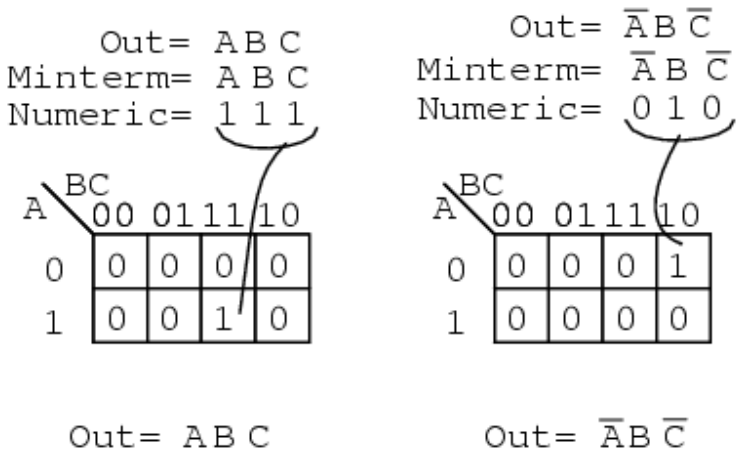
PRODUCT OF SOME FORMS(POS)

So far we have been finding Sum-Of-Product (SOP) solutions to logic reduction problems. For each of these SOP solutions, there is also a Product-Of-Sums solution (POS), which could be more useful, depending on the application. Before working a Product-Of-Sums solution, we need to introduce some new terminology. The procedure below for mapping product terms is not new to this chapter. We just want to establish a formal procedure for minterms for comparison to the new procedure for maxterms.

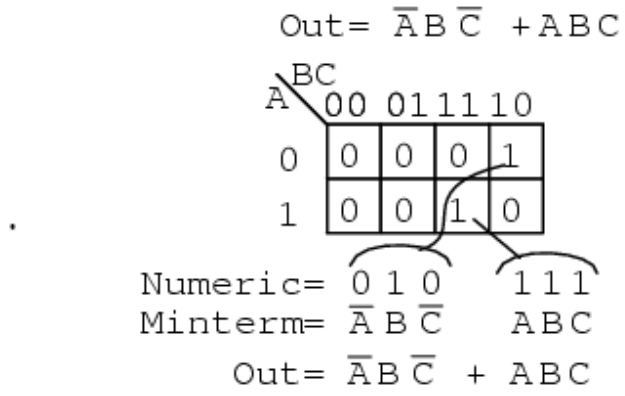


A *minterm* is a Boolean expression resulting in **1** for the output of a single cell, and **0**s for all other cells in a Karnaugh map, or truth table. If a minterm has a single **1** and the remaining cells as **0**s, it would appear to cover a minimum area of **1**s. The illustration above left shows the minterm **ABC**, a single product term, as a single **1** in a map that is otherwise **0**s. We have not shown the **0**s in our Karnaugh maps up to this point, as it is customary to omit them unless specifically needed. Another minterm **A'BC'** is shown above right. The point to review is that the address of the cell corresponds directly to the minterm being mapped. That is, the cell **111** corresponds to the minterm **ABC** above left. Above right we see that the minterm **A'BC'** corresponds directly to the cell **010**. A Boolean expression or map may have multiple minterms.

Referring to the above figure, Let's summarize the procedure for placing a minterm in a K-map:

- Identify the minterm (product term) term to be mapped.
- Write the corresponding binary numeric value.

- Use binary value as an address to place a **1** in the K-map
- Repeat steps for other minterms (P-terms within a Sum-Of-Products).



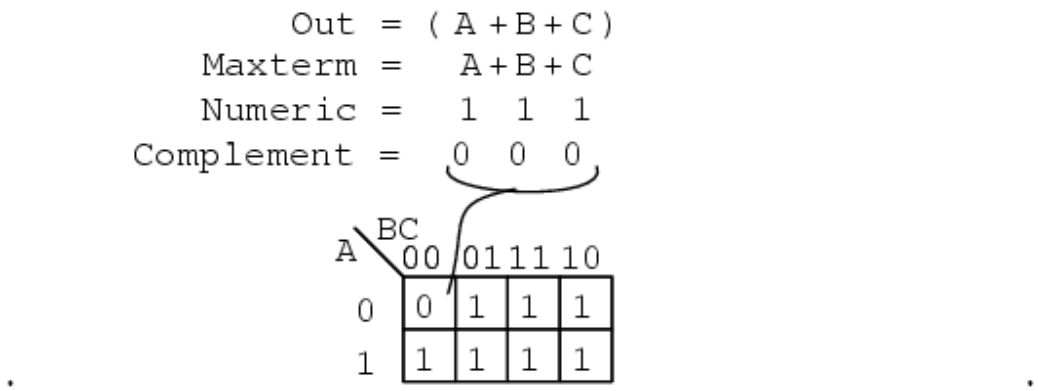
A Boolean expression will more often than not consist of multiple minterms corresponding to multiple cells in a Karnaugh map as shown above. The multiple minterms in this map are the individual minterms which we examined in the previous figure above. The point we review for reference is that the **1**s come out of the K-map as a binary cell address which converts directly to one or more product terms. By directly we mean that a **0** corresponds to a complemented variable, and a **1** corresponds to a true variable. Example: **010** converts directly to **A'BC'**. There was no reduction in this example. Though, we do have a Sum-Of-Products result from the minterms.

Referring to the above figure, Let's summarize the procedure for writing the Sum-Of-Products reduced Boolean equation from a K-map:

- Form largest groups of **1**s possible covering all minterms. Groups must be a power of 2.
- Write binary numeric value for groups.
- Convert binary value to a product term.
- Repeat steps for other groups. Each group yields a p-terms within a Sum-Of-Products.

Nothing new so far, a formal procedure has been written down for dealing with minterms. This serves as a pattern for dealing with maxterms.

Next we attack the Boolean function which is **0** for a single cell and **1**s for all others.



A *maxterm* is a Boolean expression resulting in a **0** for the output of a single cell expression, and **1**s for all other cells in the Karnaugh map, or truth table. The illustration above left shows the maxterm **(A+B+C)**, a single sum term, as a single **0** in a map that is otherwise **1**s. If a maxterm has a single **0** and the remaining cells as **1**s, it would appear to cover a maximum area of **1**s.

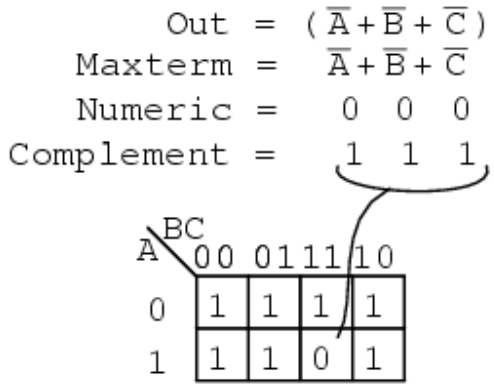
There are some differences now that we are dealing with something new, maxterms. The maxterm is a **0**, not a **1** in the Karnaugh map. A maxterm is a sum term, **(A+B+C)** in our example, not a product term.

It also looks strange that **(A+B+C)** is mapped into the cell **000**. For the equation **Out=(A+B+C)=0**, all three variables **(A, B, C)** must individually be equal to **0**. Only **(0+0+0)=0** will equal **0**. Thus we place our sole **0** for minterm **(A+B+C)** in cell **A,B,C=000** in the K-map, where the inputs are all**0** . This is the only case which will give us a **0** for our maxterm. All other cells contain **1**s because any input values other than (**(0,0,0)** for **(A+B+C)** yields **1**s upon evaluation.

Referring to the above figure, the procedure for placing a maxterm in the K-map is:

