. **BOOLEAN MINIMIZATION PROGRAM**

A PROJECT REPORT

Submitted by

HARSH KUMAR – 16BCE0252

SHIKHAR SINGH-16BIT0111

KASALANATI PAVAN-16BCE0795

UNDER THE GUIDANCE OF

Prof. BALAKRUSHNA TRIPATHY

SENIOR PROFFESOR

SCHOOL OF COMPUTER ENGINEERING

DATA STRUCTURES AND ALGORITHMS (CSE 2003)

VIT UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

VELLORE ▪ CHENNAI

www.vit.ac.in

APRIL 2017

VIT UNIVERSITY VELLORE 632014

CERTIFICATE

This is to certify that the project work entitled “**BOOLEAN MINIMIZATION PROGRAM**” that is being submitted by “ **HARSH KUMAR**”, “**SHIKHAR SINGH**”,”**KASALANATI PAVAN**”for Digital Logic And Design (CSE 1003) is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place: Vellore
Date: 27-04-2017

Signature of the Student: Harsh kumar, Shikhar Singh, Kasalanati Pavan

Signature of the Faculty: Balakrushna Tripathy

ACKNOWLEDGEMENTS

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to ***Prof. Balakrushna Tripathy.*** for his guidance and constant supervision as well as for providing necessary information regarding the project & also for his support in completing the project.

We would like to express my gratitude towards my parents for their kind co-operation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

HARSH KUMAR

Reg. No. 16BCE0252

SHIKHAR SINGH

Reg. No. 16BIT0111

K PAVAN

Reg.No. 16BCE0795

ABSTRACT

The main objective of our project was to design a program that would simplify any Boolean expression. We accomplished this by writing a dynamic program in C. We used the Quine-McCluskey algorithm, also called the tabulation method. The Quine-McCluskey method is useful in minimizing logic expressions for larger number of variables when compared with minimization by Karnaugh Map or Boolean algebra. A brief introduction and the logic of this method are discussed following which the code have been provided.
The most commonly used method for Boolean minimization in industries is the Espresso Heuristic method, which is based on Heuristic methods. Initially, we had set out with the aim of making a program to implement this method too but over the course of the project, we realized that implementing it on C/C++ alone is very difficult and requires more experience to write the code for this method.

TABLE OF CONTENTS

Contents

INTRODUCTION

The Quine–McCluskey algorithm or the method of prime implicants is a method used for minimization of boolean functions. It was developed by W.V. Quine and Edward J. McCluskey in 1956. It is functionally identical to Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and it also gives a deterministic way to check that the minimal form of a Boolean function has been reached. It is sometimes referred to as the tabulation method. The method involves two steps: 1. Finding all prime implicants of the function. 2. Use those prime implicants in a prime implicant chart to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function. In this paper, we intend to discuss the Quine-McCluskey minimization procedure as well as provide the readers with all the simulation codes which are available on net in one single paper, highlighting the variations in each of the given codes implemented using a different computer language. The procedure which is discussed in the following section 2 and 3 has also been taken from the net and for that appropriate references have been given.

METHODOLOGY

QUINE-McCLUSKEY MINIMIZATION PROCEDURE

This is basically a tabular method of minimization and as much it is suitable for computer applications. The procedure for optimization as follows:

Step 1: Describe individual minterms of the given expression by their equivalent binary numbers.

Step 2: Form a table by grouping numbers with equivalent number of 1"s in them, i.e. first numbers with no 1"s, then numbers with one 1, and then numbers with two 1"s, … etc.

Step 3: Compare each number in the top group with each minterm in the next lower group. If the two numbers are the same in every position but one, place a check sign (□) to the right of both numbers to show that they have been paired and covered. Then enter the newly formed number in the next column (a new table). The new number is the old numbers but where the literal differ, an "x" is placed in the position of that literal.

Step 4: Using (3) above, form a second table and repeat the process again until no further pairing is possible. (On second repeat, compare numbers to numbers in the next group that have the same "x" position.

Step 5: Terms which were not covered are the prime implicants and are ORed and ANDed together to form final function. Note: The procedure above gives you the prime implicant but not essential prime implicant.

