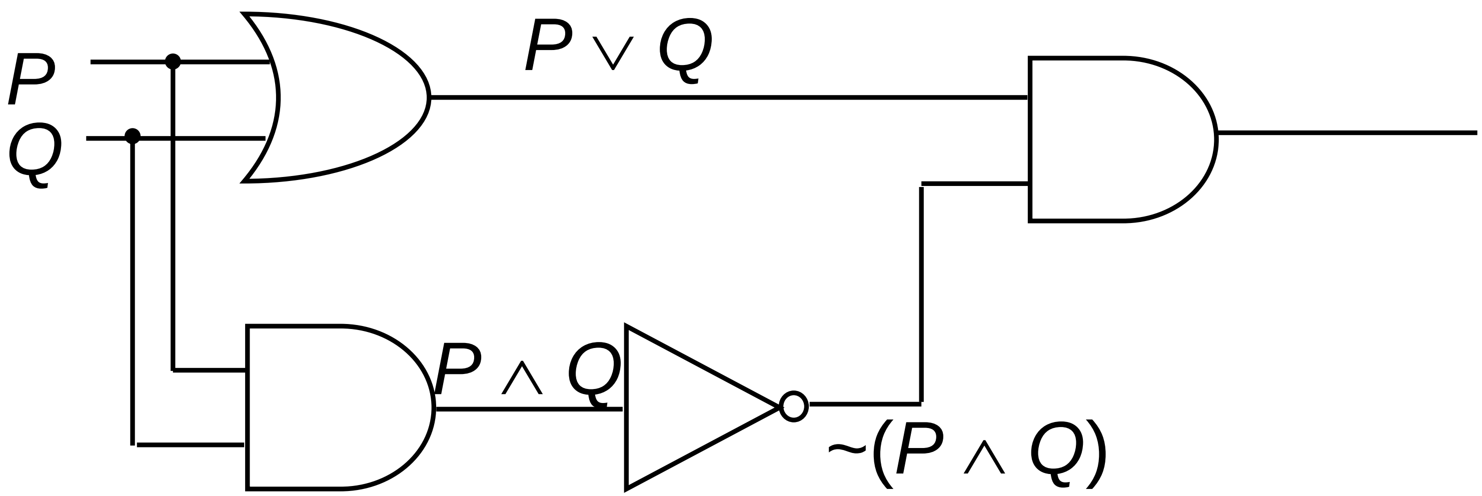
Find a Boolean expression for the circuit in example 33.



$R \equiv \sim P \wedge Q$

Example 35:

Find a Boolean expression for the circuit below.



$(P \lor Q) \land \sim(P \land Q)$

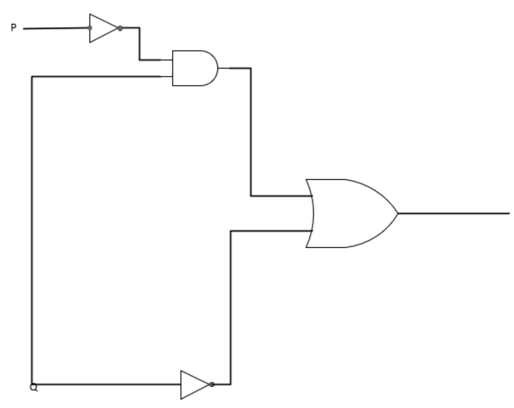
Note:

- A circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations (as example 34) is called a *recognizer*.

Example 36:

Construct a circuit for the Boolean expressions below:

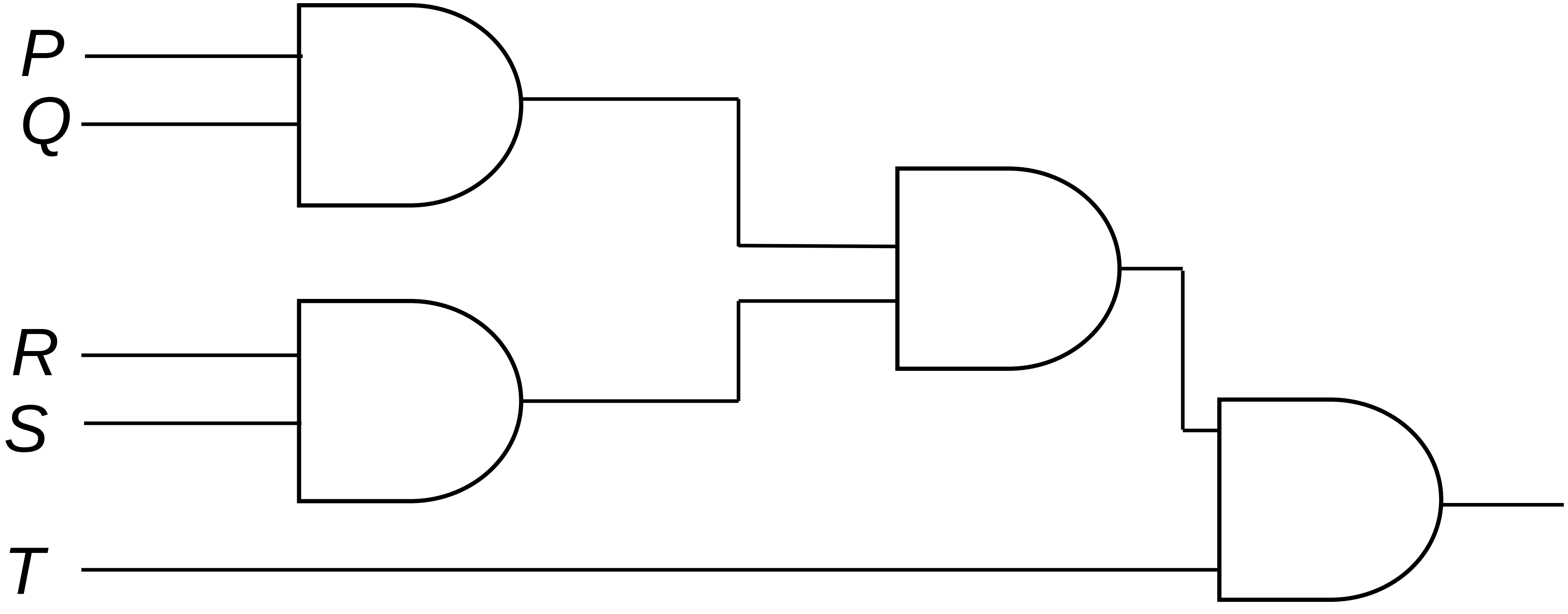
i) $(\sim P \wedge Q) \vee \sim Q$



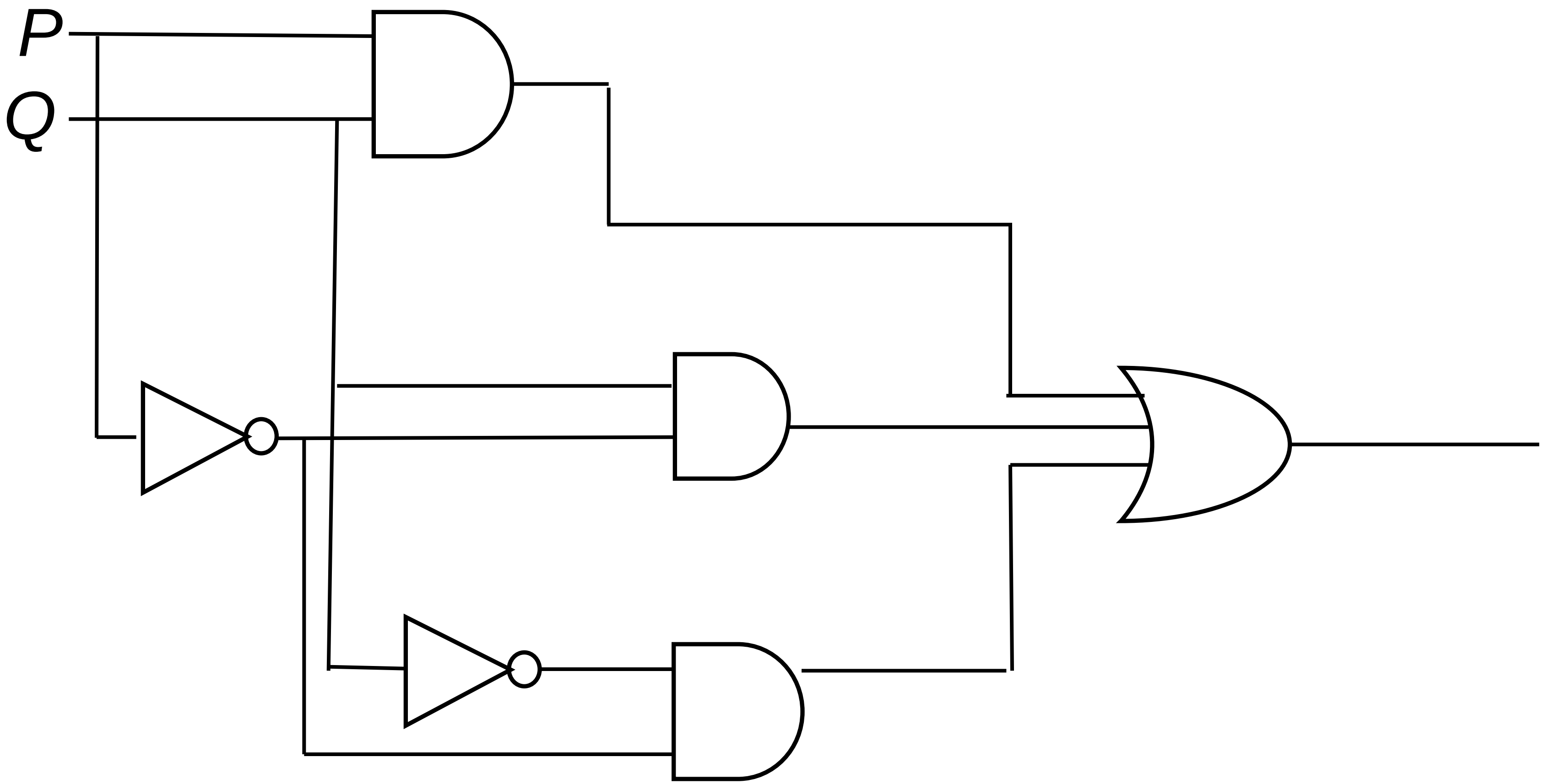
ii) $((P \wedge Q) \wedge (R \wedge S)) \wedge T$

Note:

- Such a circuit is called a *multiple-input AND-gate*.



iii) $(P \wedge Q) \vee (\sim P \wedge Q) \vee (\sim P \wedge \sim Q)$



Example 37:

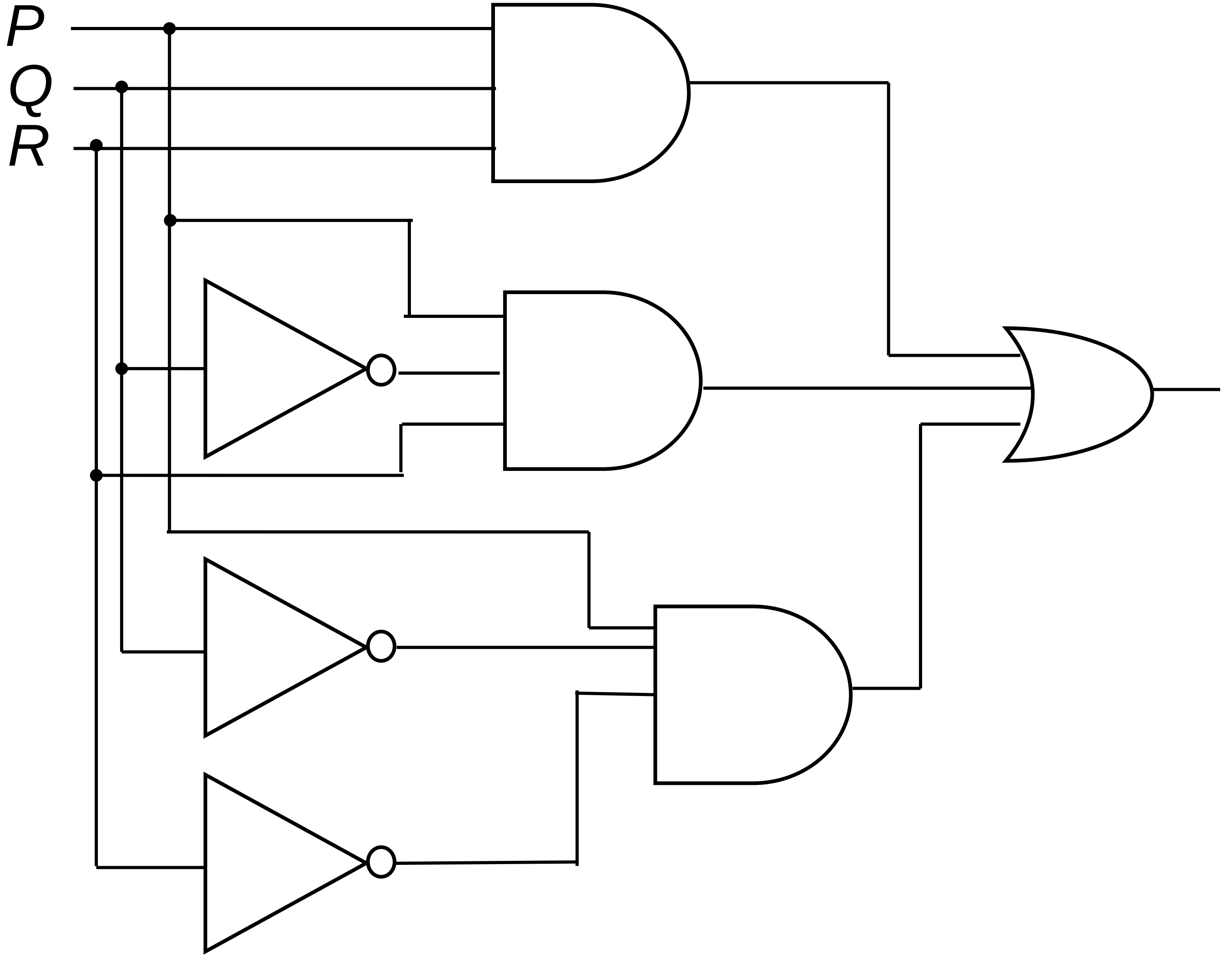
Find a circuit that correspond to the given input / output table.

Input			Output
P	*Q*	*R*	*S*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Example 37:

Find a circuit that correspond to the given input / output table.

Input			Output
P	*Q*	*R*	*S*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1


P
Q
R

Note:

- The expressions above are said to be in *disjunctive normal form* or *sum-of-products form*.

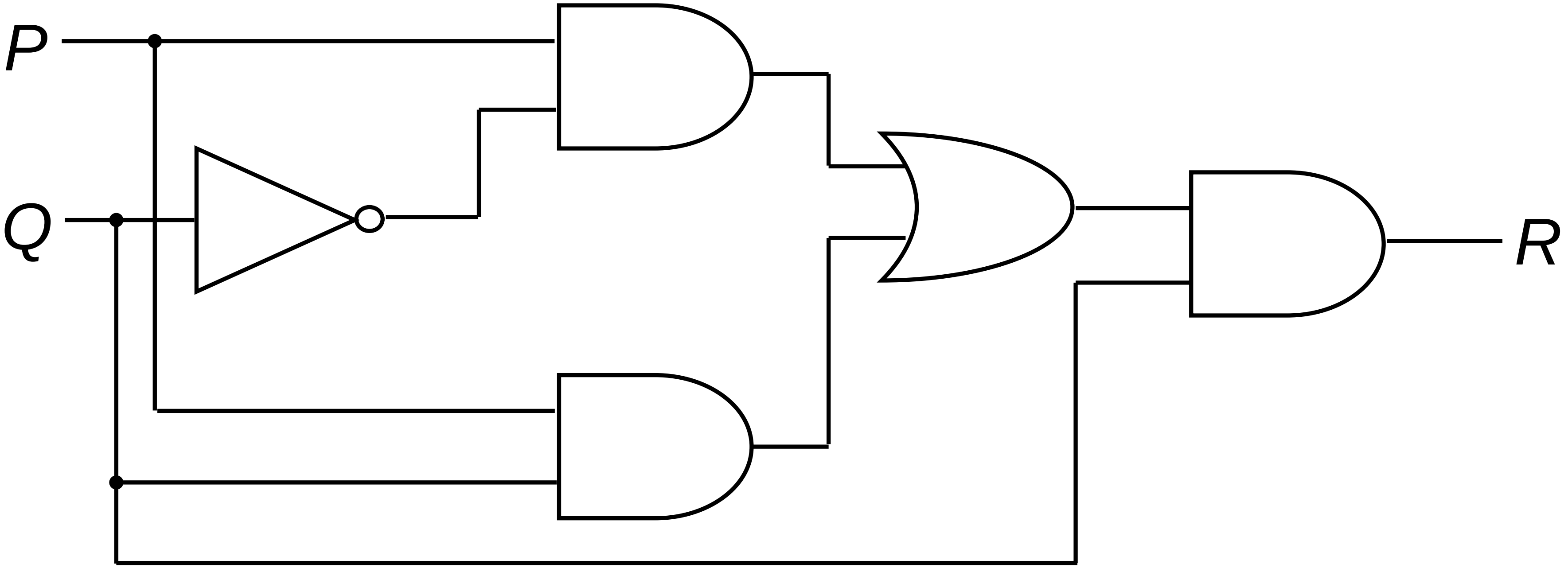
  $(p{\sim}q{\sim}r)+(p{\sim}qr)+(pqr)$

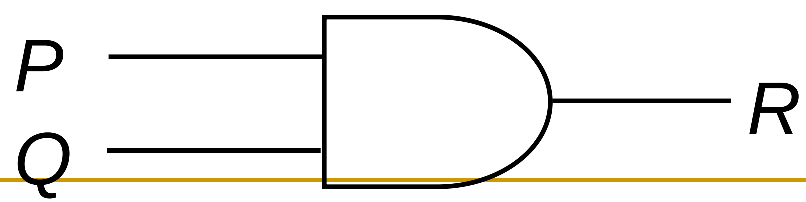
Simplifying Combinational Circuits

- Two digital logic circuits are equivalent if, and only if, their input / output tables are identical.

Example 38:

Show the following circits are equivalent.





$$[(P \wedge \sim Q) \vee (P \wedge Q)] \wedge Q$$

$=$

1.8 Application 2: Number Systems and Circuits for Addition

- Generally, decimal notation is based on the fact that any positive integer can be written uniquely as a sum of products of the form $d \cdot (10^{n})$ where $0 \leq d \leq 9$.

- E.g. $4 = 4(10^{0})$

  $25 =$ $2(10^{1})+5(10^{0})$

  $789 =$ $7(10^{2})+8(10^{1})+9(10^{0})$

  $3610 =$ $3(10^{3})+6(10^{2})+1(10^{1})+0(10^{0})$

Binary Representative of Numbers

- Base 2.
- In computer science, base 2 notation/binary notation, is especially important because the signals used in modern electronics are always in one of only two states.
- Any integer can be represented as a sum of powers of the form $d \cdot 2^n$, where $d = 0,1$.

- E.g. $23_{10} = 1(2^4) + 1(2^2) + 1(2^1) + 1(2^0)$ in binary notation,

Place	2^4, sixteens	2^3, eights	2^2, fours	2^1, twos	2^0, ones
Binary digit	1	0	1	1	1

$23_{10} = 10111_2$

Note:

- Any integer greater that 1 can be the base for a number system.
- The base normally written in subscripts.

Example 39:

$1_{10} = 1_2$

$2_{10} = 10_2$

$3_{10} = 11_2$

$4_{10} = 100_2$

$5_{10} = 101_2$

$6_{10} = 110_2$

$7_{10} = 111_2$

$8_{10} = 1000_2$

$9_{10} = 1001_2$

Example 40:

$101010_2 =$

$110110_2 =$

- Addition:
  - For base 10, $1_{10} = 1\ (2^0) = 1_2$,

    $1_{10} + 1_{10} = 2_{10}$;

    For binary notation, $1_2 + 1_2 = 10_2$

$$\begin{array}{r} 1_2 \\ +\ \ 1_2 \\ \hline 10_2 \\ \hline \end{array}$$

    so, for $1_2 + 1_2 + 1_2 = 11_2$

Example 41:

$10101_2 + 110_2 =$

Checking:

- Subtraction
  - For base 10, $10_{10} - 1_{10} = 9_{10}$ (use to borrow across several columns);
  - For binary notation, when borrow across column (borrow 1_2 from 10_2), it will remain 1_2.

$$\begin{array}{r} 1\,0_{2} \\ -\quad 1_{2} \\ \hline 1_{2} \\ \hline \end{array}$$

- So, for $11000_2 - 1001_2 =$

- So, for $11000_2 - 1001_2 = 1111_2$

$$\begin{array}{r} 11000_2 \\ -\ \ 1001_2 \\ \hline 1111_2 \\ \hline \end{array}$$

Example 42:

$101101_2 – 11001_2 =$

Checking:

Circuits for Computer Addition

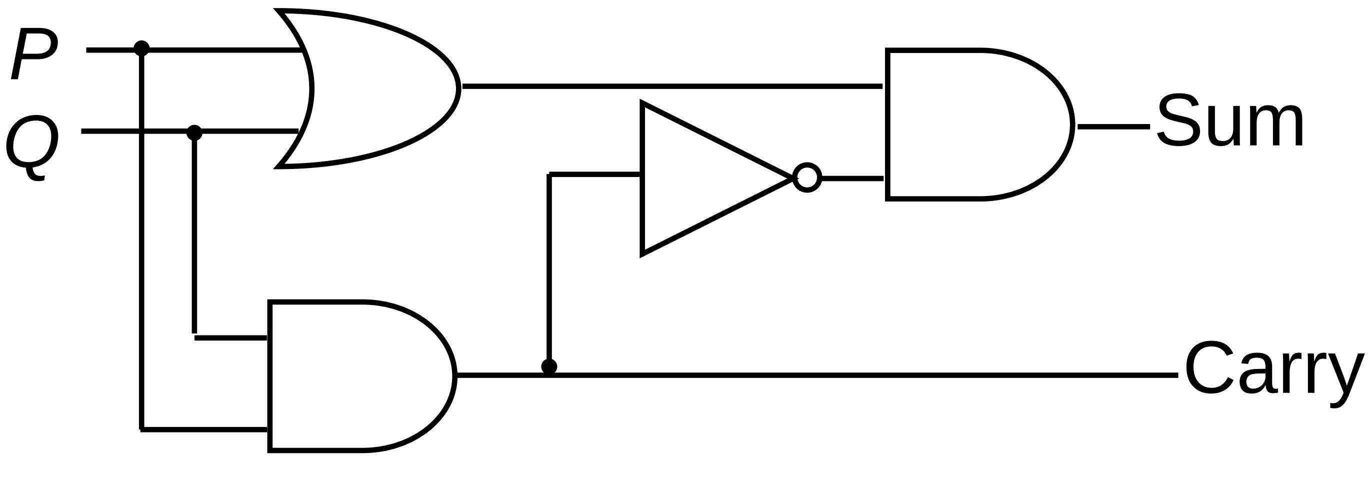
- When design a circuit to produce the sum of two binary digits P and Q, both digits can be either 0 or 1.
- Notice the below,

P		Q		Carry	Sum
0_2	+	0_2	$= 0_2 =$	0	0_2
0_2	+	1_2	$= 1_2 =$	0	1_2
1_2	+	0_2	$= 1_2 =$	0	1_2
1_2	+	1_2	$=$	1	0_2

- The *carry* output can be produced using *AND*-gate circuit, corresponding to $P \wedge Q$ in Boolean expression.
- The *sum* output can be produced using a circuit corresponding to the Boolean expression for the *exclusive or*, $P \veebar Q$ or $(P \vee Q) \wedge \sim(P \wedge Q)$.

