Standard Forms

- All Boolean functions, regardless of their form, can be converted into either of two standard forms:
  - The Sum-of- Product (SOP) form.
  - The Product -of- Sum (POS) form.
- Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

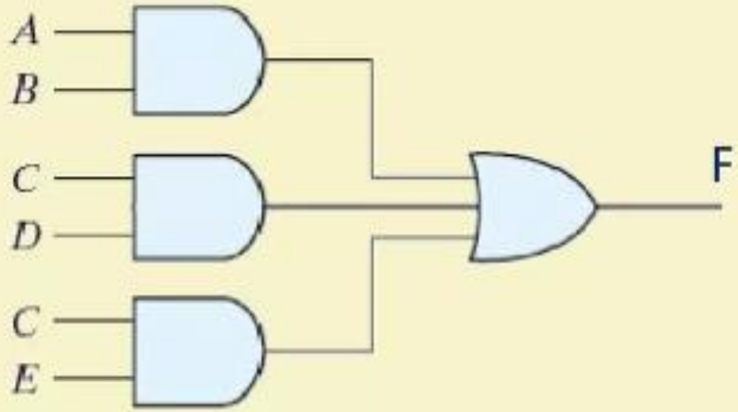
Sum of Product (SOP) Form

- The sum of products is a Boolean expression containing AND terms, called product terms, with one or more literals each. The sum denotes the ORing of these terms.
- Example : F = AB + CD + CE

  F = A + B'C

  F = AB'C' + A'B'C + A'BC

The logic diagram of a sum-of-products expression consists of a group of AND gates followed by a single OR gate.



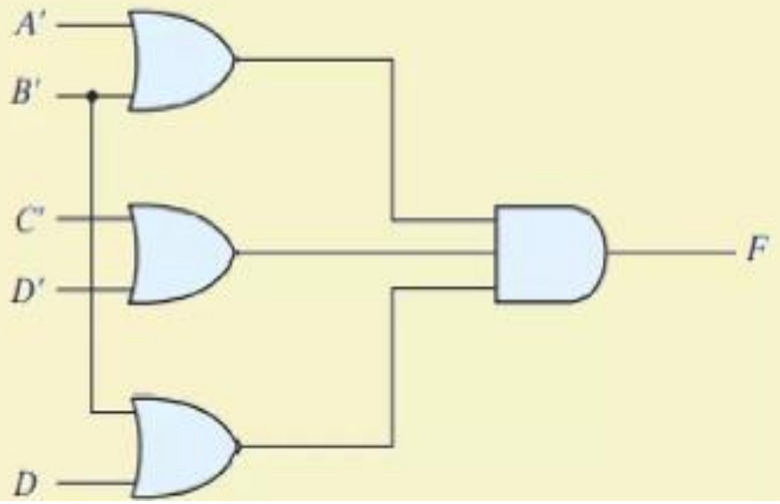
Product of Sums (POS) Form

- A product of sums is a Boolean expression containing OR terms, called sum terms. Each term may have any number of literals. The product denotes the ANDing of these terms.
- Example : $F = (A' + B')(C' + D')(B' + D)$

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

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

The logic diagram of a product-of-sums expression consists of a group of OR gates followed by a single AND gate.



Minterms and Maxterms

- For a Boolean function, a ***literal*** is defined as a variable in uncomplemented or complemented form.
  - Example: A, A′, B, B′, etc.
- Consider an n-variable Boolean function $f(x_1, x_2, \ldots, x_n)$.
  - A product term (that is, an AND operation) of all the n literals is called a ***minterm***.
  - A sum term (that is, an OR operation) of all the n literals is called a ***maxterm***.
- Consider a 3-variable function f (A, B, C).
  - Examples of minterm: A′.B′.C′, A.B′.C, A.B.C, etc.
  - Examples of maxterm: (A + B′ + C′), (A′ + B′ + C′), (A + B + C), etc.

Minterms and Maxterms

- Boolean function of n variables will have 2^n minterms, corresponding to each combination of variables.
- Boolean function of n variables will have 2^n maxterms, corresponding to each combination of variables.
- For a given row of the truth table, the minterm is formed by including variable x if $x = 1$ and by including x' if $x = 0$.
- For a given row of the truth table, the maxterm is formed by including variable x if $x = 0$ and by including x' if $x = 1$.