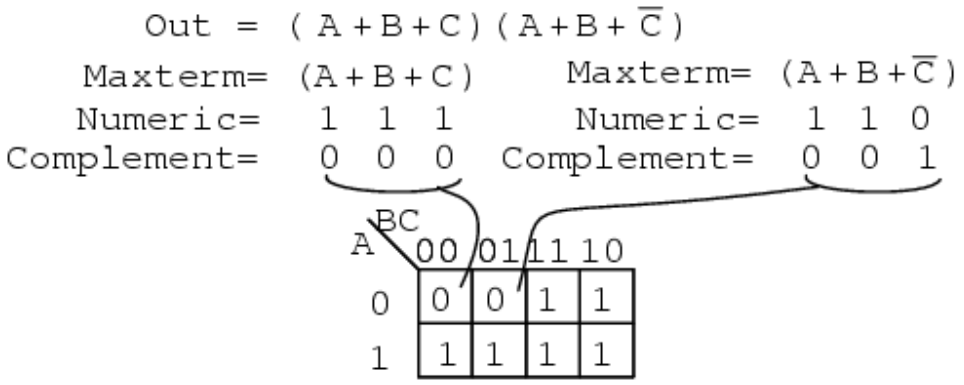
- Identify the Sum term to be mapped.

- Write corresponding binary numeric value.
- Form the complement
- Use the complement as an address to place a **0** in the K-map
- Repeat for other maxterms (Sum terms within Product-of-Sums expression).



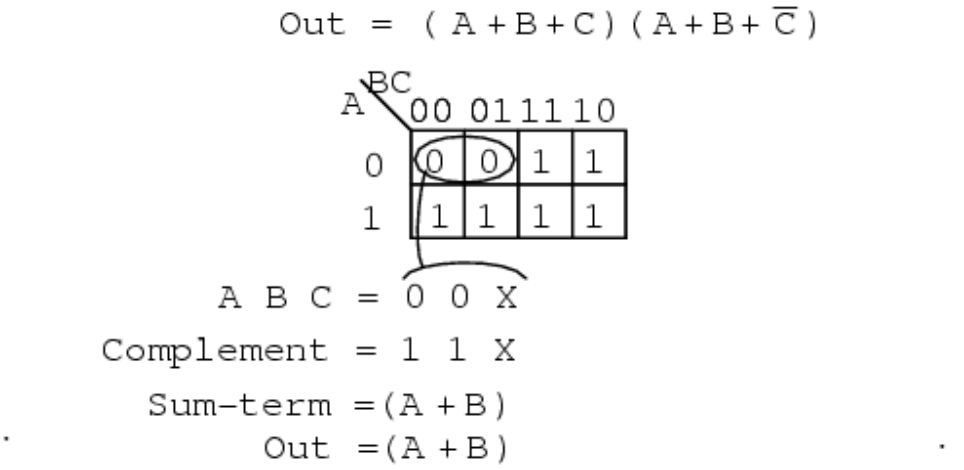
Another maxterm **A'+B'+C'** is shown above. Numeric **000** corresponds to **A'+B'+C'**. The complement is **111**. Place a **0** for maxterm **(A'+B'+C')** in this cell **(1,1,1)** of the K-map as shown above.

Why should **(A'+B'+C')** cause a **0** to be in cell **111**? When **A'+B'+C'** is **(1'+1'+1')**, all **1**s in, which is **(0+0+0)** after taking complements, we have the only condition that will give us a **0**. All the **1**s are complemented to all **0**s, which is **0** when **OR**ed.



A Boolean Product-Of-Sums expression or map may have multiple maxterms as shown above. Maxterm **(A+B+C)** yields numeric **111** which complements to **000**, placing a **0** in cell **(0,0,0)**. Maxterm **(A+B+C')** yields numeric **110** which complements to **001**, placing a **0** in cell **(0,0,1)**.

Now that we have the k-map setup, what we are really interested in is showing how to write a Product-Of-Sums reduction. Form the **0**s into groups. That would be a group of two below. Write the binary value corresponding to the sum-term which is **(0,0,X)**. Both A and B are **0** for the group. But, **C** is both **0** and **1** so we write an **X** as a place holder for **C**. Form the complement **(1,1,X)**. Write the Sum-term **(A+B)** discarding the **C** and the **X** which held its' place. In general, expect to have more sum-terms multiplied together in the Product-Of-Sums result. Though, we have a simple example here.



Let's summarize the procedure for writing the Product-Of-Sums Boolean reduction for a K-map:

- Form largest groups of **0**s possible, covering all maxterms. Groups must be a power of 2.
- Write binary numeric value for group.
- Complement binary numeric value for group.
- Convert complement value to a sum-term.
- Repeat steps for other groups. Each group yields a sum-term within a Product-Of-Sums result.