CONTENT

Introduction to Logic Gates 1

Basic Logic Gates 3

Implementing XOR gate using Basic Logic gates 10

Universal Gates 14

Electrical Circuits to model the functionality of Logic gates 19

Boolean Algebra & Laws 22

Karnaugh-Maps 31

Designs simple logic circuits using Logic gates 36

Minterms and Maxterms 40

SOP to POS and POS to SOP conversions 43

Standard and Non-standard expressions 48

Designs logic circuits for a given scenario 51

Full adders and Half adders 58

Flip-Flops 65

Integrated Circuits 74

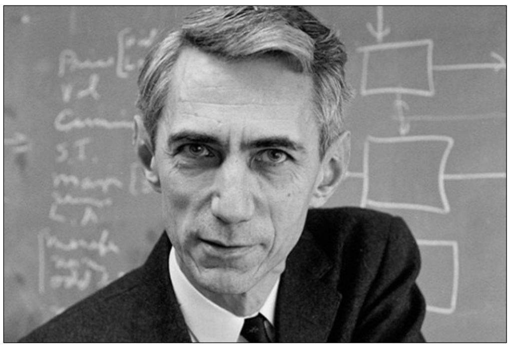
Introduction
හැඳින්වීම

George Boole (1815-1864)

George Boole, an English mathematician and logician, is known as the father of Boolean algebra, which is the mathematical framework used to describe the operation of logic gates.
ඉංග්‍රීසි ගණිතඥයෙකු සහ තාර්කිකයෙකු වන ජෝර්ජ් බූල්, තාර්කික ද්වාර ක්‍රියාකාරීත්වය විස්තර කිරීමට භාවිතා කරන ගණිතමය රාමුව වන බූලීය වීජ ගණිතයේ පියා ලෙසින් හැඳින්වේ.

Claude Shannon (1916-2001)

Claude Shannon used Boolean algebra to design digital circuits, laying the foundation for modern computing.
ක්ලෝඩ් ෂැනන් විසින් සංඛ්‍යාංක පරිපථ නිර්මාණය කිරීම සඳහා බූලියන් වීජ ගණිතය භාවිතා කළ අතර, එය නවීන පරිගණකකරණය සඳහා අඩිතාලම දැමීය.



What is a Boolean event?
බූලීය සිද්ධියක් යනු කුමක්ද?

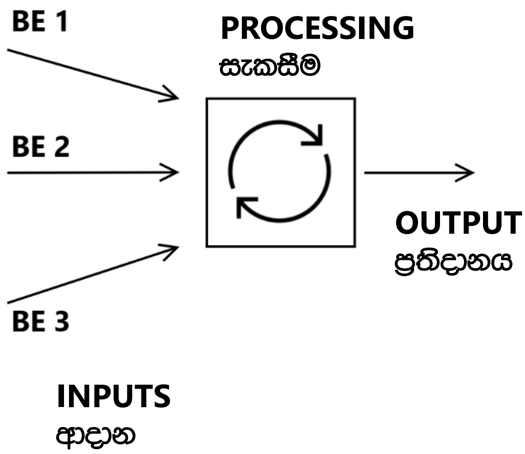
An event that has only two possible outcomes, where one outcome is the opposite of the other is known as a Boolean event.
එක් ප්‍රතිඵලයක් අනෙක් ප්‍රතිඵලයට ප්‍රතිවිරුද්ධ වන ප්‍රතිඵල දෙකක් පමණක් ඇති සිදුවීමක් බූලීය සිද්ධියක් ලෙස හැඳින්වේ.

Only one of the following can result here.
මෙහි ප්‍රථිඵල ලෙස පැමිණිය හැක්කෙ පහත ඒවායින් එකක් පමණි.

- Positive / True / Yes – 1
  ධන/සත්‍ය/ඔව්
- Negative / False / No – 0
  සෘණ/අසත්‍ය/නැත

What is a logic gate?
තාර්කික ද්වාරයක් යනු කුමක් ද?

An electronic component which takes one or several inputs, processes them according to the relevant logic and gives a single output is known as a logic gate.
ආදාන එකක් හෝ කිහිපයක් ලබාගනිමින් තමාට ආවේනික තර්කය අනුව ඒවා සකස් කර එක් ප්‍රතිදානයක් ලබා දෙන ඉලෙක්ට්‍රොනික සංරචකයක් තාර්තික ද්වාරයක් ලෙස හඳුන්වයි.



NOTES

- A logic gate can have any number of inputs.
  තාර්කික ද්වාරයකට ඕනෑ තරම් ආදාන සංඛ්‍යාවක් තිබිය හැකිය.

- Every input depends on a Boolean event.
  සෑම ආදානයක්ම බූලියානු සිද්ධියක් මත රඳා පවතී.

- A logic gate can have only one output.
  තාර්කික ද්වාරයකට තිබිය හැක්කේ එක් ප්‍රතිදානයක් පමණි.

- There are several types of logic gates according to the performing logic.
  l%shdldÍ ;¾lkh wkqj ;d¾lsl oajdr වර්ග කිහිපයක් තිබේ.

- They also have different shapes according to the logic.
  ඒවාට තර්කනය අනුව විවිධ හැඩ තලද පවතී.

Logic gates can be mainly divided into two types.
තාර්කික ද්වාර ප්‍රධාන වශයෙන් වර්ග දෙකකට බෙදා දැක්විය හැක.

1. Basic Logic Gates
   මූලික තාර්කික ද්වාර

2. Combinational Logic Gates
   සංයුක්ත තාර්කික ද්වාර

Input Combinations
ආදාන සංයෝජන

All possible combinations of unique binary values (0s and 1s) that can be applied to the inputs of a logic gate is known as input combinations.
තාර්කික ද්වාරයක ආදාන වලට ලබා දිය හැකි ද්විමය අගයන් ((0) සහ (1)) වල සියලුම එකිනෙකට වෙනස් සංයෝජන ආදාන සංයෝජන ලෙස හැඳින්වේ.

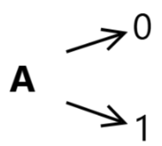
The number of possible input combinations depends on the number of inputs, following the formula:
පැවතිය හැකි ආදාන සංයෝජන ගණන පහත සූත්‍රය අනුගමනය කරන අතර එය ආදාන ගණන මත රඳා පවතී:

Number of Input Combinations = 2^n
ආදාන සංයෝජන ගණන

n is the number of input variables.
n යනු ආදාන විචල්‍ය සංඛ්‍යාව වේ.

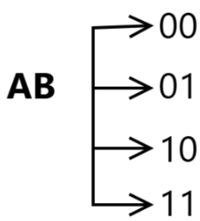
For 1 input variable (A):
එක් ආදාන විචල්‍යයක් සඳහා:

Number of Input Combinations = $2^1 = 2$
ආදාන සංයෝජන ගණන



For 2 input variables (AB):
ආදාන විචල්‍යය 2ක් සඳහා:

Number of Input Combinations = $2^2 = 4$
ආදාන සංයෝජන ගණන

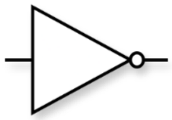


ictfromabc.com

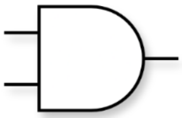
Basic Logic Gates
මූලික තාර්කික ද්වාර

Basic logic gates are the fundamental building blocks of digital circuits. They perform basic logical functions that are essential for digital computations.
මූලික තාර්කික ද්වාර යනු අංකිත පරිපථවල මූලික තැනුම් ඒකක වේ. ඒවා මගින් අංකිත ගණනය කිරීම් සඳහා අත්‍යවශ්‍ය වන මූලික තාර්කික කාර්යයන් ඉටු කරයි.

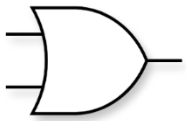
There are 3 basic logic gates.
මූලික තාර්කික ද්වාර 3 ක් පවතී.



NOT Gate



AND Gate



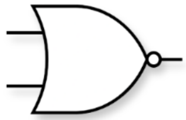
OR Gate

NOTE:
NOT gate is the only logic gate that has only one input.
NOT ද්වාරය යනු එක් ආදානයක් පමණක් පවතින එකම තාර්කික ද්වාරය වේ.

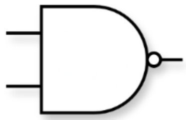
All the other logic gates at least have two inputs.
අනෙකුත් සියලුම තාර්කික ද්වාරවල අවම වශයෙන් ආදාන දෙකක්වත් ඇත.

Combinational logic gates
සංයුක්ත තාර්කික ද්වාර

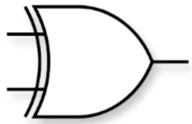
Gates which are made up of two or more basic gates are known as Combinational logic gates.
මූලික තාර්කික ද්වාර දෙකක් හෝ කිහිපයක් සංයෝජනය වීමෙන් සෑදී ඇති ද්වාර සංයුක්ත තාර්කික ද්වාර ලෙස හඳුන්වයි.



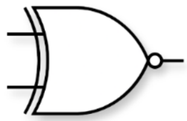
NOR Gate



NAND Gate



XOR Gate



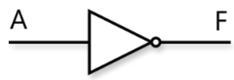
XNOR Gate

What is a truth table?
සත්‍යතා වගුවක් යනු කුමක්ද?

A truth table is a table that displays all possible input values and their corresponding output values for a logical expression or Boolean function.
සත්‍යතා වගුවක් යනු තාර්කික ප්‍රකාශනයක් හෝ බූලියන් ශ්‍රිතයක් සඳහා පැවතිය හැකි සියලුම ආදාන අගයන් සහ ඒවායේ අනුරූප ප්‍රතිදාන අගයන් පෙන්වන වගුවකි.

It helps visualize how a logic circuit or Boolean operation behaves under different conditions.
එය තාර්කික පරිපථයක් හෝ බූලියන් මෙහෙයුමක් විවිධ තත්වයන් යටතේ ක්‍රියා කරන ආකාරය දැක ගැනීමට උපකාරී වේ.

NOT Gate (Negation)
NOT ද්වාරය (නාස්ත්‍යර්ථය)



Gives the output as the inverse of the input.
ආදානයේ ප්‍රතිලෝමය ලෙස ප්‍රතිදානය ලබාදේ.

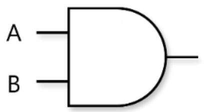
F is the inverse of A.
F යනු A හි ප්‍රතිලෝමයයි.

ICN	Input (A)	Output (F)
1	False (0)	True (1)
2	True (1)	False (0)

AND Gate (Conjuction)
AND ද්වාරය (සංයුක්ද්ජනය)

Output becomes positive only if all the inputs are positive.
ආදාන සියල්ල සත්‍ය වන විට පමණක් ප්‍රතිදානය සත්‍ය වේ.
Output becomes negative when at least one input is negative.
එක් ආදානයක් හෝ අසත්‍ය වන විට ප්‍රතිදානය අසත්‍ය වේ.
Should have two or more inputs.
ආදාන දෙකක් හෝ කිහිපයක් පැවතිය යුතුය.
Output is always the product of the inputs.
ප්‍රතිදානය යනු ආදානයන්ගේ ගුණිතයයි.

2 input AND gate
ආදාන දෙකේ AND ද්වාරය

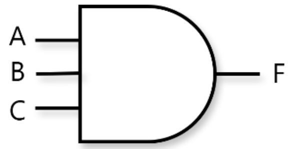


ICN	A	B	F
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Expression for $F = A.B$ (A AND B)

F becomes true when A and B is true.
A සහ B සත්‍ය වන විට F සත්‍ය වේ.

3 input AND gate
ආදාන තුනේ AND ද්වාරය



F becomes true when A AND B AND C is true.
A සහ B සහ C සත්‍ය වන විට F සත්‍ය වේ.

ICN	A	B	C	F
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

$F = A.B.C$ (A AND B AND C)

OR Gate (Disjuction)
OR ද්වාරය (වියුංජනය)

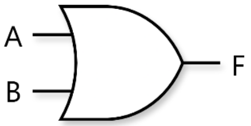
Output becomes positive if at least one input is positive.
එක් ආදානයක් හෝ සත්‍ය වන විට ප්‍රතිදානය සත්‍ය වේ.

Output becomes negative when all the inputs are negative.
සියලු ආදාන අසත්‍ය වන විට ප්‍රතිදානය අසත්‍ය වේ.

Should have two or more inputs.
ආදාන දෙකක් හෝ කිහිපයක් පැවතිය යුතුය.

ictfromabc.com

2 input OR gate
ආදාන දෙකේ OR ද්වාරය

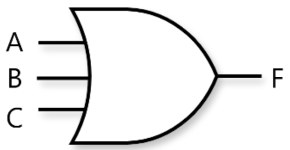


ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Expression for $F = A + B$ (A OR B)

F becomes true when A OR B is true.
A හෝ B සත්‍ය වන විට F සත්‍ය වේ.

3 input OR gate
ආදාන තුනේ OR ද්වාරය



F becomes true when A OR B OR C is true.
A හෝ B හෝ C සත්‍ය වන විට F සත්‍ය වේ.

Truth table

ICN	A	B	C	F
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

$F = A + B + C$ (A OR B OR C)

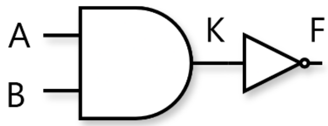
NAND gate
NAND ද්වාරය

This gate is made by providing the output of an AND gate as the input to a NOT gate.
AND ද්වාරයක ප්‍රතිදානය NOT ද්වාරයකට සම්බන්ධ කිරීම මගින් මෙම ද්වාරය සෑදේ.

Output becomes the inverse of AND gate.
ප්‍රතිදානය වන්නේ AND ද්වාරයෙහි ප්‍රතිදානයේ ප්‍රතිලෝමයයි.

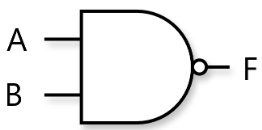
Here, output becomes negative only when all the inputs are positive. Output becomes positive when at least one input is negative.
මෙහිදී ආදාන සියල්ල සත්‍ය වන විට ප්‍රතිදානය අසත්‍ය වන අතර ආදාන එකක් හෝ අසත්‍ය වන විට ප්‍රතිදානය සත්‍ය වේ.

2 input NAND gate
ආදාන දෙකේ NAND ද්වාරය



ICN	A	B	K
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

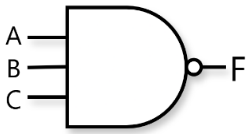
K	F
0	1
0	1
0	1
1	0



ICN	A	B	F
1	0	0	1
2	0	1	1
3	1	0	1
4	1	1	0

$F = \overline{A.B}$ (A NAND B)

3 input NAND gate
ආදාන තුනේ NAND ද්වාරය



ICN	A	B	C	F
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

$F = \overline{A.B.C}$

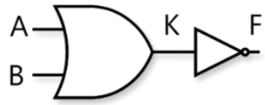
NOR gate
NOR ද්වාරය

This gate is made by providing the output of an OR gate as the input to a NOT gate.
OR ද්වාරයක ප්‍රතිදානය NOT ද්වාරයකට සම්බන්ධ කිරීම මගින් මෙම ද්වාරය සෑදේ.

Output becomes the inverse of OR gate.
ප්‍රතිදානය OR ද්වාරයෙහි ප්‍රතිදානයේ ප්‍රතිලෝමයයි.

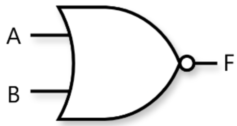
Here, output becomes positive only when all the inputs are negative. Output becomes negative when at least one input becomes positive.
මෙහිදී ආදාන සියල්ල අසත්‍ය වන විට ප්‍රතිදානය සත්‍ය වන අතර ආදාන එකක් හෝ සත්‍ය වන විට ප්‍රතිදානය අසත්‍ය වේ.

2 input NOR gate
ආදාන දෙකේ NOR ද්වාරය



ICN	A	B	K
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

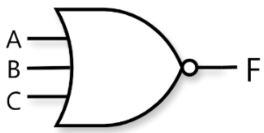
K	F
0	1
1	0
1	0
1	0



ICN	A	B	F
1	0	0	1
2	0	1	0
3	1	0	0
4	1	1	0

$F = \overline{A + B}$

3 input NOR gate
ආදාන තුනේ NOR ද්වාරය

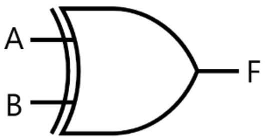


ICN	A	B	C	F
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	0

$F = \overline{A + B + C}$

XOR Gate
XOR ද්වාරය

2 input XOR gate
ආදාන දෙකේ XOR ද්වාරය



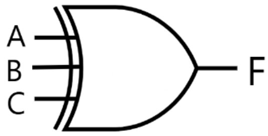
ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

$F = A \oplus B$

Output becomes positive when and only when one input is positive.
එක් ආදානයක් සත්‍ය නම් හා එක් ආදානයක් පමණක්ම සත්‍ය නම් ප්‍රතිදානය සත්‍ය වේ.

F becomes true when ONLY A OR ONLY B is true.
F සත්‍ය වන්නේ A පමණක් හෝ B පමණක් සත්‍ය වන විටදීය.

3 input XOR gate
ආදාන තුනේ XOR ද්වාරය



Output becomes positive when the positive number of inputs is an odd number.
සත්‍ය ආදාන ගණන ඔත්තේ සංඛ්‍යාවක් වන විට ප්‍රතිදානය සත්‍ය වේ.

ICN	A	B	C	F
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

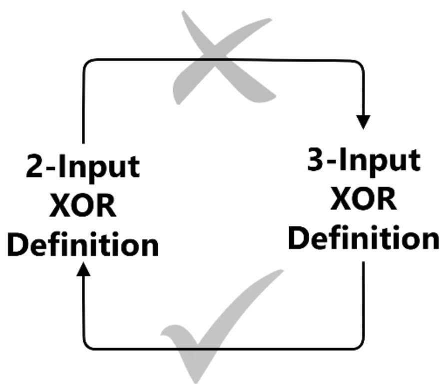
$F = A \oplus B \oplus C$

NOTE: (Basic Knowledge)

Even numbers ඉරට්ටේ සංඛ්‍යා:
Divisible by 2 (Ex. 0, 2, 4, 6).
2න් ඉතිරි නැතිව බෙදිය හැක.

Odd numbers ඔත්තේ සංඛ්‍යා:
Not divisible by 2 (Ex. 1, 3, 5, 7).
2න් ඉතිරි නැතිව බෙදිය නොහැක.

NOTE:
The definition of a 3-input XOR gate is also valid for a 2-input XOR gate. But the definition of a 2-input XOR gate is not valid for a 3-input XOR gate.
ආදාන තුනෙහි XOR ද්වාරයක අර්ථ දැක්වීම ආදාන දෙකෙහි XOR ද්වාරයක් සඳහා ද වලංගු වේ. නමුත් ආදාන දෙකෙහි XOR ද්වාරයක අර්ථ දැක්වීම ආදාන තුනෙහි XOR ද්වාරයක් සඳහා වලංගු නොවේ.



XNOR gate
XNOR ද්වාරය

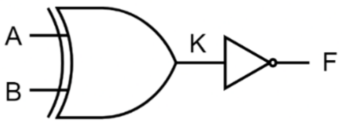
This gate is made by connecting the output of an XOR gate to the input of a NOT gate.
XOR ද්වාරයක ප්‍රතිදානය NOT ද්වාරයකට සම්බන්ධ කිරීම මගින් මෙම ද්වාරය සෑදේ.

Output becomes the inverse of XOR gate.
ප්‍රතිදානය XOR ද්වාරයහි ප්‍රතිදානයහි ප්‍රතිලෝමයයි.

2 input XNOR gate
ආදාන දෙකේ XNOR ද්වාරය

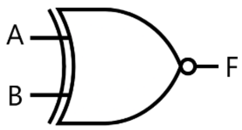
Output becomes negative when and only one input is positive.
එක් ආදානයක් සත්‍ය නම් හා එක් ආදානයක් පමණක් සත්‍ය නම් ප්‍රතිදානය අසත්‍ය වේ.

F becomes false when ONLY A OR ONLY B is true.
අසත්‍ය වන්නේ පමණක් හෝ පමණක් සත්‍ය වන විටදීය.



ICN	A	B	K
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

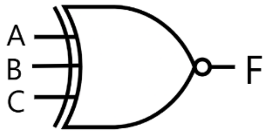
K	F
0	1
1	0
1	0
0	1



ICN	A	B	F
1	0	0	1
2	0	1	0
3	1	0	0
4	1	1	1

$F = \overline{A \oplus B}$

3 input XNOR gate
ආදාන තුනේ XNOR ද්වාරය



Output becomes negative when the positive number of inputs of an input combination is an odd number.
සත්‍ය ආදාන ගණන ඔත්තේ සංඛ්‍යාවල් වන විට ප්‍රතිදානය අසත්‍ය වේ.

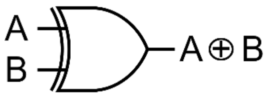
ICN	A	B	C	F
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

$F = \overline{A \oplus B \oplus C}$

ictfromabc.com

Implementing XOR gate from basic logic gate
මූලික තාර්කික ද්වාර භාවිතයෙන් XOR ද්වාරය ස්ථාපනය කිරීම

- **Two input XOR gate**



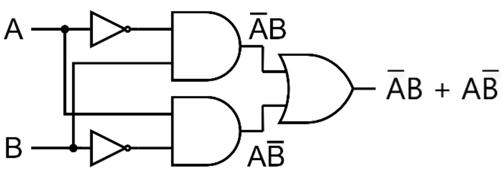
A	B	K
0	0	0
0	1	1
1	0	1
1	1	0

$K_{SOP} = \overline{A}.B + A.\overline{B}$

$K_{POS} = (A + B).(\overline{A} + \overline{B})$

By using basic logic gates
මූලික තාර්කික ද්වාර භාවිතයෙන් ගොඩනැගීම.

1. SOP

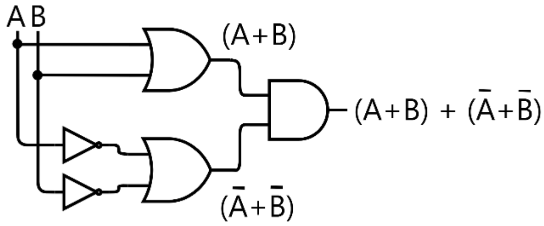


A	B	$\overline{A}$	$\overline{B}$	$\overline{A}B$	$A\overline{B}$	F
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

As we can see K = F
අපට පෙනෙන පරිදි K = F වේ.

Note: $A \oplus B = A\overline{B} + \overline{A}B$

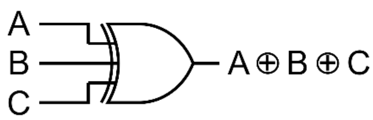
2. POS



A	B	$\overline{A}$	$\overline{B}$	$(A + B)$	$(\overline{A} + \overline{B})$	F
0	0	1	1	0	1	0
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	1	0	0	1	0	0

Note: $A \oplus B = (A + B).(\overline{A} + \overline{B})$

- **Three input XOR gate**



A	B	C	K
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

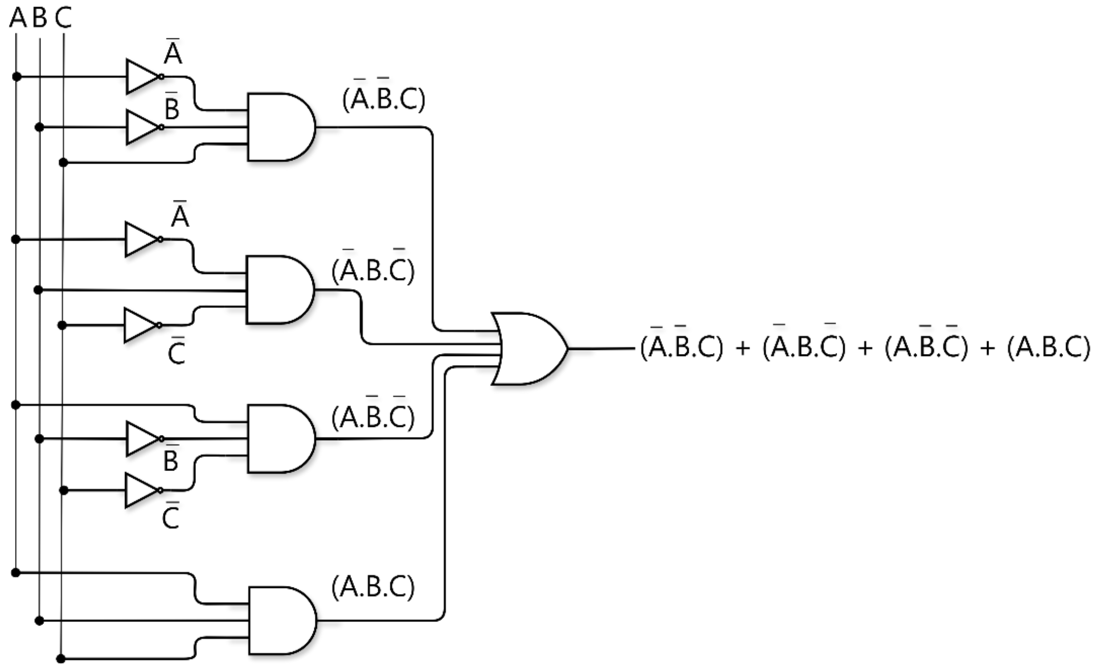
$K_{SOP} = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$

$K_{POS} = (A + B + C).(A + \overline{B} + \overline{C}).(\overline{A} + B + \overline{C}).$
$(\overline{A} + \overline{B} + C)$

ictfromabc.com

By using basic logic gates
මූලික තාර්කික ද්වාර භාවිතයෙන් ගොඩනැගීම.

1. SOP

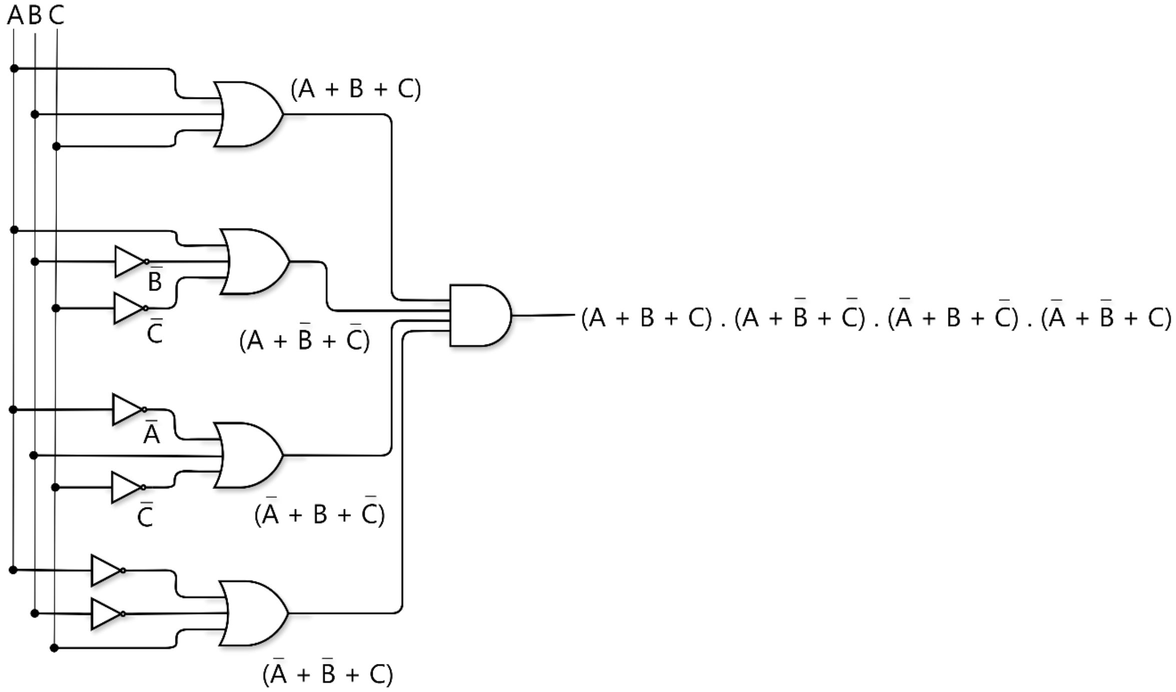


A	B	C	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{A}.\overline{B}.C$	$\overline{A}.B.\overline{C}$	$A.\overline{B}.\overline{C}$	$A.B.C$	F
0	0	0	1	1	1	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	1
0	1	0	1	0	1	0	1	0	0	1
0	1	1	1	0	0	0	0	0	0	0
1	0	0	0	1	1	0	0	1	0	1
1	0	1	0	1	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	1	1

Note: $A \oplus B \oplus C = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.B.C$

ictfromabc.com

2. POS

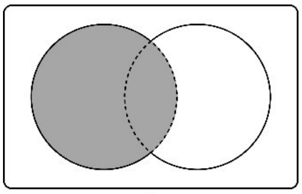


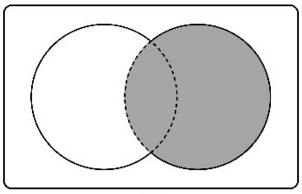
A	B	C	$\overline{A}$	$\overline{B}$	$\overline{C}$	$A + B + C$	$A + \overline{B} + \overline{C}$	$\overline{A} + B + \overline{C}$	$\overline{A} + \overline{B} + C$	F
0	0	0	1	1	1	0	1	1	1	0
0	0	1	1	1	0	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	1
0	1	1	1	0	0	1	0	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	1	0	1	0	1	1	0	1	0
1	1	0	0	0	1	1	1	1	0	0
1	1	1	0	0	0	1	1	1	1	1

Note: $A \oplus B \oplus C = (A + B + C).\left(A + \overline{B} + \overline{C}\right).\left(\overline{A} + B + \overline{C}\right).(\overline{A} + \overline{B} + C)$

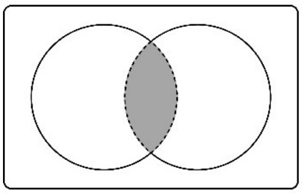
Logic gates and their relevant Ven diagram.
තාර්කික ද්වාර සහ ඒවාට අදාළ වෙන් රූප සටහන.