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
A	B	C	Term	Designation	Term	Designation
0	0	0	$A'.B'.C'$	m_0	$A + B + C$	M_0
0	0	1	$A'.B'.C$	m_1	$A + B + C'$	M_1
0	1	0	$A'.B.C'$	m_2	$A + B' + C$	M_2
0	1	1	$A'.B.C$	m_3	$A + B' + C'$	M_3
1	0	0	$A.B'.C'$	m_4	$A' + B + C$	M_4
1	0	1	$A.B'.C$	m_5	$A' + B + C'$	M_5
1	1	0	$A.B.C'$	m_6	$A' + B' + C$	M_6
1	1	1	$A.B.C$	m_7	$A' + B' + C'$	M_7

Minterms and Maxterms

- Properties of minterms and maxterms:
  - A minterm assumes value 1 for exactly one combination of variables.
  - A maxterm assumes the value 0 for exactly one combination of variables.
- **Example:** Minterm : A'B'C' assumes value 1 only for A=0, B= 0 and C=0.

  Maxterm: A' + B + C' assumes value 0 only for A= 1, B=0 and C= 1.
- Each minterm is the complement of corresponding maxterm and vice -versa .
- **Example :** Consider maxterm, $M_0 = A + B + C$

$$M_0' = (A + B + C)'$$
$$M_0' = A'B'C'$$
$$M_0' = m_0$$

$$F = A'.B'.C' + A'.B'.C + A'.B.C' + A.B.C' + A.B.C$$

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

Truth Table

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

Converting Standard SOP to Standard POS (example)

- Convert the SOP expression to an equivalent POS expression:

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

  - The evaluation is as follows:

$$000 + 010 + 011 + 101 + 111$$

  - There are 8 possible combinations. The SOP expression contains five of these, so the POS must contain the other 3 which are: 001, 100, and 110.

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

The Karnaugh Map

Karnaugh Map (K-Map)

- Karnaugh Mapping is used to minimize the number of logic gates that are required in a digital circuit.
- This will replace Boolean reduction when the circuit is large.
- Write the Boolean equation in a SOP form first and then place each term on a map.

Karnaugh Map (K-Map)

- The map is made up of a table of every possible SOP using the number of variables that are being used.
- If 2 variables are used then a 2X2 map is used
- If 3 variables are used then a 4X2 map is used
- If 4 variables are used then a 4X4 map is used
- If 5 Variables are used then a 8X4 map is used

Karnaugh Map (K-Map)

Grouping the 1s

You can group 1s on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and to minimize the number of groups.

1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells is the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

2 Variables Karnaugh Map

	$\bar{B}$	B
$\bar{A}$	0	1
A	2	3

Notice that the map is going false to true, left to right and top to bottom

The upper right hand cell is $\bar{A}\,\bar{B}$ if $X = \bar{A}\,\bar{B}$ then put an X in that cell

	$\bar{B}$	B
$\bar{A}$	1	
A		

This show the expression true when A = 0 and B = 0

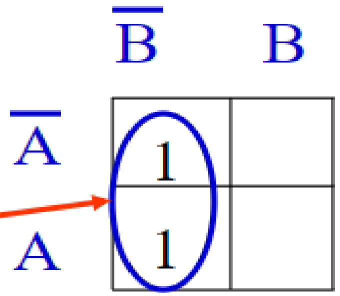
2 Variables Karnaugh Map

If $X = \overline{A}\,\overline{B} + A\overline{B}$ then put an X in both of these cells

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

From Boolean reduction we know that $\overline{A}\,\overline{B} + A\,\overline{B} = \overline{B}$

From the Karnaugh map we can circle adjacent cell and find that $X = \overline{B}$



3 Variables Karnaugh Map

Gray Code		0 $\overline{C}$	1 C
00	$\overline{A}\,\overline{B}$	0	1
01	$\overline{A}\,B$	2	3
11	$A\,B$	6	7
10	$A\,\overline{B}$	4	5

3 Variables Karnaugh Map (cont'd)

$$X = A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + AB\bar{C}$$

Gray Code		0	1
		$\bar{C}$	C
00	$\bar{A}\bar{B}$	1	1
01	$\bar{A}B$		
11	AB		
10	$A\bar{B}$	1	1

Each 3 variable term is one cell on a 4 X 2 Karnaugh map

3 Variables Karnaugh Map (cont'd)

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

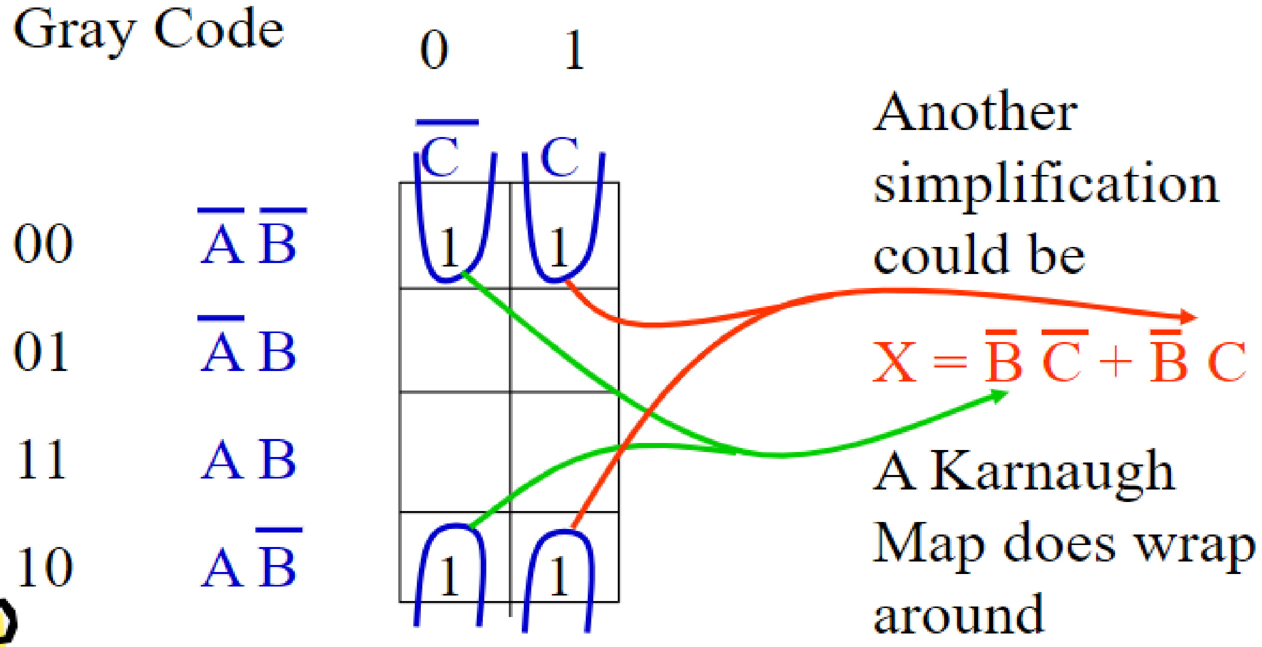
Gray Code		0	1
		$\overline{C}$	C
00	$\overline{A}\,\overline{B}$	1	1
01	$\overline{A}\,B$		
11	$A\,B$		
10	$A\,\overline{B}$	1	1

One simplification could be

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

3 Variables Karnaugh Map (cont'd)

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



3 Variables Karnaugh Map (cont'd)

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

Gray Code		0 $\overline{C}$	1 C
00	$\overline{A}\,\overline{B}$	1	1
01	$\overline{A}B$		
11	AB		
10	$A\overline{B}$	1	1

The Best simplification would be

$X = \overline{B}$

On a 3 Variables Karnaugh Map

- One cell requires 3 Variables
- Two adjacent cells require 2 variables
- Four adjacent cells require 1 variable
- Eight adjacent cells is a 1

4 Variables Karnaugh Map

Gray Code		0 0	0 1	1 1	1 0
		$\bar{C}\,\bar{D}$	$\bar{C}\,D$	$C\,D$	$C\,\bar{D}$
00	$\bar{A}\,\bar{B}$	0	1	3	2
01	$\bar{A}\,B$	4	5	7	6
11	$A\,B$	12	13	15	14
10	$A\,\bar{B}$	8	9	11	10

Simplify :

$$X = \bar{A}\,\bar{B}\,C\,D + \bar{A}\,B\,C\,D + \bar{A}\,B\,C\,\bar{D} + A\,B\,C\,D + A\,B\,\bar{C}\,D + A\,\bar{B}\,C\,D$$

