

VLSI CAD Project Report

Quine McCluskey Procedure for Minimization of SOP expressions (2-level logic minimization)

AR Athul 140100111

Shyam Ramamoorthy 140070042

Achyutha Krishna Koneti 140070053

Sujay Mundada 140070020

Description of flow

The parent GUI expects the number of variables(theoretically no limit) and the minterms and don't care cases in the respective boxes. It is parsed accordingly and minimize() function is called with necessary arguments. It converts minterms to binary strings. This function in turn has 2 logic modules. First calling __get_PI() to get the prime implicants. It gives the prime implicant chart in a structural way, using group combinations. The next module is the Patrick's method for finding the essential prime implicants. The prime implicants are converted to 2-tuples and passed. It uses the systematic combinations of exhaustive prime implicants and chooses the minimum cost combination using the standard rule of assigning cost 1 to not gate and 2 to other 2 gates.

The second problem is the joint minimization of 2 functions given in the same input format. It uses a mixed strategy. It gets the prime implicants of the 2 functions using repeated calls of __get_PI() and then minimize the joint complexity of the set cover of the two functions. This will give the shared minimal essential prime implicants of the combination.

In both the cases, the output would be a readable format of the minimized function(s). In the first section, the essential implicants in the 2-tuple format are displayed as well.

Functionality for clearing the GUI is also given.

Instructions for running the code:

Libraries enaml, qtapplication have to be installed

Run software.py

GUI will open

There are two modes, one in each tab:

i) Mode 1: "Minimization Tab"

This tab is for minimization of a single function only.

Inputs:

1. Number of variables

This is the number of variables of the function to be minimized. It should be entered in integer form.

Eg. “3” means that the function will have 3 variables and they will be named A, B and C

2. Ones

These are the minterms corresponding to the 1s of the function.

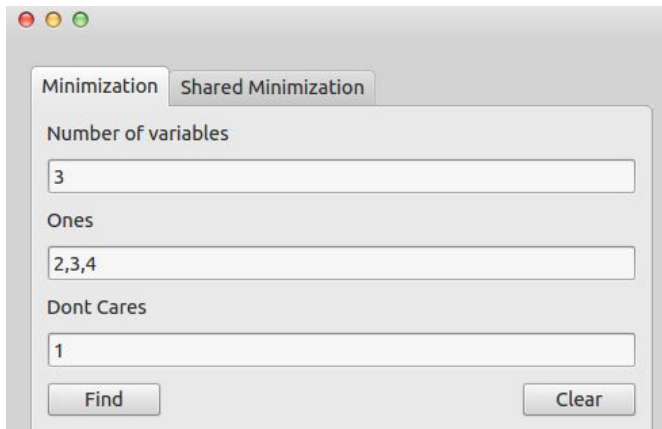
Eg. If a functions minterms are 0, 4, 5, 6 then enter “0,4,5,6” into this entry.

3. Dont Cares

These are the minterms corresponding to the don’t cares of the function.

Eg. “1”

Then click on “Find”



The image shows a software window titled "Minimization" with a sub-tab "Shared Minimization". It contains three input fields: "Number of variables" with the value "3", "Ones" with the value "2,3,4", and "Dont Cares" with the value "1". At the bottom, there are two buttons: "Find" and "Clear".

The minimized function will be displayed in the “Functions” output.

Minimization Shared Minimization

Number of variables
3

Ones
2,3,4

Dont Cares
1

Find Clear

Prime Implicants
(2,1),(4,0)

Function
B.A'+C'.B'.A

ii) Mode 2 (Simultaneous minimization of two functions)

The inputs in this case are the same as in the above mode, except that they should be inputted separately for two different functions. The output is also displayed as before. Here, the output of the joint minimization may or may not be the same as the outputs of the individual minimizations: this is because in joint minimization, we try to reduce the overall cost of the two minimizations instead of reducing the individual costs separately.

