TABULATION METHOD FOR MINIMISATION
(QUINE-McCLUSKEY METHOD)

Team Members:

1. **Harshit Tarsariya(CE 117)**
2. **Vishant Vaghani (CE 120)**

Subject:

DSA (Data Structures & Algorithms)

Semester:

3rd (III)

Batch:

D3

Date:

27/09/2019

DESCRIPTION

Tabulation method is also known as Quine-McCluskey method .This method is used for minimising or reducing boolean functions which have any numbers of variables .

We can use Karnaugh map method for reducing the Boolean fuction but this method has an advantage over Karnaugh maps when a large number of inputs are present (more than five variables). With more inputs, pattern recognition in k-maps can be tedious or sometimes even impossible. This tabulation method does not require pattern recognition. It consists of two steps:

1. Finding all prime implicants of the function .

2. Selecting a minimal set of prime implicants of the function as an essential prime implicants.

In addition, the K-map algorithm is not as straight forward to program. Therefore tabulation method is better suited for programming and thus can solve any function having any number of variables and we can use direct application for minimisation of long function. It can also help for decreasing number of gates in integrated circuits.

PROBLEM

Function:

f(A, B, C, D)= A'B'C'D' + A'B'CD +A'BC'D + A'BCD' + A'BCD + AB'CD'+ABC'D+ABC'D+*<u>A'B'CD'+AB'C'D+ABCD</u>*

=∑m(0,3,5,6,7,10,12,13) + ∑d(2,9,15)

Number of variables=4

Number of minterms=8

Number of don't care=3

Now, we have to convert this minterms (including don't cares) into binary form. for binary conversion we have to use bits equals to number of variables.

<u>Example</u> : for 03 binary is 0011

for 15 binary is 1111

SOLUTION STEPS

Step 1 : Divide all the minterms (and don’t cares) of a function into groups

MINTERMS	A	B	C	D
0	0	0	0	0
2	0	0	1	0
3	0	0	1	1
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
9	1	0	0	1
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1
15	1	1	1	1

GROUPS	MINTERMS	A	B	C	D	MARK
G0	0	0	0	0	0	
G1	2	0	0	1	0	
G2	3	0	0	1	1	
	5	0	1	0	1	
	6	0	1	1	0	
	9	1	0	0	1	
	10	1	0	1	0	
	12	1	1	0	0	
G3	7	0	1	1	1	
	13	1	1	0	1	
G4	15	1	1	1	1	

Step 2: Merge minterms from adjacent groups to form a new implicant table

GROUPS	MINTERMS	A	B	C	D	MARK
G0	0	0	0	0	0	√
G1	2	0	0	1	0	√
G2	3	0	0	1	1	√
	5	0	1	0	1	√
	6	0	1	1	0	√
	9	1	0	0	1	√
	10	1	0	1	0	√
	12	1	1	0	0	√
G3	7	0	1	1	1	√
	13	1	1	0	1	√
G4	15	1	1	1	1	√

GROUPS	MINTERMS	A	B	C	D
G0’	0 , 2	0	0	-	0
G1’	2 , 3	0	0	1	-
	2 , 6	0	-	1	0
	2 , 10	-	0	1	0
G2’	3 , 7	0	-	1	1
	5 , 7	0	1	-	1
	6 , 7	0	1	1	-
	5 , 13	-	1	0	1
	9 , 13	1	-	0	1
	12 , 13	1	1	0	-
G3’	7 , 15	-	1	1	1
	13 , 15	1	1	-	1

Step 3: Repeat step 2 until no more merging is possible

GROUPS	MINTERMS	A	**B**	**C**	D	MARK
G0’	**0 , 2**	**0**	**0**	**-**	**0**	
G1’	**2 , 3**	**0**	**0**	**1**	**-**	√
	2 , 6	**0**	**-**	**1**	**0**	√
	2 , 10	**-**	**0**	**1**	**0**	
G2’	**3 , 7**	**0**	**-**	**1**	**1**	√
	5 , 7	**0**	**1**	**-**	**1**	√
	6 , 7	**0**	**1**	**1**	**-**	√
	5 , 13	**-**	**1**	**0**	**1**	√
	9 , 13	**1**	**-**	**0**	**1**	
	12 , 13	**1**	**1**	**0**	**-**	
G3’	**7 , 15**	**-**	**1**	**1**	**1**	√
	13 , 15	**1**	**1**	**-**	**1**	√

GROUPS	MINTERMS	A	B	C	D
G1”	2 , 3 , 6 , 7	0	-	1	-
	2 , 6 , 3 , 7	0	-	1	-
G2”	5 , 7 , 13 , 15	-	1	-	1
	5 , 7 , 13 , 15	-	1	-	1

Step 3: Repeat step 2 until no more merging is possible

GROUPS	MINTERMS	A	B	C	D	MARK	PRIME IMPLICANTS
G0”	0 , 2	0	0	-	0		A’B’D’
G1”	2 , 3 , 6 , 7	0	-	1	-		A’C
	2 , 10	-	0	1	0		B’CD’
G2”	5 , 7 , 13 , 15	-	1	-	1		BD
	9 , 13	1	-	0	1		AC’D
	12 , 13	1	1	0	-		ABC’

- **No more merging possible!**

Step 4: Put all prime implicants in a cover table (don’t cares excluded)

MINTERMS	0,2	2,3,6,7	2,10	5,7,13,15	9,13	12,13
0	1					
3		1				
5				1		
6		1				
7		1		1		
10			1			
12						1
13				1	1	1

Should not include don’t care

Step 5: Identify essential minterms, and hence essential prime implicants

MINTERMS	0,2	2,3,6,7	2,10	5,7,13,15	9,13	12,13
0	1					
3		1				
5				1		
6		1				
7		1		1		
10			1			
12						1
13				1	1	1

🟧 This colour represents **E.M.T.(ESSENTIAL MINTERMS)** which has only one **1** in row. The columns which includes this M.N.T. are called as **E.P.I.(ESSENTIAL PRIME IMPLICANTS)**.

Now simplified function is sum of E.P.I.:

F(**A,B,C,D)= A’B’D’ + A’C + B’CD’ + BD + ABC’**

SOLUTION

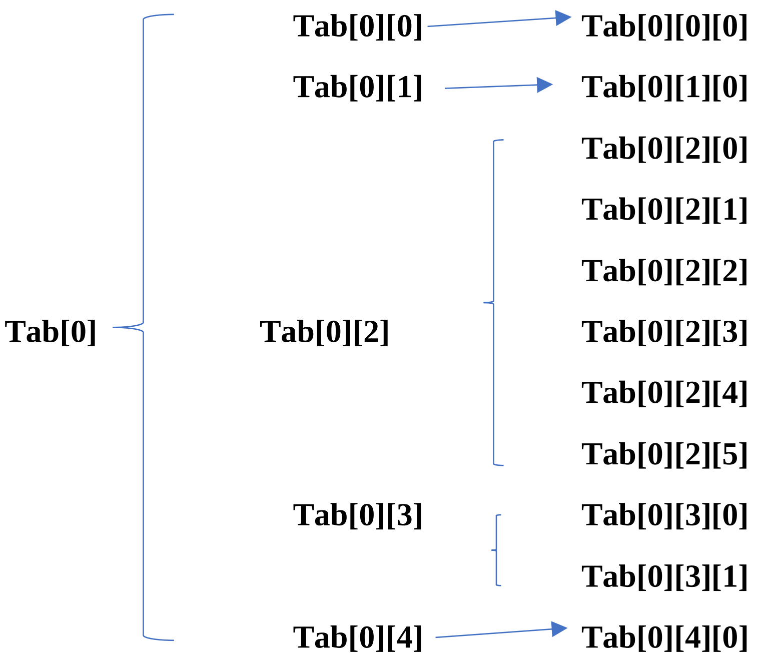
- **Binary array for converting minterms into binary code.**

i\j	0	1	2	3	4	5	6	7
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	**0**	**0**	**1**	**0**	**1**	**0**	**2**	**2**
3	**0**	**0**	**1**	**1**	**2**	**0**	**3**	**3**
5	**0**	**1**	**0**	**1**	**1**	**0**	**5**	**5**
6	**0**	**1**	**1**	**0**	**2**	**0**	**6**	**6**
7	**0**	**1**	**1**	**1**	**3**	**0**	**7**	**7**
9	**1**	**0**	**0**	**1**	**2**	**0**	**9**	**9**
10	**1**	**0**	**1**	**0**	**2**	**0**	**10**	**10**
12	**1**	**1**	**0**	**0**	**2**	**0**	**12**	**12**
13	**1**	**1**	**0**	**1**	**3**	**0**	**13**	**13**
15	**1**	**1**	**1**	**1**	**4**	**0**	**15**	**15**

Figure 1. The information of binary numbers are stored in the array Binary[][]

Here,0-3 number(in 1st row) represents binary of corresponding minterms given in 1st column.

- Number 4 represents total numbers of **1** in binary representation.
- Number 5 represents pairing of minterms(**1** for pairing and **0** for non-pairing).
- Number 6 and 7 contains minterms number.



Arrange the minterms into groups accordingto the number of 1s in their binary representation

Column 0

		A	**B**	**C**	D	
G0	**0**	**0**	**0**	**0**	**0**	√
G1	**2**	**0**	**0**	**1**	**0**	√
G2	**3**	**0**	**0**	**1**	**1**	√
	5	**0**	**1**	**0**	**1**	√
	6	**0**	**1**	**1**	**0**	√
	9	**1**	**0**	**0**	**1**	√
	10	**1**	**0**	**1**	**0**	√
	12	**1**	**1**	**0**	**0**	√
G3	**7**	**0**	**1**	**1**	**1**	√
	13	**1**	**1**	**0**	**1**	√
G4	**15**	**1**	**1**	**1**	**1**	√

Column 1

		A	**B**	**C**	D	
G0’	0 , 2	**0**	**0**	**-**	**0**	
G1’	2 , 3	**0**	**0**	**1**	**-**	√
	2 , 6	**0**	**-**	**1**	**0**	√
	2 , 10	**-**	**0**	**1**	**0**	
G2’	3 , 7	**0**	**-**	**1**	**1**	√
	5 , 7	**0**	**1**	**-**	**1**	√
	6 , 7	**0**	**1**	**1**	**-**	√
	5 , 13	**-**	**1**	**0**	**1**	√
	9 , 13	**1**	**-**	**0**	**1**	
	12 , 13	**1**	**1**	**0**	**-**	
G3’	7 , 15	**-**	**1**	**1**	**1**	√
	13 , 15	**1**	**1**	**-**	**1**	√

Column 2

		A	B	C	D
G1”	2 , 3 , 6 , 7	0	-	1	-
	~~2 , 6 , 3 , 7~~	0	-	1	-
G2”	5 , 7 , 13 , 15	-	1	-	1
	~~5 , 7 , 13 , 15~~	-	1	-	1

This is the coloumn representation in source code.

l	**Tab[i][j][k][l]**
0	Binary representation of indices. Positions of X are replaced by 2
1	
2	
3	
4	Number of 1s
5	‘1’ means have been paired off; ‘0’ means haven’t been paired off
6 . . . **(5+i)**	Positions of the ‘X’s
(6+i)	The original decimal index
(7+i) . . . **($6+i+2^i$)**	All the decimal indices included

Members of array Column[*i*][*j*][*k*][]

- **Implementation of pairing process**

L	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Tab[1][1][0][l]	0	0	1	2	1	0	0	2	2	3

l	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Tab[1][2][2][l]	0	1	1	2	2	0	0	6	6	7

After pairing 0 -> 1

L	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Tab[2][1][0][l]	0	2	1	2	1	0	0	2	2	2	3	6	7

Position of -

Addresses of unpaired PIs are recorded in the array PI_Index[][] (duplicates are omitted).

l	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Tab[2][1][0][l]	0	2	1	2	1	0	0	2	2	2	3	6	7

Is it 0?

PI	column	PI_INDEX
0 , 2	**TAB[1][0][0]**	**PI_INDEX[0]**
2 , 10	**TAB[1][1][2]**	**PI_INDEX[1]**
9 , 13	**TAB[1][2][4]**	**PI_INDEX[2]**
12 , 13	**TAB[1][2][5]**	**PI_INDEX[3]**
2 , 3 , 6 , 7	**TAB[2][1][0]**	**PI_INDEX[4]**
5 , 7 , 13 , 15	**TAB[2][2][0]**	**PI_INDEX[5]**

Has duplicate?

NO

PI_INDEX[i]={2,1,0};

Number_counter[k]++;

K=2,3,6,7

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Number_counter[i]	1	0	3	1	0	1	1	2	0	1	1	0	1	3	0	1

Omit Don’t care minterms

I	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Number_counter[i]	1	0	0	1	0	1	1	2	0	0	1	0	1	3	0	0

Set the value of NumberCounter[*i*] to ‘0’ (*i* is the Don’t-Care minterms).

- Find the minterms whose NumberCounter value is ‘1’.
- The addresses of the corresponding PIs that contain these minterms are recorded in the array essential_prime_implicants[][].

(0 , 3 , 5 , 6 , 10 , 12) following minterms which have value 1 in Number_counter[] array.13 &15 are not in this following minterms ,so we have to take PI who contains 13&15 both.

Here Essential prime implicants from prime implicants are:
0 , 2
2 , 3 , 6 , 7
5 , 7 , 13 , 15
2 , 3 , 6 , 7
2 , 10
12 , 13
5 , 7 , 13 , 15(this case for 13&15)

If we remove duplicates then new essential prime implicants table is:

E.P.I.	
2 , 3 , 6 , 7	A’C
5 , 7 , 13 , 15	BD
0 , 2	A’B’D’
2 , 10	B’CD’
12 , 13	ABC’

Simplified function is:

F(**A,B,C,D)=A’C+BD+A’B’D’+B’CD’+ABC’**

Testing with some examples

1) $F(A,B,C,D) = \sum(0,1,3,7,8,9,11,15)$

```
Input:      Enter the number of Variables
            4
            Enter the number of Min terms
            8
            Enter the  number of  don't cares(if present else 0)
            0
            Enter the min_terms
            0 1 8 9 3 7 11 15
            Reduced Equation
            B'C'+CD
            Press key to exit!!!
```

2) $F(A,B,C,D) = \sum m(4,6,9,10,11,13) + \sum d(2,12,15)$

```
Input:      Enter the number of Variables
            4
            Enter the number of Min terms
            6
            Enter the  number of  don't cares(if present else 0)
            3
            Enter the min_terms
            4 6 9 10 11 13
            Enter the  dont care
            2 12 15
            Reduced Equation
            AD+A'BD'+B'CD'
            Press key to exit!!!
```

<u>Conclusion:</u>

From this we can conclude that tabulation method is much easier to perform and doesn’t require any complex logic compared to the k-map method which require much pairing as the number of variables increases.