Module 3 - Karnaugh Map

By the end of this module, you will be able to:

1. Understand the purpose of a Karnaugh Map (K-Map) as a visual tool for simplifying digital logic.
2. Correctly create a K-Map for functions with 2, 3, or 4 input variables.
3. Translate a Boolean function or a truth table into its corresponding K-Map representation.
4. Apply the rules for grouping adjacent cells to find the simplest possible logic expression.
5. Use "don't care" conditions (X) effectively to achieve better simplification.
6. Derive the simplified Sum-of-Products (SOP) boolean function from a completed K-Map.
7. Recognize that a simplified function results in a more efficient electronic circuit with fewer logic gates.

Objective

By the end of this module, you will be able to:

1. Understand the purpose of a Karnaugh Map (K-Map) as a visual tool for simplifying digital logic.
2. Correctly create a K-Map for functions with 2, 3, or 4 input variables.
3. Translate a Boolean function or a truth table into its corresponding K-Map representation.
4. Apply the rules for grouping adjacent cells to find the simplest possible logic expression.
5. Use "don't care" conditions (X) effectively to achieve better simplification.
6. Derive the simplified Sum-of-Products (SOP) boolean function from a completed K-Map.
7. Recognize that a simplified function results in a more efficient electronic circuit with fewer logic gates.

What is a Karnaugh Map and Why Do We Use It?

In digital electronics, we often start with complex Boolean functions that describe how a circuit should behave. A complex function requires many logic gates (like AND, OR, NOT) to build. More gates mean the circuit is more expensive, consumes more power, and can be slower.

The Karnaugh Map (K-Map) is a graphical method used to simplify these complex Boolean functions. Think of it as a visual tool or a map that helps us see the simplest possible form of a function. By simplifying the function, we can build the same circuit with fewer gates, making it cheaper and more efficient. The K-Map is an easier and faster alternative to using Boolean algebra rules for simplification.

The Structure of a K-Map

A K-Map is a table made of cells or boxes. Each cell represents one possible combination of inputs from a truth table.

The total number of cells is 2^n, where n is the nu

Variabel		Y	
		0	1
X	0		
	1		

- For 2 variables (X, Y), we need $2^2 = 4$ cells.

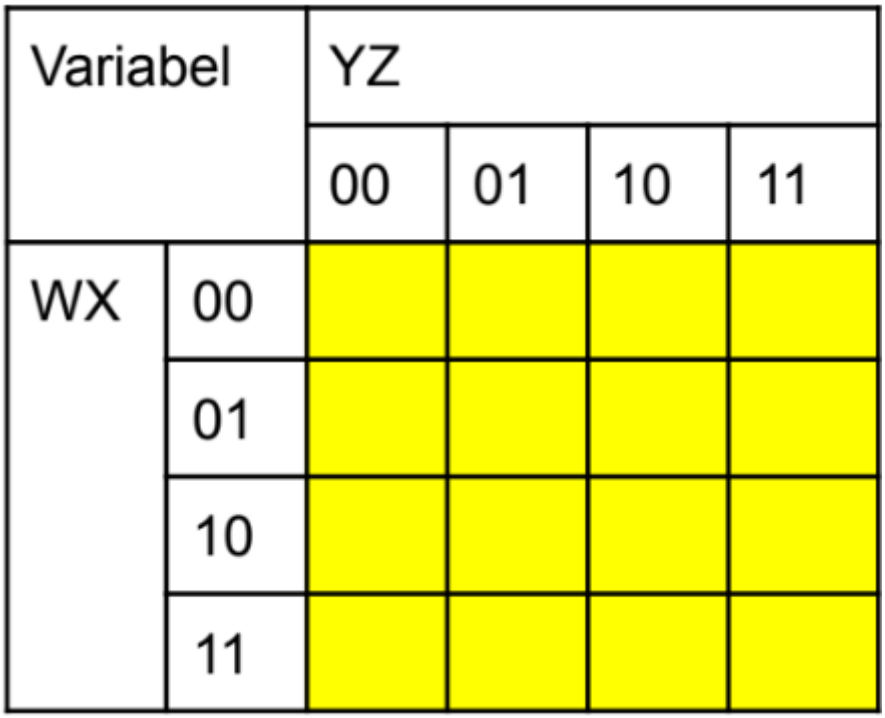
.

