DOI: 10.21917/ijme.2016.0024

STATIC STRUCTURE SIMPLIFICATION OF BOOLEAN FUNCTION FOR ‘N’ VARIABLES - A NOVEL APPROACH

Visvam Devadoss Ambeth Kumar[1] and S. Gokul Amuthan[2]
Department of Computer Science and Engineering, Panimalar Engineering College, India
E-mail: [1]ambeth_20in@yahoo.co.in, [2]iamnotgokul@gmail.com

Abstract

Digital Systems have a prominent role in everyday life that we refer to the present technological period as the digital age. Digital Systems implement digital circuits which are mostly combination of digital gates. These gates can be represented in an understandable form called Boolean expressions. These expressions formed vary for each circuit. Hence for building an efficient circuit the Boolean expression should be minimized. To achieve this, we use, Karnaugh map (K-map) and Quine-McCluskey (QM) methods which are well known methods to simplify Boolean expression. The K-map until now is introduced to solve up to 6 variable and QM which can solve ‘n’ variables, yet it involve a serious of steps and does not provide a visual way to minimising the expression. The following paper implements the idea of K-map to reduce the expression except that it could solve ‘n’ variables as QM method. The method Divide and Conquer Karnaugh map (DC-K-map) implements 4 variable K-map in a redundant way to reduce the expression for variables greater than 7.

Keywords:

Karnaugh Map, Boolean Functions, Quine-McCluskey Method, Grouping and Simplification.

1. INTRODUCTION

In 1854, George Boole introduced a systematic treatment of logic and developed for this purpose an algebraic system now called as Boolean Algebra [11]. In 1938 C.E. Shannon introduced a two valued Boolean algebra called switching algebra. Boolean algebra is a system of mathematical logic. [1][2] It differs from both ordinary algebra and binary number system. The symbol which represents an arbitrary elements of a Boolean algebra is known as variable. Any single variable can either hold a 0 or a 1 at corresponding time. These variables are the base of Boolean algebra. To implement them we use digital gates. Hence it is must to reduce a Boolean expression to the least. The more we reduce an expression the less the size, cost and speeder the process will be done.

To reduce we have many methods proposed by many greats. At first Huntington in 1904 used a set of laws called Boolean law to reduce the expression. But a problem occurred for variables greater than 4 and it’s not suitable for computer implementation. After many desperate approaches an American physicist named, Maurice Karnaugh in 1953, used the refined approach of Edward Veitch’s Veitchdiagram (1952). [4] He presented a paper “The map method for synthesis of combinational logic circuits” which stated about K-map, which is pictorial form truth table and also a pictorial way to reduce the expression. [8] It’s an array of cells or square which holds a binary value either 1 or 0 corresponding to the input variable. But sadly he proposed this method to solve for up to 6 variables. Later in 1956, W.V. Quine [10] found an innovative approach of minimizing the Boolean functions. It’s similar to K-map but it’s in tabular format. Hence it’s sometimes referred to as tabulation method. This method was best suited for computer algorithms yet it has limited range of use as it solves the problem in NP-Hard way. [5] This paper uses K-map method, with slight modification to solve greater than 7 variables.

2. SYMBOLIZING THE BOOLEAN FUNCTION

To represent the Boolean functions we use four golden representations.

2.1 TRUTH TABLE

The common way of representing the Boolean function. This represents the entire possible combinations of input variable in a tabular format.

Table.1. A truth table for 2 variable

A	B	Minterm
0	0	A'B'
0	1	A'B
1	0	AB'
1	1	AB

2.2 SUM OF PRODUCTS (SOP)

This is the more common form of Boolean expressions. The expressions are implemented as AND gates (products) feeding a single OR gate (sum). These are otherwise called as *minterms.*

Example: A SOP of two variable

$$A + A'B' + AB$$

2.3 PRODUCT OF SUM (POS)

This is less commonly used form of Boolean expressions. The expressions are implemented as OR gates (sums) feeding into a single AND gate (product). These are otherwise called as *maxterms.*

Example: A POS of two variable

$$A(A' + B')(A + B)$$

2.4 CANONICAL FORM (CF)

The Boolean expression are said to be in canonical or standard form if each term contain all available input variable.

