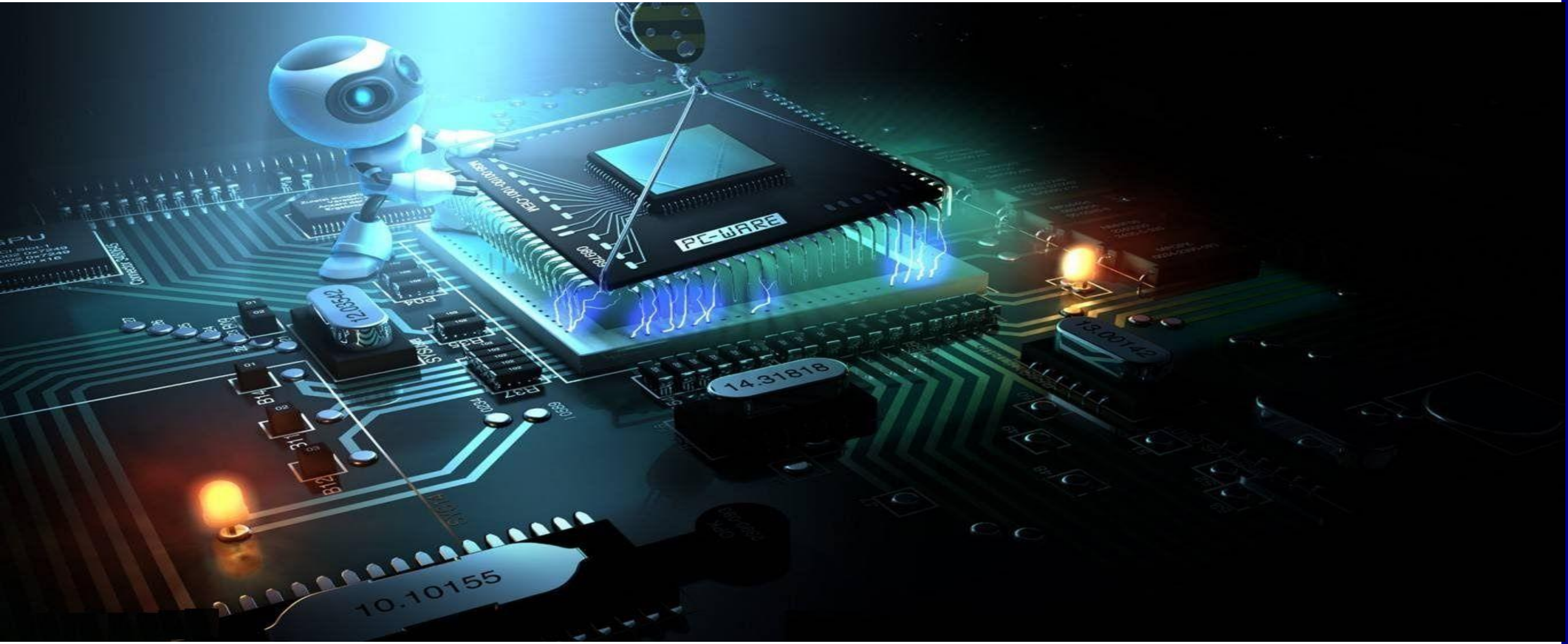
جامعة القاهرة الجديدة التكنولوجية

Fall 2023

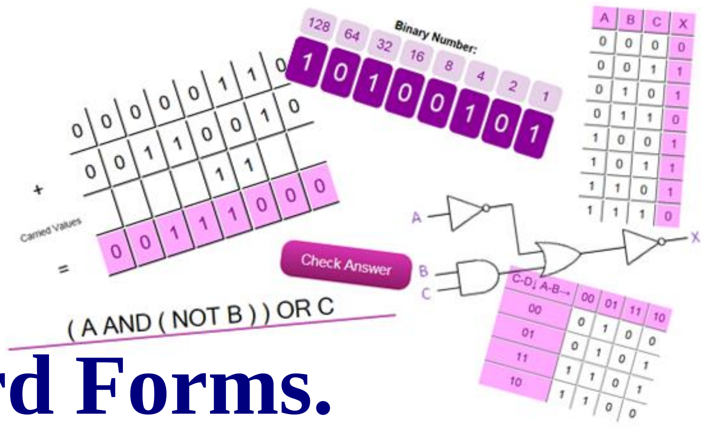
(5)



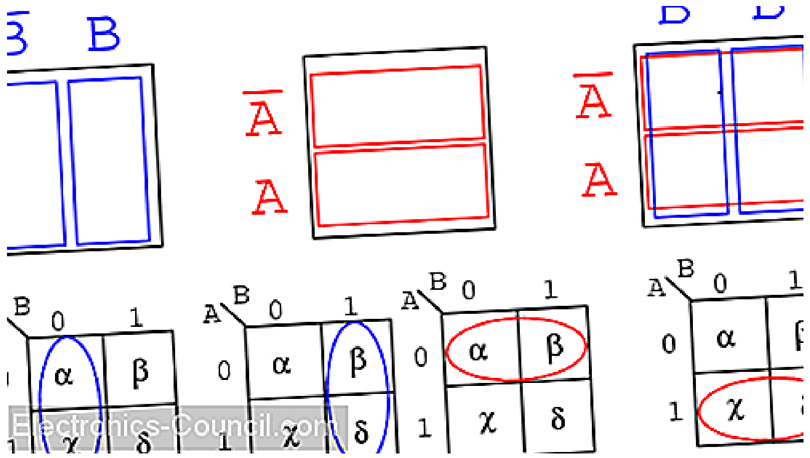
Digital Engineering

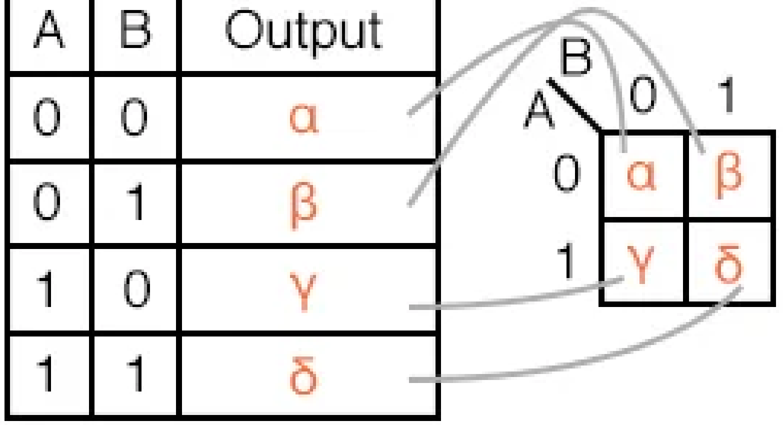
Dr. Hatem Yousry

Agenda



- Canonical and Standard Forms.
- Truth Table notation for Minterms and Maxterms.
- Karnaugh map.





