HAL
open science

Breakup Algorithm for Switching Circuit Simplifications

Sahadev Roy

To cite this version:

Sahadev Roy. Breakup Algorithm for Switching Circuit Simplifications. International Journal of Advanced Engineering and Management, 2016, 1 (1), pp.1 - 11. 10.24999/IJOAEM/02010001 . hal-01484223

HAL Id: hal-01484223

https://hal.science/hal-01484223v1

Submitted on 7 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

International Journal of Advanced Engineering and Management
Vol. 1, No.1, PP. 1-11, 2016
Received on: 22.10.2016
Revised on: 27.11.2016
www.ijoaem.com

Breakup Algorithm for Switching Circuit Simplifications

Sahadev Roy
Dept. of ECE, NIT Arunachal Pradesh, Yupia, 791112, India
e-mail:sdr.ece@nitap.in

Subject Category: Engineering
Sub Category: *Electronics*

Editor: ***M Mukherjee***

Volume 1 Issue 1 December 2016

Minimization table using octal minterms

Octal Minterms*	Pairing between common LSP	Pairing between common MSP	Simplified Product terms
0|2 √	(0|2, 2|2): –0|2 √	–0|(2, 3, 6,7) –0|–1–	$\bar{B}D$
2|2 √			
0|3 √	(0|3, 2|3): –0|3 √		
2|3 √			
0|6 √	(0|6, 2|6): –0|6 √		
2|6 √			
0|7 √	(0|7, 2|7): –0|7 √		
2|7 √			
3|0 √		3|(0, 4): 11|– 00	$AB\bar{D}\bar{E}$
1|3√		(0|3, 1|3): 0–|011	$\bar{A}\bar{C}DE$
1|4 √	(1|4, 3|4): –1|4√	– 1|(4, 5): – 1|10–	$BC\bar{D}$
3|4 √			
1|5 √	(1|5, 3|5): –1|5√		
3|5 √			

Keywords: *Breakup algorithm;*
Combinational circuit simplification;
Hexadecimal coded minterms;
Logic function simplification;
Octal coded minterms.

IJOAEM/1/1/1/1

Breakup Algorithm for Switching Circuit Simplifications

Sahadev Roy

Abstract

This paper examines the generalized schemes of the breakup algorithm. It is possible to minimize multiple variable Boolean functions by breaking up into several groups. In the proposed generalized method, the number of group depends on user choice. The maximum element in each group depends on breaking point and it is quite easy to determine logic adjacency using look up table. Four point break or three point break are easily expressed as hexadecimal minterms and octal minterms respectively and both the techniques produce the same minimized results which are analyzed in this paper. A generalized Breakup Algorithm is also presented here for sequential and combinational logic circuit minimization.

1. Introduction

Exact minimization of multiple input switching circuits is a challenging research. Before Veitch, minimization of logic circuit mainly has been done using simple Boolean algebra. A chart-method was proposed by Veitch to avoid lengthy calculations [1]. Brooks tried to extend this approach for application in computer aided designs [2]. In a technical note, he tried a systematic representation of different switching element like NOT, AND, OR, NOT-AND (NAND), NOT-OR (NOR). By combining different gates, buffers, inverters and delay elements, various designs like binary addition, subtraction, multiplication, division, shift register and also different other function generation were also demonstrated by him [3]. In 1953, Veitch's Chart method of minimization was modified by Karnaugh; his proposed Map is one of the simple methods for manual synthesis and suitable up to six variables [4]. This approach is quite systematic [5]. The results of this method are shown in the form of the SOP or sum-of-products and POS or product-of-sums. For n number of the input variables, K-map requires 2^n cells [6]. It is mainly a visual simplification procedure, which solely depends on the designer's capability.

Using variable-entered maps it is possible to generate compact parametric generalized solutions of any logic function [7]. Another traditional graphical approach is Binary Decision Diagram or BDD. The basic concepts of the BDD are derived from the Shannon expansion technique. Any logic function is possible to split into two branches which may be considered as cofactors or sub-functions by considering only one variable. These are connected by if-then-else logic. Each sub-function may be realized as a sub-tree of the BDD. Lee introduced the BDD in 1959 [8] and this techniques was further developed by Akers [9] and Bout [10]. In 1986, Randal Bryant [11] proposed the fundamental concept of fixed variable ordering especially for canonical representation of the function and shared sub-graphs particularly for compression. These two methodologies are widely used in the data structure and also for the representation of sets and different relations. One sub-graph may be utilized by several BDDs which are known as Shared Reduced Ordered Binary Decision Diagram. BDDs are comprehensively used in CAD for logic minimization and formal verification of the design [12]. Every arbitrary BDD can be realized by 2 to 1 multiplexer and directly implemented by 4-LUT [13] in an FPGA [14]. Implementation of a logic circuit using Threshold logic [15] also belongs to this category.

