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.*

Abstract

We present a theoretically simple and comparatively faster computing procedure for finding uniform integer partitions of x having y parts and the application in classification of Boolean functions. Existing procedures to find integer partitions of x is well accepted, but this procedure, restricted to N parts and able to generate the number of distinct partitions for a particular integer N. In this paper, the procedure has been explained and represented by a new generating function. We discuss the use of this procedure to achieve all possible equivalence classes of 1-variable Boolean functions and from these classes using recursion and Cartesian product of sets, 15 different classes of n-variable Boolean functions are obtained. The properties with regard to the size and the number of classes for different summands partition are also elaborated.

Keywords:
Integer Partition, Affine Boolean Function, Truth Table, Classification, Carry Value Transformation, XOR Operation, Hamming Distance.

*Corresponding author
Email addresses: ranjeetkumarrout@gmail.com (Ranjeet Kumar Rout), pabitrapalchoudhury@gmail.com (Pabitra Pal Choudhury), sudhakar.sahoo@gmail.com (Sudhakar Sahoo), camellia.ray@gmail.com (Camellia Ray)

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 $1+1+1+1$. Partitions have been the subject of extensive study for many years and the theory of partitions has various applications in different fields. The partitions of a positive integer are of descending compositions which are described in [1]. Then elliptic modular functions and representation theory came in [2] and [3], Gaussian polynomials in [4] and several other mathematical fields in [5]. The number of partitions of n, where the difference between parts being at least 2 is described in Gollnitz-Gordon identity [7]. The theory of partitions has further applications, for example, partitions can be used to model the possible outcomes of nuclear fission where the resulting fragments of nucleus correspond to a partition of the total number of protons and neutrons in the nucleus [6] and [17]. Partitions have also been used to develop a model of frequency fluctuations in a quartz crystal resonator [8], and several other physical applications [9] and [10]. The Rogers-Ramanujan identities in [11] is an important component in the theory of partitions, have several significant applications in physics and mathematics. Applications for partitions are also found in biology. In population genetics partitions are used to model the genetic variation of a set of gametes from a large population [12]. Classification/partitions of Boolean functions has been a long standing problem in the field of theoretical computer science. A systematic classification of Boolean functions with n-variable having an affine function as representative in each class is reported in [15].

In this paper, similar to the classification of Boolean functions as given in [15], 15 other equivalence classes of Boolean functions are generated by using the Cartesian product of the sets starting from the set of 1-variable Boolean functions. The properties of those classes are also elaborated. In the following sections, the paper is organized in the following way. In section 2, a different procedure has been proposed for integer partition. In section 3, Distinct classifications/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 4, the method of recursive classification of n-variable Boolean functions is introduced and the properties of these classes are discussed. In section 5, we have studied the

behavior of those classes by using different binary operations such as Hamming Distance(HD), XOR operation and Carry value transformation(CVT) [16]. Section 6 deals with concluding remarks emphasizing the key factors of the entire analysis. We are going to highlight the application of classification theory in classifying the proteins from their primary sequence levels.

2. Procedure to generate distinct Integer partitions

In this section, a procedure has been proposed to partition a positive integer x into different distinct parts by using a generating function.

Theorem 1. *In standard representation, a partition of x is given by a sequence $a_1, a_2, a_3, \ldots, a_y$ where $a_1 \geq a_2 \geq a_3 \geq \cdots \geq a_y$ and $a_1 + a_2 + a_3 + \cdots + a_y = x$. In this sequence a denotes the arbitrary partition and y is the number of parts of x. This can be generated by a recurrence relation as given below,*

$$S(x) = \sum_{y=1}^{x} T(x,y), \text{ where } T(x,y) = \sum_{r=(x-(y-1))}^{\lceil \frac{x}{y} \rceil} P(r + L(x-r, y-1)) \ ,$$

$y > 2$ and $L(x,y) = r + L(x-r, y-1)$, for all r, $\lceil \frac{x}{y} \rceil \leq r \leq x - (y-1)$.

$$\begin{aligned} &P(r + L(x-r, y-1)) \\ &= P[r + (x-r-(y-2)) + 1 + \cdots + 1((y-2) - times)] + \\ &P[r + (x-r-(y-1)) + 1 + 1 + \cdots + 2] + \cdots + \\ &P[r + \lceil \frac{x-r}{y-1} \rceil + L(x - r - \lceil \frac{x-r}{y-1} \rceil, y-2)]. \end{aligned}$$

$P(r_1 + r_2 + \cdots + r_n) = \frac{x C_{r_1} \times x - r_1 C_{r_2} \times \cdots \times x - r_1 - r_2 \cdots - r_{n-1} C_{r_n}}{q_1! q_2! \cdots q_p!}$,

$x = r_1 + r_2 + \cdots + r_n$, where $q_1, q_2, \ldots, q_n$ denotes the number of times, each similar elements has occurred $r_1 \geq r_2 \geq r_3 \geq \cdots \geq r_n$ and $L(x,y)$ denotes partitioning x into y parts and sum of these y parts gives x.

