EXAMPLES

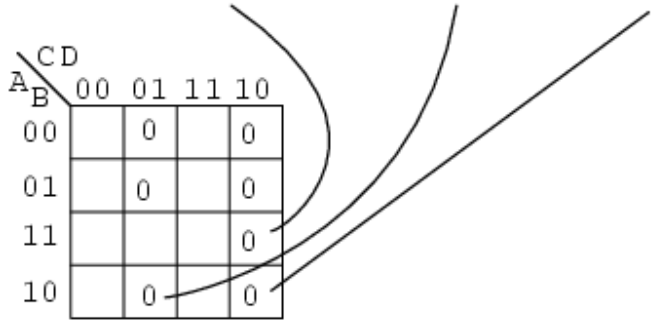
1.Simplify the Product-Of-Sums Boolean expression below, providing a result in POS form.

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

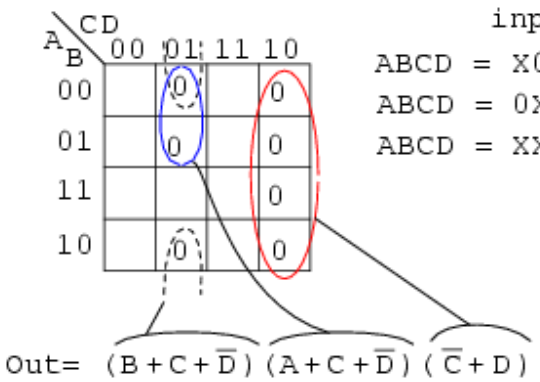
Solution:

Transfer the seven maxterms to the map below as **0**s. Be sure to complement the input variables in finding the proper cell location.

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



We map the **0**s as they appear left to right top to bottom on the map above. We locate the last three maxterms with leader lines.. Once the cells are in place above, form groups of cells as shown below. Larger groups will give a sum-term with fewer inputs. Fewer groups will yield fewer sum-terms in the result.



	input		complement		Sum-term
ABCD =	X001	>	X110	>	$(B+C+\overline{D})$
ABCD =	0X01	>	1X10	>	$(A+C+\overline{D})$
ABCD =	XX10	>	XX01	>	$(\overline{C}+D)$

We have three groups, so we expect to have three sum-terms in our POS result above. The group of 4-cells yields a 2-variable sum-term. The two groups of 2-cells give us two 3-variable sum-terms. Details are shown for how we arrived at the Sum-terms above. For a group, write the binary group input address, then complement it, converting that to the Boolean sum-term. The final result is product of the three sums.

Example:

2.Simplify the Product-Of-Sums Boolean expression below, providing a result in SOP form.

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

Solution:

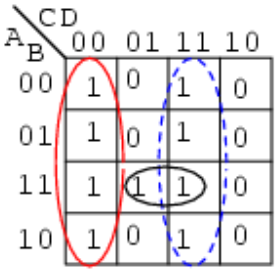
This looks like a repeat of the last problem. It is except that we ask for a Sum-Of-Products Solution instead of the Product-Of-Sums which we just finished. Map the maxterm **0**s from the Product-Of-Sums given as in the previous problem, below left.

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

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

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

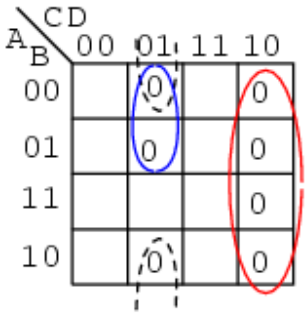
Then fill in the implied **1**s in the remaining cells of the map above right.

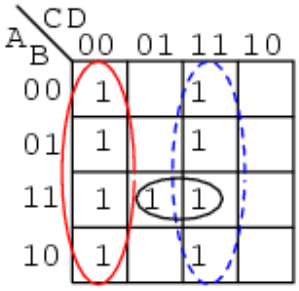


Out= $\overline{C}\overline{D}$ + CD+ ABD

Form groups of **1**s to cover all **1**s. Then write the Sum-Of-Products simplified result as in the previous section of this chapter. This is identical to a previous problem.

