Date: 21/11/2016

Quine McClusky Algorithm for Multiple Functions with Branch and Bound

Yash Bhalgat | Kalpesh Patil | Navjot Singh
13D070014 | 130040019 | 130110071
Guide: Prof. Patkar

Problem Statement:

The synthesis and optimization of problems that are encountered in the practice of digital design very often involve functions to multiple outputs. It can be seen that a function as a vector of functions to a single output, that is:

$$F(x_0, \ldots x_{n-1}) = [f_0(x_0, \ldots x_{n-1}) \cdots f_{k-1}(x_0, \ldots x_{n-1})]$$

The Quine-McCluskey method can be extended to address the simultaneous synthesis of more than one functions. This involves some changes to both the phase of expansion, and to that of coverage.

Algorithm:

For example, consider the following function to three outputs:

$$\begin{aligned} F(x,y,z,w) &= [f_0(x,y,z,w), f_1(x,y,z,w), f_2(x,y,z,w)] \\ &= \Sigma_0\{0,2,4,5,6,7\}, \\ &\quad \Sigma_1\{0,4,5,7,10,11,14,15\}, \\ &\quad \Sigma_2\{2,6,10,11,14,15\}, \end{aligned}$$

The set of all minterms is:

$$\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 = \{0,2,4,5,6,7,10,11,14,15\}$$

From this set we proceed to encoding and to the construction of the masks. These masks will be formed by three bits in which the most significant corresponds to F0 and the least significant to F2. The minterm 0, for example, is coded on 4 bits as 0000 and, since that belongs both to F0 and F1, it has associated the mask 110. And also take the AND operation of the f0-f1-f2 mask.

The table below explains the procedure of all the encodings and the corresponding masks:

Date: 21/11/2016

	$xyzw$	$f_0f_1f_2$	
0	0000	1 1 0	◇
2	0010	1 0 1	◇
4	0100	1 1 0	◇
5	0101	1 1 0	◇
6	0110	1 0 1	◇
10	1010	0 1 1	◇
7	0111	1 1 0	◇
11	1011	0 1 1	◇
14	1110	0 1 1	◇
15	1111	0 1 1	◇

	$xyzw$	$f_0f_1f_2$	
0,2	00–0	1 0 0	◇
0,4	0–00	1 1 0	
2,6	0–10	1 0 1	
2,10	–010	0 0 1	◇
4,5	010–	1 1 0	
4,6	01–0	1 0 0	◇
5,7	01–1	1 1 0	
6,7	011–	1 0 0	◇
6,14	–110	0 0 1	◇
10,11	101–	0 1 1	◇
10,14	1–10	0 1 1	◇
7,15	–111	0 1 0	
11,15	1–11	0 1 1	◇
14,15	111–	0 1 1	◇

	$xyzw$	$f_0f_1f_2$
0,2,4,6	0––0	1 0 0
2,6,10,14	––10	0 0 1
4,5,6,7	01––	1 0 0
10,11,14,15	1–1–	0 1 1

Covering problem:
The essential column, row dominance and column dominance are modified in case of optimization of multiple functions.

	f_0						f_1								f_2						
	0	2	4	5	6	7	0	4	5	7	10	11	14	15	2	6	10	11	14	15	
P_0	×		×				×	×													1
P_1		×			×										×	×					1
P_2			×	×				×	×												1
P_3				×		×			×	×											1
P_4										×				×							1
P_5	×	×	×		×																1
P_6															×	×	×		×		1
P_7			×	×	×	×															1
P_8											×	×	×	×			×	×	×	×	1

Consider this implicant table. The ‘ones’ on the right are the costs of all the implicants. We need to minimise the total cost of the function.

The principles upon which the cover is solved are those already seen: **essentiality, dominance of row and column dominance**. These concepts, however, require some changes to take account of the different nature of the problem.

Date: 21/11/2016

Reduction rules

1. Essential row rule:

A given implicant, in fact, may be essential only for some of the functions: in this case the corresponding row must not be deleted and must be eliminated only the columns corresponding to the functions for which the implying concerned is essential. If these implicants were then again chosen for the coverage of other functions, would have no impact on the overall cost of the cover.

2. Cost: To account for the fact explained above, each implicant is associated with a cost. At first, the cost is equal to 1 for all implicants. Each time a implicant is selected but remains in the table, cost changes to 0.

3. Column dominance: The column dominance principle must be redefined stating that it is applicable only between columns (ie minterms) of the same function.

4. Row dominance: A line Pi dominates the line involving the P, if

a) it covers all the minterms that are covered in P, plus at least one, and
b) the cost of P; is less than or equal to the cost of Pi

These steps are followed as follows:

$C(f_0) = \emptyset \qquad C(f_1) = \{P_0, P_8, P_3\} \qquad C(f_2) = \{P_8, P_1\}$

	f_0						f_1		f_2		
	0	2	4	5	6	7	5	7	2	6	
P_0	×		×								0
P_1		×			×				×	×	1
P_2			×	×			×				1
P_3				×		×	×	×			1
P_4								×			1
P_5	×	×	×		×						1
P_6									×	×	1
P_7			×	×	×	×					1

Cost of P0 = 0 and columns for P8 is removed.

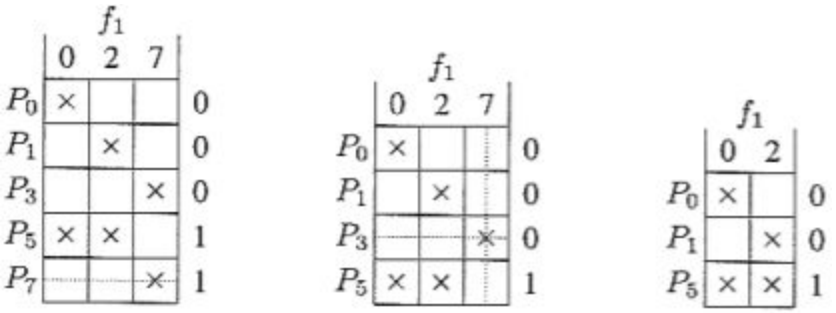
$C(f_0) = \{P_3\} \qquad C(f_1) = \{P_0, P_8, P_3\} \qquad C(f_2) = \{P_8, P_1\}$

	f_0						f_1		f_2		
	0	2	4	5	6	7	5	7	2	6	
P_0	×		×								0
P_1		×			×				×	×	1
P_2			×	×			×				1
P_3				×		×	×	×			1
P_5	×	×	×		×						1
P_7			×	×	×	×					1

	f_0						
	0	2	4	5	6	7	
P_0	×		×				0
P_1		×			×		0
P_2			×	×			1
P_3				×		×	0
P_5	×	×	×		×		1
P_7			×	×	×	×	1

Date: 21/11/2016

$$\mathcal{C}(f_0) = \{P_3\} \qquad \mathcal{C}(f_1) = \{P_0, P_8, P_3\} \qquad \mathcal{C}(f_2) = \{P_8, P_1\}$$



$$\mathcal{C}(f_0) = \{P_0, P_1, P_3\} \qquad \mathcal{C}(f_1) = \{P_0, P_3, P_8\} \qquad \mathcal{C}(f_2) = \{P_1, P_8\}$$

$$\mathcal{C}(F) = \mathcal{C}(f_0) \cup \mathcal{C}(f_1) \cup \mathcal{C}(f_2) = \{P_0, P_1, P_3, P_8\}$$

When further reduction is not possible (cyclic tables), Branch and bound algorithm is used to find the minimum cover. For this case of multiple outputs, the branch and bound is suitably modified to include all the reduction rules above.

Contributions:

All the three members have contributed equally to the implementation and brainstorming in all the sub-parts of the project. But mainly, following are the individual contributions:
Implementation of the implicant expansion done by Kalpesh,
Implementation of the Branch and bound done by Yash,
Reduction rules implemented by Navjot.

Other important part such as the Solution class and finally printing the output in a desired format were implemented togetherly.

References:

[1] Reti Logiche, Bolchini (originally in Italian, translated important parts for study purposes)
[2] Logic Synthesis and Verification Algorithms, Hacthtel and Somenzi