Gray Code		0 0	0 1	1 1	1 0
		$\bar{C}\,\bar{D}$	$\bar{C}\,D$	$C\,D$	$C\,\bar{D}$
00	$\bar{A}\,\bar{B}$			1	
01	$\bar{A}\,B$			1	1
11	$A\,B$		1	1	
10	$A\,\bar{B}$			1	

$$X = ABD + \bar{A}BC + CD$$

Now try it with Boolean reductions

On a 4 Variables Karnaugh map

- One Cell requires 4 variables
- Two adjacent cells require 3 variables
- Four adjacent cells require 2 variables
- Eight adjacent cells require 1 variable
- Sixteen adjacent cells give a 1 or true

1.2.3 Disjunctive and Conjunctive Normal Form

❐ Experiment 1: Controlling an electric network

A current network must be optically controlled to avoid overloading by built-in heating units. The three sets A, B and C have the rated powers 4 kW, 6 kW and 8 kW. If the power taken from the network exceeds 11 kW a yellow warning lamp should light up, if it exceeds 13 kW a red warning lamp should light up. A green lamp signals safe operation.

❐ Experiment 1: Checking an electric network

4 kW A	6 kW B	8 kW C	P [kW]	Warning lamp		
				green	yellow	red
0	0	0	*0*	*1*	*0*	*0*
0	0	1	*8*	*1*	*0*	*0*
0	1	0	*6*	*1*	*0*	*0*
0	1	1	*14*	*0*	*1*	*1*
1	0	0	*4*	*1*	*0*	*0*
1	0	1	*12*	*0*	*1*	*0*
1	1	0	*10*	*1*	*0*	*0*
1	1	1	*18*	*0*	*1*	*1*

Table 1.2.3.1 Value table

Green (conjunctive normal form):

$$\begin{aligned} X &= (A \vee \bar{B} \vee \bar{C}) \wedge (\bar{A} \vee B \vee \bar{C}) \wedge (\bar{A} \vee \bar{B} \vee \bar{C}) \\ &= (A\bar{A} \vee AB \vee A\bar{C} \vee \bar{A}\bar{B} \vee \bar{B}B \vee \bar{B}\bar{C} \vee \bar{A}\bar{C} \vee B\bar{C} \vee \bar{C}) \wedge (\bar{A} \vee \bar{B} \vee \bar{C}) \\ &= [\bar{C}\,(A \vee \bar{B} \vee \bar{A} \vee B \vee 1) \vee A\,B \vee \bar{A}\,\bar{B}] \wedge (\bar{A} \vee \bar{B} \vee \bar{C}) \\ &= (\bar{C} \vee A\,B \vee \bar{A}\,\bar{B}) \wedge (\bar{A} \vee \bar{B} \vee \bar{C}) \\ &= \bar{A}\,\bar{C} \vee \bar{B}\,\bar{C} \vee \bar{C} \vee A\,\bar{A}\,B \vee A\,B\,\bar{B} \vee A\,B\,\bar{C} \vee \bar{A}\,\bar{B} \vee \bar{A}\,\bar{B} \vee \bar{A}\,\bar{B}\,\bar{C} \\ &= \bar{C}\,(\bar{A} \vee \bar{B} \vee 1 \vee A\,B \vee \bar{A}\,\bar{B}) \vee \bar{A}\,\bar{B} \end{aligned}$$

Green (simplified): $\mathbf{X = \bar{C} \vee \bar{A}\,\bar{B}}$

Yellow (disjunctive normal form):

$$\begin{aligned} Y &= \bar{A}\,B\,C \vee A\,\bar{B}\,C \vee A\,B\,C \\ &= \bar{A}\,B\,C \vee A\,\bar{B}\,C \vee A\,B\,C \vee A\,B\,C \\ &= B\,C\,(\bar{A} \vee A) \vee A\,C\,(\bar{B} \vee B) \end{aligned}$$

Yellow (simplified): $\mathbf{Y = B\,C \vee A\,C}$

Red (disjunctive normal form):

$$\begin{aligned} Z &= \bar{A}\,B\,C \vee A\,B\,C \\ &= B\,C\,(\bar{A} \vee A) \end{aligned}$$

Red (simplified): $\mathbf{Z = B\,C}$

1 green → using (CNF) (POS)

$F = (\ +\ )\cdot(\ +\ )$

$0 \to A$, $1 \to \bar{A}$

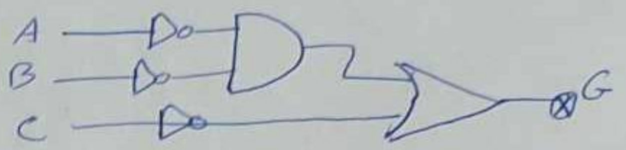
$G = (A+\bar{B}+\bar{C})\cdot(\bar{A}+B+\bar{C})\cdot(\bar{A}+\bar{B}+\bar{C})$

