CSCE230102 - Digital Design I

Spring 21

Professor:

Dr. Mona Farouk

Students:

Ahmed Essam - Sherif Sakran

Program objective:

This c++ program finds the simplest expression of a 3-variable function given its *minterms* using the *k-map* approach.

Summary:

The *k-map* allows us to find the most possible groups of 1's as powers of 2 which lead to implicants' simplifications, and here is the starting point of the program. We considered all possibilities of the groups of powers of 2 that would give implicants. Then we gave priority to prime implicants with the lowest number of variables by considering the concept of *logical adjacency* then excluded any minterm that does not provide any new information to the function. Therefore, we could obtain the function in its simplest form.

Program description:

The program consists of 2 main parts. At first, we receive the minterms from the user and validate them, then we store them in an array which will be used afterwards.

Part I

This part is used for generating all possibilities of implicants in a vector of vectors of strings. The choice of vector of vectors was important in our program since it will be needed in part 2 to solve some problems that we faced. In this part, we use the function "possibilities" that *pushes* all possible implicants in our main vector of vectors of strings "farr". We used the vector of strings "insert" for a simpler insertion into our main vector, "farr". By taking the array of minterms as a parameter to the function as well, we could generate every single possibility of implicants as the following:

• 8 minterms: the function is always true. That is, a "tautology".

• 4 minterms: we have 2 rows of 4 *adjacent* minterms, and we have 4 squares.

• 2 minterms: we have 12 implicants.

• 1 minterm: we have 8 implicants.

Our knowledge from the *discrete mathematics* course helped us not to miss any possibility by using the *combinations* concept, for example the implicants of 2 minterms was calculated as following:

3 *literals* (6 inputs) and we need *sets* of length 2 = 6C2 = 15 then we excluded the three expressions of the same literal (the variable AND its inversion) to end up with 12 implicants.

Part II

In this part, we refine those possibilities that we generated from the "possibilities" function. This is done as the following:

• The function "first_deletion" is implemented to remove the implicants from our main vector "farr" that are dispensable without affecting the definition of the function. For example, the function

F = A + AB include the implicant AB which is not important in our implementation because we can see in the k-map of the function that the minterm A was obtained by including the minterm AB. Therefore, summing AB with A will just make the function more complex without any addition. In other words, the function will be true or false as long as A is true or false respectively regardless the value of B.

• The function "second deletion" is implemented for a specific purpose that could be understood through the following example (figure 1). Let F = AB' + B'C + A'C

From the k-map, it could be explained that the possible groups of minterms that gives the highest power of 2 are 3 implicants. Therefore, we need to take only the *essential prime implicants*, and that could be done by excluding the term that could be understood from the rest of the terms. In other words, the implicant B'C (shown by the violet rectangle) is not essential because the B' exists in AB' and C exists in A'C.

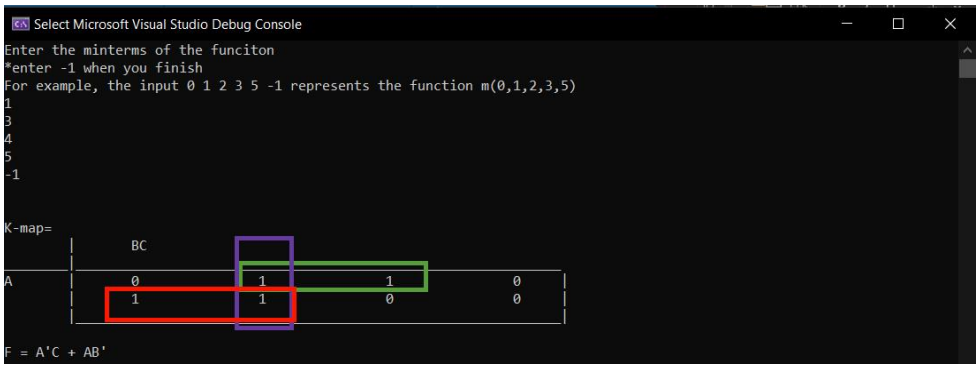

Figure 1

After refining our possibilities through the properties of the k-map, we have tested our program on a lot of different cases, and all worked well and produced the expected results.

Test cases: