Karnaugh Maps

Maurice Karnaugh is an American physicist, famous for the Karnaugh map used in Boolean algebra.

Karnaugh worked at Bell Labs (1952-66), developing the Karnaugh map (1954) as well as patents for PCM encoding

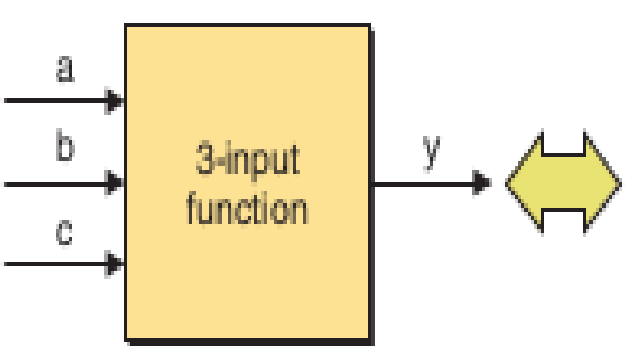
The Karnaugh map comprises a box for every line in the truth table

The binary values above the boxes are those associated with the a and b inputs.

Unlike a truth table, in which the input values typically follow a binary sequence, the Karnaugh map's input values must be ordered such that the values for adjacent columns vary by only a single bit: for example, 00, 01, 11, and 10.

This ordering is known as a Gray code, and it is a key factor with regard to the way in which Karnaugh maps work

MINIMIZATION USING KARNAUGH MAPS

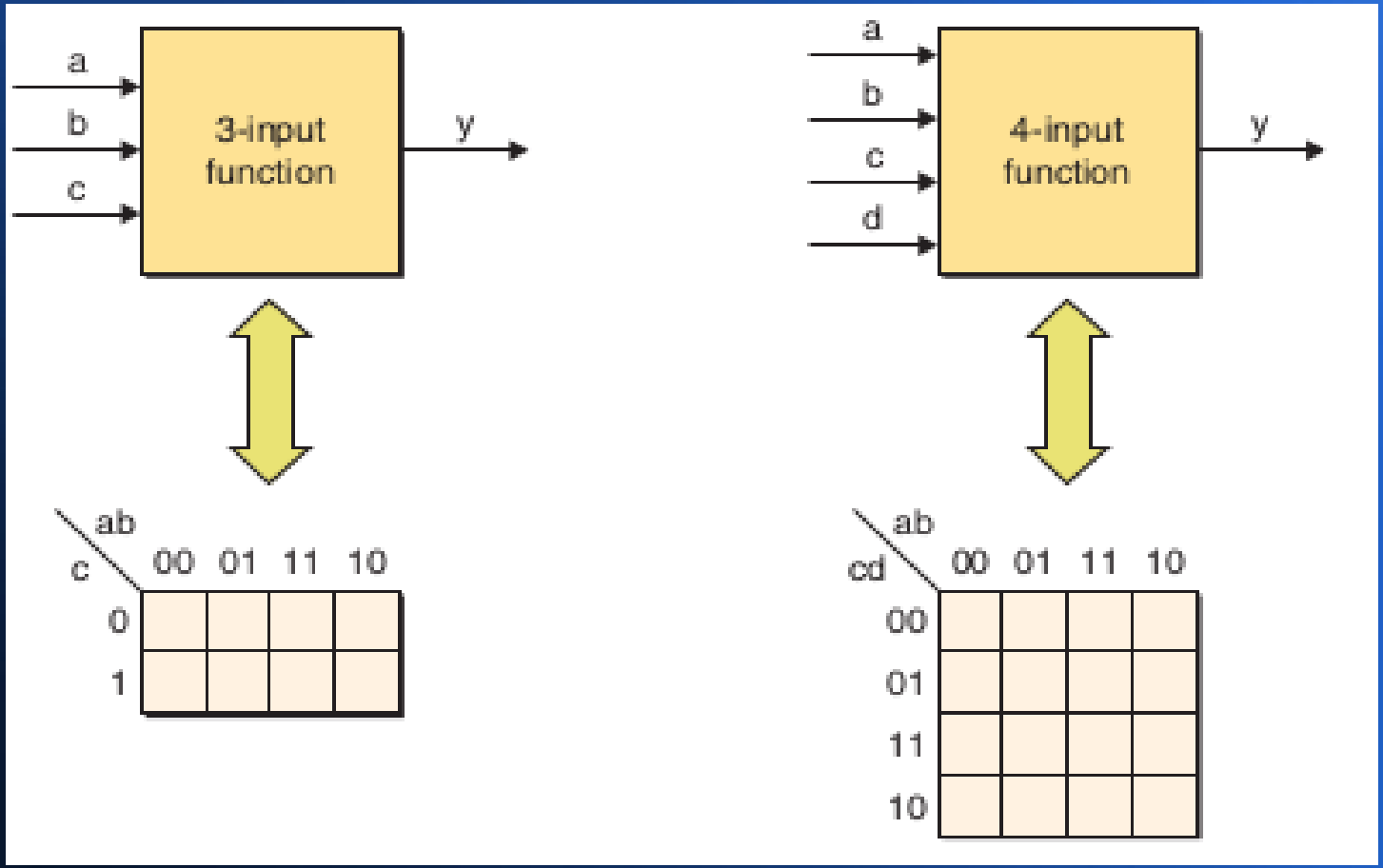


a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

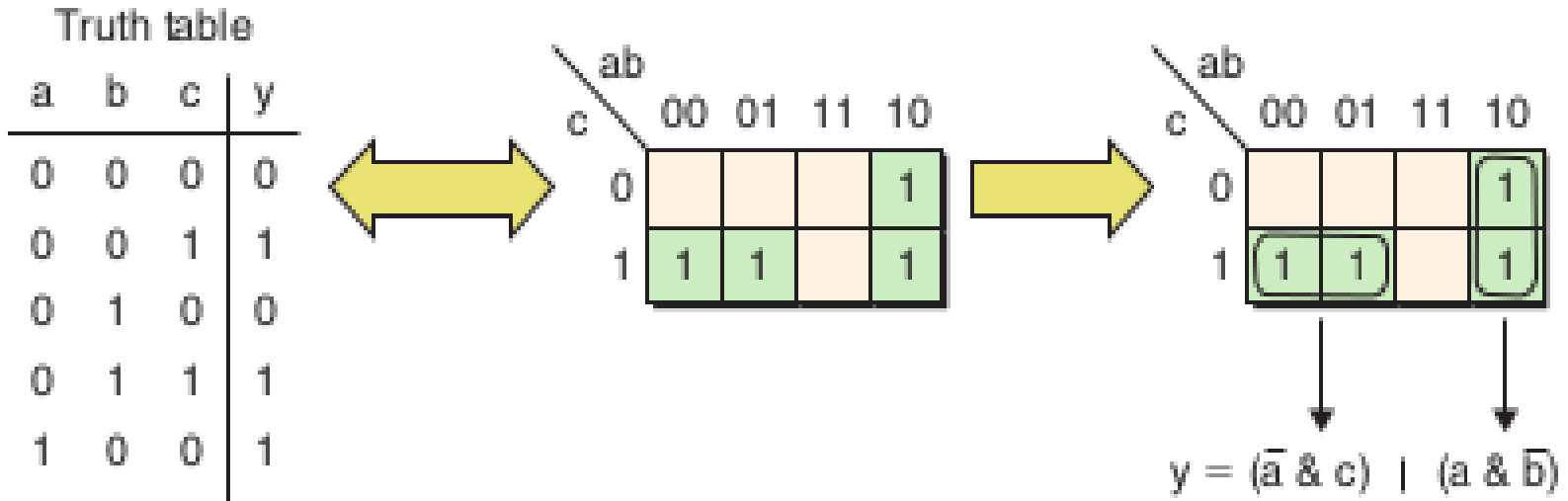
$y = (\bar{a} \,\&\, \bar{b} \,\&\, c) \mid (\bar{a} \,\&\, b \,\&\, c) \mid (a \,\&\, \bar{b} \,\&\, \bar{c}) \mid (a \,\&\, \bar{b} \,\&\, c)$

Sum-of-products expression

Generic Karnaugh maps for 3- and 4-input functions



Karnaugh map minimization of example 3-input function



In the case of a 3-input Karnaugh map, any two horizontally or vertically adjacent minterms, each composed of three variables, can be combined to form a new product term composed of only two variables.

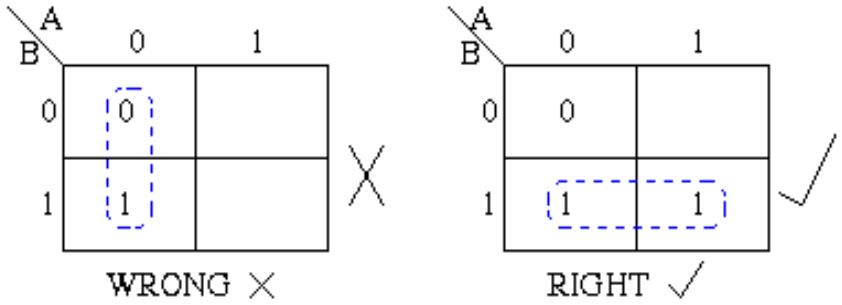
The 1s assigned to the Karnaugh map's boxes represent the same minterms as the 1s in the truth table's output column.

As the input values associated with each row and column in the map differ by only one bit, any pair of horizontally or vertically adjacent boxes corresponds to minterms that differ by only a single variable.

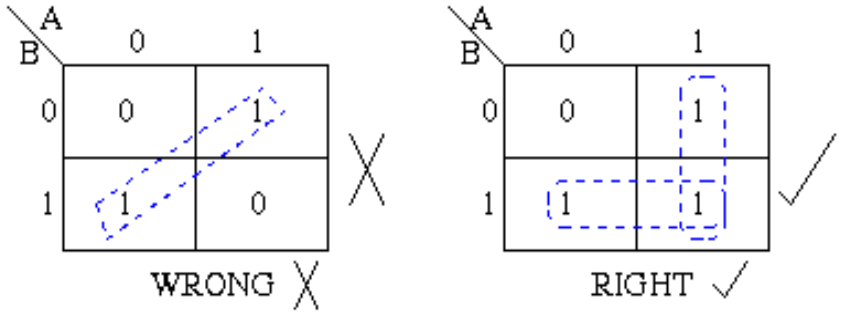
Such pairs of minterms can be grouped together and the variable That differs can be discarded, leaving a much-simplified equation