$= (A+\bar{B}+\bar{C})\cdot(\bar{A}+\bar{C}) + B\cdot\bar{B}$

$= \bar{C} + (A+\bar{B})\cdot\bar{A}$

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

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



2 Yellow → using (DNF) (SOP)

$F = (\ \cdot\ ) + (\ \cdot\ )$

$1 \to A$, $0 \to \bar{A}$

$Y = (\bar{A}BC) + (A\bar{B}C) + (ABC)$

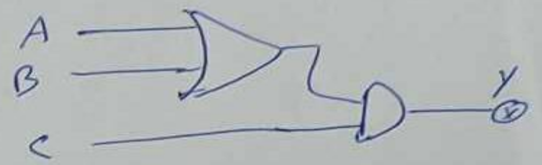
$= B\cdot C(\bar{A}+A) + A\bar{B}C$

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

$= C[B + A\bar{B}]$

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

$= C\cdot(B+A)$



3 Red → Using (DNF) (SOP)

$R = \bar{A}\cdot B\cdot C + AB\cdot C$

$= B\cdot C(\bar{A}+A)$

$= B\cdot C$

T5 : Distributive Law

(a) A (B + C) = A B + A C

(b) A + (B C) = (A + B) (A + C)

(c) A+A'B = A+B

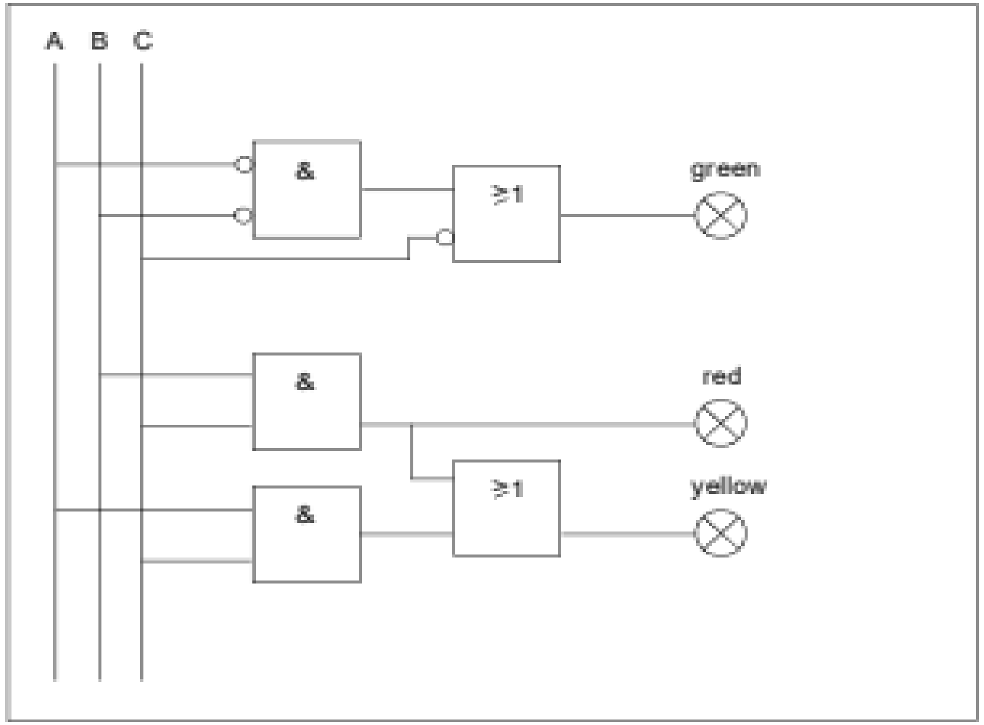


Fig. 1.2.3.2 Circuit

1.2.4 Circuit Design with the Aid of KV Diagrams

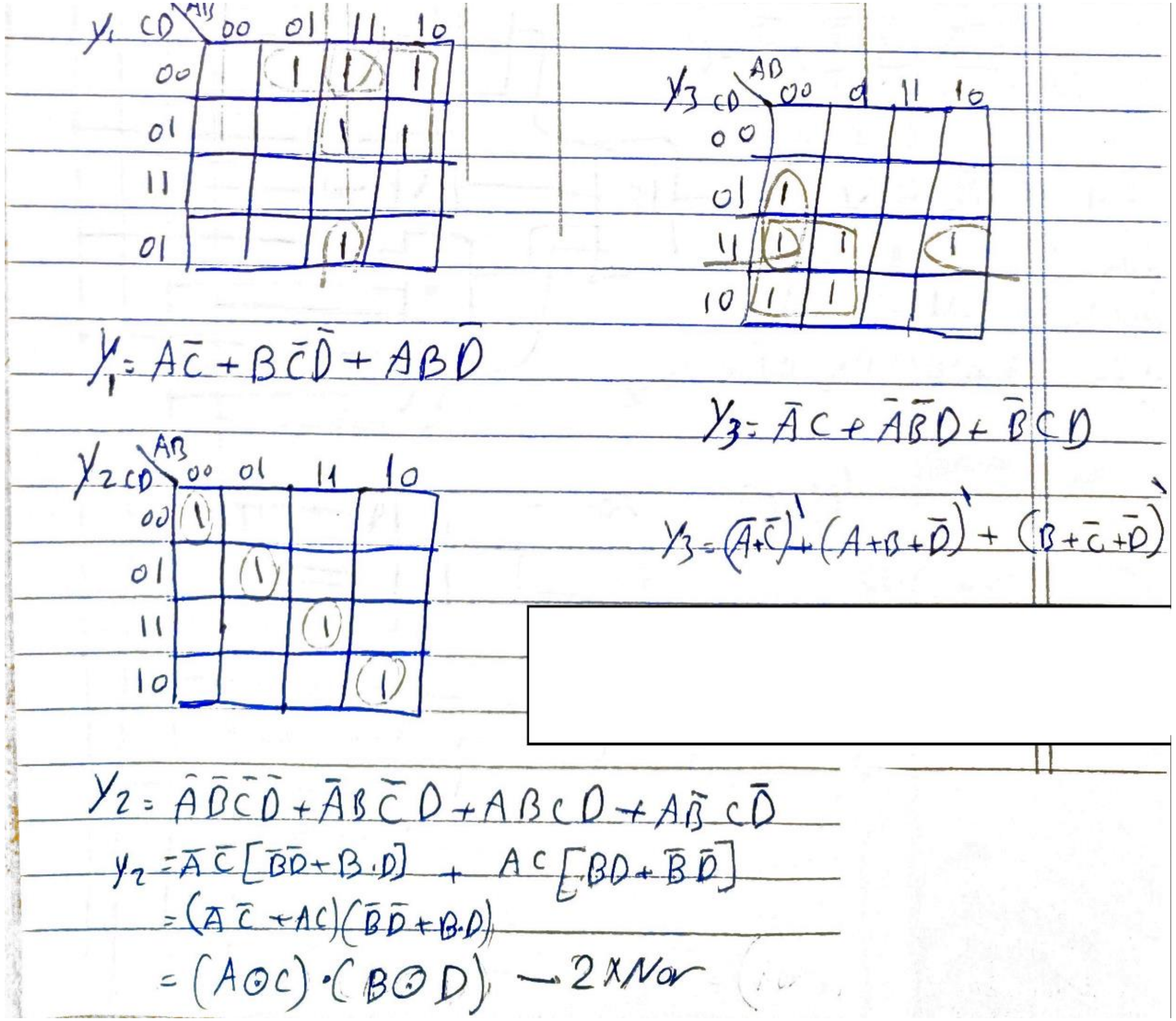
❐ Experiment 1: Number comparator for dual numbers

A digital circuit with which two dual numbers can be checked for equality or inequality is known as a number comparator.
In this experiment, a circuit is to be designed which is able to compare two dual numbers P and Q with two bits each. A greater-smaller comparison should also be carried out in the event of inequality.