Initial Condition:

$$T(x,1) = 1, L(x,1) = x, L(x,x) = (1+1+1+\cdots+1(x-times)), T(x,x) = 1 \tag{1}$$

Proof. Using Mathematical Induction, we want to show $S(x)$ is valid for all values of x. To show the validity of $S(x)$ we first need to prove that $L(x,y)$

is valid for all values of x and y. The proof of the Theorem 1 is given in two parts, in the first part we have proved $L(x, y)$ using mathematical induction. In the next part the proof for $S(x)$ is given.

1. Proof of $L(x, y)$ Using Mathematical induction

   PROOF. **Basis :** For positive integer $x = 2$ and the partition of x into two parts i.e. $y = 2$ can be done in only one way which is $2 = 1 + 1$. So the recurrence relation is as follows;
   $L(2, 2) = r + L(2 - r, 2 - 1)$, $\lceil \frac{x}{y} \rceil \leq r \leq x - (y - 1)$, so $r = 1$.
   $L(2, 2) = 1 + L(2 - 1, 2 - 1)$, $L(1, 1) = 1$ and $L(2, 2) = 1 + 1$ from equation-1.
   So the formula is valid for $x = 2$, $y = 2$.
   Induction hypothesis : Assume the formula is valid for positive integer n and the partition of n into p parts i.e. $x = n$, $y = p$. Then $L(n, p) = r + L(n - r, p - 1)$.
   Induction : Here we have to prove that, the formula is valid for the positive integer $n + 1$ i.e. partition of $n + 1$ into p parts. For $x = n + 1$ and $y = p$, we need to show that; $L(n+1, p) = r+L((n+1)-r, p-1)$. Since $n > r \geq 1$ and $(n - (r - 1)) \leq n$, according to induction hypothesis, $L(n, p)$ is valid for $x = n$. So, $L((n - (r - 1)), p - 1)$ is valid for $(n - (r - 1))$, which indicates $(n - (r - 1))$ can be partitioned into $(p-1)$ parts. Now, $r+L((n-(r-1)), p-1)$ indicates $(n-(r-1))$ can be partitioned into $p - 1$ parts and r is another part of the partition. So, there are p partitions. Hence, $r + L((n - (r - 1)), p - 1)$ is valid for p partitions.

$$\begin{aligned} r + L(n - (r - 1), p - 1) &= L(n - (r - 1) + r, p) \\ &= L(n + 1, p). \end{aligned}$$

   So, for $x = n + 1$ the formula $L(n + 1, p) = r + L(n - (r - 1), p - 1) = r + L(n + 1 - r, p - 1)$. Hence, for $x = n + 1$ the formula is valid. Hence, by the principle of mathematical induction, we conclude that $L(x, y)$ is true for all positive integers n.

2. Proof of $S(x)$ Using Mathematical induction

   PROOF. **Basis :** For positive integer $x = 1$, only one summand is possible and there is only one partition for $x = 1$. Through the recurrence

relation $S(x) = \sum_{y=1}^{x} T(x,y)$, so for the positive integer $x = 1$, we have $S(1) = \sum_{y=1}^{1} T(1,1)$. Hence the statement $S(x)$ is true for $x = 1$.

Induction hypothesis : Assume that, the formula is valid for $n = x$. Then $S(x)$ is true for x and $S(x) = \sum_{y=1}^{x} T(x,y)$ is true for $1 \leq y \leq x$. $P(r_1 + r_2 + \cdots + r_n) = \frac{x_{C_{r_1}} \times x - r_1 C_{r_2} \times \cdots \times x - r_1 - r_2 \cdots - r_{n-1} C_{r_n}}{q_1! q_2! \cdots q_p!}$, is also true for $n = x$.

Induction : Here we have to show that $S(x)$ is valid for $x = n + 1$. We have to prove that recurrence relation $S(n+1) = \sum_{y=1}^{n+1} T(n+1, y)$, $1 \leq y \leq n+1$ and $T(n+1,y) = \sum_{r=((n+1)-(y-1))}^{\lceil \frac{n+1}{y} \rceil} P(r + L((n+1) - r, y-1))$, $y > 2$ is true for $n = x+1$. By induction hypothesis it has been shown in Proof 1 that $L(x,y)$ is true for all positive integer n. So $L(x,y)$ is true for $x = n+1$, as a result $P(r + L(n+1-r, y-1))$ is true for all positive integer x.