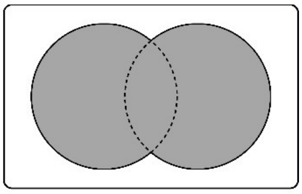
INPUT

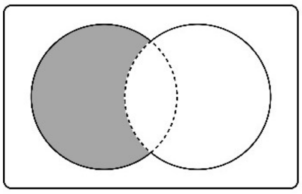

A

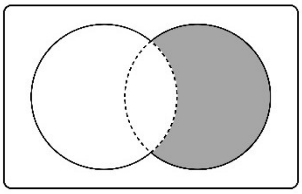

B

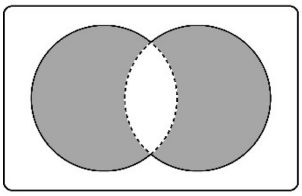
OUTPUT

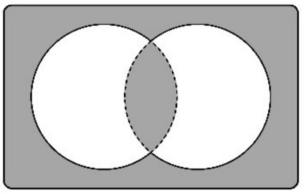

A AND B

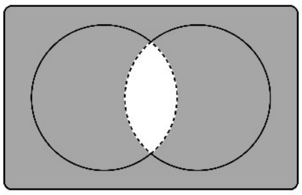

A OR B

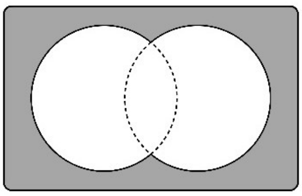

A NOT B


B NOT A


A XOR B


A XNOR B


A NAND B


A NOR B

ictfromabc.com

Universal Gates
සාර්වත්‍ර තාර්කික ද්වාර

A universal gate is a logic gate that can be used to implement any Boolean function and construct all other basic logic gates (AND, OR, NOT) using only NAND gates or only NOR gates, without needing any other gate type.
සාර්වත්‍ර ද්වාරයක් යනු ඕනෑම බූලීයානු ශ්‍රිතයක් ක්‍රියාත්මක කිරීමට, වෙනත් කිසිදු ද්වාර වර්ගයක අවශ්‍යතාවකින් තොරව NAND ද්වාර හෝ NOR ද්වාර පමණක් භාවිතා කරමින් අනෙකුත් සියලුම මූලික තාර්කික ද්වාර (AND, OR, NOT) ගොඩනැගීමට භාවිතා කළ හැකි තාර්කික ද්වාරයකි.

Reasons for using Universal gates
සාර්වත්‍ර ද්වාර භාවිතා කිරීමට හේතු

1. Lower Cost & Simpler Manufacturing
අඩු පිරිවැය සහ සරල නිෂ්පාදනය

Using only one type of gate (NAND or NOR) reduces manufacturing complexity and lowers cost in digital circuit design.
එක් වර්ගයක ද්වාරයක් (NAND හෝ NOR) පමණක් භාවිතා කිරීම නිෂ්පාදන සංකීර්ණතාව අඩු කරන අතර අංකිත පරිපථ නිර්මාණයේ පිරිවැය අඩු කරයි.

2. Easier Standardization & Optimization
පහසු ප්‍රමිතිකරණය සහ ප්‍රශස්තිකරණය

Standardizing circuits with only NAND or NOR gates simplifies design and troubleshooting.
NAND හෝ NOR ද්වාර පමණක් සහිත පරිපථ මගින් ප්‍රමිතිකරණය කිරීම, සැලසුම් කිරීම සහ දෝශ නිරාකරණය සරල කරයි.

3. Flexibility in Logic Design
තාර්කික නිර්මාණයේ නම්‍යශීලීභාවය

Any Boolean function can be implemented using just NAND or NOR gates, making them highly versatile in circuit design.
ඕනෑම බූලියානු ශ්‍රිතයක් NAND හෝ NOR ද්වාර පමණක් භාවිතයෙන් ක්‍රියාත්මක කළ හැකි අතර, ඒවා පරිපථ නිර්මාණයේදී ඉතා බහුකාර්ය වේ.

4. Reliability & Fault Tolerance
විශ්වසනීයත්වය සහ දෝෂ අඩු වීම

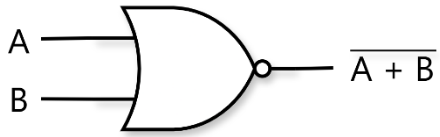
NAND and NOR gates are less prone to electrical noise and delays, improving reliability in circuits.
NAND සහ NOR ද්වාර විදුලි ඝෝෂාව සහ ප්‍රමාදයන්ට අඩු ප්‍රවණතාවක් ඇති අතර, එමගින් පරිපථවල විශ්වසනීයත්වය වැඩි දියුණු කරයි.

Efficiency Concern
කාර්යක්ෂමතාව පිළිබඳ අවධානය

Implementing a circuit with only NAND/NOR might require more logic levels, increasing propagation delay compared to using dedicated AND, OR, and NOT gates.
NAND/NOR පමණක් සහිත පරිපථයක් ක්‍රියාත්මක කිරීම සඳහා තාර්කික මට්ටම් වැඩි ගණනක් අවශ්‍ය විය හැකි අතර, එයටම වෙන්වූ AND, OR, සහ NOT ද්වාර භාවිතා කිරීමට සාපේක්ෂව මෙහි ප්‍රචාරණ ප්‍රමාදය වැඩි වේ.

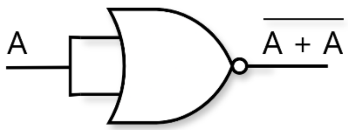
More gates mean more power usage, which may not be ideal for low-power applications.
වැඩි ද්වාර සංඛ්‍යාවක් යනු වැඩි බල පරිභෝජනයක් වන අතර, අඩු බලශක්ති යෙදුම් සඳහා සුදුසු නොවිය හැකිය.

1. **NOR GATE**



Implementing NOT gate from NOR gate (Inverter)

NOR ද්වාරයකින් NOT ද්වාරයක් ස්ථාපනය කිරීම



Proof:

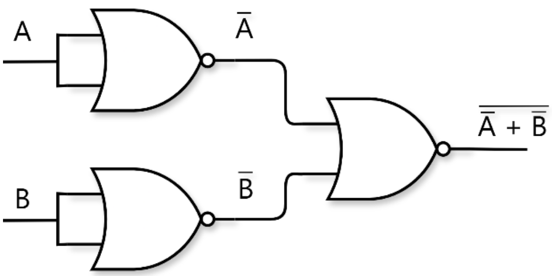
$= \overline{A + A}$

$= \overline{A} . \overline{A}$ De Morgan's Law

$= \overline{A}$ Idempotent Law

Implementing AND gate from NOR gate

NOR ද්වාරයකින් AND ද්වාරයක් ස්ථාපනය කිරීම



Proof:

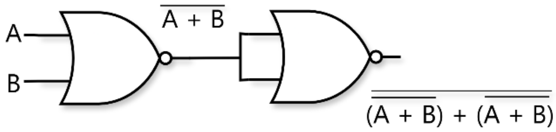
$= \overline{\overline{A} + \overline{B}}$

$= \overline{\overline{A}} . \overline{\overline{B}}$ De morgan's Law

$= A . B$ Double inverse Law

Implementing OR gate from NOR gate

NOR ද්වාරයකින් OR ද්වාරයක් ස්ථාපනය කිරීම



Proof:

$= \overline{\overline{(A + B)} + \overline{(A + B)}}$

$= \overline{\overline{(A + B)}} . \overline{\overline{(A + B)}}$ De Morgan's Law

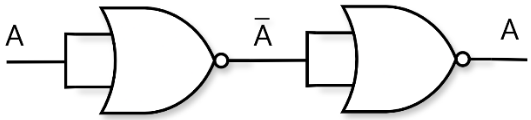
$= (A + B) . (A + B)$ Double inverse Law

$= (A + B)$ Idempotent Law

$= (A + B)$

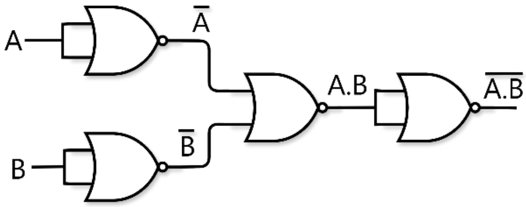
Implementing Buffer from NOR gate

NOR ද්වාරයකින් Buffer එකක් නිර්මාණය කිරීම



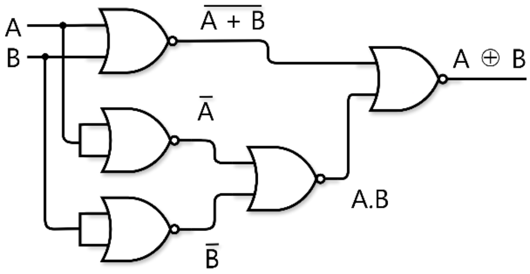
Implementing NAND gate from NOR gate

NOR ද්වාරයකින් NAND ද්වාරයක් ස්ථාපනය කිරීම



ictfromabc.com

Implementing Exclusive-OR (XOR) gate from NOR gate
NOR ද්වාරයකින් XOR ද්වාරයක් ස්ථාපනය කිරීම



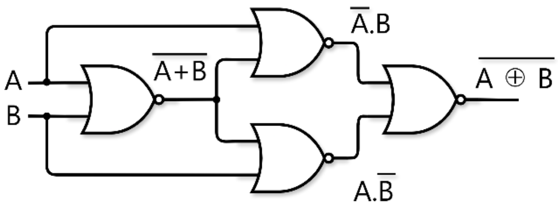
Proof:

$= \overline{\overline{(A+B)} + (AB)}$	
$= \overline{\overline{(A+B)}} \,.\, \overline{(AB)}$	De Morgan’s Law
$= (A+B).\overline{(AB)}$	Double inverse Law
$= (A+B)\,.\,(\overline{A}+\overline{B})$	De Morgan’s Law
$= A\overline{A} + A\overline{B} + \overline{A}B + B\overline{B}$	Distributive Law
$= 0 + A\overline{B} + \overline{A}B + 0$	Inverse Law
$= A\overline{B} + \overline{A}B$	Identity Law

Note

$A\overline{B} + \overline{A}B = A \oplus B$

Implementing Exclusive-NOR gate (XNOR) from NOR gate
NOR ද්වාරයකින් XNOR ද්වාරයක් ස්ථාපනය කිරීම



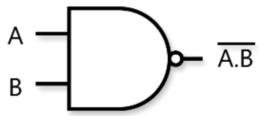
Proof:

$\overline{(A\overline{B}) + (\overline{A}B)}$	
$(\overline{A\overline{B}}).(\overline{\overline{A}B})$	De Morgan’s Law
$(\overline{A} + \overline{\overline{B}}).(\overline{\overline{A}} + \overline{B})$	Double inverse Law
$(\overline{A} + B).(A + \overline{B})$	Double inverse Law
$A\overline{A} + \overline{A}\overline{B} + AB + B\overline{B}$	Distributive Law
$0 + \overline{A}\overline{B} + AB + 0$	Inverse Law
$\overline{A}\overline{B} + AB$	Identity Law

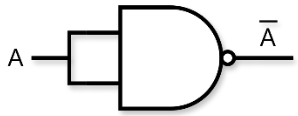
Note

$\overline{A}\overline{B} + AB = \overline{A \oplus B}$

2. NAND GATE



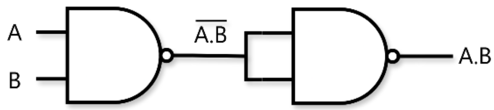
Implementing NOT gate from NAND gate (Inverter)
NAND ද්වාරයකින් NOT ද්වාරයක් ස්ථාපනය කිරීම



Proof:

$= \overline{A.A}$
$= \overline{A} + \overline{A}$ De Morgan's Law
$= \overline{A}$ Idempotent Law

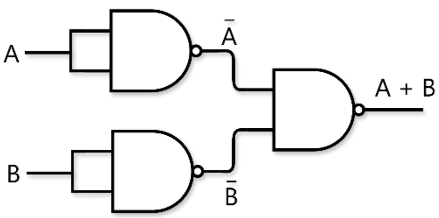
Implementing AND gate from NAND gate
NAND ද්වාරයකින් AND ද්වාරයක් ස්ථාපනය කිරීම



Proof:

$= \overline{\overline{(A.B)}.\overline{(A.B)}}$
$= \overline{\overline{(A.B)}} + \overline{\overline{(A.B)}}$ De Morgan's Law
$= (A.B) + (A.B)$ Double inverse Law
$= (A.B)$ Idempotent Law

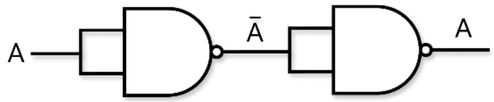
Implementing OR gate from NAND gate
NAND ද්වාරයකින් OR ද්වාරයක් ස්ථාපනය කිරීම



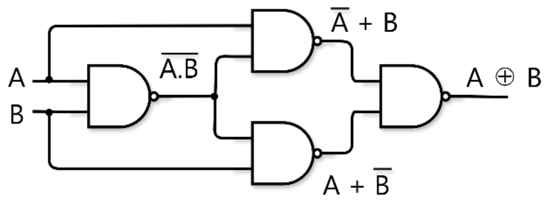
Proof:

$= \overline{\overline{A}.\overline{B}}$
$= \overline{\overline{A}} + \overline{\overline{B}}$ De Morgan's Law
$= A + B$ Double inverse Law

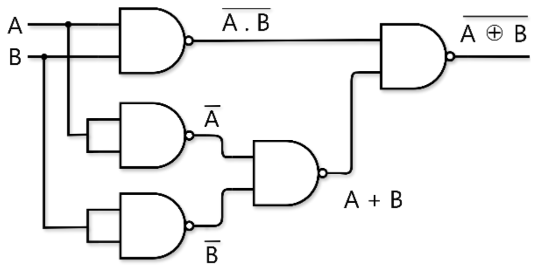
Implementing buffer from NAND gate
NAND ද්වාරයකින් Buffer එකක් නිර්මාණය කිරීම



Implementing Exclusive-OR gate (XOR) from NAND gate
NAND ද්වාරයකින් XOR ද්වාරයක් ස්ථාපනය කිරීම



Implementing Exclusive - NOR gate (XNOR) from NAND gate
NAND ද්වාරයකින් XNOR ද්වාරයක් ස්ථාපනය කිරීම

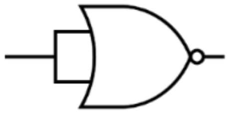
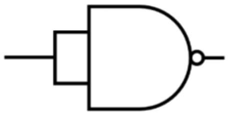
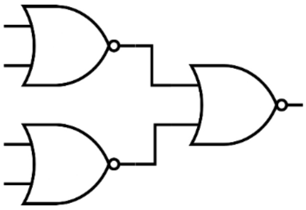
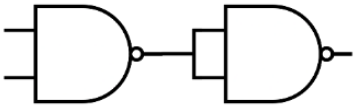
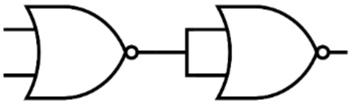
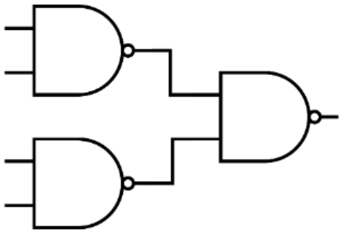
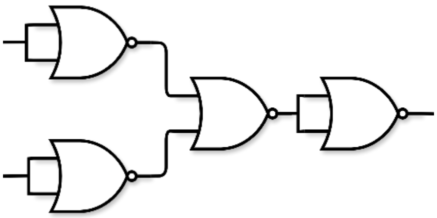
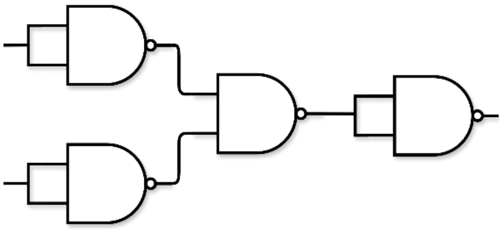
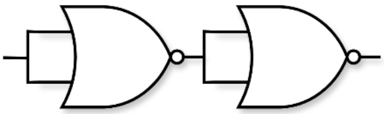
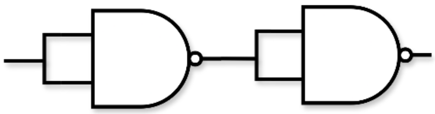


Proof:

$= \overline{\overline{(A.B)}.\overline{(\overline{A}.\overline{B})}}$
$= \overline{\overline{(A.B)}} + \overline{\overline{(\overline{A}.\overline{B})}}$ De Morgan's Law
$= (A.B) + (\overline{A}.\overline{B})$ Double inverse Law
$= (\overline{A}.\overline{B}) + (A.B)$ Associate law

ictfromabc.com

Summary

	NOR	NAND
NOT		
AND		
OR		
	NAND 	**NOR** 
Buffer		

Electric circuits to model the functionality of logic gates
විද්‍යුත් පරිපථ භාවිතයෙන් තාර්කික ද්වාරවල ක්‍රියාකාරීත්වය ආදර්ශනය කිරීම

Circuits built using electronic components such as transistors, diodes, and resistors to replicate the behavior of basic logic gates (AND, OR, NOT, etc.) in digital systems.
ඩිජිටල් පද්ධති තුළ මූලික තාර්කික ද්වාරවල (AND, OR, NOT ආදිය) හැසිරීම අනුකරණය කිරීම සඳහා ට්‍රාන්සිස්ටර, ඩයෝඩ සහ ප්‍රතිරෝධක වැනි ඉලෙක්ට්‍රොනික සංරචක භාවිතයෙන් ගොඩනගා ඇති පරිපථ වේ.

Components Used
භාවිතා වන සංරචක

- **Inputs → Switch**
  ආදාන → ස්විචය

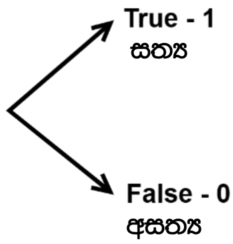
- **Outputs → Bulbs**
  ප්‍රතිදාන → බල්බ

- **Connections → Wires**
  සම්බන්ධක → වයර්

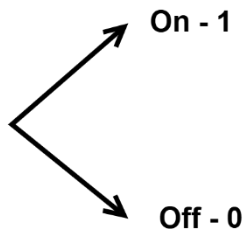
- **Power Supply → Batteries**
  බල සැපයුම → බැටරි

The convention used when drawing circuits.
පරිපථ ඇඳීමේදී භාවිතා කරන සම්මුතිය.

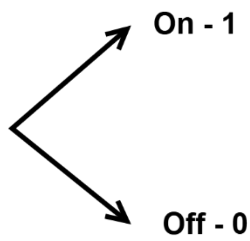
- **INPUT & OUTPUT**
  ආදාන සහ ප්‍රතිදාන



- **SWITCH**
  ස්විචය



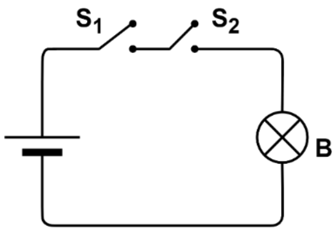
- **BULBS**
  බල්බ



ictfromabc.com

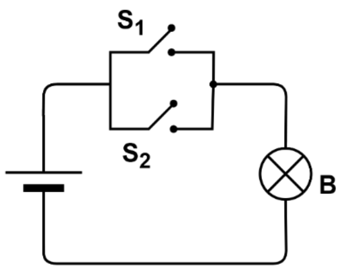
Gate Implementations
ද්වාර ක්‍රියාත්මක කිරීම

1. AND Gate
AND ද්වාරය



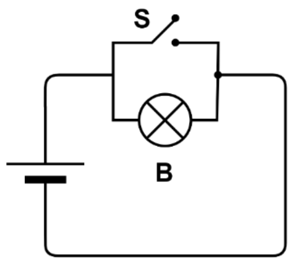
S_1	S_2	B
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate
OR ද්වාරය



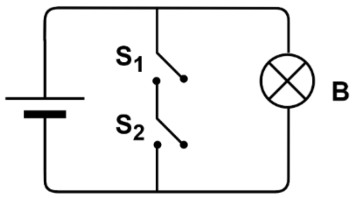
S_1	S_2	B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate (Inverter)
NOT ද්වාරය



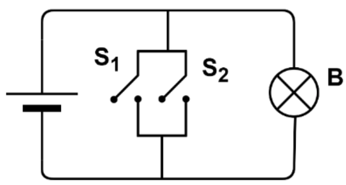
S	B
0	1
1	0

4. NAND Gate
NAND ද්වාරය



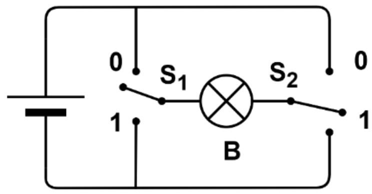
S_1	S_2	B
0	0	1
0	1	1
1	0	1
1	1	0

5. NOR Gate
NOR ද්වාරය



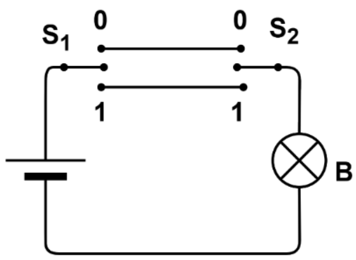
S_1	S_2	B
0	0	1
0	1	0
1	0	0
1	1	0

6. XOR Gate
XOR ද්වාරය



S_1	S_2	B
0	0	0
0	1	1
1	0	1
1	1	0

7. XNOR Gate
XNOR ද්වාරය



S_1	S_2	B
0	0	1
0	1	0
1	0	0
1	1	1

ictfromabc.com

Boolean algebra
බූලියානු වීජ ගණිතය

Boolean algebra is a mathematical framework used to manipulate and simplify logical expressions, especially in the design of digital circuits and computer systems.
බූලියානු වීජ ගණිතය යනු තාර්කික ප්‍රකාශන හැසිරවීමට සහ සරල කිරීමට, විශේෂයෙන් ඩිජිටල් පරිපථ සහ පරිගණක පද්ධති සැලසුම් කිරීමේදී භාවිතා කරන ගණිතමය රාමුවකි.

It operates with binary variables (0s and 1s) and is governed by a set of basic principles known as **postulates** or **axioms**.
එය ද්විමය විචල්‍යයන් (0 සහ 1) සමඟ ක්‍රියා කරන අතර එය පාලනය වන්නේ **postulates** හෝ **axioms** ලෙස හඳුන්වන මූලික මූලධර්ම සමූහයකිනි.

Postulates (Axioms) in Boolean Algebra
බූලියන් වීජ ගණිතයේ Postulates (Axioms)

Postulates are the fundamental assumptions in Boolean algebra that are accepted as true without proof.
Postulates යනු බූලියන් වීජ ගණිතයේ ඇති මූලික උපකල්පන වන අතර ඒවා සාක්ෂි නොමැතිව සත්‍ය ලෙස පිළිගනු ලැබේ.

These postulates define the basic behavior of Boolean operations and form the starting point for all further reasoning within the algebraic system.
මෙම postulates බූලියන් මෙහෙයුම්වල මූලික හැසිරීම් නිර්වචනය කරන අතර වීජීය පද්ධතිය තුළ වැඩිදුර තර්ක සඳහා ආරම්භක ලක්ෂ්‍යය සාදයි.

Importance of simplifying logic circuits using Boolean Laws
බූලියානු නීති භාවිතයෙන් තාර්කික පරිපථ සරල කිරීමේ වැදගත්කම

- **Reduces Complexity** – Fewer components make circuits easier to design and understand.
  සංකීර්ණතාව අඩු කරයි – අඩු සංරචක සංඛ්‍යාවක් මගින් පරිපථ සැලසුම් කිරීම සහ තේරුම් ගැනීම පහසු කරයි.

- **Lowers Cost** – Simplified circuits use fewer gates, reducing material and manufacturing costs.
  පිරිවැය අඩු කරයි - සරල කළ පරිපථ අඩු ද්වාර ගණනක් භාවිතා කරයි, මූලද්‍රව්‍ය හා නිෂ්පාදන පිරිවැය අඩු කරයි.

- **Improves Speed** – Less processing delay leads to faster operations.
  වේගය වැඩි දියුණු කරයි - අඩු සැකසුම් ප්‍රමාදය වේගවත් මෙහෙයුම් වලට මග පාදයි.

- **Saves Power** – Fewer components consume less energy.
  බල ශක්තිය ඉතිරි කරයි - අඩු සංරචක සංඛ්‍යාවක් නිසා අඩු ශක්තියක් වැය වේ.

- **Enhances Reliability** – Simple circuits have fewer failure points, increasing durability.
  විශ්වසනීයත්වය වැඩි කරයි - සරල පරිපථවල ඇති බිඳවැටුම් ලක්ෂ්‍ය අඩු වන අතර එමගින් කල්පැවැත්ම වැඩි කරයි.

- **Optimizes Space** – Requires less physical space on circuit boards.
  අවකාශය ප්‍රශස්ත කරයි - පරිපථ පුවරු මත අඩු භෞතික ඉඩක් අවශ්‍ය වේ.

Boolean Algebraic Laws

1. Double complement law
   ද්විත්ව ප්‍රතිලෝම න්‍යාය
   $\overline{\overline{A}} = A$

2. Idempotent law
   තදේවභාවී න්‍යාය
   $A + A = A$
   $A . A = A$

3. Identity law
   සර්වසාම්‍ය න්‍යාය
   $A . 0 = 0$
   $A . 1 = A$
   $A + 0 = A$
   $A + 1 = 1$

4. Inverse law / complement law
   ප්‍රතිලෝම න්‍යාය
   $A . \overline{A} = 0$
   $A + \overline{A} = 1$

5. Commutative law
   න්‍යාදේශ න්‍යාය
   $A . B = B . A$
   $A + B = B + A$

6. Associative law
   සංඝටන න්‍යාය
   $A . (B.C) = (A.B).C$
   $A + (B + C) = (A + B) + C$

7. Distributive law
   විඝටන න්‍යාය
   $A . (B + C) = A.B + A.C$
   $A + (B.C) = (A + B).(A + C)$

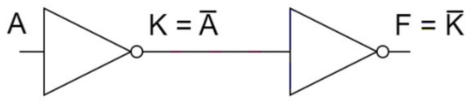
8. De Morgan's Law
   ද මුවාර් ප්‍රමේය
   $\overline{A.B} = \overline{A} + \overline{B}$
   $\overline{A + B} = \overline{A}.\overline{B}$

9. Redundancy law
   සමතිරික්ත න්‍යාය
   $A + A.B = A$
   $A + \overline{A}.B = A + B$

Double inverse law/Double complement law
ද්විත්ව ප්‍රතිලෝම න්‍යාය

Double inverse law states that inverting a variable twice will produce the same original variable again.
ද්විත්ව ප්‍රතිලෝම න්‍යාය පවසන්නේ විචල්‍යයක් දෙවරක් ප්‍රතිලෝම කිරීමෙන් මුල් විචල්‍යය නැවත ලැබෙන බවයි.

Let's take two NOT gates connected in series.
එකකට පසු එකක් සම්බනධ කල NOT ද්වාර දෙකක් සලකමු



A	K
0	1
1	0

K	F
1	0
0	1

From the above Boolean expressions
ඉහත බූලියන් ප්‍රකාශන වලින්

$K = \overline{A}$ (K is the inverse of A)
$F = \overline{K}$ (F is the inverse of K)
$F = \overline{\overline{A}}$ (F is the double inverse of A)

Looking at the truth tables for all instances,
සත්‍යතා වගු දෙස බලන විට සෑම අවස්ථාවකදීම,

$F = A$ (F looks the same as A)

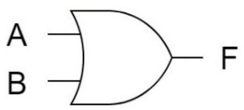
Because L.H.S of both above equations are the same it proves the double inverse law
ඉහත සමීකරණ දෙකේම L.H.S සමාන වන බැවින් එය ද්විත්ව ප්‍රතිලෝම න්‍යාය තහවුරු කරයි.

ictfromabc.com

Idempotent law (type 1)
තදේවභාවී න්‍යාය (පළමු ආකාරය)

Adding the same value again by it does not make any difference in Boolean algebra.
බූලීය වීජ ගණිතයේදී යම් අගයකට එම අගයම නැවත එකතු කිරීම මගින් වෙනසක් සිදු නොවේ

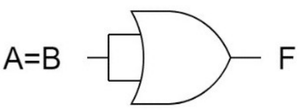
Boolean addition (OR operation) is represented by the OR gate.
බූලියන් එකතු කිරීම (OR මෙහෙයුම) OR ද්වාරය මගින් නිරූපණය කරයි



ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Input combination 1 and 4 represent the addition of the same value.
1 සහ 4 ආදාන සංයෝජන මගින් එකම අගය එකතු කිරීම නිරූපණය කරයි.

$F = A + B$
$F = A + A$ (When B = A at ICN 1 and 4)



ICN	A = B	F
1	0	0
4	1	1

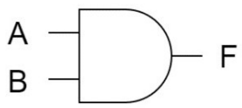
But F = A from the truth table

From the above simplified truth table, we can clearly see the output value is the same as the input. It proves the Idempotent law.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම ආදානයට සමාන බව අපට පැහැදිලිව දැකගත හැකිය. එමගින් තදේවභාවී න්‍යාය තහවුරු වේ.

Idempotent law (type 2)
තදේවභාවී න්‍යාය (දෙවන ආකාරය)

Multiplying the same value again by it does not make any difference in Boolean algebra.
බූලීය වීජ ගණිතයේදී යම් අගයකින් එම අගයම ගුණ කිරීම කිරීම මගින් වෙනසක් සිදු නොවේ

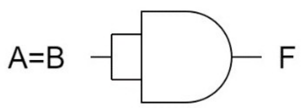
Boolean multiplication (AND operation) is represented by the AND gate.
බූලියන් ගුණ කිරීම (AND මෙහෙයුම) AND ද්වාරය මගින් නිරූපණය කරයි



ICN	A	B	F
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Input combination 1 and 4 represent the multiplication of the same value.
1 සහ 4 ආදාන සංයෝජන මගින් එකම අගය ගුණ කිරීම නිරූපණය කරයි.

$F = A.B$
$F = A.A$ (When B = A at ICN 1 and 4)



ICN	A = B	F
1	0	0
4	1	1

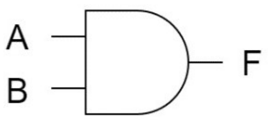
But F = A from the truth table

From the above simplified truth table, we can clearly see the output value is the same as the input. It proves the Idempotent law.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම ආදානයට සමාන බව අපට පැහැදිලිව දැකගත හැකිය. එමගින් තදේවභාවී න්‍යාය තහවුරු වේ.

Identity law (type 1)
සර්වසාම්‍ය න්‍යාය (පළමු ආකාරය)

Multiplying something by 0 in Boolean algebra always produces 0.
බූලියන් වීජ ගණිතයේ යමක් 0 න් ගුණ කිරීමෙන් සෑම විටම 0 ලැබේ

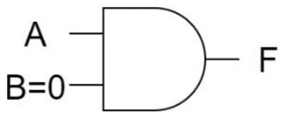
Boolean multiplication (AND operation) is represented by the AND gate.
බූලියන් ගුණ කිරීම (AND මෙහෙයුම) AND ද්වාරය මගින් නිරූපණය කරයි



ICN	A	B	F
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Input combination 1 and 3 represent multiplication by zero.
1 සහ 3 ආදාන සංයෝජන මගින් 0 න් ගුණ කිරීම නිරූපණය කරයි.

$F = A.B$
$F = A.0$ (When B = 0 at ICN 1 and 3)



ICN	A	B = 0	F
1	0	0	0
3	1	0	0

But F = 0 from the truth table

From the above simplified truth table, we can clearly see the output is 0. It proves the Identity law.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම 0 බව පෙනේ එමගින් සර්වසාම්‍ය න්‍යාය තහවුරු වේ.

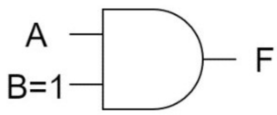
Identity law (type 2)
සර්වසාම්‍ය න්‍යාය (දෙවන ආකාරය)

Multiplying a value by 1 in Boolean algebra always produces the same value again
බූලියන් වීජ ගණිතයේ යමක් 1 න් ගුණ කිරීමෙන් සෑම විටම නැවත එම අගයම ලැබේ

ICN	A	B	F
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Input combination 2 and 4 represent multiplication by 1.
2 සහ 4 ආදාන සංයෝජන මගින් 1 න් ගුණ කිරීම නිරූපණය කරයි.

$F = A.B$
$F = A.1$ (When B = 1 at ICN 2 and 4)



ICN	A	B = 1	F
2	0	1	0
4	1	1	1

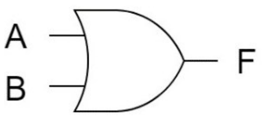
But F = A from the truth table

From the above simplified truth table, we can clearly see the output is A. It proves the Identity law.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම A බව පෙනේ

Identity law (type 3)
සර්වසාම්‍ය න්‍යාය (තුන්වන ආකාරය)

Adding 0 to something in Boolean algebra doesn't change it
බූලියන් වීජ ගණිතයේ දී යමකට 0 එකතු කල විට එය වෙනස් නොවේ

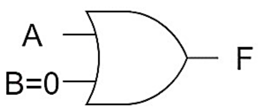
Boolean addition (OR operation) is represented by the OR gate.
බූලියන් එකතු කිරීම (OR මෙහෙයුම) OR ද්වාරය මගින් නිරූපණය කරයි



ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Input combination 1 and 3 represent adding zero.
1 සහ 3 ආදාන සංයෝජන මගින් 0 ක් එකතු කිරීම නිරූපණය කරයි

$F = A + B$
$F = A + 0$ (When B = 0 at ICN 1 and 3)



ICN	A	B = 0	F
1	0	0	0
3	1	0	1

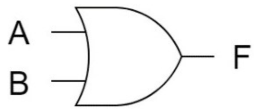
But F = A from the truth table

From the above simplified truth table, we can clearly see the output is A.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම A බව පෙනේ

It proves the Identity law.
එමගින් සර්වසාම්‍ය න්‍යාය තහවුරු වේ.

Identity law (type 4)
සර්වසාම්‍ය න්‍යාය (හතරවන ආකාරය)

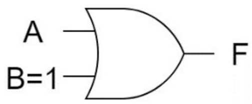
This law states that adding 1 to something in Boolean algebra always gives 1
බූලියන් වීජ ගණිතයේ යමකට 1 එකතු කල විට සෑමවිටම 1 ලැබේ



ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Input combination 2 and 4 represent addition of 1.
2 සහ 4 ආදාන සංයෝජන මගින් 1 ක් එකතු කිරීම නිරූපණය කරයි.

$F = A + B$
$F = A + 1$ (When B = 1 at ICN 2 and 4)



ICN	A	B = 0	F
1	0	1	1
3	1	1	1

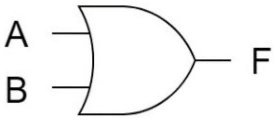
But F = A from the truth table

From the above simplified truth table, we can clearly see the output is 1.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම 1 බව පෙනේ

It proves the Identity law.
එමගින් සර්වසාම්‍ය න්‍යාය තහවුරු වේ.

Inverse law (type 1)
ප්‍රතිලෝම න්‍යාය (පළමු ආකාරය)

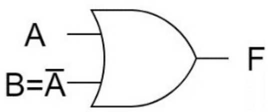
Adding its own complement to a variable always sums to 1 in Boolean algebra.
බූලියානු වීජ ගණිතයේදී විචල්‍යයකට තමනගේම අනුපූරකය එකතු කිරීම මගින් සැමවිටම 1 ලබාදේ



ICN	A	B	F
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Input combination 2 and 3 represent adding its complement.
2 සහ 3 ආදාන සංයෝජන මගින් අනුපූරකය එකතු කිරීම නිරූපණය කරයි

$F = A + B$
$F = A + \overline{A}$ (When B = $\overline{A}$ at ICN 2 and 3)



ICN	A	B = $\overline{A}$	F
2	0	1	1
3	1	0	1

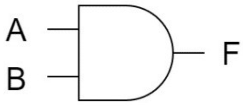
But F = 1 from the truth table

From the above simplified truth table, we can clearly see the output is 1
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම 1 බව පෙනේ

It proves the Inverse law.
එමගින් ප්‍රතිලෝම න්‍යාය තහවුරු වේ.

Inverse law (type 2)
ප්‍රතිලෝම න්‍යාය (දෙවන ආකාරය)

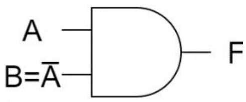
Multiplying its own complement from a variable always produces 0 in Boolean algebra.
බූලියානු වීජ ගණිතයේදී විචල්‍යක් තමනගේම අනුපූරකය මගන් ගුණ කිරීමේදී සැමවිටම 0 ලබාදේ



ICN	A	B	F
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Input combination 2 and 3 represent multiplication from its complement.
2 සහ 3 ආදාන සංයෝජන මගින් අනුපූරකයෙන් ගුණ කිරීම නිරූපණය කරයි

$F = A.B$
$F = A.\overline{A}$ (When B = $\overline{A}$ at ICN 2 and 3)



ICN	A	B = $\overline{A}$	F
2	0	1	0
3	1	0	0

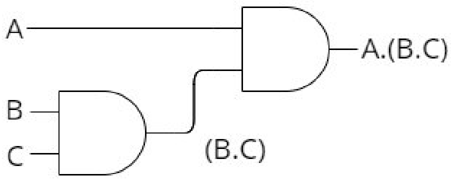
But F = 0 from the truth table

From the above simplified truth table, we can clearly see the output is always 0. It proves the Inverse law.
ඉහත සුළු කරන ලද සත්‍යතා වගුවෙන් ප්‍රතිදානයේ අගය සෑමවිටම 0 බව පෙනේ එමගින් ප්‍රතිලෝම න්‍යාය තහවුරු වේ.

Associative law (type 1)
සංඝටන න්‍යාය (පළමු ආකාරය)

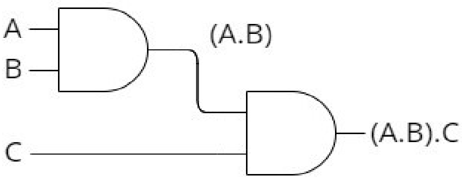
This law states that the order of multiplying Boolean variables does not affect the outcome.
බූලීය විචල්‍ය ගුණ කරන අනුපිළිවෙළ අවසාන ප්‍රතිදානයට බලපාන්නේ නැති බව මෙම නීතියෙන් කියැවේ

Left Hand Side Expression



ICN	A	B	C	B.C	P=A.(B.C)
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	1	1	1	0
5	1	0	0	0	0
6	1	0	1	0	0
7	1	1	0	0	0
8	1	1	1	1	1

Right Hand Side Expression



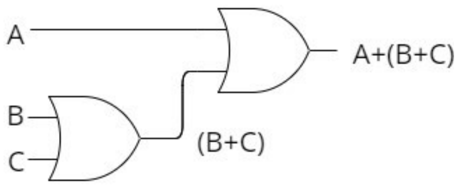
ICN	A	B	C	A.B	Q=(A.B).C
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	1	0	0	0	0
6	1	0	1	0	0
7	1	1	0	1	0
8	1	1	1	1	1

P=Q, so A.(B.C)=(A.B).C

Associative law (type 2)
සංඝටන න්‍යාය (දෙවන ආකාරය)

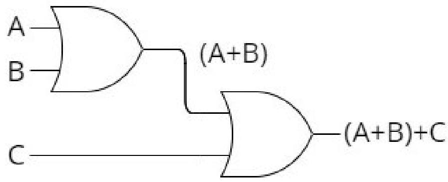
This law states that the order of adding Boolean variables does not affect the outcome
බූලීය විචල්‍ය එකතු කරන අනුපිළිවෙළ අවසාන ප්‍රතිදානයට බලපාන්නේ නැති බව මෙම නීතියෙන් කියැවේ

Left Hand Side Expression



ICN	A	B	C	B+C	P=A+(B+C)
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	0	1	1
4	0	1	1	1	1
5	1	0	0	0	1
6	1	0	1	1	1
7	1	1	0	1	1
8	1	1	1	1	1

Right Hand Side Expression

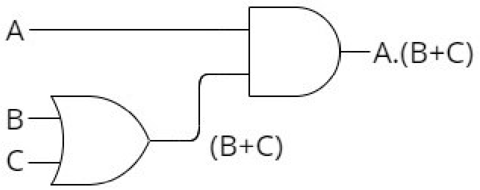


ICN	A	B	C	A+B	Q=(A+B)+C
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	1	1
4	0	1	1	1	1
5	1	0	0	1	1
6	1	0	1	1	1
7	1	1	0	1	1
8	1	1	1	1	1

P=Q, so A+(B+C)=(A+B)+C

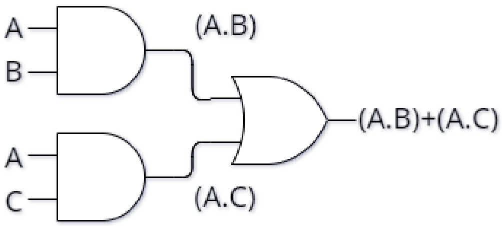
Distributive law
විඝටන න්‍යාය
A.(B+C) = A.B + A.C

Left Hand Side Expression



ICN	A	B	C	(B+C)	P=A.(B+C)
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	0
4	0	1	1	1	0
5	1	0	0	0	0
6	1	0	1	1	1
7	1	1	0	1	1
8	1	1	1	1	1

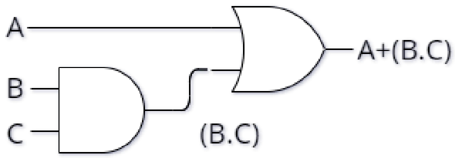
Right Hand Side Expression



ICN	A	B	C	(B+C)	Q=(A+B)+C
1	0	0	0	0	0
2	0	0	1	1	0
3	0	1	0	1	0
4	0	1	1	1	0
5	1	0	0	0	0
6	1	0	1	1	1
7	1	1	0	1	1
8	1	1	1	1	1

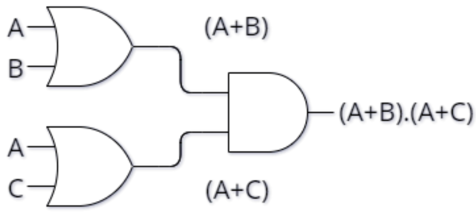
Distributive law
විඝටන න්‍යාය
A+(B.C) = (A+B).(A+C)

Left Hand Side Expression



ICN	A	B	C	B.C	P=A+B.C
1	0	0	0	0	0
2	0	0	1	0	0
3	0	1	0	0	0
4	0	1	1	1	1
5	1	0	0	0	1
6	1	0	1	0	1
7	1	1	0	0	1
8	1	1	1	1	1

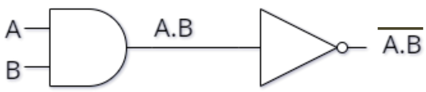
Right Hand Side Expression



ICN	A	B	C	A+B	A+C	Q=(A+B).(A+C)
1	0	0	0	0	0	0
2	0	0	1	0	1	0
3	0	1	0	1	0	0
4	0	1	1	1	1	1
5	1	0	0	1	1	1
6	1	0	1	1	1	1
7	1	1	0	1	1	1
8	1	1	1	1	1	1

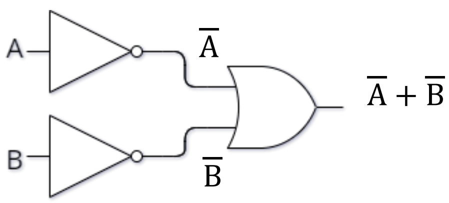
De Morgan’s Law
ද මුවාර් ප්‍රමේය
$\overline{\mathbf{A.B}} = \overline{\mathbf{A}} + \overline{\mathbf{B}}$

Left Hand Side Expression



ICN	A	B	A.B	$\overline{A.B}$
1	0	0	0	1
2	0	1	0	1
3	1	0	0	1
4	1	1	1	0

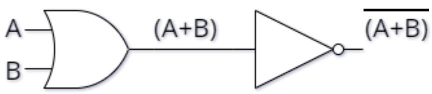
Right Hand Side Expression



ICN	A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
1	0	0	1	1	1
2	0	1	1	0	1
3	1	0	0	1	1
4	1	1	0	0	0

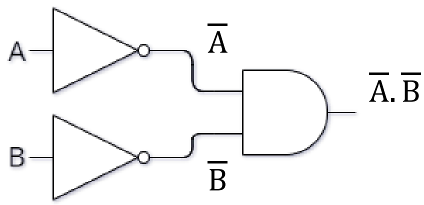
De Morgan’s Law
ද මුවාර් ප්‍රමේය
$\overline{\mathbf{A} + \mathbf{B}} = \overline{\mathbf{A}}.\overline{\mathbf{B}}$

Left Hand Side Expression



ICN	A	B	A+B	$\overline{A + B}$
1	0	0	0	1
2	0	1	1	0
3	1	0	1	0
4	1	1	1	0

Right Hand Side Expression



ICN	A	B	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
1	0	0	1	1	1
2	0	1	1	0	0
3	1	0	0	1	0
4	1	1	0	0	0

SUMMARY

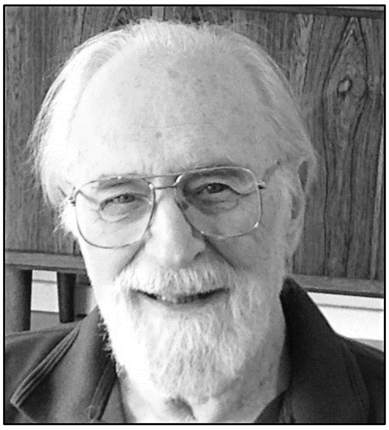
Postulates උපකල්පන
Fundamental assumptions defining Boolean algebra operations.
බූලියන් වීජ ගණිත මෙහෙයුම් නිර්වචනය කරන මූලික උපකල්පනයි.

Axioms ප්‍රත්‍යක්ෂ/ස්වයං සිද්ධි
Self-evident truths forming the basis of Boolean algebra.
බූලියන් වීජ ගණිතයේ පදනම සාදන ස්වයං-පැහැදිලි සත්‍යයන්ය.

Laws/Theorems නීති/ප්‍රමේය
Derived rules used to simplify and manipulate Boolean expressions.
බූලියන් ප්‍රකාශන සරල කිරීමට සහ හැසිරවීමට භාවිතා කරන ව්‍යුත්පන්න නීති වේ.

Karnaugh Maps
කානෝ සිතියම්

Maurice Karnaugh developed Karnaugh Maps (K-maps) in 1953.
මොරිස් කානෝ විසින් 1953 දී කානෝ සිතියම් (K-සිතියම්) නිර්මාණය කළේය.



He worked at Bell Labs, a leading research center in telecommunications and digital systems.
ඔහු විදුලි සංදේශ සහ ඩිජිටල් පද්ධති පිළිබඳ ප්‍රමුඛ පර්යේෂණ මධ්‍යස්ථානයක් වන බෙල් ලැබ්ස් හි සේවය කළේය.

K-maps were primarily designed to simplify the logic behind digital circuits used in circuit switching.
Circuit switching වලදී භාවිතා කරන අංකිත පරිපථ පිටුපස ඇති තර්කනය සරල කිරීම සඳහා කානෝ සිතියම් මූලික වශයෙන් නිර්මාණය කර ඇත.

Circuit switching is a communication method where a dedicated path is established between two endpoints before data transmission and remains reserved for the entire session.
Circuit switching යනු දත්ත සම්ප්‍රේෂණයට පෙර අන්ත ලක්ෂ්‍ය දෙකක් අතර කැපවූ මාර්ගයක් ස්ථාපිත කර මුළු සැසිය පුරාම එය වෙන් කර පවතින සන්නිවේදන ක්‍රමයකි.

K-maps simplified complex expressions by grouping terms efficiently, reducing the number of required logic gates.
කානො සිතියම් මඟින් ප්‍රකාශන කාර්යක්ෂමව කාණ්ඩගත කිරීමෙන් සංකීර්ණ ප්‍රකාශන සරල කරමින් පරිපථ සෑදීමට අවශ්‍ය තාර්කික ද්වාර ගණන අඩු කළේය.

Over time, K-maps became the standard for simplifying Boolean expressions in digital electronics.
කාලයත් සමඟ, අංකිත ඉලෙක්ට්‍රොනික තාක්ෂණයේදී බූලියන් ප්‍රකාශන සරල කිරීම සඳහා කානෝ සිතියම් ප්‍රමිතිය බවට පත්විය.

K-Maps vs. Boolean Laws

Aspect	Karnaugh Map (K-Map)	Boolean Algebra Laws
Complexity සංකීර්ණත්වය	Simple for up to 5 variables; harder beyond that විචල්‍ය 5ක් දක්වා සරල වේ; ඉන් පසු අපහසුය	Can be complex but scales better for large expressions සංකීර්ණ විය හැකි නමුත් විශාල ප්‍රකාශන සඳහා වඩා හොඳ පරිමාණයකි
Speed වේගය	Faster for small expressions (2-5 variables) කුඩා ප්‍රකාශන සඳහා වේගවත්ය	Can be time-consuming without practice පුහුණු වීමකින් තොරව කාලය ගත විය හැකිය
Ease of Use භාවිත කිරීමේ පහසුව	Intuitive and visual බුද්ධිමය සහ දෘශ්‍ය වේ	Requires understanding of Boolean theorems බූලියානු නිතී පිළිබඳ අවබෝධයක් අවශ්‍ය වේ
Best For වඩාත්ම ගැලපෙන්නේ	Circuit optimization and logic design පරිපථ ප්‍රශස්තිකරණය සහ තාර්කික නිර්මාණය	Theoretical proofs and mathematical simplifications න්‍යායාත්මක සාධනයන් සහ ගණිතමය සරල කිරීම්

ictfromabc.com

2 input variable k maps
ආදාන දෙකේ කානෝ සිතියම්

B \ A	$\overline{A}$ 0	A 1
$\overline{B}$ 0	$\overline{A}.\overline{B}$	$A.\overline{B}$
B 1	$\overline{A}.B$	$A.B$

B \ A	$\overline{A}$ 0	A 1
$\overline{B}$ 0	$A+B$	$\overline{A}+B$
B 1	$A+\overline{B}$	$\overline{A}+\overline{B}$

GREY CODE
(REFLECTED BINARY CODE)

Frank Gray, a researcher at Bell Labs, patented the concept of Grey code in 1953.
1953 දී බෙල් විද්‍යාගාර තුල පර්යේෂකයෙකු වූ ෆ්‍රෑන්ක් ග්‍රේ විසින් Grey කේත සංකල්පයට පේටන්ට් බලපත්‍රය ලබා ගත්තේය.

Frank Gray

This is a way of ordering binary numbers where only one-bit changes between any twoconsecutive values.
මෙය ඕනෑම අඛණ්ඩ අගයන් දෙකක් අතර එක්-බිට් එකක් පමණක් වෙනස් වන පරිදි ද්විමය සංඛ්‍යා පෙලගැස්වීමේ ක්‍රමයකි.

According to Grey code, only one variable is allowed to change when shifting between two nearby cells horizontally or vertically in a K-map.
කානෝ සිතියමක අනුයාත කෝෂ දෙකක් අතර තිරස්ව හෝ සිරස්ව ගමන් කිරීමේදී වෙනස් වීමට අවකාශය ඇත්තේ එක් විචල්‍යයකට පමණක් යැයි ග්‍රේ කේතයෙන් කියැවේ.

00	**01**	**10**	**11**	✗
00	01	11	10	✓
01	11	10	00	✓
11	10	00	01	✓
10	00	01	11	✓

3 input variable k maps
ආදාන තුනේ කානෝ සිතියම්

C \ AB	00	01	11	10
0	1	3	7	5
1	2	4	8	6

B \ AC	00	01	11	10
0	1	2	6	5
1	3	4	8	7

A \ BC	00	01	11	10
0	1	2	4	3
1	5	6	8	7

BC \ A	0	1
00	1	5
01	2	6
11	4	8
10	3	7

AC \ B	0	1
00	1	3
01	2	4
11	6	8
10	5	7

AB \ C	0	1
00	1	2
01	3	4
11	7	8
10	5	6

4 input variable k maps
ආදාන හතරේ කානෝ සිතියම්

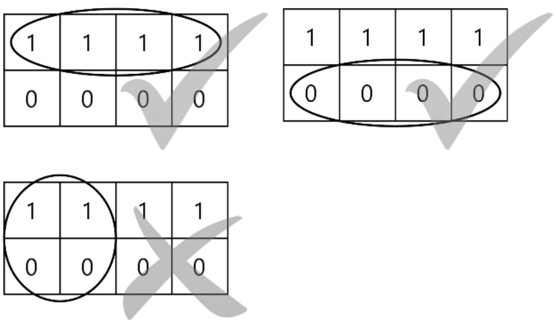
CD \ AB	00	01	11	10
00	1	5	13	9
01	2	6	14	10
11	4	8	16	12
10	3	7	15	11

ictfromabc.com

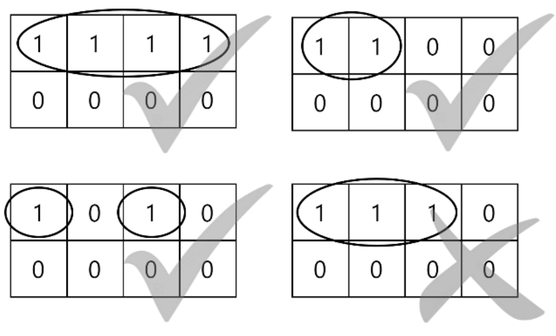
K-map grouping rules.
කානෝ සිතියම් වල කාන්ඩ සෑදීම සම්බන්ද නීති

1. All outputs of a truth table, 1s and 0s, should be represented in the k map.
සත්‍යතා වගුවේ පවතින 1 සහ 0 සියලුම සත්‍යතා වගුවේ පවතින 1 සහ 0 සියලුම ප්‍රතිදානකානෝ සිතියමක් තුල ලකුණු කරගත යුතුය.

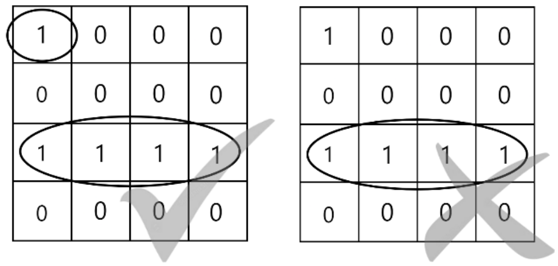
2. Only one type of element can be used tomake groups. 1s or 0s. groups cannot be made using different types of elements.
කාණ්ඩ සෑදීම සඳහා භාවිතා කල හැක්කේ එක් වර්ගයක අවයව පමණි. වෙනස් වර්ග වල අවයව කාණ්ඩ කල නොහැක. 1s or 0s



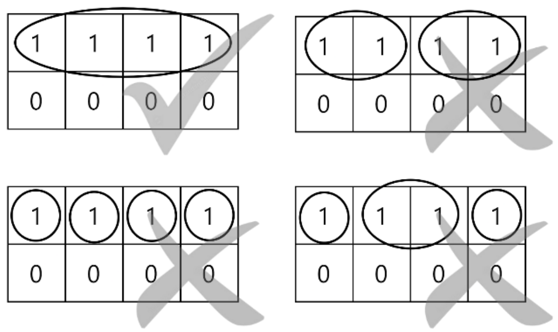
3. The number of elements in a group shouldalways be a power of 2.
කාණ්ඩයක පවතින අවයව ගණන සෑමවිටම 2 හි බලයකට සමාන විය යුතුය.



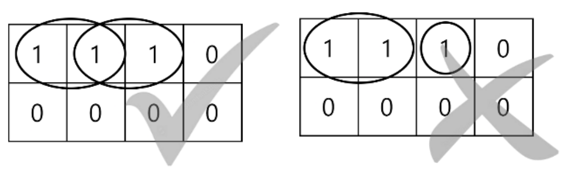
4. All elements of the selected type should be contained in a group.
කාණ්ඩ සෑදීම සඳහා තෝරාගන්නා වර්ගයේ අවයව සියල්ල කාණ්ඩ සෑදීම අවසානයේ යම් කාණ්ඩයකට අයිති විය යුතුය.



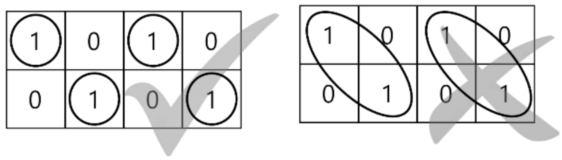
5. A group should be as large as possible, and the number of groups should be minimum.
කාණ්ඩයක් හැකිතාක් විශාල විය යුතු අතර කාණ්ඩ ගණන හැකිතාක් අවම විය යුතුය.



6. Elements which are already in one group can be shared with another group if needed.
කාණ්ඩයක් තුල දැනටමත් පවතින අවයව වෙනත් කාණ්ඩයක් සමග බෙදාහදාගැනීම සිදු කල හැකිය.

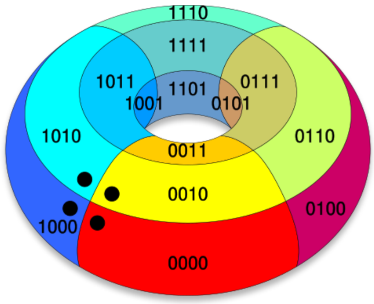


7. Groups cannot be diagonal.
විකර්ණ දිගේ කාණ්ඩ පැවතිය නොහැක.

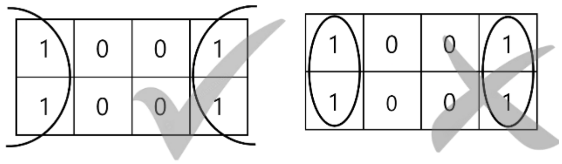


NOTE
A K-map is a torus
කානෝ සිතියමක් යනු ටෝරස් එකකි

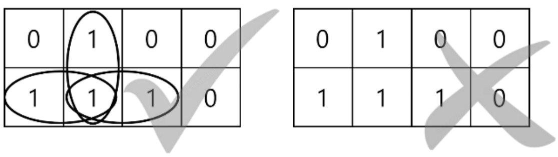
0000	0100	1100	1000
0001	0101	1101	1001
0011	0111	1111	1011
0010	0110	1110	1010

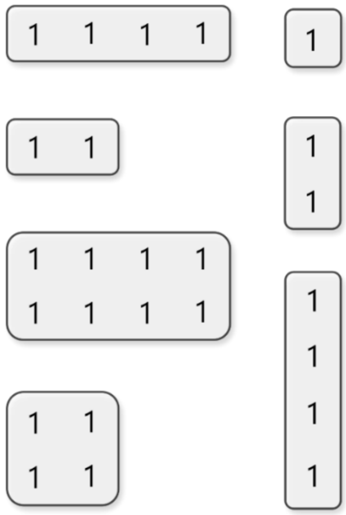


8. Groups can be wrapped around.
කාණ්ඩයක් කානෝ සිතියමක් වටා එතී පැවතිය හැක

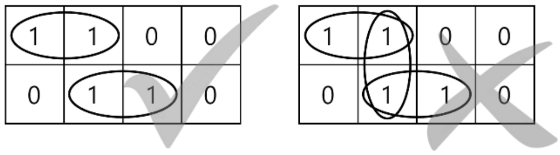


9. A group should always be horizontal or vertical. Its shape should be square or rectangular.
කාණ්ඩයක් සෑමවිටම තිරස්ව හෝ සිරස්ව පැවතිය යුතු අතර එහි හැඩය සමචතුරස්‍රාකාර හෝ සෘජුකෝණාස්‍රාකාර විය යුතුය.

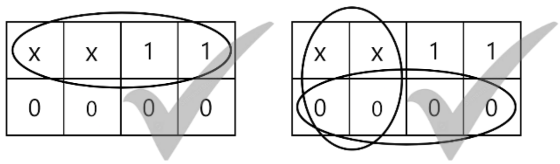




NOTE: Subgroups (groups where all their elements belong to one other group) may be created during grouping. They should be removed as it increases the number of groupsand makes the simplified Boolean expression bigger.
කාණ්ඩ සෑදීමේදී තමාගේ අවයව සියල්ල වෙනත් කාණ්ඩ එකකට හෝ අයිති වන අනු කාණ්ඩ නිර්මාණය විය හැකිය. මෙමගින් කාණ්ඩ ගණන වැඩිවී සුලු කරන ලද බූලීය ප්‍රකාශය සංකීර්ණ වන බැවින් ඒවා සොයා මකා දැමිය යුතුය.

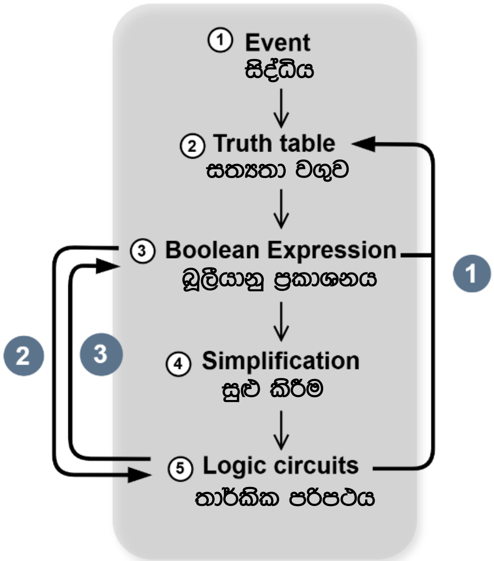


NOT IN SYLLABUS
Don't care conditions (where we don't care whether the output of that input combination is1 or 0) can be grouped or neglected considering either 1s or 0s as necessary for the instance.
යම් ආදාන සංයෝජනයකට අදාල ප්‍රතිදානය සත්‍යද අසත්‍යද නොසලකන අවස්ථා හෙවත් Don't care අවස්ථා පවතින විට කාණ්ඩ සෑදීමේදී ඒවා අවශ්‍ය පරිදි කාණ්ඩ සඳහා 1 හෝ 0 ලෙස සැලකිය හැකිය.



Designs simple digital circuits using logic gates
තාර්කික ද්වාර භාවිතයෙන් සරල අංකිත පරිපථ නිර්මාණය කරයි.

According to the syllabus, the steps involved in designing a logic circuit for a given scenario is usually as follows.
විෂය නිර්දේශයට අනුව, දී ඇති අවස්ථාවක් සඳහා තාර්කික පරිපථයක් නිර්මාණය කිරීමේදී අනුගමනය කළ යුතු පියවර සාමාන්‍යයෙන් පහත පරිදි වේ.



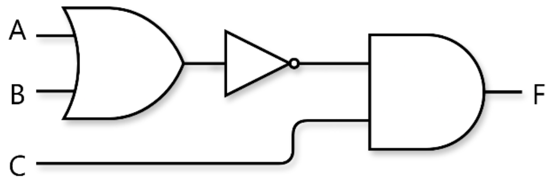
However, you should be able to,
කෙසේ වෙතත් ඔබට,

1. Obtain the truth table for a given logic circuit and a Boolean expression.
දී ඇති තාර්කික පරිපථයක් සහ බූලීයානු ප්‍රකාශනයක් සඳහා සත්‍යතා වගුව ලබා ගැනීමට හැකි විය යුතුය.

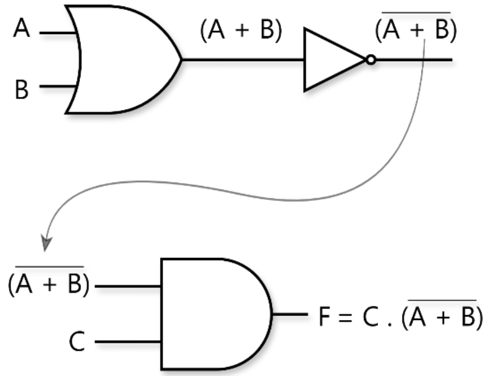
2. Draw the logic circuit for a given Boolean expression.
දී ඇති බූලියානු ප්‍රකාශනයක් සඳහා තාර්කික පරිපථය ඇඳීමට හැකි විය යුතුය.

3. Derive the Boolean expression from a given Logic circuit.
දී ඇති තාර්කික පරිපථයක් භාවිතයෙන් බූලීයානු ප්‍රකාශනය ව්‍යුත්පන්න කළ හැකි විය යුතුය.

Designing a Boolean expression for a given logic circuit.
දී ඇති තාර්කික පරිපථයක් සඳහා බූලියානු ප්‍රකාශනයක් නිර්මාණය කිරීම.



Let's derive a Boolean expression for the output F.
F ප්‍රතිදානය සඳහා බූලියානු ප්‍රකාශනයක් ව්‍යුත්පන්න කරමු.



Designing a logic circuit for a given Boolean expression.
දී ඇති බූලියානු ප්‍රකාශනයක් සඳහා තාර්කික පරිපථයක් නිර්මාණය කිරීම.

A logic circuit is derived from a Boolean expression by methodically "peeling away" its operators according to a fixed hierarchy.
බූලියානු ප්‍රකාශනයකින් තාර්කික පරිපථයක් ව්‍යුත්පන්න කර ගනු ලබන්නෙ නියත ධූරාවලියකට අනුව එහි ඇති මෙහෙයවන ක්‍රමානුකූලව "ඉවත් කිරීම" මගිනි.

In this approach, the operators are handled one level at a time, always treating the remaining expression as a fresh entity once an operator is removed.
මෙම ප්‍රවේශයේදී, මෙහෙයවන එක් මට්ටමකින් හසුරුවනු ලබන අතර, යම් මෙහෙයවනයක් ඉවත් කළ පසු ඉතිරි වන ප්‍රකාශනය සෑමවිටම නව ප්‍රකාශයක් ලෙස සලකනු ලබයි.

The operator precedence is defined as follows:
මෙහෙයවන ප්‍රමුඛතාවය පහත පරිදි අර්ථ දැක්වේ:

Bar (NOT):
When a bar is seen over an entire expression or subexpression, a NOT gate is drawn to represent the inversion.
සම්පූර්ණ ප්‍රකාශනයක් හෝ උප ප්‍රකාශනයක් මත bar එකක් පෙනෙන විට, ප්‍රතිලෝමය නිරූපණය කිරීම සඳහා NOT ද්වාරයක් අඳිනු ලැබේ.

It is important that any variables or subexpressions under a bar are not modified until that bar is removed.
Bar එකක් යටතේ ඇති ඕනෑම විචල්‍යයක් හෝ උප ප්‍රකාශනයක් එම bar එක ඉවත් කරන තෙක් වෙනස් නොකිරීම ඉතා වැදගත් වේ.

Plus (OR):
After all full bars have been processed, the expression is scanned for plus signs.
සියලුම full bars සැකසීමට ලක් කළ පසු, ප්‍රකාශනයේ ඇති ධන ලකුණු සොයා ගනු ලැබේ.

An OR gate is used for any plus operation that is not hidden inside brackets or under another bar.
වරහන් තුළ හෝ වෙනත් bar එකක් යටතේ සැඟවී නොමැති ඕනෑම ධන මෙහෙයුමක් සඳහා OR ද්වාරයක් භාවිතා කෙරේ.

Expressions within brackets are treated as atomic until they are isolated; nothing inside the bracket is "touched" until the rest of the expression has been derived.
වරහන් තුළ ඇති ප්‍රකාශන හුදකලා වන තෙක් ඒවා තනි ප්‍රකාශයක් ලෙස සලකනු ලැබේ; ප්‍රකාශනයේ ඉතිරි කොටස ව්‍යුත්පන්න වන තෙක් වරහන තුළ ඇති කිසිවක් "ස්පර්ශ" නොවේ.

Dot (AND):
Finally, the dot (AND) operations are handled.
අවසාන වශයෙන්, තිත් (AND) මෙහෙයුම් හසුරුවනු ලැබේ.

Again, only operators that are at the outer level (i.e., not nested inside brackets or under a bar) are processed.
නැවත වතාවක්, බාහිර මට්ටමේ (එනම්, වරහන් තුළ හෝ bar එකක් යටතේ තබා නොමැති) මෙහෙයවන පමණක් සකසනු ලැබේ.

The process is iterative:
මෙම ක්‍රියාවලිය පුනරාවර්තනය වේ:

The entire expression is examined for the highest-priority operator (NOT).
ඉහළම ප්‍රමුඛතා මෙහෙයවනය (NOT) සඳහා සම්පූර්ණ ප්‍රකාශනයම පරීක්ෂා කරනු ලැබේ.

Once a NOT operation is extracted (by drawing a NOT gate), the modified expression is re-evaluated as a new expression.
NOT මෙහෙයුමක් උපුටා ගත් පසු (NOT ද්වාරයක් ඇඳීමෙන්), වෙනස් කළ ප්‍රකාශනය නව ප්‍රකාශනයක් ලෙස නැවත ඇගයීමට ලක් කෙරේ.

This same procedure is applied next to OR operators, and then to AND operators, until the full expression is converted into a network of logic gates.
සම්පූර්ණ ප්‍රකාශනය තාර්කික ද්වාර ජාලයක් බවට පරිවර්තනය වන තෙක් මීලගට OR මෙහෙයවන වෙතටත් ඉන් පසුව AND මෙහෙයවනය වෙතටත් මෙම ක්‍රියා පිළිවෙලම සිදු කරනු ලැබේ.

ictfromabc.com

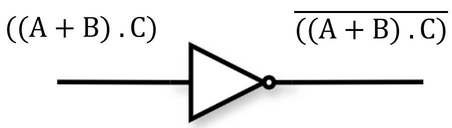
For instance, consider the expression:
උදාහරණයක් ලෙස, පහත ප්‍රකාශනය සලකා බලන්න:

$\overline{((A + B) . C)}$

The derivation proceeds as follows:
ව්‍යුත්පන්න ක්‍රියාවලිය පහත පරිදි වේ:

The outermost operator is a bar over the entire expression
පිටතම මෙහෙයවනය යනු සම්පූර්ණ ප්‍රකාශනය මත ඇති bar එකයි.

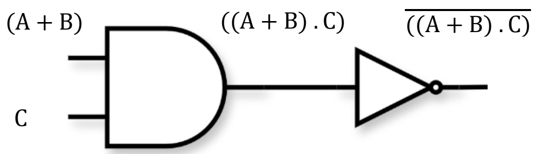
A NOT gate is drawn for the whole expression.
එබැවින් සම්පූර්ණ ප්‍රකාශනය සඳහාම NOT ද්වාරයක් අඳිනු ලැබේ.



The expression $((A + B) . C)$ is now treated as a new expression.
$((A + B) . C)$ යන ප්‍රකාශනය දැන් නව ප්‍රකාශනයක් ලෙස සලකනු ලැබේ.

The expression (A+B).C shows an AND (dot) operation connecting the entire bracketed (A+B) and C.
(A+B).C ප්‍රකාශනය මඟින් සම්පූර්ණ වරහන් තුළ ඇති (A+B) සහ C සම්බන්ධ කරන AND (dot) මෙහෙයුමක් පෙන්වනු ලබයි.

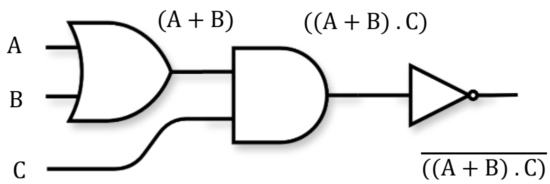
An AND gate is drawn that takes the isolated expression (A+B) and C as its inputs.
හුදකලා ප්‍රකාශනය වන (A+B) සහ C ආදාන ලෙස ගනු ලබන AND ද්වාරයක් අඳිනු ලැබේ.

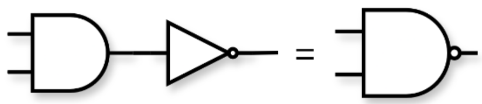


Notice that the bracketed (A+B) is treated as a single entity until it is isolated by the AND operation.
වරහන් තුළ අති (A+B) ප්‍රකාශය AND මෙහෙයුම මඟින් හුදකලා වන තෙක් තනි වස්තුවක් ලෙස සලකා ඇති බව හොඳින් නිරීක්ෂණය කරන්න.

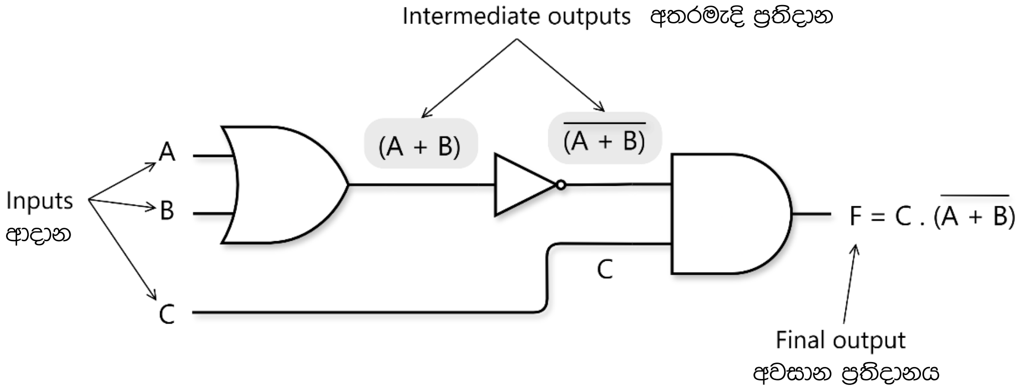
Now that (A+B) has been isolated as one of the inputs for the AND gate, it is processed internally.
දැන් (A+B) යන ප්‍රකාශය AND ද්වාරය සඳහා වන ආදාන වලින් එකක් ලෙස හුදකලා කර ඇති බැවින්, එය අභ්‍යන්තරව සකසනු ලැබේ.

The plus sign inside (A+B) is then recognized as an OR operation, and an OR gate is drawn to complete the internal logic of that expression.
(A+B) ඇතුළත ඇති ධන ලකුණ මීලඟට OR මෙහෙයුමක් ලෙස හඳුනා ගන්නා අතර, එම ප්‍රකාශනයේ අභ්‍යන්තර තර්කනය සම්පූර්ණ කිරීම සඳහා OR ද්වාරයක් අඳිනු ලැබේ.





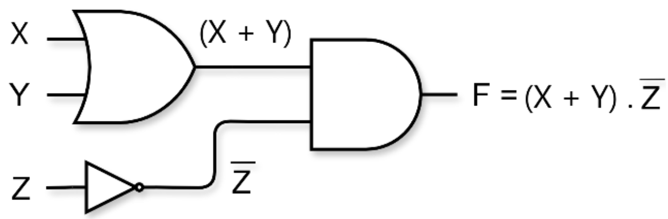
Creating a truth table for a given logic circuit
දී ඇති තාර්කික පරිපථයක් සඳහා සත්‍යතා වගුවක් නිර්මාණය කිරීම



ICN – Input Combination Number

ICN	A	B	C	$(A + B)$	$(\overline{A + B})$	$F = C\,.(\overline{A + B})$
1	0	0	0	0	1	0
2	0	0	1	0	1	1
3	0	1	0	1	0	0
4	0	1	1	1	0	0
5	1	0	0	1	0	0
6	1	0	1	1	0	0
7	1	1	0	1	0	0
8	1	1	1	1	0	0

EX.



ICN	X	Y	Z	$\overline{Z}$	$(X + Y)$	$F = (X + Y).\overline{Z}$
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	1	1	1
4	0	1	1	0	1	0
5	1	0	0	1	1	1
6	1	0	1	0	1	0
7	1	1	0	1	1	1
8	1	1	1	0	1	0

Minterms and Maxterms
ගුණිත පද සහ එකතු පද

Why do we need Minterms and Maxterms?
අපට ගුණිත පද සහ එකතු පද අවශ්‍ය වන්නේ ඇයි?

Minterms and maxterms ensure that any Boolean function can be expressed in a systematic and unambiguous way.
ගුණිත පද සහ එකතු පද මගින් ඕනෑම බූලියන් ශ්‍රිතයක් ක්‍රමානුකූලව සහ පැහැදිලි ආකාරයකින් ප්‍රකාශ කළ හැකි බව සහතික කරයි.

Helps in Simplification and Optimization
සරල කිරීම සහ ප්‍රශස්තිකරණය සඳහා උපකාරී වේ

Boolean expressions often need simplification to reduce the number of logic gates in a circuit.
පරිපථයක තාර්කික ද්වාර ගණන අඩු කිරීම සඳහා බූලියන් ප්‍රකාශන වලට බොහෝ විට සරල කිරීම අවශ්‍ය වේ.

Using minterms and maxterms, techniques like Karnaugh Maps (K-maps) can be applied for efficient simplification.
ගුණිත පද සහ එකතු පද භාවිතා කරමින්, කාර්යක්ෂමව සරල කිරීම් සිදු කිරීම සඳහා කානෝ සිතියම් (K-සිතියම්) වැනි ශිල්පීය ක්‍රම යොදා ගත හැකිය.

This results in faster, cheaper, and less power-consuming digital circuits.
මෙය වේගවත්, ලාභදායී සහ අඩු බලයක් පරිභෝජනය කරන අංකිත පරිපථවල නිපදවීමට හේතු වේ.

Design of Combinational Circuits
සංයුක්ත පරිපථ නිර්මාණය

They help in designing combinational logic circuits (e.g., adders, multiplexers, decoders) by providing a clear relationship between input combinations and output states.
මේවා මගින් ආදාන සංයෝජන සහ ප්‍රතිදාන තත්වයන් අතර පැහැදිලි සම්බන්ධතාවයක් ලබා දීමෙන් සංයුක්ත තාර්කික පරිපථ (උදා: ආකලක, multiplexers, විකේතක) නිර්මාණය කිරීමට උපකාරී වේ.

Analysis and Verification
විශ්ලේෂණය සහ සත්‍යාපනය

They make it easier to analyze and verify the correctness of a Boolean function.
මේවා බූලියානු ශ්‍රිතයක නිරවද්‍ය බව විශ්ලේෂණය කර සත්‍යාපනය කිරීම පහසු කරයි.

By examining the minterms or maxterms, you can determine whether the function covers all possible input combinations and whether it meets the design requirements.
ගුණිත පද හෝ එකතු පද පරීක්ෂා කිරීමෙන්, ශ්‍රිතය මගින් පැවතිය හැකි සියලුම ආදාන සංයෝජන ආවරණය කරන්නේද සහ එය සැලසුම් අවශ්‍යතා සපුරාලන්නේද යන්න තීරණය කිරීමට ඔබට හැකිය.

Conversion Between SOP and POS
SOP සහ POS අතර පරිවර්තනය

Allow easy conversion between SOP and POS forms, which is useful depending on the type of logic gates available or the design constraints.
SOP සහ POS ආකෘති අතර පහසු පරිවර්තනයකට ඉඩ ලබා දෙන අතර එය භාවිතා කිරීමට ඇති තාර්කික ද්වාර වර්ගය හෝ සැලසුම් සීමාවන් මත පදනම්ව ප්‍රයෝජනවත් වේ.

Minterms
ගුණිත පද

In a truth table, minterms represent how inputs must be multiplied to obtain the output 1 corresponding to a given input.
සත්‍යතා වගුවක යම් ආදානයකට අදාල ප්‍රතිදානය 1 ලැබීම සදහා ආදාන ගුණ කල යුතු ආකාරය ගුණිතපද මගින් නිරූපනය කරයි.

Minterms are used to represent a Boolean function in its Sum of Products (SOP) form.
බූලියානු ශ්‍රිතයක් එහි ගුණිතයන්ගේ එකතුව (SOP) ආකාරයෙන් නිරූපණය කිරීමට Minterms භාවිතා වේ.

Each minterm corresponds to a single row in the truth table where the function is true (1).
සෑම minterm එකක්ම සත්‍යතා වගුවේ ශ්‍රිතය true (1) වන තනි පේළියකට අනුරූප වේ.

Minterms of 2 inputs
ආදාන 2 ගුණිතපද

ICN	A	B	F
1	0	0	$m_0 = \overline{A}.\overline{B}$
2	0	1	$m_1 = \overline{A}.B$
3	1	0	$m_2 = A.\overline{B}$
4	1	1	$m_3 = A.B$

Minterms of 3 inputs
ආදාන 3 ගුණිතපද

ICN	A	B	C	F
1	0	0	0	$m_0 = \overline{A}.\overline{B}.\overline{C}$
2	0	0	1	$m_1 = \overline{A}.\overline{B}.C$
3	0	1	0	$m_2 = \overline{A}.B.\overline{C}$
4	0	1	1	$m_3 = \overline{A}.B.C$
5	1	0	0	$m_4 = A.\overline{B}.\overline{C}$
6	1	0	1	$m_5 = A.\overline{B}.C$
7	1	1	0	$m_6 = A.B.\overline{C}$
8	1	1	1	$m_7 = A.B.C$

Maxterms
එකතු පද

In a truth table, Maxterms represent how the inputs must be added to get the output 0 for a given input.
සත්‍යතා වගුවක යම් ආදානයකට අදාල ප්‍රතිදානය 0 ලැබීම සදහා ආදාන එකතු කල යුතු ආකාරය එකතු පද මගින් නිරූපනය කරයි.

Maxterms are used to represent a Boolean function in its Product of Sums (POS) form.
Maxterms භාවිතා කරනුයේ බූලියානු ශ්‍රිතයක් එහි එකතූන්ගේ ගුණිත ප්‍රකාශනයක් (POS) ආකාරයෙන් නියෝජනය කිරීමටය.

Each maxterm corresponds to a single row in the truth table where the function is false (0).
සෑම උපරිම වාරයක්ම සත්‍යතා වගුවේ ශ්‍රිතය false (0) වන තනි පේළියකට අනුරූප වේ.

Maxterms of 2 inputs
ආදාන 2 එකතුපද

ICN	A	B	F
1	0	0	$M_0 = A + B$
2	0	1	$M_1 = A + \overline{B}$
3	1	0	$M_2 = \overline{A} + B$
4	1	1	$M_3 = \overline{A} + \overline{B}$

Maxterms of 3 inputs
ආදාන 3 එකතුපද

ICN	A	B	C	F
1	0	0	0	$M_0 = A + B + C$
2	0	0	1	$M_1 = A + B + \overline{C}$
3	0	1	0	$M_2 = A + \overline{B} + C$
4	0	1	1	$M_3 = A + \overline{B} + \overline{C}$
5	1	0	0	$M_4 = \overline{A} + B + C$
6	1	0	1	$M_5 = \overline{A} + B + \overline{C}$
7	1	1	0	$M_6 = \overline{A} + \overline{B} + C$
8	1	1	1	$M_7 = \overline{A} + \overline{B} + \overline{C}$

Minterms & Maxterms

ICN	A	B	C	F	Minterms	Maxterms
1	0	0	0	0		$M_0 = A + B + C$
2	0	0	1	0		$M_1 = A + B + \overline{C}$
3	0	1	0	0		$M_2 = A + \overline{B} + C$
4	0	1	1	1	$m_3 = \overline{A}.B.C$	
5	1	0	0	0		$M_4 = \overline{A} + B + C$
6	1	0	1	1	$m_5 = A.\overline{B}.C$	
7	1	1	0	1	$m_6 = A.B.\overline{C}$	
8	1	1	1	1	$m_7 = A.B.C$	

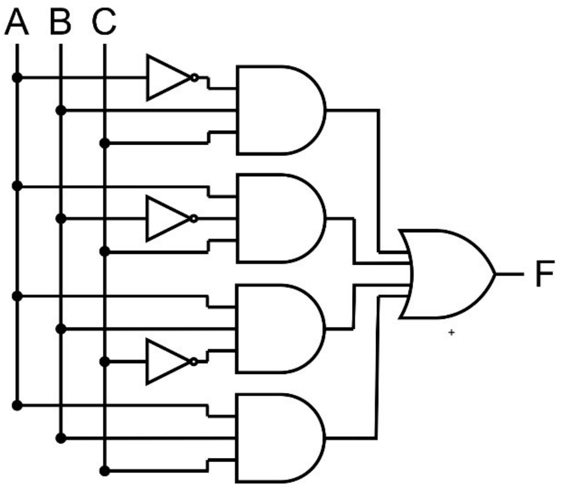
For Minterms

$$F_{ssop} = m_3 \text{ OR } m_5 \text{ OR } m_6 \text{ OR } m_7$$
$$= m_3 + m_5 + m_6 + m_7$$
$$= \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

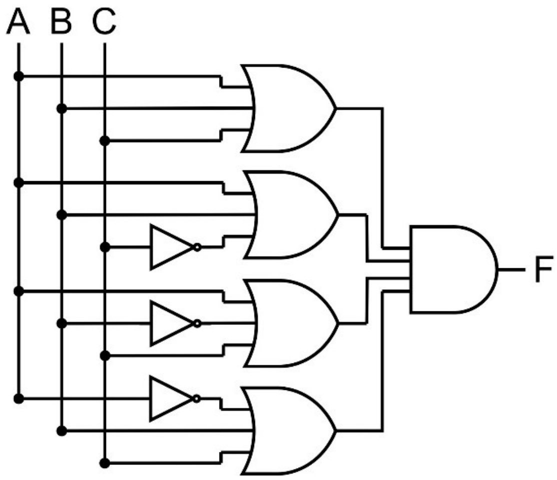
For Maxterms

$$F_{spos} = M_0 \text{ AND } M_1 \text{ AND } M_2 \text{ AND } M_4$$
$$= M_0 \,.\, M_1 \,.\, M_2 \,.\, M_4$$
$$= (A + B + C) \,.\, (A + B + \overline{C}) \,.\, (A + \overline{B} + C) \,.\, (\overline{A} + B + C)$$

Logic circuit based on SOP
SOP මත පදනම් වූ තාර්කික පරිපථය



Logic circuit based on POS
POS මත පදනම් වූ තාර්කික පරිපථය



Converting SOP to POS
SOP සිට POS බවට පරිවර්ථනය කිරීම

The logic behind converting SOP to POS is that minterms (SOP) and maxterms (POS) are complements of each other, so we use De Morgan's Theorem to switch between AND-OR (SOP) by taking the function's complement twice, ensuring the logic remains unchanged.
SOP, POS බවට පරිවර්තනය කිරීම පිටුපස ඇති තර්කය වන්නේ ගුණිත පද (POS) සහ එකතු පද (SOP) එකිනෙකට අනුපූරක වීමයි, එබැවින් අප විසින් ශ්‍රිතයේ තර්කනය නොවෙනස් වන බව සහතික කරමින් AND-OR (SOP) අතර මාරු වීමට, ශ්‍රිතයේ අනුපූරකය දෙවරක් ලබා ගැනීමෙන්, ද මෝගන්ගේ ප්‍රමේයය භාවිතා කරනු ලබයි.

These conversions are important because they help design circuits in different forms, making them easier to simplify, optimize, and implement using different types of logic gates.
මෙම පරිවර්තන වැදගත් වන්නේ ඒවා මගින් පරිපථ සරල කිරීමට, ප්‍රශස්ත කිරීමට, සහ විවිද වර්ගයේ තාර්කික ද්වාර භාවිතයෙන් ඒවා ක්‍රියාත්මක කිරීමට ඉඩ සලසමින් විවිධ ආකාරයන්ගෙන් පරිපථ නිර්මාණයට ඉඩ සලසන බැවිනි.

Steps to convert SOP into POS
SOP, POS බවට පරිවර්තනය කිරීමේ පියවර

- Find the complement ($\overline{F}$) of the given function.
  දී ඇති ශ්‍රිතයේ අනුපූරකය ($\overline{F}$) සොයාගන්න.
  Apply De Morgan's Theorem: Convert AND operations (•) to OR (+)
  ද මෝගන්ගේ ප්‍රමේයය යොදන්න: AND මෙහෙයුම් (•) OR (+) බවට පරිවර්තනය කරන්න.

- Simplify the obtained expression
  ලබා ගත් ප්‍රකාශනය සුළු කරන්න.

- Take the complement of $\overline{F}$ again ($\overline{\overline{F}} = F$) to get the original function in POS form.
  POS ආකාරයෙන් මුල් ශ්‍රිතය ලබා ගැනීමට $\overline{F}$ හි අනුපූරකය ($\overline{\overline{F}} = F$) නැවත නැවත ලබා ගන්න.

Converting POS to SOP
POS සිට SOP බවට පරිවර්ථනය කිරීම

To transform a POS expression into SOP, we apply the distributive law to expand the product of sums into a sum of products.
POS ප්‍රකාශනයක් SOP බවට පරිවර්තනය කිරීම සඳහා, එකතු වල ගුණිත පද ගුණිත වල එකතු පද බවට ප්‍රසාරණය කිරීමට විඝටන න්‍යාය යොදනු ලැබේ.

If necessary, Boolean simplifications can be applied to simplify the resulting expression.
අවශ්‍ය නම්, ප්‍රතිඵලයක් ලෙස ලැබෙන ප්‍රකාශනය සරල කිරීම සඳහා බූලියානු සරල කිරීම් යෙදිය හැකිය.

Steps to convert POS into SOP
POS, SOP බවට පරිවර්තනය කිරීමේ පියවර

- Apply the Distributive Law
  විඝටන න්‍යාය යොදන්න

  Apply this expansion to all sum terms in the given POS expression.
  දී ඇති POS ප්‍රකාශනයේ සියලුම එකතු පද සඳහා මෙම ප්‍රසාරණය යොදන්න.

- Expand Until No Parentheses Remain
  වරහන් ඉතිරි නොවන තෙක් ප්‍රසාරණය කරන්න

- Simplify If Necessary
  අවශ්‍ය නම් සරල කරන්න

  Apply Boolean simplifications such as Idempotent Law, Absorption Law, and Complement Law if needed.
  අවශ්‍ය නම් තදේවභාවී න්‍යාය, සංඝටන න්‍යාය සහ ප්‍රතිලෝම න්‍යාය වැනි බූලියානු සරල කිරීම් යොදන්න.

SOP to POS conversions

Example 1.

1. **Find the complement ($\overline{F}$) of the given function.**
   දී ඇති ශ්‍රිතයේ අනුපූරකය ($\overline{F}$) සොයාගන්න.

$Fsop = A.B + A.C + A.B.C$
$\overline{F} = \overline{A.B + A.C + A.B.C}$

2. **Simplify the obtained expression**
   ලබා ගත් ප්‍රකාශනය සුළු කරන්න.

$= \overline{A.B} \,.\, \overline{A.C} \,.\, \overline{A.B.C}$	De Morgan's law
$= (\overline{A} + \overline{B}) \,.\, (\overline{A} + \overline{C}) \,.\, (\overline{A} + \overline{B} + \overline{C})$	De Morgan's law
$= (\overline{A} + \overline{B}.\overline{C}) \,.\, (\overline{A} + \overline{B} + \overline{C})$	Redundancy law
$= \overline{A}.\overline{A} + \overline{A}.\overline{B} + \overline{A}.\overline{C} + \overline{B}.\overline{C}.\overline{A} + \overline{B}.\overline{B}.\overline{C} + \overline{C}.\overline{B}.\overline{C}$	Distributive law
$= \overline{A} + \overline{A}.\overline{B} + \overline{A}.\overline{C} + \overline{B}.\overline{C}.\overline{A} + \overline{B}.\overline{C} + \overline{B}.\overline{C}$	Idempotent law
$= \overline{A} + \overline{B}.\overline{C} + \overline{B}.\overline{C}$	Redundancy law
$= \overline{A} + \overline{B}.\overline{C}$	Idempotent law

3. **Take the complement of $\overline{F}$ again ($\overline{\overline{F}} = F$) to get the original function in POS form.**
   POS ආකාරයෙන් මුල් ශ්‍රිතය ලබා ගැනීමට $\overline{F}$ හි අනුපූරකය ($\overline{\overline{F}} = F$) නැවත නැවත ලබා ගන්න.

$\overline{F} = \overline{A} + \overline{B}.\overline{C}$	
$\overline{\overline{F}} = \overline{\overline{A} + \overline{B}.\overline{C}}$	Double complement
$F = \overline{\overline{A}}.\overline{\overline{B}.\overline{C}}$	De Morgan's
$F = A.(\overline{\overline{B}} + \overline{\overline{C}})$	De Morgan's
$Fpos = A.(B + C)$	De Morgan's

Therefore, $[A.B + A.C + A.B.C]_{sop} = [A.(B + C)]_{pos}$
එබැවින් $[A.B + A.C + A.B.C]_{sop} = [A.(B + C)]_{pos}$ වේ.

Example 2.

1. **Find the complement ($\overline{F}$) of the given function.**
   දී ඇති ශ්‍රිතයේ අනුපූරකය ($\overline{F}$) සොයාගන්න.

   $\mathrm{Fsop} = \overline{X}.Y.\overline{Z} + \overline{X}.\overline{Y} + Y.\overline{Z} + X.\overline{Y}.Z$

   $\overline{F} = \overline{\overline{X}.Y.\overline{Z} + \overline{X}.\overline{Y} + Y.\overline{Z} + X.\overline{Y}.Z}$

2. **Simplify the obtained expression**
   ලබා ගත් ප්‍රකාශනය සුළු කරන්න.