- For 3 variables (X, Y, Z), we need $2^3 = 8$ cells.

Variabel		YZ			
		00	01	10	11
X	0				
	1				

.

- For 4 variables (W, X, Y, Z), we need $2^4 = 16$ cells.



A special rule for K-Maps is the way the rows and columns are labeled. The binary numbers are ordered so that only one bit changes between any two adjacent cells. This is why the labels are `00, 01, 11, 10` instead of the usual `00, 01, 10, 11`. This special ordering is the key that makes simplification possible.

This ordering also means the map "wraps around." The far-right column is considered adjacent to the far-left column, and the top row is adjacent to the bottom row.

How to Simplify a Function Using a K-Map

We will focus on the `Sum-of-Products (SOP)` method, which involves looking for `1s` in the map.

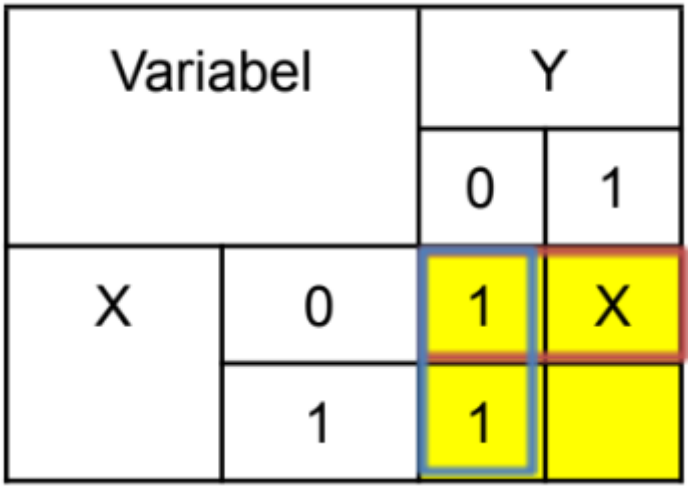
Step 1: Create and Fill the Map
Draw the correct K-Map for your number of variables. Look at your function's truth table or list of minterms. Place a `1` in every cell that corresponds to an output of `1`. If there are "don't care" conditions, place an `X` in those cells. Leave all other cells blank (or you can think of them as `0`s).

Variabel		Y	
		0	1
X	0	1	X
	1	1	

Step 2: Group the 1s
This is the most important step. You need to draw loops around groups of adjacent `1`s. Follow these rules:

- **Group Size:** Groups must contain a power-of-two number of cells (1, 2, 4, 8, or 16). You cannot have a group of 3, 5, or 6 cells.
- **Adjacency:** You can only group cells that are adjacent, either horizontally or vertically. Remember the "wrap-around" rule for the edges.
- **Make Groups as Large as Possible:** Always try to make the biggest groups you can. A single group of four is better than two separate groups of two.
- **Cover All 1s:** Every 1 on the map must be included in at least one group. A 1 can be part of multiple groups if it helps to make other groups larger.
- **Use Fewest Groups:** Your final goal is to cover all the 1s using the smallest number of groups possible.

- **Using "Don't Cares" (X):** You can include an X in a group if it helps you make a larger group of 1s. If an X doesn't help create a bigger group, just ignore it and treat it as a 0.



Step 3: Write the Simplified Function
Each group you created will become one term in your final simplified function. To find the term for each group:

1. Look at the variables along the rows and columns for that group.
2. Find the variable(s) that do not change their value inside the group.
3. If a variable stays as `1` for the entire group, include it as is (e.g., `A`).
4. If a variable stays as `0` for the entire group, include it with a NOT (e.g., `A'`).
5. If a variable changes its value (it is both `0` and `1`) within the group, it is eliminated from that term.

The final simplified function is the `OR (sum)` of all the terms you derived from each group.