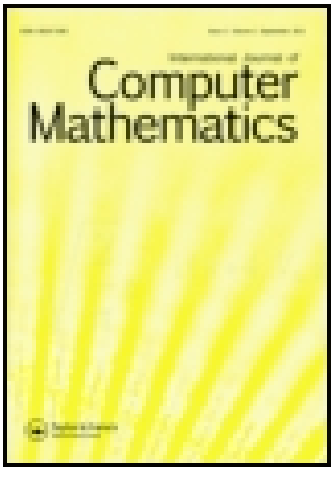

International Journal of Computer Mathematics

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcom20

Partitioning 1-variable Boolean functions for various classification of n-variable Boolean functions

Ranjeet Kumar Rout[a], Pabitra Pal Choudhury[a], Sudhakar Sahoo[b] & Camellia Ray[a]

[a] Applied Statistics Unit, Indian Statistical Institute, Kolkata-700108, India;

[b] Institute of Mathematics and Applications, Bhubaneswar-751003, India

Accepted author version posted online: 14 Oct 2014.Published online: 14 Nov 2014.

To cite this article: Ranjeet Kumar Rout, Pabitra Pal Choudhury, Sudhakar Sahoo & Camellia Ray (2014): Partitioning 1-variable Boolean functions for various classification of n-variable Boolean functions, International Journal of Computer Mathematics, DOI: 10.1080/00207160.2014.975418

To link to this article: http://dx.doi.org/10.1080/00207160.2014.975418

International Journal of Computer Mathematics, 2014
http://dx.doi.org/10.1080/00207160.2014.975418

Partitioning 1-variable Boolean functions for various classification of *n*-variable Boolean functions

Ranjeet Kumar Rout[a]*, Pabitra Pal Choudhury[a], Sudhakar Sahoo[b] and Camellia Ray[a]

[a]*Applied Statistics Unit, Indian Statistical Institute, Kolkata-700108, India;* [b]*Institute of Mathematics and Applications, Bhubaneswar-751003, India*

(*Received 18 July 2014; revised version received 3 October 2014; accepted 7 October 2014*)

This paper addresses all possible equivalence classes of 1-variable Boolean functions and from these classes using recursion and Cartesian product of sets, 15 different ways of classifications of *n*-variable Boolean functions are obtained. The properties with regard to the size and the number of classes for these 15 different ways are also elaborated.

Keywords: integer partition; affine Boolean function; truth table; classification; carry value transformation; XOR operation; Hamming distance

2005 AMS Subject Classifications: 06E30; 94C10; 94A60

1. Introduction

A partition of a positive integer n is a collection of positive integers whose sum is n. For example, there are five partitions of 4 and these can be represented as 4, $3+1$, $2+2$, $2+1+1$ and $1+1+1+1$ [1,2]. Classification or partitioning of Boolean functions has been a long standing problem in the field of theoretical computer science. Classification of Boolean functions based on a number of important properties, such as balancedness, degree of nonlinearity, high correlation immunity, neutrality, linearity, self-duality, monotonicity, reversibility, etc. have been discussed in [4,5,8,10,13,14]. Many properties of these functions are well known and other properties are yet to know; however, due to their exponentially growing number, detailed knowledge can be obtained only for functions depending on a small number of variables. In order to utilize these and other properties of Boolean functions, several approaches of classifications were suggested, an overview can be found in [3,11]. Such classifications are beneficial because large sets of Boolean functions observing a given classification rule can be represented by a single representative, and this can support the use of large numbers of Boolean functions more efficiently [12]. A systematic classification of Boolean functions with *n*-variable having an affine function as representative in each class is reported in [9]. In this paper, similar to that, 15 other equivalence classes of Boolean functions are generated by using the Cartesian product of the sets, starting from four 1-variable Boolean functions. The properties of all these classes have been thoroughly derived and have been included to make the paper complete and beneficial to the academic community.

*Corresponding author. Email: ranjeetkumarrout@gmail.com

In the following sections, the paper is organized in the following way. In Section 2, distinct partitions of 1-variable Boolean functions are discussed; the result of Section 2 will be more useful, if we start with a set of higher cardinality, instead of four. For example, we could have started with the set of sixteen 2-variable Boolean functions. In Section 3, the method of recursive classification of n-variable Boolean functions is introduced and the properties of these classes are discussed. In Section 4, we have studied the behaviour of those classes by using different binary operations such as Hamming distance (HD), Exclusive disjunction or exclusive or (XOR) operation and carry value transformation (CVT) [6,7]. Section 5, deals with concluding remarks emphasizing the key factors of the entire analysis.

2. Distinct partitions of 1-variable Boolean functions

Let $S_1 = \{\{00\},\ \{10\},\ \{11\},\ \{01\}\}$ be a set of all 1-variable Boolean functions, the cardinality of the set S_1 is 4. Therefore S_1 can be partitioned into 15 distinct partitions corresponding

Table 1. Shows distinct partition of 1-variable Boolean function.

Class type	Summand	DP of 4	Distinct partition of 1 variable BF (base case)	Equation no.
Type-I	1	1	$S_1' = \{\{00\}\{01\}\{10\}\{11\}\}$	(1)
		$2+2$	$S_1' = \{\{00\}\{01\}\}, S_1'' = \{\{10\}\{11\}\}$	(2)
			$S_1' = \{\{00\}\{10\}\}, S_1'' = \{\{01\}\{11\}\}$	(3)
			$S_1' = \{\{00\}\{11\}\}, S_1'' = \{\{01\}\{10\}\}$	(4)
Type-II	2		$S_1' = \{\{00\}\{01\}\{10\}\}, S_1'' = \{\{11\}\}$	(5)
		$3+1$	$S_1' = \{\{00\}\{01\}\{11\}\}, S_1'' = \{\{10\}\}$	(6)
			$S_1' = \{\{00\}\{10\}\{11\}\}, S_1'' = \{\{01\}\}$	(7)
			$S_1' = \{\{01\}\{10\}\{11\}\}, S_1'' = \{\{00\}\}$	(8)
			$S_1' = \{\{00\}\{01\}\}, S_1'' = \{\{10\}\}, S_1''' = \{\{11\}\}$	(9)
			$S_1' = \{\{00\}\{10\}\}, S_1'' = \{\{01\}\}, S_1''' = \{\{11\}\}$	(10)
Type-III	3	$2+1+1$	$S_1' = \{\{00\}\{11\}\}, S_1'' = \{\{01\}\}, S_1''' = \{\{10\}\}$	(11)
			$S_1' = \{\{01\}\{10\}\}, S_1'' = \{\{00\}\}, S_1''' = \{\{11\}\}$	(12)
			$S_1' = \{\{01\}\{11\}\}, S_1'' = \{\{10\}\}, S_1''' = \{\{00\}\}$	(13)
			$S_1' = \{\{10\}\{11\}\}, S_1'' = \{\{00\}\}, S_1''' = \{\{01\}\}$	(14)
Type-IV	4	1+1+1+1	$S_1' = \{\{00\}\}, S_1'' = \{\{01\}\}, S_1''' = \{\{10\}\}, S_1'''' = \{\{10\}\}$	(15)

Table 2. Shows recursion of distinct partition of n-variable Boolean function.

Summand	Recursion equation	Equation no.
1	$S_n = (S_{n-1} \times S'_{n-1})$	(16)
2	$S'_n = (S_{n-1} \times S'_{n-1}), S''_n = (S_{n-1} \times S''_{n-1}), S_n = (S'_n \cup S''_n)$	(17)
3	$\frac{S'_n = (S_{n-1} \times S'_{n-1}), S''_n = (S_{n-1} \times S''_{n-1}), S'''_n = (S_{n-1} \times S'''_{n-1}), S_n = (S'_n \cup S''_n \cup S'''_n)}{S'_n = (S_{n-1} \times S'_{n-1}), S''_n = (S_{n-1} \times S''_{n-1}), S'''_n = (S_{n-1} \times S'''_{n-1})}$,	(18)
4	$S''''_n = (S_{n-1} \times S''''_{n-1}), S_n = S'_n \cup S''_n \cup S'''_n \cup S''''_n$	(19)

Table 3. Shows properties of different type of classes.

Class type	Possible partitions($T(4,y)$)	Equation numbers	Number of classes	Size of classes	Bit positions fixed
Summand-1	4	Equation (3)	4	2^{2^n-2}	2^n, 2^n-1 (Two bits MSB and MSB-1)
		Equation (4)			$P_n - 2^k + s$, where $P_n = (2^n+1)$ and $s = 0$ for $k = 0, 1$ and $s = 1$ for $k = 2, 3, \ldots, n$
	$2+2$	Equation (5)		$2^{2^n-(n+1)}$	$P_n - 2^k$, where $P_n = (2^n+1)$ and $k = 1, 2, \ldots, n$
Summand-2		Equation (6)	2^{n+1}		2^n, 2^n-1 (Two bits MSB and MSB-1)
	$3+1$	Equation (7)			Base Case($n=1$) : $S'_1 = \{2,1\}$ and $S''_1 = \{2,1\}$
		Equation (8)		$1 \times 3^k \times 2^{2^n-2n}$	Recursion($n \geq 2$):
		Equation (9)		for $k = n-1, n-2, \ldots, 0$	$S'_n(C_{i+k}) = [S_{n-1}] + 2^{n-1}$ where $i = 1, 2, 3, 4$ and $k = 0, 4, 8, \ldots, 2^n - 4$ and
		Equation (10)			$S''_n = [[S'_n], 2^1, 2^0]$
		Equation (11)			Base Case($n=1$) : $S'_2 = \{2,1\}$, $S''_2 = \{2,1\}$, $S'''_2 = \{2,1\}$ Recursion($n \geq 2$): $S'_n(C_{i+k}) = [[S_{n-1}] + 2^{n-1}, 2^1]$, where $i = 1, 2, 3, 4$ and $k = 0, 4, 8, \ldots, 4 \times 3^{n-1} - 4$, $S''_n = [[S'_n], 2^0]$ and $S'''_n = [[S'_n], 2^0]$
Summand-3	$2+1+1$	Equation (12)	$4 \times 3^{n-1}$	$1 \times 2^k \times 2^{2^n-2n}$, for $k = n-1, n-2, \ldots, 0$	Base Case($n=1$) : $S'_1 = \{2,1\}$, $S''_1 = \{2,1\}$, $S'''_1 = \{2,1\}$ Recursion($n \geq 2$): $S'_n(C_{i+k}) = [[S_{n-1}] + 2^{n-1}, 2^0]$, where $i = 1, 2, 3, 4$ and $k = 0, 4, 8, \ldots, 4 \times 3^{n-1} - 4$, $S''_n = [[S'_n], 2^1]$ and $S'''_n = [[S'_n], 2^1]$

Continued.