$$
\begin{aligned}
&P(r + L(n+1-r, y-1)) \\
&= P[r + (n+1-r-(y-2)) + 1 + 1 + \cdots + 1((y-2) - times)] + \\
&P[r + (n+1-r-(y-1)) + 1 + 1 + \cdots + 2] + \cdots + \\
&P[r + \lceil \frac{n+1-r}{y-1} \rceil + L(n+1-r - \lceil \frac{n+1-r}{y-1} \rceil, y-2)]
\end{aligned}
$$

and $n + 1 = r_1 + r_2 + \cdots + r_n$.
$P(r_1 + r_2 + \cdots + r_n) = \frac{n+1_{C_{r_1}} \times n+1-r_1 C_{r_2} \times \cdots \times n+1-r_1-r_2 \cdots - r_{y-1} C_{r_{y-1}}}{q_1! q_2! \cdots q_p!}$,
Hence, the statement $P(r + L(n+1-r, y-1))$ is true for all x.

$T(n,y) = \sum_{r=((n)-(y-1))}^{\lceil \frac{n}{y} \rceil} P(r + L(n-r, y-1))$, $y > 2$. For $x = n$ the statement is satisfied, for $x = n+1$, we have

$$
\begin{aligned}
T(n+1,y) = &P[(n+1-(y-1)) + L(n+1-(n+1-(y-1)), y-1)] + \\
&P[(n+1-(y)) + L(n+1-(n+1-(y)), y-1)] + \cdots + \\
&P[\lceil \frac{n+1}{y} \rceil + L(n+1 - \lceil \frac{n+1}{y} \rceil, y-1)]
\end{aligned}
\tag{2}
$$

All the terms in the right side of equation-2 is valid, so $T(n+1,y)$ is valid for $x = n+1$. Hence, $T(n+1,y) = \sum_{r=((n+1)-(y-1))}^{\lceil \frac{n+1}{y} \rceil} P(r + L((n+1)-r, y-1))$, $y > 2$ is true for all $x = n+1$. So, by the principle of mathematical induction $T(x+1,y)$ is true for all x and $1 \leq y \leq x$; $S(n+1) = \sum_{y=1}^{n+1} T(n+1,y)$. From above it has been observed that $T(x,y)$ is true for all x and $1 \leq y \leq x$. $S(n+1) = \sum_{y=1}^{n+1} T(n+1,y) = \sum_{y=1}^{n} T(n+1,y) + T(n+1,n+1)$. From mathematical induction $\sum_{y=1}^{n} T(n+1,y)$ is valid for all $x = n+1$ and from initial condition $T(n+1,n+1) = 1$ is also true. Hence $S(n+1)$ is true. Hence, by the principle of mathematical induction, we conclude that $S(x)$ is true for all positive integers n.

Illustration: For $x = 4$, the number of distinct partitions are calculated as follows,

$S(4) = \sum_{y=1}^{4} T(x,y) = T(4,1) + T(4,2) + T(4,3) + T(4,4)$. The result of the recurrences are in the following:

$$T(4,1) = 1$$

$$\begin{aligned} T(4,2) &= P(r + L(4-r, 2-1)), \ \ where \lceil \frac{4}{2} \rceil \leq r \leq 4-(2-1) \ \ i.e. \ 2 \leq r \leq 3 \\ &= P(2 + L(2,1)) + P(3 + L(1,1)) = P(2+2) + P(3+1) \\ &= \frac{4_{C_2} \times 2_{C_2}}{2!} + 4_{C_3} \times 1_{C_1} = 3 + 4 = 7 \end{aligned}$$

$$\begin{aligned} T(4,3) &= P(r + L(4-r, 3-1)), \ \ where \lceil \frac{4}{3} \rceil \leq r \leq 4-(3-1) \ \ i.e. \ 2 \leq r \leq 2 \\ &= P(2 + L(2,2)) = P(2+1+1) = \frac{4_{C_2} \times 2_{C_1} \times 1_{C_1}}{2!} = 6 \end{aligned}$$

$$T(4,4) = 1$$

So, the total number of distinct partitions is $T(4,1) + T(4,2) + T(4,3) + T(4,4) = 1 + 7 + 6 + 1 = 15$.

3. 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. By using **Theorem 1**, partitioning a positive integer $n = 4$ into positive summand and the number of such partitions of $1-$variable Boolean functions are generated by the recurrence $S(4) = \sum_{y=1}^{4} T(x, y) = T(4,1) + T(4,2) + T(4,3) + T(4,4)$ as shown below in Table 1 and the corresponding recursion equations are given in Table 2.

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	$S_1^{'} = \{\{00\}\{01\}\{10\}\{11\}\}$	(3)
			$S_1^{'} = \{\{00\}\{01\}\},\ S_1^{''} = \{\{10\}\{11\}\}$	(4)
		$2+2$	$S_1^{'} = \{\{00\}\{10\}\},\ S_1^{''} = \{\{01\}\{11\}\}$	(5)
			$S_1^{'} = \{\{00\}\{11\}\},\ S_1^{''} = \{\{01\}\{10\}\}$	(6)
Type-II	2		$S_1^{'} = \{\{00\}\{01\}\{10\}\},\ S_1^{''} = \{\{11\}\}$	(7)
		$3+1$	$S_1^{'} = \{\{00\}\{01\}\{11\}\},\ S_1^{''} = \{\{10\}\}$	(8)
			$S_1^{'} = \{\{00\}\{10\}\{11\}\},\ S_1^{''} = \{\{01\}\}$	(9)
			$S_1^{'} = \{\{01\}\{10\}\{11\}\},\ S_1^{''} = \{\{00\}\}$	(10)
			$S_1^{'} = \{\{00\}\{01\}\},\ S_1^{''} = \{\{10\}\},\ S_1^{'''} = \{\{11\}\}$	(11)
			$S_1^{'} = \{\{00\}\{10\}\},\ S_1^{''} = \{\{01\}\},\ S_1^{'''} = \{\{11\}\}$	(12)
Type-III	3	$2+1+1$	$S_1^{'} = \{\{00\}\{11\}\},\ S_1^{''} = \{\{01\}\},\ S_1^{'''} = \{\{10\}\}$	(13)
			$S_1^{'} = \{\{01\}\{10\}\},\ S_1^{''} = \{\{00\}\},\ S_1^{'''} = \{\{11\}\}$	(14)
			$S_1^{'} = \{\{01\}\{11\}\},\ S_1^{''} = \{\{10\}\},\ S_1^{'''} = \{\{11\}\}$	(15)
			$S_1^{'} = \{\{10\}\{11\}\},\ S_1^{''} = \{\{00\}\},\ S_1^{'''} = \{\{01\}\}$	(16)
Type-IV	4	1+1+1+1	$S_1^{'} = \{\{00\}\},\ S_1^{''} = \{\{01\}\},\ S_1^{'''} = \{\{10\}\},\ S_1^{''''} = \{\{10\}\}$	(17)

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

Summand	Recursion Equation	Equation No.
1	$S_n = (S_{n-1} \times S_{n-1}^{'})$	(18)
2	$S_n^{'} = (S_{n-1} \times S_{n-1}^{'})$, $S_n^{''} = (S_{n-1} \times S_{n-1}^{''})$	
	$S_n = (S_n^{'} \cup S_n^{''})$	(19)
3	$S_n^{'} = (S_{n-1} \times S_{n-1}^{'})$, $S_n^{''} = (S_{n-1} \times S_{n-1}^{''})$,	
	$S_n^{'''} = (S_{n-1} \times S_{n-1}^{'''})$	
	$S_n = (S_n^{'} \cup S_n^{''} \cup S_n^{'''})$	(20)
4	$S_n^{'} = (S_{n-1} \times S_{n-1}^{'})$, $S_n^{''} = (S_{n-1} \times S_{n-1}^{''})$,	
	$S_n^{'''} = (S_{n-1} \times S_{n-1}^{'''})$, $S_n^{''''} = (S_{n-1} \times S_{n-1}^{''''})$,	
	$S_n = S_n^{'} \cup S_n^{''} \cup S_n^{'''} \cup S_n^{''''}$	(21)

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

In this section, n-variable Boolean functions has been classified on the basis of different summands of 1-variable Boolean functions and different properties has been studied. Classification of Boolean functions as per equation 5 has been discussed in [15], it has been obtained that the affine functions are uniformly distributed. Classification procedure of [15] is used for other Summands (equation 3 to 17) and accordingly 15 other classes are obtained and various properties of those class functions are found out.

4.1. 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-**3, 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:

$$S_1 \times S_1^{'} = \left\{ \begin{array}{l} \{0000, 0001, 0010, 0011\}, \{0100, 0101, 0110, 0111\} \\ \{1000, 1001, 1010, 1011\}, \{1100, 1101, 1110, 1111\} \end{array} \right\} \quad (22)$$

Note that, S_1 contains four classes each containing a 1-variable Boolean functions where as, the set $\left(S_1 \times S_1^{'}\right)$ 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 (18).

4.1.1. 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.

4.2. 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.

4.2.1. 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-4, $S_1^{'} = \{\{00\}, \{01\}\}$ and $S_1^{''} = \{\{10\}, \{11\}\}$ are two partitioned of the set $S_1^{'}$. The Cartesian product of the sets S_1 with $S_1^{'}$ and $S_1^{''}$ is defined successively as the following.

$$S_1 \times S_1^{'} = \{\{\mathbf{0000}, 0001\}, \{0100, \mathbf{0101}\}, \{1000, \mathbf{1001}\}, \{\mathbf{1100}, 1101\}\} \quad (23)$$

and

$$S_1 \times S_1^{''} = \{\{\mathbf{0011}, 0010\}, \{, \mathbf{0110}, 0111\}, \{\mathbf{1010}, 1011\}, \{1110, \mathbf{1111}\}\} \quad (24)$$

Note that, S_1 contains four classes each containing a 1-variable Boolean functions where as, the set $\left(S_1 \times S_1^{'}\right) \cup \left(S_1 \times S_1^{''}\right)$ contains eight disjoint classes of all 2-variable Boolean functions. Here, each class contains exactly one 2-variable affine Boolean function as highlighted above in (23) and (24). This process is repeated for the next higher variable, using the recursive formula of (19) as shown in Table 2. Here both the sets $S_n^{'}$ and $S_n^{''}$ are complement to each other.