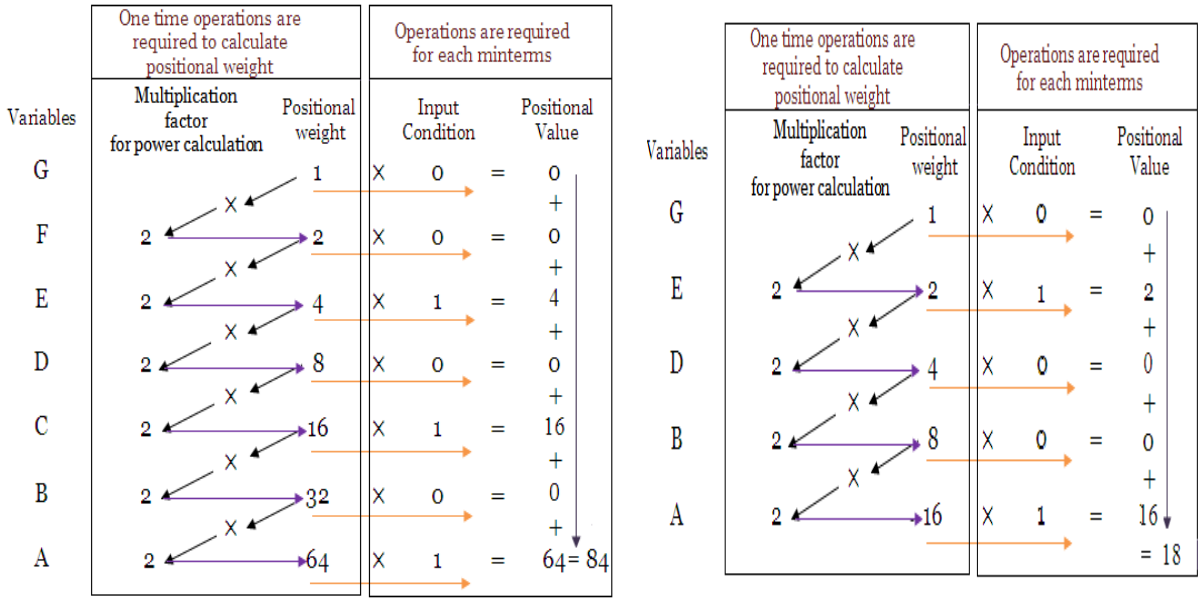
2. The Breakup Algorithm

Any switching functions can be realized into two or several sub-functions. In our proposed method, sub-functions are realized by breaking of minterms. We have divided input variables into two sets: (i) Reference Set (RS) and (ii) Derived Minterms (DM). We can optionally put any number of input variables into RS. Using the rest of the literal we have made DM. To indicate break up point the symbol " |" is used here.

The proposed "Breakup Algorithm" is illustrated using a simple example. Consider a seven variable system with inputs A, B, C, D, E, F and G. We consider the minterms $A\bar{B}C\bar{D}E\bar{F}\bar{G}$, to illustrate the proposed method. To

calculate its equivalent decimal values we have to perform the complex operations as shown in the Fig.1(a). Six multiplications are required to determine the positional weight which is one time operation. Another seven multiplications and six additions are required to calculate decimal value for each minterms. Assigned positional weights of the inputs are, $A = 64, B = 32, C = 16, D = 8, E = 4, F = 2$ and $G = 1$.

Let input C and F be considered as reference inputs. Possible combination of C and F are $\bar{C}\bar{F}(0)$, $\bar{C}F(1), C\bar{F}(2)$ and $CF\,(3)$. These combinations are considered as Reference Set (RS). Two multiplications are required to determine the positional weight of each RS which is one time operation. Another five multiplications and four additions are required to calculate derived minterms for each input combination as shown in Fig. 1(b). Reassigned positional weight of the inputs as, $\mathrm{A} = 16, \mathrm{B} = 8, \mathrm{D} = 4, \mathrm{E} = 2$ and $\mathrm{G} = 1$.



(a) (b)

Figure 1. Illustration of required operations considering seven input switching system (a) for decimal coded minterms and (b) for derived minterms, considering C and F as the reference inputs (Arrowhead indicates actions follow).

