Project 1

Ahmed Jaheen

900212943

Digital Design 1

28th of October 2022

The American University in Cairo

Author’s Note:

This project is prepared for CSCE2301, section 02, taught by Dr. Mona Farouk

Program Design

Our program design consists of 3 main stages. First, the program reads and validates the file input by the user. Second, the program takes the function and prints the write the K-map to better representation. Third and finally, the program uses an algorithm for minimization to generate the simplified Boolean expression of the function by the input of the user, then output the file in the same directory.

1. Reading and Validating User's input

At the beginning of our program, the user is asked to enter the path of the file including the number of test cases and the cases itself number. The program then validates the duplicated numbers in an unordered-map then return it in a vector to start work on the min-terms between 0 and 3 for 2-Variables or between 0 and 7 for 3-Variables because as per the program instructions, because the maximum term we can get from a 2 variables Boolean function is 3 while we can get from a 3 variables Boolean function is 7 and of course the minimum is 0. Then, these min-terms are populated in a vector to save them for future operations, and the program read the number of test cases to do a loop, and also read the first digit in each line from line   to know the number of variables. Finally, it will output a file in the same path of the input file with the minimization and the K-map.

2. Represent the function in a K-map

To represent the function in a K-map, we used a nested for loop. The outer loop is responsible to loop on the 4 places of 2-Variables or the 8 places of the 3-variables K map while the inner loop is responsible for iterating on the vector that holds the min-terms of the Boolean function. In the inner loop, the program checks if the min-term in this current loop is consistent with the current position in the K-map. If the answer is yes, then "1" is printed in this place, and if the answer is not, the program continues. If the program exits the inner for loop without printing a "1" in the current position, then "0" is printed instead.

3. Minimizing the function and generating the simplified Boolean expression

In this part, we will explain the algorithm used first then we illustrate how we implement this algorithm to get the simplified Boolean expression.

Now we have the K-map, and we need to generate the simplified Boolean expression. We used a popular method named Quine-McCluskey Minimization Technique or the Tabular Method. In the Tabular method, we can even solve more complex problems than a 3-variables Boolean function. We follow the steps of the tabular method. First, we group our min-terms depending upon the number of 1's in the Boolean representation of the min-terms. Then, we define a matched pair as 2 min-terms that only differ in one bit. We put a '-' in the place of this bit and save its index. Therefore, we are looking to n group and n+1 group and find the matched pairs from our min-terms. We collect the matched pairs from group i and i+1 in a new group numbered by n and we change the state of all the min-terms that we include in our matching pairs as "taken" for further operations. So, our groups now in this step are only 3 groups. Group 0 includes the matching pairs from 0 1's group and 1 1's group, and so on. We repeat this same step of match pairing again, so now we have only 2 groups and each one has 4 min-terms which were match paired through the previous operations. In this step, the algorithm ends and it is time to check the status of the min-terms and see if there is an "Untaken" min-term. The "Untaken" min-term is actually a prime implicant and cannot be ignored in our simplified Boolean function. Then, we need also to check if there are any essential prime implicants. Thos implicants are the ones that includes 1s that are not included in any other implicant.

To implement this algorithm, we represent the min-terms entered by the user as a struct that has a number (int), string (the corresponding binary of the decimal digit), and a Boolean variable to determine if this min-term is taken in our pair-matching processes or not. Moreover, we create more structs to be used in the processes of match pairing as in each process, we have doubled the terms in the previous process. Then, we assign to each min-term to its binary representation. We use vectors as our data structure to store the min-terms in groups that determine their number of ones. We start the process of match pairing; we loop on our table. The most outer loop iterates on the 3 groups of ones. The second loop iterates on the min-terms in the i group and the third loop iterates on the min-terms of the i+1 group and the fourth loop iterate on the binary bits to compare. If only one difference is found, we save the index of this bit to replace it with '-' and we change the status of these two min-terms as taken. They are now match paired. Finally, we push these new match-pairs back in our vector of vectors to be used in the next process of match pairing. The next processes of match pairing are exactly the same in operations. Please refer to the full code. We then check the "untaken" min-terms in all the previous operations and push them back in our prime map. After that, we determine the essential prime implicant and then we just replace our simplified Boolean equation from 0's and 1's to literals. Finally, we output the file in the same directory