P	*Q*	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

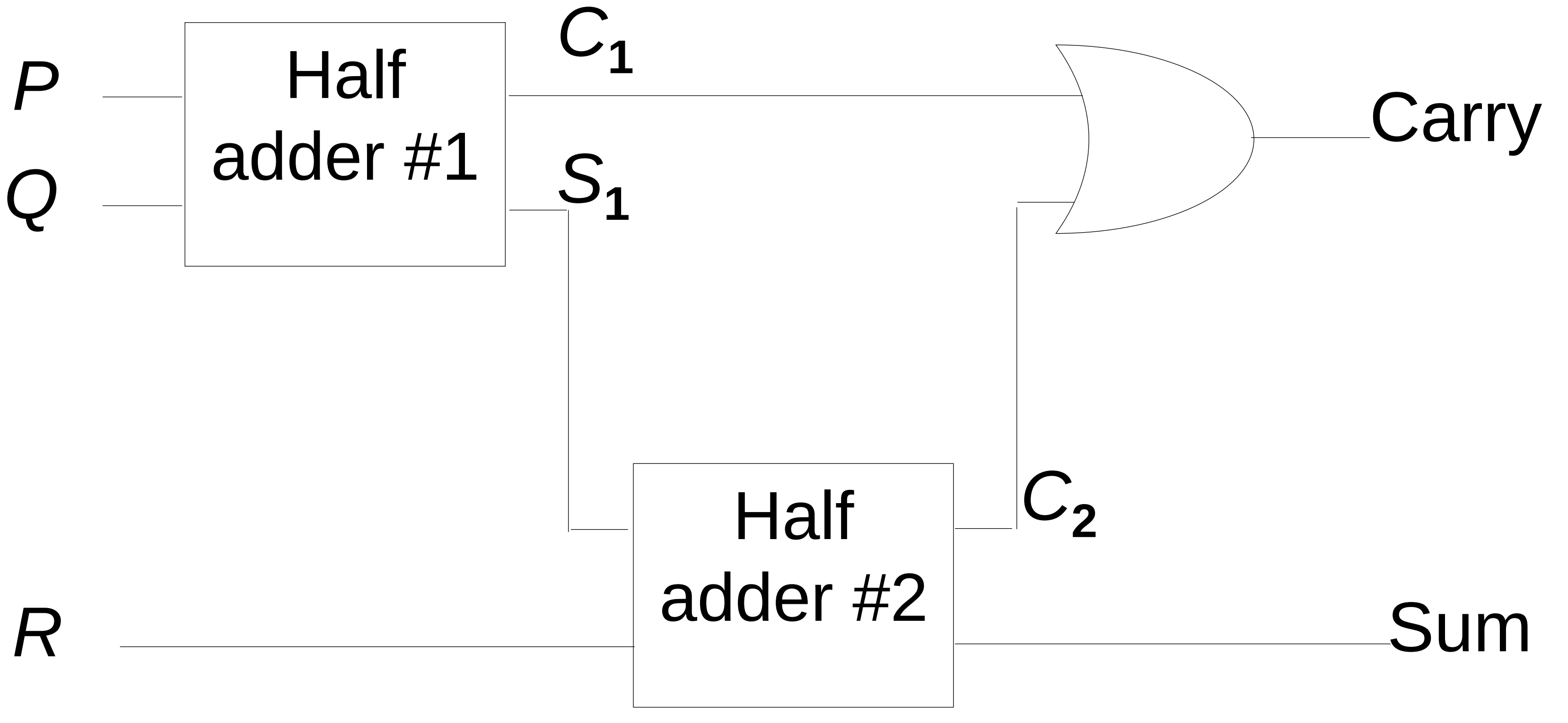
- This circuit is called a half-adder circuit.



- When adding two binary integers, each with more than 1 digit, the result may carry to the next column to the left, it may be necessary to add three binary digits at certain.
- A full-adder circuit is a circuit that will add multi digit binary numbers.
- E.g.