Example 1

Minimize the function given below by Quine-McCluskey method.

f(A,B,C,D)= $\overline{A}\overline{B}\overline{C}\overline{D}$+ $\overline{A}B\overline{C}D$+ $\overline{A}BC\overline{D}$+ $A\overline{B}\overline{C}D$+ $A\overline{B}C\overline{D}$+ $AB\overline{C}D$+ $ABC\overline{D}$+ ABCD+ $\overline{A}BCD$

Binary 0000 0101 0110 1001 1010 1101 1110 1111 0111 minterm 0 5 6 9 10 13 14 15

7 No of 1's 0 2 2 2 2 3 3 4 3 group 1 2 2 2 2 3 3 4 3

This can be written as a sum of minterms as follows:

$$f(A,B,C,D) = \sum m(0,5,6,7,9,10,13,14,15)$$

Step 1: Form a table of functions of minterms according to the number of 1‟s in each minterm as shown in Table E1.a

minterm	A	B	C	D	
* 0	0	0	0	0	
5	0	1	0	1	✓✓…
6	0	1	1	0	✓✓…
9	1	0	0	1	✓✓…
10	1	0	1	0	✓✓…
7	0	1	1	1	✓✓…
13	1	1	0	1	✓✓…
14	1	1	1	0	✓✓…
15	1	1	1	1	✓

} All numbers with no 1‟s in each minterm (a)

} All numbers with two 1‟s in each minterm

} All numbers with three 1‟s in each minterm

} All numbers with four 1‟s in each minterm

AB

CD 00 01 11 10

00 1 0 4 12 8

01 1 1 5 1 13 1 9

11 3 1 7 1 15 11

10 2 6 14 10

1 1 1

Table E1.a

Step 2: Start pairing off each element of first group with the next, however since m_0 has no 1‟s, it and the next group of numbers with one 1‟s are missing, therefore they cannot be paired off. Start by pairing elements of m_5 with m_7, m_{13}, m_{14}, and m_6 with m_7, m_{13}, m_{14}, and so on… If they pair off, write them in a separate table and ✓ the minterm that pair, i.e. m_5 and m_7 pair off 0101 and 0111 to produce 01x1, so in the next table E1.b under "minterm paired" we enter "5, 7" and under "ABCD" we enter "01x1" and place a ✓sign in front of 5 and 7 in

Table E1.a

Note: Each minterm in a group must be compared with every minterm in the other group even if either or both of them have already been checked✓.

minterms paired	A B C D	
5, 7	0 1 X 1 ✓	
5, 13	X 1 0 1 ✓✓	
6, 7	0 1 1 X ✓✓	
6, 14	X 1 1 0 ✓✓	
9, 13	1 X 0 1	………… (b)
10, 14	1 X 1 0	………… (c)
7, 15	X 1 1 1 ✓✓	
13, 15	1 1 X 1 ✓✓	
14, 15	1 1 1 X ✓✓	

Table E1.b

	Paired minterms from E1.b	A	B	C	D
… (d)	5,7 – 13,15	x	1	x	1
… (e)	6,7 – 14,15	x	1	1	x

Table E1.c

Step 3: Now repeat the same procedure by pairing each element of a group with the elements of the next group for elements that have "x" in the same position. For example, "5,7" matches "13,15" to produce x1x1. These elements are placed in table E1.c as shown, and the above elements in Table E1.b are ✓ checked. (The elements that produce the same ABCD pattern are eliminated.) Since 9,13 and 10,14 in Table E1.b do not pair off, they are prime implicants and with m_0, from E1.a, and (d) and (e) from E1.c are unpaired individuals. Therefore, it is possible to write the minimized SOP as a+b+c+d+r or

$$f(A,B,C,D) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\overline{C}D + AC\overline{D} + BD + BC$$

Note: Check this result for Example 1 by Karnaugh map approach.

Two-square implicants:

AB \ CD	00	01	11	10
00	1 0	4	12	8
01	1	1 5	1 13	1 9
11	3	1 7	1 15	11
10	2	1 6	1 14	1 10

Table E1.b represents all possible two-square implicants and the literals that they eliminate, i.e. 9 (1001_b) combined with 13 (1101_b) produces 1x01. As a result, literal "B" is eliminated. Corresponding product is $A\bar{C}D$. Since the only way of making an implicant that contains m_9 is to combine it with m_{13}, the implicant 9-13 is a prime one. The same rule applies to m_{10}.