Minimization Shared Minimization

Number of variables
3

Minterm_mins of function 1
0,1,2,6

Minterm_mins of function 2
0,1,2,5,6,7

Dont Cares of function 1

Dont Cares of function 2

Find Clear

Function 1
B'.A'+C'.B

Function 2
B'.A'+C.A+C'.B

Test Cases:

We ran over 100 test cases for each of the single minimization and joint minimization. Some of them are:

1. Minimization of a single function

I.

3 variables A, B, C

Ones: 2,4,6,7

Dont cares:3

Function: $AC' + B$

ii

3 variables A, B, C

Ones: 1, 6

Dont cares: 0, 2, 3, 4, 5

Function: $A' + C'$

lii

4 variables A, B, C, D

Ones: 8, 11

Dont cares: 1, 14

Function: $D.C.B'.A + D'.C'.B'.A$

lv

5 variables A, B, C, D, E

Ones: 8, 11, 19, 26, 27, 29

Dont cares: 0, 10, 20, 30

Function: $E.D.C'.A + E'.C'.B.A' + D.C'.B + E.D'.C.B.A$

v

11 variables A, B, C, D, E, F, G, H, I, J, K

Ones: 0, 5, 11, 14, 19, 29, 41, 44, 46, 55, 62, 78, 79, 80, 85, 87, 92, 101, 110

Dont cares: 1, 2, 3, 12, 16, 17, 32, 35, 37, 4, 51, 66, 68, 75, 81, 95, 120

Function: $E'.D.C.B + E.D'.C.B.A + E.D.C'.A' + E.D.B'.A + D'.C'.B'.A + E.D'.C.B'.A' + E'.D.C.B.A' + E'.D'.C.B.A + E.D.C.B + E'.C.B.A' + E.D'.C'.B.A' + E.C.B'.A + D'.B'.A' + E.C'.B'$

2. Simultaneous Minimization of Two functions

i)

3 variables A, B, C

Function 1 minterms: 0,1,2,6

Function 1 don't cares: None

Function 2 minterms: 0,1,2,5,6,7

Function 2 don't cares: None

Function 1: $A'B' + BC$

Function 2: $A'B' + AC + BC'$

We can see that the total number of product terms required to express both the functions is 4. However when we do individual minimization of the same functions, they are minimized to $A'B'+BC'$ and $AB + B'C + A'C'$ respectively and we need a total of 5 products to express both the functions like this. Thus join minimization has allowed us to express both the functions using fewer products than by minimizing them individually.

ii)

3 variables A, B, C,

Function 1 minterms: 0,1,2,4,6,12,13,14,9,10

Function 1 don't cares: None

Function 2 minterms: 0, 1, 4, 8, 10, 12

Function 2 don't cares: None

Function 1: $A'B'C' + BD' + CD' + AC'D$

Function 2: $A'B'C' + ABD' + C'D'$

We can see that the total number of product terms required to express both the functions is 6. However when we do individual minimization of the same functions, they are minimized to $ABC' + A'D' + B'C'D + CD'$ and $A'B'C' + AB'D' + C'D'$ respectively and we need a total of 7 products to express both the functions like this. Thus as in the earlier case, join minimization has allowed us to express both the functions using fewer products than by minimizing them individually.

References:

We looked at this link: <https://github.com/tpircher/quine-mccluskey> to get a general idea of how to find the prime implicants in python. However, our code was written on our own and we only obtained a general philosophy on how to approach the problem from the reference. Petrick's method and extension of the code to joint minimization is not mentioned in this reference and we had to attack those problems on our own.

Items contributed by each team member:

The work was more or less evenly distributed and each task was worked on by multiple people together, rather than a single person. This was the rough distribution of work:

Finding the prime implicants from the function, generation of prime implicant chart, converting prime implicants to the value+mask form, extension of the 1 function code to the case of joint minimization of two functions,:

Shyam (140070042) and Achyutha (140070053)

Petrick's method for finding essential prime implicants, finding covers of the function, calculation of costs of different minimizations in order to determine the best one converting the final output to a readable form:

Athul (140100111) and Sujay (140070020)

The GUI was mostly developed by Athul (140100111) and this report and the presentation were authored by Shyam (140070042)