The Karnaugh map uses the following rules for the simplification of expressions by grouping together adjacent cells containing ones

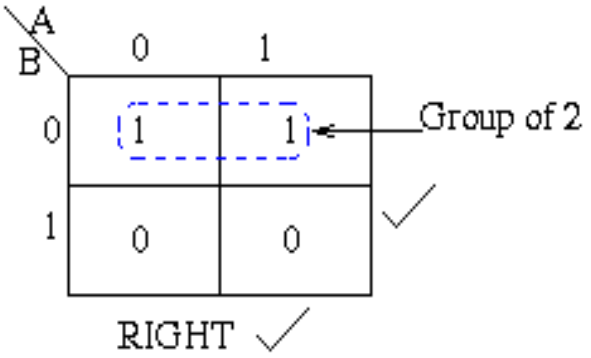
- Groups may not include any cell containing a zero



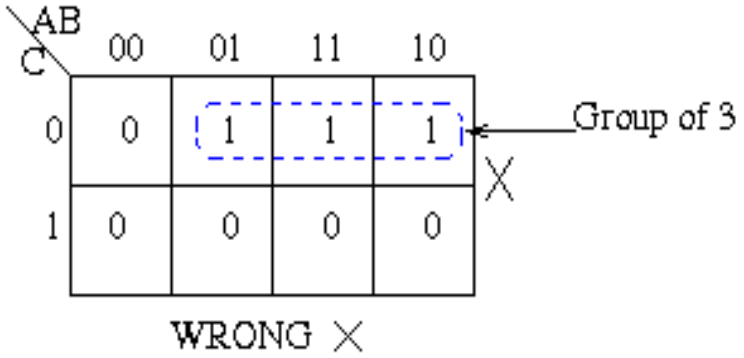
- Groups may be horizontal or vertical, but not diagonal.



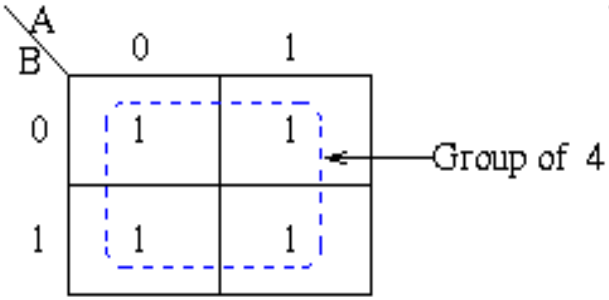
- Groups must contain 1, 2, 4, 8, or in general 2^n cells. That is if n = 1, a group will contain two 1's since $2^1 = 2$. If n = 2, a group will contain four 1's since $2^2 = 4$.



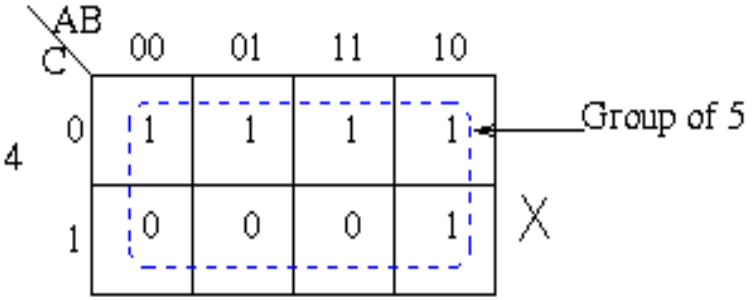
RIGHT ✓



WRONG ✗

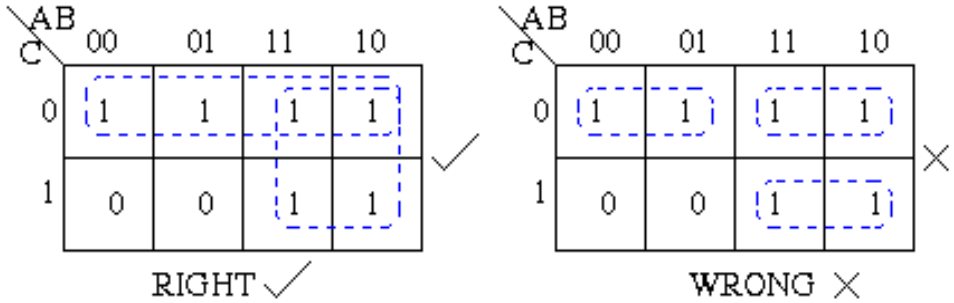


RIGHT ✓



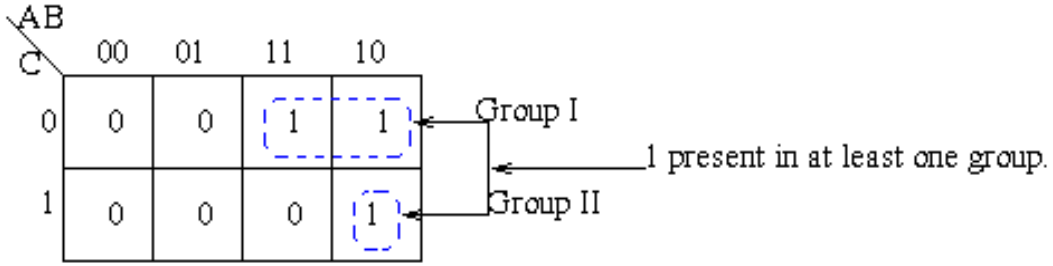
WRONG ✗

- Each group should be as large as possible.

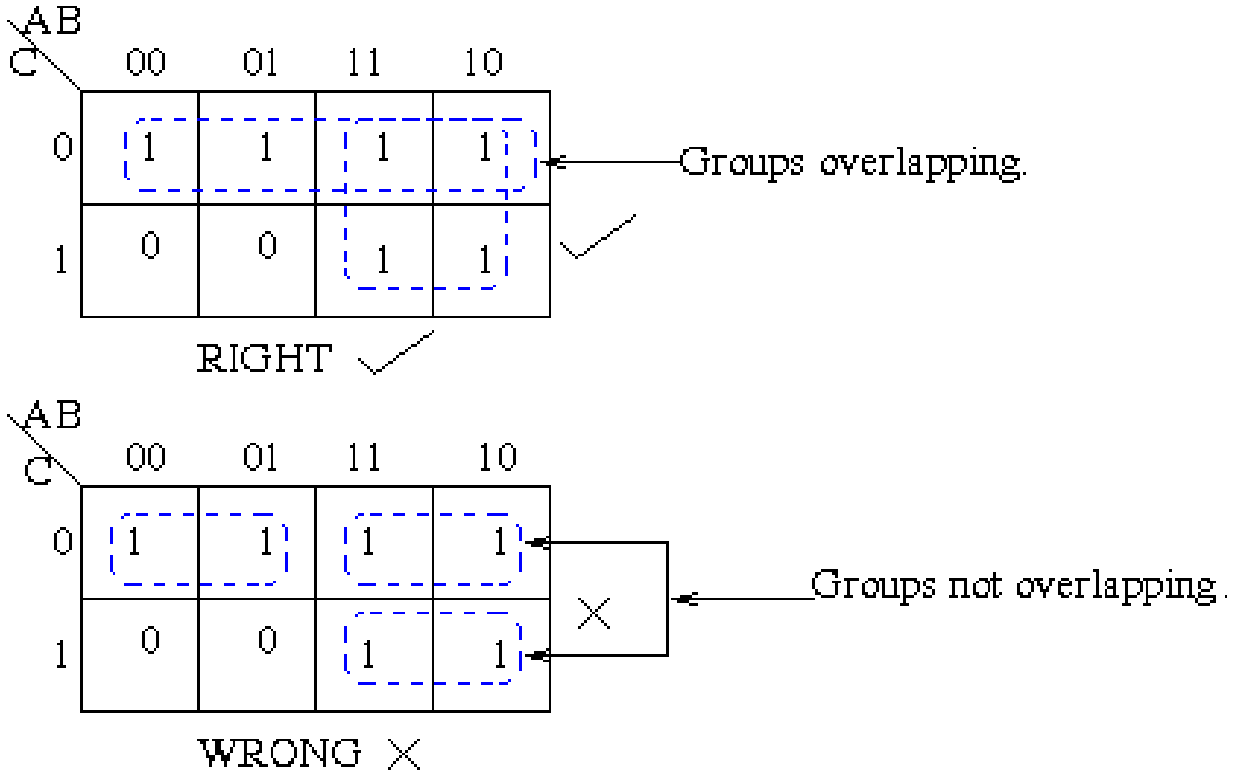


(Note that no Boolean laws broken,
but not sufficiently minimal)

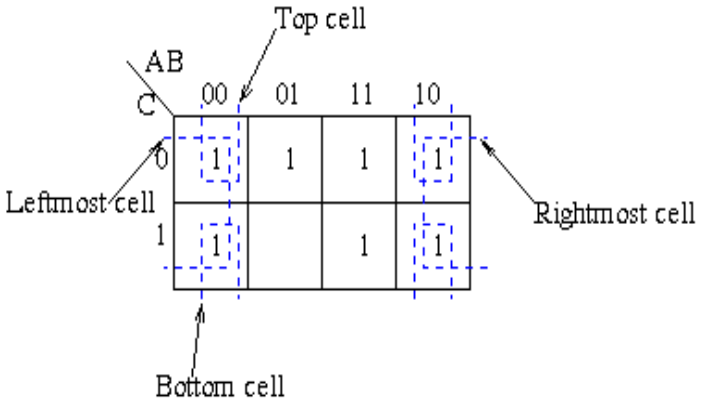
- Each cell containing a ***one*** must be in at least one group.



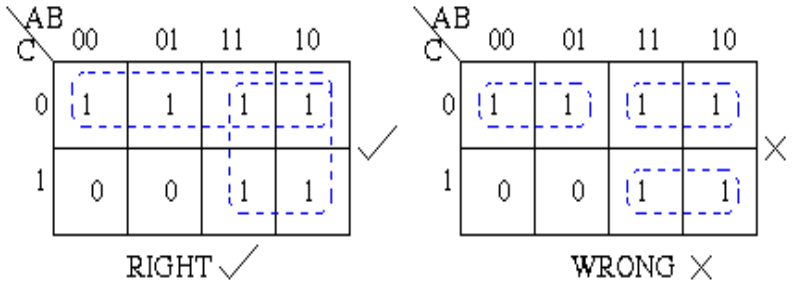
- Groups may overlap.



- Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



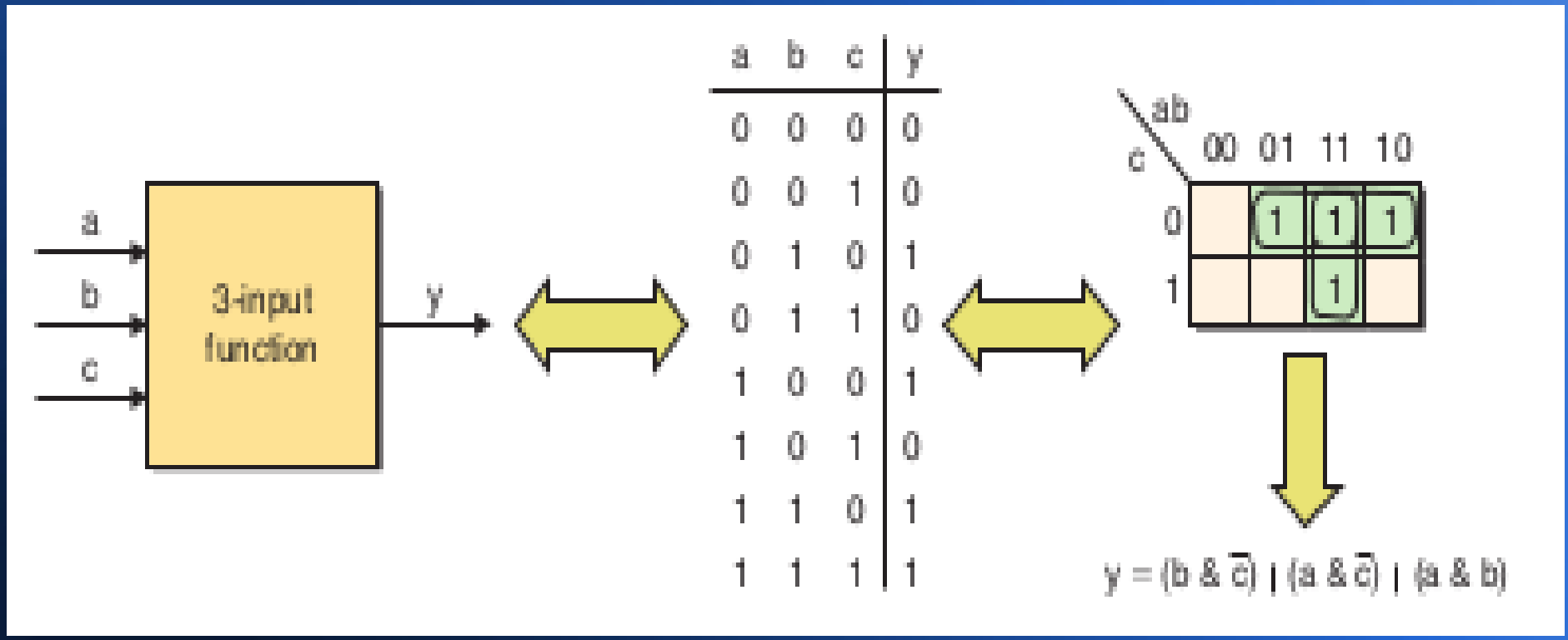
- There should be as few groups as possible, as long as this does not contradict any of the previous rules.



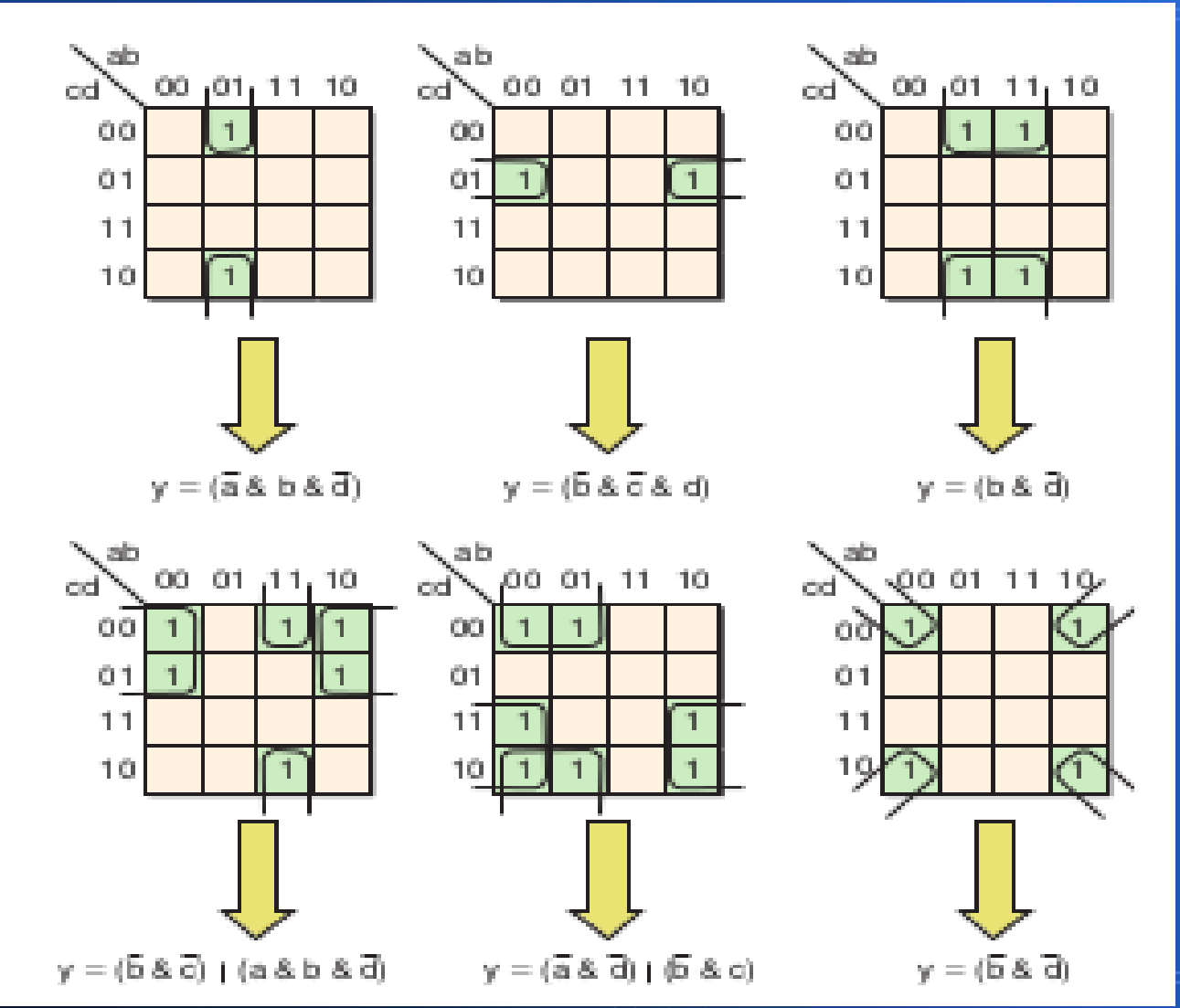
Summmary:

No zeros allowed.
No diagonals.
Only power of 2 number of cells in each group.
Groups should be as large as possible.
Every one must be in at least one group.
Overlapping allowed.
Wrap around allowed.
Fewest number of groups possible.

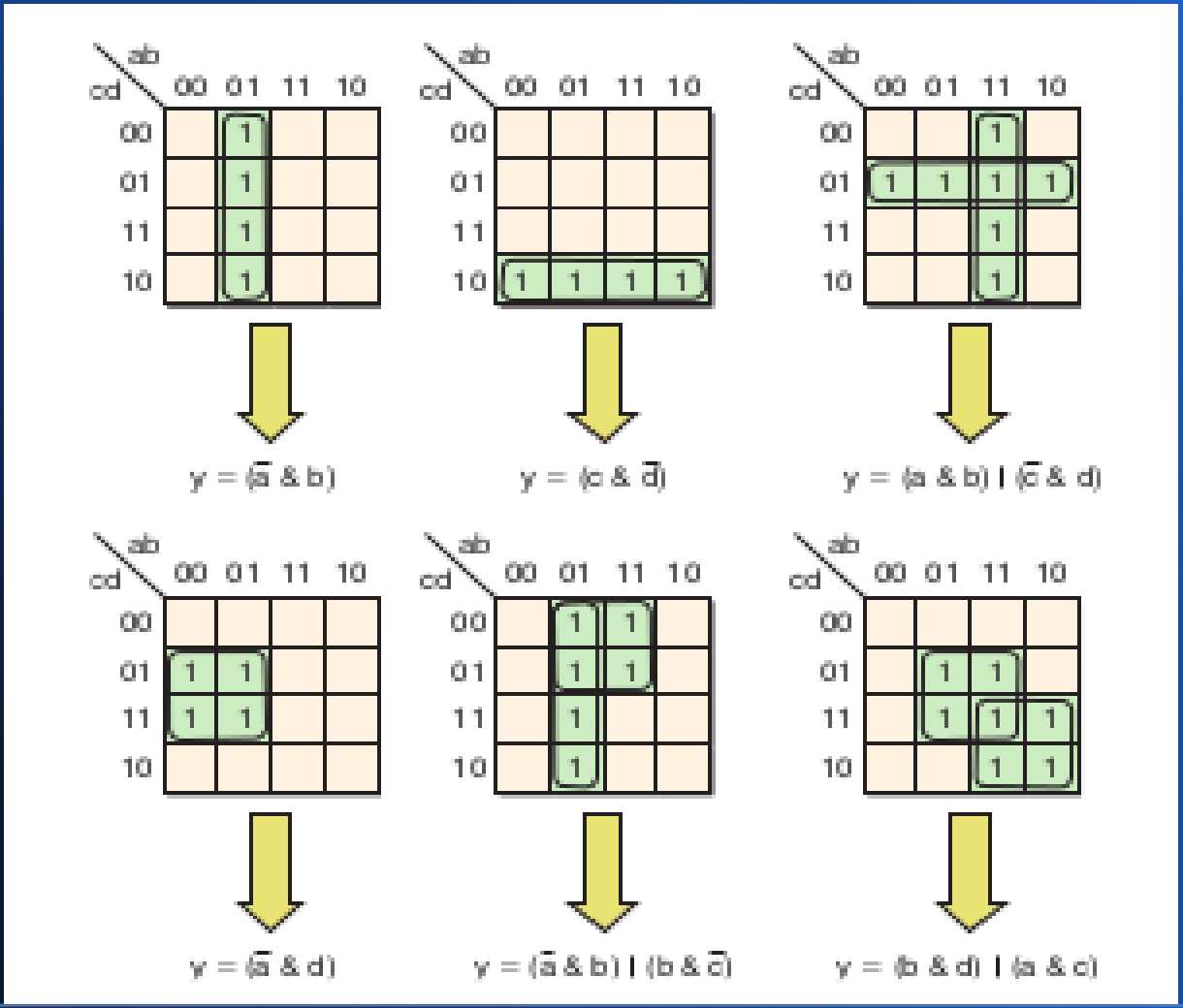
Using the same minterm to form multiple Karnaugh map groups



