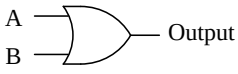
Karnaugh mapping

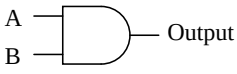
Resources and methods for learning about these subjects (list a few here, in preparation for your research):

Question 1

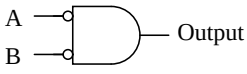
Identify each of these logic gates by name, and complete their respective truth tables:



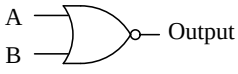
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



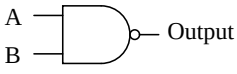
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



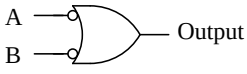
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



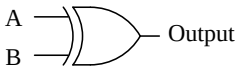
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



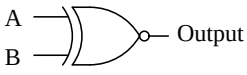
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



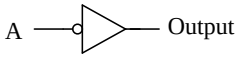
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

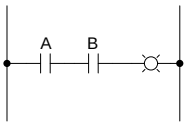


A	Output
0	1
1	0

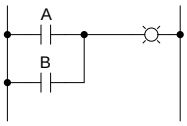
file 01249

Question 2

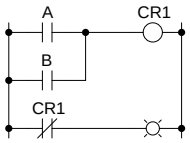
Identify each of these relay logic functions by name (AND, OR, NOR, etc.) and complete their respective truth tables:



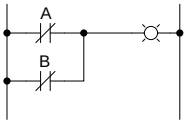
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



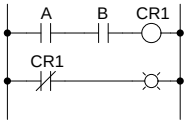
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



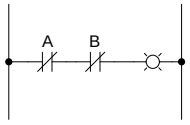
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



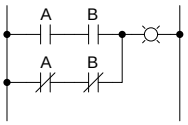
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



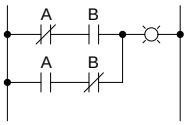
A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



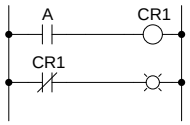
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



A	Output
0	1
1	0

file 01335

Question 3

A *Karnaugh map* is nothing more than a special form of truth table, useful for reducing logic functions into minimal Boolean expressions.

Here is a truth table for a specific three-input logic circuit:

A	B	C	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Complete the following Karnaugh map, according to the values found in the above truth table:

AB \ C	0	1
00	1	1
01	0	1
11	0	0
10	0	1

file 02834

Question 4

A *Karnaugh map* is nothing more than a special form of truth table, useful for reducing logic functions into minimal Boolean expressions.

Here is a truth table for a specific four-input logic circuit:

A	B	C	D	Out
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Complete the following Karnaugh map, according to the values found in the above truth table:

AB \ CD	00	01	11	10
00	0	1	0	0
01	0	1	0	0
11	0	1	1	0
10	0	1	1	1

file 01310

Question 5

Here is a truth table for a four-input logic circuit:

A	B	C	D	Out
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	1
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	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

If we translate this truth table into a Karnaugh map, we obtain the following result:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

Note how the only 1's in the map are clustered together in a group of four:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

If you look at the input variables (A, B, C, and D), you should notice that only two of them actually change within this cluster of four 1's. The other two variables hold the same value for each of these conditions where the output is a "1". Identify which variables change, and which stay the same, for this cluster.

file 01311

The variables A and C change.

The variables B and D stay the same.

Question 6

Here is a truth table for a four-input logic circuit:

A	B	C	D	Out
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	0
1	0	0	0	0
1	0	0	1	0
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	1

If we translate this truth table into a Karnaugh map, we obtain the following result:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

Note how the only 1's in the map all exist on the same row:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	0	0

If you look at the input variables (A, B, C, and D), you should notice that only two of them are constant for each of the "1" conditions on the Karnaugh map. Identify these variables, and remember them.

Now, write an SOP (Sum-of-Products) expression for the truth table, and use Boolean algebra to reduce that raw expression to its simplest form. What do you notice about the simplified SOP expression, in relation to the common variables noted on the Karnaugh map?

F(A, B, C) = A.B.C'.D' + A.B.C'.D + A.B.C.D' + A.B.C.D = A.B(C'.D' + C'.D + C.D' + C.D) = A.B(C'(D' + D) + C.D' + C.D) = A.B(C' + C.D' + C.D) = A.B(C' + D' + C.D) = A.B(C' + D' + D) = A.B(C' + 1) = A.B

We can observe that variables A and B are the only two inputs that remain constant for the four "1" conditions shown in the Karnaugh map. Thus, the SOP is in strong relations with the Karnaugh map.

file 02835

Question 7

One of the essential characteristics of Karnaugh maps is that the input variable sequences are always arranged in Gray code sequence. That is, you never see a Karnaugh map with the input combinations arranged in binary order:

Proper form

AB \ CD	00	01	11	10
00				
01				
11				
10				

Improper form

AB \ CD	00	01	10	11
00				
01				
10				
11				

The reason for this is apparent when we consider the use of Karnaugh maps to detect common variables in output sets. For instance, here we have a Karnaugh map with a cluster of four 1's at the center:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

Arranged in this order, it is apparent that two of the input variables have the same values for each of the four "high" output conditions. Re-draw this Karnaugh map with the input variables sequenced in binary order, and comment on what happens. Can you still tell which input variables remain the same for all four output conditions?

AB \ CD	00	01	10	11
00	0	0	0	0
01	0	1	0	1
10	0	0	0	0
11	0	1	0	1

file 01312

Looking at this, we can still tell that B & D are '1' for all four high output conditions, but this is not apparent by proximity as it was before.

Question 8

Examine this truth table and corresponding Karnaugh map:

A	B	C	D	Out
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

Though it may not be obvious from first appearances, the four "high" conditions in the Karnaugh map actually belong to the same group. To make this more apparent, I will draw a new (oversized) Karnaugh map template, with the Gray code sequences repeated twice along each axis:

AB \ CD	00	01	11	10	00	01	11	10
00	1	0	0	1	1	0	0	1
01	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
10	1	0	0	1	1	0	0	1
00	1	0	0	1	1	0	0	1
01	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
10	1	0	0	1	1	0	0	1

Fill in this map with the 0 and 1 values from the truth table, and then see if a grouping of four "high" conditions becomes apparent.

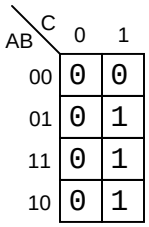
file 01342

Question 9

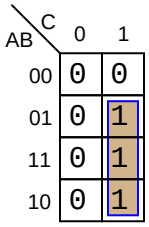
A student is asked to use Karnaugh mapping to generate a minimal SOP expression for the following truth table:

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Following the truth table shown, the student plots this Karnaugh map:



"This is easy," says the student to himself. "All the '1' conditions fall within the same group!" The student then highlights a triplet of 1's as a single group:



Looking at this cluster of 1's, the student identifies C as remaining constant (1) for all three conditions in the group. Therefore, the student concludes, the minimal expression for this truth table must simply be C.

However, a second student decides to use Boolean algebra on this problem instead of Karnaugh mapping. Beginning with the original truth table and generating a Sum-of-Products (SOP) expression for it, the simplification goes as follows:

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

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

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

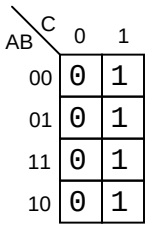
$$C(B + A\overline{B})$$

$$C(B + A)$$

$$AC + BC$$

Obviously, the answer given by the second student's Boolean reduction ($AC + BC$) does not match the answer given by the first student's Karnaugh map analysis (C).

Perplexed by the disagreement between these two methods, and failing to see a mistake in the Boolean algebra used by the second student, the first student decides to check his Karnaugh mapping again. Upon reflection, it becomes apparent that if the answer really were C, the Karnaugh map would look different. Instead of having three cells with 1's in them, there would be four cells with 1's in them (the output of the function being "1" *any* time $C = 1$:



Somewhere, there must have been a mistake made in the first student's grouping of 1's in the Karnaugh map, because the map shown above is the only one proper for an answer of C, and it is not the same as the real map for the given truth table. Explain where the mistake was made, and what the proper grouping of 1's should be. If we look at the Karnaugh map we can see the true value 3 times, thus we will group A with C (from 111 and 101) and B with C (from 011 and 111) in the end obtaining A.C + B.C

file 02836

Question 10

State the rules for properly identifying common groups in a Karnaugh map.

file 02837

Grouping:
Group adjacent 1s (ones) in powers of 2: 1, 2, 4, 8, etc. This means that groups can consist of one cell (single 1), two cells (adjacent 1s), four cells (2x2 block), eight cells (2x4 block), and so on.

Groups Should Be Rectangular:
Groups should always be rectangular in shape. This ensures that they can be represented as a simplified Boolean expression.

Groups Should Be as Large as Possible:
Create groups that cover as many 1s as possible. Larger groups generally lead to simpler Boolean expressions.

Each 1 Must Be Covered:
Every 1 in the truth table must be covered by at least one group. If a 1 is left uncovered, it will not be included in the simplified expression.

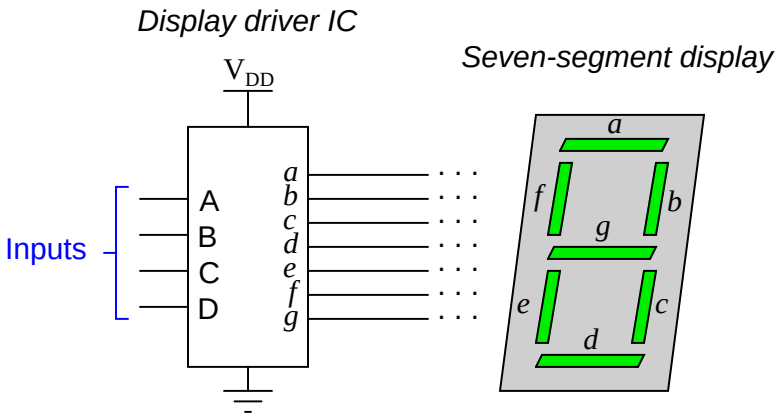
Groups Should Not Overlap:
Groups should not overlap; each cell in the K-map should be included in only one group. Overlapping groups can lead to ambiguity in the simplified expression.

Include Don't-Care Conditions:
If the truth table includes don't-care conditions, they can be used to simplify the expression further. Don't-care conditions are cells for which the output value doesn't matter, and they can be either 0 or 1.

Verify the Groups:
After grouping, verify that each group represents a valid minterm (a unique combination of inputs). Ensure that no cells are left ungrouped.

Question 11

A *seven segment decoder* is a digital circuit designed to drive a very common type of digital display device: a set of LED (or LCD) segments that render numerals 0 through 9 at the command of a four-bit code:



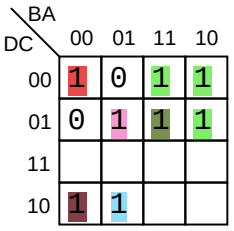
The behavior of the display driver IC may be represented by a truth table with seven outputs: one for each segment of the seven-segment display (a through g). In the following table, a "1" output represents an active display segment, while a "0" output represents an inactive segment:

D	C	B	A	a	b	c	d	e	f	g	Display
0	0	0	0	1	1	1	1	1	1	0	"0"
0	0	0	1	0	1	1	0	0	0	0	"1"
0	0	1	0	1	1	0	1	1	0	1	"2"
0	0	1	1	1	1	1	1	0	0	1	"3"
0	1	0	0	0	1	1	0	0	1	1	"4"
0	1	0	1	1	0	1	1	0	1	1	"5"
0	1	1	0	1	0	1	1	1	1	1	"6"
0	1	1	1	1	1	1	0	0	0	0	"7"
1	0	0	0	1	1	1	1	1	1	1	"8"
1	0	0	1	1	1	1	1	0	1	1	"9"

A real-life example such as this provides an excellent showcase for techniques such as Karnaugh mapping. Let's take output a for example, showing it without all the other outputs included in the truth table:

D	C	B	A	a
0	0	0	0	1
0	0	0	1	0
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	1
1	0	0	0	1
1	0	0	1	1

Plotting a Karnaugh map for output a, we get this result:

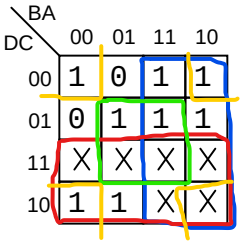


F(D, C, B, A) = D'.B + C'.B'.A' + D.C'.B' + D'.C.A

Identify adjacent groups of 1's in this Karnaugh map, and generate a minimal SOP expression from those groupings.

Note that six of the cells are blank because the truth table does not list all the possible input combinations with four variables (A, B, C, and D). With these large gaps in the Karnaugh map, it is difficult to form large groupings of 1's, and thus the resulting "minimal" SOP expression has several terms.