Four-square implicants:

AB \ CD	00	01	11	10
00	1 0	4	12	8
01	1	1 5	1 13	1 9
11	3	1 7	1 15	11
10	2	1 6	1 14	1 10

Table E1.c represents all possible four-square implicants and the literals that they eliminate, i.e. 5 (0101_b) combined with 7 (0111_b) and 13 (1101_b) and 15 (1111_b) produces x1x1. As a result, literals "A" and "C" are eliminated. Corresponding product is BD.

III. QUINE-McCLUSKEY MINIMIZATION PROCEDURE (Decimal Notation)

Step 1: List the minterms grouped according to the number of 1‟s in their binary representation in the decimal format.

Step 2: Compare each minterm with larger minterms in the next group down. If they differ by a power of 2 then they pair-off. Check both minterms and form a second table by the minterms paired and substitute the decimal difference of the corresponding minterms in the bracket, i.e. m_x, m_y (y-x).

Step 3: Compare each element of the group in the new table with elements of the next lower group and select numbers that have the same numbers in parenthesis. If the lowest minterm number of the table formed in the lower group is greater than the corresponding number by a power of 2 then they combine; place a ✓ on the right of both elements.

Step 4: Form a second table by all four minterms followed by both powers of 2 in parentheses, i.e. the previous value (the difference) and the power of 2 that is greater.

Step 5: Select the common literals from each prime implicant by comparison.

Step 6: Write the minimal SOP from the prime implicant that are not checked✓.

Note: Read the above procedure in conjunction with the worked example given below.

Example :

Minimize the function f given below by Quine-McCluskey method using decimal notation.

$$f(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}D + ABC\bar{D} + ABCD$$

Solution

Step 1: Organize minterm as follows:

$$f(A,B,C,D) = \sum m(0,5,6,7,9,10,13,14,15)$$

Arrange minterms to correspond to their number of 1's as shown in E1.1a

*

1"s	Minterms	
0	0	…(a)
2	5 ✓	
	6 ✓	
	9 ✓	
	10 ✓	
3	7 ✓ 13 ✓	
	14 ✓	

Table E1.1a

	minterm paired		
	ΦΦ 5 , 7	(2)† ✓	
	5, 13 6, 7	(8) ✓ (1)	
	6, 14	✓	
*	9, 13	(8) ✓	…(b)
*	10, 14	(4)	…(c)
		(4)	
	7, 15	(8) ✓	4 15 ✓
	13, 15	(2) ✓	15 (1) ✓

14,

Table E1.1b

	minterms paired		
*	5,7-13,15¡	(2,8)	…(d)
*	6,14-7,15	(1,8)	…(e)

Table E1.1c

Φ - Squares combined (2 squares);

† - Number in bracket shows the literal being eliminated, i.e. (2) represents C [A=8, B=4, C=2, D=1];

¡- squares combined (4 squares) and numbers in the brackets are the literals eliminated.

Step 2: Compare each element of a group with the element of the next group if the difference is a power of 2 then they pair off, i.e. the first element in group 2 is paired say with the first element in group 3, which is 7-5=2, which is power of 2. Therefore, pair (5,7) makes the first element of the next table and minterms 5 and 7 get checked ✓. The result is shown in Table E1.1b.

Step 3: Now for the 2^3-table again compare each element of the group with elements of the lower group that have the same number in parentheses. If the lowest minterm in the lower group was greater by a power of 2 then they combine, i.e. 5,7 and 13,15 are combined because they have (2) in parentheses and 13 is greater then 5 by 8. Then they are paired off and entered in the next table E1.1c with the original (2) and their difference (8) in the parentheses.

Step 4: What we are left with is (a) from Table E1.1a, (b) and (c) from Table E1.1b, and (d) and (e) from Table E1.1c. From Table E1.1c, "d" is 5,7-13,15 (2,8). That means that positions 2^1 and 2^3 are X"s. Thus, "d" represents function BD. From the same table, "e" is 6,14-7,15 (1,8), which means positions 2^0 and 2^3 are X"s. Thus, "e" represents function BC. This can also be obtained by writing the elements of minterms and selecting two remaining literals:

6	0	1	1	0
14	1	1	1	0
7	0	1	1	1
15	1	1	1	1
	x	1	1	x
		B	C	

Therefore, the minimized SOP is

f □ □ □ □ □ □a b c d e ABCD ACD ACD BD BC□ □
□ □

Note: Compare this with the method of K-map or standard Quine-McCluskey (the first approach).
The above function consists of prime implicants. However, not all of them are necessary essential prime implicants.