Table 3. Continued.

Class type	Possible partitions($T(4,y)$)	Equation numbers	Number of classes	Size of classes	Bit positions fixed
		Equation (13)			Base Case($n=1$) : $S_2' = \{2,1\}$, $S_2'' = \{2,1\}$, $S_2''' = \{2,1\}$ Recursion($n \geq 2$): $S_n'(C_{i+k}) = [[S_{n-1}] + 2^{n-1}]$, where $i = 1,2,3,4$ and $k = 0,4,8,\ldots,4\times 3^{n-1}-4$, $S_n'' = [[S_n'], 2^1, 2^0]$ and $S_n''' = [[S_n'], 2^1, 2^0]$
		Equation (14)			Same as Equation (13)
		Equation (15)			Same as Equation (12)
		Equation (16)			Same as Equation (11)
Summand-4	$1+1+1+1$	Equation (17)	2^{2^n}	$\frac{2^{2^n}}{2^{2^n}} = 1$	Each Class contains only one Boolean Function

to different types of summands of 4 as shown in Table 1. The recurrence equations of these summands that classifies n-variable Boolean functions are shown in Table 2.

In this section, our thrust is to jot down the ideas in tables. Then we elicit these ideas and put them down as theorems in Section 3 of the paper. Also in Section 3, an attempt has been made to capture the major properties of the classes generated from Table 2 and for easy understanding the entire analysis have been summed up in Table 3.

3. Proposed method for classification of n-variable Boolean functions

In this section, n-variable Boolean functions have been classified on the basis of different summands of 1-variable Boolean functions and different properties have been studied. Classification of Boolean functions as per Equation (3) has been discussed in [9], it has been obtained that the affine functions are uniformly distributed. Classification procedure of [9] is used for other Summands (Equations (1)–(15)) and accordingly 15 other classes are obtained and various properties of those class functions are found out.

3.1 *Classification of n-variable Boolean functions on the basis of Summand-1*

Let $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ be a set of all 1-variable Boolean functions. From Equation (1), the set $S_1' = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ is a set containing all the 1-variable Boolean functions. The Cartesian product of the sets S_1 with S_1' is defined as the following:

$$S_1 \times S_1' = \begin{Bmatrix} \{0000, 0001, 0010, 0011\}, & \{0100, 0101, 0110, 0111\} \\ \{1000, 1001, 1010, 1011\}, & \{1100, 1101, 1110, 1111\} \end{Bmatrix}. \tag{20}$$

Note that, S_1 contains four classes each containing a 1-variable Boolean functions where as, the set $(S_1 \times S_1')$ contains four disjoint classes of all 2-variable Boolean functions. Here, all the classes have the same cardinality. This process is repeated for the next higher variable, using the recursive formula of (16).

3.1.1 *Different properties of the classes of summand-1*

(i) The number of classes in the above classification is four for n-variables Boolean function.
(ii) The classes are of equal size and the cardinality of each class equals to 2^{2^n-2}.
(iii) For each class of n-variable the length of a Boolean function is 2^n, out of which 2 bits are fixed and $2^n - 2$ bits are changing with respect to a Boolean function of that class.
(iv) The bit positions 2^n and 2^{n-1} are fixed for all the classes of n-variable Boolean functions.
(v) The bit positions which are fixed or changed are invariant for all classes with respect to a Boolean function of that class of n-variable.

3.2 *Classification of n-variable Boolean functions on the basis of Summand-2*

In this section, n-variable Boolean functions have been classified on the basis of different summand-2 of 1-variable Boolean functions and different properties have been studied.

3.2.1 *Classification of n-variable Boolean functions on the basis of Summand-2(2 + 2)*

Let $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ be a set of all 1-variable Boolean functions. From Equation (2), $S_1' = \{\{00\}, \{01\}\}$ and $S_1'' = \{\{10\}, \{11\}\}$ are two partitions of the set S_1'. The Cartesian product

$$S_3' = \left\{ \begin{array}{l} \mathbf{00000000}, \\ 00000001, \\ 00000100, \\ 00000101, \\ 00001000, \\ 00001001, \\ 00001100, \\ 00001101, \\ 00010000, \\ 00010001, \\ 00010100, \\ 00010101, \\ 00011000, \\ 00011001, \\ 00011100, \\ 00011101, \end{array} \right. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} Class2, \ldots, Class8 \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right\}, S_3'' = \left\{ \begin{array}{l} 00000010, \\ 00000011, \\ 00000110, \\ 00000111, \\ 00001010, \\ 00001011, \\ 00001110, \\ \mathbf{00001111}, \\ 00010010, \\ 00010011, \\ 00010110, \\ 00010111, \\ 00011010, \\ 00011011, \\ 00011110, \\ 00011111 \end{array} \right. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ Class10, \ldots, Class16 \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right\}$$

Figure 1. The naming of the classes are given as Class 1 , Class 2 , ..., Class 2^{n+1} such that the complement of Class k is the Class $(2^{n+1} - (k-1))$ where $k = 1, 2, 3, \ldots, 2^n$. In the above figure, only the members of CLASS 1 and CLASS 9 are shown.

of the sets S_1 with S_1' and S_1'' is defined successively as the following.

$$S_1 \times S_1' = \{\{\mathbf{0000}, 0001\}, \{0100, \mathbf{0101}\}, \{1000, \mathbf{1001}\}, \{\mathbf{1100}, 1101\}\} \tag{21}$$

and

$$S_1 \times S_1'' = \{\{\mathbf{0011}, 0010\}, \{, \mathbf{0110}, 0111\}, \{\mathbf{1010}, 1011\}, \{1110, \mathbf{1111}\}\}. \tag{22}$$

Note that, S_1 contains four classes each containing a 1-variable Boolean functions where as, the set $(S_1 \times S_1') \cup (S_1 \times S_1'')$ contains eight disjoint classes of all 2-variable Boolean functions. Here, each class contains exactly one 2-variable affine Boolean function as highlighted in Equations (21) and (22). This process is repeated for the next higher variable, using the recursive formula of (17) as shown in Table 2. Here both the sets S_n' and S_n'' are complement to each other.

THEOREM 3.1 *The recursive procedure of Equation* (17), *when repeated up to* $(n-1)$ *times, classify the set of all n-variable Boolean functions into* 2^{n+1} *number of disjoint classes. such that, each class contains exactly one n-variable affine Boolean function along with some n-variable nonlinear Boolean functions.*

Proof The result follows because of the fact that, $(S_{n-1} \times S_{n-1}') \cup (S_{n-1} \times S_{n-1}'') = S_{n-1} \times (S_{n-1}' \cup S_{n-1}'') = S_{n-1} \times S_{n-1} = S_n$ and $(S_{n-1} \times S_{n-1}') \cap (S_{n-1} \times S_{n-1}'') = S_{n-1} \times (S_{n-1}' \cap S_{n-1}'') = S_{n-1} \times \phi = \phi$. And the property that each class contains exactly one n-variable affine Boolean function. ■

Illustration: (from 2-variable classes to 3-variable classes)
From Equations (21) and (22) the set

$$S_2 = \left\{ \begin{array}{l} \{\mathbf{0000}, 0001\}, \{0100, \mathbf{0101}\}, \{1000, \mathbf{1001}\}, \{\mathbf{1100}, 1101\} \\ \{0010, \mathbf{0011}\}, \{\mathbf{0110}, 0111\}, \{\mathbf{1010}, 1011\}, \{1110, \mathbf{1111}\} \end{array} \right\}$$

and this set contains the classes of all 2-variable Boolean functions. The set $S_2' = \{\{\mathbf{0000}, 0001\}, \{\mathbf{0100}, 0101\}, \{1000, \mathbf{1001}\}, \{\mathbf{1100}, 1101\}\}$ is the first four classes of S_2 and $S_2'' = \{\{\mathbf{0011}, 0010\}, \{0111, \mathbf{0110}\}, \{1011, \mathbf{1010}\}, \{\mathbf{1111}, 1110\}\}$ is the set containing the remaining classes of S_2 and complement of the set S_2'. Now, the classes of 3-variables are generated using the formula as $S_3' = (S_2 \times S_2')$, $S_3'' = (S_2 \times S_2'')$ and $S_3 = (S_3' \cup S_3'')$. Some of the class members are shown in Figure 1.

THEOREM 3.2 *The number of different classes in the above classification is* 2^{n+1}.

Proof The proof of this theorem is in [9]. ■

THEOREM 3.3 *The classes are of equal size and the cardinality of each class equals to* $2^{2^n-(n+1)}$.

Proof The proof of this theorem is in [9]. ■

THEOREM 3.4 *The least significant bit of* $2^{2^n-(n+1)-1}$ *number of Boolean functions of a Class in* S'_n *and* S''_n *is* 0 *and for remaining is* 1.

Proof When $n = 1$, that is for the base case of the recursion, the least significant bit position of one Boolean function in the set S'_1 is 0 and other is 1 and for the set S''_1 it is also 0 and 1. Therefore, the recursive procedure using the Cartesian product also preserve the same property for the next higher variable. ■

Interestingly, the relation defined in the recursive procedure is operating on the set of $(n-1)$-variable Boolean functions but, the partition is obtained in the set of n-variable Boolean functions. Therefore, an equivalence relation must exist on the set of n-variable Boolean functions, which divides the set into disjoint equivalence classes.