Theorem 2. *The recursive procedure of equation* (19)*, 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 non-linear 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 equation (23) and (24) the set

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

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.

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

[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.

Theorem 3. *The number of different classes in the above classification is* 2^{n+1}.

PROOF. The proof of this theorem is in [15].

Theorem 4. *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 [15].

Theorem 5. *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 6. *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* $k = 2, 3, \ldots, n$.

PROOF. (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:

$$P_{n-1} - 2^k + s, \text{ where } P_{n-1} = 2^{n-1} + 1 \text{ and } k = 0, 1, 2, \ldots, n-1. \tag{25}$$

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) bit positions are from a single class C_1. And hence by induction hypothesis, n number of bit positions are fixed and satisfies equation-(25). 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-(25). 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 $\{1, 2, 3, \ldots, 2^n\}$.

Corolarry 1. 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 6 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.

4.2.2. 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 (non-linear) of 1-variable Boolean functions. The Cartesian product of the sets S_1 with $S_1^{'}$ and $S_1^{''}$ is defined successively as following.

$$S_1 \times S_1^{'} = \{\{\mathbf{0000}, \mathbf{0011}\}, \{0100, 0111\}, \{1000, 1011\}, \{\mathbf{1100}, \mathbf{1111}\}\} \quad (26)$$

and

$$S_1 \times S_1^{''} = \{\{0001, 0010\}, \{, \mathbf{0101}, \mathbf{0110}\}, \{\mathbf{1001}, \mathbf{1010}\}, \{1101, 1110\}\} \quad (27)$$

This process is repeated for the next higher variable, using the recursive formula of (19) for classification of n-variable.

Theorem 7. *The number of different classes in the above classification is* $S_n = 2^{n+1}$.

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 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:

$$S_n = 2^{n+1} \quad (28)$$

Induction : Here we have to prove that, the formula is true for $n+1$. i.e. 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 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 [15].

Theorem 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.

4.3. 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 $7, 8, 9$ *and* 10 of Section 2 in Table 1. Let us consider the *Equation* $- 7$.

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.

$$S_1 \times S_1^{'} = \left\{ \begin{array}{l} \{0000, 0001, 0010\}, \{0100, 0101, 0110\}, \\ \{1000, 1001, 1010\}, \{1100, 1101, 1110\} \end{array} \right\} \tag{29}$$

and

$$S_1 \times S_1^{''} = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\} \tag{30}$$

This process is repeated for the next higher variable, using the recursive formula of (20) for classification of n variable.

Illustration: **(from 2-variable classes to 3-variable classes)** From equation (29) and (30) the set

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

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

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

[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-A**].

Theorem 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 7.

Theorem 11. *The distinct cardinality of the classes in summand 2(3+1)*

are calculated using the formula:

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

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 equation 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, for the set $S_2^{''}$ has four classes each having cardinality 1. Now, from the formula:

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

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:

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

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

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

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

$$\begin{aligned}
&BaseCase(n=1): \ S_1 = \{2,1\} \\
&Recursion(n \geq 2): \ S_n^{'}(C_{i+k}) = [S_{n-1}] + 2^{n-1} \ where \ i = 1,2,3,4 \\
&and \ k = 0,4,8,\ldots,2^n-4 \\
&\qquad\qquad S_n^{''} = [[S_n^{'}] + 2^{n-1}, 2^1, 2^0]
\end{aligned}$$

for the equation $7, 8, 9$ *and* 10.

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 equation 29 and equation 30, 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:

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

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:

$$S_k^{'} = [S_{k-1}] + 2^{k-1}], S_k^{''} = [[S_k^{'}], 2^1, 2^0] \tag{31}$$

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 MSB(Most significant bits). 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 $S_{k+1}^{''}$ 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, $S_{k+1}^{''}$ 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.

4.4. 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 6 ways as stated above in equation $11, 12, 13, 14, 15$ *and* 16 of Section 2.

