

**CSCE2301:**

**Digital Design I**

**Project1 #3-variable K-Map**

Name: Ahmed Osama Hassan

Name: Mohamed Ashraf Hemdan

ID: 900171850

ID: 900171923

Section: 02

Dr. Mohamed Taher

Saturday, 21st of March

**Program Design:**

**Part (1):**

To validate the value of the input, the program has to go through two phases. Since the minterms are entered in one go, as specified by the sample input given in the project description, the number of minterms and their values need to be checked.

1. **Number of minterms:**

This program is designed for a 3-variable K-map; hence, the maximum number of minterms is only or 8 different minterms. The comma-separated minterms from the input is entered into a vector. The Number of minterms is then validated through a while loop that activates if the size of the vector is more than 8. The while-loop asks the user to remove one minterm at a time, of their choice, until the size is equal to 8.

1. **The value of minterms:**

Since the minterms of a 3-variable k-map range from 0 to -1 then the value of each individual minterm can’t be more than 7 or less than 0 or else there will be no place for it in the map. A for-loop sweeps the whole vector and check for any value that doesn’t meet the requirements then ask the user to change it.

**Part (2):**

A 2-D array is going to be used to generate the k-map by storing the minterms in a 2x4 array and then printing it. To do so part (2) goes through a for-loop that sweeps all the minterms to apply on them two phases, conversion and placement.

1. **Conversion**:

The value of each minterm is nothing but the decimal representation for a 3-digit binary number (hence 3-variable k-map). Thus to proceed in creating the k-map, the values of these three digit numbers are important since they will build for the next stage, placement. In the beginning of the for-loop a set of arithmetic operations is carried out, dividing by two three times and each time the remainder represents a value of one of the three digits from LSD to MSD.

1. **Placement:**

To place the minterms a few conditions had to be set. It is observed that the MSD, the last remainder obtained when dividing by 2, determines which row the minterm is going to be stored in hence an if-else statement with both values possible for MSD, 1 or 0, is placed in the for loop just after the arithmetic conversions. In each branch, the column that the minterm will be placed into is determined based on the values of the two other digits hence other nested if-else statements are employed. When the rows and columns are determined, the value one is stored in that element of the 2-D array.

Finally after conversion and placement the K-map is generated and nested for-loops are used to print the value of each element in the 2-D array in a matrix format, each row is separated by a newline.

**Note:**

The 2-D array is created in the main program then passed by reference to optimize memory usage. Also, although in part(3) generating the function depends on the minterms, part(3) function takes the K-map first then from it generates minterms to be used later in creating the function. The reason behind that is that repetition of minterms was not checked in part (1) and repeated minterms will affect the function but won’t affect the k-map; thus the k-map can be thought of as a third layer of validation that removes repetition and the minterms generated from it in part (3) are of the right value, with no repetition and they don’t exceed 8 minterms.

**Part (3):**

1. **Generating minterms array**

A function has been made to calculate the number of minterms and to generate their respective minterms in an array (8\*4). This array is made specifically for simplifying the minterms. You could think of this array as a table as shown below although the displayed array should be extended to 8 rows. It’s made with 2 rows for simplicity. The first number indicates A whether it’s A or A’. 1 means A and 0 means A’. The same rule applies to other letters.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | minterm | Taken |
| 0 | 0 | 1 | A’B’C | 0 |
| 1 | 0 | 1 | AB’C | 1 |
| 2 | 0 | 1 | B’C | 1 |

1. **Simplifying minterms array**

The minterms array, represented above, is passed to another function called update\_mTerms(), to scan over the terms for any similarities and simplification possibilities. For example, the first two terms shown above can be simplified to B’C (-01). As A is the only mismatch between the two terms, they can be combined together which is done through a function called combine\_mTerms(). The omitted letter is represented as 2 in the array as shown in the third minterm in the above table. The new simplified terms are stored in another array called new\_terms.

As the first two rows has been simplified, they need to be indicated, another column has been added to indicate so which is the Taken colum, 1 means this minterm has been simplified or taken while 0 means the minterm is not simplified yet and needs to be added to the new\_minterms array if not simplified.

As the array’s number of rows is constant, 8 rows, the program needs to keep track of the actual number of minterms. Therefore, a k variable is passed by reference to the function update\_mTerms() to update it for future usage like printing and updating.

1. **Repeating the simplification process**

The new array is then passed again to the same function to simplify the minterms. This is done 3 times to guarantee all minterms are simplified to the least number of minterms and letters. The update is done three times because each minterm has 3 letters. Assuming the extreme case to truncate all terms. Each term needs three times to truncate all letters.

1. **Printing the final simplified minterms**

The minterms are then printed using print\_mTerms()

**Instructions for usage:**

1- Enter the minterms of the 3-variable Kmap only as comma separated values.

e.g. 1, 3, 6

2- If your number of minterms is equal or less than seven you have to do it in the first trial, because if it is more than 8 the program will ask you to enter the index of the minterm you want removed; however this will continue until the number of minterms is exactly 8 so you don’t the option to go below that any more.

3- If the value of a certain minterm is not between 0 and 7 enter the value of that minterm again.

4- Note that any repeated minterm is the user-responsibility.

e.g. if you enter 4 eight times, you will not be allowed to enter any minterms and the k-map will still only have one minterm.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |