Lecture 05
Karnaugh Map (K-Map)

ECE09/CPE07 – Digital Electronics 1:
Logic Circuit and Switching Theory

Engr. Zoren P. Mabunga, ECE

What are Karnaugh Maps?

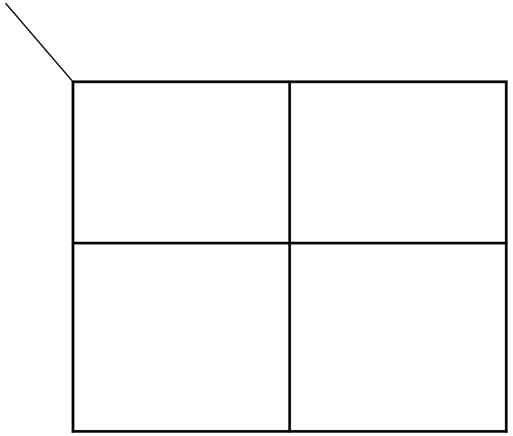
- The Karnaugh Map (or simply K-map) is similar to a truth table because it presents all the possible values of inputs and the resulting output for each value. However, instead of being organized into columns and rows like the truth table, the K-map is an array of squares (or cells) in which each squares represent a binary value of the input variables.
- 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.

Karnaugh maps

- Karnaugh maps, or K-maps, are often used to simplify logic problems with 2, 3 or 4 variables.

Cell = 2^n ,where n is a number of variables

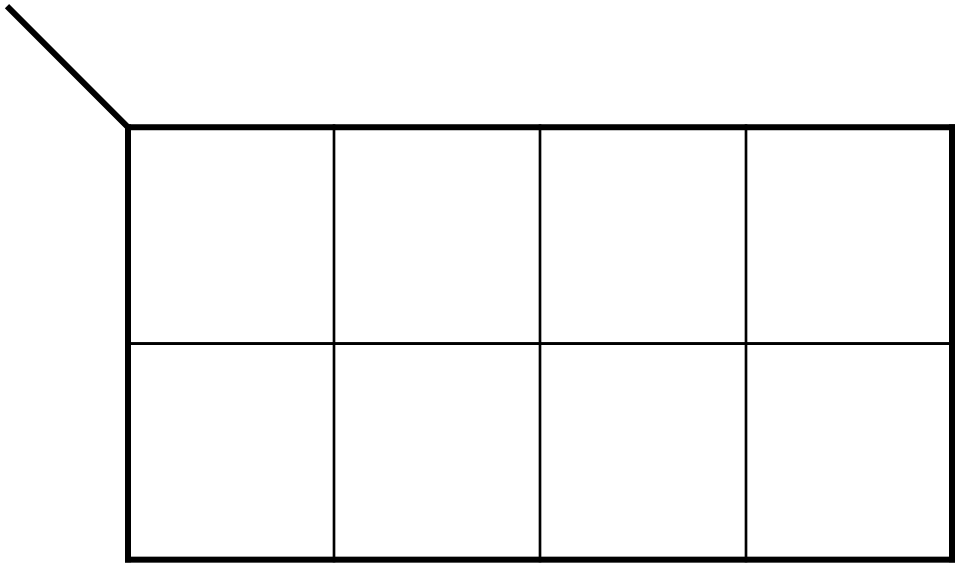
For the case of 2 variables, we form a map consisting of 2^2=4 cells as shown in Figure