From equation 11, $S_1^{'} = \{\{00\}, \{01\}$ 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.

$$S_1 \times S_1^{'} = \{\{0000, 0001\}, \{0100, 0101\}, \{1000, 1001\}, \{1100, 1101\}\} \tag{32}$$

and

$$S_1 \times S_1^{''} = \{\{0010\}, \{0110\}, \{1010\}, \{1110\}\} \tag{33}$$

$$S_1 \times S_1^{'''} = \{\{0011\}, \{0111\}, \{1011\}, \{1111\}\} \tag{34}$$

This process is repeated for the next higher variable, using the recursive formula of (20).

Illustration: **(from 2-variable classes to 3-variable classes)** From equation (32), (33) and (34) the set

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

$$S_3' = \left\{ \begin{array}{l} 00000000, \\ 00000001, \\ 00000100, \\ 00000101, \\ 00001000, \\ 00001001, \\ 00001100, \\ 00001101, \quad Class2, \ldots, Class12 \\ 00010000, \\ 00010001, \\ 00010100, \\ 00010101, \\ 00011000, \\ 00011001, \\ 00011100, \\ 00011101, \end{array} \right\}, S_3'' = \left\{ \begin{array}{l} 00000010, \\ 00000110, \\ 00001010, \\ 00001110, \quad Class14, \ldots, Class24 \\ 00010010, \\ 00010110, \\ 00011010, \\ 00011110, \end{array} \right\}, S_3''' = \left\{ \begin{array}{l} 00000011, \\ 00000111, \\ 00001011, \\ 00001111, \quad Class26, \ldots, Class36 \\ 00010011, \\ 00010111, \\ 00011011, \\ 00011111, \end{array} \right\}$$

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

Theorem 13. *The number of different classes in the above classification is* $4 \times 3^{n-1}$.

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 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 the of $k-$variable Boolean functions. Thus, by induction hypothesis, the number of classes of $k-$variable is calculated using the formula as given below:

$$f(k) = 4 \times 3^{k-1} \tag{35}$$

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 $4 \times 3^{k-2}$. 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 14. *The distinct cardinality of the classes in summand 3(2+1+1) are calculated using the formula:*

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

for $n \geq 2$.

Proof. (Using Mathematical induction)

Basis : For $n = 1$, $S_1 = \{\{00\}, \{01\}, \{10\}, \{11\}\}$ contain four Boolean functions. In case of summand 3, by using equation (32), (33) and (34) 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, for the set $S_2^{''}$ and $S_2^{'''}$ have four classes each having cardinality 1. Now, from the formula:

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

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:

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

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

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

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 15. *For each class of n-variable, the length of a Boolean function is 2^n, out of which the fixed bit possitions of a class are calculated using the*

formula:

$$BaseCase(n=1): \ S_1^{'} = \{2,1\}, \ S_1^{''} = \{2,1\} \ and \ S_1^{'''} = \{2,1\}$$
$$Recursion(n \geq 2): \ S_n^{'}(C_{i+k}) = [S_{n-1}] + 2^1 \ where \ i = 1,2,3,4$$
$$and \ k = 0,4,8,\ldots,4 \times 3^{n-1} - 4$$
$$S_n^{''} = [[S_n^{'}], 2^0]$$
$$S_n^{'''} = [[S_n^{'}], 2^0]$$

for equation-11 *and equation*-16

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. From equation 20 and equation 32, 33, 34, 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:

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

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

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(most significant bit). 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 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:*

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

for equation-12 *and equation*-15

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 equation 20 and equation 32, 33, 34,, 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:

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

$$S_k^{'}=[S_{k-1}]+2^{k-1},2^0], S_k^{''}=[[S_k^{'}],2^1] and S_k^{'''}=[[S_k^{'}],2^1] \tag{37}$$

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(Most significant bit) 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 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:*

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

for the equation 13 *and* 14.

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 equation 20 and equation 32, 33, 34, 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:

$S^{'}_2 = [[S_1] + 2^1] = [[2, 1] + 2] = [4, 3]$, $S^{''}_2 = [[S^{'}_2, 2^1, 2^0] = [4, 3, 2, 1]$, $S^{''}_2 = [[S^{'}_2, 2^1, 2^0] = [4, 3, 2, 1]$. 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:

$$S^{'}_k = [S_{k-1}] + 2^{k-1}], S^{''}_k = [[S^{'}_k], 2^1, 2^0] and S^{'''}_k = [[S^{'}_k], 2^1, 2^0] \qquad (38)$$

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.

5. 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 Hamming Distance (HD) between the Boolean functions and by considering any Boolean function as a leader in that class. Also, the classes are analyzed on performing XOR and CVT operations among the functions of a class.

5.1. Sub-classification

Hamming Distance(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 [15].

Illustration: Table 3 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 hamming distance (HD) 0, 1, 2, 3, 4 respectively. For 3-variables all classes and their sub-classes are given in **Appendix A**.

5.2. 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, only gives those entries given by the

Table 3: 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

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 4 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 B**.