A	B	Output
0	0	α
0	1	β
1	0	γ
1	1	δ

$$\begin{array}{c|cc} A\backslash B & 0 & 1 \\ \hline 0 & α & β \\ 1 & γ & δ \end{array}$$

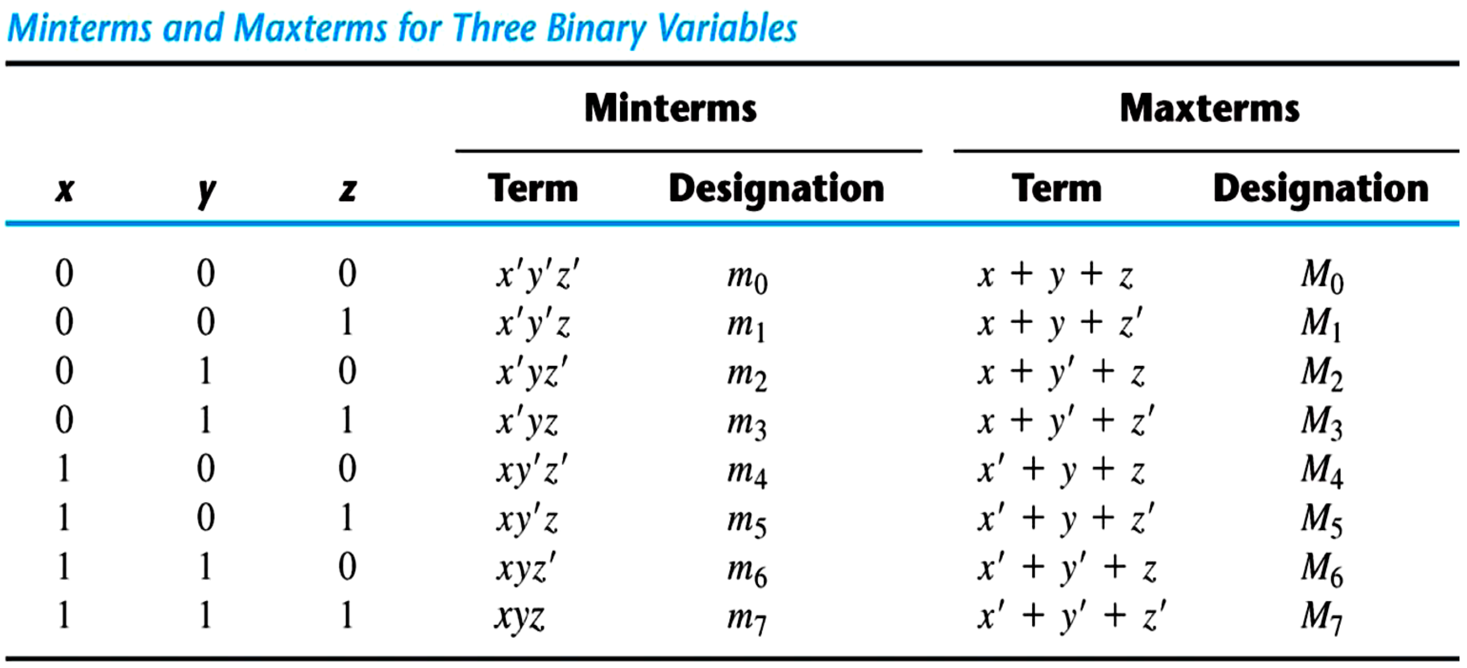
Canonical and Standard Forms

Minterms and Maxterms

- **A minterm** (standard product): an **AND** term consists of all literals in their normal form or in their complement form.
  - For example, two binary variables *x* and *y*,
    - $xy, xy', x'y, x'y'$
  - It is also called a **standard product.**
  - *n* variables con be combined to form 2^n minterms.
- **A maxterm** (standard sums): an **OR** term
  - It is also call a **standard sum.**
  - 2^n maxterms.

Minterms and Maxterms

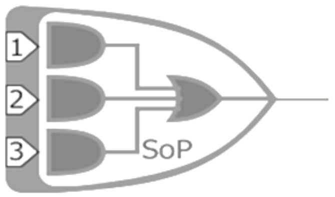
- Each *maxterm* is the complement of its corresponding *minterm*, and vice versa.

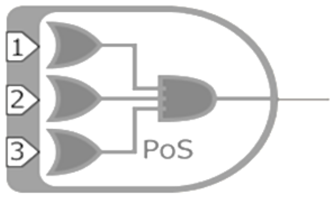


Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

SOP $\sum$ and POS $\prod$

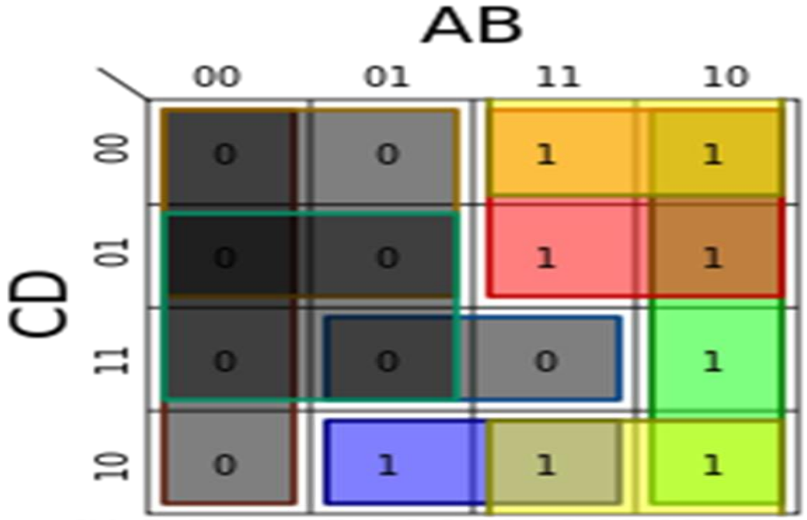




- SOP for **Minterm (m) in the form of**
- **And + And + And + ……**
- $\sum$ *(m1, m2 , …….m_j) = sum of minterms*
- **As** the **minterms** included are those m_j such that **F( ) = 1** in row *j* of the truth table for F( ).
- POS for **Maxterm (M) in the form of**
- **OR . OR . OR . ……….**
- $\prod$ ***(M1, M2, ……….M_j) = product of maxterms***
- **As** the **maxterms** included are those M_j such that **F( ) = 0** in row *j* of the truth table for F( ).
- **Observe that:** $m_j = M_j'$

Karnaugh Maps

- Karnaugh maps (K-maps) are ***graphical*** representations of boolean functions.
- One ***map cell*** corresponds to a row in the truth table.
- Also, one map cell corresponds to a minterm or a maxterm in the boolean expression
- **Multiple-cell areas of the map correspond to standard terms.**



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	0	1
10	0	1	1	1

f(A,B,C,D) = Σ(6,8,9,10,11,12,13,14)

F=AC'+AB'+BCD'+AD'

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

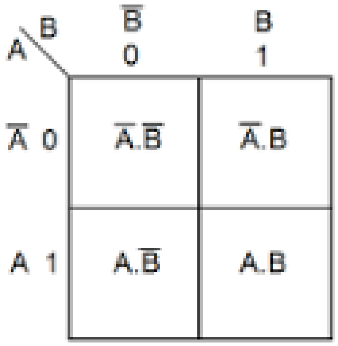
The Karnaugh map (KM or K-map)

- **Karnaugh maps** are used **to simplify real-world logic** requirements so that they can be implemented using a minimum number of physical logic gates.
- A **sum-of-products** expression can always be implemented using **AND gates feeding into an OR gate**, and a p**roduct-of-sums** expression leads to **OR gates feeding an AND gate**.
- Karnaugh maps can also be used to simplify logic expressions in **software design**. Boolean conditions can get very complicated, which makes the code difficult to read and to maintain. Once **minimized, canonical sum-of-products and product-of-sums expressions can be implemented directly using AND and OR logic operators.**

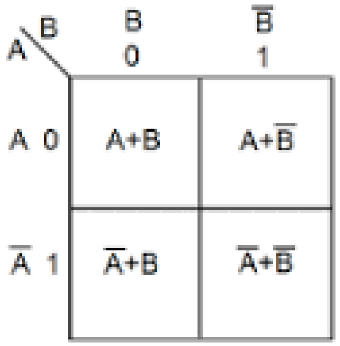
What are Karnaugh maps?

- **Karnaugh maps provide an alternative way of simplifying logic circuits.**
- Instead of using Boolean algebra simplification techniques, you can **transfer logic values from a Boolean statement or a truth table into a Karnaugh map.**
- The arrangement of 0's and 1's within the map helps you to visualize the logic relationships between the variables and leads directly to a simplified Boolean statement.

