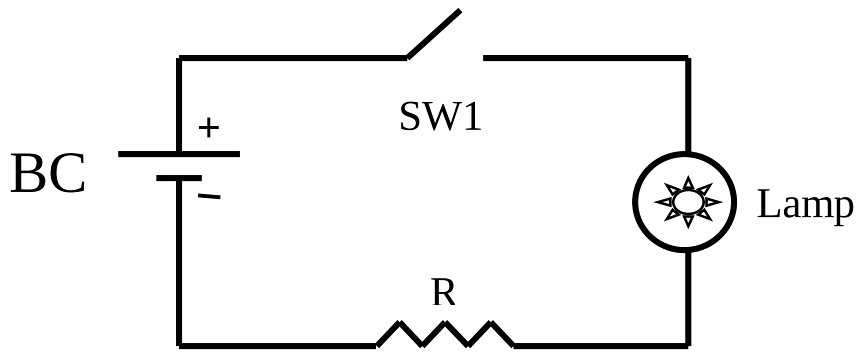
Binary Logic and Boolean algebra

Boolean algebra: Devised for dealing mathematically with philosophical propositions which have ONLY TWO possible values:
TRUE or FALSE, Light ON or OFF.



SW1 **Open** >> Lamp is **OFF**

SW1 **Closed** >> Lamp is **ON**

Two states:

SW1	**Lamp**
OPEN	OFF
CLOSED	ON

"Truth Table"

Electronic Systems:

Analog >> **Continuous** System
Digital >> **Discrete** System

In Boolean algebra the TWO possible conditions can be represented by the DIGITS "0" and "1".

Binary Digits – Bits.

Light ON = "1" = +5V = HIGH
Light OFF = "0" = 0V = LOW

If we define:
Open = "0", CLOSED = "1"

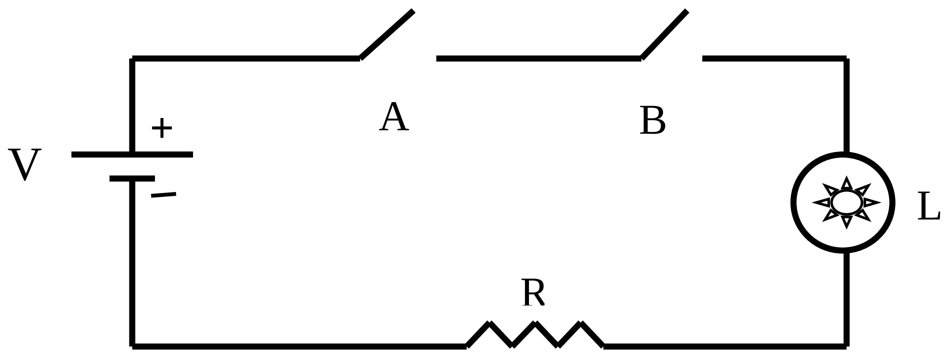
Then:

SW1	**Lamp**
0	0
1	1

Boolean algebra deals with the rules which govern various operations between the binary variables.

“AND” operation: Describes events which can occur <u>**IF and only IF**</u> 2 or more other events are TRUE.

Consider:



The truth table is:

A	B	L
OPEN	OPEN	OFF
OPEN	CLOSED	OFF
CLOSED	OPEN	OFF
CLOSED	CLOSED	ON

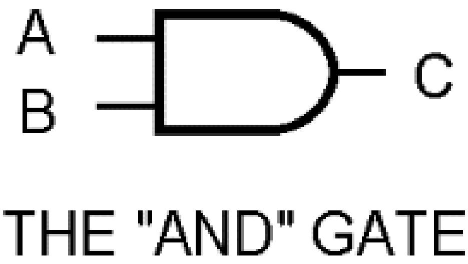
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

Lamp will light **ONLY** when the switches A and B are **CLOSED**, i.e. A and B both “1”

NOTATION: $C = A.B$

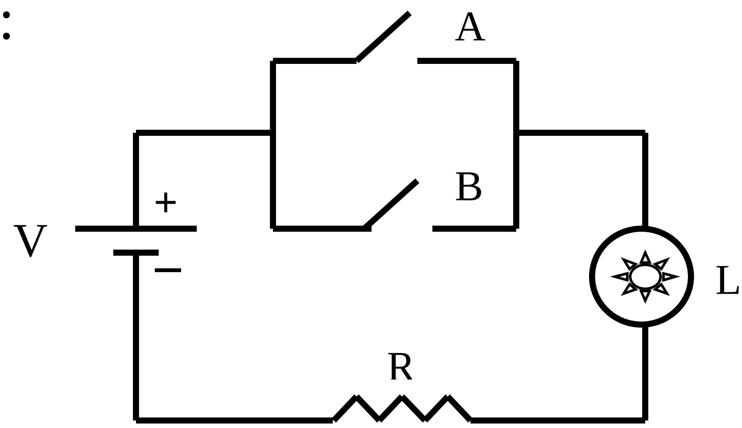
$C = AB$ **Boolean Equation**

SYMBOL: **AND** gate:



"OR" Operation: Describes events which can occur **<u>IF at LEAST ONE</u>** of the other events are TRUE.

Consider:

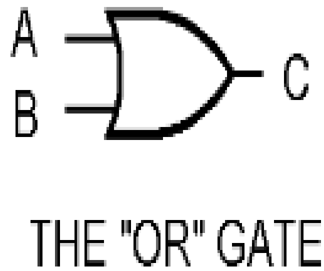


Switches in parallel, lamp will light when A OR B are closed, i.e. A or B = "1" or Both "1"

A	B	L
OPEN	OPEN	OFF
OPEN	CLOSED	ON
CLOSED	OPEN	ON
CLOSED	CLOSED	ON

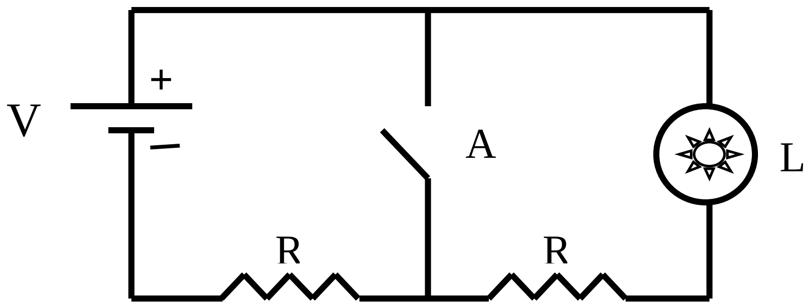
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

NOTATION: C = A + B
SYMBOL**: OR** gate:



"NOT" operation: Changes a statement from TRUE to FALSE and vice–versa, i.e. inversion

Consider:



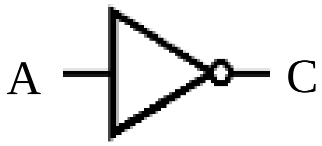
The Truth table is:

A	**L**
OPEN	ON
CLOSED	OFF

A	**L**
0	1
1	0

When A is **CLOSED** virtually NO CURRENT flows through L, so it is effectively **OFF**.

NOTATION: $C = \overline{A}$



SYMBOL: **NOT** gate

BASIC LAWS OF BOOLEAN ALGEBRA

1. **COMUTATIVE LAW:**

$$A + B = B + A$$
$$A . B = B . A$$

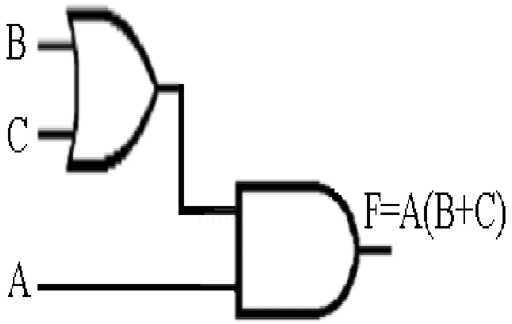
2. **ASSOCIATIVE LAW:**

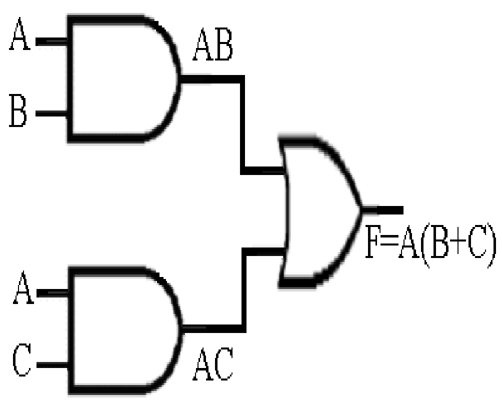
$$A + (B + C) = (A + B) + C$$
$$A(BC) = (AB)C$$

3. **DISTRIBUTIVE LAW:**

$$A(B + C) = AB + AC$$

THESE LAWS CAN BE **EXTENDED** TO INCLUDE ANY NUMBER OF VARIABLES.





BASIC RULES OF BOOLEAN ALGEBRA

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A+B)(A+C) = A+BC$

PROOF OF RULES 10, 11, 12:

Rule 10: $A+AB = A(1+B)$
$= A \cdot 1$ (Rule 2)
$= A$ (Rule 4)

Rule 11: $A + \overline{A}B =$
$= (A + AB) + \overline{A}B$ (Rule 10)
$= (AA + AB) + \overline{A}B$ (Rule 7)
$= AA + AB + A\overline{A} + \overline{A}B$ (Rule 8)
i.e. adding $A\overline{A} = 0$
$= (A+\overline{A})(A + B)$ (Factoring)
$= 1 \cdot (A + B)$ (Rule 6)
$= A + B$

Rule 12: $(A + B)(A + C) =$
$= AA + AC + AB + BC$ (Distrib.)
$= A + AC + AB + BC$ (Rule 7)
$= A(1+C) + AB + BC$ (Distrib.)
$= A \cdot 1 + AB + BC$ (Rule 4)
$= A + AB + BC$ (Rule 2)
$= A(1 + B) + BC$ (Distrib.)
$= A \cdot 1 + BC$ (Rule 2)
$= A + BC$ (Rule 4)

DE MORGAN'S THEOREMS

1. $\overline{AB} = \overline{A} + \overline{B}$

THIS STATES THAT THE INVERSE (i.e.) OF A PRODUCT [**AND**] IS EQUAL TO THE SUM [**OR**] OF THE COMPLEMENTS

2. $\overline{A + B} = \overline{A} \cdot \overline{B}$

THIS STATES THAT THE INVERSE (COMPLEMENT) OF A SUM [**OR**] IS EQUAL TO THE PRODUCT [**AND**] OF THE COMPLEMENTS

TWO VERY IMPORTANT THEOREMS

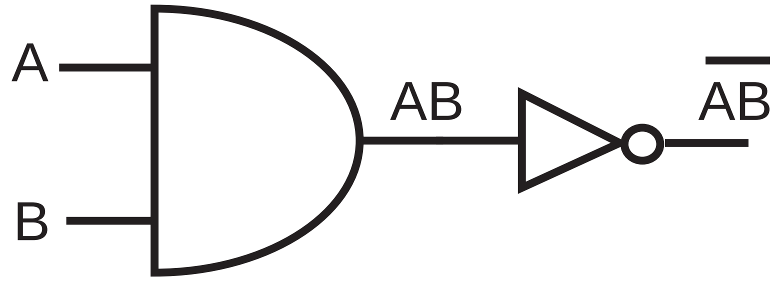
THESE THEOREMS CAN BE EXTENDED TO COVER SEVERAL VARIABLES:

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

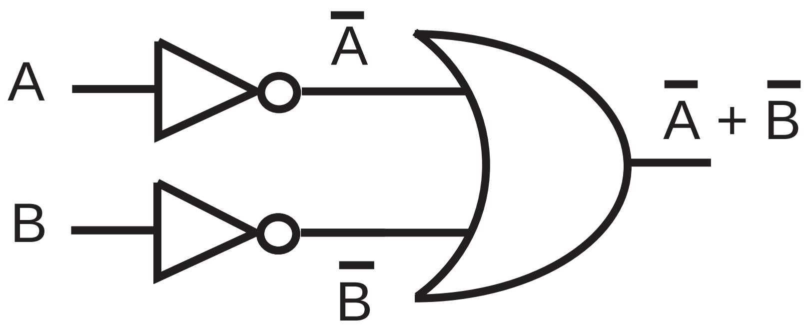
$$\overline{A + B + C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$$

PROOF OF (1): $\overline{AB} = \overline{A} + \overline{B}$

A	**B**	**AB**	$\overline{\textbf{AB}}$	$\overline{\textbf{A}}$	$\overline{\textbf{B}}$	$\overline{\textbf{A}} + \overline{\textbf{B}}$
0	0	0	**1**	1	1	**1**
0	1	0	**1**	1	0	**1**
1	0	0	**1**	0	1	**1**
1	1	1	**0**	0	0	**0**

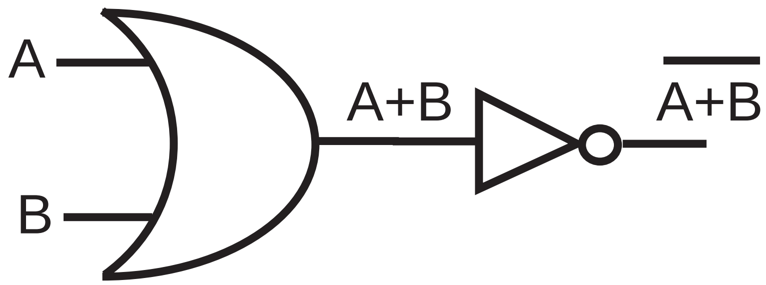


=

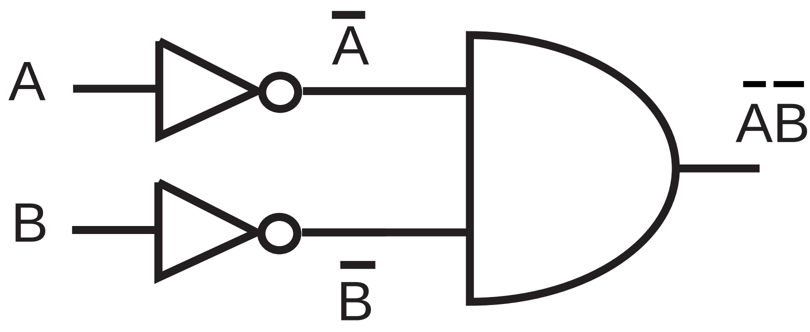


PROOF OF (2): $\overline{A + B} = \overline{A} \,.\, \overline{B}$

A	**B**	**A+B**	$\overline{A}$	$\overline{B}$	$\overline{A+B}$	$\overline{A} \,.\, \overline{B}$
0	0	0	1	1	**1**	**1**
0	1	1	1	0	**0**	**0**
1	0	1	0	1	**0**	**0**
1	1	1	0	0	**0**	**0**



=



"EXCLUSIVE OR" OPERATION

EVENTS WHICH ARE TRUE ONLY **IF AND ONLY IF ONE** OF THE MOTIVATING EVENTS ARE TRUE

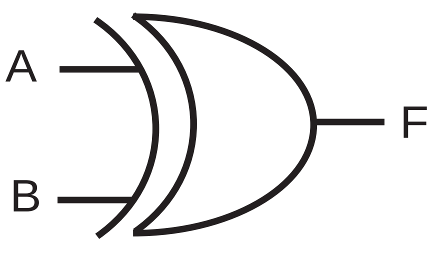
ABREVIATED: **XOR**

TRUTH TABLE:

A	**B**	**A "XOR" B**
0	0	0
0	1	1
1	0	1
1	1	0

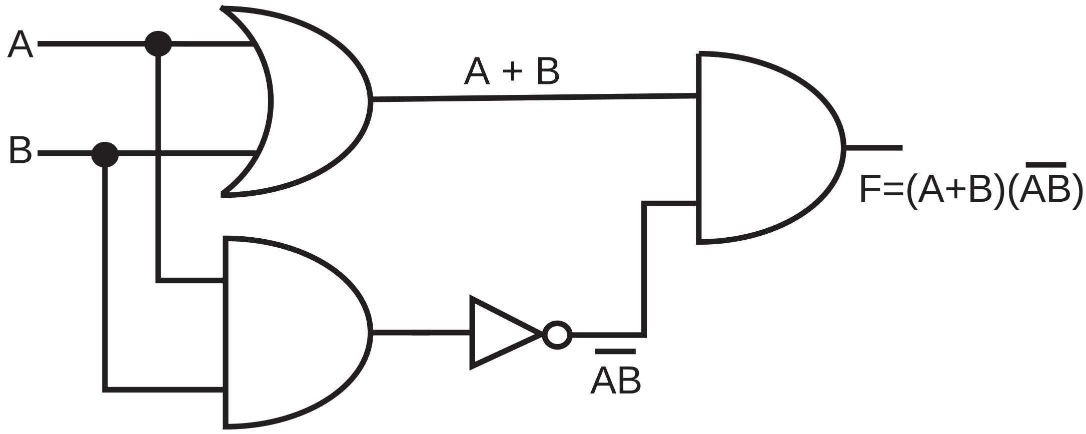
NOTATION: $\mathbf{F = A \oplus B}$

SYMBOL:



F="1" IF A="1" OR B="1" BUT NOT A = B = "1"

$$>> \ \mathbf{F = (A + B)(\overline{AB})}$$



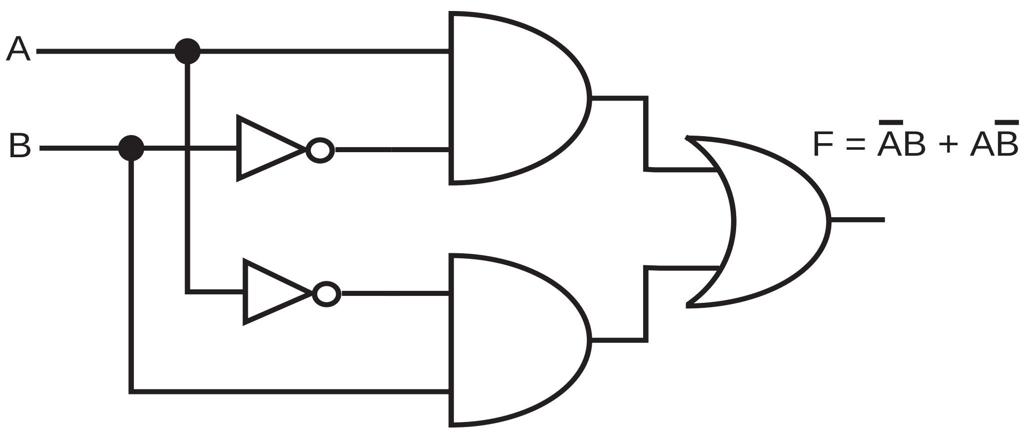
ANOTHER WAY OF EXPRESING XOR

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

$$= (A+B)(\overline{A}+\overline{B}) \qquad \text{(De Morgan)}$$

$$= A\overline{A} + A\overline{B} + A\overline{B} + B\overline{B} \qquad \text{(Distrib.)}$$

$$= A\overline{B} + \overline{A}B \qquad \text{(Rule 8)}$$



“NAND” OPERATION

DE MORGAN’S THEOREMS MEANS THAT ANY BOOLEAN OPERATION CAN BE PERFOMED BY A COMBINATION OF “**AND**” AND “**NOT**” OPERATIONS

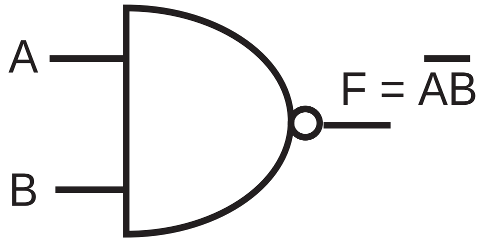
A VERY USEFULL OPERATION IS THE “NAND” i.e. AN “**AND**” OPERATION FOLLOWED BY A “**NOT**” OPERATION

TRUTH TABLE:

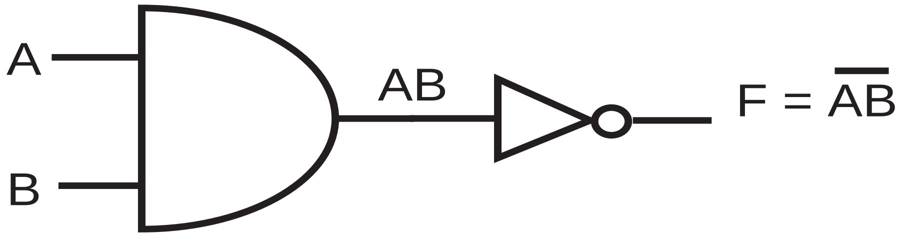
A	**B**	**A “NAND” B**
0	0	1
0	1	1
1	0	1
1	1	0

NOTATION: $\mathbf{F = \overline{A \cdot B}}$

SYMBOL **NAND** GATE:

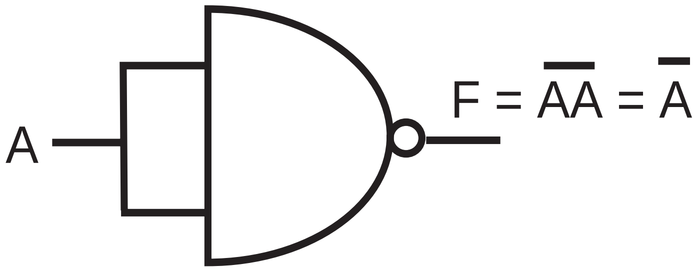


=

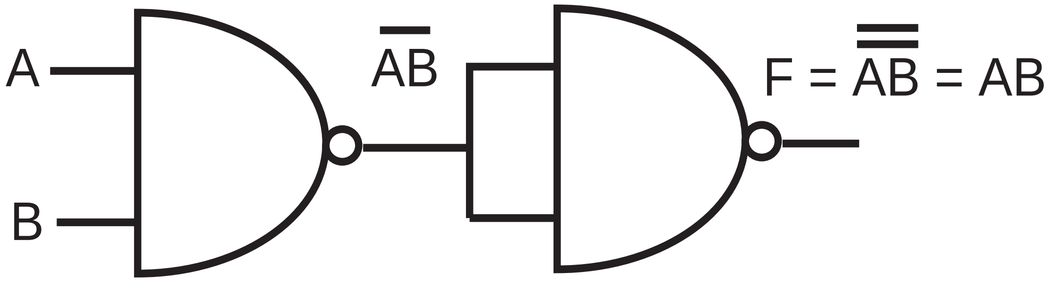


IMPORTANT NOTE: **ANY** BOOLEAN FUNCTION MAY BE IMPLEMENTED USING **ONLY** NAND GATES!

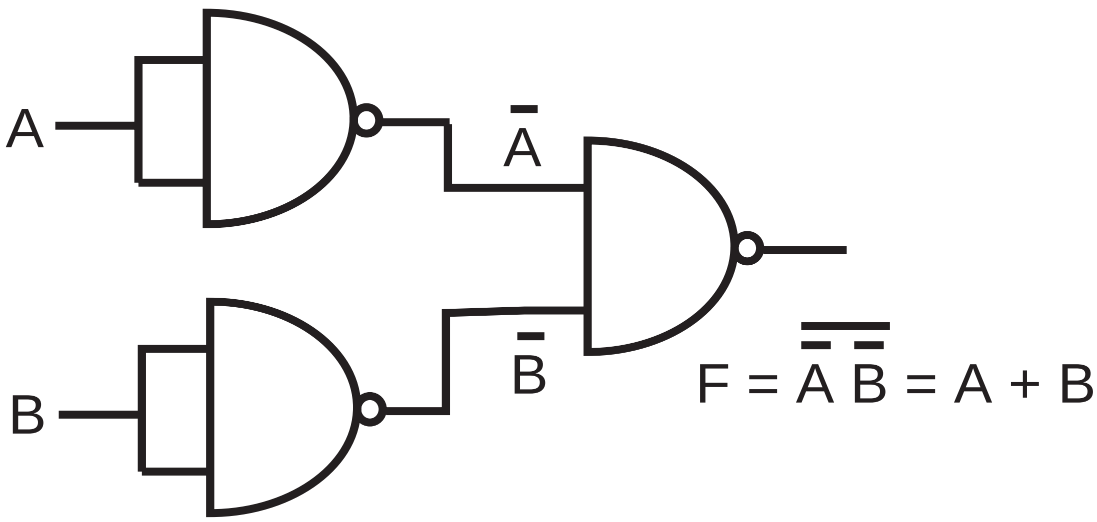
1. “NOT” GATE (INVERTER)



2. "AND" GATE



3. "OR" GATE



"NOR" OPERATION

FROM DE MORGAN'S THEOREMS, WE ALSO CAN EXPRESS **ANY** BOOLEAN FUNCTION IN TERMS OF "**OR**" AND "**NOT**" OPERATIONS

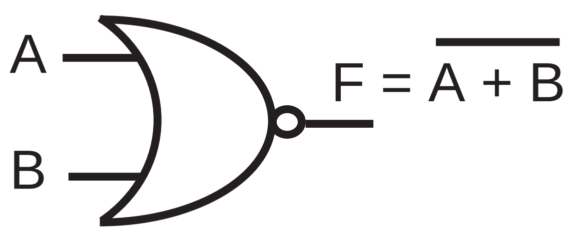
A **"NOR"** OPERATION IS A **"OR"** OPERATION FOLLOWED BY A **"NOT"** OPERATION

TRUTH TABLE:

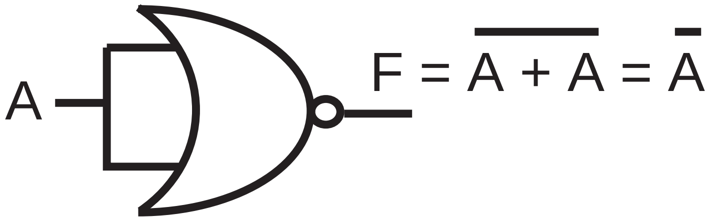
A	**B**	**A "NOR" B**
0	0	1
0	1	0
1	0	0
1	1	0

NOTATION: $F = \overline{A + B}$

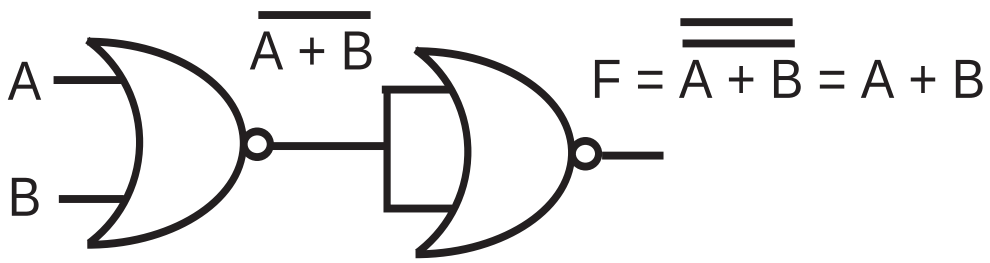
SYMBOL: **NOR** GATE



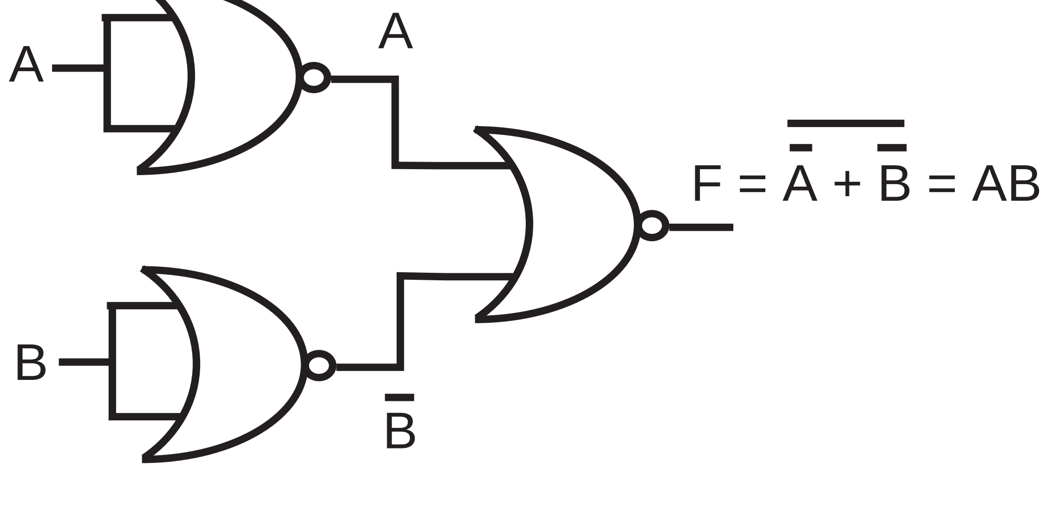
1. “NOT” GATE (INVERTER)



2. “OR” GATE



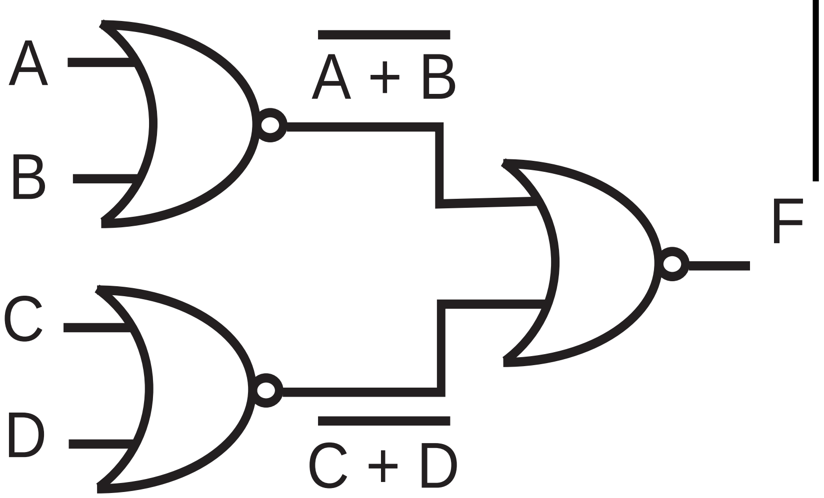
3. “AND” GATE



ANY COMPLEX BOOLEAN FUNCTION CAN BE IMPLEMENTED USING **ONLY** NOR GATES.

E.G. >> $\mathbf{F = (A + B)(C + D)}$

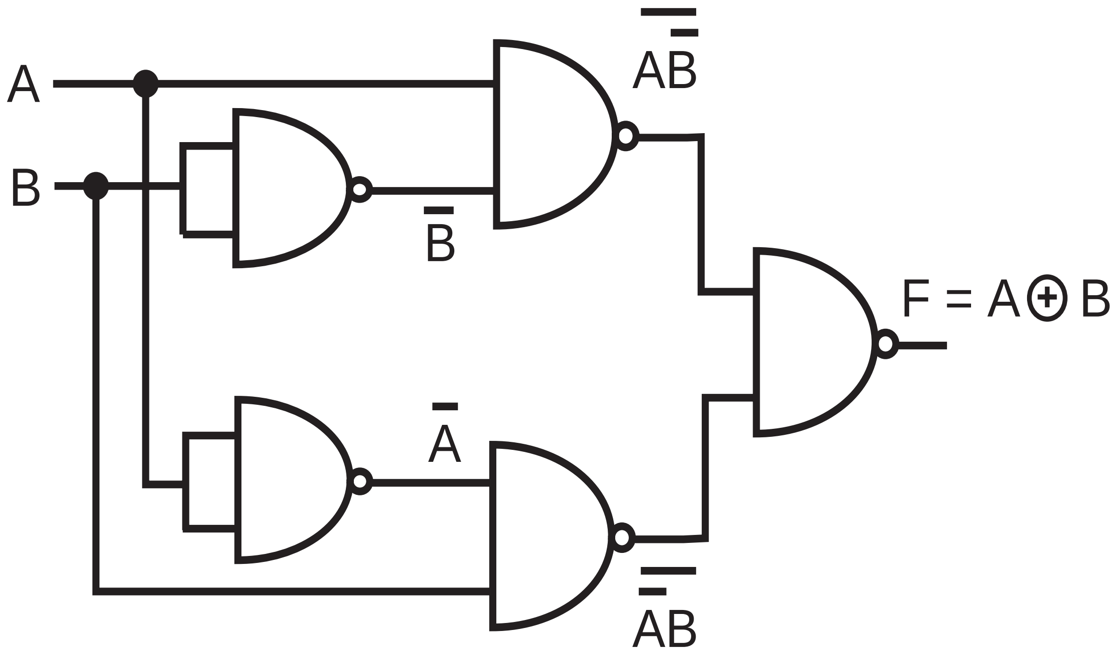
$$= \overline{\overline{(A + B)} + \overline{(C + D)}}$$



IN PRACTICE “**NAND**” GATES ARE USED MAINLY (E.G. 7400) AS THEY ARE THE CHEAPEST.

EXEMPLE 1: IMPLEMENT THE “XOR” OPERATION USING “**NAND**” GATES.

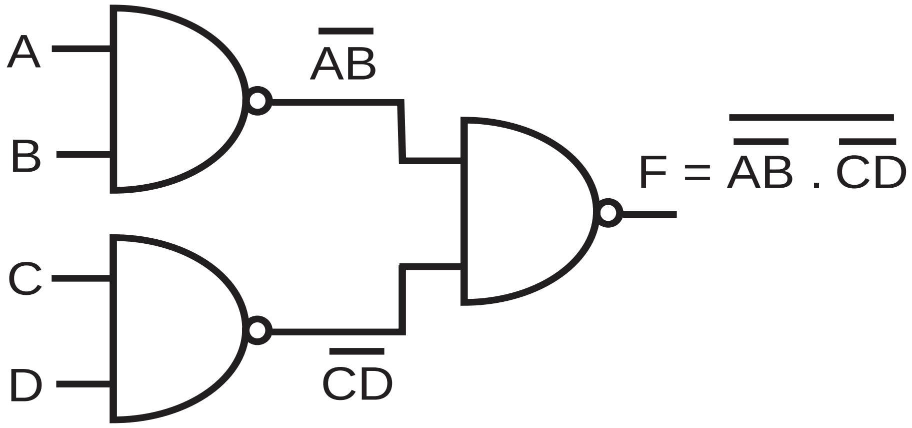
$$\mathbf{F = A \oplus B = A\overline{B} + \overline{A}B = \overline{\overline{A\overline{B}} \cdot \overline{\overline{A}B}}}$$



EXEMPLE 2: IMPLEMENT THE FOLOWING **AND-OR** FUNCTION USING “**NAND**” GATES:

F = AB + CD

$$\mathbf{F = \overline{\overline{AB + CD}} = \overline{\overline{AB} \cdot \overline{CD}}}$$



SUMMARY

- BOOLEAN ALGEBRA: **SYMBOLS, RULES**
- EXPRESS THE LOGICAL FUNCTIONS **AND**, **OR**, **NOT**, **XOR**, **NAND** AND **NOR** MATHEMATICALLY
- BASIC LAWS OF BOOLEAN ALGEBRA AND **HOW** TO APPLY THEM.
- **DE MORGAN'S THEOREMS** AND **HOW** TO APPLY THEM.

LOGIC DESIGN

AIM: TO DESIGN DIGITAL SYSTEMS USING THE RULES OF BOOLEAN ALGEBRA **(FLOYD 4-5/4-6)**.

DESIGNING A LOGIC SYSTEM:

1. **DEFINE** THE PROBLEM
2. **WRITE** THE TRUTH TABLE
3. **WRITE** THE BOOLEAN (OR LOGIC) EQUATIONS
4. **SIMPLIFY** EQUATIONS TO MINIMISE THE NUMBER OF GATES
5. **DRAW** A LOGIC DIAGRAM
6. **IMPLEMENT** THE LOGIC DIAGRAM USING ELECTRONIC CIRCUITRY

NEXT, WE WILL INVESTIGATE **MINIMISATION TECHNIQUES** USING BOOLEAN ALGEBRA LAWS.

EXAMPLE 1:

WE HAVE A CAR WITH 3 MAIN CONTROL SYSTEMS. WE WANT A WARNING LAMP TO LIGHT IF ANY OF THE FOLLOWING CONDITIONS OCCUR:

1. **ALL SYSTEMS** ARE DOWN
2. SYSTEMS **A,B DOWN** BUT **C IS OK**
3. SYSTEMS **A,C DOWN** BUT **B IS OK**
4. SYSTEM **A DOWN** BUT **B,C ARE OK**

1. DEFINE THE PROBLEM

NOTE: THERE ARE TWO POSSIBLE STATES FOR EACH SYSTEM.