Example 1.1.1. Determination of Essential Prime Implicants
For the SOP obtained in Example 1.1, determine the essential prime implicants and see if further reduction is possible.

Solution:
Construct a prime implicants table as shown in Table 1.1.1a, with prime implicants on left and minterms on top:

	Minterms / Prime implicants	0	5	6	7	9	10	13	14	15
*(2)	5, 7 – 13, 15		✓		✓			✓		✓
*(3)	6, 14 – 7, 15			✓	✓				✓	✓
*(4)	9, 13					✓		✓		
*(5)	10, 14						✓		✓	
*(1)	0	✓								
		✓(1)	✓(2)	✓(3)		✓(4)	✓(5)			
	5		✓		✓			✓		✓
	6			✓					✓	
	9					✓				

Table 1.1.1a

In each row, (except the bottom) checks ✓ are placed in the columns corresponding to minterms contained in the prime implicant listing in they row, i.e. the first prime implicant testing contains 5, 7, 13, 15. So, ✓ is placed in the first row in columns 5, 7, 13, 15. Repeat for each prime implicant.

Now inspect the table for columns that contain only one✓. That means that that prime implicant is the only term that contains that minterm, i.e. for example m_0 must be included in the SOP. This is marked with asterisks (*) in the left column and place ✓ in the bottom row. The same applies to 6, 9, and 10. Therefore, all prime implicants in this example are essential prime implicants. Other empty cells in the bottom row are covered by essential prime implicants. For example, once 5 are selected, and then 7, 13, and 15 also can be ✓ from the bottom row, and so on.

COMPUTER SIMULATION CODE Code for Quine-McCluskey Method in C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <limits.h>
#include <assert.h>
#include <stdbool.h>
#include <signal.h>
#include <inttypes.h>
#include <float.h>
#include <ctype.h>
#include <time.h>
int *in,**d1,**d2,x,y,**g,**d;
void create(int x,int y);
int staging(int x,int y);
int duplication(int x,int y);
int indexing(int x,int y3,int y);
void pimp(int x,int y3,int a);
void decode(int x,int y3);
void wxyz(int x,int y3);
int main()
{
int i,j,y1,y2,y3,a,q;
printf("\n Please give the number of variables you want to minimize - ");
scanf("%d",&x);
printf("\n\n In this program your inputs are designated as : ");
for(j=x-1;j>=0;j--)
printf("a[%d]",j);
printf("\n\n Please give the number of minterms that you want to minimize - ");
scanf("%d",&y);
in=(int *)malloc(y * sizeof(int));
d=d1=(int **)malloc(y * sizeof(int *));
for(i=0;i<y;i++)
d[i]=d1[i]=(int *)malloc((x+1)*sizeof(int));
for(i=0;i<y;i++)
{
printf("\n Please give decimal indices of minterms one at a time :: ");
```

```
scanf("%d",&in[i]);}
create(x,y);
y1=y*(y+1)/2;
d2=(int **)malloc(y1*sizeof(int *));
for(i=0;i<y1;i++)
d2[i]=(int *)malloc((x+1)*sizeof(int));
y2=staging(x,y);
y3=duplication(x,y2);
a=indexing(x,y3,y);
pimp(x,y3,a);
printf("\n\nThe essential prime implicants giving minimized expression
are:\n\n");
decode(x,y3);
}
void create(int x,int y)
{
int i,j,a;
for(i=0;i<y;i++)
{
a=in[i];
for(j=0;j<x;j++)
{
d[i][j]=d1[i][j]=a%2;
a=a/2;
}
d[i][x]=d1[i][x]=8;
}
}
int staging(int x,int y)
{
int i1,j1,k1,t1,i2,j2,t2,c;
i2=0;c=0;
for(i1=0;i1<(y-1);i1++)
{
for(j1=i1+1;j1<y;j1++)
{
t1=0;
for(k1=0;k1<x;k1++)
{
if(d1[i1][k1]!=d1[j1][k1])
```

```
{
t1++;
t2=k1;
}
}
if(t1==1)
{
for(j2=0;j2<t2;j2++)
d2[i2][j2]=d1[i1][j2];
d2[i2][t2]=3;
for(j2=t2+1;j2<y;j2++)
d2[i2][j2]=d1[i1][j2];
d2[i2][x]=8;
d1[i1][x]=9;
d1[j1][x]=9;
i2++;
}
}
}
for(i1=0;i1<y;i1++)
{
if(d1[i1][x]==8)
{
for(j1=0;j1<=x;j1++)
d2[i2][j1]=d1[i1][j1];
i2++;
}
}
for(j1=0;j1<x;j1++)
{
if(d1[0][j1]==d2[0][j1])
c++;
}
if(c<x)
{
d1=(int **)malloc(i2*sizeof(int *));
for(i1=0;i1<i2;i1++)
d1[i1]=(int *)malloc((x+1)*sizeof(int));
for(i1=0;i1<i2;i1++)
{
```

```
for(j1=0;j1<=x;j1++)
d1[i1][j1]=d2[i1][j1];
}
staging(x,i2);
}
else
return(i2);
}
int duplication(int x,int y)
{
int i1,i2,j,c,t;
t=0;
for(i1=0;i1<(y-1);i1++)
{
for(i2=i1+1;i2<y;i2++)
{
c=0;
for(j=0;j<x;j++)
{
if(d1[i1][j]==d1[i2][j])
c++;
}
if(c==x)
d1[i2][x]=9;
}
}
for(i1=0;i1<y;i1++)
{
if(d1[i1][x]==9)
t++;
}
i2=y-t;
d2=(int **)malloc(i2*sizeof(int *));
for(j=0;j<i2;j++)
d2[j]=(int *)malloc((x+1)*sizeof(int));
i2=0;
for(i1=0;i1<y;i1++)
if(d1[i1][x]==8)
{
for(j=0;j<=x;j++)
```

```
d2[i2][j]=d1[i1][j];
i2++;
}
return(i2);
}
int indexing(int x,int y3,int y)
{
int i1,j,c1,i2,c,a;
c=0;a=1;
for(j=0;j<x;j++)
if(d1[0][j]==3)c++;
for(i1=0;i1<c;i1++)
a=a*2;
g=(int **)malloc(y3*sizeof(int *));
for(j=0;j<y3;j++)
g[j]=(int *)malloc(a*sizeof(int));
for(i1=0;i1<y3;i1++)
for(j=0;j<a;j++)
g[i1][j]=-2;
for(i1=0;i1<y3;i1++)
{
c=0;
for(i2=0;i2<y;i2++)
{
c1=0;
for(j=0;j<x;j++)
{
if((d2[i1][j]==d[i2][j])||(d2[i1][j]==3))
c1++;
if(c1==x)
{
g[i1][c]=in[i2];
c++;
}
}
}
}
return(a);
}
void pimp(int x,int y3,int a)
```

```
{
int i1,i2,j1,j2,c,w,j3,c1,c2,j4,c3,c4;
c=0;
for(i1=0;i1<y3;i1++)
{
for(j1=0;j1<a;j1++)
{
if(g[i1][j1]!=-2)
{
for(i2=0;i2<y3;i2++)
{
if(i2!=i1)
{
for(j2=0;j2<a;j2++)
if(g[i1][j1]!=g[i2][j2])
c++;
}
}
if(c==a*(y3-1))
d2[i1][x]=91;c=0;
}
}
}
for(i1=0;i1<y3;i1++)
{
if(d2[i1][x]==91)
{
for(j1=0;j1<a;j1++)
{
if(g[i1][j1]!=-2)
{
for(i2=0;i2<y3;i2++)
{
if(i1!=i2)
{
for(j2=0;j2<a;j2++)
if(g[i1][j1]==g[i2][j2])
g[i2][j2]=-3;
}
}
```