A. SOP: -

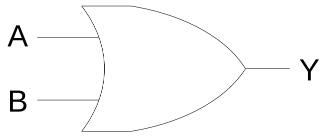


B. POS: -

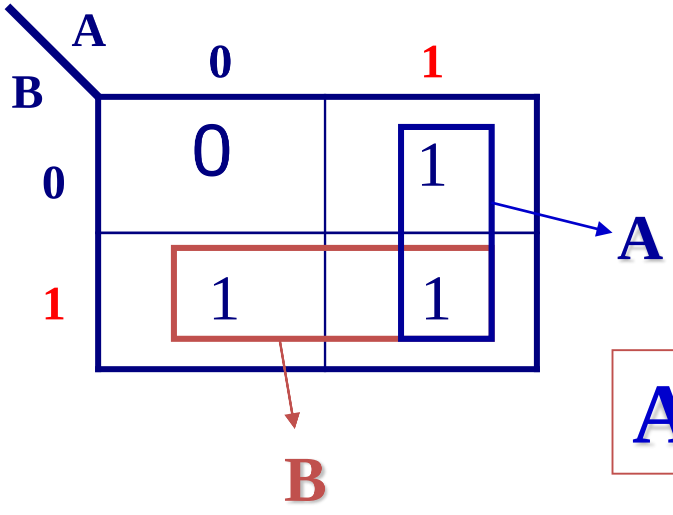


Karnaugh Maps

- The Karnaugh map is completed by **entering a '1'(or '0') in each of the appropriate cells.**
- Within the map, **adjacent cells containing 1's (or 0's) are grouped together in twos, fours, or eights.**
- **2-variable Karnaugh maps** are trivial but can be used to introduce the methods you need to learn. The map for a 2-input OR gate looks like this:



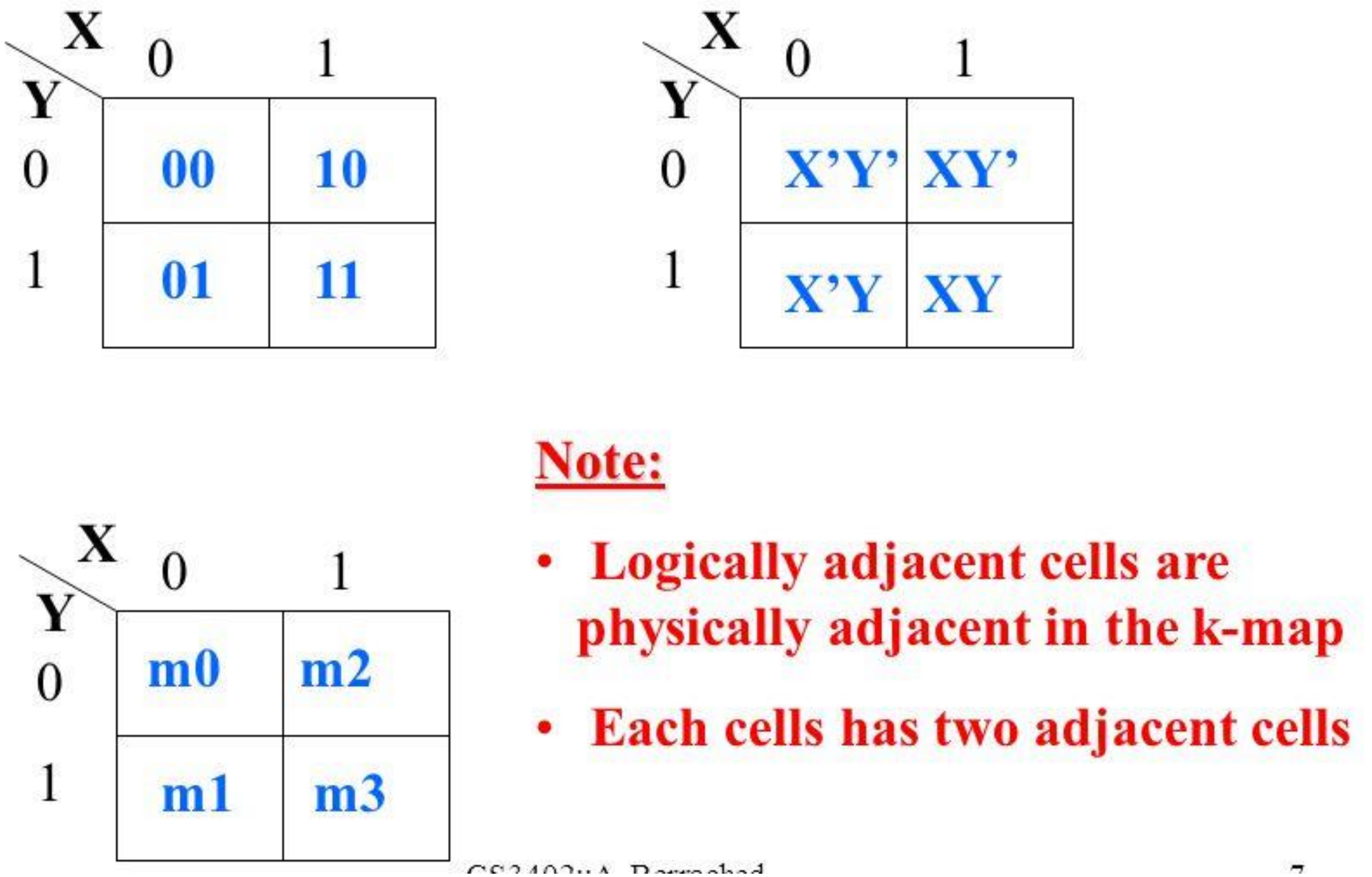
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



A+B

Two-Variable Map

- Simplifying Boolean Expressions Using K-Map Method



Y \ X	0	1
0	00	10
1	01	11

Y \ X	0	1
0	X’Y’	XY’
1	X’Y	XY

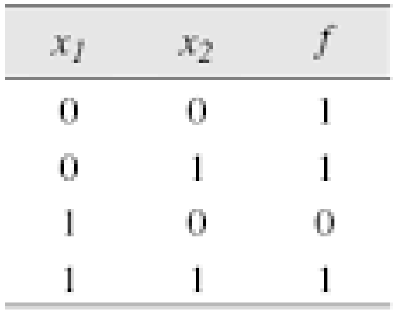
Y \ X	0	1
0	m0	m2
1	m1	m3

Note:

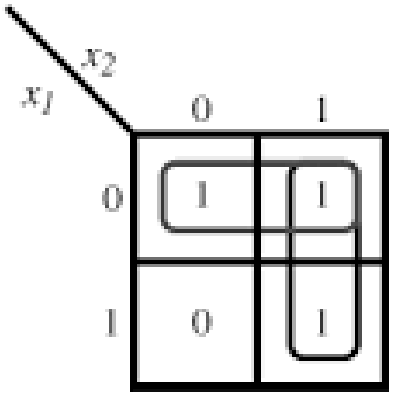
- **Logically adjacent cells are physically adjacent in the k-map**
- **Each cells has two adjacent cells**

Algorithm for Deriving the Minimal SOP

1. **Circle** all prime implicates on the k-map
2. **Identify and select** all essential prime implicates
3. **Select a minimum subset** of the remaining prime implicates to cover those minterms not covered by the essential prime implicates.

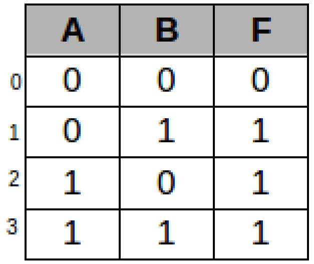


x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1

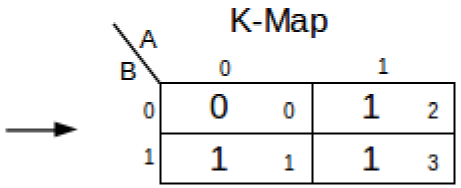


$f = \overline{x}_1 + x_2$

2 Variable Truth Table

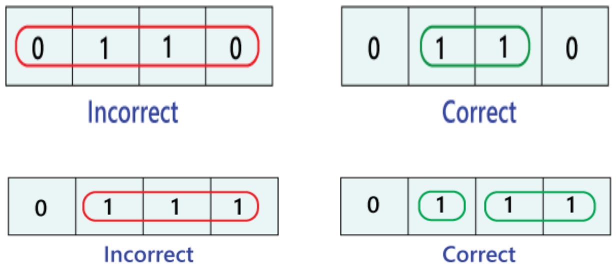


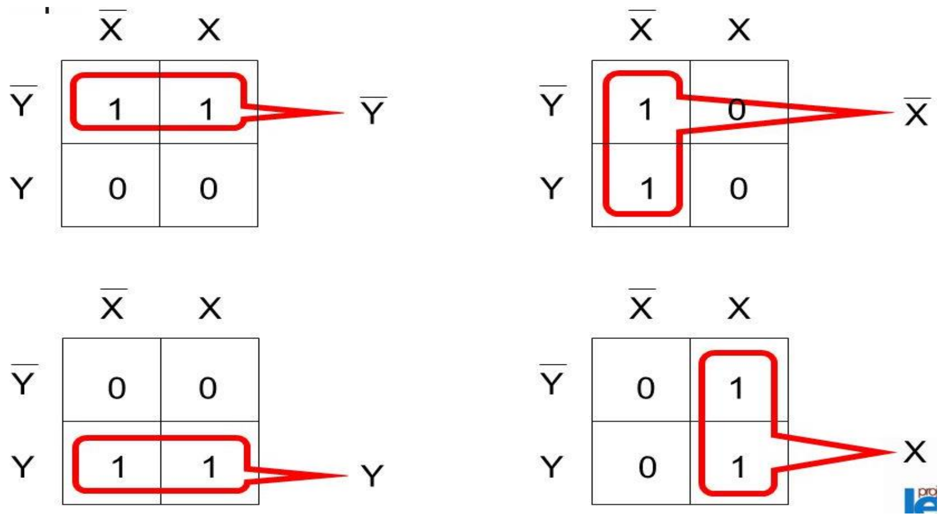
	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1



Function Minimization Using K-Maps

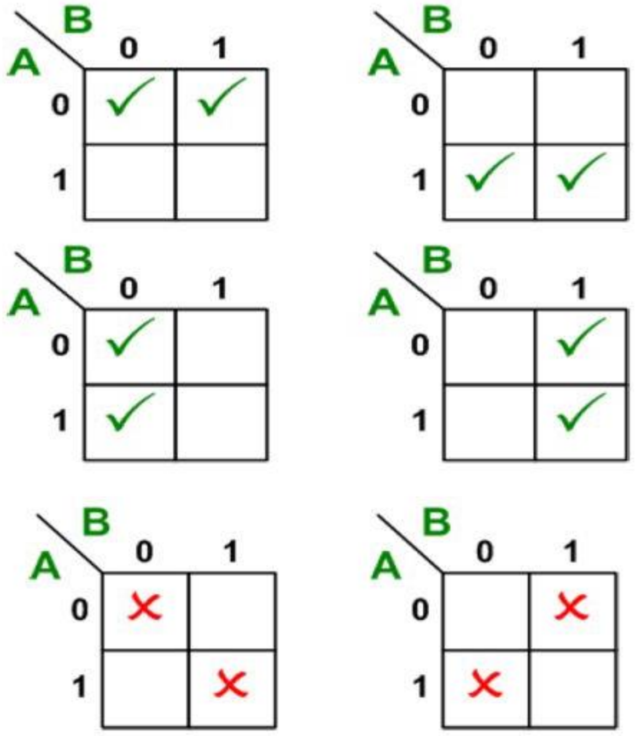
- 1. Each square **(Minterm)** in a k-map of 2 variables has **2 logically adjacent squares.**
- 2. **Combine** only the minterms for which the function is 1.
- 3. When combining terms on a k-map, group adjacent squares in groups of powers of 2 **(i.e. 2, 4, 8, etc.).**
- **Grouping two squares eliminates one variables**, grouping 4 squares eliminates 2 variables, etc. Can't combine a group of 3 minterms