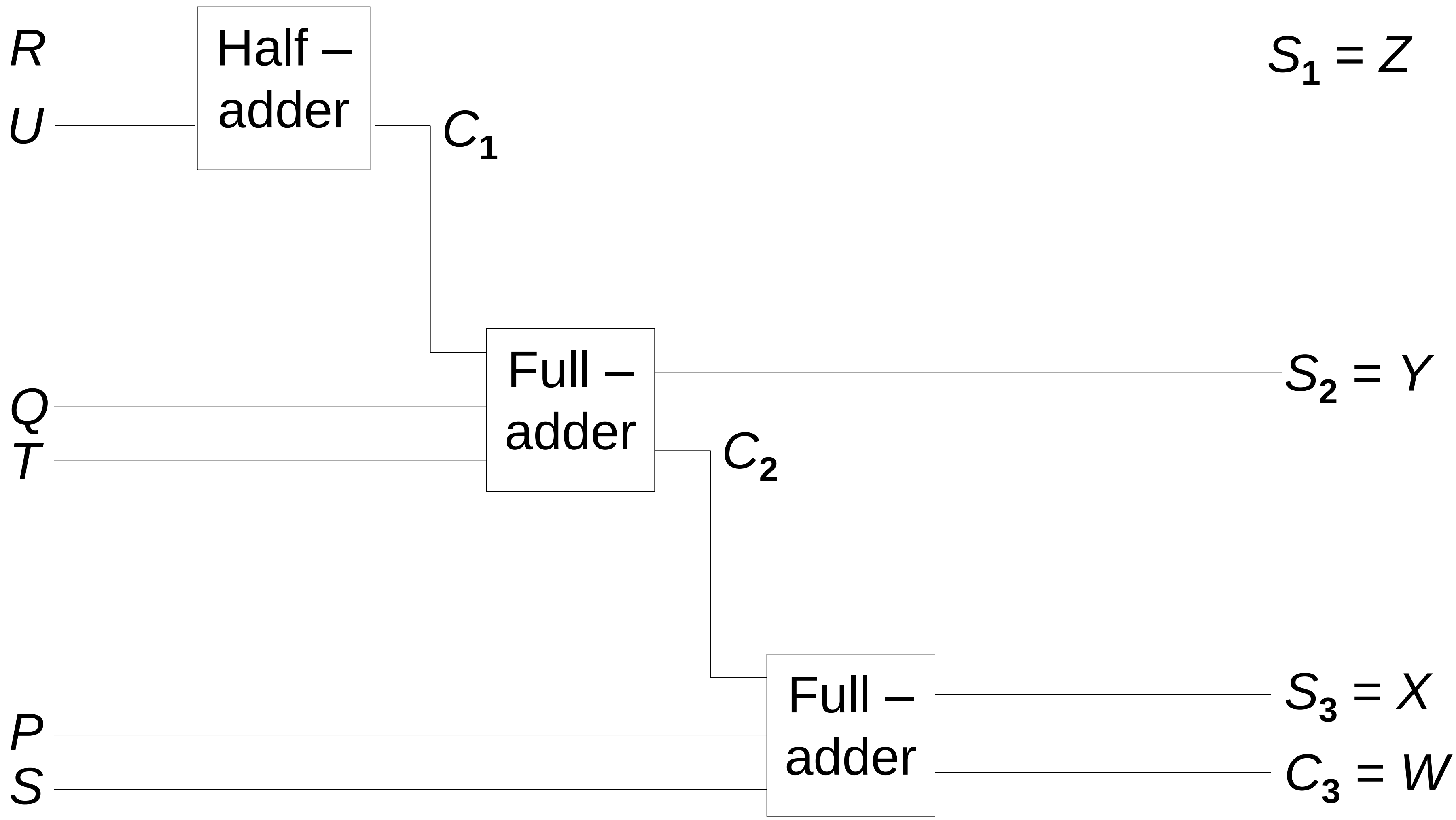
$$\begin{array}{r} 1_2 \\ 0_2 \\ +\ 1_2 \\ \hline 1\ 0_2 \end{array} \qquad\qquad \begin{array}{r} 1_2 \\ 1_2 \\ +\ 1_2 \\ \hline 1\ 1_2 \end{array}$$

carry sum carry sum


P
Q
Half
adder #1
C1
S1
Carry
Half
adder #2
C2
R
Sum

P	*Q*	*R*	C_1	C_2	S_1	Carry	Sum
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	1	0	0	0	1	0	1
0	1	1	0	1	1	1	0
1	0	0	0	0	1	0	1
1	0	1	0	1	1	1	0
1	1	0	1	0	0	1	0
1	1	1	1	0	1	1	1

- A parallel adder circuit is a circuit that built by two full-adders and one half-adder.
- This circuit will add two three-digit binary numbers to obtain the sum *WXYZ*.
- Can add binary numbers of any finite length.


R
U
Half –
adder
$S_1 = Z$
C_1
Q
T
Full –
adder
$S_2 = Y$
C_2
P
S
Full –
adder
$S_3 = X$
$C_3 = W$