```
}
}
}
}
for(i1=0;i1<y3;i1++)
{
if(d2[i1][x]==91)
{
for(j1=0;j1<a;j1++)
if(g[i1][j1]!=-2)g[i1][j1]=-1;
}
}
for(i1=0;i1<y3;i1++)
{
if(d2[i1][x]!=91)
{
for(j1=0;j1<a;j1++)
{
if(g[i1][j1]>=0)
{
for(i2=0;i2<y3;i2++)
if(i2!=i1)
{
for(j2=0;j2<a;j2++)
{
if(g[i2][j2]>=0)
{
if(g[i1][j1]==g[i2][j2])
{
w=i2;
if((d2[w][x]==90)||(d2[w][x]==8))
{
for(j3=0,c1=0;j3<x;j3++)
if(d2[i1][j3]==3)
c1++;
for(j3=0,c2=0;j3<x;j3++)
if(d2[i2][j3]==3)
c2++;
if(c1>c2)
{
```

```
d2[i1][x]=90;
g[i2][j2]=-1;
}
if(c2>c1)
{
d2[i1][x]=8;
d2[i2][x]=90;
g[i1][j1]=-1;
}
if(c2==c1)
{
for(j3=0,c3=0,c4=0;j3<a;j3++)
{
if(g[i1][j3]==-
1)c3++;
if(g[i2][j3]==-1)c4++;
}
if(c3>c4)
{
d2[i2][x]=90;d1[i1][x]=8;
g[i1][j1]=-1;
}
if(c3==c4)
{
d2[i1][x]=90;
g[i2][j2]=-1;
}
if(c3<c4)
{
d2[i1][x]=90;
g[i2][j2]=-1;
}
}
}
if(d2[w][x]==91)
d1[w][x]=8;
}
}
}
}
```

```
}
}
}
}
return;
}
void decode(int x,int y3)
{
int i,j;
for(i=0;i<y3;i++)
{
if((d2[i][x]==91)||(d2[i][x]==90))
{
for(j=x-1;j>=0;j--)
{
if(d2[i][j]==0)printf("a[%d]'",j);
if(d2[i][j]==1)printf("a[%d]",j);
}
}
printf("\n\n");
}
return ;
}
```

RESULT

```
 Please give the number of variables you want to minimize - 4

 In this program your inputs are designated as : a[3]a[2]a[1]a[0]

 Please give the number of minterms that you want to minimize - 9

 Please give decimal indices of minterms one at a time :: 0

 Please give decimal indices of minterms one at a time :: 5

 Please give decimal indices of minterms one at a time :: 6

 Please give decimal indices of minterms one at a time :: 7

 Please give decimal indices of minterms one at a time :: 9

 Please give decimal indices of minterms one at a time :: 10

 Please give decimal indices of minterms one at a time :: 13

 Please give decimal indices of minterms one at a time :: 14

 Please give decimal indices of minterms one at a time :: 15

The essential prime implicants giving minimized expression are:

a[2]a[0]

a[2]a[1]

a[3]a[1]'a[0]

a[3]a[1]a[0]'

a[3]'a[2]'a[1]'a[0]'