However, if we do not care about output a's state in the six non-specified truth table rows, we can fill in the remaining cells of the Karnaugh map with "don't care" symbols (usually the letter X) and use those cells as "wildcards" in determining groupings:



F(D, C, B, A) = C'.A' + D + B + C.A

With this new Karnaugh map, identify adjacent groups of 1's, and generate a minimal SOP expression from those groupings.

file 02838

Question 12

When designing a circuit to emulate a truth table such as this where nearly all the input conditions result in "1" output states, it is easier to use Product-of-Sums (POS) expressions rather than Sum-of-Products (SOP) expressions:

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Is it possible to use a Karnaugh map to generate the appropriate POS expression for this truth table, or are Karnaugh maps limited to SOP expressions only? Explain your answer, and how you were able to obtain it.

file 02839

C	0	1
AB 00	1	1
01	1	1
11	0	0
10	1	1

F(A, B, C) = (M6, M7) = (A' + B' + C)(A' + B' + C') = (A'+B'+C')A'+(A'+B'+C')B'+(A'+B'+C')C = A'A' + A'B' + A'C' + B'A' + B'B' + B'C' + CA' + CB' + CC' = A' + B' + CA' + CB' = A' + B'

From Karnaugh map u can see that F(A, B, C) = A' + B' if we group corectly the terms and this equals to the POS expression we solved above, thus we can say that: yes, you can use Karnaugh maps to generate POS expressions, not just SOP expressions!

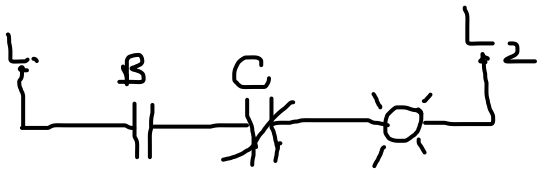
Question 13

Use a Karnaugh map to generate a simple Boolean expression for this truth table, and draw a relay logic circuit equivalent to that expression:

AB \ C	0	1
00	0	0
01	1	0
11	1	0
10	0	0

A	B	C	Output
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

F(A, B, C) = A'.B.C' + A.B.C' = B.C'(A' + A) =
= B.C'



file 02840

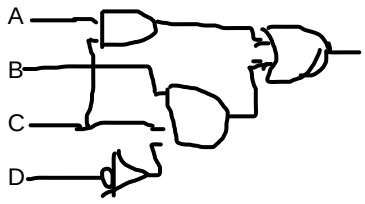
Question 14

Use a Karnaugh map to generate a simple Boolean expression for this truth table, and draw a gate circuit equivalent to that expression:

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	0	0	1	1
10	0	0	1	1

A	B	C	D	Output
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	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

F(A, B, C, D) = A'BCD' + AB'CD' + AB'CD + ABCD' + ABCD =
= ABC(D + D') + A'BCD' + AB'CD' + AB'CD =
= ABC + AC(B'D' + B'D) + A'BCD' =
= ABC +AB'C + A'BCD' =
= C(AB + AB' + A'BD') =
= C(A + A'BD') =
= C(A + BD') =
= CA + CBD' =
= AC + BCD'



file 02841

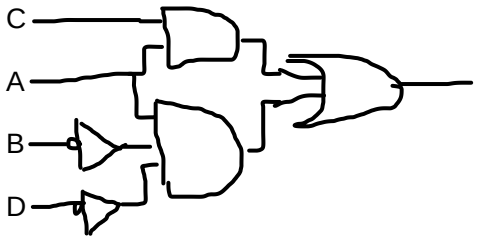
Question 15

Use a Karnaugh map to generate a simple Boolean expression for this truth table, and draw a gate circuit equivalent to that expression:

A	B	C	D	Output
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	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

CD 00 01 11 10
AB 00 0 0 0 0
01 0 0 0 0
11 0 0 1 1
10 1 0 1 1

F(A, B, C ,D) = AB'D' + AC



file 02842

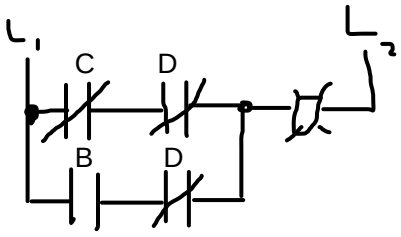
Question 16

Use a Karnaugh map to generate a simple Boolean expression for this truth table, and draw a relay circuit equivalent to that expression:

A	B	C	D	Output
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

CD 00 01 11 10
AB 00 1 0 0 0
01 1 0 0 1
11 1 0 0 1
10 1 0 0 0

F(A, B, C, D) = C'D' + BD'



file 02843

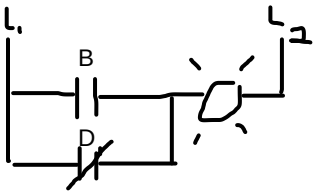
Question 17

Use a Karnaugh map to generate a simple Boolean expression for this truth table, and draw a relay circuit equivalent to that expression:

A	B	C	D	Output
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	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	1	1	1
11	1	1	1	1
10	1	0	0	1

F(A , B, C, D) = B + B'D' = B + D'



file 02844