ASSIGN:
SYSTEM: **DOWN** = "**0**", **OK** = "**1**"
LIGHT: **OFF** = "**0**", **ON** = "**1**"

LOGIC BLOCK DIAGRAM

1. TRUTH TABLE

A	B	C	F
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

3. WRITE LOGIC EQUATIONS

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

4. SIMPLYFY EQUATIONS

$$F = \overline{A}(\overline{B}\,\overline{C} + \overline{B}C + B\overline{C} + BC)$$

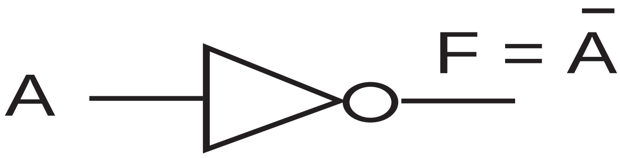
$$= \overline{A}[\overline{B}(\overline{C} + C) + B(\overline{C} + C)]$$

$$= \overline{A}(\overline{B} . 1 + B . 1)$$

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

$$= \overline{A} . 1 = \overline{A} \qquad >> \quad F = \overline{A}$$

5. LOGIC DIAGRAM



EXAMPLE 2:

4 SYSTEMS : **A, B, C** AND **D**

A WARNING BUZZER IS TO SOUND WHEN THE FOLLOWING CONDITIONS OCCUR.

(a) **A** AND **B** ARE **DOWN**
(b) **A,C** AND **D** ARE **DOWN**
(c) **B,C** AND **D** ARE **DOWN**
(d) **B** AND **D** ARE **DOWN**

1. DEFINE THE PROBLEM

SYSTEM: **DOWN** = “**0**”, **OK** = ”**1**”
BUZZER: **OFF** = “**0**”, **ON** = “**1**”

2. TRUTH TABLE:

A	**B**	**C**	**D**	**F**
0	0	0	0	**1** (a) (d)
0	0	0	1	**1** (a)
0	0	1	0	**1** (a) (d)
0	0	1	1	**1** (a)
0	1	0	0	**1** (b)
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	**1** (c) (d)
1	0	0	1	0
1	0	1	0	**1** (d)
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

3. LOGIC EQUATION:

$$F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD$$

$$+ \overline{A}B\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C\overline{D}$$

4. SYMPLIFY:

$$F = \overline{A}\,\overline{B}\,\overline{C}(\overline{D} + D) + \overline{A}\,\overline{B}C(\overline{D} + D) + \overline{A}B\overline{C}\,\overline{D}$$

$$+ A\overline{B}\,\overline{D}\,(\overline{C} + C)$$

$$= \overline{A}\,\overline{B}\,\overline{C}\ .\ 1 + \overline{A}\,\overline{B}C\ .\ 1 + A\overline{B}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D}$$

$$= \overline{A}\,\overline{B}(\overline{C} + C) + A\overline{B}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D}$$

$$= \overline{A}\,\overline{B}\ .\ 1 + A\overline{B}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D}$$

$$= \overline{B}(\mathbf{\overline{A} + A\overline{D}}) + \overline{A}B\overline{C}\,\overline{D} \qquad (*)$$

$$= \overline{B}(\overline{A} + \overline{D}) + \overline{A}B\overline{C}\,\overline{D} =$$

$$= \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D}$$

$$= \overline{A}\,\overline{B} + \overline{D}(\overline{B} + \overline{A}B\overline{C})$$

$$= \overline{A}\,\overline{B} + \overline{D}(\mathbf{\overline{B} + B\overline{A}\overline{C}}) \qquad (*)$$

$$= \overline{A}\,\overline{B} + \overline{D}\,\overline{B} + \overline{A}\,\overline{C}\,\overline{D}$$

>> $F = \overline{A}\,\overline{B} + \overline{B}\,\overline{D} + \overline{A}\,\overline{C}\,\overline{D}$

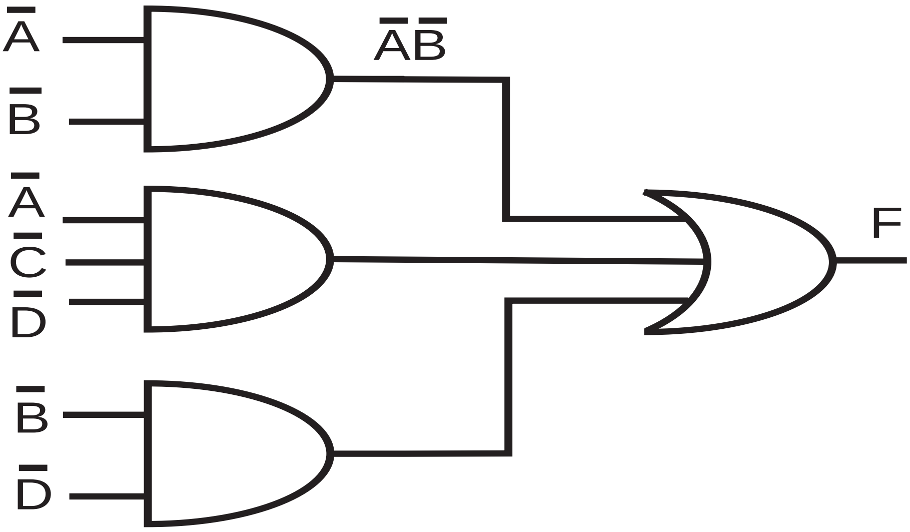
>> SUM OF PRODUCT FORM

NOTE: $A + \overline{A}B = A + B$ (Rule 11)

5. LOGIC DIAGRAM

ASSUMNING $\mathbf{\bar{A},\bar{B},\bar{C},\bar{D}}$ ARE AVAILABLE AS INPUTS, WE CA IMPLEMENT THIS 3 WAYS:

(a) AND –OR CONFIGURATION



(b) NAND CONFIGURATION:

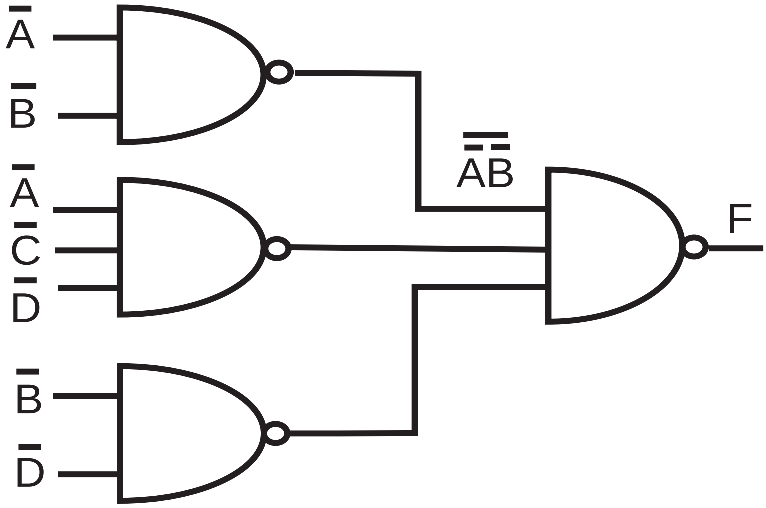
$$F = \bar{A}\bar{B} + \bar{B}\bar{D} + \bar{A}\bar{C}\bar{D}$$

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

$$= \overline{\bar{A}\bar{B}} \,.\, \overline{\bar{B}\bar{D}} \,.\, \overline{\bar{A}\bar{C}\bar{D}} \qquad \text{(De Morgan)}$$

$$F = \bar{\bar{F}}$$

$$= \overline{\overline{\bar{A}\bar{B}} \,.\, \overline{\bar{B}\bar{D}} \,.\, \overline{\bar{A}\bar{C}\bar{D}}}$$

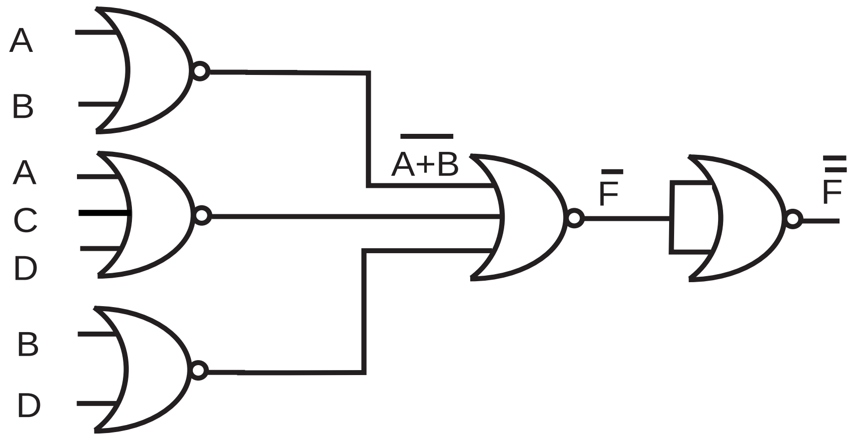


(c) NOR CONFIGURATION:

$$F = \bar{A}\bar{B} + \bar{B}\bar{D} + \bar{A}\bar{C}\bar{D}$$

$$= \overline{A+B} + \overline{B+D} + \overline{A+C+D} \quad \text{(DM)}$$

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



NOTE: MORE GATES >>> **LONGER** PROPAGATION DELAYS, i.e. TIME FOR SIGNAL TO GO FROM INPUT TO OUTPUT

MINIMISATION USING KARNAUGH MAPS

WE HAVE SEEN THAT MINIMISATION USING BOOLEAN ALGEBRA IS A BIT **CUMBERSOME**.

WE CAN REPRESENT ANY LOGICAL EXPRESION ON A DIAGRAM CALLED A **KARNAUGH** MAP.

THIS MAP PROVIDES A **SYSTEMATIC** METHOD OF SIMPLYFYING A BOOLEAN FUNCTION TO PRODUCE THE SIMPLEST SUM OF PRODUCTS EXPRESION

KARNAUGH MAP FORMAT:

FOR **N** VARIABLES WE HAVE $\mathbf{2^N}$ COMBINATIONS, EACH COMBINATION IS CONTAINED IN A **KARNAUGH CELL**

FOR 2 VARIABLES A, B:

2^2 **= 4** PRODUCTS >> **4 CELLS**:

$\mathbf{\overline{A}\overline{B}}$, $\mathbf{\overline{A}B}$, $\mathbf{A\overline{B}}$ AND **AB**. THIS IS REPRESENTED IN A KARNAUGH MAP AS FOLLOWS:

	$\overline{B}$	B
$\overline{A}$	$\overline{A}\overline{B}$	$\overline{A}B$
A	$A\overline{B}$	AB

THE KARNAUGH MAP IS FILLED IN BY PUTTING A "**1**" IN EACH CELL THAT LEADS TO A "**1**" OUTPUT. "**0**" IS PLACED IN ALL THE OTHER CELLS.

EXAMPLE:

(a) REPRESENT **F =** $\mathbf{\overline{X}\overline{Y}}$
BY ITS KARNAUGH MAP

	$\overline{X}$	X
$\overline{Y}$	1	0
Y	0	0

F =**1** WHEN $\mathbf{\overline{X}\overline{Y}}$ OCCURS

OTHER >> F=**0**

(a) REPRESENT **$F = \bar{X}Y + X\bar{Y}$**

	$\bar{X}$	X
$\bar{Y}$	0	1
Y	1	0

F=1 FOR PRODUCT TERMS **$\bar{X}Y$**, **$X\bar{Y}$** ONLY.

THE EXPRESSION FOR "F" MUST BE WRITTEN "**SUM-OF-PRODUCTS**" FORM TO BEGIN WITH.