--------------------------------
Process exited after 66.41 seconds with return value 5
Press any key to continue . . .
```

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 $3^n \ln(n)$. If n = 32 there may be over 6.5 * 1015 prime implicants. Functions with a large number of variables have to be minimized with potentially non-optimal heuristic methods, of which the Espresso heuristic logic minimizer is the de facto standard.

CONCLUSION

In this paper we have listed the codes for the implementation of Quine-McCluskey method using the computer languages C and C++. Readers well versed in any of these languages would be at ease to follow with the computer code. In preparing these codes, one primary observation we had was that the number of lines of code in C++ was about 100 lines less than what could be achieved in C. This provides us an insight about the inherent advantage we get in the object oriented design paradigm as compared to the procedural languages. Understanding the concept and theory behind the QuineMcCluskey method is the key to writing good and optimized codes in any of the computer languages.

REFERENCES

[1] Turton, Brian CH. "Extending Quine-McCluskey for exclusive-or logic synthesis." Education, IEEE Transactions on 39.1 (1996): 81-85.

[2] Chang, Hyun Sung, Sanghoon Sull, and Sang Uk Lee. "Efficient video indexing scheme for content-based retrieval." Circuits and Systems for Video Technology, IEEE Transactions on 9.8 (1999): 1269-1279.

[3] Jain, Tarun Kumar, Dharmender Singh Kushwaha, and Arun Kumar Misra. "Optimization of the quine-mccluskey method for the minimization of the boolean expressions." Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on. IEEE, 2008.

[4] Tseng, Chih-Cheng, and Kwang-Cheng Chen. "Organizing an optimal cluster-based ad hoc network architecture by the modified quine-McCluskey algorithm."Communications Letters, IEEE 11.1 (2007): 43-45.

[5] Šeda, Miloş. "Heuristic Set-Covering-Based Postprocessing for Improving the QuineMcCluskey Method." International Journal of Computational Intelligence4.2 (2008).

[6] Givone, Donald D., and Rong Luo. Digital principles and Design. McGraw-Hill, 2003.

[7] Coello, Carlos A. Coello, Alan D. Christiansen, and Arturo Hernández Aguirre. "Use of evolutionary techniques to automate the design of combinational circuits." International Journal of Smart Engineering System Design 2 (2000): 299-314.

[8] Mirsalehi, Mir M., and Thomas K. Gaylord. "Logical minimization of multilevel coded functions." Applied optics 25.18 (1986): 3078-3088.

[9] Hwa, H. R. "A method for generating prime implicants of a Boolean expression."IEEE Transactions on Computers 23.6 (1974): 637-641.

[10] Mirsalehi, Mir M., and Thomas K. Gaylord. "Logical minimization of multilevel coded functions." Applied optics 25.18 (1986): 3078-3088.

[11] Safaei, Javad, and Hamid Beigy. "Quine-McCluskey Classification." Computer Systems and Applications, 2007. AICCSA'07. IEEE/ACS International Conference on. IEEE, 2007.

[12] Prasad, P., Azam Beg, and Ashutosh Kumar Singh. "Effect of Quine-McCluskey simplification on Boolean space complexity." Innovative Technologies in Intelligent Systems and Industrial Applications, 2009. CITISIA 2009. IEEE, 2009.

[13] Coello, Carlos A., Alan D. Christiansen, and Arturo Hernández Aguirre. "Automated design of combinational logic circuits using genetic algorithms."Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms. 1997.

[14] Kamath, A. P., et al. "An interior point approach to Boolean vector function synthesis." Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on. IEEE, 1993.

[15] Tosser, A. J., and D. Dubus. "Graphical use of „don't care"symbols in the shortened Quine-McCluskey's tables." International Journal of Electronics Theoretical and Experimental 43.4 (1977): 353-359.

[16] Martins, Mayler GA, et al. "Efficient method to compute minimum decision chains of Boolean functions." Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI. ACM, 2011.

[17] Slowik, Adam, and Michal Bialko. "Evolutionary design of combinational digital circuits: State of the art, main problems, and future trends." Information Technology, 2008. IT 2008. 1st International Conference on. IEEE, 2008.

[18] Safaei, Javad, and Hamid Beigy. "Boolean function minimization: The information theoretic approach." Proc. 15th IEEE/ACM workshop on Logic and Synthesis. 2006.

[19] Coudert, Olivier, Jean Christophe Madre, and Henri Fraisse. "A new viewpoint on two-level logic minimization." Design Automation, 1993. 30th Conference on. IEEE, 1993.

[20] Jadhav, Vitthal, and Amar Buchade. "Modified Quine-McCluskey Method." arXiv preprint arXiv:1203.2289 (2012).

[21] Coello Coello, Carlos A., Arturo Hernández Aguirre, and Bill P. Buckles. "Evolutionary multiobjective design of combinational logic circuits." Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on. IEEE, 2000.

[22] BASÇÍFTÇÍ, Sirzat KAHRAMANLI Fatih. "Boolean functions simplification algorithm of O (N) complexity." Mathematical & Computational Applications 8.3 (2003): 271-278.

[23] Mahmoud, H. A. H. "A new method for two-level logic minimization." Circuits and Systems, 2003 IEEE 46th Midwest Symposium on. Vol. 3. IEEE, 2003.

[24] Brayton, Robert K., and John A. Darringer. "Logic synthesis overview." The Best of ICCAD. Springer US, 2003. 181-189.

[25] Coudert, Olivier. "Doing two-level logic minimization 100 times faster."Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms.
Society for Industrial and Applied Mathematics, 1995.