Example: A CF of two variable

$$AB + A'B' + AB' \qquad \text{(SOP)}$$

$$(A + B)(A' + B')(A + B') \qquad \text{(POS)}$$

3. MINIMIZING THE BOOLEAN FUNCTION

To minimise the Boolean function we have the following methods.

3.1 BOOLEAN LAWS

This fundamental and most basic way to solve a Boolean expression. This uses set of laws to perform such simplification of expression [12]. This was found by Huntington in 1904.

Laws used: Commutative law, Associative law, Distributive law, Duality, Identity law, Idempotent law, Involution law, Complementary law, Absorption law, DeMorgan law and Consensus theorem. [9]

Example:

Given an expression ABC + ABC' + A

Then it can be reduced as,

ABC + ABC' + A = AB(C + C') + A

= AB + A (Complementary law)

= A(B + 1)

= A (Identity law)

The disadvantage of this method is that this method requires knowledge of all the laws. Also it can never be implemented using computer. It has a major drawback that it's more difficult to solve when the variables is greater than 4.

3.2 K-MAP

Karnaugh maps to simplification of Boolean logic functions have an abundant applications in designing of digital systems and logic circuits. Among all existent methods for minimizing algebraic functions like tabulation method, using the Karnaugh maps has been mentioned as the dominant tool in digital design. It was developed by Maurice Karnaugh in 1953. [3] K-map is an array of cells or square which represent a binary value either 0 or 1 correspondingly.

Map Set-up

The number of cell in K-map is equal to 2^n, where n is the number of input variables. [8] The map is drawn to show the relation between squares and input variables. Variables are assigned to row and column. Binary marking are placed in each row and column using reflected code sequence. [6] Each cell in the map represent a combination of input variable in a given truth table.

B \ A	0	1
0	0	2
1	1	3

Fig.1. Two variable K-map

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

Fig.2. Three variable K-map

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

Fig.3. Four variable K-map

	A'				A'			
	D'E'	D'E	DE	DE'	D'E'	D'E	DE	DE'
B'C'	0	1	3	2	16	17	19	18
B'C	4	5	7	6	20	21	23	22
BC	12	13	15	14	28	29	31	30
BC'	8	9	11	10	24	25	27	26

Fig.4. Five Variable K-map

		B'				B			
		E'F'	E'F	EF	EF'	E'F'	E'F	EF	EF'
	C'D'	0	1	3	2	16	17	19	18
A'	C'D	4	5	7	6	20	21	23	22
	CD	12	13	15	14	28	29	31	30
	CD'	8	9	11	10	24	25	27	26
	C'D'	32	33	35	34	48	49	51	50
A	C'D	36	37	39	38	52	53	55	54
	CD	44	45	47	46	60	61	63	62
	CD'	40	41	43	42	56	57	59	58

Fig.5. Six variable K-map

The disadvantage of this method can go up to 6 variable only. Hence K-maps are hard to imply for variable greater than 7. And also the rule for mapping 5 and 6 variables require keen concentration to map simultaneously blocks concurrently.

4. DIVIDE AND CONQUER K-MAP (DC-K-MAP)

The disadvantage of formal K-map is that it can solve up to 6 variable only. This disadvantage can be recovered by using DC-K-Map. This is similar to formal K-map but it can be used to solve up to n-variable. It requires only the knowledge of mapping a 4 variable K-map. As it uses mapping technique up to 4 variable it’s easy to map the map the cells efficiently.