5.3. 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 Carry Value Transform (CVT) of a and b is defined in [10] as $CVT\,(a,b) = a_k \wedge b_k, a_{k-1} \wedge b_{k-1}, \ldots, a_1 \wedge b_1, 0$. Carry Value Transformation (CVT) is a kind of representation of n-variable Boolean functions and is used to produce many interesting patterns [10]. 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

Table 4: 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

10 using CVT operation is shown below in Table 5 and others are shown in **Appendix C**.

Table 5: 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	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

6. Conclusion

In this paper, we have considered a novel approach for generating distinct Integer partition in a precise way. Classification of Boolean functions on using integer partition of 1-variable gives rise to similar properties. This we have achived by using different binary operations like Hamming Distance, 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 theoretical computer science. A number of tables have been incorporated in this article for easy reference and clear comprehension showing varied sub-classes, patterns and values of different classes.

Table 6: 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)
Summand-2	$2+2$	Equation-4	2^{n+1}	$2^{2^n-(n+1)}$	$P_n - 2^k + s$, *where* $P_n = (2^n+1)$ and $s = 0$ *for* $k = 0, 1$ *and* $s = 1$ for $k = 2, 3, \ldots, n$
		Equation-5			$P_n - 2^k$, *where* $P_n = (2^n+1)$ and $k = 1, 2, \ldots, n$
		Equation-6			2^n, 2^n-1 (Two bits MSB and MSB-1)
	$3+1$	Equation-7 Equation-8 Equation-9 Equation-10		$1 \times 3^k \times 2^{2^n-2n}$ for $k = n-1, n-2, \ldots, 0$	Base Case(n=1) : $S_1' = \{2,1\}$ and $S_1'' = \{2,1\}$ Recursion(n$\geq$ 2) : $S_n'(C_{i+k}) = [S_{n-1}] + 2^{n-1}$ where $i = 1, 2, 3, 4$ and $k = 0, 4, 8, \ldots, 2^n - 4$ and $S_n'' = [[S_n'], 2^1, 2^0]$
Summand-3	$2+1+1$	Equation-11	$4 \times 3^{n-1}$	$1 \times 2^k \times 2^{2^n-2n}$ for $k = n-1, n-2, \ldots, 0$	Base Case(n=1) : $S_2' = \{2,1\}$, $S_2'' = \{2,1\}$,$S_2''' = \{2,1\}$ 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 $S_n''' = [[S_n'], 2^0]$
		Equation-12			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 $S_n''' = [[S_n'], 2^1]$
		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.

Appendix A. Sub-Classification

Abbreviations: BF-Boolean Function, DV-Decimal Value, HD-Hamming Distance, No.BF-Number of Boolean Functions.

Table A.7: 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
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 B. XOR Operations in Classes

Table B.8: Shows the XOR operation values of Class-1,Class-2 and Class-3 of 3-variable Boolean Functions

Class 1																
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

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

Class 5								
XOR	**32**	**33**	**36**	**37**	**40**	**41**	**44**	**45**
32	0	1	4	5	8	9	12	13
33	1	0	5	4	9	8	13	12
36	4	5	0	1	12	13	8	9
37	5	4	1	0	13	12	9	8
40	8	9	12	13	0	1	4	5
41	9	8	13	12	1	0	5	4
44	12	13	8	9	4	5	0	1
45	13	12	9	8	5	4	1	0

Class 6								
XOR	**96**	**97**	**100**	**101**	**104**	**105**	**108**	**109**
96	0	1	4	5	8	9	12	13
97	1	0	5	4	9	8	13	12
100	4	5	0	1	12	13	8	9
101	5	4	1	0	13	12	9	8
104	8	9	12	13	0	1	4	5
105	9	8	13	12	1	0	5	4
108	12	13	8	9	4	5	0	1
109	13	12	9	8	5	4	1	0

Class 17				
XOR	**34**	**38**	**42**	**46**
34	0	4	8	12
38	4	0	12	8
42	8	12	0	4
46	12	8	4	0

Class 18				
XOR	**98**	**102**	**106**	**110**
98	0	4	8	12
102	4	0	12	8
106	8	12	0	4
110	12	8	4	0

Appendix C. CVT Operations in Classes

Table C.9: Shows the CVT(Carry Value Transformation) patterns of Class-1, Class-2 and Class-3 of 3-variable Boolean Functions

Class 1

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	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