$= \overline{\overline{X}.Y.\overline{Z}}\,.\,\overline{\overline{X}.\overline{Y}}.\overline{Y.\overline{Z}}.\overline{X.\overline{Y}.Z}$	De Morgan's law
$= \left(\overline{\overline{X}} + \overline{Y} + \overline{\overline{Z}}\right).\left(\overline{\overline{X}} + \overline{\overline{Y}}\right).\left(\overline{Y} + \overline{\overline{Z}}\right).\left(\overline{X} + \overline{\overline{Y}} + \overline{Z}\right)$	De Morgan's law
$= \left(X + \overline{Y} + Z\right).(X + Y).\left(\overline{Y} + \overline{Z}\right).(\overline{X} + Y + \overline{Z})$	Double inverse law
$= \left(X.X + X.Y + \overline{Y}.X + \overline{Y}.Y + Z.X + Z.Y\right).\left(\overline{Y} + \overline{Z}\right).(\overline{X} + Y + \overline{Z})$	Distributive law
$= (X + Z.Y).\left(\overline{Y} + \overline{Z}\right).(\overline{X} + Y + \overline{Z})$	Redundancy law
$= (X + Z.Y).\left(\overline{Y}.\overline{X} + \overline{Y}.Y + Y.\overline{Z} + \overline{Z}.\overline{X} + \overline{Z}.Y + \overline{Z}.\overline{Z}\right)$	Redundancy law
$= (X + Z.Y).\left(\overline{Y}.\overline{X} + \overline{Z}\right)$	Redundancy law
$= (X + Z.Y).\left(\overline{Z} + \overline{Y}.\overline{X}\right)$	Associative law
$= \left(X.\overline{Z} + X.Y.\overline{X} + Z.Y.\overline{Z} + Z.Y.\overline{Y}.\overline{X}\right)$	Distributive law
$= \left(X.\overline{Z} + 0 + 0 + 0\right)$	Inverse law
$= \left(X.\overline{Z}\right)$	Identity law
$\overline{F} = \left(X.\overline{Z}\right)$	Identity law

3. **Take the complement of $\overline{F}$ again ($\overline{\overline{F}}$ =F) to get the original function in POS form.**
   POS ආකාරයෙන් මුල් ශ්‍රිතය ලබා ගැනීමට $\overline{F}$ හි අනුපූරකය ($\overline{\overline{F}}$ =F) නැවත නැවත ලබා ගන්න.

   $\overline{\overline{F}} = \overline{\left(X.\overline{Z}\right)}$
   $\mathrm{Fpos} = (\overline{X} + Z)$

   $[\overline{\mathbf{X}}.\mathbf{Y}.\overline{\mathbf{Z}} + \overline{\mathbf{X}}.\overline{\mathbf{Y}} + \mathbf{Y}.\overline{\mathbf{Z}} + \mathbf{X}.\overline{\mathbf{Y}}.\mathbf{Z}]_{sop}$ **=** $[(\overline{\mathbf{X}} + \mathbf{Z})]_{pos}$

Example 3.

1. **Find the complement ($\overline{F}$) of the given function.**
   දී ඇති ශ්‍රිතයේ අනුපූරකය ($\overline{F}$) සොයාගන්න.

   $Fsop = X.Y + \overline{X}.Z + Y.Z$

   $\overline{F} = \overline{X.Y + \overline{X}.Z + Y.Z}$

2. **Simplify the obtained expression**
   ලබා ගත් ප්‍රකාශනය සුළු කරන්න.

$= (\overline{X.Y}) . \left(\overline{\overline{X}.Z}\right) . (\overline{Y.Z})$	De Morgan's law
$= (\overline{X} + \overline{Y}) . (\overline{\overline{X}} + \overline{Z}) . (\overline{Y} + \overline{Z})$	De Morgan's law
$= (\overline{X} + \overline{Y}) . (X + \overline{Z}) . (\overline{Y} + \overline{Z})$	Double Inverse law
$= (\overline{X}.X + \overline{X}.\overline{Z} + \overline{Y}.X + Y.Z) . (\overline{Y} + \overline{Z})$	Distributive law
$= (0 + \overline{X}.\overline{Z} + \overline{Y}.X + Y.Z) . (\overline{Y} + \overline{Z})$	Inverse law
$= (\overline{X}.\overline{Z}.\overline{Y} + \overline{Y}.X.\overline{Y} + Y.Z.\overline{Y} + \overline{X}.\overline{Z}.\overline{Z} + \overline{Y}.X.\overline{Z} + Y.Z.\overline{Z})$	Distributive law
$= (\overline{X}.\overline{Z}.\overline{Y} + \overline{Y}.X.\overline{Y} + \overline{X}.\overline{Z}.\overline{Z} + \overline{Y}.X.\overline{Z})$	Inverse law
$= \overline{X}.\overline{Z}.\overline{Y} + \overline{Y}.X + \overline{X}.\overline{Z} + \overline{Y}.X.\overline{Z}$	Idempotent law

3. **Take the complement of $\overline{F}$ again ($\overline{\overline{F}} = F$) to get the original function in POS form.**
   POS ආකාරයෙන් මුල් ශ්‍රිතය ලබා ගැනීමට $\overline{F}$ හි අනුපූරකය ($\overline{\overline{F}} = F$) නැවත නැවත ලබා ගන්න.

$\overline{\overline{F}} = \overline{(\overline{X}.\overline{Z}.\overline{Y} + \overline{Y}.X + \overline{X}.\overline{Z} + \overline{Y}.X.\overline{Z})}$	
$Fpos = (\overline{\overline{X}.\overline{Z}.\overline{Y}}) . (\overline{\overline{Y}.X}) . (\overline{\overline{X}.\overline{Z}}) . (\overline{\overline{Y}.X.\overline{Z}})$	De Morgan's law
$= (\overline{\overline{X}} + \overline{\overline{Z}} + \overline{\overline{Y}}) . (\overline{\overline{Y}} + \overline{X}) . (\overline{\overline{X}} + \overline{\overline{Z}}) . (\overline{\overline{Y}} + \overline{X} + \overline{\overline{Z}})$	De Morgan's law
$= (X + Z + Y) . (Y + \overline{X}) . (X + Z) . (Y + \overline{X} + Z)$	Double Inverse law

$[\mathbf{X.Y + \overline{X}.Z + Y.Z}]_{sop} = [\mathbf{(X + Z + Y).(Y + \overline{X}).(X + Z).(Y + \overline{X} + Z)}]_{pos}$

ictfromabc.com

POS to SOP conversions

Example 1.

1. **Apply the Distributive Law**
   විඝටන න්‍යාය යොදන්න

   $F_{POS} = (X + Z)(Y + Z)$
   $= (X.Y + X.Z + Z.Y + Z.Z)$

2. **Apply Boolean simplifications.**
   බූලියන් සරල කිරීම් යොදන්න.

   $= (X.Y + Z.Y + X.Z + Z.Z)$
   $= (X.Y + Z.Y + X.Z + Z)$ Idempotent Law
   $= X.Y + Z$ Redundancy Law

   $[(\mathbf{X} + \mathbf{Z})(\mathbf{Y} + \mathbf{Z})]_{\mathbf{pos}} = [\mathbf{Z} + \mathbf{X}.\mathbf{Y}]_{\mathbf{sop}}$

Example 2.

$F_{POS} = (X + Y + Z)(\overline{X} + Y)$
$= (X.\overline{X} + Y.\overline{X} + Z.\overline{X} + X.Y + Y.Y + Z.Y)$

$= (X.\overline{X} + Y.\overline{X} + Z.\overline{X} + X.Y + Y.Y + Z.Y)$
$= (X.\overline{X} + Y.\overline{X} + Z.\overline{X} + X.Y + Y + Z.Y)$ Idempotent Law
$= (0 + Y.\overline{X} + Z.\overline{X} + X.Y + Y + Z.Y)$ Inverse Law
$= (Y.\overline{X} + Z.\overline{X} + X.Y + Y)$ Redundancy Law
$= (Y.\overline{X} + Z.\overline{X} + Y)$ Redundancy Law
$= (Z.\overline{X} + Y)$ Redundancy Law
$= Y + \overline{X}.Z$ Commutative Law

$[(\mathbf{X} + \mathbf{Y} + \mathbf{Z})(\overline{\mathbf{X}} + \mathbf{Y})]_{\mathbf{pos}} = [\mathbf{Y} + \overline{\mathbf{X}}.\mathbf{Z}]_{\mathbf{sop}}$

Example 3.

$F_{POS} = (X + Y)(\overline{X} + Z)(Y + Z)$
$= (X.\overline{X} + \overline{X}.Y + Z.X + Y.Z)(Y + Z)$
$= (X.\overline{X}.Y + \overline{X}.Y.Y + Z.X.Y + Y.Z.Y + X.\overline{X}.Z + \overline{X}.Y.Z + Z.X.Z + Y.Z.Z)$

$= (X.\overline{X}.Y + \overline{X}.Y.Y + Z.X.Y + Y.Z.Y + X.\overline{X}.Z + \overline{X}.Y.Z + Z.X.Z + Y.Z.Z)$
$= (X.\overline{X}.Y + \overline{X}.Y + Z.X.Y + Y.Z + X.\overline{X}.Z + \overline{X}.Y.Z + Z.X + Y.Z)$ Idempotent Law
$= (0.Y + \overline{X}.Y + Z.X.Y + Y.Z + 0.Z + \overline{X}.Y.Z + Z.X + Y.Z)$ Inverse Law
$= (\overline{X}.Y + Z.X.Y + Y.Z + \overline{X}.Y.Z + Z.X)$ Idempotent Law
$= \overline{X}.Y + Y.Z + X.Z$ Redundancy Law

$[(\mathbf{X} + \mathbf{Y}).(\overline{\mathbf{X}} + \mathbf{Z}).(\mathbf{Y} + \mathbf{Z})]_{\mathbf{pos}} = [\overline{\mathbf{X}}.\mathbf{Y} + \mathbf{Y}.\mathbf{Z} + \mathbf{X}.\mathbf{Z}]_{\mathbf{sop}}$

ictfromabc.com

Standard and non-standard expressions
සම්මත සහ සම්මත නොවන ප්‍රකාශන

Standard Expression
සම්මත ප්‍රකාශන

A Boolean expression where each term contains all variables of the function, either in complemented or uncomplemented form.
සෑම පදයකම ශ්‍රිතයේ ඇති සියලුම විචල්‍යයන් අනුපූරක හෝ අනුපූරක නොවන ආකාරයෙන් අඩංගු වන බූලීයානු ප්‍රකාශනයකි.

Ex. $F(A, B, C) = (A.B.C) + (A.B.C) + (A.B.C)$

Non-Standard Expression
සම්මත නොවන ප්‍රකාශනය

A Boolean expression where not all terms contain all variables.
සියලුම පදවල සියලුම විචල්‍යයන් අඩංගු නොවන බූලීයානු ප්‍රකාශනයකි.

Ex. $F(A, B, C) = A + B.C$

How to convert a non-standard expression into a standard expression
සම්මත නොවන ප්‍රකාශනයක් සම්මත ප්‍රකාශනයක් බවට පරිවර්තනය කරන ආකාරය?

To convert a non-standard SOP or POS expression into a standard SOP or POS expression, you need to ensure that each product term (minterm) in SOP and each sum term (maxterm) in POS includes all the variables in the function.
සම්මත නොවන SOP හෝ POS ප්‍රකාශනයක් සම්මත SOP හෝ POS ප්‍රකාශනයක් බවට පරිවර්තනය කිරීම සඳහා, SOP හි නම් සෑම ගුණිත පදයකම (minterm) ද, POS හි නම් සෑම එකතුව පදයකම (maxterm) ද ශ්‍රිතයේ සියලුම විචල්‍යයන් ඇතුළත් වන බවට සහතික කළ යුතුය.

This often involves expanding the expression by using Boolean algebraic properties.
මෙයට බොහෝ විට බූලියානු වීජ ගණිත නීති භාවිතයෙන් ප්‍රකාශනය පුළුල් කිරීම ඇතුළත් වේ.

1. Conversion of a non-standard SOP expression into a standard SOP expression
සම්මත නොවන SOP ප්‍රකාශනයක් සම්මත SOP ප්‍රකාශනයක් බවට පරිවර්ථනය

EX. 1
$\mathbf{F_{SOP} = A.\bar{B} + A.\bar{B}.C + A.B.\bar{C}.D}$

$= A.\bar{B} + A.\bar{B}.C.1 + A.B.\bar{C}.D$	Identity law
$= A.\bar{B} + A.\bar{B}.C.(D + \bar{D}) + A.B.\bar{C}.D$	Inverse law
$= A.\bar{B} + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Distributive law
$= A.\bar{B}.1.1 + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Identity law
$= A.\bar{B}.(C + \bar{C}).(D + \bar{D}) + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Inverse law
$= A.\bar{B}.[(C + \bar{C}).D + (C + \bar{C}).\bar{D}] + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Distributive law
$= A.\bar{B}.[C.D + \bar{C}.D + C.\bar{D} + \bar{C}.\bar{D}] + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Distributive law
$= A.\bar{B}.C.D + A.\bar{B}.\bar{C}.D + A.\bar{B}.C.\bar{D} + A.\bar{B}.\bar{C}.\bar{D} + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D$	Distributive law

$\mathbf{F_{SSOP} = A.\bar{B}.C.D + A.\bar{B}.\bar{C}.D + A.\bar{B}.C.\bar{D} + A.\bar{B}.\bar{C}.\bar{D} + A.\bar{B}.C.D + A.\bar{B}.C.\bar{D} + A.B.\bar{C}.D}$

ictfromabc.com

EX. 2

$\mathbf{F_{SOP} = A.B + A.\bar{C} + B.C}$

$= A.B.1 + A.\bar{C} + B.C$	Identity law
$= A.B.(C + \bar{C}) + A.\bar{C} + B.C$	Inverse law
$= A.B.C + A.B.\bar{C} + A.\bar{C} + B.C$	Distributive law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.1 + B.C$	Identity law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.(B + \bar{B}) + B.C$	Inverse law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.B + A.\bar{C}.\bar{B} + B.C$	Distributive law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.B + A.\bar{C}.\bar{B} + B.C.1$	Identity law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.B + A.\bar{C}.\bar{B} + B.C.(A + \bar{A})$	Inverse law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.B + A.\bar{C}.\bar{B} + B.C.A + B.C.\bar{A}$	Distributive law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.B + A.\bar{C}.\bar{B} + B.C.\bar{A}$	Idempotent law
$= A.B.C + A.B.\bar{C} + A.\bar{C}.\bar{B} + B.C.\bar{A}$	Idempotent law
$= A.B.C + A.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.B.C$	Commutative law

$\mathbf{F_{SSOP} = A.B.C + A.B.\bar{C} + A.\bar{B}.\bar{C} + \bar{A}.B.C}$

EX. 3

$\mathbf{F_{SOP} = X + X.Z + Y.Z}$

$= X.1.1 + X.1.Z + 1.Y.Z$	Identity law
$= X.(Y + \bar{Y}).(Z + \bar{Z}) + X.(Y + \bar{Y}).Z + (X + \bar{X}).Y.Z$	Inverse law
$= X.Y.Z + X.\bar{Y}.\bar{Z} + X.Y.Z + X.\bar{Y}.Z + X.Y.Z + \bar{X}.Y.Z$	Distributive law
$= X.Y.Z + X.\bar{Y}.\bar{Z} + X.\bar{Y}.Z + \bar{X}.Y.Z$	Idempotent law

$\mathbf{F_{SSOP} = X.Y.Z + X.\bar{Y}.\bar{Z} + X.\bar{Y}.Z + \bar{X}.Y.Z}$

2. Conversion of a non-standard POS statement into a standard POS statement

සම්මත නොවන POS ප්‍රකාශනයක් සම්මත POS ප්‍රකාශනයක් බවට පරිවර්ථනය

EX. 1

$\mathbf{F_{POS}} = (\mathbf{A} + \overline{\mathbf{B}}).(\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C}).(\mathbf{A} + \mathbf{B} + \overline{\mathbf{C}} + \mathbf{D})$

$= (A + \overline{B}).(A + \overline{B} + C + 0).(A + B + \overline{C} + D)$	Identity law
$= (A + \overline{B}).(A + \overline{B} + C + (D.\overline{D})).(A + B + \overline{C} + D)$	Inverse law
$= (A + \overline{B}).(A + \overline{B} + C + D).(A + \overline{B} + C + \overline{D}).(A + B + \overline{C} + D)$	Distributive law
$= (A + \overline{B} + 0 + 0).(A + \overline{B} + C + D).(A + \overline{B} + C + \overline{D}).(A + B + \overline{C} + D)$	Identity law
$= [A + \overline{B}(C.\overline{C}) + (D.\overline{D})].(A + \overline{B} + C + D).(A + \overline{B} + C + \overline{D}).(A + B + \overline{C} + D)$	Inverse law
$= \left[A + \overline{B}\left((C.\overline{C}) + D.(C.\overline{C}) + \overline{D}\right)\right].(A + \overline{B} + C + D).(A + \overline{B} + C + \overline{D}).$ $(A + B + \overline{C} + D)$	Distributive law
$= [A + \overline{B}(C + D).(\overline{C} + D).(C + \overline{D}).(\overline{C} + \overline{D})].(A + \overline{B} + C + D).$ $(A + \overline{B} + C + \overline{D}).(A + B + \overline{C} + D)$	Distributive law
$= (A + \overline{B} + C + D).(A + \overline{B} + \overline{C} + D).(A + \overline{B} + C + \overline{D}).(A + \overline{B} + \overline{C} + \overline{D}).$ $(A + \overline{B} + C + D).(A + \overline{B} + C + \overline{D}).(A + B + \overline{C} + D)$	Distributive law

$\mathbf{F_{SPOS}} = (\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C} + \mathbf{D}).(\mathbf{A} + \overline{\mathbf{B}} + \overline{\mathbf{C}} + \mathbf{D}).(\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C} + \overline{\mathbf{D}}).(\mathbf{A} + \overline{\mathbf{B}} + \overline{\mathbf{C}} + \overline{\mathbf{D}}).(\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C} + \mathbf{D}).$
$(\mathbf{A} + \overline{\mathbf{B}} + \mathbf{C} + \overline{\mathbf{D}}).(\mathbf{A} + \mathbf{B} + \overline{\mathbf{C}} + \mathbf{D})$

EX. 2

$\mathbf{F_{POS}} = (\overline{\mathbf{X}} + \mathbf{Z}).(\mathbf{Y} + \overline{\mathbf{Z}}).(\mathbf{X} + \overline{\mathbf{Y}} + \mathbf{Z})$

$= (\overline{X} + 0 + Z).(0 + Y + \overline{Z}).(X + \overline{Y} + Z)$	Identity law
$= (\overline{X} + (Y.\overline{Y}) + Z).((X.\overline{X}) + Y + \overline{Z}).(X + \overline{Y} + Z)$	Inverse law
$= (\overline{X} + Y + Z).(\overline{X} + \overline{Y} + Z).(X + Y + \overline{Z}).(\overline{X} + Y + Z).(X + \overline{Y} + Z)$	Distributive law
$= (\overline{X} + Y + Z).(\overline{X} + \overline{Y} + Z).(X + Y + \overline{Z}).(X + \overline{Y} + Z)$	Idempotent law

$\mathbf{F_{SSOP}} = (\overline{\mathbf{X}} + \mathbf{Y} + \mathbf{Z}).(\overline{\mathbf{X}} + \overline{\mathbf{Y}} + \mathbf{Z}).(\mathbf{X} + \mathbf{Y} + \overline{\mathbf{Z}}).(\mathbf{X} + \overline{\mathbf{Y}} + \mathbf{Z})$

Designs simple digital circuits using logic gates for a given scenario
තාර්කික ද්වාර භාවිතයෙන් ලබා දී ඇති අවස්ථාවක් සඳහා සරල ඩිජිටල් පරිපථ නිර්මාණය කරයි.

A company has a high-security safe that requires keys K1, K2 and K3. To unlock the safe, at least two of these keys must be used together. The presence of each key is represented as a binary variable, where 1 means the key is used and 0 means it is not used.
සමාගමකට K1, K2 සහ K3 යතුරු අවශ්‍ය වන අධි ආරක්ෂිත සේප්පුවක් තිබේ. සේප්පුව අගුළු ඇරීමට, අවම වශයෙන් මෙම යතුරු දෙකක්වත් එකට භාවිතා කළ යුතුය. එක් එක් යතුරේ පැවැත්ම ද්වීමය විචල්‍යයක් ලෙස නිරූපණය කෙරේ, එහිදී 1 යන්නෙන් අදහස් කරන්නේ යතුර භාවිතා කර ඇති බවත් 0 යන්නෙන් අදහස් කරන්නේ එය භාවිතා කර නොමැති බවත්ය.

The safe opens (S = 1) only when at least two keys are used. Otherwise, the safe remains locked (S = 0).
සේප්පුව විවෘත වන්නේ (S = 1) අවම වශයෙන් යතුරු දෙකක්වත් භාවිතා කළ විට පමණි. එසේ නොමැතිනම්, සේප්පුව අගුළු දමා පවතී (S = 0)

1. **Construct the truth table representing this situation.**
   මේ සඳහා සත්‍යතා වගුව සාදන්න.

Defining the inputs and outputs
ආදාන සහ ප්‍රතිදාන නිර්වචනය කිරීම

A key (K1, K2 or K3) is used යතුරක් (K1, K2 හෝ K3) භාවිතා කෙරේ	1
A key (K1, K2 or K3) is not used යතුරක් (K1, K2 හෝ or K3) භාවිතා නොකෙරේ	0
The safe (S) is open සේප්පුව (S) විවෘතයි	1
The safe (S) is closed සේප්පුව (S) වසා ඇත	0

The safe will only open if at least two keys are used. This means that if one key is used, the safe remains locked. If two or three keys are used, the safe opens.
සේප්පුව විවෘත වන්නේ අවම වශයෙන් යතුරු දෙකක්වත් භාවිතා කළහොත් පමණි. මෙයින් අදහස් කරන්නේ එක් යතුරක් භාවිතා කළහොත් සේප්පුව අගුළු වැටී පවතින බවයි. ඇති බවයි. යතුරු දෙකක් හෝ තුනක් භාවිතා කළහොත් සේප්පුව විවෘත වේ.

K1	K2	K3	S
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

ictfromabc.com

2. **Construct the SOP (Sum of Products) expression using minterms.**
   Minterms භාවිතයෙන් SOP එකතුන්ගේ ගුණිත ප්‍රකාශනය නිර්මාණය කරන්න.

   SOP is a simplified OR (sum) of multiple AND (product) terms. We take the rows where S=1 (safe opens) and write each as an AND expression.
   SOP යනු OR (sum) පද කිහිපයක සරල කළ AND (එකතුව) වේ. S=1 (ආරක්ෂිතව විවෘත වන) පේළි ගෙන ඒ සෑම එකක්ම AND ප්‍රකාශනයක් ලෙස ලිවිය යුතුය.

K1	K2	K3	S	Minterms
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{K1}.K2.K3$
1	0	0	0	
1	0	1	1	$K1.\overline{K2}.K3$
1	1	0	1	$K1.K2.\overline{K3}$
1	1	1	1	$K1.K2.K3$

   $S_{sop} = \overline{K1}.K2.K3 + K1.\overline{K2}.K3 + K1.K2.\overline{K3} + K1.K2.K3$

3. **Construct the POS (Product of Sums) expression using maxterms.**
   Maxterms භාවිතයෙන් POS ප්‍රකාශනය නිර්මාණය කරන්න.

   POS is a simplified AND (product) of multiple OR (sum) terms. We take the rows where S=0 (safe remains locked) and write each as an OR expression.
   POS යනු AND (එකතු) පද කිහිපයක සරල කළ OR (sum) වේ. S=0 (සේප්පුව ආරක්ෂිතව අගුළු දමා ඇති) පේළි ගෙන ඒ සෑම එකක්ම OR ප්‍රකාශනයක් ලෙස ලිවිය යුතුය.

K1	K2	K3	S	Maxterms
0	0	0	0	$K1 + K2 + K3$
0	0	1	0	$K1 + K2 + \overline{K3}$
0	1	0	0	$K1 + \overline{K2} + K3$
0	1	1	1	
1	0	0	0	$\overline{K1} + K2 + K3$
1	0	1	1	
1	1	0	1	
1	1	1	1	

   $S_{POS} = (K1 + K2 + K3).(K1 + K2 + \overline{K3}).(K1 + \overline{K2} + K3).(\overline{K1} + K2 + K3)$

ictfromabc.com

4. Simplify the SOP and POS expressions using Boolean laws.
බූලියානු නීති භාවිතයෙන් SOP සහ POS ප්‍රකාශන සරල කරන්න.

SOP

$= \overline{K1}.K2.K3 + K1.\overline{K2}.K3 + K1.K2.\overline{K3} + K1.K2.K3$

We can make the simplification easier and shorter by adding $K1.K2.K3$ for every term in the expression using the Idempotent law.
තදේවභාවී නියමය භාවිතයෙන් ප්‍රකාශනයේ සෑම පදයක් සඳහාම $K1.K2.K3$ එකතු කිරීමෙන් අපට සරල කිරීම පහසු සහ කෙටි කළ හැකිය.

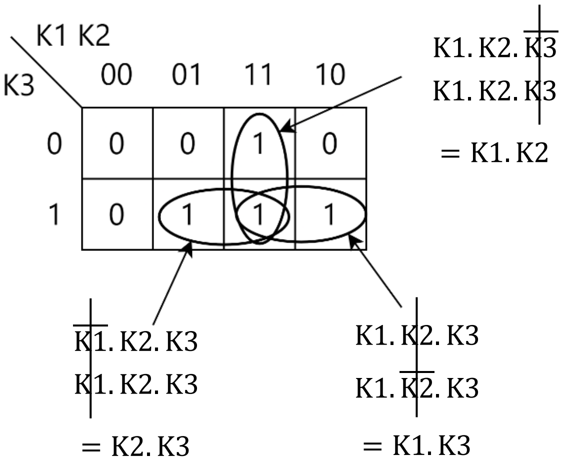
$= \overline{K1}.K2.K3 + K1.K2.K3 + K1.\overline{K2}.K3 + K1.K2.K3 + K1.K2.\overline{K3} + K1.K2.K3$
$= K2.K3.(\overline{K1} + K1) + K1.K3.(\overline{K2} + K2) + K1.K2.(\overline{K3} + K3)$ Distributive Law
$= K2.K3.(1) + K1.K3.(1) + K1.K2.(1)$ Inverse Law
$= K2.K3 + K1.K3 + K1.K2$ Identity Law
$\underline{S_{sop} = K1.K2 + K2.K3 + K1.K3}$ Commutative Law

POS

$= (K1 + K2 + K3).(K1 + K2 + \overline{K3}).(K1 + \overline{K2} + K3).(\overline{K1} + K2 + K3)$
$= (K1 + K2 + K3).(K1 + K2 + \overline{K3}).(K1 + K2 + K3)(K1 + \overline{K2} + K3).(K1 + K2 + K3).$
$(\overline{K1} + K2 + K3)$ Idempotent law
$= (K1 + K2 + (K3.\overline{K3})).(K1 + K3 + (K2.\overline{K2})).(K2 + K3 + (K1.\overline{K1}))$ Distributive law
$= (K1 + K2 + (0)).(K1 + K3 + (0)).(K2 + K3 + (0))$ Inverse law
$\underline{S_{pos} = (K1 + K2).(K1 + K3).(K2 + K3)}$ Identity law

5. Simplify the SOP and POS expressions using Karnaugh-maps.
කානෝ සිතියම් භාවිතයෙන් SOP සහ POS ප්‍රකාශන සරල කරන්න.

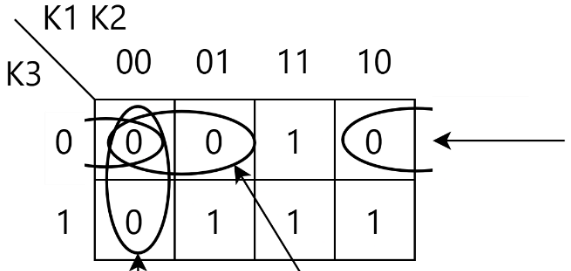
SOP



$\underline{Ssop = K1.K2 + K2.K3 + K1.K3}$

ictfromabc.com

POS



$\overline{K1} + K2 + K3$
$K1 + K2 + K3$
$= K2 + K3$

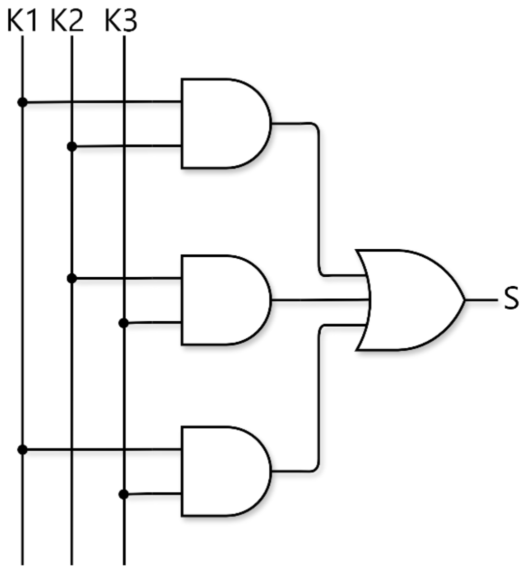
$K1 + K2 + K3$
$K1 + K2 + \overline{K3}$
$= K1 + K2$

$K1 + K2 + K3$
$K1 + \overline{K2} + K3$
$= K1 + K3$

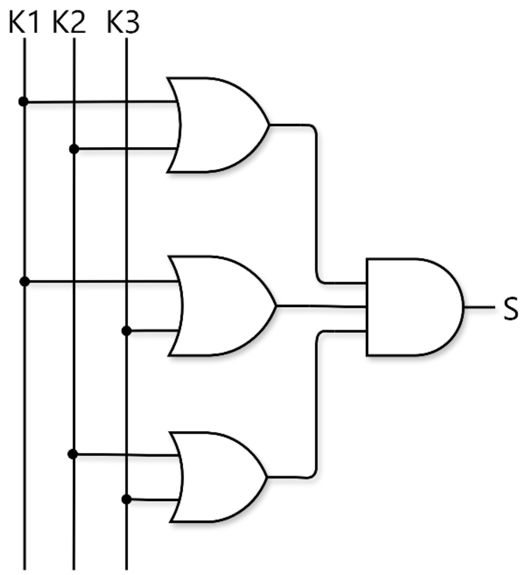
<u>$Spos = (K1 + K2).(K1 + K3).(K2 + K3)$</u>

6. **Design the circuit diagrams for the simplified SOP and POS expressions using basic logic gates.**
   මූලික තාර්කික ද්වාර භාවිතයෙන් සරල කළ **SOP** සහ **POS** ප්‍රකාශන සඳහා පරිපථ සටහන් අඳින්න.