FOR 3 VARIABLES A,B,C

2^3 = 8 PRODUCT TERMS >> **8 CELLS**

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}BC$	$\bar{A}B\bar{C}$
A	$A\bar{B}\bar{C}$	$A\bar{B}C$	ABC	$AB\bar{C}$

EXAMPLE:

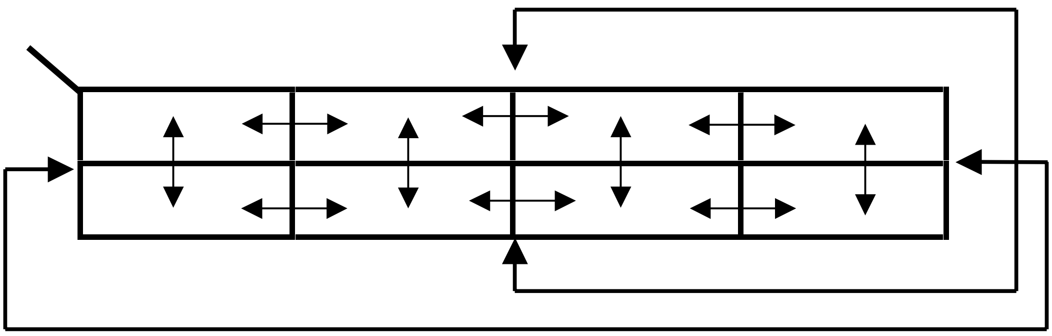
$F = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$

DO AS EXERCISE.

RESULT:

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	0	0
A	1	1	0	0

NOTE: WHEN MOVING **HORIZONTALY** OR **VERTICALLY** WE SHOULD **ONLY** ENCOUNTER A CHANGE IN **1 VARIABLE**.



FOR 4 VARIABLES A,B,C,D

2^4 = **16** PRODUCT TERMS >> **16 CELLS**

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$	$\overline{A}\,\overline{B}\,\overline{C}D$	$\overline{A}\,\overline{B}CD$	$\overline{A}\,\overline{B}C\overline{D}$
$\overline{A}B$	$\overline{A}B\overline{C}\,\overline{D}$	$\overline{A}B\overline{C}D$	$\overline{A}BCD$	$\overline{A}BC\overline{D}$
AB	$AB\overline{C}\,\overline{D}$	$AB\overline{C}D$	$ABCD$	$ABC\overline{D}$
$A\overline{B}$	$A\overline{B}\,\overline{C}\,\overline{D}$	$A\overline{B}\,\overline{C}D$	$A\overline{B}CD$	$A\overline{B}C\overline{D}$

EXAMPLE:

A	B	C	D	F
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**1**

$$F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + \bar{A}BC\bar{D}$$
$$+ A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + ABCD$$

K-MAP:

AB \ CD	0 0	0 1	1 1	1 0
0 0	1	1	0	0
0 1	0	1	0	1
1 1	0	0	1	0
1 0	0	1	0	1

NOTE: FOR A 4 VARIABLE K-MAP EACH CELL HAS 4 ADJACENT CELLS

DON'T FORGET THAT THE K-MAP IS CONSIDERED **CONTINOUS** (ROLLED OVER) SO THAT THE **TOP ROW** IS ADJACENT TO THE **BOTTOM ROW** AND THE **RIGHT COLUMN** IS ADJACENT TO THE **LEFT COLUMN**

EXAMPLE: THE 4 CELLS ADJACENT TO $\bar{A}\bar{B}\bar{C}\bar{D}$ ARE $A\bar{B}\bar{C}\bar{D}$, $\bar{A}B\bar{C}\bar{D}$, $\bar{A}\bar{B}C\bar{D}$ AND $\bar{A}\bar{B}\bar{C}D$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	●	#		#
$\bar{A}B$	#			
AB				
$A\bar{B}$	#			

MINIMISATION
BY GROUPING CELLS

WE CAN MINIMISE ANY BOOLEAN EXPRESSION BY GROUPING **ADJACENT CELLS CONTAINING “1”s** ACCORDING TO THE FOLLOWING RULES:

1. ADJACENT CELLS ARE CELLS THAT ONLY DIFFER BY A **SINGLE** VARIABLE. >> E.G. **ABCD** AND **ABC$\overline{D}$**
2. THE **“1”s** IN ADJACENT CELLS MUST BE COMBINED IN GROUPS OF $\mathbf{2^N}$, I.E. 1,2,4,8,16 … ETC.
3. EACH **GROUP OF “1”s** SHOULD BE **MAXIMISED** TO INCLUDE THE LARGEST NUMBER OF ADJACCENT CELLS POSSIBLE IN ACCORDANCE WITH **RULE 2**
4. EVERY “1” ON THE MAP MUST BE INCLUDED IN **AT LEAST** ONE GROUP (OR **SUBCUBE**). THERE CAN BE OVERLAPPING GROUPS IF THEY INCLUDE NON_COMMON “1”s

PROCEDURE: WE DRAW A LOOP ABOUT THE CELLS IN ORDER TO DEFINE OUR SUBCUBE.

EXAMPLES:

AB \ CD	0 0	0 1	1 1	1 0
0 0	0	0	1	1
0 1	1	1	1	1
1 1	1	1	1	1
1 0	0	1	0	0

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	0	0	0
A	1	1	0	0

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	0	0	0
A	1	0	0	1

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1	0	0	1
A	1	0	0	1

EXERCISE:

DRAW THE SUBCUBES FOR THE FOLLOWING EXPRESSIONS:

(a) $F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}C\overline{D} + A\overline{B}C\overline{D}$

(b) $F = \overline{A}B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + \overline{A}BCD + \overline{A}BC\overline{D}$

$+ AB\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + ABC\overline{D}$

(a)

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	0	0	1
$\overline{A}B$	0	0	0	0
AB	0	0	0	0
$A\overline{B}$	1	0	0	1

(b)

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	0	0	0	0
$\overline{A}B$	1	1	1	1
AB	1	1	1	1
$A\overline{B}$	0	0	0	0

SYMPLIFYING THE EXPRESSION

1. EACH SUBCUBE OF "**1**"s CREATES A PRODUCT TERM COMPOSED OF ALL VARIABLES THAT APPEAR IN **ONLY ONE FORM** (COMPLEMENTED OR NOT) WITHIN THE GROUP. VARIABLES THAT APPEAR BOTH **UNCOMPLEMENTED** AND **COMPLEMENTED** ARE ELLIMINATED

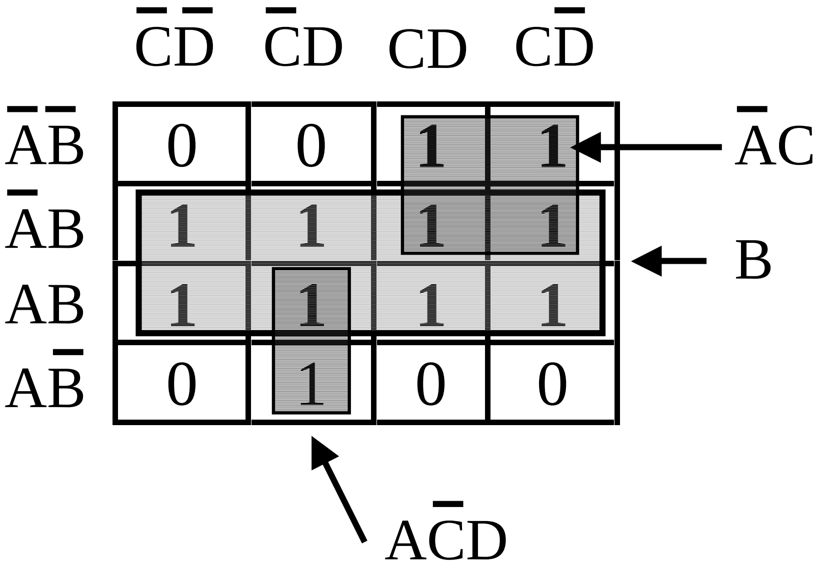
I.E. >> THE ONE THAT CHANGES **WE DROP**

2. THE **FINAL** SIMPLIFIED EXPRESSION IS FORMED BY **SUMMING** THE PRODUCT TERMS OF ALL THE SUBCUBES

THIS WILL BECAME CLEARER AFTER WE LOOK AT SOME EXAMPLES.

EXAMPLES:

1.



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

1. FROM THE 8 CELL GROUP THE PRODUCT TERM IS **B**, WITH THE REFERENCE TO RULE 1, **A** AND **$\overline{A}$**, **C** AND **$\overline{C}$**, **D** AND **$\overline{D}$** ALL APPEAR. >> THEY ARE **ELLIMINATED** FROM THE PRODUCT TERM; HENCE WE END UP WITH **B**.
2. SIMILARLY IN THE 4 CELL SUBCUBE **D**,**$\overline{D}$** AND **B**,**$\overline{B}$** ARE ELLIMINATED TO LEAVE US WITH **$\overline{A}$C**.
3. IN THE 2 CELL SUBCUBE, **B**, **$\overline{B}$** ARE ELLIMINATED TO LEAVE **A$\overline{C}$D**

THIS CAN BE UNDERSTOOD BY REMEMBERING THE FOLLOWING **BOOLEAN RULE**:

$$AX + A\overline{X} = A(\underbrace{X + \overline{X}}_{\text{ELLIMINATE}}) = A \,.\, 1 = A$$

<u>FOR EXAMPLE</u>:

$$AB\overline{C}D + A\overline{B}\,\overline{C}D = A\overline{C}D\,(B + \overline{B}) = \mathbf{A\overline{C}D}$$

$$\overline{A}\,\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BCD + \overline{A}BC\overline{D} =$$

$$= \overline{A}\,\overline{B}C(D + \overline{D}) + \overline{A}BC(D + \overline{D}) =$$

$$= \overline{A}\,\overline{B}C + \overline{A}BC =$$

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

IN GENERAL, A SUBCUBE OF $\mathbf{2^M}$ CELLS IN AN "**N**" VARIABLE K-MAP WILL HAVE "**M**" VARIABLES DIFFERING AND THE SUBCUBE CAN BE REPLACED BY **ONE** PRODUCT CONSISTING OF "**N – M**" VARIABLES WHICH REMAIN CONSTANT.

LAST EXAMPLE:

N = 4 >>> A,B,C,D

$2^M = 4$ >>> M = 2

\>>>> 4 – 2 = 2 VARIABLE PRODUCT

$= \overline{A}C$

EXAMPLES:

$$\text{(a) } F = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}\,\overline{B}C\overline{D} + \overline{A}\,\overline{B}CD$$

$$+\overline{A}B\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}C\overline{D}$$

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	1	1
$\overline{A}B$	1	0	0	0
AB	0	0	0	0
$A\overline{B}$	1	0	0	1

$\overline{A}\,\overline{B}$ $\overline{A}\,\overline{C}\,\overline{D}$ $\overline{B}\,\overline{D}$

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

3
SUBCUBES

(b)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	0	0
$\bar{A}B$	0	1	0	0
AB	0	0	1	0
$A\bar{B}$	0	0	0	0

$\bar{A}\bar{B}\bar{C}$ $\bar{A}\bar{C}D$ $ABCD$

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{C}D + ABCD$$

CONCLUDING NOTE: ALWAYS TRY TO OBTAIN THE REPRESENTATON THAT HAS THE FEWER NUMBER OF CUBES.

EXAMPLE:

1	1	0	1
0	1	1	1
0	1	1	0
0	0	0	0

4 SUBCUBES

1	1	0	1
0	1	1	1
0	1	1	0
0	0	0	0

4 SUBCUBES

BEST

1	1	0	1
0	1	1	1
0	1	1	0
0	0	0	0

3 SUBCUBES

FEWER SUBCUBES:

>> FEWER PRODUCT TERMS

>> FEWER GATES

>> MAXIMISE THE SIZE OF SUBCUBES

>> SMALLER PRODUCT TERMS

AT THIS STAGE YOU SHOULD BE ABLE TO MINIMISE A BOOLEAN EXPRESSION BY:

- **BOOLEAN** ALGEBRA
- KARNAUGH MAP **REDUCTION** TECHNIQUES

NUMBER SYSTEMS

IN ANY NUMBER SYSTEM, THE **POSITION** OF EACH OF THE DIGITS INDICATES THE **MAGNITUDE** OF THE QUANTITY REPRESENTED AND CAN BE ASSIGNED A **WEIGHT**.

THE VALUE OF THE NUMBER IS THE SUM OF THE DIGIT TIMES THEIR RESPECTIVE COLUMN WEIGHT.

