**Chapter 3**

Table of Contents

[K’ Map (Karnaugh Map) 2](#_Toc66754765)

[K’ Maps with 5/6 Variables 8](#_Toc66754766)

[Don’t Cares 9](#_Toc66754767)

[NAND and NOR Implementation 11](#_Toc66754768)

[NAND 11](#_Toc66754769)

[NOR 12](#_Toc66754770)

[Tabular Method or Quine McClusky (Q.M.) Method 13](#_Toc66754771)

[Tabular Method with Decimals 18](#_Toc66754772)

[Tabular Method with Don’t Cares 21](#_Toc66754773)

[Tabulation Method in POS Form 23](#_Toc66754774)

## K’ Map (Karnaugh Map)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Minterm Expression |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 000 m0 | 001 m1 | 011 m3 | 010 m2 |
| 1 | 100 m4 | 101 m5 | 111 m7 | 110 m6 |

The values in the map are arranged in an odd manner. This is due to the fact that adjacent cells can only differ by one variable (hamming distance = 1).

In the map, and as well as and are considered adjacent. This is called map rolling. Map rolling also applies to the top and bottom edges of the map.

Cells that are diagonally adjacent are ignored since their hamming distance is not 1.

If a function gives a result of for the values , , , and , its K’ Map will look like this.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

The standard SOP form of is thus

This can be expressed in minimal SOP form as

In K’ Maps, cells can be paired in groups of . So, a group may contain 1, 2, 4, 8, 16 or 32 cells. A group that contains only 1s is called an implicant.

For the function , we can form 2 groups in order to include all the minterms. Group 1 contains and , and group 2 contains , , and .

Group 2 contains the highest number of minterms and is called the prime implicant.

If a group exists that has at least one member that cannot be grouped in any other way, that group is called an essential prime implicant.

In group 1, the values of and remain constant, so we will consider . is used since the value of is 0.

In group 2, only the value of remains constant so we will only consider .

Thus .

K’ Maps can thus give us the minimal SOP form of a function very quickly.

Problem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

Group 1 = and so

Group 2 = and so

Group 3 = and so

Group 4 = so

Problem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Group 1 = and so

Group 2 = and so

Group 3 = and so

However, group 3 is redundant as both its members have already been included in other groups, so it must be ignored. The basic rule is to maximize implicant size but avoid redundancy.

Problem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |

Group 1 = and so

Group 2 = and so

is left and can be grouped in two ways.

Group 3 = and so

Group 4 = and so

This gives us two possible results.

or

Both of these results are correct.

Problem

A 4 variable K’ Map looks like this (Note: 4 corners can form 1 group):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABCD | 00 | 01 | 11 | 10 |
| 00 | m0 | m1 | m3 | m2 |
| 01 | m4 | m5 | m7 | m6 |
| 11 | m12 | m13 | m15 | m14 |
| 10 | m8 | m9 | m11 | m10 |

So for ,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABCD | 00 | 01 | 11 | 10 |
| 00 | 1 | 0 | 1 | 1 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 0 |

Group 1 = and so

Group 2 = , , and so

Group 3 = and so

Group 4 = and so

K’ Maps with maxterms are handled in exactly the same way, except groups of 0s are considered instead of groups of 1s.

### K’ Maps with 5/6 Variables

With 5 or 6 variable K’ Maps, the actual system becomes evident.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ABCDE | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
| 00 | 0 | 1 | 3 | 2 | 6 | 7 | 5 | 4 |
| 01 | 8 | 9 | 11 | 10 | 14 | 15 | 13 | 12 |
| 11 | 24 | 25 | 27 | 26 | 30 | 31 | 29 | 28 |
| 10 | 16 | 17 | 19 | 18 | 22 | 23 | 21 | 20 |

To find adjacent groups, the halves of the map above are folded down their centers and the overlapping columns are checked.

This means columns 1 and 8 are adjacent, 2 and 7 are adjacent, 3 and 6 are adjacent and 4 and 5 are adjacent.

Folding each half down the center also shows that columns 1 and 4 are adjacent, and columns 5 and 8 are adjacent.

The 6 variable K’ Map works in exactly the same manner.

### Don’t Cares

Don’t care are valid minterms that’s are being ignored. They are defined alongside the main function.

For example, a 4-bit function defining all possible real numbers will have the numbers 10 – 15 as Don’t Cares, since they can be written with the numbers 0 – 9.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABCD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 1 |
| 01 | 1 | 1 | 1 | 1 |
| 11 | X | X | X | X |
| 10 | 1 | 1 | X | X |

In the diagram, the X’s mark the Don’t Cares.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABCD | 00 | 01 | 11 | 10 |
| 00 | X | 1 | 1 | X |
| 01 | 0 | X | 1 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 0 | 0 | 1 | 0 |

X can be considered to be 1 and included to increase implicant size.