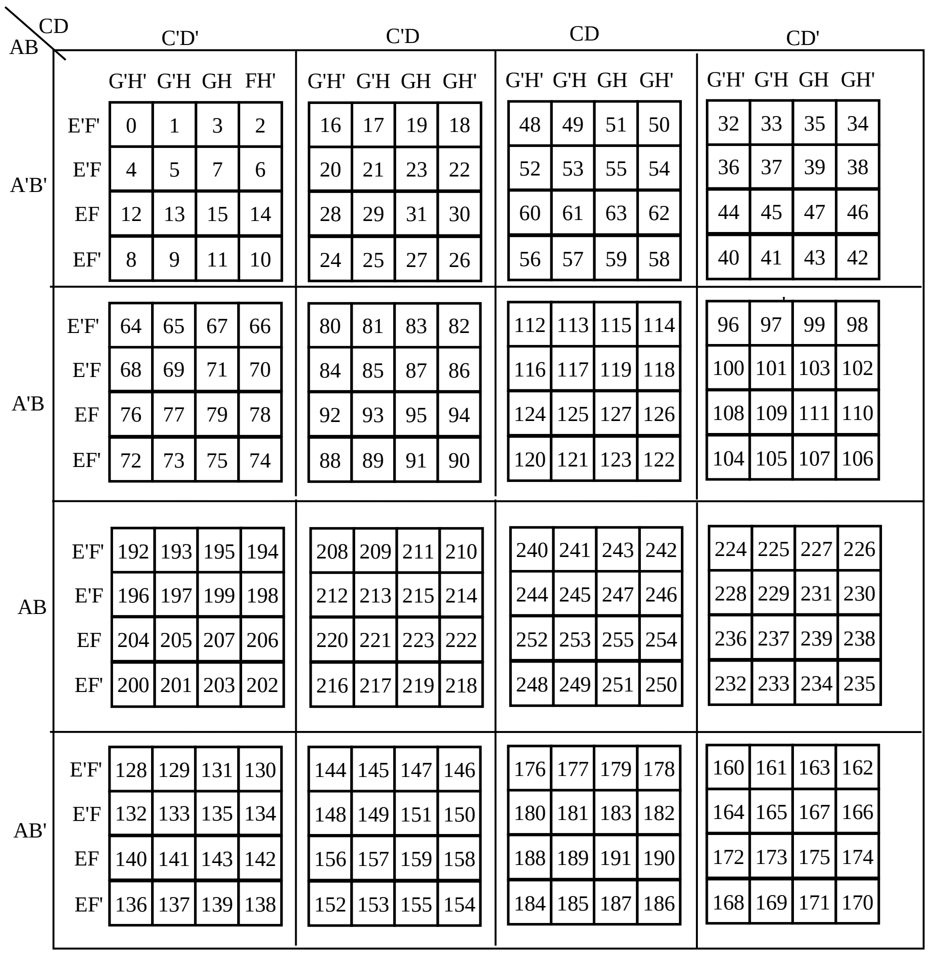
4.1 GENERALIZATION

Map Set-up

A \ BC		B'C'				B'C				BC				BC'			
		F'G'	F'G	FG	FG'	F'G'	F'G	FG	FG'	F'G'	F'G	FG	FG'	F'G'	F'G	FG	FG'
	D'E'	0	1	3	2	16	17	19	18	48	49	51	50	32	33	35	34
A'	D'E	4	5	7	6	20	21	23	22	52	53	55	54	36	37	39	38
	DE	12	13	15	14	28	29	31	30	60	61	63	62	44	45	47	46
	DE'	8	9	11	10	24	25	27	26	56	57	59	58	40	41	43	42
	D'E'	64	65	67	66	80	81	83	82	112	113	115	114	96	97	99	98
A	D'E	68	69	71	70	84	85	87	86	116	117	119	118	100	101	103	102
	DE	76	77	79	78	92	93	95	94	124	125	127	126	108	109	111	110
	DE'	72	73	75	74	88	89	91	90	120	121	123	122	104	105	107	106

Fig.6. Comprehensive Structure for DC-K-Map method (7 Variables)

The map setup is similar procedure used in K-map which follows the GrayScale method. The idea can be extended to 8 variable by following same procedure. A sample of 8 variable map is shown below.