THEOREM 3.5 *For each class of n-variable, the length of a Boolean function is* 2^n, *out of which* $(n+1)$ *bits are fixed and remaining* $(2^n-(n+1))$ *bits are changing with respect to the affine Boolean function of that class. The* $(n+1)$ *bit positions of a Boolean function which are fixed in a class are calculated using the formula;* $P_n - 2^k + s$, *where* $P_n = (2^n+1)$ *and the values of* $k = 0, 1, 2, \ldots, n$ *and* $s = 0$ *for* $k = 0, 1$ *and* $s = 1$ *for* $k = 2, 3, \ldots, n$.

(*Using Mathematical induction*) *Basis*: For $n = 1$, each class contains a single Boolean function of length 2. Hence both the first and second bit positions are fixed and it satisfies the formula $P_1 - 2^k + s = (2^1+1) - 2^k + s$ for $k = 0, 1$ and $s = 0$. So, the bit positions are $3 - 2^0 + 0 = 2$ and $3 - 2^1 + 0 = 1$. Hence the formula is valid for $n = 1$.

Induction hypothesis: Assume that, the formula is valid for the classes of $(n-1)$-variable Boolean functions; S_{n-1}. From recursive definition, the formula is also valid for all the classes of S'_{n-1} and S''_{n-1}. Thus, by induction hypothesis, the invariant bit positions of a class of S_{n-1} is calculated using the formula as given below:

$$P_{n-1} - 2^k + s, \quad \text{where } P_{n-1} = 2^{n-1} + 1 \quad \text{and} \quad k = 0, 1, 2, \ldots, n-1. \tag{23}$$

Induction: Here we have to prove that, the formula is true for all classes in S_n. According to the recursive formula $S_n = (S'_n \cup S''_n)$ where $S'_n = (S_{n-1} \times S'_{n-1})$ and $S''_n = (S_{n-1} \times S''_{n-1})$. Consider a particular class of S_{n-1} and let it be C_1. The corresponding classes of S'_n which will be generated using $(C_1 \times S'_{n-1})$, must contain the Boolean functions of length 2^n, where the first 2^{n-1} (starting from most significant bit (MSB)) bit positions are from a single class C_1. And hence by induction hypothesis, n number of bit positions are fixed and satisfies Equation (23). From Theorem 6, the second bit position of the remaining string of length 2^{n-1} is 0 for all the members of the classes of S'_n. Therefore, the bit positions of a Boolean function, which are fixed in a class of S'_n is calculated by adding 2^{n-1} to all the numbers generated from Equation (23). Along with this, we have to include the second bit position in the formula, which gives $(n+1)$ invariant positions of a class in S_n. Thus for S_n, the formula is calculated as follows:

For $k = 0, 1, 2, \ldots, n-1$,

$$\{P_{n-1} - 2^k\} + 2^{n-1} + s = \{(2^{n-1} + 1) - 2^k\} + 2^{n-1} + s = \{2^n + 1\} - 2^k + s = P_n - 2^k + s$$

for $k = n$, the value is 1:

$$2 = 1 + 1 = (2^n + 1) - 2^n + s = P_n - 2^n + s = P_n - 2^k + s.$$

So the formula is true for all the values of $k = 0, 1, 2, \ldots, n$. The above formula is also true for all the classes of S_n, as any class in S_n is either generated using the formula $(S_{n-1} \times S'_{n-1})$ or $(S_{n-1} \times S''_{n-1})$. Hence, by the principle of mathematical induction, we conclude that $P_n - 2^k + s$ is true for all positive integers n. ■

Illustration: For every 1-variable Boolean function, all the bit positions are fixed and the bit positions are $(2^1 + 1) - 2^0 + 0 = 2$ and $(2^1 + 1) - 2^1 + 0 = 1$. For every 2-variable Boolean function, three bit positions are fixed and the bit positions are $(2^2 + 1) - 2^0 + 0 = 4$, $(2^2 + 1) - 2^1 + 0 = 3$ and $(2^2 + 1) - 2^2 + 1 = 2$. Similarly, for every 3-variable Boolean function, four bit positions are fixed and the bit positions are $(2^3 + 1) - 2^0 + 0 = 8$, $(2^3 + 1) - 2^1 + 0 = 7$, $(2^3 + 1) - 2^2 + 1 = 6$ and $(2^3 + 1) - 2^3 + 1 = 2$.

The set of bit positions which are changing in a class can be calculated by subtracting the set of invariant bit positions from the set $\{1, 2, 3, \ldots, 2^n\}$.

COROLLARY 3.6 *The bit positions which are fixed or changing are invariant for all classes with respect to the concerned affine function of that class.*

Proof The formula given in Theorem 3.5 is used to calculate the bit positions which are fixed or changing and valid for an arbitrary class. Hence, it is also valid for all classes. ■

3.2.2 *Classification of n-variable Boolean functions on the basis of Summand-2*

Let $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ be a set of all 1-variable Boolean functions. From Equation (5), $S'_1 = \{\{00\}, \{11\}\}$ and $S''_1 = \{\{10\}, \{01\}\}$ be two sets containing a linear Boolean function and its complements (nonlinear) of 1-variable Boolean functions. The Cartesian product of the sets S_1 with S'_1 and S''_1 is defined successively as following.

$$S_1 \times S'_1 = \{\{\mathbf{0000}, \mathbf{0011}\}, \{0100, 0111\}, \{1000, 1011\}, \{\mathbf{1100}, \mathbf{1111}\}\} \tag{24}$$

and

$$S_1 \times S''_1 = \{\{0001, 0010\}, \{, \mathbf{0101}, \mathbf{0110}\}, \{\mathbf{1001}, \mathbf{1010}\}, \{1101, 1110\}\}. \tag{25}$$

This process is repeated for the next higher variable, using the recursive formula of (17) for classification of n-variable.

THEOREM 3.7 *The number of different classes in the above classification is* $S_n = 2^{n+1}$.

(*Using Mathematical induction*) *Basis*: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence the number of classes is $S_n = 2^{n+1}$, $S_1 = 2^{1+1} = 4$. Hence the formula is valid for $n = 1$.

Induction hypothesis: Assume that, the formula is valid for the of n-variable Boolean functions. Thus, by induction hypothesis, the number of classes of n-variable is calculated using the formula as given below:

$$S_n = 2^{n+1}. \tag{26}$$

Induction: Here we have to prove that, the formula is true for $n+1$, that is, we have to show $S_{n+1} = 2^{(n+1)+1}$. Now, to generate the number of classes of $n+1$ variable, we have to concatenate S_n' and S_n'' with S_n. So, $S_{n+1} = (S_n \times S_n') \cup (S_n \times S_n')$. Now, from induction hypothesis $S_n = 2^{n+1}$ is true for n. So $S_{n+1} = 2^{n+1} + 2^{n+1} = 2^{(n+1)+1}$ So the formula is true for all the values of $n = 1, 2, \ldots, n$. Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

THEOREM 3.8 *The classes are of equal size and the cardinality of each class equals to* $2^{2^n-(n+1)}$.

Proof The proof of this theorem is in [9]. ■

THEOREM 3.9 *For each class of n-variable, the length of a Boolean function is* 2^n, *out of which* 2 *bits are fixed and remaining* $(2^n - 2)$ *bits are changing with respect to a Boolean function of that class. The* 2 *bit positions of a Boolean function which are fixed in a class are calculated using the formula*: 2^n*and* $2^n - 1$,

Proof For $n = 1$, each class contains a single Boolean function of length 2. Hence both the first and second bit positions are fixed and it satisfies the formula 2^n and $2^n - 1$. So, the bit positions are $2^1 = 2$ and $2^1 - 1 = 1$. Similarly, for 2-variable the bit positions are $2^2 = 4$ and $2^2 - 1 = 3$. Therefore, the recursive procedure using the Cartesian product also preserve the same property for the next higher variable. ■

3.3 *Classification of n-variable Boolean functions on the Basis of summand-*$2(3+1)$

Let $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ be a set of all 1-variable Boolean functions. After partitioning the set S_1 into different subsets having cardinality 3, and 1, respectively, there are four ways as stated above in Equations (5)–(8) of Section 2 in Table 1. Let us consider Equation (5).

From Equation (7), $S_1' = \{\{00\}, \{01\}, \{10\}\}$ be a set containing three Boolean functions of 1-variable and $S_1'' = \{\{11\}\}$ contains only one Boolean function. The Cartesian product of the sets S_1 with S_1' and S_1'' is defined successively as following.

$$S_1 \times S_1' = \begin{Bmatrix} \{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \\ \{1000, 1001, 1010\}, \{1100, 1101, 1110\} \end{Bmatrix} \tag{27}$$

and

$$S_1 \times S_1'' = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}. \tag{28}$$

This process is repeated for the next higher variable, using the recursive formula of (17) for classification of n variable.

Illustration: (from 2-variable classes to 3-variable classes)

From Equations (27) and (28) the set

$$S_2 = \begin{Bmatrix} \{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \{1000, 1001, 1010\}, \\ \{1100, 1101, 1110\} \\ \{0011\}, \{0111\}, \{1011\}, \{1111\} \end{Bmatrix}$$

and this set contains the classes of all 2-variable Boolean functions.

$$S_3' = \left\{ \begin{array}{l} 00000000, 00011000, \\ 00000001, 00011001, \\ 00000010, 00011010, \\ 00000100, 00011100, \\ 00000101, 00011101, \\ 00000110, 00011110, \\ 00001000, 00100000, \\ 00001001, 00100001, \\ 00001010, 00100010, Class2, \ldots, Class8 \\ 00001100, 00100100, \\ 00001101, 00100101, \\ 00001110, 00100110, \\ 00010000, 00101000, \\ 00010001, 00101001, \\ 00010010, 00101010, \\ 00010100, 00101100, \\ 00010101, 00101101, \\ 00010110, 00101110, \end{array} \right\}, S_3'' = \left\{ \begin{array}{l} 00000011, \\ 00000111, \\ 00001011, \\ 00001111, \\ 00010011, \\ 00010111, Class10, \ldots, Class16 \\ 00011011, \\ 00011111, \\ 00100011, \\ 00100111, \\ 00101011, \\ 00101111, \end{array} \right\}$$

Figure 2. The naming of the classes is given as Class 1 , Class 2, . . . , Class 2^{n+1}. In the above figure, only the members of CLASS 1 and CLASS 9 are shown and other classes of Boolean functions are shown in Appendix 1.