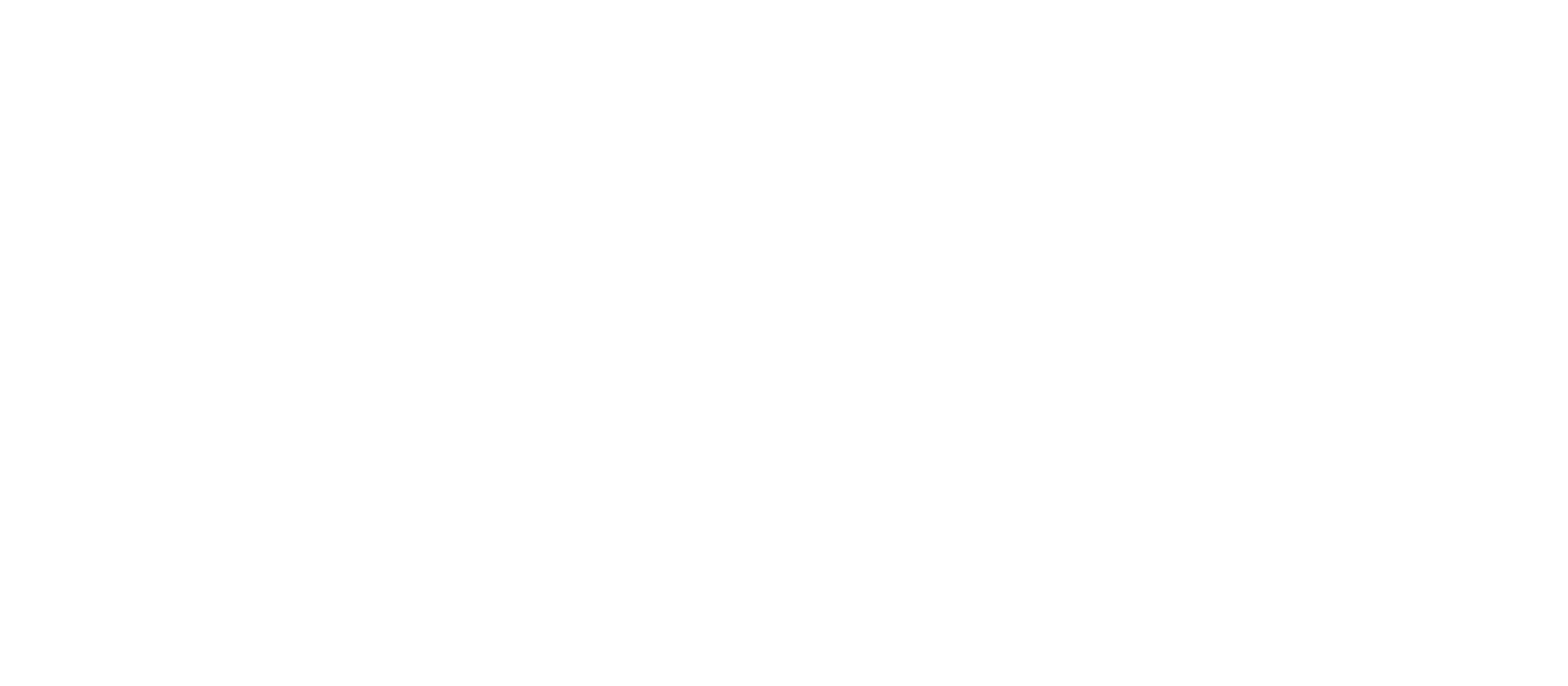
One of the X’s is being ignored since it does not contribute to the size of any implicants.

Maxterms are handled similarly, with X’s being considered as 0’s.

