BOOLEAN CIRCUIT SIMPLIFICATION AND CIRCUIT DRAWING IN SVG FORMAT

POOJA DASHOTTAR (510514057)

ALOKEDIP CHOUDHURI (510514064)

Under guidance of Prof. Apurba Sarkar

Department of Computer Science and Technology

IIEST, Shibpur

INTRODUCTION

In Boolean algebra, **circuit minimization** is the problem of obtaining the smallest logic circuit (Boolean formula) that represents a given Boolean function or truth table .There are some effective techniques by which we can minimize or simplify the Boolean expression. Like Karnaugh maps and Tabular-method(Quine–McCluskey algorithm). The problem with having a complicated circuit (i.e. one with many elements, such as logical gates) is that each element takes up physical space in its implementation and costs time and money to produce in itself. Circuit minimization may be one form of logic optimization used to reduce the area of complex logic in integrated circuits. Beside this if we are given a Boolean expression and if we have to know how many logic gates we need or if we

have to design the circuit then this technique is very useful in this current era of digital circuits and VLSI design. But if we are provided with some very complex Boolean expression which consists of a large number of min terms then it will be very difficult to design the circuit with optimum cost. For that we have to simplify the given Boolean expression and then we can be able to draw the circuit efficiently.

SIMPLIFICATION OF AN BOOLEAN EXPRESSION

There are various methods by which we can simplify the Boolean expression. Among them two methods are mostly used. They are:

1. Karnaugh maps
2. Quine–McCluskey algorithm

But among those above methods we generally use the “Karnaugh Map” when there are atmost four different variables in the Boolean expression like F (a, b, c, d). But if we have a Boolean expression in which we have more than four variables then to follow the karnaugh map will become very difficult for a human being and also in the computer coding part implementation of Karnaugh map is very difficult than Quine-McCluskey method. As Quine-

McCluskey method is a generalized algorithm for simplification of Boolean Expression So while simplifying the Boolean expression we will use Tabular-Method (Quine-McCluskey Method).

ALGORITHM FOR CONVERTING THE GIVEN BOOLEAN EXPRESSION INTO A LOGICAL CIRCUIT:

To convert the given Boolean expression into a logical circuit, consist of minimum number of logic gates (OR Gate and AND Gate) we have followed the following steps sequentially:

Algorithm:

1. User asked to enter the number of variable that he/she want to simplify
2. User asked to enter the total number of min term corresponding the expression.
3. User asked to enter the min terms.
4. Applying Quine-McCluskey (Tabular-Method) we have generated a simplified Boolean expression from the given set of min terms.
5. After getting the simplified expression we convert the expression into legal infix expression and then convert the infix expression into the postfix expression by Stack implementation.
6. After getting the postfix expression we have the expression where the precedence of the operator(~✉*✉+)is maintained. Then we evaluate the postfix expression and while doing that we draw the logic circuit by applying proper rule for drawing any image in SVG.

Quine –McCluskey Algorithm

The corresponding algorithm to determine the simplified expression from a given Boolean expression is as follows:

1. Group binary representation of the min terms according to the number of 1's contained. This is done by grouping the min terms into the five sections (only for 4 variables) The first section contains the number with no 1's in it. The second section contains those numbers those numbers with only one 1's and so on.

2. Any two min terms which differ from each other by only one variable can be combined, and the unmatched variable removed and replaced with a dash (-). The min term in one section is compared with those min terms in the very next section.

3. After combining those particular min terms, we make another column and store those combined min terms and repeat the step 2 until we will get those min terms which does not have difference only in one bit.

4. While checking a pair of min terms if they can be combined then we will mark those pairs. After finishing those comparison, we will get those last min terms and those unmarked ones.

5. The Unchecked Terms in the table will form essential prime implicants. The sum of the prime implicants gives a simplified expression for the function which has been simplified and those unchecked entries are the terms left to formulate the function.

6. **Selection of prime implicants:** After getting all those prime implicants we have to choose those prime implicants that can simplify the given expression. To choose those particular prime implicants we form a table where the columns are representing those given min terms and the rows are representing the total implicants that we have so far in our algorithm. Then we give check mark to the boxes whose column numbers (representing the min terms are combined) and we choose those columns that have minimum number of check mark and those columns are the essential prime implicants. And in this way we will generate the simplified Boolean expression.

COMPLEXITY:

Although more practical than “Karnaugh Mapping” when dealing with more than four variables, the “Quine–McCluskey” algorithm also has a limited range of use since the problem it solves is **NP-hard**:

The **runtime of the “Quine–McCluskey”** algorithm grows exponentially with the number of variables. It can be shown that for a function of n variables the upper bound on the number of prime implicants is $\mathbf{O(3^n/n)}$.

Example:

If $n = 32$ there may be over $6.5 * 10^{15}$ prime implicants.

CONCLUTION:

As circuit simplification and circuit satisfiability is NP hard problem so the complexity of the algorithm increases exponentially with increasing the no of variables. Beside the complexity of this algorithm in the first step of the algorithm the user has to give the detail information of the expression if the user simply five an expression then this algorithm will not work. This can be a back side of this overall algorithm.