ITM1010
Computer and Communication Technologies

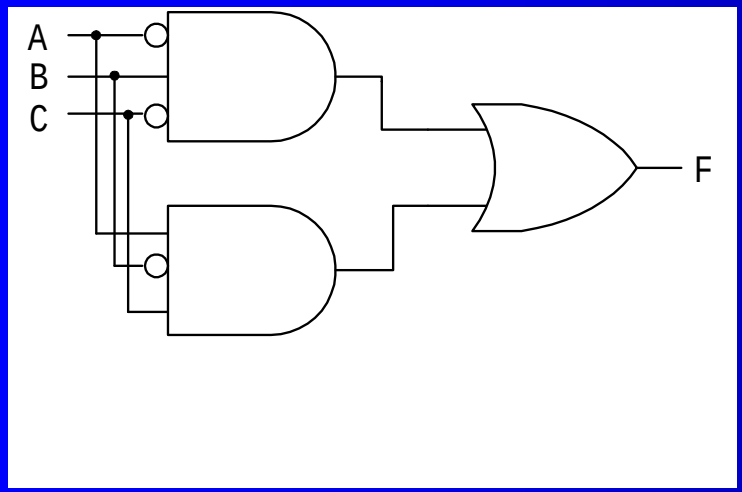
Lecture #4

Part I: Introduction to Computer Technologies

Logic Circuit Design & Simplification

Logic function implementation - SOP

This is usually called a sum-of-products (SOP) configuration.



$$F = \overline{A}B\overline{C} + A\overline{B}C$$

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

Product-Of-Sum (POS) Configuration

Product Of Sum obtained from truth table by making use of DeMorgan:

- OR (sum) the complemented inputs needed to get a low output in the truth table and AND (multiply) all such sums together

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

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

Product-Of-Sum (POS) Configuration

In the POS extraction, each variable in a set of input variables that produces a low output is ORed together with other variables in that set. Afterward, each set of ORed variables that produce a low output is ANDed with other sets that produce low outputs. Extraction of the variables that produce low outputs is done in complementary form.

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

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

SOP Extraction vs. POS Extraction

- When an SOP expression is extracted from a truth table, the expression is written to represent each high output condition. This is because the OR gate will output high when any of the sets of input variables produces a high output from the AND gates.
- When a POS expression is extracted from a truth table, the expression is written to represent every low output condition on the truth table. This is because the AND gate will output low when any of the sets of input variables produce a low output from the OR gates.
- Using DeMorgan's theorem, expressions for SOP and POS are proved to be equal.

Some Definitions

- Minterm: product term containing all input variables of a function in either true or complementary form

  e.g. F= ABC

- Maxterm: sum term containing all input variables of a function in either true or complementary form

  e.g. F=A+B+C

- Canonical Form: a function expressed in either fully minterms or fully maxterms
- Literal: each occurrence of a variable of a function in either true or complementary form

Design Minimization

- Reduce Hardware
- Reduce Number of Inputs

May be realized in Boolean expression by having

- minimum number of terms
- minimum number of literals

Design Minimization using Boolean Algebra

Example

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

no. of terms = 6

no. of literals = 18

The above expression may be simplified using Boolean algebra to:

$$F = \overline{A}B\overline{C} + A\overline{B}C$$

Karnaugh Map (K-Map)

- A Karnaugh map (K-map) is a pictorial method used to minimize Boolean expressions without having to use Boolean algebra theorems and equation manipulations. A K-map can be thought of as a special version of a truth table.
- Using a K-map, expressions with two to four variables are easily minimized. Expressions with five to six variables are more difficult but achievable, and expressions with seven or more variables are extremely difficult (if not impossible) to minimize using a K-map.

Simplification using K-Map

Truth Table

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

2-input K-map

	$\overline{B}$	B
$\overline{A}$	1	1
A	0	0

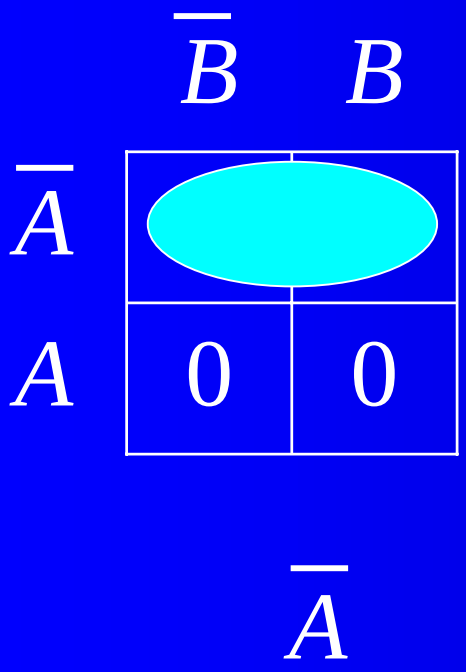
K-map simplification

	$\overline{B}$	B
$\overline{A}$		
A	0	0

$$F = \overline{A}\,\overline{B} + \overline{A}B \quad \Longrightarrow \quad \overline{A}$$

$$\overline{A}\,\overline{B} + \overline{A}B = \overline{A}\,(\overline{B} + B) = \overline{A}\,(1) = \overline{A}$$

Simplification using K-Map

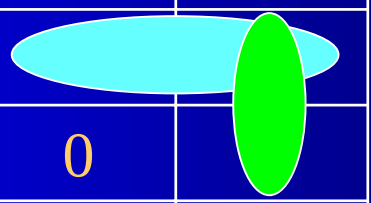


Any expression plotted on a K-map may be simplified by looping horizontally and/or vertically adjacent 1s. As shown on the right, once the looping has been completed, all complementary variables can be eliminated from the original expression. This will result in a simplified, yet equivalent expression. The *A* that is horizontally adjacent to the loop stays in the output expression. Since *B* and *B* appear vertically adjacent to the horizontal loop, they may be eliminated.