3. Generalized Minimization Rule For RS|DM

Any particular DM may be associated with any number of RS. If the associated RS are adjacent then as a whole they are adjacent. Any non-adjacent RS pairs of particular DM are not allowed for minimizations. This fact also true for a particular RS associated with adjacent DM. In the following sub sections we explain our proposed method briefly.

a) Illustration of three points breakup

Here, we illustrated three variables grouping starting from LSB.

I. Example 1:

Consider, f (A, B, C, D, E) has two minterms $\bar{A}\bar{B}CD\bar{E}$ i.e. 00110 and $\bar{A}BCD\bar{E}$ i.e. 01110. Variables A and B are treated as RS and C, D and E considered as DM, hence we can represent as three point breakup, which is same as octal coding techniques. The minterms are represented as $\bar{A}\bar{B}CD\bar{E}$ as 00|110 and $\bar{A}BCD\bar{E}$ as 01|110. Here, input conditions of variables A, C, D and E are same i.e. 110. Variable B, has the only has different input conditions in RS hence these two minterms may be consider as adjacent. It may be minimized as (0–|110) i.e. $\bar{A}CD\bar{E}$.

II. Example 2:

Let we consider a 6 inputs combinational logic system and 2 inputs treat as referenced input. Input variables are identified with alphabets A to F and A treat as highest weighted literals and F as lowest weighted literals. Now we

consider two adjacent minterms, $\bar{A}BC\bar{D}\bar{E}\bar{F}$ (011|000) and $ABC\bar{D}\bar{E}\bar{F}$ (111|000) for an example. DM0 ($\bar{D}\bar{E}\bar{F}$) is the common factor in both of the references set RS3 ($\bar{A}BC$) and RS7 (ABC) hence we can combine two minterms as (3, 7|0) the both the RS1 and RS3 are adjacent set with respect to DM0. RS3 and RS7 can be written as ($-$BC) so the overall minimized terms is $BC\bar{D}\bar{E}\bar{F}$ or –11000.

III. Example 3:

For the case of (1, 2|6) can't be pair because 1 and 2 are not adjacent i.e. $\bar{A}B\bar{C}DE\bar{F}$, $AB\bar{C}DE\bar{F}$ are not adjacent.

IV. Example 4:

For minterms, $\bar{A}\,\bar{B}C\bar{D}E\bar{F}$, $\bar{A}\,BC\bar{D}E\bar{F}$, $A\,\bar{B}C\bar{D}E\bar{F}$, and $ABC\bar{D}E\bar{F}$, it can be represent as (1, 3, 5, 7|2). Here, DM2 ($\bar{D}E\bar{F}$) is common for all possible combinations of A and B variable in the RS keeping C as fixed, simplified terms is $--1|010$ or simply $C\bar{D}E\bar{F}$.

b) Illustration of four points breakup

Here, we illustrated four variables grouping starting from LSB.

For 'Example 1' these two minterms, $\bar{A}\bar{B}CD\bar{E}$ i.e. 00110 and $\bar{A}BCD\bar{E}$ i.e. 01110 can be represent as 0|0110 and $\bar{A}BCD\bar{E}$ as 0|1110. It may be minimized as (0|–110) i.e. $\bar{A}CD\bar{E}$.

For *'Example 2'* As per four point breakup we may represent those minterms as, $\bar{A}B|C\bar{D}\bar{E}\bar{F}$ (01|1000) and $AB|C\bar{D}\bar{E}\bar{F}$ (11|1000). DM 8 ($C\bar{D}\bar{E}\bar{F}$) is the common factor in both of the references set RS1 ($\bar{A}B$) and RS3 (AB) i.e. DM (1,3|8). RS1 and RS3 can be minimized as B, so the overall minimized terms is $BC\bar{D}\bar{E}\bar{F}$ or –11000.

For 'Example 3' the DM (1,2|6) can't be paired because 1 and 2 are not adjacent i.e. $\bar{A}B|\bar{C}DE\bar{F}$, $A\bar{B}|\bar{C}DE\bar{F}$.

For 'Example 4' DM (0,1,2,3|10) i.e. DM 10 is common for all possible RS. Then Actual minterms are, $\bar{A}\,\bar{B}|C\bar{D}E\bar{F}$, $\bar{A}\,B|C\bar{D}E\bar{F}$, $A\,\bar{B}|C\bar{D}E\bar{F}$, and $AB|C\bar{D}E\bar{F}$, here both literals A and B have all possible combination and it product terms is , $--1010$ or $C\bar{D}E\bar{F}$.