In the case of a 4-input map, any two adjacent minterms, each composed of four variables, can be combined to form a new product term composed of only three variables.



Karnaugh map groupings of four adjacent minterms.



Karnaugh map input values are ordered so that the values associated with adjacent rows and columns differ by only a single bit.

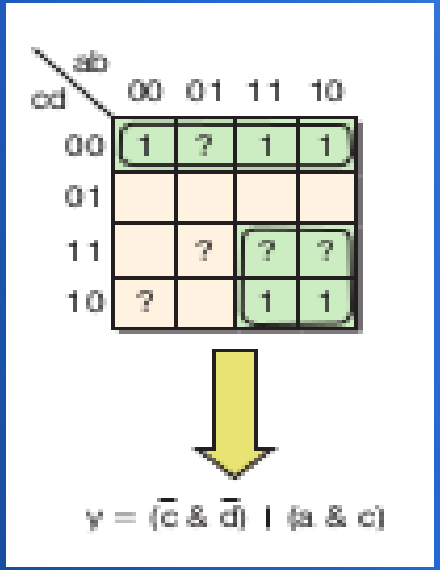
One result of this ordering is that the top and bottom rows are also separated by only a single bit (it may help to visualize the map rolled into a horizontal cylinder such that the top and bottom edges are touching).

Similarly, the left and right columns are separated by only a single bit (in this case it may help to visualize the map rolled into a vertical cylinder such that the left and right edges are touching). This leads To some additional groupings, a few of which are shown

INCOMPLETELY SPECIFIED FUNCTIONS

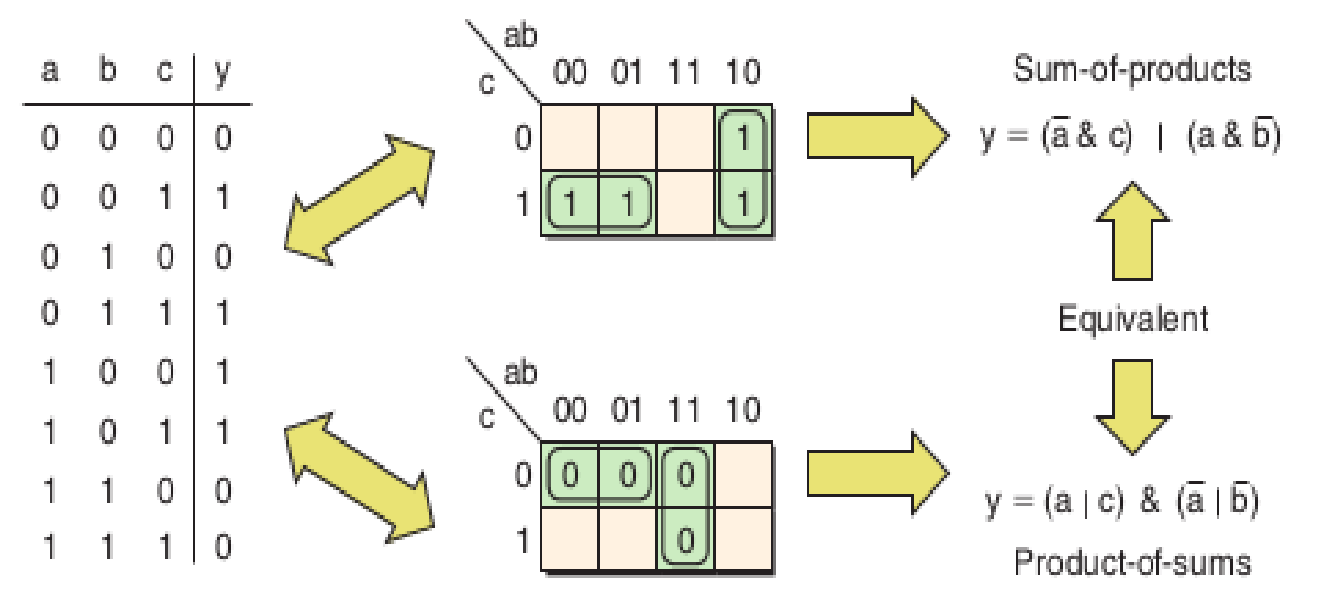
If, for example, the designer knows that certain input combinations will never occur, then the value assigned to the output for these combinations is irrelevant.