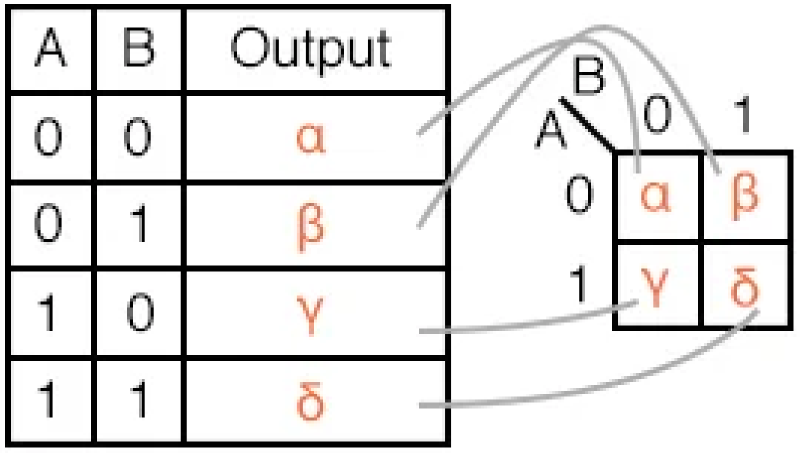




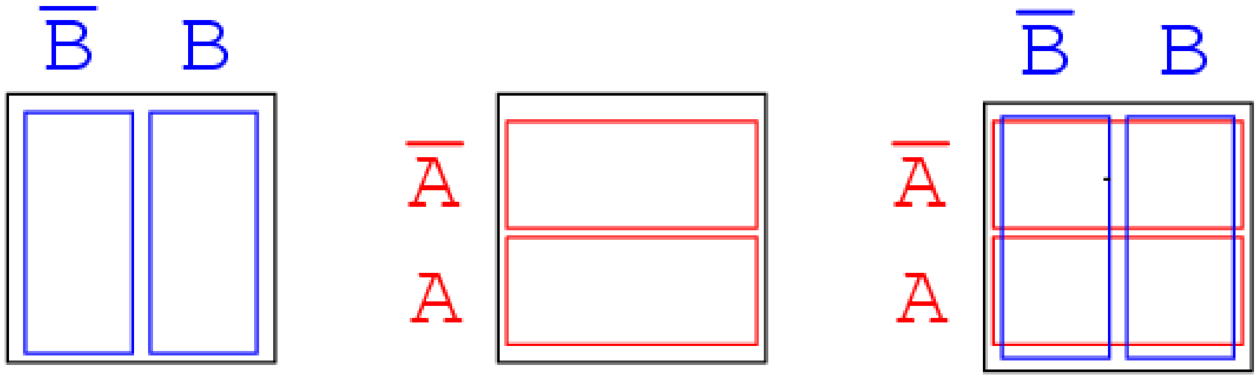
Function Minimization Using K-Maps

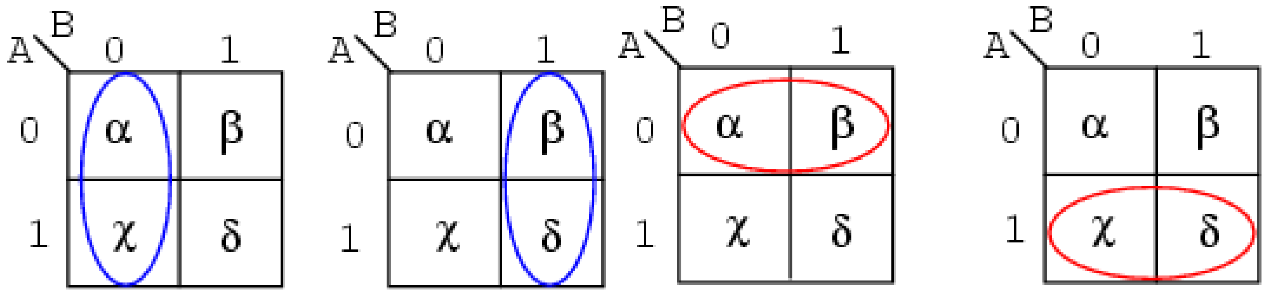
- 4. **Group as many squares together** as possible; the larger the group is the fewer the number of literals in the resulting product term
- 5. **Select as few groups** as possible to cover all the minterms of the functions. A Minterm is covered if it is included in at least one group. Each Minterm may be covered as many times as it is needed; however, it must be covered at least once.
- 6. **In combining squares** on the map, always begin with those squares for which there are the fewest number of adjacent squares (the "loneliest" squares on the map).





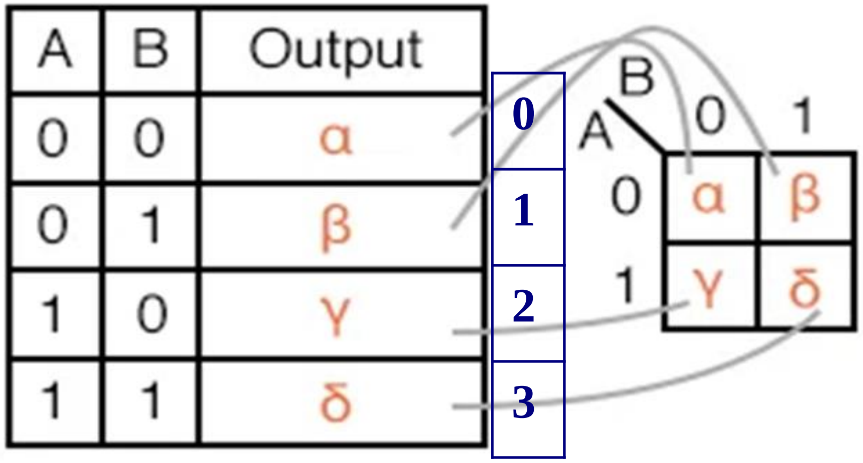
A	B	Output
0	0	α
0	1	β
1	0	γ
1	1	δ

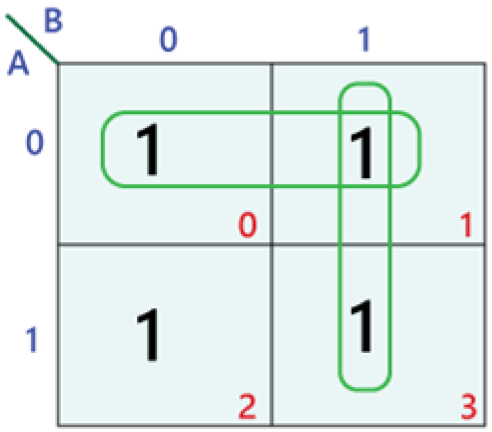




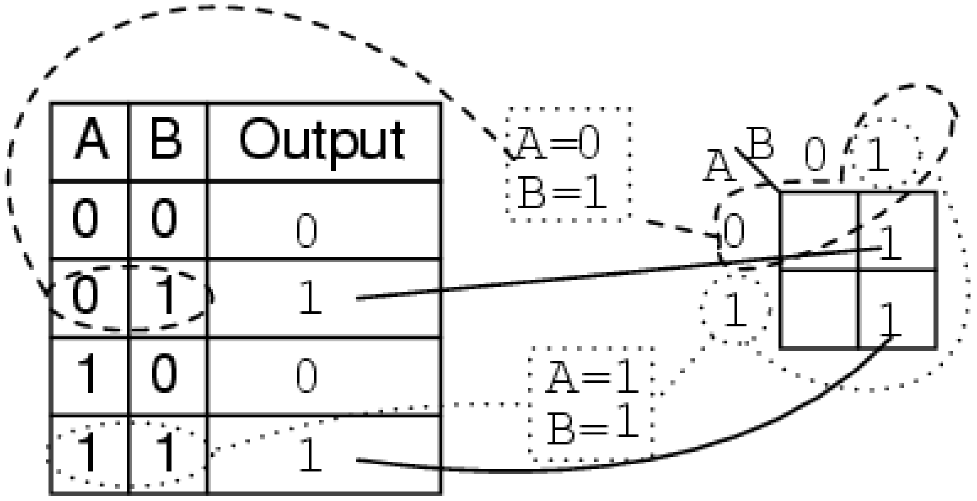
Example - Two-Variable Map

- **Y=A'B' + A'B+AB**

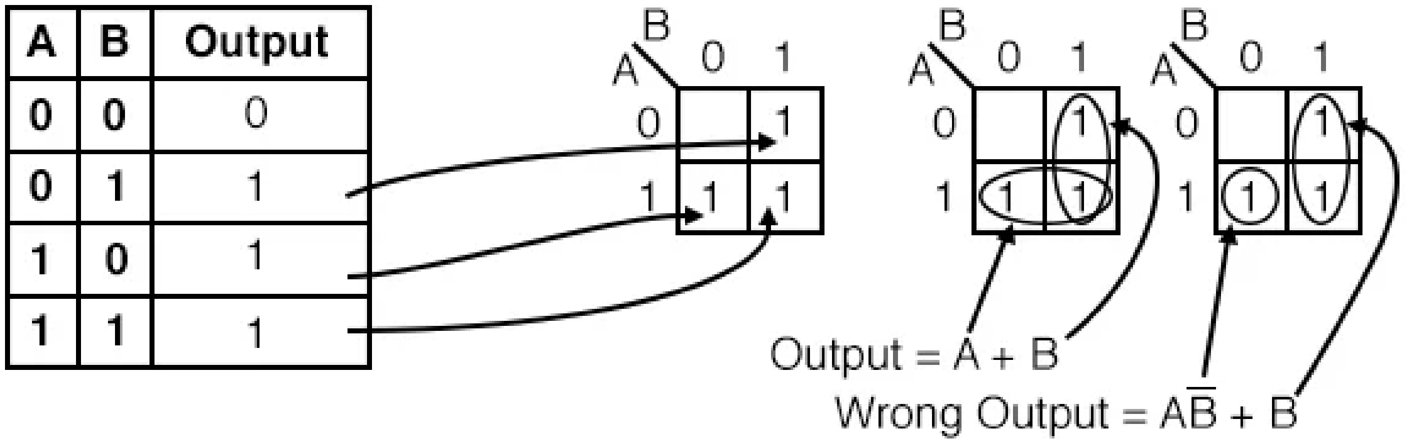




- **Simplified expression: Y=A'+B**



A	B	Output
0	0	0
0	1	1
1	0	0
1	1	1



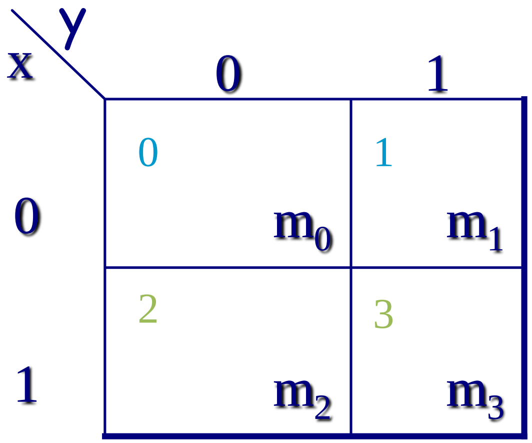
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Output = $A + B$