Class 2

CVT	64	65	68	69	72	73	76	77	80	81	84	85	88	89	92	93
64	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128	128
65	128	130	128	130	128	130	128	130	128	130	128	130	128	130	128	130
68	128	128	136	136	128	128	136	136	128	128	136	136	128	128	136	136
69	128	130	136	138	128	130	136	138	128	130	136	138	128	130	136	138
72	128	128	128	128	144	144	144	144	128	128	128	128	144	144	144	144
73	128	130	128	130	144	146	144	146	128	130	128	130	144	146	144	146
76	128	128	136	136	144	144	152	152	128	128	136	136	144	144	152	152
77	128	130	136	138	144	146	152	154	128	130	136	138	144	146	152	154
80	128	128	128	128	128	128	128	160	160	160	160	160	160	160	160	160
81	128	130	128	130	128	130	128	130	160	162	160	162	160	162	160	162
84	128	128	136	136	128	128	136	136	160	160	168	168	160	160	168	168
85	128	130	136	138	128	130	136	138	160	162	168	170	160	162	168	170
88	128	128	128	128	144	144	144	144	160	160	160	160	176	176	176	176
89	128	130	128	130	144	146	144	146	160	162	160	162	176	178	176	178
92	128	128	136	136	144	144	152	152	160	160	168	168	176	176	184	184
93	128	130	136	138	144	146	152	154	160	162	168	170	176	178	184	186

Class 6

CVT	32	33	36	37	40	41	44	45
32	64	64	64	64	64	64	64	64
33	64	66	64	66	64	66	64	66
36	64	64	72	72	64	64	72	72
37	64	66	72	74	64	66	72	74
40	64	64	64	64	80	80	80	80
41	64	66	64	66	80	82	80	82
44	64	64	72	72	80	80	88	88
45	64	66	72	74	80	82	88	90

Class 7

CVT	96	97	100	101	104	105	108	109
96	192	192	192	192	192	192	192	192
97	192	194	192	194	192	194	192	194
100	192	192	200	200	192	192	200	200
101	192	194	200	202	192	194	200	202
104	192	192	192	192	208	208	208	208
105	192	194	192	194	208	210	208	210
108	192	192	194	194	208	208	216	216
109	192	194	200	202	208	210	216	218

Class 17

CVT	34	38	42	46
34	68	68	68	68
38	68	76	68	76
42	68	68	84	84
46	68	76	84	92

Class 18

CVT	98	102	106	110
98	196	196	196	196
102	196	204	196	204
106	196	196	212	212
110	196	204	212	220

References

[1] George E. Andrews. *The Theory of Partitions. Encyclopedia of mathematics and its Applications*. Addison-Wesley, London, 1976.

[2] Manfred R. Schroeder. *Number Theory in Science and Communication with Applications in Cryptography*, Physics, Digital Information, Computing, and Self-Similarity. Springer-Verlag, Berlin, second enlarged edition, 1986.

[3] Stuart Martin. *Schur Algebras and Representation Theory*. Cambridge University Press, 1999.

[4] George E. Andrews and Kimmo Eriksson. *Integer Partitions*. Cambridge University Press, 2004.

[5] Scott Ahlgren and Ken Ono.*Addition and counting: The arithmetic of partitions*. Notices of the AMS, 48(9):978-984, October 2001.

[6] P. Desesquelles. *Calculation of the number of partitions with constraints on the fragment size*. Physical Review C (Nuclear Physics), 65(3):034603 March 2002.

[7] H. L. Alder. *Partition identities from Euler to the present*. The American Mathematical Monthly, 76(7):733-746, August- September 1969.

[8] Steven S. Skiena. *The Algorithm Design Manual*. TELOS - the Electronic Library of Science, Santa Clara, California, 1998.

[9] Michel Planat. *Thermal 1/f noise from the theory of partitions: application to a quartz resonator*. Physica A: Statistical Mechanics and its Applications, 318(3-4):371-386, February 2003.

[10] H. N. V. Temperley. *Statistical mechanics and the partition of numbers*. I. The transition of liquid helium. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 199(1058):361-375, November 1949.

[11] Alfred Arthur Actor. *Infinite products, partition functions, and the Meinardus theorem*. Journal of Mathematical Physics, 35(11):5749-5764, November 1994.

[12] Anders Bjoner and Richard P. Stanley. *A combinatorial miscellany*

[13] J. F. C. Kingman. *Random partitions in population genetics*. Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences, 361(1704):1-20, May 1978.

[14] B. K Nayak, S. Sahoo, S. Biswal, "*Cellular Automata Rules and Linear Numbers*", arxiv.org/pdf /1204.3999.

[15] R. K Rout, P. Pal Choudhury, S. Sahoo,(2013) "*Classification of Boolean Functions Where Affine Functions Are Uniformly Distributed*", Journal of Discrete Mathematics, vol. 2013, Article ID 270424, 12 pages, 2013. doi:10.1155/2013/270424.

[16] P. Pal Choudhury, S. Sahoo, B. K Nayak, Sk. Sarif Hassan(2010), "*Theory of Carry Value Transformation (CVT) and its Application in Fractal formation*", Global Journal of Computer Science and Technology, Vol. 10, Issue 14, Version 1.0, pp. 98-107.

[17] S. WolForm(2002), "*A New Kind of Science*", Wolfram Media, Inc., 2002.