EXAMPLE – DECIMAL NUMBERS

23 = **2** * 10 + **3** * 1 = 20 + 3
I.E. >> DIGIT **2** HAS A WEIGHT OF **10**
3 HAS A WEIGHT OF **1**
AS INDICATED BY THEIR RESPECTIVE POSITIONS

THE **BASE** OF A NUMBER SYSTEM IS THE NUMBER OF **DIFFERENT DIGITS** THAT CAN OCCUR IN **EACH POSITION**

DECIMAL SYSTEM = **BASE 10**

10 DIGITS >>> $0 \rightarrow 9$

E.G. $26_{10} = 26$

HEXADECIMAL SYSTEM = **BASE 16**

16 DIGITS >>> $0 \rightarrow 9$, $A \rightarrow F$

E.G. $3A_{16}$

BINARY NUMBERS

THE **BINARY** NUMBER SYSTEM HAS A BASE OF **2**, THE TWO BINARY DIGITS ARE **0** AND **1**

EACH **B**INARY DIGIT IS CALLED A **BIT**

THE POSITION OF A BIT DETERMINES ITS WEIGHT BUT NOW THE WEIGHT ASCENDS IN POWERS OF 2.

EXAMPLES:

DECIMAL: $11_{10} = 1 * 10^1 + 1*10^0$

BINARY

$101_2 = 1*2^2 + 0*2^1 + 1*2^0 =$
$= 4 \quad + \ 0 \quad + 1 = 5_{10}$

$1010_2 = 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0$
$= \ 8 \quad + \quad 0 \quad + \ 2 \quad + \ 0 = 10_{10}$

$11.011_2 = 1*2^1 + 1*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3}$
$= \ 2 \quad + \quad 1 \ + \ 0 \quad + \ \frac{1}{2^2} + \frac{1}{2^3} = 3\frac{3}{8}$

IN GENERAL:

$A_nA_{n-1}\ldots A_1A_0A_{-1}A_{-2}\ldots A_{-m}$

$= A_n*2^n + A_{n-1}*2^{n-1} + \ldots A_1*2^1 + A_0*2^0$

$+ A_{-1}*2^{-1} + A_{-2}*2^{-2} + \ldots + A_{-m}*2^{-m}$

WE FOLLOW THE ABOVE PROCEDURE WHEN WE WISH TO CONVERT FROM **BINARY** TO DECIMAL **FORM**

DECIMAL TO BINARY CONVERSION

TO CONVERT FROM DECIMAL TO BINARY WE REPEATEDLY **DIVIDE BY 2** THE **DECIMAL NUMBER**, AND **THE REMAINDERS** ARE THE **BITS** OF THE RESULTING **BINARY NUMBER**

EXAMPLES:

1. $\mathbf{19}_{10}$

19 : 2	•	
9 : 2	**1**	LSB
4 : 2	**1**	
2 : 2	**0**	
1 : 2	**0**	
0	**1**	MSB

READ UP

REMAINDER

$\mathbf{19}_{10} = \mathbf{10011}_2$

NOTE THE POSITION OF THE DECIMAL POINT, YOU **READ UP** FROM THE **MSB**

MSB = **MOST** SIGNIFICANT BIT
LSB = **LEAST** SIGNIFICANT BIT

2. **28_{10}**

28 : 2	•	
14 : 2	**0**	LSB
7 : 2	**0**	↑
3 : 2	**1**	↑
1 : 2	**1**	↑
0	**1**	MSB

28_{10} = 11100_2

DECIMAL FRACTIONS:

THE ABOVE DESCRIBED METHOD DOES NOT WORK FOR FRACTIONS. HERE WE REPEATEDLY **MULTIPLY** THE FRACTION BY **TWO** (UNTIL THE FRACTIONAL PRODUCT IS ZERO) AND THE **WHOLE NUMBER CARRYS** ARE THE BITS OF THE RESULTING BINARY NUMBER.

EXAMPLE:

$.4375_{10}$

READ DOWN ↓

.4375 * 2		•	
.8750 * 2	CARRY	**0**	MSB
.7500 * 2	CARRY	**1**	
.5000 * 2	CARRY	**1**	
.0000	CARRY	**1**	LSB

$.4375_{10}$ = $.0111_2$

AGAIN, NOTE THE DECIMAL POINT POSITION AND **READ DOWN**.

MIXED DECIMAL NUMBERS

THESE MUST BE SPLIT INTO THEIR **WHOLE** AND **FRACTIONAL** PARTS, EACH PART **CONVERTED SEPARATELY** AND THE TWO PARTS ARE THEN **ADDED**.

EXAMPLE

$\mathbf{13.75}_{10} = \mathbf{13}_{10} + \mathbf{.75}_{10}$

13 : 2	•	.75 * 2	•
6 : 2	**1**	.50 * 2	**1**
3 : 2	**0**	.00	**1**
1 : 2	**1**		
0	**1**		

$\mathbf{13.75}_{10} = \mathbf{1101.11}_{2}$

BINARY NUMBERS

BINARY				DECIMAL
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15
2^3	2^2	2^1	2^0	
MSB			LSB	

BINARY ADDITION

SIMILAR TO DECIMAL ADDITION BUT SIMPLER AS ONLY **0**'s AND **1**'s ARE ALLOWED. THE FOUR BASIC RULES ARE:

$$\mathbf{0 + 0 = 0}$$
$$\mathbf{0 + 1 = 1}$$
$$\mathbf{1 + 0 = 1}$$
$$\mathbf{1 + 1 = 10}_2$$

>> i.e. **0** WITH A CARRY OF **1**

EXAMPLES:

(a)

$$\begin{array}{r} \mathbf{11}_2 + \\ \mathbf{11}_2 \\ \hline \mathbf{110}_2 \end{array} \quad >> \quad \begin{array}{r} \mathbf{3}_{10} + \\ \mathbf{3}_{10} \\ \hline \mathbf{6}_{10} \end{array}$$

i.e. **1+1+1 = 1** WITH A CARRY **1**

(b)

$$\begin{array}{r} \mathbf{1111} + \\ \mathbf{10100} \\ \hline \mathbf{100011}_2 \end{array} \quad >> \quad \begin{array}{r} \mathbf{15} + \\ \mathbf{20} \\ \hline \mathbf{35}_{10} \end{array}$$

(c)

$$\begin{array}{r} \mathbf{11.01} + \\ \mathbf{101.11} \\ \hline \mathbf{1001.00}_2 \end{array} \quad >> \quad \begin{array}{r} \mathbf{3.25} + \\ \mathbf{5.75} \\ \hline \mathbf{9.00}_{\mathbf{10}} \end{array}$$

LOGIC CIRCUITS

<u>THE HALF ADDER:</u>

TO ADD 2 **LEAST** SIGNIFICANT BITS (LSB) WE DO NOT NEED A CARRY INPUT FROM A PREVIOUS STAGE.

>> WE ONLY NEED A **HALF ADDER**

THIS WILL HAVE TWO INPUTS $\mathbf{A_0}$, $\mathbf{B_0}$ AND TWO OUTPUTS $\mathbf{S_0}$ AND $\mathbf{C_0}$

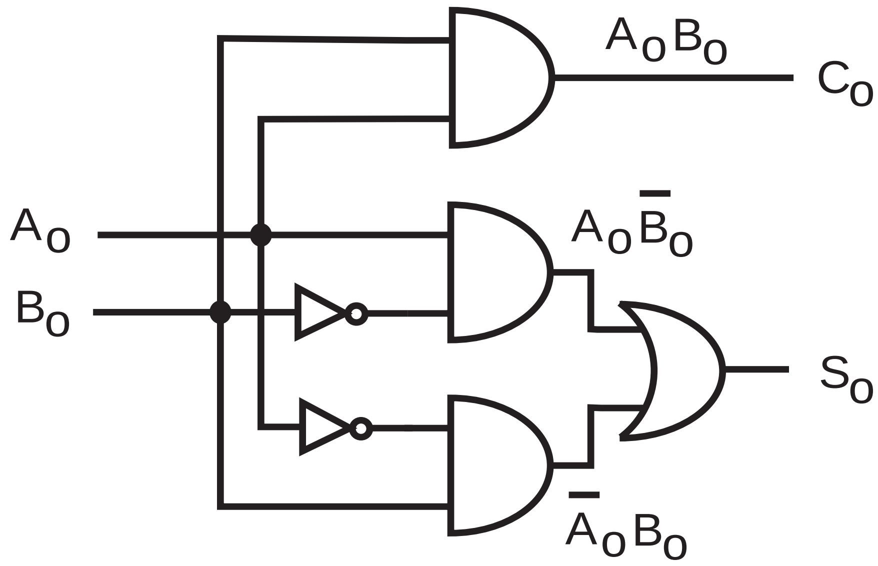
A_0	B_0	S_0	C_0
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

$\mathbf{S_0}$: **SUM OUT**
$\mathbf{C_0}$: **CARRY OUT**

THE **LOGIC** EQUATIONS ARE:

$$\mathbf{S_0} = \overline{A}_0 B_0 + A_0 \overline{B}_0 = A \oplus B \quad > \quad \text{(XOR Gate)}$$
$$\mathbf{C_0} = A_0 B_0 \quad > \quad \text{(AND Gate)}$$

THE **LOGIC DIAGRAM** CAN ALSO BE EXPRESSED AS FOLLOWS:



<u>THE FULL ADDER:</u>

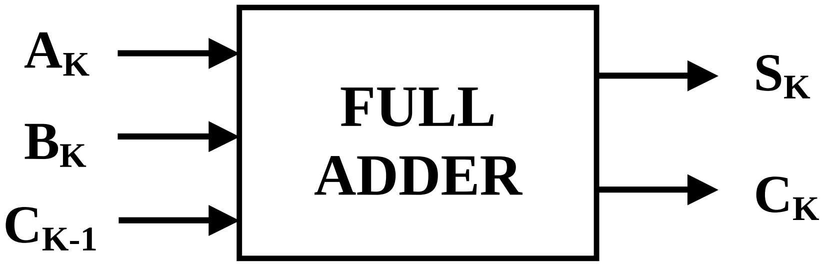
FOR ALL OTHER BITS (EXCEPT THE LSB) A HALF ADDER **WILL NOT** SUFFICE BECAUSE THERE **MAY BE A CARRY INPUT** FROM A PREVIOUS STAGE.

A FULL ADDER HAS **3 INPUTS**: $\mathbf{A_K}$, $\mathbf{B_K}$, $\mathbf{C_{K-1}}$

AND **2 OUTPUTS**: $\mathbf{S_K}$, $\mathbf{C_K}$

$\mathbf{C_{K-1}}$ **=** CARRY IN FROM THE **PREVIOUS** STAGE

$\mathbf{C_k}$ = CARRY OUT TO THE **NEXT** STAGE



A_k	B_k	C_{k-1}	S_k	C_k
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**

THE **LOGIC** EQUATIONS ARE:

$$\mathbf{S_K} = \overline{A}_K\overline{B}_K C_{K-1} + \overline{A}_K B_K \overline{C}_{K-1} + A_K \overline{B}_K \overline{C}_{k-1} + A_K B_K C_{K-1}$$

$$\mathbf{C_K} = \overline{A}_K B_K C_{K-1} + A_K \overline{B}_K C_{K-1} + A_K B_K \overline{C}_{k-1} + A_K B_K C_{K-1}$$