These results are same with three point breakup results.

4. Generalized Breakup Technique

Let any n inputs switching system and m inputs treat as referenced variables. Variables are identified by 'V' with proper positional suffix and b is identified same input condition and by 0 for a complement of that variable otherwise 1. Variables of position 0 to n–m–1 form DM and rest form RS.

For a RS (here, $bV_{n-m-1} \cdots bV_0$) are adjacent with respect to V_{n-j} which are associated with DM, ($bV_{n-m-1} \cdots bV_0$). Where j any position in between (n – 1) to (n – m). Here, in the two minterms only V_{n-j} values are different hence they are adjacent and V_{n-j} variable may consider as don't care. In all other nonadjacent cases there are exist no such literal in RS part hence they are not adjacent. Due to that reason, simplification further minimizations is not possible.

If any DM is common to all RS then all literals of RS are consider as don't care and the minimized result must be only DM.

As example for these four minterms 00|111, 01|111, 10|111 and 11|111; right hand part 111 are same left hand part are 00, 01, 10 and 11 i.e. all possible combinations of 0 and 1 hence they are minimized as - -|111. For minimization we only compare left part hence less number of comparison are required with existing methodology.

5. Software Implementations Technique of the Proposed Four Points Breakup Method

The tabular technique is extended here to generate SOP term. This method can be used to implement in computer. We consider minterms that's directly given in hexadecimal code obtain from truth table. A complete analysis is performing on the basics of required number of comparison. A lookup table is formed based on adjacent minterms (Fig. 2). All the required information of adjacent minterms may be acquired from K-map [16].

For an example, we consider here a five variables switching function as Eq. (1)

$$f(A,B,C,D,E) = \sum m(2,3,6,7,11,12,13,18,19,22,24,28,29) \quad \textbf{(1)}$$

Since minterms are given in decimal coded we have to convert in Hex-minterms. This step may be avoided at the beginning of conversion from the truth table. Anyway, equivalent hex-minterms representation is given below,

$$f(A,B,C,D,E) = \sum m_H(\mathbf{02}, \mathbf{03}, \mathbf{06}, \mathbf{07}, \mathbf{0B}, \mathbf{0C}, \mathbf{0D}, \mathbf{12}, \mathbf{13}, \mathbf{16}, \mathbf{17}, \mathbf{18}, \mathbf{1C}, \mathbf{1D}\ ) \quad \textbf{(2)}$$

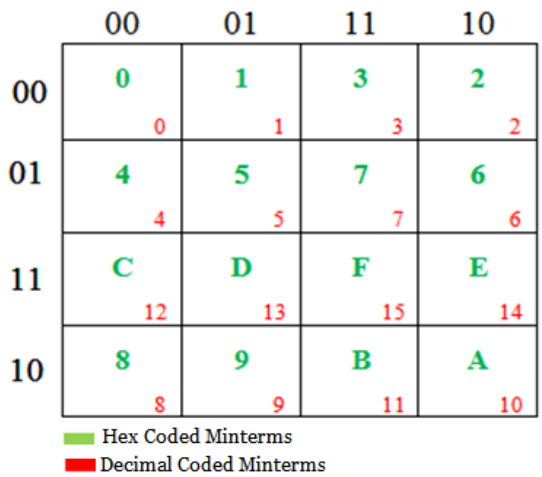


Figure 2. *Hex terms adjacent chart.*

First step shows all pairing between common LSP (Column 2) and Second step shows pairing between common MSP (column 3) and last column represent minimal realization in Table I. Under '()' pairing are indicated here for easy realization. Since all minterms are included in single in prime implicants so all are essential prime implicants, Prime implicants chart not required hence minimization expression is,

$$f(A,B,C,D,E) = \bar{B}D + AB\bar{D}\bar{E} + \bar{A}\bar{C}DE + BC\bar{D} \quad \textbf{(3)}$$

Table 1. Illustration Minimization Table using Hex-term

Hex-minterms*	Pairing between common LSP	Pairing between common MSP	Simplified Product term
0|2 √	(0|2, 1|2): –|**2** √	**{ 0|(2, 3, 6,7), 1|(2, 3, 6,7)}→ –|0–1–**	$\bar{B}D$
1|2 √			
0|3 √	(0|3, 1|3): –|**3** √		
1|3 √			
0|6 √	(0|6, 1|6): –|**6** √		
1|6 √			
0|7 √	(0|7, 1|7): –|**7** √		
1|7 √			
1|8 √		**(1|8,1|C) → 1|1– 00**	$AB\bar{D}\bar{E}$
0|B √		**(0|3, 0B)→ 0|–011**	$\bar{A}\bar{C}DE$
0|C √	(0|C, 1|C): –|**C** √	**(–|C, –|D) → –|110–**	$BC\bar{D}$
1|C √			
0|D √	(0|D, 1|D): –|**D** √		
1|D √			

* MSP is single bit due to five input switching system; 0 and 1 (Hexadecimal Number) in MSP is same with 0 and 1 otherwise equivalent with four bit binary number. Read colour digits are binary numbers.