Karnaugh maps

- 3 variables Karnaugh map

Cell = 2^3=8



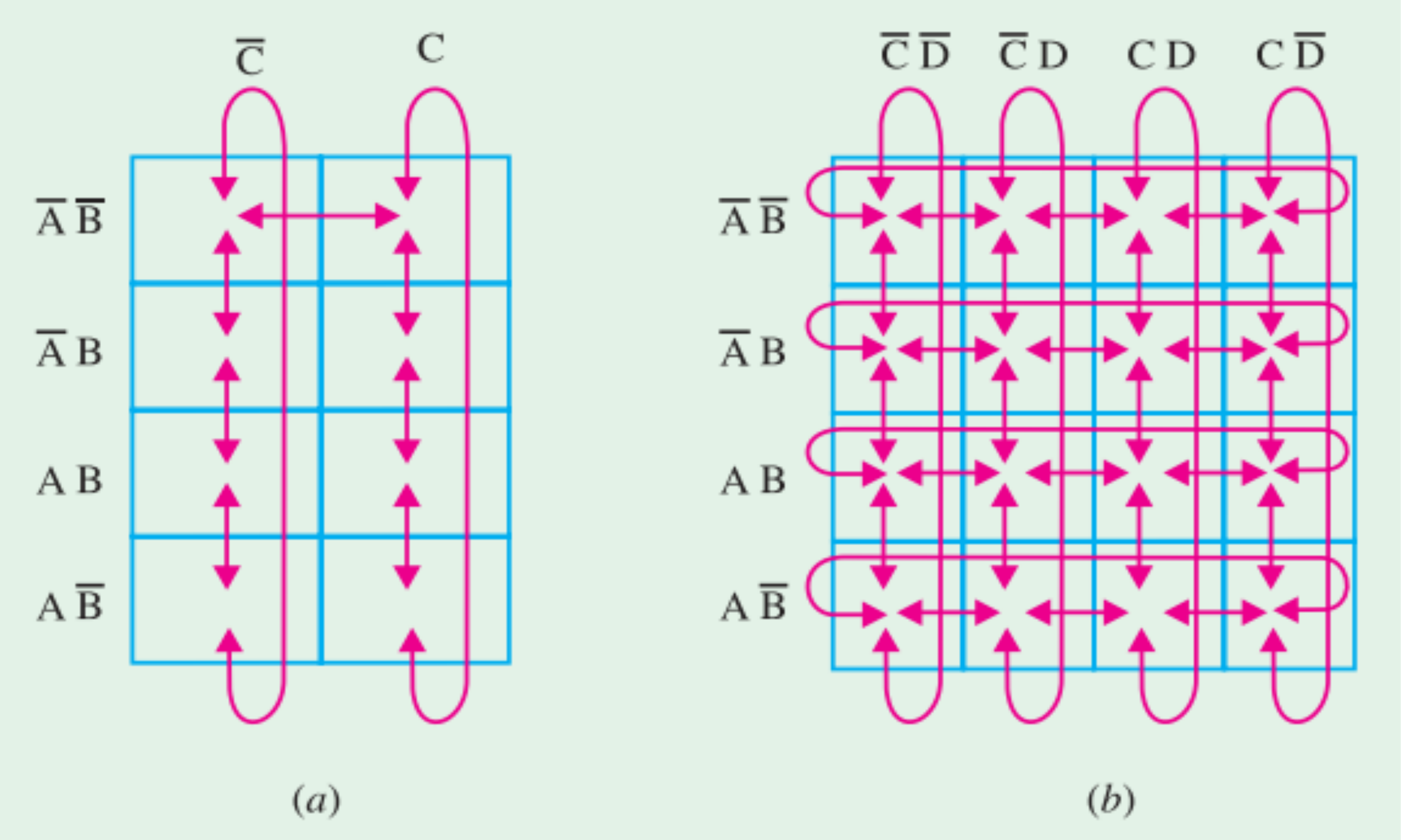
Karnaugh maps

- 4 variables Karnaugh map

	00	**01**	**11**	**10**
00				
01				
11				
10				

Square Adjacency in K-Map

- Adjacency is defined as a single-variable change. It means that the squares that differ by only one variable are adjacent.
- Each square is adjacent to the squares that are immediately next to it on any of its four sides. However, a square is not adjacent to the squares that diagonally touch any of its corners.
- It may also be noted that squares in the top row are adjacent to the corresponding squares in the bottom row and squares in the outer left column are adjacent to the corresponding squares in the outer right column. (wrap-around adjacency)


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

Simplification of Boolean Expression Using K-Map

- Step 1: GROUPING THE 1’s:
  - We can group 1’s on the map according to the following rules by enclosing those adjacent squares containing 1s. The objective is to maximize the size of groups and to minimize the number of groups.

a) A group must contain 1, 2, 4, 8, 16 squares.

b) Each square in the group must be adjacent to one or more squares in the group but all squares in the group do not have to be adjacent to each other.

c) Always include the largest number of 1s in a group in accordance with rule 1.

d) Each 1 on the Karnaugh Map must be included in at least one group.

Simplification of Boolean Expression Using K-Map

- Step 2: DETERMINING THE PRODUCT TERM FOR EACH GROUP
- Step 3: SUMMING THE RESULTING PRODUCTS

Examples

1. Simplify the following Boolean Expression using the Karnaugh Map

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

Examples

1. Simplify the following Boolean Expression using the Karnaugh Map

$$X = \bar{A}B\bar{C}D + A\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}D + AB\bar{C}D + AB\overline{CD} + ABCD$$