The set $S_2' = \{\{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \{1000, 1001, 1010\}, \{1100, 1101, 1110\}\}$ is the first four classes of S_2 and $S_2'' = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$ is the set containing the remaining classes of S_2. Now, the classes of 3-variables are generated using the formula as $S_3' = (S_2 \times S_2')$, $S_3'' = (S_2 \times S_2'')$ and $S_3 = (S_3' \cup S_3'')$. Some of the class members are shown in Figure 2.

THEOREM 3.10 *The number of different classes in the above classification is* 2^{n+1}.

Proof (Using Mathematical induction)
The proof of this theorem is same as Theorem 3.7. ■

THEOREM 3.11 *The distinct cardinality of the classes in summand* $2(3+1)$ *are calculated using the formula*:

$$\begin{aligned} S_n &= 3^k \times 2^{2^n - 2n} \quad \text{for } k = n-1, n-2, \ldots, 0, \\ S_n' &= 3^{k+1} \times 2^{2^n - 2n} \quad \text{for } k = n-2, \ldots, 0, \\ S_n'' &= 3^k \times 2^{2^n - 2n} \quad \text{for } k = n-2, \ldots, 0. \end{aligned}$$

Proof (Using Mathematical induction)
Basis: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions. In case of summand 2, by using Equations (29) and (30), when $n = 2$ the sets are as follows, $S_2' = \{\{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \{1000, 1001, 1010\}\}$, $S_2'' = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$. From the above set S_2' has four classes each have cardinality 3. Similarly, the set S_2'' has four classes each having cardinality 1. Now, from the formula:

$$\begin{aligned} S_2 &= 3^1 \times 2^{2^2 - 2\times 2} = 3 \times 2^0 = 3 \quad \text{for } k = 1, \\ &= 3^0 \times 2^{2^2 - 2\times 2} = 1 \times 2^0 = 1 \quad \text{for } k = 0, \\ S_2' &= 3^{0+1} \times 2^{2^2 - 2\times 2} = 3 \times 2^0 = 3 \quad \text{for } k = 0, \\ S_2'' &= 3^0 \times 2^{2^2 - 2\times 2} = 1 \times 2^0 = 1 \quad \text{for } k = 0. \end{aligned}$$

Hence, the formula is true for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for n-variable Boolean functions. Thus, by induction hypothesis, the cardinality of different classes of n-variable are calculated using the formula as given below:

$$S_n = 3^k \times 2^{2^n-2n} \quad \text{for } k = n-1, n-2, \ldots, 0,$$
$$S'_n = 3^{k+1} \times 2^{2^n-2n} \quad \text{for } k = n-2, \ldots, 0,$$
$$S''_n = 3^k \times 2^{2^n-2n} \quad \text{for } k = n-2, \ldots, 0.$$

Induction: Here we have to prove that, the formula is true for $n+1$ variable. So,

$$S_{n+1} = 3^k \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-1, (n+1)-2, \ldots, 0,$$
$$S'_{n+1} = 3^{k+1} \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-2, \ldots, 0,$$
$$S''_{n+1} = 3^k \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-2, \ldots, 0.$$

In case of summand 2 $(3+1)$, for k variable the total number of Boolean functions is 2^{2^n}, after partitioning $S_n = S'_n \cup S''_n$ the cardinality of the individual set are $3 \times 2^{2^n-2}$ and 2^{2^n-2}, respectively. According to Induction Hypothesis, S'_n and S'' is valid. To generate the cardinality of S'_{n+1}, we have to concatenate S_n with S'_n, Hence, the formula:

$$\begin{aligned} S'_{n+1} &= S_n \times S'_n \\ &= 3^k \times 2^{2^n-2n} \times 3 \times 2^{2^n-2} \\ &= 3^{k+1} \times 2^{2^n-2n+2^n-2} \\ &= 3^{k+1} \times 2^{2^{n+1}-2(n+1)} \\ S''_{n+1} &= S_n \times S''_n = 3^k \times 2^{2^n-2n} \times 2^{2^n-2} \\ &= 3^k \times 2^{2^n-2n+2^n-2} \\ &= 3^k \times 2^{2^{n+1}-2(n+1)}. \end{aligned}$$

Hence, the formula is valid for S'_{n+1} and S''_{n+1}. Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

THEOREM 3.12 *For each class of n-variable, the length of a Boolean function is* 2^n. *The fixed bit position of a class are calculated using the formula*:

$$\textit{Base Case}(n=1): \; S_1 = \{2, 1\}$$
$$\textit{Recursion}(n \geq 2): \; S'_n(C_{i+k}) = [S_{n-1}] + 2^{n-1} \quad \textit{where } i = 1, 2, 3, 4$$
$$\textit{and } k = 0, 4, 8, \ldots, 2^n - 4$$
$$S''_n = [[S'_n] + 2^{n-1}, 2^1, 2^0]$$

for Equations (5)–(8).

Proof (Using Mathematical induction)

Basis: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence all bit positions are fixed for each class. By using recursion on Equations (27) and (28), when $n = 2$ the sets are as follows, $S'_2 =$

$\{\{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \{1000, 1001, 1010\}, \{1100, 1101, 1110\}\}$, and $S_2^{''} = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$. From the above set S_2' has four classes and in each class, 4 and 3 bit positions are fixed. Similarly for the set, $S_2^{''}$ has four classes where the bit positions 4, 3, 2, and 1 are fixed. Now, from the formula:

$$S_2' = [[S_1] + 2^1] = [[2, 1] + 2] = [4, 3], \quad S_2^{''} = [[S_2', 2^1, 2^0] = [4, 3, 2, 1].$$

Hence the formula is valid for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for k-variable Boolean functions. Thus, by induction hypothesis, the number of classes of k-variable is calculated using the formula as given below:

$$S_k' = [[S_{k-1}] + 2^{k-1}], \quad S_k^{''} = [[S_k'], 2^1, 2^0]. \tag{29}$$

Induction: Here we have to prove that, the formula is true for $k + 1$ variable. So, $S_{k+1}' = [[S_{k+1-1}] + 2^{k+1-1}]$, $S_{k+1}^{''} = [[S_{k+1}'], 2^1, 2^0]$. By using the method of concatenation from k to $k + 1$ variable 2^k more bits are added to the MSBs. So that, we have $S_{k+1}' = [[S_{k-1}] + 2^{k-1}] + 2^k$. Hence, $S_{k+1}' = [[S_k'] + 2^k] = [[S_{k+1-1}'] + 2^{k+1-1}]$ and the bit positions which are fixed in $S_{k+1}^{''}$ depends on S_{k+1}'. The first and second bit position is added. Since, S_{k+1}' is valid for $n = k + 1$. So, $S_{k+1}^{''}$ is also true for $n = k + 1$. So the formula is true for all the values of $k = 1, 2, \ldots, n$.

Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

3.4 *Classification of n-variable Boolean functions on the Basis of summand-*$3(2 + 1 + 1)$

Let $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ be a set of all 1-variable Boolean functions. After partitioning the set S_1 into different subsets having cardinality 2, 1, and 1, respectively, there are six ways as stated above in Equations (9)–(14) of Section 2.

From Equation (9), $S_1' = \{\{00\}, \{01\}$ be a set containing two Boolean functions of 1-variable and $S_1^{''} = \{10\}\}$, $S_1^{'''} = \{\{11\}\}$ contains only one Boolean function, respectively. The Cartesian product of the sets S_1 with S_1', $S_1^{''}$ and $S_1^{'''}$ is defined successively as following.

$$S_1 \times S_1' = \{\{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \{1100, 1101\}\} \tag{30}$$

and

$$S_1 \times S_1^{''} = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\} \tag{31}$$

$$S_1 \times S_1^{'''} = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}. \tag{32}$$

This process is repeated for the next higher variable, using the recursive formula of (18).

Illustration: (from 2-variable classes to 3-variable classes)

From Equations (30)–(32) the set

$$S_2 = \left\{ \begin{matrix} \{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \\ \{1100, 1101\} \\ \{0010\}, \{0110\}, \{1010\}, \{1110\} \\ \{0011\}, \{0111\}, \{1011\}, \{1111\} \end{matrix} \right\}$$

and this set contains the classes of all 2-variable Boolean functions. The set $S_2' = \{\{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \{1100, 1101\}\}$ is the first four classes of S_2 and $S_2^{''} = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\}$ and $S_2^{'''} = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$ is the set

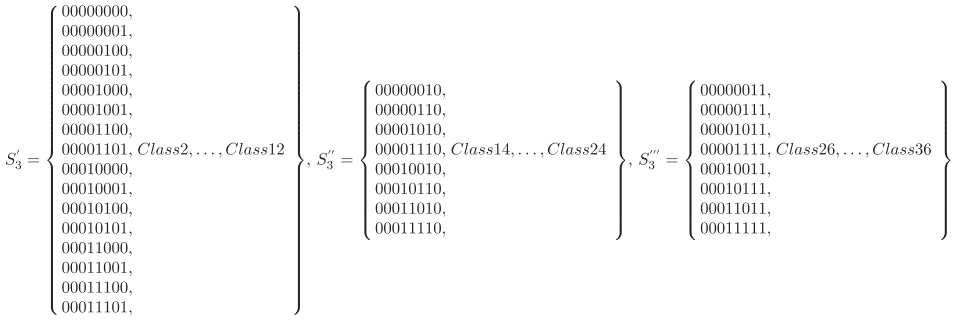

Figure 3. The naming of the classes is given as Class 1 , Class 2, . . . , Class $4 \times 3^{n-1}$. In Figure 3, only the members of CLASS 1, CLASS 12 and CLASS 24 are shown and other classes of Boolean functions are shown in Appendix 1.

containing the remaining classes of S_2. Now, the classes of 3-variables are generated using the formula as $S_3' = (S_2 \times S_2')$, $S_3'' = (S_2 \times S_2'')$, $S_3''' = (S_2 \times S_2''')$ and $S_3 = (S_3' \cup S_3'' \cup S_3'')$. Some of the class members are shown in Figure 3.