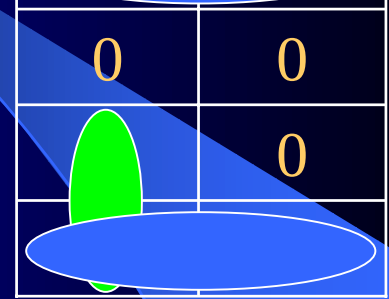
3-input K-map

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

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	0	0
$\overline{A}B$		
$A\overline{B}$	0	
AB	0	0



	$\overline{C}$	C
$\overline{A}\,\overline{B}$		
$\overline{A}B$	0	0
$A\overline{B}$		0
AB		



Layout of a 3-input K-map based on Gray Code

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

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

$\overline{A}B+BC(1)$

$\overline{A}B+BC$

$\overline{A}\,\overline{B}+A\overline{B}+AB\overline{C}+A\overline{B}\,\overline{C}$

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

$\overline{B}(1)+A\overline{C}(1)$

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

4-input K-map example

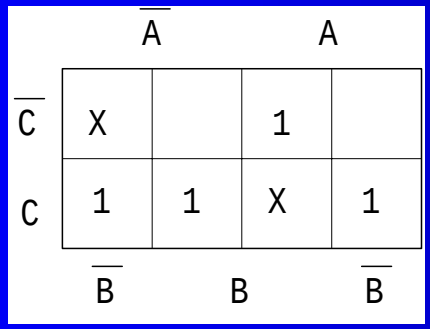
$$BD + C\bar{D}$$

K-Map Minimization Guideline

- Loop all isolated 1s;
- Consider each remaining 1 separately. If it can be looped in more than one way, try include it in the largest possible loop;
- A minimal solution is derived as soon as all 1s are covered. In the process of making the largest loop, it is permissible to use previously covered 1s.

Don't Care Condition in K-Map

Certain combinations of inputs may be immaterial to a given function. For such don't care states the output is irrelevant and may be 1 or 0.



X – don't care input condition

If we use X=0, $F = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$

This can be simplified (with X=0) to $F = \overline{A}C + A\overline{B}C + AB\overline{C}$

With K-map, 1 can be assigned to any don't care position to

- form largest possible loop
- Combine isolated 1 to form a loop
- In the example we can further simplify the expression by taking the bottom X as 1: $F = C + AB$

Design Example: Majority Detector

Design a 4-input circuit that will function as a majority detector. The circuit should output high when a majority of the inputs are high.

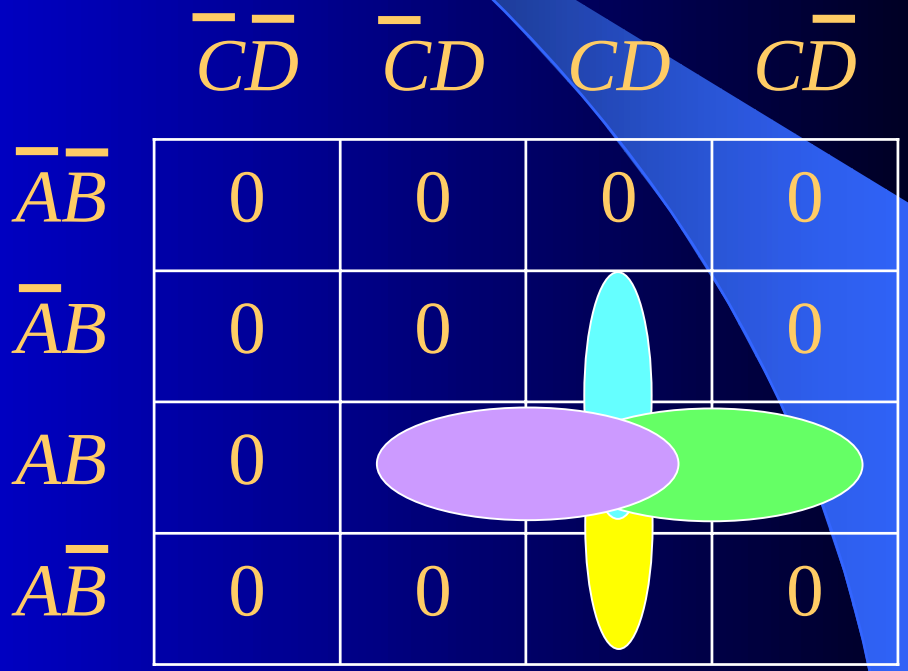
The first step is to complete a truth table and mark high outputs for every set of input conditions that contains three or four (majority) 1s. This is shown in the truth table on the right.

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0

Design Example: Majority Detector

A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Next, plot the Boolean expression from the truth table on a K-map as shown and simplify the expression.



$$X = ABD + BCD + ABC + ACD$$

Design Example: Majority Detector

The simplified expression is shown on the right, which has four terms since four pairs of 1s can be looped on the K-map.

Implementation is straightforward as shown on the right because this is an SOP expression. The circuit requires four 3-input AND gates and one 4-input OR gate.

$$X = ABD + BCD + ABC + ACD$$