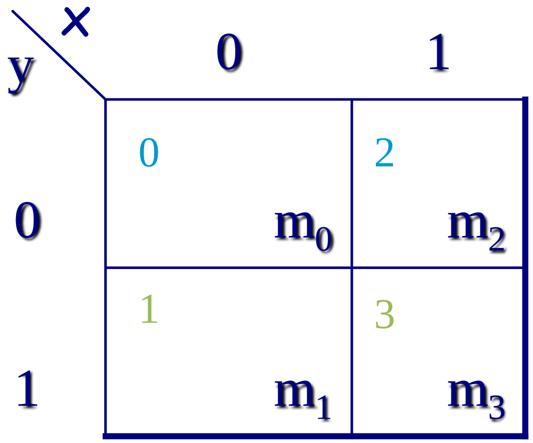
Wrong Output = $A\overline{B} + B$

Two-Variable Map

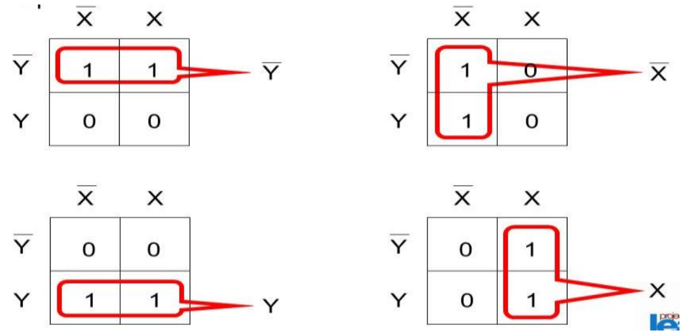
- NOTE: ordering of variables is IMPORTANT for f(x,y), x is the row, y is the column.
- **Cell 0 represents x'y'; Cell 1 represents x'y; etc.**
- **If a Minterm is present in the function, then a 1 is placed in the corresponding cell.**



OR

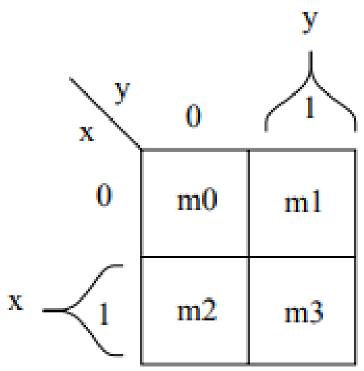


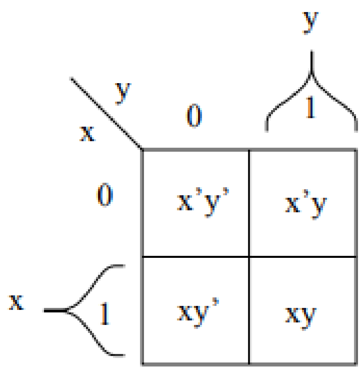
Two-Variable Map



- Any two adjacent cells in the map differ by ONLY one variable, which appears complemented in one cell and un-complemented in the other.
- **Example:**
  m_0 (=x'y') is adjacent to m_1 (=x'y) and m_2 (=xy') but NOT m_3 (=xy)

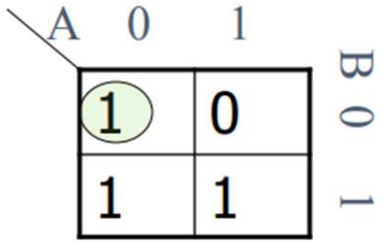
Two variables, F =f(x,y)





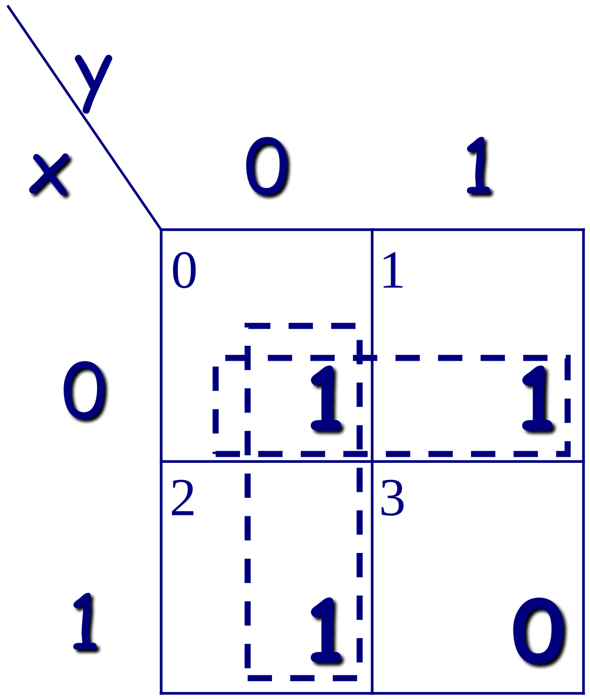
Two variable K-map

f(A,B)=∑m(0,1,3)=A\`B\`+A\`B+AB



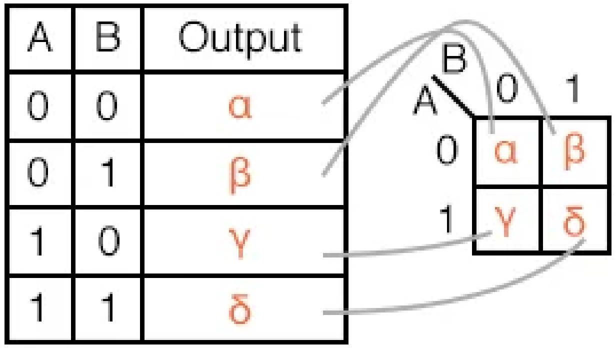
2-Variable Map -- Example

- f(x,y) = **x'y'+ x'y + xy'**
  $= m_0 + m_1 + m_2$
  $= x' + y'$
- 1s placed in K-map for specified minterms m_0, m_1, m_2
- Grouping (ORing) of 1s allows simplification
- What (simpler) function is represented by each dashed rectangle?
  - **x' = $m_0 + m_1$**
  - **y' = $m_0 + m_2$**
- Note m_0 covered twice



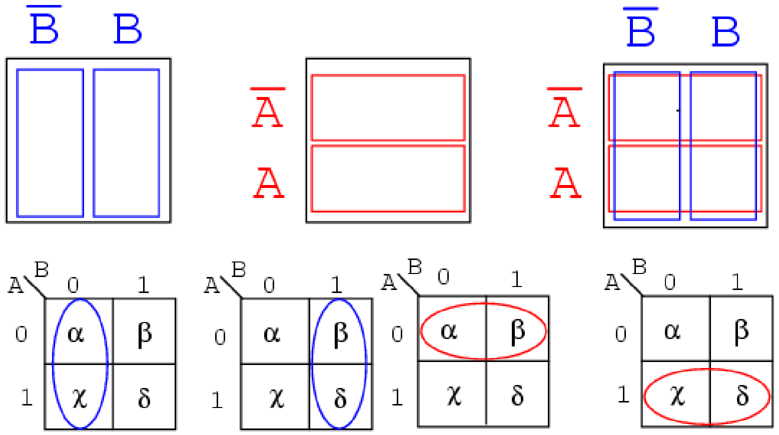
Rules to obtain the most simplified expression

- **A Karnaugh map is a graphical method used to obtained the most simplified form of an expression in a standard form (Sum-of-Products or Product-of-Sums).**
- The Karnaugh map is completed by entering a '1‘ = mj (or ‘0’ = Mj) in each of the appropriate cells.
- Within the map, **adjacent cells containing 1's (or 0’s) are grouped** together in twos, fours, or eights.