THEOREM 3.13 *The number of different classes in the above classification is* $4 \times 3^{n-1}$.

(*Using Mathematical induction*) *Basis*: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence the number of classes are $f(n) = 4 \times 3^{n-1}$, $f(1) = 4 \times 3^{1-1} = 4$. Hence the formula is valid for $n = 1$.

Induction hypothesis: Assume that, the formula is valid for k-variable Boolean functions. Thus, by induction hypothesis, the number of classes of k-variable is calculated using the formula as given below:

$$f(k) = 4 \times 3^{k-1}. \tag{33}$$

Induction: Here we have to prove that, the formula is true for $k + 1$. We have to show that $f(k + 1) = 4 \times 3^{(k+1)-1}$. Now for each partitions S_k', S_k'' and S_k''' have $4 \times 3^{(k-2)}$ number of classes. By using the method of concatenation, we know that $S_{k+1} = (S_{k+1}' \cup S_{k+1}'' \cup S_{k+1}''')$ where $S_{k+1}' = (S_k \times S_k')$, $S_{k+1}'' = (S_k \times S_k'')$, $S_{k+1}''' = (S_k \times S_k''')$. Each time S_k will generate $4 \times 3^{k-1}$ number elements. So the number of classes of $S_{k+1}' = 4 \times 3^{k-1}$. Similarly for the set S_{k+1}'' and S_{k+1}'' the number of classes is equals to $4 \times 3^{k-1}$. So, the total number of classes of

$$\begin{aligned} S_k &= 4 \times 3^{k-1} + 4 \times 3^{k-1} + 4 \times 3^{k-1} \\ &= 4 \times 3^{k-1} \times (1 + 1 + 1) = 4 \times 3^{k+1-1}. \end{aligned}$$

So the formula is true for all the values of $k = 1, 2, \ldots, n$. Hence, by the principle of mathematical induction, we conclude that $f(n)$ is true for all positive integers n. ■

THEOREM 3.14 *The distinct cardinality of the classes in summand* $3(2 + 1 + 1)$ *are calculated using the formula*:

$$\begin{aligned} S_n &= 2^k \times 2^{2^n - 2n} \quad \text{for } k = n-1, n-2, \ldots, 0, \\ S_n' &= 2^{k+1} \times 2^{2^n - 2n} \quad \text{for } k = n-2, \ldots, 0, \\ S_n'' \quad \text{and} \quad S_n''' &= 2^k \times 2^{2^n - 2n} \quad \text{for } k = n-2, \ldots, 0. \end{aligned}$$

for $n \geq 2$.

(*Using Mathematical induction*) *Basis*: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions. In case of summand 3, by using Equations (30)–(32) when $n = 2$ the sets are as follows, $S_2' = \{\{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \{1100, 1101\}\}$, $S_2'' = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\}$ and $S_2''' = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$. From the above set S_2' has four classes each have cardinality 2. Similarly, the set S_2'' and S_2''' have four classes each having cardinality 1. Now, from the formula:

$$\begin{aligned} S_2 &= 2^1 \times 2^{2^2-2\times 2} = 2 \times 2^0 = 2 \quad \text{for } k = 1, \\ &= 2^0 \times 2^{2^2-2\times 2} = 1 \times 2^0 = 1 \quad \text{for } k = 0, \\ S_2' &= 2^{0+1} \times 2^{2^2-2\times 2} = 2 \times 2^0 = 2 \quad \text{for } k = 0, \\ S_2'' \quad \text{and} \quad S_2''' &= 2^0 \times 2^{2^2-2\times 2} = 1 \times 2^0 = 1 \quad \text{for } k = 0. \end{aligned}$$

Hence, the formula is true for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for n-variable Boolean functions. Thus, by induction hypothesis, the cardinality of different classes of n-variable is calculated using the formula as given below:

$$\begin{aligned} S_n &= 2^k \times 2^{2^n-2n} \quad \text{for } k = n-1, n-2, \ldots, 0, \\ S_n' &= 2^{k+1} \times 2^{2^n-2n} \quad \text{for } k = n-2, \ldots, 0, \\ S_n'' \text{ and } S_n''' &= 2^k \times 2^{2^n-2n} \quad \text{for } k = n-2, \ldots, 0. \end{aligned}$$

Induction: Here we have to prove that, the formula is true for $n + 1$ variable. We have to show that,

$$\begin{aligned} S_{n+1} &= 2^k \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-1, (n+1)-2, \ldots, 0, \\ S_{n+1}' &= 2^{k+1} \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-2, \ldots, 0, \\ S_{n+1}'' \text{ and } S_n''' &= 2^k \times 2^{2^{n+1}-2\times(n+1)} \quad \text{for } k = (n+1)-2, \ldots, 0. \end{aligned}$$

In case of summand 3 $(2 + 1 + 1)$, for n variable the total number of Boolean functions is 2^{2^n}, after partitioning into S_n', S_n'' and S_n'' the cardinality of the individual set are 2^{2^n-1}, 2^{2^n-2} and 2^{2^n-2} respectively. According to Induction Hypothesis, S_n', S_n'' and S_n''' is valid. To generate the cardinality of S_{n+1}', we have to concatenate S_n with S_n'. Hence, the formula:

$$\begin{aligned} S_{n+1}' &= S_n \times S_n' = 2^k \times 2^{2^n-2n} \times 2 \times 2^{2^n-2} = 2^{k+1} \times 2^{2^n-2n+2^n-2} \\ &= 2^{k+1} \times 2^{2^{n+1}-2(n+1)}, \\ S_{n+1}'' &= S_n \times S_n'' = 2^k \times 2^{2^n-2n} \times 2^{2^n-2} = 2^k \times 2^{2^n-2n+2^n-2} = 2^k \times 2^{2^{n+1}-2(n+1)}, \\ S_{n+1}''' &= S_n \times S_n''' = 2^k \times 2^{2^n-2n} \times 2^{2^n-2} = 2^k \times 2^{2^n-2n+2^n-2} = 2^k \times 2^{2^{n+1}-2(n+1)}. \end{aligned}$$

$S_{n+1} = S_{n+1}' \cup S_{n+1}'' \cup S_{n+1}''$. Hence, the formula is valid for S_{n+1}', S_{n+1}'' and S_{n+1}'''. Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

THEOREM 3.15 *For each class of n-variable, the length of a Boolean function is* 2^n, *out of which the fixed bit positions of a class are calculated using the formula*:

$$\textit{Base Case}(n = 1 : \ S_1' = \{2, 1\}, \quad S_1'' = \{2, 1\} \quad \textit{and} \quad S_1''' = \{2, 1\}$$

$$\begin{aligned}
&\textit{Recursion}(n \geq 2): \; S'_n(C_{i+k}) = [S_{n-1}] + 2^1 \quad \textit{where } i = 1, 2, 3, 4\\
&\textit{and } k = 0, 4, 8, \ldots, 4 \times 3^{n-1} - 4\\
&\quad S''_n = [[S'_n], 2^0]\\
&\quad S'''_n = [[S'_n], 2^0]
\end{aligned}$$

for Equations (11) *and* (16)

(*Using Mathematical induction*) *Basis*: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence all bit positions are fixed for each class. From Equations (18) and (30)–(32), when $n = 2$ the sets are as follows, $S'_2 = \{\{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \{1100, 1101\}\}$, $S''_2 = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\}$ and $S'''_2 = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$. From the above set S'_2 has four classes and in each class 4,3,2 bit positions are fixed. Similarly, for the set S''_2 and S'''_2 have four classes each and in each class 4,3,2,1 bit positions are fixed. Now, from the formula:

$$\begin{aligned}
&S'_2 = [[S_1] + 2^1, 2^1] = [[2, 1] + 2, 2] = [4, 3, 2], S''_2 = [[S'_2, 2^0] = [4, 3, 2, 1],\\
&S''_2 = [[S'_2, 2^0] = [4, 3, 2, 1].
\end{aligned}$$

Hence the formula is valid for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for k-variable Boolean functions. Thus, by induction hypothesis, the fixed bit positions of k-variable Boolean functions are calculated using the formula as given below:

$$S'_k = [[S_{k-1}] + 2^{k-1}, 2^1], \quad S''_k = [[S'_k], 2^0] \quad \text{and} \quad S'''_k = [[S'_k], 2^0]. \tag{34}$$

Induction: Here we have to prove that, the formula is true for $k + 1$ variable. We have to show that $S'_{k+1} = [S_{k+1-1}] + 2^{k+1-1}, 2^1]$, $S''_{k+1} = [[S'_{k+1}], 2^0]$ and $S'''_{k+1} = [[S'_{k+1}], 2^0]$. By using the method of concatenation from k to $k + 1$ variable 2^k more bits are added to MSB. So that, we have $S'_{k+1} = [[S_{k-1}] + 2^{k-1}, 2^1] + 2^k$ and the 2^1 bit position is fixed through out in S'_k. Hence, $S'_{k+1} = [S'_k] + 2^k, 2^1] = [S'_{k+1-1}] + 2^{k+1-1}, 2^1]$ and the bit positions which are fixed in S''_{k+1} and S''_{k+1} is depends on S'_{k+1} and the unit bit position is added. Since, S'_{k+1} is valid for $n = k + 1$. So, S''_{k+1} and S''_{k+1} is also true for the value of $n = k + 1$. So the formula is true for all the values of $k = 1, 2, \ldots, n$.

Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

THEOREM 3.16 *For each class of n-variable, the length of a Boolean function is* 2^n*, out of which bits which are fixed in a class are calculated using the formula:*

$$\begin{aligned}
&\textit{Base Case}(n = 1): \; S'_2 = \{2, 1\}, \quad S''_2 = \{2, 1\} \quad \textit{and} \quad S'''_2 = \{2, 1\}\\
&\textit{Recursion}(n \geq 2): \; S'_n(C_{i+k}) = [S_{n-1}] + 2^1 \quad \textit{where } i = 1, 2, 3, 4\\
&\textit{and } k = 0, 4, 8, \ldots, 4 \times 3^{n-1} - 4\\
&\quad S''_n = [[S'_n], 2^1]\\
&\quad S'''_n = [[S'_n], 2^1]
\end{aligned}$$

for Equations (9) *and* (14)

(*Using Mathematical induction*) *Basis*: For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence all bit positions are fixed for each class. By using recursion on Equations (18) and (30–(32), when $n = 2$ the sets are as follows, $S_2' = \{\{0000, 0010\}, \{0100, 0110\}, \{1000, 1010\}, \{1100, 1110\}\}$, $S_2'' = \{\{0001\}, \{0101\}, \{1001\}, \{1101\}\}$ and $S_2''' = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\}$. From the above set S_2' has four classes and in each class 4, 3, 1 bit positions are fixed. Similarly, for the set S_2'' and S_2''' have four classes each and in each class 4, 3, 2, 1 bit positions are fixed. Now, from the formula:

$$S_2' = [[S_1] + 2^1, 2^0] = [[2, 1] + 2, 1] = [4, 3, 1], \quad S_2'' = [[S_2', 2^1] = [4, 3, 2, 1],$$
$$S_2'' = [[S_2', 2^1] = [4, 3, 2, 1].$$

Hence the formula is valid for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for k-variable Boolean functions. Thus, by induction hypothesis, the number of classes of k-variable is calculated using the formula as given below:

$$S_k' = [S_{k-1}] + 2^{k-1}, 2^0], S_k'' = [[S_k'], 2^1] \quad \text{and} \quad S_k''' = [[S_k'], 2^1]. \tag{35}$$

Induction: Here we have to prove that, the formula is true for $k + 1$ variable. We have to show that, $S_{k+1}' = [S_{k+1-1}] + 2^{k+1-1}, 2^0]$, $S_{k+1}'' = [[S_{k+1}'], 2^1]$ and $S_{k+1}''' = [[S_{k+1}'], 2^1]$. To generate all the Boolean functions of $k + 1$ variable by method of concatenation 2^k more bits are concatenated in MSB of k variable Boolean functions. So that, we have $S_{k+1}' = [S_{k-1}] + 2^{k-1}, 2^0] + 2^k$ and the 2^0 bit position is fixed through out in S_{k+1}'. Hence, $S_{k+1}' = [S_k'] + 2^k, 2^0] = [S_{k+1-1}'] + 2^{k+1-1}, 2^0]$ and the bit positions which are fixed in S_{k+1}'' and S_{k+1}'' depends on S_{k+1}' and the second bit position is fixed. Since, S_{k+1}' is valid for $n = k + 1$. So that, S_{k+1}'' and S_{k+1}'' is also true for the value of $n = k + 1$. So the formula is true for all the values of $k = 1, 2, \ldots, n$.

Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

THEOREM 3.17 *For each class of n-variable, the length of a Boolean function is* 2^n, *out of which bits which are fixed in a class are calculated using the formula*:

$$\textit{Base Case}(n = 2) : \; S_2' = \{4, 3\}, \quad S_2'' = \{4, 3, 2, 1\} \quad \textit{and} \quad S_2''' = \{4, 3, 2, 1\}$$
$$\textit{Recursion}(n \geq 3) : \; S_n'(C_{i+k}) = [S_{n-1}] + 2^{n-1} \quad \textit{where } i = 1, 2, 3, 4$$
$$\textit{and} \quad k = 0, 4, 8, \ldots, 4 \times 3^{n-1} - 4$$
$$S_n'' = [[S_n'], 2^1, 2^0]$$
$$S_n''' = [[S_n'], 2^1, 2^0]$$

for Equations (11) *and* (12).

Proof (Using Mathematical induction)

Basis: For$n = 1$,$S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions, each Boolean functions are treated as one Class. Hence all bit positions are fixed for each class. By using recursion on Equations (18) and (31)–(33), when $n = 2$ the sets are as follows, $S_2' = \{\{0000, 0011\}, \{0100, 0111\}, \{1000, 1011\}, \{1100, 1111\}\}$, $S_2'' = \{\{0001\}, \{0101\}, \{1001\}, \{1101\}\}$ and $S_2''' = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\}$. From the above set S_2' has four classes and

in each class 4 and 3 bit positions are fixed. Similarly, for the set $S_2^{''}$ and $S_2^{'''}$ have four classes each and in each class 4, 3, 2, 1 bit positions are fixed. Now, from the formula:

$$S_2' = [[S_1] + 2^1] = [[2,1] + 2] = [4,3], \quad S_2^{''} = [[S_2', 2^1, 2^0] = [4,3,2,1],$$
$$S_2^{''} = [[S_2', 2^1, 2^0] = [4,3,2,1].$$

Hence the formula is valid for $n = 2$.

Induction hypothesis: Assume that, the formula is valid for k-variable Boolean functions. Thus, by induction hypothesis, the number of classes of k-variable is calculated using the formula as given below:

$$S_k' = [S_{k-1}] + 2^{k-1}], \quad S_k^{''} = [[S_k'], 2^1, 2^0] \quad \text{and} \quad S_k^{'''} = [[S_k'], 2^1, 2^0]. \tag{36}$$

Induction: Here we have to prove that, the formula is true for $k+1$ variable. According to the formula $S_{k+1}' = [S_{k+1-1}] + 2^{k+1-1}]$, $S_{k+1}^{''} = [[S_{k+1}'], 2^1, 2^0]$ and $S_{k+1}^{'''} = [[S_{k+1}'], 2^1, 2^0]$. By using the method of concatenation from k to $k+1$ variable 2^k more bits are added. So that, we have $S_{k+1}' = [S_{k-1}] + 2^{k-1}] + 2^k$. Hence, $S_{k+1}' = [S_k'] + 2^k] = [S_{k+1-1}'] + 2^{k+1-1}]$ and the bit positions which are fixed in $S_{k+1}^{''}$ and $S_{k+1}^{''}$ is depends on S_{k+1}'. The first and second bit position is added. Since, S_{k+1}' is valid for $n = k+1$. So that, $S_{k+1}^{''}$ and $S_{k+1}^{''}$ is also true for the value of $n = k+1$. So the formula is true for all the values of $k = 1, 2, \ldots, n$.

Hence, by the principle of mathematical induction, we conclude that S_n is true for all positive integers n. ■

4. Different operations in classes

In this section, All distinct classes generated by Type I, Type II, Type III and Type IV are divided into several sub-classes on using the HD between the Boolean functions and by considering any Boolean function as a leader in that class. Also, the classes are analysed on performing XOR and CVT operations among the functions of a class.

Table 4. Shows different sub-classes of Class-1.

Boolean functions	Decimal value	HD wrt Affine Boolean function	No. of Boolean function
00000000	0	0	1
00000001	1		
00010000	16	1	4
00000100	4		
00001000	8		
00010001	17		
00001010	5		
00101000	20	2	6
00001100	9		
00000110	24		
00100100	12		
00010101	21		
00011001	25	3	4
00001101	13		
00011100	28		
00011101	29	4	1

4.1 *Sub-classification*

HD between two Boolean functions, denoted as HD $(f, g) = k$, where k can be $0, 1, 2, \ldots, 2^n - 2$ where f and g are any two Boolean function and both belongs to the same class of n-variable. Further, Boolean functions in a class having HD $= k$ with respect to the corresponding Boolean function in that class forms sub-classes, whose cardinality are *Binomial Coefficients* of the form $2^n - 2_{C_k}$, where $k = 0, 1, 2, \ldots, 2^n - 2$ has been discussed in [9].

Illustration: Table 4 shows the 3-variable Boolean functions belonging to Class 1 of Type III Equation (10), where the leader Boolean function is $0 = (00000000)$. There are five sub-classes having cardinality 1, 4, 6, 4, and 1 with HD 0, 1, 2, 3, 4 respectively. For 3-variables all classes and their sub-classes are given in Appendix 1.

4.2 *XOR operation in classes*

Let $a = (a_{2^n}, a_{2^n-1}, \ldots, a_1)$ and $b = (b_{2^n}, b_{2^n-1}, \ldots, b_1)$ be two n-variable Boolean functions belonging to a particular Class. The XOR operation of all the classes when arranged in a

Table 5. Shows XOR values of Class-1 of 3-variable Boolean functions.

XOR	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29
0	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29
1	1	0	5	4	9	8	13	12	17	16	21	20	25	24	29	28
4	4	5	0	1	12	13	8	9	20	21	16	17	28	29	24	25
5	5	4	1	0	13	12	9	8	21	20	17	16	29	28	25	24
8	8	9	12	13	0	1	4	5	24	25	28	29	16	17	20	21
9	9	8	13	12	1	0	5	4	25	24	29	28	17	16	21	20
12	12	13	8	9	4	5	0	1	28	29	24	25	20	21	16	17
13	13	12	9	8	5	4	1	0	29	28	25	24	21	20	17	16
16	16	17	20	21	24	25	28	29	0	1	4	5	8	9	12	13
17	17	16	21	20	25	24	29	28	1	0	5	4	9	8	13	12
20	20	21	16	17	28	29	24	25	4	5	0	1	12	13	8	9
21	21	20	17	16	29	28	25	24	5	4	1	0	13	12	9	8
24	24	25	28	29	16	17	20	21	8	9	12	13	0	1	4	5
25	25	24	29	28	17	16	21	20	9	8	13	12	1	0	5	4
28	28	29	24	25	20	21	16	17	12	13	8	9	4	5	0	1
29	29	28	25	24	21	20	17	16	13	12	9	8	5	4	1	0

Table 6. Shows CVT patterns of Class-1 of 3-variable Boolean functions.