AB \ CD		C'D'				C'D				CD				CD'			
		G'H'	G'H	GH	FH'	G'H'	G'H	GH	GH'	G'H'	G'H	GH	GH'	G'H'	G'H	GH	GH'
A'B'	E'F'	0	1	3	2	16	17	19	18	48	49	51	50	32	33	35	34
	E'F	4	5	7	6	20	21	23	22	52	53	55	54	36	37	39	38
	EF	12	13	15	14	28	29	31	30	60	61	63	62	44	45	47	46
	EF'	8	9	11	10	24	25	27	26	56	57	59	58	40	41	43	42
A'B	E'F'	64	65	67	66	80	81	83	82	112	113	115	114	96	97	99	98
	E'F	68	69	71	70	84	85	87	86	116	117	119	118	100	101	103	102
	EF	76	77	79	78	92	93	95	94	124	125	127	126	108	109	111	110
	EF'	72	73	75	74	88	89	91	90	120	121	123	122	104	105	107	106
AB	E'F'	192	193	195	194	208	209	211	210	240	241	243	242	224	225	227	226
	E'F	196	197	199	198	212	213	215	214	244	245	247	246	228	229	231	230
	EF	204	205	207	206	220	221	223	222	252	253	255	254	236	237	239	238
	EF'	200	201	203	202	216	217	219	218	248	249	251	250	232	233	234	235
AB'	E'F'	128	129	131	130	144	145	147	146	176	177	179	178	160	161	163	162
	E'F	132	133	135	134	148	149	151	150	180	181	183	182	164	165	167	166
	EF	140	141	143	142	156	157	159	158	188	189	191	190	172	173	175	174
	EF'	136	137	139	138	152	153	155	154	184	185	187	186	168	169	171	170

Fig.7. Comprehensive Structure for DC-K-Map method (8 Variables)

5.2 ADJACENCY RULE

Each cell in K-map are positioned adjacent and arranged in such a way that only a single variable changes between adjacent cells. [7] Cell that differ by a single value is said to be adjacent and can group those only.

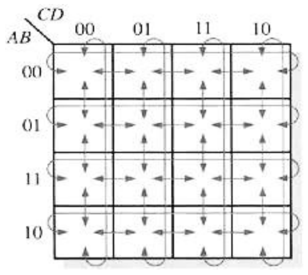


Fig.8. Adjacency Rule Map

5.3 MAPPING RULE

Place 1 on position on respective position on K-map only when the minterm number is given.

1. Always group only the adjacent cells and not others that to in the powers of 2, that is (1, 2, 4, 8, 16, 32, 64, 128 ....etc.).
2. Always group the largest possible pair first to reduce the number of literals.
3. Treat each map as separate 4 variable K-map.
4. Ensure the no 1's are overlapped or belonging to another group.
5. Avoid redundant grouping.
6. After solving each 4 variable separately, now construct a DC-K-map as shown below.

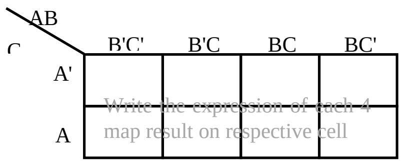

AB
C
B'C'
B'C
BC
BC'
A'
A
Write the expression of each 4 map result on respective cell

Fig.9. Structure of DC-K-Map

7. Write the expression result of each 4 variable on respective cell.
8. Now see to that there exist two literal with same minterm and are adjacent to each other
9. If so the perform the same reduction by eliminating the non-grouping terms.
10. Then write the result by combining the boundary variables.

Note: The rule for don't care is similar as that of formal K-map. Map the 'don't care terms' only if it's necessary.

Example

Σ_m(A, B, C, D, E, F, G) = (1, 3, 5, 7, 8, 9, 10, 18, 20, 21, 23, 24, 29, 31, 33, 35, 37, 39, 40, 43, 48, 51, 53, 55, 57, 59, 61, 63, 64, 65, 66, 69, 73, 77, 79, 81, 83, 84, 85, 89, 91, 93, 97, 99, 101, 102, 103, 105, 106, 108, 113, 115, 116, 118, 121, 123, 124, 127)

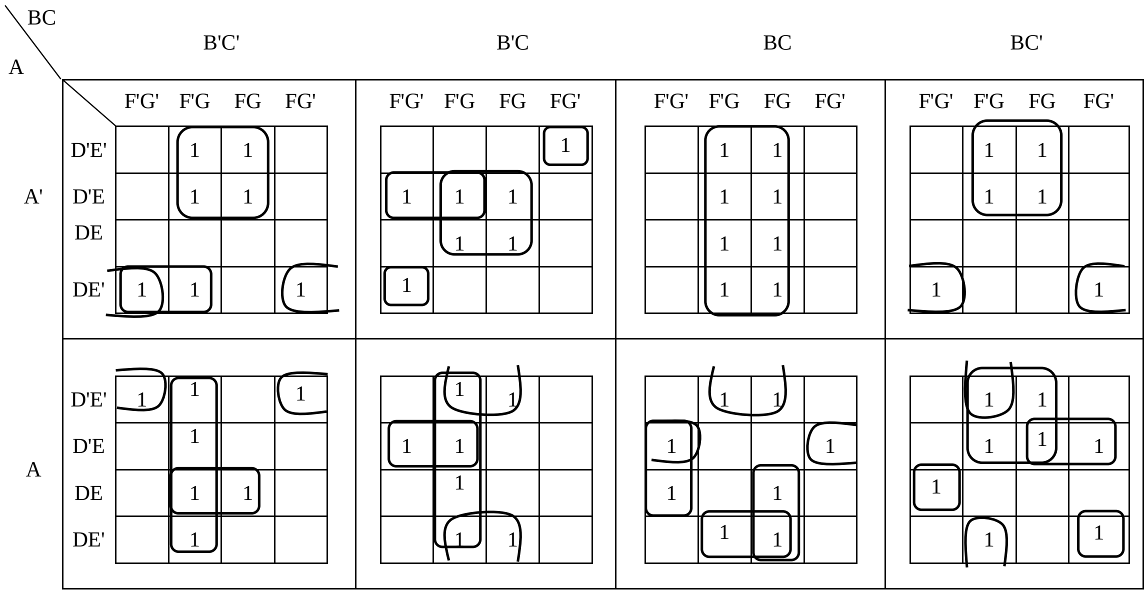
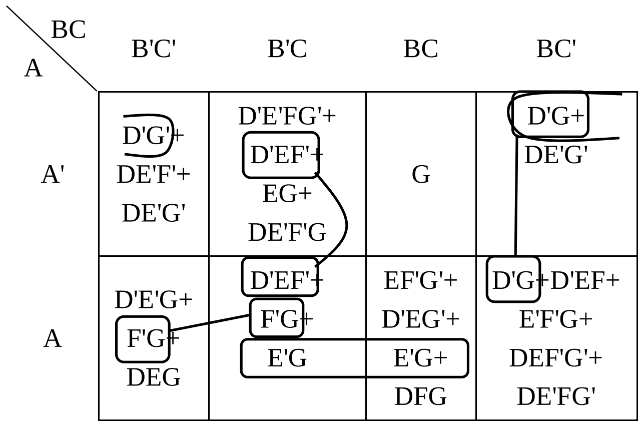


Fig.10. Collection of 4 variable K-map to form 7 variable



A \ BC	B'C'	B'C	BC	BC'
A'	D'G'+ DE'F'+ DE'G'	D'E'FG'+ D'EF'+ EG+ DE'F'G	G	D'G+ DE'G'
A	D'E'G+ F'G+ DEG	D'EF'+ F'G+ E'G	EF'G'+ D'EG'+ E'G+ DFG	D'G+D'EF+ E'F'G+ DEF'G'+ DE'FG'

Fig.11. DC-K-map

A'C'D'G + A'B'C'DE'F' + A'B'C'DE'G' + A'B'CD'E'FG' + B'CD'EF' +A'B'CEG + A'B'CDE'F'G' + A'BCG + A'BC'DE'G' + AB'C'D'E'G' + AB'F'G + AB'C'DEG + ACE'F + ABCEF'G' + ABCD'EG' + ABCDFG + BC'D'G + ABC'D'EF + ABC'E'F'G + ABC'DEF'G' + ABC'DE'FG'

Fig.12. Solution of 7 variable

5. COMPARISON OF DC-K-MAP AND QM-METHOD

To compare the both methods in an efficient way we consider an example which is proceeded in both ways, that is through DC-K-Map and also through QM method.

Example: Considering a Boolean function of seven variable which has minterms defined as below.

Σ_m (A, B, C, D, E, F, G) = (1, 3, 5, 7, 20, 21, 33, 61, 63, 73, 75, 77, 79, 81, 83, 85, 96, 97, 99, 124,125,127)

Solving through QM method

Now to solve the function through QM method, the main idea is to generate prime implicants. To do so the following steps are proceeded.

Step 1:

Generating the binary value for each of the minterms.

Minterms	Binary representation	No. of one's
1	0000001	1
3	0000011	2
5	0000101	2
7	0000111	3
20	0010100	2
21	0010101	3
33	0100001	2
61	0111101	5
63	0111111	6
73	1001001	3
75	1001011	4
77	1001101	4
79	1001111	5
81	1010001	3
83	1010011	4
85	1010101	4
96	1100000	2
97	1100001	3
99	1100011	4
124	1111100	5
125	1111101	6
127	1111111	7

Step 2:

Sorting the minterms according to number of one’s in the binary representation of minterms.

(Sorting is again an exhaustive process, in QM method and hence can increase the time complexity of the algorithm. Note that there is no sorting process in the DC-K-Map method, so the time complexity will drastically decrease when compared to QM method).

Step 3:

Compare each binary number with every term in the adjacent next higher category and if they differ by only one position put a check mark and copy the term in the next column with ‘-’ in the position that they differed.

(Again in this step, we sacrifice time complexity greatly for comparison operation. And it would result in exhaustive comparison if there are no term with specified difference in position)

Step 4:

Apply the same process described in Step 3 for the resultant column and continue these cycles until a step that yields no further elimination.

Minterms	Binary representation
1,3,5,7 1,5,3,7	0000__1 0000__1
73,75,77,79 73,77,75,79	1001__1 1001__1

Hence the solution generated by QM is

A'C'D'E'F'G + A'B'CD'EF' + ABC'D'E'F' + AB'CD'E'G + AB'CD'E'F + ABC'D'E'G + A'BCDEG + ABCDEF' + ABCDEFG + A'B'C'D'G +AB'C'DG

Solving through DC-K-Map method

Now solving the same seven variable function using DC-K-map method.

Step 1:

Constructing eight 4 variable map and plotting 1’s on respective cells. Then by using the proposed procedure map the cells individually.

Step 2:

Constructing DC-K-Map and write the respective expression of each 4 variable K-map on respective cell. Solve the DC-K-map according to proposed procedure.

A \ BC	B'C'	B'C	BC	BC'
A'	D'G'	D'EF'	DEG	D'E'F'G
A	DG	D'F'G+ D'E'G	DEG+ DEF'	D'E'F'+ D'E'G

Fig.14. DC-K-map

Step 3:

Writing the expression from the DC-K-Map.

A'B'C'D'G + A'B'CD'E'F' + BCDEG + A'BC'D'E'F'G + AB'C'DG + AB'CD'F'G + AB'CD'E'G + ABCDEF' + ABC'D'E'F' + ABC'D'E'G

Fig.15. Solution of 7 variable

This is the solution obtained from DC-K-Map

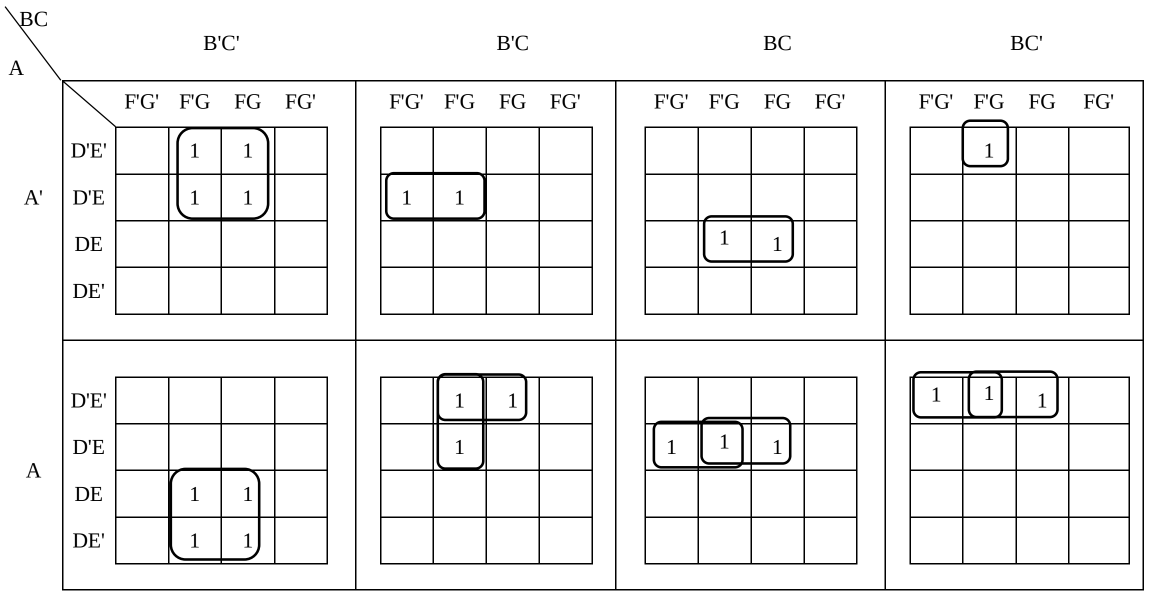


Fig.13. Collection of 4 variable K-map to form 7 variable

6. FACTORS TO BE NOTED

6.1 NUMBER OF MINTERMS

The number of minterms in expression plays a vital role in designing the circuit. Hence the lesser the number of minterms the cheaper and faster the circuit responses.

In our considered example the number of minterms in

QM method = 11

DC-K-Map = 10

Hence from which we can conclude that *the number of minterms in QM method and DC-K-Map will be utmost the same and in some cases DC-K-Map would provide fewer minterms than QM method.*

6.2 EXECUTION EFFICIENCY

Each and every method proposed has to simple such that it can reduce the complexity.

In *QM Method*, for any number of variables *it requires at least of 4 passes to minimize the function.*

Whereas in *DC-K-Map*, for all number of variables *the number of passes required is comparatively less than number of passes required in QM method as it’s linear.*

6.3 SPACE REQUIREMENT

Each method written must be written to an algorithm and to implement them it should have appropriate data structure. The size of data structure also plays a vital role as it’s more precious to conserve the space.

In *QM Method*, for n variable the size requirement would be $3^n/n$ (number of prime implicants generated), which is huge value and it grows exponentially.

But in *DC-K-Map*, for n variable the size requirement would be 2^n, which is comparatively lesser than QM Method.

6.4 TIME EFFICIENCY

Any algorithm should have a least amount of execution time, as it’s an important factor an algorithm must be designed in such a way that it should be less.

In *QM Method*, the total number of comparisons would be

$$\sum_{i=0}^{n-1} nC_i * nC_{i+1}$$

But in DC-K-Map the total number of comparisons would be as follows,

The total number of comparisons for 4 variable K-map is 64 (2^6). The DC-K-Map uses 4 variable K-Map repeatedly for higher value of n.

Then for 7 variable number of comparison is $2^6 * 8$, for 8 variable number of comparisons would be $2^6 * 16$ and we can predict so on.

Then in general we can write as $2^6 * 2^{n-4} = 2^{n+2}$ (where n is greater than equal to 7). Now for additional DC-K-Map it would require 2^{n-4} comparisons.

Hence in total we can conclude that it would require $2^{n+2}+2^{n-4}$ comparisons, which could be further reduced as $65*(2^{n-4})$ comparisons.

Number of variable	QM Method	DC-K-Map
7	3003	520
8	11440	1040
-----	------	---------
-----	------	---------
32	$6.5 * 10^{15}$	$1.7 * 10^{10}$

Hence we can achieve greater efficiency in DC-K-Map rather than going for QM Method.

6.5 ADVANTAGE

The most important advantage is that it has lesser space and time efficiency than QM method. Even though this method add extra table computation which is not necessary in formal K-Map, yet it provides a detail view of expression which can be combined together as one. Another advantage is that it reuses the formal 4 variant K-map and there is need for creation of algorithm for the DC-K-Map only.

6.6 DISADVANTAGE

The disadvantage of this method is that it require additional map to compute the final expression.

7. CONCLUSION

The new method DC-K-map proposed in this paper can implemented to achieve greater efficiency in digital circuit and also can provide a visual way of reducing expression like K-map except that it’s possible for greater than n-variable. The problem of redundancy (in some cases) and simultaneous mapping found in K-map for 5 and 6 variable can also be recovered through DC-K-Map method.

REFERENCES

[1] M. Morris Mano and Michael D. Ciletti, “*Digital Design*”, 4th Edition, Pearson, 2008.

[2] R. Mohan Ranga Rao , “An Innovative Procedure to Minimize Boolean Function”, *International Journal of Advanced Engineering Sciences and Technologies*, Vol. 3, No. 1, pp. 12-14, 2011.

[3] M. Karnaugh, “The Map for Synthesis of Combinational Logic Circuits”, *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, Vol. 72, No. 5, pp. 593-599, 1953.

[4] Arunachalam Solairaju and Rajupillai Periyasamy, “Optimal Boolean Function Simplification through K-Map using Object-Oriented Algorithm”, *International Journal of Computer Applications*, Vol. 15, No. 7, pp. 28-32, 2011.

[5] Karnaugh map, Available at: http://en.wikipedia.org/wiki/Karnaugh_map

[6] Vertical Horizon, Digital Electronics, http://verticalhorizons.in/wp=content/uploads/2012/12/6_v

ariable_k_map1.bmp" and "http://verticalhorizons.in/wp-content/uploads/2012/12/5_variable_k_map1.bmp".

[7] Barmak Honarvar Shakibaei Asli, "Survey of a New Note on Karnaugh Maps", *International Journal of Control and Automation*, Vol. 5, No. 2, pp. 21-24, 2012.

[8] Bob Harbort and Bob Brown, "Karnaugh Maps", Southern Polytechnic State University, Jan 2001.

[9] John F. Wakerly, "*Digital Design Principles and Practices*", 4th Edition, Pearson, 2008.

[10] W.V. Quine, "The Problem of Simplifying Truth Tables", *The American Mathematical Monthly*, Vol. 59, No. 8, pp. 521-531, 1952.

[11] Claude E. Shannon, "Symbolic Analysis of Relay and Switching Circuits", *Transactions American Institute of Electrical Engineers*, Vol. 57, No. 6, pp. 713-723, 1938.

[12] S.K. Petrick, "On the Minimization of Boolean Functions", *Proceedings of the International Conference on Information Processing*, pp. 422-423, 1959.