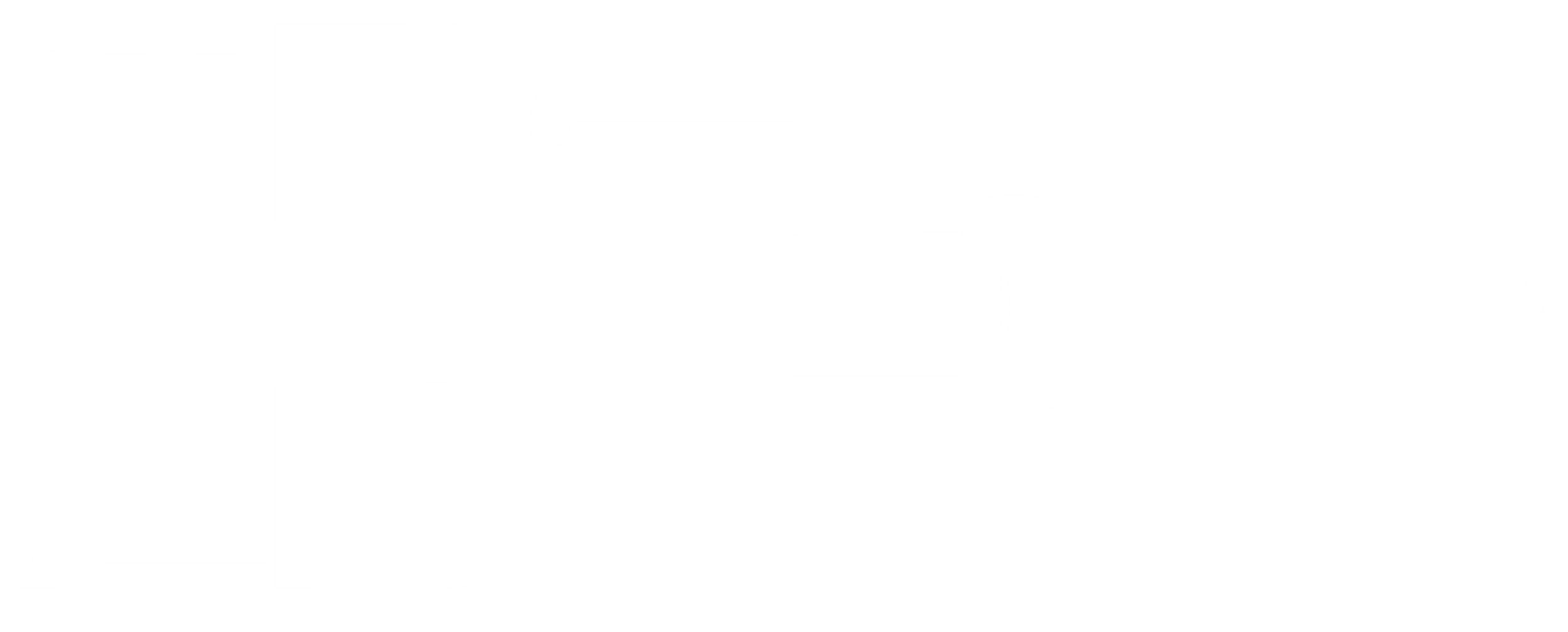
## NAND and NOR Implementation

### NAND

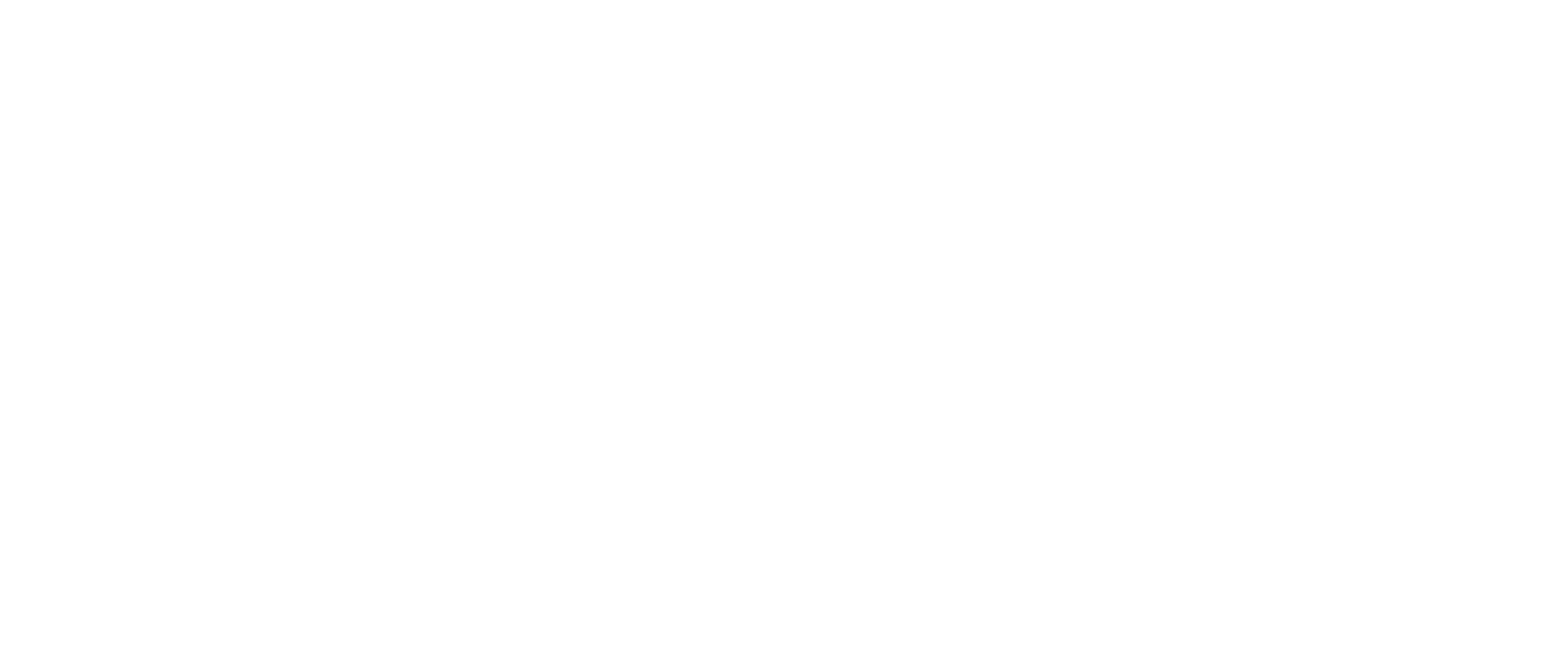
* Convert expression to standard SOP form
* Draw circuit with AND and OR gates



* Change AND gates to AND INVERT gates
* Change OR gates to OR INVERT gates



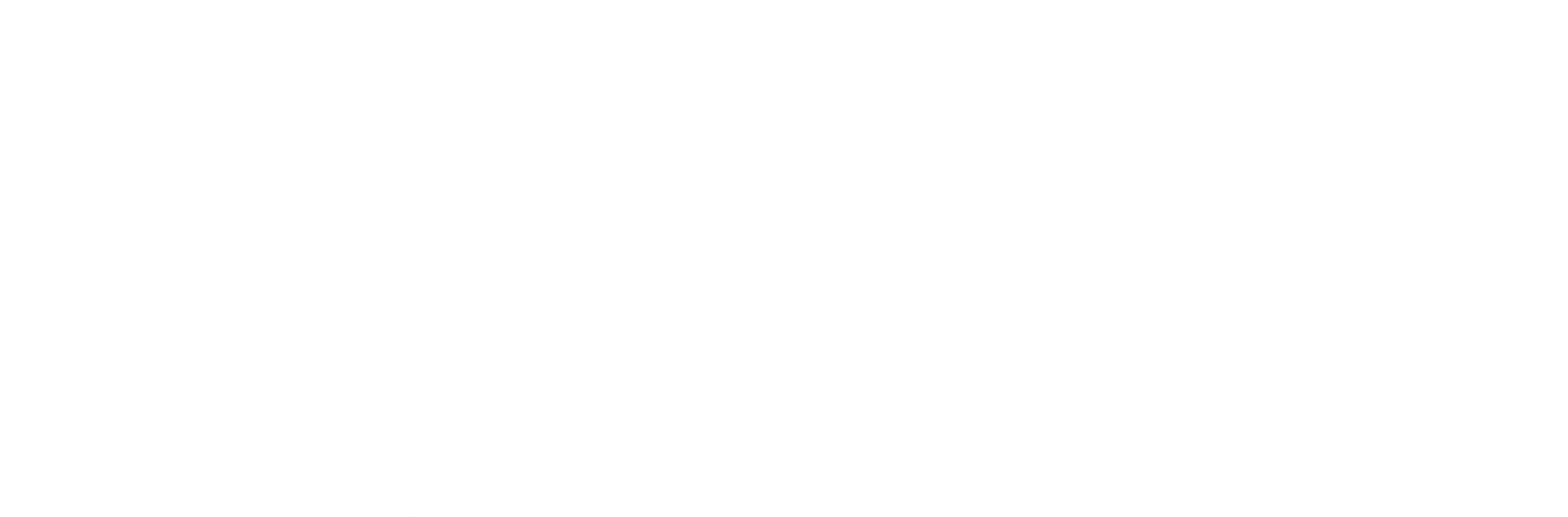
* INVERT OR = AND INVERT
* Change INVERT OR gates to AND INVERT gates



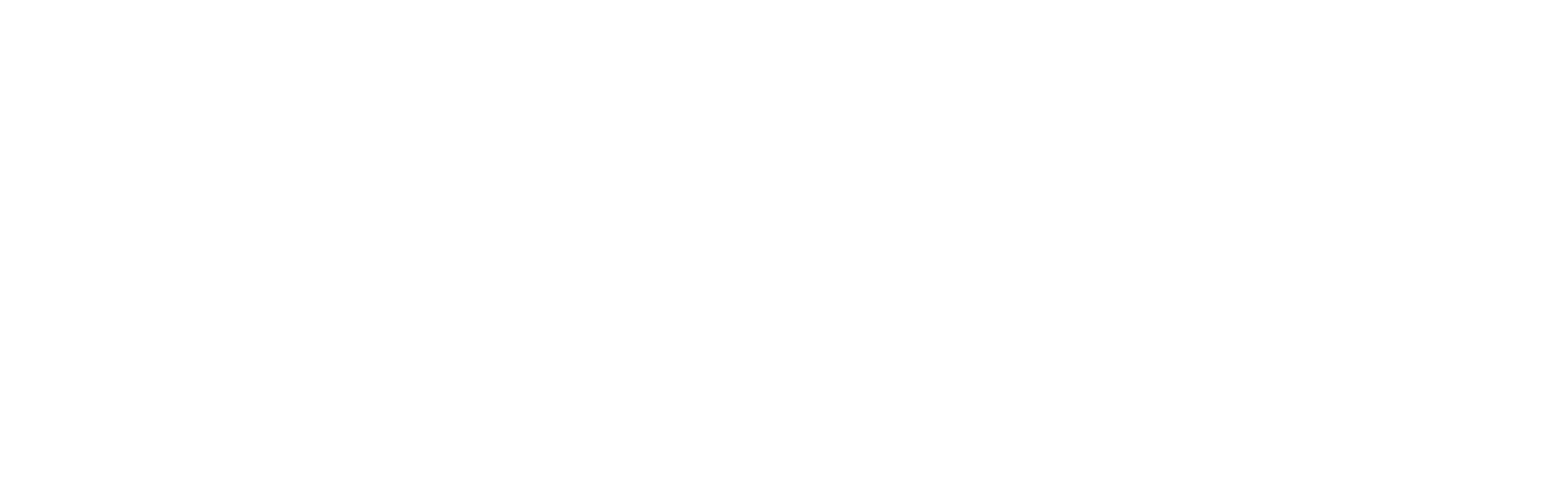
F = ab+ cd

### NOR

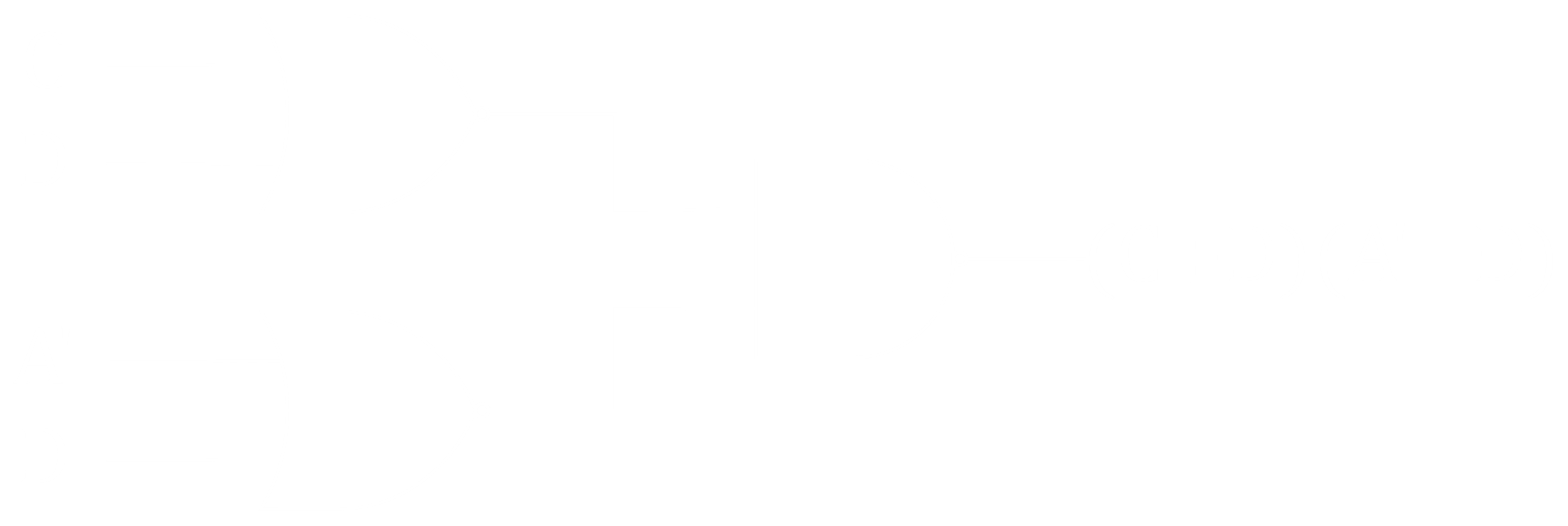
* Convert expression to standard POS Form
* Draw circuit with AND and OR gates



* Change AND gates to INVERT AND gates
* Change OR gates to OR INVERT gates



* INVERT AND = OR INVERT
* Change INVERT AND gates to OR INVERT gates



F = bd’ + ac’ = (b’+d)(a’+c)

Notes:

* For both NAND and NOR implementation, input and output may need to be inverted as required.
* The process for multilevel NAND and NOR implementation is exactly the same.
* The given method is for 2 Level NAND and NOR implementation. 3 Level NAND and NOR implementation starts with inverted values, thus requiring the final output to be inverted an extra time (the 3rd level).

## Tabular Method or Quine McClusky (Q.M.) Method

For more than 6 variables, the K’ Map method is difficult to implement. So instead, the tabular method is used.

For the given function, the binary representation of the minterms is found.

Next, a table is made where the minterms that have the save number of 1’s are grouped together.

First List:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group No. | Minterm | Binary Representation | | | |  |
| A | B | C | D |
| 0 |  | 0 | 0 | 0 | 0 | ✓ |
| 1 |  | 0 | 0 | 0 | 1 | ✓ |
|  | 1 | 0 | 0 | 0 | ✓ |
| 2 |  | 0 | 0 | 1 | 1 | ✓ |
|  | 1 | 0 | 0 | 1 | ✓ |
| 3 |  | 0 | 1 | 1 | 1 | ✓ |
|  | 1 | 0 | 1 | 1 | ✓ |
| 4 |  | 1 | 1 | 1 | 1 | ✓ |

Note that all the rows have a checkmark at the end. This will be explained in a moment.

Now, another table is made, grouping the minterms from the previous table that only differ by one variable, i.e. comparing the and groups and grouping the minterms that differ by one variable. In the binary representation column, an underscore is used in place of the variable that does differ, meaning that variable is discounted. In the previous table, a checkmark is placed at the end of the row for any group member that is being used in the next group. If any group member remains without a check, that member is a prime implicant. For the first list, no such member exists.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group No. | Matched Pairs | Binary Representation | | | |  |
| A | B | C | D |
| 0 |  | 0 | 0 | 0 | \_ | ✓ |
|  | \_ | 0 | 0 | 0 | ✓ |
| 1 |  | 0 | 0 | \_ | 1 | ✓ |
|  | \_ | 0 | 0 | 1 | ✓ |
|  | 1 | 0 | 0 | \_ | ✓ |
| 2 |  | 0 | \_ | 1 | 1 | ✓ |
|  | \_ | 0 | 1 | 1 | ✓ |
|  | 1 | 0 | \_ | 1 | ✓ |
| 3 |  | \_ | 1 | 1 | 1 | ✓ |
|  | 1 | \_ | 1 | 1 | ✓ |

This process is repeated until a list is found which does not have a single checked group member, i.e. all group members are prime implicants. There is no formal method for this so the lists do not have to be separate or organized as shown. They just need to contain the information.

Third List:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Group No. | Matched Pairs | Binary Representation | | | |  |
| A | B | C | D |
| 0 |  | \_ | 0 | 0 | \_ |  |
|  | \_ | 0 | 0 | \_ |  |
| 1 |  | \_ | 0 | \_ | 1 |  |
|  | \_ | 0 | \_ | 1 |  |
| 2 |  | \_ | \_ | 1 | 1 |  |
|  | \_ | \_ | 1 | 1 |  |

A fourth list cannot be made. The first and second lists have no unchecked members, so they contain no prime implicants. The third list does.

From the group members that are prime implicants, the non-discounted variables are considered. Thus, the prime implicants from the third list are , and .

Once the prime implicants are found, a final table is made to find the essential prime implicants. A cross is placed if a specific given minterm is present in the minterms present in the prime implicant.

Prime Implicant Table:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prime Implicant | Minterms Present | Minterms Given | | | | | | | |
|  |  |  |  |  |  |  |  |
|  | , , , | X | X |  |  | X | X |  |  |
|  | , , , |  | X | X |  |  | X | X |  |
|  | , , , |  |  | X | X |  |  | X | X |
|  |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Each of the columns is checked to see if any exist that have only one cross in them, i.e. a minterm exists in only one prime implicant. If they are found, then the prime implicant corresponding to that cross is an essential prime implicant. Every column which is affected by the prime implicant is given a checkmark at the end of the column. If any column exists which remains unchecked, another implicant must be added to the function, such that the implicant covers all the unchecked columns. In the given table, this case does not exist.

From the given table, it can be seen that , and are present only once, meaning the essential prime implicants are and .

Thus, .

This can be confirmed using the K’ Map method.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABCD | 00 | 01 | 11 | 10 |
| 00 | 1 | 1 | 1 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 1 | 1 | 1 | 0 |

Another example is given.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First List | | | Second List | | | Third List | | |
|  | 0001 | ✓ |  | \_001 |  |  | 10\_ \_ |  |
|  | 0100 | ✓ |  | 01\_0 |  |  | 10\_ \_ |  |
|  | 1000 | ✓ |  | 100\_ | ✓ |  |  |  |
|  | 0110 | ✓ |  | 10\_0 | ✓ |  |  |  |
|  | 1001 | ✓ |  | 011\_ |  |  |  |  |
|  | 1010 | ✓ |  | 10\_1 | ✓ |  |  |  |
|  | 0111 | ✓ |  | 101\_ | ✓ |  |  |  |
|  | 1011 | ✓ |  | \_111 |  |  |  |  |
|  | 1111 | ✓ |  | 1\_11 |  |  |  |  |

From the second list, , , , and remain unchecked. For the third list, and remain unchecked. So, the prime implicants are , , , , and .

Prime Implicant Table:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prime Implicant | Minterms Present |  | Minterms Given | | | | | | | |
|  |  |  |  |  |  |  |  |  |
|  | , | X |  |  |  |  | X |  |  |  |
|  | , |  | X | X |  |  |  |  |  |  |
|  | , |  |  | X | X |  |  |  |  |  |
|  | , |  |  |  | X |  |  |  |  | X |
|  | , |  |  |  |  |  |  |  | X | X |
|  | , , , |  |  |  |  | X | X | X | X |  |
|  |  | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ |  |

, and are obvious essential prime implicants. However, the columns for and remain unchecked. These columns can be covered by and or by alone, so is picked. Thus .

### Tabular Method with Decimals

The tabular method can be done without converting the decimal numbers to binary numbers as well. This makes the process slightly faster.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First List | | | Second List | | Third List | |
| 0001 | 1 | ✓ | 1, 9 (8) |  | 8, 9, 10, 11 (1, 2) |  |
| 0100 | 4 | ✓ | 4, 6 (2) |  | 8, 9, 10, 11 (1, 2) |  |
| 1000 | 8 | ✓ | 8, 9 (1) | ✓ |  |  |
| 0110 | 6 | ✓ | 8, 10 (2) | ✓ |  |  |
| 1001 | 9 | ✓ | 6, 7 (1) |  |  |  |
| 1010 | 10 | ✓ | 9, 11 (2) | ✓ |  |  |
| 0111 | 7 | ✓ | 10, 11 (1) | ✓ |  |  |
| 1011 | 11 | ✓ | 7, 15 (8) |  |  |  |
| 1111 | 15 | ✓ | 11, 15 (4) |  |  |  |

The first list is done normally, dividing the numbers into different groups based on the number of 1’s in its binary representation.

For the second list, each number of one group from the first list is compared with each number of the group immediately after it. If the difference between two numbers is a power of 2 (i.e. 1, 2, 4, 8 etc.), then they are grouped together and written as a member of the second list. Their difference is written in the bracket. For example, 1 and 6 cannot be grouped, since their difference is 5, but 1 and 9 can be grouped since their difference is 8 (23). The second list is divided into groups based on which consecutive groups were compared in the first list, as was done in the binary form of the tabular method.

For the third list, every two consecutive groups of the second list are compared as before. However, for two members to be compared, they must have the same number written in the bracket (i.e. the difference between the numbers in the members must be the same), and the numbers in the second group’s member must be larger than that of the first. For example, 8, 9 and 6, 7 cannot be grouped even though they both have 1 written in the bracket, because 6, 7 is smaller than 8, 9. However, 8, 9 and 10, 11 can be grouped. The number in the previous bracket and the difference between the two newly groups members is now written in the bracket in ascending order. For 8, 9 and 10, 11, this is (1, 2).

As before, every time a member is used in the next list, a checkmark is placed beside it. All unchecked members are prime implicants.

To obtain the prime implicants, the binary representation of the members must be found. For members with more than 1 number in them, the binary representation of any one of the numbers can be used, placing an underscore in the place value given in the bracket. For example, 1, 9 (8) is a prime implicant. It will be represented in binary form as \_001. Thus, the prime implicant is .

|  |  |  |
| --- | --- | --- |
| 1, 9 (8) | \_001 |  |
| 4, 6 (2) | 01\_0 |  |
| 6, 7 (1) | 011\_ |  |
| 7, 15 (8) | \_111 |  |
| 11, 15 (4) | 1\_11 |  |
| 8, 9, 10, 11 (1, 2) | 10\_ \_ |  |

Now the prime implicant table is made as before.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prime Implicant | Minterms Present |  | Minterms Given | | | | | | | |
| 1 | 4 | 6 | 7 | 8 | 9 | 10 | 11 | 15 |
|  | 1, 9 | X |  |  |  |  | X |  |  |  |
|  | 4, 6 |  | X | X |  |  |  |  |  |  |
|  | 6, 7 |  |  | X | X |  |  |  |  |  |
|  | 7, 15 |  |  |  | X |  |  |  |  | X |
|  | 11, 15 |  |  |  |  |  |  |  | X | X |
|  | 8, 9, 10, 11 |  |  |  |  | X | X | X | X |  |
|  |  | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ |  |

