K-Map 2

Lesson Objectives:

In this lesson you will learn:

1. The difference between prime implicants and essential prime implicants.
2. How to get a minimized POS function using a K-map.
3. How to minimize a combinational circuit that is not completely specified (has don't care conditions).
4. How to make a 5 and 6 variable K-map given a truth table or a SOP representation.

Definitions/Notations:

A product term of a function is said to be an **implicant**.

A **Prime Implicant** (**PI**) is a product term obtained by combining the maximum possible number of adjacent *1-squares* in the map.

If a minterm is covered only by one prime implicant then this prime implicant is said to be an **Essential Prime Implicant** (**EPI**).

Examples: **(see authorware version)**

POS Simplification:

Until now we have derived simplified Boolean functions from the maps in SOP form. Procedure for deriving simplified Boolean functions **POS** is slightly different. Instead of making groups of **1's**, make the groups of **0's**.

Since the simplified expression obtained by making group of 1's of the function (say F) is always in SOP form. Then the simplified function obtained by making group of 0's of the function will be the complement of the function (i.e., F') in SOP form.

Applying DeMorgan's theorem to F' (in SOP) will give F in POS form.

Examples: **(see authorware version)**

Don't Care Conditions:

In some cases, the function is not specified for certain combinations of input variables as 1 or 0.

There are two cases in which it occurs:
1. The input combination never occurs.
2. The input combination occurs but we do not care what the outputs are in response to these inputs.

In both cases, the outputs are called as **unspecified** and the functions having them are called as incompletely specified functions.

In most applications, we simply do not care what value is assumed by the function for unspecified minterms.

Unspecified minterms of a function are called as ***don't care conditions***. They provide further simplification of the function, and they are denoted by X's to distinguish them from 1's and 0's.

In choosing adjacent squares to simplify the function in a map, the don't care minterms can be assumed either 1 or 0, depending on which combination gives the simplest expression.

A don't care minterm need not be chosen at all if it does not contribute to produce a larger implicant.

Five-Variable K-Maps:

There are **32** minterms for a Boolean function with **five-variables**. Hence, Five-variable map consists of 32 squares.

It consists of **2** four-variable maps. Variable **A** distinguishes between the two maps, as indicated on the top of the diagram. The left-hand four-variable map represents the 16 squares where **A=0**, and the other four-variable map represents the squares where **A=1**.

Minterms **0** through **15** belong to the four-variable map with **A=0** and minterms **16** through **31** belong to the four-variable map with **A=1**.

	A'				A			
BC \ DE	00	01	11	10	00	01	11	10
00	m_0	m_1	m_3	m_2	m_{16}	m_{17}	m_{19}	m_{18}
01	m_4	m_5	m_7	m_6	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{12}	m_{13}	m_{15}	m_{14}	m_{28}	m_{29}	m_{31}	m_{30}
10	m_8	m_9	m_{11}	m_{10}	m_{24}	m_{25}	m_{27}	m_{26}

Each four-variable map retains the previously defined adjacency when taken separately. In addition, each square in the A=0 map is adjacent to the corresponding square in the A=1 map. For example, minterm 4 is adjacent to minterm 20 and minterm 15 to 31.

The best way to visualize this new rule for adjacent squares is to consider the two half maps as being one on top of the other. Any two squares that fall one over the other are considered adjacent.

Six-Variable K-Maps:

There are 64 minterms for a Boolean function with six-variables. Hence, Six-variable map consists of 64 squares.

By following the procedure used for the five-variable map, it is possible to construct a six-variable map with 4 four-variable maps to obtain the required 64 squares.