K-MAP FOR $\mathbf{S_K}$**:**

C_{K-1} \ $A_K B_K$	0 0	0 1	1 1	1 0
0	0	1	0	1
1	1	0	1	0

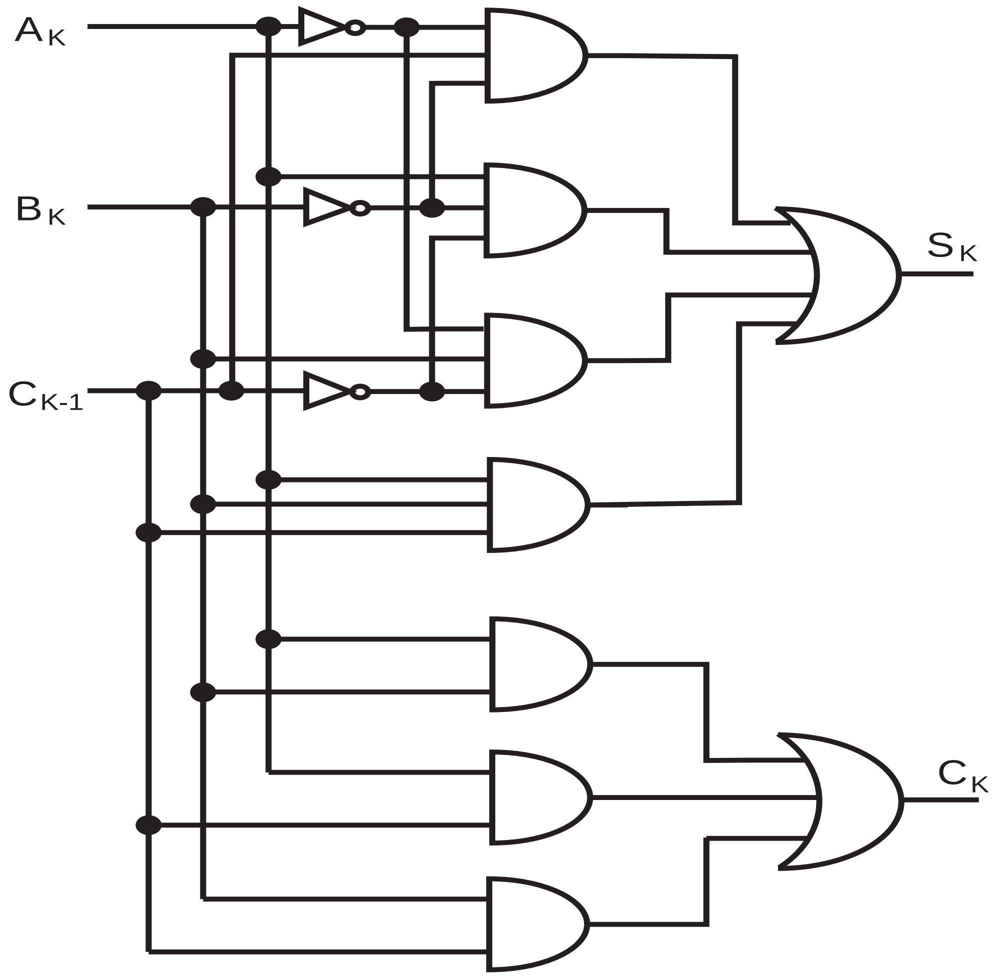
NO SIMPLIFICATION POSSIBLE

K-MAP FOR **C_K:**

C_{K-1} \ $A_K B_K$	0 0	0 1	1 1	1 0
0	0	0	1	0
1	0	1	1	1

$$C_K = B_K C_{K-1} + A_K B_K + A_K C_{K-1}$$

THE **LOGIC** DIAGRAM FOR A **FULL ADDER** IS:

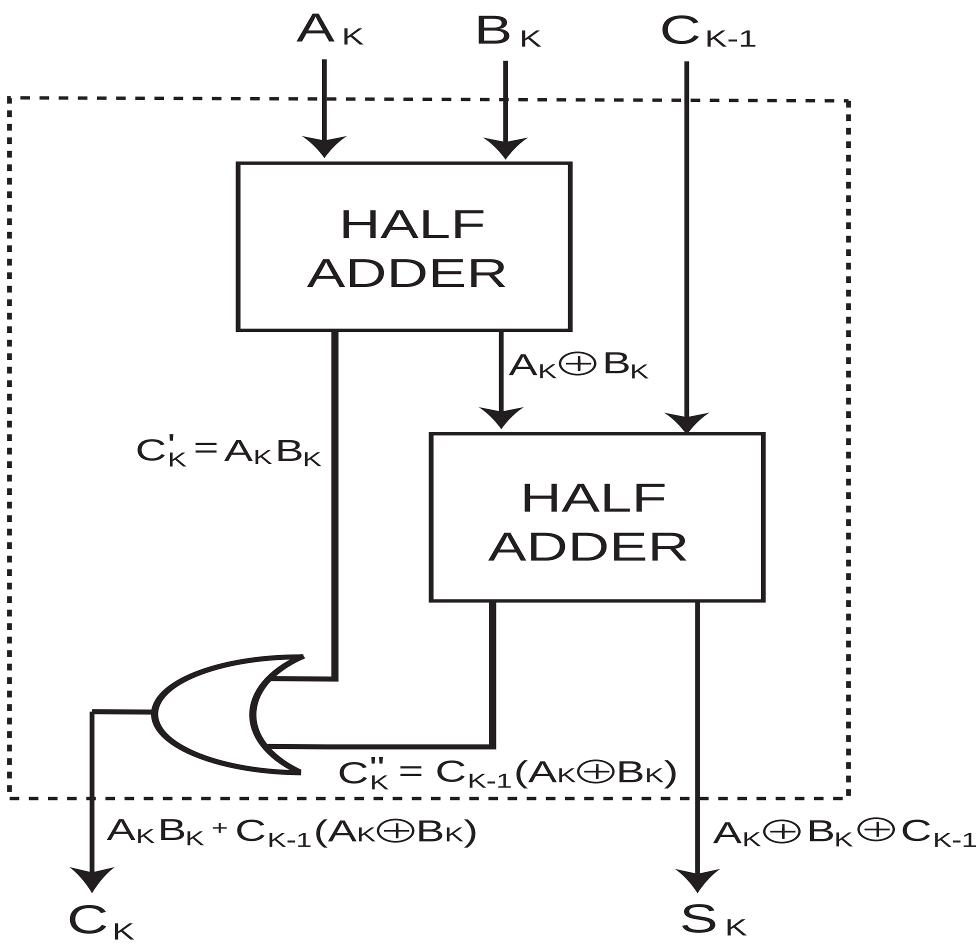


FULL ADDER USING HALF ADDERS

ONE HLF ADDER ADDS $\mathbf{A_K}$ TO $\mathbf{B_K}$ TO GIVE AN INTERMEDIATE SUM $\mathbf{S'_K}$ AND CARRY $\mathbf{C'_K}$

ANOTHER HALF ADDER ADDS $\mathbf{S'_K}$ AND $\mathbf{C_{K-1}}$ TO GIVE THE FINAL SUM $\mathbf{S_K}$ AND ANOTHER INTERMEDIATE CARRY $\mathbf{C''_K}$

THERE WILL BE A FINAL CARRY IF EITHER $\mathbf{C'_K}$ OR $\mathbf{C''_K}$ ARE “**1**”



REMEMBER THE BOOLEAN OPERATION FOR **HALF ADDER**:

$$S'_K = A_K \oplus B_K$$
$$C'_K = A_K B_K$$

FOR FULL ADDER:

$$S_K = A_K \oplus B_K \oplus C_{K-1}$$
$$C_K = A_K B_K + C_{K-1}(A_K \oplus B_K)$$

<u>PROOF:</u>

$$\mathbf{S_K} = \overline{A}_K\overline{B}_K C_{K-1} + \overline{A}_K B_K \overline{C}_{K-1} + A_K\overline{B}_K\overline{C}_{k-1}$$
$$+ A_K B_K C_{K-1} = C_{K-1}(\overline{A}_K\overline{B}_K + A_K B_K) +$$
$$+ \overline{C}_{K-1}(\overline{A}_K B_K + A_K\overline{B}_K) = C_{K-1}(\overline{A_K \oplus B_K}) +$$
$$+ \overline{C}_{K-1}(A_K \oplus B_K) = \mathbf{A_K \oplus B_K \oplus C_{K-1}}$$

$$\mathbf{C_K =} \overline{A}_K B_K C_{K-1} + A_K\overline{B}_K C_{K-1} + A_K B_K \overline{C}_{k-1}$$
$$+ A_K B_K C_{K-1} =$$
$$= A_K B_K(C_{K-1} + \overline{C}_{K-1}) + C_{K-1}(\overline{A}_K B_K + A_K\overline{B}_K) =$$
$$= \mathbf{A_K B_K + C_{K-1}(A_K \oplus B_K)}$$

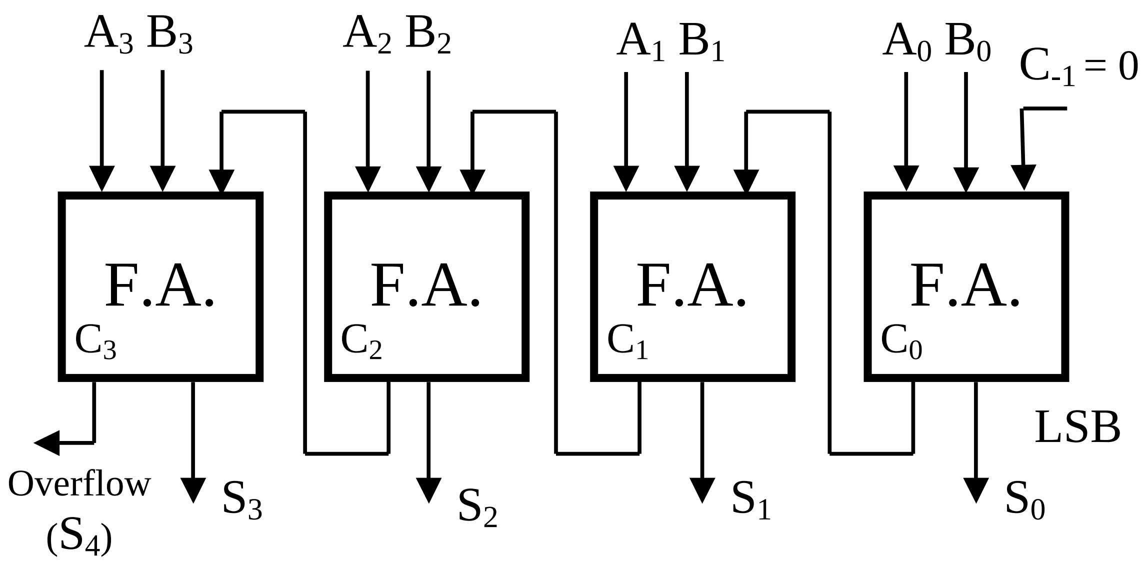
THE PARALLEL ADDER:

ALSO CALLED **RIPPLE CARRY** ADDER USED TO ADD **TWO N-BIT** NUMBERS.

IT CONSISTS OF N FULL ADDERS WHERE THE CARRY OUTPUT OF EACH STAGE IS THE CARRY IN OF THE NEXT STAGE.

EXAMPLE: 4-BIT PARALLEL ADDER

ADD $\mathbf{A_3A_2A_1A_0}$ AND $\mathbf{B_3B_2B_1B_0}$



$C_3 = 1$ >>> OVERFLOW

E.G. A = **1010** (10_{10})

B = **1001** (9_{10})

Overflow ← (**1**)0011 → 3_{10}

BINARY NEGATIVE NUMBERS

IN THE DECIMAL NUMBER SYSTEM **NEGATIVE** NUMBERS ARE DENOTED BY A **MINUS** SIGN. SINCE **ONLY** “**0**”s AND “**1**”s **ARE ALLOWED BY BOOLEAN ALGEBRA** WE MUST USE THESE DIGITS TO REPRESENT POSITIVE AND NEGATIVE NUMBERS.

THERE ARE 3 DIFFERENT FORMS TO REPRESENT POSITIVE AND NEGATIVE NUMBERS IN BINARY

1. SIGNED MAGNITUDE FORM:

- THE MAGNITUDE OF THE NUMBER IS IN NORMAL BINARY FORM
- THE EXTRA LEADING BIT IS USED FOR THE SIGN:

  “0” >>> POSITIVE
  “1” >>> NEGATIVE

- THIS REPRESENTATION IS ALSO CALLED **SIGN PLUS MAGNITUDE**

EXAMPLES:

+3 = 0 ⋮ 0011
+14 = 0 ⋮ 1110
+5 = 0 ⋮ 0101

SIGN MAG.

-3 = 1 ⋮ 0011
-14 = 1 ⋮ 1111
-5 = 1 ⋮ 0101

SIGN MAG.

NOTE THE ZERO AMBIGUITY:

ZERO = 0 ⋮ 0000 OR 1 ⋮ 0000

2. ONE'S COMPLEMENT FORM:

FOR POSITIVE NUMBERS THE "1"s COMPLEMENT IS FORMED BY PLACING A LEADING "0" TO THE LEFT OF THE MAGNITUDE.

E.G. $+12_{10} = 0 \vdots 1100_2$

$+2_{10} = 0 \vdots 0010_2$

FOR NEGATIVE NUMBERS WE INVERT (COMPLEMENT) EACH BIT OF THE CORRESPONDING POSITIVE NUMBER.