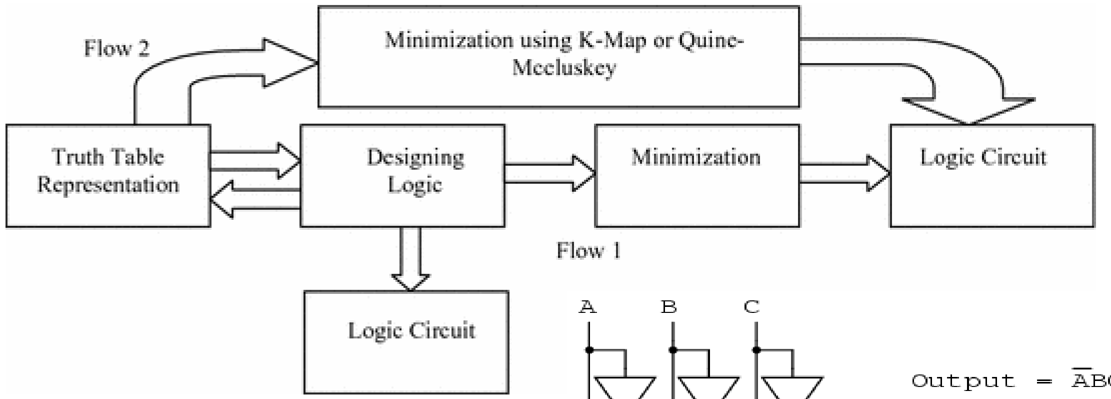
A	B	Output
0	0	α
0	1	β
1	0	γ
1	1	δ

$$\begin{array}{c|cc} A\backslash B & 0 & 1 \\ \hline 0 & \alpha & \beta \\ 1 & \gamma & \delta \end{array}$$

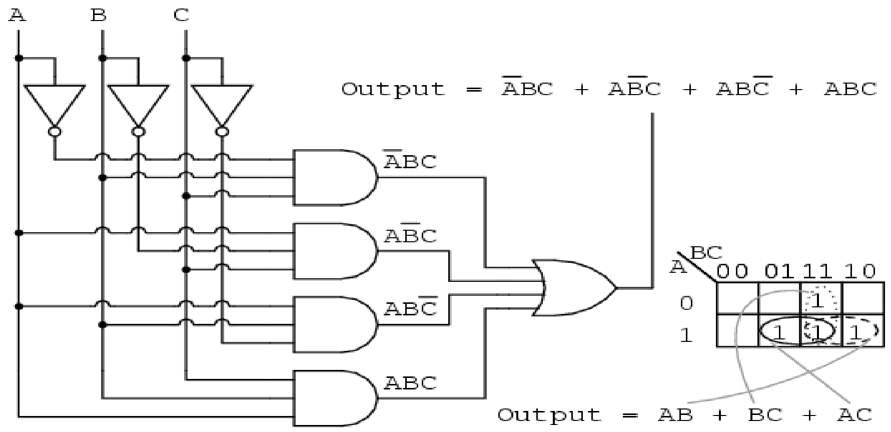


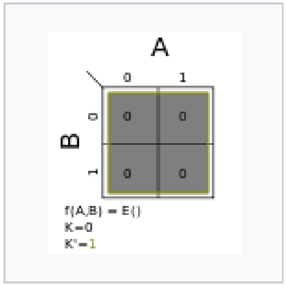
Mathematical Study for Reduction of Variables in Karnaugh Map

- For designing any logic circuit, the major important stages may be truth table representation, framing logical statements, reduction using minimizing technique, and the most important and the desired part is design.

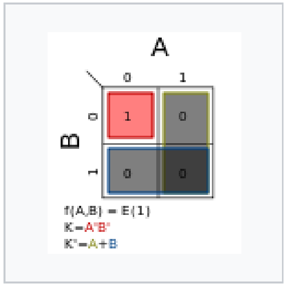


Flow 2
Minimization using K-Map or Quine-Mccluskey
Truth Table Representation
Designing Logic
Minimization
Logic Circuit
Flow 1
Logic Circuit

