KMAP PROJECT Report

Digital Design I

Mohamed Ayman Mohamed 900182267

AbdelRahman Fawzy Shabaan 900183004

The Design of our Code:

We have tried to create a useful code using classes and structs. First, we were thinking how each cell can be linked with other cells using a polynomial complexity, so we have created a C struct called cell having the identities of (int b(to make comparisons with cells), int place(the decimal name), a string of binaries(to make comparisons with each other without any errors), and, certainly, the bool state. The reason for decimal and state is that when the user enters the miniterms (in decimals), the state of the cell will be changed directly to true. After knowing the usefulness of the cell, we need to create a class that can implement the whole process (input, output, removing the duplicates and the expression) using object oriented programming to make it easier for the reader of the code. The class Kmap_Process has three methods and seven members. First, we need to take from the user the miniterms (it happens in the input method) and we defined each binary with each cell to make it easier in the comparison (we will see soon in my explanation). In the Kmap process, we convert each decimal to binary and use the string binary and define each cell state. Simple, right! Now, we are going to explain the most interesting and crucial method. It is to generate the kmap. We have 5 generic conditions. If the "true" state has appeared one time only, two times, four times with different shapes (1*4 or 2*2), we can get the all possibilities, but the problem is that we have many duplicates. Here is the part of removing duplicates comes. We have a member in the class (vector of vectors from type struct cell) - I think it is more useful than multi-dimensional arrays because we do not know the size of arrays, so vectors are much better. We have know vector of implicants and inside each implicant, we have a myriad of cells. Now, we need to compare. You observe that we need 4 loops to make this comparison. After the comparison, we need to make the expression of unduplicated implicants, so the Expression function does so.

The Problem of our code:

We have successfully rendered our programs cope with the all conditions. So **NO problems our code encounter.**

Instruction how to build and use the program:

We endeavored not to make a complicated program that help the user understand them with 100%. What you need to do is just following the instructions.

1. You need to have any type of debugger.
2. Open the source code.
3. Execute
4. After then, you need to enter the number of miniterms(you know the max is 8 and the min is 0)
5. After then you will enter your miniterms.
6. You will see the KMAP and the most simplified expression for this kmap.