Quine-McCluskey Method

Andrew Kennings

April 3, 2017

Rather than using Karnaugh maps which is a graphical method to optimize logic functions with only a few inputs, we can chose to use a tabular method instead. This tabular method is commonly referred to as the *Quine-McCluskey method*. This method will work for logic functions with any number of inputs.

The method is based on using a table to generate all of the prime implicants for a logic function and, subsequently, using a matrix to decide on which prime implicants to select to cover the function. The Quine-McCluskey method will be illustrated through an example. Consider the 4-input logic function given by $f = f(a,b,c,d) = \sum m(0,2,3,5,10,11,12,13,15)$. These minterms are listed as shown in Figure 1. There are a couple of important points to note about

0	0000
2	0010
3	0011
5	0101
12	1100
10	1010
11	1011
13	1101
15	1111

Figure 1: The first step of the Quine-McCluskey method which involves listing all of the minterms and rows of the truth table grouped according to the number of variables which are equal to 1. At this point all terms involved all input variables.

Figure 1. Minterms are *grouped* according to the number of positive literals (or, equivalently, the number of 1s in that row of the truth table). This is because the Quine-McCluskey method relies on the Boolean operation $x_i x_j + x_i x_j^{'} = x_i$. This Boolean operation means that we are looking for situations in which rows (minterms) are identical *except for a single variable* which is 0 in one row and 1 in another row. By arranging the minterms as shown in Figure 1, we only need to compare a minterm with minterms if the immediately preceding group.

We now start to perform pairwise comparisons of minterms looking for minterms which differ by only the value of a single variable. Performing this task leads to the table shown in Figure 2. In Figure 2, the removed variables which result from the pairwise comparison of minterms are marked with an X. We now proceed to perform the same pairwise comparisons on the product terms in Figure 2. Specifically, we look compare all product terms in a group with all product terms in the immediately preceeding group. Further, the

$0, 2$	$00X0$
$2, 3$	$001X$
$2, 10$	$X010$
$3, 11$	$X011$
$10, 11$	$101X$
$5, 13$	$X101$
$12, 13$	$110X$
$11, 15$	$1X11$
$13, 15$	$11X1$

Figure 2: Next step of the Quine-McCluskey method which involves comparing minterms to identify variables which can be removed. The result is a set of product terms in which one variable has been removed.

variables marked with an X must match — when comparing product terms we must make sure to only compare product terms involving the same set of input variables. Performing these comparisons leads to the table in Figure 3. In this step, we find only one new prod-

$2, 3, 10, 11$	$X01X$

Figure 3: Next step of the Quine-McCluskey method which involves comparing product terms to identify more variables which can be removed. The result is a set of product terms in which two variable has been removed.

uct term can be generated which combines minterms 2, 3, 10 and 11.

The next step of the Quine-McClusky algorthm is to look at all the product terms generated and to mark which product terms were successfully paired when generating the next table — since these product terms have been combine into other product terms, they cannot be prime implicants and are therefore not required to implement f (since f can be generated using only prime implicants). The tables of product terms are repeated in Figure 4 and those product terms (minterms) which have been combine are checked. As marked

0	0000	✓	$0, 2$	$00X0$				
2	0010	✓	$2, 3$	$001X$	✓			
3	0011	✓	$2, 10$	$X010$	✓			
5	0101	✓	$3, 11$	$X011$	✓			
12	1100	✓	$10, 11$	$101X$	✓	$2, 3, 10, 11$	$X01X$	
10	1010	✓	$5, 13$	$X101$				
11	1011	✓	$12, 13$	$110X$				
13	1101	✓	$11, 15$	$1X11$				
15	1111	✓	$13, 15$	$11X1$				

Figure 4: All product terms generated by applying the Quine-McCluskey method. Those product terms which are **not** required are checked.

in Figure 4, none of the minterms will be required to implement f as they have all been included in larger product terms. Similarly, several of the product terms themselves have been included in a larger product term. In summary, the Quine-McCluskey method has indicated that the prime implicants for f are the product terms $a'b'c'$, $bc'd$, abc',

acd, abd, and $b'c$.

The final step of the Quine-McCluskey method is to compose a matrix in which the rows of the matrix show the prime implicants and the columns show the minterms. A matrix entry has a check mark if the corresponding minterm is included by the prime implicant. This is shown in Figure 5. We now consider the matrix and

Prime Implicant	**Minterm** 0	2	3	5	10	11	12	13	15
$0, 2 = 00X0$	✓	✓							
$5, 13 = X101$				✓				✓	
$12, 13 = 110X$							✓	✓	
$11, 15 = 1X11$						✓			✓
$13, 15 = 11X1$								✓	✓
$2, 3, 10, 11 = X01X$		✓	✓		✓	✓			

Figure 5: Matrix showing the minterms and prime implicants.

look at the columns of the matrix. If we see a column with only a single checkmark, then we **must** include the appropriate product term to ensure that minterm is included in the final expression for f. This is equivalent to determining which of the prime implicants are essential prime implicants. In our example, it is clear that we must include product term $(0, 2)$ to cover minterm 0, product term $(2, 3, 10, 11)$ to cover minterms 3 and 10, product term $(5, 13)$ to cover minterm 5, and and product term $(12, 13)$ to cover minterm 12.

By including these minterms, we can generate a smaller matrix showing the remaining prime implicants and any remaining minterms which have not already been covered. This is shown in Figure 6. From Figure 6 we can see that we must still select a

Prime Implicant	**Minterm** 15
$11, 15 = 1X11$	✓
$13, 15 = 11X1$	✓

Figure 6: Matrix showing remaining minterms to cover after removing the essential prime implicants and all minterms included in the essential prime implicants.

prime implicant to cover minterm 15. It turns out that we have two choices and we can select either option. The final implementation of f is given by

$$f = f(a, b, c, d) = a'b'd' + bc'd + abc' + b'c + \begin{cases} acd \\ abd \end{cases}$$

We will not do an example, but don't cares are also easily handled. For those rows of the truth table which are don't cares, we include them in the initial table along with the minterms and use the don't cares to aid in the generation of prime implicants. However, when we

form the matrix we do **not** include the don't cares as columns in the matrix. This is because we are not required to cover the don't cares in the final implementation of f.