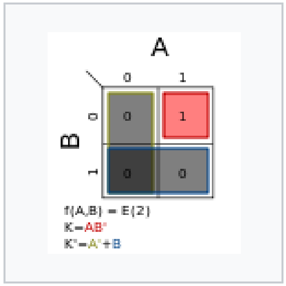




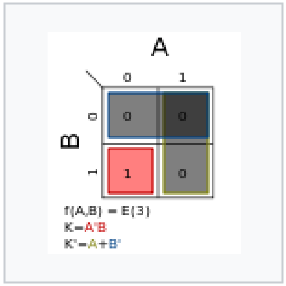
Σ*m*(0); *K* = 0



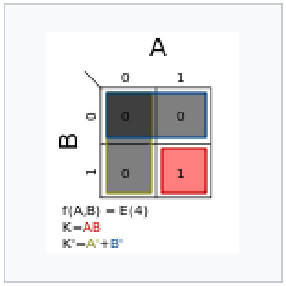
Σ*m*(1); *K* = *A'B'*



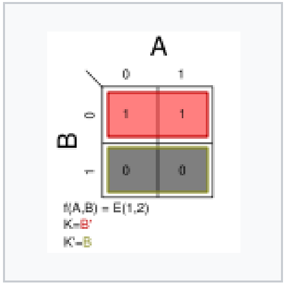
Σ*m*(2); *K* = *AB'*



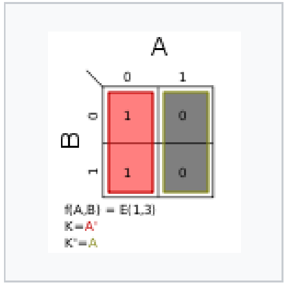
Σ*m*(3); *K* = *A'B*



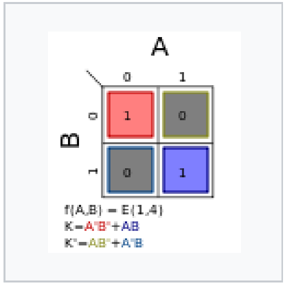
Σ*m*(4); *K* = *AB*



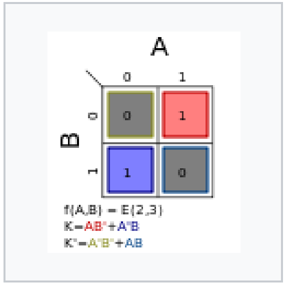
Σ*m*(1,2); *K* = *B'*



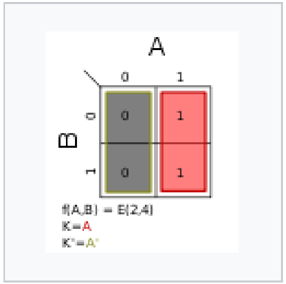
Σ*m*(1,3); *K* = *A'*



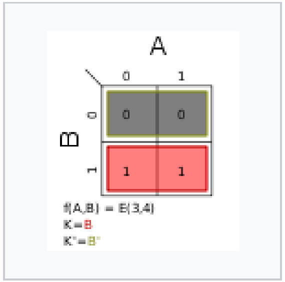
Σ*m*(1,4); *K* = *A'B'* + *AB*



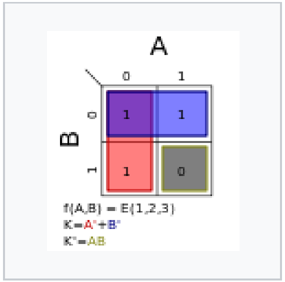
Σ*m*(2,3); *K* = *AB'* + *A'B*



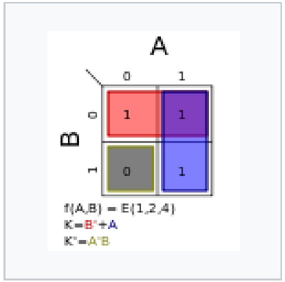
Σ*m*(2,4); *K* = *A*



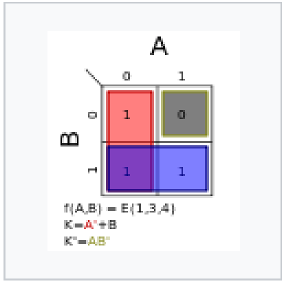
Σ*m*(3,4); *K* = *B*



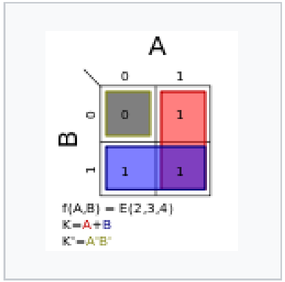
Σ*m*(1,2,3); *K* = *A'* + *B'*



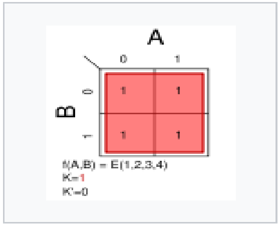
Σ*m*(1,2,4); *K* = *A* + *B'*



Σ*m*(1,3,4); *K* = *A'* + *B*



Σ*m*(2,3,4); *K* = *A* + *B*



Σ*m*(1,2,3,4); *K* = 1

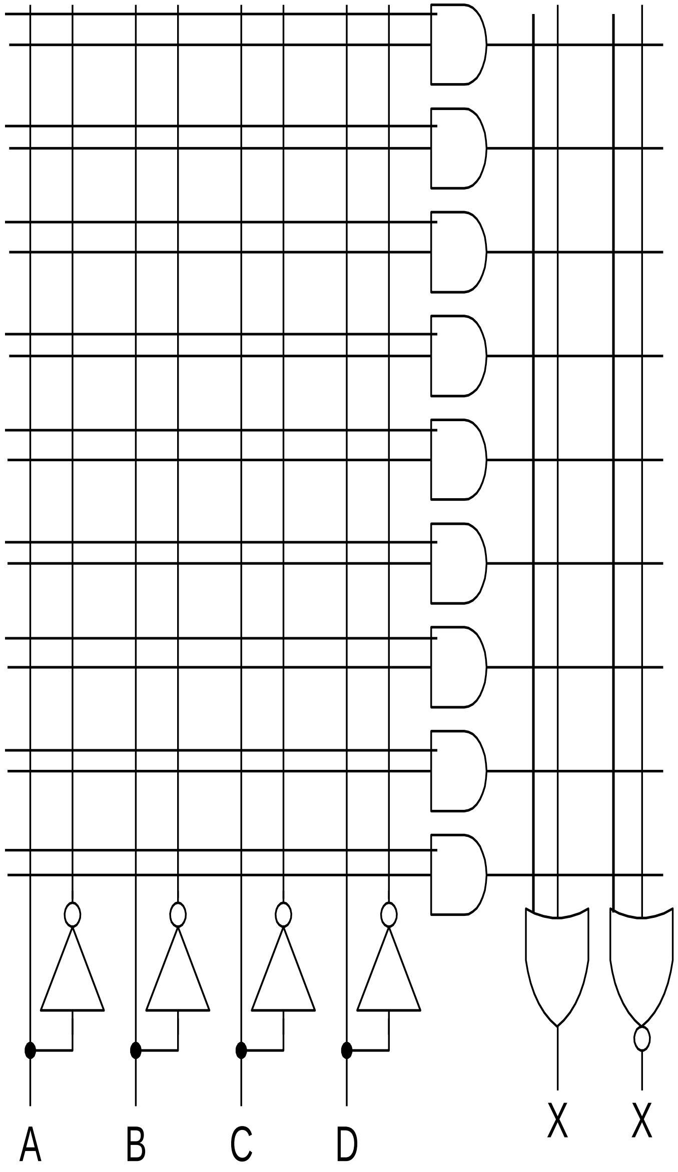
Name:

	Inputs				Output		
0	0	0	0	0			
1	0	0	0	1			
2	0	0	1	0			
3	0	0	1	1			
4	0	1	0	0			
5	0	1	0	1			
6	0	1	1	0			
7	0	1	1	1			
8	1	0	0	0			
9	1	0	0	1			
10	1	0	1	0			
11	1	0	1	1			
12	1	1	0	0			
13	1	1	0	1			
14	1	1	1	0			
15	1	1	1	1			

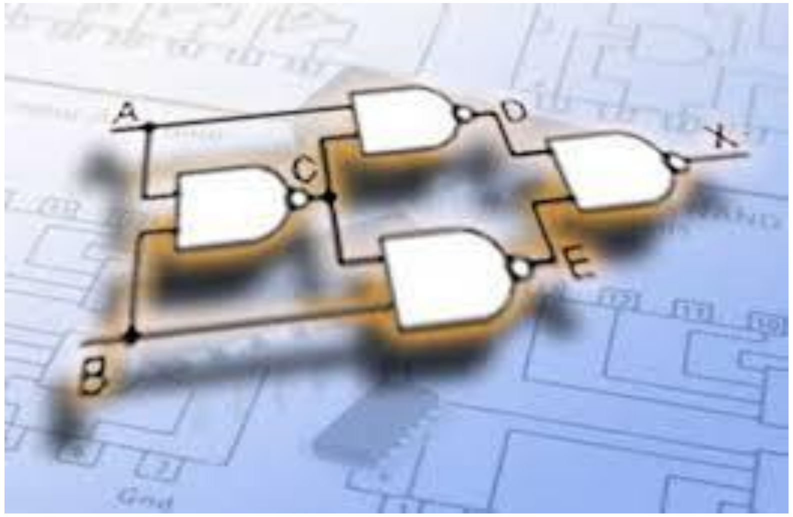
X =

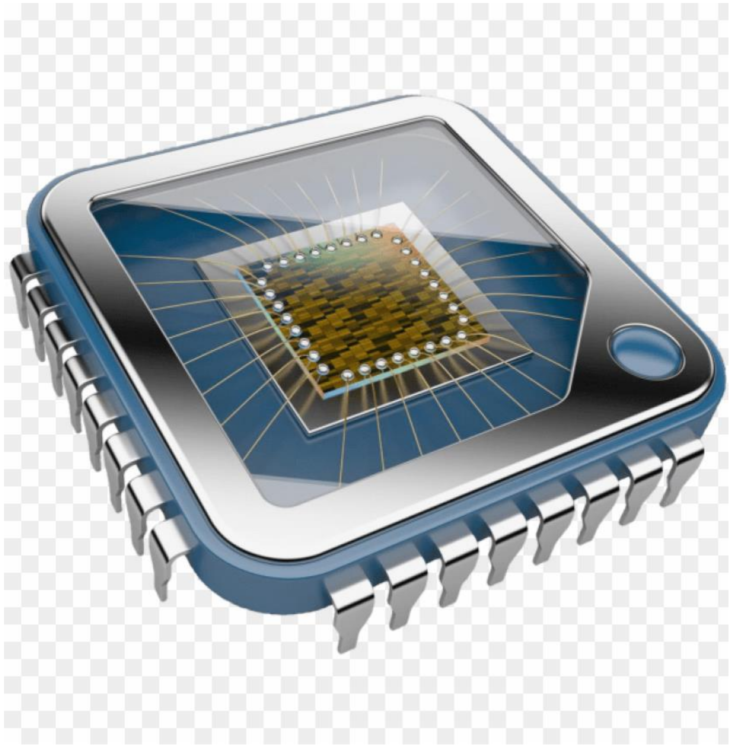
K- Map

X =



Thank You





Dr. Hatem Yousry
E-mail: Hyousry@nctu.edu.eg