Step 1: Grouping of common MSP:

Based on Eq. (2) all minterms are divided in group 0 and group 1 which are shown in Table 2.

Table 2. Step Wise Grouping Procedure

Group 0	*Group 1*
2	2
3	3
6	6
7	7
B	8
C	C
D	D

Here for segregation of minterms in two group required comparison (A) = 14.

Step 2: Listed common entry in adjacent MSP:

Each elements of group 0 are compared with elements of group 1.

In first iteration all seven elements of group 2 compare with first element of group 1 i.e. with 2 and in first comparison match is found then break the iteration and matched elements are store in 'common cube 1'.

Second iteration is started from second element i.e. excluding match elements. Non match terms are kept in location 'cube 1' with group information. The whole processed is completed in seven iteration and 11 comparisons, shown in Table 3.

Table 3. Grouping of minterms for first cube formation

Group	Elements							No of iteration
0	2	3	6	7	B	C	D	
1	2	3	6	7	8	8	8	1
	3	6	7	8	C	C	D	2
	6	7	8	C	D	D		3
	7	8	C	D				4
	8	C	D					5
	C	D						6
	D							7
Comparison	1	1	1	1	3	2	2	Total comparison = **11 (B)**
Common cube 1	2	3	6	7		C	D	
Cube 1	0B, 18							

Step 3: Searching for adjacent LSP from common LSP of both group for formation of cube 2:

Only all elements of 'common cube 1' satisfy essential condition for formation of cube 2. For checking of adjacency lookup table is useful to avoid unnecessary comparison. First elements of 'common cube 1' is compare to rest of elements and for all adjacent elements equivalent adjacent term are placed in cube 2. The element which compared with the reset element we consider as reference elements. In second iteration reference elements is the second elements and comparison start from next element.

If any references hex term unable to combine it is simply converted in binary form and stored in array otherwise corresponding combine terms from the lookup table store in the same array, for this purpose a probe or flag may be used to determine un-combined terms. This technique effectively reduced number of comparison and also less number of duplicate pair as like Quine McCluskey method. Details comparisons are shown in the Table 4 and Table 5. Un-combined term in Table II is 0B and 18. The minterms are compared with in its group because of their not existence of any minterms in other group with same LSP or MSP. So, searching must be limited in between intra group minterms only. To find adjacent minterms in their corresponding group are determined by table 6. Since 0B compare within group 0 and 18 compare within group 1 so it is not required to compare whole term only LSP comparison is sufficient.

Table 4. Grouping of minterms for second cube formation

Iteration No	Ref. LSP	Elements/ combine term					Output	No of comparison
1	2	3	6	7	C	D	001-, 0-10	5
		001-	0-10					
2	3	6	7	C	D		0-11	4
			0-11					
3	6	7	C	D			011-	3
		011-						
4	7	C	D					2
5	C	D					110-	1
		110-						
Total comparison								15 (C)

Table 5. Grouping of minterms for third cube formation

Reference term	Intermediate Steps					Output	Final output	No of Comparison
001-	Remaining terms	0-10	0-11	011-	110-	0-1-	{0|(2, 3, 6, 7), 1|(2, 3, 6, 7)} → -0-1- → ($\overline{B}$D) {(0, 1)|C, (0,1)|D}→ -110- → BC$\overline{D}$	3+ 3 + 4 +3 = 13
	Combine terms	NIL	NIL	0-1-	NIL			
0-10	Remaining terms	NIL	0-11	011-	110-	0-1-		4+3+2 = 9
	Combine terms	NIL	0-1-	NIL	NIL			
0-11	Remaining terms	NIL	NIL	011-	110-			3+2 = 5
	Combine terms	NIL	NIL	NIL	NIL			
110-	Remaining terms	NIL	NIL	NIL	011-			4
	Combine terms	NIL	NIL	NIL	NIL			
Cancellation of duplicate entry								8
Total Comparison								39(D)

Same set of minterms of the Eq. (1) break into three points can easily expressed into octal code as Eq. (4),

$$f(A, B, C, D, E) = \sum m_8(02, 03, 06, 07, 13, 14, 15, 22, 23, 26, 30, 34, 35) \qquad (4)$$

Using adjacent octal chart shown in Fig. 5 the above expression can be easily minimized.

This Eq. (4) can be easily minimized by the Table VIII. Under '()' pairing is indicated here for easy realization. Since all minterms included in single in prime implicants so all are essential prime implicants,

Prime implicants chart not required hence minimization expression is given in Eq. (3)

Detailed analysis is done using Quine-McCluskey method it requires total 639 comparisons, but using the proposed method only 91 comparisons is required.

Table 6. Intermediate steps to find adjacent terms 0b and 18

Intermediate Steps							Comparison	Final output
B							6 X 1 = 6	0|(B, 3)**0-011** $\overline{A}\,\overline{C}$DE
Minterms of Group 0	2	3	6	7	C	D		
Minimized Term		-011						
No. of Comparison	1	1	1	1	1	1		
8								1|(8, C) **11-00** $AB\overline{D}\overline{E}$
Minterms of Group 0	2	3	6	7	C	D	6 X 1 = 6	
Minimized Term					1-00			
No. of Comparison	1	1	1	1	1	1		
Total Comparison							**12 (E)**	

Table 7. Total comparison required for 5 variable 13 minterms

From Table	Sub comparison	Value	
7.6	A	14	Using Quine-McCluskey method it's required total 639 comparisons
7.7	B	11	
7.8	C	15	
7.9	D	39	
7.10	E	12	
Total comparison		91	

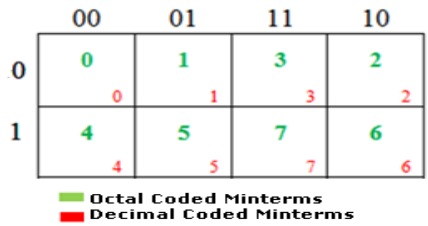


Figure 3. Octal minterms adjacent chart.

Table 8. Minimization table using octal minterms

Octal Minterms*	Pairing between common LSP	Pairing between common MSP	Simplified Product terms
0|2 √	(**0**|2, **2**|2): –0|2 √	–0|(2, 3, 6,7) **–0|–1–**	$\overline{B}D$
2|2 √			
0|3 √	(**0**|3, 2|3): –0|3 √		
2|3 √			
0|6 √	(**0**|6, 2|6): –0|6 √		
2|6 √			
0|7 √	(**0**|7, 2|7): –0|7 √		
2|7 √			
3|**0** √		3|(**0**, 4): **11|– 00**	$AB\overline{D}\overline{E}$
1|3√		(**0**|3, **1**|3): **0–|011**	$\overline{A}\overline{C}DE$
1|4 √	(**1**|4, 3|4): –1|4√	– 1|(4, 5): **– 1|10–**	$BC\overline{D}$
3|4 √			
1|5 √	(**1**|5, 3|5): –1|5√		
3|5 √			

* MSP is single bit due to five input switching system;0 and 1 (Octal Number) in MSP. In MSP octal numbers are consider two bit otherwise equivalent with three bit binary number. Read colour digits are binary numbers.

6. Comparison and Analysis

In general for traditional approach, for n number input functions $n-1$ multiplication are required to determine the positional weight and it is one time operations. For each minterms required number of multiplication and addition is n and $n-1$ respectively (Fig.1). Total numbers of operation required for p minterms are $(n-1)+p(2n-1)$.

In the proposed approach, if m number inputs treated as reference input $(n-m-1)$ multiplication are required to determine the positional weight and it is one time operations. For each minterms required number of multiplication and addition is $(n-m)$ and $(n-m-1)$ respectively (Fig.3). Total numbers of operation required for k minterms are $(n-m-1)+p(2n-2m-1)$. At least $(2p+1)m$ operations are not required for m point breakup. For three point breakup or octal coded minterms maximum number of groups are $(n/3)\uparrow$. The symbol "↑" used here to indicate if the division become fraction we have to consider next higher integer. The required number of multiplications to determine the positional weight are $2(n/3\uparrow)$ because three variables present in each group. For each group three multiplications and two additions are required i.e. five operations in each group. Hence, maximum $5(n/3\uparrow)$ number operations are required for each minterms along with $2(n/3\uparrow)$ number one

time operations. For p minterms to represent in octal minterms or three points Breakup Algorithm, $(5p + 2)(n/3 \uparrow)$ operations are required. Similarly, for four points Breakup Algorithm or hexadecimal minterms maximum $(7p + 3)(n/4 \uparrow)$ operations are required for p minterms.

The benchmark circuit "Z9sym" consists of 420 minterms and it is a function of nine inputs variables [17]. To represents all its minterms in traditional decimal code and octal code required 7148 and 6306 operations. For hexadecimal code representation three digits are required to represent nine bits. The eight bits from the LSB form two group and for each groups seven operations are required and for the last digit which contain single bit has required one operation per minterms. So, fifteen operations needed to perform for each minterms to convert into hexadecimal minterms along with six multiplication operation for weight calculations. Interestingly, same number operations are required for octal minterms and hexadecimal minterms for Z9sym. When highest digits of the hexadecimal contain only single digit, the required operations for octal code and hexadecimal code is same. In general octal code representation less number of operations is needed to perform in comparison with hexadecimal minterms and decimal minterms.

For example 2, if it is solved using K-map method then it requires two blocks and which is also a complex procedure to determine adjacent cell between two blocks. The cost of the minimal expression is same with Map Enter Variable method [25] but to solve in this process required less number of steps. A vast investigation is done based on required comparison proposed method required comparisons are 91 only but using Quine-McCluskey method 639 comparisons are required. The Quine-McCluskey algorithm has provided the solution based on combinational optimization technique which belongs to NP-hard type solution. Runtime of the Quine-McCluskey algorithm increases with the number of input variable due to a large number of comparisons which provides the solution of the limited range of input variable.

The proposed minimization method, the required compression is less than the Quine-McCluskey algorithm and also reduces the required runtime effectively. For example 3 designs cost is 19 PI with 98 inputs using proposed method which is same with exact analysis mode. The major drawback of the most popular representation using truth table is unable to simplify the logic function. Another drawback is its size as e.g. for four input variable required sixteen rows and five columns to shows all combinations. For eight variable systems, possible numbers of minterms are 256. Approximate 90 rows of 0.1inch height are fitted in A4 size paper, to fit all minterms into a truth table required approximately three pages. Representations of minterms of multiple inputs are the big issues. If minterms are given in decimal number then all minterms first convert into the binary number for further process. Quine-McCluskey minimization method also deals with binary coded minterms. Representation of minterms in binary coding also required more space and not user-friendly. In another hand, Karnaugh MAP representation technique is more compact and required only four rows and four columns. But the major advantage of Karnaugh MAP is minterms are placed in the logic adjacent cell directly. Using logical adjacency property easily obtains minimal SOP from by less effort compare to other methods, by which this technique able to keep its popularity till the date. A serious drawback of this method is not suitable for more than six variables. K-map for more than four variables divided in level and each level contain sixteen cells. For six inputs system, possible numbers of minterms are 64 so a number of levels are four. For seven and eight inputs system required the number of map level are eight and sixteen respectively and also identification of adjacent cell become more difficult. This process also increases complexity minimization of multiple inputs logic synthesis. Decimal coded minterms is not played any significance role in minimization process except representations of input conditions. These difficulties successfully overcome using proposed hex-minterms representation and minimization technique. Karnaugh map is also used in manual minimization technique using DCM, but proper minimization method becomes complicated for more than four input switching system. The Karnaugh map method of minimization procedure is quite simple and straight a forward approach for four variable switching systems. From the case of five inputs, variable map divided into two blocks and for six input system have four blocks. In general for n input systems, $2^{(n-4)}$ number sub-blocks for and each sub-block has sixteen cell and complicated procedure to determine adjacent cell of different blocks. Another drawback of this method is that numbers of cells are 2^n though the numbers of minterms may be less. Karnaugh map is manual minimization methods and software implementation this process is quite tricky.

Another advantage to using reduced adjacent pair table, it is not required to search for the pair in lower weighted minterms with respected to reference bit, because it is already done searches when those lowest weighted bits acts as reference bit. This method prevents to produce dummy pairs and dummy prime implicants. The Quine - McCluskey algorithm is NP-hard type minimization technique and the number of comparison increase in the

number of input variables which increase the runtime of this algorithm. Propose minimization method is required less compression and less runtime than the Quine-McCluskey algorithm. In Quine-McCluskey Minimum three comparisons are required to simplify product terms for four pairs, but using proposed method, by one comparison is required. This technique effectively reduces complexity and also improves the speed of simplification due to 33% less operations are required. The results, which are verified with available experimental data, indicated that the minimal sum-of-product terms are achieved with minimal effort. DCM is not suitable for minimization because all simplifications approached are based on binary representation or Gray code. But binary numbers are not suitable for realization due to a number of digits. The method easily overcome or minimized all those difficulties successfully.

7. Conclusion

A new approach is proposed here for simplification of multi-input switching circuit based on octal minterms and hexadecimal minterms which reduce complexity effectively. The proposed hex coded and octal coded minterms representations techniques are shorter and easily converted in the binary number which reduces the complexity of minimization. The proposed method is cost effective. This method can easily be used for any number of variables. The proposed method also effectively reduces the required number of comparison with respect to other exiting methods. A large number of minterms reduces complexity due to less number of comparisons required in this method. The tabular technique is extended here to generate SOP term. This method can be used to implement in computer. The proposed method is applicable to single and multi-output switching circuits for any number of switching variables. Both octal coded minterms and hexadecimal coded minterms provide same result hence three point breakups or four point breakups provide same result.

References

[1] E. W. Veitch, “A chart method for simplifying truth functions,” *Transactions of the ACM Annual Meeting,* pp. 127-133, 1952.

[2] R. W. Brooks, “Extension of the Veitch Chart Method in Computer Design,” *Meeting of Association for Computing Machinery, Cambridge, Massachusetts,* pp. 1-15, September, (1953).

[3] R. W. Brooks, “Symbolic logic, binary calculation, and 3C- PACS,” *Computer Control Co. Inc, Wellesley,* vol. 57, pp. 1-15, 1955. Available at: http://www.ddp116.org/products/pacs/logic_1955.pdf.

[4] M. Karnaugh, “The map method for synthesis of combinational logic circuits,” *AIEE Committee on Technical Operations for presentation at the AIEE summer General Meeting,* pp. 593-599, 1953.

[5] S. Roy and C. T. Bhunia, “A Set Combine Map Method for Manual Synthesis of Logic Circuits,” *International Journal of Applied Engineering Research,* vol. 11, no. 4, pp. 2663-2670, 2016.

[6] S. Roy and C. T. Bhunia, “On Synthesis of Combinational Logic Circuits,” *International Journal of Computer Applications,* vol. 127, no. 1, pp. 21-26, 2015.

[7] A. M. Rushdi, “Efficient Solution of Boolean Equations Using Variable-Entered Kamaugh Maps,” *Engineering Sciences,* vol. 15, no. 1, pp. 105-120, 2004.

[8] C. Y. Lee, “Representation of Switching Circuits by Binary-Decision Programs,” *Bell Systems Technical Journal,* vol. 38, pp. 985-999, 1959.

[9] S. B. Akers, “Binary Decision Diagrams,” *IEEE Transactions on Computers,* Vols. C-27, no. 6, pp. 509-516, June 1978.

[10] R. T. Boute, “The Binary Decision Machine as a programmable controller,” *EUROMICRO Newsletter,* vol. 1, no. 2, pp. 16-22, January 1976.

[11] R. E. Bryant, “Graph-based algorithms for boolean function manipulation,” *IEEE Transactions on Computers,* vol. 100, no. 8, pp. 677-691, 1986.

[12] G. Lee, “Logic synthesis for cellular architecture FPGAs using BDDs,” *Design Automation Conference, 1997. Proceedings of the ASP-DAC'97 Asia and South Pacific,* vol. IEEE, pp. 253-258, 1997.

[13] J. Wawrzynek, “EECS150-Digital Design: Lecture 5—Field Programmable Gate Arrays (FPGAs),” pp. 1-20, Feb. 4 Feb 4, 2002.

[14] G. Lee, “Logic synthesis for cellular architecture FPGAs using BDDs,” *Design Automation Conference, 1997. Proceedings of the ASP-DAC'97 Asia and South Pacific,IEEE,* pp. 253-258, 1997.

[15] R. Kumar, P. Kumar and R. Sahadev, "Efficient minimization Techniques for Threshold Logic Gate," *International Research Journal of Engineering and Technology,* vol. 3, no. 4, pp. 1375-1382, 2016.

[16] S. Roy and C. T. Bhunia, " Simplification of Switching Functions Using Hex-Minterms," *International Journal of Applied Engineering Research,* vol. 10, no. 24, pp. 45619-45624, 2015.

[17] S. Yang, "Logic synthesis and optimization benchmarks user guide: version 3.0.," *Microelectronics Center of North Carolina (MCNC),* 1991.