S_{sop} $= \mathbf{K1.K2 + K2.K3 + K1.K3}$

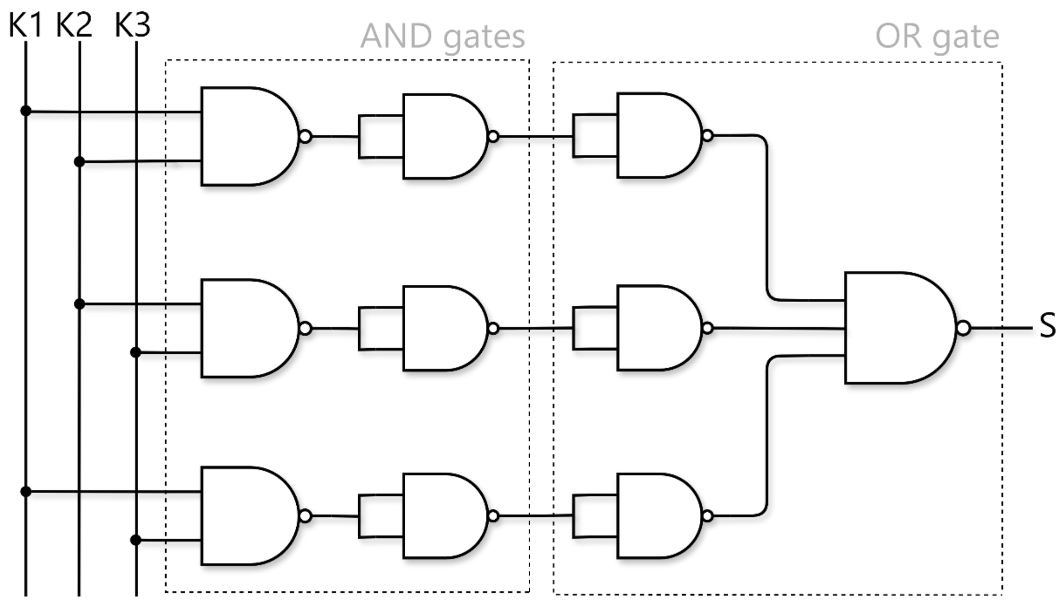


S_{pos} $= \mathbf{(K1 + K2).(K1 + K3).(K2 + K3)}$

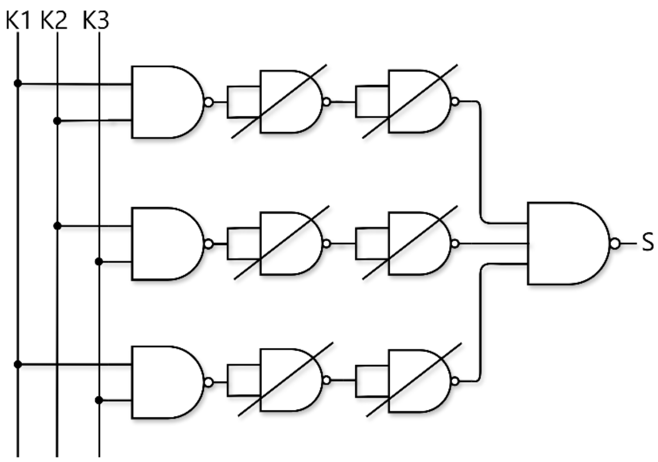


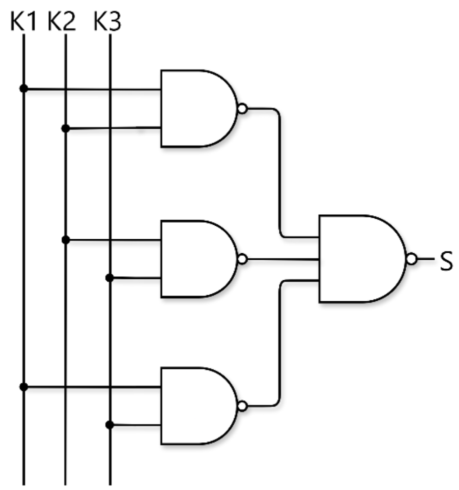
7. **Design the circuit diagrams for the simplified SOP and POS expressions using universal gates. (NAND and NOR)**
සාර්වත්‍ර ද්වාර භාවිතයෙන් සරල කළ SOP සහ POS ප්‍රකාශන සඳහා පරිපථ සටහන් අදින්න.

SOP - NAND



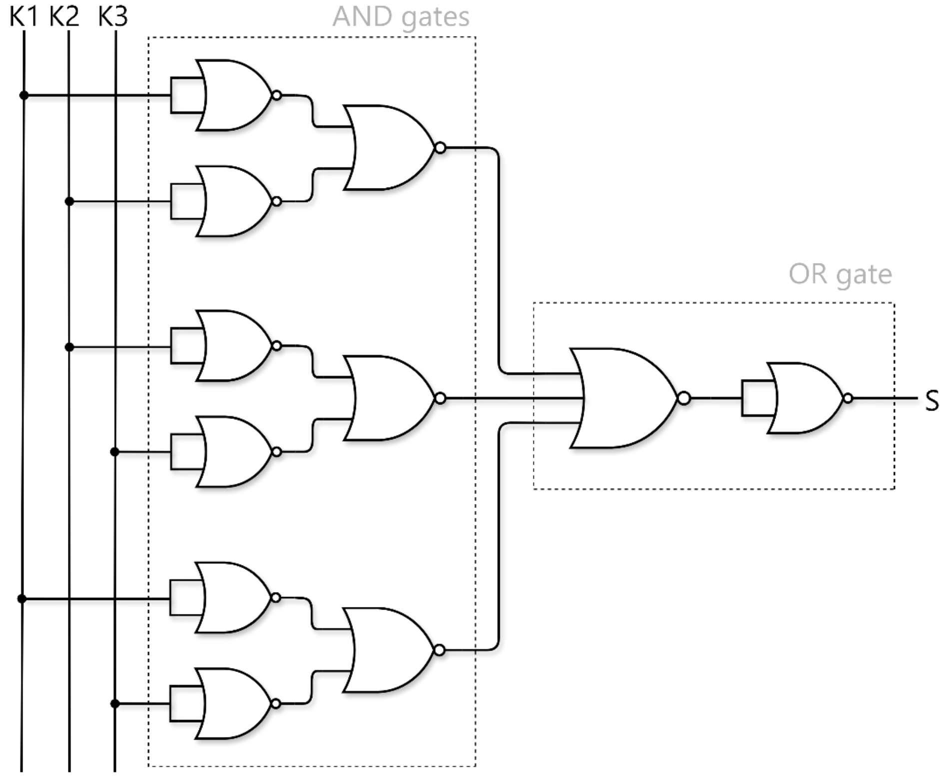
This circuit can be further simplified by removing the repeating NOT gates as follows
පුනරාවර්තනය වන NOT ද්වාර ඉවත් කිරීමෙන් මෙම පරිපථය පහත පරිදි තවදුරටත් සරල කළ හැක.



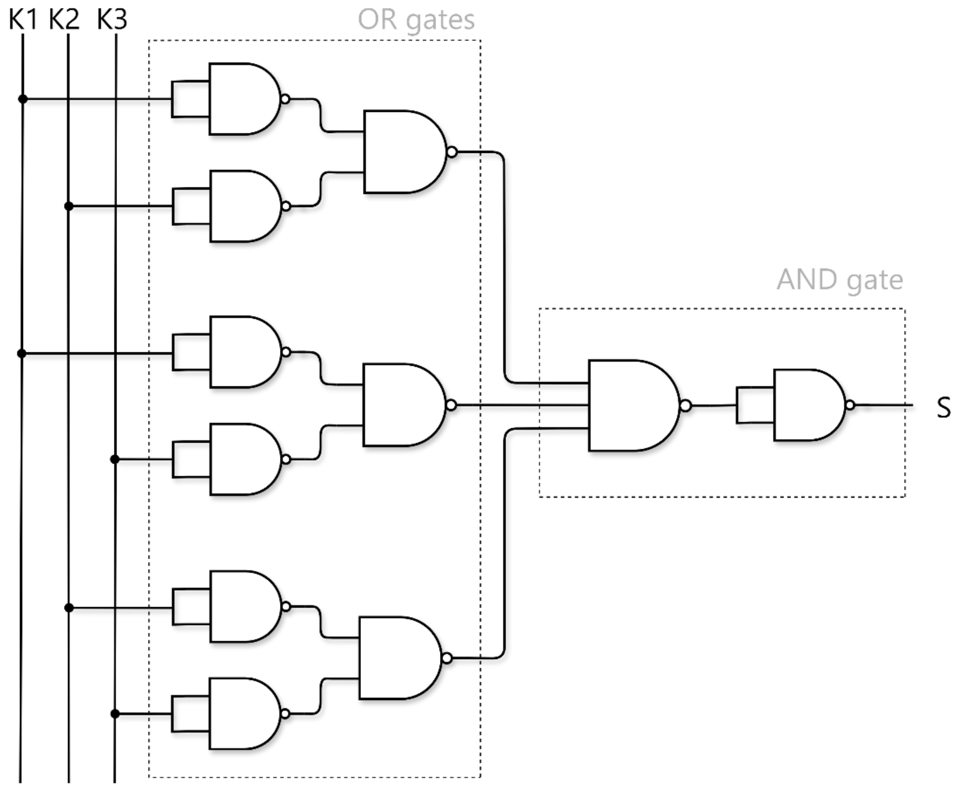


ictfromabc.com

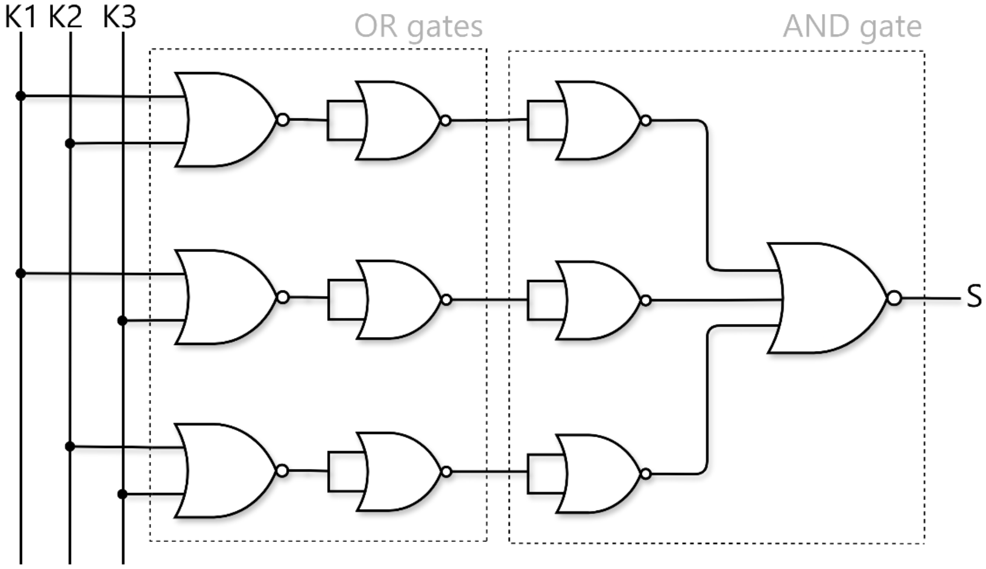
SOP – NOR



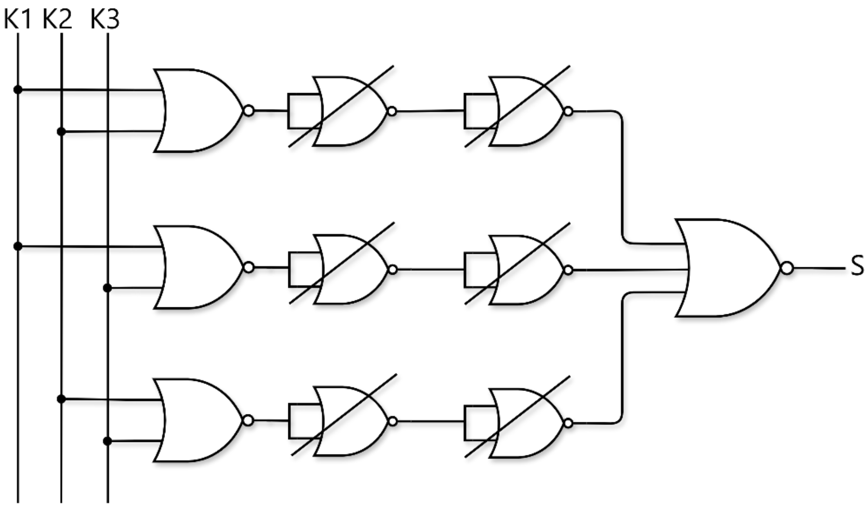
POS – NAND

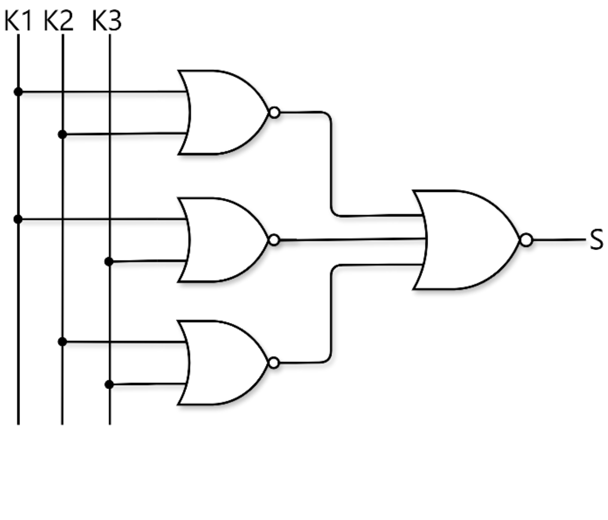


POS – NAND



This circuit can also be further simplified by removing the repeating NOT gates as follows
පුනරාවර්තනය වන NOT ද්වාර ඉවත් කිරීමෙන් මෙම පරිපථය ද පහත පරිදි තවදුරටත් සරල කළ හැක.





Building blocks of CPU
CPU හි තැනුම් ඒකක

Adders
ආකලක

An adder is a digital logic circuit that performs binary addition.
ආකලකයක් යනු ද්විමය එකතු කිරීම සිදු කරන අංකිත තාර්කික පරිපථයකි.

1. Half Adder
අර්ධ ආකලකය

Adds two single-bit binary numbers and produces two outputs called Sum (A ⊕ B) and Carry (A • B), without considering any carry input from a previous addition.
පෙර සිදු කරන ලද එකතු කිරීමකින් කිසිදු රැගෙන යාමේ ආදානයක් සැලකිල්ලට නොගෙන, තනි-බිටු සහිත ද්විමය සංඛ්‍යා දෙකක් එකතු කර Sum එක (A ⊕ B) සහ Carry එක (A • B) ලෙසින් හඳුන්වන ප්‍රතිදාන දෙකක් නිපදවයි.

Inputs
ආදාන

Two single-bit binary numbers, usually denoted as A and B.
සාමාන්‍යයෙන් A සහ B ලෙස දැක්වෙන තනි-බිටු ද්විමය සංඛ්‍යා දෙකක්.

Outputs
ප්‍රතිදාන

Sum (S) - This is the result of adding the two input bits.
මෙය ආදාන බිටු දෙක එකතු කිරීමේ ප්‍රතිඵලයයි.

Carry (C) - This indicates if there is a carry-out from the addition. A carry is produced when the sum of the two bits is greater than what can be represented by a single bit (ex:- greater than 1).
රැගෙන යාම (C) - එකතු කිරීමෙන් ගෙනයාමක් තිබේද යන්න මෙයින් පෙන්නුම් කරයි. බිටු දෙකේ එකතුව තනි බිටුවකින් (එනම්, 1 ට වඩා වැඩි) නියෝජනය කළ හැකි ප්‍රමාණයට වඩා වැඩි වූ විට රැගෙන යාමක් නිර්මාණය වේ.

Functionality
ක්‍රියාකාරිත්වය

The Sum (S) output is generated using an XOR gate, which performs the binary addition of A and B.
එකතු (S) ප්‍රතිදානය උත්පාදනය කරනු ලබන්නේ XOR ද්වාරයක් භාවිතයෙන් වන අතර එය A සහ B හි ද්විමය එකතු කිරීම සිදු කරයි.

The Carry (C) output is generated using an AND gate, which produces a 1 only when both A and B are 1, indicating a carry over.
රැගෙන යාම (C) ප්‍රතිදානය ජනනය කරනු ලබන්නේ AND ද්වාරයක් භාවිතයෙන් වන අතර, එමගින් රැගෙන යාමක් පෙන්නුම් කරමින් 1ක් නිපදවන්නේ A සහ B යන දෙකම 1 වන විටදීය

The truth table for a half adder is as follows
අර්ධ ආකලකය සඳහා සත්‍යතා වගුව පහත පරිදි වේ

A	B	Sum(S)	Carry(c)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic Equations
තාර්කික සමීකරණ

From the truth table, the logic equations for the sum and carry can be derived
සත්‍යතා වගුවෙන්, එකතුව සහ රැගෙන යාම සඳහා තාර්කික සමීකරණ ව්‍යුත්පන්න කළ හැක

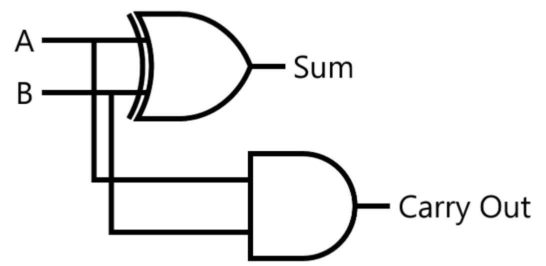
Sum (s) = A ⊕ B (A XOR B)
Carry (s) = A.B (A AND B)

Logic Circuit
තාර්කික පරිපථය

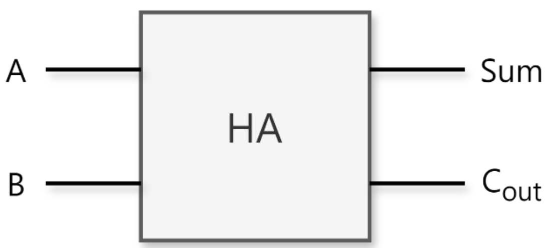
A half adder circuit can be implemented using basic logic gates:
මූලික තාර්කික ද්වාර භාවිතයෙන් අර්ධ ආකලක පරිපථයක් ක්‍රියාත්මක කළ හැකිය:

XOR Gate - Used to produce the Sum output.
XOR ද්වාරය - Sum ප්‍රතිදානය නිපදවීමට භාවිතා කරයි.

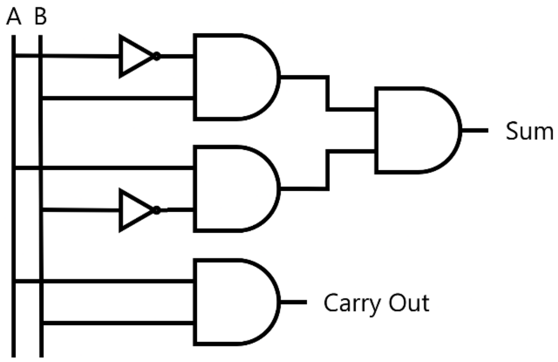
AND Gate - Used to produce the Carry output.
AND ද්වාරය - Carry ප්‍රතිදානය නිපදවීමට භාවිතා කරයි.



Block Diagram



Logic Circuit for Half Adder without XOR gate
XOR ද්වාරය රහිත අර්ධ ආකලකය සඳහා තාර්කික පරිපථය



Practical Applications
ප්‍රායෝගික යෙදුම්

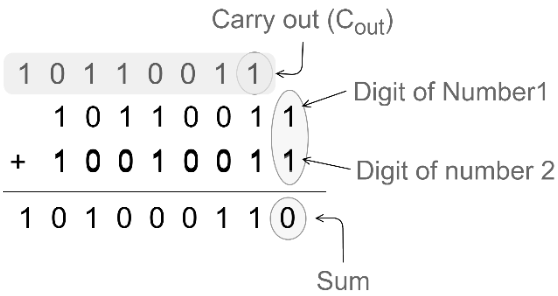
Half adders are the building blocks for more complex adders used in computers, such as full adders and multi-bit adders.
පූර්ණ ආකලක සහ multi-bit ආකලකය වැනි පරිගණකවල භාවිතා කරන වඩාත් සංකීර්ණ ආකලක සඳහා වන තැනුම් ඒකක අර්ධ ආකලක ලෙස සැලකිය හැකිය.

Used in various digital systems where binary addition is required.
ද්විමය එකතු කිරීම් අවශ්‍ය වන විවිධ අංකිත පද්ධතිවල භාවිතා වේ.

Understanding half adders is a fundamental step in learning how computers perform arithmetic operations and is crucial for anyone studying digital electronics or computer engineering.
අර්ධ ආකලකය තේරුම් ගැනීම පරිගණක ගණිතමය මෙහෙයුම් සිදු කරන ආකාරය ඉගෙන ගැනීමේ මූලික පියවරක් වන අතර අංකිත ඉලෙක්ට්‍රොනික විද්‍යාව හෝ පරිගණක ඉංජිනේරු විද්‍යාව හදාරන ඕනෑම කෙනෙකුට එය ඉතා වැදගත් වේ.

Full Adder
පූර්ණ ආකලකය

Adds two single-bit binary numbers along with a carry input (C_{in}) from a previous stage, producing a Sum and a Carry-out, making it suitable for multi-bit addition.
පෙර අදියරකින් ගෙන යන ආදානයක් (C_{in}) සමගින් තනි-බිටු ද්විමය සංඛ්‍යා දෙකක් එකතු කරන අතර, Sum එකක් සහ Carry-out එකක් නිපදවනු ලබයි. මෙය බිටු කිහිපයක් එකතු කිරීම සඳහා සුදුසු වේ.



Digit 1	Digit 2	Sum	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Inputs
ආදාන

A and B are the two binary digits (0 or 1) that are to be added.
A සහ B යනු එකතු කළ යුතු ද්විමය ඉලක්කම් දෙක (0 හෝ 1) වේ.

C_{in} is the carry bit from the previous stage of addition.
C_{in} යනු එකතු කිරීමේ පෙර අදියරේ සිට රැගෙන එනු ලබන බිටුවයි.

Outputs
ප්‍රතිදාන

The Sum (S) – The result of the addition
එකතු කිරීමෙන් ලැබෙන ප්‍රතිඵලය

The Carry (C_{out}) – The carried-out bit after the addition.
එකතු කිරීමෙන් පසු රැගෙන යන ලද බිටුව.

Functionality
ක්‍රියාකාරීත්වය

The Sum (S) output is generated using two XOR gates in series.
එකතු (S) ප්‍රතිදානය උත්පාදනය කරනු ලබන්නේ ආදාන තුනෙහි XOR ද්වාරයක් භාවිතා කරමිනි.

The Carry (C_{out}) output is generated using three AND gates and one OR gate.
රැගෙන යාම් (C_{out}) ප්‍රතිදානය ජනනය කරනු ලබන්නේ AND ද්වාර දෙකක් සහ එක් OR ද්වාරයක් භාවිතා කරමිනි.

The truth table for a full adder is as follows
පූර්ණ ආකලකය සඳහා සත්‍ය වගුව පහත පරිදි වේ

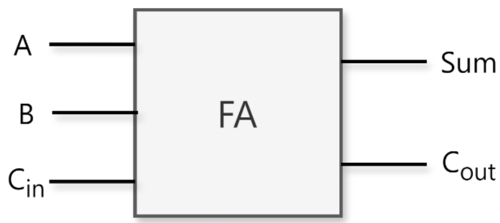
A	B	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Logic Equations
තාර්කික සමීකරණ

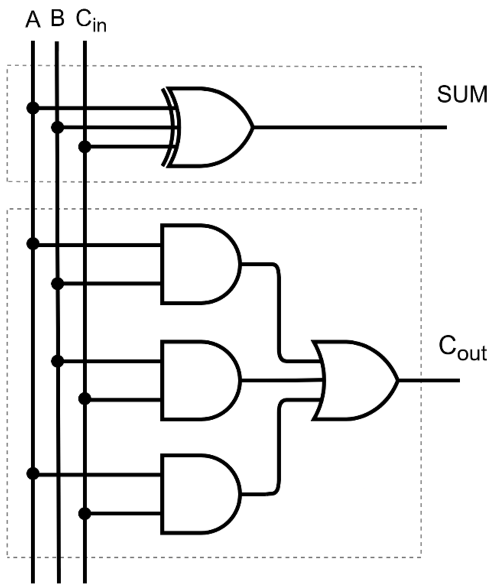
From the truth table, the logic equations for the sum and carry-out can be derived.
සත්‍යතා වගුවෙන්, එකතුව සහ රැගෙනයාම සඳහා තාර්කික සමීකරණ ව්‍යුත්පන්න කළ හැක.

Sum (S) = $A \oplus B \oplus C_{in}$
Carry out (C_{out}) = $(A.B) + (B.C_{in}) + (A.C_{in})$

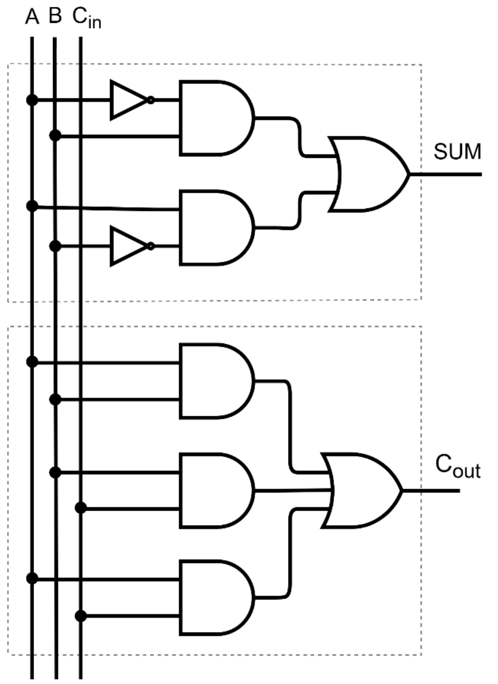
Block Diagram



Logic Circuit
තාර්කික පරිපථය



Logic Circuit for Full Adder without XOR gate
XOR ද්වාරය රහිත පූර්ණ ආකලකය සඳහා තාර්කික පරිපථය



Obtaining another equation for C_{out} of a full adder.
පූර්ණ අකලයක C_{out} සඳහා තවත් සමීකරණයක් ලබා ගැනීම.

First, let's get the truth table for C_{out}
පලමුවෙන් C_{out} සදහා සත්‍යතා වගුව ලබා ගනිමු

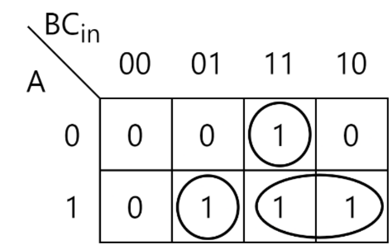
A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Let's put this output in a Kmap.
මෙම ප්‍රතිදානය කානො සිතියමක ඇතුළත් කරමු.

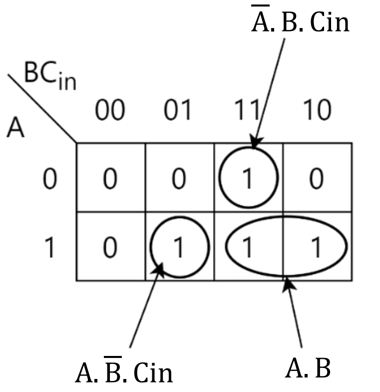
But here, it is done in a different way than the usual way.
නමුත් මෙහිදී එය සාමාන්‍ය ක්‍රමයට වඩා වෙනස් ආකාරයකට සිදු කෙරේ.

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

When grouping this Kmap, it should be done as follows.
මෙම කානො සිතියම කාණ්ඩ කිරීමේදී එය පහත ආකාරයට සිදු කළ යුතුය.



The next step is to obtain the simplified expression using the grouped Kmap.
මීළඟ පියවර වන්නේ කාණ්ඩගත කානො සිතියම භාවිතයෙන් සරල කළ ප්‍රකාශනය ලබා ගැනීමයි.



$C_{out} = A.B + A.\overline{B}.Cin + \overline{A}.B.Cin$

We can further simplify this as follows.
මෙය පහත පරිදි තවදුරටත් සරල කළ හැකිය.

$C_{out} = A.B + Cin.(A.\overline{B} + \overline{A}.B)$
$C_{out} = A.B + Cin.(A \oplus B)$

Implementing a full adder using half adders.
අර්ධ ආකලක භාවිතයෙන් පූර්ණ අකලයක් ක්‍රියාත්මක කිරීම.

The two outputs of a half adder are,
අර්ධ ආකලකයක ප්‍රතිදාන දෙක වන්නෙ,

Sum = $A \oplus B$
C_{out} = A.B

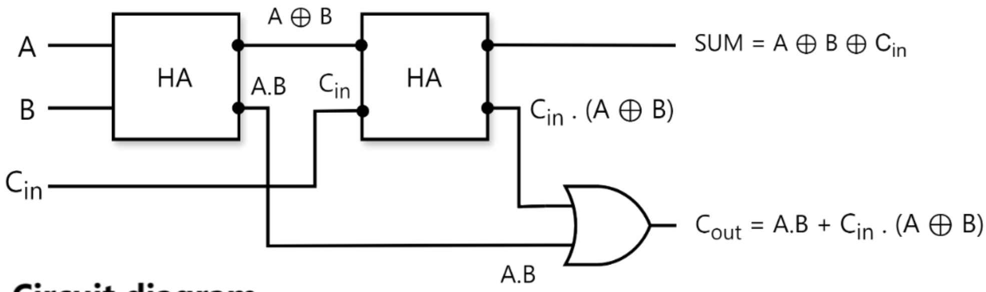
The two outputs of a full adder are,
පූර්ණ ආකලකය ප්‍රතිදාන දෙක වන්නෙ,

Sum = $A \oplus B \oplus Cin$
$C_{out} = A.B + Cin.(A \oplus B)$

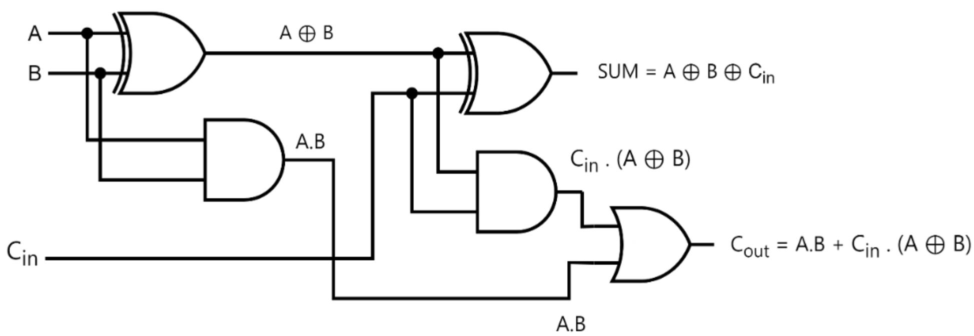
So, we need to design a circuit that outputs these two expressions using half adders.
එබැවින්, මෙම ප්‍රකාශන දෙක ප්‍රතිදානය කරන පරිපථයක් අර්ධ ආකලක භාවිතයෙන් නිර්මාණය කළ යුතුයි.

We need two half adders and an OR gate to implement a full adder.
අපට අර්ධ ආකලක දෙකක් සහ OR ද්වාරයක් පූර්ණ ආකලකයක් ක්‍රියාත්මක කිරීමට අවශ්‍ය වේ.

Block diagram



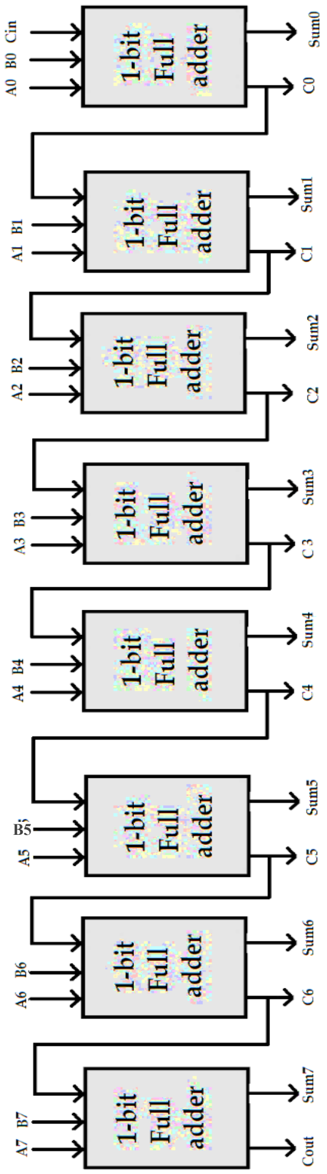
Circuit diagram



8-bit adder block diagram

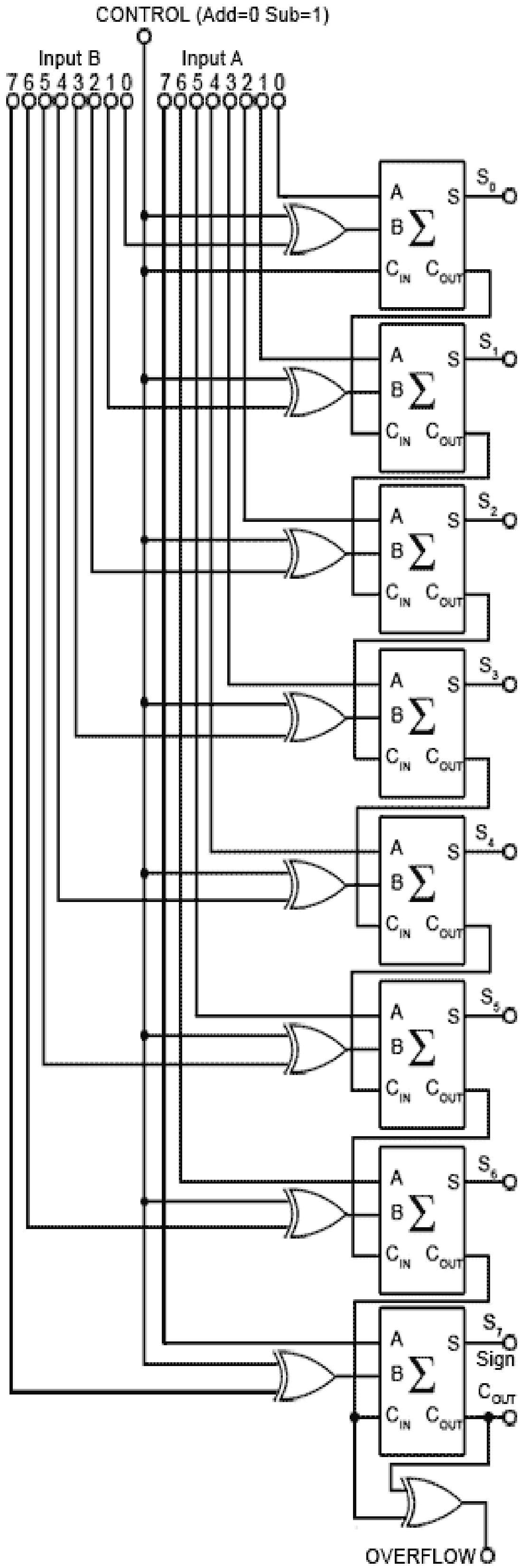
Each full adder takes two input bits (A and B) and a carry-in bit from the previous adder and produces a sum bit and a carry-out bit to the next adder.
සෑම පූර්ණ ආකලකයකටම පෙර එකතු කරන්නා වෙතින් ආදාන බිටු දෙකක් (A සහ B) සහ රැගෙන යාමේ බිටු එකක් ගෙන ඊළඟ ආකලකය වෙත එකතු කරන බිටු එකක් සහ රැගෙන යාමේ බිටු එකක් නිපදවයි.

The first adder in the series is a half adder because it does not have a carry-in.
ශ්‍රේණියේ පළමු ආකලකය එය රැගෙන යාමක් නොමැති නිසා අර්ධ ආකලකය වේ



8-bit adder block diagram

8-bit adder circuit diagram



ictfromabc.com

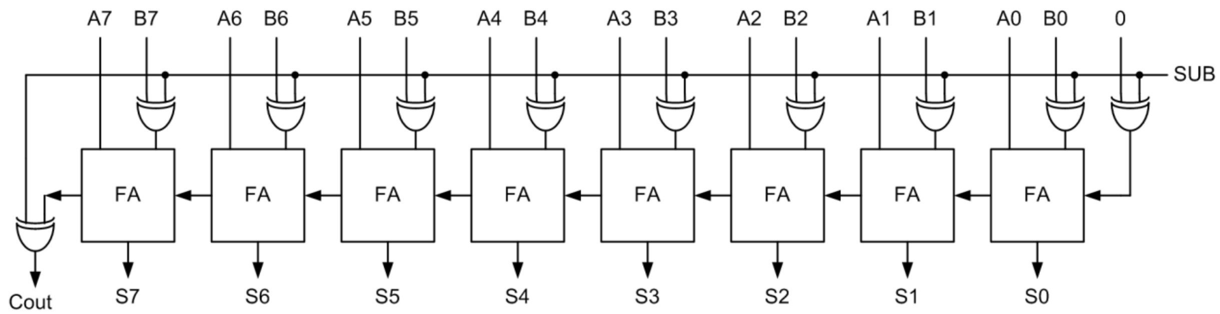
The circuit diagram of an 8-bit adder shows the detailed connections of the eight full adders.
8-bit adder එකක පරිපථ සටහන පූර්ණ ආකලක 8 ක සවිස්තරාත්මක සම්බන්ධතා පෙන්වයි.

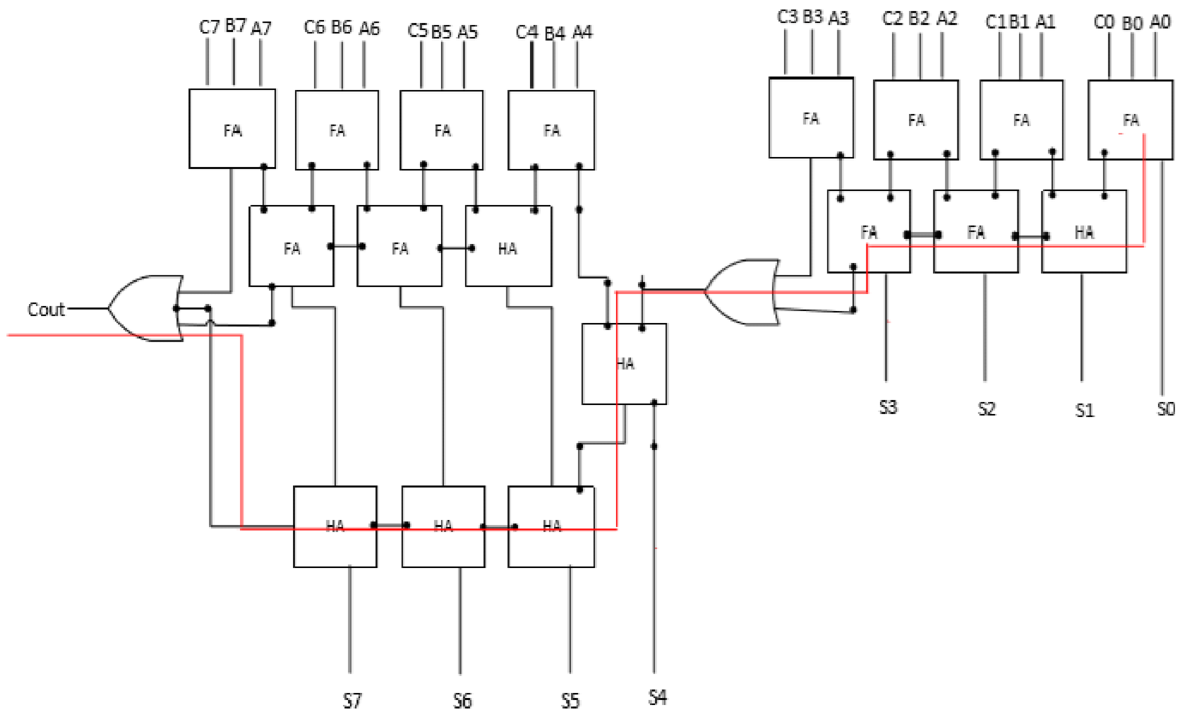
Each full adder consists of logic gates (AND, OR, XOR) that perform the addition operation.
එක් එක් පූර්ණ ආකලක එකතු කිරීමේ මෙහෙයුම සිදු කරන තාර්කික ද්වාර (AND, OR, XOR) වලින් සමන්විත වේ.

The diagram below represents a typical layout for an 8-bit adder.
පහත රූප සටහන 8-bit adder සඳහා සාමාන්‍ය පිරිසැලසුමක් නියෝජනය කරයි.

These diagrams are essential for understanding how binary addition is performed in digital circuits and are fundamental in designing arithmetic units in computers.
මෙම රූප සටහන් සංඛ්‍යාංක පරිපථවල ද්විමය එකතු කිරීම සිදු කරන ආකාරය අවබෝධ කර ගැනීම සඳහා අත්‍යවශ්‍ය වන අතර පරිගණකවල අංක ගණිත ඒකක සැලසුම් කිරීමේදී මූලික වේ.

Some other 8-bit adder circuit diagrams





ictfromabc.com

Storing bits in digital circuits
ඩිජිටල් පරිපථවල බිටු ගබඩා කිරීම

Combinational Circuits
සංයුක්ත පරිපථ

Combinational circuits are types of digital circuits where the output depends only on the current input values.
සංයුක්ත පරිපථ යනු ප්‍රතිදානය වත්මන් ආදාන අගයන් මත පමණක් රඳා පවතින සංඛ්‍යාංක පරිපථ වර්ග වේ.

There is no concept of memory or past input states in these circuits.
මෙම පරිපථවල මතකය හෝ පෙර ආදාන තත්වයන් පිළිබඳ සංකල්පයක් නොමැත.

These circuits perform specific operations like arithmetic calculations, data routing, and logic operations based on the input signals provided at that moment.
මෙම පරිපථ එම මොහොතේ ලබා දී ඇති ආදාන සංඥා මත පදනම්ව ගණිතමය ගණනය කිරීම්, දත්ත මාර්ගගත කිරීම සහ තාර්කික මෙහෙයුම් වැනි විශේෂිත මෙහෙයුම් සිදු කරයි.

Sequential Circuits
අනුක්‍රමික පරිපථ

Sequential circuits, on the other hand, are digital circuits where the output depends on both the current input and the history of past inputs.
අනුක්‍රමික පරිපථ යනු ප්‍රතිදානය වත්මන් ආදානය සහ අතීත ආදාන වල ඉතිහාසය යන දෙකම මත රඳා පවතින සංඛ්‍යාංක පරිපථ වේ.

These circuits have memory elements, typically flip-flops or latches, that store information about previous states.
මෙම පරිපථවල මතක මූලාවයව ඇත, සාමාන්‍යයෙන් පිළි පොළ හෝ latches පෙර තත්වයන් පිළිබඳ තොරතුරු ගබඩා කරයි.

Sequential circuits are essential for tasks like counting, storing data, and executing instructions in a specific sequence.
ගණනය කිරීම, දත්ත ගබඩා කිරීම සහ නිශ්චිත අනුපිළිවෙලකට උපදෙස් ක්‍රියාත්මක කිරීම වැනි කාර්යයන් සඳහා අනුක්‍රමික පරිපථ අත්‍යවශ්‍ය වේ.

Feature	**Combinational Circuits** **සංයුක්ත පරිපථ**	**Sequential Circuits** **අනුක්‍රමික පරිපථ**
Output Dependency ප්‍රතිදාන යැපීම	Only current inputs වත්මන් ආදාන පමණි	Current inputs + past states වත්මන් ආදාන (+) පෙර තත්වයන්
Memory මතකය	No memory මතකයක් නැත	Has memory මතකයක් ඇත
Examples උදාහරණ	Logic gates, adders තාර්කික ද්වාර, ආකලක	Counters, registers ගණක, රෙජිස්තර
Complexity සංකීර්ණතාව	Simpler, faster සරලයි, වේගවත් වේ	More complex වඩාත් සංකීර්ණ වේ
Application භාවිත	Arithmetic and logic operations අංක ගණිතමය සහ තාර්කික මෙහෙයුම්	Data storage, sequencing දත්ත ගබඩා කිරීම,
Clock Signal ඝටිකා සංඥාව	Not required අවශ්‍ය නොවේ	Required අවශ්‍යයි

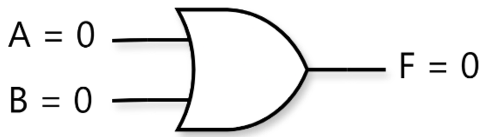
ictfromabc.com

Feedback loops
ප්‍රතිපෝෂණ ලූප

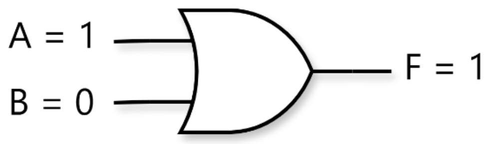
A feedback loop is a process where a logic gate's output is fed back into its input, affecting future behavior.
ප්‍රතිපෝෂණ ලූපයක් යනු තාර්කික ද්වාරයක අනාගත හැසිරීම් වලට බලපාන ආකාරයෙන් එහි ප්‍රතිදානය එහි ආදානයට නැවත ලබා දීමේ ක්‍රියාවලියකි.

To better understand a feedback loops, let's use a real world scenario where a thief tries to break into a house.
ප්‍රතිපෝෂණ ලූපයක් වඩා හොඳින් තේරුම් ගැනීමට, සොරෙකු නිවසකට ඇතුළු වීමට උත්සාහ කරන සැබෑ අවස්ථාවක් යොදා ගනිමු.

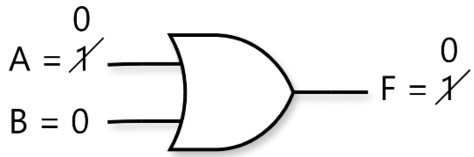
1. When the thief tries to open the door, an alarm should be triggered. Let's assume we have an OR gate set up for triggering the alarm with two inputs; A and B.
සොරා දොර විවෘත කිරීමට උත්සාහ කරන විට, අනතුරු ඇඟවීමේ නලාවක් ක්‍රියාත්මක කළ යුතුය. අනතුරු ඇඟවීම ක්‍රියාත්මක කිරීම සඳහා A සහ B ලෙසින් ආදාන දෙකක් සහිත OR ද්වාරයක් සකසා ඇතැයි සිතමු.



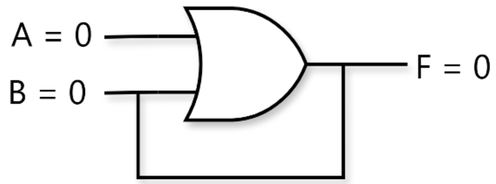
When the thief triggers one of the inputs, the alarm turns on.
සොරා විසින් එක් ආදානයක් ක්‍රියාත්මක කළ විට, අනතුරු ඇඟවීම ක්‍රියාත්මක වේ.



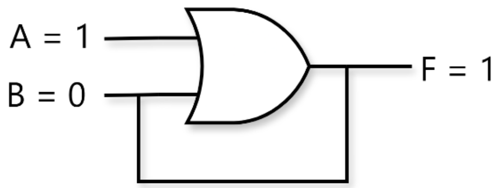
The problem here is, when the thief turns off the trigger, the alarm also turns off.
මෙහි ඇති ගැටළුව වන්නේ සොරා විසින් trigger එක ක්‍රියා විරහිත කළ විට, අනතුරු ඇඟවීම ද ක්‍රියා විරහිත වීමයි.

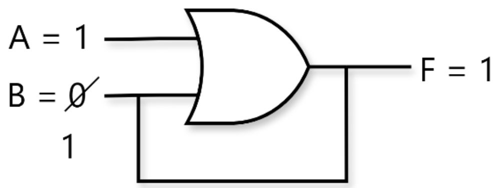


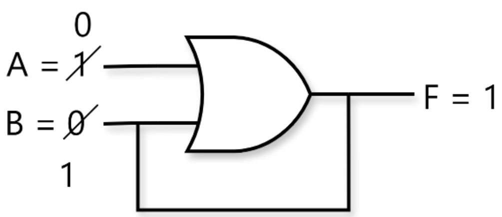
To make our circuit memorize that the thief triggered the alarm, we can use a feedback loop.
සොරා අනතුරු ඇඟවීම ක්‍රියාත්මක කළ බව අපගේ පරිපථයට මතක තබා ගැනීම සඳහා, ප්‍රතිපෝෂණ ලූපයක් භාවිතා කළ හැකිය.



Now, when the thief triggers the alarm, the circuit is going to remember that.
දැන්, සොරා අනතුරු ඇඟවීම ක්‍රියාත්මක කළ විට, පරිපථය එය මතක තබා ගනී.







But if we wanted to stop the alarm at one point, there's no option for us to do that.
නමුත් අපට එක් අවස්ථාවකදී අනතුරු ඇඟවීම නැවැත්වීමට අවශ්‍ය වුවහොත්, එය සිදු කිරීමට කිසිදු විකල්පයක් නොමැත.

Flip Flops
පිළි පොළ

Flip-flops are sequential circuits built using logic gates that store one bit of data.
පිළි-පොළ යනු දත්ත බිටු එකක් ගබඩා කරන තාර්කික ද්වාර භාවිතයෙන් ගොඩනගා ඇති අනුක්‍රමික පරිපථ වේ.

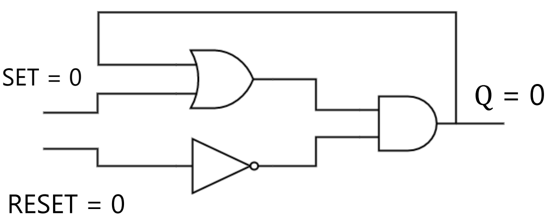
They act as basic memory elements, storing state until an input signal changes it, fundamental in Random-Access Memory (RAM) and storage applications.
ඒවා මූලික මතක මූලාංග ලෙස ක්‍රියා කරන අතර ආදාන සංඥාවකින් එහි තත්ත්වය වෙනස් කරන තෙක් එය ගබඩා කර තබා ගනී. සසම්භාවී-ප්‍රවේශ මතකය (RAM) සහ ගබඩා යෙදුම්වල මූලික වේ.

This is an extended version of a feedback loop.
මෙය ප්‍රතිපෝෂණ ලූපයක දිගු කළ අවස්තාවකි

To reset a triggered feedback loop, a **Reset** input is added to force the output back to zero.
වී ඇති ප්‍රතිපෝෂණ ලූපයක් නැවත සැකසීමට, ප්‍රතිදානය නැවත 0ට සකස් කිරීම සඳහා ආදානයක් එකතු කරනු ලැබේ.

Creating a simple SR latch
සරල SR පරිපථයක් නිර්මාණය කිරීම

Let's modify the feedback loop a step more to get an SR latch
SR ලැච් එකක් ලබා ගැනීමට ප්‍රතිපෝෂණ පුඩුව තවත් පියවරක් වෙනස් කරමු



Initially, both inputs and the output are 0.
මුලදී ආදාන දෙකම සහ ප්‍රතිදානය 0 වේ.

Instance 1 (Set)
When you turn the Set input to 1 OR gate will output 1
ඔබ Set ආදානය 1 ට හරවන විට OR ද්වාරය 1 ප්‍රතිදානය කරයි

As the RESET is still 0, the NOT gate will also output 1.
RESET තවමත් 0 බැවින්, NOT ද්වාරය ද 1 ප්‍රතිදානය කරයි

This will make the AND gate output 1. So the final Q output is now 1.
මෙය AND ද්වාරයේ ප්‍රතිදානය 1 බවට පත් කරයි
එබැවින් අවසාන Q ප්‍රතිදානය දැන් 1 වේ

Instance 2 (Memory)
When you turn the Set input back to 0 again, as we use a feedback loop here, the OR gate will still output 1.
ඔබ Set ආදානය නැවත 0 වෙත හරවන විට, මෙහි ප්‍රතිපෝෂණ ලූපයක් භාවිතා කරන නිසා, OR ද්වාරය තවමත් 1 ප්‍රතිදානය කරයි

So, the final "Q" output will not get affected and remain 1
එබැවින්, අවසාන "Q" ප්‍රතිදානයට බලපෑමක් නැති අතර එය 1 ලෙසම පවතිනු ඇත

This is a memory state
මෙය මතක තත්ත්වයකි

Instance 3 (Reset)
Now, when you turn the Reset to 1
NOT gate will output 0
දැන්, ඔබ Reset ආදානය 1 ට හරවන විට NOT ද්වාරය 0 ප්‍රතිදානය කරයි

So, final output turns 0, as the AND gate outputs 0,
එබැවින්, AND ද්වාරයේ ප්‍රතිදානය 0 නිසා අවසන් ප්‍රතිදානය 0 බවට පත් වේ

Instance 4 (Memory)
When you turn the RESET input back to 0, the final output doesn't get affected and remain 0 since the OR gate still outputs 0.
ඔබ RESET ආදානය නැවත 0 වෙත හරවන විට OR ද්වාරය විසින් තවමත් 0 ප්‍රතිදානය කරන බැවින් අවසාන ප්‍රතිදානයට බලපෑමක් නොවී එය 0 ලෙස ම පවතී

This is also another memory state
මෙයද තවත් මතක තත්ත්වයකි

Universal Flip Flops

As manufacturers can't make specific circuits for each and every situation they make use of the idea of universal circuits.
නිෂ්පාදකයින්ට සෑම අවස්ථාවකම නිශ්චිත පරිපථ සෑදිය නොහැකි බැවින් ඔවුන් විසින් සාර්වත්‍ර ද්වාර සංකල්පය භාවිතා කරයි.

Here, the SR-Latch is made using universal gates so that it can be used anywhere.
මෙහිදී SR-Latch එක ඕනෑම තැනක භාවිතා කළ හැකි පරිදි සාරවත්ත්‍ර ද්වාර මගින් සාදා ඇත.

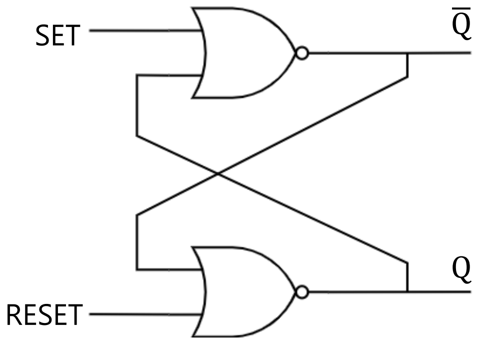
Some circuits may need the opposite (inverse) of the output to for several use cases.
සමහර පරිපථ සඳහා භාවිත අවස්ථා කිහිපයක් වෙනුවෙන් ප්‍රතිදානයේ ප්‍රතිවිරුද්ධය (ප්‍රතිලෝම) අවශ්‍ය විය හැකිය.

To achieve this, there's both an output terminal and an inverse of the output terminal in an SR latch.
මෙය සාක්ෂාත් කර ගැනීම සඳහා, ි(සෘ) එකක ප්‍රතිදාන ප්‍රතිදාන පර්යන්තයකුත්, ප්‍රතිදානයේ ප්‍රතිලෝමය සහිත පර්යන්තයකුත් යන දෙකම ඇත.

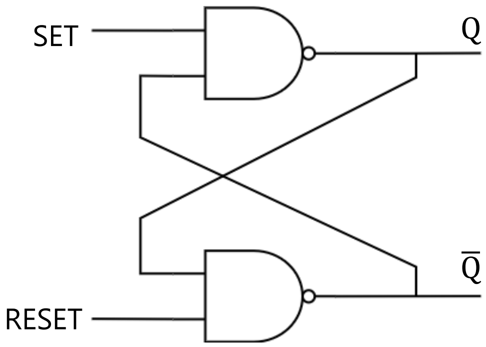
If Q is the output there's also a $\overline{Q}$ output too.
Q යනු ප්‍රතිදානය නම් $\overline{Q}$ යනුවෙන් ප්‍රතිදානයක්ද පවතී.

This can be easily achieved when using universal logic gates.
සාරවත්ත්‍ර ද්වාර භාවිතා කරන විට මෙය පහසුවෙන් සාක්ෂාත් කරගත හැකිය.

SR-Latch using NOR gates



SR-Latch using NAND gates



4 States of a NOR SR-Latch

There are 4 main stages here, lets analyse them using the anti-theft alarm concept.
මෙහි ප්‍රධාන අදියර 4ක් ඇත, anti-theft alarm සංකල්පය භාවිතයෙන් ඒවා විශ්ලේෂණය කරමු.

First let's revise the truth table of a 2 input NOR gate
මුලින්ම ආදාන 2කේ NOR ද්වාරයක සත්‍යතා වගුව දෙස නැවත බලමු.

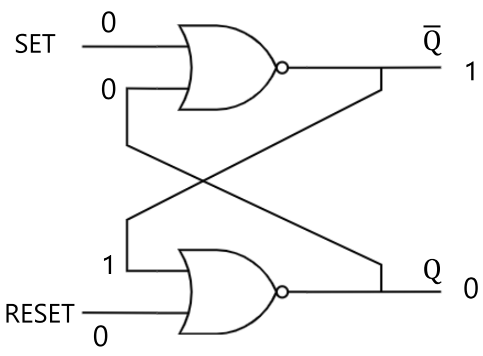
ICN	A	B	F
1	0	0	1
2	0	1	0
3	1	0	0
4	1	1	0

Here the output is 0 even if any input is 1.
මෙහිදී ඕනෑම ආදානයක් 1ක් වුවද ප්‍රතිදානය 0 වේ.

Instance 1 (We're sleeping)

Initially the inputs (Set and Reset) of the circuit are both turned to 0 and the output (Q) is also 0, here we also have the inverse of the output terminal ($\overline{Q}$) so it should be turned to .
මුලදී පරිපථයේ ආදාන වන Set සහ Reset දෙකම 0 ට හරවන අතර (Q) ප්‍රතිදානය ද 0 වේ, මෙහි අපට ප්‍රතිදාන අග්‍රයේ ($\overline{Q}$) ප්‍රතිලෝමය ද ඇති බැවින් එය 1ශ වෙත හැරවිය යුතුය.

First we set the circuit as below.
මුලින්ම අපි පරිපථය පහත පරිදි සකස් කරමු



Instance 2 (Thief Breaks-In)

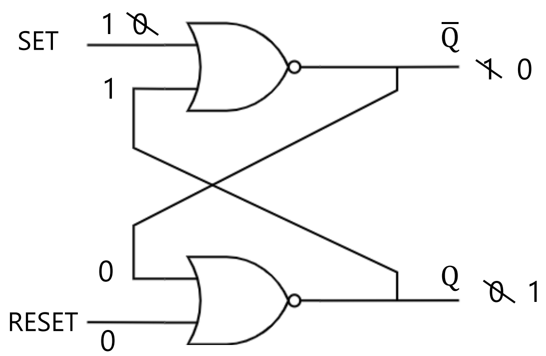
Now the thief opens the car door. Assume that there is a door sensor as the SET input which will turn to 1 when the door is opened.
දැන් සොරා මෝටර් රථයේ එකේ දොර විවෘත කරනු ලබයි. SET ආදානය ලෙස දොරෙහි සවි කර ඇති සංවේදකයක් ඇති බව උපකල්පනය කරන්න, එය දොර විවෘත කළ විට 1 ට හැරෙනු ඇත.

- When the thief opens the door, this will make the SET input to 1.
  සොරා දොර විවෘත කළ විට එමගින් SET ආදානය 1 ට සකස් කරනු ලබයි.

- In a NOR gate, even if one input is 1, the output will be 0, so this will make $\overline{Q}$ turn to 0.
  NOR ද්වාරයක එක් ආදානයක් 1 වුවද ප්‍රතිදානය 0 වනු ඇත, එබැවින් මෙය $\overline{Q}$ ප්‍රතිදානය වෙත හරවනු ඇත.

- This 0 will be fed to the other NOR gate.
  මෙම 0 අනෙක් NOR ද්වාරයට ප්‍රතිපෝෂණය කරනු ලැබේ.

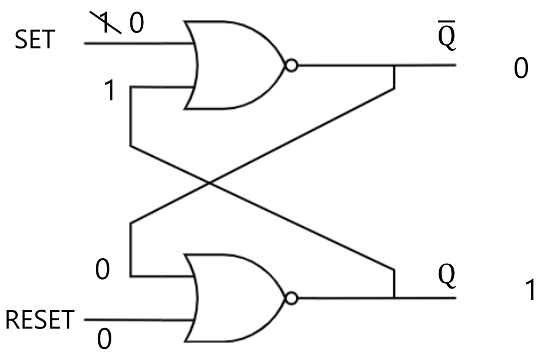
- Now both the inputs of the 2nd NOR gate is 0 it will make the Q output flip to 1
  දැන් 2 වන NOR ද්වාරයෙහි ආදාන දෙකම 0 වේ. එය Q ප්‍රතිදානය 1 ට හරවනු ලබයි.



This will trigger the alarm as the Q output is now 1.
Q ප්‍රතිදානය දැන් 1 වන බැවින් මෙය අනතුරු ඇඟවීම ආරම්භ කරයි

Instance 3 (Thief panics)

Thief will now try to turn off the alarm by closing the door back but will not be successful as this circuit keeps memory.
සොරා දැන් දොර ආපසු වසා අනතුරු ඇඟවීම නවතා දැමීමට උත්සාහ කරනු ඇත, නමුත් මෙම පරිපථය මතකය තබා ගන්නා බැවින් එය සාර්ථක නොවනු ඇත.

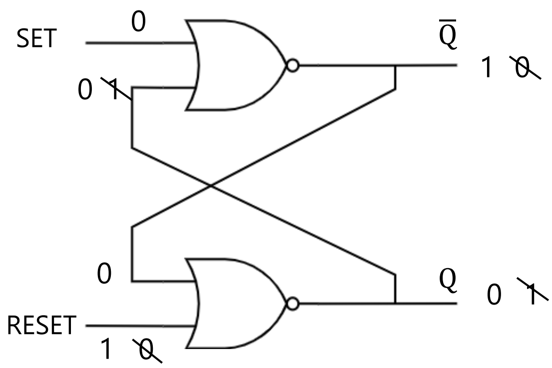


Even though the door is closed, alarm will not turn off as the previous output is fed back to the input of the 1st NOR gate.
දොර වසා තිබුනත්, පෙර ප්‍රතිදානය වන පළමු වන NOR ද්වාරයට නැවත ලබා දෙන බැවින් එලාම් එක ක්‍රියා විරහිත නොවේ.