CVT	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2
4	0	0	8	8	0	0	8	8	0	0	8	8	0	0	8	
5	0	2	8	10	0	2	8	10	0	2	8	10	0	2	8	10
8	0	0	0	0	16	16	16	16	0	0	0	0	16	16	16	16
9	0	2	0	2	16	18	16	18	0	2	0	2	16	18	16	18
12	0	0	8	8	16	16	24	24	0	0	8	8	16	16	24	24
13	0	2	8	10	16	18	24	26	0	2	8	10	16	18	24	26
16	0	0	0	0	0	0	0	0	32	32	32	32	32	32	32	32
17	0	2	0	2	0	2	0	2	32	34	32	34	32	34	32	34
20	0	0	8	8	0	0	8	8	32	32	40	40	32	32	40	40
21	0	0	8	10	0	2	8	10	32	34	40	42	32	34	40	42
24	0	0	0	0	16	16	16	16	32	32	32	32	48	48	48	48
25	0	2	0	2	16	18	16	18	32	34	32	34	48	50	48	50
28	0	0	8	8	16	16	24	24	32	32	40	40	48	48	56	56
29	0	2	8	10	16	18	24	26	32	34	40	42	48	50	56	58

table, only gives those entries given by the Class 1 functions, as $(a+k) \oplus (b+k) = (a \oplus b) + (k \oplus k) = (a \oplus b)$. Where, the XOR operation of a and b is defined as $a \oplus b = (a_{2^n} \oplus b_{2^n} a_{2^n-1} \oplus b_{2^n-1}, \ldots, a_1 \oplus b_1)$.

Illustration: Suppose we want the XOR operation of $(16)_{10} = (00010000)_2$ and $(13)_{10} = (00001101)_2$ both belong to Class 1 generated by Equation 10 of 3-variables. And $16 \oplus 13 = (00010000) \oplus (00011101) = (00011101) = 29$. Table 5 is constructed for all classes of n-variable Boolean functions that contain only the XOR values of all the functions in a class. The functions are arranged in ascending order in both rows and columns of the table. It can be proved that the content of each table remain invariant under the XOR operation and the decimal values of the content in the table are same as in Class 1. For 3-variables the XOR operation of other classes generated by Equation 10 are given in Appendix 2.

4.3 *CVT operation in classes*

Let $a = (a_k, a_{k+1}, \ldots, a_1)$ and $b = (b_k, b_{k+1}, \ldots, b_1)$ be two Boolean functions in a Class. Then the CVT of a and b is defined in [7] as $\mathrm{CVT}(a, b) = a_k \wedge b_k, a_{k-1} \wedge b_{k-1}, \ldots, a_1 \wedge b_1, 0$. CVT is a kind of representation of n-variable Boolean functions and is used to produce many interesting patterns [6]. Under the CVT operation, we have observed some interesting self similar fractal patterns which are invariant for all classes of n-variable Boolean functions.

Illustration: The CVT operation of $(16)_{10} = (00010000)_2$ and $(13)_{10} = (00001101)_2$ is 0. The patterns for Class 1 functions generated by Equation (10) using CVT operation is shown in Table 6 and others are shown in Appendix 3.

5. Conclusion

The paper begins with a discussion on the classification process emphasizing on the process and its outcome rather than its application. Here the importance of undergoing the process of classification starting with four 1-variable Boolean functions, generating n-variable Boolean functions, and how is crucial. The distinct classifications of 1-variable Boolean function gives rise to 15 different ways to classify n-variable Boolean functions having similarity with each other within a class are shown. These have been achieved by using different binary operations like HD, XOR operation and CVT operations among all the Classes. The procedures followed in this article are very handy and useful even for our future experimental research in this domain of Boolean functions. A number of tables have been incorporated in this article for easy reference and clear comprehension showing varied sub-classes, patterns and values obtained in different classes.

References

[1] G.E. Andrews, *The Theory of Partitions. Encyclopedia of mathematics and its Applications*, Addison-Wesley, London, 1976.
[2] G.E. Andrews and K. Eriksson, *Integer Partitions*, Cambridge University Press, Cambridge, 2004.
[3] J.T. Astola and R.S. Stankovic, *Fundamentals of Switching Theory and Logic Design*, Springer, Heidelberg, 2006.
[4] H. Dobbertin, *Construction of bent functions and balanced Boolean functions with high nonlinearity*, Fast Software Encryption, number 1008 in Lecture Notes in Computer Science, Springer-Verlag, 1994, pp. 61–74.
[5] X. Guo-Zhen and J. Massey, *A spectral characterization of correlation immune combining functions*, IEEE Trans. Inf. Theory 34(3) (1988), pp. 569–571.
[6] B.K Nayak, S. Sahoo, and S. Biswal, *Cellular Automata Rules and Linear Numbers*, arxiv.org/pdf /1204.3999.
[7] P. Pal Choudhury, S. Sahoo, B.K. Nayak, and S. Sarif Hassan, *Theory of carry value transformation (CVT) and its application in fractal formation*, Global J. Comput. Sci. Technol. 10(14) (2010), pp. 98–107. Version 1.0 .
[8] Ch. Posthoff and B. Steinbach, *Logic Functions and Equations*, Binary Models for Computer Science, Springer, Dordrecht, The Netherlands, 2004.

[9] R.K. Rout, P. Pal Choudhury, and S. Sahoo, *Classification of Boolean functions where affine functions are uniformly distributed*, J. Discrete Math. 2013(Article ID 270424) (2013),12 pages. doi:10.1155/2013/270424.
[10] C.E. Shannon, *A symbolic analysis of relay and switching circuits*, Trans. AIEE 57 (1938), pp. 713–723.
[11] R. Stankovic, J. Astola, and B. Steinbach, *Former and recent work in classification of switching functions*, Boolean Problems, Proceedings of the 8th International Workshops on Boolean Problems 18–19. September 2008, Freiberg University of Mining and Technology, Freiberg, 2008, pp. 115–126.
[12] B. Steinbach and C. Posthoff, *New Results for Sets of Boolean Functions*, Proceedings of the 9th International Workshops on Boolean Problems 16–17. September 2010, Freiberg University of Mining and Technology, Freiberg, 2010, pp. 21–28.
[13] B. Steinbach and Cj. Posthoff, *Logic Functions and Equations – Examples and Exercises*, Springer Science + Business Media B.V., 2009, pp 1–231.
[14] C.C. Tsai and M. Marek-Sadowska, *Boolean functions classification via fixed polarity Reed–Muller forms*, IEEE Trans. Comput. 46(2) (1997), pp. 173–186.

Appendix 1. Sub-classification

Abbreviations: BF – Boolean Function, DV – decimal value, HD – Hamming distance, No.BF – number of Boolean functions.

Table A1. Shows classes and sub-classes of 3-variable Boolean functions.

Class 1				Class 2				Class 3				Class 4			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00000000	0	0	1	**01000000**	64	0	1	**10000000**	128	0	1	**11000000**	192	0	1
00000001	1			01000001	65			10000001	129			11000001	193		
00010000	16	1	4	01010000	80	1	4	10010000	144	1	4	11010000	208	1	4
00000100	4			01000100	68			10000100	132			11000100	196		
00001000	8			01001000	72			10001000	136			11001000	200		
00010001	17			01010001	81			10010001	145			1101001	209		
00001010	5			10000010	69			11001010	133			01000010	197		
00101000	20	2	6	10001000	84	2	6	11101000	148	2	6	01101010	212	2	6
00001100	9			10101100	73			11101110	137			01101100	201		
00000110	24			10001110	88			11000110	152			01001110	216		
00100100	12			10100110	76			11100100	140			01000100	204		
00010101	21			01010101	85			10010101	149			11010101	213		
00011001	25	3	4	01011001	89	3	4	10011001	153	3	4	11011001	217	3	4
00001101	13			01001101	77			10001101	141			11001101	205		
00011100	28			01011100	92			10011100	156			11011100	220		
00011101	29	4	1	01011101	93	4	1	10011101	157	4	1	11011101	221	4	1

Class 5				Class 6				Class 7				Class 8			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00100000	32	0	1	**01100000**	96	0	1	**10100000**	160	0	1	**11100000**	224	0	1
00100001	33			01100001	97			10100001	161			11100001	225		
00100100	36	1	3	01100100	100	1	3	10100100	164	1	4	11100100	228	1	3
00101000	40			01101000	104			010101000	168			11101000	232		
00100101	37			01100101	101			10100101	168			11100101	229		
00101001	41			001101001	105			10101001	169			11101001	233		
00101100	44	2	3	01101100	108	2	3	10101100	172	2	3	11101100	236	2	3
00101101	45	3	1	01101101	109	3	1	10101101	173	3	1	11101101	237	3	1

Class 9				Class 10				Class 11				Class 12			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00110000	48	0	1	**01110000**	112	0	1	**10110000**	176	0	1	**11110000**	240	0	1
00110001	49			01110001	113			10110001	177			11110001	241		
00110100	52	1	3	01110100	116	1	3	10110100	180	1	3	11110100	244	1	3
00111000	56			01111000	120			10111000	184			11111000	248		
00110101	53			01110101	117			10110101	181			11110101	245		
00111001	57			01111001	121			10111001	185			11111001	249		
00111000	60	2	3	01111100	124	2	3	10111100	188	2	3	11111100	252	2	3
00111101	61	3	1	01111101	125	3	1	10111101	189	3	1	11111101	253	3	1

Class 13				Class 14				Class 15				Class 16			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00000010	2	0	1	**01000010**	66	0	1	**10000010**	130	0	1	**11000010**	194	0	1
00010010	18			01010010	82			10010010	146			11010010	210		
00000110	6	1	3	01000110	70	1	3	10000110	134	1	3	11000110	198	1	3
00001010	10			01001010	74			10001010	138			11001010	202		
00010110	22			01010110	86			10010110	150			11010110	214		
00011010	26			01011010	90			10011010	154			11011010	218		
00001110	14	2	3	01001110	78	2	3	10001110	142	2	3	11001110	206	2	3
00011110	30	3	1	01011110	94	3	1	10011110	158	3	1	11011110	222	3	1

Continued.

Table A1. Continued.