❐ Experiment 1: Number comparator for dual numbers

Inputs				Outputs		
Number P		Number Q		P > Q	P = Q	P < Q
A	B	C	D	Y1	Y2	Y3
0	0	0	0	*0*	*1*	*0*
0	0	0	1	*0*	*0*	*1*
0	0	1	0	*0*	*0*	*1*
0	0	1	1	*0*	*0*	*1*
0	1	0	0	*1*	*0*	*0*
0	1	0	1	*0*	*1*	*0*
0	1	1	0	*0*	*0*	*1*
0	1	1	1	*0*	*0*	*1*
1	0	0	0	*1*	*0*	*0*
1	0	0	1	*1*	*0*	*0*
1	0	1	0	*0*	*1*	*0*
1	0	1	1	*0*	*0*	*1*
1	1	0	0	*1*	*0*	*0*
1	1	0	1	*1*	*0*	*0*
1	1	1	0	*1*	*0*	*0*
1	1	1	1	*0*	*1*	*0*

Table 1.2.4.1 Truth table



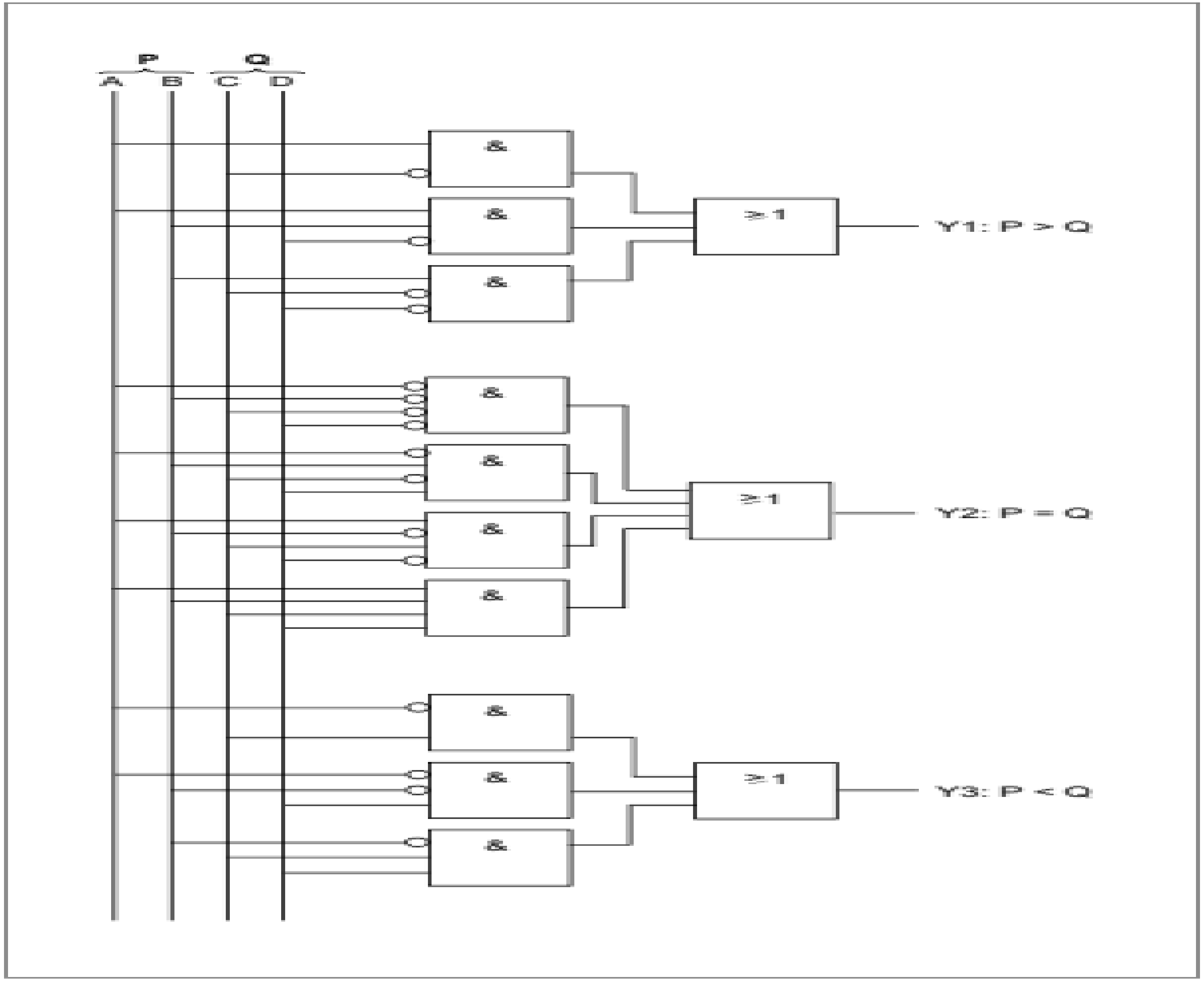


Fig. 1.2.4.4 Circuit