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



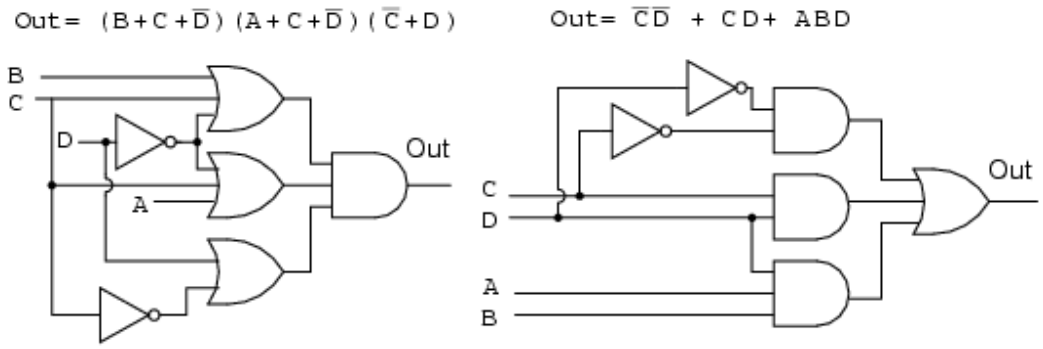


Out= $\overline{C}\overline{D}$ + CD+ ABD

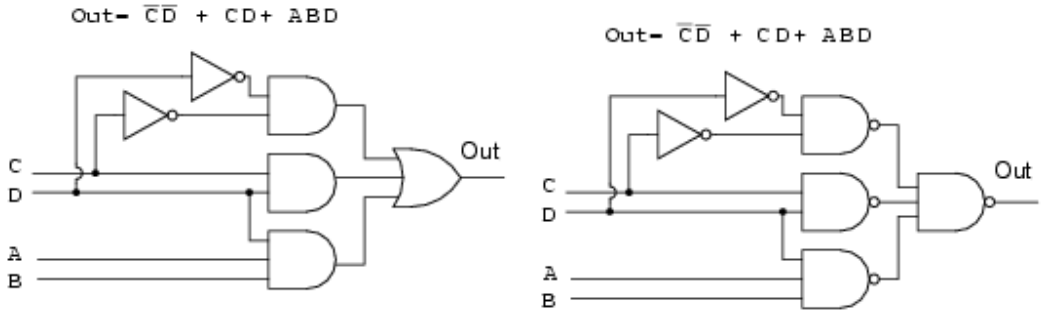
Out= $(B+C+\overline{D})(A+C+\overline{D})(\overline{C}+D)$

Above we show both the Product-Of-Sums solution, from the previous example, and the Sum-Of-Products solution from the current problem for comparison. Which is the simpler solution? The POS uses 3-OR gates and 1-AND gate, while the SOP uses 3-AND gates and 1-OR gate. Both use four gates each. Taking a closer look, we count the number of gate inputs. The POS uses 8-inputs; the SOP uses 7-inputs. By the definition of minimal cost solution, the SOP solution is simpler. This is an example of a technically correct answer that is of little use in the real world.

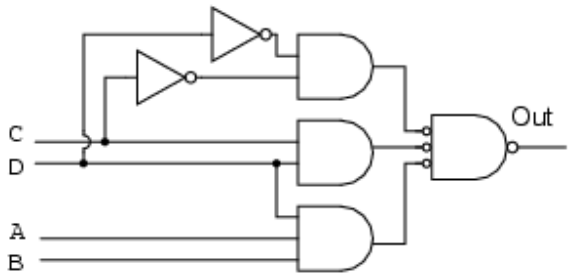
The better solution depends on complexity and the logic family being used. The SOP solution is usually better if using the TTL logic family, as NAND gates are the basic building block, which works well with SOP implementations. On the other hand, A POS solution would be acceptable when using the CMOS logic family since all sizes of NOR gates are available.

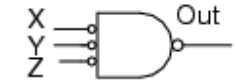


The gate diagrams for both cases are shown above, Product-Of-Sums left, and Sum-Of-Products right. Below, we take a closer look at the Sum-Of-Products version of our example logic, which is repeated at left.



Above all AND gates at left have been replaced by NAND gates at right.. The OR gate at the output is replaced by a NAND gate. To prove that AND-OR logic is equivalent to NAND-NAND logic, move the inverter invert bubbles at the output of the 3-NAND gates to the input of the final NAND as shown in going from above right to below left.

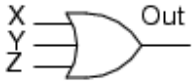




$Out = \overline{X\,Y\,Z}$ DeMorgans

$Out = \overline{X} + \overline{Y} + \overline{Z}$ Double negation

$Out = X+Y+Z$



$Out = X+Y+Z$

Above right we see that the output NAND gate with inverted inputs is logically equivalent to an OR gate by DeMorgan's theorem and double negation. This information is useful in building digital logic in a laboratory setting where TTL logic family NAND gates are more readily available in a wide variety of configurations than other types.

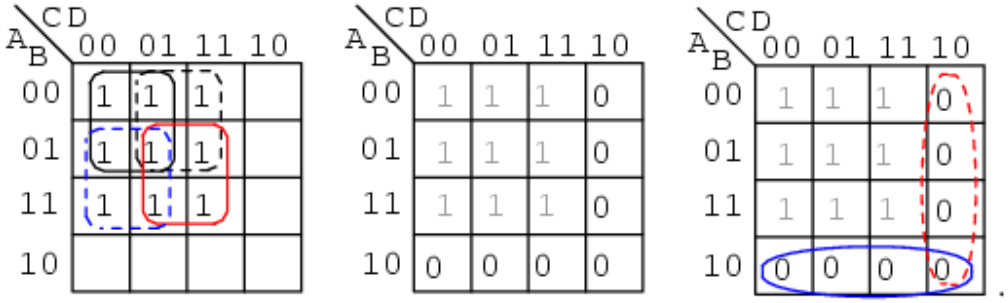
The Procedure for constructing NAND-NAND logic, in place of AND-OR logic is as follows:

- Produce a reduced Sum-Of-Products logic design.
- When drawing the wiring diagram of the SOP, replace all gates (both AND and OR) with NAND gates.
- Unused inputs should be tied to logic High.
- In case of troubleshooting, internal nodes at the first level of NAND gate outputs do NOT match AND-OR diagram logic levels, but are inverted. Use the NAND-NAND logic diagram. Inputs and final output are identical, though.
- Label any multiple packages U1, U2,.. etc.
- Use data sheet to assign pin numbers to inputs and outputs of all gates.

Example:

Let us revisit a previous problem involving an SOP minimization. Produce a Product-Of-Sums solution. Compare the POS solution to the previous SOP.

$$\text{Out} = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{B}\,C\,D + \overline{A}\,B\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + \overline{A}\,B\,C\,D + A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,B\,C\,D$$

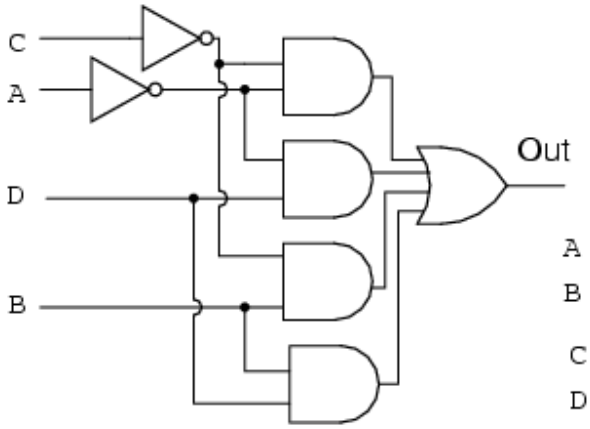


$$\text{Out} = \overline{A}\,\overline{C} + \overline{A}\,D + B\,\overline{C} + B\,D \qquad \text{Out} = (\overline{A}+B)(\overline{C}+D)$$

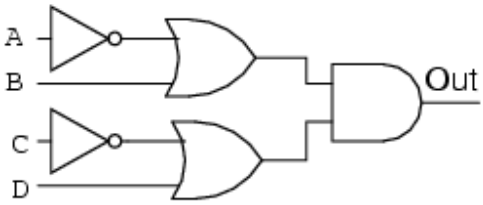
Solution:

Above left we have the original problem starting with a 9-minterm Boolean unsimplified expression. Reviewing, we formed four groups of 4-cells to yield a 4-product-term SOP result, lower left. In the middle figure, above, we fill in the empty spaces with the implied **0**s. The **0**s form two groups of 4-cells. The solid blue group is **(A'+B)**, the dashed red group is **(C'+D)**. This yields two sum-terms in the Product-Of-Sums result, above right **Out = (A'+B)(C'+D)**

Comparing the previous SOP simplification, left, to the POS simplification, right, shows that the POS is the least cost solution. The SOP uses 5-gates total, the POS uses only 3-gates. This POS solution even looks attractive when using TTL logic due to simplicity of the result. We can find AND gates and an OR gate with 2-inputs.



Out= $\overline{A}\,\overline{C} + \overline{A}D + B\overline{C} + BD$



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