Instance 4 (Thief Busted)

Now we come in, deal with the thief and turn off the alarm using the car key, here the car key is the RESET input
දැන් අප සොරා අල්ලාගෙන යතුර භාවිතයෙන් අනතුරු ඇඟවීම නවතා දමනු ලබයි. මෙහි යතුර RESET ආදානය වේ

- We make the RESET input 1 using the car key
  අප යතුර භාවිතයෙන් RESET ආදානය 1 කරන්නෙමු
- Even when a single input is 1, the output Q will turn to 0
  තනි ආදානයක් 1 වන විට පවා Q ප්‍රතිදානය 0 වෙත හැරේ
- This 0 will be fed to the other input of the 1st NOR gate
  මෙම 0 පලමු වන NOR ද්වාරයෙහි අනෙක් ආදානයට ප්‍රති පෝෂණය කරයි.
- Now as both the inputs are 0 the output $\bar{Q}$ will turn to 1
  දැන් ආදාන දෙකම 0 වන බැවින් $\bar{Q}$ ප්‍රතිදානය 1 වෙත හැරෙනු ඇත



The alarm will turn off as the Q output is now 0.
Q ප්‍රතිදානය 0 වන බැවින් අනතුරු ඇඟවීම ක්‍රියා විරහිත වේ

4 States of a NAND SR-Latch
NAND SR පරිපථයක අවස්ථා (4)

Here the functioning is completely the opposite to that of a NOR SR-Latch
මෙහි ක්‍රියාකාරීත්වය NOR SR පරිපථයකට සම්පූර්ණයෙන්ම ප්‍රතිවිරුද්ධ වේ

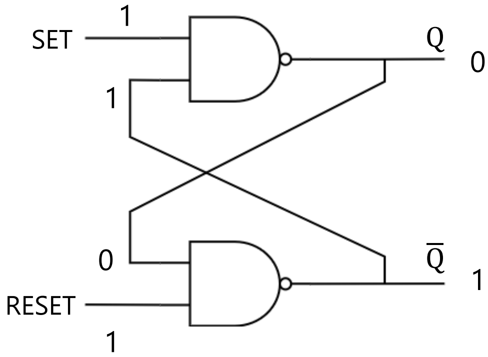
First let's revise the truth table of a 2 input NAND gate
මුලින්ම ආදාන 2කේ NAND ද්වාරයක සත්‍යතා වගුව දෙස නැවත බලමු.

ICN	A	B	F
1	0	0	1
2	0	1	1
3	1	0	1
4	1	1	0

Here, the output is 1 even if any input is 0.
මෙහිදී ඕනෑම ආදානයක් 0 වුවද ප්‍රතිදානය 1 වේ.

Instance 1

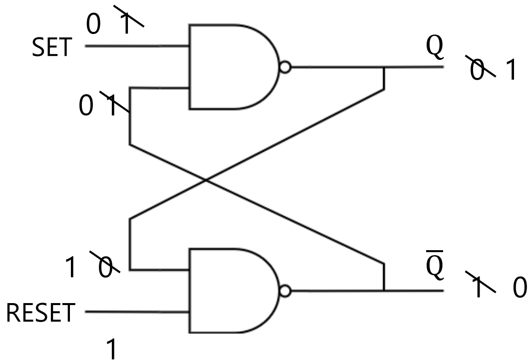
In the initial state both the inputs are set to 1 and the Q output is turned to 0
ආරම්භක තත්වයේදී ආදාන දෙකම 1 ලෙස සකසා ඇති අතර Q ප්‍රතිදානය 0 වෙත හැරේ



Instance 2

Now the SET input is turned to 0.
දැන් SET ආදානය 0 වෙත හැරී ඇත.

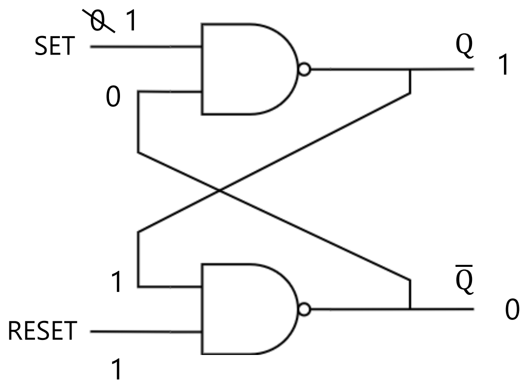
- Even a single input is 0 output Q will turn to 1
  එක් ආදානයක් හෝ 0 වුවද Q ප්‍රතිදානය 1 වෙත හැරේ
- This will be fed as an input to the other NAND gate
  මෙය අනෙක් NAND ද්වාරයට ආදානයක් ලෙස ප්‍රතිපෝෂණය කරනු ලැබේ.
- When both inputs are 1 the output $\bar{Q}$ will turn to 0
  ආදාන දෙකම 1 වන විට $\bar{Q}$ ප්‍රතිදානය 0 වෙත හැරේ



Instance 3

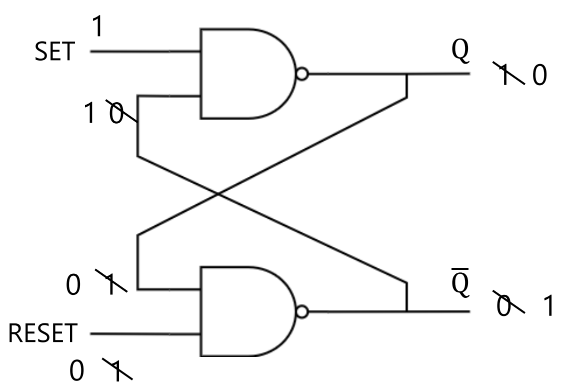
When the laser beam disconnects for a small amount of time and reconnects again the alarm should still sound.
ලේසර් කදම්භය සුළු වේලාවක් විසන්ධි වී නැවත සම්බන්ධ වූ විට අනතුරු ඇඟවීම තවමත් නාද විය යුතුය.

Now, even though the beam is reconnected, alarm will not turn off as the previous output is fed back to the input of the 1st NAND gate
දැන් කදම්භය නැවත සම්බන්ධ කළද, එලාමය ක්‍රියා විරහිත නොවේ, එයට හේතුව වන්නේ පෙර ප්‍රතිදානය පලමු වන NAND ද්වාරයේ ආදානය වෙත නැවත ලබා දෙන බැවිනි.



Instance 4

Now, the RESET input is turned to 0 to turn off the alarm
දැන් එලාම් එක ක්‍රියා විරහිත කිරීම සඳහා RESET ආදානය 0 වෙත හරවනු ලබයි.



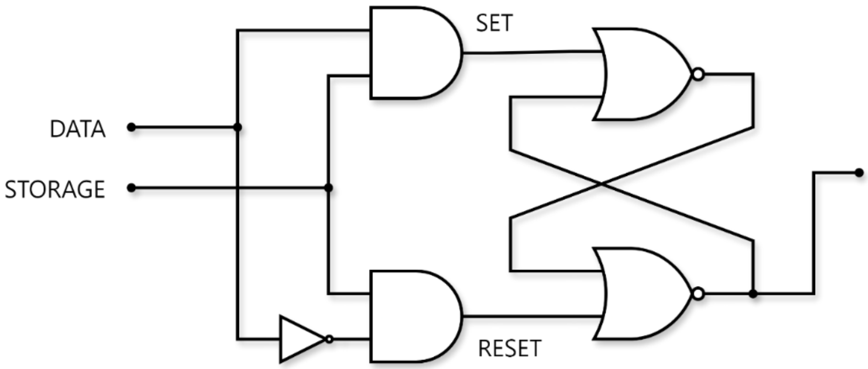
Truth Table for a NOR SR-Latch

S	R	Q	$\overline{Q}$	Description
0	0	0	1	Initial State
1	0	1	0	Alarm trigger state
0	0	1	0	Memory state
0	1	0	1	Alarm off state
0	0	0	1	Memory/initial state
1	1	-	-	Not defined

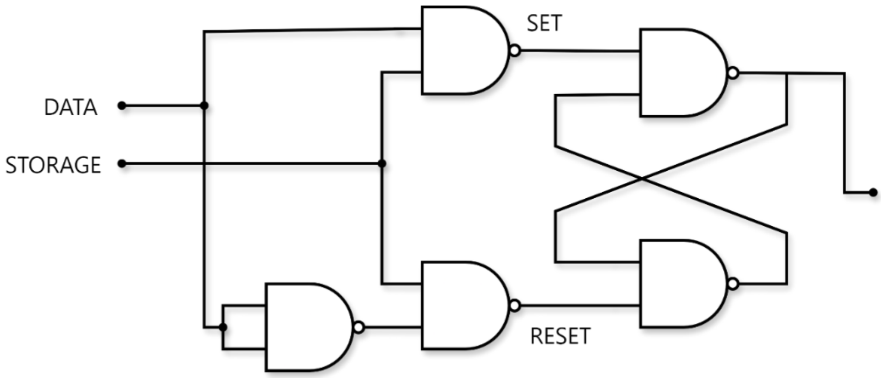
Truth Table for a NAND SR-Latch

S	R	Q	$\overline{Q}$	Description
1	1	0	1	Initial State
0	1	1	0	Alarm trigger state
1	1	1	0	Memory state
1	0	0	1	Alarm off state
1	1	0	1	Memory/initial state
0	0	-	-	Not defined

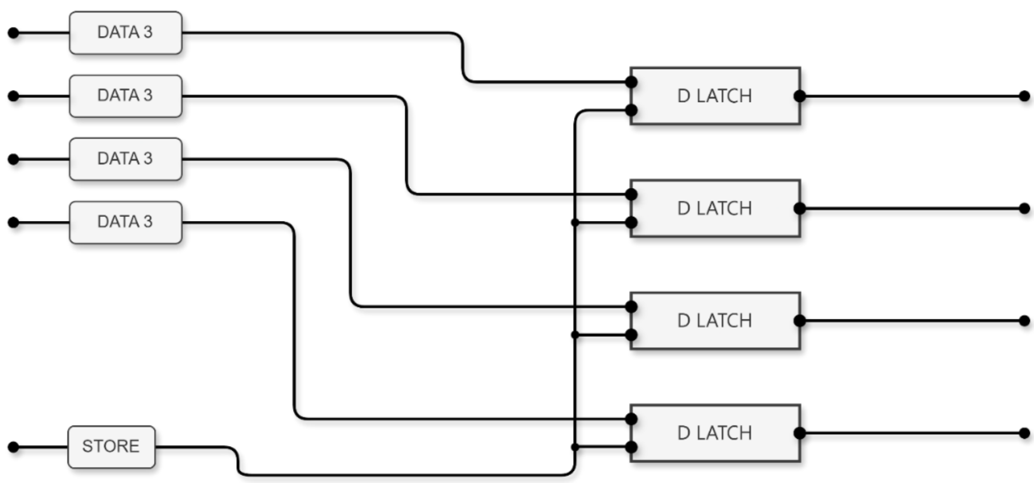
Data Latch using basic logic gates



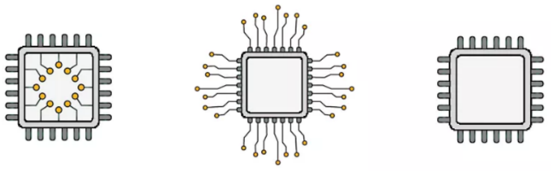
Data Latch using NAND gates

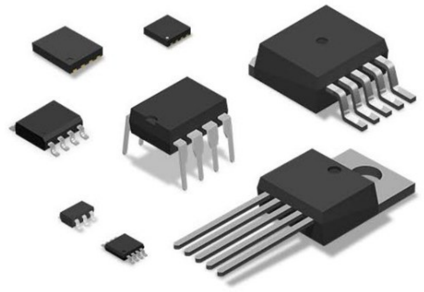


A Collection of Data Latches



What are Integrated Circuits (ICs)?
සංගෘහිත පරිපථ (ICs) යනු කුමක්ද?





An Integrated Circuit (IC) is a compact semiconductor chip that contains multiple electronic components like transistors, resistors, and capacitors, designed to perform specific functions in electronic devices.
සංයුක්ත පරිපථයක් (IC) යනු ඉලෙක්ට්‍රොනික උපාංගවල නිශ්චිත කාර්යයන් ඉටු කිරීම සඳහා නිර්මාණය කර ඇති ට්‍රාන්සිස්ටර, ප්‍රතිරෝධක සහ ධාරිත්‍රක වැනි බහු ඉලෙක්ට්‍රොනික සංරචක අඩංගු සංයුක්ත අර්ධ සන්නායක චිපයකි.

These components are interconnected to execute specific tasks such as amplification, computation, signal processing, and memory storage.
විස්තාරණය, ගණනය කිරීම, සංඥා සැකසීම සහ මතක ගබඩා කිරීම වැනි නිශ්චිත කාර්යයන් ඉටු කිරීම සඳහා මෙම සංරචක අන්තර් සම්බන්ධිත වේ.

ICs have revolutionized the electronics industry by enabling the creation of small, efficient, and complex devices, ranging from simple logic gates to advanced microprocessors.
සරල තාර්කික ද්වාරවල සිට දියුණු මයික්‍රොප්‍රොසෙසර දක්වා කුඩා, කාර්යක්ෂම හා සංකීර්ණ උපාංග නිර්මාණය කිරීමට ICs ඉලෙක්ට්‍රොනික කර්මාන්තයේ විප්ලවීය වෙනසක් සිදු කර ඇත.

4000 Series ICs (CMOS Logic Family)

The 4000 series ICs are based on CMOS (Complementary Metal-Oxide-Semiconductor) technology, known for their low power consumption and high noise immunity.
4000 series ICs , CMOS (Complementary Metal-Oxide-Semiconductor) තාක්ෂණය මත පදනම් වී ඇති අතර, ඒවායේ අඩු බල පරිභෝජනය සහ ඉහළ ශබ්ද ප්‍රතිශක්තිය සඳහා ප්‍රසිද්ධය.

These ICs typically operate within a wide voltage range of 3V to 15V, making them highly versatile for various applications.
මෙම IC සාමාන්‍යයෙන් 3V සිට 15V දක්වා පුළුල් වෝල්ටීයතා පරාසයක් තුළ ක්‍රියාත්මක වන අතර, ඒවා විවිධ යෙදුම් සඳහා ඉතා බහුකාර්ය වේ.

Although they are slower than their TTL counterparts, with propagation delays usually in the tens of nanoseconds, their energy efficiency makes them ideal for battery-powered and portable devices.
ඒවා TTL ට වඩා මන්දගාමී වුවද, සාමාන්‍යයෙන් නැනෝ තත්පර දස ගණනක ප්‍රචාරණ ප්‍රමාදයන් සමඟ, ඒවායේ බලශක්ති කාර්යක්ෂමතාව බැටරි බලයෙන් ක්‍රියාත්මක වන සහ අතේ ගෙන යා හැකි උපාංග සඳහා වඩාත් සුදුසු වේ.

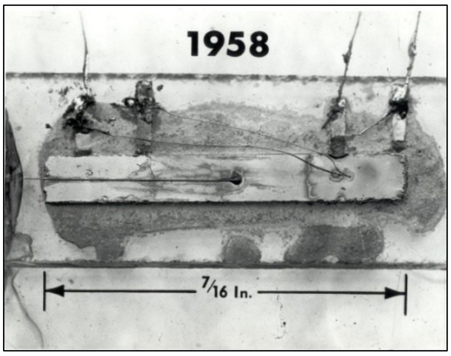
Examples of 4000 series ICs include the CD4001, a quad 2-input NOR gate, and the CD4011, a quad 2-input NAND gate. These ICs are particularly favored in applications where conserving power is a priority.
4000 series ICs සඳහා උදාහරණ ලෙස CD4001, a quad 2-input NOR gate, සහ CD4011, a quad 2-input NAND gate ඇතුළත් වේ. බලය සංරක්ෂණය ප්‍රමුඛතාවයක් වන යෙදුම්වල මෙම ICs විශේෂයෙන් ප්‍රිය කරයි.

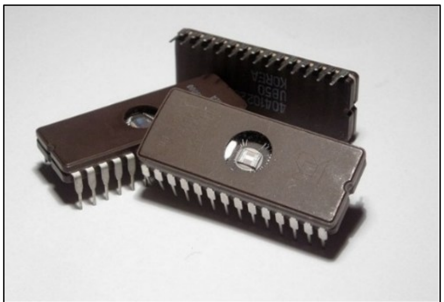
ictfromabc.com

7400 Series ICs (TTL Logic Family)

The 7400 series ICs utilize TTL (Transistor-Transistor Logic) technology and are recognized for their high-speed performance.
7400 series ICs , TTL (Transistor-Transistor Logic) තාක්ෂණය භාවිතා කරන අතර ඒවායේ අධිවේගී ක්‍රියාකාරිත්වය සඳහා පිළිගැනේ.

Operating typically at a fixed 5V, these ICs feature low propagation delays, often just a few nanoseconds, making them suitable for high-speed digital circuits.
සාමාන්‍යයෙන් ස්ථාවර 5V දී ක්‍රියාත්මක වන මෙම ICs බොහෝ විට නැනෝ තත්පර කිහිපයක් පමණක් වන අඩු ප්‍රචාරණ ප්‍රමාදයන් දක්වයි, ඒවා අධිවේගී ඩිජිටල් පරිපථ සඳහා සුදුසු වේ.





25 - April - 1961

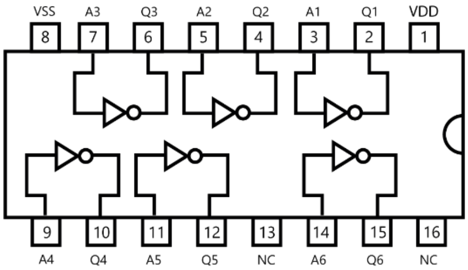
Robert Noyce is granted a patent for an integrated circuit.

However, this speed comes at the cost of higher power consumption and lower noise immunity compared to CMOS ICs.
කෙසේ වෙතත්, මෙම වේගය CMOS IC වලට සාපේක්ෂව ඉහළ බලශක්ති පරිභෝජනයක් සහ අඩු ශබ්ද ප්‍රතිශක්තියක් සමග පැමිණේ.

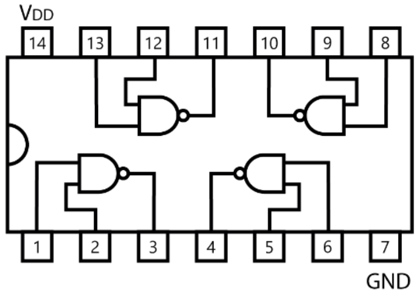
The 7400 series is widely used in computers and other high-speed applications where performance is critical.
7400 ශ්‍රේණිය පරිගණකවල සහ කාර්ය සාධනය ඉතා වැදගත් වන අනෙකුත් අධිවේගී යෙදුම්වල බහුලව භාවිතා වේ.

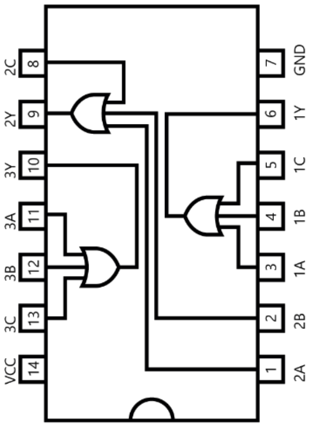
4000 Series ICs (CMOS Logic Family)

4049

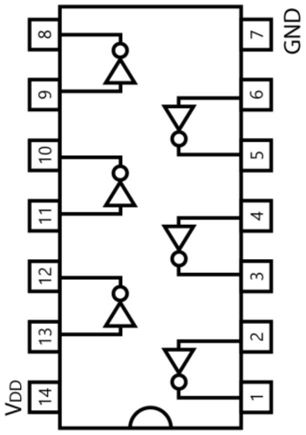


4011 quad NOR gate

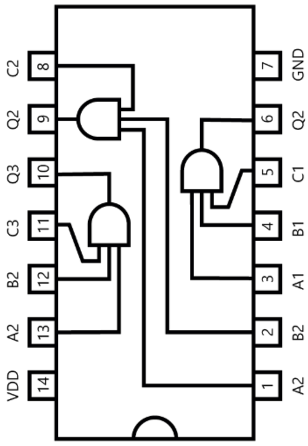




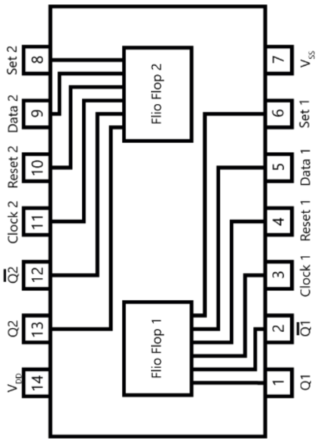
CD4075



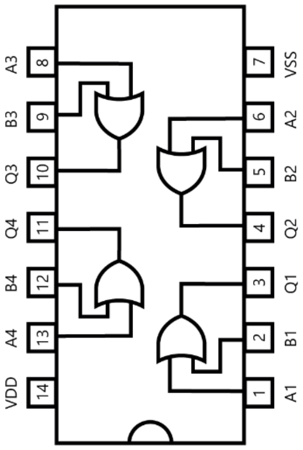
4069



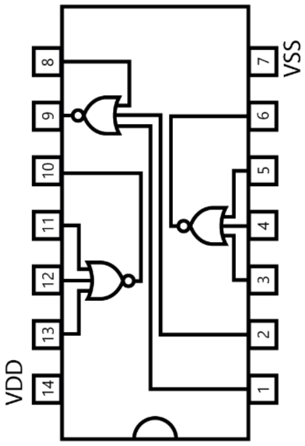
4073



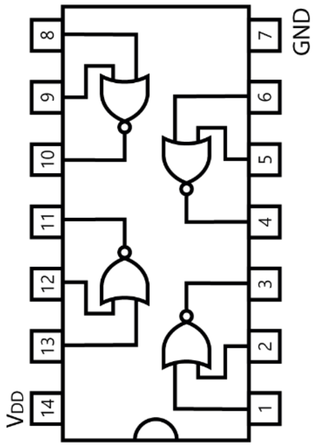
4013 quad NOR gate



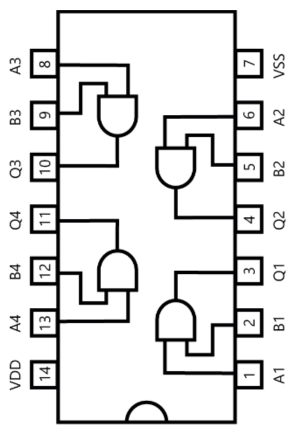
4071



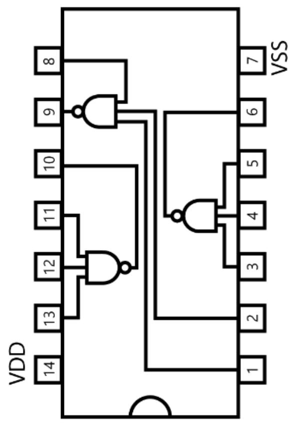
CD4025



4001 quad NOR gate



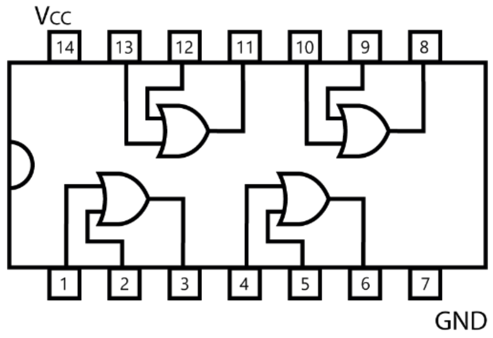
4081B



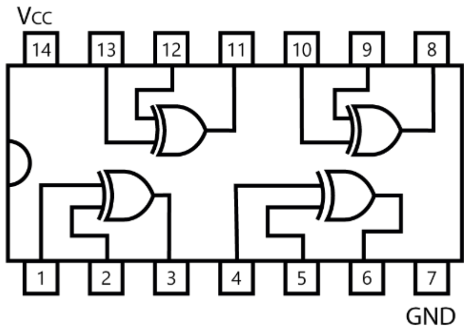
CD4023

7400 Series ICs (TTL Logic Family)

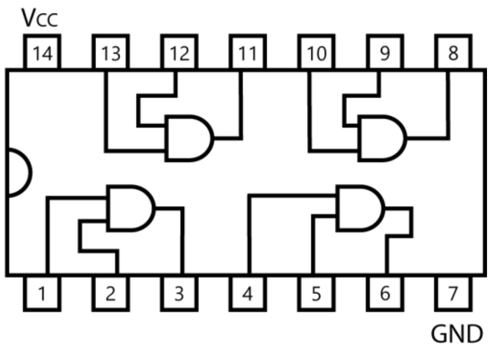
7432 4x2inputXOR



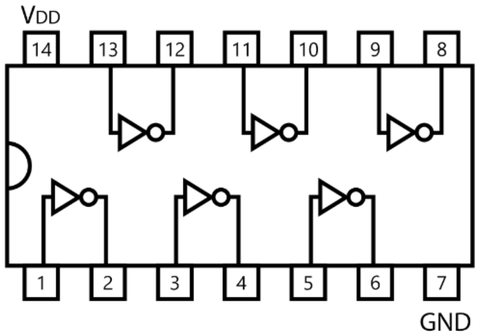
7486 4x2inputXOR



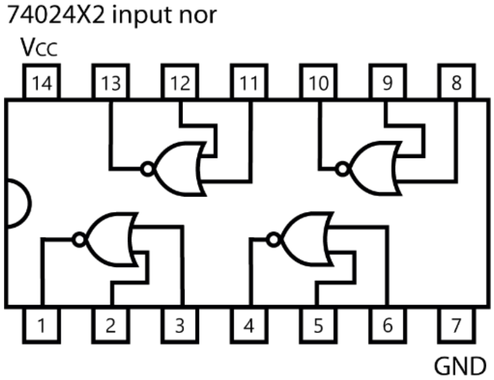
7408 4X2 input nor



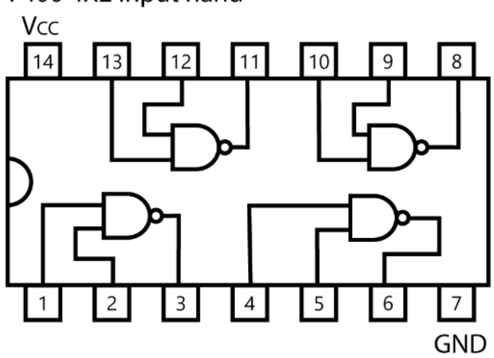
7404 6xNOT



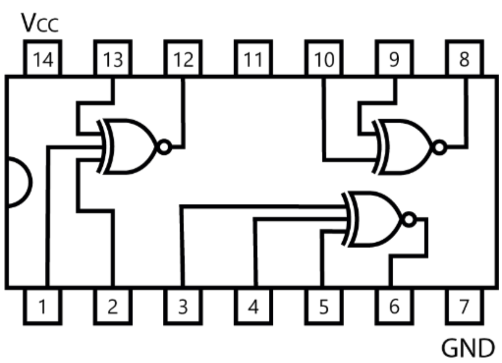
74024X2 input nor



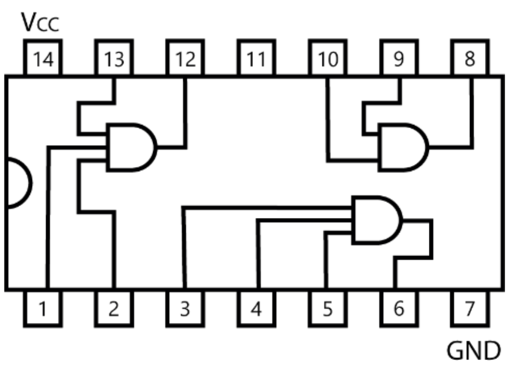
7400 4X2 input nand



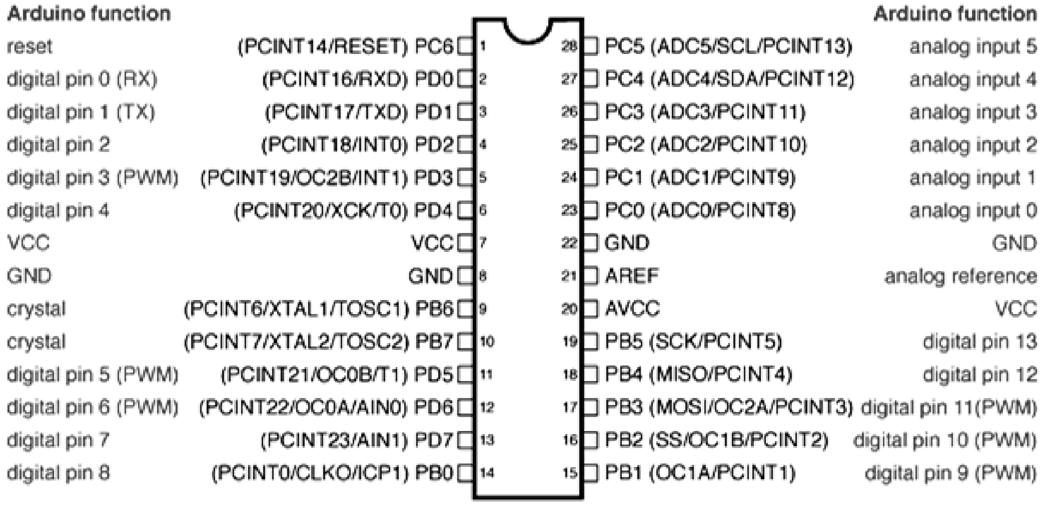
74LS266 PINOUT



74LS11

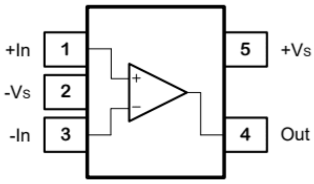


ATMega328P and Arduino Uno Pin Mapping

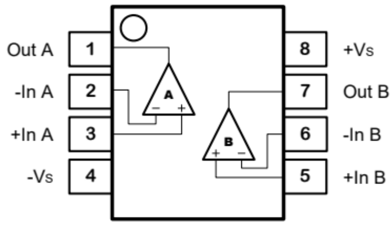


Digital Pins 11,12 & 13 are used by the ICSP header for MOSI, MISO, SCK connections (Atmega168 pins 17,18 & 19). Avoid low-impedance loads on these pins when using the ICSP header.

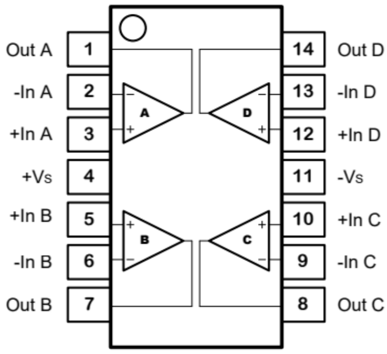
LM321
5-Pin SOT23
(-T Suffix)

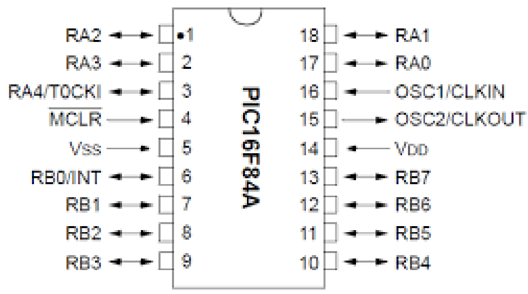


LM358
8-Pin SOIC/MSOP
(-S and -V Suffixes)



LM324
14-Pin SOIC/TSSOP
(-S and -T Suffixes)





LM741 Pinout Diagram