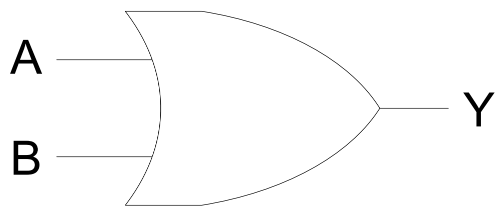
Examples

1. Simplify the following Boolean Expression using the Karnaugh Map

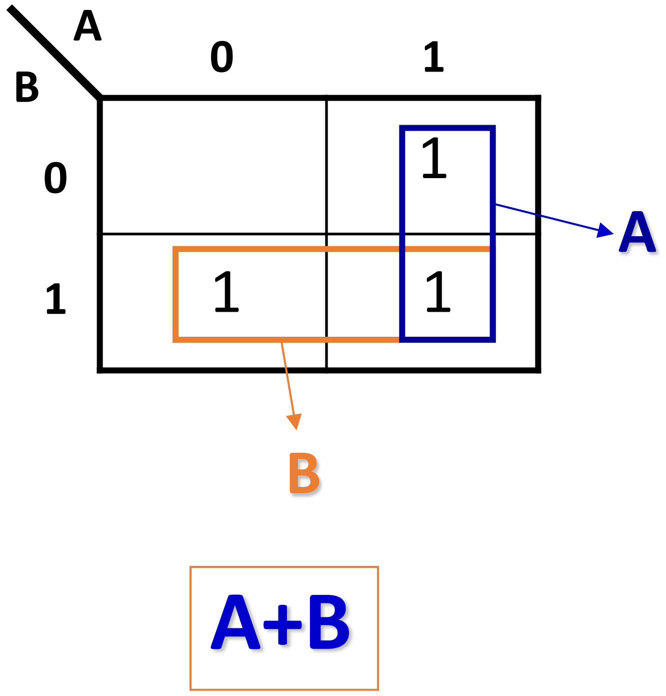
$$X = B\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}D + \overline{A}BCD + ABCD$$

Mapping Directly from Truth Table to K-Map

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



Example

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

Example

x	Y	z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Example

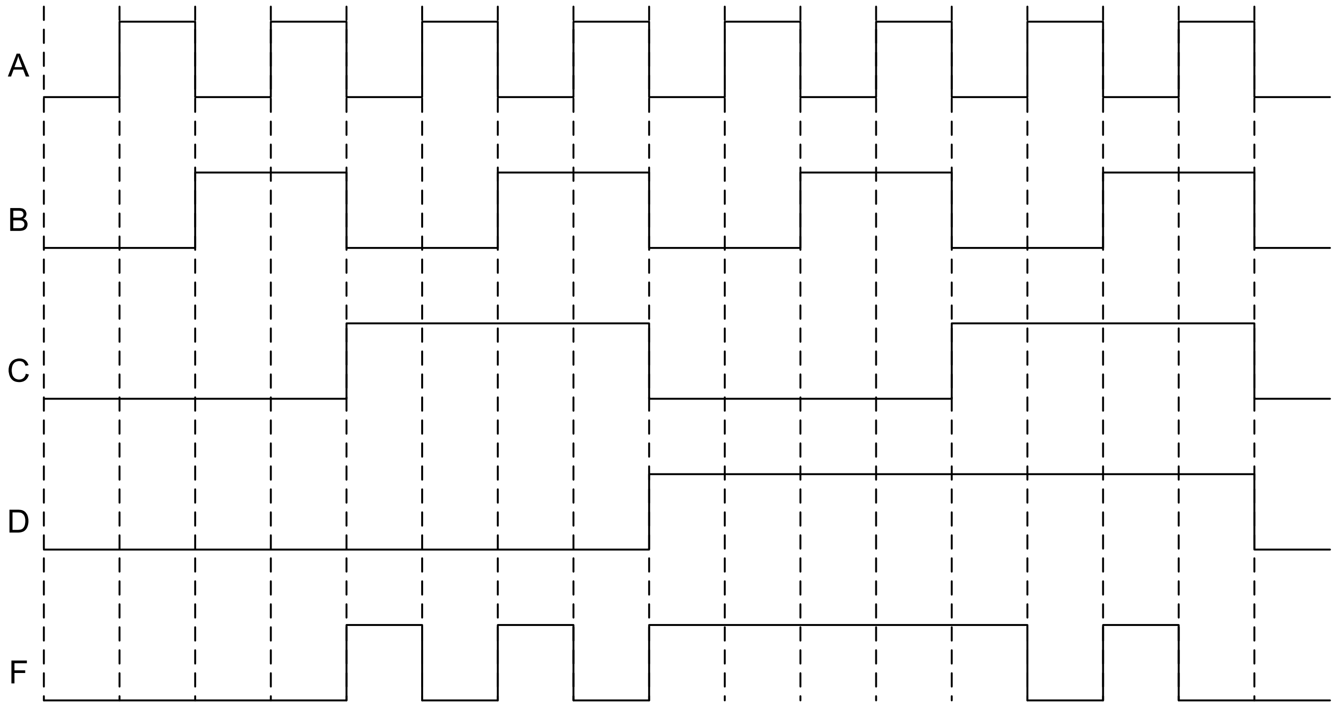
w	x	Y	z	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Example

w	x	Y	z	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
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	1
1	1	1	1	1

Exercise

- Design two-level NAND-gate logic circuit from the follow timing diagram.