### Tabular Method with Don’t Cares

If don’t cares become involved, they are included in the initial table from which the prime implicants are found as though they are normal minterms. However, they are not included under the ‘Minterms Given’ column of the final prime implicants table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First List | | | Second List | | Third List | |
| 0000 | 0 | ✓ | 0, 1 (1) |  | 0, 2, 8, 10 (2, 8) |  |
| 0001 | 1 | ✓ | 0, 2 (2) | ✓ | 0, 4, 8, 12 (4, 8) |  |
| 0010 | 2 | ✓ | 0, 4 (4) | ✓ |  |  |
| 0100 | 4 | ✓ | 0, 8 (8) |  |  |  |
| 1000 | 8 | ✓ | 2, 10 (8) |  |  |  |
| 1010 | 10 | ✓ | 4, 12 (8) |  |  |  |
| 1100 | 12 | ✓ | 8, 10 (2) | ✓ |  |  |
| 1101 | 13 | ✓ | 8, 12 (4) | ✓ |  |  |
| 1111 | 15 | ✓ | 12, 13 (1) |  |  |  |
|  |  |  | 13, 15 (2) |  |  |  |

Prime Implicants

|  |  |  |
| --- | --- | --- |
| 0, 1 (1) | 000\_ |  |
| 0, 2 (2) | 00\_0 |  |
| 0, 8 (8) | \_000 |  |
| 2, 10 (8) | \_010 |  |
| 4, 12 (8) | \_100 |  |
| 12, 13 (1) | 110\_ |  |
| 13, 15 (2) | 11\_1 |  |
| 0, 2, 8, 10 (2, 8) | \_0\_0 |  |
| 0, 4, 8, 12 (4, 8) | \_ \_00 |  |

Prime Implicant Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prime Implicant | Minterms Present | Minterms Given | | | | | | |
| 0 | 4 | 8 | 10 | 12 | 13 | 15 |
|  | 0, 1 | X |  |  |  |  |  |  |
|  | 0, 2 | X |  |  |  |  |  |  |
|  | 0, 8 | X |  | X |  |  |  |  |
|  | 2, 10 |  |  |  | X |  |  |  |
|  | 4, 12 |  | X |  |  | X |  |  |
|  | 12, 13 |  |  |  |  | X | X |  |
|  | 13, 15 |  |  |  |  |  | X | X |
|  | 0, 2, 8, 10 | X |  | X | X |  |  |  |
|  | 0, 4, 8, 12 | X | X | X |  | X |  |  |
|  |  |  |  |  |  |  | ✓ | ✓ |

Here, only is an essential prime implicant. The prime implicants and were chosen since just these two can cover the remaining unchecked minterms, thus giving the minimum number of terms. Another possible solution could be (this is obtained from the K’ Map method). These solutions are equivalent, meaning one problem can have multiple solutions.

### Tabulation Method in POS Form

The tabulation method for the POS form is the exact same, except the prime variables are written in the POS form instead of the SOP form.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| First List | | | Second List | | Third List | |
| 0001 | 1 | ✓ | 1, 9 (8) |  | 8, 9, 10, 11 (1, 2) |  |
| 0100 | 4 | ✓ | 4, 6 (2) |  | 8, 9, 10, 11 (1, 2) |  |
| 1000 | 8 | ✓ | 8, 9 (1) | ✓ |  |  |
| 0110 | 6 | ✓ | 8, 10 (2) | ✓ |  |  |
| 1001 | 9 | ✓ | 6, 7 (1) |  |  |  |
| 1010 | 10 | ✓ | 9, 11 (2) | ✓ |  |  |
| 0111 | 7 | ✓ | 10, 11 (1) | ✓ |  |  |
| 1011 | 11 | ✓ | 7, 15 (8) |  |  |  |
| 1111 | 15 | ✓ | 11, 15 (4) |  |  |  |

Prime Implicants:

|  |  |  |
| --- | --- | --- |
| 1, 9 (8) | \_001 |  |
| 4, 6 (2) | 01\_0 |  |
| 6, 7 (1) | 011\_ |  |
| 7, 15 (8) | \_111 |  |
| 11, 15 (4) | 1\_11 |  |
| 8, 9, 10, 11 (1, 2) | 10\_ \_ |  |

Prime Implicant Table:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prime Implicant | Minterms Present |  | Minterms Given | | | | | | | |
| 1 | 4 | 6 | 7 | 8 | 9 | 10 | 11 | 15 |
|  | 1, 9 | X |  |  |  |  | X |  |  |  |
|  | 4, 6 |  | X | X |  |  |  |  |  |  |
|  | 6, 7 |  |  | X | X |  |  |  |  |  |
|  | 7, 15 |  |  |  | X |  |  |  |  | X |
|  | 11, 15 |  |  |  |  |  |  |  | X | X |
|  | 8, 9, 10, 11 |  |  |  |  | X | X | X | X |  |
|  |  | ✓ | ✓ | ✓ |  | ✓ | ✓ | ✓ | ✓ |  |