Alternatively, for some input combinations, the designer may simply not care about the value on the output.



The ? characters indicate don't care states, which can be considered to represent either 0 or 1 values at the designer's discretion

Populating Karnaugh maps with 0s versus 1s



Karnaugh maps are most often used to represent 3-input and 4-input functions. It is possible to create similar maps for 5-input and 6-input functions, but these maps can quickly become unwieldy and difficult to use

A switching circuit has two control inputs (C_1 and C_2), two data inputs (X_1 and X_2), and one output (Z). The circuit performs one of the logic operations AND, OR, EQU (equivalence), or XOR (exclusive OR) on the two data inputs. The function performed depends on the control inputs:

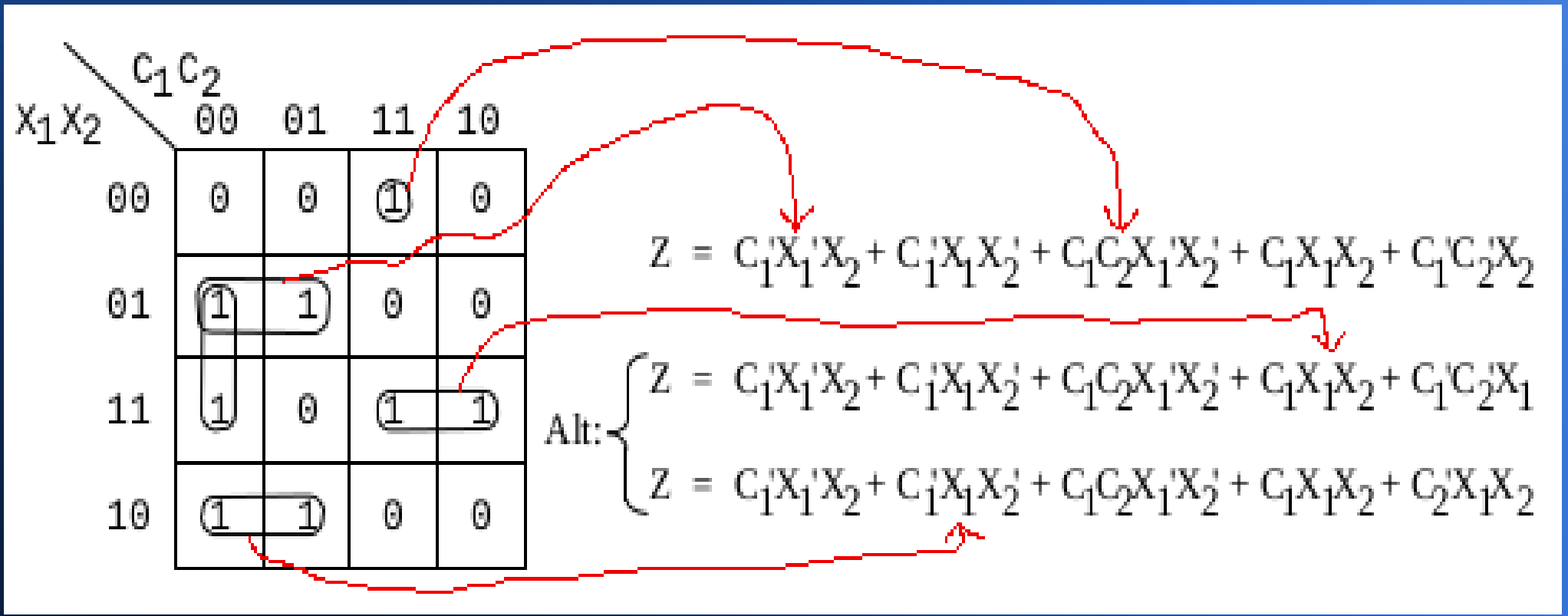
C_1	C_2	Function Performed by Circuit
0	0	OR
0	1	XOR
1	0	AND
1	1	EQU

(a) Derive a truth table for Z.
(b) Use a Karnaugh map to find a minimum AND-OR gate circuit to realize Z.

(a)

C_1	C_2	X_1	X_2	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

(b) $Z = C_1' X_1' X_2 + C_1' X_1 X_2' + C_1 X_1 X_2 + C_1 C_2 X_1' X_2' + \{C_1' C_2' X_2$ **or** $C_1' C_2' X_1$ **or** $C_2' X_1 X_2\}$


C1C2
X1X2
00 01 11 10
00 0 0 1 0
01 1 1 0 0
11 1 0 1 1
10 1 1 0 0
Z = C1'X1'X2 + C1'X1X2' + C1C2X1'X2' + C1X1X2 + C1'C2X2
Alt:
Z = C1'X1'X2 + C1'X1X2' + C1C2X1'X2' + C1X1X2 + C1'C2X1
Z = C1'X1'X2 + C1'X1X2' + C1C2X1'X2' + C1X1X2 + C2X1X2