E.G. $-12_{10} = 1 \vdots 0011_2$

$-2_{10} = 1 \vdots 1101_2$

IF WE HAVE A 5-BIT **1's COMPLEMENT** NUMBER THEN:

- FIRST BIT (MSB) >>> SIGN
- 4 OTHER BITS >>> MAGNITUDE

0 1111 $+15_{10}$

0 1110 $+14_{10}$

0 1101 $+13_{10}$

⋮

0 0001 $+1_{10}$

0 0000 → ZERO

1 1111 → AMBIGUITY

1 1110 -1_{10}

1 1101 -2_{10}

⋮

1 0001 -14_{10}

1 0000 -15_{10}

ONE'S COMPLEMENT ADDITION

(a) ADDITION OF **2 POSITIVE** NUMBERS

E.G.

+7	00111	+11	01011
+5	00101	+4	00100
+12	01100	+15	01111

(b) ADDITION OF **POSITIVE** AND **NEGATIVE** NUMBERS

E.G.

+3	0 0011
-12	1 0011
-9	1 0110

NEGATIVE NUMBER

= 3 + (-12)

1 0110 >>> - [$\overline{0110}$] = -1001 = -9

+9	0 1001
-4	1 1011
+5	(**1**)0 0100
	1

+5 = 0 0101

ADD END AROUND CARRY

*** DON'T FORGET END AROUND CARRY

+12	0 1100	
-12	1 0011	
0	1 1111	= 0

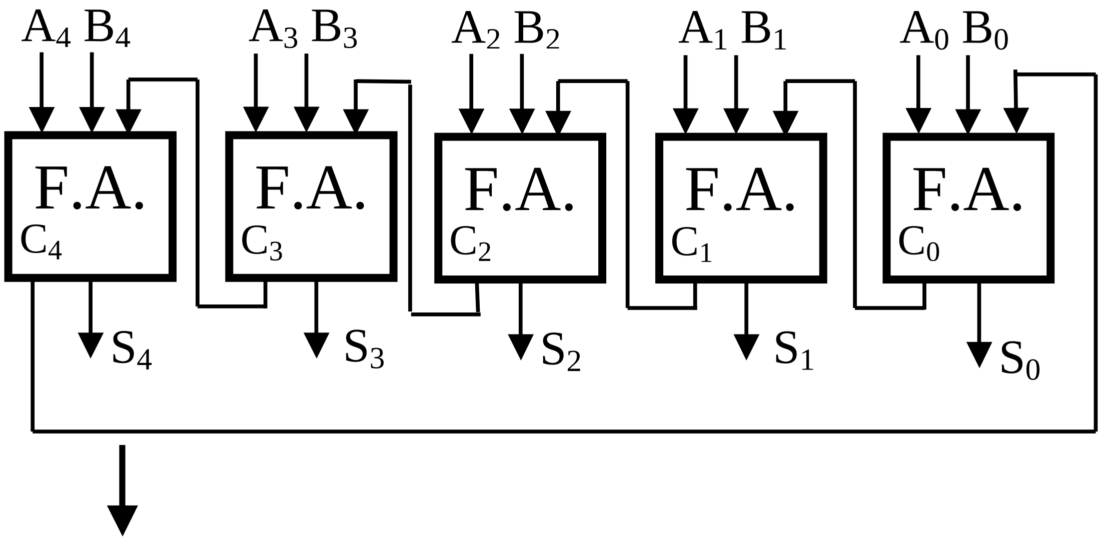
(c) ADD **2 NEGATIVE** NUMBERS

- **ALWAYS** WILL BE AN END AROUND CARRY

-3	1 1100	-4	1 1011
-11	1 0100	-7	1 1000
-14	(**1**)1 0000	-11	(**1**)1 0011
	1		**1**
	1 0001		1 0100

1 0001 >> $-[\overline{0001}] = -14$ 1 0100 >> $-[\overline{0100}] = -11$

- THUS, A 5-BIT **1's COMPLEMENT** ADDER CAN BE BUILT FROM **5 FULL ADDERS**



END AROUND CARRY

RANGE: **+15 ÷ -15**

THE **MOST** SIGNIFICANT CARRY OUTPUT IS **ALWAYS** CONNECTED TO THE **LEAST** SIGNIFICANT CARRY INPUT

THERE IS AN END AROUND CARRY **IF C_4=1**, OTHERWISE THE **LEAST** SIGNIFICANT CARRY INPUT IS **0**.

3. TWO'S COMPLEMENT FORM:

MOST POPULAR BINARY FORM

SIMILAR TO **1's COMPLEMENT** BUT FOR **NEGATIVE** NUMBERS **ADD "1"** TO THE **1's COMNPLEMENT** RESULT

E.G. +5 = 0 0101
-5 = 1 1010 **1's COMPLEMENT**
1
-5 = 1 1011 **2's COMPLEMENT**

+0 = 0 0000
1 1111 **INVERT**
1 **ADD 1**
(1) 0 0000 = -0

***** UNAMBIGOUS ZERO !**

+16 >>> **NOT POSSIBLE**
-16 >>> **1 0000**

(a) ADDING TWO POSITIVE NUMBERS

STRAIGHTFORWARD PROVIDING THERE IS NO OVERFLOW

E.G. 5-BIT NUMBERS

+5	0 0101		+12	0 1100
+9	0 1001		+3	0 0011
+14	0 1110		+15	0 1111

(b) ADDING TWO NEGATIVE NUMBERS

-6	1 1010		-12	1 0100
-9	1 0111		-4	1 1100
-15	(1) 1 0001		-16	(1) 1 0000
	IGNORE			IGNORE

(c) ADDING POSITIVE AND NEGATIVE NUMBERS

-9	1 0111		+12	0 1100
+3	0 0011		-4	1 1100
-6	1 1010		+8	(**1**) 0 1000

+15	0 1111		-8	1 1000
-1	1 1111		+8	0 1000
+14	(**1**) 0 1110		0	(**1**) 0 0000

ALWAYS IGNORE THE MOST SIGNIFICANT CARRY

- **ADVANTAGE** OVER **1's COMPLEMENT** IN THAT **NO END AROUND CARRY NEEDED**

N BITS >> **2^N** NUMBERS MAY BE REPRESENTED (**INCLUDING ZERO**)

4 BITS >> **16** NUMBERS

RANGE: **-8 ÷ +7**

OVERFLOW AND UNDERFLOW

REMEMBER THE NUMBER OF DIGITS **IS RESTRICTED**

WHEN ADDING NUMBERS OF **OPOSITE** SIGN THE RESULT CAN **NEVER** EXCEED THE **PERMITED RANGE**

BUT WHEN **TWO POSITIVE** NUMBERS ARE ADDED THE RESULT MAY BE **TOO LARGE** i.e. **OVERFLOW** OCCURS

E.G. 0 1101 +13

0 0101 +5

1 0010 -14 <<< **WRONG**

OVERFLOW

A 0 1000 +8

B 0 1000 +8

S 1 0000 -16 <<< **WRONG**

OVERFLOW

IF EACH NUMBER HAS **N+1** BITS

OVERFLOW = $\overline{A}_N\overline{B}_N S_N$

OVERFLOW INDICATES WE ARE **OUTSIDE THE RANGE** AND WE **CANNOT** REPRESENT THE RESULT IN OUR RESTRICTED NUMBER OF BITS

UNDERFLOW

UNDERFLOW MAY OCCUR WHEN **TWO NEGATIVE** NUMBERS ARE ADDED THE RESULT IS **OUTSIDE THE RANGE**

E.G.	1 0011	-13	
	1 1000	-8	
	(1) 0 1011	+11	<<< **WRONG**

A	1 0000	-16	
B	1 1111	-1	
S	(1) 0 1111	+15	<<< **WRONG**

SHOULD BE **–17** BUT THIS **IS OUTSIDE** THE PERMITED RANGE

IF EACH NUMBER HAS **N+1** BITS

UNDERFLOW = $A_N B_N \overline{S}_N$

SUBTRACTION

DIRECT SUBTRACTION (i.e. NOT USING SPECIAL REPRESENTATIONS TO GENERATE NEGATIVE NUMBERS) CAN BE PERFORMED BY DETERMINING THE **TRUTH TABLE** AND THEN DESIGNING A **LOGIC DIAGRAM**

THE FOUR BASIC RULES ARE:

$0 - 0 = 0$
$1 - 1 = 0$
$1 - 0 = 1$
$0 - 1 = 10_2 - 1 = 1$ WITH A BORROW OF **1**

A	B	A - B	BORROW B_0
0	0	**0**	**0**
0	1	**1**	**1**
1	0	**1**	**0**
1	1	**0**	**0**

HALF SUBTRACTOR

DIFFERENCE = $\mathbf{A - B = A \oplus B}$

BORROW = $\mathbf{B_0 = \overline{A}B}$

FULL SUBTRACTOR:

A	B	BORROW IN B_{IN}	DIFF $A-(B+B_{IN})$	BORROW OUT B_{OUT}
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**1**	**1**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**0**
1	1	1	**1**	**1**

USING K-MAPS IT CAN BE SHOWN THAT:

$$\mathbf{B_{OUT}} = \overline{A}B_{IN} + \overline{A}B + BB_{IN}$$

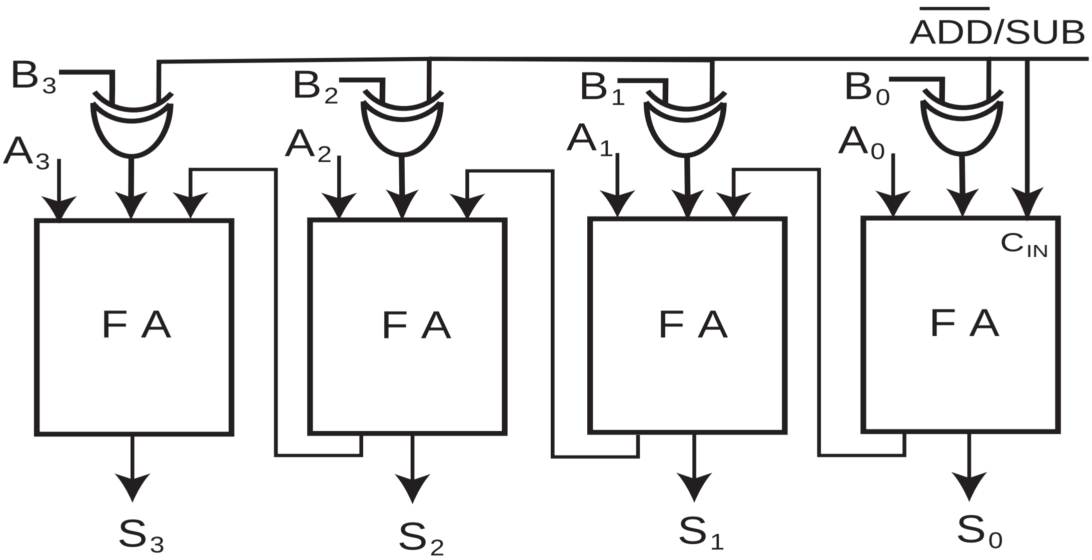
$$\mathbf{DIFF} = \overline{A}\,\overline{B}B_{IN} + \overline{A}B\overline{B}_{IN} + ABB_{IN} + A\overline{B}\,\overline{B}_{IN}$$

NOTE: SUBTRACTION MAY BE DONE MORE ECONOMICALLY BY REPRESENTING THE NEGATIVE NUMBERS USING **2's COMPLEMENT** FORM

E.G. 9 – 6 = 9 + (-6)

TWO's COMPLEMENT ADDER/SUBTRACTOR

IT IS POSSIBLE TO BUILD A CIRCUIT WHICH **ADDS AND SUBTRACTS** USING ONLY **FULL ADDERS** AND SOME ADDITIONAL CIRCUITRY TO GENERATE THE **2's COMPLEMENT** WHEN WISH TO DO SUBTRACTION



$\overline{\text{ADD}}$/SUB = **0** >> **ADDER**

= **1** >> **SUBTRACTOR**

- **XOR** GATES ACT AS "**TRUE/COMPLEMENT**" GATES
- INVERT **B** WHEN **$\overline{\text{ADD}}$/SUB** =1
- "**1**" IS ADDED TO THE **LSB** TO GENERATE THE **2's COMPLEMENT** OF **B**