Class 17				Class 18				Class 19				Class 20			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00100010	34	0	1	**01100010**	98	0	1	**10100010**	162	0	1	**11100010**	226	0	1
00100110	38			01100110	102			10100110	166			11100110	230		
00101010	42	1	2	01101010	106	1	2	10101010	170	1	2	11101010	234	1	2
00101110	46	2	1	01101110	110	2	1	11101101	174	2	1	11101110	238	2	1
Class 21				Class 22				Class 23				Class 24			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00110010	50	0	1	**01110010**	114	0	1	**10110010**	178	0	1	**11110010**	242	0	1
00110110	54			01110110	118			10110110	182			11110110	246		
00111010	58	1	2	01111010	122	1	2	10111010	186	1	2	11111010	250	1	4
00111110	62	2	1	01111110	126	2	1	10111110	190	2	1	11111110	254	2	1
Class 25				Class 26				Class 27				Class 28			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00000011	3	0	1	**01000011**	67	0	1	**10000011**	131	0	1	**11000011**	195	0	1
00010011	19			01010011	83			1010011	147			11010011	211		
00000111	7	1	3	01000111	71	1	3	10000111	135	1	3	11000111	199	1	3
00001011	11			01001011	75			10001011	139			11001011	203		
00010111	23			01010111	87			10010111	151			11010111	215		
00011011	27			01011011	91			1011011	155			11011011	219		
00001111	15	2	3	01001111	79	2	3	10001111	143	2	3	11001111	207	2	3
00011111	31	3	1	01011111	95	3	1	10011111	159	3	1	011011111	223	3	1
Class 29				Class 30				Class 31				Class 32			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00100011	35	0	1	**01100011**	99	0	1	**10100011**	163	0	1	**11100011**	227	0	1
00100111	39			01100111	103			10100111	167			11100111	231		
00101011	43	1	2	001101011	107	1	2	10101011	171	1	2	11101011	235	1	2
00101111	47	2	1	01101111	111	2	1	10101111	175	2	1	11101111	239	2	1
Class 33				Class 34				Class 35				Class 36			
BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF	BF	DV	HD	No. of BF
00110011	51	0	1	**01110011**	115	0	1	**10110011**	179	0	1	**11110011**	243	0	1
00110111	55			01110111	119			10110111	183			11110111	247		
00111011	59	1	2	01111011	123	1	2	10111011	187	1	2	11111011	251	1	2
00111111	63	2	1	01111111	127	2	1	10111111	191	3	4	11111111	255	2	1

Appendix 2. XOR operations in classes

Table A2. Shows the XOR operation values of Class-1,Class-2 and Class-3 of 3-variable Boolean functions.

	Class 1																Class 2													
XOR	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29	XOR	64	65	68	69	72	73	76	77	80	81	84	85	88	89	92	93
0	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29	64	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29
1	1	0	5	4	9	8	13	12	17	16	21	20	25	24	29	28	65	1	0	5	4	9	8	13	12	17	16	21	20	25	24	29	28
4	4	5	0	1	12	13	8	9	20	21	16	17	28	29	24	25	68	4	5	0	1	12	13	8	9	20	21	16	17	28	29	24	25
5	5	4	1	0	13	12	9	8	21	20	17	16	29	28	25	24	69	5	4	1	0	13	12	9	8	21	20	17	16	29	28	25	24
8	8	9	12	13	0	1	4	5	24	25	28	29	16	17	20	21	72	8	9	12	13	0	1	4	5	24	25	28	29	16	17	20	21
9	9	8	13	12	1	0	5	4	25	24	29	28	17	16	21	20	73	9	8	13	12	1	0	5	4	25	24	29	28	17	16	21	20
12	12	13	8	9	4	5	0	1	28	29	24	25	20	21	16	17	76	12	13	8	9	4	5	0	1	28	29	24	25	20	21	16	17
13	13	12	9	8	5	4	1	0	29	28	25	24	21	20	17	16	77	13	12	9	8	5	4	1	0	29	28	25	24	21	20	17	16
16	16	17	20	21	24	25	28	29	0	1	4	5	8	9	12	13	80	16	17	20	21	24	25	28	29	0	1	4	5	8	9	12	13
17	17	16	21	20	25	24	29	28	1	0	5	4	9	8	13	12	81	17	16	21	20	25	24	29	28	1	0	5	4	9	8	13	12
20	20	21	16	17	28	29	24	25	4	5	0	1	12	13	8	9	84	20	21	16	17	28	29	24	25	4	5	0	1	12	13	8	9
21	21	20	17	16	29	28	25	24	5	4	1	0	13	12	9	8	85	21	20	17	16	29	28	25	24	5	4	1	0	13	12	9	8
24	24	25	28	29	16	17	20	21	8	9	12	13	0	1	4	5	88	24	25	28	29	16	17	20	21	8	9	12	13	0	1	4	5
25	25	24	29	28	17	16	21	20	9	8	13	12	1	0	5	4	89	25	24	29	28	17	16	21	20	9	8	13	12	1	0	5	4
28	28	29	24	25	20	21	16	17	12	13	8	9	4	5	0	1	92	28	29	24	25	20	21	16	17	12	13	8	9	4	5	0	1
29	29	28	25	24	21	20	17	16	13	12	9	8	5	4	1	0	93	29	28	25	24	21	20	17	16	13	12	9	8	5	4	1	0

	Class 5									Class 6								
XOR	32	33	36	37	40	41	44	45	XOR	96	97	100	101	104	105	108	109
32	0	1	4	5	8	9	12	13	96	0	1	4	5	8	9	12	13
33	1	0	5	4	9	8	13	12	97	1	0	5	4	9	8	13	12
36	4	5	0	1	12	13	8	9	100	4	5	0	1	12	13	8	9
37	5	4	1	0	13	12	9	8	101	5	4	1	0	13	12	9	8
40	8	9	12	13	0	1	4	5	104	8	9	12	13	0	1	4	5
41	9	8	13	12	1	0	5	4	105	9	8	13	12	1	0	5	4
44	12	13	8	9	4	5	0	1	108	12	13	8	9	4	5	0	1
45	13	12	9	8	5	4	1	0	109	13	12	9	8	5	4	1	0

	Class 17				Class 18				
XOR	34	38	42	46	XOR	98	102	106	110
34	0	4	8	12	98	0	4	8	12
38	4	0	12	8	102	4	0	12	8
42	8	12	0	4	106	8	12	0	4
46	12	8	4	0	110	12	8	4	0

Appendix 3. CVT operations in classes

Table A3. Shows the CVT patterns of Class-1, Class-2 and Class-3 of 3-variable Boolean functions.

Class 1																	Class 2															
CVT	0	1	4	5	8	9	12	13	16	17	20	21	24	25	28	29	CVT	64	65	68	69	72	73	76	77	80	81	84	85	88	89	92	93
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	64	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
1	0	2	0	2	0	2	0	2	0	2	0	2	0	2	0	2	65	128	130	128	130	128	130	128	130	128	130	128	130	128	130	128	130
4	0	0	8	8	0	0	8	8	0	0	8	8	0	0	8	8	68	128	128	136	136	128	128	136	136	128	128	136	136	128	128	136	136
5	0	2	8	10	0	2	8	10	0	2	8	10	0	2	8	10	69	128	130	136	138	128	130	136	138	128	130	136	138	128	130	136	138
8	0	0	0	0	16	16	16	16	0	0	0	0	16	16	16	16	72	128	128	128	128	144	144	144	144	128	128	128	128	144	144	144	144
9	0	2	0	2	16	18	16	18	0	2	0	2	16	18	16	18	73	128	130	128	130	144	146	144	146	128	130	128	130	144	146	144	146
12	0	0	8	8	16	16	24	24	0	0	8	8	16	16	24	24	76	128	128	136	136	144	144	152	152	128	128	136	136	144	144	152	152
13	0	2	8	10	16	18	24	26	0	2	8	10	16	18	24	26	77	128	130	136	138	144	146	152	154	128	130	136	138	144	146	152	154
16	0	0	0	0	0	0	0	0	32	32	32	32	32	32	32	32	80	128	128	128	128	128	128	128	160	160	160	160	160	160	160	160	160
17	0	2	0	2	0	2	0	2	32	34	32	34	32	34	32	34	81	128	130	128	130	128	130	128	130	160	162	160	162	160	162	160	162
20	0	0	8	8	0	0	8	8	32	32	40	40	32	32	40	40	84	128	128	136	136	128	128	136	136	160	160	168	168	160	160	168	168
21	0	0	8	10	0	2	8	10	32	34	40	42	32	34	40	42	85	128	130	136	138	128	130	136	138	160	162	168	170	160	162	168	170
24	0	0	0	0	16	16	16	16	32	32	32	32	48	48	48	48	88	128	128	128	128	144	144	144	144	160	160	160	160	176	176	176	176
25	0	2	0	2	16	18	16	18	32	34	32	34	48	50	48	50	89	128	130	128	130	144	146	144	146	160	162	160	162	176	178	176	178
28	0	0	8	8	16	16	24	24	32	32	40	40	48	48	56	56	92	128	128	136	136	144	144	152	152	160	160	168	168	176	176	184	184
29	0	2	8	10	16	18	24	26	32	34	40	42	48	50	56	58	93	128	130	136	138	144	146	152	154	160	162	168	170	176	178	184	186

(*Continued*)

Table A3. Continued.

	Class 6									Class 7								
CVT	32	33	36	37	40	41	44	45	CVT	96	97	100	101	104	105	108	109
32	64	64	64	64	64	64	64	64	96	192	192	192	192	192	192	192	192
33	64	66	64	66	64	66	64	66	97	192	194	192	194	192	194	192	194
36	64	64	72	72	64	64	72	72	100	192	192	200	200	192	192	200	200
37	64	66	72	74	64	66	72	74	101	192	194	200	202	192	194	200	202
40	64	64	64	64	80	80	80	80	104	192	192	192	192	208	208	208	208
41	64	66	64	66	80	82	80	82	105	192	194	192	194	208	210	208	210
44	64	64	72	72	80	80	88	88	108	192	192	194	194	208	208	216	216
45	64	66	72	74	80	82	88	90	109	192	194	200	202	208	210	216	218

	Class 17					Class 18				
CVT	34	38	42	46	CVT	98	102	106	110
34	68	68	68	68	98	196	196	196	196
38	68	76	68	76	102	196	204	196	204
42	68	68	84	84	106	196	196	212	212
46	68	76	84	92	110	196	204	212	220