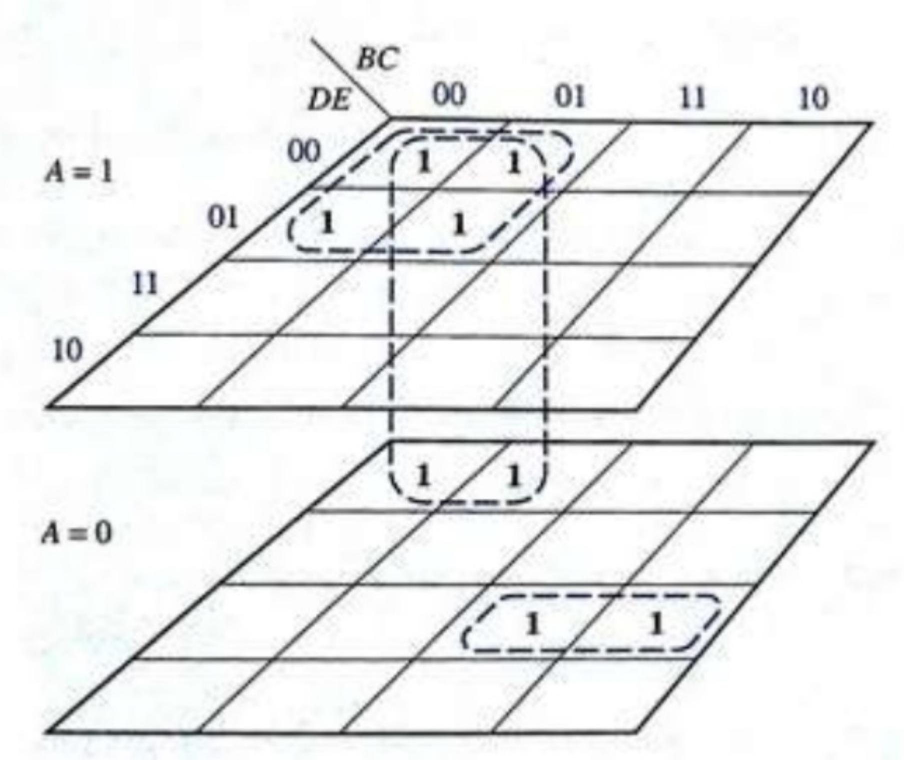


$$F(w, x, y, z) = \Sigma(0, 1, 5, 7, 9, 11, 12, 14)$$

$$F(w, x, y, z) = \Sigma(2, 6, 9, 10, 13, 15)$$

Five Variable K-Map

- Consists of 2 four variable maps



Example

- Find the equivalent simplified Boolean expression of the equation below.

$$F(A, B, C, D, E) = \sum(0,2,4,6,10,13,21,22,25,29,31)$$

Don’t Care Conditions

- In digital systems design sometimes a situation arises in which some input variable conditions are not allowed. For example in a BCD code, there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, 1111.
- Since these unallowed states will never occur in an application involving BCD code, they can be treated as “don’t care” terms with respect to their effective output.

Example

$F(a,b,c,d) = \Sigma(0, 1, 3, 5, 9, 12)$ $D(a,b,c,d) = \Sigma(2, 4, 6, 11)$

$F(a,b,c,d) = \Sigma(1,4,5,6,9,13)$ $D(a,b,c,d) = \Sigma(0,3,11)$

Example

- Design a logic circuit that will give an output of 1 when the BCD input is equal or greater than 6.

Example

- The ECE students of SLSU was tasked to build a lamp logic for a stationary bicycle exhibit at the local science museum. As a rider increases his pedaling speed, lamps will light on a bar graph display. No lamps will light for no motion. As speed increases, the lower lamp, L1 lights, then L1 and L2, then, L1, L2, and L3, until all lamps light at the highest speed. Once all the lamps illuminate, no further increase in speed will have any effect on the display. A small DC generator coupled to the bicycle tire outputs a voltage proportional to speed. It drives a tachometer board which limits the voltage at the high end of speed where all lamps light. No further increase in speed can increase the voltage beyond this level. This is crucial because the downstream A to D (Analog to Digital) converter puts out a 3-bit code, **ABC**, 2^3 or 8-codes, but we only have five lamps. **A** is the most significant bit, **C** the least significant bit.